Santa Clara University
DEPARTMENT of COMPUTER ENGINEERING

Date: June 6, 2003

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

David Cuccias, Nick Foster, and Matt Strathman

ENTITLED

Wirelessly Controlled Power Outlets

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER ENGINEERING

____
THESIS ADVISOR

____
DEPARTMENTCHAIR

Wireless Controlled Power Outlets

by

David Cuccias, Nick Foster, and Matt Strathman

SENIOR DESIGN PROJECT REPORT

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Engineering
School of Engineering
Santa Clara University

Santa Clara, California

June 6, 2003

Wirelessly Controlled Power Outlets

David Cuccias, Nick Foster, and Matt Strathman

Department of Computer Engineering
Santa Clara University
2003

ABSTRACT

Currently home and business owners are limited to manipulating power outlets with hard wired switches and are blind to where power is consumed in a building. In attempt to solve this problem we created a system using advanced wireless and circuit technology to allow the user to manipulate his or her wall outlets via Intranet and WAP capable cell phones as well as viewing power consumption totals to monitor individual outlets. We were able to produce a system to accomplish this goal with a control box that communicates to circuitry implanted in each outlet and accessible via a web site hosted on the control box. The implementation what we used was not necessarily the best or most cost effective, for mass production other considerations would need to be considered in order to make the product more viable.

Keywords: Wireless Communication, Power Regulation, Embedded Circuitry, COM Port Communication, ODBC, Software UART, Firmware Development

Senior Design:

Outlet Power Regulation

Senior Design: Acknowledgements

We would like to thank Neil Quinn, our professor, for his guidance and help throughout this entire process. We wouldn't have completed it on time without his extensive knowledge and helpful attitude.

We would also like to thank Michael Strathman for his help in designing the circuitry for the wall unit. Without his knowledge and previous experience our project would have never gotten off the ground.

Senior Design:

Table of Contents

Senior Design:

Table of Figures

Senior Design:
Introduction

Proximity limits the amount of control people have over their wall switches and the appliances attached to them. We developed a system that controls and monitors the power flowing through wall outlets allowing users to have complete control over their homes and businesses no matter where in the world they are. Additionally, our system will show the power usage history of each individual outlet allowing the user to better understand specifics of power usage in their home or business. The system is able to control all wall outlets in a building and actively communicate with them gaining information and storing it in a database for later retrieval. The main goal of the project was to combine as many disciplines that we have learned from Santa Clara University together and learn and include others as well.

Similar systems are already on the market; however they don't include everything ours has to offer. These systems use X-10 technology which uses existing power lines to send bits of information slowly. These systems do not offer power regulation like our system would provide, and they do not gain power information to be stored for later reference. Our main goal behind this project was not to build a better competing system to be placed on the market but to learn and use as many technologies as possible.

To complete these goals the following steps were completed to produce our system.

Embedded Circuitry: A circuit was designed capable of dropping the input voltage from the "hot" wire of a power outlet down to a measurable level. Measurements taken are proportional to the actual value, in order to be converted back to a useable value when transmitted to the control box. This circuitry had to be integrated with a microcontroller capable of communicating with the control box to provide a programmable interface with the circuitry and a link between the base station and the wall units. All components purchased for power control and sampling had to be integrated together, including a transceiver, a microcontroller, an optically isolated TRIAC for each outlet, and each of the sampling networks to completely implement all necessary functionality of the embedded component.

Firmware for Wall Units: The hardware cannot control itself and needed software running on the chip to perform specified functions. This software is called the firmware for the embedded circuitry. The firmware for the wall units was created so that after receiving a command it performs the appropriate task. This includes all UART communication and PIN control on the chip. This software was developed using the Cygnal development kit that was purchased with the Cygnal chip that was used.

Wireless Communication: The design decision was made to use wireless communication between the wall units and the control box to gain a better understanding and practical usage of this new technology. We preformed extensive research on wireless communication so that we could learn how to harness the technology. Once research was completed we decided to use the TR1000 wireless transceiver because of its development kit and the simplicity of how to connect it to the rest of the embedded circuitry. After installing the device we had to start to take baby steps in understanding exactly how to send bytes of information from one transceiver to another.

Wireless Packet Protocols: we had to create protocols for ending all the wall unit information and command structure. We had to take into account some wireless limitations as well as limit the number of bits being sent across the airwaves. We developed a simple set of command structures including opcodes and check sums to ensure that all the data reached its destination correctly.

Building Control Box Hardware: Before any work could start on the control box we had to choose the hardware and build it. The hardware was chosen to both optimize the performance and limit the cost. We decided on a basic Athalon chipset with 512 Mb of RAM and a 20 Gb hard drive. We also needed to choose software to perform low level program handling tasks to allow the many separate parts of the software to communicate. We choose to use Microsoft Windows 2000 Advanced Server because of its capabilities and our prior knowledge setting it up and using it.

Creating Database Tables: System information has to be stored in a safe and protected way, a database provides this function. The specifics of the database had to be created including all tables, relationships and variables. Power consumption totals has to be stored for each outlet so that users can learn about power usage for the system. Other system information includes name, location, and address for each wall unit and event sequence. Microsoft Access was chosen to meet this requirement because of the simple connectivity to the Microsoft ODBC drivers used with Windows 2000 Server.

Designing Program to Interface with Database and Serial Communication: A program had to be created to provide the main functionality of the control box. The program has to interface with the database and relay all commands to the serial port so that the wireless transceiver can send them out the wall outlets. In addition the program had to keep track of the time to know when to send commands. This program was developed using Rhapsody 4.0 C++ UML development kit. It allowed for simple tread creation and auto generation of code from graphical UML drawings.

Creating User Interfaces: Design the dynamic website and WAP interface to allow the user to view and manage the system. These interfaces had to be simple enough for even the most non-technical user to use hassle-free. The dynamic webpage was created using Microsoft .NET software which allowed for easy on the fly generated websites. The WAP WML cell phone website was created using a trial version of MobileDev studio which allowed a graphical implementation of WML code. Both systems interface with the database to provide on the fly generated sites for the user to view and manipulate the system.

Senior Design:
Design Overview

1. Overview

As we develop technology to make, what used to be impossible, tasks simpler, we in turn make our lives more complicated. There used to be a time where all we had to remember was when to feed the stock animals and when to plow the fields to survive. However as we have moved away from the simple fields and into a down town metropolis our lives have been over whelmed with responsibilities which may seem simple but by the mere magnitude of them they become dizzying to any person. Tasks like turning off lights as one leave a room, or waiting to use the washer and dryer until after 7pm to conserve power are things that many don't have time to remember let alone complete. These easy tasks can actually cost people lots of money if high power appliances are left on while owners are on vacations and some can actually cause disaster. In these complicated times there is a way to help people handle all the burdens that technology puts on us, or at least some of them. Our project does exactly this, by allowing users to manage the power using appliances in their house.

To solve the problem we propose a system to help users take charge of their appliances in their household. We propose a system that takes charge of a users home to allow the user to relax and let computers handle these tasks. The system needs to be compiled of three major parts, a browser on the users home PC to interface the system, a central control box to manipulate the system, and all the devices implanted into the wall sockets. These devices allow power to flow through them when turned on by the main control and restrict power when turned off, main control can also tell the devices to regulate the power flowing across. The main control box needs to be connected to both of the other components so that it function correctly. It needs to be able to receive information from the devices, send commands to the devices and receive information from the home users PC. The control box must be able to keep track of each device and store information about the device including hardware address, device pass code (so device knows its talking to central box), users given name for outlet, location of device given by user, current total of power flowed across. Main control also must keep track of user events that manipulate any number of devices, and main control must be able to complete an event's tasks when the start time occurs. Each event will have a start time and a list of devices and what to do with them. These events are created by the user and can be edited from the home PC at the users whim.

The devices must be small enough to fit into a wall socket and be able to pass the current fire safety code. Each device must be connected to the main control so that it can receive any orders and be able to send its current power flow information when asked. The device must be able to calculate the amount of power that is flowing across the outlet when polled and send it back. The devices must be able to know that they are communicating with the main control and not some other computer. The devices must be within 100ft of the central box so that they can be assured of talking to control box and not another computer. The device needs to be able to restrict power flowing across the outlet so it can comply with the instructions of main control and the user. The device needs to be inexpensive because of the number of devices needed will make the price of the entire system enormous.

The software that will run on the users home PC must be easy to use and understand. It must give the user an easy way to interface the network of devices in the outlets. The PC must be connected to the control box so that information can be passed back and forth. The software must allow capabilities to edit, delete, or create new schemes, turn on or off any device in the house, setup up the house graphically, change the attributes of any of the devices and change any user information.

Senior Design:
User Manual

1. SYSTEM REQUIREMENTS

Once the wall units have been installed and the server is up and running use any type of web browser to access your system. Accessing your system can be broken down into two categories; Managing and Viewing your system.

2. MANAGE SYSTEM BY WALL UNITS

There are two main categories of system management; by Wall Unit and by Event Sequence. The features for both are the same but have different requests.

2.1 ADD WALL UNIT

By clicking on the button in the left-hand side menu that says “Add Wall Unit” you send a command for the system to search through every wall unit present.

WARNING

This may take some time and your system will be temporarily out of service during this search.

Once the search has completed the wall units that were found will appear with input boxes. These input boxes are to allow you to name your wall units and give them a location. These attributes will make it easier in the future to know which wall unit you are managing and viewing.

See Flow Chart below for the web pages in sequence.

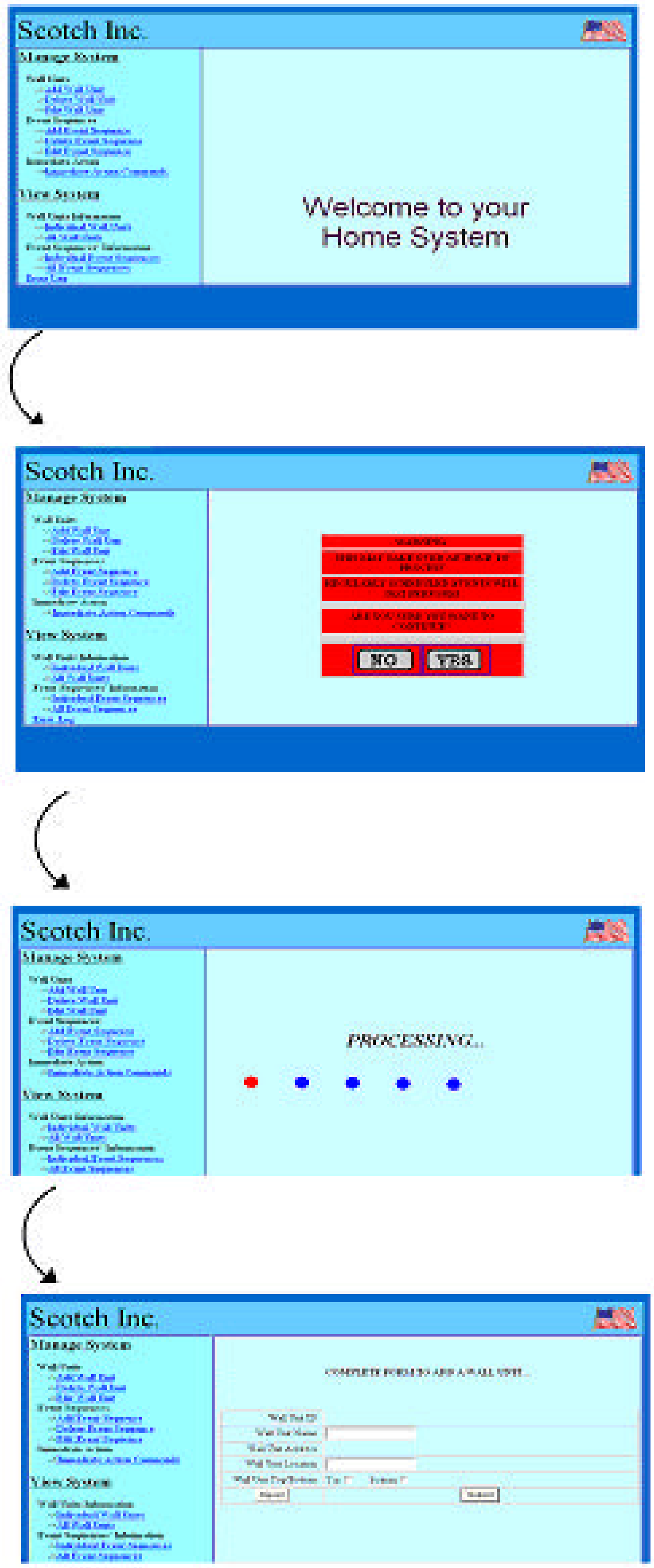


Figure 1: Add Wall Unit

2.2 DELETE WALL UNIT

By clicking the button in the left-hand side menu that says “Delete Wall Unit” you go to a table of the attributes of all the wall units that are in your system. On the right side of the table there is a column titled “Delete?”

To delete a wall unit simply click on the “Delete” button in the “Delete?” column of the desired wall unit.

See Flow Chart below for the web pages in sequence.

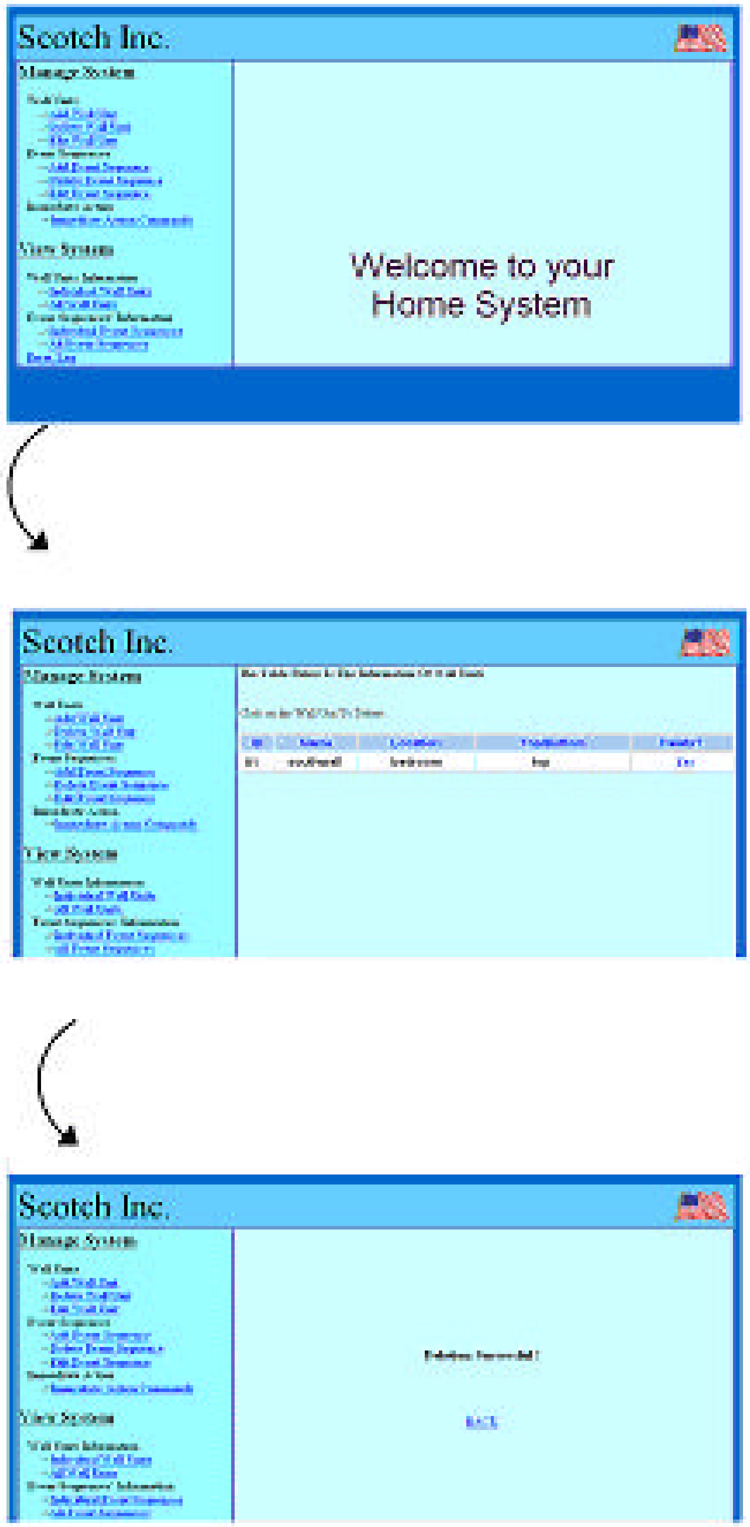


Figure 2: Delete Wall Unit

2.3 EDIT WALL UNIT

By clicking the button in the left-hand side menu that says “Edit Wall Unit” you go to a table of the attributes of all the wall units that are in your system. On the right side of the table there is a column titled “Edit?”

To edit a wall unit simply click on the “Edit” button in the “Edit?” column of the desired wall unit.

The selected wall unit information will appear with the current information entered. To change an attribute simply delete the current information and type in the changes. To reset all the values click on “Reset”. Click on the “Submit” button at the bottom when you have made the changes.

See Flow Chart below for the web pages in sequence.

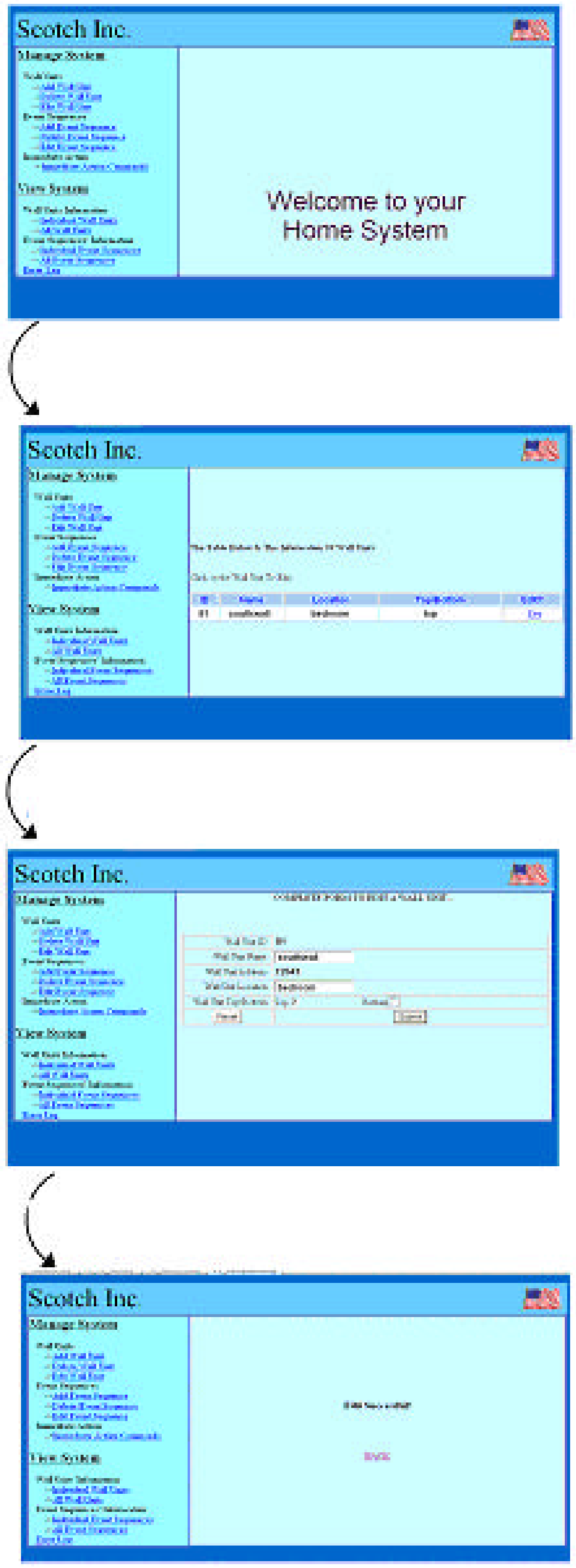


Figure 3: Edit Wall Unit

3. MANAGE SYSTEM BY EVENT SEQUENCE

3.1 ADD EVENT SEQUENCE

When you click the button in the left-hand side menu that says “Add Event Sequence” a form will appear. Enter the values for the event sequence name, start time, start date, end date, and the days for it to operate on.

When you are done click on the “Submit” button.

Another selection appears that shows all the available wall units to add to this event sequence. To select a wall unit for operation left click your mouse on the name of the desired wall unit. To select multiple wall units hold down the ‘Ctrl’ while clicking on the wall unit names.

Once all the desired wall units are selected click on the “Wall Units Selected” button.

The table that appears will list all the selected wall units and the possible operations; Turn On, Turn Off, Low Power, Low/Medium Power, Medium Power, High/Medium Power, High Power.

Click on the desired operation for each wall unit and they will perform that action at the time the event sequence starts.

Click on the “Submit” button to add the event sequence.

See Flow Chart below for the web pages in sequence.

Figure 4: Add Event Sequence

3.2 DELETE EVENT SEQUENCE

By clicking the button in the left-hand side menu that says “Delete Event Sequence” you go to a table of the attributes of all the event sequences and their related wall units that are in your system. On the right side of the table there is a column titled “Delete?”

To delete a event sequence simply click on the “Delete” button in the “Delete?” column of the desired event sequence.

See Flow Chart below for the web pages in sequence.

Figure 5: Delete Event Sequence

3.3 EDIT EVENT SEQUENCE

By clicking the button in the left-hand side menu that says “Edit Event Sequence” you go to a table of the attributes of all the event sequences that are in your system. On the right side of the table there is a column titled “Edit?”

To edit a event sequence simply click on the “Edit” button in the “Edit?” column of the desired event sequence.

The selected event sequence information will appear with the current information entered. To change an attribute simply delete the current information and type in the changes. To reset all the values click on “Reset”. Click on the “Submit” button at the bottom when you have made the changes.

See Flow Chart below for the web pages in sequence.

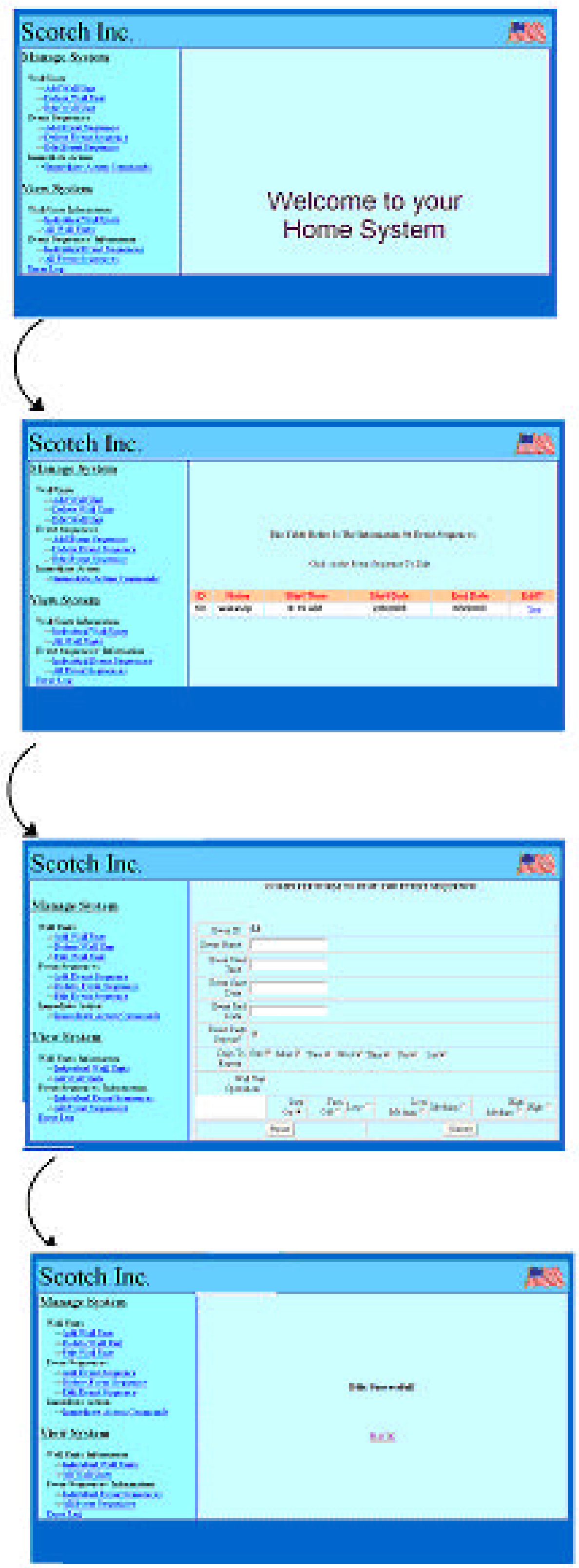


Figure 6: Edit Event Sequence

4. MANAGE SYSTEM BY IMMEDIATE ACTIONS

Immediate actions operate selected wall units and event sequences in real time.

4.1 WALL UNIT IMMEDIATE ACTIONS

Click on the “Immediate Action” button on the left-hand side menu.

Click on the “Wall Unit” button in the center of the page to operate individual wall units.

A table appears with all the wall units and their attributes in the system.

Select the desired wall unit.

A form appears with the options for operating the specified wall unit. These options are Turn On, Turn Off, Low Power, Low/Medium Power, Medium Power, High/Medium Power, High Power.

Once the option is selected click on the “Operate” button.

See Flow Chart below for the web pages in sequence.

4.2 EVENT SEQUENCE IMMEDIATE ACTIONS

Click on the “Immediate Action” button on the left-hand side menu.

Click on the “Event Sequence” button in the center of the page to operate individual event sequence.

A table appears with all the event sequences and their attributes in the system.

Select the desired event sequence.

This will run the wall units and their specific operations that are defined in the selected event sequence.

See Flow Chart below for the web pages in sequence.

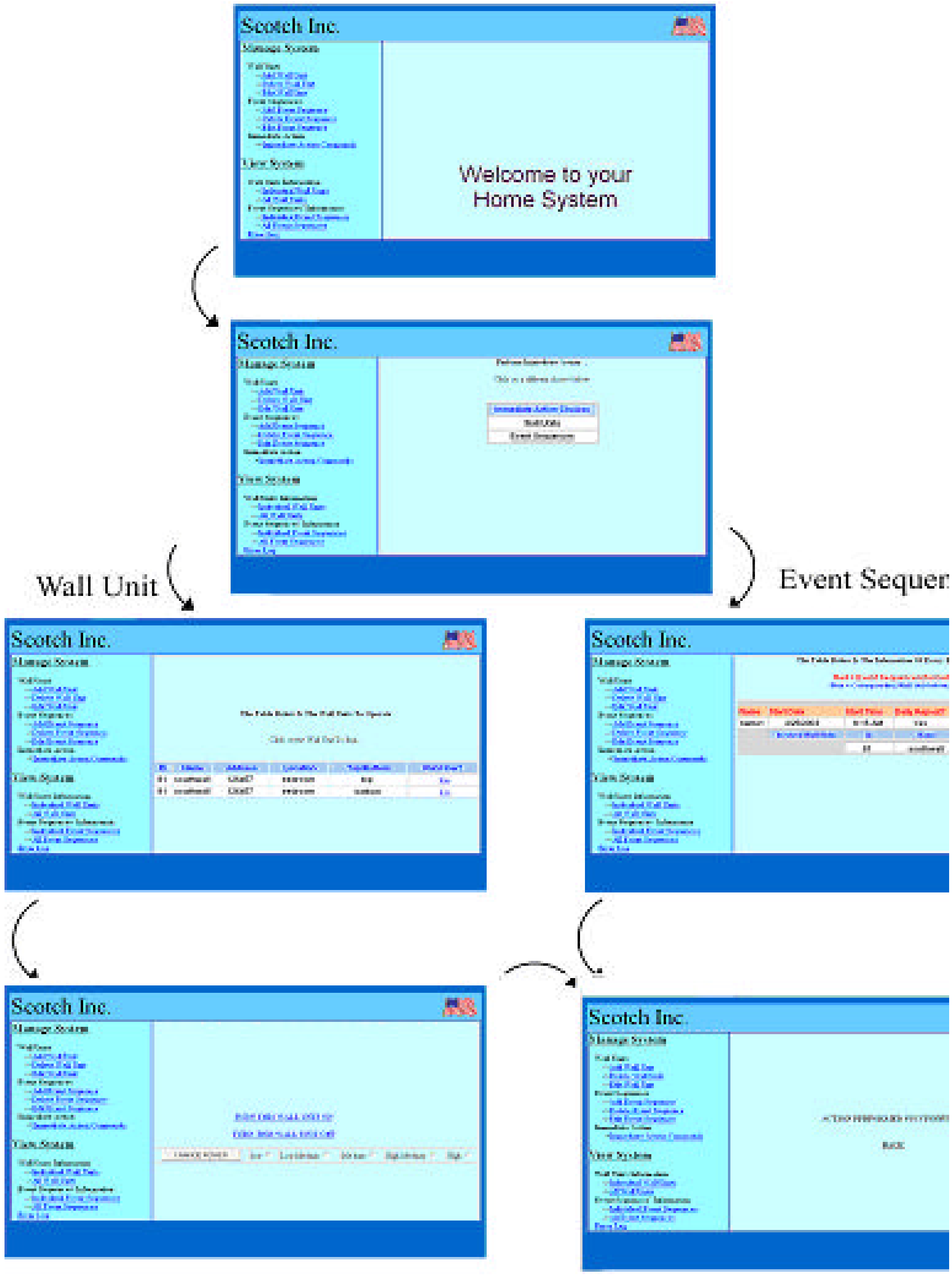


Figure 7: Immediate Action

5. VIEW SYSTEM

You can view the system performance in terms of Power, Voltage, and Current consumption for wall units.

5.1 VIEW INDIVIDUAL WALL UNITS

Click on the “View Individual Wall Units” on the left-hand side menu.

A list will appear showing all the different filters for selecting the wall units; “by Name”, “by Location”, and “by ID”.

Each of these buttons will display a list of all the attribute you selected. For example if you selected the “by Name”, a list of all the wall units will appear and the names will be the buttons.

Select on the desired wall unit.

A list of dates will appear. These dates are the dates that this wall unit was active.

Select on the desired date to view its performance.

A graph will appear with the selected wall unit’s power consumption throughout the course of the selected date.

NOTICE

The power consumption graph is the default graph. To view different graphs for the selected wall unit click on the buttons below.

The “Current” button will show the current consumed by this wall unit during the selected date. The “Voltage” button will show the voltage consumed by this wall unit during the selected date.

See Flow Chart below for the web pages in sequence.

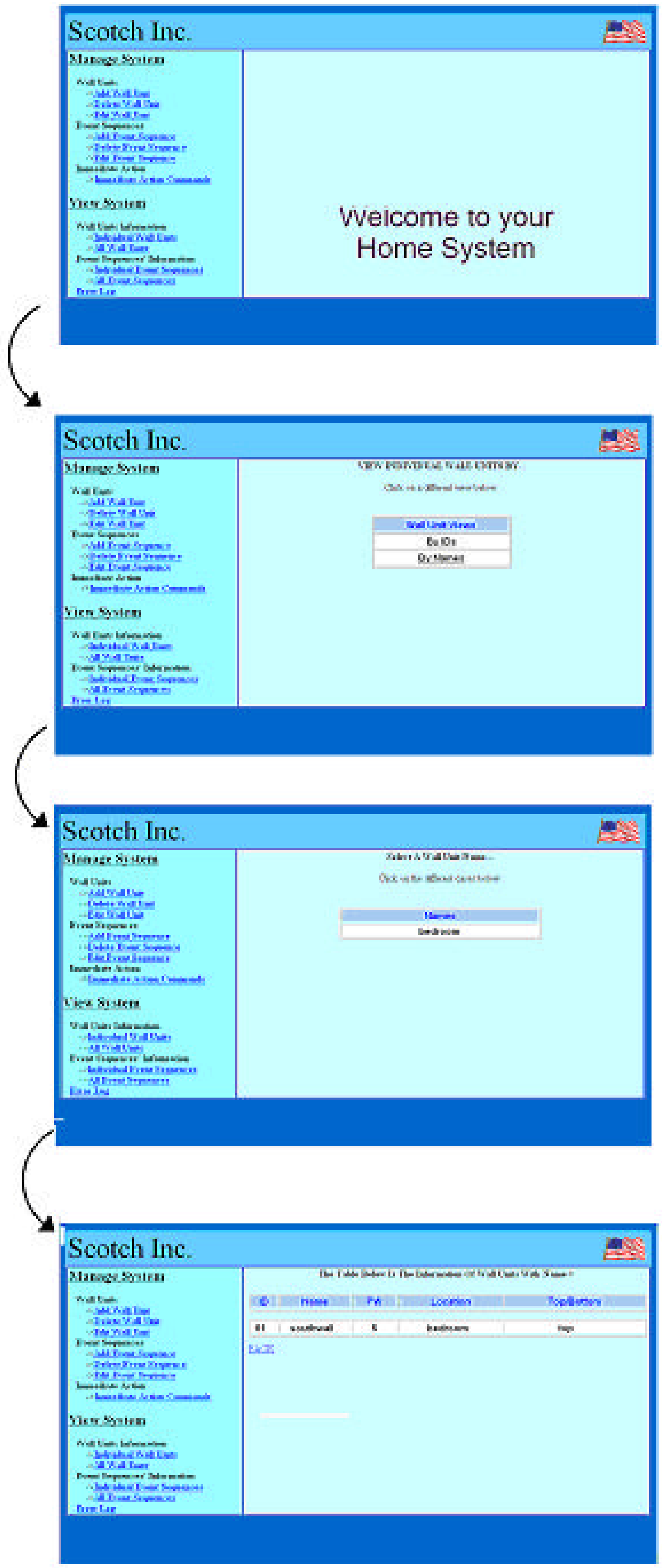


Figure 8: View Wall Unit

5.1 VIEW INDIVIDUAL ERRORS

Click on the "Error Log" on the left-hand side menu.

A list will appear showing all the errors the system collected.

Senior Design:
User Interface

1. Overview

User interfaces allow a user to control a system or device. In this manner our user interface will allow the customer to manage, view, and operate their system of devices. This document explores the implementation details of the user interface, as well as specifics concerning design decisions regarding components, protocols, and performance.

2. Basic Functionality

The user interface is meant to provide the customer with a method for interaction with the system through the control box. The control box will host a web site that will provide this functionality. This web page will be on the Internet either home or business. The web site will allow the user to operate the system on any operating system and computer with a browser.

2.1 User Interface Functionality

The user interface serves several purposes, including:

- Add events & schemes (An event is a scheduled operation to a specific device. Schemes contain multiple events and execute them concurrently.)
- Deleting events & schemes
- Editing events & schemes
- View events & schemes by:
  - Types of devices
  - Individual devices
  - Individual schemes
  - Overall
- Providing Security
- Notify user on specified events

2.1.1 Adding/Deleting/Editing Events & Schemes

The user can create an event with the attributes; Name of Device, Time of Operation, Devices & What they do, Days of the Week, Reoccurring (Y/N), and Length of Operation. These will be sent to the control box for storage and scheduling.

Deleting an event or scheme will send the control box the information to delete said event or scheme from storage.

Editing events or schemes will update the control box with the necessary information. The user can edit events individually, or by schemes.

2.1.2 View Events & Schemes

The web site displays the following scenarios for viewing the system and it's devices. This will also show the devices' information, including the power usage, schemes, name, location, and status.

2.1.2.1 Types of Devices

There are different types of devices (fluorescent lights, incandescent lights, and machinery). This option allows the user to view all the devices' information and their schemes according to their type.

2.1.2.2 Individual Devices

This option allows the user to single out a specific device and view it's information.

2.1.2.3 Individual Schemes

This option allows the user to single out a specific scheme and view its devices and their information.

2.1.2.4 Overall

This option shows all the devices and their information.

2.1.3 Providing Security

First the web page will require a user login and password. We will use security constraints to ensure that the system isn't tampered with. We will force the login name and the password to be different from each other. The control box then checks login/password combinations that are basic hacker attempts such as ROOT/ROOT and GUEST/GUEST. The web page will lock for a, yet to be determined, time if there are ten consecutive incorrect login attempts within a certain amount of time. These measures block attempts to break into the system and provide the user with peace of mind and confidence in the system.

3. Explanations and Options

3.1 Choosing a Web Site

The system controller could be in the form of a program installed onto the user's PC. This would require various versions of the program for different computers' requirements. The issue of connecting the control box to the users PC through the COMM port is a complex issue when taking into account the different types of computers on the market today. Setting up a web site for the system controller allows any user with a PC that can browse the Internet to use the system.

3.2 Creating the Web Pages

The web site was created using Microsoft Development Studios Professional .NET. The control box will host the web pages. Any other web design program would be acceptable for the development of the web site.

3.3 User Requirements

The user's home computer would require an Ethernet card, which can be purchased for a relatively low cost. The Ethernet card is needed to communicate with the control box.

4. Cell Phone Interface

In order to allow users to manipulate the system from outside their homes there needs to be a interface that can be accessed almost anywhere. We chose to use a cell phone to interface the system because of the accessibility of cell phones. However this is not an easy feature to include. To accomplish this task we would either need to attach the control box to a phone line and have the user call into and access the server or use a Wireless Access Protocol (WAP) enabled server to host a cell phone web page. We chose to use a WAP enabled server because there is no requirement for a modem or a hard dedicated phone line to the server.

4.1 Software Involved

This cell phone functionality requires some extra software be added to the control box as well as a few software protocols that need to be observed. All cell phones that can access the web use a special internet to access the web. These sites are WAP enabled to limit the amount of information that needs to be passed between the cell phone and the site.

4.1.1WAP

Wireless Access Protocol. This is the current protocol used by cell phones to pass information over the cell phone web. We have to observe this protocol if we want to send information to to a cell phone and if we want to send information back to the site. Luckily WAP does allow for ASP pages to be enabled for cell phone web use.

4.1.2WML

Wireless Markup Language. This is the current language that cell phone web browsers can understand. It is a subset of XML with a pre-defined set of tags. We used WML version 1.1 which is the latest version enabled on all phones. The tags have been pre-defined to limit the amount of software needed on the cell phone.

4.2 Development Software

We used a development suite to create and manage our cell phone web site. It is called MobileDev Studio and offers many features helping a user create a cell phone web site. It offers a graphical relationship view which helps visually create the site including how a visitor would move from one page to the next. It also offered a set of templates in creating a page that helped with lots of the low lever WML functionality. However this software was very frustrating because it would reject code that was correct but not in the prefect pre-defined order. So it was very tough to learn their pre-defined order but after the few speed bumps our site was up and running in no time.

Senior Design:

Control Box Specification

1 Overview

The main control box acts as the central point between the other two portions of the system. It communicates to the Wall Units, send commands to them and receive power information from them. It also is to be able to communicate with the users home PC so that it can receive commands from the user and send information about the house's wall units when requested. The system is able to perform these functions within a few seconds but, if any function gets delayed the system can handle an extra second or two wait, and even up to a delay of less than 30 seconds. The system cannot have delays of over 30 seconds because every 30 seconds polling orders are sent to the devices, so if the delay was over 30 seconds the orders queue could become bigger and exceed space limitations. The central control box has two major components, the hardware and the software. There are no size requirements for the central control box, and there are relatively few other requirements and restrictions but those that do exist will be discussed in the appropriate section. The cost of the box should be minimized as much as possible to allow the feasibility of placing the final product on the market for consumers to buy.

2 Hardware

The hardware is composed of individual components that allow software to run. The hardware has little requirements that need to be considered. Those that do exist are met with the hardware that we chose. The three main requirements are: a connection between the central control box and the wall units, another is the cost of the hardware, and finally the central control box needs be connected with the users home PC. They will be discussed later in the section. To limit cost and complexity, most of the control box will be composed of software to complete the required tasks. The hardware's only responsibility is to run the software and provide the control box with appropriate ways of communicating with the rest of the system.

2.1 Server

The hardware needs to meet all the requirements for the control box. We choose to use a Microsoft server to act as the back ground operating system. We need to have hardware to meet the minimum hardware requirements of Windows 2000 server. However we also want to choose hardware better then the minimum to allow the server to function better and faster. There are 4 major parts of the server that we had to consider: CPU, Memory, Extra Disk, and Network.

2.1.1 CPU

The minimum CPU to run windows server is a Intel Pentium 100 MHz. However this would offer slow performance, we decided to use an AMD Athalon 2000+ XP. This chip offers us all the computing power that we Windows 2000 Server needs and also allows for fast speeds.

2.1.2 Memory

The configuration that we have chosen does require a fair amount of RAM. We have chosen to use 512Mb which should be more than enough to complete all the functions of the control box.

2.1.3 Hard Disk

Windows Server requires at least 2 Gigabytes of hard disk space to run, but with the database software that we have chosen to use much more hard disk space is needed to perform correctly. The system that we bought offered a 60 Gigabyte hard drive. This will be more then enough to let our system run well.

2.1.4 Network

The central control box needs to be connected with both the wall devices and the users home PC. The wall devices will be connected wirelessly. The wireless component is connected to the server via the serial port so the network between the hardware and the wall units is done by RS232 communication. The users PC will be connected to the hardware via Ethernet connection to allow a fast connection.

2.1.4.1 Ethernet

The board computer comes with Ethernet capability built into it. This is what connects the users home pc to the central control box and allows the user to interact with the system. Ethernet was chosen because the User Interface that was chosen is web based and Ethernet is the networking system of the Internet. Using the Ethernet port simplifies the connection of the users home PC to the system and lowers the cost because now we do not have to add a wireless component to the users PC and we don’t have to install new software.

2.1.4.2 Serial Port

The serial port is built into the board computer. This is the component that will connect the central computer with the wall units via the added wireless component. Using the serial port for the connection of the wireless component allows us to buy a pre-built wireless board with serial connection, which limits the work involved when connecting the components.   Because the

2.2 Wireless

The connection between the central computer and the Wall Units will be done via wireless communication. This was decided to avoid re-wiring of the users home. We could also have chosen to use X-10 technology however we wanted to gain experience with wireless technology and therefore made the decision to use it. The wireless portion of the hardware can be broken down into the Component and the Protocols used to send data across this network.

2.2.1 Component

The Component is a simple chip and connected antenna to send out the data. The chip receives data over the RS232 port and then modifies the data to be in conformance of the protocols needed to send it across the wireless network. This chip was chosen because it was the most affordable with the best development kit that we could find. It only uses 900 MHz technology which is not top of the line, and if this were to be developed 2.4 GHz should be considered.

2.2.2 Hardware side Protocols

The wireless component uses the data that is being sent across the network to charge the capacitor. Therefore we need an equal number of 1's and 0's so that we don't over or under charge the capacitor.

2.3 Universal Backup System

The database automatically saves any updates that are made. However if there is a power failure we want to make sure that the system shuts down correctly. This is why we included a backup system with the box. Instead of building one we purchased a 5 minute APC backup power supply. The APC not only provides an extra 5 minutes of power but it also provides software to create a proper shutdown sequence.

3 Software

The software side of the system provides all the functionality of the system that the hardware doesn't. The main functionality of the software is to check for event sequences, keep track of time, send commands to the devices, poll the devices for current power information, and host the user interface. We are using Microsoft's CTime function in the AFX classes to handle keeping track of time. Event sequences handling are discussed in section 3.2.1. Commands are sent via the RS232 port with the SerialIO thread in section 3.2.8. The devices are polled every 30 seconds and the information is stored in the database. The user interface is hosted by the system with the internet hosting service provided by windows 2000 advanced server.

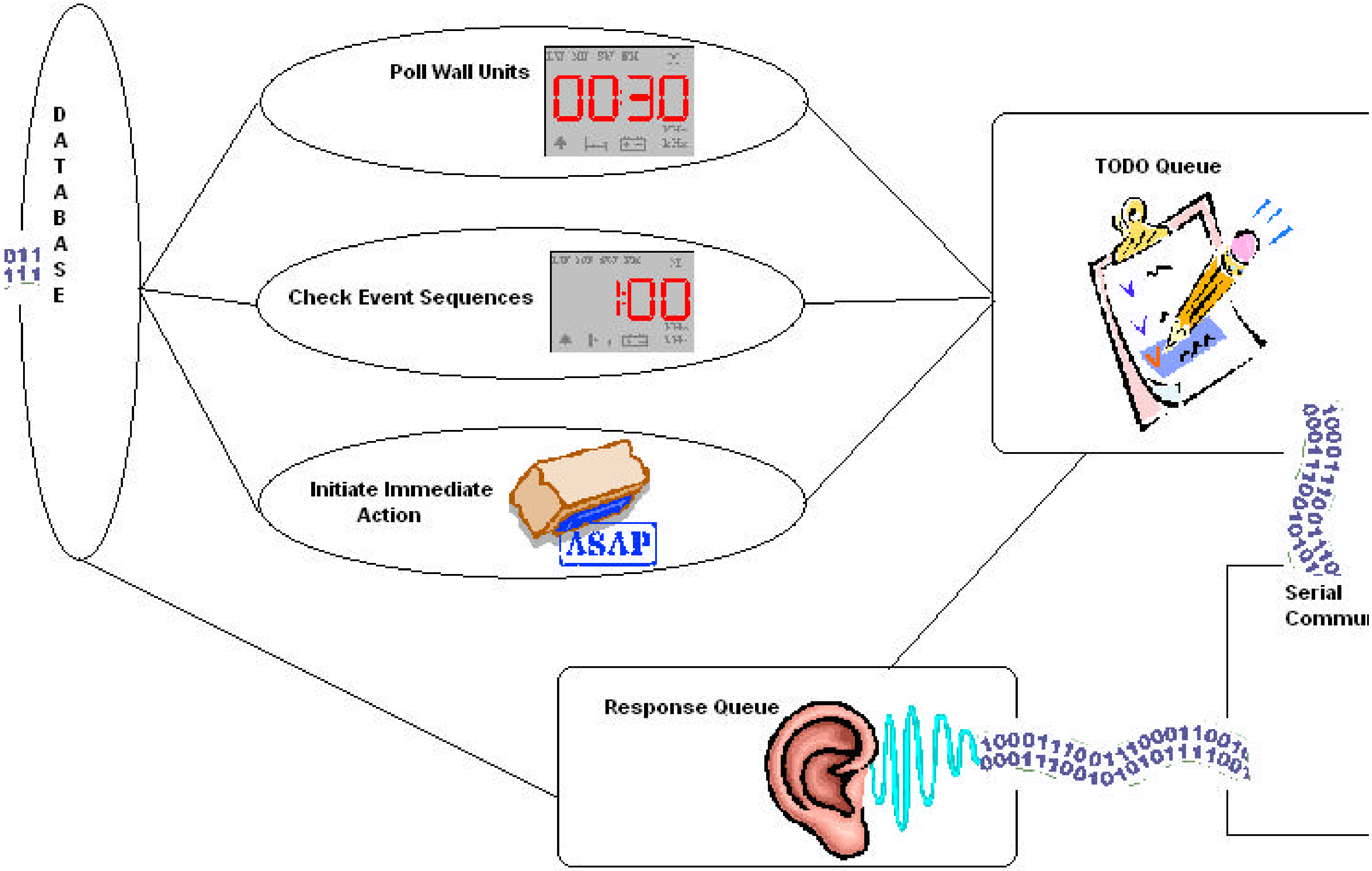


Figure 9: Control Box Flow Chart

This is the control box flow chart. The program interacts with the database and reacts to the information. It checks the database at specified intervals to do different tasks and create a todo object which is then executed and sent out through the serialIO thread and inserted into the response queue.

3.1 Operating System

The central control box will be powered by Microsoft Windows 2000 server, which will in turn run the program that allows the system to function. Windows Server is a well-defined operating system that is compatible with many sets of hardware including the one that we have chosen. We needed an operating system that could host a web page for the user interface, keep up the database, and run the program.

3.1.1 Web Server

Because of our decision to use an intranet web site as the user interface the server needs be capable of serving the web page. Microsoft 2000 Advanced server accomplishes this function. It uses web hosting software to allow information to be passed when requested for on specified ports. We used port 80 which is the default web port.

3.1.2 WAP Wireless Cell Phone WML Server

Because we wanted to offer a service to allow users to manipulate their system anywhere in the world we needed to develop a cell phone web page system that could perform basic tasks. However we couldn't just use the web page that we already created because it uses a much different technology and contains way too much information to send to a cell phone. We used a development studio to create the WML pages called MobileDev. This allowed us to create the pages using default templates and then make modifications after creation. It also gave a graphical flow chart of how the site goes from one state to the next.

3.1.3 ODBC Drivers

Microsoft Windows 2000 Advanced server also offers drivers that connect databases to other running programs including web sites. We used these drivers to connect and interact with the database. It stands for Open DataBase Connectivity.

3.2 Program

The program will be the main functionality of the central computer. It will handle keeping track of everything, including time and the event sequences that are set by the user. The program will also be responsible for hosting the user interface and listening for new commands from the user. The program has 11 classes to complete all the operations required. Full documentation is included in appendix C.

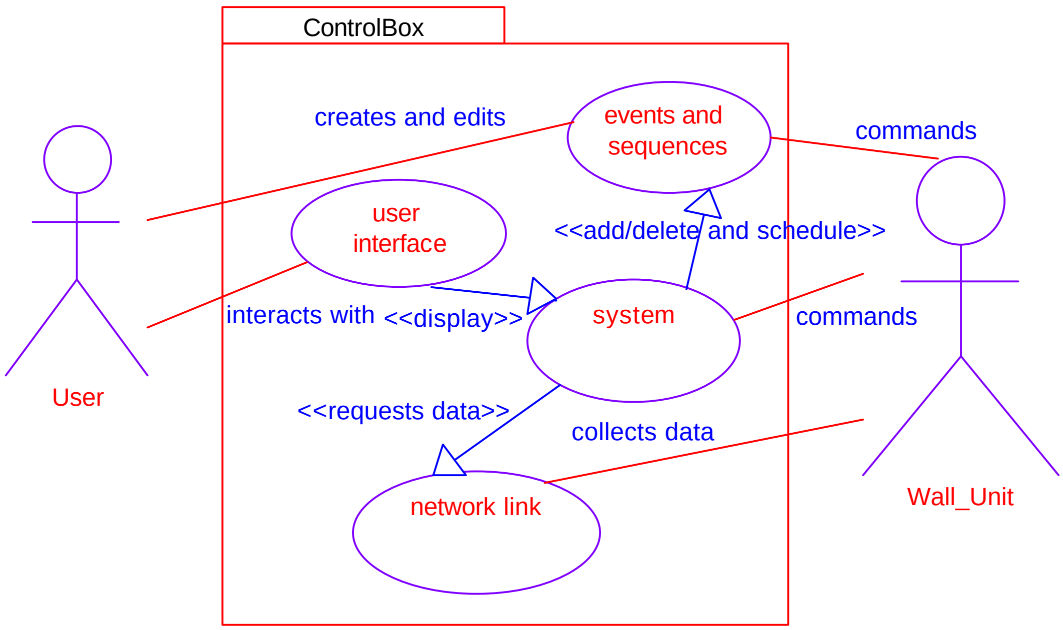


Figure 10: Control Box Use Case

This is the use case. It shows how parts of our system interact with each other. It provides information on how the user and wall unit interact with the control system and also how the parts of the control system interact with each other and the external actors.

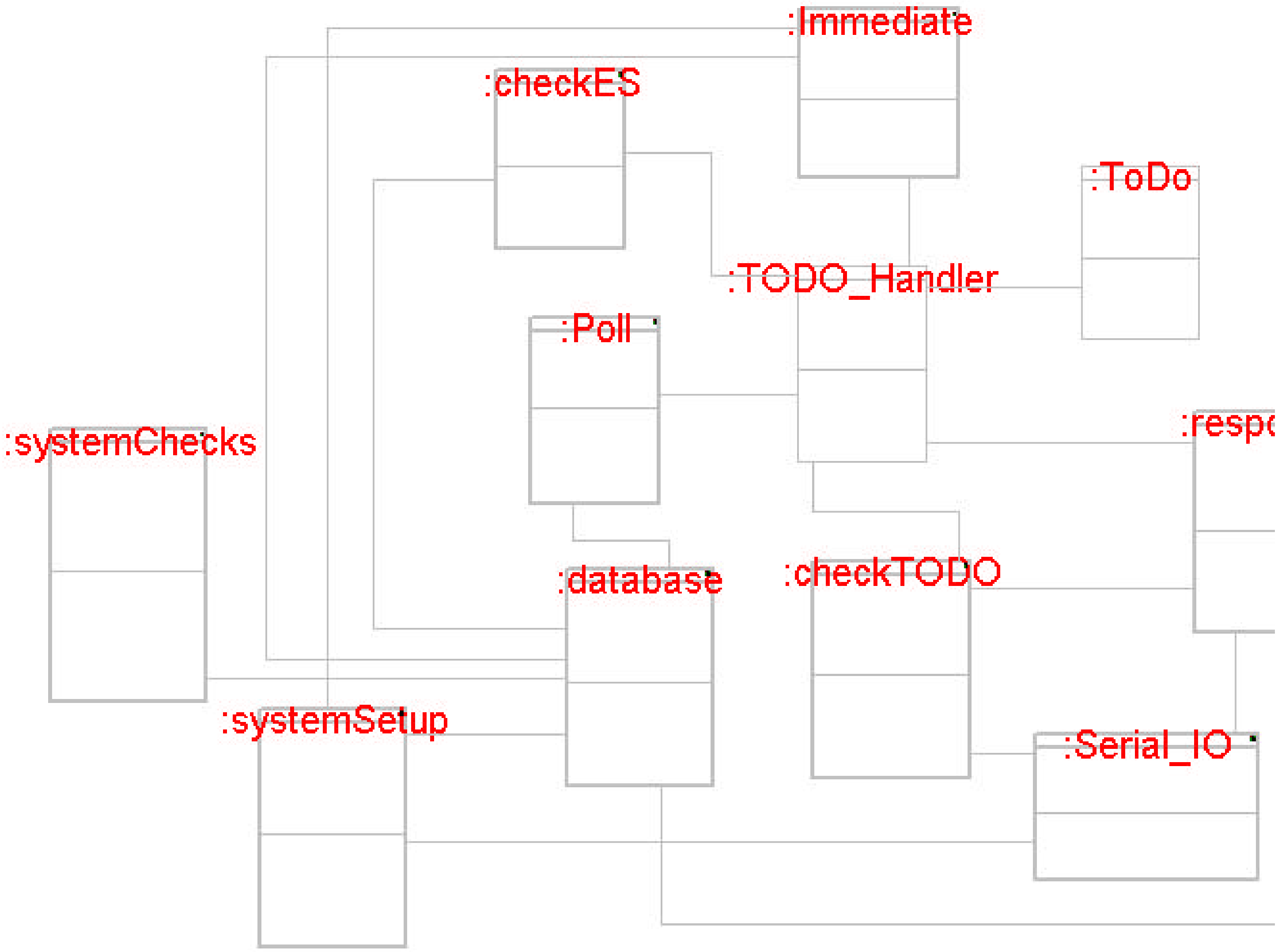


Figure 11: Control Box Object Model Diagram

This is the object model diagram. It shows how the system uses each class and how they interact together. Each class has a special function discussed in a section below. These duties range form database communication to creating polling commands.

We have chosen to write the program in the C++ language through the Rhapsody compiler. This allows us to create our program visually and then transform it into code.

3.2.1 CheckES

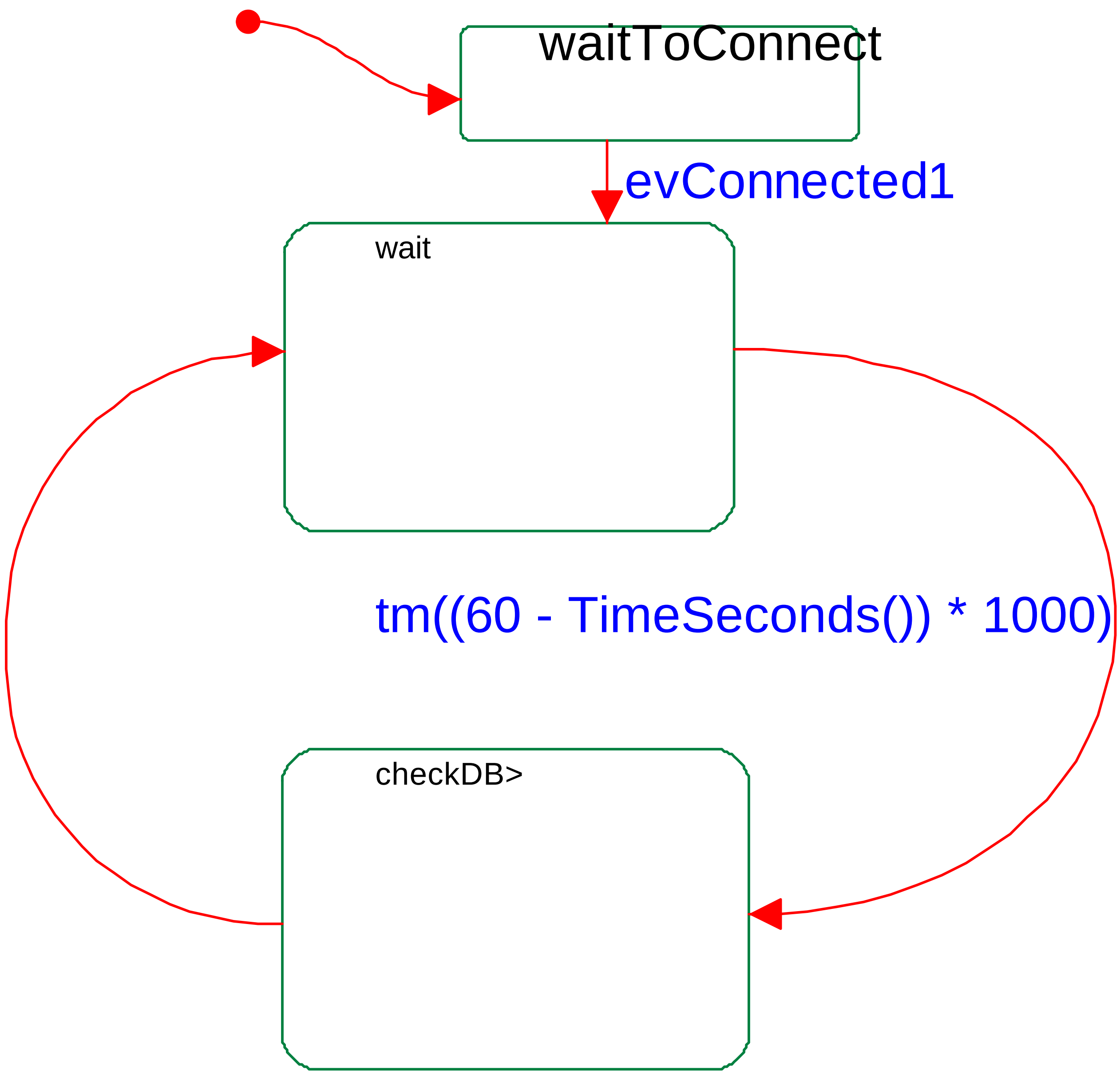


Figure 12: Check Event Sequence UML Diagram

The CheckES class checks the database at the start of every minute to see if an event sequence needs to start. This class interfaces the database and receives information about all the event sequences stored within. When an event sequence is started this class creates a ToDo through the TODO_Handler with the opcode in the database.

3.2.2 Poll

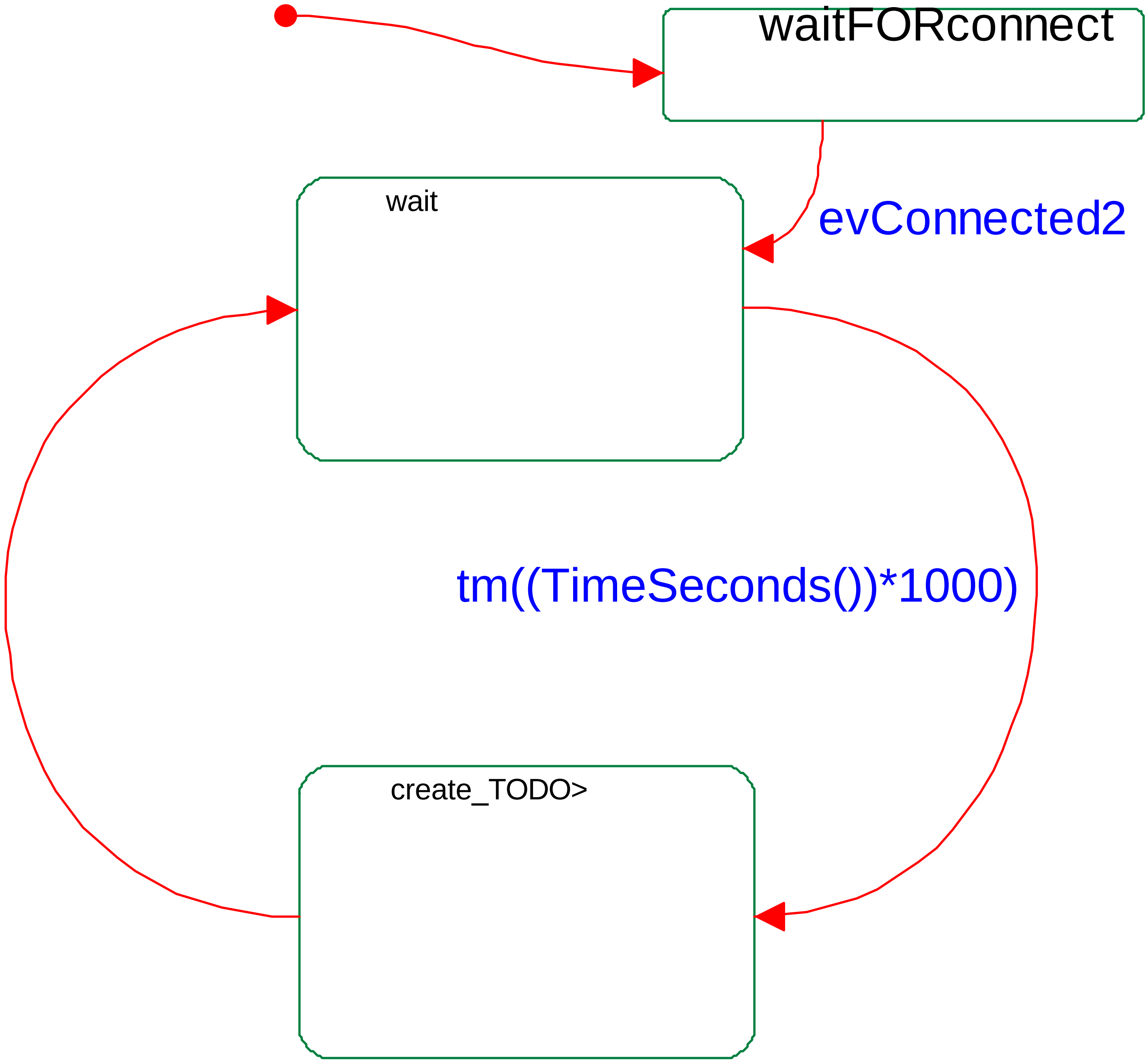


Figure 13: Polling UML Diagram

This class creates polling commands for every wall unit in the database. It performs this operation on every half-minute mark. When it needs to poll the devices it creates a ToDo through the TODO_Handler with a polling opcode.

3.2.3 Immediate

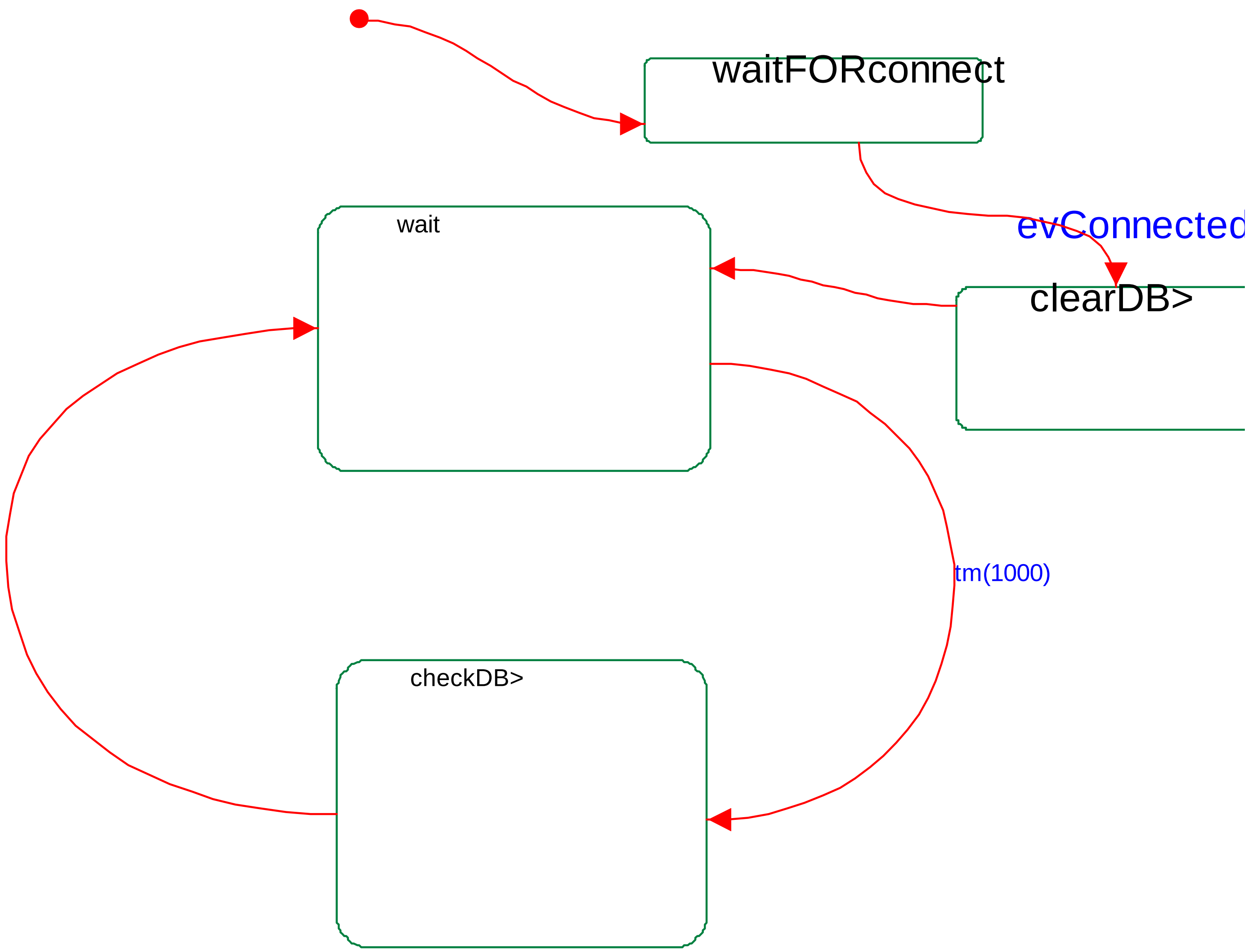


Figure 14: Immediate Action UML Diagram

This class continuly checks the database to see if there are any immediate action commands that the user has initiated. If there are then this class creates the required todo's and initiates the process of executing the command. It also has two special immediate actions, one starts a search for new wall units and the other immediately starts an event sequence.

3.2.4 TODO_Handler

This class handles the creation and handing out of assignments to be done. When either of the CheckES or Poll class needs to create a todo so the system can send the command to a device they use the NewTODO() function within the TODO_Handler. Then the TODO_Handler creates a todo it sends an event to the checkTODO class. CHeckTODO then uses the getNextTODO function to get the attributes of the next todo in the queue.

3.2.5 ToDo

This class holds the attributes of a command that need to be completed. The TODO_Handler creates and keeps track of the current ToDo’s.

3.2.6 CheckTODO

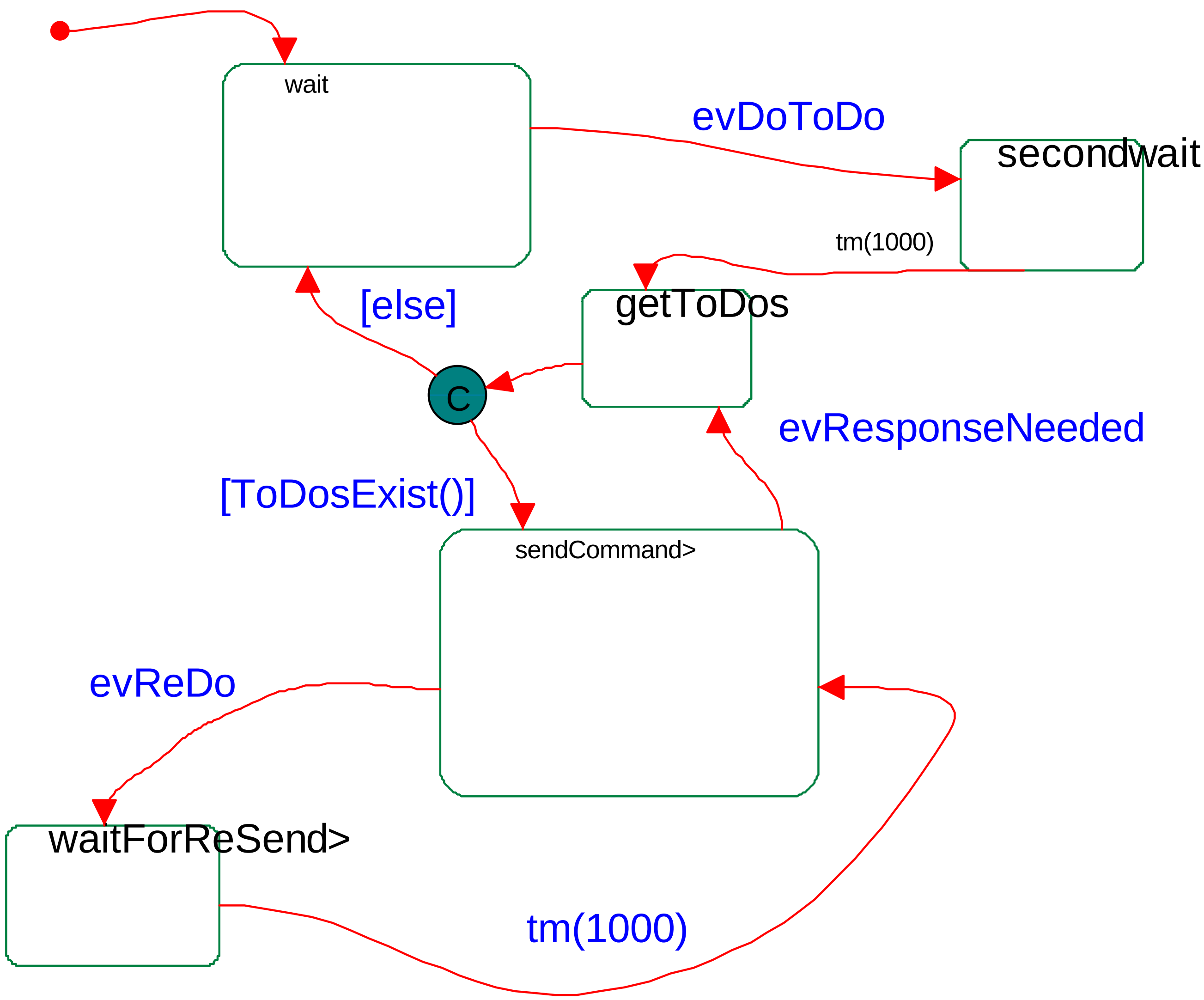


Figure 15: Check ToDo Queue UML Diagram

This class waits in an idle state for an event to let it know that there are todo's in the queue that need to be done. When it receives the event it continues and runs the function todosexist() which returns a true if todos exist or false if they don't. If a false is returned it goes back to the idle state until it receives the event again. If a true is returned it proceeds and builds the command to be sent to the effected device. CheckTODO then uses the serialIO class to send the built command to the respected device. On the successful completion of sending the command, CheckTODO creates sends the information about the todo to the response class.

3.2.7 Response

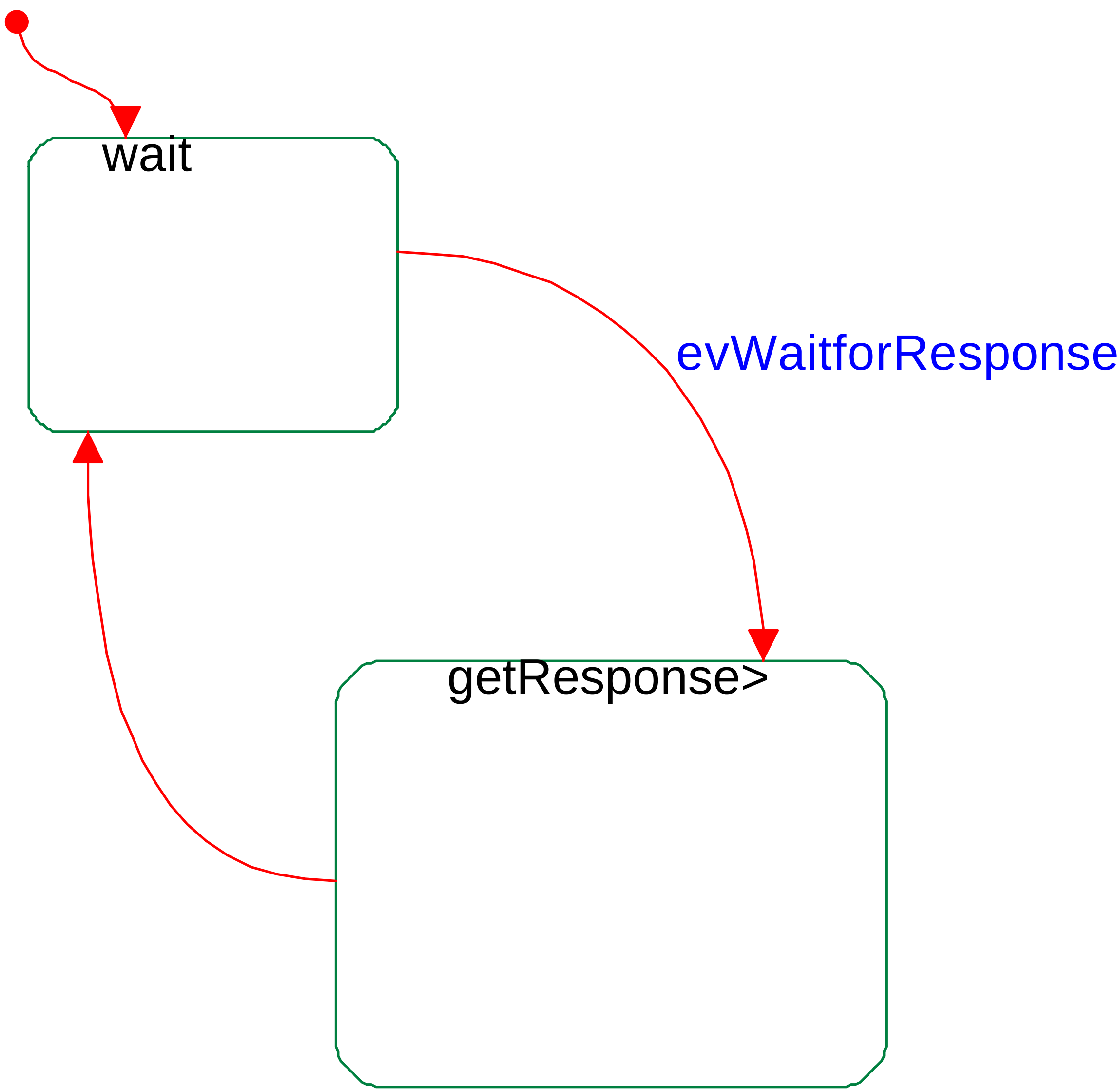


Figure 16: Response UML Diagram

After a command has been sent out this class is signaled. This class waits for a response on the serial port and compares the response to what is expected. This class uses the serialIO class to read and write on the serial port. If the read fails response then re-creates the todo that failed and it goes back through the queue to be sent out.

3.2.8 SerialIO

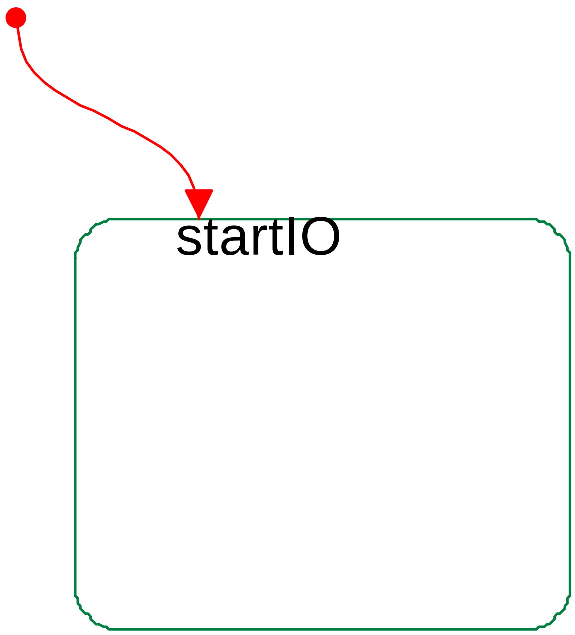


Figure 17: Serial I/O UML Diagram

This class operates the communication to the serial port. It is always in a running state incase CheckTODO or Response need to send or receive something on the port.

3.2.9 Database

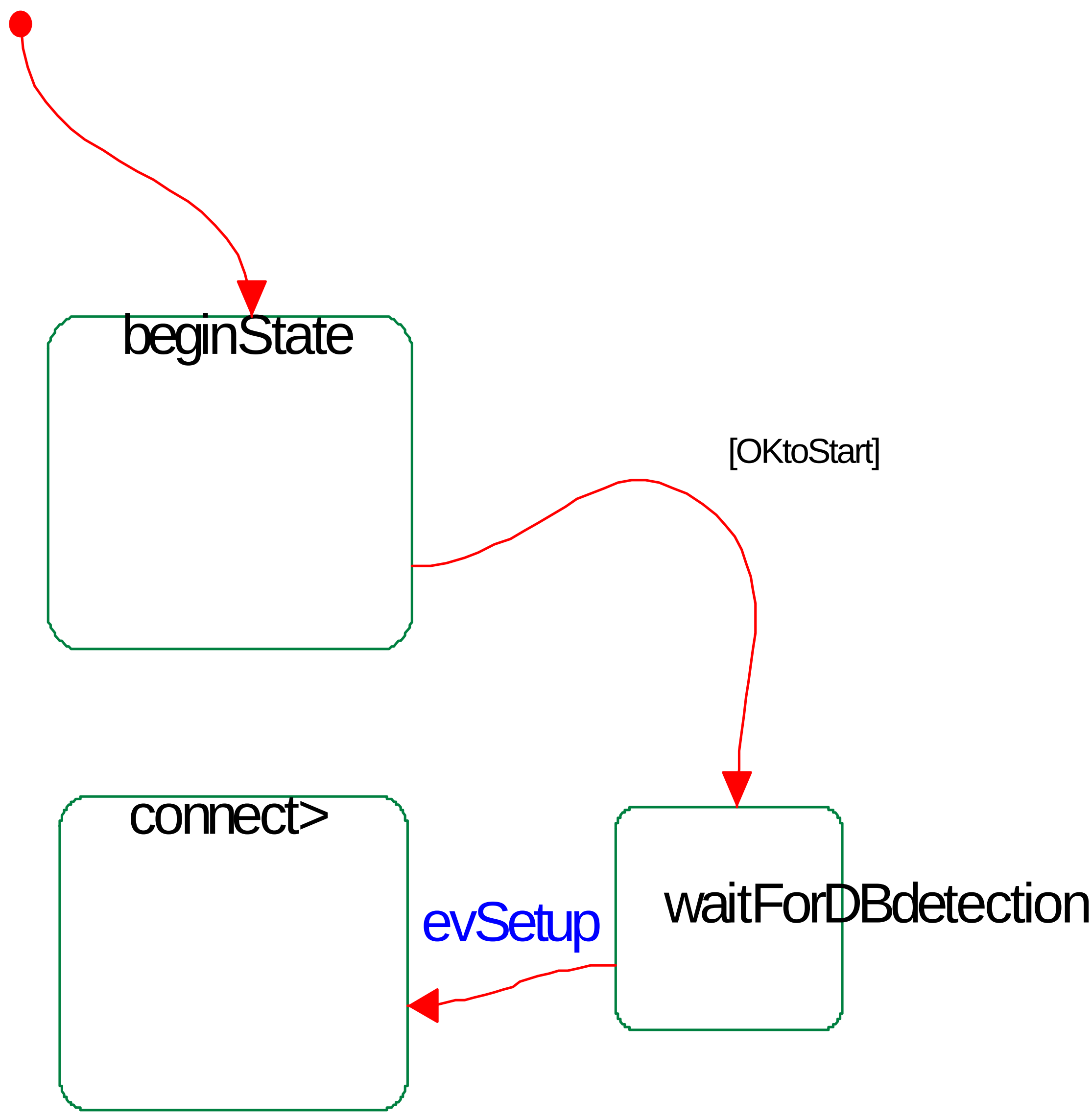


Figure 18: Database Connection UML Diagram

The database class is used to connect every other class in the program that needs to access the database to the database. It simply sets up the ODBC connection using a pre-defined name and the Microsoft ODBC drivers to connect. Then classes use functions to create a database object, execute a SQL command, and execute a SQL command with requesting information back.

3.2.10 System Checks

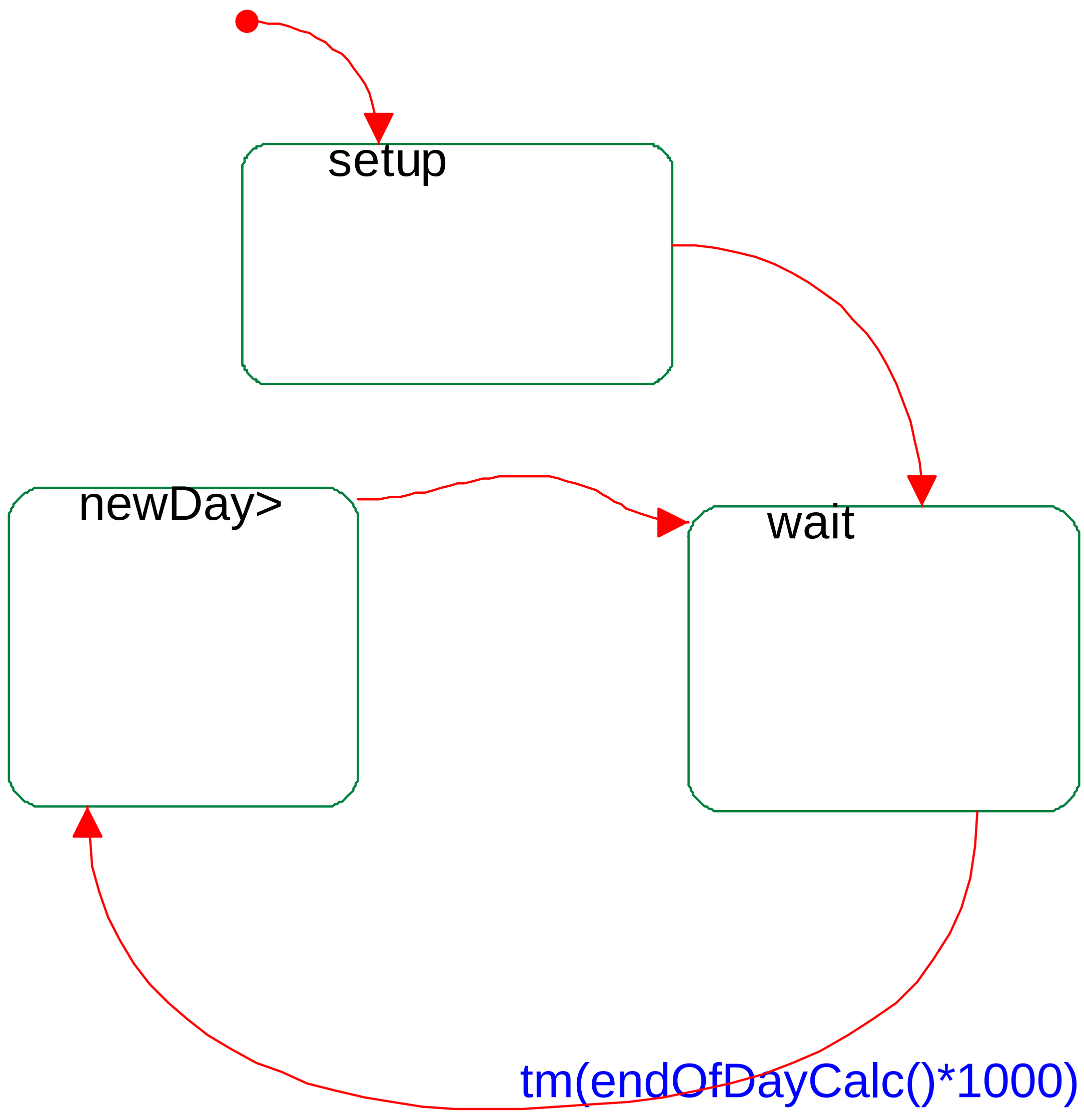


Figure 19: System Checks UML Diagram

The system checks class is used to complete system checks at the end of each day and at the start of a new month. At the end of each day the system checks the event sequence tables in the database for any entries that are past their end dates. This class also compacts power total information in the database at the start of each new month.

3.2.11 System Setup

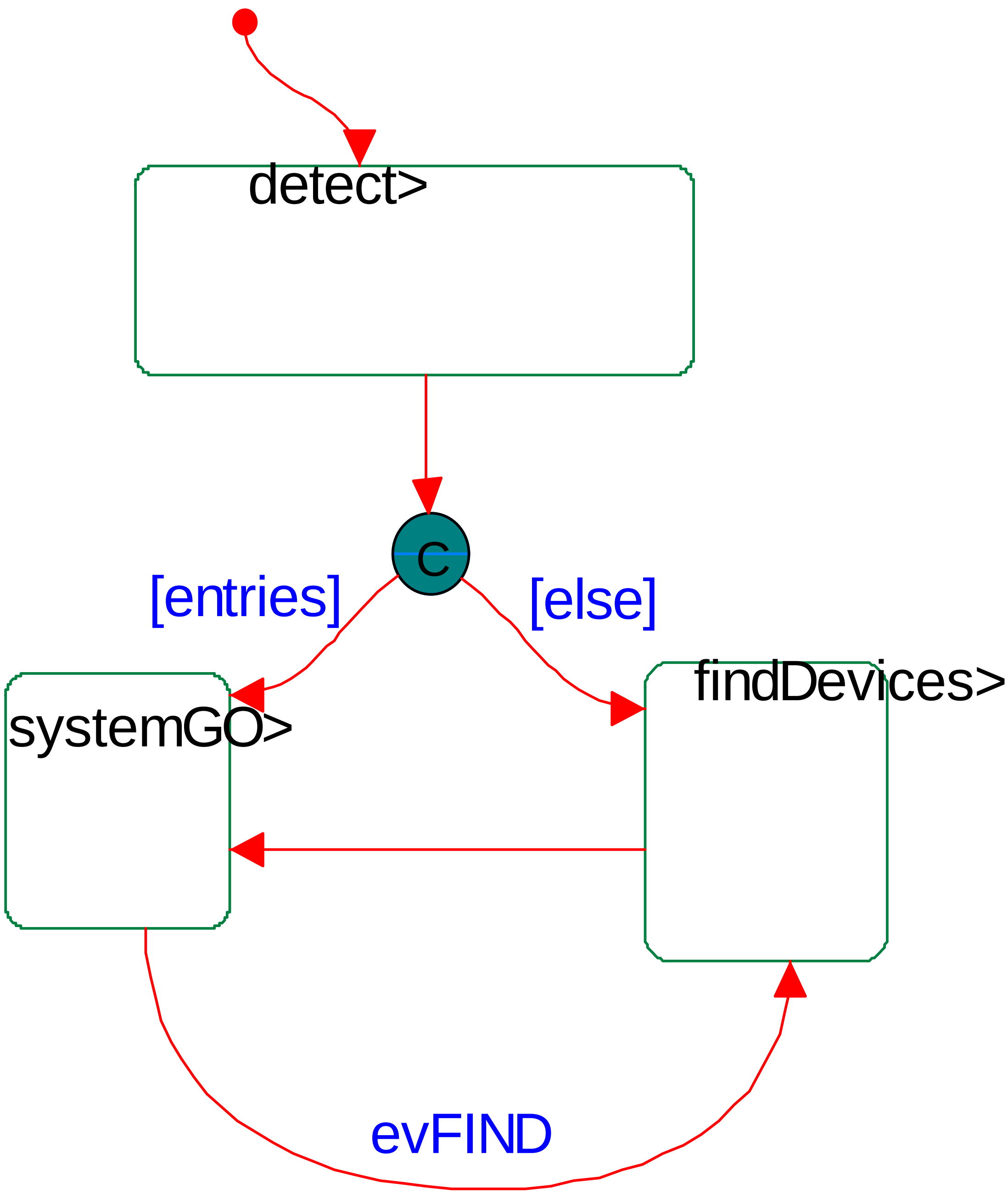


Figure 20: System Setup UML Diagram

The system setup class is used to check the database when the power is turned on to check to see if there are any wall unit entries. If no this means that it is the first time the system has been turned on or there has been a major error.

In either case the system needs to search for wall units. It does this by systematically checking each possible address combination for each 8 bit portion of the wall unit address. Then it combines an answers to find out the entire address for each wall unit. This function can also be induced by the user from the web page.

3.3 Software side Protocols

There are certain protocols that need to be followed to make sure that everything can communicate correctly. Talking over a network needs to be regulated to make sure that messages are sent across the network correctly.

3.3.1 Ethernet

The connection between the control box and the users pc will be running over regular Ethernet and therefore must take on the protocols of an Ethernet network. It will be using TCP/IP to allow the user to access the hosted interface easily with only using an IP address. The system must manipulate the data being sent over the Ethernet to adhere to the standards of Ethernet and TCP/IP protocols.

3.3.2 Wireless

There are many protocols that need to be considered when sending out commands to and from these wall units wirelessly. The main one is to make sure that commands don't come in on top of each other. This is important because the system needs to be able to receive single packets at a time not, and not garbage of many packets on top of one another.

3.3.2.1 Control Box Sending

The control box will send out commands to the devices at given intervals. The system will then wait to receive an Ack or a Nak from the devices. If no Ack is received the system will re-send the data again. This will happen 5 times, and if after 5 times the device can still not receive data an error message will be logged and shown to the user next time they log onto the user interface.

3.3.2.2 Control Box Receiving

The Control box doesn't need to send an Ack or a Nak after it receives information from the device because the device will assume that the control box received the information. If there is something wrong with the checksum or the packet is not received then the control box will simply re-send the command to request information.

3.3.2.3 Sending Packets

This is what is sent out to the wall devices by the control box when a command is processed. 70 bits will be sent out over the wireless network.

To/From: 48 bits, what is used to tell the devices where the message is from and to single out the device that needs to be commanded: 16 bits for the "from" part and 32 bits for the "to" portion.

Start Symbol: 8 bits like a password to let the device know the message is for it and it starts right afterwards.

Message: 12 bits, 3 for the OP code, 8 for the raise/lower offset, and one bit to let the device know which outlet is being changed.

Checksum: This is what the device uses to confirm the message is complete. The size will be 12 bits to.

Command 40 bit breakdown

Start Symbol	To	From	Message	Checksum
8 bits	32 bits	16 bits	12 bits	12 bits

Message 12 bit breakdown

OpCode	Offset	Outlet select
3 bits	8 bits	1 bit

Figure 21: 40 Bit Send Command

This is the packet that is sent back to the control box after the device has received its commands. Ack means that the device has received the command completely; Nak means that the device received something intended for it but there was a problem during the transmission. They both have the same format of 14 bits.

To/From: 48 bits total; 16 bits for address of central control box, 32 bits for address of intended device. This is used to let systems know who is supposed to receive the message.

Ack/Nak: 1 bit to let the system know if the message was received or not.

Ack/Nak 14 bit breakdown

To	From	Ack/Nak
16 bits	32 bits	1 bit

Figure 22: Ack/Nak Command

3.3.2.4 Sending Report

When polled, the devices need to send the current voltage and current levels to the central control box. It will do this by creating a packet with a size of 112 bits.

To/From: 48 bits total; 16 bits for address of central control box, 32 bits for address of intended device. This is used to let systems know who is supposed to receive the message.

Start Symbol: 8 bits; this is like a password to let make sure that the signal is sent to the correct device and is in tact.

Message: 48 bits; this is the selected sample that the control box is requesting. The control box is on a three minute cycle of which sample total to request for; voltage, current 0, current 1. This number is a summation of the sample in a ½ cycle. There is also 16 number which is the number of samples taken for the sample. This number is then used to divide the summation of the sample total.

Checksum: 8 bits; this is what the device uses to make sure that the message is complete.

Report 96 bit breakdown

To	From	Start Symbol	Message	Checksum
16 bits	32 bits	8 bits	48 bits	8 bits

Message 64 bit breakdown

Selected Sample	Num Samples
32 bits	16 bits

Figure 23: Report Command

3.3.3 OP code

We have created a protocol for sending commands to the wall units and have called it OP code. There are few but important commands that will be sent to the units. They include: Turn On, Turn Off, Regulate Power Level, and Report. There are only 4 commands that need to be sent to devices, so the OP Code can consist of 3 bits:

000: Report

010:Turn On

011:Turn Off

100:Regulate Power

his implementation allows us to easily check to see what the command is by bit comparison. If the command is a raise power or lower power command then an extra 8 bits will be attached for the amount needed to raise or lower. This will be the phase angle needed to have the TRIAC on the device cut the power curve at the right points.

4 Database

Data storage on the control box will be down with an Access database. Using a database allows a safe, reliable way to store all the information that the system will need and access. The database allows us to relate the data together. Through the ODBC drivers our program can use Microsoft's afx classes to communicate between the database and the program. The data relationships are shown below.

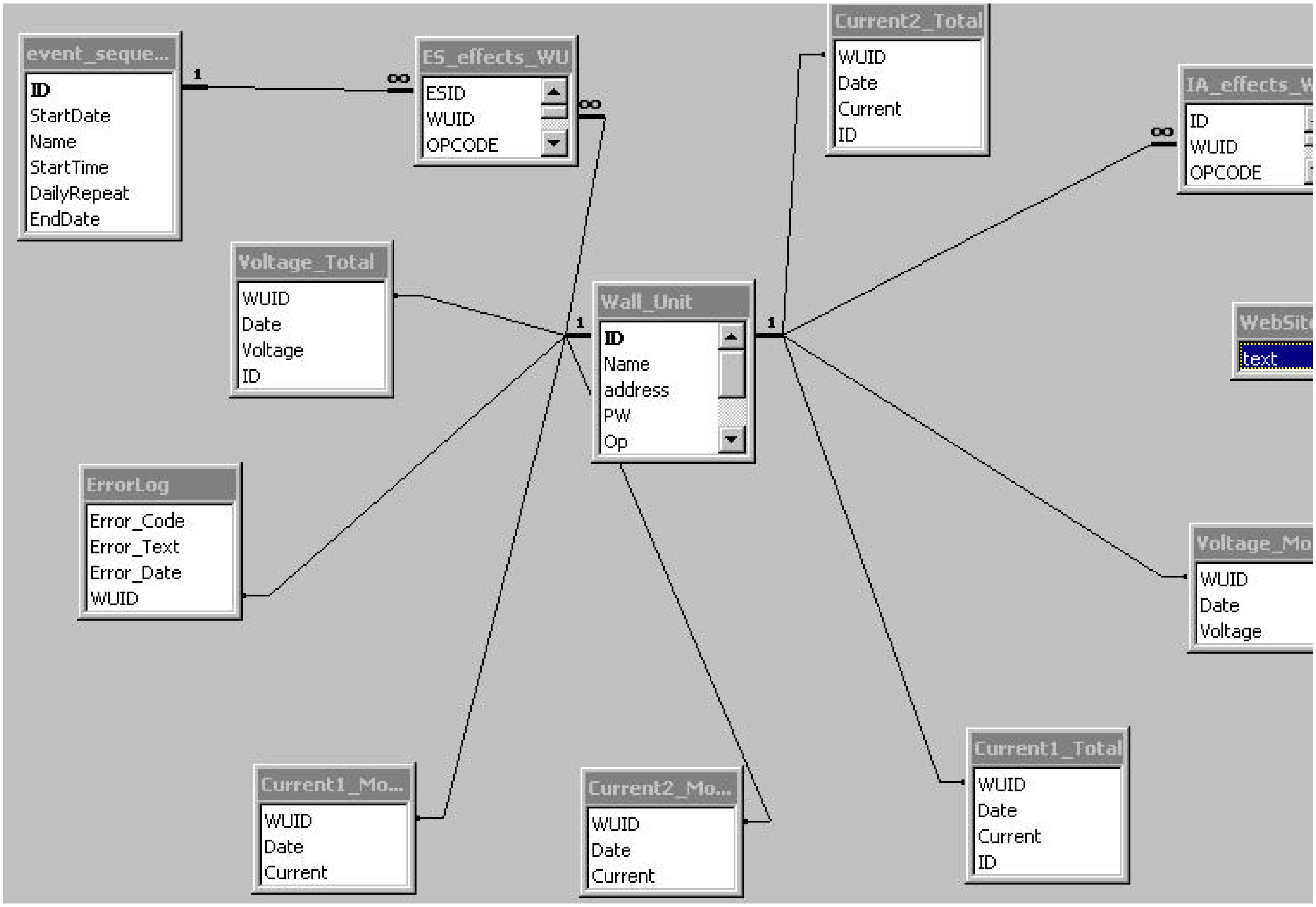


Figure 24: Database Relationships

The database also allows us to save the information on demand so that when requested by the web based user interface could be queried and reported to the user.

5 Alternatives

There are many alternatives to the hardware and software that we choose for this project. In this section we discuss some of the alternatives that we could have used or the requirements that need to be met.

5.1 Hardware Alternatives

5.1.1 Alternative Computer options

The central control box doesn't have many requirements that it needs to fill. Almost any hardware would have worked yet we choose the AMD chipset and Windows server and the capabilities it has. Also being a component we will be buying, we don't want to have to worry about issues of combining incompatible hardware together.

5.1.2 Alternative Wireless options

There are many wireless options out there and most would work for this system. We chose this one because it seems the best for the cost was available and dev kit gave two devices and better value for the money and satisfied requirements.

5.2 Software Alternatives

5.2.1 Alternative Operating systems

Any Operating system that offered web hosting abilities and database connection techniques could be used. Linux was our other option but we made the decision to go with Microsoft because of simplicity.

5.2.1.1 Linux/Unix

Linux could have been used, it is an excellent choice to run web applications. It also is a good operating system to host a database however our group has more experience using Microsoft products and the connection techniques.

5.2.2 Alternative Compiling software

5.2.2.1 Other C++ compilers

We could use another compiling program to help us create our system but we feel that Rhapsody is better because it allows us to compile the written UML code, which is used to design the system, into a working program. This saves a step of work. This makes the process easier to create our project.

5.2.2.2 Other Languages

Other languages could be used, but because the devices will be written in C we feel it best to write the system in C++. Plus the school has a license for Rhapsody with C++ only.

6 Previous Implementation

This hardware implementation was not the first we thought of but this it was the one that satisfied the goals of the project. We originally were going to use a more embedded set of hardware including a single board computer with limited space and resources. However we ran into limitations; disk space to store system information, VxWorks operating system transfer failures, and Webify web server short comings. These problems were not found until the third week of the second quarter and therefore the change to a new set of hardware was needed quickly. This is the main reason why we choose to use the windows 2000 advanced server setup because it was easy to obtain through the school and we all understood how it functioned. Here is the previous set of design decisions.

Here is the previous implementation design document.

2 Hardware

The hardware is composed of individual components that allow software to run. The hardware has little requirements that need to be considered. Those that do exist are met with the hardware that we chose. The three main requirements are: a connection between the central control box and the wall units, another is the cost of the hardware, and finally the central control box needs be connected with the users home PC. They will be discussed later in the section. To limit cost and complexity, most of the control box will be composed of software to complete the required tasks. The hardware's only responsibility is to run the software and provide the control box with appropriate ways of communicating with the rest of the system.

2.1 Board Computer

The board computer will be the hardware for our central control box. It is a 386 PC based computer. The size is small enough that the home user can put it anywhere in their home. The board computer can be mounted into a box just bigger then the size of a hardback book. The cost is minimized to a couple hundred dollars. The board computer is an all in one 386 PC with Ethernet and serial connections built in. The gain for buying the already built computer is that it limits the compatibility issues involved with combining different hardware components. There are 4 major parts of the hardware: CPU, Memory, Extra Disk, and Network. Full manufacture technical specs can be viewed in an Appendix.

2.1.1 CPU

The CPU that comes with the board computer is a 386 chip. Speed was not a requirement that we had to consider and this chip is plenty fast for our central control box. This chip offers more than enough performance power with a small cost and is compatible with most PC based hardware and software on the market.

2.1.2 Memory

There is 512 kilobytes of RAM. This will be plenty of space to store the software of the central control box and all of the data that the system needs. If more is needed, then swapping to the flash disk will be necessary.

2.1.3 Flash/Extra Disk

There is 512 kilobytes of Flash Disk. This will allow us to write the software and data to save incase the system goes down due to loss of power. The Flash Disk has a limit on the number of writes that can be executed, only 10,000 writes are guaranteed before the disk starts to degrade to the point where writing and reading are not possible. We have chosen to write to the disk once a week when changes have been made to backup the system as well as the simple UPS system when power is detected as unavailable. If 512K is not enough, an extra 32Mb Flash Disk can be used for additional storage, but it has the same constraints of the onboard Flash Disk.

2.1.4 Network

Same as current implementation.

2.1.4.1 Ethernet

Same as current implementation.

2.1.4.2 Serial Port

Same as current implementation.

2.2 Wireless

Same as current implementation.

2.3 Universal Backup System

Same as current implementation.

3 Software

The software side of the system provides all the functionality of the system that the hardware doesn't. The main functionality of the software is to check for event sequences, keep track of time, send commands to the devices, poll the devices for current power information, and host the user interface. A simple thread discussed in section 3.2.1.1 will handle keeping track of time. Event sequences handling are discussed in section 3.2.1.2. Commands are sent via the RS232 port. The devices are polled every 30 seconds and the information is stored on the central computer. The user interface is hosted by the system through the Webify component of Rhapsody; this is discussed in section 3.2.1.4.

3.1 Operating System

The central control box will be powered by VxWorks, which will in turn run the program that allows the system to function. VxWorks is a well defined operating system that is compatible with many sets of hardware including the one that we have chosen. VxWorks also gives us libraries to connect our software to the ports and hardware that our system needs to function. It also provides a powerful set of debugging tools that will help us perfect our system. Due to storage limitations we needed to choose an operating system that could be contained in a very small amount of space. We approximate that VxWorks kernel and the bare trimmings of extra software needed to run the system can be contained in less than 128K of disk space. (Plus, the school has a license for it so we wouldn't have to buy any software.)

7 System Changes

Currently the system design that we used to complete this project is not the most optimal. This design is too costly for a user to buy and insert into his or her home. The product would need to be smaller and reduced in scope to limit cost. The control box doesn't need to have all the functionality of a full server. All the control box needs is a web server, a database, and a limited OS to handle multithreaded applications. The control box would most likely look like a small box about the size of a Linksys switch router. In fact the control box would probably offer other functions such as a router and a switch. This way we could sell the box as an all in one unit that the user would just need to plug into the incoming internet connection and offer a home network to the user.

8 Gold Platting additions

During the process of contemplating and creating this system, we came up with many extra parts that could be added to the system to make it have more functionality or a better product for purchase but there wasn't enough time to implement them. In this section we will discuss the gold plating additions that could be added with more work or new hardware.

8.1 Dynamic Light Switches

One possible addition to our project would be to add the functionality to light switches. This would require removing them from the hardwire on and off to an outlet and making them work through the control box. This would allow for dynamic re-allocation of switches. It would save home owners from the pesky light switch that people always seem to turn off that resets the alarm clock or turns off the TV. This would require a little tinkering with the software because currently it is only listening for communication after a command has been sent out and this new feature would require constant listening.

8.2 Full Duplex Comm communication

Currently we are using half duplex communication which is a major limitation because we can not send and receive at the same time. In future versions full duplex communication would be desired to limit problems of communicating on the same wires for both sending and receiving.

Senior Design:
Wall Unit Specification

1. Overview

The "wall unit" is a piece of hardware that is inserted into an outlet or switch fixture located in the wall of a house or place of business, and is implanted in such a way that it is connected to the incoming power cables in order to monitor and regulate the power consumption. This information is then passed back to a central "command unit," which in turn performs a series of computations and stores the data for later use in regulation schemes and power consumption graphs. This document explores the implementation details of the "wall unit," as well as specifics concerning design decisions regarding components, protocols, and performance.

2. Basic Functionality

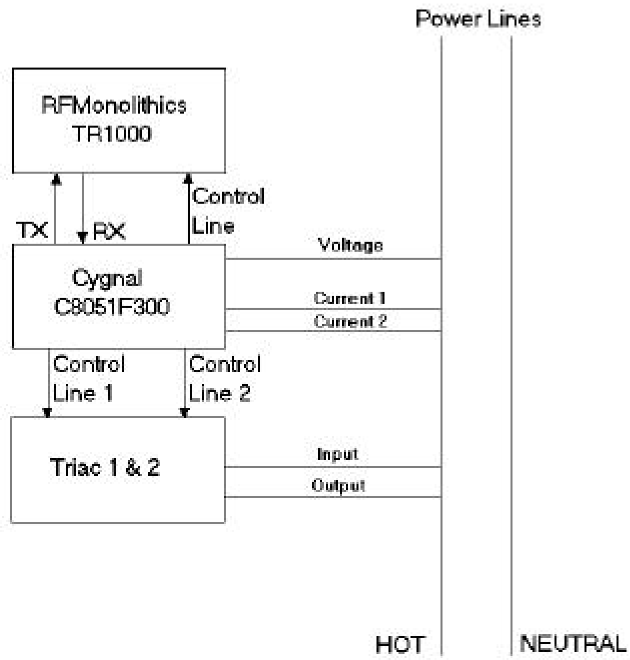


Figure 25: Wall Unit Block Diagram

The wall units provide a means of sampling power consumption on a given outlet by recording the current flow and voltage, this information is then sent back to the central control unit for further computation and stored away for later use. The other major functionality the wall units offer is power regulation, through the triac. These units remain idol listening for commands when not executing a received instruction and are slaves to the central control unit.

2.1 Microcontroller Functionality

The micro-controller serves several purposes, including:

- Establish UART for communication with the control unit
- Encode data transitions for DC balance
- Sample voltage and current through outlets
- Perform tasks based on commands form the control unit
- Provide system for power regulation

2.1.1 UART

The wall unit uses a software implementation for the UART. Additionally, the UART is required to run at 2400 baud to allow for integration with the wireless ASH transceiver that we have chosen to use. It is not essential to have high-speed data rates because the amount of information being transmitted will not be overwhelming, thus 2400 baud is sufficient.

UART transmission begins with a two byte preamble followed by an automatically appended start symbol to the beginning of every packet. Another start symbol is used at the beginning of every 32 bit word in order to guard against random data reception. In the receive direction the UART waits for reception of these two unique start symbols in order to begin data reception. Once a packet is either transmitted or received a flag is raised in order to notify completion and signal to start processing data in the case of a received packet.

2.1.2 Encoding Transactions for DC Balance

The wireless transceiver we have picked requires data to be DC balanced in order to achieve accurate sampling and data slicing over the course of a receive or transmit. This is due to the way in which the transceiver is designed; it uses an AC-coupled capacitor to maintain precision while performing data slicing. The problem with this scheme is, if the capacitor's charge becomes skewed to either the '1' or the '0' side of the information stream, data corruption becomes more and more likely; thus, a method of balancing transaction must be implemented.

There are two popular methods of achieving this balance, Manchester encoding and symbol conversion tables. Manchester coding, unfortunately, doubles the length of a transaction because a one is represented by a '1' followed by a '0' and a zero is represented by the exact opposite. On the other hand, conversion tables only require a 6:4 ratio of encoded data to original data. Symbol tables will usually, either be base on converting a byte or a nibble at time, into a DC balanced pattern. Also, these schemes must take into account the number of consecutive ones or zeros allowed during transmission. Manchester coding does this without any additional thought because it uses both a one and a zero for each data bit, affectively limiting the reoccurrence of successive ones or zeros to one. Symbol tables on the other hand, look to balance over the course of a bytes transmission, by applying consideration for balance during development of the symbols used and limiting the number of reoccurring successive ones or zeros.

We have implemented Manchester encoding in our system by sending out a bit followed by its inverse in the transmit direction and receiving only every other byte in the receive direction.

2.1.3 Analog Sampling of Current and Voltage

The microcontroller is required to have an onboard analog to digital converter. The resolution of the ADC is not overly important because we can increase accuracy using over sampling and averaging. Additionally, this is possible because we are sampling a signal of such low frequency (60 Hz) that over sampling was not be a problem.

The chip we have selected has an onboard ADC with 8 bit resolution, and inputs can be programmed from any of the eight I/O pins supplied. One pin is used to sample voltage for both outlets, and another two pins sample current, one for each outlet. The microcontroller samples the current and voltage values evenly over one half cycle and stores the summation of these samples into memory for transmission back to the control station when polled.

2.1.4 Decipher and Execute Commands

The microcontroller receives commands in the form of opcodes, packaged inside the payload of packets. These opcodes are deciphered upon reception, and the individual commands are executed. Commands include: turn on, turn off, regulate power to a given level, and return voltage and current data.

2.1.5 System for Regulating Power

The microcontroller, additionally, provides a method by which a TRIAC inserted into the power line will be controlled. Dedicating an I/O pin to providing the gating signal on the TRIAC will do this.

2.2 Sampling Networks

The sampling networks allow the microcontroller's ADC to monitor the voltage and current level on the power lines attached to the outlets. These circuits must bring the power down to a level that is below the maximum allowable by the microcontroller to provide a scaled sample of the actual voltage and currents.

2.2.1 Voltage Sampling

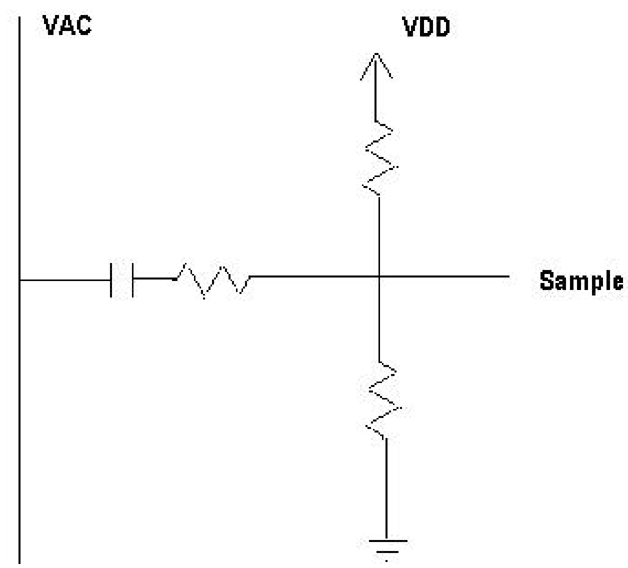


Figure 26: Voltage Sampling

As the diagram above illustrates the voltage sampling network is composed of a system of resistors and a capacitor. The two vertical resistors, R_2 and R_3, are used to keep the sampled voltage between 0V and 3V, fluctuating around the midpoint with the AC input current. The AC current is dropped down from its original value by the R_1 to +/-1.5V, in order use the entire spectrum of voltage values provided by the on chip ADC. The capacitor is used to strip DC voltage off the input voltage.

2.2.2 Current Sampling

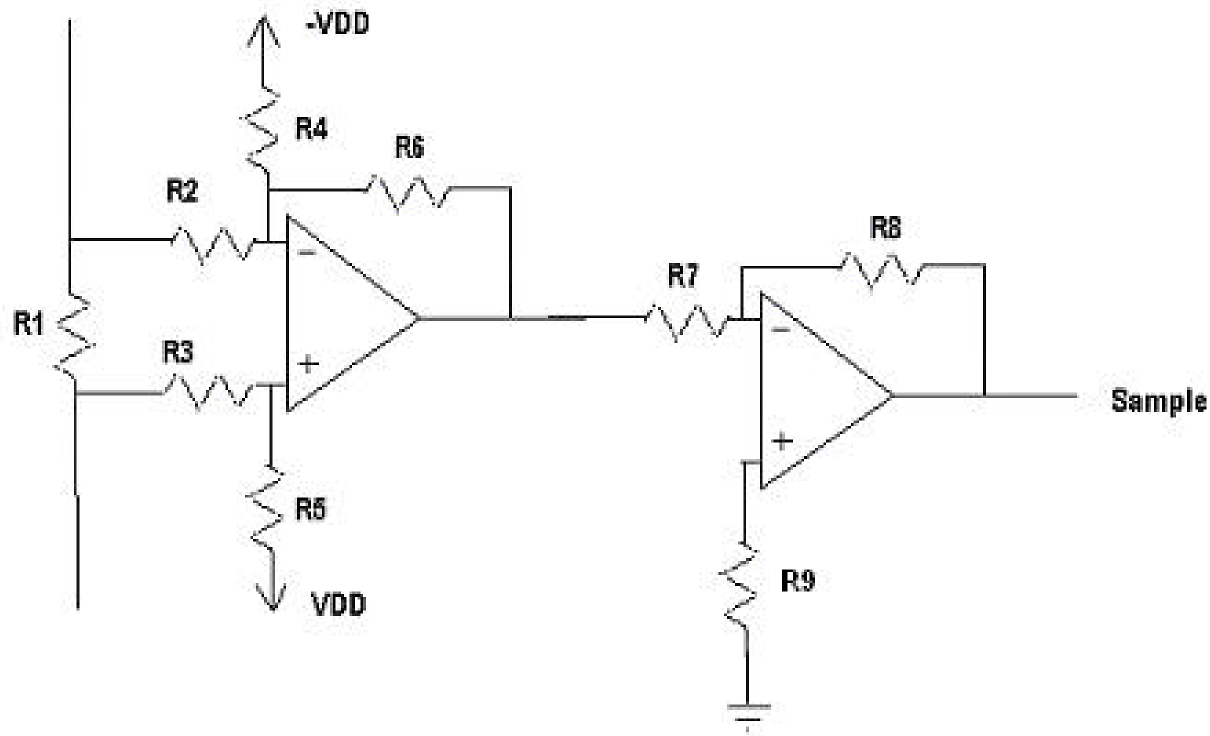


Figure 27: Current Sampling

The Current sampling network is based around a high common-mode difference amplifier balanced for unity gain. However, for our purposes we have added a resistor to the negative input in order to provide a +30X gain. This is necessary because the difference between our positive and negative input pins is only 50mV, effectively lessening the resolution of our ADC by not utilizing the full range of values available. The gain, additionally, allows for maximum resolution because we are again using the two resistors and 3VDC to keep the sampled voltage between 0 and 3 volts; this gain allows for +/-1.5V swing at the sample point.

2.3 Triac Control

A triac is a solid-state switch consisting of a pnpn junction and a gate to control the on/off state of the component. Pictured below is a block diagram and IV characteristics of the device. In our design the triac will be responsible for regulating the current flow to an outlet controlled by our devices.

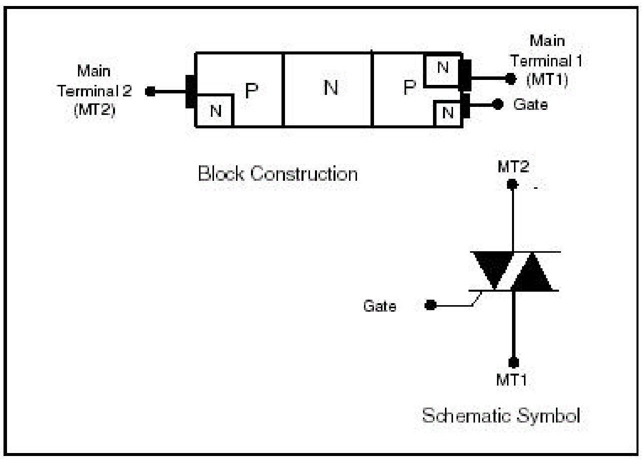


Figure 28: Triac Block Diagram

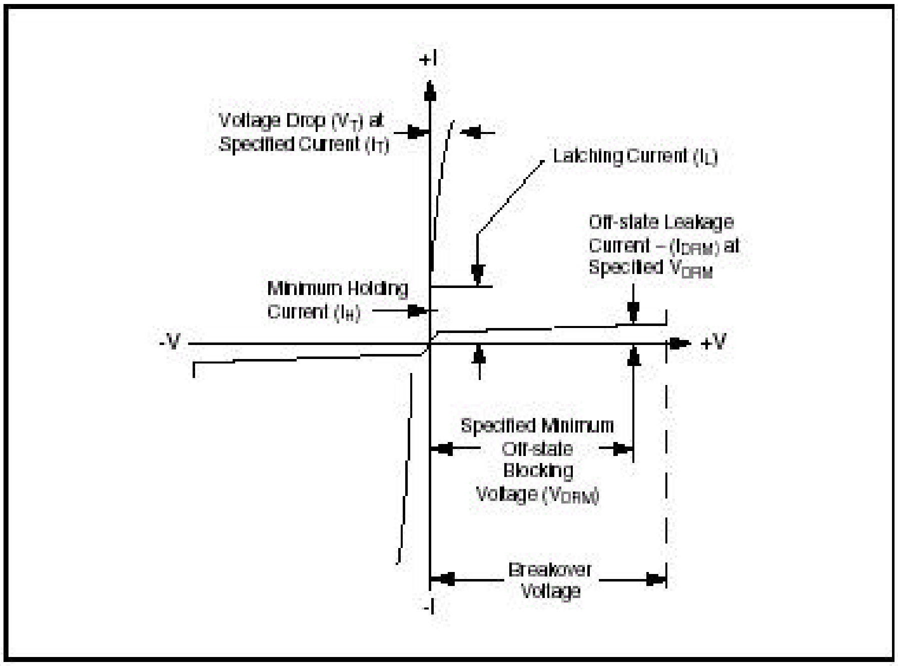


Figure 29: Triac IV Characteristics

The triac needs to be gated on and it will latch on for the remainder of the half cycle, so power regulation is done by firing the triac at a given phase angle and the average power allowed to pass is dropped based on the angle.

3. Design Considerations and Alternatives

There are many microcontroller alternatives available with similar capabilities, but the simplicity and package size of the Cygnal chip were ultimately more desirable. Maxim chips were also considered for our project; they fit the criteria of additions, such as ADC and 8051 or PIC core for a standardized instruction set. However, the cost of the Maxim chips was prohibitive, because they tended to offer a little more horsepower than is necessary for our project. The Motorola DSP56F801 was also considered, and this chip is a little more robust in that is has dual ADC for sampling current and voltage at the same time and component available onboard able to be configured into a power line modem, but the chip is a little more complex and a bit more expensive. Additionally, the development kit offered by Cygnal cost substantially less and offers much more. The Cygnal development environment offer great debug tools over JTAG cable, and excellent control of the internal workings of the chip.

As for the transceiver, there was a more limited selection. The major deciding factor was the development kit. The RFM TR100 development kit contains two transceivers programmed to talk to each other right out of the box, and they can be detached from the base microcontroller board. This allows for the transceiver to be attached easily to our Cygnal chip, to speed up the development process. This transceiver, also, operates in the correct frequency band and has adequate range to satisfy our needs. The cost of this component is be a bit impractical, but the experience may be invaluable.

Senior Design:
Other Issues

1. Ethical

The only ethical issue that develops from our project is a concern for more security. If these wall units were installed into businesses, factories, and homes then a person could potentially breech the system and essentially take control of the wall units overriding the owner's wishes. In this regard the lack of ultimate security could be an ethical issue.

2. Social

The social effect our product would have is added flexibility in personal lives and more efficient businesses. Home users can use this product to turn on appliances at a home before they arrive. Business owners can maximize profit by eliminating excessive electric bills by using the precise amount for their production.

3. Political

There are no political considerations for our product.

4. Economic

Other options that are mentioned previously like using X-10 instead of wireless technology would reduce the cost of each wall unit and subsequently the overall cost of the system. The cost of product development would be significantly lower than the initial presentable due to the cost of development kits.

5. Health and Safety

The only Health and Safety issue would be an appliance that is left on for more time than deemed safe. An example would be a toaster or oven left on would be a threat to a home.

6. Manufacturability

This product has been built and can be built smaller to better fit into a wall outlet. The development time issue is negligible. Potential problems that would arise are getting individual Microsoft Servers for each user and automating the manufacturing for each wall unit.

7. Sustainability

Our wall units are built to handle the maximum allowable current and voltage that is supplied to a home or business by power companies. For there to be any issues, the power supplier would have to alter the standard.

8. Environmental Impact

There are no environmental concerns with our product.

9. Usability

Our product is very user-friendly. Once the wall units are installed it is a step-by-step process to set the system up.

10, Lifelong Learning

The development and research of this product forced us to learn material not covered at Santa Clara University. We did however use most of what we have learned at Santa Clara University on top of the new material in the development of this product.

Senior Design:

Conclusion

Many technologies were incorporated in the development of this project, most of which we had a base for reference but some forced us to start from scratch and completely broaden our scope of knowledge. Among the technologies disciplines included in this system were wireless communications, embedded systems programming, circuit design, ASP pages, WML, multithreaded program development, serial communications, and ODBC commands. All of these components were combined to produce a system that allows an end user to interface and control his/her outlets thought the use of a web interface, ether on a cell phone or computer. Additionally, power totals can be displayed, to allow the user to monitor power consumption across individual outlets.

This project allowed us the opportunity to make design decisions based on many factors including funding, time, component integration, and design requirements. The final design was the result of tough choices in some cases and obvious decisions in others. Additional factors considered in design were quality of development kits in the case of hardware, where robust development kits were able to accelerate the design process tremendously. In particular the Cygnal IDE included with the C8051F300 development kit was indispensable, allowing viewing of registers in the chip, stepping through firmware code, and activating break points. Without this software product development may have been impossible in the allotted amount of time. In addition, the rigid time table forced us to budget time, and make decisions concerning scope based on the issues with time pressure.

The design, although an overall success, could be improved if redone or revised. Ideally, the control box would not be a conventional computer running Windows 2000 Server; instead, it would be some form of box running a cheaper OS capable of supporting multithreaded applications, hosting web pages and a database. The circuitry in the wall units could also be improved to handle different types of loads on the outlets more effectively, and would probably not be implemented with a wireless data link due to issues with cost.

This experience gained while implementing this project was invaluable to our growth as engineers and will be an important reference in the years to come.

Senior Design:

Appendix A

Foundation.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
```

```
                                                      Wall Unit </strong>
                                                <br>
                                          </A>   <strong>Event Sequences </strong>
                                          <br>
                                                 <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                                      Event Sequence </strong>
                                                <br>
                                          </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                                      Event Sequence </strong>
                                                <br>
                                          </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                                      Event Sequence </strong>
                                                <br>
                                          </A>   <strong>Immediate Action </strong>
                                          <br>
                                                
                                          <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                                <br>
                                          </A>
                                          <H2><u>View System</u></H2>
                                             <strong>Wall Units Information</strong><br>
                                                 <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                                      Wall Units</strong><br>
                                          </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                                      Wall Units</strong><br>
                                          </A>   <strong>Event Sequences' Information</strong><br>
                                                 <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                                      Event Sequences</strong><br>
                                          </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                                      Event Sequences</strong></A>
```

```
					<br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
					</td>
				<% ' ====================================================== WELCOME PAGE
=================================== %>
				<td align="center" width="80%" bgcolor="#ccffff">
					<img src="squirrels.jpg"><br>
					<font align="center" size="24" color="#330033" face="Arial">Welcome to your
</font>
					<br>
					<font align="center" size="24" color="#330033" face="Arial">Home System
</font>
				</td>
			</tr>
		</table>
	</body>
```

AllWU2.asp

```
<%@ Language=VBScript %>
<HTML>
	<HEAD>
		<TITLE>Menu Page</TITLE>
		<META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
		<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
		<meta name="publisher" content="ScotchInc">
		<meta name="owner" content="ScotchInc">
		<meta name="author" content="ScotchInc">
		<meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
		<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
	</HEAD>
	<body bgcolor=#0066cc>
		<Table bgcolor=#66ccff width="100%">
			<tr>
				<td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
				<td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
```

```
			</tr>
		</Table>
		<table border="1" bgcolor="#0033cc" width="100%">
			<tr>
				<td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
					   <strong>Wall Units</strong><br>
					       <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
							Wall Unit </strong>
						<br>
					</A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
							Wall Unit </strong>
						<br>
					</A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
							Wall Unit </strong>
						<br>
					</A>   <strong>Event Sequences </strong>
					<br>
					       <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
							Event Sequence </strong>
						<br>
					</A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
							Event Sequence </strong>
						<br>
					</A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
							Event Sequence </strong>
						<br>
					</A>   <strong>Immediate Action </strong>
					<br>
					      
					<stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
						<br>
```

```
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
====================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Information Of Every Wall Unit</strong>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString = "SELECT * FROM Wall_Unit ORDER BY ID"
    Set DB_Results = DB.Execute(SearchString)

      'temporary table just to handle Date problems
      'makeTable = "CREATE TABLE tempDate (id NUMBER, Date CHAR(10))"
      'set try = DB.Execute(makeTable)
      'response.Write(makeTable)
%>

<table border="0" width="60%" height="50%" ID="Table1">
  <tr>
```

```
<table bgcolor=#cccccc border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table2">

  <tr bgcolor=lightblue>
    <td bgcolor=LightBlue><p align="left"><font color="blue" face="Arial"
size="3"><b>ID</b></font></p></td>
    <td bgcolor=LightBlue><p align="left"><font color="blue" face="Arial"
size="3"><b>Name</b></font></p></td>
    <td bgcolor=LightBlue><p align="left"><font color="blue" face="Arial"
size="3"><b>Location</b></font></p></td>
      <td bgcolor=LightBlue><p align="left"><font color="blue" face="Arial" size="3"><b>Power
Graph</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof
            set thisID = DB_Results("ID")

            'get the date in the Power_Totals table
            getDate = "SELECT * FROM Voltage_Total WHERE WUID = " & thisID & " ORDER BY ID"
            set gotDate = DB.Execute(getDate)
            set thisDate = gotDate("Date")

      %>
  <tr bgcolor=white>
    <td bgcolor=White><p align="left"><font color="#000000" face="Arial" size="3"><%response.Write (thisID)
%></font></p></td>
    <td bgcolor=White><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td bgcolor=White><p align="left"><font color="#000000" face="Arial"
size="3"><%=DB_Results("Location")%> </font></p></td>
      <td bgcolor=White><p align="left"><a href=graphIWUDate.asp?WUID=<%response.Write
(thisID)%>>Isolated</a>   or  <a
href=graphAllWU.asp?WUID=<%response.Write(thisID)%>>Related</a></p></td>
</tr>

      <%
            DB_Results.MoveNext
      wend
      %>
</table>
```

```
<br><br><br>
<a href=foundation.asp><strong>BACK</strong></a></td>
                  </tr>
            </table>
      </body>
```

Errorpage.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr valign=top>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
```

```
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
```

```
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Information Of Wall Units With ID = </strong>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

   Set DB_Results = DB.Execute("SELECT * FROM ErrorLog")
%>
<table border="0" width="100%" height="100%" ID="Table1">
  <tr>
    <td colspan="2" height="10" align="middle">

  <br></td>
  </tr>
 <tr>
    <td width="120" valign="top">

    </td>
    <td width="100%" valign="top">

<div align="left">
<table border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table2">

  <tr bgcolor=lightblue>
    <td>
      <p align="left"><font color="blue" face="Arial" size="3"><b>Error Code</b></font></p></td>
    <td>
      <p align="left"><font color="blue" face="Arial" size="3"><b>Error Text</b></font></p></td>
    <td>
```

```
      <p align="left"><font color="blue" face="Arial" size="3"><b>Error Date</b></font></p>  </td>
   </tr>
      <%    while not DB_Results.eof  %>
  <tr bgcolor=white>
    <td>
      <p align="left"><font color="#000000" face="Arial"
size="3"><%=DB_Results("Error_Code")%></font></p></td>
    <td>
      <p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("Error_Text")%>
</font></p></td>
    <td>
      <p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("Error_Date")%>
</font></p></td>
</tr>

      <%
            DB_Results.MoveNext
      wend
      %>
</table>
<br><br><br>
<a href=foundation.asp><srtong>BACK</srtong></a>
</td>
                  </tr>
            </table>
      </body>
```

ES_AddInsert2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
```

```
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%" ID="Table3">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
```

```
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">

                        <%

' get data from registration form
      set fName = Request.Form("SName")
```

```
      set fSDm = Request.Form("MonthSD")
'     response.Write fSDm
      set fSDd = Request.Form("DaySD")
'     response.Write fSDd
      set fSDy = Request.Form("YearSD")
'     response.Write fSDy
      set fEDm = Request.Form("MonthED")
    set fEDd = Request.Form("DayED")
    set fEDy = Request.Form("YearED")
    set fSTh = Request.Form("HourST")
    set fSTm = Request.Form("MinST")
    set fAM  = Request.Form("AM")
    set fPM  = Request.Form("PM")
    set fDRY = Request.Form("Yes")
    set fDRN = Request.Form("No")

'     response.Write (fDRY)

    set fS   = Request.Form("Sun")
    set fM   = Request.Form("Mon")
    set fT   = Request.Form("Tues")
    set fW   = Request.Form("Wed")
    set fR   = Request.Form("Thur")
    set fF   = Request.Form("Fri")
      set fSat = Request.Form("Sat")

'     response.Write (fPM)
'     response.write (fS)
'     response.Write (fM)
'     response.write (fT)
'     response.Write (fW)
'     response.write (fR)
'     response.Write (fF)
'     response.Write (fSat)

Dim thisOpcode
thisOpcode = 0
```

```
If (fDRY="DRY") Then
'      response.Write "it gets here"
        thisOpcode = thisOpcode + &H80
end If

If (fS = "Sun") Then
      thisOpcode = thisOpcode + &H40
end If

If (fM = "Mon") Then
      thisOpcode = thisOpcode + &H20
end If

If (fT = "Tues") Then
      thisOpcode = thisOpcode + &H10
end If

If (fW = "Wed") Then
      thisOpcode = thisOpcode + &H08
end If

If (fR = "Thur") Then
      thisOpcode = thisOpcode + &H04
end If

If (fF = "Fri") Then
      thisOpcode = thisOpcode + &H02
end If

If (fSat = "Sat") Then
      thisOpcode = thisOpcode + &H01
end If

If (fPM = "pm") Then
      fSTh = fSTh +12
end If

'response.Write fAM
```

```
%><br><%
'response.Write fPM

' Check for invalid Info----------------------------------------------
' DO THIS NEXT QUARTER
'
' Display the Wall Units and the related Op Codes

      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      ' Insert values in database adding the ES
    InsertString = "INSERT INTO event_sequence (StartDate, Name, StartTime, DailyRepeat, EndDate)"
    InsertString = InsertString & " VALUES (" &"'" &fSDm & "/"& fSDd &"/"& fSDy & "',"
    InsertString = InsertString & "'" & fName & "', "
    InsertString = InsertString & "'" & fSTh & ":" & fSTm &"',"
    InsertString = InsertString & "'" & thisOpcode & "', "
    InsertString = InsertString &  "'"&fEDm & "/"& fEDd &"/"& fEDy & "')"
    'response.write InsertString  & "<BR>"
      DB.Execute(InsertString)

      'get the ID of the event Sequence
      SearchString = "Select * FROM event_sequence WHERE ID ="
      SearchString = SearchString & "(Select MAX(ID) FROM event_sequence)"
      Set DB_Results = DB.Execute(SearchString)
      'Set thisID = DB_Results("ID")

      ' Insert values in database adding the Daily Repeat days and it's opcode
      InsertString2 = "INSERT INTO IA_effects_WU (WUID, offset) VALUES "
      InsertString2 = InsertString2 & "(" & "NULL, " & thisID & ")"
      'DB.Execute(InsertString2)


%>

<strong>THIS IS THE NEWLY ADDED EVENT SEQUENCE...</strong>
<br><br><br><br><br><br>
<table border="0" width="100%" height="100%" ID="Table1">
```

```
  <tr>
    <td colspan="2" height="10" align="middle">

  <br></td>
  </tr>
 <tr>
    <td width="120" valign="top"></td>
    <td width="100%" valign="top">

    <div align="left">
<table border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table2">
  <tr bgcolor="#ffbe9c">
    <td><p align="left"><font color="red" face="Arial" size="3"><b>ID</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Name</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Start Date</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>End Date</b></font></p>  </td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Start Time</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof
      Dim thisID
      thisID =DB_Results("ID")

      %>
  <tr bgcolor=white>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartDate")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("EndDate")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartTime")%>
</font></p></td>
</tr>

      <%
            DB_Results.MoveNext
      wend
```

```
      %>
</table>
<br><br><br><br><br><br><br>
<a href=\MS_ES_pickWUs2.asp?eID=<%response.Write thisID%>><strong>Select Wall Units</strong></a>

</div>
    </td>
  </tr>
</table>




                        </td>
                  </tr>
            </table>
      </body>
```

ES_DeleteInsert2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
```

```
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
```

```
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                              </td>
                        <% ' ======================================================= WELCOME PAGE
==================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                  <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

' get data from registration form
    set fID = Request.QueryString("ESID")

' update values in database
    InsertString = "DELETE FROM event_sequence WHERE ID = " & fID
    DB.Execute(InsertString)

%>
<h3>Deletion Successful!</h3>
<br>
```

```
<br>
<br>
<a href=MS_ES_Delete2.asp><strong>BACK</strong></a>

                        </td>
                  </tr>
            </table>
      </body>
```

ES_EditInsert2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
```

```
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
```

```
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                              </td>
                        <% ' ======================================================= WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      thisID = Request.QueryString("EVID")

'get the information without having it sent here
      SearchString = "SELECT * FROM event_sequence Where ID="&thisID

    'response.write SearchString  & "<BR>"
      Set DB_Results = DB.Execute(SearchString)

      set dr = DB_Results("DailyRepeat")

      Dim n
      n = 0

      dim drt
      dim sun
      dim mon
      dim tues
      dim wed
      dim thur
```

```
dim fri
dim sat

dim tdrt
dim tsun
dim tmon
dim ttues
dim twed
dim tthur
dim tfri
dim tsat

drt = 0
sun =0
mon =0
tues=0
wed =0
thur=0
fri =0
sat =0

dim moddr
moddr = dr -254
'response.Write(moddr)

Dim subNum
subNum =0


if ((dr MOD 128)>0) then
      subNum = 128
      drt = 1
end if
if ((dr-subNum)>0) AND(((dr - subNum) MOD 64)>0) then
      subNum = subNum +64
      sun = 1
end if
if ((dr-subNum)>0) AND(((dr - subNum) MOD 32)>0) then
```

```
            subNum = subNum +32
            mon = 1
      end if
      if ((dr-subNum)>0) AND(((dr - subNum) MOD 16)>0) then
            subNum = subNum +16
            tues = 1
      end if
      if ((dr-subNum)>0) AND(((dr - subNum) MOD 8)>0) then
            subNum = subNum +8
            wed = 1
      end if
      if ((dr-subNum)>0) AND(((dr - subNum) MOD 4)>0) then
            subNum = subNum +4
            thur = 1
      end if
      if ((dr-subNum)>0) AND(((dr - subNum) MOD 2)>0) then
            subNum = subNum +2
            fri = 1
      end if
      if ((dr-subNum)=1) then
            subNum = subNum +1
            sat = 1
      end if




%>

<strong>COMPLETE FORM TO EDIT THE EVENT SEQUENCE...</strong>
<br><br><br><br>

<table border="0" width="100%" ID="Table1">
  <tr>
    <td width="120" valign="top"></td>
    <td width="100%" valign="top">
<form action="ES_EditUpdate2.asp?ID=<%response.Write thisID%>" id="WallUnitEntry" method="post"
name="Form1">
```

```
      </form>
      <div align="left"><table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table2">
        <tr>
          <td><div align="right"><p> Event ID  </td>
          <td><div align="left"><p><strong><%=DB_Results("ID")%></strong></td>
        </tr>
        <tr>
        <td><div align="right"><p> Event Name  </td>
                        <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input
value="<%=DB_Results("Name")%>" name="WUName" size="30" ID="Text1"></font></td>
        </tr>
        <tr>
          <td><div align="right"><p> Event Start Time  </td>
                        <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input
value="<%=DB_Results("StartTime")%>" name="WUST" size="30" ID="Text2"></font></td>
        </tr>
        <tr>
          <td><div align="right"><p> Event Start Date  </td>
                        <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input
value="<%=DB_Results("StartDate")%>" name="WUSD" size="30" ID="Text3"></font></td>
        </tr>
        <tr>
          <td><div align="right"><p> Event End Date  </td>
                        <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input
value="<%=DB_Results("EndDate")%>" name="WUED" size="30" ID="Text4"></font></td>
        </tr>
        <tr>
          <td><div align="right"><p> Event Daily Repeat?  </td>

            <td><div align="left"><p><font color="#000000" face="Arial" size="3">
                              <input type="checkbox" name="DR" <%
                        If drt = "1" Then
                              Response.Write " checked=""checked"""
                        end If %>
             size="30" ID="Text5" value="ON"></font></td>

        </tr>
        <tr>
```

```
          <td><div align="right"><p> Days To Repeat  </td>
          <td>Sun<INPUT type="checkbox" ID="Checkbox11" NAME="Sun" <%
                        If sun = "1" Then
                              Response.Write " checked=""checked"""
                        end If %> value="ON">     Mon<INPUT type="checkbox" ID="Checkbox3"
NAME="Mon" <%
                        If mon = "1" Then
                              Response.Write " checked=""checked"""
                        end If %> value="ON">     Tues<INPUT type="checkbox" ID="Checkbox6"
NAME="Tues" <%
                        If tues = "1" Then
                              Response.Write " checked=""checked"""
                        end If %> value="ON">    Wed<INPUT type="checkbox" ID="Checkbox7"
NAME="Wed" <%
                        If wed = "1" Then
                              Response.Write " checked=""checked"""
                        end If %> value="ON">    Thur<INPUT type="checkbox" ID="Checkbox8"
NAME="Thur" <%
                        If thur = "1" Then
                              Response.Write " checked=""checked"""
                        end If %> value="ON">    Fri<INPUT type="checkbox" ID="Checkbox9"
NAME="Fri" <%
                        If fri = "1" Then
                              Response.Write " checked=""checked"""
                        end If %> value="ON">    Sat<INPUT type="checkbox" ID="Checkbox10"
NAME="Sat" <%
                        If sat = "1" Then
                              Response.Write " checked=""checked"""
                        end If %> value="ON"></td>
        </tr>
        <%
                  set allWallUnits = DB.Execute("SELECT * FROM ES_effects_WU WHERE ESID = " & thisID)
                  set getWallUnits = DB.Execute("SELECT * FROM Wall_Unit WHERE ID = (SELECT WUID FROM
ES_effects_WU WHERE ESID = " & thisID & ")")

                  dim op
                  'op = getWallUnits("OPCODE")
        %>
```

```
        </table>
        <table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table3">
      <tr>
                  <td><div align="right"><p> Wall Unit Operations  </td>
            <tr>
            <%
                  while not getWallUnits.eof

            %>
                  <td align=right bgcolor=white> <%response.Write (getWallUnits("Name"))%></td>
                  <td align=right>  Turn On<input align=right type=radio name="turnOn" value=125
ID="Radio2"> </td>
                  <td align=right>  Turn Off<input align=right type=radio name="turnOff" value=100
ID="Radio3"> </td>
                  <td align=right>  Low<input align=right type=radio name="c" value=125 ID="Checkbox1">
</td>
                  <td align=right>  Low Medium<input align=right type=radio name="c" value=100
ID="Checkbox2"> </td>
                  <td align=right>  Medium<input align=right type=radio name="c" value=75 ID="Radio1">
</td>
                  <td align=right>  High Medium<input align=right type=radio name="c" value=50
ID="Checkbox4"> </td>
                  <td align=right>  High<input align=right type=radio name="c" value=10 ID="Checkbox5">
</td>
            </tr>
            <%
                  getWallUnits.MoveNext
                  wend
            %>
    </tr>
        <tr>
        </table>
        <table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table4">

          <td><div align="center"><center><p><input id="reset1" name="reset1" type="reset"
value="Reset"></td>
          <td><div align="center"><center><p><input id="submit1" name="submit1" type="submit"
value="Submit"></td>
```

```
          </tr>
        </table>
        </div>
      </form>
      </td>
    </tr>
  </table>
  </td>
                  </tr>
            </table>
      </body>
```

ES_EditUpdate2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
```

```
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
```

```
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                              </td>
                        <% ' ====================================================== WELCOME PAGE
==================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                        <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      'tempID = Request.QueryString("WUID")
      set thisID = Request.QueryString("ID")
'     response.Write(thisID)

' get data from registration form
      set fName = Request.Form("WUName")
      set sdm = Request.Form("WUSD")

Dim   fSDm
Dim fSDd
Dim fSDy
fSDm = month(sdm)
fSDd = day(sdm)
fSDy = year(sdm)

      set ed = Request.Form("WUED")
```

```
Dim   fEDm
Dim fEDd
Dim fEDy
fEDm = month(ed)
fEDd = day(ed)
fEDy = year(ed)

      set st = Request.Form("WUST")

Dim   fSTh
Dim fSTm
fSTh = hour(st)
fSTm = minute(st)

    set fAM  = Request.Form("AM")
    set fPM  = Request.Form("PM")
    set fDRY = Request.Form("DR")
'    set fDRN = Request.Form("No")

'     response.Write (fDRY)

    set fS   = Request.Form("Sun")
    set fM   = Request.Form("Mon")
    set fT   = Request.Form("Tues")
    set fW   = Request.Form("Wed")
    set fR   = Request.Form("Thur")
    set fF   = Request.Form("Fri")
      set fSat = Request.Form("Sat")

'     response.Write (fPM)
'     response.Write (fDRY)
'     response.write (fS)
'     response.Write (fM)
'     response.write (fT)
'     response.Write (fW)
'     response.write (fR)
'     response.Write (fF)
```

```
'     response.Write (fSat)

Dim thisOpcode
thisOpcode = 0

If (fDRY="on") Then
'     response.Write "it gets here"
       thisOpcode = thisOpcode + &H80
end If

If (fS = "on") Then
      thisOpcode = thisOpcode + &H40
end If

If (fM = "on") Then
      thisOpcode = thisOpcode + &H20
end If

If (fT = "on") Then
      thisOpcode = thisOpcode + &H10
end If

If (fW = "on") Then
      thisOpcode = thisOpcode + &H08
end If

If (fR = "on") Then
      thisOpcode = thisOpcode + &H04
end If

If (fF = "on") Then
      thisOpcode = thisOpcode + &H02
end If

If (fSat = "on") Then
      thisOpcode = thisOpcode + &H01
end If
```

```
If (fPM = "on") Then
      fSTh = fSTh +12
end If

'response.Write fAM
%><br><%
'response.Write fPM

' Check for invalid Info----------------------------------------------
' DO THIS NEXT QUARTER
'
' Display the Wall Units and the related Op Codes

      ' Update the ES
      InsertString = "UPDATE event_sequence SET Name ="&"'"& fName&"'" & " WHERE ID="&thisID
      InsertString2 = "UPDATE event_sequence SET StartDate ="&"'" &fSDm & "/"& fSDd &"/"& fSDy & "'" & "
WHERE ID="&thisID
      InsertString3 = "UPDATE event_sequence SET StartTime ="& "'" & fSTh & ":" & fSTm &"'" & " WHERE
ID="&thisID
      InsertString4 = "UPDATE event_sequence SET DailyRepeat ="&"'"  & thisOpcode & "'" & " WHERE
ID="&thisID
      '     response.Write(InsertString4)
      InsertString5 = "UPDATE event_sequence SET EndDate ="& "'" &fEDm & "/"& fEDd &"/"& fEDy & "'" & "
WHERE ID="&thisID

      DB.Execute(InsertString)
    DB.Execute(InsertString2)
    DB.Execute(InsertString3)
      DB.Execute(InsertString4)
    DB.Execute(InsertString5)

      'get the ID of the event Sequence
      SearchString = "Select * FROM event_sequence WHERE ID =" & thisID
      'Set DB_Results = DB.Execute(SearchString)
      'Set thisID = DB_Results("ID")

      ' Insert values in database adding the Daily Repeat days and it's opcode
      'InsertString2 = "INSERT INTO IA_effects_WU (WUID, offset) VALUES "
```

```
	'InsertString2 = InsertString2 & "(" & "NULL, " & thisID & ")"
	'DB.Execute(InsertString2)


%>


<h3>Edit Successful!</h3>
<br><br><br>
<a href=foundation.asp><strong>BACK</strong></a></td>
			</tr>
		</table>
	</body>
```

EventIDlist2.asp

```
<%@ Language=VBScript %>
<HTML>
	<HEAD>
		<TITLE>Menu Page</TITLE>
		<META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1) "http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l 0))'>
		<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
		<meta name="publisher" content="ScotchInc">
		<meta name="owner" content="ScotchInc">
		<meta name="author" content="ScotchInc">
		<meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
		<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
	</HEAD>
	<body bgcolor=#0066cc>
		<Table bgcolor=#66ccff width="100%">
			<tr>
				<td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
				<td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
			</tr>
		</Table>
```

```
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
```

```
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                              </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString1 = "SELECT DISTINCT ID FROM event_sequence ORDER BY ID"
'    response.write SearchString  & "<BR>"
    Set DB_Results = DB.Execute(SearchString1)

%>

<strong>Select An Event Sequence ID...</strong>
<br><br>Click on the different cases below:<BR><br><br>

<table border="0" width="50%" height="50%" ID="Table1">
  <tr>
<div align="center">
<table border="1" cellPadding="1" cellSpacing="2" width="50%" ID="Table2">

  <tr bgcolor=#ffbe9c>
```

```
    <td>
      <p align="center"><font color="red" face="Arial" size="3"><b>ID Numbers</b></font></p></td>

  </tr>
      <%    while not DB_Results.eof  %>
  <tr bgcolor=white>
    <td>
            <p align="center"><A
href=\EventsID2.asp?thisID=<%=DB_Results("ID")%>><strong><%=DB_Results("ID")%></strong></A></p>
    </td>
  </tr>

      <%
            DB_Results.MoveNext
      wend
      %>
</table>

</div>
    </td>
  </tr>
</table>
</td>
                  </tr>
            </table>
      </body>
```

EventINameList2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
```

```
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
```

```
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                              </td>
                        <% ' ================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString1 = "SELECT DISTINCT ID FROM event_sequence ORDER BY ID"
'    response.write SearchString  & "<BR>"
```

```
    Set DB_Results = DB.Execute(SearchString1)

%>

<strong>Select An Event Sequence ID...</strong>
<br><br>Click on the different cases below:<BR><br><br>

<table border="0" width="50%" height="50%" ID="Table1">
  <tr>
<div align="center">
<table border="1" cellPadding="1" cellSpacing="2" width="50%" ID="Table2">

  <tr bgcolor=#ffbe9c>
    <td>
      <p align="center"><font color="red" face="Arial" size="3"><b>ID Numbers</b></font></p></td>

  </tr>
      <%    while not DB_Results.eof  %>
  <tr bgcolor=white>
    <td>
            <p align="center"><A
href=\EventsID2.asp?thisID=<%=DB_Results("ID")%>><strong><%=DB_Results("ID")%></strong></A></p>
    </td>
  </tr>

      <%
            DB_Results.MoveNext
      wend
      %>
</table>

</div>
    </td>
  </tr>
</table>
</td>
                  </tr>
            </table>
```

```
      </body>
```

Events2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr valign=top>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
```

```
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
```

```
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
====================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Information Of Every Event Sequence</strong>
<br>
<br>
<font color="red" face="Arial" size="3"><strong>Red = Event Sequence Information</strong>
<br>
<font color="blue" face="Arial" size="2"><strong>Blue = Corresponding Wall Unit Information</strong>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString = "SELECT * FROM event_sequence "
    Set DB_Results = DB.Execute(SearchString)

    Set DB2 = Server.CreateObject("ADODB.connection")
      DB2.Open "CB"

%>
<table border="0" width="100%" height="100%" ID="Table1">
  <tr>
    <td colspan="2" height="10" align="middle">
  <br></td>
  </tr>
 <tr>
    <td width="120" valign="top">
    </td>
    <td width="100%" valign="top">
<div align="left">
<table bgcolor="#cccccc" border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table2">
```

```
      <%    while not DB_Results.eof      '================================= GO THROUGH EVENT SEQUENCE
TABLE
            set esID = DB_Results("ID")
      %>
      <tr></tr>
      <tr></tr>
      <tr></tr>
      <tr></tr>
      <tr></tr>
      <tr></tr>
  <tr>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>ID</b></font></p></td>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>Start
Date</b></font></p></td>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>Name</b></font></p></td>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>Start
Time</b></font></p></td>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>Daily
Repeat?</b></font></p>  </td>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>End
Date</b></font></p></td>
  </tr>
<%    Dim dr
      If (DB_Results("DailyRepeat") = 0) Then
            dr = "No"
      Else
            dr = "Yes"
      end IF
      %>
  <tr>
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial" size="3"><b><%response.Write
esID%></b></font></p></td>
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial"
size="3"><b><%=DB_Results("StartDate")%></b></font></p></td>
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial"
size="3"><b><%=DB_Results("Name")%></b> </font></p></td>
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial"
size="3"><b><%=DB_Results("StartTime")%></b> </font></p></td>
```

```
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial" size="3"><b><%response.Write
dr%></b> </font></p></td>
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial"
size="3"><b><%=DB_Results("EndDate")%></b> </font></p></td>
</tr>
      <%
            tempString = "SELECT * FROM ES_effects_WU WHERE ESID ="&esID & " ORDER BY WUID"
            set temp_Results = DB2.Execute(tempString)

      %>
      <tr></tr>
      <tr></tr>
      <tr>
            <td></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial" size="2"><b>Involved
Wall Units</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>ID</b></font></p></td>
          <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Name</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Location</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Operation</b></font></p></td>
      </tr>

<%
            while not temp_Results.eof

            set tempWUID = temp_Results("WUID")

            SearchString2 = "SELECT * FROM Wall_Unit WHERE ID =" & tempWUID
            set DB_Results2 = DB2.Execute(SearchString2)

                        Dim op
                        set t = temp_Results("OPCODE")
                        If (t = 2) Then
                              op = "Turn On"
```

```
                        end If
                        If (t = 3) Then
                              op = "Turn Off"
                        end If
                        If (t = 4) Then
                              op = "Raise Power"
                        end If
                        If (t = 5) Then
                              op = "Lower Off"
                        end If
                        If (t = 0) Then
                              op = "Report"
                        end If

                  %>
            <tr>
                  <td></td>
                  <td></td>
                  <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("ID")%></font></p></td>
                  <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("Name")%> </font></p></td>
                  <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("Location")%> </font></p></td>
                  <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%response.Write op%> </font></p></td>
            </tr>
                  <%
                        temp_Results.MoveNext
                        wend
                  %>

      <%
            DB_Results.MoveNext
            wend
      %>
</table>
<br>
```

```
<br>
<br>
      <a href=foundation.asp><font color=black><strong>BACK</strong></font></a>   </td>
                  </tr>
            </table>
      </body>
```

EventsID2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                    Wall Unit </strong>
```

```
                              <br>
                        </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                    Wall Unit </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                    Wall Unit </strong>
                              <br>
                        </A>   <strong>Event Sequences </strong>
                        <br>
                               <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                    Event Sequence </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                    Event Sequence </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                    Event Sequence </strong>
                              <br>
                        </A>   <strong>Immediate Action </strong>
                        <br>
                              
                        <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                              <br>
                        </A>
                        <H2><u>View System</u></H2>
                           <strong>Wall Units Information</strong><br>
                               <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                    Wall Units</strong><br>
                        </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                    Wall Units</strong><br>
```

```
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
====================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Information Of Event Sequences With ID =
</strong>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      pageID = Request.QueryString("thisID")
      response.Write(pageID)

      Set DB_Results = DB.Execute("SELECT * FROM event_sequence WHERE ID = " & pageID)
%>
<table border="0" width="100%" height="100%" ID="Table1">
  <tr>

<div align="left">
<table border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table2">

  <tr bgcolor=#ffbe9c>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>ID</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Start Date</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Name</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Start Time</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Daily Repeat?</b></font></p>  </td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>End Date</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof  %>
```

```
  <tr bgcolor=white>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td><p align="left"><font color="#000000" face="Arial"
size="3"><%=DB_Results("StartDate")%></font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartTime")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("DailyRepeat")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("EndDate")%>
</font></p></td>
</tr>
<%
            tempString = "SELECT * FROM ES_effects_WU WHERE ESID ="& pageID & " ORDER BY WUID"
            set temp_Results = DB.Execute(tempString)

      %>
      <tr></tr>
      <tr></tr>
      <tr>
            <td></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial" size="2"><b>Involved
Wall Units</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>ID</b></font></p></td>
          <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Name</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Location</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Operation</b></font></p></td>
      </tr>

<%
            while not temp_Results.eof

            set tempWUID = temp_Results("WUID")
```

```
            SearchString2 = "SELECT * FROM Wall_Unit WHERE ID =" & tempWUID
            set DB_Results2 = DB.Execute(SearchString2)

                        Dim op
                        set t = temp_Results("OPCODE")
                        If (t = 2) Then
                              op = "Turn On"
                        end If
                        If (t = 3) Then
                              op = "Turn Off"
                        end If
                        If (t = 4) Then
                              op = "Raise Power"
                        end If
                        If (t = 5) Then
                              op = "Lower Off"
                        end If
                        If (t = 0) Then
                              op = "Report"
                        end If

                        If (DB_Results2("ID") = "") Then
                        %>
                        <tr>
                              <td></td>
                              <td></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2">There </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2">Are No </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2">Wall  </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2">Units</font></p></td>
                        </tr>

                        <%
```

```
                        Else
                        %>
                        <tr>
                              <td></td>
                              <td></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("ID")%></font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("Name")%> </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("Location")%> </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%response.Write op%> </font></p></td>
                        </tr>
                        <%
                        end If
                        temp_Results.MoveNext
                        wend
                  %>

      <%
            DB_Results.MoveNext
            wend
      %>
</table>

</div>
    </td>
  </tr>
</table>
</td>
                  </tr>
            </table>
      </body>
```

EventsName2.asp

```
<%@ Language=VBScript %>
```

```
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
```

```
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
```

```
                              <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Information Of Event Sequences With Name = </strong>

                                    <%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

  '  Dim pageName
    pageName = Request.QueryString("thisName")
   response.Write pageName

   'SearchString = "SELECT * FROM event_sequence WHERE Name="& pageName
   ' response.write SearchString  & "<BR>"
   Set DB_Results = DB.Execute("SELECT * FROM event_sequence WHERE Name = "&"'"& pageName&"'")
%>
<table border="0" width="100%" height="100%" ID="Table1">
  <tr>

<div align="left">
<table border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table2">

  <tr bgcolor=#ffbe9c>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>ID</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Start Date</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Name</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Start Time</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Daily Repeat?</b></font></p>  </td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>End Date</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof  %>
  <tr bgcolor=white>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td><p align="left"><font color="#000000" face="Arial"
size="3"><%=DB_Results("StartDate")%></font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartTime")%>
</font></p></td>
```

```
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("DailyRepeat")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("EndDate")%>
</font></p></td>
</tr>
<%
            set getID = DB.Execute("SELECT * FROM event_sequence WHERE Name = " & "'" &pageName & "'")
            'response.Write(getID("ID"))

            tempString = "SELECT * FROM ES_effects_WU WHERE ESID ="& getID("ID") & " ORDER BY WUID"
            'response.Write (tempString)
            set temp_Results = DB.Execute(tempString)

      %>
      <tr></tr>
      <tr></tr>
      <tr>
            <td></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial" size="2"><b>Involved
Wall Units</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>ID</b></font></p></td>
          <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Name</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Location</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Operation</b></font></p></td>
      </tr>

<%
            while not temp_Results.eof

            set tempWUID = temp_Results("WUID")

            SearchString2 = "SELECT * FROM Wall_Unit WHERE ID =" & tempWUID
            set DB_Results2 = DB.Execute(SearchString2)
```

```
                        Dim op
                        set t = temp_Results("OPCODE")
                        If (t = 2) Then
                              op = "Turn On"
                        end If
                        If (t = 3) Then
                              op = "Turn Off"
                        end If
                        If (t = 4) Then
                              op = "Raise Power"
                        end If
                        If (t = 5) Then
                              op = "Lower Off"
                        end If
                        If (t = 0) Then
                              op = "Report"
                        end If

                        If (DB_Results2("ID") = "") Then
                        %>
                        <tr>
                              <td></td>
                              <td></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2">There </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2">Are No </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2">Wall  </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2">Units</font></p></td>
                        </tr>

                        <%
                        Else
                        %>
                        <tr>
                              <td></td>
```

```
                              <td></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("ID")%></font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("Name")%> </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("Location")%> </font></p></td>
                              <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%response.Write op%> </font></p></td>
                        </tr>
                        <%
                        end If
                        temp_Results.MoveNext
                        wend
                  %>

      <%
            DB_Results.MoveNext
            wend
      %>
</table>

</div>
    </td>
  </tr>
</table></td>
                  </tr>
            </table>
      </body>
```

graphAllWU.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
```

```
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
```

```
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
==================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              graph of stacked bar or pie graph
                              <%
                                    set var = Request.QueryString("WUID")
```

```
                              %>

                              <br>
<br>
<br>
<a href=AllWU2.asp><strong>BACK</strong></a>
                              </td>
                  </tr>
            </table>
      </body>
```

graphIWU.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor="#0066cc">
            <Table bgcolor="#66ccff" width="100%">
                  <tr>
                        <td bgcolor="#66ccff"><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor="#66ccff" align="right" height="12"><IMG alt="flag"
src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
```

```
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
```

```
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ===================================================== WELCOME PAGE
==================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              graph of single wall unit...
                              <% ' ====================================================== graph
==================================== %>
                              <%
Sub ShowChart(ByRef aValues, ByRef aLabels, ByRef strTitle, ByRef strXAxisLabel, ByRef strYAxisLabel)
      ' Some user changable graph defining constants
      ' All units are in screen pixels
      Const GRAPH_WIDTH  = 450  ' The width of the body of the graph
      Const GRAPH_HEIGHT = 250  ' The heigth of the body of the graph
      Const GRAPH_BORDER = 5    ' The size of the black border
      Const GRAPH_SPACER = 2    ' The size of the space between the bars

      ' Debugging constant so I can eaasily switch on borders in case
      ' the tables get messed up.  Should be left at zero unless you're
      ' trying to figure out which table cells doing what.
      Const TABLE_BORDER = 0
      'Const TABLE_BORDER = 10

      ' Declare our variables
      Dim I
```

```
      Dim iMaxValue
      Dim iBarWidth
      Dim iBarHeight

      ' Get the maximum value in the data set
      iMaxValue = 0
      For I = 0 To UBound(aValues)
            If iMaxValue < aValues(I) Then iMaxValue = aValues(I)
      Next 'I
      'Response.Write iMaxValue ' Debugging line


      ' Calculate the width of the bars
      ' Take the overall width and divide by number of items and round down.
      ' I then reduce it by the size of the spacer so the end result
      ' should be GRAPH_WIDTH or less!
      iBarWidth = (GRAPH_WIDTH \ (UBound(aValues) + 1)) - GRAPH_SPACER
      'Response.Write iBarWidth ' Debugging line


      ' Start drawing the graph
      %>
                              <TABLE BORDER="<%= TABLE_BORDER %>" CELLSPACING="0" CELLPADDING="0"
ID="Table1">
                                    <TR>
                                          <TD COLSPAN="3" ALIGN="center"><H2><%= strTitle %></H2>
                                          </TD>
                                    </TR>
                                    <TR>
                                          <TD VALIGN="center"><B><%= strYAxisLabel %></B></TD>
                                          <TD VALIGN="top">
                                                <TABLE BORDER="<%= TABLE_BORDER %>" CELLSPACING="0"
CELLPADDING="0" ID="Table2">
                                                      <TR>
                                                            <TD ROWSPAN="2"><IMG SRC="./images/spacer.gif"
BORDER="0" WIDTH="1" HEIGHT="<%= GRAPH_HEIGHT %>"></TD>
                                                            <TD VALIGN="top" ALIGN="right"><%= iMaxValue
%> </TD>
```

```
                                        </TR>
                                        <TR>
                                              <TD VALIGN="bottom" ALIGN="right">0 </TD>
                                        </TR>
                                  </TABLE>
                            </TD>
                            <TD>
                                  <TABLE BORDER="<%= TABLE_BORDER %>" CELLSPACING="0"
CELLPADDING="0" ID="Table3">
                                        <TR>
                                              <TD VALIGN="bottom"><IMG SRC="black.bmp"
BORDER="0" WIDTH="<%= GRAPH_BORDER %>" HEIGHT="<%= GRAPH_HEIGHT %>"></TD>
                                              <%
                              ' We're now in the body of the chart.  Loop through the data showing the
bars!
                              For I = 0 To UBound(aValues)
                                    iBarHeight = Int((aValues(I) / iMaxValue) * GRAPH_HEIGHT)

                                    ' This is a hack since browsers ignore a 0 as an image dimension!
                                    If iBarHeight = 0 Then iBarHeight = 1
                              %>
                                              <TD VALIGN="bottom"><IMG SRc="blue.bmp"
BORDER="0" WIDTH="<%= GRAPH_SPACER %>" HEIGHT="1"></TD>
                                              <TD VALIGN="bottom"><IMG SRC="blue.bmp"
BORDER="0" WIDTH="<%= iBarWidth %>" HEIGHT="<%= iBarHeight %>" ALT="<%= aValues(I) %>"></TD>
                                              <%
                              Next 'I
                              %>
                                        </TR>
                                        <!-- I was using GRAPH_BORDER + GRAPH_WIDTH but it
was moving the last x axis label -->
                                        <TR>
                                              <TD COLSPAN="<%= (2 * (UBound(aValues) + 1)) +
1 %>"><IMG SRC="black.bmp" BORDER="0" WIDTH="<%= GRAPH_BORDER + ((UBound(aValues) + 1) * (iBarWidth +
GRAPH_SPACER)) %>" HEIGHT="<%= GRAPH_BORDER %>"></TD>
                                        </TR>
                                        <% ' The label array is optional and is really only
useful for small data sets with very short labels! %>
```

```
                                                            ;
                                                            <% If IsArray(aLabels) Then %>
                                                            <TR>
                                                                  <TD><!-- Spacing for Left Border Column --
></TD>
                                                                  <% For I = 0 To UBound(aValues)  %>
                                                                  <TD><!-- Spacing for Spacer Column --></TD>
                                                                  <TD ALIGN="center"><FONT SIZE="1"><%=
aLabels(I) %></FONT></TD>
                                                                  <% Next 'I %>
                                                                  ;
                                                            </TR>
                                                            <% End If %>
                                                      </TABLE>
                                                </TD>
                                          </TR>
                                          <TR>
                                                <TD COLSPAN="2"><!-- Place holder for X Axis label centering--
></TD>
                                                <TD ALIGN="center"><BR>
                                                      <B>
                                                            <%= strXAxisLabel %>
                                                      </B>
                                                </TD>
                                          </TR>
                                    </TABLE>
                                    <%
End Sub
%>
                                    <%
      Dim timeArray(24)
      Dim i
      FOR i=0 TO 24
            timeArray(i) = 0
            Next

      Dim thisDate
      Dim theDate
```

```
	set thisID = Request.QueryString("WUID")
	thisDate = Request.QueryString("thisSD")

	Set DB = Server.CreateObject("ADODB.connection")
	DB.Open "CB"

	Dim counter
	counter = 0
	set bob = DB.Execute("SELECT * FROM Voltage_Total WHERE WUID= "&thisID)
	'response.Write(Day(bob("Date")))
	'set DB_Results = DB.Execute(SelectCount)
	while not bob.eof
		theDate = Day(bob("Date"))
		theHour = Hour(bob("Date"))

		if (int (thisDate)= int (theDate)) then
			set thisV = bob("Voltage")
			set tempString = DB.Execute("INSERT INTO tempDate(Hour,Voltage) VALUES(" & theHour & ", "
& thisV & ")")
		end if
		bob.MoveNext

	wend

	set DB_Results = DB.Execute("SELECT DISTINCT Hour FROM tempDate")
	while not DB_Results.eof
		set hour1 = DB.Execute("SELECT Voltage FROM tempDate WHERE Hour = " & DB_Results("Hour"))
		timeArray( DB_Results("Hour") ) = 0
		counter = 0
		while not hour1.eof
			timeArray( DB_Results("Hour") ) = timeArray( DB_Results("Hour") ) + hour1("Voltage")
			counter = counter + 1
			hour1.MoveNext
		wend
		timeArray( DB_Results("Hour") ) = timeArray( DB_Results("Hour") ) / counter

		DB_Results.MoveNext
	wend
```

```
'     response.Write (SelectCount)

'     response.Write "this is the count: "
'     response.Write (thisCount)

'     SelectV = "SELECT (A Voltage FROM Power_Totals WHERE ID=" & thisID
'     SelectI1 = "SELECT Current1 FROM Power_Totals WHERE ID=" & thisID
'     SelectI2 = "SELECT Current2 FROM Power_Totals WHERE ID=" & thisID

'     set thisV = DB.Execute(SelectV)
'     set thisC1 = DB.Execute(SelectI1)
'     set thisC2 = DB.Execute(SelectI2)

'     set thisC = thisC1 + thisC2



' Static Chart (with Bar Labels)
ShowChart Array(timeArray(1), timeArray(2), timeArray(3), timeArray(4), timeArray(5), timeArray(6),
timeArray(7), timeArray(8), timeArray(9), timeArray(10), timeArray(11), timeArray(12), timeArray(13),
timeArray(14), timeArray(15), timeArray(16), timeArray(17), timeArray(18), timeArray(19), timeArray(20),
timeArray(21), timeArray(22), timeArray(23), timeArray(24) ), Array("1", "2", "3", "4", "5", "6", "7", "8",
"9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24"), "Voltage
(V)", "Time (Hrs)", "Power Levels"


' Spacing
Response.Write "<BR>" & vbCrLf
Response.Write "<BR>" & vbCrLf
Response.Write "<BR>" & vbCrLf


' Random number chart
'Dim I
'Dim aTemp(49)
```

```
'Randomize
'For I = 0 to 49
'     aTemp(I) = Int((50 + 1) * Rnd)
'Next 'I

' Chart made from random numbers (without Bar Labels)
'ShowChart aTemp, "Note that this isn't an Array!", "Chart of 50 Random Numbers", "Index", "Value"
%>
                              <br>
                              <br>
                              <br>
                              <a href="AllWU2.asp"><strong>BACK</strong></a></td>
                  </tr>
            </table>
      </body>
```

IAChoose2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
```

```
                </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                <tr>
                    <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                           <strong>Wall Units</strong><br>
                               <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                Wall Unit </strong>
                            <br>
                        </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                Wall Unit </strong>
                            <br>
                        </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                Wall Unit </strong>
                            <br>
                        </A>   <strong>Event Sequences </strong>
                        <br>
                               <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                Event Sequence </strong>
                            <br>
                        </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                Event Sequence </strong>
                            <br>
                        </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                Event Sequence </strong>
                            <br>
                        </A>   <strong>Immediate Action </strong>
                        <br>
                              
                        <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                            <br>
```

```
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Wall Units To
Operate</strong><br><br><br>
<p>Click on the Wall Unit To Run...</p>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString = "SELECT * FROM Wall_Unit ORDER BY ID"
    'response.write SearchString  & "<BR>"
      Set DB_Results = DB.Execute(SearchString)
%>

  <form action="IAWUSelect.asp" id="WallUnitEditEntry" method="post" name="Form1">
  <br>
 <div align="left"><table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table4">
 <tr bgcolor=lightblue>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>ID</b></font></p></td>
```

```
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Name</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Address</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Location</b></font></p></td>
      <td><p align="center"><font color="blue" face="Arial" size="3"><b>Top/Bottom</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Run Now?</b></font></p></td>
  </tr>
      <%
            dim loc
            while not DB_Results.eof
            loc = DB_Results("Location")
      %>
  <tr bgcolor=white>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("address")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%response.Write(loc)%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%response.Write "Top"%>
</font></p></td>
    <td><p align="center"><A
href=\IAWUSelect2.asp?HL=1&WUID=<%=DB_Results("ID")%>><strong>Yes</strong></A></p></td>
</tr>
  <tr bgcolor=white>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("address")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%response.Write(loc)%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%response.Write "Bottom" %>
</font></p></td>
    <td><p align="center"><A
href=\IAWUSelect2.asp?HL=0&WUID=<%=DB_Results("ID")%>><strong>Yes</strong></A></p></td>
</tr>
      <%
```

```
            DB_Results.MoveNext
      wend
      %>
</div>
    </td>
  </tr></td>
                  </tr>
            </table>
      </body>
```

IAES2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body  bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr valign=top>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
```

```
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
```

```
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
====================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Information Of Every Event Sequence</strong>
<br>
<br>
<font color="red" face="Arial" size="3"><strong>Red = Event Sequence Information</strong>
<br>
<font color="blue" face="Arial" size="2"><strong>Blue = Corresponding Wall Unit Information</strong>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString = "SELECT * FROM event_sequence "
    Set DB_Results = DB.Execute(SearchString)

    Set DB2 = Server.CreateObject("ADODB.connection")
      DB2.Open "CB"

%>
<table border="0" width="100%" height="100%" ID="Table1">
  <tr>
    <td colspan="2" height="10" align="middle">
  <br></td>
  </tr>
 <tr>
```

```
    <td width="120" valign="top">
    </td>
    <td width="100%" valign="top">
<div align="left">
<table bgcolor="#cccccc" border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table2">

      <%    while not DB_Results.eof      '================================= GO THROUGH EVENT SEQUENCE
TABLE
            set esID = DB_Results("ID")
      %>
      <tr></tr>
      <tr></tr>
      <tr></tr>
      <tr></tr>
      <tr></tr>
      <tr></tr>
  <tr>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>Name</b></font></p></td>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>Start
Date</b></font></p></td>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>Start
Time</b></font></p></td>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>Daily
Repeat?</b></font></p>  </td>
    <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>End
Date</b></font></p></td>
      <td bgcolor=#ffbe9c><p align="left"><font color="red" face="Arial" size="3"><b>Run
Now?</b></font></p></td>
  </tr>
<%    Dim dr
      If (DB_Results("DailyRepeat") = 0) Then
            dr = "No"
      Else
            dr = "Yes"
      end IF
      %>
  <tr>
```

```
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial"
size="3"><b><%=DB_Results("Name")%></b> </font></p></td>
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial"
size="3"><b><%=DB_Results("StartDate")%></b></font></p></td>
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial"
size="3"><b><%=DB_Results("StartTime")%></b> </font></p></td>
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial" size="3"><b><%response.Write
dr%></b> </font></p></td>
    <td bgcolor="White"><p align="left"><font color="#000000" face="Arial"
size="3"><b><%=DB_Results("EndDate")%></b> </font></p></td>
      <td bgcolor="white"><p align="center"><A
href=\IAESOP2.asp?ESID=<%=DB_Results("ID")%>><strong>Yes</strong></A></p></td>
</tr>
      <%
            tempString = "SELECT * FROM ES_effects_WU WHERE ESID ="&esID & " ORDER BY WUID"
            set temp_Results = DB2.Execute(tempString)

      %>
      <tr></tr>
      <tr></tr>
      <tr>
            <td></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial" size="2"><b>Involved
Wall Units</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>ID</b></font></p></td>
          <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Name</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Location</b></font></p></td>
            <td bgcolor=LightBlue><p align="center"><font color="Blue" face="Arial"
size="2"><b>Operation</b></font></p></td>
      </tr>

<%
            while not temp_Results.eof

            set tempWUID = temp_Results("WUID")
```

```
            SearchString2 = "SELECT * FROM Wall_Unit WHERE ID =" & tempWUID
            set DB_Results2 = DB2.Execute(SearchString2)

                        Dim op
                        set t = temp_Results("OPCODE")
                        If (t = 2) Then
                              op = "Turn On"
                        end If
                        If (t = 3) Then
                              op = "Turn Off"
                        end If
                        If (t = 4) Then
                              op = "Raise Power"
                        end If
                        If (t = 5) Then
                              op = "Lower Off"
                        end If
                        If (t = 0) Then
                              op = "Report"
                        end If

                  %>
            <tr>
                  <td></td>
                  <td></td>
                  <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("ID")%></font></p></td>
                  <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("Name")%> </font></p></td>
                  <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%=DB_Results2("Location")%> </font></p></td>
                  <td bgcolor="White"><p align="center"><font color="#000000" face="Arial"
size="2"><%response.Write op%> </font></p></td>
            </tr>
                  <%
                        temp_Results.MoveNext
                        wend
```

```
                  %>

      <%
            DB_Results.MoveNext
            wend
      %>
</table>
                  </tr>
            </table>
      </body>
```

IAESOP2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor="#0066cc">
            <Table bgcolor="#66ccff" width="100%">
                  <tr>
                        <td bgcolor="#66ccff"><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor="#66ccff" align="right" height="12"><IMG alt="flag"
src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
```

```
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
```

```
                                     <strong>-></strong><A 
href="IWUPage2.asp"><strong>Individual 
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A 
href="AllWU2.asp"><strong>All 
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A 
href="IEventPage2.asp"><strong>Individual 
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A 
href="Events2.asp"><strong>All 
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ======================================================= WELCOME PAGE 
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      tempID = Request.QueryString("ESID")
'     response.Write tempID

'     set getWUs = DB.Execute("SELECT * FROM ES_effects_WU WHERE ESID = "& tempID)

      Dim InsertString
      InsertString = "INSERT INTO IA_effects_WU (WUID, OPCODE, offset, HI_LOW) VALUES (0, 200, " & tempID & 
", 0)"
      DB.Execute(InsertString)
'     response.Write InsertString


'     while not getWUs.eof
'           set InsertString = DB.Execute("INSERT INTO IA_effects_WU (WUID, OPCODE, offset) VALUES (" & 
getWUs("WUID") & "," & getWUs("OPCODE") & ", '" & getWUs("offset")  & "')")
'     getWUs.MoveNext
```

```
'     wend

%>
                              <strong>ACTION PERFORMED SUCCESSFULLY!</strong>
                              <br>
                              <br>
                              <br>
                              <a href="foundation.asp"><font
color="black"><strong>BACK</strong></font></a></td>
                  </tr>
            </table>
      </body>
```

IAOP2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
```

```
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
```

```
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                              </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      tempID = Request.QueryString("wuid")
      tempOp = Request.QueryString("op")
      tempoffset = Request.Form("c")
      thisHL = Request.QueryString("HL")

      'response.Write tempID
      'response.Write tempoffset

      If (tempOp = "t") Then
             InsertString = "INSERT INTO IA_effects_WU (WUID, OPCODE, HI_LOW) VALUES (" & "'" & tempID &
"'," & hex(2) & ",'" & thisHL & "')"
      end If
      If (tempOp = "f") Then
             InsertString = "INSERT INTO IA_effects_WU (WUID, OPCODE, HI_LOW) VALUES (" & "'" & tempID &
"'," & hex(3) & ", '" & thisHL & "')"
      end If
```

```
      If (tempOp = "r") Then
             InsertString = "INSERT INTO IA_effects_WU (WUID, OPCODE, offset, HI_LOW) VALUES (" & "'" &
tempID & "'," & hex(4) & ",'" & tempoffset & "','" & thisHL & "')"
      end If

      'response.Write InsertString
      DB.Execute(InsertString)
%>

<strong>ACTION PERFORMED SUCCESSFULLY!</strong>
<br>
<br>
<br>
<a href=foundation.asp><font color=black><strong>BACK</strong></font></a></td>
                  </tr>
            </table>
      </body>
```

IAWU2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
```

```
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
```

```
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
==================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                  <strong>Perform Immediate Action ...</strong>
<br><br>Click on a different choice below:<BR><br><br>

<table border="0" width="50%" height="50%" ID="Table1">
 <tr>

    <td width="50%" valign="top">
<div align="center">
<table bgcolor=#cccccc border="1" cellPadding="3" cellSpacing="2" width="70%" ID="Table2">

  <tr>
    <td bgcolor=LightBlue><p align="center"><font color="blue" face="Arial" size="3"><b>Immediate Action
Choices</b></font></p></td>
  </tr>
```

```
  <tr>
    <td bgcolor=White><p align="center"><A href=IAChoose2.asp><font color="black" face="Arial"
size="3"><strong>Wall Units</strong></font></A></p></td>
  </tr>
  <tr>
    <td bgcolor=White><p align="center"><A href=IAES2.asp><font color="black" face="Arial"
size="3"><strong>Event Sequences</strong></A></p></td>
  </tr>

</table>
                  </tr>
            </table>
      </body>
```

IAWUSelect2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
```

```
			<table border="1" bgcolor="#0033cc" width="100%">
				<tr>
					<td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
						   <strong>Wall Units</strong><br>
						       <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
								Wall Unit </strong>
							<br>
						</A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
								Wall Unit </strong>
							<br>
						</A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
								Wall Unit </strong>
							<br>
						</A>   <strong>Event Sequences </strong>
						<br>
						       <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
								Event Sequence </strong>
							<br>
						</A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
								Event Sequence </strong>
							<br>
						</A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
								Event Sequence </strong>
							<br>
						</A>   <strong>Immediate Action </strong>
						<br>
						      
						<stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
							<br>
						</A>
						<H2><u>View System</u></H2>
```

```
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                            Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                            Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                            Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                            Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      tempID = Request.QueryString("WUID")
      thisTB = Request.QueryString("tb")
      thisHL = Request.QueryString("HL")

%>
<br><br><br><br>
<a href=\IAOP2.asp?HL=<%response.Write thisHL%>&op=t&wuid=<%response.Write tempID%>><strong>TURN THIS WALL
UNIT ON</strong></a>
<br><br>
<a href=\IAOP2.asp?HL=<%response.Write thisHL%>&op=f&wuid=<%response.Write tempID%>><strong>TURN THIS WALL
UNIT OFF</strong></a>
<br><br>

<form action="IAOP2.asp?HL=<%response.Write thisHL%>&op=r&wuid=<%response.Write tempID%>"
id="WallUnitEntry" method="post" name="Form1">
```

```
     <div align="left"><table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table2">
      <tr>
            <td><div align="center"><center><p><input id="submit1" name="submit1" type="submit"
value="CHANGE POWER"></td>
            <td>  low               <input type=radio name="c" value=125 ID="Checkbox1"> </td>
            <td>  Low Medium  <input type=radio name="c" value=100 ID="Checkbox2"> </td>
            <td>  Medium            <input type=radio name="c" value=75 ID="Checkbox3"> </td>
            <td>  High Medium <input type=radio name="c" value=50 ID="Checkbox4"> </td>
            <td>  High        <input type=radio name="c" value=10 ID="Checkbox5"> </td>
      </tr></td>
                  </tr>
            </table>
      </body>
```

IEventPage2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
```

```
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
```

```
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>VIEW INDIVIDUAL EVENT SEQUENCES BY...</strong>
<br><br>Click on the different cases below:<BR><br><br>

<table border="0" width="50%" height="50%" ID="Table1">
 <tr>
    <td width="50%" valign="top">
<table border="1" cellPadding="1" cellSpacing="2" width="90%" ID="Table2">

  <tr>
    <td bgcolor=#ffbe9c><p align="center"><font color="red" face="Arial" size="3"><b>Event Sequence
Views</b></font></p></td>
  </tr><tr></tr><tr></tr><tr></tr>
  <tr>
    <td bgcolor=white><p align="center"><A href=EventIDlist2.asp><font color="black" face="Arial"
size="3"><b>By IDs</b></A></p></td>
  </tr><tr></tr><tr></tr>
  <tr>
    <td bgcolor=white><p align="center"><A href=EventINameList2.asp><font color="black" face="Arial"
size="3"><b>By Names</b></A></p></td>
```

```
  </tr><tr></tr><tr></tr>

</table>
</td>
                  </tr>
            </table>
      </body>
```

IWUPage2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
```

```
                                    Wall Unit </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                    Wall Unit </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                    Wall Unit </strong>
                              <br>
                        </A>   <strong>Event Sequences </strong>
                        <br>
                               <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                    Event Sequence </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                    Event Sequence </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                    Event Sequence </strong>
                              <br>
                        </A>   <strong>Immediate Action </strong>
                        <br>
                              
                        <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                              <br>
                        </A>
                        <H2><u>View System</u></H2>
                           <strong>Wall Units Information</strong><br>
                               <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                    Wall Units</strong><br>
                        </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
```

```
                                        Wall Units</strong><br>
                                </A>   <strong>Event Sequences' Information</strong><br>
                                       <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                        Event Sequences</strong><br>
                                </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                        Event Sequences</strong></A>
                                <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                                <strong>VIEW INDIVIDUAL WALL UNITS BY...</strong>
<br><br>Click on a different view below:<BR><br><br>

<table border="0" width="50%" height="50%" ID="Table1">
 <tr>

    <td width="50%" valign="top">
<div align="center">
<table bgcolor=#cccccc border="1" cellPadding="3" cellSpacing="2" width="70%" ID="Table2">

  <tr>
    <td bgcolor=LightBlue><p align="center"><font color="blue" face="Arial" size="3"><b>Wall Unit
Views</b></font></p></td>
  </tr>
  <tr>
    <td bgcolor=White><p align="center"><A href=WUIDlist2.asp><font color="black" face="Arial"
size="3"><strong>By IDs</strong></font></A></p></td>
  </tr>
  <tr>
    <td bgcolor=White><p align="center"><A href=WUNameList2.asp><font color="black" face="Arial"
size="3"><strong>By Names</strong></A></p></td>
  </tr>
  </table>
<br>
            </tr>
```

```
			</table>
		</body>
```

MS_ES_add2.asp

```
<%@ Language=VBScript %>
<HTML>
	<HEAD>
		<TITLE>Menu Page</TITLE>
		<META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
		<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
		<meta name="publisher" content="ScotchInc">
		<meta name="owner" content="ScotchInc">
		<meta name="author" content="ScotchInc">
		<meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
		<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
	</HEAD>
	<body>
		<body bgcolor=#0066cc>
		<Table bgcolor=#66ccff width="100%" ID="Table3">
			<tr>
				<td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
				<td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
			</tr>
		</Table>
		<table border="1" bgcolor="#0033cc" width="100%">
			<tr>
				<td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
					   <strong>Wall Units</strong><br>
					       <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
							Wall Unit </strong>
						<br>
					</A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
```

```
                                    Wall Unit </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                    Wall Unit </strong>
                              <br>
                        </A>   <strong>Event Sequences </strong>
                        <br>
                               <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                    Event Sequence </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                    Event Sequence </strong>
                              <br>
                        </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                    Event Sequence </strong>
                              <br>
                        </A>   <strong>Immediate Action </strong>
                        <br>
                              
                        <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                              <br>
                        </A>
                        <H2><u>View System</u></H2>
                           <strong>Wall Units Information</strong><br>
                               <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                    Wall Units</strong><br>
                        </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                    Wall Units</strong><br>
                        </A>   <strong>Event Sequences' Information</strong><br>
                               <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
```

```
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">

                        <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      SearchString = "Select * FROM Wall_Unit ORDER BY ID"
      Set DB_Results = DB.Execute(SearchString)

%>

<strong>COMPLETE FORM TO ADD AN EVENT SEQUENCE...</strong>
<br><br><br><br>

<table border="0" width="100%" ID="Table1">
  <tr>
    <td width="120" valign="top"></td>
    <td width="100%" valign="top">
<form action="ES_AddInsert2.asp" id="WallUnitEntry" method="post" name="Form1">
      </form>
      <div align="left"><table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table2">
        <tr>
          <td><div align="right"><p> Sequence Name  </td>
          <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input name="SName"
size="30" ID="Text1"></font></td>
        </tr>
        <tr>
          <td><div align="right"><p> Sequence Start Date  </td>
```

```
          <td>Month:(mm)<INPUT type="text" Size="1" ID="Text2" NAME="MonthSD">  Day:(dd)<INPUT
type="text" size="1" ID="Text8" NAME="DaySD">  Year:(yyyy)<INPUT type="text" size="2" ID="Text9"
NAME="YearSD"></td>
         </tr>
        <tr>
          <td><div align="right"><p> Sequence End Date  </td>
          <td>Month:(mm)<INPUT type="text" Size="1" ID="Hour" NAME="MonthED">  Day:(dd)<INPUT
type="text" size="1" ID="Text4" NAME="DayED">  Year:(yyyy)<INPUT type="text" size="2" ID="Text5"
NAME="YearED"></td>
        </tr>
        <tr>
          <td><div align="right"><p> Sequence Start Time  </td>
         <td>Hour:<INPUT type="text" Size="1" ID="Text3" NAME="HourST">  Minute:<INPUT
type="text" size="1" ID="Text6" NAME="MinST">     AM<INPUT type="radio"
ID="Checkbox4" NAME="PM" value="am">     PM<INPUT type="radio" ID="Checkbox5" NAME="PM"
value="pm"></td>
          </tr>
        <tr>
          <td><div align="right"><p> Daily Repeat?  </td>
          <td>Yes<INPUT type=radio ID="Checkbox1" NAME="Yes" value="DRY">     No<INPUT
type=radio ID="Checkbox2" NAME="Yes" value="DRN"></td>
        </tr>
        <tr>
          <td><div align="right"><p> Days To Repeat  </td>
          <td>Sun<INPUT type="checkbox" ID="Checkbox11" NAME="Sun" value="Sun">    
Mon<INPUT type="checkbox" ID="Checkbox3" NAME="Mon" value="Mon">     Tues<INPUT
type="checkbox" ID="Checkbox6" NAME="Tues" value="Tues">    Wed<INPUT type="checkbox"
ID="Checkbox7" NAME="Wed" value="Wed">    Thur<INPUT type="checkbox" ID="Checkbox8"
NAME="Thur" value="Thur">    Fri<INPUT type="checkbox" ID="Checkbox9" NAME="Fri"
value="Fri">    Sat<INPUT type="checkbox" ID="Checkbox10" NAME="Sat" value="Sat"></td>
        </tr>
        </tr>
        <tr>
          <td><div align="center"><center><p><input id="reset1" name="reset1" type="reset"
value="Reset"></td>
          <td><div align="center"><center><p><input id="submit1" name="submit1" type="submit"
value="Submit"></td>
        </tr>
```

```
      </table>
      </div>
    </form>

    </td>
  </tr>
</table>


                        </td>
                  </tr>
            </table>
      </body>
```

MS_ES_delete2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
```

```
<table border="1" bgcolor="#0033cc" width="100%">
      <tr>
            <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                     <strong>Wall Units</strong><br>
                         <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                              Wall Unit </strong>
                        <br>
                  </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                              Wall Unit </strong>
                        <br>
                  </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                              Wall Unit </strong>
                        <br>
                  </A>   <strong>Event Sequences </strong>
                  <br>
                         <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                              Event Sequence </strong>
                        <br>
                  </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                              Event Sequence </strong>
                        <br>
                  </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                              Event Sequence </strong>
                        <br>
                  </A>   <strong>Immediate Action </strong>
                  <br>
                        
                  <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                        <br>
                  </A>
                  <H2><u>View System</u></H2>
```

```
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                        <strong>The Table Below Is The Information Of Event Sequences</strong><br><br><br>
<p>Click on the Event Sequence To Delete...</p>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString = "SELECT * FROM event_sequence ORDER BY ID"
      Set DB_Results = DB.Execute(SearchString)
%>
<%'   <td colspan="2" height="10" align="middle">     %>
   <form action="WU_DeleteInsert.asp" id="WallUnitDeleteEntry" method="post" name="Form1">
  <br>
  <%
  '  <td width="120" valign="top">
  '  </td>
  '  <td width="100%" valign="top">
%>
<div align="left"><table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table4">
```

```
  <tr bgcolor=#ffbe9c>
    <td><p align="center"><font color="red" face="Arial" size="3"><b>ID</b></font></p></td>
    <td><p align="center"><font color="red" face="Arial" size="3"><b>Name</b></font></p></td>
    <td><p align="center"><font color="red" face="Arial" size="3"><b>Start Time</b></font></p></td>
    <td><p align="center"><font color="red" face="Arial" size="3"><b>Start Date</b></font></p>  </td>
    <td><p align="center"><font color="red" face="Arial" size="3"><b>End Date</b></font></p></td>
    <td><p align="center"><font color="red" face="Arial" size="3"><b>Delete?</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof  %>
  <tr bgcolor=white>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td> <p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartTime")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartDate")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("EndDate")%>
</font></p></td>
    <td> <p align="center"><A
href=\ES_DeleteInsert2.asp?ESID=<%=DB_Results("ID")%>><strong>Yes</strong></A></p></td>
  </tr>
      <%
            DB_Results.MoveNext
      wend
      %>

      </div>
    </td>
  </tr>

                        </td>
                  </tr>
            </table>
      </body>
```

MS_ES_edit2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
```

```
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
```

```
                        <% ' ====================================================== WELCOME PAGE
====================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Information Of Event
Sequences</strong><br><br><br>
<p>Click on the Event Sequence To Edit...</p>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString = "SELECT * FROM event_sequence ORDER BY ID"
    'response.write SearchString  & "<BR>"
      Set DB_Results = DB.Execute(SearchString)
%>


<%'    <td colspan="2" height="10" align="middle">    %>
   <form action="ES_EditInsert.asp" id="WallUnitDeleteEntry" method="post" name="Form1">
  <br>
<%
  '  <td width="120" valign="top">

  '  </td>
  '  <td width="100%" valign="top">
%>

<div align="left"><table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table4">



  <tr bgcolor=#ffbe9c>
    <td>
      <p align="center"><font color="red" face="Arial" size="3"><b>ID</b></font></p></td>
    <td>
      <p align="center"><font color="red" face="Arial" size="3"><b>Name</b></font></p></td>
    <td>
      <p align="center"><font color="red" face="Arial" size="3"><b>Start Time</b></font></p></td>
    <td>
```

```
      <p align="center"><font color="red" face="Arial" size="3"><b>Start Date</b></font></p>  </td>
    <td>
      <p align="center"><font color="red" face="Arial" size="3"><b>End Date</b></font></p></td>
    <td>
      <p align="center"><font color="red" face="Arial" size="3"><b>Edit?</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof  %>
  <tr bgcolor=white>
    <td>
      <p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td>
      <p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td>
      <p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartTime")%>
</font></p></td>
    <td>
      <p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartDate")%>
</font></p></td>
    <td>
      <p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("EndDate")%>
</font></p></td>
    <td>
      <p align="center"><A
href=\ES_EditInsert2.asp?EVID=<%=DB_Results("ID")%>><strong>Yes</strong></A></p></td>


</tr>



      <%
            DB_Results.MoveNext
      wend
      %>


</div>
    </td>
```

```
  </tr></td>
                </tr>
            </table>
        </body>
```

MS_ES_pickedWUs2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                    Wall Unit </strong>
                                    <br>
```

```
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
```

```
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>Addition Successful!</strong>
                              <%

Set DB = Server.CreateObject("ADODB.connection")
DB.Open "CB"

String1 = "Select * FROM Wall_Unit WHERE Op = '1'"
set DB_Results = DB.Execute(String1)

Set EDB = Server.CreateObject("ADODB.connection")
EDB.Open "CB"

while not DB_Results.eof

'     response.Write "hi there"

      Dim thisN
      thisN =DB_Results("Name")

      Set Effects = Server.CreateObject("ADODB.connection")
      Effects.Open "CB"

      SString = "Select * FROM Wall_Unit WHERE Name =" & "'" & thisN & "'"    '================= Find The
Wall Unit IDs
      set Eff_Results = Effects.Execute(SString)

      tempWUID = Eff_Results("ID")
```

```
	Dim op
	dim opt
	opt = Request.Form("ct")
	op = Request.Form("c")

	Dim OpTOn
	Dim OpTOff
	Dim OpRP

	OpTOn = Hex(2)
	OpTOff = Hex(3)
	OpRP = Hex(4)

	thisID = Request.QueryString("EVID")
	'response.Write thisReport

	Set LoopRun = Server.CreateObject("ADODB.connection")
	LoopRun.Open "CB"

	Dim highLow
	highLow = 0

	if (opt = "ont") OR (opt = "offt") OR (opt = "125t") OR (opt = "100t") OR (opt = "75") OR (opt =
"50t") OR (opt = "10t") then
		highLow = 1
	end if


	%><br><%

	If (op="ont") OR (op="onb") Then		'=========================== If there is  a Turn ON
selected
		'response.Write "It did not get clicked ON"
'	Else
		'response.Write "It got turned ON"
		StringT= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, HI_LOW) VALUES "
		StringT = StringT & "('"&thisID&"',"& "'" &tempWUID&"'," & "'" & OpTOn & "', " & highLow & ")"
```

```
            'response.Write StringT
            LoopRun.Execute(StringT)
      end If

      If (op="offt") OR (op="offb") Then        '============================ If there is  a Turn Off 
selected
            'response.Write ""
'     Else
            StringTf= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, HI_LOW) VALUES "
            StringTf = StringTf & "('"&thisID&"',"& "'" &tempWUID&"'," & "'" & OpTOff & "', " & highLow & 
")"
            'response.Write StringTf
            LoopRun.Execute(StringTf)
      end If

      If (op="125t") OR (op="125b") Then        '============================ If there is  a Raise Power 
selected
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&thisID&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 125, "  & highLow 
& ")"
            'response.Write StringPR
            LoopRun.Execute(StringPR)
      end If

      If (op="100t") OR (op="100b") Then        '============================ If there is  a Raise Power 
selected
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&thisID&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 100, "  & highLow 
& ")"
            'response.Write StringPR
            LoopRun.Execute(StringPR)
      end If
```

```
      If (op="75t") OR (op="75b") Then          '============================ If there is  a Raise Power
selected
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&thisID&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 75, "  & highLow
& ")"
            'response.Write StringPR
            LoopRun.Execute(StringPR)
      end If

      If (op="50t") OR (op="50b") Then          '============================ If there is  a Raise Power
selected
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&thisID&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 50, "  & highLow
& ")"
            'response.Write StringPR
            LoopRun.Execute(StringPR)
      end If

      If (op="10t") OR (op="10b") Then          '============================ If there is  a Raise Power
selected
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&thisID&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 10, "  & highLow
& ")"
            'response.Write StringPR
            LoopRun.Execute(StringPR)
      end If


      DB_Results.MoveNext

wend
```

```
	%><br><%

	InsertString3 = "UPDATE Wall_Unit SET OP = '0';"
	EDB.Execute(InsertString3)

%></td>
				</tr>
			</table>
	</body>
```

MS_ES_pickWUs2.asp

```
<%@ Language=VBScript %>
<HTML>
	<HEAD>
		<TITLE>Menu Page</TITLE>
		<META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
		<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
		<meta name="publisher" content="ScotchInc">
		<meta name="owner" content="ScotchInc">
		<meta name="author" content="ScotchInc">
		<meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
		<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
	</HEAD>
	<body bgcolor=#0066cc>
		<Table bgcolor=#66ccff width="100%">
			<tr>
				<td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
				<td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
			</tr>
		</Table>
		<table border="1" bgcolor="#0033cc" width="100%">
			<tr valign=top>
				<td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
					   <strong>Wall Units</strong><br>
```

```
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
```

```
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>
                        <% ' ====================================================== WELCOME PAGE
==================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>PLEASE SELECT THE WALL UNITS FOR THE EVENT SEQUENCE BELOW...</strong>

<%

      response.Write "firstbase"

      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      SearchString = "Select * FROM event_sequence WHERE ID ="
      SearchString = SearchString & "(Select MAX(ID) FROM event_sequence)"
      Set DB_Results = DB.Execute(SearchString)

      'response.Write (DB_Results("ID"))

      String1 = "SELECT * FROM Wall_Unit WHERE Op = '1'"
      set getWU = DB.Execute(String1)

      Dim maxid
      maxid = DB_Results("ID")
'     response.Write maxid

      Dim more
```

```
more = Request.QueryString("more")

if (more=1) then

response.Write "secondbase"

      while not getWU.eof

            response.Write "thirdbase"

            Dim thisN
            thisN =getWU("Name")

            SString = "Select * FROM Wall_Unit WHERE Name =" & "'" & thisN & "'"
'================== Find The Wall Unit IDs
            set Eff_Results = DB.Execute(SString)
            'response.Write SString
            tempWUID = Eff_Results("ID")

            Dim op
            dim opt
            opt = Request.Form("ct")
            op = Request.Form("c")

            response.Write opt
            response.Write op

            Dim OpTOn
            Dim OpTOff
            Dim OpRP

            OpTOn = Hex(2)
            OpTOff = Hex(3)
            OpRP = Hex(4)

            Set LoopRun = Server.CreateObject("ADODB.connection")
            LoopRun.Open "CB"
```

```
                  Dim highLow
                  highLow = 0

                  if (opt = "ont") OR (opt = "offt") OR (opt = "125t") OR (opt = "100t") OR (opt = "75") OR
(opt = "50t") OR (opt = "10t") then
                        highLow = 1
                  end if

      If (op="ont") OR (op="onb") Then          '============================ If there is  a Turn ON
selected
            'response.Write "It did not get clicked ON"
'     Else
            'response.Write "It got turned ON"
            StringT= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, HI_LOW) VALUES "
            StringT = StringT & "('"&maxid&"',"& "'" &tempWUID&"'," & "'" & OpTOn & "', " & highLow & ")"
            response.Write StringT
            LoopRun.Execute(StringT)
      end If

      If (op="offt") OR (op="offb") Then        '============================ If there is  a Turn Off
selected
            'response.Write ""
'     Else
            StringTf= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, HI_LOW) VALUES "
            StringTf = StringTf & "('"&maxid&"',"& "'" &tempWUID&"'," & "'" & OpTOff & "', " & highLow &
")"
            'response.Write StringTf
            LoopRun.Execute(StringTf)
      end If

      If (op="125t") OR (op="125b") Then        '============================ If there is  a Raise Power
selected
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&maxid&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 125, "  & highLow
& ")"
            'response.Write StringPR
```

```
            LoopRun.Execute(StringPR)
      end If

      If (op="100t") OR (op="100b") Then        '============================ If there is  a Raise Power
selected
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&maxid&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 100, "  & highLow
& ")"
            'response.Write StringPR
            LoopRun.Execute(StringPR)
      end If

      If (op="75t") OR (op="75b") Then          '============================ If there is  a Raise Power
selected
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&maxid&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 75, "  & highLow &
")"
            'response.Write StringPR
            LoopRun.Execute(StringPR)
      end If

      If (op="50t") OR (op="50b") Then          '============================ If there is  a Raise Power
selected
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&maxid&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 50, "  & highLow &
")"
            'response.Write StringPR
            LoopRun.Execute(StringPR)
      end If

      If (op="10t") OR (op="10b") Then          '============================ If there is  a Raise Power
selected
```

```
            'response.Write ""
'     Else
            StringPR= "Insert INTO ES_effects_WU (ESID, WUID, OPCODE, offset, HI_LOW) VALUES "
            StringPR = StringPR & "('"&maxid&"',"& "'" &tempWUID&"'," & "'" & OpRP & "', 10, "  & highLow &
")"
            'response.Write StringPR
            LoopRun.Execute(StringPR)
      end If



            getWU.MoveNext
            wend
      end if
      InsertString3 = "UPDATE Wall_Unit SET OP = '0';"
      DB.Execute(InsertString3)

%>

<br><br>
<table border="0" width="100%" height="100%" ID="Table2">
  <tr>
    <td colspan="2" height="10" align="middle">

  <br></td>
  </tr>
 <tr>
    <td width="120" valign="top"></td>
    <td width="100%" valign="top">

    <div align="left">
<table border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table3">

  <tr bgcolor="#ffbe9c">
    <td><p align="left"><font color="red" face="Arial" size="3"><b>ID</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Name</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Start Date</b></font></p></td>
    <td><p align="left"><font color="red" face="Arial" size="3"><b>End Date</b></font></p>  </td>
```

```
    <td><p align="left"><font color="red" face="Arial" size="3"><b>Start Time</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof
      Dim thisID
      thisID =DB_Results("ID")

      %>
  <tr bgcolor=WHITE>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%response.Write
thisID%></font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartDate")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("EndDate")%>
</font></p></td>
    <td><p align="left"><font color="#000000" face="Arial" size="3"><%=DB_Results("StartTime")%>
</font></p></td>
</tr>
      <%
            DB_Results.MoveNext
      wend
      %>

</table>
<p><b>Select Wall Units Involved</b></p>
(Hold down Control to select multiple Units)

<table ID="Table1">
       <tr>
          <td>
<% ' --------------==========================------------------------==================== wall units
table%>

<%
Set SDB = Server.CreateObject("ADODB.connection")
SDB.Open "CB"
```

```
SearchString1 = "Select * FROM Wall_Unit"

Set SDB_Results = SDB.Execute(SearchString1)

Dim intNumberSelected  ' Count of items selected
Dim strSelectedTeams   ' String returned from QS (or Form)
Dim arrSelectedTeams   ' Variable to hold team array
Dim I                  ' Looping variable
Dim WallUnitsIDs

' Retrieve the count of items selected
intNumberSelected = Request.Form("WallUnits").Count
WallUnitsIDs = Request.Form("WallUnits")

      %>
      <BR>

      <FORM ACTION="MS_ES_pickWUs2.asp" METHOD="post" ID="Form1">

            </form>

            <SELECT NAME="WallUnits" MULTIPLE SIZE="5" ID="Select1">
            <%
                  while not SDB_Results.eof
                  Dim thisName
                  thisName =SDB_Results("Name")
                  response.Write thisName
            %>
                  <OPTION><%response.Write thisName%></OPTION>
            <%
                  SDB_Results.MoveNext
                  wend

                  If intNumberSelected = 0 Then
            %>
                  </SELECT>

                  <BR>
```

```
                  <div align="center"><center><p><input id="reset1" name="reset1" type="reset"
value="Reset">
                  <INPUT type="submit" value="Units Selected" ID="Submit1" NAME="Submit1">

                  </FORM>
            <%
                  Else
                        strSelectedTeams = Request.Form("WallUnits")
                        arrSelectedTeams = Split(strSelectedTeams, ", ", -1, 1)
            %>
                  </select>
                  </form>
                  <FORM ACTION="MS_ES_pickedWUs2.asp?EVID=<%response.Write thisID%>" METHOD="post"
ID="Form2">

                  <TABLE BORDER="1" ID="Table4"><br>

                  <p><H3>Choose the Operation For Each Wall Unit</H3></p>

                  <TR bgcolor=lightblue>
                        <td><font color="blue" face="Arial" size="3"><b>Wall Unit Name<b></font></p></td>

                  </TR>
            <%          Set EDB = Server.CreateObject("ADODB.connection")
                        EDB.Open "CB"

                        Dim multiSelect
                        multiSelect = 0

                  For I = LBound(arrSelectedTeams) To UBound(arrSelectedTeams)      %>
            <TR >
                        <%    Dim theseNames
                              theseNames = arrSelectedTeams(I)
                              theseR = theseNames &"R"
                              theseTon = theseNames &"Ton"
                              theseToff = theseNames &"Toff"
                              thesePr = theseNames &"Pr"
                              thesePl = theseNames &"Pl"
```

```
                              InsertString3 = "UPDATE Wall_Unit SET OP = '1' WHERE Name =" & "'"
&theseNames&"'"

                  '           response.Write InsertString3
                              EDB.Execute(InsertString3)

                        %>
                              <FORM ACTION="MS_ES_pickWUs2.asp?more=1" METHOD="post" ID="Form3">

                        </form>

                        <TD bgcolor=white><%response.Write theseNames & "            (top)" %></TD>

                        <td align=right>  Turn On           <INPUT type=radio ID="Checkbox6" NAME="ct"
value="ont"></TD>
                        <td align=right>  Turn Off    <INPUT type=radio ID="Checkbox7" NAME="ct"
value="offt"></TD>
                        <td align=right>  Low               <input align=right type=radio name="ct"
value="125t" ID="Checkbox1"> </td>
                        <td align=right>  Low Medium  <input align=right type=radio name="ct" value="100t"
ID="Radio1"> </td>
                        <td align=right>  Medium            <input align=right type=radio name="ct"
value="75t" ID="Radio2"> </td>
                        <td align=right>  High Medium <input align=right type=radio name="ct" value="50t"
ID="Checkbox4"> </td>
                        <td align=right>  High        <input align=right type=radio name="ct" value="10t"
ID="Checkbox5"> </td>
                  </tr>
                  <tr>
                        <TD bgcolor=white><%response.Write theseNames & "          (bottom)" %></TD>

                        <td align=right>  Turn On           <INPUT type=radio ID="Radio3" NAME="c"
value="onb"></TD>
                        <td align=right>  Turn Off    <INPUT type=radio ID="Radio4" NAME="c"
value="offb"></TD>
                        <td align=right>  Low               <input align=right type=radio name="c"
value="125b" ID="Radio5"> </td>
```

```
                        <td align=right>  Low Medium  <input align=right type=radio name="c" value="100b"
ID="Radio6"> </td>
                        <td align=right>  Medium            <input align=right type=radio name="c"
value="75b" ID="Radio7"> </td>
                        <td align=right>  High Medium <input align=right type=radio name="c" value="50b"
ID="Radio8"> </td>
                        <td align=right>  High        <input align=right type=radio name="c" value="10b"
ID="Radio9"> </td>
            </TR>
            <%
                  Next 'I
            %>
                  </TABLE>

                  <div align="center"><center><p>
                  <input id="Reset2" name="reset1" type="reset" value="Reset">
                  <INPUT type="submit" value="Operations Selected" ID="Submit2" NAME="Submit1">
                  </form>

<!--
                  <FORM ACTION="MS_ES_pickWUs2.asp?more=1" METHOD="post" ID="Form3">
                  <INPUT type="submit" value="Select More Operations" ID="Submit3" NAME="Submit1">
                  </form>
-->
                  </td>
                  </tr>
                  </table>
            <%    End If      %>

            </td>
                  </tr>
            </table>
      </body>
```

MS_WU_add2.asp

```
<%@ Language=VBScript %>
<HTML>
```

```
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%" ID="Table1">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
      <table border="1" bgcolor=#0033cc width="100%" ID="Table2">
      <tr>
            <td  width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
  
      <strong>Wall Units</strong><br>      
            <strong>-></strong><A href=MS_WU_add2.asp><strong>Add Wall Unit
</strong><br></A>      
            <strong>-></strong><A href=MS_WU_Delete2.asp><strong>Delete Wall Unit </strong><br></A>
      
            <strong>-></strong><A href=MS_WU_edit2.asp><strong>Edit Wall Unit </strong><br></A>
  
      <strong>Event Sequences </strong><br>      
            <strong>-></strong><A href=MS_ES_add2.asp><strong>Add Event Sequence
</strong><br></A>      
            <strong>-></strong><A href=MS_ES_Delete2.asp><strong>Delete Event Sequence
</strong><br></A>      
            <strong>-></strong><A href=MS_ES_edit2.asp><strong>Edit Event Sequence </strong><br></A>
  
```

```
      <strong>Immediate Action </strong><br>      
            <stong>-></stong><A href=IAWU2.asp><strong>Immediate Action Commands</strong><br><br></A>

<H2><u>View System</u></H2>
  
      <strong>Wall Units Information</strong><br>      
            <strong>-></strong><A href=IWUPage2.asp><strong>Individual Wall Units</strong><br></A>
      
            <strong>-></strong><A href=AllWU2.asp><strong>All Wall Units</strong><br></A>

  
      <strong>Event Sequences' Information</strong><br>      
            <strong>-></strong><A href=IEventPage2.asp><strong>Individual Event Sequences</strong><br></A>
      
            <strong>-></strong><A href=Events2.asp><strong>All Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>

<% ' ====================================================== ms_wu_ADD PAGE
==================================== %>
            <td width="80%" bgcolor=#ccffff >
<br><br><br><br>

<table align=center border="0" width="70%" height="50%" ID="Table3">
 <tr>

    <td width="7%" valign="top">
<div align="center">
<table align=center bgcolor=#cccccc border="1" cellPadding="3" cellSpacing="2" width="70%" ID="Table4">
      <tr>
            <td bgcolor=RED><div align="center"> <b>WARNING</b>  </td>
      </tr>
    <tr>
            <td bgcolor=red><div align="center"><b>THIS MAY TAKE OVER AN HOUR TO PROCESS</b></td>
    </tr>
    <tr>
            <td bgcolor=red><div align="center"><b>REGULARLY SCHEDULED EVENTS WILL <u>NOT</u>
PERFORM</b></td>
```

```
    </tr>
    <tr></tr>
    <tr></tr>
    <tr></tr>
    <tr></tr>
    <tr>
            <td bgcolor=red><div align="center"><b>ARE YOU SURE YOU WANT TO CONTINUE?</b></td>
    </tr>
    <tr></tr>
    <tr></tr>
    <tr></tr>
    <tr></tr>
    <tr></tr>
      <tr>
            <td bgcolor=red><div align="center"><a href=FOUNDATION.asp><img src="No.bmp"</a><a
href=searching1.asp?add=0><img src="Yes.bmp"></a></div></td>
    </tr>
    </table>

<%
'<form action="WU_AddInsert2.asp" id="WallUnitEntry" method="post" name="Form1">
'      <div align="left"><table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table4">
'        <tr>
'          <td bgcolor=white><div align="right"><p> Wall Unit Name  </td>
'          <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input name="WUName"
size="30" ID="Text1"></font></td>
'        </tr>
'        <tr>
'          <td bgcolor=white><div align="right"><p> Wall Unit PW  </td>
'          <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input name="WUPW"
size="10" ID="Text3"></font></td>
'        </tr>
'        <tr>
'          <td bgcolor=white><div align="right"><p> Wall Unit Location  </td>
'          <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input name="WULocation"
size="50" ID="Text4"></font></td>
'        </tr>
'        <tr bgcolor=#ccffff>
```

```
'           <td><div align="center"><center><p><input id="reset1" name="reset1" type="reset"
value="Reset"></td>
'           <td><div align="center"><center><p><input id="submit1" name="submit1" type="submit"
value="Submit"></td>
'         </tr>
'       </table>
'       </div>
'     </form>
'     </td>
'   </tr>
  %>
</table>

            </td>
      </table>
</body>
```

MS_WU_delete2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>

<body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%" ID="Table1">
                  <tr>
```

```
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
      <table border="1" bgcolor=#0033cc width="100%">
      <tr valign=top>
            <td  width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
  
      <strong>Wall Units</strong><br>      
            <strong>-></strong><A href=MS_WU_add2.asp><strong>Add Wall Unit
</strong><br></A>      
            <strong>-></strong><A href=MS_WU_Delete2.asp><strong>Delete Wall Unit </strong><br></A>
      
            <strong>-></strong><A href=MS_WU_edit2.asp><strong>Edit Wall Unit </strong><br></A>
  
      <strong>Event Sequences </strong><br>      
            <strong>-></strong><A href=MS_ES_add2.asp><strong>Add Event Sequence
</strong><br></A>      
            <strong>-></strong><A href=MS_ES_Delete2.asp><strong>Delete Event Sequence
</strong><br></A>      
            <strong>-></strong><A href=MS_ES_edit2.asp><strong>Edit Event Sequence </strong><br></A>
  
      <strong>Immediate Action </strong><br>      
            <stong>-></stong><A href=IAWU2.asp><strong>Immediate Action Commands</strong><br><br></A>

<H2><u>View System</u></H2>
  
      <strong>Wall Units Information</strong><br>      
            <strong>-></strong><A href=IWUPage2.asp><strong>Individual Wall Units</strong><br></A>
      
            <strong>-></strong><A href=AllWU2.asp><strong>All Wall Units</strong><br></A>

  
      <strong>Event Sequences' Information</strong><br>      
            <strong>-></strong><A href=IEventPage2.asp><strong>Individual Event Sequences</strong><br></A>
      
            <strong>-></strong><A href=Events2.asp><strong>All Event Sequences</strong></A>
```

```
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>
                        </td>

<% ' ====================================================== WELCOME PAGE
====================================== %>
            <td width="80%" bgcolor=#ccffff>

            <strong>The Table Below Is The Information Of Wall Units</strong><br><br><br>
<p>Click on the Wall Unit To Delete...</p>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString = "SELECT * FROM Wall_Unit ORDER BY ID"
    'response.write SearchString  & "<BR>"
      Set DB_Results = DB.Execute(SearchString)
%>


<%'   <td colspan="2" height="10" align="middle">     %>
   <form action="WU_DeleteInsert2.asp" id="WallUnitDeleteEntry" method="post" name="Form1">
 <% ' <br>

  '  <td width="120" valign="top">

  '  </td>
  '  <td width="100%" valign="top">
%>

<div align="left"><table bgcolor=#ccffff border="1" cellPadding="2" cellSpacing="1" width="100%"
ID="Table4">
 <tr bgcolor=lightblue>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>ID</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Name</b></font></p></td>
<%'    <td><p align="center"><font color="red" face="Arial" size="3"><b>PW</b></font></p>  </td>      %>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Location</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Top/Bottom</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Delete?</b></font></p></td>
```

```
  </tr>
      <%    while not DB_Results.eof

            dim top

            if (DB_Results("statusHIGH") = "ON") then
                  top = "Top"
            else
                  top = "Bottom"
            end if
      %>
  <tr bgcolor=white>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
<%'    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("PW")%><%'
</font></p></td>        %>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Location")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%response.Write (top)%>
</font></p></td>
    <td><p align="center"><A
href=\WU_DeleteInsert2.asp?WUID=<%=DB_Results("ID")%>><strong>Yes</strong></A></p></td>
  </tr>

      <%
            DB_Results.MoveNext
      wend
      %>

</div>
    </td>
  </tr>


</td>
</table>
</body>
```

MS_WU_edit2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>

<body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%" ID="Table1">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
      <table border="1" bgcolor=#0033cc width="100%">
      <tr>
            <td  width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
  
      <strong>Wall Units</strong><br>      
            <strong>-></strong><A href=MS_WU_add2.asp><strong>Add Wall Unit
</strong><br></A>      
            <strong>-></strong><A href=MS_WU_Delete2.asp><strong>Delete Wall Unit </strong><br></A>
      
            <strong>-></strong><A href=MS_WU_edit2.asp><strong>Edit Wall Unit </strong><br></A>
  
      <strong>Event Sequences </strong><br>      
```

```
            <strong>-></strong><A href=MS_ES_add2.asp><strong>Add Event Sequence
</strong><br></A>      
            <strong>-></strong><A href=MS_ES_Delete2.asp><strong>Delete Event Sequence
</strong><br></A>      
            <strong>-></strong><A href=MS_ES_edit2.asp><strong>Edit Event Sequence </strong><br></A>
  
      <strong>Immediate Action </strong><br>      
            <stong>-></stong><A href=IAWU2.asp><strong>Immediate Action Commands</strong><br><br></A>

<H2><u>View System</u></H2>
  
      <strong>Wall Units Information</strong><br>      
            <strong>-></strong><A href=IWUPage2.asp><strong>Individual Wall Units</strong><br></A>
      
            <strong>-></strong><A href=AllWU2.asp><strong>All Wall Units</strong><br></A>

  
      <strong>Event Sequences' Information</strong><br>      
            <strong>-></strong><A href=IEventPage2.asp><strong>Individual Event Sequences</strong><br></A>
      
            <strong>-></strong><A href=Events2.asp><strong>All Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

</td>

<% ' ===================================================== WELCOME PAGE
===================================== %>
            <td width="80%" bgcolor=#ccffff>

            <strong>The Table Below Is The Information Of Wall Units</strong><br><br><br>
<p>Click on the Wall Unit To Edit...</p>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString = "SELECT * FROM Wall_Unit ORDER BY ID"
    'response.write SearchString  & "<BR>"
      Set DB_Results = DB.Execute(SearchString)
```

```
%>


 <%'   <td colspan="2" height="10" align="middle">    %>
   <form action="WU_EditInsert.asp" id="WallUnitEditEntry" method="post" name="Form1">
        </form>
<% '  <br>
'   <td width="120" valign="top">
'   </td>
'     <td width="100%" valign="top">      %>


<div align="left"><table bgcolor=#ccffff border="1" cellPadding="2" cellSpacing="1" width="100%"
ID="Table4">



  <tr bgcolor=lightblue>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>ID</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Name</b></font></p></td>
<%'    <td><p align="center"><font color="red" face="Arial" size="3"><b>PW</b></font></p>  </td>      %>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Location</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Top/Bottom</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Edit?</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof

            dim top

            if (DB_Results("statusHIGH") = "ON") then
                  top = "Top"
            else
                  top = "Bottom"
            end if
      %>
  <tr bgcolor=white>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
```

```
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
<%'    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("PW")%><%'
</font></p></td>        %>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Location")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%response.Write (top)%>
</font></p></td>
  <td><p align="center"><A
href=\WU_EditInsert2.asp?WUID=<%=DB_Results("ID")%>><strong>Yes</strong></A></p></td>
  </tr>

      <%
            DB_Results.MoveNext
      wend
      %>


</div>
    </td>
  </tr>
  </form>

            </td>
      </tr>
</body>
```

SearchDone.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
```

```
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%" ID="Table3">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr valign=top>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
```

```
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ====================================================== WELCOME PAGE
================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
<%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

'     thisID = Request.QueryString("WUID")
```

```
'get the information without having it sent here
      'SearchString = "SELECT * FROM Wall_Unit Where Name=''"

    'response.write SearchString  & "<BR>"
      Set DB_Results = DB.Execute("SELECT * FROM Wall_Unit Where Name= '' ")

      'get data not from form
      set pLocation = DB_Results("Location")

' get data from registration form
    set fName = Request.QueryString("SName")
    set fPW = Request.QueryString("SelPW")
    set fLocation = Request.QueryString("SLocation")

      Dim thisID

      if DB_Results.eof then
%>
      <h1>THERE ARE NO NEW WALL UNITS</h1>
<%
      else
%>

<strong>COMPLETE FORM TO ADD A WALL UNIT...</strong>
<br><br><br><br>
<%
      while not DB_Results.eof
      thisID = DB_Results("ID")
%>
<table border="0" width="100%" ID="Table1">
  <tr>
    <td width="120" valign="top"></td>
    <td width="100%" valign="top">
<form action="WU_EditUpdate2.asp?WUID=<%response.Write thisID%>" id="WallUnitEntry" method="post"
name="Form1">
      </form>
      <div align="left"><table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table2">
```

```
        <tr>
          <td><div align="right"><p> Wall Unit ID  </td>
          <td><div align="left"><p><strong><%=DB_Results("ID")%></strong></td>
        </tr>
        <tr>
        <td><div align="right"><p> Wall Unit Name  </td>
                        <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input
value="<%=DB_Results("Name")%>" name="WUName" size="30" ID="Text1"></font></td>
        </tr>
        <tr>
          <td><div align="right"><p> Wall Unit Address  </td>
          <td><div align="left"><p><strong><%=DB_Results("address")%></strong></td>
        </tr>
        <tr>
          <td><div align="right"><p> Wall Unit Location  </td>
                        <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input
value="<%=DB_Results("Location")%>" name="WULocation" size="30" ID="Text4"></font></td>
        </tr>
        <tr>
          <td><div align="right"><p> Wall Unit Top/Bottom  </td>
                        <td>Top<INPUT type="checkbox" ID="Top" NAME="Top"
value="ON">      Bottom<INPUT type="checkbox" ID="Bottom" NAME="Bottom"
value="ON"></td>
             </tr>
        <tr>
          <td><div align="center"><center><p><input id="reset1" name="reset1" type="reset"
value="Reset"></td>
          <td><div align="center"><center><p><input id="submit1" name="submit1" type="submit"
value="Submit"></td>
        </tr>
      </table>
      </div>
    </form>
    </td>
  </tr>
</table>

<%
```

```
	DB_Results.MoveNext
	wend
	end if
%>

			</td>
		</tr>
	</table>
	</body>
```

Searching1.asp

```
<%@ Language=VBScript %>
<HTML>
	<HEAD>
		<TITLE>Menu Page</TITLE>
		<META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
		<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
		<meta name="publisher" content="ScotchInc">
		<meta name="owner" content="ScotchInc">
		<meta name="author" content="ScotchInc">
		<meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
		<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
	</HEAD>
	<body bgcolor=#0066cc>
		<Table bgcolor=#66ccff width="100%">
			<tr>
				<td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
				<td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
			</tr>
		</Table>
		<table border="1" bgcolor="#0033cc" width="100%">
			<tr>
				<td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
					   <strong>Wall Units</strong><br>
```

```
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
```

```
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">

      <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      tempID = Request.QueryString("ESID")
      tempOffset = 100

      set insert = Request.QueryString("add")

      if (insert = 0) then

            InsertString = "INSERT INTO IA_effects_WU (WUID, OPCODE) VALUES (" & "NULL," & tempOffset & ")"
            setupString = "INSERT INTO WebSiteInfo VALUES ('NULL')"

            DB.Execute(InsertString)
      end if
      %>


<h1><i>PROCESSING...</i></h1>
<img align=middle src="s1.bmp">
```

```
      <html>
<head>
<META content="1;url='searching2.asp'" http-equiv=refresh delay="1">

</head>
<body>

<script type=text/javascript>
function goNow() {
  document.location=_l;
}
function setUp() {
  setTimeout("goNow()",_time);// 3 seconds
}
var _l = "/searching2.asp";
var _time = 3000;// msecs.
onLoad='setUp()';
</script>
                        </td>
                  </tr>
            </table>
      </body>
```

Searching2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
```

```
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
```

```
                                                    Event Sequence </strong>
                                              <br>
                                        </A>   <strong>Immediate Action </strong>
                                        <br>
                                              
                                        <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                              <br>
                                        </A>
                                        <H2><u>View System</u></H2>
                                           <strong>Wall Units Information</strong><br>
                                               <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                                    Wall Units</strong><br>
                                        </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                                    Wall Units</strong><br>
                                        </A>   <strong>Event Sequences' Information</strong><br>
                                               <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                                    Event Sequences</strong><br>
                                        </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                                    Event Sequences</strong></A>
                                        <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                                  </td>
                                  <% ' ====================================================== WELCOME PAGE
==================================== %>
                                  <td align="center" width="80%" bgcolor="#ccffff">
<h1><i>PROCESSING...</i></h1>
<img align=middle src="s2.bmp">

      <html>
<head>
<!--
This file retrieved from the JS-Examples archives
http://www.js-examples.com
```

```
1000s of free ready to use scripts, tutorials, forums.
Author: JS-Examples - http://www.js-examples.com/
-->

<META content="1;url='searching3.asp'" http-equiv=refresh delay="1">


</head>
<body>

<script type=text/javascript>
function goNow() {
  document.location=_l;
}
function setUp() {
  setTimeout("goNow()",_time);// 3 seconds
}
//var _l = "http://www.js-examples.com/js/";
var _l = "/searching3.asp";
var _time = 3000;// msecs.
onLoad='setUp()';
</script>

<!--<center><a href='searching2.asp'>JS-Examples.com</a></center> -->





                        </td>
                  </tr>
            </table>
      </body>
```

Searching3.asp

```
<%@ Language=VBScript %>
```

```
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
```

```
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
```

```
                        <% ' ====================================================== WELCOME PAGE
====================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
<h1><i>PROCESSING...</i></h1>
<img align=middle src="s3.bmp">

      <html>
<head>
<META content="1;url='searching4.asp'" http-equiv=refresh delay="1">

</head>
<body>

<script type=text/javascript>
function goNow() {
  document.location=_l;
}
function setUp() {
  setTimeout("goNow()",_time);// 3 seconds
}
var _l = "/searching4.asp";
var _time = 3000;// msecs.
onLoad='setUp()';
</script>
                        </td>
                  </tr>
            </table>
      </body>
```

Searching4.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
```

```
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
```

```
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ====================================================== WELCOME PAGE
================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
<h1><i>PROCESSING...</i></h1>
<img align=middle src="s4.bmp">
```

```
      <html>
<head>
<META content="1;url='searching5.asp'" http-equiv=refresh delay="1">

</head>
<body>

<script type=text/javascript>
function goNow() {
  document.location=_l;
}
function setUp() {
  setTimeout("goNow()",_time);// 3 seconds
}
var _l = "/searching5.asp";
var _time = 3000;// msecs.
onLoad='setUp()';
</script>
                        </td>
                  </tr>
            </table>
      </body>
```

Searching5.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
```

```
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
```

```
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ====================================================== WELCOME PAGE
==================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
<h1><i>PROCESSING...</i></h1>
<img align=middle src="s5.bmp">

<%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      set finished = DB.Execute("Select text FROM WebSiteInfo")
```

```
      if(finished("text")="Searching") then
%>

      <html>
<head>
<META content="1;url='searching1.asp?add=1'" http-equiv=refresh delay="1">

</head>
<body>

<script type=text/javascript>
function goNow() {
  document.location=_l;
}
function setUp() {
  setTimeout("goNow()",_time);// 3 seconds
}
var _l = "/searching1.asp";
var _time = 3000;// msecs.
onLoad='setUp()';
</script>

<%
      elseIf (finished("text")="done") then
%>

<html>
<head>
<META content="1;url='searchDone.asp?add=1'" http-equiv=refresh delay="1">

</head>
<body>

<script type=text/javascript>
function goNow() {
  document.location=_l;
}
```

```
function setUp() {
  setTimeout("goNow()",_time);// 3 seconds
}
var _l = "/searchDone.asp";
var _time = 3000;// msecs.
onLoad='setUp()';
</script>


<%
      end if
%>






                        </td>
                  </tr>
            </table>
      </body>
```

WU_AddInsert2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
```

```
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%" ID="Table1">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%" ID="Table2">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                      <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                      <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
```

```
                              <br>
                        </A>   <strong>Immediate Action </strong>
                        <br>
                              
                        <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                              <br>
                        </A>
                        <H2><u>View System</u></H2>
                           <strong>Wall Units Information</strong><br>
                               <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                    Wall Units</strong><br>
                        </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                    Wall Units</strong><br>
                        </A>   <strong>Event Sequences' Information</strong><br>
                               <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                    Event Sequences</strong><br>
                        </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                    Event Sequences</strong></A>
                        <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                  </td>
                  <% ' ====================================================== ms_wu_ADD PAGE
==================================== %>
                  <td align="center" width="80%" bgcolor="#ccffff">
                        <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

' get data from registration form
      set fName = Request.Form("WUName")
    set fPW = Request.Form("WUPW")
    set fLocation = Request.Form("WULocation")
' Insert values in database
```

```
    InsertString = "INSERT INTO Wall_Unit (Name, PW, Location)"
    InsertString = InsertString & " VALUES ('" & fName & "', "
    InsertString = InsertString & "'" & fPW & "', "
    InsertString = InsertString & "'" & fLocation & "')"

    DB.Execute(InsertString)
%>
                              <p><h3>Addition Successful !</h3>
                              </p>
                              <br>
                              <br>
                              <br>
                              <p><a href="foundation.asp"><strong>BACK</strong></a></p>
                        </td>
            </table>
      </body>
```

WU_DeleteInsert2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%" ID="Table1">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
```

```
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
```

```
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ======================================================= WELCOME PAGE
====================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff"><br>
                              <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"



' get data from registration form
    set fID = Request.QueryString("WUID")
    'set fAddress = Request.Form("WUAddress")
    'set fPW = Request.Form("WUPW")
    'set fLocation = Request.Form("WULocation")

'     response.Write "hi there "
'     response.Write fID
```

```
' update values in database
    InsertString = "DELETE FROM Wall_Unit WHERE ID = " & fID

    '(Name, address, PW, Location)"
    'InsertString = InsertString & " VALUES ('" & fName & "', "
    'InsertString = InsertString & "'" & fAddress & "', "
    'InsertString = InsertString & "'" & fPW & "', "
    'InsertString = InsertString & "'" & fLocation & "')"

    DB.Execute(InsertString)

%>
                              <p><h3>Deletion Successful !</h3>
                              </p>
                              <br>
                              <br>
                              <br>
                              <a href="MS_WU_Delete2.asp"><strong>BACK</strong></a>
                        </td>
            </table>
      </body>
```

WU_EditInsert2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
```

```
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%" ID="Table3">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr valign=top>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
```

```
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
<%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      thisID = Request.QueryString("WUID")

'get the information without having it sent here
      SearchString = "SELECT * FROM Wall_Unit Where ID="&thisID

    'response.write SearchString  & "<BR>"
```

```
      Set DB_Results = DB.Execute(SearchString)

      'get data not from form
      set pLocation = DB_Results("Location")

' get data from registration form
    set fName = Request.QueryString("SName")
    set fPW = Request.QueryString("SelPW")
    set fLocation = Request.QueryString("SLocation")

    dim topY
    topY = DB_Results("statusHIGH")
%>

<strong>COMPLETE FORM TO EDIT A WALL UNIT...</strong>
<br><br><br><br>

<table border="0" width="100%" ID="Table1">
  <tr>
    <td width="120" valign="top"></td>
    <td width="100%" valign="top">
<form action="WU_EditUpdate2.asp?WUID=<%response.Write thisID%>" id="WallUnitEntry" method="post"
name="Form1">
      </form>
      <div align="left"><table border="1" cellPadding="2" cellSpacing="1" width="100%" ID="Table2">
        <tr>
          <td><div align="right"><p> Wall Unit ID  </td>
          <td><div align="left"><p><strong><%=DB_Results("ID")%></strong></td>
        </tr>
        <tr>
        <td><div align="right"><p> Wall Unit Name  </td>
                        <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input
value="<%=DB_Results("Name")%>" name="WUName" size="30" ID="Text1"></font></td>
        </tr>
        <tr>
          <td><div align="right"><p> Wall Unit Address  </td>
          <td><div align="left"><p><strong><%=DB_Results("address")%></strong></td>
        </tr>
```

```
        <tr>
          <td><div align="right"><p> Wall Unit Location  </td>
                        <td><div align="left"><p><font color="#000000" face="Arial" size="3"><input
value="<%=DB_Results("Location")%>" name="WULocation" size="30" ID="Text4"></font></td>
        </tr>
        <tr>
          <td><div align="right"><p> Wall Unit Top/Bottom  </td>
                        <td>Top<INPUT type="checkbox" ID="Top" NAME="Top"  <%
                        If (topY = "ON") Then
                              Response.Write " checked=""checked"""
                        end If %> value="ON"
                        >      Bottom<INPUT type="checkbox" ID="Bottom"
NAME="Bottom" <%
                        If (topY = "OFF") Then
                              Response.Write " checked=""checked"""
                        end If %> value="ON"></td>
             </tr>
        <tr>
          <td><div align="center"><center><p><input id="reset1" name="reset1" type="reset"
value="Reset"></td>
          <td><div align="center"><center><p><input id="submit1" name="submit1" type="submit"
value="Submit"></td>
        </tr>
      </table>
      </div>
    </form>
    </td>
  </tr>
</table>



                        </td>
                  </tr>
            </table>
      </body>
```

WU_EditUpdate2.asp

```
<%@ Language=VBScript %>
<HTML>
	<HEAD>
		<TITLE>Menu Page</TITLE>
		<META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
		<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
		<meta name="publisher" content="ScotchInc">
		<meta name="owner" content="ScotchInc">
		<meta name="author" content="ScotchInc">
		<meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
		<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
	</HEAD>
	<body>
		<body bgcolor=#0066cc>
		<Table bgcolor=#66ccff width="100%" ID="Table1">
			<tr>
				<td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
				<td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
			</tr>
		</Table>
		<table border="1" bgcolor="#0033cc" width="100%">
			<tr>
				<td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
					   <strong>Wall Units</strong><br>
					       <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
							Wall Unit </strong>
						<br>
					</A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
							Wall Unit </strong>
						<br>
```

```
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
```

```
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">

                        <%
      Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

      tempID = Request.QueryString("WUID")
'     response.Write tempID

' get data from registration form
    set fName = Request.Form("WUName")
    set fLocation = Request.Form("WULocation")
    set top = Request.Form("Top")

      if (top= "on") then
            InsertString4 = "UPDATE Wall_Unit SET statusHIGH = 'ON' WHERE ID="&tempID
            InsertString5 = "UPDATE Wall_Unit SET statusLOW = 'OFF' WHERE ID="&tempID
      else
            InsertString4 = "UPDATE Wall_Unit SET statusHIGH = 'OFF' WHERE ID="&tempID
            InsertString5 = "UPDATE Wall_Unit SET statusLOW = 'ON' WHERE ID="&tempID
      end if

      InsertString = "UPDATE Wall_Unit SET Name ="&"'"& fName&"'" & " WHERE ID="&tempID
      InsertString3 = "UPDATE Wall_Unit SET Location ="& "'"&fLocation &"'" &"WHERE ID="&tempID

    DB.Execute(InsertString)
    DB.Execute(InsertString3)
    DB.Execute(InsertString4)
    DB.Execute(InsertString5)
```

```
%>
<p><h3>Edit Successful!</h3></p>
<br><br><br>
<a href=foundation.asp><strong>BACK</strong></a>



                        </td>
                  </tr>
            </table>
      </body>
```

WUIDList2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
```

```
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
```

```
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ====================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    SearchString1 = "SELECT DISTINCT ID FROM Wall_Unit ORDER BY ID"
'    response.write SearchString  & "<BR>"
    Set DB_Results = DB.Execute(SearchString1)

%>

<strong>Select A Wall Unit ID...</strong>
<br><br>Click on the different cases below:<BR><br><br>

<table border="0" width="50%" height="50%" ID="Table1">
 <tr>

    <td width="50%" valign="top">

<div align="center">
```

```
<table border="1" cellPadding="1" cellSpacing="2" width="50%" ID="Table2">

  <tr>
    <td bgcolor=LightBlue>
      <p align="center"><font color="blue" face="Arial" size="3"><b>IDs</b></font></p></td>

  </tr>
      <%    while not DB_Results.eof  %>
  <tr>
    <td bgcolor=white>
            <p align="center"><A
href=\WUsID2.asp?thisID=<%=DB_Results("ID")%>><strong><%=DB_Results("ID")%></strong></A></p>
    </td>
  </tr>

      <%
            DB_Results.MoveNext
      wend
      %>
</table>

</div>
    </td>
  </tr>
</table></td>
                  </tr>
            </table>
      </body>
```

WUNameList2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
```

```
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
```

```
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ====================================================== WELCOME PAGE
================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <%
    Set DB = Server.CreateObject("ADODB.connection")
```

```
      DB.Open "CB"

    SearchString1 = "SELECT DISTINCT Name FROM Wall_Unit "
'     response.write SearchString  & "<BR>"
    Set DB_Results = DB.Execute(SearchString1)

%>

<strong>Select A Wall Unit Name...</strong>
<br><br>Click on the different cases below:<BR><br><br>

<table border="0" width="50%" height="50%" ID="Table1">
  <tr>

<div align="center">
<table border="1" cellPadding="1" cellSpacing="2" width="50%" ID="Table2">

  <tr>
    <td bgcolor=lightblue>
      <p align="center"><font color="blue" face="Arial" size="3"><b>Names</b></font></p></td>

  </tr>
      <%    while not DB_Results.eof  %>
  <tr>
    <td bgcolor=white>
            <p align="center"><A
href=\WUsName2.asp?thisName=<%=DB_Results("Name")%>><strong><%=DB_Results("Name")%></strong></A></p>
    </td>
  </tr>

      <%
            DB_Results.MoveNext
      wend
      %>
</table></td>
                  </tr>
            </table>
      </body>
```

WUsID2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
```

```
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
```

```
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ========================================================= WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Information Of Wall Units With ID = </strong>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    Dim pageID
    pageID = Request.QueryString("thisID")
    response.Write pageID

   Set DB_Results = DB.Execute("SELECT * FROM Wall_Unit WHERE ID="& pageID)
%>
<table border="0" width="100%" height="100%" ID="Table1">
  <tr>
    <td colspan="2" height="10" align="middle">

  <br></td>
  </tr>
 <tr>
    <td width="120" valign="top">

    </td>
    <td width="100%" valign="top">

<div align="left">
<table border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table2">

  <tr bgcolor=lightblue>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>ID</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Name</b></font></p></td>
```

```
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>PW</b></font></p>  </td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Location</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Top/Bottom</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof

            dim top

            if (DB_Results("statusHIGH") = "ON") then
                  top = "Top"
            else
                  top = "Bottom"
            end if

      %>
<tr bgcolor=white>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("PW")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Location")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%response.Write (top)%>
</font></p></td>
</tr>

      <%
            DB_Results.MoveNext
      wend
      %>
</table>
<br><br><br>
<a href=IWUpage2.asp><srtong>BACK</srtong></a>
</td>
                  </tr>
            </table>
      </body>
```

WUsName2.asp

```
<%@ Language=VBScript %>
<HTML>
      <HEAD>
            <TITLE>Menu Page</TITLE>
            <META http-equiv="PICS-Label" content='(PICS-1.1 "http://www.icra.org/ratingsv02.html" l gen
true for "http://www.elizabeth-carson.com" r (ca 1 lz 1 na 1 nb 1 nc 1 nd 1 ne 1 nf 1 ng 1 nh 1 oz 1 vz 1)
"http://www.rsac.org/ratingsv01.html" l gen true for "http://www.elizabeth-carson.com" r (n 4 s 4 v 0 l
0))'>
            <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
            <meta name="publisher" content="ScotchInc">
            <meta name="owner" content="ScotchInc">
            <meta name="author" content="ScotchInc">
            <meta name="copyright" content="Copyright(c)2003 by ScotchInc / all rights reserved">
            <meta name="GENERATOR" content="Microsoft FrontPage 5.0">
      </HEAD>
      <body bgcolor=#0066cc>
            <Table bgcolor=#66ccff width="100%">
                  <tr>
                        <td bgcolor=#66ccff><font size="14" face="TimesNewRoman">Scotch Inc.</font></td>
                        <td bgcolor=#66ccff align=right height="12"><IMG alt="flag" src="flagus.gif"></td>
                  </tr>
            </Table>
            <table border="1" bgcolor="#0033cc" width="100%">
                  <tr>
                        <td width="25%" colspan="15" bgcolor="#99ffff"><H2><u>Manage System</u></H2>
                                 <strong>Wall Units</strong><br>
                                     <strong>-></strong><A
href="MS_WU_add2.asp"><strong>Add
                                          Wall Unit </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_WU_Delete2.asp"><strong>Delete
                                          Wall Unit </strong>
                                    <br>
```

```
                              </A>       <strong>-></strong><A
href="MS_WU_edit2.asp"><strong>Edit
                                          Wall Unit </strong>
                                    <br>
                              </A>   <strong>Event Sequences </strong>
                              <br>
                                     <strong>-></strong><A
href="MS_ES_add2.asp"><strong>Add
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_Delete2.asp"><strong>Delete
                                          Event Sequence </strong>
                                    <br>
                              </A>       <strong>-></strong><A
href="MS_ES_edit2.asp"><strong>Edit
                                          Event Sequence </strong>
                                    <br>
                              </A>   <strong>Immediate Action </strong>
                              <br>
                                    
                              <stong>-></stong><A href="IAWU2.asp"><strong>Immediate Action
Commands</strong><br>
                                    <br>
                              </A>
                              <H2><u>View System</u></H2>
                                 <strong>Wall Units Information</strong><br>
                                     <strong>-></strong><A
href="IWUPage2.asp"><strong>Individual
                                          Wall Units</strong><br>
                              </A>       <strong>-></strong><A
href="AllWU2.asp"><strong>All
                                          Wall Units</strong><br>
                              </A>   <strong>Event Sequences' Information</strong><br>
                                     <strong>-></strong><A
href="IEventPage2.asp"><strong>Individual
                                          Event Sequences</strong><br>
```

```
                              </A>       <strong>-></strong><A
href="Events2.asp"><strong>All
                                          Event Sequences</strong></A>
                              <br>   <strong><A href="errorpage.asp">Error Log</strong></a><br>

                        </td>
                        <% ' ========================================================== WELCOME PAGE
=================================== %>
                        <td align="center" width="80%" bgcolor="#ccffff">
                              <strong>The Table Below Is The Information Of Wall Units With Name =
</strong>
<%
    Set DB = Server.CreateObject("ADODB.connection")
      DB.Open "CB"

    Dim pageName
    pageName = Request.QueryString("thisName")
    response.Write pageName

   Set DB_Results = DB.Execute("SELECT * FROM Wall_Unit WHERE Name = "&"'"&pageName&"'")
%>
<table border="0" width="100%" height="100%" ID="Table1">
  <tr>
    <td colspan="2" height="10" align="middle">

  <br></td>
  </tr>
 <tr>
    <td width="120" valign="top">

    </td>
    <td width="100%" valign="top">

<div align="left">
<table border="1" cellPadding="1" cellSpacing="1" width="100%" ID="Table2">

  <tr bgcolor=lightblue>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>ID</b></font></p></td>
```

```
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Name</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>PW</b></font></p>  </td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Location</b></font></p></td>
    <td><p align="center"><font color="blue" face="Arial" size="3"><b>Top/Bottom</b></font></p></td>
  </tr>
      <%    while not DB_Results.eof

            dim top

            if (DB_Results("statusHIGH") = "ON") then
                  top = "Top"
            else
                  top = "Bottom"
            end if

      %>
<tr bgcolor=white>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("ID")%></font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Name")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("PW")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%=DB_Results("Location")%>
</font></p></td>
    <td><p align="center"><font color="#000000" face="Arial" size="3"><%response.Write (top)%>
</font></p></td>
</tr>
      <%
            DB_Results.MoveNext
            wend
      %>
</table>
<br><br><br>
<a href=IWUpage2.asp><srtong>BACK</srtong></a>

</td>
                  </tr>
            </table>
```

</body>

Senior Design:

Appendix B

Welcome.asp

```
<%@ language=VBScript%>
<% 'Document generated by MobileDev 3.00 %>
<% 'for Microsoft Active Application Server (ASP) %>
<% 'on Tuesday, Apr 22 2003 at 14:13:24 %>

<% '@ENVIRONMENT:Begin %>
<% Response.ContentType = "text/vnd.wap.wml" %>
<% '@ENVIRONMENT:End %>

<% '@QUERYSTRING:Begin %>
<%
  Dim WallUnit

  WallUnit = Request.QueryString("WallUnit")
%>
<% '@QUERYSTRING:End %>

<% '@PROCESSING:Begin %>
<!--  #include file="Welcome.tmpl" -->
<% '@PROCESSING:End %>
```

Welcome.tmpl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
      "http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- Document generated by MobileDev 3.00 -->
<!-- for a Nokia 7110  WML 1.1 browser -->
<!-- on Tuesday, Apr 22 2003 at 14:13:24 -->

<wml>
  <head>
    <meta http-equiv="Cache-Control" content="max-age=0"/>
  </head>
  <card id="Welcome" title="wirelessWAP">
    <p id="Welcome:" mode="wrap">
      Welcome to the wirelessly controlled outlets WAP page
      <a id="Welcome:Continue" href="/cgi-
bin/wirelessWAP/GetWallUnit.asp">Continue</a>
    </p>
  </card>
</wml>
```

WallUnit.asp

```
<%@ language=VBScript%>
<% 'Document generated by MobileDev 3.00 %>
<% 'for Microsoft Active Application Server (ASP) %>
<% 'on Wednesday, Apr 23 2003 at 11:30:04 %>

<% '@ENVIRONMENT:Begin %>
<% Response.ContentType = "text/vnd.wap.wml" %>
<% '@ENVIRONMENT:End %>
```

```
<% '@QUERYSTRING:Begin %>
<% '@QUERYSTRING:End %>

<% '@PROCESSING:Begin %>

<!--  #include file="WallUnit.tmpl" -->
<% '@PROCESSING:End %>
```

WallUnit.tmpl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
      "http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- Document generated by MobileDev 3.00 -->
<!-- for a Nokia 7110  WML 1.1 browser -->
<!-- on Wednesday, Apr 23 2003 at 11:30:04 -->

<wml>
  <head>
    <meta http-equiv="Cache-Control" content="max-age=0"/>
  </head>
  <card id="WallUnit" title="wirelessWAP">
    <do type="prev"><prev/></do>
    <p id="WallUnit:" mode="wrap">
    <%
    Set DB = Server.CreateObject("ADODB.connection")
    DB.Open "CB"

    Set DB_Results = DB.Execute("SELECT ID, Name FROM Wall_Unit")
    %>
    Please Select The Wall Unit you want to Turn on or off<br></br>

    <% while not DB_Results.eof %>
    <A href =" /cgi-bin/wirelessWAP/OutletState.asp?WUID=
<%=DB_Results("ID")%> "> <%=DB_Results("Name")%> </A><br></br>
    <%
    DB_Results.MoveNext
    wend
    DB_Results.Close
    DB.Close
    %>
    </p>
  </card>
</wml>
```

OutletState.asp

```
<%@ language=VBScript%>
<% 'Document generated by MobileDev 3.00 %>
<% 'for Microsoft Active Application Server (ASP) %>
<% 'on Tuesday, Apr 22 2003 at 21:11:45 %>

<% '@ENVIRONMENT:Begin %>
<% Response.ContentType = "text/vnd.wap.wml" %>
```

```
<% '@ENVIRONMENT:End %>

<% '@QUERYSTRING:Begin %>

<% '@QUERYSTRING:End %>

<% '@PROCESSING:Begin %>
<!--  #include file="OutletState.tmpl" -->
<% '@PROCESSING:End %>
```

OutletState.tmpl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
      "http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- Document generated by MobileDev 3.00 -->
<!-- for a Nokia 7110  WML 1.1 browser -->
<!-- on Tuesday, Apr 22 2003 at 21:11:45 -->

<wml>
  <head>
    <meta http-equiv="Cache-Control" content="max-age=0"/>
  </head>
  <template>
    <do type="prev"><prev/></do>
  </template>
  <card id="OPCODE" title="wirelessWAP">
    <p id="OPCODE:">

      On of Off<select name="OPCODE" title="On or Off?"
value="2"><option title="OK" value="2">Turn On</option><option
title="OK" value="3">Turn Off</option></select>
      <br></br>
      <a id="OPCODE:Continue" href="/cgi-
bin/wirelessWAP/GetOutlet.asp?OPCODE=$(OPCODE)&amp;WUID=<%=Request.Quer
yString("WUID")%>">Continue</a>
    </p>
  </card>

</wml>
```

GetOutlet.asp

```
<%@ language=VBScript%>
<% 'Document generated by MobileDev 3.00 %>
<% 'for Microsoft Active Application Server (ASP) %>
<% 'on Tuesday, Apr 22 2003 at 21:12:59 %>

<% '@ENVIRONMENT:Begin %>
<% Response.ContentType = "text/vnd.wap.wml" %>
<% '@ENVIRONMENT:End %>

<% '@QUERYSTRING:Begin %>

<% '@QUERYSTRING:End %>
```

```
<% '@PROCESSING:Begin %>
<!--  #include file="GetOutlet.tmpl" -->
<% '@PROCESSING:End %>
```

GetOutlet.tmpl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
      "http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- Document generated by MobileDev 3.00 -->
<!-- for a Nokia 7110  WML 1.1 browser -->
<!-- on Tuesday, Apr 22 2003 at 21:12:59 -->

<wml>
  <head>
    <meta http-equiv="Cache-Control" content="max-age=0"/>
  </head>
  <template>
    <do type="prev"><prev/></do>
  </template>
  <card id="HILOW" title="wirelessWAP">
    <p id="HILOW:">

      Hi or Low<select name="HILOW" title="Which Outlet?"
value="0"><option title="OK" value="1">Upper Outlet</option><option
title="OK" value="0">Lower Outlet</option></select>
      <br></br>
      <a id="HILOW:Continue" href="/cgi-
bin/wirelessWAP/Finshed.asp?HILOW=$(HILOW)&amp;WUID=<%=Request.QueryStr
ing("WUID")%>&amp;OPCODE=<%=Request.QueryString("OPCODE")%>">Continue</
a>
    </p>
  </card>

</wml>
```

Finished.asp

```
<%@ language=VBScript%>
<% 'Document generated by MobileDev 3.00 %>
<% 'for Microsoft Active Application Server (ASP) %>
<% 'on Tuesday, Apr 22 2003 at 20:57:55 %>

<% '@ENVIRONMENT:Begin %>
<% Response.ContentType = "text/vnd.wap.wml" %>
<% '@ENVIRONMENT:End %>

<% '@QUERYSTRING:Begin %>
<%
  Dim HILOW, WUID, OPCODE

  HILOW = Request.QueryString("HILOW")
  WUID = Request.QueryString("WUID")
  OPCODE = Request.QueryString("OPCODE")
```

```
%>
<% '@QUERYSTRING:End %>

<% '@PROCESSING:Begin %>
<%
Dim Choices
Set Choices = CreateObject("Scripting.Dictionary")
Dim Action, Choice

Set objConnection = Server.CreateObject("ADODB.Connection")
objConnection.Open "Provider=MSDASQL.1;Persist Security Info=False;Data
Source=CB;Mode=ReadWrite"
' Prepare query statement...
objConnection.Execute("INSERT INTO IA_effects_WU (WUID,OPCODE,HI_LOW)
VALUES('"&WUID&"','"&OPCODE&"','"&HILOW&"')")
objConnection.Close
Set objConnection = Nothing

%>
<!--  #include file="Finshed.tmpl" -->
<% '@PROCESSING:End %>
```

Finished.tmpl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
      "http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- Document generated by MobileDev 3.00 -->
<!-- for a Nokia 7110  WML 1.1 browser -->
<!-- on Tuesday, Apr 22 2003 at 20:57:55 -->

<wml>
  <head>
    <meta http-equiv="Cache-Control" content="max-age=0"/>
  </head>
  <card id="Finshed" title="wirelessWAP" ontimer="/cgi-
bin/wirelessWAP/GetWallUnit.asp">
  <timer value="50"/>
    <p id="Finshed:">
        Action Completed. <br></br> This page will automaticly refresh
to Wall Unit Selection in 5 seconds.
    </p>
  </card>
</wml>
```

Senior Design:

Appendix C

ControlBox

Report on Configuration DefaultConfig

Overridden Properties
Subjects:
General
Metaclasses:
Model
Properties:
SourceFont: Courier New 16 NoBold NoItalic
ModelCodeAssociativityFineTune: RoundTrip
Graphics
Properties:
ClassBoxFont: Arial 16 NoBold NoItalic
NameFont: Arial 16 NoBold NoItalic
NoteFont: Arial 16 NoBold NoItalic
DiagramConnectorFont: Arial 16 NoBold NoItalic
LabelFont: Arial 16 NoBold NoItalic

PACKAGES

ControlBox

The control-box is the portion of the system that provides an interface between the user and the outlets. The control-box is responsible for recieving user input in the form of events and regulation data, and scheduling the events as well as seeing to it that the regulation of outlets is occomplished. Additionally, the box is resposible for polling the wall outlets for information concerning current and voltage values at regular intervals to be used in compiling graphs and power regulation tasks. The box is also the commander of the network link, meaning that it originates all commands and receives acknowldgement that they have been completed correctly.

Overridden Properties

Subjects:

CG

Metaclasses:

CGGeneral

Properties:

GeneratedCodeInBrowser: True

General

Metaclasses:

Model

Properties:

SourceFont: Courier New 16 NoBold NoItalic

Graphics

Properties:

ClassBoxFont: Arial 16 NoBold NoItalic
NameFont: Arial 16 NoBold NoItalic
NoteFont: Arial 16 NoBold NoItalic
DiagramConnectorFont: Arial 16 NoBold NoItalic
LabelFont: Arial 16 NoBold NoItalic

WebComponents

Metaclasses:

Class

Properties:

WebManaged: False

Operation

Properties:

WebManaged: False

File

Properties:

WebManaged: False

Attribute

Properties:

WebManaged: False

OBJECT MODEL DIAGRAMS:

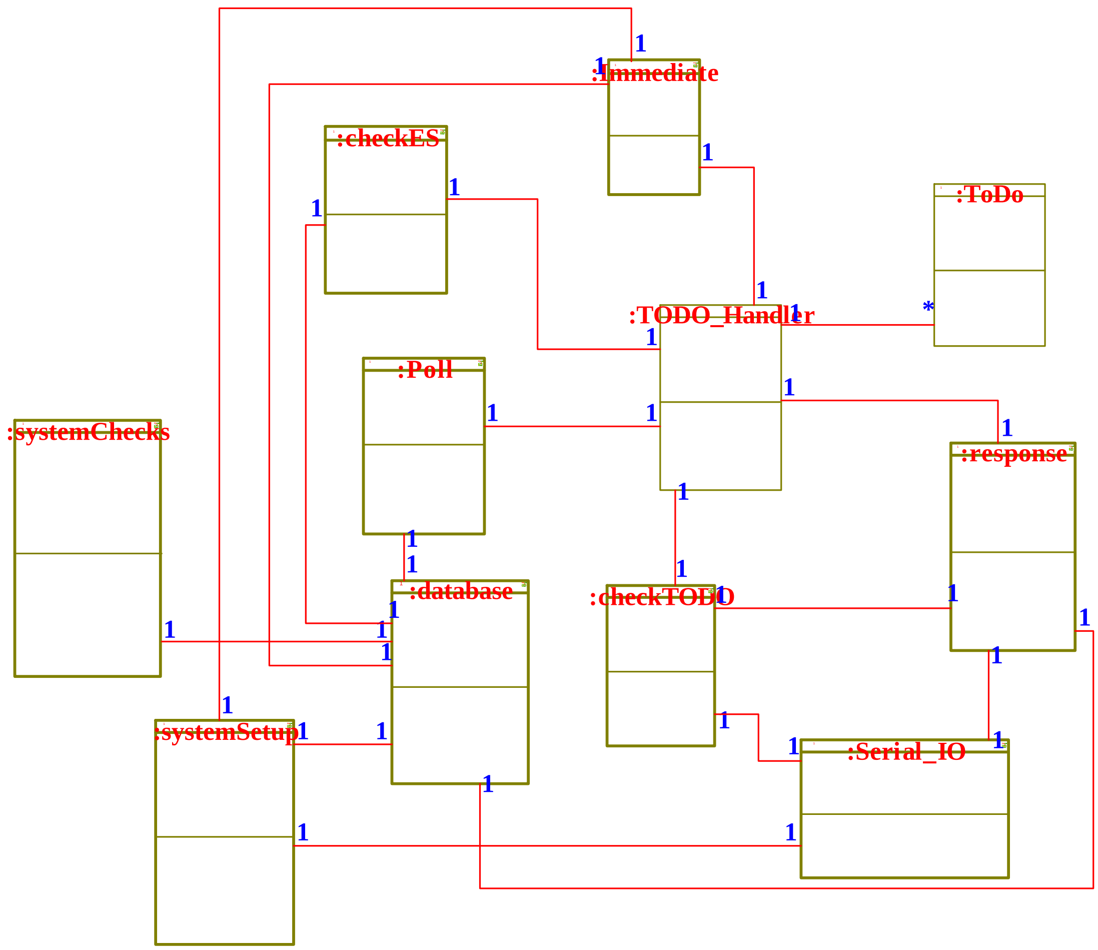
Overridden Properties

Subjects:

General

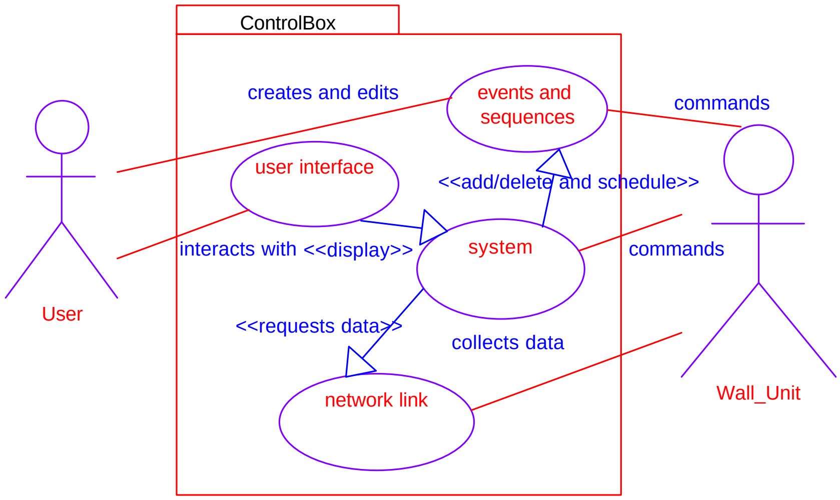
Metaclasses:
Model
Properties:
SourceFont: Courier New 16 NoBold NoItalic
Graphics
Properties:
ClassBoxFont: Arial 16 NoBold NoItalic
NameFont: Arial 16 NoBold NoItalic
NoteFont: Arial 16 NoBold NoItalic
DiagramConnectorFont: Arial 16 NoBold NoItalic
LabelFont: Arial 16 NoBold NoItalic

OMD



USE CASE DIAGRAMS:

UseCase



TYPES

todoattributes

```
struct %s {
long address;
unsigned char OPCODE;
unsigned char HI_LOW;
unsigned char PW;
int attempts;
int offset;
int whatToPoll;
}
```

COMMAND_STRUCTURE

used to create the command to send to the wall units

```
struct %s {
unsigned int StartSymbol        : 8;
unsigned int To                         : 32;
unsigned int From               : 16;
unsigned int OPCODE                     : 3;
unsigned int HI_LOW                     : 1;
unsigned int OFFSet                     : 8;
unsigned int CheckSUM           : 12;
};
```

SentTotals

```
struct %s {
   unsigned long address;
   double v;
   double c1;
   double c2;
};
```

EVENTS:

evConnected

evConnected1

event used to let checkes know when it is oaky to start operation

evConnected2

evConnected3

evDoToDo

evFIND

evReDo

Args:

todoattributes tempAtts

evResponseNeeded

evSetup

evWaitforResponse

Args:

todoattributes atts

atts of the expected response

GLOBALS:

Relations:

itsCheckES

Composition of checkES, Multiplicity of 1, Uni-directional

itsPoll

Composition of Poll, Multiplicity of 1, Uni-directional

itsTODO_Handler

Composition of TODO_Handler, Multiplicity of 1, Uni-directional

itsToDo

Composition of ToDo, Multiplicity of 1, Uni-directional

itsImmediate

Composition of Immediate, Multiplicity of 1, Uni-directional

itsCheckTODO

Composition of checkTODO, Multiplicity of 1, Uni-directional

itsResponse

Composition of response, Multiplicity of 1, Uni-directional

itsSerial_IO

Composition of Serial_IO, Multiplicity of 1, Uni-directional

itsDatabase

Composition of database, Multiplicity of 1, Uni-directional

itsSystemSetup

Composition of systemSetup, Multiplicity of 1, Uni-directional

itsSystemChecks

Composition of systemChecks, Multiplicity of 1, Uni-directional

Instantiated Relations:

itsTODO_Handler
of itsCheckES with itsTODO_Handler
itsCheckES
of itsTODO_Handler with itsCheckES
itsTODO_Handler
of itsPoll with itsTODO_Handler
itsPoll
of itsTODO_Handler with itsPoll
itsToDo
of itsTODO_Handler with itsToDo
itsTODO_Handler
of itsToDo with itsTODO_Handler
itsTODO_Handler
of itsImmediate with itsTODO_Handler
itsImmediate
of itsTODO_Handler with itsImmediate
itsCheckTODO
of itsTODO_Handler with itsCheckTODO
itsTODO_Handler
of itsCheckTODO with itsTODO_Handler
itsResponse
of itsCheckTODO with itsResponse
itsCheckTODO
of itsResponse with itsCheckTODO
itsResponse
of itsTODO_Handler with itsResponse
itsTODO_Handler
of itsResponse with itsTODO_Handler
itsSerial_IO
of itsCheckTODO with itsSerial_IO
itsCheckTODO
of itsSerial_IO with itsCheckTODO
itsSerial_IO
of itsResponse with itsSerial_IO
itsResponse
of itsSerial_IO with itsResponse
itsDatabase
of itsCheckES with itsDatabase
itsCheckES
of itsDatabase with itsCheckES
itsDatabase
of itsPoll with itsDatabase
itsPoll
of itsDatabase with itsPoll
itsDatabase
of itsResponse with itsDatabase
itsResponse
of itsDatabase with itsResponse
itsDatabase
of itsImmediate with itsDatabase
itsImmediate
of itsDatabase with itsImmediate
itsDatabase
of itsSystemSetup with itsDatabase
itsSystemSetup

of itsDatabase with itsSystemSetup

itsSerial_IO

of itsSystemSetup with itsSerial_IO

itsSystemSetup

of itsSerial_IO with itsSystemSetup

itsSystemSetup

of itsImmediate with itsSystemSetup

itsImmediate

of itsSystemSetup with itsImmediate

itsDatabase

of itsSystemChecks with itsDatabase

itsSystemChecks

of itsDatabase with itsSystemChecks

Functions:

ControlBox_initRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
itsCheckES = new checkES;
itsCheckTODO = new checkTODO;
itsDatabase = new database;
itsImmediate = new Immediate;
itsPoll = new Poll;
itsResponse = new response;
itsSerial_IO = new Serial_IO;
itsSystemChecks = new systemChecks;
itsSystemSetup = new systemSetup;
itsTODO_Handler = new TODO_Handler;
itsToDo = new ToDo;
itsCheckES->setItsTODO_Handler(itsTODO_Handler);
itsPoll->setItsTODO_Handler(itsTODO_Handler);
itsImmediate->setItsTODO_Handler(itsTODO_Handler);
itsTODO_Handler->setItsCheckTODO(itsCheckTODO);
itsCheckTODO->setItsResponse(itsResponse);
itsTODO_Handler->setItsResponse(itsResponse);
itsCheckTODO->setItsSerial_IO(itsSerial_IO);
itsResponse->setItsSerial_IO(itsSerial_IO);
itsCheckES->setItsDatabase(itsDatabase);
itsPoll->setItsDatabase(itsDatabase);
itsResponse->setItsDatabase(itsDatabase);
itsImmediate->setItsDatabase(itsDatabase);
itsSystemSetup->setItsDatabase(itsDatabase);
itsSystemSetup->setItsSerial_IO(itsSerial_IO);
itsImmediate->setItsSystemSetup(itsSystemSetup);
itsSystemChecks->setItsDatabase(itsDatabase);
```

ControlBox_startBehavior

Generated , Primitive-operation , Public, Return type is OMBoolean

Body

```
OMBoolean done = FALSE;
itsCheckES->startBehavior();
itsCheckTODO->startBehavior();
itsDatabase->startBehavior();
itsImmediate->startBehavior();
itsPoll->startBehavior();
itsResponse->startBehavior();
itsSerial_IO->startBehavior();
itsSystemChecks->startBehavior();
itsSystemSetup->startBehavior();
return done;
```

CLASSES:

checkES

This class operates at the start of every minute and checks the database to see if there is any new event sequences that are scheduled to be done this minute

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsTODO_Handler

Association with TODO_Handler, Multiplicity of 1, Bi-directional

itsDatabase

Association with database, Multiplicity of 1, Bi-directional

Operations:

__setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
itsDatabase = p_database;
```

__setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
itsTODO_Handler = p_TODO_Handler;
```

_clearItsDatabase

Generated , Primitive-operation , Public, Return type is void

Body

```
itsDatabase = NULL;
```

_clearItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Body

```
itsTODO_Handler = NULL;
```

_setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(itsDatabase != NULL)
    itsDatabase->__setItsCheckES(NULL);
__setItsDatabase(p_database);
```

_setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(itsTODO_Handler != NULL)
    itsTODO_Handler->__setItsCheckES(NULL);
__setItsTODO_Handler(p_TODO_Handler);
```

checkES

Generated , Constructor , Public

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsDatabase != NULL)
    {

        checkES* p_checkES = itsDatabase->getItsCheckES();
        if(p_checkES != NULL)
            itsDatabase->__setItsCheckES(NULL);
        itsDatabase = NULL;
    }
if(itsTODO_Handler != NULL)
    {

        checkES* p_checkES = itsTODO_Handler->getItsCheckES();
        if(p_checkES != NULL)
            itsTODO_Handler->__setItsCheckES(NULL);
        itsTODO_Handler = NULL;
    }
```

evConnected

Event

evConnected1

event used to let checkes know when it is oaky to start operation

Event

getItsDatabase

Generated , Primitive-operation , Public, Return type is 'database*'

Constant

Body

```
return itsDatabase;
```

getItsTODO_Handler

Generated , Primitive-operation , Public, Return type is 'TODO_Handler*'

Constant

Body

```
return itsTODO_Handler;
```

setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(p_database != NULL)
    p_database->_setItsCheckES(this);
_setItsDatabase(p_database);
```

setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(p_TODO_Handler != NULL)
    p_TODO_Handler->_setItsCheckES(this);
_setItsTODO_Handler(p_TODO_Handler);
```

startBehavior

Virtual, Generated , Primitive-operation , Public, Return type is OMBoolean

Body

```
OMBoolean done = FALSE;
done = OMReactive::startBehavior();
if(done)
```

```
        start();
    return done;
```

TimeSeconds

function to find the number of seconds until the start of the next minute

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is int

Body

```
return (CTime::GetCurrentTime().GetSecond());
```

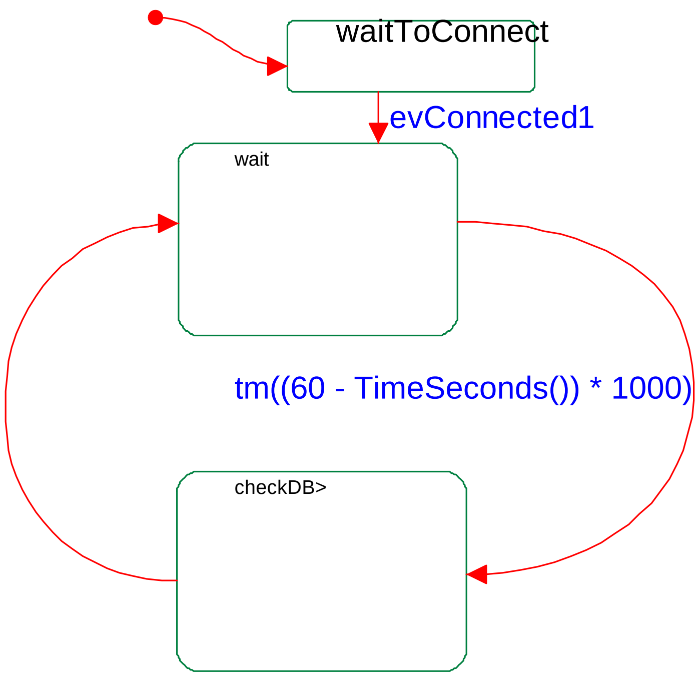
~checkES

Generated , Destructor , Public

Body

```
cleanUpRelations();
```

Statechart



ROOT

Or-state

Substates:

checkDB

wait

waitToConnect

Default Transition

Target:

waitToConnect

checkDB

this function checks the database and checks the start time for each event sequence that exists. If there is something that needs to be initiated then it creates a todo for that function

Or-state

EntryAction

```
CString errorStr;
char exeStr[200];
char timebuf[20];
CString temp;
CRecordset rs( (itsDatabase->getDB()));
CRecordset rs1( (itsDatabase->getDB()));
CRecordset rs2( (itsDatabase->getDB()));
if (itsDatabase->isConnected()) {
    itsDatabase->query(&rs,"SELECT StartTime, ID, StartDate,
DailyRepeat FROM event_sequence");
    CDBVariant varValue;
    CDBVariant OPCODE, DAILY;
    CString tempString;
    CDBVariant password, address, offset, hi_low;
    CString temp;
    TIMESTAMP_STRUCT *ts;
    while (!rs.IsEOF()) {
            rs.GetFieldValue(short(2), varValue);
            if (varValue.m_dwType == DBVT_DATE) {
                    ts = varValue.m_pdate;
                    if( ((*ts).year < CTime::GetCurrentTime().GetYear())
||
                            (       ((*ts).year ==
CTime::GetCurrentTime().GetYear()) &&
                                    ((*ts).month <
CTime::GetCurrentTime().GetMonth()) ) ||
                                    (       ((*ts).year ==
CTime::GetCurrentTime().GetYear()) &&
                                            ((*ts).month ==
CTime::GetCurrentTime().GetMonth()) &&
                                            ((*ts).day <=
CTime::GetCurrentTime().GetDay()) )
                            ) {
                                    cout<<"less than startdate"<<endl;
                                    rs.GetFieldValue(short(0), varValue);
                                    rs.GetFieldValue(short(3), DAILY);
                                    if (varValue.m_dwType == DBVT_DATE) {
                                            ts = varValue.m_pdate;
                                            if((*ts).hour ==
CTime::GetCurrentTime().GetHour()) {
                                                    if((*ts).minute ==
CTime::GetCurrentTime().GetMinute()) {
                                                            cout<<"Daily :
"<<(int)DAILY.m_chVal<<" with and shit : "<<(DAILY.m_chVal & ( 0x1 << (
CTime::GetCurrentTime().GetDayOfWeek() - 1) ) )<<endl;
                                                            if(
(DAILY.m_chVal & ( 0x1 << ( CTime::GetCurrentTime().GetDayOfWeek() - 1)
) ) != 0x00 ) {

    cout<<"damn this biatch works"<<endl;

    rs.GetFieldValue(short(1), tempString);

    itsDatabase->query(&rs1,_T("SELECT WUID, OPCODE, offset, HI_LOW
from ES_effects_WU where ESID = " + tempString));

    //cout<<"does it get past this query"<<endl;

    if(!rs1.IsEOF()) {

    while (!rs1.IsEOF()) {

    rs1.GetFieldValue(short(1), OPCODE);

    rs1.GetFieldValue(short(0), temp);
```

```
    rs1.GetFieldValue(short(2), offset);

    rs1.GetFieldValue(short(3), hi_low);

    cout<<"trying to get offset"<<endl;

    cout<<":::"<<offset.m_lVal<<endl;

    itsDatabase->query(&rs2,_T("SELECT PW, address from Wall_Unit where
ID = " + temp));

    if(!rs2.IsEOF()) {

            rs2.GetFieldValue(short(0), password);

            rs2.GetFieldValue(short(1), address);

            if ( (password.m_dwType == DBVT_UCHAR) && (address.m_dwType
== DBVT_LONG) && (hi_low.m_dwType == DBVT_UCHAR) && (offset.m_dwType ==
DBVT_LONG) ) {

                    //check PW and address are of correct type

                    while (!rs2.IsEOF()) {

                            cout<<"creating a todo"<<endl;

                            itsTODO_Handler-
>NewTODO(address.m_lVal,password.m_chVal,OPCODE.m_chVal,0,offset.m_lVal
,hi_low.m_chVal,5);

                            rs2.MoveNext();

                    }

                    rs2.Close();

            }

            else {

                    //keep track of error

                    //types are not correct

                    /*

                    temp = "ERROR";

                    errorStr = "INSERT into ErrorLog values(3,'Types of
password and address are not correct from CheckES' ,'";

                    temp = timebuf;

                    errorStr = errorStr + CString(temp);

                    errorStr += "' ,NULL)";

                    cout<<errorStr<<endl;

                    */


    strftime(timebuf,20,"%c",CTime::GetCurrentTime().GetLocalTm());

                    (void)sprintf(exeStr,"INSERT into ErrorLog
values(3,'Types of password and address are not correct from CheckES'
,'%s',NULL)",timebuf);

                    itsDatabase->execute(exeStr);
```

```
        }

    }

    rs1.MoveNext();

    }

    rs1.Close();
                                                            }
                                                            else {

    //there is no es_effects_wu so send error and delete the es

    /*

    cout<<"here we have the error"<<endl;

    temp = "ERROR";

    errorStr = "INSERT into ErrorLog values(5,'Event sequence is not
complete deleting' ,'";

    temp = timebuf;

    errorStr = errorStr + CString(temp);

    errorStr += "' ,NULL)";

    cout<<errorStr<<endl;

    */

    strftime(timebuf,20,"%c",CTime::GetCurrentTime().GetLocalTm());

    (void)sprintf(exeStr,"INSERT into ErrorLog values(5,'Event sequence
is not complete deleting' ,'%s',NULL)",timebuf);

    itsDatabase->execute(exeStr);

    (void)sprintf(exeStr,"DELETE * FROM event_sequence WHERE ID =
%s",tempString);

    //cout<<"deleting non repeated es : "<<exeStr<<endl;

    itsDatabase->execute(exeStr);
                                                            }

                                                            if (
(DAILY.m_chVal & 0x80) != 0x80) {

    //need to delete the event sequence cuase its not a repeater

    (void)sprintf(exeStr,"DELETE * FROM ES_effects_WU WHERE ESID =
%s",tempString);

    //cout<<"deleting non repeated es : "<<exeStr<<endl;

    itsDatabase->execute(exeStr);

    (void)sprintf(exeStr,"DELETE * FROM event_sequence WHERE ID =
%s",tempString);

    //cout<<"deleting non repeated es : "<<exeStr<<endl;

    itsDatabase->execute(exeStr);
                                                            }

                                                    }
                                            }
```

```
                                        }
                                }
                                else {
                                        //track error
                                        //type of StartTime is not
correct
                                        /*
                                        temp = "ERROR";
                                        errorStr = "INSERT into
ErrorLog values(3,'Type of StartTime are not correct from CheckES' ,'";
                                        temp = timebuf;
                                        errorStr = errorStr +
CString(temp);
                                        errorStr += "' ,NULL";
                                        cout<<errorStr<<endl;
                                        */

    strftime(timebuf,20,"%c",CTime::GetCurrentTime().GetLocalTm());
                                        (void)sprintf(exeStr,"INSERT
into ErrorLog values(3,'Type of StartTime are not correct from CheckES'
,'%s',NULL)",timebuf);
                                        itsDatabase->execute(exeStr);
                                }
                        }
                }
                else {
                        //track error
                        //type of StartDate is not correct
                        /*
                        temp = "ERROR";
                        errorStr = "INSERT into ErrorLog values(3,'Type of
StartDate are not correct from CheckES' ,'";
                        temp = timebuf;
                        errorStr = errorStr + CString(temp);
                        errorStr += "' ,NULL";
                        cout<<errorStr<<endl;
                        */

    strftime(timebuf,20,"%c",CTime::GetCurrentTime().GetLocalTm());
                        (void)sprintf(exeStr,"INSERT into ErrorLog
values(3,'Type of StartDate are not correct from CheckES'
,'%s',NULL)",timebuf);
                        itsDatabase->execute(exeStr);
                }
    rs.MoveNext();
    }
}
else {
//write error file to db
//CODE 01 could not operate select statement "Select StartTime, ID,
StartDate FROM event_Sequence"
strftime(timebuf,20,"%c",CTime::GetCurrentTime().GetLocalTm());
(void)sprintf(exeStr,"INSERT into ErrorLog values(1,'could not operate
select statement |Select StartTime, ID, StartDate FROM event_Sequence|'
,'%s',NULL)",timebuf);
itsDatabase->execute(exeStr);

}
rs.Close();
```

Out Transition

<u>Target:</u>

wait

wait

temp state until the minute starts

Or-state

Out Transition

```
tm((60 - TimeSeconds()) * 1000)
```

Target:

checkDB

waitToConnect

dummy state to allow for an event to start the functionality of the class

Or-state

Out Transition

`evConnected1`

Target:

wait

---

checkTODO

this class checks to see if there are todos in the queue and if there are it sends them out to the wall units and creates a response queue

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsTODO_Handler

Association with TODO_Handler, Multiplicity of 1, Bi-directional

itsResponse

Association with response, Multiplicity of 1, Bi-directional

itsSerial_IO

Association with Serial_IO, Multiplicity of 1, Bi-directional

Operations:

__setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
itsResponse = p_response;
```

__setItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Args:

'Serial_IO*' p_Serial_IO

Body

```
itsSerial_IO = p_Serial_IO;
```

__setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
itsTODO_Handler = p_TODO_Handler;
```

_clearItsResponse

Generated , Primitive-operation , Public, Return type is void

Body

```
itsResponse = NULL;
```

_clearItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Body

```
itsSerial_IO = NULL;
```

_clearItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Body

```
itsTODO_Handler = NULL;
```

_setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
if(itsResponse != NULL)
    itsResponse->__setItsCheckTODO(NULL);
__setItsResponse(p_response);
```

_setItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Args:

'Serial_IO*' p_Serial_IO

Body

```
if(itsSerial_IO != NULL)
    itsSerial_IO->__setItsCheckTODO(NULL);
__setItsSerial_IO(p_Serial_IO);
```

_setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(itsTODO_Handler != NULL)
    itsTODO_Handler->__setItsCheckTODO(NULL);
__setItsTODO_Handler(p_TODO_Handler);
```

BuildCommand

This function builds a command that is to be sent out to a device. There are special ways the command needs to be formatted so it needs to be completed before sending out to a device

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is 'unsigned char *'

Args:

'COMMAND_STRUCTURE * %s' COMMAND

Body

```
#define byteswap32(a)
((a<<24)&0xff000000)|((a<<8)&0x00ff0000)|((a>>8)&0x0000ff00)|((a>>24
)&0x000000ff)
#define byteswap16(a) ((a>>8)&0x00ff)|((a<<8)&0xff00)

//char tmpStr[10];
//tmpStr[0] = 'E';
tmpStr[0] = (COMMAND->StartSymbol & 0xFF);
*((long*)&tmpStr[1]) = ((long)byteswap32(COMMAND->To));
*((int*)&tmpStr[5]) = ((int)byteswap16(COMMAND->From));
tmpStr[7] = (char)((COMMAND->OPCODE << 5) & 0xE0);
tmpStr[7] |= (char)((COMMAND->HI_LOW << 4) & 0x10);
tmpStr[7] |= (char)((COMMAND->OFFSet >> 4) & 0x0F);
tmpStr[8] = (char)((COMMAND->OFFSet << 4) & 0xF0);
tmpStr[8] |= (char)((COMMAND->CheckSUM >> 8) & 0x0F);
```

```
unsigned char byte = 0;
for(int i=1;i<9;i++) {
 byte = byte ^ (unsigned char)tmpStr[i];
 //cout<<"byte : "<<byte<<" | (unsigned char)ToSend[i]) : "<< (
(unsigned char)ToSend[i] ) <<endl;
}

(unsigned char)tmpStr[9] = byte;

printf("command: ");
for(i = 0; i < 10; i++) {
printf("%.02x ",tmpStr[i]);
}
printf("");
return tmpStr;
```

checkTODO

Generated , Constructor , Public

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsResponse != NULL)
    {

        checkTODO* p_checkTODO = itsResponse->getItsCheckTODO();
        if(p_checkTODO != NULL)
            itsResponse->__setItsCheckTODO(NULL);
        itsResponse = NULL;
    }
if(itsSerial_IO != NULL)
    {

        checkTODO* p_checkTODO = itsSerial_IO->getItsCheckTODO();
        if(p_checkTODO != NULL)
            itsSerial_IO->__setItsCheckTODO(NULL);
        itsSerial_IO = NULL;
    }
if(itsTODO_Handler != NULL)
    {

        checkTODO* p_checkTODO = itsTODO_Handler->getItsCheckTODO();
        if(p_checkTODO != NULL)
            itsTODO_Handler->__setItsCheckTODO(NULL);
        itsTODO_Handler = NULL;
    }
```

evDoToDo

Event

evReDo

Event

evResponseNeeded

Event

getAtts

Generated , Primitive-operation , Public, Return type is todoattributes

Constant

Body

```
return atts;
```

getCOMMAND

Generated , Primitive-operation , Public, Return type is COMMAND_STRUCTURE

Constant

Body

```
return COMMAND;
```

getItsResponse

Generated , Primitive-operation , Public, Return type is 'response*'

Constant

Body

```
return itsResponse;
```

getItsSerial_IO

Generated , Primitive-operation , Public, Return type is 'Serial_IO*'

Constant

Body

```
return itsSerial_IO;
```

getItsTODO_Handler

Generated , Primitive-operation , Public, Return type is 'TODO_Handler*'

Constant

Body

```
return itsTODO_Handler;
```

getTmpStr

Generated , Primitive-operation , Public, Return type is 'unsigned char'

Args:

'int' i1

Constant

Body

```
return tmpStr[i1];
```

setAtts

Generated , Primitive-operation , Public, Return type is todoattributes

Args:

todoattributes p_atts

Body

```
atts = p_atts;
```

setCOMMAND

Generated , Primitive-operation , Public, Return type is COMMAND_STRUCTURE

Args:

COMMAND_STRUCTURE p_COMMAND

Body

```
COMMAND = p_COMMAND;
```

setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
if(p_response != NULL)
    p_response->_setItsCheckTODO(this);
_setItsResponse(p_response);
```

setItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Args:

'Serial_IO*' p_Serial_IO

Body

```
if(p_Serial_IO != NULL)
    p_Serial_IO->_setItsCheckTODO(this);
_setItsSerial_IO(p_Serial_IO);
```

setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(p_TODO_Handler != NULL)
    p_TODO_Handler->_setItsCheckTODO(this);
_setItsTODO_Handler(p_TODO_Handler);
```

setTmpStr

Generated , Primitive-operation , Public, Return type is 'void'

Args:

'int' i1

'unsigned char' p_tmpStr

Body

```
tmpStr[i1] = p_tmpStr;
```

startBehavior

Virtual, Generated , Primitive-operation , Public, Return type is OMBoolean

Body

```
OMBoolean done = FALSE;
done = OMReactive::startBehavior();
if(done)
    start();
return done;
```

ToDosExist

function used to see if there are any todos in the queue that need to be completed
true - yes
false - no

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is OMBoolean

Body

```
if(itsTODO_Handler->getTODOcount() > 0) {
 atts = itsTODO_Handler->getNextTODO();
 cout<<"cool"<<endl;
 //cout<<"atts : addr: "<<atts.address<<" PW: "<<atts.PW<<" OPCODE:
"<<atts.OPCODE<<endl;
 return true;
}
return false;
```

~checkTODO

Generated , Destructor , Public

Body

```
cleanUpRelations();
```

Attributes:

atts

a structure holding all the important information about a todo.

Type of todoattributes, Public

COMMAND

the command to send to a device

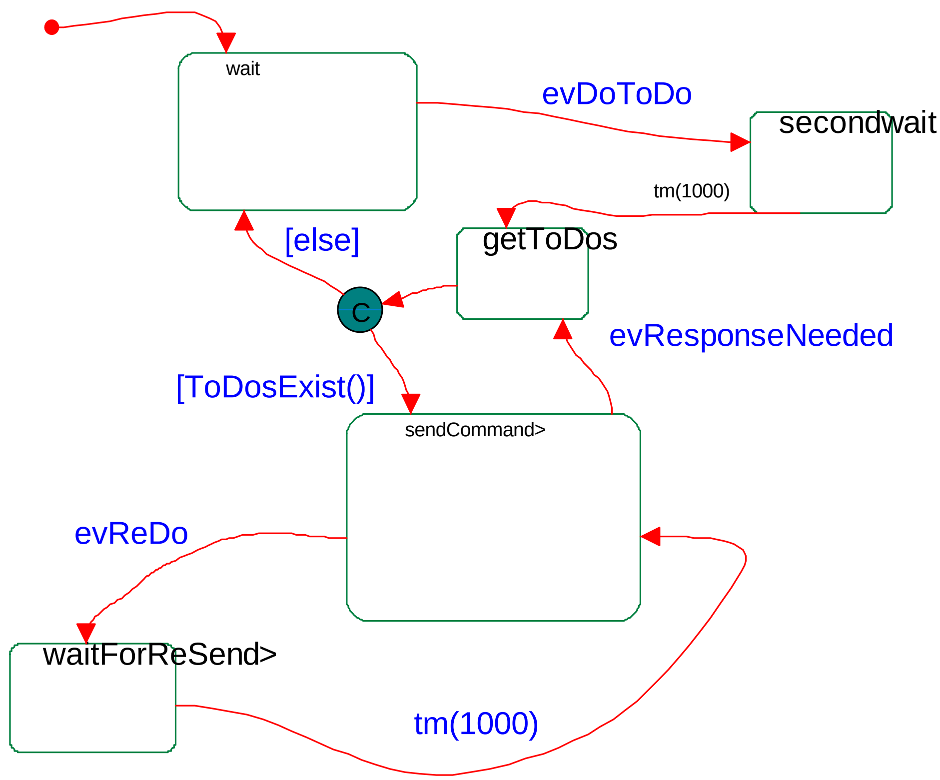
Type of COMMAND_STRUCTURE, Public

tmpStr

a temerary string used for various functions

Type of 'unsigned char %s[12];', Public

Statechart



ROOT

Or-state

<u>Substates:</u>

getToDos

secondwait

sendCommand

wait

waitForReSend

Default Transition

<u>Target:</u>

wait

getToDos

temp state to allow ToDosExist to obtain next todo

Or-state

Out Transition

Condition Connector

<u>Branches:</u>

`[ToDosExist()]`

Target:

sendCommand

`[else]`

Target:

wait

secondwait

another temp state to allow for a one second delay between sending out commands to devices to lower the ammount of information being sent

Or-state

Out Transition

```
tm(1000)
```

Target:

getToDos

sendCommand

create and send command over RS232
add to response queue

Or-state

EntryAction

```
cout<<"we have a todo"<<endl;
int RecieveType;
unsigned char * ToSend;

//while(!itsSerial_IO->getReady()){/*cant do anything until serial port
is open*/}
cout<<"build command"<<endl;
//need to build the command
cout<<"opcode:::"<<atts.OPCODE<<endl;
cout<<"offset:::"<<atts.offset<<endl;
COMMAND.StartSymbol = 0xDB;
COMMAND.To = atts.address;
COMMAND.From = 0x0001;
COMMAND.OPCODE = ((atts.OPCODE)&0x7);
COMMAND.HI_LOW = (atts.HI_LOW)&0x1;
if (COMMAND.OPCODE == 0x4) {
COMMAND.OFFSet = atts.offset;
}
else {
COMMAND.OFFSet = 0xFF;
}
if(COMMAND.OPCODE == 0) {
RecieveType = 2;
}
else {
RecieveType = 1;
}
COMMAND.CheckSUM = 0xE00; //E is the constant and the rest will be
calculated after been through Build Command
//if ack is wanted
//RecieveType = 1;
//if poll is wanted
//RecieveType = 2;
cout<<endl<<"command built"<<endl;
ToSend = BuildCommand(&COMMAND);
/*
for(int l = 0; l < 10; l++) {
printf("%.02x ",(unsigned char)ToSend[l]);
}
printf("");
*/

/*
unsigned char byte = 0;
for(int i=1;i<9;i++) {
    byte = byte ^ (unsigned char)ToSend[i];
    //cout<<"byte : "<<byte<<" | (unsigned char)ToSend[i]) : "<< (
(unsigned char)ToSend[i] ) <<endl;
}

(unsigned char)ToSend[9] = byte;
*/
//cout<<"Command with checksum"<<endl;
/*
for(int l = 0; l < 10; l++) {
printf("%.02x ",(unsigned char)ToSend[l]);
}
printf("");
*/
```

```
if(itsSerial_IO->WRITE(ToSend)) {
    cout<<"yahoo"<<endl;
    itsResponse->GEN(evWaitforResponse(atts));
}
else {
    cout<<"shibby^2: didnt write command"<<endl;
    //re-create todo


    GEN(evResponseNeeded);
}
```

Out Transition

```
evResponseNeeded
```

Target:

getToDos

Out Transition

```
evReDo
```

Target:

waitForReSend

wait

temp state used to wait until there is something to do

Or-state

Out Transition

```
evDoToDo
```

Target:

secondwait

waitForReSend

wait to see if there needs to be a re-sending of the command

Or-state

EntryAction

```
atts = params->tempAtts;
```

Out Transition

```
tm(1000)
```

Target:

sendCommand

---

database

function used to interact with the database

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsCheckES

Association with checkES, Multiplicity of 1, Bi-directional

itsPoll

Association with Poll, Multiplicity of 1, Bi-directional

itsResponse

Association with response, Multiplicity of 1, Bi-directional

itsImmediate

Association with Immediate, Multiplicity of 1, Bi-directional

itsSystemSetup

Association with systemSetup, Multiplicity of 1, Bi-directional

itsSystemChecks

Association with systemChecks, Multiplicity of 1, Bi-directional

Operations:

__setItsCheckES

Generated , Primitive-operation , Public, Return type is void

Args:

'checkES*' p_checkES

Body

```
itsCheckES = p_checkES;
```

__setItsImmediate

Generated , Primitive-operation , Public, Return type is void

Args:

'Immediate*' p_Immediate

Body

```
itsImmediate = p_Immediate;
```

__setItsPoll

Generated , Primitive-operation , Public, Return type is void

Args:

'Poll*' p_Poll

Body

```
itsPoll = p_Poll;
```

__setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
itsResponse = p_response;
```

__setItsSystemChecks

Generated , Primitive-operation , Public, Return type is void

Args:

'systemChecks*' p_systemChecks

Body

```
itsSystemChecks = p_systemChecks;
```

__setItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Args:

'systemSetup*' p_systemSetup

Body

```
itsSystemSetup = p_systemSetup;
```

_clearItsCheckES

Generated , Primitive-operation , Public, Return type is void

Body

```
itsCheckES = NULL;
```

_clearItsImmediate

Generated , Primitive-operation , Public, Return type is void

Body

```
itsImmediate = NULL;
```

_clearItsPoll

Generated , Primitive-operation , Public, Return type is void

Body

```
itsPoll = NULL;
```

_clearItsResponse

Generated , Primitive-operation , Public, Return type is void

Body
```
itsResponse = NULL;
```

_clearItsSystemChecks

Generated , Primitive-operation , Public, Return type is void

Body
```
itsSystemChecks = NULL;
```

_clearItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Body
```
itsSystemSetup = NULL;
```

_setItsCheckES

Generated , Primitive-operation , Public, Return type is void

Args:

'checkES*' p_checkES

Body
```
if(itsCheckES != NULL)
    itsCheckES->__setItsDatabase(NULL);
__setItsCheckES(p_checkES);
```

_setItsImmediate

Generated , Primitive-operation , Public, Return type is void

Args:

'Immediate*' p_Immediate

Body
```
if(itsImmediate != NULL)
    itsImmediate->__setItsDatabase(NULL);
__setItsImmediate(p_Immediate);
```

_setItsPoll

Generated , Primitive-operation , Public, Return type is void

Args:

'Poll*' p_Poll

Body
```
if(itsPoll != NULL)
    itsPoll->__setItsDatabase(NULL);
__setItsPoll(p_Poll);
```

_setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body
```
if(itsResponse != NULL)
    itsResponse->__setItsDatabase(NULL);
__setItsResponse(p_response);
```

_setItsSystemChecks

Generated , Primitive-operation , Public, Return type is void

Args:

'systemChecks*' p_systemChecks

Body
```
if(itsSystemChecks != NULL)
    itsSystemChecks->__setItsDatabase(NULL);
__setItsSystemChecks(p_systemChecks);
```

_setItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Args:

'systemSetup*' p_systemSetup

Body

```
if(itsSystemSetup != NULL)
    itsSystemSetup->__setItsDatabase(NULL);
__setItsSystemSetup(p_systemSetup);
```

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsCheckES != NULL)
    {

        database* p_database = itsCheckES->getItsDatabase();
        if(p_database != NULL)
            itsCheckES->__setItsDatabase(NULL);
        itsCheckES = NULL;
    }
if(itsImmediate != NULL)
    {

        database* p_database = itsImmediate->getItsDatabase();
        if(p_database != NULL)
            itsImmediate->__setItsDatabase(NULL);
        itsImmediate = NULL;
    }
if(itsPoll != NULL)
    {

        database* p_database = itsPoll->getItsDatabase();
        if(p_database != NULL)
            itsPoll->__setItsDatabase(NULL);
        itsPoll = NULL;
    }
if(itsResponse != NULL)
    {

        database* p_database = itsResponse->getItsDatabase();
        if(p_database != NULL)
            itsResponse->__setItsDatabase(NULL);
        itsResponse = NULL;
    }
if(itsSystemChecks != NULL)
    {

        database* p_database = itsSystemChecks->getItsDatabase();
        if(p_database != NULL)
            itsSystemChecks->__setItsDatabase(NULL);
        itsSystemChecks = NULL;
    }
if(itsSystemSetup != NULL)
    {

        database* p_database = itsSystemSetup->getItsDatabase();
        if(p_database != NULL)
            itsSystemSetup->__setItsDatabase(NULL);
        itsSystemSetup = NULL;
    }
```

database

setup up the database connection

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Constructor , Public

Body

```
//cout<<"opening database connection"<<endl;
char * dsnSTR = "DSN=CB";
try {
DB.OpenEx(dsnSTR,0);
 //cout<<"connected sort of"<<endl;
 //send commands to start other classes Database is now operating
        OKtoStart = true;
}
catch(CDBException) {
 //cout<<"attempt didnt work"<<endl;
 //put error message up,
 //and give option to select DSN
 try{
 DB.OpenEx("",CDatabase::forceOdbcDialog);
        //sorry for inconvience
        OKtoStart = true;
 }
 catch(CDBException){
        //cout<<"database not opened"<<endl;
        //big error message and shut down program
 }
}
```

evSetup

Event

execute

function used to execute a sql statement with no desire for getting information back

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is void

Args:

'CString %s' Estring

Body

```
DB.ExecuteSQL(Estring);
```

getDB

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is 'CDatabase *'

Body

```
return &DB;
```

getItsCheckES

Generated , Primitive-operation , Public, Return type is 'checkES*'

Constant

Body

```
return itsCheckES;
```

getItsImmediate

Generated , Primitive-operation , Public, Return type is 'Immediate*'
Constant
Body

```
return itsImmediate;
```

getItsPoll

Generated , Primitive-operation , Public, Return type is 'Poll*'
Constant
Body

```
return itsPoll;
```

getItsResponse

Generated , Primitive-operation , Public, Return type is 'response*'
Constant
Body

```
return itsResponse;
```

getItsSystemChecks

Generated , Primitive-operation , Public, Return type is 'systemChecks*'
Constant
Body

```
return itsSystemChecks;
```

getItsSystemSetup

Generated , Primitive-operation , Public, Return type is 'systemSetup*'
Constant
Body

```
return itsSystemSetup;
```

getOKtoStart

Generated , Primitive-operation , Public, Return type is OMBoolean
Constant
Body

```
return OKtoStart;
```

isConnected

function to let other classes know if the databasse is connected

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common
Inline: none

Primitive-operation , Public, Return type is OMBoolean
Body

```
return (DB.IsOpen() && OKtoStart );
```

query

function to query the database using a recordset to view requested data

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common
Inline: none

Primitive-operation , Public, Return type is void

Args:

'CRecordset * %s' rSet

'CString %s' SQLStatement

Body

```
rSet->Open(CRecordset::forwardOnly, SQLStatement);
```

setItsCheckES

Generated , Primitive-operation , Public, Return type is void

Args:

'checkES*' p_checkES

Body

```
if(p_checkES != NULL)
    p_checkES->_setItsDatabase(this);
_setItsCheckES(p_checkES);
```

setItsImmediate

Generated , Primitive-operation , Public, Return type is void

Args:

'Immediate*' p_Immediate

Body

```
if(p_Immediate != NULL)
    p_Immediate->_setItsDatabase(this);
_setItsImmediate(p_Immediate);
```

setItsPoll

Generated , Primitive-operation , Public, Return type is void

Args:

'Poll*' p_Poll

Body

```
if(p_Poll != NULL)
    p_Poll->_setItsDatabase(this);
_setItsPoll(p_Poll);
```

setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
if(p_response != NULL)
    p_response->_setItsDatabase(this);
_setItsResponse(p_response);
```

setItsSystemChecks

Generated , Primitive-operation , Public, Return type is void

Args:

'systemChecks*' p_systemChecks

Body

```
if(p_systemChecks != NULL)
    p_systemChecks->_setItsDatabase(this);
_setItsSystemChecks(p_systemChecks);
```

setItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Args:

'systemSetup*' p_systemSetup

Body

```
if(p_systemSetup != NULL)
    p_systemSetup->_setItsDatabase(this);
_setItsSystemSetup(p_systemSetup);
```

setOKtoStart

Generated , Primitive-operation , Public, Return type is OMBoolean

Args:

OMBoolean p_OKtoStart

Body

```
OKtoStart = p_OKtoStart;
```

startBehavior

Virtual, Generated , Primitive-operation , Public, Return type is OMBoolean

Body

```
OMBoolean done = FALSE;
done = OMReactive::startBehavior();
if(done)
    start();
return done;
```

~database

Generated , Destructor , Public

Body

```
cleanUpRelations();
```

Attributes:

DB

an object that uses microsoft ODBC drivers to handle the database

Overridden Properties

Subjects:

CG

Metaclasses:

Attribute

Properties:

IsConst: False

CPP_CG

Metaclasses:

Attribute

Properties:

MutatorGenerate: False

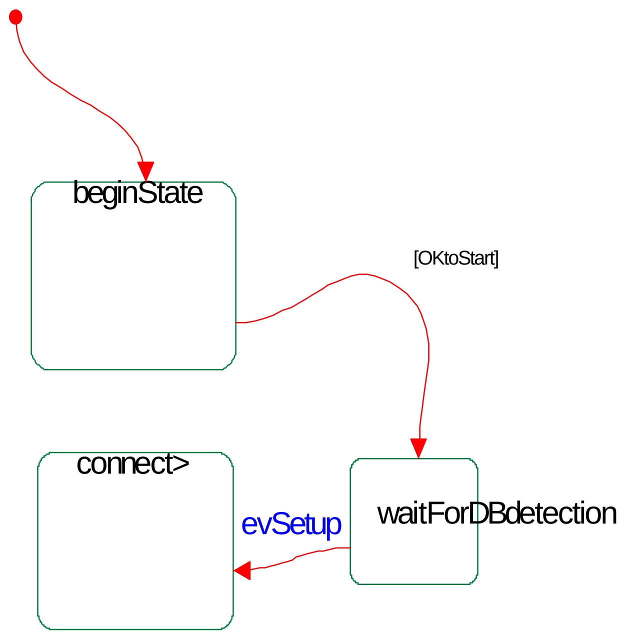
Type of 'CDatabase %s;', Public

OKtoStart

boolean to let the class know if it is okay to start system functions

Type of OMBoolean, Public, Initial Value: false

Statechart



ROOT

Or-state

Substates:

beginState
connect
waitForDBdetection

Default Transition

Target:

beginState

beginState

temp state to check to see if it is okay to start connection

Or-state

Out Transition

```
[OKtoStart]
```

Target:

waitForDBdetection

connect

send out start commands to active classes

Or-state

EntryAction

```
//cout<<"okay starting serviceis"<<endl;
itsCheckES->GEN(evConnected1);
itsPoll->GEN(evConnected2);
itsImmediate->GEN(evConnected3);
//cout<<"servicies started"<<endl;
```

waitForDBdetection

temp state waiting for assured connection to database

Or-state

Out Transition

```
evSetup
```

Target:

connect

Immediate

this class checks the database every second to see if the user has created and immediate action commands from the web site

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsTODO_Handler

Association with TODO_Handler, Multiplicity of 1, Bi-directional

itsDatabase

Association with database, Multiplicity of 1, Bi-directional

itsSystemSetup

Association with systemSetup, Multiplicity of 1, Bi-directional

Operations:

__setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
itsDatabase = p_database;
```

__setItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Args:

'systemSetup*' p_systemSetup

Body

```
itsSystemSetup = p_systemSetup;
```

__setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
itsTODO_Handler = p_TODO_Handler;
```

_clearItsDatabase

Generated , Primitive-operation , Public, Return type is void

Body

```
itsDatabase = NULL;
```

_clearItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Body

```
itsSystemSetup = NULL;
```

_clearItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Body

```
itsTODO_Handler = NULL;
```

_setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(itsDatabase != NULL)
    itsDatabase->__setItsImmediate(NULL);
__setItsDatabase(p_database);
```

_setItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Args:

'systemSetup*' p_systemSetup

Body

```
if(itsSystemSetup != NULL)
    itsSystemSetup->__setItsImmediate(NULL);
__setItsSystemSetup(p_systemSetup);
```

_setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(itsTODO_Handler != NULL)
    itsTODO_Handler->__setItsImmediate(NULL);
__setItsTODO_Handler(p_TODO_Handler);
```

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsDatabase != NULL)
    {

        Immediate* p_Immediate = itsDatabase->getItsImmediate();
        if(p_Immediate != NULL)
            itsDatabase->__setItsImmediate(NULL);
        itsDatabase = NULL;
    }
if(itsSystemSetup != NULL)
    {

        Immediate* p_Immediate = itsSystemSetup->getItsImmediate();
        if(p_Immediate != NULL)
            itsSystemSetup->__setItsImmediate(NULL);
        itsSystemSetup = NULL;
    }
if(itsTODO_Handler != NULL)
    {

        Immediate* p_Immediate = itsTODO_Handler->getItsImmediate();
        if(p_Immediate != NULL)
            itsTODO_Handler->__setItsImmediate(NULL);
        itsTODO_Handler = NULL;
    }
```

evConnected

Event

evConnected3

Event

getItsDatabase

Generated , Primitive-operation , Public, Return type is 'database*'

Constant

Body

```
return itsDatabase;
```

getItsSystemSetup

Generated , Primitive-operation , Public, Return type is 'systemSetup*'

Constant

Body

```
return itsSystemSetup;
```

getItsTODO_Handler

Generated , Primitive-operation , Public, Return type is 'TODO_Handler*'
Constant

Body

```
return itsTODO_Handler;
```

Immediate

Generated , Constructor , Public

setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(p_database != NULL)
    p_database->_setItsImmediate(this);
_setItsDatabase(p_database);
```

setItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Args:

'systemSetup*' p_systemSetup

Body

```
if(p_systemSetup != NULL)
    p_systemSetup->_setItsImmediate(this);
_setItsSystemSetup(p_systemSetup);
```

setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(p_TODO_Handler != NULL)
    p_TODO_Handler->_setItsImmediate(this);
_setItsTODO_Handler(p_TODO_Handler);
```

startBehavior

Virtual, Generated , Primitive-operation , Public, Return type is OMBoolean

Body

```
OMBoolean done = FALSE;
done = OMReactive::startBehavior();
if(done)
    start();
return done;
```

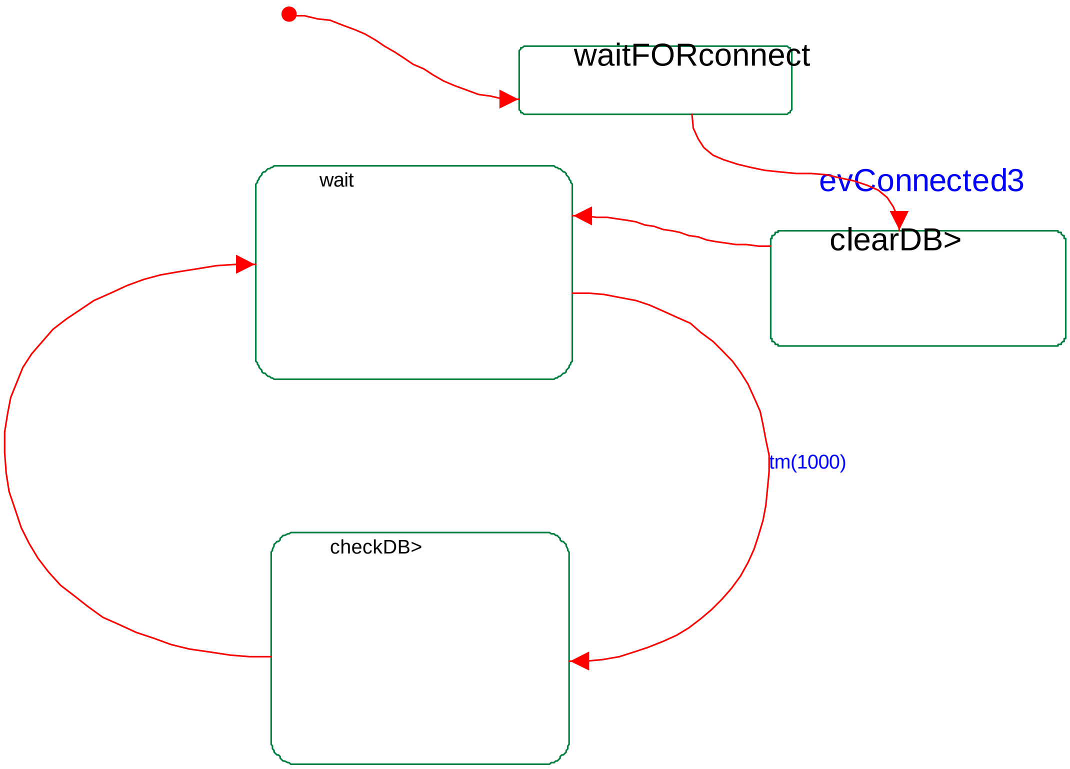
~Immediate

Generated , Destructor , Public

Body

```
cleanUpRelations();
```

Statechart



ROOT

Or-state

Substates:

- checkDB
- clearDB
- wait
- waitFORconnect

Default Transition

Target:

waitFORconnect

checkDB

check the database, if there is a immediate action to complete then create a todo for it otherwise go to wait state

Or-state

EntryAction

```
CString errorStr;
char tempStr[250];
char timebuf[20];
CString temp;
CDBVariant OPCODE, offset;
CRecordset rs( itsDatabase->getDB());
CRecordset rs1( itsDatabase->getDB());
//cout<<"here in immediate"<<endl;
if( itsDatabase->isConnected()) {
    itsDatabase->query(&rs, "SELECT ID, OPCODE, offset FROM
IA_effects_WU");
    CDBVariant val1, val2, val3, val4, hi_low;
    while (!rs.IsEOF()) {
            cout<<"first while"<<endl;
            rs.GetFieldValue(short(1), OPCODE);
            rs.GetFieldValue(short(2), offset);
            if(OPCODE.m_chVal <= 10) {
```

```
                    itsDatabase->query(&rs1, ("SELECT Wall_Unit.PW,
Wall_Unit.address, OPCODE, offset, HI_LOW from Wall_Unit, IA_effects_WU
where WUID = Wall_Unit.ID") );
                    while(!rs1.IsEOF()) {
                    cout<<"second while"<<endl;
                            rs1.GetFieldValue(short(0), val1);
                            rs1.GetFieldValue(short(1), val2);
                            rs1.GetFieldValue(short(2), val3);
                            rs1.GetFieldValue(short(3), val4);
                            rs1.GetFieldValue(short(4), hi_low);
                            if( (val1.m_dwType == DBVT_UCHAR) &&
(val2.m_dwType == DBVT_LONG) && (val3.m_dwType == DBVT_UCHAR) &&
(val4.m_dwType == DBVT_SHORT) && (hi_low.m_dwType == DBVT_UCHAR) ) {
                                    cout<<"--------------------
offset:opcode"<<val4.m_iVal<<":"<<val3.m_chVal<<endl;
                                    itsTODO_Handler-
>NewTODO(val2.m_lVal,val1.m_chVal,val3.m_chVal,0,val4.m_iVal,hi_low.m_c
hVal,5);

                            }
                            else {
                                    //error types dont match
                                    temp = "ERROR";
                                    errorStr = "INSERT into ErrorLog
values(3,'Types of PW and address and OPCODE are not correct from
Immediate' ,'";

    strftime(timebuf,20,"%c",CTime::GetCurrentTime().GetLocalTm());
                                    temp = timebuf;
                                    errorStr = errorStr + CString(temp);
                                    errorStr += "' ,NULL";
                                    cout<<errorStr<<endl;
                                    itsDatabase->execute(errorStr);
                            }
                            rs1.MoveNext();
                    }
                    rs1.Close();
            }else {
                    if(OPCODE.m_chVal == 100) {
                            itsSystemSetup->GEN(evFIND);
                    }
                    if(OPCODE.m_chVal == 200) {
                            //do immediate event sequence
                            cout<<"opcode of 200"<<endl;
                            (void)sprintf(tempStr,"SELECT OPCODE,
offset, HI_LOW, Wall_Unit.address FROM ES_effects_WU, Wall_Unit WHERE
Wall_Unit.ID = WUID AND ESID = %d",offset.m_iVal);
                            itsDatabase->query(&rs1,tempStr);
                            while(!rs1.IsEOF()) {
                                    rs1.GetFieldValue(short(0), val1);
                                    rs1.GetFieldValue(short(1), val2);
                                    rs1.GetFieldValue(short(2), val3);
                                    rs1.GetFieldValue(short(3), val4);
                                    cout<<"i keep making new
todos"<<endl;
                                    itsTODO_Handler-
>NewTODO(val4.m_lVal,(unsigned char)
1,val1.m_chVal,0,val2.m_iVal,val3.m_chVal,5);
                                    rs1.MoveNext();
                            }
                            rs1.Close();
                    }
            }
            rs.MoveNext();
    }
    itsDatabase->execute("Delete * from IA_effects_WU");
    rs.Close();
}
```

Out Transition

Target:
wait

clearDB

we need to clear all immediate actions that were created with the system not running because they are errors

Or-state

EntryAction

```
if(itsDatabase->isConnected()) {
    itsDatabase->execute("DELETE * FROM IA_effects_WU");
}
```

Out Transition

Target:
wait

wait

temp state to wait until the next second

Or-state

Out Transition

```
tm(1000)
```

Target:
checkDB

waitFORconnect

temp state to wait for the confirmation if can start

Or-state

Out Transition

```
evConnected3
```

Target:
clearDB

Poll

class that operates at every 30 second mark anc creates a polling command for each wall unit in the database. It works on a three minute cycle to request the totals for voltage, current high, and current low. WE do this to lower teh ammount of information being sent accross the wireless connection every minute

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsTODO_Handler

Association with TODO_Handler, Multiplicity of 1, Bi-directional

itsDatabase

Association with database, Multiplicity of 1, Bi-directional

Operations:

__setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
itsDatabase = p_database;
```

__setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
itsTODO_Handler = p_TODO_Handler;
```

_clearItsDatabase

Generated , Primitive-operation , Public, Return type is void

Body

```
itsDatabase = NULL;
```

_clearItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Body

```
itsTODO_Handler = NULL;
```

_setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(itsDatabase != NULL)
    itsDatabase->__setItsPoll(NULL);
__setItsDatabase(p_database);
```

_setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(itsTODO_Handler != NULL)
    itsTODO_Handler->__setItsPoll(NULL);
__setItsTODO_Handler(p_TODO_Handler);
```

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsDatabase != NULL)
    {

        Poll* p_Poll = itsDatabase->getItsPoll();
        if(p_Poll != NULL)
            itsDatabase->__setItsPoll(NULL);
        itsDatabase = NULL;
    }
if(itsTODO_Handler != NULL)
    {

        Poll* p_Poll = itsTODO_Handler->getItsPoll();
        if(p_Poll != NULL)
            itsTODO_Handler->__setItsPoll(NULL);
        itsTODO_Handler = NULL;
    }
```

evConnected

Event

evConnected2

Event

getItsDatabase

Generated , Primitive-operation , Public, Return type is 'database*'

Constant

Body

```
return itsDatabase;
```

getItsTODO_Handler

Generated , Primitive-operation , Public, Return type is 'TODO_Handler*'

Constant

Body

```
return itsTODO_Handler;
```

getWhatToPoll

Generated , Primitive-operation , Public, Return type is int

Constant

Body

```
return whatToPoll;
```

Poll

Generated , Constructor , Public

setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(p_database != NULL)
    p_database->_setItsPoll(this);
_setItsDatabase(p_database);
```

setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(p_TODO_Handler != NULL)
    p_TODO_Handler->_setItsPoll(this);
_setItsTODO_Handler(p_TODO_Handler);
```

setWhatToPoll

Generated , Primitive-operation , Public, Return type is int

Args:

int p_whatToPoll

Body

```
whatToPoll = p_whatToPoll;
```

startBehavior

Virtual, Generated , Primitive-operation , Public, Return type is OMBoolean

Body

```
OMBoolean done = FALSE;
done = OMReactive::startBehavior();
if(done)
    start();
return done;
```

TimeSeconds

function returning the number of seconds until the next 30 second mark

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is int

Body

```
if( CTime::GetCurrentTime().GetSecond() < 30 ) {
 return (30 - CTime::GetCurrentTime().GetSecond());
}
else {
```

```
    return (30 + (60 - CTime::GetCurrentTime().GetSecond()));
}
```

~Poll

Generated , Destructor , Public

Body

```
cleanUpRelations();
```

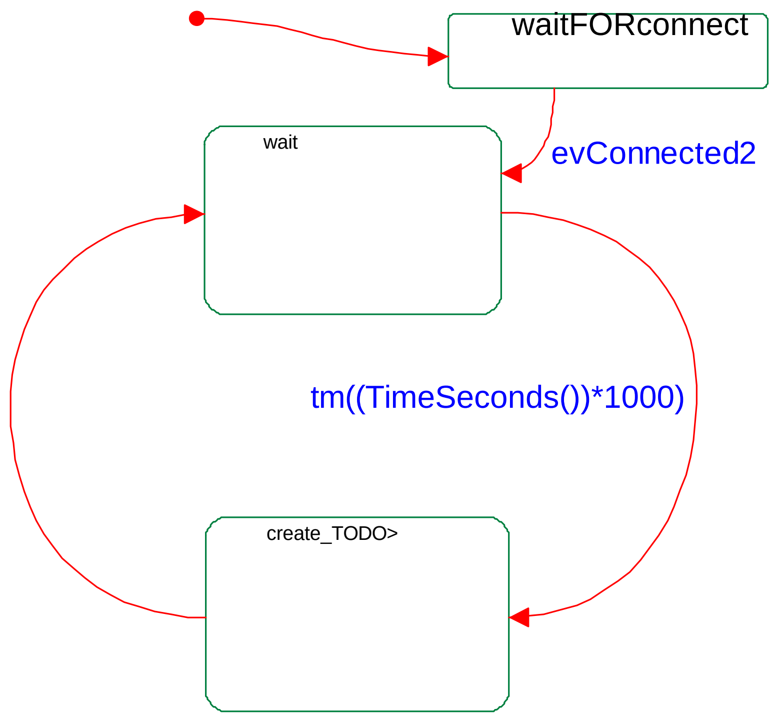
Attributes:

whatToPoll

keeps track of what needs to be polled this minute

Type of int, Public

Statechart



ROOT

Or-state

Substates:

create_TODO

wait

waitFORconnect

Default Transition

Target:

waitFORconnect

create_TODO

create a poll command for each wall unit in the database

Or-state

EntryAction

```
CString errorStr;
char timebuf[20];
char exeStr[200];
unsigned char opcode = 0x00;
CString temp;
if(itsDatabase->isConnected()) {
    CRecordset rs( itsDatabase->getDB());
    itsDatabase->query(&rs,"SELECT PW, address FROM Wall_Unit");
```

```
    CDBVariant varValue;
    CDBVariant PW, Address;
    while (!rs.IsEOF()) {
            rs.GetFieldValue(short(0), PW);
            rs.GetFieldValue(short(1), Address);
            if( (PW.m_dwType == DBVT_UCHAR) && (Address.m_dwType ==
DBVT_LONG) ) {
                    if (whatToPoll == 0) {
                            opcode = 0x00;
                    }else if(whatToPoll == 1) {
                            opcode = 0x05;
                    }else if(whatToPoll == 2) {
                            opcode = 0x06;
                    }
                    cout<<"poll what to poll: "<<whatToPoll<<endl;
                    //itsTODO_Handler->NewTODO(Address.m_lVal,
PW.m_chVal,opcode,0,0,0,whatToPoll);
                    whatToPoll = (++whatToPoll)%3;
            }
            else {
                    //error types dont match report it
                    /*
                    temp = "ERROR";
                    errorStr = "INSERT into ErrorLog values(3,'Types of
PW amd Address are not correct from Poll' ,'";
                    temp = timebuf;
                    errorStr = errorStr + CString(temp);
                    errorStr += "' ,NULL";
                    cout<<errorStr<<endl;
                    */

    strftime(timebuf,20,"%c",CTime::GetCurrentTime().GetLocalTm());
                    (void)sprintf(exeStr,"INSERT into ErrorLog
values(3,'Types of PW amd Address are not correct from Poll'
,'%s',NULL)",timebuf);
                    itsDatabase->execute(exeStr);
            }
            rs.MoveNext();
    }
rs.Close();
}
```

Out Transition

Target:

wait

wait

temp state to wait for the next 30 second mark

Or-state

Out Transition

```
tm((TimeSeconds())*1000)
```

Target:

create_TODO

waitFORconnect

temp state to wait for database connectivity

Or-state

Out Transition

```
evConnected2
```

Target:

wait

---

response

class that waits for a response from the wall units, if none occurs then the command needs to be sent out once again

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsTODO_Handler

Association with TODO_Handler, Multiplicity of 1, Bi-directional

itsCheckTODO

Association with checkTODO, Multiplicity of 1, Bi-directional

itsSerial_IO

Association with Serial_IO, Multiplicity of 1, Bi-directional

itsDatabase

Association with database, Multiplicity of 1, Bi-directional

Operations:

__setItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Args:

'checkTODO*' p_checkTODO

Body

```
itsCheckTODO = p_checkTODO;
```

__setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
itsDatabase = p_database;
```

__setItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Args:

'Serial_IO*' p_Serial_IO

Body

```
itsSerial_IO = p_Serial_IO;
```

__setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
itsTODO_Handler = p_TODO_Handler;
```

_clearItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Body

```
itsCheckTODO = NULL;
```

_clearItsDatabase

Generated , Primitive-operation , Public, Return type is void

Body

```
itsDatabase = NULL;
```

_clearItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Body

```
itsSerial_IO = NULL;
```

_clearItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Body

```
itsTODO_Handler = NULL;
```

_setItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Args:

'checkTODO*' p_checkTODO

Body

```
if(itsCheckTODO != NULL)
    itsCheckTODO->__setItsResponse(NULL);
__setItsCheckTODO(p_checkTODO);
```

_setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(itsDatabase != NULL)
    itsDatabase->__setItsResponse(NULL);
__setItsDatabase(p_database);
```

_setItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Args:

'Serial_IO*' p_Serial_IO

Body

```
if(itsSerial_IO != NULL)
    itsSerial_IO->__setItsResponse(NULL);
__setItsSerial_IO(p_Serial_IO);
```

_setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(itsTODO_Handler != NULL)
    itsTODO_Handler->__setItsResponse(NULL);
__setItsTODO_Handler(p_TODO_Handler);
```

BuildCommand

build the command to send to wall units

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is 'unsigned char *'

Args:

'COMMAND_STRUCTURE * %s' COMMAND

Body

```
#define byteswap32(a)
((a<<24)&0xff000000)|((a<<8)&0x00ff0000)|((a>>8)&0x0000ff00)|((a>>24
)&0x000000ff)
#define byteswap16(a) ((a>>8)&0x00ff)|((a<<8)&0xff00)

//char tmpStr[10];
//tmpStr[0] = 'E';
tmpStr[0] = (COMMAND->StartSymbol & 0xFF);
*((long*)&tmpStr[1]) = ((long)byteswap32(COMMAND->To));
*((int*)&tmpStr[5]) = ((int)byteswap16(COMMAND->From));
```

```
tmpStr[7] = (char)((COMMAND->OPCODE << 5) & 0xE0);
tmpStr[7] |= (char)((COMMAND->HI_LOW << 4) & 0x10);
tmpStr[7] |= (char)((COMMAND->OFFSet >> 4) & 0x0F);
tmpStr[8] = (char)((COMMAND->OFFSet << 4) & 0xF0);
tmpStr[8] |= (char)((COMMAND->CheckSUM >> 8) & 0x0F);
unsigned char byte = 0;
for(int i=1;i<9;i++) {
 byte = byte ^ (unsigned char)tmpStr[i];
 //cout<<"byte : "<<byte<<" | (unsigned char)ToSend[i]) : "<< (
(unsigned char)ToSend[i] ) <<endl;
}

(unsigned char)tmpStr[9] = byte;

printf("command: ");
for(i = 0; i < 10; i++) {
printf("%.02x ",tmpStr[i]);
}
printf("");
return tmpStr;
```

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsCheckTODO != NULL)
    {

        response* p_response = itsCheckTODO->getItsResponse();
        if(p_response != NULL)
            itsCheckTODO->__setItsResponse(NULL);
        itsCheckTODO = NULL;
    }
if(itsDatabase != NULL)
    {

        response* p_response = itsDatabase->getItsResponse();
        if(p_response != NULL)
            itsDatabase->__setItsResponse(NULL);
        itsDatabase = NULL;
    }
if(itsSerial_IO != NULL)
    {

        response* p_response = itsSerial_IO->getItsResponse();
        if(p_response != NULL)
            itsSerial_IO->__setItsResponse(NULL);
        itsSerial_IO = NULL;
    }
if(itsTODO_Handler != NULL)
    {

        response* p_response = itsTODO_Handler->getItsResponse();
        if(p_response != NULL)
            itsTODO_Handler->__setItsResponse(NULL);
        itsTODO_Handler = NULL;
    }
```

decifer

function used to decifier a response that has come in

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is int

Args:

'unsigned char * %s' Command
int whatToPoll

Body

```
#define byteswap32(a)
((a<<24)&0xff000000)|((a<<8)&0x00ff0000)|((a>>8)&0x0000ff00)|((a>>24
)&0x000000ff)
#define byteswap16(a) ((a>>8)&0x00ff)|((a<<8)&0xff00)
#define byteswap8(a)  ((a>>4)&0xf)|((a<<4)&0xf0)

unsigned long Voltage;
double DBVolt, DBCurrent;
unsigned char checksum=0;
int chk;
unsigned int numSamples1;
//cout<<"trying int"<<endl;

cout<<"whatPoll: "<<whatToPoll<<endl;

for(int i =0 ; i < 16; i++) {
    printf(" %.02x ",Command[i]);
}

cout<<endl;
if( Command[0] == 0xDB) {    //voltage
 //cout<<"good start symbol"<<endl;
 //printf("right is Nick : %.04x",*((unsigned int*)&(Command[1])));
 for(chk=0;chk<13;chk++) {
        checksum = checksum ^ Command[chk];
 }
 //if(byteswap8(Command[15]) == checksum) {
        if( whatToPoll == 0) {
               if( byteswap16( *((unsigned int*)&(Command[1])) ) ==
0x0001) {
                       cout<<"good addressV"<<endl;
                       // if address is good
                              Totals.address = byteswap32(
*((unsigned long*)&(Command[3])) );
                              //cout<<"addr recieved :
"<<Totals.address<<endl;
                              Voltage = byteswap32( *((unsigned
long*)&(Command[7])) );
                              cout<<"voltage: "<<Voltage<<endl;
                              //cout<< "pre swap samples :
"<<*((unsigned int*)&(Command[11]))<<endl;
                              numSamples1 = byteswap16( *((unsigned
int*)&(Command[11])) );
                              cout<<"samplesV :
"<<numSamples1<<endl;
                              if (numSamples1 != 0) {
                                     DBVolt =
(double)((double)Voltage / (double)numSamples1+1);
                                     cout<<"V2: "<<DBVolt<<endl;
                                     DBVolt = (double)(DBVolt /
(double)(.00419947506562));
                                     cout<<"V1: "<<DBVolt<<endl;
                                     DBVolt = (double) (DBVolt *
(double)( (double)(.0083333333) / (double)(85.3) ) );
                                     cout<<"V is :
"<<DBVolt<<endl;
                                     Totals.v =
(double)(DBVolt*79);
                              }
                              else
                                     Totals.v = (double)(120.6);
                              Totals.c1= 0.0;
                              Totals.c2 = 0.0;
                              //if checksum is good
                              return 0;
                              //else
```

```
                              //return -1;
                      }
              }else if(whatToPoll == 1) {   //current1
                      if( byteswap16( *((unsigned int*)&(Command[1])) ) ==
0x0001) {
                              //cout<<"good address"<<endl;
                              // if address is good
                                      Totals.address = byteswap32(
*((unsigned long*)&(Command[3])) );
                                      //cout<<"addr recieved :
"<<Totals.address<<endl;
                                      Voltage = byteswap32( *((unsigned
long*)&(Command[7])) );
                                      cout<<"voltage: "<<Voltage<<endl;
                                      //cout<< "pre swap samples :
"<<*((unsigned int*)&(Command[11]))<<endl;
                                      numSamples1 = byteswap16( *((unsigned
int*)&(Command[11])) );
                                      cout<<"samplesC :
"<<numSamples1<<endl;
                                      if (numSamples1 != 0) {
                                              DBCurrent =
(double)((double)Voltage / (double)numSamples1+1);
                                              cout<<"C2:
"<<DBCurrent<<endl;
                                              DBCurrent = (double)
(DBCurrent * (double)( (double)(.0083333333) / (double)(85.3) ) );
                                              cout<<"C is :
"<<DBCurrent<<endl;
                                              DBCurrent =
(double)(DBCurrent / (double)(.00419947506562));
                                              cout<<"C1:
"<<DBCurrent<<endl;
                                              Totals.c1 =
((double)((DBCurrent/30)/.005));
                                      }
                                      else
                                              Totals.c1 = 0.0;
                                      Totals.v= 0.0;
                                      Totals.c2 = 0.0;
                                      //if checksum is good
                                      return 0;
                                      //else
                                      //return -1;
                      }
              }else if(whatToPoll == 2) {  //current 2
                      if( byteswap16( *((unsigned int*)&(Command[1])) ) ==
0x0001) {
                              //cout<<"good address"<<endl;
                              // if address is good
                                      Totals.address = byteswap32(
*((unsigned long*)&(Command[3])) );
                                      //cout<<"addr recieved :
"<<Totals.address<<endl;
                                      Voltage = byteswap32( *((unsigned
long*)&(Command[7])) );
                                      cout<<"voltage: "<<Voltage<<endl;
                                      //cout<< "pre swap samples :
"<<*((unsigned int*)&(Command[11]))<<endl;
                                      numSamples1 = byteswap16( *((unsigned
int*)&(Command[11])) );
                                      cout<<"samplesC2 :
"<<numSamples1<<endl;
                                      if (numSamples1 != 0) {
                                              DBCurrent =
(double)((double)Voltage / (double)numSamples1+1);
                                              cout<<"V2:
"<<DBCurrent<<endl;
                                              DBCurrent =
(double)(DBCurrent / (double)(.00419947506562));
                                              cout<<"V1:
```

```
"<<DBCurrent<<endl;
                                        DBCurrent = (double)
(DBCurrent * (double)( (double)(.0083333333) / (double)(85.3) ) );
                                        cout<<"V is :
"<<DBCurrent<<endl;
                                        Totals.c2 =
((double)((DBCurrent/30)/.005));
                                }
                                else
                                        Totals.c2 = 0.0;
                                Totals.c1= 0.0;
                                Totals.v = 0.0;
                                //if checksum is good
                                return 0;
                                //else
                                //return -1;
                }
        }
 //}
 //else {
        //cout<<"Command[15]: "<<Command[15]<<endl;
        //cout<<"checksum   : "<<checksum<<endl;
        printf("byteswap8(Command[15]): %.02x
",byteswap8(Command[15]));
        printf("checksum   : %.02x ",checksum);
        cout<<"bad checksum"<<endl;
 //}
 return -1;
}
else if( (unsigned)Command[0] == 0xBD ) {
 //ack
 cout<<"got ack"<<endl;
 if( byteswap16( *((unsigned int*)&(Command[1])) ) == 0x0001) {
        cout<<"good address"<<endl;
        // if address is good
                Totals.address = byteswap32( *((unsigned
long*)&(Command[3])) );
                cout<<"addr recieved : "<<Totals.address<<endl;
                Totals.v = 0.0;
                Totals.c1 = 0.0;
                Totals.c2 = 0.0;
                return 1;
 }
 else {
                Totals.v = 0.0;
                Totals.c1 = 0.0;
                Totals.c2 = 0.0;
                return -1;
 }
}
else {
 Totals.v = 0.0;
 Totals.c1 = 0.0;
 Totals.c2 = 0.0;
 return -1;
}
```

evWaitforResponse

Event

getCOMMAND

Generated , Primitive-operation , Public, Return type is COMMAND_STRUCTURE

Constant

<u>Body</u>

```
return COMMAND;
```

getItsCheckTODO

Generated , Primitive-operation , Public, Return type is 'checkTODO*'

Constant

Body

```
return itsCheckTODO;
```

getItsDatabase

Generated , Primitive-operation , Public, Return type is 'database*'

Constant

Body

```
return itsDatabase;
```

getItsSerial_IO

Generated , Primitive-operation , Public, Return type is 'Serial_IO*'

Constant

Body

```
return itsSerial_IO;
```

getItsTODO_Handler

Generated , Primitive-operation , Public, Return type is 'TODO_Handler*'

Constant

Body

```
return itsTODO_Handler;
```

getTempAtts

Generated , Primitive-operation , Public, Return type is todoattributes

Constant

Body

```
return tempAtts;
```

getTmpStr

Generated , Primitive-operation , Public, Return type is 'unsigned char'

Args:

'int' i1

Constant

Body

```
return tmpStr[i1];
```

getTotals

Generated , Primitive-operation , Public, Return type is SentTotals

Constant

Body

```
return Totals;
```

response

Generated , Constructor , Public

setCOMMAND

Generated , Primitive-operation , Public, Return type is COMMAND_STRUCTURE

Args:

COMMAND_STRUCTURE p_COMMAND

Body

```
COMMAND = p_COMMAND;
```

setItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Args:

'checkTODO*' p_checkTODO

Body

```
if(p_checkTODO != NULL)
    p_checkTODO->_setItsResponse(this);
_setItsCheckTODO(p_checkTODO);
```

setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(p_database != NULL)
    p_database->_setItsResponse(this);
_setItsDatabase(p_database);
```

setItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Args:

'Serial_IO*' p_Serial_IO

Body

```
if(p_Serial_IO != NULL)
    p_Serial_IO->_setItsResponse(this);
_setItsSerial_IO(p_Serial_IO);
```

setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(p_TODO_Handler != NULL)
    p_TODO_Handler->_setItsResponse(this);
_setItsTODO_Handler(p_TODO_Handler);
```

setTempAtts

Generated , Primitive-operation , Public, Return type is todoattributes

Args:

todoattributes p_tempAtts

Body

```
tempAtts = p_tempAtts;
```

setTmpStr

Generated , Primitive-operation , Public, Return type is 'void'

Args:

'int' i1

'unsigned char' p_tmpStr

Body

```
tmpStr[i1] = p_tmpStr;
```

setTotals

Generated , Primitive-operation , Public, Return type is SentTotals

Args:

SentTotals p_Totals

Body

```
Totals = p_Totals;
```

startBehavior

Virtual, Generated , Primitive-operation , Public, Return type is OMBoolean

Body

```
OMBoolean done = FALSE;
done = OMReactive::startBehavior();
if(done)
    start();
return done;
```

~response

Generated , Destructor , Public

Body

```
cleanUpRelations();
```

Attributes:

COMMAND

structure used to hold the information of the command that has been sent out

Type of COMMAND_STRUCTURE, Public

tempAtts

structure holding the attributes of the command that got sent out so updates can be made and then it can get sent back to the todo queue if necessary

Type of todoattributes, Public

tmpStr

temp string used for various functions

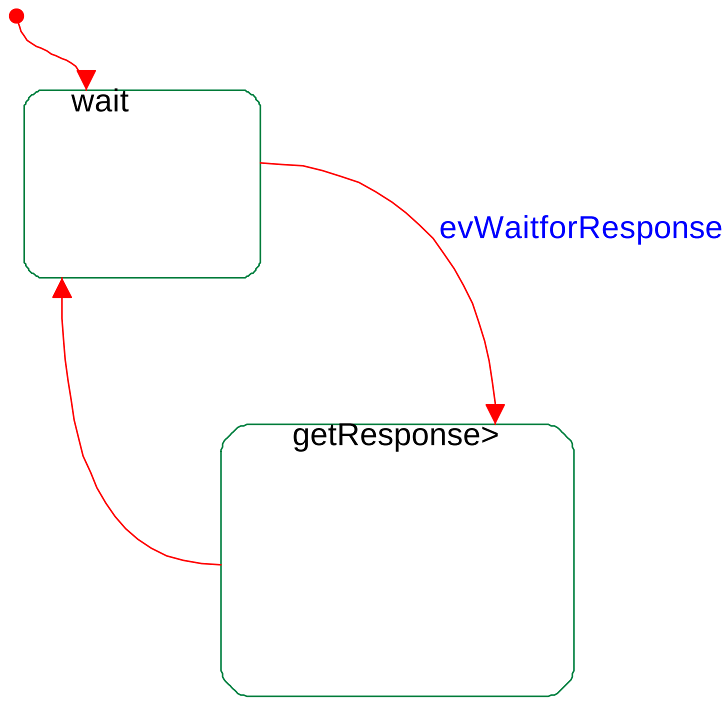
Type of 'unsigned char %s[12];', Public

Totals

structure holding the totals of current and voltage for a given outlet

Type of SentTotals, Public

Statechart



ROOT

Or-state

Substates:

getResponse

wait

Default Transition

Target:

wait

getResponse

wait here for response

Or-state

EntryAction

```
unsigned char sRecieved[128];
unsigned char * ToSend;
CString errorStr, statusStr;
CString ID, total;
CDBVariant newID, errorDate;
CString TotalUpdate;
char * temp = "0";
char timebuf[20];
CRecordset rs( (itsDatabase->getDB()));
```

```
CString getIDStr;
int attempts = 0;
char errStr[200];
int tempADDR;
CString tempCS;
char tempStr[200];
char statusSTR[10];
int errorCount = 0;
TIMESTAMP_STRUCT *ts;
int statusInt;

//while(!itsSerial_IO->getReady()){/*cant do anything until serial port
is open*/}

//while (attempts < 2) {
    cout<<"startWHILE"<<endl;

    if(itsSerial_IO->READ(sRecieved)) {
            cout<<"got something back in allotted amount of
time"<<endl;
            //cout<<"first byte : "<<sRecieved[0]<<endl;
        /*
        for(int i =0 ; i < 14; i++) {
            printf("attempt %d: %.02x",i,sRecieved[i]);
        }
        */
            switch(decifer(sRecieved,params->atts.whatToPoll)) {
                    case 0: // poll
                                    //cout<<"got polling command"<<endl;
                                    if (itsDatabase->isConnected()) {
                                            (void)sprintf(tempStr,"SELECT
ID FROM Wall_Unit where address = %d",params->atts.address);
                                            itsDatabase-
>query(&rs,tempStr);
                                            if(!rs.IsEOF()) {

    rs.GetFieldValue(short(0), ID);
                                            }
                                    }
                                    rs.Close();
                                    if(params->atts.whatToPoll == 0) {
//voltage
                                            (void)sprintf(tempStr,"INSERT
into Voltage_Total (WUID,Voltage) Values ( %s, %f)",ID,Totals.v);
                                            cout<<tempStr<<endl;
                                            //itsDatabase-
>execute(tempStr);
                                    }else if(params->atts.whatToPoll ==
1) {   //voltage
                                            (void)sprintf(tempStr,"INSERT
into Current1_Total (WUID,Current) Values ( %s, %f)",ID,Totals.c1);
                                            cout<<tempStr<<endl;
                                            //itsDatabase-
>execute(tempStr);
                                    }else if(params->atts.whatToPoll ==
2) {   //voltage
                                            (void)sprintf(tempStr,"INSERT
into Current2_Total (WUID,Current) Values ( %s, %f)",ID,Totals.c2);
                                            cout<<tempStr<<endl;
                                            //itsDatabase-
>execute(tempStr);
                                    }
                                    //(void)sprintf(tempStr,"INSERT into
Power_Totals (Date) Values ( '%s' ) WHERE WUID = %s",timebuf,ID);
                                    //cout<<tempStr<<endl;
                                    //itsDatabase->execute(tempStr);
                                    attempts = 5; //some number bigger
then two
                                    itsCheckTODO->GEN(evResponseNeeded);
                                    break;
                    case 1: // ack
```

```
                                    //cout<<"got ack command"<<endl;
                                    cout<<"should have to do
nothing"<<endl;
                                    switch (params->atts.OPCODE) {
                                            case 2 :

    (void)sprintf(statusSTR,"ON");
                                                                break;
                                            case 3 :

    (void)sprintf(statusSTR,"OFF");
                                                                break;
                                            case 4 :

    (void)sprintf(statusSTR,"%d",(int)params->atts.offset);
                                                                break;
                                            default :

    (void)sprintf(statusSTR,"ERROR");
                                                                break;
                                    }
                                    cout<<"new shit on ack"<<endl;

                                    if(params->atts.HI_LOW != 0) {
                                            (void)sprintf(tempStr,"UPDATE
Wall_Unit SET statusHIGH = '%s' WHERE address = %d",statusSTR,params-
>atts.address);
                                    }else {
                                            (void)sprintf(tempStr,"UPDATE
Wall_Unit SET statusLOW = '%s' WHERE address = %d",statusSTR,params-
>atts.address);
                                    }
                                    cout<<"new shit:"<<tempStr<<endl;
                                    attempts = 5; //some number bigger
then two
                                    //itsCheckTODO-
>GEN(evResponseNeeded);
                                    itsDatabase->execute(tempStr);
                                    itsCheckTODO->GEN(evResponseNeeded);
                                    break;
                   default : //shitty
                                    cout<<"bad news bears"<<endl;
                                    //if ( attempts == 1 ) {
                                            if(params->atts.attempts < 2)
{
                                                    //itsTODO_Handler-
>NewTODO(params->atts.address,params->atts.PW,params-
>atts.OPCODE,((params->atts.attempts)+1),params->atts.offset,params-
>atts.HI_LOW,params->atts.whatToPoll);

                                                    tempAtts = params-
>atts;
                                                    tempAtts.attempts +=
1;
                                                    itsCheckTODO-
>GEN(evReDo(tempAtts));
                                            }
                                            else {

                                            if(itsDatabase-
>isConnected()) {


COMMAND.StartSymbol = 0xDB;
COMMAND.To = params->atts.address;
COMMAND.From = 0x0001;
COMMAND.OPCODE = 0x1;
COMMAND.HI_LOW = 0x1;
COMMAND.OFFSet = 0xFF;
COMMAND.CheckSUM = 0xE00; //E is the constant and the rest will be
calculated after been through Build Command
```

```
ToSend = BuildCommand(&COMMAND);
//itsSerial_IO->WRITE(ToSend);



    (void)sprintf(tempStr,"SELECT ID FROM Wall_Unit where address =
%d",params->atts.address);
                                                  itsDatabase-
>query(&rs,tempStr);
                                                  if(!rs.IsEOF()) {

    rs.GetFieldValue(short(0),newID);
                                                          tempADDR =
newID.m_lVal;
                                    }
                                    else {
                                          tempADDR = -1;
                                    }
                                    rs.Close();

    strftime(timebuf,20,"%c",CTime::GetCurrentTime().GetLocalTm());
                                    (void)sprintf(errStr,"INSERT into
ErrorLog values(2,'Attempted 3 times to send command : %d on unit :
%d','%s',%d)",params->atts.OPCODE,tempADDR,timebuf,/*tempADDR*/2);
                                    itsDatabase->execute(errStr);
                                                          if(params-
>atts.whatToPoll != 5) {
                                                                  if
(itsDatabase->isConnected()) {

    (void)sprintf(tempStr,"SELECT ID FROM Wall_Unit WHERE address =
%d",params->atts.address);

    cout<<"line: "<<tempStr<<endl;

    itsDatabase->query(&rs,tempStr);

    if(!rs.IsEOF()) {

    rs.GetFieldValue(short(0), ID);

    rs.Close();
                                                        //cout<<"switch
statement"<<endl;

    switch(params->atts.whatToPoll) {

    case 0:

                                  if (itsDatabase->isConnected()) {


    /*(void)sprintf(tempStr,"SELECT Voltage FROM Voltage_Total WHERE
WUID = %s",ID);

                                          //cout<<"switch:
"<<tempStr<<endl;

                                          itsDatabase-
>query(&rs,tempStr);

                                          if(!rs.IsEOF()) {

                                                  //cout<<"doing the
shit"<<endl;

                                                  while(!rs.IsEOF()) {


    rs.GetFieldValue(short(0), total);
```

```
                                                rs.MoveNext();
                                        }
                                        rs.Close();*/

    (void)sprintf(errStr,"INSERT into Voltage_Total (WUID,Voltage)
Values ( %s, %f)",ID,0.0);
                                        itsDatabase-
>execute(errStr);
                                //}
                        }
                break;
            case 1:
                        if (itsDatabase->isConnected()) {

    /*(void)sprintf(tempStr,"SELECT Current FROM Current1_Total WHERE
WUID = %s",ID);
                                itsDatabase-
>query(&rs,tempStr);
                                if(!rs.IsEOF()) {
                                        while(!rs.IsEOF()) {

    rs.GetFieldValue(short(0), total);
                                                rs.MoveNext();
                                        }
                                        rs.Close();*/

    (void)sprintf(errStr,"INSERT INTO Current1_Total (WUID,Current)
VALUES (%s,%f)",ID,0.0);
                                        itsDatabase-
>execute(errStr);
                                //}
                        }
                break;
            case 2:
                        if (itsDatabase->isConnected()) {

    /*(void)sprintf(tempStr,"SELECT Current FROM Current2_Total WHERE
WUID = %s",ID);
                                itsDatabase-
>query(&rs,tempStr);
                                if(!rs.IsEOF()) {
                                        while(!rs.IsEOF()) {
```

```
    rs.GetFieldValue(short(0), total);
                                                    rs.MoveNext();
                                            }
                                            rs.Close(); */

    (void)sprintf(errStr,"INSERT INTO Current2_Total (WUID,Current)
VALUES (%s,%f)",ID,0.0);
                                            itsDatabase-
>execute(errStr);
                                    //}
                            }
                    break;
            default :
                            cout<<"bad news"<<endl;
                    break;
    }
    //cout<<"doen with switch statement"<<endl;
    }//if eof
                                                            }//if
database
                                                    }//params what
to poll !=5
                            }
                            itsCheckTODO->GEN(evResponseNeeded);
                                    }
                                    //itsCheckTODO-
>GEN(evResponseNeeded);
                            //}
                            //attempts ++;
                            break; //break for default
                    }  //close switch
            }       // if serial read
            else {
                    cout<<"nothing back"<<endl;
                    //if(attempts == 1) {
                            //cout<<"local attempts = 2"<<endl;
                            if(params->atts.attempts < 2) {
                                    //cout<<"total attempts < 2"<<endl;
                                    //itsTODO_Handler->NewTODO(params-
>atts.address,params->atts.PW,params->atts.OPCODE,((params-
>atts.attempts)+1),params->atts.offset,params->atts.HI_LOW,params-
>atts.whatToPoll);
                                    tempAtts = params->atts;
                                    tempAtts.attempts += 1;
                                    itsCheckTODO->GEN(evReDo(tempAtts));
                             }
                             else {
                                    //cout<<"total attempts > 3"<<endl;
                                    if(itsDatabase->isConnected()) {
                                            //cout<<"the address :
"<<params->atts.address<<endl;
                                            (void)sprintf(tempStr,"SELECT
ID FROM Wall_Unit where address = %d",params->atts.address);
                                            //cout<<"this sql statement
"<<tempStr<<endl;
                                            itsDatabase-
```

```
>query(&rs,tempStr);
                                            //cout<<"did that sql
statement"<<endl;
                                            if(!rs.IsEOF()) {

    rs.GetFieldValue(short(0),newID);
                                                    tempADDR =
newID.m_lVal;
                         }
                         else {
                           tempADDR = -1;
                         }
                         rs.Close();
                         errorCount = 0;
                         cout<<"about to check to see if there are many
errors"<<endl;
                                            (void)sprintf(tempStr,"SELECT
Error_Date FROM ErrorLog WHERE WUID = %d",tempADDR);
                                            itsDatabase-
>query(&rs,tempStr);
                                            while(!rs.IsEOF()) {
                                                    //cout<<"thre are some
errors checking"<<endl;

    rs.GetFieldValue(short(0),errorDate);
                                                    ts =
errorDate.m_pdate;
                                                    if( ((*ts).year <
CTime::GetCurrentTime().GetYear()) ||
                                                            (
    ((*ts).year == CTime::GetCurrentTime().GetYear()) &&

    ((*ts).month < CTime::GetCurrentTime().GetMonth()) ) ||
                                                                    (
    ((*ts).year == CTime::GetCurrentTime().GetYear()) &&

    ((*ts).month == CTime::GetCurrentTime().GetMonth()) &&

    ((*ts).day <= CTime::GetCurrentTime().GetDay()) )
                                                            ) {

    if((*ts).hour == CTime::GetCurrentTime().GetHour() ) {

    cout<<"found one in this hour"<<endl;

    errorCount++;
                                                                    }
                                                            rs.MoveNext();
                                                    }
                                            }
                                            rs.Close();

                                            if(errorCount >= 3) {
                                                    //reset the device
                                                    cout<<"more then three
in this hour"<<endl;
                                                    COMMAND.StartSymbol =
0xDB;
                                                    COMMAND.To = params-
>atts.address;
                                                    COMMAND.From = 0x0001;
                                                    COMMAND.OPCODE = 0x1;
                                                    COMMAND.HI_LOW = 0x1;
                                                    COMMAND.OFFSet = 0xFF;
                                                    COMMAND.CheckSUM =
0xE00; //E is the constant and the rest will be calculated after been
through Build Command
                                                    ToSend =
BuildCommand(&COMMAND);
                                                    //itsSerial_IO-
>WRITE(ToSend);
```

```
                                                  //now need to re-send
last known state
                                                  if(params->atts.HI_LOW
== 1) {

    (void)sprintf(tempStr,"SELECT statusHIGH FROM Wall_Unit WHERE ID =
%d",tempADDR);
                                                          itsDatabase-
>query(&rs,tempStr);

    if(!rs.IsEOF()) {

    rs.GetFieldValue(short(0),statusStr);

    if(statusStr.Compare("ON") == 0 ) {

    //send on command

    //itsTODO_Handler->NewTODO(params->atts.address,params-
>atts.PW,0x2,0,params->atts.offset,1,params->atts.whatToPoll);

    cout<<"built on reset command should be sending next"<<endl;
                                                                  }
                                                                  else
if(statusStr.Compare("ON") == 0 ) {

    //itsTODO_Handler->NewTODO(params->atts.address,params-
>atts.PW,0x3,0,params->atts.offset,1,params->atts.whatToPoll);

cout<<"built off reset command should be sending next"<<endl;
                                                                  }
                                                                  else {

    statusInt = atoi(statusStr);

    //itsTODO_Handler->NewTODO(params->atts.address,params-
>atts.PW,statusInt,0,params->atts.offset,1,params->atts.whatToPoll);
                                                                  }
                                                          }
                                                  }
                                                  else {

    (void)sprintf(tempStr,"SELECT statusLOW FROM Wall_Unit WHERE ID =
%d",tempADDR);
                                                          itsDatabase-
>query(&rs,tempStr);

    if(!rs.IsEOF()) {

    rs.GetFieldValue(short(0),statusStr);

    cout<<endl<<"THe status string is : "<<statusStr<<endl<<endl;

    if(statusStr.Compare("ON") == 0 ) {


    COMMAND.StartSymbol = 0xDB;

    COMMAND.To = params->atts.address;

    COMMAND.From = 0x0001;

    COMMAND.OPCODE = 0x2;

    COMMAND.HI_LOW = 0x1;

    COMMAND.OFFSet = 0xFF;

    COMMAND.CheckSUM = 0xE00;
```

```
    //ToSend = BuildCommand(&COMMAND);

    //itsSerial_IO->WRITE(ToSend);


    //send on command

    //itsTODO_Handler->NewTODO(params->atts.address,params-
>atts.PW,0x2,0,params->atts.offset,0,params->atts.whatToPoll);

    cout<<"built on reset command should be sending next"<<endl;
                                                                }
                                                                else
if(statusStr.Compare("OFF") == 0 ) {

    COMMAND.StartSymbol = 0xDB;

    COMMAND.To = params->atts.address;

    COMMAND.From = 0x0001;

    COMMAND.OPCODE = 0x3;

    COMMAND.HI_LOW = 0x1;

    COMMAND.OFFSet = 0xFF;

    COMMAND.CheckSUM = 0xE00;

    //ToSend = BuildCommand(&COMMAND);

    //itsSerial_IO->WRITE(ToSend);

    //itsTODO_Handler->NewTODO(params->atts.address,params-
>atts.PW,0x3,0,params->atts.offset,0,params->atts.whatToPoll);

cout<<"built off reset command should be sending next"<<endl;
                                                                }
                                                                else {

    COMMAND.StartSymbol = 0xDB;

    COMMAND.To = params->atts.address;

    COMMAND.From = 0x0001;

    COMMAND.OPCODE = 0x2;

    COMMAND.HI_LOW = 0x4;

    COMMAND.OFFSet = atoi(statusStr);

    COMMAND.CheckSUM = 0xE00;

    //ToSend = BuildCommand(&COMMAND);

    //itsSerial_IO->WRITE(ToSend);

    //statusInt = atoi(statusStr);

    //itsTODO_Handler->NewTODO(params->atts.address,params-
>atts.PW,statusInt,0,params->atts.offset,0,params->atts.whatToPoll);
                                                                }
                                                        }
                                                        rs.Close();
                                                }


                                        }
                                        errorCount = 0;
```

```
                        //cout<<"got the id"<<endl;

    strftime(timebuf,20,"%c",CTime::GetCurrentTime().GetLocalTm());
                                        //cout<<"got the time"<<endl;
                                        if(tempADDR == -1) {

    (void)sprintf(errStr,"INSERT into
ErrorLog(Error_Code,Error_Text,Error_Date) values (2, 'Attempted 3
times to send command : %d on unit : %d','%s')",params-
>atts.OPCODE,tempADDR,timebuf);
                                        }
                                        else {
                          (void)sprintf(errStr,"INSERT into ErrorLog
values(2,'Attempted 3 times to send command : %d on unit :
%d','%s',%d)",params->atts.OPCODE,tempADDR,timebuf,tempADDR);
                        }
                        cout<<"made the string"<<endl<<errStr<<endl;
                        itsDatabase->execute(errStr);
                        cout<<"executed the statement"<<endl;
                                        if(params->atts.whatToPoll !=
5) {
                                                if (itsDatabase-
>isConnected()) {

    (void)sprintf(tempStr,"SELECT ID FROM Wall_Unit WHERE address =
%d",params->atts.address);
                                                        cout<<"line:
"<<tempStr<<endl;
                                                        itsDatabase-
>query(&rs,tempStr);

    if(!rs.IsEOF()) {

    rs.GetFieldValue(short(0), ID);

    rs.Close();
                                      cout<<"switch
statement1"<<endl;
                                                switch(params-
>atts.whatToPoll) {
                                                        case 0:

                  if (itsDatabase->isConnected()) {

                          /*(void)sprintf(tempStr,"SELECT Voltage FROM
Voltage_Total WHERE WUID = %s",ID);

                          //cout<<"switch: "<<tempStr<<endl;

                          itsDatabase->query(&rs,tempStr);

                          //cout<<"did statement"<<endl;

                          if(!rs.IsEOF()) {

                                  //cout<<"doing shit"<<endl;

                                  while(!rs.IsEOF()) {

                                          rs.GetFieldValue(short(0),
total);

                                          rs.MoveNext();

                                  }

                                  rs.Close(); */

                                  (void)sprintf(errStr,"INSERT INTO
Voltage_Total (WUID,Voltage) VALUES (%s,%f)",ID,0.0);
```

```
                //itsDatabase->execute(errStr);
            //}
        }
    break;
    case 1:
        if (itsDatabase->isConnected()) {
            /*(void)sprintf(tempStr,"SELECT Current FROM Current1_Total WHERE WUID = %s",ID);
            cout<<"switch: "<<tempStr<<endl;
            itsDatabase->query(&rs,tempStr);
            while(!rs.IsEOF()) {
                rs.GetFieldValue(short(0), total);
                rs.MoveNext();
            }
        rs.Close();  */
        (void)sprintf(errStr,"INSERT INTO Current2_Total (WUID,Current) VALUES (%s,%f)",ID,0.0);
        //itsDatabase->execute(errStr);
        }
    break;
    case 2:
        if (itsDatabase->isConnected()) {
            /*(void)sprintf(tempStr,"SELECT Current FROM Current2_Total WHERE WUID = %s",ID);
            cout<<"switch: "<<tempStr<<endl;
            itsDatabase->query(&rs,tempStr);
            while(!rs.IsEOF()) {
                rs.GetFieldValue(short(0), total);
                rs.MoveNext();
            }

        rs.Close(); */
        (void)sprintf(errStr,"INSERT INTO Current2_Total (WUID,Current) VALUES (%s,%f)",ID,0.0);
        //itsDatabase->execute(errStr);
        }
    break;
```

```
default :

                //cout<<"bad news"<<endl;

                //error message into database

break;
                                                              }

//cout<<"doen with switch statement1"<<endl;
                                                      }
                                              }
                                      }
                 //itsCheckTODO->GEN(evResponseNeeded);
                 }
                 itsCheckTODO->GEN(evResponseNeeded);
                         }
                       //itsCheckTODO->GEN(evResponseNeeded);

                //}
                attempts++;
        }//end of els for read
//}  // end of function while
//cout<<"cout of while moving on"<<endl;
//itsCheckTODO->GEN(evResponseNeeded);
cout<<"okay its not in response i think"<<endl;
```

Out Transition

Target:

wait

wait

temp state for waiting until a response is comming

Or-state

Out Transition

```
evWaitforResponse
```

Target:

getResponse

Serial_IO

this class offers serial communication. functions to be used are read and write

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsCheckTODO

Association with checkTODO, Multiplicity of 1, Bi-directional

itsResponse

Association with response, Multiplicity of 1, Bi-directional

itsSystemSetup

Association with systemSetup, Multiplicity of 1, Bi-directional

Operations:

__setItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Args:

'checkTODO*' p_checkTODO

Body

```
itsCheckTODO = p_checkTODO;
```

__setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
itsResponse = p_response;
```

__setItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Args:

'systemSetup*' p_systemSetup

Body

```
itsSystemSetup = p_systemSetup;
```

_clearItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Body

```
itsCheckTODO = NULL;
```

_clearItsResponse

Generated , Primitive-operation , Public, Return type is void

Body

```
itsResponse = NULL;
```

_clearItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Body

```
itsSystemSetup = NULL;
```

_setItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Args:

'checkTODO*' p_checkTODO

Body

```
if(itsCheckTODO != NULL)
    itsCheckTODO->__setItsSerial_IO(NULL);
__setItsCheckTODO(p_checkTODO);
```

_setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
if(itsResponse != NULL)
    itsResponse->__setItsSerial_IO(NULL);
__setItsResponse(p_response);
```

_setItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Args:

'systemSetup*' p_systemSetup

Body

```
if(itsSystemSetup != NULL)
    itsSystemSetup->__setItsSerial_IO(NULL);
__setItsSystemSetup(p_systemSetup);
```

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsCheckTODO != NULL)
    {

        Serial_IO* p_Serial_IO = itsCheckTODO->getItsSerial_IO();
        if(p_Serial_IO != NULL)
```

```
                itsCheckTODO->__setItsSerial_IO(NULL);
            itsCheckTODO = NULL;
        }
    if(itsResponse != NULL)
        {

            Serial_IO* p_Serial_IO = itsResponse->getItsSerial_IO();
            if(p_Serial_IO != NULL)
                itsResponse->__setItsSerial_IO(NULL);
            itsResponse = NULL;
        }
    if(itsSystemSetup != NULL)
        {

            Serial_IO* p_Serial_IO = itsSystemSetup->getItsSerial_IO();
            if(p_Serial_IO != NULL)
                itsSystemSetup->__setItsSerial_IO(NULL);
            itsSystemSetup = NULL;
        }
```

getHComm

Generated , Primitive-operation , Public, Return type is 'HANDLE '

Constant

Body

```
return hComm;
```

getItsCheckTODO

Generated , Primitive-operation , Public, Return type is 'checkTODO*'

Constant

Body

```
return itsCheckTODO;
```

getItsResponse

Generated , Primitive-operation , Public, Return type is 'response*'

Constant

Body

```
return itsResponse;
```

getItsSystemSetup

Generated , Primitive-operation , Public, Return type is 'systemSetup*'

Constant

Body

```
return itsSystemSetup;
```

getReady

Generated , Primitive-operation , Public, Return type is OMBoolean

Constant

Body

```
return Ready;
```

READ

funciton used to read data from the serial port

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is OMBoolean

Args:

'unsigned char *' sResult

Body

```
BOOL    bReadRC;
DWORD   iBytesRead;
DWORD   dwError;
char    sMsg[512];

    //cout<<"here reading"<<endl;

    Ready = false;

bReadRC = ReadFile(hComm, (char*)sResult, 30, &iBytesRead, NULL);
if (bReadRC && iBytesRead > 0)
{
        //sResult = sBuffer;
}
else
{
        dwError = GetLastError();

        sprintf(sMsg, "Read length failed: RC=%d Bytes read=%d, "
                       "Error=%d ",
                       bReadRC, iBytesRead, dwError);
        //AfxMessageBox(sMsg);
        //cout<<"nothing to read, matt fix your shit"<<endl;
        Ready = true;
        return false;
} // end if
//cout<<"bytes read are : "<<sResult<<endl;
Ready = true;
return true;
```

Serial_IO

Constructor , Public

Body

```
//HANDLE hComm;
DWORD   dwError;
DWORD   dwRC;
char    sMsg[512];
BOOL m_bPortReady = TRUE;
DCB             m_dcb;
COMMTIMEOUTS m_CommTimeouts;
//int timer = 0;
hComm = CreateFile( "Com1",
                    GENERIC_READ | GENERIC_WRITE,
                    0,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL);
if (hComm == INVALID_HANDLE_VALUE) {
 //cout<<"error read comm"<<endl;
 dwError = GetLastError();
        dwError = GetLastError();

        // example error code expansion follows
        LPVOID lpMsgBuf;
        lpMsgBuf = NULL;
        dwRC = FormatMessage(
                FORMAT_MESSAGE_ALLOCATE_BUFFER |
                FORMAT_MESSAGE_FROM_SYSTEM |
                FORMAT_MESSAGE_IGNORE_INSERTS,
                 NULL,
                dwError, //  from GetLastError(),
                MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), //
Default language
                (LPTSTR) &lpMsgBuf,
                0,
                NULL);

        if (dwRC && lpMsgBuf)
```

```
        {
                sprintf(sMsg, "COM open failed: Port=%s Error=%d -
%s",
                        hComm, dwError, lpMsgBuf);
                AfxMessageBox(sMsg);
        }
        else
        {
                //sprintf(sMsg, "COM open failed: Port=%s Error=%d
",
                        //hComm, dwError);
                //AfxMessageBox(sMsg);
                cout<<"there is something using your com1 port
please fix and restart"<<endl;
        } // end if
    //error opening port; abort
 m_bPortReady = FALSE;
}
 if (m_bPortReady)
 {
        m_bPortReady = SetupComm(hComm,
                        1200, 1200); // set buffer sizes
        if (!m_bPortReady)
        {
                dwError = GetLastError();
                sprintf(sMsg, "SetupComm failed: Port=%s Error=%d",
                        "Com1", dwError);
                AfxMessageBox(sMsg);

        } // end if
 } // end if

 if (m_bPortReady)
 {
        m_bPortReady = GetCommState(hComm, &m_dcb);
        if (!m_bPortReady)
        {
                dwError = GetLastError();
                sprintf(sMsg, "GetCommState failed: Port=%s
Error=%d",
                        "Com1", dwError);
                AfxMessageBox(sMsg);
        } // end if
 } // end if

 if (m_bPortReady)
 {
        m_dcb.BaudRate = CBR_2400;
        m_dcb.fParity = FALSE;
        m_dcb.fDsrSensitivity = FALSE;
        m_dcb.fOutxCtsFlow = FALSE;
        m_dcb.fOutX = FALSE;
        m_dcb.fInX = FALSE;
        m_dcb.fOutxDsrFlow = FALSE;
        m_dcb.fDtrControl = DTR_CONTROL_DISABLE;
        m_dcb.fRtsControl = RTS_CONTROL_DISABLE;
        m_dcb.ByteSize = 8;
        m_dcb.Parity = NOPARITY;
        m_dcb.StopBits = ONESTOPBIT;
        m_dcb.fAbortOnError = TRUE;

        m_bPortReady = SetCommState(hComm, &m_dcb);
        if (!m_bPortReady)
        {
                dwError = GetLastError();
                sprintf(sMsg, "SetCommState failed: Port=%s Error =
%d",
                        "Com1", dwError);
                AfxMessageBox(sMsg);
        }
 } // end if
```

```
 if (m_bPortReady)
 {
        m_bPortReady = GetCommTimeouts (hComm, &m_CommTimeouts);
        if (!m_bPortReady)
        {
               dwError = GetLastError();
               sprintf(sMsg, "GetCommTimeouts failed: Port=%s Error
= %d",
                       "Com1", dwError);
               AfxMessageBox(sMsg);
        } // end if
 } // end if

 if (m_bPortReady)
 {
        m_CommTimeouts.ReadIntervalTimeout = 40;
        m_CommTimeouts.ReadTotalTimeoutConstant = 3000;
        m_CommTimeouts.ReadTotalTimeoutMultiplier = 1;
        //cout<<"now we have set the timeout"<<endl;
        m_CommTimeouts.WriteTotalTimeoutConstant = 50;
        m_CommTimeouts.WriteTotalTimeoutMultiplier = 10;
        m_bPortReady = SetCommTimeouts (hComm, &m_CommTimeouts);
        if (!m_bPortReady)
        {
               dwError = GetLastError();
               sprintf(sMsg, "SetCommTimeouts failed: Port=%s Error
= %d",
                       "Com1", dwError);
               AfxMessageBox(sMsg);
        } // end if
 } // end if
 if (m_bPortReady) {
        Ready = true;
        cout<<"ready"<<endl;
 }
 else {
        Ready = false;
        cout<<"notready"<<endl;
 }
```

setHComm

Generated , Primitive-operation , Public, Return type is 'void'

Args:

'HANDLE %s' p_hComm

Body

```
hComm = p_hComm;
```

setItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Args:

'checkTODO*' p_checkTODO

Body

```
if(p_checkTODO != NULL)
    p_checkTODO->_setItsSerial_IO(this);
_setItsCheckTODO(p_checkTODO);
```

setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
if(p_response != NULL)
    p_response->_setItsSerial_IO(this);
_setItsResponse(p_response);
```

setItsSystemSetup

Generated , Primitive-operation , Public, Return type is void

Args:

'systemSetup*' p_systemSetup

Body

```
if(p_systemSetup != NULL)
    p_systemSetup->_setItsSerial_IO(this);
_setItsSystemSetup(p_systemSetup);
```

setReady

Generated , Primitive-operation , Public, Return type is OMBoolean

Args:

OMBoolean p_Ready

Body

```
Ready = p_Ready;
```

startBehavior

Virtual, Generated , Primitive-operation , Public, Return type is OMBoolean

Body

```
OMBoolean done = FALSE;
done = OMReactive::startBehavior();
if(done)
    start();
return done;
```

WRITE

function used to write data to the serial port

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is OMBoolean

Args:

'unsigned char * %s' sWrite

Body

```
DWORD   iBytesWritten;
BOOL    bWriteRC;
DWORD   dwError;
char    sMsg[512];
char * DataTOsend = "Hello Matt";
Ready = false;
cout<<"lets see"<<endl;
/*(char*)&DataTOsend[0] = (char*)sWrite.StartSymbol;
(char*)&DataTOsend[0] = DataTOsend + *((char*)sWrite.To);
(char*)&DataTOsend[0] = DataTOsend + *((char*)sWrite.From);
(char*)&DataTOsend[0] = DataTOsend + *((char*)sWrite.HI_LOW);
(char*)&DataTOsend[0] = DataTOsend + *((char*)sWrite.OPCODE);
(char*)&DataTOsend[0] = DataTOsend + *((char*)sWrite.OFFSet);
(char*)&DataTOsend[0] = DataTOsend + *((char*)sWrite.CheckSUM);
*/
cout<<"hmmm2mmmm"<<endl;
iBytesWritten = 0;
bWriteRC = WriteFile(hComm,
sWrite/*DataTOsend*/,12,&iBytesWritten,NULL);
if (!bWriteRC || iBytesWritten == 0)
{
 dwError = GetLastError();

 sprintf(sMsg, "Write of length query failed: RC=%d, "
                       "Bytes Written=%d, Error=%d",
                       bWriteRC, iBytesWritten, dwError);
 AfxMessageBox(sMsg);
 Ready = true;
```

```
 return false;
} // end if
Ready = true;
return true;
```

~Serial_IO

Destructor , Public

Body

```
CloseHandle(hComm);
cleanUpRelations();
```

Attributes:

hComm

handle to the comm port

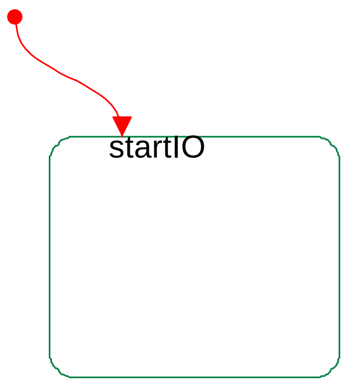
Type of 'HANDLE %s;', Public

Ready

boolean to let the system know if the serial port is ready for communication

Type of OMBoolean, Public, Initial Value: FALSE

Statechart



ROOT

Or-state

Substates:

startIO

Default Transition

Target:

startIO

startIO

temp state to start the class

Or-state

---

systemChecks

thsi class waits for the end of each day and then cleans up the database. if it is the end of the month then if reduces the voltage and current totals into one entry

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsDatabase

Association with database, Multiplicity of 1, Bi-directional

Operations:

__setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
itsDatabase = p_database;
```

_clearItsDatabase

Generated , Primitive-operation , Public, Return type is void

Body

```
itsDatabase = NULL;
```

_setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(itsDatabase != NULL)
    itsDatabase->__setItsSystemChecks(NULL);
__setItsDatabase(p_database);
```

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsDatabase != NULL)
    {

        systemChecks* p_systemChecks = itsDatabase-
>getItsSystemChecks();
        if(p_systemChecks != NULL)
            itsDatabase->__setItsSystemChecks(NULL);
        itsDatabase = NULL;
    }
```

endOfDayCalc

function to calculate the number of seconds until the end of the day

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is int

Body

```
//cout<<"seconds till end of day"<<( (24*60*60) - (
(CTime::GetCurrentTime().GetHour()*60*60) +
(CTime::GetCurrentTime().GetMinute()*60) +
CTime::GetCurrentTime().GetSecond() ) )<<endl;
return ( (24*60*60) - ( (CTime::GetCurrentTime().GetHour()*60*60) +
(CTime::GetCurrentTime().GetMinute()*60) +
CTime::GetCurrentTime().GetSecond() ) );
```

getItsDatabase

Generated , Primitive-operation , Public, Return type is 'database*'

Constant

Body

```
return itsDatabase;
```

setItsDatabase

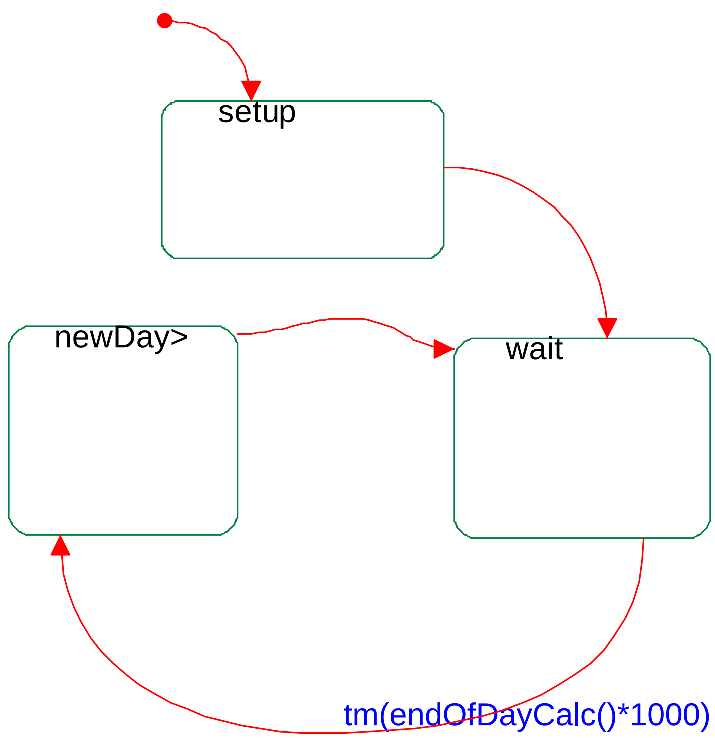
Generated , Primitive-operation , Public, Return type is void

Args:
'database*' p_database
Body
```
if(p_database != NULL)
    p_database->_setItsSystemChecks(this);
_setItsDatabase(p_database);
```

startBehavior
Virtual, Generated , Primitive-operation , Public, Return type is OMBoolean
Body
```
OMBoolean done = FALSE;
done = OMReactive::startBehavior();
if(done)
    start();
return done;
```

systemChecks
Generated , Constructor , Public
~systemChecks
Generated , Destructor , Public
Body
```
cleanUpRelations();
```

Statechart



ROOT
Or-state
Substates:
newDay
setup
wait
Default Transition
Target:
setup
newDay

clean up the database

Or-state

EntryAction

```
CRecordset rs( (itsDatabase->getDB()));
CRecordset rs1( (itsDatabase->getDB()));
char exeStr[200];
CDBVariant WUID,DATE,VOLTAGE,CURRENT1,CURRENT2, ID;
TIMESTAMP_STRUCT *ts;
double MVoltage, MCurrent1, MCurrent2;
char timebuf[20];
int counter;

if(itsDatabase->isConnected()) {

    if ( CTime::GetCurrentTime().GetDay() == 26 ) {
            //this means its the first day of the month time to
condense power totals

            itsDatabase->query(&rs,"SELECT WUID FROM Voltage_Total");
            while(!rs.IsEOF()) {
                rs.GetFieldValue(short(0),WUID);
                //cout<<"here1"<<endl;
                MVoltage = 0;
                MCurrent1 = 0;
                MCurrent2 = 0;
                counter = 0;
                    (void)sprintf(exeStr,"SELECT Date, Voltage, ID FROM
Voltage_Total WHERE WUID = %d",WUID.m_lVal);
                    itsDatabase->query(&rs1,exeStr);
                    while(!rs1.IsEOF()) {
                            //cout<<"here2"<<endl;
                            counter++;
                            rs1.GetFieldValue(short(0),DATE);
                            ts = DATE.m_pdate;
                            if( (*ts).year ==
CTime::GetCurrentTime().GetYear() ) {
                                    if ( (*ts).month ==
CTime::GetCurrentTime().GetMonth() ) {
                                            //cout<<"here3"<<endl;

    rs1.GetFieldValue(short(1),VOLTAGE);

    //rs1.GetFieldValue(short(2),CURRENT1);

    //rs1.GetFieldValue(short(3),CURRENT2);

    rs1.GetFieldValue(short(2),ID);

                                            MVoltage += VOLTAGE.m_dblVal;
                                            //MCurrent1 +=
CURRENT1.m_dblVal;
                                            //MCurrent2 +=
CURRENT2.m_dblVal;
                                            //cout<<"values :
"<<VOLTAGE.m_dblVal<<" : "<<CURRENT1.m_dblVal<<" :
"<<CURRENT2.m_dblVal<<endl;
                                            (void)sprintf(exeStr,"DELETE
* FROM Voltage_Total WHERE WUID = %d AND ID =
%d",WUID.m_lVal,ID.m_lVal);
                                            //cout<<"the delete :
"<<endl<<exeStr<<endl;
                                            itsDatabase->execute(exeStr);

                                    }
                            }
                            //cout<<"here4"<<endl;
                            rs1.MoveNext();
                    }
                    rs1.Close();
```

```
                   //cout<<"trying this time thing"<<endl;
                   CTime
time(CTime::GetCurrentTime().GetYear(),CTime::GetCurrentTime().GetMonth
(),1,0,0,0,-1);
                   strftime(timebuf,20,"%c",time.GetLocalTm());
               MVoltage = MVoltage / counter;
               //MCurrent1 = MCurrent1 / counter;
               //MCurrent2 = MCurrent2 / counter;
                   (void)sprintf(exeStr,"INSERT into Voltage_Monthly
VALUES (%d,'%s',%f)",WUID.m_lVal,timebuf,MVoltage);
                   //cout<<"it should explain itself
:"<<endl<<exeStr<<endl;
                   itsDatabase->execute(exeStr);
                   //cout<<"died"<<endl;
                   rs.MoveNext();
           }
           rs.Close();
        //cout<<"voltage done"<<endl;

    //***********************************CURRENT1*******************
*************************

           itsDatabase->query(&rs,"SELECT WUID FROM Current1_Total");
           while(!rs.IsEOF()) {
               rs.GetFieldValue(short(0),WUID);
               //cout<<"here1"<<endl;
               MVoltage = 0;
               MCurrent1 = 0;
               MCurrent2 = 0;
               counter = 0;
                   (void)sprintf(exeStr,"SELECT Date, Current, ID FROM
Current1_Total WHERE WUID = %d",WUID.m_lVal);
                   itsDatabase->query(&rs1,exeStr);
                   while(!rs1.IsEOF()) {
                           //cout<<"here2"<<endl;
                           counter++;
                           rs1.GetFieldValue(short(0),DATE);
                           ts = DATE.m_pdate;
                           if( (*ts).year ==
CTime::GetCurrentTime().GetYear() ) {
                                   if ( (*ts).month ==
CTime::GetCurrentTime().GetMonth() ) {
                                           //cout<<"here3"<<endl;

    //rs1.GetFieldValue(short(1),VOLTAGE);

    rs1.GetFieldValue(short(1),CURRENT1);

    //rs1.GetFieldValue(short(3),CURRENT2);

    rs1.GetFieldValue(short(2),ID);

                                           //MVoltage +=
VOLTAGE.m_dblVal;
                                           MCurrent1 +=
CURRENT1.m_dblVal;
                                           //MCurrent2 +=
CURRENT2.m_dblVal;
                                           //cout<<"values :
"<<VOLTAGE.m_dblVal<<" : "<<CURRENT1.m_dblVal<<" :
"<<CURRENT2.m_dblVal<<endl;
                                           (void)sprintf(exeStr,"DELETE
* FROM Current1_Total WHERE WUID = %d AND ID =
%d",WUID.m_lVal,ID.m_lVal);
                                           //cout<<"the delete :
"<<endl<<exeStr<<endl;
                                           itsDatabase->execute(exeStr);

                                   }
                           }
                           //cout<<"here4"<<endl;
```

```
                        rs1.MoveNext();
                }
                rs1.Close();

                //cout<<"trying this time thing"<<endl;
                CTime
time(CTime::GetCurrentTime().GetYear(),CTime::GetCurrentTime().GetMonth
(),1,0,0,0,-1);
                strftime(timebuf,20,"%c",time.GetLocalTm());
            //MVoltage = MVoltage / counter;
            MCurrent1 = MCurrent1 / counter;
            //MCurrent2 = MCurrent2 / counter;
                (void)sprintf(exeStr,"INSERT into Current1_Monthly
VALUES (%d,'%s',%f)",WUID.m_lVal,timebuf,MCurrent1);
                //cout<<"it should explain itself
:"<<endl<<exeStr<<endl;
                itsDatabase->execute(exeStr);
                //cout<<"died"<<endl;
                rs.MoveNext();
        }
        rs.Close();
        //cout<<"current1 done"<<endl;

    //***********************************CURRENT2********************
************************

        itsDatabase->query(&rs,"SELECT WUID FROM Current2_Total");
        while(!rs.IsEOF()) {
            rs.GetFieldValue(short(0),WUID);
            //cout<<"here1"<<endl;
            MVoltage = 0;
            MCurrent1 = 0;
            MCurrent2 = 0;
            counter = 0;
                (void)sprintf(exeStr,"SELECT Date, Current, ID FROM
Current2_Total WHERE WUID = %d",WUID.m_lVal);
                itsDatabase->query(&rs1,exeStr);
                while(!rs1.IsEOF()) {
                        //cout<<"here2"<<endl;
                        counter++;
                        rs1.GetFieldValue(short(0),DATE);
                        ts = DATE.m_pdate;
                        if( (*ts).year ==
CTime::GetCurrentTime().GetYear() ) {
                                if ( (*ts).month ==
CTime::GetCurrentTime().GetMonth() ) {
                                        //cout<<"here3"<<endl;

    //rs1.GetFieldValue(short(1),VOLTAGE);

    //rs1.GetFieldValue(short(2),CURRENT1);

    rs1.GetFieldValue(short(1),CURRENT2);

    rs1.GetFieldValue(short(2),ID);

                                        //MVoltage +=
VOLTAGE.m_dblVal;
                                        //MCurrent1 +=
CURRENT1.m_dblVal;
                                        MCurrent2 +=
CURRENT2.m_dblVal;
                                        //cout<<"values :
"<<VOLTAGE.m_dblVal<<" : "<<CURRENT1.m_dblVal<<" :
"<<CURRENT2.m_dblVal<<endl;
                                        (void)sprintf(exeStr,"DELETE
* FROM Current2_Total WHERE WUID = %d AND ID =
%d",WUID.m_lVal,ID.m_lVal);
                                        //cout<<"the delete :
"<<endl<<exeStr<<endl;
                                        itsDatabase->execute(exeStr);
```

```
                                        }
                                }
                                //cout<<"here4"<<endl;
                                rs1.MoveNext();
                        }
                        rs1.Close();

                        //cout<<"trying this time thing"<<endl;
                        CTime
time(CTime::GetCurrentTime().GetYear(),CTime::GetCurrentTime().GetMonth
(),1,0,0,0,-1);
                        strftime(timebuf,20,"%c",time.GetLocalTm());
                    //MVoltage = MVoltage / counter;
                    //MCurrent1 = MCurrent1 / counter;
                    MCurrent2 = MCurrent2 / counter;
                        (void)sprintf(exeStr,"INSERT into Current2_Monthly
VALUES (%d,'%s',%f)",WUID.m_lVal,timebuf,MCurrent2);
                        //cout<<"it should explain itself
:"<<endl<<exeStr<<endl;
                        itsDatabase->execute(exeStr);
                        //cout<<"died"<<endl;
                        rs.MoveNext();
                }
                rs.Close();


        }

        //monthly totals are done, now we can check for event sequences
that are past date

        cout<<"deleteing event sequences"<<endl;

        itsDatabase->query(&rs,"SELECT EndDate, ID FROM event_sequence");
        while(!rs.IsEOF()) {

                rs.GetFieldValue(short(0), DATE);
                rs.GetFieldValue(short(1), ID);
                if (DATE.m_dwType == DBVT_DATE) {
                        ts = DATE.m_pdate;
                        if(             ((*ts).year ==
CTime::GetCurrentTime().GetYear()) &&
                                        ((*ts).month ==
CTime::GetCurrentTime().GetMonth()) &&
                                        ((*ts).day ==
CTime::GetCurrentTime().GetDay()-1)

                                                                    ) {
                                (void)sprintf(exeStr,"DELETE * FROM
ES_effects_WU WHERE ESID = %d",ID.m_lVal);
                                itsDatabase->execute(exeStr);
                                (void)sprintf(exeStr,"DELETE * FROM
event_sequence WHERE ID = %d",ID.m_lVal);
                                itsDatabase->execute(exeStr);

                        }
                }
                rs.MoveNext();
        }
        rs.Close();
}
```

Out Transition

Target:

wait

setup

dummy start state

Or-state

Out Transition

Target:

wait

wait

wait untilt the end of the day

Or-state

Out Transition

```
tm(endOfDayCalc()*1000)
```

Target:

newDay

---

systemSetup

function that checks if there are wall unit entries in the database when the system is turned on. if not then it searches for wall units. this can also be iniciated by the user with an immediate action

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

itsDatabase

Association with database, Multiplicity of 1, Bi-directional

itsSerial_IO

Association with Serial_IO, Multiplicity of 1, Bi-directional

itsImmediate

Association with Immediate, Multiplicity of 1, Bi-directional

Operations:

__setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
itsDatabase = p_database;
```

__setItsImmediate

Generated , Primitive-operation , Public, Return type is void

Args:

'Immediate*' p_Immediate

Body

```
itsImmediate = p_Immediate;
```

__setItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Args:

'Serial_IO*' p_Serial_IO

Body

```
itsSerial_IO = p_Serial_IO;
```

_clearItsDatabase

Generated , Primitive-operation , Public, Return type is void

Body

```
itsDatabase = NULL;
```

_clearItsImmediate

Generated , Primitive-operation , Public, Return type is void

Body

```
itsImmediate = NULL;
```

_clearItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Body

```
itsSerial_IO = NULL;
```

_setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(itsDatabase != NULL)
    itsDatabase->__setItsSystemSetup(NULL);
__setItsDatabase(p_database);
```

_setItsImmediate

Generated , Primitive-operation , Public, Return type is void

Args:

'Immediate*' p_Immediate

Body

```
if(itsImmediate != NULL)
    itsImmediate->__setItsSystemSetup(NULL);
__setItsImmediate(p_Immediate);
```

_setItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Args:

'Serial_IO*' p_Serial_IO

Body

```
if(itsSerial_IO != NULL)
    itsSerial_IO->__setItsSystemSetup(NULL);
__setItsSerial_IO(p_Serial_IO);
```

BuildCommand

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is 'unsigned char *'

Args:

'COMMAND_STRUCTURE * %s' COMMAND

Body

```
#define byteswap32(a)
((a<<24)&0xff000000)|((a<<8)&0x00ff0000)|((a>>8)&0x0000ff00)|((a>>24
)&0x000000ff)
#define byteswap16(a) ((a>>8)&0x00ff)|((a<<8)&0xff00)

//char tmpStr[10];
//tmpStr[0] = 'E';
tmpStr[0] = (COMMAND->StartSymbol & 0xFF);
*((long*)&tmpStr[1]) = ((long)byteswap32(COMMAND->To));
*((int*)&tmpStr[5]) = ((int)byteswap16(COMMAND->From));
tmpStr[7] = (char)((COMMAND->OPCODE << 5) & 0xE0);
tmpStr[7] |= (char)((COMMAND->HI_LOW << 4) & 0x10);
tmpStr[7] |= (char)((COMMAND->OFFSet >> 4) & 0x0F);
tmpStr[8] = (char)((COMMAND->OFFSet << 4) & 0xF0);
tmpStr[8] |= (char)((COMMAND->CheckSUM >> 8) & 0x0F);
unsigned char byte = 0;
```

```
for(int i=1;i<9;i++) {
 byte = byte ^ (unsigned char)tmpStr[i];
 //cout<<"byte : "<<byte<<" | (unsigned char)ToSend[i]) : "<< (
(unsigned char)ToSend[i] ) <<endl;
}

(unsigned char)tmpStr[9] = byte;

printf("command: ");
for(i = 0; i < 10; i++) {
printf("%.02x ",tmpStr[i]);
}
printf("");
return tmpStr;
```

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsDatabase != NULL)
    {

        systemSetup* p_systemSetup = itsDatabase-
>getItsSystemSetup();
        if(p_systemSetup != NULL)
            itsDatabase->__setItsSystemSetup(NULL);
        itsDatabase = NULL;
    }
if(itsImmediate != NULL)
    {

        systemSetup* p_systemSetup = itsImmediate-
>getItsSystemSetup();
        if(p_systemSetup != NULL)
            itsImmediate->__setItsSystemSetup(NULL);
        itsImmediate = NULL;
    }
if(itsSerial_IO != NULL)
    {

        systemSetup* p_systemSetup = itsSerial_IO-
>getItsSystemSetup();
        if(p_systemSetup != NULL)
            itsSerial_IO->__setItsSystemSetup(NULL);
        itsSerial_IO = NULL;
    }
```

evFIND

Event

getCOMMAND

Generated , Primitive-operation , Public, Return type is COMMAND_STRUCTURE

Constant

Body

```
return COMMAND;
```

getEntries

Generated , Primitive-operation , Public, Return type is OMBoolean

Constant

Body

```
return entries;
```

getItsDatabase

Generated , Primitive-operation , Public, Return type is 'database*'

Constant

Body

```
return itsDatabase;
```

getItsImmediate

Generated , Primitive-operation , Public, Return type is 'Immediate*'

Constant

Body

```
return itsImmediate;
```

getItsSerial_IO

Generated , Primitive-operation , Public, Return type is 'Serial_IO*'

Constant

Body

```
return itsSerial_IO;
```

getTmpStr

Generated , Primitive-operation , Public, Return type is 'unsigned char'

Args:

'int' i1

Constant

Body

```
return tmpStr[i1];
```

setCOMMAND

Generated , Primitive-operation , Public, Return type is COMMAND_STRUCTURE

Args:

COMMAND_STRUCTURE p_COMMAND

Body

```
COMMAND = p_COMMAND;
```

setEntries

Generated , Primitive-operation , Public, Return type is OMBoolean

Args:

OMBoolean p_entries

Body

```
entries = p_entries;
```

setItsDatabase

Generated , Primitive-operation , Public, Return type is void

Args:

'database*' p_database

Body

```
if(p_database != NULL)
    p_database->_setItsSystemSetup(this);
_setItsDatabase(p_database);
```

setItsImmediate

Generated , Primitive-operation , Public, Return type is void

Args:

'Immediate*' p_Immediate

Body

```
if(p_Immediate != NULL)
    p_Immediate->_setItsSystemSetup(this);
_setItsImmediate(p_Immediate);
```

setItsSerial_IO

Generated , Primitive-operation , Public, Return type is void

Args:

'Serial_IO*' p_Serial_IO

Body

```
if(p_Serial_IO != NULL)
    p_Serial_IO->_setItsSystemSetup(this);
_setItsSerial_IO(p_Serial_IO);
```

setTmpStr

Generated , Primitive-operation , Public, Return type is 'void'

Args:

'int' i1

'unsigned char' p_tmpStr

Body

```
tmpStr[i1] = p_tmpStr;
```

startBehavior

Virtual, Generated , Primitive-operation , Public, Return type is OMBoolean

Body

```
OMBoolean done = FALSE;
done = OMReactive::startBehavior();
if(done)
    start();
return done;
```

systemSetup

Generated , Constructor , Public

~systemSetup

Generated , Destructor , Public

Body

```
cleanUpRelations();
```

Attributes:

COMMAND

Type of COMMAND_STRUCTURE, Public

entries

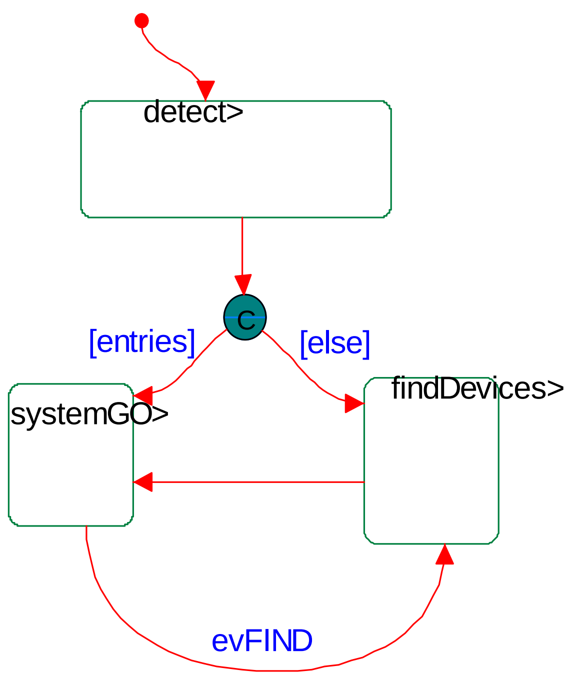
true = wall unit entries in the database
false = no entries

Type of OMBoolean, Public, Initial Value: true

tmpStr

Type of 'unsigned char %s[12]', Public

Statechart



ROOT

Or-state

Substates:

detect
findDevices
systemGO

Default Transition

Target:

detect

detect

check to see if there are any wall unit entries

Or-state

EntryAction

```
CRecordset rs( itsDatabase->getDB());
if( itsDatabase->isConnected()) {
    itsDatabase->query(&rs, "SELECT ID FROM Wall_Unit");
    if (rs.IsEOF()) {
            entries = false;
    }
    else {
            entries = true;
    }
}
```

Out Transition

Condition Connector

Branches:

```
[entries]
```

Target:

systemGO

```
[else]
```

Target:

findDevices

findDevices

used to find wall units. searches every possible combination of 8 bit address and then combines the replies to figure out the real address of the wall units

Or-state

EntryAction

```
unsigned char * ToSend;
unsigned char response[128];
char tableName[10];
char exeStr[50];
CString toSend;
int re_do = 0;
int ugly=0;

CRecordset rs( itsDatabase->getDB());
CRecordset rs1 (itsDatabase->getDB());
if( itsDatabase->isConnected()) {
CDBVariant temp, temp1;

    itsDatabase->setOKtoStart(false);  //haults any other threads from
continueing
    itsDatabase->execute("INSERT INTO WebSiteInfo VALUES
('Searching')");

    do {

    for(int delTable = 0; delTable <= 6; delTable++) {
            (void)sprintf(exeStr,"Delete * from table%d",delTable);
            itsDatabase->execute(exeStr);
    }
    //cout<<"now tables deleted"<<endl;


    for (int bytes = 0; bytes < 4; bytes++) {
            //this loop will go through each of the 4 bytes in the
addresses
            //cout<<"0-4 for loop"<<endl;
            //create table in db
```

```
            //assign name to variable tableName
            (void)sprintf(tableName,"table%d",bytes);
            //cout<<"table name :"<<tableName<<endl;
            (void)sprintf(exeStr,"Create Table %s ( addrByte int
)",tableName);
            toSend = exeStr;
            //itsDatabase->execute(toSend);

            for(int bits = 0; bits < 256; bits++) {

                            if(bits == 12) {
                                    break;
                            }

                    //this loop goes through each possible pattern in
the byte
                    //cout<<"0-256 for loop"<<endl;
                    //create a todo that has the search command with the
data of bytes and bits

                    COMMAND.StartSymbol = 0xDB;
                    COMMAND.To = (unsigned char) bits << ( (bytes*8)
&~(0xFF << (bytes*8)) ); // THis wont work need to talk to strath about
it
                    //cout<<"the to: "<<COMMAND.To<<endl;
                    COMMAND.From = 0x001;
                    COMMAND.OPCODE = 0x7;
                    COMMAND.HI_LOW = 0x1;
                    COMMAND.OFFSet = (unsigned char) bytes;
                    COMMAND.CheckSUM = 0x101;
                    ToSend = BuildCommand(&COMMAND);
                    //cout<<"passed build command"<<endl;
                    ugly = 0;
                    if(itsSerial_IO->WRITE(ToSend)) { //we sent out the
find command lets see if there are any out there
                            //cout<<"bout to read"<<endl;
                            if (itsSerial_IO->READ(response)) {
                                    //got something back which means
there is a device out there with that part of address
                                    //check to make sure not garbage or
should we????  I think we should just assume
                                    (void)sprintf(exeStr,"Insert into %s
values (%d)",tableName,bits);
                                    toSend = exeStr;
                                    cout<<"got something back :
"<<toSend<<endl;
                                    ugly = 1;
                                    //itsDatabase->execute(exeStr);

                            }
                            cout<<"im done reading"<<endl;
                    }
                    if(itsSerial_IO->WRITE(ToSend)) { //we sent out the
find command lets see if there are any out there
                            //cout<<"bout to read"<<endl;
                            if (itsSerial_IO->READ(response)) {
                                    //got something back which means
there is a device out there with that part of address
                                    //check to make sure not garbage or
should we????  I think we should just assume
                                    (void)sprintf(exeStr,"Insert into %s
values (%d)",tableName,bits);
                                    toSend = exeStr;
                                    //cout<<"got something back :
"<<toSend<<endl;
                                    ugly = 1;
                                    //itsDatabase->execute(exeStr);

                            }
                            //cout<<"im done reading"<<endl;
                    }
```

```
                    if (ugly == 1 ) {
                            itsDatabase->execute(exeStr);
                    }
                    ugly = 0;

            }

            (void)sprintf(exeStr,"SELECT COUNT(addrByte) FROM
table%d",bytes);
            //cout<<exeStr<<endl;
            toSend = exeStr;
            itsDatabase->query(&rs,exeStr);
            //cout<<"its not what i think it is"<<endl;
            if(!rs.IsEOF()) {
                    rs.GetFieldValue(short(0),temp);
/*                  switch(temp.m_dwType) {
                            case DBVT_NULL : cout<<"Null"<<endl; break;
                            case DBVT_BOOL : cout<<"Bool"<<endl; break;
                            case DBVT_UCHAR : cout<<"Char"<<endl; break;
                            case DBVT_SHORT : cout<<"Short"<<endl;
break;
                            case DBVT_LONG : cout<<"Long"<<endl; break;
                            case DBVT_SINGLE : cout<<"Single"<<endl;
break;
                            case DBVT_DOUBLE : cout<<"double"<<endl;
break;
                            case DBVT_DATE : cout<<"date"<<endl; break;
                            case DBVT_STRING : cout<<"string"<<endl;
break;
                            case DBVT_BINARY : cout<<"binary"<<endl;
break;
                            default : cout<<"fuck"<<endl; break;
                    }*/ //code to determine what the type of the databse
variable really is
             //cout<<"is it"<<endl;
                    if(temp.m_lVal <= 0) {
                            bytes--;                              //this
means that there were no devices found so we had an error so re-check
                            (void)sprintf(exeStr,"Drop Table
table%d",bytes+1);
                            //cout<<exeStr<<endl;
                            //itsDatabase->execute(exeStr);
                    }
                    //cout<<"yeah its dying where i think it is"<<endl;

            }
            //cout<<"i bet i know what it is"<<endl;
            rs.Close();
    }
            // next set of shit to do for putting these things together
    do {
            //cout<<"starting do while for table 4"<<endl;
            itsDatabase->query(&rs,"SELECT addrByte FROM table0");
            while(!rs.IsEOF()) {
                    //cout<<"first table check"<<endl;
                    rs.GetFieldValue(short(0), temp);
                    itsDatabase->query(&rs1,"SELECT addrByte FROM
table1");
                    while(!rs1.IsEOF()) {
                            //cout<<"second table check"<<endl;
                            COMMAND.StartSymbol = 0xDB;
                            rs1.GetFieldValue(short(0), temp1);
                            //cout<<"got second value"<<endl;
                            COMMAND.To = (unsigned long) ( temp.m_iVal |
temp1.m_iVal << 8) ; // THis wont work need to talk to strath about it
                            //cout<<"the to: "<<COMMAND.To<<endl;
                            COMMAND.From = 0x001;
                            COMMAND.OPCODE = 0x7;
                            COMMAND.HI_LOW = 0x1;
```

```
                          COMMAND.OFFSet = 0x10;
                          COMMAND.CheckSUM = 0x101;
                          ToSend = BuildCommand(&COMMAND);
                          ugly = 0;
                          if(itsSerial_IO->WRITE(ToSend)) { //we sent
out the find command lets see if there are any out there
                        //cout<<"wrote it"<<endl;
                                 if (itsSerial_IO->READ(response)) {
                                        //got something back which
means there is a device out there with that part of address
                                        //check to make sure not
garbage or should we????  I think we should just assume
                                        (void)sprintf(exeStr,"Insert
into table4 values (%d)",( temp.m_iVal | temp1.m_iVal << 8));
                                        ugly = 1;
                                 }
                }
                          if(itsSerial_IO->WRITE(ToSend)) { //we sent
out the find command lets see if there are any out there
                        //cout<<"wrote it"<<endl;
                                 if (itsSerial_IO->READ(response)) {
                                        //got something back which
means there is a device out there with that part of address
                                        //check to make sure not
garbage or should we????  I think we should just assume
                                        (void)sprintf(exeStr,"Insert
into table4 values (%d)",( temp.m_iVal | temp1.m_iVal << 8));
                                        ugly = 1;
                                 }
                }
                if(ugly ==1) {
                   itsDatabase->execute(exeStr);
                          }
                          ugly = 0;
                rs1.MoveNext();

                   }
                   rs1.Close();
                   rs.MoveNext();
           }
           rs.Close();
           itsDatabase->query(&rs,"SELECT COUNT(addrWord) FROM
table4");
           if(!rs.IsEOF()) {
                   rs.GetFieldValue(short(0),temp);
           }
        rs.Close();
        //cout<<"at the end of do while"<<endl;
    }while(temp.m_lVal <=0);

    //cout<<"do we get here?"<<endl;

    do {

           itsDatabase->query(&rs,"SELECT addrWord FROM table4");
           while(!rs.IsEOF()) {
                   //cout<<"first table check"<<endl;
                   rs.GetFieldValue(short(0), temp);
                   itsDatabase->query(&rs1,"SELECT addrByte FROM
table2");
                   while(!rs1.IsEOF()) {
                          //cout<<"second table check"<<endl;
                          COMMAND.StartSymbol = 0xDB;
                          rs1.GetFieldValue(short(0), temp1);
                          //cout<<"got second value"<<endl;
                          COMMAND.To = (unsigned long) ( temp.m_lVal |
temp1.m_iVal << 16) ; // THis wont work need to talk to strath about it
                          //cout<<"the to: "<<COMMAND.To<<endl;
                          COMMAND.From = 0x001;
                          COMMAND.OPCODE = 0x7;
                          COMMAND.HI_LOW = 0x1;
```

```
                    COMMAND.OFFSet = 0x20;
                    COMMAND.CheckSUM = 0x101;
                    ToSend = BuildCommand(&COMMAND);
                    ugly = 0;
                    if(itsSerial_IO->WRITE(ToSend)) { //we sent
out the find command lets see if there are any out there
                  //cout<<"wrote it"<<endl;
                         if (itsSerial_IO->READ(response)) {
                              //got something back which
means there is a device out there with that part of address
                              //check to make sure not
garbage or should we????  I think we should just assume
                              (void)sprintf(exeStr,"Insert
into table5 values (%d)",( temp.m_lVal | temp1.m_iVal << 16));
                              ugly = 1;
                         }
                }
                    if(itsSerial_IO->WRITE(ToSend)) { //we sent
out the find command lets see if there are any out there
                  //cout<<"wrote it"<<endl;
                         if (itsSerial_IO->READ(response)) {
                              //got something back which
means there is a device out there with that part of address
                              //check to make sure not
garbage or should we????  I think we should just assume
                              (void)sprintf(exeStr,"Insert
into table5 values (%d)",( temp.m_lVal | temp1.m_iVal << 16));
                              ugly = 1;
                         }
                }

                    if(ugly == 1) {
                         itsDatabase->execute(exeStr);
                    }
                 ugly = 0;

               rs1.MoveNext();
                 }
                 rs1.Close();
                 rs.MoveNext();
            }
            rs.Close();
            itsDatabase->query(&rs,"SELECT COUNT(addrWord_byte) FROM
table5");
            if(!rs.IsEOF()) {
                 rs.GetFieldValue(short(0),temp);
            }
        rs.Close();
        //cout<<"at the end of do while"<<endl;
    }while(temp.m_lVal <=0);



    do {

            itsDatabase->query(&rs,"SELECT addrWord_byte FROM table5");
            while(!rs.IsEOF()) {
                 //cout<<"first table check"<<endl;
                 rs.GetFieldValue(short(0), temp);
                 itsDatabase->query(&rs1,"SELECT addrByte FROM
table3");
                 while(!rs1.IsEOF()) {
                      //cout<<"second table check"<<endl;
                      COMMAND.StartSymbol = 0xDB;
                      rs1.GetFieldValue(short(0), temp1);
                      //cout<<"got second value"<<endl;
                      COMMAND.To = (unsigned long) ( temp.m_lVal |
temp1.m_iVal << 24) ; // THis wont work need to talk to strath about it
                      //cout<<"the to: "<<COMMAND.To<<endl;
                      COMMAND.From = 0x001;
                      COMMAND.OPCODE = 0x7;
```

```
                          COMMAND.HI_LOW = 0x1;
                          COMMAND.OFFSet = 0x30;
                          COMMAND.CheckSUM = 0x101;
                          ToSend = BuildCommand(&COMMAND);
                          ugly = 0;
                          if(itsSerial_IO->WRITE(ToSend)) { //we sent
out the find command lets see if there are any out there
                                  if (itsSerial_IO->READ(response)) {
                                          //got something back which
means there is a device out there with that part of address
                                          //check to make sure not
garbage or should we????  I think we should just assume
                                          (void)sprintf(exeStr,"Insert
into table6 values (%d)",( temp.m_lVal | temp1.m_iVal << 24));
                                          ugly = 1;
                                  }
                }
                          if(itsSerial_IO->WRITE(ToSend)) { //we sent
out the find command lets see if there are any out there
                                  if (itsSerial_IO->READ(response)) {
                                          //got something back which
means there is a device out there with that part of address
                                          //check to make sure not
garbage or should we????  I think we should just assume
                                          (void)sprintf(exeStr,"Insert
into table6 values (%d)",( temp.m_lVal | temp1.m_iVal << 24));
                                          ugly = 1;
                                  }
                }

                if(ugly = 1) {
                   itsDatabase->execute(exeStr);
                }
                ugly = 0;

                rs1.MoveNext();
                   }
                   rs1.Close();
                   rs.MoveNext();
           }
           rs.Close();

                   itsDatabase->query(&rs,"SELECT COUNT(addrDWord) FROM
table6");
           if(!rs.IsEOF()) {
                   rs.GetFieldValue(short(0),temp);
           }
        rs.Close();
        //cout<<"at the end of do while"<<endl;
    }while(temp.m_lVal <=0);

    itsDatabase->query(&rs,"SELECT addrDWord FROM table6");
    while(!rs.IsEOF()) {

           COMMAND.StartSymbol = 0xDB;
           rs.GetFieldValue(short(0), temp);
           //cout<<"got second value"<<endl;
           COMMAND.To = (unsigned long)temp.m_lVal; // THis wont work
need to talk to strath about it
           //cout<<"the to: "<<COMMAND.To<<endl;
           COMMAND.From = 0x001;
           COMMAND.OPCODE = 0x7;
           COMMAND.HI_LOW = 0x1;
           COMMAND.OFFSet = 0x40;
           COMMAND.CheckSUM = 0x101;
           ToSend = BuildCommand(&COMMAND);
           ugly = 0;
           if(itsSerial_IO->WRITE(ToSend)) { //we sent out the find
command lets see if there are any out there
            //cout<<"wrote it"<<endl;
                   if (itsSerial_IO->READ(response)) {
```

```
                        ugly = 1;
                }
        }
        if(itsSerial_IO->WRITE(ToSend)) { //we sent out the find
command lets see if there are any out there
         //cout<<"wrote it"<<endl;
                if (itsSerial_IO->READ(response)) {
                        ugly = 1;
                }
        }

        if(ugly = 1) {
                rs.MoveNext();
        }
        ugly = 0;

    }
    rs.Close();
    //cout<<"done checking sending the fucked up restart check"<<endl;

    COMMAND.StartSymbol = 0xDB;
    COMMAND.To = (unsigned long)0x00000000; // THis wont work need to
talk to strath about it
    //cout<<"the to: "<<COMMAND.To<<endl;
    COMMAND.From = 0x001;
    COMMAND.OPCODE = 0x7;
    COMMAND.HI_LOW = 0x1;
    COMMAND.OFFSet = 0x50;
    COMMAND.CheckSUM = 0x101;
    ToSend = BuildCommand(&COMMAND);
    if(itsSerial_IO->WRITE(ToSend)) { //we sent out the find command
lets see if there are any out there
        //cout<<"wrote it"<<endl;
        if (itsSerial_IO->READ(response)) {
                re_do = 1;
        }
    }

    if(itsSerial_IO->WRITE(ToSend)) { //we sent out the find command
lets see if there are any out there
       // cout<<"wrote it"<<endl;
        if (itsSerial_IO->READ(response)) {
                re_do = 1;
        }
    }


    }while(re_do == 1);
    re_do=0;

    itsDatabase->query(&rs,"SELECT addrDWord FROM table6");
    while(!rs.IsEOF()) {
        rs.GetFieldValue(short(0),temp);
        itsDatabase->query(&rs1,"SELECT address FROM Wall_Unit");
        while(!rs1.IsEOF()) {
                rs1.GetFieldValue(short(0),temp1);
                if(temp.m_lVal == temp1.m_lVal) {
                        (void)sprintf(exeStr,"Delete addrDWord FROM
table6 WHERE addrDWord = %d",temp.m_lVal);
                        itsDatabase->execute(exeStr);
                }
                else {
                }
                rs1.MoveNext();
        }
        rs1.Close();
        rs.MoveNext();
    }
    rs.Close();

    itsDatabase->query(&rs,"SELECT addrDWord FROM table6");
```

```
    while(!rs.IsEOF()) {
            rs.GetFieldValue(short(0),temp);
            (void)sprintf(exeStr,"INSERT into Wall_Unit (address)
Values( %d )",temp.m_lVal);
            itsDatabase->execute(exeStr);
            rs.MoveNext();
    }
    rs.Close();

    //here goes the last part where we insert into wall_unit



    for(int delTable = 0; delTable <= 6; delTable++) {
            (void)sprintf(exeStr,"Delete * from table%d",delTable);
            toSend = exeStr;
            itsDatabase->execute(exeStr);
    }
    //cout<<"now tables deleted"<<endl;

    itsDatabase->execute("DELETE * FROM WebSiteInfo");
    itsDatabase->execute("INSERT INTO WebSiteInfo VALUES ('done')");

    itsDatabase->setOKtoStart(true);
}
```

Out Transition

Target:

systemGO

systemGO

start system and wait

Or-state

EntryAction

```
itsDatabase->GEN(evSetup);
```

Out Transition

```
evFIND
```

Target:

findDevices

ToDo

Relations:

itsTODO_Handler

Association with TODO_Handler, Multiplicity of 1, Bi-directional

Operations:

__setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
itsTODO_Handler = p_TODO_Handler;
```

_clearItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Body

```
itsTODO_Handler = NULL;
```

_setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(itsTODO_Handler != NULL)
    itsTODO_Handler->_removeItsToDo(this);
__setItsTODO_Handler(p_TODO_Handler);
```

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsTODO_Handler != NULL)
    {

        TODO_Handler* current = itsTODO_Handler;
        if(current != NULL)
            current->_removeItsToDo(this);
        itsTODO_Handler = NULL;
    }
```

getAddress

Generated , Primitive-operation , Public, Return type is long

Constant

Body

```
return Address;
```

getAttempts

Generated , Primitive-operation , Public, Return type is int

Constant

Body

```
return Attempts;
```

getHILOW

Generated , Primitive-operation , Public, Return type is 'unsigned char '

Constant

Body

```
return HILOW;
```

getItsTODO_Handler

Generated , Primitive-operation , Public, Return type is 'TODO_Handler*'

Constant

Body

```
return itsTODO_Handler;
```

getOffset

Generated , Primitive-operation , Public, Return type is int

Constant

Body

```
return Offset;
```

getOPCode

Generated , Primitive-operation , Public, Return type is unsigned char

Constant

Body

```
return OPCode;
```

getPassWord

Generated , Primitive-operation , Public, Return type is unsigned char

Constant

Body

```
return PassWord;
```

getWhatToPoll

Generated , Primitive-operation , Public, Return type is int

Constant

Body

```
return WhatToPoll;
```

setAddress

Generated , Primitive-operation , Public, Return type is long

Args:

long p_Address

Body

```
Address = p_Address;
```

setAttempts

Generated , Primitive-operation , Public, Return type is int

Args:

int p_Attempts

Body

```
Attempts = p_Attempts;
```

setHILOW

Generated , Primitive-operation , Public, Return type is 'void'

Args:

'unsigned char %s' p_HILOW

Body

```
HILOW = p_HILOW;
```

setItsTODO_Handler

Generated , Primitive-operation , Public, Return type is void

Args:

'TODO_Handler*' p_TODO_Handler

Body

```
if(p_TODO_Handler != NULL)
    p_TODO_Handler->_addItsToDo(this);
_setItsTODO_Handler(p_TODO_Handler);
```

setOffset

Generated , Primitive-operation , Public, Return type is int

Args:

int p_Offset

Body

```
Offset = p_Offset;
```

setOPCode

Generated , Primitive-operation , Public, Return type is unsigned char

Args:

unsigned char p_OPCode

Body

```
OPCode = p_OPCode;
```

setPassWord

Generated , Primitive-operation , Public, Return type is unsigned char

Args:

unsigned char p_PassWord

Body

```
PassWord = p_PassWord;
```

setWhatToPoll

Generated , Primitive-operation , Public, Return type is int

Args:

int p_WhatToPoll

Body

```
WhatToPoll = p_WhatToPoll;
```

ToDo

Generated , Constructor , Public

ToDo

Overridden Properties

Subjects:
CPP_CG
Metaclasses:
Operation
Properties:
Kind: common
Inline: none
Constructor , Public
Args:
long address
unsigned char PW
unsigned char OPCODE
int attempts = 0
number of tries
int offset = 0
'unsigned char %s' hilow = 1
Body
```
Address = address;
OPCode = OPCODE;
PassWord = PW;
Attempts = attempts;
Offset = offset;
HILOW = hilow;
WhatToPoll = 5;
```

ToDo
Overridden Properties
Subjects:
CPP_CG
Metaclasses:
Operation
Properties:
Kind: common
Inline: none
Constructor , Public
Args:
long address
unsigned char PW
unsigned char OPCODE
int attempts
int offset
unsigned char hilow
int whatToPoll
Body
```
Address = address;
OPCode = OPCODE;
PassWord = PW;
Attempts = attempts;
Offset = offset;
HILOW = hilow;
WhatToPoll = whatToPoll;
```

~ToDo
Generated , Destructor , Public
Body
```
cleanUpRelations();
```

Attributes:
Address
Type of long, Public

Attempts

num tries

Type of int, Public

HILOW

Type of 'unsigned char %s', Public

Offset

Type of int, Public

OPCode

Type of unsigned char, Public

PassWord

Type of unsigned char, Public

WhatToPoll

Type of int, Public

TODO_Handler

Overridden Properties

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: sequential

Relations:

itsCheckES

Association with checkES, Multiplicity of 1, Bi-directional

itsPoll

Association with Poll, Multiplicity of 1, Bi-directional

itsImmediate

Association with Immediate, Multiplicity of 1, Bi-directional

itsCheckTODO

Association with checkTODO, Multiplicity of 1, Bi-directional

itsToDo

Association with ToDo, Multiplicity of *, Bi-directional

itsResponse

Association with response, Multiplicity of 1, Bi-directional

Operations:

__setItsCheckES

Generated , Primitive-operation , Public, Return type is void

Args:

'checkES*' p_checkES

Body

```
itsCheckES = p_checkES;
```

__setItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Args:

'checkTODO*' p_checkTODO

Body

```
itsCheckTODO = p_checkTODO;
```

__setItsImmediate

Generated , Primitive-operation , Public, Return type is void

Args:

'Immediate*' p_Immediate

Body

```
itsImmediate = p_Immediate;
```

__setItsPoll

Generated , Primitive-operation , Public, Return type is void
Args:
'Poll*' p_Poll
Body
```
itsPoll = p_Poll;
```
__setItsResponse
Generated , Primitive-operation , Public, Return type is void
Args:
'response*' p_response
Body
```
itsResponse = p_response;
```
_addItsToDo
Generated , Primitive-operation , Public, Return type is void
Args:
'ToDo*' p_ToDo
Body
```
itsToDo.add(p_ToDo);
```
_clearItsCheckES
Generated , Primitive-operation , Public, Return type is void
Body
```
itsCheckES = NULL;
```
_clearItsCheckTODO
Generated , Primitive-operation , Public, Return type is void
Body
```
itsCheckTODO = NULL;
```
_clearItsImmediate
Generated , Primitive-operation , Public, Return type is void
Body
```
itsImmediate = NULL;
```
_clearItsPoll
Generated , Primitive-operation , Public, Return type is void
Body
```
itsPoll = NULL;
```
_clearItsResponse
Generated , Primitive-operation , Public, Return type is void
Body
```
itsResponse = NULL;
```
_clearItsToDo
Generated , Primitive-operation , Public, Return type is void
Body
```
itsToDo.removeAll();
```
_removeItsToDo
Generated , Primitive-operation , Public, Return type is void
Args:
'ToDo*' p_ToDo
Body
```
itsToDo.remove(p_ToDo);
```
_setItsCheckES
Generated , Primitive-operation , Public, Return type is void
Args:
'checkES*' p_checkES

Body

```
if(itsCheckES != NULL)
    itsCheckES->__setItsTODO_Handler(NULL);
__setItsCheckES(p_checkES);
```

_setItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Args:

'checkTODO*' p_checkTODO

Body

```
if(itsCheckTODO != NULL)
    itsCheckTODO->__setItsTODO_Handler(NULL);
__setItsCheckTODO(p_checkTODO);
```

_setItsImmediate

Generated , Primitive-operation , Public, Return type is void

Args:

'Immediate*' p_Immediate

Body

```
if(itsImmediate != NULL)
    itsImmediate->__setItsTODO_Handler(NULL);
__setItsImmediate(p_Immediate);
```

_setItsPoll

Generated , Primitive-operation , Public, Return type is void

Args:

'Poll*' p_Poll

Body

```
if(itsPoll != NULL)
    itsPoll->__setItsTODO_Handler(NULL);
__setItsPoll(p_Poll);
```

_setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
if(itsResponse != NULL)
    itsResponse->__setItsTODO_Handler(NULL);
__setItsResponse(p_response);
```

addItsToDo

Generated , Primitive-operation , Public, Return type is void

Args:

'ToDo*' p_ToDo

Body

```
if(p_ToDo != NULL)
    p_ToDo->_setItsTODO_Handler(this);
_addItsToDo(p_ToDo);
```

cleanUpRelations

Generated , Primitive-operation , Protected, Return type is void

Body

```
if(itsCheckES != NULL)
    {

        TODO_Handler* p_TODO_Handler = itsCheckES-
>getItsTODO_Handler();
        if(p_TODO_Handler != NULL)
            itsCheckES->__setItsTODO_Handler(NULL);
        itsCheckES = NULL;
    }
if(itsCheckTODO != NULL)
    {
```

```
        TODO_Handler* p_TODO_Handler = itsCheckTODO-
>getItsTODO_Handler();
        if(p_TODO_Handler != NULL)
            itsCheckTODO->__setItsTODO_Handler(NULL);
        itsCheckTODO = NULL;
    }
if(itsImmediate != NULL)
    {

        TODO_Handler* p_TODO_Handler = itsImmediate-
>getItsTODO_Handler();
        if(p_TODO_Handler != NULL)
            itsImmediate->__setItsTODO_Handler(NULL);
        itsImmediate = NULL;
    }
if(itsPoll != NULL)
    {

        TODO_Handler* p_TODO_Handler = itsPoll-
>getItsTODO_Handler();
        if(p_TODO_Handler != NULL)
            itsPoll->__setItsTODO_Handler(NULL);
        itsPoll = NULL;
    }
if(itsResponse != NULL)
    {

        TODO_Handler* p_TODO_Handler = itsResponse-
>getItsTODO_Handler();
        if(p_TODO_Handler != NULL)
            itsResponse->__setItsTODO_Handler(NULL);
        itsResponse = NULL;
    }
{
    OMIterator<ToDo*> iter(itsToDo);iter.reset();
    while (*iter){
        TODO_Handler* p_TODO_Handler = (*iter)-
>getItsTODO_Handler();
        if(p_TODO_Handler != NULL)
            (*iter)->__setItsTODO_Handler(NULL);
        iter++;
    }
    itsToDo.removeAll();
}
```

clearItsToDo

Generated , Primitive-operation , Public, Return type is void

Body

```
OMIterator<ToDo*> iter(itsToDo);iter.reset();
while (*iter){
    (*iter)->_clearItsTODO_Handler();
    iter++;
}
_clearItsToDo();
```

getAttributes

Generated , Primitive-operation , Public, Return type is todoattributes Constant

Body

```
return attributes;
```

getItsCheckES

Generated , Primitive-operation , Public, Return type is 'checkES*' Constant

Body

```
return itsCheckES;
```

getItsCheckTODO

Generated , Primitive-operation , Public, Return type is 'checkTODO*'

Constant
Body
```
return itsCheckTODO;
```

getItsImmediate
Generated , Primitive-operation , Public, Return type is 'Immediate*'
Constant
Body
```
return itsImmediate;
```

getItsPoll
Generated , Primitive-operation , Public, Return type is 'Poll*'
Constant
Body
```
return itsPoll;
```

getItsResponse
Generated , Primitive-operation , Public, Return type is 'response*'
Constant
Body
```
return itsResponse;
```

getItsToDo
Generated , Primitive-operation , Public, Return type is 'OMIterator<ToDo*>'
Constant
Body
```
OMIterator<ToDo*> iter(itsToDo);iter.reset();
return iter;
```

getNextTODO
gets the next set of attributes from the todo's
Overridden Properties
Subjects:
CPP_CG
Metaclasses:
Operation
Properties:
Kind: common
Inline: none
Primitive-operation , Public, Return type is todoattributes
Body
```
OMIterator<ToDo*> todoIter(itsToDo);
//todoIter.reset();
if(*todoIter) {
attributes.address = (*todoIter)->getAddress();
attributes.PW = (*todoIter)->getPassWord();
attributes.OPCODE = (*todoIter)->getOPCode();
attributes.attempts = (*todoIter)->getAttempts();
attributes.offset = (*todoIter)->getOffset();
attributes.HI_LOW = (*todoIter)->getHILOW();
attributes.whatToPoll = (*todoIter)->getWhatToPoll();
//remove this todo
removeItsToDo(*todoIter);
TODOcount--;
cout<<"TODO count decresed to : "<<TODOcount<<endl;
}
else {
//this means no todos
todoIter.reset();

attributes.address = NULL;
attributes.PW = NULL;
attributes.OPCODE = NULL;
attributes.offset = NULL;
attributes.HI_LOW = NULL;
```

```
attributes.whatToPoll = NULL;
}
return attributes;
```

getTODOcount

Generated , Primitive-operation , Public, Return type is int

Constant

Body

```
return TODOcount;
```

getTodoIter

Generated , Primitive-operation , Public, Return type is 'ToDo* '

Constant

Body

```
return todoIter;
```

NewTODO

Overridden Properties

Subjects:

CPP_CG

Metaclasses:

Operation

Properties:

Kind: common

Inline: none

Primitive-operation , Public, Return type is void

Args:

long address

unsigned char PW

unsigned char OPCODE

int count = 0

number of tried to complete this todo

int offset = 0

'unsigned char %s' hilow = 1

int whatToPoll = 5

Body

```
//create a todo here with
//the three attributes
cout<<"handler what to poll: "<<whatToPoll<<endl;
addItsToDo(new
ToDo(address,PW,OPCODE,count,offset,hilow,whatToPoll));
TODOcount++;
cout<<"TODO count increased to : "<<TODOcount<<endl;
//create event in checkTODO
itsCheckTODO->GEN(evDoToDo);
```

removeItsToDo

Generated , Primitive-operation , Public, Return type is void

Args:

'ToDo*' p_ToDo

Body

```
if(p_ToDo != NULL)
    p_ToDo->__setItsTODO_Handler(NULL);
_removeItsToDo(p_ToDo);
```

setAttributes

Generated , Primitive-operation , Public, Return type is todoattributes

Args:

todoattributes p_attributes

Body

```
attributes = p_attributes;
```

setItsCheckES

Generated , Primitive-operation , Public, Return type is void

Args:

'checkES*' p_checkES

Body

```
if(p_checkES != NULL)
    p_checkES->_setItsTODO_Handler(this);
_setItsCheckES(p_checkES);
```

setItsCheckTODO

Generated , Primitive-operation , Public, Return type is void

Args:

'checkTODO*' p_checkTODO

Body

```
if(p_checkTODO != NULL)
    p_checkTODO->_setItsTODO_Handler(this);
_setItsCheckTODO(p_checkTODO);
```

setItsImmediate

Generated , Primitive-operation , Public, Return type is void

Args:

'Immediate*' p_Immediate

Body

```
if(p_Immediate != NULL)
    p_Immediate->_setItsTODO_Handler(this);
_setItsImmediate(p_Immediate);
```

setItsPoll

Generated , Primitive-operation , Public, Return type is void

Args:

'Poll*' p_Poll

Body

```
if(p_Poll != NULL)
    p_Poll->_setItsTODO_Handler(this);
_setItsPoll(p_Poll);
```

setItsResponse

Generated , Primitive-operation , Public, Return type is void

Args:

'response*' p_response

Body

```
if(p_response != NULL)
    p_response->_setItsTODO_Handler(this);
_setItsResponse(p_response);
```

setTODOcount

Generated , Primitive-operation , Public, Return type is int

Args:

int p_TODOcount

Body

```
TODOcount = p_TODOcount;
```

setTodoIter

Generated , Primitive-operation , Public, Return type is 'void'

Args:

'ToDo* %s' p_todoIter

Body

```
todoIter = p_todoIter;
```

TODO_Handler

Generated , Constructor , Public

~TODO_Handler

Generated , Destructor , Public

Body

```
cleanUpRelations();
```

Attributes:

attributes

temp variable to return todo attributes when called

Type of todoattributes, Public

TODOcount

the number of current todos in the queue

Type of int, Public, Initial Value: 0

todoIter

Type of 'ToDo* %s;', Public

ACTORS:

User

The user in this case is composed of both a home computer which provides a display device for the user interface, as well as input devices such as a mouse and a keyboard for opperator input. The opperator is responsible for interacting with the user interface and inputing sequences events and viewing results.

Relations:

itsEvents and sequences

The user creates and edits events for the system.

creates and edits

Association with events and sequences, Multiplicity of 1, Bi-directional

itsUser interface

The user interacts with the user interface, allowing him/her to perform functional tasks such as creating events, viewing power data, and entering regulation information.

interacts with

Association with user interface, Multiplicity of 1, Bi-directional

Operations:

User

Generated , Constructor , Public

~User

Generated , Destructor , Public

Wall_Unit

The wall units consist of devices that are installed into wall sockets, having the capability of collecting data on voltage and current and tramitting this information back to the system via the network link upon request. Additionally, these units are responsable for recieving commands through the network conduit and executing the commands.

Relations:

itsEvents and sequences

The events command the wall unit to perform a specified task and the wall unit must respond by executing that task.

commands

Association with events and sequences, Multiplicity of 1, Bi-directional

itsSystem

The system send out the command for the wall unit through the network link, which is in turn executed by the wall unit.

commands

Association with system, Multiplicity of 1, Bi-directional

itsNetwork link

The network link is the conduit by which data and commands are passed between the wall units and the control-box system.

collects data

Association with network link, Multiplicity of 1, Bi-directional

Operations:

Wall_Unit

Generated , Constructor , Public

~Wall_Unit

Generated , Destructor , Public

USE CASES:

---

events and sequences

Events and sequences are tasks to be scheduled by the control box for execution through the system. Events are individual actions, where as sequences are a group of actions. Events and sequences must either be stored for later use or scheduled for emidiate execution. A sequence may include turn off wall unit 1, 2 , 4 and turn on unit 3 at 8:00am every day. This event would be created by the user, through the UI and would then be sent to the system for scheduling. Events can come be labled as single occurance, recurring (daily, weekly, monthly), or as part of a sequence.

Relations:

itsUser

itsWall_Unit

SubUseCases:

system

---

network link

The network link provides a connection between the wall units collecting voltage and current data for computation and allows for the data to be requested and received for system use.

Relations:

itsWall_Unit

SubUseCases:

system

---

system

The system is responsible for recieving all event and sequence information and allocating time for them with some form of scheduling algorithm. Also, the system must request and store data from the network link to be computed and stored for use in power charts and power regulation. Power charts will be displayed on the user interface, and power regulation of outlets can be user defined or based on user parameters, such as limit power on outlet one or restrict power to outlet four.

Relations:

itsWall_Unit

SuperUseCases:

network link

Public

Stereotype: requests data

events and sequences

Public

Stereotype: add/delete and schedule

SubUseCases:

user interface

user interface

Display user interface and allow for user interaction with system for power regulation, event scheduling.

<u>Relations:</u>

itsUser

<u>SuperUseCases:</u>

system

Public

Stereotype: display

COMPONENTS

DefaultComponent

COMPONENT SETTINGS:

Build type: Executable

CONFIGURATIONS:

DefaultConfig

Overridden Properties

Subjects:

WebComponents

Metaclasses:

WebFramework

Properties:

GenerateInstrumentationCode: False

Scope type: Explicit
Instrumentation type: None
Time-model type: Real-time
Statechart generation type: Flat
Standard headers: afxdb.h,afxwin.h,afx.h,conio.h,stdio.h,string.h,iostream.h,iomanip.h
Include path: "C:\Program Files\Microsoft Visual Studio\VC98\MFC\Include"

FILES AND FOLDERS:

Senior Design:

Appendix D

```
//-----------------------------------------------------------------------------
// wallunit.c
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <c8051f300.h>                    // SFR declarations

//-----------------------------------------------------------------------------
// SFR Definitions for 'F30x
//-----------------------------------------------------------------------------

sfr16 DP       = 0x82;                    // data pointer
sfr16 TMR2RL   = 0xca;                    // Timer2 reload value
sfr16 TMR2     = 0xcc;                    // Timer2 counter
sfr16 PCA0CP1  = 0xe9;                    // PCA0 Module 1 Capture/Compare
sfr16 PCA0CP2  = 0xeb;                    // PCA0 Module 2 Capture/Compare
sfr16 PCA0     = 0xf9;                    // PCA0 counter
sfr16 PCA0CP0  = 0xfb;                    // PCA0 Module 0 Capture/Compare
sfr   EIP1     = 0xF6;                    // EXTENDED INTERRUPT PRIORITY

//-----------------------------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------------------------

#define SYSCLK                      24500000/4           // SYSCLK frequency in Hz
#define ON  1
#define OFF 2
#define REG 3
//#define PASS 0
//#define FAIL 1

//SW_UART TIMING VARIABLES
#define BAUD_RATE             2400
#define TIME_COUNT                  SYSCLK/BAUD_RATE/4
#define TH_TIME_COUNT         TIME_COUNT*3/2
```

```
#define MY_ADDRESS                  0x09060502

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------
bit ADDRESS_OK=0;
//Sample gloables
typedef struct
{
      unsigned long voltage;
      unsigned short vsamples;
      //unsigned long current[2];
      //unsigned short csamples[2];
}sample_t;
sample_t sample;

unsigned long result = 0L;
unsigned short index = 0;
unsigned char crossing = 0;
bit START_SAMPLE = 0;
//bit SAMPLE_COMPLETE = 0;
bit SAMPLING=0;
unsigned short EIE1_TEMP=0x80;
unsigned short AMX0SL_PIN=0x82;
//unsigned short adc_timeout=0;

//Packet TX/RX gloables
unsigned long transmit_packet[4];
unsigned long receive_packet[4];

//Software UART gloables
bit SRI=0;
bit STI=0;
bit STXBUSY;
bit SW_DONE;
bit SREN;
```

```
//bit SES;


sbit SW_RX = P0^0;
sbit SW_TX = P0^1;

unsigned long TDR;
unsigned long RDR;

short rx_index=0;
short tx_index=0;

bit first_word = 0;

//Power Regulation gloables
bit REGULATE0=0;
bit REGULATE1=0;

char TRIACSTATE[2]=OFF;

sbit TRIAC0 = P0^6;
sbit TRIAC1 = P0^4;

unsigned char counts[2] = 130;

//-----------------------------------------------------------------------------
// Function PROTOTYPES
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
void PORT_Init (void);
void Timer0_Init(unsigned char counts);
void Timer1_Init (unsigned char counts);
void Timer2_Init (int counts);
void ADC_Init (void);
void ADC_Window_ISR (void);
void ADC_Complete_ISR (void);
void SWUART_Init(void);
void SWUART_Enable(void);
```

```
void descifer_command(char * command);
void check_sample(void);
void build_packet(void);
void Timer0_ISR(void);
void Timer1_ISR(void);
void build_ack(void);
void build_nack(void);
unsigned char calculate_checksum(unsigned char *, unsigned char);
//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void
main (void)
{
      int i=0, j=0, delay=0;

      PCA0MD &= ~0x40;                       //WDTE = 0 (clear watchdog timer

      SYSCLK_Init ();                        // Initialize system clock to
                                           // (24.5)/4MHz
      PORT_Init ();                          // Initialize crossbar and GPIO


      Timer2_Init (5);
      ADC_Init ();
      SWUART_Init();
      SWUART_Enable();

      CKCON &= ~0x09;
      CKCON |=  0x02;

      REGULATE0 = 0;
      REGULATE1 = 0;                            // not regulating power
      TRIAC0 = 0;                               // signal on pin 6 is 0; triac off
      TRIAC1 = 0;                               // signal on pin 7 is 0; triac off
      TRIACSTATE[0] = OFF;                // triac is in off state initially
      TRIACSTATE[1] = OFF;                // triac is in off state initially
      SREN = 1;
```

```
EA   = 1;                                // enable global interrupts

while (1)                                // spin forever
{
      if(START_SAMPLE)
      {
            START_SAMPLE=0;
            SAMPLING=1;

            AMX0SL=AMX0SL_PIN;
            //sample_data();
      }
      //else if(SAMPLING)
      else//if(!START_SAMPLE)
      {
            if(SRI && SREN)
            {

                  SRI=0;

                  receive_packet[rx_index]=RDR;

                  if((receive_packet[0] & 0xdb000000)==0xdb000000)
                        rx_index+=1;

                  if(rx_index == 4)
                  {
                        descifer_command(receive_packet);
                        rx_index=0;
                  }

            }
            if(STI)
            {

                  STI=0;

                  if(tx_index==0)
```

```
                        {
                              STXBUSY=0;
                              SW_TX = 1;
                              P0 |= 0x08;

                              while((delay++)<100)
                              {
                              }

                              delay=0;
                        }

                        if(tx_index<4)
                        {
                              if(tx_index == 1)
                                    first_word = 1;
                              STXBUSY = 1;
                              TDR = transmit_packet[tx_index];
                              tx_index+=1;
                              CCF1 = 1;
                        }
                        else
                        {
                              P0 &=~0x08;
                              SW_DONE = 1;
                              tx_index = 0;
                              first_word = 0;
                        }
                  }
                  //AMX0SL=0x82;
            }
      }
}

//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
//
```

```
// This routine initializes the system clock to use the internal 24.5MHz / 4
// oscillator as its clock source.  Also enables missing clock detector reset.
//
void
SYSCLK_Init (void)
{
   OSCICN = 0x05;                          // configure internal oscillator for
                                           // its lowest frequency
   RSTSRC = 0x04;                          // enable missing clock detector
}

//-----------------------------------------------------------------------------
// PORT_Init
//-----------------------------------------------------------------------------
//
// Configure the Crossbar and GPIO ports.
// P0.0 - SW_RX
// P0.1 - SW_TX
// P0.2 - VOLTAGE SAMPLE
// P0.3 - TX/RX CONTROL
// P0.4 - TRIAC 0
// P0.5 - CURRENT SAMPLE 0
// P0.6 - TRIAC 1
// P0.7 - CURRENT SAMPLE 1; C2D
//
void
PORT_Init (void)
{
   XBR0     =  0x2C;    // skip P0.3
   XBR1     =  0x40;    // PCA and UART peripherals selected
   XBR2     =  0xC0;    // weak pull-ups and cross bar enabled
   P0       =  0x00;    // initialize all port pins to zero
   P0MDOUT  =  0x5A;    // enable digital output on pins 1, 3, 6, 4
   P0MDIN  &= ~0xA4;    // anolog inputs on pins 2, 5, 7
}

//-----------------------------------------------------------------------------
// Timer2_Init
```

```
//-----------------------------------------------------------------------------
//
// Configure Timer2 to 16-bit auto-reload and generate an interrupt at
// interval specified by <counts> using SYSCLK/48 as its time base.
//
void
Timer2_Init (int counts)
{
      TMR2CN  = 0x01;                          // Stop Timer2; Clear TF2;
                                            // use SYSCLK/12 as timebase
      CKCON  |= 0x60;                          // Timer2 clocked based on T2XCLK;

      TMR2RL  = -counts;                       // Init reload values
      TMR2    = 0xffff;                        // set to reload immediately
      ET2     = 1;                             // enable Timer2 interrupts
      TR2     = 1;
      //PT2       = 1;                             // start Timer2
}

void
Timer0_Init (char counts)
{
      TCON  &= ~0x30;
      TCON  |=  0x01;

      TMOD  &= ~0x0F;
      TMOD  |=  0x02;

      TH0 = 0xff - counts;
      TL0 = 0xff - counts;

      ET0 = 1;
      IP  |= 0x02;
      TCON  |= 0x10;
}

void
Timer1_Init (char counts)
```

```
{
      TCON  &= ~0xC0;
      TCON  |=  0x04;

      TMOD  &= ~0xF0;
      TMOD  |=  0x20;

      TH1 = 0xff - counts;
      TL1 = 0xff - counts;

      ET1 = 1;
      IP  |= 0x08;
      TCON  |= 0x40;
}

//-----------------------------------------------------------------------------
// void SWUART_Init(void)
//-----------------------------------------------------------------------------
// Initializes Software UART for use on base station controller, used to
// establish the link to the wall units
//
void
SWUART_Init(void)
{
      PCA0CPM0 = 0x10;  //Module 0 in negative capture mode; interrupt disabled
      PCA0CPM1 = 0x48;  //Module 1 in time capture compare mode; interrupt disabled

      PCA0CN   = 0x00;  //clear all interrupt flags that may be pending
      PCA0MD   = 0x02;  //SYSCLK/12; timer overflow interrupts disable

      CCF0     = 0x0 ;  //clear capture module0 interrupt flag
      CCF1     = 0x0 ;  //clear capture module1 interrupt flag

      SRI      = 0x0 ;  //clear SWUART receive flag
      STI      = 0x0 ;  //clear SWUART transmit flag

      SW_TX    = 0x1 ;  //set SWUART TX pin high
      STXBUSY  = 0x0 ;  //set TXBSY to 0
```

```
}

//-----------------------------------------------------------------------------
//    void SWUART_Enable(void)
//-----------------------------------------------------------------------------
//    Enable the SWUART for transmit and receive
//
void
SWUART_Enable(void)
{
      PCA0CPM0 |= 0x01; //Enable module 0 (receive) interrupt
      PCA0CPM1 |= 0x01; //Enable module 1 (transmit) interrupt

      CR    = 1;              //Start PCA counter
      EIE1 |= 0x08;           //Enable PCA interrupts
      EIP1 |= 0x08;
}

//------------------------------------------------------------------------------
// ADC_Init
//------------------------------------------------------------------------------
//
// Configure ADC to sample when signal is above 1.5V
//
void
ADC_Init (void)
{
  REF0CN = 0x0A;                                // Set VREF to VDD

  AMX0SL  = 0x82;                   // Sample Voltage on Port 3
  //AMX0SL = 0x81;                                      // Select single ended mode by making GRND the
                                                              // the lower bound, also selects P0.1 as
input signal

  ADC0GT = 0x7a;//80;                                   // Set greater than reg (lower window bound) to
                                                              // VREF*(128/256)

  ADC0LT = 0x00;                                // Set less than reg (upper window bound ) to
```

```
                                                              // VREF*(255/256)

  ADC0CF = 0x01;                          // SYSCLK/8;

  ADC0CN = 0x82;                                        // Set interupt to be generated when sample is
inside
                                                              // spec window, and enables ADC
  EIP1  |= 0x06;
  EIE1   |= 0x02;
}

//------------------------------------------------------------------------------
//------------------------------------------------------------------------------
//    Interrupt Service Routines
//------------------------------------------------------------------------------
//------------------------------------------------------------------------------

//------------------------------------------------------------------------------
// ADC_Window_ISR
//------------------------------------------------------------------------------
// This routine takes the ADC0 performs the opperation result += sample every 128 times
//
void
ADC_Window_ISR (void) interrupt 7
{

      if(SAMPLING)
            crossing+=1;

      if(ADC0GT == 0x7a)
      {
            ADC0GT = 0xFF;                              // Set greater than reg (upper window bound) to
                                                              // VREF*(128/256)

            ADC0LT = 0x7a;//0xFF - ADC0;                // Set less than reg (lower window bound) to
                                                              // ADC0
            while(ADC0GT != 0xff)
            {
```

```
		}

		AD0WINT = 0;

		if(crossing == 3)
		{
			if(ADC0>0x7a)
			{
				EIE1_TEMP=EIE1;
				EIE1=0x04;
				result= ADC0-0x7a;
				AD0INT = 0;
			}
			else
				crossing=0;
		}
	}
	else
	{
		ADC0GT = 0x7a;//0x80;			// Set greater than reg (upper window bound) to
								// VREF*(128/256)

		ADC0LT = 0x00;					// Set less than reg (lower window bound) to
								// ADC0
		while(ADC0GT != 0x7a)
		{
		}

		AD0WINT = 0;

		if(crossing > 3)
		{
			//SAMPLE_COMPLETE=1;
			crossing=0;
		}

	}
```

```
	//adc_timeout+=1;


	//if(AMX0SL==0x82)
	{
		if(REGULATE0)
		{
			Timer0_Init(5);
		}

		if(REGULATE1)
		{
			Timer1_Init(5);
		}
	}
	return;
}

//---------------------------------------------------------------------------------
// ADC_Complete_ISR
//---------------------------------------------------------------------------------
// This routine takes the ADC0 performs the opperation result += ADC0 within
// one crossing

void
ADC_Complete_ISR(void) interrupt 8
{

	AD0INT = 0;

	if(crossing==3)
	{
		index+=1;

		if(ADC0 >= 0x7a)
			result += ADC0 - 0x7a;
		else
```

```
            {
                  result +=(0x7a - ADC0);
                  SAMPLING=0;
                  //SAMPLE_COMPLETE=1;
                  EIE1 &= ~0x04;
                  AD0WINT = 0;
                  crossing=0;

                  EIE1=EIE1_TEMP;

                  check_sample();
                        //build_packet();
            }
      }
      else
      {
            //timeout=0;
            SAMPLING=0;
            //SAMPLE_COMPLETE=1;
            EIE1 &= ~0x04;
            //AMX0SL  = 0x82;
            AD0WINT = 0;
            //EIE1 |=  0x02;
            crossing=0;

            EIE1=EIE1_TEMP;
      }
}

//-----------------------------------------------------------------------------
//    PCA_ISR(void)
//-----------------------------------------------------------------------------
//    Interrupt service routine for the PCA used to implement the SW_UART
//
void
PCA_ISR(void) interrupt 9
{
      static char SUTXST = 0;             //SW_UART TX state variable
```

```
static char SURXST = 0;             //SW_UART RX state variable
static char timeout = 0;
static unsigned long RXSHIFT; //SW_UART RX shift register

unsigned short PCA_TEMP;            //Temporary storage variable for
                                          //manipulating PCA module high & low bytes

//if(!START_SAMPLE)
//{
      //Check receive interrupt flag first; service if CCF0 is set
      if(CCF0)
      {
            CCF0 = 0;

            if(SURXST==0)
            {
                  if(SREN & ~SW_RX)
                  {
                        PCA_TEMP  = (PCA0CPH0 << 8);
                        PCA_TEMP |= PCA0CPL0;

                        PCA_TEMP += TH_TIME_COUNT*9/4;

                        PCA0CPL0  = PCA_TEMP;
                        PCA0CPH0  = (PCA_TEMP >>8);

                        PCA0CPM0  |= 0x48;

                        SURXST++;
                  }
            }
            else if(SURXST>0 && SURXST<9)
            {
                  RXSHIFT = RXSHIFT >> 1;
                  if(SW_RX)
                        RXSHIFT |= 0x80000000;

                  PCA_TEMP  = (PCA0CPH0 << 8);
```

```
        PCA_TEMP |= PCA0CPL0;

        PCA_TEMP += TIME_COUNT*4/3;

        PCA0CPL0  = PCA_TEMP;
        PCA0CPH0  = (PCA_TEMP >>8);

        SURXST++;

        if((SURXST==9)&&((RXSHIFT & 0x96000000)!=0x96000000))
        {
              SURXST--;
              timeout++;
        }
        if(timeout>8)
        {
              PCA0CPM0 = 0x11;
              SURXST=0;
              timeout = 0;
        }
  }
  else if(SURXST>8 && SURXST<73)
  {
        if(SURXST%2)
        {
              RXSHIFT = RXSHIFT >> 1;
              if(SW_RX)
                    RXSHIFT |= 0x80000000;
        }

        PCA_TEMP  = (PCA0CPH0 << 8);
        PCA_TEMP |= PCA0CPL0;

        PCA_TEMP += TIME_COUNT*4/3;

        PCA0CPL0  = PCA_TEMP;
        PCA0CPH0  = (PCA_TEMP >>8);
```

```
            SURXST++;
        }
        else if(SURXST>72)
        {

            RDR = RXSHIFT;

            SRI = 1;

            PCA0CPM0 = 0x11;

            SURXST = 0;
        }
    }
    else if(CCF1)
    {
        CCF1 = 0;

        if(SUTXST<12)
        {
            if(SUTXST==1 || SUTXST==3 || SUTXST==5)
                SW_TX = 1;
            else if(SUTXST==0 || SUTXST==2 || SUTXST==4)
                SW_TX = 0;
            else if(SUTXST==6)
                SW_TX = 1;
            else if(SUTXST==7)
                SW_TX = 0;
            else if(SUTXST==8)
                SW_TX = 1;
            else if(SUTXST==9)
                SW_TX = 0;
            else if(SUTXST==10)
                SW_TX = 0;
            else if(SUTXST==11)
                SW_TX = 1;

            PCA_TEMP  = PCA0L;
```

```
        PCA_TEMP |= (PCA0H << 8);

        if(SUTXST==0)
              PCA_TEMP += TH_TIME_COUNT;
        else
              PCA_TEMP += TIME_COUNT;

        PCA0CPL1  = PCA_TEMP;
        PCA0CPH1  = (PCA_TEMP >> 8);

        if(SUTXST == 0)
              PCA0CPM1 = 0x49;

        SUTXST++;
  }
  else if(SUTXST>11 && SUTXST<76)
  {

        if(SUTXST%2)
        {
              SW_TX = !(TDR & 0x01);
              TDR >>= 1;
              TDR  |= 0x80000000;
        }
        else
              SW_TX = (TDR & 0x01);


        PCA_TEMP  = PCA0L;
        PCA_TEMP |= (PCA0H << 8);

        PCA_TEMP += TIME_COUNT;

        PCA0CPL1  = PCA_TEMP;
        PCA0CPH1  = (PCA_TEMP >> 8);

        SUTXST++;
  }
```

```
                  else if(SUTXST>75)
                  {
                        SUTXST   = 0;
                        SW_TX    = 1;
                        PCA0CPM1 = 0x01;
                        STXBUSY  = 0;
                        STI      = 1;

                  }
            }
}

void
Timer0_ISR(void) interrupt 1
{
      static unsigned char turn_on = 0;

      TF0 = 0;
      turn_on+=1;

      if(turn_on==counts[0])
      {
            TRIAC0 = 1;
      }
      if(turn_on>(counts[0]))
      {
            turn_on  =0;

            ET0      =0;
            TR0      =0;

            TRIAC0   =0;
      }

      return;
}

void
```

```
Timer1_ISR(void) interrupt 3
{
      static unsigned char turn_on = 0;

      TF1 = 0;
      turn_on+=1;

      if(turn_on==counts[1])
      {
            TRIAC1 = 1;
      }
      if(turn_on>(counts[1]))
      {
            turn_on  =0;

            ET1      =0;
            TR1      =0;

            TRIAC1   =0;
      }

      return;
}

void
descifer_command(char * command)
{
      unsigned char temp_counts;
      long delay=0;
      bit hi_low = 0;
      delay= (((command[8]>>4) & 0x0F)*8);

      if((command[8]&0xe) && command[9]==calculate_checksum(command, 8))
      {
            if (((command[7]>>5)&0x7)==0x7)
            {
                  switch(((command[7])&0xf))
                  {
```

```
			case 0:
				ADDRESS_OK=0;
				if(((MY_ADDRESS >> (((command[8]>>4) & 0x0F)*8)) & 0xff)== (((*((long
*)&command[1])) >> (((command[8]>>4) & 0x0F)*8)) & 0xff))
				{
					build_ack();
				}
				break;
			case 1:
				if((MY_ADDRESS & 0xffff)== ((*((long *)&command[1])) & 0xffff))
				{
					build_ack();
				}
				break;
			case 2:
				if((MY_ADDRESS & 0x00ffffff)== ((*((long *)&command[1])) & 0x00ffffff))
				{
					build_ack();
				}
				break;
			case 3:
				if(MY_ADDRESS== *((long *)&command[1]))
				{
					build_ack();
				}
				break;
			case 4:
				if(MY_ADDRESS== *((long *)&command[1]))
				{
					build_ack();
					ADDRESS_OK=1;
				}
				break;
			default:
				if(((MY_ADDRESS >> (((command[8]>>4) & 0x0F)*8)) & 0xff)== (((*((long
*)&command[1])) >> (((command[8]>>4) & 0x0F)*8)) & 0xff))
				{
					build_ack();
```

```
                        }
                }
        }
        else if(MY_ADDRESS == *((long *)&command[1]))
        {
                if(command[7]&0x10)
                        hi_low = 1;
                else
                        hi_low = 0;

                switch((command[7]>>5)&0x7)
                {
                        case 0x0:
                                AMX0SL_PIN  = 0x82;                          // Sample Voltage on Port 3
                                //SAMPLE_COMPLETE = 0;
                                START_SAMPLE = 1;
                                //sample_data();
                                command[0]='e';
                                break;
                        case 0x1:
                                RSTSRC|=0x10;
                                //SAMPLE_COMPLETE=1;
                                break;
                        case 0x2:
                                if(hi_low)
                                {
                                        REGULATE1 = 0;
                                        ET1   = 0;
                                        IP   &= ~0x08;
                                        TCON &= ~0x40;

                                        TRIAC1 = 1;
                                }
                                else
                                {
                                        REGULATE0 = 0;
                                        ET0   = 0;
                                        IP   &= ~0x02;
```

```
                TCON &= ~0x10;

                TRIAC0 = 1;
          }
          TRIACSTATE[hi_low]=ON;
          build_ack();
          command[0]='e';
          break;
    case 0x3:
          if(hi_low)
          {
                REGULATE1 = 0;
                ET1   = 0;
                IP   &= ~0x08;
                TCON &= ~0x40;
                TRIAC1 = 0;
          }
          else
          {
                REGULATE0 = 0;
                ET0   = 0;
                IP   &= ~0x02;
                TCON &= ~0x10;
                TRIAC0 = 0;
          }
          TRIACSTATE[hi_low]=OFF;
          build_ack();
          command[0]='e';
          break;
    case 0x4:
          temp_counts  = 0;
          temp_counts  = (command[7]<<4) & 0xF0;
          temp_counts |= (command[8]>>4) & 0x0F;

          counts[hi_low]=temp_counts;

          if(hi_low)
                REGULATE1 = 1;
```

```
                        else
                              REGULATE0 = 1;
                        TRIACSTATE[hi_low]=REG;
                        build_ack();
                        command[0]='e';
                        break;
                  case 0x5:
                        AMX0SL_PIN=0x82;
                        index=0;
                        result=0;
                        //SAMPLE_COMPLETE = 0;
                        START_SAMPLE = 1;
                        //sample_data();
                        command[0]='e';
                        break;
                  case 0x6:
                        AMX0SL_PIN  = 0x82;                        // Sample Voltage on Port 3
                        index=0;
                        result=0;
                        //SAMPLE_COMPLETE = 0;
                        START_SAMPLE = 1;
                        //sample_data();
                        command[0]='e';
                        break;
                  default:
                        build_nack();
                        command[0]='e';
                        break;
            }
      }
   }
   //else
   //    build_nack();
}

//-------------------------------------------------------------------------------
//    void sample_data()
//-------------------------------------------------------------------------------
```

```
// Samples voltage and current data and stores them in the sample struct for
// transmission back to the control unit
//

void
check_sample(void)
{
	static char samples=0;

	SAMPLING=0;
	sample.voltage  = result;	// Store Sample in Sample Structure for Transmit
	sample.vsamples = index;
	index = 0;
	result=0;

	samples+=1;

	if(sample.vsamples >200 && sample.vsamples <450 && (sample.voltage)<30000)
	{
		samples=0;
		build_packet();
		//return(PASS);
	}
	else if((samples)>1)
	{
		crossing=0;
		AD0WINT = 0;
		EIE1=EIE1_TEMP;
		while(EIE1!=EIE1_TEMP){}
		build_nack();
		samples=0;
		AMX0SL  = 0x82;
		while(AMX0SL != 0x82){};
		//return(FAIL);
	}
	else
		SAMPLING=1;
```

```
	//AMX0SL  = 0x82;

	return;
}

void
build_packet(void)
{
	//unsigned char i=0, check_sum=0;
	unsigned char temp=0;
	if(!STXBUSY)
	{
		transmit_packet[0] = (unsigned long)((0xDB000000)|(0x0100)|(MY_ADDRESS>>24));
		transmit_packet[1] = (unsigned long)((MY_ADDRESS<<8)|(sample.voltage>>24));
		transmit_packet[2] = (unsigned long)((sample.voltage<<8)|(sample.vsamples>>8));
		(unsigned short)transmit_packet[3] = sample.vsamples<<8;
		temp = calculate_checksum((unsigned char *)(transmit_packet), 12);
		//for(i=1; i<=12; i++)
		//	check_sum ^= transmit_packet[i];
		//((unsigned char *)&(transmit_packet))[15]|= check_sum;
		transmit_packet[15]|= temp;
		STI = 1;
	}
}

void
build_ack(void)
{
	transmit_packet[0] = (unsigned long)((0xBD000000)|(0x0100)|(MY_ADDRESS>>24));
	transmit_packet[1] = (unsigned long)(MY_ADDRESS<<8);
	STI =1;
	return;
}

void
build_nack(void)
{
	transmit_packet[0] = (unsigned long)((0xBB000000)|(0x0100)|(MY_ADDRESS>>24));
```

```
	transmit_packet[1] = (unsigned long)(MY_ADDRESS<<8);
	STI = 1;
	return;
}

unsigned char
calculate_checksum(unsigned char *ptr, unsigned char num_bytes)
{
	unsigned char i=0, check_sum=0;

	for(i=1; i<=num_bytes; i++)
		check_sum ^= ptr[i];

	return(check_sum);

}
```

Baseunit.c

```
//-----------------------------------------------------------------------------
// baseunit.c
//-----------------------------------------------------------------------------
// This code is to be used to control the wall unit for our senior design project.
// It will eventually be sampling an anolog signal, to compute the power disipated
// in a wall socket.  Also, there will eventually be the ability to regulate the power
// for apliances and lights, etc.
//
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <c8051f300.h>                    // SFR declarations

//-----------------------------------------------------------------------------
// SFR Definitions for 'F30x
//-----------------------------------------------------------------------------

sfr16 DP       = 0x82;                    // data pointer
sfr16 TMR2RL   = 0xca;                    // Timer2 reload value
sfr16 TMR2     = 0xcc;                    // Timer2 counter
sfr16 PCA0CP1  = 0xe9;                    // PCA0 Module 1 Capture/Compare
sfr16 PCA0CP2  = 0xeb;                    // PCA0 Module 2 Capture/Compare
sfr16 PCA0     = 0xf9;                    // PCA0 counter
sfr16 PCA0CP0  = 0xfb;                    // PCA0 Module 0 Capture/Compare
sfr   EIP1     = 0xF6;                    // EXTENDED INTERRUPT PRIORITY

//-----------------------------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------------------------

#define SYSCLK                         24500000/4          // SYSCLK frequency in Hz

//SW_UART TIMING VARIABLES
#define BAUD_RATE              2400
#define TIME_COUNT                     SYSCLK/BAUD_RATE/4
#define TH_TIME_COUNT          TIME_COUNT*3/2
```

```
//
#define MAX_PACKET_LENGTH     15
#define MY_ADDRESS                  0x00000004
//#define ESCAPE_CHAR

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------
//Sample gloables
typedef struct
{
      unsigned long voltage;
      unsigned short vsamples;
      unsigned long current[2];
}sample_t;
sample_t sample;

//Packet TX/RX gloables
unsigned long relay_rx_packet[4];
unsigned long relay_tx_packet[4];
unsigned char hw_rx_packet[16];
unsigned char hw_tx_packet[16];

//Software UART gloables
bit SRI;
bit STI;
bit STXBUSY;
bit SW_DONE;
bit SREN = 0;
bit SES;
bit first_word=0;

sbit SW_RX = P0^0;
sbit SW_TX = P0^1;

unsigned long TDR;
unsigned long RDR;
```

```
bit HW_DONE;
//-----------------------------------------------------------------------------
// Function PROTOTYPES
//-----------------------------------------------------------------------------
void SYSCLK_Init (void);
void PORT_Init (void);
void Timer1_Init(void);
void UART_Init(void);
void UART_ISR(void);
void SWUART_Init(void);
void SWUART_Enable(void);
void sample_data(void);
void build_packet(void);
unsigned char balanced_byte(unsigned char unbalanced_byte);
//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void
main (void)
{
      int x=0, j=0, delay=0, p=0;

      PCA0MD &= ~0x40;                       //WDTE = 0 (clear watchdog timer

      SYSCLK_Init ();                        // Initialize system clock to
                                           // (24.5)/4MHz
      PORT_Init ();                          // Initialize crossbar and GPIO

      Timer1_Init();
      UART_Init();
      SWUART_Init();
      SWUART_Enable();

      SREN = 0x1;
      EA   = 1;                                                 // enable global interrupts

      while (1)                                           // spin forever
```

```
{

	if(HW_DONE)
	{
		HW_DONE=0;
	}
	if(SRI && SREN)
	{

		SRI=0;
		relay_rx_packet[p]=RDR;
		if ( ( ( relay_rx_packet[0] ) & ( 0xdb000000 ) ) == 0xdb000000 ) {
			p+=1;
		} else if ( ( ( relay_rx_packet[0] ) & ( 0xbd000000 ) ) == 0xbd000000 ) {
			p+=1;
		} else if ( ( ( relay_rx_packet[0] ) & ( 0xbb000000 ) ) == 0xbb000000 ) {
			p+=1;
		}
		if(p == 4)
		{
			for(j=0; j<16; j+=4)
			{
				*((long*)(&hw_tx_packet[j])) = relay_rx_packet[j/4];
				relay_rx_packet[j/4]=0;
			}
			p=0;
			TI0 =  1;
		}

	}
	if(STI && !STXBUSY)
	{
		STI=0;

		if(x==0)
		{
			SW_TX = 1;
			P0 |= 0x08;
```

```
                        while((delay++)<100)
                        {
                        }

                        delay=0;
                  }

                  if(x<4)
                  {
                        if(x == 1)
                              first_word = 1;
                        STXBUSY = 1;
                        TDR = relay_tx_packet[x];
                        x+=1;
                        CCF1 = 1;
                  }
                  else
                  {
                        //for(j=0; j<16; j+=4)
                        //    *((long*)(&relay_tx_packet[j])) = 0;
                        P0 &=~0x08;
                        SW_DONE = 1;
                        x = 0;
                        first_word = 0;
                  }
            }
      }
}

//-----------------------------------------------------------------------------
// SYSCLK_Init
//-----------------------------------------------------------------------------
//
// This routine initializes the system clock to use the internal 24.5MHz / 4
// oscillator as its clock source.  Also enables missing clock detector reset.
//
void
```

```
SYSCLK_Init (void)
{
   OSCICN = 0x05;                          // configure internal oscillator for
                                           // its lowest frequency
   RSTSRC = 0x04;                          // enable missing clock detector
}

//-----------------------------------------------------------------------------
// PORT_Init
//-----------------------------------------------------------------------------
//
// Configure the Crossbar and GPIO ports.
// P0.0 - SW_RX
// P0.1 - SW_TX
// P0.2 -
// P0.3 - CONTROL
// P0.4 - TX
// P0.5 - RX
// P0.6 -
// P0.7 - C2D
//
void
PORT_Init (void)
{
   XBR0     =  0x08;    // skip P0.3
   XBR1     =  0x43;    // PCA and UART peripherals selected
   XBR2     =  0xc0;    // weak pull-ups and cross bar enabled
   P0       =  0x00;    // initialize all pins to 0
   P0MDOUT  =  0x1a;    // enable digital output on pins 1, 3, 4
   P0MDIN  &= ~0x00;    // all inputs are digital
}

//-----------------------------------------------------------------------------
// Timer1_Init
//-----------------------------------------------------------------------------
//
// Configur Timer1 to 8 bit auto-reload for use with the UART clock generation
//
```

```
void
Timer1_Init(void)
{
      TMOD  |= 0x28;
      CKCON &= ~0x03;

      TL1    = 0x96;
      TH1    = 0x96;

      TCON  |= 0x40;

      IE    |= 0x08;

}

//-------------------------------------------------------------------------------
// UART_Init
//-------------------------------------------------------------------------------
//
// Configure the UART for single processor comms, ignoring stop bit logic level
// and enable recieve
//
void
UART_Init(void)
{
      //enable revceive
      SCON0 = 0x10;
      IE   |= 0x10;
      RI0   = 0;
      TI0   = 0;
      IP    = 0x10;

      HW_DONE = 0;
}

//------------------------------------------------------------------------------
// void SWUART_Init(void)
//------------------------------------------------------------------------------
```

```
// Initializes Software UART for use on base station controller, used to
// establish the link to the wall units
//
void
SWUART_Init(void)
{
      PCA0CPM0 = 0x10;  //Module 0 in negative capture mode; interrupt disabled
      PCA0CPM1 = 0x48;  //Module 1 in time capture compare mode; interrupt disabled

      PCA0CN   = 0x00;  //clear all interrupt flags that may be pending
      PCA0MD   = 0x02;  //SYSCLK/4; timer overflow interrupts disable

      CCF0     = 0x0 ;  //clear capture module0 interrupt flag
      CCF1     = 0x0 ;  //clear capture module1 interrupt flag

      SRI      = 0x0 ;  //clear SWUART receive flag
      STI      = 0x0 ;  //clear SWUART transmit flag

      SW_TX    = 0x1 ;  //set SWUART TX pin high
      STXBUSY  = 0x0 ;  //set TXBSY to 0
      //SREN     = 0x0 ;
}

//------------------------------------------------------------------------------
//    void SWUART_Enable(void)
//------------------------------------------------------------------------------
//    Enable the SWUART for transmit and receive
//
void
SWUART_Enable(void)
{
      PCA0CPM0 |= 0x01; //Enable module 0 (receive) interrupt
      PCA0CPM1 |= 0x01; //Enable module 1 (transmit) interrupt

      CR    = 1;              //Start PCA counter
      EIE1 |= 0x08;           //Enable PCA interrupts
      EIP1 |= 0x08;
}
```

```
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//    Interrupt Service Routines
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------

//-----------------------------------------------------------------------------
// UART_ISR()
//-----------------------------------------------------------------------------
// This routine handles both TX and RX interrupts for the UART
//
void
UART_ISR(void) interrupt 4
{
      static idata char u=0;
      static int i=0;
      int j =0;

      if(RI0)
      {

            RI0=0;

            hw_rx_packet[u]=SBUF0;

            if(hw_rx_packet[0]==(char)(0xDB))
                  u+=1;

            if(u == 10)
            {
                  for(j=0; j<16; j+=4)
                  {
                        relay_tx_packet[j/4] = *((long*)(&hw_rx_packet[j]));
                        *((long*)(&hw_rx_packet[j]))=0;
                  }
                  u=0;
                  STI = 1;
```

```
		}

	}
	if(TI0)
	{
		TI0=0;

		if(i < 16)
		{
			SBUF0 = hw_tx_packet[i];
			i=i+1;
		}
		else
		{
			//for(j=0; j<16; j+=4)
			//		*((long*)(&hw_tx_packet[j])) = 0;
			i=0;
		}
	}
}

//-----------------------------------------------------------------------------
//	PCA_ISR(void)
//-----------------------------------------------------------------------------
//	Interrupt service routine for the PCA used to implement the SW_UART
//
void
PCA_ISR(void) interrupt 9
{
	static char SUTXST = 0;			//SW_UART TX state variable
	static char SURXST = 0;			//SW_UART RX state variable
	static char timeout = 0;
	static unsigned long RXSHIFT; //SW_UART RX shift register

	unsigned short PCA_TEMP;		//Temporary storage variable for
						//manipulating PCA module high & low bytes

	//Check receive interrupt flag first; service if CCF0 is set
```

```
if(CCF0)
{
      CCF0 = 0;

      if(SURXST==0)
      {
            if(~SW_RX)
            {
                  PCA_TEMP  = (PCA0CPH0 << 8);
                  PCA_TEMP |= PCA0CPL0;

                  PCA_TEMP += TH_TIME_COUNT*9/4;

                  PCA0CPL0  = PCA_TEMP;
                  PCA0CPH0  = (PCA_TEMP >>8);

                  PCA0CPM0  |= 0x48;

                  SURXST++;
            }
      }
      else if(SURXST>0 && SURXST<9)
      {
            RXSHIFT = RXSHIFT >> 1;
            if(SW_RX)
                  RXSHIFT |= 0x80000000;

            PCA_TEMP  = (PCA0CPH0 << 8);
            PCA_TEMP |= PCA0CPL0;

            PCA_TEMP += TIME_COUNT*4/3;

            PCA0CPL0  = PCA_TEMP;
            PCA0CPH0  = (PCA_TEMP >>8);

            SURXST++;

            if((SURXST==9)&&((RXSHIFT & 0x96000000)!=0x96000000))
```

```
		{
			SURXST--;
			timeout++;
		}
		if(timeout>8)
		{
			PCA0CPM0 = 0x11;
			SURXST=0;
			timeout = 0;
		}
	}
	else if(SURXST>8 && SURXST<73)
	{
		if(SURXST%2)
		{
			RXSHIFT = RXSHIFT >> 1;
			if(SW_RX)
				RXSHIFT |= 0x80000000;
		}

		PCA_TEMP  = (PCA0CPH0 << 8);
		PCA_TEMP |= PCA0CPL0;

		PCA_TEMP += TIME_COUNT*4/3;

		PCA0CPL0  = PCA_TEMP;
		PCA0CPH0  = (PCA_TEMP >>8);

		SURXST++;
	}
	else if(SURXST>72)
	{

		RDR = RXSHIFT;

		SRI = 1;

		PCA0CPM0 = 0x11;
```

```
            SURXST = 0;
        }
}
else if(CCF1)
{
        CCF1 = 0;

        if(SUTXST<12)
        {
            if(SUTXST==1 || SUTXST==3 || SUTXST==5)
                SW_TX = 1;
            else if(SUTXST==0 || SUTXST==2 || SUTXST==4)
                SW_TX = 0;
            else if(SUTXST==6)
                SW_TX = 1;
            else if(SUTXST==7)
                SW_TX = 0;
            else if(SUTXST==8)
                SW_TX = 1;
            else if(SUTXST==9)
                SW_TX = 0;
            else if(SUTXST==10)
                SW_TX = 0;
            else if(SUTXST==11)
                SW_TX = 1;

            PCA_TEMP  = PCA0L;
            PCA_TEMP |= (PCA0H << 8);

            if(SUTXST==0)
                PCA_TEMP += TH_TIME_COUNT;
            else
                PCA_TEMP += TIME_COUNT;

            PCA0CPL1  = PCA_TEMP;
            PCA0CPH1  = (PCA_TEMP >> 8);
```

```
            if(SUTXST == 0)
                  PCA0CPM1 = 0x49;

            SUTXST++;
      }
      else if(SUTXST>11 && SUTXST<76)
      {

            if(SUTXST%2)
            {
                  SW_TX = !(TDR & 0x01);
                  TDR >>= 1;
                  TDR  |= 0x80000000;
            }
            else
                  SW_TX = (TDR & 0x01);


            PCA_TEMP  = PCA0L;
            PCA_TEMP |= (PCA0H << 8);

            PCA_TEMP += TIME_COUNT;

            PCA0CPL1  = PCA_TEMP;
            PCA0CPH1  = (PCA_TEMP >> 8);

            SUTXST++;
      }
      else if(SUTXST>75)
      {
            SUTXST   = 0;
            SW_TX    = 1;
            PCA0CPM1 = 0x01;
            STXBUSY  = 0;
            STI      = 1;

      }
}
```

```
}
```

c8051f300.h

```
/*-----------------------------------------------------------------------------
;     Copyright (C) 2001 CYGNAL INTEGRATED PRODUCTS, INC.
;     All rights reserved.
;
;
;     FILE NAME   : C8051F300.H
;     TARGET MCUs : C8051F300, 'F301, 'F302, 'F303
;     DESCRIPTION : Register/bit definitions for the C8051F30x product family.
;
;     REVISION 1.1
;
;-----------------------------------------------------------------------------*/

/*  BYTE Registers  */
sfr P0       = 0x80;    /* PORT 0                                                 */
sfr SP       = 0x81;    /* STACK POINTER                                          */
sfr DPL      = 0x82;    /* DATA POINTER - LOW BYTE                                */
sfr DPH      = 0x83;    /* DATA POINTER - HIGH BYTE                               */
sfr PCON     = 0x87;    /* POWER CONTROL                                          */
sfr TCON     = 0x88;    /* TIMER CONTROL                                          */
sfr TMOD     = 0x89;    /* TIMER MODE                                             */
sfr TL0      = 0x8A;    /* TIMER 0 - LOW BYTE                                     */
sfr TL1      = 0x8B;    /* TIMER 1 - LOW BYTE                                     */
sfr TH0      = 0x8C;    /* TIMER 0 - HIGH BYTE                                    */
sfr TH1      = 0x8D;    /* TIMER 1 - HIGH BYTE                                    */
sfr CKCON    = 0x8E;    /* CLOCK CONTROL                                          */
sfr PSCTL    = 0x8F;    /* PROGRAM STORE R/W CONTROL                              */
sfr SCON0    = 0x98;    /* SERIAL PORT 0 CONTROL                                  */
sfr SBUF0    = 0x99;    /* SERIAL PORT 0 BUFFER                                   */
sfr CPT0MD   = 0x9D;    /* COMPARATOR 0 MODE                                      */
sfr CPT0MX   = 0x9F;    /* COMPARATOR 0 MUX                                       */
sfr P0MDOUT  = 0xA4;    /* PORT 0 OUTPUT MODE                                     */
sfr IE       = 0xA8;    /* INTERRUPT ENABLE                                       */
sfr OSCXCN   = 0xB1;    /* EXTERNAL OSCILLATOR CONTROL                            */
sfr OSCICN   = 0xB2;    /* INTERNAL OSCILLATOR CONTROL                            */
sfr OSCICL   = 0xB3;    /* INTERNAL OSCILLATOR CALIBRATION                        */
```

```
sfr FLKEY    = 0xB7;    /* FLASH LOCK & KEY                                         */
sfr IP       = 0xB8;    /* INTERRUPT PRIORITY                                       */
sfr AMX0SL   = 0xBB;    /* ADC 0 MUX CHANNEL SELECTION                              */
sfr ADC0CF   = 0xBC;    /* ADC 0 CONFIGURATION                                      */
sfr ADC0     = 0xBE;    /* ADC 0 DATA                                               */
sfr SMB0CN   = 0xC0;    /* SMBUS CONTROL                                            */
sfr SMB0CF   = 0xC1;    /* SMBUS CONFIGURATION                                      */
sfr SMB0DAT  = 0xC2;    /* SMBUS DATA                                               */
sfr ADC0GT   = 0xC4;    /* ADC0 GREATER-THAN                                        */
sfr ADC0LT   = 0xC6;    /* ADC0 LESS-THAN                                           */
sfr TMR2CN   = 0xC8;    /* TIMER 2 CONTROL                                          */
sfr TMR2RLL  = 0xCA;    /* TIMER 2 RELOAD LOW                                       */
sfr TMR2RLH  = 0xCB;    /* TIMER 2 RELOAD HIGH                                      */
sfr TMR2L    = 0xCC;    /* TIMER 2 LOW BYTE                                         */
sfr TMR2H    = 0xCD;    /* TIMER 2 HIGH BYTE                                        */
sfr PSW      = 0xD0;    /* PROGRAM STATUS WORD                                      */
sfr REF0CN   = 0xD1;    /* VOLTAGE REFERENCE 0 CONTROL                              */
sfr PCA0CN   = 0xD8;    /* PCA0 CONTROL                                             */
sfr PCA0MD   = 0xD9;    /* PCA0 MODE                                                */
sfr PCA0CPM0 = 0xDA;    /* PCA0 MODULE 0 MODE                                       */
sfr PCA0CPM1 = 0xDB;    /* PCA0 MODULE 1 MODE                                       */
sfr PCA0CPM2 = 0xDC;    /* PCA0 MODULE 2 MODE                                       */
sfr ACC      = 0xE0;    /* ACCUMULATOR                                              */
sfr XBR0     = 0xE1;    /* DIGITAL CROSSBAR CONFIGURATION REGISTER 0                */
sfr XBR1     = 0xE2;    /* DIGITAL CROSSBAR CONFIGURATION REGISTER 1                */
sfr XBR2     = 0xE3;    /* DIGITAL CROSSBAR CONFIGURATION REGISTER 2                */
sfr IT01CF   = 0xE4;    /* INT0/INT1 CONFIGURATION                                  */
sfr EIE1     = 0xE6;    /* EXTERNAL INTERRUPT ENABLE 1                              */
sfr ADC0CN   = 0xE8;    /* ADC 0 CONTROL                                            */
sfr PCA0CPL1 = 0xE9;    /* PCA0 MODULE 1 CAPTURE/COMPARE REGISTER LOW BYTE          */
sfr PCA0CPH1 = 0xEA;    /* PCA0 MODULE 1 CAPTURE/COMPARE REGISTER HIGH BYTE         */
sfr PCA0CPL2 = 0xEB;    /* PCA0 MODULE 2 CAPTURE/COMPARE REGISTER LOW BYTE          */
sfr PCA0CPH2 = 0xEC;    /* PCA0 MODULE 2 CAPTURE/COMPARE REGISTER HIGH BYTE         */
sfr RSTSRC   = 0xEF;    /* RESET SOURCE                                             */
sfr B        = 0xF0;    /* B REGISTER                                               */
sfr P0MDIN   = 0xF1;    /* PORT 0 INPUT MODE REGISTER                               */
sfr CPT0CN   = 0xF8;    /* COMPARATOR 0 CONTROL                                     */
sfr PCA0L    = 0xF9;    /* PCA0 COUNTER REGISTER LOW BYTE                           */
```

```
sfr PCA0H    = 0xFA;    /* PCA0 COUNTER REGISTER HIGH BYTE                         */
sfr PCA0CPL0 = 0xFB;    /* PCA MODULE 0 CAPTURE/COMPARE REGISTER LOW BYTE          */
sfr PCA0CPH0 = 0xFC;    /* PCA MODULE 0 CAPTURE/COMPARE REGISTER HIGH BYTE         */

/*  BIT Registers  */

/*  TCON  0x88 */
sbit IT0     = TCON ^ 0;   /* EXT INTERRUPT 0 TYPE                                 */
sbit IE0     = TCON ^ 1;   /* EXT INTERRUPT 0 EDGE FLAG                            */
sbit IT1     = TCON ^ 2;   /* EXT INTERRUPT 1 TYPE                                 */
sbit IE1     = TCON ^ 3;   /* EXT INTERRUPT 1 EDGE FLAG                            */
sbit TR0     = TCON ^ 4;   /* TIMER 0 ON/OFF CONTROL                               */
sbit TF0     = TCON ^ 5;   /* TIMER 0 OVERFLOW FLAG                                */
sbit TR1     = TCON ^ 6;   /* TIMER 1 ON/OFF CONTROL                               */
sbit TF1     = TCON ^ 7;   /* TIMER 1 OVERFLOW FLAG                                */

/*  SCON0  0x98 */
sbit RI0     = SCON0 ^ 0;  /* RECEIVE INTERRUPT FLAG                               */
sbit TI0     = SCON0 ^ 1;  /* TRANSMIT INTERRUPT FLAG                              */
sbit RB80    = SCON0 ^ 2;  /* RECEIVE BIT 8                                        */
sbit TB80    = SCON0 ^ 3;  /* TRANSMIT BIT 8                                       */
sbit REN0    = SCON0 ^ 4;  /* RECEIVE ENABLE                                       */
sbit MCE0    = SCON0 ^ 5;  /* MULTIPROCESSOR COMMUNICATION ENABLE                  */
sbit S0MODE  = SCON0 ^ 7;  /* SERIAL MODE CONTROL BIT 0                            */

/*  IE  0xA8 */
sbit EX0     = IE ^ 0;     /* EXTERNAL INTERRUPT 0 ENABLE                          */
sbit ET0     = IE ^ 1;     /* TIMER 0 INTERRUPT ENABLE                             */
sbit EX1     = IE ^ 2;     /* EXTERNAL INTERRUPT 1 ENABLE                          */
sbit ET1     = IE ^ 3;     /* TIMER 1 INTERRUPT ENABLE                             */
sbit ES0     = IE ^ 4;     /* UART0 INTERRUPT ENABLE                               */
sbit ET2     = IE ^ 5;     /* TIMER 2 INTERRUPT ENABLE                             */
sbit EA      = IE ^ 7;     /* GLOBAL INTERRUPT ENABLE                              */

/*  IP  0xB8 */
sbit PX0     = IP ^ 0;     /* EXTERNAL INTERRUPT 0 PRIORITY                        */
sbit PT0     = IP ^ 1;     /* TIMER 0 PRIORITY                                     */
sbit PX1     = IP ^ 2;     /* EXTERNAL INTERRUPT 1 PRIORITY                        */
```

```
sbit PT1     = IP ^ 3;     /* TIMER 1 PRIORITY                                    */
sbit PS0     = IP ^ 4;     /* UART0 PRIORITY                                      */
sbit PT2     = IP ^ 5;     /* TIMER 2 PRIORITY                                    */

/* SMB0CN 0xC0 */
sbit SI      = SMB0CN ^ 0; /* SMBUS0 INTERRUPT FLAG                               */
sbit ACK     = SMB0CN ^ 1; /* ACKNOWLEDGE FLAG                                    */
sbit ARBLOST = SMB0CN ^ 2; /* ARBITRATION LOST INDICATOR                          */
sbit ACKRQ   = SMB0CN ^ 3; /* ACKNOWLEDGE REQUEST                                 */
sbit STO     = SMB0CN ^ 4; /* STOP FLAG                                           */
sbit STA     = SMB0CN ^ 5; /* START FLAG                                          */
sbit TXMODE  = SMB0CN ^ 6; /* TRANSMIT MODE INDICATOR                             */
sbit MASTER  = SMB0CN ^ 7; /* MASTER/SLAVE INDICATOR                              */

/*  TMR2CN 0xC8 */
sbit T2XCLK  = TMR2CN ^ 0; /* TIMER 2 EXTERNAL CLOCK SELECT                       */
sbit TR2     = TMR2CN ^ 2; /* TIMER 2 ON/OFF CONTROL                              */
sbit T2SPLIT = TMR2CN ^ 3; /* TIMER 2 SPLIT MODE ENABLE                           */
sbit TF2LEN  = TMR2CN ^ 5; /* TIMER 2 LOW BYTE INTERRUPT ENABLE                   */
sbit TF2L    = TMR2CN ^ 6; /* TIMER 2 LOW BYTE OVERFLOW FLAG                      */
sbit TF2H    = TMR2CN ^ 7; /* TIMER 2 HIGH BYTE OVERFLOW FLAG                     */

/*  PSW 0xD0 */
sbit P       = PSW ^ 0;    /* ACCUMULATOR PARITY FLAG                             */
sbit F1      = PSW ^ 1;    /* USER FLAG 1                                         */
sbit OV      = PSW ^ 2;    /* OVERFLOW FLAG                                       */
sbit RS0     = PSW ^ 3;    /* REGISTER BANK SELECT 0                              */
sbit RS1     = PSW ^ 4;    /* REGISTER BANK SELECT 1                              */
sbit F0      = PSW ^ 5;    /* USER FLAG 0                                         */
sbit AC      = PSW ^ 6;    /* AUXILIARY CARRY FLAG                                */
sbit CY      = PSW ^ 7;    /* CARRY FLAG                                          */

/* PCA0CN 0xD8H */
sbit CCF0    = PCA0CN ^ 0; /* PCA0 MODULE 0 CAPTURE/COMPARE FLAG                  */
sbit CCF1    = PCA0CN ^ 1; /* PCA0 MODULE 1 CAPTURE/COMPARE FLAG                  */
sbit CCF2    = PCA0CN ^ 2; /* PCA0 MODULE 2 CAPTURE/COMPARE FLAG                  */
sbit CR      = PCA0CN ^ 6; /* PCA0 COUNTER RUN CONTROL                            */
sbit CF      = PCA0CN ^ 7; /* PCA0 COUNTER OVERFLOW FLAG                          */
```

```
/* ADC0CN 0xE8H */
sbit AD0CM0  = ADC0CN ^ 0; /* ADC0 CONVERSION MODE SELECT 0                           */
sbit AD0CM1  = ADC0CN ^ 1; /* ADC0 CONVERSION MODE SELECT 1                           */
sbit AD0CM2  = ADC0CN ^ 2; /* ADC0 CONVERSION MODE SELECT 2                           */
sbit AD0WINT = ADC0CN ^ 3; /* ADC0 WINDOW COMPARE INTERRUPT FLAG                      */
sbit AD0BUSY = ADC0CN ^ 4; /* ADC0 BUSY FLAG                                          */
sbit AD0INT  = ADC0CN ^ 5; /* ADC0 CONVERISION COMPLETE INTERRUPT FLAG                */
sbit AD0TM   = ADC0CN ^ 6; /* ADC0 TRACK MODE                                         */
sbit AD0EN   = ADC0CN ^ 7; /* ADC0 ENABLE                                             */

/* CPT0CN 0xF8H */
sbit CP0HYN0 = CPT0CN ^ 0; /* COMPARATOR 0 NEGATIVE HYSTERESIS 0                      */
sbit CP0HYN1 = CPT0CN ^ 1; /* COMPARATOR 0 NEGATIVE HYSTERESIS 1                      */
sbit CP0HYP0 = CPT0CN ^ 2; /* COMPARATOR 0 POSITIVE HYSTERESIS 0                      */
sbit CP0HYP1 = CPT0CN ^ 3; /* COMPARATOR 0 POSITIVE HYSTERESIS 1                      */
sbit CP0FIF  = CPT0CN ^ 4; /* COMPARATOR 0 FALLING-EDGE INTERRUPT FLAG                */
sbit CP0RIF  = CPT0CN ^ 5; /* COMPARATOR 0 RISING-EDGE INTERRUPT FLAG                 */
sbit CP0OUT  = CPT0CN ^ 6; /* COMPARATOR 0 OUTPUT STATE                               */
sbit CP0EN   = CPT0CN ^ 7; /* COMPARATOR 0 ENABLE                                     */
```

Senior Design:

Appendix E

PRELIMINARY

C8051F300/1/2/3 C8051F304/5

Mixed-Signal ISP FLASH MCU Family

ANALOG PERIPHERALS

- **8-Bit ADC**
  - Up to 500 ksps
  - Up to 8 External Inputs
  - Programmable Amplifier Gains of 4, 2, 1, & 0.5
  - VREF from External Pin or VDD
  - Built-in Temperature Sensor
  - External Conversion Start Input
- **Comparator**
  - Programmable Hysteresis and Response Time
  - Configurable as Interrupt or Reset Source
  - Low Current (< 0.5µA)

ON-CHIP DEBUG

- On-Chip Debug Circuitry Facilitates Full Speed, Non-Intrusive In-System Debug (No Emulator Required!)
- Provides Breakpoints, Single Stepping, Inspect/Modify Memory and Registers
- Superior Performance to Emulation Systems Using ICE-Chips, Target Pods, and Sockets
- Complete Development Kit: $99

SUPPLY VOLTAGE 2.7V TO 3.6V

- Typical Operating Current: 5mA @ 25 MHz; 11µA @ 32 kHz
- Typical Stop Mode Current: 0.1 µA
- Temperature Range: -40°C to +85°C

HIGH SPEED 8051 µC Core

- Pipe-lined Instruction Architecture; Executes 70% of Instructions in 1 or 2 System Clocks
- Up to 25 MIPS Throughput with 25 MHz Clock
- Expanded Interrupt Handler

MEMORY

- 256 Bytes Internal Data RAM
- 8k Bytes FLASH; In-System Programmable in 512 byte Sectors

DIGITAL PERIPHERALS

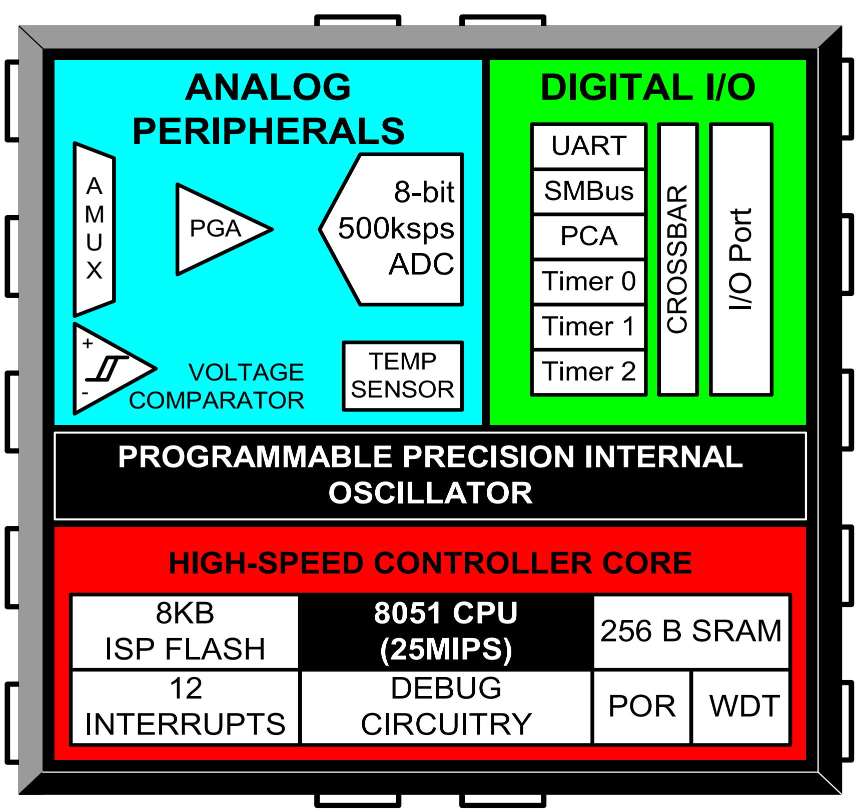
- 8 Port I/O; All 5 V tolerant with High Sink Current
- Hardware Enhanced UART and SMBus™ Serial Ports
- Three General Purpose 16-Bit Counter/Timers
- 16-Bit Programmable Counter Array (PCA) with Three Capture/Compare Modules
- Real Time Clock Mode using PCA or Timer and External Clock Source

CLOCK SOURCES

- Internal Oscillator: 24.5 MHz with ±2% Accuracy Supports UART Operation
- External Oscillator: Crystal, RC, C, or Clock (1 or 2 Pin Modes)
- Can Switch Between Clock Sources on-the-fly; Useful in Power Saving Modes

11-PIN MICRO LEAD PACKAGE

- 3x3mm PWB Footprint; Actual MLP Size:



Notes

TABLE OF CONTENTS

Notes

LIST OF FIGURES AND TABLES

1. SYSTEM OVERVIEW

C8051F300/1/2/3/4/5 devices are fully integrated mixed-signal System-on-a-Chip MCUs. Highlighted features are listed below. Refer to Table 1.1 on page 12 for specific product feature selection.

- High-speed pipelined 8051-compatible microcontroller core (up to 25 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- True 8-bit 500 ksps 11-channel ADC with programmable gain pre-amplifier and analog multiplexer
- Precision programmable 25 MHz internal oscillator
- 2/4/8k bytes of on-chip FLASH memory
- 256 bytes of on-chip RAM
- SMBus/I^{2}C and Enhanced UART serial interfaces implemented in hardware
- Three general-purpose 16-bit timers
- Programmable Counter/Timer Array (PCA) with three capture/compare modules and Watchdog Timer function
- On-chip Power-On Reset, VDD Monitor, and Temperature Sensor
- On-chip Voltage Comparator
- Byte-wide I/O Port (5V tolerant)

With on-chip Power-On Reset, VDD monitor, Watchdog Timer, and clock oscillator, the C8051F300/1/2/3/4/5 devices are truly stand-alone System-on-a-Chip solutions. The FLASH memory can be reprogrammed even in-circuit, providing non-volatile data storage, and also allowing field upgrades of the 8051 firmware. User software has complete control of all peripherals, and may individually shut down any or all peripherals for power savings.

The on-chip Cygnal 2-Wire (C2) Development Interface allows non-intrusive (uses no on-chip resources), full speed, in-circuit debugging using the production MCU installed in the final application. This debug logic supports inspection and modification of memory and registers, setting breakpoints, single stepping, run and halt commands. All analog and digital peripherals are fully functional while debugging using C2. The two C2 interface pins can be shared with user functions, allowing in-system debugging without occupying package pins.

Each device is specified for 2.7 V-to-3.6 V operation over the industrial temperature range (-45°C to +85°C). The Port I/O and /RST pins are tolerant of input signals up to 5 V. The C8051F300/1/2/3/4/5 are available in the 11-pin MLP package shown in Figure 4.2.

Table 1.1. Product Selection Guide

	MIPS (Peak)	FLASH Memory	RAM	Calibrated Internal Oscillator	SMBus/I^{2}C	UART	Timers (16-bit)	Programmable Counter Array	Digital Port I/Os	8-bit 500ksps ADC	Temperature Sensor	Analog Comparators	Package
C8051F300	25	8k	256	✓	✓	✓	3	✓	8	✓	✓	1	MLP-11
C8051F301	25	8k	256	✓	✓	✓	3	✓	8	-	-	1	MLP-11
C8051F302	25	8k	256	-	✓	✓	3	✓	8	✓	✓	1	MLP-11
C8051F303	25	8k	256	-	✓	✓	3	✓	8	-	-	1	MLP-11
C8051F304	25	4k	256	-	✓	✓	3	✓	8	-	-	1	MLP-11
C8051F305	25	2k	256	-	✓	✓	3	✓	8	-	-	1	MLP-11

Figure 1.1. C8051F300/2 Block Diagram

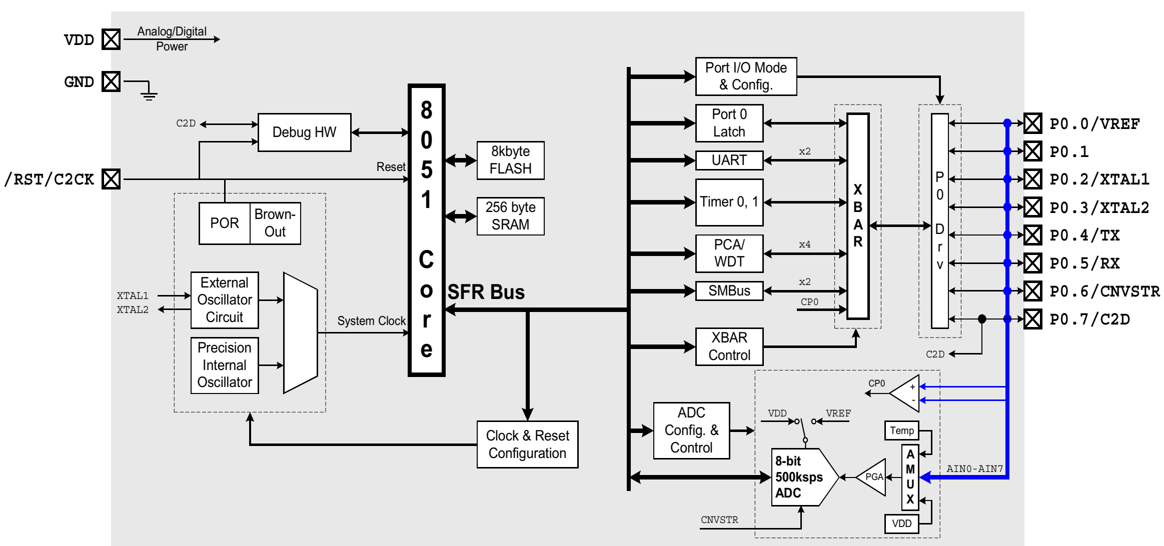
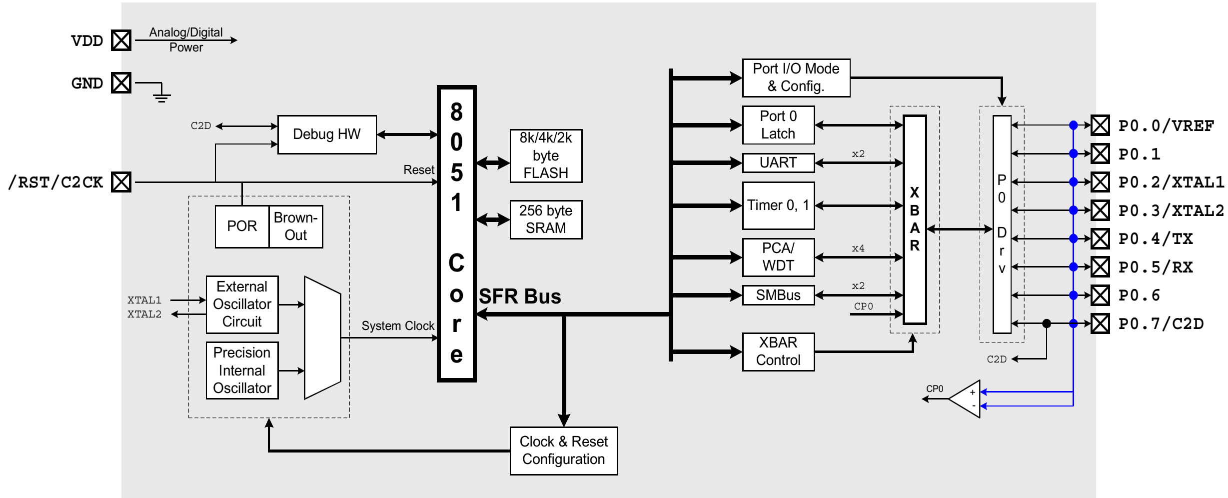


Figure 1.2. C8051F301/3/4/5 Block Diagram



1.1. CIP-51™ Microcontroller Core

1.1.1. Fully 8051 Compatible

The C8051F300/1/2/3/4/5 family utilizes Cygnal's proprietary CIP-51 microcontroller core. The CIP-51 is fully compatible with the MCS-51™ instruction set; standard 803x/805x assemblers and compilers can be used to develop software. The CIP-51 core offers all the peripherals included with a standard 8052, including two standard 16-bit counter/timers, one enhanced 16-bit counter/timer with external oscillator input, a full-duplex UART with extended baud rate configuration, 256 bytes of internal RAM, 128 byte Special Function Register (SFR) address space, and a byte-wide I/O Port.

1.1.2. Improved Throughput

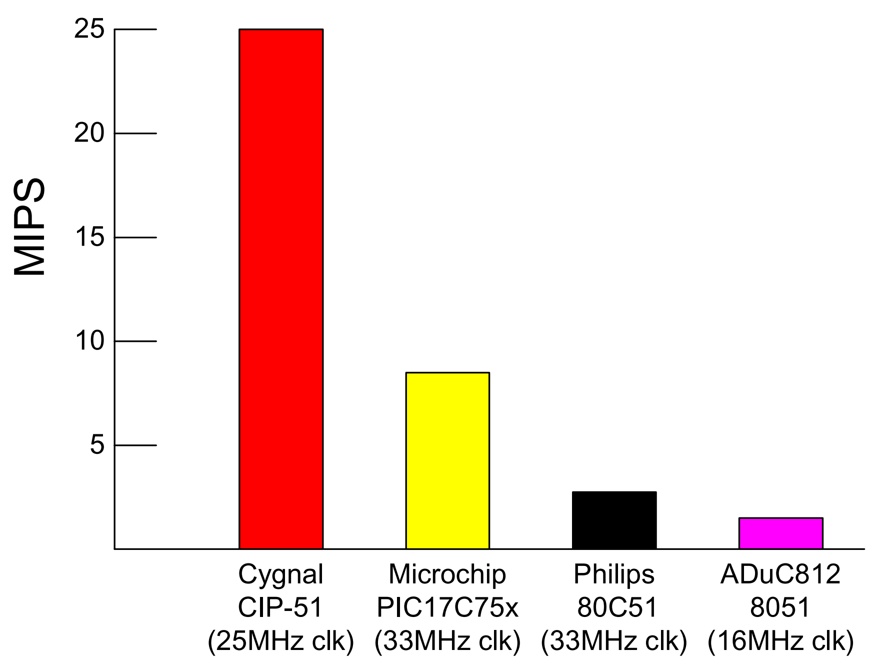
The CIP-51 employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. In a standard 8051, all instructions except for MUL and DIV take 12 or 24 system clock cycles to execute with a maximum system clock of 12-to-24 MHz. By contrast, the CIP-51 core executes 70% of its instructions in one or two system clock cycles, with only four instructions taking more than four system clock cycles.

The CIP-51 has a total of 109 instructions. The table below shows the total number of instrutions that require each execution time.

Clocks to Execute	1	2	2/3	3	3/4	4	4/5	5	8
Number of Instructions	26	50	5	14	7	3	1	2	1

With the CIP-51's maximum system clock at 25 MHz, it has a peak throughput of 25 MIPS. Figure 1.3 shows a comparison of peak throughputs for various 8-bit microcontroller cores with their maximum system clocks.

Figure 1.3. Comparison of Peak MCU Execution Speeds



1.1.3. Additional Features

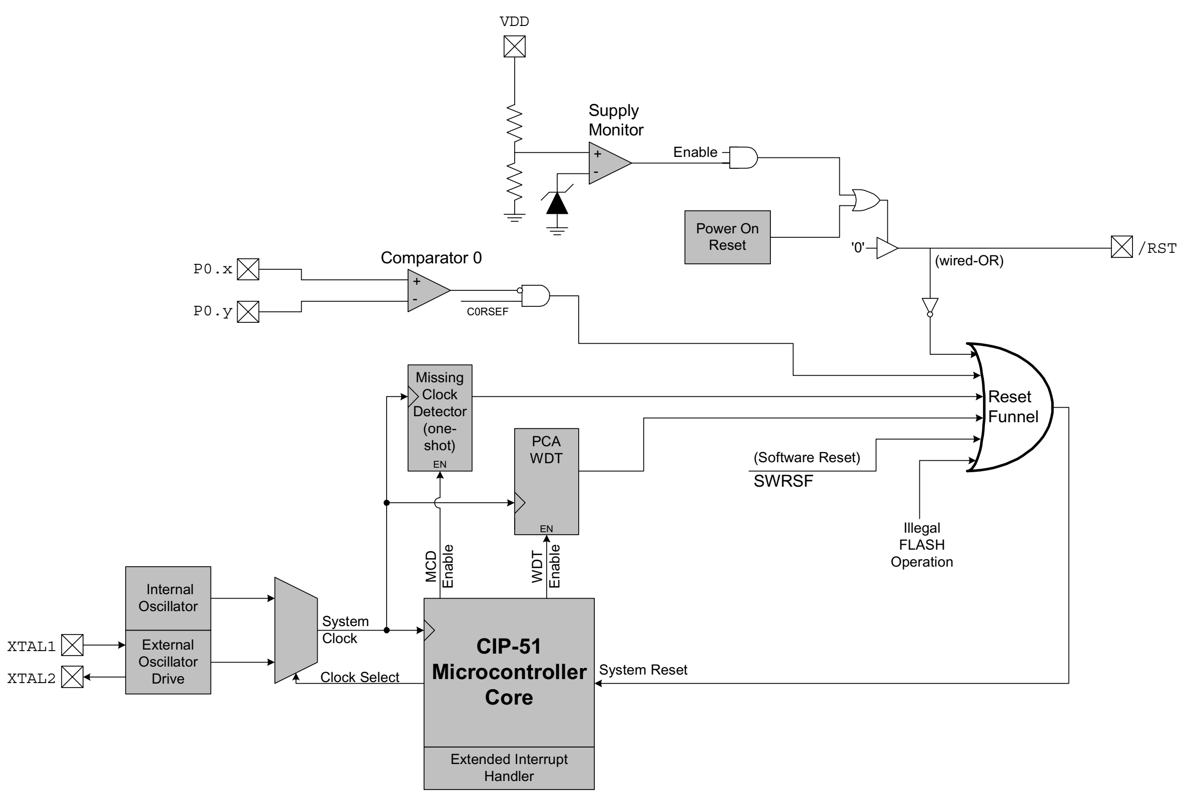
The C8051F300/1/2/3/4/5 SoC family includes several key enhancements to the CIP-51 core and peripherals to improve performance and ease of use in end applications.

The extended interrupt handler provides 12 interrupt sources into the CIP-51 (as opposed to 7 for the standard 8051), allowing numerous analog and digital peripherals to interrupt the controller. An interrupt driven system requires less intervention by the MCU, giving it more effective throughput. The extra interrupt sources are very useful when building multi-tasking, real-time systems.

Eight reset sources are available: power-on reset circuitry (POR), an on-chip VDD monitor (forces reset when power supply voltage drops below 2.7 V), a Watchdog Timer, a Missing Clock Detector, a voltage level detection from Comparator0, a forced software reset, an external reset pin, and an illegal FLASH read/write protection circuit. Each reset source except for the POR, Reset Input Pin, or FLASH protection may be disabled by the user in software. The WDT may be permanently enabled in software after a power-on reset during MCU initialization.

The internal oscillator is available as a factory calibrated 24.5 MHz ±2% (C8051F300/1 devices); an uncalibrated version is available on C8051F302/3/4/5 devices. On all C8051F300/1/2/3/4/5 devices, the internal oscillator period may be user programmed in ~0.5% increments. An external oscillator drive circuit is also included, allowing an external crystal, ceramic resonator, capacitor, RC, or CMOS clock source to generate the system clock. If desired, the system clock source may be switched on-the-fly to the external oscillator circuit. An external oscillator can be extremely useful in low power applications, allowing the MCU to run from a slow (power saving) external crystal source, while periodically switching to the fast (up to 25 MHz) internal oscillator as needed.

Figure 1.4. On-Chip Clock and Reset

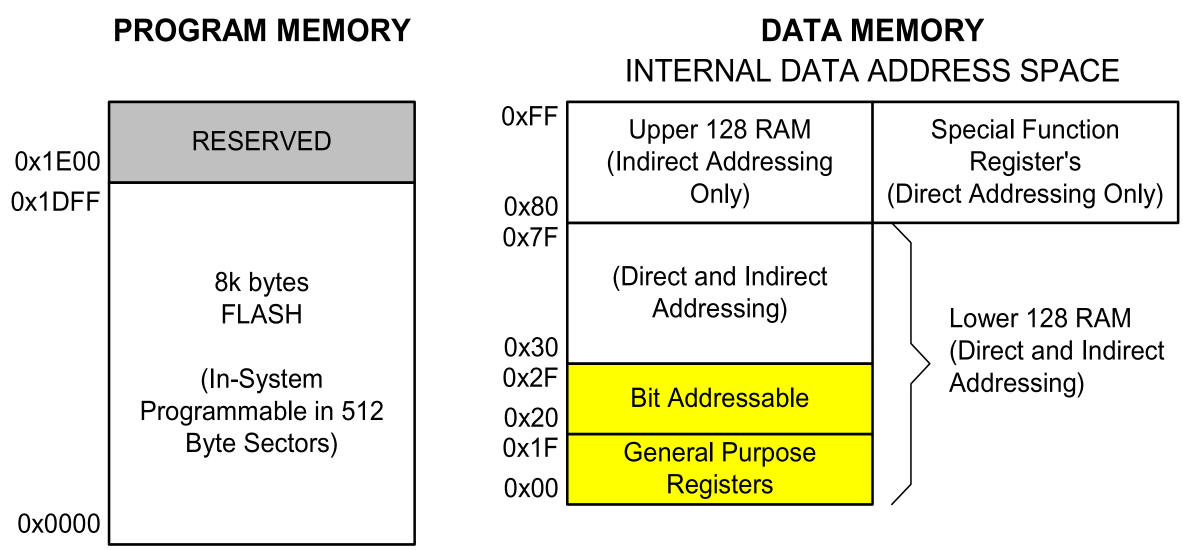


1.2. On-Chip Memory

The CIP-51 has a standard 8051 program and data address configuration. It includes 256 bytes of data RAM, with the upper 128 bytes dual-mapped. Indirect addressing accesses the upper 128 bytes of general purpose RAM, and direct addressing accesses the 128 byte SFR address space. The lower 128 bytes of RAM are accessible via direct and indirect addressing. The first 32 bytes are addressable as four banks of general purpose registers, and the next 16 bytes can be byte addressable or bit addressable.

The C8051F300/1/2/3 includes 8k bytes of FLASH program memory (the C8051F304 includes 4k bytes; the C8051F305 includes 2k bytes). This memory may be reprogrammed in-system in 512 byte sectors, and requires no special off-chip programming voltage. See Figure 1.5 for the C8051F300/1/2/3 system memory map.

Figure 1.5. On-chip Memory Map (C8051F300/1/2/3 shown)



1.3. On-Chip Debug Circuitry

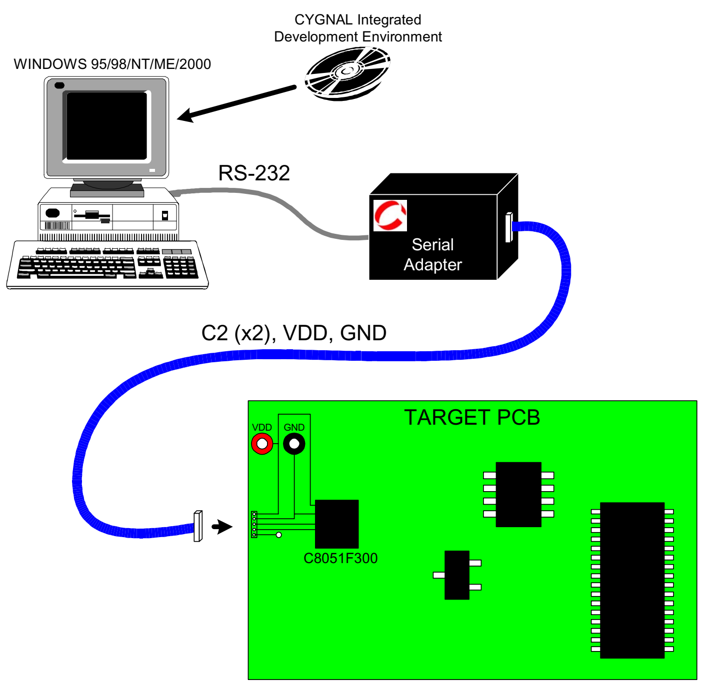
The C8051F300/1/2/3/4/5 devices include on-chip Cygnal 2-Wire (C2) debug circuitry that provides non-intrusive, full speed, in-circuit debugging of the production part *installed in the end application.*

Cygnal's debugging system supports inspection and modification of memory and registers, breakpoints, and single stepping. No additional target RAM, program memory, timers, or communications channels are required. All the digital and analog peripherals are functional and work correctly while debugging. All the peripherals (except for the ADC and SMBus) are stalled when the MCU is halted, during single stepping, or at a breakpoint in order to keep them synchronized.

The C8051F300DK development kit provides all the hardware and software necessary to develop application code and perform in-circuit debugging with the C8051F300/1/2/3/4/5 MCUs. The kit includes software with a developer's studio and debugger, an integrated 8051 assembler, and an RS-232 to C2 serial adapter. It also has a target application board with the associated MCU installed and large prototyping area, plus the RS-232 and C2 cables, and wall-mount power supply. The Development Kit requires a Windows 95/98/NT/ME/2000 computer with one available RS-232 serial port. As shown in Figure 1.6, the PC is connected via RS-232 to the Serial Adapter. A six-inch ribbon cable connects the Serial Adapter to the user's application board, picking up the two C2 pins and VDD and GND. The Serial Adapter takes its power from the application board; it requires roughly 20 mA at 2.7-3.6V. For applications where there is not sufficient power available from the target board, the provided power supply can be connected directly to the Serial Adapter.

The Cygnal IDE interface is a vastly superior developing and debugging configuration, compared to standard MCU emulators that use on-board "ICE Chips" and require the MCU in the application board to be socketed. Cygnal's debug paradigm increases ease of use and preserves the performance of the precision analog peripherals.

Figure 1.6. Development/In-System Debug Diagram

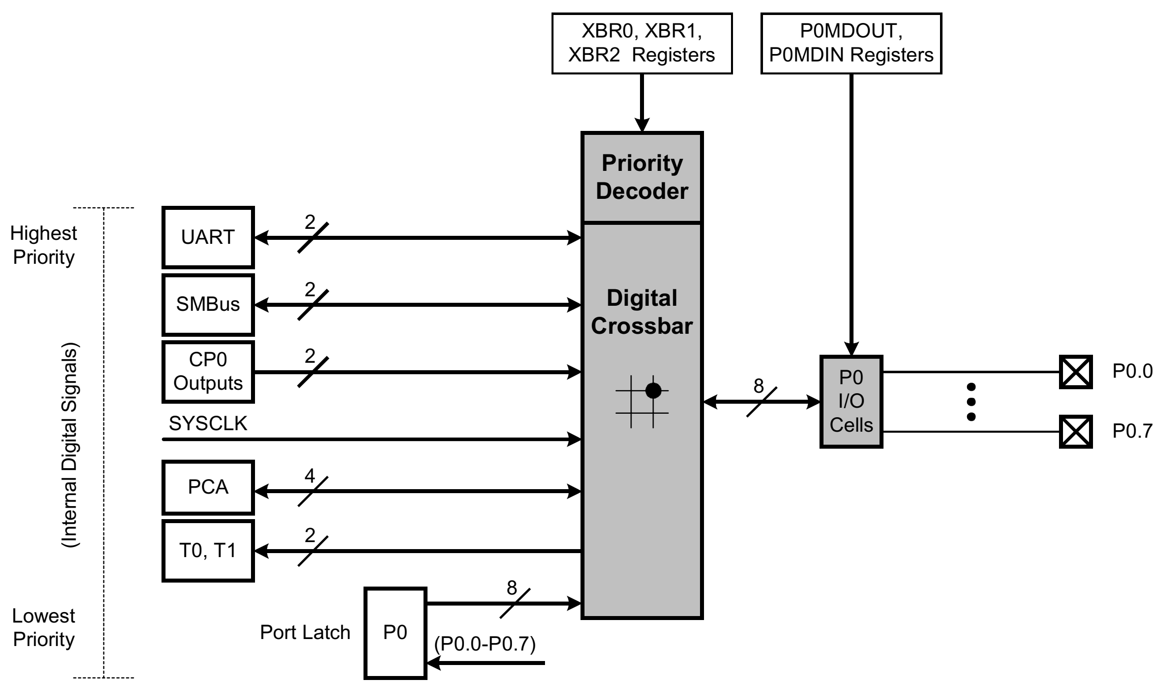


1.4. Programmable Digital I/O and Crossbar

C8051F300/1/2/3/4/5 devices include a byte-wide I/O Port that behaves like a typical 8051 Port with a few enhancements. Each Port pin may be configured as an analog input or a digital I/O pin. Pins selected as digital I/Os may additionally be configured for push-pull or open-drain output. The “weak pull-ups” that are fixed on typical 8051 devices may be globally disabled, providing power savings capabilities.

Perhaps the most unique Port I/O enhancement is the Digital Crossbar. This is essentially a digital switching network that allows mapping of internal digital system resources to Port I/O pins (See Figure 1.7). On-chip counter/timers, serial buses, HW interrupts, comparator output, and other digital signals in the controller can be configured to appear on the Port I/O pins specified in the Crossbar Control registers. This allows the user to select the exact mix of general purpose Port I/O and digital resources needed for the particular application.

Figure 1.7. Digital Crossbar Diagram



1.5. Serial Ports

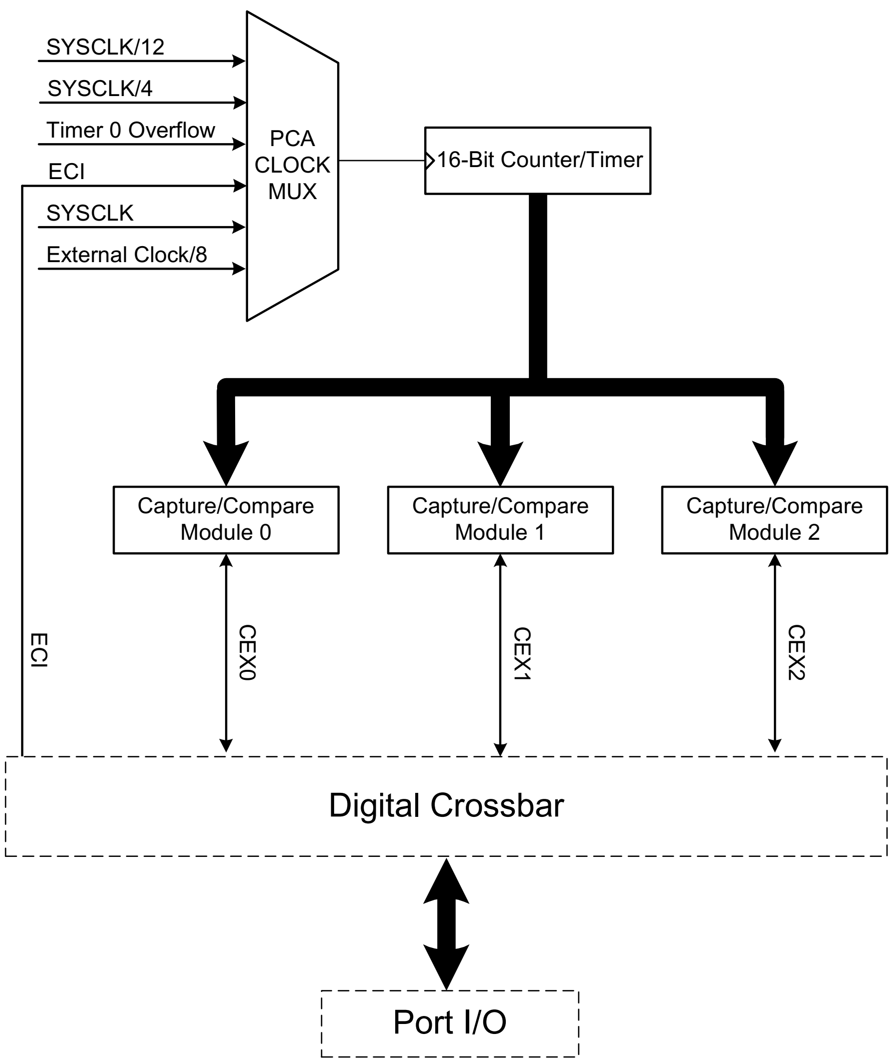
The C8051F300/1/2/3/4/5 Family includes an SMBus/I^2C interface and a full-duplex UART with enhanced baud rate configuration. Each of the serial buses is fully implemented in hardware and makes extensive use of the CIP-51's interrupts, thus requiring very little CPU intervention.

1.6. Programmable Counter Array

An on-chip Programmable Counter/Timer Array (PCA) is included in addition to the three 16-bit general purpose counter/timers. The PCA consists of a dedicated 16-bit counter/timer time base with three programmable capture/compare modules. The PCA clock is derived from one of six sources: the system clock divided by 12, the system clock divided by 4, Timer 0 overflows, an External Clock Input (ECI), the system clock, or the external oscillator clock source divided by 8. The external clock source selection is useful for real-time clock functionality, where the PCA is clocked by an external source while the internal oscillator drives the system clock.

Each capture/compare module can be configured to operate in one of six modes: Edge-Triggered Capture, Software Timer, High Speed Output, 8- or 16-bit Pulse Width Modulator, or Frequency Output. Additionally, Capture/Compare Module 2 offers watchdog timer (WDT) capabilities. Following a system reset, Module 2 is configured and enabled in WDT mode. The PCA Capture/Compare Module I/O and External Clock Input may be routed to Port I/O via the Digital Crossbar.

Figure 1.8. PCA Block Diagram



1.7. 8-Bit Analog to Digital Converter (C8051F300/2 Only)

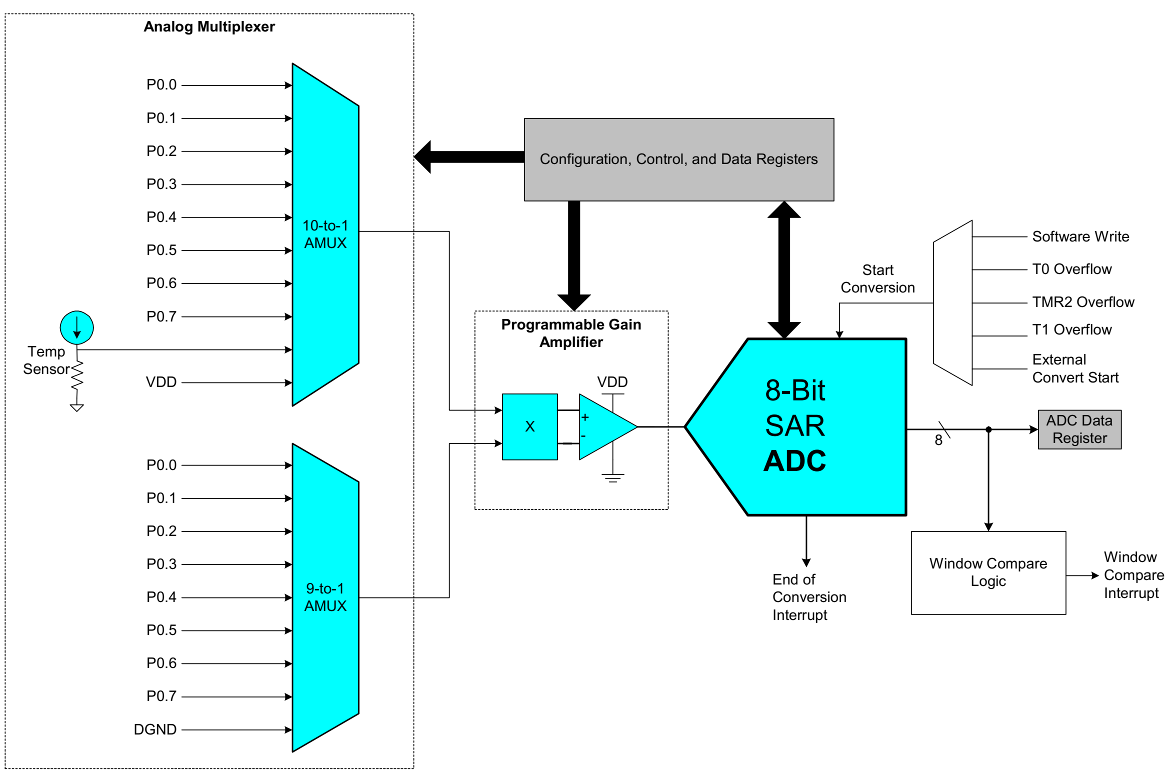
The C8051F300/2 includes an on-chip 8-bit SAR ADC with a 10-channel differential input multiplexer and programmable gain amplifier. With a maximum throughput of 500 ksps, the ADC offers true 8-bit accuracy with an INL of ±1LSB. The ADC system includes a configurable analog multiplexer that selects both positive and negative ADC inputs. Each Port pin is available as an ADC input; additionally, the on-chip Temperature Sensor output and the power supply voltage (VDD) are available as ADC inputs. User firmware may shut down the ADC to save power.

The integrated programmable gain amplifier (PGA) amplifies the the ADC input by 0.5, 1, 2, or 4 as defined by user software. The gain stage is especially useful when different ADC input channels have widely varied input voltage signals, or when it is necessary to "zoom in" on a signal with a large DC offset.

Conversions can be started in five ways: a software command, an overflow of Timer 0, 1, or 2, or an external convert start signal. This flexibility allows the start of conversion to be triggered by software events, a periodic signal (timer overflows), or external HW signals. Conversion completions are indicated by a status bit and an interrupt (if enabled). The resulting 8-bit data word is latched into an SFR upon completion of a conversion.

Window compare registers for the ADC data can be configured to interrupt the controller when ADC data is either within or outside of a specified range. The ADC can monitor a key voltage continuously in background mode, but not interrupt the controller unless the converted data is within/outside the specified range.

Figure 1.9. 8-Bit ADC Block Diagram

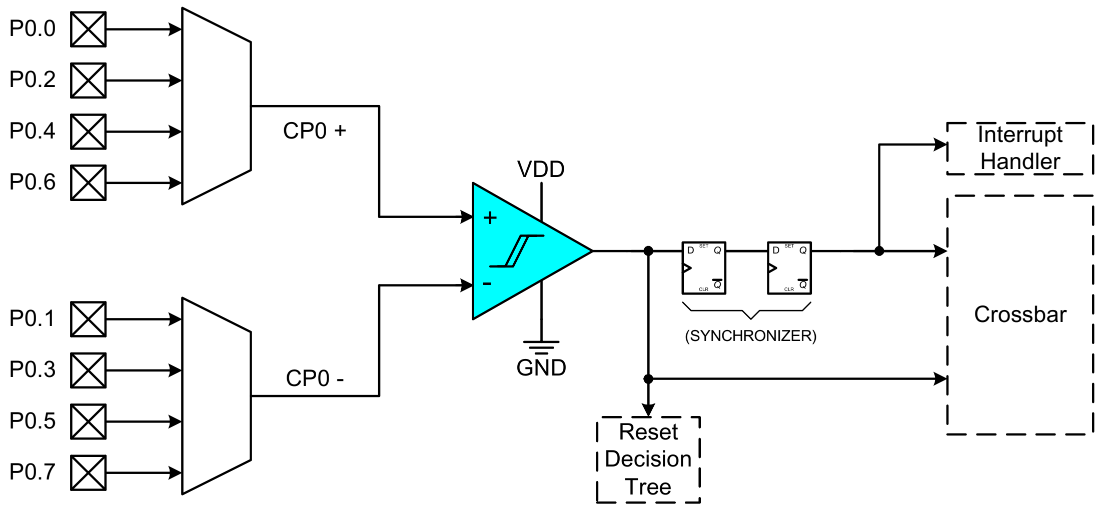


1.8. Comparator

C8051F300/1/2/3/4/5 devices include an on-chip voltage comparator that is enabled/disabled and configured via user software. All Port I/O pins may be configurated as comparator inputs. Two comparator outputs may be routed to a Port pin if desired: a latched output and/or an unlatched (asynchronous) output. Comparator response time is programmable, allowing the user to select between high-speed and low-power modes. Positive and negative hysteresis is also configurable.

Comparator interrupts may be generated on rising, falling, or both edges. When in IDLE mode, these interrupts may be used as a “wake-up” source. The comparator may also be configured as a reset source.

Figure 1.10. Comparator Block Diagram



2. ABSOLUTE MAXIMUM RATINGS

Table 2.1. Absolute Maximum Ratings*

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Ambient temperature under bias		-55		125	°C
Storage Temperature		-65		150	°C
Voltage on any Port I/O Pin or /RST with respect to GND		-0.3		5.8	V
Voltage on VDD with respect to GND		-0.3		4.2	V
Maximum Total current through VDD and GND				500	mA
Maximum output current sunk by /RST or any Port pin				100	mA

*Note: stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the devices at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

3. GLOBAL DC ELECTRICAL CHARACTERISTICS

Table 3.1. Global DC Electrical Characteristics

-40°C to +85°C, 25 MHz System Clock unless otherwise specified.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Digital Supply Voltage		2.7	3.0	3.6	V
Digital Supply Current with CPU active	VDD=2.7V, Clock=25MHz VDD=2.7V, Clock=1MHz VDD=2.7V, Clock=32kHz		5.8 0.34 12		mA mA µA
Digital Supply Current with CPU inactive (not accessing FLASH)	VDD=2.7V, Clock=25MHz VDD=2.7V, Clock=1MHz VDD=2.7V, Clock=32kHz		2.1 83 2.8		mA µA µA
Digital Supply Current (shut-down)	Oscillator not running		< 0.1		µA
Digital Supply RAM Data Retention Voltage			1.5		V
Specified Operating Temperature Range		-40		+85	°C
SYSCLK (system clock frequency)		0†		25	MHz
Tsysl (SYSCLK low time)		18			ns
Tsysh (SYSCLK high time)		18			ns

†SYSCLK must be at least 32 kHz to enable debugging.

4. PINOUT AND PACKAGE DEFINITIONS

Table 4.1. Pin Definitions for the C8051F300/1/2/3/4/5

Pin Number	Name	Type	Description
1	VREF / P0.0	A In D I/O or A In	External Voltage Reference Input. Port 0.0. See **Section 12** for complete description.
2	P0.1	D I/O or A In	Port 0.1. See **Section 12** for complete description.
3	VDD		Power Supply Voltage.
4	XTAL1 / P0.2	A In D I/O or A In	Crystal Input. This pin is the external oscillator circuit return for a crystal or ceramic resonator. See **Section 11.2**. Port 0.2. See **Section 12** for complete description.
5	XTAL2 / P0.3	A Out D I/O	Crystal Input/Output. For an external crystal or resonator, this pin is the excitation driver. This pin is the external clock input for CMOS, capacitor, or RC network configurations. See **Section 11.2**. Port 0.3. See **Section 12** for complete description.
6	P0.4	D I/O or A In	Port 0.4. See **Section 12** for complete description.
7	P0.5	D I/O or A In	Port 0.5. See **Section 12** for complete description.
8	C2CK / /RST	D I/O D I/O	Clock signal for the C2 Development Interface. Device Reset. Open-drain output of internal POR or VDD monitor. An external source can initiate a system reset by driving this pin low for at least 10 µs.
9	P0.6 / CNVSTR	D I/O or A In D I/O	Port 0.6. See **Section 12** for complete description. ADC External Convert Start Input Strobe.
10	C2D / P0.7	D I/O D I/O or A In	Data signal for the C2 Development Interface. Port 0.7. See **Section 12** for complete description.
11	GND		Ground.

Figure 4.1. MLP-11 Pinout Diagram (Top View)

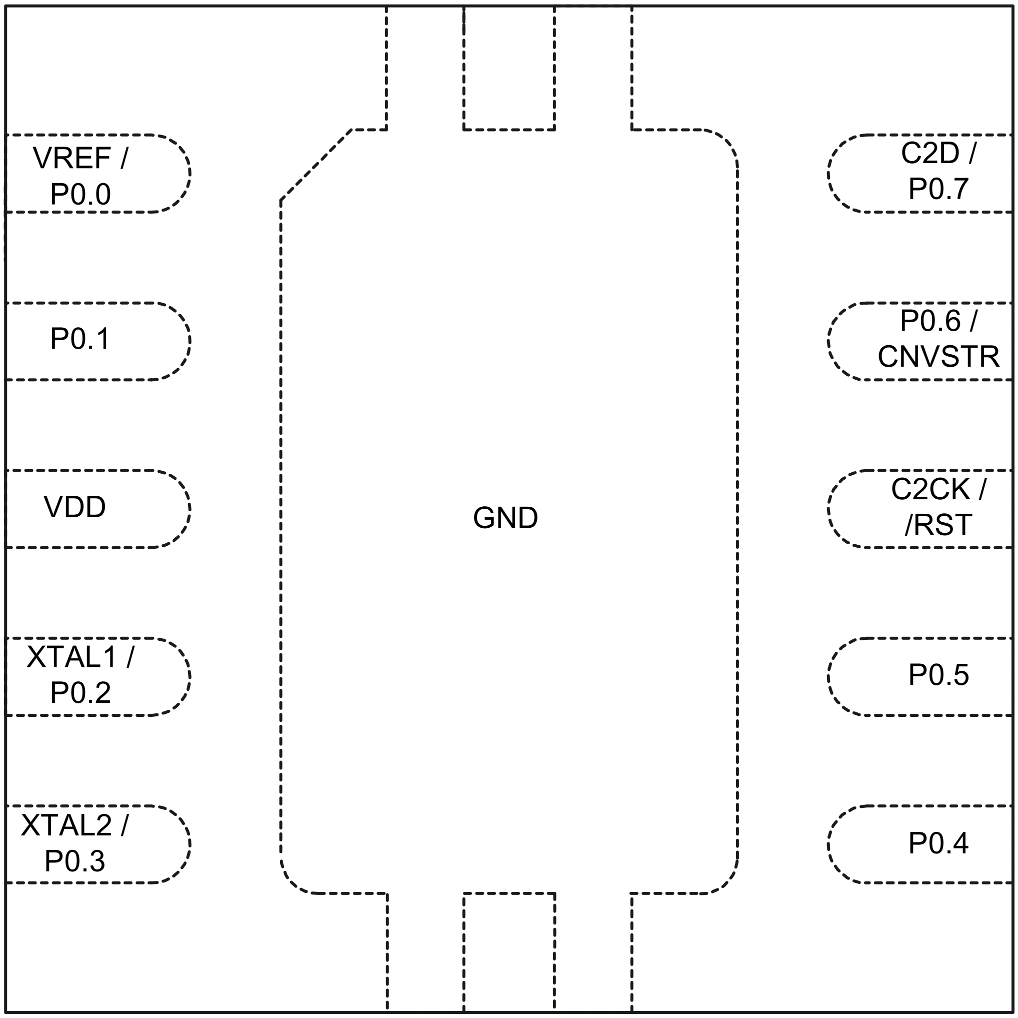


Figure 4.2. MLP-11 Package Drawing

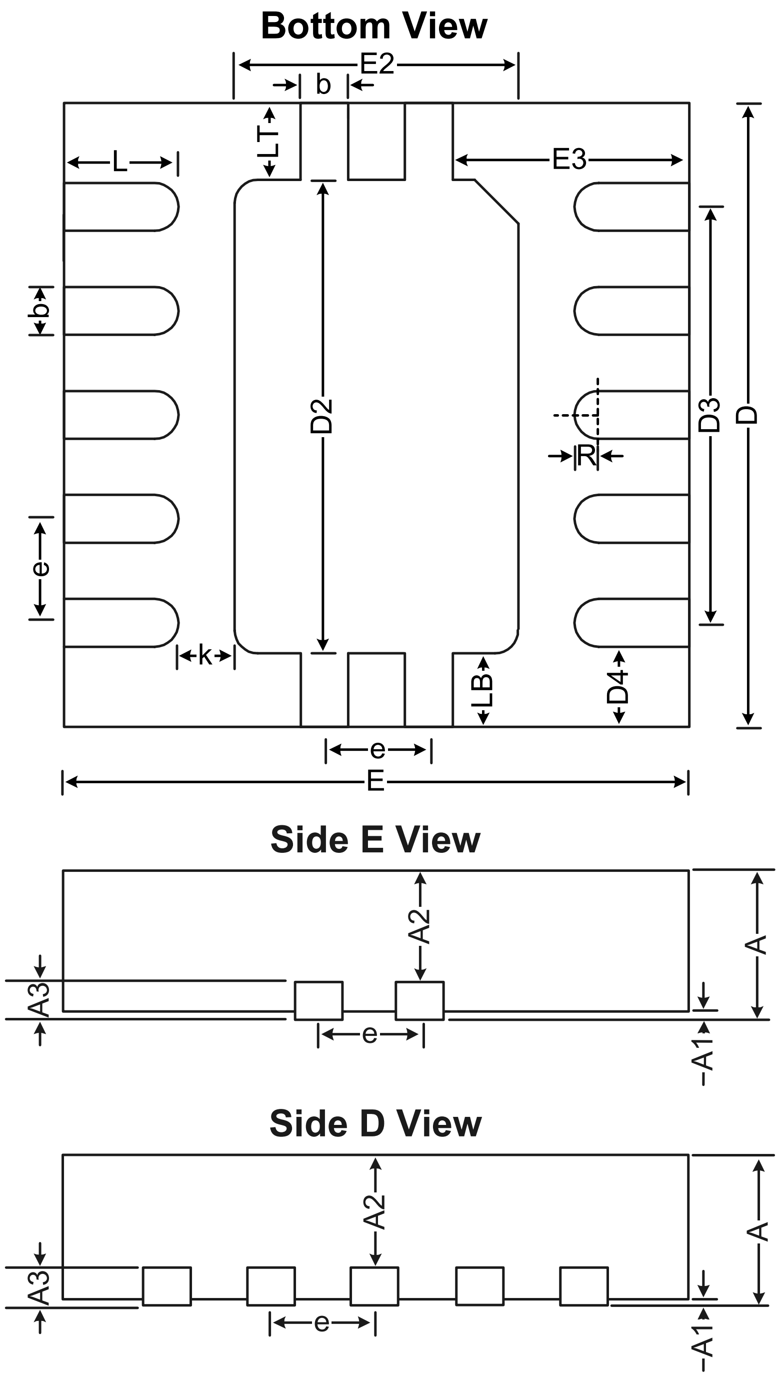


Table 4.2. MLP-11 Package Dimensions

	MM		
	MIN	TYP	MAX
A	0.80	0.90	1.00
A1	0	0.02	0.05
A2	0	0.65	1.00
A3		0.25	
b	0.18	0.23	0.30
D		3.00	
D2		2.20	2.25
D3		2.00	
D4		0.386	
E		3.00	
E2		1.36	
E3		1.135	
e		0.5	
k		0.27	
L	0.45	0.55	0.65
LB		0.36	
LT		0.37	
R	0.09		

Figure 4.3. Typical MLP-11 Solder Mask

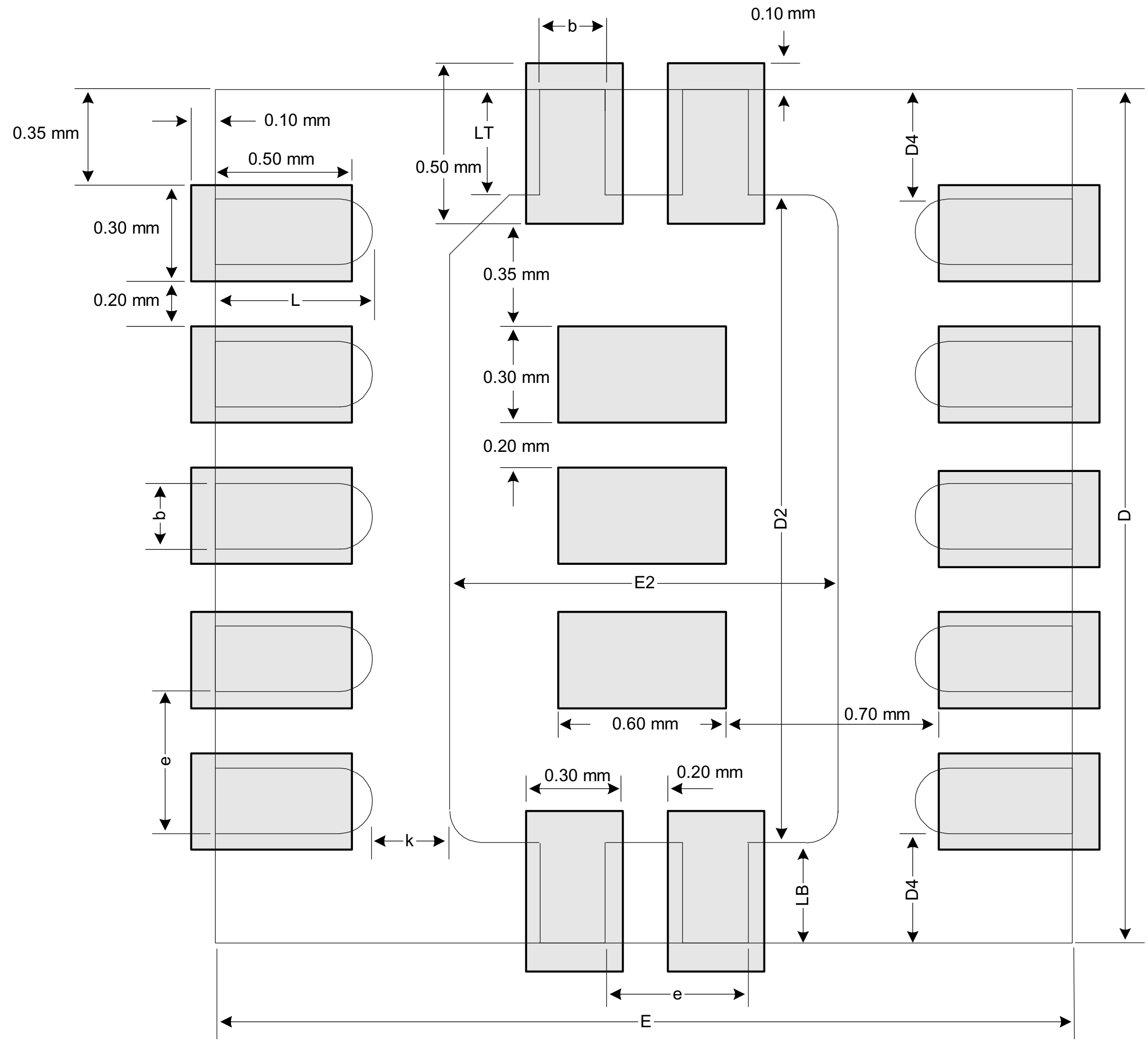
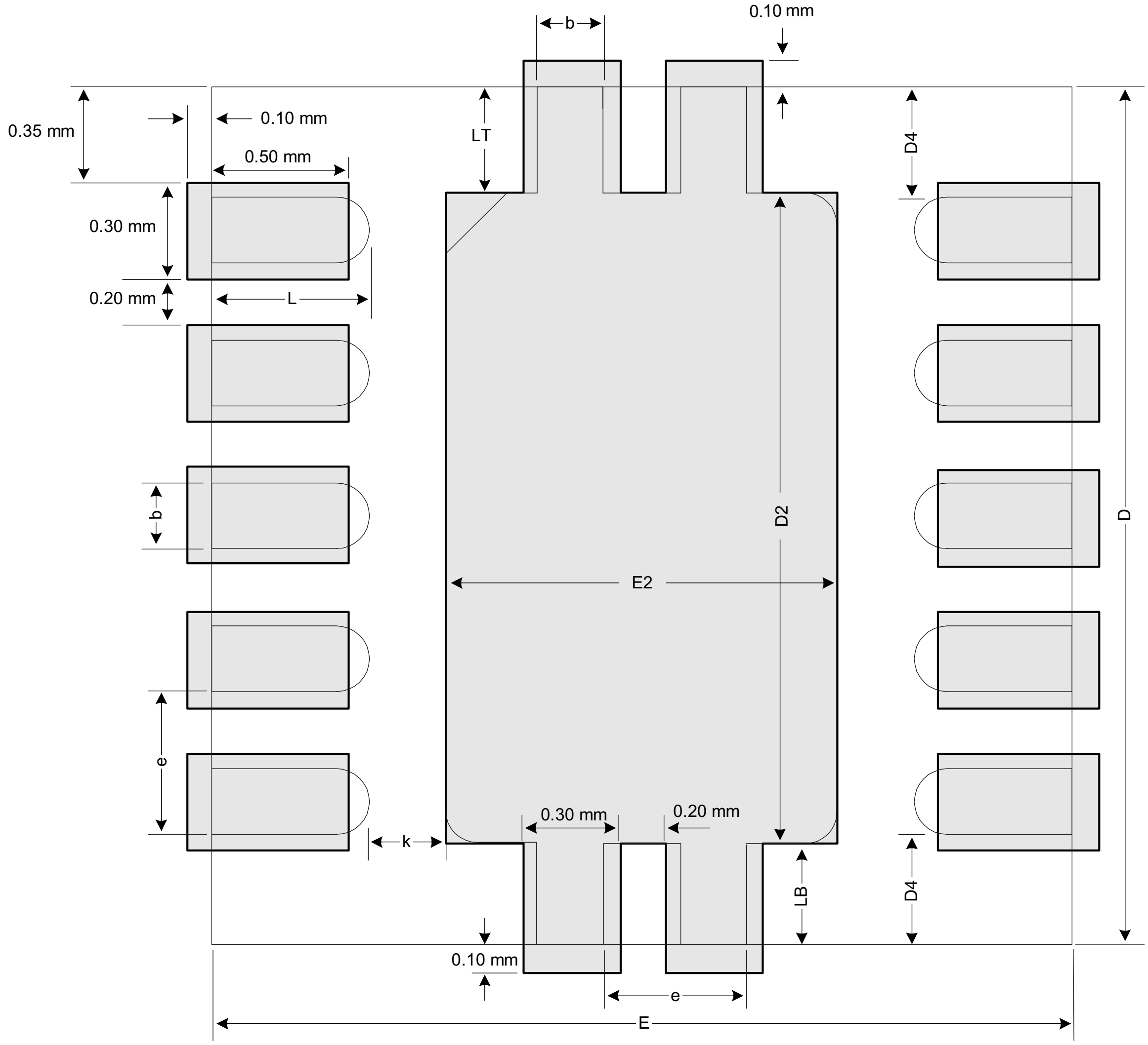


Figure 4.4. Typical MLP-11 Landing Diagram



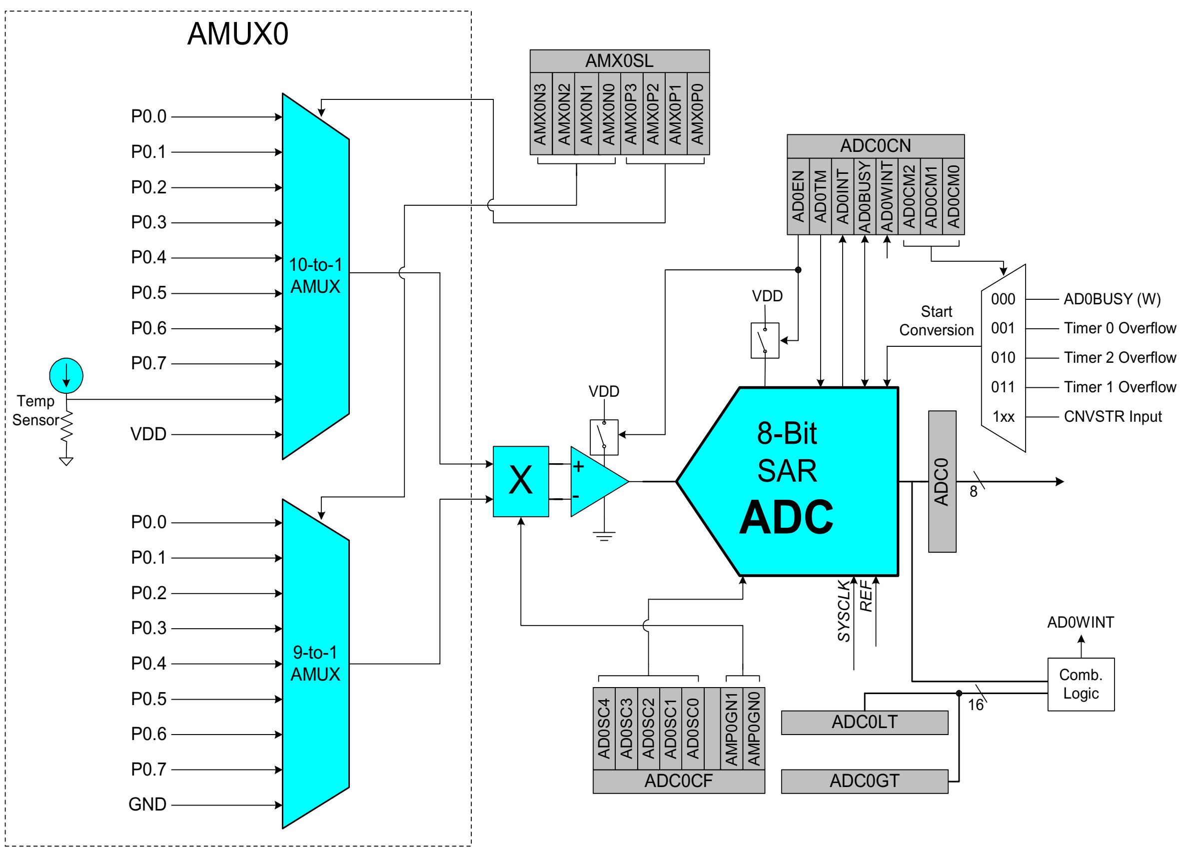
.

Notes

5. ADC0 (8-BIT ADC, C8051F300/2)

The ADC0 subsystem for the C8051F300/2 consists of two analog multiplexers (referred to collectively as AMUX0) with 11 total input selections, a differential programmable gain amplifier (PGA), and a 500 ksps, 8-bit successive-approximation-register ADC with integrated track-and-hold and programmable window detector (see block diagram in Figure 5.1). The AMUX0, PGA, data conversion modes, and window detector are all configurable under software control via the Special Function Registers shown in Figure 5.1. ADC0 operates in both Single-ended and Differential modes, and may be configured to measure any Port pin, the Temperature Sensor output, or VDD with respect to any Port pin or GND. The ADC0 subsystem is enabled only when the AD0EN bit in the ADC0 Control register (ADC0CN) is set to logic 1. The ADC0 subsystem is in low power shutdown when this bit is logic 0.

Figure 5.1. ADC0 Functional Block Diagram



5.1. Analog Multiplexer and PGA

The analog multiplexers (AMUX0) select the positive and negative inputs to the PGA, allowing any Port pin to be measured relative to any other Port pin or GND. Additionally, the on-chip temperature sensor or the positive power supply (VDD) may be selected as the positive PGA input. **When GND is selected as the negative input, ADC0 operates in Single-ended Mode; all other times, ADC0 operates in Differential Mode.** The ADC0 input channels are selected in the AMX0SL register as described in Figure 5.6.

The conversion code format differs in Single-ended versus Differential modes, as shown below. When in Single-ended Mode (negative input is selected GND), conversion codes are represented as 8-bit unsigned integers. Inputs are measured from ‘0’ to VREF * 255/256. Example codes are shown below.

Input Voltage	ADC0 Output (Conversion Code)
VREF * 255/256	0xFF
VREF * 128/256	0x80
VREF * 64/256	0x40
0	0x00

When in Differential Mode (negative input is not selected as GND), conversion codes are represented as 8-bit signed 2’s complement numbers. Inputs are measured from -VREF to VREF * 127/128. Example codes are shown below.

Input Voltage	ADC0 Output (Conversion Code)
VREF * 127/128	0x7F
VREF * 64/128	0x40
0	0x00
-VREF * 64/128	0xC0
- VREF	0x80

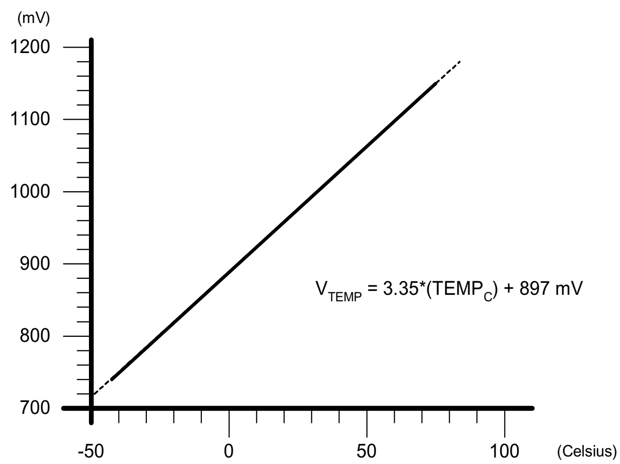
Important Note About ADC0 Input Configuration: Port pins selected as ADC0 inputs should be configured as analog inputs and should be skipped by the Digital Crossbar. To configure a Port pin for analog input, set to ‘0’ the corresponding bit in register P0MDIN. To force the Crossbar to skip a Port pin, set to ‘1’ the corresponding bit in register XBR0. See **Section “12. Port Input/Output” on page 95** for more Port I/O configuration details.

The PGA amplifies the AMUX0 output signal as defined by the AMP0GN1-0 bits in the ADC0 Configuration register (Figure 5.7). The PGA is software-programmable for gains of 0.5, 1, 2, or 4. The gain defaults to 0.5 on reset.

5.2. Temperature Sensor

The typical temperature sensor transfer function is shown in Figure 5.2. The output voltage (V_{TEMP}) is the positive PGA input when the temperature sensor is selected by bits AMX0P2-0 in register AMX0SL; this voltage will be amplified by the PGA according to the user-programmed PGA settings.

Figure 5.2. Typical Temperature Sensor Transfer Function

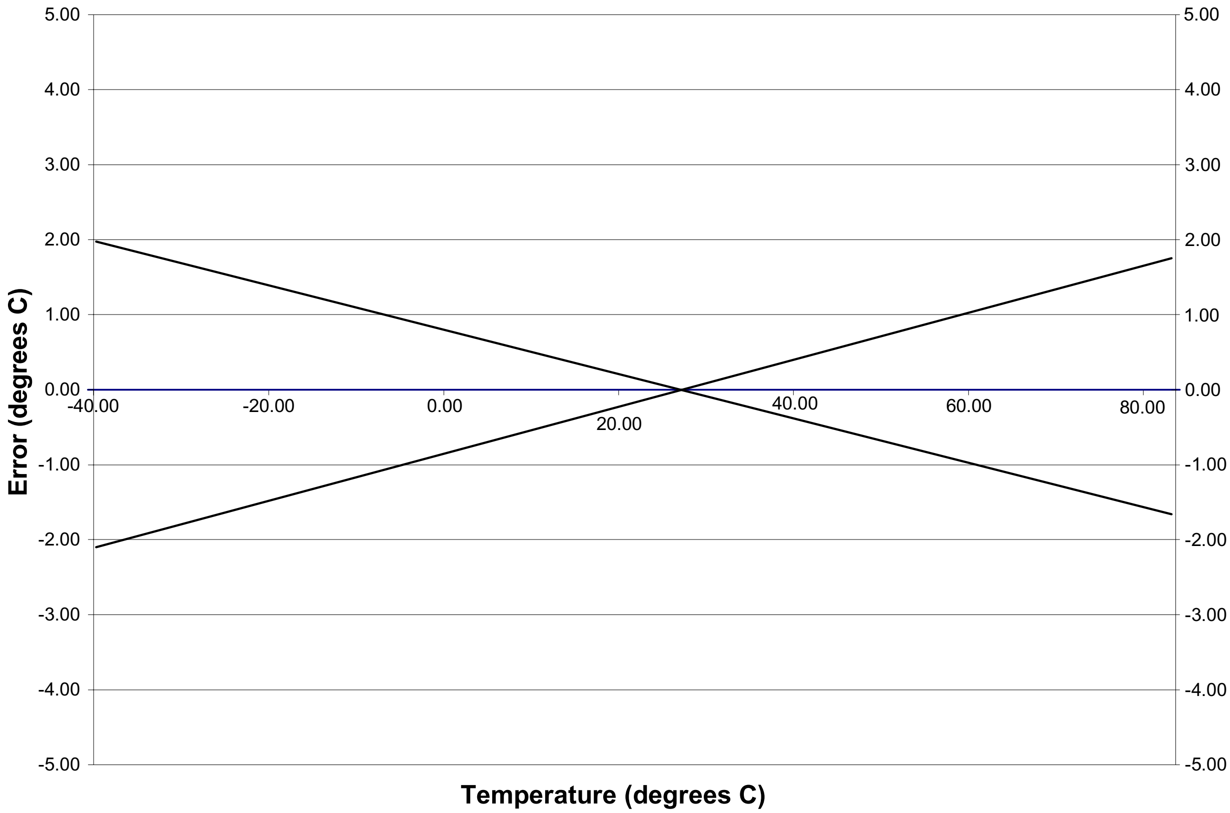


The uncalibrated temperature sensor output is extremely linear and suitable for relative temperature measurements (see Table 5.1 for linearity specifications). For absolute temperature measurements, gain and/or offset calibration is recommended. Typically a 1-point calibration includes the following steps:

Step 1. Control/measure the ambient temperature (this temperature must be known).
Step 2. Power the device, and delay for a few seconds to allow for self-heating.
Step 3. Perform an ADC conversion with the temperature sensor selected as the positive input and GND selected as the negative input.
Step 4. Calculate the offset and/or gain characteristics, and store these values in non-volatile memory for use with subsequent temperature sensor measurements.

Figure 5.3 shows the typical temperature sensor error assuming a 1-point calibration at 25 °C. **Note that parameters which affect ADC measurement, in particular the voltage reference value, will also affect temperature measurement.**

Figure 5.3. Temperature Sensor Error with 1-Point Calibration (VREF = 2.40 V)



5.3. Modes of Operation

ADC0 has a maximum conversion speed of 500 ksps. The ADC0 conversion clock is a divided version of the system clock, determined by the AD0SC bits in the ADC0CF register (system clock divided by (AD0SC + 1) for $0 \leq AD0SC \leq 31$).

5.3.1. Starting a Conversion

A conversion can be initiated in one of five ways, depending on the programmed states of the ADC0 Start of Conversion Mode bits (AD0CM2-0) in register ADC0CN. Conversions may be initiated by one of the following:

1. Writing a '1' to the AD0BUSY bit of register ADC0CN
2. A Timer 0 overflow (i.e. timed continuous conversions)
3. A Timer 2 overflow
4. A Timer 1 overflow
5. A rising edge on the CNVSTR input signal (pin P0.6)

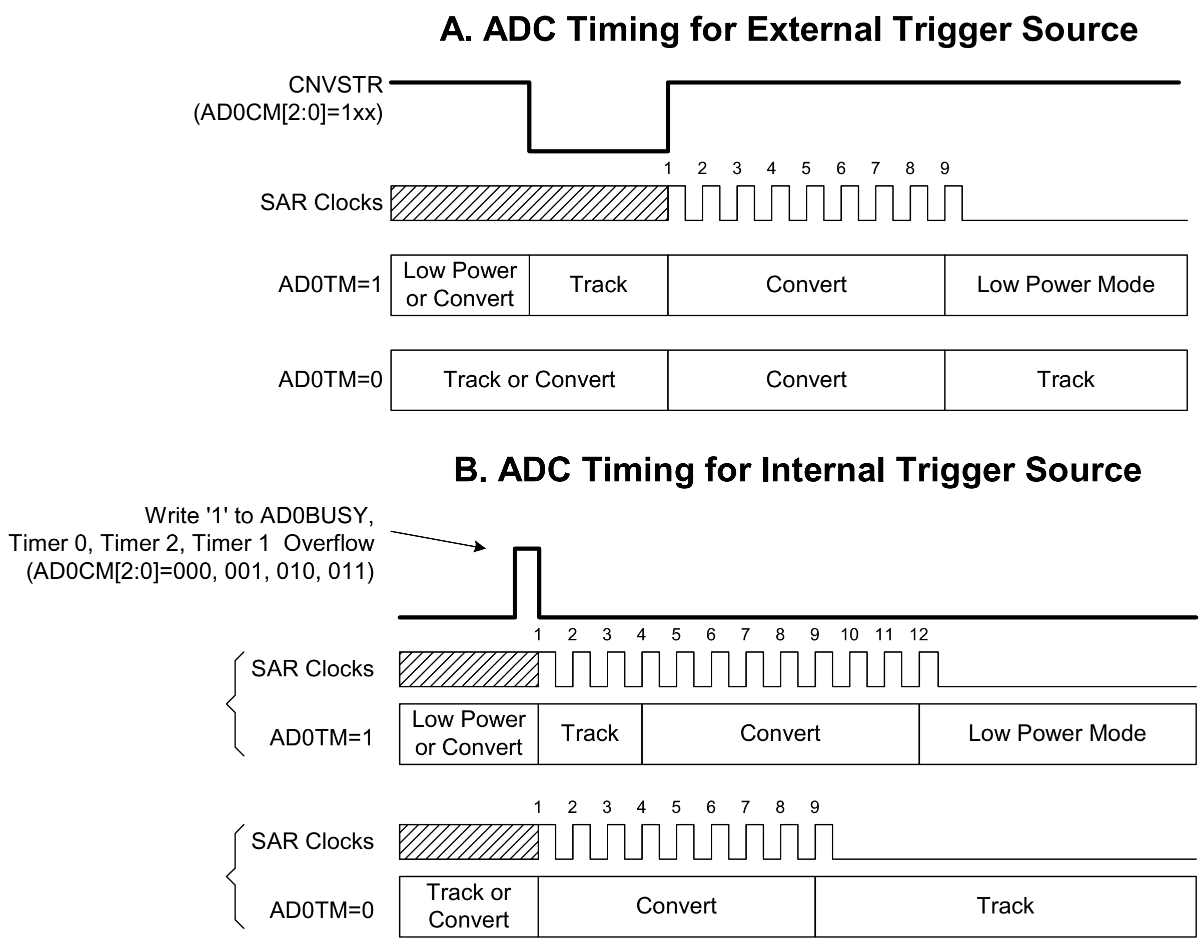
Writing a '1' to AD0BUSY provides software control of ADC0 whereby conversions are performed "on-demand". During conversion, the AD0BUSY bit is set to logic 1 and reset to logic 0 when the conversion is complete. The falling edge of AD0BUSY triggers an interrupt (when enabled) and sets the ADC0 interrupt flag (AD0INT). Note: When polling for ADC conversion completions, the ADC0 interrupt flag (AD0INT) should be used. Converted data is available in the ADC0 data register, ADC0, when bit AD0INT is logic 1. Note that when Timer 2 overflows are used as the conversion source, Timer 2 Low Byte overflows are used if Timer 2 is in 8-bit mode; Timer 2 High byte overflows are used if Timer 2 is in 16-bit mode. See **Section "15. Timers" on page 133** for timer configuration.

Important Note About Using CNVSTR: The CNVSTR input pin also functions as Port pin P0.6. When the CNVSTR input is used as the ADC0 conversion source, Port pin P0.6 should be skipped by the Digital Crossbar. To configure the Crossbar to skip P0.6, set to '1' Bit6 in register XBR0. See **Section "12. Port Input/Output" on page 95** for details on Port I/O configuration.

5.3.2. Tracking Modes

The AD0TM bit in register ADC0CN controls the ADC0 track-and-hold mode. In its default state, the ADC0 input is continuously tracked except when a conversion is in progress. When the AD0TM bit is logic 1, ADC0 operates in low-power track-and-hold mode. In this mode, each conversion is preceded by a tracking period of 3 SAR clocks (after the start-of-conversion signal). When the CNVSTR signal is used to initiate conversions in low-power tracking mode, ADC0 tracks only when CNVSTR is low; conversion begins on the rising edge of CNVSTR (see Figure 5.4). Tracking can also be disabled (shutdown) when the device is in low power standby or sleep modes. Low-power track-and-hold mode is also useful when AMUX or PGA settings are frequently changed, due to the settling time requirements described in **Section "5.3.3. Settling Time Requirements" on page 37**.

Figure 5.4. 8-Bit ADC Track and Conversion Example Timing



5.3.3. Settling Time Requirements

When the ADC0 input configuration is changed (i.e., a different AMUX0 or PGA selection is made), a minimum tracking time is required before an accurate conversion can be performed. This tracking time is determined by the AMUX0 resistance, the ADC0 sampling capacitance, any external source resistance, and the accuracy required for the conversion. Note that in low-power tracking mode, three SAR clocks are used for tracking at the start of every conversion. For most applications, these three SAR clocks will meet the minimum tracking time requirements.

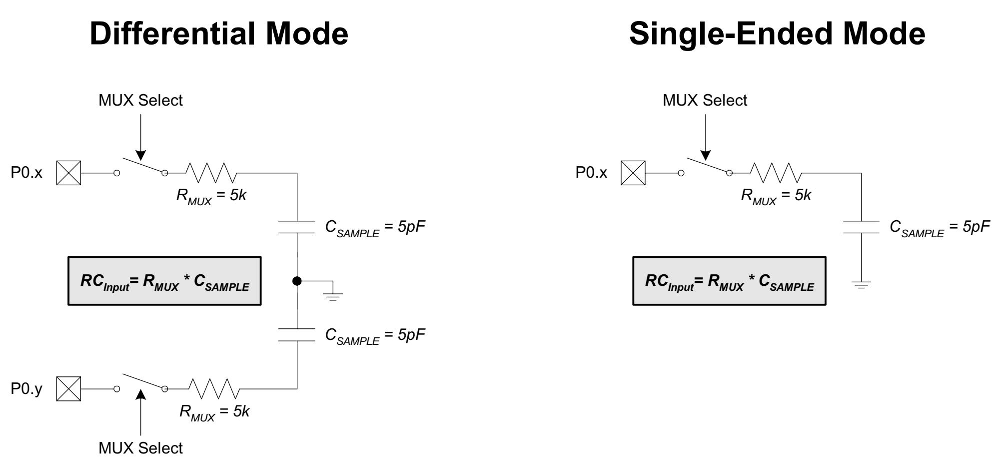
Figure 5.5 shows the equivalent ADC0 input circuits for both Differential and Single-ended modes. Notice that the equivalent time constant for both input circuits is the same. The required ADC0 settling time for a given settling accuracy (SA) may be approximated by Equation 5.1. When measuring the Temperature Sensor output or VDD with respect to GND, R_{TOTAL} reduces to R_{MUX}. See Table 5.1 for ADC0 minimum settling time (track/hold time) requirements.

Equation 5.1. ADC0 Settling Time Requirements

$$t = \ln\left(\frac{2^n}{SA}\right) \times R_{TOTAL}C_{SAMPLE}$$

Where:
SA is the settling accuracy, given as a fraction of an LSB (for example, 0.25 to settle within 1/4 LSB)
t is the required settling time in seconds
R_{TOTAL} is the sum of the AMUX0 resistance and any external source resistance.
n is the ADC resolution in bits (8).

Figure 5.5. ADC0 Equivalent Input Circuits



Note: When the PGA gain is set to 0.5, C_{SAMPLE} = 3pF

Figure 5.6. AMX0SL: AMUX0 Channel Select Register (C8051F300/2)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
AMX0N3	AMX0N2	AMX0N1	AMX0N0	AMX0P3	AMX0P2	AMX0P1	AMX0P0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xBB

Bits7-4: AMX0N3-0: AMUX0 Negative Input Selection.
Note that when GND is selected as the Negative Input, ADC0 operates in Single-ended mode. For all other Negative Input selections, ADC0 operates in Differential mode.
0000-1000b: ADC0 Negative Input selected per the chart below.

AMX0N3-0	ADC0 Negative Input
0000	P0.0
0001	P0.1
0010	P0.2
0011	P0.3
0100	P0.4
0101	P0.5
0110	P0.6
0111	P0.7
1xxx	GND (ADC in Single-Ended Mode)

Bits3-0: AMX0P3-0: AMUX0 Positive Input Selection.
0000-1001b: ADC0 Positive Input selected per the chart below.
1010-1111b: RESERVED.

AMX0P3-0	ADC0 Positive Input
0000	P0.0
0001	P0.1
0010	P0.2
0011	P0.3
0100	P0.4
0101	P0.5
0110	P0.6
0111	P0.7
1000	Temperature Sensor
1001	VDD

Figure 5.7. ADC0CF: ADC0 Configuration Register (C8051F300/2)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
AD0SC4	AD0SC3	AD0SC2	AD0SC1	AD0SC0	-	AMP0GN1	AMP0GN0	11111000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xBC

Bits7-3: AD0SC4-0: ADC0 SAR Conversion Clock Period Bits.
SAR Conversion clock is derived from system clock by the following equation, where *AD0SC* refers to the 5-bit value held in bits AD0SC4-0. SAR Conversion clock requirements are given in Table 5.1.

$$AD0SC = \frac{SYSCLK}{CLK_{SAR}} - 1$$

Bit2: UNUSED. Read = 0b; Write = don't care.
Bits1-0: AMP0GN1-0: ADC0 Internal Amplifier Gain (PGA).
00: Gain = 0.5
01: Gain = 1
10: Gain = 2
11: Gain = 4

Figure 5.8. ADC0: ADC0 Data Word Register (C8051F300/2)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xBE

Bits7-0: ADC0 Data Word.
ADC0 holds the output data byte from the last ADC0 conversion. When in Single-ended mode, ADC0 holds an 8-bit unsigned integer. When in Differential mode, ADC0 holds a 2's complement signed 8-bit integer.

Figure 5.9. ADC0CN: ADC0 Control Register (C8051F300/2)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
AD0EN	AD0TM	AD0INT	AD0BUSY	AD0WINT	AD0CM2	AD0CM1	AD0CM0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xE8

Bit7: AD0EN: ADC0 Enable Bit.
0: ADC0 Disabled. ADC0 is in low-power shutdown.
1: ADC0 Enabled. ADC0 is active and ready for data conversions.

Bit6: AD0TM: ADC0 Track Mode Bit.
0: Normal Track Mode: When ADC0 is enabled, tracking is continuous unless a conversion is in progress.
1: Low-power Track Mode: Tracking Defined by AD0CM2-0 bits (see below).

Bit5: AD0INT: ADC0 Conversion Complete Interrupt Flag.
0: ADC0 has not completed a data conversion since the last time AD0INT was cleared.
1: ADC0 has completed a data conversion.

Bit4: AD0BUSY: ADC0 Busy Bit.
Read: Unused.
Write:
0: No Effect.
1: Initiates ADC0 Conversion if AD0CM2-0 = 000b

Bit3: AD0WINT: ADC0 Window Compare Interrupt Flag.
0: ADC0 Window Comparison Data match has not occurred since this flag was last cleared.
1: ADC0 Window Comparison Data match has occurred.

Bits2-0: AD0CM2-0: ADC0 Start of Conversion Mode Select.
When AD0TM = 0:
000: ADC0 conversion initiated on every write of ‘1’ to AD0BUSY.
001: ADC0 conversion initiated on overflow of Timer 0.
010: ADC0 conversion initiated on overflow of Timer 2.
011: ADC0 conversion initiated on overflow of Timer 1.
1xx: ADC0 conversion initiated on rising edge of external CNVSTR.
When AD0TM = 1:
000: Tracking initiated on write of ‘1’ to AD0BUSY and lasts 3 SAR clocks, followed by conversion.
001: Tracking initiated on overflow of Timer 0 and lasts 3 SAR clocks, followed by conversion.
010: Tracking initiated on overflow of Timer 2 and lasts 3 SAR clocks, followed by conversion.
011: Tracking initiated on overflow of Timer 1 and lasts 3 SAR clocks, followed by conversion.
1xx: ADC0 tracks only when CNVSTR input is logic low; conversion starts on rising CNVSTR edge.

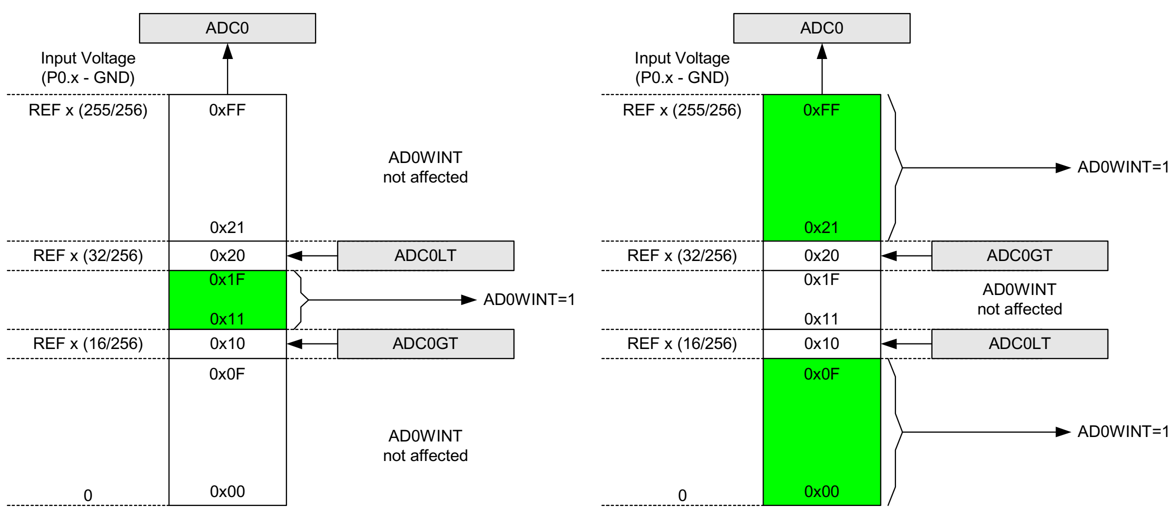
5.4. Programmable Window Detector

The ADC Programmable Window Detector continuously compares the ADC0 output to user-programmed limits, and notifies the system when a desired condition is detected. This is especially effective in an interrupt-driven system, saving code space and CPU bandwidth while delivering faster system response times. The window detector interrupt flag (AD0WINT in register ADC0CN) can also be used in polled mode. The ADC0 Greater-Than (ADC0GT) and Less-Than (ADC0LT) registers hold the comparison values. Example comparisons for Single-ended and Differential modes are shown in Figure 5.10 and Figure 5.11, respectively. Notice that the window detector flag can be programmed to indicate when measured data is inside or outside of the user-programmed limits depending on the contents of the ADC0LT and ADC0GT registers.

5.4.1. Window Detector In Single-Ended Mode

Figure 5.10 shows two example window comparisons for Single-ended mode, with ADC0LT = 0x20 and ADC0GT = 0x10. Notice that in Single-ended mode, the codes vary from 0 to VREF*(255/256) and are represented as 8-bit unsigned integers. In the left example, an AD0WINT interrupt will be generated if the ADC0 conversion word (ADC0) is within the range defined by ADC0GT and ADC0LT (if 0x10 < ADC0 < 0x20). In the right example, and AD0WINT interrupt will be generated if ADC0 is outside of the range defined by ADC0GT and ADC0LT (if ADC0 < 0x10 or ADC0 > 0x20).

Figure 5.10. ADC Window Compare Examples, Single-Ended Mode



5.4.2. Window Detector In Differential Mode

Figure 5.11 shows two example window comparisons for differential mode, with ADC0LT = 0x10 (+16d) and ADC0GT = 0xFF (-1d). Notice that in Differential mode, the codes vary from -VREF to VREF*(127/128) and are represented as 8-bit 2's complement signed integers. In the left example, an AD0WINT interrupt will be generated if the ADC0 conversion word (ADC0L) is within the range defined by ADC0GT and ADC0LT (if 0xFF (-1d) < ADC0 < 0x10 (16d)). In the right example, an AD0WINT interrupt will be generated if ADC0 is outside of the range defined by ADC0GT and ADC0LT (if ADC0 < 0xFF (-1d) or ADC0 > 0x10 (+16d)).

Figure 5.11. ADC Window Compare Examples, Differential Mode

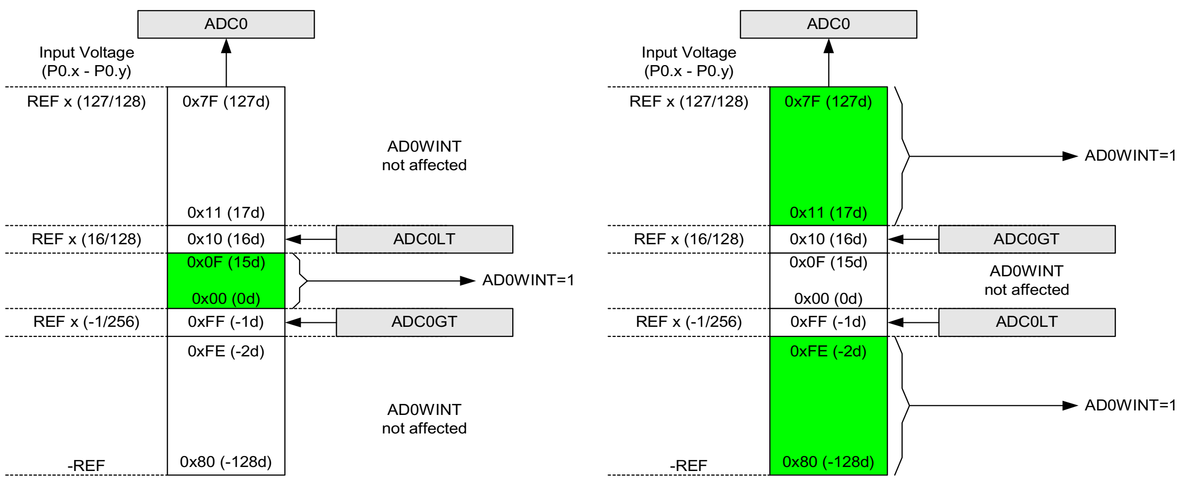


Figure 5.12. ADC0GT: ADC0 Greater-Than Data Byte Register (C8051F300/2)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								11111111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xC4

Bits7-0: ADC0 Greater-Than Data Word.

Figure 5.13. ADC0LT: ADC0 Less-Than Data Byte Register (C8051F300/2)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xC6

Bits7-0: ADC0 Less-Than Data Word.

Table 5.1. ADC0 Electrical Characteristics

VDD = 3.0 V, VREF = 2.40 V (REFSL=0), PGA Gain = 1, -40°C to +85°C unless otherwise specified

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
DC ACCURACY					
Resolution			8		bits
Integral Nonlinearity			±0.5	±1	LSB
Differential Nonlinearity	Guaranteed Monotonic		±0.5	±1	LSB
Offset Error	Note 1		0.5±0.6		LSB
Full Scale Error	Differential mode; See Note 1		-1±0.5		LSB
Offset Temperature Coefficient			TBD		ppm/°C
DYNAMIC PERFORMANCE (10 kHz sine-wave Differential input, 1 dB below Full Scale, 500 ksps)					
Signal-to-Noise Plus Distortion		45	48		dB
Total Harmonic Distortion	Up to the 5^{th} harmonic		-56		dB
Spurious-Free Dynamic Range			58		dB
CONVERSION RATE					
SAR Conversion Clock				6	MHz
Conversion Time in SAR Clocks		8			clocks
Track/Hold Acquisition Time		300			ns
Throughput Rate				500	ksps
ANALOG INPUTS					
Input Voltage Range		0		VREF	V
Input Capacitance			5		pF
TEMPERATURE SENSOR					
Linearity	Notes 1, 2, 3		±0.5		°C
Gain	Notes 1, 2, 3		3350 ±110		µV / °C
Offset	Notes 1, 2, 3 (Temp = 0 °C)		897±31		mV
POWER SPECIFICATIONS					
Power Supply Current (VDD supplied to ADC0)	Operating Mode, 500 ksps		400	900	µA
Power Supply Rejection			±0.3		mV/V

Note 1: Represents one standard deviation from the mean.
Note 2: Measured with PGA Gain = 2.
Note 3: Includes ADC offset, gain, and linearity variations.

6. VOLTAGE REFERENCE (C8051F300/2)

The voltage reference MUX on C8051F300/2 devices is configurable to use an externally connected voltage reference or the power supply voltage, VDD (see Figure 6.1). The REFSL bit in the Reference Control register (REF0CN) selects the reference source. For an external source, REFSL should be set to ‘0’; For VDD as the reference source, REFSL should be set to ‘1’.

The BIASE bit enables the internal voltage bias generator, which is used by the ADC, Temperature Sensor, and Internal Oscillator. This bit is forced to logic 1 when any of the aforementioned peripherals is enabled. The bias generator may be enabled manually by writing a ‘1’ to the BIASE bit in register REF0CN; see Figure 6.2 for REF0CN register details. The electrical specifications for the voltage reference circuit are given in Table 6.1.

Important Note About the VREF Input: Port pin P0.0 is used as the external VREF input. When using an external voltage reference, P0.0 should be configured as analog input and skipped by the Digital Crossbar. To configure P0.0 as analog input, set to ‘1’ Bit0 in register P0MDIN. To configure the Crossbar to skip P0.0, set to ‘1’ Bit0 in register XBR0. Refer to **Section “12. Port Input/Output” on page 95** for complete Port I/O configuration details. The external reference voltage must be within the range 0 ≤ VREF ≤ VDD.

On C8051F300/2 devices, the temperature sensor connects to the highest order input of the ADC0 positive input multiplexer (see **Section “5.1. Analog Multiplexer and PGA” on page 32** for details). The TEMPE bit in register REF0CN enables/disables the temperature sensor. While disabled, the temperature sensor defaults to a high impedance state and any ADC0 measurements performed on the sensor result in meaningless data.

Figure 6.1. Voltage Reference Functional Block Diagram

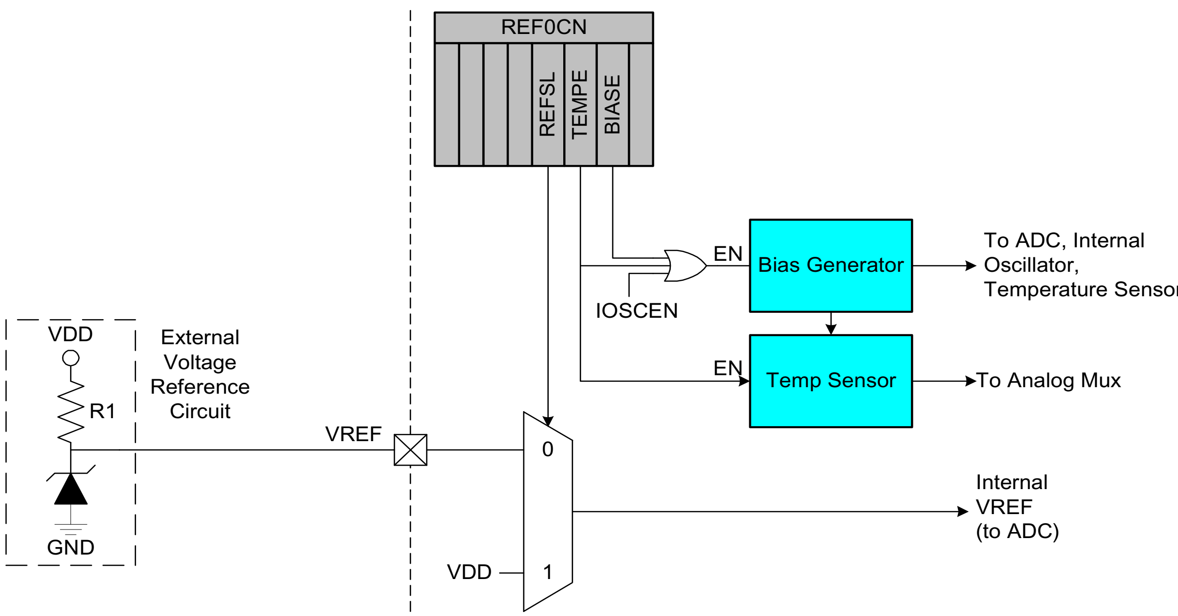


Figure 6.2. REF0CN: Reference Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	-	-	REFSL	TEMPE	BIASE	-	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xD1

Bits7-3: UNUSED. Read = 00000b; Write = don't care.
Bit3: REFSL: Voltage Reference Select.
This bit selects the source for the internal voltage reference.
0: VREF input pin used as voltage reference.
1: VDD used as voltage reference.
Bit2: TEMPE: Temperature Sensor Enable Bit.
0: Internal Temperature Sensor off.
1: Internal Temperature Sensor on.
Bit1: BIASE: Internal Analog Bias Generator Enable Bit. (Must be '1' if using ADC).
0: Internal Bias Generator off.
1: Internal Bias Generator on.
Bit0: UNUSED. Read = 0b. Write = don't care.

Table 6.1. External Voltage Reference Circuit Electrical Characteristics

VDD = 3.0 V; -40°C to +85°C unless otherwise specified

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Input Voltage Range		0		VDD	V
Input Current	Sample Rate = 500 ksps; VREF = 3.0 V		12		µA

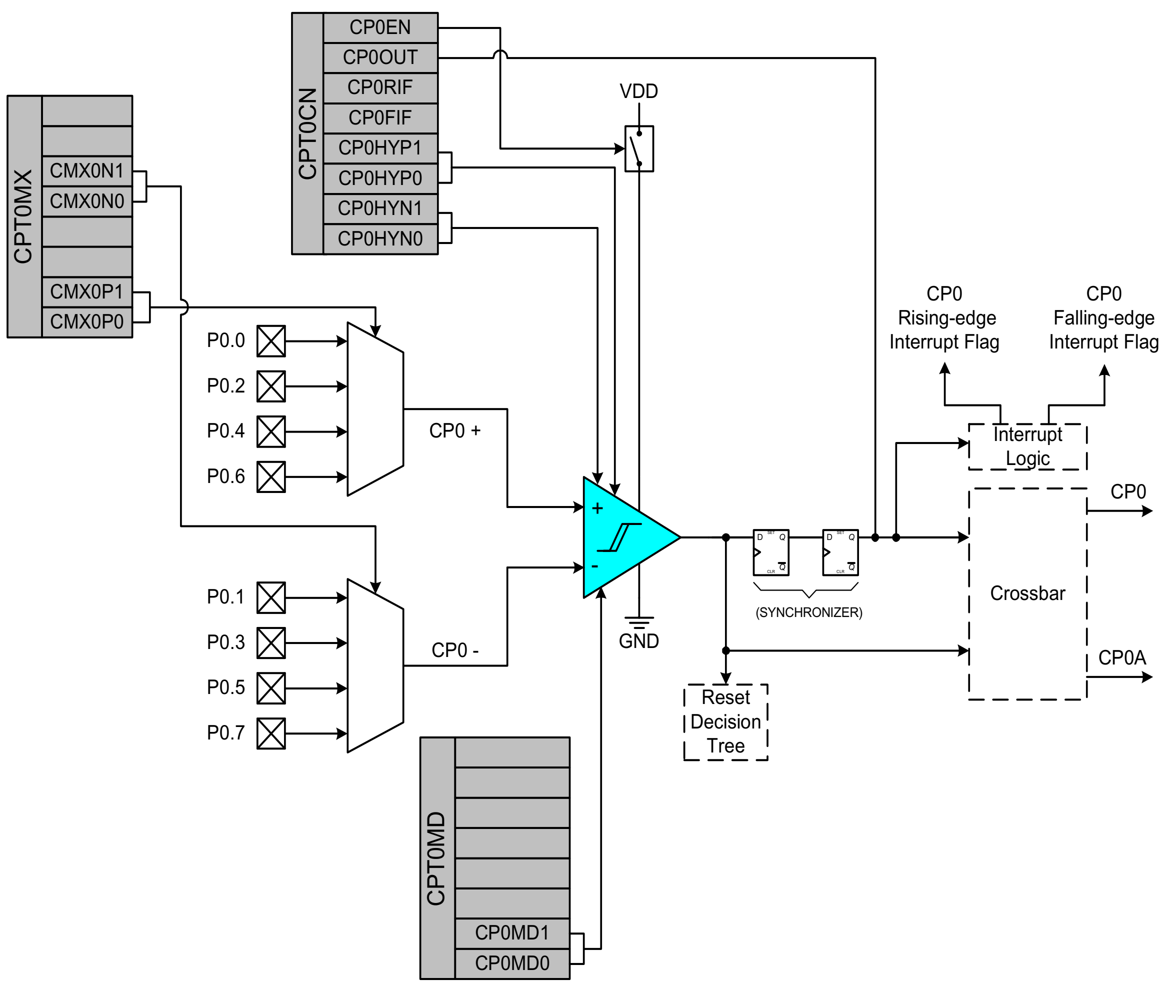
7. COMPARATOR0

C8051F300/1/2/3/4/5 devices include an on-chip programmable voltage comparator, which is shown in Figure 7.1. Comparator0 offers programmable response time and hysteresis, an analog input multiplexer, and two outputs that are optionally available at the Port pins: a synchronous “latched” output (CP0), or an asynchronous “raw” output (CP0A). The asynchronous CP0A signal is available even when the system clock is not active. This allows Comparator0 to operate and generate an output with the device in STOP mode. When assigned to a Port pin, the Comparator0 output may be configured as open drain or push-pull (see **Section “12.2. Port I/O Initialization” on page 98**). Comparator0 may also be used as a reset source (see **Section “9.5. Comparator0 Reset” on page 79**).

The inputs for Comparator0 are selected in the CPT0MX register (Figure 7.4). The CMX0P1-CMX0P0 bits select the Comparator0 positive input; the CMX0N1-CMX0N0 bits select the Comparator0 negative input.

Important Note About Comparator Inputs: The Port pins selected as comparator inputs should be configured as analog inputs in their associated Port configuration register, and configured to be skipped by the Crossbar (for details on Port configuration, see **Section “12.3. General Purpose Port I/O” on page 101**).

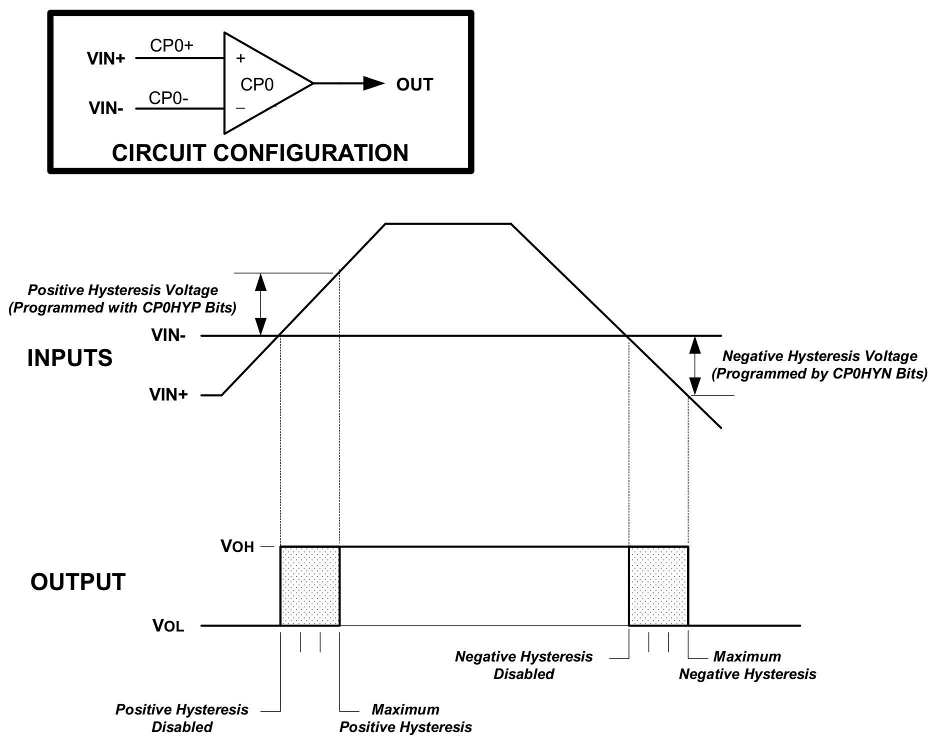
Figure 7.1. Comparator0 Functional Block Diagram



The output of Comparator0 can be polled in software, used as an interrupt source, and/or routed to a Port pin. When routed to a Port pin, the Comparator0 output is available asynchronous or synchronous to the system clock; the asynchronous output is available even in STOP mode (with no system clock active). When disabled, the Comparator0 output (if assigned to a Port I/O pin via the Crossbar) defaults to the logic low state, and its supply current falls to less than 100 nA. See **Section "12.1. Priority Crossbar Decoder" on page 96** for details on configuring the Comparator0 output via the digital Crossbar. Comparator0 inputs can be externally driven from -0.25 V to (VDD) + 0.25 V without damage or upset. The complete electrical specifications for Comparator0 are given in Table 7.1.

The Comparator0 response time may be configured in software via the CP0MD1-0 bits in register CPT0MD (see Figure 7.5). Selecting a longer response time reduces the amount of power consumed by Comparator0. See Table 7.1 for complete timing and power consumption specifications.

Figure 7.2. Comparator Hysteresis Plot



The hysteresis of Comparator0 is software-programmable via its Comparator0 Control register (CPT0CN). The user can program both the amount of hysteresis voltage (referred to the input voltage) and the positive and negative-going symmetry of this hysteresis around the threshold voltage.

The Comparator0 hysteresis is programmed using Bits3-0 in the Comparator0 Control Register CPT0CN (shown in Figure 7.3). The amount of negative hysteresis voltage is determined by the settings of the CP0HYN bits. As shown in Figure 7.2, settings of 20, 10 or 5 mV of negative hysteresis can be programmed, or negative hysteresis can be disabled. In a similar way, the amount of positive hysteresis is determined by the setting the CP0HYP bits.

Comparator0 interrupts can be generated on both rising-edge and falling-edge output transitions. (For Interrupt enable and priority control, see **Section “8.3. Interrupt Handler” on page 67**). The CP0FIF flag is set to logic 1 upon a Comparator0 falling-edge interrupt, and the CP0RIF flag is set to logic 1 upon the Comparator0 rising-edge interrupt. Once set, these bits remain set until cleared by software. The output state of Comparator0 can be obtained at any time by reading the CP0OUT bit. Comparator0 is enabled by setting the CP0EN bit to logic 1, and is disabled by clearing this bit to logic 0.

Figure 7.3. CPT0CN: Comparator0 Control Register

R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
CP0EN	CP0OUT	CP0RIF	CP0FIF	CP0HYP1	CP0HYP0	CP0HYN1	CP0HYN0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xF8

Bit7: CP0EN: Comparator0 Enable Bit.
0: Comparator0 Disabled.
1: Comparator0 Enabled.

Bit6: CP0OUT: Comparator0 Output State Flag.
0: Voltage on CP0+ < CP0-.
1: Voltage on CP0+ > CP0-.

Bit5: CP0RIF: Comparator0 Rising-Edge Interrupt Flag.
0: No Comparator0 Rising Edge Interrupt has occurred since this flag was last cleared.
1: Comparator0 Rising Edge Interrupt has occurred.

Bit4: CP0FIF: Comparator0 Falling-Edge Interrupt Flag.
0: No Comparator0 Falling-Edge Interrupt has occurred since this flag was last cleared.
1: Comparator0 Falling-Edge Interrupt has occurred.

Bits3-2: CP0HYP1-0: Comparator0 Positive Hysteresis Control Bits.
00: Positive Hysteresis Disabled.
01: Positive Hysteresis = 5 mV.
10: Positive Hysteresis = 10 mV.
11: Positive Hysteresis = 20 mV.

Bits1-0: CP0HYN1-0: Comparator0 Negative Hysteresis Control Bits.
00: Negative Hysteresis Disabled.
01: Negative Hysteresis = 5 mV.
10: Negative Hysteresis = 10 mV.
11: Negative Hysteresis = 20 mV.

Figure 7.4. CPT0MX: Comparator0 MUX Selection Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	CMX0N1	CMX0N0	-	-	CMX0P1	CMX0P0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x9F

Bits7-6: UNUSED. Read = 00b, Write = don't care.
Bits6-4: CMX0N1-CMX0N0: Comparator0 Negative Input MUX Select.
These bits select which Port pin is used as the Comparator0 negative input.

CMX0N1	CMX0N0	Negative Input
0	0	P0.1
0	1	P0.3
1	0	P0.5
1	1	P0.7

Bits3-2: UNUSED. Read = 00b, Write = don't care.
Bits1-0: CMX0P1-CMX0P0: Comparator0 Positive Input MUX Select.
These bits select which Port pin is used as the Comparator0 positive input.

CMX0P1	CMX0P0	Positive Input
0	0	P0.0
0	1	P0.2
1	0	P0.4
1	1	P0.6

Figure 7.5. CPT0MD: Comparator0 Mode Selection Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	-	-	-	-	CP0MD1	CP0MD0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x9D

Bits7-2: UNUSED. Read = 000000b, Write = don't care.
Bits1-0: CP0MD1-CP0MD0: Comparator0 Mode Select.
These bits select the response time for Comparator0.

Mode	CP0MD1	CP0MD0	CP0 Response Time (TYP)
0	0	0	100 ns
1	0	1	175 ns
2	1	0	320 ns
3	1	1	1050 ns

Table 7.1. Comparator0 Electrical Characteristics

VDD = 3.0 V, -40°C to +85°C unless otherwise specified.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Response Time: Mode 0, Vcm† = 1.5 V	CP0+ - CP0- = 100 mV		100		ns
	CP0+ - CP0- = -100 mV		250		ns
Response Time: Mode 1, Vcm† = 1.5 V	CP0+ - CP0- = 100 mV		175		ns
	CP0+ - CP0- = -100 mV		500		ns
Response Time: Mode 2, Vcm† = 1.5 V	CP0+ - CP0- = 100 mV		320		ns
	CP0+ - CP0- = -100 mV		1100		ns
Response Time: Mode 3, Vcm† = 1.5 V	CP0+ - CP0- = 100 mV		1050		ns
	CP0+ - CP0- = -100 mV		5200		ns
Common-Mode Rejection Ratio			1.5	4	mV/V
Positive Hysteresis 1	CP0HYP1-0 = 00		0	1	mV
Positive Hysteresis 2	CP0HYP1-0 = 01	3	5	7	mV
Positive Hysteresis 3	CP0HYP1-0 = 10	7	10	15	mV
Positive Hysteresis 4	CP0HYP1-0 = 11	15	20	25	mV
Negative Hysteresis 1	CP0HYN1-0 = 00		0	1	mV
Negative Hysteresis 2	CP0HYN1-0 = 01	3	5	7	mV
Negative Hysteresis 3	CP0HYN1-0 = 10	7	10	15	mV
Negative Hysteresis 4	CP0HYN1-0 = 11	15	20	25	mV
Inverting or Non-Inverting Input Voltage Range		-0.25		VDD + 0.25	V
Input Capacitance			7		pF
Input Bias Current		-5	0.001	+5	nA
Input Offset Voltage		-5		+5	mV
POWER SUPPLY					
Power Supply Rejection			0.1	1	mV/V
Power-up Time			10		µs
Supply Current at DC	Mode 0		7.6		µA
	Mode 1		3.2		µA
	Mode 2		1.3		µA
	Mode 3		0.4		µA

†Vcm is the common-mode voltage on CP0+ and CP0-.

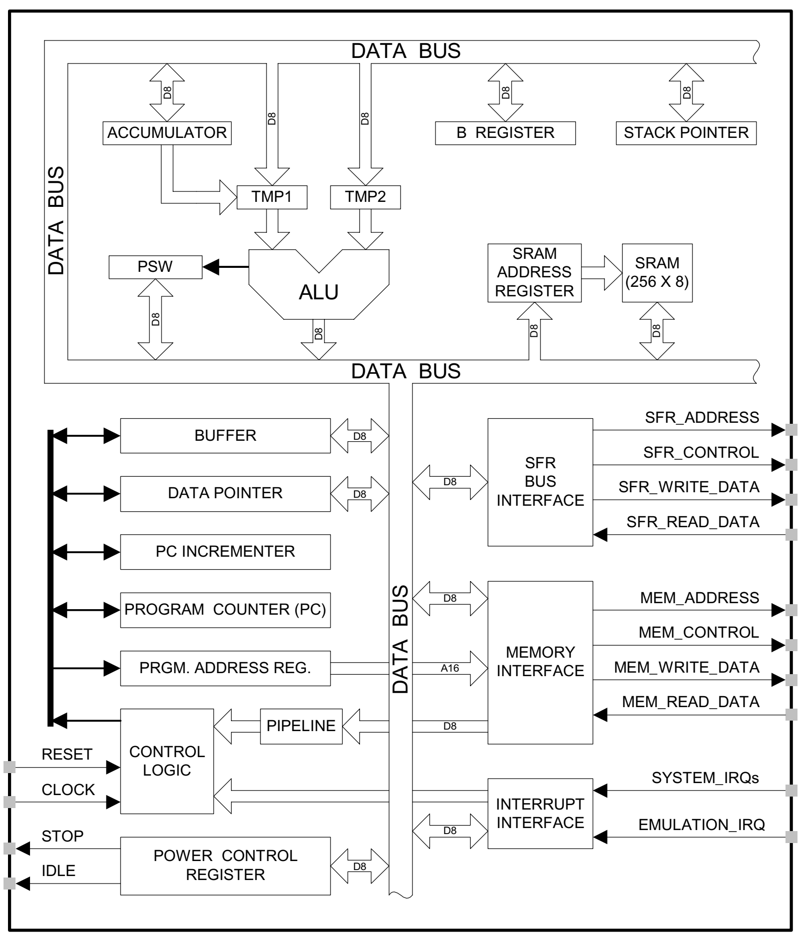
8. CIP-51 MICROCONTROLLER

The MCU system controller core is the CIP-51 microcontroller. The CIP-51 is fully compatible with the MCS-51™ instruction set; standard 803x/805x assemblers and compilers can be used to develop software. The MCU family has a superset of all the peripherals included with a standard 8051. Included are three 16-bit counter/timers (see description in **Section 15**), an enhanced full-duplex UART (see description in **Section 14**), 256 bytes of internal RAM, 128 byte Special Function Register (SFR) address space (**Section 8.2.6**), and one byte-wide I/O Port (see description in **Section 12**). The CIP-51 also includes on-chip debug hardware (see description in **Section 17**), and interfaces directly with the analog and digital subsystems providing a complete data acquisition or control-system solution in a single integrated circuit.

The CIP-51 Microcontroller core implements the standard 8051 organization and peripherals as well as additional custom peripherals and functions to extend its capability (see Figure 8.1 for a block diagram). The CIP-51 includes the following features:

- Fully Compatible with MCS-51 Instruction Set
- 25 MIPS Peak Throughput with 25 MHz Clock
- 0 to 25 MHz Clock Frequency
- 256 Bytes of Internal RAM
- Byte-Wide I/O Port
- Extended Interrupt Handler
- Reset Input
- Power Management Modes
- On-chip Debug Logic
- Program and Data Memory Security

Figure 8.1. CIP-51 Block Diagram



Performance

The CIP-51 employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. In a standard 8051, all instructions except for MUL and DIV take 12 or 24 system clock cycles to execute, and usually have a maximum system clock of 12 MHz. By contrast, the CIP-51 core executes 70% of its instructions in one or two system clock cycles, with no instructions taking more than eight system clock cycles.

With the CIP-51's maximum system clock at 25 MHz, it has a peak throughput of 25 MIPS. The CIP-51 has a total of 109 instructions. The table below shows the total number of instrutions that require each execution time.

Clocks to Execute	1	2	2/3	3	3/4	4	4/5	5	8
Number of Instructions	26	50	5	14	7	3	1	2	1

Programming and Debugging Support

In-system programming of the FLASH program memory and communication with on-chip debug support logic is accomplished via the Cygnal 2-Wire Development Interface (C2). Note that the re-programmable FLASH can also be read and changed a single byte at a time by the application software using the MOVC and MOVX instructions. This feature allows program memory to be used for non-volatile data storage as well as updating program code under software control.

The on-chip debug support logic facilitates full speed in-circuit debugging, allowing the setting of hardware breakpoints, starting, stopping and single stepping through program execution (including interrupt service routines), examination of the program's call stack, and reading/writing the contents of registers and memory. This method of on-chip debugging is completely non-intrusive, requiring no RAM, Stack, timers, or other on-chip resources. C2 details can be found in **Section "17. C2 Interface" on page 161**.

The CIP-51 is supported by development tools from Cygnal Integrated Products and third party vendors. Cygnal provides an integrated development environment (IDE) including editor, macro assembler, debugger and programmer. The IDE's debugger and programmer interface to the CIP-51 via the C2 interface to provide fast and efficient in-system device programming and debugging. Third party macro assemblers and C compilers are also available.

8.1. INSTRUCTION SET

The instruction set of the CIP-51 System Controller is fully compatible with the standard MCS-51™ instruction set. Standard 8051 development tools can be used to develop software for the CIP-51. All CIP-51 instructions are the binary and functional equivalent of their MCS-51™ counterparts, including opcodes, addressing modes and effect on PSW flags. However, instruction timing is different than that of the standard 8051.

8.1.1. Instruction and CPU Timing

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the CIP-51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles.

Due to the pipelined architecture of the CIP-51, most instructions execute in the same number of clock cycles as there are program bytes in the instruction. Conditional branch instructions take one less clock cycle to complete when the branch is not taken as opposed to when the branch is taken. Table 8.1 is the **CIP-51 Instruction Set Summary**, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

8.1.2. MOVX Instruction and Program Memory

The MOVX instruction is typically used to access external data memory (Note: the C8051F300/1/2/3/4/5 does not support external data or program memory). In the CIP-51, the MOVX instruction accesses the on-chip program memory space implemented as re-programmable FLASH memory. This feature provides a mechanism for the CIP-51 to update program code and use the program memory space for non-volatile data storage. Refer to **Section "10. FLASH Memory" on page 83** for further details.

Table 8.1. CIP-51 Instruction Set Summary

Mnemonic	Description	Bytes	Clock Cycles
	ARITHMETIC OPERATIONS		
ADD A, Rn	Add register to A	1	1
ADD A, direct	Add direct byte to A	2	2
ADD A, @Ri	Add indirect RAM to A	1	2
ADD A, #data	Add immediate to A	2	2
ADDC A, Rn	Add register to A with carry	1	1
ADDC A, direct	Add direct byte to A with carry	2	2
ADDC A, @Ri	Add indirect RAM to A with carry	1	2
ADDC A, #data	Add immediate to A with carry	2	2
SUBB A, Rn	Subtract register from A with borrow	1	1
SUBB A, direct	Subtract direct byte from A with borrow	2	2
SUBB A, @Ri	Subtract indirect RAM from A with borrow	1	2
SUBB A, #data	Subtract immediate from A with borrow	2	2
INC A	Increment A	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	2
INC @Ri	Increment indirect RAM	1	2
DEC A	Decrement A	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	2
DEC @Ri	Decrement indirect RAM	1	2
INC DPTR	Increment Data Pointer	1	1

Table 8.1. CIP-51 Instruction Set Summary

Mnemonic	Description	Bytes	Clock Cycles
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1
LOGICAL OPERATIONS			
ANL A, Rn	AND Register to A	1	1
ANL A, direct	AND direct byte to A	2	2
ANL A, @Ri	AND indirect RAM to A	1	2
ANL A, #data	AND immediate to A	2	2
ANL direct, A	AND A to direct byte	2	2
ANL direct, #data	AND immediate to direct byte	3	3
ORL A, Rn	OR Register to A	1	1
ORL A, direct	OR direct byte to A	2	2
ORL A, @Ri	OR indirect RAM to A	1	2
ORL A, #data	OR immediate to A	2	2
ORL direct, A	OR A to direct byte	2	2
ORL direct, #data	OR immediate to direct byte	3	3
XRL A, Rn	Exclusive-OR Register to A	1	1
XRL A, direct	Exclusive-OR direct byte to A	2	2
XRL A, @Ri	Exclusive-OR indirect RAM to A	1	2
XRL A, #data	Exclusive-OR immediate to A	2	2
XRL direct, A	Exclusive-OR A to direct byte	2	2
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3
CLR A	Clear A	1	1
CPL A	Complement A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through Carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through Carry	1	1
SWAP A	Swap nibbles of A	1	1
DATA TRANSFER			
MOV A, Rn	Move Register to A	1	1
MOV A, direct	Move direct byte to A	2	2
MOV A, @Ri	Move indirect RAM to A	1	2
MOV A, #data	Move immediate to A	2	2
MOV Rn, A	Move A to Register	1	1
MOV Rn, direct	Move direct byte to Register	2	2
MOV Rn, #data	Move immediate to Register	2	2
MOV direct, A	Move A to direct byte	2	2
MOV direct, Rn	Move Register to direct byte	2	2
MOV direct, direct	Move direct byte to direct byte	3	3
MOV direct, @Ri	Move indirect RAM to direct byte	2	2
MOV direct, #data	Move immediate to direct byte	3	3
MOV @Ri, A	Move A to indirect RAM	1	2
MOV @Ri, direct	Move direct byte to indirect RAM	2	2
MOV @Ri, #data	Move immediate to indirect RAM	2	2

Table 8.1. CIP-51 Instruction Set Summary

Mnemonic	Description	Bytes	Clock Cycles
MOV DPTR, #data16	Load DPTR with 16-bit constant	3	3
MOVC A, @A+DPTR	Move code byte relative DPTR to A	1	3
MOVC A, @A+PC	Move code byte relative PC to A	1	3
MOVX A, @Ri	Move external data (8-bit address) to A	1	3
MOVX @Ri, A	Move A to external data (8-bit address)	1	3
MOVX A, @DPTR	Move external data (16-bit address) to A	1	3
MOVX @DPTR, A	Move A to external data (16-bit address)	1	3
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A, Rn	Exchange Register with A	1	1
XCH A, direct	Exchange direct byte with A	2	2
XCH A, @Ri	Exchange indirect RAM with A	1	2
XCHD A, @Ri	Exchange low nibble of indirect RAM with A	1	2
BOOLEAN MANIPULATION			
CLR C	Clear Carry	1	1
CLR bit	Clear direct bit	2	2
SETB C	Set Carry	1	1
SETB bit	Set direct bit	2	2
CPL C	Complement Carry	1	1
CPL bit	Complement direct bit	2	2
ANL C, bit	AND direct bit to Carry	2	2
ANL C, /bit	AND complement of direct bit to Carry	2	2
ORL C, bit	OR direct bit to carry	2	2
ORL C, /bit	OR complement of direct bit to Carry	2	2
MOV C, bit	Move direct bit to Carry	2	2
MOV bit, C	Move Carry to direct bit	2	2
JC rel	Jump if Carry is set	2	2/3
JNC rel	Jump if Carry is not set	2	2/3
JB bit, rel	Jump if direct bit is set	3	3/4
JNB bit, rel	Jump if direct bit is not set	3	3/4
JBC bit, rel	Jump if direct bit is set and clear bit	3	3/4
PROGRAM BRANCHING			
ACALL addr11	Absolute subroutine call	2	3
LCALL addr16	Long subroutine call	3	4
RET	Return from subroutine	1	5
RETI	Return from interrupt	1	5
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (relative address)	2	3
JMP @A+DPTR	Jump indirect relative to DPTR	1	3
JZ rel	Jump if A equals zero	2	2/3
JNZ rel	Jump if A does not equal zero	2	2/3
CJNE A, direct, rel	Compare direct byte to A and jump if not equal	3	3/4
CJNE A, #data, rel	Compare immediate to A and jump if not equal	3	3/4
CJNE Rn, #data, rel	Compare immediate to Register and jump if not equal	3	3/4

Table 8.1. CIP-51 Instruction Set Summary

Mnemonic	Description	Bytes	Clock Cycles
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal	3	4/5
DJNZ Rn, rel	Decrement Register and jump if not zero	2	2/3
DJNZ direct, rel	Decrement direct byte and jump if not zero	3	3/4
NOP	No operation	1	1

Notes on Registers, Operands and Addressing Modes:

Rn - Register R0-R7 of the currently selected register bank.

@Ri - Data RAM location addressed indirectly through R0 or R1.

rel - 8-bit, signed (two's complement) offset relative to the first byte of the following instruction. Used by SJMP and all conditional jumps.

direct - 8-bit internal data location's address. This could be a direct-access Data RAM location (0x00-0x7F) or an SFR (0x80-0xFF).

#data - 8-bit constant

#data16 - 16-bit constant

bit - Direct-accessed bit in Data RAM or SFR

addr11 - 11-bit destination address used by ACALL and AJMP. The destination must be within the same 2K-byte page of program memory as the first byte of the following instruction.

addr16 - 16-bit destination address used by LCALL and LJMP. The destination may be anywhere within the 8K-byte program memory space.

There is one unused opcode (0xA5) that performs the same function as NOP.
All mnemonics copyrighted © Intel Corporation 1980.

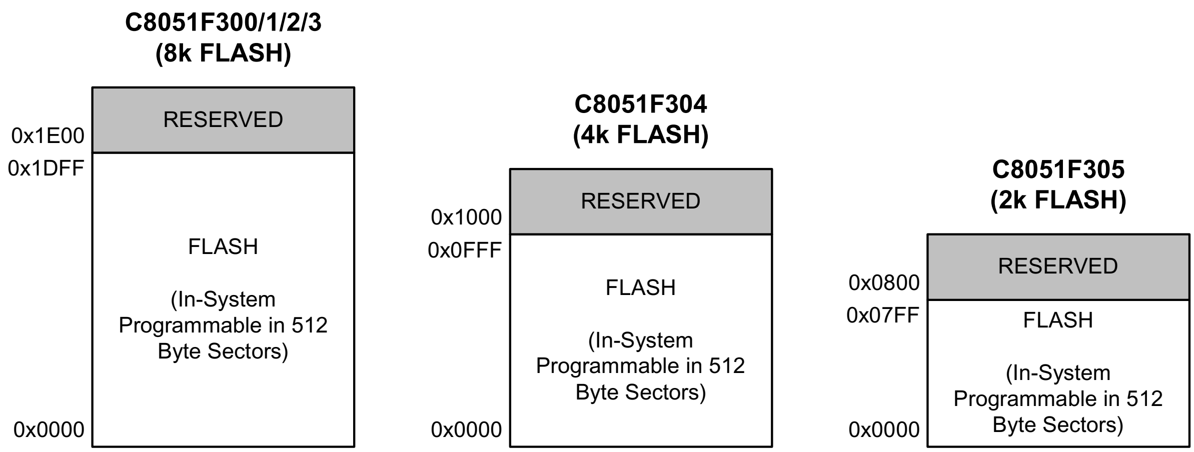
8.2. MEMORY ORGANIZATION

The memory organization of the CIP-51 System Controller is similar to that of a standard 8051. There are two separate memory spaces: program memory and data memory. Program and data memory share the same address space but are accessed via different instruction types. The CIP-51 memory organization is shown in Figure 8.2 and Figure 8.3.

8.2.1. Program Memory

The CIP-51 core has a 64k-byte program memory space. The C8051F300/1/2/3 implements 8192 bytes of this program memory space as in-system, re-programmable FLASH memory, organized in a contiguous block from addresses 0x0000 to 0x1FFF. Note: 512 bytes (0x1E00 - 0x1FFF) of this memory are reserved for factory use and are not available for user program storage. The C8051F304 implements 4096 bytes of re-programmable FLASH program memory space; the C8051F305 implements 2048 bytes of re-programmable FLASH program memory space. Figure 8.2 shows the program memory maps for C8051F300/1/2/3/4/5 devices.

Figure 8.2. Program Memory Maps



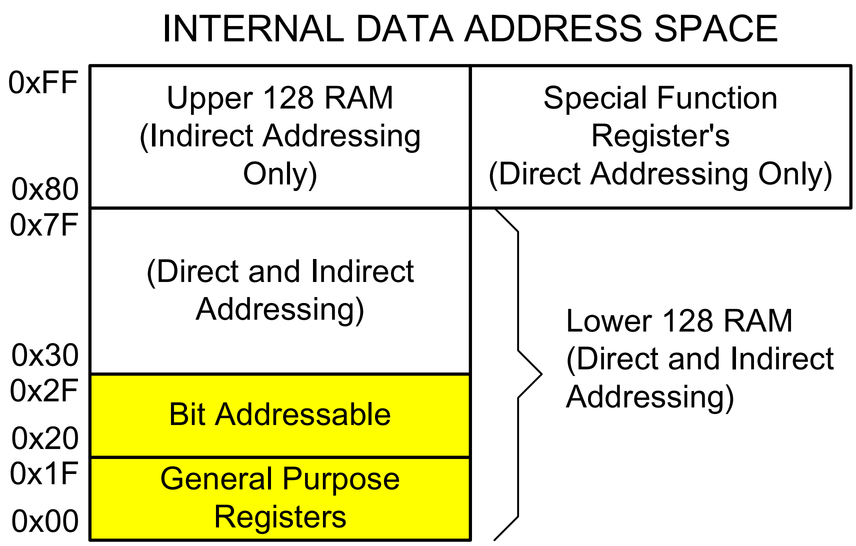
Program memory is normally assumed to be read-only. However, the CIP-51 can write to program memory by setting the Program Store Write Enable bit (PSCTL.0) and using the MOVX instruction. This feature provides a mechanism for the CIP-51 to update program code and use the program memory space for non-volatile data storage. Refer to Section "10. FLASH Memory" on page 83 for further details.

8.2.2. Data Memory

The CIP-51 includes 256 bytes of internal RAM mapped into the data memory space from 0x00 through 0xFF. The lower 128 bytes of data memory are used for general purpose registers and scratch pad memory. Either direct or indirect addressing may be used to access the lower 128 bytes of data memory. Locations 0x00 through 0x1F are addressable as four banks of general purpose registers, each bank consisting of eight byte-wide registers. The next 16 bytes, locations 0x20 through 0x2F, may either be addressed as bytes or as 128 bit locations accessible with the direct addressing mode.

The upper 128 bytes of data memory are accessible only by indirect addressing. This region occupies the same address space as the Special Function Registers (SFR) but is physically separate from the SFR space. The addressing mode used by an instruction when accessing locations above 0x7F determines whether the CPU accesses the upper 128 bytes of data memory space or the SFRs. Instructions that use direct addressing will access the SFR space. Instructions using indirect addressing above 0x7F access the upper 128 bytes of data memory. Figure 8.3 illustrates the data memory organization of the CIP-51.

Figure 8.3. Data Memory Map



8.2.3. General Purpose Registers

The lower 32 bytes of data memory, locations 0x00 through 0x1F, may be addressed as four banks of general-purpose registers. Each bank consists of eight byte-wide registers designated R0 through R7. Only one of these banks may be enabled at a time. Two bits in the program status word, RS0 (PSW.3) and RS1 (PSW.4), select the active register bank (see description of the PSW in Figure 8.7). This allows fast context switching when entering subroutines and interrupt service routines. Indirect addressing modes use registers R0 and R1 as index registers.

8.2.4. Bit Addressable Locations

In addition to direct access to data memory organized as bytes, the sixteen data memory locations at 0x20 through 0x2F are also accessible as 128 individually addressable bits. Each bit has a bit address from 0x00 to 0x7F. Bit 0 of the byte at 0x20 has bit address 0x00 while bit 7 of the byte at 0x20 has bit address 0x07. Bit 7 of the byte at 0x2F has bit address 0x7F. A bit access is distinguished from a full byte access by the type of instruction used (bit source or destination operands as opposed to a byte source or destination).

The MCS-51™ assembly language allows an alternate notation for bit addressing of the form XX.B where XX is the byte address and B is the bit position within the byte. For example, the instruction:

```
MOV    C, 22.3h
```

moves the Boolean value at 0x13 (bit 3 of the byte at location 0x22) into the Carry flag.

8.2.5. Stack

A programmer's stack can be located anywhere in the 256-byte data memory. The stack area is designated using the Stack Pointer (SP, 0x81) SFR. The SP will point to the last location used. The next value pushed on the stack is placed at SP+1 and then SP is incremented. A reset initializes the stack pointer to location 0x07. Therefore, the first value pushed on the stack is placed at location 0x08, which is also the first register (R0) of register bank 1. Thus, if more than one register bank is to be used, the SP should be initialized to a location in the data memory not being used for data storage. The stack depth can extend up to 256 bytes.

8.2.6. Special Function Registers

The direct-access data memory locations from 0x80 to 0xFF constitute the special function registers (SFRs). The SFRs provide control and data exchange with the CIP-51's resources and peripherals. The CIP-51 duplicates the SFRs found in a typical 8051 implementation as well as implementing additional SFRs used to configure and access the sub-systems unique to the MCU. This allows the addition of new functionality while retaining compatibility with the MCS-51™ instruction set. Table 8.2 lists the SFRs implemented in the CIP-51 System Controller.

The SFR registers are accessed anytime the direct addressing mode is used to access memory locations from 0x80 to 0xFF. SFRs with addresses ending in 0x0 or 0x8 (e.g. P0, TCON, SCON0, IE, etc.) are bit-addressable as well as byte-addressable. All other SFRs are byte-addressable only. Unoccupied addresses in the SFR space are reserved for future use. Accessing these areas will have an indeterminate effect and should be avoided. Refer to the corresponding pages of the datasheet, as indicated in Table 8.3, for a detailed description of each register.

Table 8.2. Special Function Register (SFR) Memory Map

F8	CPT0CN	PCA0L	PCA0H	PCA0CPL0	PCA0CPH0			
F0	B	P0MDIN					EIP1	
E8	ADC0CN	PCA0CPL1	PCA0CPH1	PCA0CPL2	PCA0CPH2			RSTSRC
E0	ACC	XBR0	XBR1	XBR2	IT01CF		EIE1	
D8	PCA0CN	PCA0MD	PCA0CPM0	PCA0CPM1	PCA0CPM2			
D0	PSW	REF0CN						
C8	TMR2CN		TMR2RLL	TMR2RLH	TMR2L	TMR2H		
C0	SMB0CN	SMB0CF	SMB0DAT		ADC0GT		ADC0LT	
B8	IP			AMX0SL	ADC0CF		ADC0	
B0		OSCXCN	OSCICN	OSCICL			FLSCL	FLKEY
A8	IE							
A0					P0MDOUT			
98	SCON0	SBUF0				CPT0MD		CPT0MX
90								
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PSCTL
80	P0	SP	DPL	DPH				PCON
	0(8) (bit addressable)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)

Table 8.3. Special Function Registers

SFRs are listed in alphabetical order. All undefined SFR locations are reserved

Register	Address	Description	Page No.
ACC	0xE0	Accumulator	66
ADC0CF	0xBC	ADC0 Configuration	39
ADC0CN	0xE8	ADC0 Control	40
ADC0GT	0xC4	ADC0 Greater-Than Compare Word	43
ADC0LT	0xC6	ADC0 Less-Than Compare Word	43
ADC0	0xBE	ADC0 Data Word	39
AMX0SL	0xBB	ADC0 Multiplexer Channel Select	38
B	0xF0	B Register	66
CKCON	0x8E	Clock Control	139
CPT0CN	0xF8	Comparator0 Control	49
CPT0MD	0x9D	Comparator0 Mode Selection	51
CPT0MX	0x9F	Comparator0 MUX Selection	50
DPH	0x83	Data Pointer High	64
DPL	0x82	Data Pointer Low	64
EIE1	0xE6	Extended Interrupt Enable 1	72
EIP1	0xF6	External Interrupt Priority 1	73

Table 8.3. Special Function Registers

Register	Address	Description	Page
FLKEY	0xB7	FLASH Lock and Key	87
FLSCL	0xB6	FLASH Scale	87
IE	0xA8	Interrupt Enable	70
IP	0xB8	Interrupt Priority	71
IT01CF	0xE4	INT0/INT1 Configuration Register	74
OSCICL	0xB3	Internal Oscillator Calibration	90
OSCICN	0xB2	Internal Oscillator Control	90
OSCXCN	0xB1	External Oscillator Control	92
P0	0x80	Port 0 Latch	101
P0MDIN	0xF1	Port 0 Input Mode Configuration	101
P0MDOUT	0xA4	Port 0 Output Mode Configuration	102
PCA0CN	0xD8	PCA Control	156
PCA0MD	0xD9	PCA Mode	157
PCA0CPH0	0xFC	PCA Capture 0 High	160
PCA0CPH1	0xEA	PCA Capture 1 High	160
PCA0CPH2	0xEC	PCA Capture 2 High	160
PCA0CPL0	0xFB	PCA Capture 0 Low	160
PCA0CPL1	0xE9	PCA Capture 1 Low	160
PCA0CPL2	0xEB	PCA Capture 2 Low	160
PCA0CPM0	0xDA	PCA Module 0 Mode Register	158
PCA0CPM1	0xDB	PCA Module 1 Mode Register	158
PCA0CPM2	0xDC	PCA Module 2 Mode Register	158
PCA0H	0xFA	PCA Counter High	159
PCA0L	0xF9	PCA Counter Low	159
PCON	0x87	Power Control	76
PSCTL	0x8F	Program Store R/W Control	86
PSW	0xD0	Program Status Word	65
REF0CN	0xD1	Voltage Reference Control	45
RSTSRC	0xEF	Reset Source Configuration/Status	81
SBUF0	0x99	UART 0 Data Buffer	129
SCON0	0x98	UART 0 Control	128
SMB0CF	0xC1	SMBus Configuration	110
SMB0CN	0xC0	SMBus Control	112
SMB0DAT	0xC2	SMBus Data	114
SP	0x81	Stack Pointer	65
TMR2CN	0xC8	Timer/Counter 2 Control	143
TCON	0x88	Timer/Counter Control	137
TH0	0x8C	Timer/Counter 0 High	140
TH1	0x8D	Timer/Counter 1 High	140
TL0	0x8A	Timer/Counter 0 Low	140
TL1	0x8B	Timer/Counter 1 Low	140
TMOD	0x89	Timer/Counter Mode	138
TMR2RLH	0xCB	Timer/Counter 2 Reload High	144
TMR2RLL	0xCA	Timer/Counter 2 Reload Low	144
TMR2H	0xCD	Timer/Counter 2 High	144
TMR2L	0xCC	Timer/Counter 2 Low	144

Table 8.3. Special Function Registers

Register	Address	Description	Page
XBR0	0xE1	Port I/O Crossbar Control 0	99
XBR1	0xE2	Port I/O Crossbar Control 1	99
XBR2	0xE3	Port I/O Crossbar Control 2	100
0x97, 0xAE, 0xAF, 0xB4, 0xB6, 0xBF, 0xCE, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xDD, 0xDE, 0xDF, 0xF5		Reserved	

8.2.7. Register Descriptions

Following are descriptions of SFRs related to the operation of the CIP-51 System Controller. Reserved bits should not be set to logic l. Future product versions may use these bits to implement new features in which case the reset value of the bit will be logic 0, selecting the feature's default state. Detailed descriptions of the remaining SFRs are included in the sections of the datasheet associated with their corresponding system function.

Figure 8.4. DPL: Data Pointer Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x82

Bits7-0: DPL: Data Pointer Low.
The DPL register is the low byte of the 16-bit DPTR. DPTR is used to access indirectly addressed FLASH memory.

Figure 8.5. DPH: Data Pointer High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x83

Bits7-0: DPH: Data Pointer High.
The DPH register is the high byte of the 16-bit DPTR. DPTR is used to access indirectly addressed FLASH memory.

Figure 8.6. SP: Stack Pointer

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x81

Bits7-0: SP: Stack Pointer.
The Stack Pointer holds the location of the top of the stack. The stack pointer is incremented before every PUSH operation. The SP register defaults to 0x07 after reset.

Figure 8.7. PSW: Program Status Word

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	Reset Value
CY	AC	F0	RS1	RS0	OV	F1	PARITY	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xD0

Bit7: CY: Carry Flag.
This bit is set when the last arithmetic operation resulted in a carry (addition) or a borrow (subtraction). It is cleared to logic 0 by all other arithmetic operations.

Bit6: AC: Auxiliary Carry Flag
This bit is set when the last arithmetic operation resulted in a carry into (addition) or a borrow from (subtraction) the high order nibble. It is cleared to logic 0 by all other arithmetic operations.

Bit5: F0: User Flag 0.
This is a bit-addressable, general purpose flag for use under software control.

Bits4-3: RS1-RS0: Register Bank Select.
These bits select which register bank is used during register accesses.

RS1	RS0	Register Bank	Address
0	0	0	0x00 - 0x07
0	1	1	0x08 - 0x0F
1	0	2	0x10 - 0x17
1	1	3	0x18 - 0x1F

Bit2: OV: Overflow Flag.
This bit is set to logic 1 if the last arithmetic operation resulted in a carry (addition), borrow (subtraction), or overflow (multiply or divide). It is cleared to logic 0 by all other arithmetic operations.

Bit1: F1: User Flag 1.
This is a bit-addressable, general purpose flag for use under software control.

Bit0: PARITY: Parity Flag.
This bit is set to logic 1 if the sum of the eight bits in the accumulator is odd and cleared if the sum is even.

Figure 8.8. ACC: Accumulator

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xE0

Bits7-0: ACC: Accumulator.
This register is the accumulator for arithmetic operations.

Figure 8.9. B: B Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xF0

Bits7-0: B: B Register.
This register serves as a second accumulator for certain arithmetic operations.

8.3. Interrupt Handler

The CIP-51 includes an extended interrupt system supporting a total of 12 interrupt sources with two priority levels. The allocation of interrupt sources between on-chip peripherals and external inputs pins varies according to the specific version of the device. Each interrupt source has one or more associated interrupt-pending flag(s) located in an SFR. When a peripheral or external source meets a valid interrupt condition, the associated interrupt-pending flag is set to logic 1.

If interrupts are enabled for the source, an interrupt request is generated when the interrupt-pending flag is set. As soon as execution of the current instruction is complete, the CPU generates an LCALL to a predetermined address to begin execution of an interrupt service routine (ISR). Each ISR must end with an RETI instruction, which returns program execution to the next instruction that would have been executed if the interrupt request had not occurred. If interrupts are not enabled, the interrupt-pending flag is ignored by the hardware and program execution continues as normal. (The interrupt-pending flag is set to logic 1 regardless of the interrupt's enable/disable state.)

Each interrupt source can be individually enabled or disabled through the use of an associated interrupt enable bit in an SFR (IE-EIE1). However, interrupts must first be globally enabled by setting the EA bit (IE.7) to logic 1 before the individual interrupt enables are recognized. Setting the EA bit to logic 0 disables all interrupt sources regardless of the individual interrupt-enable settings.

Some interrupt-pending flags are automatically cleared by the hardware when the CPU vectors to the ISR. However, most are not cleared by the hardware and must be cleared by software before returning from the ISR. If an interrupt-pending flag remains set after the CPU completes the return-from-interrupt (RETI) instruction, a new interrupt request will be generated immediately and the CPU will re-enter the ISR after the completion of the next instruction.

8.3.1. MCU Interrupt Sources and Vectors

The MCUs support 12 interrupt sources. Software can simulate an interrupt by setting any interrupt-pending flag to logic 1. If interrupts are enabled for the flag, an interrupt request will be generated and the CPU will vector to the ISR address associated with the interrupt-pending flag. MCU interrupt sources, associated vector addresses, priority order and control bits are summarized in Table 8.4 on page 69. Refer to the datasheet section associated with a particular on-chip peripheral for information regarding valid interrupt conditions for the peripheral and the behavior of its interrupt-pending flag(s).

8.3.2. External Interrupts

The /INT0 and /INT1 external interrupt sources are configurable as active high or low, edge or level sensitive. The IN0PL (/INT0 Polarity) and IN1PL (/INT1 Polarity) bits in the IT01CF register select active high or active low; the IT0 and IT1 bits in TCON (**Section "15.1. Timer 0 and Timer 1" on page 133**) select level or edge sensitive. The table below lists the possible configurations.

IT0	IN0PL	/INT0 Interrupt
1	0	Active low, edge sensitive
1	1	Active high, edge sensitive
0	0	Active low, level sensitive
0	1	Active high, level sensitive

IT1	IN1PL	/INT1 Interrupt
1	0	Active low, edge sensitive
1	1	Active high, edge sensitive
0	0	Active low, level sensitive
0	1	Active high, level sensitive

/INT0 and /INT1 are assigned to Port pins as defined in the IT01CF register (see Figure 8.14). Note that /INT0 and /INT0 Port pin assignments are independent of any Crossbar assignments. /INT0 and /INT1 will monitor their assigned Port pins without disturbing the peripheral that was assigned the Port pin via the Crossbar. To assign a Port pin only to /INT0 and/or /INT1, configure the Crossbar to skip the selected pin(s). This is accomplished by setting the associated bit in register XBR0 (see **Section "12.1. Priority Crossbar Decoder" on page 96** for complete details on configuring the Crossbar).

IE0 (TCON.1) and IE1 (TCON.3) serve as the interrupt-pending flags for the /INT0 and /INT1 external interrupts, respectively. If an /INT0 or /INT1 external interrupt is configured as edge-sensitive, the corresponding interrupt-pending flag is automatically cleared by the hardware when the CPU vectors to the ISR. When configured as level sensitive, the interrupt-pending flag remains logic 1 while the input is active as defined by the corresponding polarity bit (IN0PL or IN1PL); the flag remains logic 0 while the input is inactive. The external interrupt source must hold the input active until the interrupt request is recognized. It must then deactivate the interrupt request before execution of the ISR completes or another interrupt request will be generated.

8.3.3. Interrupt Priorities

Each interrupt source can be individually programmed to one of two priority levels: low or high. A low priority interrupt service routine can be preempted by a high priority interrupt. A high priority interrupt cannot be preempted. Each interrupt has an associated interrupt priority bit in an SFR (IP or EIP1) used to configure its priority level. Low priority is the default. If two interrupts are recognized simultaneously, the interrupt with the higher priority is serviced first. If both interrupts have the same priority level, a fixed priority order is used to arbitrate, given in Table 8.4.

8.3.4. Interrupt Latency

Interrupt response time depends on the state of the CPU when the interrupt occurs. Pending interrupts are sampled and priority decoded each system clock cycle. Therefore, the fastest possible response time is 5 system clock cycles: 1 clock cycle to detect the interrupt and 4 clock cycles to complete the LCALL to the ISR. If an interrupt is pending when a RETI is executed, a single instruction is executed before an LCALL is made to service the pending interrupt. Therefore, the maximum response time for an interrupt (when no other interrupt is currently being serviced or the new interrupt is of greater priority) occurs when the CPU is performing an RETI instruction followed by a DIV as the next instruction. In this case, the response time is 18 system clock cycles: 1 clock cycle to detect the interrupt, 5 clock cycles to execute the RETI, 8 clock cycles to complete the DIV instruction and 4 clock cycles to execute the LCALL to the ISR. If the CPU is executing an ISR for an interrupt with equal or higher priority, the new interrupt will not be serviced until the current ISR completes, including the RETI and following instruction.

Table 8.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y	N	ES0 (IE.4)	PS0 (IP.4)
Timer 2 Overflow	0x002B	5	TF2H (TMR2CN.7) TF2L (TMR2CN.6)	Y	N	ET2 (IE.5)	PT2 (IP.5)
SMBus Interface	0x0033	6	SI (SMB0CN.0)	Y	N	ESMB0 (EIE1.0)	PSMB0 (EIP1.0)
ADC0 Window Compare	0x003B	7	AD0WINT (ADC0CN.3)	Y	N	EWADC0 (EIE1.1)	PWADC0 (EIP1.1)
ADC0 Conversion Complete	0x0043	8	AD0INT (ADC0CN.5)	Y	N	EADC0C (EIE1.2)	PADC0C (EIP1.2)
Programmable Counter Array	0x004B	9	CF (PCA0CN.7) CCFn (PCA0CN.n)	Y	N	EPCA0 (EIE1.3)	PPCA0 (EIP1.3)
Comparator0 Falling Edge	0x0053	10	CP0FIF (CPT0CN.4)	N	N	ECP0F (EIE1.4)	PCP0F (EIP1.4)
Comparator0 Rising Edge	0x005B	11	CP0RIF (CPT0CN.5)	N	N	ECP0R (EIE1.5)	PCP0R (EIP1.5)

8.3.5. Interrupt Register Descriptions

The SFRs used to enable the interrupt sources and set their priority level are described below. Refer to the datasheet section associated with a particular on-chip peripheral for information regarding valid interrupt conditions for the peripheral and the behavior of its interrupt-pending flag(s).

Figure 8.10. IE: Interrupt Enable

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EA	IEGF0	ET2	ES0	ET1	EX1	ET0	EX0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xA8

Bit7: EA: Enable All Interrupts.
This bit globally enables/disables all interrupts. It overrides the individual interrupt mask settings.
0: Disable all interrupt sources.
1: Enable each interrupt according to its individual mask setting.

Bit6: IEGF0: General Purpose Flag 0.
This is a general purpose flag for use under software control.

Bit5: ET2: Enable Timer 2 Interrupt.
This bit sets the masking of the Timer 2 interrupt.
0: Disable Timer 2 interrupt.
1: Enable interrupt requests generated by the TF2L or TF2H flags.

Bit4: ES0: Enable UART0 Interrupt.
This bit sets the masking of the UART0 interrupt.
0: Disable UART0 interrupt.
1: Enable UART0 interrupt.

Bit3: ET1: Enable Timer 1 Interrupt.
This bit sets the masking of the Timer 1 interrupt.
0: Disable all Timer 1 interrupt.
1: Enable interrupt requests generated by the TF1 flag.

Bit2: EX1: Enable External Interrupt 1.
This bit sets the masking of external interrupt 1.
0: Disable external interrupt 1.
1: Enable interrupt requests generated by the /INT1 input.

Bit1: ET0: Enable Timer 0 Interrupt.
This bit sets the masking of the Timer 0 interrupt.
0: Disable all Timer 0 interrupt.
1: Enable interrupt requests generated by the TF0 flag.

Bit0: EX0: Enable External Interrupt 0.
This bit sets the masking of external interrupt 0.
0: Disable external interrupt 0.
1: Enable interrupt requests generated by the /INT0 input.

Figure 8.11. IP: Interrupt Priority

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	PT2	PS0	PT1	PX1	PT0	PX0	11000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xB8

Bits7-6: UNUSED. Read = 11b, Write = don't care.

Bit5: PT2: Timer 2 Interrupt Priority Control.
This bit sets the priority of the Timer 2 interrupt.
0: Timer 2 interrupt priority determined by default priority order.
1: Timer 2 interrupts set to high priority level.

Bit4: PS0: UART0 Interrupt Priority Control.
This bit sets the priority of the UART0 interrupt.
0: UART0 interrupt priority determined by default priority order.
1: UART0 interrupts set to high priority level.

Bit3: PT1: Timer 1 Interrupt Priority Control.
This bit sets the priority of the Timer 1 interrupt.
0: Timer 1 interrupt priority determined by default priority order.
1: Timer 1 interrupts set to high priority level.

Bit2: PX1: External Interrupt 1 Priority Control.
This bit sets the priority of the External Interrupt 1 interrupt.
0: External Interrupt 1 priority determined by default priority order.
1: External Interrupt 1 set to high priority level.

Bit1: PT0: Timer 0 Interrupt Priority Control.
This bit sets the priority of the Timer 0 interrupt.
0: Timer 0 interrupt priority determined by default priority order.
1: Timer 0 interrupt set to high priority level.

Bit0: PX0: External Interrupt 0 Priority Control.
This bit sets the priority of the External Interrupt 0 interrupt.
0: External Interrupt 0 priority determined by default priority order.
1: External Interrupt 0 set to high priority level.

Figure 8.12. EIE1: Extended Interrupt Enable 1

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	ECP0R	ECP0F	EPCA0	EADC0C	EWADC0	ESMB0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE6

Bits7-6: UNUSED. Read = 00b. Write = don’t care.

Bit5: ECP0R: Enable Comparator0 (CP0) Rising Edge Interrupt.
This bit sets the masking of the CP0 Rising Edge interrupt.
0: Disable CP0 Rising Edge interrupt.
1: Enable interrupt requests generated by the CP0RIF flag.

Bit4: ECP0F: Enable Comparator0 (CP0) Falling Edge Interrupt.
This bit sets the masking of the CP0 Falling Edge interrupt.
0: Disable CP0 Falling Edge interrupt.
1: Enable interrupt requests generated by the CP0FIF flag .

Bit3: EPCA0: Enable Programmable Counter Array (PCA0) Interrupt.
This bit sets the masking of the PCA0 interrupts.
0: Disable all PCA0 interrupts.
1: Enable interrupt requests generated by PCA0.

Bit2: EADC0C: Enable ADC0 Conversion Complete Interrupt.
This bit sets the masking of the ADC0 Conversion Complete interrupt.
0: Disable ADC0 Conversion Complete interrupt.
1: Enable interrupt requests generated by the AD0INT flag.

Bit1: EWADC0: Enable Window Comparison ADC0 Interrupt.
This bit sets the masking of ADC0 Window Comparison interrupt.
0: Disable ADC0 Window Comparison interrupt.
1: Enable interrupt requests generated by ADC0 Window Compare flag.

Bit0: ESMB0: Enable SMBus Interrupt.
This bit sets the masking of the SMBus interrupt.
0: Disable all SMBus interrupts.
1: Enable interrupt requests generated by the SI flag.

Figure 8.13. EIP1: Extended Interrupt Priority 1

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	PCP0R	PCP0F	PPCA0	PADC0C	PWADC0	PSMB0	11000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xF6

Bits7-6: UNUSED. Read = 11b. Write = don't care.
Bit5: PCP0R: Comparator0 (CP0) Rising Interrupt Priority Control.
This bit sets the priority of the CP0 rising-edge interrupt.
0: CP0 rising interrupt set to low priority level.
1: CP0 rising interrupt set to high priority level.
Bit4: PCP0F: Comparator0 (CP0) Falling Interrupt Priority Control.
This bit sets the priority of the CP0 falling-edge interrupt.
0: CP0 falling interrupt set to low priority level.
1: CP0 falling interrupt set to high priority level.
Bit3: PPCA0: Programmable Counter Array (PCA0) Interrupt Priority Control.
This bit sets the priority of the PCA0 interrupt.
0: PCA0 interrupt set to low priority level.
1: PCA0 interrupt set to high priority level.
Bit2: PADC0C ADC0 Conversion Complete Interrupt Priority Control
This bit sets the priority of the ADC0 Conversion Complete interrupt.
0: ADC0 Conversion Complete interrupt set to low priority level.
1: ADC0 Conversion Complete interrupt set to high priority level.
Bit1: PWADC0: ADC0 Window Comparator Interrupt Priority Control.
This bit sets the priority of the ADC0 Window interrupt.
0: ADC0 Window interrupt set to low priority level.
1: ADC0 Window interrupt set to high priority level.
Bit0: PSMB0: SMBus Interrupt Priority Control.
This bit sets the priority of the SMBus interrupt.
0: SMBus interrupt set to low priority level.
1: SMBus interrupt set to high priority level.

Figure 8.14. IT01CF: INT0/INT1 Configuration Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
IN1PL	IN1SL2	IN1SL1	IN1SL0	IN0PL	IN0SL2	IN0SL1	IN0SL0	00000001
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE4

Note: Refer to Figure 15.4 for INT0/1 edge- or level-sensitive interrupt selection.

Bit7: IN1PL: /INT1 Polarity
0: /INT1 input is active low.
1: /INT1 input is active high.

Bits6-4: IN1SL2-0: /INT1 Port Pin Selection Bits
These bits select which Port pin is assigned to /INT1. Note that this pin assignment is independent of the Crossbar; /INT1 will monitor the assigned Port pin without disturbing the peripheral that has been assigned the Port pin via the Crossbar. The Crossbar will not assign the Port pin to a peripheral if it is configured to skip the selected pin (accomplished by setting to '1' the corresponding bit in register XBR0).

IN1SL2-0	/INT1 Port Pin
000	P0.0
001	P0.1
010	P0.2
011	P0.3
100	P0.4
101	P0.5
110	P0.6
111	P0.7

Bit3: IN0PL: /INT0 Polarity
0: /INT0 interrupt is active low.
1: /INT0 interrupt is active high.

Bits2-0: INT0SL2-0: /INT0 Port Pin Selection Bits
These bits select which Port pin is assigned to /INT0. Note that this pin assignment is independent of the Crossbar. /INT0 will monitor the assigned Port pin without disturbing the peripheral that has been assigned the Port pin via the Crossbar. The Crossbar will not assign the Port pin to a peripheral if it is configured to skip the selected pin (accomplished by setting to '1' the corresponding bit in register XBR0).

IN0SL2-0	/INT0 Port Pin
000	P0.0
001	P0.1
010	P0.2
011	P0.3
100	P0.4
101	P0.5
110	P0.6
111	P0.7

8.4. Power Management Modes

The CIP-51 core has two software programmable power management modes: Idle and Stop. Idle mode halts the CPU while leaving the peripherals and clocks active. In Stop mode, the CPU is halted, all interrupts and timers (except the Missing Clock Detector) are inactive, and the system clock is stopped (analog peripherals remain in their selected states). Since clocks are running in Idle mode, power consumption is dependent upon the system clock frequency and the number of peripherals left in active mode before entering Idle. Stop mode consumes the least power. Figure 1.15 describes the Power Control Register (PCON) used to control the CIP-51's power management modes.

Although the CIP-51 has Idle and Stop modes built in (as with any standard 8051 architecture), power management of the entire MCU is better accomplished by enabling/disabling individual peripherals as needed. Each analog peripheral can be disabled when not in use and placed in low power mode. Digital peripherals, such as timers or serial buses, draw little power when they are not in use. Turning off the oscillators lowers power consuption considerably; however a reset is required to restart the MCU.

8.4.1. Idle Mode

Setting the Idle Mode Select bit (PCON.0) causes the CIP-51 to halt the CPU and enter Idle mode as soon as the instruction that sets the bit completes execution. All internal registers and memory maintain their original data. All analog and digital peripherals can remain active during Idle mode.

Idle mode is terminated when an enabled interrupt is asserted or a reset occurs. The assertion of an enabled interrupt will cause the Idle Mode Selection bit (PCON.0) to be cleared and the CPU to resume operation. The pending interrupt will be serviced and the next instruction to be executed after the return from interrupt (RETI) will be the instruction immediately following the one that set the Idle Mode Select bit. If Idle mode is terminated by an internal or external reset, the CIP-51 performs a normal reset sequence and begins program execution at address 0x0000.

If enabled, the Watchdog Timer (WDT) will eventually cause an internal watchdog reset and thereby terminate the Idle mode. This feature protects the system from an unintended permanent shutdown in the event of an inadvertent write to the PCON register. If this behavior is not desired, the WDT may be disabled by software prior to entering the Idle mode if the WDT was initially configured to allow this operation. This provides the opportunity for additional power savings, allowing the system to remain in the Idle mode indefinitely, waiting for an external stimulus to wake up the system. Refer to **Section "16.3. Watchdog Timer Mode" on page 154** for more information on the use and configuration of the WDT.

8.4.2. Stop Mode

Setting the Stop Mode Select bit (PCON.1) causes the CIP-51 to enter Stop mode as soon as the instruction that sets the bit completes execution. In Stop mode the internal oscillator, CPU, and all digital peripherals are stopped; the state of the external oscillator circuit is not affected. Each analog peripheral (including the external oscillator circuit) may be shut down individually prior to entering Stop Mode. Stop mode can only be terminated by an internal or external reset. On reset, the CIP-51 performs the normal reset sequence and begins program execution at address 0x0000.

If enabled, the Missing Clock Detector will cause an internal reset and thereby terminate the Stop mode. The Missing Clock Detector should be disabled if the CPU is to be put to in STOP mode for longer than the MCD timeout of 100 µsec.

Figure 8.15. PCON: Power Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
GF5	GF4	GF3	GF2	GF1	GF0	STOP	IDLE	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x87

Bits7-2: GF5-GF0: General Purpose Flags 5-0.
These are general purpose flags for use under software control.

Bit1: STOP: Stop Mode Select.
Setting this bit will place the CIP-51 in Stop mode. This bit will always be read as 0.
1: CPU goes into Stop mode (turns off internal oscillator).

Bit0: IDLE: Idle Mode Select.
Setting this bit will place the CIP-51 in Idle mode. This bit will always be read as 0.
1: CPU goes into Idle mode (shuts off clock to CPU, but clock to Timers, Interrupts, Serial Ports, and Analog Peripherals are still active).

9. RESET SOURCES

Reset circuitry allows the controller to be easily placed in a predefined default condition. On entry to this reset state, the following occur:

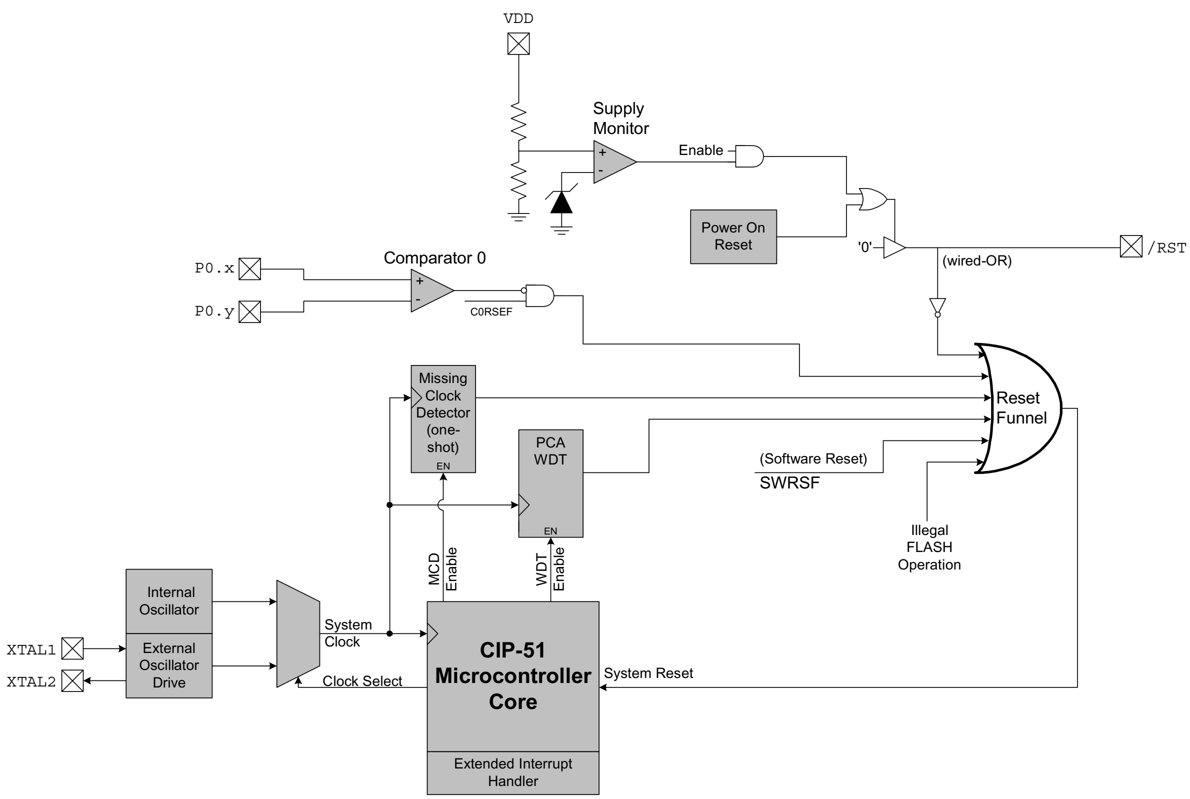
- CIP-51 halts program execution
- Special Function Registers (SFRs) are initialized to their defined reset values
- External Port pins are forced to a known state
- Interrupts and timers are disabled.

All SFRs are reset to the predefined values noted in the SFR detailed descriptions. The contents of internal data memory are unaffected during a reset; any previously stored data is preserved. However, since the stack pointer SFR is reset, the stack is effectively lost even though the data on the stack is not altered.

The Port I/O latches are reset to 0xFF (all logic ones) in open-drain mode. Weak pull-ups are enabled during and after the reset. For VDD Monitor and power-on resets, the /RST pin is driven low until the device exits the reset state.

On exit from the reset state, the program counter (PC) is reset, and the system clock defaults to the internal oscillator. Refer to **Section “11. Oscillators” on page 89** for information on selecting and configuring the system clock source. The Watchdog Timer is enabled with the system clock divided by 12 as its clock source (**Section “16.3. Watchdog Timer Mode” on page 154** details the use of the Watchdog Timer). Once the system clock source is stable, program execution begins at location 0x0000.

Figure 9.1. Reset Sources

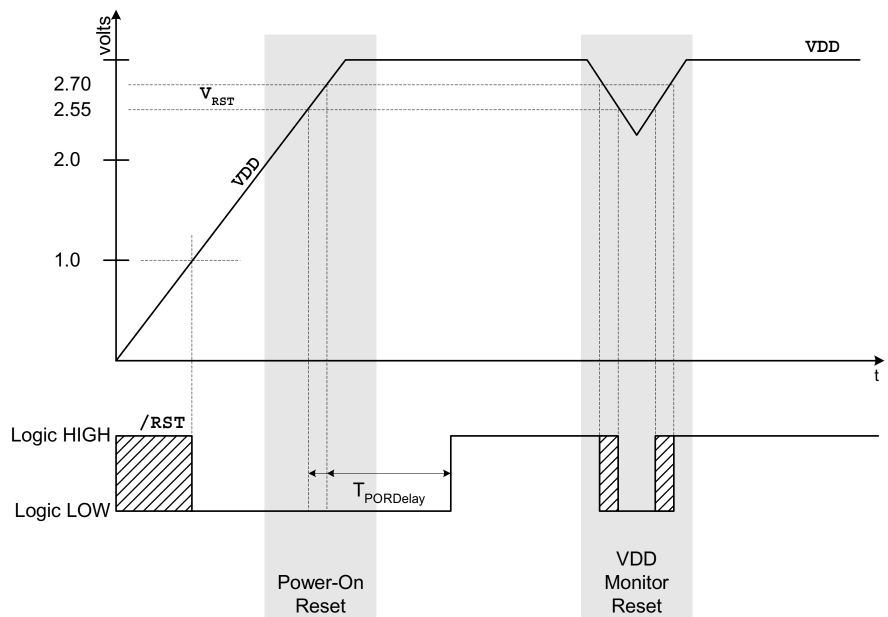


9.1. Power-On Reset

During power-up, the device is held in a reset state and the /RST pin is driven low until VDD settles above V_{RST}. A delay occurs before the device is released from reset; the delay decreases as the VDD ramp time increases (VDD ramp time is defined as how fast VDD ramps from 0 V to 2.7 V). Figure 9.2. plots the power-on and VDD monitor reset timing. The maximum VDD ramp time is 1 ms; slower ramp times may cause the device to be released from reset before VDD reaches the V_{RST} level. For ramp times less than 1 ms, the power-on reset delay ($T_{PORDelay}$) is typically less than 0.3 ms.

On exit from a power-on reset, the PORSF flag (RSTSRC.1) is set by hardware to logic 1. When PORSF is set, all of the other reset flags in the RSTSRC Register are indeterminate (PORSF is cleared by all other resets). Since all resets cause program execution to begin at the same location (0x0000) software can read the PORSF flag to determine if a power-up was the cause of reset. The content of internal data memory should be assumed to be undefined after a power-on reset. The VDD monitor is disabled following a power-on reset.

Figure 9.2. Power-On and VDD Monitor Reset Timing



9.2. Power-Fail Reset / VDD Monitor

When a power-down transition or power irregularity causes VDD to drop below V_{RST}, the power supply monitor will drive the /RST pin low and hold the CIP-51 in a reset state (see Figure 9.2). When VDD returns to a level above V_{RST}, the CIP-51 will be released from the reset state. Note that even though internal data memory contents are not altered by the power-fail reset, it is impossible to determine if VDD dropped below the level required for data retention. If the PORSF flag reads '1', the data may no longer be valid. The VDD monitor is disabled after power-on resets; however its defined state (enabled/disabled) is not altered by any other reset source. For example, if the VDD monitor is enabled and a software reset is performed, the VDD monitor will still be enabled after the reset. The VDD monitor is enabled by writing a '1' to the PORSF bit in register RSTSRC. See Figure 9.2 for VDD monitor timing; note that the reset delay is not incurred after a VDD monitor reset. See Table 9.2 for electrical characteristics of the VDD monitor.

Important Note: Enabling the VDD monitor will immediately generate a system reset. The device will then return from the reset state with the VDD monitor enabled. **Writing a logic '1' to the PORSF flag when the VDD monitor is enabled does not cause a system reset.**

9.3. External Reset

The external /RST pin provides a means for external circuitry to force the device into a reset state. Asserting an active-low signal on the /RST pin generates a reset; an external pull-up and/or decoupling of the /RST pin may be necessary to avoid erroneous noise-induced resets. See Table 9.2 for complete /RST pin specifications. The PINRSF flag (RSTSRC.0) is set on exit from an external reset.

9.4. Missing Clock Detector Reset

The Missing Clock Detector (MCD) is a one-shot circuit that is triggered by the system clock. If the system clock remains high or low for more than 100 µs, the one-shot will time out and generate a reset. After a MCD reset, the MCDRSF flag (RSTSRC.2) will read '1', signifying the MCD as the reset source; otherwise, this bit reads '0'. Writing a '1' to the MCDRSF bit enables the Missing Clock Detector; writing a '0' disables it. The state of the /RST pin is unaffected by this reset.

9.5. Comparator0 Reset

Comparator0 can be configured as a reset source by writing a '1' to the C0RSEF flag (RSTSRC.5). Comparator0 should be enabled and allowed to settle prior to writing to C0RSEF to prevent any turn-on chatter on the output from generating an unwanted reset. The Comparator0 reset is active-low: if the non-inverting input voltage (on CP0+) is less than the inverting input voltage (on CP0-), the device is put into the reset state. After a Comparator0 reset, the C0RSEF flag (RSTSRC.5) will read '1' signifying Comparator0 as the reset source; otherwise, this bit reads '0'. The state of the /RST pin is unaffected by this reset.

9.6. PCA Watchdog Timer Reset

The programmable Watchdog Timer (WDT) function of the Programmable Counter Array (PCA) can be used to prevent software from running out of control during a system malfunction. The PCA WDT function can be enabled or disabled by software as described in **Section "16.3. Watchdog Timer Mode" on page 154**; the WDT is enabled and clocked by SYSCLK / 12 following any reset. If a system malfunction prevents user software from updating the WDT, a reset is generated and the WDTRSF bit (RSTSRC.5) is set to '1'. The state of the /RST pin is unaffected by this reset.

9.7. FLASH Error Reset

If a FLASH read/write/erase or program read targets an illegal address, a system reset is generated. This may occur due to any of the following:

- A FLASH write or erase is attempted above user code space. This occurs when PSWE is set to '1' and a MOVX operation is attempted above the user code space address limit.
- A FLASH read is attempted above user code space. This occurs when a MOVC operation is attempted above the user code space address limit.
- A Program read is attempted above user code space. This occurs when user code attempts to branch to an address above the user code space address limit.

Table 9.1. User Code Space Address Limits

Device	User Code Space Address Limit
C8051F300/1/2/3	0x1DFF
C8051F304	0x0FFF

Table 9.1. User Code Space Address Limits

Device	User Code Space Address Limit
C8051F305	0x07FF

The FERROR bit (RSTSRC.6) is set following a FLASH error reset. The state of the /RST pin is unaffected by this reset.

9.8. Software Reset

Software may force a reset by writing a '1' to the SWRSF bit (RSTSRC.4). The SWRSF bit will read '1' following a software forced reset. The state of the /RST pin is unaffected by this reset.

Table 9.2. Reset Electrical Characteristics

-40°C to +85°C unless otherwise specified.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
/RST Output Low Voltage	I_{OL} = 8.5 mA, VDD = 2.7 V to 3.6 V			0.6	V
/RST Input High Voltage		0.7 x VDD			V
/RST Input Low Voltage				0.3 x VDD	
/RST Input Leakage Current	/RST = 0.0 V		25	40	µA
VDD POR Threshold (V_{RST})		2.40	2.55	2.70	V
Missing Clock Detector Timeout	Time from last system clock rising edge to reset initiation	100	220	500	µs
Reset Time Delay	Delay between release of any reset source and code execution at location 0x0000	5.0			µs
Minimum /RST Low Time to Generate a System Reset		15			µs

Figure 9.3. RSTSRC: Reset Source Register

R	R	R/W	R/W	R	R/W	R/W	R	Reset Value
-	FERROR	C0RSEF	SWRSF	WDTRSF	MCDRSF	PORSF	PINRSF	Variable
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xEF

(Note: Do not use read-modify-write operations (ORL, ANL) on this register)

Bit7: UNUSED. Read = 0. Write = don't care.

Bit6: FERROR: FLASH Error Indicator.
0: Source of last reset was not a FLASH read/write/erase error.
1: Source of last reset was a FLASH read/write/erase error.

Bit5: C0RSEF: Comparator0 Reset Enable and Flag.
Write
0: Comparator0 is not a reset source.
1: Comparator0 is a reset source (active-low).
Read
0: Source of last reset was not Comparator0.
1: Source of last reset was Comparator0.

Bit4: SWRSF: Software Reset Force and Flag.
Write
0: No Effect.
1: Forces a system reset.
Read
0: Source of last reset was not a write to the SWRSF bit.
1: Source of last was a write to the SWRSF bit.

Bit3: WDTRSF: Watchdog Timer Reset Flag.
0: Source of last reset was not a WDT timeout.
1: Source of last reset was a WDT timeout.

Bit2: MCDRSF: Missing Clock Detector Flag.
Write:
0: Missing Clock Detector disabled.
1: Missing Clock Detector enabled; triggers a reset if a missing clock condition is detected.
Read:
0: Source of last reset was not a Missing Clock Detector timeout.
1: Source of last reset was a Missing Clock Detector timeout.

Bit1: PORSF: Power-On Reset Force and Flag.
This bit is set anytime a power-on reset occurs. This may be due to a true power-on reset or a VDD monitor reset. In either case, data memory should be considered indeterminate following the reset. Writing this bit enables/disables the VDD monitor.
Write:
0: VDD monitor disabled.
1: VDD monitor enabled.
Read:
0: Last reset was not a power-on or VDD monitor reset.
1: Last reset was a power-on or VDD monitor reset; all other reset flags indeterminate.

Bit0: PINRSF: HW Pin Reset Flag.
0: Source of last reset was not /RST pin.
1: Source of last reset was /RST pin.

Notes

10. FLASH MEMORY

On-chip, re-programmable FLASH memory is included for program code and non-volatile data storage. The FLASH memory can be programmed in-system, a single byte at a time, through the C2 interface or by software using the MOVX instruction. Once cleared to logic 0, a FLASH bit must be erased to set it back to logic 1. FLASH bytes would typically be erased (set to 0xFF) before being reprogrammed. The write and erase operations are automatically timed by hardware for proper execution; data polling to determine the end of the write/erase operation is not required. Code execution is stalled during a FLASH write/erase operation. Refer to Table 10.1 for complete FLASH memory electrical characteristics.

10.1. Programming The FLASH Memory

The simplest means of programming the FLASH memory is through the C2 interface using programming tools provided by Cygnal or a third party vendor. This is the only means for programming a non-initialized device. For details on the C2 commands to program FLASH memory, see **Section “17. C2 Interface” on page 161**.

To ensure the integrity of FLASH contents, it is strongly recommended that the on-chip VDD Monitor be enabled in any system that includes code that writes and/or erases FLASH memory from software.

10.1.1. FLASH Lock and Key Functions

FLASH writes and erases by user software are protected with a lock and key function; FLASH reads by user software are unrestricted. The FLASH Lock and Key Register (FLKEY) must be written with the correct key codes, in sequence, before FLASH operations may be performed. The key codes are: 0xA5, 0xF1. The timing does not matter, but the codes must be written in order. If the key codes are written out of order, or the wrong codes are written, FLASH writes and erases will be disabled until the next system reset. FLASH writes and erases will also be disabled if a FLASH write or erase is attempted before the key codes have been written properly. The FLASH lock resets after each write or erase; the key codes must be written again before a following FLASH operation can be performed. The FLKEY register is detailed in Figure 10.3.

10.1.2. FLASH Erase Procedure

The FLASH memory can be programmed by software using the MOVX instruction with the address and data byte to be programmed provided as normal operands. Before writing to FLASH memory using MOVX, FLASH write operations must be enabled by: (1) setting the PSWE Program Store Write Enable bit (PSCTL.0) to logic 1 (this directs the MOVX writes to target FLASH memory); and (2) Writing the FLASH key codes in sequence to the FLASH Lock register (FLKEY). The PSWE bit remains set until cleared by software.

A write to FLASH memory can clear bits but cannot set them; only an erase operation can set bits in FLASH. **A byte location to be programmed should be erased before a new value is written.** The 8k byte FLASH memory is organized in 512-byte pages. The erase operation applies to an entire page (setting all bytes in the page to 0xFF). To erase an entire 512-byte page, perform the following steps:

Step 5. Disable interrupts (recommended).
Step 6. Set the Program Store Erase Enable bit (PSEE in the PSCTL register).
Step 7. Set the Program Store Write Enable bit (PSWE in the PSCTL register).
Step 8. Write the first key code to FLKEY: 0xA5.
Step 9. Write the second key code to FLKEY: 0xF1.
Step 10. Using the MOVX instruction, write a data byte to any location within the 512-byte page to be erased.

10.1.3. FLASH Write Procedure

FLASH bytes are programmed by software with the following sequence:

Step 1. Disable interrupts (recommended).
Step 2. Erase the 512-byte FLASH page containing the target location, as described in **Section 10.1.2**.
Step 3. Set the PSWE bit in PSCTL.
Step 4. Clear the PSEE bit in PSCTL.
Step 5. Write the first key code to FLKEY: 0xA5.
Step 6. Write the second key code to FLKEY: 0xF1.
Step 7. Using the MOVX instruction, write a single data byte to the desired location within the 512-byte sector.

Steps 5-7 must be repeated for each byte to be written. After FLASH writes are complete, PSWE should be cleared so that MOVX instructions do not target program memory.

Table 10.1. FLASH Electrical Characteristics

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
FLASH Size	C8051F300/1/2/3	8192†			bytes
FLASH Size	C8051F304	4096			bytes
FLASH Size	C8051F305	2048			bytes
Endurance		20k	100k		Erase/Write
Erase Cycle Time	25 MHz System Clock	10	15	20	ms
Write Cycle Time	25 MHz System Clock	40	55	70	µs
SYSCLK Frequency (FLASH writes from application code)		100			kHz

†Note: 512 bytes at location 0x1E00 to 0x1FFF are reserved for factory use

10.2. Non-volatile Data Storage

The FLASH memory can be used for non-volatile data storage as well as program code. This allows data such as calibration coefficients to be calculated and stored at run time. Data is written using the MOVX instruction and read using the MOVC instruction.

10.3. Security Options

The CIP-51 provides security options to protect the FLASH memory from inadvertent modification by software as well as to prevent the viewing of proprietary program code and constants. The Program Store Write Enable (bit PSWE in register PSCTL) and the Program Store Erase Enable (bit PSEE in register PSCTL) bits protect the FLASH memory from accidental modification by software. PSWE must be explicitly set to '1' before software can modify the FLASH memory; both PSWE and PSEE must be set to '1' before software can erase FLASH memory. Additional security features prevent proprietary program code and data constants from being read or altered across the C2 interface.

A security lock byte stored at the last byte of FLASH user space protects the FLASH program memory from being read or altered across the C2 interface. See Table 10.2 for the security byte description; see Figure 10.1 for a program memory map and the security byte locations for each device.

Table 10.2. Security Byte Decoding

Bits	Description
7-4	**Write Lock:** Clearing any of these bits to logic 0 prevents all FLASH memory from being written or page-erased across the C2 interface
3-0	**Read/Write Lock:** Clearing any of these bits to logic 0 prevents all FLASH memory from being read, written, or page-erased across the C2 interface.

The lock bits can always be read and cleared to logic 0 regardless of the security settings.

Important note: The only means of removing a lock (write or read/write) once set is to erase the entire program memory space via a C2 Device Erase command.

Figure 10.1. FLASH Program Memory Map

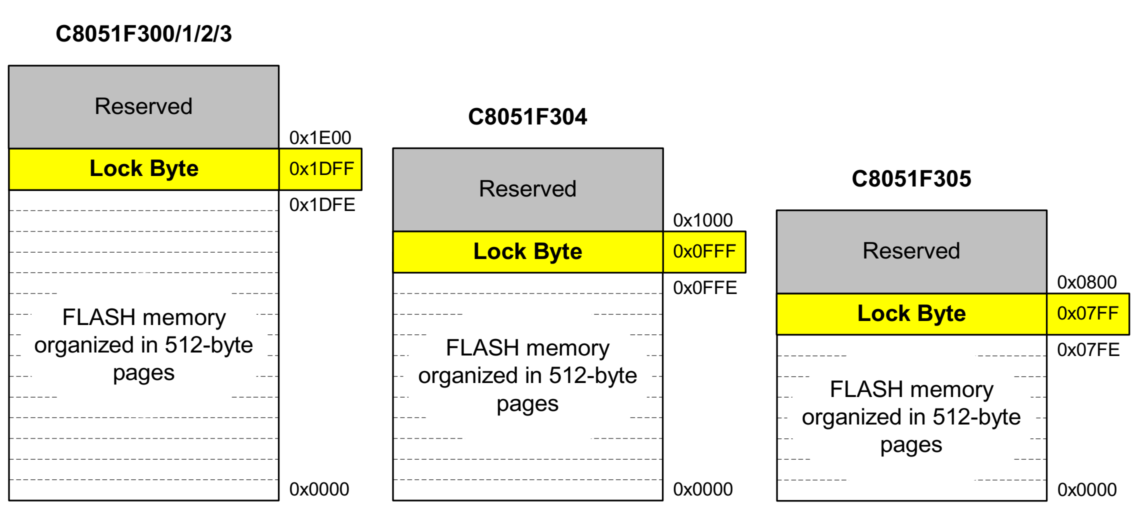


Figure 10.2. PSCTL: Program Store R/W Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	-	-	-	-	PSEE	PSWE	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x8F

Bits7-2: UNUSED: Read = 000000b, Write = don't care.

Bit1: PSEE: Program Store Erase Enable
Setting this bit (in combination with PSWE) allows an entire page of FLASH program memory to be erased. If this bit is logic 1 and FLASH writes are enabled (PSWE is logic 1), a write to FLASH memory using the MOVX instruction will erase the entire page that contains the location addressed by the MOVX instruction. The value of the data byte written does not matter.
0: FLASH program memory erasure disabled.
1: FLASH program memory erasure enabled.

Bit0: PSWE: Program Store Write Enable
Setting this bit allows writing a byte of data to the FLASH program memory using the MOVX instruction. The FLASH location should be erased before writing data.
0: Writes to FLASH program memory disabled.
1: Writes to FLASH program memory enabled; the MOVX instruction targets FLASH memory.

Figure 10.3. FLKEY: FLASH Lock and Key Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xB7

Bits7-0: FLKEY: FLASH Lock and Key Register
Write:
This register must be written to before FLASH writes or erases can be performed. FLASH remains locked until this register is written to with the following key codes: 0xA5, 0xF1. The timing of the writes does not matter, as long as the codes are written in order. The key codes must be written for each FLASH write or erase operation. FLASH will be locked until the next system reset if the wrong codes are written or if a FLASH operation is attempted before the codes have been written correctly.
Read:
When read, bits 1-0 indicate the current FLASH lock state.
00: FLASH is write/erase locked.
01: The first key code has been written (0xA5).
10: FLASH is unlocked (writes/erases allowed).
11: FLASH writes/erases disabled until the next reset.

Figure 10.4. FLSCL: FLASH Scale Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
FOSE	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	10000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xB6

Bits7: FOSE: FLASH One-shot Enable
This bit enables the 50 ns FLASH read one-shot. When the FLASH one-shot disabled, the FLASH sense amps are enabled for a full clock cycle during FLASH reads.
0: FLASH one-shot disabled.
1: FLASH one-shot enabled.

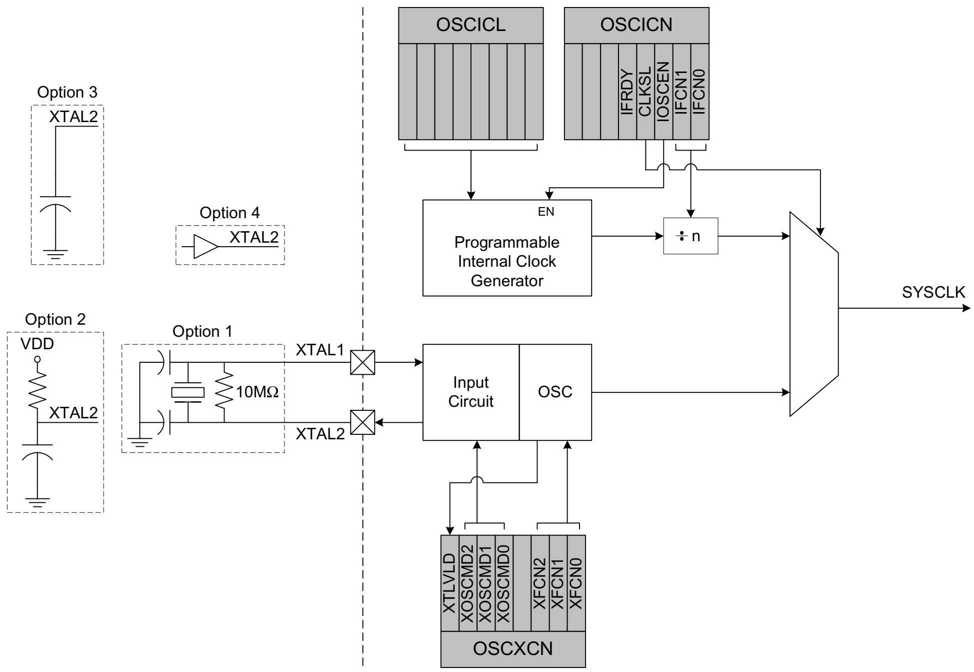
Bits6-0: RESERVED. Read = 0. Must Write 0.

Notes

11. OSCILLATORS

C8051F300/1/2/3/4/5 devices include a programmable internal oscillator and an external oscillator drive circuit. The internal oscillator can be enabled/disabled and calibrated using the OSCICN and OSCICL registers, as shown in Figure 11.1. The system clock can be sourced by the external oscillator circuit, the internal oscillator, or a scaled version of the internal oscillator. The internal oscillator's electrical specifications are given in Table 11.1 on page 91.

Figure 11.1. Oscillator Diagram



11.1. Programmable Internal Oscillator

All C8051F300/1/2/3/4/5 devices include a programmable internal oscillator that defaults as the system clock after a system reset. The internal oscillator period can be adjusted via the OSCICL register as defined by Figure 11.2. On C8051F300/1 devices, OSCICL is factory calibrated to obtain a 24.5 MHz frequency. On C8051F302/3/4/5 devices, the oscillator frequency is a nominal 20 MHz and may vary ±20% from device-to-device.

Electrical specifications for the precision internal oscillator are given in Table 11.1 on page 91. The programmed internal oscillator frequency must not exceed 25 MHz. Note that the system clock may be derived from the programmed internal oscillator divided by 1, 2, 4, or 8, as defined by the IFCN bits in register OSCICN. The divide value defaults to 8 following a reset.

Figure 11.2. OSCICL: Internal Oscillator Calibration Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-								Variable
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xB3

Bit7: UNUSED. Read = 0. Write = don’t care.
Bits 6-0: OSCICL: Internal Oscillator Calibration Register.
This register calibrates the internal oscillator period. The reset value for OSCICL defines the internal oscillator base frequency. On C8051F300/1 devices, the reset value is factory calibrated to generate an internal oscillator frequency of 24.5 MHz.

Figure 11.3. OSCICN: Internal Oscillator Control Register

R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	Reset Value
-	-	-	IFRDY	CLKSL	IOSCEN	IFCN1	IFCN0	00010100
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xB2

Bits7-5: UNUSED. Read = 000b, Write = don't care.
Bit4: IFRDY: Internal Oscillator Frequency Ready Flag.
0: Internal Oscillator is not running at programmed frequency.
1: Internal Oscillator is running at programmed frequency.
Bit3: CLKSL: System Clock Source Select Bit.
0: SYSCLK derived from the Internal Oscillator, and scaled as per the IFCN bits.
1: SYSCLK derived from the External Oscillator circuit.
Bit2: IOSCEN: Internal Oscillator Enable Bit.
0: Internal Oscillator Disabled.
1: Internal Oscillator Enabled.
Bits1-0: IFCN1-0: Internal Oscillator Frequency Control Bits.
00: SYSCLK derived from Internal Oscillator divided by 8.
01: SYSCLK derived from Internal Oscillator divided by 4.
10: SYSCLK derived from Internal Oscillator divided by 2.
11: SYSCLK derived from Internal Oscillator divided by 1.

Table 11.1. Internal Oscillator Electrical Characteristics

-40°C to +85°C unless otherwise specified

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Calibrated Internal Oscillator Frequency	C8051F300/1 devices	24	24.5	25	MHz
Uncalibrated Internal Oscillator Frequency	C8051F302/3/4/5 devices	16	20	24	MHz
Internal Oscillator Supply Current (from VDD)	OSCICN.2 = 1		450		µA

11.2. External Oscillator Drive Circuit

The external oscillator circuit may drive an external crystal, ceramic resonator, capacitor, or RC network. A CMOS clock may also provide a clock input. For a crystal or ceramic resonator configuration, the crystal/resonator must be wired across the XTAL1 and XTAL2 pins as shown in Option 1 of Figure 11.1. A 10 MΩ resistor also must be wired across the XTAL2 and XTAL1 pins for the crystal/resonator configuration. In RC, capacitor, or CMOS clock configuration, the clock source should be wired to the XTAL2 pin as shown in Option 2, 3, or 4 of Figure 11.1. The type of external oscillator must be selected in the OSCXCN register, and the frequency control bits (XFCN) must be selected appropriately (see Figure 11.4).

Important Note on External Oscillator Usage: Port pins must be configured when using the external oscillator circuit. When the external oscillator drive circuit is enabled in crystal/resonator mode, Port pins P0.2 and P0.3 are occupied as XTAL1 and XTAL2 respectively. When the external oscillator drive circuit is enabled in capacitor, RC, or CMOS clock mode, Port pin P0.3 is occupied as XTAL2. The Port I/O Crossbar should be configured to skip the occupied Port pins; see **Section "12.1. Priority Crossbar Decoder" on page 96** for Crossbar configuration. Additionally, when using the external oscillator circuit in crystal/resonator, capacitor, or RC mode, the associated Port pins should be configured as **analog inputs**. In CMOS clock mode, the associated pin should be configured as a **digital input**. See **Section "12.2. Port I/O Initialization" on page 98** for details on Port input mode selection.

11.3. System Clock Selection

The CLKSL bit in register OSCICN selects which oscillator is used as the system clock. CLKSL must be set to '1' for the system clock to run from the external oscillator; however the external oscillator may still clock peripherals (timers, PCA) when the internal oscillator is selected as the system clock. The system clock may be switched on-the-fly between the internal and external oscillator, so long as the selected oscillator is enabled and has settled. The internal oscillator requires little start-up time and may be enabled and selected as the system clock in the same write to OSCICN. External crystals and ceramic resonators typically require a start-up time before they are settled and ready for use as the system clock. The Crystal Valid Flag (XTLVLD in register OSCXCN) is set to '1' by hardware when the external oscillator is settled. To avoid reading a false XTLVLD, in crystal mode software should delay at least 1 ms between enabling the external oscillator and checking XTLVLD. RC and C modes typically require no startup time.

Figure 11.4. OSCXCN: External Oscillator Control Register

R	R/W	R/W	R/W	R	R/W	R/W	R/W	Reset Value
XTLVLD	XOSCMD2	XOSCMD1	XOSCMD0	-	XFCN2	XFCN1	XFCN0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xB1

Bit7: XTLVLD: Crystal Oscillator Valid Flag.
(Read only when XOSCMD = 11x.)
0: Crystal Oscillator is unused or not yet stable.
1: Crystal Oscillator is running and stable.

Bits6-4: XOSCMD2-0: External Oscillator Mode Bits.
00x: External Oscillator circuit off.
010: External CMOS Clock Mode.
011: External CMOS Clock Mode with divide by 2 stage.
100: RC Oscillator Mode with divide by 2 stage.
101: Capacitor Oscillator Mode with divide by 2 stage.
110: Crystal Oscillator Mode.
111: Crystal Oscillator Mode with divide by 2 stage.

Bit3: RESERVED. Read = 0, Write = don't care.

Bits2-0: XFCN2-0: External Oscillator Frequency Control Bits.
000-111: See table below:

XFCN	Crystal (XOSCMD = 11x)	RC (XOSCMD = 10x)	C (XOSCMD = 10x)
000	f ≤ 32kHz	f ≤ 25kHz	K Factor = 0.87
001	32kHz < f ≤ 84kHz	25kHz < f ≤ 50kHz	K Factor = 2.6
010	84kHz < f ≤ 225kHz	50kHz < f ≤ 100kHz	K Factor = 7.7
011	225kHz < f ≤ 590kHz	100kHz < f ≤ 200kHz	K Factor = 22
100	590kHz < f ≤ 1.5MHz	200kHz < f ≤ 400kHz	K Factor = 65
101	1.5MHz < f ≤ 4MHz	400kHz < f ≤ 800kHz	K Factor = 180
110	4MHz < f ≤ 10MHz	800kHz < f ≤ 1.6MHz	K Factor = 664
111	10MHz < f ≤ 30MHz	1.6MHz < f ≤ 3.2MHz	K Factor = 1590

CRYSTAL MODE (Circuit from Figure 11.1, Option 1; XOSCMD = 11x)
Choose XFCN value to match crystal frequency.

RC MODE (Circuit from Figure 11.1, Option 2; XOSCMD = 10x)
Choose XFCN value to match frequency range:
$\mathbf{f = 1.23(10^3) / (R * C)}$, where
f = frequency of oscillation in MHz
C = capacitor value in pF
R = Pull-up resistor value in kΩ

C MODE (Circuit from Figure 11.1, Option 3; XOSCMD = 10x)
Choose K Factor (KF) for the oscillation frequency desired:
$\mathbf{f = KF / (C * VDD)}$, where
f = frequency of oscillation in MHz
C = capacitor value the XTAL2 pin in pF
VDD = Power Supply on MCU in volts

11.4. External Crystal Example

If a crystal or ceramic resonator is used as an external oscillator source for the MCU, the circuit should be configured as shown in Figure 11.1, Option 1. The External Oscillator Frequency Control value (XFCN) should be chosen from the Crystal colum of the table in Figure 11.4 (OSCXCN register). For example, an 11.0592 MHz crystal requires an XFCN setting of 111b.

When the crystal oscillator is first enabled, the oscillator amplitude detection circuit requires a settling time to achieve proper bias. Introducing a delay of 1 ms between enabling the oscillator and checking the XTLVLD bit will prevent a premature switch to the external oscillator as the system clock. Switching to the external oscillator before the crystal oscillator has stabilized can result in unpredictable behavior. The recommended procedure is:

Step 1. Enable the external oscillator.
Step 2. Wait at least 1 ms.
Step 3. Poll for XTLVLD => '1'.
Step 4. Switch the system clock to the external oscillator.

Important Note on External Crystals: Crystal oscillator circuits are quite sensitive to PCB layout. The crystal should be placed as close as possible to the XTAL pins on the device. The traces should be as short as possible and shielded with ground plane from any other traces which could introduce noise or interference.

11.5. External RC Example

If an RC network is used as an external oscillator source for the MCU, the circuit should be configured as shown in Figure 11.1, Option 2. The capacitor should be no greater than 100 pF; however for very small capacitors, the total capacitance may be dominated by parasitic capacitance in the PCB layout. To determine the required External Oscillator Frequency Control value (XFCN) in the OSCXCN Register, first select the RC network value to produce the desired frequency of oscillation. If the frequency desired is 100 kHz, let R = 246 kΩ and C = 50 pF:

$f = 1.23(10^3) / RC = 1.23 (10^3) / [246 * 50] = 0.1$ MHz = 100 kHz

Referring to the table in Figure 11.4, the required XFCN setting is 010b.

11.6. External Capacitor Example

If a capacitor is used as an external oscillator for the MCU, the circuit should be configured as shown in Figure 11.1, Option 3. The capacitor should be no greater than 100 pF; however for very small capacitors, the total capacitance may be dominated by parasitic capacitance in the PCB layout. To determine the required External Oscillator Frequency Control value (XFCN) in the OSCXCN Register, select the capacitor to be used and find the frequency of oscillation from the equations below. Assume VDD = 3.0 V and C = 50 pF:

f = KF / (C * VDD) = KF / (50 * 3) MHz
f = KF / 150 MHz

If a frequency of roughly 150 kHz is desired, select the K Factor from the table in Figure 11.4 as KF = 22:

f = 22 / 150 = 0.146 MHz, or 146 kHz

Therefore, the XFCN value to use in this example is 011b.

Notes

12. PORT INPUT/OUTPUT

Digital and analog resources are available through a byte-wide digital I/O Port, Port0. Each of the Port pins can be defined as general-purpose I/O (GPIO), analog input, or assigned to one of the internal digital resources as shown in Figure 12.3. The designer has complete control over which functions are assigned, limited only by the number of physical I/O pins. This resource assignment flexibility is achieved through the use of a Priority Crossbar Decoder. Note that the state of a Port I/O pin can always be read in the corresponding Port latch, regardless of the Crossbar settings.

The Crossbar assigns the selected internal digital resources to the I/O pins based on the Priority Decoder (Figure 12.3 and Figure 12.4). The registers XBR0, XBR1, and XBR2, defined in Figure 12.5, Figure 12.6, and Figure 12.7 are used to select internal digital functions.

All Port I/Os are 5 V tolerant (refer to Figure 12.2 for the Port cell circuit). The Port I/O cells are configured as either push-pull or open-drain in the Port0 Output Mode register (P0MDOUT). Complete Electrical Specifications for Port I/O are given in Table 12.1 on page 102.

Figure 12.1. Port I/O Functional Block Diagram

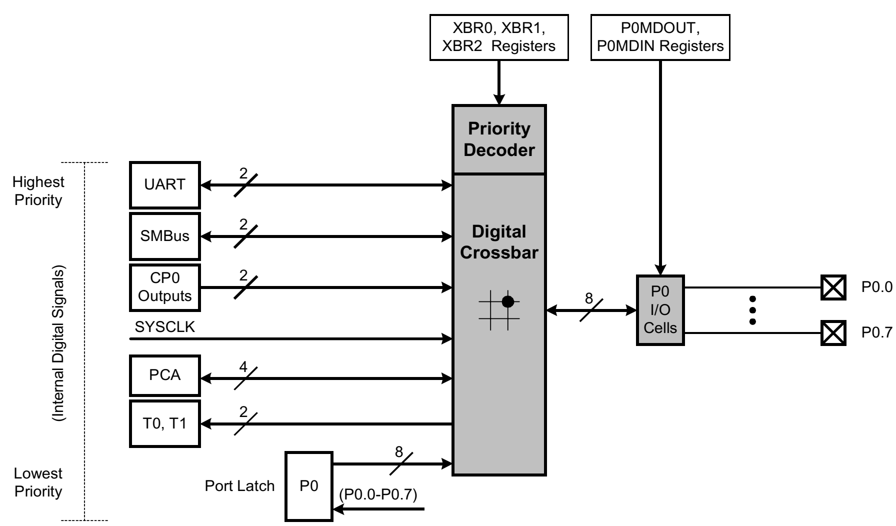
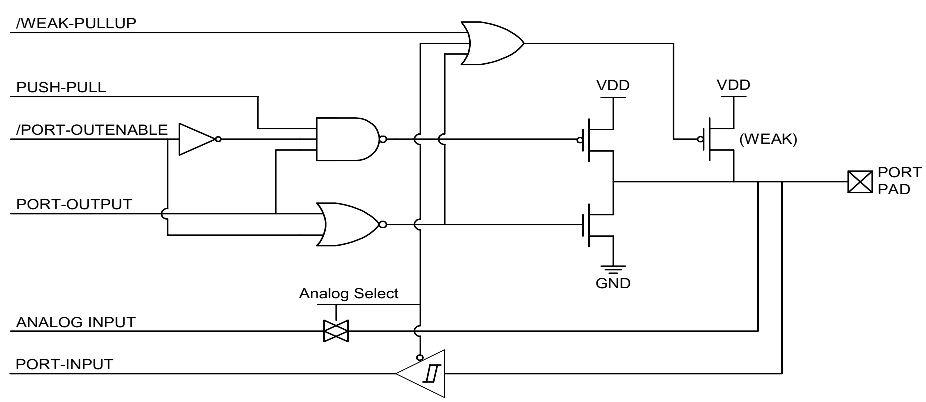


Figure 12.2. Port I/O Cell Block Diagram



12.1. Priority Crossbar Decoder

The Priority Crossbar Decoder (Figure 12.3) assigns a priority to each I/O function, starting at the top with UART0. When a digital resource is selected, the least-significant unassigned Port pin is assigned to that resource (excluding UART0, which is always at pins 4 and 5). If a Port pin is assigned, the Crossbar skips that pin when assigning the next selected resource. Additionally, the Crossbar will skip Port pins whose associated bits in the XBR0 register are set. The XBR0 register allows software to skip Port pins that are to be used for analog input or GPIO.

Important Note on Crossbar Configuration: If a Port pin is claimed by a peripheral without use of the Crossbar, its corresponding XBR0 bit should be set. This applies to P0.0 if VREF is enabled, P0.3 and/or P0.2 if the external oscillator circuit is enabled, P0.6 if the ADC is configured to use the external conversion start signal (CNVSTR), and any selected ADC or Comparator inputs. The Crossbar skips selected pins as if they were already assigned, and moves to the next unassigned pin. Figure 12.3 shows the Crossbar Decoder priority with no Port pins skipped (XBR0 = 0x00); Figure 12.4 shows the Crossbar Decoder priority with pins 6 and 2 skipped (XBR0 = 0x44).

Figure 12.3. Crossbar Priority Decoder with XBR0 = 0x00

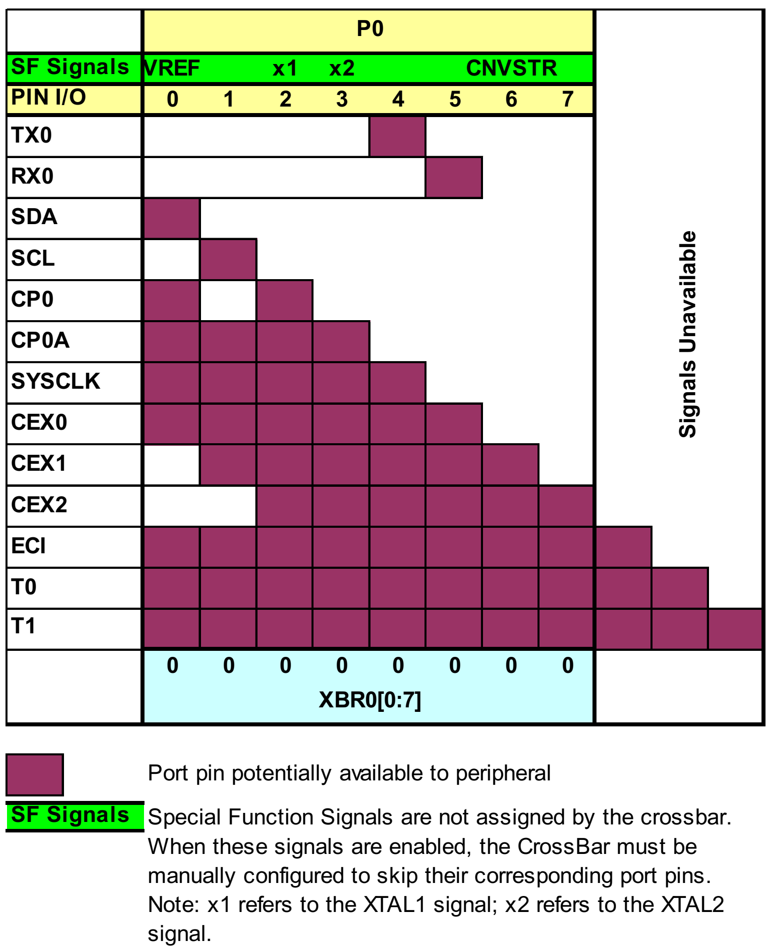
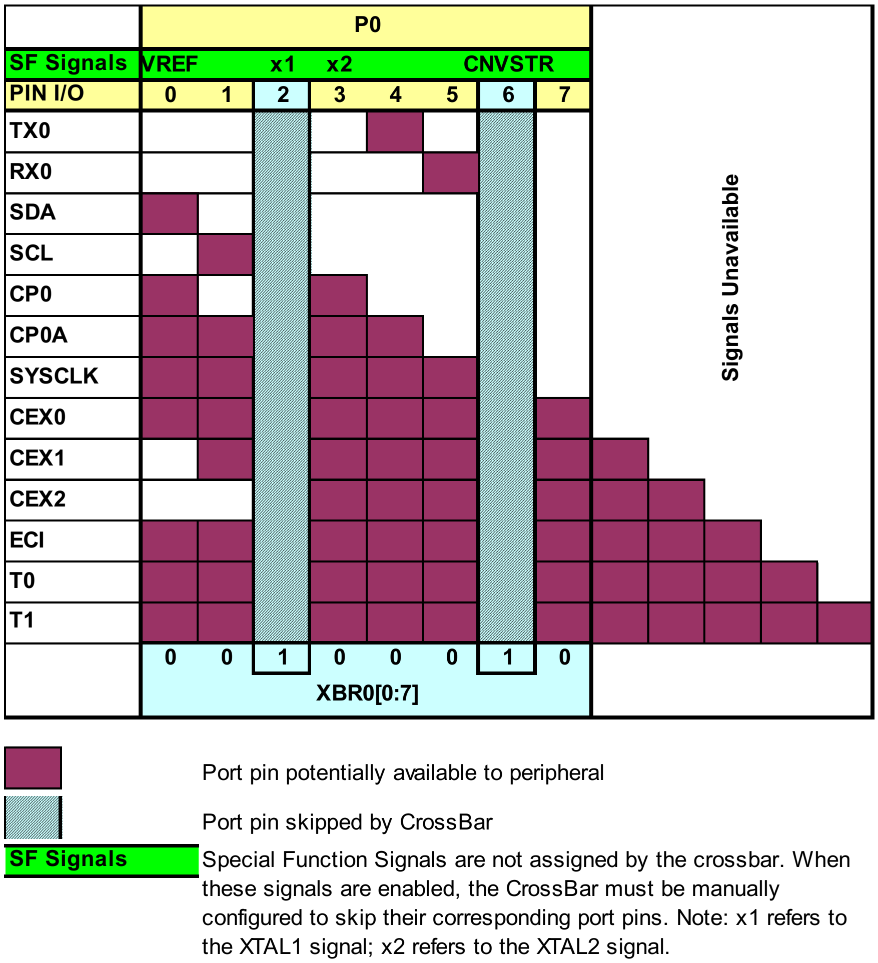


Figure 12.4. Crossbar Priority Decoder with XBR0 = 0x44



Registers XBR1 and XBR2 are used to assign the digital I/O resources to the physical I/O Port pins. Note that when the SMBus is selected, the Crossbar assigns both pins associated with the SMBus (SDA and SCL). Either or both of the UART signals may be selected by the Crossbar. UART0 pin assignments are fixed for bootloading purposes: when UART TX0 is selected, it is always assigned to P0.4; when UART RX0 is selected, it is always assigned to P0.5. Standard Port I/Os appear contiguously after the prioritized functions have been assigned. For example, if assigned functions that take the first 3 Port I/O (P0.[2:0]), 5 Port I/O are left for analog or GPIO use.

12.2. Port I/O Initialization

Port I/O initialization consists of the following steps:

Step 1. Select the input mode (analog or digital) for all Port pins, using the Port0 Input Mode register (P0MDIN).
Step 2. Select the output mode (open-drain or push-pull) for all Port pins, using the Port0 Output Mode register (P0MDOUT).
Step 3. Set XBR0 to skip any pins selected as analog inputs or special functions.
Step 4. Assign Port pins to desired peripherals.
Step 5. Enable the Crossbar.

All Port pins must be configured as either analog or digital inputs. Any pins to be used as Comparator or ADC inputs should be configured as an analog inputs. When a pin is configured as an analog input, its weak pull-up, digital driver, and digital receiver is disabled. This process saves power and reduces noise on the analog input. Pins configured as digital inputs may still be used by analog peripherals; however this practice is not recommended.

Additionally, all analog input pins should be configured to be skipped by the Crossbar (accomplished by setting the associated bits in XBR0). Port input mode is set in the P0MDIN register, where a ‘1’ indicates a digital input, and a ‘0’ indicates an analog input. All pins default to digital inputs on reset. See Figure 12.9 for the P0MDIN register details.

The output driver characteristics of the I/O pins are defined using the Port0 Output Mode register P0MDOUT (see Figure 12.10). Each Port Output driver can be configured as either open drain or push-pull. This selection is required even for the digital resources selected in the XBRn registers, and is not automatic. The only exception to this is the SMBus (SDA, SCL) pins, which are configured as open-drain regardless of the P0MDOUT settings. When the WEAKPUD bit in XBR2 is ‘0’, a weak pull-up is enabled for all Port I/O configured as open-drain. WEAKPUD does not affect the push-pull Port I/O. Furthermore, the weak pull-up is turned off on an open-drain output that is driving a ‘0’ to avoid unnecessary power dissipation.

Registers XBR0, XBR1 and XBR2 must be loaded with the appropriate values to select the digital I/O functions required by the design. Setting the XBARE bit in XBR2 to ‘1’ enables the Crossbar. Until the Crossbar is enabled, the external pins remain as standard digital inputs (output drivers disabled) regardless of the XBRn Register settings. For given XBRn Register settings, one can determine the I/O pin-out using the Priority Decode Table; as an alternative, the Configuration Wizard utility of the Cygnal IDE software will determine the Port I/O pin-assignments based on the XBRn Register settings.

Figure 12.5. XBR0: Port I/O Crossbar Register 0

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	XSKP6	XSKP5	XSKP4	XSKP3	XSKP2	XSKP1	XSKP0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE1

Bit7: UNUSED. Read = 0b; Write = don't care.
Bits6-0: XSKP[6:0]: Crossbar Skip Enable Bits
These bits select Port pins to be skipped by the Crossbar Decoder. Port pins used as analog inputs (for ADC or Comparator) or used as special functions (VREF input, external oscillator circuit, CNVSTR input) should be skipped by the Crossbar.
0: Corresponding P0.n pin is not skipped by the Crossbar.
1: Corresponding P0.n pin is skipped by the Crossbar.

Figure 12.6. XBR1: Port I/O Crossbar Register 1

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
PCA0ME		CP0AOEN	CP0OEN	SYSCKE	SMB0OEN	URX0EN	UTX0EN	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE2

Bits7-6: PCA0ME: PCA Module I/0 Enable Bits
00: All PCA I/O unavailable at Port pins.
01: CEX0 routed to Port pin.
10: CEX0, CEX1 routed to Port pins.
11: CEX0, CEX1, CEX2 routed to Port pins.
Bit5: CP0AOEN: Comparator0 Asynchronous Output Enable
0: Asynchronous CP0 unavailable at Port pin.
1: Asynchronous CP0 routed to Port pin.
Bit4: CP0OEN: Comparator0 Output Enable
0: CP0 unavailable at Port pin.
1: CP0 routed to Port pin.
Bit3: SYSCKE: /SYSCLK Output Enable
0: /SYSCLK unavailable at Port pin.
1: /SYSCLK output routed to Port pin.
Bit2: SMB0OEN: SMBus I/O Enable
0: SMBus I/O unavailable at Port pins.
1: SDA, SCL routed to Port pins.
Bit1: URX0EN: UART RX Enable
0: UART RX0 unavailable at Port pin.
1: UART RX0 routed to Port pin P0.5.
Bit0: UTX0EN: UART TX Output Enable
0: UART TX0 unavailable at Port pin.
1: UART TX0 routed to Port pin P0.4.

Figure 12.7. XBR2: Port I/O Crossbar Register 2

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
WEAKPUD	XBARE	-	-	-	T1E	T0E	ECIE	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE3

Bit7: WEAKPUD: Port I/O Weak Pull-up Disable.
0: Weak Pull-ups enabled (except for Ports whose I/O are configured as push-pull).
1: Weak Pull-ups disabled.
Bit6: XBARE: Crossbar Enable.
0: Crossbar disabled.
1: Crossbar enabled.
Bits5-3: UNUSED: Read=000b. Write = don't care.
Bit2: T1E: T1 Enable.
0: T1 unavailable at Port pin.
1: T1 routed to Port pin.
Bit1: T0E: T0 Enable.
0: T0 unavailable at Port pin.
1: T0 routed to Port pin.
Bit0: ECIE: PCA0 Counter Input Enable.
0: ECI unavailable at Port pin.
1: ECI routed to Port pin.

12.3. General Purpose Port I/O

Port pins that remain unassigned by the Crossbar and are not used by analog peripherals can be used for general purpose I/O. Port0 is accessed through a corresponding special function register (SFR) that is both byte addressable and bit addressable. When writing to a Port, the value written to the SFR is latched to maintain the output data value at each pin. When reading, the logic levels of the Port's input pins are returned regardless of the XBRn settings (i.e., even when the pin is assigned to another signal by the Crossbar, the Port register can always read its corresponding Port I/O pin). The exception to this is the execution of the read-modify-write instructions. The read-modify-write instructions when operating on a Port SFR are the following: ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ and MOV, CLR or SET, when the destination is an individual bit in a Port SFR. For these instructions, the value of the register (not the pin) is read, modified, and written back to the SFR.

Figure 12.8. P0: Port0 Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	11111111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0x80

Bits7-0: P0.[7:0]
Write - Output appears on I/O pins per XBR0, XBR1, and XBR2 Registers
0: Logic Low Output.
1: Logic High Output (open-drain if corresponding P0MDOUT.n bit = 0)
Read - Always reads '1' if selected as analog input in register P0MDIN. Directly reads Port pin when configured as digital input.
0: P0.n pin is logic low.
1: P0.n pin is logic high.

Figure 12.9. P0MDIN: Port0 Input Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								11111111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0xF1

Bits7-0: Input Configuration Bits for P0.7-P0.0 (respectively)
Port pins configured as analog inputs have their weak pull-up, digital driver, and digital receiver disabled.
0: Corresponding P0.n pin is configured as an analog input.
1: Corresponding P0.n pin is configured as a digital input.

Figure 12.10. P0MDOUT: Port0 Output Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xA4

Bits7-0: Output Configuration Bits for P0.7-P0.0 (respectively): ignored if corresponding bit in register P0MDIN is logic 0.
0: Corresponding P0.n Output is open-drain.
1: Corresponding P0.n Output is push-pull.

(Note: When SDA and SCL appear on any of the Port I/O, each are open-drain regardless of the value of P0MDOUT).

Table 12.1. Port I/O DC Electrical Characteristics

VDD = 2.7 to 3.6V, -40°C to +85°C unless otherwise specified

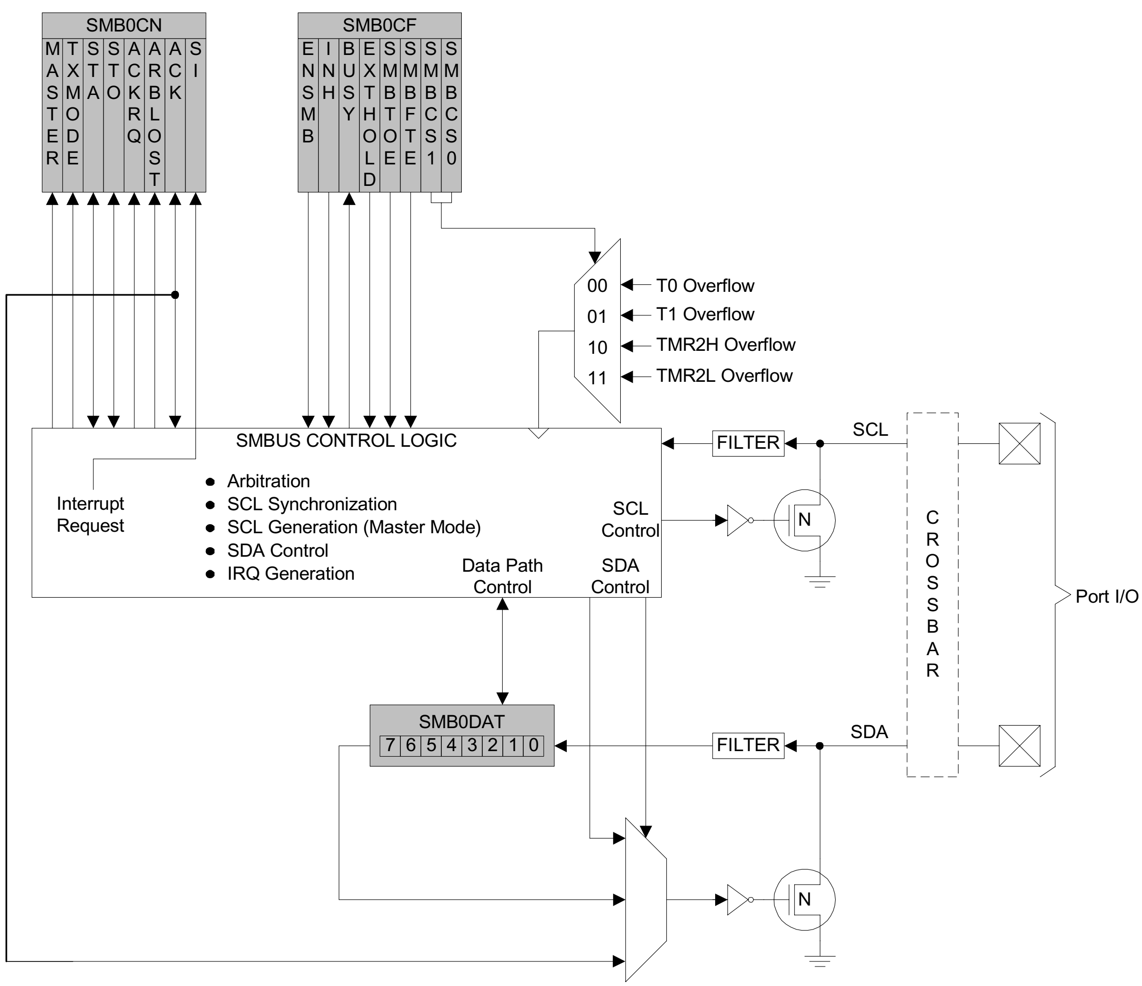
PARAMETERS	CONDITIONS	MIN	TYP	MAX	UNITS
Output High Voltage	I_{OH} = -3mA, Port I/O push-pull I_{OH} = -10µA, Port I/O push-pull I_{OH} = -10mA, Port I/O push-pull	VDD-0.7 VDD-0.1	 VDD-0.8		V
Output Low Voltage	I_{OL} = 8.5mA I_{OL} = 10µA I_{OL} = 25mA		 1.0	0.6 0.1	V
Input High Voltage		2.0			V
Input Low Voltage				0.8	V
Input Leakage Current	Weak Pull-up Off Weak Pull-up On, V_{IN} = 0 V		 25	±1 40	µA

13. SMBUS

The SMBus I/O interface is a two-wire, bi-directional serial bus. The SMBus is compliant with the System Management Bus Specification, version 1.1, and compatible with the I^2C serial bus. Reads and writes to the interface by the system controller are byte oriented with the SMBus interface autonomously controlling the serial transfer of the data. Data can be transferred at up to 1/10th of the system clock operating as master or slave (this can be faster than allowed by the SMBus specification, depending on the system clock used). A method of extending the clock-low duration is available to accommodate devices with different speed capabilities on the same bus.

The SMBus interface may operate as a master and/or slave, and may function on a bus with multiple masters. The SMBus provides control of SDA (serial data), SCL (serial clock) generation and synchronization, arbitration logic, and START/STOP control and generation. Three SFRs are associated with the SMBus: SMB0CF configures the SMBus; SMB0CN controls the status of the SMBus; and SMB0DAT is the data register, used for both transmitting and receiving SMBus data and slave addresses.

Figure 13.1. SMBus Block Diagram



13.1. Supporting Documents

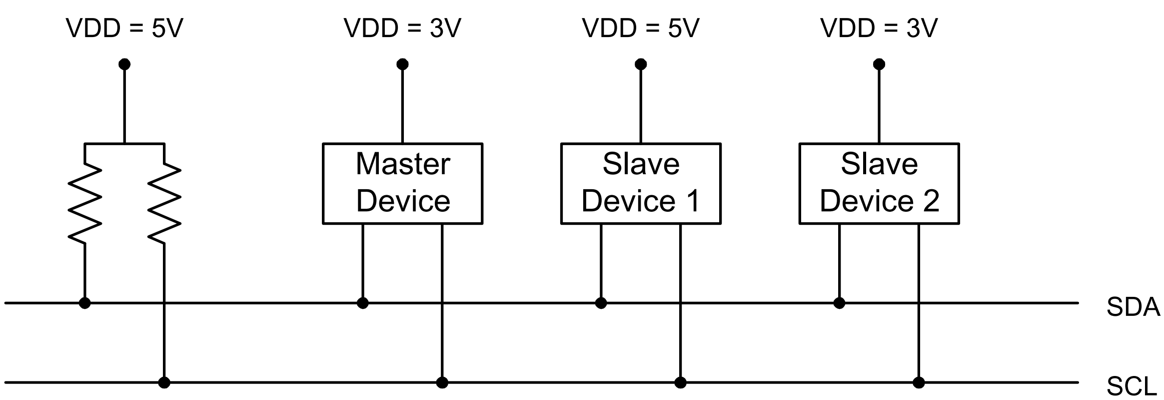
It is assumed the reader is familiar with or has access to the following supporting documents:

1. The I^2C-Bus and How to Use It (including specifications), Philips Semiconductor.
2. The I^2C-Bus Specification -- Version 2.0, Philips Semiconductor.
3. System Management Bus Specification -- Version 1.1, SBS Implementers Forum.

13.2. SMBus Configuration

Figure 13.2 shows a typical SMBus configuration. The SMBus specification allows any recessive voltage between 3.0 V and 5.0 V; different devices on the bus may operate at different voltage levels. The bi-directional SCL (serial clock) and SDA (serial data) lines must be connected to a positive power supply voltage through a pull-up resistor or similar circuit. Every device connected to the bus must have an open-drain or open-collector output for both the SCL and SDA lines, so that both are pulled high (recessive state) when the bus is free. The maximum number of devices on the bus is limited only by the requirement that the rise and fall times on the bus not exceed 300 ns and 1000 ns, respectively.

Figure 13.2. Typical SMBus Configuration



13.3. SMBus Operation

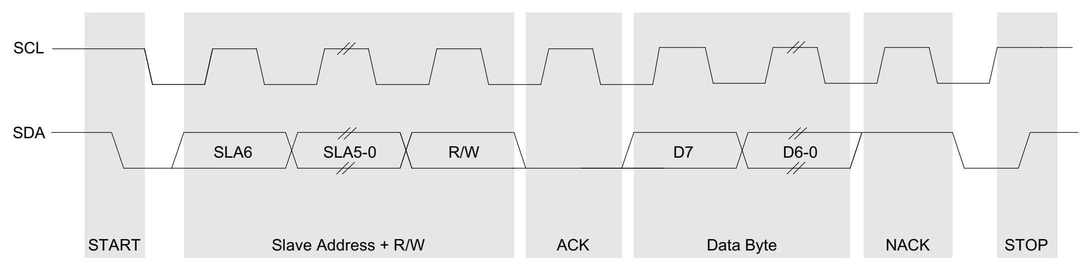
Two types of data transfers are possible: data transfers from a master transmitter to an addressed slave receiver (WRITE), and data transfers from an addressed slave transmitter to a master receiver (READ). The master device initiates both types of data transfers and provides the serial clock pulses on SCL. The SMBus interface may operate as a master or a slave, and multiple master devices on the same bus are supported. If two or more masters attempt to initiate a data transfer simultaneously, an arbitration scheme is employed with a single master always winning the arbitration. Note that it is not necessary to specify one device as the Master in a system; any device that transmits a START and a slave address becomes the master for the duration of that transfer.

A typical SMBus transaction consists of a START condition followed by an address byte (Bits7-1: 7-bit slave address; Bit0: R/W direction bit), one or more bytes of data, and a STOP condition. Each byte that is received (by a master or slave) must be acknowledged (ACK) with a low SDA during a high SCL (see Figure 13.3). If the receiving device does not ACK, the transmitting device will read a NACK (not acknowledge), which is a high SDA during a high SCL.

The direction bit (R/W) occupies the least-significant bit position of the address byte. The direction bit is set to logic 1 to indicate a "READ" operation and cleared to logic 0 to indicate a "WRITE" operation.

All transactions are initiated by a master, with one or more addressed slave devices as the target. The master generates the START condition and then transmits the slave address and direction bit. If the transaction is a WRITE operation from the master to the slave, the master transmits the data a byte at a time waiting for an ACK from the slave at the end of each byte. For READ operations, the slave transmits the data waiting for an ACK from the master at the end of each byte. At the end of the data transfer, the master generates a STOP condition to terminate the transaction and free the bus. Figure 13.3 illustrates a typical SMBus transaction.

Figure 13.3. SMBus Transaction



13.3.1. Arbitration

A master may start a transfer only if the bus is free. The bus is free after a STOP condition or after the SCL and SDA lines remain high for a specified time (see **Section “13.3.4. SCL High (SMBus Free) Timeout” on page 106**). In the event that two or more devices attempt to begin a transfer at the same time, an arbitration scheme is employed to force one master to give up the bus. The master devices continue transmitting until one attempts a HIGH while the other transmits a LOW. Since the bus is open-drain, the bus will be pulled LOW. The master attempting the HIGH will detect a LOW SDA and lose the arbitration. The winning master continues its transmission without interruption; the losing master becomes a slave and receives the rest of the transfer if addressed. This arbitration scheme is non-destructive: one device always wins, and no data is lost.

13.3.2. Clock Low Extension

SMBus provides a clock synchronization mechanism, similar to I^2C, which allows devices with different speed capabilities to coexist on the bus. A clock-low extension is used during a transfer in order to allow slower slave devices to communicate with faster masters. The slave may temporarily hold the SCL line LOW to extend the clock low period, effectively decreasing the serial clock frequency.

13.3.3. SCL Low Timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than 25 ms as a "timeout" condition. Devices that have detected the timeout condition must reset the communication no later than 10 ms after detecting the timeout condition.

When the SMBTOE bit in SMB0CF is set, Timer 2 is used to detect SCL low timeouts. Timer 2 is forced to reload when SCL is high, and allowed to count when SCL is low. With Timer 2 enabled and configured to overflow after 25 ms (and SMBTOE set), the Timer 2 interrupt service routine can be used to reset (disable and re-enable) the SMBus in the event of an SCL low timeout. Timer 2 configuration details can be found in **Section "15.2. Timer 2" on page 141**.

13.3.4. SCL High (SMBus Free) Timeout

The SMBus specification stipulates that if the SCL and SDA lines remain high for more that 50 µs, the bus is designated as free. When the SMBFTE bit in SMB0CF is set, the bus will be considered free if SCL and SDA remain high for more than 10 SMBus clock source periods. If the SMBus is waiting to generate a Master START, the START will be generated following this timeout. Note that a clock source is required for free timeout detection, even in a slave-only implementation.

13.4. Using the SMBus

The SMBus can operate in both Master and Slave modes. The interface provides timing and shifting control for serial transfers; higher level protocol is determined by user software. The SMBus interface provides the following application-independent features:

- Byte-wise serial data transfers
- Clock signal generation on SCL (Master Mode only) and SDA data synchronization
- Timeout/bus error recognition, as defined by the SMB0CF configuration register
- START/STOP timing, detection, and generation
- Bus arbitration
- Interrupt generation
- Status information

SMBus interrupts are generated for each data byte or slave address that is transferred. When transmitting, this interrupt is generated after the ACK cycle so that software may read the received ACK value; when receiving data, this interrupt is generated before the ACK cycle so that software may define the outgoing ACK value. See **Section "13.5. SMBus Transfer Modes" on page 115** for more details on transmission sequences.

Interrupts are also generated to indicate the beginning of a transfer when a master (START generated), or the end of a transfer when a slave (STOP detected). Software should read the SMB0CN (SMBus Control register) to find the cause of the SMBus interrupt. The SMB0CN register is described in **Section "13.4.2. SMB0CN Control Register" on page 111**; Table 13.4 provides a quick SMB0CN decoding reference.

SMBus configuration options include:

- Timeout detection (SCL Low Timeout and/or Bus Free Timeout)
- SDA setup and hold time extensions
- Slave event enable/disable
- Clock source selection

These options are selected in the SMB0CF register, as described in **Section "13.4.1. SMBus Configuration Register" on page 108**.

13.4.1. SMBus Configuration Register

The SMBus Configuration register (SMB0CF) is used to enable the SMBus Master and/or Slave modes, select the SMBus clock source, and select the SMBus timing and timeout options. When the ENSMB bit is set, the SMBus is enabled for all master and slave events. Slave events may be disabled by setting the INH bit. With slave events inhibited, the SMBus interface will still monitor the SCL and SDA pins; however, the interface will NACK all received addresses and will not generate any slave interrupts. When the INH bit is set, all slave events will be inhibited following the next START (interrupts will continue for the duration of the current transfer).

Table 13.1. SMBus Clock Source Selection

SMBCS1	SMBCS0	SMBus Clock Source
0	0	Timer 0 Overflow
0	1	Timer 1 Overflow
1	0	Timer 2 High Byte Overflow
1	1	Timer 2 Low Byte Overflow

The SMBCS1-0 bits select the SMBus clock source, which is used only when operating as a master or when the Free Timeout detection is enabled. When operating as a master, overflows from the selected source determine the absolute minimum SCL low and high times as defined in Equation 13.1. Note that the selected clock source may be shared by other peripherals so long as the timer is left running at all times. For example, Timer 1 overflows may generate the SMBus and UART baud rates simultaneously. Timer configuration is covered in Section “15. Timers” on page 133.

Equation 13.1. Minimum SCL High and Low Times

$$T_{HighMin} = T_{LowMin} = \frac{1}{f_{ClockSourceOverflow}}$$

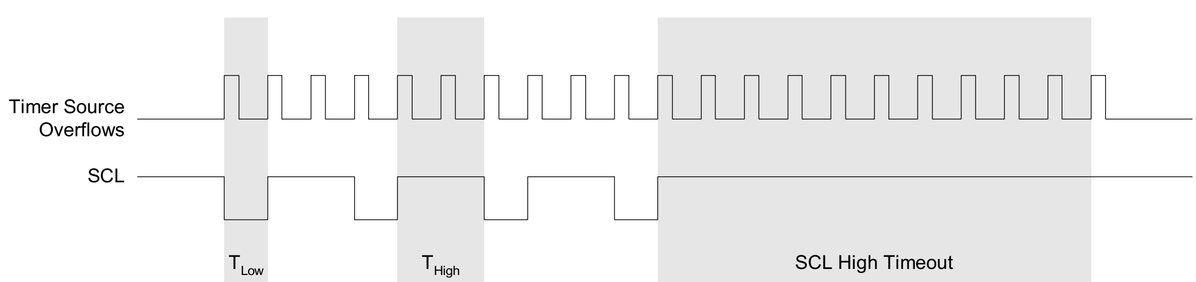
The selected clock source should be configured to establish the minimum SCL High and Low times as per Equation 13.1. When the interface is operating as a master (and SCL is not driven or extended by any other devices on the bus), the typical SMBus bit rate is approximated by Equation 13.2.

Equation 13.2. Typical SMBus Bit Rate

$$BitRate = \frac{f_{ClockSourceOverflow}}{3}$$

Figure 13.4 shows the typical SCL generation described by Equation 13.2. Notice that T_{HIGH} is typically twice as large as T_{LOW}. The actual SCL output may vary due to other devices on the bus (SCL may be extended low by slower slave devices, or driven low by contending master devices). The bit rate when operating as a master will never exceed the limits defined by equation Equation 13.1.

Figure 13.4. Typical SMBus SCL Generation



Setting the EXTHOLD bit extends the minimum setup and hold times for the SDA line. The minimum SDA setup time defines the absolute minimum time that SDA is stable before SCL transitions from low-to-high. The minimum SDA hold time defines the absolute minimum time that the current SDA value remains stable after SCL transitions from high-to-low. EXTHOLD should be set so that the minimum setup and hold times meet the SMBus Specification requirements of 250 ns and 300 ns, respectively. Table 13.2 shows the minimum setup and hold times for the two EXTHOLD settings. Setup and hold time extensions are typically necessary when SYSCLK is above 10 MHz.

Table 13.2. Minimum SDA Setup and Hold Times

EXTHOLD	Minimum SDA Setup Time	Minimum SDA Hold Time
0	T_{low} - 4 system clocks OR 1 system clock + s/w delay†	3 system clocks
1	11 system clocks	12 system clocks

†Setup Time for ACK bit transmissions and the MSB of all data transfers. The s/w delay occurs between the time SMB0DAT or ACK is written and when SI is cleared. Note that if SI is cleared in the same write that defines the outgoing ACK value, s/w delay is zero.

With the SMBTOE bit set, Timer 2 should be configured to overflow after 25 ms in order to detect SCL low timeouts (see **Section “13.3.3. SCL Low Timeout” on page 106**). The SMBus interface will force Timer 2 to reload while SCL is high, and allow Timer 2 to count when SCL is low. The Timer 2 interrupt service routine should be used to reset SMBus communication by disabling and re-enabling the SMBus. Timer 2 configuration is described in **Section “15.2. Timer 2” on page 141**.

SMBus Free Timeout detection can be enabled by setting the SMBFTE bit. When this bit is set, the bus will be considered free if SDA and SCL remain high for more than 10 SMBus clock source periods (see Figure 13.4). When a Free Timeout is detected, the interface will respond as if a STOP was detected (an interrupt will be generated, and STO will be set).

Figure 13.5. SMB0CF: SMBus Clock/Configuration Register

R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	Reset Value
ENSMB	INH	BUSY	EXTHOLD	SMBTOE	SMBFTE	SMBCS1	SMBCS0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xC1

Bit7: ENSMB: SMBus Enable.
This bit enables/disables the SMBus interface. When enabled, the interface constantly monitors the SDA and SCL pins.
0: SMBus interface disabled.
1: SMBus interface enabled.

Bit6: INH: SMBus Slave Inhibit.
When this bit is set to logic 1, the SMBus does not generate an interrupt when slave events occur. This effectively removes the SMBus slave from the bus. Master Mode interrupts are not affected.
0: SMBus Slave Mode enabled.
1: SMBus Slave Mode inhibited.

Bit5: BUSY: SMBus Busy Indicator.
This bit is set to logic 1 by hardware when a transfer is in progress. It is cleared to logic 0 when a STOP or free-timeout is sensed.

Bit4: EXTHOLD: SMBus Setup and Hold Time Extension Enable.
This bit controls the SDA setup and hold times according to Table 13.2.
0: SDA Extended Setup and Hold Times disabled.
1: SDA Extended Setup and Hold Times enabled.

Bit3: SMBTOE: SMBus SCL Timeout Detection Enable.
This bit enables SCL low timeout detection. If set to logic 1, the SMBus forces Timer 2 to reload while SCL is high and allows Timer 2 to count when SCL goes low. Timer 2 should be programmed to generate interrupts at 25 ms, and the Timer 2 interrupt service routine should reset SMBus communication.

Bit2: SMBFTE: SMBus Free Timeout Detection Enable.
When this bit is set to logic 1, the bus will be considered free if SCL and SDA remain high for more than 10 SMBus clock source periods.

Bits1-0: SMBCS1-SMBCS0: SMBus Clock Source Selection.
These two bits select the SMBus clock source, which is used to generate the SMBus bit rate. The selected device should be configured according to Equation 13.1.

SMBCS1	SMBCS0	SMBus Clock Source
0	0	Timer 0 Overflow
0	1	Timer 1 Overflow
1	0	Timer 2 High Byte Overflow
1	1	Timer 2 Low Byte Overflow

13.4.2. SMB0CN Control Register

SMB0CN is used to control the interface and to provide status information (see Figure 13.6). The higher four bits of SMB0CN (MASTER, TXMODE, STA, and STO) form a status vector that can be used to jump to service routines. MASTER and TXMODE indicate the master/slave state and transmit/receive modes, respectively.

The STA bit indicates that a START has been detected or generated since the last SMBus interrupt. When set to '1', the STA bit will cause the SMBus to enter Master mode and generate a START when the bus becomes free. STA is not cleared by hardware after the START is generated; it must be cleared by software.

As a master, writing the STO bit will cause the hardware to generate a STOP condition and end the current transfer after the next ACK cycle. STO is cleared by hardware after the STOP condition is generated. As a slave, STO indicates that a STOP condition has been detected since the last SMBus interrupt. STO is also used in slave mode to manage the transition from slave receiver to slave transmitter; see **Section 13.5.4** for details on this procedure.

If STO and STA are both set to '1' (while in Master Mode), a STOP followed by a START will be generated.

As a receiver, writing the ACK bit defines the outgoing ACK value; as a transmitter, reading the ACK bit indicates the value received on the last ACK cycle. ACKRQ is set each time a byte is received, indicating that an outgoing ACK value is needed. When ACKRQ is set, software should write the desired outgoing value to the ACK bit before clearing SI. A NACK will be generated if software does not write the ACK bit before clearing SI. SDA will reflect the defined ACK value immediately following a write to the ACK bit; however SCL will remain low until SI is cleared. If a received slave address is not acknowledged, further slave events will be ignored until the next START is detected.

The ARBLOST bit indicates that the interface has lost an arbitration. This may occur anytime the interface is transmitting (master or slave). A lost arbitration while operating as a slave indicates a bus error condition. ARBLOST is cleared by hardware each time SI is cleared.

The SI bit (SMBus Interrupt Flag) is set at the beginning and end of each transfer, after each byte frame, or when an arbitration is lost; see Table 13.3 for more details.
Important Note About the SI Bit: The SMBus interface is stalled while SI is set; thus SCL is held low, and the bus is stalled until software clears SI.

Table 13.3 lists all sources for hardware changes to the SMB0CN bits. Refer to Table 13.4 for SMBus status decoding using the SMB0CN register.

Figure 13.6. SMB0CN: SMBus Control Register

R	R	R/W	R/W	R	R	R/W	R/W	Reset Value
MASTER	TXMODE	STA	STO	ACKRQ	ARBLOST	ACK	SI	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xC0

Bit7: MASTER: SMBus Master/Slave Indicator.
This read-only bit indicates when the SMBus is operating as a master.
0: SMBus operating in Slave Mode.
1: SMBus operating in Master Mode.

Bit6: TXMODE: SMBus Transmit Mode Indicator.
This read-only bit indicates when the SMBus is operating as a transmitter.
0: SMBus in Receiver Mode.
1: SMBus in Transmitter Mode.

Bit5: STA: SMBus Start Flag.
Write:
0: No Start generated.
1: When operating as a master, a START condition is transmitted if the bus is free (If the bus is not free, the START is transmitted after a STOP is received or a free timeout is detected). If STA is set by software as an active Master, a repeated START will be generated after the next ACK cycle.
Read:
0: No Start or repeated Start detected.
1: Start or repeated Start detected.

Bit4: STO: SMBus Stop Flag.
Write:
As a master, setting this bit to '1' causes a STOP condition to be transmitted after the next ACK cycle. STO is cleared to '0' by hardware when the STOP is generated.
As a slave, software manages this bit when switching from Slave Receiver to Slave Transmitter mode. See **Section 13.5.4** for details.
Read:
0: No Stop condition detected.
1: Stop condition detected (if in Slave Mode) or pending (if in Master Mode).

Bit3: ACKRQ: SMBus Acknowledge Request.
This read-only bit is set to logic 1 when the SMBus has received a byte and needs the ACK bit to be written with the correct ACK response value.

Bit2: ARBLOST: SMBus Arbitration Lost Indicator.
This read-only bit is set to logic 1 when the SMBus loses arbitration while operating as a transmitter. A lost arbitration while a slave indicates a bus error condition.

Bit1: ACK: SMBus Acknowledge Flag.
This bit defines the out-going ACK level and records incoming ACK levels. It should be written each time a byte is received (when ACKRQ=1), or read after each byte is transmitted.
0: A "not acknowledge" has been received (if in Transmitter Mode) OR will be transmitted (if in Receiver Mode).
1: An "acknowledge" has been received (if in Transmitter Mode) OR will be transmitted (if in Receiver Mode).

Bit0: SI: SMBus Interrupt Flag.
This bit is set by hardware under the conditions listed in Table 13.3. SI must be cleared by software. While SI is set, SCL is held low and the SMBus is stalled.

Table 13.3. Sources for Hardware Changes to SMB0CN

Bit	Set by Hardware When:	Cleared by Hardware When:
MASTER	• A START is generated.	• A STOP is generated. • Arbitration is lost.
TXMODE	• START is generated. • The SMBus interface enters transmitter mode (after SMB0DAT is written before the start of an SMBus frame).	• A START is detected. • Arbitration is lost. • SMB0DAT is not written before the start of an SMBus frame.
STA	• A START followed by an address byte is received.	• Must be cleared by software.
STO	• A STOP is detected while addressed as a slave. • Arbitration is lost due to a detected STOP.	• A pending STOP is generated.
ACKRQ	• A byte has been received and an ACK response value is needed.	• After each ACK cycle.
ARBLOST	• A repeated START is detected as a MASTER when STA is low (unwanted repeated START). • SCL is sensed low while attempting to generate a STOP or repeated START condition. • SDA is sensed low while transmitting a ‘1’ (excluding ACK bits).	• Each time SI is cleared.
ACK	• The incoming ACK value is low (ACKNOWLEDGE).	• The incoming ACK value is high (NOT ACKNOWLEDGE).
SI	• A START has been generated. • Lost arbitration. • A byte has been transmitted and an ACK/NACK received. • A byte has been received. • A START or repeated START followed by a slave address + R/W has been received. • A STOP has been received.	• Must be cleared by software.

13.4.3. Data Register

The SMBus Data register SMB0DAT holds a byte of serial data to be transmitted or one that has just been received. Software may safely read or write to the data register when the SI flag is set. Software should not attempt to access the SMB0DAT register when the SMBus is enabled and the SI flag is cleared to logic 0, as the interface may be in the process of shifting a byte of data into or out of the register.

Data in SMB0DAT is always shifted out MSB first. After a byte has been received, the first bit of received data is located at the MSB of SMB0DAT. While data is being shifted out, data on the bus is simultaneously being shifted in. SMB0DAT always contains the last data byte present on the bus. In the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data or address in SMB0DAT.

Figure 13.7. SMB0DAT: SMBus Data Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0xC2

Bits7-0: SMB0DAT: SMBus Data.
The SMB0DAT register contains a byte of data to be transmitted on the SMBus serial interface or a byte that has just been received on the SMBus serial interface. The CPU can read from or write to this register whenever the SI serial interrupt flag (SMB0CN.0) is set to logic one. The serial data in the register remains stable as long as the SI flag is set. When the SI flag is not set, the system may be in the process of shifting data in/out and the CPU should not attempt to access this register.

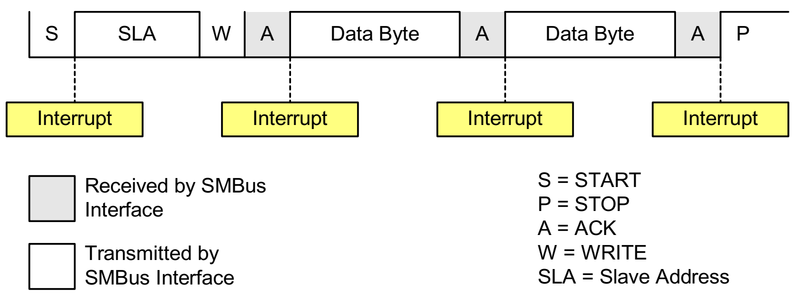
13.5. SMBus Transfer Modes

The SMBus interface may be configured to operate as master and/or slave. At any particular time, it will be operating in one of the following four modes: Master Transmitter, Master Receiver, Slave Transmitter, or Slave Receiver. The SMBus interface enters Master Mode any time a START is generated, and remains in Master Mode until it loses arbitration or generates a STOP. An SMBus interrupt is generated at the end of all SMBus byte frames; however, note that the interrupt is generated before the ACK cycle when operating as a receiver, and after the ACK cycle when operating as a transmitter.

13.5.1. Master Transmitter Mode

Serial data is transmitted on SDA while the serial clock is output on SCL. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 0 (WRITE). The master then transmits one or more bytes of serial data. After each byte is transmitted, an acknowledge bit is generated by the slave. The transfer is ended when the STO bit is set and a STOP is generated. Note that the interface will switch to Master Receiver Mode if SMB0DAT is not written following a Master Transmitter interrupt. Figure 13.8 shows a typical Master Transmitter sequence. Two transmit data bytes are shown, though any number of bytes may be transmitted. Notice that the 'data byte transferred' interrupts occur **after** the ACK cycle in this mode.

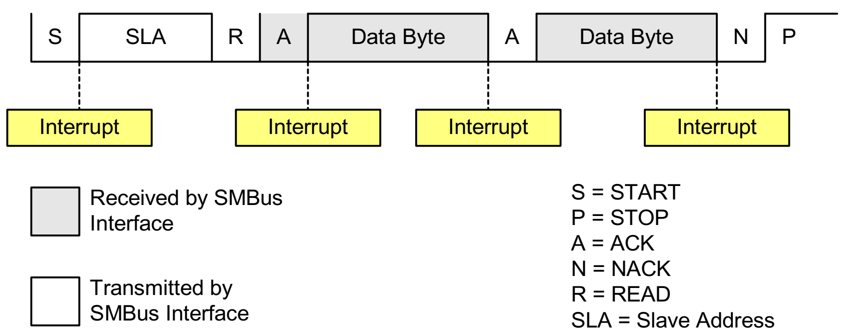
Figure 13.8. Typical Master Transmitter Sequence



13.5.2. Master Receiver Mode

Serial data is received on SDA while the serial clock is output on SCL. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 1 (READ). Serial data is then received from the slave on SDA while the SMBus outputs the serial clock. The slave transmits one or more bytes of serial data. After each byte is received, ACKRQ is set to '1' and an interrupt is generated. Software must write the ACK bit (SMB0CN.1) to define the outgoing acknowledge value (Note: writing a '1' to the ACK bit generates an ACK; writing a '0' generates a NACK). Software should write a '0' to the ACK bit after the last byte is received, to transmit a NACK. The interface exits Master Receiver Mode after the STO bit is set and a STOP is generated. Note that the interface will switch to Master Transmitter Mode if SMB0DAT is written while an active Master Receiver. Figure 13.9 shows a typical Master Receiver sequence. Two received data bytes are shown, though any number of bytes may be received. Notice that the 'data byte transferred' interrupts occur **before** the ACK cycle in this mode.

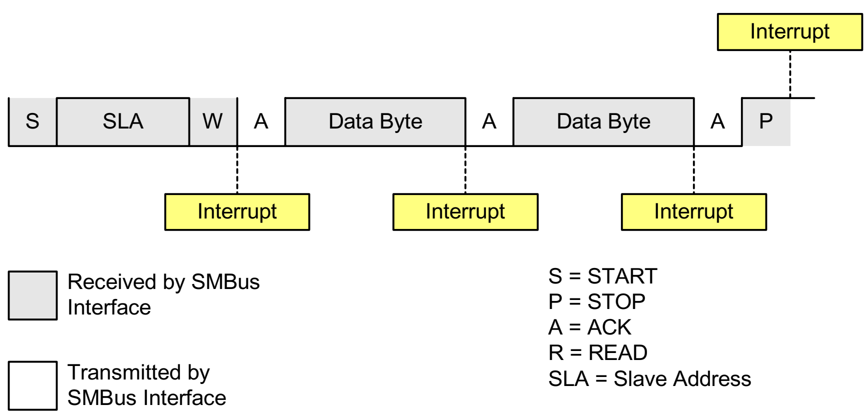
Figure 13.9. Typical Master Receiver Sequence



13.5.3. Slave Receiver Mode

Serial data is received on SDA and the clock is received on SCL. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode when a START followed by a slave address and direction bit (WRITE in this case) is received. Upon entering Slave Receiver Mode, an interrupt is generated and the ACKRQ bit is set. Software responds to the received slave address with an ACK, or ignores the received slave address with a NACK. If the received slave address is ignored, slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are received. Software must write the ACK bit after each received byte to ACK or NACK the received byte. The interface exits Slave Receiver Mode after receiving a STOP. Note that the interface will switch to Slave Transmitter Mode if SMB0DAT is written while an active Slave Receiver; see **Section 13.5.4** for details on this procedure. Figure 13.10 shows a typical Slave Receiver sequence. Two received data bytes are shown, though any number of bytes may be received. Notice that the 'data byte transferred' interrupts occur **before** the ACK cycle in this mode.

Figure 13.10. Typical Slave Receiver Sequence



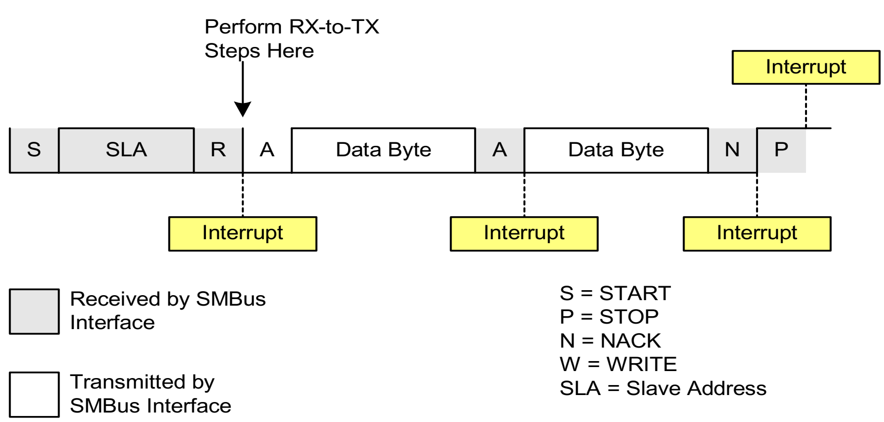
13.5.4. Slave Transmitter Mode

Serial data is transmitted on SDA and the clock is received on SCL. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode (to receive the slave address) when a START followed by a slave address and direction bit (READ in this case) is received. Software responds to the received slave address with an ACK, or ignores the received slave address with a NACK. If the received address is ignored, slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, software should write data to SMB0DAT to force the SMBus into Slave Transmitter Mode. The switch from Slave Receiver to Slave Transmitter requires software management. Software should perform the steps outlined below only when a valid slave address is received (indicated by the label “RX-to-TX Steps” in Figure 13.11).

Step 1. Set ACK to ‘1’.
Step 2. Write outgoing data to SMB0DAT.
Step 3. Check SMB0DAT.7; if ‘1’, do not perform steps 4, 6 or 7.
Step 4. Set STO to ‘1’.
Step 5. Clear SI to ‘0’.
Step 6. Poll for TXMODE => ‘1’.
Step 7. Clear STO to ‘0’ (must be done before the next ACK cycle).

The interface enters Slave Transmitter Mode and transmits one or more bytes of data (the above steps are only required before the first byte of the transfer). After each byte is transmitted, the master sends an acknowledge bit; if the acknowledge bit is an ACK, SMB0DAT should be written with the next data byte. If the acknowledge bit is a NACK, SMB0DAT should not be written to before SI is cleared (Note: an error condition may be generated if SMB0DAT is written following a received NACK while in Slave Transmitter Mode). The interface exits Slave Transmitter Mode after receiving a STOP. Note that the interface will switch to Slave Receiver Mode if SMB0DAT is not written following a Slave Transmitter interrupt. Figure 13.11 shows a typical Slave Transmitter sequence. Two transmitted data bytes are shown, though any number of bytes may be transmitted. Notice that the ‘data byte transferred’ interrupts occur **after** the ACK cycle in this mode.

Figure 13.11. Typical Slave Transmitter Sequence



13.6. SMBus Status Decoding

The current SMBus status can be easily decoded using the SMB0CN register. In the table below, STATUS VECTOR refers to the four upper bits of SMB0CN: MASTER, TXMODE, STA, and STO. Note that the shown response options are only the typical responses; application-specific procedures are allowed as long as they conform with the SMBus specification. Highlighted responses are allowed but do not conform to the SMBus specification.

Table 13.4. SMBus Status Decoding

MODE	VALUES READ				CURRENT SMBUS STATE	TYPICAL RESPONSE OPTIONS	VALUES WRITTEN		
	STATUS VECTOR	ACKRQ	ARBLOST	ACK			STA	STO	ACK
MASTER TRANSMITTER	1110	0	0	X	A master START was generated.	Load slave address + R/W into SMB0DAT.	0	0	X
	1100	0	0	0	A master data or address byte was transmitted; NACK received.	Set STA to restart transfer.	1	0	X
						Abort transfer.	0	1	X
		0	0	1	A master data or address byte was transmitted; ACK received.	Load next data byte into SMB0DAT	0	0	X
						End transfer with STOP	0	1	X
						End transfer with STOP and start another transfer.	1	1	X
						Send repeated START	1	0	X
						Switch to Master Receiver Mode (clear SI without writing new data to SMB0DAT).	0	0	X
MASTER RECEIVER	1000	1	0	X	A master data byte was received; ACK requested.	Acknowledge received byte; Read SMB0DAT.	0	0	1
						Send NACK to indicate last byte, and send STOP.	0	1	0
						Send NACK to indicate last byte, and send STOP followed by START.	1	1	0
						Send ACK followed by repeated START.	1	0	1
						Send NACK to indicate last byte, and send repeated START.	1	0	0
						Send ACK and switch to Master Transmitter Mode (write to SMB0DAT before clearing SI).	**0**	**0**	**1**
						Send NACK and switch to Master Transmitter Mode (write to SMB0DAT before clearing SI).	**0**	**0**	**0**

Table 13.4. SMBus Status Decoding

MODE	VALUES READ				CURRENT SMBUS STATE	TYPICAL RESPONSE OPTIONS	VALUES WRITTEN		
	STATUS VECTOR	ACKRQ	ARBLOST	ACK			STA	STO	ACK
SLAVE TRANSMITTER	0100	0	0	0	A slave byte was transmitted; NACK received.	No action required (expecting STOP condition).	0	0	X
		0	0	1	A slave byte was transmitted; ACK received.	Load SMB0DAT with next data byte to transmit.	0	0	X
		0	1	X	A Slave byte was transmitted; error detected.	No action required (expecting Master to end transfer).	0	0	X
	0101	0	X	X	A STOP was detected while an addressed Slave Transmitter.	No action required (transfer complete).	0	0	X

Table 13.4. SMBus Status Decoding

MODE	VALUES READ: STATUS VECTOR	VALUES READ: ACKRQ	VALUES READ: ARBLOST	VALUES READ: ACK	CURRENT SMBUS STATE	TYPICAL RESPONSE OPTIONS	VALUES WRITTEN: STA	VALUES WRITTEN: STO	VALUES WRITTEN: ACK
SLAVE RECEIVER	0010	1	0	X	A slave address was received; ACK requested.	Acknowledge received address (received slave address match, R/W bit = READ).	0	0	1
						Do not acknowledge received address.	0	0	0
						Acknowledge received address, and switch to transmitter mode (received slave address match, R/W bit = WRITE); see Section 13.5.4 for procedure.	0	0	1
		1	1	X	Lost arbitration as master; slave address received; ACK requested.	Acknowledge received address (received slave address match, R/W bit = READ).	0	0	1
						Do not acknowledge received address.	0	0	0
						Acknowledge received address, and switch to transmitter mode (received slave address match, R/W bit = WRITE); see Section 13.5.4 for procedure.	0	0	1
						Reschedule failed transfer; do not acknowledge received address	1	0	0
	0010	0	1	X	Lost arbitration while attempting a repeated START.	Abort failed transfer.	0	0	X
						Reschedule failed transfer.	1	0	X
	0001	1	1	X	Lost arbitration while attempting a STOP.	No action required (transfer complete/aborted).	0	0	0
		0	0	X	A STOP was detected while an addressed slave receiver.	No action required (transfer complete).	0	0	X
		0	1	X	Lost arbitration due to a detected STOP.	Abort transfer.	0	0	X
						Reschedule failed transfer.	1	0	X
	0000	1	0	X	A slave byte was received; ACK requested.	Acknowledge received byte; Read SMB0DAT.	0	0	1
						Do not acknowledge received byte.	0	0	0
		1	1	X	Lost arbitration while transmitting a data byte as master.	Abort failed transfer.	0	0	0
						Reschedule failed transfer.	1	0	0

Notes

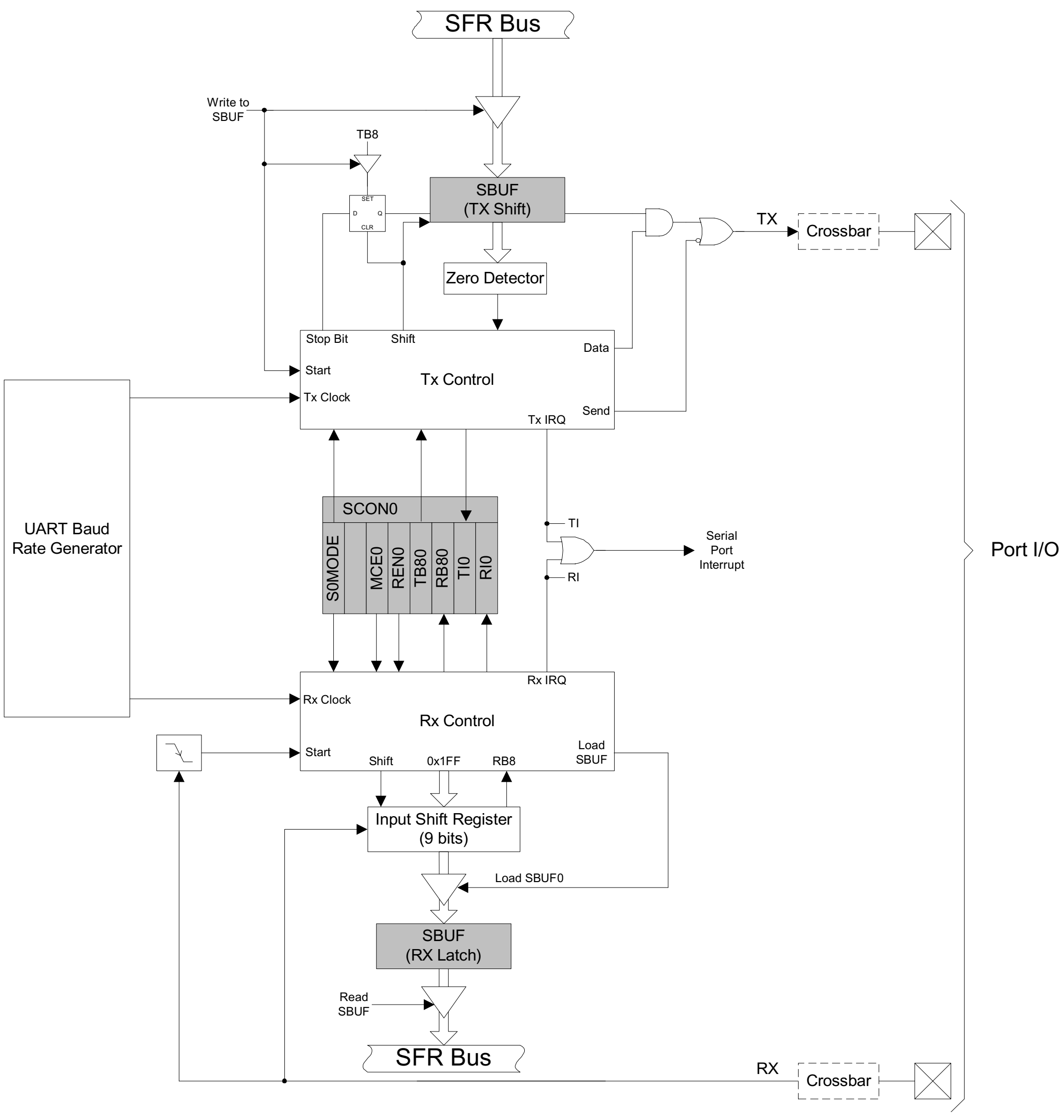
14. UART0

UART0 is an asynchronous, full duplex serial port offering modes 1 and 3 of the standard 8051 UART. Enhanced baud rate support allows a wide range of clock sources to generate standard baud rates (details in Section "14.1. Enhanced Baud Rate Generation" on page 124). Received data buffering allows UART0 to start reception of a second incoming data byte before software has finished reading the previous data byte.

UART0 has two associated SFRs: Serial Control Register 0 (SCON0) and Serial Data Buffer 0 (SBUF0). The single SBUF0 location provides access to both transmit and receive registers. Reading SBUF0 accesses the buffered Receive register; writing SBUF0 accesses the Transmit register.

With UART0 interrupts enabled, an interrupt is generated each time a transmit is completed (TI0 is set in SCON0), or a data byte has been received (RI0 is set in SCON0). The UART0 interrupt flags are not cleared by hardware when the CPU vectors to the interrupt service routine. They must be cleared manually by software, allowing software to determine the cause of the UART0 interrupt (transmit complete or receive complete).

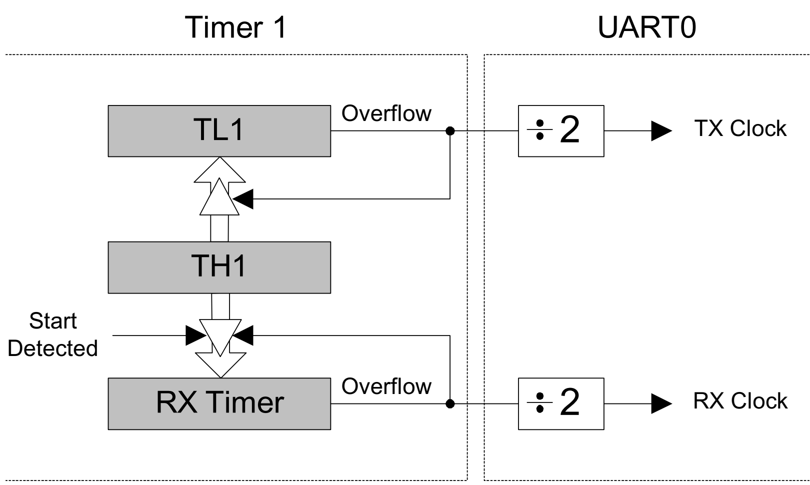
Figure 14.1. UART0 Block Diagram



14.1. Enhanced Baud Rate Generation

The UART0 baud rate is generated by Timer 1 in 8-bit auto-reload mode. The TX clock is generated by TL1; the RX clock is generated by a copy of TL1 (shown as RX Timer in Figure 14.2), which is not user-accessible. Both TX and RX Timer overflows are divided by two to generate the TX and RX baud rates. The RX Timer runs when Timer 1 is enabled, and uses the same reload value (TH1). However, an RX Timer reload is forced when a START condition is detected on the RX pin. This allows a receive to begin any time a START is detected, independent of the TX Timer state.

Figure 14.2. UART0 Baud Rate Logic



Timer 1 should be configured for Mode 2, 8-bit auto-reload (see **Section "15.1.3. Mode 2: 8-bit Counter/Timer with Auto-Reload" on page 135**). The Timer 1 reload value should be set so that overflows will occur at two times the desired UART baud rate frequency. Note that Timer 1 may be clocked by one of five sources: SYSCLK, SYSCLK / 4, SYSCLK / 12, SYSCLK / 48, or the external oscillator clock / 8. For any given Timer 1 clock source, the UART0 baud rate is determined by Equation 14.1.

Equation 14.1. UART0 Baud Rate

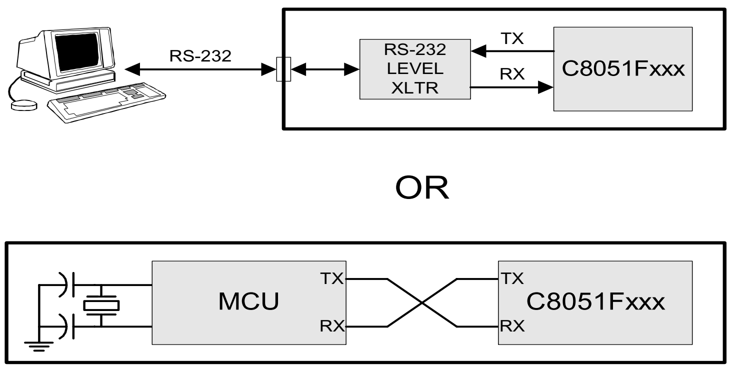
$$UartBaudRate = \frac{T1_{CLK}}{(256 - T1H)} \times \frac{1}{2}$$

Where $T1_{CLK}$ is the frequency of the clock supplied to Timer 1, and $T1H$ is the high byte of Timer 1 (reload value). Timer 1 clock frequency is selected as described in **Section "15.2. Timer 2" on page 141**. A quick reference for typical baud rates and system clock frequencies is given in Tables 14.1 through 14.6. Note that the internal oscillator may still generate the system clock when the external oscillator is driving Timer 1 (see **Section "15.1. Timer 0 and Timer 1" on page 133** for more details).

14.2. Operational Modes

UART0 provides standard asynchronous, full duplex communication. The UART mode (8-bit or 9-bit) is selected by the S0MODE bit (SCON0.7). Typical UART connection options are shown below.

Figure 14.3. UART Interconnect Diagram



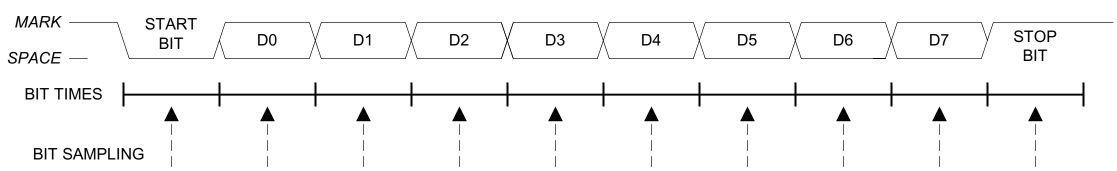
14.2.1. 8-Bit UART

8-Bit UART mode uses a total of 10 bits per data byte: one start bit, eight data bits (LSB first), and one stop bit. Data are transmitted LSB first from the TX pin and received at the RX pin. On receive, the eight data bits are stored in SBUF0 and the stop bit goes into RB80 (SCON0.2).

Data transmission begins when software writes a data byte to the SBUF0 register. The TI0 Transmit Interrupt Flag (SCON0.1) is set at the end of the transmission (the beginning of the stop-bit time). Data reception can begin any time after the REN0 Receive Enable bit (SCON0.4) is set to logic 1. After the stop bit is received, the data byte will be loaded into the SBUF0 receive register if the following conditions are met: RI0 must be logic 0, and if MCE0 is logic 1, the stop bit must be logic 1. In the event of a receive data overrun, the first received 8 bits are latched into the SBUF0 receive register and the following overrun data bits are lost.

If these conditions are met, the eight bits of data is stored in SBUF0, the stop bit is stored in RB80 and the RI0 flag is set. If these conditions are not met, SBUF0 and RB80 will not be loaded and the RI0 flag will not be set. An interrupt will occur if enabled when either TI0 or RI0 is set.

Figure 14.4. 8-Bit UART Timing Diagram

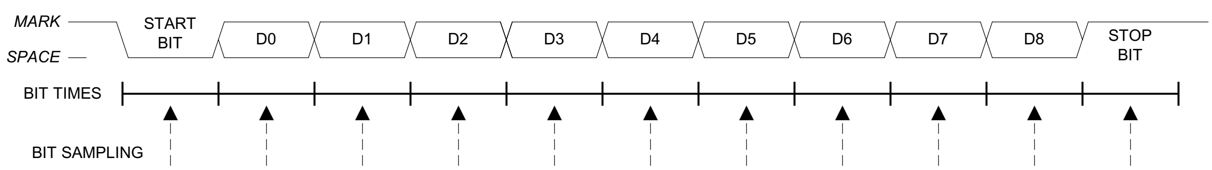


14.2.2. 9-Bit UART

9-bit UART mode uses a total of eleven bits per data byte: a start bit, 8 data bits (LSB first), a programmable ninth data bit, and a stop bit. The state of the ninth transmit data bit is determined by the value in TB80 (SCON0.3), which is assigned by user software. It can be assigned the value of the parity flag (bit P in register PSW) for error detection, or used in multiprocessor communications. On receive, the ninth data bit goes into RB80 (SCON0.2) and the stop bit is ignored.

Data transmission begins when an instruction writes a data byte to the SBUF0 register. The TI0 Transmit Interrupt Flag (SCON0.1) is set at the end of the transmission (the beginning of the stop-bit time). Data reception can begin any time after the REN0 Receive Enable bit (SCON0.4) is set to ‘1’. After the stop bit is received, the data byte will be loaded into the SBUF0 receive register if the following conditions are met: (1) RI0 must be logic 0, and (2) if MCE0 is logic 1, the 9th bit must be logic 1 (when MCE0 is logic 0, the state of the ninth data bit is unimportant). If these conditions are met, the eight bits of data are stored in SBUF0, the ninth bit is stored in RB80, and the RI0 flag is set to ‘1’. If the above conditions are not met, SBUF0 and RB80 will not be loaded and the RI0 flag will not be set to ‘1’. A UART0 interrupt will occur if enabled when either TI0 or RI0 is set to ‘1’.

Figure 14.5. 9-Bit UART Timing Diagram



14.3. Multiprocessor Communications

9-Bit UART mode supports multiprocessor communication between a master processor and one or more slave processors by special use of the ninth data bit. When a master processor wants to transmit to one or more slaves, it first sends an address byte to select the target(s). An address byte differs from a data byte in that its ninth bit is logic 1; in a data byte, the ninth bit is always set to logic 0.

Setting the MCE0 bit (SCON.5) of a slave processor configures its UART such that when a stop bit is received, the UART will generate an interrupt only if the ninth bit is logic one (RB80 = 1) signifying an address byte has been received. In the UART interrupt handler, software will compare the received address with the slave's own assigned 8-bit address. If the addresses match, the slave will clear its MCE0 bit to enable interrupts on the reception of the following data byte(s). Slaves that weren't addressed leave their MCE0 bits set and do not generate interrupts on the reception of the following data bytes, thereby ignoring the data. Once the entire message is received, the addressed slave resets its MCE0 bit to ignore all transmissions until it receives the next address byte.

Multiple addresses can be assigned to a single slave and/or a single address can be assigned to multiple slaves, thereby enabling "broadcast" transmissions to more than one slave simultaneously. The master processor can be configured to receive all transmissions or a protocol can be implemented such that the master/slave role is temporarily reversed to enable half-duplex transmission between the original master and slave(s).

Figure 14.6. UART Multi-Processor Mode Interconnect Diagram

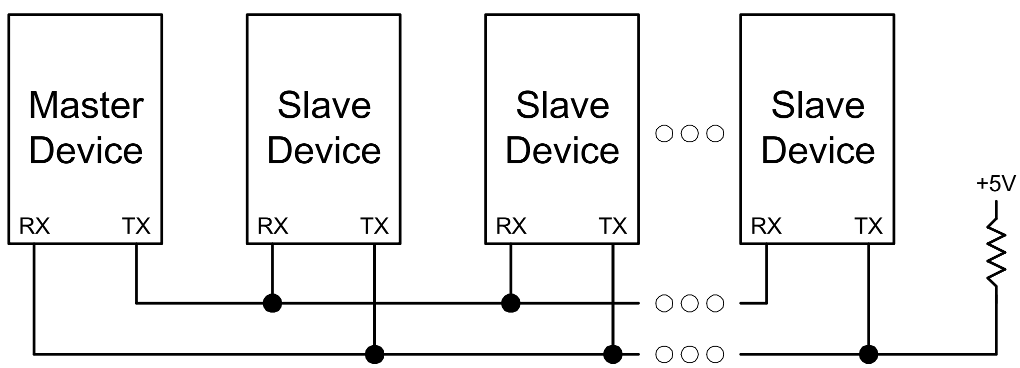


Figure 14.7. SCON0: Serial Port 0 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
S0MODE	-	MCE0	REN0	TB80	RB80	TI0	RI0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0x98

Bit7: S0MODE: Serial Port 0 Operation Mode.
This bit selects the UART0 Operation Mode.
0: Mode 0: 8-bit UART with Variable Baud Rate
1: Mode 1: 9-bit UART with Variable Baud Rate

Bit6: UNUSED. Read = 1b. Write = don't care.

Bit5: MCE0: Multiprocessor Communication Enable.
The function of this bit is dependent on the Serial Port 0 Operation Mode.
Mode 0: Checks for valid stop bit.
0: Logic level of stop bit is ignored.
1: RI0 will only be activated if stop bit is logic level 1.
Mode 1: Multiprocessor Communications Enable.
0: Logic level of ninth bit is ignored.
1: RI0 is set and an interrupt is generated only when the ninth bit is logic 1.

Bit4: REN0: Receive Enable.
This bit enables/disables the UART receiver.
0: UART0 reception disabled.
1: UART0 reception enabled.

Bit3: TB80: Ninth Transmission Bit.
The logic level of this bit will be assigned to the ninth transmission bit in 9-bit UART Mode. It is not used in 8-bit UART Mode. Set or cleared by software as required.

Bit2: RB80: Ninth Receive Bit.
RB80 is assigned the value of the STOP bit in Mode 0; it is assigned the value of the 9th data bit in Mode 1.

Bit1: TI0: Transmit Interrupt Flag.
Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in 8-bit UART Mode, or at the beginning of the STOP bit in 9-bit UART Mode). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software

Bit0: RI0: Receive Interrupt Flag.
Set to '1' by hardware when a byte of data has been received by UART0 (set at the STOP bit sampling time). When the UART0 interrupt is enabled, setting this bit to '1' causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.

Figure 14.8. SBUF0: Serial (UART0) Port Data Buffer Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x99

Bits7-0: SBUF0[7:0]: Serial Data Buffer Bits 7-0 (MSB-LSB)
This SFR accesses two registers; a transmit shift register and a receive latch register. When data is written to SBUF0, it goes to the transmit shift register and is held for serial transmission. Writing a byte to SBUF0 is what initiates the transmission. A read of SBUF0 returns the contents of the receive latch.

Table 14.1. Timer Settings for Standard Baud Rates Using The Internal Oscillator

	Frequency: 24.5 MHz						
	Target Baud Rate (bps)	**Baud Rate % Error**	**Oscillator Divide Factor**	**Timer Clock Source**	**SCA1-SCA0 (pre-scale select)†**	**T1M†**	**Timer 1 Reload Value (hex)**
SYSCLK from Internal Osc.	230400	-0.32%	106	SYSCLK	XX	1	0xCB
	115200	-0.32%	212	SYSCLK	XX	1	0x96
	57600	0.15%	426	SYSCLK	XX	1	0x2B
	28800	-0.32%	848	SYSCLK / 4	01	0	0x96
	14400	0.15%	1704	SYSCLK / 12	00	0	0xB9
	9600	-0.32%	2544	SYSCLK / 12	00	0	0x96
	2400	-0.32%	10176	SYSCLK / 48	10	0	0x96
	1200	0.15%	20448	SYSCLK / 48	10	0	0x2B

X = Don’t care

†SCA1-SCA0 and T1M bit definitions can be found in **Section 15.1**.

Table 14.2. Timer Settings for Standard Baud Rates Using an External Oscillator

	Frequency: 25.0 MHz						
	Target Baud Rate (bps)	**Baud Rate % Error**	**Oscillator Divide Factor**	**Timer Clock Source**	**SCA1-SCA0 (pre-scale select)†**	**T1M†**	**Timer 1 Reload Value (hex)**
SYSCLK from External Osc.	230400	-0.47%	108	SYSCLK	XX	1	0xCA
	115200	0.45%	218	SYSCLK	XX	1	0x93
	57600	-0.01%	434	SYSCLK	XX	1	0x27
	28800	0.45%	872	SYSCLK / 4	01	0	0x93
	14400	-0.01%	1736	SYSCLK / 4	01	0	0x27
	9600	0.15%	2608	EXTCLK / 8	11	0	0x5D
	2400	0.45%	10464	SYSCLK / 48	10	0	0x93
	1200	-0.01%	20832	SYSCLK / 48	10	0	0x27
SYSCLK from Internal Osc.	57600	-0.47%	432	EXTCLK / 8	11	0	0xE5
	28800	-0.47%	864	EXTCLK / 8	11	0	0xCA
	14400	0.45%	1744	EXTCLK / 8	11	0	0x93
	9600	0.15%	2608	EXTCLK / 8	11	0	0x5D

X = Don’t care

†SCA1-SCA0 and T1M bit definitions can be found in **Section 15.1**.

Table 14.3. Timer Settings for Standard Baud Rates Using an External Oscillator

	Frequency: 22.1184 MHz						
	Target Baud Rate (bps)	**Baud Rate % Error**	**Oscillator Divide Factor**	**Timer Clock Source**	**SCA1-SCA0 (pre-scale select)†**	**T1M†**	**Timer 1 Reload Value (hex)**
SYSCLK from External Osc.	230400	0.00%	96	SYSCLK	XX	1	0xD0
	115200	0.00%	192	SYSCLK	XX	1	0xA0
	57600	0.00%	384	SYSCLK	XX	1	0x40
	28800	0.00%	768	SYSCLK / 12	00	0	0xE0
	14400	0.00%	1536	SYSCLK / 12	00	0	0xC0
	9600	0.00%	2304	SYSCLK / 12	00	0	0xA0
	2400	0.00%	9216	SYSCLK / 48	10	0	0xA0
	1200	0.00%	18432	SYSCLK / 48	10	0	0x40
SYSCLK from Internal Osc.	230400	0.00%	96	EXTCLK / 8	11	0	0xFA
	115200	0.00%	192	EXTCLK / 8	11	0	0xF4
	57600	0.00%	384	EXTCLK / 8	11	0	0xE8
	28800	0.00%	768	EXTCLK / 8	11	0	0xD0
	14400	0.00%	1536	EXTCLK / 8	11	0	0xA0
	9600	0.00%	2304	EXTCLK / 8	11	0	0x70

X = Don't care

†SCA1-SCA0 and T1M bit definitions can be found in **Section 15.1**.

Table 14.4. Timer Settings for Standard Baud Rates Using an External Oscillator

	Frequency: 18.432 MHz						
	Target Baud Rate (bps)	**Baud Rate % Error**	**Oscillator Divide Factor**	**Timer Clock Source**	**SCA1-SCA0 (pre-scale select)†**	**T1M†**	**Timer 1 Reload Value (hex)**
SYSCLK from External Osc.	230400	0.00%	80	SYSCLK	XX	1	0xD8
	115200	0.00%	160	SYSCLK	XX	1	0xB0
	57600	0.00%	320	SYSCLK	XX	1	0x60
	28800	0.00%	640	SYSCLK / 4	01	0	0xB0
	14400	0.00%	1280	SYSCLK / 4	01	0	0x60
	9600	0.00%	1920	SYSCLK / 12	00	0	0xB0
	2400	0.00%	7680	SYSCLK / 48	10	0	0xB0
	1200	0.00%	15360	SYSCLK / 48	10	0	0x60
SYSCLK from Internal Osc.	230400	0.00%	80	EXTCLK / 8	11	0	0xFB
	115200	0.00%	160	EXTCLK / 8	11	0	0xF6
	57600	0.00%	320	EXTCLK / 8	11	0	0xEC
	28800	0.00%	640	EXTCLK / 8	11	0	0xD8
	14400	0.00%	1280	EXTCLK / 8	11	0	0xB0
	9600	0.00%	1920	EXTCLK / 8	11	0	0x88

X = Don't care

†SCA1-SCA0 and T1M bit definitions can be found in **Section 15.1**.

Table 14.5. Timer Settings for Standard Baud Rates Using an External Oscillator

	Target Baud Rate (bps)	Baud Rate % Error	Oscillator Divide Factor	Timer Clock Source	SCA1-SCA0 (pre-scale select)†	T1M†	Timer 1 Reload Value (hex)
	Frequency: 11.0592 MHz						
SYSCLK from External Osc.	230400	0.00%	48	SYSCLK	XX	1	0xE8
	115200	0.00%	96	SYSCLK	XX	1	0xD0
	57600	0.00%	192	SYSCLK	XX	1	0xA0
	28800	0.00%	384	SYSCLK	XX	1	0x40
	14400	0.00%	768	SYSCLK / 12	00	0	0xE0
	9600	0.00%	1152	SYSCLK / 12	00	0	0xD0
	2400	0.00%	4608	SYSCLK / 12	00	0	0x40
	1200	0.00%	9216	SYSCLK / 48	10	0	0xA0
SYSCLK from Internal Osc.	230400	0.00%	48	EXTCLK / 8	11	0	0xFD
	115200	0.00%	96	EXTCLK / 8	11	0	0xFA
	57600	0.00%	192	EXTCLK / 8	11	0	0xF4
	28800	0.00%	384	EXTCLK / 8	11	0	0xE8
	14400	0.00%	768	EXTCLK / 8	11	0	0xD0
	9600	0.00%	1152	EXTCLK / 8	11	0	0xB8

X = Don't care

†SCA1-SCA0 and T1M bit definitions can be found in Section 15.1.

Table 14.6. Timer Settings for Standard Baud Rates Using an External Oscillator

	Target Baud Rate (bps)	Baud Rate % Error	Oscillator Divide Factor	Timer Clock Source	SCA1-SCA0 (pre-scale select)†	T1M†	Timer 1 Reload Value (hex)
	Frequency: 3.6864 MHz						
SYSCLK from External Osc.	230400	0.00%	16	SYSCLK	XX	1	0xF8
	115200	0.00%	32	SYSCLK	XX	1	0xF0
	57600	0.00%	64	SYSCLK	XX	1	0xE0
	28800	0.00%	128	SYSCLK	XX	1	0xC0
	14400	0.00%	256	SYSCLK	XX	1	0x80
	9600	0.00%	384	SYSCLK	XX	1	0x40
	2400	0.00%	1536	SYSCLK / 12	00	0	0xC0
	1200	0.00%	3072	SYSCLK / 12	00	0	0x80
SYSCLK from Internal Osc.	230400	0.00%	16	EXTCLK / 8	11	0	0xFF
	115200	0.00%	32	EXTCLK / 8	11	0	0xFE
	57600	0.00%	64	EXTCLK / 8	11	0	0xFC
	28800	0.00%	128	EXTCLK / 8	11	0	0xF8
	14400	0.00%	256	EXTCLK / 8	11	0	0xF0
	9600	0.00%	384	EXTCLK / 8	11	0	0xE8

X = Don't care

†SCA1-SCA0 and T1M bit definitions can be found in Section 15.1.

15. TIMERS

Each MCU includes 3 counter/timers: two are 16-bit counter/timers compatible with those found in the standard 8051, and one is a 16-bit auto-reload timer for use with the ADC, SMBus, or for general purpose use. These timers can be used to measure time intervals, count external events and generate periodic interrupt requests. Timer 0 and Timer 1 are nearly identical and have four primary modes of operation. Timer 2 offers 16-bit and split 8-bit timer functionality with auto-reload.

Timer 0 and Timer 1 Modes:	Timer 2 Modes:
13-bit counter/timer	16-bit timer with auto-reload
16-bit counter/timer	
8-bit counter/timer with auto-reload	Two 8-bit timers with auto-reload
Two 8-bit counter/timers (Timer 0 only)	

Timers 0 and 1 may be clocked by one of five sources, determined by the Timer Mode Select bits (T1M-T0M) and the Clock Scale bits (SCA1-SCA0). The Clock Scale bits define a pre-scaled clock from which Timer 0 and/or Timer 1 may be clocked (See Figure 15.6 for pre-scaled clock selection).

Timer 0/1 may then be configured to use this pre-scaled clock signal or the system clock. Timer 2 may be clocked by the system clock, the system clock divided by 12, or the external oscillator clock source divided by 8.

Timer 0 and Timer 1 may also be operated as counters. When functioning as a counter, a counter/timer register is incremented on each high-to-low transition at the selected input pin. Events with a frequency of up to one-fourth the system clock's frequency can be counted. The input signal need not be periodic, but it should be held at a given level for at least two full system clock cycles to ensure the level is properly sampled.

15.1. Timer 0 and Timer 1

Each timer is implemented as 16-bit register accessed as two separate bytes: a low byte (TL0 or TL1) and a high byte (TH0 or TH1). The Counter/Timer Control register (TCON) is used to enable Timer 0 and Timer 1 as well as indicate their status. Timer 0 interrupts can be enabled by setting the ET0 bit in the IE regitser (**Section "8.3.5. Interrupt Register Descriptions" on page 70**); Timer 1 interrupts can be enabled by setting the ET1 bit in the IE register (**Section 8.3.5**). Both counter/timers operate in one of four primary modes selected by setting the Mode Select bits T1M1-T0M0 in the Counter/Timer Mode register (TMOD). Each timer can be configured independently. Each operating mode is described below.

15.1.1. Mode 0: 13-bit Counter/Timer

Timer 0 and Timer 1 operate as 13-bit counter/timers in Mode 0. The following describes the configuration and operation of Timer 0. However, both timers operate identically, and Timer 1 is configured in the same manner as described for Timer 0.

The TH0 register holds the eight MSBs of the 13-bit counter/timer. TL0 holds the five LSBs in bit positions TL0.4-TL0.0. The three upper bits of TL0 (TL0.7-TL0.5) are indeterminate and should be masked out or ignored when reading. As the 13-bit timer register increments and overflows from 0x1FFF (all ones) to 0x0000, the timer overflow flag TF0 (TCON.5) is set and an interrupt will occur if Timer 0 interrupts are enabled.

The C/T0 bit (TMOD.2) selects the counter/timer's clock source. When C/T0 is set to logic 1, high-to-low transitions at the selected Timer 0 input pin (T0) increment the timer register (Refer to **Section "12.1. Priority Crossbar Decoder" on page 96** for information on selecting and configuring external I/O pins). Clearing C/T selects the clock defined by the T0M bit (CKCON.3). When T0M is set, Timer 0 is clocked by the system clock. When T0M is cleared, Timer 0 is clocked by the source selected by the Clock Scale bits in CKCON (see Figure 15.6).

Setting the TR0 bit (TCON.4) enables the timer when either GATE0 (TMOD.3) is logic 0 or the input signal /INT0 is active as defined by bit IN0PL in register INT01CF (see Figure 8.14). Setting GATE0 to '1' allows the timer to be controlled by the external input signal /INT0 (see **Section "8.3.5. Interrupt Register Descriptions" on page 70**), facilitating pulse width measurements.

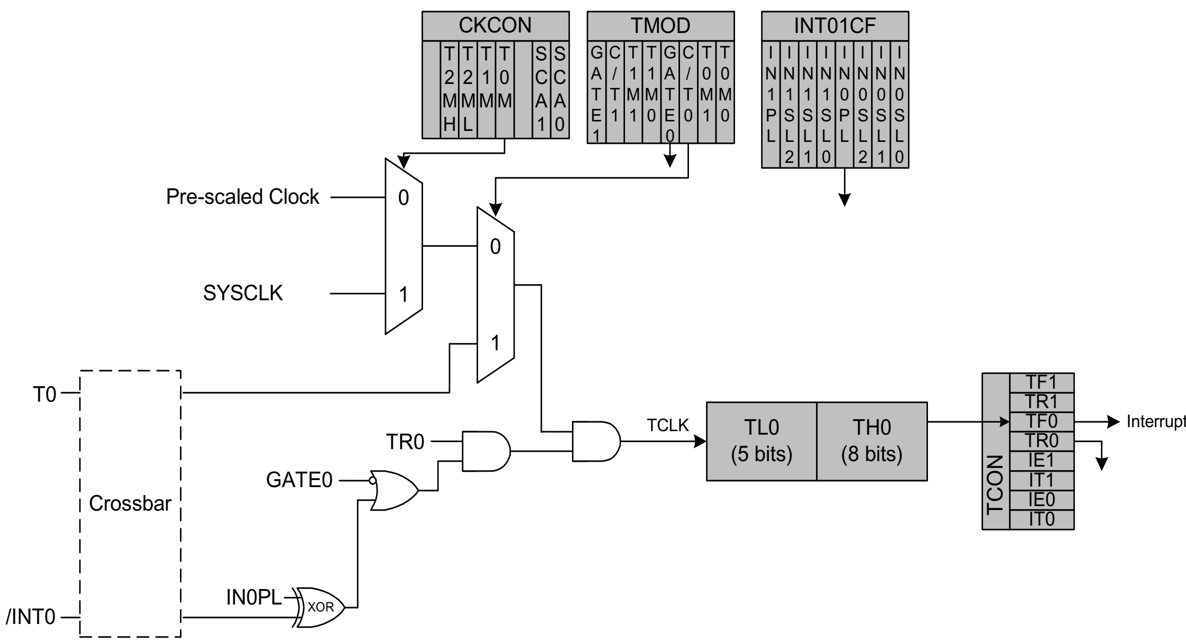
TR0	GATE0	/INT0	Counter/Timer
0	X	X	Disabled
1	0	X	Enabled
1	1	0	Disabled
1	1	1	Enabled

X = Don't Care

Setting TR0 does not force the timer to reset. The timer registers should be loaded with the desired initial value before the timer is enabled.

TL1 and TH1 form the 13-bit register for Timer 1 in the same manner as described above for TL0 and TH0. Timer 1 is configured and controlled using the relevant TCON and TMOD bits just as with Timer 0. The input signal /INT1 is used with Timer 1; the /INT1 polarity is defined by bit IN1PL in register INT01CF (see Figure 8.14).

Figure 15.1. T0 Mode 0 Block Diagram



15.1.2. Mode 1: 16-bit Counter/Timer

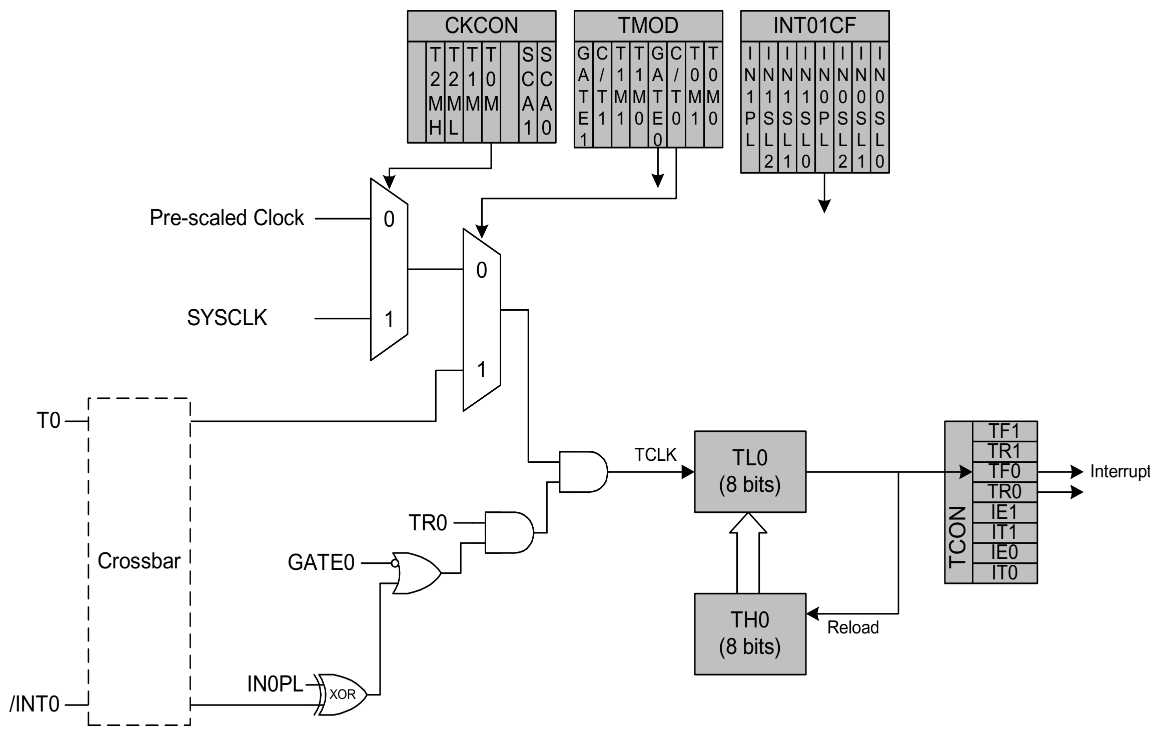
Mode 1 operation is the same as Mode 0, except that the counter/timers are enabled and configured in Mode 1 in the same manner as for Mode 0.

15.1.3. Mode 2: 8-bit Counter/Timer with Auto-Reload

Mode 2 configures Timer 0 and Timer 1 to operate as 8-bit counter/timers with automatic reload of the start value. TL0 holds the count and TH0 holds the reload value. When the counter in TL0 overflows from all ones to 0x00, the timer overflow flag TF0 (TCON.5) is set and the counter in TL0 is reloaded from TH0. If Timer 0 interrupts are enabled, an interrupt will occur when the TF0 flag is set. The reload value in TH0 is not changed. TL0 must be initialized to the desired value before enabling the timer for the first count to be correct. When in Mode 2, Timer 1 operates identically to Timer 0.

Both counter/timers are enabled and configured in Mode 2 in the same manner as Mode 0. Setting the TR0 bit (TCON.4) enables the timer when either GATE0 (TMOD.3) is logic 0 or when the input signal /INT0 is active as defined by bit IN0PL in register INT01CF (see **Section "8.3.2. External Interrupts" on page 68** for details on the external input signals /INT0 and /INT1).

Figure 15.2. T0 Mode 2 Block Diagram



15.1.4. Mode 3: Two 8-bit Counter/Timers (Timer 0 Only)

In Mode 3, Timer 0 is configured as two separate 8-bit counter/timers held in TL0 and TH0. The counter/timer in TL0 is controlled using the Timer 0 control/status bits in TCON and TMOD: TR0, C/T0, GATE0 and TF0. TL0 can use either the system clock or an external input signal as its timebase. The TH0 register is restricted to a timer function sourced by the system clock or prescaled clock. TH0 is enabled using the Timer 1 run control bit TR1. TH0 sets the Timer 1 overflow flag TF1 on overflow and thus controls the Timer 1 interrupt.

Timer 1 is inactive in Mode 3. When Timer 0 is operating in Mode 3, Timer 1 can be operated in Modes 0, 1 or 2, but cannot be clocked by external signals nor set the TF1 flag and generate an interrupt. However, the Timer 1 overflow can be used to generate baud rates for the SMBus and/or UART, and/or initiate ADC conversions. While Timer 0 is operating in Mode 3, Timer 1 run control is handled through its mode settings. To run Timer 1 while Timer 0 is in Mode 3, set the Timer 1 Mode as 0, 1, or 2. To disable Timer 1, configure it for Mode 3.

Figure 15.3. T0 Mode 3 Block Diagram

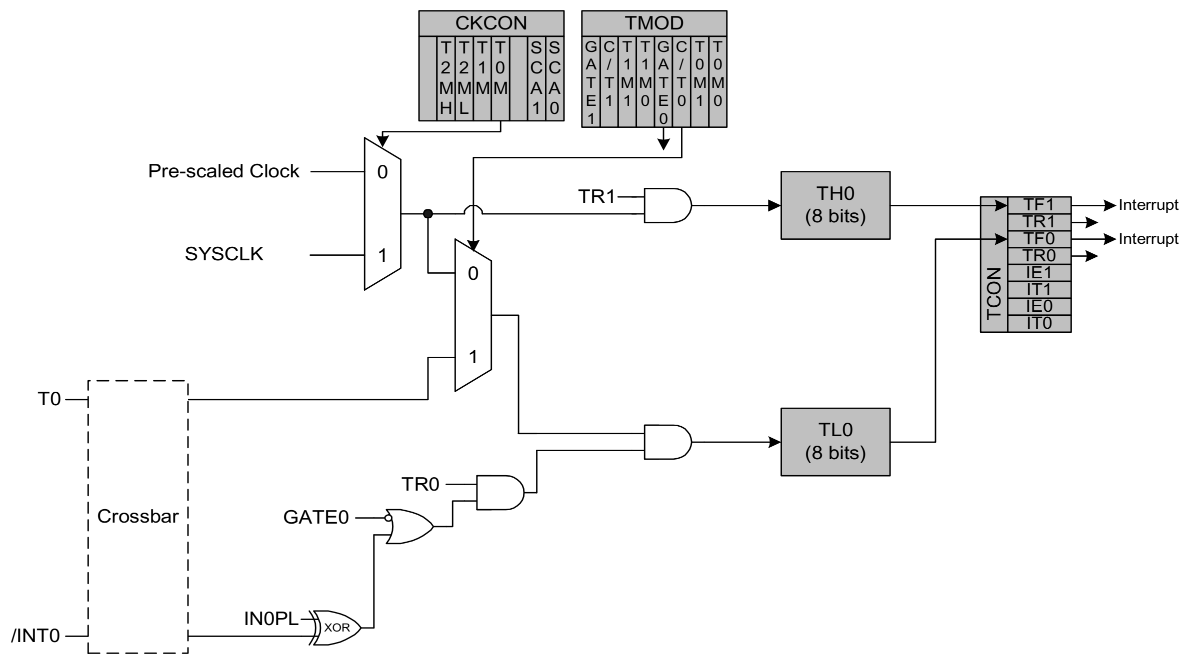


Figure 15.4. TCON: Timer Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0x88

Bit7: TF1: Timer 1 Overflow Flag.
Set by hardware when Timer 1 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine.
0: No Timer 1 overflow detected.
1: Timer 1 has overflowed.

Bit6: TR1: Timer 1 Run Control.
0: Timer 1 disabled.
1: Timer 1 enabled.

Bit5: TF0: Timer 0 Overflow Flag.
Set by hardware when Timer 0 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine.
0: No Timer 0 overflow detected.
1: Timer 0 has overflowed.

Bit4: TR0: Timer 0 Run Control.
0: Timer 0 disabled.
1: Timer 0 enabled.

Bit3: IE1: External Interrupt 1.
This flag is set by hardware when an edge/level of type defined by IT1 is detected. It can be cleared by software but is automatically cleared when the CPU vectors to the External Interrupt 1 service routine if IT1 = 1. When IT1 = 0, this flag is set to ‘1’ when /INT1 is active as defined by bit IN1PL in register INT01CF (see Figure 8.14).

Bit2: IT1: Interrupt 1 Type Select.
This bit selects whether the configured /INT1 interrupt will be edge or level sensitive. /INT1 is configured active low or high by the IN1PL bit in the IT01CF register (see Figure 8.14).
0: /INT1 is level triggered.
1: /INT1 is edge triggered.

Bit1: IE0: External Interrupt 0.
This flag is set by hardware when an edge/level of type defined by IT0 is detected. It can be cleared by software but is automatically cleared when the CPU vectors to the External Interrupt 0 service routine if IT0 = 1. When IT0 = 0, this flag is set to ‘1’ when /INT0 is active as defined by bit IN0PL in register INT01CF (see Figure 8.14).

Bit0: IT0: Interrupt 0 Type Select.
This bit selects whether the configured /INT0 interrupt will be edge or level sensitive. /INT0 is configured active low or high by the IN0PL bit in register IT01CF (see Figure 8.14).
0: /INT0 is level triggered.
1: /INT0 is edge triggered.

Figure 15.5. TMOD: Timer Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x89

Bit7: GATE1: Timer 1 Gate Control.
0: Timer 1 enabled when TR1 = 1 irrespective of /INT1 logic level.
1: Timer 1 enabled only when TR1 = 1 AND /INT1 is active as defined by bit IN1PL in register INT01CF (see Figure 8.14).

Bit6: C/T1: Counter/Timer 1 Select.
0: Timer Function: Timer 1 incremented by clock defined by T1M bit (CKCON.4).
1: Counter Function: Timer 1 incremented by high-to-low transitions on external input pin (T1).

Bits5-4: T1M1-T1M0: Timer 1 Mode Select.
These bits select the Timer 1 operation mode.

T1M1	T1M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Timer 1 inactive

Bit3: GATE0: Timer 0 Gate Control.
0: Timer 0 enabled when TR0 = 1 irrespective of /INT0 logic level.
1: Timer 0 enabled only when TR0 = 1 AND /INT0 is active as defined by bit IN0PL in register INT01CF (see Figure 8.14).

Bit2: C/T0: Counter/Timer Select.
0: Timer Function: Timer 0 incremented by clock defined by T0M bit (CKCON.3).
1: Counter Function: Timer 0 incremented by high-to-low transitions on external input pin (T0).

Bits1-0: T0M1-T0M0: Timer 0 Mode Select.
These bits select the Timer 0 operation mode.

T0M1	T0M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Two 8-bit counter/timers

Figure 15.6. CKCON: Clock Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	T2MH	T2ML	T1M	T0M	-	SCA1	SCA0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x8E

Bit7: UNUSED. Read = 0b, Write = don't care.

Bit6: T2MH: Timer 2 High Byte Clock Select
This bit selects the clock supplied to the Timer 2 high byte if Timer 2 is configured in split 8-bit timer mode. T2MH is ignored if Timer 2 is in any other mode.
0: Timer 2 high byte uses the clock defined by the T2XCLK bit in TMR2CN.
1: Timer 2 high byte uses the system clock.

Bit5: T2ML: Timer 2 Low Byte Clock Select
This bit selects the clock supplied to Timer 2. If Timer 2 is configured in split 8-bit timer mode, this bit selects the clock supplied to the lower 8-bit timer.
0: Timer 2 low byte uses the clock defined by the T2XCLK bit in TMR2CN.
1: Timer 2 low byte uses the system clock.

Bit4: T1M: Timer 1 Clock Select.
This select the clock source supplied to Timer 1. T1M is ignored when C/T1 is set to logic 1.
0: Timer 1 uses the clock defined by the prescale bits, SCA1-SCA0.
1: Timer 1 uses the system clock.

Bit3: T0M: Timer 0 Clock Select.
This bit selects the clock source supplied to Timer 0. T0M is ignored when C/T0 is set to logic 1.
0: Counter/Timer 0 uses the clock defined by the prescale bits, SCA1-SCA0.
1: Counter/Timer 0 uses the system clock.

Bit2: UNUSED. Read = 0b, Write = don't care.

Bits1-0: SCA1-SCA0: Timer 0/1 Prescale Bits
These bits control the division of the clock supplied to Timer 0 and/or Timer 1 if configured to use prescaled clock inputs.

SCA1	SCA0	Prescaled Clock
0	0	System clock divided by 12
0	1	System clock divided by 4
1	0	System clock divided by 48
1	1	External clock divided by 8

Note: External clock divided by 8 is synchronized with the system clock, and the external clock must be less than or equal to the system clock to operate in this mode.

Figure 15.7. TL0: Timer 0 Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x8A

Bits 7-0: TL0: Timer 0 Low Byte.
The TL0 register is the low byte of the 16-bit Timer 0

Figure 15.8. TL1: Timer 1 Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x8B

Bits 7-0: TL1: Timer 1 Low Byte.
The TL1 register is the low byte of the 16-bit Timer 1.

Figure 15.9. TH0: Timer 0 High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x8C

Bits 7-0: TH0: Timer 0 High Byte.
The TH0 register is the high byte of the 16-bit Timer 0.

Figure 15.10. TH1: Timer 1 High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x8D

Bits 7-0: TH1: Timer 1 High Byte.
The TH1 register is the high byte of the 16-bit Timer 1.

15.2. Timer 2

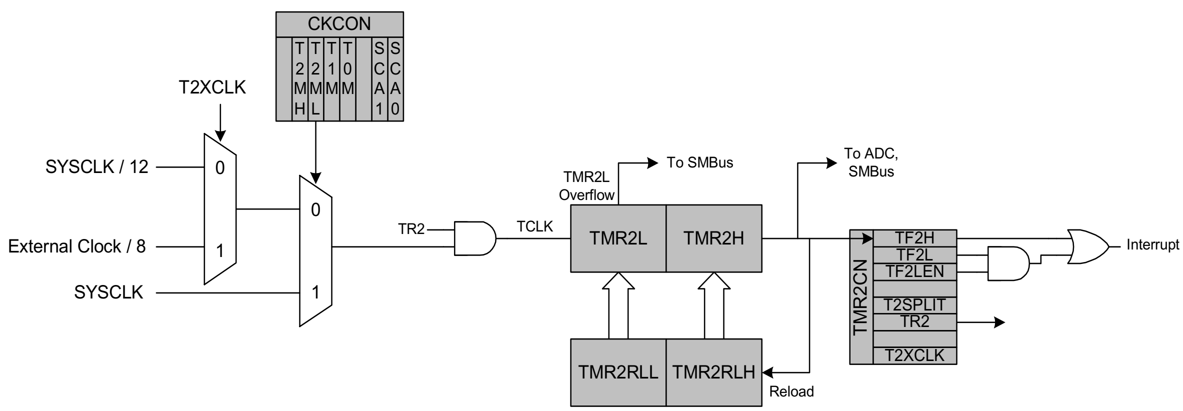
Timer 2 is a 16-bit timer formed by two 8-bit SFRs: TMR2L (low byte) and TMR2H (high byte). Timer 2 may operate in 16-bit auto-reload mode or (split) 8-bit auto-reload mode. The T2SPLIT bit (TMR2CN.3) defines the Timer 2 operation mode.

Timer 2 may be clocked by the system clock, the system clock divided by 12, or the external oscillator source divided by 8. The external clock mode is ideal for real-time clock (RTC) functionality, where the internal oscillator drives the system clock while Timer 2 (and/or the PCA) is clocked by an external precison oscillator. Note that the external oscillator source divided by 8 is synchronized with the system clock.

15.2.1. 16-bit Timer with Auto-Reload

When T2SPLIT (TMR2CN.3) is zero, Timer 2 operates as a 16-bit timer with auto-reload. Timer 2 can be clocked by SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. As the 16-bit timer register increments and overflows from 0xFFFF to 0x0000, the 16-bit value in the Timer 2 reload registers (TMR2RLH and TMR2RLL) is loaded into the Timer 2 register as shown in Figure 15.11, and the Timer 2 High Byte Overflow Flag (TMR2CN.7) is set. If Timer 2 interrupts are enabled (if IE.5 is set), an interrupt will be generated on each Timer 2 overflow. Additionally, if Timer 2 interrupts are enabled and the TF2LEN bit is set (TMR2CN.5), an interrupt will be generated each time the lower 8 bits (TL2) overflow from 0xFF to 0x00.

Figure 15.11. Timer 2 16-Bit Mode Block Diagram



15.2.2. 8-bit Timers with Auto-Reload

When T2SPLIT is set, Timer 2 operates as two 8-bit timers (TMR2H and TMR2L). Both 8-bit timers operate in auto-reload mode as shown in Figure 15.12. TMR2RLL holds the reload value for TMR2L; TMR2RLH holds the reload value for TMR2H. The TR2 bit in TMR2CN handles the run control for TMR2H. TMR2L is always running when configured for 8-bit Mode.

Each 8-bit timer may be configured to use SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. The Timer 2 Clock Select bits (T2MH and T2ML in CKCON) select either SYSCLK or the clock defined by the Timer 2 External Clock Select bit (T2XCLK in TMR2CN), as follows:

T2MH	T2XCLK	TH2 Clock Source
0	0	SYSCLK / 12
0	1	External Clock / 8
1	X	SYSCLK

T2ML	T2XCLK	TL2 Clock Source
0	0	SYSCLK / 12
0	1	External Clock / 8
1	X	SYSCLK

Note: External clock divided by 8 is synchronized with the system clock, and the external clock must be less than or equal to the system clock to operate in this mode.

The TF2H bit is set when TMR2H overflows from 0xFF to 0x00; the TF2L bit is set when TMR2L overflows from 0xFF to 0x00. When Timer 2 interrupts are enabled (IE.5), an interrupt is generated each time TMR2H overflows. If Timer 2 interrupts are enabled and TF2LEN (TMR2CN.5) is set, an interrupt is generated each time either TMR2L or TMR2H overflows. When TF2LEN is enabled, software must check the TF2H and TF2L flags to determine the source of the Timer 2 interrupt. The TF2H and TF2L interrupt flags are not cleared by hardware and must be manually cleared by software.

Figure 15.12. Timer 2 8-Bit Mode Block Diagram

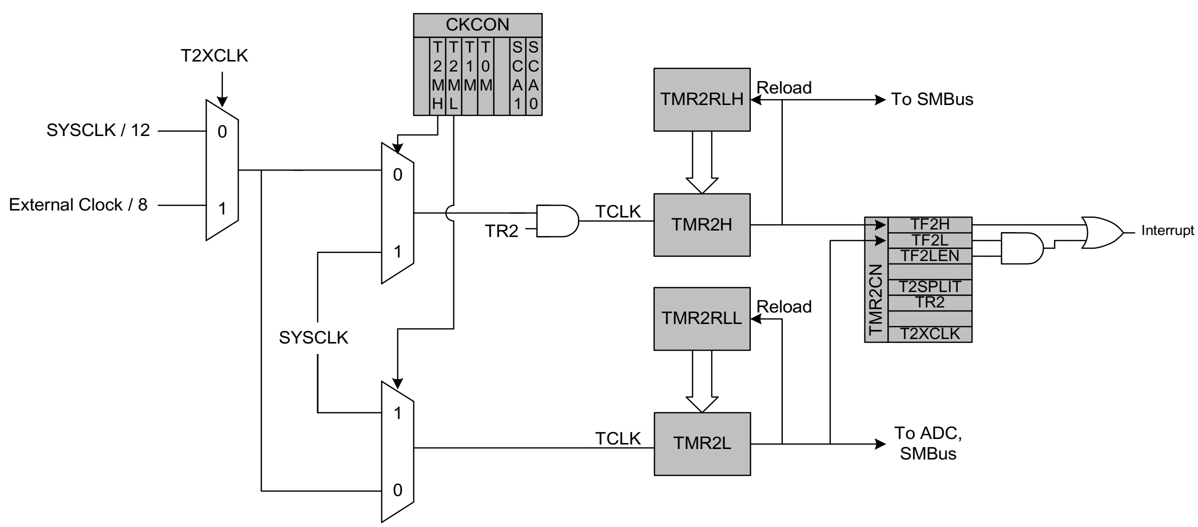


Figure 15.13. TMR2CN: Timer 2 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF2H	TF2L	TF2LEN	-	T2SPLIT	TR2	-	T2XCLK	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xC8

Bit7: TF2H: Timer 2 High Byte Overflow Flag
Set by hardware when the Timer 2 high byte overflows from 0xFF to 0x00. In 16 bit mode, this will occur when Timer 2 overflows from 0xFFFF to 0x0000. When the Timer 2 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 2 interrupt service routine. TF2H is not automatically cleared by hardware and must be cleared by software.

Bit6: TF2L: Timer 2 Low Byte Overflow Flag
Set by hardware when the Timer 2 low byte overflows from 0xFF to 0x00. When this bit is set, an interrupt will be generated if TF2LEN is set and Timer 2 interrupts are enabled. TF2L will set when the low byte overflows regardless of the Timer 2 mode. This bit is not automatically cleared by hardware.

Bit5: TF2LEN: Timer 2 Low Byte Interrupt Enable.
This bit enables/disables Timer 2 Low Byte interrupts. If TF2LEN is set and Timer 2 interrupts are enabled, an interrupt will be generated when the low byte of Timer 2 overflows. This bit should be cleared when operating Timer 2 in 16-bit mode.
0: Timer 2 Low Byte interrupts disabled.
1: Timer 2 Low Byte interrupts enabled.

Bit4: UNUSED. Read = 0b. Write = don't care.

Bit3: T2SPLIT: Timer 2 Split Mode Enable
When this bit is set, Timer 2 operates as two 8-bit timers with auto-reload.
0: Timer 2 operates in 16-bit auto-reload mode.
1: Timer 2 operates as two 8-bit auto-reload timers.

Bit2: TR2: Timer 2 Run Control.
This bit enables/disables Timer 2. In 8-bit mode, this bit enables/disables TH2 only; TL2 is always enabled in this mode.
0: Timer 2 disabled.
1: Timer 2 enabled.

Bit1: UNUSED. Read = 0b. Write = don't care.

Bit0: T2XCLK: Timer 2 External Clock Select
This bit selects the external clock source for Timer 2. If Timer 2 is in 8-bit mode, this bit selects the external oscillator clock source for both timer bytes. However, the Timer 2 Clock Select bits (T2MH and T2ML in register CKCON) may still be used to select between the external clock and the system clock for either timer.
0: Timer 2 external clock selection is the system clock divided by 12.
1: Timer 2 external clock selection is the external clock divided by 8. Note that the external oscillator source divided by 8 is synchronized with the system clock.

Figure 15.14. TMR2RLL: Timer 2 Reload Register Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0xCA

Bits 7-0: TMR2RLL: Timer 2 Reload Register Low Byte.
TMR2RLL holds the low byte of the reload value for Timer 2.

Figure 15.15. TMR2RLH: Timer 2 Reload Register High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0xCB

Bits 7-0: TMR2RLH: Timer 2 Reload Register High Byte.
The TMR2RLH holds the high byte of the reload value for Timer 2.

Figure 15.16. TMR2L: Timer 2 Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0xCC

Bits 7-0: TMR2L: Timer 2 Low Byte.
In 16-bit mode, the TMR2L register contains the low byte of the 16-bit Timer 2. In 8-bit mode, TMR2L contains the 8-bit low byte timer value.

Figure 15.17. TMR2H Timer 2 High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0xCD

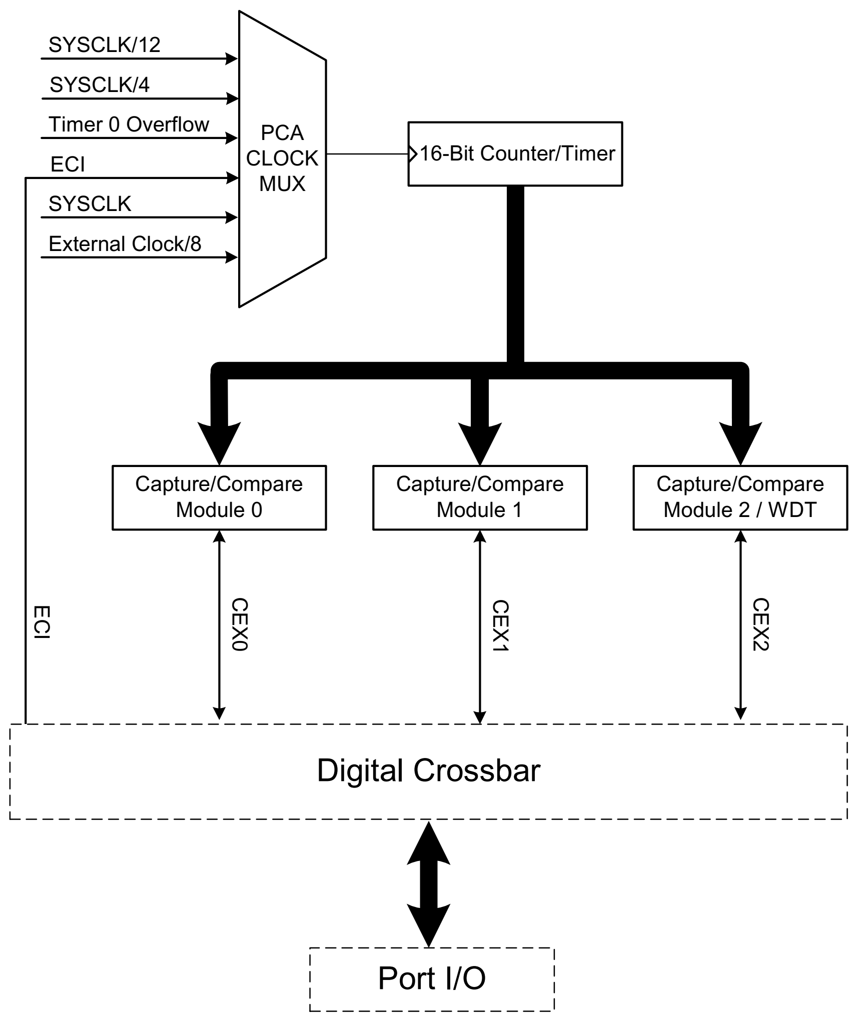
Bits 7-0: TMR2H: Timer 2 High Byte.
In 16-bit mode, the TMR2H register contains the high byte of the 16-bit Timer 2. In 8-bit mode, TMR2H contains the 8-bit high byte timer value.

16. PROGRAMMABLE COUNTER ARRAY

The Programmable Counter Array (PCA0) provides enhanced timer functionality while requiring less CPU intervention than the standard 8051 counter/timers. The PCA consists of a dedicated 16-bit counter/timer and three 16-bit capture/compare modules. Each capture/compare module has its own associated I/O line (CEXn) which is routed through the Crossbar to Port I/O when enabled (See Section "12.1. Priority Crossbar Decoder" on page 96 for details on configuring the Crossbar). The counter/timer is driven by a programmable timebase that can select between six sources: system clock, system clock divided by four, system clock divided by twelve, the external oscillator clock source divided by 8, Timer 0 overflow, or an external clock signal on the ECI input pin. Each capture/compare module may be configured to operate independently in one of six modes: Edge-Triggered Capture, Software Timer, High-Speed Output, Frequency Output, 8-Bit PWM, or 16-Bit PWM (each mode is described in Section "16.2. Capture/Compare Modules" on page 147). The external oscillator clock option is ideal for real-time clock (RTC) functionality, allowing the PCA to be clocked by a precision external oscillator while the internal oscillator drives the system clock. The PCA is configured and controlled through the system controller's Special Function Registers. The basic PCA block diagram is shown in Figure 16.1.

Important Note: The PCA Module 2 may be used as a watchdog timer (WDT), and is enabled in this mode following a system reset. Access to certain PCA registers is restricted while WDT mode is enabled. See Section 16.3 for details.

Figure 16.1. PCA Block Diagram



16.1. PCA Counter/Timer

The 16-bit PCA counter/timer consists of two 8-bit SFRs: PCA0L and PCA0H. PCA0H is the high byte (MSB) of the 16-bit counter/timer and PCA0L is the low byte (LSB). Reading PCA0L automatically latches the value of PCA0H into a "snapshot" register; the following PCA0H read accesses this "snapshot" register. **Reading the PCA0L Register first guarantees an accurate reading of the entire 16-bit PCA0 counter**. Reading PCA0H or PCA0L does not disturb the counter operation. The CPS2-CPS0 bits in the PCA0MD register select the timebase for the counter/timer as shown in Table 16.1. **Note that in 'External oscillator source divided by 8' mode, the external oscillator source is synchronized with the system clock, and must have a frequency less than or equal to the system clock.**

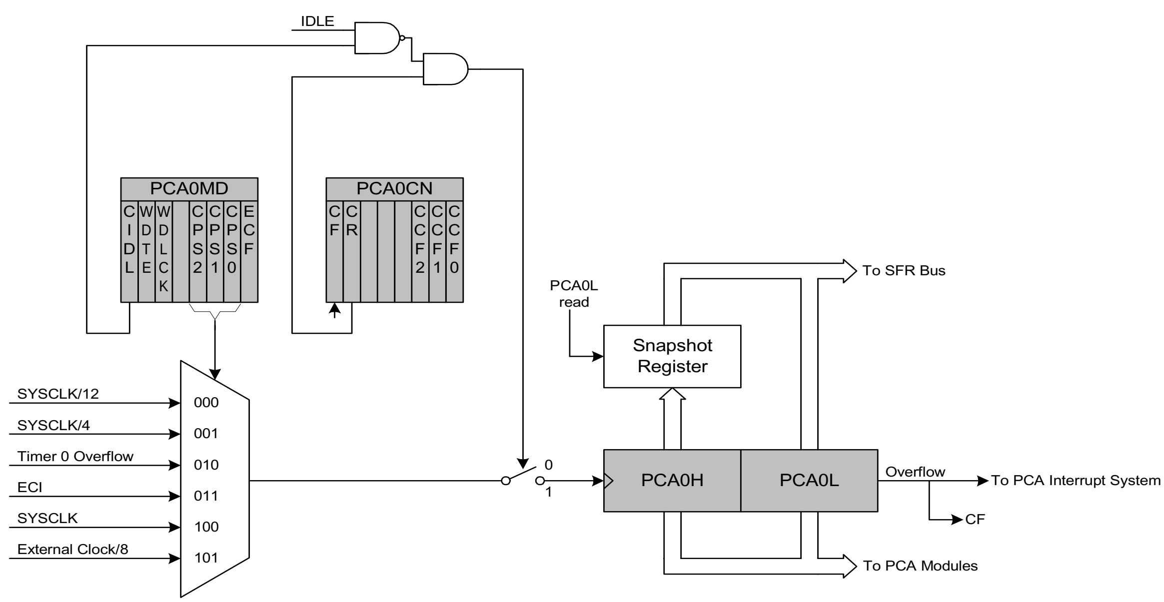
When the counter/timer overflows from 0xFFFF to 0x0000, the Counter Overflow Flag (CF) in PCA0MD is set to logic 1 and an interrupt request is generated if CF interrupts are enabled. Setting the ECF bit in PCA0MD to logic 1 enables the CF flag to generate an interrupt request. The CF bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software (Note: PCA0 interrupts must be globally enabled before CF interrupts are recognized. PCA0 interrupts are globally enabled by setting the EA bit and the EPCA0 bit to logic 1). Clearing the CIDL bit in the PCA0MD register allows the PCA to continue normal operation while the CPU is in Idle mode.

Table 16.1. PCA Timebase Input Options

CPS2	CPS1	CPS0	Timebase
0	0	0	System clock divided by 12
0	0	1	System clock divided by 4
0	1	0	Timer 0 overflow
0	1	1	High-to-low transitions on ECI (max rate = system clock divided by 4)
1	0	0	System clock
1	0	1	External oscillator source divided by 8†

†External oscillator source divided by 8 is synchronized with the system clock.

Figure 16.2. PCA Counter/Timer Block Diagram



 DS004-2.2 APR 03

16.2. Capture/Compare Modules

Each module can be configured to operate independently in one of six operation modes: Edge-triggered Capture, Software Timer, High Speed Output, Frequency Output, 8-Bit Pulse Width Modulator, or 16-Bit Pulse Width Modulator. Each module has Special Function Registers (SFRs) associated with it in the CIP-51 system controller. These registers are used to exchange data with a module and configure the module's mode of operation.

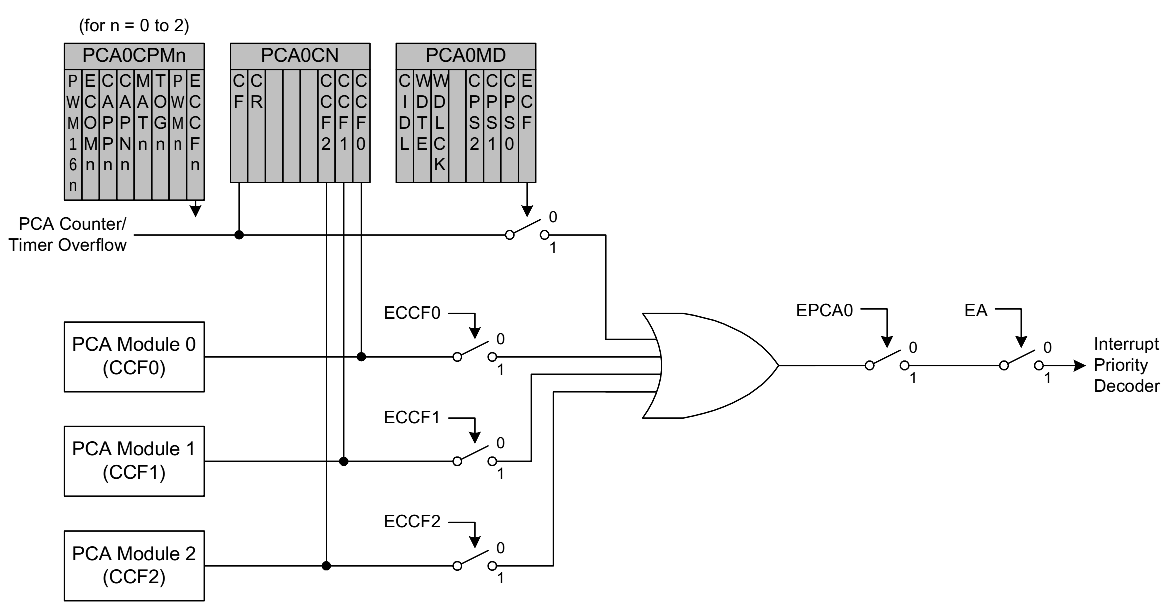
Table 16.2 summarizes the bit settings in the PCA0CPMn registers used to select the PCA capture/compare module's operating modes. Setting the ECCFn bit in a PCA0CPMn register enables the module's CCFn interrupt. Note: PCA0 interrupts must be globally enabled before individual CCFn interrupts are recognized. PCA0 interrupts are globally enabled by setting the EA bit and the EPCA0 bit to logic 1. See Figure 16.3 for details on the PCA interrupt configuration.

Table 16.2. PCA0CPM Register Settings for PCA Capture/Compare Modules

PWM16	ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	Operation Mode
X	X	1	0	0	0	0	X	Capture triggered by positive edge on CEXn
X	X	0	1	0	0	0	X	Capture triggered by negative edge on CEXn
X	X	1	1	0	0	0	X	Capture triggered by transition on CEXn
X	1	0	0	1	0	0	X	Software Timer
X	1	0	0	1	1	0	X	High Speed Output
X	1	0	0	X	1	1	X	Frequency Output
0	1	0	0	X	0	1	X	8-Bit Pulse Width Modulator
1	1	0	0	X	0	1	X	16-Bit Pulse Width Modulator

X = Don't Care

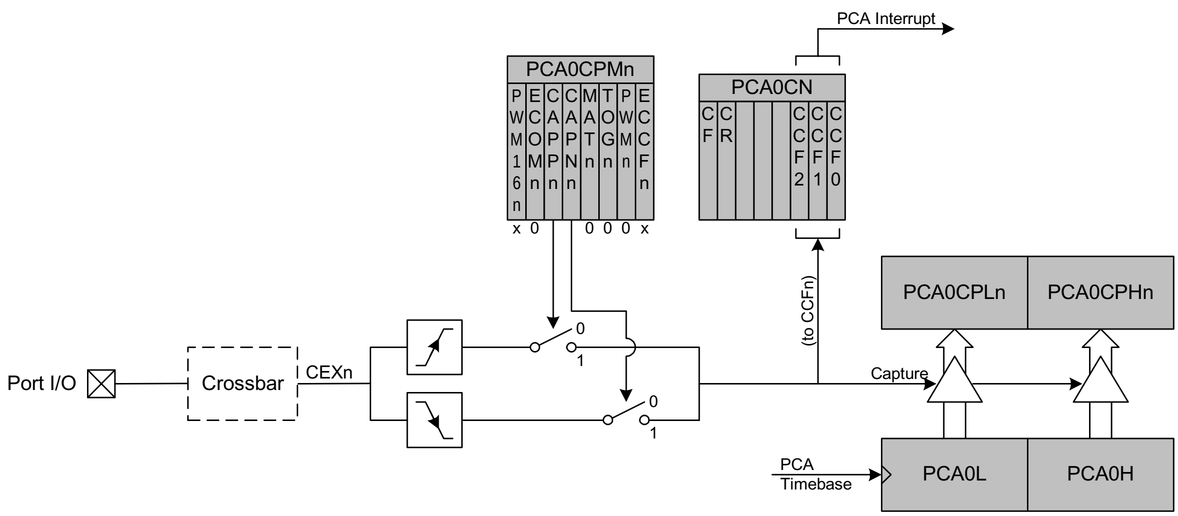
Figure 16.3. PCA Interrupt Block Diagram



16.2.1. Edge-triggered Capture Mode

In this mode, a valid transition on the CEXn pin causes the PCA to capture the value of the PCA counter/timer and copy it into the corresponding module's 16-bit capture/compare register (PCA0CPLn and PCA0CPHn). The CAPPn and CAPNn bits in the PCA0CPMn register are used to select the type of transition that triggers the capture: low-to-high transition (positive edge), high-to-low transition (negative edge), or either transition (positive or negative edge). When a capture occurs, the Capture/Compare Flag (CCFn) in PCA0CN is set to logic 1 and an interrupt request is generated if CCF interrupts are enabled. The CCFn bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. If both CAPPn and CAPNn bits are set to logic 1, then the state of the Port pin associated with CEXn can be read directly to determine whether a rising-edge or falling-edge caused the capture.

Figure 16.4. PCA Capture Mode Diagram



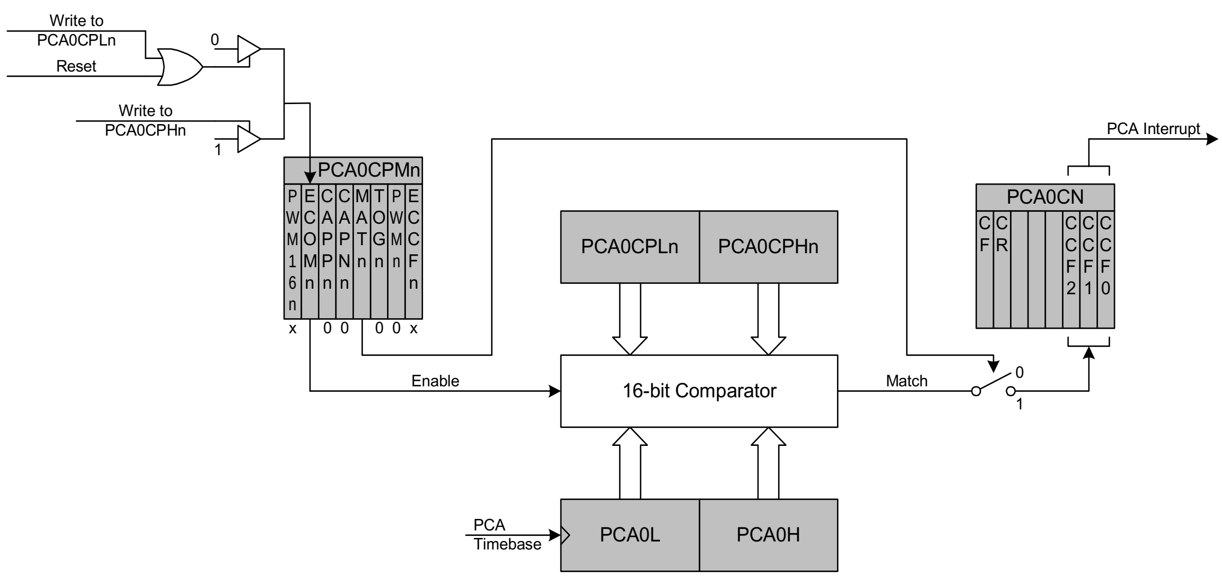
Note: The CEXn input signal must remain high or low for at least 2 system clock cycles to be recognized by the hardware.

16.2.2. Software Timer (Compare) Mode

In Software Timer mode, the PCA counter/timer value is compared to the module's 16-bit capture/compare register (PCA0CPHn and PCA0CPLn). When a match occurs, the Capture/Compare Flag (CCFn) in PCA0CN is set to logic 1 and an interrupt request is generated if CCF interrupts are enabled. The CCFn bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Setting the ECOMn and MATn bits in the PCA0CPMn register enables Software Timer mode.

Important Note About Capture/Compare Registers: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to ‘0’; writing to PCA0CPHn sets ECOMn to ‘1’.

Figure 16.5. PCA Software Timer Mode Diagram

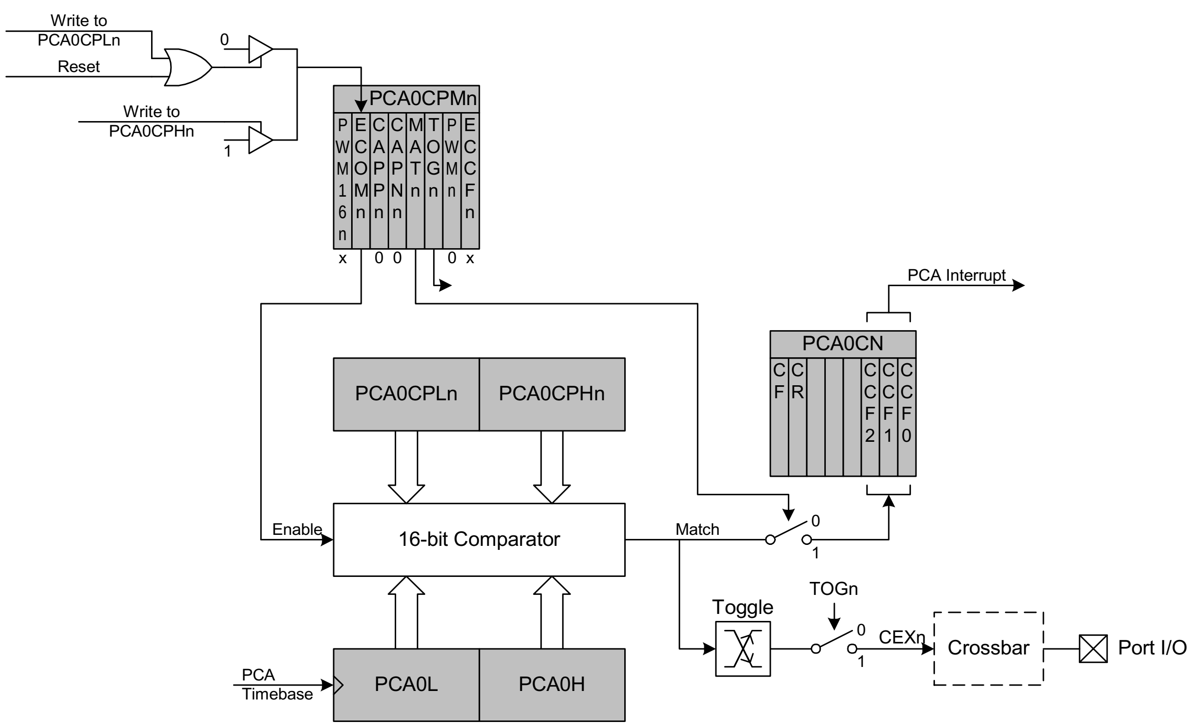


16.2.3. High Speed Output Mode

In High Speed Output mode, a module's associated CEXn pin is toggled each time a match occurs between the PCA Counter and the module's 16-bit capture/compare register (PCA0CPHn and PCA0CPLn) Setting the TOGn, MATn, and ECOMn bits in the PCA0CPMn register enables the High-Speed Output mode.

Important Note About Capture/Compare Registers: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to '0'; writing to PCA0CPHn sets ECOMn to '1'.

Figure 16.6. PCA High Speed Output Mode Diagram



16.2.4. Frequency Output Mode

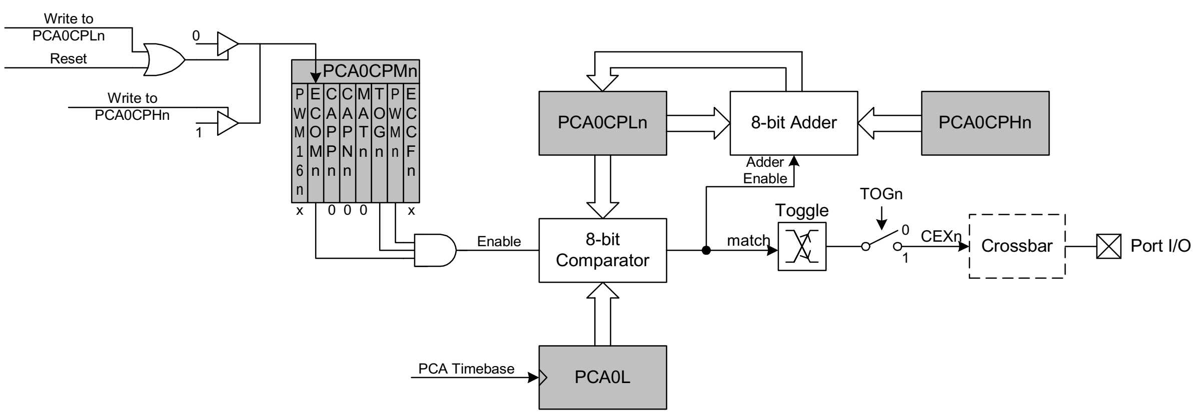
Frequency Output Mode produces a programmable-frequency square wave on the module's associated CEXn pin. The capture/compare module high byte holds the number of PCA clocks to count before the output is toggled. The frequency of the square wave is then defined by Equation 16.1.

Equation 16.1. Square Wave Frequency Output

$$F_{CEXn} = \frac{F_{PCA}}{2 \times PCA0CPHn}$$

Where F_{PCA} is the frequency of the clock selected by the CPS2-0 bits in the PCA mode register, PCA0MD. The lower byte of the capture/compare module is compared to the PCA counter low byte; on a match, CEXn is toggled and the offset held in the high byte is added to the matched value in PCA0CPLn. Frequency Output Mode is enabled by setting the ECOMn, TOGn, and PWMn bits in the PCA0CPMn register.

Figure 16.7. PCA Frequency Output Mode



16.2.5. 8-Bit Pulse Width Modulator Mode

Each module can be used independently to generate a pulse width modulated (PWM) output on its associated CEXn pin. The frequency of the output is dependent on the timebase for the PCA counter/timer. The duty cycle of the PWM output signal is varied using the module's PCA0CPLn capture/compare register. When the value in the low byte of the PCA counter/timer (PCA0L) is equal to the value in PCA0CPLn, the output on the CEXn pin will be set to ‘1’. When the count value in PCA0L overflows, the CEXn output will be set to ‘0’ (see Figure 16.8). Also, when the counter/timer low byte (PCA0L) overflows from 0xFF to 0x00, PCA0CPLn is reloaded automatically with the value stored in the module’s capture/compare high byte (PCA0CPHn) without software intervention. Setting the ECOMn and PWMn bits in the PCA0CPMn register enables 8-Bit Pulse Width Modulator mode. The duty cycle for 8-Bit PWM Mode is given by Equation 16.2.

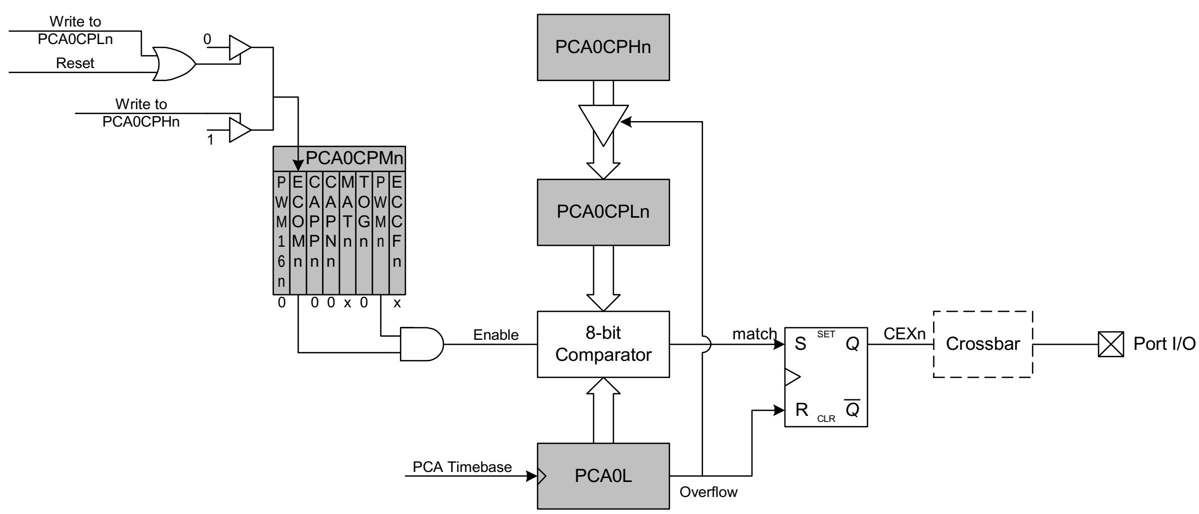
Important Note About Capture/Compare Registers: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to ‘0’; writing to PCA0CPHn sets ECOMn to ‘1’.

Equation 16.2. 8-Bit PWM Duty Cycle

$$DutyCycle = \frac{(256 - PCA0CPHn)}{256}$$

Using Equation 16.2, the largest duty cycle is 100% (PCA0CPHn = 0), and the smallest duty cycle is 0.39% (PCA0CPHn = 0xFF). A 0% duty cycle may be generated by clearing the ECOMn bit to ‘0’.

Figure 16.8. PCA 8-Bit PWM Mode Diagram



16.2.6. 16-Bit Pulse Width Modulator Mode

A PCA module may also be operated in 16-Bit PWM mode. In this mode, the 16-bit capture/compare module defines the number of PCA clocks for the low time of the PWM signal. When the PCA counter matches the module contents, the output on CEXn is set to ‘1’; when the counter overflows, CEXn is set to ‘0’. To output a varying duty cycle, new value writes should be synchronized with PCA CCFn match interrupts. 16-Bit PWM Mode is enabled by setting the ECOMn, PWMn, and PWM16n bits in the PCA0CPMn register. For a varying duty cycle, match interrupts should be enabled (ECCFn = 1 AND MATn = 1) to help synchronize the capture/compare register writes. The duty cycle for 16-Bit PWM Mode is given by Equation 16.3.

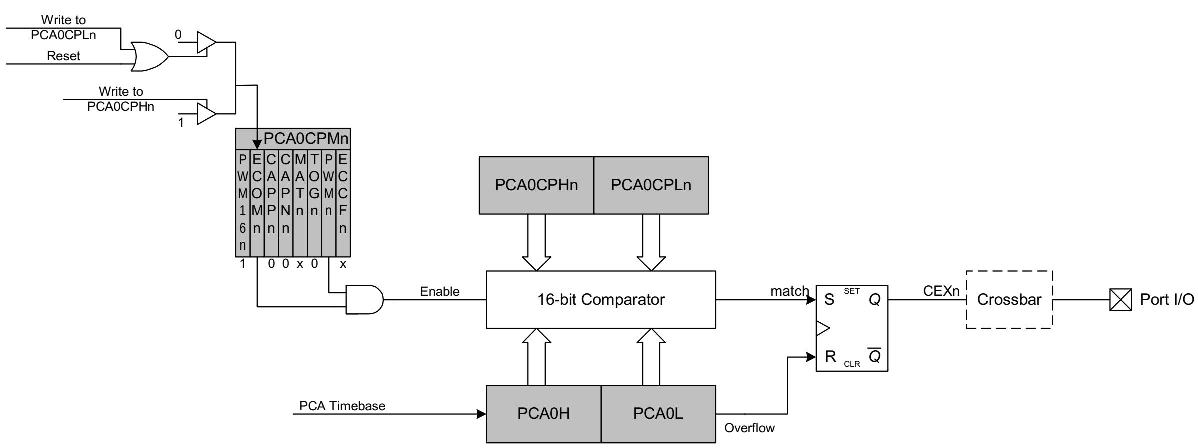
Important Note About Capture/Compare Registers: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to ‘0’; writing to PCA0CPHn sets ECOMn to ‘1’.

Equation 16.3. 16-Bit PWM Duty Cycle

$$DutyCycle = \frac{(65536 - PCA0CPn)}{65536}$$

Using Equation 16.3, the largest duty cycle is 100% (PCA0CPn = 0), and the smallest duty cycle is 0.0015% (PCA0CPn = 0xFFFF). A 0% duty cycle may be generated by clearing the ECOMn bit to ‘0’.

Figure 16.9. PCA 16-Bit PWM Mode



16.3. Watchdog Timer Mode

A programmable watchdog timer (WDT) function is available through the PCA Module 2. The WDT is used to generate a reset if the time between writes to the WDT update register (PCA0CPH2) exceed a specified limit. The WDT can be configured and enabled/disabled as needed by software.

With the WDTE bit set in the PCA0MD register, Module 2 operates as a watchdog timer (WDT). The Module 2 high byte is compared to the PCA counter high byte; the Module 2 low byte holds the offset to be used when WDT updates are performed. **The Watchdog Timer is enabled on reset. Writes to some PCA registers are restricted while the Watchdog Timer is enabled.**

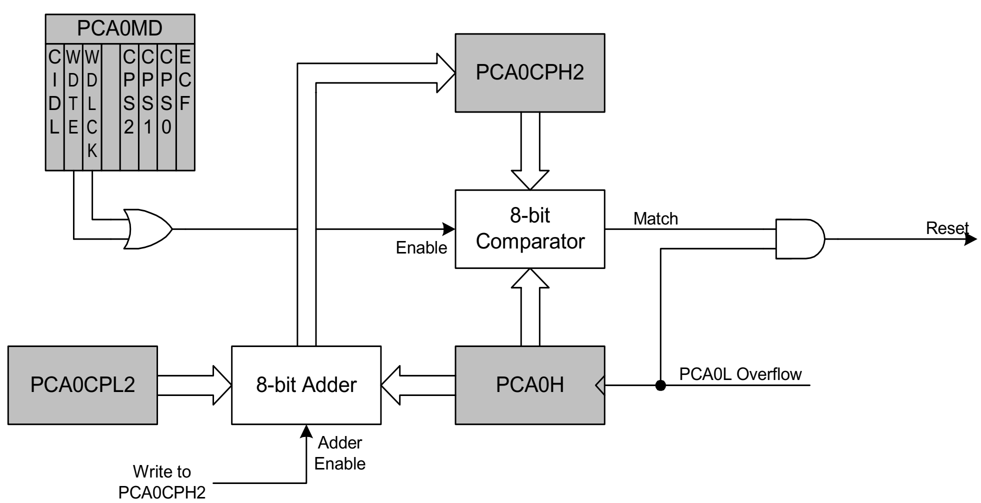
16.3.1. Watchdog Timer Operation

While the WDT is enabled:

- PCA counter is forced on.
- Writes to PCA0L and PCA0H are not allowed.
- PCA clock source bits (CPS2-CPS0) are frozen.
- PCA Idle control bit (CIDL) is frozen.
- Module 2 is forced into software timer mode.
- Writes to the module 2 mode register (PCA0CPM2) are disabled.

While the WDT is enabled, writes to the CR bit will not change the PCA counter state; the counter will run until the WDT is disabled. The PCA counter run control (CR) will read zero if the WDT is enabled but user software has not enabled the PCA counter. If a match occurs between PCA0CPH2 and PCA0H while the WDT is enabled, a reset will be generated. To prevent a WDT reset, the WDT may be updated with a write of any value to PCA0CPH2. Upon a PCA0CPH2 write, PCA0H plus the offset held in PCA0CPL2 is loaded into PCA0CPH2 (See Figure 16.10).

Figure 16.10. PCA Module 2 with Watchdog Timer Enabled



Note that the 8-bit offset held in PCA0CPH2 is compared to the upper byte of the 16-bit PCA counter. This offset value is the number of PCA0L overflows before a reset. Up to 256 PCA clocks may pass before the first PCA0L overflow occurs, depending on the value of the PCA0L when the update is performed. The total offset is then given (in PCA clocks) by Equation 16.4, where PCA0L is the value of the PCA0L register at the time of the update.

Equation 16.4. Watchdog Timer Offset in PCA Clocks

$$Offset = (256 \times PCA0CPL2) + (256 - PCA0L)$$

The WDT reset is generated when PCA0L overflows while there is a match between PCA0CPH2 and PCA0H. Software may force a WDT reset by writing a '1' to the CCF2 flag (PCA0CN.2) while the WDT is enabled.

16.3.2. Watchdog Timer Usage

To configure the WDT, perform the following tasks:

- Disable the WDT by writing a '0' to the WDTE bit.
- Select the desired PCA clock source (with the CPS2-CPS0 bits).
- Load PCA0CPL2 with the desired WDT update offset value.
- Configure the PCA Idle mode (set CIDL if the WDT should be suspended while the CPU is in Idle mode).
- Enable the WDT by setting the WDTE bit to '1'.

The PCA clock source and Idle mode select cannot be changed while the WDT is enabled. The Watchdog Timer is enabled by setting the WDTE or WDLCK bits in the PCA0MD register. When WDLCK is set, the WDT cannot be disabled until the next system reset. If WDLCK is not set, the WDT is disabled by clearing the WDTE bit.

The WDT is enabled following any reset. The PCA0 counter clock defaults to the system clock divided by 12, PCA0L defaults to 0x00, and PCA0CPL2 defaults to 0x00. Using Equation 16.4, this results in a WDT timeout interval of 3072 system clock cycles. Table 16.3 lists some example timeout intervals for typical system clocks, assuming SYSCLK / 12 as the PCA clock source.

Table 16.3. Watchdog Timer Timeout Intervals†

System Clock (Hz)	PCA0CPL2	Timeout Interval (ms)
24,500,000	255	32.1
24,500,000	128	16.2
24,500,000	32	4.1
18,432,000	255	42.7
18,432,000	128	21.5
18,432,000	32	5.5
11,059,200	255	71.1
11,059,200	128	35.8
11,059,200	32	9.2
3,060,000††	255	257
3,060,000††	128	129.5
3,060,000††	32	33.1
32,000	255	24576
32,000	128	12384
32,000	32	3168

†Assumes SYSCLK / 12 as the PCA clock source, and a PCA0L value of 0x00 at the update time.

††Internal oscillator reset frequency.

16.4. Register Descriptions for PCA

Following are detailed descriptions of the special function registers related to the operation of the PCA.

Figure 16.11. PCA0CN: PCA Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
CF	CR	-	-	-	CCF2	CCF1	CCF0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xD8

Bit7: CF: PCA Counter/Timer Overflow Flag.
Set by hardware when the PCA Counter/Timer overflows from 0xFFFF to 0x0000. When the Counter/Timer Overflow (CF) interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit6: CR: PCA Counter/Timer Run Control.
This bit enables/disables the PCA Counter/Timer.
0: PCA Counter/Timer disabled.
1: PCA Counter/Timer enabled.

Bits5-3: UNUSED. Read = 000b, Write = don't care.

Bit2: CCF2: PCA Module 2 Capture/Compare Flag.
This bit is set by hardware when a match or capture occurs. When the CCF2 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit1: CCF1: PCA Module 1 Capture/Compare Flag.
This bit is set by hardware when a match or capture occurs. When the CCF1 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit0: CCF0: PCA Module 0 Capture/Compare Flag.
This bit is set by hardware when a match or capture occurs. When the CCF0 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Figure 16.12. PCA0MD: PCA Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
CIDL	WDTE	WDLCK	-	CPS2	CPS1	CPS0	ECF	01000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xD9

Bit7: CIDL: PCA Counter/Timer Idle Control.
Specifies PCA behavior when CPU is in Idle Mode.
0: PCA continues to function normally while the system controller is in Idle Mode.
1: PCA operation is suspended while the system controller is in Idle Mode.

Bit6: WDTE: Watchdog Timer Enable
If this bit is set, PCA Module 2 is used as the Watchdog Timer.
0: Watchdog Timer disabled.
1: PCA Module 2 enabled as Watchdog Timer.

Bit5: WDLCK: Watchdog Timer Lock
This bit locks/unlocks the Watchdog Timer Enable. When WDLCK is set, the Watchdog Timer may not be disabled until the next system reset.
0: Watchdog Timer Enable unlocked.
1: Watchdog Timer Enable locked.

Bit4: UNUSED. Read = 0b, Write = don't care.

Bits3-1: CPS2-CPS0: PCA Counter/Timer Pulse Select.
These bits select the clock source for the PCA counter

CPS2	CPS1	CPS0	Timebase
0	0	0	System clock divided by 12
0	0	1	System clock divided by 4
0	1	0	Timer 0 overflow
0	1	1	High-to-low transitions on ECI (max rate = system clock divided by 4)
1	0	0	System clock
1	0	1	External clock divided by 8†
1	1	0	Reserved
1	1	1	Reserved

†External oscillator source divided by 8 is synchronized with the system clock.

Bit0: ECF: PCA Counter/Timer Overflow Interrupt Enable.
This bit sets the masking of the PCA Counter/Timer Overflow (CF) interrupt.
0: Disable the CF interrupt.
1: Enable a PCA Counter/Timer Overflow interrupt when CF (PCA0CN.7) is set.

Note: When the WDTE bit is set to '1', the PCA0MD register cannot be modified. To change the contents of the PCA0MD register, the Watchdog Timer must first be disabled.

Figure 16.13. PCA0CPMn: PCA Capture/Compare Mode Registers

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
PWM16n	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xDA, 0xDB, 0xDC

PCA0CPMn Address: PCA0CPM0 = 0xDA (n = 0)
PCA0CPM1 = 0xDB (n = 1)
PCA0CPM2 = 0xDC (n = 2)

Bit7: PWM16n: 16-bit Pulse Width Modulation Enable.
This bit selects 16-bit mode when Pulse Width Modulation mode is enabled (PWMn = 1).
0: 8-bit PWM selected.
1: 16-bit PWM selected.

Bit6: ECOMn: Comparator Function Enable.
This bit enables/disables the comparator function for PCA Module n.
0: Disabled.
1: Enabled.

Bit5: CAPPn: Capture Positive Function Enable.
This bit enables/disables the positive edge capture for PCA Module n.
0: Disabled.
1: Enabled.

Bit4: CAPNn: Capture Negative Function Enable.
This bit enables/disables the negative edge capture for PCA Module n.
0: Disabled.
1: Enabled.

Bit3: MATn: Match Function Enable.
This bit enables/disables the match function for PCA Module n. When enabled, matches of the PCA counter with a module's capture/compare register cause the CCFn bit in PCA0MD register to be set to logic 1.
0: Disabled.
1: Enabled.

Bit2: TOGn: Toggle Function Enable.
This bit enables/disables the toggle function for PCA Module n. When enabled, matches of the PCA counter with a module's capture/compare register cause the logic level on the CEXn pin to toggle. If the PWMn bit is also set to logic 1, the module operates in Frequency Output Mode.
0: Disabled.
1: Enabled.

Bit1: PWMn: Pulse Width Modulation Mode Enable.
This bit enables/disables the PWM function for PCA Module n. When enabled, a pulse width modulated signal is output on the CEXn pin. 8-bit PWM is used if PWM16n is cleared; 16-bit mode is used if PWM16n is set to logic 1. If the TOGn bit is also set, the module operates in Frequency Output Mode.
0: Disabled.
1: Enabled.

Bit0: ECCFn: Capture/Compare Flag Interrupt Enable.
This bit sets the masking of the Capture/Compare Flag (CCFn) interrupt.
0: Disable CCFn interrupts.
1: Enable a Capture/Compare Flag interrupt request when CCFn is set.

Figure 16.14. PCA0L: PCA Counter/Timer Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xF9

Bits 7-0: PCA0L: PCA Counter/Timer Low Byte.
The PCA0L register holds the low byte (LSB) of the 16-bit PCA Counter/Timer.

Figure 16.15. PCA0H: PCA Counter/Timer High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xFA

Bits 7-0: PCA0H: PCA Counter/Timer High Byte.
The PCA0H register holds the high byte (MSB) of the 16-bit PCA Counter/Timer.

Figure 16.16. PCA0CPLn: PCA Capture Module Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xFB, 0xE9, 0xEB

PCA0CPLn Address: PCA0CPL0 = 0xFB (n = 0)
PCA0CPL1 = 0xE9 (n = 1)
PCA0CPL2 = 0xEB (n = 2)

Bits7-0: PCA0CPLn: PCA Capture Module Low Byte.
The PCA0CPLn register holds the low byte (LSB) of the 16-bit capture Module n.

Figure 16.17. PCA0CPHn: PCA Capture Module High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xFC, 0xEA, 0xEC

PCA0CPHn Address: PCA0CPH0 = 0xFC (n = 0)
PCA0CPH1 = 0xEA (n = 1)
PCA0CPH2 = 0xEC(n = 2)

Bits7-0: PCA0CPHn: PCA Capture Module High Byte.
The PCA0CPHn register holds the high byte (MSB) of the 16-bit capture Module n.

17. C2 INTERFACE

C8051F300/1/2/3/4/5 devices include an on-chip Cygnal 2-Wire (C2) debug interface to allow FLASH programming, boundary scan functions, and in-system debugging with the production part installed in the end application. The C2 interface operates similar to JTAG, where the three JTAG data signals (TDI, TDO, TMS) are mapped into one bi-directional C2 data signal (C2D). See the C2 Interface Specification for details on the C2 protocol.

17.1. C2 Interface Registers

The following describes the C2 registers necessary to perform FLASH programming and boundary scan functions through the C2 interface. All C2 registers are accessed through the C2 interface as described in the C2 Interface Specification.

Figure 17.1. C2ADD: C2 Address Register

								Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

Bits7-0: The C2ADD register is accessed via the C2 interface to select the target Data register for C2 Data Read and Data Write commands.

Address	Description
0x00	Selects the Device ID register for Data Read instructions
0x01	Selects the Revision ID register for Data Read instructions
0x02	Selects the C2 FLASH Programming Control register for Data Read/Write instructions
0xB4	Selects the C2 FLASH Programming Data register for Data Read/Write instructions
0x80	Selects the Port0 register for Data Read/Write instructions
0xF1	Selects the Port0 Input Mode register for Data Read/Write instructions
0xA4	Selects the Port0 Output Mode register for Data Read/Write instructions

Figure 17.2. DEVICEID: C2 Device ID Register

								Reset Value
								00000100
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

This read-only register returns the 8-bit device ID: 0x04 (C8051F300/1/2/3).

Figure 17.3. REVID: C2 Revision ID Register

								Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

This read-only register returns the 8-bit revision ID: 0x00 (Revision A)

Figure 17.4. FPCTL: C2 FLASH Programming Control Register

								Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

Bits7-0 FPCTL: FLASH Programming Control Register
This register is used to enable FLASH programming via the C2 interface. To enable C2 FLASH programming, the following codes must be written in order: 0x02, 0x01. Note that once C2 FLASH programming is enabled, a system reset must be issued to resume normal operation.

Figure 17.5. FPDAT: C2 FLASH Programming Data Register

								Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

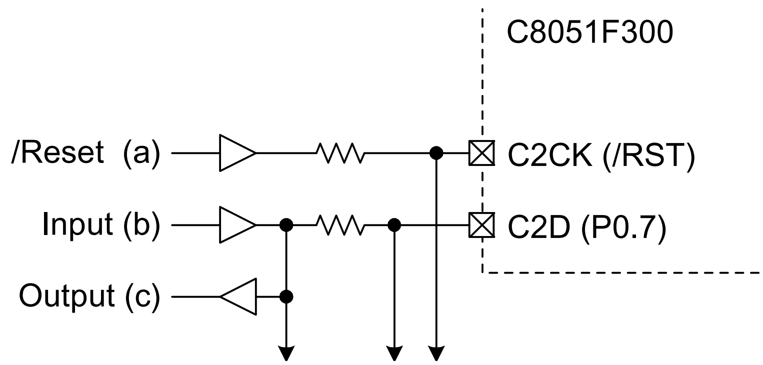
Bits7-0: FPDAT: C2 FLASH Programming Data Register
This register is used to pass FLASH commands, addresses, and data during C2 FLASH accesses. Valid commands are listed below.

Code	Command
0x06	FLASH Block Read
0x07	FLASH Block Write
0x08	FLASH Page Erase
0x03	Device Erase

17.2. C2 Pin Sharing

The C2 protocol allows the C2 pins to be shared with user functions so that in-system debugging, FLASH programming, and boundary scan functions may be performed. This is possible because C2 communication is typically performed when the device is in the halt state, where all on-chip peripherals and user software are stalled. In this halted state, the C2 interface can safely 'borrow' the C2CK (normally /RST) and C2D (normally P0.7) pins. In most applications, external resistors are required to isolate C2 interface traffic from the user application. A typical isolation configuration is shown in Figure 17.6.

Figure 17.6. Typical C2 Pin Sharing



The configuration in Figure 17.6 assumes the following:

1. The user input (b) cannot change state while the target device is halted.
2. The /RST pin on the target device is used as an input only.

Additional resistors may be necessary depending on the specific application.

Disclaimers

Life support: These products are not designed for use in life support appliances or systems where malfunction of these products can reasonably be expected to result in personal injury. Cygnal Integrated Products customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Cygnal Integrated Products for any damages resulting from such applications.

Right to make changes: Cygnal Integrated Products reserves the right to make changes, without notice, in the products, including circuits and/or software, described or contained herein in order to improve design and/or performance. Cygnal Integrated Products assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work infringement, unless otherwise specified.

CIP-51 is a trademark of Cygnal Integrated Products, Inc.
MCS-51 and SMBus are trademarks of Intel Corporation.
I^2C is a trademark of Philips Semiconductor.

CYGNAL INTEGRATED PRODUCTS
4301 Westbank Drive
Suite B-100
Austin, TX 78746
www.cygnal.com

Senior Design:

Appendix F

RFM®

TR1000

916.50 MHz Hybrid Transceiver

- ***Designed for Short-Range Wireless Data Communications***
- ***Supports RF Data Transmission Rates Up to 115.2 kbps***
- ***3 V, Low Current Operation plus Sleep Mode***
- ***Stable, Easy to Use, Low External Parts Count***

The TR1000 hybrid transceiver is ideal for short-range wireless data applications where robust operation, small size, low power consumption and low cost are required. The TR1000 employs RFM's amplifier-sequenced hybrid (ASH) architecture to achieve this unique blend of characteristics. All critical RF functions are contained in the hybrid, simplifying and speeding design-in. The receiver section of the TR1000 is sensitive and stable. A wide dynamic range log detector, in combination with digital AGC and a compound data slicer, provide robust performance in the presence of on-channel interference or noise. Two stages of SAW filtering provide excellent receiver out-of-band rejection. The transmitter includes provisions for both on-off keyed (OOK) and amplitude-shift keyed (ASK) modulation. The transmitter employs SAW filtering to suppress output harmonics, facilitating compliance with FCC 15.249 and similar regulations.

Absolute Maximum Ratings

Rating	Value	Units
Power Supply and All Input/Output Pins	-0.3 to +4.0	V
Non-Operating Case Temperature	-50 to +100	°C
Soldering Temperature (10 seconds)	230	°C

Electrical Characteristics (typical values given for 3.0 Vdc power supply, 25 °C)

Characteristic	Sym	Notes	Minimum	Typical	Maximum	Units
Operating Frequency	f_O		916.30		916.70	MHz
Modulation Type			OOK/ASK			
OOK Data Rate					30	kbps
ASK Data Rate					115.2	kbps
Receiver Performance, High Sensitivity Mode						
Sensitivity, 2.4 kbps, 10-3 BER, AM Test Method		1		-106		dBm
Sensitivity, 2.4 kbps, 10-3 BER, Pulse Test Method		1		-100		dBm
Current, 2.4 kbps (R_{PR} = 330 K)		2		3.0		mA
Sensitivity, 19.2 kbps, 10-3 BER, AM Test Method		1		-101		dBm
Sensitivity, 19.2 kbps, 10-3 BER, Pulse Test Method		1		-95		dBm
Current, 19.2 kbps (R_{PR} = 330 K)		2		3.1		mA
Sensitivity, 115.2 kbps, 10-3 BER, AM Test Method		1		-97		dBm
Sensitivity, 115.2 kbps, 10-3 BER, Pulse Test Method		1		-91		dBm
Current, 115.2 kbps				3.8		mA
Receiver Performance, Low Current Mode						
Sensitivity, 2.4 kbps, 10-3 BER, AM Test Method		1		-104		dBm
Sensitivity, 2.4 kbps, 10-3 BER, Pulse Test Method		1		-98		dBm
Current, 2.4 kbps (R_{PR} = 1100 K)		2		1.8		mA
Receiver Out-of-Band Rejection, ±5% f_O	$R_{\pm 5\%}$	3		80		dB
Receiver Ultimate Rejection	R_{ULT}	3		100		dB

Electrical Characteristics (typical values given for 3.0 Vdc power supply, 25 °C)

Characteristic	Sym	Notes	Minimum	Typical	Maximum	Units
Transmitter Performance						
Peak RF Output Power, 450 µA TXMOD Current	P_O	3		1.5		dBm
Peak Current, 450 µA TXMOD Current	I_{TP}	3		12		mA
2nd - 4th Harmonic Outputs		3			-50	dBm
5th - 10th Harmonic Outputs		3			-55	dBm
Non-harmonic Spurious Outputs		3			-50	dBm
OOK Turn On/Turn Off Times	t_{ON}/t_{OFF}	4			12/6	µs
ASK Output Rise/Fall Times	t_{TR}/t_{TF}	4			1.1/1.1	µs
Sleep Mode Current	I_S			0.7		µs
Power Supply Voltage Range	V_{CC}		2.2		3.7	Vdc
Power Supply Votage Ripple					10	mV_{P-P}
Ambient Operating Temperature	T_A		-40		85	°C

Notes:

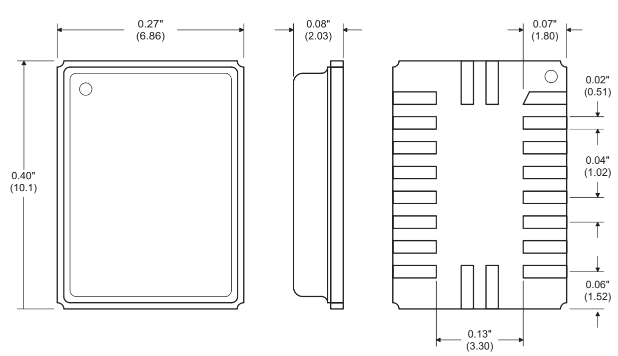
1. Typical sensitivity data is based on a 10^{-3} bit error rate (BER), using DC-balanced data. There are two test methods commonly used to measure OOK/ASK receiver sensitivity, the “100% AM” test method and the “Pulse” test method. Sensitivity data is given for both test methods. See Appendix 3.8 in the *ASH Transceiver Designer’s Guide* for the details of each test method, and for sensitivity curves for a 2.2 to 3.7 V supply voltage range at five operating temperatures. The application/test circuit and component values are shown on the next page and in the *Designer’s Guide*.

2. At low data rates it is possible to adjust the ASH pulse generator to trade-off some receiver sensitivity for lower operating current. Sensitivity data and receiver current are given at 2.4 kbps for both high sensitivity operation (R_{PR} = 330 K) and low current operation (R_{PR} = 1100 K).

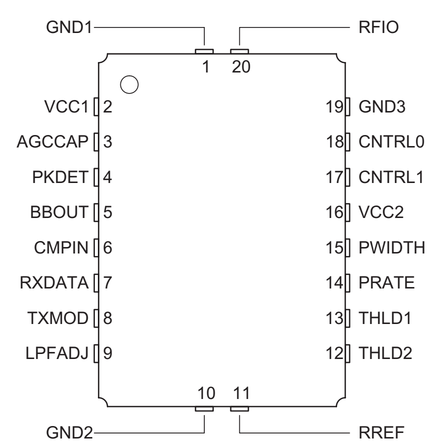
3. Data is given with the ASH radio matched to a 50 ohm load. Matching component values are given on the next page.

4. See Table 1 on Page 8 for additional information on ASH radio event timing.

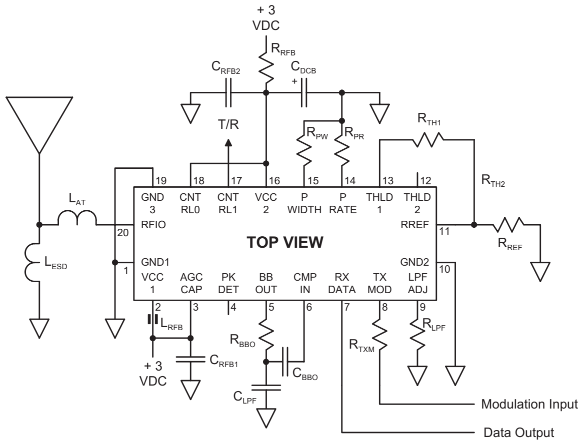
SM-20H Package Drawing



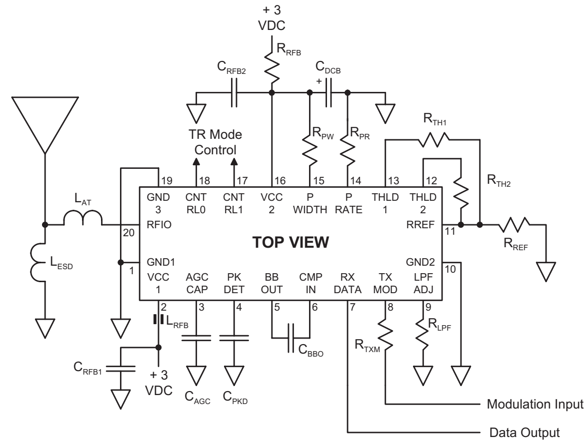
ASH Transceiver Pin Out



ASH Transceiver Application Circuit
OOK Configuration



ASH Transceiver Application Circuit
ASK Configuration



Transceiver Set-Up, 3.0 Vdc, -40 to +85 ^{0}C

Item	Symbol	OOK	OOK	ASK	Units	Notes
Encoded Data Rate	DR_{NOM}	2.4	19.2	115.2	kbps	see pages 1 & 2
Minimum Signal Pulse	SP_{MIN}	416.67	52.08	8.68	μs	single bit
Maximum Signal Pulse	SP_{MAX}	1666.68	208.32	34.72	μs	4 bits of same value
AGCCAP Capacitor	C_{AGC}	-	-	2200	pF	±10% ceramic
PKDET Capacitor	C_{PKD}	-	-	0.001	μF	±10% ceramic
BBOUT Capacitor	C_{BBO}	0.1	0.015	0.0027	μF	±10% ceramic
BBOUT Resistor	R_{BBO}	12	0	0	K	±5%
LPFAUX Capacitor	C_{LPF}	0.0047	-	-	μF	±5%
TXMOD Resistor	R_{TXM}	4.7	4.7	4.7	K	±5%, for 1.5 dBm output
LPFADJ Resistor	R_{LPF}	330	100	15	K	±5%
RREF Resistor	R_{REF}	100	100	100	K	±1%
THLD2 Resistor	R_{TH2}	-	-	100	K	±1%, for 6 dB below peak
THLD1 Resistor	R_{TH1}	0	0	10	K	±1%, typical values
PRATE Resistor	R_{PR}	330	330	160	K	±5%
PWIDTH Resistor	R_{PW}	270 to GND	270 to GND	1000 to Vcc	K	±5%
DC Bypass Capacitor	C_{DCB}	4.7	4.7	4.7	μF	tantalum
RF Bypass Capacitor 1	C_{RFB1}	27	27	27	pF	±5% NPO
RF Bypass Capacitor 2	C_{RFB2}	100	100	100	pF	±5% NPO
RF Bypass Bead	L_{RFB}	Fair-Rite	Fair-Rite	Fair-Rite	vendor	2506033017YO or equivalent
Series Tuning Inductor	L_{AT}	10	10	10	nH	50 ohm antenna
Shunt Tuning/ESD Inductor	L_{ESD}	100	100	100	nH	50 ohm antenna

CAUTION: Electrostatic Sensitive Device. Observe precautions when handling.

ASH Transceiver Theory of Operation

Introduction

RFM's amplifier-sequenced hybrid (ASH) transceiver is specifically designed for short-range wireless data communication applications. The transceiver provides robust operation, very small size, low power consumption and low implementation cost. All critical RF functions are contained in the hybrid, simplifying and speeding design-in. The ASH transceiver can be readily configured to support a wide range of data rates and protocol requirements. The transceiver features excellent suppression of transmitter harmonics and virtually no RF emissions when receiving, making it easy to certify to short-range (unlicensed) radio regulations.

Amplifier-Sequenced Receiver Operation

The ASH transceiver's unique feature set is made possible by its system architecture. The heart of the transceiver is the amplifier-sequenced receiver section, which provides more than 100 dB of stable RF and detector gain without any special shielding or decoupling provisions. Stability is achieved by distributing the total RF gain over *time*. This is in contrast to a superheterodyne receiver, which achieves stability by distributing total RF gain over multiple frequencies.

Figure 1 shows the basic block diagram and timing cycle for an amplifier-sequenced receiver. Note that the bias to RF amplifiers RFA1 and RFA2 are independently controlled by a pulse generator, and that the two amplifiers are coupled by a surface acoustic wave (SAW) delay line, which has a typical delay of 0.5 µs.

An incoming RF signal is first filtered by a narrow-band SAW filter, and is then applied to RFA1. The pulse generator turns RFA1 ON for 0.5 µs. The amplified signal from RFA1 emerges from the SAW delay line at the input to RFA2. RFA1 is now switched OFF and RFA2 is switched ON for 0.55 µs, amplifying the RF signal further. The ON time for RFA2 is usually set at 1.1 times the ON time for RFA1, as the filtering effect of the SAW delay line stretches the signal pulse from RFA1 somewhat. As shown in the timing diagram, RFA1 and RFA2 are never on at the same time, assuring excellent receiver stability. Note that the narrow-band SAW filter eliminates sampling sideband responses outside of the receiver passband, and the SAW filter and delay line act together to provide very high receiver ultimate rejection.

Amplifier-sequenced receiver operation has several interesting characteristics that can be exploited in system design. The RF amplifiers in an amplifier-sequenced receiver can be turned on and off almost instantly, allowing for very quick power-down (sleep) and wake-up times. Also, both RF amplifiers can be off between ON sequences to trade-off receiver noise figure for lower average current consumption. The effect on noise figure can be modeled as if RFA1 is on continuously, with an attenuator placed in front of it with a loss equivalent to $10^{*}\log_{10}$(RFA1 duty factor), where the duty factor is the average amount of time RFA1 is ON (up to 50%). Since an amplifier-sequenced receiver is inherently a sampling receiver, the overall cycle time between the start of one RFA1 ON sequence and

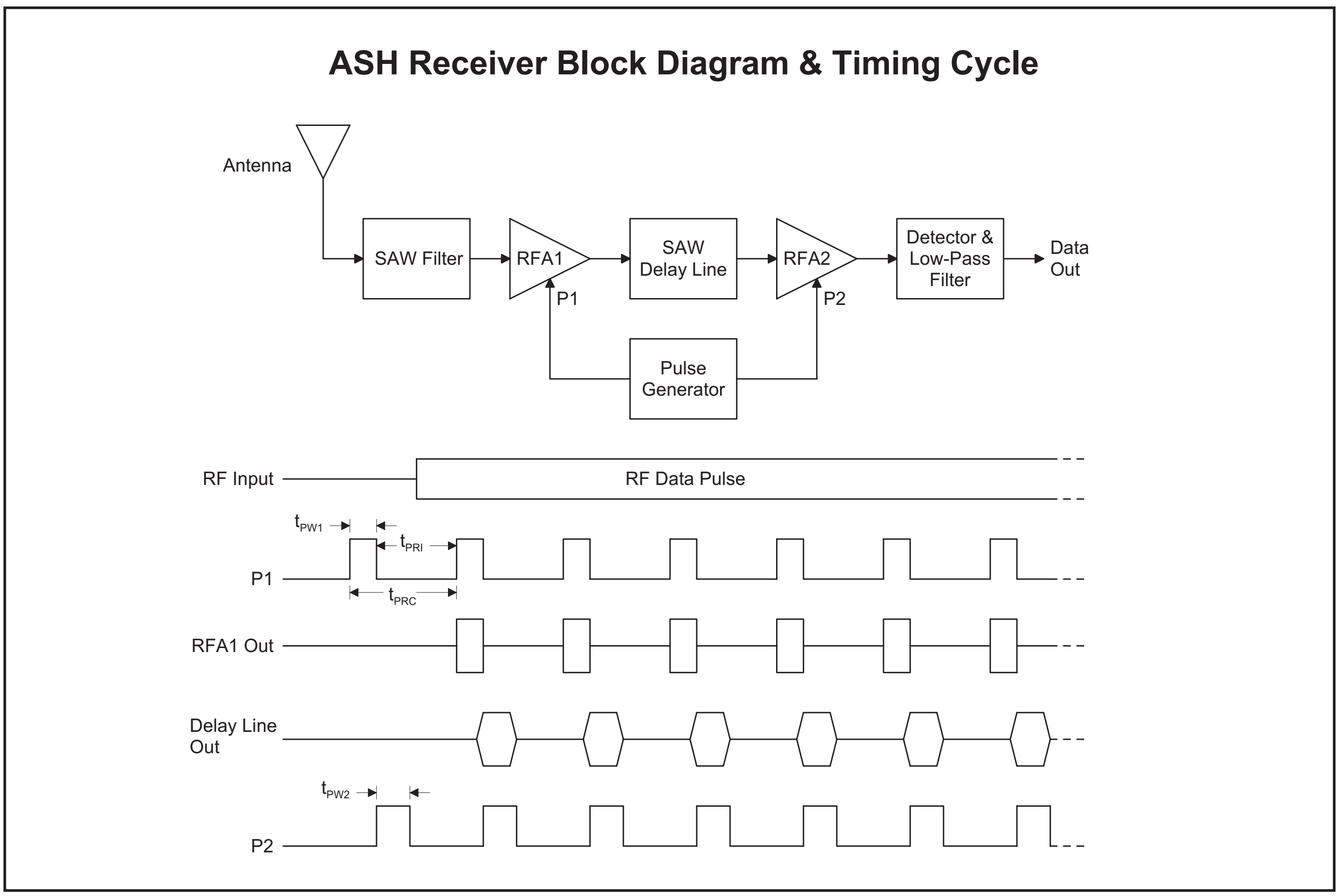


Figure 1

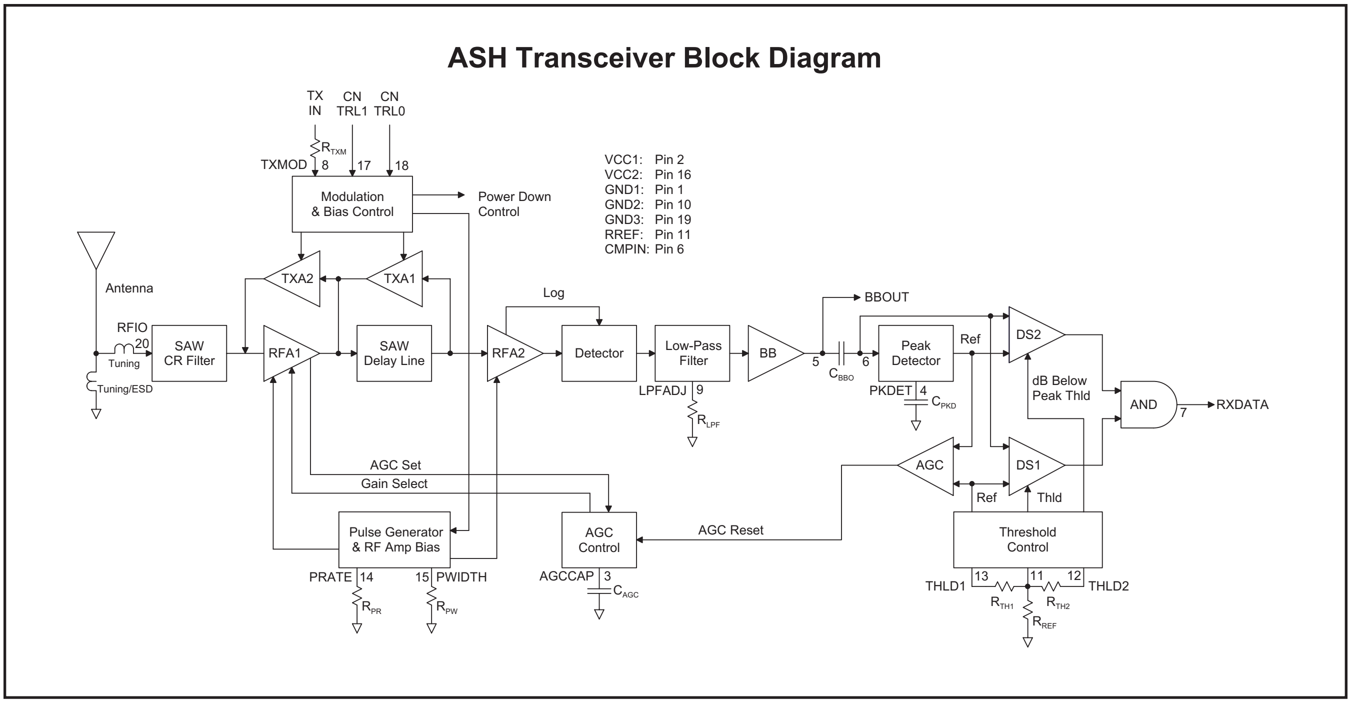


Figure 2

the start of the next RFA1 ON sequence should be set to sample the narrowest RF data pulse at least 10 times. Otherwise, significant edge jitter will be added to the detected data pulse.

ASH Transceiver Block Diagram

Figure 2 is the general block diagram of the ASH transceiver. Please refer to Figure 2 for the following discussions.

Antenna Port

The only external RF components needed for the transceiver are the antenna and its matching components. Antennas presenting an impedance in the range of 35 to 72 ohms resistive can be satisfactorily matched to the RFIO pin with a series matching coil and a shunt matching/ESD protection coil. Other antenna impedances can be matched using two or three components. For some impedances, two inductors and a capacitor will be required. A DC path from RFIO to ground is required for ESD protection.

Receiver Chain

The output of the SAW filter drives amplifier RFA1. This amplifier includes provisions for detecting the onset of saturation (AGC Set), and for switching between 35 dB of gain and 5 dB of gain (Gain Select). AGC Set is an input to the AGC Control function, and Gain Select is the AGC Control function output. ON/OFF control to RFA1 (and RFA2) is generated by the Pulse Generator & RF Amp Bias function. The output of RFA1 drives the SAW delay line, which has a nominal delay of 0.5 µs.

The second amplifier, RFA2, provides 51 dB of gain below saturation. The output of RFA2 drives a full-wave detector with 19 dB of threshold gain. The onset of saturation in each section of RFA2 is detected and summed to provide a logarithmic response. This is added to the output of the full-wave detector to produce an overall detector response that is square law for low signal levels, and transitions into a log response for high signal levels. This combination provides excellent threshold sensitivity and more than 70 dB of detector dynamic range. In combination with the 30 dB of AGC range in RFA1, more than 100 dB of receiver dynamic range is achieved.

The detector output drives a gyrator filter. The filter provides a three-pole, 0.05 degree equiripple low-pass response with excellent group delay flatness and minimal pulse ringing. The 3 dB bandwidth of the filter can be set from 4.5 kHz to 1.8 MHz with an external resistor.

The filter is followed by a base-band amplifier which boosts the detected signal to the BBOUT pin. When the receiver RF amplifiers are operating at a 50%-50% duty cycle, the BBOUT signal changes about 10 mV/dB, with a peak-to-peak signal level of up to 685 mV. For lower duty cycles, the mV/dB slope and peak-to-peak signal level are proportionately less. The detected signal is riding on a 1.1 Vdc level that varies somewhat with supply voltage, temperature, etc. BBOUT is coupled to the CMPIN pin or to an external data recovery process (DSP, etc.) by a series capacitor. The correct value of the series capacitor depends on data rate, data run length, and other factors as discussed in the *ASH Transceiver Designer's Guide.*

When an external data recovery process is used with AGC, BBOUT must be coupled to the external data recovery process and CMPIN by separate series coupling capacitors. The AGC reset function is driven by the signal applied to CMPIN.

When the transceiver is placed in power-down (sleep) or in a transmit mode, the output impedance of BBOUT becomes very high. This feature helps preserve the charge on the coupling capacitor to minimize data slicer stabilization time when the transceiver switches back to the receive mode.

Data Slicers

The CMPIN pin drives two data slicers, which convert the analog signal from BBOUT back into a digital stream. The best data slicer choice depends on the system operating parameters. Data slicer DS1 is a capacitively-coupled comparator with provisions for an adjustable threshold. DS1 provides the best performance at low signal-to-noise conditions. The threshold, or squelch, offsets the comparator's slicing level from 0 to 90 mV, and is set with a resistor between the RREF and THLD1 pins. This threshold allows a trade-off between receiver sensitivity and output noise density in the no-signal condition. For best sensitivity, the threshold is set to 0. In this case, noise is output continuously when no signal is present. This, in turn, requires the circuit being driven by the RXDATA pin to be able to process noise (and signals) continuously.

This can be a problem if RXDATA is driving a circuit that must "sleep" when data is not present to conserve power, or when it its necessary to minimize false interrupts to a multitasking processor. In this case, noise can be greatly reduced by increasing the threshold level, but at the expense of sensitivity. The best 3 dB bandwidth for the low-pass filter is also affected by the threshold level setting of DS1. The bandwidth must be increased as the threshold is increased to minimize data pulse-width variations with signal amplitude.

Data slicer DS2 can overcome this compromise once the signal level is high enough to enable its operation. DS2 is a "dB-below-peak" slicer. The peak detector charges rapidly to the peak value of each data pulse, and decays slowly in between data pulses (1:1000 ratio). The slicer trip point can be set from 0 to 120 mV below this peak value with a resistor between RREF and THLD2. A threshold of 60 mV is the most common setting, which equates to "6 dB below peak" when RFA1 and RFA2 are running a 50%-50% duty cycle. Slicing at the "6 dB-below-peak" point reduces the signal amplitude to data pulse-width variation, allowing a lower 3 dB filter bandwidth to be used for improved sensitivity.

DS2 is best for ASK modulation where the transmitted waveform has been shaped to minimize signal bandwidth. However, DS2 is subject to being temporarily "blinded" by strong noise pulses, which can cause burst data errors. Note that DS1 is active when DS2 is used, as RXDATA is the logical AND of the DS1 and DS2 outputs. DS2 can be disabled by leaving THLD2 disconnected. A non-zero DS1 threshold is required for proper AGC operation.

AGC Control

The output of the Peak Detector also provides an AGC Reset signal to the AGC Control function through the AGC comparator. The purpose of the AGC function is to extend the dynamic range of the receiver, so that two transceivers can operate close together when running ASK and/or high data rate modulation. The onset of saturation in the output stage of RFA1 is detected and generates the AGC Set signal to the AGC Control function. The AGC Control function then selects the 5 dB gain mode for RFA1. The AGC Comparator will send a reset signal when the Peak Detector output (multiplied by 0.8) falls below the threshold voltage for DS1.

A capacitor at the AGCCAP pin avoids AGC "chattering" during the time it takes for the signal to propagate through the low-pass filter and charge the peak detector. The AGC capacitor also allows the hold-in time to be set longer than the peak detector decay time to avoid AGC chattering during runs of "0" bits in the received data stream. Note that AGC operation requires the peak detector to be functioning, even if DS2 is not being used. AGC operation can be defeated by connecting the AGCCAP pin to Vcc. The AGC can be latched ON once engaged by connecting a 150 kilohm resistor between the AGCCAP pin and ground in lieu of a capacitor.

Receiver Pulse Generator and RF Amplifier Bias

The receiver amplifier-sequence operation is controlled by the Pulse Generator & RF Amplifier Bias module, which in turn is controlled by the PRATE and PWIDTH input pins, and the Power Down (sleep) Control Signal from the Modulation & Bias Control function.

In the low data rate mode, the interval between the falling edge of one RFA1 ON pulse to the rising edge of the next RFA1 ON pulse t_{PRI} is set by a resistor between the PRATE pin and ground. The interval can be adjusted between 0.1 and 5 µs. In the high data rate mode (selected at the PWIDTH pin) the receiver RF amplifiers operate at a nominal 50%-50% duty cycle. In this case, the start-to-start period t_{PRC} for ON pulses to RFA1 are controlled by the PRATE resistor over a range of 0.1 to 1.1 µs.

In the low data rate mode, the PWIDTH pin sets the width of the ON pulse t_{PW1} to RFA1 with a resistor to ground (the ON pulse width t_{PW2} to RFA2 is set at 1.1 times the pulse width to RFA1 in the low data rate mode). The ON pulse width t_{PW1} can be adjusted between 0.55 and 1 µs. However, when the PWIDTH pin is connected to Vcc through a 1 M resistor, the RF amplifiers operate at a nominal 50%-50% duty cycle, facilitating high data rate operation. In this case, the RF amplifiers are controlled by the PRATE resistor as described above.

Both receiver RF amplifiers are turned off by the Power Down Control Signal, which is invoked in the sleep and transmit modes.

Transmitter Chain

The transmitter chain consists of a SAW delay line oscillator followed by a modulated buffer amplifier. The SAW filter suppresses transmitter harmonics to the antenna. Note that the same SAW devices used in the amplifier-sequenced receiver are reused in the transmit modes.

Transmitter operation supports two modulation formats, on-off keyed (OOK) modulation, and amplitude-shift keyed (ASK) modulation. When OOK modulation is chosen, the transmitter output turns completely off between "1" data pulses. When ASK modulation is chosen, a "1" pulse is represented by a higher transmitted power level, and a "0" is represented by a lower transmitted power level. OOK modulation provides compatibility with first-generation ASH technology, and provides for power conservation. ASK modulation must be used for high data rates (data pulses less than 30 µs). ASK modulation also reduces the effects of some types of interference and allows the transmitted pulses to be shaped to control modulation bandwidth.

The modulation format is chosen by the state of the CNTRL0 and the CNTRL1 mode control pins, as discussed below. When either modulation format is chosen, the receiver RF amplifiers are turned off. In the OOK mode, the delay line oscillator amplifier TXA1 and buffer amplifier TXA2 are turned off when the voltage to the TXMOD input falls below 220 mV. In the OOK mode, the data rate is limited by the turn-on and turn-off times of the delay line oscillator, which are 12 and 6 µs respectively. In the ASK mode TXA1 is biased ON continuously, and the output of TXA2 is modulated by the TXMOD input current. Minimum output power occurs in the ASK mode when the modulation driver sinks about 10 µA of current from the TXMOD pin.

The transmitter RF output power is proportional to the input current to the TXMOD pin. A series resistor is used to adjust the peak transmitter output power. 1.5 dBm of output power requires about 450 µA of input current.

Transceiver Mode Control

The four transceiver operating modes – receive, transmit ASK, transmit OOK, and power-down (sleep), are controlled by the Modulation & Bias Control function, and are selected with the CNTRL1 and CNTRL0 control pins. Setting CNTRL1 and CNTRL0 both high place the unit in the receive mode. Setting CNTRL1 high and CNTRL0 low place the unit in the ASK transmit mode. Setting CNTRL1 low and CNTRL0 high place the unit in the OOK transmit mode. Setting CNTRL1 and CNTRL0 both low place the unit in the power-down (sleep) mode. Note that the resistor driving TXMOD must be low in the receive and power-down modes. The PWIDTH resistor must also be low in the power down mode to minimize current. CNTRL1 and CNTRL0 are CMOS compatible inputs. These inputs must be held at a logic level; they cannot be left unconnected.

Transceiver Event Timing

Transceiver event timing is summarized in Table 1. Please refer to this table for the following discussions.

Turn-On Timing

The maximum time t_{PR} required for the receive function to become operational at turn on is influenced by two factors. All receiver circuitry will be operational 5 ms after the supply voltage reaches 2.2 Vdc. The BBOUT-CMPIN coupling-capacitor is then DC stabilized in 3 time constants ($3*t_{BBC}$). The total turn-on time to stable receiver operation for a 10 ms power supply rise time is:

$t_{PR} = 15\text{ ms} + 3*t_{BBC}$

The maximum time required for either the OOK or ASK transmitter mode to become operational is 5 ms after the supply voltage reaches 2.2 Vdc.

Receive-to-Transmit Timing

After turn on, the maximum time required to switch from receive to either transmit mode is 12 µs. Most of this time is due to the start-up of the transmitter oscillator.

Transmit-to-Receive Timing

The maximum time required to switch from the OOK or ASK transmit mode to the receive mode is $3*t_{BBC}$, where t_{BBC} is the BBOUT-CMPIN coupling-capacitor time constant. When the operating temperature is limited to 60 ^{o}C, the time required to switch from transmit to receive is dramatically less for short transmissions, as less charge leaks away from the BBOUT-CMPIN coupling capacitor.

Sleep and Wake-Up Timing

The maximum transition time from the receive mode to the power-down (sleep) mode t_{RS} is 10 µs after CNTRL1 and CNTRL0 are both low (1 µs fall time).

The maximum transition time from either transmit mode to the sleep mode (t_{TOS} and t_{TAS}) is 10 µs after CNTRL1 and CNTRL0 are both low (1 µs fall time).

The maximum transition time t_{SR} from the sleep mode to the receive mode is $3*t_{BBC}$, where t_{BBC} is the BBOUT-CMPIN coupling-capacitor time constant. When the operating temperature is limited to 60 ^{o}C, the time required to switch from sleep to receive is dramatically less for short sleep times, as less charge leaks away from the BBOUT-CMPIN coupling capacitor.

The maximum time required to switch from the sleep mode to either transmit mode (t_{STO} and t_{STA}) is 16 µs. Most of this time is due to the start-up of the transmitter oscillator.

AGC Timing

The maximum AGC engage time t_{AGC} is 5 µs after the reception of a -30 dBm RF signal with a 1 µs envelope rise time.

The minimum AGC hold-in time is set by the value of the capacitor at the AGCCAP pin. The hold-in time $t_{AGH} = C_{AGC}/19.1$, where t_{AGH} is in µs and C_{AGC} is in pF.

Peak Detector Timing

The Peak Detector attack time constant is set by the value of the capacitor at the PKDET pin. The attack time $t_{PKA} = C_{PKD}/4167$, where t_{PKA} is in µs and C_{PKD} is in pF. The Peak Detector decay time constant $t_{PKD} = 1000*t_{PKA}$.

Pulse Generator Timing

In the low data rate mode, the interval t_{PRI} between the falling edge of an ON pulse to the first RF amplifier and the rising edge of the next ON pulse to the first RF amplifier is set by a resistor R_{PR} between the PRATE pin and ground. The interval can be adjusted between 0.1 and 5 µs with a resistor in the range of 51 K to 2000 K. The value of the R_{PR} is given by:

$R_{PR} = 404*\ t_{PRI} + 10.5$, where t_{PRI} is in µs, and R_{PR} is in kilohms

In the high data rate mode (selected at the PWIDTH pin) the receiver RF amplifiers operate at a nominal 50%-50% duty cycle. In this case, the period t_{PRC} from the start of an ON pulse to the first RF amplifier to the start of the next ON pulse to the first RF amplifier is controlled by the PRATE resistor over a range of 0.1 to 1.1 µs using a resistor of 11 K to 220 K. In this case R_{PR} is given by:

$R_{PR} = 198*\ t_{PRC} - 8.51$, where t_{PRC} is in µs and R_{PR} is in kilohms

In the low data rate mode, the PWIDTH pin sets the width of the ON pulse to the first RF amplifier t_{PW1} with a resistor R_{PW} to ground (the ON pulse width to the second RF amplifier t_{PW2} is set at 1.1 times the pulse width to the first RF amplifier in the low data rate mode). The ON pulse width t_{PW1} can be adjusted between 0.55 and 1 µs with a resistor value in the range of 200 K to 390 K. The value of R_{PW} is given by:

$R_{PW} = 404*\ t_{PW1} - 18.6$, where t_{PW1} is in µs and R_{PW} is in kilohms

However, when the PWIDTH pin is connected to Vcc through a 1 M resistor, the RF amplifiers operate at a nominal 50%-50% duty cycle, facilitating high data rate operation. In this case, the RF amplifiers are controlled by the PRATE resistor as described above.

LPF Group Delay

The low-pass filter group delay is a function of the filter 3 dB bandwidth, which is set by a resistor R_{LPF} to ground at the LPFADJ pin. The minimum 3 dB bandwidth $f_{LPF} = 1445/R_{LPF}$, where f_{LPF} is in kHz, and R_{LPF} is in kilohms.

The maximum group delay $t_{FGD} = 1750/f_{LPF} = 1.21*R_{LPF}$, where t_{FGD} is in µs, f_{LPF} in kHz, and R_{LPF} in kilohms.

Transceiver Event Timing, 3.0 Vdc, -40 to +85 ^{0}C

Event	Symbol	Time	Min/Max	Test Conditions	Notes
Turn On to Receive	t_{PR}	$3*t_{BBC}$ + 15 ms	max	10 ms supply voltage rise time	time until receiver operational
Turn On to TXOOK	t_{PTO}	15 ms	max	10 ms supply voltage rise time	time until TXMOD can modulate transmitter
Turn On to TXASK	t_{PTA}	15 ms	max	10 ms supply voltage rise time	time until TXMOD can modulate transmitter
RX to TXOOK	t_{RTO}	12 µs	max	1 µs CNTRL1 fall time	TXMOD low 1 µs before CNTRL1 falls
RX to TXASK	t_{RTA}	12 µs	max	1 µs CNTRL0 fall time	TXMOD low 1 µs before CNTRL0 falls
TXOOK to RX	t_{TOR}	$3*t_{BBC}$	max	1 µs CNTRL1 rise time	time until receiver operational
TXASK to RX	t_{TAR}	$3*t_{BBC}$	max	1 µs CNTRL0 rise time	time until receiver operational
Sleep to RX	t_{SR}	$3*t_{BBC}$	max	1µs CNTRL0/CNTROL1 rise times	time until receiver operational
Sleep to TXOOK	t_{STO}	16 µs	max	1 µs CNTRL0 rise time	time until TXMOD can modulate transmitter
Sleep to TXASK	t_{STA}	16 µs	max	1 µs CNTRL1 rise time	time until TXMOD can modulate transmitter
RX to Sleep	t_{RS}	10 µs	max	1µs CNTRL0/CNTROL1 fall times	time until transceiver is in power-down mode
TXOOK to Sleep	t_{TOS}	10 µs	max	1 µs CNTRL0 fall time	time until transceiver is in power-down mode
TXASK to Sleep	t_{TAS}	10 µs	max	1 µs CNTRL1 fall time	time until transceiver is in power-down mode
AGC Engage	t_{AGC}	5 µs	max	1 µs rise time, -30 dBm signal	RFA1 switches from 35 to 5 dB gain
AGC Hold-In	t_{AGH}	$C_{AGC}/19.1$	min	CAGC in pF, t_{AGH} in µs	user selected; longer than t_{PKD}
PKDET Attack Time Constant	t_{PKA}	$C_{PKD}/4167$	min	C_{PKD} in pF, t_{PKA} in µs	user selected
PKDET Decay Time Constant	t_{PKD}	$1000*t_{PKA}$	min	t_{PKD} and t_{PKA} in µs	slaved to attack time
PRATE Interval	t_{PRI}	0.1 to 5 µs	range	low data rate mode	user selected mode
PWIDTH RFA1	t_{PW1}	0.55 to 1 µs	range	low data rate mode	user selected mode
PWIDTH RFA2	t_{PW2}	$1.1*t_{PW1}$	range	low data rate mode	user selected mode
PRATE Cycle	t_{PRC}	0.1 to 1.1 µs	range	high data rate mode	user selected mode
PWIDTH High (RFA1 & RFA2)	t_{PWH}	0.05 to 0.55 µs	range	high data rate mode	user selected mode
LPF Group Delay	t_{FGD}	$1750/f_{LPF}$	max	t_{FGD} in µs, f_{LPF} in kHz	user selected
LPF 3 dB Bandwidth	f_{LPF}	$1445/R_{LPF}$	min	f_{LPF} in kHz, R_{LPF} in kilohms	user selected
BBOUT-CMPIN Time Constant	t_{BBC}	$0.064*C_{BBO}$	min	t_{BBC} in µs, C_{BBO} in pF	user selected

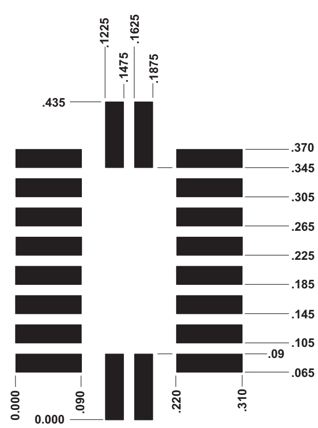
Table 1

Pin Descriptions

Pin	Name	Description
1	GND1	GND1 is the RF ground pin. GND2 and GND3 should be connected to GND1 by short, low-inductance traces.
2	VCC1	VCC1 is the positive supply voltage pin for the transmitter output amplifier and the receiver base-band circuitry. VCC1 is usually connected to the positive supply through a ferrite RF decoupling bead, which is bypassed by an RF capacitor on the *supply side*. See the *ASH Transceiver Designer's Guide* for additional information.
3	AGCCAP	This pin controls the AGC reset operation. A capacitor between this pin and ground sets the minimum time the AGC will hold-in once it is engaged. The hold-in time is set to avoid AGC chattering. For a given hold-in time t_{AGH}, the capacitor value C_{AGC} is: $C_{AGC} = 19.1^* t_{AGH}$, where t_{AGH} is in µs and C_{AGC} is in pF A ±10% ceramic capacitor should be used at this pin. The value of C_{AGC} given above provides a hold-in time between t_{AGH} and $2.65^* t_{AGH}$, depending on operating voltage, temperature, etc. The hold-in time is chosen to allow the AGC to ride through the longest run of zero bits that can occur in a received data stream. The AGC hold-in time can be greater than the peak detector decay time, as discussed below. However, the AGC hold-in time should not be set too long, or the receiver will be slow in returning to full sensitivity once the AGC is engaged by noise or interference. The use of AGC is optional when using OOK modulation with data pulses of at least 30 µs. AGC operation can be defeated by connecting this pin to Vcc. Active or latched AGC operation is required for ASK modulation and/or for data pulses of less than 30 µs. The AGC can be latched on once engaged by connecting a 150 K resistor between this pin and ground, instead of a capacitor. AGC operation depends on a functioning peak detector, as discussed below. The AGC capacitor is discharged in the receiver power-down (sleep) mode and in the transmit modes.
4	PKDET	This pin controls the peak detector operation. A capacitor between this pin and ground sets the peak detector attack and decay times, which have a fixed 1:1000 ratio. For most applications, these time constants should be coordinated with the base-band time constant. For a given base-band capacitor C_{BBO}, the capacitor value C_{PKD} is: $C_{PKD} = 0.33^* C_{BBO}$, where C_{BBO} and C_{PKD} are in pF A ±10% ceramic capacitor should be used at this pin. This time constant will vary between t_{PKA} and $1.5^* t_{PKA}$ with variations in supply voltage, temperature, etc. The capacitor is driven from a 200 ohm "attack" source, and decays through a 200 K load. The peak detector is used to drive the "dB-below-peak" data slicer and the AGC release function. The AGC hold-in time can be extended beyond the peak detector decay time with the AGC capacitor, as discussed above. Where low data rates and OOK modulation are used, the "dB-below-peak" data slicer and the AGC are optional. In this case, the PKDET pin and the THLD2 pin can be left unconnected, and the AGC pin can be connected to Vcc to reduce the number of external components needed. The peak detector capacitor is discharged in the receiver power-down (sleep) mode and in the transmit modes.
5	BBOUT	BBOUT is the receiver base-band output pin. This pin drives the CMPIN pin through a coupling capacitor C_{BBO} for internal data slicer operation. The time constant t_{BBC} for this connection is: $t_{BBC} = 0.064^*C_{BBO}$, where t_{BBC} is in µs and C_{BBO} is in pF A ±10% ceramic capacitor should be used between BBOUT and CMPIN. The time constant can vary between t_{BBC} and 1.8^*t_{BBC} with variations in supply voltage, temperature, etc. The optimum time constant in a given circumstance will depend on the data rate, data run length, and other factors as discussed in the *ASH Transceiver Designer's Guide*. A common criteria is to set the time constant for no more than a 20% voltage droop during SP_{MAX}. For this case: $C_{BBO} = 70^*SP_{MAX}$, where SP_{MAX} is the maximum signal pulse width in µs and C_{BBO} is in pF The output from this pin can also be used to drive an external data recovery process (DSP, etc.). The nominal output impedance of this pin is 1 K. When the receiver RF amplifiers are operating at a 50%-50% duty cycle, the BBOUT signal changes about 10 mV/dB, with a peak-to-peak signal level of up to 685 mV. For lower duty cycles, the mV/dB slope and peak-to-peak signal level are proportionately less. The signal at BBOUT is riding on a 1.1 Vdc value that varies somewhat with supply voltage and temperature, so it should be coupled through a capacitor to an external load. A load impedance of 50 K to 500 K in parallel with no more than 10 pF is recommended. When an external data recovery process is used with AGC, BBOUT must be coupled to the external data recovery process and CMPIN by separate series coupling capacitors. The AGC reset function is driven by the signal applied to CMPIN. When the transceiver is in power-down (sleep) or in a transmit mode, the output impedance of this pin becomes very high, preserving the charge on the coupling capacitor.
6	CMPIN	This pin is the input to the internal data slicers. It is driven from BBOUT through a coupling capacitor. The input impedance of this pin is 70 K to 100 K.
7	RXDATA	RXDATA is the receiver data output pin. This pin will drive a 10 pF, 500 K parallel load. The peak current available from this pin increases with the receiver low-pass filter cutoff frequency. In the power-down (sleep) or transmit modes, this pin becomes high impedance. If required, a 1000 K pull-up or pull-down resistor can be used to establish a definite logic state when this pin is high impedance. If a pull-up resistor is used, the positive supply end should be connected to a voltage no greater than Vcc + 200 mV.

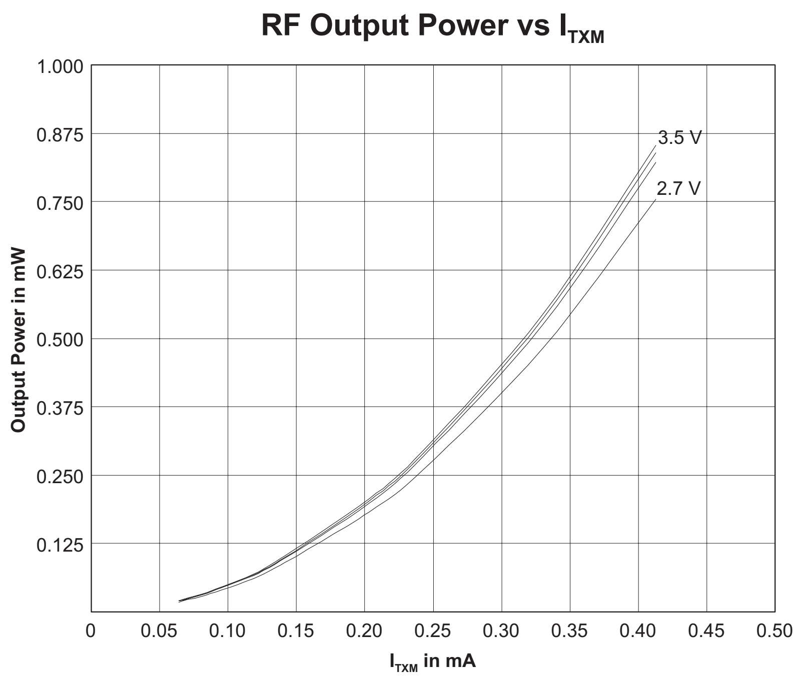
Pin	Name	Description
8	TXMOD	The transmitter RF output voltage is proportional to the input current to this pin. A series resistor is used to adjust the peak transmitter output voltage. 1.5 dBm of output power requires about 450 µA of input current. In the ASK mode, minimum output power occurs when the modulation driver sinks about 10 µA of current from this pin. In the OOK mode, input signals less than 220 mV completely turn the transmitter oscillator off. Internally, this pin appears to be a diode in series with a small resistor. Peak transmitter output power P_O for a 3 Vdc supply voltage is approximately: $P_O = 7*(I_{TXM})^2$, where P_O is in mW, and the peak modulation current I_{TXM} is in mA A ±5% resistor value is recommended. In the OOK mode, this pin is usually driven with a logic-level data input (unshaped data pulses). OOK modulation is practical for data pulses of 30 µs or longer. In the ASK mode, this pin accepts analog modulation (shaped or unshaped data pulses). ASK modulation is practical for data pulses 8.7 µs or longer. *The resistor driving this pin must be low in the receive and power-down (sleep) modes.* Please refer to the *ASH Transceiver Designer's Guide* for additional information on modulation techniques.
9	LPFADJ	This pin is the receiver low-pass filter bandwidth adjust. The filter bandwidth is set by a resistor R_{LPF} between this pin and ground. The resistor value can range from 330 K to 820 ohms, providing a filter 3 dB bandwidth f_{LPF} from 4.5 kHz to 1.8 MHz. The resistor value is determined by: $R_{LPF} = 1445/ f_{LPF}$, where R_{LPF} is in kilohms, and f_{LPF} is in kHz A ±5% resistor should be used to set the filter bandwidth. This will provide a 3 dB filter bandwidth between f_{LPF} and 1.3* f_{LPF} with variations in supply voltage, temperature, etc. The filter provides a three-pole, 0.05 degree equiripple phase response. The peak drive current available from RXDATA increases in proportion to the filter bandwidth setting.
10	GND2	GND2 is an IC ground pin. It should be connected to GND1 by a short, low inductance trace.
11	RREF	RREF is the external reference resistor pin. A 100 K reference resistor is connected between this pin and ground. A ±1% resistor tolerance is recommended. It is important to keep the total capacitance between ground, Vcc and this node to less than 5 pF to maintain current source stability. If THLD1 and/or THDL2 are connected to RREF through resistor values less that 1.5 K, their node capacitance must be added to the RREF node capacitance and the total should not exceed 5 pF.
12	THLD2	THLD2 is the "dB-below-peak" data slicer (DS2) threshold adjust pin. The threshold is set by a 0 to 200 K resistor R_{TH2} between this pin and RREF. Increasing the value of the resistor decreases the threshold below the peak detector value (increases difference) from 0 to 120 mV. For most applications, this threshold should be set at 6 dB below peak, or 60 mV for a 50%-50% RF amplifier duty cycle. The value of the THLD2 resistor is given by: $R_{TH2} = 1.67*V$, where R_{TH2} is in kilohms and the threshold V is in mV A ±1% resistor tolerance is recommended for the THLD2 resistor. Leaving the THLD2 pin open disables the dB-below-peak data slicer operation.
13	THLD1	The THLD1 pin sets the threshold for the standard data slicer (DS1) through a resistor R_{TH1} to RREF. The threshold is increased by increasing the resistor value. Connecting this pin directly to RREF provides zero threshold. The value of the resistor depends on whether THLD2 is used. For the case that THLD2 is not used, the acceptable range for the resistor is 0 to 100 K, providing a THLD1 range of 0 to 90 mV. The resistor value is given by: $R_{TH1} = 1.11*V$, where R_{TH1} is in kilohms and the threshold V is in mV For the case that THLD2 is used, the acceptable range for the THLD1 resistor is 0 to 200 K, again providing a THLD1 range of 0 to 90 mV. The resistor value is given by: $R_{TH1} = 2.22*V$, where R_{TH1} is in kilohms and the threshold V is in mV A ±1% resistor tolerance is recommended for the THLD1 resistor. Note that a non-zero DS1 threshold is required for proper AGC operation.
14	PRATE	The interval between the falling edge of an ON pulse to the first RF amplifier and the rising edge of the next ON pulse to the first RF amplifier t_{PRI} is set by a resistor R_{PR} between this pin and ground. The interval t_{PRI} can be adjusted between 0.1 and 5 µs with a resistor in the range of 51 K to 2000 K. The value of R_{PR} is given by: $R_{PR} = 404* t_{PRI} + 10.5$, where t_{PRI} is in µs, and R_{PR} is in kilohms A ±5% resistor value is recommended. When the PWIDTH pin is connected to Vcc through a 1 M resistor, the RF amplifiers operate at a nominal 50%-50% duty cycle, facilitating high data rate operation. In this case, the period t_{PRC} from start-to-start of ON pulses to the first RF amplifier is controlled by the PRATE resistor over a range of 0.1 to 1.1 µs using a resistor of 11 K to 220 K. In this case the value of R_{PR} is given by: $R_{PR} = 198* t_{PRC} - 8.51$, where t_{PRC} is in µs and R_{PR} is in kilohms A ±5% resistor value should also be used in this case. Please refer to the *ASH Transceiver Designer's Guide* for additional amplifier duty cycle information. It is important to keep the total capacitance between ground, Vcc and this pin to less than 5 pF to maintain stability.

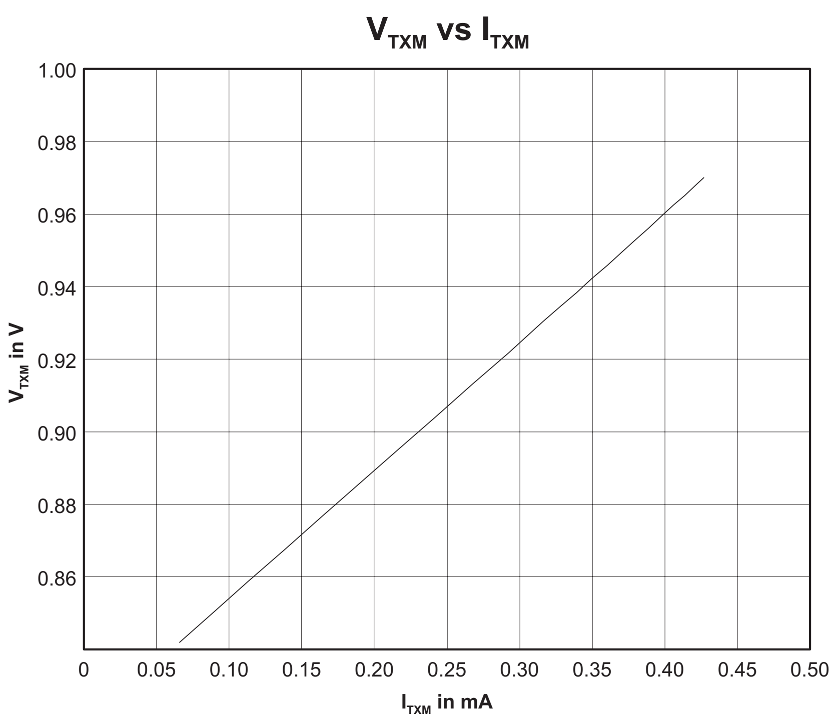
Pin	Name	Description
15	PWIDTH	The PWIDTH pin sets the width of the ON pulse to the first RF amplifier t_{PW1} with a resistor R_{PW} to ground (the ON pulse width to the second RF amplifier t_{PW2} is set at 1.1 times the pulse width to the first RF amplifier). The ON pulse width t_{PW1} can be adjusted between 0.55 and 1 µs with a resistor value in the range of 200 K to 390 K. The value of R_{PW} is given by: $R_{PW} = 404^* t_{PW1} - 18.6$, where t_{PW1} is in µs and R_{PW} is in kilohms A ±5% resistor value is recommended. When this pin is connected to Vcc through a 1 M resistor, the RF amplifiers operate at a nominal 50%-50% duty cycle, facilitating high data rate operation. In this case, the RF amplifier ON times are controlled by the PRATE resistor as described above. It is important to keep the total capacitance between ground, Vcc and this node to less than 5 pF to maintain stability. When using the high data rate operation with the sleep mode, connect the 1 M resistor between this pin and CNTRL1 (Pin 17), so this pin is low in the sleep mode.
16	VCC2	VCC2 is the positive supply voltage pin for the receiver RF section and transmitter oscillator. Pin 16 must be bypassed with an RF capacitor, and must also be bypassed with a 1 to 10 µF tantalum or electrolytic capacitor. See the *ASH Transceiver Designer's Guide* for additional information.
17	CNTRL1	CNTRL1 and CNTRL0 select the receive and transmit modes. CNTRL1 and CNTRL0 both high place the unit in the receive mode. CNTRL1 high and CNTRL0 low place the unit in the ASK transmit mode. CNTRL1 low and CNTRL0 high place the unit in the OOK transmit mode. CNTRL1 and CNTRL0 both low place the unit in the power-down (sleep) mode. CNTRL1 is a high-impedance input (CMOS compatible). An input voltage of 0 to 300 mV is interpreted as a logic low. An input voltage of Vcc - 300 mV or greater is interpreted as a logic high. An input voltage greater than Vcc + 200 mV should not be applied to this pin. A logic high requires a maximum source current of 40 µA. A logic low requires a maximum sink current of 25 µA (1 µA in sleep mode). This pin must be held at a logic level; it cannot be left unconnected.
18	CNTRL0	CNTRL0 is used with CNTRL1 to control the receive and transmit modes of the transceiver. CNTRL0 is a high-impedance input (CMOS compatible). An input voltage of 0 to 300 mV is interpreted as a logic low. An input voltage of Vcc - 300 mV or greater is interpreted as a logic high. An input voltage greater than Vcc + 200 mV should not be applied to this pin. A logic high requires a maximum source current of 40 µA. A logic low requires a maximum sink current of 25 µA (1 µA in sleep mode). This pin must be held at a logic level; it cannot be left unconnected.
19	GND3	GND3 is an IC ground pin. It should be connected to GND1 by a short, low inductance trace.
20	RFIO	RFIO is the RF input/output pin. This pin is connected directly to the SAW filter transducer. Antennas presenting an impedance in the range of 35 to 72 ohms resistive can be satisfactorily matched to this pin with a series matching coil and a shunt matching/ESD protection coil. Other antenna impedances can be matched using two or three components. For some impedances, two inductors and a capacitor will be required. A DC path from RFIO to ground is required for ESD protection.



Dimensions in inches.

SM-20H PCB Pad Layout





Note: Specifications subject to change without notice.

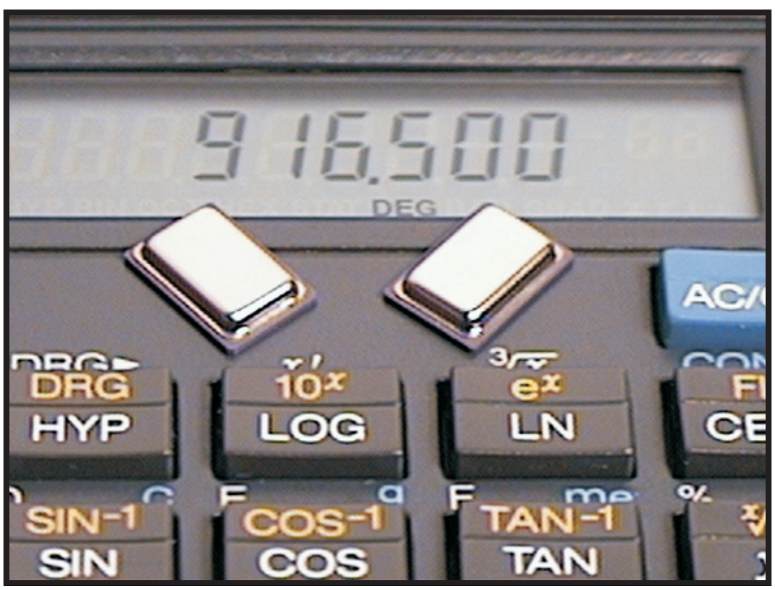
file: tr1000t.vp, 2003.04.06 rev

Senior Design:

Appendix G

ASH Transceiver Designer's Guide

Updated 2003.02.22



ASH Transceiver Designer's Guide

1 Introduction

2 ASH Transceiver Set-Up

Tables

Drawings

1 Introduction

1.1 Short-Range Wireless Data Communications

Short-range wireless systems transmit 0.0001 to 10 mW of RF power on frequencies from 300 to 960 MHz, and operate over distances of 3 to 100 meters (single hop). Once certified to comply with local communications regulations, they do not require a license or "air-time fee" for operation. Short-range wireless systems can be designed to operate from small batteries for extended periods of time. More than 150 million products will be manufactured this year that utilize short-range wireless for security, control and data transmission. Many new applications are emerging, and RFM estimates that more than 250 million short-range wireless products will be manufactured in 2005.

The classical uses for short-range wireless systems are one-way remote control and alarm links, including garage door openers, automotive "keyless entry" transmitters, and home security systems. Recently, a strong interest has also developed in two-way data communications applications. Short-range wireless data systems are used to eliminate nuisance cables on all types of digital products, much as cordless phones have eliminated cumbersome phone wires.

The following list of example applications demonstrates the diversity of uses for short-range wireless data systems:

- Wireless bar-code and credit-card readers
- Wireless and bar-code label printers and credit-card receipt printers
- Smart ID tags for inventory tracking and identification
- Wireless automatic utility meter reading systems
- Communications links for hand-held terminals, HPCs and PDAs
- Wireless keyboards, joysticks, mice and game controls
- Portable and field data logging
- Location tracking (follow-me phone extensions, etc.)
- Sports and medical telemetry
- Surveying system data links
- Engine diagnostic links
- Polled wireless security alarm sensors
- Authentication and access control tags

RFM's second-generation amplifier-sequenced hybrid (ASH) radios are specifically designed for short-range wireless applications. These radios provide robust operation, very small size, low power consumption and low implementation cost. All critical RF functions are contained in the hybrids, simplifying and speeding design-in. ASH radios can be readily configured to support a wide range of data rates and protocol requirements. These radios features excellent suppression of transmitter harmonics and virtually no RF emissions when receiving, making them easy to certify to short-range radio regulations. The ASH transceiver is the flagship of RFM's second-generation ASH radio product line.

While this designer's guide focuses on the ASH transceiver, most of the information provided is directly applicable to second-generation ASH transmitters and receivers. The exceptions are discussed in section 3.4 of the Appendix.

1.2 Operating Authorities

Low-power wireless products do not have to be individually licensed, but they are subject to regulation. Before low-power wireless systems can be marketed in most countries, they must be certified to comply with specific technical regulations. While these regulations vary from country to country, they follow the same general philosophy of assuring that short-range wireless systems will not significantly interfere with licensed radio systems. Regulations specify limitations on transmitted power, harmonic and spurious emission levels, transmitter frequency stability, and modulation bandwidth. See section 1.4.6 below for additional details.

1.3 Operating Distance

The operating distance of a low-power wireless system depends on transmitter power, receiver sensitivity, choice of antennas, data encoding, data rate, bit error rate (BER) requirements, the communication protocol used, the threshold (squelch) level used, the required fading margin, and especially the propagation environment. A "textbook" approach to estimating operating distance is as follows:

1. Determine the acceptable "clear channel" packet error rate (PER) you would like your system to achieve.

2. Estimate the bit error rate BER = PER/(number of bits per packet) based on the protocol used.

3. Estimate the signal-to-noise ratio (per bit) required to achieve the BER.

4. Estimate the needed signal strength at the receiver from the signal-to-noise ratio, receiver noise figure, implementation loss and receiver filter bandwidth.

5. Estimate the allowed path loss by adding the transmitter power (dB) to the transmitter and receiver antenna gains, and subtracting the fading margin and the required receiver signal strength.

6. Estimate the operating distance from the allowed path loss and the propagation characteristics of the local (application) environment.

This procedure is obviously complex, and many factors have to be estimated to make the calculation. The propagation loss of the local environment is especially difficult to estimate. Propagation loss in "free space" is proportional to $1/d^2$, but can be higher than $1/d^4$ in dense cubical office space. In many cases, a better estimate of operating distance can be made by using a Virtual Wire® Development Kit as a propagation survey tool.

An example operating distance calculation based on the above procedure is provided in the Appendix. Table 1.3.1 gives interference-free operating distance estimates for a number of environments. We stress again that it is very important to conduct "real world" range testing in several locations for your application in making an assessment of operating range.

Typical 916.5 MHz Operating Distances vs Data Rate, Byte to 12-bit Symbol Encoding, 20 dB Fade Margin

Environment	2.4 kbps		19.2 kbps		57.6 kbps		115.2 kbps	
	meters	feet	meters	feet	meters	feet	meters	feet
Free Space	117.0	385.0	104.0	339.0	92.3	302.0	65.3	214.0
Large Open Area, 1.5 m height	45.3	149.0	40.1	132.0	37.3	122.0	28.3	92.9
Open Office/Retail, 1.5 meter height	24.0	78.7	21.8	72.2	20.4	67.0	16.2	53.2
Dense Cubical office space	10.8	35.6	10.2	33.0	9.6	31.5	8.1	26.5

Notes:

1. 2.4 and 19.2 kbps data rate using OOK with DS1 low noise threshold
2. 57.6 and 115.2 kbps data rate using ASK with 6 dB below peak DS2 threshold
3. Transmitter power level based on FCC 15.249 limit

Table 1.3.1

1.4 Key System Issues

RFM supports hundreds of customers that engineer and manufacture short-range wireless products. The most successful customers approach their short-range wireless designs from a system point of view. In addition to the choice of radio technology, there are six other key system issues to consider in developing a short-range wireless product:

1.4.1 Fail-safe system design

Most short-range wireless systems operate with few interference problems. However, these systems operate on shared radio channels, so interference can occur at any place and at any time. *Products that incorporate short-range wireless technology must be designed so that a loss of communications due to radio interference or any other reason will not create a dangerous situation, damage equipment or property, or cause loss of valuable data.* The single most important consideration in designing a product that uses any short-range wireless technology is safety.

1.4.2 Antennas and propagation

Antenna choice and location - suitable antennas are crucial to the success of a low-power wireless application. Here are several key points to consider in using antennas in your application:

- Where possible, the antenna should be placed on the outside of the product. Also, try to place the antenna on the top of the product. If the product is "body worn", try to get the antenna away from the body as far as possible.

- Regulatory agencies prefer antennas that are permanently fixed to the product. In some cases, antennas can be supplied with a cable, provided a non-standard connector is used to discourage antenna substitution (these connectors are often referred to as “Part 15” connectors).
- An antenna can not be placed inside a metal case, as the case will shield it. Also, some plastics (and coatings) significantly attenuate RF signals and these materials should not be used for product cases, if the antenna is going to be inside the case.
- Many suitable antenna designs are possible, but efficient antenna development requires access to antenna test equipment such as a network analyzer, calibrated test antenna, antenna range, etc. Unless you have access to this equipment, the use of a standard antenna design or a consultant is recommended.
- A patch or slot antenna can be used in some applications where an external antenna would be subject to damage.

The human body readily absorbs RF radiation in the UHF frequency range, especially above 750 MHz. The signal from a body-worn transmitter can be attenuated 20 to 30 dB in any direction that passes through the user’s body. When designing body-worn products, you have to plan for this extra attenuation.

Mounting the antenna close to the user’s body will also reduce signal strength in directions away from the user’s body. Try not to mount the antenna any closer than 1.5 cm from the user’s body, with 2 to 3 cm preferred.

RF Propagation **-** indoor radio propagation is an issue for special consideration. In most indoor locations, “dead spots” can be found where reception is very difficult. These can occur even if there appears to be a line-of-sight relationship between the transmitter and receiver locations. These “dead spots”, or nulls, are due to multiple transmission paths existing between two points because of reflections off metal objects such as steel beams, concrete rebar, metal door, window and ceiling tile frames, etc. Nulls occur when the path lengths effectively differ by an odd half-wavelength. Deep nulls are usually very localized, and can be avoided by moving slightly. Hand-held applications usually involve some movement, so automatic packet retransmission often succeeds in completing the transmission as hand motion moves the node through the null and back into a good transmission point.

Diversity reception systems - diversity reception techniques are very helpful in reducing indoor null problems. Many short-range wireless systems involve communications between a master and multiple slave units. In this case, the master transmission can be sent twice; first from one master and then again from a second master in a different location. The nulls for each master will tend to be in different locations, so a slave is very likely to hear the transmission from one or the other master. Likewise, a transmission from a slave is likely to be heard by at least one of the masters.

For further information, see RFM's application note, *Antennas for Low Power Applications,* on RFM's web site (http://www.rfm.com). The application note includes test results on eleven types of antennas for short-range wireless applications, along with an introductory tutorial on antennas and techniques for antenna testing and tuning.

1.4.3 Data coding for radio transmission

Data streams must be encoded to add the characteristics needed for efficient radio transmission. As a minimum, encoding must make it possible to AC-couple the transmitted signal. This greatly simplifies the design of a radio system and helps to improve its performance. The encoding technique should also produce frequent transitions in the transmitted signal, which facilitates data clock synchronization and efficient data recovery at the receiver.

Radio transmissions must be bandwidth limited to control the signal-to-noise ratio observed at the receiver, as the noise power added during a radio transmission is proportional to the receiver bandwidth. The bandwidth required to transmit a data stream depends both on its data rate and how it has been encoded. Figure 1.4.3.1 shows three encoding schemes for single bits. Note that although the data rate is the same in each case,

Bit Coding and Receiver Bandwidth

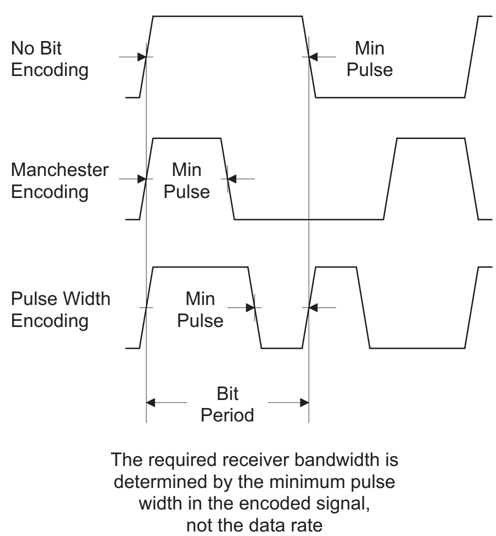


Figure 1.4.3.1

the minimum pulse width in the encoded signals vary 3:1. *The minimum bandwidth that can be used in the receiver depends on the minimum pulse (or gap) width in the encoded data stream, not the data rate.* It should be noted that encoding does not have to be done at the bit level, it can be done over a range of bits, such as a byte. Bit level encoding can usually be considered a modulation technique. Encoding over a range of bits is frequently referred to as symbolization.

The performance of a radio system depends on how well the data encoding scheme conditions the signal for AC-coupling. The encoding scheme should achieve DC-balance, which means that the encoded signal has a "1" value 50% of the time, and a "0" value 50% of the time. The encoding scheme should also limit the run length, or for how many bit periods the encoded signal remains at a "1" (or a "0") value. *The run length determines the maximum pulse (or gap) width that can occur in the transmitted signal.*

As shown in Figure 1.4.3.2, the way the receiver processes the transmitted signal depends on the minimum and maximum width of the pulses or gaps in the signal, *not the underlying encoded data rate*.

Receiver Signal Processing

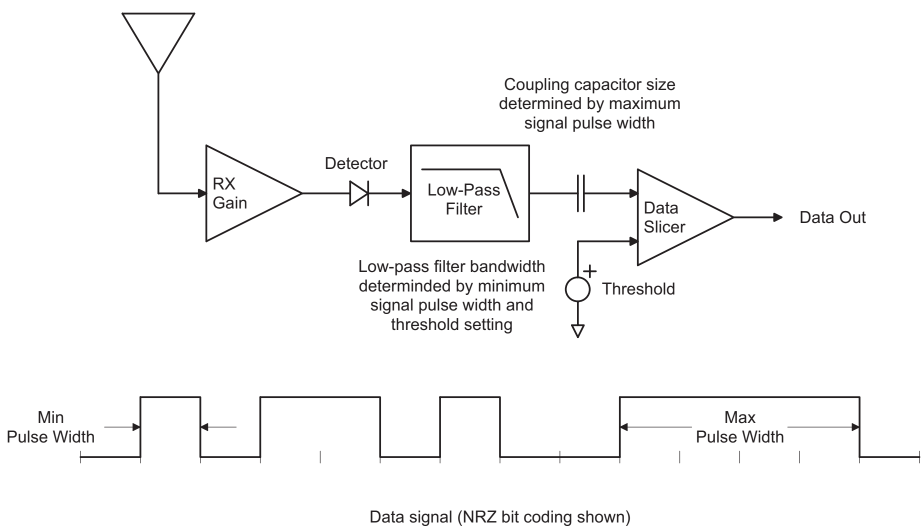


Figure 1.4.3.2

The ASH transceiver is AC-coupled between the receiver base-band output (Pin 5) and the comparator input (Pin 6). For this reason, the data bit stream being received should be encoded or modulated for good DC-balance, as explained above.

DC-balance can be accomplished a number of ways. Two of the most popular techniques for achieving DC-balance are Manchester encoding and symbol conversion. Manchester encoding is accomplished by encoding a "1" bit as a "1" + "0" signal pulse sequence, and encoding a "0" bit as a "0" + "1" pulse sequence. From another point of view, Manchester encoding is a form of BPSK modulation. This encoding scheme is very robust, but doubles the number of data bits that must be transmitted to send a message.

Another popular choice is byte to 12-bit symbol conversion, where each byte of a message is encoded as 12 bits, always with six "1" bits and six "0" bits. This encoding scheme is almost as robust as Manchester encoding, but only increases the number of data bits that must be transmitted to send a message by 50%. Refer to the program DC_BAL.BAS in the Appendix for an example of "byte to 12-bit conversion". In this example, conversion is done by mapping between nibbles (4 bits) and 6-bit half-symbols, using a lookup table.

Closely related to the need for DC-balance is the need to limit the number of "1" pulses or "0" pulses that occur together (run length), or in high concentration, in the transmitted signal. Note that Manchester encoding does an excellent job, limiting the run length to just two encoded bits. Using the byte to 12-bit symbol conversion technique shown in the Appendix, the run length is limited to 4 bits, which is also satisfactory.

Scrambling algorithms are also used on occasion to encode transmitted data. The advantage of scrambling is that there is no increase in the number of bits transmitted to send a message.

Scrambling does ensure frequent bit transitions and average DC-balance. However, scrambling does not control run length and bit concentration very well. This limits its use as an encoding scheme to applications where data rate is more important than transmission range.

As mentioned above, the reason that data is encoded to provide DC-balance and to control bit concentration and run length is receiver performance. Data encoding provides for maximum noise rejection. DC-balance charges the capacitor between Pin 5 and Pin 6 on the ASH transceiver to a value that makes the comparator "slice" the signal at a voltage halfway between the average value for a "1" and a "0". This means that the encoded data will be recovered error free so long as the noise level is less than one-half the voltage value between a "1" and a "0" pulse. When a received signal is unbalanced and a strong bias toward a "1" or a "0" value develops, noise rejection is severely reduced.

The value of the capacitor between Pin 5 and Pin 6 must be "tuned" for best receiver performance. It is desirable that this capacitor value not be too large, so that it quickly charges to the correct DC value for best noise performance when it starts receiving a transmission. On the other hand, it has to be large enough to pass the maximum signal run length without developing a strong bias in its slicing level. Thus, the optimum capacitor value depends on the message encoding scheme. Section 2.6.1 below discusses the specifics of selecting the base-band coupling capacitor for the ASH transceiver.

1.4.4 Packet communication protocols

All radio channels are subject to noise, interference and fading. In many cases, radio channels are shared by several users or services. Packet communication protocols are widely used to achieve error-free communications over imperfect and/or shared communication channels. Communication systems that use packet protocols include:

- The Internet
- Local area networks
- PC telephone modems
- Spread spectrum radios and wireless LANs
- Digital cellular phones and cellular modems (CDPD, etc.)

Almost all short-range wireless data communications use some form of packet protocol to automatically assure information is received correctly at the correct destination. A packet is a data structure that generally includes a training preamble, a start symbol, routing information (to/from, etc.) a packet ID, all or part of a message and error detection bits. Other information may be included depending on the protocol.

Figure 1.4.4.1 shows one of the packet formats used in RFM's Virtual Wire® Development Kits. The structure begins with a training preamble, which improves weak signal detection at the receiver by "training" the data slicer for best noise immunity, and providing signal transitions to train the clock recovery process. The training preamble usually consists of several bytes of a 1-0-1-0-1-0 … sequence. The length of the preamble needed depends on the receiver base-band coupling time constant, t_{BBC}. The time constant, in turn, depends on the data coding scheme used, as discussed in section 1.4.3 above, and section 2.6.1 below. A typical preamble is three-four bytes long.

General Virtual Wire RF Link Packet Format

Preamble	Start Symbol	To Byte	From Byte	Packet Number	Size/Status Byte*	Message	FCS High Byte	FCS Low Byte

General Virtual Wire Computer Link Packet Format

To Byte	From Byte	Packet Number	Size/Status Byte*	Message

Figure 1.4.4.1

The preamble is followed by a start symbol (often called a start vector), which is a distinct pattern of bits marking the start of the information section of the packet. The longer the start symbol, the lower the probability that a random noise pattern will match the start symbol and trigger a false packet reception. A 12 to 16 bit start symbol provides reasonable discrimination.

The start symbol is followed by “to” and “from” address information. RFM uses 4 bit and 8 bit “to” and “from” addresses in its protocols. It is a common practice to reserve one address for broadcasting to all nodes in a packet system. If a very large number of unique addresses are needed, 48 or more address bits may be used. The packet (ID) number allows specific packets to be identified and their error-free reception to be acknowledged. The packet ID number also makes it possible to assemble a multi-packet message when the packets are received out of sequence. In the RFM protocol, the packet ID is followed by message size or status information.

The message then follows. The last two bytes of the packet comprise a 16 bit error checking code (frame check sequence), based on the X.25 packet standard (ISO 3309). The error checking code is recomputed at the destination to confirm error-free detection. The ISO 3309 frame check sequence provides very high confidence of error detection for packets up to 256 bytes in length.

In summary, RFM Virtual Wire® protocols provides the following features:

- 16-bit ISO 3309 error detection calculation to test message integrity
- To/from address routing with programmable node addresses
- ASCII or binary message support
- Automatic packet retransmission until an acknowledgment is received; up to 8 retries with semi-random back off plus “acknowledge” and “link failure” alarm messages.

Each byte transmitted by the radio is converted into a 12 bit, DC-balanced symbol. DC-balance promotes good noise immunity by keeping the data slicer threshold set half way between a “1” and “0” value. The DC-balanced symbols used have no more than 4 bits of the same value in a row. This limited “run length” allows the receiver data slicer to be tuned to recover quickly from a heavy noise burst or strong interfering signal.

Further information on data encoding and packet protocols, plus a discussion of software techniques for clock and data recovery can be found in the ASH Transceiver *Software* Designer’s Guide. The Software Guide includes tutorial source code examples. Also, no-cost source code licenses are available from RFM for several versions of the Virtual Wire® data link layer protocol. Contact RFM’s application engineering group for additional information.

1.4.5 Noise control

Short-range wireless systems are especially sensitive to RF noise in the passband of the receiver, because the desired signals are transmitted at relatively low power levels. Commonly encountered internal noise sources include microcontrollers (for both control functions and data functions), brush-type motors and high-speed logic circuits. If the rise times and/or fall times of the clocking in a microcontroller are fast enough to produce harmonics in the frequency range of the receiver input and a harmonics fall directly within the passband of the receiver, special care must be taken to reduce the level of the harmonic at the antenna port of the receiver. If you have the option, choose a microprocessor with the slowest rise and fall time you can use for the application to minimize the generation of harmonics in the UHF band.

If possible, brush-type motors should be avoided in your application, since arcing of the brushes on the commutator makes a very effective spark-gap transmitter. If it is necessary to use a brush-type motor, spark suppression techniques must be used. Brush motors can often be purchased with spark suppression built-in. If the motor does not have built-in spark suppression, bypass capacitors, series resistors and shielding can be employed.

High-speed logic circuits produce noise similar to microprocessors. Once again, you should use logic with the slowest rise and fall times that will work in your application.

The items listed below should be considered for an application that has one or more of the above noise sources included. It may not be possible to follow all of these guidelines in a particular application.

- Locate the RF transceiver and its antenna as far from the noise source as possible.
- If the transceiver must be enclosed with the noise source, locate the antenna remotely using a coaxial cable.
- Terminate high-speed logic circuits with their characteristic impedance and use microstrip interconnect lines designed for that impedance.
- Keep line lengths at a minimum that carry high-speed logic signals or supply brush-type motors. Such lines are antennas that radiate the unwanted noise.
- If possible, enclose the noise source in a grounded metal box and use RF decoupling on the input/output lines.
- Avoid using the same power lines for the RF transceiver and the noise source or at least thoroughly filter (RF decouple) the power lines. It is advisable to use separate voltage regulators, if possible.
- If the antenna cannot be remotely located, place it as far from the noise source as possible (on the opposite end of the PC board). Orient the antenna such that its axis is in the same plane with the PC board containing the noise source. Do not run wires that supply the noise source in close proximity to the antenna.

Microcontroller clock frequency selection - you should check the computer or microcontroller clocks being used in your system to be sure they are not at or near a subharmonic of the receiver operating frequency. (For example, a 30.55 MHz clock would be the 30th subharmonic of 916.5 MHz.) It can be very difficult to suppress RF noise that is a harmonic of a clock being used in a digital system (especially odd harmonics). It is far better to chose a clock frequency that avoids this problem in the beginning.

Many microprocessors and microcontrollers "count down" the clock internally by factors such as 4, 8, etc. If this is the case with the processor you are using, confirm that the "count down" frequency is also not at or near a subharmonic of your RF input frequency.

1.4.6 Regulatory certification

Worldwide, man-made electromagnetic (radio) emissions are controlled by international treaty and the ITU (International Telecommunications Union) committee recommendations. These treaties require countries within a geographical region to use comparable tables for channel allocations and emission limits, to assure that all users can operate with minimum levels of interference.

Recognizing a need to protect their limited frequency resources, most countries have additional local laws, regulations and government decrees for acceptable emission levels from various types of electronic equipment, both military and commercial. By requiring that each model of equipment be tested and an authorization permit issued after the payment of a fee, governmental bodies prevent the sale of poor quality equipment and also create a record of equipment manufacturers.

Technical regulations and enforcement criteria vary from location to location. The USA, Canada and most European countries have adopted ITU tables for their respective radio regions. Australia, Hong Kong and Japan also have extensive rules and regulations for short-range transmitters and receivers, but with significant differences in the tables for their geographic regions. Most other countries have a set of less formal regulations, often modeled on either USA or EU regulations.

In any country, it is important to contact the Ministry of Telecommunications or Postal Services to determine the local allocations, regulations and required certifications prior to marketing your product there. The mildest penalty is often total loss of your import, export and foreign exchange privileges.

These laws and requirements are applicable to a finished product in the configuration that it will be sold the general public or the end user. OEM components often can not be certified, since they require additional non-standard attachments before they have any functional purpose.

Unless otherwise marked, RFM modules (such as development kits) have not been certified to any particular set of regulations. Each module has suggested countries for use, depending on current allocations and technical limits.

Product certification - general requirements for emissions and ingressions (called electromagnetic susceptibility) are controlled by engineering standards of performance, regulations, and the customer's expectations.

In USA and Canada, for example, you must formally measure your product's emissions, file for and receive a certification or authorization, and affix a permanent marking label to every device prior to retail sale. Regulations allow you to build a small number of products (usually 5 pieces) for testing and in-company use before certification and marketing. Trade shows and product announcements can be a problem for marketing, when the products are advertised without proper disclaimers. With Internet access, go to "www.fcc.org" for USA information or "www.ic.gc.ca" for Canada. The Canadian rules are RCC-210, Revision 2. FCC CFR 47, Parts 2 and 15, contains the needed information for USA sales.

European Union (EU) requirements allow self-certification of some systems but require formal measurement reports for other systems. In all cases, however, the directives demand that a "CE mark" be added to all compliant devices before they can be freely shipped in commerce. In the EU, the EMC Directive also adds various tests and expectations for levels of signal that will permit acceptable operation.

In April of 2000, the Radio Equipment and Telecommunications Terminal Equipment (R&TTE) Directive was issued that greatly simplifies short-range radio certification requirements in Europe. The R&TTE requires manufacturers to take full responsibility for the conformance of their equipment, but it also greatly streamlines the certification process.

A good general discussion of the introduction of the R&TTE Directive is available on the web site of the UK Radiocommunications Agency. The link to this discussion is:

http://www.radio.gov.uk/document/misc/rtte/rtteman/rtteman.htm

Additional information can be found on the European Radiocommunications Office (ERO) web site at:

http://www.ero.dk

RFM recommends you check these sites frequently as some additional changes to the ETSI short-range device specifications and EMC specifications are expected in the near future.

Certification testing

The emissions are measured in a calibrated environment defined by the regulations. USA and Canada use an "open field" range with 3 meters between the device under test (DUT) and the antenna. The range is calibrated by measurement of known signal sources to generate range attenuation (correction) curves in accordance with ANSI C63.4-1992.

EU measurement rules are based on a similar arrangement, but a "standard dipole" antenna is substituted for the DUT to calibrate the range attenuation. Since the EU measurements are comparison or substitution rules, they are often easier to follow for informal pre-testing by the designer. ETSI-300-220 has drawings to completely describe a typical test configuration.

The USA and Canadian requirements are contained in ANSI C63.4-1992, including a step-by-step test calibration and measurement procedure. Since these rules include range attenuation factors, one must make twice the measurements of the EU test method. Other countries follow one of these two techniques, with exception for a 10 meter range (separation) measurement or a different group of test frequencies.

Each of the listed contacts will have resources to provide current regulations and certification forms. They can also suggest sources for your formal tests, either commercial labs or the government testing office. Unless you want to invest in a qualified radiated signals test range, the commercial labs can help you with preliminary measurements and some expertise in correcting any difficulties that are noted.

Contacts for further information and current test facilities listings:

ANSI
Institute of Electrical & Electronics Engineers,
345 East 47th Street, New York, NY 10017 USA
http://www.ansi.org

ETSI
European Telecommunications Standard Institute
F-06921 Sophia Antipolis Cedex   FRANCE
http://www.etsi.fr

FCC
Federal Communications Commission
Washington DC 20554 USA
http://www.fcc.gov

Canada DOC
Industrie Canada
Attn: Certification, Engineering and Operations Section,
1241 Clyde Avenue, Ottawa K1A 0C8 CANADA
http://info.ic.gc.ca

UNITED KINGDOM
Radiocommunications Agency

Waterloo Bridge House, Waterloo Road
London SE1 8UA
http://www.open.gov.uk/radiocom

JATE
Japan Approvals Institute (JATE)
Isomura Bldg, 1-1-3 Toranomon
Minato-ku Tokyo JAPAN
http://www.mpt.go.jp

Please refer to RFM's web site at http://www.rfm.com for additional information on regulatory agencies.

2 ASH Transceiver Set-Up

2.1 Theory of Operation

The ASH transceiver's unique feature set is made possible by its system architecture. The heart of the transceiver is the amplifier-sequenced receiver section, which provides more than 100 dB of stable RF and detector gain without any special shielding or decoupling provisions. Stability is achieved by distributing the total RF gain over *time*. This is in contrast to a superheterodyne receiver, which achieves stability by distributing total RF gain over multiple frequencies.

ASH Receiver Block Diagram & Timing Cycle

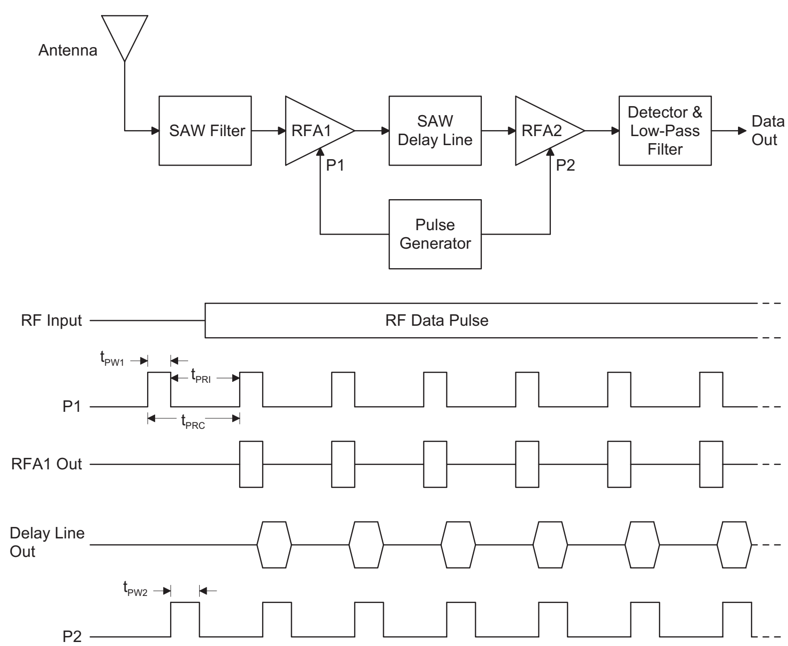


Figure 2.1.1

Figure 2.1.1 shows the basic block diagram and timing cycle for an amplifier-sequenced receiver. Note that the bias to RF amplifiers RFA1 and RFA2 are independently controlled by a pulse generator, and that the two amplifiers are coupled by a surface acoustic wave (SAW) delay line, which has a typical delay of 0.5 µs.

An incoming RF signal is first filtered by a narrow-band SAW filter, and is then applied to RFA1. The pulse generator turns RFA1 ON for 0.5 µs. The amplified signal from RFA1 emerges from the SAW delay line at the input to RFA2. RFA1 is now switched OFF and RFA2 is switched ON for 0.55 µs, amplifying the RF signal further. The ON time for RFA2 is usually set at 1.1 times the ON time for RFA1, as the filtering effect of the SAW delay line stretches the signal pulse from RFA1 somewhat. As shown in the timing diagram, RFA1 and RFA2 are never on at the same time, assuring excellent receiver stability. Note that the narrow-band SAW filter eliminates sampling sideband responses outside of the receiver passband, and the SAW filter and delay line act together to provide very high receiver ultimate rejection.

Amplifier-sequenced receiver operation has several interesting characteristics that can be exploited in system design. The RF amplifiers in an amplifier-sequenced receiver can be turned on and off almost instantly, allowing for very quick power-down (sleep) and wake-up times. Also, both RF amplifiers can be off between ON sequences to trade-off receiver noise figure for lower average current consumption. The effect on noise figure can be modeled as if RFA1 is on continuously, with an attenuator placed in front of it with a loss equivalent to $10*\log_{10}$(RFA1 duty factor), where the duty factor is the average amount of time RFA1 is ON (up to 50%). Since an amplifier-sequenced receiver is inherently a sampling receiver, the overall cycle time between the start of one RFA1 ON sequence and the start of the next RFA1 ON sequence should be set to sample the narrowest RF data pulse at least 10 times. Otherwise, significant edge jitter will be added to the detected data pulse.

Figure 2.1.2 is the overall block diagram of the ASH transceiver, and Figure 2.1.3 is the pin-out diagram. Please refer to these figures for the following discussions.

Antenna port - The only external RF components needed for the transceiver are the antenna and its matching components. Antennas presenting an impedance in the range of 35 to 72 ohms resistive can be satisfactorily matched to the RFIO pin with a series matching coil and a shunt matching/ESD protection coil. Other antenna impedances can be matched using two or three components. For some impedances, two inductors and a capacitor will be required. A DC path from RFIO to ground is required for ESD protection.

Receiver chain - the SAW RF filter has a nominal insertion loss of 3.5 dB, a 3 dB band-width of 600 kHz, and an ultimate rejection of 55 dB. The output of the SAW filter drives amplifier RFA1. This amplifier includes provisions for detecting the onset of saturation (AGC Set), and for switching between 35 dB of gain and 5 dB of gain (Gain Select). AGC Set is an input to the AGC Control function, and Gain Select is the AGC Control function output. ON/OFF control to RFA1 (and RFA2) is generated by the Pulse Generator & RF Amp Bias function. The output of RFA1 drives the low-loss SAW delay

ASH Transceiver Block Diagram

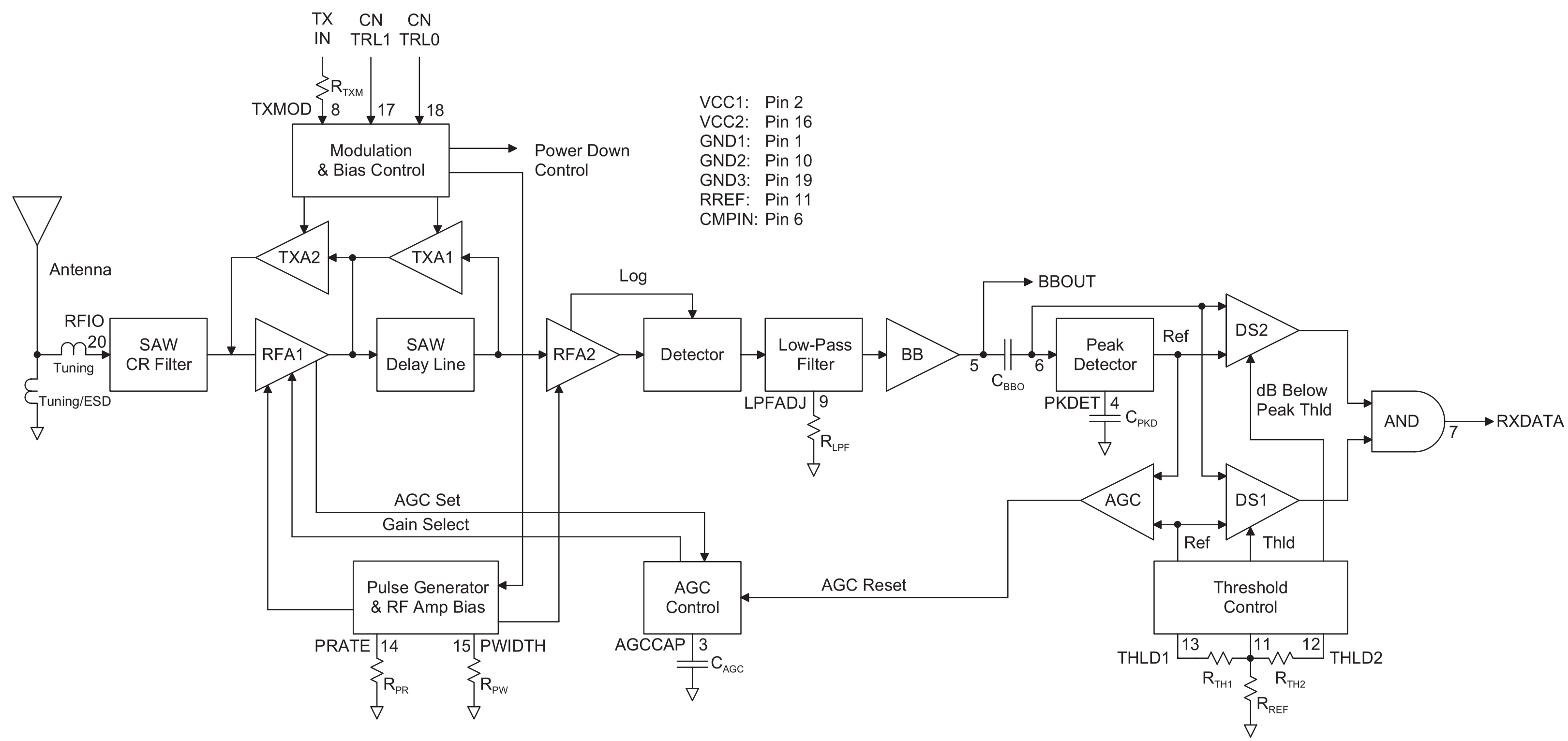


Figure 2.1.2

ASH Transceiver Pin Out

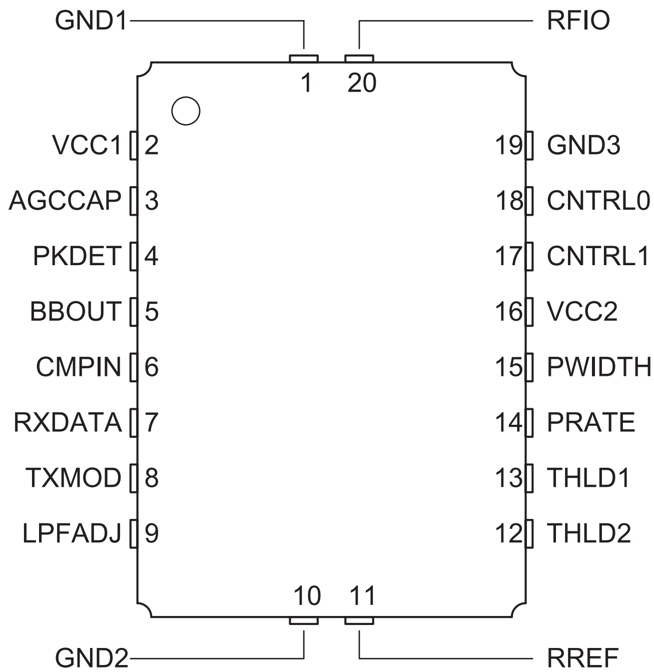


Figure 2.1.3

line, which has a nominal delay of 0.5 µs, an insertion loss of 6 dB, and an ultimate rejection of 50 dB. Note that the combined out-of-band rejection of the SAW RF filter and SAW delay line provides excellent receiver ultimate rejection.

The second amplifier, RFA2, provides 51 dB of gain below saturation. The output of RFA2 drives a full-wave (rectifier) detector with 19 dB of threshold gain. The onset of saturation in each section of RFA2 is detected and summed to provide a logarithmic response. This is added to the output of the full-wave detector to produce an overall detector response that is square law for low signal levels, and transitions into a log response for high signal levels. This combination provides excellent threshold sensitivity and more than 70 dB of detector dynamic range. In combination with the 30 dB of AGC range in RFA1, more than 100 dB of receiver dynamic range is achieved.

The detector output drives a gyrator filter. The filter provides a three-pole, 0.05 degree equiripple low-pass response with excellent group delay flatness and minimal pulse ringing. The 3 dB bandwidth of the filter can be set from 4.5 kHz to 1.8 MHz with an external resistor.

The filter is followed by a base-band amplifier which boosts the detected signal to the BBOUT pin. When the receiver RF amplifiers are operating at a 50%-50% duty cycle, the BBOUT signal changes about 10 mV/dB, with a peak-to-peak signal level of up to 685 mV. For lower duty cycles, the mV/dB slope and peak-to-peak signal level are proportionately less. The detected signal is riding on a 1.1 Vdc level that varies somewhat with supply voltage, temperature, etc. BBOUT is coupled to the CMPIN pin or to an external data recovery process (DSP, etc.) by a series capacitor.

When an external data recovery process is used with AGC, BBOUT must be coupled to the external data recovery process and CMPIN by separate series coupling capacitors. The AGC reset function is driven by the signal applied to CMPIN.

When the transceiver is placed in power-down or in a transmit mode, the output impedance of BBOUT becomes very high. This feature helps preserve the charge on the coupling capacitor to minimize data slicer stabilization time when the transceiver switches back to the receive mode.

Data Slicers - The CMPIN pin drives two data slicers, which convert the analog signal from BBOUT back into a digital stream. The best data slicer choice depends on the system operating parameters. Data slicer DS1 is a capacitor-coupled comparator with provisions for an adjustable threshold. DS1 provides the best performance at low signal-to-noise conditions. The threshold, or squelch, offsets the comparator's slicing level from 0 to 90 mV, and is set with a resistor between the RREF and THLD1 pins. This threshold allows a trade-off between receiver sensitivity and output noise density in the no-signal condition.

DS2 is a "dB-below-peak" slicer. The peak detector charges rapidly to the peak value of each data pulse, and decays slowly in between data pulses (1:1000 ratio). The DS2 slicer trip point can be set from 0 to 120 mV below this peak value with a resistor between RREF and THLD2. A threshold of 60 mV is the most common setting, which equates to "6 dB below peak" when RFA1 and RFA2 are running a 50%-50% duty cycle. DS2 is best for ASK modulation where the transmitted waveform has been shaped to minimize signal bandwidth. However, DS2 can be temporarily "blinded" by strong noise pulses, which causes burst data errors. Note that DS1 is active when DS2 is used, as RXDATA is the logical AND of the DS1 and DS2 outputs. DS1 and DS2 must both be high to generate a high RXDATA output. DS2 can be disabled by leaving THLD2 disconnected.

AGC Control - The output of the Peak Detector also provides an AGC Reset signal to the AGC Control function through the AGC comparator. The purpose of the AGC function is to extend the dynamic range of the receiver, so that two transceivers can operate close together when running ASK and/or high data rate modulation. The AGC also prevents receiver saturation by a strong in-band interfering signal, allowing operation to continue at short range in the presence of the interference. The onset of saturation in the output stage of RFA1 is detected and generates the AGC Set signal to the AGC Control function. The AGC Control function then selects the 5 dB gain mode for RFA1. The AGC Comparator will send a reset signal when the Peak Detector output (multiplied by 0.8) falls below the threshold voltage for DS1 (note that the DS1 threshold must be greater than zero for correct AGC operation).

A capacitor at the AGCCAP pin avoids AGC “chattering” during the time it takes for the signal to propagate through the low-pass filter and charge the peak detector. The AGC capacitor also allows the AGC hold-in time to be set longer than the peak detector decay time to avoid AGC chattering during runs of “0” bits in the received data stream.

Note that AGC operation requires the peak detector to be functioning, even if DS2 is not being used. AGC operation can be defeated by connecting the AGCCAP pin to Vcc. The AGC can be latched on once engaged by connecting a 150 kilohm resistor between the AGCCAP pin and ground in lieu of a capacitor.

Receiver pulse generator and RF amplifier bias - The receiver amplifier-sequence operation is controlled by the Pulse Generator & RF Amplifier Bias module, which in turn is controlled by the PRATE and PWIDTH input pins, and the Power Down Control Signal from the Modulation & Bias Control function.

In the low data rate mode, the interval between the falling edge of one RFA1 ON pulse to the rising edge of the next RFA1 ON pulse t_{PRI} is set by a resistor between the PRATE pin and ground. The interval can be adjusted between 0.1 and 5 µs. In the high data rate mode (selected at the PWIDTH pin) the receiver RF amplifiers operate at a nominal 50%-50% duty cycle. In this case, the start-to-start period t_{PRC} for ON pulses to RFA1 are controlled by the PRATE resistor over a range of 0.1 to 1.1 µs.

In the low data rate mode, the PWIDTH pin sets the width of the ON pulse t_{PW1} to RFA1 with a resistor to ground (the ON pulse width t_{PW2} to RFA2 is set at 1.1 times the pulse width to RFA1 in the low data rate mode). The ON pulse width t_{PW1} can be adjusted between 0.55 and 1 µs. However, when the PWIDTH pin is connected to Vcc through a 1 M resistor, the RF amplifiers operate at a nominal 50%-50% duty cycle, facilitating high data rate operation. In this case, the RF amplifiers are controlled by the PRATE resistor as described above.

Both receiver RF amplifiers are turned off by the Power Down Control Signal, which is invoked in the power-down and transmit modes.

Transmitter chain - the transmitter chain consists of a SAW delay line oscillator followed by a modulated buffer amplifier. The SAW filter suppresses transmitter harmonics to the antenna. Note that the same SAW devices used in the amplifier-sequenced receiver are reused in the transmit modes.

Transmitter operation supports two modulation formats, on-off keyed (OOK) modulation, and amplitude-shift keyed (ASK) modulation. When OOK modulation is chosen, the transmitter output turns completely off between “1” data pulses. When ASK modulation is chosen, a “1” pulse is represented by a higher transmitted power level, and a “0” is represented by a lower transmitted power level. OOK modulation provides compatibility with first-generation ASH technology, and provides for power conservation. ASK modulation must be used for high data rates (data pulses less than 30 µs). ASK modulation also reduces the effects of some types of interference and allows the transmitted pulses to be shaped to control modulation bandwidth.

The modulation format is chosen by the state of the CNTRL0 and the CNTRL1 mode control pins, as discussed below. When either modulation format is chosen, the receiver RF amplifiers are turned off. In the OOK mode, the delay line oscillator amplifier TXA1 and the output buffer amplifier TXA2 are turned off when the voltage to the TXMOD input falls below 220 mV. In the OOK mode, the data rate is limited by the turn-on and turn-off times of the delay line oscillator. In the ASK mode TXA1 is biased ON continuously, and the output of TXA2 is modulated by the TXMOD input current.

The transmitter RF output power is proportional to the input current to the TXMOD pin. A resistor in series with the TXMOD pin is used to adjust the peak transmitter output power. Rated output power requires 250 to 450 µA of input current, depending on the frequency of operation.

The four transceiver operating modes - receive, transmit ASK, transmit OOK, and power-down ("sleep"), are controlled by the Modulation & Bias Control function, and are selected with the CNTRL1 and CNTRL0 control pins. Setting CNTRL1 and CNTRL0 both high place the unit in the receive mode. Setting CNTRL1 high and CNTRL0 low place the unit in the ASK transmit mode. Setting CNTRL1 low and CNTRL0 high place the unit in the OOK transmit mode. Setting CNTRL1 and CNTRL0 both low place the unit in the power-down (sleep) mode. CNTRL1 and CNTRL0 are CMOS compatible inputs. These inputs must be held at a logic level; they cannot be left unconnected.

2.2 Power Supply Requirements

As shown in Figure 2.1.3, VCC1 (Pin 2) is the positive supply voltage pin for the transmitter output amplifier and the receiver base-band circuitry. Pin 2 is usually connected to the positive supply through a ferrite RF decoupling bead which is bypassed by an RF capacitor on the *supply side*. The ferrite bead eliminates the possibility of RF feedback from the antenna to Pin 2 and should be used except for specific "EMI robust" layouts. VCC2 (Pin 16) is the positive supply voltage pin for the receiver RF section and transmitter oscillator. Pin 16 must be bypassed with an RF capacitor, and must also be bypassed with a 1 to 10 µF tantalum or electrolytic capacitor. The power supply voltage range for standard operation is now characterized from 2.2 to 3.7 Vdc. Power supply ripple *must be less than 10 mV* peak-to-peak.

2.2.1. Low voltage set-up

Second-generation ASH radios were characterized for operation from 2.2 to 3.7 Vdc over the temperature range of -40 to +85 ^{o}C in the Summer of 2002 (see section 3.8 in the Appendix). Where transmitter output power stability is important over this extended voltage range, the TXMOD input (Pin 8) should be driven from a true current source rather than a voltage source through a relatively large resistor. When using a current source, an RF isolation resistor of at least 220 ohms should be used between Pin 8 and the current source.

2.3 RF Input/Output

Pin 20 (RFIO) is the RF input/output pin. This pin is connected directly to the input transducer of a high-Q quartz SAW filter, as shown in Figure 2.3.1. The antenna impedance must be transformed by a matching network to present a specific impedance Z_L to the SAW filter for proper operation. *Connecting Pin 20 directly to a 50 ohm antenna will result in poor performance.*

ASH Radio Antenna Interface Circuit Model

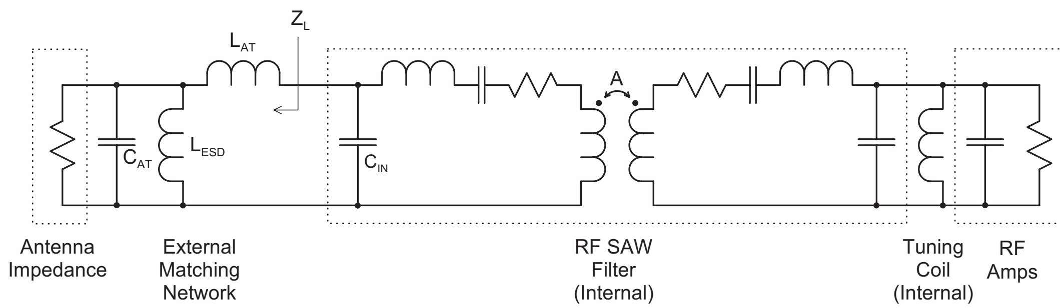


Figure 2.3.1

2.3.1 Antenna matching

Referring to Figure 2.3.1 again, the RF SAW filter can be modeled as a two-pole band-pass filter with acoustic coupling between the two sections of the filter. The right section of the filter is internally matched to the receiver input amplifier and transmitter power amplifier by a shunt tuning coil. When the proper impedance Z_L is presented to the left side of the filter, the correct RF filter response, receiver match for low noise figure, transmitter output match, and other RF parameters are automatically achieved.

Transforming a 50 ohm antenna impedance to the correct Z_L for an ASH radio can usually be accomplished using a series chip inductor and a shunt chip inductor on the antenna side (a shunt chip capacitor is also used for the TR1100 and TR3100.) The values for these impedance matching components are listed in Table 2.3.1.1. However, many applications will involve interfacing an ASH radio to an antenna whose impedance is not 50 ohms. To accomplish this task, first measure the input impedance of the antenna using a network analyzer. Next, determine the matching network to transform the antenna impedance to 50 ohms. Cascade this antenna matching network with the impedance transformation network from Table 2.3.1.1 to the get the overall matching network. Finally, combine component values where possible to simplify the overall matching network.

Let's consider an example. Assume we are working with a monopole antenna, either a simple length of wire or a copper trace on a PC board. If the length of the antenna is less than one-fourth of a wavelength at the frequency of interest, the network analyzer will indictate that the impedance is of the form, R - jX , or that the impedance is equivalent to a resistor in series with a capacitor. This antenna can be matched by using a series inductor whose reactance is equal to the reactance X of the capacitor. This results in a matched antenna impedance of R. For such an antenna, the value of R is usually some where between 35 and 72 ohms. This is close enough to 50 ohms to avoid significant impedance mismatch loss. Once the antenna matching inductance value has been determined, the overall matching network is developed by combining the antenna matching inductance value with the matching component values listed in Table 2.3.1.1. In most cases, this allows using two chip inductors to transform the antenna impedance to the required Z_L.

ASH Radio Matching Component Values for a 50 ohm Antenna						
Part Number	**Frequency**	**L_{AT} ***	**L_{ESD}**	**C_{AT}**	**Z_L**	**Y_L**
	MHz	**nH**	**nH**	**pF**	**ohms**	**mmho**
TR1100	916.50	18	100	6.8	13 + j83	1.8 - j11.8
TR1000, TX6000, RX6000	916.50	10	100	-	51 + j62	7.9 - j9.7
TR1004, TX6004, RX6004	914.00	10	100	-	51 + j62	7.9 - j9.7
TR1001, TX6001, RX6001	868.35	10	100	-	51 + j59	8.4 - j9.8
RX6501	868.35	10	100	-	51 + j59	8.4 - j9.8
TR3100	433.92	68	220	6.8	31 + j160	1.1 - j6.0
TR3000, TX5000, RX5000	433.92	56	220	-	53 + j157	1.9 - j5.7
RX5500	433.92	56	220	-	53 + j157	1.9 - j5.7
TR3002, TX5002, RX5002	418.00	56	220	-	52 + j151	2.0 - j5.9
TR3001, TX5001, RX5001	315.00	82	33	-	35 + j186	1.0 - j5.2
RX5501	315.00	82	33	-	35 + j186	1.0 - j5.2
TR3003, TX5003, RX5003	303.83	82	33	-	33 + j180	1.0 - j5.4
* Q of at least 50						

Table 2.3.1.1

Another example is matching a monopole antenna whose length is greater than one-fourth of a wavelength. The impedance of such an antenna will be of the form R + jX, indicating that the impedance is equivalent to a resistor in series with an inductor. This antenna can be matched by using a series capacitor whose reactance is equal to the reactance X of the inductor, resulting in a matched antenna impedance of R. Once again, the value of R will usually be close enough to 50 ohms to avoid significant mismatch loss. The negative reactance of this matching capacitor can then be combined with the reactances of the matching components listed in Table 2.3.1.1 to obtain the reactance of the matching components that will match the antenna to the ASH radio. Of course, if the resulting reactance is negative, the matching components will include a capacitor rather than an inductor.

For more information (plus examples) on developing matching networks starting with the component values for matching an ASH radio to a 50 ohm antenna, see the *ASH Transceiver Antenna Impedance Matching* paper in the application notes section of RFM's web site at www.rfm.com.

Table 2.3.1.1 also lists the Z_L impedance values and corresponding Y_L admittance values for all standard ASH radios. These values can be transferred directly to a Smith Chart or RF CAD package to support the design and evaluation of various antenna matching network topologies. Note that it is desirable to use a matching network topology that includes a series inductor L_{AT} in the matching network. L_{AT} and the RF SAW filter input capacitance C_{IN} (Figure 2.3.1) form a low-pass filter above the operating frequency of the ASH radio, providing additional high-side signal rejection. Also, a shunt inductor across the antenna must be present in the matching network for ESD protection, as discussed in section 2.3.2 below.

2.3.2 ESD protection

The SAW input transducer (Pin 20) is static sensitive and must be protected by a shunt RF choke to GND1 (Pin 1). The ESD choke may also function as part of the antenna tuning network as shown in Table 2.3.1.1. To provide further ESD protection, externally mounted antennas should have an insulating jacket. The ESD choke should have a very low series resistance (less than 0.1 ohm) to be fully effective.

2.4 Pulse Generator

The receiver amplifier-sequence operation is controlled by the Pulse Generator & RF Amplifier Bias module, which in turn is controlled by the PRATE and PWIDTH input pins, and the Power Down Control Signal from the Modulation & Bias Control function.

Pulse Generator Timing

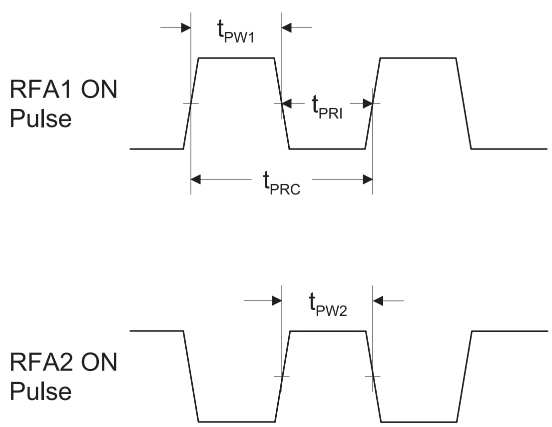


Figure 2.4.1.1

Both receiver RF amplifiers are turned off by the Power Down Control Signal, which is invoked in the power-down and transmit modes.

2.4.1 Pulse rate and pulse width

The pulse generator timing terminology is shown in Figure 2.4.1.1. The pulse generator has two operating modes; one for low data rate (low current) applications and one for high data rate (high sensitivity) applications. In the low data rate mode, the interval between the falling edge of one RFA1 ON pulse to the rising edge of the next RFA1 ON pulse t_{PRI} is set by a resistor between the PRATE pin and ground. The interval can be adjusted between 0.1 and 5 µs.

In the high data rate mode (selected at the PWIDTH pin) the receiver RF amplifiers operate at a nominal 50%-50% duty cycle. In this case, the start-to-start period t_{PRC} for ON pulses to RFA1 are controlled by the PRATE resistor over a range of 0.1 to 1.1 µs.

In the low data rate mode, the PWIDTH pin sets the width of the ON pulse t_{PW1} to RFA1 with a resistor to ground (the ON pulse width t_{PW2} to RFA2 is set at 1.1 times the pulse width to RFA1 in the low data rate mode). The ON pulse width t_{PW1} can be adjusted between 0.55 and 1 µs.

However, when the PWIDTH pin is connected to Vcc through a 1 M resistor, the RF amplifiers operate at a nominal 50%-50% duty cycle, facilitating high data rate operation. In this case, the RF amplifiers are controlled by the PRATE resistor as described above.

2.4.2 Low data rate set-up

The interval between the falling edge of an ON pulse to the first RF amplifier and the rising edge of the next ON pulse to the first RF amplifier t_{PRI} is set by resistor R_{PR} between Pin 14 and ground. The interval t_{PRI} can be adjusted between 0.1 and 5 µs with a resistor in the range of 51 K to 2000 K. The value of R_{PR} is given by:

$R_{PR} = 404 * t_{PRI} + 10.5$, where t_{PRI} is in µs, and R_{PR} is in kilohms

A ±5% resistor value is recommended. It is important to keep the total capacitance between ground, Vcc and this pin to less than 5 pF to maintain stability.

Pin 15 (PWIDTH) sets the width of the ON pulse to the first RF amplifier t_{PW1} with a resistor R_{PW} to ground (the ON pulse width to the second RF amplifier t_{PW2} is set at 1.1 times the pulse width to the first RF amplifier). The ON pulse width t_{PW1} can be adjusted between 0.55 and 1 µs with a resistor value in the range of 200 K to 390 K. The value of R_{PW} is given by:

$R_{PW} = 404 * t_{PW1} - 18.6$, where t_{PW1} is in µs and R_{PW} is in kilohms

A ±5% resistor value is recommended. It is important to keep the total capacitance between ground, Vcc and this node to less than 5 pF to maintain stability.

Testing has shown that setting t_{PW1} to 0.7 µs matches the SAW delay line pulse response characteristics for best sensitivity. In this case, the interval t_{PRI} is normally set between 0.77 µs and 2.5 µs. Setting t_{PRI} at 0.77 µs provides maximum sensitivity; 2.5 µs provides a 55% reduction in average RF amplifier current in trade-off for a 3.6 dB reduction in sensitivity. A t_{PRI} setting of 2.5 µs or less also assures a sequential-amplifier sampling rate of 333 ksps or more, providing at least 10 samples of the narrowest OOK pulse width of 30 µs. The low data rate set-up is recommended for signal pulse widths of 17.4 µs of greater. The high data rate set-up is recommended for signal pulse widths less than 17.4 µs.

2.4.3 High data rate set-up

When Pin 15 (PWIDTH) is connected to Vcc through a 1 M resistor, the RF amplifiers operate at a nominal 50%-50% duty cycle, facilitating high data rate operation. In this case, the period t_{PRC} from start-to-start of ON pulses to the first RF amplifier is controlled by the PRATE resistor (Pin 14) over a range of 0.1 to 1.1 µs using a resistor of 11 K to 220 K. In this case the value of R_{PR} is given by:

$R_{PR} = 198 * t_{PRC} - 8.51$, where t_{PRC} is in µs and R_{PR} is in kilohms

A ±5% resistor value should also be used in this case.

For minimum signal pulse widths between 8.7 and 17.4 µs, t_{PRC} should be set to 0.87 µs. This value provides a nominal sampling rate of 10 samples for an 8.7 µs signal pulse, and takes advantage of the pulse stretching through the SAW delay line to provide near-optimum RF gain.

2.5 Low-Pass Filter

The low-pass filter used in the ASH transceiver is a three-pole, 0.05 degree equiripple design which features excellent group delay flatness and minimal pulse ringing.

2.5.1 3 dB bandwidth adjustment

Pin 9 is the receiver low-pass filter bandwidth adjust. The filter bandwidth is set by a resistor R_{LPF} between this pin and ground. The resistor value can range from 330 kilohms to 820 ohms, providing a filter 3 dB bandwidth f_{LPF} from 4.4 kHz to 1.8 MHz. The resistor value is determined by:

$R_{LPF} = 1445/ f_{LPF}$, where R_{LPF} is in kilohms, and f_{LPF} is in kHz

A ±5% resistor should be used to set the filter bandwidth. This will provide a 3 dB filter bandwidth between f_{LPF} and $1.3 * f_{LPF}$ with variations in supply voltage, temperature, etc.

It should be noted that the peak drive current available from RXDATA increases in proportion to the filter bandwidth setting. R_{LPF} cannot be larger than 330 kilohms (4.4 kHz bandwidth). For low data rate operation a simple external R-C filter can be added at Pin 5 to further improve receiver sensitivity. See Figure 4.2 in the ASH Transceiver *Software* Designer's Guide for additional details.

2.5.2 Bandwidth selection

When using data slicer DS2 or data slicer DS1 with no threshold, the recommended 3 dB bandwidth of the filter for DC-balanced data (12-bit symbol or Manchester encoding) is:

$f_{LPF} = 750/SP_{MIN}$, where f_{LPF} is in kHz and minimum signal pulse width SP_{MIN} is in µs

The recommended 3 dB bandwidth when using DS1 (only) with a mild threshold is:

$f_{LPF} = 1000/SP_{MIN}$

The recommended 3 dB bandwidth when using DS1 (only) with a strong threshold is:

$f_{LPF} = 2500/SP_{MIN}$

2.6 Base-Band Coupling

Pin 5 is the receiver base-band output pin (BBOUT). This pin drives the CMPIN (Pin 6) through coupling capacitor C_{BBO} for internal data slicer operation. The time constant t_{BBC} for this connection is:

$t_{BBC} = 0.064 * C_{BBO}$, where t_{BBC} is in µs and C_{BBO} is in pF

A ±10% ceramic capacitor should be used between BBOUT and CMPIN. The time constant can vary between t_{BBC} and $1.8 * t_{BBC}$ with variations in supply voltage, temperature, etc. The optimum time constant in a given circumstance will depend on the data rate, data run length, and other factors as discussed in section 2.6.1 below.

When the transceiver is in power-down or in a transmit mode, the output impedance of Pin 5 becomes very high. This feature helps preserve the charge on the coupling capacitor to minimize data slicer stabilization time when the transceiver switches back to the receive mode.

2.6.1 Base-band coupling capacitor selection

The correct value of the base-band coupling capacitor depends on the maximum pulse width (or gap) that can occur in the signal. The maximum pulse width, in turn, depends on the data stream encoding, the data rate, and the maximum run length that occurs in the data. If no data stream encoding is used, the maximum pulse width is equal to a bit period multiplied by the maximum run length. If byte to 12-bit symbol encoding is used, the maximum pulse width is four encoded bit periods. For Manchester encoding, the maximum pulse width is two encoded bit periods.

Time constant t_{BBC} should be chosen so that the signal “droops” no more than 20% during a maximum pulse width event, or:

$t_{BBC} = 4.48 * SP_{MAX}$, where t_{BBC} and maximum signal pulse width SP_{MAX} are in µs

$C_{BBO} = 15.625 * t_{BBC}$, where t_{BBC} is in µs and C_{BBO} is in pF, or

$C_{BBO} = 70 * SP_{MAX}$, where SP_{MAX} is in µs and C_{BBO} is in pF

It takes a packet training preamble equal to 1.6 times t_{BBC} to train C_{BBO} to a voltage of 80% of its optimum slicing value. Using Manchester encoding, this equates to nominally two AA hex bytes. Using byte to 12-bit symbolization, this equates to four AA hex bytes. Attempting to transmit data with an SP_{MAX} of 16 bits or more requires an impracticably long training preamble. This is one reason that data encoding is important.

2.6.2 Base-band output signal levels

BBOUT can also be used to drive an external data recovery process (DSP, etc.). When the receiver RF amplifiers are operating at a 50%-50% duty cycle, the BBOUT signal changes about 10 mV/dB, with a peak-to-peak signal level of up to 685 mV. For lower duty cycles, the mV/dB slope and peak-to-peak signal level are proportionately less. The detected signal is riding on a 1.1 Vdc level that varies somewhat with supply voltage, temperature, etc. BBOUT is coupled to the CMPIN pin or to an external data recovery process by a series capacitor. The nominal output impedance of this pin is 1 K. A load impedance of 50 K to 500 K in parallel with no more than 10 pF is recommended.

When an external data recovery process is used with AGC, BBOUT must be coupled to the external data recovery process and CMPIN by separate series coupling capacitors. The AGC reset function is derived from the Peak Detector Circuit which is driven by the signal applied to CMPIN.

2.7 Data Slicers

CMPIN (Pin 6) drives two data slicers, which convert the analog signal from BBOUT back into a digital stream. The best data slicer choice depends on the system operating parameters. Data slicer DS1 is a capacitor-coupled comparator with provisions for an adjustable threshold. DS1 provides the best performance at low signal-to-noise conditions. The threshold, or squelch, offsets the comparator’s slicing level from 0 to 90 mV, and is set with a resistor between the RREF and THLD1 pins. This threshold allows a trade-off between receiver sensitivity and output noise density in the no-signal condition. For best sensitivity, the threshold is set to 0. In this case, noise is output continuously when no signal is present. This, in turn, requires the circuit being driven by the RXDATA pin to be able to process noise (and signals) continuously.

This can be a problem if RXDATA is driving a circuit that must "sleep" when data is not present to conserve power, or when it is necessary to minimize false interrupts to a multitasking processor. In this case, noise can be greatly reduced by increasing the threshold level, but at the expense of sensitivity. A threshold of 50 mV provides a good trade-off between excessive false interrupts and excessive loss of sensitivity for a filter bandwidth of 48 kHz (19.2 kbps NRZ data rate). If you are using a different filter bandwidth, start with a threshold value of:

$$V = 7.2*(f_{LPF})^{1/2} \text{ where V is in mV and } f_{LPF} \text{ is in kHz}$$

Thresholds of 60 to 90 mV may be required to suppress hash from some computers. Note that the best 3 dB bandwidth for the low-pass filter is affected by the threshold level setting of DS1, as discussed in section 2.5.2. *Also note that the AGC reset operation requires a non-zero threshold on DS1.*

Data slicer DS2 can substantially overcome the compromise between the DS1 threshold value and filter bandwidth once the signal level is high enough to enable its operation. DS2 is a "dB-below-peak" slicer. The peak detector charges rapidly to the peak value of each data pulse, and decays slowly in between data pulses (1:1000 ratio). The DS2 slicer trip point can be set from 0 to 120 mV below this peak value with a resistor between RREF and THLD2. A threshold of 60 mV is the most common setting, which equates to "6 dB below peak" when RFA1 and RFA2 are running a 50%-50% duty cycle. Slicing at the "6 dB-below-peak" point reduces the signal amplitude to data pulse-width variation, allowing a lower 3 dB filter bandwidth to be used for improved sensitivity.

DS2 is used with high data rate ASK modulation and/or to reject weak interference. However, DS2 can be temporarily "blinded" by strong noise pulses, which causes burst data errors. Note that DS1 is active when DS2 is used, as RXDATA is the logical AND of the DS1 and DS2 outputs. When DS2 is used, the DS1 threshold is usually set to less than 60 mV (25 mV typical). DS2 is disabled by leaving THLD2 disconnected.

2.7.1 Data slicer 1 threshold selection

RREF is the external reference resistor pin. A 100 K reference resistor is connected between this pin and ground. A ±1% resistor tolerance is recommended. It is important to keep the total capacitance between ground, Vcc and this node to less than 5 pF to maintain current source stability. If THLD1 and/or THDL2 are connected to RREF through resistor values less that 1.5 K, their node capacitance must be added to the RREF node capacitance and the total should not exceed 5 pF.

The THLD1 pin sets the threshold for the standard data slicer through a resistor R_{TH1} to RREF. The threshold is increased by increasing the value of the resistor. Connecting this pin directly to RREF provides zero threshold. The value of the resistor depends on whether THLD2 is used. For the case that THLD2 is not used, the acceptable range for the resistor is 0 to 100 K, providing a THLD1 range of 0 to 90 mV. The resistor value is given by:

R_{TH1} = 1.11*V, where R_{TH1} is in kilohms and the threshold V is in mV

For the case that THLD2 is used, the acceptable range for the THLD1 resistor is 0 to 200 K, again providing a THLD1 range of 0 to 90 mV. The resistor value is given by:

R_{TH1} = 2.22*V, where R_{TH1} is in kilohms and the threshold V is in mV

A ±1% resistor tolerance is recommended for the THLD1 resistor.

2.7.2 Data slicer 2 enable and threshold

The operation of data slicer 2 and the AGC depend on the peak detector circuit. Pin 4 controls the peak detector operation. A capacitor between this pin and ground sets the peak detector attack and decay times, which have a fixed 1:1000 ratio. For most applications, these time constants should be coordinated with the base-band time constant. For a given base-band capacitor C_{BBO}, the capacitor value C_{PKD} is:

C_{PKD} = 0.33*C_{BBO} , where C_{PKD} and C_{BBO} are in pF

A ±10% ceramic capacitor should be used at this pin. This time constant will vary between 1:1 and 1.5:1 with variations in supply voltage, temperature, etc. The capacitor is driven from a 200 ohm “attack” source, and decays through a 200 K load. The peak detector is used to drive the “dB-below-peak” data slicer and the AGC release function. The AGC hold-in time can be extended beyond the peak detector decay time with the AGC capacitor, as discussed in section 2.8.1. Where low data rates and OOK modulation are used, the “dB-below-peak” data slicer and the AGC are optional. In this case, the PKDET pin and the THLD2 pin can be left unconnected, and the AGC pin can be connected to Vcc to reduce the number of external components needed. The peak detector capacitor is discharged in the receiver power-down mode and in the transmit modes.

THLD2 is the “dB-below-peak” data slicer threshold adjust pin. The threshold is set by a 0 to 200 K resistor R_{TH2} between this pin and RREF. Increasing the value of the resistor decreases the threshold below the peak detector value (increases difference) from 0 to 120 mV. For most applications, this threshold should be set at 6 dB below peak, or 60 mV for a 50%-50% RF amplifier duty cycle. The value of the THLD2 resistor is given by:

R_{TH2} = 1.67*V, where R_{TH2} is in kilohms and the threshold V is in mV.

A ±1% resistor tolerance is recommended for the THLD2 resistor. Leaving the THLD2 pin open disables the dB-below-peak data slicer operation.

2.8 AGC

The purpose of the AGC function is to extend the dynamic range of the receiver, so that two transceivers can operate close together when running ASK and/or high data rate modulation. The AGC also allows limited-range operation when using either ASK or OOK modulation in the presence of strong interference that would otherwise saturate the receiver. If operating distances are always short, the AGC can be latched on to deliberately limit operating range and reduce susceptibility to interference, as described in section 2.8.2.

The AGC circuit operates as follows. The output of the Peak Detector provides an AGC Reset signal to the AGC Control function through the AGC comparator. The onset of saturation in the output stage of RFA1 is detected and generates the AGC Set signal to the AGC Control function. The AGC Control function then selects the 5 dB gain mode for the first RX amplifier. The AGC Comparator will send a reset signal when the Peak Detector output (multiplied by 0.8) falls below the threshold voltage for DS1 (the DS1 threshold must be greater than zero for proper AGC operation). A capacitor at the AGCCAP input (Pin 3) stabilizes the AGC "set" operation, and allows the AGC hold-in time to be set longer than the peak detector decay time. This feature can be used to avoid AGC chattering during runs of "0" bits in the received data stream. Note that AGC operation requires the peak detector to be functioning, even if DS2 is not being used.

2.8.1 Hold-in capacitor

As discussed, Pin 3 controls the AGC set and reset operations. A capacitor between this pin and ground sets the minimum time the AGC will hold-in once it is engaged. The hold-in time is set to avoid AGC chattering. For a given hold-in time t_{AGH}, the capacitor value C_{AGC} is:

$$C_{AGC} = 19.1 * t_{AGH}, \text{ where } t_{AGH} \text{ is in } \mu s \text{ and } C_{AGC} \text{ is in pF}$$

A ±10% ceramic capacitor should be used at this pin. The value of C_{AGC} given above provides a hold-in time between t_{AGH} and $2.65 * t_{AGH}$, depending on operating voltage, temperature, etc. The hold-in time is chosen to allow the AGC to ride through the longest run of zero bits that can occur in a received data stream. The AGC hold-in time can be greater than the peak detector decay time, as discussed above. However, the AGC hold-in time should not be set too long, or the receiver will be slow in returning to full sensitivity once the AGC is engaged by noise or interference.

The use of AGC is optional when using OOK modulation with data pulses of at least 30 µs. AGC operation can be defeated by connecting this pin to Vcc. Active or latched AGC operation is required for ASK modulation and/or for data pulses of less than 30 µs. The AGC can be latched on once engaged by connecting a resistor between this pin and ground (see 2.8.2 below). AGC operation depends on a functioning peak detector, as discussed above. The AGC capacitor is discharged in the receiver power-down mode and in the transmit modes.

The maximum AGC engage time t_{AGC} is 5 µs after the reception of a -30 dBm RF signal with a 1 µs envelope rise time.

2.8.2 AGC disabling or latching

AGC operation can be defeated by connecting the AGCCAP pin to Vcc. The AGC can be latched on once engaged by connecting a 150 kilohm resistor between the AGCCAP pin and ground in lieu of a capacitor. Latched AGC operation minimizes noise and interference sensitivity where the operating range is always short.

2.9 Transmitter Modulation

The transmitter chain consists of a SAW delay line oscillator followed by a modulated buffer amplifier. The SAW filter suppresses transmitter harmonics to the antenna. Note that the same SAW devices used in the amplifier-sequenced receiver are reused in the transmit modes.

Transmitter operation supports two modulation formats, on-off keyed (OOK) modulation, and amplitude-shift keyed (ASK) modulation. When OOK modulation is chosen, the transmitter output turns completely off between “1” data pulses. When ASK modulation is chosen, a “1” pulse is represented by a higher transmitted power level, and a “0” is represented by a lower transmitted power level. OOK modulation provides compatibility with first-generation ASH technology, and provides for power conservation. ASK modulation must be used for high data rates (data pulses less than 30 µs). ASK modulation also reduces the effects of some types of interference and allows the transmitted pulses to be shaped to control modulation bandwidth.

The modulation format is chosen by the state of the CNTRL0 and the CNTRL1 mode control pins. When either modulation format is chosen, the receiver RF amplifiers are turned off. In the OOK mode, the delay line oscillator amplifier TXA1 and the output buffer amplifier TXA2 are turned off when the voltage to the TXMOD input falls below 220 mV. In the OOK mode, the data rate is limited by the turn-on and turn-off times of the delay line oscillator, which are 12 and 6 µs respectively . In the ASK mode TXA1 is biased ON continuously, and TXA2 is modulated by the TXMOD input current.

2.9.1 OOK/ASK selection

On-off keyed (OOK) modulation should be chosen when compatibility with RFM’s HX-series transmitters and RX-series receivers is desired. OOK modulation also provides some power savings in the transmit mode, and can be used when the minimum pulse width in the transmitted signal is 30 µs or greater.

Amplitude-shift keyed (ASK) modulation should be chosen when the minimum pulse width in the transmitted signal is less than 30 µs. ASK modulation should also be used when the transmitted signal has been shaped for spectrum bandwidth control and/or when a specific modulation depth is required.

The modulation mode is selected with control lines CNTRL1 (Pin 17) and CNTRL0 (Pin 18), as described in section 2.11.1 below.

2.9.2 Transmitter power adjustment

Transmitter output power is proportional to the input current to TXMOD (Pin 8) as shown for the TR1000 in Figure 2.9.2.1. A series resistor is used to adjust the peak transmitter output power. Rated output power requires 250 to 450 µA of input current depending on operating frequency. In the ASK mode, minimum output power occurs when the modulation driver sinks about 10 µA of current from this pin. Figure 2.9.2.2 shows the relationship between V_{TXM} and I_{TXM}, again for the TR1000. Peak transmitter output power P_O for a 3 Vdc supply voltage is:

$P_O = 7*(I_{TXM})^2$ for 800 - 930 MHz operation, where P_O is in mW, and the peak modulation current I_{TXM} is in mA

$P_O = 16*(I_{TXM})^2$ for 400 - 450 MHz operation, where P_O is in mW, and the peak modulation current I_{TXM} is in mA

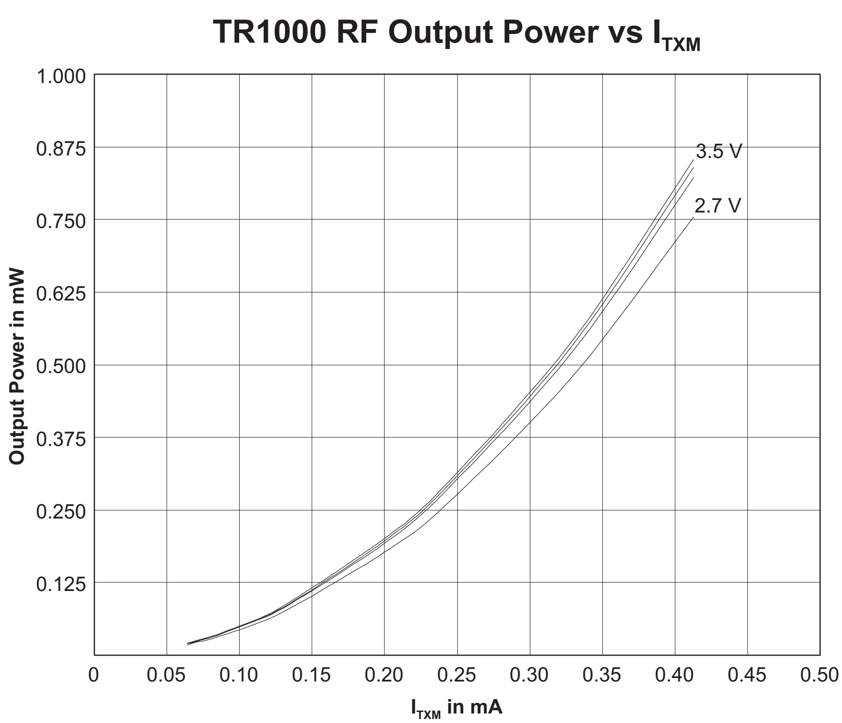


Figure 2.9.2.1

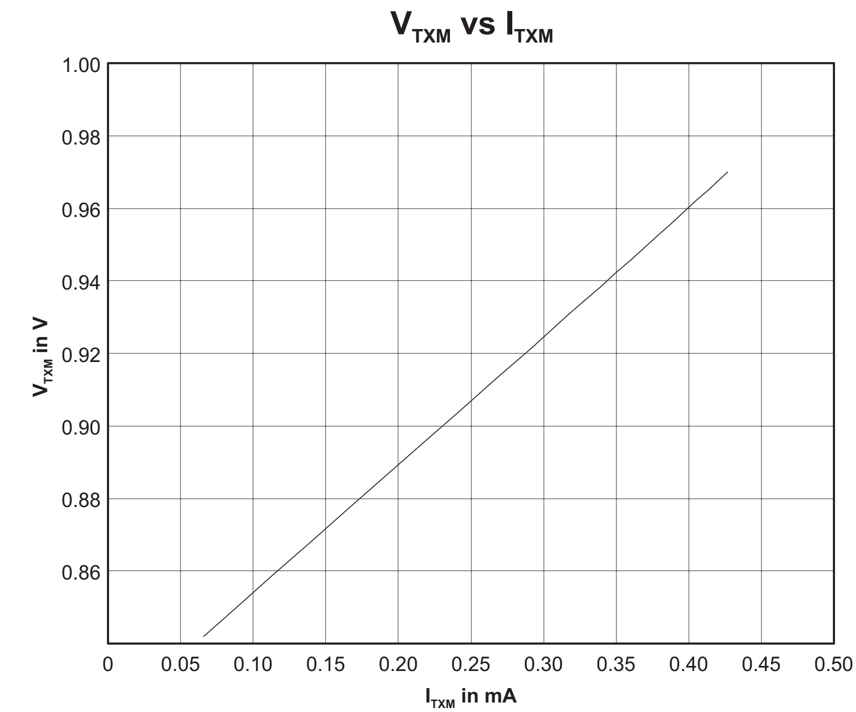


Figure 2.9.2.2

$P_O = 24*(I_{TXM})^2$ for 300 - 330 MHz operation, where P_O is in mW, and the peak modulation current I_{TXM} is in mA

A ±5% resistor is recommended. Typical resistor values for FCC Part 15 applications range from 4.7 to 11 K (TR1000), depending on the gain of the antenna used. Peak transmitter output power varies somewhat with supply voltage. Products operating from batteries should be adjusted for peak output power using a "fresh" battery to assure regulatory compliance. Supply voltage regulation should be used in systems where maximum operating range must be maintained over the operating life of the battery.

In the OOK mode, the TXMOD pin is usually driven with a logic-level data input (unshaped data pulses). OOK modulation is practical for data pulses of 30 µs or longer. In the ASK mode, the TXMOD pin accepts analog modulation (shaped or unshaped data pulses). As discussed above, ASK modulation is used for data pulses shorter than 30 µs. Note that the TXMOD input must be low in the power-down (sleep) mode.

ASK modulation depth adjustment

If the ASK transmitter mode is being used to allow the transmission of data pulses shorter than 30 µs, the same simple resistor calculation described above can be used to set peak transmitter output power. When the signal to the TXMOD resistor is brought close to

ASK Modulation Depth Control Circuit

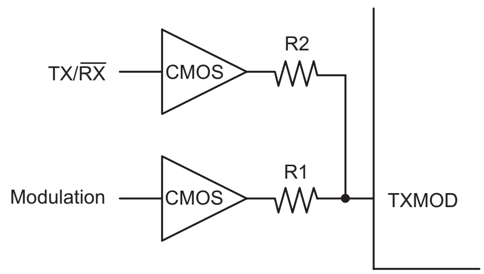


Figure 2.9.3.1

0 volts, maximum modulation depth is obtained. The modulation depth is usually greater than 45 dB, and is determined by the OFF isolation of TXA2.

The ASK modulation depth can be controlled over a range of 30 dB with relatively simple circuitry, as shown in Figure 2.9.3.1. Limiting ASK modulation depth is useful in improving system performance when certain types of weak interference are constantly present on an operating channel. Refer to RFM's application note, *Comparison of OOK, ASK and FSK Modulation*, at http://www.rfm.com for further information on this topic.

Referring to Figure 2.9.3.1, to control ASK modulation depth it is necessary to provide one TXMOD input current level (I_{MAX}) for peak output power, and a second input level (I_{MIN}) for the minimum output power. One approach to achieving this uses two CMOS buffers. The "TX/RX" buffer is held at a logic 1 during transmit and at a logic 0 during receive. The "Modulation" buffer is driven high and low by the transmit pulse stream. When the modulation buffer output is low, the transmitter output power is determined by the current through R1 minus the current going back into R2. The peak transmitter power is determined by the sum of the currents supplied by both gates through R1 and R2.

The values of R1 and R2 are calculated as follows. Using the peak output power P_O from 2.9.2 above as the high power level (TR1000 example):

> $I_{MAX} = (V_{TXH} - V_{TXMH})/R_{TXM}$, so $G_{TXM} = G1 + G2 = I_{MAX}/(V_{TXH} - V_{TXMH})$, where I_{MAX} is in mA, G_{TXM}, G1 and G2 are in millimho, and V_{TXH} is the logic 1 voltage, and V_{TXMH} is the V_{TXM} voltage for I_{MAX}

Next choose the low output power level (TR1000 example) :

> $I_{MIN} = (P_{MIN} / 4.8)^{0.5}$, where P_{MIN} is in mW and I_{MIN} is in mA

$G2 = (I_{MIN} - (I_{MAX} * ((V_{TXL} - V_{TXML}) / (V_{TXH} - V_{TXML})))) / (V_{TXH} - V_{TXL})$, where V_{TXL} is the logic 0 voltage level (0.2 V typical), V_{TXML} is the V_{TXM} voltage for I_{MIN}, and conductances are in millimho

and $G1 = ((I_{MAX} / (V_{TXH} - V_{TXMH})) - G2)$

R1 = 1/G1, and R2 = 1/(G2), where R1 and R2 are in kilohms

The above calculation provides starting point resistor values for a modulation depth of 30 dB or less. Figure 2.9.2.2 allows V_{TXMH} and V_{TXML} to be estimated for I_{MAX} and I_{MIN}.

2.10 Data Output

Pin 7 is the receiver data output pin (RXDATA). This pin will drive a 10 pF, 500 K parallel load (one CMOS gate). The peak current available from this pin increases with the receiver low-pass filter cutoff frequency. In the power-down or transmit modes, this pin becomes high impedance. If required, a 1000 K pull-up or pull-down resistor can be used to establish a definite logic state when this pin is high impedance.

Receiver Output Buffers

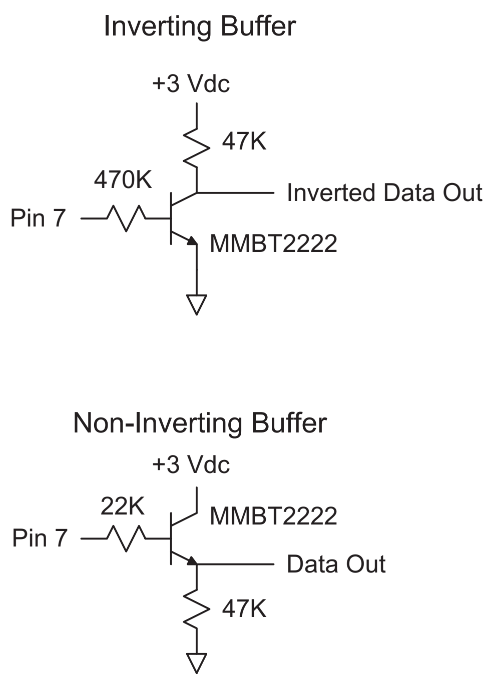


Figure 2.10.1

2.10.1 Buffering options

Figure 2.10.1 shows inverting and non-inverting buffer amplifiers for RXDATA. The buffers are suitable for driving loads down to 10 kilohms. Data communication through an ASH transceiver is non inverting; a positive data pulse transmits a (larger) radio signal that is output from the receiver as a positive pulse. The inverting buffer amplifier can be used to invert the RXDATA signal when desired.

2.11 Mode Control and Timing

The four transceiver operating modes – receive, transmit ASK, transmit OOK and power-down (sleep), are controlled by the Modulation & Bias Control function, and are selected with the CNTRL1 (Pin 17) and CNTRL0 (Pin 18) inputs. CNTRL1 and CNTRL0 are CMOS compatible. These inputs must be held at a logic level (not floating).

2.11.1 Mode control lines

Setting CNTRL1 and CNTRL0 both high place the unit in the receive mode. Setting CNTRL1 high and CNTRL0 low place the unit in the ASK transmit mode. Setting CNTRL1 low and CNTRL0 high place the unit in the OOK transmit mode. Setting CNTRL1 and CNTRL0 both low place the unit in the power-down (sleep) mode. Note that the resistor driving TXMOD (Pin 8) must be low in the receive and sleep modes. PWIDTH (Pin 15) must also be low in the sleep mode to minimize power supply current. When using the pulse generator in the high data rate mode, connect the 1 M resistor from the PWIDTH pin to the CNTRL1 pin, so that the "hot" side of the resistor is brought low when CNTRL1 and CNTRL0 are brought low to select the sleep mode.

2.11.2 Turn-on timing

The maximum time t_{PR} required for the receive function to become operational at turn on is influenced by two factors. All receiver circuitry will be operational 5 ms after the supply voltage reaches 2.7 Vdc. The BBOUT-CMPIN coupling-capacitor is then DC stabilized in 3 time constants. The total turn-on time t_{PR} to stable receiver operation for a 10 ms power supply rise time is 15 ms + 3*t_{BBC}, where t_{BBC} is coupling capacitor time constant (see section 2.6). The transceiver should be turned on in the receive mode until the supply voltage reaches 2.7 Vdc..

The maximum time required for either the OOK or ASK transmitter mode to become operational is 5 ms after the supply voltage reaches 2.7 Vdc (switch from receive mode).

2.11.3 Transmit-to-receive timing

The maximum time required to switch from the OOK or ASK transmit mode to the receive mode is 3*t_{BBC}, where t_{BBC} is the BBOUT-CMPIN coupling-capacitor time constant. When the operating temperature is limited to 60 ^{o}C, the time required to switch

Receive-Mode Timing, 115.2 kbps, 60⁰ C,
-76 dBm Input (t_{BBC} = 173 µs)

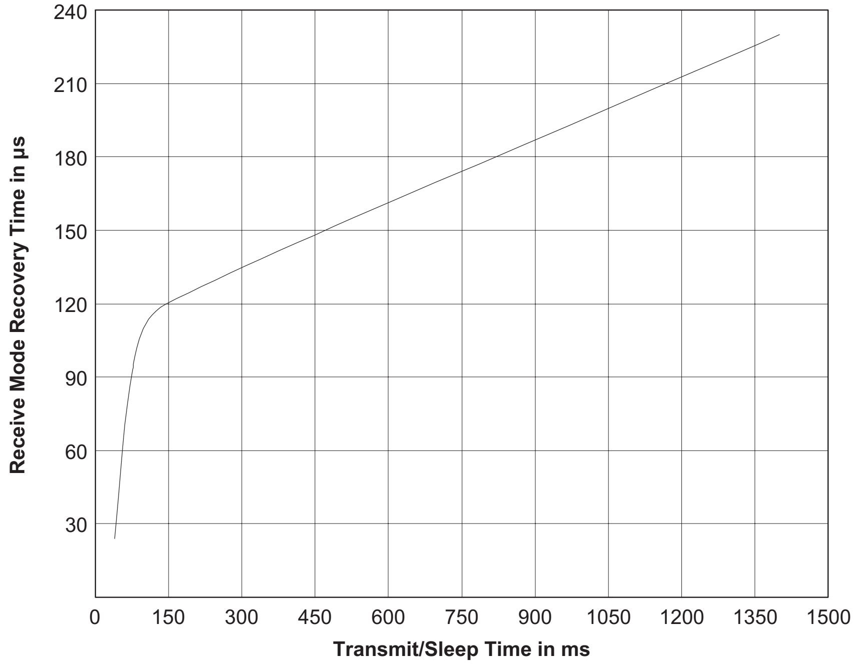


Figure 2.11.3.1

from transmit to receive is dramatically less for short transmissions, as less charge leaks away from the BBOUT-CMPIN coupling capacitor. Figure 2.11.3.1 shows a typical curve for operation at 115.2 kbps

2.11.4 Receive-to-transmit timing

After turn-on stabilization, the maximum time required to switch from receive to either transmit mode is 12 µs. Most of this time is the start-up of the transmitter oscillator.

2.11.5 Power-down and wake-up timing

The maximum transition time from the receive mode to the power-down (sleep) mode t_{RS} is 10 µs after CNTRL1 and CNTRL0 are both low (1 µs fall time). The maximum transition time from either transmit mode to the power-down mode (t_{TOS} and t_{TAS}) is 10 µs after CNTRL1 and CNTRL0 are both low (1 µs fall time).

The maximum transition time t_{SR} from the sleep mode to the receive mode is 3*t_{BBC}, where t_{BBC} is the BBOUT-CMPIN coupling-capacitor time constant. When the operating temperature is limited to 60 ^{o}C, the time required to switch from sleep to receive is dramatically less for short sleep times, as less charge leaks away from the BBOUT- CMPIN coupling capacitor. Figure 2.11.3.1 shows a typical curve for operation at 115.2 kbps.

The maximum time required to switch from the sleep mode to either transmit mode (t_{STO} and t_{STA}) is 16 µs. Most of this time is due to the start-up of the transmitter oscillator.

2.12 Application Circuits

The ASH transceiver can be tailored to a wide variety of applications requirements, allowing emphasis to be placed on simplicity or high performance. The four most common application circuit configurations are presented below.

2.12.1 Minimum OOK configuration

The minimum OOK configuration is shown in Figure 2.12.1. This circuit is suitable for transmitting data with a minimum pulse width of 30 µs. The power-down mode is not implemented, allowing a single control line (CNTRL1) to select OOK transmit or receive. Data slicer DS1 is implemented with threshold. Data slicer DS2 and AGC are not implemented. Only 14 external components are required to implement this transceiver configuration. This configuration is compatible with first-generation HX/RX technology.

ASH Transceiver Application Circuit
Minimum OOK Configuration

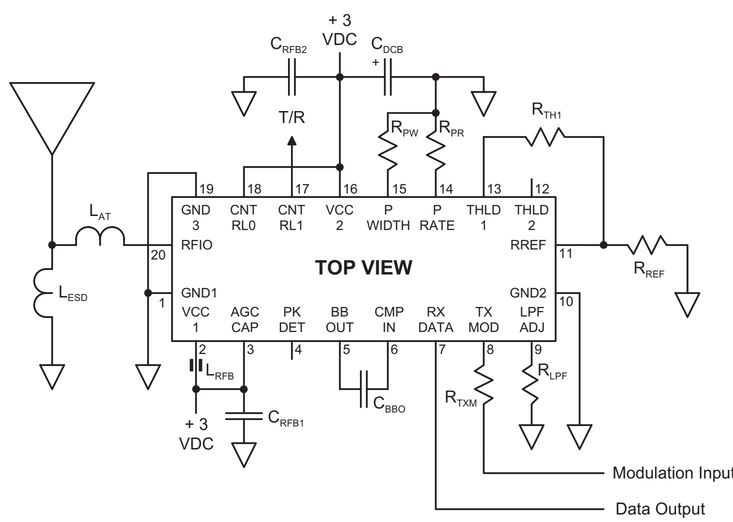


Figure 2.12.1

2.12.2 Standard OOK/ASK configuration

The standard OOK/ASK configuration is shown in Figure 2.12.2. This circuit is suitable for transmitting OOK data with a minimum pulse width of 30 µs, or ASK data at any data rate supported by the ASH transceiver being used. Both control lines to the transceiver can be toggled, allowing for the selection of receive, power-down, OOK transmit and

RFM.

ASH Transceiver Application Circuit
Standard OOK/ASK Configuration

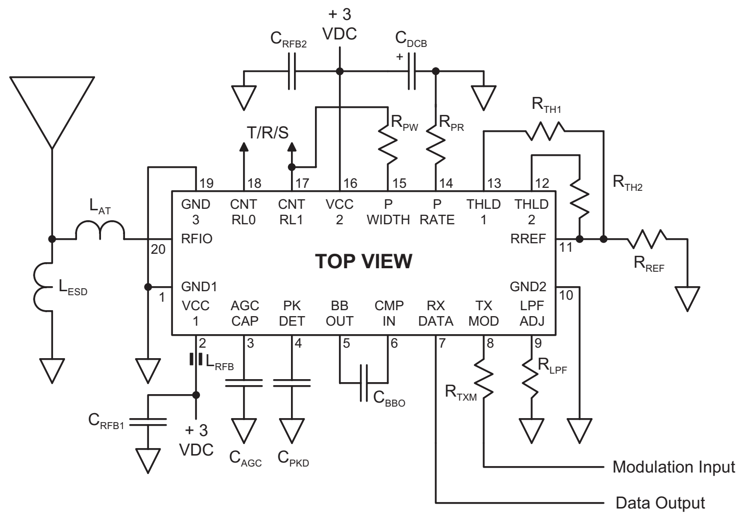


Figure 2.12.2

ASH Radio Application Circuit
Receive-Only Configuration (OOK)

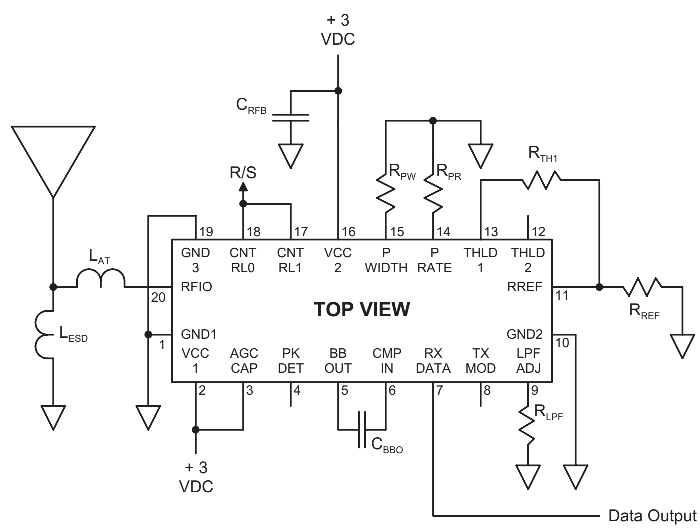


Figure 2.12.3

RFM

ASK transmit. Data slicer DS2 is implemented for good performance at higher data rates. AGC is also implemented for high dynamic range ASK operation, and to support limited-range OOK or ASK operation in the presence of strong interference. Seventeen external components are required to implement this flexible configuration.

2.12.3 Receive-only configuration (OOK)

Figure 2.12.3 shows the receive-only configuration for OOK. It can be used with either an ASH transceiver or a second-generation ASH receiver. Receive and sleep modes are implemented using a single control line, which can be tied to Vcc for continuous operation. Data slicer DS1 is implemented with threshold. Data slicer DS2 and AGC are not implemented. Only nine external components are required to implement the OOK receive-only configuration.

2.12.4 Transmit-only configuration (OOK)

Figure 2.12.4 shows the transmit-only configuration (OOK). It can be used with either an ASH transceiver or a second-generation ASH transmitter. Only eight external components are required to implement this configuration. The modulation input line must held below 220 mV between transmissions to minimize transmitter current consumption.

ASH Radio Application Circuit
Transmit-Only Configuration (OOK)

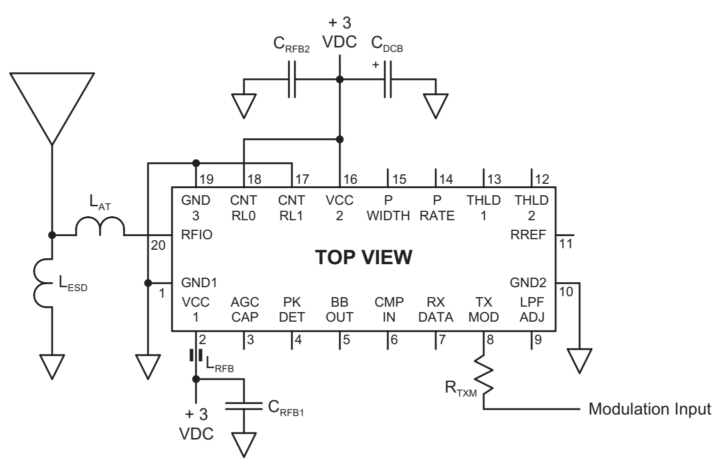


Figure 2.12.4

2.12.5 Set-up table

Table 2.12.5 provides component values for the above configurations at a number of standard data rates. Component values for other data rates can be computed using the formulas provided above and in the ASH transceiver data sheets.

TR1000 Transceiver Set-Up, 3 Vdc, -40 to +85 deg C

Item	Symbol	OOK	OOK	ASK	ASK	Units	Notes
Nominal NRZ Data Rate	DR_{NOM}	2.4	19.2	57.6	115.2	kbps	
Minimum Signal Pulse	SP_{MIN}	416.67	52.08	17.36	8.68	µs	single bit
Maximim Signal Pulse	SP_{MAX}	1666.68	208.32	69.44	34.72	µs	4 bits of same value
AGCCAP Capacitor	C_{AGC}	-	-	4700	2200	pF	±10% ceramic
PKDET Capacitor	C_{PKD}	-	-	0.002	0.001	µF	±10% ceramic
BBOUT Capacitor	C_{BBO}	0.1	0.015	0.0056	0.0027	µF	±10% ceramic
TXMOD Resistor	R_{TXM}	8.2	8.2	8.2	8.2	K	±5%, for 0.25 mW output
LPFADJ Resistor	R_{LPF}	240	30	25	12	K	±5%
RREF Resistor	R_{REF}	100	100	100	100	K	±1%
THLD2 Resistor	R_{TH2}	-	-	100	100	K	±1%, for 6 dB below peak
THLD1 Resistor	R_{TH1}	10	27	100	100	K	±1%, typical values
PRATE Resistor	R_{PR}	1100	330	160	160	K	±5%
PWIDTH Resistor	R_{PW}	270 to GND	270 to GND	1000 to VCC	1000 to VCC	K	±5% (see section 2.11.1)
RF Bypass Resistor	R_{RFB}	100	100	100	100	ohm	±5%
DC Bypass Capacitor	C_{DCB}	10	10	10	10	µF	Tantalum
RF Bypass Capacitor 1	C_{RFB1}	27	27	27	27	pF	±5% NPO Ceramic
RF Bypass Capacitor 2	C_{RFB2}	100	100	100	100	pF	±5% NPO Ceramic
Antenna Tuning Inductor	L_{AT}	10	10	10	10	nH	50 ohm antenna
ESD Choke	L_{ESD}	100	100	100	100	nH	50 ohm antenna
RF Bypass Bead	L_{RFB}	2506033017YO	2506033017YO	2506033017YO	2506033017YO	P/N	Fair-Rite or equivalent
LPF 3 dB Bandwidth	f_{LPF}	6	48	57.6	115.2	kHz	see section 2.5.2
LPF Group Delay	t_{FGD}	292	36	30	15	µs	$1750/f_{LPF}$
BBOUT Time Const	t_{BBC}	6400	960	358	173	µs	less than 20% "droop" for SP_{MAX}
Samples/bit	f_B	126	34.7	20	10	spb	
PWIDTH RFA1	t_{PW1}	0.71	0.71	0.435	0.435	µs	
PWIDTH RFA2	t_{PW2}	0.79	0.79	0.435	0.435	µs	
RFA1 % ON Time	t_{PO1}	21	47	50	50	%	
PRATE Interval	t_{PRI}	2.6	0.79	-	-	µs	
PRATE Cycle	t_{PRC}	-	-	0.87	0.87	µs	
PWIDTH High (RFA1/2)	t_{PWH}	-	-	0.435	0.435	µs	
PKDET Attack Time Const	t_{PKA}	-	-	480	240	µs	~20% bit period
PKDET Decay Time Const	t_{PKD}	-	-	0.48	0.24	ms	$1000*t_{PKA}$
AGC Hold-In	t_{AGH}	-	-	246	115	µs	~16 bit hold-in
TXOOK to RX	t_{TOR}	19.2	2.9	1.1	0.52	ms	see section 2.11.3
TXASK to RX	t_{TAR}	19.2	2.9	1.1	0.52	ms	see section 2.11.3
Sleep to RX	t_{SR}	19.2	2.9	1.1	0.52	ms	see section 2.11.5
TX Peak Output Power	P_{OP}	0.25	0.25	0.25	0.25	mW	typical for FCC 15.249

Table 2.12.5

2.13 PCB Layout and Assembly

Figure 2.13.1 is the schematic of the DR1200 data radio board, which is used in the 916.5 MHz DR1200-DK development kit. The following discussions will use the DR1200 and an example. Note that the board is designed to allow testing of both OOK and ASK modulation, and to allow pulse generator operation in either the low data rate or high data rate mode. Depending on which modulation and pulse generator set-up is chosen, some resistors may be left off the board or replaced with “zero ohm” jumper resistors. The complete manual for the DR1200-DK is available on RFM’s web site at http://www.rfm.com. The manual includes the bill of materials and other information on the DR1200 data radio board.

DR1200 Data Radio Schematic

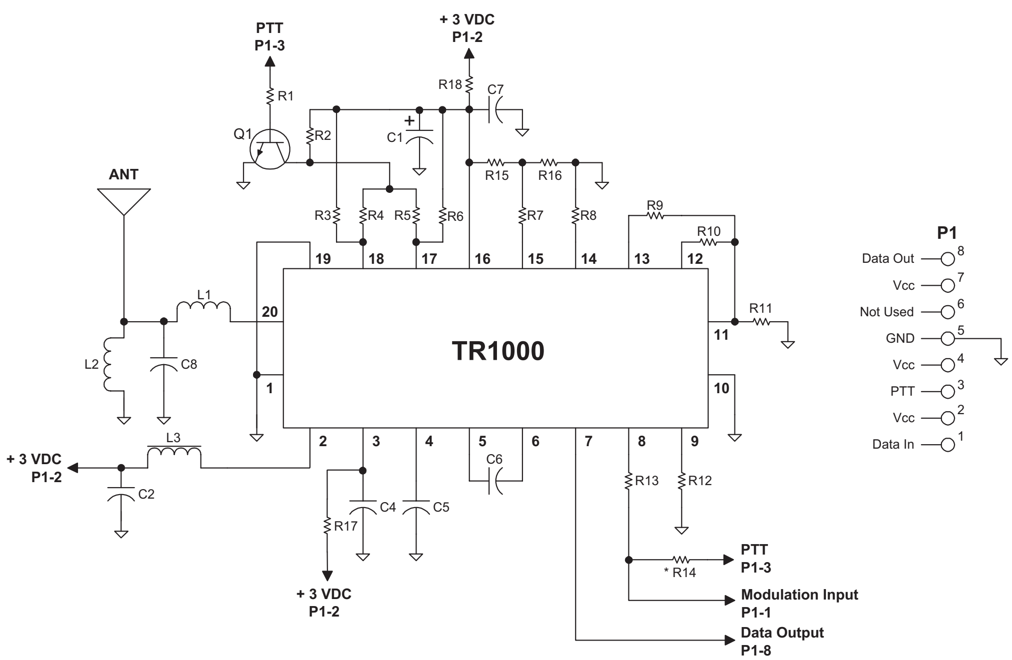


Figure 2.13.1

2.13.1 PCB layout

Figure 2.13.1.1 shows the outline drawing of the TR1000 and 2.13.1.2 shows the DR1200 printed circuit board (PCB) layout. The DR1200 layout is done on a two-layer printed circuit board. The bottom of the board is a solid ground plane. Ground connections are made from the top of the circuit board to the ground plane using plated-through holes. Note the special care used in the layout to keep all PCB traces as short as possible.

Pin 2 is the power supply pin to TXA2, and is decoupled with a ferrite bead. The D1 component shown on the layout is an optional ESD protection diode for severe ESD environments. The C8 component shown in the layout is an optional RF capacitor that can be used to tune reactive antennas.

ASH Transceiver SM-20H Outline Drawing

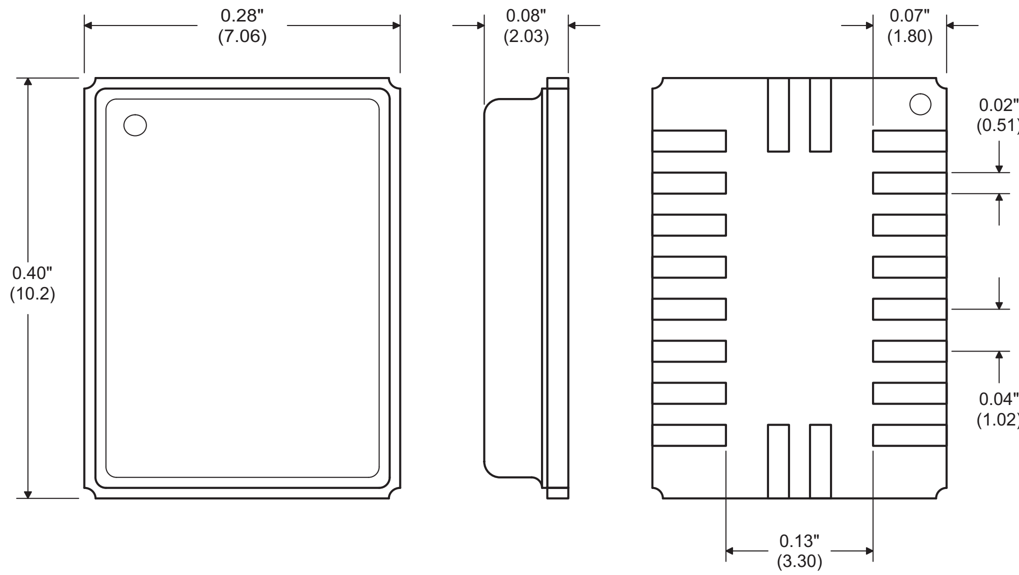


Figure 2.13.1.1

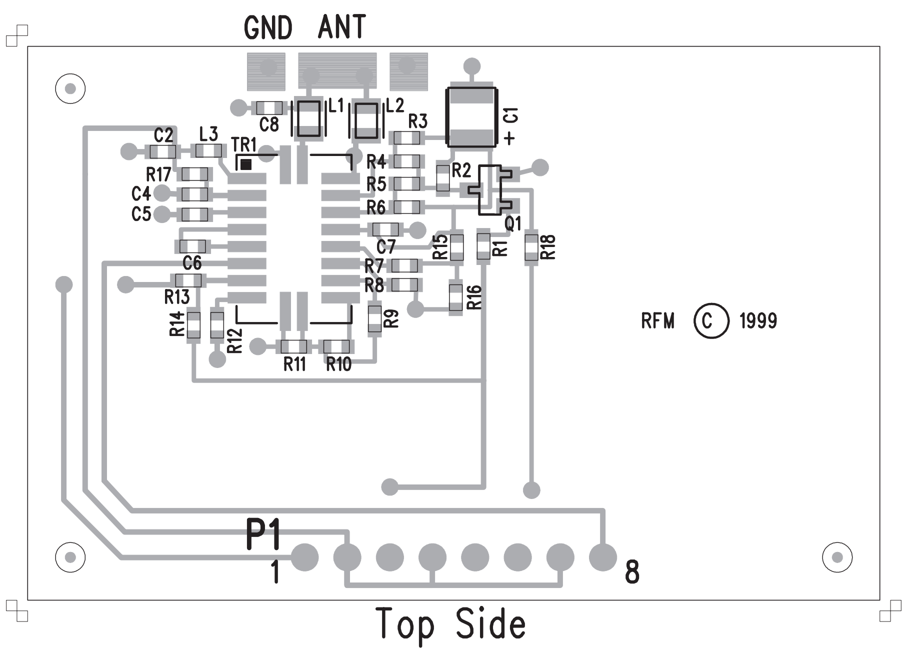


Figure 2.13.1.2

2.13.2 PCB assembly

Figure 2.13.2.1 shows the recommended temperature profile for reflow soldering second-generation ASH radio hybrids. The hybrid package consists of a ceramic base with a metal lid that is attached with high-temperature solder. The transceiver package is hermetic and the solder seal must not be compromised with excessive heat in assembly. It is critical that the transceiver package is never heated above 250 ^{o}C. It is recommended that the transceiver package be heated no higher than 240 ^{o}C for no more than 10 seconds.

Reflow Soldering Profile (SN63)

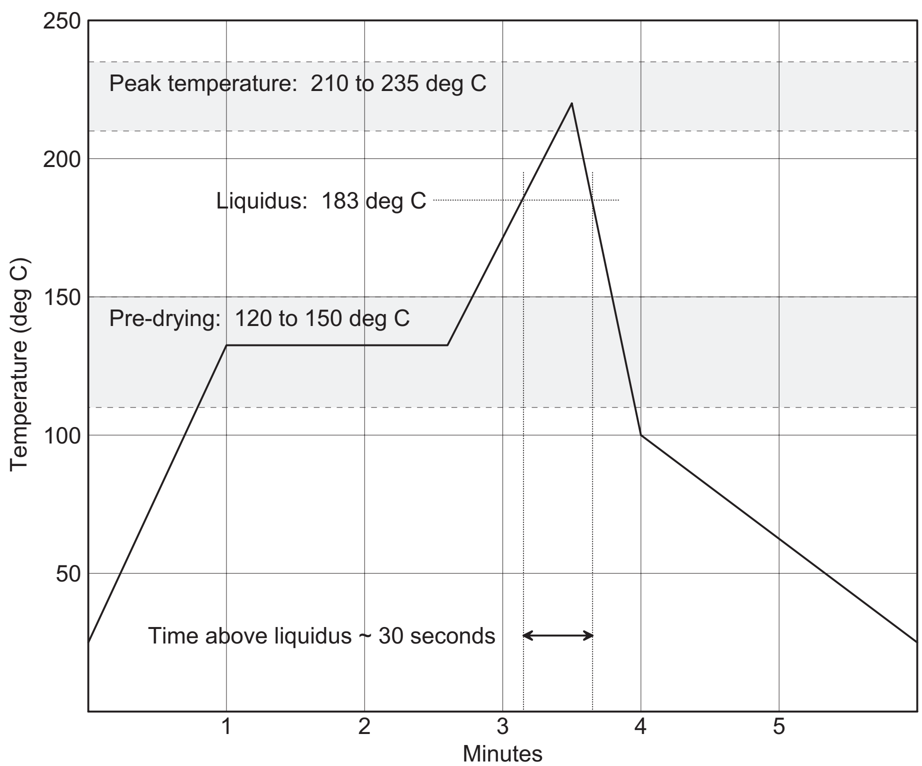


Figure 2.13.2.1

Note: Specifications subject to change without notice.

3 Appendices

3.1 Example Operating Distance Calculation

This example estimates the operating distance of a short-range wireless system transmitting 12-bit encoded data at 19.2 kbps using OOK modulation and no threshold at the receiver. A 3 dB filter bandwidth of 14.4 kHz is used (noise BW = 1.25 * 3 dB BW). Average transmitter output power is -9 dBm. A receiver noise figure of 7.5 dB is assumed. Antennas with 1 dB of gain are used. A 20 dB fade margin is chosen (99% Rayleigh probability). Packets are 38 bytes long (excluding preamble), or 456 bits. The system goal is to achieve 90% packet reads on the first try. The operating frequency is 916.5 MHz. Estimate the interference-free operating range:

A single bit error will result in a packet error. Only one bit error in 10 packets can be accepted or:

$$BER = 1/(456*10) = 2.193E-4$$

The required signal-to-noise ratio to achieve this BER using non-coherent detection of OOK modulation is:

$$10*\log_{10}(-2*\ln(2*BER)) = 12 \text{ dB}$$

adding 7.5 dB for receiver noise figure, 3 dB for sampling loss and 7 dB for implementation loss:

$$12 + 17.5 = 29.5 \text{ dB}$$

The detected noise power (double sideband) through the 14.4 kHz filter is:

$$N = -174 \text{ dBm} + 10*\log_{10}(2*1.25*14400) = -128.4 \text{ dBm}$$

The signal level required is then:

$$-128.4 + 29.5 = -98.9 \text{ dBm}$$

The allowed path loss is:

$$L_{PATH} = P_O + G_{ATX} + G_{ARX} - L_{FADE} - S_{RX}$$

where P_O is the transmitter peak output power, G_{ATX} is the transmitter antenna gain (over isotropic), G_{ARX} is the receiver antenna gain, L_{FADE} is the fade margin, and S_{RX} is the required received signal strength. Assuming a 20 dB fade margin:

$$L_{PATH} = -9 \text{ dBm} + 1 \text{ dB} + 1 \text{ dB} - 20 \text{ dB} - (-98.9 \text{ dBm}) = 71.9 \text{ dB}$$

Now comes the trickiest part of the estimate. For ideal free space propagation, path loss is directly proportional to the square of the distance, or $20*\log_{10}(D)$, and is also directly proportional to the square of the operating frequency, or $20*\log_{10}(f)$. The equation for distance in meters is:

L_{PATH} = -27.6 dB + $20*\log_{10}(f) + 20*\log_{10}(D)$, where f is in MHz and D is in m

71.9 = -27.6 dB + 59.2 + $20*\log_{10}(D)$; D = 103.5 meters, or 339.4 feet

Again, this range can only be achieved under ideal free space conditions, approximated by mounting your equipment at the top of two 100 meter towers spaced 103.5 meters apart. Down on the ground, and especially in dense cubical office space where propagation loss can be higher than $1/d^4$, the practical operating range is much less. One of the more commonly used propagation models for near ground and/or indoor use is the simplified Keenan-Motley (IBM Zurich) equation:

L_{PATH} = -27.6 dB + $20*\log_{10}(f) + N*10*\log_{10}(D)$, where N is 2 or greater, f is in MHz and D is in m

As before, N = 2 for free space propagation. N = 2.5 is typical for UHF propagation 1.5 meter above the ground in an open field or large, open indoor space. N = 3 is typical for indoor open office and retail space, and N = 4 is typical of dense cubical office space. For N = 2.5, N = 3 and N = 4 our estimated operating distance is:

N = 2.5:

71.9 = -27.6 dB + 59.2 + $25*\log_{10}(D)$; D = 40.9 meters, or 134.1 feet

N = 3:

71.9 = -27.6 dB + 59.2 + $30*\log_{10}(D)$; D = 22.0 meters, or 72.2 feet

N = 4:

71.9 = -27.6 dB + 59.2 + $40*\log_{10}(D)$; D = 10.2 meters, or 33.0 feet

These range estimates are generally less than "real world" observations made using RFM Virtual Wire Development Kits as propagation survey tools. This is due to the conservative choice of a 20 dB fade margin, and the stringent packet error rate performance criteria used in these calculations.

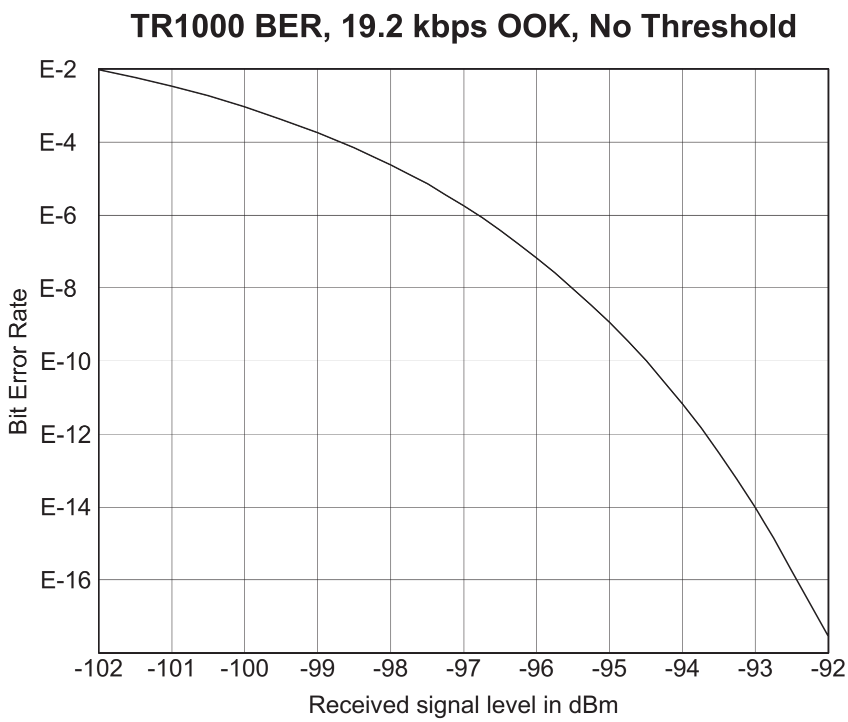


Figure 3.1.1

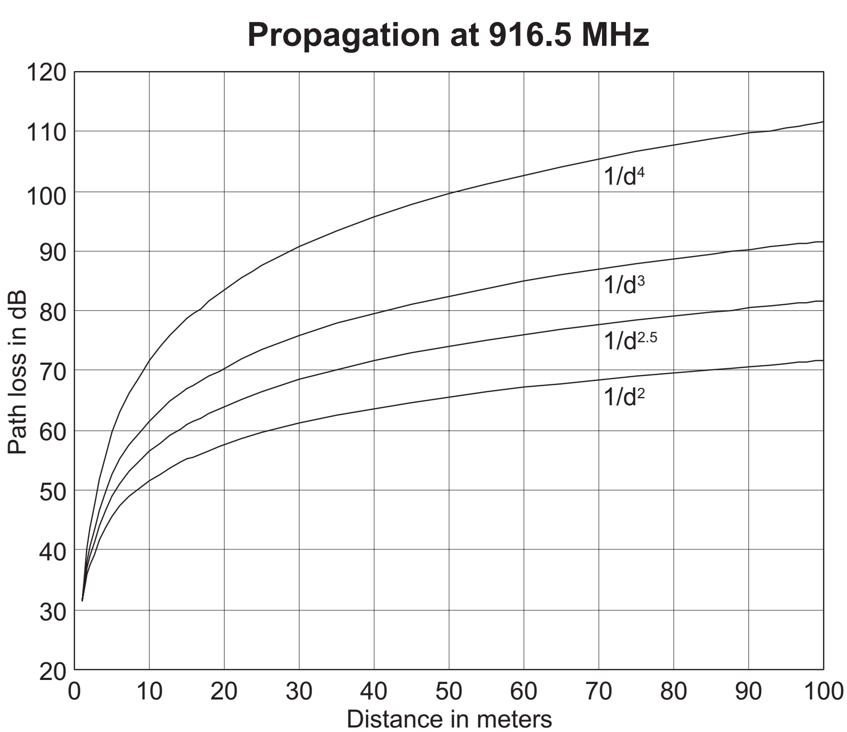
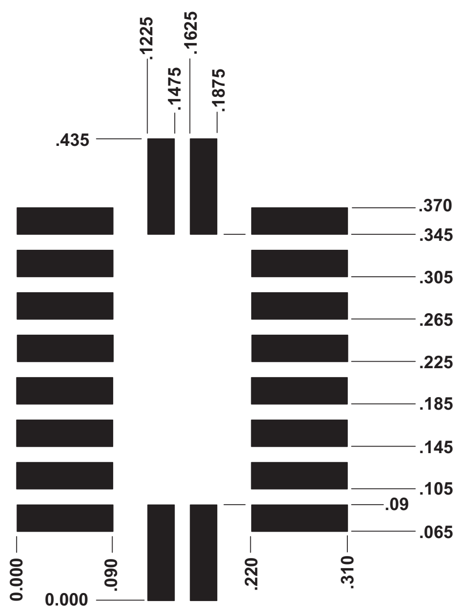


Figure 3.1.2

3.2 PCB Pad Layouts

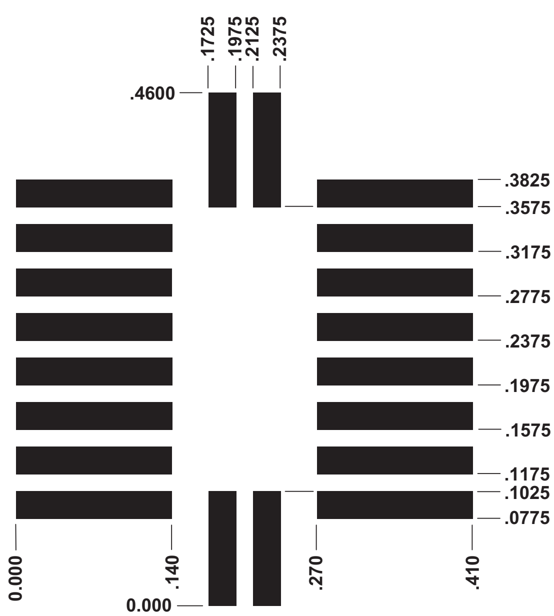
The SM-20H is the high frequency (800 - 1000 MHz) ASH radio package. The SM20-L is the low frequency (300 - 450 MHz) ASH radio package.



Dimensions in inches.

SM-20H PCB Pad Layout

Figure 3.2.1



Dimensions in inches

SM-20L PCB Pad Layout

Figure 3.2.2

3.3 Byte to 12-Bit DC-Balanced Symbol Conversion

The QuickBasic program below is an example of DC-balanced encoding and decoding. Encoding and decoding are done by mapping between nibbles (4 bits) and 6-bit half-symbols using a look-up table.

```
' DC_BAL.BAS  2000.12.22 @ 10:00 CST
' Copyright 2000, RF Monolithics, Inc.
' Converts any 4-bit pattern to 6-bit DC-balanced pattern

SCREEN 0
WIDTH 80
CLS

DEFINT A-Z                                          ' 16 bit integers
DIM BTbl(0 TO 15)                                   ' BTbl holds 6-bit patterns
GOSUB BldTbl                                        ' build symbol BTbl

DO

  INPUT "Input nibble (0 to 15):  ", N              ' get test nibble
  IF (N < 0) OR (N > 15) THEN EXIT DO               ' exit if out of range
  S = BTbl(N)                                       ' get half-symbol from table
  PRINT
  PRINT N; "maps to"; S; "("; HEX$(S); " Hex)"      ' print nibble and half-symbol
  GOSUB GetNibl                                     ' now get nibble back from half-symbol
  PRINT
  PRINT S; "maps back to"; NN; "("; HEX$(NN); " Hex)"
  PRINT

LOOP

PRINT
PRINT "Input out of range"

END

GetNibl:

  Q = 0                                             ' zero table index
  DO                                                ' nibble look-up loop
    IF S = BTbl(Q) THEN                             ' look-up test
      EXIT DO                                       ' got match so exit
    END IF
    Q = Q + 1                                       ' else increment index
     IF Q > 15 THEN                                 ' if not in table
       PRINT " Not in table!"                       ' print warning
       END                                          ' exit program
     END IF
  LOOP
  NN = Q                                            ' Q is decoded nibble

RETURN

BldTbl:

 BTbl(0) = 13        ' 0D hex
 BTbl(1) = 14        ' 1E hex
 BTbl(2) = 19        ' 13 hex
 BTbl(3) = 21        ' 15 hex
 BTbl(4) = 22        ' 16 hex
 BTbl(5) = 25        ' 19 hex
 BTbl(6) = 26        ' 1A hex
 BTbl(7) = 28        ' 1C hex
 BTbl(8) = 35        ' 23 hex
 BTbl(9) = 37        ' 25 hex
 BTbl(10) = 38       ' 26 hex
 BTbl(11) = 41       ' 29 hex
 BTbl(12) = 42       ' 2A hex
 BTbl(13) = 44       ' 2C hex
 BTbl(14) = 50       ' 32 hex
 BTbl(15) = 52       ' 34 hex

RETURN
```

3.4 Second-Generation ASH Transmitters and Receivers

The same technology developed for the ASH transceiver is used in the second-generation ASH transmitter and receiver hybrids to support demanding one-way control and telemetry applications. All second-generation ASH radios utilize a standardized 20 pin layout. Pins related to the transmit function are in the same location and have the same input/output electrical characteristics on both second-generation ASH transmitters and transceivers. Likewise, all active pins related to the receive function are in the same location and have the same input/output electrical characteristics on both second-generation ASH receivers and transceivers. This makes it possible to do a single PCB layout and build it as a transmitter, receiver or transceiver.

There are a few differences between second-generation ASH transmitter operation and ASH transceiver operation in the transmit mode. In the OOK mode, the transmit turn-on and the turn-off times are greater in the ASH transmitter than in the ASH transceiver (in ASK mode, turn-on and turn-off times are comparable). Also, the transmit-to-sleep and sleep-to-transmit times are greater for the ASH transmitter than for the ASH transceiver.

Second-generation ASH receivers with RX50xx and RX60xx part numbers operate identically to ASH transceivers in the receive mode. Second-generation ASH receivers with RX55xx and RX65xx part numbers do not have data slicer DS2, the peak detector or the AGC implemented. RX55xx and RX65xx receivers are intended for operation with encoded data rates up to 19.2 kbps. RX55xx and RX65xx receivers are also characterized using test methods common to control rather than data applications.

Please refer to the individual product data sheets for further information.

3.5 EMI Robust ASH Radio PCB Layouts

Electromagnetic compatibility (EMC) testing is required prior to marketing short-range wireless devices in Europe, and for certain industrial applications worldwide. EMC testing is done by applying an RF field of a specific strength (measured in V/m) to confirm the product's operation is not disrupted due to electromagnetic interference (EMI). The minimum field strength used in EMC testing is 3 V/m. EMC testing is typically done over a range of frequencies from 10 MHz to 1 GHz, except for an exclusion band around the operating frequency of the radio.

Second-generation ASH radios have been specifically developed for EMI robustness. For best results, however, these radios must be used in application circuits and PCB layouts designed for robust EMI performance. Figure 3.5.1 shows the schematic of an EMI robust application circuit, and Figures 3.5.2 and Figures 3.5.3 show EMI robust PCB layouts for the SM-20H and SM-20L ASH radio packages. The Gerber files for these layouts are located on RFM's web site, http://www.rfm.com, under Application Notes.

Referring to Figure 3.5.1, note that mode control pin 17 is decoupled with RF capacitor C7. Referring to Figures 3.5.1, 3.5.2 and 3.5.3, note the special Vcc routing under the ASH radio and the Vcc RF decoupling capacitors on both sides of the radio package. Also note the heavy use of ground plane on the top of the PCB, connected directly to the solid ground plane underneath with many feed-through connections.

For EMC testing at 3 V/m, special grounding of the ASH radio hybrid lid is not usually required. But for higher field strengths, it may be necessary to ground the lid with a small clip or wire, or cover the top of the PCB with a small "tin plate" shield.

ASH Radio Application Circuit
EMI Robust Configuration

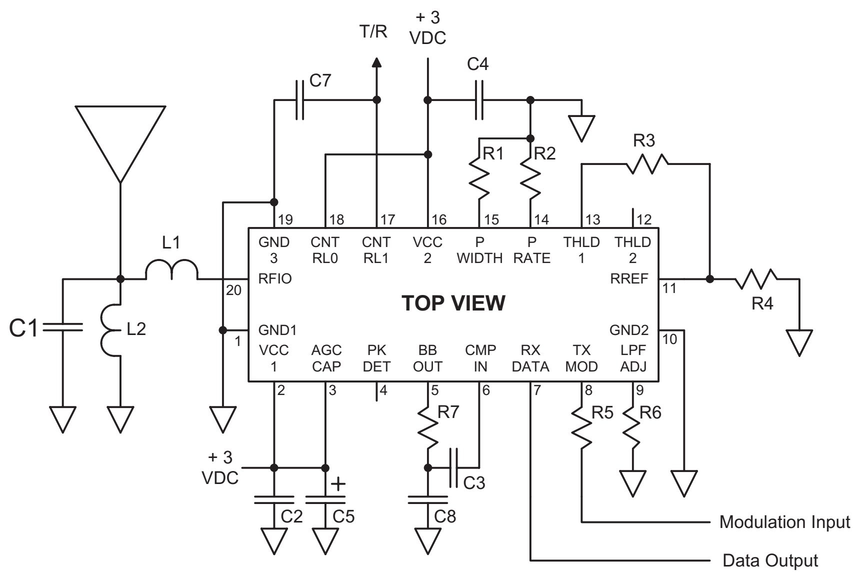


Figure 3.5.1

The strong RF fields used in EMC testing can disrupt the operation of op-amps, regulators, analog-to-digital converters and even logic circuits. It is important to use compact PCB layouts and adequate RF decoupling in the electronics throughout the product. *It is especially important to decoupe RF from the Vcc supply to the ASH radio.*

EMI Robust ASH Radio PCB Layout
SM-20H Package

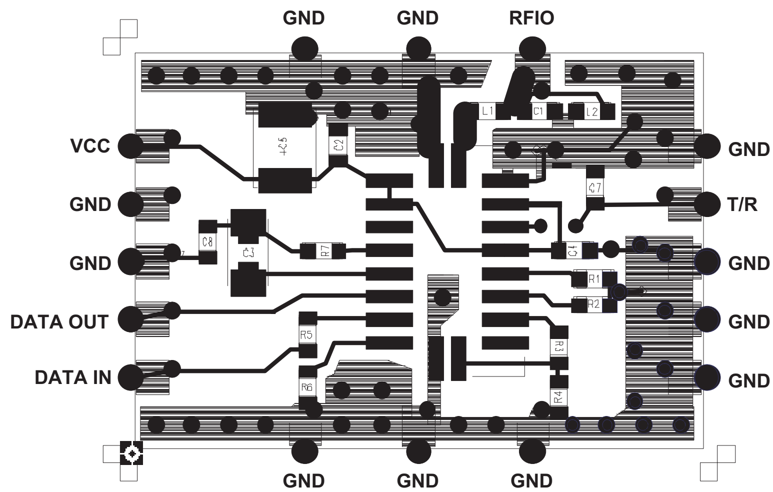


Figure 3.5.2

EMI Robust ASH Radio PCB Layout
SM-20L Package

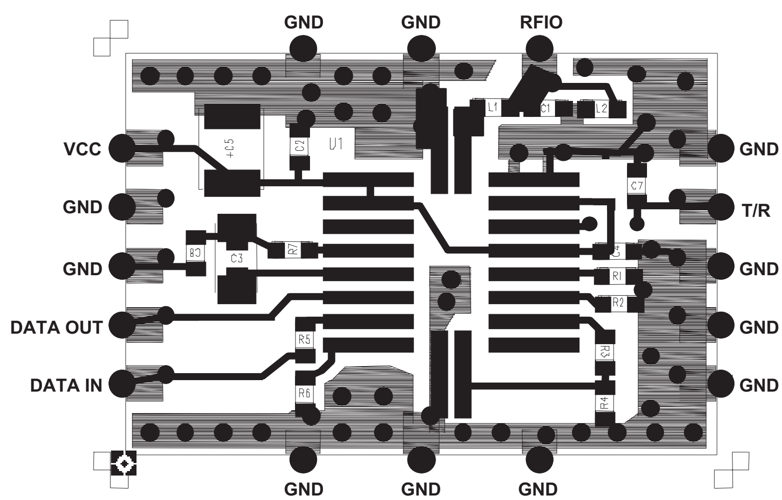


Figure 3.5.3

Bill of Materials, 868.35 MHz (SM-20H Package)

Part Identifier	Description	Qty/Assy	Reference
sm20hemi.pho	Printed Circuit Board	1	PCB1
TR1001	ASH Transceiver, 868.35 MHz	1	U1
500-0619-100	Inductor, 0603 SMT, 10 nH, ±10%	1	L1
500-0619-101	Inductor, 0603 SMT, 100 nH, ±10%	1	L2
500-0621-101	Capacitor, 0603 SMT, 100 pF	3	C2, C4, C7
500-0621-104	Capacitor, 0805 SMT, 0.1 µF	1	C3
500-0675-106	Capacitor, SMT, 10 µF, Kermit T491B106K006AS	1	C5
500-0620-274	Resistor, 0603 SMT, 270 K, 1/16 W	1	R1
500-0620-472	Resistor, 0603 SMT, 4.7 K, 1/16 W	2	R3, R5
500-0828-104	Resistor, 0603 SMT, 100 K, 1/16 W	1	R4
500-0620-334	Resistor, 0603 SMT, 330 K, 1/16 W	2	R2, R6
500-0620-000	Resistor, 0603 SMT, 0 K, 1/16 W	1	R7
N/A	Not Used	N/A	C1, C6, C8

Bill of Materials, 433.92 MHz (SM-20L Package)

Part Identifier	Description	Qty/Assy	Reference
sm20lemi.pho	Printed Circuit Board	1	PCB1
TR3000	ASH Transceiver, 433.92 MHz	1	U1
500-0619-680	Inductor, 0603 SMT, 68 nH, ±10%	1	L1
500-0619-101	Inductor, 0603 SMT, 100 nH, ±10%	1	L2
500-0621-080	Capacitor, 0603 SMT, 8 pF	1	C1
500-0621-101	Capacitor, 0603 SMT, 100 pF	3	C2, C4, C7
500-0621-104	Capacitor, 0805 SMT, 0.1 µF	1	C3
500-0675-106	Capacitor, SMT, 10 µF, Kermit T491B106K006AS	1	C5
500-0620-274	Resistor, 0603 SMT, 270 K, 1/16 W	1	R1
500-0620-472	Resistor, 0603 SMT, 4.7 K, 1/16 W	2	R3, R5
500-0828-104	Resistor, 0603 SMT, 100 K, 1/16 W	1	R4
500-0620-334	Resistor, 0603 SMT, 330 K, 1/16 W	2	R2, R6
500-0620-000	Resistor, 0603 SMT, 0 K, 1/16 W	1	R7
N/A	Not Used	N/A	C6, C8

3.6 Modulation Bandwidth Control

To comply with ETSI EN 300 220-1 regulations, SRD transmitter modulation sidebands must be suppressed to at least 250 nW (-36 dBm) outside of the band or sub-band of operation (see EN 300 220-1 Section 8.6 for test details). The modulation bandwidth of an ASH transmitter or transceiver can be controlled by low-pass filtering the signal to the TXMOD input (Pin 8). For transmitted data rates up to 20 kbps (data pulses 50.0 µs or greater), the simple low-pass filter shown in Figure 3.6.1 below can be used to meet ETSI requirements under most circumstances. The filter in Figure 3.6.1 can be used with either

R-C TXMOD Low-Pass Filter for Modulation Bandwidth Control

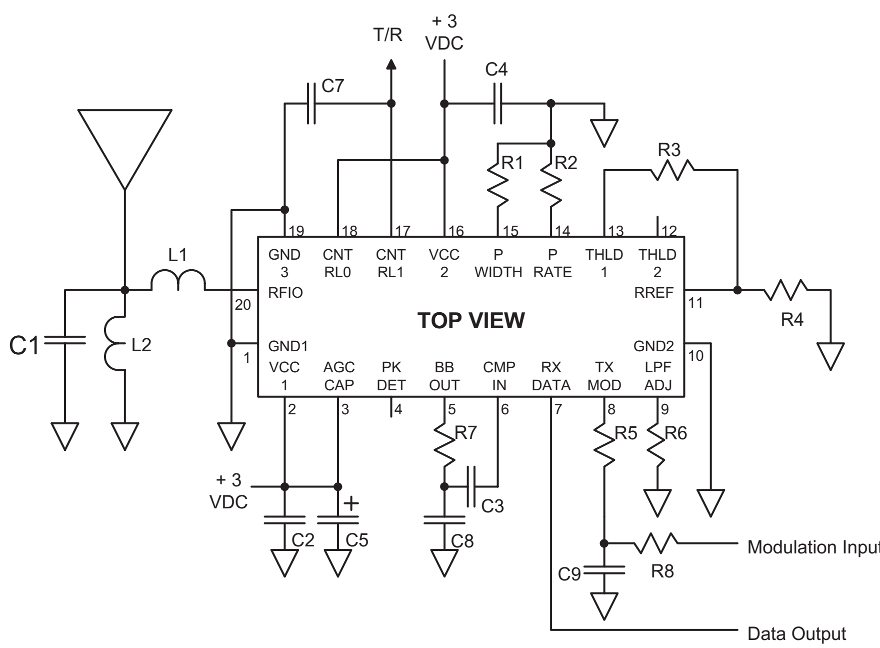


Figure 3.6.1

OOK or ASK modulation. When designing a low-pass filter, note that the dynamic input impedance of the TXMOD Pin is about 350 ohms. This value will vary some with temperature and drive level. For consistent filter behavior, a resistor of several kilohms is used between the capacitor in the low-pass filter and the TXMOD Pin. Table 3.6.1 provides starting-point filter values for a range of data rates. The driving point impedance of the data source will influence the component values used in the low-pass filter. If the driving point impedance is relatively high, the value of C9 in Table 3.6.1 will need to be reduced. Note that the driving point voltage, driving point impedance, and the values of resistors R5 and R8 set the peak TXMOD current. Refer to the individual ASH radio data sheets for recommended peak TXMOD current values.

ASH Radio	Data Rate, bps	SP_{MIN}, µs	R8	C9	R5
TR1001/TX6001	1200	833.3	2.4 K	0.1 µF	2.4 K
TR1001/TX6001	2400	416.7	2.4 K	0.056 µF	2.4 K
TR1001/TX6001	4800	208.3	2.4 K	0.027 µF	2.4 K
TR1001/TX6001	9600	104.2	2.4 K	0.015 µF	2.4 K
TR1001/TX6001	19200	52.1	2.4 K	0.0068 µF	2.4 K
TR3000/TX5000	1200	833.3	4.3 K	0.056 µF	3.9 K
TR3000/TX5000	2400	416.7	4.3 K	0.027 µF	3.9 K
TR3000/TX5000	4800	208.3	4.3 K	0.015 µF	3.9 K
TR3000/TX5000	9600	104.2	4.3 K	0.0068 µF	3.9 K
TR3000/TX5000	19200	52.1	4.3 K	0.0033 µF	3.9 K

Table 3.6.1

For data rates above 20 kbps, a more sophisticated low-pass filter may be required for some ETSI bands (868.00 - 868.60 MHz, etc.), such as the filter shown in Figure 3.6.2. In this example, an active RC filter is used to implement a 4-pole Bessel low-pass filter. The component values given are for a 26.8 kHz 3 dB bandwidth, which is suitable for a 50 kbps data rate. The Bessel transfer function is chosen because of its relatively flat group delay. The ASH radio must be operated in the ASK transmit mode at this data rate.

Active R-C TXMOD Low-Pass Filter,
4 Pole Bessel, 26.6 kHz 3 dB BW,
50 kbps Data Rate

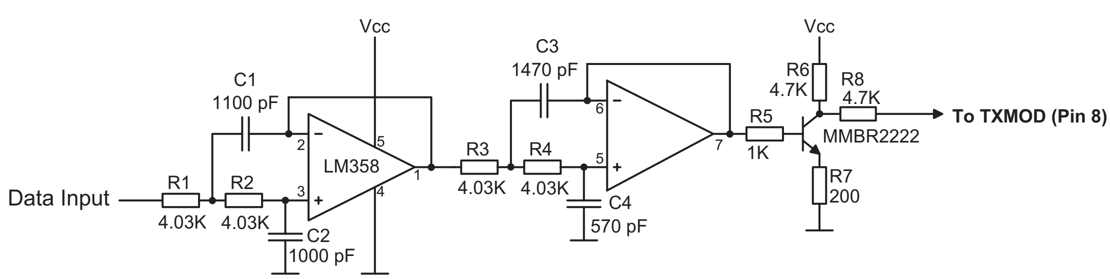


Figure 3.6.2

3.7 ASH Radio RSSI Circuits

A received signal strength indication (RSSI) can be readily derived from Pin 5 of an ASH receiver or transceiver. Under no-signal conditions, the DC value at Pin 5 is about 1.1 volts. When a signal is received, the voltage at Pin 5 increases 10 mV/dB, assuming the PRATE and PWIDTH resistors are set for maximum receiver sensitivity, or for high data rate operation. When DC-balanced data encoding is used, a 5 mV/dB DC change will be observed by low-pass filtering the received data stream at the output of Pin 5. The log detector driving Pin 5 saturates at about 685 mV, providing a 342.5 mV "full scale" DC change at the output of the low-pass filter.

Basic ASH Radio RSSI Circuit

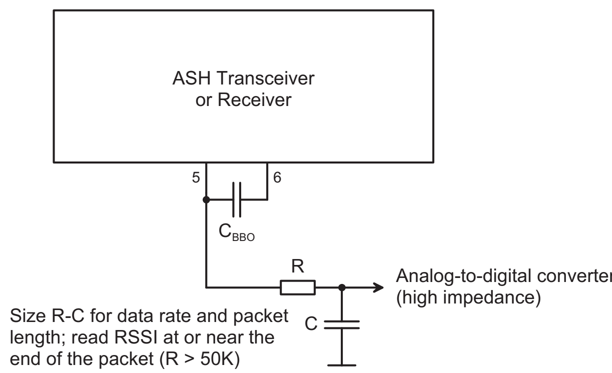


Figure 3.7.1

Figure 3.7.1 shows the basic ASH radio RSSI circuit. The best R-C time constant depends on the data rate, packet length and the analog-to-digital converter (ADC) input impedance. If the ADC input impedance is high and your shortest transmitted packet has at least 100 bits including the C_{BBO} training preamble, start with an R-C time constant 20% to 35% of the transmission time of your shortest transmitted packet. Make the RSSI measurement at or near the end of the packet, so that the DC value at the output of the low-pass filter has maximum time to settle. If you are sending packets shorter than 100 bits, set the R-C time constant for the best trade-off between the residual ripple from the data pattern and the DC transient settling time of the filter. Resistor R should not be less than 50 kilohms, with a value of 100 kilohms to 470 kilohms preferred.

The no-signal DC value at Pin 5 can vary ±250 mV due to unit-to-unit variations, temperature drift and supply voltage drift. When using the circuit in Figure 3.7.1, the RSSI software routine must track the no-signal DC value at Pin 5 for calibration purposes. Unless packet activity is very dense, the no-signal DC value will be the lowest DC value seen at the output of the low-pass filter over several hundred R-C time constants.

Figure 3.7.2 shows an op amp RSSI circuit implementation. The no-signal DC value seen at the cathode of D1 is close to 24.4% of the DC supply voltage. If the supply voltage is regulated, the requirement for the RSSI software to track the no-signal DC value is re-

Op Amp ASH Radio RSSI Circuit

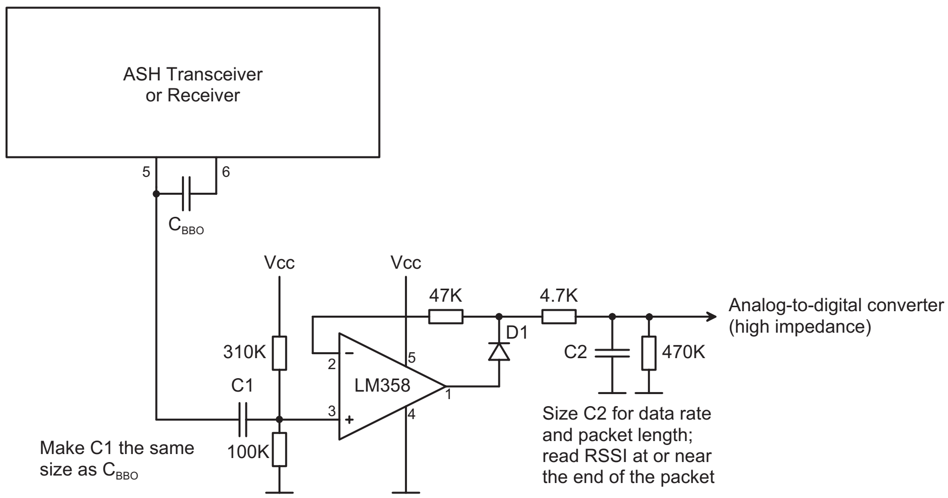


Figure 3.7.2

laxed. The op amp circuit acts as a fast attack/slow decay peak detector. The attack time constant is close to 4.7K*C2, and the decay time constant is close to 470K*C2. Again the RSSI measurement should be made at or near the end of the received packet to allow the transients in the circuit to settle. The decay time constant must be short enough to allow the preamble training transient at C1 to settle before the ADC measurement. This is usually not an issue unless the decay time constant is very large or the packet payload is very small.

Note that if the receiver AGC option is used, the detected signal level at Pin 5 will “fall back” when the AGC engages.

3.8 ASH Radio Performance Curves

Second-generation ASH radios are capable of operating over a supply voltage range of 2.2 to 3.7 Vdc from -40 to +85 ^oC. Typical performance curves for operation from 2.2 to 3.7 Vdc are presented in this section. Individual curves are presented for operation at -40, 0, 25, 70 and 85 ^oC. Note that data sets are presented for second-generation ASH radios operating in the 850 to 950 MHz frequency range (SM-20H package) and in the 300 to 450 MHz frequency range (SM-20L package). Curves are given for receiver sensitivity and receiver current for encoded data rates of 2.4, 19.2 and 115.2 kbps. Transmitter output power for two levels of modulation drive current are also given for each frequency range.

ASH Transceiver Test Circuit
OOK Configuration

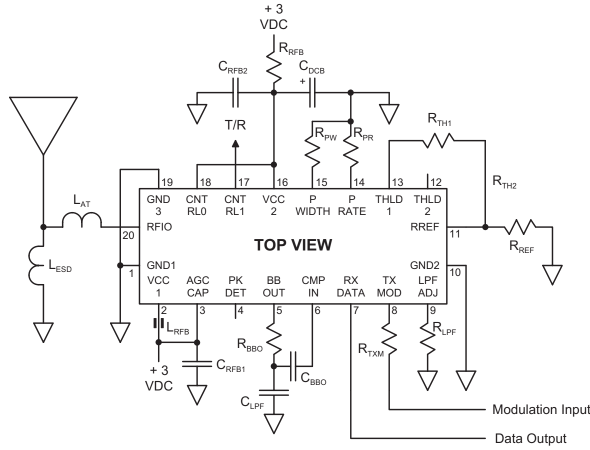


Figure 3.8.1

ASH Transceiver Test Circuit
ASK Configuration

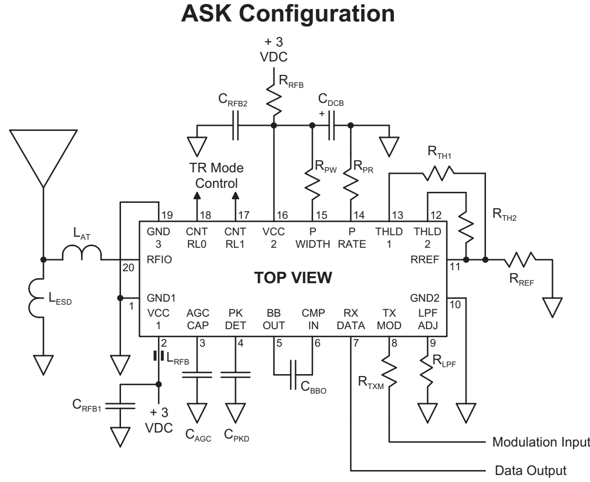


Figure 3.8.2

Refer to Figures 3.8.1 and 3.8.2 plus Table 3.8.1 for details of the test circuits used for characterization. Note that receiver sensitivity is given for a bit error rate (BER) of 10E-3, which is the most commonly used sensitivity reference for short-range radio applications. Two test methods are in common use for making short-range radio sensitivity measurements, the “100% AM” (or 99% AM) test method, and the “OOK Pulse” test method. The “100% AM” test method starts with a unmodulated (CW) signal level and then applies the data to the signal with amplitude modulation. The modulation swings the signal voltage almost ±100% of the CW level. The “OOK Pulse” test method starts with a CW signal level and “chops” the signal with the data stream. The signal voltage swings between the CW level and almost zero. For both test methods a “0” bit swings the signal voltage to almost zero. However, the “100% AM” test method swings the signal voltage to almost twice the CW level for a “1” bit in contrast to the “OOK Pulse” method which sets the signal voltage to just the CW level. For this reason, the “100% AM” test method will make any OOK/ASK receiver look 6 dB more sensitive than the “OOK Pulse” test method. The left scales on the receiver sensitivity plots are for the “100% AM” test method and the right scales on the receiver sensitivity plots are for the “OOK Pulse” test method. The “100% AM test” method is more commonly used because many RF signal generators do not include provisions for pulse modulation.

Receiver current is given for “high sensitivity” receiver operation. When using the low data rate pulse generator set-up, the PRATE resistor R_{PR} can be adjusted to trade-off some receiver sensitivity for reduced current operation. Please see section 2.4.2 for additional information on this topic.

Parameter	Symbol	OOK	OOK	ASK	Units
Encoded Data Rate	DR_{NOM}	2.4	19.2	115.2	kbps
Minimum Signal Pulse	SP_{MIN}	416.67	52.08	8.68	µs
Maximum Signal Pulse	SP_{MAX}	1666.67	208.33	34.72	µs
AGCCAP Capacitor	C_{AGC}	-	-	2200	pF
PKDET Capacitor	C_{PKD}	-	-	0.001	pF
BBOUT Capacitor	C_{BBO}	0.1	0.015	0.0027	µF
BBOUT Resistor	R_{BBO}	12	0	0	K
LPFAUX Capacitor	C_{LPF}	0.0047	-	-	µF
TXMOD Resistor	R_{TXM}	adjusted	adjusted	adjusted	K
LPFADJ Resistor	R_{LPF}	330	100	15	K
RREF Resistor	R_{REF}	100	100	100	K
THLD1 Resistor	R_{TH1}	0	0	10	K
THLD2 Resistor	R_{TH2}	-	-	100	K
PRATE Resistor	R_{PR}	330	330	160	K
PWIDTH Resistor	R_{PW}	270 to GND	270 to GND	1000 to Vcc	K
DC Bypass Capacitor	C_{DCB}	4.7	4.7	4.7	µF
RF Bypass Capacitor 1	C_{RFB1}	27/100*	27/100*	27/100*	pF
RF Bypass Capacitor 2	C_{RFB2}	100	100	100	pF
RF Bypass Bead	L_{RFB}	Fair-Rite	Fair-Rite	Fair-Rite	vendor
Series Tuning Inductor (see table)	L_{AT}	2.3.1.1	2.3.1.1	2.3.1.1	nH
Shunt Tuning/ESD Inductor (see table)	L_{ESD}	2.3.1.1	2.3.1.1	2.3.1.1	nH
* 27 pF for SM-20H, 100 pF for SM-20L					

Table 3.8.1

850 to 950 MHz ASH Radio Receiver Sensitivity
2.4 kbps Data Rate, High Sensitivity Mode

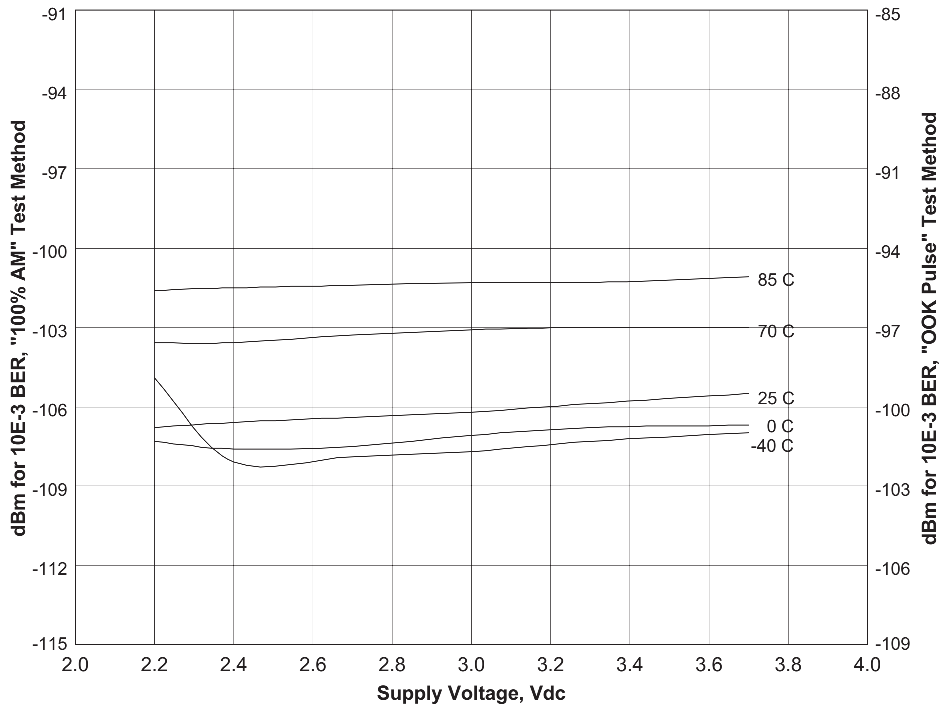


Figure 3.8.3

850 to 950 MHz ASH Radio Receiver Sensitivity
19.2 kbps Data Rate, High Sensitivity Mode

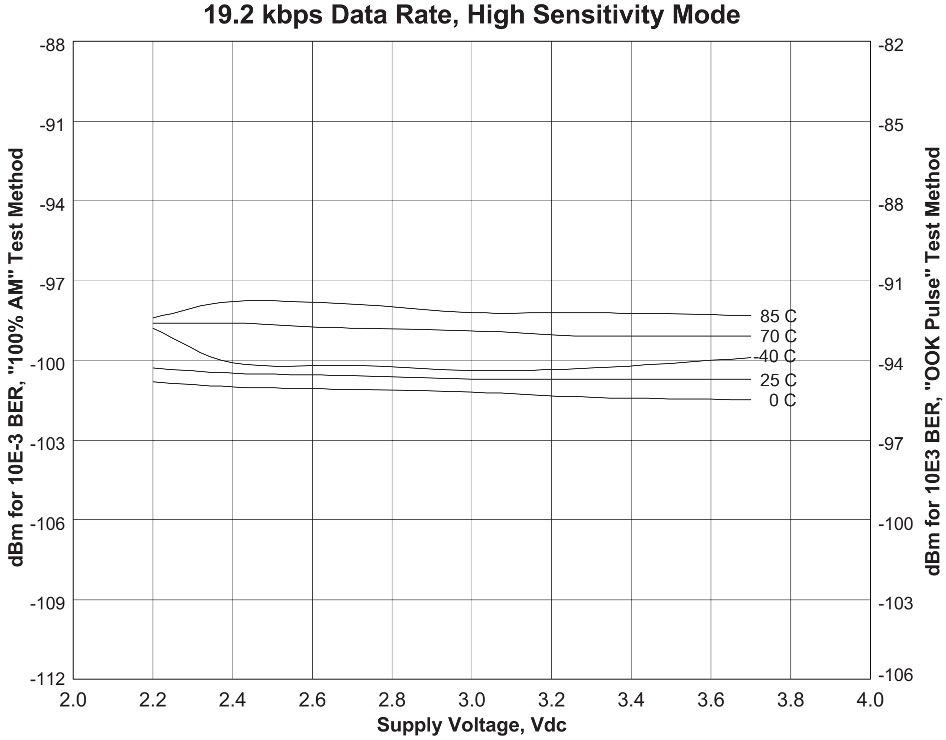


Figure 3.8.4

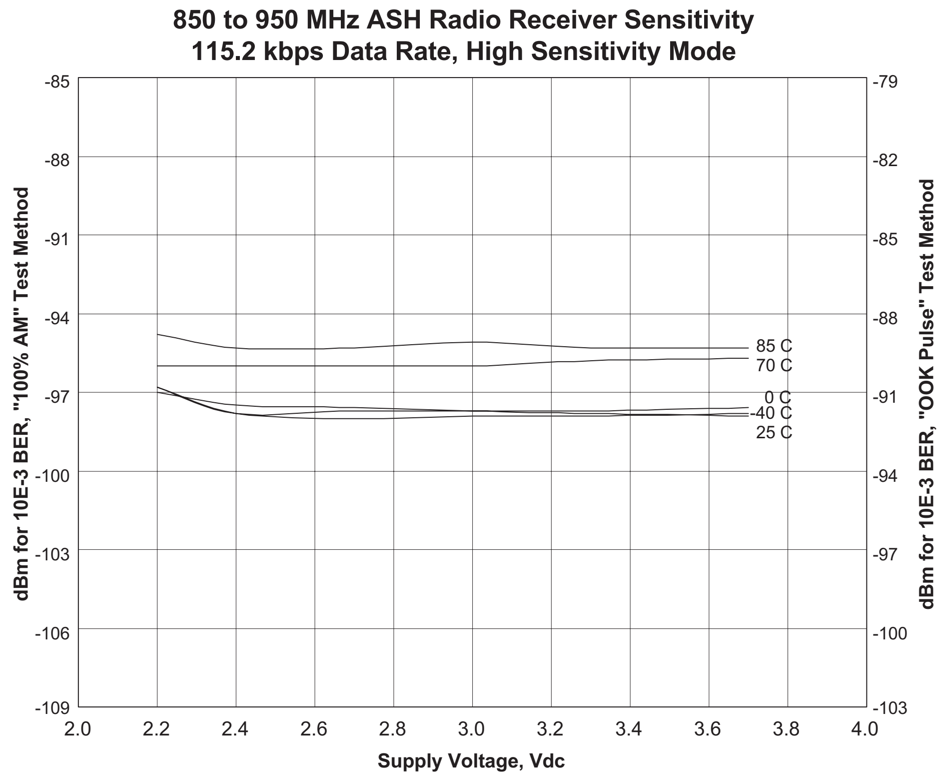


Figure 3.8.5

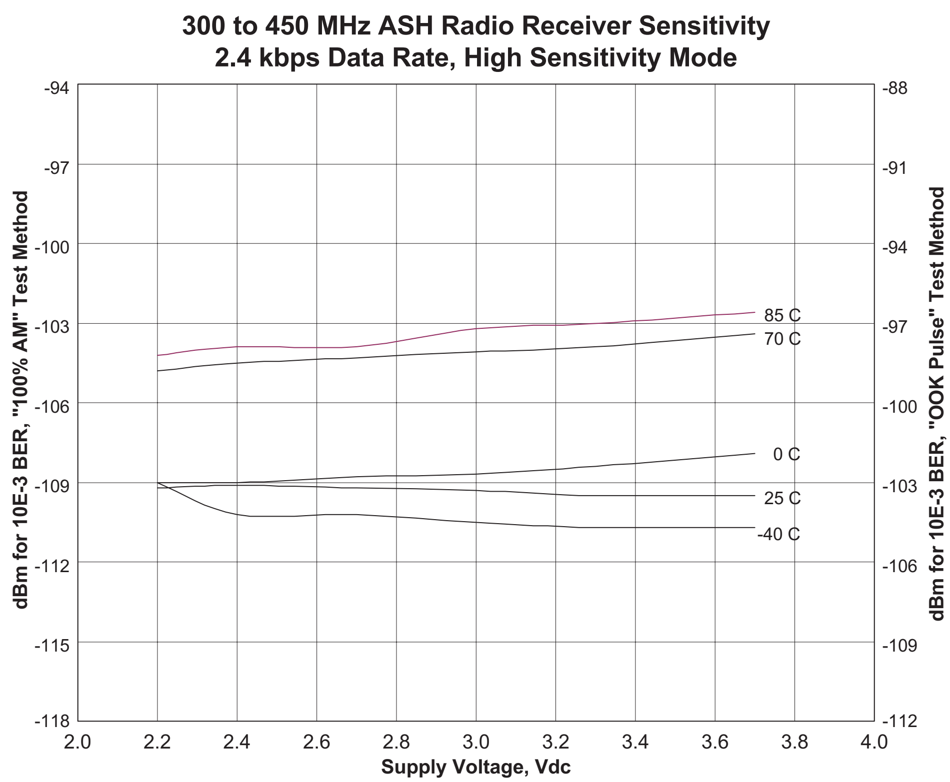


Figure 3.8.6

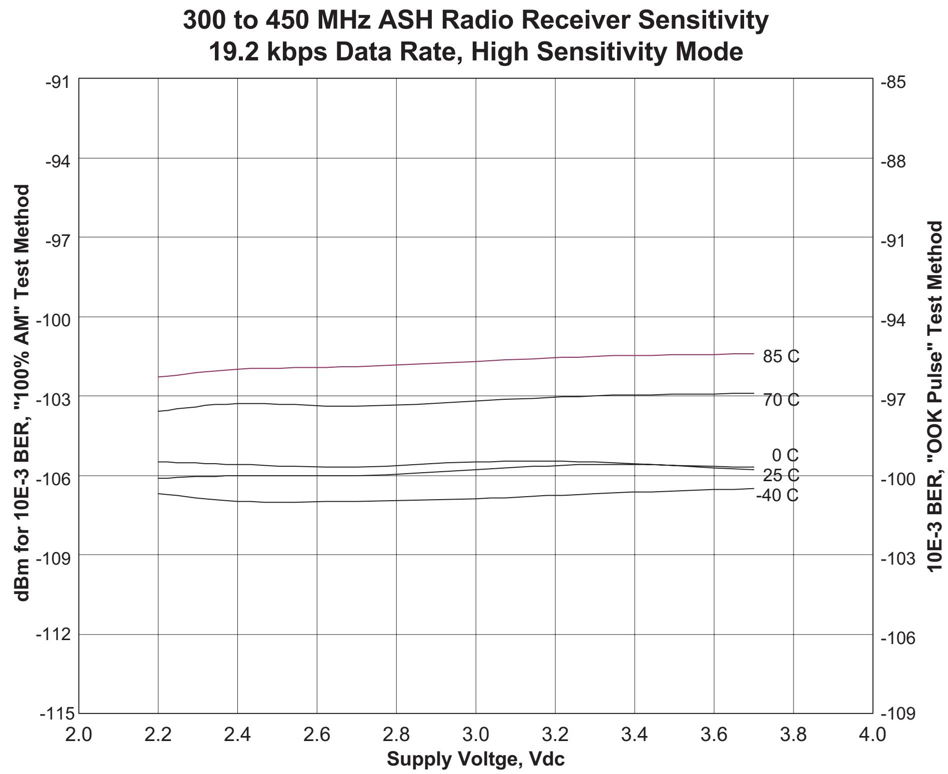


Figure 3.8.7

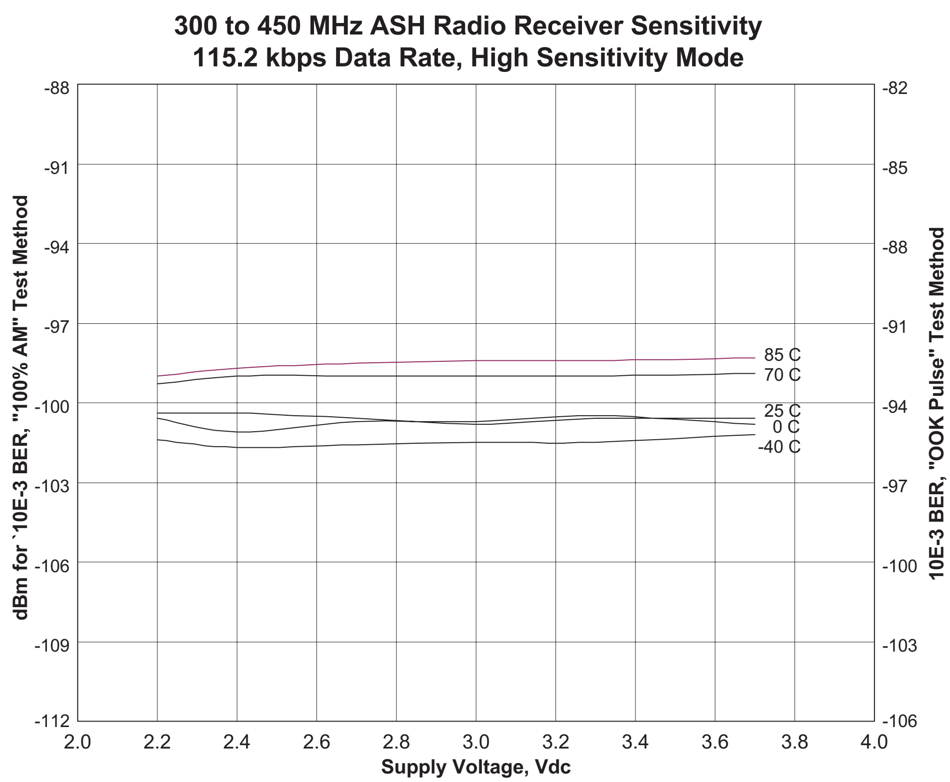


Figure 3.8.8

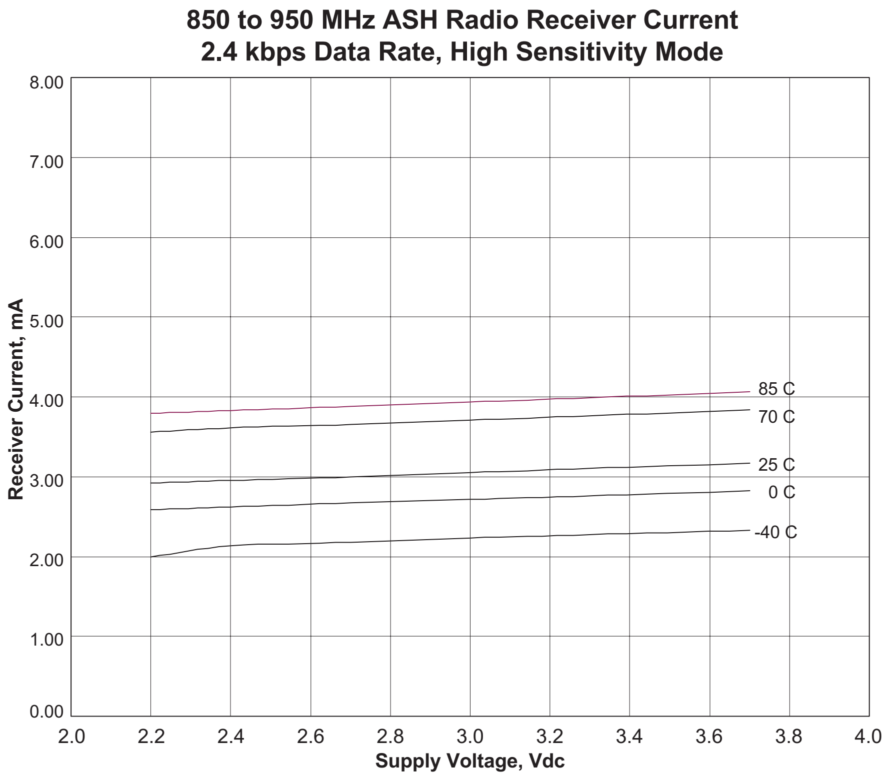


Figure 3.8.9

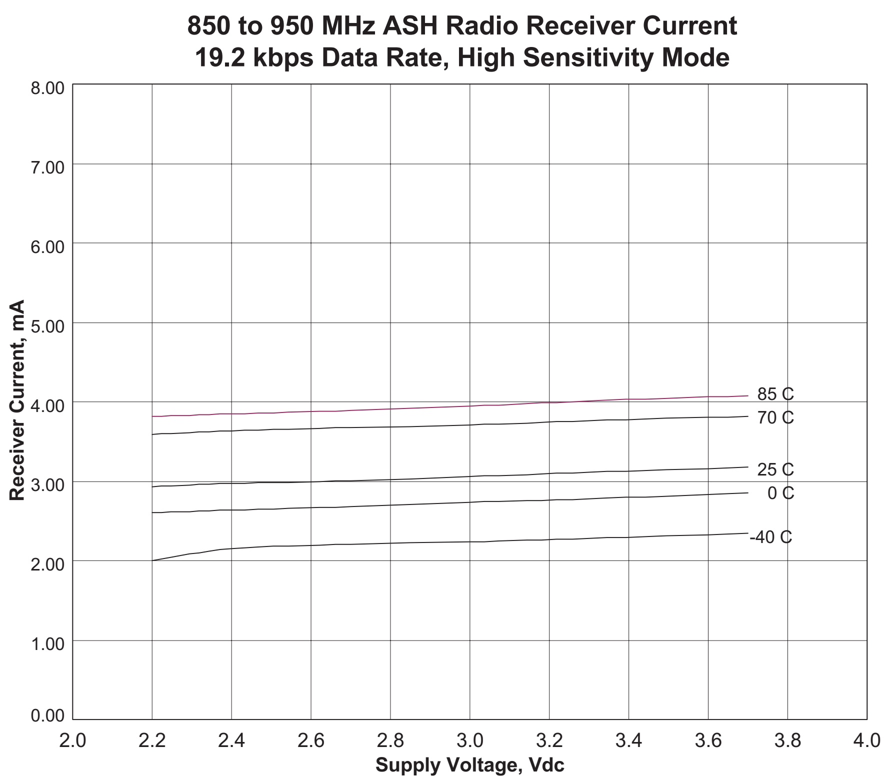


Figure 3.8.10

850 to 950 MHz ASH Radio Receiver Current
115.2 kbps Data Rate, High Sensitivity Mode

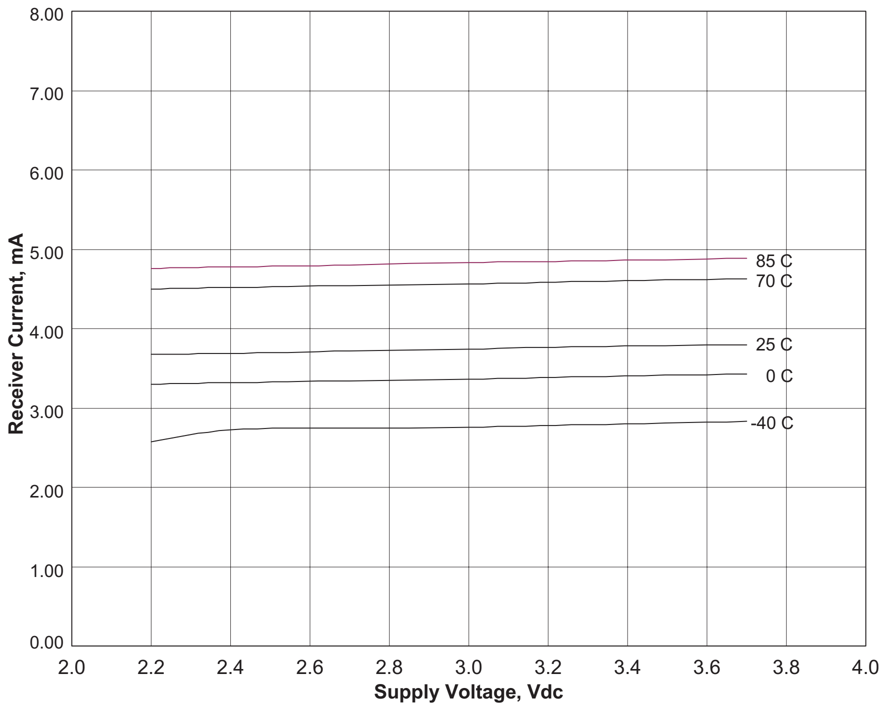


Figure 3.8.11

300 to 450 MHz ASH Radio Receiver Current
2.4 kbps Data Rate, High Sensitivity Mode

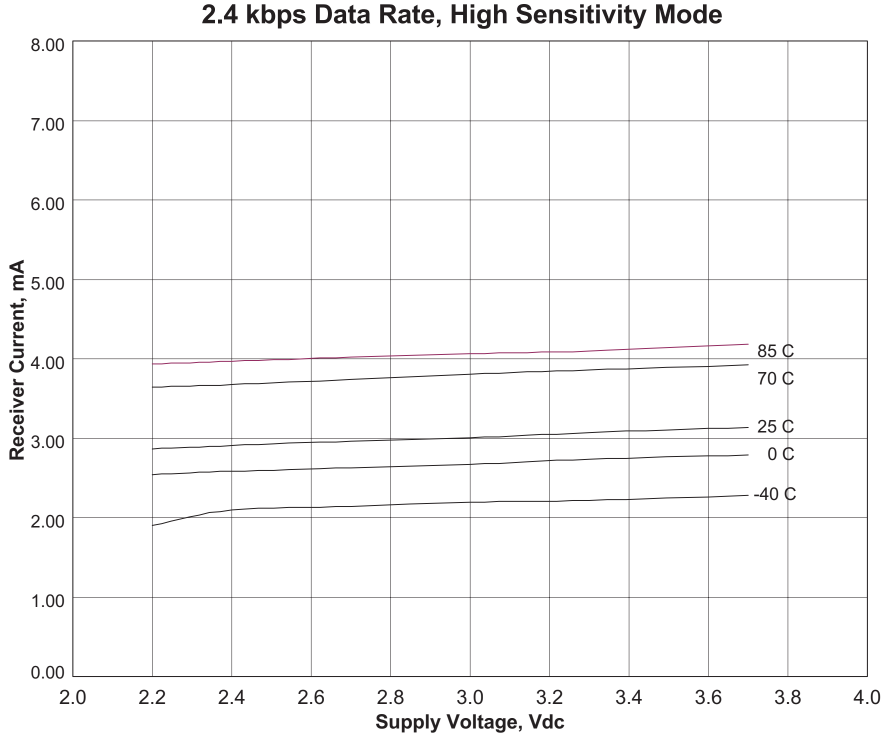


Figure 3.8.12

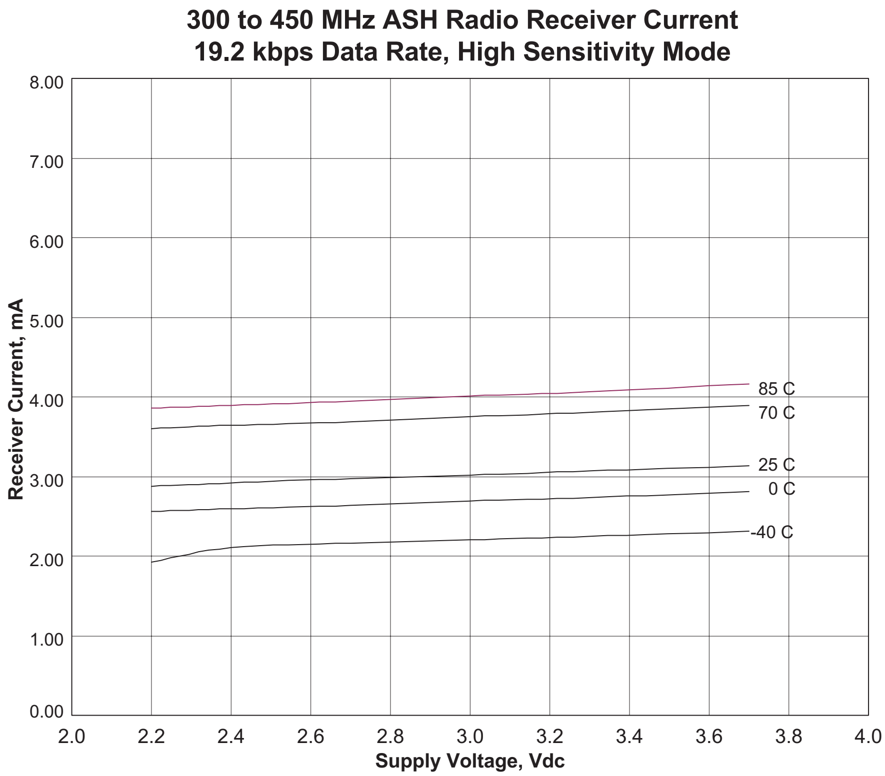


Figure 3.8.13

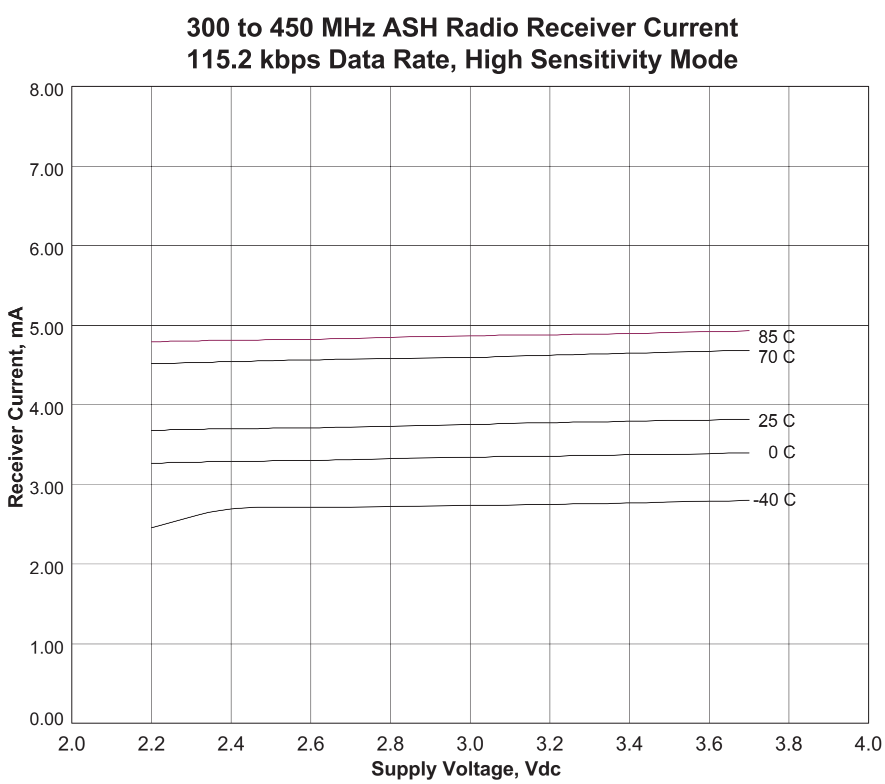


Figure 3.8.14

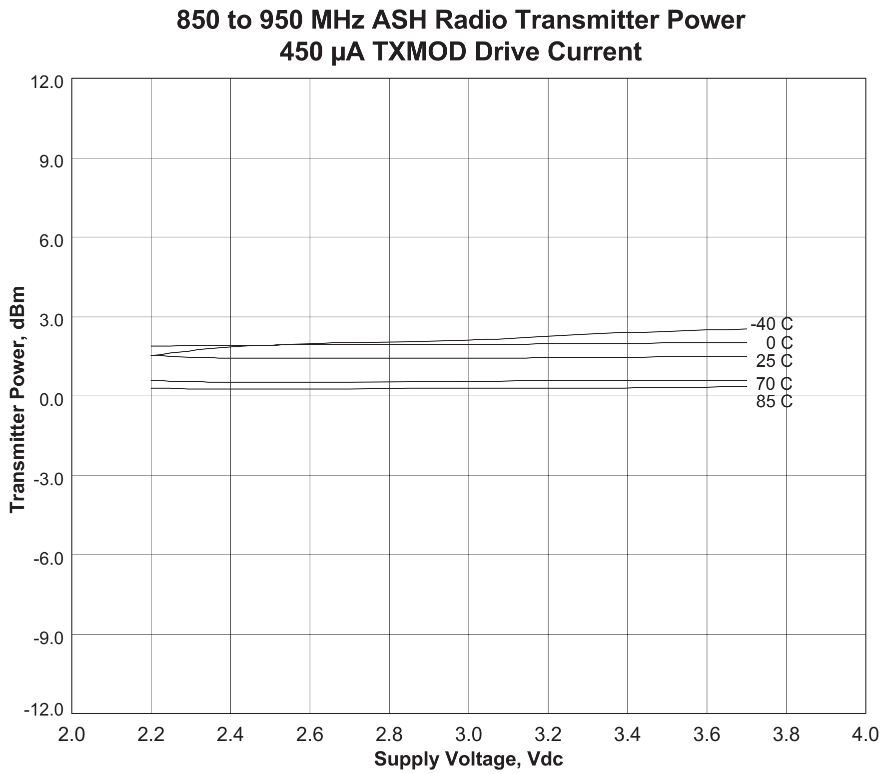


Figure 3.8.15

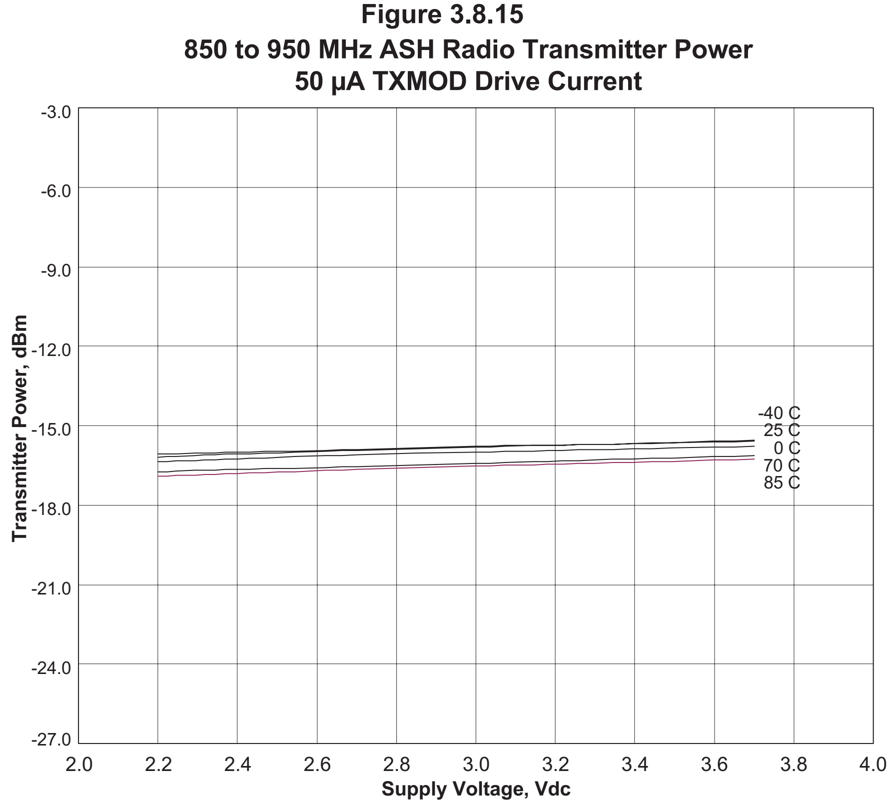


Figure 3.8.16

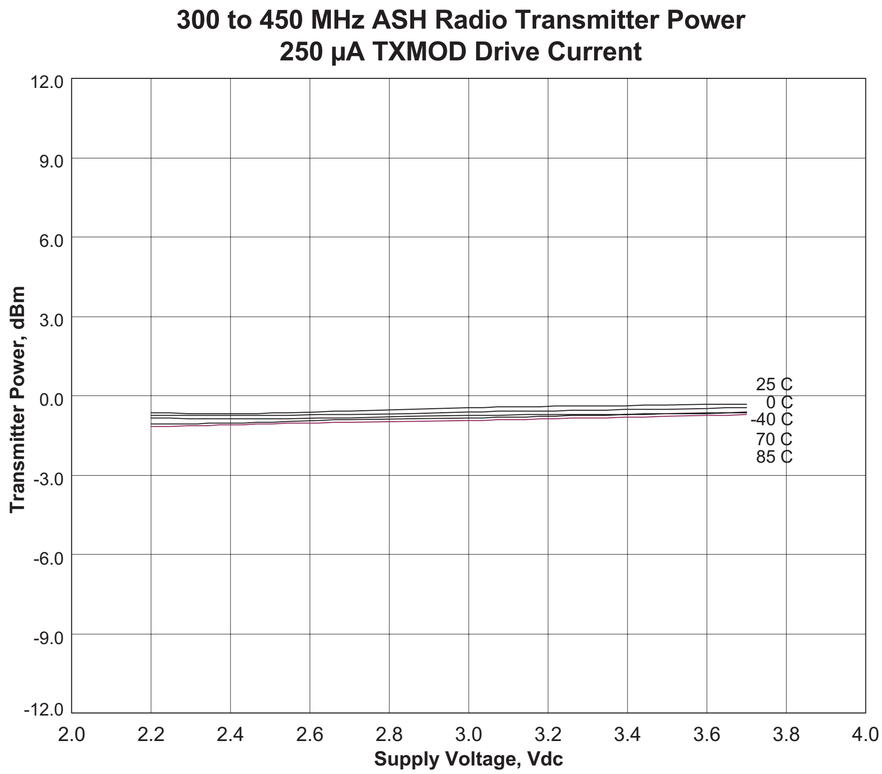


Figure 3.8.17

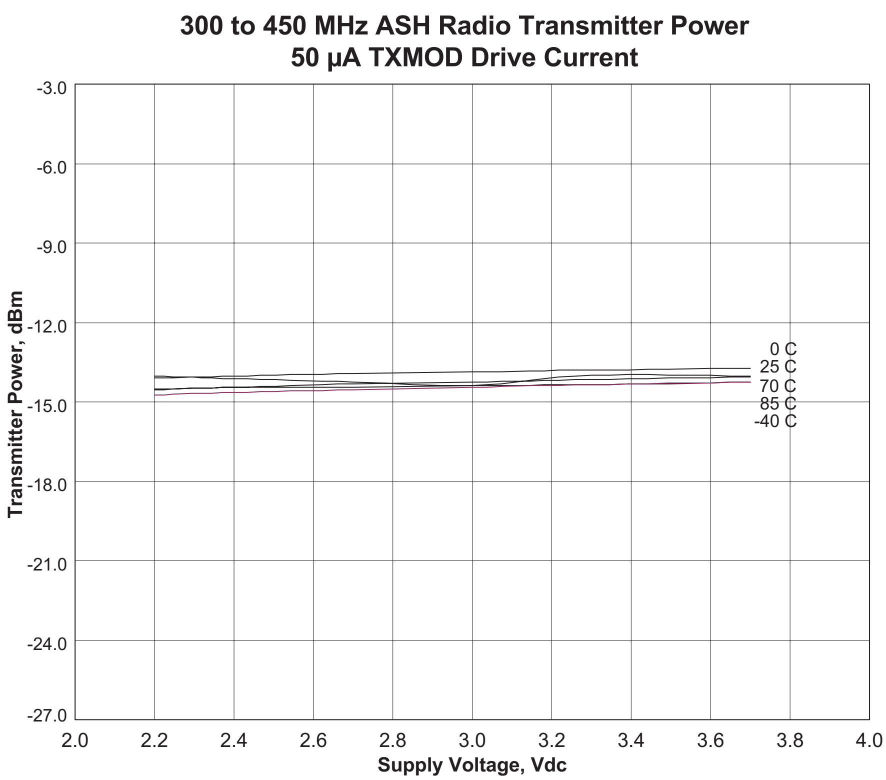


Figure 3.8.18

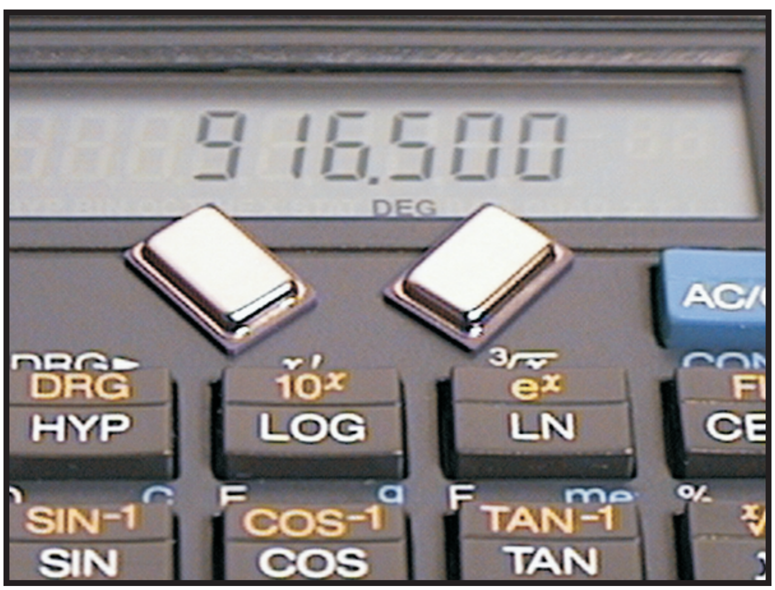
file: tr_des41.vp, 2003.02.22 rev

Senior Design:

Appendix H

ASH Transceiver *Software* Designer's Guide

Updated 2002.08.07



RFM®

ASH Transceiver Software Designer's Guide

Drawings

1 Introduction

1.1 Why Can't I Just Use a UART?

Why can't I just use a UART and a couple of transistors to invert the TX and RX data signals to and from your ASH transceiver and get my application on the air? Well, you can if you don't need maximum performance and you make the necessary provisions in your software for the characteristics of radio communications. But, you are going to leave a lot of performance on the table. A radio link is a type of communication channel, and it has specific properties and characteristics, just as an ordinary phone line is another type of communication channel with its own properties and characteristics. To get usable data communications over your phone line, you place a modem between your PC's UART and the phone line. And to get good performance from your ASH radio link, you are going to need to put something more than a couple of transistors between the UART and the transceiver.

1.2 The Radio Channel – Magic and Imperfect

Radio is magic. It allows commands, data, messages, voice, pictures and other information to be conveyed with no physical or visible connection. A radio wave can penetrate most materials, and it can get around most barriers it cannot directly penetrate. It is arguably the most useful electronic communication channel so far discovered.

But from a software developer's point of view, a radio channel has some aggravating properties and characteristics. The good news is there are strategies for dealing with them.

1.2.1 Modeling a radio system

Figure 1.2.1 is a block diagram of a radio system. The antenna on the transmitter launches energy into the RF channel, and the antenna on the receiver retrieves some of the energy and amplifies it back to a useful level. No big deal, right? Well its no small deal either.

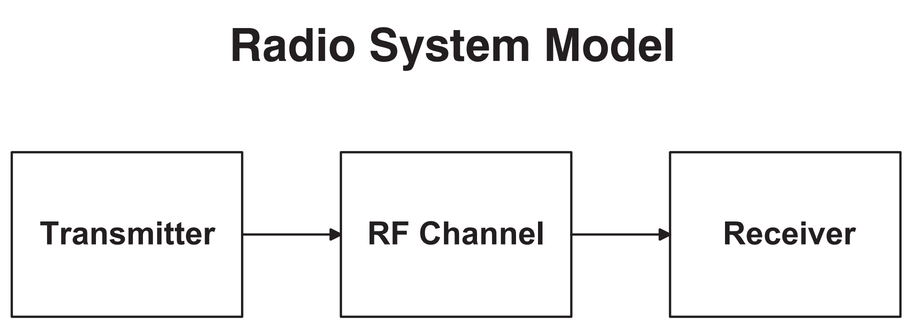


Figure 1.2.1

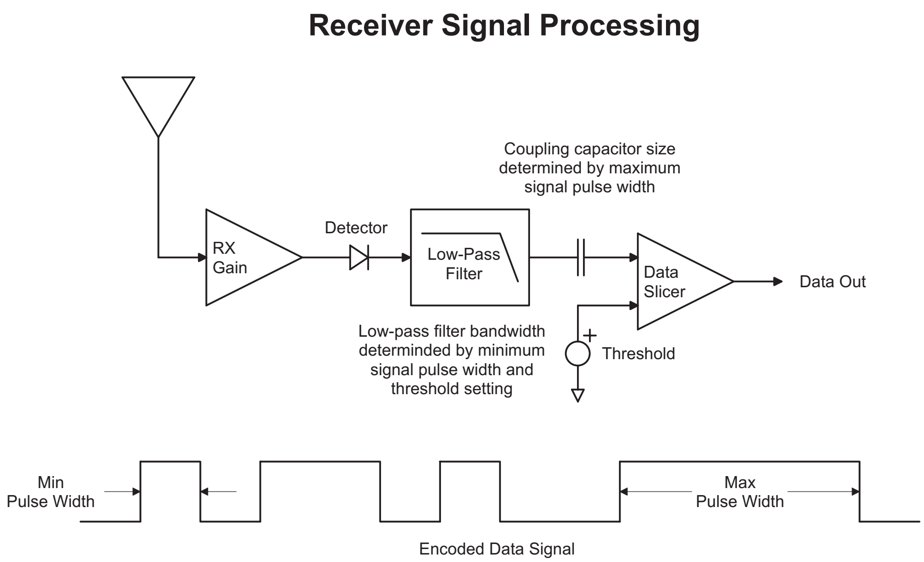


Figure 1.2.2

1.2.2 Data rate and bandwidth

Figure 1.2.2 is a generic block diagram of an RF receiver. This is where most of the action takes place in a radio communication system. There are two filters in this block diagram that you need to know about before you start writing code. The low-pass filter limits the rate that data can be sent through the radio system. And it also has a major impact on the range of the system. As you probably guessed, there is a trade-off here. For a fixed amount of transmitter power, you can transmit farther if you transmit at a lower data rate. The coupling capacitor in the block diagram creates a high-pass filter (in other words, your signal is AC coupled). You have to choose a data rate and use a data encoding scheme that lets your information flow successfully through these two filters. And if you get this right, these filters will greatly contribute to the overall performance of your system.

It is best to think in terms of the most narrow pulse (or most narrow gap) in your encoded signal, which must match the bandwidth of the low-pass filter, and the widest pulse in your encoded signal (or the widest gap), which must correctly match the time constant formed by the coupling capacitor and its associated circuitry. It is the minimum and maximum pulse widths (and gaps) in the encoded data that must be “in tune” with the filters in the receiver – not the underlying data rate.

1.2.3 Noise and interference

Unlicensed radio regulations, such as FCC regulation 15.249, limit the amount of RF power you can transmit to roughly 0.001% of the power dissipated in a 25 watt light bulb. But you only need to capture about 0.00000002% of this transmitted power level to receive properly encoded data at 2000 bps under typical conditions. Using decent antennas chest-high above the ground, this equates to more than one-eighth of a mile of range outdoors and much farther if one or both ends of the system are elevated.

There is a limit on how weak an RF signal can get and still convey information. This limit is due to electrical noise. One source of noise is everywhere present on the surface of the earth and is due to thermally-generated random electrical voltages and currents. Any device with electrical resistance becomes a source of this noise. Two other noise contributors are important in RF communications – semiconductor noise and attenuation. Semiconductor devices such as RF amplifiers contain noise generation mechanisms in addition to resistive thermal noise. Also, any component that attenuates a signal and is a thermal noise generator itself reduces the signal-to-noise ratio by the amount of the attenuation. An RF filter is an example of this type of component.

A signal transmitted through a radio system will be at its lowest power level when it reaches the first amplifier stage in the receiver. The noise added to the signal at this point places an upper limit on the signal-to-noise ratio that can be achieved by the receiver (for a given low-pass filter bandwidth). A good antenna helps keep the signal-to-noise ratio up by delivering more signal power. In addition, using a low-loss RF filter between the antenna and the first amplifier helps keep the signal-to-noise ratio up by minimizing signal attenuation. Using RF IC technology with low inherent RF semiconductor noise minimizes the amount of noise that is added to the signal beyond the ever-present resistive thermal noise. And yes, there are software tricks to take maximum advantage of whatever signal-to-noise ratio the hardware guys get for you.

Figure 1.2.3.1 shows the probability distribution, or histogram, of the noise voltage you would see at the base-band output of the ASH transceiver (R_{LPF} = 330 K). Notice that the noise has a Gaussian probability distribution. About 70% of the time the noise voltage will be between ±9 mV, or one standard deviation. Occasionally, noise spikes will reach ±18 mV, or two standard deviations. On rare occasions, spikes will reach ±27 mV, and on very rare occasions noise spikes will reach ±36 mV or more. So every now and then a noise spike or "pop" will occur that is strong enough to corrupt even a strong received signal. This characteristic of thermal noise (and thermal-like semiconductor noise) means that no RF channel can be perfectly error free. You have to plan for data transmission errors when designing your software.

From DC to frequencies much higher than RF, thermal noise exhibits a flat power spectrum. The power spectrum of semiconductor noise can also be considered flat across the RF bandwidth of a typical receiver. If you halve the bandwidth of the low-pass filter in a receiver, you halve the thermal noise power that comes through it. This is why you can transmit longer distances at a lower data rate. It allows you to reduce the bandwidth of

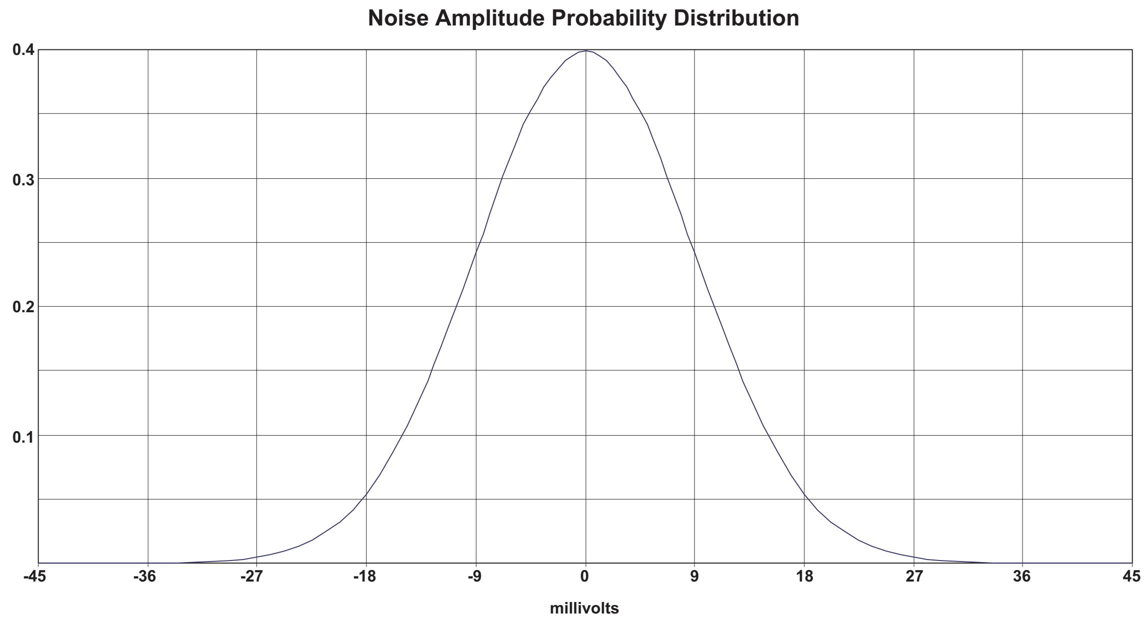


Figure 1.2.3.1

the low-pass filter so less noise gets through. You can then successfully recover data from a weaker received signal.

Lets go back and look at Figure 1.2.2 again. The job of the data slicer is to convert the signal that comes through the low-pass filter and coupling capacitor back into a data stream. And when everything is set up properly, the data slicer will output almost perfect data from an input signal distorted with so much noise that it is hard to tell there is a signal there at all. For the time being, assume the threshold voltage to the data slicer is zero. In this case, anytime the signal applied to the data slicer is zero volts or less, the data slicer will output a logic 0. Anytime the signal is greater than zero volts, the data slicer will output a logic 1. Through software techniques, you can assure that the signal reaching the data slicer swings symmetrically about 0 volts. Noise spikes, either positive or negative, that are slightly less than one half of the peak-to-peak voltage of the desired signal will not appear as spikes in the data output. The ability to recover almost perfect data from a signal with a lot of added noise is one of the main reasons that digital has overtaken analog as the primary format for transmitting information.

In the way of a preview, look at Figures 1.2.3.2, 1.2.3.3, 1.2.3.4 and 1.2.3.5, which are simulations of a radio system with various amounts of noise added to the signal. The top trace in Figure 1.2.3.2 is the signal seen at the input to the data slicer.

The horizontal line through this signal is the slicing level. Notice that the signal droops down as it starts from left to right, so that is swinging symmetrically around the slicing level by about the fifth vertical grid line. This is the transient response of the base-band coupling capacitor, and its associated circuitry, as it starts blocking the DC component of the received signal. The steady 1-0-1-0… bit pattern seen to the left of the fifth grid line is a training preamble. It sets up the slicing symmetry. To the right of the fifth grid line there is a 12 bit start symbol and then the encoded message bits, etc. You will notice that

Signal Reception with No Noise

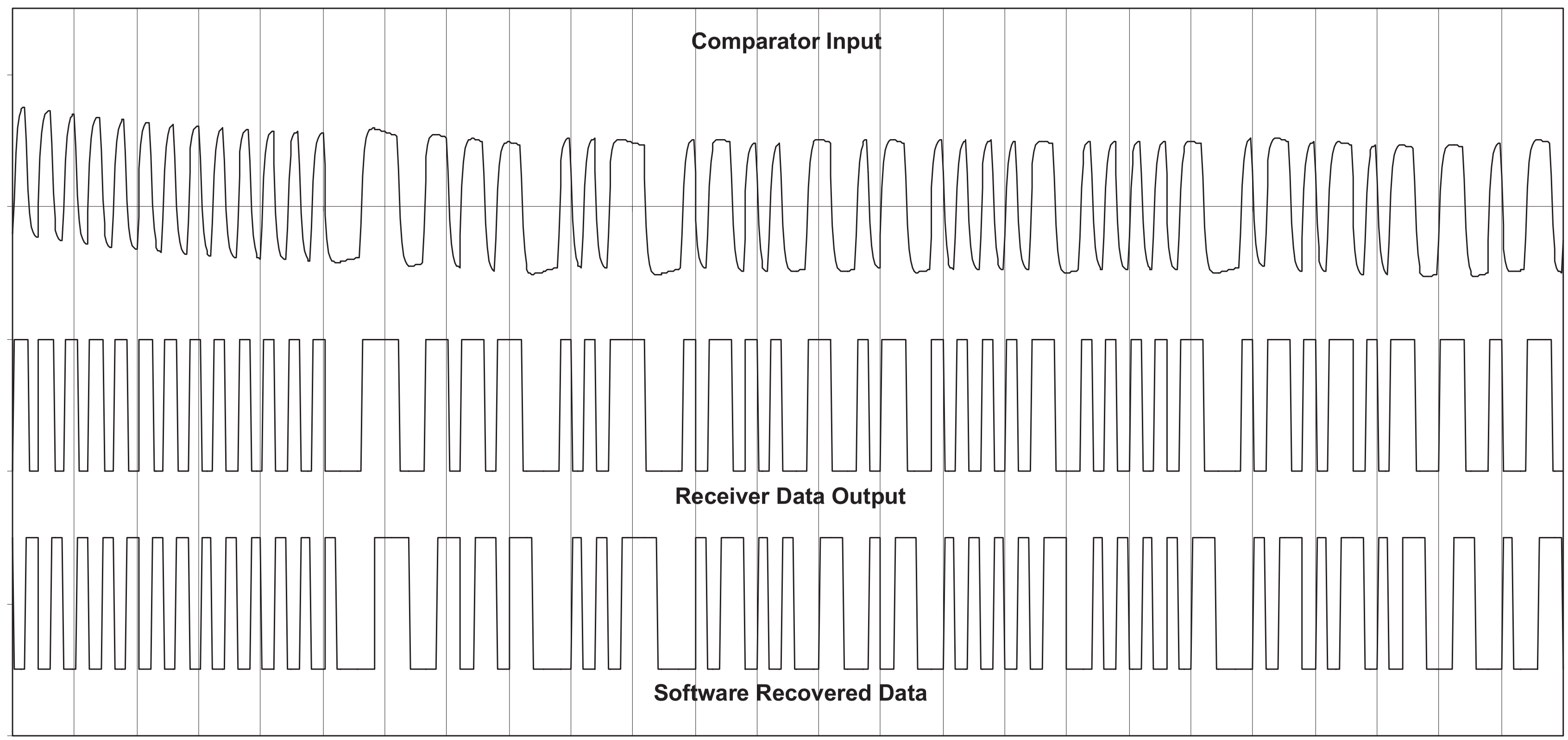


Figure 1.2.3.2

Signal Reception with Moderate Noise

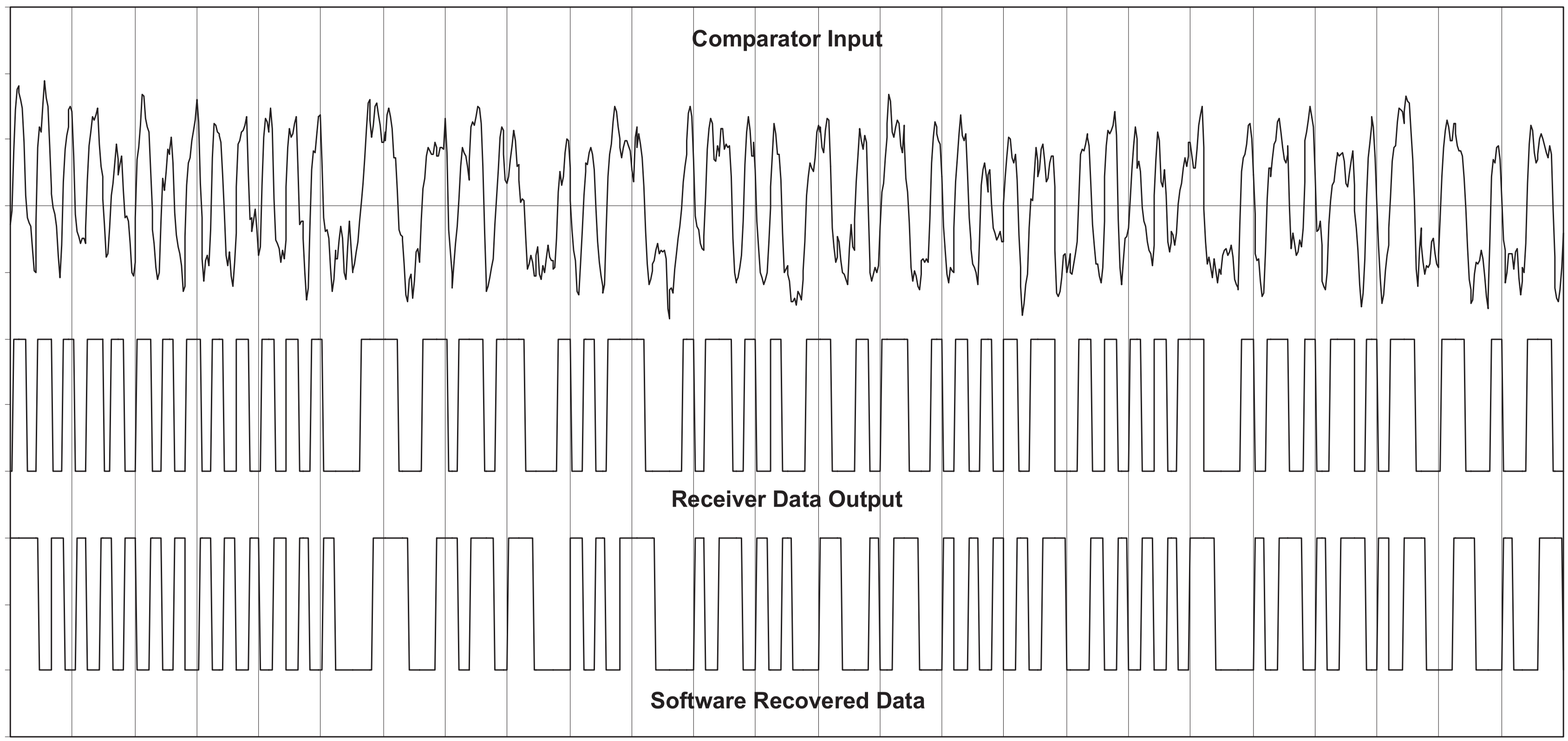


Figure 1.2.3.3

Signal Reception with Heavy Noise

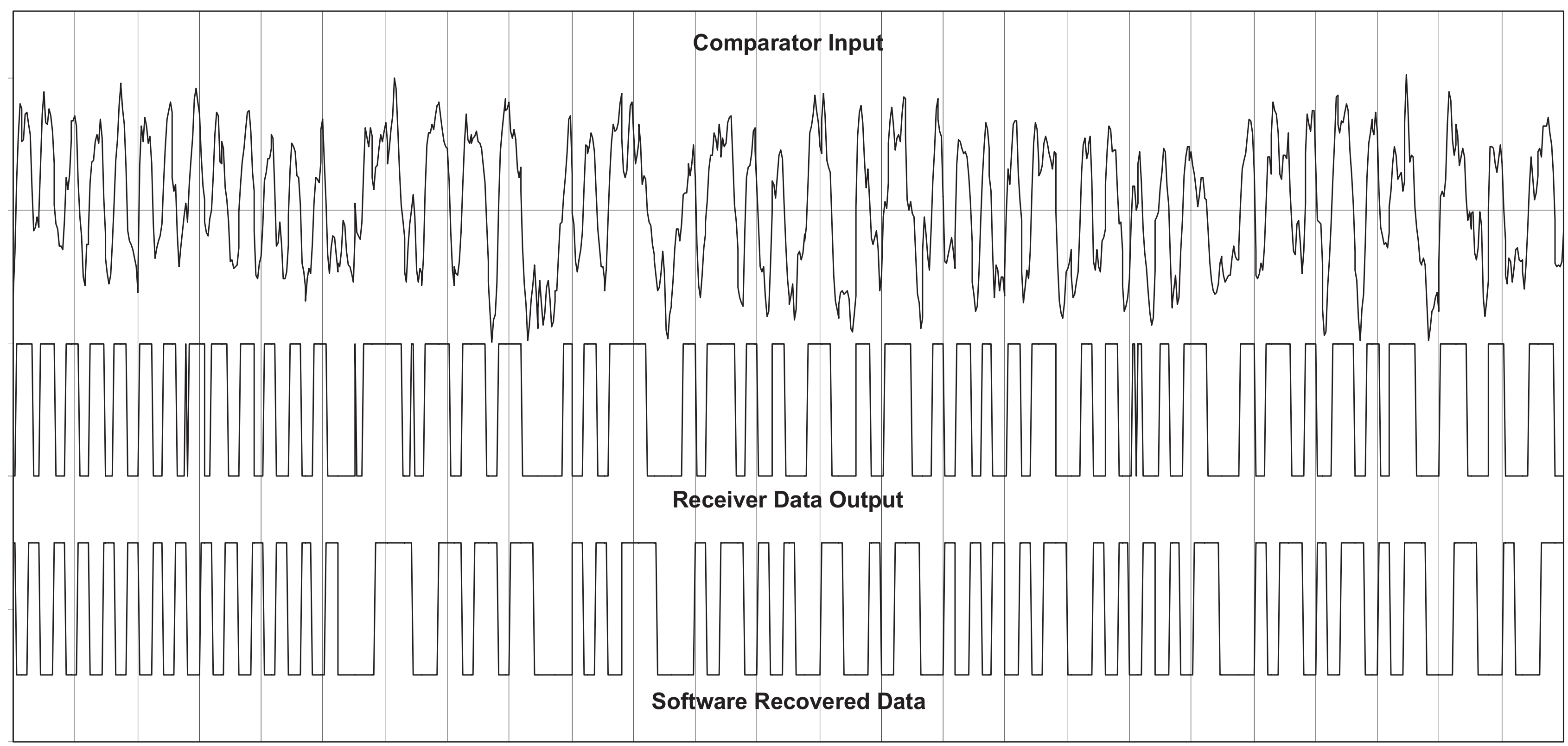


Figure 1.2.3.4

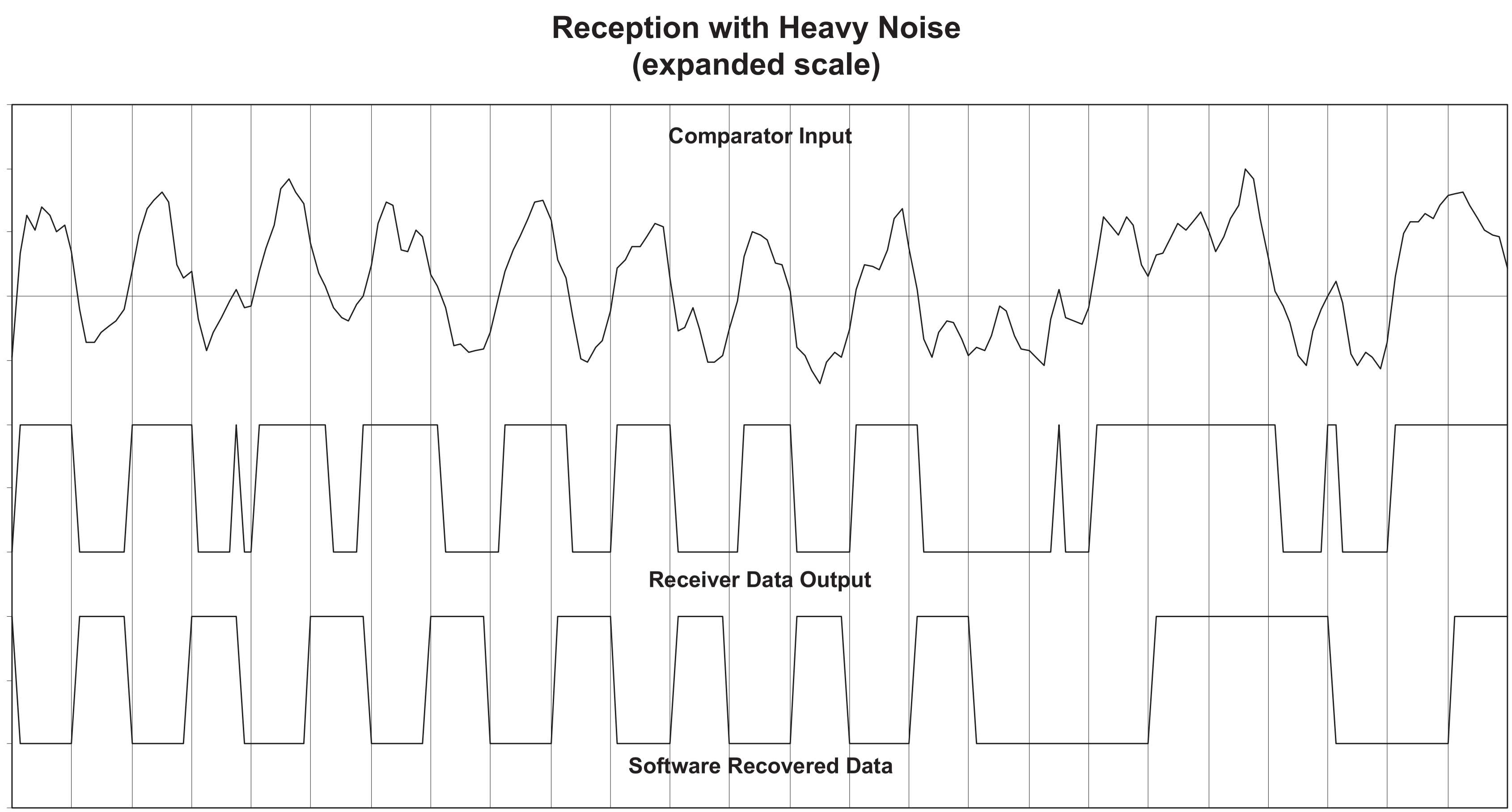

Reception with Heavy Noise
(expanded scale)
Comparator Input
Receiver Data Output
Software Recovered Data

Figure 1.2.3.5

the signal has been “rounded off” so that the 1-0-1-0… bit sequences almost look sinusoidal. This shaping effect is due to the low-pass filter. If you set the bandwidth of the filter too low for a given data rate, it will start seriously reducing the amplitude of these 1-0-1-0… bit sequences and/or smearing them into each other.

The output of the data slicer is the middle trace, and the output of the software recovery subroutine is the bottom trace. Notice that the bottom trace is shifted to the right one bit period. This is because the software “studies” the receiver data output for a complete bit period before estimating the bit value. It will soon become apparent why this is done.

Figure 1.2.3.3 shows the same signal with a moderate amount of noise added. You now have to look at the top trace carefully to see the data pattern (look right at the slicing level). The middle trace shows the output of the data slicer, which has recovered the data accurately other than for some jitter in the width of the bits. The data recovered by the software matches the middle trace again, shifted one bit period to the right.

Figure 1.2.3.4 shows the signal with heavy noise added. The data pattern has become even more obscure in the top trace. With this much noise, the output from the data slicer shows occasional errors. Note that the software subroutine has been able to overcome these errors by deciding the most likely bit value at the end of each bit period. Figure 1.2.3.5 is a section of 1.2.3.4 on an expanded scale to show more bit-by-bit detail.

Interference is defined as an unwanted RF signal radiated by another system (RF or digital). Like noise, interference that is not too strong can be eliminated by the data slicer and/or software subroutine. Of course, the data has to be encoded so that it swings symmetrically around the slicing level to get maximum noise and interference rejection.

1.2.4 Indoor RF propagation

It is intuitive that the farther away from a transmitter you get, the less power you can capture from it with your receiver. This is what you would see in free space, far away from the ground and other physical objects. But on the ground, and especially indoors, you will find that the signal strength varies up and down rapidly as the distance between the transmitter and the receiver is steadily increased. The reason this happens is both good news and bad news. It turns out that the radio waves from the transmitter antenna are taking many different paths to the receiver antenna. Radio waves strongly reflect off the ground and off metal surfaces as light reflects off a mirror. And radio waves will also partially reflect off non-metallic walls, etc. as light does off a window pane. The good news is that all this bouncing around allows radio waves to diffuse around barriers they cannot directly penetrate. The bad news is that all the bouncing around makes the RF power you receive vary rapidly (flutter) as you move around and hit small reception “dead spots”. You can even see reception flutter if you stand still and other people, vehicles, etc. move nearby. Any radio system that operates near the ground (mobile phones, wireless microphones, broadcast radios in cars, etc.) must deal with this multi-path flutter problem. And yes, it is a consideration when you start writing your code.

Studies on indoor propagation show that you will find only a few spots in a room that have really bad reception, and these severe “dead spots” tend to occupy a very small space. Mild dead spots are far more common, and you will also find some places where reception is especially good. As a rule of thumb, you need 100 times more transmitted power indoors than in free space to get adequate reception at comparable distances. This is called a 20 dB fading margin, and it provides about 99% coverage indoors. If you are in a severe dead spot at UHF frequencies, moving just an inch or two gets you out of it.

When you look at a professional wireless microphone, you will notice that the base unit is equipped with a “rabbit ear” antenna. Actually, there are two separate antennas and two separate receivers in the wireless microphone base unit, with the antennas at right angles to each other. This arrangement provides diversity reception, which greatly mitigates the dead spot problem indoors. Since the paths between the two base station antennas and the microphone are different, it is unlikely that the microphone will hit a dead spot for both antennas at the same time. Mobile phone base stations also use diversity reception as do many other radio systems, including a number of ASH transceiver systems.

1.2.5 Regulatory considerations

Systems based on ASH transceiver technology operate under various low power, unlicensed UHF radio regulations. From a software point of view, the main differences in these regulations are the maximum power you are allowed to transmit, and the allowed transmitter duty cycle. European regulations (ETSI) allow the most transmitted power, American regulations are in the middle, and Japan allows the least transmitted power. At lower power levels, you have to transmit at a low data rate to get a useful amount of range. At higher power levels you have more flexibility.

Duty cycle refers to the percentage of time each transmitter in your system can be on. Some regulations, such as FCC 15.249 place no restrictions on duty cycle. Some bands in Europe also have no current duty cycle limit - for example, the 433.92 MHz band. Other bands in Europe do have a duty cycle limit. At 868.35 MHz, the duty cycle limit is 36 seconds in any 60 minute interval. Duty cycle requirements influence the choice of band to operate in, and the design of your software. RFM’s web site has links to many radio regulatory sites. Be sure to thoroughly familiarize yourself with the regulations in each geographical market for your product. We have seen cases where a customer had to redo a well-engineered system to accommodate a regulatory subtlety.

2 Key Software Design Issues

There are at least four key issues to consider in designing ASH transceiver software. You may identify others depending on the specifics of your product’s application. It is worth giving it some thought before you start designing your code.

2.1 Fail-Safe System Design

Most unlicensed UHF radio systems operate with few interference problems. However, these systems operate on shared radio channels, so interference can occur at any time and at any place. Products that incorporate unlicensed UHF radio technology must be designed so that *a loss of communications due to radio interference or any other reason will not create a dangerous situation, damage equipment or property, or cause loss of valuable data*. The single most important consideration in designing a product that uses unlicensed radio technology is safety.

2.2 Message Encoding for Robust RF Transmission

Look at Figure 1.2.2 again, and note the threshold input to the data slicer. When you set the threshold voltage to a value greater than zero you move the slicing level up. This provides a noise squelching action. Compare Figures 2.2.1 and 2.2.2. In Figure 2.2.1, the threshold is set to zero. With no signal present, noise is continuously present at the receiver data output, and at the output of the software data recovery routine. Software downstream of the data recovery subroutine has to be able to distinguish between noise and a desired signal. Figure 2.2.2 shows the effect of adding a moderate threshold. Notice that just a few noise spikes appear at the receiver data output and no noise spikes come out of the software data recovery routine (it could still happen occasionally). As we raise the threshold more, even fewer noise spikes will appear at the receiver data output. Don't expect to eliminate all noise spikes – noise amplitude has that Gaussian probability distribution we discussed earlier. Even using a very heavy threshold, you have to plan for noise spikes now and then, as well as strong bursts of interference.

As you raise the threshold from zero, you reduce the receiver's sensitivity to desired signals, and you make it more vulnerable to propagation flutter. If you need all the range and system robustness possible, you will want to use little or no threshold. On the other hand, using a threshold can reduce the amount of work your software has to do on data recovery. This allows you to support a higher data rate with the same processing power, or reduce average processor current consumption in applications where this is critical. If you decide to use an ordinary UART on the radio side, a strong threshold is a must. Also, some remote control decoder chips will not tolerate much noise.

The ASH transceiver is equipped with two thresholds, DS1 and DS2. DS1 works basically as shown in Figures 1.2.2, 2.2.1, and 2.2.2. DS2 is used in conjunction with DS1 and its primary job is to support high data rate transmissions. The details on how to adjust these thresholds are given in the ASH Transceiver Designer's Guide, Sections 2.7.1 and 2.7.2.

Your message encoding strategy and several adjustments on the ASH transceiver depend on whether you use a threshold, and on how strongly the threshold is set. Let's start with the "no threshold" case, which offers the best potential performance. Referring to Figure 1.2.3.2, we start the transmission with a 1-0-1-0… training preamble. This preamble needs to be long enough to establish good signal slicing symmetry at the input to the

Noise Reception with No Signal and No Threshold

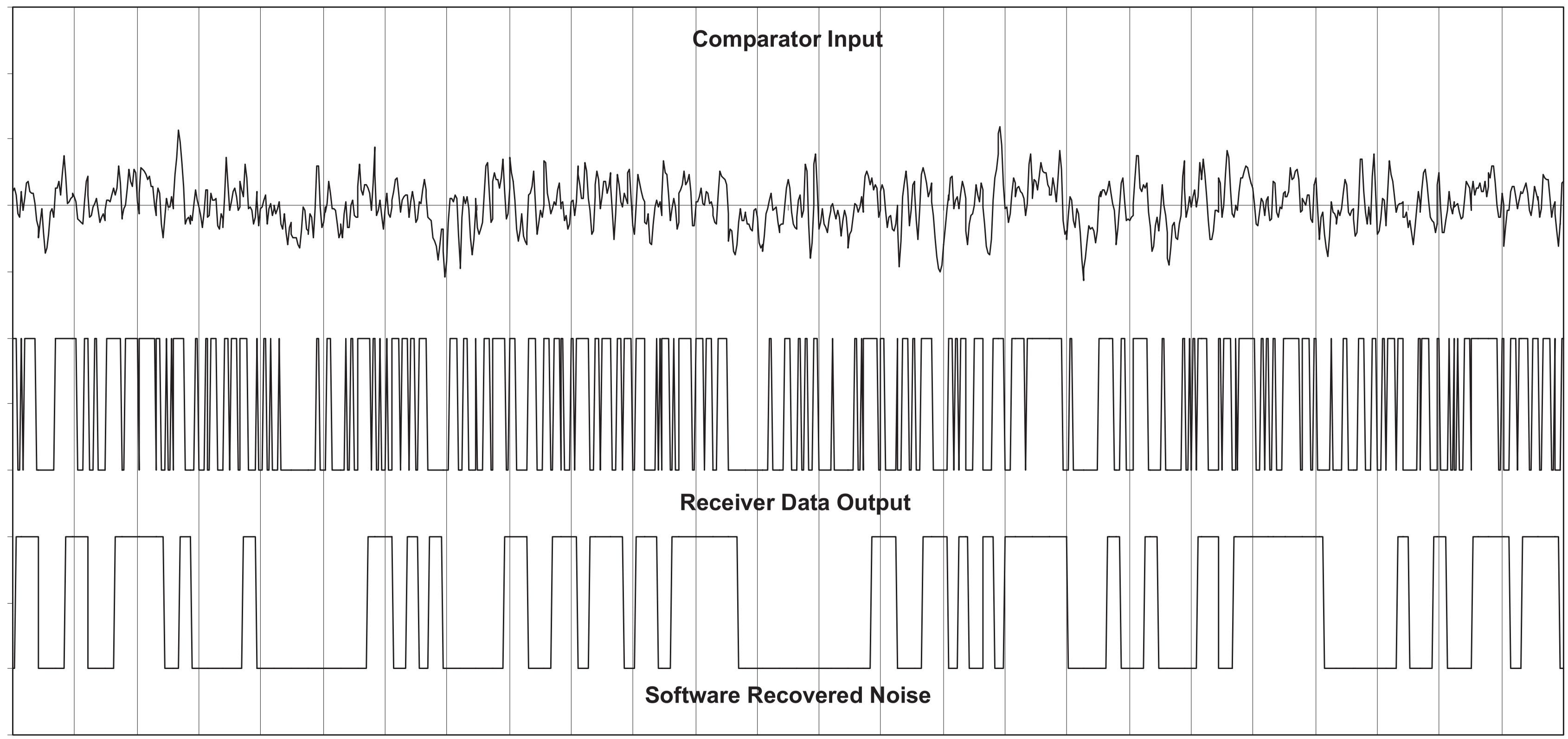


Figure 2.2.1

Noise Reception with No Signal and Moderate Threshold

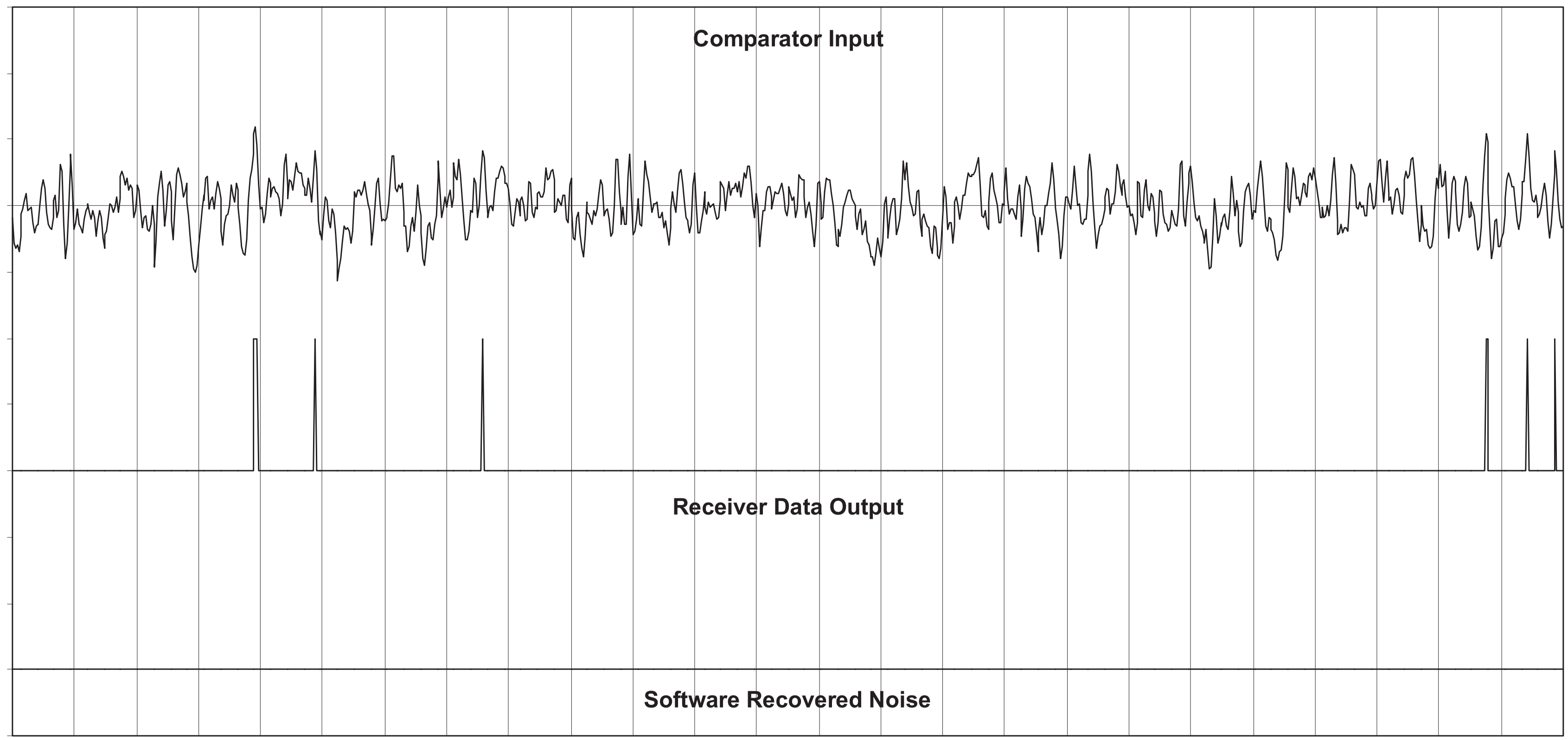


Figure 2.2.2

comparator. The preamble is followed by a specific pattern of bits that will not occur anywhere else in the message. This pattern is often called a “sync vector”, and makes it possible to distinguish data from noise with high reliability (the sync vector is 12 bits in this example). The balance of the message consists of encoded data and error detection bits.

The purpose of encoding your data is to maintain good slicing symmetry at the input to the comparator. This is called DC-balanced encoding. Look at Figure 1.2.3.2 again. There are five bit periods between each vertical grid line. Notice that you will not find more than three 1 or 0 bits in a row in the data shown, and that there are always six ones and six zeros in any sequence of 12 bits. This is because each message byte has been encoded as 12 bits, always with six ones and six zeros, and with no more than four bits of the same type in a row for any combination of adjacent encoded characters. This is one type of coding that maintains good dynamic DC balance, and is similar to techniques used in fiber-optic data transmissions. Another popular encoding scheme is Manchester encoding, which encodes each 1 bit in the message as a 1-0 bit sequence, and each 0 bit in the message as a 0-1 bit sequence. Both 12-bit encoding and Manchester encoding work well. Manchester encoding has a maximum of two bits of the same type in a row, but requires 16 bits to encode a byte. 12-bit encoding can have up to 4 bits of the same type in a row, and requires, of course, 12 bits to encode a byte. By the way, your start vector should also be dynamically DC balanced in most cases.

The data rate and the encoding scheme you use affects two adjustments on the ASH transceiver (or vice versa). The most narrow pulse or gap in your encoded data sets the low-pass filter bandwidth. For the two encoding schemes we have discussed, this is one encoded bit period. Once you know the bit period, Section 2.5 in the ASH Transceiver Designer’s Guide explains how to set the low-pass filter bandwidth. The widest pulse or gap in your encoded data sets the value of the coupling capacitor. Once you know the maximum number of 1 bits or 0 bits that can occur in a row, you know the width of the maximum pulse or gap that can occur in your encoded data. Section 2.6 in the ASH Transceiver Designer’s Guide explains how to determine the coupling capacitor value and the required training preamble length from the maximum pulse or gap width.

Trying to send data without encoding is generally a disaster. Without a threshold, any long sequence of 1’s or 0’s in your data will charge or discharge the coupling capacitor, unbalancing the symmetry of the signal into the data slicer and ruining the noise rejection performance.

When you use one of the data encoding schemes discussed above with no slicer threshold, the coupling-capacitor transient response automatically adjusts the slicing symmetry as variations occur in received signal strength. This greatly improves system robustness to signal flutter. You usually want to make the coupling-capacitor value no larger than needed, so that fast signal fluctuations can be followed.

Let’s now consider message encoding schemes and ASH transceiver adjustments when a threshold is used. Again, a threshold trades-off sensitivity and flutter robustness for less noise in the no-signal condition. If you are using a strong threshold, you may decide you do not need a training preamble or start vector (this depends on the way you design your code). But if you are using AGC and/or data slicer DS2 in your ASH transceiver, you will need at least one 1-0-1-0… preamble byte for training these hardware functions. The threshold in DS1 has a built-in hysteresis. When the input voltage to the data slicer exceeds the threshold level, DS1 will output a logic 1, and it will continue to output a logic 1 until the input voltage swings below zero. The DC-balanced data encoding methods already discussed work satisfactorily with the DS1 hysteresis. Again, once you know the bit period of your encoded data, Section 2.5 in the ASH Transceiver Designer's Guide explains how to set the low-pass filter bandwidth. Note that a larger bandwidth is recommended for the same bit period when a threshold is used. Using the coupling capacitor value as determined in Section 2.6 of the ASH Transceiver Designer's Guide is a good default choice. When you use a threshold, 1 bits tend to drop out of weak and/or fluttering signals at the data slicer. Message patterns that contain a few less 1 bits than 0 bits work somewhat better with a strong threshold than classical DC-balanced codes. In some cases you may work with encoder and decoder chips designed to send command codes. Some of these chips send code messages with short preambles and relatively large gaps between the messages. These chips often work better if you use a moderate threshold and a relatively large coupling capacitor, so it is worth doing some experimenting.

2.3 Clock and Data Recovery

The clock and data recovery techniques used at the receiver are critical to overall system performance. Even at moderate signal-to-noise ratios, the output of the data slicer will exhibit some jitter in the position of the logic transitions. At lower signal-to-noise ratios, the jitter will become more severe and spikes of noise will start to appear at the data slicer output, as shown in Figure1.2.3.5. The better your clock and data recovery techniques can handle edge jitter and occasional noise spikes, the more robust your radio link will be. There is some good news about edge jitter due to Gaussian noise. The average position of the logic transitions are in the same place as the noise-free case. This allows you to use a phase-locked loop (PLL) that hones in on the average position of the data edges for clock recovery. Once your clock recovery PLL is lined up, you can use the logic state at the middle of each bit period, or the dominant logic state across each bit period as your recovered bit value. Testing mid-bit works best when the low-pass filter is well-matched to the data rate. On the other hand, determining the dominant logic state across a bit period can improve performance when the low-pass filter is not so well matched. The dominant logic state is often determined using an "integrate and dump" algorithm, which is a type of averaging filter itself.

It is possible to use simple data recovery techniques for less demanding applications (close operating range so the signal-to-noise ratio is high). The standard protocol software that comes in the DR1200-DK, DR1201-DK and DR1300-DK Virtual Wire® Development Kits uses a simplified data recovery technique to achieve air transmission rates of 22.5 kbps with a modest microcontroller. And yes, ordinary UARTs are being used successfully in non-demanding applications. But a word of caution. It appears the UARTs built into some microcontroller chips really don't like even moderate edge jitter. If you are considering using a built-in UART on the radio side, do some testing before you commit your design to that direction.

About now you may be wondering if anybody builds an “RF UART”, which is designed for low signal-to-noise ratio applications. The IC1000 discussed below is one example of this concept.

2.4 Communication Protocols

So far, we have discussed message encoding techniques for robust RF data transmission, and clock and data recovery techniques that can work with some noise-induced edge jitter and occasional noise spikes. Even so, transmission errors and drop outs will occur. The main job of your communication protocol is to achieve near-perfect communications over an imperfect RF communication channel, or to alarm you when a communication problem occurs. And channel sharing is often another requirement.

A protocol is a set of standard structures and procedures for communicating digital information. A complete protocol is often visualized as a stack of structures and procedures that are very specific to the communication hardware and channel characteristics at the bottom, and more general-purpose and/or application oriented at the top.

Packet-based protocols are widely used for digital RF communications (and for sending data on many other types of communications channels.) Even simple command transmissions usually employ a packet-style data structure.

2.4.1 Digital command transmissions

In addition to ASH transceivers, RFM’s second-generation ASH radio product line includes transmitter and receiver derivatives for one-way RF communications. Most one-way command applications are actually two-way; RF in one direction and audible or visual in the other direction. For example, you press the “open” button until you see the garage door or gate start moving. The data encoding and data recovery techniques discussed above can be used to build a robust one-way RF communications system. But often, off-the-shelf command encoder and decoder ICs are used. Among the most popular are the Microchip KeeLoq™ ICs. Figure 2.4.1 shows RFM’s suggested application circuit for second-generation ASH receivers driving KeeLoq™ decoders. You can usually derive enough information from the data sheets of other encoder and decoder ICs to calculate the component values to use with second-generation ASH receivers. The calculations are the same as discussed in the ASH Transceiver Designer’s Guide.

There is a growing trend to replace one-way RF communication links with two-way links for added system integrity. This is especially true for one-way RF communication links that are not activated by the user. Wireless home security systems are one example.

ASH Receiver Application Circuit
KeeLoq Configuration

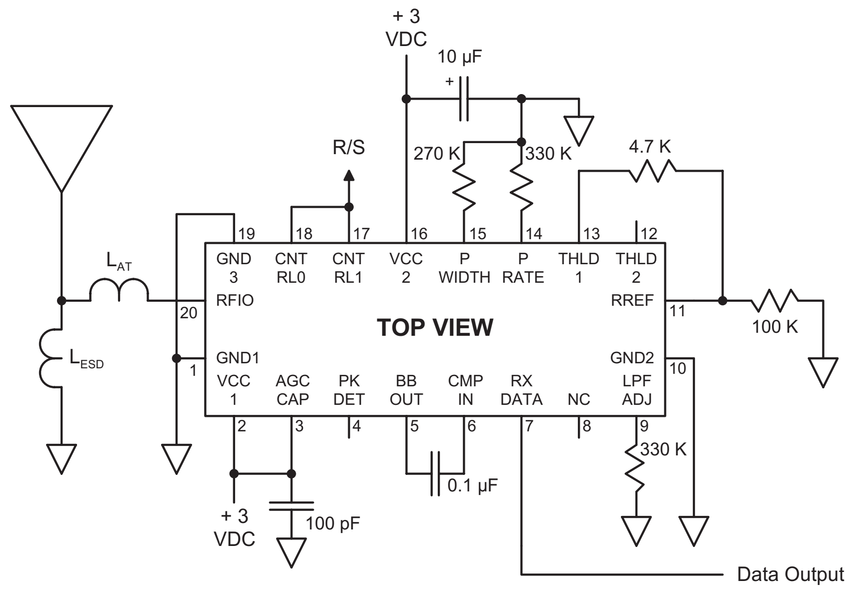


Figure 2.4.1

2.4.2 Data transmissions using packet protocols

A packet structure generally includes a training preamble, start symbol, routing information (to/from, etc.) packet ID, all or part of a message, and error detection bits. Other information may be included depending on the protocol. Communications between nodes in a packet-based system may be uncoordinated (talk when you want to) or coordinated (talk only when it is your turn). In the case of uncoordinated transmissions, packet collisions are possible. Theorists note that the collision problem limits the throughput of an uncoordinated channel to about 18% of its steady one-way capacity. Coordinated transmissions have higher potential throughput but are more complex to code. Many applications that use ASH radio technology transmit relatively infrequently, so uncoordinated transmissions work very successfully.

In both uncoordinated and coordinated systems, transmission errors can and will occur. An acknowledgment (ACK) transmission back to the sending node is used to confirm that the destination node has received the packet error free. Error-detection bits are added to a packet so the destination node can determine if the packet was received accurately. Simple parity checks or checksums are not considered strong enough for error checking RF transmissions. The error-detection bits added to the end of a packet are often called a frame check sequence (FCS). An FCS is usually 16 to 24 bits long, and is generated using a cyclic redundancy code (CRC) method. IBM developed such a code many years ago for their X.25 protocol and it is still widely used for RF packet transmissions. The ISO3309

Standard details the generation of this error detection code, and it is used in the protocol code example below.

It is time to bring up the real challenge in designing and writing protocol software. Events can happen in any sequence, and data coming into the protocol software can be corrupted in any bit or in every bit (remember, short packets work best on a low signal-to-noise radio channel). It is worth doing a careful “what if” study relevant to your protocol and your application before doing the detailed design and coding of your software. Consider how you can force unlikely sequences of events in your testing. Thorough front end planning can avoid a lot of downstream problems.

3 IC1000 “Radio UART”

RFM has introduced the IC1000 to support fast-track product development cycles using ASH radio technology. The IC1000 implements the clock and data recovery tasks that often constitute a lot of the learning curve in your first RF protocol project. The IC1000 is designed to operate with no threshold, which is the key to good system sensitivity.

3.1 IC1000 Description

The IC1000 is implemented in an industrial temperature range PIC12LC508A-04I\SN microcontroller using internal clocking. Nominal operating current is 450 µA, consistent with the low operating current emphasis of the second-generation ASH radio product line. The IC1000 is provided in a miniature eight-pin SMT package.

3.2 IC1000 Application

A typical IC1000 application is shown in Figure 3.2.1. The data (slicer) output from the second-generation ASH transceiver is buffered by an inverting buffer and is applied to Pin 3 of the IC1000 and the Data In pin of the host microprocessor. When the IC1000 detects the presence of a specific start-of-data pulse sequence, it outputs a Start Detect pulse on Pin 2. This pulse is applied to an interrupt pin on the host processor. The IC1000 generates data clocking (data valid) pulses in the middle of each following bit period using an oversampled clock extraction method. The IC1000 is designed to tolerate continuous input noise while searching for a start-of-data pulse sequence.

The IC1000 supports four data rates - 2400, 4800, 9600, and 19200 bits per second (bps). The data rate is selected by setting the logic input levels to Pin 6 (Speed 1) and Pin 7 (Speed 0). Please refer to the IC1000 data sheet for additional information.

4 Example Data Link Layer Protocol

The data link protocol discussed below is tuned for high-sensitivity, low data rate requirements. The protocol code is designed to run on the ATMEL AT89C2051 microcontroller used in the DR1200-DK/DR1200A-DK Series Virtual Wire® Development Kits. The “A” version kits (DR1200A-DK, etc.) ship with this software and require no hardware

Typical IC1000 Application

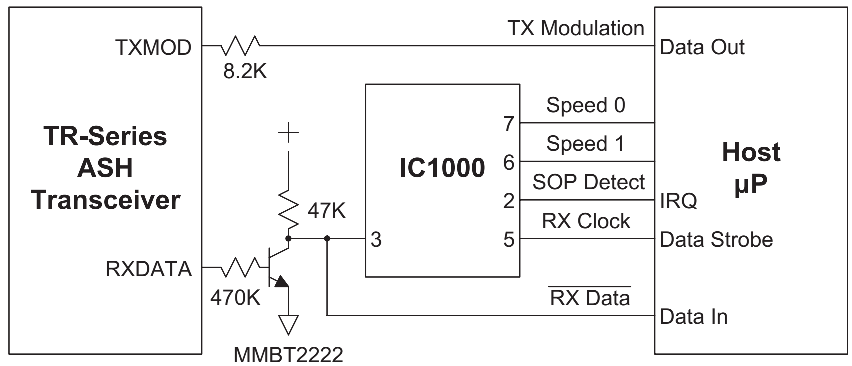


Figure 3.2.1

modifications. It is necessary to replace the radio boards used in the standard kits with "A" version radio boards before using this code, or to modify the standard radio boards as detailed below. Figure 4.1 shows the circuit modification used between the ASH transceiver base-band output, Pin 5, and the comparator (data-slicer) input, Pin 6. Figure 4.2 shows how these components are installed and their values. This modification reduces the

ASH Transceiver Application Circuit
Low Data Rate OOK

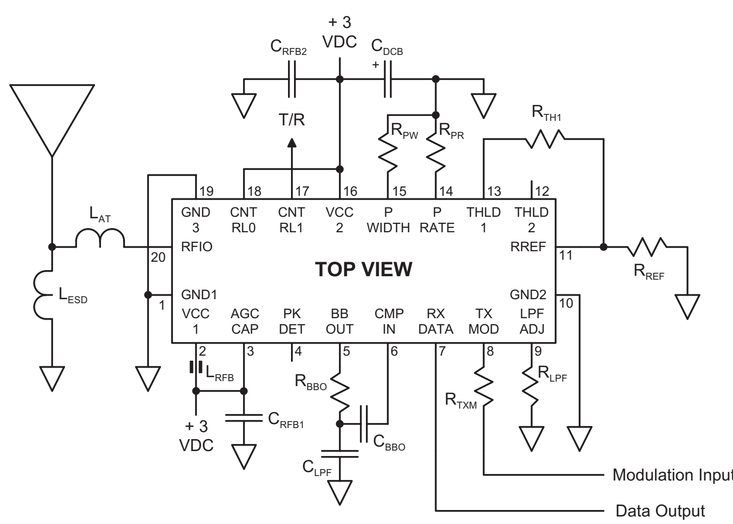


Figure 4.1

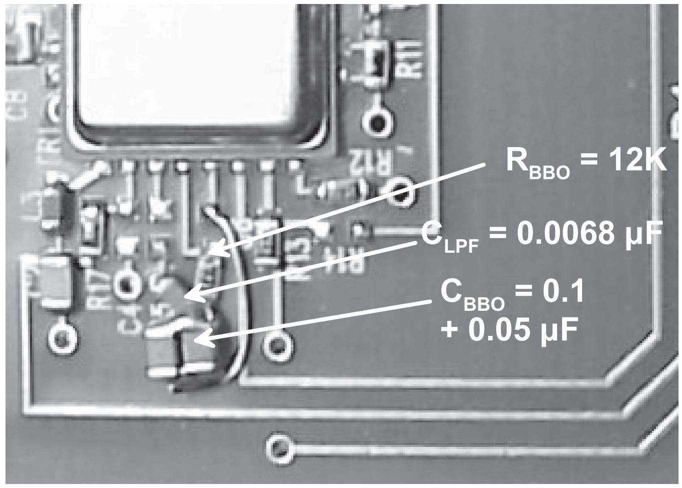


Figure 4.2

noise bandwidth of the receiver. In addition, R9 on the DR1200, DR1201 and DR1300 radio boards should be changed to a zero-ohm jumper (no DS1 threshold). R12 should be changed to 330 K on all three radio boards. Note that the DR1200A, DR1201A and DR1300A already incorporate these modifications.

4.1 Link Layer Protocol Source Code

The link layer protocol is implemented in 8051 assembly language and the source, DK200A.ASM (RFM P/N SW0012.V01), is compatible with the popular TASM 3.01 shareware assembler. You can get TASM 3.01 at www.rehn.org/YAM51/files.shtml.

By the way, this "A" link layer protocol uses the programming pins differently than the protocol supplied in the standard development kits. See Picture 4.3. Placing a jumper next to the "dot" end (ID0) enables the AutoSend mode (do this on one protocol board only). Placing a jumper at the far end (ID3) strips the packet framing and header characters off

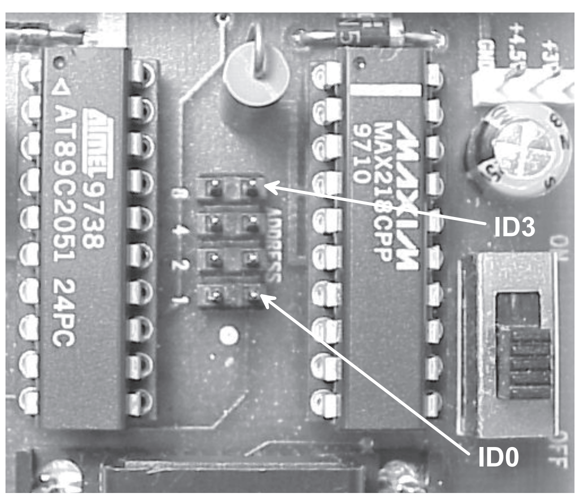


Figure 4.3

received packets. This can be handy for driving small serial printers, etc. You do not use jumpers to set the FROM address with this protocol.

Details of the packet and byte structures used by the protocol are shown in Figure 4.4. The host-protocol packet structure begins and ends with a 0C0H framing character (FEND) that cannot be used elsewhere in the packet. For example, you cannot use 0C0H in the TO/FROM address byte. This will otherwise not be a problem using seven-bit ASCII message characters. Eight-bit data can be sent using seven-bit ASCII characters to represent numerical values, or a framing character substitution scheme like the one used in the Internet SLIP protocol can be employed. The framing character helps deal with the "non real time" nature of serial ports on your typical PC. The host-protocol packet structure within the frame includes the TO/FROM address byte, with the high nibble the TO address and the low nibble the FROM address. The ID byte indicates which packet this is. Each packet can hold up to 24 additional message bytes. As mentioned, short packets should be used on radio channels.

Framing characters are not needed in the transmitted packet structure as the protocol is real time on the radio side. The transmitted packet structure beings with a 1-0-1-0… preamble which establishes good signal slicing symmetry at the input to the radio comparator and then trains the clock and data recovery processes in the software. The preamble is followed by a 12-bit start symbol that provides good discrimination to random noise patterns. The number of bytes in the packet (beyond the start symbol), the TO/FROM address, packet ID, message bytes and FCS then follow. The start symbol and all bytes following are 12-bit encoded for good dynamic DC balance.

Packet and Byte Structure Details

Host-Protocol Packet Structure:

FEND	TO/FROM	ID	Message	FEND

Transmitted Packet Structure:

Preamble	Start Symbol	# Bytes	TO/FROM	ID	Message	FCS

Host-Protocol ACK/NAK Structure:

FEND	TO/FROM	IDS	FEND

Transmitted ACK Structure:

Preamble	Start Symbol	69	TO/FROM	ID	FCS

TO/FROM Byte Detail:

TO Nibble	FROM Nibble

IDS Byte Detail:

ACK/NAK Bit	3 ID Bits	4 Retry # Bits

Figure 4.4

ACK and NAK packets contain an IDS byte which is detailed in Figure 4.4. The most significant bit in this byte is set to 1 for an ACK or 0 for a NAK. The next three bits are the packet ID, and the lower nibble of the byte holds the retry number for the ACK.

On power up the program is initialized by a call to the `setup` subroutine. The program then begins running in the `main` loop. The `tick` subroutine is called every 104.18 microseconds through `t_isr`, the interrupt service routine for timer T0. The `tick` subroutine always runs, and provides support for data reception, data transmission and event timing. The `tick` subroutine has a number of operating modes, controlled by the state of several flags.

Most of the time, `tick` will call `pll`, the receiver clock and data recovery subroutine. The `pll` subroutine uses two simple but effective signal processing techniques for accurately recovering bits from a data input steam with edge jitter and occasional noise spikes. The first signal processing technique is PLL clock alignment and the second technique is integrate-and-dump (I&D) bit estimation.

Register `R2` acts as a modulo 0 to 159 ramp counter that wraps on overflow about every 8 sampling ticks, (one bit period). This provides an 500 microsecond bit period, which equates to a nominal RF data rate of 2000 bits per second. Unless an edge occurs in the incoming bit stream, the ramp is incremented by 12.5% on each tick. If an edge occurs (change of logic state between ticks), the ramp is incremented 6.875% if the ramp value is below 80, or is incremented 18.125% if the ramp value is equal to or greater than 80. This causes the ramp period to gradually slide either backward or forward into alignment with the average bit period of the incoming data. After alignment, the position of the ramp can only change ±5.625% on each incoming data edge. Moderate edge jitter and occasional noise spikes will not seriously affect the ramp's alignment with the incoming data. Note that a preamble is needed to train the PLL (slide it into alignment).

Once the ramp is aligned, the I&D bit estimate becomes meaningful. The count in buffer `RXID` is incremented on each tick within a bit period if input sample `RXSMP` is a logic 1. At the end of the bit period (`R2` overflow wrap), the incoming bit is estimated to be a 0 if the count is four or less, or a 1 if the count is five or more. `RXID` is then cleared (dumped) in preparation for the next bit estimate. Integrate-and-dump estimation provides additional noise filtering by effectively averaging the value of the input samples within a bit period.

Once a bit value is determined, subroutine `pll` either inputs it into a 12-bit buffer (lower nibble of `RXBH` plus `RXBL`) used to detect the message start symbol, or adds it to buffer `RXBB`, which collects six-bit half symbols from the incoming encoded message. Flag `SOPFLG` controls which of these actions are taken.

You will notice that `tick` samples the RX input pin near the start of the subroutine, and when transmitting, outputs a TX bit sample as one of the first tasks. This helps minimize changes in the delay between timer T0 activating `t_isr` and these input/output events. If these activities are placed further down in the `tick` code or in the `pll` subroutine, an effect similar to adding extra noise-induced jitter can occur as different branches are taken through the code.

In addition to supporting data reception and transmission, the `tick` subroutine runs several timer functions. One timer provides a time-out for partial messages arriving from the host. The AutoSend timer and the transmit retry timer are also part of the `tick` subroutine.

The other interrupt service routine used by the protocol software is `s_isr`, which supports serial port interrupts by calling `srio`. The function of `srio` is to provide priority reception of messages from the host. An acknowledgment back to the host confirms the serial interrupt was enabled and the protocol received the host's message.

As mentioned, the code starts running in the `main` loop. A number of subroutines can be called from this loop, depending on the state of their associated control flags. Here are these subroutines and what they do:

The `do_as` subroutine automatically transmits a "Hello" test message paced by a timer in `tick`. This AutoSend function is activated by a call from `setup` if a jumper is detected across the pins near the "dot" end on the protocol board, as discussed above.

The `do_rt` subroutine retransmits a message if an ACK has not been received. Retransmissions are paced by a timer in `tick`. The timer is randomly loaded with one of eight different delays, which helps reduce the possibility of repeated collisions between two nodes trying to transmit a message at the same time. The protocol will attempt to transmit a message up to eight times. The `do_rt` subroutine manages attempts two through eight as needed.

The `aksnd` subroutine sends an ACK/NAK message back to the protocol's host to indicate the outcome of attempting to transmit a message. When called directly from the `main` subroutine, it sends a NAK message. When called from `do_rx`, it sends an ACK.

The `rxsop` subroutine detects the message start symbol (SOP) by comparing the bit pattern in the 12-bit correlation buffer updated by `pll` to the start symbol pattern. When the SOP pattern is detected, `rxsop` modifies flag states and clears buffers in preparation for receiving the encoded message. As mentioned, this protocol uses 12-bit encoding to achieve dynamic DC balance. The start symbol is not one of the 12-bit symbols used in the encoding table, but it is also DC balanced.

The `do_rx` subroutine receives and decodes the incoming message, tests the FCS for message accuracy, returns an ACK to the sender if it has received an error-free data message for this node, sends an ACK message to the host if it has received an ACK message for this node, and sends an error-free data message to the host if the message is for this node. These tasks are done by calling subroutines from `do_rx`. Here are these subroutines and what they do:

The `rxmsg` subroutine receives each six-bit half symbol from `pll` and converts it to a decoded nibble using the `smbl` table near the end of the listing. Decoded nibbles are assembled into bytes and added to the received message buffer. When all the message is received, control is returned to `do_rx`. If a message length overflow occurs, `rxmsg` fakes a short message that will fail the FCS test.

The `rxfcs` subroutine tests the message for errors by recalculating the FCS with the transmitted FCS bits included in the calculation. If there are no errors, the received FCS calculation will equal 0F0B8H. The `rxfcs` subroutine uses calls to `b_rfcs` and `a_rfcs` to do the FCS calculation and to test the results.

The `acktx` subroutine determines if the received message is an ACK for a packet (ID) being transmitted from this node. If so, `acktx` idles transmission attempts and signals `rxmsg` to send an ACK message to the host by setting flag states.

When called from `rxsmg`, `aksnd` sends an ACK message to the host. Notice that when `aksnd` is called from `main`, it sends a NAK message.

The `ackrx` subroutine transmits an ACK message back to the sending node when it receives a valid data message from the sending node addressed to it. The subroutines used by `ackrx` are “borrowed” from the transmit side of the protocol and will be discussed later.

The `rxsnd` subroutine sends a received data message to the host, provided the message is for its node and has passed the FCS test.

The `rxrst` subroutine resets flags and initializes buffers in preparation for receiving the next packet.

The first byte of a packet sent from the host triggers the serial interrupt service routine `t_isr` which calls subroutine `srio`. The serial interrupt is disabled and the `do_tx` subroutine is called. This subroutine takes in the message from the host, computes the FCS, turns the transmitter on, sends the preamble and start symbol, encodes and sends the message, and turns the transmitter off. The `do_tx` subroutine accomplishes these actions by calling other subroutines. Here are these transmit subroutines and what they do:

The `txget` subroutine receives the message from the host and loads it into the transmit message buffer. Provisions are made in `txget` to exit on a null message (just two FENDs), time-out on partial messages, or send the first part of an incoming message that is overflowing in length. Since the serial interrupt service routine is disabled from time-to-time, a short packet transfer acknowledgment message (PAC) is sent back to the host to confirm the protocol has the message and is attempting to transmit it. No PAC is sent on a null message or a time-out as there is nothing to send.

The `txfcs` subroutine calculates the FCS that will be used for error detection at the receive end. It uses calls to `b_tfcs` and `a_tfcs` to do the FCS calculation and to add the results to the message.

The `txpre` subroutine turns on the transmitter and after a short delay sends the preamble and start symbol using the data in the `tstrt` table near the end of the listing. Note that `txpre` is supported by `tick` to provide sample-by-sample bit transmission.

The `txmsg` subroutine encodes the message bytes as 12-bit symbols and transmits them in cooperation with `tick`. This subroutine uses the `smbl` table to encode each nibble in each message byte into six bits.

The `txrst` subroutine can either reset to send the same message again or can reset to receive a new message from the host, based on flag states.

The `do_tx` subroutine receives a message from the host and attempts to transmit it once. Additional transmit attempts are done by `do_rt`, which is called from `main` as needed. The `do_rt` subroutine uses most of the same subroutines as `do_tx`. The `do_as` subroutine can also be called from `main` to provide the AutoSend test transmission and it also uses many of the same subroutines as `do_tx`. And as mentioned earlier, `ackrx` uses several of these subroutines to transmit an ACK back for a received message.

4.2 Terminal Program Source

V110T30C.FRM is the Visual Basic source code for the companion terminal program to DK200A.ASM. After initializing flags, variables, etc., the form window is shown and the program starts making periodic calls to the `Timer1_Timer` "heartbeat" subroutine. The `Xfer` subroutine provides time-outs for PAC, ACK or NAK messages expected back from the protocol. `Xfer` is also handy for reminding you to turn on the power switch or put fresh batteries in the protocol board. The PC's serial input buffer is set up for polling (no interrupts) and is serviced by calling `RxPtk` from `Timer1_Timer`. The terminal program also has an AutoSend subroutine, `ASPkt`, that is called from `Timer1_Timer` when AutoSend is active. (No, you are not supposed to use the AutoSend feature in the protocol and the host program at the same time.) Here is a listing of the terminal program subroutines and what they do:

`RxPkt` is called from `Timer1_Timer` when bytes are found in the serial port input buffer. `RxPkt` calls two other subroutines, `InCom` and `ShowPkt`.

`InCom` collects bytes from the serial port input buffer for a period of time set by the `InDel!` variable. These bytes are added to the end of the `RPkt$` string variable, which acts as byte FIFO.

`ShowPkt` is then called to display or otherwise process the bytes in `RPkt$`. The outer `Do, Loop Until (J = 0)` structure takes advantage of the framing characters to separate individual packets in `RPkt$`. This avoids the need for reading the PC's serial port input buffer at precise times which you probably can't do anyway. As each packet is removed from the left side of `RPkt$`, it is checked to see if it is a one-character PAC (0FFH character), a two-character ACK or NAK, or a data message of three or more characters. Flags `TFlag`, `ANFlag`, `NAFlag` and `TNFlag` are reset by `ShowPkt` as appropriate and are used by the `Xfer` monitoring subroutine to confirm messages are flowing back from the protocol in a timely manner. The `NAFlag` enables the next AutoSend transmission. The `ShwACK` flag selects either to display inbound messages (and PID Skips) only, or inbound messages plus PAC, ACK/NAK, TO/FROM and ID information.

`Text1_KeyPress` is used to build messages for transmission. Editing is limited to backspacing, and the message is sent by pressing the Enter key or entering the 240th character.

`SndPkt` breaks the message into packets, adds the framing characters, the TO/FROM address and the ID number to each packet and sends them out. `SndPkt` sets the `TFlag` and `ANFlag` flags and clears the value of several variables. NxtPkt is a small subroutine used by `SndPkt` that picks a new ID number for each packet.

`Xfer` monitors the elapsed time from when a packet is sent out (to the protocol) and a PAC is received back, and the elapsed time from when a packet is sent out and an ACK or NAK is received back. `Xfer` will display error messages and reset control flags and other variables through `ReSetTX` if these elapsed times get too long.

`ASPkt` automatically sends test packets using the `NxtPkt` and `SndPkt` subroutines. It is paced by the state of the `NAFlag`.

`GetPkt` is a small subroutine that supplies `ASPkt` with a message. Until the first message is typed in, `GetPkt` provides a default message. It otherwise provides the last message typed in.

`LenTrap` clears a text window when 32,000 bytes of text have accumulated in it.

The remaining subroutines in the terminal program are classical event procedures related to mouse clicks on the terminal program window. Most of these relate to the Menu bar.

The three top level choices on the Menu bar are *File*, *Edit* and *View*. Under *File* you can choose to *Exit* the terminal program. Under *Edit*, the next level of choices are the *To Address* and the *From Address*. Under the *To Address* you can choose *Nodes 1*, *2*, *3*, or *4*, with *Node 2* the default. Under the *From Address* you can choose *Nodes 1*, *2*, *3*, or *4*, again with *Node 2* the default.

Under *View* you can choose *Clear* (screen), *Show RX Dups, Show ACK/NAK*, and *AutoSend*, as discussed earlier. The status bar and its embedded progress bar at the bottom of the form monitors outbound packets even when *Show ACK/NAK* is not enabled.

4.3 Variations and Options

In most real world applications, `s_isr`, `srio`, `txget`, `rxsnd` and `aksnd` would be replaced with resident application subroutines. Your real-world application is left as a homework assignment. Test, test, test!

Another pair of programs are provided for your experimentation. DK110K.ASM is a simplified "shell" protocol that transmits a message received from the host (once) and sends any message received with a valid FCS to the host. PAC/ACK/NAK handshaking between the host and the protocol and between protocol nodes is not implemented. Also, no TO/FROM address filtering is provided at the protocol level. This gives you the flexibility to add these types of features either to the protocol or the terminal program yourself. Terminal Program V110T05B.FRM works with DK110K.ASM and provides a simple implementation of ACK/NAK handshaking at the host level. Of course, DK110K.ASM is not intended to work with V110T30C.FRM and DK200A.ASM is not intended to work with V110T05B.FRM.

4.4 Test Results

Laboratory tests show that a 916.5 MHz ASH radio system using the example software achieves a bit-error-rate between 10^{-4} and 10^{-3} at a received signal level of -101 dBm using pulse modulation (or -107 dBm using 100% amplitude modulation). Open-field range tests using commercial half-wave dipole antennas (Astron Antenna Model AXH9NSMS) demonstrate good performance chest-high at distances of one-eighth mile or more.

5 Source Code Listings

5.1 DK200A.ASM

```
;  DK200A.ASM 2002.07.31 @ 20:00 CST
;  See RFM Virtual Wire(r) Development Kit Warranty & License for terms of use
;  Experimental software - NO representation is
;  made that this software is suitable for any purpose
;  Copyright(c) 2000 - 2002, RF Monolithics, Inc.
;  AT89C2051 assembler source code file (TASM 3.01 assembler)
;  Low signal-to-noise protocol for RFM ASH transceiver
;  Integrate & dump PLL (I&D) - 62.40 us tick

          .NOLIST
          #INCLUDE "8051.H"        ;  tasm 8051 include file
          .LIST

;  constants:

ITMOD     .EQU      022H           ;  set timers 0 and 1 to mode 2
ITICK     .EQU      141            ;  set timer T0 for 62.40 us tick
ISMOD     .EQU      080H           ;  SMOD = 1 in PCON
IBAUD     .EQU      0FAH           ;  19.2 kbps @ 22.1184 MHz, SMOD = 1
ISCON     .EQU      050H           ;  UART mode 1

RMPT      .EQU      159            ;  PLL ramp top value (modulo 0 to 159)
RMPW      .EQU      159            ;  PLL ramp reset (wrap) value
RMPS      .EQU      80             ;  PLL ramp switch value
RMPI      .EQU      20             ;  PLL ramp increment value
RMPA      .EQU      29             ;  PLL 5.625% advance increment value (20 + 9)
RMPR      .EQU      11             ;  PLL 5.625% retard increment value (20 - 9)

AKMB      .EQU      03EH           ;  ACK message buffer start address
TXMB      .EQU      043H           ;  TX message buffer start address
TFTX      .EQU      044H           ;  TO/FROM TX message buffer address
IDTX      .EQU      045H           ;  packet ID TX message buffer address
RXMB      .EQU      061H           ;  RX message buffer start address
TFRX      .EQU      062H           ;  TO/FROM RX message buffer address
IDRX      .EQU      063H           ;  packet ID RX message buffer address
FEND      .EQU      0C0H           ;  FEND framing character (192)
SOPL      .EQU      08AH           ;  SOP low correlator pattern
SOPH      .EQU      0B3H           ;  SOP high correlator pattern
TXR0      .EQU      026H           ;  TX retry timer count

FCSS      .EQU      0FFH           ;  FCS seed
FCSH      .EQU      084H           ;  FCS high XOR mask
FCSL      .EQU      08H            ;  FCS low XOR mask
FCVH      .EQU      0F0H           ;  FCS valid high byte pattern
FCVL      .EQU      0B8H           ;  FCS valid low byte pattern

;  stack:  08H - 021H (26 bytes)

;  bit labels:

WBFLG     .EQU      010H           ;  warm boot flag (future use)
PLLON     .EQU      011H           ;  RX PLL control flag
RXISM     .EQU      012H           ;  RX inverted input sample
RXSMP     .EQU      013H           ;  RX input sample
LRXSM     .EQU      014H           ;  last RX input sample
RXBIT     .EQU      015H           ;  RX input bit
RXBFLG    .EQU      016H           ;  RX input bit flag
SOPFLG    .EQU      017H           ;  SOP detect flag
RXSFLG    .EQU      018H           ;  RX symbol flag
RM        .EQU      019H           ;  RX FCS message bit
OKFLG     .EQU      01AH           ;  RX FCS OK flag

SIFLG     .EQU      01BH           ;  serial in active flag
TSFLG     .EQU      01CH           ;  output TX sample flag
TXBIT     .EQU      01DH           ;  TX message bit
TM        .EQU      01EH           ;  TX FCS message bit
TXFLG     .EQU      01FH           ;  TX active flag
TMFLG     .EQU      020H           ;  TX message flag
TOFLG     .EQU      021H           ;  get message time out flag

AMFLG     .EQU      022H           ;  AutoSend message flag
ASFLG     .EQU      023H           ;  AutoSend active flag
ANFLG     .EQU      024H           ;  ACK/NAK status flag
```

```
SAFLG     .EQU      025H          ;  send ACK/NAK flag
NHFLG     .EQU      026H          ;  no RX FEND/header flag

SFLG1     .EQU      027H          ;  spare flag 1
SFLG2     .EQU      028H          ;  spare flag 2
SFLG3     .EQU      029H          ;  spare flag 3
SFLG4     .EQU      02AH          ;  spare flag 4
SFLG5     .EQU      02BH          ;  spare flag 5
SFLG6     .EQU      02CH          ;  spare flag 6
SFLG7     .EQU      02DH          ;  spare flag 7
SFLG8     .EQU      02EH          ;  spare flag 8
SFLG9     .EQU      02FH          ;  spare flag 9

;  register usage:

;  R0                                RX data pointer
;  R1                                TX data pointer
;  R2                                PLL ramp buffer
;  R3                                RX FCS buffer A
;  R4                                not used
;  R5                                TX FCS buffer A
;  R6                                TX FCS buffer B
;  R7                                RX FCS buffer B

;  byte labels:

BOOT      .EQU      022H          ;  1st byte of flags

RXID      .EQU      026H          ;  RX integrate & dump buffer
RXBL      .EQU      027H          ;  RX low buffer, SOP correlator, etc.
RXBH      .EQU      028H          ;  RX high buffer, SOP correlator, etc.
RXBB      .EQU      029H          ;  RX symbol decode byte buffer
RMDC      .EQU      02AH          ;  RX symbol decode loop counter
RMBIC     .EQU      02BH          ;  RX symbol decode index pointer
RMBYC     .EQU      02CH          ;  RX message byte counter
RMFCS     .EQU      02DH          ;  RX FCS byte buffer
RMSBC     .EQU      02EH          ;  RX symbol bit counter
RMLPC     .EQU      02FH          ;  RX message loop counter
RMFCC     .EQU      030H          ;  RX message FCS counter, etc.

TMFCC     .EQU      031H          ;  TX timer & loop counter
TXSMC     .EQU      032H          ;  TX output sample counter
TMBIC     .EQU      033H          ;  TX message bit counter
TMBYT     .EQU      034H          ;  TX message byte buffer
TMBYC     .EQU      035H          ;  TX message byte counter
TXSL      .EQU      036H          ;  TX message symbol low buffer
TXSH      .EQU      037H          ;  TX message symbol high buffer
TMFCS     .EQU      038H          ;  TX FCS byte buffer
TXTL      .EQU      039H          ;  TX timer low byte
TXTH      .EQU      03AH          ;  TX timer high byte
TXCNT     .EQU      03BH          ;  TX retry counter
IDBUF     .EQU      03CH          ;  packet ID buffer
TFBUF     .EQU      03DH          ;  TO/FROM address buffer

;  I/O pins:

MAX       .EQU      P1.6          ;  Maxim 218 power (on = 1)

RXPIN     .EQU      P3.2          ;  RX input pin (inverted data)
TXPIN     .EQU      P3.3          ;  TX output pin (on = 1)
PTT       .EQU      P1.7          ;  transmit enable (TX = 0)

PCRCV     .EQU      P3.7          ;  PC (host) input LED (on = 0)
RFRCV     .EQU      P3.5          ;  RX FCS OK LED (on = 0)
RXI       .EQU      P3.4          ;  RX activity LED (on = 0)

ID0       .EQU      P1.2          ;  jumper input bit 0 (dot end)
ID1       .EQU      P1.3          ;  jumper input bit 1
ID2       .EQU      P1.4          ;  jumper input bit 2
ID3       .EQU      P1.5          ;  jumper input bit 3

;  start of code:

          .ORG      00H           ;  hardware reset
          SETB      WBFLG         ;  set warm boot flag
reset:    AJMP      start         ;  jump to start

          .ORG      0BH           ;  timer 0 interrupt vector
t_isr:    ACALL     tick          ;  sampling tick subroutine
          RETI                    ;  interrupt done
```

```
        .ORG    023H            ;  serial interrupt vector
s_isr:  ACALL   srio            ;  serial I/O subroutine
        CLR     TI              ;  clear TI (byte sent) flag
        CLR     RI              ;  clear RI (byte received) flag
        RETI                    ;  interrupt done

        .ORG    040H            ;  above interrupt code space
start:  ACALL   setup           ;  initialization code

main:   JNB     AMFLG,mn0       ;  skip if AutoSend idle
        CLR     PCRCV           ;  else turn PCRCV LED on
        ACALL   do_as           ;  do AutoSend
        SETB    PCRCV           ;  turn PCRCV LED off
        AJMP    mn1             ;  and jump to RX SOP detect
mn0:    JNB     TMFLG,mn1       ;  skip if TX message idle
        CLR     PCRCV           ;  else turn PCRCV LED on
        ACALL   do_rt           ;  do TX retry
        SETB    PCRCV           ;  turn PCRCV LED off
mn1:    JNB     SAFLG,mn2       ;  skip if send ACK/NAK flag reset
        ACALL   aksnd           ;  else send NAK to host
mn2:    ACALL   rxsop           ;  do RX SOP detect
        JNB     SOPFLG,main     ;  if not SOP loop to main
        ACALL   do_rx           ;  else do RX message
mn_d:   AJMP    main            ;  and loop to main

do_rx:  CLR     ES              ;  deactivate serial interrupts
        ACALL   rxmsg           ;  decode RX message
        CLR     PLLON           ;  idle RX PLL
        ACALL   rxfcs           ;  test RX message FCS
        JNB     OKFLG,rx2       ;  reset if FCS error
        JNB     TXFLG,rx0       ;  skip if send TX idle
        ACALL   acktx           ;  if TX ACK, set send ACK flag
        JNB     SAFLG,rx0       ;  skip if send ACK/NAK flag reset
        ACALL   aksnd           ;  else send ACK message to host
        AJMP    rx2             ;  and jump to reset RX
rx0:    JB      ASFLG,rx1       ;  don't ACK AutoSend
        ACALL   ackrx           ;  ACK RX message
rx1:    ACALL   rxsnd           ;  send RX message to host
rx2:    ACALL   rxrst           ;  reset for next RX message
        SETB    PLLON           ;  enable RX PLL
        CLR     TI              ;  clear TI flag
        CLR     RI              ;  clear RI flag
        SETB    ES              ;  activate serial interrupts
rx_d:   RET                     ;  RX done

tick:   PUSH    PSW             ;  push status
        PUSH    ACC             ;  push accumulator
        MOV     C,RXPIN         ;  read RX input pin
        MOV     RXISM,C         ;  store as inverted RX sample
        JNB     TSFLG,tic0      ;  skip if TX sample out idle
        MOV     A,TXSMC         ;  else get sample count
        JZ      tic0            ;  skip if 0
        MOV     C,TXBIT         ;  else load TX bit
        MOV     TXPIN,C         ;  into TX output pin
        DEC     TXSMC           ;  decrement sample count
tic0:   JNB     PLLON,tic1      ;  skip if PLL idle
        ACALL   pll             ;  else run RX PLL
tic1:   JNB     TOFLG,tic2      ;  skip if get message timeout idle
        INC     TMFCC           ;  else bump timeout counter
        MOV     A,TMFCC         ;  get counter
        CJNE    A,#50,tic2      ;  skip if counter <> 50 (5.2 ms)
        CLR     TOFLG           ;  else reset time out flag
        MOV     TMFCC,#0        ;  reset counter
tic2:   INC     TXTL            ;  bump TX timer low
        MOV     A,TXTL          ;  load TX timer low
        JNZ     tick_d          ;  done if no rollover
        JNB     ASFLG,tic3      ;  skip if AutoSend idle
        DJNZ    TXTH,tick_d     ;  decrement TXTH, done if <> 0
        SETB    AMFLG           ;  else set AM message flag
        MOV     TXTL,#0         ;  clear TX delay low
        MOV     TXTH,#TXR0      ;  reload TX delay high
        AJMP    tick_d          ;  and jump to tic6
tic3:   JNB     TXFLG,tick_d    ;  skip if TX idle
        DJNZ    TXTH,tick_d     ;  decrement TXTH, done if <> 0
        SETB    TMFLG           ;  else set TM message flag
        MOV     DPTR,#delay     ;  point to delay table
        MOV     A,TL1           ;  get random table offset
        ANL     A,#07H          ;  mask out upper 5 bits
        MOVC    A,@A+DPTR       ;  load byte from table
        MOV     TXTH,A          ;  into TX delay high
        MOV     TXTL,#0         ;  clear TX delay low
```

```
        MOV     A,TXCNT         ;  load retry count
        CJNE    A,#9,tick_d     ;  if <> 9 jump to tick_d
        CLR     TMFLG           ;  else reset send TX message
        CLR     ANFLG           ;  reset ACK/NAK flag (NAK)
        SETB    SAFLG           ;  set send ACK/NAK flag
        CLR     TXFLG           ;  reset TX active flag
tick_d: POP     ACC             ;  pop accumulator
        POP     PSW             ;  pop status
        RET                     ;  tick done

pll:    MOV     C,RXSMP         ;  load RX sample
        MOV     LRXSM,C         ;  into last RX sample
        MOV     C,RXISM         ;  get inverted RX sample
        CPL     C               ;  invert sample
        MOV     RXSMP,C         ;  and store RX sample
        JNC     pll0            ;  if <> 1 jump to pll0
        INC     RXID            ;  else increment I&D
pll0:   JNB     LRXSM,pll1      ;  if last sample 1
        CPL     C               ;  invert current sample
pll1:   JNC     pll4            ;  if no edge jump to pll4
        MOV     A,R2            ;  else get PLL value
        CLR     C               ;  clear borrow
        SUBB    A,#RMPS         ;  subtract ramp switch value
        JC      pll3            ;  if < 0 then retard PLL
pll2:   MOV     A,R2            ;  else get PLL value
        ADD     A,#RMPA         ;  add (RMPI + 5.625%)
        MOV     R2,A            ;  store PLL value
        AJMP    pll5            ;  and jump to pll5
pll3:   MOV     A,R2            ;  get PLL value
        ADD     A,#RMPR         ;  add (RMPI - 5.625%)
        MOV     R2,A            ;  store PLL value
        AJMP    pll5            ;  and jump to pll5
pll4:   MOV     A,R2            ;  get PLL value
        ADD     A,#RMPI         ;  add ramp increment
        MOV     R2,A            ;  store new PLL value
pll5:   CLR     C               ;  clear borrow
        MOV     A,R2            ;  get PLL ramp value
        SUBB    A,#RMPT         ;  subtract ramp top
        JC      pllD            ;  if < 0 don't wrap
pll6:   MOV     A,R2            ;  else get PLL value
        CLR     C               ;  clear borrow
        SUBB    A,#RMPW         ;  subtract reset value
        MOV     R2,A            ;  and store result
        CLR     C               ;  clear borrow
        MOV     A,RXID          ;  get I&D buffer
        SUBB    A,#5            ;  subtract 5
        JNC     pll7            ;  if I&D count => 5 jump to pll7
        CLR     RXBIT           ;  else RX bit = 0 for I&D count < 5
        SETB    RXBFLG          ;  set new RX bit flag
        MOV     RXID,#0         ;  clear the I&D buffer
        AJMP    pll8            ;  and jump to pll8
pll7:   SETB    RXBIT           ;  RX bit = 1 for I&D count => 5
        SETB    RXBFLG          ;  set new RX bit flag
        MOV     RXID,#0         ;  clear the I&D buffer
pll8:   JB      SOPFLG,pllA     ;  skip after SOP detect
        MOV     A,RXBH          ;  else get RXBH
        CLR     C               ;  clear carry
        RRC     A               ;  rotate right
        JNB     RXBIT,pll9      ;  if bit = 0 jump to pll9
        SETB    ACC.7           ;  else set 7th bit
pll9:   MOV     RXBH,A          ;  store RXBH
        MOV     A,RXBL          ;  get RXBL
        RRC     A               ;  shift and pull in carry
        MOV     RXBL,A          ;  store RXBL
        AJMP    pll_d           ;  done for now
pllA:   MOV     A,RXBL          ;  get RXBL
        CLR     C               ;  clear carry
        RRC     A               ;  shift right
        JNB     RXBIT,pllB      ;  if bit = 0 jump to pllB
        SETB    ACC.5           ;  else set 5th bit
pllB:   MOV     RXBL,A          ;  store RXBL
        INC     RMSBC           ;  bump bit counter
        MOV     A,RMSBC         ;  get counter
        CJNE    A,#6,pllC       ;  if <> 6 jump to pllC
        MOV     RXBB,RXBL       ;  else get symbol
        MOV     RMSBC,#0        ;  reset counter
        SETB    RXSFLG          ;  set symbol flag
pllC:   AJMP    pll_d           ;  done for now
pllD:   CLR     RXBFLG          ;  clear RXBFLG
pll_d:  RET                     ;  PLL done
```

```
rxsop:    JNB       RXBFLG,sop_d    ;  done if no RX bit flag
          CLR       RXBFLG          ;  else clear RX bit flag
          MOV       A,RXBL          ;  get low RX buffer
          CJNE      A,#SOPL,sop_d   ;  done if <> SOPL
          MOV       A,RXBH          ;  else get high RX buffer
          CJNE      A,#SOPH,sop_d   ;  done if <> SOPH
          CLR       A               ;  else clear A
          MOV       RXBL,A          ;  clear RX low buffer
          MOV       RXBH,A          ;  clear RX high buffer
          MOV       RMSBC,A         ;  clear RX symbol bit counter
          CLR       RXSFLG          ;  clear RX symbol flag
          SETB      SOPFLG          ;  set SOP detected flag
          CLR       RXI             ;  RXI LED on
sop_d:    RET                       ;  SOP detect done

rxmsg:    JNB       RXSFLG,rxmsg    ;  wait for RX symbol flag
          CLR       RXSFLG          ;  clear RX symbol flag
rxm1:     MOV       DPTR,#smbl      ;  point to RX symbol decode table
          MOV       RMDC,#16        ;  16 symbol decode table entries
          MOV       RMBIC,#0        ;  index into symbol table
rxm2:     MOV       A,RMBIC         ;  load index into A
          MOVC      A,@A+DPTR       ;  get table entry
          XRL       A,RXBB          ;  XOR to compare with RXBB
          JZ        rxm3            ;  exit loop with decoded nibble
          INC       RMBIC           ;  else bump index
          DJNZ      RMDC,rxm2       ;  and try to decode again
rxm3:     MOV       A,RMBIC         ;  get decoded nibble
          SWAP      A               ;  swap to high nibble
          MOV       RXBH,A          ;  into RXBH (low nibble is high)
rxm4:     JNB       RXSFLG,rxm4     ;  wait for symbol flag
          CLR       RXSFLG          ;  clear flag
rxm5:     MOV       DPTR,#smbl      ;  point to symbol decode table
          MOV       RMDC,#16        ;  16 symbol decode table entries
          MOV       RMBIC,#0        ;  reset symbol table index
rxm6:     MOV       A,RMBIC         ;  load index into A
          MOVC      A,@A+DPTR       ;  get table entry
          XRL       A,RXBB          ;  XOR to compare with RXBB
          JZ        rxm7            ;  exit loop with decoded nibble
          INC       RMBIC           ;  else bump index
          DJNZ      RMDC,rxm6       ;  and try to decode again
rxm7:     MOV       A,RMBIC         ;  get decoded nibble
          ORL       A,RXBH          ;  add RXBH low
          SWAP      A               ;  nibbles now in right order
          MOV       RXBH,A          ;  store in RXBH
          MOV       @R0,RXBH        ;  and store in RX message buffer
          CJNE      R0,#RXMB,rxm8   ;  skip if not 1st message byte
          MOV       A,RXBH          ;  else get 1st byte
          ANL       A,#63           ;  mask upper 2 bits
          MOV       RMBYC,A         ;  load message byte counter
          MOV       RMFCC,A         ;  and RX message loop counter
          CLR       C               ;  clear borrow
          SUBB      A,#30           ;  compare number of bytes to 30
          JC        rxm8            ;  skip if < 30
          MOV       RMBYC,#4        ;  else force byte counter to 4
          MOV       RMFCC,#4        ;  and force loop counter to 4
rxm8:     INC       R0              ;  bump pointer
          DJNZ      RMFCC,rxmsg     ;  if <> 0 get another byte
          MOV       R0,#RXMB        ;  reset RX message pointer
          SETB      RXI             ;  turn LED off
rxm_d:    RET                       ;  RX message done

rxfcs:    MOV       RMFCC,RMBYC     ;  move byte count to loop counter
rxf0:     MOV       RMFCS,@R0       ;  get next message byte
          INC       R0              ;  bump pointer
          ACALL     b_rfcs          ;  build FCS
          DJNZ      RMFCC,rxf0      ;  loop for next byte
          ACALL     a_rfcs          ;  test FCS
rxf_d:    RET                       ;  RX FCS done

acktx:    MOV      A,RXMB           ;  get 1st RX byte
          ANL      A,#64            ;  mask ACK bit
          CJNE     A,#64,atx_d      ;  done if <> ACK
          MOV      A,TFBUF          ;  else get TX TO/FROM
          SWAP     A                ;  swap for FROM/TO
          CJNE     A,TFRX,atx_d     ;  done if <> RX TO/FROM
          MOV      A,IDBUF          ;  else get TX packet ID
          CJNE     A,IDRX,atx_d     ;  done if <> TX ID
          SETB     ANFLG            ;  else set ACK/NAK flag (ACK)
          SETB     SAFLG            ;  set send ACK/NAK message flag
          CLR      TXFLG            ;  clear TX active flag
atx_d:    RET                       ;  ACK TX done
```

```
ackrx:    MOV       A,TFBUF        ;  get local TO/FROM address
          ANL       A,#15          ;  mask to get local FROM address
          MOV       B,A            ;  store FROM address
          MOV       A,TFRX         ;  get T/F address from RX buffer
          SWAP      A              ;  swap - FROM/TO
          ANL       A,#15          ;  mask to get TO address
          CJNE      A,B,arx0       ;  done if not to this node
          MOV       R1,#AKMB       ;  load ACK pointer
          MOV       @R1,#69        ;  ACK bit + 5 bytes
          MOV       TMFCS,#69      ;  load TX message FCS byte
          ACALL     b_tfcs         ;  and build FCS
          INC       R1             ;  bump pointer
          MOV       A,TFRX         ;  get TO/FROM byte
          SWAP      A              ;  swap TO/FROM addresses
          MOV       @R1,A          ;  add to ACK buffer
          MOV       TMFCS,A        ;  load TX message FCS byte
          ACALL     b_tfcs         ;  and build FCS
          INC       R1             ;  bump pointer
          MOV       A,IDRX         ;  get packet ID byte
          MOV       @R1,A          ;  add ID to ACK message
          MOV       TMFCS,A        ;  load TX message FCS byte
          ACALL     b_tfcs         ;  and build FCS
          INC       R1             ;  bump pointer
          ACALL     a_tfcs         ;  add FCS
          MOV       R1,#AKMB       ;  reset ACK pointer
          PUSH      TMBYC          ;  push TX message TMBYC
          MOV       TMBYC,#5       ;  5 bytes in ACK
          ACALL     txpre          ;  send TX preamble
          ACALL     txmsg          ;  send TX message
          CLR       A              ;  reset for next TX
          MOV       TMBYT,A        ;  clear TX message byte
          MOV       TXSMC,A        ;  clear TX out count
          MOV       TXSL,A         ;  clear TX symbol low
          MOV       TXSH,A         ;  clear TX symbol high
          MOV       R1,#TXMB       ;  point R1 to message start
          POP       TMBYC          ;  restore TX message TMBYC
arx0:     SETB      RFRCV          ;  turn FCS LED off
arx_d:    RET                      ;  RX ACK done (rxsnd sets ES)

rxsnd:    CLR       PCRCV          ;  turn PC LED on
          MOV       A,TFBUF        ;  get local TO/FROM address
          ANL       A,#15          ;  mask to get local FROM address
          MOV       B,A            ;  store FROM address
          MOV       A,TFRX         ;  get T/F address from RX buffer
          SWAP      A              ;  swap - FROM/TO
          ANL       A,#15          ;  mask to get TO address
          CJNE      A,B,rxs4       ;  if <> don’t send to host
          DEC       RMBYC          ;  don’t send
          DEC       RMBYC          ;  the 2 FCS bytes
          MOV       R0,#RXMB       ;  reset RX message pointer
          MOV       @R0,#FEND      ;  replace # bytes with 1st FEND
          JNB       NHFLG,rxs0     ;  skip if no FEND/header flag reset
          INC       R0             ;  bump past FEND
          DEC       RMBYC          ;  decrement byte count
          INC       R0             ;  bump past TO/FROM
          DEC       RMBYC          ;  decrement byte count
          INC       R0             ;  bump past ID
          DEC       RMBYC          ;  decrement byte count
rxs0:     CLR       TI             ;  clear TI flag
rxs1:     MOV       SBUF,@R0       ;  send byte
rxs2:     JNB       TI,rxs2        ;  wait until byte sent
          CLR       TI             ;  clear TI flag
          INC       R0             ;  bump pointer
          DJNZ      RMBYC,rxs1     ;  loop to echo message
          JB        NHFLG,rxs4     ;  skip if no FEND/header flag set
          MOV       SBUF,#FEND     ;  add 2nd FEND
rxs3:     JNB       TI,rxs3        ;  wait until byte sent
          CLR       TI             ;  clear TI flag
rxs4:     SETB      RFRCV          ;  turn FCS LED off
          SETB      PCRCV          ;  turn PC LED off
rxs_d:    RET                      ;  send RX message done

aksnd:    CLR       ES             ;  disable serial interrupts
          CLR       PCRCV          ;  turn PC LED on
          CLR       SAFLG          ;  reset send ACK/NAK flag
          CLR       TXFLG          ;  reset TX active flag
          MOV       A,IDBUF        ;  get local ID
          ANL       A,#7           ;  mask unused bits
          SWAP      A              ;  swap ID to upper IDS nibble
          ADD       A,TXCNT        ;  add retry count to IDS
          JNB       ANFLG,aks0     ;  skip if NAK
```

```
        ADD     A,#128          ; else set ACK bit
aks0:   MOV     B,A             ; hold IDS in B
        MOV     A,TFBUF         ; get local TO/FROM
        SWAP    A               ; switch TO and FROM
        CLR     TI              ; clear TI flag
        MOV     SBUF,#FEND      ; send 1st FEND
aks1:   JNB     TI,aks1         ; wait until byte sent
        CLR     TI              ; clear TI flag
        MOV     SBUF,A          ; send TO/FROM
aks2:   JNB     TI,aks2         ; wait until byte sent
        CLR     TI              ; clear TI flag
        MOV     SBUF,B          ; send IDS
aks3:   JNB     TI,aks3         ; wait until byte sent
        CLR     TI              ; clear TI flag
        MOV     SBUF,#FEND      ; send 2nd FEND
aks4:   JNB     TI,aks4         ; wait until byte sent
        ACALL   txrst           ; reset TX state
        SETB    RFRCV           ; turn FCS LED off
        SETB    PCRCV           ; turn PC LED off
        CLR     TI              ; clear TI flag
        CLR     RI              ; clear RI flag
        SETB    ES              ; enable serial interrupts
aks_d:  RET                     ; send ACK message done

rxrst:  CLR     A               ; clear A
        MOV     RXBH,A          ; clear buffer
        MOV     RXBL,A          ; clear buffer
        MOV     RXBB,A          ; clear buffer
        MOV     RMBYC,A         ; clear RX byte count
        MOV     RMFCC,A         ; clear loop counter
        MOV     R0,#RXMB        ; point R0 to message start
        CLR     OKFLG           ; clear FCS OK flag
        CLR     SOPFLG          ; enable SOP test
        SETB    RXI             ; turn RXI LED off
rxr_d:  RET                     ; RX reset done

b_rfcs: MOV     RMLPC,#8        ; load loop count of 8
brf0:   CLR     C               ; clear carry bit
        MOV     A,RMFCS         ; load RX message byte
        RRC     A               ; shift lsb into carry
        MOV     RMFCS,A         ; store shifted message byte
        MOV     RM,C            ; load RM with lsb
        CLR     C               ; clear carry bit
        MOV     A,R3            ; load high FCS byte
        RRC     A               ; shift right
        MOV     R3,A            ; store shifted high FCS
        MOV     A,R7            ; load low FCS byte
        RRC     A               ; shift and pull in bit for FCS high
        MOV     R7,A            ; store shifted low FCS
        JNB     RM,brf1         ; if lsb of low FCS = 0, jump to brf1
        CPL     C               ; else complement carry bit
brf1:   JNC     brf2            ; if RM XOR (low FCS lsb) = 0 jump to brf2
        MOV     A,R3            ; else load high FCS
        XRL     A,#FCSH         ; and XOR with high FCS poly
        MOV     R3,A            ; store high FCS
        MOV     A,R7            ; load low FCS
        XRL     A,#FCSL         ; XOR with low FCS poly
        MOV     R7,A            ; store low FCS
brf2:   DJNZ    RMLPC,brf0      ; loop through bits in message byte
brfcs_d: RET                    ; done this pass

a_rfcs: MOV     A,R3            ; load FCS high
        XRL     A,#FCVH         ; compare with 0F0H
        JNZ     arf0            ; if <> 0 jump to arf0
        MOV     A,R7            ; load FCS low
        XRL     A,#FCVL         ; else compare with 0B8H
        JNZ     arf0            ; if <> 0 jump to arf0
        CLR     RFRCV           ; else turn FCS LED on
        SETB    OKFLG           ; set FCS OK flag
arf0:   MOV     R3,#FCSS        ; reseed FCS high
        MOV     R7,#FCSS        ; reseed FCS low
arfcs_d: RET                    ; RX FCS done

srio:   PUSH    PSW             ; save
        PUSH    ACC             ; environment
        JNB     TI,sr_0         ; skip if not TI flag
        CLR     TI              ; else clear TI flag
sr_0:   JNB     RI,sr_1         ; skip if not RI flag
        CLR     RI              ; and clear RI flag
        JNB     SIFLG,sr_1      ; skip if serial in inactive
        CLR     PCRCV           ; else turn PC LED on
```

```
        ACALL   do_tx          ;  get & transmit message from host
        SETB    PCRCV          ;  turn PC LED off
sr_1:   POP     ACC            ;  restore
        POP     PSW            ;  environment
        RET                    ;  serial in done

do_as:  CLR     PLLON          ;  idle RX PLL
        ACALL   hello2         ;  get AutoSend message
        ACALL   txfcs          ;  build and add FCS
        ACALL   txpre          ;  send TX preamble
        ACALL   txmsg          ;  send TX message
        ACALL   txrst          ;  reset TX
        SETB    PLLON          ;  enable RX PLL
        RET                    ;  TX message done

do_tx:  ACALL   txget          ;  get TX message from host
        JNB     TXFLG,do1      ;  skip if send TX idle
        CLR     PLLON          ;  else idle RX PLL
        ACALL   txfcs          ;  build and add FCS
        ACALL   txpre          ;  send TX preamble
        ACALL   txmsg          ;  send TX message
        INC     TXCNT          ;  increment TX count
do1:    ACALL   txrst          ;  reset TX
        SETB    PLLON          ;  enable RX PLL
        RET                    ;  TX message done

do_rt:  CLR     PLLON          ;  idle RX PLL
        ACALL   txpre          ;  send TX preamble
        ACALL   txmsg          ;  send TX message
        INC     TXCNT          ;  increment TX count
        ACALL   txrst          ;  reset TX
        SETB    PLLON          ;  enable RX PLL
        RET                    ;  TX message done

txget:  MOV     A,SBUF         ;  get byte
        MOV     TMBYT,A        ;  copy to TMBYT
        XRL     A,#FEND        ;  compare to FEND
        JZ      txg0           ;  if FEND jump to txg0
        AJMP    txg_d          ;  else done
txg0:   MOV     @R1,TMBYT      ;  store 1st FEND
        INC     TMBYC          ;  bump TX byte counter
txg1:   MOV     TMFCC,#0       ;  reset timeout counter
        SETB    TOFLG          ;  set timeout flag
        CLR     RI             ;  clear RI flag
txg2:   JNB     TOFLG,txg3     ;  if TOFLG reset jump to txg3
        JNB     RI,txg2        ;  else loop until next byte
        CLR     RI             ;  clear RI flag
        CLR     TOFLG          ;  clear TOFLG
        AJMP    txg4           ;  and jump to txg4
txg3:   MOV     TMBYC,#2       ;  look like null message
        AJMP    txg6           ;  and jump to txg6
txg4:   MOV     A,SBUF         ;  get byte
        MOV     TMBYT,A        ;  copy to TMBYT
        INC     TMBYC          ;  bump byte counter
        INC     R1             ;  bump pointer R1
        MOV     @R1,TMBYT      ;  store byte
        MOV     A,TMBYC        ;  load counter
        CLR     C              ;  clear carry
        SUBB    A,#28          ;  test for 28 bytes
        JZ      txg5           ;  if 28 handle overflow at txg5
        MOV     A,TMBYT        ;  else load byte
        CJNE    A,#FEND,txg1   ;  if <> FEND loop to txg1
        AJMP    txg6           ;  else jump to txg6 on 2nd FEND
txg5:   MOV     @R1,#FEND      ;  force 2nd FEND
txg6:   MOV     R1,#TXMB       ;  reset TX message pointer
        MOV     A,TMBYC        ;  get byte count
        CJNE    A,#2,txg7      ;  if <> 2 jump to txg7
        MOV     TMBYC,#0       ;  else reset byte counter
        AJMP    txg_d          ;  jump to txg_d
txg7:   CLR     SIFLG          ;  idle serial in
        CLR     TOFLG          ;  clear timeout flag
        SETB    TXFLG          ;  set TX active flag
        MOV     TFBUF,TFTX     ;  update local TO/FROM buffer
        MOV     IDBUF,IDTX     ;  update local ID buffer
        CLR     TI             ;  clear TI flag
        MOV     SBUF,#FEND     ;  send 1st FEND
txg8:   JNB     TI,txg8        ;  wait until byte sent
        CLR     TI             ;  clear TI flag
        MOV     SBUF,#255      ;  send PAK byte
txg9:   JNB     TI,txg9        ;  wait until byte sent
        CLR     TI             ;  clear TI flag
```

```
        MOV     SBUF,#FEND      ;  send 2nd FEND
txgA:   JNB     TI,txgA         ;  wait until byte sent
        CLR     TI              ;  clear TI flag
txg_d:  RET                     ;  get TX message done

txfcs:  INC     TMBYC           ;  # bytes including FCS
        MOV     @R1,TMBYC       ;  replace 1st FEND with # bytes
        MOV     TMFCC,TMBYC     ;  move byte count to loop counter
        DEC     TMFCC           ;  loop count is 2 less
        DEC     TMFCC           ;  than # bytes including FCS
txf0:   MOV     TMFCS,@R1       ;  get next message byte
        INC     R1              ;  bump pointer
        ACALL   b_tfcs          ;  build FCS
        DJNZ    TMFCC,txf0      ;  loop for next byte
        ACALL   a_tfcs          ;  add FCS
        MOV     R1,#TXMB        ;  reset TX message pointer
        JB      ASFLG,txf1      ;  skip if AutoSend
        MOV     DPTR,#delay     ;  point to delay table
        MOV     A,TL1           ;  get random table offset
        ANL     A,#07H          ;  mask upper 5 bits
        MOVC    A,@A+DPTR       ;  load table byte
        MOV     TXTH,A          ;  into TX delay high
        AJMP    txf2            ;  skip AutoSend delay
txf1:   MOV     TXTH,#TXR0      ;  load AutoSend delay
txf2:   MOV     TXTL,#0         ;  clear TX delay low
        SETB    TMFLG           ;  set TX message flag
txf_d:  RET                     ;  TX FCS done

txpre:  CLR     PTT             ;  turn PTT on
        MOV     B,#200          ;  load PTT delay count
txp0:   DJNZ    B,txp0          ;  loop to delay
txp1:   MOV     DPTR,#tstrt     ;  point to TX start table
        MOV     B,#0            ;  clear B
        MOV     A,B             ;  B holds table offset
        MOVC    A,@A+DPTR       ;  load table entry
        MOV     TMBYT,A         ;  into TMBYT
        MOV     TMBIC,#4        ;  load bit count
        MOV     TXSMC,#0        ;  clear sample count
        SETB    TSFLG           ;  turn TX sample out on
txp2:   MOV     A,TXSMC         ;  get sample count
        JNZ     txp2            ;  loop until sample count 0
        MOV     A,TMBIC         ;  get bit count
        JNZ     txp3            ;  if <> 0 jump to txp3
        MOV     A,B             ;  else get current offset (0 to 11)
        CLR     C               ;  clear carry
        SUBB    A,#11           ;  subtract ending offset
        JZ      txp_d           ;  if 0  done
        INC     B               ;  else bump byte count
        MOV     A,B             ;  get count/offset
        MOVC    A,@A+DPTR       ;  load table entry
        MOV     TMBYT,A         ;  into TMBYT
        MOV     TMBIC,#4        ;  reload bit count
txp3:   MOV     A,TMBYT         ;  get TX message byte
        CLR     C               ;  clear carry
        RRC     A               ;  shift right into carry
        MOV     TXBIT,C         ;  load next bit
        MOV     TMBYT,A         ;  store shifted message byte
        DEC     TMBIC           ;  decrement bit count
        MOV     TXSMC,#8        ;  reload sample count
        AJMP    txp2            ;  loop again
txp_d:  RET                     ;  TX preamble done

txmsg:  MOV     B,#1            ;  count 1st byte sent
        MOV     A,@R1           ;  get 1st TX message byte
        MOV     TMBYT,A         ;  into TMBYT
        MOV     DPTR,#smbl      ;  point to symbol table
        ANL     A,#0FH          ;  clean offset
        MOVC    A,@A+DPTR       ;  get 6-bit symbol
        MOV     TXSL,A          ;  move to TXSL
        MOV     A,TMBYT         ;  get TMBYT
        SWAP    A               ;  swap nibbles
        ANL     A,#0FH          ;  clean offset
        MOVC    A,@A+DPTR       ;  get 6-bit symbol
        MOV     TXSH,A          ;  move to TXSH
        MOV     TMBIC,#12       ;  set bit count to 12
        MOV     TXSMC,#0        ;  clear sample count
txm0:   MOV     A,TXSMC         ;  get sample count
        JNZ     txm0            ;  loop until sample count 0
        MOV     A,TMBIC         ;  get bit count
        CLR     C               ;  clear carry
        SUBB    A,#7            ;  subtract 7
```

```
            JNC     txm1            ;  if => 7 jump to txm1
            MOV     A,TMBIC         ;  else get bit count
            JNZ     txm2            ;  if > 0 jump to txm2
            MOV     A,B             ;  else get current byte number
            CLR     C               ;  clear carry
            SUBB    A,TMBYC         ;  subtract TX message byte count
            JZ      txm3            ;  if 0  done
            INC     R1              ;  else bump byte pointer
            INC     B               ;  and bump byte counter
            MOV     A,@R1           ;  get next byte
            MOV     TMBYT,A         ;  into TMBYT
            MOV     DPTR,#smbl      ;  point to symbol table
            ANL     A,#0FH          ;  offset
            MOVC    A,@A+DPTR       ;  get 6-bit symbol
            MOV     TXSL,A          ;  move to TXSL
            MOV     A,TMBYT         ;  get TMBYT
            SWAP    A               ;  swap nibbles
            MOV     DPTR,#smbl      ;  point to symbol table
            ANL     A,#0FH          ;  clean offset
            MOVC    A,@A+DPTR       ;  get 6-bit symbol
            MOV     TXSH,A          ;  move to TXSH
            MOV     TMBIC,#12       ;  set bit count to 12
txm1:       MOV     A,TXSL          ;  get low TX symbol
            CLR     C               ;  clear carry
            RRC     A               ;  shift right into carry
            MOV     TXBIT,C         ;  load next bit
            MOV     TXSL,A          ;  store shifted message byte
            DEC     TMBIC           ;  decrement bit count
            MOV     TXSMC,#8        ;  reload sample count
            AJMP    txm0            ;  loop again
txm2:       MOV     A,TXSH          ;  get high TX symbol
            CLR     C               ;  clear carry
            RRC     A               ;  shift right into carry
            MOV     TXBIT,C         ;  load next bit
            MOV     TXSH,A          ;  store shifted message byte
            DEC     TMBIC           ;  decrement bit count
            MOV     TXSMC,#8        ;  reload sample count
            AJMP    txm0            ;  loop again
txm3:       CLR     TSFLG           ;  clear TX sample out flag
            CLR     TXPIN           ;  clear TX out pin
            SETB    PTT             ;  turn PTT off
txm_d:      RET                     ;  TX message done

txrst:      CLR     TMFLG           ;  clear TX message flag
            CLR     AMFLG           ;  clear AutoSend message flag
            CLR     A               ;  reset for next TX
            MOV     TMBYT,A         ;  clear TX message byte
            MOV     TMFCC,A         ;  clear TX FCS count
            MOV     TXSMC,A         ;  clear TX out count
            MOV     TXSL,A          ;  clear TX symbol low
            MOV     TXSH,A          ;  clear TX symbol high
            MOV     R1,#TXMB        ;  point R1 to message start
            JB      ASFLG,txr_d     ;  skip if in AutoSend
            JB      TXFLG,txr_d     ;  skip if send TX active
            MOV     TMBYC,A         ;  reset TX message byte count
            MOV     TXCNT,A         ;  reset TX retry count
            MOV     TXTL,A          ;  clear TX timer low
            MOV     TXTH,A          ;  clear TX timer high
            SETB    SIFLG           ;  enable serial in
txr_d:      RET                     ;  TX reset done

b_tfcs:     MOV     B,#8            ;  load loop count of 8
btf0:       CLR     C               ;  clear carry bit
            MOV     A,TMFCS         ;  load TX message byte
            RRC     A               ;  shift lsb into carry
            MOV     TMFCS,A         ;  store shifted message byte
            MOV     TM,C            ;  load TM with lsb
            CLR     C               ;  clear carry bit
            MOV     A,R5            ;  load high FCS byte
            RRC     A               ;  shift right
            MOV     R5,A            ;  store shifted high FCS
            MOV     A,R6            ;  load low FCS byte
            RRC     A               ;  shift and pull in bit for FCS high
            MOV     R6,A            ;  store shifted low FCS
            JNB     TM,btf1         ;  if lsb of low FCS = 0, jump to btf1
            CPL     C               ;  else complement carry bit
btf1:       JNC     btf2            ;  if TM XOR (low FCS lsb) = 0 jump to btf2
            MOV     A,R5            ;  else load high FCS
            XRL     A,#FCSH         ;  and XOR with high FCS poly
            MOV     R5,A            ;  store high FCS
            MOV     A,R6            ;  load low FCS
```

```
            XRL       A,#FCSL        ;  XOR with low FCS poly
            MOV       R6,A           ;  store low FCS
btf2:       DJNZ      B,btf0         ;  loop through bits in message byte
btfcs_d:    RET                      ;  done this pass

a_tfcs:     MOV       A,R6           ;  load FCS (high/low switch)
            CPL       A              ;  1's complement
            MOV       @R1,A          ;  store at end of TX message
            INC       R1             ;  increment TX message byte pointer
            MOV       A,R5           ;  load FCS (high/low switch)
            CPL       A              ;  1's complement
            MOV       @R1,A          ;  store at end of TX message
            MOV       R5,#FCSS       ;  reseed FCS high
            MOV       R6,#FCSS       ;  reseed FCS low
atfcs_d:    RET                      ;  add TX FCS done

setup:      CLR       EA             ;  disable interrupts
            SETB      PTT            ;  turn PTT off
            CLR       TXPIN          ;  turn TX modulation off
tick_su:    MOV       TMOD,#ITMOD    ;  set timers T0 and T1 to mode 2
            CLR       TR0            ;  stop timer T0
            CLR       TF0            ;  clear T0 overflow
            MOV       TH0,#ITICK     ;  load count for 62.40 us tick
            MOV       TL0,#ITICK     ;  load count for 62.40 us tick
            SETB      TR0            ;  start timer T0
            SETB      ET0            ;  unmask T0 interrupt
uart_su:    SETB      MAX            ;  power up Maxim RS232 converter
            CLR       TR1            ;  stop timer T1
            CLR       TF1            ;  clear T1 overflow
            MOV       TH1,#IBAUD     ;  load baud rate count
            MOV       TL1,#IBAUD     ;  load baud rate count
            MOV       PCON,#ISMOD    ;  SMOD = 1 for baud rate @ 22.1184 MHz
            SETB      TR1            ;  start baud rate timer T1
            MOV       SCON,#ISCON    ;  enable UART mode 1
            MOV       A,SBUF         ;  clear out UART RX buffer
            CLR       A              ;  clear A
            CLR       RI             ;  clear RI (byte received) flag
            CLR       TI             ;  clear TI (byte sent) flag
            ACALL     hello          ;  send start up message
            ACALL     initr          ;  initialize TX & RX
            MOV       TXTH,#TXR0     ;  load default AutoSend delay
            SETB      SIFLG          ;  set serial in flag active
            MOV       C,ID3          ;  read ID3
            JC        as_set         ;  skip if no ID3 jumper
            SETB      NHFLG          ;  else set no FEND/header flag
as_set:     MOV       C,ID0          ;  read ID0
            JC        ser_on         ;  skip if no ID0 jumper
            ACALL     hello2         ;  else do AutoSend
ser_on:     SETB      ES             ;  enable serial ISR
isr_on:     SETB      EA             ;  enable interrupts
            SETB      PLLON          ;  activate RX PLL
setup_d:    RET                      ;  setup done

initr:      ANL       BOOT,#1        ;  warm boot (don't reset WBFLG)
            MOV       R0,#35         ;  starting here
            MOV       B,#93          ;  for 93 bytes
            CLR       A              ;  clear A
clr_r:      MOV       @R0,A          ;  clear RAM
            INC       R0             ;  bump RAM pointer
            DJNZ      B,clr_r        ;  loop again
            MOV       R0,#RXMB       ;  load RX buffer pointer
            MOV       R1,#TXMB       ;  load TX buffer pointer
            MOV       R2,A           ;  clear R2
            MOV       R3,#FCSS       ;  seed R3
            MOV       R5,#FCSS       ;  seed R5
            MOV       R6,#FCSS       ;  seed R6
            MOV       R7,#FCSS       ;  seed R7
            MOV       TFBUF,#34      ;  initialize TO/FROM 2 & 2
            MOV       IDBUF,#3       ;  initialize ID = 3
            CLR       SOPFLG         ;  clear SOPFLG
            SETB      PT0            ;  tick is 1st priority
ini_d:      RET                      ;  done

hello:      MOV       DPTR,#table    ;  point to table
            MOV       B,#13          ;  load loop count in B
            MOV       R7,#0          ;  R7 has 1st table entry
snd_h:      MOV       A,R7           ;  move table offset into A
            MOVC      A,@A+DPTR      ;  load table byte
            CLR       TI             ;  clear TI flag
            MOV       SBUF,A         ;  send byte
```

```
nxt_tx:   JNB     TI,nxt_tx      ;  wait until sent
          INC     R7             ;  bump index
          DJNZ    B,snd_h        ;  loop to send message
hello_d:  RET                    ;  done

hello2:   MOV     DPTR,#tbl_2    ;  point to table 2
          MOV     R1,#TXMB       ;  reset TX buffer pointer
          MOV     B,#10          ;  loop count for 9 bytes
          MOV     TMBYC,#0       ;  offset for 1st table entry
snd_h2:   MOV     A,TMBYC        ;  move table offset into A
          MOVC    A,@A+DPTR      ;  load table byte
          MOV     @R1,A          ;  into TX buffer
          INC     TMBYC          ;  increment TMBYC
          INC     R1             ;  increment R1
          DJNZ    B,snd_h2       ;  loop to load message
          MOV     R1,#TXMB       ;  reset TX pointer
          CLR     SIFLG          ;  reset serial input
          SETB    ASFLG          ;  set AutoSend flag
helo2_d   RET

;  tables:

tstrt:    .BYTE   10             ;  preamble/SOP table
          .BYTE   10             ;  table data
          .BYTE   10             ;  table data
          .BYTE   10             ;  table data
          .BYTE   10             ;  table data
          .BYTE   10             ;  table data
          .BYTE   10             ;  table data
          .BYTE   10             ;  table data
          .BYTE   10             ;  table data
          .BYTE   8              ;  table data
          .BYTE   3              ;  table data
          .BYTE   11             ;  table data

smbl:     .BYTE   13             ;  4-to-6 bit table
          .BYTE   14             ;  table data
          .BYTE   19             ;  table data
          .BYTE   21             ;  table data
          .BYTE   22             ;  table data
          .BYTE   25             ;  table data
          .BYTE   26             ;  table data
          .BYTE   28             ;  table data
          .BYTE   35             ;  table data
          .BYTE   37             ;  table data
          .BYTE   38             ;  table data
          .BYTE   41             ;  table data
          .BYTE   42             ;  table data
          .BYTE   44             ;  table data
          .BYTE   50             ;  table data
          .BYTE   52             ;  table data
          .BYTE   00             ;  overflow

delay:    .BYTE   020H           ;  0.50 second
          .BYTE   044H           ;  1.10 second
          .BYTE   032H           ;  0.80 second
          .BYTE   058H           ;  1.40 second
          .BYTE   028H           ;  0.65 second
          .BYTE   04EH           ;  1.25 second
          .BYTE   03CH           ;  0.95 second
          .BYTE   062H           ;  1.55 second

table:    .BYTE   192            ;  start up message
          .BYTE   34             ;  table data
          .BYTE   3              ;  table data
          .BYTE   'D'            ;  table data
          .BYTE   'K'            ;  table data
          .BYTE   '2'            ;  table data
          .BYTE   '0'            ;  table data
          .BYTE   '0'            ;  table data
          .BYTE   'A'            ;  table data
          .BYTE   ':'            ;  table data
          .BYTE   ' '            ;  table data
          .BYTE   ' '            ;  table data
          .BYTE   192            ;  table data

tbl_2:    .BYTE   192            ;  table data
          .BYTE   34             ;  table data
          .BYTE   3              ;  table data
          .BYTE   'H'            ;  table data
          .BYTE   'e'            ;  table data
```

```
        .BYTE     'l'           ;  table data
        .BYTE     'l'           ;  table data
        .BYTE     'o'           ;  table data
        .BYTE     ' '           ;  table data
        .BYTE     192           ;  table data

        .END                    ;  end of source code
```

5.2 V110T30C.FRM

```
VERSION 5.00
Object = "{648A5603-2C6E-101B-82B6-000000000014}#1.1#0"; "MSCOMM32.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Begin VB.Form Form1
   Caption         =   "V110T30C Terminal Program for DK200A Protocol - 2002.08.07 Rev"
   ClientHeight    =   5235
   ClientLeft      =   225
   ClientTop       =   630
   ClientWidth     =   7785
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   ScaleHeight     =   5951.697
   ScaleMode       =   0  'User
   ScaleWidth      =   7905
   Begin MSComctlLib.ProgressBar ProgressBar1
      Height          =   251
      Left            =   1162
      TabIndex        =   3
      Top             =   4934
      Width           =   4875
      _ExtentX        =   8599
      _ExtentY        =   450
      _Version        =   393216
      Appearance      =   0
      Scrolling       =   1
   End
   Begin MSComctlLib.StatusBar StatusBar1
      Align           =   2  'Align Bottom
      Height          =   375
      Left            =   0
      TabIndex        =   2
      Top             =   4860
      Width           =   7785
      _ExtentX        =   13732
      _ExtentY        =   661
      _Version        =   393216
      BeginProperty Panels {8E3867A5-8586-11D1-B16A-00C0F0283628}
         NumPanels       =   4
         BeginProperty Panel1 {8E3867AB-8586-11D1-B16A-00C0F0283628}
            Alignment       =   1
            Bevel           =   0
            Object.Width    =   148
            MinWidth        =   148
         EndProperty
         BeginProperty Panel2 {8E3867AB-8586-11D1-B16A-00C0F0283628}
            Alignment       =   1
            Object.Width    =   1737
            MinWidth        =   1737
            Text            =   "TX Buffer"
            TextSave        =   "TX Buffer"
         EndProperty
         BeginProperty Panel3 {8E3867AB-8586-11D1-B16A-00C0F0283628}
            Object.Width    =   8755
            MinWidth        =   8755
         EndProperty
         BeginProperty Panel4 {8E3867AB-8586-11D1-B16A-00C0F0283628}
            Alignment       =   1
            Text            =   "Keyboard"
            TextSave        =   "Keyboard"
         EndProperty
      EndProperty
   End
   Begin MSComDlg.CommonDialog CommonDialog1
      Left            =   240
      Top             =   4320
      _ExtentX        =   688
      _ExtentY        =   688
```

```
   _Version        =   393216
End
Begin VB.TextBox Text2
   Height          =   2323
   Left            =   148
   Locked          =   -1  'True
   MultiLine       =   -1  'True
   ScrollBars      =   2  'Vertical
   TabIndex        =   1
   Top             =   0
   Width           =   7460
End
Begin VB.Timer Timer1
   Left            =   720
   Top             =   4320
End
Begin MSCommLib.MSComm MSComm1
   Left            =   1200
   Top             =   4320
   _ExtentX        =   794
   _ExtentY        =   794
   _Version        =   393216
   DTREnable       =   -1  'True
End
Begin VB.TextBox Text1
   Height          =   2323
   Left            =   120
   MultiLine       =   -1  'True
   ScrollBars      =   2  'Vertical
   TabIndex        =   0
   Top             =   2513
   Width           =   7460
End
Begin VB.Menu mnuFile
   Caption         =   "&File"
   Begin VB.Menu mnuExit
      Caption         =   "E&xit"
   End
End
Begin VB.Menu mnuEdit
   Caption         =   "&Edit"
   Begin VB.Menu mnuToAdr
      Caption         =   "To Address"
      Begin VB.Menu mnuTN1
         Caption         =   "Node 1"
      End
      Begin VB.Menu mnuTN2
         Caption         =   "Node 2"
         Checked         =   -1  'True
      End
      Begin VB.Menu mnuTN3
         Caption         =   "Node 3"
      End
      Begin VB.Menu mnuTN4
         Caption         =   "Node 4"
      End
   End
   Begin VB.Menu mnuFrmAdr
      Caption         =   "From Address"
      Begin VB.Menu mnuFN1
         Caption         =   "Node 1"
      End
      Begin VB.Menu mnuFN2
         Caption         =   "Node 2"
         Checked         =   -1  'True
      End
      Begin VB.Menu mnuFN3
         Caption         =   "Node 3"
      End
      Begin VB.Menu mnuFN4
         Caption         =   "Node 4"
      End
   End
End
Begin VB.Menu mnuView
   Caption         =   "&View"
   Begin VB.Menu mnuClear
      Caption         =   "&Clear"
   End
```

```
      Begin VB.Menu mnuDups
         Caption         =   "Show RX &Dups"
         Checked         =   -1  'True
      End
      Begin VB.Menu mnuShw
         Caption         =   "&Show ACK/NAK"
         Checked         =   -1  'True
      End
      Begin VB.Menu mnuAutoSnd
         Caption         =   "&AutoSend"
      End
   End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

' V110T30C.FRM, 2002.08.07 @ 08:00 CDT
' See RFM Virtual Wire(r) Development Kit Warranty & License for terms of use
' Tutorial software - NO representation is made that this software
' is suitable for any purpose
' Copyright(c) 2000-2002, RF Monolithics, Inc.
' For experimental use with the RFM DR1200A-DK and DR1201A-DK
' and DR1300A-DK ASH Transceiver Virtual Wire(R) Development Kits
' For protocol software version DK200A.ASM
' Check www.rfm.com for latest software updates
' Compiled in Microsoft Visual Basic 6.0

' global variables:
  Dim ComData$                                            ' com input string
  Dim ComTime!                                            ' com input reference time
  Dim KeyIn$                                              ' keystroke input buffer
  Dim TXFlag As Integer                                   ' send TX message flag
  Dim TNFlag As Integer                                   ' send next TX packet flag
  Dim TPkt$                                               ' keyboard input string
  Dim TSPkt$                                              ' SLIP encoded input string
  Dim TXPkt$                                              ' transmit message string
  Dim SPkt$                                               ' transmit packet string
  Dim TFlag As Integer                                    ' packet transfer flag
  Dim ANFlag As Integer                                   ' ACK/NAK flag
  Dim TCnt As Integer                                     ' TX timeout counter
  Dim XCnt As Integer                                     ' TX transfer retry counter
  Dim Temp$                                               ' temp string buffer
  Dim Temp1$                                              ' temp1 string buffer
  Dim FRM  As Integer                                     ' RX From address
  Dim ID As Integer                                       ' RX packet ID
  Dim DupFltr As Integer                                  ' duplicate RX filter flag
  Dim PID(15) As Integer                                  ' packet ID array (dup/skip detector)
  Dim DpSkp As Integer                                    ' dup/skip status
  Dim pSLIP As Integer                                    ' SLIP pointer
  Dim G As Integer                                        ' ID compare
  Dim I As Integer                                        ' general purpose index/counter
  Dim K As Integer                                        ' SLIP encoded packet length
  Dim N As Integer                                        ' keyboard byte counter
  Dim P As Integer                                        ' TX packet ID #, 1 - 7
  Dim FEND$                                               ' SLIP framing character
  Dim ESC$                                                ' SLIP escape character
  Dim TFEND$                                              ' SLIP transpose frame
  Dim TESC$                                               ' SLIP transpose escape
  Dim PktHdr$                                             ' packet header
  Dim J As Integer                                        ' FEND$ string position
  Dim Q As Integer                                        ' RPkt$ length
  Dim RPkt$                                               ' RX message FIFO string
  Dim R2Pkt$                                              ' RX message display string
  Dim ASFlag As Integer                                   ' AutoSend enable flag
  Dim NAFlag As Integer                                   ' AutoSend next message flag
  Dim InDel!                                              ' delay for com input
  Dim PCnt As Integer                                     ' packet TX tries counter
  Dim ShwACK As Integer                                   ' show ACK/NAK flag
  Dim TNode As Integer                                    ' To node numeric value
  Dim FNode As Integer                                    ' From node numeric value
  Dim TF As Integer                                       ' To/From node numeric value
  Dim ASStr$                                              ' AutoSend string

Private Sub Form_Load()

' initialize variables:
  ComData$ = ""                                           ' clear string
  ComTime! = 0                                            ' clear reference time
```

```
  KeyIn$ = ""                                          ' clear keystroke buffer
  TXFlag = 0                                           ' clear TX message flag
  TNFlag = 0                                           ' clear next TX packet flag
  TPkt$ = ""                                           ' clear TX packet string
  TSPkt$ = ""                                          ' clear SLIP encoded string
  TXPkt$ = ""                                          ' clear TX message string
  SPkt$ = ""                                           ' clear send packet string
  TFlag = 0                                            ' clear transfer flag
  ANFlag = 0                                           ' clear ACK/NAK flag
  TCnt = 0                                             ' clear TX timeout counter
  XCnt = 0                                             ' clear transfer counter
  Temp$ = ""                                           ' clear temp string buffer
  Temp1$ = ""                                          ' clear 2nd temp string buffer
  FRM = 0                                              ' set RX From to 0
  ID = 0                                               ' set RX packet ID to 0
  DupFltr = 0                                          ' clear duplicate filter
  pSLIP = 0                                            ' clear SLIP pointer
  G = 0                                                ' clear ID compare
  I = 0                                                ' clear index/counter
  K = 0                                                ' clear SLIP packet length
  N = 0                                                ' clear keyboard byte counter
  P = 3                                                ' set packet ID to 3
  FEND$ = Chr$(192)                                    ' initialize SLIP framing character
  ESC$ = Chr$(219)                                     ' initialize SLIP escape character
  TFEND$ = Chr$(220)                                   ' initialize SLIP transpose frame
  TESC$ = Chr$(221)                                    ' initialize SLIP transpose escape
  PktHdr$ = Chr$(34)                                   ' set To/From default = 2/2
  J = 0                                                ' clear string position
  Q = 0                                                ' clear string length
  RPkt$ = ""                                           ' clear RX FIFO string
  R2Pkt$ = ""                                          ' clear RX display string
  ASFlag = 0                                           ' clear AutoSend flag
  NAFlag = 0                                           ' clear next AutoSend flag
  PCnt = 0                                             ' clear TX tries counter
  ShwACK = 1                                           ' set show ACK/NAK flag
  TNode = 2                                            ' set To node default = 2
  FNode = 2                                            ' set From node default = 2
  TF = 34                                              ' set TF default = 34
  For B = 0 To 15
    PID(B) = -1                                        ' set PID array elements = -1
  Next B

  ASStr$ = "**Auto Test Message**" & vbCrLf            ' default AutoSend message

  Form1.Left = (Screen.Width - Form1.Width) / 2        ' center form left-right
  Form1.Top = (Screen.Height - Form1.Height) / 2       ' center form top-bottom
  Text1.BackColor = QBColor(0)                         ' black background
  Text1.ForeColor = QBColor(15)                        ' white letters
  Text1.FontSize = 10                                  ' 10 point font
  Text2.BackColor = QBColor(0)                         ' black background
  Text2.ForeColor = QBColor(15)                        ' white letters
  Text2.FontSize = 10                                  ' 10 point font

  MSComm1.CommPort = 1                                 ' initialize com port
  MSComm1.Settings = "19200,N,8,1"                     ' at 19.2 kbps
  MSComm1.RThreshold = 0                               ' poll only, no interrupts
  MSComm1.InputLen = 0                                 ' read all bytes
  MSComm1.PortOpen = True                              ' open com port
  InDel! = 0.1                                         ' initialize get com delay at 100 ms

  StatusBar1.Panels(4).Text = "Keyboard Active"        ' keyboard active status message
  ProgressBar1.Min = 0                                 ' progress bar min number of TX bytes
  ProgressBar1.Max = 240                               ' progress bar max number of TX bytes

  Show                                                 ' show form
  Text1.Text = "**TX Message Window**" & vbCrLf        ' 1st line of TX start up message
  Text1.Text = Text1.Text & "**Set for Node 2 & 2**" _
  & vbCrLf & vbCrLf                                    ' 2nd line of TX start up message
  Text1.SelStart = Len(Text1.Text)                     ' put cursor at end of text
  Text2.Text = "**RX Message Window**" & vbCrLf        ' RX start up message
  Text2.SelStart = Len(Text2.Text)                     ' put cursor at end of text

  Randomize                                            ' initialize random # generator

  Timer1.Interval = 300                                ' 300 ms timer interval
  Timer1.Enabled = True                                ' start timer

End Sub
```

```
Private Sub Timer1_Timer()
  If ANFlag = 1 Then                                     ' if ACK/NAK flag set
    Call Xfer                                            ' call Xfer (detect switch OFF, etc.)
  End If
  If MSComm1.InBufferCount > 0 Then                      ' if com input buffer has bytes
    Call RxPkt                                           ' call RxPkt
  End If
  If TXFlag = 1 Then                                     ' if TX message flag set
    If TNFlag = 1 Then                                   ' and next TX packet flag set
      Call SndPkt                                        ' call SndPkt
    End If
  End If
  If ASFlag = 1 Then                                     ' if AutoSend flag set
    If TXFlag = 0 Then                                   ' and TX message flag clear
      Call ASPkt                                         ' call AutoSend
    End If
  End If
End Sub

Public Sub RxPkt()
  Call InCom                                             ' InCom gets RX message bytes
  Call ShowPkt                                           ' ShowPkt shows RX message bytes
End Sub

Public Sub InCom()
  On Error Resume Next                                   ' set up error handler
  ComTime! = Timer                                       ' get current time
  Do Until Abs(Timer - ComTime!) > InDel!                ' get bytes for InDel! interval
    Do While MSComm1.InBufferCount > 0                   ' while bytes are in com buffer
     ComData$ = ComData$ & MSComm1.Input                 ' put them in ComData$
    Loop
  Loop
End Sub

Public Sub ShowPkt()
  RPkt$ = RPkt$ & ComData$                               ' add ComData$ bytes to RPkt$ FIFO
  ComData$ = ""                                          ' and clear ComData$
  Do                                                     ' do until FEND$s are gone
    Q = Len(RPkt$)                                       ' Q is RPkt$ packet length
    J = InStr(1, RPkt$, FEND$)                           ' find position of next FEND$
    If (J < 2) Then                                      ' if FEND$ is in the first position
      RPkt$ = Right$(RPkt$, (Q - J))                     ' just delete it
    Else                                                 ' else
      R2Pkt$ = Left$(RPkt$, (J - 1))                     ' R2Pkt$ what's left of this FEND$
      RPkt$ = Right$(RPkt$, (Q - J))                     ' RPkt$ what's right of this FEND$
      If Len(R2Pkt$) = 1 Then                            ' only PAC is a 1 byte message
        If (R2Pkt$ = Chr$(255)) Then                     ' if PAC byte
          TFlag = 0                                      ' reset transfer flag
          If ShwACK = 1 Then                             ' if show ACK/NAK flag set
            Call LenTrap                                 ' manage textbox memory
            Text1.SelStart = Len(Text1.Text)             ' put cursor at end of text
            Text1.SelText = "<Xfer on try " _
            & Str(XCnt + 1) & "> "                       ' show try number for transfer
          End If
          R2Pkt$ = ""                                    ' and clear R2Pkt$
        End If
      ElseIf Len(R2Pkt$) = 2 Then                        ' only ACK/NAK are 2 byte messages
        ANFlag = 0                                       ' reset ACK/NAK flag
        NAFlag = 0                                       ' reset next AutoSend flag
        TNFlag = 1                                       ' set next TX packet flag
        Temp$ = Str((Asc(Left$(R2Pkt$, 1)) And &HF))     ' get From address
        Temp1$ = Str((Int(Asc(Mid$(R2Pkt$, 2, 1)) / 16)) _
        And &H7)                                         ' get packet ID number
        If (Asc(Right$(R2Pkt$, 1)) And &H80) = 128 Then  ' if ACK bit set
          PCnt = (Asc(Right$(R2Pkt$, 1)) And &HF)        ' get ACK retry number
          If ShwACK = 1 Then                             ' if show ACK/NAK flag set
            Call LenTrap                                 ' manage textbox memory
            Text1.SelStart = Len(Text1.Text)             ' put cursor at end of text
            Text1.SelText = "<ACK from N" _
            & Temp$ & " : P" & Temp1$ & " on " _
            & Str(PCnt) & ">" & vbCrLf                   ' show ACK From, ID and retry number
          End If
          R2Pkt$ = ""                                    ' and clear R2Pkt$
        Else
          If ShwACK = 1 Then                             ' if show ACK/NAK flag set
            Call LenTrap                                 ' manage textbox memory
            Text1.SelStart = Len(Text1.Text)             ' put cursor to end of text
            Text1.SelText = "<NAK from N" _
            & Temp$ & " : P" & Temp1$ & ">" & vbCrLf     ' show NAK received
          End If
```

```
          R2Pkt$ = ""                                        ' and clear R2Pkt$
        End If
      ElseIf Len(R2Pkt$) > 2 Then                            ' other messages are > 2 bytes
        Do                                                   ' decode FEND$ escape sequences
          pSLIP = InStr(R2Pkt$, (ESC$ & TFEND$))             ' find position of next ESC$ & TFEND$
          If pSLIP <> 0 Then                                 ' if (ESC$ & TFEND$) present
            K = Len(R2Pkt$)
            If K >= (pSLIP + 2) Then                         ' if escape sequence not last bytes
              R2Pkt$ = Left$(R2Pkt$, (pSLIP - 1)) & FEND$ _
              & Mid$(R2Pkt$, (pSLIP + 2))                    ' replace escape sequence with FEND$
            Else
              R2Pkt$ = Left$(R2Pkt$, (pSLIP - 1)) & FEND$    ' else replace with FEND$ at end
            End If
          Else
            Exit Do                                          ' else done
          End If
        Loop
        Do                                                   ' decode ESC$ escape sequences
          pSLIP = InStr(R2Pkt$, (ESC$ & TESC$))              ' find position of next ESC$ & TESC$
          If pSLIP <> 0 Then                                 ' if (ESC$ & TESC$) string(s) present
            I = Len(R2Pkt$)
            If I >= (pSLIP + 2) Then                         ' if escape sequence not last bytes
              R2Pkt$ = Left$(R2Pkt$, (pSLIP - 1)) & ESC$ _
              & Mid$(R2Pkt$, (pSLIP + 2))                    ' replace escape sequence with ESC$
            Else
              R2Pkt$ = Left$(R2Pkt$, (pSLIP - 1)) & ESC$     ' else replace with ESC$ at end
            End If
          Else
            Exit Do                                          ' else done
          End If
        Loop
        FRM = Asc(Left$(R2Pkt$, 1)) And &HF                  ' get RX packet From address
        ID = Asc(Mid$(R2Pkt$, 2, 1)) And &H7                 ' get RX packet ID
        Call ChkPkt                                          ' check packet for skip/dup
        If DpSkp <> 0 Or DupFltr = 0 Then                    ' if not dup or dup filter off
          If ShwACK = 1 Then                                 ' if show ACK/NAK flag set
            Temp$ = Str(FRM)                                 ' make From address string
            Temp1$ = Str(ID)                                 ' make packet ID string
            R2Pkt$ = Right$(R2Pkt$, (Len(R2Pkt$) - 2))       ' strip off TO/FROM and ID bytes
            If Right$(R2Pkt$, 2) = vbCrLf Then               ' check for vbCrLf
              R2Pkt$ = Left$(R2Pkt$, (Len(R2Pkt$) - 2))      ' remove vbCrLf if present
            ElseIf Right$(R2Pkt$, 1) = Chr$(13) Then         ' also check for a trailing Cr
              R2Pkt$ = Left$(R2Pkt$, (Len(R2Pkt$) - 1))      ' remove Cr if present
            End If
            If Left$(R2Pkt$, 1) = Chr$(10) Then              ' check for a leading Lf
              R2Pkt$ = Right$(R2Pkt$, (Len(R2Pkt$) - 1))     ' remove Lf if present
            End If
            Call LenTrap                                     ' manage textbox memory
            If DpSkp = 1 Then                                ' if skipped packet(s) detected
              Text2.SelStart = Len(Text2.Text)               ' put cursor at end of text
              Text2.SelText = " [PID Skip] "                 ' show where skip(s) occurred
            End If
            Text2.SelStart = Len(Text2.Text)                 ' put cursor at end of text
            Text2.SelText = R2Pkt$ & "  <from N" _
            & Temp$ & " : P" & Temp1$ & ">" & vbCrLf         ' show message, From, ID, new line
            R2Pkt$ = ""                                      ' and clear R2Pkt$
          Else
            R2Pkt$ = Right$(R2Pkt$, (Len(R2Pkt$) - 2))       ' else strip off TO/FROM and ID bytes
            Call LenTrap                                     ' manage textbox memory
            If DpSkp = 1 Then                                ' if skipped packet(s) detected
              Text2.SelStart = Len(Text2.Text)               ' put cursor at end of text
              Text2.SelText = " [PID Skip] "                 ' show where skip(s) occurred
            End If
            Text2.SelStart = Len(Text2.Text)                 ' put cursor at end of text
            Text2.SelText = R2Pkt$                           ' show message
            R2Pkt$ = ""                                      ' and clear R2Pkt$
          End If
        End If
      End If
    End If
  Loop Until (J = 0)                                         ' done when there are no more FEND$s
End Sub

Public Sub ChkPkt()
  G = PID(FRM)                                               ' G is last stored ID
  If G = -1 Then                                             ' if -1 it's the first check
    DpSkp = -1                                               ' so signal no skip/dup
  ElseIf G = ID Then                                         ' else if G = ID it's a dup
    DpSkp = 0                                                ' signal dup
  Else                                                       ' else if G <> to ID
    G = G + 1                                                ' increment G
```

```
    If G > 7 Then                                             ' if greater than 7
      G = 0                                                   ' reset to 0
    End If
    If G = ID Then                                            ' if updated G = ID
      DpSkp = -1                                              ' signal no skip/dup
    Else
      DpSkp = 1                                               ' else signal skip
    End If
  End If
  PID(FRM) = ID                                               ' store current PID for next check
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
  If TXFlag = 0 Then                                          ' if TX message flag reset
    KeyIn$ = Chr$(KeyAscii)                                   ' convert keystroke to character
    If KeyIn$ = Chr$(8) Then                                  ' if it is a backspace from keyboard
      If N > 0 Then                                           ' and if keyboard byte counter > 0
        TPkt$ = Left$(TPkt$, (N - 1))                         ' trim right end of packet
        N = N - 1                                             ' back up byte counter
      End If
    ElseIf KeyIn$ = Chr$(13) Then                             ' else if it is a Cr
      TPkt$ = TPkt$ & vbCrLf                                  ' add vbCrLf to TX packet
      ASStr$ = TPkt$                                          ' update AutoSend string
      N = 0                                                   ' reset keyboard byte counter
      TXFlag = 1                                              ' set TX message flag
      TNFlag = 1                                              ' set next TX packet flag
      StatusBar1.Panels(4).Text = "Keyboard Locked"           ' show keyboard locked
    Else
      TPkt$ = TPkt$ & KeyIn$                                  ' else add byte to TX packet
      N = N + 1                                               ' increment byte counter
    End If
    If (N = 238) Then                                         ' if keyboard byte counter is 238
      TPkt$ = TPkt$ & vbCrLf                                  ' add vbCrLf to TX message
      ASStr$ = TPkt$                                          ' update AutoSend string
      Text1.SelStart = Len(Text1.Text)                        ' place cursor at end
      Text1.SelText = KeyIn$ & vbCrLf                         ' show key input and vbCrLf
      KeyAscii = 0                                            ' block double key display
      N = 0                                                   ' reset keyboard byte counter
      TXFlag = 1                                              ' set TX message flag
      TNFlag = 1                                              ' set next TX packet flag
      StatusBar1.Panels(4).Text = "Keyboard Locked"           ' show keyboard locked
    End If
    Call LenTrap                                              ' manage textbox memory
  Else
    KeyAscii = 0                                              ' block keystroke if TX flag set
  End If
End Sub

Public Sub SndPkt()
  If TNFlag = 1 Then                                          ' if next TX packet flag set
    If TPkt$ <> "" Then                                       ' if TPkt$ has new bytes
      L = Len(TPkt$)                                          ' get number of bytes in TPkt
      For I = 1 To L                                          ' for each byte in TPkt$
        Temp$ = Mid$(TPkt$, I, 1)                             ' load byte in Temp$
        If Temp$ = FEND$ Then                                 ' if byte in Temp$ is a FEND$
          TSPkt$ = TSPkt$ & ESC$ & TFEND$                     ' add ESC$ & TFEND$ to TSPkt$
        ElseIf Temp$ = ESC$ Then                              ' else if byte is an ESC$
          TSPkt$ = TSPkt$ & ESC$ & TESC$                      ' add ESC$ & TESC$ to TSPkt$
        Else
          TSPkt$ = TSPkt$ & Temp$                             ' else just add Temp$ byte to TSPkt$
        End If
      Next I
      TXPkt$ = TXPkt$ & TSPkt$                                ' add new message to TX FIFO
      TPkt$ = ""                                              ' clear new message string
      TSPkt$ = ""                                             ' clear SLIP encoded string
    End If
    If Int(4 * Rnd) > 0 Then                                  ' skip 25% to allow other traffic
      TNFlag = 0                                              ' clear next TX packet flag
      L = Len(TXPkt$)                                         ' get number of bytes in TXPkt$
      If L <= 240 Then                                        ' if less than 240 bytes
        ProgressBar1.Value = L                                ' show number on TX progress bar
      Else
        ProgressBar1.Value = 240                              ' else cap TX progress bar at 240
      End If
      If L > 0 Then                                           ' if TXPkt$ holds bytes
        If L > 24 Then                                        ' and there are more than 24 bytes
          SPkt$ = Left$(TXPkt$, 24)                           ' put the first 24 bytes in SPkt$
          TXPkt$ = Right$(TXPkt$, (L - 24))                   ' and hold the rest in TXPkt$
        Else
          SPkt$ = TXPkt$                                      ' else put all TXPkt$ bytes in SPkt$
          TXPkt$ = ""                                         ' and clear TXPkt$
```

```
        End If
        Call NxtPkt                                       ' bump packet ID number
        SPkt$ = FEND$ & PktHdr$ & Chr$(P) & SPkt$ & FEND$   ' build packet
        MSComm1.Output = SPkt$                            ' send packet
        TFlag = 1                                         ' set transfer flag
        ANFlag = 1                                        ' set ACK/NAK flag
        TCnt = 0                                          ' clear TX timeout counter
        XCnt = 0                                          ' clear TX transfer retry counter
      Else
        TXFlag = 0                                        ' clear TX flag when all bytes sent
        StatusBar1.Panels(4).Text = "Keyboard Active"     ' show keyboard active
      End If
    End If
  End If
End Sub

Public Sub Xfer()
  TCnt = TCnt + 1                                         ' increment TX timeout counter
  If TCnt > 4 Then                                        ' if trying for more than 1 second
    If TFlag = 1 Then                                     ' and transfer flag still set
      TCnt = 0                                            ' reset TCnt
      XCnt = XCnt + 1                                     ' increment transfer retry counter
      If XCnt < 17 Then                                   ' if XCnt not greater than 16
        MSComm1.Output = SPkt$                            ' resend packet
        TCnt = 0                                          ' reset TX timeout counter
      Else
        Call ReSetTX                                      ' else reset TX after eight tries
        Call LenTrap                                      ' manage textbox memory
        Text1.SelStart = Len(Text1.Text)                  ' put cursor to end of text
        Text1.SelText = "  <xfer fault>" & vbCrLf         ' show transfer fault message
      End If
    End If
  End If
  If TCnt > 64 Then                                       ' if more than 16 seconds
    If ANFlag = 1 Then                                    ' and if ACK/NAK flag still set
      Call ReSetTX                                        ' reset TX
      Call LenTrap                                        ' manage textbox memory
      Text1.SelStart = Len(Text1.Text)                    ' put cursor to end of text
      Text1.SelText = "  <ACK/NAK fault>" _
      & vbCrLf                                            ' show ACK/NAK fault message
    End If
  End If
End Sub

Public Sub ReSetTX()
  TFlag = 0                                               ' reset transfer flag
  TXFlag = 0                                              ' reset TX message flag
  TNFlag = 0                                              ' reset next TX packet flag
  ANFlag = 0                                              ' reset ACK/NAK flag
  NAFlag = 0                                              ' reset next AutoSend flag
  TCnt = 0                                                ' reset TCnt
  XCnt = 0                                                ' reset XCnt
  TXPkt$ = ""                                             ' clear TX message string
  SPkt$ = ""                                              ' clear send packet string
  ProgressBar1.Value = 0                                  ' clear progress bar
  StatusBar1.Panels(4).Text = "Keyboard Active"           ' show keyboard active
End Sub

Public Sub ASPkt()
  If NAFlag = 0 Then                                      ' if next AutoSend flag reset
    Call GetPkt                                           ' get next message packet(s)
    Temp$ = TPkt$                                         ' use Temp$ for local display
    Call LenTrap                                          ' manage textbox memory
    Text1.SelStart = Len(Text1.Text)                      ' put cursor at end of text
    Text1.SelText = Temp$                                 ' add text to textbox
    TXFlag = 1                                            ' set TX message flag
    TNFlag = 1                                            ' set next TX packet flag
    StatusBar1.Panels(4).Text = "Keyboard Locked"         ' show keyboard locked
    Call SndPkt                                           ' send via SndPkt
    NAFlag = 1                                            ' set next AutoSend flag
  End If
End Sub

Public Sub GetPkt()
  TPkt$ = ASStr$                                          ' message string for AutoSend
End Sub

Public Sub NxtPkt()
  P = P + 1                                               ' increment packet number
```

```
  If P = 8 Then                                    ' if packet number greater than 7
    P = 0                                          ' reset to 0
  End If
End Sub

Public Sub LenTrap()
  If Len(Text1.Text) > 16000 Then                  ' avoid textbox memory overflow
    Text1.Text = ""                                ' clear TX textbox
    Text1.SelStart = Len(Text1.Text)               ' put cursor at end of text
  End If
  If Len(Text2.Text) > 16000 Then                  ' avoid textbox memory overflow
    Text2.Text = ""                                ' clear RX textbox
    Text2.SelStart = Len(Text2.Text)               ' put cursor at end of text
  End If
End Sub

Private Sub mnuExit_Click()
  MSComm1.PortOpen = False                         ' close com port
  End                                              ' done!
End Sub

Private Sub Form_Unload(Cancel As Integer)
  MSComm1.PortOpen = False                         ' close com port
  End                                              ' done!
End Sub

Private Sub mnuClear_Click()
  Text1.Text = ""                                  ' clear TX textbox
  Text1.SelStart = Len(Text1.Text)                 ' put cursor at end of text
  Text2.Text = ""                                  ' clear RX textbox
  Text2.SelStart = Len(Text2.Text)                 ' put cursor at end of text
End Sub

Private Sub mnuDups_Click()
  If DupFltr = 0 Then                              ' if show RX dups active
    DupFltr = 1                                    ' toggle to inactive
    mnuDups.Checked = False                        ' and uncheck Show RX Dups
  Else
    DupFltr = 0                                    ' else toggle active
    mnuDups.Checked = True                         ' and check Show RX Dups
  End If
End Sub

Private Sub mnuShw_Click()
  If ShwACK = 1 Then                               ' if show ACK/NAK active
    ShwACK = 0                                     ' toggle to inactive
    mnuShw.Checked = False                         ' and uncheck Show ACK/NAK
  Else
    ShwACK = 1                                     ' else toggle active
    mnuShw.Checked = True                          ' and check Show ACK/NAK
  End If
End Sub

Private Sub mnuAutoSnd_Click()
  ASFlag = ASFlag Xor 1                            ' toggle AutoSend flag
  If ASFlag = 0 Then                               ' if flag reset
    Call ReSetTX                                   ' reset TX
    Text1.ForeColor = QBColor(15)                  ' make letters white
    mnuAutoSnd.Checked = False                     ' uncheck AutoSend
  End If
  If ASFlag = 1 Then                               ' if flag active
    PCnt = 0                                       ' clear TX tries counter
    NAFlag = 0                                     ' clear next AutoSend flag
    Text1.ForeColor = QBColor(10)                  ' make letters green
    mnuAutoSnd.Checked = True                      ' check AutoSend
  End If
End Sub

Private Sub mnuFN1_Click()
  FNode = 1                                        ' from Node = 1
  Call BldHdr                                      ' build new packet header
  Call RstFrmChk                                   ' reset all From check marks
  mnuFN1.Checked = True                            ' check Node 1
End Sub

Private Sub mnuFN2_Click()
  FNode = 2                                        ' from Node = 2
  Call BldHdr                                      ' build new packet header
  Call RstFrmChk                                   ' reset all From check marks
  mnuFN2.Checked = True                            ' check Node 2
End Sub
```

```
Private Sub mnuFN3_Click()
  FNode = 3                                                   ' from Node = 3
  Call BldHdr                                                 ' build new packet header
  Call RstFrmChk                                              ' reset all From check marks
  mnuFN3.Checked = True                                       ' check Node 3
End Sub

Private Sub mnuFN4_Click()
  FNode = 4                                                   ' from Node = 4
  Call BldHdr                                                 ' build new packet header
  Call RstFrmChk                                              ' reset all From check marks
  mnuFN4.Checked = True                                       ' check Node 4
End Sub

Public Sub RstFrmChk()
  mnuFN1.Checked = False                                      ' uncheck From Node 1
  mnuFN2.Checked = False                                      ' uncheck From Node 2
  mnuFN3.Checked = False                                      ' uncheck From Node 3
  mnuFN4.Checked = False                                      ' uncheck From Node 4
End Sub

Private Sub mnuTN1_Click()
  TNode = 1                                                   ' To Node = 1
  Call BldHdr                                                 ' build new packet header
  Call RstToChk                                               ' reset all To check marks
  mnuTN1.Checked = True                                       ' check Node 1
End Sub

Private Sub mnuTN2_Click()
  TNode = 2                                                   ' To Node = 2
  Call BldHdr                                                 ' build new packet header
  Call RstToChk                                               ' reset all To check marks
  mnuTN2.Checked = True                                       ' check Node 2
End Sub

Private Sub mnuTN3_Click()
  TNode = 3                                                   ' To Node = 3
  Call BldHdr                                                 ' build new packet header
  Call RstToChk                                               ' reset all To check marks
  mnuTN3.Checked = True                                       ' check Node 3
End Sub

Private Sub mnuTN4_Click()
  TNode = 4                                                   ' To Node = 4
  Call BldHdr                                                 ' build new packet header
  Call RstToChk                                               ' reset all To check marks
  mnuTN4.Checked = True                                       ' check Node 4
End Sub

Public Sub RstToChk()
  mnuTN1.Checked = False                                      ' uncheck To Node 1
  mnuTN2.Checked = False                                      ' uncheck To Node 2
  mnuTN3.Checked = False                                      ' uncheck To Node 3
  mnuTN4.Checked = False                                      ' uncheck To Node 4
End Sub

Public Sub BldHdr()
  TF = (16 * TNode) + FNode                                   ' TF is numeric To/From node address
  PktHdr$ = Chr$(TF)                                          ' Chr$(TF) is To/From packet header
End Sub
```

5.3 DK110K.ASM

```
;  DK110K.ASM 2002.08.01 @ 20:00 CDT
;  See RFM Virtual Wire(r) Development Kit Warranty & License for terms of use
;  Experimental software - NO representation is
;  made that this software is suitable for any purpose
;  Copyright(c) 2000 - 2002, RF Monolithics, Inc.
;  AT89C2051 assembler source code file (TASM 3.01 assembler)
;  Low signal-to-noise protocol for RFM ASH transceiver
;  Integrate & dump PLL (I&D) - 62.40 us tick

          .NOLIST
          #INCLUDE "8051.H"        ;  tasm 8051 include file
          .LIST
```

```
; constants:

ITMOD     .EQU     022H          ; set timers 0 and 1 to mode 2
ITICK     .EQU     141           ; set timer T0 for 62.40 us tick
ISMOD     .EQU     080H          ; SMOD = 1 in PCON

IBAUD     .EQU     0FAH          ; 19.2 kbps @ 22.1184 MHz, SMOD = 1
ISCON     .EQU     050H          ; UART mode 1

RMPT      .EQU     159           ; PLL ramp top value (modulo 0 to 159)
RMPW      .EQU     159           ; PLL ramp reset (wrap) value
RMPS      .EQU     80            ; PLL ramp switch value
RMPI      .EQU     20            ; PLL ramp increment value
RMPA      .EQU     29            ; PLL 5% advance increment value (20 + 9)
RMPR      .EQU     11            ; PLL 5% retard increment value (20 - 9)

TXMB      .EQU     044H          ; TX message buffer start address
RXMB      .EQU     062H          ; RX message buffer start address
FEND      .EQU     0C0H          ; FEND framing character (192)
SOPL      .EQU     08AH          ; SOP low correlator pattern
SOPH      .EQU     0B3H          ; SOP high correlator pattern
TXRO      .EQU     020H          ; TX retry timer count

FCSS      .EQU     0FFH          ; FCS seed
FCSH      .EQU     084H          ; FCS high XOR mask
FCSL      .EQU     08H           ; FCS low XOR mask
FCVH      .EQU     0F0H          ; FCS valid high byte pattern
FCVL      .EQU     0B8H          ; FCS valid low byte pattern

; stack:  08H - 021H (26 bytes)

; bit labels:

WBFLG     .EQU     010H          ; warm boot flag (future use)
PLLON     .EQU     011H          ; RX PLL control flag
RXISM     .EQU     012H          ; RX inverted input sample
RXSMP     .EQU     013H          ; RX input sample
LRXSM     .EQU     014H          ; last RX input sample
RXBIT     .EQU     015H          ; RX input bit
RXBFLG    .EQU     016H          ; RX input bit flag
SOPFLG    .EQU     017H          ; SOP detect flag
RXSFLG    .EQU     018H          ; RX symbol flag
RM        .EQU     019H          ; RX FCS message bit
OKFLG     .EQU     01AH          ; RX FCS OK flag

SIFLG     .EQU     01BH          ; serial in active flag
TSFLG     .EQU     01CH          ; output TX sample flag
TXSMP     .EQU     01DH          ; TX output sample
TXBIT     .EQU     01EH          ; TX message bit
TM        .EQU     01FH          ; TX FCS message bit
TXFLG     .EQU     020H          ; TX active flag
TMFLG     .EQU     021H          ; TX message flag
TOFLG     .EQU     022H          ; get message time out flag
AMFLG     .EQU     023H          ; AutoSend message flag
ASFLG     .EQU     024H          ; AutoSend active flag

SFLG0     .EQU     025H          ; spare flag 0
SFLG1     .EQU     026H          ; spare flag 1
SFLG2     .EQU     027H          ; spare flag 2
SFLG3     .EQU     028H          ; spare flag 3
SFLG4     .EQU     029H          ; spare flag 4
SFLG5     .EQU     02AH          ; spare flag 5
SFLG6     .EQU     02BH          ; spare flag 6
SFLG7     .EQU     02CH          ; spare flag 7
SFLG8     .EQU     02DH          ; spare flag 8
SFLG9     .EQU     02EH          ; spare flag 9
SFLGA     .EQU     02FH          ; spare flag A

; register usage:

; R0                               RX data pointer
; R1                               TX data pointer
; R2                               PLL ramp buffer
; R3                               RX FCS buffer A
; R4                               not used
; R5                               TX FCS buffer A
; R6                               TX FCS buffer B
; R7                               RX FCS buffer B
```

```
;  byte labels:

BOOT      .EQU      022H          ;  1st byte of flags

RXID      .EQU      026H          ;  RX integrate & dump buffer
RXBL      .EQU      027H          ;  RX low buffer, SOP correlator etc.
RXBH      .EQU      028H          ;  RX high buffer, SOP correlator etc.
RXBB      .EQU      029H          ;  RX symbol decode byte buffer
RMDC      .EQU      02AH          ;  RX symbol decode loop counter
RMBIC     .EQU      02BH          ;  RX symbol decode index pointer
RMBYC     .EQU      02CH          ;  RX message byte counter
RMFCS     .EQU      02DH          ;  RX FCS byte buffer
RMSBC     .EQU      02EH          ;  RX symbol bit counter
RMLPC     .EQU      02FH          ;  RX message loop counter
RMFCC     .EQU      030H          ;  RX message FCS counter, etc.

TMFCC     .EQU      031H          ;  TX timer & loop counter
TXSMC     .EQU      032H          ;  TX output sample counter
TMBIC     .EQU      033H          ;  TX message bit counter
TMBYT     .EQU      034H          ;  TX message byte buffer
TMBYC     .EQU      035H          ;  TX message byte counter
TXSL      .EQU      036H          ;  TX message symbol low buffer
TXSH      .EQU      037H          ;  TX message symbol high buffer
TMFCS     .EQU      038H          ;  TX FCS byte buffer
TXTL      .EQU      039H          ;  TX timer low byte
TXTH      .EQU      03AH          ;  TX timer high byte


BUF0      .EQU      03BH          ;  spare buffer 0
BUF1      .EQU      03CH          ;  spare buffer 1
BUF2      .EQU      03DH          ;  spare buffer 2
BUF3      .EQU      03EH          ;  spare buffer 3
BUF4      .EQU      03FH          ;  spare buffer 4
BUF5      .EQU      040H          ;  spare buffer 5
BUF6      .EQU      041H          ;  spare buffer 6
BUF7      .EQU      042H          ;  spare buffer 7
BUF8      .EQU      043H          ;  spare buffer 8

;  I/O pins:

MAX       .EQU      P1.6          ;  Maxim 218 power (on = 1)

RXPIN     .EQU      P3.2          ;  RX input pin (inverted data)
TXPIN     .EQU      P3.3          ;  TX output pin (on = 1)
PTT       .EQU      P1.7          ;  transmit enable (TX = 0)

PCRCV     .EQU      P3.7          ;  PC (host) input LED (on = 0)
RFRCV     .EQU      P3.5          ;  RX FCS OK LED (on = 0)
RXI       .EQU      P3.4          ;  RX activity LED (on = 0)

ID0       .EQU      P1.2          ;  jumper input bit 0 (dot end)
ID1       .EQU      P1.3          ;  jumper input bit 1
ID2       .EQU      P1.4          ;  jumper input bit 2
ID3       .EQU      P1.5          ;  jumper input bit 3

;  start of code:

          .ORG      00H           ;  hardware reset
          SETB      WBFLG         ;  set warm boot flag
reset:    AJMP      start         ;  jump to start

          .ORG      0BH           ;  timer 0 interrupt vector
t_isr:    ACALL     tick          ;  sampling tick subroutine
          RETI                    ;  interrupt done

          .ORG      023H          ;  serial interrupt vector
s_isr:    ACALL     srio          ;  serial I/O subroutine
          CLR       TI            ;  clear TI (byte sent) flag
          CLR       RI            ;  clear RI (byte received) flag
          RETI                    ;  interrupt done

          .ORG      040H          ;  above interrupt code space
start:    ACALL     setup         ;  initialization code

main:     JNB       AMFLG,mn0     ;  skip if AutoSend idle
          CLR       PCRCV         ;  else turn PCRCV LED on
          ACALL     do_as         ;  do AutoSend
          SETB      PCRCV         ;  turn PCRCV LED off
mn0:      ACALL     rxsop         ;  do RX SOP detect
          JNB       SOPFLG,main   ;  if not SOP loop to main
          ACALL     do_rx         ;  else do RX message
```

```
mn_d:     AJMP      main           ;  and loop to main

do_rx:    CLR       ES             ;  deactivate serial interrupts
          ACALL     rxmsg          ;  decode RX message
          CLR       PLLON          ;  idle RX PLL
          ACALL     rxfcs          ;  test RX message FCS
          JNB       OKFLG,rx0      ;  reset if FCS error
          ACALL     rxsnd          ;  else send RX message to host
rx0:      ACALL     rxrst          ;  reset for next RX message
          SETB      PLLON          ;  enable RX PLL
          CLR       TI             ;  clear TI flag
          CLR       RI             ;  clear RI flag
          SETB      ES             ;  activate serial interrupts
rx_d:     RET                      ;  RX done

tick:     PUSH      PSW            ;  push status
          PUSH      ACC            ;  push accumulator
          MOV       C,RXPIN        ;  read RX input pin
          MOV       RXISM,C        ;  store as inverted RX sample
          JNB       TSFLG,tic0     ;  skip if TX sample out idle
          MOV       A,TXSMC        ;  else get sample count
          JZ        tic0           ;  skip if 0
          MOV       C,TXBIT        ;  else load TX bit
          MOV       TXPIN,C        ;  into TX output pin
          DEC       TXSMC          ;  decrement sample count
tic0:     JNB       PLLON,tic1     ;  skip if PLL idle
          ACALL     pll            ;  else run RX PLL
tic1:     JNB       TOFLG,tic2     ;  skip if get message timeout idle
          INC       TMFCC          ;  else bump timeout counter
          MOV       A,TMFCC        ;  get counter
          CJNE      A,#50,tic2     ;  skip if counter <> 50 (5.2 ms)
          CLR       TOFLG          ;  else reset time out flag
          MOV       TMFCC,#0       ;  reset counter
tic2:     JNB       ASFLG,tick_d   ;  done if AutoSend idle
          INC       TXTL           ;  else bump TX timer low
          MOV       A,TXTL         ;  load TX timer low
          JNZ       tick_d         ;  done if no rollover
          INC       TXTH           ;  else bump TX timer high
          MOV       A,TXTH         ;  load timer
          CLR       C              ;  clear borrow
          SUBB      A,#TXRO        ;  subtract TX retry count
          JNZ       tick_d         ;  if <> 0 done for now
          SETB      AMFLG          ;  else set AM message flag
          CLR       A              ;  clear A
          MOV       TXTL,A         ;  clear TX timer low
          MOV       TXTH,A         ;  clear TX timer high
tick_d:   POP       ACC            ;  pop accumulator
          POP       PSW            ;  pop status
          RET                      ;  tick done

pll:      MOV       C,RXSMP        ;  load RX sample
          MOV       LRXSM,C        ;  into last RX sample
          MOV       C,RXISM        ;  get inverted RX sample
          CPL       C              ;  invert
          MOV       RXSMP,C        ;  and store
          JNC       pll0           ;  if <> 1 jump to pll0
          INC       RXID           ;  else increment I&D
pll0:     JNB       LRXSM,pll1     ;  if last sample 1
          CPL       C              ;  invert current sample
pll1:     JNC       pll4           ;  if no edge jump to pll4
          MOV       A,R2           ;  else get PLL value
          CLR       C              ;  clear borrow
          SUBB      A,#RMPS        ;  subtract ramp switch value
          JC        pll3           ;  if < 0 then retard PLL
pll2:     MOV       A,R2           ;  else get PLL value
          ADD       A,#RMPA        ;  add (RMPI + 5%)
          MOV       R2,A           ;  store PLL value
          AJMP      pll5           ;  and jump to pll5
pll3:     MOV       A,R2           ;  get PLL value
          ADD       A,#RMPR        ;  add (RMPI - 5%)
          MOV       R2,A           ;  store PLL value
          AJMP      pll5           ;  and jump to pll5
pll4:     MOV       A,R2           ;  get PLL value
          ADD       A,#RMPI        ;  add ramp increment
          MOV       R2,A           ;  store new PLL value
pll5:     CLR       C              ;  clear borrow
          MOV       A,R2           ;  get PLL ramp value
          SUBB      A,#RMPT        ;  subtract ramp top
          JC        pllD           ;  if < 0 don't wrap
pll6:     MOV       A,R2           ;  else get PLL value
          CLR       C              ;  clear borrow
```

```
         SUBB      A,#RMPW         ;  subtract reset value
         MOV       R2,A            ;  and store result
         CLR       C               ;  clear borrow
         MOV       A,RXID          ;  get I&D buffer
         SUBB      A,#5            ;  subtract 5
         JNC       pll7            ;  if I&D count => 5 jump to pll7
         CLR       RXBIT           ;  else RX bit = 0 for I&D count < 5
         SETB      RXBFLG          ;  set new RX bit flag
         MOV       RXID,#0         ;  clear the I&D buffer
         AJMP      pll8            ;  and jump to pll8
pll7:    SETB      RXBIT           ;  RX bit = 1 for I&D count => 5
         SETB      RXBFLG          ;  set new RX bit flag
         MOV       RXID,#0         ;  clear the I&D buffer
pll8:    JB        SOPFLG,pllA     ;  skip after SOP detect
         MOV       A,RXBH          ;  else get RXBH
         CLR       C               ;  clear carry
         RRC       A               ;  rotate right
         JNB       RXBIT,pll9      ;  if bit = 0 jump to pll9
         SETB      ACC.7           ;  else set 7th bit
pll9:    MOV       RXBH,A          ;  store RXBH
         MOV       A,RXBL          ;  get RXBL
         RRC       A               ;  shift and pull in carry
         MOV       RXBL,A          ;  store RXBL
         AJMP      pll_d           ;  done for now
pllA:    MOV       A,RXBL          ;  get RXBL
         CLR       C               ;  clear carry
         RRC       A               ;  shift right
         JNB       RXBIT,pllB      ;  if bit = 0 jump to pllB
         SETB      ACC.5           ;  else set 5th bit
pllB:    MOV       RXBL,A          ;  store RXBL
         INC       RMSBC           ;  bump bit counter
         MOV       A,RMSBC         ;  get counter
         CJNE      A,#6,pllC       ;  if <> 6 jump to pllC
         MOV       RXBB,RXBL       ;  else get symbol
         MOV       RMSBC,#0        ;  reset counter
         SETB      RXSFLG          ;  set symbol flag
pllC:    AJMP      pll_d           ;  done
pllD:    CLR       RXBFLG          ;  clear RXBFLG
pll_d:   RET                       ;  PLL done

rxsop:   JNB       RXBFLG,sop_d    ;  done if no RX bit flag
         CLR       RXBFLG          ;  else clear RX bit flag
         MOV       A,RXBL          ;  get low RX buffer
         CJNE      A,#SOPL,sop_d   ;  done if <> SOPL
         MOV       A,RXBH          ;  else get high RX buffer
         CJNE      A,#SOPH,sop_d   ;  done if <> SOPH
         CLR       A               ;  else clear A
         MOV       RXBL,A          ;  clear RX low buffer
         MOV       RXBH,A          ;  clear RX high buffer
         MOV       RMSBC,A         ;  clear RX symbol bit counter
         CLR       RXSFLG          ;  clear RX symbol flag
         SETB      SOPFLG          ;  set SOP detected flag
         CLR       RXI             ;  RXI LED on
sop_d:   RET                       ;  SOP detect done

rxmsg:   JNB       RXSFLG,rxmsg    ;  wait for RX symbol flag
         CLR       RXSFLG          ;  clear RX symbol flag
rxm1:    MOV       DPTR,#smbl      ;  point to RX symbol decode table
         MOV       RMDC,#16        ;  16 symbol decode table entries
         MOV       RMBIC,#0        ;  index into symbol table
rxm2:    MOV       A,RMBIC         ;  load index into A
         MOVC      A,@A+DPTR       ;  get table entry
         XRL       A,RXBB          ;  XOR to compare with RXBB
         JZ        rxm3            ;  exit loop with decoded nibble
         INC       RMBIC           ;  else bump index
         DJNZ      RMDC,rxm2       ;  and try to decode again
rxm3:    MOV       A,RMBIC         ;  get decoded nibble
         SWAP      A               ;  swap to high nibble
         MOV       RXBH,A          ;  into RXBH (low nibble is high)
rxm4:    JNB       RXSFLG,rxm4     ;  wait for symbol flag
         CLR       RXSFLG          ;  clear flag
rxm5:    MOV       DPTR,#smbl      ;  point to symbol decode table
         MOV       RMDC,#16        ;  16 symbol decode table entries
         MOV       RMBIC,#0        ;  reset symbol table index
rxm6:    MOV       A,RMBIC         ;  load index into A
         MOVC      A,@A+DPTR       ;  get table entry
         XRL       A,RXBB          ;  XOR to compare with RXBB
         JZ        rxm7            ;  exit loop with decoded nibble
         INC       RMBIC           ;  else bump index
         DJNZ      RMDC,rxm6       ;  and try to decode again
rxm7:    MOV       A,RMBIC         ;  get decoded nibble
```

```
          ORL       A,RXBH          ;  add RXBH low
          SWAP      A               ;  nibbles now in right order
          MOV       RXBH,A          ;  store in RXBH
          MOV       @R0,RXBH        ;  and store in RX message buffer
          CJNE      R0,#RXMB,rxm8   ;  skip if not 1st message byte
          MOV       A,RXBH          ;  else get 1st byte
          ANL       A,#63           ;  mask upper 2 bits
          MOV       RMBYC,A         ;  load message byte counter
          MOV       RMFCC,A         ;  and RX message loop counter
          CLR       C               ;  clear borrow
          SUBB      A,#28           ;  compare # bytes to 28
          JC        rxm8            ;  skip if < 28
          MOV       RMBYC,#4        ;  else force byte counter to 4
          MOV       RMFCC,#4        ;  and force loop counter to 4
rxm8:     INC       R0              ;  bump pointer
          DJNZ      RMFCC,rxmsg     ;  if <> 0 get another byte
          MOV       R0,#RXMB        ;  reset RX message pointer
          SETB      RXI             ;  turn LED off
rxm_d:    RET                       ;  RX message done

rxfcs:    MOV       RMFCC,RMBYC     ;  move byte count to loop counter
rxf0:     MOV       RMFCS,@R0       ;  get next message byte
          INC       R0              ;  bump pointer
          ACALL     b_rfcs          ;  build FCS
          DJNZ      RMFCC,rxf0      ;  loop for next byte
          ACALL     a_rfcs          ;  test FCS
rxf_d:    RET                       ;  RX FCS done
rxsnd:    CLR       PCRCV           ;  turn PC LED on
          DEC       RMBYC           ;  don't send
          DEC       RMBYC           ;  the 2 FCS bytes
          MOV       R0,#RXMB        ;  reset RX message pointer
          MOV       @R0,#FEND       ;  replace # bytes with 1st FEND
          CLR       TI              ;  clear TI flag
rxs1:     MOV       SBUF,@R0        ;  send byte
rxs2:     JNB       TI,rxs2         ;  wait until byte sent
          CLR       TI              ;  clear TI flag
          INC       R0              ;  bump pointer
          DJNZ      RMBYC,rxs1      ;  loop to echo message
          MOV       SBUF,#FEND      ;  add 2nd FEND
rxs3:     JNB       TI,rxs3         ;  wait until byte sent
          CLR       TI              ;  clear TI flag
          SETB      RFRCV           ;  turn FCS LED off
          SETB      PCRCV           ;  turn PC LED off
rxs_d:    RET                       ;  send RX message done

rxrst:    CLR       A               ;  clear A
          MOV       RXBH,A          ;  clear buffer
          MOV       RXBL,A          ;  clear buffer
          MOV       RXBB,A          ;  clear buffer
          MOV       RMBYC,A         ;  clear rx byte count
          MOV       RMFCC,A         ;  clear loop counter
          MOV       R0,#RXMB        ;  point R0 to message start
          CLR       OKFLG           ;  clear packet OK flag
          CLR       SOPFLG          ;  enable SOP test
          SETB      RXI             ;  RXI LED off
rxr_d:    RET                       ;  RX reset done

b_rfcs:   MOV       RMLPC,#8        ;  load loop count of 8
brf0:     CLR       C               ;  clear carry bit
          MOV       A,RMFCS         ;  load RX message byte
          RRC       A               ;  shift lsb into carry
          MOV       RMFCS,A         ;  store shifted message byte
          MOV       RM,C            ;  load RM with lsb
          CLR       C               ;  clear carry bit
          MOV       A,R3            ;  load high FCS byte
          RRC       A               ;  shift right
          MOV       R3,A            ;  store shifted high FCS
          MOV       A,R7            ;  load low FCS byte
          RRC       A               ;  shift and pull in bit for FCS high
          MOV       R7,A            ;  store shifted low FCS
          JNB       RM,brf1         ;  if lsb of low FCS = 0, jump to brf1
          CPL       C               ;  else complement carry bit
brf1:     JNC       brf2            ;  if RM XOR (low FCS lsb) = 0 jump to brf2
          MOV       A,R3            ;  else load high FCS
          XRL       A,#FCSH         ;  and XOR with high FCS poly
          MOV       R3,A            ;  store high FCS
          MOV       A,R7            ;  load low FCS
          XRL       A,#FCSL         ;  XOR with low FCS poly
          MOV       R7,A            ;  store low FCS
brf2:     DJNZ      RMLPC,brf0      ;  loop through bits in message byte
brfcs_d:  RET                       ;  done this pass
```

```
a_rfcs:    MOV      A,R3          ;  load FCS high
           XRL      A,#FCVH       ;  compare with 0F0H
           JNZ      arf0          ;  if <> 0 jump to arf0
           MOV      A,R7          ;  load FCS low
           XRL      A,#FCVL       ;  else compare with 0B8H
           JNZ      arf0          ;  if <> 0 jump to arf0
           CLR      RFRCV         ;  else turn FCS LED on
           SETB     OKFLG         ;  set FCS OK flag
arf0:      MOV      R3,#FCSS      ;  reseed FCS high
           MOV      R7,#FCSS      ;  reseed FCS low
arfcs_d:   RET                    ;  RX FCS done

srio:      PUSH     PSW           ;  save
           PUSH     ACC           ;  environment
           JNB      TI,sr_0       ;  skip if TI flag clear
           CLR      TI            ;  else clear TI flag
sr_0:      JNB      RI,sr_1       ;  skip if RI flag clear
           CLR      RI            ;  else clear RI flag
           JNB      SIFLG,sr_1    ;  skip if serial in flag reset
           CLR      PCRCV         ;  else turn PC LED on
           ACALL    do_tx         ;  get & TX host message
           SETB     PCRCV         ;  turn PC LED off
sr_1:      POP      ACC           ;  restore
           POP      PSW           ;  environment
           RET                    ;  serial in done

do_as:     CLR      PLLON         ;  idle RX PLL
           ACALL    hello2        ;  get AutoSend message
           ACALL    txfcs         ;  build and add FCS
           ACALL    txpre         ;  send TX preamble
           ACALL    txmsg         ;  send TX message
           ACALL    txrst         ;  reset TX
           SETB     PLLON         ;  enable RX PLL
           RET                    ;  TX message done

do_tx:     ACALL    txget         ;  get TX message from host
           JNB      TXFLG,do0     ;  skip if TXFLG not set
           CLR      PLLON         ;  else idle RX PLL
           ACALL    txfcs         ;  build and add FCS
           ACALL    txpre         ;  send TX preamble
           ACALL    txmsg         ;  send TX message
do0:       ACALL    txrst         ;  reset TX
           SETB     PLLON         ;  enable RX PLL
           RET                    ;  TX message done

txget:     MOV      A,SBUF        ;  get byte
           MOV      TMBYT,A       ;  copy to TMBYT
           XRL      A,#FEND       ;  compare to FEND
           JZ       txg0          ;  if FEND jump to txg0
           AJMP     txg_d         ;  else done
txg0:      MOV      @R1,TMBYT     ;  store 1st FEND
           INC      TMBYC         ;  bump TX byte counter
txg1:      MOV      TMFCC,#0      ;  reset timeout counter
           SETB     TOFLG         ;  set timeout flag
           CLR      RI            ;  reset flag
txg2:      JNB      TOFLG,txg3    ;  if TOFLG reset jump to txg3
           JNB      RI,txg2       ;  else loop until next byte
           CLR      RI            ;  reset RI flag
           CLR      TOFLG         ;  reset TOFLG
           AJMP     txg4          ;  and jump to txg4
txg3:      MOV      TMBYC,#2      ;  look like null message
           AJMP     txg6          ;  and jump to txg6
txg4:      MOV      A,SBUF        ;  get byte
           MOV      TMBYT,A       ;  copy to TMBYT
           INC      TMBYC         ;  bump byte counter
           INC      R1            ;  bump pointer R1
           MOV      @R1,TMBYT     ;  store byte
           MOV      A,TMBYC       ;  load counter
           CLR      C             ;  clear carry
           SUBB     A,#26         ;  test for 26 bytes
           JZ       txg5          ;  if 26 handle overflow at txg5
           MOV      A,TMBYT       ;  else load byte
           CJNE     A,#FEND,txg1  ;  if <> FEND loop to txg1
           AJMP     txg6          ;  else jump to txg6 on 2nd FEND
txg5:      MOV      @R1,#FEND     ;  force 2nd FEND
txg6:      MOV      R1,#TXMB      ;  reset TX message pointer
           MOV      A,TMBYC       ;  get byte count
           CJNE     A,#2,txg7     ;  if <> 2 jump to txg7
           MOV      TMBYC,#0      ;  else reset byte counter
           AJMP     txg_d         ;  jump to txg_d
```

```
txg7:     CLR      SIFLG          ; idle serial in
          SETB     TXFLG          ; set TX flag
txg_d:    RET                     ; get TX message done

txfcs:    INC      TMBYC          ; # bytes including FCS
          MOV      @R1,TMBYC      ; replace 1st FEND with # bytes
          MOV      TMFCC,TMBYC    ; move byte count to loop counter
          DEC      TMFCC          ; loop count is 2 less
          DEC      TMFCC          ; than # bytes including FCS
txf0:     MOV      TMFCS,@R1      ; get next message byte
          INC      R1             ; bump pointer
          ACALL    b_tfcs         ; build FCS
          DJNZ     TMFCC,txf0     ; loop for next byte
          ACALL    a_tfcs         ; add FCS
          MOV      R1,#TXMB       ; reset TX message pointer
          SETB     TMFLG          ; set TX message flag
txf_d:    RET                     ; TX FCS done

txpre:    CLR      PTT            ; turn PTT on
          MOV      B,#200         ; load PTT delay count
txp0:     DJNZ     B,txp0         ; loop to delay
txp1:     MOV      DPTR,#tstrt    ; point to TX start table
          MOV      B,#0           ; clear B
          MOV      A,B            ; B holds table offset
          MOVC     A,@A+DPTR      ; load table entry
          MOV      TMBYT,A        ; into TMBYT
          MOV      TMBIC,#4       ; load bit count
          MOV      TXSMC,#0       ; clear sample count
          SETB     TSFLG          ; turn TX sample out on
txp2:     MOV      A,TXSMC        ; get sample count
          JNZ      txp2           ; loop until sample count 0
          MOV      A,TMBIC        ; get bit count
          JNZ      txp3           ; if <> 0 jump to txp3
          MOV      A,B            ; else get current offset (0 to 11)
          CLR      C              ; clear carry
          SUBB     A,#11          ; subtract ending offset
          JZ       txp_d          ; if 0  done
          INC      B              ; else bump byte count
          MOV      A,B            ; get count/offset
          MOVC     A,@A+DPTR      ; load table entry
          MOV      TMBYT,A        ; into TMBYT
          MOV      TMBIC,#4       ; reload bit count
txp3:     MOV      A,TMBYT        ; get TX message byte
          CLR      C              ; clear carry
          RRC      A              ; shift right into carry
          MOV      TXBIT,C        ; load next bit
          MOV      TMBYT,A        ; store shifted message byte
          DEC      TMBIC          ; decrement bit count
          MOV      TXSMC,#8       ; reload sample count
          AJMP     txp2           ; loop again
txp_d:    RET                     ; TX preamble done

txmsg:    MOV      B,#1           ; count 1st byte sent
          MOV      R1,#TXMB       ; reset TX message pointer
          MOV      A,@R1          ; get 1st TX message byte
          MOV      TMBYT,A        ; into TMBYT
          MOV      DPTR,#smbl     ; point to symbol table
          ANL      A,#0FH         ; clean offset
          MOVC     A,@A+DPTR      ; get 6-bit symbol
          MOV      TXSL,A         ; move to TXSL
          MOV      A,TMBYT        ; get TMBYT
          SWAP     A              ; swap nibbles
          ANL      A,#0FH         ; clean offset
          MOVC     A,@A+DPTR      ; get 6-bit symbol
          MOV      TXSH,A         ; move to TXSH
          MOV      TMBIC,#12      ; set bit count to 12
          MOV      TXSMC,#0       ; clear sample count
txm0:     MOV      A,TXSMC        ; get sample count
          JNZ      txm0           ; loop until sample count 0
          MOV      A,TMBIC        ; get bit count
          CLR      C              ; clear carry
          SUBB     A,#7           ; subtract 7
          JNC      txm1           ; if => 7 jump to txm1
          MOV      A,TMBIC        ; else get bit count
          JNZ      txm2           ; if > 0 jump to txm2
          MOV      A,B            ; else get current byte number
          CLR      C              ; clear carry
          SUBB     A,TMBYC        ; subtract TX message byte count
          JZ       txm3           ; if 0 done
          INC      R1             ; else bump byte pointer
          INC      B              ; and bump byte counter
```

```
          MOV       A,@R1          ;  get next byte
          MOV       TMBYT,A        ;  into TMBYT
          MOV       DPTR,#smbl     ;  point to symbol table
          ANL       A,#0FH         ;  offset
          MOVC      A,@A+DPTR      ;  get 6-bit symbol
          MOV       TXSL,A         ;  move to TXSL
          MOV       A,TMBYT        ;  get TMBYT
          SWAP      A              ;  swap nibbles
          MOV       DPTR,#smbl     ;  point to symbol table
          ANL       A,#0FH         ;  clean offset
          MOVC      A,@A+DPTR      ;  get 6-bit symbol
          MOV       TXSH,A         ;  move to TXSH
          MOV       TMBIC,#12      ;  set bit count to 12
txm1:     MOV       A,TXSL         ;  get low TX symbol
          CLR       C              ;  clear carry
          RRC       A              ;  shift right into carry
          MOV       TXBIT,C        ;  load next bit
          MOV       TXSL,A         ;  store shifted message byte
          DEC       TMBIC          ;  decrement bit count
          MOV       TXSMC,#8       ;  reload sample count
          AJMP      txm0           ;  loop again
txm2:     MOV       A,TXSH         ;  get high TX symbol
          CLR       C              ;  clear carry
          RRC       A              ;  shift right into carry
          MOV       TXBIT,C        ;  load next bit
          MOV       TXSH,A         ;  store shifted message byte
          DEC       TMBIC          ;  decrement bit count
          MOV       TXSMC,#8       ;  reload sample count
          AJMP      txm0           ;  loop again
txm3:     CLR       TSFLG          ;  clear TX sample out flag
          CLR       TXPIN          ;  clear TX out pin
          SETB      PTT            ;  turn PTT off
txm_d:    RET                      ;  TX message done

txrst:    CLR       TMFLG          ;  clear TX message flag
          CLR       AMFLG          ;  clear AutoSend message flag
          CLR       A              ;  reset for next TX
          MOV       TMBYT,A        ;  clear TX message byte
          MOV       TMFCC,A        ;  clear TX FCS count
          MOV       TXSMC,A        ;  clear TX out count
          MOV       TXSL,A         ;  clear TX symbol low
          MOV       TXSH,A         ;  clear TX symbol high
          MOV       R1,#TXMB       ;  point R1 to message start
          JB        ASFLG,txr_d    ;  skip if in AutoSend
          MOV       TMBYC,A        ;  clear TX message byte count
          CLR       TXFLG          ;  clear TX flag
          SETB      SIFLG          ;  set serial in flag active
txr_d:    RET                      ;  TX reset done

b_tfcs:   MOV       B,#8           ;  load loop count of 8
btf0:     CLR       C              ;  clear carry bit
          MOV       A,TMFCS        ;  load TX message byte
          RRC       A              ;  shift lsb into carry
          MOV       TMFCS,A        ;  store shifted message byte
          MOV       TM,C           ;  load TM with lsb
          CLR       C              ;  clear carry bit
          MOV       A,R5           ;  load high FCS byte
          RRC       A              ;  shift right
          MOV       R5,A           ;  store shifted high FCS
          MOV       A,R6           ;  load low FCS byte
          RRC       A              ;  shift and pull in bit for FCS high
          MOV       R6,A           ;  store shifted low FCS
          JNB       TM,btf1        ;  if lsb of low FCS = 0, jump to btf1
          CPL       C              ;  else complement carry bit
btf1:     JNC       btf2           ;  if TM XOR (low FCS lsb) = 0 jump to btf2
          MOV       A,R5           ;  else load high FCS
          XRL       A,#FCSH        ;  and XOR with high FCS poly
          MOV       R5,A           ;  store high FCS
          MOV       A,R6           ;  load low FCS
          XRL       A,#FCSL        ;  XOR with low FCS poly
          MOV       R6,A           ;  store low FCS
btf2:     DJNZ      B,btf0         ;  loop through bits in message byte
btfcs_d:  RET                      ;  done this pass

a_tfcs:   MOV       A,R6           ;  load FCS (high/low switch)
          CPL       A              ;  1's complement
          MOV       @R1,A          ;  store at end of TX message
          INC       R1             ;  increment TX message byte pointer
          MOV       A,R5           ;  load FCS (high/low switch)
          CPL       A              ;  1's complement
          MOV       @R1,A          ;  store at end of TX message
```

```
            MOV     R5,#FCSS        ;  reseed FCS high
            MOV     R6,#FCSS        ;  reseed FCS low
atfcs_d:    RET                     ;  add TX FCS done

setup:      CLR     EA              ;  disable interrupts
            SETB    PTT             ;  turn PTT off
            CLR     TXPIN           ;  turn TX modulation off

tick_su:    MOV     TMOD,#ITMOD     ;  set timers T0 and T1 to mode 2
            CLR     TR0             ;  stop timer T0
            CLR     TF0             ;  clear T0 overflow
            MOV     TH0,#ITICK      ;  load count for 62.40 us tick
            MOV     TL0,#ITICK      ;  load count for 62.40 us tick
            SETB    TR0             ;  start timer T0
            SETB    ET0             ;  unmask T0 interrupt
uart_su:    SETB    MAX             ;  power up Maxim RS232 converter
            CLR     TR1             ;  stop timer T1
            CLR     TF1             ;  clear T1 overflow
            MOV     TH1,#IBAUD      ;  load baud rate count
            MOV     TL1,#IBAUD      ;  load baud rate count
            MOV     PCON,#ISMOD     ;  SMOD = 1 for baud rate @ 22.1184 MHz
            SETB    TR1             ;  start baud rate timer T1
            MOV     SCON,#ISCON     ;  enable UART mode 1
            MOV     A,SBUF          ;  clear out UART RX buffer
            CLR     A               ;  clear A
            CLR     RI              ;  clear get flag
            CLR     TI              ;  clear TI flag
            ACALL   hello           ;  send start up message
            ACALL   initr           ;  initialize TX & RX
            SETB    SIFLG           ;  set serial in flag active
            MOV     C,ID0           ;  read ID0
            JC      ser_on          ;  skip if no ID0 jumper
            ACALL   hello2          ;  else do AutoSend
ser_on:     SETB    ES              ;  enable serial ISR
isr_on:     SETB    EA              ;  enable interrupts
            SETB    PLLON           ;  activate RX PLL
setup_d:    RET                     ;  setup done

initr:      ANL     BOOT,#1         ;  warm boot (don't reset WBFLG)
            MOV     R0,#35          ;  starting here
            MOV     B,#93           ;  for 93 bytes
            CLR     A               ;  clear A
clr_r:      MOV     @R0,A           ;  clear RAM
            INC     R0              ;  bump RAM pointer
            DJNZ    B,clr_r         ;  loop again
            MOV     R0,#RXMB        ;  load RX buffer pointer
            MOV     R1,#TXMB        ;  load TX buffer pointer
            MOV     R2,A            ;  clear R2
            MOV     R3,#FCSS        ;  seed R3
            MOV     R5,#FCSS        ;  seed R5
            MOV     R6,#FCSS        ;  seed R6
            MOV     R7,#FCSS        ;  seed R7
            CLR     SOPFLG          ;  clear SOPFLG
            SETB    PT0             ;  tick is 1st priority
ini_d:      RET                     ;  done

hello:      MOV     DPTR,#table     ;  point to table
            MOV     B,#12           ;  load loop count in B
            MOV     R7,#0           ;  R7 has 1st table entry
snd_h:      MOV     A,R7            ;  move table offset into A
            MOVC    A,@A+DPTR       ;  load table byte
            CLR     TI              ;  reset TI flag
            MOV     SBUF,A          ;  send byte
nxt_tx:     JNB     TI,nxt_tx       ;  wait until sent
            INC     R7              ;  bump index
            DJNZ    B,snd_h         ;  loop to send message
hello_d:    RET                     ;  done

hello2:     MOV     DPTR,#tbl_2     ;  point to table 2
            MOV     R1,#TXMB        ;  reset TX buffer pointer
            MOV     B,#8            ;  loop count for 8 bytes
            MOV     TMBYC,#0        ;  offset for 1st table entry
snd_h2:     MOV     A,TMBYC         ;  move table offset into A
            MOVC    A,@A+DPTR       ;  load table byte
            MOV     @R1,A           ;  into TX buffer
            INC     TMBYC           ;  increment TMBYC
            INC     R1              ;  increment R1
            DJNZ    B,snd_h2        ;  loop to load message
            MOV     R1,#TXMB        ;  reset TX pointer
            CLR     SIFLG           ;  reset serial input
            SETB    TXFLG           ;  set TX flag
```

```
         SETB      ASFLG          ;  set AutoSend flag
helo2_d  RET

;  tables:

tstrt:   .BYTE     10             ;  preamble/SOP table
         .BYTE     10             ;  table data
         .BYTE     10             ;  table data
         .BYTE     10             ;  table data
         .BYTE     10             ;  table data
         .BYTE     10             ;  table data
         .BYTE     10             ;  table data
         .BYTE     10             ;  table data
         .BYTE     10             ;  table data
         .BYTE     8              ;  table data
         .BYTE     3              ;  table data
         .BYTE     11             ;  table data

smbl:    .BYTE     13             ;  4-to-6 bit table
         .BYTE     14             ;  table data
         .BYTE     19             ;  table data
         .BYTE     21             ;  table data
         .BYTE     22             ;  table data
         .BYTE     25             ;  table data
         .BYTE     26             ;  table data
         .BYTE     28             ;  table data
         .BYTE     35             ;  table data
         .BYTE     37             ;  table data
         .BYTE     38             ;  table data
         .BYTE     41             ;  table data
         .BYTE     42             ;  table data
         .BYTE     44             ;  table data
         .BYTE     50             ;  table data
         .BYTE     52             ;  table data
         .BYTE     00             ;  overflow

table:   .BYTE     192            ;  start up message
         .BYTE     ‘ ‘            ;  table data
         .BYTE     ‘D’            ;  table data
         .BYTE     ‘K’            ;  table data
         .BYTE     ‘1’            ;  table data
         .BYTE     ‘1’            ;  table data
         .BYTE     ‘0’            ;  table data
         .BYTE     ‘K’            ;  table data
         .BYTE     ‘:’            ;  table data
         .BYTE     ‘ ‘            ;  table data
         .BYTE     ‘ ‘            ;  table data
         .BYTE     192            ;  table data

tbl_2:   .BYTE     192            ;  table data
         .BYTE     ‘H’            ;  table data
         .BYTE     ‘e’            ;  table data
         .BYTE     ‘l’            ;  table data
         .BYTE     ‘l’            ;  table data
         .BYTE     ‘o’            ;  table data
         .BYTE     ‘ ‘            ;  table data
         .BYTE     192            ;  table data

         .END                     ;  end of source code
```

5.4 V110T05B.FRM

```
VERSION 5.00
Object = “{648A5603-2C6E-101B-82B6-000000000014}#1.1#0”; ”MSCOMM32.OCX"
Object = “{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0”; ”COMDLG32.OCX"
Begin VB.Form Form1
   Caption         =   “V110T05B Terminal Program for DK110K Protocol”
   ClientHeight    =   4335
   ClientLeft      =   165
   ClientTop       =   735
   ClientWidth     =   6375
   BeginProperty Font
      Name            =   “MS Sans Serif”
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   ‘False
      Italic          =   0   ‘False
```

```
      Strikethrough   =   0   'False
   EndProperty
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   ScaleHeight     =   4335
   ScaleWidth      =   6375
   StartUpPosition =   3  'Windows Default

   Begin MSComDlg.CommonDialog CommonDialog1
      Left            =   240
      Top             =   3600
      _ExtentX        =   688
      _ExtentY        =   688
      _Version        =   393216
   End
   Begin VB.TextBox Text2
      BeginProperty Font
         Name            =   "System"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   2052
      Left            =   120
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      ScrollBars      =   2  'Vertical
      TabIndex        =   1
      Top             =   0
      Width           =   6135
   End
   Begin VB.Timer Timer1
      Left            =   720
      Top             =   3600
   End
   Begin MSCommLib.MSComm MSComm1
      Left            =   1200
      Top             =   3600
      _ExtentX        =   794
      _ExtentY        =   794
      _Version        =   393216
      DTREnable       =   -1  'True
   End
   Begin VB.TextBox Text1
      BeginProperty Font
         Name            =   "System"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   2052
      Left            =   120
      MultiLine       =   -1  'True
      ScrollBars      =   2  'Vertical
      TabIndex        =   0
      Top             =   2160
      Width           =   6135
   End
   Begin VB.Menu mnuFile
      Caption         =   "&File"
      Begin VB.Menu mnuClear
         Caption         =   "&Clear"
      End
      Begin VB.Menu mnuAutoSnd
         Caption         =   "&AutoSend"
      End
      Begin VB.Menu mnuExit
         Caption         =   "E&xit"
      End
   End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
```

```
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

' V110T05B.FRM, 2002.08.07 @ 08:00 CDT
' See RFM Virtual Wire(r) Development Kit Warranty & License for terms of use
' Experimental software - NO representation is
' made that this software is suitable for any purpose
' Copyright(c) 2000 - 2002, RF Monolithics, Inc.
' For experimental use with the RFM DR1200-DK and DR1201-DK
' and DR1300-DK ASH Transceiver Virtual Wire(R) Development Kits
' For protocol software version DK110K.ASM
' Check www.rfm.com for latest software updates
' Compiled in Microsoft Visual Basic 6.0

' global variables
  Dim ASMsg$                                                ' AutoSend string
  Dim ComData$                                              ' string from com input
  Dim ComTime!                                              ' InCom timer
  Dim InDel!                                                ' InCom timer delay value
  Dim FEND$                                                 ' packet framing character
  Dim J As Integer                                          ' FEND$ string position
  Dim Q As Integer                                          ' RPkt$ length
  Dim RPkt$                                                 ' RX message FIFO string
  Dim R2Pkt$                                                ' RX message display string
  Dim KeyIn$                                                ' keystroke input buffer
  Dim Pkt$                                                  ' TX message string
  Dim Temp$                                                 ' temp string buffer
  Dim N As Integer                                          ' TX message byte counter
  Dim TXFlag As Integer                                     ' TX flag
  Dim TXCnt As Integer                                      ' TX try counter
  Dim TXTO As Integer                                       ' TX timeout counter
  Dim ASFlag As Integer                                     ' AutoSend flag

Private Sub Form_Load()

'initialize variables:
  ASMsg$ = "12345678901234567890" & vbCrLf
  ComData$ = ""
  ComTime! = 0
  FEND$ = Chr$(192)
  J = 1
  Q = 0
  RPkt$ = ""
  R2Pkt$ = ""
  KeyIn$ = ""
  Pkt$ = ""
  Temp$ = ""
  N = 0
  TXFlag = 0
  TXCnt = 0
  TXTO = 0
  ASFlag = 0

  Form1.Left = (Screen.Width - Form1.Width) / 2             ' center form left-right
  Form1.Top = (Screen.Height - Form1.Height) / 2            ' center form top-bottom
  Text1.BackColor = QBColor(0)                              ' black background
  Text1.ForeColor = QBColor(15)                             ' white letters
  Text1.FontSize = 10                                       ' 10 point font
  Text2.BackColor = QBColor(0)                              ' black background
  Text2.ForeColor = QBColor(15)                             ' white letters
  Text2.FontSize = 10                                       ' 10 point font

  MSComm1.CommPort = 1                                      ' initialize com port 1
  MSComm1.Settings = "19200,N,8,1"                          ' at 19.2 kbps
  MSComm1.RThreshold = 0                                    ' poll only, no interrupts
  MSComm1.InputLen = 0                                      ' read all characters
  MSComm1.PortOpen = True                                   ' open com port
  InDel! = 0.1                                              ' initialize delay at 100 ms

  Randomize                                                 ' initialize random number generator

  Show                                                      ' show form
  Text1.Text = "**TX Message Window**" & vbCrLf             ' display TX start up message
  Text1.SelStart = Len(Text1.Text)                          ' put cursor at end of text
  Text2.Text = "**RX Message Window**" & vbCrLf             ' display RX start up message
  Text2.SelStart = Len(Text2.Text)                          ' put cursor at end of text

  Timer1.Interval = 300                                     ' 300 ms timer interval
  Timer1.Enabled = True                                     ' start timer

End Sub
```

```
Private Sub Timer1_Timer()
  If TXFlag = 1 Then                                          ' if TX flag set
    Call DoTX                                                 ' send/resend/NAK
  End If
  If MSComm1.InBufferCount > 0 Then                           ' if bytes in input buffer
    Call RxPkt                                                ' call RxPkt
  End If

  If ASFlag = 1 Then                                          ' if AutoSend flag set
    Call ASPkt                                                ' call Autosend
  End If
End Sub

Public Sub RxPkt()
  Call InCom                                                  ' InCom will get it
  Call ShowPkt                                                ' ShowPkt will show it
End Sub

Public Sub InCom()
  On Error Resume Next                                        ' set up error handler
  ComTime! = Timer                                            ' get current time
  Do Until Abs(Timer - ComTime!) > InDel!                     ' get bytes for InDel! interval
    Do While MSComm1.InBufferCount > 0                        ' while bytes are in com buffer
      ComData$ = ComData$ & MSComm1.Input                     ' put them in ComData$
    Loop
  Loop
End Sub

Public Sub ShowPkt()
  RPkt$ = RPkt$ & ComData$                                    ' add ComData$ to end of RPkt$ FIFO
  ComData$ = ""                                               ' clear ComData$ for next time
  Do                                                          ' do until FEND$s gone
    Q = Len(RPkt$)                                            ' Q is RPkt$ packet length
    J = InStr(1, RPkt$, FEND$)                                ' find position of next FEND$
    If (J < 2) Then                                           ' if FEND$ is in the 1st position
      RPkt$ = Right$(RPkt$, (Q - J))                          ' just delete it
    Else                                                      ' else
      R2Pkt$ = Left$(RPkt$, (J - 1))                          ' R2Pkt$ what's left of FEND$
      RPkt$ = Right$(RPkt$, (Q - J))                          ' RPkt$ what's right of FEND$
      If R2Pkt$ <> " ACK" Then                                ' if it's not an ACK
        Call LenTrap                                          ' manage textbox memory
        Text2.SelStart = Len(Text2.Text)                      ' put cursor at end of text
        Text2.SelText = R2Pkt$                                ' show RX message
        Call SndACK                                           ' send ACK back
        R2Pkt$ = ""                                           ' and clear R2Pkt$ for the next time
      ElseIf R2Pkt$ = " ACK" Then                             ' if it is an ACK
        Call LenTrap                                          ' manage textbox memory
        Text1.SelStart = Len(Text1.Text)                      ' put cursor at end of text
        Text1.SelText = " <OK> " & vbCrLf                     ' show OK
        TXFlag = 0                                            ' reset TX flag
        TXCnt = 0                                             ' clear TX counter
        TXTO = 0                                              ' clear TX timeout counter
        Pkt$ = ""                                             ' clear TX packet string
        R2Pkt$ = ""                                           ' and clear RPkt$
      End If
    End If
  Loop Until (J = 0)                                          ' done when there are no more FEND$s
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
  If TXFlag = 0 Then                                          ' if not TX cycle
    KeyIn$ = Chr$(KeyAscii)                                   ' get KeyIn
    If KeyIn$ = Chr$(8) Then                                  ' if it's a backspace from keyboard
      If N > 0 Then                                           ' and character counter > 0
        Pkt$ = Left$(Pkt$, (N - 1))                           ' trim right end of packet
        N = N - 1                                             ' back up character counter
      End If
    ElseIf KeyIn$ = Chr$(13) Then                             ' else if it's a Cr
      Pkt$ = Pkt$ & vbCrLf                                    ' add vbCrLf
      ASMsg$ = Pkt$                                           ' update AutoSend message
      Pkt$ = FEND$ & Pkt$ & FEND$                             ' add framing FENDs
      N = 0                                                   ' reset N
      TXFlag = 1                                              ' set TX flag
      TXCnt = 0                                               ' clear TX try counter
      TXTO = 0                                                ' clear TX timeout counter
    Else
      Pkt$ = Pkt$ & KeyIn$                                    ' else add character to TX message
      N = N + 1                                               ' increment character counter
    End If
    If (N = 23) Then                                          ' if character count 23
      ASMsg$ = Pkt$                                           ' update AutoSend message
```

```
      Pkt$ = FEND$ & Pkt$ & FEND$                          ' add packet framing characters
      N = 0                                                ' reset N
      TXFlag = 1                                           ' set TX flag
      TXCnt = 0                                            ' clear TX try counter
      TXTO = 0                                             ' clear TX timeout counter
    End If
    Call LenTrap                                           ' manage textbox memory
  Else
    KeyAscii = 0                                           ' else don't echo to the screen
  End If
End Sub

Public Sub DoTX()
  If TXTO = 0 Then                                         ' if TX timeout zero
    TXCnt = TXCnt + 1                                      ' increment TX try counter
    If TXCnt = 1 Then                                      ' if TX try count 1
      Call SndPkt                                          ' send packet
      TXTO = 4                                             ' set 0.8 second timeout
    ElseIf (TXCnt > 1) And (TXCnt < 7) Then                ' for try counts 2 through 6
      Call SndPkt                                          ' send packet
      TXTO = 4 + Int(8 * Rnd)                              ' load random TX timeout count
    ElseIf TXCnt >= 7 Then                                 ' else if past 6th try
      Call LenTrap                                         ' manage textbox memory
      Text1.SelStart = Len(Text1.Text)                     ' put cursor at end of text
      Text1.SelText = " <NAK>" & vbCrLf                    ' show NAK
      TXFlag = 0                                           ' reset TX flag
      TCnt = 0                                             ' clear TX counter
      TXTO = 0                                             ' clear TX timeout counter
      Pkt$ = ""                                            ' clear TX packet string
      R2Pkt$ = ""                                          ' clear RPkt$
    End If
  Else                                                     ' else if TX timeout counter not 0
    TXTO = TXTO - 1                                        ' decrement it one count
  End If
End Sub

Public Sub SndPkt()
  If Pkt$ <> "" Then                                       ' if Pkt$ not null
    MSComm1.Output = Pkt$                                  ' send packet
  End If
End Sub

Public Sub ASPkt()
  If TXFlag = 0 Then                                       ' if TXFlag not set
    Temp$ = ASMsg$                                         ' use Temp$ for local display
    Call LenTrap                                           ' manage textbox memory
    Text1.SelStart = Len(Text1.Text)                       ' put cursor at end of text
    Text1.SelText = Temp$                                  ' add message to textbox
    Pkt$ = FEND$ & ASMsg$ & FEND$                          ' add packet framing to message
    TXFlag = 1                                             ' set ACK flag
    TXCnt = 0                                              ' clear TX try counter
    TXTO = 0                                               ' clear TX timeout counter
  End If
End Sub

Public Sub SndACK()
  MSComm1.Output = FEND$ & " ACK" & FEND$                  ' send ACK back
End Sub

Public Sub LenTrap()
  If Len(Text1.Text) > 16000 Then                          ' manage textbox memory
    Text1.Text = ""                                        ' clear TX textbox
    Text1.SelStart = Len(Text1.Text)                       ' put cursor at end of text
  End If
  If Len(Text2.Text) > 16000 Then                          ' manage textbox memory
    Text2.Text = ""                                        ' clear RX textbox
    Text2.SelStart = Len(Text2.Text)                       ' put cursor at end of text
  End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
  MSComm1.PortOpen = False                                 ' close com port
  End                                                      ' done!
End Sub

Private Sub mnuAutoSnd_Click()
  ASFlag = ASFlag Xor 1                                    ' toggle AutoSend flag
  If ASFlag = 0 Then                                       ' if flag reset
    mnuAutoSnd.Checked = False                             ' uncheck AutoSend
    Text1.ForeColor = QBColor(15)                          ' white characters
```

```
  Else                                                    ` else
    mnuAutoSnd.Checked = True                             ` check AutoSend
    Text1.ForeColor = QBColor(10)                         ` green characters
  End If
End Sub

Private Sub mnuClear_Click()
  Text1.Text = ""                                         ` clear TX textbox
  Text1.SelStart = Len(Text1.Text)                        ` put cursor at end of text
  Text2.Text = ""                                         ` clear RX textbox
  Text2.SelStart = Len(Text2.Text)                        ` put cursor at end of text
End Sub

Private Sub mnuExit_Click()
  MSComm1.PortOpen = False                                ` close com port
  End                                                     ` done!
End Sub
```

6 Revisions and Disclaimers

There are several improvements in the example software in this revision. The RF data rate in both link layer protocol examples has been increased from 1200 to 2000 bps, and the packet retry back off interval in DK200A.ASM has been better randomized. The V110T30C host terminal program now supports multi-packet messages and both host terminal programs provide better Windows efficiency. Component values in Figure 4.2 have been adjusted to match the higher RF data rate.

The information in this design guide is for tutorial purposes only. Any software developed using the information provided in this guide should be thoroughly tested before use. No representation is made that the software techniques and example code documented in this guide will work in any specific application. Please refer to the Virtual Wire® Development Kit Software License and Warranty for additional information.

file: tr_swg19.vp, 2002.08.07