21-S3-C825A/P825A-032002

USER'S MANUAL

S3C825A/P825A
8-Bit CMOS
Microcontroller
Revision 1

SAMSUNG ELECTRONICS

S3C825A/P825A

8-BIT CMOS
MICROCONTROLLERS
USER'S MANUAL

Revision 1

SAMSUNG ELECTRONICS

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product.

S3C825A/P825A 8-Bit CMOS Microcontrollers
User's Manual, Revision 1
Publication Number: 21-S3-C825A/P825A-032002

Samsung Electronics' microcontroller business has been awarded full ISO-14001 certification (BSI Certificate No. FM24653). All semiconductor products are designed and manufactured in accordance with the highest quality standards and objectives.

Samsung Electronics Co., Ltd.
San #24 Nongseo-Ri, Giheung-Eup
Yongin-City, Gyeonggi-Do, Korea
C.P.O. Box #37, Suwon 440-900

TEL: (82)-(31)-209-1934
FAX: (82)-(31)-209-1899
Home Page: http://www.samsungsemi.com

Printed in the Republic of Korea

Preface

The S3C825A/P825A *Microcontroller User's Manual* is designed for application designers and programmers who are using the S3C825A/P825A microcontroller for application development. It is organized in two main parts:

Part I Programming Model

Part II Hardware Descriptions

Part I contains software-related information to familiarize you with the microcontroller's architecture, programming model, instruction set, and interrupt structure. It has six chapters:

Chapter 1	Product Overview	Chapter 4	Control Registers
Chapter 2	Address Spaces	Chapter 5	Interrupt Structure
Chapter 3	Addressing Modes	Chapter 6	Instruction Set

Chapter 1, "Product Overview," is a high-level introduction to S3C825A/P825A with general product descriptions, as well as detailed information about individual pin characteristics and pin circuit types.

Chapter 2, "Address Spaces," describes program and data memory spaces, the internal register file, and register addressing. Chapter 2 also describes working register addressing, as well as system stack and user-defined stack operations.

Chapter 3, "Addressing Modes," contains detailed descriptions of the addressing modes that are supported by the S3C8-series CPU.

Chapter 4, "Control Registers," contains overview tables for all mapped system and peripheral control register values, as well as detailed one-page descriptions in a standardized format. You can use these easy-to-read, alphabetically organized, register descriptions as a quick-reference source when writing programs.

Chapter 5, "Interrupt Structure," describes the S3C825A/P825A interrupt structure in detail and further prepares you for additional information presented in the individual hardware module descriptions in Part II.

Chapter 6, "Instruction Set," describes the features and conventions of the instruction set used for all S3C8-series microcontrollers. Several summary tables are presented for orientation and reference. Detailed descriptions of each instruction are presented in a standard format. Each instruction description includes one or more practical examples of how to use the instruction when writing an application program.

A basic familiarity with the information in Part I will help you to understand the hardware module descriptions in Part II. If you are not yet familiar with the S3C8-series microcontroller family and are reading this manual for the first time, we recommend that you first read Chapters 1–3 carefully. Then, briefly look over the detailed information in Chapters 4, 5, and 6. Later, you can reference the information in Part I as necessary.

Part II "hardware Descriptions," has detailed information about specific hardware components of the S3C825A/P825A microcontroller. Also included in Part II are electrical, mechanical, OTP, and development tools data. It has 15 chapters:

Chapter 7	Clock Circuit	Chapter 15	LCD Controller/Driver
Chapter 8	RESET and Power-Down	Chapter 16	10-bit Analog-to-Digital Converter
Chapter 9	I/O Ports	Chapter 17	Serial I/O Interface
Chapter 10	Basic Timer and Timer 0	Chapter 18	UART
Chapter 11	Timer 1	Chapter 19	Electrical Data
Chapter 12	8-bit Timer 2	Chapter 20	Mechanical Data
Chapter 13	16-bit Timer 3	Chapter 21	S3P825A OTP
Chapter 14	Watch Timer	Chapter 22	Development Tools

Two order forms are included at the back of this manual to facilitate customer order for S3C825A/P825A microcontrollers: the Mask ROM Order Form, and the Mask Option Selection Form. You can photocopy these forms, fill them out, and then forward them to your local Samsung Sales Representative.

Table of Contents

Part I — Programming Model

Chapter 1 Product Overview

Chapter 2 Address Spaces

Chapter 3 Addressing Modes

Table of Contents (Continued)

Chapter 4 Control Registers

Chapter 5 Interrupt Structure

Chapter 6 Instruction Set

Table of Contents (Continued)

Part II Hardware Descriptions

Chapter 7 Clock Circuit

Chapter 8 RESET and Power-Down

Chapter 9 I/O Ports

Chapter 10 Basic Timer and Timer 0

Table of Contents (Continued)

Chapter 11 Timer 1

Chapter 12 8-bit Timer 2

Chapter 13 16-bit Timer 3

Chapter 14 Watch Timer

Chapter 15 LCD Controller/Driver

Table of Contents (Continued)

Chapter 16 10-bit Analog-to-Digital Converter

Chapter 17 Serial I/O Interface

Chapter 18 UART

Table of Contents (Concluded)

List of Figures

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Concluded)

List of Tables

List of Tables (Continued)

List of Programming Tips

List of Register Descriptions

List of Instruction Descriptions

List of Instruction Descriptions (Continued)

1 PRODUCT OVERVIEW

S3C8-SERIES MICROCONTROLLERS

Samsung's S3C8 series of 8-bit single-chip CMOS microcontrollers offers a fast and efficient CPU, a wide range of integrated peripherals, and various mask-programmable ROM sizes. Among the major CPU features are:

— Efficient register-oriented architecture
— Selectable CPU clock sources
— Idle and Stop power-down mode release by interrupt
— Built-in basic timer with watchdog function

A sophisticated interrupt structure recognizes up to eight interrupt levels. Each level can have one or more interrupt sources and vectors. Fast interrupt processing (within a minimum of four CPU clocks) can be assigned to specific interrupt levels.

S3C825A MICROCONTROLLER

The S3C825A single-chip microcontroller are fabricated using the highly advanced CMOS process. Its design is based on the powerful SAM88RC CPU core. Stop and idle (power-down) modes were implemented to reduce power consumption.

The S3C825A is a microcontroller with a 48K-byte mask-programmable ROM embedded.
The S3P825A is a microcontroller with a 48K-byte one-time-programmable ROM embedded.

Using the SAM88RC modular design approach, the following peripherals were integrated with the SAM88RC CPU core:

— Large number of programmable I/O ports (Total 67 pins)
— Synchronous SIO module
— Two 8-bit timer/counters
— Two 16-bit timer/counters
— LCD controller/driver
— A/D converter with 4 selectable input pins

OTP

The S3C825A microcontroller is also available in OTP (One Time Programmable) version, S3P825A.
The S3P825A microcontroller has an on-chip 48K-byte one-time-programmable EPROM instead of masked ROM. The S3P825A is comparable to S3C825A, both in function and in pin configuration.

FEATURES

CPU
- SAM88RC CPU core

Memory
- 2064-byte internal register file (including LCD display RAM)
- 48K-byte internal program memory area

Instruction Set
- 78 instructions
- Idle and Stop instructions

67 I/O Pins
- 31 normal I/O pins
- 36 pins sharing with LCD signals

Interrupts
- 8 interrupt levels and 23 internal sources
- Fast interrupt processing feature

8-Bit Basic Timer
- Watchdog timer function
- 4 kinds of clock source

Timer/Counter 0
- Programmable 8-bit internal timer
- External event counter function
- PWM and capture function

Timer/Counter 1
- One 16-bit timer/counter mode
- Two 8-bit timer/counters A/B mode
- External event counter function

Timer/Counter 2
- Programmable 8-bit interval timer
- External event counter function

Timer/Counter 3
- Programmable 16-bit interval timer
- External event counter function
- PWM and capture function

Watch Timer
- Interval Time: 3.19ms, 0.25s, 0.5s, 1.0s at 32.768 kHz
- 0.5/1/2/4 kHz buzzer output selectable

Analog to Digital Converter
- 4-channel analog input
- 10-bit conversion resolution
- 25μs conversion time

Serial I/O Interface
- 8-bit transmit/receive mode
- 8-bit receive mode
- Selectable baud rate or external clock source

UART
- Full-duplex serial I/O interface
- Four programmable operating modes

LCD Controller/Driver
- 28 segments and 8 common terminals
- 3, 4, and 8 common selectable
- Internal resistor circuit for LCD bias

Two Power-Down Modes
- Idle mode: only CPU clock stops
- Stop mode: system clock and CPU clock stop

Oscillation Source
- Crystal, ceramic, or RC for main clock
- Crystal for sub clock (32.768 kHz)

Instruction Execution Time
- 500 ns at fx=8 MHz (minimum, main clock)
- 122μs at fxt=32.768 kHz (sub clock)

Operating Temperature Range
- –25 °C to +85 °C

Operating Voltage Range
- 2.0 V to 5.5 V at 4 MHz (main clock)
- 2.2 V to 5.5 V at 8 MHz (main clock)
- 2.0 V to 5.5 V at 32.768 kHz (sub clock)

Package Type
- 80-pin TQFP-1212, 80-pin QFP-1420

BLOCK DIAGRAM

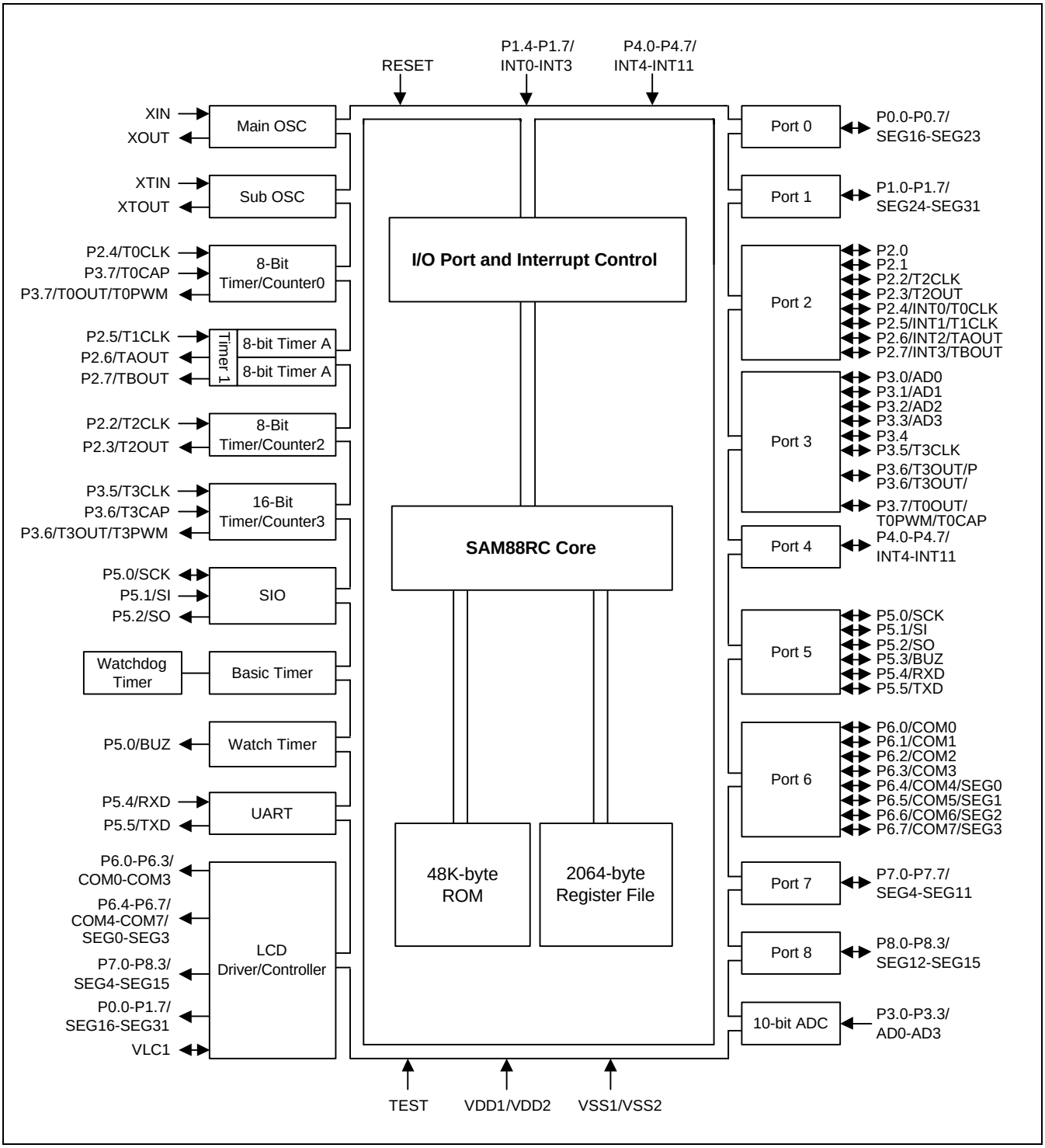


Figure 1-1. Block Diagram

PIN ASSIGNMENT

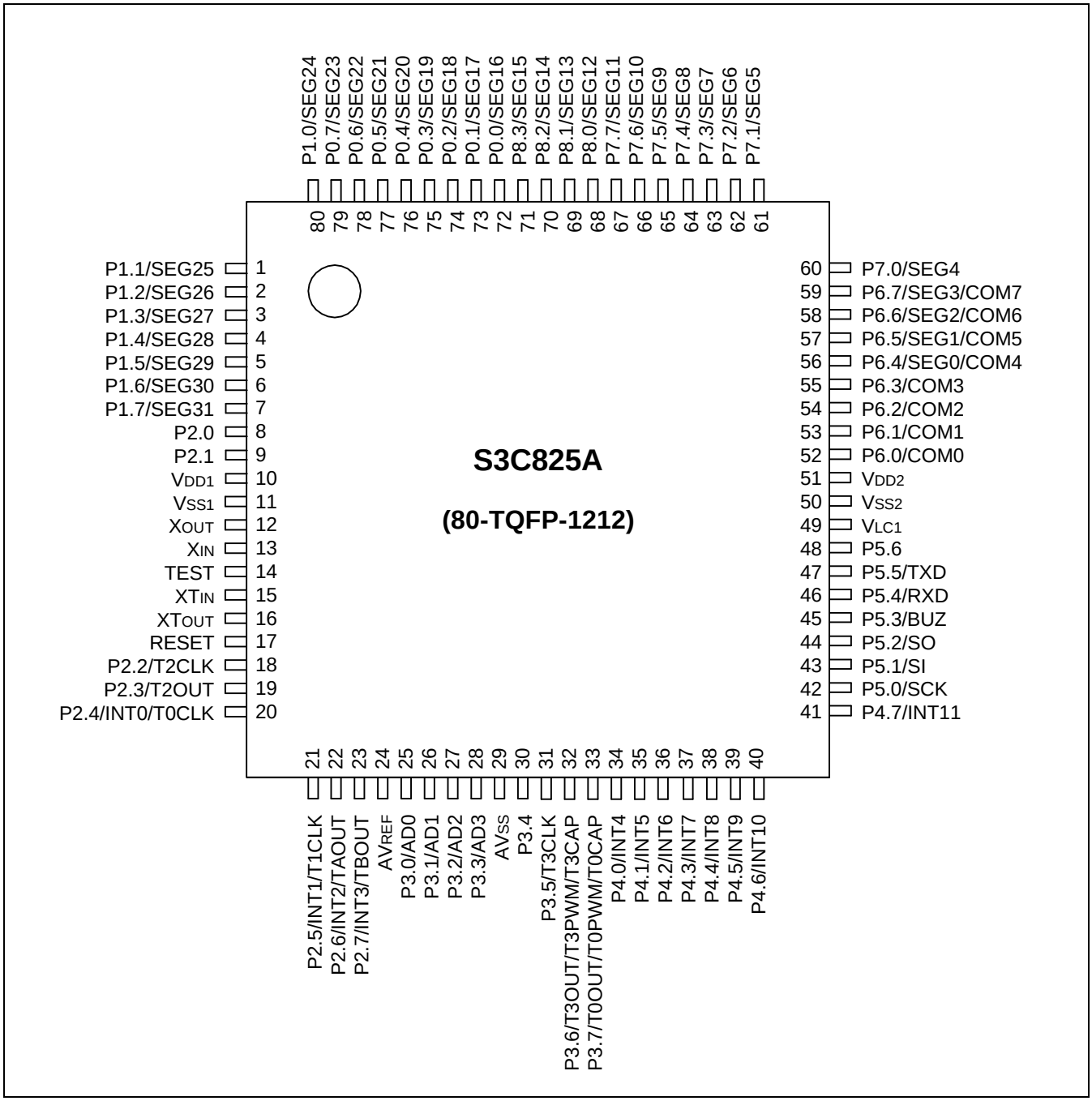


Figure 1-2. S3C825A Pin Assignments (80-TQFP-1212)

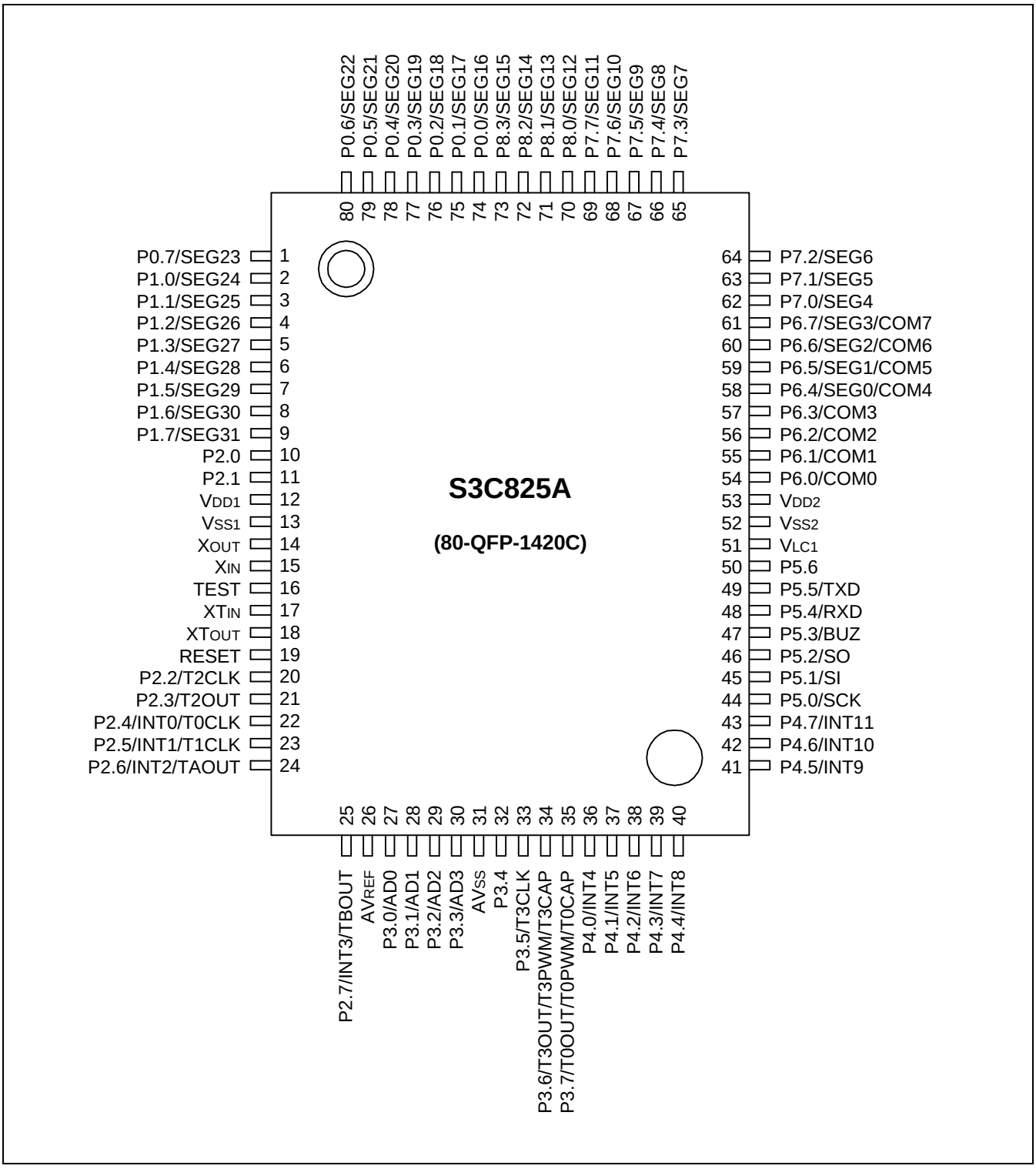


Figure 1-3. S3C825A Pin Assignments (80-QFP-1420)

PIN DESCRIPTIONS

Table 1-1. S3C825A Pin Descriptions

Pin Names	Pin Type	Pin Description	Circuit Type	Pin Numbers	Share Pins
P0.0–P0.7	I/O	4-bit-programmable I/O port. Input or push-pull, open-drain output and software assignable pull-ups.	H-32	72–79 (74-80, 1)	SEG16 – SEG23
P1.0–1.7	I/O	4-bit-programmable I/O port. Input or push-pull, open-drain output and software assignable pull-ups.	H-32	80, 1–7 (2-9)	SEG24 – SEG31
P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7	I/O	1-bit-programmable I/O port. Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups. P2.4–P2.7: Alternately used for external interrupt input (Noise filters, interrupt enable and pending control)	E-4	8 (10) 9 (11) 18 (20) 19 (21) 20 (22) 21 (23) 22 (24) 23 (25)	– – T2CLK T2OUT INT0/T0CLK INT1/T1CLK INT2/TAOUT INT3/TBOUT
P3.0–P3.3	I/O	1-bit-programmable I/O port. Schmitt trigger input or push-pull, open-drain output, and software assignable pull-ups.	F-16	25–28 (27–30)	AD0–AD3
P3.4 P3.5 P3.6 P3.7			E-4	30(32) 31(33) 32(34) 33(35)	– T3CLK T3OUT/T3PWM/T3CAP T0OUT/T0PWM/T0CAP
P4.0–P4.7	I/O	1-bit-programmable I/O port. Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups. P4.0–P4.7: Alternately used for external interrupt input (Noise filters, interrupt enable and pending control)	E-4	34–41 (36–43)	INT4–INT11
P5.0 P5.1 P5.2 P5.3 P5.4 P5.5 P5.6	I/O	1-bit-programmable I/O port. Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups.	E-4	42 (44) 43 (45) 44 (46) 45 (47) 46 (48) 47 (49) 48 (50)	SCK SI SO BUZ RXD TXD –
P6.0-P6.3 P6.4–P6.7	I/O	4-bit programmable I/O port. Input or push-pull, open-drain output and software assignable pull-ups.	H-32	52-55(54-57) 56-59(58-61)	COM0–COM3 COM4–COM7/ SEG0–SEG3
P7.0–P7.7	I/O		H-32	60-67(62-69)	SEG4–SEG11
P8.0–P8.3	I/O		H-32	68-71(70-73)	SEG12–SEG15

NOTE: Parentheses indicate pin number for 80-QFP-1420 package.

Table 1-1. S3C825A Pin Descriptions (Continued)

Pin Names	Pin Type	Pin Description	Circuit Type	Pin Numbers (note)	Share Pins
V_{SS1}, V_{DD1}	–	Power input pins for core block	–	10, 11 (12, 13)	–
X_{OUT}, X_{IN}	–	Main oscillator pins	–	12, 13 (14, 15)	–
TEST	–	Test signal input pin (must be connected to V_{SS})	–	14 (16)	–
XT_{IN}, XT_{OUT}	–	Sub oscillator pins	–	15, 16 (17,18)	–
RESET	I	System reset pin	B	17 (19)	–
INT0–INT3	I/O	External interrupt input pins	E-4	20–23 (22–25)	P2.4–P2.7
T0CLK	I/O	Timer 0 external clock input.	E-4	20 (22)	P2.4
T0OUT	I/O	Timer 0 clock output	E-4	33 (35)	P3.7
T0PWM	I/O	Timer 0 PWM output	E-4	33 (35)	P3.7
T0CAP	I/O	Timer 0 capture input	E-4	33 (35)	P3.7
T1CLK	I/O	Timer 1/A external clock input.	E-4	21 (23)	P2.5
TAOUT	I/O	Timer 1/A clock output	E-4	22 (24)	P2.6
TBOUT	I/O	Timer B clock output	E-4	23 (25)	P2.7
T2CLK	I/O	Timer 2 external clock input.	E-4	18 (20)	P2.2
T2OUT	I/O	Timer 2 clock output	E-4	19 (21)	P2.3
T3CLK	I/O	Timer 3 external clock input.	E-4	31 (33)	P3.5
T3OUT	I/O	Timer 3 clock output	E-4	32 (34)	P3.6
T3PWM	I/O	Timer 3 PWM output	E-4	32 (34)	P3.6
T3CAP	I/O	Timer 3 capture input	E-4	32 (34)	P3.6
AD0–AD3	I/O	Analog input pins for A/D convert module	F-16	25–28 (27–30)	P3.0–P3.3
AV_{REF}, AV_{SS}	–	A/D converter reference voltage and ground	–	24, 29 (26, 31)	–
INT4–INT11	I/O	External interrupt input pins	E-4	34–41 (36–43)	P4.0–P4.7
BUZ	I/O	Buzzer signal output	E-4	45 (47)	P5.3
SCK, SI, SO	I/O	Serial clock, serial data input, serial data output	E-4	42–44 (44–46)	P5.0–P5.2
RXD, TXD	I/O	UART data input, output	E-4	46-47 (48-49)	P5.4-P5.5
V_{LC1}	–	LCD bias voltage input pins	–	49 (51)	–
V_{SS2}, V_{DD2}	–	Power input pins for peripheral block	–	50, 51 (52, 53)	–
COM0–COM3	I/O	LCD Common signal output	H-32	52–55 (54–57)	P6.0-P6.3
SEG0–SEG3 (COM4–COM7)	I/O	LCD Common or Segment signal output	H-32	56–59 (58–61)	P6.4-P6.7
SEG4–SEG11	I/O	LCD segment signal output	H-32	60–67 (62–69)	P7.0-P7.7
SEG12–SEG15	I/O	LCD segment signal output	H-32	68–71 (70–73)	P8.0-P8.3
SEG16–SEG23	I/O	LCD segment signal output	H-32	72–79 (74–80, 1)	P0.0–P0.7
SEG24–SEG31	I/O	LCD segment signal output	H-32	80, 1–7 (2–9)	P1.0–P1.7

NOTE: Parentheses indicate pin number for 80-QFP-1420 package.

PIN CIRCUITS

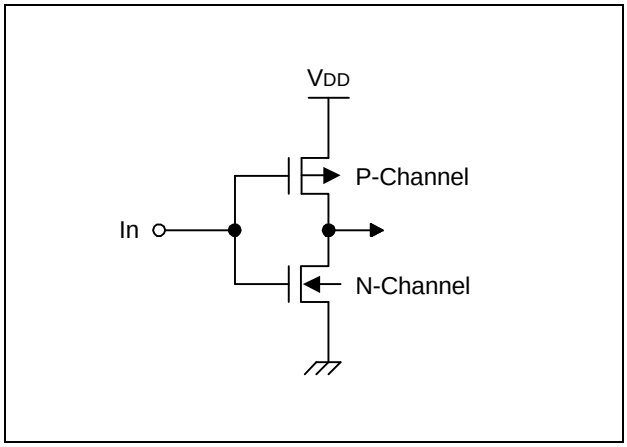


Figure 1-4. Pin Circuit Type A

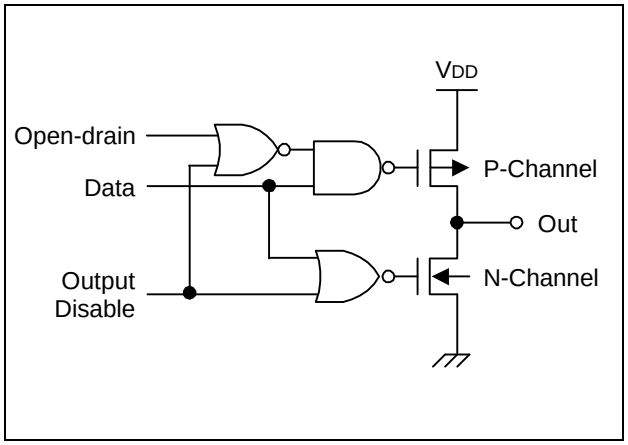


Figure 1-6. Pin Circuit Type C

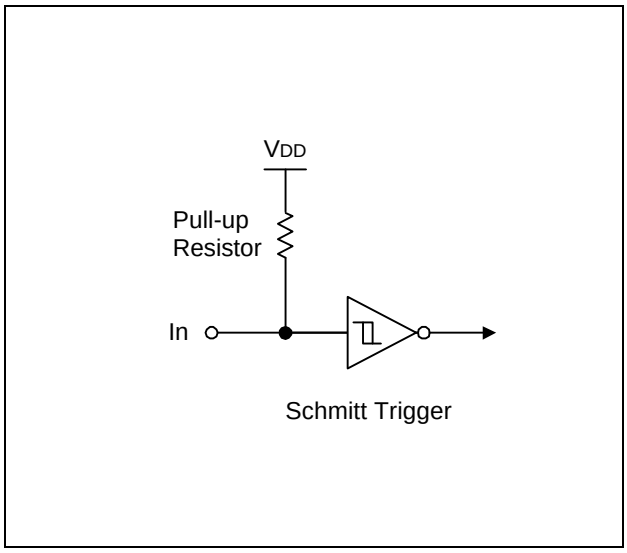


Figure 1-5. Pin Circuit Type B (RESET)

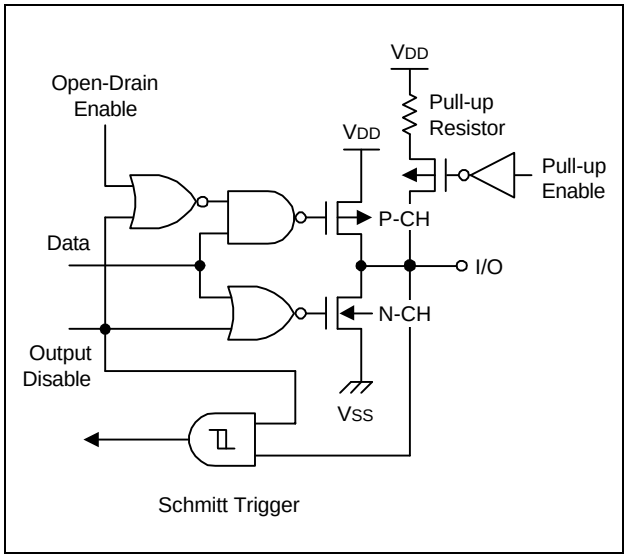


Figure 1-7. Pin Circuit Type E-4 (P2, P3.4–P3.7, P4, P5)

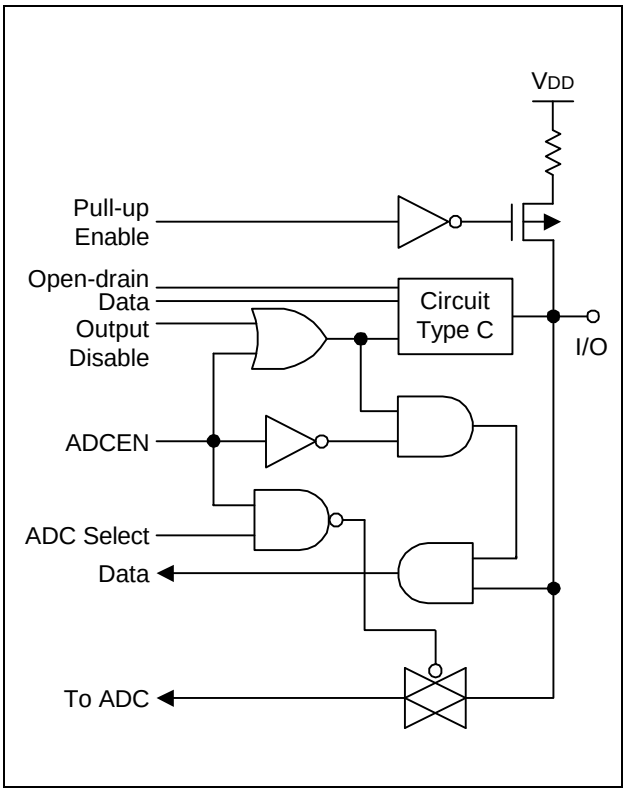


Figure 1-8. Pin Circuit Type F-16 (P3.0-P3.3)

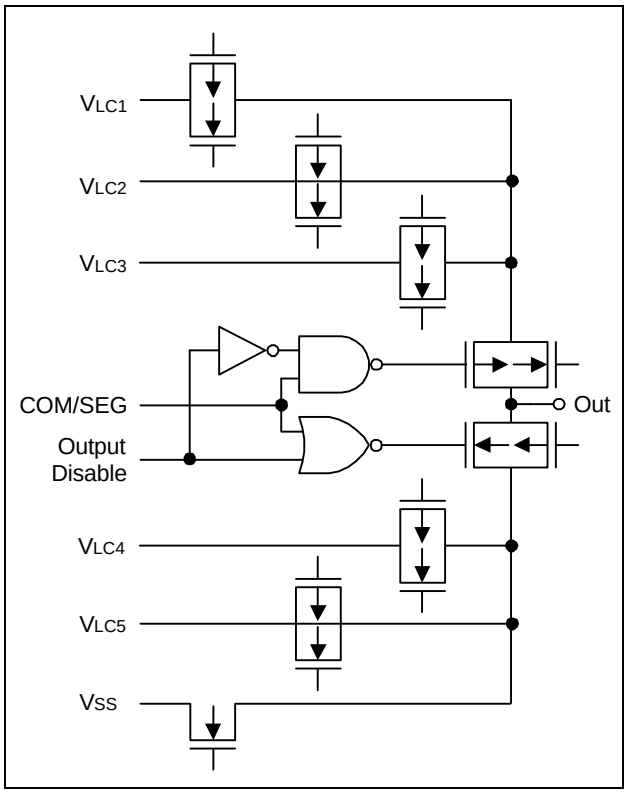


Figure 1-9. Pin Circuit Type H-23

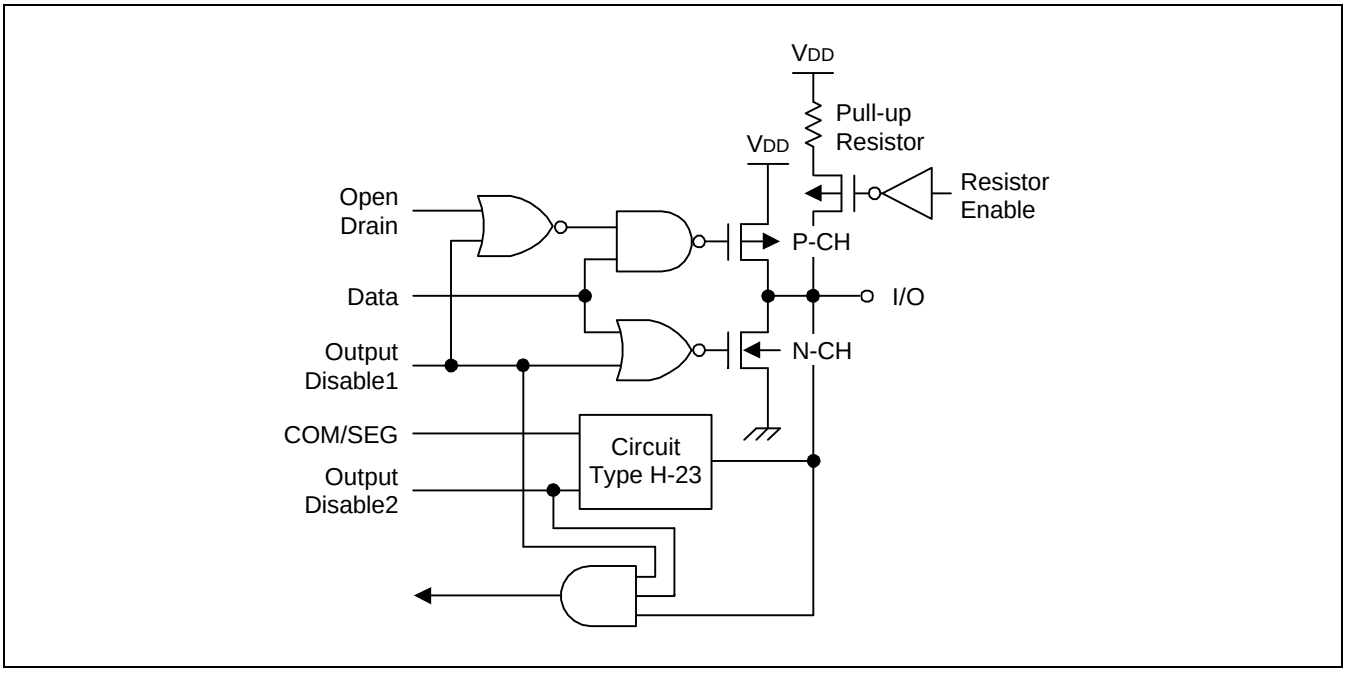


Figure 1-10. Pin Circuit Type H-32(P0, P1, P6-P8)

NOTES

2 ADDRESS SPACES

OVERVIEW

The S3C825A microcontroller has two types of address space:

— Internal program memory (ROM)

— Internal register file

A 16-bit address bus supports program memory operations. A separate 8-bit register bus carries addresses and data between the CPU and the register file.

The S3C825A has an internal 48-Kbyte mask-programmable ROM.

The 256-byte physical register space is expanded into an addressable area of 320 bytes using addressing modes.

A 32-byte LCD display register file is implemented.

There are 2,137 mapped registers in the internal register file. Of these, 2,064 are for general-purpose. (This number includes a 16-byte working register common area used as a "scratch area" for data operations, eight 192-byte prime register areas, and eight 64-byte areas (Set 2)). Thirteen 8-bit registers are used for the CPU and the system control, and 60 registers are mapped for peripheral controls and data registers. Seven register locations are not mapped.

PROGRAM MEMORY (ROM)

Program memory (ROM) stores program codes or table data. The S3C825A has 48K bytes internal mask-programmable program memory.

The first 256 bytes of the ROM (0H–0FFH) are reserved for interrupt vector addresses. Unused locations in this address range can be used as normal program memory. If you use the vector address area to store a program code, be careful not to overwrite the vector addresses stored in these locations.

The ROM address at which a program execution starts after a reset is 0100H.

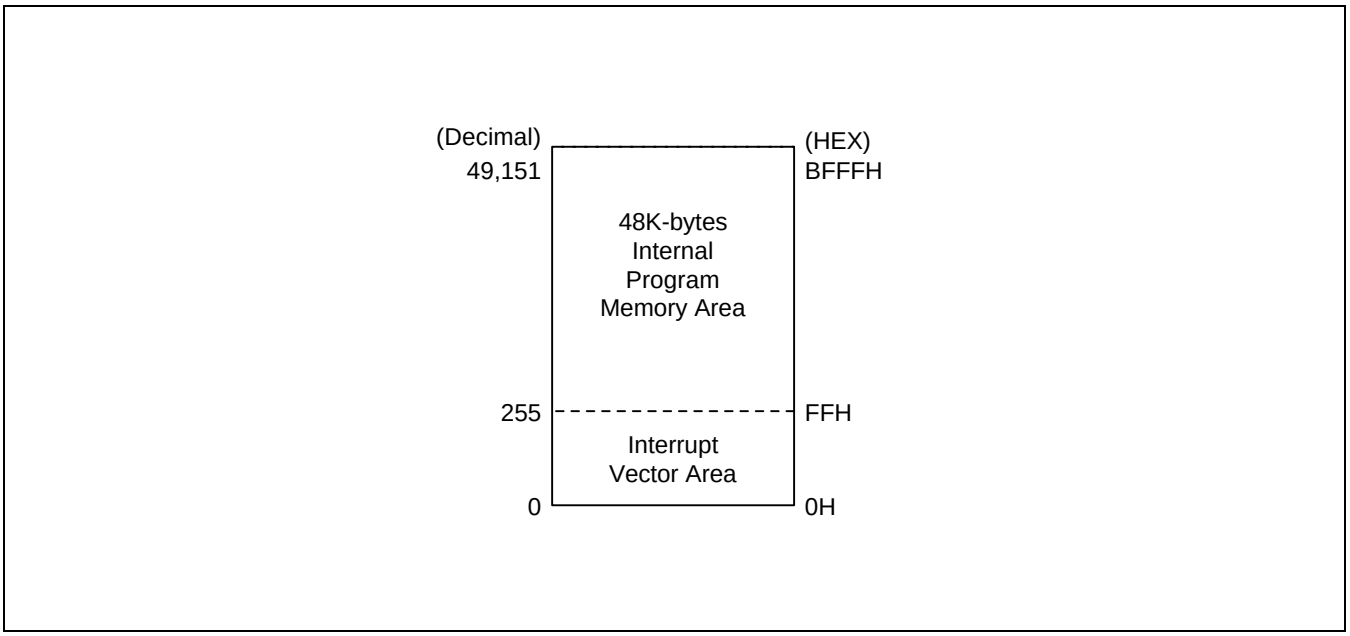


Figure 2-1. Program Memory Address Space

REGISTER ARCHITECTURE

In the S3C825A implementation, the upper 64-byte area of register files is expanded two 64-byte areas, called *set 1* and *set 2*. The upper 32-byte area of set 1 is further expanded two 32-byte register banks (bank 0 and bank 1), and the lower 32-byte area is a single 32-byte common area.

In case of S3C825A the total number of addressable 8-bit registers is 2137. Of these 2137 registers, 13 bytes are for CPU and system control registers, 60 bytes are for peripheral control and data registers, 16 bytes are used as a shared working registers, and 2048 registers are for general-purpose use, page 0-page 7 (including 32 bytes for LCD display registers).

You can always address set 1 register locations, regardless of which of the eight register pages is currently selected. Set 1 locations, however, can only be addressed using register addressing modes.

The extension of register space into separately addressable areas (sets, banks, and pages) is supported by various addressing mode restrictions, the select bank instructions, SB0 and SB1, and the register page pointer (PP).

Specific register types and the area (in bytes) that they occupy in the register file are summarized in Table 2–1.

Table 2-1. S3C825A Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16-byte common working register area, eight 192-byte prime register area (including LCD data registers), and eight 64-byte set 2 area).	2,064
CPU and system control registers	13
Mapped clock, peripheral, I/O control, and data registers	60
Total Addressable Bytes	2,137

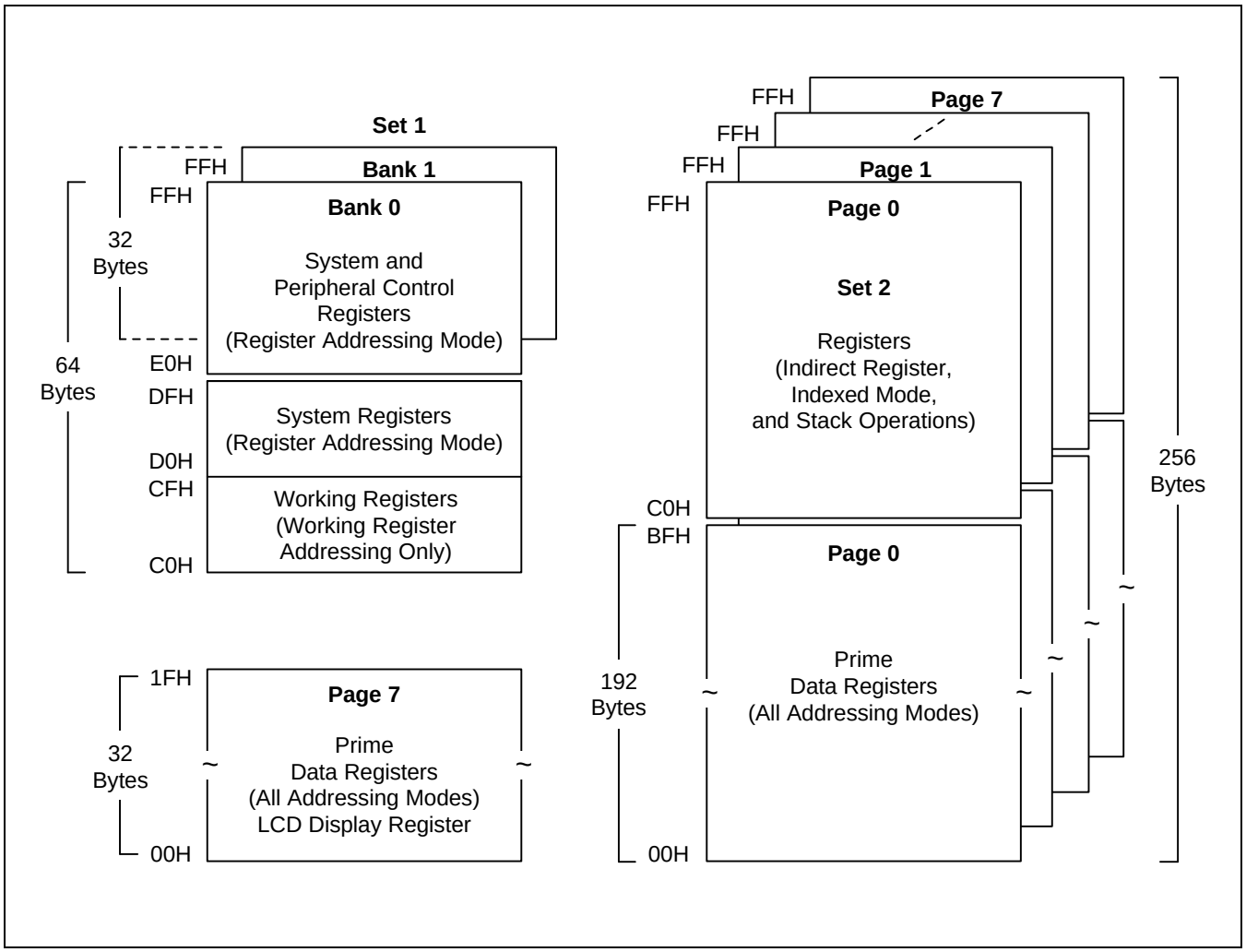


Figure 2-2. Internal Register File Organization

REGISTER PAGE POINTER (PP)

The S3C8-series architecture supports the logical expansion of the physical 256-byte internal register file (using an 8-bit data bus) into as many as 16 separately addressable register pages. Page addressing is controlled by the register page pointer (PP, DFH). In the S3C825A microcontroller, a paged register file expansion is implemented for LCD data registers, and the register page pointer must be changed to address other pages.

After a reset, the page pointer's source value (lower nibble) and the destination value (upper nibble) are always "0000", automatically selecting page 0 as the source and destination page for register addressing.

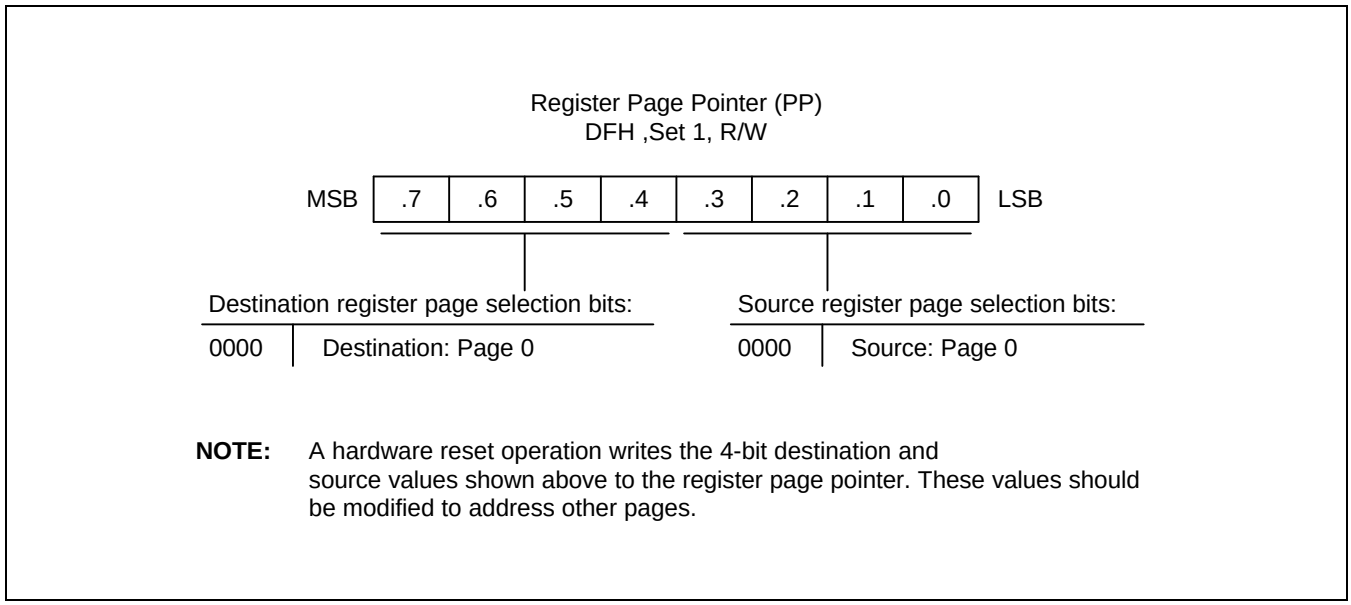


Figure 2-3. Register Page Pointer (PP)

☞ PROGRAMMING TIP — Using the Page Pointer for RAM clear (Page 0, Page 1)

```
        LD      PP,#00H         ;  Destination ← 0, Source ← 0
        SRP     #0C0H
        LD      R0,#0FFH        ;  Page 0 RAM clear starts
RAMCL0  CLR     @R0
        DJNZ    R0,RAMCL0
        CLR     @R0             ;  R0 = 00H

        LD      PP,#10H         ;  Destination ← 1, Source ← 0
        LD      R0,#0FFH        ;  Page 1 RAM clear starts
RAMCL1  CLR     @R0
        DJNZ    R0,RAMCL1
        CLR     @R0             ;  R0 = 00H
```

NOTE: You should refer to page 6-39 and use DJNZ instruction properly when DJNZ instruction is used in your program.

REGISTER SET 1

The term *set 1* refers to the upper 64 bytes of the register file, locations C0H–FFH.

The upper 32-byte area of this 64-byte space (E0H–FFH) is expanded two 32-byte register banks, *bank 0* and *bank 1*. The set register bank instructions, SB0 or SB1, are used to address one bank or the other. A hardware reset operation always selects bank 0 addressing.

The upper two 32-byte areas (bank 0 and bank 1) of set 1 (E0H–FFH) contains 57 mapped system and peripheral control registers. The lower 32-byte area contains 16 system registers (D0H–DFH) and a 16-byte common working register area (C0H–CFH). You can use the common working register area as a “scratch” area for data operations being performed in other areas of the register file.

Registers in set 1 locations are directly accessible at all times using Register addressing mode. The 16-byte working register area can only be accessed using working register addressing (For more information about working register addressing, please refer to Chapter 3, "Addressing Modes.")

REGISTER SET 2

The same 64-byte physical space that is used for set 1 locations C0H–FFH is logically duplicated to add another 64 bytes of register space. This expanded area of the register file is called set *2*. For the S3C825A, the set 2 address range (C0H–FFH) is accessible on pages 0-7.

The logical division of set 1 and set 2 is maintained by means of addressing mode restrictions. You can use only Register addressing mode to access set 1 locations. In order to access registers in set 2, you must use Register Indirect addressing mode or Indexed addressing mode.

The set 2 register area of page 0 is commonly used for stack operations.

PRIME REGISTER SPACE

The lower 192 bytes (00H–BFH) of the S3C825A's eight 256-byte register pages is called *prime register area.* Prime registers can be accessed using any of the seven addressing modes (see Chapter 3, "Addressing Modes.")

The prime register area on page 0 is immediately addressable following a reset. In order to address prime registers on pages 0, 1, 2, 3, 4, 5, 6, or 7, you must set the register page pointer (PP) to the appropriate source and destination values.

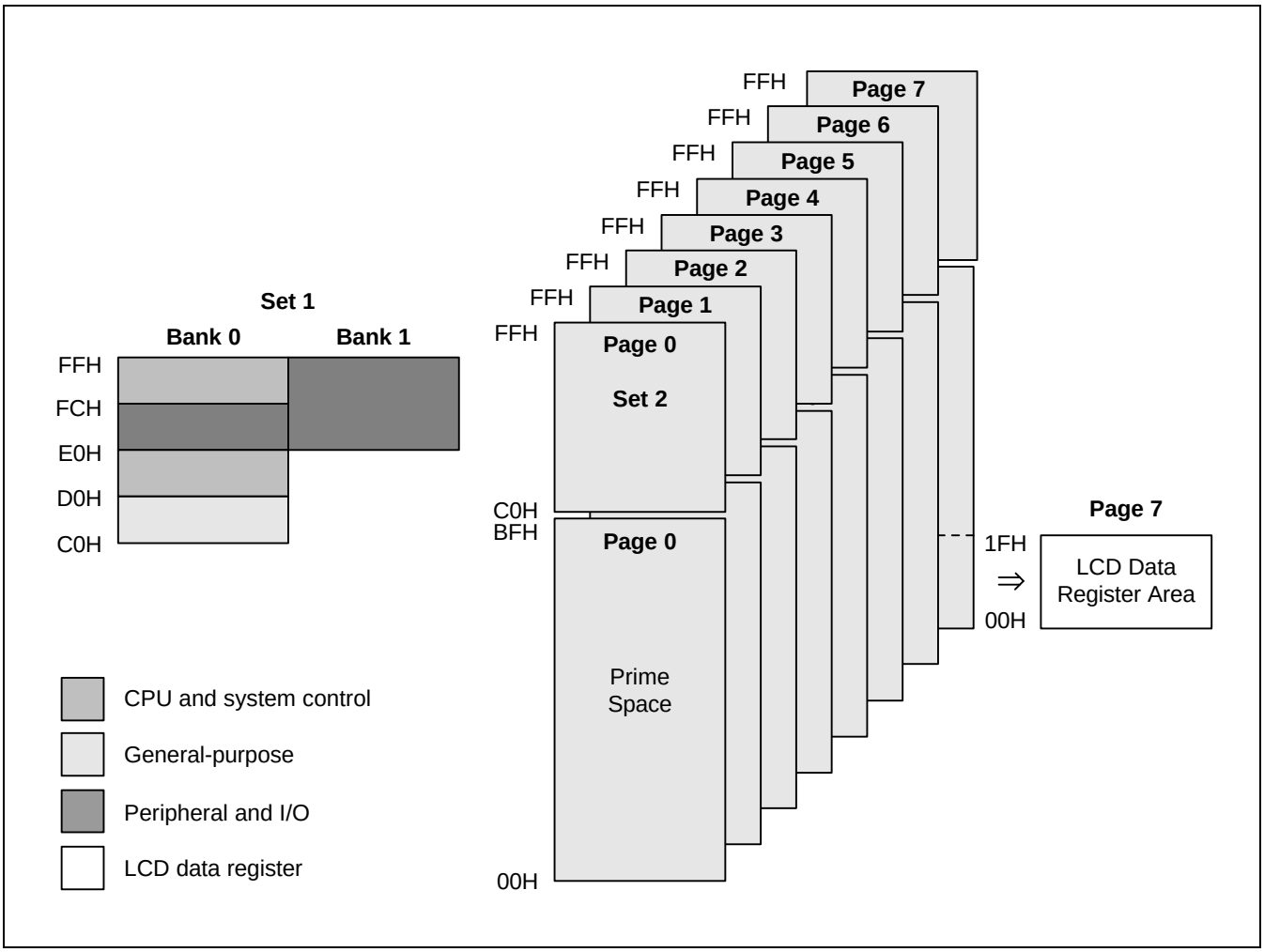


Figure 2-4. Set 1, Set 2, Prime Area Register, and LCD Data Register Map

WORKING REGISTERS

Instructions can access specific 8-bit registers or 16-bit register pairs using either 4-bit or 8-bit address fields. When 4-bit working register addressing is used, the 256-byte register file can be seen by the programmer as one that consists of 32 8-byte register groups or "slices." Each slice comprises of eight 8-bit registers.

Using the two 8-bit register pointers, RP1 and RP0, two working register slices can be selected at any one time to form a 16-byte working register block. Using the register pointers, you can move this 16-byte register block anywhere in the addressable register file, except the set 2 area.

The terms slice and block are used in this manual to help you visualize the size and relative locations of selected working register spaces:

— One working register *slice* is 8 bytes (eight 8-bit working registers, R0–R7 or R8–R15)

— One working register *block* is 16 bytes (sixteen 8-bit working registers, R0–R15)

All the registers in an 8-byte working register slice have the same binary value for their five most significant address bits. This makes it possible for each register pointer to point to one of the 24 slices in the register file. The base addresses for the two selected 8-byte register slices are contained in register pointers RP0 and RP1.

After a reset, RP0 and RP1 always point to the 16-byte common area in set 1 (C0H–CFH).

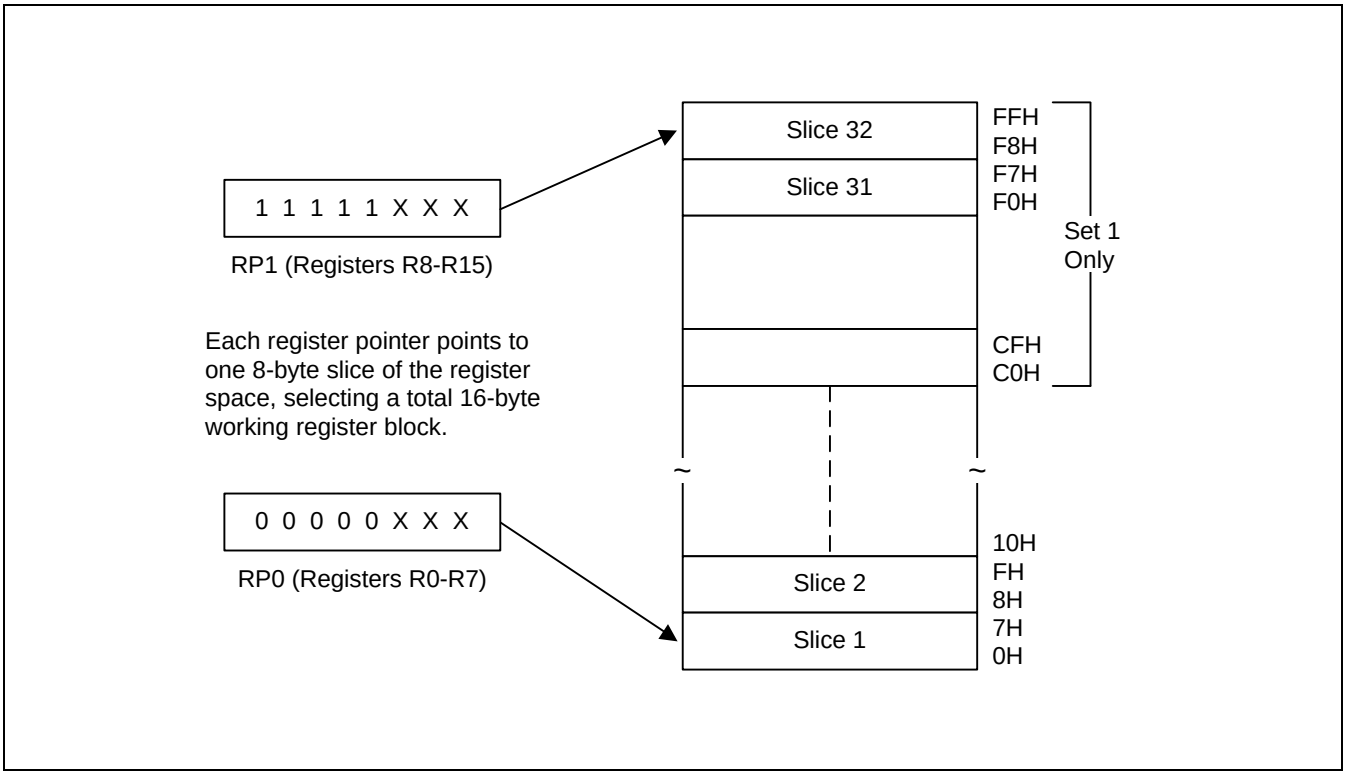


Figure 2-5. 8-Byte Working Register Areas (Slices)

USING THE REGISTER POINTS

Register pointers RP0 and RP1, mapped to addresses D6H and D7H in set 1, are used to select two movable 8-byte working register slices in the register file. After a reset, they point to the working register common area: RP0 points to addresses C0H–C7H, and RP1 points to addresses C8H–CFH.

To change a register pointer value, you load a new value to RP0 and/or RP1 using an SRP or LD instruction. (see Figures 2-6 and 2-7).

With working register addressing, you can only access those two 8-bit slices of the register file that are currently pointed to by RP0 and RP1. You cannot, however, use the register pointers to select a working register space in set 2, C0H–FFH, because these locations can be accessed only using the Indirect Register or Indexed addressing modes.

The selected 16-byte working register block usually consists of two contiguous 8-byte slices. As a general programming guideline, it is recommended that RP0 point to the "lower" slice and RP1 point to the "upper" slice (see Figure 2-6). In some cases, it may be necessary to define working register areas in different (non-contiguous) areas of the register file. In Figure 2-7, RP0 points to the "upper" slice and RP1 to the "lower" slice.

Because a register pointer can point to either of the two 8-byte slices in the working register block, you can flexibly define the working register area to support program requirements.

☞ PROGRAMMING TIP — Setting the Register Pointers

```
SRP     #70H        ;  RP0 ← 70H, RP1 ← 78H
SRP1    #48H        ;  RP0 ← no change, RP1 ← 48H,
SRP0    #0A0H       ;  RP0 ← A0H, RP1 ← no change
CLR     RP0         ;  RP0 ← 00H, RP1 ← no change
LD      RP1,#0F8H   ;  RP0 ← no change, RP1 ← 0F8H
```

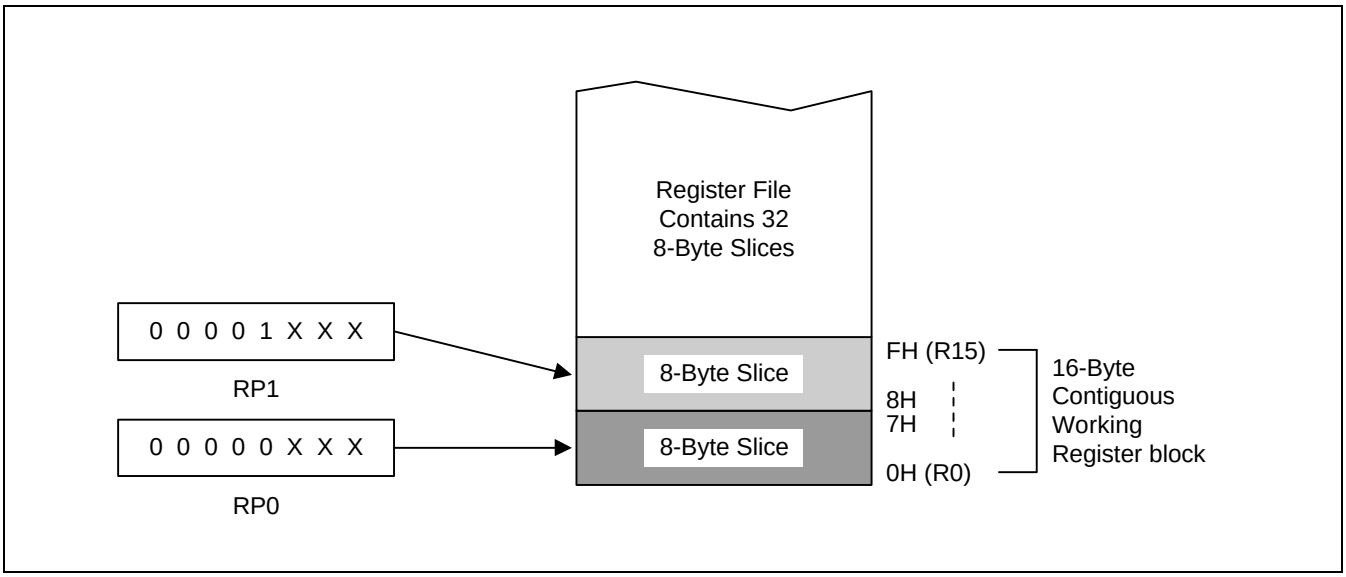


Figure 2-6. Contiguous 16-Byte Working Register Block

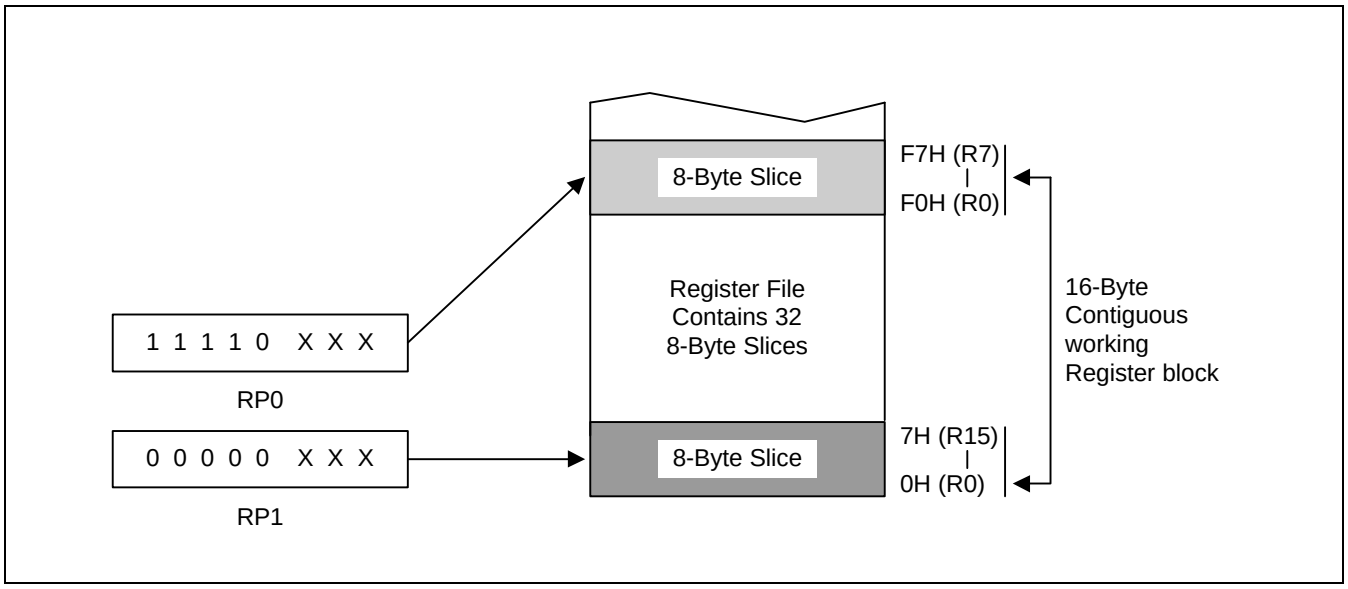


Figure 2-7. Non-Contiguous 16-Byte Working Register Block

☞ PROGRAMMING TIP — Using the RPs to Calculate the Sum of a Series of Registers

Calculate the sum of registers 80H–85H using the register pointer. The register addresses from 80H through 85H contain the values 10H, 11H, 12H, 13H, 14H, and 15 H, respectively:

```
SRP0    #80H        ;  RP0 ← 80H
ADD     R0,R1       ;  R0 ← R0 + R1
ADC     R0,R2       ;  R0 ← R0 + R2 + C
ADC     R0,R3       ;  R0 ← R0 + R3 + C
ADC     R0,R4       ;  R0 ← R0 + R4 + C
ADC     R0,R5       ;  R0 ← R0 + R5 + C
```

The sum of these six registers, 6FH, is located in the register R0 (80H). The instruction string used in this example takes 12 bytes of instruction code and its execution time is 36 cycles. If the register pointer is not used to calculate the sum of these registers, the following instruction sequence would have to be used:

```
ADD     80H,81H     ;  80H ← (80H) + (81H)
ADC     80H,82H     ;  80H ← (80H) + (82H) + C
ADC     80H,83H     ;  80H ← (80H) + (83H) + C
ADC     80H,84H     ;  80H ← (80H) + (84H) + C
ADC     80H,85H     ;  80H ← (80H) + (85H) + C
```

Now, the sum of the six registers is also located in register 80H. However, this instruction string takes 15 bytes of instruction code rather than 12 bytes, and its execution time is 50 cycles rather than 36 cycles.

REGISTER ADDRESSING

The S3C8-series register architecture provides an efficient method of working register addressing that takes full advantage of shorter instruction formats to reduce execution time.

With Register (R) addressing mode, in which the operand value is the content of a specific register or register pair, you can access any location in the register file except for set 2. With working register addressing, you use a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space.

Registers are addressed either as a single 8-bit register or as a paired 16-bit register space. In a 16-bit register pair, the address of the first 8-bit register is always an even number and the address of the next register is always an odd number. The most significant byte of the 16-bit data is always stored in the even-numbered register, and the least significant byte is always stored in the next (+1) odd-numbered register.

Working register addressing differs from Register addressing as it uses a register pointer to identify a specific 8-byte working register space in the internal register file and a specific 8-bit register within that space.

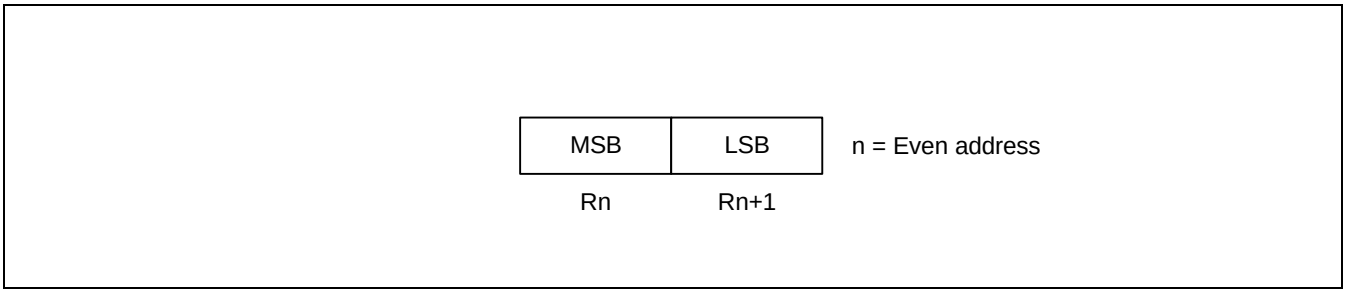


Figure 2-8. 16-Bit Register Pair

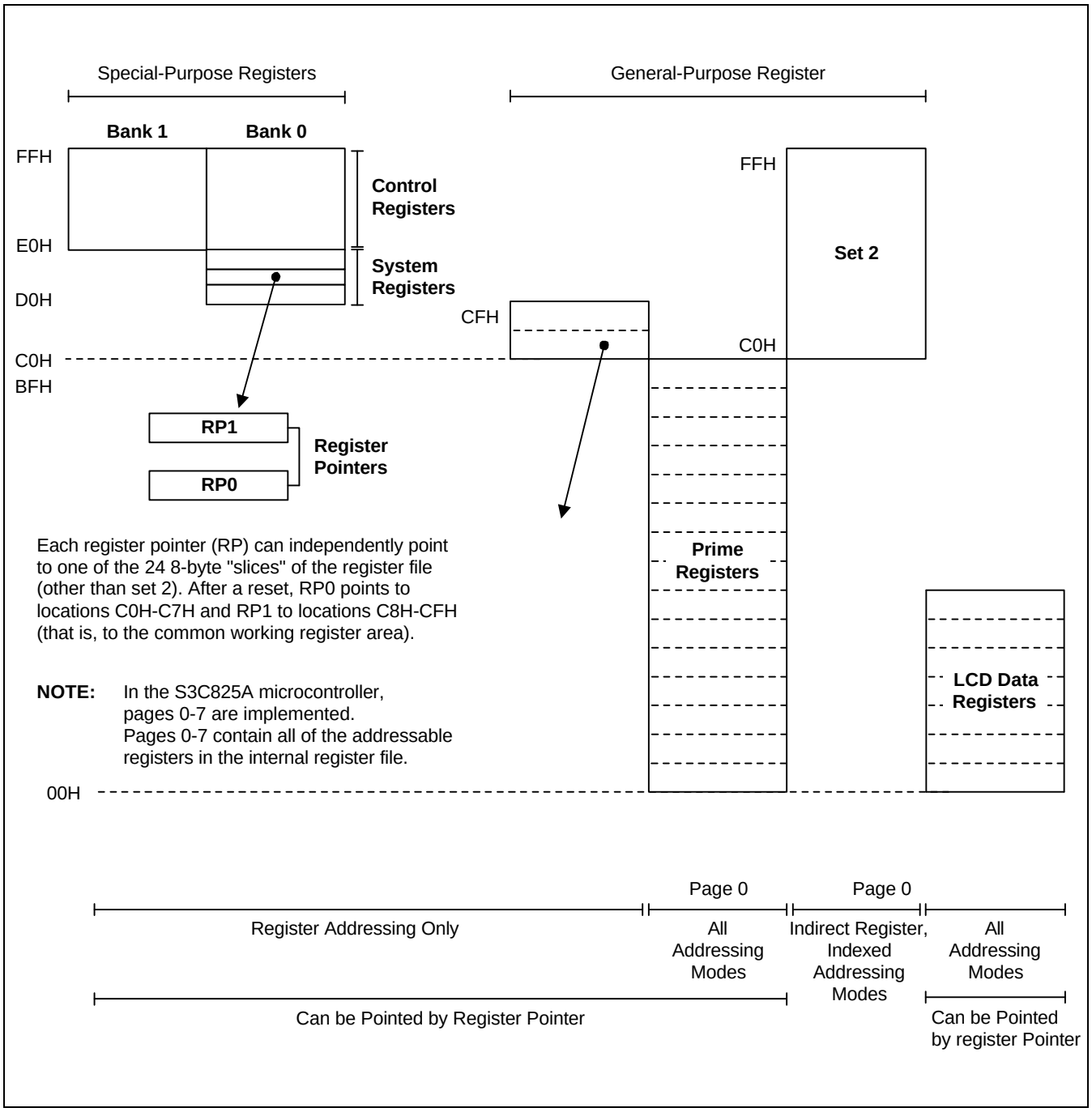


Figure 2-9. Register File Addressing

COMMON WORKING REGISTER AREA (C0H–CFH)

After a reset, register pointers RP0 and RP1 automatically select two 8-byte register slices in set 1, locations C0H–CFH, as the active 16-byte working register block:

RP0 → C0H–C7H

RP1 → C8H–CFH

This 16-byte address range is called *common area*. That is, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages.

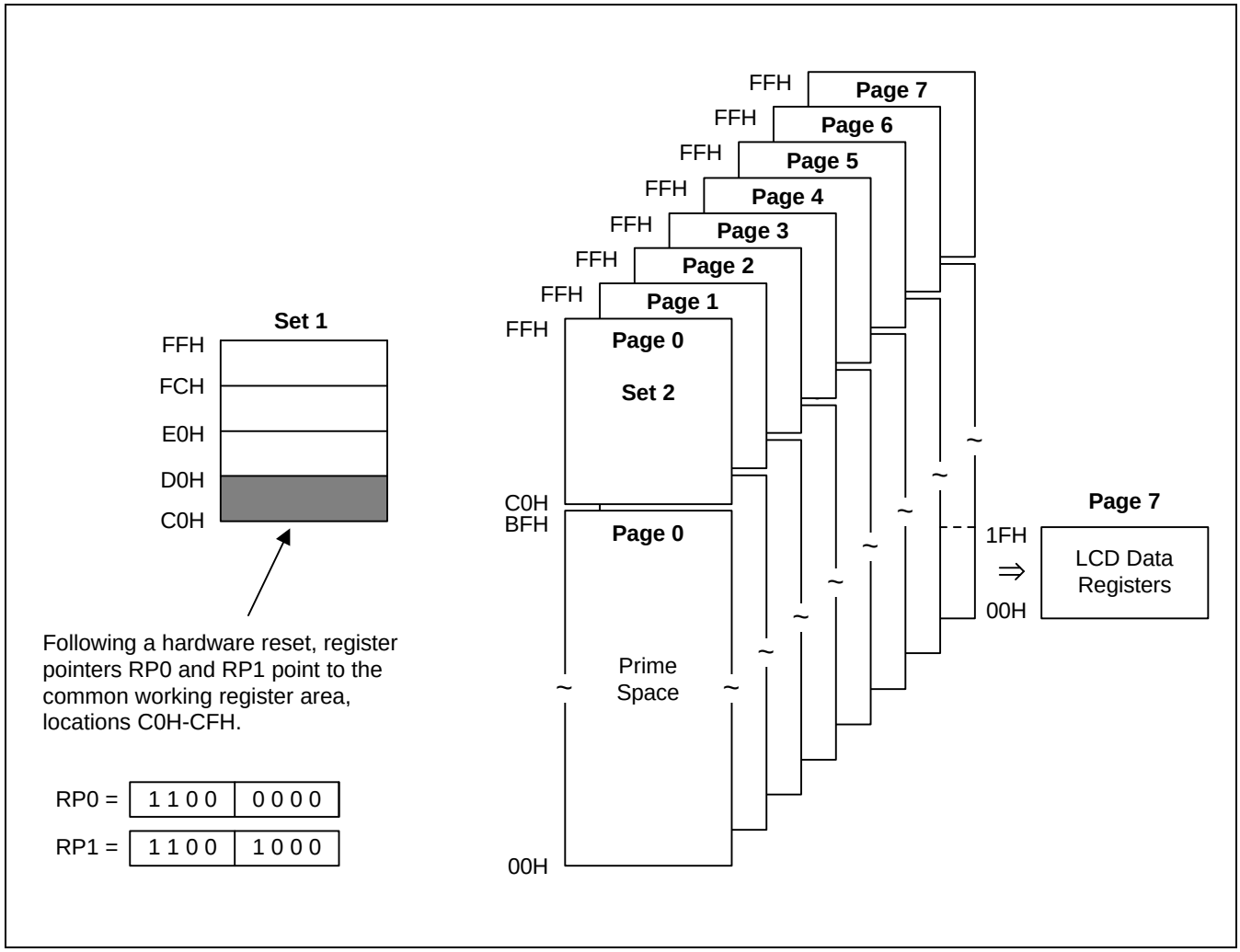


Figure 2-10. Common Working Register Area

☞ PROGRAMMING TIP — Addressing the Common Working Register Area

As the following examples show, you should access working registers in the common area, locations C0H–CFH, using working register addressing mode only.

Examples 1. LD 0C2H,40H ; Invalid addressing mode!

Use working register addressing instead:

```
SRP     #0C0H
LD      R2,40H          ; R2 (C2H) ← the value in location 40H
```

2. ADD 0C3H,#45H ; Invalid addressing mode!

Use working register addressing instead:

```
SRP     #0C0H
ADD     R3,#45H         ; R3 (C3H) ← R3 + 45H
```

4-BIT WORKING REGISTER ADDRESSING

Each register pointer defines a movable 8-byte slice of working register space. The address information stored in a register pointer serves as an addressing "window" that makes it possible for instructions to access working registers very efficiently using short 4-bit addresses. When an instruction addresses a location in the selected working register area, the address bits are concatenated in the following way to form a complete 8-bit address:

— The high-order bit of the 4-bit address selects one of the register pointers ("0" selects RP0, "1" selects RP1).

— The five high-order bits in the register pointer select an 8-byte slice of the register space.

— The three low-order bits of the 4-bit address select one of the eight registers in the slice.

As shown in Figure 2-11, the result of this operation is that the five high-order bits from the register pointer are concatenated with the three low-order bits from the instruction address to form the complete address. As long as the address stored in the register pointer remains unchanged, the three bits from the address will always point to an address in the same 8-byte register slice.

Figure 2-12 shows a typical example of 4-bit working register addressing. The high-order bit of the instruction "INC R6" is "0", which selects RP0. The five high-order bits stored in RP0 (01110B) are concatenated with the three low-order bits of the instruction's 4-bit address (110B) to produce the register address 76H (01110110B).

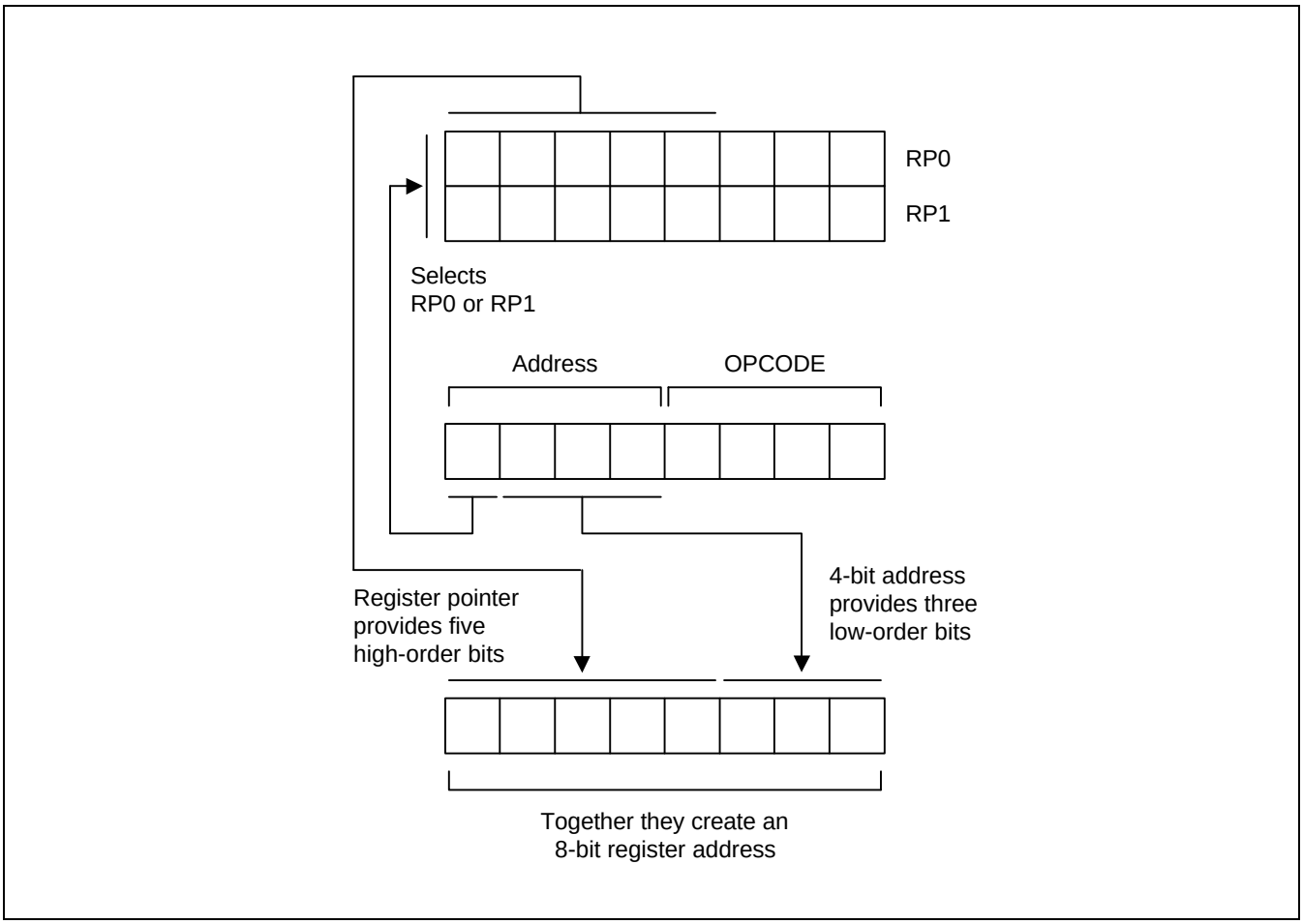


Figure 2-11. 4-Bit Working Register Addressing

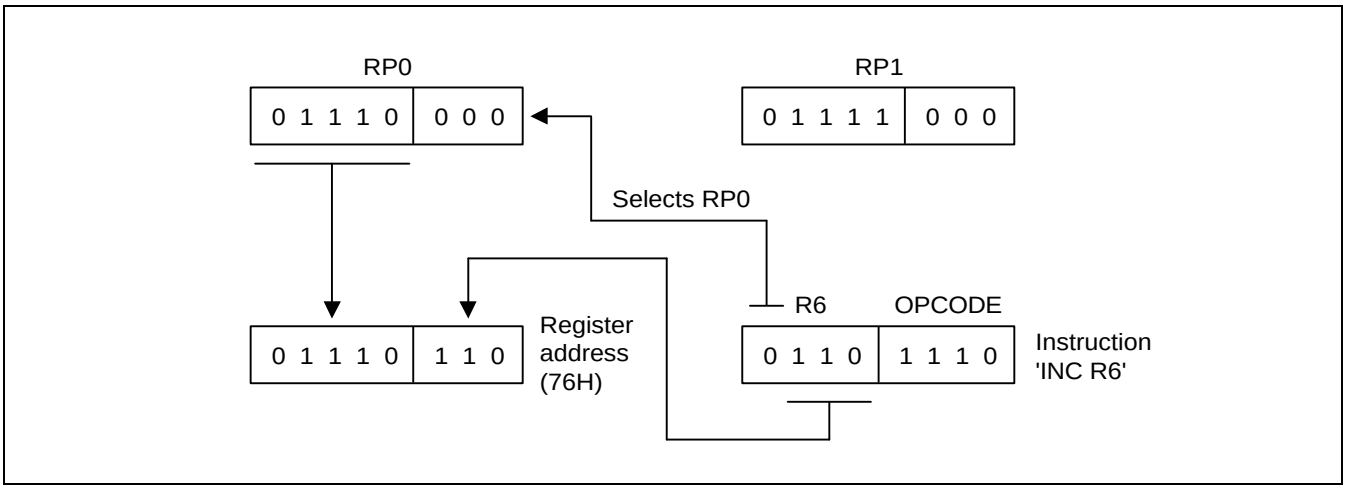


Figure 2-12. 4-Bit Working Register Addressing Example

8-BIT WORKING REGISTER ADDRESSING

You can also use 8-bit working register addressing to access registers in a selected working register area. To initiate 8-bit working register addressing, the upper four bits of the instruction address must contain the value "1100B." This 4-bit value (1100B) indicates that the remaining four bits have the same effect as 4-bit working register addressing.

As shown in Figure 2-13, the lower nibble of the 8-bit address is concatenated in much the same way as for 4-bit addressing: Bit 3 selects either RP0 or RP1, which then supplies the five high-order bits of the final address; the three low-order bits of the complete address are provided by the original instruction.

Figure 2-14 shows an example of 8-bit working register addressing. The four high-order bits of the instruction address (1100B) specify 8-bit working register addressing. Bit 4 ("1") selects RP1 and the five high-order bits in RP1 (10101B) become the five high-order bits of the register address. The three low-order bits of the register address (011) are provided by the three low-order bits of the 8-bit instruction address. The five address bits from RP1 and the three address bits from the instruction are concatenated to form the complete register address, 0ABH (10101011B).

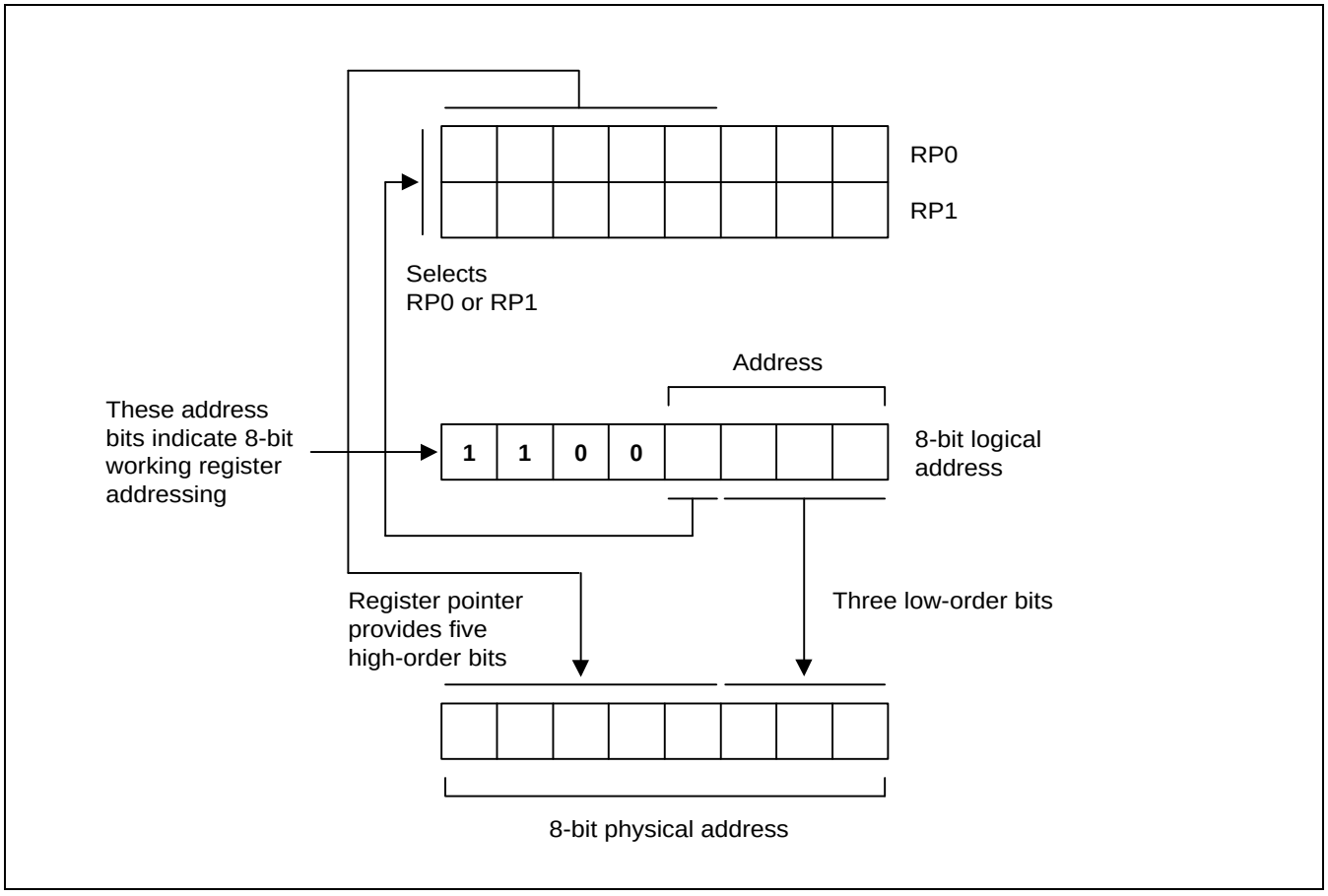


Figure 2-13. 8-Bit Working Register Addressing

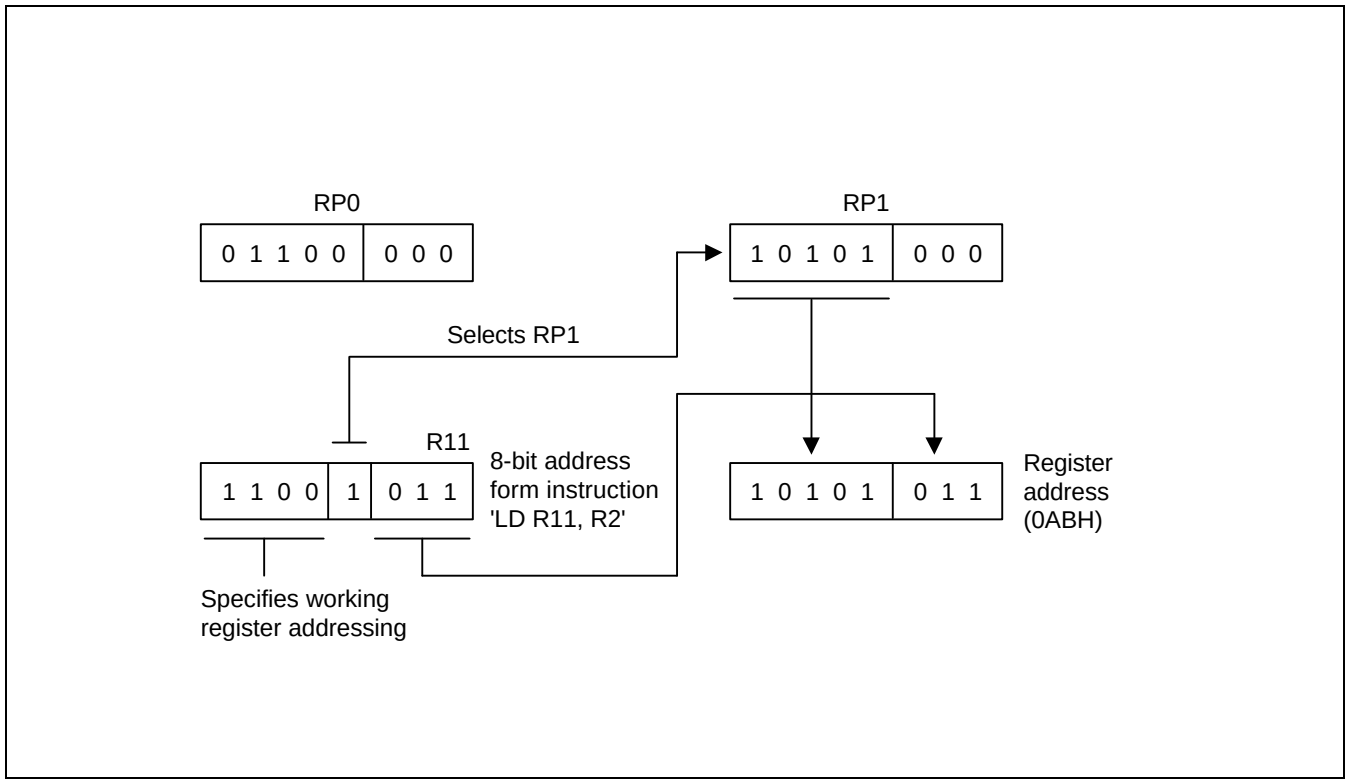


Figure 2-14. 8-Bit Working Register Addressing Example

SYSTEM AND USER STACK

The S3C8-series microcontrollers use the system stack for data storage, subroutine calls and returns. The PUSH and POP instructions are used to control system stack operations. The S3C825A architecture supports stack operations in the internal register file.

Stack Operations

Return addresses for procedure calls, interrupts, and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the PC and the FLAGS register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address value is always decreased by one before a push operation and increased by one *after* a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of the stack, as shown in Figure 2-15.

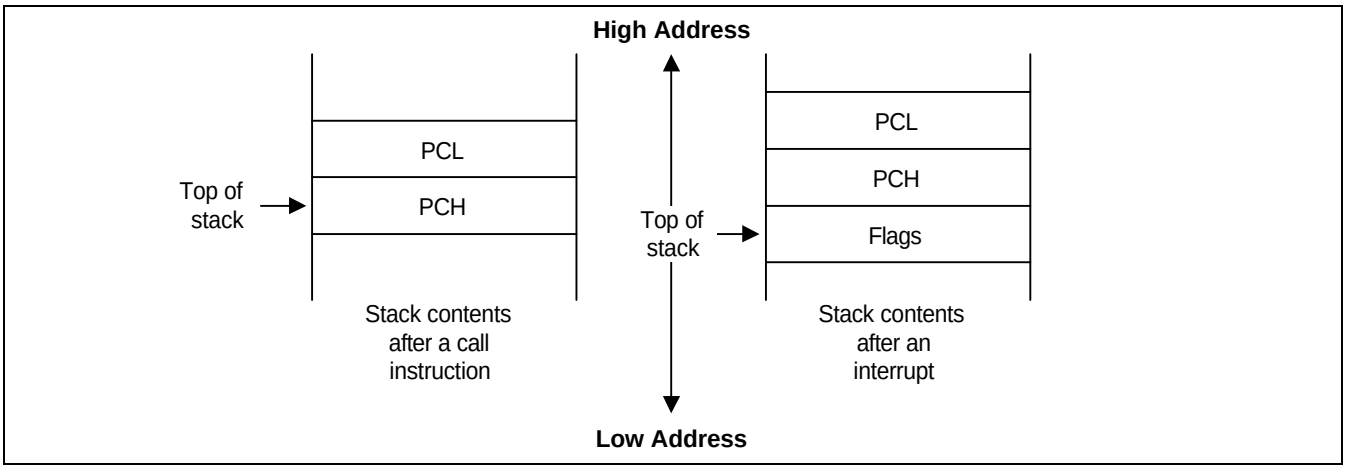


Figure 2-15. Stack Operations

User-Defined Stacks

You can freely define stacks in the internal register file as data storage locations. The instructions PUSHUI, PUSHUD, POPUI, and POPUD support user-defined stack operations.

Stack Pointers (SPL, SPH)

Register locations D8H and D9H contain the 16-bit stack pointer (SP) that is used for system stack operations. The most significant byte of the SP address, SP15–SP8, is stored in the SPH register (D8H), and the least significant byte, SP7–SP0, is stored in the SPL register (D9H). After a reset, the SP value is undetermined.

Because only internal memory space is implemented in the S3C825A, the SPL must be initialized to an 8-bit value in the range 00H–FFH. The SPH register is not needed and can be used as a general-purpose register, if necessary.

When the SPL register contains the only stack pointer value (that is, when it points to a system stack in the register file), you can use the SPH register as a general-purpose data register. However, if an overflow or underflow condition occurs as a result of increasing or decreasing the stack address value in the SPL register during normal stack operations, the value in the SPL register will overflow (or underflow) to the SPH register, overwriting any other data that is currently stored there. To avoid overwriting data in the SPH register, you can initialize the SPL value to "FFH" instead of "00H".

☞ PROGRAMMING TIP — Standard Stack Operations Using PUSH and POP

The following example shows you how to perform stack operations in the internal register file using PUSH and POP instructions:

```
LD      SPL,#0FFH       ; SPL ← FFH
                        ; (Normally, the SPL is set to 0FFH by the initialization
                        ; routine)
•
•
•
PUSH    PP              ; Stack address 0FEH ← PP
PUSH    RP0             ; Stack address 0FDH ← RP0
PUSH    RP1             ; Stack address 0FCH ← RP1
PUSH    R3              ; Stack address 0FBH ← R3
•
•
•
POP     R3              ; R3 ← Stack address 0FBH
POP     RP1             ; RP1 ← Stack address 0FCH
POP     RP0             ; RP0 ← Stack address 0FDH
POP     PP              ; PP ← Stack address 0FEH
```

NOTES

3 ADDRESSING MODES

OVERVIEW

Instructions that are stored in program memory are fetched for execution using the program counter. Instructions indicate the operation to be performed and the data to be operated on. Addressing mode is the method used to determine the location of the data operand. The operands specified in SAM88RC instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The S3C8-series instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction. The seven addressing modes and their symbols are:

— Register (R)
— Indirect Register (IR)
— Indexed (X)
— Direct Address (DA)
— Indirect Address (IA)
— Relative Address (RA)
— Immediate (IM)

REGISTER ADDRESSING MODE (R)

In Register addressing mode (R), the operand value is the content of a specified register or register pair (see Figure 3-1).

Working register addressing differs from Register addressing in that it uses a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space (see Figure 3-2).

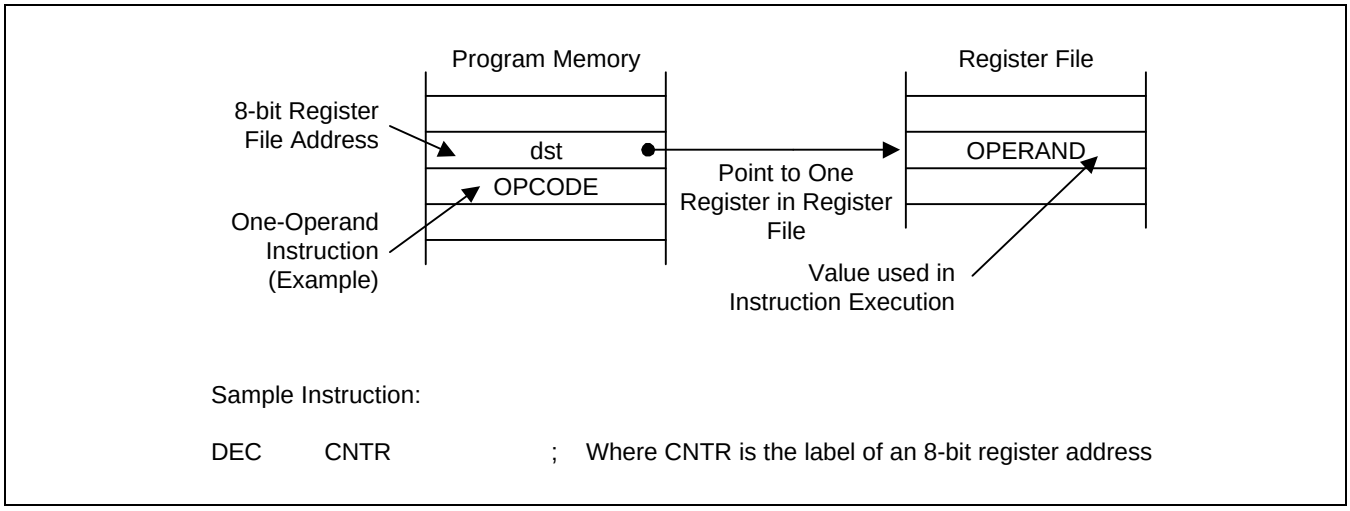


Figure 3-1. Register Addressing

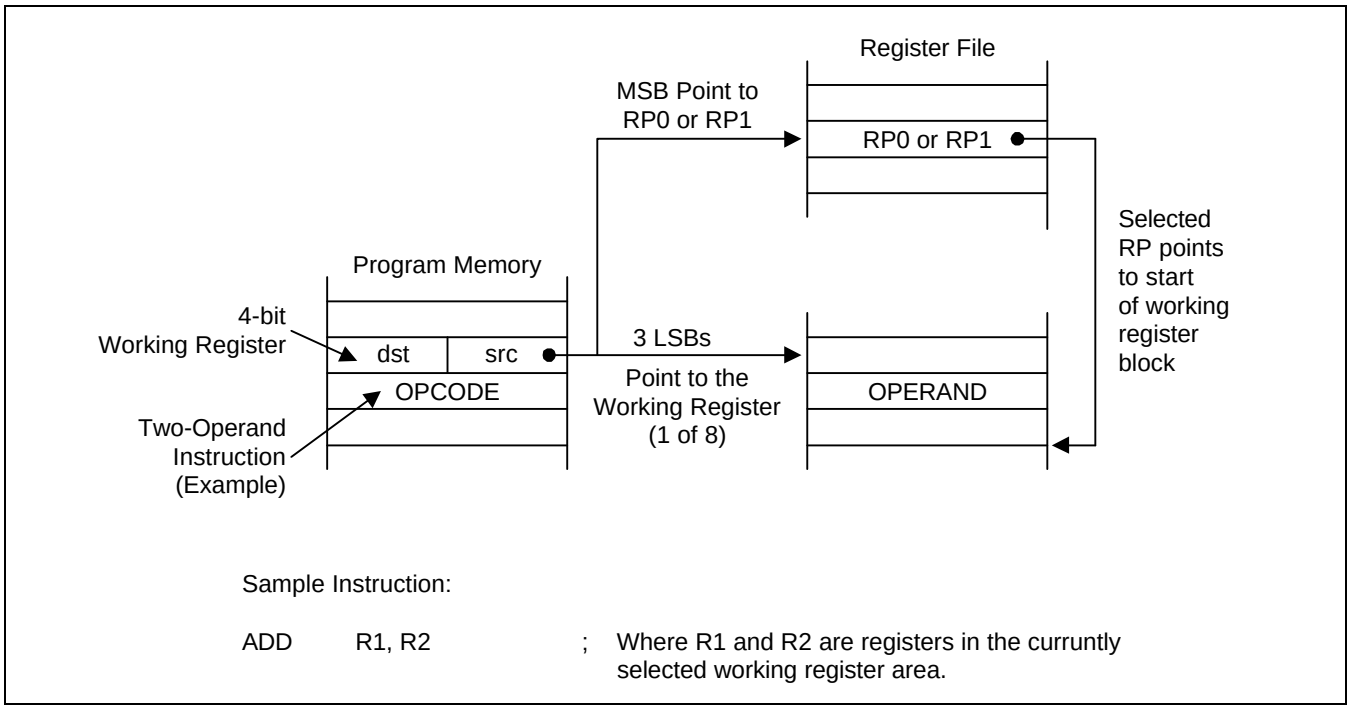
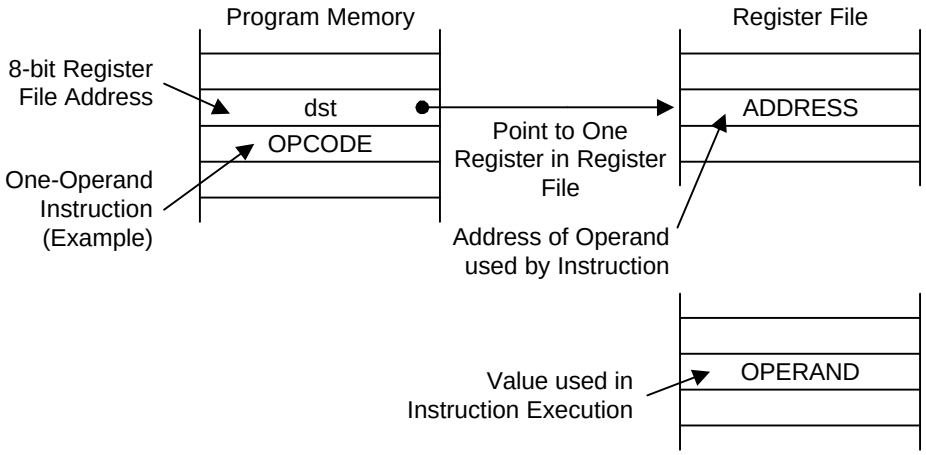


Figure 3-2. Working Register Addressing

INDIRECT REGISTER ADDRESSING MODE (IR)

In Indirect Register (IR) addressing mode, the content of the specified register or register pair is the address of the operand. Depending on the instruction used, the actual address may point to a register in the register file, to program memory (ROM), or to an external memory space (see Figures 3-3 through 3-6).

You can use any 8-bit register to indirectly address another register. Any 16-bit register pair can be used to indirectly address another memory location. Please note, however, that you cannot access locations C0H–FFH in set 1 using the Indirect Register addressing mode.



Sample Instruction:

```
RL      @SHIFT          ;   Where SHIFT is the label of an 8-bit register address
```

Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

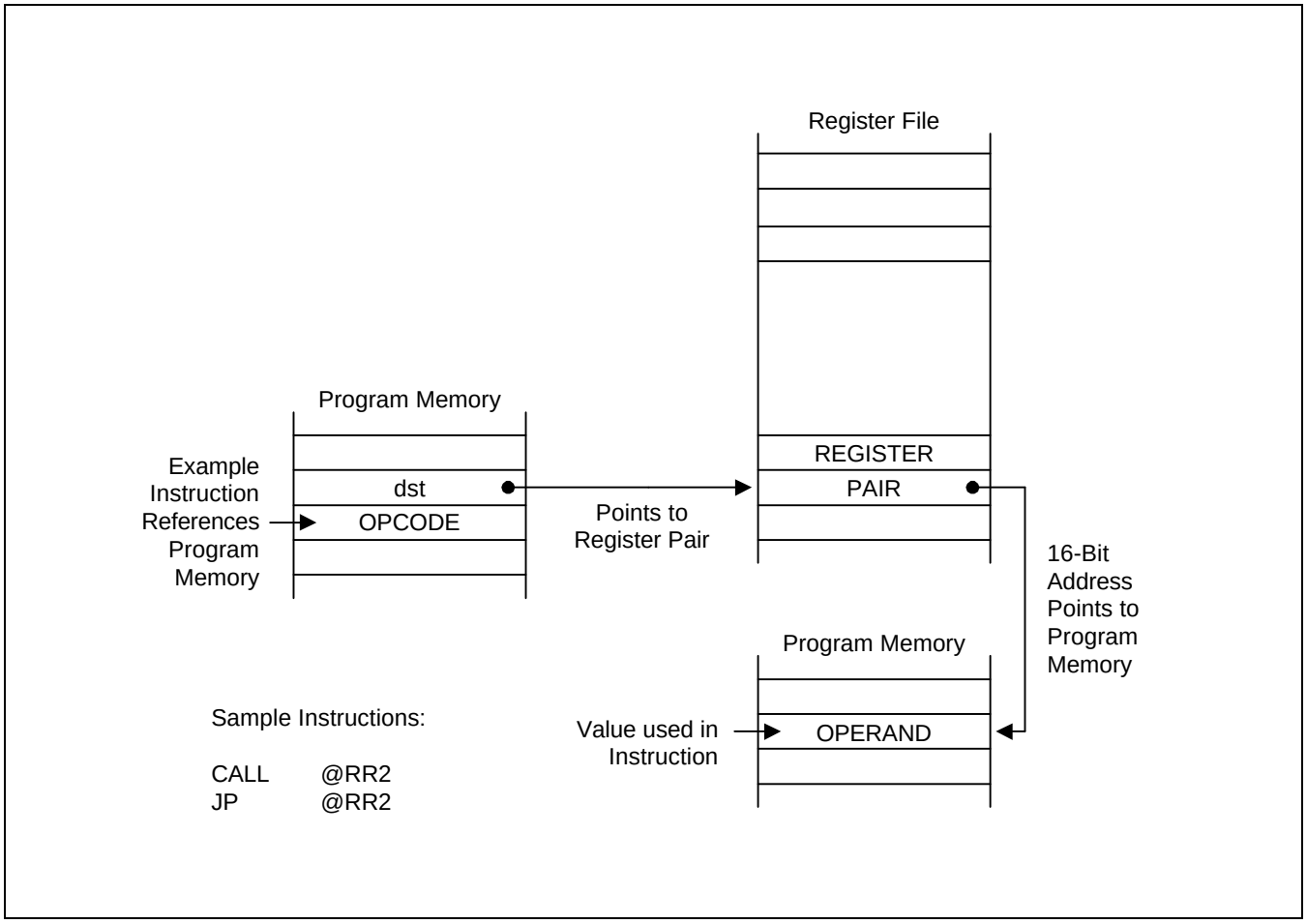


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

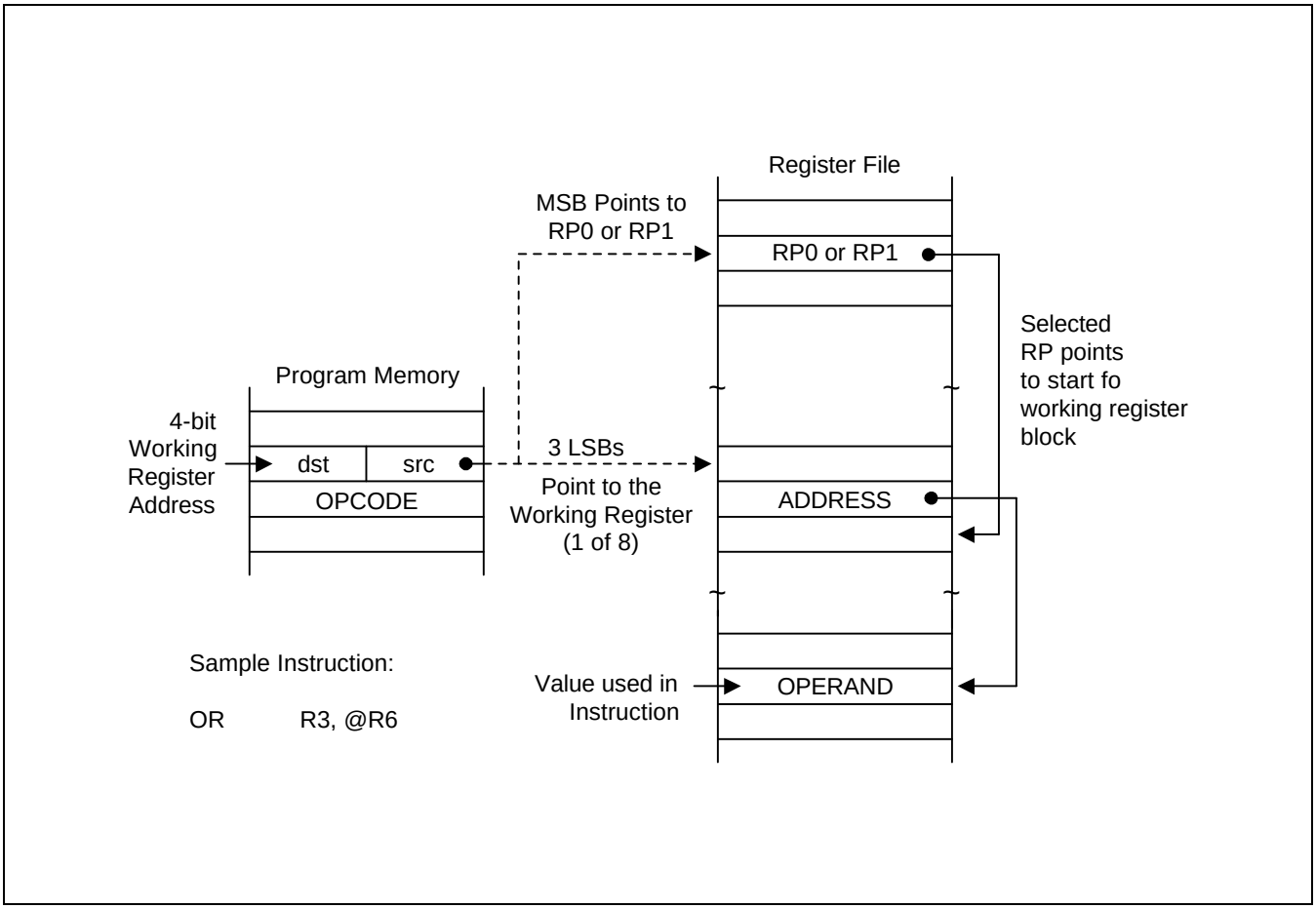
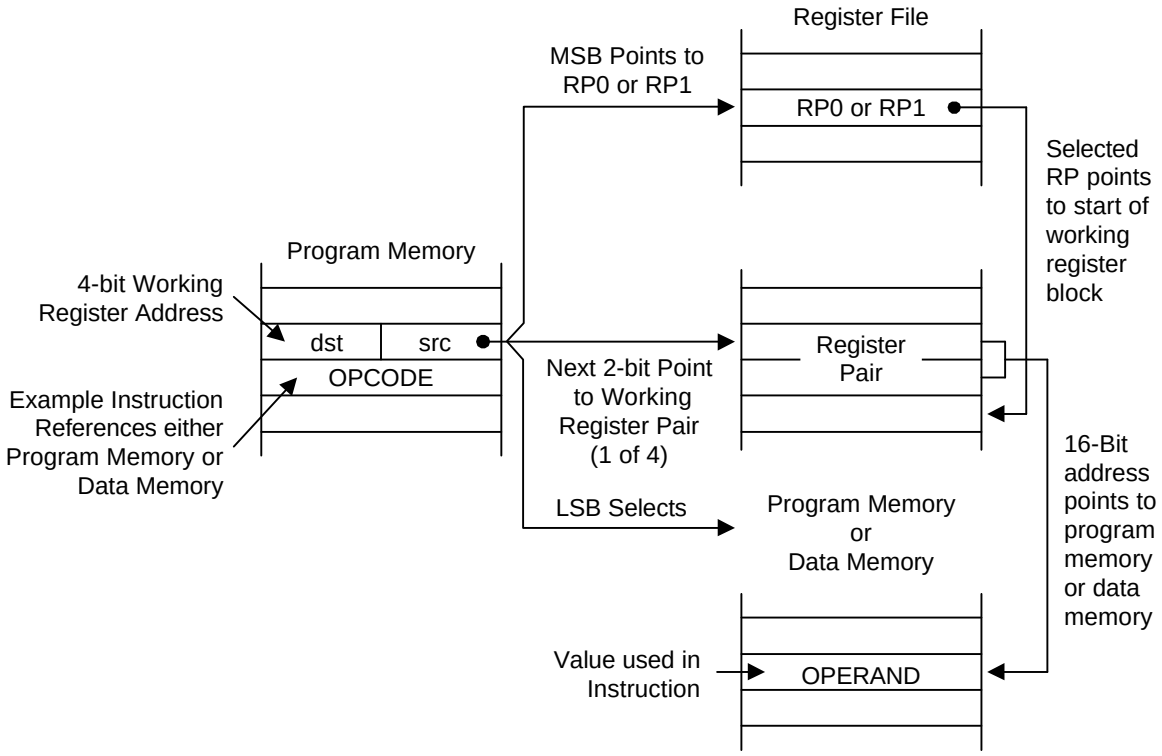


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)



Sample Instructions:

```
LDC     R5,@RR6       ;  Program memory access
LDE     R3,@RR14      ;  External data memory access
LDE     @RR4, R8      ;  External data memory access
```

Figure 3-6. Indirect Working Register Addressing to Program or Data Memory

INDEXED ADDRESSING MODE (X)

Indexed (X) addressing mode adds an offset value to a base address during instruction execution in order to calculate the effective operand address (see Figure 3-7). You can use Indexed addressing mode to access locations in the internal register file or in external memory. Please note, however, that you cannot access locations C0H–FFH in set 1 using Indexed addressing mode.

In short offset Indexed addressing mode, the 8-bit displacement is treated as a signed integer in the range –128 to +127. This applies to external memory accesses only (see Figure 3-8.)

For register file addressing, an 8-bit base address provided by the instruction is added to an 8-bit offset contained in a working register. For external memory accesses, the base address is stored in the working register pair designated in the instruction. The 8-bit or 16-bit offset given in the instruction is then added to that base address (see Figure 3-9).

The only instruction that supports Indexed addressing mode for the internal register file is the Load instruction (LD). The LDC and LDE instructions support Indexed addressing mode for internal program memory and for external data memory, when implemented.

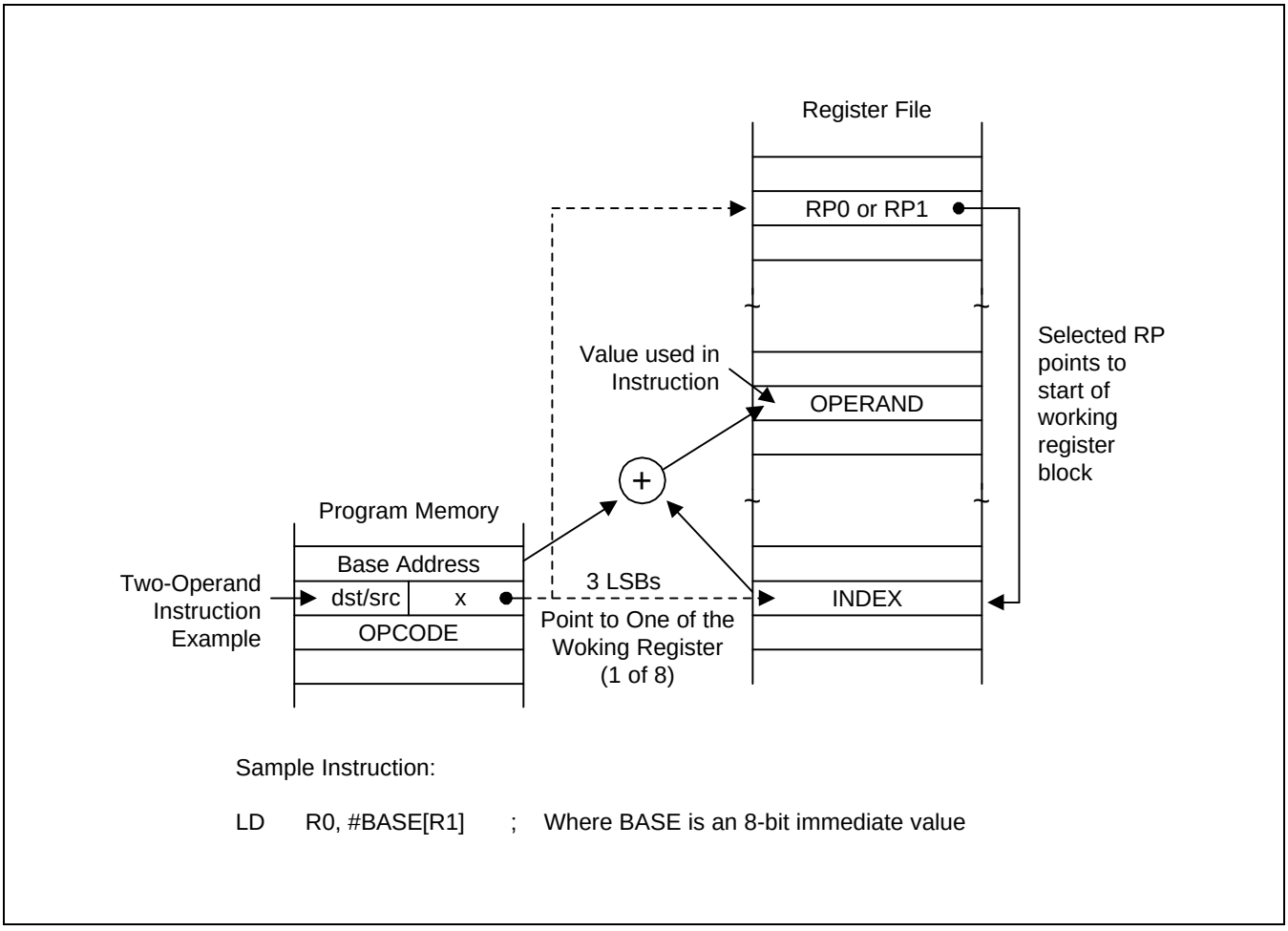
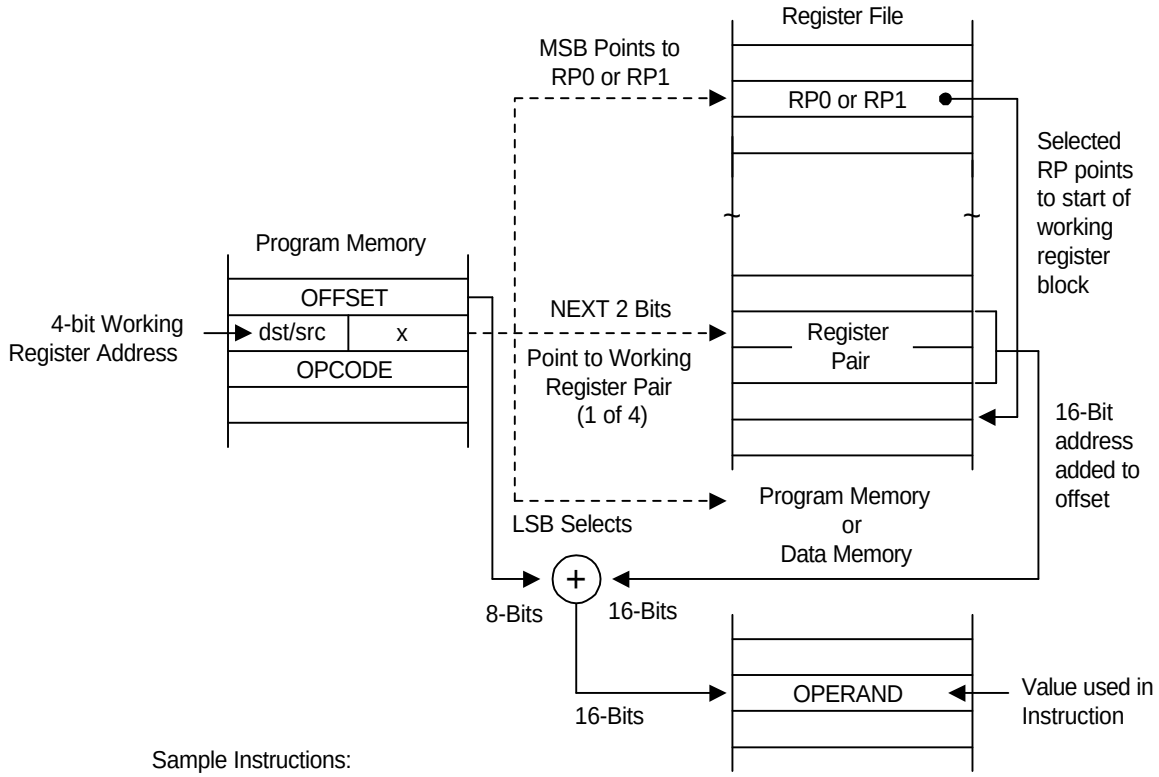


Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)

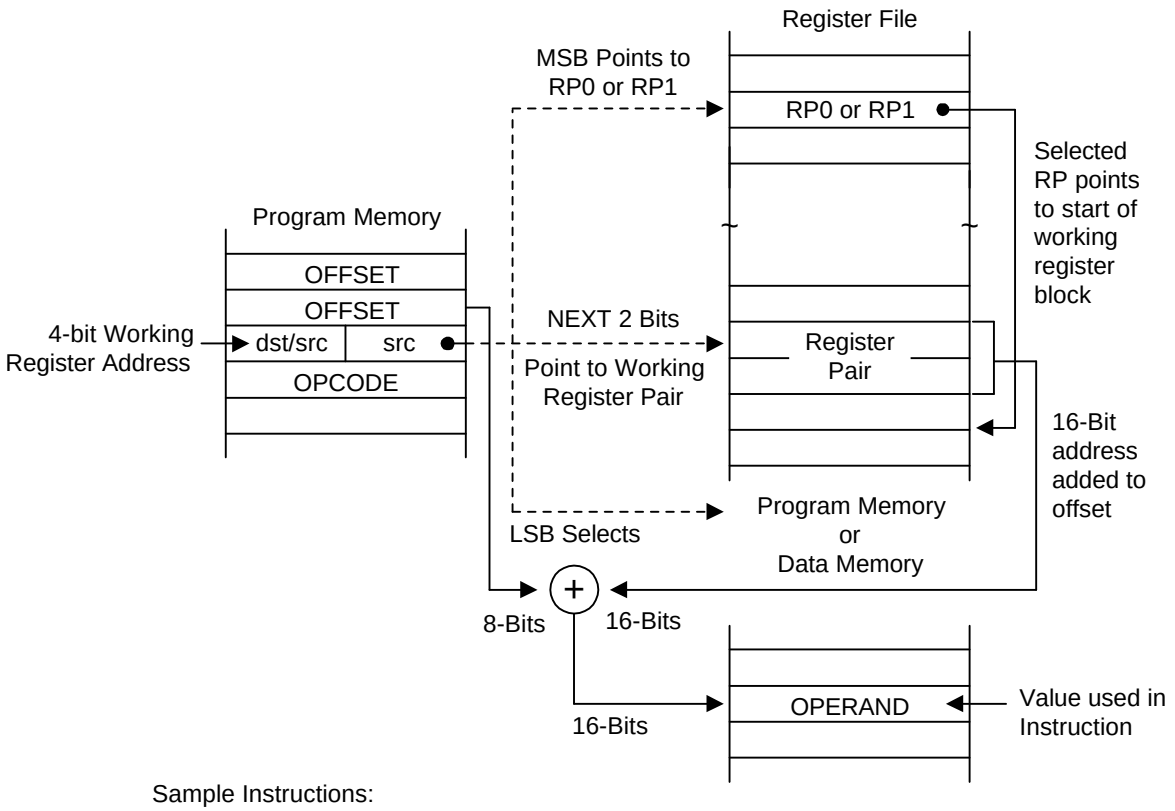


Sample Instructions:

```
LDC     R4, #04H[RR2]     ; The values in the program address (RR2 + 04H)
                          ; are loaded into register R4.
LDE     R4,#04H[RR2]      ; Identical operation to LDC example, except that
                          ; external program memory is accessed.
```

Figure 3-8. Indexed Addressing to Program or Data Memory with Short Offset

INDEXED ADDRESSING MODE (Concluded)



Sample Instructions:

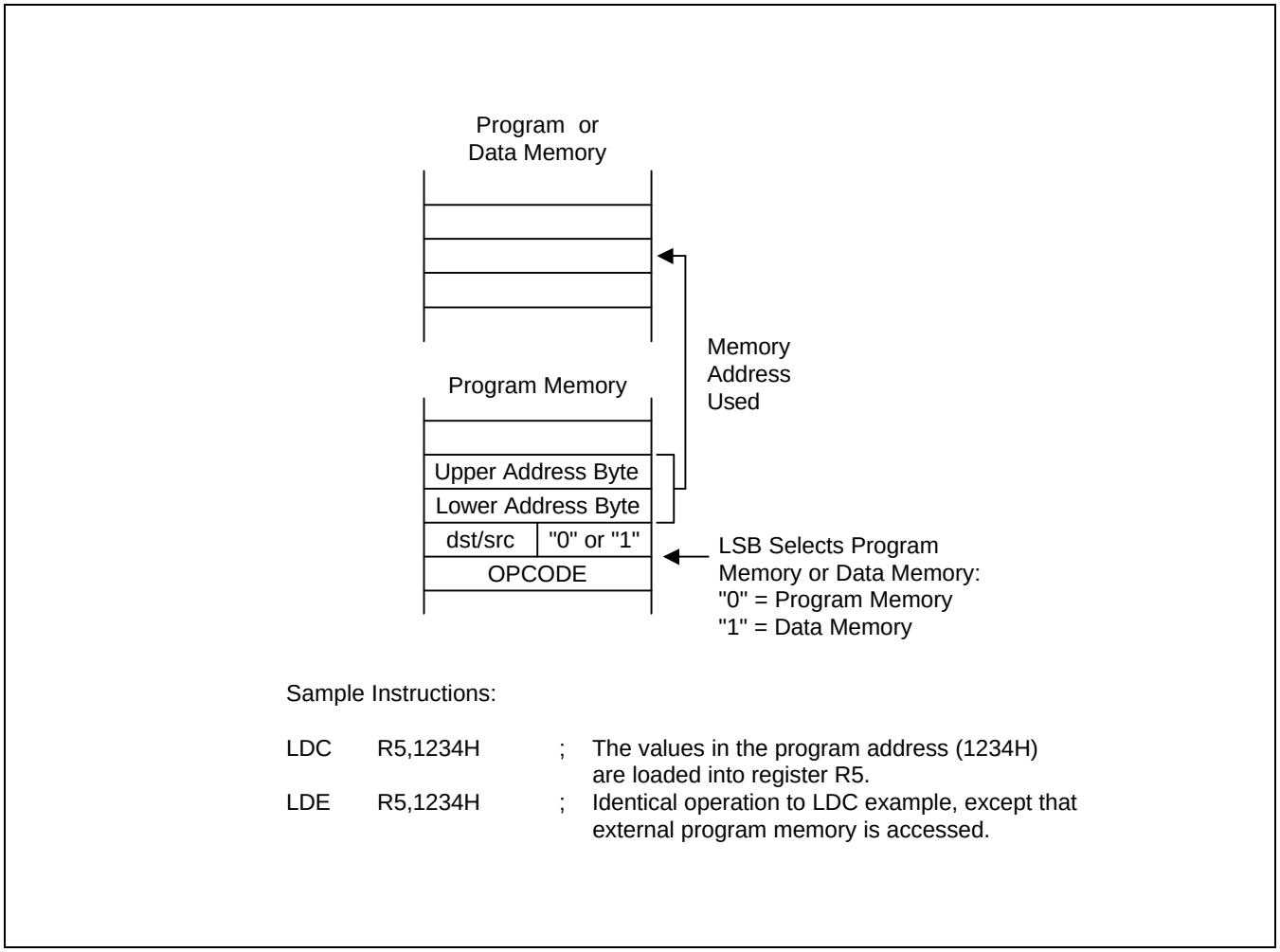
```
LDC     R4, #1000H[RR2]     ; The values in the program address (RR2 + 1000H)
                              are loaded into register R4.
LDE     R4,#1000H[RR2]      ; Identical operation to LDC example, except that
                              external program memory is accessed.
```

Figure 3-9. Indexed Addressing to Program or Data Memory

DIRECT ADDRESS MODE (DA)

In Direct Address (DA) mode, the instruction provides the operand's 16-bit memory address. Jump (JP) and Call (CALL) instructions use this addressing mode to specify the 16-bit destination address that is loaded into the PC whenever a JP or CALL instruction is executed.

The LDC and LDE instructions can use Direct Address mode to specify the source or destination address for Load operations to program memory (LDC) or to external data memory (LDE), if implemented.

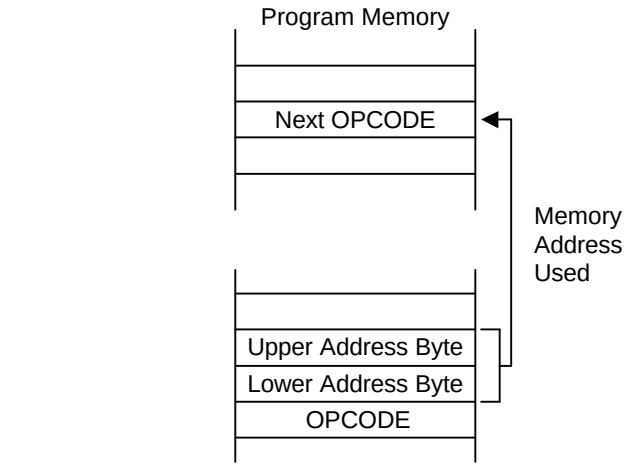


Sample Instructions:

```
LDC     R5,1234H    ;   The values in the program address (1234H)
                        are loaded into register R5.
LDE     R5,1234H    ;   Identical operation to LDC example, except that
                        external program memory is accessed.
```

Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)



Sample Instructions:

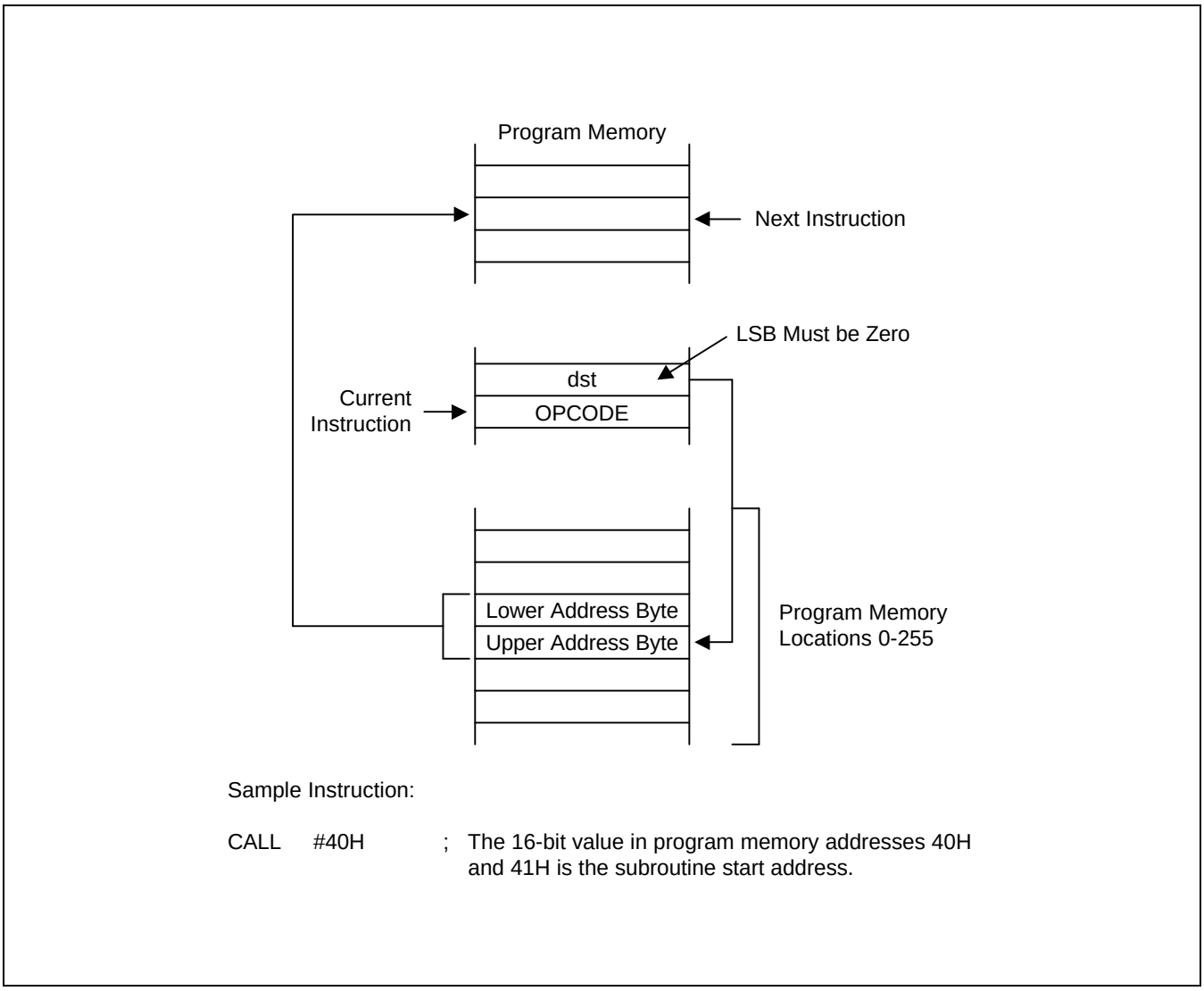
```
JP      C,JOB1      ;  Where JOB1 is a 16-bit immediate address
CALL    DISPLAY     ;  Where DISPLAY is a 16-bit immediate address
```

Figure 3-11. Direct Addressing for Call and Jump Instructions

INDIRECT ADDRESS MODE (IA)

In Indirect Address (IA) mode, the instruction specifies an address located in the lowest 256 bytes of the program memory. The selected pair of memory locations contains the actual address of the next instruction to be executed. Only the CALL instruction can use the Indirect Address mode.

Because the Indirect Address mode assumes that the operand is located in the lowest 256 bytes of program memory, only an 8-bit address is supplied in the instruction; the upper bytes of the destination address are assumed to be all zeros.



Sample Instruction:

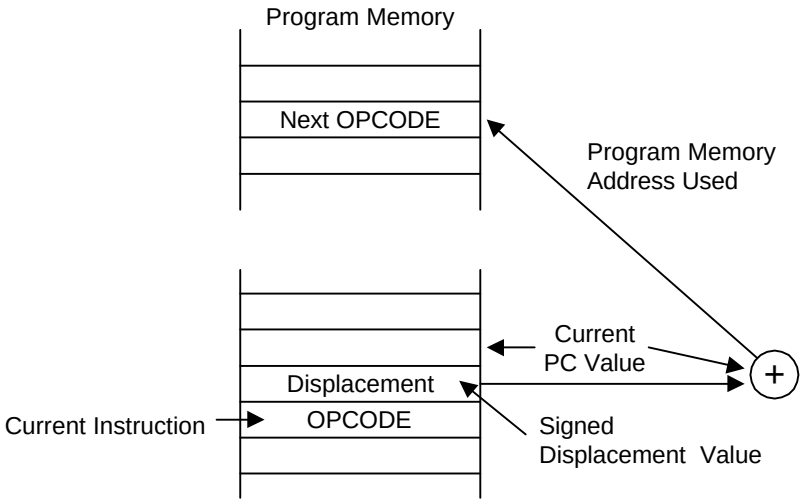
```
CALL    #40H        ;  The 16-bit value in program memory addresses 40H
                       and 41H is the subroutine start address.
```

Figure 3-12. Indirect Addressing

RELATIVE ADDRESS MODE (RA)

In Relative Address (RA) mode, a twos-complement signed displacement between – 128 and + 127 is specified in the instruction. The displacement value is then added to the current PC value. The result is the address of the next instruction to be executed. Before this addition occurs, the PC contains the address of the instruction immediately following the current instruction.

Several program control instructions use the Relative Address mode to perform conditional jumps. The instructions that support RA addressing are BTJRF, BTJRT, DJNZ, CPIJE, CPIJNE, and JR.



Sample Instructions:

```
JR      ULT,$+OFFSET    ;   Where OFFSET is a value in the range +127 to -128
```

Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

In Immediate (IM) addressing mode, the operand value used in the instruction is the value supplied in the operand field itself. The operand may be one byte or one word in length, depending on the instruction used. Immediate addressing mode is useful for loading constant values into registers.

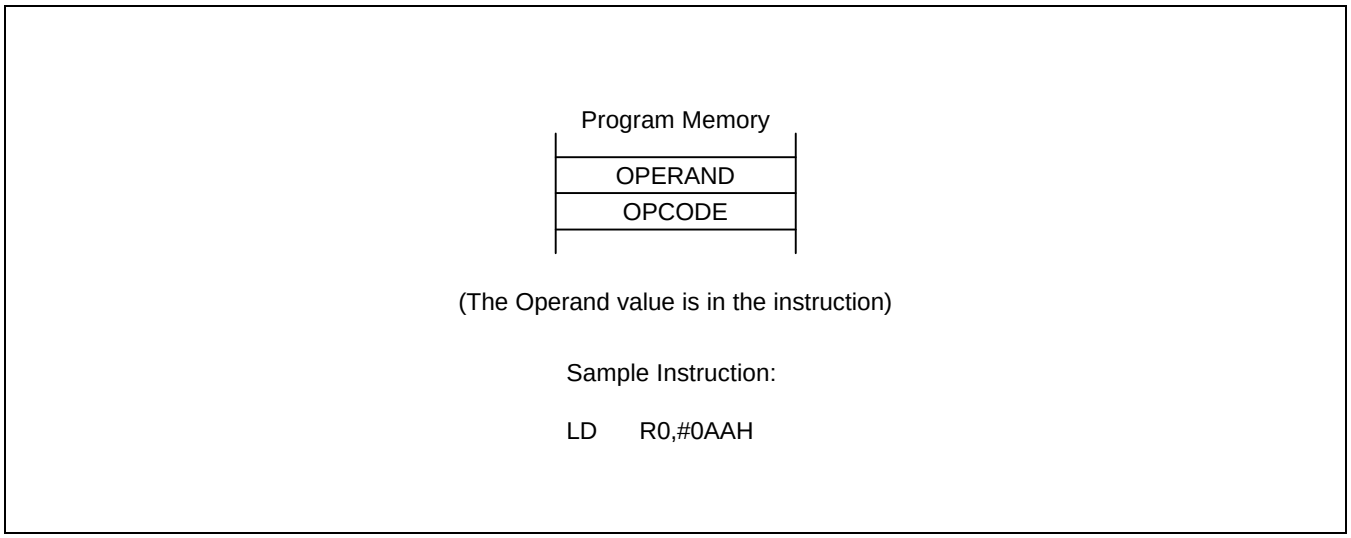


Figure 3-14. Immediate Addressing

4 CONTROL REGISTERS

OVERVIEW

In this chapter, detailed descriptions of the S3C825A control registers are presented in an easy-to-read format. You can use this chapter as a quick-reference source when writing application programs. Figure 4-1 illustrates the important features of the standard register description format.

Control register descriptions are arranged in alphabetical order according to register mnemonic. More detailed information about control registers is presented in the context of the specific peripheral hardware descriptions in Part II of this manual.

Data and counter registers are not described in detail in this reference chapter. More information about all of the registers used by a specific peripheral is presented in the corresponding peripheral descriptions in Part II of this manual.

The locations and read/write characteristics of all mapped registers in the S3C825A register file are listed in Table 4-1. The hardware reset value for each mapped register is described in Chapter 8, “RESET and Power-Down."

Table 4-1. Set 1 Registers (INTPND/STPCON/OSCCON are in bank 0 of set 1)

Register Name	Mnemonic	Address		R/W	RESET Values (bit)							
		Decimal	Hex		7	6	5	4	3	2	1	0
Interrupt pending register	INTPND	208	D0H	R/W	–	–	0	0	0	0	0	0
STOP control register	STPCON	209	D1H	R/W	0	0	0	0	0	0	0	0
Oscillator control register	OSCCON	210	D2H	R/W	–	–	–	–	0	0	–	0
Basic timer control register	BTCON	211	D3H	R/W	0	0	0	0	0	0	0	0
System clock control register	CLKCON	212	D4H	R/W	0	0	0	0	0	0	0	0
System flags register	FLAGS	213	D5H	R/W	x	x	x	x	x	x	0	0
Register pointer 0	RP0	214	D6H	R/W	1	1	0	0	0	–	–	–
Register pointer 1	RP1	215	D7H	R/W	1	1	0	0	1	–	–	–
Stack pointer (high byte)	SPH	216	D8H	R/W	x	x	x	x	x	x	x	x
Stack pointer (low byte)	SPL	217	D9H	R/W	x	x	x	x	x	x	x	x
Instruction pointer (high byte)	IPH	218	DAH	R/W	x	x	x	x	x	x	x	x
Instruction pointer (low byte)	IPL	219	DBH	R/W	x	x	x	x	x	x	x	x
Interrupt request register	IRQ	220	DCH	R	0	0	0	0	0	0	0	0
Interrupt mask register	IMR	221	DDH	R/W	x	x	x	x	x	x	x	x
System mode register	SYM	222	DEH	R/W	0	–	–	x	x	x	0	0
Register page pointer	PP	223	DFH	R/W	0	0	0	0	0	0	0	0

Table 4-2. Set 1, Bank 0 Registers

Register Name	Mnemonic	Address		R/W	RESETB Value(bit)							
		Decimal	Hex		7	6	5	4	3	2	1	0
SIO Control Register	SIOCON	224	E0H	R/W	0	0	0	0	0	0	0	0
SIO Data Register	SIODATA	225	E1H	R/W	0	0	0	0	0	0	0	0
SIO Prescaler Register	SIOPS	226	E2H	R/W	0	0	0	0	0	0	0	0
Timer 0 Counter Register	T0CNT	227	E3H	R	0	0	0	0	0	0	0	0
Timer 0 Data Register	T0DATA	228	E4H	R/W	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	229	E5H	R/W	0	0	0	0	0	0	0	0
Timer B Counter Register	TBCNT	230	E6H	R	0	0	0	0	0	0	0	0
Timer A Counter Register	TACNT	231	E7H	R	0	0	0	0	0	0	0	0
Timer B Data Register	TBDATA	232	E8H	R/W	1	1	1	1	1	1	1	1
Timer A Data Register	TADATA	233	E9H	R/W	1	1	1	1	1	1	1	1
Timer B Control Register	TBCON	234	EAH	R/W	–	–	0	0	0	0	0	0
Timer 1/A Control Register	TACON	235	EBH	R/W	0	0	0	0	0	0	0	0
Timer 2 Counter Register	T2CNT	236	ECH	R	0	0	0	0	0	0	0	0
Timer 2 Data Register	T2DATA	237	EDH	R/W	1	1	1	1	1	1	1	1
Timer 2 Control Register	T2CON	238	EEH	R/W	0	0	0	0	0	0	0	0
A/D Converter Control Register	ADCON	239	EFH	R/W	–	–	0	0	0	0	0	0
A/D Converter Data Register (high byte)	ADDATAH	240	F0H	R/W	x	x	x	x	x	x	x	x
A/D Converter Data Register (low byte)	ADDATAL	241	F1H	R/W	–	–	–	–	–	–	x	x
LCD Control Register	LCON	242	F2H	R/W	0	0	–	–	0	0	0	0
LCD Mode Register	LMOD	243	F3H	R/W	–	–	–	–	0	0	0	0
Timer 3 Counter (high byte)	T3CNTH	244	F4H	R	0	0	0	0	0	0	0	0
Timer 3 Counter (low byte)	T3CNTL	245	F5H	R	0	0	0	0	0	0	0	0
Timer 3 Data Register (high byte)	T3DATAH	246	F6H	R/W	1	1	1	1	1	1	1	1
Timer 3 Data Register (low byte)	T3DATAL	247	F7H	R/W	1	1	1	1	1	1	1	1
Timer 3 Control Register	T3CON	248	F8H	R/W	0	0	0	0	0	0	0	0
UART data register	UDATA	249	F9H	R/W	x	x	x	x	x	x	x	x
UART control register	UARTCON	250	FAH	R/W	0	0	0	0	0	0	0	0
UART Baud Rate data register	BRDATA	251	FBH	R/W	1	1	1	1	1	1	1	1
Location FCH is not mapped.												
Basic Timer Counter	BTCNT	253	FDH	R/W	0	0	0	0	0	0	0	0
Location FEH is not mapped.												
Interrupt Priority Register	IPR	255	FFH	R/W	x	x	x	x	x	x	x	x

Table 4-3. Set 1, Bank 1 Registers

Register Name	Mnemonic	Address		R/W	RESETB Value(bit)							
		Decimal	Hex		7	6	5	4	3	2	1	0
Port 2 Control Register(High Byte)	P2CONH	224	E0H	R/W	0	0	0	0	0	0	0	0
Port 2 Control Register(Low Byte)	P2CONL	225	E1H	R/W	0	0	0	0	0	0	0	0
Port 2 Pull-up Resistors enable Register	P2PUR	226	E2H	R/W	0	0	0	0	0	0	0	0
Port 2 Interrupt Control Register	P2INT	227	E3H	R/W	0	0	0	0	0	0	0	0
Port 3 Control Register(High Byte)	P3CONH	228	E4H	R/W	0	0	0	0	0	0	0	0
Port 3 Control Register(Low Byte)	P3CONL	229	E5H	R/W	0	0	0	0	0	0	0	0
Port 3 Pull-up Resistors enable Register	P3PUR	230	E6H	R/W	0	0	0	0	0	0	0	0
Port 4 Interrupt Edge Selection Register	P4EDGE	231	E7H	R/W	0	0	0	0	0	0	0	0
Port 4 Control Register(High Byte)	P4CONH	232	E8H	R/W	0	0	0	0	0	0	0	0
Port 4 Control Register(Low Byte)	P4CONL	233	E9H	R/W	0	0	0	0	0	0	0	0
Port 4 Interrupt Control Register	P4INT	234	EAH	R/W	0	0	0	0	0	0	0	0
Port 4 Interrupt Pending Register	P4PND	235	EBH	R/W	0	0	0	0	0	0	0	0
Port 5 Control Register(High Byte)	P5CONH	236	ECH	R/W	0	0	0	0	0	0	0	0
Port 5 Control Register(Low Byte)	P5CONL	237	EDH	R/W	0	0	0	0	0	0	0	0
Port 5 Pull-up Resistors enable Register	P5PUR	238	EEH	R/W	0	0	0	0	0	0	0	0
Watch timer control register	WTCON	239	EFH	R/W	0	0	0	0	0	0	0	0
Port 0 Data Register	P0	240	F0H	R/W	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	241	F1H	R/W	0	0	0	0	0	0	0	0
Port 2 Data Register	P2	242	F2H	R/W	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	243	F3H	R/W	0	0	0	0	0	0	0	0
Port 4 Data Register	P4	244	F4H	R/W	0	0	0	0	0	0	0	0
Port 5 Data Register	P5	245	F5H	R/W	0	0	0	0	0	0	0	0
Port 6 Data Register	P6	246	F6H	R/W	0	0	0	0	0	0	0	0
Port 7 Data Register	P7	247	F7H	R/W	0	0	0	0	0	0	0	0
Port 8 Data Register	P8	248	F8H	R/W	0	0	0	0	0	0	0	0
Port Group 0 Control Register	PG0CON	251	F9H	R/W	0	0	0	0	0	0	0	0
Port Group 1 Control Register	PG1CON	252	FAH	R/W	0	0	0	0	0	0	0	0
Locations FBH-FFH are not mapped.												

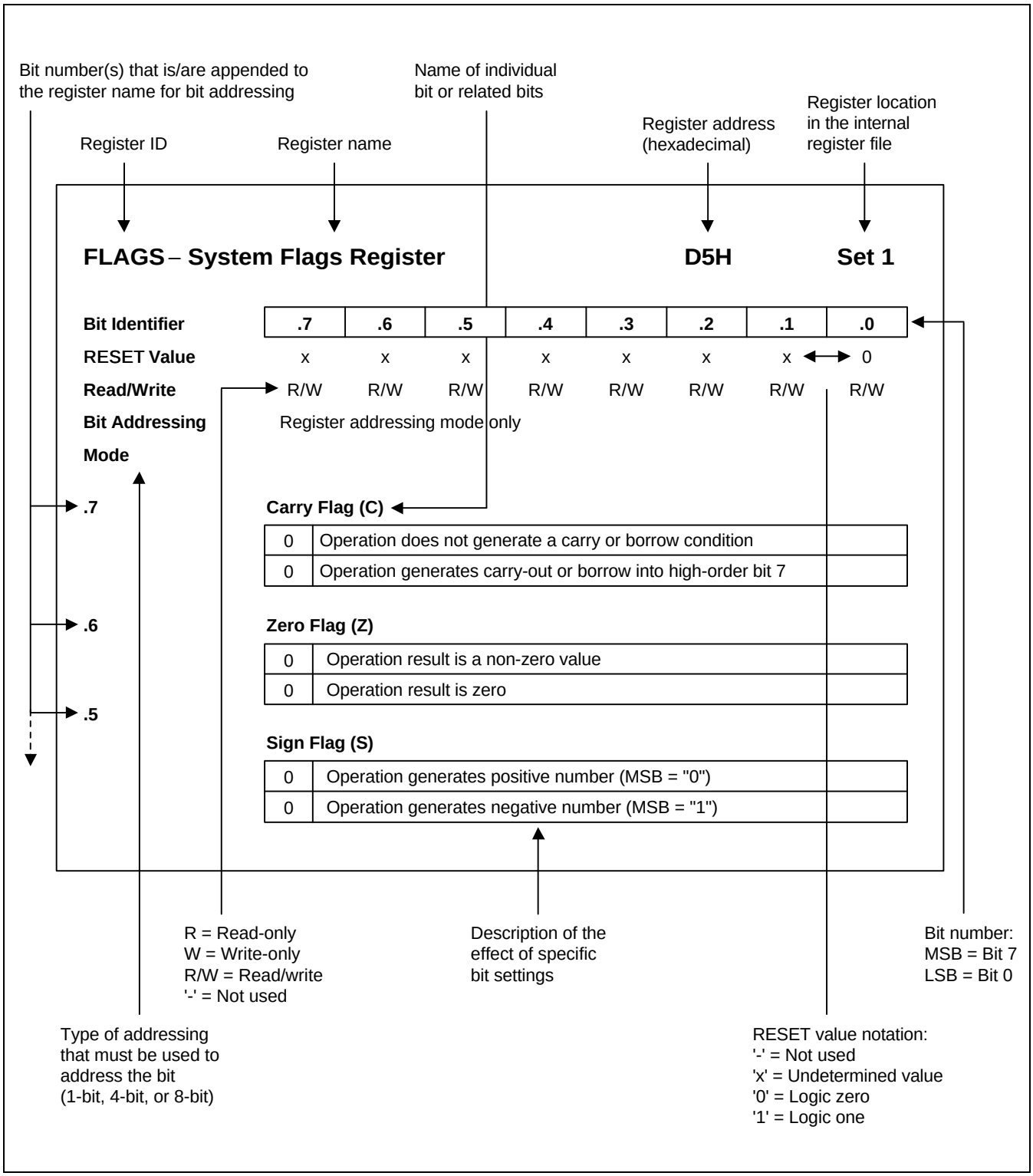


Figure 4-1. Register Description Format

ADCON — A/D Converter Control Register EFH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	0	0	0	0	0	0
Read/Write	–	–	R/W	R/W	R	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.6

Not used for the S3C825A

.5–.4 A/D Input Pin Selection Bits

0	0	AD0 (P3.0)
0	1	AD1 (P3.1)
1	0	AD2 (P3.2)
1	1	AD3 (P3.3)

.3 End-of-Conversion Bit (read-only)

0	Conversion not complete
1	Conversion complete

.2–.1 Clock Source Selection Bits

0	0	fxx/16
0	1	fxx/8
1	0	fxx/4
1	1	fxx

.0 Start or Enable Bit

0	Disable operation
1	Start operation (automatically disable operation after conversion complete).

BTCON — Basic Timer Control Register D3H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.4 Watchdog Timer Function Disable Code (for System Reset)

1	0	1	0	Disable watchdog timer function
Others				Enable watchdog timer function

.3–.2 Basic Timer Input Clock Selection Bits

0	0	fxx/4096 (3)
0	1	fxx/1024
1	0	fxx/128
1	1	fxx/16

.1 Basic Timer Counter Clear Bit (1)

0	No effect
1	Clear the basic timer counter value

.0 Clock Frequency Divider Clear Bit for all Timers (2)

0	No effect
1	Clear both clock frequency dividers

NOTES:

1. When you write a “1” to BTCON.1, the basic timer counter value is cleared to “00H”. Immediately following the write operation, the BTCON.1 value is automatically cleared to “0”.
2. When you write a “1” to BTCON.0, the corresponding frequency divider is cleared to “00H”. Immediately following the write operation, the BTCON.0 value is automatically cleared to “0”.
3. The fxx is selected clock for system.

CLKCON — System Clock Control Register D4H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	–	–	R/W	R/W	–	–	–
Addressing Mode	Register addressing mode only							

.7 Oscillator IRQ Wake-up Function Bit

0	Enable IRQ for main wake-up in power down mode
1	Disable IRQ for main wake-up in power down mode

.6–.5 Not used for the S3C825A.

.4–.3 CPU Clock (System Clock) Selection Bits (note)

0	0	fxx/16
0	1	fxx/8
1	0	fxx/2
1	1	fxx

.2–.0 Not used for the S3C825A.

NOTE: After a reset, the slowest clock (divided by 16) is selected as the system clock. To select faster clock speeds, load the appropriate values to CLKCON.3 and CLKCON.4.

FLAGS — System Flags Register D5H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Addressing Mode	Register addressing mode only							

.7 Carry Flag (C)

0	Operation does not generate a carry or borrow condition
1	Operation generates a carry-out or borrow into high-order bit 7

.6 Zero Flag (Z)

0	Operation result is a non-zero value
1	Operation result is zero

.5 Sign Flag (S)

0	Operation generates a positive number (MSB = "0")
1	Operation generates a negative number (MSB = "1")

.4 Overflow Flag (V)

0	Operation result is ≤ +127 or ≥ –128
1	Operation result is > +127 or < –128

.3 Decimal Adjust Flag (D)

0	Add operation completed
1	Subtraction operation completed

.2 Half-Carry Flag (H)

0	No carry-out of bit 3 or no borrow into bit 3 by addition or subtraction
1	Addition generated carry-out of bit 3 or subtraction generated borrow into bit 3

.1 Fast Interrupt Status Flag (FIS)

0	Interrupt return (IRET) in progress (when read)
1	Fast interrupt service routine in progress (when read)

.0 Bank Address Selection Flag (BA)

0	Bank 0 is selected
1	Bank 1 is selected

IMR — Interrupt Mask Register DDH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Interrupt Level 7 (IRQ7) Enable Bit; P4.0-P4.3

0	Disable (mask)
1	Enable (unmask)

.6 Interrupt Level 6 (IRQ6) Enable Bit; P4.4-P4.7

0	Disable (mask)
1	Enable (unmask)

.5 Interrupt Level 5 (IRQ5) Enable Bit; P2.4-P2.7

0	Disable (mask)
1	Enable (unmask)

.4 Interrupt Level 4 (IRQ4) Enable Bit; Watch Timer

0	Disable (mask)
1	Enable (unmask)

.3 Interrupt Level 3 (IRQ3) Enable Bit; SIO, UART Transmit, UART Receive

0	Disable (mask)
1	Enable (unmask)

.2 Interrupt Level 2 (IRQ2) Enable Bit;Timer 2, Timer 3 match/capture or overflow

0	Disable (mask)
1	Enable (unmask)

.1 Interrupt Level 1 (IRQ1) Enable Bit; Timer B, Timer 1/A

0	Disable (mask)
1	Enable (unmask)

.0 Interrupt Level 0 (IRQ0) Enable Bit; Timer 0 Match/Capture or Overflow

0	Disable (mask)
1	Enable (unmask)

NOTE: When an interrupt level is masked, any interrupt requests that may be issued are not recognized by the CPU.

INTPND — Interrupt Pending Register D0H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	0	0	0	0	0	0
Read/Write	–	–	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.6

Not used for the S3C825A

.5 Rx Interrupt Pending Bit (for UART)

0	Interrupt request is not pending (when read), pending bit clear (when write 0)
1	Interrupt request is pending

.4 Tx Interrupt Pending Bit (for UART)

0	Interrupt request is not pending (when read), pending bit clear (when write 0)
1	Interrupt request is pending

.3 Timer 3 Match/Capture Interrupt Pending Bit

0	Interrupt request is not pending (when read), pending bit clear (when write 0)
1	Interrupt request is pending

.2 Timer 3 Overflow Interrupt Pending bit

0	Interrupt request is not pending (when read), pending bit clear (when write 0)
1	Interrupt request is pending

.1 Timer 0 Match/Capture Interrupt Pending Bit

0	Interrupt request is not pending (when read), pending bit clear (when write 0)
1	Interrupt request is pending

.0 Timer 0 Overflow Interrupt Pending bit

0	Interrupt request is not pending (when read), pending bit clear (when write 0)
1	Interrupt request is pending

IPH — Instruction Pointer (High Byte) DAH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 Instruction Pointer Address (High Byte)

The high-byte instruction pointer value is the upper eight bits of the 16-bit instruction pointer address (IP15–IP8). The lower byte of the IP address is located in the IPL register (DBH).

IPL — Instruction Pointer (Low Byte) DBH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 Instruction Pointer Address (Low Byte)

The low-byte instruction pointer value is the lower eight bits of the 16-bit instruction pointer address (IP7–IP0). The upper byte of the IP address is located in the IPH register (DAH).

IPR — Interrupt Priority Register FFH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7, .4, and .1 **Priority Control Bits for Interrupt Groups A, B, and C (note)**

0	0	0	Group priority undefined
0	0	1	B > C > A
0	1	0	A > B > C
0	1	1	B > A > C
1	0	0	C > A > B
1	0	1	C > B > A
1	1	0	A > C > B
1	1	1	Group priority undefined

.6 **Interrupt Subgroup C Priority Control Bit**

0	IRQ6 > IRQ7
1	IRQ7 > IRQ6

.5 **Interrupt Group C Priority Control Bit**

0	IRQ5 > (IRQ6, IRQ7)
1	(IRQ6, IRQ7) > IRQ5

.3 **Interrupt Subgroup B Priority Control Bit**

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

.2 **Interrupt Group B Priority Control Bit**

0	IRQ2 > (IRQ3, IRQ4)
1	(IRQ3, IRQ4) > IRQ2

.0 **Interrupt Group A Priority Control Bit**

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

NOTE: Interrupt Group A - IRQ0, IRQ1
Interrupt Group B - IRQ2, IRQ3, IRQ4
Interrupt Group C - IRQ5, IRQ6, IRQ7

IRQ — Interrupt Request Register DCH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R	R	R	R	R	R	R	R
Addressing Mode	Register addressing mode only							

.7 Level 7 (IRQ7) Request Pending Bit; External Interrupt P4.0-P4.3

0	Not pending
1	Pending

.6 Level 6 (IRQ6) Request Pending Bit; External Interrupt P4.4-P4.7

0	Not pending
1	Pending

.5 Level 5 (IRQ5) Request Pending Bit; External Interrupt P2.4-P2.7

0	Not pending
1	Pending

.4 Level 4 (IRQ4) Request Pending Bit; Watch Timer

0	Not pending
1	Pending

.3 Level 3 (IRQ3) Request Pending Bit; SIO, UART Transmit, UART Receive

0	Not pending
1	Pending

.2 Level 2 (IRQ2) Request Pending Bit; Timer 2, Timer 3 Match/Capture or Overflow

0	Not pending
1	Pending

.1 Level 1 (IRQ1) Request Pending Bit; Timer B, Timer 1/A

0	Not pending
1	Pending

.0 Level 0 (IRQ0) Request Pending Bit; Timer 0 Match/Capture or Overflow

0	Not pending
1	Pending

LCON — LCD Control Register F2H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	–	–	0	0	0	0
Read/Write	R/W	R/W	–	–	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6 **LCD Display Control Bits**

0	0	Display off, P-Tr off
0	1	Normal display (using V_{LC1} with external voltage), P-Tr off
1	1	Normal display (using V_{LC1} with internal voltage), P-Tr on

.5 and .4

Not used for the S3C825A.

.3 and .2 **LCD Duty and Bias Selection Bits**

0	0	1/3 duty, 1/3 bias; COM0–COM2/SEG0–SEG31
0	1	1/4 duty, 1/3 bias; COM0–COM3/SEG0–SEG31
1	0	1/8 duty, 1/4 bias; COM0–COM7/SEG4–SEG31
1	1	1/8 duty, 1/5 bias; COM0–COM7/SEG4–SEG31

.1 and .0 **LCD Clock Selection Bits**

0	0	$fw/2^7$ (256 Hz when fw is 32.768 kHz)
0	1	$fw/2^6$ (512 Hz when fw is 32.768 kHz)
1	0	$fw/2^5$ (1,024 Hz when fw is 32.768 kHz)
1	1	$fw/2^4$ (2,048 Hz when fw is 32.768 kHz)

LMOD — LCD Mode Control Register F3H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	0	0	0	0
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7-.4 Not used for the S3C825A.

.3–.0 LCD Port Selection Bit

				P6.0-P6.3	P6.4-P6.7	P7.0-P7.3	P7.4-P7.7	P8.0-P8.3	P0.0-P0.3	P0.4-P0.7	P1.0-P1.3	P1.4-P1.7
				COM 0-3	COM4-7/ SEG0-3	SEG 4-7	SEG 8-11	SEG 12-15	SEG 16-19	SEG 20-23	SEG 24-27	SEG 28-31
0	0	0	0	Port	Port	Port	Port	Port	Port	Port	Port	Port
0	0	0	1	COM	COM/SEG	Port	Port	Port	Port	Port	Port	Port
0	0	1	0	COM	COM/SEG	SEG	Port	Port	Port	Port	Port	Port
0	0	1	1	COM	COM/SEG	SEG	SEG	Port	Port	Port	Port	Port
0	1	0	0	COM	COM/SEG	SEG	SEG	SEG	Port	Port	Port	Port
0	1	0	1	COM	COM/SEG	SEG	SEG	SEG	SEG	Port	Port	Port
0	1	1	0	COM	COM/SEG	SEG	SEG	SEG	SEG	SEG	Port	Port
0	1	1	1	COM	COM/SEG	SEG	SEG	SEG	SEG	SEG	SEG	Port
1	0	0	0	COM	COM/SEG	SEG	SEG	SEG	SEG	SEG	SEG	SEG

NOTE: The SEG0-SEG3 or COM4-COM7 signals are controlled by LCON.3-.2.

OSCCON — Oscillator Control Register D2H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	0	0	–	0
Read/Write	–	–	–	–	R/W	R/W	–	R/W

.7-.4

Not used for S3C825A

.3 Main Oscillator Control Bit

0	Main oscillator RUN
1	Main oscillator STOP

.2 Sub Oscillator Control Bit

0	Sub oscillator RUN
1	Sub oscillator STOP

.1

Not used for S3C825A

.0 System Clock Selection Bit

0	Select main oscillator for system clock
1	Select sub oscillator for system clock

P2CONH — Port 2 Control Register (High Byte) E0H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.6 **P2.7/INT3/TBOUT**

0	0	Input mode; interrupt on falling edge
0	1	Output mode, open-drain
1	0	Alternative function (TBOUT)
1	1	Output mode, push-pull

.5–.4 **P2.6/INT2/TAOUT**

0	0	Input mode; interrupt on falling edge
0	1	Output mode, open-drain
1	0	Alternative function (TAOUT)
1	1	Output mode, push-pull

.3–.2 **P2.5/INT1/T1CLK**

0	0	Input mode (T1CLK); interrupt on falling edge
0	1	Output mode, open-drain
1	0	Not available
1	1	Output mode, push-pull

.1–.0 **P2.4/INT0/T0CLK**

0	0	Input mode (T0CLK); interrupt on falling edge
0	1	Output mode, open-drain
1	0	Not available
1	1	Output mode, push-pull

P2CONL — Port 2 Control Register (Low Byte) E1H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.6 **P2.3/T2OUT**

0	0	Input mode
0	1	Output mode, open-drain
1	0	Alternative function (T2OUT)
1	1	Output mode, push-pull

.5–.4 **P2.2/T2CLK**

0	0	Input mode (T2CLK)
0	1	Output mode, open-drain
1	0	Not available
1	1	Output mode, push-pull

.3–.2 **P2.1**

0	0	Input mode
0	1	Output mode, open-drain
1	0	Not available
1	1	Output mode, push-pull

.1–.0 **P2.0**

0	0	Input mode
0	1	Output mode, open-drain
1	0	Not available
1	1	Output mode, push-pull

P2PUR — Port 2 Pull-up Control Register E2H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 P2.7 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.6 P2.6 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.5 P2.5 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.4 P2.4 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.3 P2.3 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.2 P2.2 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.1 P2.1 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.0 P2.0 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

P2INT — Port 2 Interrupt Control Register E3H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Port 2 Interrupt Request Pending Bit (P2.7/INT3)

0	No interrupt request pending
0	Clear pending bit (when write)
1	Interrupt request is pending

.6 Interrupt Control Settings (P2.7/INT3)

0	Disable interrupt on P2.7
1	Enable interrupt at falling edge on P2.7

.5 Port 2 Interrupt Request Pending Bit (P2.6/INT2)

0	No interrupt request pending
0	Clear pending bit (when write)
1	Interrupt request is pending

.4 Interrupt Control Settings (P2.6/INT2)

0	Disable interrupt on P2.6
1	Enable interrupt at falling edge on P2.6

.3 Port 2 Interrupt Request Pending Bit (P2.5/INT1)

0	No interrupt request pending
0	Clear pending bit (when write)
1	Interrupt request is pending

.2 Interrupt Control Settings (P2.5/INT1)

0	Disable interrupt on P2.5
1	Enable interrupt at falling edge on P2.5

.1 Port 2 Interrupt Request Pending Bit (P2.4/INT0)

0	No interrupt request pending
0	Clear pending bit (when write)
1	Interrupt request is pending

.0 Interrupt Control Settings (P2.4/INT0)

0	Disable interrupt on P2.4
1	Enable interrupt at falling edge on P2.4

P3CONH — Port 3 Control Register (High Byte) E4H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.6 P3.7/T0OUT/T0PWM/T0CAP

0	0	Input mode (T0CAP)
0	1	Output mode, open-drain
1	0	Alternative function (T0OUT/T0PWM)
1	1	Output mode, push-pull

.5–.4 P3.6/T3OUT/T3PWM/T3CAP

0	0	Input mode (T3CAP)
0	1	Output mode, open-drain
1	0	Alternative function (T3OUT/T3PWM)
1	1	Output mode, push-pull

.3–.2 P3.5/T3CLK

0	0	Input mode(T3CLK)
0	1	Output mode, open-drain
1	0	Not available
1	1	Output mode, push-pull

.1–.0 P3.4

0	0	Input mode
0	1	Output mode, open-drain
1	0	Not available
1	1	Output mode, push-pull

P3CONL — Port 3 Control Register (Low Byte) E5H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.6 **P3.3/AD3**

0	0	Input mode
0	1	Output mode, open-drain
1	0	Alternative function (ADC mode)
1	1	Output mode, push-pull

.5–.4 **P3.2/AD2**

0	0	Input mode
0	1	Output mode, open-drain
1	0	Alternative function (ADC mode)
1	1	Output mode, push-pull

.3–.2 **P3.1/AD1**

0	0	Input mode
0	1	Output mode, open-drain
1	0	Alternative function (ADC mode)
1	1	Output mode, push-pull

.1–.0 **P3.0/AD0**

0	0	Input mode
0	1	Output mode, open-drain
1	0	Alternative function (ADC mode)
1	1	Output mode, push-pull

P3PUR — Port 3 Pull-up Control Register E6H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 P3.7 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.6 P3.6 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.5 P3.5 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.4 P3.4 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.3 P3.3 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.2 P3.2 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.1 P3.1 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.0 P3.0 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

P4CONH — Port 4 Control Register (High Byte) E8H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6 **P4.7/INT11 Mode Selection Bits**

0	0	Input mode
0	1	Input, pull-up mode
1	0	Open-drain output mode
1	1	Push-pull output mode

.5 and .4 **P4.6/INT10 Mode Selection Bits**

0	0	Input mode
0	1	Input, pull-up mode
1	0	Open-drain output mode
1	1	Push-pull output mode

.3 and .2 **P4.5/INT9 Mode Selection Bits**

0	0	Input mode
0	1	Input, pull-up mode
1	0	Open-drain output mode
1	1	Push-pull output mode

.1 and .0 **P4.4/INT8 Mode Selection Bits**

0	0	Input mode
0	1	Input, pull-up mode
1	0	Open-drain output mode
1	1	Push-pull output mode

NOTE: Pins configured as input can be used as interrupt input with noise filter.

P4CONL — Port 4 Control Register (Low Byte) E9H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6 **P4.3/INT7 Mode Selection Bits**

0	0	Input mode
0	1	Input, pull-up mode
1	0	Open-drain output mode
1	1	Push-pull output mode

.5 and .4 **P4.2/INT6 Mode Selection Bits**

0	0	Input mode
0	1	Input, pull-up mode
1	0	Open-drain output mode
1	1	Push-pull output mode

.3 and .2 **P4.1/INT5 Mode Selection Bits**

0	0	Input mode
0	1	Input, pull-up mode
1	0	Open-drain output mode
1	1	Push-pull output mode

.1 and .0 **P4.0/INT4 Mode Selection Bits**

0	0	Input mode
0	1	Input, pull-up mode
1	0	Open-drain output mode
1	1	Push-pull output mode

NOTE: Pins configured as input can be used as interrupt input with noise filter.

P4EDGE — Port 4 Interrupt Edge Selection Register E7H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 P4.7 External Interrupt (INT11) State Bit

0	Falling edge detection
1	Rising edge detection

.6 P4.6 External Interrupt (INT10) State Bit

0	Falling edge detection
1	Rising edge detection

.5 P4.5 External Interrupt (INT9) State Bit

0	Falling edge detection
1	Rising edge detection

.4 P4.4 External Interrupt (INT8) State Bit

0	Falling edge detection
1	Rising edge detection

.3 P4.3 External Interrupt (INT7) State Bit

0	Falling edge detection
1	Rising edge detection

.2 P4.2 External Interrupt (INT6) State Bit

0	Falling edge detection
1	Rising edge detection

.1 P4.1 External Interrupt (INT5) State Bit

0	Falling edge detection
1	Rising edge detection

.0 P4.0 External Interrupt (INT4) State Bit

0	Falling edge detection
1	Rising edge detection

P4INT — Port 4 Interrupt Control Register EAH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 P4.7 External Interrupt (INT11) Enable Bit

0	Disable interrupt
1	Enable interrupt

.6 P4.6 External Interrupt (INT10) Enable Bit

0	Disable interrupt
1	Enable interrupt

.5 P4.5 External Interrupt (INT9) Enable Bit

0	Disable interrupt
1	Enable interrupt

.4 P4.4 External Interrupt (INT8) Enable Bit

0	Disable interrupt
1	Enable interrupt

.3 P4.3 External Interrupt (INT7) Enable Bit

0	Disable interrupt
1	Enable interrupt

.2 P4.2 External Interrupt (INT6) Enable Bit

0	Disable interrupt
1	Enable interrupt

.1 P4.1 External Interrupt (INT5) Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 P4.0 External Interrupt (INT4) Enable Bit

0	Disable interrupt
1	Enable interrupt

P4PND — Port 4 Interrupt Pending Register EBH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 P4.7 External Interrupt (INT11) Pending Flag

0	No interrupt request pending (When read), Clear pending bit (when write)
1	P4.7 interrupt request is pending (when read)

.6 P4.6 External Interrupt (INT10) Pending Flag

0	No interrupt request pending (When read), Clear pending bit (when write)
1	P4.6 interrupt request is pending (when read)

.5 P4.5 External Interrupt (INT9) Pending Flag

0	No interrupt request pending (When read), Clear pending bit (when write)
1	P4.5 interrupt request is pending (when read)

.4 P4.4 External Interrupt (INT8) Pending Flag

0	No interrupt request pending (When read), Clear pending bit (when write)
1	P4.4 interrupt request is pending (when read)

.3 P4.3 External Interrupt (INT7) Pending Flag

0	No interrupt request pending (When read), Clear pending bit (when write)
1	P4.3 interrupt request is pending (when read)

.2 P4.2 External Interrupt (INT6) Pending Flag

0	No interrupt request pending (When read), Clear pending bit (when write)
1	P4.2 interrupt request is pending (when read)

.1 P4.1 External Interrupt (INT5) Pending Flag

0	No interrupt request pending (When read), Clear pending bit (when write)
1	P4.1 interrupt request is pending (when read)

.0 P4.0 External Interrupt (INT4) Pending Flag

0	No interrupt request pending (When read), Clear pending bit (when write)
1	P4.0 interrupt request is pending (when read)

NOTE: Writing a “1” to an interrupt pending flag (P4PND.0–.7) has no effect.

P5CONH — Port 5 Control Register (High Byte) ECH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 TXD output control bit

0	Disable TXD output at P5.5
1	Enable TXD output at P5.5

.6 RXD output control bit

0	Disable RXD output at P5.4
1	Enable RXD output at P5.4

.5–.4 P5.6

0	0	Input mode
0	1	Output mode, open-drain
1	0	Not available
1	1	Output mode, push-pull

.3–.2 P5.5/TXD

0	0	Input mode
0	1	Output mode, open-drain (TXD output depends on ***P5CONH.7***)
1	0	Not available
1	1	Output mode, push-pull (TXD output depends on ***P5CONH.7***)

.1–.0 P5.4/RXD

0	0	Input mode (RXD)
0	1	Output mode, open-drain (RXD output depends on ***P5CONH.6***)
1	0	Not available
1	1	Output mode, push-pull (RXD output depends on ***P5CONH.6***)

P5CONL — Port 5 Control Register (Low Byte) EDH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.6 **P5.3/BUZ**

0	0	Input mode
0	1	Output mode, open-drain
1	0	Alternative function (BUZ)
1	1	Output mode, push-pull

.5–.4 **P5.2/SO**

0	0	Input mode
0	1	Output mode, open-drain
1	0	Alternative function (SO)
1	1	Output mode, push-pull

.3–.2 **P5.1/SI**

0	0	Input mode (SI)
0	1	Output mode, open-drain
1	0	Not available
1	1	Output mode, push-pull

.1–.0 **P5.0/SCK**

0	0	Input mode (SCK)
0	1	Output mode, open-drain
1	0	Alternative function (SCK out)
1	1	Output mode, push-pull

P5PUR — Port 5 Pull-up Control Register EEH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Not used for the S3C825A

.6 P5.6 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.5 P5.5 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.4 P5.4 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.3 P5.3 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.2 P5.2 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.1 P5.1 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

.0 P5.0 Pull-up Resistor Enable Bit

0	Pull-up disable
1	Pull-up enable

PG0CON — Port Group 0 Control Register F9H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.6 P1.4-1.7/SEG28-31 Mode Selection Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Open-drain output mode
1	1	Push-pull output mode

.5–.4 P1.0-1.3/SEG24-27 Mode Selection Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Open-drain output mode
1	1	Push-pull output mode

.3–.2 P0.4-0.7/SEG20-23 Mode Selection Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Open-drain output mode
1	1	Push-pull output mode

.1–.0 P0.0-0.3/SEG16-19 Mode Selection Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Open-drain output mode
1	1	Push-pull output mode

PG1CON — Port Group 1 Control Register FAH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.6 P8.0-P8.3/SEG12-15 Mode Selection Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Open-drain output mode
1	1	Push-pull output mode

.5–.4 P7.4-P7.7/SEG8-11 Mode Selection Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Open-drain output mode
1	1	Push-pull output mode

.3–.2 P7.0-P7.3/SEG4-7 Mode Selection Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Open-drain output mode
1	1	Push-pull output mode

.1–.0 P6.0-P6.3/COM0-3 and P6.4-P6.7/COM4-7/SEG0-3 Mode Selection Bits

0	0	Input mode
0	1	Input mode, pull-up
1	0	Open-drain output mode
1	1	Push-pull output mode

PP — Register Page Pointer DFH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.4 Destination Register Page Selection Bits

0	0	0	0	Destination: page 0
0	0	0	1	Destination: page 1
0	0	1	0	Destination: page 2
0	0	1	1	Destination: page 3
0	1	0	0	Destination: page 4
0	1	0	1	Destination: page 5
0	1	1	0	Destination: page 6
0	1	1	1	Destination: page 7

.3 – .0 Source Register Page Selection Bits

0	0	0	0	Source: page 0
0	0	0	1	Source: page 1
0	0	1	0	Source: page 2
0	0	1	1	Source: page 3
0	1	0	0	Source: page 4
0	1	0	1	Source: page 5
0	1	1	0	Source: page 6
0	1	1	1	Source: page 7

NOTE: In the S3C825A microcontroller, the internal register file is configured as eight pages (Pages 0-7). The pages 0-6 are used for general purpose register file, and page 4 is used for LCD data register or general purpose registers.

RP0 — Register Pointer 0 D6H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	1	1	0	0	0	–	–	–
Read/Write	R/W	R/W	R/W	R/W	R/W	–	–	–
Addressing Mode	Register addressing only							

.7–.3 **Register Pointer 0 Address Value**

Register pointer 0 can independently point to one of the 256-byte working register areas in the register file. Using the register pointers RP0 and RP1, you can select two 8-byte register slices at one time as active working register space. After a reset, RP0 points to address C0H in register set 1, selecting the 8-byte working register slice C0H–C7H.

.2–.0 Not used for the S3C825A

RP1 — Register Pointer 1 D7H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	1	1	0	0	1	–	–	–
Read/Write	R/W	R/W	R/W	R/W	R/W	–	–	–
Addressing Mode	Register addressing only							

.7 – .3 **Register Pointer 1 Address Value**

Register pointer 1 can independently point to one of the 256-byte working register areas in the register file. Using the register pointers RP0 and RP1, you can select two 8-byte register slices at one time as active working register space. After a reset, RP1 points to address C8H in register set 1, selecting the 8-byte working register slice C8H–CFH.

.2 – .0 Not used for the S3C825A

SIOCON — SIO Control Register E0H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 SIO Shift Clock Selection Bit

0	Internal clock (P.S clock)
1	External clock (SCK)

.6 Data Direction Control Bit

0	MSB-first mode
1	LSB-first mode

.5 SIO Mode Selection Bit

0	Receive-only mode
1	Transmit/receive mode

.4 Shift Clock Edge Selection Bit

0	Tx at falling edges, Rx at rising edges
1	Tx at rising edges, Rx at falling edges

.3 SIO Counter Clear and Shift Start Bit

0	No action
1	Clear 3-bit counter and start shifting

.2 SIO Shift Operation Enable Bit

0	Disable shifter and clock counter
1	Enable shifter and clock counter

.1 SIO Interrupt Enable Bit

0	Disable SIO Interrupt
1	Enable SIO Interrupt

.0 SIO Interrupt Pending Bit

0	No interrupt pending
0	Clear pending condition (when write)
1	Interrupt is pending

SPH — Stack Pointer (High Byte) D8H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 Stack Pointer Address (High Byte)

The high-byte stack pointer value is the upper eight bits of the 16-bit stack pointer address (SP15–SP8). The lower byte of the stack pointer value is located in register SPL (D9H). The SP value is undefined following a reset.

SPL — Stack Pointer (Low Byte) D9H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 Stack Pointer Address (Low Byte)

The low-byte stack pointer value is the lower eight bits of the 16-bit stack pointer address (SP7–SP0). The upper byte of the stack pointer value is located in register SPH (D8H). The SP value is undefined following a reset.

STPCON — Stop Control Register D1H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 STOP Control Bits

1 0 1 0 0 1 0 1	Enable stop instruction
Other values	Disable stop instruction

NOTE: Before execute the STOP instruction, set this STPCON register as “10100101b”. Otherwise the STOP instruction will not execute as well as reset will be generated.

SYM — System Mode Register DEH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	–	–	x	x	x	0	0
Read/Write	R/W	–	–	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Not used, But you must keep "0"

.6–.5 Not used for the S3C825A

.4–.2 **Fast Interrupt Level Selection Bits (1)**

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

.1 **Fast Interrupt Enable Bit (2)**

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0 **Global Interrupt Enable Bit (3)**

0	Disable all interrupt processing
1	Enable all interrupt processing

NOTES:
1. You can select only one interrupt level at a time for fast interrupt processing.
2. Setting SYM.1 to "1" enables fast interrupt processing for the interrupt level currently selected by SYM.2-SYM.4.
3. Following a reset, you must enable global interrupt processing by executing an EI instruction (not by writing a "1" to SYM.0).

T0CON — Timer 0 Control Register E5H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.5 Timer 0 Input Clock Selection Bits

0	0	0	fxx/1024
0	0	1	fxx/256
0	1	0	fxx/64
0	1	1	fxx/8
1	0	0	fxx
1	0	1	External clock (T0CLK) falling edge
1	1	0	External clock (T0CLK) rising edge
1	1	1	Counter stop

.4–.3 Timer 0 Operating Mode Selection Bits

0	0	Interval mode
0	1	Capture mode (capture on rising edge, counter running, OVF can occur)
1	0	Capture mode (capture on falling edge, counter running, OVF can occur)
1	1	PWM mode (OVF & match interrupt can occur)

.2 Timer 0 Counter Clear Bit (note)

0	No effect
1	Clear the timer 0 counter (when write)

.1 Timer 0 Match/Capture Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer 0 Overflow Interrupt Enable

0	Disable overflow interrupt
1	Enable overflow interrupt

NOTE: When you write a "1" to T0CON.2, the timer 0 counter value is cleared to "00H". Immediately following the write operation, the T0CON.2 value is automatically cleared to "0".

TACON — Timer 1/A Control Register EBH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Timer 1 Operating Mode Selection Bit

0	Two 8-bit timers mode (Timer A/B)
1	One 16-bit timer mode (Timer 1)

.6–.4 Timer 1/A Clock Selection Bits

0	0	0	fxx/256
0	0	1	fxx/64
0	1	0	fxx/8
0	1	1	fxx
1	1	1	External clock (T1CLK) rising edge

.3 Timer 1/A Counter Clear Bit (NOTE)

0	No effect
1	Clear the timer 1/A counter (when write)

.2 Timer 1/A Counter Run Enable Bit

0	Disable Counter Running
1	Enable Counter Running

.1 Timer 1/A Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer 1/A Interrupt Pending Bit

0	No interrupt pending (when read)
0	Clear pending bit (when write)
1	Interrupt is pending (when read)

NOTE: When you write a "1" to TACON.3, the Timer 1/A counter value is cleared to "00H". Immediately following the write operation, the TACON.3 value is automatically cleared to "0".

TBCON — Timer B Control Register EAH Set 1, Bank0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	0	0	0	0	0	0
Read/Write	–	–	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6

Not used for the S3C825A

.5 and .4 Timer B Clock Selection Bits

0	0	fxx/256
0	1	fxx/64
1	0	fxx/8
1	1	fxx

.3 Timer B Counter Clear Bit (NOTE)

0	No effect
1	Clear the timer B counter (when write)

.2 Timer B Counter Run Enable Bit

0	Disable Counter Running
1	Enable Counter Running

.1 Timer B Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer B Interrupt Pending Bit

0	No interrupt pending (when read)
0	Clear pending bit (when write)
1	Interrupt is pending (when read)

NOTE: When you write a "1" to TBCON.3, the Timer B counter value is cleared to "00H". Immediately following the write operation, the TBCON.3 value is automatically cleared to "0".

T2CON — Timer 2 Control Register EEH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.5 Timer 2 Input Clock Selection Bits

0	0	0	fxx/256
0	0	1	fxx/64
0	1	0	fxx/8
0	1	1	fxx
1	1	1	External clock (T2CLK) input

.4

Not used for the S3C825A

.3 Timer 2 Counter Clear Bit (Note)

0	No effect
1	Clear the timer 2 counter (when write)

.2 Timer 2 Counter Enable Bit

0	Disable counting operation
1	Enable counting operation

.1 Timer 2 Interrupt Enable Bit

0	Disable timer 2 interrupt
1	Enable timer 2 interrupt

.0 Timer 2 Interrupt Pending Bit

0	No timer 2 interrupt pending (when read)
0	*Clear timer 2 interrupt pending bit (when write)*
1	T2 interrupt is pending

NOTE: When you write a "1" to T2CON.3, the timer 2 counter value is cleared to "00H". Immediately following the write operation, the T2CON.3 value is automatically cleared to "0".

T3CON — Timer 3 Control Register F8H Set 1, Bank0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.5 Timer 3 Input Clock Selection Bits

0	0	0	fxx/1024
0	0	1	fxx/256
0	1	0	fxx/64
0	1	1	fxx/8
1	0	0	fxx
1	0	1	External clock (T3CLK) falling edge
1	1	0	External clock (T3CLK) rising edge
1	1	1	Counter stop

.4 - .3 Timer 3 Operating Mode Selection Bits

0	0	Interval mode
0	1	Capture mode (capture on rising edge, counter running, OVF can occur)
1	0	Capture mode (capture on falling edge, counter running, OVF can occur)
1	1	PWM mode (OVF & match interrupt can occur)

.2 Timer 3 Counter Clear Bit (NOTE)

0	No effect
1	Clear the timer 3 counter (when write)

.1 Timer 3 match/capture interrupt enable bit

0	Disable interrupt
1	Enable interrupt

.0 Timer 3 overflow interrupt enable

0	Disable overflow interrupt
1	Enable overflow interrupt

NOTE: When you write a "1" T3CON.2, the timer 3 counter value is cleared to "00H". Immediately following the write operation, the T3CON.2 value is automatically cleared to "0".

UARTCON — UART Control Register FAH Set 1, Bank0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 – .6

0	0	Mode 0: shift register (f_{OSC}/(16 × (BRDATA + 1))
0	1	Mode 1: 8-Bit UART (f_{OSC}/(16 × (BRDATA + 1))
1	0	Mode 2: 9-Bit UART f_{OSC}/16
1	1	Mode 3: 9-Bit UART (f_{OSC}/(16 × (BRDATA + 1))

.5 Multiprocessor Communication Enable Bit (for modes 2 and 3 only)

0	Disable
1	Enable

.4 Serial Data Receive Enable Bit

0	Disable
1	Enable

.3 TB8

Location of the 9th data bit to be transmitted in UART mode 2 or 3 (“0” or “1”)

.2 RB8

Location of the 9th data bit to be received in UART mode 2 or 3 (“0” or “1”)

.1 Receive Interrupt Enable Bit

0	Disable Rx interrupt
1	Enable Rx interrupt

.0 Transmit Interrupt Enable Bit

0	Disable Tx interrupt
1	Enable Tx interrupt

NOTES:

1. In mode 2 or 3, if the MCE bit is set to "1" then the receive interrupt will not be activated if the received 9th data bit "0". In mode 1, if MCE = "1" the receive interrupt will not be activated if a valid stop bit was not received. In mode 0, the MCE bit should be “0”.
2. The descriptions for 8-bit and 9-bit UART mode do not include start and stop bits for serial data receive and transmit.
3. Rx / Tx interrupt pending bits are in INTPND register.

WTCON — Watch Timer Control Register EFH Set 1, Bank1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7 Watch Timer Clock Selection Bit

0	Select main clock divided by 2^7 (fx/128)
1	Select sub clock(fxt)

.6 Watch Timer Interrupt Enable Bit

0	Disable watch timer interrupt
1	Enable watch timer interrupt

.5-.4 Buzzer Signal Selection Bits

0	0	0.5 kHz
0	1	1 kHz
1	0	2 kHz
1	1	4 kHz

.3-.2 Watch Timer Speed Selection Bits

0	0	Set watch timer interrupt to 1s
0	1	Set watch timer interrupt to 0.5s
1	0	Set watch timer interrupt to 0.25s
1	1	Set watch timer interrupt to 3.91ms

.1 Watch Timer Enable Bit

0	Disable watch timer; Clear frequency dividing circuits
1	Enable watch timer

.0 Watch Timer Interrupt Pending Bit

0	No interrupt pending (when read)
0	Clear pending bit (when write)
1	Interrupt is pending (when read)

5 INTERRUPT STRUCTURE

OVERVIEW

The S3C8-series interrupt structure has three basic components: levels, vectors, and sources. The SAM88RC CPU recognizes up to eight interrupt levels and supports up to 128 interrupt vectors. When a specific interrupt level has more than one vector address, the vector priorities are established in hardware. A vector address can be assigned to one or more sources.

Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight possible interrupt levels: IRQ0–IRQ7, also called level 0–level 7. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. The S3C825A interrupt structure recognizes eight interrupt levels.

The interrupt level numbers 0 through 7 do not necessarily indicate the relative priority of the levels. They are just identifiers for the interrupt levels that are recognized by the CPU. The relative priority of different interrupt levels is determined by settings in the interrupt priority register, IPR. Interrupt group and subgroup logic controlled by IPR settings lets you define more complex priority relationships between different levels.

Vectors

Each interrupt level can have one or more interrupt vectors, or it may have no vector address assigned at all. The maximum number of vectors that can be supported for a given level is 128 (The actual number of vectors used for S3C8-series devices is always much smaller). If an interrupt level has more than one vector address, the vector priorities are set in hardware. S3C825A uses twenty three vectors.

Sources

A source is any peripheral that generates an interrupt. A source can be an external pin or a counter overflow. Each vector can have several interrupt sources. In the S3C825A interrupt structure, there are twenty three possible interrupt sources.

When a service routine starts, the respective pending bit should be either cleared automatically by hardware or cleared "manually" by program software. The characteristics of the source's pending mechanism determine which method would be used to clear its respective pending bit.

INTERRUPT TYPES

The three components of the S3C8 interrupt structure described before — levels, vectors, and sources — are combined to determine the interrupt structure of an individual device and to make full use of its available interrupt logic. There are three possible combinations of interrupt structure components, called interrupt types 1, 2, and 3. The types differ in the number of vectors and interrupt sources assigned to each level (see Figure 5-1):

Type 1: One level (IRQn) + one vector (V_1) + one source (S_1)

Type 2: One level (IRQn) + one vector (V_1) + multiple sources ($S_1 – S_n$)

Type 3: One level (IRQn) + multiple vectors ($V_1 – V_n$) + multiple sources ($S_1 – S_n$, $S_{n+1} – S_{n+m}$)

In the S3C825A microcontroller, two interrupt types are implemented.

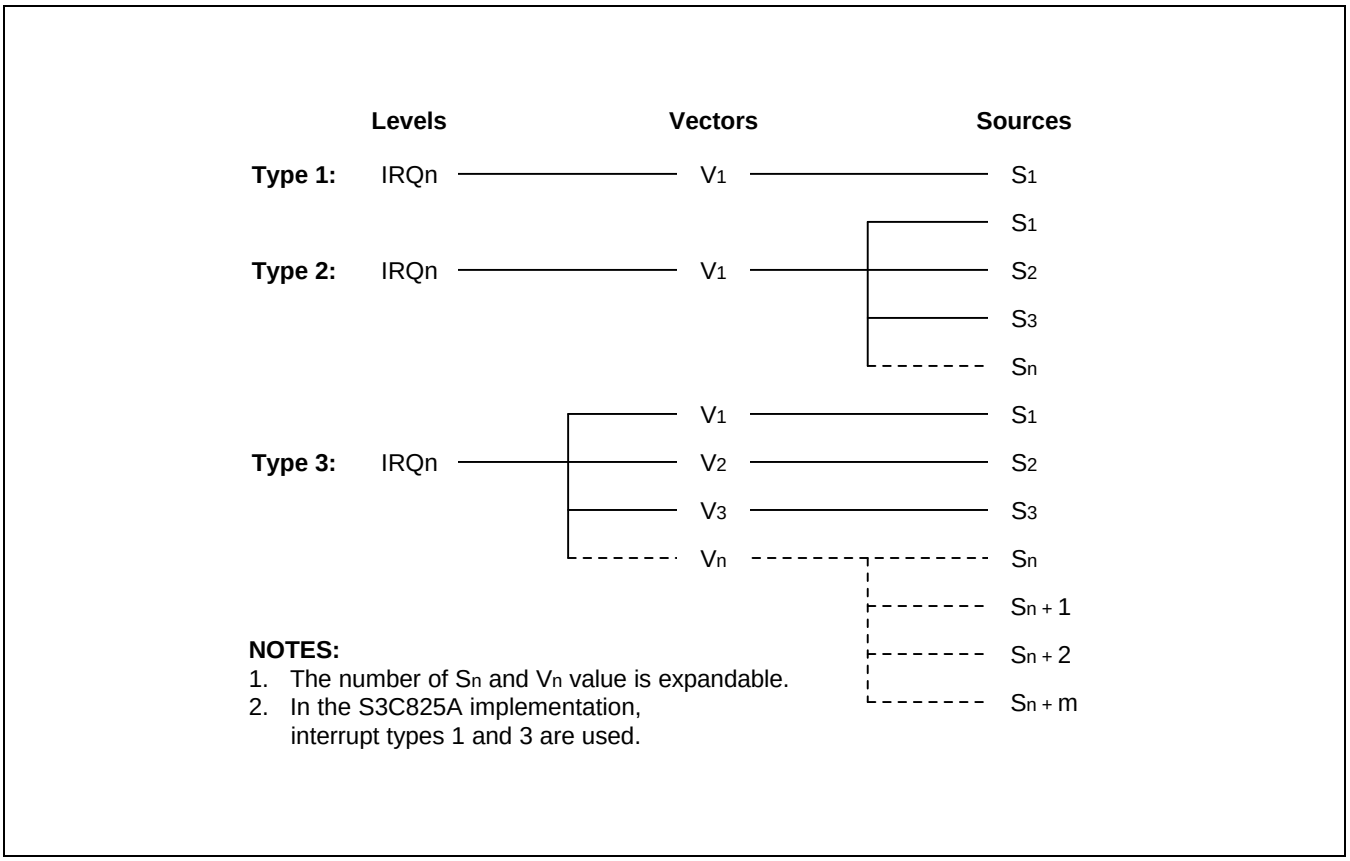


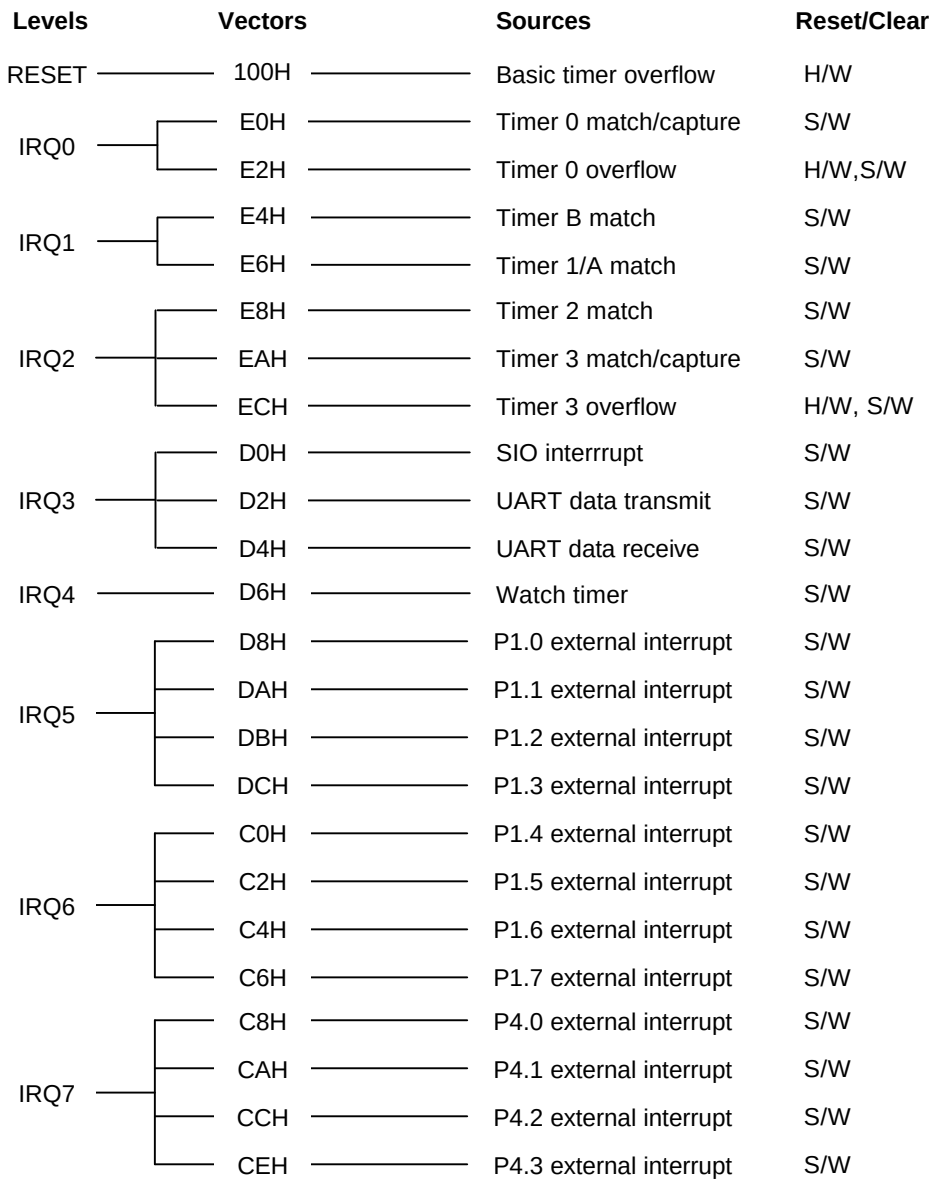
Figure 5-1. S3C8-Series Interrupt Types

S3C825A INTERRUPT STRUCTURE

The S3C825A microcontroller supports twenty three interrupt sources. All twenty three of the interrupt sources have a corresponding interrupt vector address. Eight interrupt levels are recognized by the CPU in this device-specific interrupt structure, as shown in Figure 5-2.

When multiple interrupt levels are active, the interrupt priority register (IPR) determines the order in which contending interrupts are to be serviced. If multiple interrupts occur within the same interrupt level, the interrupt with the lowest vector address is usually processed first (The relative priorities of multiple interrupts within a single level are fixed in hardware).

When the CPU grants an interrupt request, interrupt processing starts. All other interrupts are disabled and the program counter value and status flags are pushed to stack. The starting address of the service routine is fetched from the appropriate vector address (plus the next 8-bit value to concatenate the full 16-bit address) and the service routine is executed.



NOTES:

1. Within a given interrupt level, the low vector address has high priority. For example, E0H has higher priority than E2H within the level IRQ0 the priorities within each level are set at the factory.
2. External interrupts are triggered by a rising or falling edge, depending on the corresponding control register setting.

Figure 5-2. S3C825A Interrupt Structure

INTERRUPT VECTOR ADDRESSES

All interrupt vector addresses for the S3C825A interrupt structure are stored in the vector address area of the first 256 bytes of the program memory (ROM).

You can allocate unused locations in the vector address area as normal program memory. If you do so, please be careful not to overwrite any of the stored vector addresses (Table 5-1 lists all vector addresses).

The program reset address in the ROM is 0100H.

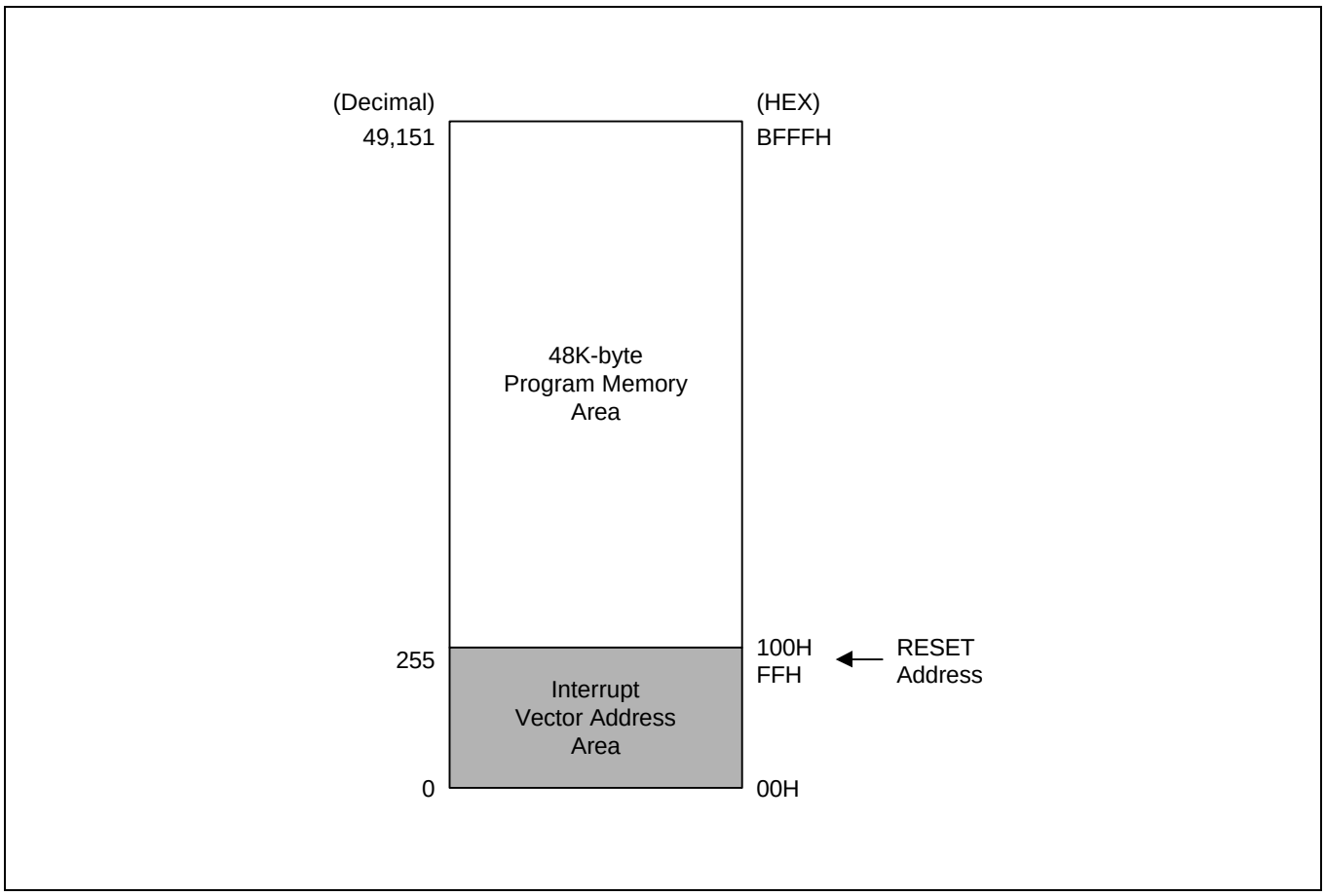


Figure 5-3. ROM Vector Address Area

Table 5-1. Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
256	100H	Basic timer overflow	RESET	–	√	
226	E2H	Timer 0 overflow	IRQ0	1	√	√
224	E0H	Timer 0 match/capture		0		√
230	E6H	Timer 1/A match	IRQ1	1		√
228	E4H	Timer B match		0		√
236	ECH	Timer 3 overflow	IRQ2	2	√	√
234	EAH	Timer 3 match/capture		1		√
232	E8H	Timer 2 match		0		√
212	D4H	UART data receive	IRQ3	2		√
210	D2H	UART data transmit		1		√
208	D0H	SIO interrupt		0		√
214	D6H	Watch timer	IRQ4	–		√
222	DEH	P2.7 external interrupt	IRQ5	3		√
220	DCH	P2.6 external interrupt		2		√
218	DAH	P2.5 external interrupt		1		√
216	D8H	P2.4 external interrupt		0		√
198	C6H	P4.7 external interrupt	IRQ6	3		√
196	C4H	P4.6 external interrupt		2		√
194	C2H	P4.5 external interrupt		1		√
192	C0H	P4.4 external interrupt		0		√
206	CEH	P4.3 external interrupt	IRQ7	3		√
204	CCH	P4.2 external interrupt		2		√
202	CAH	P4.1 external interrupt		1		√
200	C8H	P4.0 external interrupt		0		√

NOTES:

1. Interrupt priorities are identified in inverse order: "0" is the highest priority, "1" is the next highest, and so on.
2. If two or more interrupts within the same level contend, the interrupt with the lowest vector address usually has priority over one with a higher vector address. The priorities within a given level are fixed in hardware.

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

Executing the Enable Interrupts (EI) instruction globally enables the interrupt structure. All interrupts are then serviced as they occur according to the established priorities.

NOTE

The system initialization routine executed after a reset must always contain an EI instruction to globally enable the interrupt structure.

During the normal operation, you can execute the DI (Disable Interrupt) instruction at any time to globally disable interrupt processing. The EI and DI instructions change the value of bit 0 in the SYM register.

SYSTEM-LEVEL INTERRUPT CONTROL REGISTERS

In addition to the control registers for specific interrupt sources, four system-level registers control interrupt processing:

— The interrupt mask register, IMR, enables (un-masks) or disables (masks) interrupt levels.

— The interrupt priority register, IPR, controls the relative priorities of interrupt levels.

— The interrupt request register, IRQ, contains interrupt pending flags for each interrupt level (as opposed to each interrupt source).

— The system mode register, SYM, enables or disables global interrupt processing (SYM settings also enable fast interrupts and control the activity of external interface, if implemented).

Table 5-2. Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable or disable interrupt processing for each of the eight interrupt levels: IRQ0–IRQ7.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The eight levels of S3C825A are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, group B is IRQ2, IRQ3 and IRQ4, and group C is IRQ5, IRQ6, and IRQ7.
Interrupt request register	IRQ	R	This register contains a request pending bit for each interrupt level.
System mode register	SYM	R/W	This register enables/disables fast interrupt processing, and dynamic global interrupt processing.

NOTE: Before IMR register is changed to any value, all interrupts must be disable. Using DI instruction is recommended.

INTERRUPT PROCESSING CONTROL POINTS

Interrupt processing can therefore be controlled in two ways: globally or by specific interrupt level and source. The system-level control points in the interrupt structure are:

— Global interrupt enable and disable (by EI and DI instructions or by direct manipulation of SYM.0)

— Interrupt level enable/disable settings (IMR register)

— Interrupt level priority settings (IPR register)

— Interrupt source enable/disable settings in the corresponding peripheral control registers

NOTE

When writing an application program that handles interrupt processing, be sure to include the necessary register file address (register pointer) information.

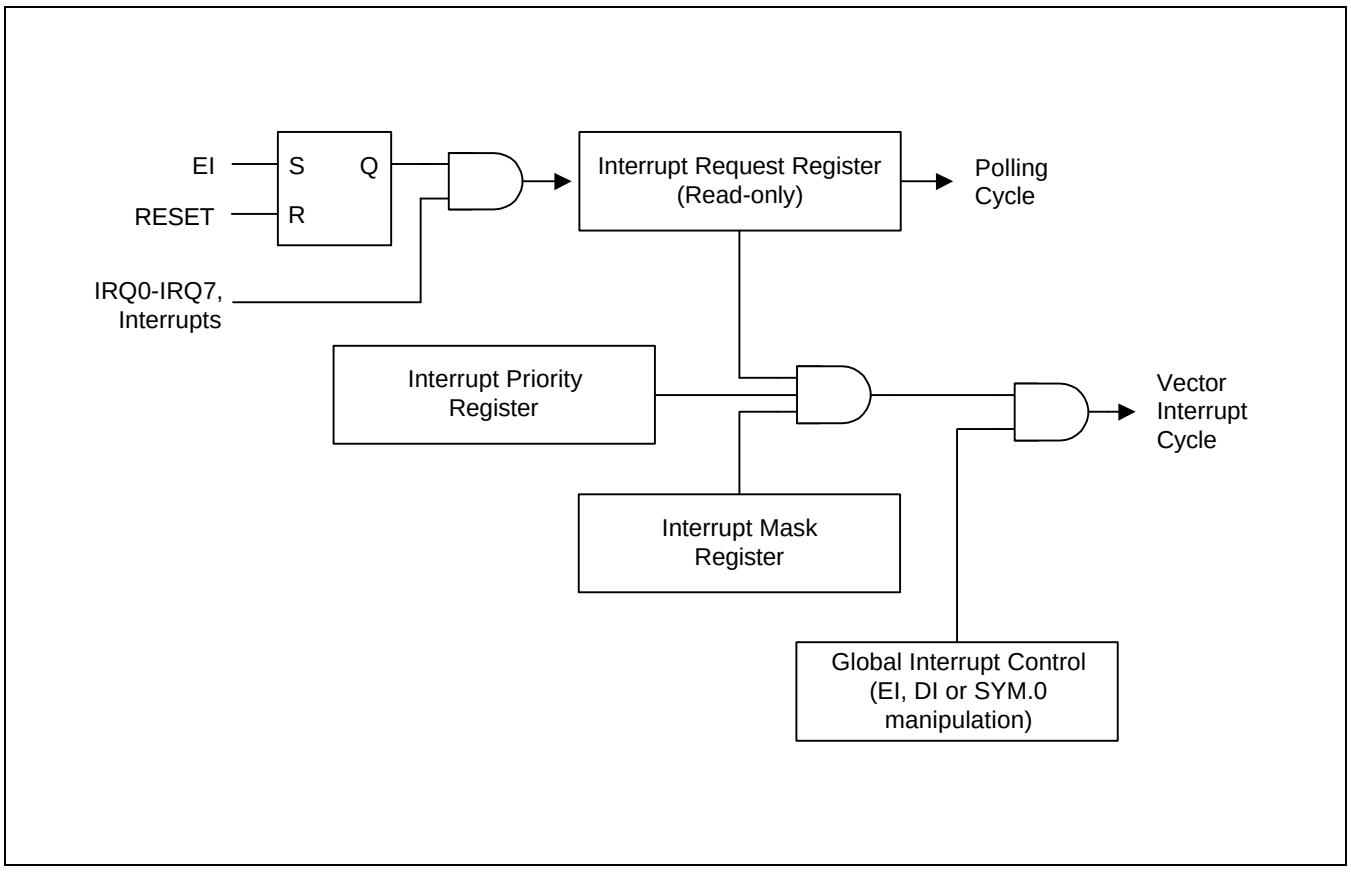


Figure 5-4. Interrupt Function Diagram

PERIPHERAL INTERRUPT CONTROL REGISTERS

For each interrupt source there is one or more corresponding peripheral control registers that let you control the interrupt generated by the related peripheral (see Table 5-3).

Table 5-3. Interrupt Source Control and Data Registers

Interrupt Source	Interrupt Level	Register(s)	Location(s) in Set 1
Timer 0 overflow Timer 0 match/capture	IRQ0	T0CON T0CNT T0DATA INTPND	E5H, bank 0 E3H, bank 0 E4H, bank 0 D0H, bank 0
Timer 1/A match Timer B match	IRQ1	TACON TACNT TADATA TBCON TBCNT TBDATA	EBH, bank 0 E7H, bank 0 E9H, bank0 EAH, bank 0 E6H, bank 0 E8H, bank 0
Timer 3 overflow Timer 3 match/capture Timer 2 match	IRQ2	T3CON T3CNTH, T3CNTL T3DATAH, T3DATAL T2CON T2CNT T2DATA	F8H, bank 0 F4H, F5H, bank 0 F6H, F7H, bank 0 EEH, bank 0 ECH, bank 0 EDH, bank 0
UART data receive UART data transmit SIO interrupt	IRQ3	UARTCON UDATA BRDATA SIOCON SIODATA SIOPS	FAH, bank 0 F9H, bank 0 FBH, bank 0 E0H, bank 0 E1H, bank 0 E2H, bank 0
Watch timer	IRQ4	WTCON	EFH, bank 1
P2.7 external interrupt P2.6 external interrupt P2.5 external interrupt P2.4 external interrupt	IRQ5	P2CONH P2INT	E0H, bank 1 E3H, bank 1
P4.7 external interrupt P4.6 external interrupt P4.5 external interrupt P4.4 external interrupt	IRQ6	P4CONH P4INT P4PND P4EDGE	E8H, bank 1 EAH, bank 1 EBH, bank 1 E7H, bank 1
P4.3 external interrupt P4.2 external interrupt P4.1 external interrupt P4.0 external interrupt	IRQ7	P4CONL P4INT 4PND P4EDGE	E9H, bank 1 EAH, bank 1 EBH, bank 1 E7H, bank 1

SYSTEM MODE REGISTER (SYM)

The system mode register, SYM (set 1, DEH), is used to globally enable and disable interrupt processing and to control fast interrupt processing (see Figure 5-5).

A reset clears SYM.1, and SYM.0 to "0". The 3-bit value for fast interrupt level selection, SYM.4–SYM.2, is undetermined.

The instructions EI and DI enable and disable global interrupt processing, respectively, by modifying the bit 0 value of the SYM register. In order to enable interrupt processing an Enable Interrupt (EI) instruction must be included in the initialization routine, which follows a reset operation. Although you can manipulate SYM.0 directly to enable and disable interrupts during the normal operation, it is recommended to use the EI and DI instructions for this purpose.

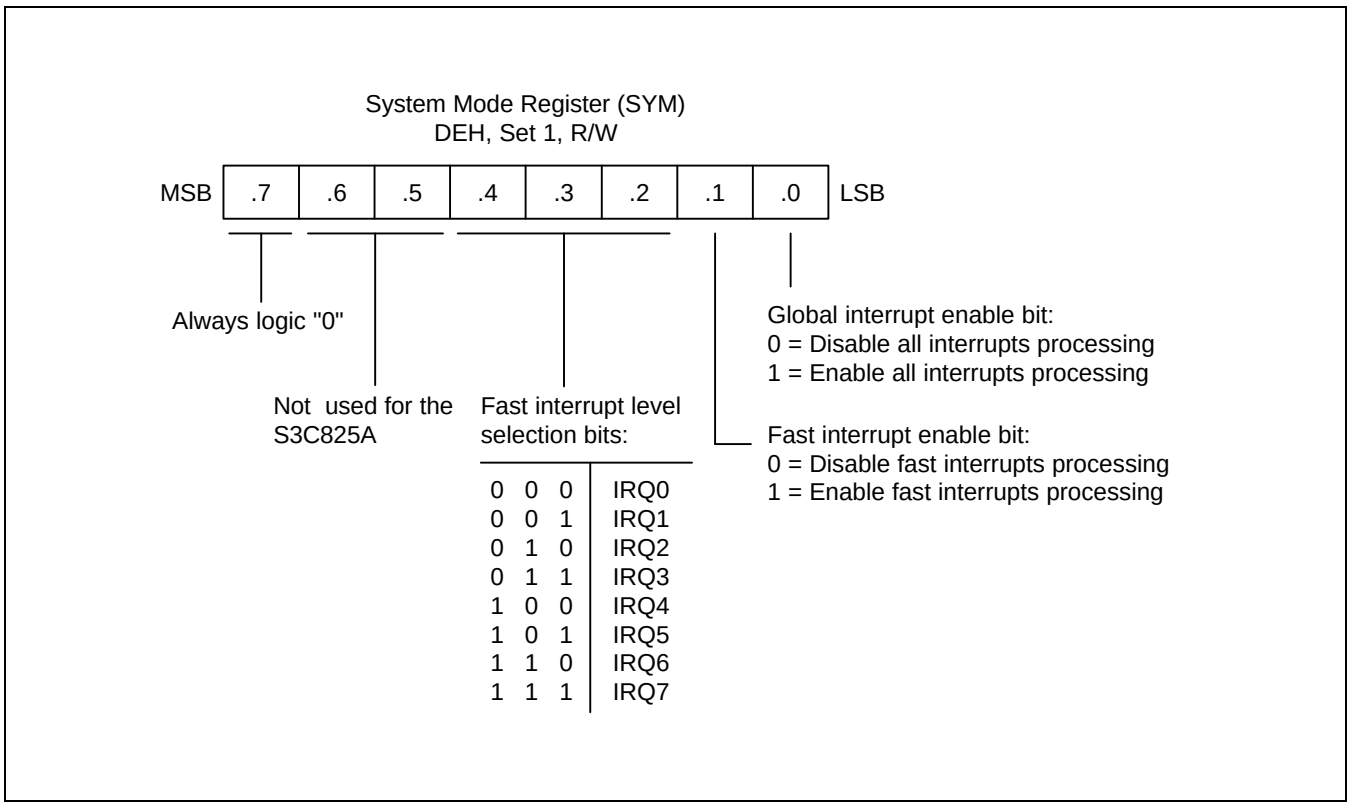


Figure 5-5. System Mode Register (SYM)

INTERRUPT MASK REGISTER (IMR)

The interrupt mask register, IMR (set 1, DDH) is used to enable or disable interrupt processing for individual interrupt levels. After a reset, all IMR bit values are undetermined and must therefore be written to their required settings by the initialization routine.

Each IMR bit corresponds to a specific interrupt level: bit 1 to IRQ1, bit 2 to IRQ2, and so on. When the IMR bit of an interrupt level is cleared to "0", interrupt processing for that level is disabled (masked). When you set a level's IMR bit to "1", interrupt processing for the level is enabled (not masked).

The IMR register is mapped to register location DDH in set 1. Bit values can be read and written by instructions using the Register addressing mode.

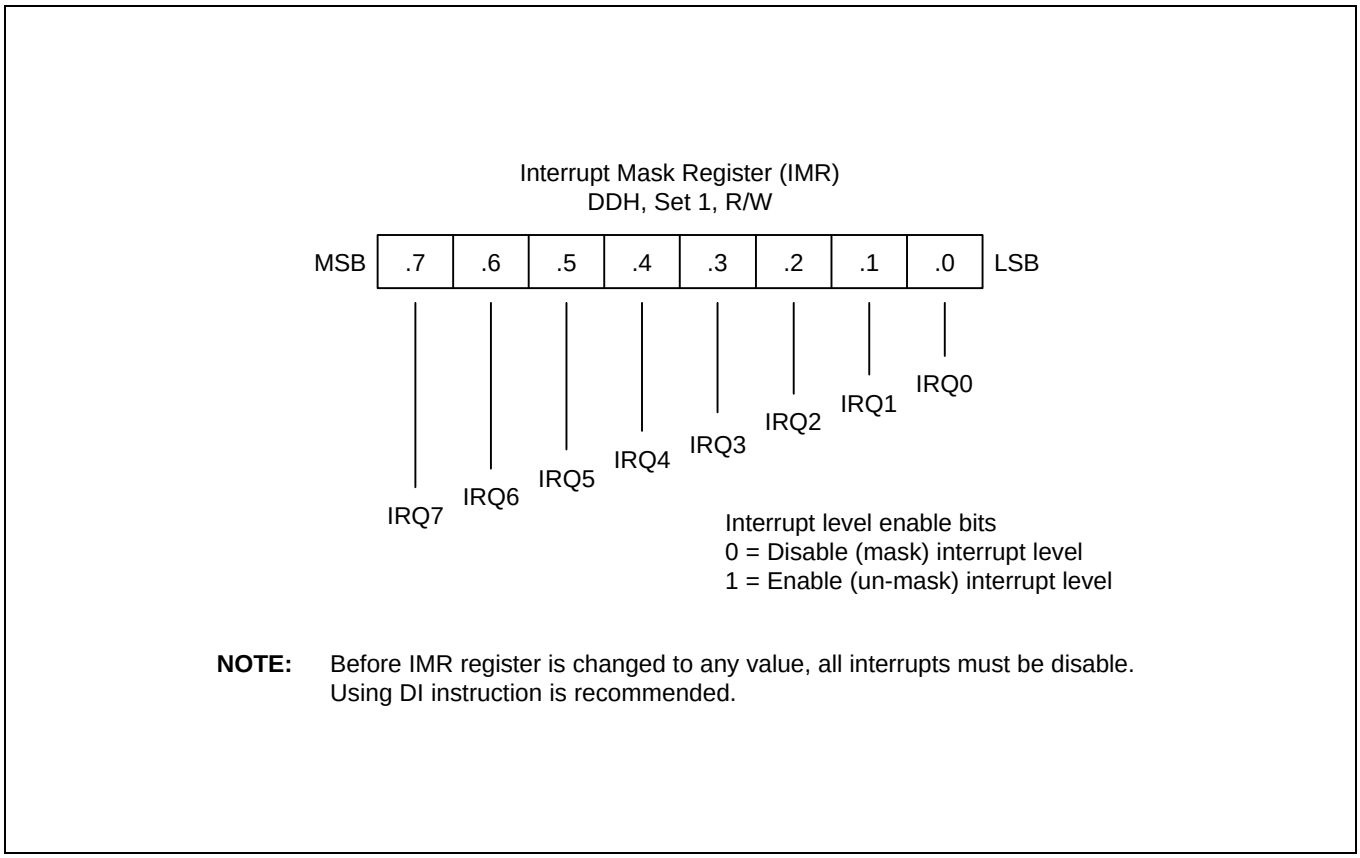


Figure 5-6. Interrupt Mask Register (IMR)

INTERRUPT PRIORITY REGISTER (IPR)

The interrupt priority register, IPR (set 1, bank 0, FFH), is used to set the relative priorities of the interrupt levels in the microcontroller's interrupt structure. After a reset, all IPR bit values are undetermined and must therefore be written to their required settings by the initialization routine.

When more than one interrupt sources are active, the source with the highest priority level is serviced first. If two sources belong to the same interrupt level, the source with the lower vector address usually has the priority (This priority is fixed in hardware).

To support programming of the relative interrupt level priorities, they are organized into groups and subgroups by the interrupt logic. Please note that these groups (and subgroups) are used only by IPR logic for the IPR register priority definitions (see Figure 5-7):

Group A IRQ0, IRQ1

Group B IRQ2, IRQ3, IRQ3

Group C IRQ5, IRQ6, IRQ7

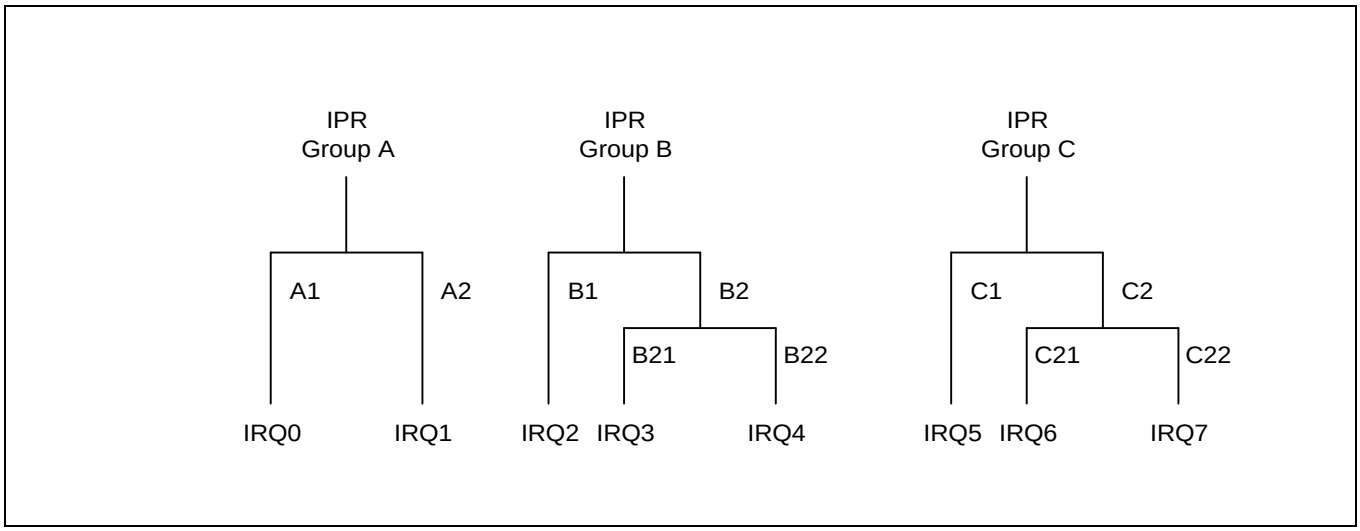


Figure 5-7. Interrupt Request Priority Groups

As you can see in Figure 5-8, IPR.7, IPR.4, and IPR.1 control the relative priority of interrupt groups A, B, and C. For example, the setting "001B" for these bits would select the group relationship B > C > A. The setting "101B" would select the relationship C > B > A.

The functions of the other IPR bit settings are as follows:

— IPR.5 controls the relative priorities of group C interrupts.

— Interrupt group C includes a subgroup that has an additional priority relationship among the interrupt levels 5, 6, and 7. IPR.6 defines the subgroup C relationship. IPR.5 controls the interrupt group C.

— IPR.0 controls the relative priority setting of IRQ0 and IRQ1 interrupts.

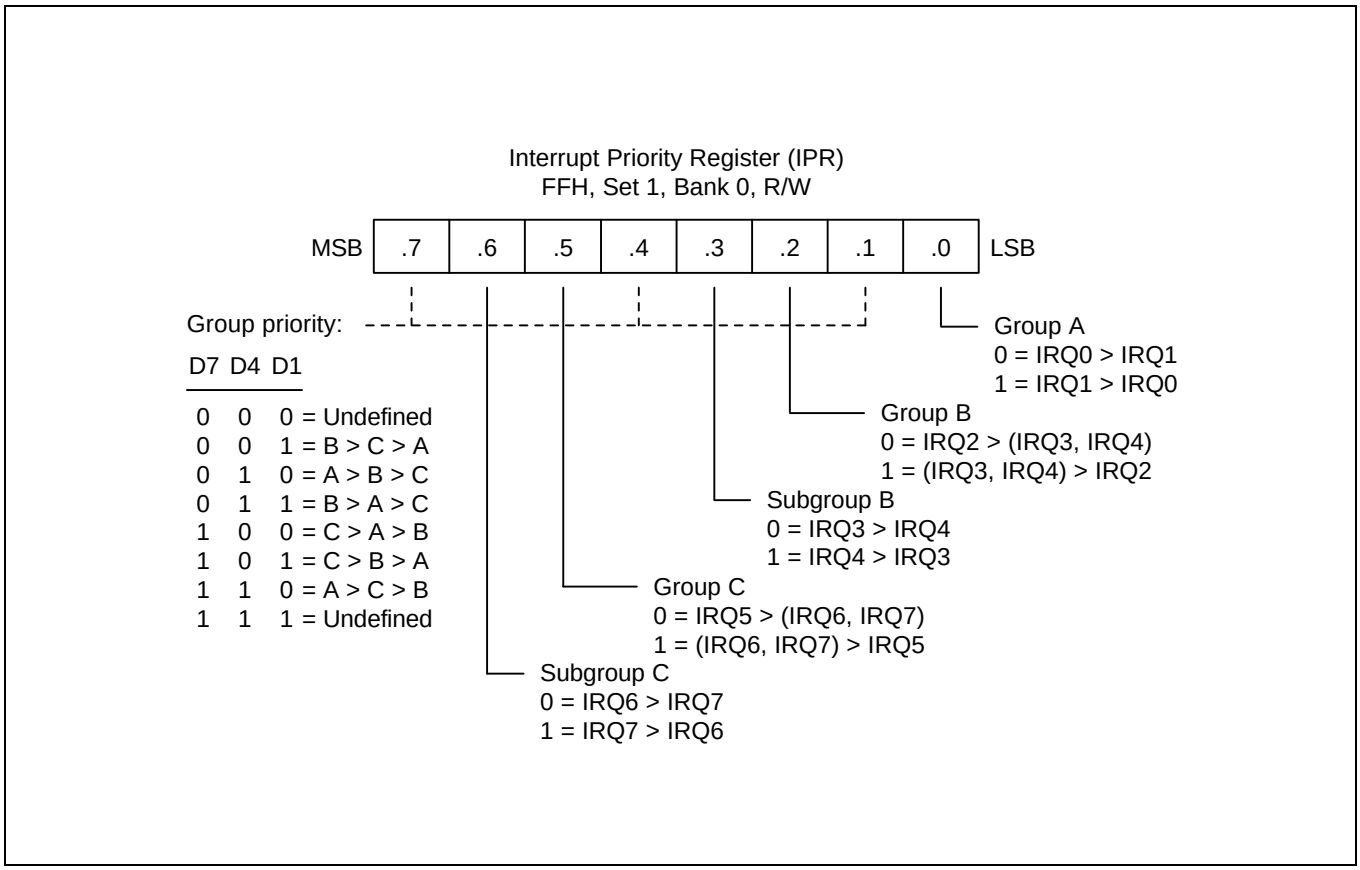


Figure 5-8. Interrupt Priority Register (IPR)

INTERRUPT REQUEST REGISTER (IRQ)

You can poll bit values in the interrupt request register, IRQ (set 1, DCH), to monitor interrupt request status for all levels in the microcontroller's interrupt structure. Each bit corresponds to the interrupt level of the same number: bit 0 to IRQ0, bit 1 to IRQ1, and so on. A "0" indicates that no interrupt request is currently being issued for that level. A "1" indicates that an interrupt request has been generated for that level.

IRQ bit values are read-only addressable using Register addressing mode. You can read (test) the contents of the IRQ register at any time using bit or byte addressing to determine the current interrupt request status of specific interrupt levels. After a reset, all IRQ status bits are cleared to "0".

You can poll IRQ register values even if a DI instruction has been executed (that is, if global interrupt processing is disabled). If an interrupt occurs while the interrupt structure is disabled, the CPU will not service it. You can, however, still detect the interrupt request by polling the IRQ register. In this way, you can determine which events occurred while the interrupt structure was globally disabled.

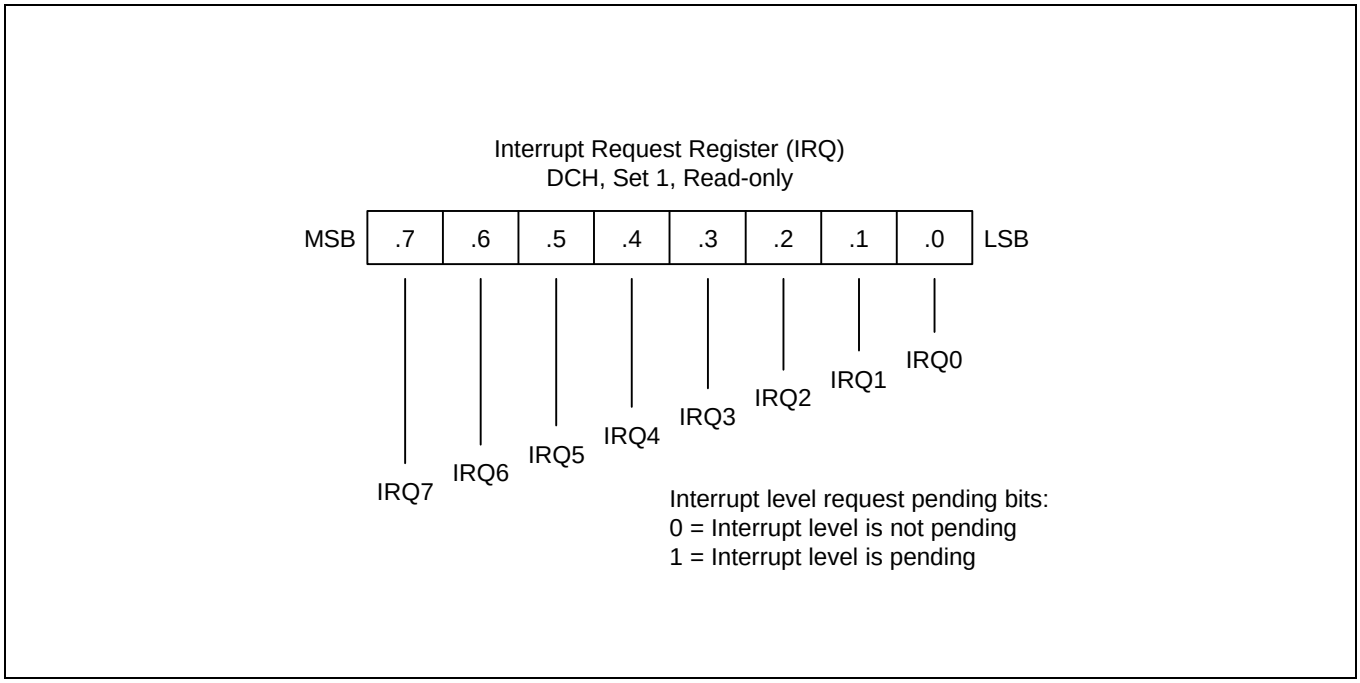


Figure 5-9. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

There are two types of interrupt pending bits: one type that is automatically cleared by hardware after the interrupt service routine is acknowledged and executed; the other that must be cleared in the interrupt service routine.

Pending Bits Cleared Automatically by Hardware

For interrupt pending bits that are cleared automatically by hardware, interrupt logic sets the corresponding pending bit to "1" when a request occurs. It then issues an IRQ pulse to inform the CPU that an interrupt is waiting to be serviced. The CPU acknowledges the interrupt source by sending an IACK, executes the service routine, and clears the pending bit to "0". This type of pending bit is not mapped and cannot, therefore, be read or written by application software.

In the S3C825A interrupt structure, the timer 0 overflow interrupt (IRQ0) and Timer 3 overflow interrupt (IRQ2) belongs to this category of interrupts in which pending condition is cleared automatically by hardware.

Pending Bits Cleared by the Service Routine

The second type of pending bit is the one that should be cleared by program software. The service routine must clear the appropriate pending bit before a return-from-interrupt subroutine (IRET) occurs. To do this, a "0" must be written to the corresponding pending bit location in the source's mode or control register.

INTERRUPT SOURCE POLLING SEQUENCE

The interrupt request polling and servicing sequence is as follows:

1. A source generates an interrupt request by setting the interrupt request bit to "1".
2. The CPU polling procedure identifies a pending condition for that source.
3. The CPU checks the source's interrupt level.
4. The CPU generates an interrupt acknowledge signal.
5. Interrupt logic determines the interrupt's vector address.
6. The service routine starts and the source's pending bit is cleared to "0" (by hardware or by software).
7. The CPU continues polling for interrupt requests.

INTERRUPT SERVICE ROUTINES

Before an interrupt request is serviced, the following conditions must be met:

— Interrupt processing must be globally enabled (EI, SYM.0 = "1")
— The interrupt level must be enabled (IMR register)
— The interrupt level must have the highest priority if more than one levels are currently requesting service
— The interrupt must be enabled at the interrupt's source (peripheral control register)

When all the above conditions are met, the interrupt request is acknowledged at the end of the instruction cycle. The CPU then initiates an interrupt machine cycle that completes the following processing sequence:

1. Reset (clear to "0") the interrupt enable bit in the SYM register (SYM.0) to disable all subsequent interrupts.
2. Save the program counter (PC) and status flags to the system stack.
3. Branch to the interrupt vector to fetch the address of the service routine.
4. Pass control to the interrupt service routine.

When the interrupt service routine is completed, the CPU issues an Interrupt Return (IRET). The IRET restores the PC and status flags, setting SYM.0 to "1". It allows the CPU to process the next interrupt request.

GENERATING INTERRUPT VECTOR ADDRESSES

The interrupt vector area in the ROM (00H–FFH) contains the addresses of interrupt service routines that correspond to each level in the interrupt structure. Vectored interrupt processing follows this sequence:

1. Push the program counter's low-byte value to the stack.
2. Push the program counter's high-byte value to the stack.
3. Push the FLAG register values to the stack.
4. Fetch the service routine's high-byte address from the vector location.
5. Fetch the service routine's low-byte address from the vector location.
6. Branch to the service routine specified by the concatenated 16-bit vector address.

NOTE

A 16-bit vector address always begins at an even-numbered ROM address within the range of 00H–FFH.

NESTING OF VECTORED INTERRUPTS

It is possible to nest a higher-priority interrupt request while a lower-priority request is being serviced. To do this, you must follow these steps:

1. Push the current 8-bit interrupt mask register (IMR) value to the stack (PUSH IMR).
2. Load the IMR register with a new mask value that enables only the higher priority interrupt.
3. Execute an EI instruction to enable interrupt processing (a higher priority interrupt will be processed if it occurs).
4. When the lower-priority interrupt service routine ends, restore the IMR to its original value by returning the previous mask value from the stack (POP IMR).
5. Execute an IRET.

Depending on the application, you may be able to simplify the procedure above to some extent.

INSTRUCTION POINTER (IP)

The instruction pointer (IP) is adopted by all the S3C8-series microcontrollers to control the optional high-speed interrupt processing feature called *fast interrupts*. The IP consists of register pair DAH and DBH. The names of IP registers are IPH (high byte, IP15–IP8) and IPL (low byte, IP7–IP0).

FAST INTERRUPT PROCESSING

The feature called *fast interrupt processing* allows an interrupt within a given level to be completed in approximately 6 clock cycles rather than the usual 16 clock cycles. To select a specific interrupt level for fast interrupt processing, you write the appropriate 3-bit value to SYM.4–SYM.2. Then, to enable fast interrupt processing for the selected level, you set SYM.1 to “1”.

FAST INTERRUPT PROCESSING (Continued)

Two other system registers support fast interrupt processing:

— The instruction pointer (IP) contains the starting address of the service routine (and is later used to swap the program counter values), and

— When a fast interrupt occurs, the contents of the FLAGS register is stored in an unmapped, dedicated register called FLAGS' ("FLAGS prime").

NOTE

For the S3C825A microcontroller, the service routine for any one of the eight interrupt levels: IRQ0–IRQ7, can be selected for fast interrupt processing.

Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

1. Load the start address of the service routine into the instruction pointer (IP).
2. Load the interrupt level number (IRQn) into the fast interrupt selection field (SYM.4–SYM.2)
3. Write a "1" to the fast interrupt enable bit in the SYM register.

Fast Interrupt Service Routine

When an interrupt occurs in the level selected for fast interrupt processing, the following events occur:

1. The contents of the instruction pointer and the PC are swapped.
2. The FLAG register values are written to the FLAGS' (“FLAGS prime”) register.
3. The fast interrupt status bit in the FLAGS register is set.
4. The interrupt is serviced.
5. Assuming that the fast interrupt status bit is set, when the fast interrupt service routine ends, the instruction pointer and PC values are swapped back.
6. The content of FLAGS' ("FLAGS prime") is copied automatically back to the FLAGS register.
7. The fast interrupt status bit in FLAGS is cleared automatically.

Relationship to Interrupt Pending Bit Types

As described previously, there are two types of interrupt pending bits: One type that is automatically cleared by hardware after the interrupt service routine is acknowledged and executed; the other that must be cleared by the application program's interrupt service routine. You can select fast interrupt processing for interrupts with either type of pending condition clear function — by hardware or by software.

Programming Guidelines

Remember that the only way to enable/disable a fast interrupt is to set/clear the fast interrupt enable bit in the SYM register, SYM.1. Executing an EI or DI instruction globally enables or disables all interrupt processing, including fast interrupts. If you use fast interrupts, remember to load the IP with a new start address when the fast interrupt service routine ends.

6 INSTRUCTION SET

OVERVIEW

The SAM88RC instruction set is specifically designed to support the large register files that are typical of most SAM8 microcontrollers. There are 78 instructions. The powerful data manipulation capabilities and features of the instruction set include:

— A full complement of 8-bit arithmetic and logic operations, including multiply and divide

— No special I/O instructions (I/O control/data registers are mapped directly into the register file)

— Decimal adjustment included in binary-coded decimal (BCD) operations

— 16-bit (word) data can be incremented and decremented

— Flexible instructions for bit addressing, rotate, and shift operations

DATA TYPES

The SAM8 CPU performs operations on bits, bytes, BCD digits, and two-byte words. Bits in the register file can be set, cleared, complemented, and tested. Bits within a byte are numbered from 7 to 0, where bit 0 is the least significant (right-most) bit.

REGISTER ADDRESSING

To access an individual register, an 8-bit address in the range 0-255 or the 4-bit address of a working register is specified. Paired registers can be used to construct 16-bit data or 16-bit program memory or data memory addresses. For detailed information about register addressing, please refer to Section 2, "Address Spaces."

ADDRESSING MODES

There are seven explicit addressing modes: Register (R), Indirect Register (IR), Indexed (X), Direct (DA), Relative (RA), Immediate (IM), and Indirect (IA). For detailed descriptions of these addressing modes, please refer to Section 3, "Addressing Modes."

Table 6-1. Instruction Group Summary

Mnemonic	Operands	Instruction
Load Instructions		
CLR	dst	Clear
LD	dst,src	Load
LDB	dst,src	Load bit
LDE	dst,src	Load external data memory
LDC	dst,src	Load program memory
LDED	dst,src	Load external data memory and decrement
LDCD	dst,src	Load program memory and decrement
LDEI	dst,src	Load external data memory and increment
LDCI	dst,src	Load program memory and increment
LDEPD	dst,src	Load external data memory with pre-decrement
LDCPD	dst,src	Load program memory with pre-decrement
LDEPI	dst,src	Load external data memory with pre-increment
LDCPI	dst,src	Load program memory with pre-increment
LDW	dst,src	Load word
POP	dst	Pop from stack
POPUD	dst,src	Pop user stack (decrementing)
POPUI	dst,src	Pop user stack (incrementing)
PUSH	src	Push to stack
PUSHUD	dst,src	Push user stack (decrementing)
PUSHUI	dst,src	Push user stack (incrementing)

Table 6-1. Instruction Group Summary (Continued)

Mnemonic	Operands	Instruction
Arithmetic Instructions		
ADC	dst,src	Add with carry
ADD	dst,src	Add
CP	dst,src	Compare
DA	dst	Decimal adjust
DEC	dst	Decrement
DECW	dst	Decrement word
DIV	dst,src	Divide
INC	dst	Increment
INCW	dst	Increment word
MULT	dst,src	Multiply
SBC	dst,src	Subtract with carry
SUB	dst,src	Subtract
Logic Instructions		
AND	dst,src	Logical AND
COM	dst	Complement
OR	dst,src	Logical OR
XOR	dst,src	Logical exclusive OR

Table 6-1. Instruction Group Summary (Continued)

Mnemonic	Operands	Instruction
Program Control Instructions		
BTJRF	dst,src	Bit test and jump relative on false
BTJRT	dst,src	Bit test and jump relative on true
CALL	dst	Call procedure
CPIJE	dst,src	Compare, increment and jump on equal
CPIJNE	dst,src	Compare, increment and jump on non-equal
DJNZ	r,dst	Decrement register and jump on non-zero
ENTER		Enter
EXIT		Exit
IRET		Interrupt return
JP	cc,dst	Jump on condition code
JP	dst	Jump unconditional
JR	cc,dst	Jump relative on condition code
NEXT		Next
RET		Return
WFI		Wait for interrupt
Bit Manipulation Instructions		
BAND	dst,src	Bit AND
BCP	dst,src	Bit compare
BITC	dst	Bit complement
BITR	dst	Bit reset
BITS	dst	Bit set
BOR	dst,src	Bit OR
BXOR	dst,src	Bit XOR
TCM	dst,src	Test complement under mask
TM	dst,src	Test under mask

Table 6-1. Instruction Group Summary (Concluded)

Mnemonic	Operands	Instruction
Rotate and Shift Instructions		
RL	dst	Rotate left
RLC	dst	Rotate left through carry
RR	dst	Rotate right
RRC	dst	Rotate right through carry
SRA	dst	Shift right arithmetic
SWAP	dst	Swap nibbles
CPU Control Instructions		
CCF		Complement carry flag
DI		Disable interrupts
EI		Enable interrupts
IDLE		Enter Idle mode
NOP		No operation
RCF		Reset carry flag
SB0		Set bank 0
SB1		Set bank 1
SCF		Set carry flag
SRP	src	Set register pointers
SRP0	src	Set register pointer 0
SRP1	src	Set register pointer 1
STOP		Enter Stop mode

FLAGS REGISTER (FLAGS)

The flags register FLAGS contains eight bits that describe the current status of CPU operations. Four of these bits, FLAGS.7–FLAGS.4, can be tested and used with conditional jump instructions; two others FLAGS.3 and FLAGS.2 are used for BCD arithmetic.

The FLAGS register also contains a bit to indicate the status of fast interrupt processing (FLAGS.1) and a bank address status bit (FLAGS.0) to indicate whether bank 0 or bank 1 is currently being addressed. FLAGS register can be set or reset by instructions as long as its outcome does not affect the flags, such as, Load instruction.

Logical and Arithmetic instructions such as, AND, OR, XOR, ADD, and SUB can affect the Flags register. For example, the AND instruction updates the Zero, Sign and Overflow flags based on the outcome of the AND instruction. If the AND instruction uses the Flags register as the destination, then simultaneously, two write will occur to the Flags register producing an unpredictable result.

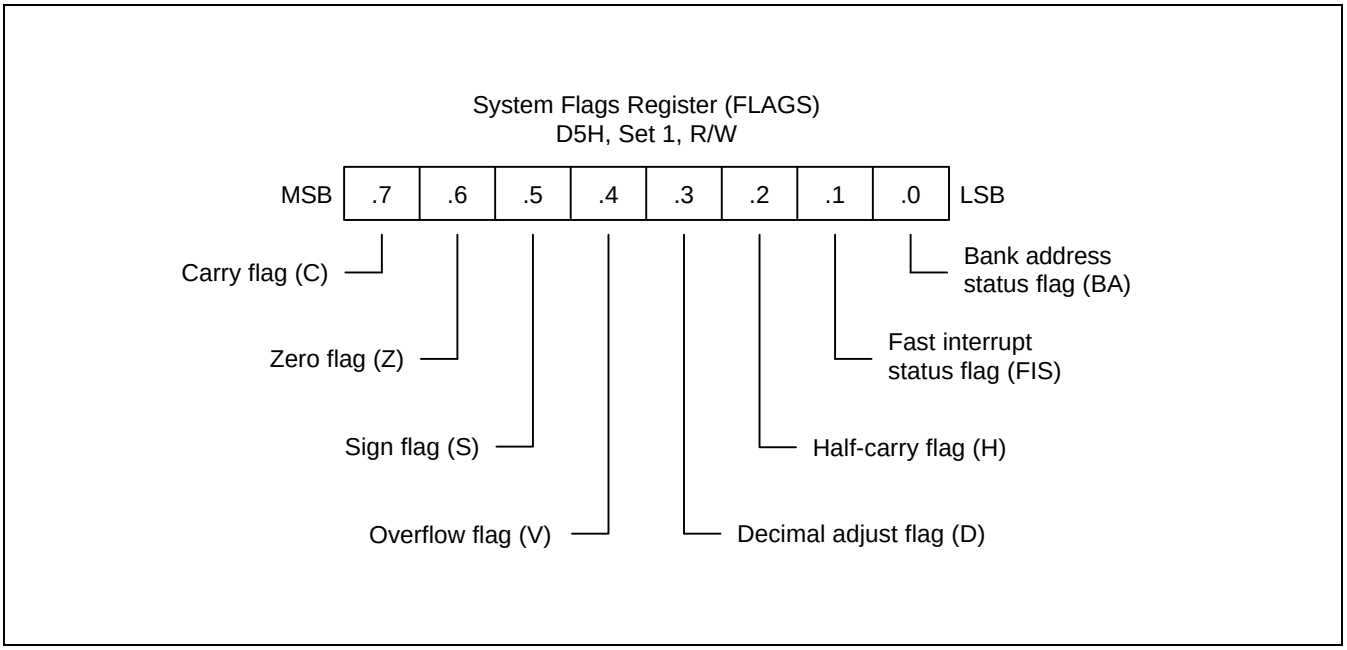


Figure 6-1. System Flags Register (FLAGS)

FLAG DESCRIPTIONS

C Carry Flag (FLAGS.7)

The C flag is set to "1" if the result from an arithmetic operation generates a carry-out from or a borrow to the bit 7 position (MSB). After rotate and shift operations, it contains the last value shifted out of the specified register. Program instructions can set, clear, or complement the carry flag.

Z Zero Flag (FLAGS.6)

For arithmetic and logic operations, the Z flag is set to "1" if the result of the operation is zero. For operations that test register bits, and for shift and rotate operations, the Z flag is set to "1" if the result is logic zero.

S Sign Flag (FLAGS.5)

Following arithmetic, logic, rotate, or shift operations, the sign bit identifies the state of the MSB of the result. A logic zero indicates a positive number and a logic one indicates a negative number.

V Overflow Flag (FLAGS.4)

The V flag is set to "1" when the result of a two's-complement operation is greater than + 127 or less than – 128. It is also cleared to "0" following logic operations.

D Decimal Adjust Flag (FLAGS.3)

The DA bit is used to specify what type of instruction was executed last during BCD operations, so that a subsequent decimal adjust operation can execute correctly. The DA bit is not usually accessed by programmers, and cannot be used as a test condition.

H Half-Carry Flag (FLAGS.2)

The H bit is set to "1" whenever an addition generates a carry-out of bit 3, or when a subtraction borrows out of bit 4. It is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. The H flag is seldom accessed directly by a program.

FIS Fast Interrupt Status Flag (FLAGS.1)

The FIS bit is set during a fast interrupt cycle and reset during the IRET following interrupt servicing. When set, it inhibits all interrupts and causes the fast interrupt return to be executed when the IRET instruction is executed.

BA Bank Address Flag (FLAGS.0)

The BA flag indicates which register bank in the set 1 area of the internal register file is currently selected, bank 0 or bank 1. The BA flag is cleared to "0" (select bank 0) when you execute the SB0 instruction and is set to "1" (select bank 1) when you execute the SB1 instruction.

INSTRUCTION SET NOTATION

Table 6-2. Flag Notation Conventions

Flag	Description
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag
0	Cleared to logic zero
1	Set to logic one
*	Set or cleared according to operation
–	Value is unaffected
x	Value is undefined

Table 6-3. Instruction Set Symbols

Symbol	Description
dst	Destination operand
src	Source operand
@	Indirect register address prefix
PC	Program counter
IP	Instruction pointer
FLAGS	Flags register (D5H)
RP	Register pointer
#	Immediate operand or register address prefix
H	Hexadecimal number suffix
D	Decimal number suffix
B	Binary number suffix
opc	Opcode

Table 6-4. Instruction Notation Conventions

Notation	Description	Actual Operand Range
cc	Condition code	See list of condition codes in Table 6-6.
r	Working register only	Rn (n = 0–15)
rb	Bit (b) of working register	Rn.b (n = 0–15, b = 0–7)
r0	Bit 0 (LSB) of working register	Rn (n = 0–15)
rr	Working register pair	RRp (p = 0, 2, 4, ..., 14)
R	Register or working register	reg or Rn (reg = 0–255, n = 0–15)
Rb	Bit 'b' of register or working register	reg.b (reg = 0–255, b = 0–7)
RR	Register pair or working register pair	reg or RRp (reg = 0–254, even number only, where p = 0, 2, ..., 14)
IA	Indirect addressing mode	addr (addr = 0–254, even number only)
Ir	Indirect working register only	@Rn (n = 0–15)
IR	Indirect register or indirect working register	@Rn or @reg (reg = 0–255, n = 0–15)
Irr	Indirect working register pair only	@RRp (p = 0, 2, ..., 14)
IRR	Indirect register pair or indirect working register pair	@RRp or @reg (reg = 0–254, even only, where p = 0, 2, ..., 14)
X	Indexed addressing mode	#reg [Rn] (reg = 0–255, n = 0–15)
XS	Indexed (short offset) addressing mode	#addr [RRp] (addr = range –128 to +127, where p = 0, 2, ..., 14)
xl	Indexed (long offset) addressing mode	#addr [RRp] (addr = range 0–65535, where p = 0, 2, ..., 14)
da	Direct addressing mode	addr (addr = range 0–65535)
ra	Relative addressing mode	addr (addr = number in the range +127 to –128 that is an offset relative to the address of the next instruction)
im	Immediate addressing mode	#data (data = 0–255)
iml	Immediate (long) addressing mode	#data (data = range 0–65535)

Table 6-5. Opcode Quick Reference

OPCODE MAP									
LOWER NIBBLE (HEX)									
	–	0	1	2	3	4	5	6	7
U	0	DEC R1	DEC IR1	ADD r1,r2	ADD r1,Ir2	ADD R2,R1	ADD IR2,R1	ADD R1,IM	BOR r0–Rb
P	1	RLC R1	RLC IR1	ADC r1,r2	ADC r1,Ir2	ADC R2,R1	ADC IR2,R1	ADC R1,IM	BCP r1.b, R2
P	2	INC R1	INC IR1	SUB r1,r2	SUB r1,Ir2	SUB R2,R1	SUB IR2,R1	SUB R1,IM	BXOR r0–Rb
E	3	JP IRR1	SRP/0/1 IM	SBC r1,r2	SBC r1,Ir2	SBC R2,R1	SBC IR2,R1	SBC R1,IM	BTJR r2.b, RA
R	4	DA R1	DA IR1	OR r1,r2	OR r1,Ir2	OR R2,R1	OR IR2,R1	OR R1,IM	LDB r0–Rb
	5	POP R1	POP IR1	AND r1,r2	AND r1,Ir2	AND R2,R1	AND IR2,R1	AND R1,IM	BITC r1.b
N	6	COM R1	COM IR1	TCM r1,r2	TCM r1,Ir2	TCM R2,R1	TCM IR2,R1	TCM R1,IM	BAND r0–Rb
I	7	PUSH R2	PUSH IR2	TM r1,r2	TM r1,Ir2	TM R2,R1	TM IR2,R1	TM R1,IM	BIT r1.b
B	8	DECW RR1	DECW IR1	PUSHUD IR1,R2	PUSHUI IR1,R2	MULT R2,RR1	MULT IR2,RR1	MULT IM,RR1	LD r1, x, r2
B	9	RL R1	RL IR1	POPUD IR2,R1	POPUI IR2,R1	DIV R2,RR1	DIV IR2,RR1	DIV IM,RR1	LD r2, x, r1
L	A	INCW RR1	INCW IR1	CP r1,r2	CP r1,Ir2	CP R2,R1	CP IR2,R1	CP R1,IM	LDC r1, Irr2, xL
E	B	CLR R1	CLR IR1	XOR r1,r2	XOR r1,Ir2	XOR R2,R1	XOR IR2,R1	XOR R1,IM	LDC r2, Irr2, xL
	C	RRC R1	RRC IR1	CPIJE Ir,r2,RA	LDC r1,Irr2	LDW RR2,RR1	LDW IR2,RR1	LDW RR1,IML	LD r1, Ir2
H	D	SRA R1	SRA IR1	CPIJNE Irr,r2,RA	LDC r2,Irr1	CALL IA1		LD IR1,IM	LD Ir1, r2
E	E	RR R1	RR IR1	LDCD r1,Irr2	LDCI r1,Irr2	LD R2,R1	LD R2,IR1	LD R1,IM	LDC r1, Irr2, xs
X	F	SWAP R1	SWAP IR1	LDCPD r2,Irr1	LDCPI r2,Irr1	CALL IRR1	LD IR2,R1	CALL DA1	LDC r2, Irr1, xs

Table 6-5. Opcode Quick Reference (Continued)

OPCODE MAP									
LOWER NIBBLE (HEX)									
	–	8	9	A	B	C	D	E	F
U	0	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NEXT
P	1	↓	↓	↓	↓	↓	↓	↓	ENTER
P	2								EXIT
E	3								WFI
R	4								SB0
	5								SB1
N	6								IDLE
I	7	↓	↓	↓	↓	↓	↓	↓	STOP
B	8								DI
B	9								EI
L	A								RET
E	B								IRET
	C								RCF
H	D	↓	↓	↓	↓	↓	↓	↓	SCF
E	E								CCF
X	F	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NOP

CONDITION CODES

The opcode of a conditional jump always contains a 4-bit field called the condition code (cc). This specifies under which conditions it is to execute the jump. For example, a conditional jump with the condition code for "equal" after a compare operation only jumps if the two operands are equal. Condition codes are listed in Table 6-6.

The carry (C), zero (Z), sign (S), and overflow (V) flags are used to control the operation of conditional jump instructions.

Table 6-6. Condition Codes

Binary	Mnemonic	Description	Flags Set
0000	F	Always false	–
1000	T	Always true	–
0111 (note)	C	Carry	C = 1
1111 (note)	NC	No carry	C = 0
0110 (note)	Z	Zero	Z = 1
1110 (note)	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110 (note)	EQ	Equal	Z = 1
1110 (note)	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	(Z OR (S XOR V)) = 0
0010	LE	Less than or equal	(Z OR (S XOR V)) = 1
1111 (note)	UGE	Unsigned greater than or equal	C = 0
0111 (note)	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1

NOTES:

1. It indicates condition codes that are related to two different mnemonics but which test the same flag. For example, Z and EQ are both true if the zero flag (Z) is set, but after an ADD instruction, Z would probably be used; after a CP instruction, however, EQ would probably be used.
2. For operations involving unsigned numbers, the special condition codes UGE, ULT, UGT, and ULE must be used.

INSTRUCTION DESCRIPTIONS

This section contains detailed information and programming examples for each instruction in the SAM8 instruction set. Information is arranged in a consistent format for improved readability and for fast referencing. The following information is included in each instruction description:

— Instruction name (mnemonic)
— Full instruction name
— Source/destination format of the instruction operand
— Shorthand notation of the instruction's operation
— Textual description of the instruction's effect
— Specific flag settings affected by the instruction
— Detailed description of the instruction's format, execution time, and addressing mode(s)
— Programming example(s) explaining how to use the instruction

ADC — Add with carry

ADC dst,src

Operation: dst ← dst + src + c

The source operand, along with the setting of the carry flag, is added to the destination operand and the sum is stored in the destination. The contents of the source are unaffected. Two's-complement addition is performed. In multiple precision arithmetic, this instruction permits the carry from the addition of low-order operands to be carried into the addition of high-order operands.

Flags:

C: Set if there is a carry from the most significant bit of the result; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.
D: Always cleared to "0".
H: Set if there is a carry from the most significant bit of the low-order four bits of the result; cleared otherwise.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | src		2	4	12	r	r
				6	13	r	lr
opc	src	dst	3	6	14	R	R
				6	15	R	IR
opc	dst	src	3	6	16	R	IM

Examples: Given: R1 = 10H, R2 = 03H, C flag = "1", register 01H = 20H, register 02H = 03H, and register 03H = 0AH:

ADC R1,R2 → R1 = 14H, R2 = 03H
ADC R1,@R2 → R1 = 1BH, R2 = 03H
ADC 01H,02H → Register 01H = 24H, register 02H = 03H
ADC 01H,@02H → Register 01H = 2BH, register 02H = 03H
ADC 01H,#11H → Register 01H = 32H

In the first example, destination register R1 contains the value 10H, the carry flag is set to "1", and the source working register R2 contains the value 03H. The statement "ADC R1,R2" adds 03H and the carry flag value ("1") to the destination value 10H, leaving 14H in register R1.

ADD — Add

ADD dst,src

Operation: dst ← dst + src

The source operand is added to the destination operand and the sum is stored in the destination. The contents of the source are unaffected. Two's-complement addition is performed.

Flags:
C: Set if there is a carry from the most significant bit of the result; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.
D: Always cleared to "0".
H: Set if a carry from the low-order nibble occurred.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc | dst | src	2	4	02	r	r
		6	03	r	lr
opc | src | dst	3	6	04	R	R
		6	05	R	IR
opc | dst | src	3	6	06	R	IM

Examples: Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

ADD R1,R2 → R1 = 15H, R2 = 03H
ADD R1,@R2 → R1 = 1CH, R2 = 03H
ADD 01H,02H → Register 01H = 24H, register 02H = 03H
ADD 01H,@02H → Register 01H = 2BH, register 02H = 03H
ADD 01H,#25H → Register 01H = 46H

In the first example, destination working register R1 contains 12H and the source working register R2 contains 03H. The statement "ADD R1,R2" adds 03H to 12H, leaving the value 15H in register R1.

AND — Logical AND

AND dst,src

Operation: dst ← dst AND src

The source operand is logically ANDed with the destination operand. The result is stored in the destination. The AND operation results in a "1" bit being stored whenever the corresponding bits in the two operands are both logic ones; otherwise a "0" bit value is stored. The contents of the source are unaffected.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always cleared to "0".
D: Unaffected.
H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | src		2	4	52	r	r
				6	53	r	lr
opc	src	dst	3	6	54	R	R
				6	55	R	IR
opc	dst	src	3	6	56	R	IM

Examples: Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

AND R1,R2 → R1 = 02H, R2 = 03H
AND R1,@R2 → R1 = 02H, R2 = 03H
AND 01H,02H → Register 01H = 01H, register 02H = 03H
AND 01H,@02H → Register 01H = 00H, register 02H = 03H
AND 01H,#25H → Register 01H = 21H

In the first example, destination working register R1 contains the value 12H and the source working register R2 contains 03H. The statement "AND R1,R2" logically ANDs the source operand 03H with the destination operand value 12H, leaving the value 02H in register R1.

BAND — Bit AND

BAND dst,src.b

BAND dst.b,src

Operation: dst(0) ← dst(0) AND src(b)

or

dst(b) ← dst(b) AND src(0)

The specified bit of the source (or the destination) is logically ANDed with the zero bit (LSB) of the destination (or source). The resultant bit is stored in the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Cleared to "0".
V: Undefined.
D: Unaffected.
H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | b | 0	src	3	6	67	r0	Rb
opc	src | b | 1	dst	3	6	67	Rb	r0

NOTE: In the second byte of the 3-byte instruction formats, the destination (or source) address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit in length.

Examples: Given: R1 = 07H and register 01H = 05H:

BAND R1,01H.1 → R1 = 06H, register 01H = 05H

BAND 01H.1,R1 → Register 01H = 05H, R1 = 07H

In the first example, source register 01H contains the value 05H (00000101B) and destination working register R1 contains 07H (00000111B). The statement "BAND R1,01H.1" ANDs the bit 1 value of the source register ("0") with the bit 0 value of register R1 (destination), leaving the value 06H (00000110B) in register R1.

BCP — Bit Compare

BCP dst,src.b

Operation: dst(0) – src(b)

The specified bit of the source is compared to (subtracted from) bit zero (LSB) of the destination. The zero flag is set if the bits are the same; otherwise it is cleared. The contents of both operands are unaffected by the comparison.

Flags:
C: Unaffected.
Z: Set if the two bits are the same; cleared otherwise.
S: Cleared to "0".
V: Undefined.
D: Unaffected.
H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | b | 0	src	3	6	17	r0	Rb

NOTE: In the second byte of the instruction format, the destination address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H and register 01H = 01H:

BCP R1,01H.1 → R1 = 07H, register 01H = 01H

If destination working register R1 contains the value 07H (00000111B) and the source register 01H contains the value 01H (00000001B), the statement "BCP R1,01H.1" compares bit one of the source register (01H) and bit zero of the destination register (R1). Because the bit values are not identical, the zero flag bit (Z) is cleared in the FLAGS register (0D5H).

BITC — Bit Complement

BITC dst.b

Operation: dst(b) ← NOT dst(b)

This instruction complements the specified bit within the destination without affecting any other bits in the destination.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Cleared to "0".
V: Undefined.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst | b | 0	2	4	57	rb

NOTE: In the second byte of the instruction format, the destination address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H

BITC R1.1 → R1 = 05H

If working register R1 contains the value 07H (00000111B), the statement "BITC R1.1" complements bit one of the destination and leaves the value 05H (00000101B) in register R1. Because the result of the complement is not "0", the zero flag (Z) in the FLAGS register (0D5H) is cleared.

BITR — Bit Reset

BITR dst.b

Operation: dst(b) ← 0

The BITR instruction clears the specified bit within the destination without affecting any other bits in the destination.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst | b | 0	2	4	77	rb

NOTE: In the second byte of the instruction format, the destination address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H:

BITR R1.1 → R1 = 05H

If the value of working register R1 is 07H (00000111B), the statement "BITR R1.1" clears bit one of the destination register R1, leaving the value 05H (00000101B).

BITS — Bit Set

BITS dst.b

Operation: dst(b) ← 1

The BITS instruction sets the specified bit within the destination without affecting any other bits in the destination.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst | b | 1	2	4	77	rb

NOTE: In the second byte of the instruction format, the destination address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H:

BITS R1.3 → R1 = 0FH

If working register R1 contains the value 07H (00000111B), the statement "BITS R1.3" sets bit three of the destination register R1 to "1", leaving the value 0FH (00001111B).

BOR — Bit OR

BOR dst,src.b

BOR dst.b,src

Operation: dst(0) ← dst(0) OR src(b)

or

dst(b) ← dst(b) OR src(0)

The specified bit of the source (or the destination) is logically ORed with bit zero (LSB) of the destination (or the source). The resulting bit value is stored in the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Cleared to "0".
V: Undefined.
D: Unaffected.
H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | b | 0	src	3	6	07	r0	Rb
opc	src | b | 1	dst	3	6	07	Rb	r0

NOTE: In the second byte of the 3-byte instruction formats, the destination (or source) address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit.

Examples: Given: R1 = 07H and register 01H = 03H:

BOR R1, 01H.1 → R1 = 07H, register 01H = 03H

BOR 01H.2, R1 → Register 01H = 07H, R1 = 07H

In the first example, destination working register R1 contains the value 07H (00000111B) and source register 01H the value 03H (00000011B). The statement "BOR R1,01H.1" logically ORs bit one of register 01H (source) with bit zero of R1 (destination). This leaves the same value (07H) in working register R1.

In the second example, destination register 01H contains the value 03H (00000011B) and the source working register R1 the value 07H (00000111B). The statement "BOR 01H.2,R1" logically ORs bit two of register 01H (destination) with bit zero of R1 (source). This leaves the value 07H in register 01H.

BTJRF — Bit Test, Jump Relative on False

BTJRF dst,src.b

Operation: If src(b) is a "0", then PC ← PC + dst

The specified bit within the source operand is tested. If it is a "0", the relative address is added to the program counter and control passes to the statement whose address is now in the PC; otherwise, the instruction following the BTJRF instruction is executed.

Flags: No flags are affected.

Format:

(Note 1)

opc	src | b | 0	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
3	10	37	RA	rb

NOTE: In the second byte of the instruction format, the source address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H:

BTJRF SKIP,R1.3 → PC jumps to SKIP location

If working register R1 contains the value 07H (00000111B), the statement "BTJRF SKIP,R1.3" tests bit 3. Because it is "0", the relative address is added to the PC and the PC jumps to the memory location pointed to by the SKIP. (Remember that the memory location must be within the allowed range of + 127 to – 128.)

BTJRT — Bit Test, Jump Relative on True

BTJRT dst,src.b

Operation: If src(b) is a "1", then PC ← PC + dst

The specified bit within the source operand is tested. If it is a "1", the relative address is added to the program counter and control passes to the statement whose address is now in the PC; otherwise, the instruction following the BTJRT instruction is executed.

Flags: No flags are affected.

Format:

(Note 1)

opc	src | b | 1	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
3	10	37	RA	rb

NOTE: In the second byte of the instruction format, the source address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit in length.

Example: Given: R1 = 07H:

```
BTJRT   SKIP,R1.1
```

If working register R1 contains the value 07H (00000111B), the statement "BTJRT SKIP,R1.1" tests bit one in the source register (R1). Because it is a "1", the relative address is added to the PC and the PC jumps to the memory location pointed to by the SKIP. (Remember that the memory location must be within the allowed range of + 127 to – 128.)

BXOR — Bit XOR

BXOR dst,src.b

BXOR dst.b,src

Operation: dst(0) ← dst(0) XOR src(b)

or

dst(b) ← dst(b) XOR src(0)

The specified bit of the source (or the destination) is logically exclusive-ORed with bit zero (LSB) of the destination (or source). The result bit is stored in the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Cleared to "0".
V: Undefined.
D: Unaffected.
H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | b | 0	src	3	6	27	r0	Rb
opc	src | b | 1	dst	3	6	27	Rb	r0

NOTE: In the second byte of the 3-byte instruction formats, the destination (or source) address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit in length.

Examples: Given: R1 = 07H (00000111B) and register 01H = 03H (00000011B):

BXOR R1,01H.1 → R1 = 06H, register 01H = 03H

BXOR 01H.2,R1 → Register 01H = 07H, R1 = 07H

In the first example, destination working register R1 has the value 07H (00000111B) and source register 01H has the value 03H (00000011B). The statement "BXOR R1,01H.1" exclusive-ORs bit one of register 01H (source) with bit zero of R1 (destination). The result bit value is stored in bit zero of R1, changing its value from 07H to 06H. The value of source register 01H is unaffected.

CALL — Call Procedure

CALL dst

Operation:

SP ← SP – 1
@SP ← PCL
SP ← SP –1
@SP ← PCH
PC ← dst

The current contents of the program counter are pushed onto the top of the stack. The program counter value used is the address of the first instruction following the CALL instruction. The specified destination address is then loaded into the program counter and points to the first instruction of a procedure. At the end of the procedure the return instruction (RET) can be used to return to the original program flow. RET pops the top of the stack back into the program counter.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	3	14	F6	DA
opc | dst	2	12	F4	IRR
opc | dst	2	14	D4	IA

Examples: Given: R0 = 35H, R1 = 21H, PC = 1A47H, and SP = 0002H:

CALL 3521H → SP = 0000H
(Memory locations 0000H = 1AH, 0001H = 4AH, where 4AH is the address that follows the instruction.)

CALL @RR0 → SP = 0000H (0000H = 1AH, 0001H = 49H)

CALL #40H → SP = 0000H (0000H = 1AH, 0001H = 49H)

In the first example, if the program counter value is 1A47H and the stack pointer contains the value 0002H, the statement "CALL 3521H" pushes the current PC value onto the top of the stack. The stack pointer now points to memory location 0000H. The PC is then loaded with the value 3521H, the address of the first instruction in the program sequence to be executed.

If the contents of the program counter and stack pointer are the same as in the first example, the statement "CALL @RR0" produces the same result except that the 49H is stored in stack location 0001H (because the two-byte instruction format was used). The PC is then loaded with the value 3521H, the address of the first instruction in the program sequence to be executed. Assuming that the contents of the program counter and stack pointer are the same as in the first example, if program address 0040H contains 35H and program address 0041H contains 21H, the statement "CALL #40H" produces the same result as in the second example.

CCF — Complement Carry Flag

CCF

Operation: C ← NOT C

The carry flag (C) is complemented. If C = "1", the value of the carry flag is changed to logic zero; if C = "0", the value of the carry flag is changed to logic one.

Flags: **C:** Complemented.

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	EF

Example: Given: The carry flag = "0":

CCF

If the carry flag = "0", the CCF instruction complements it in the FLAGS register (0D5H), changing its value from logic zero to logic one.

CLR — Clear

CLR dst

Operation: dst ← "0"

The destination location is cleared to "0".

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	B0	R
		4	B1	IR

Examples: Given: Register 00H = 4FH, register 01H = 02H, and register 02H = 5EH:

CLR 00H → Register 00H = 00H

CLR @01H → Register 01H = 02H, register 02H = 00H

In Register (R) addressing mode, the statement "CLR 00H" clears the destination register 00H value to 00H. In the second example, the statement "CLR @01H" uses Indirect Register (IR) addressing mode to clear the 02H register value to 00H.

COM — Complement

COM dst

Operation: dst ← NOT dst

The contents of the destination location are complemented (one's complement); all "1s" are changed to "0s", and vice-versa.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to "0".
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	60	R
		4	61	IR

Examples: Given: R1 = 07H and register 07H = 0F1H:

COM R1 → R1 = 0F8H
COM @R1 → R1 = 07H, register 07H = 0EH

In the first example, destination working register R1 contains the value 07H (00000111B). The statement "COM R1" complements all the bits in R1: all logic ones are changed to logic zeros, and vice-versa, leaving the value 0F8H (11111000B).

In the second example, Indirect Register (IR) addressing mode is used to complement the value of destination register 07H (11110001B), leaving the new value 0EH (00001110B).

CP — Compare

CP dst,src

Operation: dst – src

The source operand is compared to (subtracted from) the destination operand, and the appropriate flags are set accordingly. The contents of both operands are unaffected by the comparison.

Flags:
C: Set if a "borrow" occurred (src > dst); cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc | dst | src	2	4	A2	r	r
		6	A3	r	lr
opc | src | dst	3	6	A4	R	R
		6	A5	R	IR
opc | dst | src	3	6	A6	R	IM

Examples: 1. Given: R1 = 02H and R2 = 03H:

```
CP    R1,R2  →    Set the C and S flags
```

Destination working register R1 contains the value 02H and source register R2 contains the value 03H. The statement "CP R1,R2" subtracts the R2 value (source/subtrahend) from the R1 value (destination/minuend). Because a "borrow" occurs and the difference is negative, C and S are "1".

2. Given: R1 = 05H and R2 = 0AH:

```
      CP    R1,R2
      JP    UGE,SKIP
      INC   R1
SKIP  LD    R3,R1
```

In this example, destination working register R1 contains the value 05H which is less than the contents of the source working register R2 (0AH). The statement "CP R1,R2" generates C = "1" and the JP instruction does not jump to the SKIP location. After the statement "LD R3,R1" executes, the value 06H remains in working register R3.

CPIJE — Compare, Increment, and Jump on Equal

CPIJE dst,src,RA

Operation: If dst – src = "0", PC ← PC + RA

Ir ← Ir + 1

The source operand is compared to (subtracted from) the destination operand. If the result is "0", the relative address is added to the program counter and control passes to the statement whose address is now in the program counter. Otherwise, the instruction immediately following the CPIJE instruction is executed. In either case, the source pointer is incremented by one before the next instruction is executed.

Flags: No flags are affected.

Format:

				Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	src	dst	RA	3	12	C2	r	Ir

NOTE: Execution time is 18 cycles if the jump is taken or 16 cycles if it is not taken.

Example: Given: R1 = 02H, R2 = 03H, and register 03H = 02H:

CPIJE R1,@R2,SKIP → R2 = 04H, PC jumps to SKIP location

In this example, working register R1 contains the value 02H, working register R2 the value 03H, and register 03 contains 02H. The statement "CPIJE R1,@R2,SKIP" compares the @R2 value 02H (00000010B) to 02H (00000010B). Because the result of the comparison is *equal*, the relative address is added to the PC and the PC then jumps to the memory location pointed to by SKIP. The source register (R2) is incremented by one, leaving a value of 04H. (Remember that the memory location must be within the allowed range of + 127 to – 128.)

CPIJNE — Compare, Increment, and Jump on Non-Equal

CPIJNE dst,src,RA

Operation: If dst – src "0", PC ← PC + RA

Ir ← Ir + 1

The source operand is compared to (subtracted from) the destination operand. If the result is not "0", the relative address is added to the program counter and control passes to the statement whose address is now in the program counter; otherwise the instruction following the CPIJNE instruction is executed. In either case the source pointer is incremented by one before the next instruction.

Flags: No flags are affected.

Format:

				Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	src	dst	RA	3	12	D2	r	Ir

NOTE: Execution time is 18 cycles if the jump is taken or 16 cycles if it is not taken.

Example: Given: R1 = 02H, R2 = 03H, and register 03H = 04H:

CPIJNE R1,@R2,SKIP → R2 = 04H, PC jumps to SKIP location

Working register R1 contains the value 02H, working register R2 (the source pointer) the value 03H, and general register 03 the value 04H. The statement "CPIJNE R1,@R2,SKIP" subtracts 04H (00000100B) from 02H (00000010B). Because the result of the comparison is *non-equal*, the relative address is added to the PC and the PC then jumps to the memory location pointed to by SKIP. The source pointer register (R2) is also incremented by one, leaving a value of 04H. (Remember that the memory location must be within the allowed range of + 127 to – 128.)

DA — Decimal Adjust

DA dst

Operation: dst ← DA dst

The destination operand is adjusted to form two 4-bit BCD digits following an addition or subtraction operation. For addition (ADD, ADC) or subtraction (SUB, SBC), the following table indicates the operation performed. (The operation is undefined if the destination operand was not the result of a valid addition or subtraction of BCD digits):

Instruction	Carry Before DA	Bits 4–7 Value (Hex)	H Flag Before DA	Bits 0–3 Value (Hex)	Number Added to Byte	Carry After DA
	0	0–9	0	0–9	00	0
	0	0–8	0	A–F	06	0
	0	0–9	1	0–3	06	0
ADD	0	A–F	0	0–9	60	1
ADC	0	9–F	0	A–F	66	1
	0	A–F	1	0–3	66	1
	1	0–2	0	0–9	60	1
	1	0–2	0	A–F	66	1
	1	0–3	1	0–3	66	1
	0	0–9	0	0–9	00 = – 00	0
SUB	0	0–8	1	6–F	FA = – 06	0
SBC	1	7–F	0	0–9	A0 = – 60	1
	1	6–F	1	6–F	9A = – 66	1

Flags:
C: Set if there was a carry from the most significant bit; cleared otherwise (see table).
Z: Set if result is "0"; cleared otherwise.
S: Set if result bit 7 is set; cleared otherwise.
V: Undefined.
D: Unaffected.
H: Unaffected.

Format:

opc	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode dst
2	4	40	R
	4	41	IR

DA — Decimal Adjust

DA (Continued)

Example: Given: Working register R0 contains the value 15 (BCD), working register R1 contains 27 (BCD), and address 27H contains 46 (BCD):

```
ADD     R1,R0       ;       C ← "0", H ← "0", Bits 4–7 = 3, bits 0–3 = C, R1 ← 3CH
DA      R1          ;       R1 ← 3CH + 06
```

If addition is performed using the BCD values 15 and 27, the result should be 42. The sum is incorrect, however, when the binary representations are added in the destination location using standard binary arithmetic:

```
    0 0 0 1   0 1 0 1       15
+   0 0 1 0   0 1 1 1       27
    0 0 1 1   1 1 0 0   =   3CH
```

The DA instruction adjusts this result so that the correct BCD representation is obtained:

```
    0 0 1 1   1 1 0 0
+   0 0 0 0   0 1 1 0
    0 1 0 0   0 0 1 0   =   42
```

Assuming the same values given above, the statements

```
SUB     27H,R0 ;        C ← "0", H ← "0", Bits 4–7 = 3, bits 0–3 = 1
DA      @R1     ;       @R1 ← 31–0
```

leave the value 31 (BCD) in address 27H (@R1).

DEC — Decrement

DEC dst

Operation: dst ← dst – 1

The contents of the destination operand are decremented by one.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	00	R
		4	01	IR

Examples: Given: R1 = 03H and register 03H = 10H:

DEC R1 → R1 = 02H
DEC @R1 → Register 03H = 0FH

In the first example, if working register R1 contains the value 03H, the statement "DEC R1" decrements the hexadecimal value by one, leaving the value 02H. In the second example, the statement "DEC @R1" decrements the value 10H contained in the destination register 03H by one, leaving the value 0FH.

DECW — Decrement Word

DECW dst

Operation: dst ← dst – 1

The contents of the destination location (which must be an even address) and the operand following that location are treated as a single 16-bit value that is decremented by one.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc dst	2	8	80	RR
		8	81	IR

Examples: Given: R0 = 12H, R1 = 34H, R2 = 30H, register 30H = 0FH, and register 31H = 21H:

DECW RR0 → R0 = 12H, R1 = 33H
DECW @R2 → Register 30H = 0FH, register 31H = 20H

In the first example, destination register R0 contains the value 12H and register R1 the value 34H. The statement "DECW RR0" addresses R0 and the following operand R1 as a 16-bit word and decrements the value of R1 by one, leaving the value 33H.

NOTE: A system malfunction may occur if you use a Zero flag (FLAGS.6) result together with a DECW instruction. To avoid this problem, we recommend that you use DECW as shown in the following example:

```
LOOP: DECW  RR0
      LD    R2,R1
      OR    R2,R0
      JR    NZ,LOOP
```

DI — Disable Interrupts

DI

Operation: SYM (0) ← 0

Bit zero of the system mode control register, SYM.0, is cleared to "0", globally disabling all interrupt processing. Interrupt requests will continue to set their respective interrupt pending bits, but the CPU will not service them while interrupt processing is disabled.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	8F

Example: Given: SYM = 01H:

DI

If the value of the SYM register is 01H, the statement "DI" leaves the new value 00H in the register and clears SYM.0 to "0", disabling interrupt processing.

Before changing IMR, interrupt pending and interrupt source control register, be sure DI state.

DIV — Divide (Unsigned)

DIV dst,src

Operation: dst ÷ src

dst (UPPER) ← REMAINDER

dst (LOWER) ← QUOTIENT

The destination operand (16 bits) is divided by the source operand (8 bits). The quotient (8 bits) is stored in the lower half of the destination. The remainder (8 bits) is stored in the upper half of the destination. When the quotient is ≥ 2^8, the numbers stored in the upper and lower halves of the destination for quotient and remainder are incorrect. Both operands are treated as unsigned integers.

Flags:
C: Set if the V flag is set and quotient is between 2^8 and 2^9 –1; cleared otherwise.
Z: Set if divisor or quotient = "0"; cleared otherwise.
S: Set if MSB of quotient = "1"; cleared otherwise.
V: Set if quotient is ≥ 2^8 or if divisor = "0"; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc src dst	3	26/10	94	RR	R
		26/10	95	RR	IR
		26/10	96	RR	IM

NOTE: Execution takes 10 cycles if the divide-by-zero is attempted; otherwise it takes 26 cycles.

Examples: Given: R0 = 10H, R1 = 03H, R2 = 40H, register 40H = 80H:

DIV RR0,R2 → R0 = 03H, R1 = 40H
DIV RR0,@R2 → R0 = 03H, R1 = 20H
DIV RR0,#20H → R0 = 03H, R1 = 80H

In the first example, destination working register pair RR0 contains the values 10H (R0) and 03H (R1), and register R2 contains the value 40H. The statement "DIV RR0,R2" divides the 16-bit RR0 value by the 8-bit value of the R2 (source) register. After the DIV instruction, R0 contains the value 03H and R1 contains 40H. The 8-bit remainder is stored in the upper half of the destination register RR0 (R0) and the quotient in the lower half (R1).

DJNZ — Decrement and Jump if Non-Zero

DJNZ r,dst

Operation: r ← r – 1

If r ≠ 0, PC ← PC + dst

The working register being used as a counter is decremented. If the contents of the register are not logic zero after decrementing, the relative address is added to the program counter and control passes to the statement whose address is now in the PC. The range of the relative address is +127 to –128, and the original value of the PC is taken to be the address of the instruction byte following the DJNZ statement.

NOTE: In case of using DJNZ instruction, the working register being used as a counter should be set at the one of location 0C0H to 0CFH with SRP, SRP0, or SRP1 instruction.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
r | opc | dst	2	8 (jump taken)	rA	RA
		8 (no jump)	r = 0 to F	

Example: Given: R1 = 02H and LOOP is the label of a relative address:

```
SRP    #0C0H
DJNZ   R1,LOOP
```

DJNZ is typically used to control a "loop" of instructions. In many cases, a label is used as the destination operand instead of a numeric relative address value. In the example, working register R1 contains the value 02H, and LOOP is the label for a relative address.

The statement "DJNZ R1, LOOP" decrements register R1 by one, leaving the value 01H. Because the contents of R1 after the decrement are non-zero, the jump is taken to the relative address specified by the LOOP label.

EI — Enable Interrupts

EI

Operation: SYM (0) ← 1

An EI instruction sets bit zero of the system mode register, SYM.0 to "1". This allows interrupts to be serviced as they occur (assuming they have highest priority). If an interrupt's pending bit was set while interrupt processing was disabled (by executing a DI instruction), it will be serviced when you execute the EI instruction.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	9F

Example: Given: SYM = 00H:

EI

If the SYM register contains the value 00H, that is, if interrupts are currently disabled, the statement "EI" sets the SYM register to 01H, enabling all interrupts. (SYM.0 is the enable bit for global interrupt processing.)

ENTER — Enter

ENTER

Operation:

SP ← SP – 2
@SP ← IP
IP ← PC
PC ← @IP
IP ← IP + 2

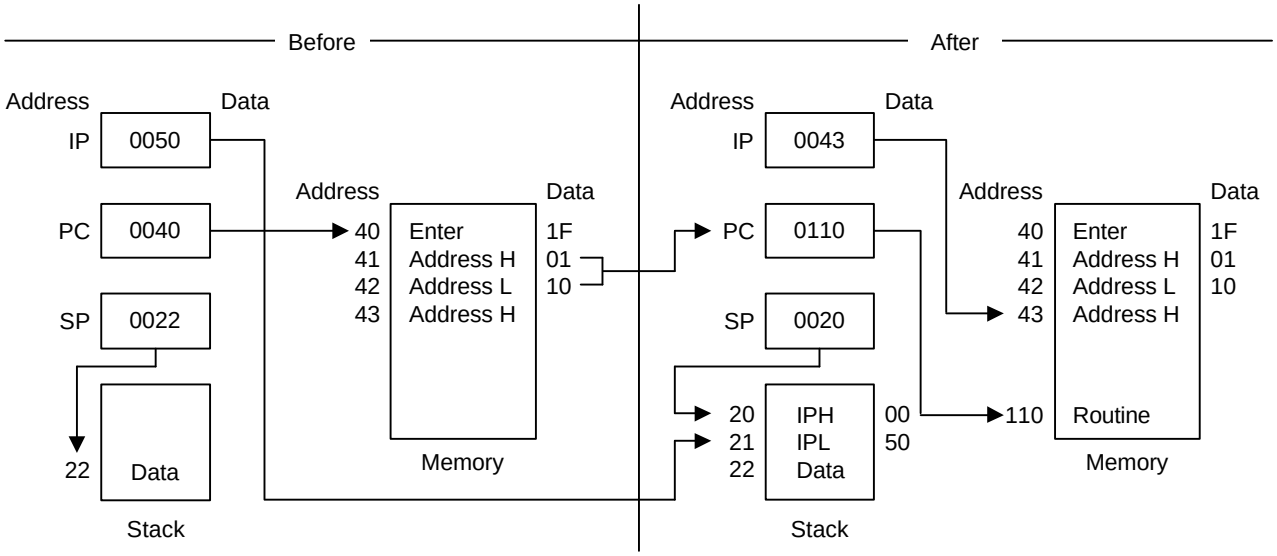
This instruction is useful when implementing threaded-code languages. The contents of the instruction pointer are pushed to the stack. The program counter (PC) value is then written to the instruction pointer. The program memory word that is pointed to by the instruction pointer is loaded into the PC, and the instruction pointer is incremented by two.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	14	1F

Example: The diagram below shows one example of how to use an ENTER statement.



EXIT — Exit

EXIT

Operation:

IP ← @SP
SP ← SP + 2
PC ← @IP
IP ← IP + 2

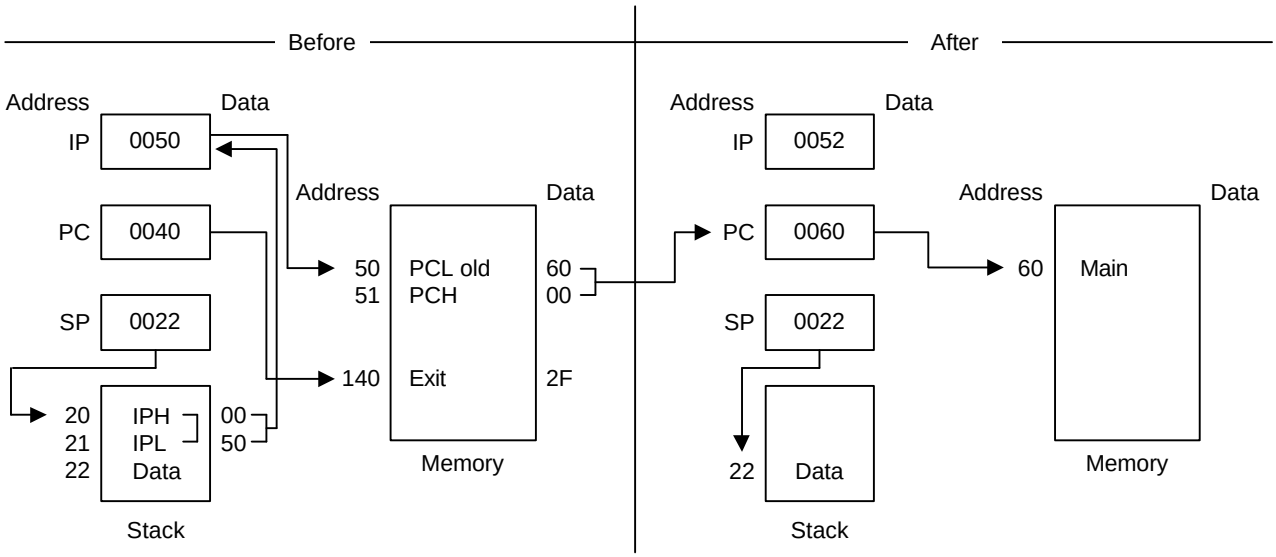
This instruction is useful when implementing threaded-code languages. The stack value is popped and loaded into the instruction pointer. The program memory word that is pointed to by the instruction pointer is then loaded into the program counter, and the instruction pointer is incremented by two.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	14 (internal stack) 16 (internal stack)	2F

Example: The diagram below shows one example of how to use an EXIT statement.



IDLE — Idle Operation

IDLE

Operation:

The IDLE instruction stops the CPU clock while allowing system clock oscillation to continue. Idle mode can be released by an interrupt request (IRQ) or an external reset operation.
In application programs, a IDLE instruction must be immediately followed by at least three NOP instructions. This ensures an adeguate time interval for the clock to stabilize before the next instruction is executed. If three or more NOP instructons are not used after IDLE instruction, leakage current could be flown because of the floating state in the internal bus.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	1	4	6F	–	–

Example: The instruction

```
IDLE          ; stops the CPU clock but not the system clock
NOP
NOP
NOP
```

INC — Increment

INC dst

Operation: dst ← dst + 1

The contents of the destination operand are incremented by one.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
dst | opc	1	4	rE r = 0 to F	r
opc | dst	2	4	20	R
		4	21	IR

Examples: Given: R0 = 1BH, register 00H = 0CH, and register 1BH = 0FH:

INC R0 → R0 = 1CH
INC 00H → Register 00H = 0DH
INC @R0 → R0 = 1BH, register 01H = 10H

In the first example, if destination working register R0 contains the value 1BH, the statement "INC R0" leaves the value 1CH in that same register.

The next example shows the effect an INC instruction has on register 00H, assuming that it contains the value 0CH.

In the third example, INC is used in Indirect Register (IR) addressing mode to increment the value of register 1BH from 0FH to 10H.

INCW — Increment Word

INCW dst

Operation: dst ← dst + 1

The contents of the destination (which must be an even address) and the byte following that location are treated as a single 16-bit value that is incremented by one.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	8	A0	RR
		8	A1	IR

Examples: Given: R0 = 1AH, R1 = 02H, register 02H = 0FH, and register 03H = 0FFH:

INCW RR0 → R0 = 1AH, R1 = 03H
INCW @R1 → Register 02H = 10H, register 03H = 00H

In the first example, the working register pair RR0 contains the value 1AH in register R0 and 02H in register R1. The statement "INCW RR0" increments the 16-bit destination by one, leaving the value 03H in register R1. In the second example, the statement "INCW @R1" uses Indirect Register (IR) addressing mode to increment the contents of general register 03H from 0FFH to 00H and register 02H from 0FH to 10H.

NOTE: A system malfunction may occur if you use a Zero (Z) flag (FLAGS.6) result together with an INCW instruction. To avoid this problem, we recommend that you use INCW as shown in the following example:

```
LOOP:   INCW    RR0
        LD      R2,R1
        OR      R2,R0
        JR      NZ,LOOP
```

IRET — Interrupt Return

IRET IRET (Normal) IRET (Fast)

Operation:

IRET (Normal)	IRET (Fast)
FLAGS ← @SP	PC ↔ IP
SP ← SP + 1	FLAGS ← FLAGS'
PC ← @SP	FIS ← 0
SP ← SP + 2	
SYM(0) ← 1	

This instruction is used at the end of an interrupt service routine. It restores the flag register and the program counter. It also re-enables global interrupts. A "normal IRET" is executed only if the fast interrupt status bit (FIS, bit one of the FLAGS register, 0D5H) is cleared (= "0"). If a fast interrupt occurred, IRET clears the FIS bit that was set at the beginning of the service routine.

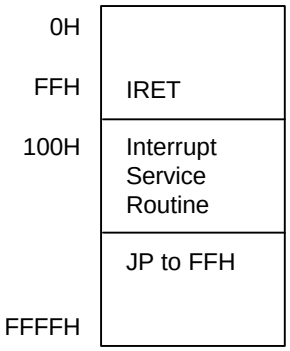
Flags: All flags are restored to their original settings (that is, the settings before the interrupt occurred).

Format:

IRET (Normal)	Bytes	Cycles	Opcode (Hex)
opc	1	10 (internal stack) 12 (internal stack)	BF

IRET (Fast)	Bytes	Cycles	Opcode (Hex)
opc	1	6	BF

Example: In the figure below, the instruction pointer is initially loaded with 100H in the main program before interrupts are enabled. When an interrupt occurs, the program counter and instruction pointer are swapped. This causes the PC to jump to address 100H and the IP to keep the return address. The last instruction in the service routine normally is a jump to IRET at address FFH. This causes the instruction pointer to be loaded with 100H "again" and the program counter to jump back to the main program. Now, the next interrupt can occur and the IP is still correct at 100H.



NOTE: In the fast interrupt example above, if the last instruction is not a jump to IRET, you must pay attention to the order of the last two instructions. The IRET cannot be immediately proceeded by a clearing of the interrupt status (as with a reset of the IPR register).

JP — Jump

JP cc,dst (Conditional)

JP dst (Unconditional)

Operation: If cc is true, PC ← dst

The conditional JUMP instruction transfers program control to the destination address if the condition specified by the condition code (cc) is true; otherwise, the instruction following the JP instruction is executed. The unconditional JP simply replaces the contents of the PC with the contents of the specified register pair. Control then passes to the statement addressed by the PC.

Flags: No flags are affected.

Format: (1)

(2)		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
cc | opc	dst	3	8	ccD cc = 0 to F	DA
opc	dst	2	8	30	IRR

NOTES:
1. The 3-byte format is used for a conditional jump and the 2-byte format for an unconditional jump.
2. In the first byte of the three-byte instruction format (conditional jump), the condition code and the opcode are both four bits.

Examples: Given: The carry flag (C) = "1", register 00 = 01H, and register 01 = 20H:

JP C,LABEL_W → LABEL_W = 1000H, PC = 1000H

JP @00H → PC = 0120H

The first example shows a conditional JP. Assuming that the carry flag is set to "1", the statement
"JP C,LABEL_W" replaces the contents of the PC with the value 1000H and transfers control to that location. Had the carry flag not been set, control would then have passed to the statement immediately following the JP instruction.

The second example shows an unconditional JP. The statement "JP @00" replaces the contents of the PC with the contents of the register pair 00H and 01H, leaving the value 0120H.

JR — Jump Relative

JR cc,dst

Operation: If cc is true, PC ← PC + dst

If the condition specified by the condition code (cc) is true, the relative address is added to the program counter and control passes to the statement whose address is now in the program counter; otherwise, the instruction following the JR instruction is executed. (See list of condition codes).

The range of the relative address is +127, –128, and the original value of the program counter is taken to be the address of the first instruction byte following the JR statement.

Flags: No flags are affected.

Format:

(1)	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
cc | opc | dst	2	6	ccB	RA
			cc = 0 to F	

NOTE: In the first byte of the two-byte instruction format, the condition code and the opcode are each four bits.

Example: Given: The carry flag = "1" and LABEL_X = 1FF7H:

JR C,LABEL_X → PC = 1FF7H

If the carry flag is set (that is, if the condition code is true), the statement "JR C,LABEL_X" will pass control to the statement whose address is now in the PC. Otherwise, the program instruction following the JR would be executed.

LD — Load

LD dst,src

Operation: dst ← src

The contents of the source are loaded into the destination. The source's contents are unaffected.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
dst | opc / src	2	4	rC	r	IM
		4	r8	r	R
src | opc / dst	2	4	r9	R	r
			r = 0 to F		
opc / dst | src	2	4	C7	r	lr
		4	D7	lr	r
opc / src / dst	3	6	E4	R	R
		6	E5	R	IR
opc / dst / src	3	6	E6	R	IM
		6	D6	IR	IM
opc / src / dst	3	6	F5	IR	R
opc / dst | src / x	3	6	87	r	x [r]
opc / src | dst / x	3	6	97	x [r]	r

LD — Load

LD (Continued)

Examples: Given: R0 = 01H, R1 = 0AH, register 00H = 01H, register 01H = 20H, register 02H = 02H, LOOP = 30H, and register 3AH = 0FFH:

LD	R0,#10H	→	R0 = 10H
LD	R0,01H	→	R0 = 20H, register 01H = 20H
LD	01H,R0	→	Register 01H = 01H, R0 = 01H
LD	R1,@R0	→	R1 = 20H, R0 = 01H
LD	@R0,R1	→	R0 = 01H, R1 = 0AH, register 01H = 0AH
LD	00H,01H	→	Register 00H = 20H, register 01H = 20H
LD	02H,@00H	→	Register 02H = 20H, register 00H = 01H
LD	00H,#0AH	→	Register 00H = 0AH
LD	@00H,#10H	→	Register 00H = 01H, register 01H = 10H
LD	@00H,02H	→	Register 00H = 01H, register 01H = 02, register 02H = 02H
LD	R0,#LOOP[R1]	→	R0 = 0FFH, R1 = 0AH
LD	#LOOP[R0],R1	→	Register 31H = 0AH, R0 = 01H, R1 = 0AH

LDB — Load Bit

LDB dst,src.b

LDB dst.b,src

Operation: dst(0) ← src(b)

or

dst(b) ← src(0)

The specified bit of the source is loaded into bit zero (LSB) of the destination, or bit zero of the source is loaded into the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags: No flags are affected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | b | 0	src	3	6	47	r0	Rb
opc	src | b | 1	dst	3	6	47	Rb	r0

NOTE: In the second byte of the instruction formats, the destination (or source) address is four bits, the bit address 'b' is three bits, and the LSB address value is one bit in length.

Examples: Given: R0 = 06H and general register 00H = 05H:

LDB R0,00H.2 → R0 = 07H, register 00H = 05H

LDB 00H.0,R0 → R0 = 06H, register 00H = 04H

In the first example, destination working register R0 contains the value 06H and the source general register 00H the value 05H. The statement "LD R0,00H.2" loads the bit two value of the 00H register into bit zero of the R0 register, leaving the value 07H in register R0.

In the second example, 00H is the destination register. The statement "LD 00H.0,R0" loads bit zero of register R0 to the specified bit (bit zero) of the destination register, leaving 04H in general register 00H.

LDC/LDE — Load Memory

LDC/LDE dst,src

Operation: dst ← src

This instruction loads a byte from program or data memory into a working register or vice-versa. The source values are unaffected. LDC refers to program memory and LDE to data memory. The assembler makes 'Irr' or 'rr' values an even number for program memory and odd an odd number for data memory.

Flags: No flags are affected.

Format:

					Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
1.	opc	dst | src			2	10	C3	r	Irr
2.	opc	src | dst			2	10	D3	Irr	r
3.	opc	dst | src	XS		3	12	E7	r	XS [rr]
4.	opc	src | dst	XS		3	12	F7	XS [rr]	r
5.	opc	dst | src	XL_L	XL_H	4	14	A7	r	XL [rr]
6.	opc	src | dst	XL_L	XL_H	4	14	B7	XL [rr]	r
7.	opc	dst | 0000	DA_L	DA_H	4	14	A7	r	DA
8.	opc	src | 0000	DA_L	DA_H	4	14	B7	DA	r
9.	opc	dst | 0001	DA_L	DA_H	4	14	A7	r	DA
10.	opc	src | 0001	DA_L	DA_H	4	14	B7	DA	r

NOTES:

1. The source (src) or working register pair [rr] for formats 5 and 6 cannot use register pair 0–1.
2. For formats 3 and 4, the destination address 'XS [rr]' and the source address 'XS [rr]' are each one byte.
3. For formats 5 and 6, the destination address 'XL [rr] and the source address 'XL [rr]' are each two bytes.
4. The DA and r source values for formats 7 and 8 are used to address program memory; the second set of values, used in formats 9 and 10, are used to address data memory.

LDC/LDE — Load Memory

LDC/LDE (Continued)

Examples: Given: R0 = 11H, R1 = 34H, R2 = 01H, R3 = 04H; Program memory locations 0103H = 4FH, 0104H = 1A, 0105H = 6DH, and 1104H = 88H. External data memory locations 0103H = 5FH, 0104H = 2AH, 0105H = 7DH, and 1104H = 98H:

```
LDC     R0,@RR2          ; R0 ← contents of program memory location 0104H
                         ; R0 = 1AH, R2 = 01H, R3 = 04H

LDE     R0,@RR2          ; R0 ← contents of external data memory location 0104H
                         ; R0 = 2AH, R2 = 01H, R3 = 04H

LDC (note) @RR2,R0       ; 11H (contents of R0) is loaded into program memory
                         ; location 0104H (RR2),
                         ; working registers R0, R2, R3 → no change

LDE     @RR2,R0          ; 11H (contents of R0) is loaded into external data memory
                         ; location 0104H (RR2),
                         ; working registers R0, R2, R3 → no change

LDC     R0,#01H[RR2]     ; R0 ← contents of program memory location 0105H
                         ; (01H + RR2),
                         ; R0 = 6DH, R2 = 01H, R3 = 04H

LDE     R0,#01H[RR2]     ; R0 ← contents of external data memory location 0105H
                         ; (01H + RR2), R0 = 7DH, R2 = 01H, R3 = 04H

LDC (note) #01H[RR2],R0  ; 11H (contents of R0) is loaded into program memory location
                         ; 0105H (01H + 0104H)

LDE     #01H[RR2],R0     ; 11H (contents of R0) is loaded into external data memory
                         ; location 0105H (01H + 0104H)

LDC     R0,#1000H[RR2]   ; R0 ← contents of program memory location 1104H
                         ; (1000H + 0104H), R0 = 88H, R2 = 01H, R3 = 04H

LDE     R0,#1000H[RR2]   ; R0 ← contents of external data memory location 1104H
                         ; (1000H + 0104H), R0 = 98H, R2 = 01H, R3 = 04H

LDC     R0,1104H         ; R0 ← contents of program memory location 1104H, R0 = 88H

LDE     R0,1104H         ; R0 ← contents of external data memory location 1104H,
                         ; R0 = 98H

LDC (note) 1105H,R0      ; 11H (contents of R0) is loaded into program memory location
                         ; 1105H, (1105H) ← 11H

LDE     1105H,R0         ; 11H (contents of R0) is loaded into external data memory
                         ; location 1105H, (1105H) ← 11H
```

NOTE: These instructions are not supported by masked ROM type devices.

LDCD/LDED — Load Memory and Decrement

LDCD/LDED dst,src

Operation: dst ← src

rr ← rr – 1

These instructions are used for user stacks or block transfers of data from program or data memory to the register file. The address of the memory location is specified by a working register pair. The contents of the source location are loaded into the destination location. The memory address is then decremented. The contents of the source are unaffected.

LDCD references program memory and LDED references external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for data memory.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | src	2	10	E2	r	Irr

Examples: Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory location 1033H = 0CDH, and external data memory location 1033H = 0DDH:

```
LDCD    R8,@RR6     ; 0CDH (contents of program memory location 1033H) is loaded
                    ; into R8 and RR6 is decremented by one
                    ; R8 = 0CDH, R6 = 10H, R7 = 32H (RR6 ← RR6 – 1)

LDED    R8,@RR6     ; 0DDH (contents of data memory location 1033H) is loaded
                    ; into R8 and RR6 is decremented by one (RR6 ← RR6 – 1)
                    ; R8 = 0DDH, R6 = 10H, R7 = 32H
```

LDCI/LDEI — Load Memory and Increment

LDCI/LDEI dst,src

Operation: dst ← src

rr ← rr + 1

These instructions are used for user stacks or block transfers of data from program or data memory to the register file. The address of the memory location is specified by a working register pair. The contents of the source location are loaded into the destination location. The memory address is then incremented automatically. The contents of the source are unaffected.

LDCI refers to program memory and LDEI refers to external data memory. The assembler makes 'Irr' even for program memory and odd for data memory.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | src	2	10	E3	r	Irr

Examples: Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory locations 1033H = 0CDH and 1034H = 0C5H; external data memory locations 1033H = 0DDH and 1034H = 0D5H:

```
LDCI    R8,@RR6    ; 0CDH (contents of program memory location 1033H) is loaded
                   ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                   ; R8 = 0CDH, R6 = 10H, R7 = 34H

LDEI    R8,@RR6    ; 0DDH (contents of data memory location 1033H) is loaded
                   ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                   ; R8 = 0DDH, R6 = 10H, R7 = 34H
```

LDCPD/LDEPD — Load Memory with Pre-Decrement

LDCPD/ LDEPD dst,src

Operation: rr ← rr – 1

dst ← src

These instructions are used for block transfers of data from program or data memory from the register file. The address of the memory location is specified by a working register pair and is first decremented. The contents of the source location are then loaded into the destination location. The contents of the source are unaffected.

LDCPD refers to program memory and LDEPD refers to external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for external data memory.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	src | dst	2	14	F2	Irr	r

Examples: Given: R0 = 77H, R6 = 30H, and R7 = 00H:

```
LDCPD   @RR6,R0     ; (RR6 ← RR6 – 1)
                    ; 77H (contents of R0) is loaded into program memory location
                    ; 2FFFH (3000H – 1H)
                    ; R0 = 77H, R6 = 2FH, R7 = 0FFH

LDEPD   @RR6,R0     ; (RR6 ← RR6 – 1)
                    ; 77H (contents of R0) is loaded into external data memory
                    ; location 2FFFH (3000H – 1H)
                    ; R0 = 77H, R6 = 2FH, R7 = 0FFH
```

LDCPI/LDEPI — Load Memory with Pre-Increment

LDCPI/ LDEPI dst,src

Operation: rr ← rr + 1

dst ← src

These instructions are used for block transfers of data from program or data memory from the register file. The address of the memory location is specified by a working register pair and is first incremented. The contents of the source location are loaded into the destination location. The contents of the source are unaffected.

LDCPI refers to program memory and LDEPI refers to external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for data memory.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	src | dst	2	14	F3	Irr	r

Examples: Given: R0 = 7FH, R6 = 21H, and R7 = 0FFH:

```
LDCPI   @RR6,R0     ; (RR6 ← RR6 + 1)
                    ; 7FH (contents of R0) is loaded into program memory
                    ; location 2200H (21FFH + 1H)
                    ; R0 = 7FH, R6 = 22H, R7 = 00H

LDEPI   @RR6,R0     ; (RR6 ← RR6 + 1)
                    ; 7FH (contents of R0) is loaded into external data memory
                    ; location 2200H (21FFH + 1H)
                    ; R0 = 7FH, R6 = 22H, R7 = 00H
```

LDW — Load Word

LDW dst,src

Operation: dst ← src

The contents of the source (a word) are loaded into the destination. The contents of the source are unaffected.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | src | dst	3	8	C4	RR	RR
		8	C5	RR	IR
opc | dst | src	4	8	C6	RR	IML

Examples: Given: R4 = 06H, R5 = 1CH, R6 = 05H, R7 = 02H, register 00H = 1AH, register 01H = 02H, register 02H = 03H, and register 03H = 0FH:

LDW RR6,RR4 → R6 = 06H, R7 = 1CH, R4 = 06H, R5 = 1CH

LDW 00H,02H → Register 00H = 03H, register 01H = 0FH, register 02H = 03H, register 03H = 0FH

LDW RR2,@R7 → R2 = 03H, R3 = 0FH,

LDW 04H,@01H → Register 04H = 03H, register 05H = 0FH

LDW RR6,#1234H → R6 = 12H, R7 = 34H

LDW 02H,#0FEDH → Register 02H = 0FH, register 03H = 0EDH

In the second example, please note that the statement "LDW 00H,02H" loads the contents of the source word 02H, 03H into the destination word 00H, 01H. This leaves the value 03H in general register 00H and the value 0FH in register 01H.

The other examples show how to use the LDW instruction with various addressing modes and formats.

MULT — Multiply (Unsigned)

MULT dst,src

Operation: dst ← dst × src

The 8-bit destination operand (even register of the register pair) is multiplied by the source operand (8 bits) and the product (16 bits) is stored in the register pair specified by the destination address. Both operands are treated as unsigned integers.

Flags:
C: Set if result is > 255; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if MSB of the result is a "1"; cleared otherwise.
V: Cleared.
D: Unaffected.
H: Unaffected.

Format:

opc	src	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
3	22	84	RR	R
	22	85	RR	IR
	22	86	RR	IM

Examples: Given: Register 00H = 20H, register 01H = 03H, register 02H = 09H, register 03H = 06H:

MULT 00H, 02H → Register 00H = 01H, register 01H = 20H, register 02H = 09H
MULT 00H, @01H → Register 00H = 00H, register 01H = 0C0H
MULT 00H, #30H → Register 00H = 06H, register 01H = 00H

In the first example, the statement "MULT 00H,02H" multiplies the 8-bit destination operand (in the register 00H of the register pair 00H, 01H) by the source register 02H operand (09H). The 16-bit product, 0120H, is stored in the register pair 00H, 01H.

NEXT — Next

NEXT

Operation: PC ← @ IP

IP ← IP + 2

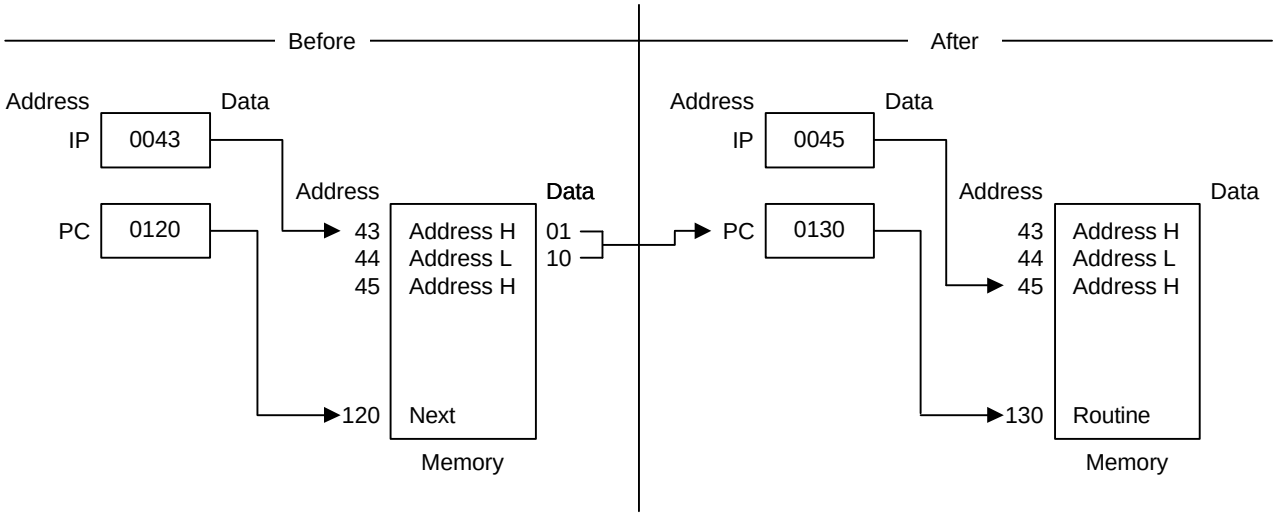
The NEXT instruction is useful when implementing threaded-code languages. The program memory word that is pointed to by the instruction pointer is loaded into the program counter. The instruction pointer is then incremented by two.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	10	0F

Example: The following diagram shows one example of how to use the NEXT instruction.



NOP — No Operation

NOP

Operation: No action is performed when the CPU executes this instruction. Typically, one or more NOPs are executed in sequence in order to effect a timing delay of variable duration.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	FF

Example: When the instruction

NOP

is encountered in a program, no operation occurs. Instead, there is a delay in instruction execution time.

OR — Logical OR

OR dst,src

Operation: dst ← dst OR src

The source operand is logically ORed with the destination operand and the result is stored in the destination. The contents of the source are unaffected. The OR operation results in a "1" being stored whenever either of the corresponding bits in the two operands is a "1"; otherwise a "0" is stored.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always cleared to "0".
D: Unaffected.
H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	42	r	r
				6	43	r	lr
opc	src	dst	3	6	44	R	R
				6	45	R	IR
opc	dst	src	3	6	46	R	IM

Examples: Given: R0 = 15H, R1 = 2AH, R2 = 01H, register 00H = 08H, register 01H = 37H, and register 08H = 8AH:

OR R0,R1 → R0 = 3FH, R1 = 2AH
OR R0,@R2 → R0 = 37H, R2 = 01H, register 01H = 37H
OR 00H,01H → Register 00H = 3FH, register 01H = 37H
OR 01H,@00H → Register 00H = 08H, register 01H = 0BFH
OR 00H,#02H → Register 00H = 0AH

In the first example, if working register R0 contains the value 15H and register R1 the value 2AH, the statement "OR R0,R1" logical-ORs the R0 and R1 register contents and stores the result (3FH) in destination register R0.

The other examples show the use of the logical OR instruction with the various addressing modes and formats.

POP — Pop From Stack

POP dst

Operation: dst ← @SP

SP ← SP + 1

The contents of the location addressed by the stack pointer are loaded into the destination. The stack pointer is then incremented by one.

Flags: No flags affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	8	50	R
		8	51	IR

Examples: Given: Register 00H = 01H, register 01H = 1BH, SPH (0D8H) = 00H, SPL (0D9H) = 0FBH, and stack register 0FBH = 55H:

POP 00H → Register 00H = 55H, SP = 00FCH

POP @00H → Register 00H = 01H, register 01H = 55H, SP = 00FCH

In the first example, general register 00H contains the value 01H. The statement "POP 00H" loads the contents of location 00FBH (55H) into destination register 00H and then increments the stack pointer by one. Register 00H then contains the value 55H and the SP points to location 00FCH.

POPUD — Pop User Stack (Decrementing)

POPUD dst,src

Operation: dst ← src

IR ← IR – 1

This instruction is used for user-defined stacks in the register file. The contents of the register file location addressed by the user stack pointer are loaded into the destination. The user stack pointer is then decremented.

Flags: No flags are affected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	src	dst	3	8	92	R	IR

Example: Given: Register 00H = 42H (user stack pointer register), register 42H = 6FH, and register 02H = 70H:

POPUD 02H,@00H → Register 00H = 41H, register 02H = 6FH, register 42H = 6FH

If general register 00H contains the value 42H and register 42H the value 6FH, the statement "POPUD 02H,@00H" loads the contents of register 42H into the destination register 02H. The user stack pointer is then decremented by one, leaving the value 41H.

POPUI — Pop User Stack (Incrementing)

POPUI dst,src

Operation: dst ← src

IR ← IR + 1

The POPUI instruction is used for user-defined stacks in the register file. The contents of the register file location addressed by the user stack pointer are loaded into the destination. The user stack pointer is then incremented.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc src dst	3	8	93	R	IR

Example: Given: Register 00H = 01H and register 01H = 70H:

POPUI 02H,@00H → Register 00H = 02H, register 01H = 70H, register 02H = 70H

If general register 00H contains the value 01H and register 01H the value 70H, the statement "POPUI 02H,@00H" loads the value 70H into the destination general register 02H. The user stack pointer (register 00H) is then incremented by one, changing its value from 01H to 02H.

PUSH — Push To Stack

PUSH src

Operation: SP ← SP – 1

@SP ← src

A PUSH instruction decrements the stack pointer value and loads the contents of the source (src) into the location addressed by the decremented stack pointer. The operation then adds the new value to the top of the stack.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc src	2	8 (internal clock)	70	R
		8 (external clock)		
		8 (internal clock)		
		8 (external clock)	71	IR

Examples: Given: Register 40H = 4FH, register 4FH = 0AAH, SPH = 00H, and SPL = 00H:

PUSH 40H → Register 40H = 4FH, stack register 0FFH = 4FH, SPH = 0FFH, SPL = 0FFH

PUSH @40H → Register 40H = 4FH, register 4FH = 0AAH, stack register 0FFH = 0AAH, SPH = 0FFH, SPL = 0FFH

In the first example, if the stack pointer contains the value 0000H, and general register 40H the value 4FH, the statement "PUSH 40H" decrements the stack pointer from 0000 to 0FFFFH. It then loads the contents of register 40H into location 0FFFFH and adds this new value to the top of the stack.

PUSHUD — Push User Stack (Decrementing)

PUSHUD dst,src

Operation: IR ← IR – 1

dst ← src

This instruction is used to address user-defined stacks in the register file. PUSHUD decrements the user stack pointer and loads the contents of the source into the register addressed by the decremented stack pointer.

Flags: No flags are affected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst	src	3	8	82	IR	R

Example: Given: Register 00H = 03H, register 01H = 05H, and register 02H = 1AH:

PUSHUD @00H,01H → Register 00H = 02H, register 01H = 05H, register 02H = 05H

If the user stack pointer (register 00H, for example) contains the value 03H, the statement "PUSHUD @00H,01H" decrements the user stack pointer by one, leaving the value 02H. The 01H register value, 05H, is then loaded into the register addressed by the decremented user stack pointer.

PUSHUI — Push User Stack (Incrementing)

PUSHUI dst,src

Operation: IR ← IR + 1

dst ← src

This instruction is used for user-defined stacks in the register file. PUSHUI increments the user stack pointer and then loads the contents of the source into the register location addressed by the incremented user stack pointer.

Flags: No flags are affected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst	src	3	8	83	IR	R

Example: Given: Register 00H = 03H, register 01H = 05H, and register 04H = 2AH:

PUSHUI @00H,01H → Register 00H = 04H, register 01H = 05H, register 04H = 05H

If the user stack pointer (register 00H, for example) contains the value 03H, the statement "PUSHUI @00H,01H" increments the user stack pointer by one, leaving the value 04H. The 01H register value, 05H, is then loaded into the location addressed by the incremented user stack pointer.

RCF — Reset Carry Flag

RCF RCF

Operation: C ← 0

The carry flag is cleared to logic zero, regardless of its previous value.

Flags: **C:** Cleared to "0".

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	CF

Example: Given: C = "1" or "0":

The instruction RCF clears the carry flag (C) to logic zero.

RET — Return

RET

Operation: PC ← @SP

SP ← SP + 2

The RET instruction is normally used to return to the previously executing procedure at the end of a procedure entered by a CALL instruction. The contents of the location addressed by the stack pointer are popped into the program counter. The next statement that is executed is the one that is addressed by the new program counter value.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	8 (internal stack) 10 (internal stack)	AF

Example: Given: SP = 00FCH, (SP) = 101AH, and PC = 1234:

RET → PC = 101AH, SP = 00FEH

The statement "RET" pops the contents of stack pointer location 00FCH (10H) into the high byte of the program counter. The stack pointer then pops the value in location 00FEH (1AH) into the PC's low byte and the instruction at location 101AH is executed. The stack pointer now points to memory location 00FEH.

RL — Rotate Left

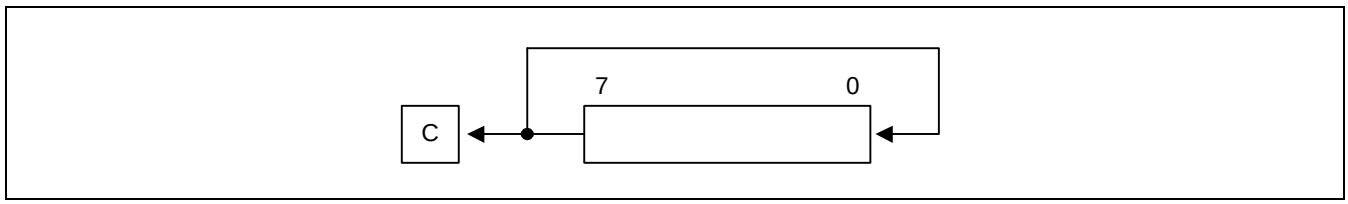
RL dst

Operation: C ← dst (7)

dst (0) ← dst (7)

dst (n + 1) ← dst (n), n = 0–6

The contents of the destination operand are rotated left one bit position. The initial value of bit 7 is moved to the bit zero (LSB) position and also replaces the carry flag.



Flags:
C: Set if the bit rotated from the most significant bit position (bit 7) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc dst	2	4	90	R
		4	91	IR

Examples: Given: Register 00H = 0AAH, register 01H = 02H and register 02H = 17H:

RL 00H → Register 00H = 55H, C = "1"

RL @01H → Register 01H = 02H, register 02H = 2EH, C = "0"

In the first example, if general register 00H contains the value 0AAH (10101010B), the statement "RL 00H" rotates the 0AAH value left one bit position, leaving the new value 55H (01010101B) and setting the carry and overflow flags.

RLC — Rotate Left Through Carry

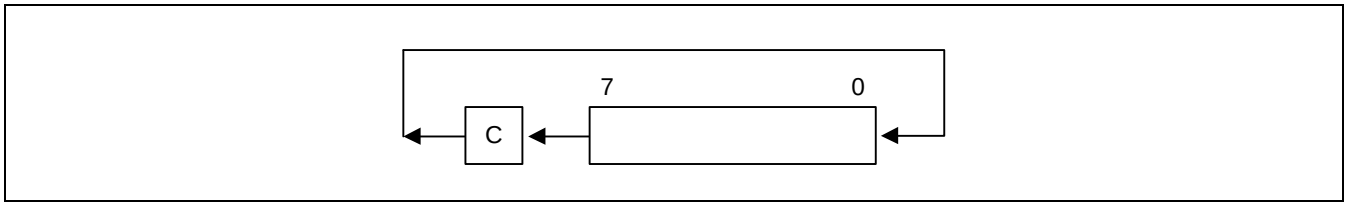
RLC dst

Operation: dst (0) ← C

C ← dst (7)

dst (n + 1) ← dst (n), n = 0–6

The contents of the destination operand with the carry flag are rotated left one bit position. The initial value of bit 7 replaces the carry flag (C); the initial value of the carry flag replaces bit zero.



Flags:
C: Set if the bit rotated from the most significant bit position (bit 7) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc dst	2	4	10	R
		4	11	IR

Examples: Given: Register 00H = 0AAH, register 01H = 02H, and register 02H = 17H, C = "0":

RLC 00H → Register 00H = 54H, C = "1"
RLC @01H → Register 01H = 02H, register 02H = 2EH, C = "0"

In the first example, if general register 00H has the value 0AAH (10101010B), the statement "RLC 00H" rotates 0AAH one bit position to the left. The initial value of bit 7 sets the carry flag and the initial value of the C flag replaces bit zero of register 00H, leaving the value 55H (01010101B). The MSB of register 00H resets the carry flag to "1" and sets the overflow flag.

RR — Rotate Right

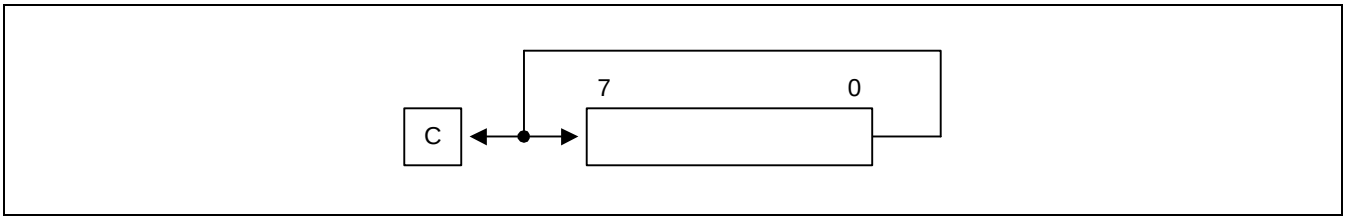
RR dst

Operation: C ← dst (0)

dst (7) ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

The contents of the destination operand are rotated right one bit position. The initial value of bit zero (LSB) is moved to bit 7 (MSB) and also replaces the carry flag (C).



Flags:
- **C:** Set if the bit rotated from the least significant bit position (bit zero) was "1".
- **Z:** Set if the result is "0"; cleared otherwise.
- **S:** Set if the result bit 7 is set; cleared otherwise.
- **V:** Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
- **D:** Unaffected.
- **H:** Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc dst	2	4	E0	R
		4	E1	IR

Examples: Given: Register 00H = 31H, register 01H = 02H, and register 02H = 17H:

RR 00H → Register 00H = 98H, C = "1"

RR @01H → Register 01H = 02H, register 02H = 8BH, C = "1"

In the first example, if general register 00H contains the value 31H (00110001B), the statement "RR 00H" rotates this value one bit position to the right. The initial value of bit zero is moved to bit 7, leaving the new value 98H (10011000B) in the destination register. The initial bit zero also resets the C flag to "1" and the sign flag and overflow flag are also set to "1".

RRC — Rotate Right Through Carry

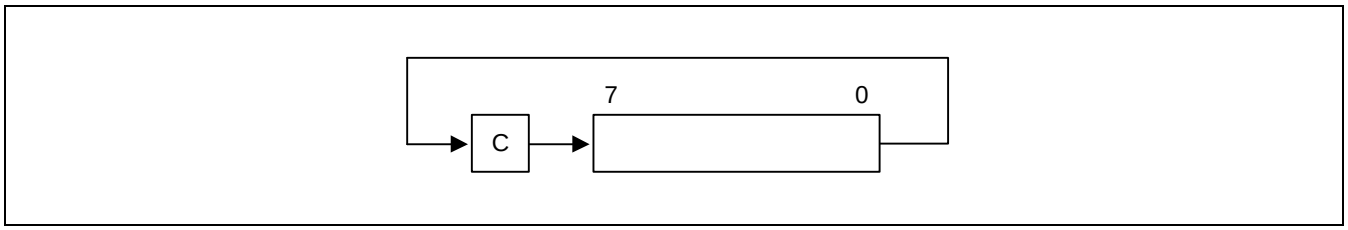
RRC dst

Operation: dst (7) ← C

C ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

The contents of the destination operand and the carry flag are rotated right one bit position. The initial value of bit zero (LSB) replaces the carry flag; the initial value of the carry flag replaces bit 7 (MSB).



Flags:

C: Set if the bit rotated from the least significant bit position (bit zero) was "1".
Z: Set if the result is "0" cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc dst	2	4	C0	R
		4	C1	IR

Examples: Given: Register 00H = 55H, register 01H = 02H, register 02H = 17H, and C = "0":

RRC 00H → Register 00H = 2AH, C = "1"
RRC @01H → Register 01H = 02H, register 02H = 0BH, C = "1"

In the first example, if general register 00H contains the value 55H (01010101B), the statement "RRC 00H" rotates this value one bit position to the right. The initial value of bit zero ("1") replaces the carry flag and the initial value of the C flag ("1") replaces bit 7. This leaves the new value 2AH (00101010B) in destination register 00H. The sign flag and overflow flag are both cleared to "0".

SB0 — Select Bank 0

SB0

Operation: BANK ← 0

The SB0 instruction clears the bank address flag in the FLAGS register (FLAGS.0) to logic zero, selecting bank 0 register addressing in the set 1 area of the register file.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	4F

Example: The statement

SB0

clears FLAGS.0 to "0", selecting bank 0 register addressing.

SB1 — Select Bank 1

SB1

Operation: BANK ← 1

The SB1 instruction sets the bank address flag in the FLAGS register (FLAGS.0) to logic one, selecting bank 1 register addressing in the set 1 area of the register file. (Bank 1 is not implemented in some S3C8-series microcontrollers.)

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	5F

Example: The statement

SB1

sets FLAGS.0 to "1", selecting bank 1 register addressing, if implemented.

SBC — Subtract with Carry

SBC dst,src

Operation: dst ← dst – src – c

The source operand, along with the current value of the carry flag, is subtracted from the destination operand and the result is stored in the destination. The contents of the source are unaffected. Subtraction is performed by adding the two's-complement of the source operand to the destination operand. In multiple precision arithmetic, this instruction permits the carry ("borrow") from the subtraction of the low-order operands to be subtracted from the subtraction of high-order operands.

Flags:

C: Set if a borrow occurred (src > dst); cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite sign and the sign of the result is the same as the sign of the source; cleared otherwise.
D: Always set to "1".
H: Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise, indicating a "borrow".

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	32	r	r
		6	33	r	lr
opc | src | dst	3	6	34	R	R
		6	35	R	IR
opc | dst | src	3	6	36	R	IM

Examples: Given: R1 = 10H, R2 = 03H, C = "1", register 01H = 20H, register 02H = 03H, and register 03H = 0AH:

SBC R1,R2 → R1 = 0CH, R2 = 03H
SBC R1,@R2 → R1 = 05H, R2 = 03H, register 03H = 0AH
SBC 01H,02H → Register 01H = 1CH, register 02H = 03H
SBC 01H,@02H → Register 01H = 15H,register 02H = 03H, register 03H = 0AH
SBC 01H,#8AH → Register 01H = 95H; C, S, and V = "1"

In the first example, if working register R1 contains the value 10H and register R2 the value 03H, the statement "SBC R1,R2" subtracts the source value (03H) and the C flag value ("1") from the destination (10H) and then stores the result (0CH) in register R1.

SCF — Set Carry Flag

SCF

Operation: C ← 1

The carry flag (C) is set to logic one, regardless of its previous value.

Flags: **C:** Set to "1".

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	DF

Example: The statement

SCF

sets the carry flag to logic one.

SRA — Shift Right Arithmetic

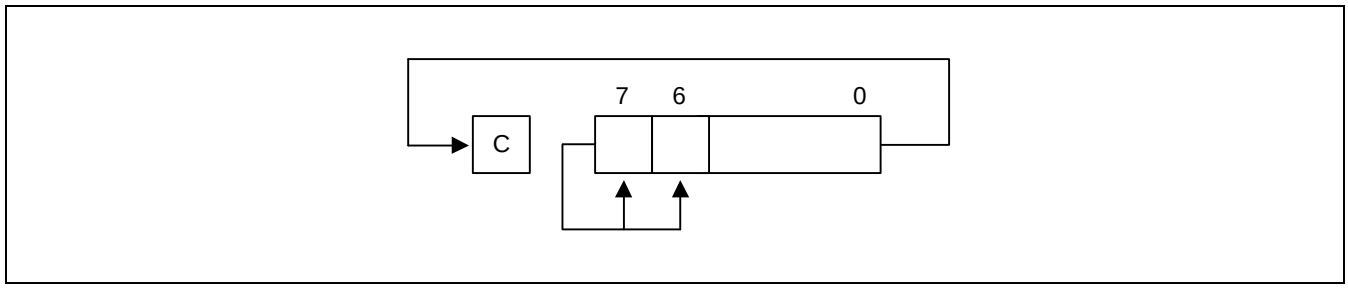
SRA dst

Operation: dst (7) ← dst (7)

C ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

An arithmetic shift-right of one bit position is performed on the destination operand. Bit zero (the LSB) replaces the carry flag. The value of bit 7 (the sign bit) is unchanged and is shifted into bit position 6.



Flags:
C: Set if the bit shifted from the LSB position (bit zero) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Always cleared to "0".
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	D0	R
		4	D1	IR

Examples: Given: Register 00H = 9AH, register 02H = 03H, register 03H = 0BCH, and C = "1":

SRA 00H → Register 00H = 0CD, C = "0"

SRA @02H → Register 02H = 03H, register 03H = 0DEH, C = "0"

In the first example, if general register 00H contains the value 9AH (10011010B), the statement "SRA 00H" shifts the bit values in register 00H right one bit position. Bit zero ("0") clears the C flag and bit 7 ("1") is then shifted into the bit 6 position (bit 7 remains unchanged). This leaves the value 0CDH (11001101B) in destination register 00H.

SRP/SRP0/SRP1 — Set Register Pointer

SRP src

SRP0 src

SRP1 src

Operation:

If src (1) = 1 and src (0) = 0 then: RP0 (3–7) ← src (3–7)

If src (1) = 0 and src (0) = 1 then: RP1 (3–7) ← src (3–7)

If src (1) = 0 and src (0) = 0 then: RP0 (4–7) ← src (4–7),

RP0 (3) ← 0

RP1 (4–7) ← src (4–7),

RP1 (3) ← 1

The source data bits one and zero (LSB) determine whether to write one or both of the register pointers, RP0 and RP1. Bits 3–7 of the selected register pointer are written unless both register pointers are selected. RP0.3 is then cleared to logic zero and RP1.3 is set to logic one.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode src
opc	src	2	4	31	IM

Examples: The statement

SRP #40H

sets register pointer 0 (RP0) at location 0D6H to 40H and register pointer 1 (RP1) at location 0D7H to 48H.

The statement "SRP0 #50H" sets RP0 to 50H, and the statement "SRP1 #68H" sets RP1 to 68H.

STOP — Stop Operation

STOP

Operation:

The STOP instruction stops the both the CPU clock and system clock and causes the microcontroller to enter Stop mode. During Stop mode, the contents of on-chip CPU registers, peripheral registers, and I/O port control and data registers are retained. Stop mode can be released by an external reset operation or by external interrupts. For the reset operation, the RESET pin must be held to Low level until the required oscillation stabilization interval has elapsed.
In application programs, a STOP instruction must be immediately followed by at least three NOP instructions. This ensures an adeguate time interval for the clock to stabilize before the next instruction is executed. If three or more NOP instructons are not used after STOP instruction, leakage current could be flown because of the floating state in the internal bus.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	1	4	7F	–	–

Example: The statement

```
STOP                ; halts all microcontroller operations
NOP
NOP
NOP
```

SUB — Subtract

SUB dst,src

Operation: dst ← dst – src

The source operand is subtracted from the destination operand and the result is stored in the destination. The contents of the source are unaffected. Subtraction is performed by adding the two's complement of the source operand to the destination operand.

Flags:

C: Set if a "borrow" occurred; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite signs and the sign of the result is of the same as the sign of the source operand; cleared otherwise.
D: Always set to "1".
H: Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise indicating a "borrow".

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	22	r	r
				6	23	r	lr
opc	src	dst	3	6	24	R	R
				6	25	R	IR
opc	dst	src	3	6	26	R	IM

Examples: Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

SUB R1,R2 → R1 = 0FH, R2 = 03H
SUB R1,@R2 → R1 = 08H, R2 = 03H
SUB 01H,02H → Register 01H = 1EH, register 02H = 03H
SUB 01H,@02H → Register 01H = 17H, register 02H = 03H
SUB 01H,#90H → Register 01H = 91H; C, S, and V = "1"
SUB 01H,#65H → Register 01H = 0BCH; C and S = "1", V = "0"

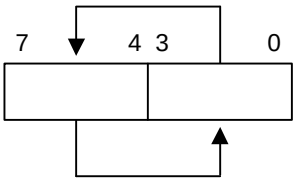
In the first example, if working register R1 contains the value 12H and if register R2 contains the value 03H, the statement "SUB R1,R2" subtracts the source value (03H) from the destination value (12H) and stores the result (0FH) in destination register R1.

SWAP — Swap Nibbles

SWAP dst

Operation: dst (0 – 3) ↔ dst (4 – 7)

The contents of the lower four bits and upper four bits of the destination operand are swapped.



Flags:
C: Undefined.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Undefined.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	F0	R
		4	F1	IR

Examples: Given: Register 00H = 3EH, register 02H = 03H, and register 03H = 0A4H:

SWAP 00H → Register 00H = 0E3H
SWAP @02H → Register 02H = 03H, register 03H = 4AH

In the first example, if general register 00H contains the value 3EH (00111110B), the statement "SWAP 00H" swaps the lower and upper four bits (nibbles) in the 00H register, leaving the value 0E3H (11100011B).

TCM — Test Complement Under Mask

TCM dst,src

Operation: (NOT dst) AND src

This instruction tests selected bits in the destination operand for a logic one value. The bits to be tested are specified by setting a "1" bit in the corresponding position of the source operand (mask). The TCM statement complements the destination operand, which is then ANDed with the source mask. The zero (Z) flag can then be checked to determine the result. The destination and source operands are unaffected.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always cleared to "0".
D: Unaffected.
H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	62	r	r
				6	63	r	lr
opc	src	dst	3	6	64	R	R
				6	65	R	IR
opc	dst	src	3	6	66	R	IM

Examples: Given: R0 = 0C7H, R1 = 02H, R2 = 12H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

TCM	R0,R1	→	R0 = 0C7H, R1 = 02H, Z = "1"
TCM	R0,@R1	→	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = "0"
TCM	00H,01H	→	Register 00H = 2BH, register 01H = 02H, Z = "1"
TCM	00H,@01H	→	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = "1"
TCM	00H,#34	→	Register 00H = 2BH, Z = "0"

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TCM R0,R1" tests bit one in the destination register for a "1" value. Because the mask value corresponds to the test bit, the Z flag is set to logic one and can be tested to determine the result of the TCM operation.

TM — Test Under Mask

TM dst,src

Operation: dst AND src

This instruction tests selected bits in the destination operand for a logic zero value. The bits to be tested are specified by setting a "1" bit in the corresponding position of the source operand (mask), which is ANDed with the destination operand. The zero (Z) flag can then be checked to determine the result. The destination and source operands are unaffected.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to "0".
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	72	r	r
		6	73	r	lr
opc | src | dst	3	6	74	R	R
		6	75	R	IR
opc | dst | src	3	6	76	R	IM

Examples: Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

TM	R0,R1	→	R0 = 0C7H, R1 = 02H, Z = "0"
TM	R0,@R1	→	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = "0"
TM	00H,01H	→	Register 00H = 2BH, register 01H = 02H, Z = "0"
TM	00H,@01H	→	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = "0"
TM	00H,#54H	→	Register 00H = 2BH, Z = "1"

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TM R0,R1" tests bit one in the destination register for a "0" value. Because the mask value does not match the test bit, the Z flag is cleared to logic zero and can be tested to determine the result of the TM operation.

WFI — Wait for Interrupt

WFI

Operation:

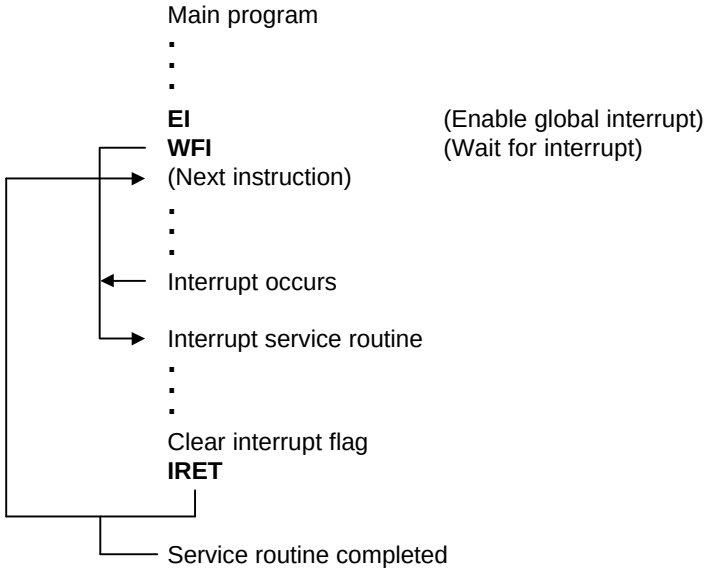
The CPU is effectively halted until an interrupt occurs, except that DMA transfers can still take place during this wait state. The WFI status can be released by an internal interrupt, including a fast interrupt .

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4n	3F
		(n = 1, 2, 3, …)	

Example: The following sample program structure shows the sequence of operations that follow a "WFI" statement:



XOR — Logical Exclusive OR

XOR dst,src

Operation: dst ← dst XOR src

The source operand is logically exclusive-ORed with the destination operand and the result is stored in the destination. The exclusive-OR operation results in a "1" bit being stored whenever the corresponding bits in the operands are different; otherwise, a "0" bit is stored.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to "0".
D: Unaffected.
H: Unaffected.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	dst | src		2	4	B2	r	r
				6	B3	r	lr
opc	src	dst	3	6	B4	R	R
				6	B5	R	IR
opc	dst	src	3	6	B6	R	IM

Examples: Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

XOR R0,R1 → R0 = 0C5H, R1 = 02H
XOR R0,@R1 → R0 = 0E4H, R1 = 02H, register 02H = 23H
XOR 00H,01H → Register 00H = 29H, register 01H = 02H
XOR 00H,@01H → Register 00H = 08H, register 01H = 02H, register 02H = 23H
XOR 00H,#54H → Register 00H = 7FH

In the first example, if working register R0 contains the value 0C7H and if register R1 contains the value 02H, the statement "XOR R0,R1" logically exclusive-ORs the R1 value with the R0 value and stores the result (0C5H) in the destination register R0.

NOTES

7 CLOCK CIRCUIT

OVERVIEW

The S3C825A microcontroller has two oscillator circuits: a main clock and a sub clock circuit. The CPU and peripheral hardware operate on the system clock frequency supplied through these circuits. The maximum CPU clock frequency of S3C825A is determined by CLKCON register settings.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

— External crystal, ceramic resonator, RC oscillation source, or an external clock source
— Oscillator stop and wake-up functions
— Programmable frequency divider for the CPU clock (fxx divided by 1, 2, 8, or 16)
— System clock control register, CLKCON
— STOP control register, STPCON

CPU Clock Notation

In this document, the following notation is used for descriptions of the CPU clock;

fx: main clock
fxt: sub clock
fxx: selected system clock

MAIN OSCILLATOR CIRCUITS

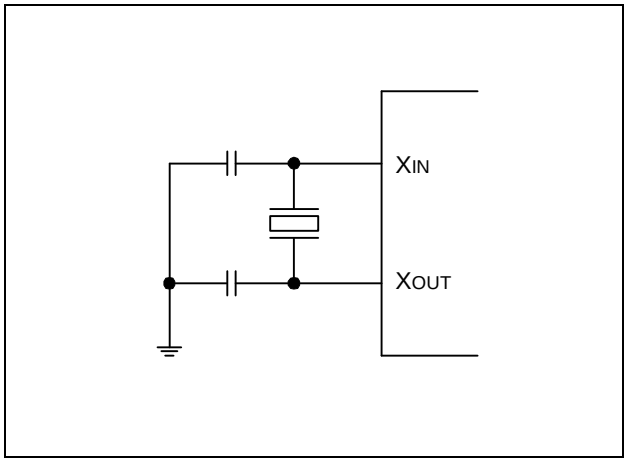


Figure 7-1. Crystal/Ceramic Oscillator (fx)

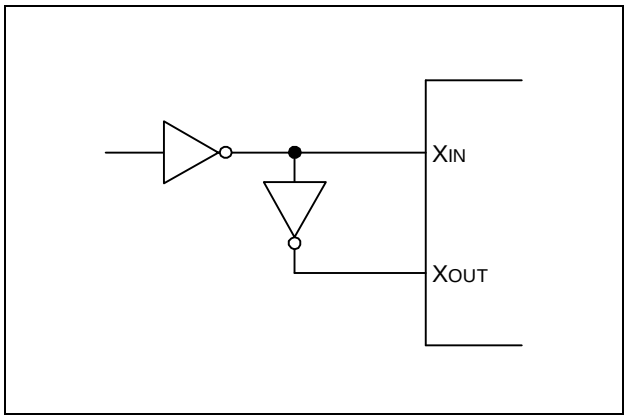


Figure 7-2. External Oscillator (fx)

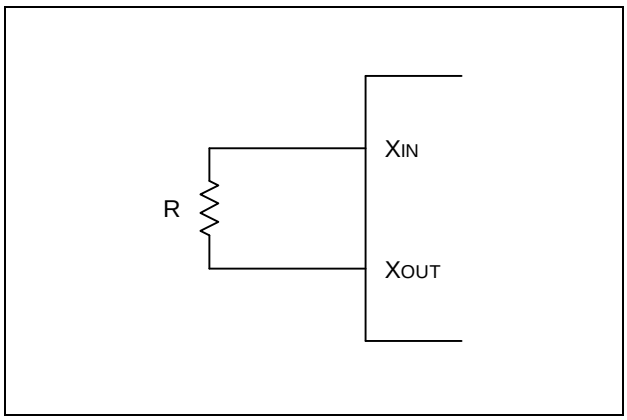


Figure 7-3. RC Oscillator (fx)

SUB OSCILLATOR CIRCUITS

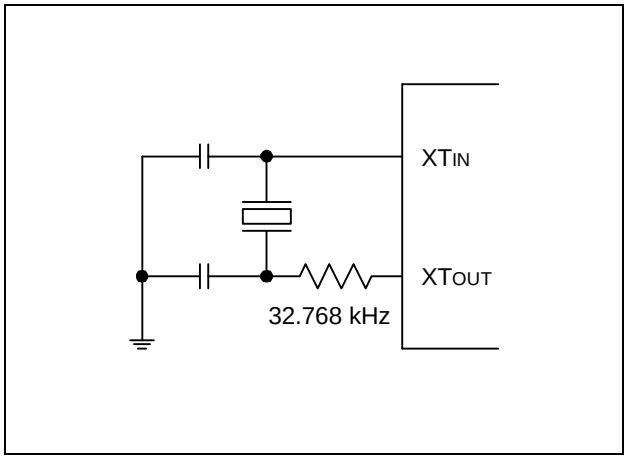


Figure 7-4. Crystal/Ceramic Oscillator (fxt)

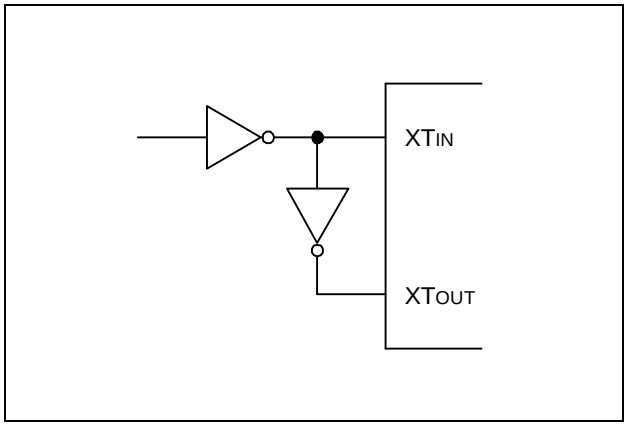


Figure 7-5. External Oscillator (fxt)

CLOCK STATUS DURING POWER-DOWN MODES

The two power-down modes, Stop mode and Idle mode, affect the system clock as follows:

— In Stop mode, the main oscillator is halted. Stop mode is released, and the oscillator is started, by a reset operation or an external interrupt (with RC delay noise filter).

— In Idle mode, the internal clock signal is gated to the CPU, but not to interrupt structure, timers, timer/counters, and watch timer. Idle mode is released by a reset or by an external or internal interrupt.

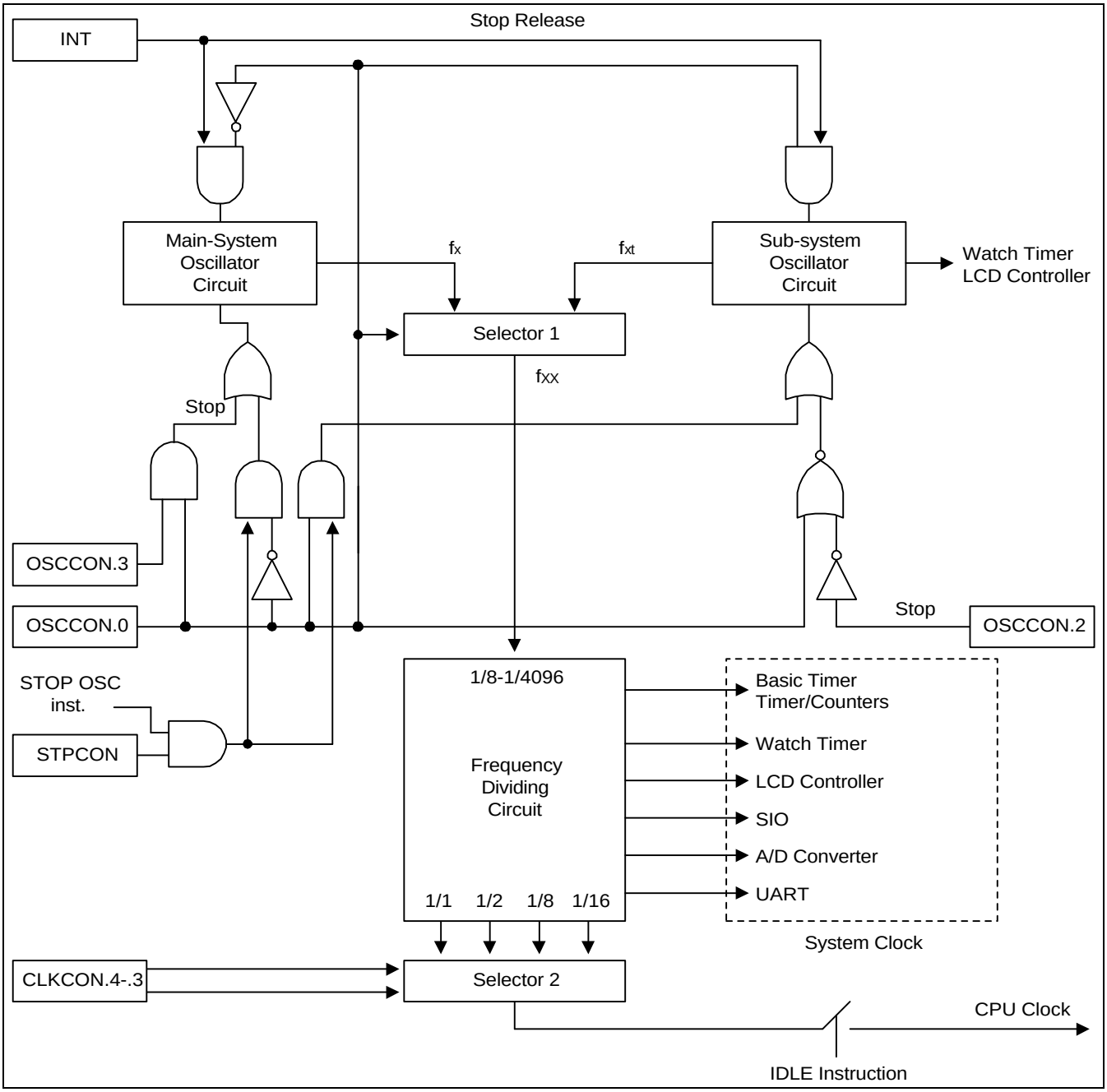


Figure 7-6. System Clock Circuit Diagram

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

The system clock control register, CLKCON, is located in the set 1, address D4H. It is read/write addressable and has the following functions:

— Oscillator frequency divide-by value

After the main oscillator is activated, and the fxx/16 (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed fxx/8, fxx/2, or fxx/1.

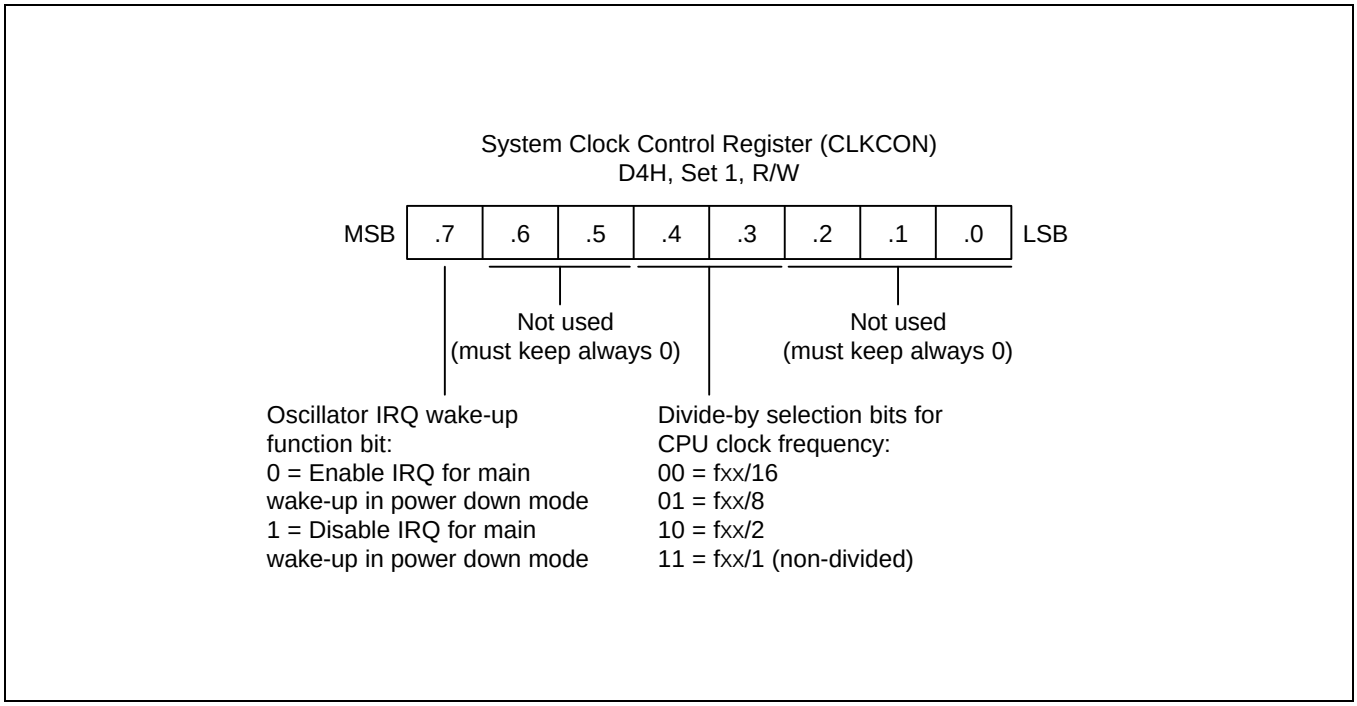


Figure 7-7. System Clock Control Register (CLKCON)

OSCILLATOR CONTROL REGISTER (OSCCON)

The oscillator control register, OSCCON, is located in set 1, bank 0, at address D2H. It is read/write addressable and has the following functions:

— System clock selection
— Main oscillator control
— Sub oscillator control

OSCCON.0 register settings select Main clock or Sub clock as system clock.
After a reset, Main clock is selected for system clock because the reset value of OSCCON.0 is "0".

The main oscillator can be stopped or run by setting OSCCON.3.

The sub oscillator can be stopped or run by setting OSCCON.2.

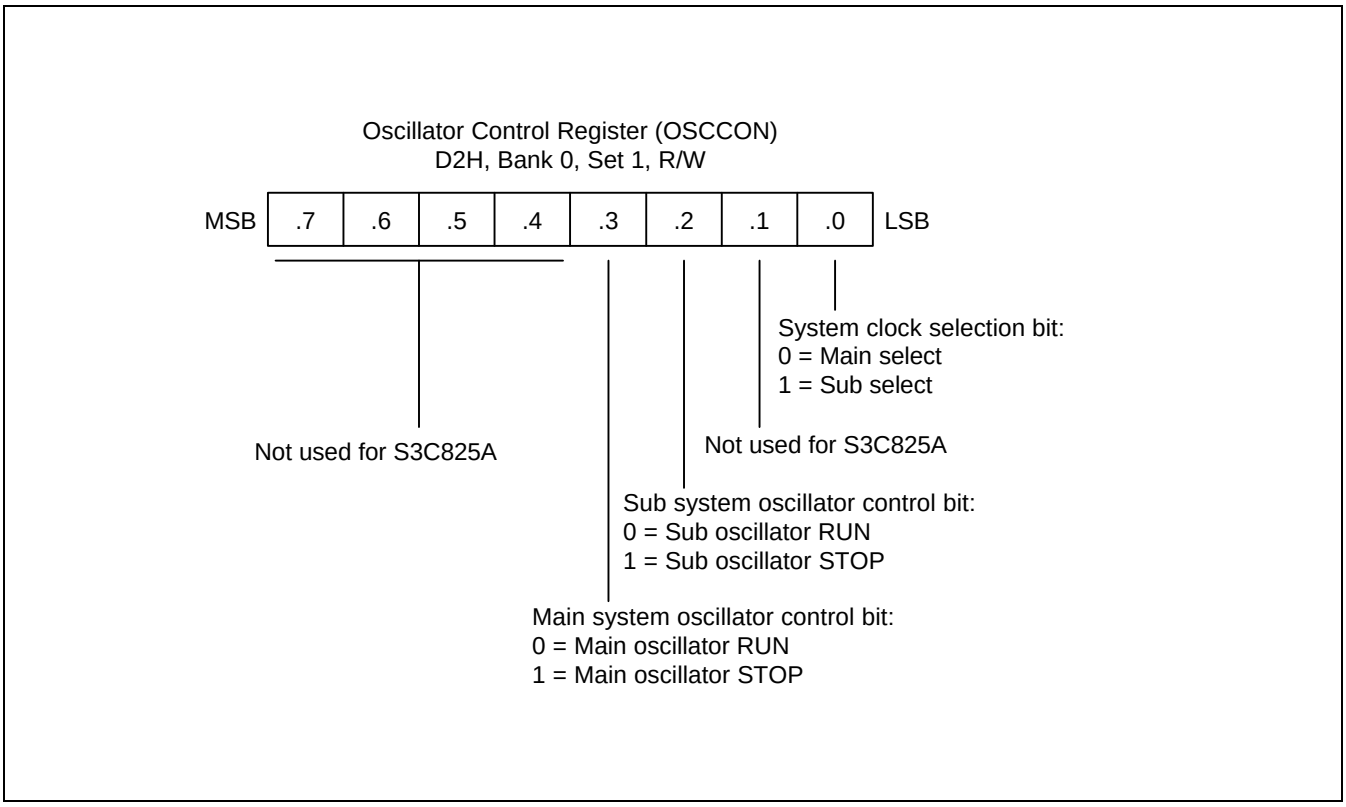


Figure 7-8. Oscillator Control Register (OSCCON)

SWITCHING THE CPU CLOCK

Data loading in the oscillator control register, OSCCON, determine whether a main or a sub clock is selected as the CPU clock, and also how this frequency is to be divided by setting CLKCON. This makes it possible to switch dynamically between main and sub clocks and to modify operating frequencies.

OSCCON.0 select the main clock (fx) or the sub clock (fxt) for the CPU clock. OSCCON .3 start or stop main clock oscillation, and OSCCON.2 start or stop sub clock oscillation. CLKCON.4–.3 control the frequency divider circuit, and divide the selected fxx clock by 1, 2, 8, 16.

For example, you are using the default CPU clock (normal operating mode and a main clock of fx/16) and you want to switch from the fx clock to a sub clock and to stop the main clock. To do this, you need to set CLKCON.4-.3 to "11", OSCCON.0 to “1”, and OSCCON.3 to “1” simultaneously. This switches the clock from fx to fxt and stops main clock oscillation.

The following steps must be taken to switch from a sub clock to the main clock: first, set OSCCON.3 to “0” to enable main clock oscillation. Then, after a certain number of machine cycles has elapsed, select the main clock by setting OSCCON.0 to “0”.

☞ PROGRAMMING TIP — Switching the CPU clock

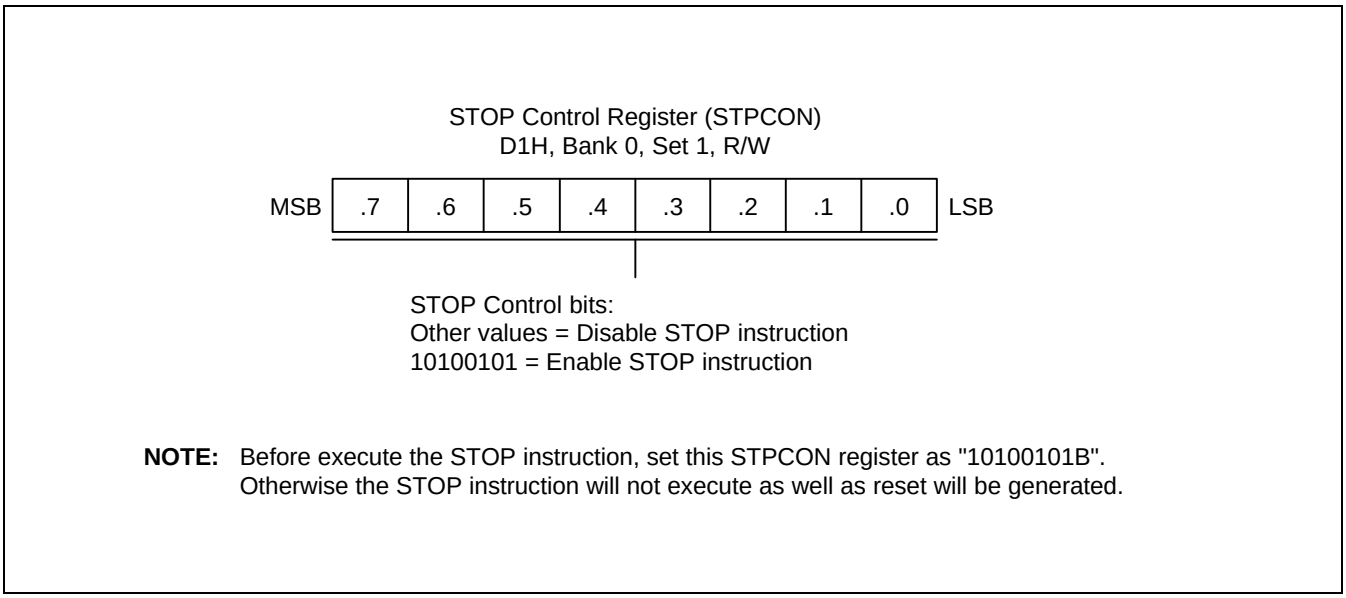
1. This example shows how to change from the main clock to the sub clock:

```
MA2SUB  LD      OSCCON,#01H     ;  Switches to the sub clock
                                ;  Stop the main clock oscillation

        RET
```

2. This example shows how to change from sub clock to main clock:

```
SUB2MA  AND     OSCCON,#07H     ;  Start the main clock oscillation
        CALL    DLY16           ;  Delay 16 ms
        AND     OSCCON,#06H     ;  Switch to the main clock
        RET
DLY16   SRP     #0C0H
        LD      R0,#20H
DEL     NOP
        DJNZ    R0,DEL
        RET
```



NOTE: Before execute the STOP instruction, set this STPCON register as "10100101B". Otherwise the STOP instruction will not execute as well as reset will be generated.

Figure 7-9. STOP Control Register (STPCON)

NOTES

8 RESET and POWER-DOWN

SYSTEM RESET

OVERVIEW

During a power-on reset, the voltage at V_{DD} goes to High level and the RESET pin is forced to Low level. The RESET signal is input through a schmitt trigger circuit where it is then synchronized with the CPU clock. This procedure brings the S3C825A into a known operating status.

To allow time for internal CPU clock oscillation to stabilize, the RESET pin must be held to Low level for a minimum time interval after the power supply comes within tolerance. The minimum required time of a reset operation for oscillation stabilization is 1 millisecond.

Whenever a reset occurs during normal operation (that is, when both V_{DD} and RESET are High level), the RESET pin is forced Low level and the reset operation starts. All system and peripheral control registers are then reset to their default hardware values

In summary, the following sequence of events occurs during a reset operation:

— All interrupt is disabled.
— The watchdog function (basic timer) is enabled.
— Ports 0-8 are set to input mode, and all pull-up resistors are disabled for the I/O port.
— Peripheral control and data register settings are disabled and reset to their default hardware values.
— The program counter (PC) is loaded with the program reset address in the ROM, 0100H.
— When the programmed oscillation stabilization time interval has elapsed, the instruction stored in ROM location 0100H (and 0101H) is fetched and executed.

NORMAL MODE RESET OPERATION

In normal (masked ROM) mode, the Test pin is tied to V_{SS}. A reset enables access to the 48-Kbyte on-chip ROM.

NOTE

To program the duration of the oscillation stabilization interval, you make the appropriate settings to the basic timer control register, BTCON, *before* entering Stop mode. Also, if you do not want to use the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), you can disable it by writing "1010B" to the upper nibble of BTCON.

HARDWARE RESET VALUES

Table 8-1, 8-2, 8-3 list the reset values for CPU and system registers, peripheral control registers, and peripheral data registers following a reset operation. The following notation is used to represent reset values:

— A "1" or a "0" shows the reset bit value as logic one or logic zero, respectively.

— An "x" means that the bit value is undefined after a reset.

— A dash ("–") means that the bit is either not used or not mapped, but read 0 is the bit value.

Table 8-1. S3C825A Set 1 Register and Values after RESET

Register Name	Mnemonic	Address		Bit Values after RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Interrupt Pending Register (NOTE)	INTPND	208	D0H	–	–	0	0	0	0	0	0
STOP Control Register (NOTE)	STPCON	209	D1H	0	0	0	0	0	0	0	0
Oscillator Control Register (NOTE)	OSCCON	210	D2H	–	–	–	–	0	0	–	0
Basic Timer Control Register	BTCON	211	D3H	0	0	0	0	0	0	0	0
Clock Control Register	CLKCON	212	D4H	0	0	0	0	0	0	0	0
System Flags Register	FLAGS	213	D5H	x	x	x	x	x	x	0	0
Register Pointer (High Byte)	RP0	214	D6H	1	1	0	0	0	–	–	–
Register Pointer (Low Byte)	RP1	215	D7H	1	1	0	0	1	–	–	–
Stack Pointer (High Byte)	SPH	216	D8H	x	x	x	x	x	x	x	x
Stack Pointer (Low Byte)	SPL	217	D9H	x	x	x	x	x	x	x	x
Instruction Pointer (High Byte)	IPH	218	DAH	x	x	x	x	x	x	x	x
Instruction Pointer (Low Byte)	IPL	219	DBH	x	x	x	x	x	x	x	x
Interrupt Request Register	IRQ	220	DCH	0	0	0	0	0	0	0	0
Interrupt Mask Register	IMR	221	DDH	x	x	x	x	x	x	x	x
System Mode Register	SYM	222	DEH	0	–	–	x	x	x	0	0
Register Page Pointer	PP	223	DFH	0	0	0	0	0	0	0	0

NOTE: The registers, INTPND/STPCON/OSCCON, are in bank 0 of set 1.

Table 8-2. S3C825A Set 1, Bank 0 Register Values after RESET

Register Name	Mnemonic	Address		Bit Values after RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
SIO Control Register	SIOCON	224	E0H	0	0	0	0	0	0	0	0
SIO Data Register	SIODATA	225	E1H	0	0	0	0	0	0	0	0
SIO Prescaler Register	SIOPS	226	E2H	0	0	0	0	0	0	0	0
Timer 0 Counter Register	T0CNT	227	E3H	0	0	0	0	0	0	0	0
Timer 0 Data Register	T0DATA	228	E4H	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	229	E5H	0	0	0	0	0	0	0	0
Timer B Counter Register	TBCNT	230	E6H	0	0	0	0	0	0	0	0
Timer A Counter Register	TACNT	231	E7H	0	0	0	0	0	0	0	0
Timer B Data Register	TBDATA	232	E8H	1	1	1	1	1	1	1	1
Timer A Data Register	TADATA	233	E9H	1	1	1	1	1	1	1	1
Timer B Control Register	TBCON	234	EAH	–	–	0	0	0	0	0	0
Timer 1/A Control Register	TACON	235	EBH	0	0	0	0	0	0	0	0
Timer 2 Counter Register	T2CNT	236	ECH	0	0	0	0	0	0	0	0
Timer 2 Data Register	T2DATA	237	EDH	1	1	1	1	1	1	1	1
Timer 2 Control Register	T2CON	238	EEH	0	0	0	0	0	0	0	0
A/D Converter Control Register	ADCON	239	EFH	–	–	0	0	0	0	0	0
A/D Converter Data Register (high byte)	ADDATAH	240	F0H	x	x	x	x	x	x	x	x
A/D Converter Data Register (low byte)	ADDATAL	241	F1H	–	–	–	–	–	–	x	x
LCD Control Register	LCON	242	F2H	0	0	–	–	0	0	0	0
LCD Mode Register	LMOD	243	F3H	–	–	–	–	0	0	0	0
Timer 3 Counter (high byte)	T3CNTH	244	F4H	0	0	0	0	0	0	0	0
Timer 3 Counter (low byte)	T3CNTL	245	F5H	0	0	0	0	0	0	0	0
Timer 3 Data Register (high byte)	T3DATAH	246	F6H	1	1	1	1	1	1	1	1
Timer 3 Data Register (low byte)	T3DATAL	247	F7H	1	1	1	1	1	1	1	1
Timer 3 Control Register	T3CON	248	F8H	0	0	0	0	0	0	0	0
UART data register	UDATA	249	F9H	x	x	x	x	x	x	x	x
UART control register	UARTCON	250	FAH	0	0	0	0	0	0	0	0
UART Baud Rate data register	BRDATA	251	FBH	1	1	1	1	1	1	1	1
Location FCH is not mapped.											
Basic Timer Counter	BTCNT	253	FDH	0	0	0	0	0	0	0	0
Location FEH is not mapped.											
Interrupt Priority Register	IPR	255	FFH	x	x	x	x	x	x	x	x

Table 8-3. S3C825A Set 1, Bank 1 Register Values after RESET

Register Name	Mnemonic	Address		Bit Values after RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 2 Control Register(High Byte)	P2CONH	224	E0H	0	0	0	0	0	0	0	0
Port 2 Control Register(Low Byte)	P2CONL	225	E1H	0	0	0	0	0	0	0	0
Port 2 Pull-up Resistors enable Register	P2PUR	226	E2H	0	0	0	0	0	0	0	0
Port 2 Interrupt Control Register	P2INT	227	E3H	0	0	0	0	0	0	0	0
Port 3 Control Register(High Byte)	P3CONH	228	E4H	0	0	0	0	0	0	0	0
Port 3 Control Register(Low Byte)	P3CONL	229	E5H	0	0	0	0	0	0	0	0
Port 3 Pull-up Resistors enable Register	P3PUR	230	E6H	0	0	0	0	0	0	0	0
Port 4 Interrupt Edge Selection Register	P4EDGE	231	E7H	0	0	0	0	0	0	0	0
Port 4 Control Register(High Byte)	P4CONH	232	E8H	0	0	0	0	0	0	0	0
Port 4 Control Register(Low Byte)	P4CONL	233	E9H	0	0	0	0	0	0	0	0
Port 4 Interrupt Control Register	P4INT	234	EAH	0	0	0	0	0	0	0	0
Port 4 Interrupt Pending Register	P4PND	235	EBH	0	0	0	0	0	0	0	0
Port 5 Control Register(High Byte)	P5CONH	236	ECH	0	0	0	0	0	0	0	0
Port 5 Control Register(Low Byte)	P5CONL	237	EDH	0	0	0	0	0	0	0	0
Port 5 Pull-up Resistors enable Register	P5PUR	238	EEH	0	0	0	0	0	0	0	0
Watch timer control register	WTCON	239	EFH	0	0	0	0	0	0	0	0
Port 0 Data Register	P0	240	F0H	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	241	F1H	0	0	0	0	0	0	0	0
Port 2 Data Register	P2	242	F2H	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	243	F3H	0	0	0	0	0	0	0	0
Port 4 Data Register	P4	244	F4H	0	0	0	0	0	0	0	0
Port 5 Data Register	P5	245	F5H	0	0	0	0	0	0	0	0
Port 6 Data Register	P6	246	F6H	0	0	0	0	0	0	0	0
Port 7 Data Register	P7	247	F7H	0	0	0	0	0	0	0	0
Port 8 Data Register	P8	248	F8H	0	0	0	0	0	0	0	0
Port Group 0 Control Register	PG0CON	251	F9H	0	0	0	0	0	0	0	0
Port Group 1 Control Register	PG1CON	252	FAH	0	0	0	0	0	0	0	0
Locations FBH-FFH are not mapped.											

POWER-DOWN MODES

STOP MODE

Stop mode is invoked by the instruction STOP (opcode 7FH). In Stop mode, the operation of the CPU and all peripherals is halted. That is, the on-chip oscillator for system clock stops and the supply current is reduced to less than 3 μA. All system functions stop when the clock "freezes", but data stored in the internal register file is retained. Stop mode can be released in one of two ways: by a reset or by external interrupts, for more details see Figure 7-3.

NOTE

Do not use stop mode if you are using an external clock source because X_{IN} or XT_{IN} input must be restricted internally to V_{SS} to reduce current leakage.

Using RESET to Release Stop Mode

Stop mode is released when the RESET signal is released and returns to high level: all system and peripheral control registers are reset to their default hardware values and the contents of all data registers are retained. A reset operation automatically selects a slow clock fxx/16 because CLKCON.3 and CLKCON.4 are cleared to '00B'. After the programmed oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in ROM location 0100H.

Using an External Interrupt to Release Stop Mode

External interrupts with an RC-delay noise filter circuit can be used to release Stop mode. Which interrupt you can use to release Stop mode in a given situation depends on the microcontroller's current internal operating mode. The external interrupts in the S3C825A interrupt structure that can be used to release Stop mode are:

— External interrupts P2.4–P2.7 (INT0–INT3) and P4.0–P4.7 (INT4–INT11)

Please note the following conditions for Stop mode release:

— If you release Stop mode using an external interrupt, the current values in system and peripheral control registers are unchanged except STPCON register.

— If you use an internal or external interrupt for stop mode release, you can also program the duration of the oscillation stabilization interval. To do this, you must make the appropriate control and clock settings *before* entering stop mode.

— When the Stop mode is released by external interrupt, the CLKCON.4 and CLKCON.3 bit-pair setting remains unchanged and the currently selected clock value is used.

— The external interrupt is serviced when the Stop mode release occurs. Following the IRET from the service routine, the instruction immediately following the one that initiated Stop mode is executed.

How to Enter into Stop Mode

Handling STPCON register then writing Stop instruction (keep the order).

```
LD      STPCON, #10100101B
STOP
NOP
NOP
NOP
```

IDLE MODE

Idle mode is invoked by the instruction IDLE (opcode 6FH). In idle mode, CPU operations are halted while some peripherals remain active. During idle mode, the internal clock signal is gated away from the CPU, but all peripherals remain active. Port pins retain the mode (input or output) they had at the time idle mode was entered.

There are two ways to release idle mode:

1. Execute a reset. All system and peripheral control registers are reset to their default values and the contents of all data registers are retained. The reset automatically selects the slow clock fxx/16 because CLKCON.4 and CLKCON.3 are cleared to '00B'. If interrupts are masked, a reset is the only way to release idle mode.
2. Activate any enabled interrupt, causing idle mode to be released. When you use an interrupt to release idle mode, the CLKCON.4 and CLKCON.3 register values remain unchanged, and the currently selected clock value is used. The interrupt is then serviced. When the return-from-interrupt (IRET) occurs, the instruction immediately following the one that initiated idle mode is executed.

9 I/O PORTS

OVERVIEW

The S3C825A microcontroller has four bit-programmable and five nibble-programmable I/O ports, P0–P8. The port 0-4, 6, and 7 are 8-bit ports, the port 5 is 7-bit port, and the port 8 is 4-bit port. This gives a total of 67 I/O pins. Each port can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required. All ports of the S3C825A can be configured to input or output mode and P0, P6-P8 are shared with LCD signals.

Table 9-1 gives you a general overview of the S3C825A I/O port functions.

Table 9-1. S3C825A Port Configuration Overview

Port	Configuration Options
0	4-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-ups. P0.0–P0.7 can alternately be used as outputs for LCD segment signals.
1	4-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-ups. P1.0–P1.7 can alternately be used as outputs for LCD segment signals.
2	1-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output mode selected by software; software assignable pull-ups. P2.4–P2.7 can be used as input for external interrupts INT0–INT3 (with noise filter and interrupt control, and can alternately be used as T0CLK, T1CLK, TAOUT, and TBOUT.
3	1-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output mode selected by software; software assignable pull-ups. Alternately P3.0–P3.7 can be used as AD0-AD3, T3CLK, T3OUT/T3PWM/T3CAP, T0OUT/T0PWM/T0CAP.
4	1-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output mode selected by software; software assignable pull-ups. P4.0–P4.7 can alternately be used as inputs for external interrupts INT4–INT11 (with noise filter and interrupt control).
5	1-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output mode selected by software; software assignable pull-ups. Alternately P5.0–P5.5 can be used as SCK, SI, SO, BUZ, RXD, and TXD.
6	4-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-ups. P6.0–P6.7 can alternately be used as outputs for LCD signals.
7	4-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-ups. P7.0–P7.7 can alternately be used as outputs for LCD segment signals.
8	4-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-ups. P8.0–P8.3 can alternately be used as outputs for LCD segment signals.

PORT DATA REGISTERS

Table 9-2 gives you an overview of the register locations of all nine S3C825A I/O port data registers. Data registers for ports 0, 1, 2, 3, 4, 5, 6, 7 and 8 have the general format shown in Figure 9-1.

Table 9-2. Port Data Register Summary

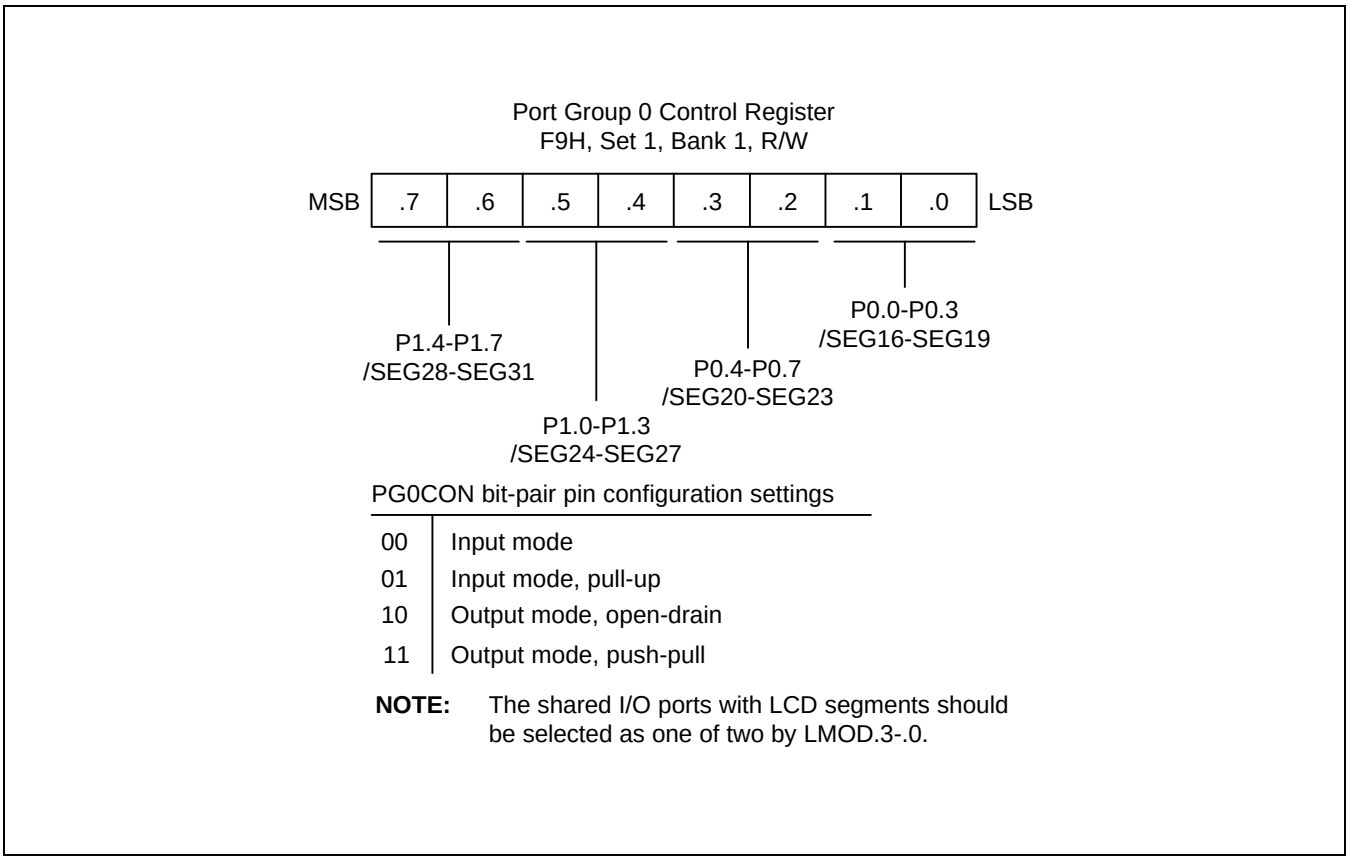
Register Name	Mnemonic	Decimal	Hex	Location	R/W
Port 0 data register	P0	240	F0H	Set 1, Bank 1	R/W
Port 1 data register	P1	241	F1H	Set 1, Bank 1	R/W
Port 2 data register	P2	242	F2H	Set 1, Bank 1	R/W
Port 3 data register	P3	243	F3H	Set 1, Bank 1	R/W
Port 4 data register	P4	244	F4H	Set 1, Bank 1	R/W
Port 5 data register	P5	245	F5H	Set 1, Bank 1	R/W
Port 6 data register	P6	246	F6H	Set 1, Bank 1	R/W
Port 7 data register	P7	247	F7H	Set 1, Bank 1	R/W
Port 8 data register	P8	248	F8H	Set 1, Bank 1	R/W

PORT 0, 1

Port 0 and 1 are 8-bit I/O ports with nibble configurable pins, respectively. Port 0 and 1 pins are accessed directly by writing or reading the Port 0 and 1 data registers, P0 at location F0H and P1 at location F1H in set 1, bank 1. P0.0–P0.7 and P1.0–P1.7 can serve as inputs (with or without pull-ups), as output (open drain or push-pull). And they can serve as segment pins for LCD, also.

Port Group 0 Control Register (PG0CON)

Port 0 and 1 have a 8-bit control register: PG0CON.0–.3 for P0.0–P0.7 and PG0CON.4–.7 for P1.0–P1.7. A reset clears the PG0CON register to “00H”, configuring all pins to input mode.



PG0CON bit-pair pin configuration settings

00	Input mode
01	Input mode, pull-up
10	Output mode, open-drain
11	Output mode, push-pull

NOTE: The shared I/O ports with LCD segments should be selected as one of two by LMOD.3-.0.

Figure 9-1. Port Group 0 Control Register (PG0CON)

PORT 2

Port 2 is an 8-bit I/O port with individually configurable pins. Port 2 pins are accessed directly by writing or reading the port 2 data register, P2 at location F2H in set 1, bank 1. P2.0–P2.7 can serve as inputs, as outputs (push-pull or open-drain) or it can be configured the following functions.

— Low-nibble pins (P2.0–P2.3): T2CLK, T2OUT

— High-nibble pins (P2.4–P2.7): T0CLK, T1CLK, TAOUT, TBOUT, INT0–INT3

Port 2 Control Registers (P2CONH, P2CONL)

Port 2 has two 8-bit control registers: P2CONH for P2.4–P2.7 and P2CONL for P2.0–P2.3. A reset clears the P2CONH and P2CONL registers to "00H", configuring P2.4–P2.7 pins to input mode with interrupt on falling edge and P2.0–P2.3 pins to input mode. You use control registers setting to select input or output mode(push-pull or open-drain) and enable the alternative functions.

When programming the port, please remember that any alternative peripheral I/O function you configure using the port 2 control registers must also be enabled in the associated peripheral module.

Port 2 Pull-Up Resistor Control Register (P2PUR)

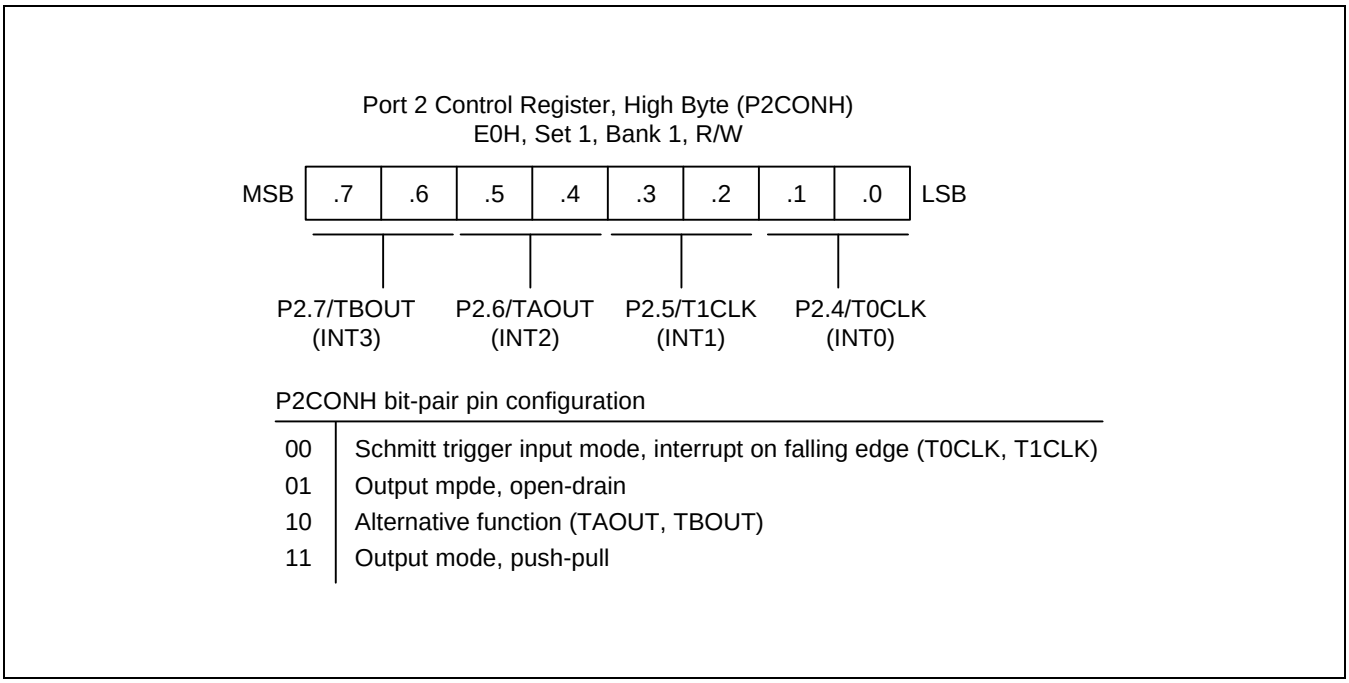
Using the port2 pull-up resistor control register, P2PUR (E2H, set 1, bank 1), you can configure pull-up resistors to individual port 2 pins.

Port 2 Interrupt Control Registers (P2INT)

To process external interrupts at the port 2 pins, a additional control register is provided: the port 2 interrupt control register P2INT (E3H, set 1, bank 1).

The port 2 interrupt control register P2INT lets you check for interrupt pending conditions and clear the pending condition when the interrupt service routine has been initiated. The application program detects interrupt requests by polling the P2INT register at regular intervals.

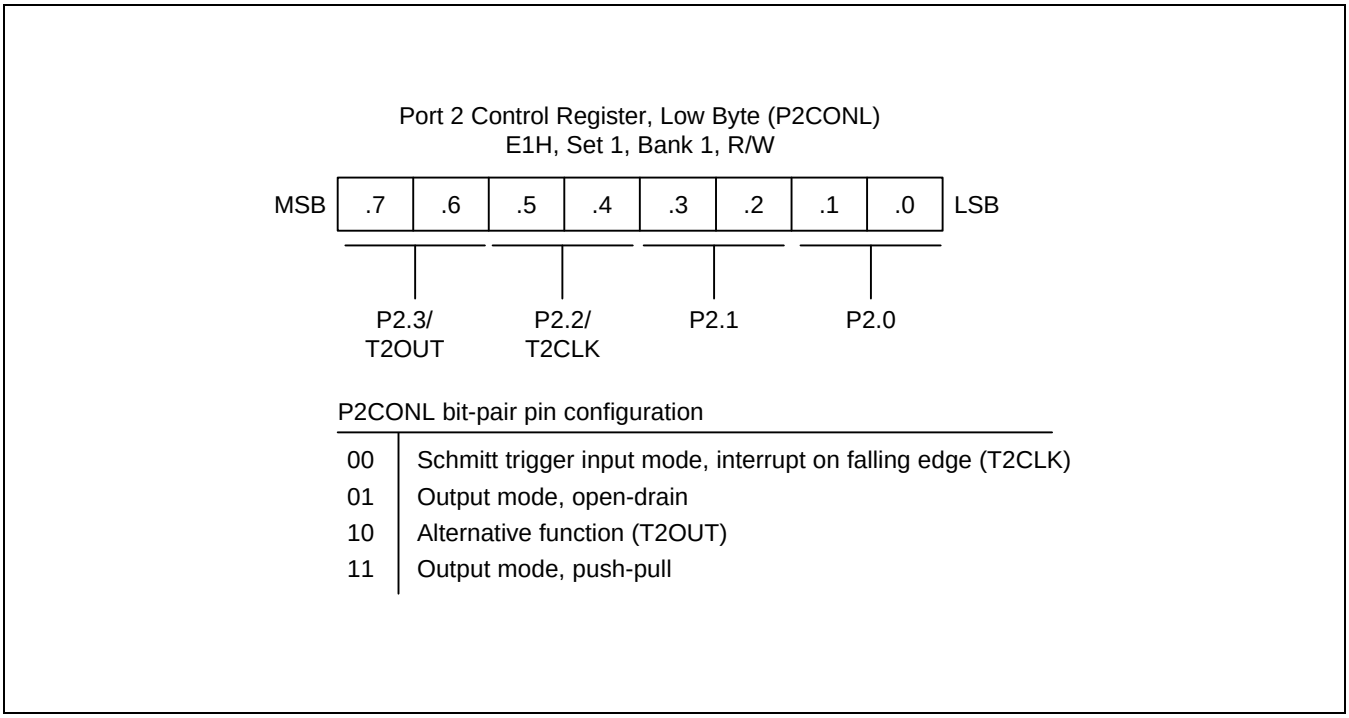
When the interrupt enable bit of any port 2 pin is “1”, a falling edge at that pin will generate an interrupt request. The corresponding pending bit is then automatically set to “1” and the IRQ level goes low to signal the CPU that an interrupt request is waiting. When the CPU acknowledges the interrupt request, application software must the clear the pending condition by writing a “0” to the corresponding P2INT bit.



P2CONH bit-pair pin configuration

00	Schmitt trigger input mode, interrupt on falling edge (T0CLK, T1CLK)
01	Output mpde, open-drain
10	Alternative function (TAOUT, TBOUT)
11	Output mode, push-pull

Figure 9-2. Port 2 High-Byte Control Register (P2CONH)



P2CONL bit-pair pin configuration

00	Schmitt trigger input mode, interrupt on falling edge (T2CLK)
01	Output mode, open-drain
10	Alternative function (T2OUT)
11	Output mode, push-pull

Figure 9-3. Port 2 Low-Byte Control Register (P2CONL)

Port 2 Pull-up Control Register (P2PUR)
E2H, Set 1, Bank 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	

P2PUR bit configuration settings:

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: The corresponding pull-up resistor is disabled automatically when a bit of port 2 is selected as output mode.

Figure 9-4. Port 2 Pull-up Control Register (P2PUR)

Port 2 Interrupt Control Register (P2INT)
E3H, Set 1, Bank 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	PND3	INT3	PND2	INT2	PND1	INT1	PND0	INT0	

INTn bit configuration settings:

0	Disable interrupt
1	Enable interrupt

PNDn bit configuration settings:

0	Interrupt request is not pending, pending bit clear when write 0
1	Interrupt request is pending

NOTE: "n" is 0, 1, 2, and 3.

Figure 9-5. Port 2 Interrupt Control Register (P2INT)

PORT 3

Port 3 is an 8-bit I/O port with individually configurable pins. Port 3 pins are accessed directly by writing or reading the Port 3 data register, P3 at location F3H in set 1, bank 1. P3.0–P3.7 can serve as inputs, as outputs (push pull or open-drain) or you can configure the following alternative functions:

— Low-nibble pins (P3.0-P3.3): AD0–AD3

— High-nibble pins (P3.4-P3.7): T3CLK, T3OUT/T3PWM/T3CAP, T0OUT/T0PWM/T0CAP

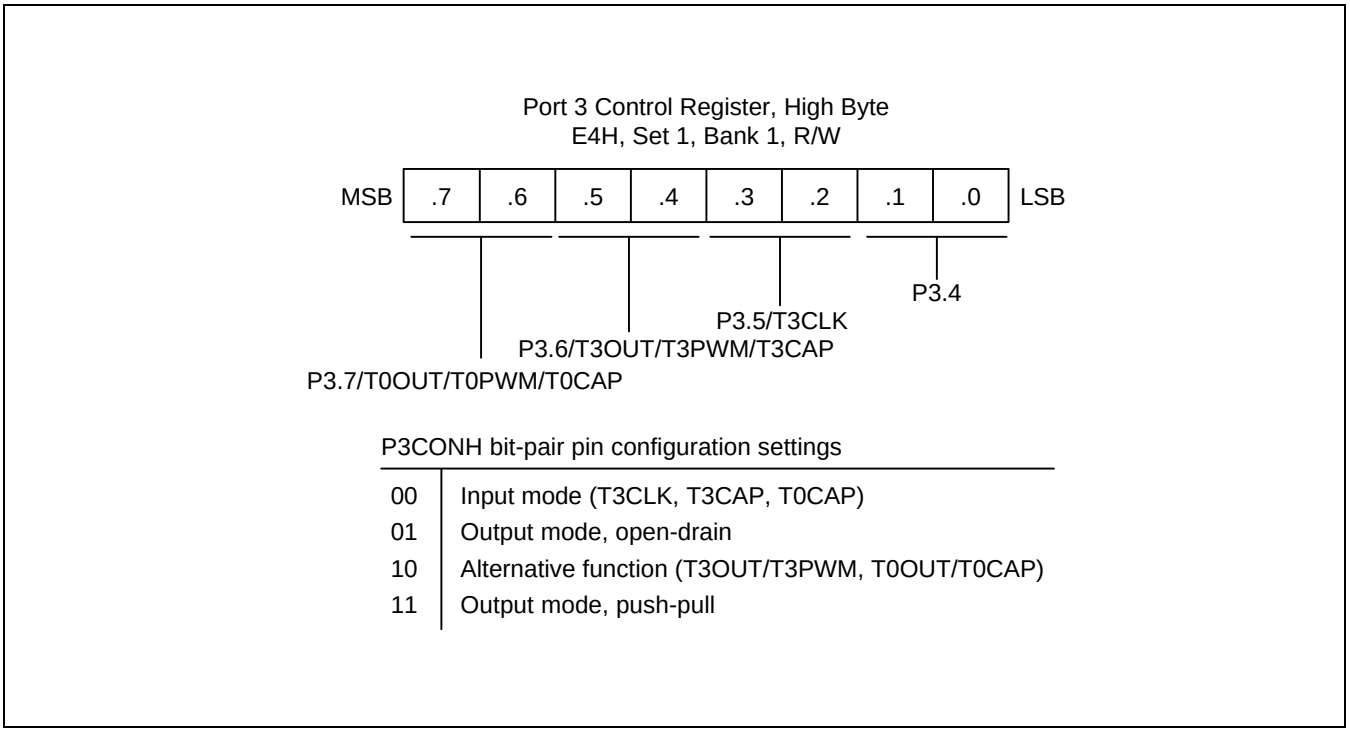
Port 3 Control Registers (P3CONH, P3CONL)

Port 3 has two 8-bit control registers: P3CONH for P3.4–P3.7 and P3CONL for P3.0–P3.3. A reset clears the P3CONH and P3CONL registers to "00H", configuring all pins to input mode. You use control registers settings to select input or output mode (push-pull or open drain) and enable the alternative functions.

When programming the port, please remember that any alternative peripheral I/O function you configure using the port 3 control registers must also be enabled in the associated peripheral module.

Port 3 Pull-up Resistor Control Register (P3PUR)

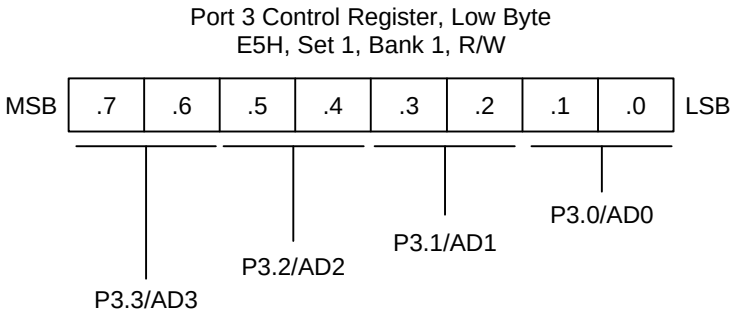
Using the port 3 pull-up resistor control register, P3PUR (E6H, set 1, bank 1), you can configure pull-up resistors to individual port 3 pins.



P3CONH bit-pair pin configuration settings

00	Input mode (T3CLK, T3CAP, T0CAP)
01	Output mode, open-drain
10	Alternative function (T3OUT/T3PWM, T0OUT/T0CAP)
11	Output mode, push-pull

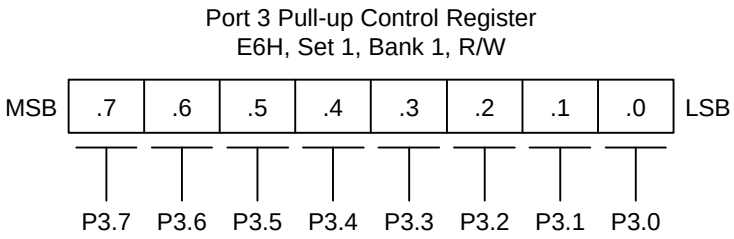
Figure 9-6. Port 3 High-Byte Control Register (P3CONH)



P3CONL bit-pair pin configuration settings

00	Input mode
01	Output mode, open-drain
10	Alternative function (AD0-AD3)
11	Output mode, push-pull

Figure 9-7. Port 3 Low-Byte Control Register (P3CONL)



P3PUR bit configuration settings:

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: The corresponding pull-up resistor is disabled automatically, when a bit of port 3 is selected as output mode.

Figure 9-8. Port 3 Pull-up Control Register (P3PUR)

PORT 4

Port 4 is an 8-bit I/O port with individually configurable pins. Port 4 pins are accessed directly by writing or reading the port 4 data register, P4 at location F4H in set 1, bank 1. P4.0–P4.7 can serve as inputs (with or without pull-up), as outputs (push-pull or open-drain) or you can be configured the following functions.

— Low-nibble pins (P4.0–P4.3): INT4–INT7

— High-nibble pins (P4.4–P4.7): INT8–INT11

Port 4 Control Registers (P4CONH, P4CONL)

Port 4 has two 8-bit control registers: P4CONH for P4.4–P4.7 and P4CONL for P4.0–P4.3. A reset clears the P4CONH and P4CONL registers to "00H", configuring pins to input mode. You use control registers setting to select input or output mode(push-pull or open-drain).

Port 4 Interrupt Enable, Pending, and Edge Selection Registers (P4INT, P4PND, P4EDGE)

To process external interrupts at the port 4 pins, three additional control registers are provided: the port 4 interrupt enable register P4INT (EAH, set 1, bank 1), the port 4 interrupt pending register P4PND (EBH, set1, bank 1), and the port 4 interrupt edge selection register P4EDGE (E7H, set 1, bank 1).

The port 4 interrupt pending register P4PND lets you check for interrupt pending conditions and clear the pending condition when the interrupt service routine has been initiated. The application program detects interrupt requests by polling the P4PND register at regular intervals.

When the interrupt enable bit of any port 4 pin is "1", a rising or falling edge at that pin will generate an interrupt request. The corresponding P4PND bit is then automatically set to "1" and the IRQ level goes low to signal the CPU that an interrupt request is waiting. When the CPU acknowledges the interrupt request, application software must the clear the pending condition by writing a "0" to the corresponding P4PND bit.

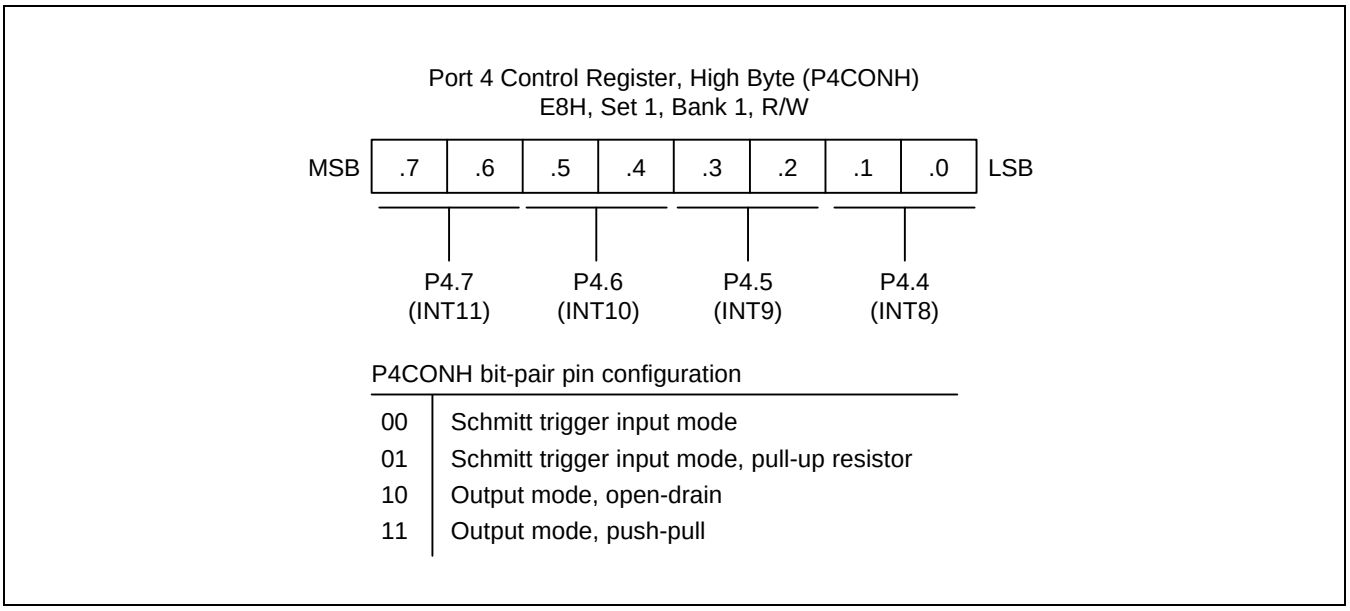


Figure 9-9. Port 4 High-Byte Control Register (P4CONH)

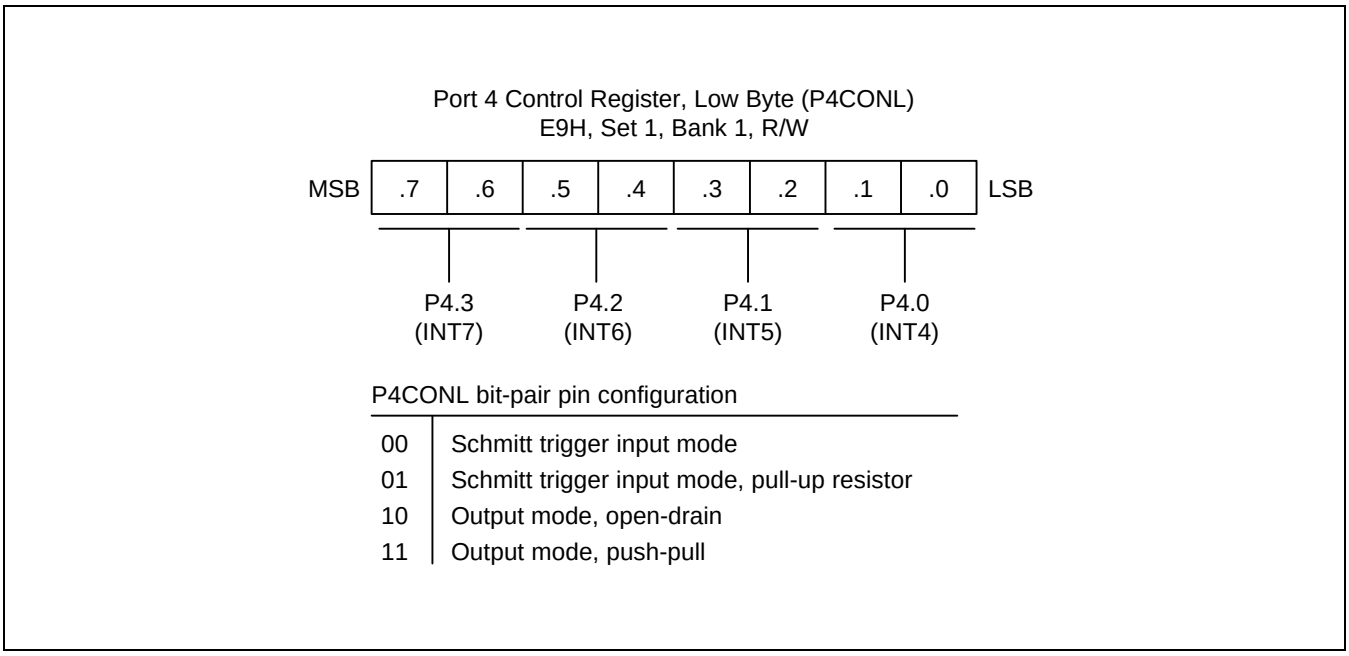


Figure 9-10. Port 4 Low-Byte Control Register (P4CONL)

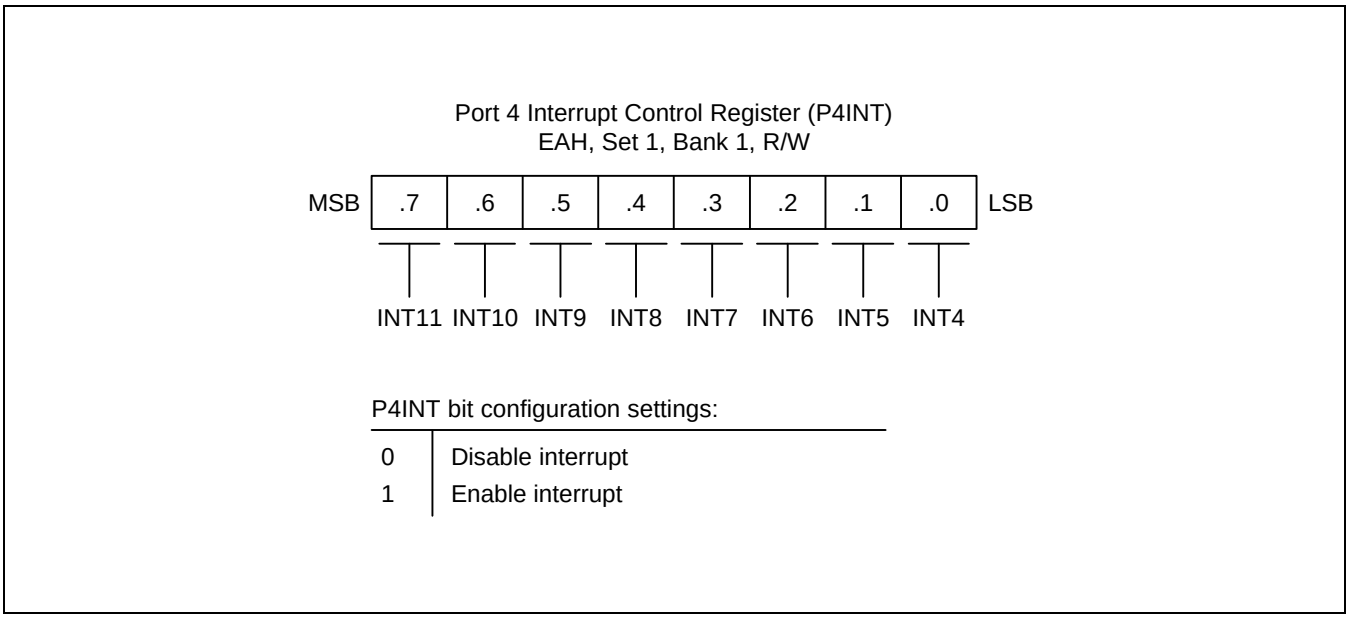


Figure 9-11. Port 4 Interrupt Control Register (P4INT)

Port 4 Interrupt Pending Register (P4PND)
EBH, Set 1, Bank 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	PND11	PND10	PND9	PND8	PND7	PND6	PND5	PND4	

P4PND bit configuration settings:

0	Interrupt request is not pending, pending bit clear when write 0
1	Interrupt request is pending

Figure 9-12. Port 4 Interrupt Pending Control Register (P4PND)

Port 4 Interrupt Edge Selection Register (P4EDGE)
E7H, Set 1, Bank 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	EDGE11	EDGE10	EDGE9	EDGE8	EDGE7	EDGE6	EDGE5	EDGE4	

P4EDGE bit configuration settings:

0	Falling edge detection
1	Rising edge detection

Figure 9-13. Port 4 Interrupt Edge Selection Register (P4EDGE)

PORT 5

Port 5 is an 7-bit I/O port with individually configurable pins. Port 5 pins are accessed directly by writing or reading the port 5 data register, P5 at location F5H in set 1, bank 1. P5.0–P5.6 can serve as inputs or as push-pull, open-drain outputs. You can configure the following alternative functions:

— Low-nibble pins (P5.0–P5.3): SCK, SI, SO, BUZ

— High-nibble pins (P5.4–P5.6): RXD, TXD

Port 5 Control Registers (P5CONH, P5CONL)

Port 5 has two 8-bit control registers: P5CONH for P5.4–P5.6 and P5CONL for P5.0–P5.3, and P5CONH.7–.6 for TXD/RXD output control. A reset clears the P5CONH and P5CONL registers to “00H”, configuring all pins to input mode and TXD/RXD to output disable. You use control registers settings to select input or output mode, and enable the alternative functions.

When programming this port, please remember that any alternative peripheral I/O function you configure using the port 5 control registers must also be enabled in the associated peripheral module.

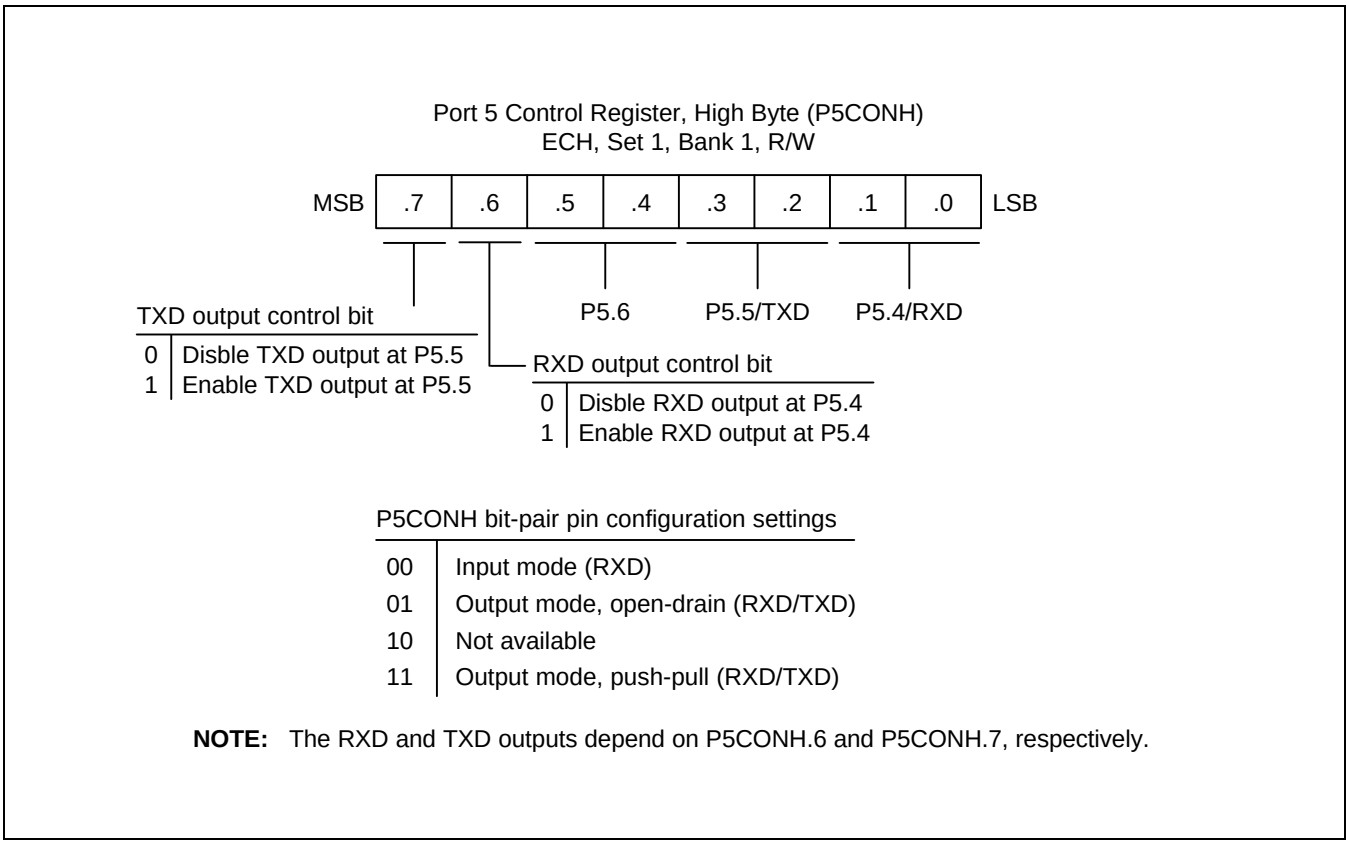


Figure 9-14. Port 5 High-Byte Control Register (P5CONH)

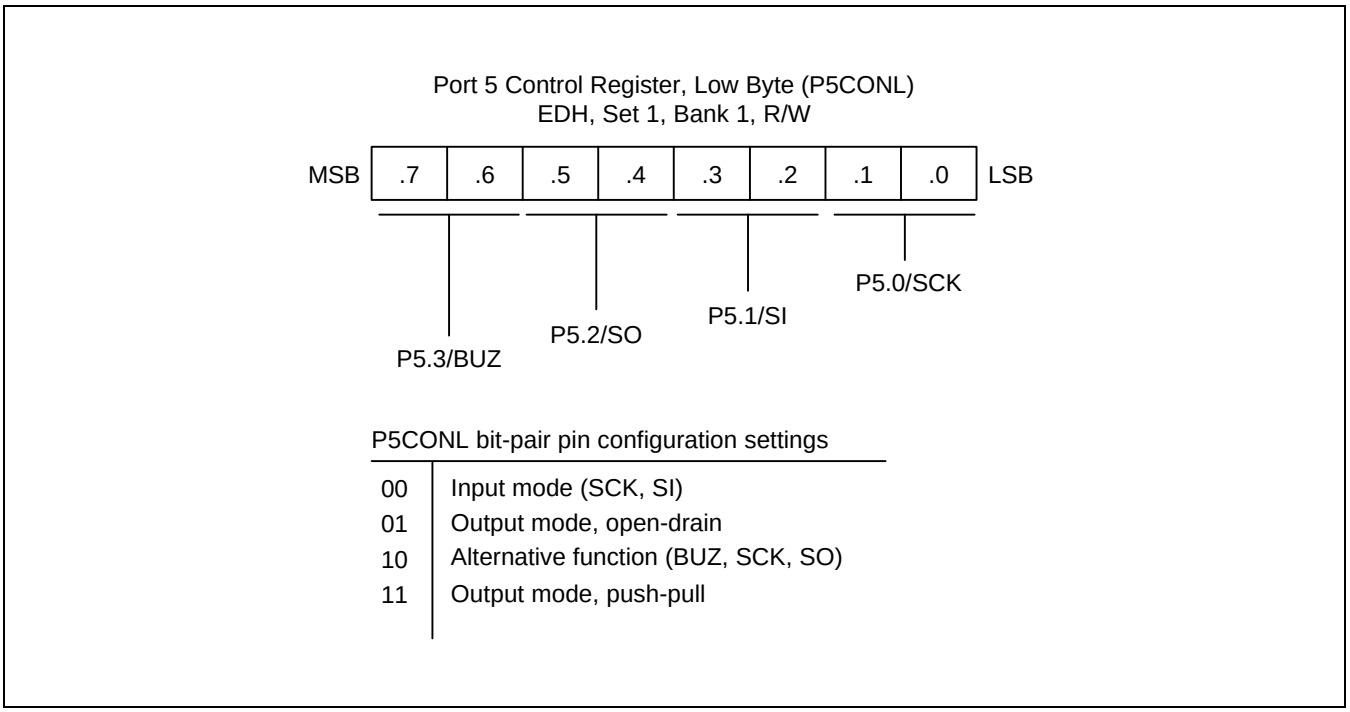


Figure 9-15. Port 5 Low-Byte Control Register (P5CONL)

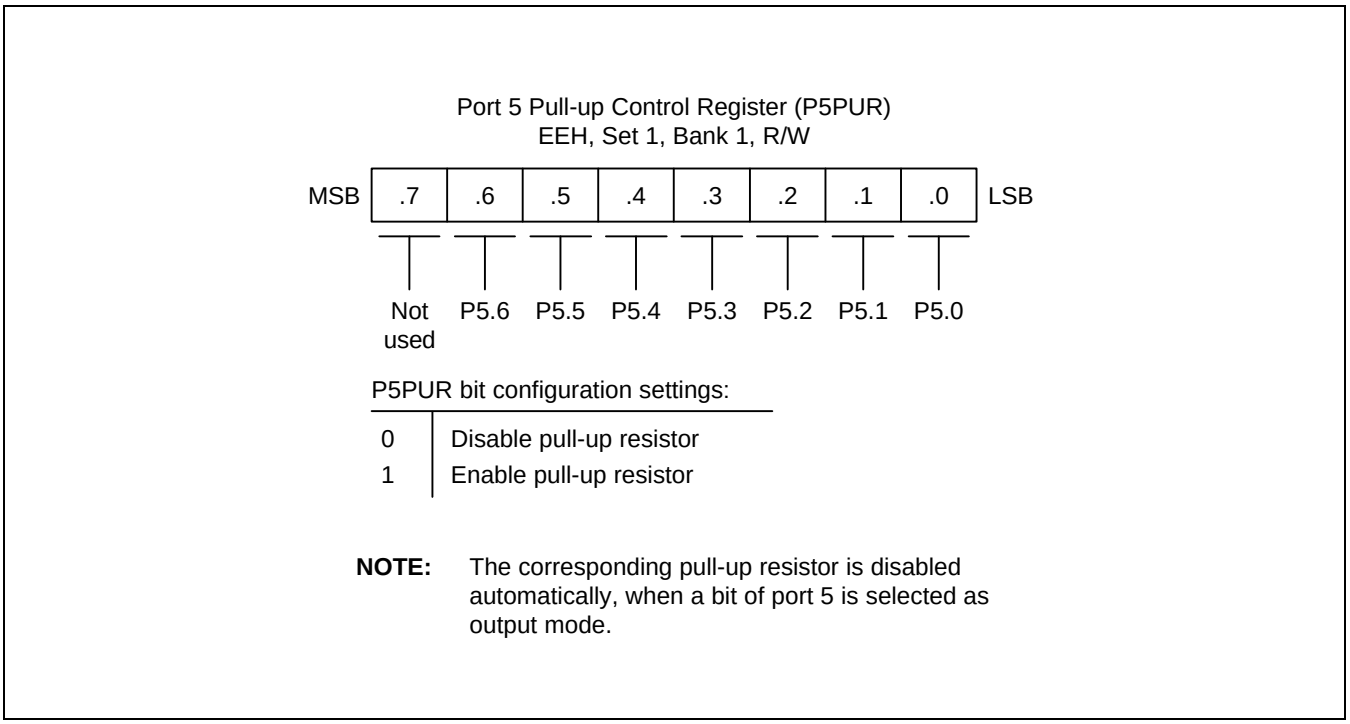


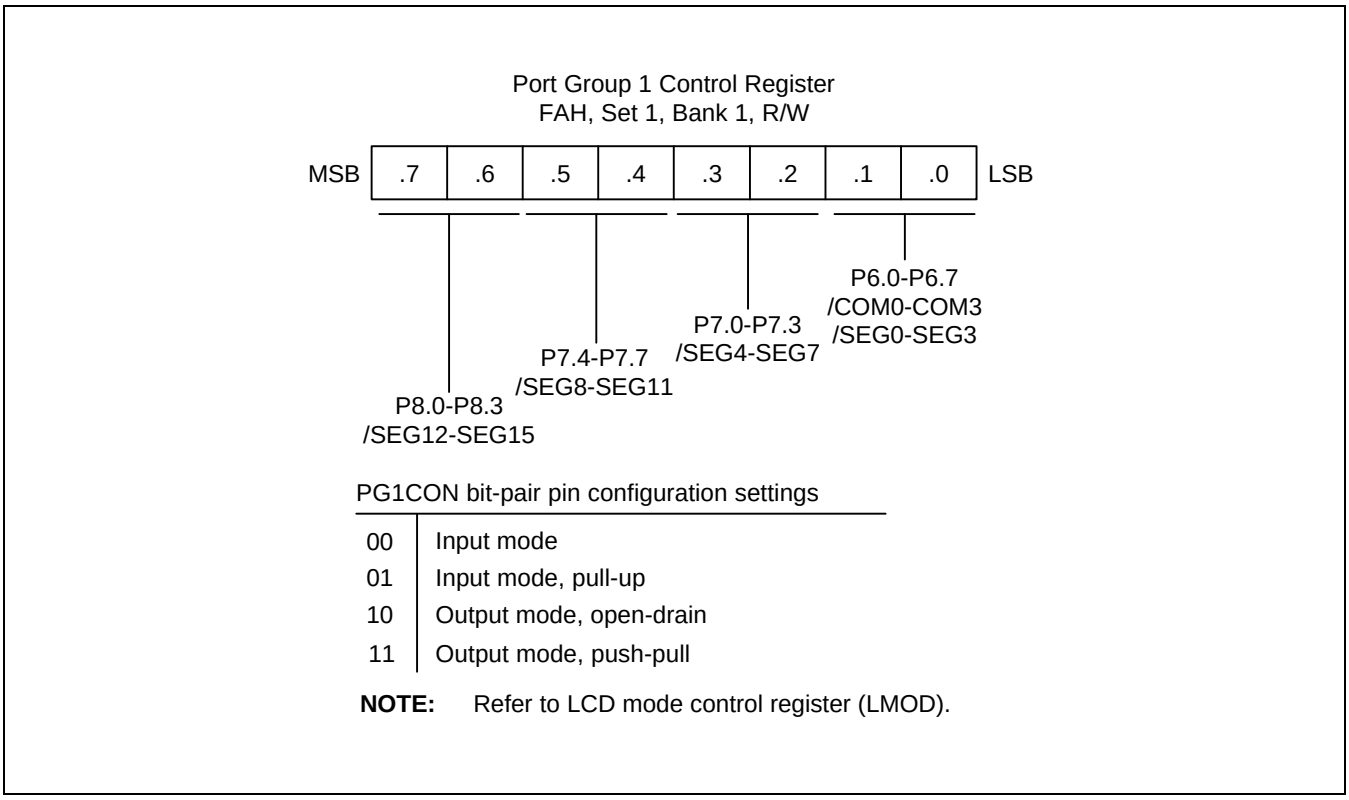
Figure 9-16. Port 5 Pull-up Control Register (P5PUR)

PORT 6, 7, 8

Port 6 and 7 are 8-bit I/O port and port 8 is 4-bit I/O port with nibble configurable pins, respectively. Port 6, 7, and 8 pins are accessed directly by writing or reading the port 6, 7, and 8 data registers, P6 at location F6H, P7 at location F7H, and P8 at location F8H in set 1, bank 1. P6.0–P6.7, P7.0–P7.7, and P8.0–P8.3 can serve as inputs (with or without pull-ups), as output (open drain or push-pull). And they can serve as segment or common pins for LCD also.

Port Group 1 Control Register (PG1CON)

Port 6, 7, and 8 have an 8-bit control register: PG1CON.0–.1 for P6.0–P6.7, PG1CON.2–.5 for P7.0–P7.7, and PG1CON.6–.7 for P8.0–P8.3. A reset clears the PG1CON register to “00H”, configuring all pins to input mode.



PG1CON bit-pair pin configuration settings

00	Input mode
01	Input mode, pull-up
10	Output mode, open-drain
11	Output mode, push-pull

NOTE: Refer to LCD mode control register (LMOD).

Figure 9-17. Port Group 1 Control Register (PG1CON)

NOTES

10 BASIC TIMER and TIMER 0

OVERVIEW

The S3C825A has two default timers: an 8-bit *basic timer* and one 8-bit general-purpose timer/counter. The 8-bit timer/counter is called *timer 0.*

BASIC TIMER (BT)

You can use the basic timer (BT) in two different ways:

— As a watchdog timer to provide an automatic reset mechanism in the event of a system malfunction.
— To signal the end of the required oscillation stabilization interval after a reset or a stop mode release.

The functional components of the basic timer block are:

— Clock frequency divider (fxx divided by 4096, 1024, 128, or 16) with multiplexer
— 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH, read-only)
— Basic timer control register, BTCON (set 1, D3H, read/write)

BASIC TIMER CONTROL REGISTER (BTCON)

The basic timer control register, BTCON, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function. It is located in set 1, address D3H, and is read/write addressable using Register addressing mode.

A reset clears BTCON to “00H”. This enables the watchdog function and selects a basic timer clock frequency of fxx/4096. To disable the watchdog function, you must write the signature code “1010B” to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH), can be cleared at any time during normal operation by writing a "1" to BTCON.1. To clear the frequency dividers for all timers input clock, you write a "1" to BTCON.0.

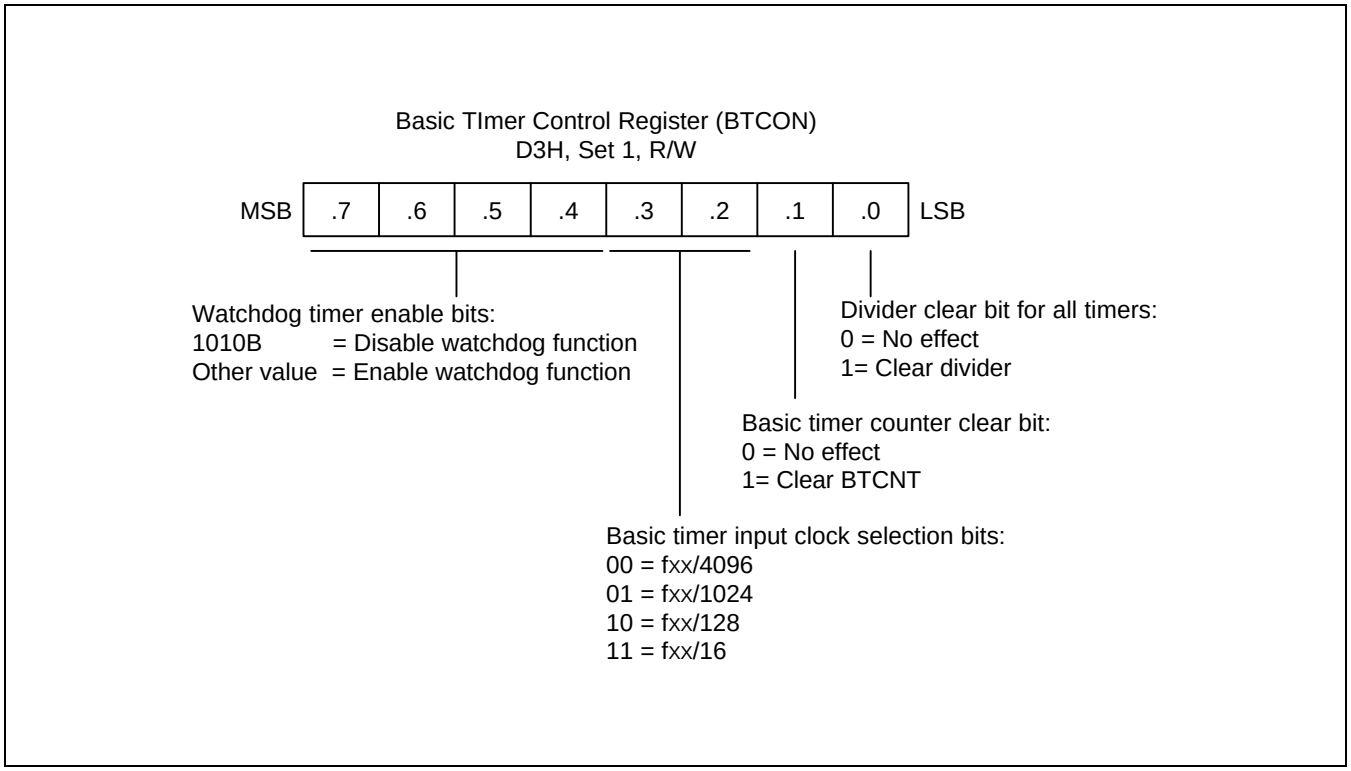


Figure 10-1. Basic Timer Control Register (BTCON)

BASIC TIMER FUNCTION DESCRIPTION

Watchdog Timer Function

You can program the basic timer overflow signal (BTOVF) to generate a reset by setting BTCON.7–BTCON.4 to any value other than “1010B”. (The “1010B” value disables the watchdog function.) A reset clears BTCON to “00H”, automatically enabling the watchdog timer function. A reset also selects the CPU clock (as determined by the current CLKCON register setting), divided by 4096, as the BT clock.

A reset whenever a basic timer counter overflow occurs. During normal operation, the application program must prevent the overflow, and the accompanying reset operation, from occurring. To do this, the BTCNT value must be cleared (by writing a "1" to BTCON.1) at regular intervals.

If a system malfunction occurs due to circuit noise or some other error condition, the BT counter clear operation will not be executed and a basic timer overflow will occur, initiating a reset. In other words, during normal operation, the basic timer overflow loop (a bit 7 overflow of the 8-bit basic timer counter, BTCNT) is always broken by a BTCNT clear instruction. If a malfunction does occur, a reset is triggered automatically.

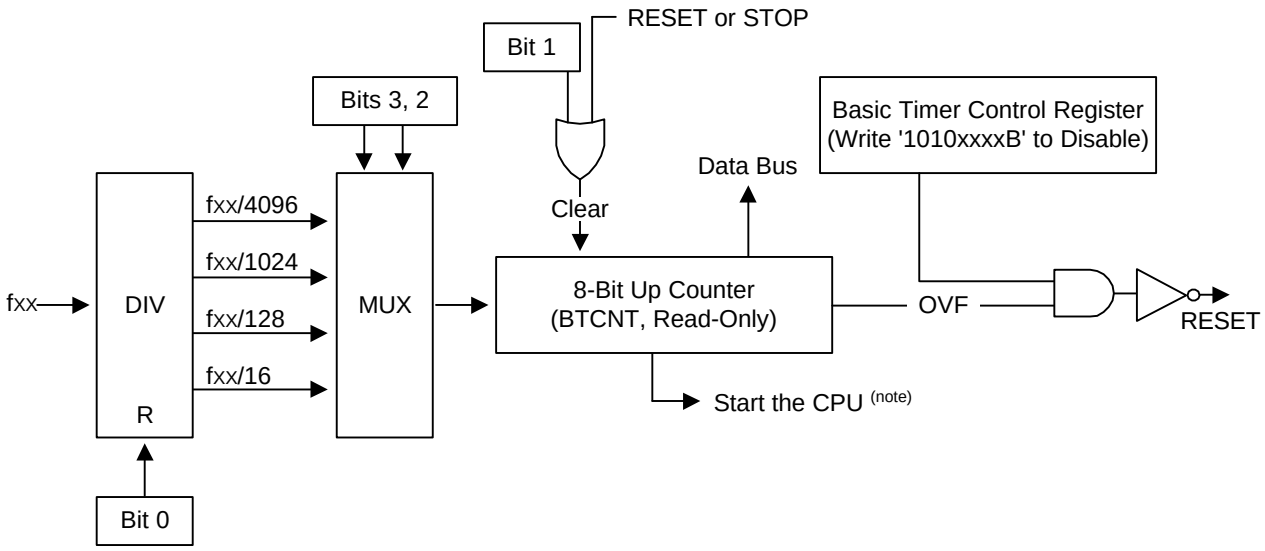
Oscillation Stabilization Interval Timer Function

You can also use the basic timer to program a specific oscillation stabilization interval following a reset or when stop mode has been released by an external interrupt.

In stop mode, whenever a reset or an internal and an external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of fxx/4096 (for reset), or at the rate of the preset clock source (for an internal and an external interrupt). When BTCNT.3 overflows, a signal is generated to indicate that the stabilization interval has elapsed and to gate the clock signal off to the CPU so that it can resume normal operation.

In summary, the following events occur when stop mode is released:

1. During stop mode, a power-on reset or an internal and an external interrupt occurs to trigger the stop mode release and oscillation starts.
2. If a power-on reset occurred, the basic timer counter will increase at the rate of fxx/4096. If an internal and an external interrupt is used to release stop mode, the BTCNT value increases at the rate of the preset clock source.
3. Clock oscillation stabilization interval begins and continues until bit 3 of the basic timer counter overflows.
4. When a BTCNT.3 overflow occurs, normal CPU operation resumes.



NOTE: During a power-on reset operation, the CPU is idle during the required oscillation stabilization interval (until bit 4 of the basic timer counter overflows).

Figure 10-2. Basic Timer Block Diagram

8-BIT TIMER/COUNTER 0

Timer/counter 0 has three operating modes, one of which you select using the appropriate T0CON setting:

— Interval timer mode
— Capture input mode with a rising or falling edge trigger at the P3.7 pin
— PWM mode

Timer/counter 0 has the following functional components:

— Clock frequency divider (fxx divided by 1024, 256, 64, 8, or 1) with multiplexer
— External clock input (P2.4, T0CLK)
— 8-bit counter (T0CNT), 8-bit comparator, and 8-bit reference data register (T0DATA)
— I/O pins for capture input, match output, or PWM output (P3.7/T0CAP, P3.7/T0OUT, P3.7/T0PWM)
— Timer 0 overflow interrupt (IRQ0, vector E2H) and match/capture interrupt (IRQ0, vector E0H) generation
— Timer 0 control register, T0CON (set 1, E5H, bank 0, read/write)

TIMER/COUNTER 0 CONTROL REGISTER (T0CON)

You use the timer 0 control register, T0CON, to

— Select the timer 0 operating mode (interval timer, capture mode, or PWM mode)
— Select the timer 0 input clock frequency
— Clear the timer 0 counter, T0CNT
— Enable the timer 0 overflow interrupt or timer 0 match/capture interrupt
— Clear timer 0 match/capture interrupt pending condition

T0CON is located in set 1, bank 0, at address E5H, and is read/write addressable using Register addressing mode.

A reset clears T0CON to “00H”. This sets timer 0 to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all timer 0 interrupts. You can clear the timer 0 counter at any time during normal operation by writing a "1" to T0CON.2.

The timer 0 overflow interrupt (T0OVF) is interrupt level IRQ0 and has the vector address E2H. When a timer 0 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the timer 0 match/capture interrupt (IRQ0, vector E0H), you must write T0CON.1 to "1". To detect a match/capture interrupt pending condition, the application program polls INTPND.1. When a "1" is detected, a timer 0 match or capture interrupt is pending. When the interrupt request has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 0 match/capture interrupt pending bit, INTPND.1.

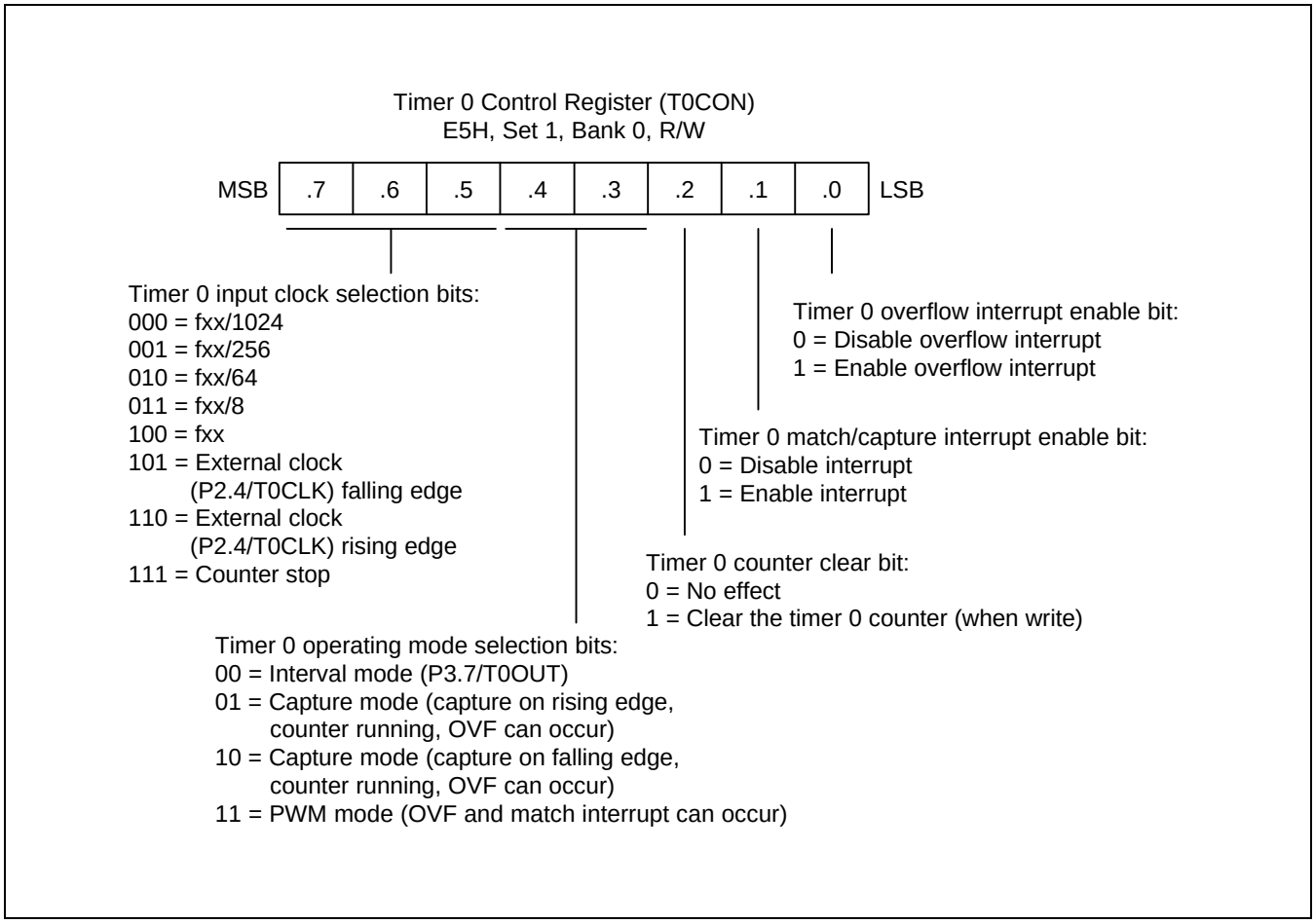


Figure 10-3. Timer 0 Control Register (T0CON)

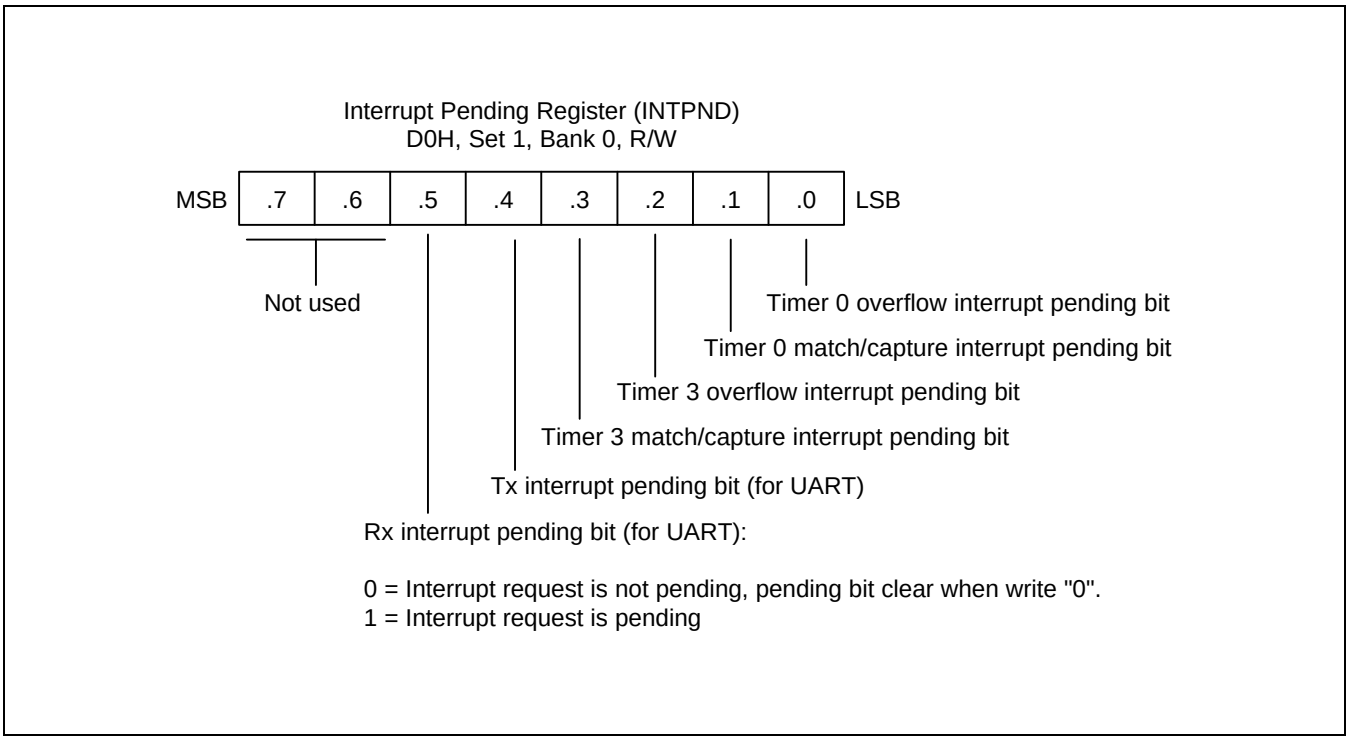


Figure 10-4. Interrupt Pending Register (INTPND)

TIMER 0 FUNCTION DESCRIPTION

Timer 0 Interrupts (IRQ0, Vectors E0H and E2H)

The timer 0 can generate two interrupts: the timer 0 overflow interrupt (T0OVF), and the timer 0 match/ capture interrupt (T0INT). T0OVF is belongs to interrupt level IRQ0, vector E2H. T0INT also belongs to interrupt level IRQ0, but is assigned the separate vector address, E0H.

A timer 0 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced or should be cleared by software in the interrupt service routine by writing a “0” to the INTPND.0 interrupt pending bit. However, the timer 0 match/capture interrupt pending condition must be cleared by the application’s interrupt service routine by writing a "0" to the INTPND.1 interrupt pending bit.

Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 0 reference data register, T0DATA. The match signal generates a timer 0 match interrupt (T0INT, vector E0H) and clears the counter.

If, for example, you write the value "10H" to T0DATA, the counter will increment until it reaches “10H”. At this point, the timer 0 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer 0 output pin is inverted (see Figure 10-5).

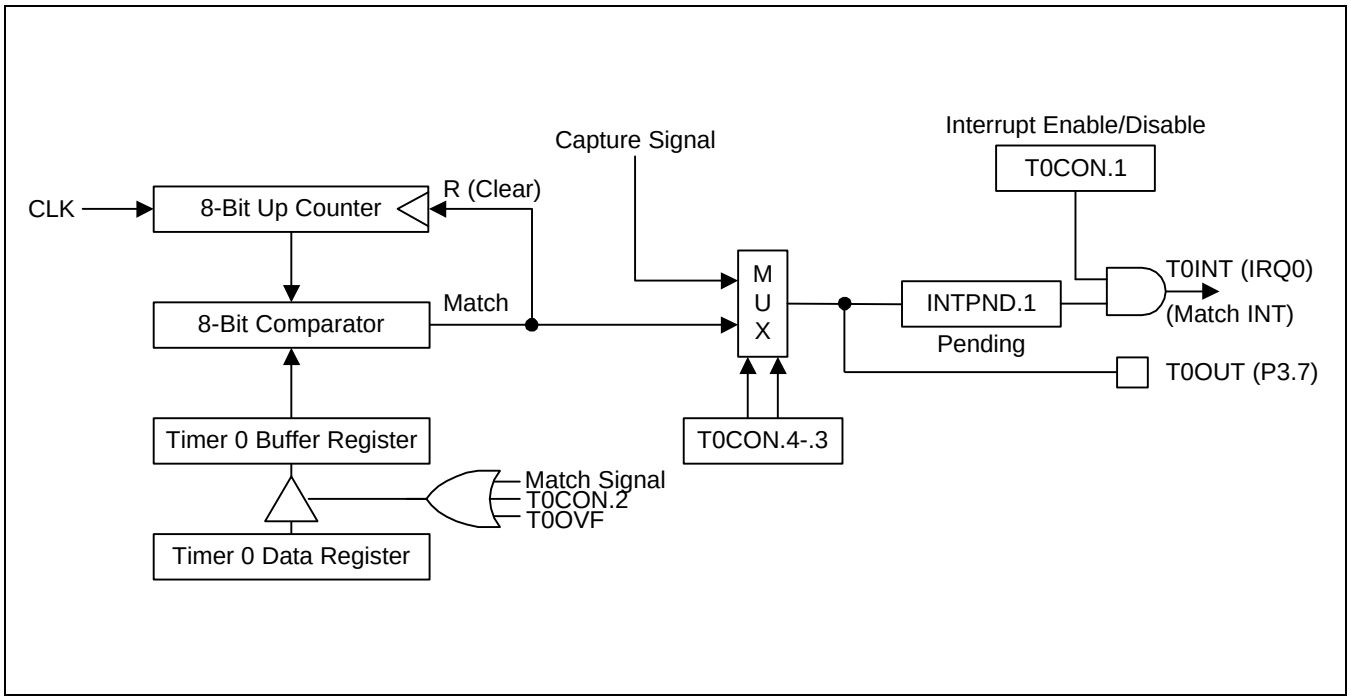


Figure 10-5. Simplified Timer 0 Function Diagram: Interval Timer Mode

Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the T0PWM (P3.7) pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 0 data register. In PWM mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at "FFH", and then continues incrementing from "00H".

Although you can use the match signal to generate a timer 0 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the T0PWM (P3.7) pin is held to Low level as long as the reference data value is *less than or equal to* (≤) the counter value and then the pulse is held to High level for as long as the data value is *greater than* (>) the counter value. One pulse width is equal to $t_{CLK} \times 256$ (see Figure 10-6).

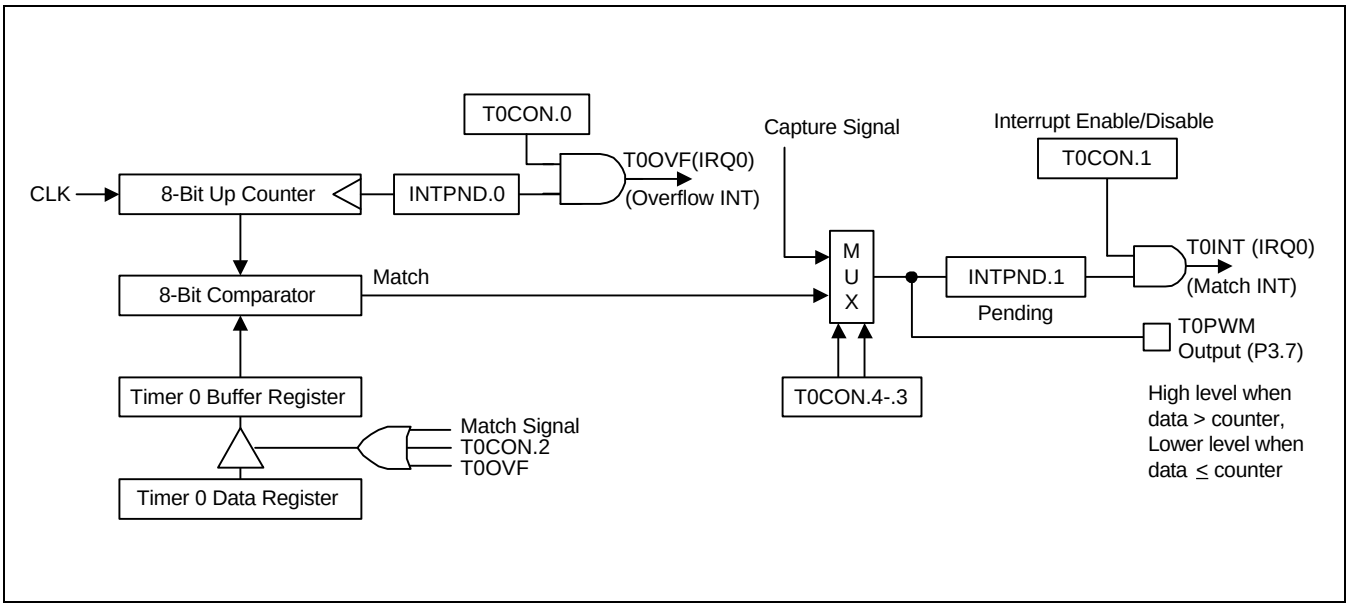


Figure 10-6. Simplified Timer 0 Function Diagram: PWM Mode

Capture Mode

In capture mode, a signal edge that is detected at the T0CAP (P3.7) pin opens a gate and loads the current counter value into the timer 0 data register. You can select rising or falling edges to trigger this operation.

Timer 0 also gives you capture input source: the signal edge at the T0CAP (P3.7) pin. You select the capture input by setting the values of the timer 0 capture input selection bits in the port 3 control register, P3CONH.7–.6, (set 1, bank 1, E4H). When P3CONH.7–.6 is "00", the T0CAP input is selected.

Both kinds of timer 0 interrupts can be used in capture mode: the timer 0 overflow interrupt is generated whenever a counter overflow occurs; the timer 0 match/capture interrupt is generated whenever the counter value is loaded into the timer 0 data register.

By reading the captured data value in T0DATA, and assuming a specific value for the timer 0 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the T0CAP pin (see Figure 10-7).

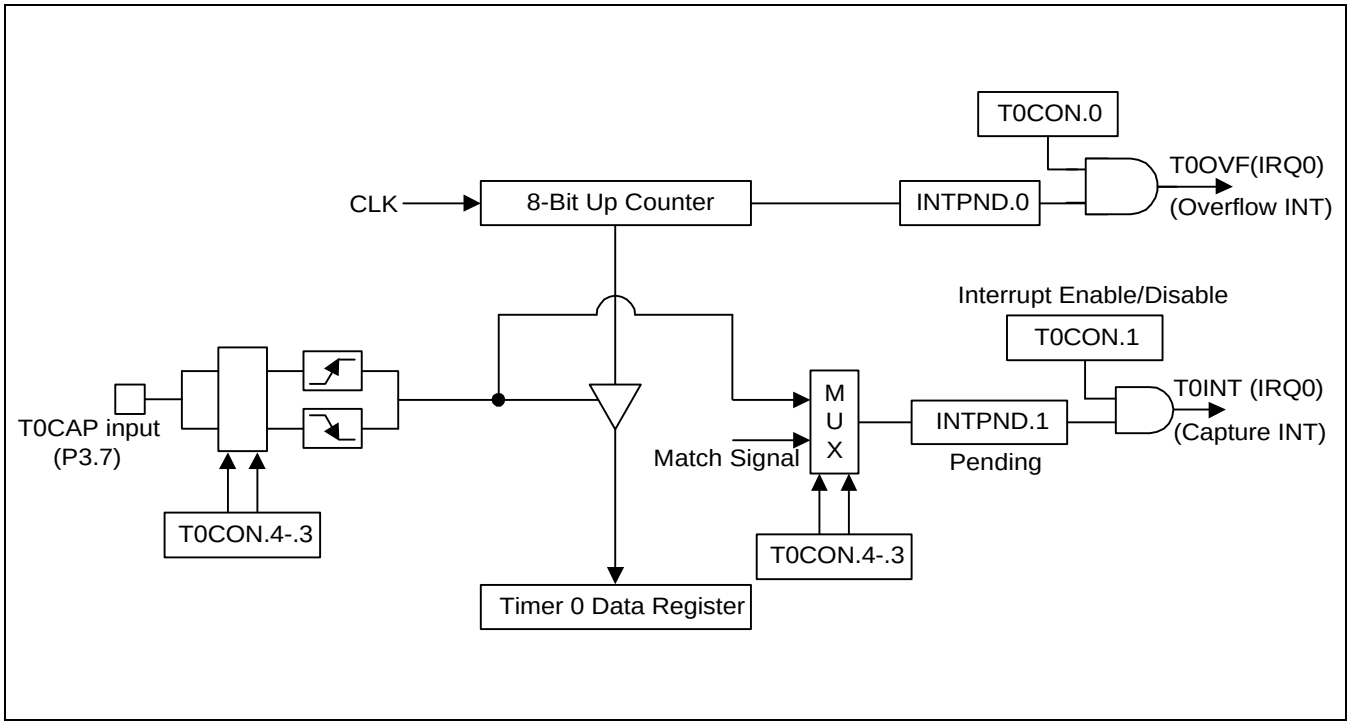


Figure 10-7. Simplified Timer 0 Function Diagram: Capture Mode

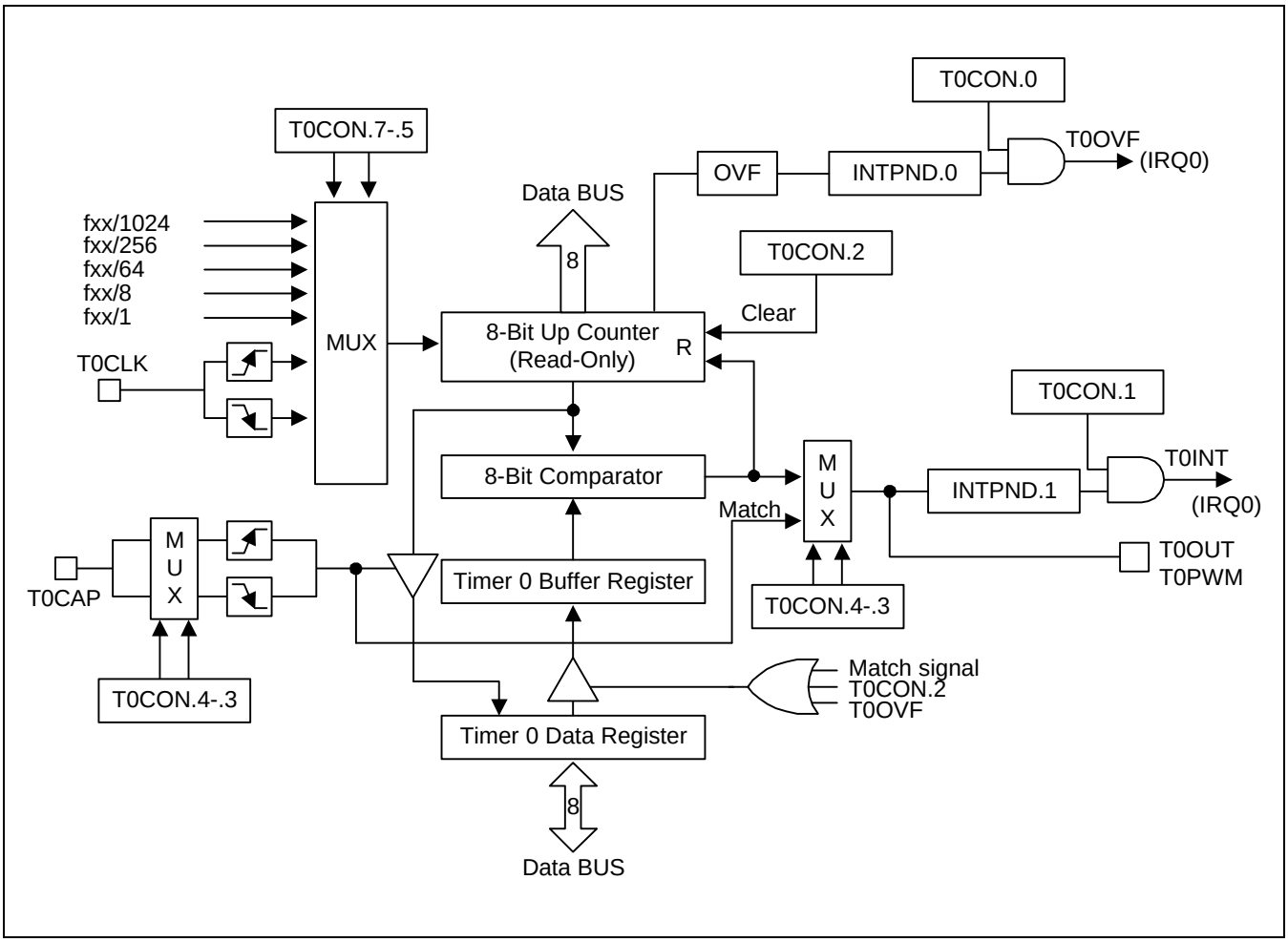


Figure 10-8. Timer 0 Block Diagram

NOTES

11 TIMER 1

ONE 16-BIT TIMER MODE (TIMER 1)

The 16-bit timer 1 is used in one 16-bit timer or two 8-bit timers mode. When TACON.7 is set to "1", it is in one 16-bit timer mode. When TACON.7 is set to "0", the timer 1 is used as two 8-bit timers.

— One 16-bit timer mode (Timer 1)
— Two 8-bit timers mode (Timer A and B)

OVERVIEW

The 16-bit timer 1 is an 16-bit general-purpose timer. Timer 1 includes interval timer mode using appropriate TACON setting.

Timer 1 has the following functional components:

— Clock frequency divider (fxx divided by 256, 64, 8, or 1 and T1CLK: External clock) with multiplexer
— 16-bit counter (TACNT, TBCNT), 16-bit comparator, and 16-bit reference data register (TADATA, TBDATA)
— Timer 1 match interrupt (IRQ1, vector E6H) generation
— Timer 1 control register, TACON (set 1, bank 0,EBH, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

The timer 1 module can generate an interrupt, the timer 1 match interrupt (T1INT). T1INT belongs to the interrupt level IRQ1, and is assigned a separate vector address, E6H.

The T1INT pending condition should be cleared by software after IRQ1 is serviced. The T1INT pending bit must be cleared by the application sub-routine by writing a "0" to the TACON.0 pending bit.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the T1 reference data registers, TADATA and TBDATA. The match signal generates a timer 1 match interrupt (T1INT, vector E6H) and clears the counter.

If, for example, you write the value 10H and 32H to TADATA and TBDATA, respectively, and 8EH to TACON, the counter will increment until it reaches 3210H. At this point, the T1 interrupt request is generated, the counter value is reset, and counting resumes.

Timer 1 Control Register (TACON)

You use the timer 1 control register, TACON, to

— Enable the timer 1 operating (interval timer)

— Select the timer 1 input clock frequency

— Clear the timer 1 counter, TACNT and TBCNT

— Enable the timer 1 interrupt

— Clear timer 1 interrupt pending conditions

TACON is located in set 1, bank 0, at address EBH, and is read/write addressable using register addressing mode.

A reset clears TACON to "00H". This sets timer 1 to disable interval timer mode, selects an input clock frequency of fxx/256, and disables timer 1 interrupt. You can clear the timer 1 counter at any time during the normal operation by writing a "1" to TACON.3.

To enable the timer 1 interrupt (IRQ1, vector E6H), you must write TACON.7, TACON.2, and TACON.1 to "1". To generate the exact time interval, you should set TACON.3 and TACON.0 to “10B”, which clear counter and interrupt pending bit. When the T1INT sub-routine is serviced, the pending condition must be cleared by software by writing a "0" to the timer 1 interrupt pending bit, TACON.0.

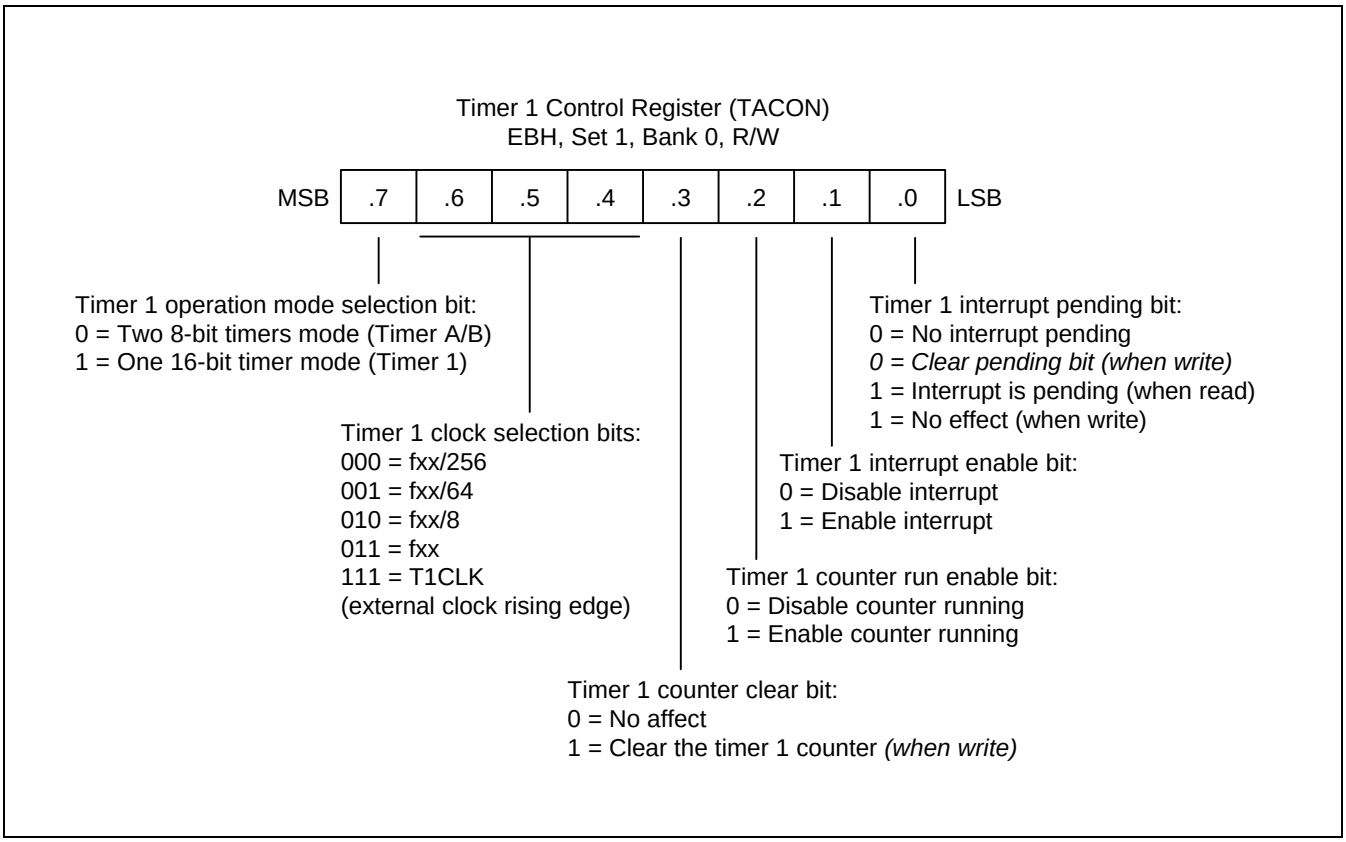
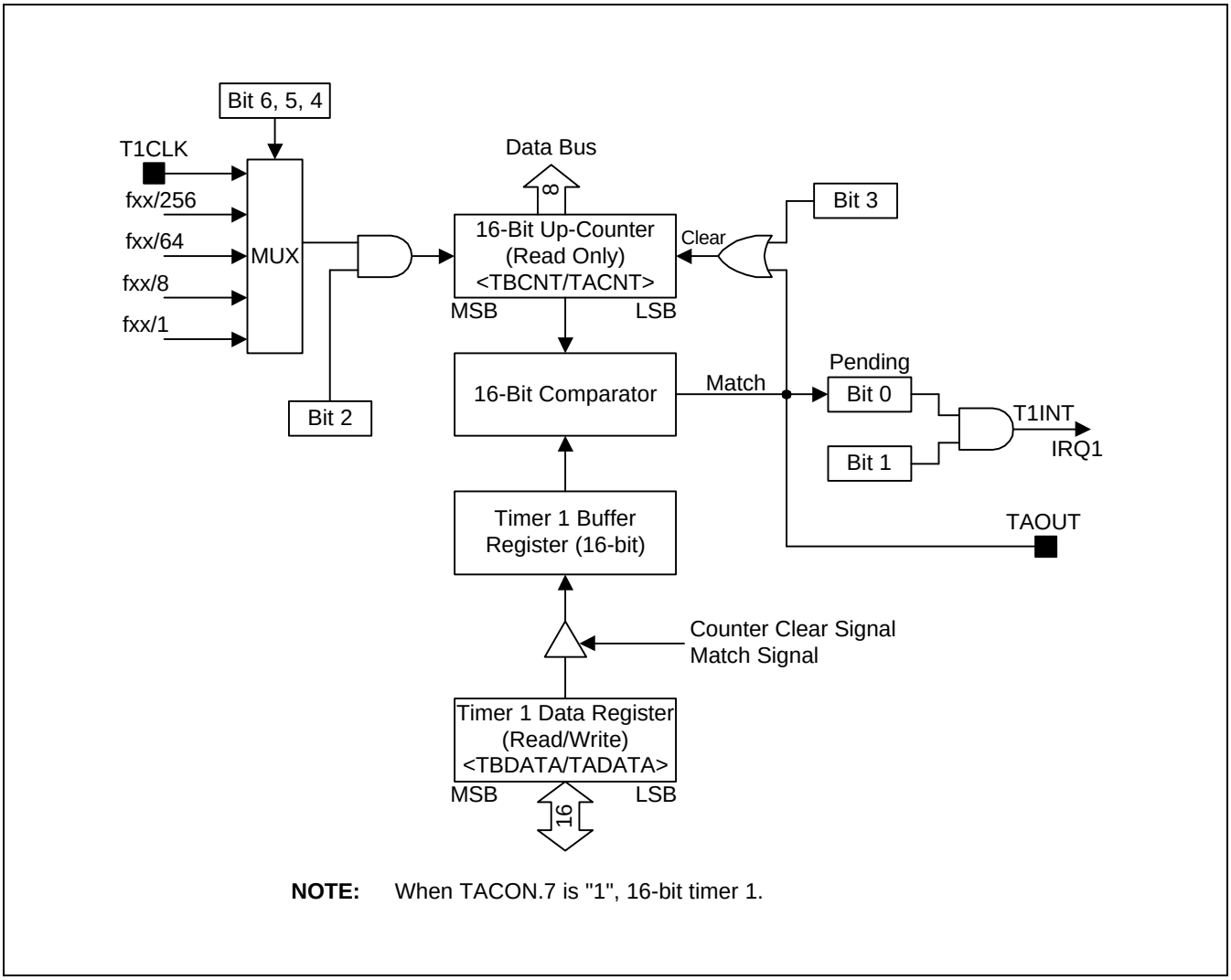


Figure 11-1. Timer 1 Control Register (TACON)

BLOCK DIAGRAM



NOTE: When TACON.7 is "1", 16-bit timer 1.

Figure 11-2. Timer 1 Functional Block Diagram

TWO 8-BIT TIMERS MODE (TIMER A and B)

OVERVIEW

The 8-bit timer A and B are the 8-bit general-purpose timers. Timer A and B support interval timer mode using appropriate TACON and TBCON setting, respectively.

Timer A and B have the following functional components:

— Clock frequency divider with multiplexer
  – fxx divided by 256, 64, 8, or 1 and T1CLK (External clock) for timer A
  – fxx divided by 256, 64, 8, or 1 for timer B
— 8-bit counter (TACNT, TBCNT), 8-bit comparator, and 8-bit reference data register (TADATA, TBDATA)
— Timer A match interrupt (IRQ1, vector E6H) generation
— Timer A control register, TACON (set 1, bank 0, EBH, read/write)
— Timer B match interrupt (IRQ1, vector E4H) generation
— Timer B control register, TBCON (set 1, bank 0, EAH, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

The timer A and B module can generate an interrupt: the timer A match interrupt (TAINT) and the timer B match interrupt (TBINT). TAINT belongs to the interrupt level IRQ1, and is assigned a separate vector address, E6H. TBINT belongs to the interrupt level IRQ1 and is assigned a separate vector address, E4H.

The TAINT and TBINT pending condition should be cleared by software after they are serviced.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the TA or TB reference data registers, TADATA or TBDATA. The match signal generates corresponding match interrupt (TAINT, vector E6H; TBINT, vector E4H) and clears the counter.

If, for example, you write the value 10H to TBDATA, "0" to TACON.7, and 0EH to TBCON, the counter will increment until it reaches 10H. At this point, the TB interrupt request is generated, the counter value is reset, and counting resumes.

Timer A and B Control Register (TACON, TBCON)

You use the timer A and B control register, TACON and TBCON, to

— Enable the timer A and B operating (interval timer)
— Select the timer A and B input clock frequency
— Clear the timer A and B counter, TACNT and TBCNT
— Enable the timer A and B interrupt
— Clear timer A and B interrupt pending conditions

TACON and TBCON are located in set 1, bank 0, at address EBH and EAH, and is read/write addressable using register addressing mode.

A reset clears TACON and TBCON to "00H". This sets timer A and B to disable interval timer mode, selects an input clock frequency of fxx/256, and disables timer A and B interrupt. You can clear the timer A and B counter at any time during normal operation by writing a "1" to TACON.3 and TBCON.3.

To enable the timer A and B interrupt (IRQ1, vector E6H, E4H), you must write TACON.7 to "0", TACON.2 (TBCON.2) and TACON.1 (TBCON.1) to "1". To generate the exact time interval, you should set TACON.3 (TBCON.3) and TACON.0 (TBCON.0) to “10B”, which clear counter and interrupt pending bit, respectively. When the TAINT or TBINT sub-routine is serviced, the pending condition must be cleared by software by writing a "0" to the timer A or B interrupt pending bits, TACON.0 or TBCON.0.

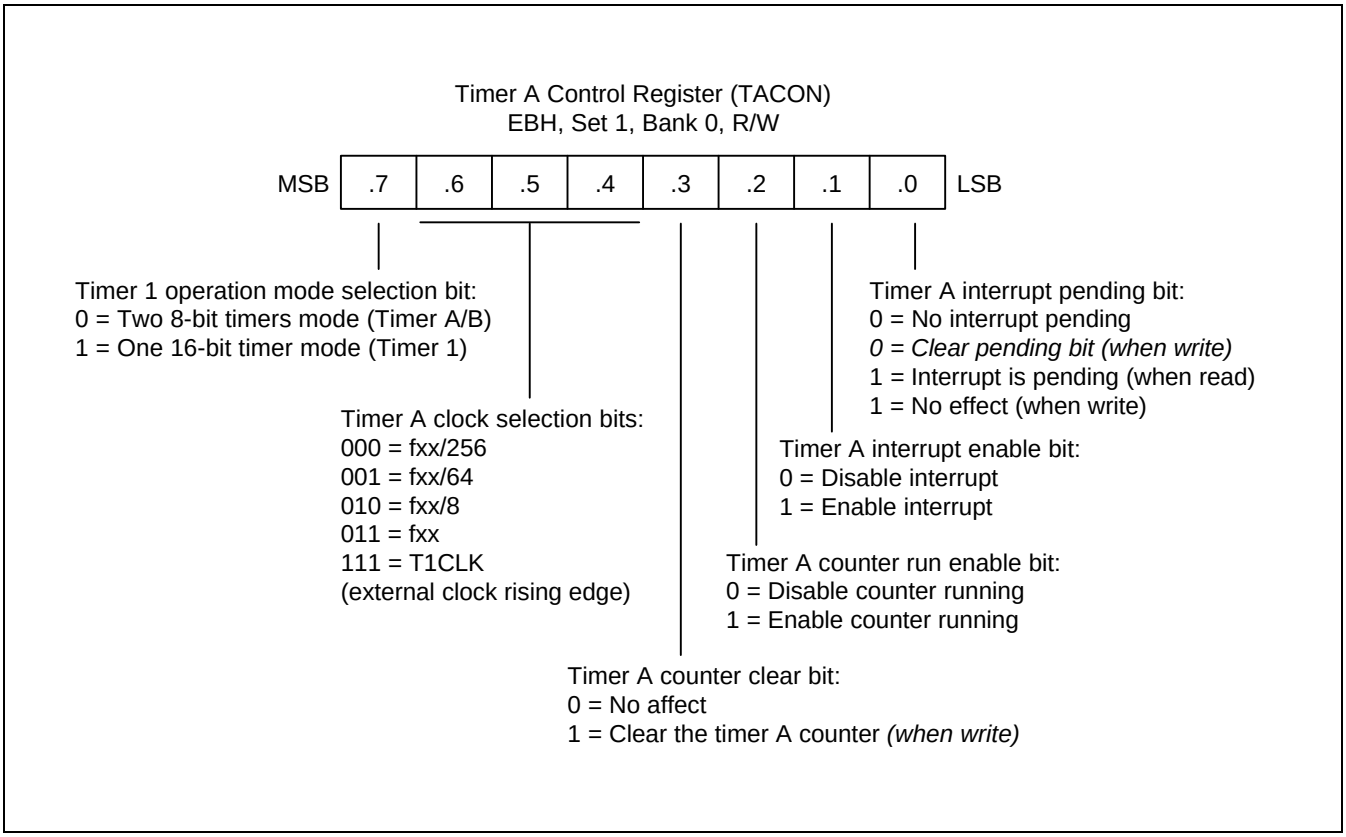


Figure 11-3. Timer A Control Register (TACON)

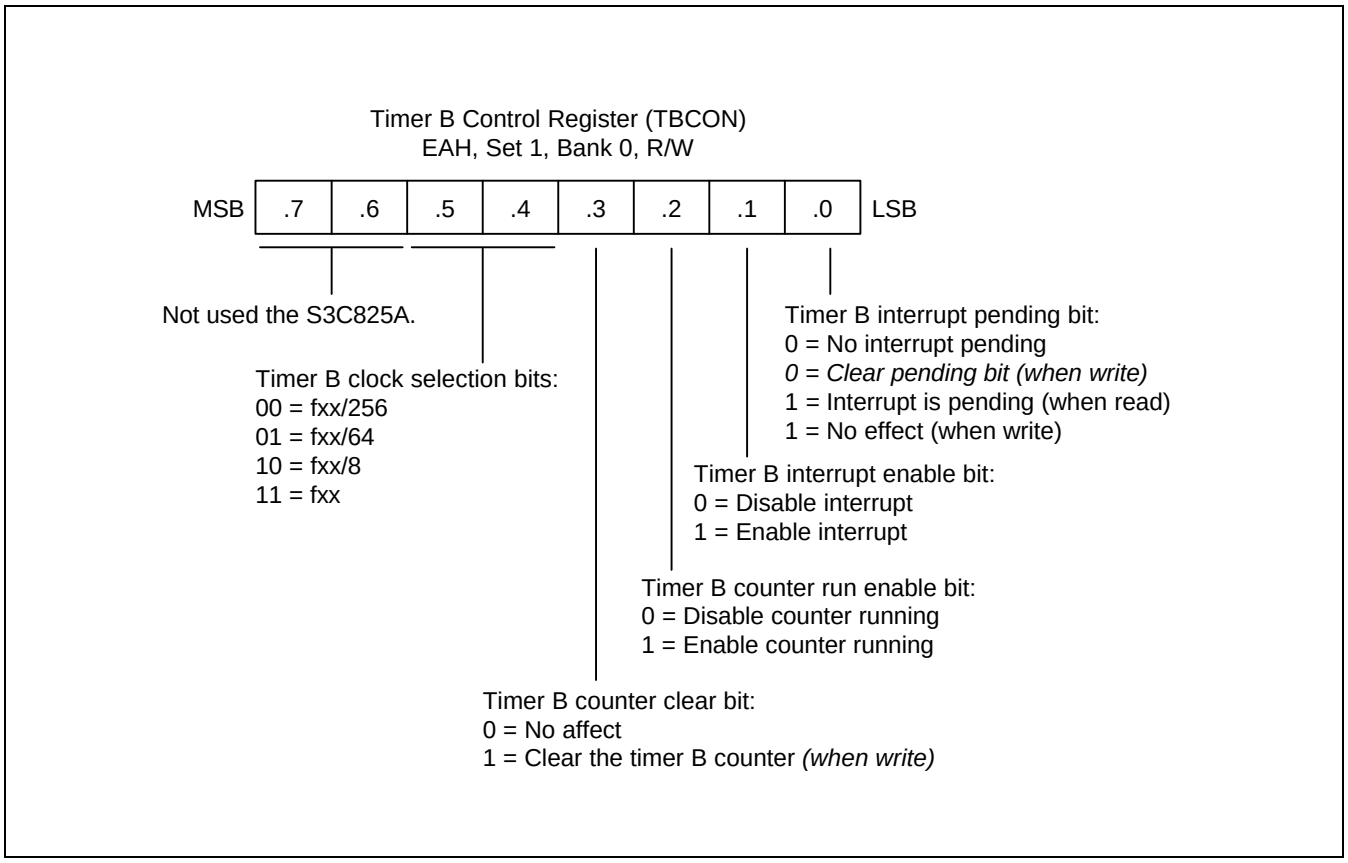
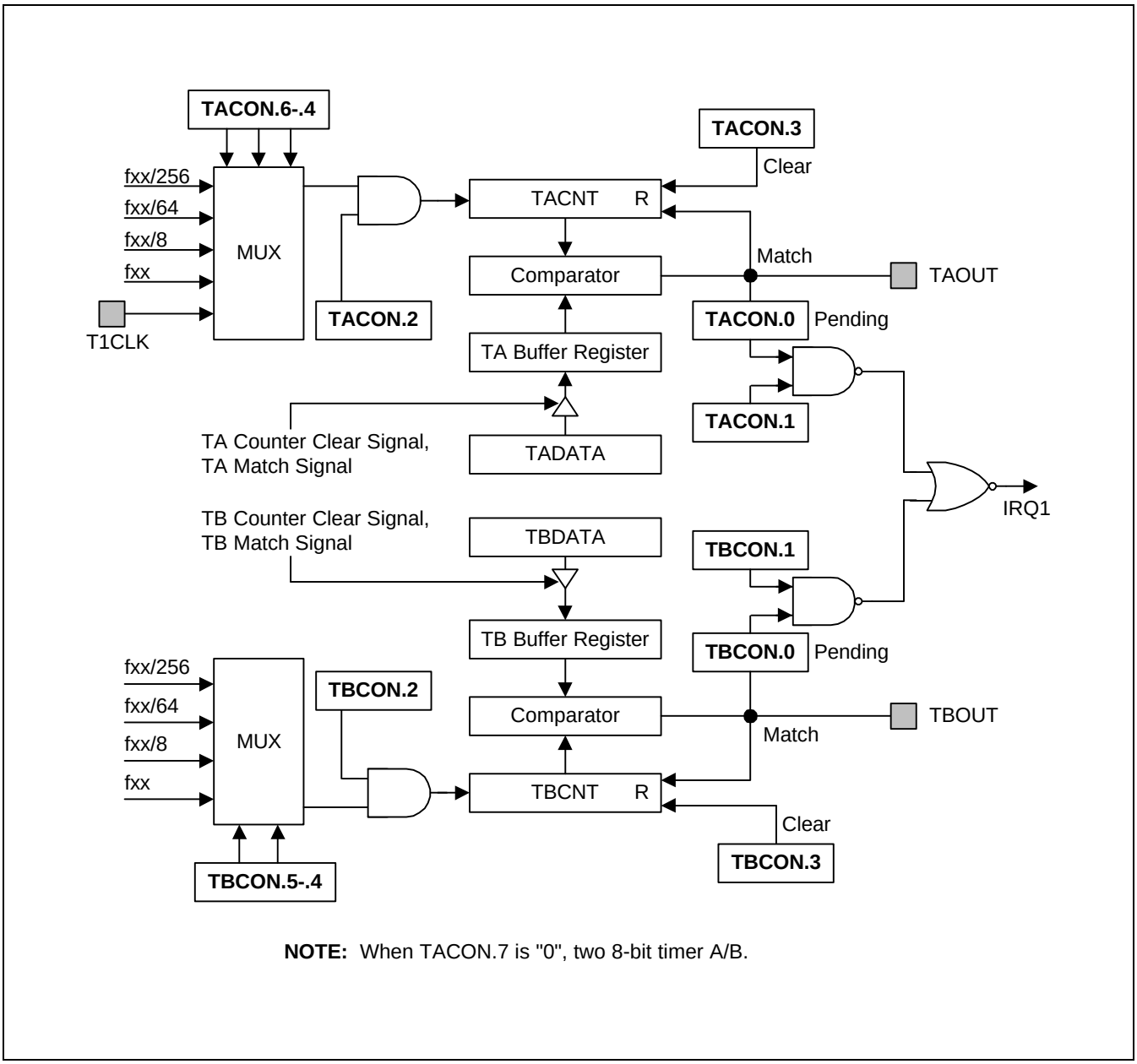


Figure 11-4. Timer B Control Register (TBCON)



NOTE: When TACON.7 is "0", two 8-bit timer A/B.

Figure 11-5. Timer A and B Function Block Diagram

NOTES

12 8-BIT TIMER 2

OVERVIEW

The 8-bit timer 2 is an 8-bit general-purpose timer. Timer 2 has the interval timer mode by using the appropriate T2CON setting.

Timer 2 has the following functional components:

— Clock frequency divider (fxx divided by 256, 64, 8 or 1) with multiplexer

— External clock input (P2.2/T2CLK)

— 8-bit counter (T2CNT), 8-bit comparator, and 8-bit reference data register (T2DATA)

— Timer 2 interrupt (IRQ2, vector E8H) generation

— Timer 2 control register, T2CON (set 1, Bank 0, EEH, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

The timer 2 can generate an interrupt, the timer 2 match interrupt (T2INT). T2INT belongs to interrupt level IRQ2, and is assigned the separate vector address, E8H.

The T2INT pending condition should be cleared by software when it has been serviced. Even though T2INT is disabled, the application's service routine can detect a pending condition of T2INT by the software and execute it's sub-routine. When this case is used, the T2INT pending bit must be cleared by the application subroutine by writing a "0" to the T2CON.0 pending bit.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the Timer 2 reference data registers, T2DATA. The match signal generates a timer 2 match interrupt (T2INT, vector E8H) and clears the counter.

If, for example, you write the value 10H to T2DATA and 0EH to T2CON, the counter will increment until it reaches 10H. At this point, the Timer 2 interrupt request is generated, the counter value is reset, and counting resumes.

TIMER 2 CONTROL REGISTER (T2CON)

You use the timer 2 control register, T2CON, to

— Enable the timer 2 operating (interval timer)

— Select the timer 2 input clock frequency

— Clear the timer 2 counter, T2CNT

— Enable the timer 2 interrupt and clear timer 2 interrupt pending condition

T2CON is located in set 1, bank 0, at address EEH, and is read/write addressable using register addressing mode.

A reset clears T2CON to "00H". This sets timer 2 to disable interval timer mode, and disables timer 2 interrupt. You can clear the timer 2 counter at any time during normal operation by writing a “1” to T2CON.3

To enable the timer 2 interrupt (IRQ2, vector E8H), you must write T2CON.2, and T2CON.1 to "1". To detect an interrupt pending condition when T2INT is disabled, the application program polls pending bit, T2CON.0. When a "1" is detected, a timer 2 interrupt is pending. When the T2INT sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 2 interrupt pending bit, T2CON.0.

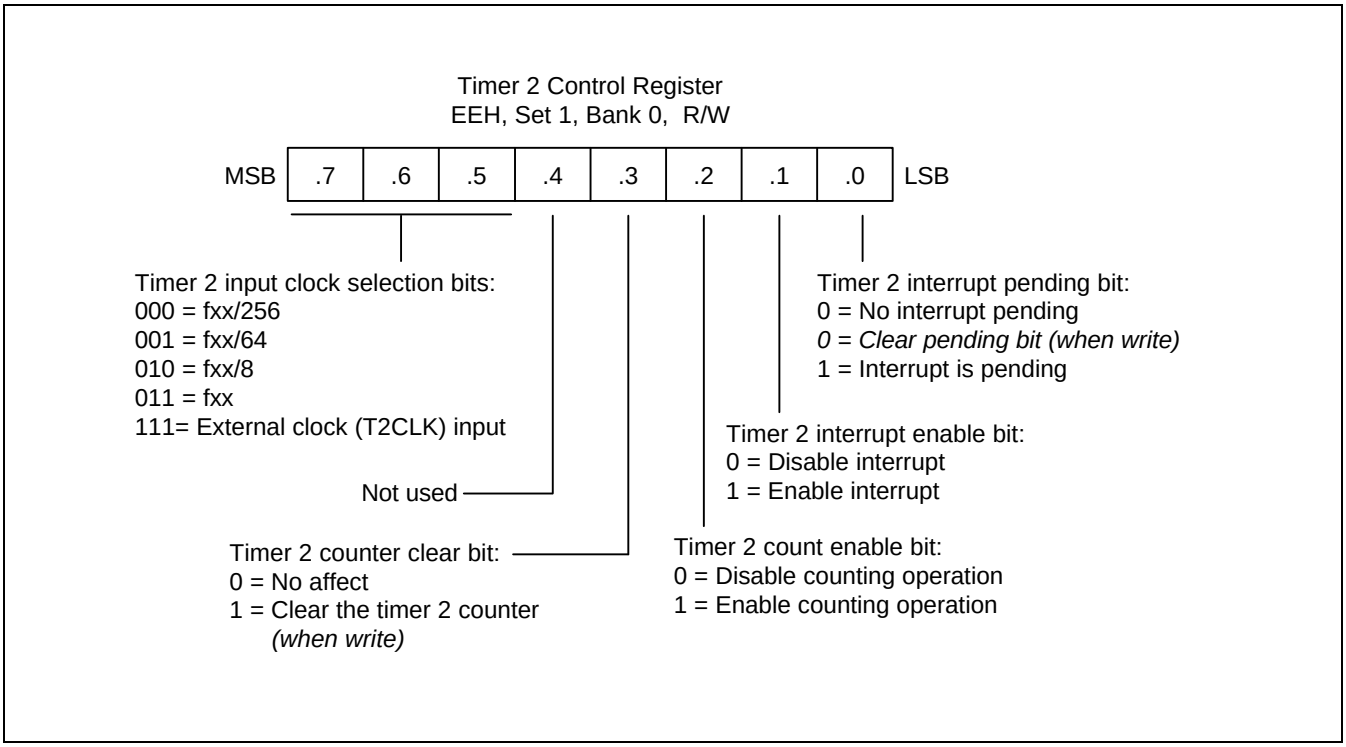


Figure 12-1. Timer 2 Control Register (T2CON)

BLOCK DIAGRAM

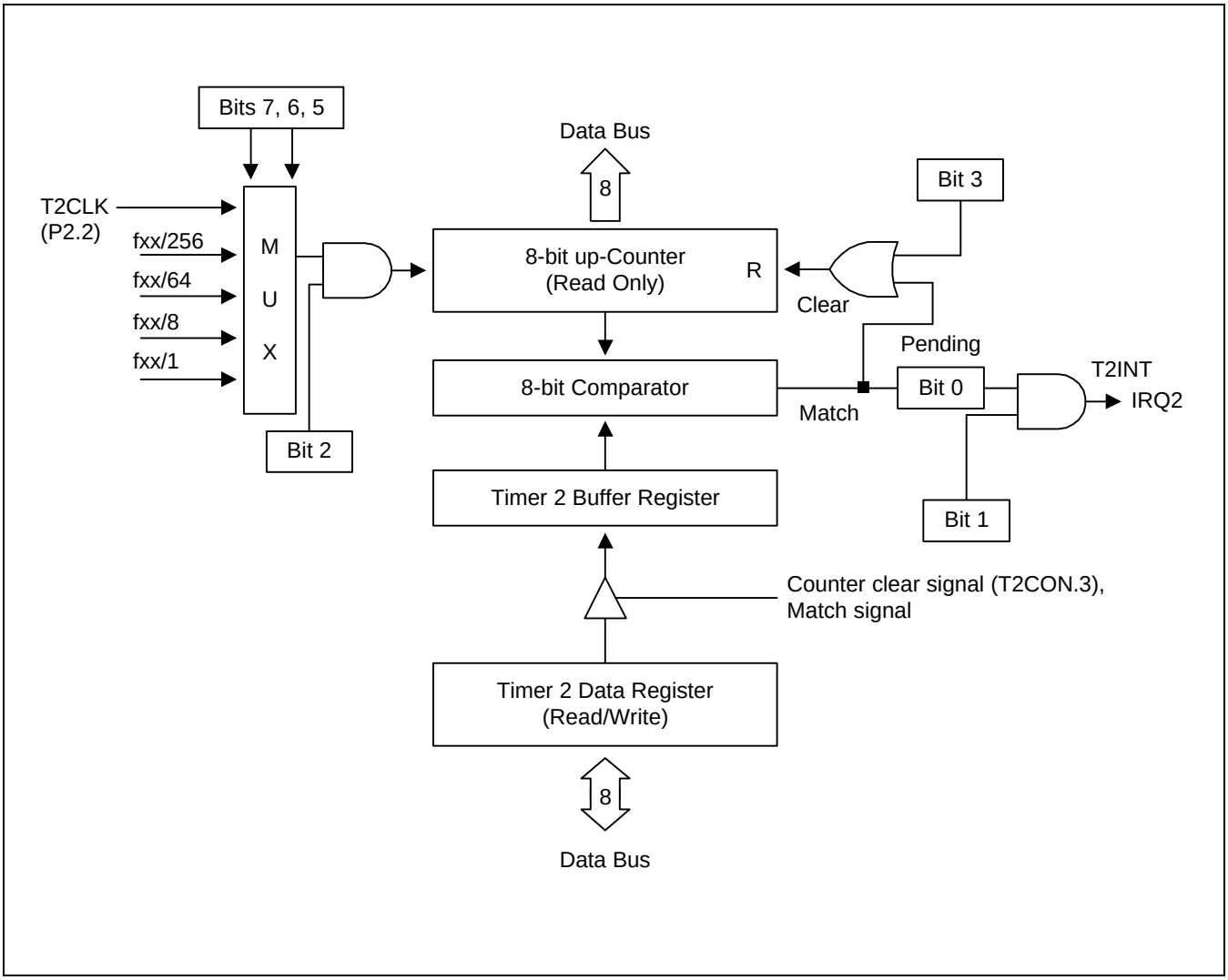


Figure 12-2. Timer 2 Functional Block Diagram

NOTES

13 16-BIT TIMER 3

OVERVIEW

Timer/counter 3 has three operating modes, one of which you select using the appropriate T3CON setting:

— Interval timer mode
— Capture input mode with a rising or falling edge trigger at the P3.6 pin
— PWM mode

Timer/counter 3 has the following functional components:

— Clock frequency divider (fxx divided by 1024, 256, 64, 8, or 1) with multiplexer
— External clock input (P3.5, T3CLK)
— 16-bit counter(T3CNTH,T3CNTL), 16-bit comparator, and 16-bit reference data register(T3DATAH,T3DATAL)
— I/O pins for capture input, match output, or PWM output (P3.6/T3CAP, P3.6/T3OUT, P3.6/T3PWM)
— Timer 3 overflow interrupt (IRQ2, vector ECH) and match/capture interrupt (IRQ2, vector EAH) generation
— Timer 3 control register, T3CON (set 1, F8H, bank 0, read/write)

TIMER/COUNTER 3 CONTROL REGISTER (T3CON)

You use the timer 3 control register, T3CON, to

— Select the timer 3 operating mode (interval timer, capture mode, or PWM mode)
— Select the timer 3 input clock frequency
— Clear the timer 3 counter, T3CNTH/T3CNTL
— Enable the timer 3 overflow interrupt or timer 3 match/capture interrupt
— Clear timer 3 match/capture interrupt pending condition

T3CON is located in set 1, bank 0, at address F8H, and is read/write addressable using Register addressing mode.

A reset clears T3CON to “00H”. This sets timer 3 to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all timer 3 interrupts. You can clear the timer 3 counter at any time during normal operation by writing a "1" to T3CON.2.

The timer 3 overflow interrupt (T3OVF) is interrupt level IRQ2 and has the vector address ECH. When a timer 3 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the timer 3 match/capture interrupt (IRQ2, vector EAH), you must write T3CON.1 to "1". To detect a match/capture interrupt pending condition, the application program polls INTPND.3. When a "1" is detected, a timer 3 match or capture interrupt is pending. When the interrupt request has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 3 match/capture interrupt pending bit, INTPND.3.

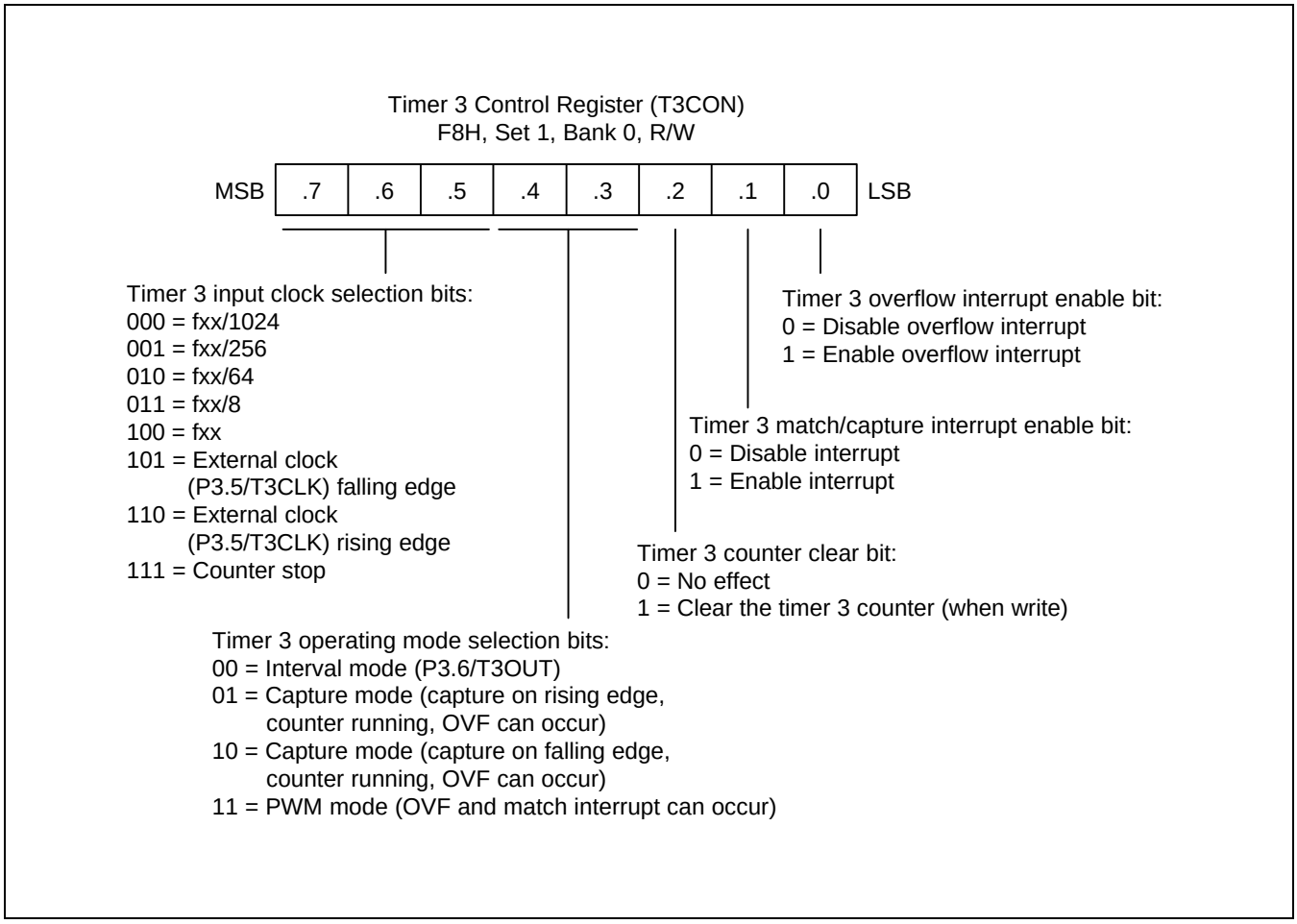


Figure 13-1. Timer 3 Control Register (T3CON)

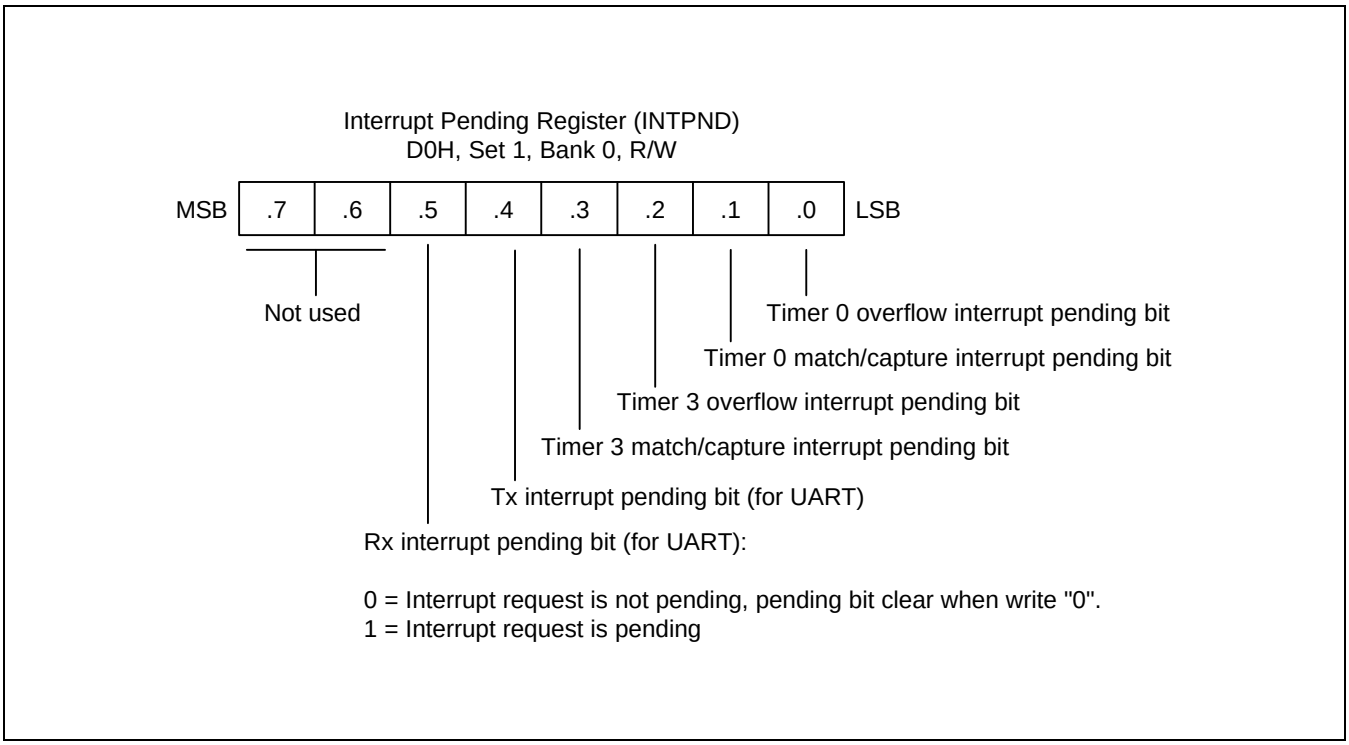


Figure 13-2. Interrupt Pending Register (INTPND)

TIMER 3 FUNCTION DESCRIPTION

Timer 3 Interrupts (IRQ2, Vectors EAH and ECH)

The timer 3 can generate two interrupts: the timer 3 overflow interrupt (T3OVF), and the timer 3 match/ capture interrupt (T3INT). T3OVF is belongs to interrupt level IRQ2, vector ECH. T3INT also belongs to interrupt level IRQ2, but is assigned the separate vector address, EAH.

A timer 3 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced or should be cleared by software in the interrupt service routine by writing a “0” to the INTPND.2 interrupt pending bit. However, the timer 3 match/capture interrupt pending condition must be cleared by the application’s interrupt service routine by writing a "0" to the INTPND.3 interrupt pending bit.

Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 3 reference data register, T3DATAH/T3DATAL. The match signal generates a timer 3 match interrupt (T3INT, vector EAH) and clears the counter.

If, for example, you write the value "1087H" to T3DATAH/T3DATAL, the counter will increment until it reaches “1087H”. At this point, the timer 3 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer 3 output pin is inverted (see Figure 13-3).

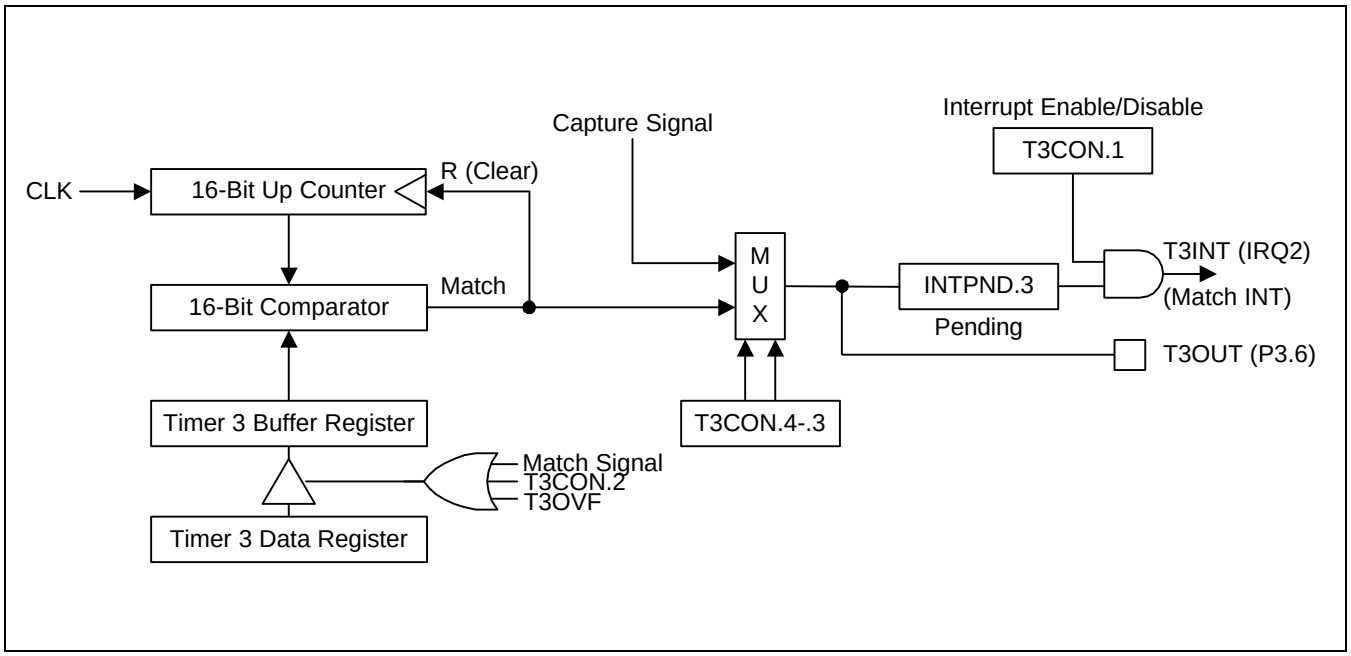


Figure 13-3. Simplified Timer 3 Function Diagram: Interval Timer Mode

Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the T3PWM (P3.6) pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 3 data register. In PWM mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at "FFFFH", and then continues incrementing from "0000H".

Although you can use the match signal to generate a timer 3 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the T3PWM (P3.6) pin is held to Low level as long as the reference data value is *less than or equal to* (≤) the counter value and then the pulse is held to High level for as long as the data value is *greater than* (>) the counter value. One pulse width is equal to $t_{CLK} \times 65536$ (see Figure 13-4).

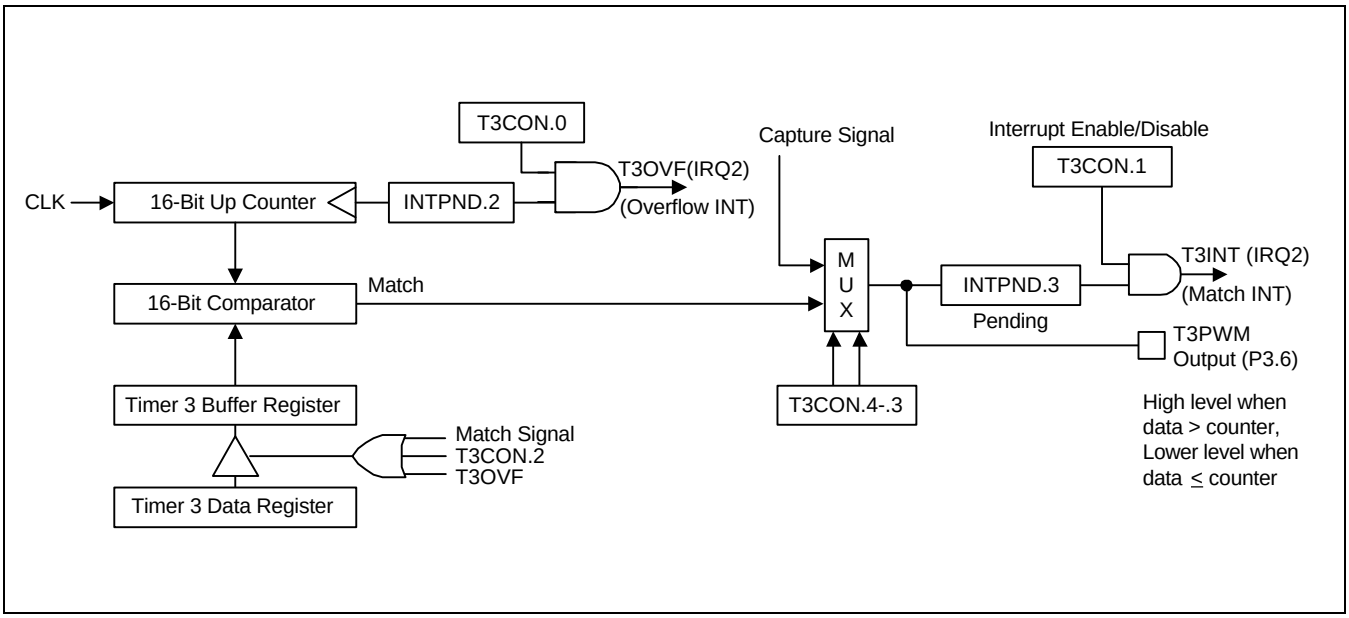


Figure 13-4. Simplified Timer 3 Function Diagram: PWM Mode

Capture Mode

In capture mode, a signal edge that is detected at the T3CAP (P3.6) pin opens a gate and loads the current counter value into the timer 3 data register. You can select rising or falling edges to trigger this operation.

Timer 3 also gives you capture input source: the signal edge at the T3CAP (P3.6) pin. You select the capture input by setting the values of the timer 3 capture input selection bits in the port 3 control register, P3CONH.5–.4, (set 1, bank 1, E4H). When P3CONH.5–.4 is "00", the T3CAP input is selected.

Both kinds of timer 3 interrupts can be used in capture mode: the timer 3 overflow interrupt is generated whenever a counter overflow occurs; the timer 3 match/capture interrupt is generated whenever the counter value is loaded into the timer 3 data register.

By reading the captured data value in T3DATAH/T3DATAL, and assuming a specific value for the timer 3 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the T3CAP pin (see Figure 13-5).

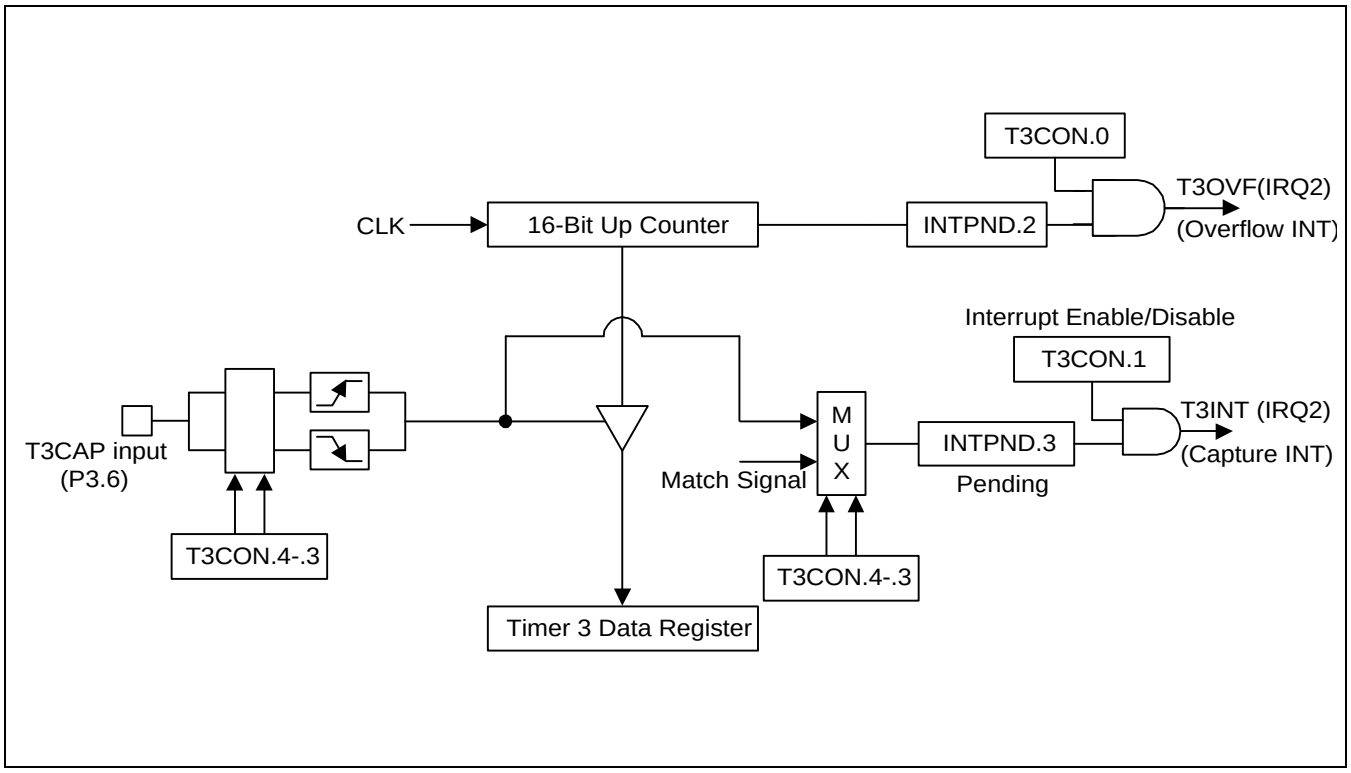


Figure 13-5. Simplified Timer 3 Function Diagram: Capture Mode

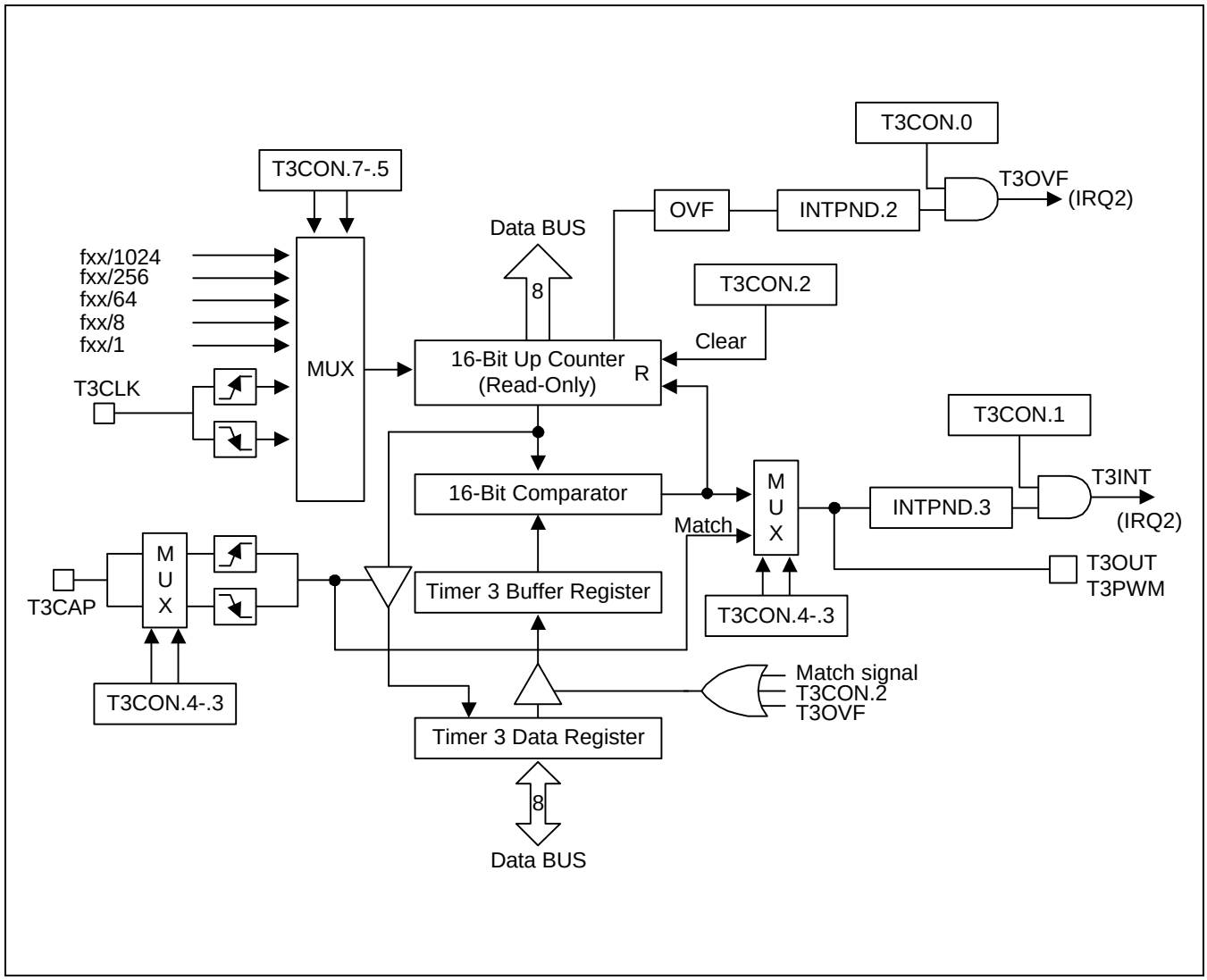


Figure 13-6. Timer 3 Block Diagram

NOTES

14 WATCH TIMER

OVERVIEW

Watch timer functions include real-time and watch-time measurement and interval timing for the system clock. To start watch timer operation, set bit 1 of the watch timer control register, WTCON.1 to "1".
And if you want to service watch timer overflow interrupt (IRQ4, vector D6H), then set the WTCON.6 to “1”.
The watch timer overflow interrupt pending condition (WTCON.0) must be cleared by software in the application’s interrupt service routine by means of writing a "0" to the WTCON.0 interrupt pending bit.
After the watch timer starts and elapses a time, the watch timer interrupt pending bit (WTCON.0) is automatically set to "1", and interrupt requests commence in 3.91 ms, 0.25, 0.5 and 1-second intervals by setting Watch timer speed selection bits (WTCON.3 – .2).

The watch timer can generate a steady 0.5 kHz, 1 kHz, 2 kHz, or 4 kHz signal to BUZ output pin for Buzzer. By setting WTCON.3 and WTCON.2 to "11b", the watch timer will function in high-speed mode, generating an interrupt every 3.91 ms. High-speed mode is useful for timing events for program debugging sequences.

The watch timer supplies the clock frequency for the LCD controller (f_{LCD}). Therefore, if the watch timer is disabled, the LCD controller does not operate.

Watch timer has the following functional components:

— Real Time and Watch-Time Measurement
— Using a Main Clock Source or Sub clock
— Clock Source Generation for LCD Controller (f_{LCD})
— I/O pin for Buzzer Output Frequency Generator (P5.3, BUZ)
— Timing Tests in High-Speed Mode
— Watch timer overflow interrupt (IRQ4, vector D6H) generation
— Watch timer control register, WTCON (set 1, bank 1, EFH, read/write)

WATCH TIMER CONTROL REGISTER (WTCON)

The watch timer control register, WTCON is used to select the watch timer interrupt time and Buzzer signal, to enable or disable the watch timer function. It is located in set 1, bank 1 at address EFH, and is read/write addressable using register addressing mode.
A reset clears WTCON to "00H". This disable the watch timer.
So, if you want to use the watch timer, you must write appropriate value to WTCON.

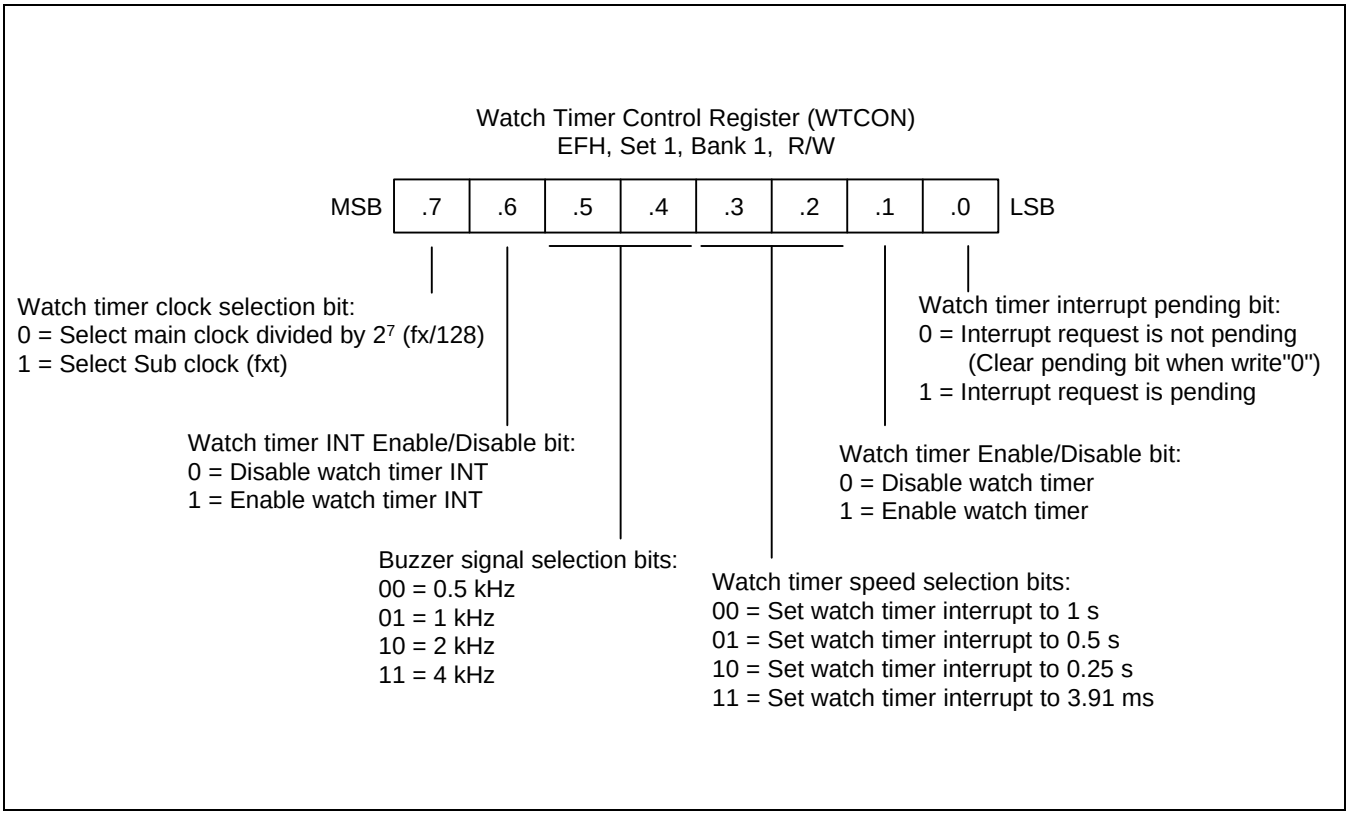


Figure 14-1. Watch Timer Control Register (WTCON)

WATCH TIMER CIRCUIT DIAGRAM

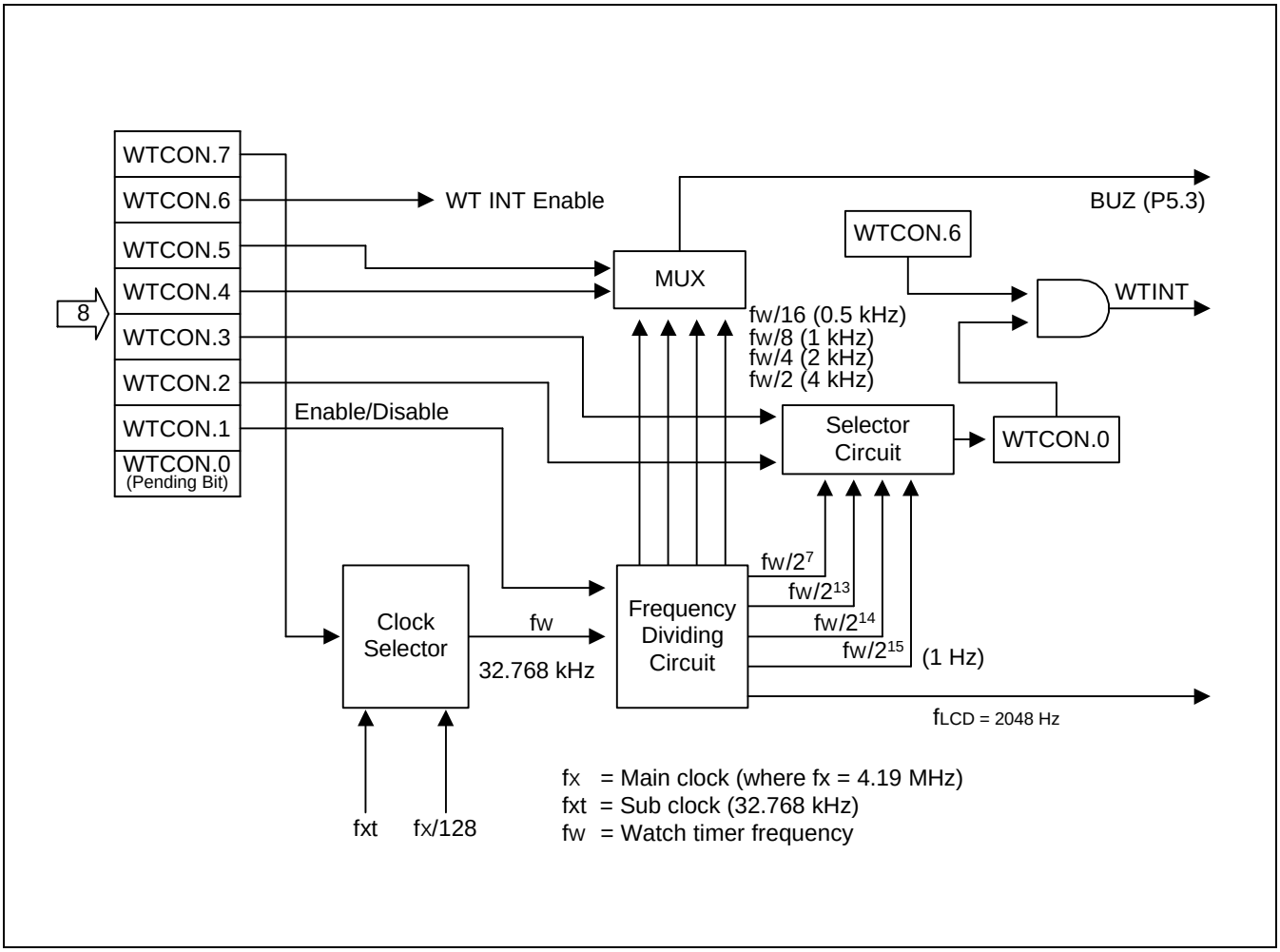


Figure 14-2. Watch Timer Circuit Diagram

NOTES

15 LCD CONTROLLER/DRIVER

OVERVIEW

The S3C825A microcontroller can directly drive an up-to-224-dot (28 segments x 8 commons) LCD panel. Its LCD block has the following components:

— LCD controller/driver
— Display RAM for storing display data
— 4 common/segment output pins (COM4/SEG0–COM7/SEG3)
— 28 segment output pins (SEG4–SEG31)
— 4 common output pins (COM0–COM3)
— Internal resistor circuit for LCD bias
— V_{LC1} pin for controlling the driver and bias voltage

The LCD control register, LCON, is used to turn the LCD display on and off, switch the current to the dividing resistors for the LCD display, and frame frequency. Data written to the LCD display RAM can be automatically transferred to the segment signal pins without any program control.

When a subsystem clock is selected as the LCD clock source, the LCD display is enabled even in the main clock stop or idle mode.

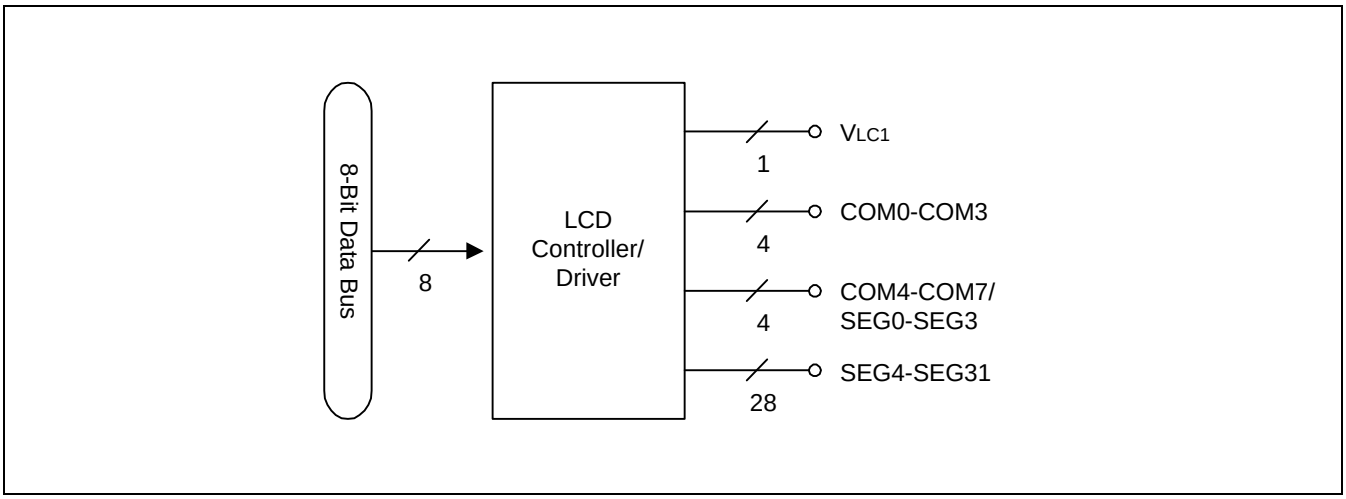


Figure 15-1. LCD Function Diagram

LCD CIRCUIT DIAGRAM

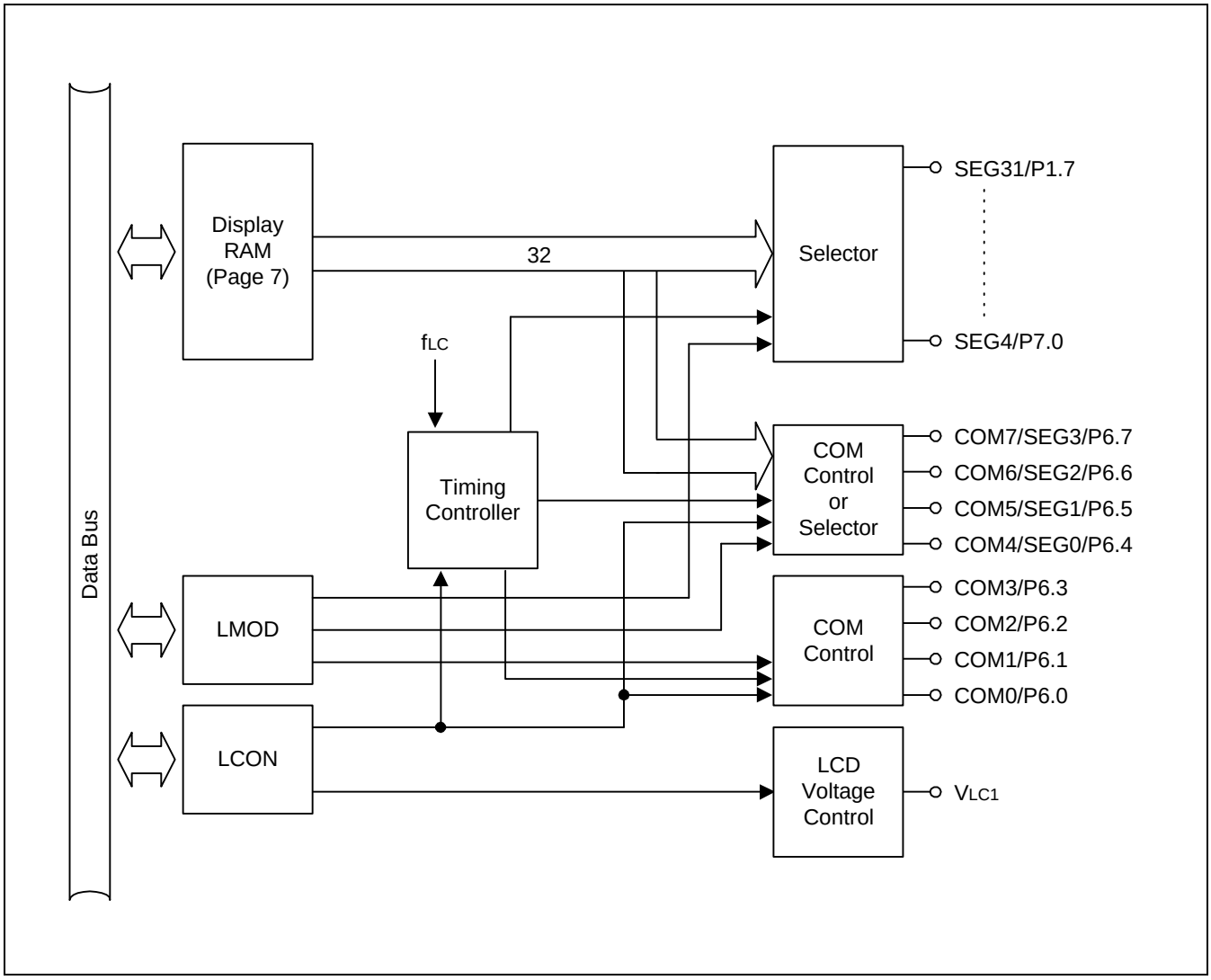


Figure 15-2. LCD Circuit Diagram

LCD RAM ADDRESS AREA

RAM addresses of page 7 are used as LCD data memory. These locations can be addressed by 1-bit or 8-bit instructions. If the bit value of a display segment is "1", the LCD display is turned on. If the bit value is "0", the display is turned off.

Display RAM data are sent out through the segment pins, SEG0–SEG31, using the direct memory access (DMA) method that is synchronized with the f_{LCD} signal. RAM addresses in this location that are not used for LCD display can be allocated to general-purpose use.

COM	Bit	SEG0	SEG1	SEG2	SEG3	SEG4	------	SEG30	SEG31
COM0 COM1 COM2 COM3 COM4 COM5 COM6 COM7	.0 .1 .2 .3 .4 .5 .6 .7	700H	701H	702H	703H	704H	------	71EH	71FH

Figure 15-3. LCD Display Data RAM Organization

LCD CONTROL REGISTER (LCON)

The LCD control register (LCON) is used to turn the LCD display on and off, LCD frame frequency, and control the flow of the current to the dividing resistors in the LCD circuit. After a RESET, all LCON values are cleared to "0". This turns the LCD display off and stops the flow of the current to the dividing resistors.

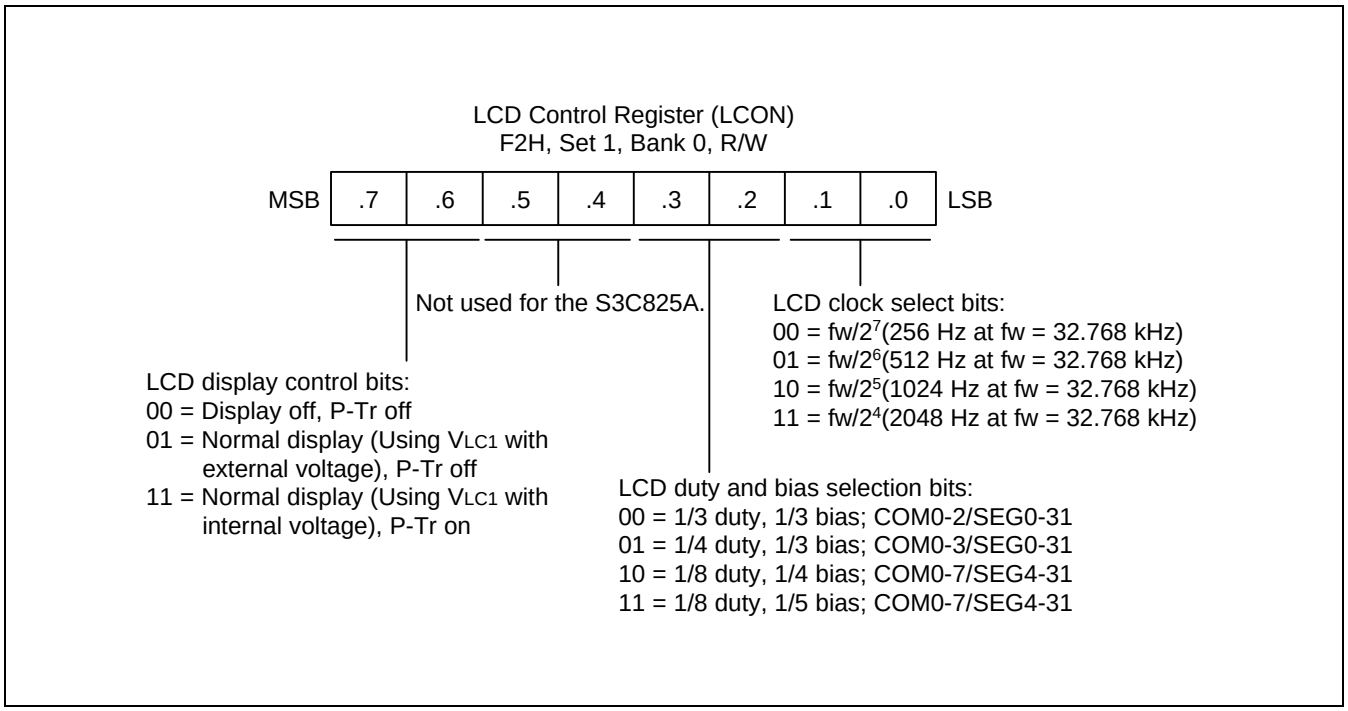


Figure 15-4. LCD Control Register (LCON)

LCD MODE CONTROL REGISTER (LMOD), F3H at BANK 0 of SET 1

Table 15-1. LCD Mode Control Register (LMOD) Organization

LCON Bit	Setting	Description
LMOD.7–.4	Not used for the S3C825A.	
LMOD.3–.0	0000	All I/O port (P0, P1, P6–P8)
	0001	Select LCD COM0–3, COM4–7/SEG0–3. P0, P1, P7, and P8 are I/O port.
	0010	Select LCD COM0–3, COM4–7/SEG0–3, SEG4–7. P0, P1, P7.4–P7.7, and P8 are I/O port.
	0011	Select LCD COM0–3, COM4–7/SEG0–3, SEG4–11. P0, P1, and P8 are I/O port.
	0100	Select LCD COM0–3, COM4–7/SEG0–3, SEG4–15. P0 and P1are I/O port.
	0101	Select LCD COM0–3, COM4–7/SEG0–3, SEG4–19. P0.4–P0.7 and P1 are I/O port.
	0110	Select LCD COM0–3, COM4–7/SEG0–3, SEG4–23. P1 is I/O port.
	0111	Select LCD COM0–3, COM4–7/SEG0–3, SEG4–27. P1.4–P1.7 is I/O port.
	1000	Select LCD COM0–3, COM4–7/SEG0–3, SEG4–31.

NOTE: The COM4–7/SEG0–3 signals are controlled by LCON.3–.2.

LCD VOLTAGE DIVIDING RESISTORS

On-chip voltage dividing resistors for the LCD drive power supply are fixed to the V_{LC1}–V_{LC5} pins. Figure 15-5 shows the bias connections for the S3C825A LCD drive power supply. To cut off the flow of current through the dividing resistor, manipulate bits 7 and 6 of the LCON register.

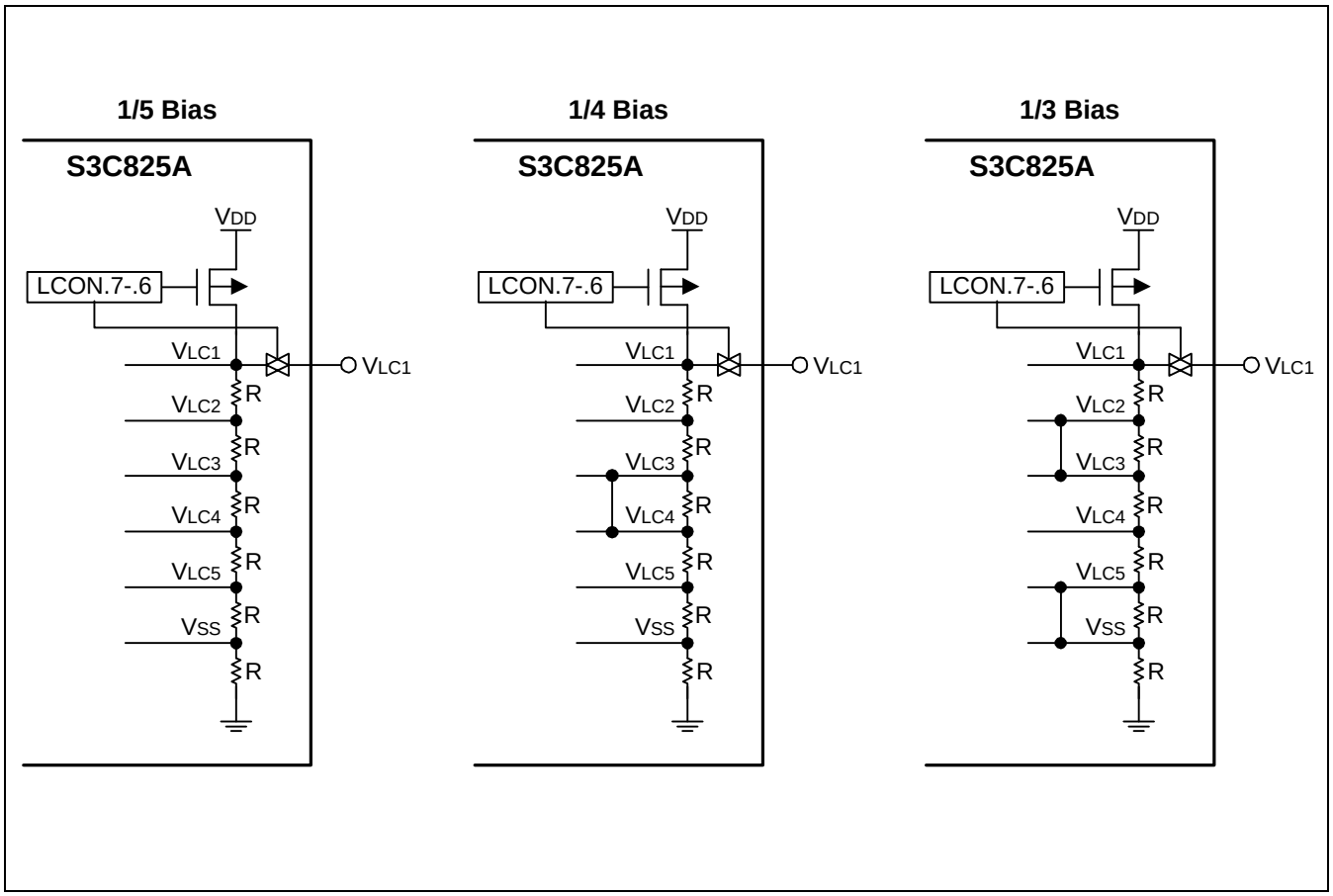
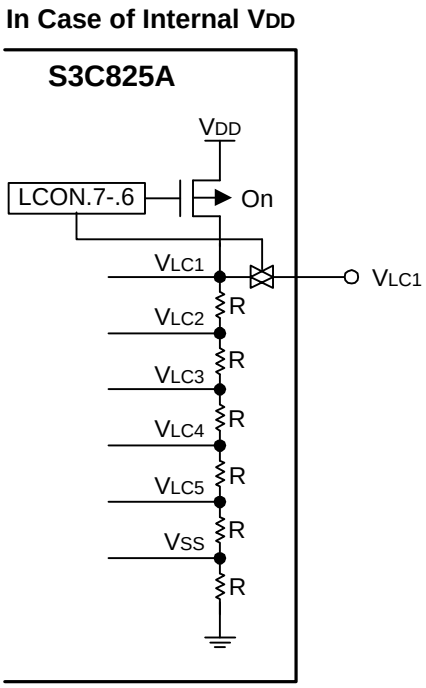
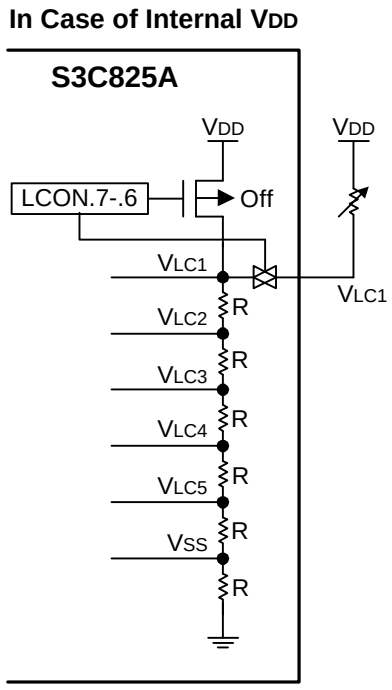


Figure 15-5. LCD Bias Circuit Connection



NOTE: For power-saving and LCD-off, LCON.7-.6 must be set to 00B.
For a normal display, LCON.7-.6 must be set to 11B.

Figure 15-6. Example 1 for the Usage of LCON.7–.6



NOTE: For power-saving and LCD-off, LCON.7-.6 must be set to 00B.
For a normal display, LCON.7-.6 must be set to 01B.

Figure 15-7. Example 2 for the Usage of LCON.7–.6

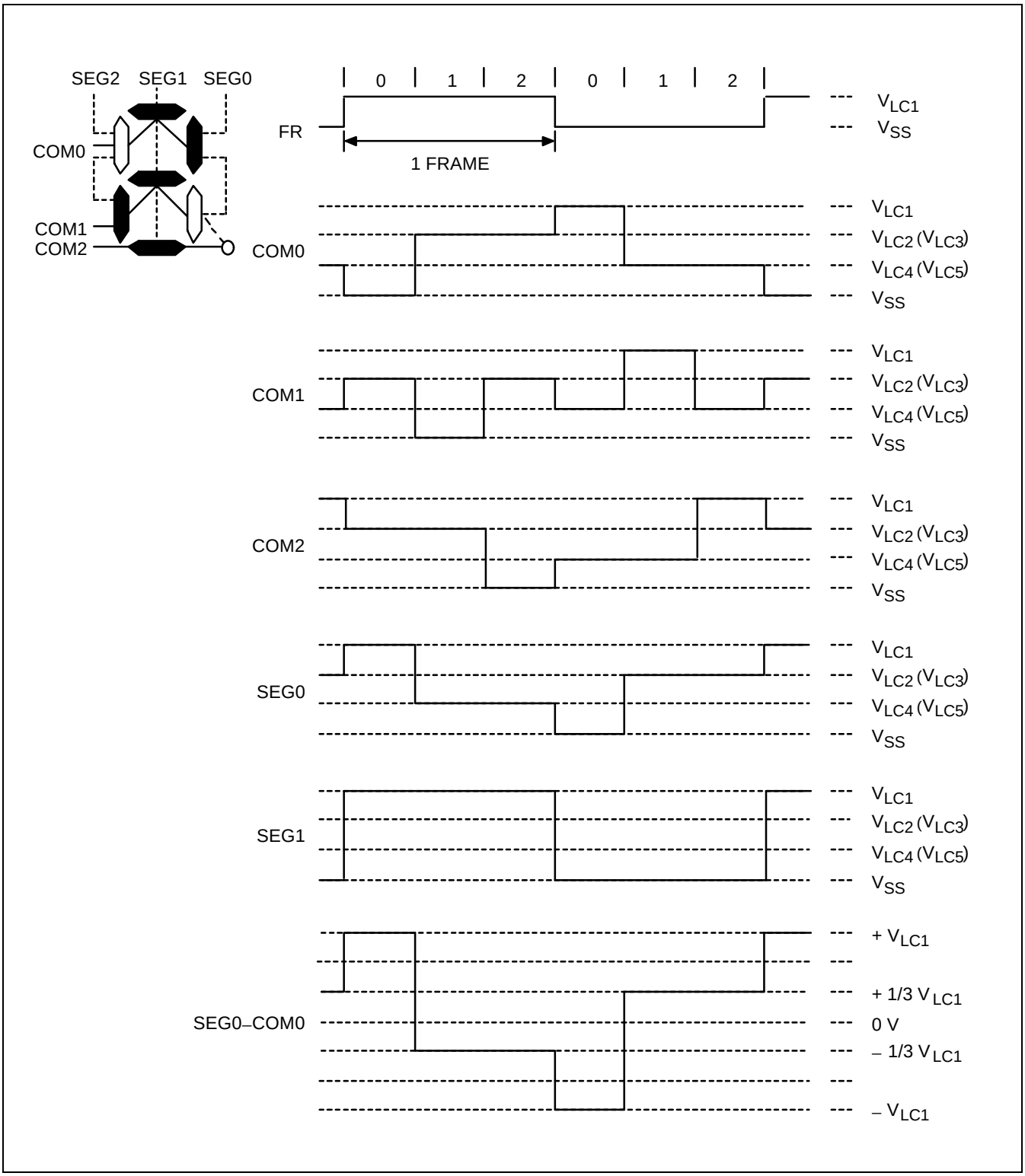


Figure 15-8. LCD Signal Waveforms (1/3 Duty, 1/3 Bias)

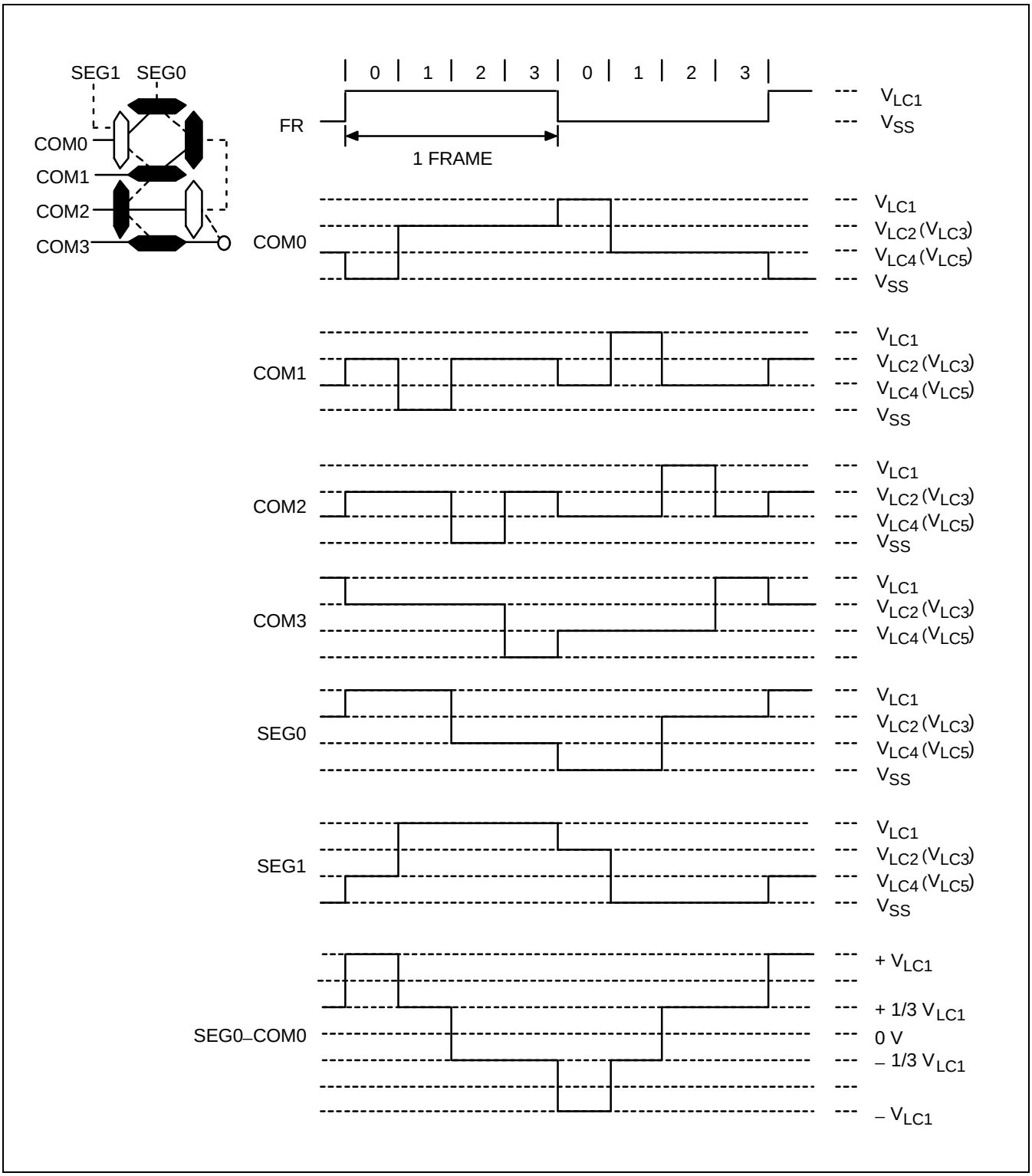


Figure 15-9. LCD Signal Waveforms (1/4 Duty, 1/3 Bias)

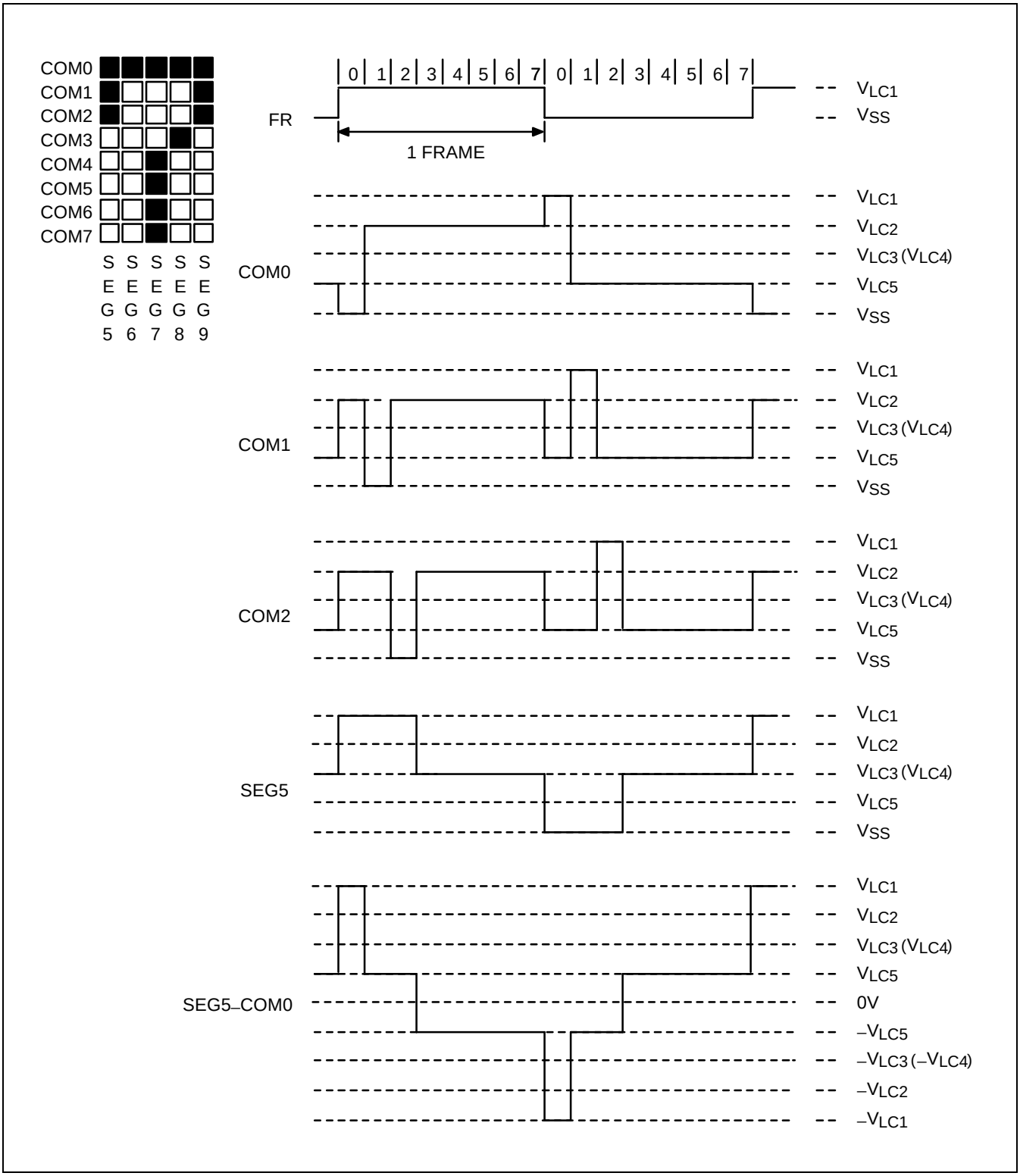


Figure 15-10. LCD Signal Waveforms (1/8 Duty, 1/4 Bias)

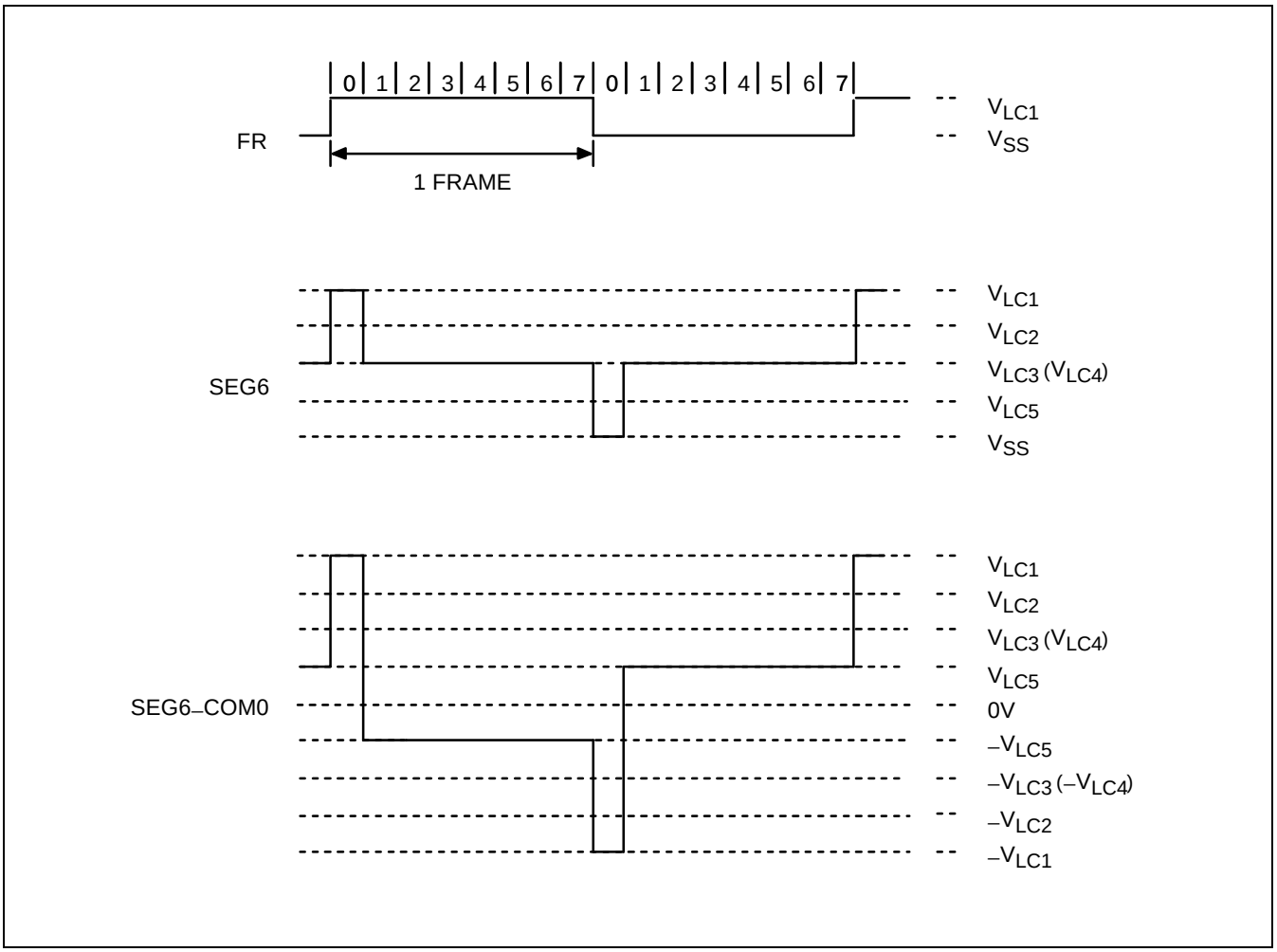


Figure 15-10. LCD Signal Waveforms (1/8 Duty, 1/4 Bias) (Continued)

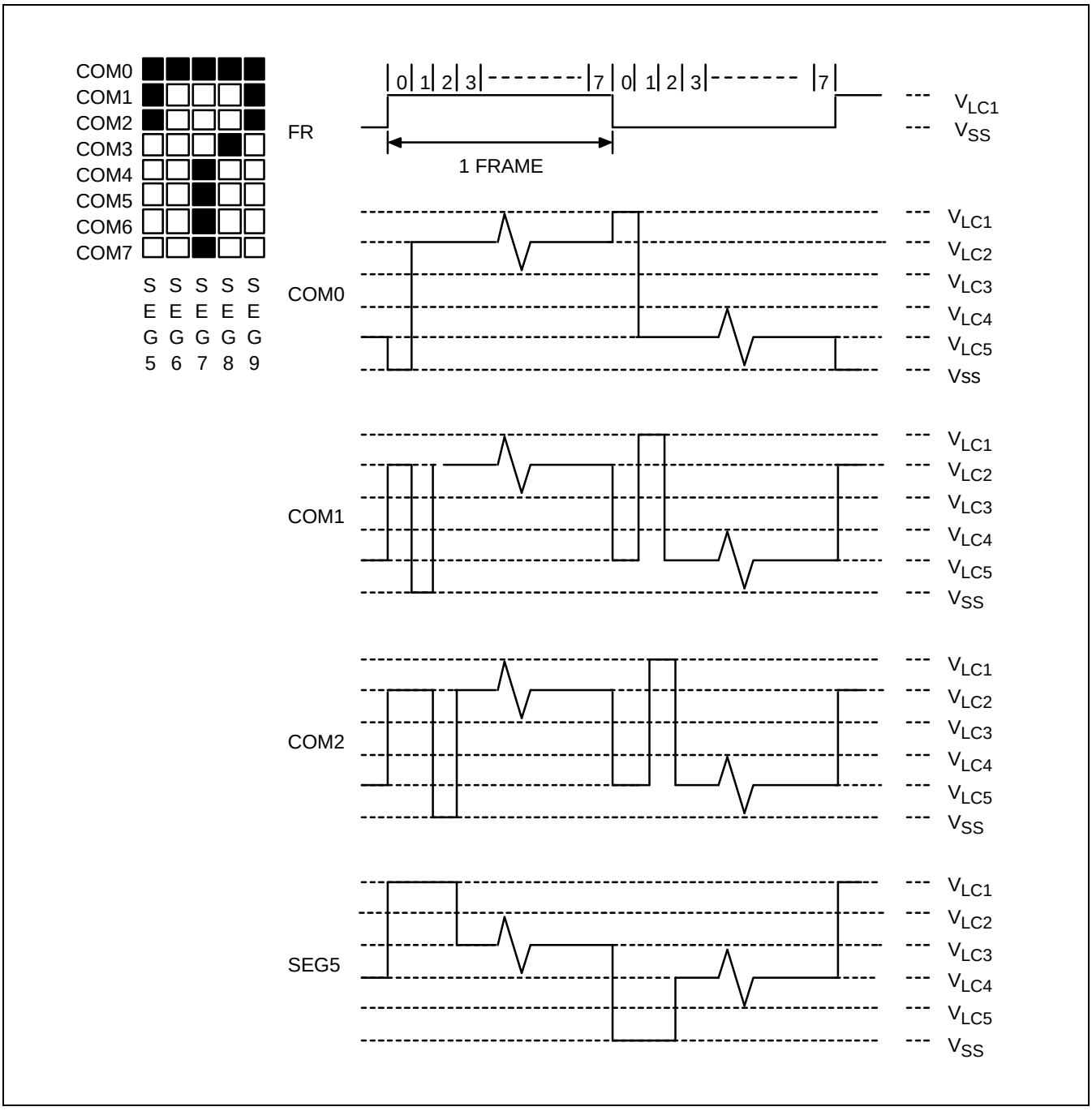


Figure 15-11. LCD Signal Waveforms (1/8 Duty, 1/5 Bias)

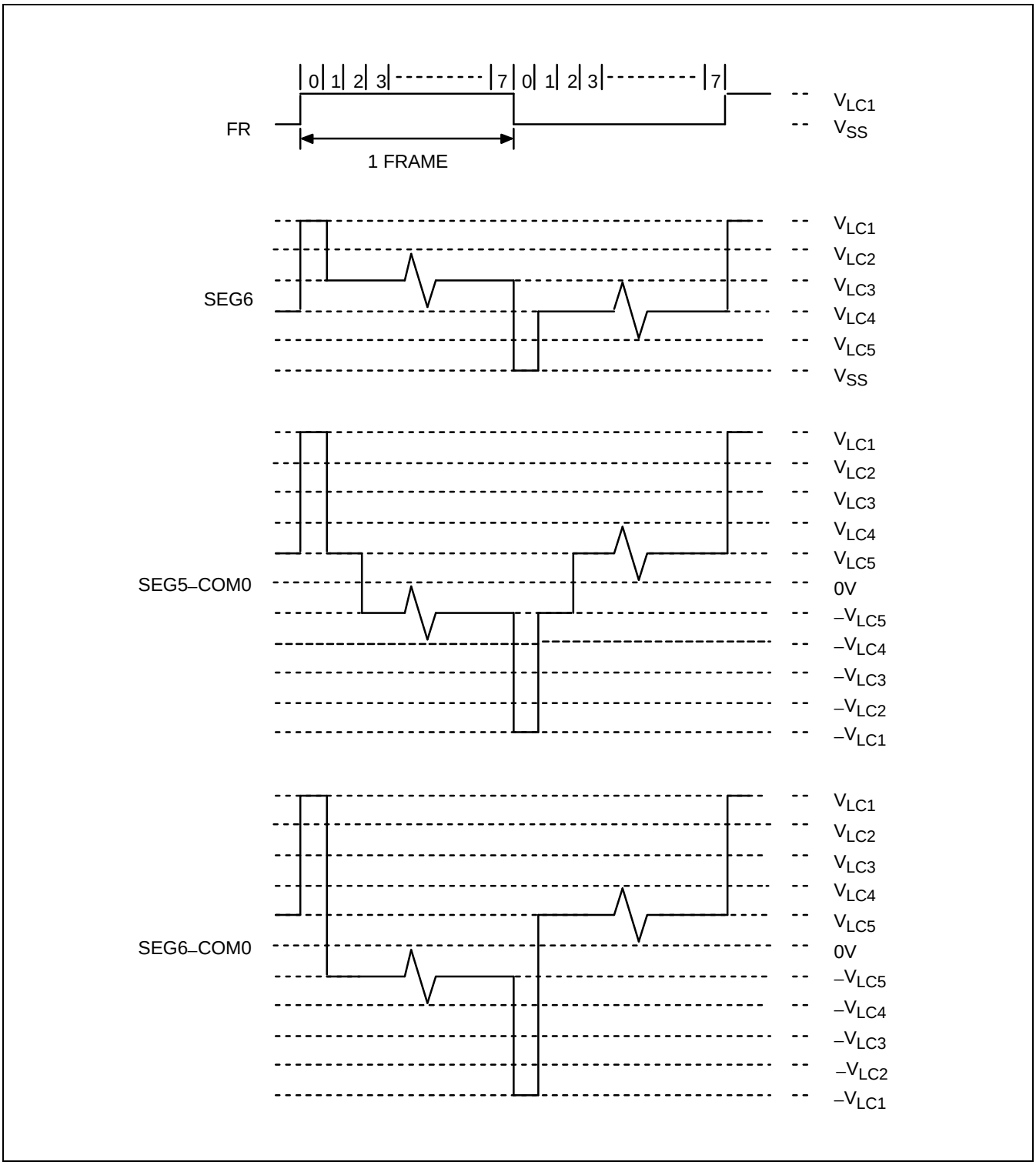


Figure 15-11. LCD Signal Waveforms (1/8 Duty, 1/5 Bias) (Continued)

16 10-BIT ANALOG-TO-DIGITAL CONVERTER

OVERVIEW

The 10-bit A/D converter (ADC) module uses successive approximation logic to convert analog levels entering at one of the four input channels to equivalent 10-bit digital values. The analog input level must lie between the AV_{REF} and AV_{SS} values. The A/D converter has the following components:

— Analog comparator with successive approximation logic

— D/A converter logic (resistor string type)

— ADC control register (ADCON)

— Four multiplexed analog data input pins (AD0–AD3)

— 10-bit A/D conversion data output register (ADDATAH/ADDATAL)

— 4-bit digital input port (Alternately, I/O port)

FUNCTION DESCRIPTION

To initiate an analog-to-digital conversion procedure, at first you must set with alternative function for ADC input enable at port 3, the pin set with alternative function can be used for ADC analog input. And you write the channel selection data in the A/D converter control register ADCON.4–.5 to select one of the four analog input pins (AD0–3) and set the conversion start or enable bit, ADCON.0. The read-write ADCON register is located in set 1, bank 0, at address EFH. The pins witch are not used for ADC can be used for normal I/O.

During a normal conversion, ADC logic initially sets the successive approximation register to 800H (the approximate half-way point of an 10-bit register). This register is then updated automatically during each conversion step. The successive approximation block performs 10-bit conversions for one input channel at a time. You can dynamically select different channels by manipulating the channel selection bit value (ADCON.5–4) in the ADCON register. To start the A/D conversion, you should set the enable bit, ADCON.0. When a conversion is completed, ADCON.3, the end-of-conversion(EOC) bit is automatically set to 1 and the result is dumped into the ADDATAH/ADDATAL register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATAH/ADDATAL before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

NOTE

Because the A/D converter has no sample-and-hold circuitry, it is very important that fluctuation in the analog level at the AD0–AD3 input pins during a conversion procedure be kept to an absolute minimum. Any change in the input level, perhaps due to noise, will invalidate the result. If the chip enters to STOP or IDLE mode in conversion process, there will be a leakage current path in A/D block. You must use STOP or IDLE mode after ADC operation is finished.

CONVERSION TIMING

The A/D conversion process requires 4 steps (4 clock edges) to convert each bit and 10 clocks to set-up A/D conversion. Therefore, total of 50 clocks are required to complete an 10-bit conversion: When fxx/8 is selected for conversion clock with an 4.5 MHz fxx clock frequency, one clock cycle is 1.78 us. Each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

4 clocks/bit × 10-bit + set-up time = 50 clocks, 50 clock × 1.78 us = 89 us at 0.56 MHz (4.5 MHz/8)

Note that A/D converter needs at least 25μs for conversion time.

A/D CONVERTER CONTROL REGISTER (ADCON)

The A/D converter control register, ADCON, is located at address EFH in set 1, bank 0. It has three functions:

— Analog input pin selection (bits 4 and 5)

— End-of-conversion status detection (bit 3)

— ADC clock selection (bits 2 and 1)

— A/D operation start or enable (bit 0)

After a reset, the start bit is turned off. You can select only one analog input channel at a time. Other analog input pins (AD0–AD3) can be selected dynamically by manipulating the ADCON.4–5 bits. And the pins not used for analog input can be used for normal I/O function.

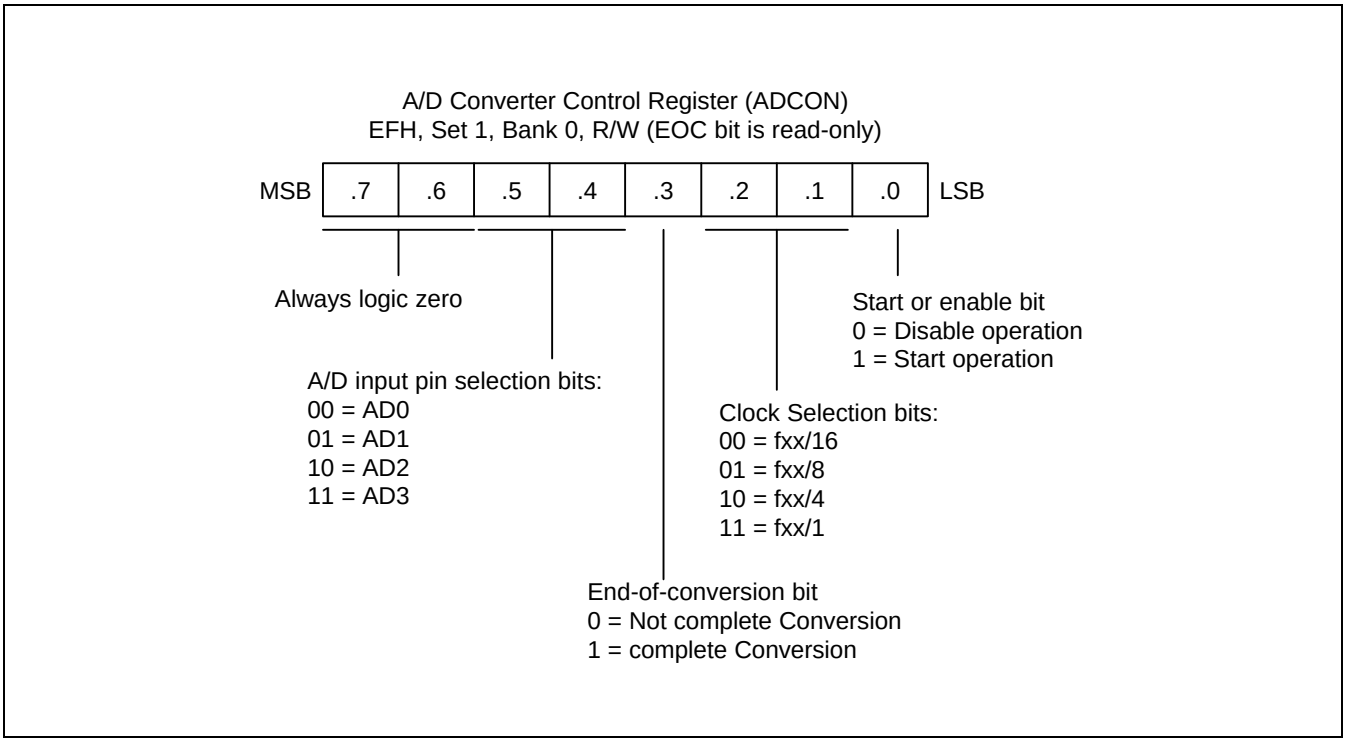


Figure 16-1. A/D Converter Control Register (ADCON)

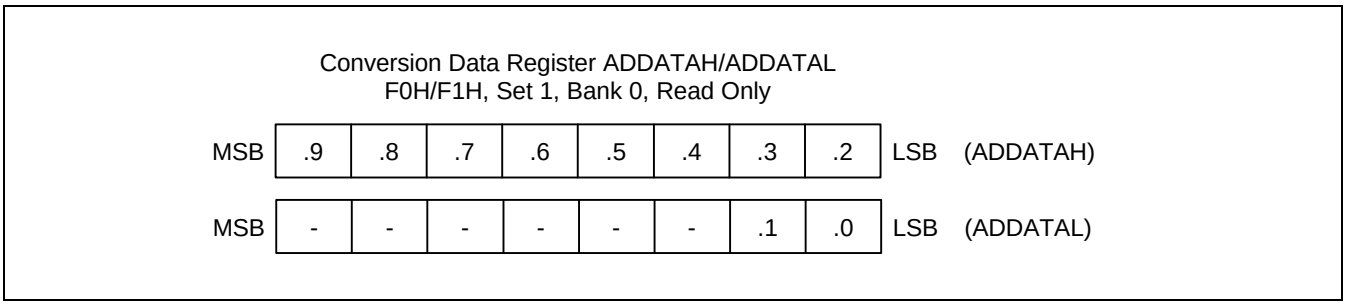


Figure 16-2. A/D Converter Data Register (ADDATAH/ADDATAL)

INTERNAL REFERENCE VOLTAGE LEVELS

In the ADC function block, the analog input voltage level is compared to the reference voltage. The analog input level must remain within the range AV_{SS} to AV_{REF}.

Different reference voltage levels are generated internally along the resistor tree during the analog conversion process for each conversion step. The reference voltage level for the first conversion bit is always 1/2 AV_{REF}.

BLOCK DIAGRAM

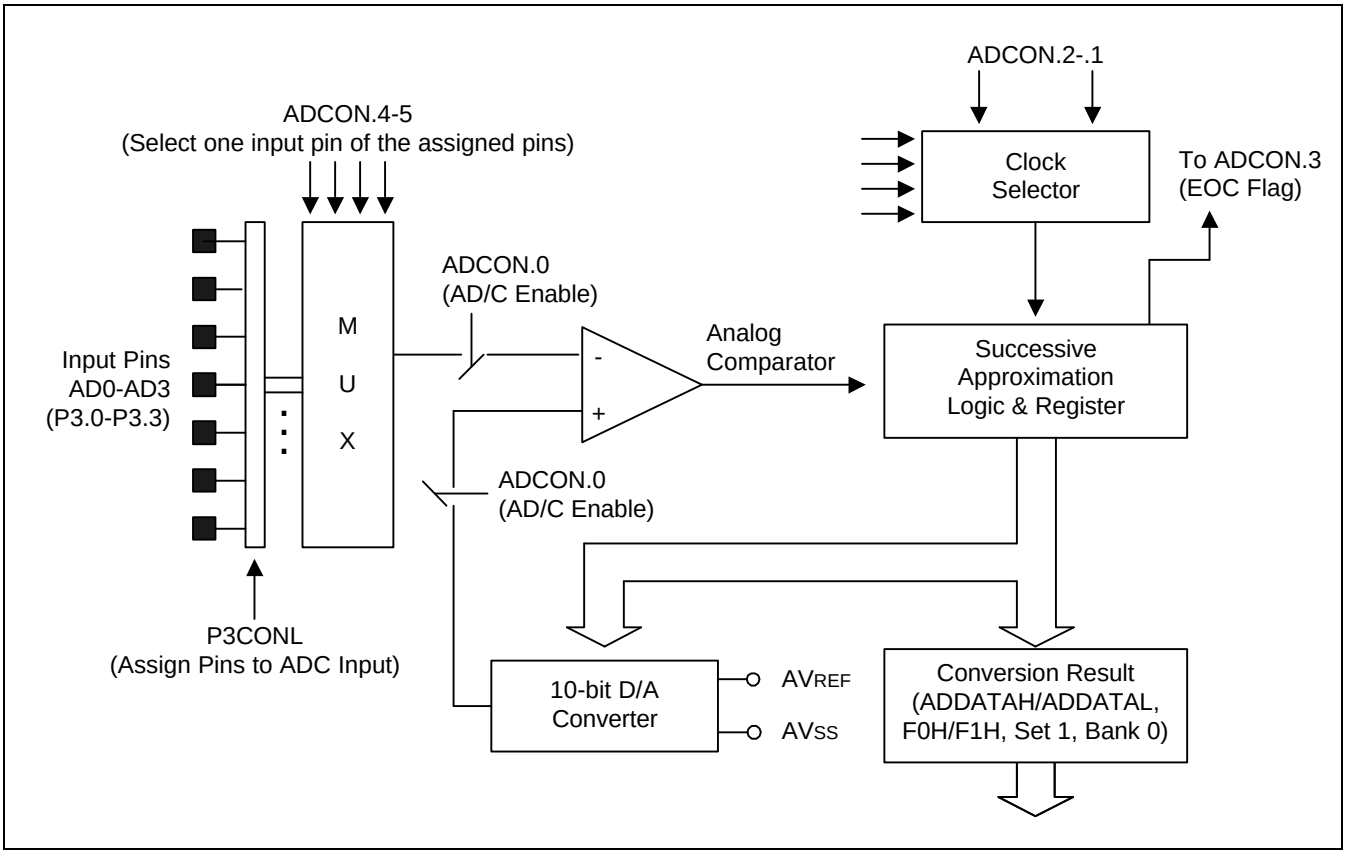


Figure 16-3. A/D Converter Functional Block Diagram

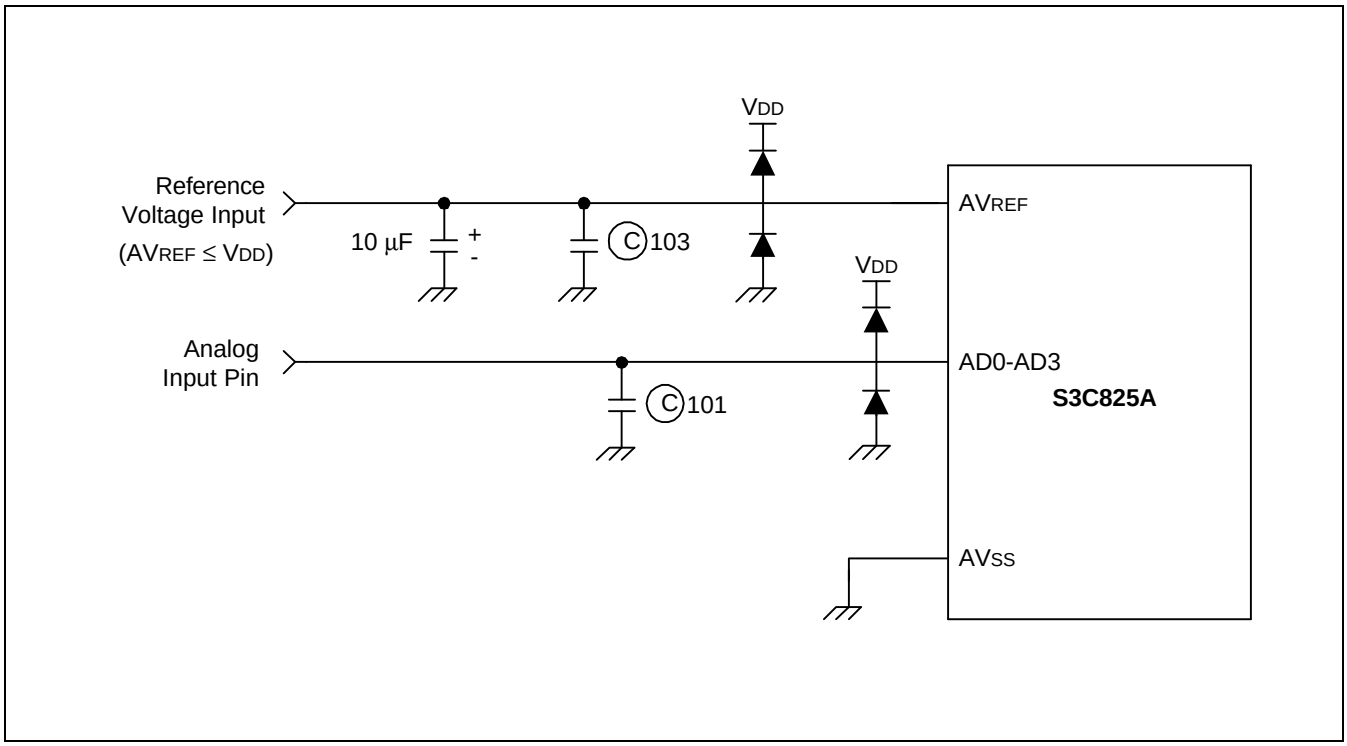


Figure 16-4. Recommended A/D Converter Circuit for Highest Absolute Accuracy

17 SERIAL I/O INTERFACE

OVERVIEW

Serial I/O modules, SIO can interface with various types of external device that require serial data transfer. The components of SIO function block are:

— 8-bit control register (SIOCON)
— Clock selector logic
— 8-bit data buffer (SIODATA)
— 8-bit prescaler (SIOPS)
— 3-bit serial clock counter
— Serial data I/O pins (SI, SO)
— External clock input/output pin (SCK)

The SIO module can transmit or receive 8-bit serial data at a frequency determined by its corresponding control register settings. To ensure flexible data transmission rates, you can select an internal or external clock source.

PROGRAMMING PROCEDURE

To program the SIO module, follow these basic steps:

1. Configure the I/O pins at port (SCK/SI/SO) by loading the appropriate value to the P5CONL register if necessary.
2. Load an 8-bit value to the SIOCON control register to properly configure the serial I/O module. In this operation, SIOCON.2 must be set to "1" to enable the data shifter.
3. For interrupt generation, set the serial I/O interrupt enable bit (SIOCON) to "1".
4. When you transmit data to the serial buffer, write data to SIODATA and set SIOCON.3 to 1, the shift operation starts.
5. When the shift operation (transmit/receive) is completed, the SIO pending bit (SIOCON.0) are set to "1" and SIO interrupt request is generated.

SIO CONTROL REGISTERS (SIOCON)

The control register for serial I/O interface module, SIOCON, is located at E0H in set 1, bank 0. It has the control setting for SIO module.

— Clock source selection (internal or external) for shift clock
— Interrupt enable
— Edge selection for shift operation
— Clear 3-bit counter and start shift operation
— Shift operation (transmit) enable
— Mode selection (transmit/receive or receive-only)
— Data direction selection (MSB first or LSB first)

A reset clears the SIOCON value to "00H". This configures the corresponding module with an internal clock source at the SCK, selects receive-only operating mode, and clears the 3-bit counter. The data shift operation and the interrupt are disabled. The selected data direction is MSB-first.

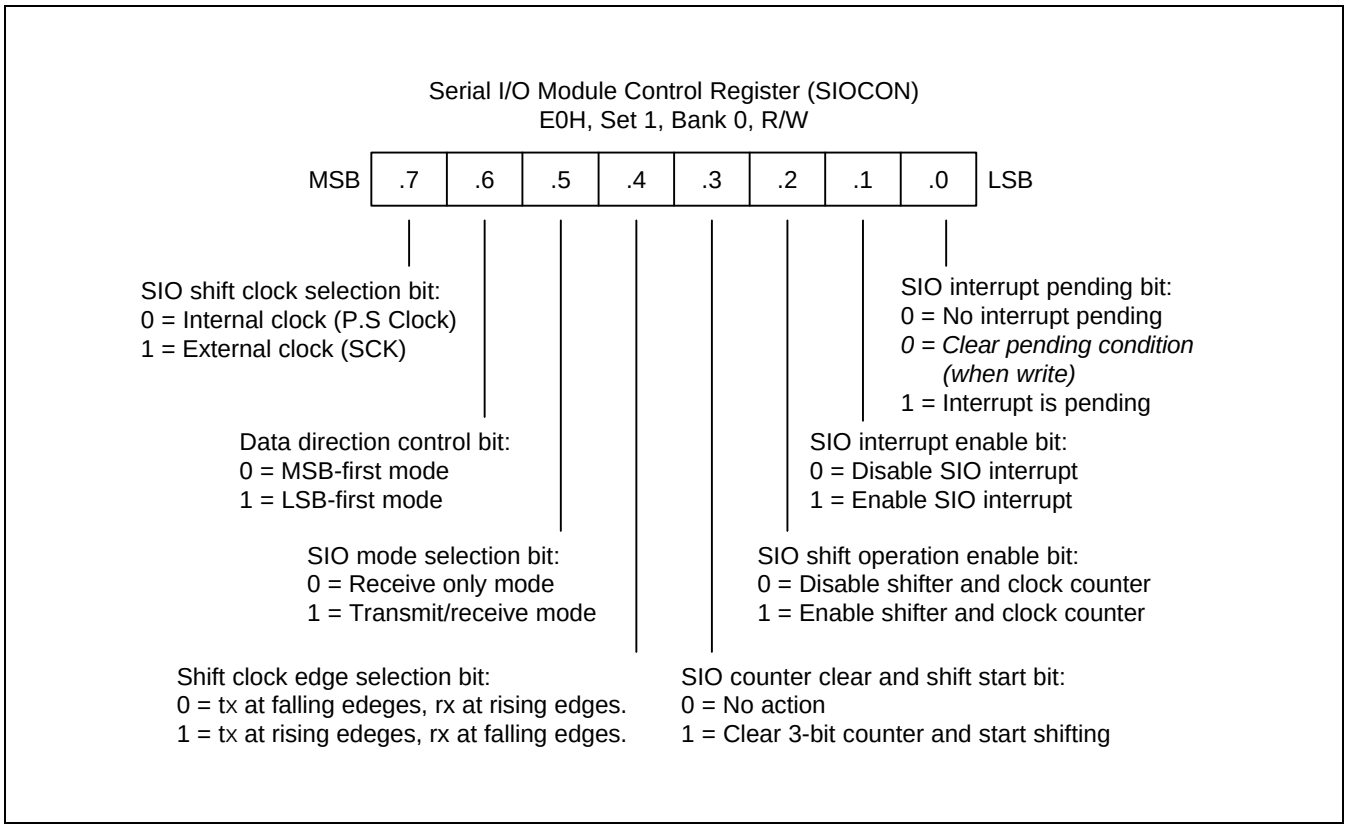


Figure 17-1. Serial I/O Module Control Register (SIOCON)

SIO PRE-SCALER REGISTER (SIOPS)

The prescaler register for serial I/O interface module, SIOPS, are located at E2H in set 1, bank 0.
The value stored in the SIO pre-scale register, SIOPS, lets you determine the SIO clock rate (baud rate) as follows:

Baud rate = Input clock (fxx/4)/(Prescaler value + 1), or SCK input clock.

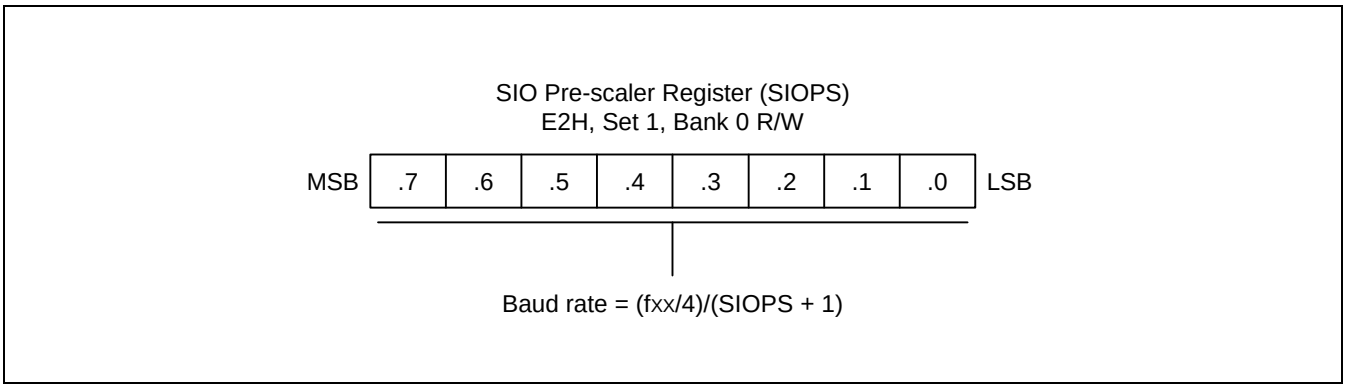


Figure 17-2. SIO Prescaler Register (SIOPS)

SIO BLOCK DIAGRAM

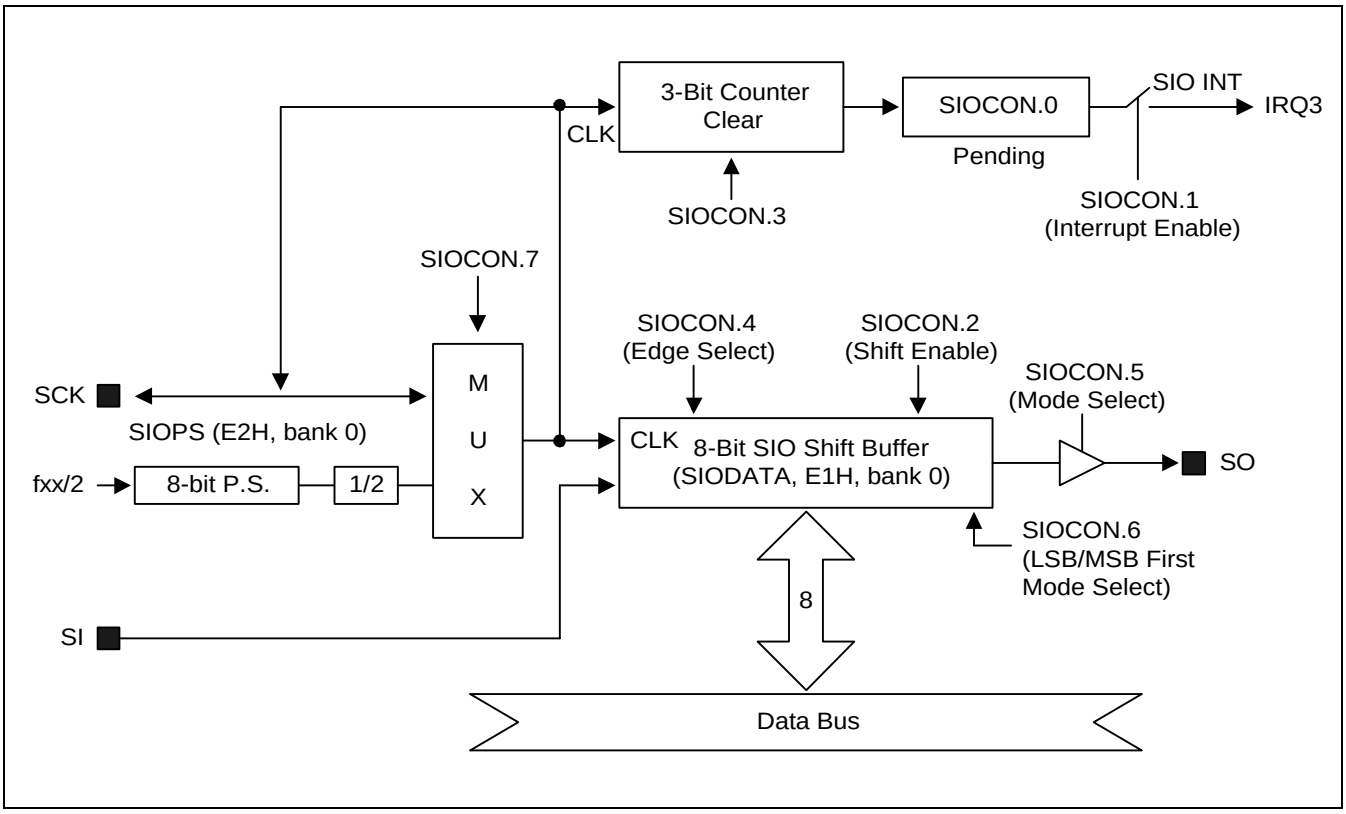


Figure 17-3. SIO Functional Block Diagram

SERIAL I/O TIMING DIAGRAM (SIO)

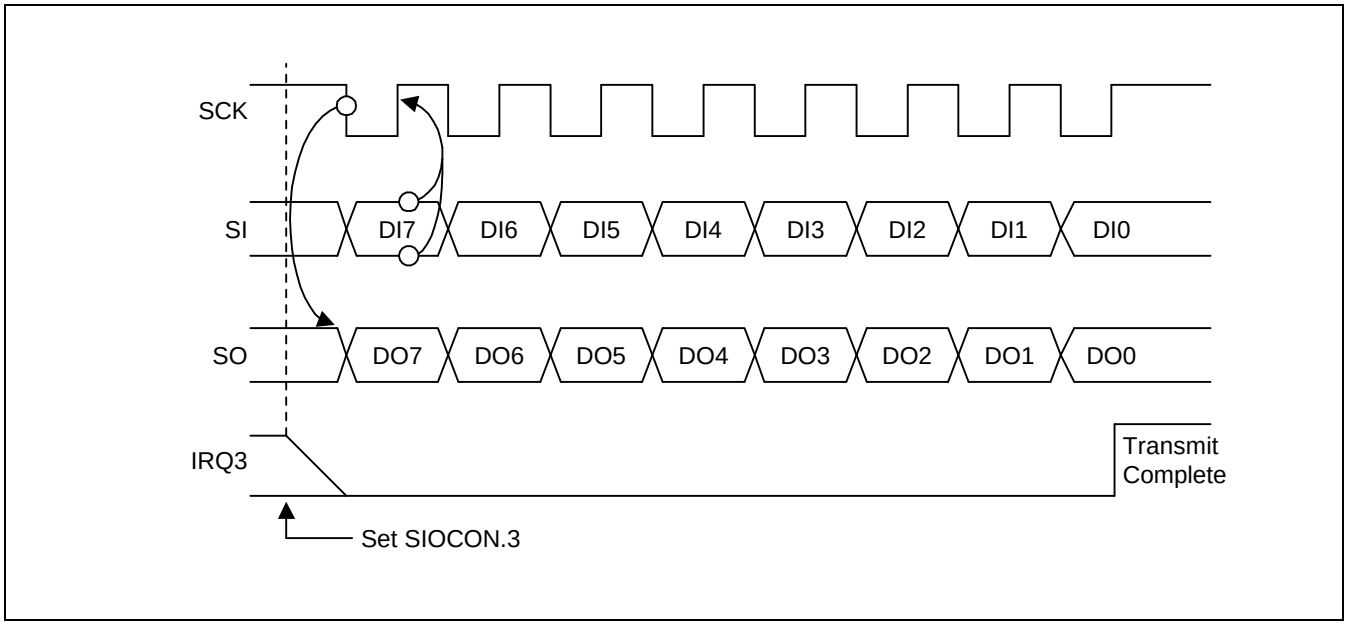


Figure 17-4. Serial I/O Timing in Transmit/Receive Mode (Tx at falling, SIOCON.4 = 0)

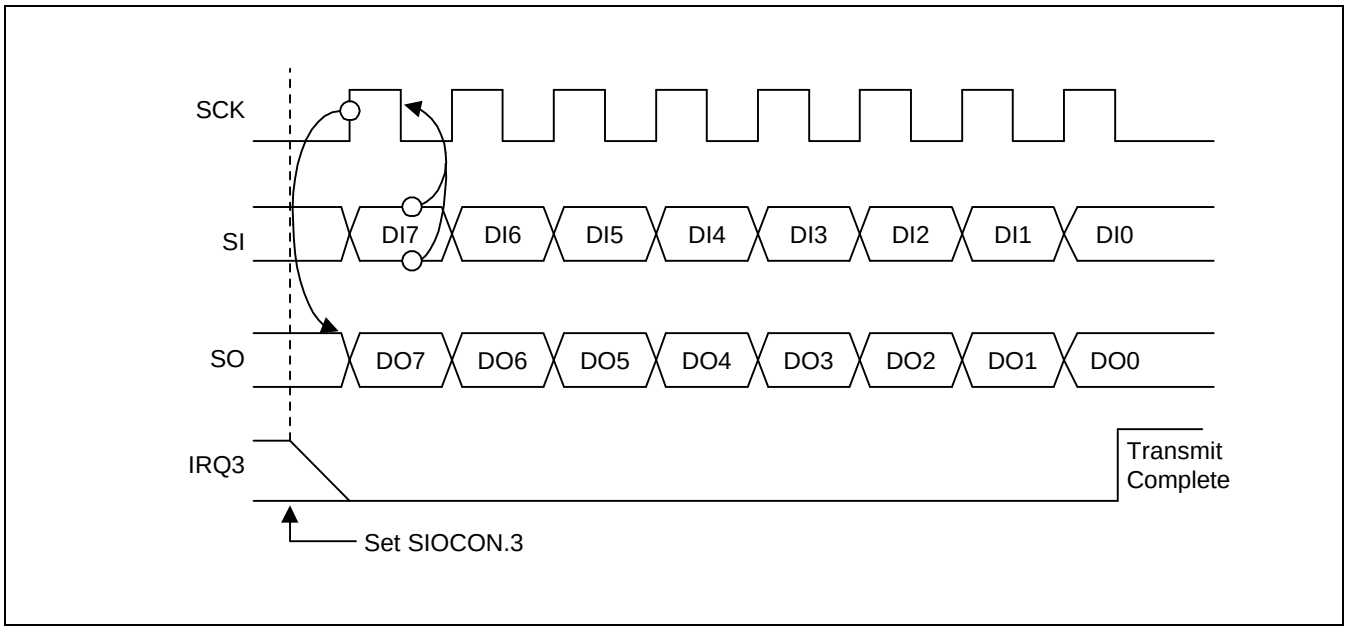


Figure 17-5. Serial I/O Timing in Transmit/Receive Mode (Tx at rising, SIOCON.4 = 1)

18 UART

OVERVIEW

The UART block has a full-duplex serial port with programmable operating modes: There is one synchronous mode and three UART (Universal Asynchronous Receiver/Transmitter) modes:

— Serial I/O with baud rate of fxx/(16 × (BRDATA+1))

— 8-bit UART mode; variable baud rate

— 9-bit UART mode; fxx/16

— 9-bit UART mode, variable baud rate

UART receive and transmit buffers are both accessed via the data register, UDATA, is set 1, bank 0 at address F9H. Writing to the UART data register loads the transmit buffer; reading the UART data register accesses a physically separate receive buffer.

When accessing a receive data buffer (shift register), reception of the next byte can begin before the previously received byte has been read from the receive register. However, if the first byte has not been read by the time the next byte has been completely received, one of the bytes will be lost.

In all operating modes, transmission is started when any instruction (usually a write operation) uses the UDATA register as its destination address. In mode 0, serial data reception starts when the receive interrupt pending bit (INTPND.5) is "0" and the receive enable bit (UARTCON.4) is "1". In mode 1, 2, and 3, reception starts whenever an incoming start bit ("0") is received and the receive enable bit (UARTCON.4) is set to "1".

PROGRAMMING PROCEDURE

To program the UART modules, follow these basic steps:

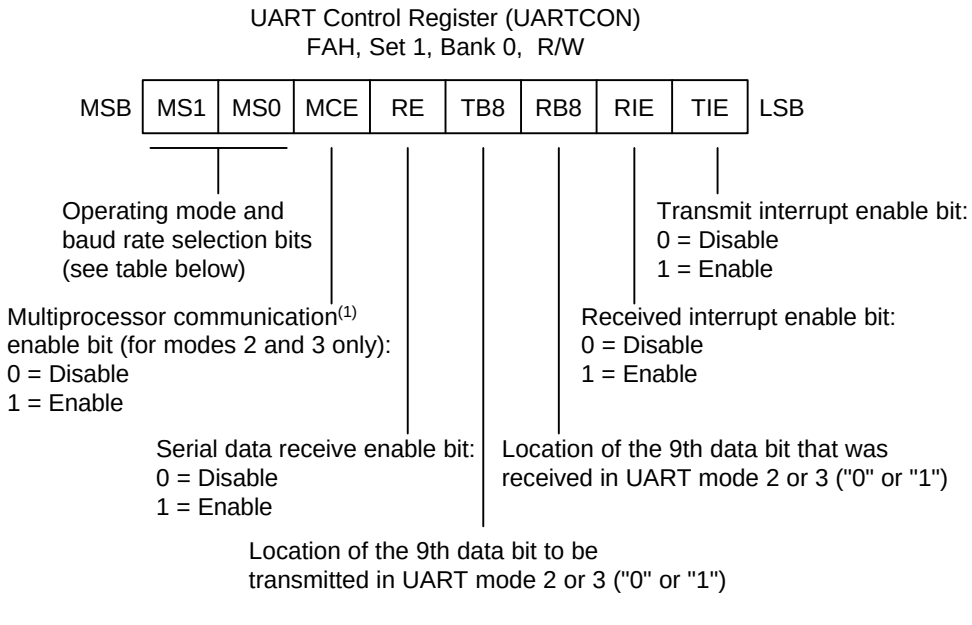
1. Configure P5.4 and P5.5 to alternative function (RxD (P5.4), TxD (P5.5)) for UART module by setting the P5CONH register to appropriatly value.
2. Load an 8-bit value to the UARTCON control register to properly configure the UART I/O module.
3. For interrupt generation, set the UART I/O interrupt enable bit (UARTCON.1 or UARTCON.0) to "1".
4. When you transmit data to the UART buffer, write data to UDATA, the shift operation starts.
5. When the shift operation (transmit/receive) is completed, UART pending bit (INTPND.4 or INTPND.5) is set to "1" and an UART interrupt request is generated.

UART CONTROL REGISTER (UARTCON)

The control register for the UART is called UARTCON in set 1, bank 0 at address FAH. It has the following control functions:

— Operating mode and baud rate selection
— Multiprocessor communication and interrupt control
— Serial receive enable/disable control
— 9th data bit location for transmit and receive operations (modes 2 and 3 only)
— UART transmit and receive interrupt control

A reset clears the UARTCON value to "00H". So, if you want to use UART module, you must write appropriate value to UARTCON.



MS1	MS0	Mode	Description(2)	Baud Rate
0	0	0	Shift register	fxx/(16 x (BRDATA + 1))
0	1	1	8-bit UART	fxx/(16 x (BRDATA + 1))
1	0	2	9-bit UART	fxx/16
1	1	3	9-bit UART	fxx/(16 x (BRDATA + 1))

NOTES:
1. In mode 2 or 3, if the UARTCON.5 bit is set to "1" then the receive interrupt will not be activated if the received 9th data bit is "0". In mode 1, if UARTCON.5 = "1" then the receive interrut will not be activated if a valid stop bit was not received. In mode 0, the UARTCON.5 bit should be "0"
2. The descriptions for 8-bit and 9-bit UART mode do not include start and stop bits for serial data receive and transmit.

Figure 18-1. UART Control Register (UARTCON)

UART INTERRUPT PENDING BITS

The UART interrupt pending bits, INTPND.5–.4, are located in set 1, bank 0 at address D0H, it contains the UART data transmit interrupt pending bit (INTPND.4) and the receive interrupt pending bit (INTPND.5).

In mode 0, the receive interrupt pending bit INTPND.5 is set to "1" when the 8th receive data bit has been shifted. In mode 1, 2, and 3, the INTPND.5 bit is set to "1" at the halfway point of the stop bit's shift time. When the CPU has acknowledged the receive interrupt pending condition, the INTPND.5 bit must then be cleared by software in the interrupt service routine.

In mode 0, the transmit interrupt pending bit INTPND.4 is set to "1" when the 8th transmit data bit has been shifted. In mode 1, 2, or 3, the INTPND.4 bit is set at the start of the stop bit. When the CPU has acknowledged the transmit interrupt pending condition, the INTPND.4 bit must then be cleared by software in the interrupt service routine.

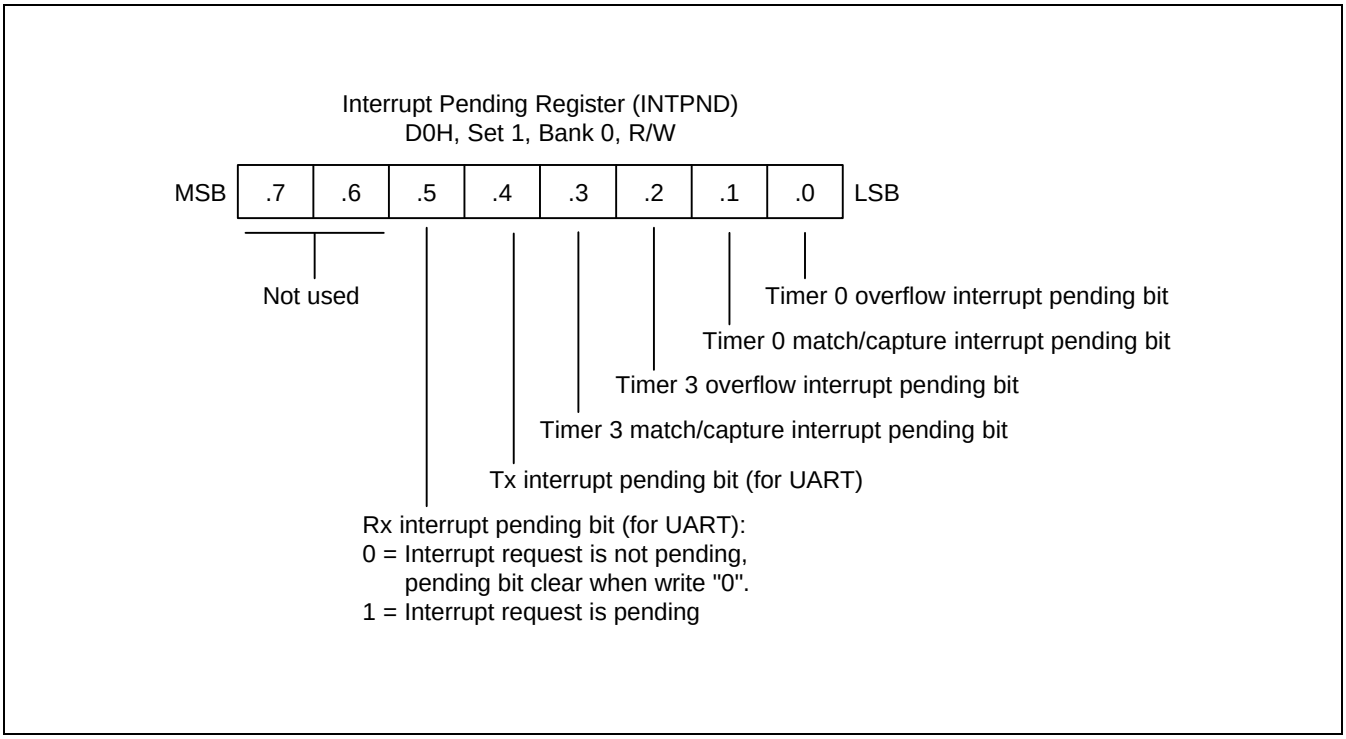


Figure 18-2. UART Interrupt Pending Bits (INTPND.5–.4)

UART DATA REGISTER (UDATA)

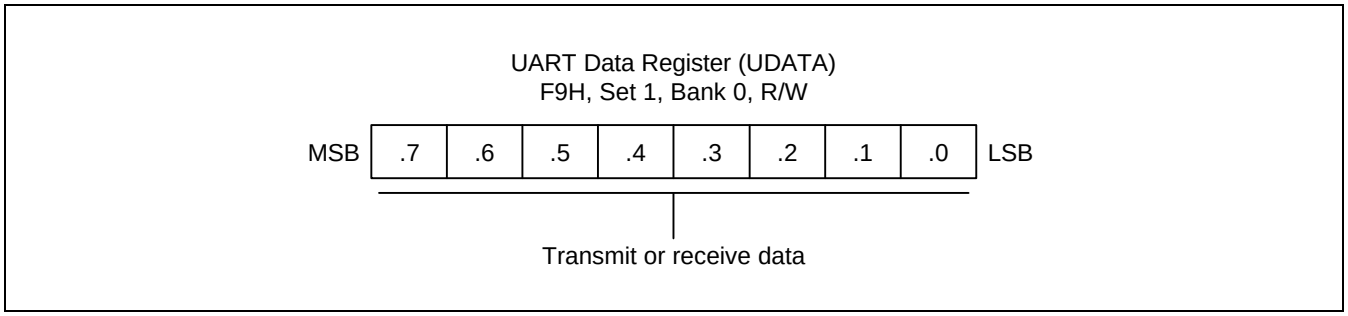


Figure 18-3. UART Data Register (UDATA)

UART BAUD RATE DATA REGISTER (BRDATA)

The value stored in the UART baud rate register, BRDATA, lets you determine the UART clock rate (baud rate).

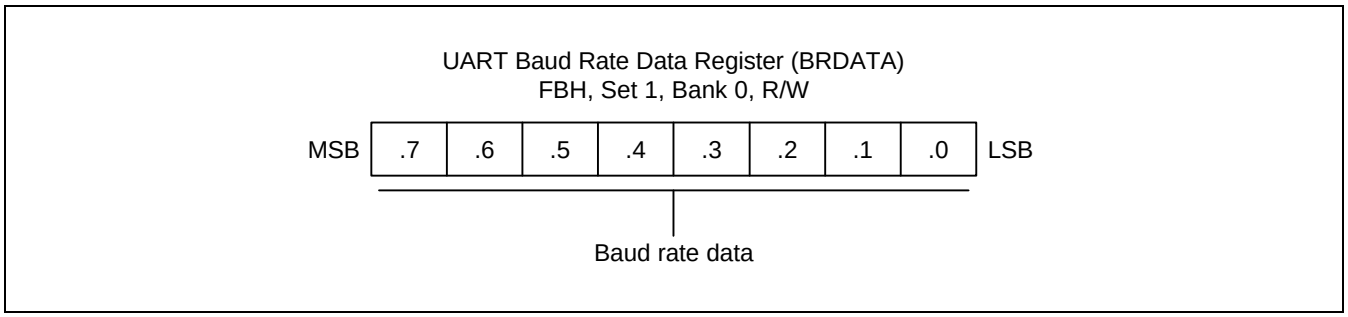


Figure 18-4. UART Baud Rate Data Register (BRDATA)

BAUD RATE CALCULATIONS

Mode 0 Baud Rate Calculation

In mode 0, the baud rate is determined by the UART baud rate data register, BRDATA in set 1, bank 0 at address FBH: Mode 0 baud rate = fxx/(16 × (BRDATA + 1)).

Mode 2 Baud Rate Calculation

The baud rate in mode 2 is fixed at the f_{OSC} clock frequency divided by 16: Mode 2 baud rate = fxx/16

Modes 1 and 3 Baud Rate Calculation

In modes 1 and 3, the baud rate is determined by the UART baud rate data register, BRDATA in set 1, bank 0 at address FBH: Mode 1 and 3 baud rate = fxx/(16 × (BRDATA + 1))

Table 18-1. Commonly Used Baud Rates Generated by BRDATA

Mode	Baud Rate	Oscillation Clock	BRDATA	
			Decimal	Hexdecimal
Mode 2	0.5 MHz	8 MHz	x	x
Mode 0	38.461 Hz	8 MHz	12	0CH
Mode 1	19.230 Hz	8 MHz	25	19H
Mode 3	9.615 Hz	8 MHz	51	33H
	4.808 Hz	8 MHz	103	67H
	19.230 Hz	4 MHz	12	0CH
	9.615 Hz	4 MHz	25	19H
	4.808 Hz	4 MHz	51	33H

BLOCK DIAGRAM

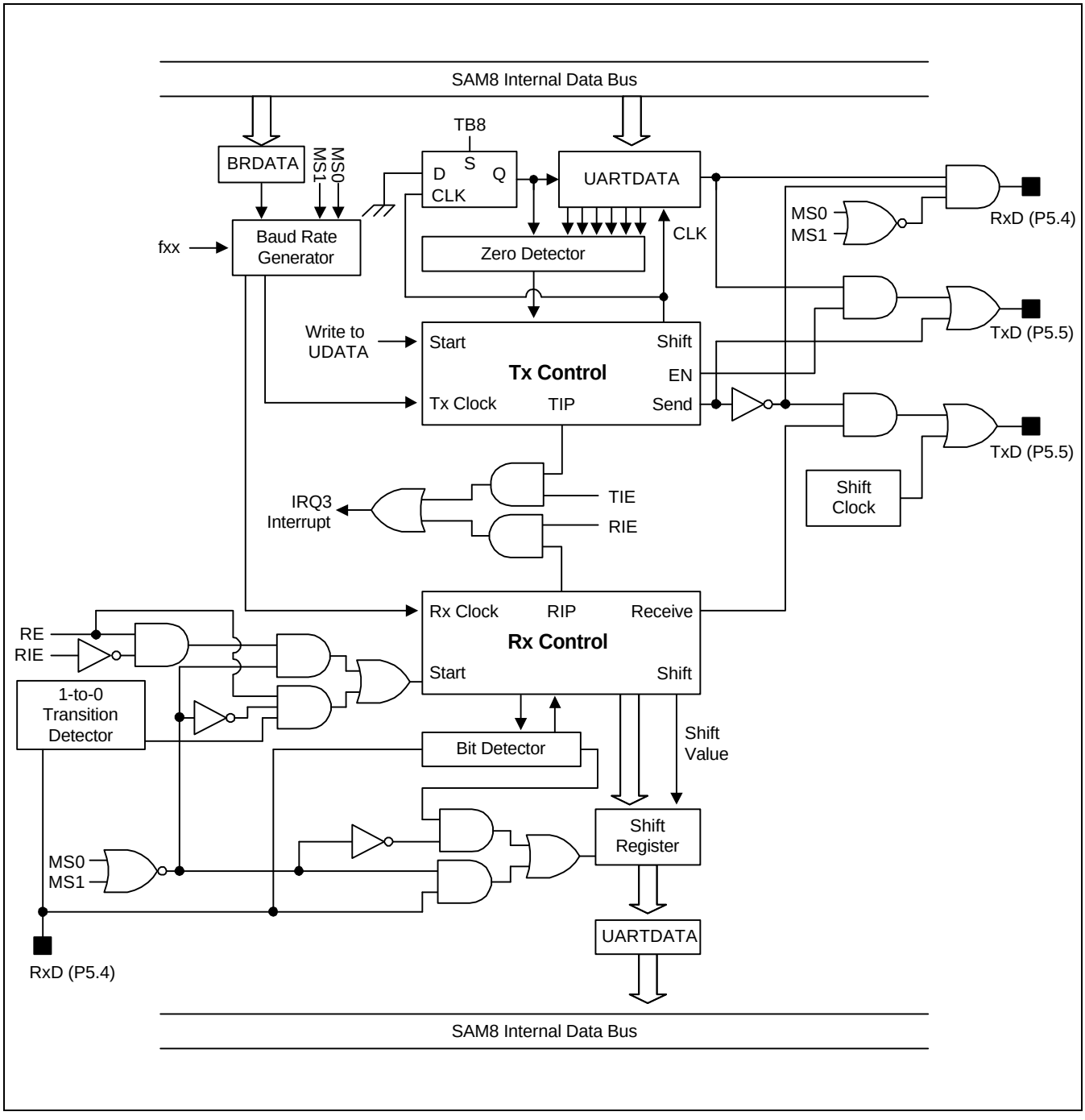


Figure 18-5. UART Functional Block Diagram

UART MODE 0 FUNCTION DESCRIPTION

In mode 0, UART is input and output through the RxD (P5.4) pin and TxD (P5.5) pin outputs the shift clock. Data is transmitted or received in 8-bit units only. The LSB of the 8-bit value is transmitted (or received) first.

Mode 0 Transmit Procedure

1. Select mode 0 by setting UARTCON.6 and .7 to "00B".
2. Write transmission data to the shift register UDATA (F9H, set 1, bank 0) to start the transmission operation.

Mode 0 Receive Procedure

1. Select mode 0 by setting UARTCON.6 and .7 to "00B".
2. Clear the receive interrupt pending bit (INTPND.5) by writing a "0" to INTPND.5.
3. Set the UART receive enable bit (UARTCON.4) to "1".
4. The shift clock will now be output to the TxD (P5.5) pin and will read the data at the RxD (P5.4) pin. A UART receive interrupt (IRQ3, vector D4H) occurs when UARTCON.1 is set to "1".

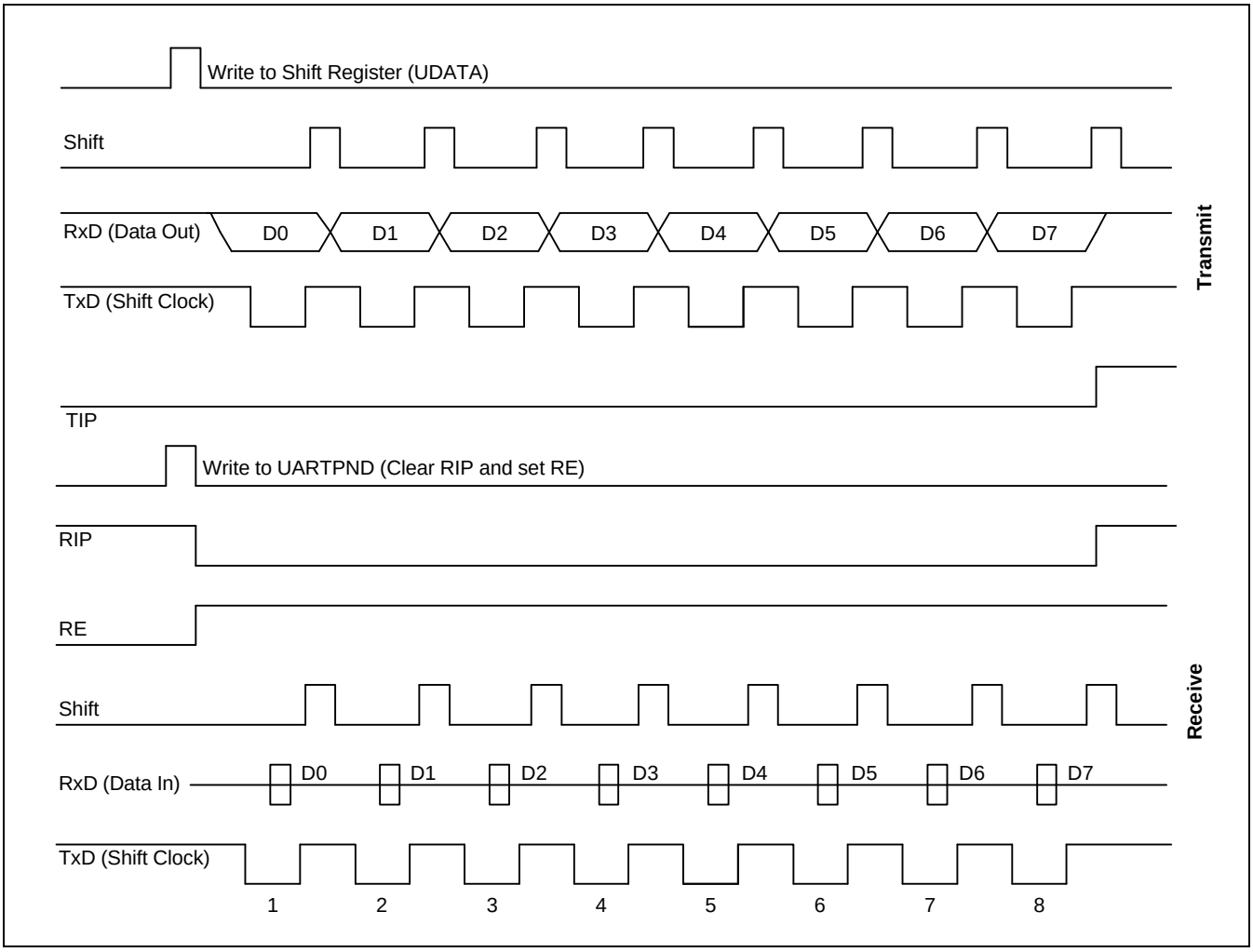


Figure 18-6. Timing Diagram for Serial Port Mode 0 Operation

SERIAL PORT MODE 1 FUNCTION DESCRIPTION

In mode 1, 10-bits are transmitted (through the TxD (P5.5) pin) or received (through the RxD (P5.4) pin). Each data frame has three components:

— Start bit ("0")

— 8 data bits (LSB first)

— Stop bit ("1")

When receiving, the stop bit is written to the RB8 bit in the UARTCON register. The baud rate for mode 1 is variable.

Mode 1 Transmit Procedure

1. Select the baud rate generated by BRDATA.
2. Select mode 1 (8-bit UART) by setting UARTCON bits 7 and 6 to '01B'.
3. Write transmission data to the shift register UDATA (F9H, set 1, bank 0). The start and stop bits are generated automatically by hardware.

Mode 1 Receive Procedure

1. Select the baud rate to be generated by BRDATA.
2. Select mode 1 and set the RE (Receive Enable) bit in the UARTCON register to "1".
3. The start bit low ("0") condition at the RxD (P5.4) pin will cause the UART module to start the serial data receive operation.

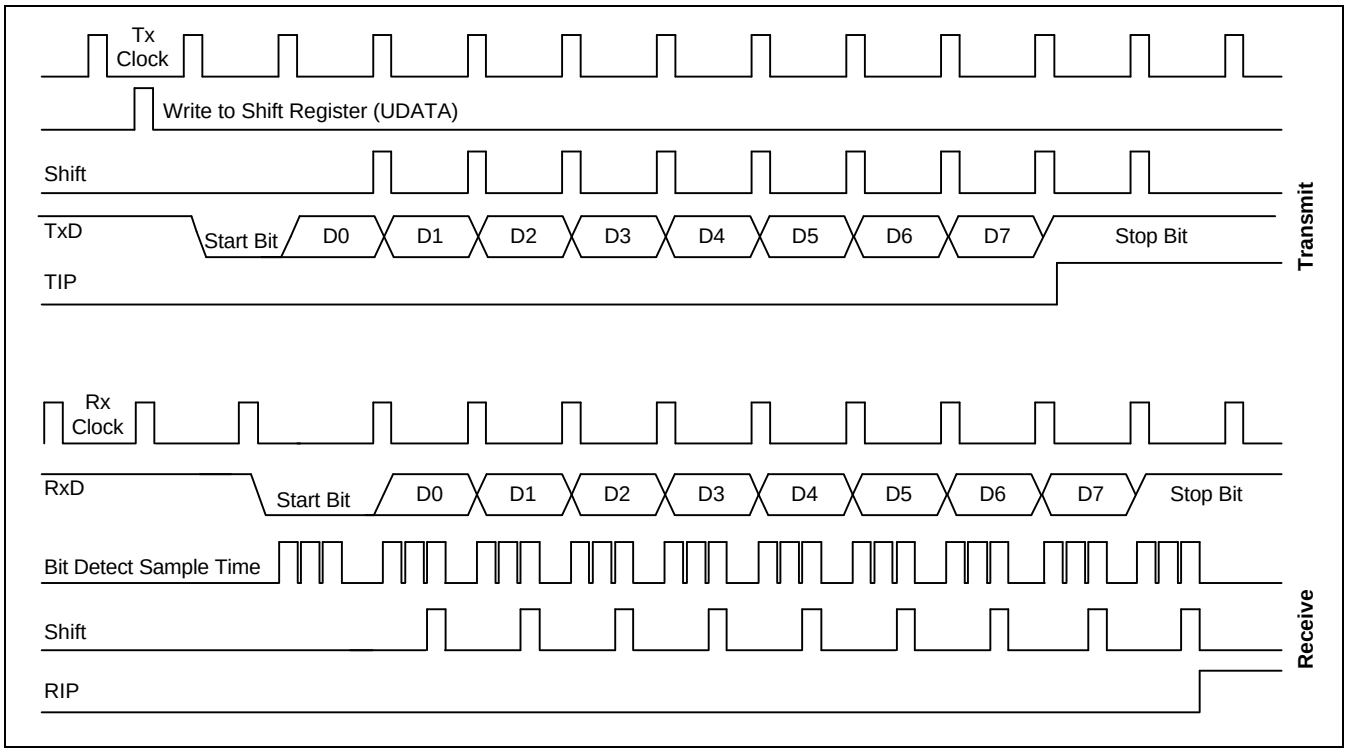


Figure 18-7. Timing Diagram for Serial Port Mode 1 Operation

SERIAL PORT MODE 2 FUNCTION DESCRIPTION

In mode 2, 11-bits are transmitted (through the TxD pin) or received (through the RxD pin). Each data frame has four components:

— Start bit ("0")

— 8 data bits (LSB first)

— Programmable 9th data bit

— Stop bit ("1")

The 9th data bit to be transmitted can be assigned a value of "0" or "1" by writing the TB8 bit (UARTCON.3). When receiving, the 9th data bit that is received is written to the RB8 bit (UARTCON.2), while the stop bit is ignored. The baud rate for mode 2 is fosc/16 clock frequency.

Mode 2 Transmit Procedure

1. Select mode 2 (9-bit UART) by setting UARTCON bits 6 and 7 to '10B'. Also, select the 9th data bit to be transmitted by writing TB8 to "0" or "1".
2. Write transmission data to the shift register, UDATA (F9H, set 1, bank 0), to start the transmit operation.

Mode 2 Receive Procedure

1. Select mode 2 and set the receive enable bit (RE) in the UARTCON register to "1".
2. The receive operation starts when the signal at the RxD pin goes to low level.

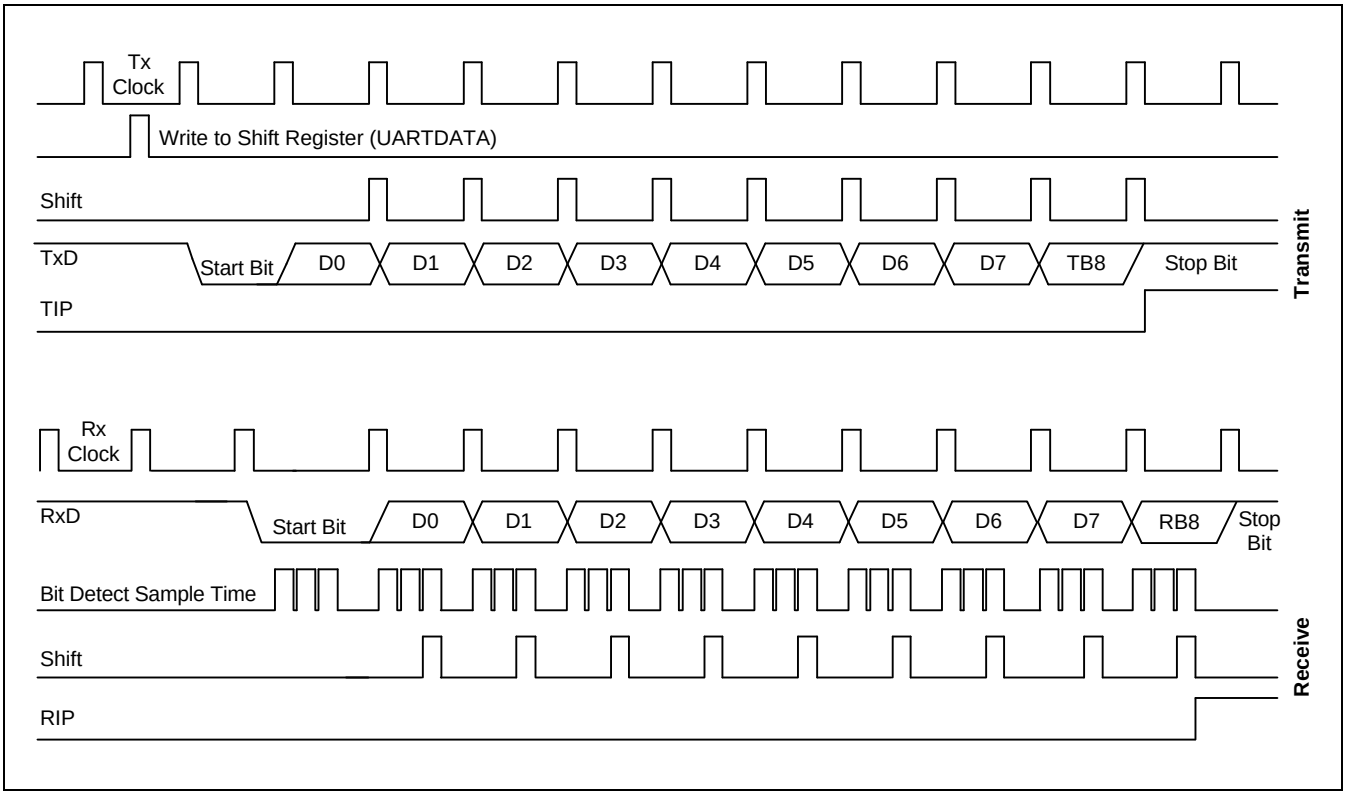


Figure 18-8. Timing Diagram for Serial Port Mode 2 Operation

SERIAL PORT MODE 3 FUNCTION DESCRIPTION

In mode 3, 11-bits are transmitted (through the TxD (P5.5) pin) or received (through the RxD (P5.4) pin). Mode 3 is identical to mode 2 except for baud rate, which is variable. Each data frame has four components:

— Start bit ("0")
— 8 data bits (LSB first)
— Programmable 9th data bit
— Stop bit ("1")

Mode 3 Transmit Procedure

1. Select the baud rate generated by BRDATA.
2. Select mode 3 operation (9-bit UART) by setting UARTCON bits 6 and 7 to '11B'. Also, select the 9th data bit to be transmitted by writing UARTCON.3 (TB8) to "0" or "1".
3. Write transmission data to the shift register, UDATA (F9H, set 1, bank 0), to start the transmit operation.

Mode 3 Receive Procedure

1. Select the baud rate to be generated by BRDATA.
2. Select mode 3 and set the RE (Receive Enable) bit in the UARTCON register to "1".
3. The receive operation will be started when the signal at the RxD (P5.4) pin goes to low level.

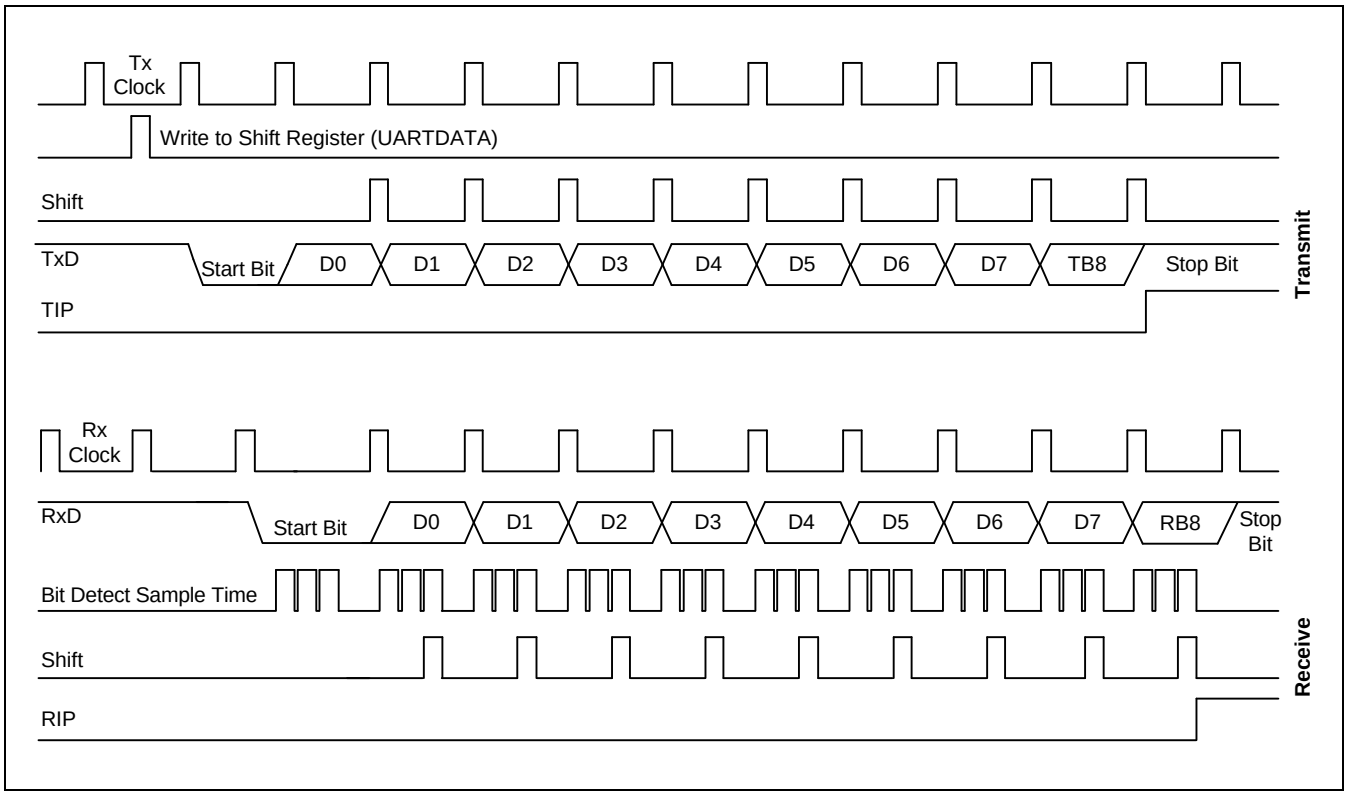


Figure 18-9. Timing Diagram for Serial Port Mode 3 Operation

SERIAL COMMUNICATION FOR MULTIPROCESSOR CONFIGURATIONS

The S3C8-series multiprocessor communication features lets a "master" S3C825A send a multiple-frame serial message to a "slave" device in a multi-S3C825A configuration. It does this without interrupting other slave devices that may be on the same serial line.

This feature can be used only in UART modes 2 or 3. In these modes 2 and 3, 9 data bits are received. The 9th bit value is written to RB8 (UARTCON.2). The data receive operation is concluded with a stop bit. You can program this function so that when the stop bit is received, the serial interrupt will be generated only if RB8 = "1".

To enable this feature, you set the MCE bit in the UARTCON register. When the MCE bit is "1", serial data frames that are received with the 9th bit = "0" do not generate an interrupt. In this case, the 9th bit simply separates the address from the serial data.

Sample Protocol for Master/Slave Interaction

When the master device wants to transmit a block of data to one of several slaves on a serial line, it first sends out an address byte to identify the target slave. Note that in this case, an address byte differs from a data byte: In an address byte, the 9th bit is "1" and in a data byte, it is "0".

The address byte interrupts all slaves so that each slave can examine the received byte and see if it is being addressed. The addressed slave then clears its MCE bit and prepares to receive incoming data bytes.

The MCE bits of slaves that were not addressed remain set, and they continue operating normally while ignoring the incoming data bytes.

While the MCE bit setting has no effect in mode 0, it can be used in mode 1 to check the validity of the stop bit. For mode 1 reception, if MCE is "1", the receive interrupt will be issue unless a valid stop bit is received.

Setup Procedure for Multiprocessor Communications

Follow these steps to configure multiprocessor communications:

1. Set all S3C825A devices (masters and slaves) to UART mode 2 or 3.
2. Write the MCE bit of all the slave devices to "1".
3. The master device's transmission protocol is:
   - First byte: the address identifying the target slave device (9th bit = "1")
   - Next bytes: data (9th bit = "0")
4. When the target slave receives the first byte, all of the slaves are interrupted because the 9th data bit is "1". The targeted slave compares the address byte to its own address and then clears its MCE bit in order to receive incoming data. The other slaves continue operating normally.

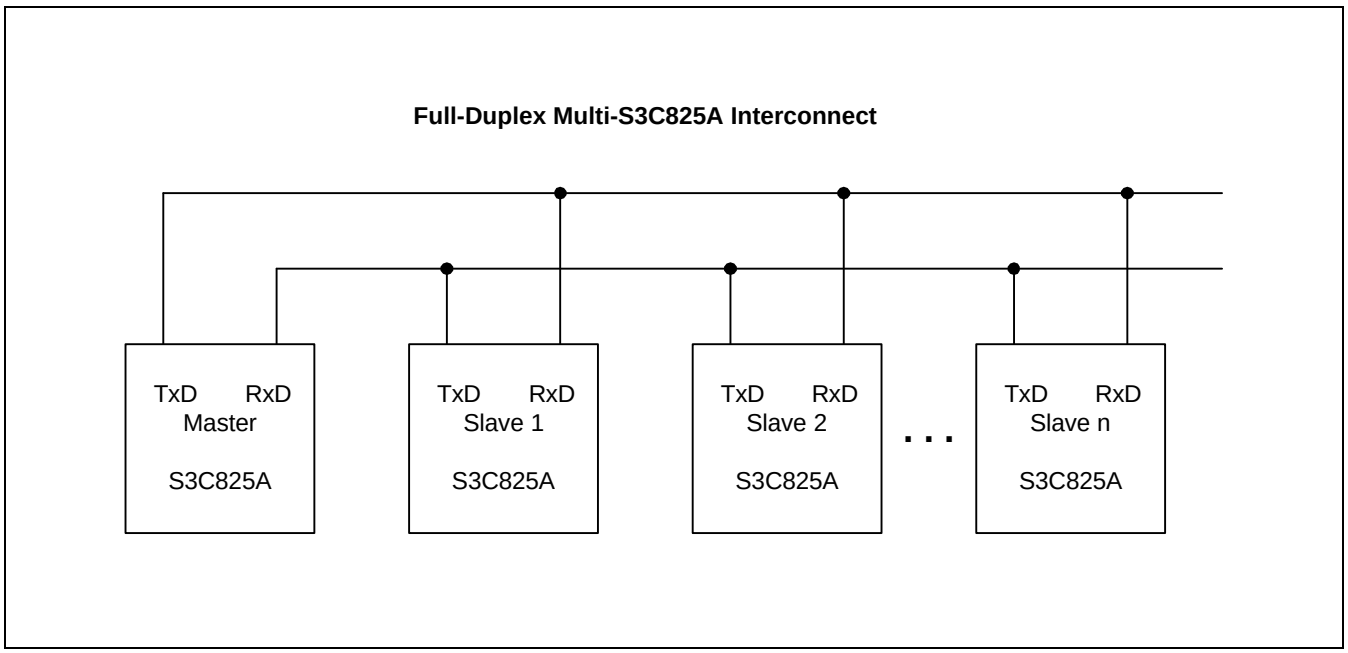


Figure 18-10. Connection Example for Multiprocessor Serial Data Communications

19 ELECTRICAL DATA

OVERVIEW

In this chapter, S3C825A electrical characteristics are presented in tables and graphs.
The information is arranged in the following order:

— Absolute maximum ratings
— D.C. electrical characteristics
— A.C. electrical characteristics
— Input/output capacitance
— Oscillation characteristics
— Oscillation stabilization time
— Data retention supply voltage in stop mode
— Serial I/O timing characteristics
— A/D converter electrical characteristics

Table 19-1. Absolute Maximum Ratings

(T_A= 25 °C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to +6.5	V
Input voltage	V_I	Ports 0–8	– 0.3 to V_{DD} + 0.3	
Output voltage	V_O	–	– 0.3 to V_{DD} + 0.3	
Output current high	I_{OH}	One I/O pin active	– 15	mA
		All I/O pins active	– 60	
Output current low	I_{OL}	One I/O pin active	+ 30	
		Total pin current for port	+ 100	
Operating temperature	T_A		– 25 to + 85	°C
Storage temperature	T_{STG}		– 65 to + 150	

Table 19-2. D.C. Electrical Characteristics

(T_A = –25 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Operating voltage	V_{DD}	fx = 0.4–4 MHz, 32.8 kHz	2.0	–	5.5	V
		fx = 0.4–8 MHz	2.2	–	5.5	
Input high voltage	V_{IH1}	Ports 0–8	0.8 V_{DD}	–	V_{DD}	
	V_{IH2}	RESET	0.8 V_{DD}		V_{DD}	
	V_{IH3}	X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}	V_{DD}–0.1		V_{DD}	
Input low voltage	V_{IL1}	Ports 0–8	–	–	0.2 V_{DD}	
	V_{IL2}	RESET			0.2 V_{DD}	
	V_{IL3}	X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}			0.1	
Output high voltage	V_{OH}	V_{DD} = 4.5 V to 5.5 V All output ports; I_{OH} = –1 mA	V_{DD} – 1.0	–	V_{DD}	
Output low voltage	V_{OL}	V_{DD} = 4.5 V to 5.5 V All output ports; I_{OL} = 10 mA	–	–	2.0	

Table 19-2. D.C. Electrical Characteristics (Continued)

(T_A = -25 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions		Min.	Typ.	Max.	Unit
Input high leakage current	I_{LIH1}	$V_{IN} = V_{DD}$ All input pins except X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}		–	–	3	uA
	I_{LIH2}	$V_{IN} = V_{DD}$, X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}				20	
Input low leakage current	I_{LIL1}	V_{IN} = 0 V All input pins except RESET, X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}		–	–	-3	
	I_{LIL2}	V_{IN} = 0 V, X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}				-20	
Output high leakage current	I_{LOH}	$V_{OUT} = V_{DD}$ All output pins		–	–	3	
Output low leakage current	I_{LOL}	V_{OUT} = 0 V All output pins		–	–	-3	
Pull-up resistor	R_{L1}	V_{IN} = 0 V Port 0–8	V_{DD} = 5 V	25	50	100	kΩ
			V_{DD} = 3 V	50	100	150	
	R_{L2}	V_{IN} = 0 V RESET	V_{DD} = 5 V	150	250	400	
			V_{DD} = 3 V	300	500	700	
Oscillator feed back resistors	R_{OSC1}	V_{DD} = 5 V, T_A = 25 °C $X_{IN} = V_{DD}$, X_{OUT} = 0 V		300	750	1500	kΩ
	R_{OSC2}	V_{DD} = 5 V, T_A = 25 °C $XT_{IN} = V_{DD}$, XT_{OUT} = 0 V		1500	3000	4500	
LCD voltage dividing resistor	R_{LCD}	T_A = 25 °C		40	60	80	kΩ
$\|V_{LCD} - COMi\|$ voltage drop (I = 0–7)	V_{DC}	V_{DD} = 2.7 V to 5.5 V – 15 μA per common pin		–	–	120	mV
$\|V_{LCD} - SEGx\|$ voltage drop (x = 0–31)	V_{DS}	V_{DD} = 2.7 V to 5.5 V – 15 μA per common pin		–	–	120	mV
Middle output voltage	V_{LC2}	V_{DD} = 2.7 V to 5.5 V LCD clock = 0 Hz $V_{LC1} = V_{DD}$		$0.8V_{DD}$–0.2	$0.8V_{DD}$	$0.8V_{DD}$+0.2	V
	V_{LC3}			$0.6V_{DD}$–0.2	$0.6V_{DD}$	$0.6V_{DD}$+0.2	
	V_{LC4}			$0.4V_{DD}$–0.2	$0.4V_{DD}$	$0.4V_{DD}$+0.2	
	V_{LC5}			$0.2V_{DD}$–0.2	$0.2V_{DD}$	$0.2V_{DD}$+0.2	

Table 19-2. D.C. Electrical Characteristics (Concluded)

(T_A = -25 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions		Min	Typ	Max	Units
Supply Current (1)	I_{DD1}	Run mode : V_{DD} = 5 V ± 10% Crystal oscillator	6.0 MHz	–	4.0	9.0	mA
		C1 = C2 = 22pF	4.19 MHz		3.0	6.0	
		V_{DD} = 3 V ± 10%	6.0 MHz		2.0	4.0	
			4.19 MHz		1.5	3.4	
	I_{DD2}	Idle mode : V_{DD} = 5 V ± 10% Crystal oscillator	6.0 MHz		1.0	2.5	
		C1 = C2 = 22pF	4.19 MHz		0.9	2.0	
		V_{DD} = 3 V ± 10%	6.0 MHz		0.6	1.2	
			4.19 MHz		0.4	0.8	
	I_{DD3}	Run mode : V_{DD} = 3 V ± 10% 32kHz Crystal oscillator			15	30	uA
	I_{DD4}	Idle mode : V_{DD} = 3 V ± 10% 32kHz Crystal oscillator			6	15	
	I_{DD5}	Stop mode : V_{DD} = 5 V ± 10% T_A = 25 °C			0.5	3	
		Stop mode : V_{DD} = 3 V ± 10% T_A = 25 °C			0.3	2	

NOTES:

1. Supply current does not include current drawn through internal pull-up resistors, LCD voltage dividing resistors, and ADC.
2. I_{DD1} and I_{DD2} include power consumption for subsystem clock oscillation.
3. I_{DD3} and I_{DD4} are current when main system clock oscillation stops and the subsystem clock is used.
4. I_{DD5} is current when main system clock and subsystem clock oscillation stops.
5. Every values in this table is measured when bits 4–3 of the system clock control register (CLKCON.4–.3) is set to “11B”.

Table 19-3. A.C. Electrical Characteristics

(T_A = -25 °C to +85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input high, low width (P2.4–P2.7, P4.0–P4.7)	tINTH, tINTL	V_{DD} = 5 V	200	–		ns
RESET input low width	tRSL	V_{DD} = 5 V	10	–	–	us

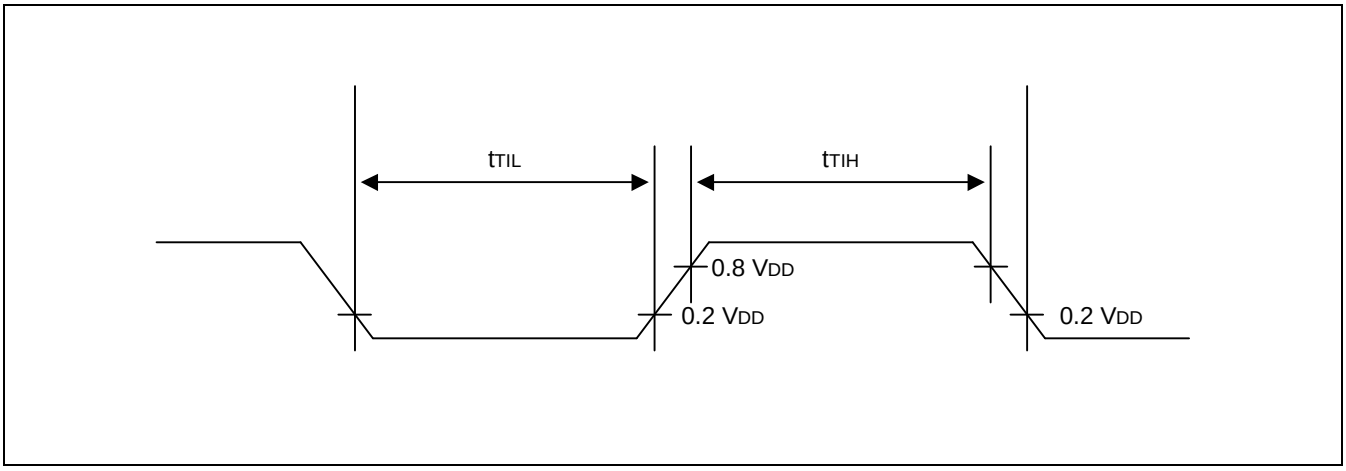


Figure 19-1. Input Timing for External Interrupts (P2.4–P2.7, P4)

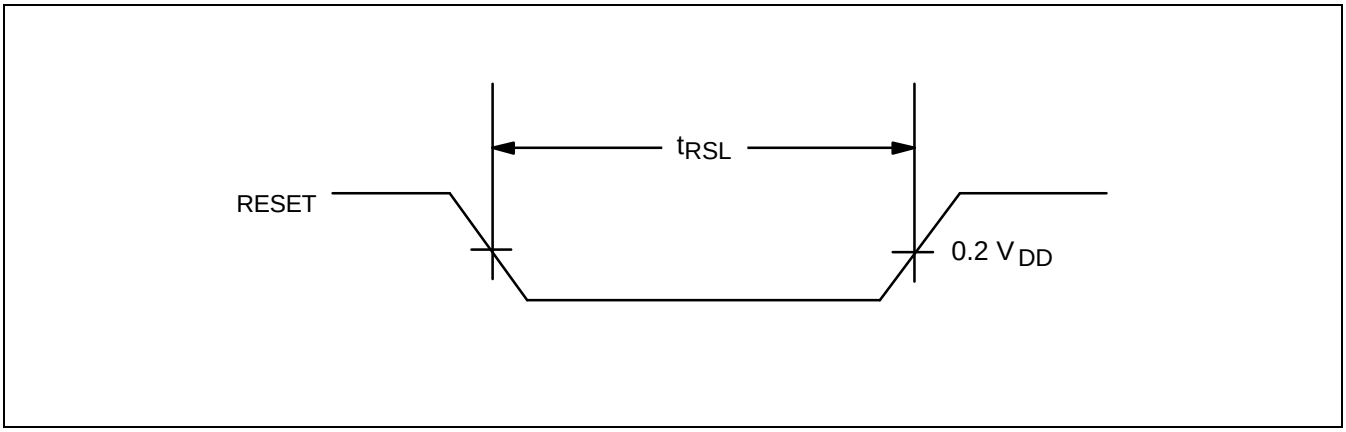


Figure 19-2. Input Timing for RESET

Table 19-4. Input/Output Capacitance

(T_A = -25 °C to +85 °C, V_{DD} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input capacitance	C_{IN}	f = 1 MHz; unmeasured pins are returned to V_{SS}	–	–	10	pF
Output capacitance	C_{OUT}					
I/O capacitance	C_{IO}					

Table 19-5. Data Retention Supply Voltage in Stop Mode

(T_A = -25 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}		2.0	–	5.5	V
Data retention supply current	I_{DDDR}	V_{DDDR} = 2 V (TA = 25 °C) Stop mode	–	–	1	uA

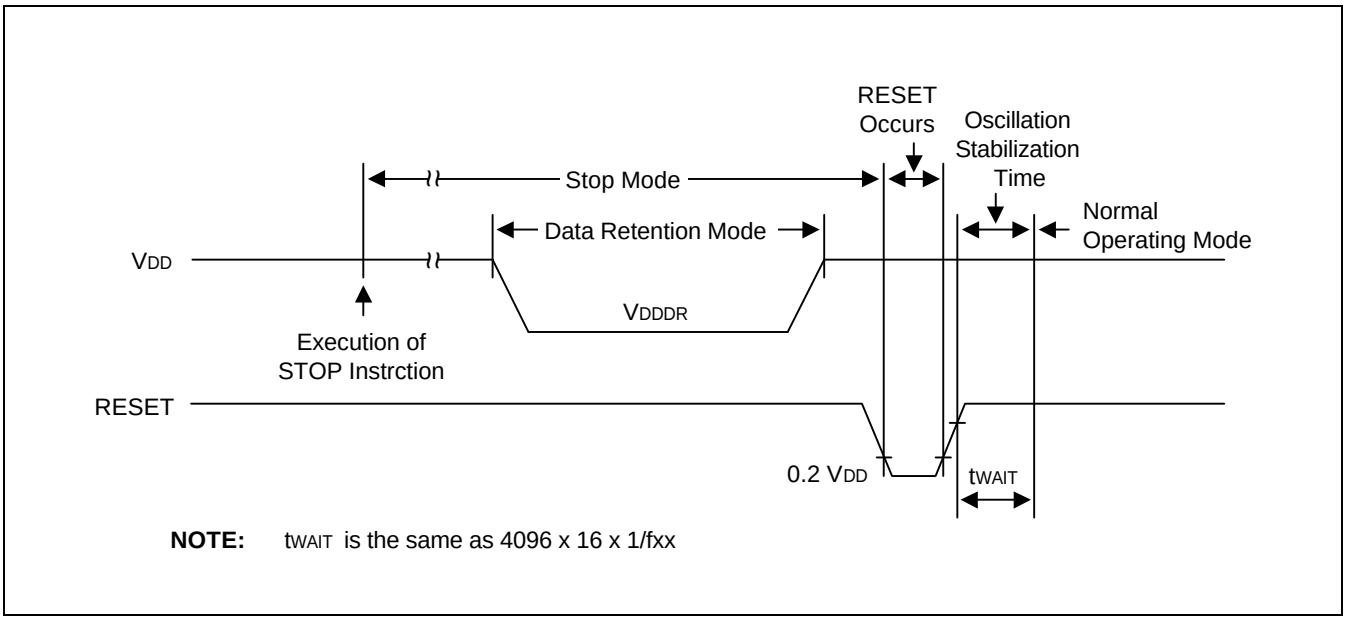


Figure 19-3. Stop Mode Release Timing Initiated by RESET

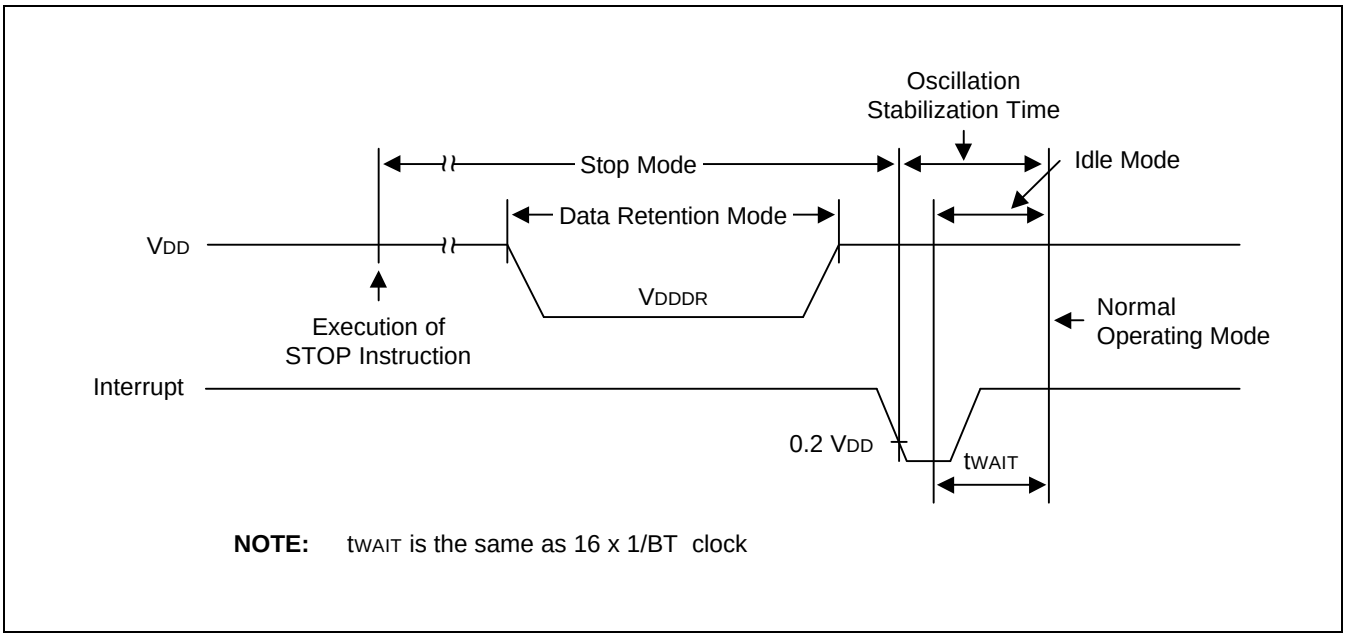


Figure 19-4. Stop Mode Release Timing Initiated by Interrupts

Table 19-6. A/D Converter Electrical Characteristics

(T_A = -25 °C to +85 °C, V_{DD} = 2.7 V to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Resolution			–	10	–	bit
Total Accuracy			–	–	± 3	LSB
Integral Linearity Error	ILE	V_{DD} = 5.12 V	–	–	± 2	
Differential Linearity Error	DLE	AV_{REF} = 5.12 V AV_{SS} = 0 V	–	–	± 1	
Offset Error of Top	EOT	CPU clock = 8 MHz	–	± 1	± 3	
Offset Error of Bottom	EOB		–	± 1	± 3	
Conversion Time (1)	T_{CON}	10-bit resolution 50 x fxx/4, fxx = 8 MHz	25	–	–	µS
Analog Input Voltage	V_{IAN}	–	AV_{SS}	–	AV_{REF}	V
Analog Input Impedance	R_{AN}	–	2	1000	–	MΩ
Analog Reference Voltage	AV_{REF}	–	2.7	–	V_{DD}	V
Analog Ground	AV_{SS}	–	V_{SS}	–	V_{SS} + 0.3	
Analog Input Current	I_{ADIN}	AV_{REF} = V_{DD} = 5V		–	10	µA
Analog Block Current (2)	I_{ADC}	AV_{REF} = V_{DD} = 5V	–	1	3	mA
		AV_{REF} = V_{DD} = 3V		0.5	1.5	
		AV_{REF} = V_{DD} = 5V When power down mode		100	500	nA

NOTES:

1. 'Conversion time' is the time required from the moment a conversion operation starts until it ends.
2. I_{ADC} is an operating current of A/D converter.

Table 19-7. Synchronous SIO Electrical Characteristics

(T_A = –25 °C to +85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
SCK cycle time	t_{CKY}	External SCK source	1000	–	–	ns
		Internal SCK source	1000			
SCK high, low width	t_{KH}, t_{KL}	External SCK source	500	–	–	
		Internal SCK source	t_{KCY}/2–50			
SI setup time to SCK high	t_{SIK}	External SCK source	250	–	–	
		Internal SCK source	250			
SI hold time to SCK high	t_{KSI}	External SCK source	400	–	–	
		Internal SCK source	400			
Output delay for SCK to SO	t_{KSO}	External SCK source	–	–	300	
		Internal SCK source			250	

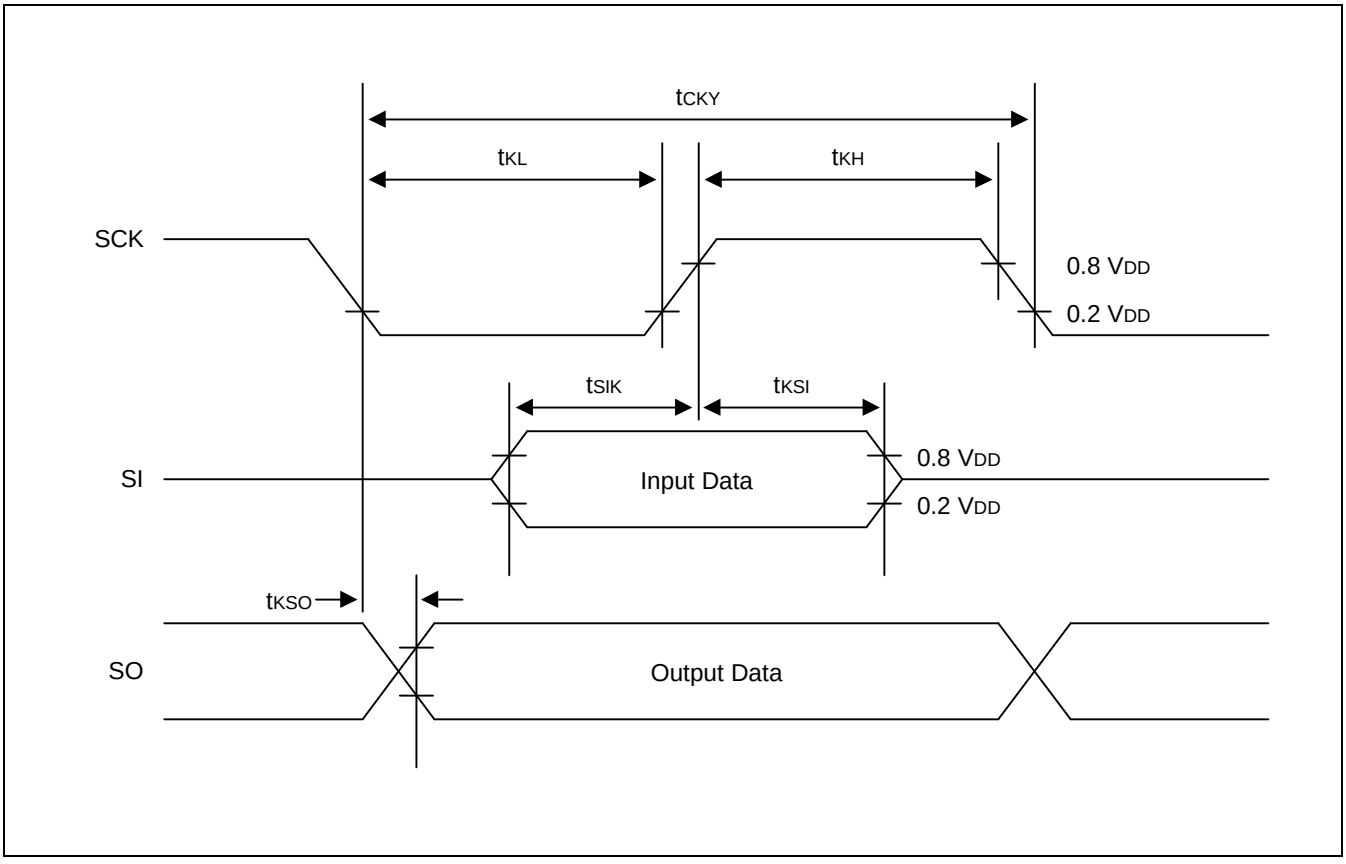


Figure 19-5. Serial Data Transfer Timing

Table 19-8. Main Oscillator Characteristics

(T_A = –25 °C to + 85 °C)

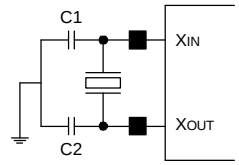
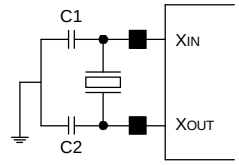
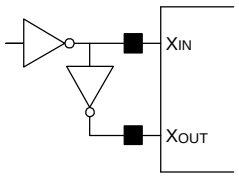
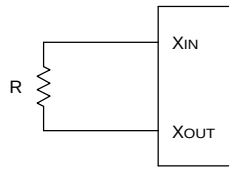
Oscillator	Clock Circuit	Parameter	Condition (V_{DD})	Min	Typ	Max	Unit
Crystal		Main oscillation frequency	2.2 V – 5.5 V	0.4	–	8	MHz
			2.0 V – 5.5 V	0.4	–	4	
Ceramic		Main oscillation frequency	2.2 V – 5.5 V	0.4	–	8	
			2.0 V – 5.5 V	0.4	–	4	
External clock		X_{IN} input frequency	2.2 V – 5.5 V	0.4	–	8	
			2.0 V – 5.5 V	0.4	–	4	
RC		Frequency	5.0 V	0.4	–	2	MHz
		Frequency	3.0 V	0.4	–	1	

Table 19-9. Sub Oscillator Characteristics

(T_A = – 25 °C to + 85 °C)

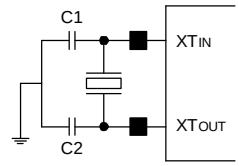
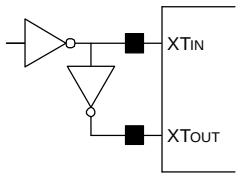
Oscillator	Clock Circuit	Parameter	Condition (V_{DD})	Min	Typ	Max	Unit
Crystal		Sub oscillation frequency	2.0 V–5.5 V	32	32.768	35	kHz
External clock		XT_{IN} input frequency	2.0 V–5.5 V	32	–	100	kHz

Table 19-10. Main Oscillator Stabilization Time

(T_A = -25 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	fx > 400 kHz	–	–	40	ms
Ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	10	ms
External clock	X_{IN} input high and low level width (t_{XH}, t_{XL})	62.5	–	1250	ns

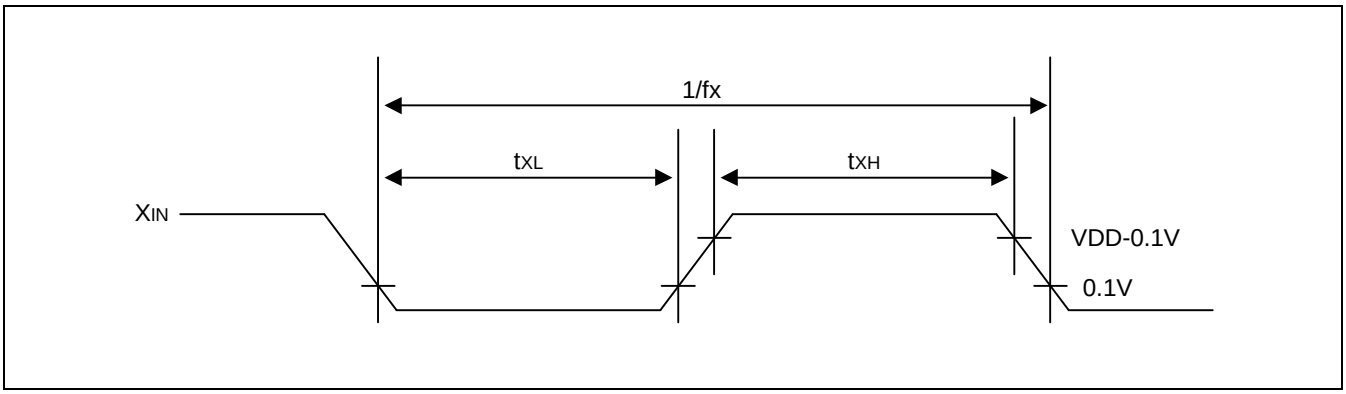


Figure 19-6. Clock Timing Measurement at X_{IN}

Table 19-11. Sub Oscillator Stabilization Time

(T_A = -25 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	–	–	–	10	s
External clock	XT_{IN} input high and low level width (t_{XTL}, t_{XTH})	5	–	15	µs

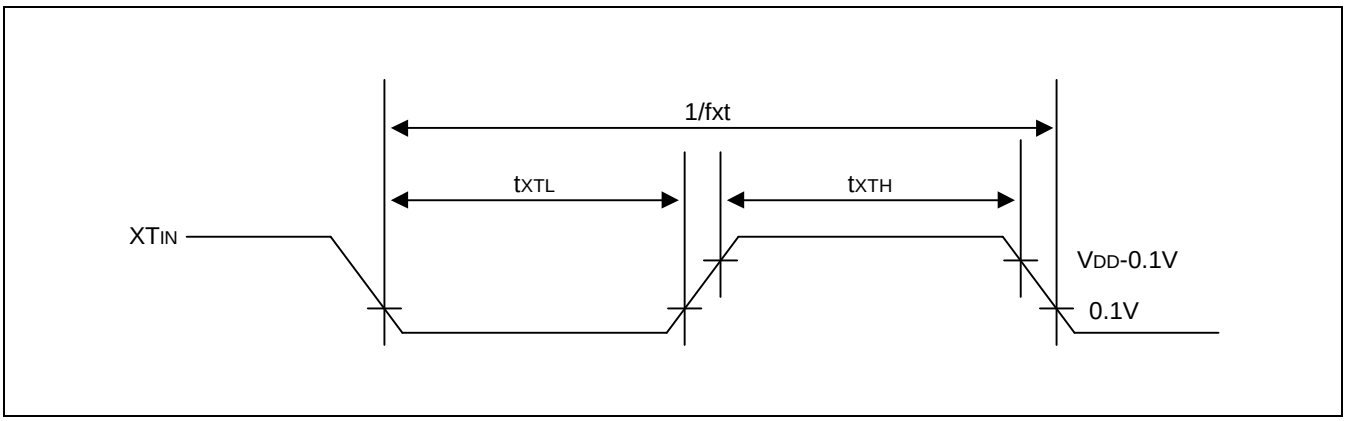


Figure 19-7. Clock Timing Measurement at XT_{IN}

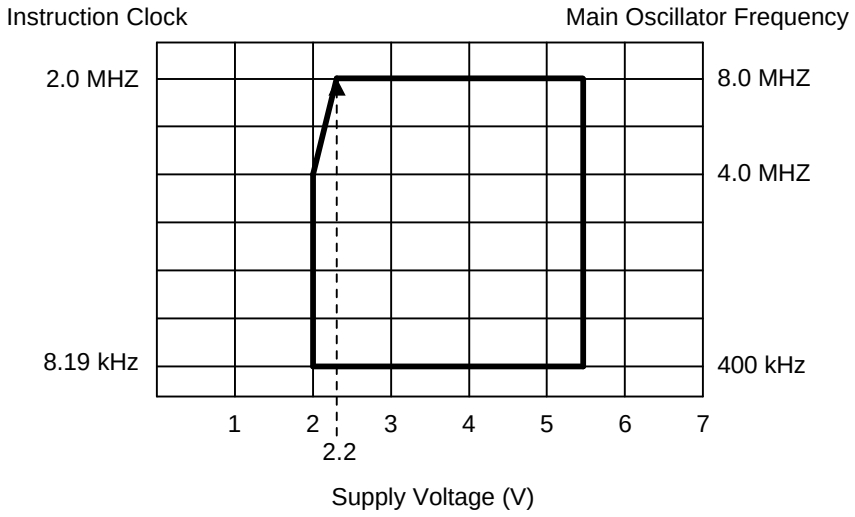


Figure 19-8. Operating Voltage Range

Table 19- 12. UART Timing Characteristics in Mode 0 (8 MHz)

(T_A = – 25°C to + 85°C, V_{DD} = 2.0 V to 5.5 V, Load capacitance = 80 pF)

Parameter	Symbol	Min	Typ	Max	Unit
Serial port clock cycle time	t_{SCK}	625	$t_{CPU} \times 6$	875	ns
Output data setup to clock rising edge	t_{S1}	300	$t_{CPU} \times 5$	–	
Clock rising edge to input data valid	t_{S2}	–	–	300	
Output data hold after clock rising edge	t_{H1}	$t_{CPU} – 50$	t_{CPU}	–	
Input data hold after clock rising edge	t_{H2}	0	–	–	
Serial port clock High, Low level width	t_{HIGH}, t_{LOW}	250	$t_{CPU} \times 3$	500	

NOTES:
1. All timings are in nanoseconds (ns) and assume a 10-MHz CPU clock frequency.
2. The unit t_{CPU} means one CPU clock period.

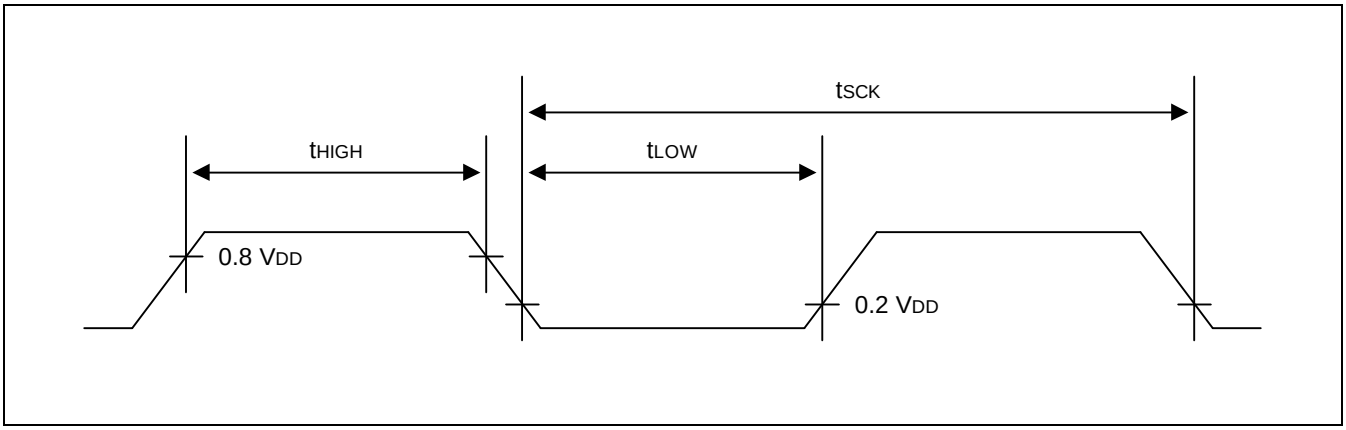
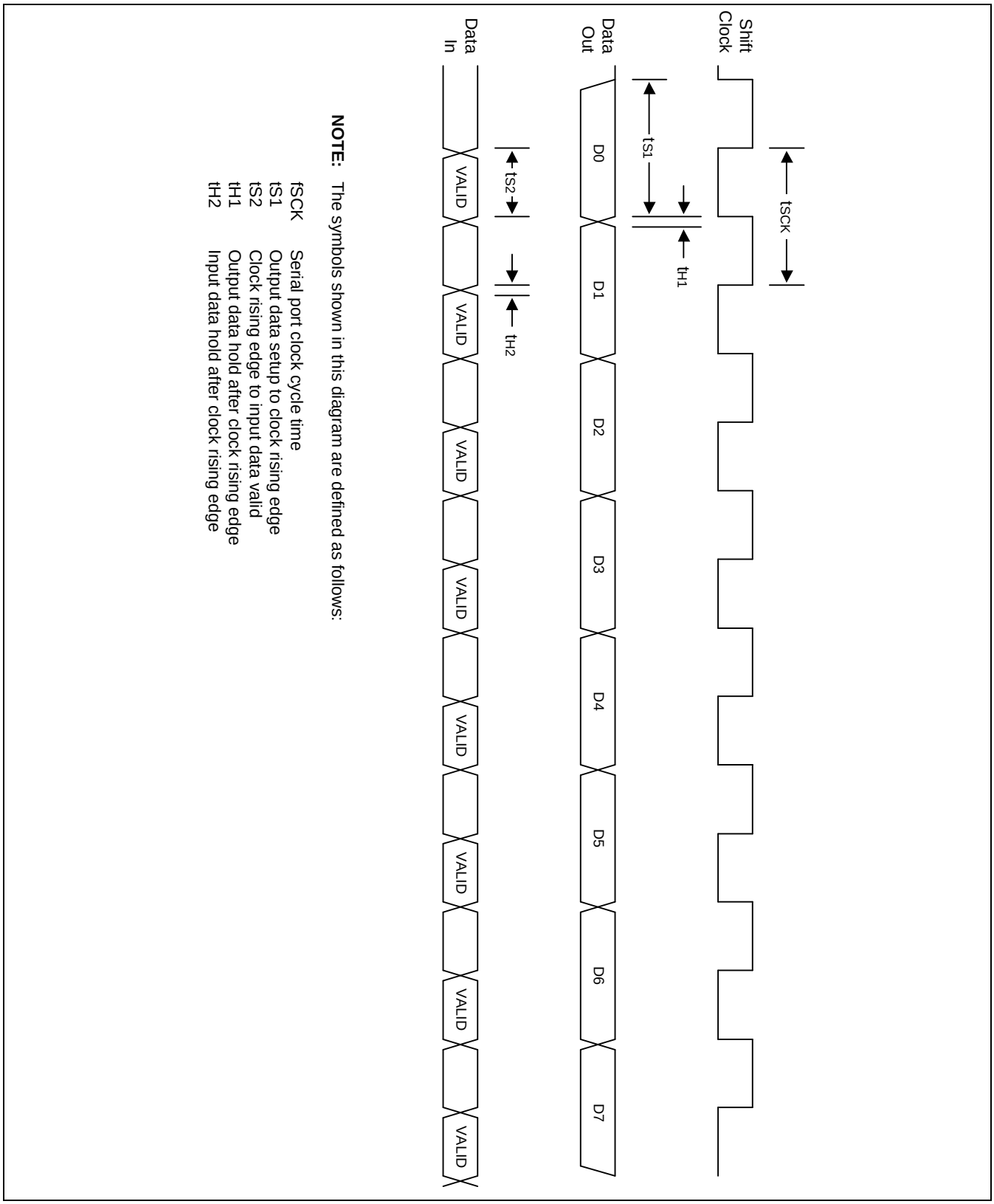


Figure 19-9. Waveform for UART Timing Characteristics



NOTE: The symbols shown in this diagram are defined as follows:

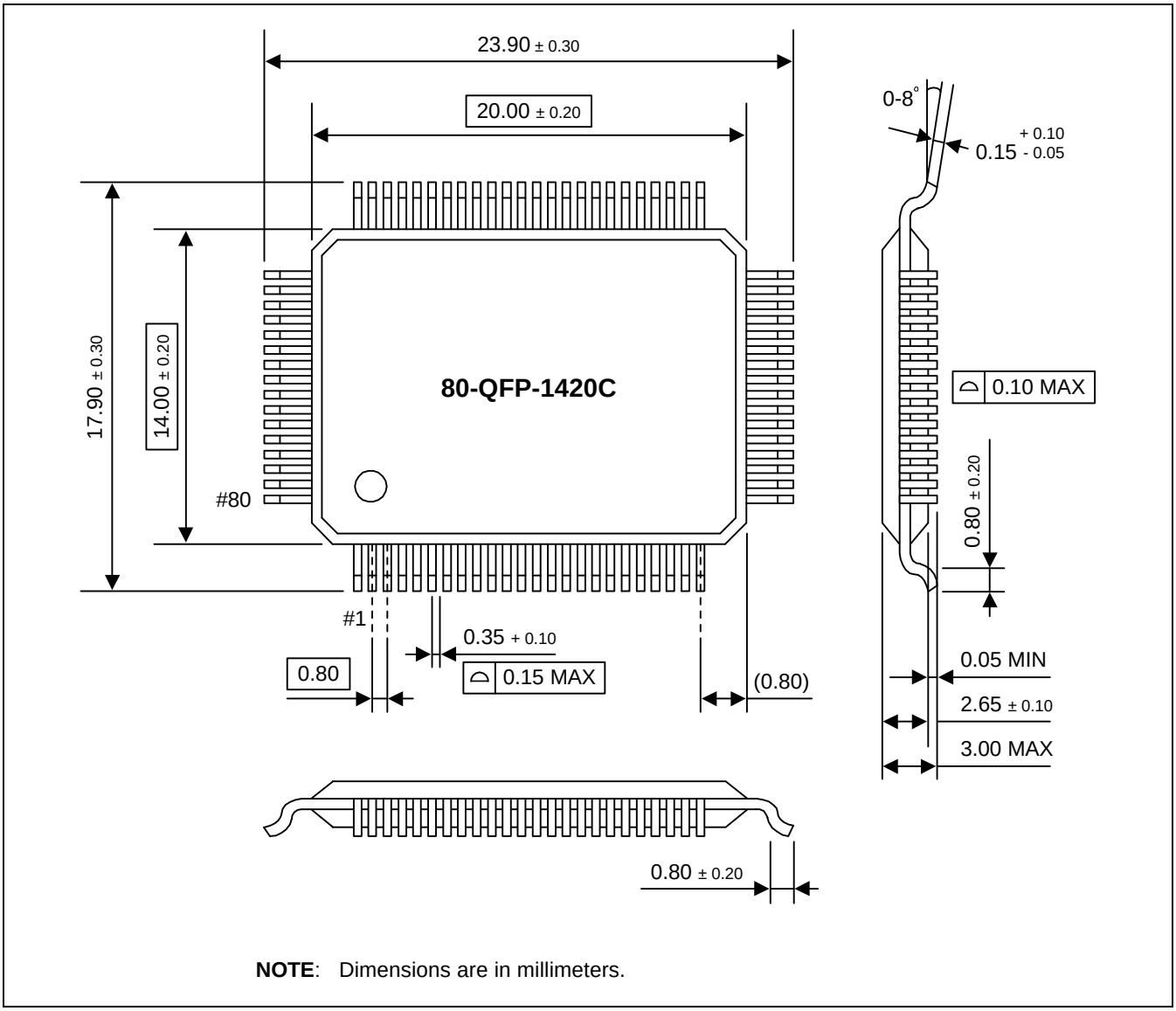
fSCK	Serial port clock cycle time
tS1	Output data setup to clock rising edge
tS2	Clock rising edge to input data valid
tH1	Output data hold after clock rising edge
tH2	Input data hold after clock rising edge

Figure 19-10. A.C. Timing Waveform for the UART Module

20 MECHANICAL DATA

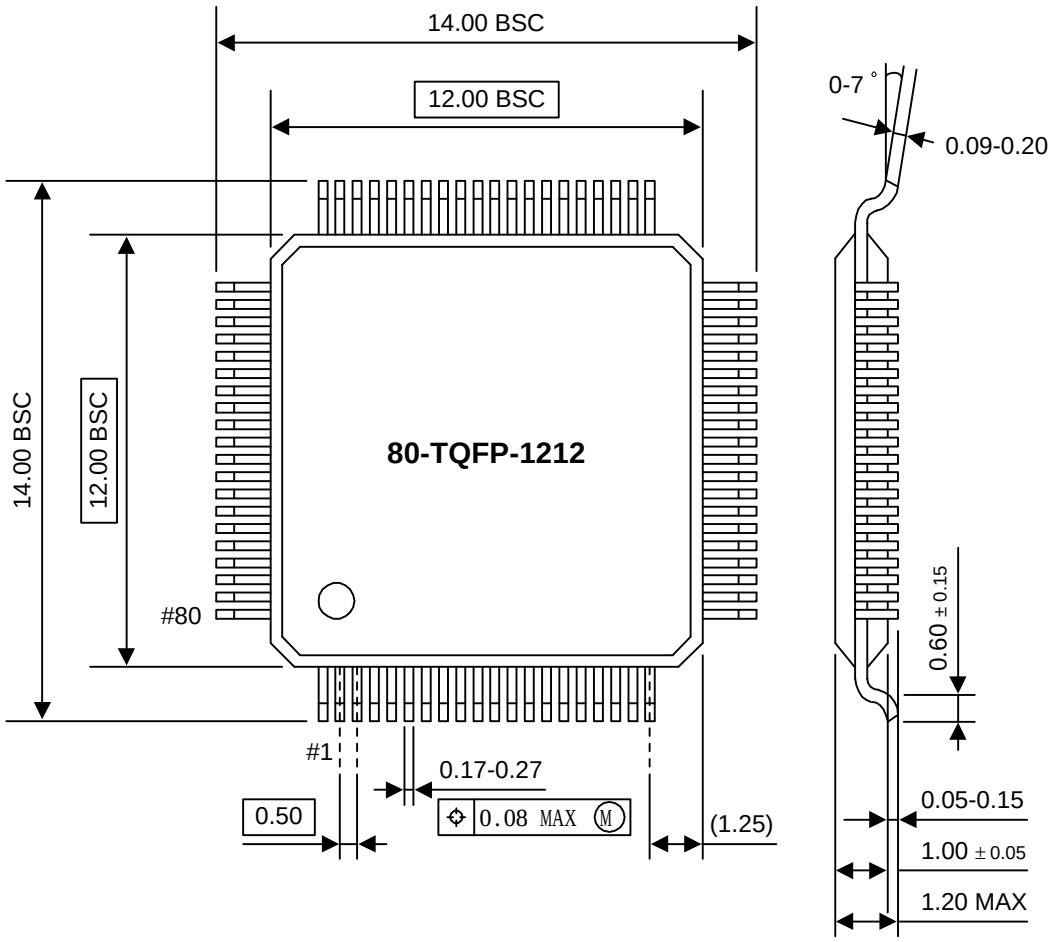
OVERVIEW

The S3C825A microcontroller is currently available in 80-pin QFP and TQFP package.



NOTE: Dimensions are in millimeters.

Figure 20-1. Package Dimensions (80-QFP-1420C)



NOTE: Dimensions are in millimeters.

Figure 20-2. Package Dimension (80-TQFP-1212)

21 S3P825A OTP

OVERVIEW

The S3P825A single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the S3C825A microcontroller. It has an on-chip OTP ROM instead of a masked ROM. The EPROM is accessed by serial data format.

The S3P825A is fully compatible with the S3C825A, both in function in D.C. electrical characteristics and in pin configuration. Because of its simple programming requirements, the S3P825A is ideal as an evaluation chip for the S3C825A.

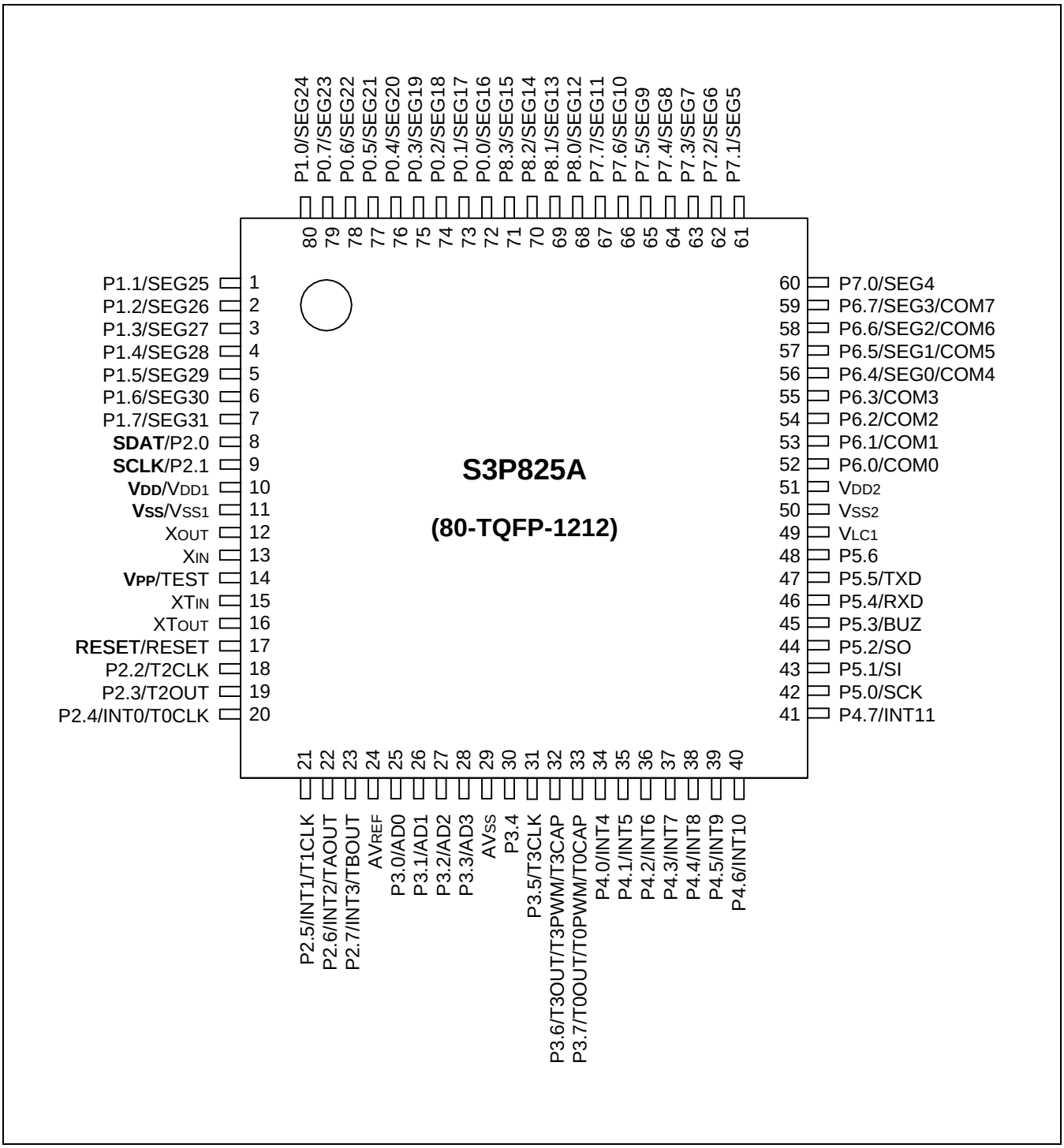


Figure 21-1. S3P825A Pin Assignments (80-Pin TQFP Package)

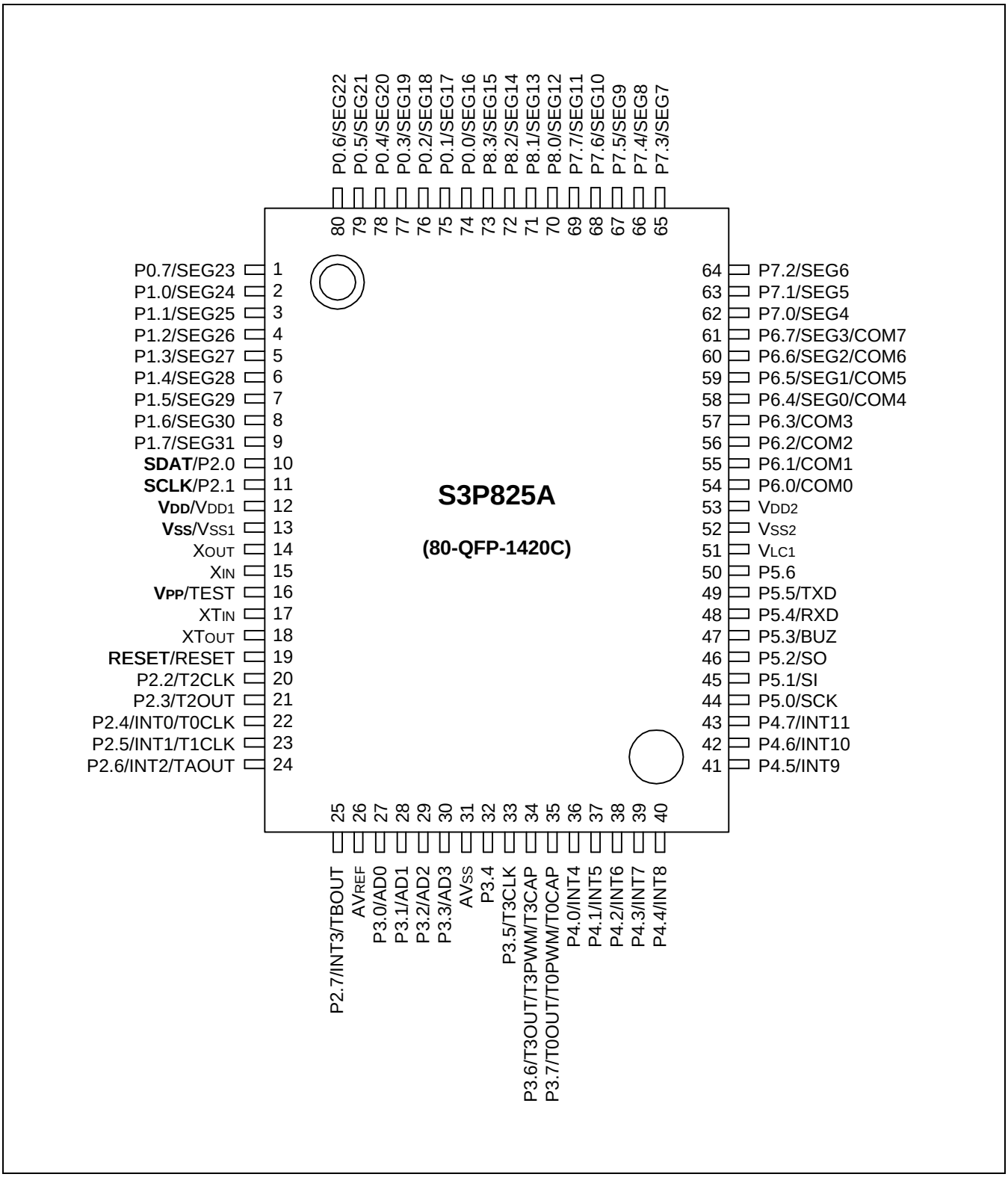


Figure 21-2. S3P825A Pin Assignments (80-Pin QFP Package)

Table 21-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip	During Programming			
Pin Name	Pin Name	Pin No.	I/O	Function
P2.0	SDAT	8(10)	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input/push-pull output port.
P2.1	SCLK	9(11)	I	Serial clock pin. Input only pin.
TEST	V_{PP}	14(16)	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is applied, OTP is in reading mode. (Option)
RESET	RESET	17(19)	I	Chip Initialization
V_{DD1}/V_{SS1}	V_{DD}/V_{SS}	10/11(12/13)	–	Logic power supply pin. VDD should be tied to +5 V during programming.

NOTE: Parentheses indicate pin for 80-pin-QFP-1420 package.

Table 21-2. Comparison of S3P825A and S3C825A Features

Characteristic	S3P825A	S3C825A
Program Memory	48-Kbyte EPROM	48-Kbyte mask ROM
Operating Voltage (V_{DD})	2.0 V to 5.5 V	2.0 V to 5.5 V
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (TEST) = 12.5 V	
Pin Configuration	80 TQFP, 80 QFP	80 TQFP, 80 QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP} (TEST) pin of the S3P825A, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 21-3 below.

Table 21-3. Operating Mode Selection Criteria

VDD	VPP (TEST)	REG/MEM	Address(A15–A0)	R/W	Mode
5 V	5 V	0	0000H	1	EPROM read
	12.5 V	0	0000H	0	EPROM program
	12.5 V	0	0000H	1	EPROM verify
	12.5 V	1	0E3FH	0	EPROM read protection

NOTE: "0" means Low level; "1" means High level.

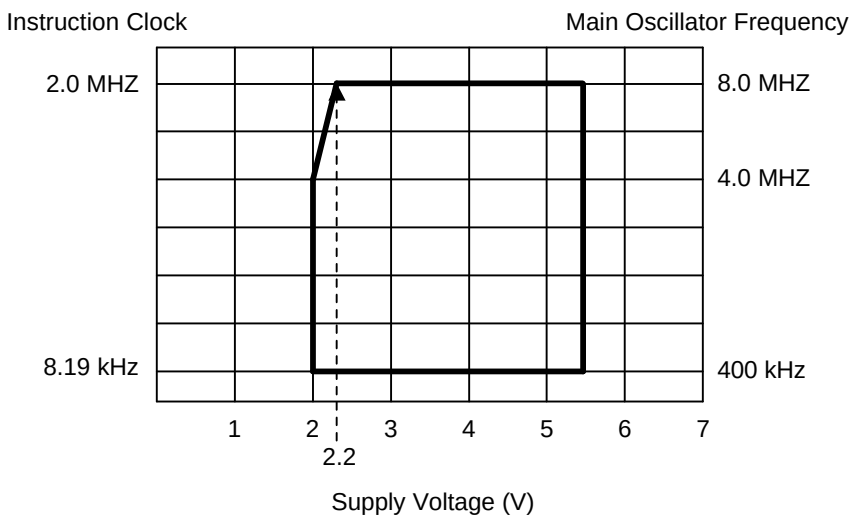


Figure 21-3. Operating Voltage Range

NOTES

22 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system in turnkey form. The development support system is configured with a host system, debugging tools, and support software. For the host system, any standard computer that operates with MS-DOS as its operating system can be used. One type of debugging tool including hardware and software is provided: the sophisticated and powerful in-circuit emulator, SMDS2+, for S3C7, S3C6, S3C8 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

SHINE

Samsung Host Interface for In-Circuit Emulator, SHINE, is a multi-window based debugger for SMDS2+. SHINE provides pull-down and pop-up menus, mouse support, function/hot keys, and context-sensitive hyper-linked help. It has an advanced, multiple-windowed user interface that emphasizes ease of use. Each window can be sized, moved, scrolled, highlighted, added, or removed completely.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generates object code in standard hexadecimal format. Assembled program code includes the object code that is used for ROM data and required SMDS program control data. To assemble programs, SAMA requires a source file and an auxiliary definition (DEF) file with device specific information.

SASM88

The SASM88 is a relocatable assembler for Samsung's S3C8-series microcontrollers. The SASM88 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM88 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces the relocatable object code only, so the user should link object file. Object files can be linked with other object files and loaded into memory.

HEX2ROM

HEX2ROM file generates ROM code from HEX file which has been produced by assembler. ROM code must be needed to fabricate a microcontroller which has a mask ROM. When generating the ROM code (.OBJ file) by HEX2ROM, the value "FF" is filled into the unused ROM area up to the maximum ROM size of the target device automatically.

TARGET BOARDS

Target boards are available for all S3C8-series microcontrollers. All required target system cables and adapters are included with the device-specific target board.

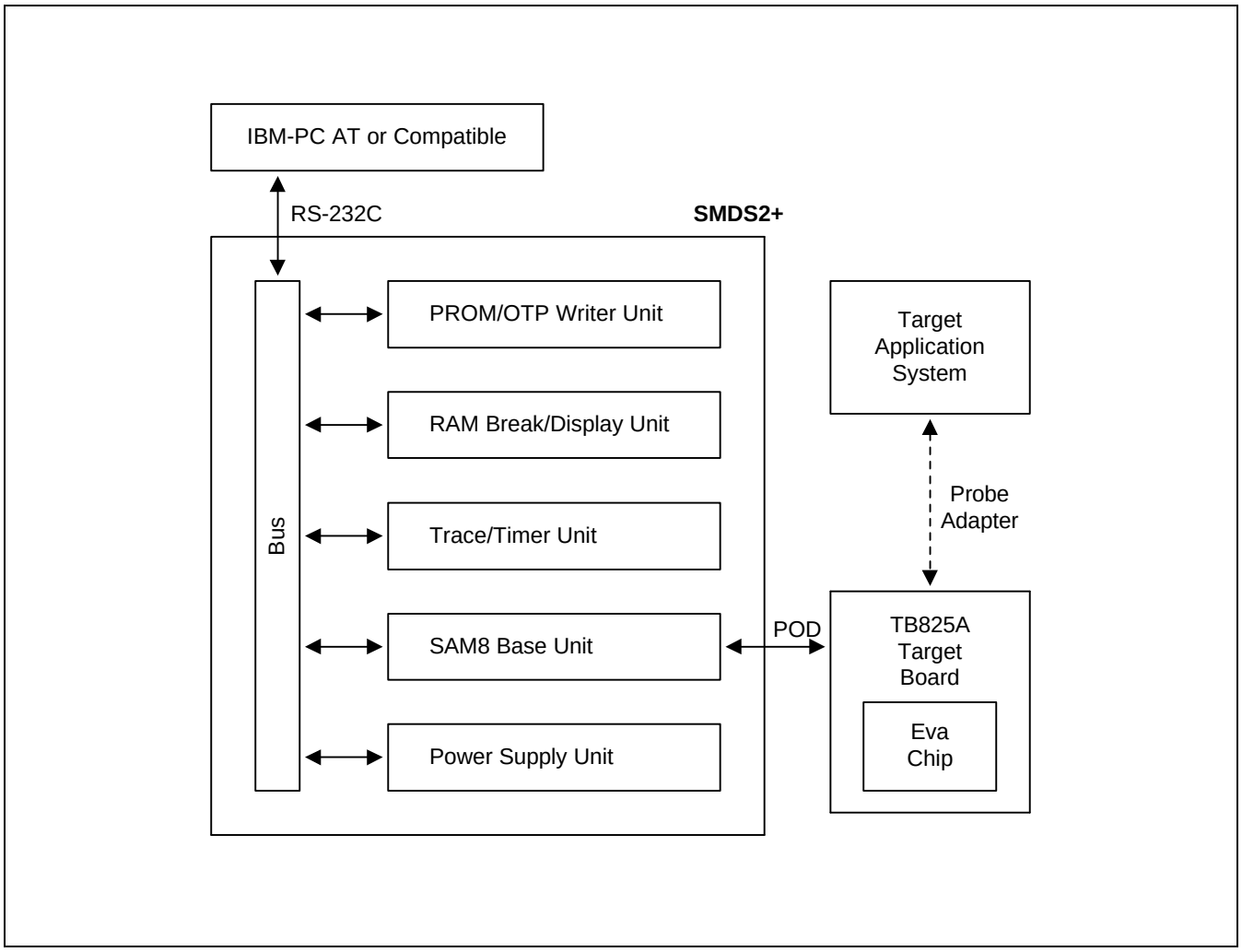


Figure 22-1. SMDS Product Configuration (SMDS2+)

TB825A TARGET BOARD

The TB825A target board is used for the S3C825A microcontroller. It is supported with the SMDS2+.

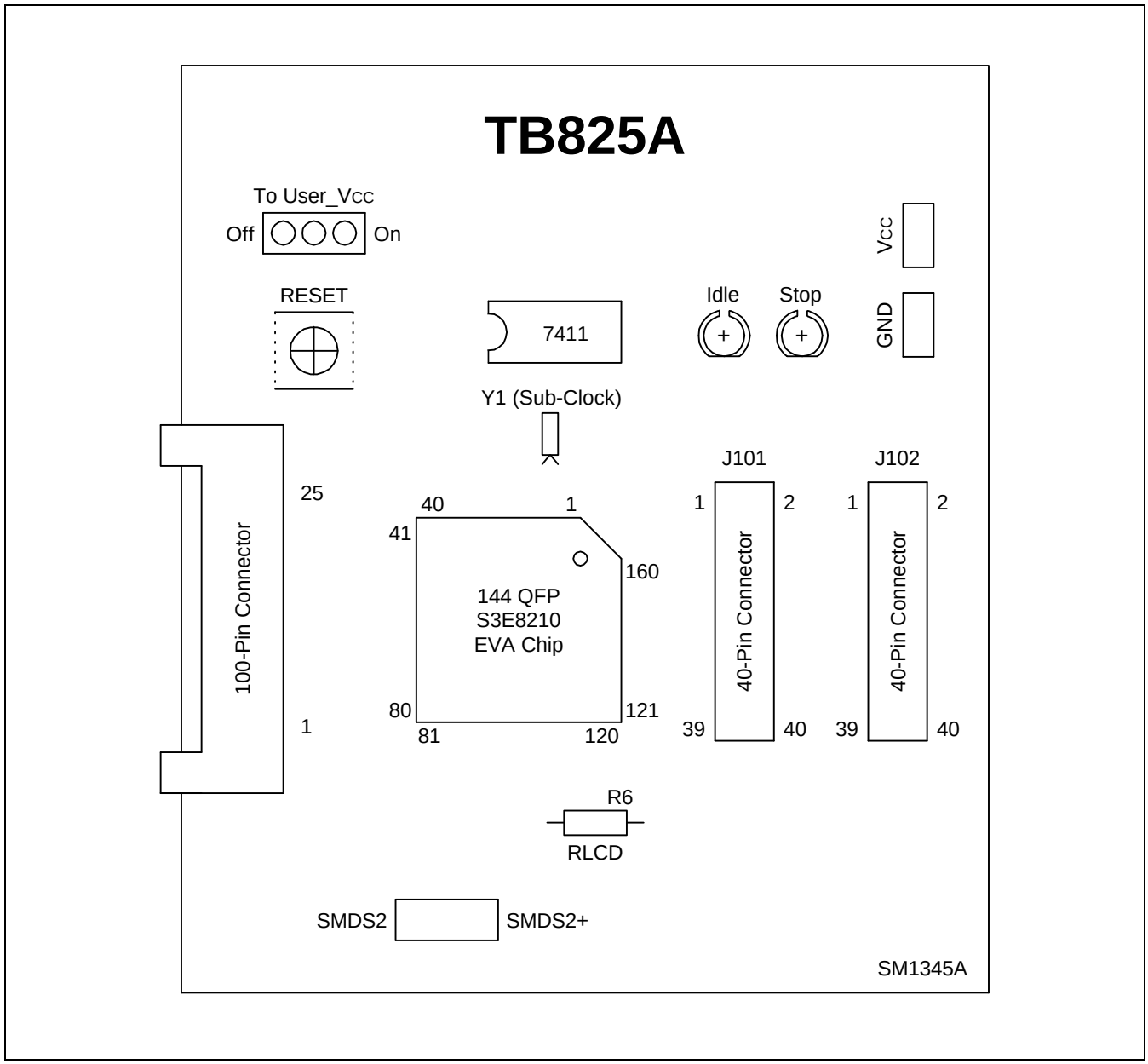
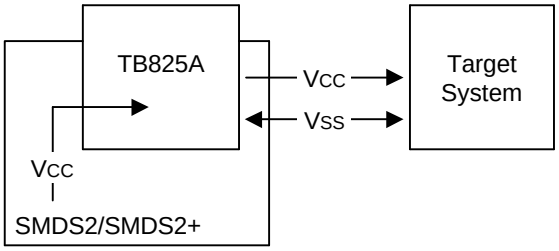
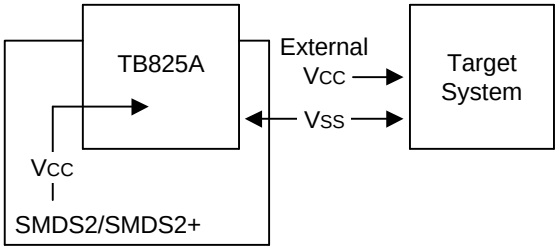


Figure 22-2. TB825A Target Board Configuration

Table 22-1. Power Selection Settings for TB825A

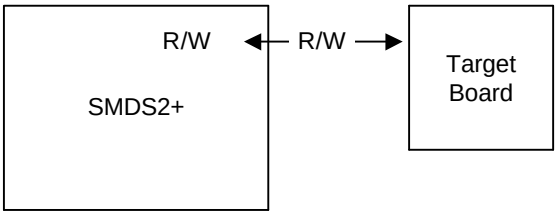
"To User_V_{CC}" Settings	Operating Mode	Comments
To User_Vcc Off ○ ● ● On		The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.
To User_Vcc Off ● ● ○ On		The SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

NOTE: The following symbol in the "To User_Vcc" Setting column indicates the electrical short (off) configuration:

SMDS2+ Selection (SAM8)

In order to write data into program memory that is available in SMDS2+, the target board should be selected to be for SMDS2+ through a switch as follows. Otherwise, the program memory writing function is not available.

Table 22-2. The SMDS2+ Tool Selection Setting

"SW1" Setting	Operating Mode
SMDS2 ○ ● ● SMDS2+	

IDLE LED

The Yellow LED is ON when the evaluation chip (S3E8250) is in idle mode.

STOP LED

The Red LED is ON when the evaluation chip (S3E8250) is in stop mode.

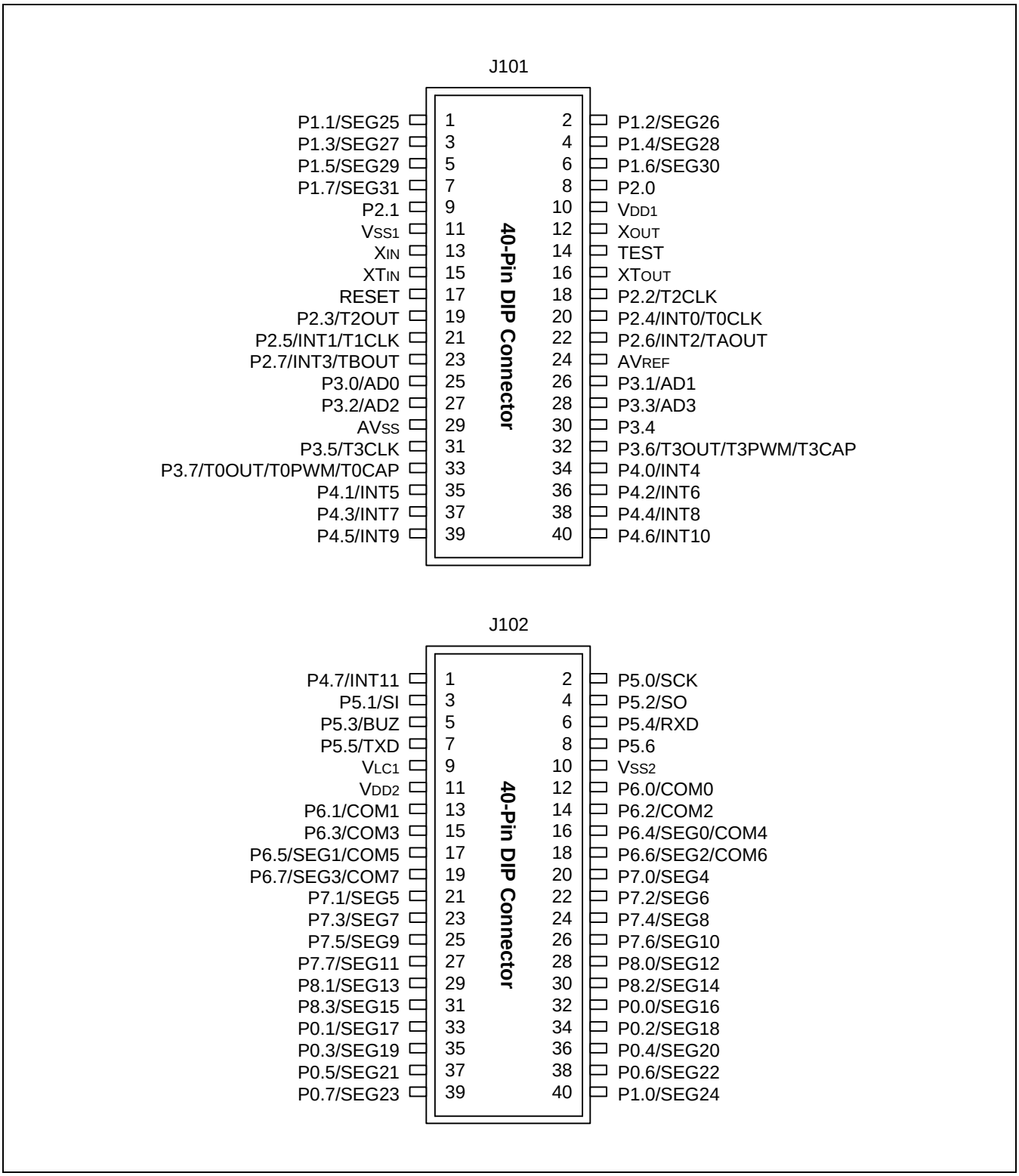


Figure 22-3. 40-Pin Connectors (J101, J102) for TB825A

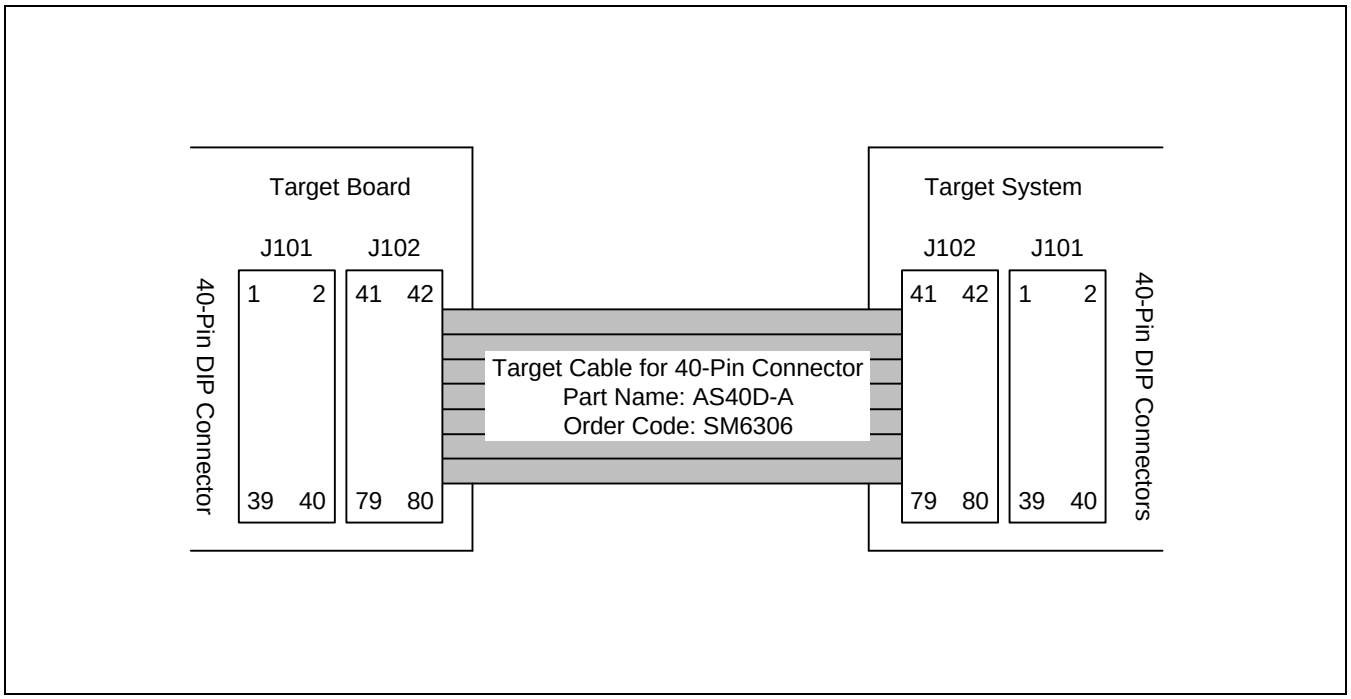


Figure 22-4. S3C825A Cables for 80-TQFP Package