PERSISTOR® CF2 Getting Started Guide

Revision 2.0 – July 2005

Standard Terms and Conditions of Use

Life Support and Safety Equipment: PERSISTOR INSTRUMENTS INC (PII) PRODUCTS ARE NOT DESIGNED, TESTED, OR INTENDED FOR USE IN EQUIPMENT THAT MAY, BY FAILURE, MISUSE OR PROPER USE, CAUSE DEATH, INJURY, OR LOSS OF PROPERTY. Designers of such equipment assume full responsibility for protection against, and the liability for damages arising from, the failure of PII products.

Specifications: Product specifications are subject to change without notice. PII reserves the right to improve or change the specifications at any time.

Limited Warranty: PII warrants hardware, which is manufactured or certified by PII, to perform within the published specifications at the time of purchase, for a period of one year from the time of purchase. The buyer assumes the responsibility for confirming products purchased from PII perform within the published specifications when placed in the buyer's application. This warranty does not cover damage or depreciation due to operating conditions or environments, or damage due to abuse, accident, or customer modification. Damage due to static dissipation or corrosive environments are specifically excluded. PII warrants software, which is produced by PII, to perform within the published specifications at the time of purchase, for a period of 90 days from the time of purchase. Products sold by PII, but which are not manufactured or certified by PII, are not warranted by PII.

Remedy for Warranty Failures: PII will, as PII chooses, repair or replace a defective hardware or software product, under the following conditions: the item is returned within the warranty period, a reproducible symptom of the defect can be described, the defective item is not found to be damaged by misuse or abuse.

Limitation of Warranty: TO THE FULLEST EXTENT OF LAW, PII MAKES NO OTHER WARRANTY, EXPRESSED OR IMPLIED; INCLUDING ANY PRESUMED, EXPRESSED OR IMPLIED WARRANTY AS TO THE SUITABILITY OF THE PRODUCT FOR ANY PARTICULAR PURPOSE. TO THE FULLEST EXTENT OF LAW, PII SPECIFICALLY EXCLUDES ANY IMPLIED WARRANTY OF TITLE OR NON-INFRINGEMENT. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY HAVE OTHER LEGAL RIGHTS, WHICH VARY FROM STATE TO STATE.

Limitation of Liability: TO THE MAXIMUM EXTENT ALLOWED BY LAW, IN NO EVENT SHALL PII OR ITS SUPPLIERS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION INJURY OR DEATH, LOST OR INADEQUATE DATA OR INFORMATION, LOST REVENUES OR PROFITS, LOST USE OR BUSINESS INTERRUPTION, REPLACEMENT OR RENTAL COSTS ARRISING FROM THE FAILURE OF, OR INABILITY TO USE PII PRODUCTS. IN NO EVENT SHALL PII'S LIABILITY FOR ANY DAMAGES EXCEED THE PURCHASE PRICE OF THE PRODUCT.

Trademark Notice

Persistor®, PicoDOS®, and MotoCross® are registered trademarks of Persistor Instruments Inc. RecipeCard™ and SandwichCard™ are unregistered trademarks of Persistor Instruments Inc. All other names, brands, and trademarks are the property of their respective owners.

Table of Contents

General Information

Introduction

Thank you for your recent purchase of a Persistor CF2. We welcome your comments and suggestions for improving our documentation.

This getting started guide describes how to setup and install the various components that make up your Persistor CF2 Starter Kit. This printed manual covers just the essentials of getting started with the CF2. After you have installed all of the software, the Persistor directory will contain much more documentation in the form of PDFs for the various components. You will also have access to an HTML file that will have descriptions of the PicoDOS API as well as more details on the operation of the CF2.

The combination of a CF2 and RecipeCard allow you to get started on your experiment goals without putting a lot of front-end effort into wiring up the hardware. In many cases, this will be all you need for the initial prototype, proof-of-concept, or feasibility studies. Complete schematics, part lists, and design notes for Persistor RecipeCards are available in the data sheets for the RecipeCards..

What’s in the Kit?

Your CF2 Starter Kit should contain the following items:

1	Persistor CF2	The particular model supplied will depend upon the starter kit ordered.	
		PERCF2I5	has 512K of RAM installed.
		PERCF21M	has 1MB of RAM installed
1	RecipeCard	Once again, the particular model supplied will depend upon the starter kit ordered. The RecipeCard minimally provides connections for power and the primary UART.	
		R212	3” X 5” with 12 bit A/D
		R216AU	3” X 5” with 16 bit A/D, 1 SCI + 2 TPU UARTs
		MRCP	1.4” X 6” with 16 bit A/D, 1 SCI + 1 TPU UARTs
1	CompactFlash card	We may change the size and type as market conditions require.	
1	Communication cable	Type supplied depends on the particular RecipeCard included.	
1	Power cable	With tinned wire ends.	
1	CF-CD	PicoDEV development CD contains documentation, MotoCross post compiler and terminal application, CF2 function libraries, and programming examples	

Additional Equipment Required

A current limiting power supply (Please read the Warnings and Precautions section)
A PC running Win95/98/2000/NT
Metrowerks CodeWarrior for Palm OS Version 8 or 9

Warnings and Precautions

We really don't want to dampen the excitement of exploring your new board, but there's some stuff you really ought to know. Even the old-timers may find something new to worry about with this 3.3-volt system and its lowest power suspend mode. Just take a minute to skim this short section and save the possible embarrassment and expense of having to admit you leaped before looking.

The CF2 is a 3.3 volt system, and cannot tolerate any voltages above 3.6 volts on any of its I/O or BUS lines, except for the RS-232 signals (RSTXD, RSRXD, RSCTS, RSRTS). Even momentary connection to 5-volt signals will likely result in permanent damage to the board's components. ***Do not attempt to get around this by running the board at 5 volts as the RAM, and especially the flash will suffer stress damage!***

Suspend Mode: When the CF2 goes into suspend mode, all of the I/O and BUS pins (except for the RS232, /WAKE, and /SHDN) look like very low impedance current sinks with about a 1.3-1.8 volt forward drop. CF2 I/O or BUS lines being driven from off-board peripherals will try very hard (and succeed) at pulling these levels low, which is probably not what you want. Any I/O lines being pulled high to an external V+ source will be pulled down to this forward voltage drop. Both of these situations will consume lots of current which defeats the purpose of suspend mode.

Static Sensitive CMOS: Every component on the CF2 is CMOS and susceptible to immediate damage, or worse, premature field failures if you don't take precautions to guard against damage from static electricity.

Develop with a current limited power supply! You can save yourself a lot of grief by running the board from a bench supply current limited to about 100mA. Jumpers, test probes, and programming bugs make it very easy to send the CF2 into some horrible current sucking latchup mode and current limiting can help keep a spurious slip from destroying your board.

Develop at low voltage! The CF2 onboard voltage regulator can handle +/-20 volt inputs, but nothing else on the board can. Just like current limiting, developing at around 4 volts is a good way to keep a simple slip of the hand from destroying your board. RecipeCards and SandwichCards may have lower power supply voltage limits than the CF2.

Floating Inputs: Most of the I/O lines on the CF2 do not have onboard pull-up resistors, and most of these are left in their default input state at reset. CMOS floating inputs draw current in a somewhat unpredictable fashion - nowhere near enough to do any damage, but enough to defeat the gains of some of the power saving modes. You should either pull unused I/O lines to VREG or GND, or force them to be outputs.

Don't Stick Probes in the Header Sockets: The header strips used in the RecipeCards are meant to accept 0.025" square posts. Anything else is likely to permanently deform the connectors and cause your system to fail or behave erratically. We did this here with a scope probe tip (0.037" diameter), and a customer did it with a miniature DMM probe (0.044" diameter). We both spent many frustrating hours searching for the source of bizarre problems. The deformation damage is quite visible, but only with the help of a microscope - and no, this would not be covered under the warranty.

Backup battery: The CF2 does not require a backup battery to operate when used with supervisor 5.20 or later. However, if you intend to use the low power suspend modes of the CF2 or if you will use dynamic CompactFlash card changes (CCC from PicoDOS) then you WILL need a backup battery.

QSPI: The CF2's 68332 and MSP430 supervisor communicate over the SPI bus using MISO, MOSI, and SCK. Do not allow your SPI peripherals to drive MISO when they are not selected.

CompactFlash Compatibility: Persistor Instruments sells and supports Silicon Systems brand Industrial Grade CompactFlash for the CF2 for several reasons:

Wear leveling which is performed inside Silicon Systems cards in a manner transparent to the CF2. Wear leveling is mandatory for reasonable card life in DOS/Windows compatible FAT file systems where a few crucial system sectors experience vastly disproportionate erase/write wear. Silicon Systems holds patents on their wear leveling technology.

The CF2 will not work with all brands of CompactFlash. Version 1.4 of the CompactFlash specification released in mid-1999 allows optional pull-down data bus. The CF2 ties directly to the data bus. Pull down resistors on a Motorola CPU32 based controller can keep the processor from starting and we have seen this with LexarMedia cards. PicoDOS takes special measures to minimize the occurrence, but it still can happen on "dirty" power ups.

For cost sensitive applications, Persistor also resells standard retail outlet SanDisk CompactFlash cards but these are not recommended for important data acquisition projects, and are not covered by Persistor Instruments' warranty.

Software Installation

Overview

Persistor's PicoDEV tools support CF2 68K cross-development using the Palm OS version of the Metrowerks CodeWarrior compiler suite. This section describes the three installation steps required to prepare your Windows PC for building CF2 programs:

1) **Install CodeWarrior for Palm OS® Version 8 or 9**
2) **Install PicoDEV Tools for Persistor CF2**
3) **Specify Source Trees for your CF2Development**

In order to program your CF2, two separate software installations are required on your Windows PC. **Metrowerks CodeWarrior™ for Palm OS®** is the required C programming shell and compiler for your programming projects.

A Note to CodeWarrior™ Pro 6 Users

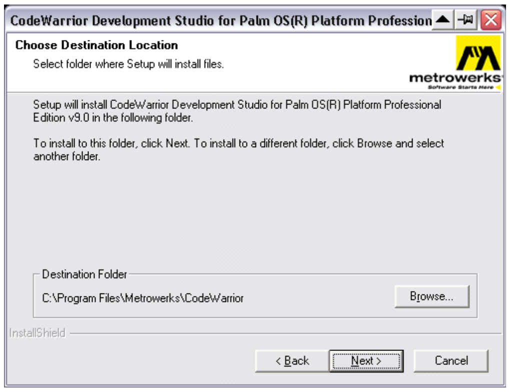
Although CodeWarrior Pro 7 and beyond no longer support 68K programming, we do support CF2 programming using CodeWarrior Pro 6 for customers migrating from the CF1. If your PC is already setup for CWPro6 with CF1 tools, you will need to skip the Install CWPalm step and manually remove the legacy CF1 components as follows. You will still be able to program the CF1 but this will use newer tools compatible with both CF1 and CF2 that get installed in the Install PicoDEV step.

1) Using Windows Explorer, navigate to the CodeWarrior directory. This should be something like **C:\Program Files\Metrowerks\CodeWarrior**.
2) Remove the "**MotoCross Support**" folder.
3) Move into the folder labeled "**Stationery**". Remove all of the folders that begin with CF1 or MotoCross.

Install CodeWarrior™

Your first step toward CF2 development requires installing Metrowerks CodeWarrior for Palm OS. The minimum requirements are:

1) Windows 95, 98, ME, NT, 2000, XP
2) Pentium (recommended), 80386, or 80486
3) 64MB RAM,
4) 220 MB available hard disk space
5) CD-ROM for installation



Choose Destination Location

Select folder where Setup will install files.

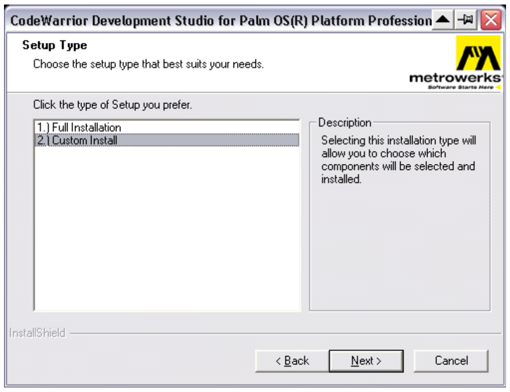
Setup will install CodeWarrior Development Studio for Palm OS(R) Platform Professional Edition v9.0 in the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder

C:\Program Files\Metrowerks\CodeWarrior

CF2 development requires that you install CodeWarrior using the Metrowerks CD installer. NOTE: If you have an existing CWPalm version 8 or 9 installation that compiles Palm OS 68K programs, you can skip right to the Install MotoCross section. If you're using CodeWarrior solely for CF2 development, you need only select the options checked in the screen captures below, which will require about 40MB of disk space. Keep an eye on the dialog's Space Required text. If it varies a lot from the screen snapshots, carefully review your selections. If you have trouble, it's perfectly safe to reinstall CodeWarrior in part or whole, and that generally will get you going.


CodeWarrior Development Studio for Palm OS(R) Platform Profession
metrowerks
Software Starts Here
InstallShield
< Back
Next >
Cancel

Setup Type

Choose the setup type that best suits your needs.

Click the type of Setup you prefer.

1.) Full Installation
2.) Custom Install

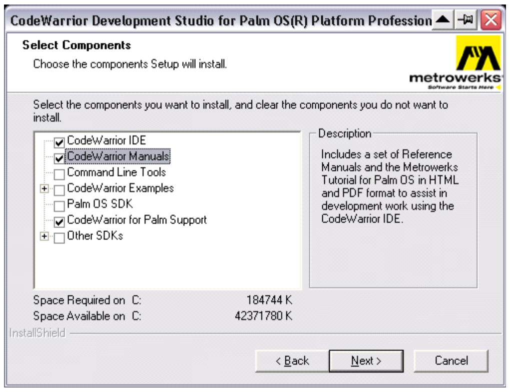
Description

Selecting this installation type will allow you to choose which components will be selected and installed.

Start by inserting the CodeWarrior CD. On most PCs, you will automatically be presented with the installation dialog. On some others, you will have to double click on the CD icon and setup.exe to get things started. Click your way through the first half-dozen or so introductory and license information dialogs. Begin to pay careful attention when you get to the **Choose Destination Location** dialog.

For a painless introduction to CF2 development, you really do want to accept the default location offered by the installer:

C:\Program Files\Metrowerks\CodeWarrior


CodeWarrior Development Studio for Palm OS(R) Platform Profession
metrowerks
Software Starts Here
InstallShield
< Back
Next >
Cancel

Select Components

Choose the components Setup will install.

Select the components you want to install, and clear the components you do not want to install.

- [✓] CodeWarrior IDE
- [✓] CodeWarrior Manuals
- [ ] Command Line Tools
- [ ] CodeWarrior Examples
- [ ] Palm OS SDK
- [✓] CodeWarrior for Palm Support
- [ ] Other SDKs

Description

Includes a set of Reference Manuals and the Metrowerks Tutorial for Palm OS in HTML and PDF format to assist in development work using the CodeWarrior IDE.

Space Required on C:	184744 K
Space Available on C:	42371780 K

At the **Setup Type** screen, select the **Custom Install** option.

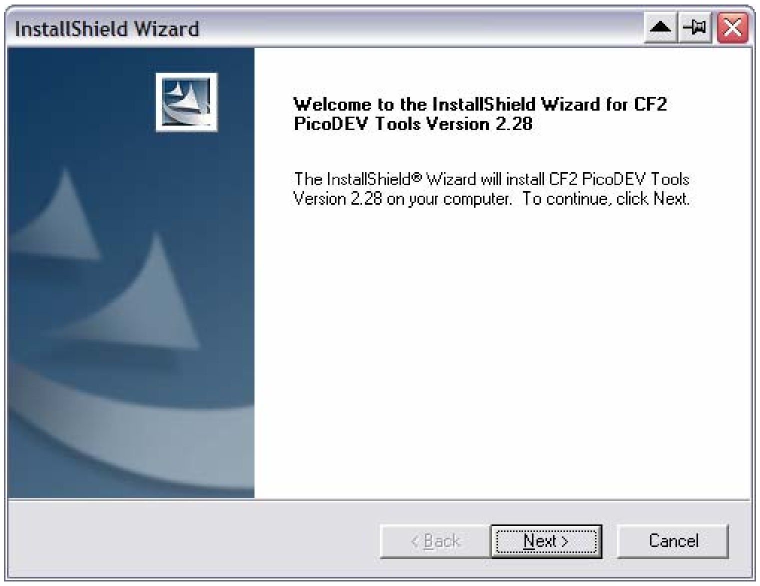
At the **Select Components** screen uncheck everything except: "**CodeWarrior IDE**", "**CodeWarrior Manuals**" and "**CodeWarrior for Palm Support**"

Click your way through the remaining installer screens.

After completing the installation the Programs selection in your Start Menu will now contain an option for Metrowerks CodeWarrior IDE.

Install PicoDEV with MotoCross®

After you've installed CodeWarrior, insert the PicoDEV CD. On most PCs, you will automatically be presented with the installation dialog. On some others, you will have to double click on the CD icon and setup.exe to get things started. Work your way through the setup screens accepting the default options until the installation is complete. The suggested directory for the software to be installed is C:\Program Files\Metrowerks\CodeWarrior. In order for everything to work properly we strongly suggest that you accept this path for the installation.


InstallShield Wizard

Welcome to the InstallShield Wizard for CF2 PicoDEV Tools Version 2.28

The InstallShield® Wizard will install CF2 PicoDEV Tools Version 2.28 on your computer. To continue, click Next.

The Programs selection in your Start Menu will now contain an entry named Persistor containing the PicoDEV documentation index and the application; MotoCross for PicoDEV.

Setup the Source Trees

MotoCross uses a powerful feature of CodeWarrior called Source Trees to steer projects to the proper directories for locating target components. The first time you use CodeWarrior with MotoCross, you must tell it where to look for the correct CFX build model by importing one of the pre-configured XML Source Tree panels. After that, source tree operation becomes completely transparent.

1) Start CodeWarrior from the Start / Programs menu.
2) Select **Preferences** from the main CodeWarrior IDE **Edit** menu
3) Click-select "**Source Trees**" from the IDE Preference Panels (left)
4) Click the **Import Panel...** button from the Source Trees pane (lower right)
   Navigate to C:\Program Files\Persistor\MotoCross Support\CFX\XMLSettingsPanels\IDE
   You should see the two primary configurations:

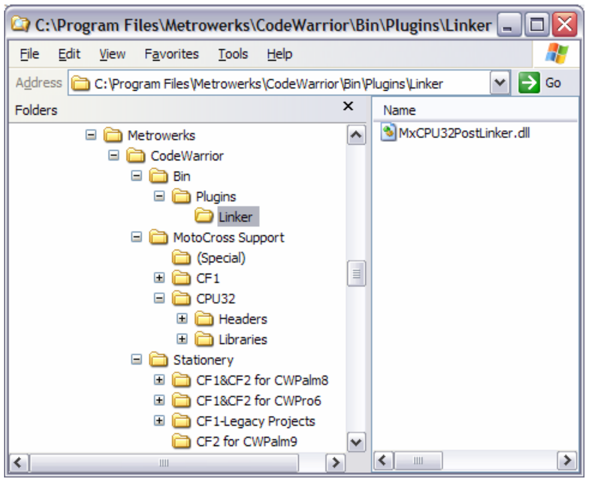
   Source Trees_CF2_IDEGlobalPref.xml
   Source Trees_CF1_IDEGlobalPref.xml

5) Choose Source Trees_CF2_IDEGlobalPref.xml
6) Click the **OK** button back in the Source Trees pane

Installed Files

Metrowerks\CodeWarrior

The screen snapshot below shows the critical PicoDEV files and directories that were installed into the CodeWarrior directory.



C:\Program Files\Metrowerks\CodeWarrior\Bin\Plugins\Linker

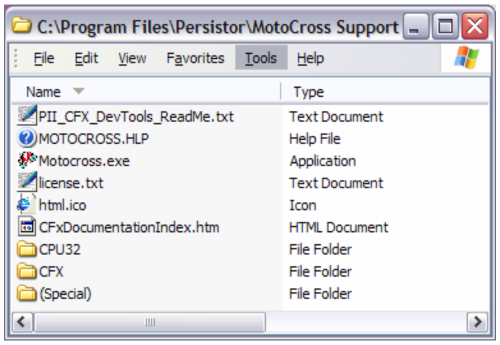
Address: C:\Program Files\Metrowerks\CodeWarrior\Bin\Plugins\Linker

```
Folders
Metrowerks
  CodeWarrior
    Bin
      Plugins
        Linker
    MotoCross Support
      (Special)
      CF1
      CPU32
        Headers
        Libraries
    Stationery
      CF1&CF2 for CWPalm8
      CF1&CF2 for CWPro6
      CF1-Legacy Projects
      CF2 for CWPalm9
```

Name: MxCPU32PostLinker.dll

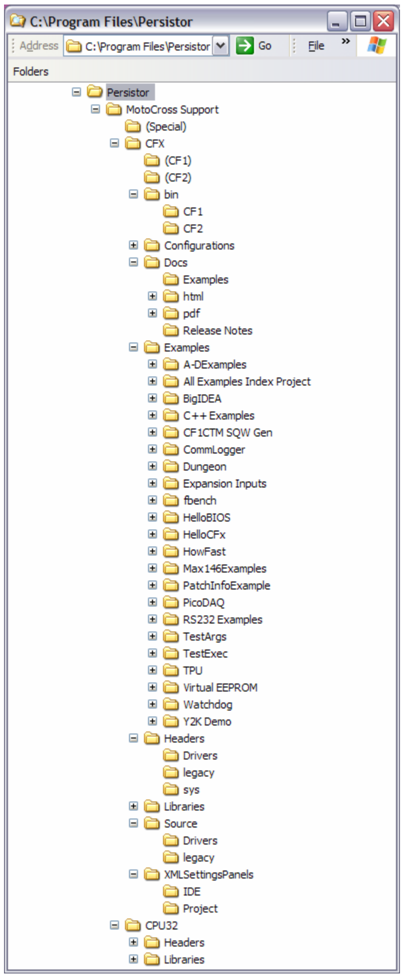
Persistor\MotoCross Support

The snapshot below gives a high level overview of the installation into the Persistor directory. Use this to familiarize yourself with the general layout and structure of the MotoCross development tools and as a map for quickly locating the files and directories referenced in this Getting Started Guide.



C:\Program Files\Persistor\MotoCross Support

Name	Type
PII_CFX_DevTools_ReadMe.txt	Text Document
MOTOCROSS.HLP	Help File
Motocross.exe	Application
license.txt	Text Document
html.ico	Icon
CFxDocumentationIndex.htm	HTML Document
CPU32	File Folder
CFX	File Folder
(Special)	File Folder



C:\Program Files\Persistor

Address: C:\Program Files\Persistor

```
Folders
Persistor
  MotoCross Support
    (Special)
    CFX
      (CF1)
      (CF2)
      bin
        CF1
        CF2
      Configurations
      Docs
        Examples
        html
        pdf
        Release Notes
      Examples
        A-DExamples
        All Examples Index Project
        BigIDEA
        C++ Examples
        CF1CTM SQW Gen
        CommLogger
        Dungeon
        Expansion Inputs
        fbench
        HelloBIOS
        HelloCFx
        HowFast
        Max146Examples
        PatchInfoExample
        PicoDAQ
        RS232 Examples
        TestArgs
        TestExec
        TPU
        Virtual EEPROM
        Watchdog
        Y2K Demo
      Headers
        Drivers
        legacy
        sys
      Libraries
      Source
        Drivers
        legacy
      XMLSettingsPanels
        IDE
        Project
    CPU32
      Headers
      Libraries
```

Location, Location, Location

Don't put your files or projects in either the Persistor or Metrowerks directories!

The Persistor folder above right shows a more exploded view of its contents. CodeWarrior is very fussy about the relative locations of its various components, and the pre-configured Persistor stationery makes similar location-relative assumptions. If things get moved around, expect your compilations to start failing.

All of your work should be done in completely separate directories with whatever convention you find best. For our projects, we typically use:

C:\piisoft\CF2\

You can even have projects scattered over many directories or drives. The main point is: ***Don't put your files or projects in either the Persistor or Metrowerks directories!*** They will be lost when you install updated versions of CodeWarrior or PicoDEV, and this is likely to happen at some point. The only things that should go into the Metrowerks directories are installations from Metrowerks. You should also assume that files in the Metrowerks directory will get deleted during update installations.

Hookup and Sign On

Introduction to RecipeCards™ and SandwichCards™

Common Persistor terms used in the CF2 documentation are RecipeCard™ and SandwichCard™. A RecipeCard is typically 3” X 5” and provides a socket set for mounting, powering, and communicating to the CF2. A SandwichCard is designed to be placed in the center of a “sandwich” between a RecipeCard and a CF2.

Persistor Instruments publishes a design guide that allows third parties to manufacture RecipeCards and SandwichCards which are bus compatible and addressable by the CF2. If you are interested in producing cards that add functionality to the CF2 platform, please contact Persistor for a SandwichCard Design Guide.

What Are You Using?

It is possible to order various CF2 starter kits which contain different RecipeCards; each provided with specific information. For your first hookup and run of the CF2 you will need to make sure you can identify the power cable and the primary communication cable. As of this writing, all Persistor brand RecipeCards use the same power cable. The primary communication cable may vary however. Here is a guide to make sure you have the correct primary communication cable.

Persistor RecipeCard	**Primary Comm Cable**	**Secondary Comm Cable**
PRCPDAQ	CAB-COM-RCP	none
R212	CAB-COM-RCP	none
R216AU	CAB-COM-RCP	CAB-COM2-RCP
MRCP	CAB-COM2-RCP	CAB-COM2-RCP

The CAB-COM-RCP is a ribbon cable with a 2X5 square pin header on the RecipeCard end and a DB-9F connector on the PC end. The CAB-COM2-RCP has a 3.5mm stereo plug on the RecipeCard end.

Hook Up

Follow these steps to get your CF2 to sign on.

1) Use a static safe work area.

2) Insert the CF2 into the sockets for it on the RecipeCard. Make sure the CF2 is not offset in any direction from the connectors. The CompactFlash card is not required for CF2 operation. For our first hook up, however, install the card.

3) Install the primary communication cable between the RecipeCard and your PC. Make sure you are using the primary communication jack and cable. See 'What Are You Using' above if you are unsure.

4) Launch MotoCross from the Persistor options under the Start Menu on your PC. You should get a blank monitor window. (If you wish to use another monitor program, set it to: direct connection, 9600 baud, no parity, 8 data bits, and 1 stop bit.)

5) Use a current limited power supply. For this first sign on, you need no more than 4 volts, and your power supply should be limited to about 40mA. Be sure to read the section of 'Warnings and Precautions'.

6) Apply your current limited power to the power cable. Black is ground, and red is positive.

7) You should see the CF2 sign on in the monitor window, something like this:

```
----------------------------------------------------------------
  Persistor CF21M   SN 03117    PicoDOS V3.10r1      PBM V2.27
  (C) 1998-2004 Persistor Instruments Inc. - www.persistor.com
----------------------------------------------------------------
C:\>
```

If your CF2 does not sign on, immediately remove the power and recheck your connections.

The PicoDOS prompt (C:\>) is a standard DOS-like drive identifier. You should see the drive letter C: as a part of the prompt when a CompactFlash card is installed. When no CompactFlash card is present in the CF2 when power is applied, PicoDOS displays a warning that many functions are disabled and reverts to the CF2 prompt like this:

```
CompactFlash card missing, most features disabled
CF2>
```

Eject the CompactFlash card and type `reset↵`. (r e s e t and then the enter key) You should see:

```
----------------------------------------------------------------
  Persistor CF21M   SN 03117    PicoDOS V3.10r1      PBM V2.27
  (C) 1998-2004 Persistor Instruments Inc. - www.persistor.com
----------------------------------------------------------------
CompactFlash card missing, most features disabled
CF2>
```

From either prompt in PicoDOS you can type the VERsion command. Type `ver⏎`. You should see system details like this:

```
CF21M   SN 03117
PicoDOS 3.10r1
BIOS    3.10r1
PBM     2.27
SPV     5.20-P
TPU     P.01
TLC     1.03
```

Typing `help⏎` will always give you a list of the available PicoDOS commands. So will h, or ?.

CF2 C Programming Tutorial

Before we get started with the tutorial, we should mention a few things.

First, this tutorial assumes that you understand a little about the Metrowerks IDE. There is no substitute for going through the CodeWarrior IDE Users Guide (you can find this under the Help menu in CodeWarrior). But if you want to get a jump-start on developing, just go through this next example where we will tell you exactly what to do. However, before you begin this tutorial, we must first explain a few terms that you will see:

IDE – This stands for Integrated Development Environment. An IDE is a program that contains all of the various elements needed for software development. The Metrowerks IDE provides an editor, and a compiler in one program. It also gives us a way to manage the individual files that makes up the program that you are writing (see Project Window).

Project Window – The project window is a tabbed dialog box that holds information about the files and settings of the project loaded by the IDE. The tabs are Files (the files that make up the project e.g. C files, headers and library files), Segments (object code), and Targets (settings about the project(s).

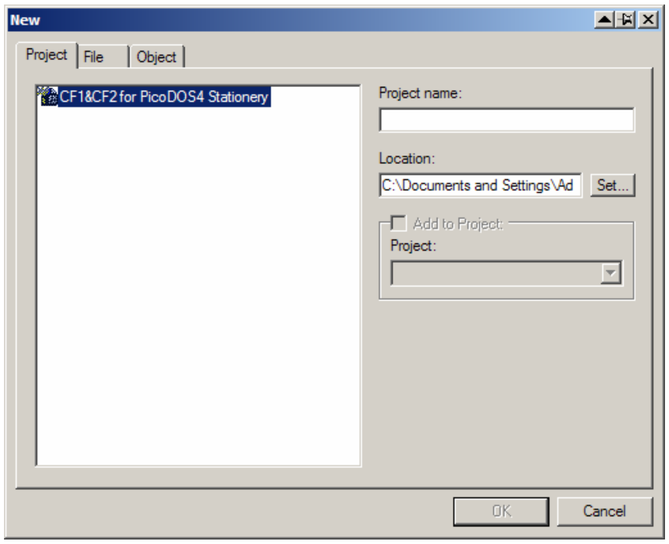
Target – A project can have more than one Target. A Target contains specific settings. If you are new to CodeWarrior you may only build projects with one Target. However, as you gain experience and become more advanced, multiple Targets will become a powerful way to build applications.

68K Target – This refers to the actual 68XXX code that is generated by the compiler (a program destined for the CF2 in our case).

Stationery – Just like paper stationery serves as a foundation for writing letters, Stationery in CodeWarrior is the term for starter code for applications. Several different kinds of Stationery are provided to get you started with your application.

The following tutorial assumes that you have followed the correct installation procedure for both CodeWarrior and MotoCross. It also assumes that you understand the software development process of writing and compiling C programs.

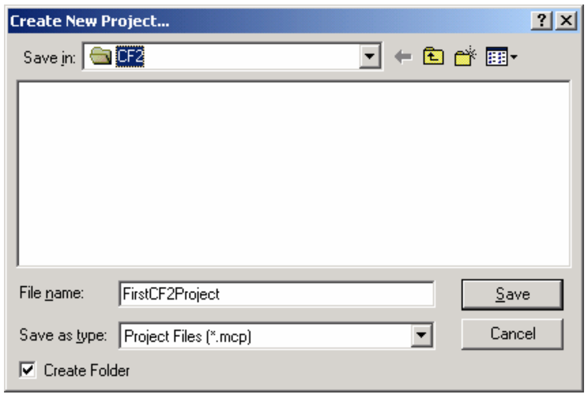
Your First Project



To begin, launch CodeWarrior. Select File:New…. A "New" window will appear allowing you to select from a list of stationery. You will see options for several types of projects. Choose **CF1&CF2 for PicoDOS4 Stationery**.

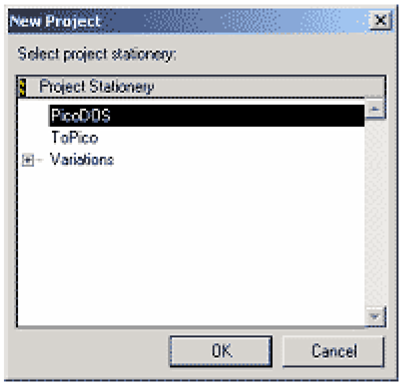
Type the desired name in the Project name box (we'll use FirstCF2Project). You will also see a field in the dialog box that shows the Location of the project.

Another way to set the project name and the location at the same time is to click on the Set button to the right of the Location field. This will bring up a standard dialog box that will allow you to navigate to a location and type the name of the project. When you click OK on that dialog box you will go back to the New project dialog box. You will then notice that both the Project name and Location are filled in for you.



After you have entered a Project name just click "OK" to go to the next step.

The next "New Project" window is for you to specify which type of CF2 project you want to create. You should choose PicoDOS for a first project. Select PicoDOS by left clicking on it to highlight it. When you have done that click the "OK" button.



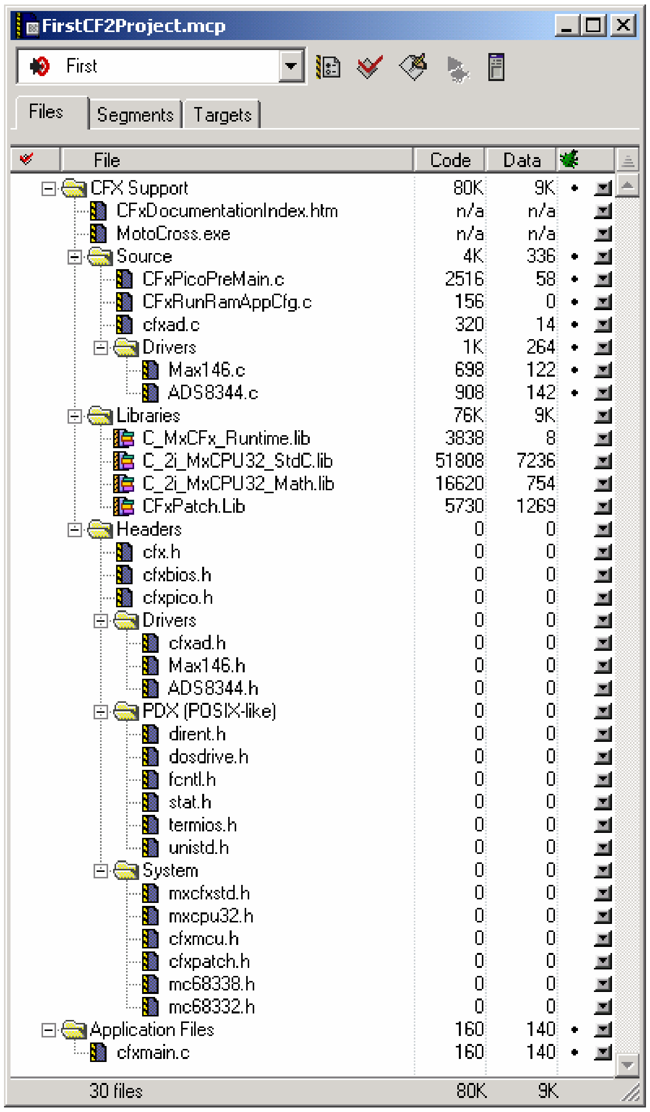
Once the above steps are completed, you will be presented with a CodeWarrior Project window. There are several "groups" within the project window. These are merely organizational tools and do not reflect any "on disk" structure for the project. When first opened, these groups are collapsed and you can click on their expander controls (the boxes on the left) to view the files in the groups. You may open any of the files by double-clicking on them. What follows is a list of groups, files contained within them and a brief explanation.

1. **CFx Support** - This group contains files crucial to compiling projects for the CF2 as well as documentation files. Other files include a special version of the C standard library, a math support library for the CF2, a special startup library that sets up your C program to run in the CF2 environment and header files. You will generally never make changes to this group or to the files in this group.
   a) CfxDocumentationIndex.htm – If double-clicked on, launches the CF1/CF2 Documentation Index which is an HTML file created during installation.
   b) MotoCross.exe – The Pii communications program for Windows. You use MotoCross to communicate with and load software to the CF2.

c) **Source**
   i) CFxPicoPreMain.c – startup code for the project
   ii) CfxRunRamAppCfg.c - Persistor CFx PicoDOS Program Configuration Data
   iii) cfxad.c - Generic SPI A-D QPB Driver for PicoDOS
   iv) Drivers
   v) Max146.c – File for working with the MAXIM MAX146 12 bit SPI A/D converter.
   vi) ADS8344.c – File for working with the Burr-Brown ADS8344 16 bit SPI A-D converter.

d) **Libraries**
   i) C_MxCFx_Runtime.lib
   ii) C_2i_MxCPU32_StdC.lib
   iii) C_2i_MxCPU32_Math.lib
   iv) CfxPatch.Lib

e) **Headers**
   This folder contains links to the header files used in development for the CF2. If you need to look up any function prototypes or typedefs used with any of the Persistor API functions or PicoDOS, you can find those files here. Once again, you can open the files by simply double clicking on them.
   i) Cfx.h – CFx Target selection
   ii) Cfxbios.h - Persistor BIOS and I/O Definitions
   iii) Cfxpico.h - Persistor PicoDOS Definitions
   iv) Drivers
      (1) Cfxad.h
      (2) Max146.h
      (3) ADS8344.h
   v) PDX
      (1) Dirent.h - PicoDOS POSIX-like Directory Access Defines
      (2) Dosdrive.h - PicoDOS DOS Drive and Directory Definitions
      (3) Fcntl.h - PicoDOS POSIX-like File Access Definitions
      (4) Stat.h - PicoDOS POSIX-like File Status Definitions
      (5) Termios.h - PicoDOS POSIX-like Terminal I/O Definitions
      (6) Unistd.h - PicoDOS POSIX-like UNIX Function Definitions
   vi) System
      (1) Mxcfxstd.h - Persistor Prefix File for MotoCross and CodeWarrior
      (2) Mxcpu32.h - Persistor Prefix File for MotoCross and CodeWarrior
      (3) Cfxmcu.h - Processor Module Map
      (4) Cfxpatch.h - Persistor BIOS and PicoDOS Patches
      (5) Mc68338.h - Persistor CF1 68CK338 Module Map
      (6) Mc68332.h - Persistor CF2 68332 Module Map



```
File                                Code    Data
CFX Support                          80K     9K  •
    CFxDocumentationIndex.htm        n/a     n/a
    MotoCross.exe                    n/a     n/a
    Source                            4K     336 •
        CFxPicoPreMain.c             2516     58 •
        CFxRunRamAppCfg.c             156      0 •
        cfxad.c                       320     14 •
        Drivers                        1K    264 •
            Max146.c                  698    122 •
            ADS8344.c                 908    142 •
    Libraries                         76K     9K
        C_MxCFx_Runtime.lib          3838      8
        C_2i_MxCPU32_StdC.lib       51808   7236
        C_2i_MxCPU32_Math.lib       16620    754
        CFxPatch.Lib                 5730   1269
    Headers                             0      0
        cfx.h                           0      0
        cfxbios.h                       0      0
        cfxpico.h                       0      0
        Drivers                         0      0
            cfxad.h                     0      0
            Max146.h                    0      0
            ADS8344.h                   0      0
        PDX (POSIX-like)                0      0
            dirent.h                    0      0
            dosdrive.h                  0      0
            fcntl.h                     0      0
            stat.h                      0      0
            termios.h                   0      0
            unistd.h                    0      0
        System                          0      0
            mxcfxstd.h                  0      0
            mxcpu32.h                   0      0
            cfxmcu.h                    0      0
            cfxpatch.h                  0      0
            mc68338.h                   0      0
            mc68332.h                   0      0
Application Files                     160    140 •
    cfxmain.c                         160    140 •
30 files                              80K     9K
```

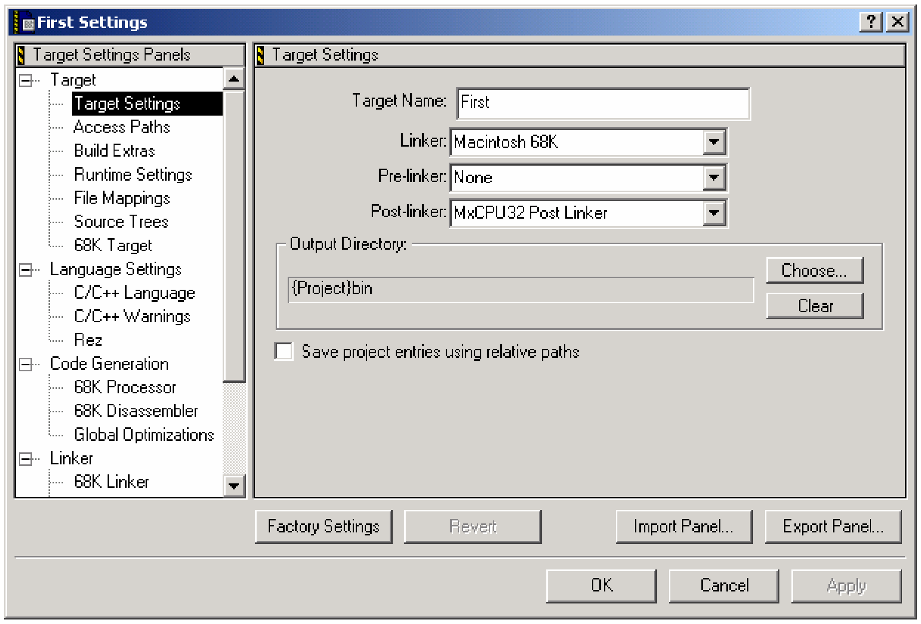
2. **Application Files** - This is the folder where you can store all of your application source files. You will notice that it already contains a file called **cfxmain.c**. This is a starter file that we have provided to help make it easier for you to begin development. It contains the all of the standard ANSI header file includes as well as all of the device specific includes that you will need to work with the CF2. The actual location of these files is in the directory: C:\Piisoft\ which was created when you installed the PicoDEV CD.

Once you become familiar with CodeWarrior and CF2 development, you can begin to tailor the organization of projects to meet the specific needs of your application. The format described above works well for most projects, but just like the arrangement of files on your PC desktop, it's really just an organizational convenience.

Renaming the Target

Before you try to compile the test program we have created you should first rename the target that we will create. You can see that the default name in the box at the top of the project window is Alt+F7>TgtSet>Rename!>>. This is done as a reminder for you to rename the project.

To rename the project, click on the tab at the top of the project window titled 'Targets.' In the project window under Targets, double-click on Alt+F7>TgtSet>Rename!>>. This will bring up the Target settings dialog box.



Target Name: First
Linker: Macintosh 68K
Pre-linker: None
Post-linker: MxCPU32 Post Linker
Output Directory: {Project}bin
☐ Save project entries using relative paths

In the Target Settings Panel on the left you should see a tree of items already expanded for you. The top one is titled 'Target' and under it you will see an item labeled 'Target Settings.' Left clicking on this will bring up the Target Settings panel on the right side of the dialog box. In the Target settings area on the right, change the Target Name to something else. We will choose 'First.'

Now left click on '68L Target in the expanded tree on the left side of the dialog box. This will bring up new information about the output file on the right side of the dialog box. Change the name under File Name to something else. This is the name of the output file that will be generated. We will choose the name First here as well but the name can be different from the Target Name.

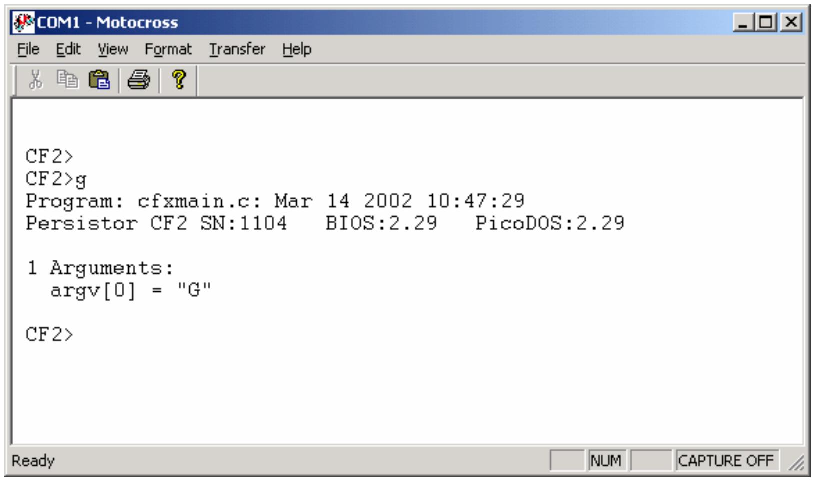
Click 'OK' to close the Target Settings dialog box.

Compiling

The PicoDOS project stationery contains a simple bit of C code that prints out the serial number of your CF2 as well as information regarding the program build and the versions of the BIOS and PicoDOS that are burned into your CF2's flash. You may wish to look at the file **cfxmain.c** before compiling it. Once you are comfortable with the code you can compile the code by selecting Project:Make from the menu or by simply pressing F7. If everything is installed correctly this should build a binary executable that the MotoCross program can then load into your CF2.

During the Make or 'build' operation, the files and groups in your project window will update with information about the size of the code and data required for each module as shown. These numbers reflect the worst-case usage, and the smart linker will generally reduce these number by quite a bit. Map files created by CodeWarrior and MotoCross give complete details on memory usage and mapping.

If you don't already have MotoCross running, you can launch it by double-clicking the MotoCross.exe file under CFx Support in your project window. Select Transfer:Load… then navigate to your project's bin directory and select the file with the ".run" extension. This is made easier for you if you click on the drop-down item **Files of type** and select **RAM Application (*.RUN)** from the list of choices. Once you have selected the file click on the **Open** button to begin the transfer. In just a few seconds, MotoCross will load the file into the CF2 and leave you in its terminal window where you'll see something like what is shown.

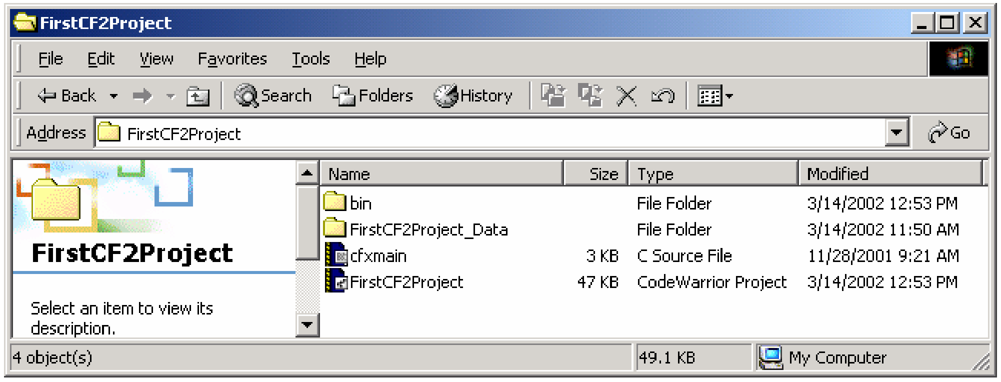


```
CF2>
CF2>g
Program: cfxmain.c: Mar 14 2002 10:47:29
Persistor CF2 SN:1104   BIOS:2.29   PicoDOS:2.29

1 Arguments:
  argv[0] = "G"

CF2>
```

This prompt indicates that the code was loaded into your CF2 and the G (which was automatically sent by MotoCross) is an abbreviation for the GO command that will launch your program. At this point, if you press Enter, your program will execute.

What Happened?

Congratulations, you have just compiled and run your first program for your CF2. When you chose the CF2 PicoDOS stationery and gave it the name FirstCF2Project, CodeWarrior created a new folder and populated it with the project file (FirstCF2Project.mcp), a starter C source file (cfxmain.c), a project data folder (FirstCF2Project Data) and a bin folder to hold compiled binary code. CodeWarrior also automatically opened the project and added it to the list of recent projects so you can quickly open it again from the File menu.



Name	Size	Type	Modified
bin		File Folder	3/14/2002 12:53 PM
FirstCF2Project_Data		File Folder	3/14/2002 11:50 AM
cfxmain	3 KB	C Source File	11/28/2001 9:21 AM
FirstCF2Project	47 KB	CodeWarrior Project	3/14/2002 12:53 PM

When you chose Project:Make, CodeWarrior checked all of the file dependencies, compiled all of the C source files, then linked the C code with the libraries. If it had found any errors, it would have displayed an error window with a list of problems for you to fix by double-clicking on the error message.

If you made no changes to the sample file created by the stationery you should have gotten no errors. If you move into the bin directory under your project you will see the files created by Codewarrior during the Make. If you have been following along with our example all the files will be name first and each will have a different extension. The two most important files in the group are:

- First.RUN – This is a CF2 executable program which runs from CF2 RAM. You will usually load the .RUN during the initial testing of your programs. You can also load .RUN files and save them as executable programs on a CompactFlash card. Programs saved in this way can be run by simply typing their name at the PicoDOS prompt. To save a file to the CompactFlash card load the program the way we did above. When MotoCross presents you with the G for GO, simply add S and the name you want to give the program. When you hit enter, PicoDOS will save the file on the CompactFlash card with a PicoDOS executable extension (.PXE).

- First.APP – This is a CF2 executable program that runs from the CF2's on-board Flash memory. Loading a .APP file will place the program directly into Flash memory. You can run the .APP by typing APP at the PicoDOS prompt. You can also set PicoDOS to automatically load and run the .APP when PicoDOS starts. You do this at the PicoDOS prompt by typing `BOOT APP↵`.

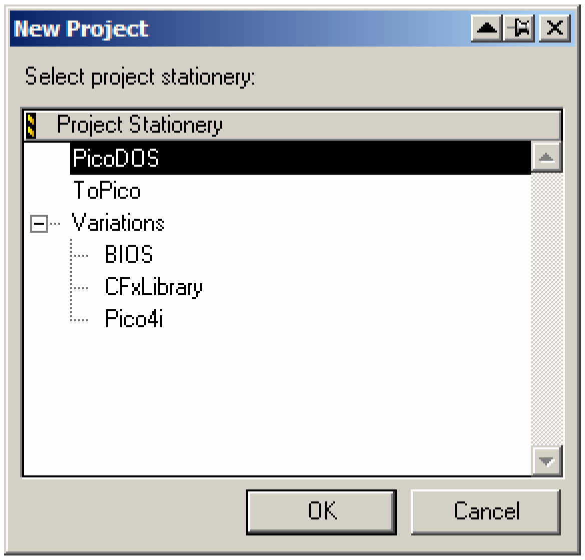
The RHX and AHX files are also applications, but in Motorola S Record (hex) format. The RMP and AMP files contain text listing the functions and global variables, exactly as they are used on the CF2. You can open and read these maps directly from CodeWarrior.

Beyond Standard C Libraries

As you may have read elsewhere, the CF2 supports the ANSI C Standard Library. This fact should put an experienced C programmer well on their way to writing more meaningful and useful programs for the CF2. However, it is unlikely that you bought the CF2 solely to write programs with the C Standard Library. You probably want to take advantage of the CF2's Stationery choices, I/O features, low-power operation and more of its many specialty subsystems.

Stationery

Whenever you are creating a new project you are given several different choices for Stationery. In the tutorial, we used the PicoDOS Stationery. We will now list the different types of Stationery and give a brief description.



PicoDOS – This is a standard PicoDOS executable written using C.

ToPico - This is an application that serves as a foundation for building your custom version of PicoDOS. You can use this Stationery to add new commands to PicoDOS and even remove any that you don't want or need. For example, you could add a command to turn on and initialize your own custom electronics. You could also remove the FORMAT command if you wanted to prevent someone from accidentally formatting a card in the field.

Variations

BIOS – This stationery is for an advanced user that wishes to write BIOS functions.

CfxLibrary - Use this to build a library (.LIB). If you wish to write software that you would like to distribute to others but without having to share the source code itself, you will want to write a library. You would only need to distribute the .LIB file and a header file to the user.

Pico4i - Same as PicoDOS except that it defaults to a 32 bit integer. This is a standard under UNIX and so here it serves as an aid for anyone who is porting code over from the UNIX/LINUX environment.

Subsystems

The CF2 as an embedded controller is very diverse and agile. It can do almost anything. There are many different sections of APIs for accessing each of the internal capabilities. The following is a list of the major subsystems that you will find described in further detail in other pieces of documentation.

ATA Device Drivers - The CF2 API has a suite of functions that allow you to manipulate ATA storage devices (usually CompactFlash) from within your programs. Although most developers will never have a need to use these in light of the standard file routine compatibility, they are nonetheless provided for your programming convenience.

CompactFlash Low Level Drivers - The CompactFlash Low Level drivers cover programming needs specific to CompactFlash card management and use. It is unlikely that you would ever use these but they are provided as an additional abstraction layer between the hardware and the ATA device drivers.

Checksums and Cyclic Redundancy Check Functions - Because many CF2 applications involve the transfer of data between the CF2 and a host computer as well as any other mating systems, there is often a need for error checking and data integrity tests. These functions provide a built in mechanism for performing checksum verification and cyclic redundancy checks.

Chip Select Drivers - One of the more interesting and distinguishing features of the CF2 is the ease with which you can add memory-mapped peripherals. The functions in the API subsection provide relatively high level mechanisms for mapping and configuring the two bus chip select lines that are available for your use. Examples are also provided which make use of these functions for mapping in new I/O.

Time Processing Unit (TPU) - The CF2 uses the Motorola MC68CK332 processor as its main brain. The '332 contains several hardware sub modules that are specifically tailored to the common tasks in embedded computing. One of these sub modules is the Time Processing Unit or TPU. The TPU has 16 channels. One TPU channel is reserved for internal use by the CF2. The other 15 channels are available for your use. Here is a list of TPU Time Functions:

- Input Capture or Input Transition Counter
- Period or Pulse Width Accumulation
- Output Compare
- Pulse Width Modulation
- Software UART

One of the most powerful features of the TPU is the ability for a channel to be configured as half of a UART. A single channel can be an asynchronous receiver or transmitter. This will be one of the most used functions of the TPU. You can add an extra UART to your CF2 application simply by setting aside two TPU channels for the task. Persistor Instruments has a new RecipeCard that provides extra communications drivers and interface jacks to aid in prototyping TPU UART applications (R216AU).

For a more detailed description of the TPU please see the included PDF documentation from Motorola.

Flash Memory Functions - The CF2 has 1MB of non-volatile flash memory built-in. This is used to hold both the BIOS libraries and PicoDOS as well as other system internals. However, there is about 768K free for non-volatile application storage or data storage. Because flash memory has special requirements and does not support random access unlike conventional RAM, we have developed a suite of functions that perform the writing and erasing tasks on the flash as well as other maintenance chores and diagnostic utilities. If you wish to develop programs that will make use of the Flash, you will need to familiarize yourself with this section in the User's Manual.

Interrupt and Exception Vector Wrapper Functions - We recognize that many applications in embedded computing are time sensitive or could be best serviced by a hardware or software interrupt mechanism. These functions provide you with a simple, easy to understand method for creating your own interrupt handlers. We also provide functions that install your new interrupt handlers, written either in C using our prototyping and definition tools or written directly in assembly language, into the Vector Table of the 68332.

LED Signal Functions - The CF2 has two dual color LEDs mounted near the sides of the CompactFlash header. They can be used as simple indicators of program status, as a visual watchdog or whatever else you can think of. This set of simple, almost self explanatory functions control the state and behavior of these LEDs making it easy for your programs to provide the most minimal of visual feedback.

Pin I/O Drivers, Functions and Macros - These functions and macros allow you to control and manipulate the behavior of the CF2's general-purpose I/O lines. Some of the I/O pins on the CF2 have alternate functions related to other subsystems. This section contains functions that allow you to manipulate these alternate functions and correctly manage the use of the I/O pins on connector C on the CF2. Furthermore, there is a set of macros that provide ultra high-speed access allowing you to perform basic pin operations in under a microsecond with certain restrictions.

Periodic Interrupt Timer - Another hardware feature of the Motorola MC68CK332 is the Periodic Interrupt Timer or PIT. The PIT allows the '332 to trigger an interrupt on a set period. The period can be adjusted in increments of 100 microseconds from 100μs to 25.5ms and in increments of approximately 51ms from 51ms to around 13s. The CF2 API also provides a chore management system for the PIT allowing you to perform more than one chore, written in standard C with no special considerations (other than speed), on each interrupt and to shield the programmer from the necessity of writing low-level interrupt handlers.

PicoDOS Library Functions - Providing that PicoDOS is resident in flash in your particular application, which is normally the case, you can access most of the core functionality of PicoDOS from within your applications. Also provided is a subsystem called the CMD processor that provides a deep and valuable framework for setting up a command line interface to your program. The CMD Processor allows your program to accept commands from a user interactively over the serial port, parse their arguments and dispatch the appropriate function. This can be a huge time saver if you want your program to be interactive. For an example of using the CMD processor, create a new project with CodeWarrior and select the ToPico stationery.

Power Management Drivers - One of the more crucial features of the CF2 is its low power consumption with non-executing power-down modes as low as 5 µA. The CF2 really is a low-power panacea for the embedded system designer. This driver section gives you access to all the different power modes and power conservation functions the CF2 has to offer. There are many ways to reduce power in an embedded system. In this API section we have attempted to provide an intuitive management scheme for these many and diverse options.

Queued PicoBUS: Another Persistor Exclusive - The Motorola QSPI bus is a powerful way to add peripherals such as A/D converters and various other sensors to an embedded system. Until now there hasn't been an intuitive, managed, yet performance oriented software layer to control this bus. Our Queued PicoBUS (QPB) has solved this problem. The QPB API allows you to configure and manage your SPI devices. Furthermore, our internal library of device configurations is growing all the time. We may already have done some of the groundwork for the device you want to use.

Real Time Clock – Instead of a separate Real Time Clock, the CF2 uses a Texas Instruments MSP430 microcontroller to manage time and control power. When the CF2 is awake it uses a 1PPS signal from the MSP430 to maintain time in the 68332 with a custom TPU function. This allows the CF2 to quickly respond to time requests. A suite of API functions gives you access to setting and reading the system clock.

Serial Controller Interface Driver - The serial controller interface (SCI) is the main line of communication with the outside world during development. It consists of a low-level driver for the onboard UART and line drivers. Once again all of the ANSI C functions are built on top of this driver but if power conservation is your game you may need to use some of these low-level calls to configure the UART in a specific way to help meet your design needs. We have attempted to cover all of the bases in this driver, from normal operation to the lowest power modes.

System Clock and Wait State Management - One of the easiest ways to conserve power in an embedded controller is to turn down the speed. Power consumption in a processor varies almost linearly with speed, and we provide functions for you to adjust the system clock anywhere from 160KHz to 16MHz and beyond. Right there you can reduce your power consumption by an order of magnitude. Furthermore, these functions provide a great level of control regarding the wait states and access speeds of the various bus peripherals. If you are interested in measuring and characterizing the performance of your CF2, you may wish to see our example program called **Hurry Up and Wait.** This program provides a great example of how to change the system clock and the number of wait states used with each peripheral.

Utility Functions - Lastly, there are functions that don't seem to fit anywhere else. These functions include a hexdump function that can be extremely useful during development and other useful, but difficult to classify functions.

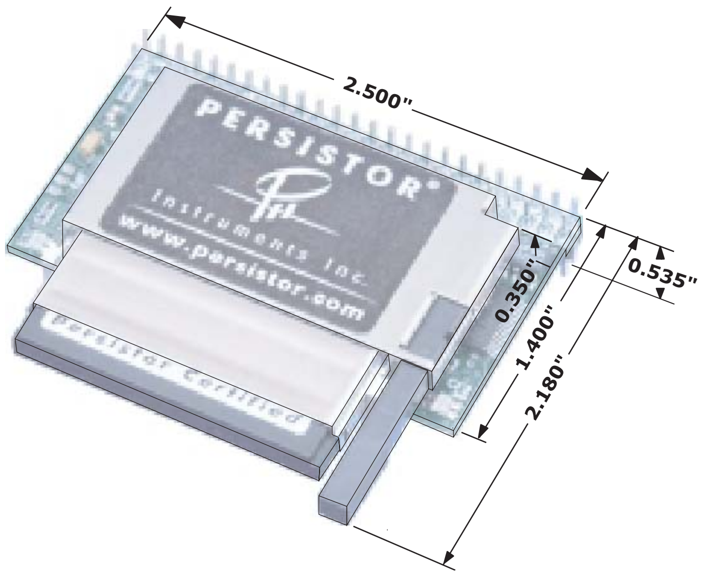
Specifications

Absolute Maximum Ratings

VBAT to GND:	±20V
VREG and VBBK to GND:	+3.9V
Digital Signals to GND:	-0.3V to VREG+0.3V
Operation Temperature:	-40°C to +85°C

Except for the LEDs, whose operation temperature is –25°C to +85°C

Storage Temperature:	-40°C to +85°C
Humidity:	0 to 95% (non-cond)



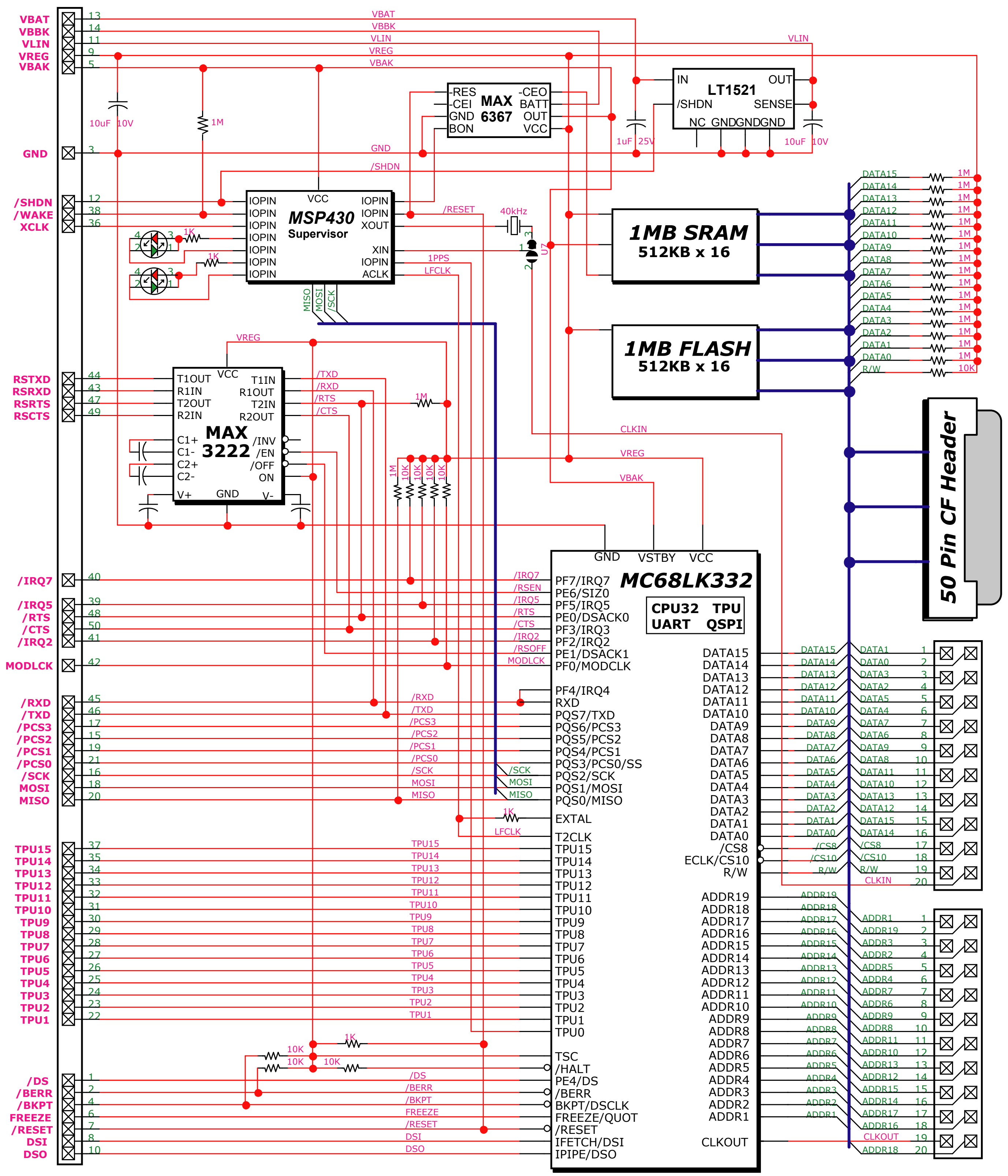
Physical Specifications

PARAMETER	SI	US
Weight		< 1.0 OZ
With CF Card		< 1.2 OZ
Length	63.5 mm	2.5 in
Width	35.5 mm	1.4 in
With CF Card	51.0 mm	2.0 in
Thickness	17.8 mm	0.70 in

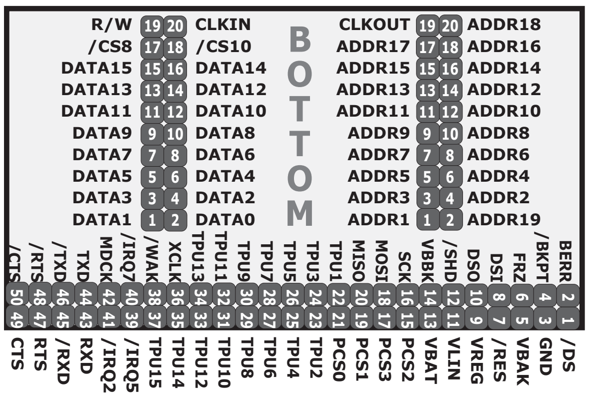
Electrical Characteristics

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
VBAT Operating Voltage	-25° C to +75º C at 40mA	3.6		20	V
	-25° C to +30º C at 75mA	3.6		20	V
	-25° C to +75º C at 75mA	3.6		12.8	V
	-25° C to +30º C at 250mA	5.0		8.4	V
	-25° C to +75º C at 250mA	5.0		6.0	V
VREG Operating Voltage		3.1	3.3	3.6	V
VREG Operating Current	16MHz constant CompactFlash Write		100	125	mA
	16MHz CompactFlash Idle		55	70	mA
	8MHz CompactFlash Idle		32	40	mA
	4MHz CompactFlash Idle		20	25	mA
	320KHz CompactFlash Idle		4	10	mA
	LPSTOP CompactFlash Idle		300	500	uA
	LPSTOP CompactFlash Idle (no CF card)		240	400	uA
	Suspend		6	20	uA
VBBK Operating Voltage		2.7	3.0	VREG	V
VBBK Operating Current	Backup / Suspend		2.8	20	uA
	VREG > VBBK		0.1	1	uA
System Clock		0.16		16	MHz
System Crystal	+/- 100 ppm @ 25°C	39.996	40.000	40.004	KHz

Block Diagram and Signal Connections



Pin Descriptions



The CF2 interfaces to your circuitry using three standard double-row 0.1" headers. Many CF2 based systems will only need to access the signals on the 2x25 pin "C" connector, which brings out pins on both sides of the board (OEM versions may specify pins on only one side or the other).The 2x10 "A" and "B" connectors bring out the address and data bus along with control signals for system expansion.

The first ten pins of connector "C" form a standard BDM (Background Debugger Mode) connector block at the top of the CF2, and these are identified by a solid white silk-screen. In addition, every fifth pin of connector "C" is marked with white to help quickly identify the proper pin. In the table of connections for connector C, we have BOLDED the connections which are identified by the silk screen.

Pin type designations are taken from the MC68CK332 Technical Summary, and where appropriate, suffixed with the value of onboard pullup resistors or special notes.

Static Sensitive CMOS! All of the CF2 pins connect to static sensitive CMOS circuitry. You must take precautions to guard against damage to these parts from static electricity.

Key to type abbreviations

A	Output signals that are always driven.
Ao	Type A that can operate in open drain mode.
Aw	Type A output with weak pull-up during reset.
B	Three state output that includes circuitry to pull up output before high impedance is established to ensure rapid rise time.
Bo	Type B that can operate in open drain mode.
C	TPU I/O lines are always driven when outputs.
..Pxx	Has an onboard pull up resistor of value xx.
..PxxG	Has an onboard pull down resistor of value xx.
..RSI	RS-232 (EIA) level input.
..RSO	RS-232 (EIA) level output.

A	Signal	Description	Dir.	Func.	Type
1	ADDR1	Address Bus 1	Out	BUS	A
3	ADDR3	Address Bus 3	Out	BUS	A
5	ADDR5	Address Bus 5	Out	BUS	A
7	ADDR7	Address Bus 7	Out	BUS	A
9	ADDR9	Address Bus 9	Out	BUS	A
11	ADDR11	Address Bus 11	Out	BUS	A
13	ADDR13	Address Bus 13	Out	BUS	A
15	ADDR15	Address Bus 15	Out	BUS	A
17	ADDR17	Address Bus 17	Out	BUS	A
19	CLKOUT	System Clock	Out	CLK	A

A	Signal	Description	Dir.	Func.	Type
2	ADDR19	Address Bus 19	Out	BUS	A
4	ADDR2	Address Bus 2	Out	BUS	A
6	ADDR4	Address Bus 4	Out	BUS	A
8	ADDR6	Address Bus 6	Out	BUS	A
10	ADDR8	Address Bus 8	Out	BUS	A
12	ADDR10	Address Bus 10	Out	BUS	A
14	ADDR12	Address Bus 12	Out	BUS	A
16	ADDR14	Address Bus 14	Out	BUS	A
18	ADDR16	Address Bus 16	Out	BUS	A
20	ADDR18	Address Bus 18	Out	BUS	A

B	Signal	Description	Dir.	Func.	Type
1	DATA 1	Data Bus 1	I/O	BUS	AwP1M
3	DATA 3	Data Bus 3	I/O	BUS	AwP1M
5	DATA 5	Data Bus 5	I/O	BUS	AwP1M
7	DATA 7	Data Bus 7	I/O	BUS	AwP1M
9	DATA 9	Data Bus 9	I/O	BUS	AwP1M
11	DATA 11	Data Bus 11	I/O	BUS	AwP1M
13	DATA 13	Data Bus 13	I/O	BUS	AwP1M
15	DATA 15	Data Bus 15	I/O	BUS	BP10K
17	/CS8	Chip Select 8	Out	BUS	A
19	R/W	Read / Write	Out	BUS	AP10K

B	Signal	Description	Dir.	Func.	Type
2	DATA0	Data Bus 0	I/O	BUS	AwP1M
4	DATA2	Data Bus 2	I/O	BUS	AwP1M
6	DATA4	Data Bus 4	I/O	BUS	Aw *1
8	DATA6	Data Bus 6	I/O	BUS	AwP1M
10	DATA8	Data Bus 8	I/O	BUS	AwP1M
12	DATA10	Data Bus 10	I/O	BUS	AwP1M
14	DATA12	Data Bus 12	I/O	BUS	AwP1M
16	DATA14	Data Bus 14	I/O	BUS	AwP1M
18	/CS10	Chip Select 10	Out	BUS	AwP1M
20	CLKIN	40Khz Clock	In	CLK	*2

C	Signal	Description	Dir.	Func.	Type
1	/DS	Data Strobe	OUT	BDM/BUS	B *8
3	GND	Ground	PWR	BDM/PWR	
5	VBAK	VREG | VBBK	PWR	PWR	*10
7	/RESET	Reset	I/0	BDM/BUS	BoP1K
9	VREG	3.3 V Power	IN	BDM/PWR	
11	VLIN	3.3v Regulator	OUT	PWR	
13	VBAT	Main Battery	IN	PWR	
15	PCS2	SPI Chip Sel 2	I/O	QSPI/GPIO	Bo*9
17	PCS3	SPI Chip Sel 3	I/O	QSPI/GPIO	Bo*9
19	PCS1	SPI Chip Sel 1	I/O	QSPI/GPIO	Bo*9
21	PCS0	SPI Chip Sel 0	I/O	QSPI/GPIO	Bo*9
23	TPU2	TPU Chan 2	I/O	TPU/GPIO	C
25	TPU4	TPU Chan 4	I/O	TPU/GPIO	C
27	TPU6	TPU Chan 6	I/O	TPU/GPIO	C
29	TPU8	TPU Chan 8	I/O	TPU/GPIO	C
31	TPU10	TPU Chan 10	I/O	TPU/GPIO	C
33	TPU12	TPU Chan 12	I/O	TPU/GPIO	C
35	TPU14	TPU Chan 14	I/O	TPU/GPIO	C
37	TPU15	TPU Chan 15	I/O	TPU/GPIO	C
39	/IRQ5	Int Req 5	I/O	IRQ/GPIO	B10K *5
41	/IRQ2	Int Req 2	I/O	IRQ/GPIO	B10K *5
43	RSRXD	UART RX EIA	IN	UART	RSIP5KG
45	/RXD	UART Rx CMOS	IN	UART	*7
47	RSRTS	UART RTS EIA	OUT	UART	RSO
49	RSCTS	UART CTS EIA	IN	UART	RSIP5KG

C	Signal	Description	Dir.	Func.	Type
2	/BERR	Bus Error	OUT	BDM/BUS	BP10K
4	/BKPT	Break Point	IN	BDM	P10K
6	FREEZE	Freeze	OUT	BDM	A
8	DSI	BDM Input	IN	BDM	A
10	DSO	BDM Output	OUT	BDM	A
12	/SHDN	Shutdown	OUT	PWR	*3
14	VBBK	Backup Battery	IN	PWR	
16	SCK	SPI Clock	I/0	QSPI/GPIO	Bo *9
18	MOSI	SPI Data Out	I/O	QSPI/GPIO	Bo *9
20	MISO	SPI Data In	I/O	QSPI/GPIO	BoP1M
22	TPU1	TPU Chan 1	I/O	TPU/GPIO	C
24	TPU3	TPU Chan 3	I/O	TPU/GPIO	C
26	TPU5	TPU Chan 5	I/O	TPU/GPIO	C
28	TPU7	TPU Chan 7	I/O	TPU/GPIO	C
30	TPU9	TPU Chan 9	I/O	TPU/GPIO	C
32	TPU11	TPU Chan 11	I/O	TPU/GPIO	C
34	TPU13	TPU Chan 13	I/O	TPU/GPIO	C
36	XCLK	Don't Connect			
38	/WAKE	Ext. Wakeup	IN	PWR	*4
40	/IRQ7	Int Req 7	I/O	IRQ/GPIO	B10K *5
42	MODCLK	Alternate Clock	I/O	CLK/GPIO	B10K *6
44	RSTXD	UART TX EIA	OUT	UART	RSO
46	/TXD	UART TX CMOS	OUT	UART	Bo
48	/RTS	UART RTS CMOS	OUT	UART	AP1M
50	IRQ3/CTS	UART CTS CMOS	IN	UART	B *7

***1**. DATA4 is pulled low with 1K during reset.

***2**. CLKIN is optional and not connected. Contact the factory for information on using CLKIN with an external precision clock source.

***3**. /SHDN is an output signal controlled by the power management circuitry. Your peripheral circuitry can monitor this signal, but only with inputs having less than 2uA leakage current. When low, all of the CF2 circuitry is powered off. Do not attempt to assert or load this line.

***4**. /WAKE is input to the power management circuitry and is pull high with 1M to the internal VBAK voltage. External circuitry (coordinated with CF2 driver software) may use this line to pull the CF2 out of suspend mode.

***5**. These lines must be left floating, or asserted high at reset.

***6**. This line must be floating or asserted high at reset for normal operation. It may be pulled low during reset to disable the onboard PLL clock oscillator and insert an external clock. Contact the factory for application notes.

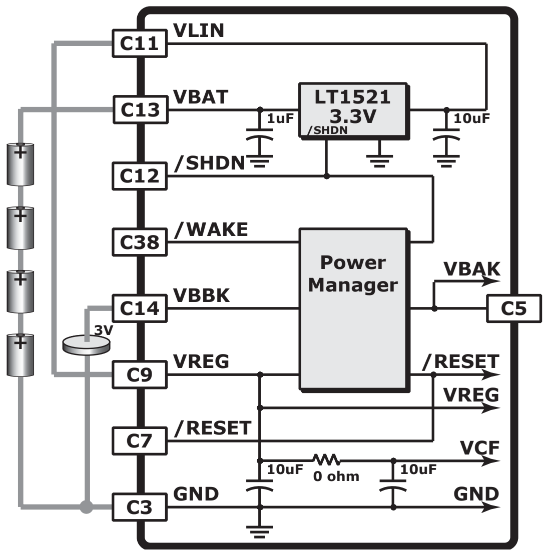
***7**. /RXD and /CTS are inputs to the 68332, but are normally driven by the RS232 driver chip. You can disable the RS232 driver under software control to allow the CF2 to run the UART at CMOS levels.

***8**. DS comes out of reset as an active driving bus output, but is not used by the CF2 except when connected to a BDM debugger. When your program gets control, you can define this line as a BUS signal (for decoding), an input signal (though you must not drive into it until you redefine it), or an output signal. The convenient location and the fact that it is less useful as a general purpose control line (since it flails at reset) makes this an ideal pin for diagnostics and timing or profiling your functions with a scope using the fast I/O macros.

***9**. All of the QSPI (PicoBUS) signals revert to inputs at reset which means they may assume any state, and that could be trouble for attached SPI devices which could interpret reset flailing as commands, which in turn could have the SPI device do something the CF2 would not like. You should add 10K to 100K pull ups to the /CS lines of your SPI devices to prevent trouble.

***10**. VBAK is provided to supply power to external circuitry in shutdown or off modes. On the older CF1 this pin was PASS, a no connect, which may have been used to pass a signal from a top mounted expansion board to another bottom mounted board.

Power Connections



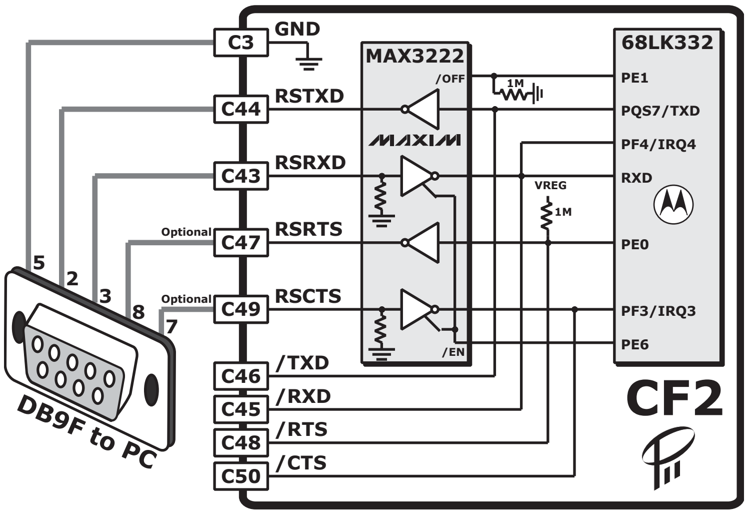
The CF2 has very flexible power management support circuitry to simplify integration with your electronics. The onboard 3.3 volt linear regulator with reverse battery protection and thermal current limiting that "floats" on the CF2 board to allow you to bypass, augment, or reassign it to other circuitry. Normally, you simply wire your positive supply to the VBAT input, then jumper VLIN (the regulator output) to VREG.

An onboard power supervisor ensures that the CF2 will receive an orderly reset and that the memory and real-time clock will be preserved in the event of a power disruption.

A separate power managing controller lets the CF2 drop into a SUSPEND mode where the system draws less than 10 microamps. It can wake and resume after a programmed delay or on detection of /WAKE signal or CompactFlash cardchange event.

C	Signal	IN	Description
3	GND	IN	This is the negative return for all the CF2 power signals and digital ground for all the CF2 logic.
13	VBAT	IN	This is the unregulated DC input voltage to the onboard LT1521 linear regulator. The LT1521 will allow the CF2 to work with inputs from 3.6 volts to 20 volts and provides reverse power protection to -20 volts. This input connects to a 1uF filter capacitor. Leave this input unconnected if you are not using the onboard regulator.
11	VLIN	OUT	This is the regulated 3.3 volts from the onboard LT1521 linear regulator. The LT1521 has a rated output of 300mA maximum, but the actual continuous current will vary with temperature and input voltage. The regulated output does not connect directly to any onboard components and is normally wired to VREG through an off board connection. This output connects to a 10uF tantalum filter capacitor.
9	VREG	IN	This is the regulated 3.3 volts to the CF2 electronics. It is normally wired to VLIN, but may be driven by any 3.3 volt supply. VREG connects to an on board 10uF tantalum filter capacitor. It also connects to the MAX6367 power supervisor which resets the CPU and switches over to VBBK when the input drops below (2.85 to) 3 volts.
	VCF		This on board only power signal feeds the CompactFlash header through a 0Ω shunt. It also has its own 10uF tantalum filter capacitor.
14	VBBK	IN	This is the 3 volt backup battery supply input. It feeds into the MAX6367 power supervisor which then supplies it to the MSP430 coprocessor and the onboard SRAM when the main voltage drops below 3 volts.
5	VBAK	OUT	This onboard only power signal comes from the MAX6367 power supervisor and is the input to Vcc of the MSP430 and SRAM. The voltage MAX6367 feeds this with VREG if VREG is over 3 volts or VBBK if VREG is less than 3 volts.
7	/RESET	I/O	This open drain bidirectional signal resets the CF2 electronics. This is pulled to VREG with a 1K resistor. Both the MAX6367 power supervisor and the MSP430 coprocessor can assert this signal.
12	/SHDN	OUT	This is an output signal controlled by the power management circuitry. Your peripheral circuitry can monitor this signal, but only with inputs having less than 100nA leakage and less than 100pF capacitance. When low, all of the CF1 circuitry is powered off. DO NOT attempt to assert or load this line.
38	/WAKE	IN	This is the input to the power management circuitry and is pulled high with 1M to the internal VBAK voltage. External circuitry (coordinated with CF2 driver software) may use this line to pull the CF2 out of suspend mode.

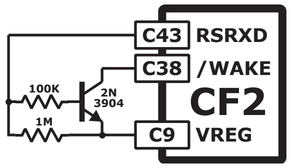
UART Connections



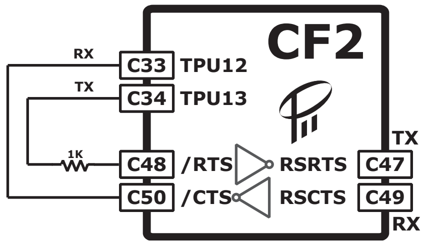
The 68332 has a Serial Controller Interface (SCI) that provides standard UART functions at rates from 64 to 500 kbaud with advanced error detection circuitry. The SCI supports full or half duplex operation, double buffering, optional parity generation and detection, and wakeup on idle line or address detection.

The CF2 has an onboard MAX3222 dual EIA driver to interface directly to any standard RS-232 terminal or device. Either or both the receive and transmit drivers can be disabled under software control to conserve power or allow connection to alternate RS-485 or RS-422 drivers.

BIOS UART Support: The CF2 BIOS software handles port configuration, polled or buffered receive and transmit, and flow control with several dozen high-level C functions. One of the example projects that installed with the Persistor CD demonstrates how to stream incoming RS-232 data to a file on the CompactFlash card at any standard BAUD rate while automatically dropping to less than 2mA whenever the serial line idles. This particular example is 180 lines of C code, over 100 of which are just comments and formatting.



RS-232 Wakeup Call: Both the RXD and CTS signals connect to interrupt request pins on the 68332 to allow wakeup from deep sleep modes on any UART activity. To wake from the 10uA SUSPEND mode, add the simple circuit at right.



2nd RS-232 Level TPU UART borrowing spare EIA drivers: All of the CMOS level UART signals are brought out to allow driver replacement, and the RTS/CTS signals can be used for EIA level flow control or reassigned to work as RXD and TXD signals for a second UART built from a pair of TPU channels running the TPU UART functions.

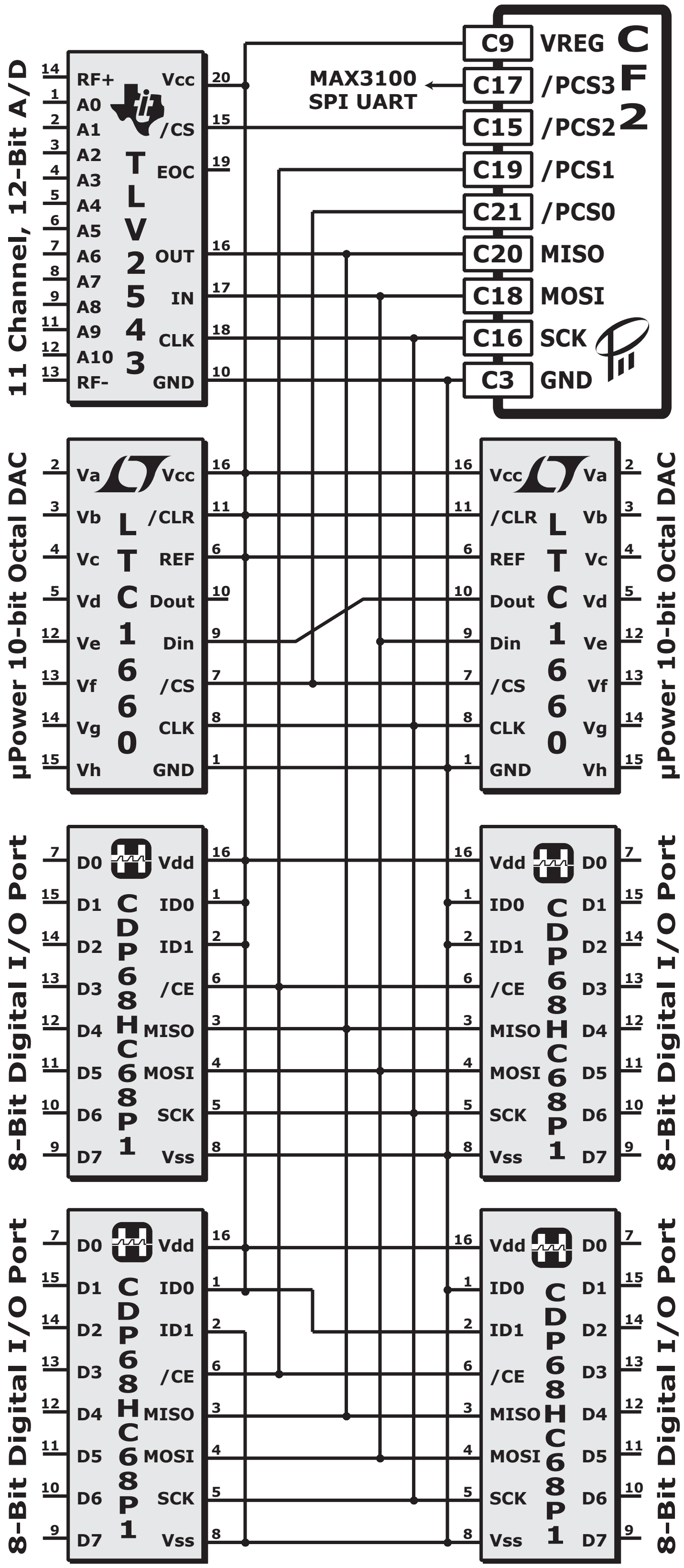
Programming Note!

For this circuit to work, you must add the following line of code to your program to force the CF2 to stop asserting the TXX/RTS signal:

```
PIORead(48);  // tri-state TXX drive signal for external control
```

QSPI Connections

The 68332 has a powerful Queued Serial Peripheral Interface that allows simple hardware expansion to over five hundreds different SPI compatible devices. The circuit at right demonstrates glueless expansion adding yet another UART (using a MAX3100 which is not shown), 11 A-D channels with 12-bit resolution, 16 voltage output 10-bit D-A channels, and 32 individually programmable bi-directional digital I/O pins.



The table below lists some of the manufacturers of SPI compatible devices and their product offerings. This is by no means complete, but it does give you an idea of the expansion possibilities just working the serial bus.

SPI Devices

	A-D 8/10 bit	A-D 12/14 bit	A-D 16+ bit	D-A 8/10 bit	D-A 12/14 bit	D-A 16+ bit	Analog MUX	EEPROM	Driver	RTC	PLL	MISC
Allegro MicroSystems									●			
American Microsystems									●			
Analog Devices	●	●	●	●	●	●	●					
Atmel								●				
Burr-Brown		●	●		●	●						
Catalyst								●				
Dallas Semiconductor										●		
Exar		●		●	●							
Exel Microelectronics										●		
Fujitsu Microelectronics											●	
Harris Semiconductor	●	●	●						●	●		●
Linear Technology	●	●	●		●							
Maxim	●	●	●	●	●	●	●		●			
Micrel Semiconductor									●			
Microchip Technology								●				
Micro Linear	●	●		●								
Motorola	●	●		●					●	●	●	
National Semiconductor	●	●		●					●	●	●	
Ramtron								●				
Signal Processing Tech.	●											
SGS-Thomson								●				
Siemens									●		●	
Sipex				●	●							
Texas Instruments	●	●		●								
Xicor								●				

TPU – Time Processor Unit

The TPU is actually a separate RISC controller integrated into the 68332 with timer optimized hardware that runs an assortment of microcoded, time-sliced, state machines functions. Each of the 16 TPU channels is associated with an I/O pin and each can independently run separate functions and even link to each other under program control. Fifteen of the TPU channels are available for your applications and one is reserved for system use by PicoDOS. Below are Motorola's descriptions for the TPU functions included by default in the CF2. If these are not quite to your liking, you can choose from over two-dozen prewritten functions or even write your own TPU microcode.

Discrete Input/Output TPU Function (DIO)

The discrete input/output (DIO) function allows the user to configure a time processor unit (TPU) channel as an input or output. As an input, the channel can be read at any time or sampled at a periodic rate. As an output, the channel can be driven high or low upon command by the CPU.

Frequency Measurement TPU Function (FQM)

The FQM function counts the number of pulses that are presented to a channel pin within a user specified time window as a 16-bit number. Either rising or falling edges can be used as the beginning of a pulse. In single shot mode, pulses are accumulated for a single window time. In continuous mode, pulses are automatically accumulated in repetitive windows.

New Input Capture/Input Transition Counter TPU Function (NITC)

The NITC function can detect rising and/or falling input transitions. When a transition is detected, the current TCR timer value or a parameter RAM value is captured. The channel continues to detect and count input transitions until it has counted the maximum programmable number stored in the parameter MAX_COUNT. The NITC function can count the programmed maximum number of transitions continually, or it can count the programmed number of transitions once, then cease channel activity until reinitialized.

Output Compare TPU Function (OC)

The output compare (OC) function can generate a single output transition, a single pulse, or a continuous 50% duty cycle pulse train upon receiving a link from another channel. The first two actions require the CPU to initiate each output edge or pulse. The third action generates a continuous square wave without CPU intervention.

Programmable Time Accumulator TPU Function (PTA)

The programmable time accumulator (PTA) function measures either period, high time or low time of an input signal over a programmable number of periods. The number of periods over which the 32-bit measurement is made is selectable over the range 1 to 255.

Pulse Width Modulation TPU Function (PWM)

This output function generates a pulse-width-modulated waveform in which the period and/or the high time can be changed at any time by the CPU. PWM uses two modes of operation: level and normal. In level mode, a 0% or a 100% duty-cycle waveform can be generated. In normal mode, waveforms with duty-cycles between 0% and 100% can be generated.

Queued Output Match TPU Function (QOM)

The QOM function generates complex pulse trains without CPU intervention using a sequence of output matches. An output match causes a programmable pin response when a user-defined value is matched by the value of a free-running counter. QOM generates multiple output matches using a queue of offset times and pin responses in parameter RAM.

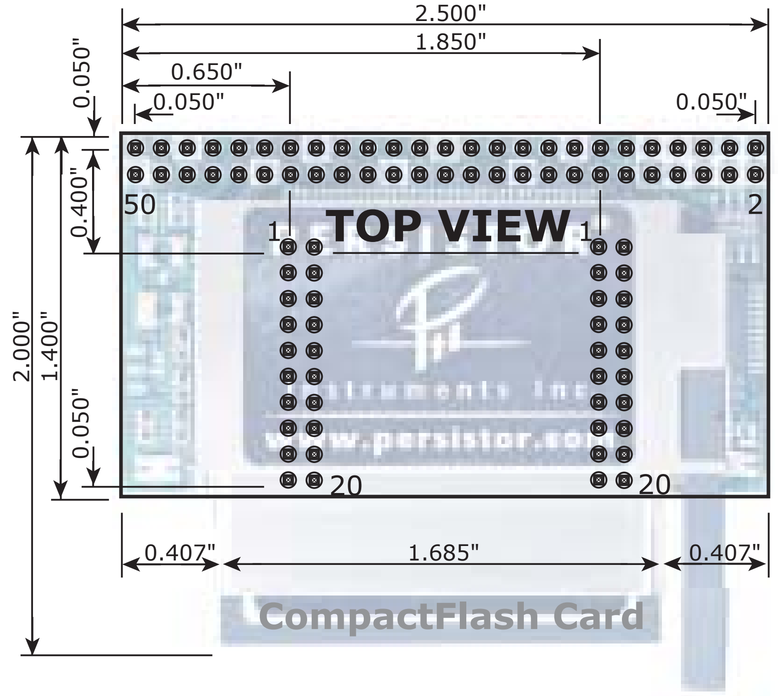
Serial Input/Output Port (SIOP)

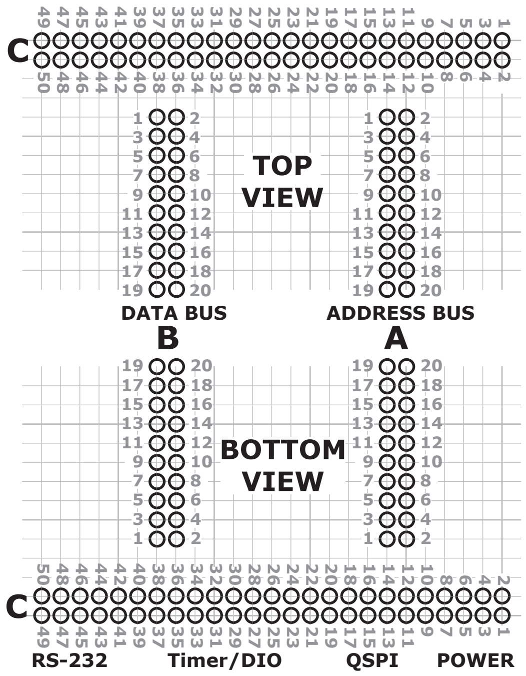
This function uses two or three TPU channels to form a uni or bi-directional synchronous serial port that can be used to communicate with a wide variety of devices. Features such as baud rate and transfer size are user programmable.

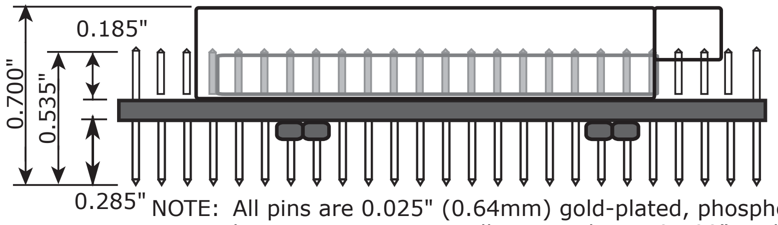
Asynchronous Serial Interface TPU Function (UART)

The UART function uses two TPU channels to provide a 3-wire (RxD, TxD and GND) asynchronous serial interface. All standard baud rates and parity checking can be selected. The CPU interface to UART consists of a command register, which defines the operation (number of data bits, baud rate, parity); a status register, which gives information about the data register (empty or full) and errors (framing and parity); and a data register, which holds the data to be transmitted or data that has been received.

CF2 Dimensions

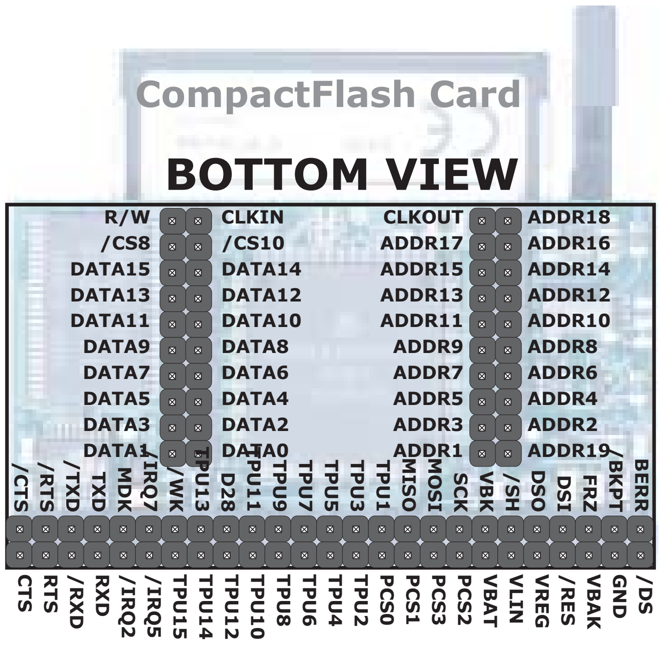






NOTE: All pins are 0.025" (0.64mm) gold-plated, phosphor-bronze square posts, all centered on a 0.100" grid.

PCB Pad Placement



Common CF2 Problems and their Solutions

Problem	Solution
I programmed my application in Flash but now I want to load a new version of it and I cannot. I know I need to get to PicoDOS but I don't know how. What do I do?	Buried within the CF2 is software that sits below PicoDOS. This is the Persistor Boot Monitor or PBM. You can get to PBM when you apply power to the CF2. This can be accomplished by grounding pin 39 (or depressing the PBM button on most RecipeCards) before applying power. Once power is applied you will jump to the PBM. A 60 second countdown will start. You can ignore the countdown for now. Keep pin 39 grounded and type WREN at the PBM prompt and hit Enter. You will then be asked by the PBM to release pin 39 and confirm. Disconnect the ground on pin 39 and type 'Y' and Enter. Now that PBM writes are enabled (WREN), you can ask the PBM to boot to PicoDOS. Type BOOT and hit Enter. You will be given a list of boot addresses with numbers. Type '9' and hit enter. This instructs the PBM to boot PicoDOS at power-up and NOT the application in Flash. The next time you apply power you should see the PicoDOS prompt.
I don't want to load my program into Flash. I would like to have it available to run as a file on the Compact Flash card kind of like DOS program used to run. Can I do this with PicoDOS?	Yes. Sometimes you may have a need for keeping several CF2 applications stored on a Compact Flash card. You would then be able to run them by just typing their names. Load a .RUN file version of your application using Motocross. When the load is complete you will see a 'G' appear on the screen Now add an 'S' to the 'G' so it forms 'GS' and type one space and enter an 8-character filename. Hit Enter when done. If you type 'DIR' and Enter you will get a directory listing. In the list you should see the name you typed with an extension of .PXE (for PicoDOS eXEcutable). To run your application, just type the filename without the extension and hit Enter.
I loaded my program into Flash. How do I make it run whenever I turn on my CF2?	At the PicoDOS prompt type BOOT APP. This instructs the CF2 to run your application out of Flash whenever power is applied.
I loaded my program into Flash. How can I tell PicoDOS to run the program in Flash?	If you want to just run a program in Flash from PicoDOS, all you have to do is type APP at the PicoDOS prompt
Should I format my Compact Flash on my PC or should I format using PicoDOS?	Because of some nuances with formatting on PCs today, we recommend formatting Compact Flash cards using PicoDOS.
When I apply power to my CF2, I can't get a PicoDOS prompt to appear. How do I fix this?	This could be one of several different problems. Make sure that you are running a communications program (we will assume Motocross) and that the CF2 is connected to an available COM port on the PC. Make sure that Motocross is using the correct COM port (the one the CF2 is connected to). Make sure that the baud rates are the same for the CF2 and Motocross (CF2 default baud rate is 9600 8N1). Check to see that power is getting to the CF2. You may have a program running in the CF2 that prevents PicoDOS from running. The solution to this is explained in an earlier problem/solution above.
What voltage should I use for CF2 development?	Keeping the voltage low whenever you are developing is a good way to help minimize the damage that can occur if you were to momentarily short a pin while you probe the board. We develop with a current limited supply. We keep the voltage at 4 volts and the current limited to about 100-150 mA.