S3C9498/F9498

8-BIT CMOS

MICROCONTROLLER

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.

S3C9498/F9498 8-Bit CMOS Microcontroller
User's Manual, Revision 1
Publication Number: 21-S3-C9498/F9498-102004

Samsung Electronics' microcontroller business has been awarded full ISO-14001 certification (BVQ1 Certificate No. 9330). 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 449-900

TEL: (82)-(031)-209-1934
FAX: (82) (331) 209-1889

Home-Page URL: Http://www.samsungsemi.com/
Printed in the Republic of Korea

Preface

The *S3C9498/F9498 Microcontroller User's Manual* is designed for application designers and programmers who are using the S3C9498/F9498 microcontroller for application development. It is organized in two 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	SAM88RCRI Instruction Set

Chapter 1, "Product Overview," is a high-level introduction to the S3C9498/F9498 with a general product description, and detailed information about individual pin characteristics and pin circuit types.

Chapter 2, "Address Spaces," explains the S3C9498/F9498 program and data memory, internal register file, and mapped control registers, and explains how to address them. Chapter 2 also describes working register addressing, as well as system and user-defined stack operations.

Chapter 3, "Addressing Modes," contains detailed descriptions of the six addressing modes that are supported by the 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 standard 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 S3C9498/F9498 interrupt structure in detail and further prepares you for additional information presented in the individual hardware module descriptions in part II.

Chapter 6, "SAM88RCRI Instruction Set," describes the features and conventions of the instruction set used for all S3C9-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 SAM88RCRI product 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 contains detailed information about the peripheral components of the S3C9498/F9498 microcontrollers. Also included in part II are electrical, mechanical, MTP, and development tools data. It has 14 chapters:

Chapter 7	Clock Circuit	Chapter 14	UART
Chapter 8	RESET and Power-Down	Chapter 15	Serial I/O Interface
Chapter 9	I/O Ports	Chapter 16	PWM
Chapter 10	Basic Timer	Chapter 17	ADC
Chapter 11	8-bit Timer A/B	Chapter 18	Electrical Data
Chapter 12	16-bit Timer 1	Chapter 19	Mechanical Data
Chapter 13	Timer 0	Chapter 20	MTP
		Chapter 21	Development Tools

Two order forms are included at the back of this manual to facilitate customer order for S3C9498/F9498 microcontroller: 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 SAM88RCRI 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

Table of Contents (Continued)

Chapter 11 8-bit Timer A/B

Chapter 12 16-bit Timer 1

Chapter 13 Timer 0

Table of Contents (Continued)

Chapter 14 UART

Chapter 15 Serial I/O Interface

Chapter 16 PWM

Chapter 17 A/D Converter

Table of Contents (Continued)

List of Figures

List of Figures (Continued)

List of Figures (Continued)

List of Figures (Continued)

List of Tables

List of Tables (Continued)

List of Register Descriptions

List of Instruction Descriptions

1 PRODUCT OVERVIEW

S3C9-SERIES MICROCONTROLLERS

Samsung's SAM88RCRI family of 8-bit single-chip CMOS microcontrollers offer a fast and efficient CPU, a wide range of integrated peripherals, and various mask-programmable ROM sizes.
A address/data bus architecture and a large number of bit-configurable I/O ports provide a flexible programming environment for applications with varied memory and I/O requirements. Timer/counters with selectable operating modes are included to support real-time operations.

S3C9498/F9498 MICROCONTROLLER

The S3C9498/F9498single-chip CMOS microcontrollers are fabricated using the highly advanced CMOS process technology based on Samsung's latest CPU architecture.

The S3C9498/F9498 is a microcontroller with a 8K-byte multi time programmable ROM embedded.

Using a proven modular design approach, Samsung engineers have successfully developed the S3C9498/F9498 by integrating the following peripheral modules with the powerful SAM88 RCRI core:

— Four configurable I/O ports (22 pins / 24pins / 26pin)
— Fifteen interrupt sources with one vector and one interrupt level
— One watchdog timer function (Basic Timer overflow)
— One 8-bit basic timer for oscillation stabilization
— Four 8-bit timer/counter with time interval, PWM, and Capture mode
(Timer C and Timer D can be used for 16-bit Timer 0)
— One 16-bit timer/counter with three operating modes; Interval timer, Capture and PWM mode
(If Timer C and Timer D is used for Timer 0, S3C9498/F9498 has two 16-bit Timer; Timer 0 and Timer 1)
— Analog to digital converter with 8 input channels and 10-bit resolution
— One asynchronous UART and one synchronous SIO

The S3C9498/F9498 microcontroller is ideal for use in a wide range of home applications requiring simple timer/counter, ADC, etc. They are currently available in 32-pin SOP/SDIP, 28-pin SOP, 30-pin SDIP package.

MTP

The S3C9498/F9498 has on-chip 8-Kbyte multi time programmable (MTP) ROM instead of masked ROM. The S3C9498/F9498 is fully compatible to the S3C9498, in function, in D.C. electrical characteristics and in pin configuration.

FEATURES

CPU

- SAM88RCRI CPU core

Memory

- 208-byte general purpose register (RAM)
- 8K-byte internal mask program memory
- 8K-byte internal multi time program memory (S3C9498/F9498) Half-Flash

Oscillation Sources

- Crystal, Ceramic
- CPU clock divider (1/1, 1/2, 1/8, 1/16)

Instruction Set

- 41 instructions
- IDLE and STOP instructions added for power-down modes

Instruction Execution Time

- 500 ns at 8-MHz f_{OSC} (minimum)

Interrupts

- 15 interrupt sources with one vector / one level

I/O Ports

- Total 22/24/26 bit-programmable pins

Basic Timer

- One programmable 8-bit basic timer (BT) for Oscillation stabilization control

Timers

- One 8-bit timer/counter (**Timer A**) with three operating modes; Interval mode, capture mode and PWM mode
- One 8-bit timer/counter (**Timer B**) Carrier frequency (or PWM) generator
- One 16-bit capture timer/counter (**Timer 1**) with three operating modes; Interval mode, Capture mode for pulse period or duty and PWM mode.

.

Timer 0 Timer/Counters

- The programmable 8-bit timer/counters
- Configurable as on 16-bit timer/counters

PWM module

- 12-bit PWM (Max: 250KHz)
- 6-bit base + 6-bit extension frame

A/D Converter

- Eight analog input channels
- 25us conversion speed at 8MHz f_{ADC} clock

Serial I/O

- One synchronous serial I/O module
- Selectable transmit and receive rates.

Asynchronous UART

- Programmable baud rate generator
- Support serial data transmit/receive operations with 8-bit, 9-bit UART

Low Voltage Reset (LVR)

- Low Voltage Check to make system reset
- V_{LVR} = 3V (by smart option)

Operating Temperature Range

- –25 °C to + 85 °C

Operating Voltage Range

- 3 V to 5.5 V (LVR)
  2.2 V to 5.5V (No LVR)

Package Type

- 28-pin SOP, 30-pin SDIP, 32-pin SDIP/SOP

BLOCK DIAGRAM

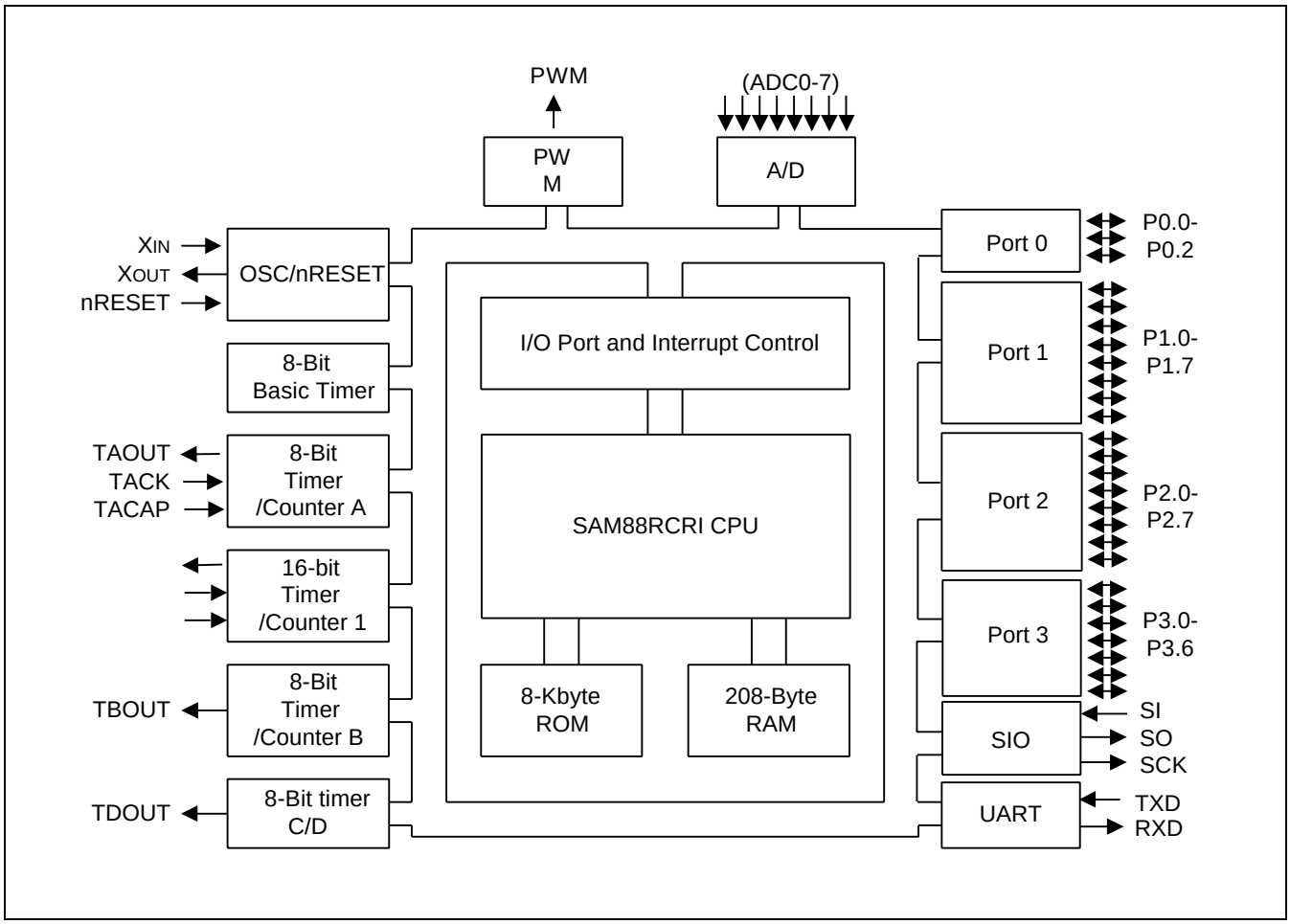


Figure 1-1. S3C9498/F9498 Block Diagram

PIN ASSIGNMENT

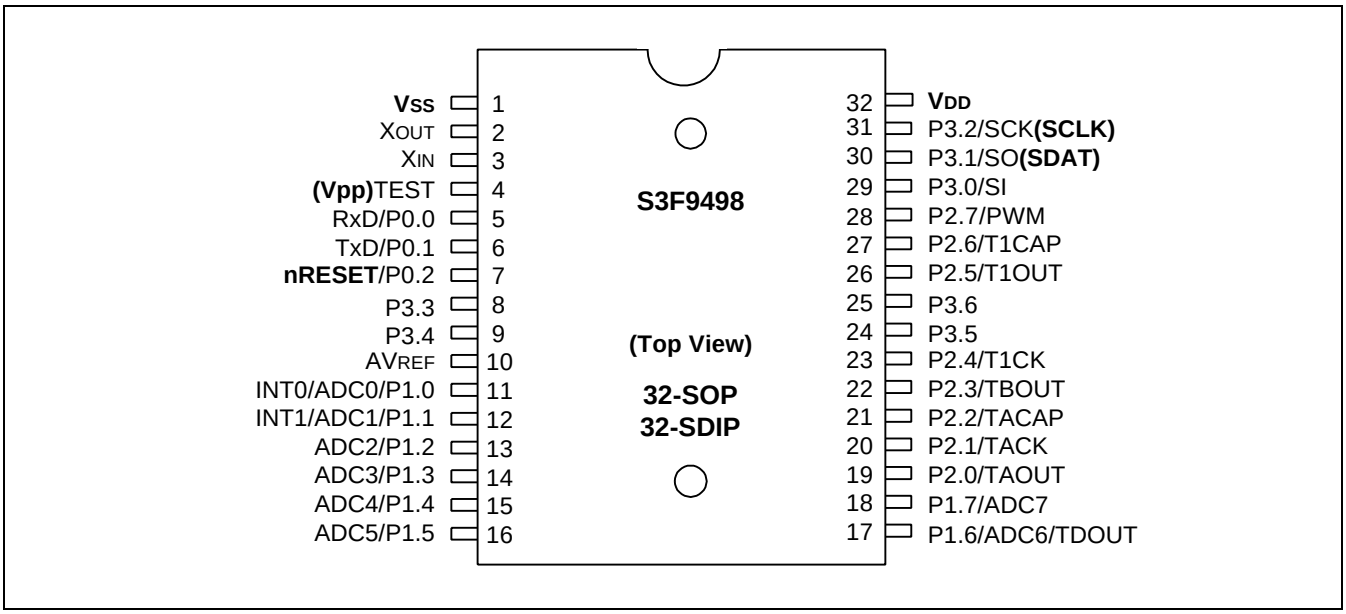


Figure 1-2. S3C9498/F9498 Pin Assignment (32-DIP, 32-SOP)

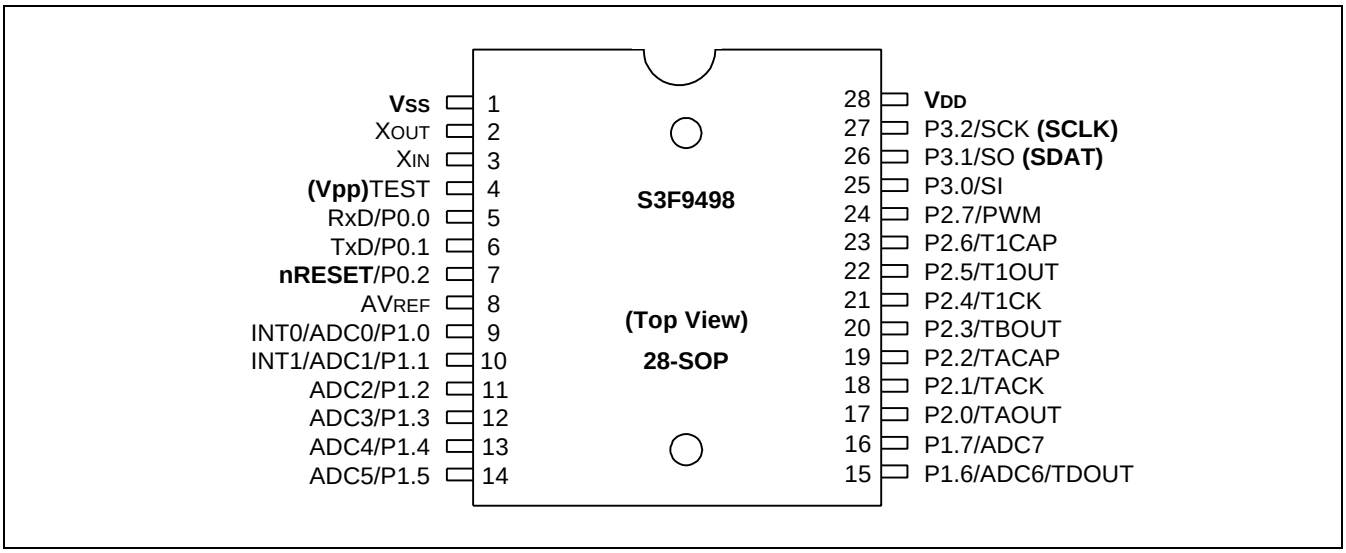


Figure 1-3. S3C9498/F9498 Pin Assignment (28-SOP)

PIN ASSIGNMENT

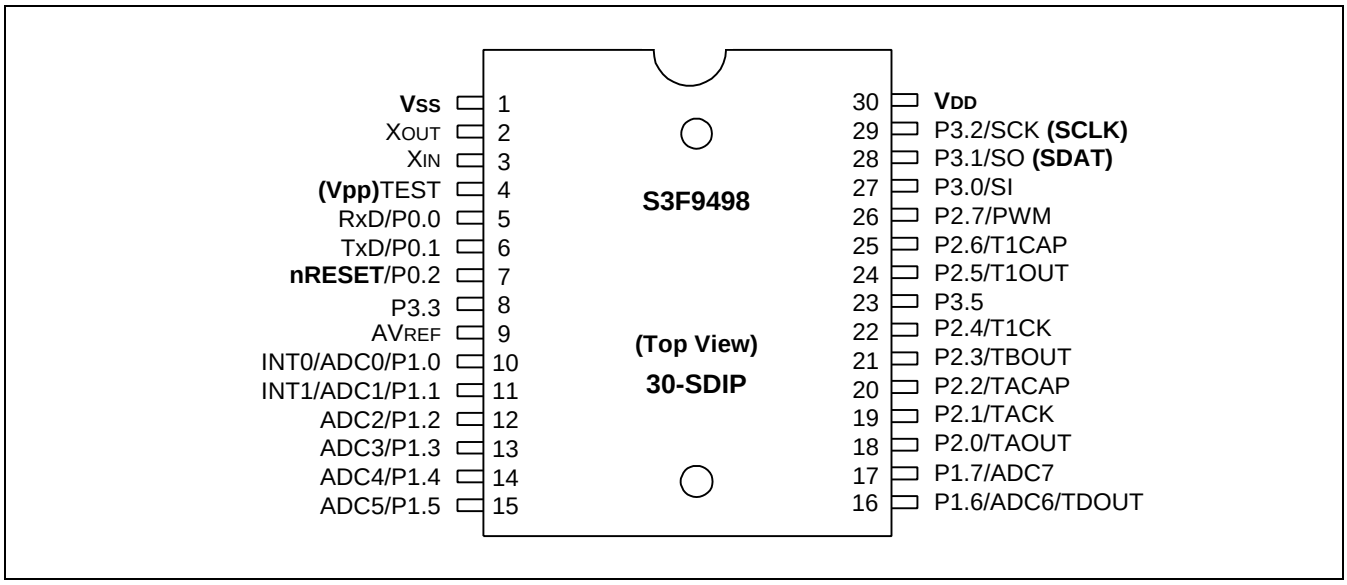


Figure 1-4. S3C9498/F9498 Pin Assignment (30-SDIP)

PIN DESCRIPTIONS

Table 1-1. Pin Descriptions of 28-SOP (32-SOP,32-SDIP / 30-SDIP)

Pin Names	Pin Type	Pin Description	Circuit Type	28 Pin No.	Shared Functions
P0.0, P0.1 P0.2	I/O	I/O port with bit-programmable pins. Configurable to input or push-pull output mode. Pull-up resistors can be assigned by software. Pins can also be assigned individually as alternative function pins.	D-1 E-2	5–7	RxD, TxD RESETB
P1.0 – P1.1 P1.2 – P1.7	I/O	I/O port with bit-programmable pins. Configurable to input or push-pull output mode. Pull-up resistors can be assigned by software. Pins can also be assigned individually as alternative function pins.	D-2 E-1	9–16 (11–18 / 11- 18)	INT0–INT1 ADC0–ADC7 TDOUT
P2.0,P2.3 P2.5,P2.7 P2.1-P2.2 P2.4,P2.6	I/O	I/O port with bit-programmable pins. Configurable to input mode, push-pull output mode, or n-channel open-drain output mode. Input mode with pull-up resistors can be assigned by software. The port 2 pins have high current drive capability. Pins can also be assigned individually as alternative function pins.	D-1 E	17–24 (19–23, 26–28 / 19–23, 24–26)	TAOUT/TACK TACAP TBOUT T1CAP/T1CK T1OUT PWM
P3.0–P3.2 (P3.3–P3.6)	I/O	I/O port with bit-programmable pins. Configurable to input or push-pull output mode. Pull-up resistors can be assigned by software. Pins can also be assigned individually as alternative function pins.	D-1 (E)	25–27 (29–31, 8-9, 24-25 / 27–29, 8, 23)	SI/SO/SCK
X_{IN}, X_{OUT}	I, O	System clock input and output pins	–	2,3	–
TEST	I	Test signal input pin (for factory use only; must be connected to V_{SS}.)	–	4	–
V_{DD}	–	Power supply input pin	–	28 (32/30)	–
V_{SS}	–	Ground pin	–	1	–

Table 1-2. Pin Descriptions of 28-SOP (32-SOP,32-SDIP / 30-SDIP)

Pin Names	Pin Type	Pin Description	Circuit Type	28 Pin No.	Shared Functions
SCK	I/O	Serial interface clock input or output	D-1	27(31/29)	P3.2
SO	O	Serial data output	D-1	26(30/28)	P3.1
SI	I	Serial data output	D-1	25(29/27)	P3.0
PWM	O	PWM output	D-1	24(28/26)	P2.7
ADC0–7	I	A/D converter analog input channels	E-1	9-16 (11-18 / 10-17)	P1.0–P1.7
AVREF	I	A/D converter reference voltage		8(10/9)	
RxD	I/O	Serial data RXD pin for receive input and transmit output (mode 0)	D-1	5	P0.0
TxD	O	Serial data TXD pin for transmit output and shift clock output (mode 0)	D-1	6	P0.1
INT0 INT1	I	External interrupts.	D-2	9–10 (11,12 /10,11)	P1.0 P1.1
TAOUT	O	Timer/counter(A) match output, or Timer/counter(A) PWM output	D-1	17(19/18)	P2.0
TACK	I	Timer/counter(A) external clock input	E	18(20/19)	P2.1
TACAP	I	Timer/counter(A) external capture input	E	19(21/20)	P2.2
T1OUT	O	Timer/counter(0) match output, or Timer/counter(0) PWM output	D-1	22(26/24)	P2.5
T1CK	I	Timer/counter(0) external clock input	E	21(23/22)	P2.4
T1CAP	I	Timer/counter(0) external capture input	E	23(27/25)	P2.6
TBOUT	O	Timer/counter(B) match output, or Timer/counter(B) PWM output	D-1	20(22/21)	P2.3
TDOUT	O	Timer/counter(B) match output, or Timer/counter(B) PWM output	D-1	15(17/16)	P1.6 ADC6
RESETB	I	System reset signal input pin	B	7	P0.2

PIN CIRCUITS

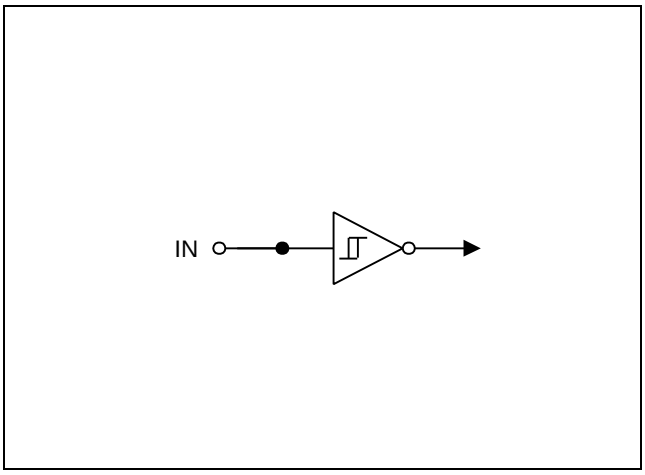


Figure 1-5. Pin Circuit Type B (nRESET)

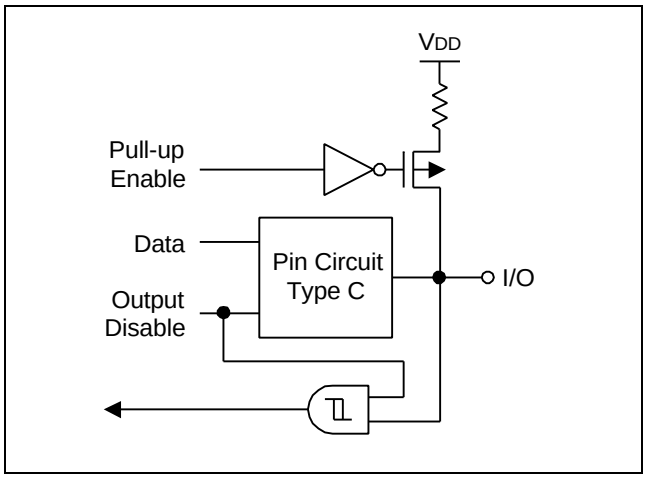


Figure 1-6. Pin Circuit Type D-1

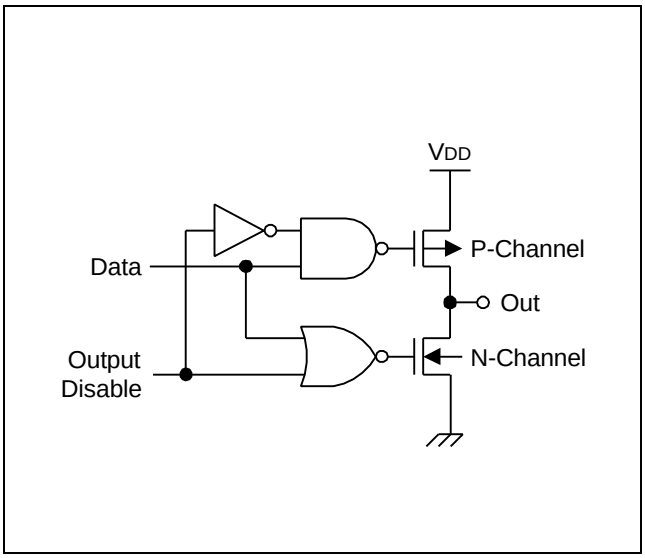


Figure 1-7. Pin Circuit Type C

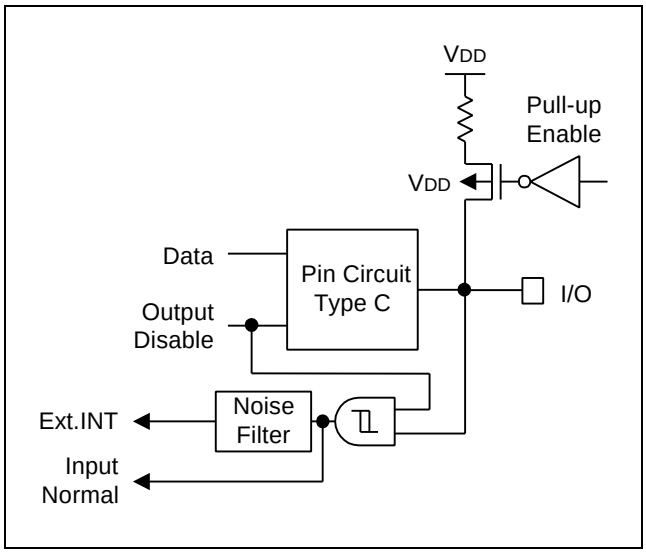


Figure 1-8. Pin Circuit Type D-2

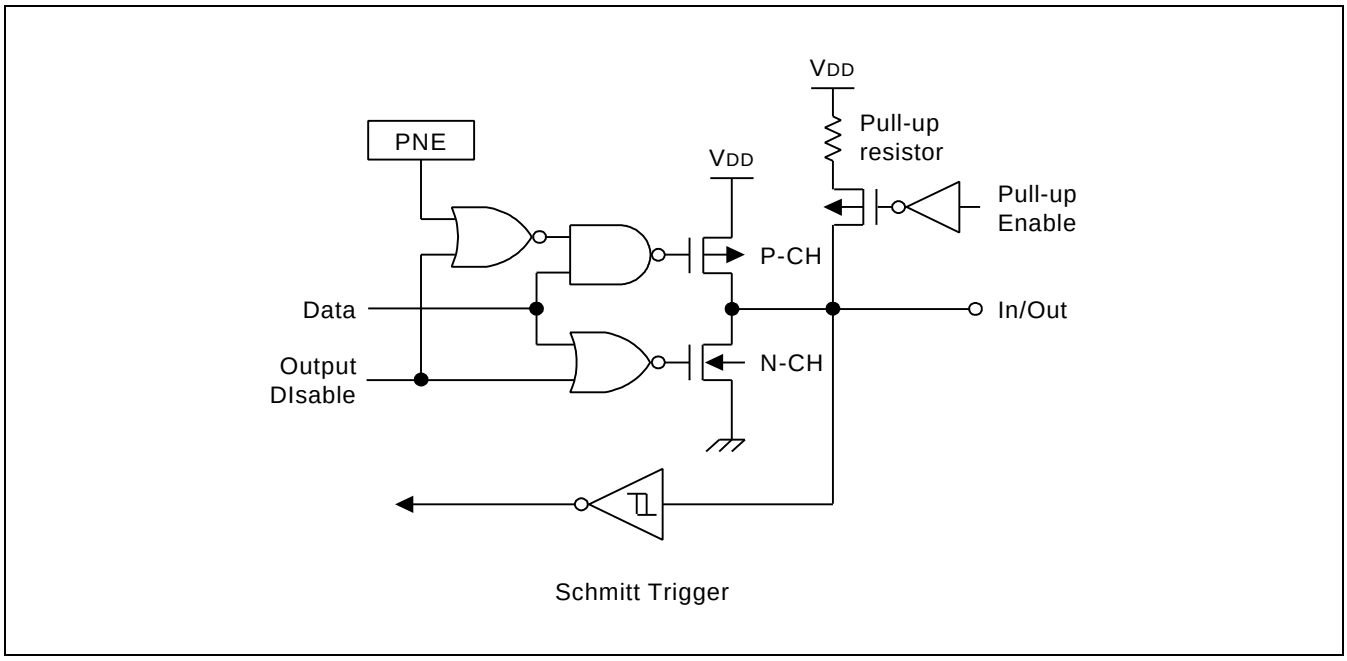


Figure 1-9. Pin Circuit Type E

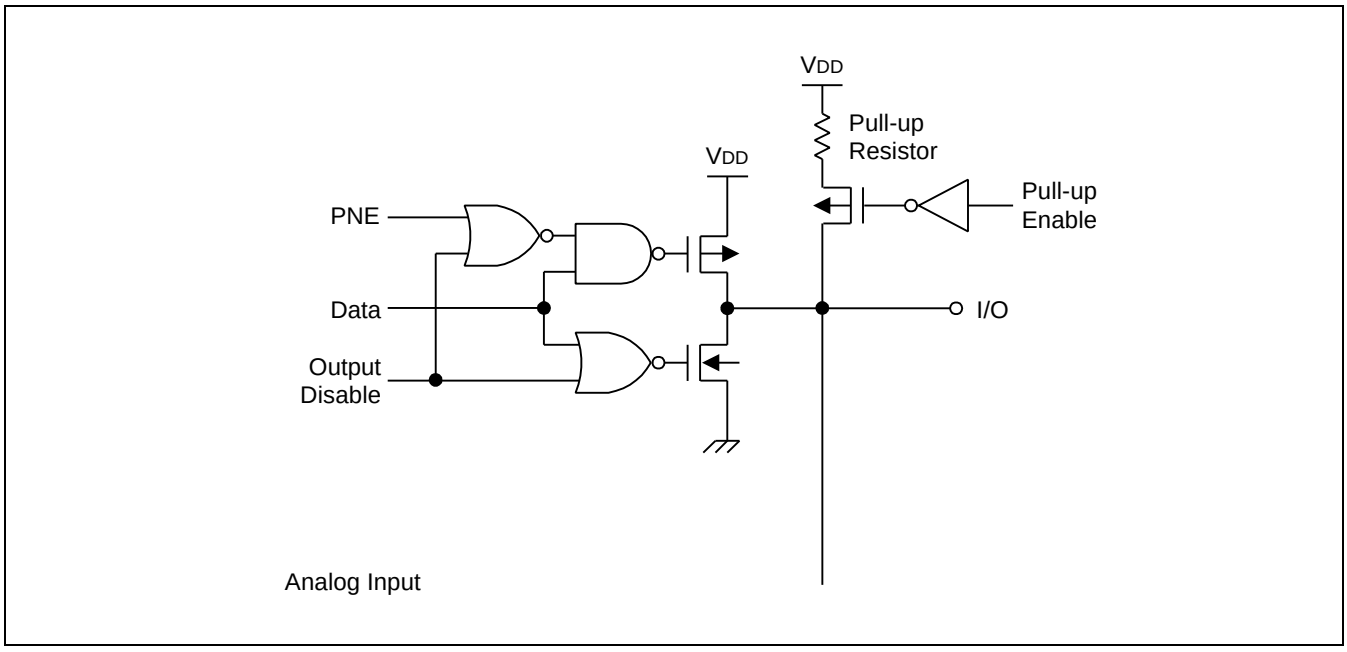


Figure 1-9. Pin Circuit Type E-1

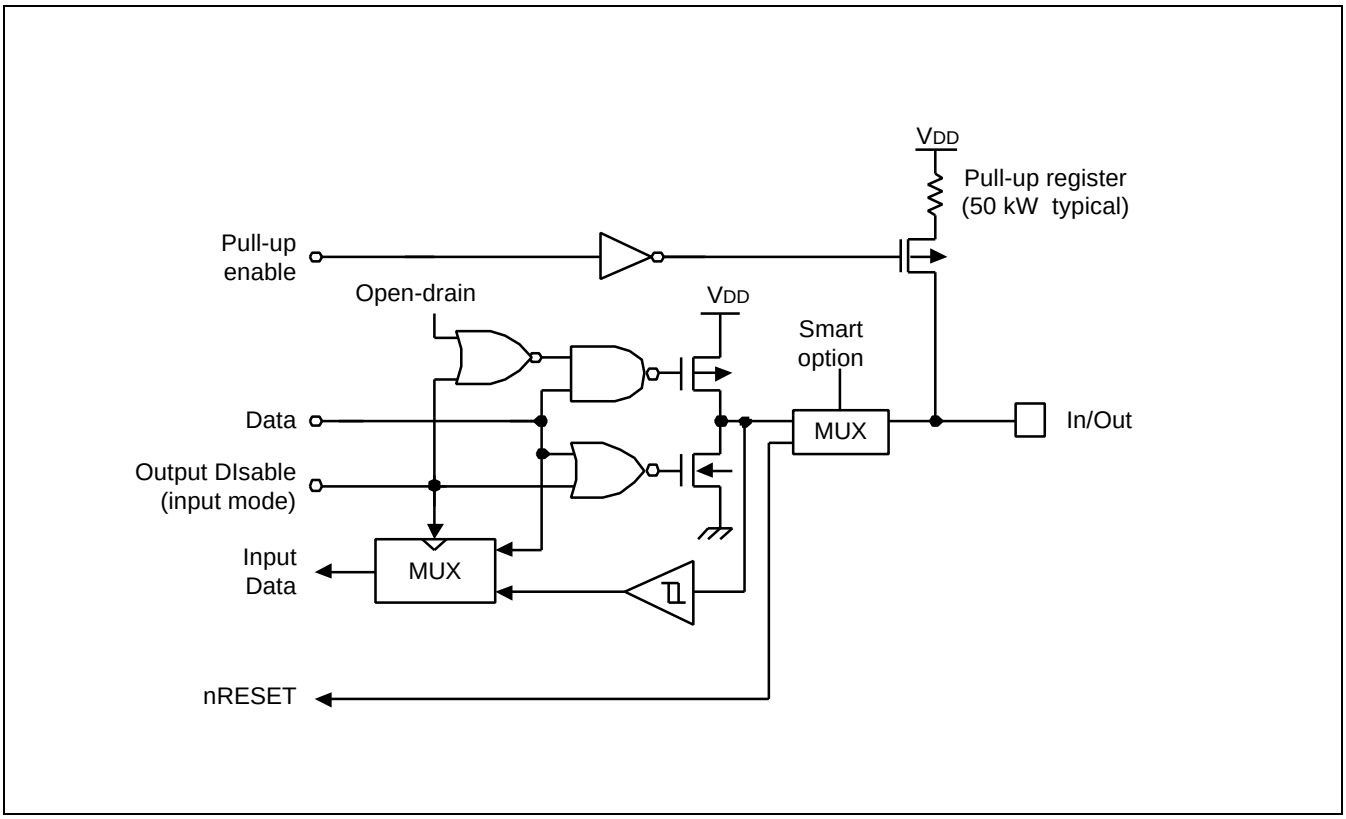


Figure 1-11. Pin Circuit Type E-2

2 ADDRESS SPACES

OVERVIEW

The S3C9498/F9498 microcontroller has two kinds of address space:

— Internal program memory (ROM)

— Internal register file

A 13-bit address bus supports program memory operations. A separate 8-bit register bus carries addresses and data between the CPU and the internal register file.

The S3C9498/F9498 have 8-Kbytes of on-chip program memory, which is configured as the Internal ROM mode, all of the 8-Kbyte internal program memory is used.

The S3C9498/F9498 microcontroller has 208 general-purpose registers in its internal register file. 45 bytes in the register file are mapped for system and peripheral control functions.

PROGRAM MEMORY (ROM)

Program memory (ROM) stores program codes or table data. The S3C9498/F9498 have 8Kbytes of internal multi time programmable (MTP) program memory (see Figure 2-1).

The first 2-bytes of the ROM (0000H–0001H) are interrupt vector address.
Unused locations (0002H–00FFH except 3CH, 3DH, 3EH, and 3FH) can be used as normal program memory.
The location 3CH, 3DH, 3EH, and 3FH is used as smart option ROM cell.

The program reset address in the ROM is 0100H.

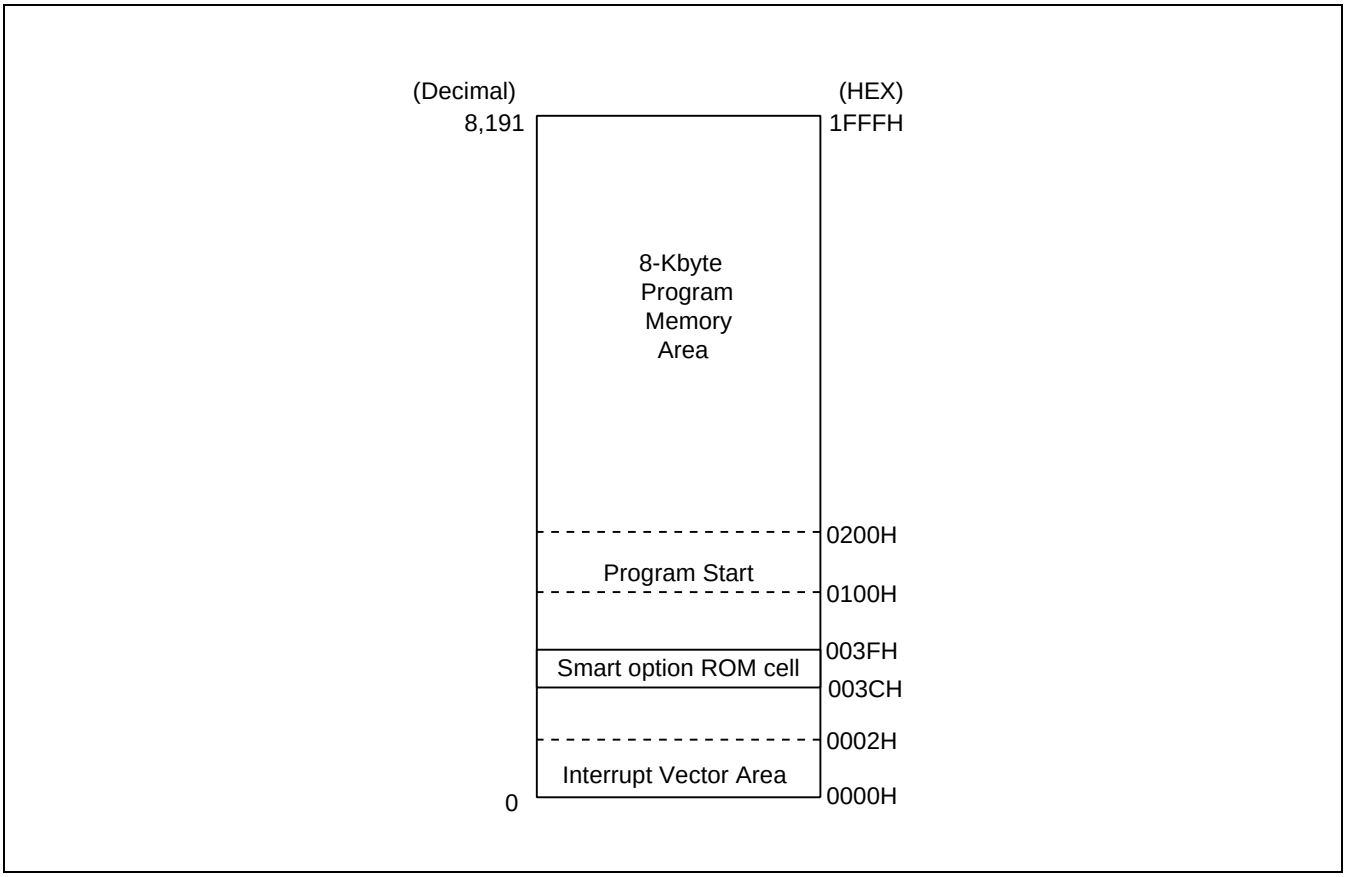
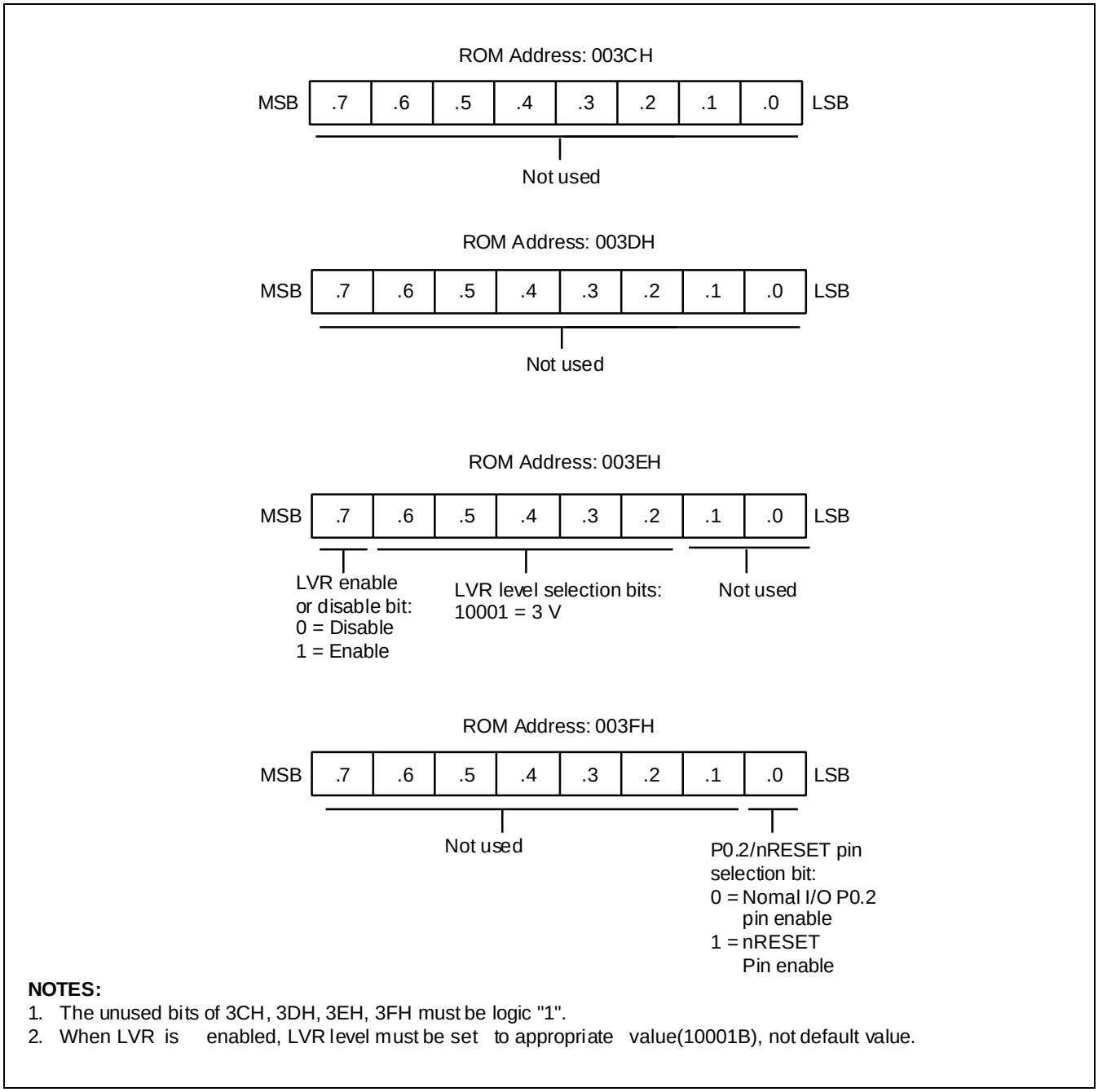


Figure 2-1. Program Memory Address Space

Smart Option

Smart option is the ROM option for starting condition of the chip. The ROM addresses used by smart option are from 003CH to 003FH. The default value of ROM is FFH.



NOTES:

1. The unused bits of 3CH, 3DH, 3EH, 3FH must be logic "1".
2. When LVR is enabled, LVR level must be set to appropriate value(10001B), not default value.

Figure 2-2. Smart Option

REGISTER ARCHITECTURE

The upper 64-bytes (C0H-FFH) of the S3C9498/F9498 internal register file are addressed as working registers, system control registers and peripheral control registers. The lower 192-bytes of internal register file (00H–BFH) is called the *general-purpose register space*. 253 registers in this space can be accessed; 208 are available for general-purpose use.

For many SAM88RCRI microcontrollers, the addressable area of the internal register file is further expanded by additional register pages at space of the general purpose register (00H–BFH). This register file expansion is not implemented in the S3C9498/F9498, however.

The specific register types and the area (in bytes) that they occupy in the internal register file are summarized in Table 2-1.

Table 2-1. Register Type Summary

Register Type	Number of Bytes
System and peripheral registers	45
General-purpose registers (including the 16-bit common working register area)	208
Total Addressable Bytes	**253**

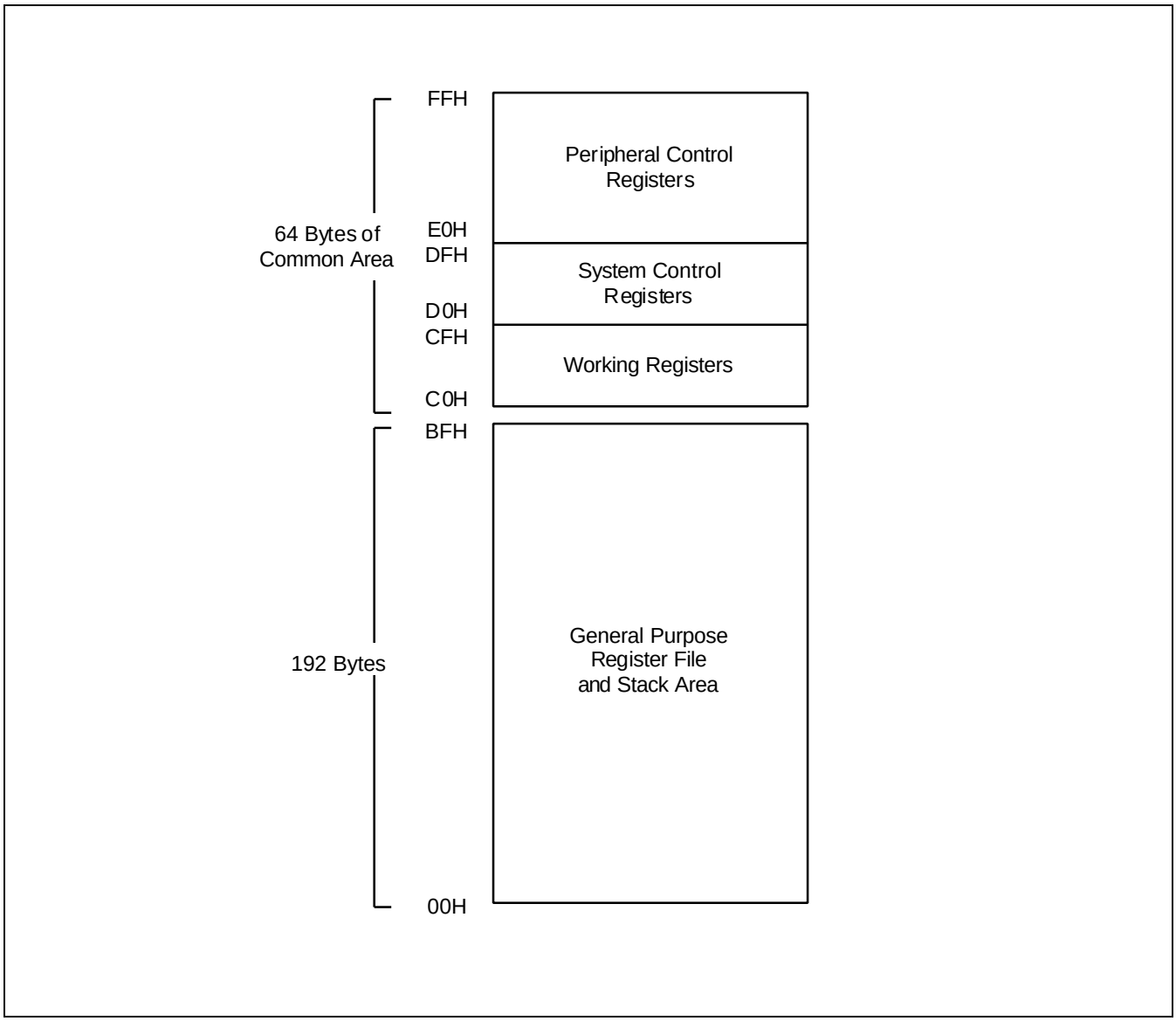


Figure 2-3. Internal Register File Organization

COMMON WORKING REGISTER AREA (C0H–CFH)

The SAM88RCRI register architecture provides an efficient method of working register addressing that takes full advantage of shorter instruction formats to reduce execution time.

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.

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

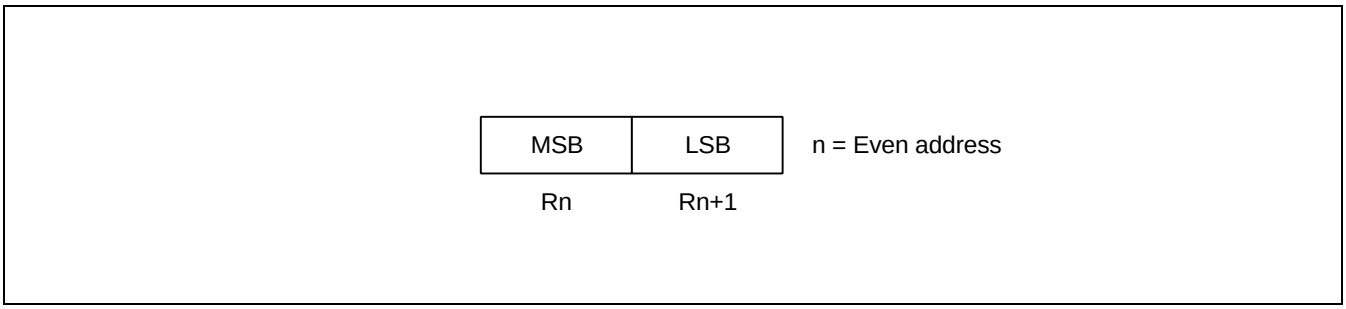


Figure 2-4. 16-Bit Register Pairs

SYSTEM STACK

S3F9-series microcontrollers use the system stack for subroutine calls and returns and to store data. The PUSH and POP instructions are used to control system stack operations. The S3C9498/F9498 architecture supports stack operations in the internal register file.

Stack Operations

Return addresses for procedure calls and 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 registers are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address always decrements *before* a push operation and increments *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 2-5.

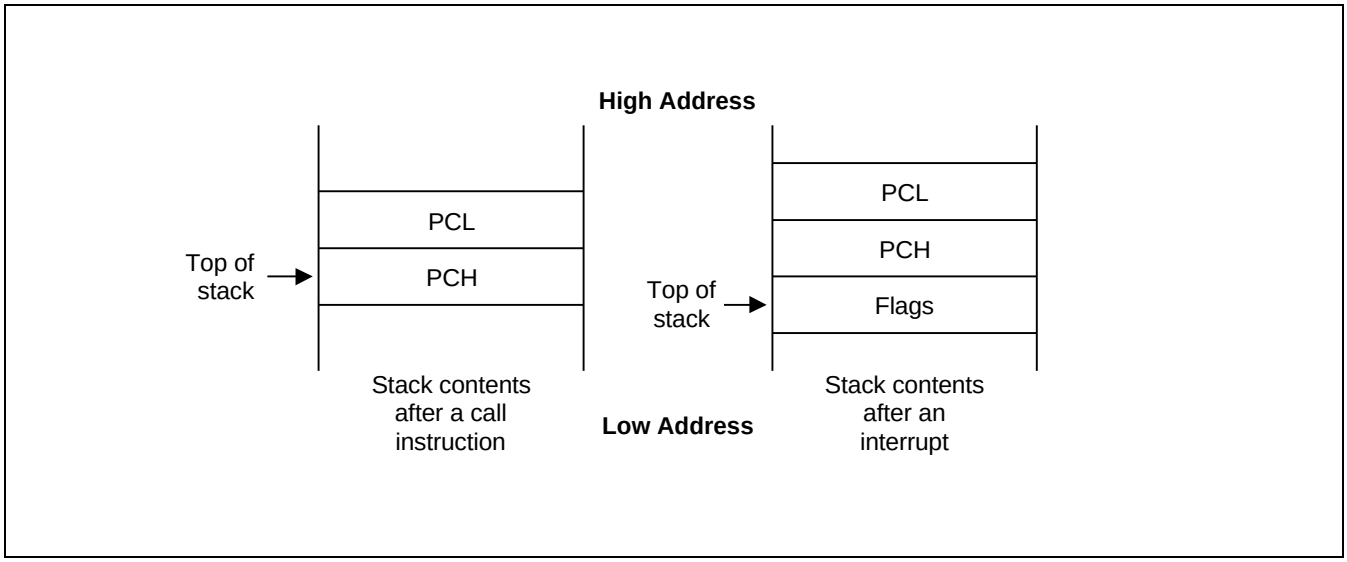


Figure 2-5. Stack Operations

Stack Pointer (SP)

Register location D9H contains the 8-bit stack pointer (SP) that is used for system stack operations. After a reset, the SP value is undetermined.

Because only internal memory space is implemented in the S3C9498/F9498, the SP must be initialized to an 8-bit value in the range 00H–0C0H.

NOTE

In case a Stack Pointer is initialized to 00H, it is decreased to FFH when stack operation starts. This means that a Stack Pointer access invalid stack area. We recommend that a stack pointer is initialized to C0H to set upper address of stack to BFH.

☞ 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      SP,#0C0H        ;  SP ← C0H (Normally, the SP is set to C0H by the
                        ;  initialization routine)
•
•
•
PUSH    SYM             ;  Stack address 0BFH ← SYM
PUSH    R15             ;  Stack address 0BEH ← R15
PUSH    20H             ;  Stack address 0BDH ← 20H
PUSH    R3              ;  Stack address 0BCH ← R3
•
•
•
POP     R3              ;  R3 ← Stack address 0BCH
POP     20H             ;  20H ← Stack address 0BDH
POP     R15             ;  R15 ← Stack address 0BEH
POP     SYM             ;  SYM ← Stack address 0BFH
```

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 SAM88RCRI instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

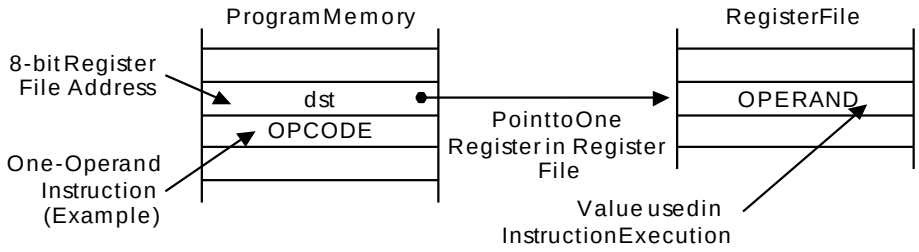
The SAM88RCRI instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction. The six addressing modes and their symbols are:

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

REGISTER ADDRESSING MODE (R)

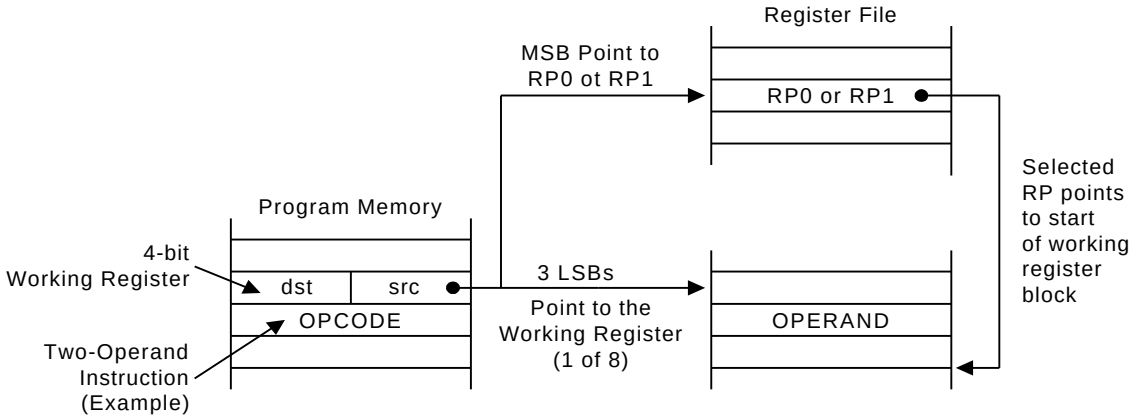
In Register addressing mode (R), the operand value is the content of a specified register or register pair (see Figure 3-1).

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).




```
Sample Instruction:

DEC     CNTR            ;  Where CNTR is the label of an 8-bit register address
```

Figure 3-1. Register Addressing



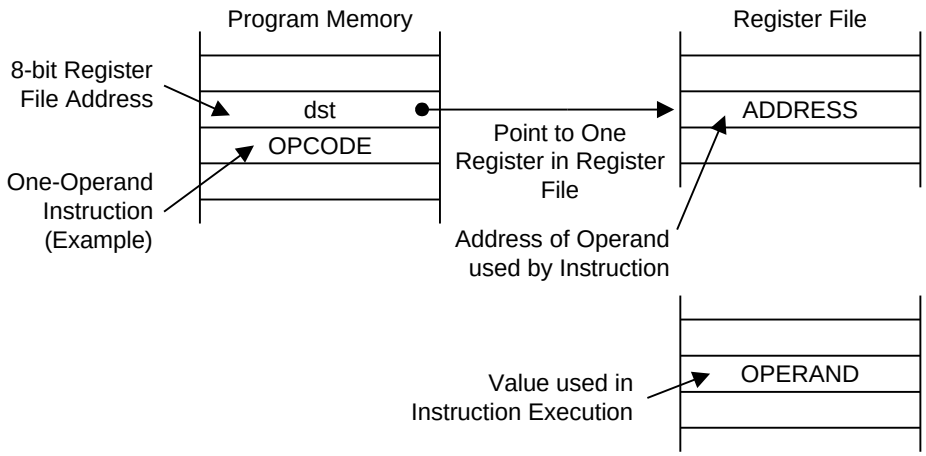

```
Sample Instruction:

ADD     R1, R2          ;  Where R1 and R2 are registers in the currently
                           selected working register area.
```

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.



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)

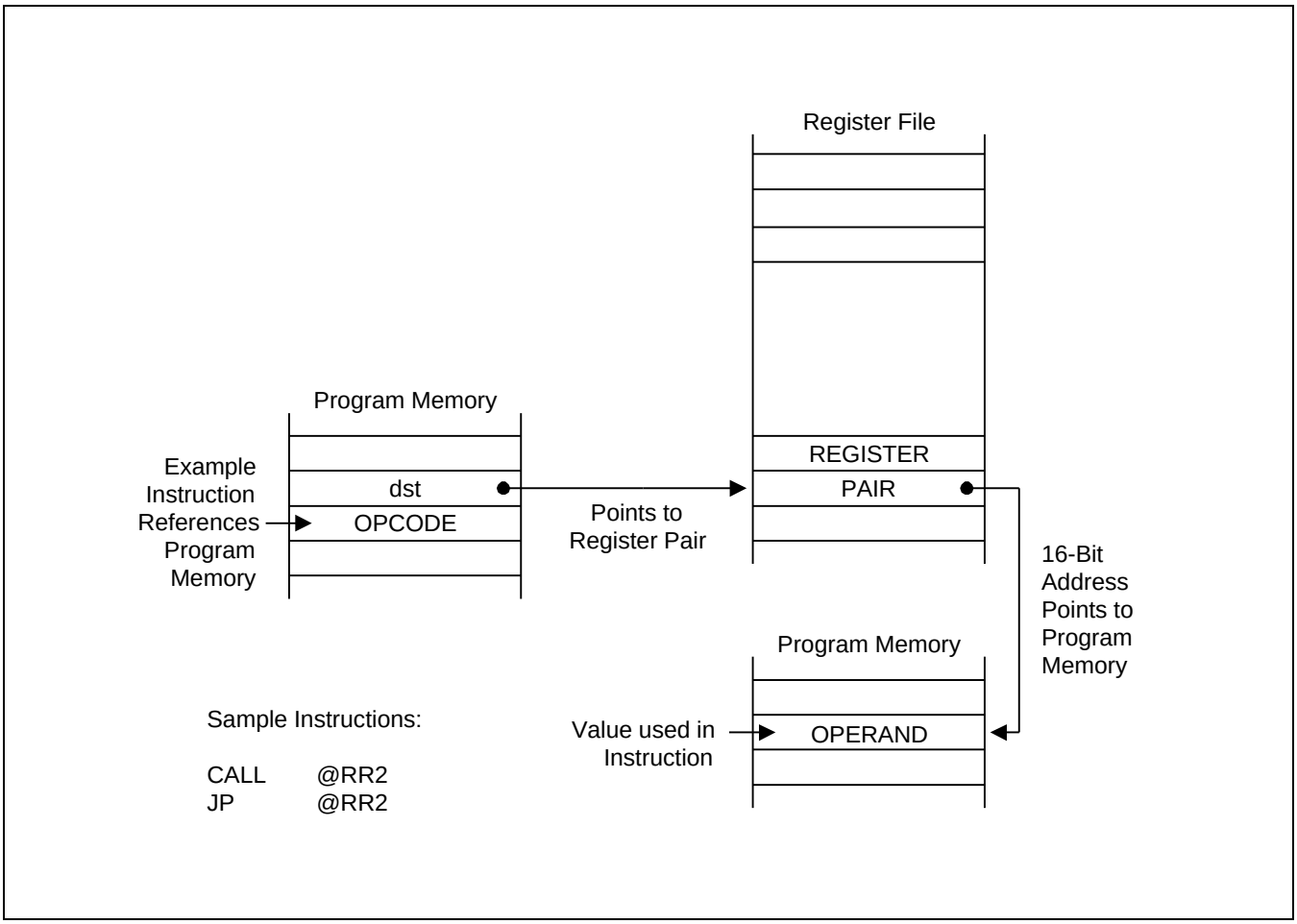


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

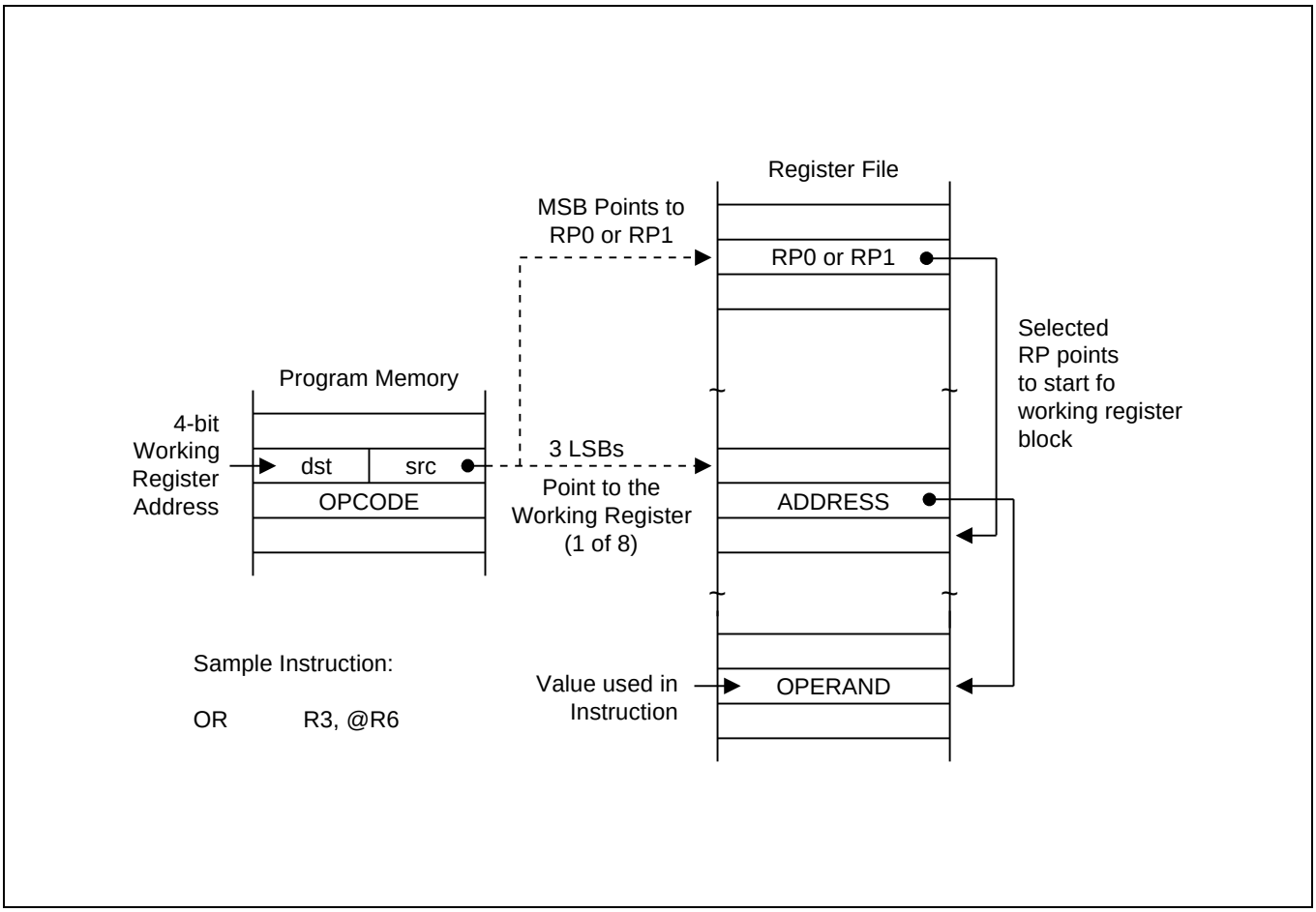
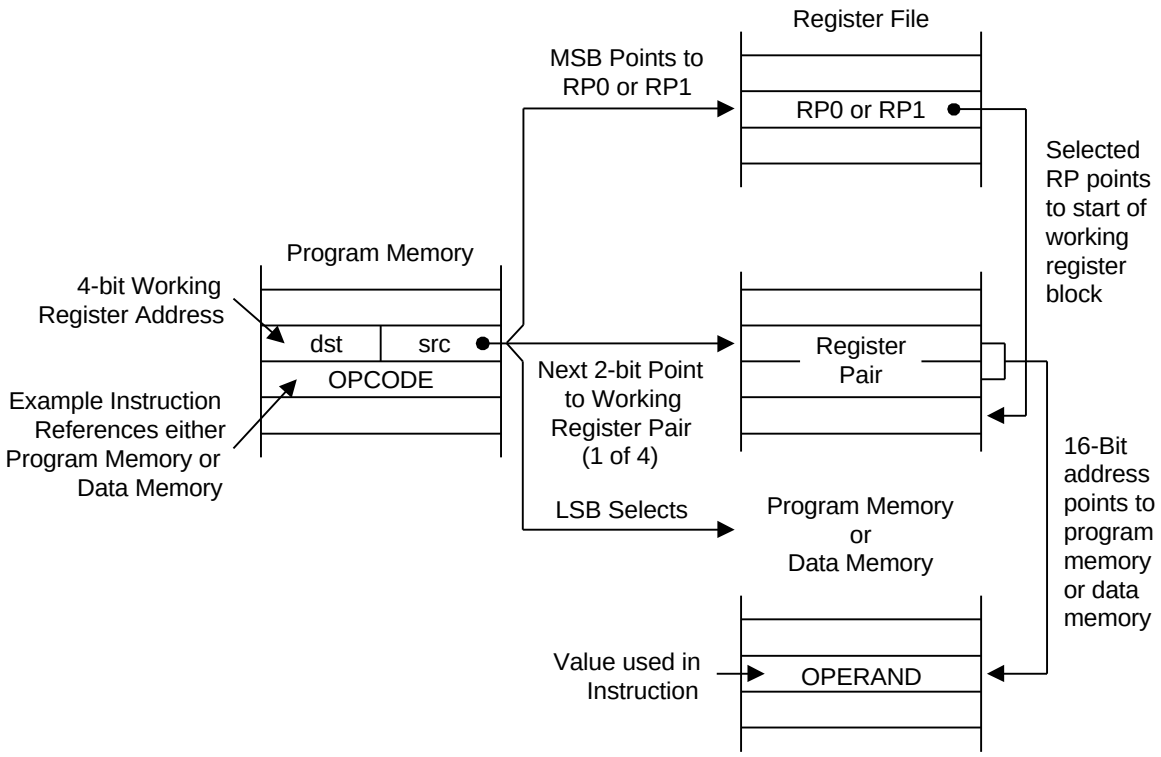


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)



Sample Instructions:

```
LCD     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.

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.

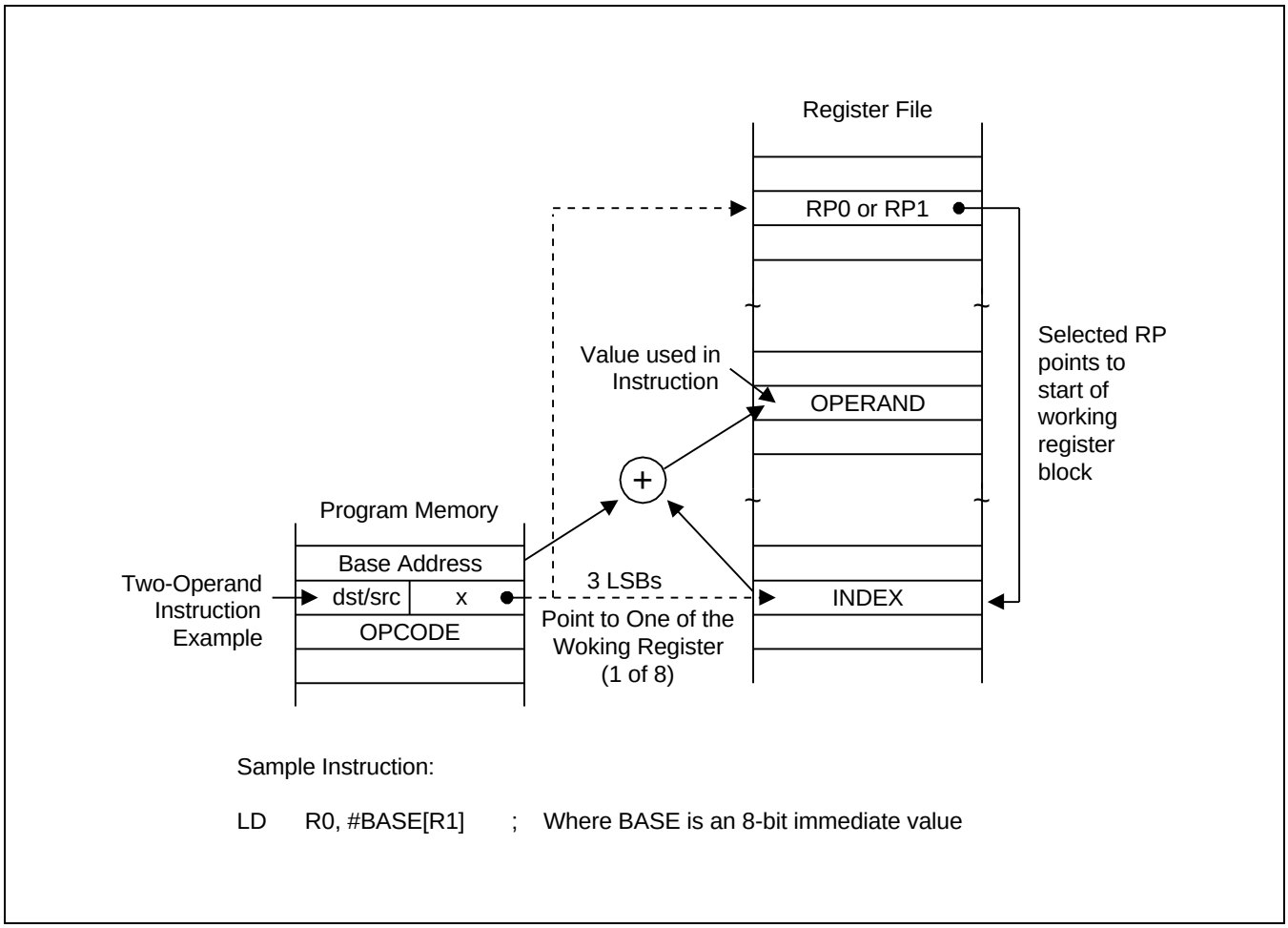
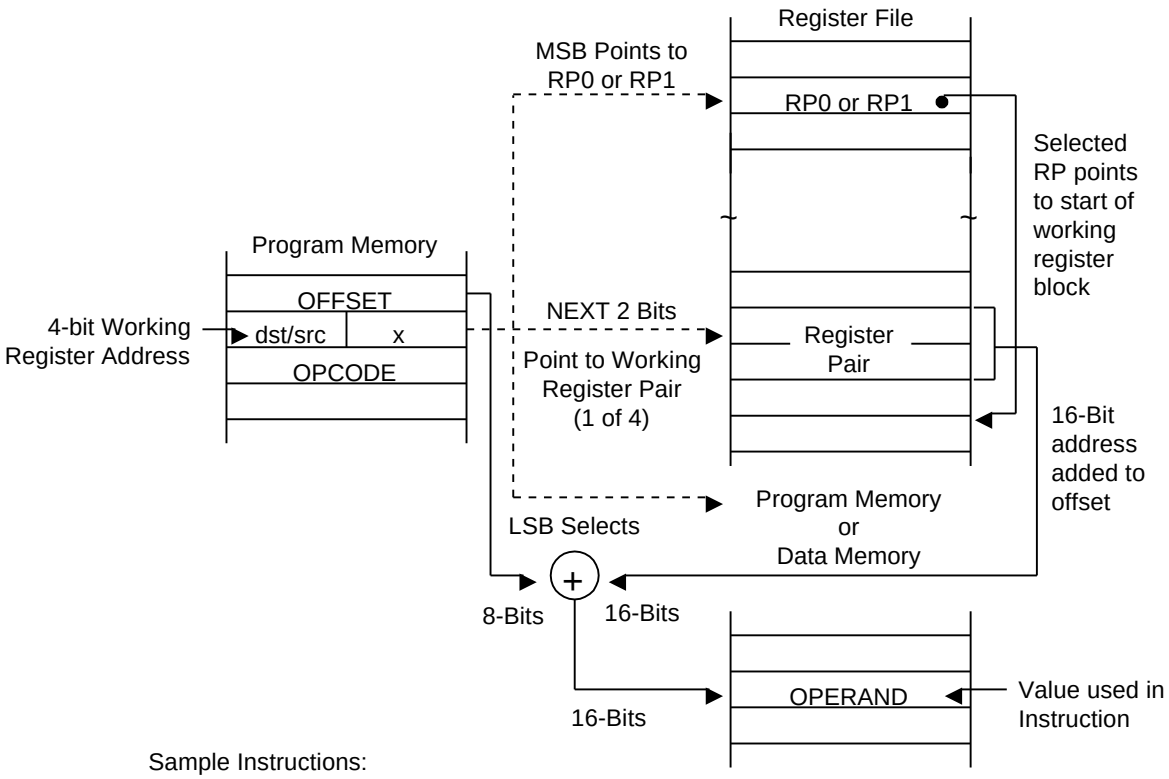


Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)

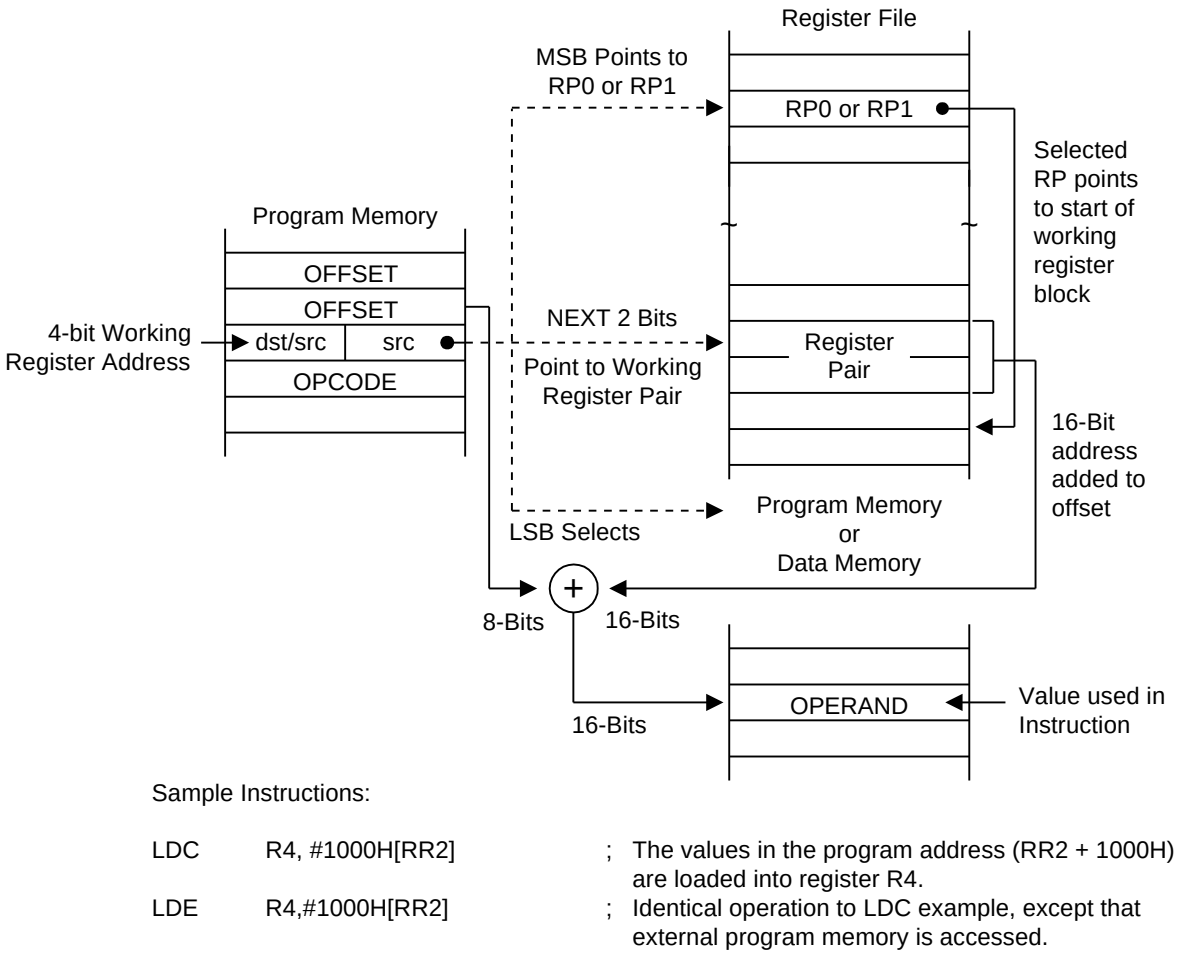


Sample Instructions:

```
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:

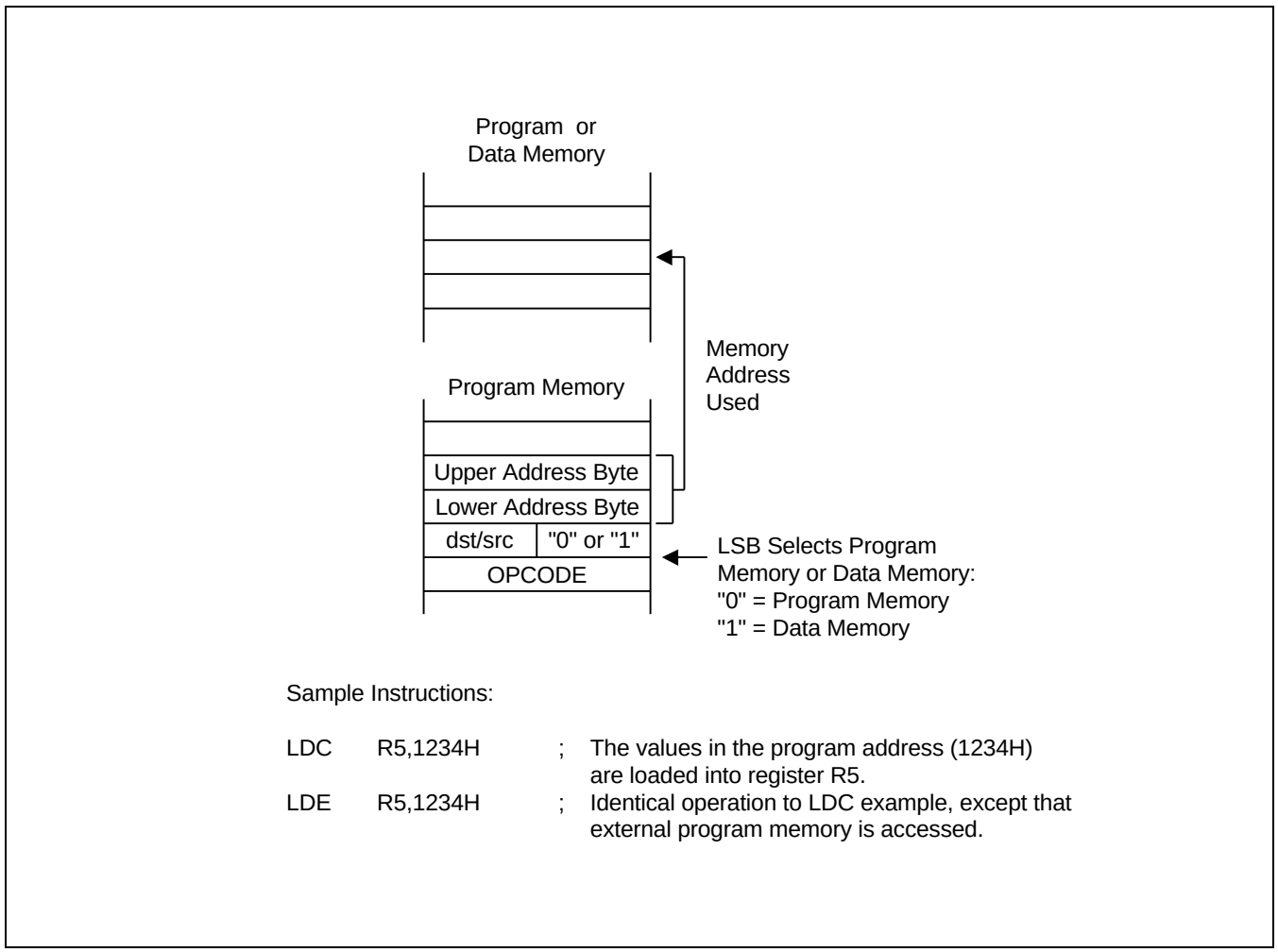
```
LDC     R4, #1000H[RR2]     ; The values in the program address (RR2 + 1000H)
                            ; are loaded into register R4.
LDE     R4,#1000H[RR2]      ; Identical operation to LDC example, except that
                            ; external program memory is accessed.
```

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.

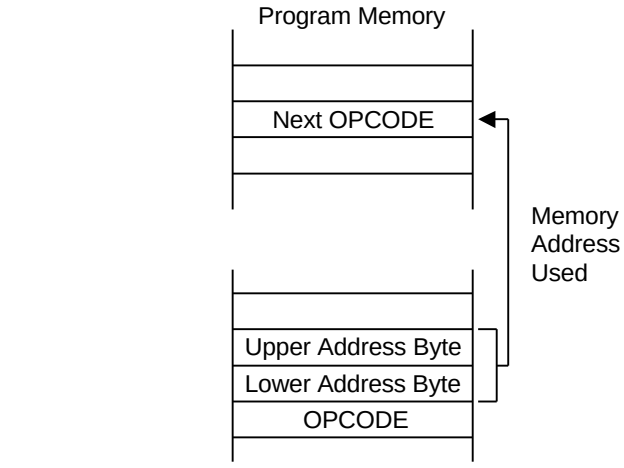


Sample Instructions:

```
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

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.

The instructions that support RA addressing is JR.

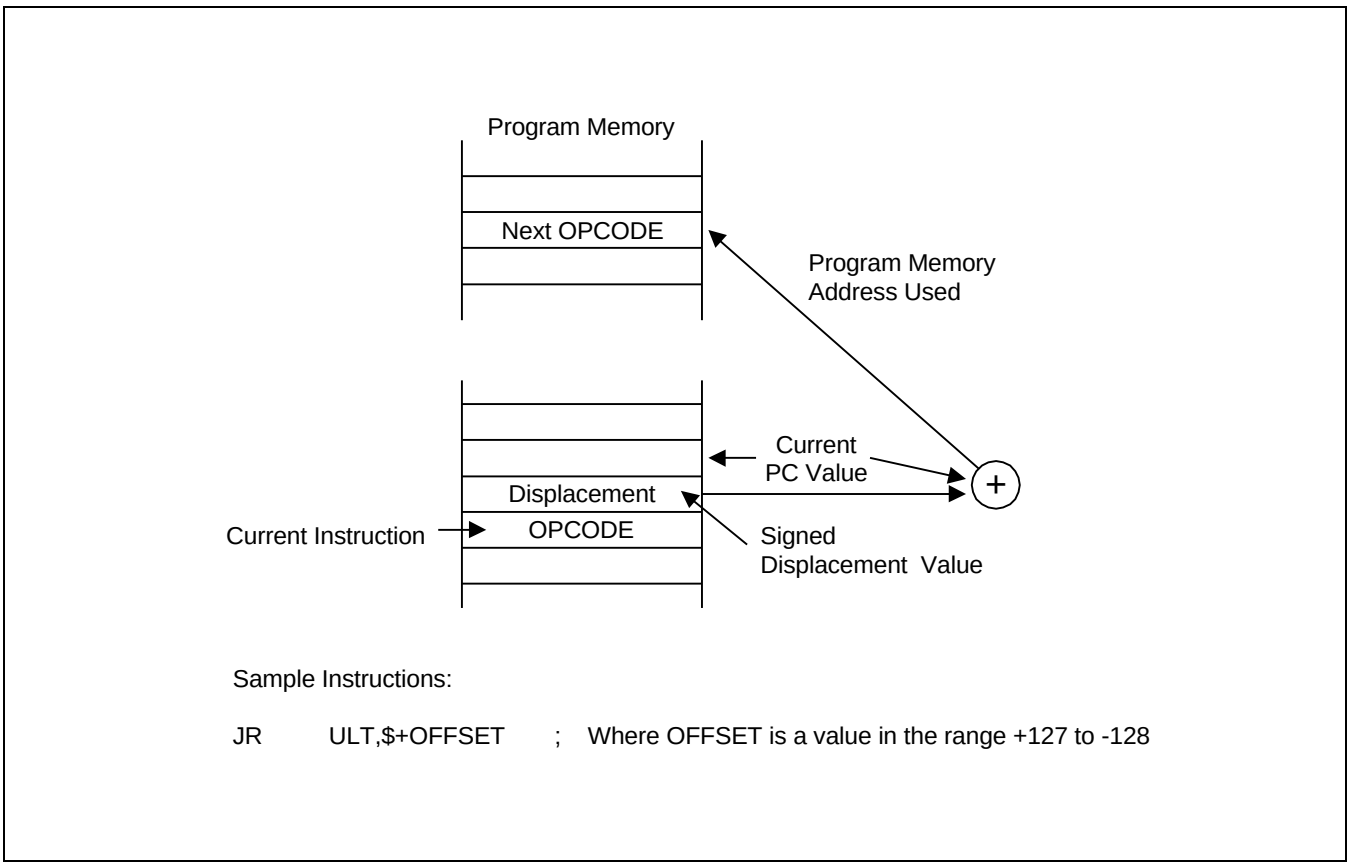


Figure 3-12. Relative Addressing

IMMEDIATE MODE (IM)

In Immediate (IM) addressing mode, the operand value used in the instruction is the value supplied in the operand field itself. Immediate addressing mode is useful for loading constant values into registers.

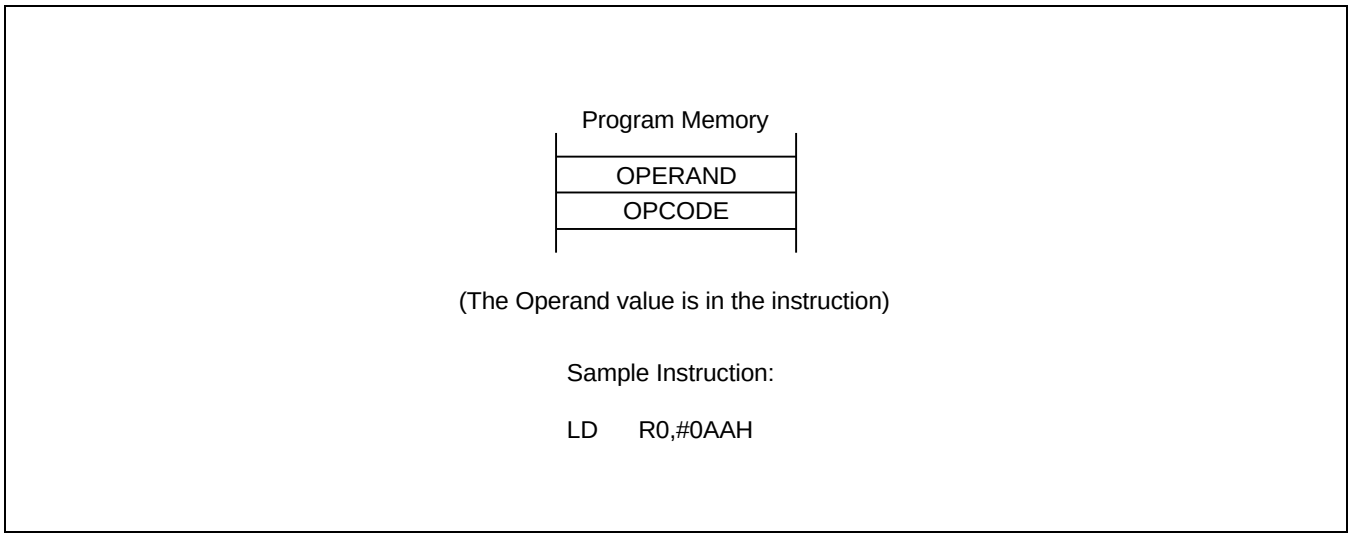


Figure 3-13. Immediate Addressing

NOTES

4 CONTROL REGISTERS

OVERVIEW

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.

The locations and read/write characteristics of all mapped registers in the S3C9498/F9498 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. System and Peripheral Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Timer C control register	TCCON	208	D0H	R/W
Timer D control register	TDCON	209	D1H	R/W
Timer C data register register	TCDATA	210	D2H	R/W
Timer D data register register	TDDATA	211	D3H	R/W
System Clock control register	CLKCON	212	D4H	R/W
System flags register	FLAGS	213	D5H	R/W
UART Baud rate data register	BRDATA	214	D6H	R/W
STOP control register	STPCON	215	D7H	R/W
Timer Counter selection register	TCNTSEL	216	D8H	R/W
Stack pointer register	SP	217	D9H	R/W
Timer counter register	TCNT	218	DAH	R
Location DBH is not mapped				
Basic timer control register	BTCON	220	DCH	R/W
Basic timer counter register	BTCNT	221	DDH	R
Location DEH is not mapped				
System mode register	SYM	223	DFH	R/W

Table 4-1. System and Peripheral Registers (continued)

Register Name	Mnemonic	Decimal	Hex	R/W
Port 0 Data Register	P0	224	E0H	R/W
Port 1 Data Register	P1	225	E1H	R/W
Port 2 Data Register	P2	226	E2H	R/W
Port 3 Data Register	P3	227	E3H	R/W
PWM data register	PWMDATA	228	E4H	R/W
PWM extension data register	PWMEX	229	E5H	R/W
Port 0 control register	P0CON	230	E6H	R/W
P1 interrupt control register	P1INT	231	E7H	R/W
Port 1 control High register	P1CONH	232	E8H	R/W
Port 1 control Low register	P1CONL	233	E9H	R/W
Port 2 control High register	P2CONH	234	EAH	R/W
Port 2 control Low register	P2CONL	235	EBH	R/W
Port 3 control register	P3CON	236	ECH	R/W
PWM control register	PWMCON	237	EDH	R/W
Timer 1 data register(high byte)	T1DATAH	238	EEH	R/W
Timer 1 data register(low byte)	T1DATAL	239	EFH	R/W
Timer 1 control register	T1CON	240	F0H	R/W
Serial I/O control register	SIOCON	241	F1H	R/W
Timer Interrupt pending register	TINTPND	242	F2H	R/W
Timer A control register	TACON	243	F3H	R/W
SIO pre-scalar register	SIOPS	244	F4H	R/W
Timer A data register	TADATA	245	F5H	R/W
Timer B data register	TBDATA	246	F6H	R/W
Location F7H is not mapped				
Timer B control register	TBCON	248	F8H	R/W
SIO data register	SIODATA	249	F9H	R/W
A/D converter data register(high byte)	ADDATAH	250	FAH	R
A/D converter data register(low byte)	ADDATAL	251	FBH	R
A/D converter control register	ADCON	252	FCH	R/W
UART control register	UARTCON	253	FDH	R/W
UART pending register	UARTPND	254	FEH	R/W
UART data register	UDATA	255	FFH	R/W

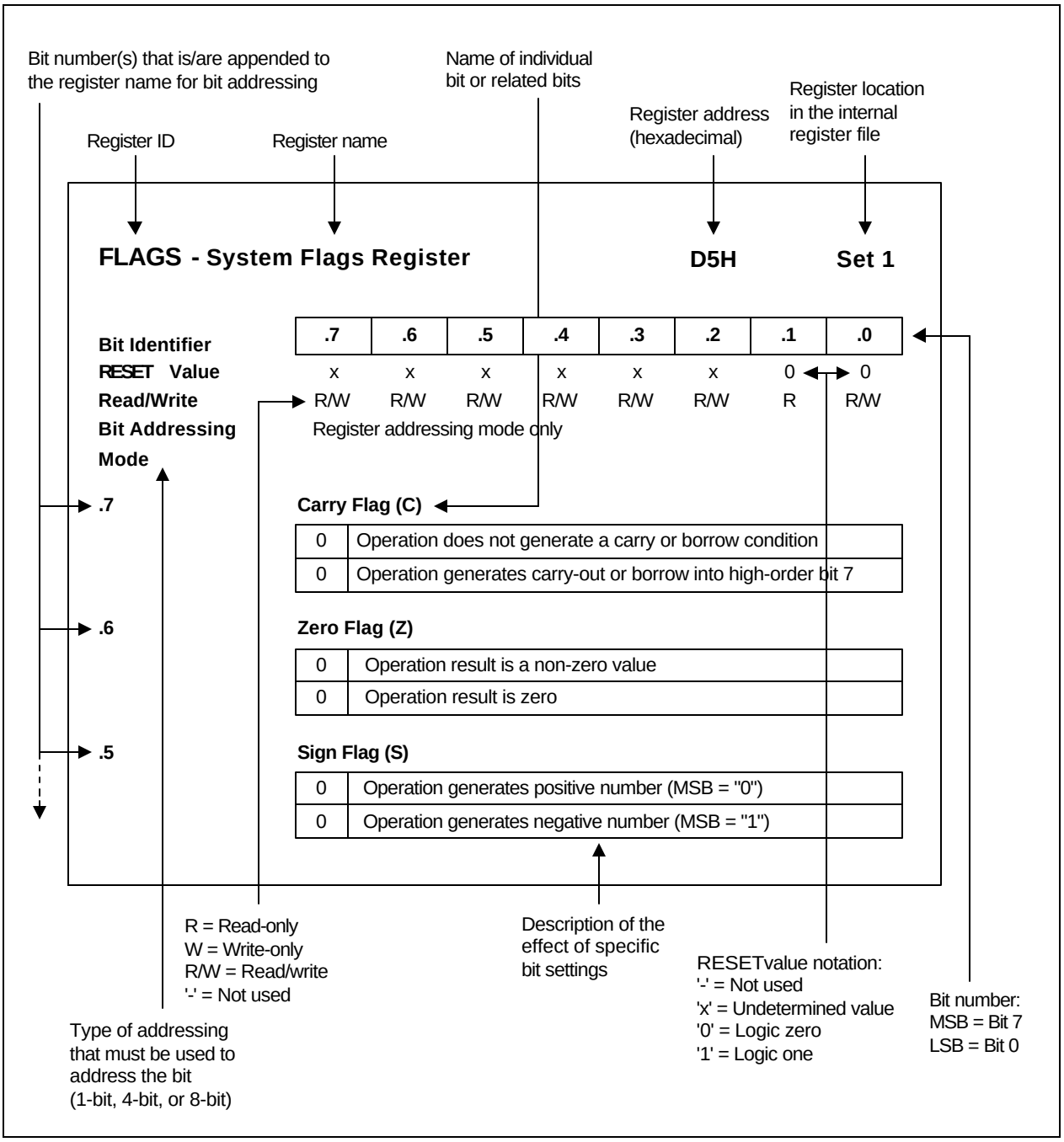


Figure 4-1. Register Description Format

ADCON — A/D Converter Control Register FCH

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-.4 A/D Input Pin Selection Bits

0	0	0	0	ADC0
0	0	0	1	ADC1
0	0	1	0	ADC2
0	0	1	1	ADC3
0	1	0	0	ADC4
0	1	0	1	ADC5
0	1	1	0	ADC6
0	1	1	1	ADC7
Other value				Connected with GND internally

.3 End-Of-Conversion (EOC) Status Bit

0	A/D conversion is in progress
1	A/D conversion complete

.2-.1 Clock Source Selection Bits

0	0	fxx/16 ($_{OSC}$ 8MHz)
0	1	fxx/8 ($_{OSC}$ 8MHz)
1	0	fxx/4 ($_{OSC}$ 8MHz)
1	1	Fxx (f_{OSC} 2.5MHz)

.0 A/D Conversion Start Bit

0	Disable operation
1	Start operation

NOTE: Maximum ADC clock input = 4MHz.

BTCON — Basic Timer Control Register DCH

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–.4 Watchdog Timer Function Enable Bit

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

.3–.2 Basic Timer Input Clock Selection Code

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/1024$
1	0	$f_{OSC}/128$
1	1	Invalid setting

.1 Basic Timer 8-Bit Counter Clear Bit

0	No effect
1	Clear the basic timer counter value

.0 Basic Timer Divider Clear Bit

0	No effect
1	Clear both dividers

NOTE: When you write a "1" to BTCON.0 (or BTCON.1), the basic timer counter (or basic timer divider) is cleared. The bit is then cleared automatically to "0".

CLKCON — System Clock Control Register D4H

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

.7 Oscillator IRQ Wake-up Function Enable Bit

0	Enable IRQ for main system oscillator wake-up function
1	Disable IRQ for main system oscillator wake-up function

.6-.5

Not used for the S3C9498/F9498

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

0	0	fxx/16
0	1	fxx/8
1	0	fxx/2
1	1	fxx/1 (non-divided)

.2-.0

Not used for the S3C9498/F9498

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

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

.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–.0 Not used for the S3C9498/F9498

P0CON — Port 0 Control Register E6H

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

.7-.6 **Not used for S3C9498/F9498**

.5-.4 **P0.2**

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

.3–.2 **P0.1/TxD**

0	0	Input mode with pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative function, TxD output

.1-.0 **P0.0/RxD**

0	0	Input mode with pull-up; RxD input
0	1	Input mode; RxD input
1	0	Push-pull output
1	1	Alternative function; RxD output

NOTE: When users use Port 0, users must be care of the pull-up resistance status.

P1CONH — Port 1 Control Register (High Byte) E8H

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-.6 **P1.7/ADC7**

0	0	Input mode with pull-up
0	1	Not used for S3C9498/F9498
1	0	Push-pull output
1	1	Alternative function; ADC7 input

.5-.4 **P1.6/ADC6/TDOUT**

0	0	Input mode with pull-up
0	1	Alternative function; TDOUT output
1	0	Push-pull output
1	1	Alternative function; ADC6 input

.3-.2 **P1.5/ADC5**

0	0	Input mode with pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative function; ADC5 input

.1-.0 **P1.4/ADC4**

0	0	Input mode with pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative function; ADC4 input

P1CONL — Port 1 Control Register (Low Byte) E9H

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-.6 **P1.3/ADC3**

0	0	Input mode with pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative function; ADC3 input

.5-.4 **P1.2/ADC2**

0	0	Input mode with pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative function; ADC2 input

.3-.2 **P1.1/ADC1/INT1**

0	0	Input mode with pull-up; INT1 input
0	1	Input mode; INT1 input
1	0	Push-pull output
1	1	Alternative function; ADC1 input

.1-.0 **P1.0/ADC0/INT0**

0	0	Input mode with pull-up; INT0 input
0	1	Input mode; INT0 input
1	0	Push-pull output
1	1	Alternative function; ADC0 input

P1INT — Port 1 Interrupt Control Register E7H

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

.7-.6 **Not used for S3C9498/F9498**

.5-.4 **P1.1/ INT1 Interrupt Enable/Disable Selection Bits**

0	X	Interrupt Disable
1	0	Interrupt Enable; falling edge
1	1	Interrupt Enable; rising edge

.3-.2 **P1.0/ INT0 Interrupt Enable/Disable Selection Bits**

0	X	Interrupt Disable
1	0	Interrupt Enable; falling edge
1	1	Interrupt Enable; rising edge

.1 **INT1 Interrupt Pending Bit**

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

.0 **INT0 Interrupt Pending Bit**

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

P2CONH — Port 2 Control Register (High Byte) EAH

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–.6 **P2.7/PWM**

0	0	Input mode with pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative function; PWM signal output

.5-.4 **P2.6/T1CAP**

0	0	Input mode with pull-up; T1CAP input
0	1	Input mode; T1CAP input
1	0	Push-pull output
1	1	Open-drain output

.3–.2 **P2.5/T1OUT**

0	0	Input mode with pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative function; T1OUT signal output

.1–.0 **P2.4/T1CK**

0	0	Input mode with pull-up; T1CK input
0	1	Input mode; T1CK input
1	0	Push-pull output
1	1	Open-drain output

P2CONL — Port 2 Control Register (Low Byte) EBH

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-.6 P2.3/TBOUT

0	0	Input mode with pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative function; TBOUT signal output

.5-.4 P2.2/TACAP

0	0	Input mode with pull-up; TACAP input
0	1	Input mode; TACAP input
1	0	Push-pull output
1	1	Open-drain output

.3-.2 P2.1/TACK

0	0	Input mode with pull-up; TACK input
0	1	Input mode; TACK input
1	0	Push-pull output
1	1	Open-drain output

.1-.0 P2.0/TAOUT

0	0	Input mode with pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative function; TAOUT signal output

P3CON — Port 3 Control Register ECH

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–.6 **P3.6/P3.5/P3.4/P3.3**

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

.5–.4 **P3.2/SCK**

0	0	Input mode, pull-up (SCK input)
0	1	Input mode (SCK input)
1	0	Push-pull output
1	1	Alternative output mode (SCK output)

.3–.2 **P3.1/SO**

0	0	Input mode, pull-up
0	1	Input mode
1	0	Push-pull output
1	1	Alternative output mode (SO)

.1–.0 **P3.0/SI**

0	0	Input mode(SI) , pull-up
0	1	Input mode (SI)
1	0	Push-pull output
1	1	Alternative output mode(Not used)

PWMCON — PWM Control Register EDH

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–.6 PWM Input Clock Selection Bit

0	0	fosc/256
0	1	fosc/64
1	0	fosc/8
1	1	fosc/1

.5 PWM Data Reload Interval Selection Bit

0	Reload from 12-bit up counter overflow
1	Reload from 6-bit up counter overflow

.4 Not used for S3C9498/F9498

.3 PWM Counter Clear Bit

0	No effect
1	Clear 12-bit up counter (when write)

.2 PWM Counter Enable Bit

0	Stop counter
1	Start (Resume counting)

.1 PWM Overflow Interrupt Enable bit (12-bit Counter Overflow)

0	Disable interrupt
1	Enable interrupt

.0 PWM 12-Bit Counter Overflow Interrupt Pending Bit

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

SIOCON — Serial I/O Module Control Registers F1H

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 SIO Shift Clock Selection Bit

0	Interval 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 shift and clock counter
1	Enable shift 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
1	Interrupt pending (*Clear pending bit when write)*

SP — Stack Pointer D9H

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

.7–.0 Stack Pointer Address

The stack pointer value is 8-bit stack pointer address (SP7–SP0). The SP value is undefined following a reset.

STPCON — Stop Control Register D7H

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–.0 STOP Control Bits

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

NOTE: Before executing the STOP instruction, you must set this STPCON register as “10100101b”. Otherwise the STOP instruction will not be executed.

SYM — System Mode Register DFH

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

.7–.4

Not used for S3C9498/F9498

.3 Global Interrupt Enable Bit

0	Disable all interrupts
1	Enable all interrupt

.2–.0 Page Select Bits

0	0	0	Page 0
0	0	1	Page 1 (Not used for S3C9498/F9498)
0	1	0	Page 2 (Not used for S3C9498/F9498)
0	1	1	Page 3 (Not used for S3C9498/F9498)
1	0	0	Page 4 (Not used for S3C9498/F9498)
1	0	1	Page 5 (Not used for S3C9498/F9498)
1	1	0	Page 6 (Not used for S3C9498/F9498)
1	1	1	Page 7 (Not used for S3C9498/F9498)

NOTE: Following a reset, you must enable global interrupt processing by executing an EI instruction (not by writing a "1" to SYM.3).

T1CON — Timer 1 Control Register F0H

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 1 Input Clock Selection Bits

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

.4-.3 Timer 1 Operating Mode Selection Bits

0	0	Interval mode
0	1	Capture mode (Capture on rising edge, OVF can occur)
1	0	Capture mode (Capture on falling edge, OVF can occur)
1	1	PWM mode

.2 Timer 1 Counter Clear Bit

0	No effect
1	Clear the timer 1 counter (Auto-clear bit)

.1 Timer 1 Match/Capture Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer 1 Overflow Interrupt Enable

0	Disable overflow interrupt
1	Enable overflow interrupt

TACON — Timer A Control Register F3H

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-.6 Timer A Input Clock Selection Bits

0	0	fxx/1024
0	1	fxx/256
1	0	fxx/64
1	1	External clock (TACK)

.5-.4 Timer A Operating Mode Selection Bits

0	0	Internal mode (TAOUT 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 interrupt can occur)

.3 Timer A Counter Clear Bit

0	No effect
1	Clear the timer A counter (After clearing, return to zero)

.2 Timer A Overflow Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.1 Timer A Match/Capture Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer A Start/Stop Bit

0	Stop Timer A
1	Start Timer A

TBCON — Timer B Control Register F8H

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

.7-.6 Timer B Input Clock Selection Bits

0	0	fxx/8
0	1	fxx/4
1	0	fxx/2
1	1	fxx/1

.5 Not used for S3C9498/F9498

.4 Timer B Operating Mode Selection Bits

0	Interval mode (TBOUT mode)
1	PWM mode (OVF interrupt can occur)

.3 Timer B Counter Clear Bit

0	No effect
1	Clear the timer B counter (After clearing, return to zero)

.2 Timer B Overflow Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.1 Timer B Match Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer B Start/Stop Bit

0	Stop Timer B
1	Start Timer B

TCCON — Timer C Control Register D0H

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 Timer 0 operation mode selection bit

0	Two 8-bit timers mode (Timer C/D)
1	One 16-bit timer mode (Timer 0)

.6

Not used for S3C9498/F9498 (Must be kept '0')

.5–.4 Timer C Input Clock Selection Bits

0	0	Fxx/1024
0	1	fxx/512
1	0	fxx/8
1	1	fxx

.3 Timer C Counter Clear Bit

0	No affect
1	Clear the timer C counter *(when write)*

.2 Timer C Counter Run Enable Bit

0	Disable counter running
1	Enable counter running

.1 Timer C Interrupt Enable Bit

0	Disable Interrupt
1	Enable Interrupt

.0 Timer C Interrupt pending Bit

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

TCNTSEL — Timer Counter read selection Register D8H

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-.3 Not used for S3C9498/F9498

.2-.0 Timer counter read selection bits

0	0	0	Select Timer A counter
0	0	1	Select Timer B counter
0	1	0	Select Timer C counter
0	1	1	Select Timer D counter
1	X	0	Select Timer 1 counter high byte
1	X	1	Select Timer 1 counter low byte

TDCON — Timer D Control Register D1H

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 Timer D operating Mode Selection Bits

0	0	Interval mode
0	1	6-bit PWM mode (OVF interrupt can occur)
1	0	7-bit PWM mode (OVF interrupt can occur)
1	1	8-bit PWM mode (OVF interrupt can occur)

.5–.4 Timer D Clock Selection Bits

0	0	fxx/8
0	1	fxx/4
1	0	fxx/2
1	1	fxx

.3 Timer D Counter Clear Bit

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

.2 Timer D Count Enable Bit

0	Disable count operation
1	Enable count operation

.1 Timer D match Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer D overflow interrupt enable bit

0	Disable interrupt
1	Enable interrupt

TINTPND — Interrupt Pending Register F2H

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 Timer 1 Overflow Interrupt Pending Bit

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

.6 Timer 1 Match/Capture Interrupt Pending Bit

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

.5 Timer D Overflow Interrupt Pending Bit

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

.4 Timer D Match Interrupt Pending Bit

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

.3 Timer B Overflow Interrupt Pending Bit

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

.2 Timer B Match/Capture Interrupt Pending Bit

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

.1 Timer A Overflow Interrupt Pending Bit

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

.0 Timer A Match/Capture Interrupt Pending Bit

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

UARTCON — UART Control Register FDH

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 Operating mode and baud rate selection bits

0	0	Mode 0: SIO mode [fxx/(16 × (BRDATA1 + 1))]
0	1	Mode 1: 8-bit UART [fxx/(16 × (BRDATA1 + 1))]
1	0	Mode 2: 9-bit UART [fxx/16]
1	1	Mode 3: 9-bit UART [fxx/(16 × (BRDATA1 + 1))]

.5 Multiprocessor communication(1) enable bit (for modes 2 and 3 only)

0	Disable
1	Enable

.4 Serial data receive enable bit

0	Disable
1	Enable

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

.2 Location of the 9th data bit that was received in UART mode 2 or 3 ("0" or "1")

.1 Receive interrupt enable bit

0	Disable Receive interrupt
1	Enable Receive interrupt

.0 Transmit interrupt enable bit

0	Disable Transmit interrupt
1	Enable Transmit Interrupt

NOTES:

1. In mode 2 or 3, if the MCE (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 MCE = "1", then the receive interrupt will not be activated if a valid stop bit was not received. In mode 0, the MCE(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.

UARTPND — UART Pending and parity control FEH

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

.7-.3 **Not used for the S3C9498/F9498**

.1 **UART receive interrupt pending flag**

0	Not pending
0	*Clear pending bit (when write)*
1	Interrupt pending

.0 **UART transmit interrupt pending flag**

0	Not pending
0	*Clear pending bit (when write)*
1	Interrupt pending

NOTES:

1. In order to clear a data transmit or receive interrupt pending flag, you must write a "0" to the appropriate pending bit.
2. To avoid programming errors, we recommend using load instruction (except for LDB), when manipulating UARTPND values.

NOTES

5 INTERRUPT STRUCTURE

OVERVIEW

The SAM88RCRI interrupt structure has two basic components: a vector, and sources. The number of interrupt sources can be serviced through an interrupt vector which is assigned in ROM address 0000H.

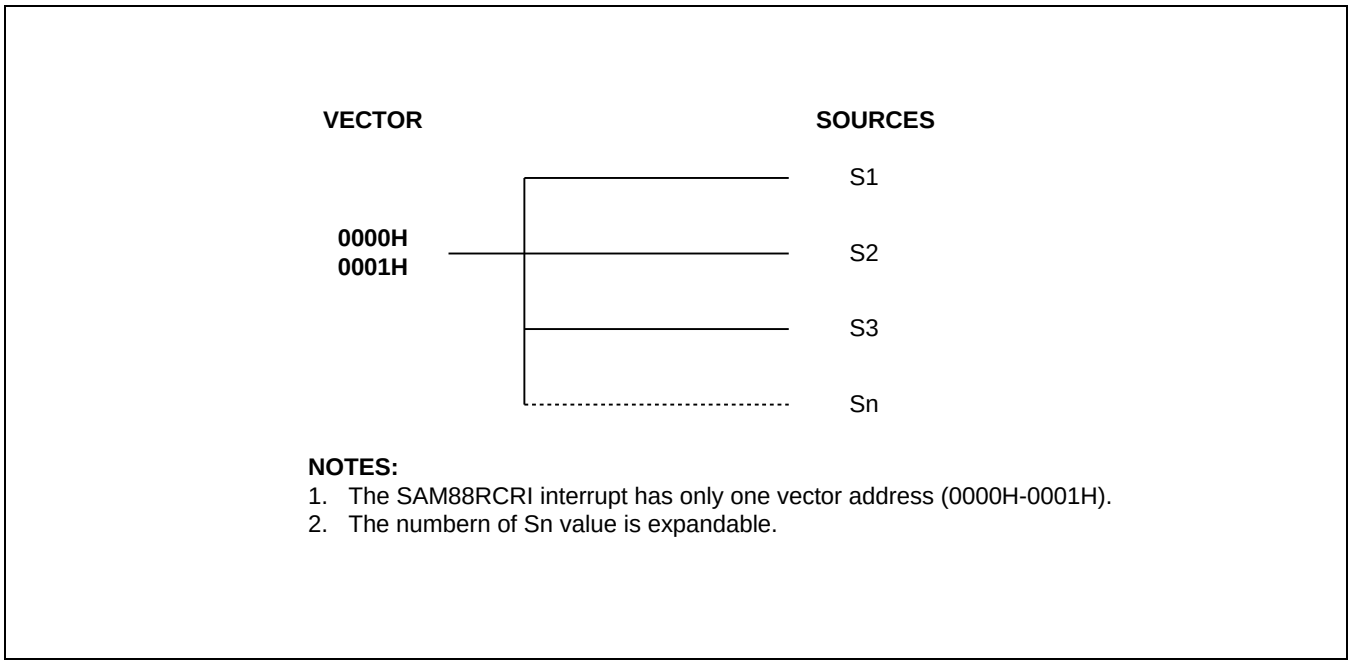


Figure 5-1. S3C9-Series Interrupt Type

INTERRUPT PROCESSING CONTROL POINTS

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

— Global interrupt enable and disable (by EI and DI instructions)

— Interrupt source enable and disable settings in the corresponding peripheral control register(s)

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

The system mode register, SYM (DFH), is used to enable and disable interrupt processing.

SYM.3 is the enable and disable bit for global interrupt processing respectively, by modifying SYM.3. An Enable Interrupt (EI) instruction must be included in the initialization routine that follows a reset operation in order to enable interrupt processing. Although you can manipulate SYM.3 directly to enable and disable interrupts during normal operation, we recommend that you use the EI and DI instructions for this purpose.

INTERRUPT PENDING FUNCTION TYPES

When the interrupt service routine has executed, the application program's service routine must clear the appropriate pending bit before the return from interrupt subroutine (IRET) occurs.

INTERRUPT PRIORITY

Because there is not a interrupt priority register in SAM88RCRI, the order of service is determined by a sequence of source which is executed in interrupt service routine.

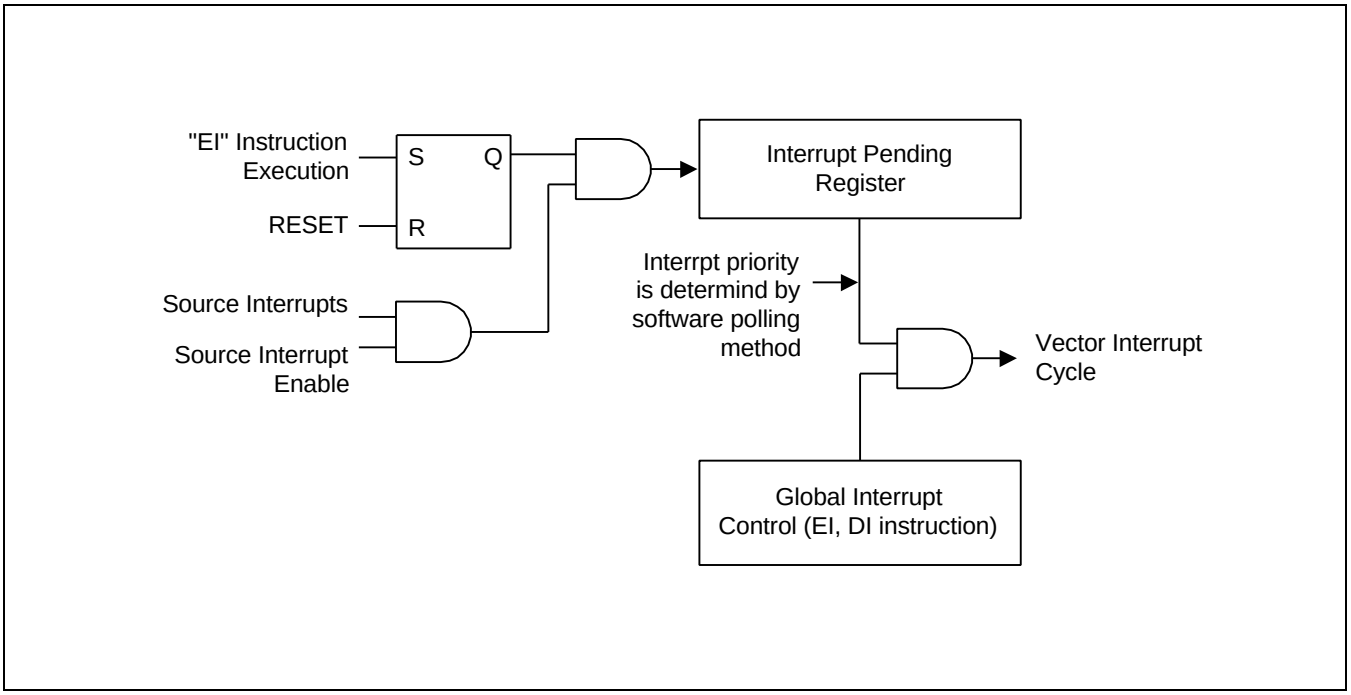


Figure 5-2. Interrupt Function Diagram

INTERRUPT SOURCE SERVICE SEQUENCE

The interrupt request polling and servicing sequence is as follows:

1. A source generates an interrupt request by setting the interrupt request pending bit to "1".
2. The CPU generates an interrupt acknowledge signal.
3. The service routine starts and the source's pending flag is cleared to "0" by software.
4. Interrupt priority must be determined by software polling method.

INTERRUPT SERVICE ROUTINES

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

— Interrupt processing must be enabled (EI, SYM.3 = "1")

— Interrupt must be enabled at the interrupt's source (peripheral control register)

If all of 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 global interrupt enable bit in the SYM register (DI, SYM.3 = "0") to disable all subsequent interrupts.
2. Save the program counter and status flags to stack.
3. Branch to the interrupt vector to fetch the service routine's address.
4. Pass control to the interrupt service routine.

When the interrupt service routine is completed, an Interrupt Return instruction (IRET) occurs. The IRET restores the PC and status flags and sets SYM.3 to "1" (EI), allowing the CPU to process the next interrupt request.

GENERATING INTERRUPT VECTOR ADDRESSES

The interrupt vector area in the ROM contains the address of the interrupt service routine. Vectored interrupt processing follows this sequence:

1. Push the program counter's low-byte value to stack.
2. Push the program counter's high-byte value to stack.
3. Push the FLAGS register values to stack.
4. Fetch the service routine's high-byte address from the vector address 0000H.
5. Fetch the service routine's low-byte address from the vector address 0001H.
6. Branch to the service routine specified by the 16-bit vector address.

S3C9498/F9498 INTERRUPT STRUCTURE

The S3F9498 microcontroller has four peripheral interrupt sources:

— Timer A/ B match / overflow, Timer C / D interrupt, Timer 1 match / overflow interrupt
— UART transmit interrupt / receive interrupt, PWM overflow interrupt
— SIO interrupt, INT0, INT1 external interrupt

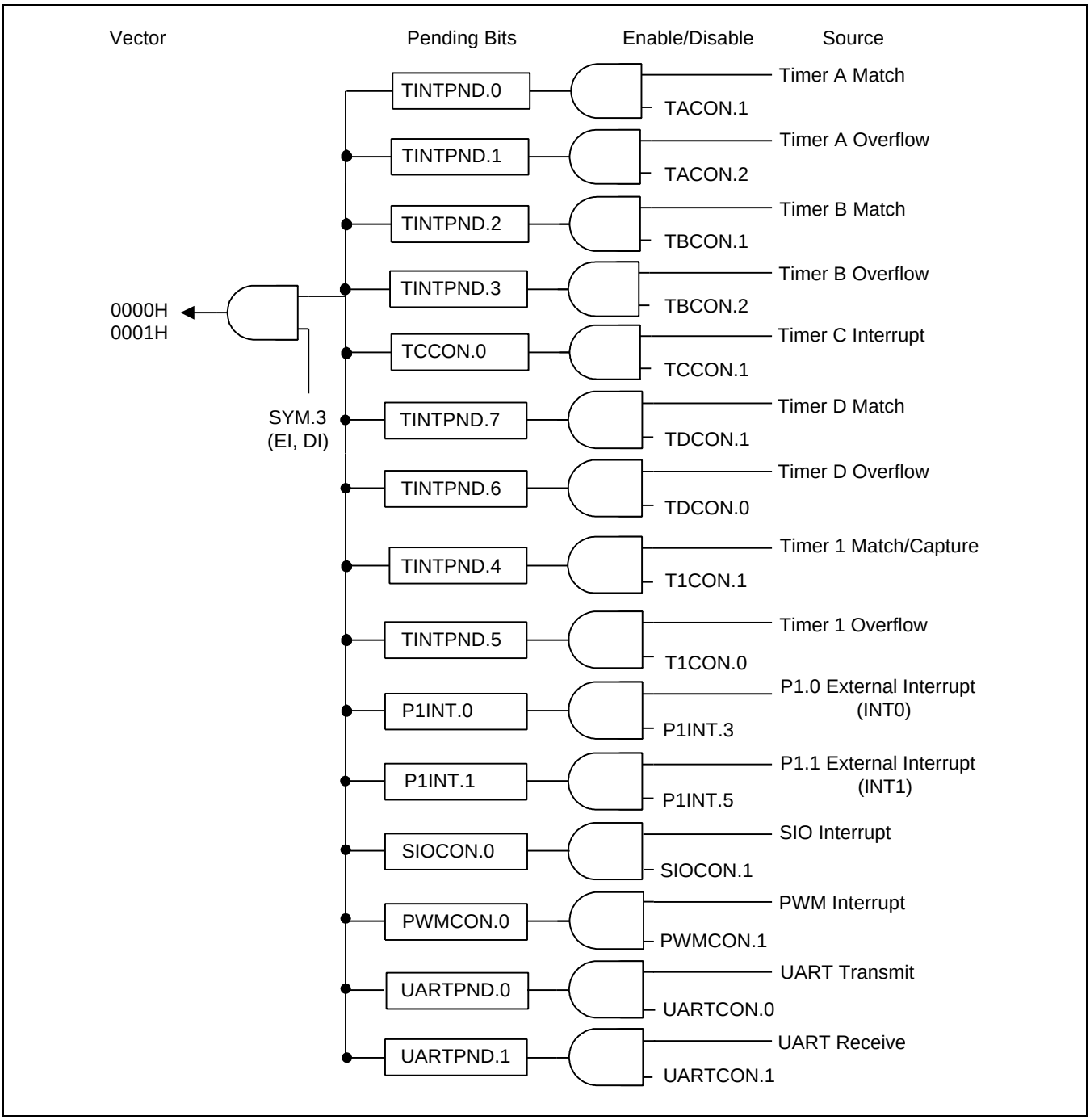


Figure 5-3. S3F9498 Interrupt Structure

6 SAM88RCRI INSTRUCTION SET

OVERVIEW

The SAM88RCRI instruction set is designed to support the large register file. It includes a full complement of 8-bit arithmetic and logic operations. There are 41 instructions. No special I/O instructions are necessary because I/O control and data registers are mapped directly into the register file. Flexible instructions for bit addressing, rotate, and shift operations complete the powerful data manipulation capabilities of the SAM88RCRI instruction set.

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 13-bit program memory or data memory addresses. For detailed information about register addressing, please refer to Chapter 2, "Address Spaces".

ADDRESSING MODES

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

Table 6-1. Instruction Group Summary

Mnemonic	Operands	Instruction
Load Instructions		
CLR	dst	Clear
LD	dst,src	Load
LDC	dst,src	Load program memory
LDE	dst,src	Load external data memory
LDCD	dst,src	Load program memory and decrement
LDED	dst,src	Load external data memory and decrement
LDCI	dst,src	Load program memory and increment
LDEI	dst,src	Load external data memory and increment
POP	dst	Pop from stack
PUSH	src	Push to stack
Arithmetic Instructions		
ADC	dst,src	Add with carry
ADD	dst,src	Add
CP	dst,src	Compare
DEC	dst	Decrement
INC	dst	Increment
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		
CALL	dst	Call procedure
IRET		Interrupt return
JP	cc,dst	Jump on condition code
JP	dst	Jump unconditional
JR	cc,dst	Jump relative on condition code
RET		Return
Bit Manipulation Instructions		
TCM	dst,src	Test complement under mask
TM	dst,src	Test under mask
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
CPU Control Instructions		
CCF		Complement carry flag
DI		Disable interrupts
EI		Enable interrupts
IDLE		Enter Idle mode
NOP		No operation
RCF		Reset carry flag
SCF		Set carry flag
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.4–FLAGS.7, can be tested and used with conditional jump instructions;

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.

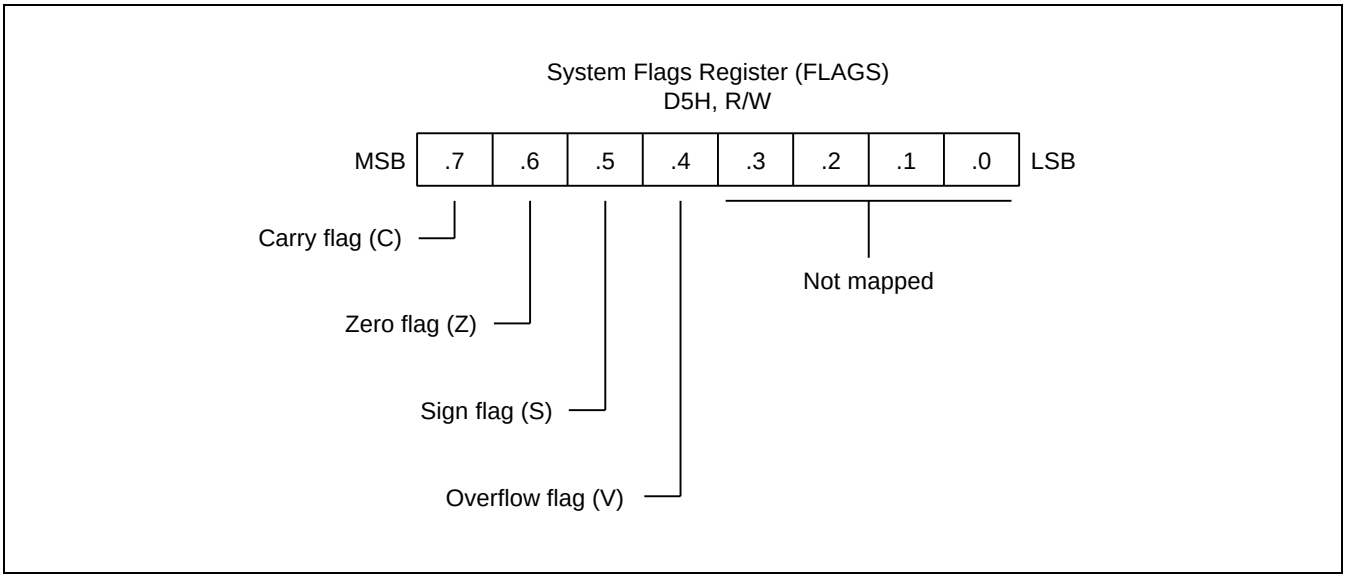


Figure 6-1. System Flags Register (FLAGS)

FLAG DESCRIPTIONS

Overflow Flag (FLAGS.4, V)

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.

Sign Flag (FLAGS.5, S)

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.

Zero Flag (FLAGS.6, Z)

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.

Carry Flag (FLAGS.7, C)

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.

INSTRUCTION SET NOTATION

Table 6-2. Flag Notation Conventions

Flag	Description
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow 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
FLAGS	Flags register (D5H)
#	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)
rr	Working register pair	RRp (p = 0, 2, 4, ..., 14)
R	Register or working register	reg or Rn (reg = 0–255, n = 0–15)
RR	Register pair or working register pair	reg or RRp (reg = 0–254, even number only, where p = 0, 2, ..., 14)
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 0–8191, where p = 0, 2, ..., 14)
da	Direct addressing mode	addr (addr = range 0–8191)
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)

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	
P	1	RLC R1	RLC IR1	ADC r1,r2	ADC r1,Ir2	ADC R2,R1	ADC IR2,R1	ADC R1,IM	
P	2	INC R1	INC IR1	SUB r1,r2	SUB r1,Ir2	SUB R2,R1	SUB IR2,R1	SUB R1,IM	
E	3	JP IRR1		SBC r1,r2	SBC r1,Ir2	SBC R2,R1	SBC IR2,R1	SBC R1,IM	
R	4			OR r1,r2	OR r1,Ir2	OR R2,R1	OR IR2,R1	OR R1,IM	
	5	POP R1	POP IR1	AND r1,r2	AND r1,Ir2	AND R2,R1	AND IR2,R1	AND R1,IM	
N	6	COM R1	COM IR1	TCM r1,r2	TCM r1,Ir2	TCM R2,R1	TCM IR2,R1	TCM R1,IM	
I	7	PUSH R2	PUSH IR2	TM r1,r2	TM r1,Ir2	TM R2,R1	TM IR2,R1	TM R1,IM	
B	8								LD r1, x, r2
B	9	RL R1	RL IR1						LD r2, x, r1
L	A			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		LDC r1,Irr2				LD r1, Ir2
H	D	SRA R1	SRA IR1		LDC r2,Irr1			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					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		JR cc,RA	LD r1,IM	JP cc,DA	INC r1	
P	1	–	–		–	–	–	–	
P	2								
E	3								
R	4								
	5								
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		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 (1)	C	Carry	C = 1
1111 (1)	NC	No carry	C = 0
0110 (1)	Z	Zero	Z = 1
1110 (1)	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 (1)	EQ	Equal	Z = 1
1110 (1)	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 (1)	UGE	Unsigned greater than or equal	C = 0
0111 (1)	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 SAM88RCRI 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.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode 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.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode 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".

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode 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.

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

Examples: Given: R0 = 15H, R1 = 21H, PC = 1A47H, and SP = 0B2H:

CALL 1521H ® SP = 0B0H
(Memory locations 00H = 1AH, 01H = 4AH, where 4AH is the address that follows the instruction.)

CALL @RR0 ® SP = 0B0H (00H = 1AH, 01H = 49H)

In the first example, if the program counter value is 1A47H and the stack pointer contains the value 0B2H, the statement "CALL 1521H" pushes the current PC value onto the top of the stack. The stack pointer now points to memory location 00H. The PC is then loaded with the value 1521H, 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 01H (because the two-byte instruction format was used). The PC is then loaded with the value 1521H, the address of the first instruction in the program sequence to be executed.

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

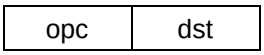
COM dst

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".

Format:



Bytes	Cycles	Opcode (Hex)	Addr Mode 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, 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.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode 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.

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, that is, dst value is – 128 (80H) and result value is + 127 (7FH); cleared otherwise.

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.

DI — Disable Interrupts

DI

Operation: SYM (3) _ 0

Bit zero of the system mode register, SYM.3, 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 = 08H:

DI

If the value of the SYM register is 08H, the statement "DI" leaves the new value 00H in the register and clears SYM.3 to "0", disabling interrupt processing.

EI — Enable Interrupts

EI

Operation: SYM (3) _ 1

An EI instruction sets bit 3 of the system mode register, SYM.3 to "1". This allows interrupts to be serviced as they occur. 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 08H, enabling all interrupts. (SYM.3 is the enable bit for global interrupt processing.)

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.

Flags: No flags are affected.

Format:

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

Example: The instruction

```
IDLE
NOP
NOP
NOP
```

stops the CPU clock but not the system clock.

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, that is dst value is + 127 (7FH) and result is – 128 (80H); cleared otherwise.

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.

IRET — Interrupt Return

IRET IRET

Operation:
FLAGS _ @SP
SP _ SP + 1
PC _ @SP
SP _ SP + 2
SYM(2) _ 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.

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	BF
		12	

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 op code 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:

(note)	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 op code 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

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, R4 = 00H, R5 = 60H; Program memory locations 0061 = AAH, 0103H = 4FH, 0104H = 1A, 0105H = 6DH, and 1104H = 88H. External data memory locations 0061H = BBH, 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[RR4]     ;  R0 _ contents of program memory location 0061H
                         ;  (01H + RR4),
                         ;  R0 = AAH, R2 = 00H, R3 = 60H

LDE     R0,#01H[RR4]     ;  R0 _ contents of external data memory location 0061H
                         ;  (01H + RR4), R0 = BBH, R4 = 00H, R5 = 60H

LDC (note) #01H[RR4],R0  ;  11H (contents of R0) is loaded into program memory location
                         ;  0061H (01H + 0060H)

LDE     #01H[RR4],R0     ;  11H (contents of R0) is loaded into external data memory
                         ;  location 0061H (01H + 0060H)

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
```

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".

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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:

opc	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode dst
2	8	50	R
	8	51	IR

Examples: Given: Register 00H = 01H, register 01H = 1BH, SP (0D9H) = 0BBH, and stack register 0BBH = 55H:

POP 00H ® Register 00H = 55H, SP = 0BCH

POP @00H ® Register 00H = 01H, register 01H = 55H, SP = 0BCH

In the first example, general register 00H contains the value 01H. The statement "POP 00H" loads the contents of location 0BBH (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 0BCH.

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	70	R
		8	71	IR

Examples: Given: Register 40H = 4FH, register 4FH = 0AAH, SP = 0C0H:

PUSH 40H ® Register 40H = 4FH, stack register 0BFH = 4FH, SP = 0BFH

PUSH @40H ® Register 40H = 4FH, register 4FH = 0AAH, stack register 0BFH = 0AAH, SP = 0BFH

In the first example, if the stack pointer contains the value 0C0H, and general register 40H the value 4FH, the statement "PUSH 40H" decrements the stack pointer from 0C0 to 0BFH. It then loads the contents of register 40H into location 0BFH. Register 0BFH then contains the value 4FH and SP points to location 0BFH.

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 10	AF

Example: Given: SP = 0BCH, (SP) = 101AH, and PC = 1234:

RET ® PC = 101AH, SP = 0BEH

The statement "RET" pops the contents of stack pointer location 0BCH (10H) into the high byte of the program counter. The stack pointer then pops the value in location 0BDH (1AH) into the PC's low byte and the instruction at location 101AH is executed. The stack pointer now points to memory location 0BEH.

RL — Rotate Left

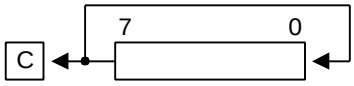
RL dst

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, that is, if the sign of the destination changed during rotation; cleared otherwise.

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

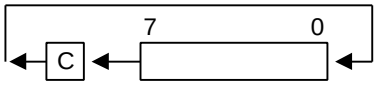
RLC dst

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.

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

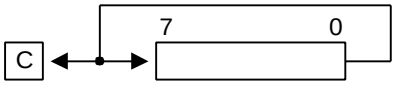
RR dst

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.

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

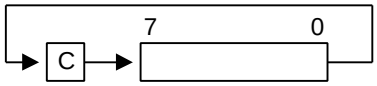
RRC dst

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.

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".

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.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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

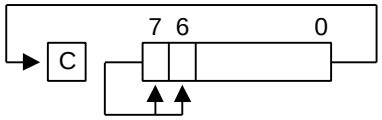
SRA dst

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".

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.

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 External interrupt input. For the reset operation, the RESET pin must be held to Low level until the required oscillation stabilization interval has elapsed.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	1	4	7F	–	–

Example: The statement

```
LD      STOPCON, #0A5H
STOP
NOP
NOP
NOP
```

halts all microcontroller operations. When STOPCON register is not #0A5H value, if you use STOP instruction, PC is changed to reset address.

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.

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.

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".

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	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".

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.

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".

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode 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.

7 CLOCK CIRCUIT

OVERVIEW

The clock frequency generation for the S3C9498/F9498 by an external crystal can range from 1 MHz to 8 MHz. The maximum CPU clock frequency is 8 MHz. The X_{IN} and X_{OUT} pins connect the external oscillator or clock source to the on-chip clock circuit.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

— External crystal or ceramic resonator 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

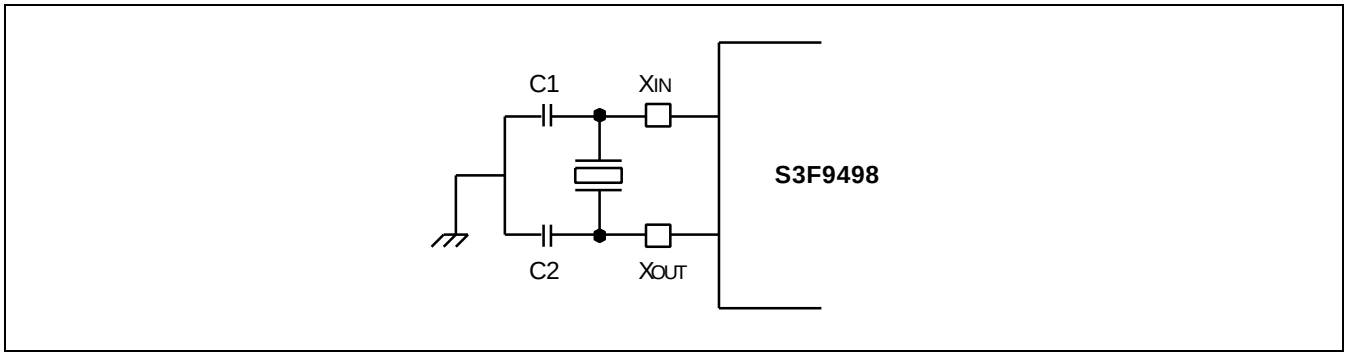


Figure 7-1. Main Oscillator Circuit (Crystal or Ceramic Oscillator)

CLOCK STATUS DURING POWER-DOWN MODES

The two power-down modes, Stop mode and Idle mode, affect clock oscillation as follows:

— In Stop mode, the main oscillator "freezes", halting the CPU and peripherals. The contents of the register file and current system register values are retained. Stop mode is released, and the oscillator started, by a reset operation or by an external interrupt with RC-delay noise filter (for S3C9498/F9498, INT0-INT1).
— In Idle mode, the internal clock signal is gated off to the CPU, but not to interrupt control and the timer. The current CPU status is preserved, including stack pointer, program counter, and flags. Data in the register file is retained. Idle mode is released by a reset or by an interrupt (external or internally-generated).

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

The system clock control register, CLKCON, is located at address D0H. 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 to fxx/8, fxx/2, or fxx/1.

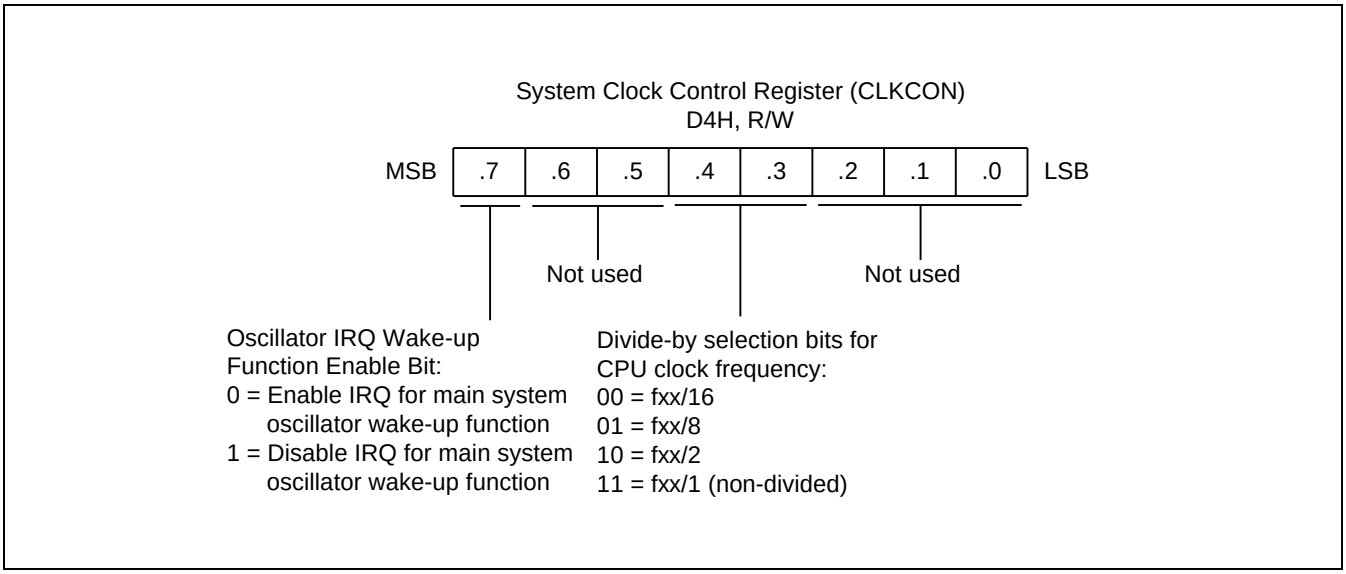


Figure 7-2. System Clock Control Register (CLKCON)

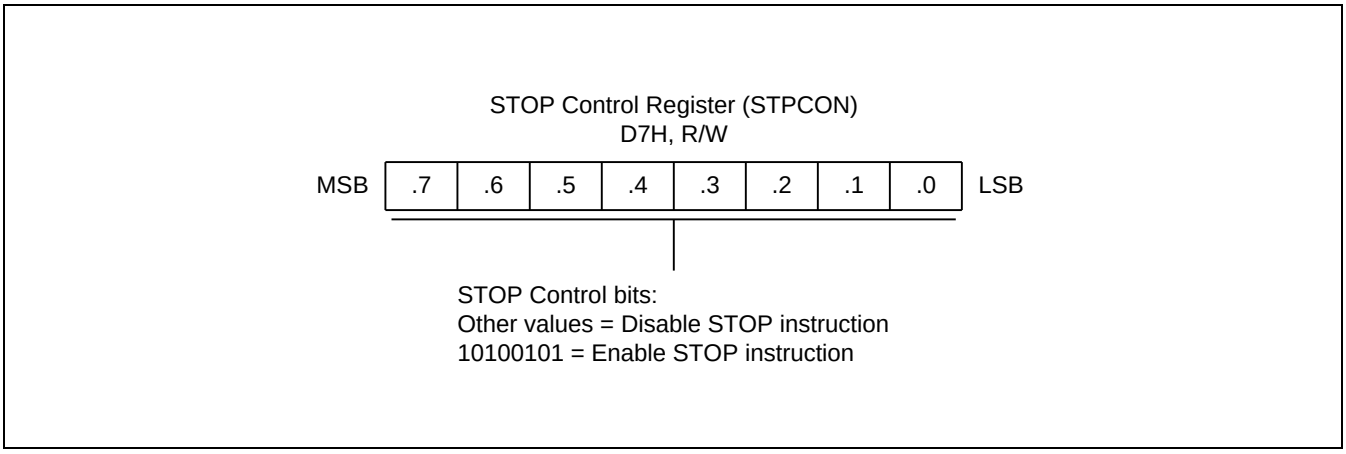


Figure 7-3. STOP Control Register (STPCON)

8 RESET and POWER-DOWN

SYSTEM RESET

OVERVIEW

During a power-on Reset, the voltage at V_{DD} goes to High level and the nRESET 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 S3C9498/F9498 into a known operating status.

To allow time for internal CPU clock oscillation to stabilize, the nRESET pin must be held to Low level for a minimum time interval after the power supply comes within tolerance. The minimum required oscillation stabilization time for a Reset operation is 1millisecond.

Whenever a reset occurs during normal operation (that is, when both V_{DD} and nRESET are High level), the nRESET pin is forced Low 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:

— Interrupt is disabled.

— The watchdog function is enabled.

— Ports 0-3 are set to input mode.

— Peripheral control and data registers 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 8 on-chip ROM. (The external interface is not automatically configured).

HARDWARE RESET VALUES

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. S3C9498/F9498Registers Values after RESET (Continued)

Register Name	Mnemonic	Address		Bit Values After RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Timer C control register	TCCON	208	D0H	0	0	0	0	0	0	0	0
Timer D control register	TDCON	209	D1H	0	0	0	0	0	0	0	0
Timer C data register register	TCDATA	210	D2H	1	1	1	1	1	1	1	1
Timer D data register register	TDDATA	211	D3H	1	1	1	1	1	1	1	1
System Clock control register	CLKCON	212	D4H	0	–	–	0	0	–	–	–
System flags register	FLAGS	213	D5H	x	x	x	x	–	–	–	–
UART Baud rate data register	BRDATA	214	D6H	1	1	1	1	1	1	1	1
STOP control register	STPCON	215	D7H	0	0	0	0	0	0	0	0
Timer Counter selection register	TCNTSEL	216	D8H	–	–	–	–	–	0	0	0
Stack pointer register	SP	217	D9H	x	x	x	x	x	x	x	x
Timer counter register	TCNT	218	DAH	x	x	x	x	x	x	x	x
Location DBH is not mapped											
Basic timer control register	BTCON	220	DCH	0	0	0	0	0	0	0	0
Basic timer counter register	BTCNT	221	DDH	0	0	0	0	0	0	0	0
Location DEH is not mapped											
System mode register	SYM	223	DFH	–	–	–	–	0	0	0	0

Table 8-2. S3C9498/F9498Registers Values after RESET (Concluded)

Register Name	Mnemonic	Address		Bit Values After RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 0 Data Register	P0	224	E0H	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	225	E1H	0	0	0	0	0	0	0	0
Port 2 Data Register	P2	226	E2H	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	227	E3H	0	0	0	0	0	0	0	0
PWM data register	PWMDATA	228	E4H	–	–	1	1	1	1	1	1
PWM Extension data register	PWMEX	229	E5H	1	1	1	1	1	1	–	–
Port 0 control register	P0CON	230	E6H	–	–	0	0	0	0	0	0
P1 interrupt control register	P1INT	231	E7H	–	–	0	0	0	0	0	0
Port 1 control High register	P1CONH	232	E8H	0	0	0	0	0	0	0	0
Port 1 control Low register	P1CONL	233	E9H	0	0	0	0	0	0	0	0
Port 2 control High register	P2CONH	234	EAH	0	0	0	0	0	0	0	0
Port 2 control Low register	P2CONL	235	EBH	0	0	0	0	0	0	0	0
Port 3 control register	P3CON	236	ECH	0	0	0	0	0	0	0	0
PWM control register	PWMCON	237	EDH	0	0	0	0	0	0	0	0
Timer 1 data register(high byte)	T1DATAH	238	EEH	1	1	1	1	1	1	1	1
Timer 1 data register(low byte)	T1DATAL	239	EFH	1	1	1	1	1	1	1	1
Timer 1 control register	T1CON	240	F0H	0	0	0	0	0	0	0	0
Serial I/O control register	SIOCON	241	F1H	0	0	0	0	0	0	0	0
Timer Interrupt pending register	TINTPND	242	F2H	–	0	0	0	0	0	0	0
Timer A control register	TACON	243	F3H	0	0	0	0	0	0	0	0
SIO pre-scalar register	SIOPS	244	F4H	0	0	0	0	0	0	0	0
Timer A data register	TADATA	245	F5H	0	0	0	0	0	0	0	0
Timer B data register	TBDATA	246	F6H	0	0	0	0	0	0	0	0
Location F7H is not mapped											
Timer B control register	TBCON	248	F8H	0	0	–	0	0	0	0	0
SIO data register	SIODATA	249	F9H	0	0	0	0	0	0	0	0
A/D converter data register(high byte)	ADDATAH	250	FAH	x	x	x	x	x	x	x	x
A/D converter data register(low byte)	ADDATAL	251	FBH	–	–	–	–	–	–	x	x
A/D converter control register	ADCON	252	FCH	0	0	0	0	0	0	0	0
UART control register	UARTCON	253	FDH	0	0	0	0	0	0	0	0
UART pending register	UARTPND	254	FEH	–	–	–	–	–	–	0	0
UART data register	UDATA	255	FFH	x	x	x	x	x	x	x	x

NOTE: – : Not mapped or not used, x: Undefined.

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 main oscillator 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 interrupts.

NOTE

Do not use stop mode if you are using an external clock source because X_{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 nRESET 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 (1/16) 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 (and 0101H).

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 S3C9498/F9498interrupt structure that can be used to release Stop mode are:

— External interrupts P1.0-P1.1 (INT0-INT1)

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 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.

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 timers 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.

NOTES

9 I/O PORTS

OVERVIEW

The S3C9498/F9498 microcontroller has four bit-programmable I/O ports, P0-P3. The port 0 and 3 are 3-bit /7-bits ports and the others are 8-bit ports. This gives a total of 22/24/26 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.

Table 9-1 gives you a general overview of the S3C9498/F9498 I/O port functions.

Table 9-1. S3C9498/F9498 Port Configuration Overview

Port	Configuration Options
0	I/O port with bit-programmable pins. Configurable to input or push-pull output mode. Pull-up resistors can be assigned by software. Pins can also be assigned individually as alternative function pins.
1	I/O port with bit-programmable pins. Configurable to input or push-pull output mode. Pull-up resistors can be assigned by software. Pins can also be assigned individually as alternative function pins.
2	I/O port with bit-programmable pins. Configurable to input mode, push-pull output mode. Pins can also be assigned individually as alternative function pins.
3	I/O port with bit-programmable pins. Configurable to input mode, push-pull output mode. Pins can also be assigned individually as alternative function pins.

PORT DATA REGISTERS

Table 9-2 gives you an overview of the register locations of all five S3C9498/F9498I/O port data registers. Data registers for ports 0, 1, 2, and 3 have the general format shown in Figure 9-1.

Table 9-2. Port Data Register Summary

Register Name	Mnemonic	Decimal	Hex	R/W
Port 0 data register	P0	224	E0H	R/W
Port 1 data register	P1	225	E1H	R/W
Port 2 data register	P2	226	E2H	R/W
Port 3 data register	P3	227	E3H	R/W

PORT 0

Port 0 is a 3-bit I/O Port that you can use two ways:

— General-purpose I/O

— Alternative function

Port 0 is accessed directly by writing or reading the port 0 data register, P0 at location E0H.

Port 0 Control Register (P0CON)

Port 0 pins are configured individually by bit-pair settings in three control registers located : P0CON.

When you select output mode, a push-pull or an open-drain circuit is configured. In input mode, many different selections are available:

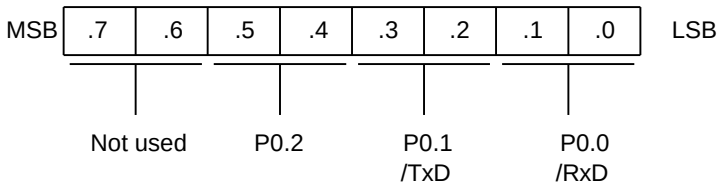
— Input mode.

— Output mode(Push-pull or Open-drain)

— Alternative function: UART module – TXD/RXD

— Alternative function: RESETB

Port 0 Control Register (P0CON)
E6H, R/W, Reset value:00H



.7 .6 bit

XX	Not used for 9498

.5 .4 bit/P0.2

00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Open-drain Output

.3 .2 bit/P0.1/TxD

00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Alternative function: TxD Output

.1 .0 bit/P0.0/RxD

00	Input mode with pull-up; RxD output
01	Input mode; RxD output
10	Push-pull output
11	Alternative function: RxD output

NOTE: When users use Port 0, users must be care of the pull-up resistance status register value.

Figure 9-1. Port 0 High-Byte Control Register (P0CON)

PORT 1

Port 1 is an 8-bit I/O port that you can use two ways:

— General-purpose I/O

— Alternative function

Port 1 is accessed directly by writing or reading the port 1 data register, P1 at location E1H.

Port 1 Control Register (P1CONH, P1CONL)

Port 1 pins are configured individually by bit-pair settings in three control registers located:
P1CONL(low byte, E9H) and P1CONH(high byte, E8H).

When you select output mode, a push-pull circuit is configured. In input mode, many different selections are available:

— Input mode.

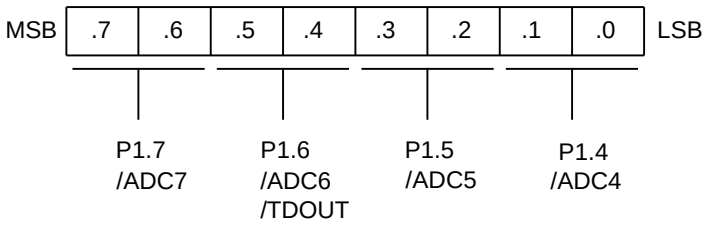
— Push-pull output mode

— Alternative function: External Interrupt – INT0, INT1

— Alternative function: Timer D output- TDOUT

— Alternative function: ADC input mode – ADC0, ADC1, ADC2, ADC3, ADC4, ADC5, ADC6, ADC7

Port 1 Control Register, High Byte (P1CONH)
E8H, R/W, Reset value:00H



.7 .6 bit/P1.7/ADC7

00	Input mode with pull-up
01	Not used for S3F9498
10	Push-pull output
11	Alternative function: ADC7 input

.5 .4 bit/P1.6/ADC6

00	Input mode with pull-up
01	Alternative function: TDOUT mode
10	Push-pull output
11	Alternative function: ADC6 input

.3 .2 bit/P1.5/ADC5

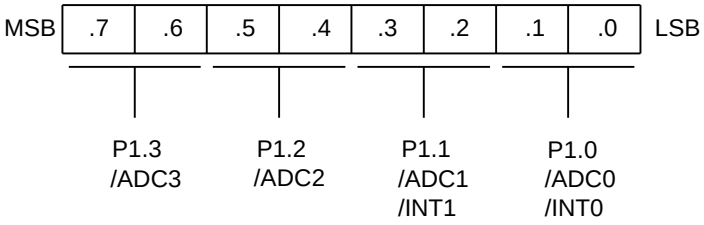
00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Alternative function: ADC5 input

.1 .0 bit/P1.4/ADC4

00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Alternative function: ADC4 input

Figure 9-2. Port 1 High-Byte Control Register (P1CONH)

Port 1 Control Register, Low Byte (P1CONL)
E9H, R/W, Reset value:00H



.7 .6 bit/P1.3/ADC3

00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Alternative function: ADC3 input

.5 .4 bit/P1.2/ADC2

00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Alternative function: ADC2 input

.3 .2 bit/P1.1/ADC1/INT1

00	Input mode with pull-up; INT1 input
01	Input mode; INT1 input
10	Push-pull output
11	Alternative function: ADC1 input

.1 .0 bit/P1.0/ADC0/INT0

00	Input mode with pull-up; INT0 input
01	Input mode; INT0 input
10	Push-pull output
11	Alternative function: ADC0 input

Figure 9-3. Port 1 Low-Byte Control Register (P1CONL)

Port 1 External Interrupt Register (P1INT)
E7H, R/W, Reset value:00H

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	Not used		INT1		INT0		INT1	INT0	

.7 .6 bits Not used for S3F9498

.5 .4 bits INT1 Interrupt Enable/Disable Selection

0x	Interrupt disable
10	Interrupt enable; falling edge
11	Interrupt enable; rising edge

.3 .2 bits INT0 Interrupt Enable/Disable Selection

0x	Interrupt disable
10	Interrupt enable; falling edge
11	Interrupt enable; rising edge

.1 bits INT1 Pending bit

0	No interrupt Pending *(clear pending bit when write)*
1	Interrupt enable

.0 bits INT0 Pending bit

0	No interrupt Pending *(clear pending bit when write)*
1	Interrupt enable

Figure 9-4. Port 1 Interrupt Control Register P1PND)

PORT 2

Port 2 is an 8-bit I/O port that you can use two ways:

— General-purpose I/O

— Alternative function

Port 2 is accessed directly by writing or reading the port 2 data register, P2 at location E2H.

Port 2 Control Register (P2CONH, P2CONL)

Port 2 pins are configured individually by bit-pair settings in two control registers located :
P2CONL (low byte, EBH) and P2CONH (high byte, EAH).

When you select output mode, a push-pull, an open-drain circuit is configured. In input mode, many different selections are available:

— Input mode.

— Output mode(Push-pull or Open-drain)

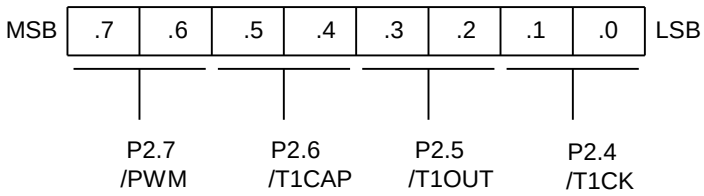
— Alternative function: Timer A signal in/out mode – TAOUT, TACAP, TACK

— Alternative function: Timer B signal out mode – TBOUT

— Alternative function: Timer 1 signal in/out mode – T1OUT, T1CAP, T1CK

— Alternative function: PWM out mode – PWM

Port 2 Control Register, High Byte (P2CONH)
EAH, R/W, Reset value:00H



.7 .6 bit/P2.7/PWM

00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Alternative function: PWM signal output

.5 .4 bit/P2.6/T1CAP

00	Input mode with pull-up; T1CAP input
01	Input mode; T1CAP input
10	Push-pull output
11	Open-drain output

.3 .2 bit/P2.5/T1OUT

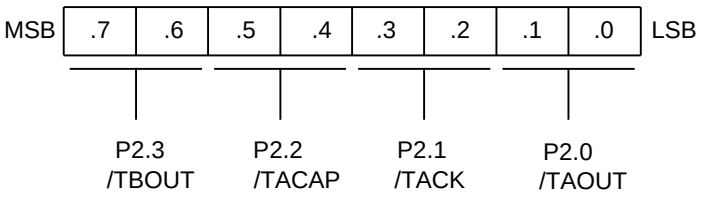
00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Alternative function: T1OUT signal output

.1 .0 bit/P2.4/T1CK

00	Input mode with pull-up; T1CK input
01	Input mode;T1CK input
10	Push-pull output
11	Open-drain output

Figure 9-5. Port 2 High-Byte Control Register (P2CONH)

Port 2 Control Register, Low Byte (P2CONL)
EBH, R/W, Reset value:00H



.7 .6 bit/P2.3/TBOUT

00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Alternative function: TBOUT signal output

.5 .4 bit/P2.2/TACAP

00	Input mode with pull-up; TACAP input
01	Input mode; TACAP input
10	Push-pull output
11	Open-drain output

.3 .2 bit/P2.1/TACK

00	Input mode with pull-up; TACK input
01	Input mode; TACK input
10	Push-pull output
11	Open-drain output

.1 .0 bit/P2.0/TAOUT

00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Alternative function: TAOUT signal output

Figure 9-6. Port 2 Low-Byte Control Register (P2CONL)

PORT 3

Port 3 is a 7-bit I/O Port that you can use two ways:

— General-purpose I/O

— Alternative function

Port 3 is accessed directly by writing or reading the port 3 data register, P3 at location E3H.

Port 3 Control / Interrupt Control Register (P3CON)

Port 3 pins are configured individually by bit-pair settings in two control registers located : P3CON (ECH).

When you select output mode, a push-pull or an open-drain circuit is configured. In input mode, many different selections are available:

— Input mode.

— Output mode(Push-pull or Open-drain)

— Alternative function: SIO module – SI/SO/SCK

Port 3 Control Register (P3CON)
ECH, R/W, Reset value:00H

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

P3.6/P3.5 P3.4/P3.3 — P3.2 / SCK — P3.1 / SO — P3.2 / SI

.7 .6 bit/P3.6/P3.5/P3.4/P3.3

00	Input mode with pull-up
01	Input mode
10	Push-pull output
11	Open-drain output

.5 .4 bit/P3.2/SCK

00	Input mode, pull-up (SCK input)
01	Input mode(SCK input)
10	Push-pull output
11	Alternative output mode (SCK output)

.5 .4 bit/P3.1/SO

00	Input mode, pull-up
01	Input mode
10	Push-pull output
11	Alternative output mode (SO)

.5 .4 bit/P3.0/SI

00	Input mode, pull-up (SI)
01	Input mode(SI)
10	Push-pull output
11	Open-drain output mode

Figure 9-7. Port 3 High-Byte Control Register (P3CON)

NOTES

10 BASIC TIMER

OVERVIEW

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 (f_{OSC} divided by 4096, 1024, or 128) with multiplexer

— 8-bit basic timer counter, BTCNT (DDH, read-only)

— Basic timer control register, BTCON (DCH, read/write)

BASIC TIMER (BT)

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.

A reset clears BTCON to "00H". This enables the watchdog function and selects a basic timer clock frequency of f_{OSC}/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, can be cleared during normal operation by writing a "1" to BTCON.1. To clear the frequency dividers for the basic timer input clock, you write a "1" to BTCON.0.

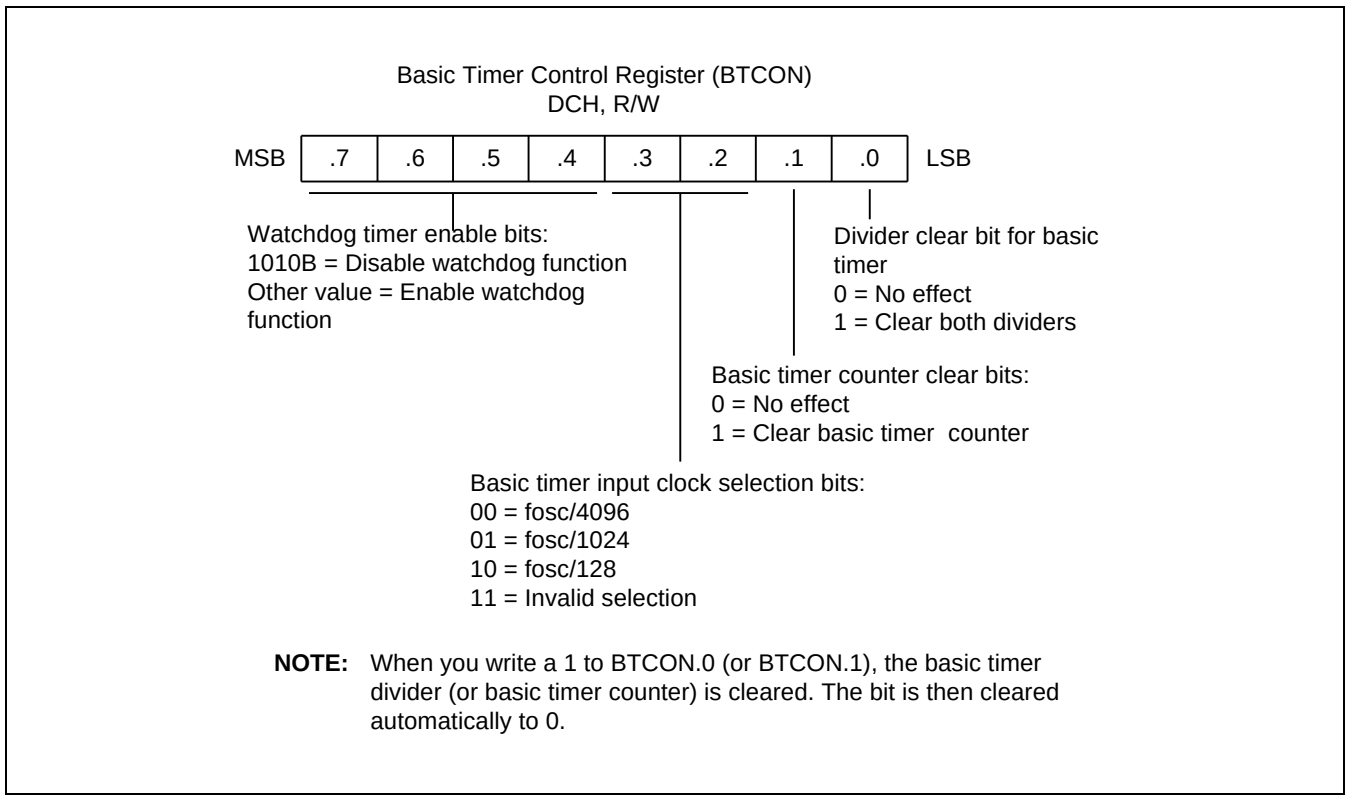


Figure 10-1. Basic Timer Control Register (BTCON)

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 oscillator clock 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

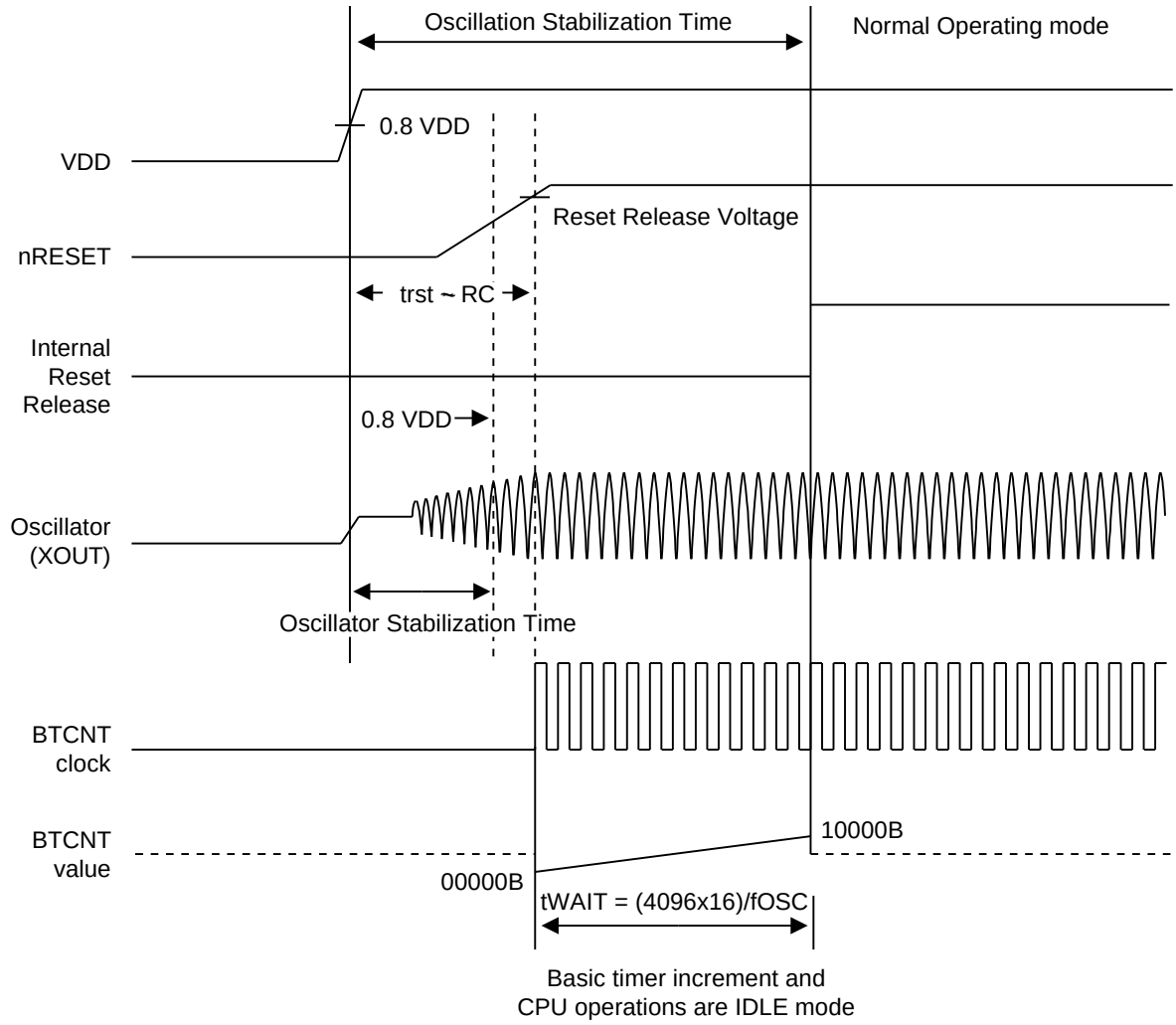
You can also use the basic timer to program a specific oscillation stabilization interval following a reset or when Stop mode has been released by an external interrupt.

In Stop mode, whenever a reset or an external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of $f_{OSC}/4096$ (for reset), or at the rate of the preset clock source (for an external interrupt). When BTCNT.4 is set, 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:

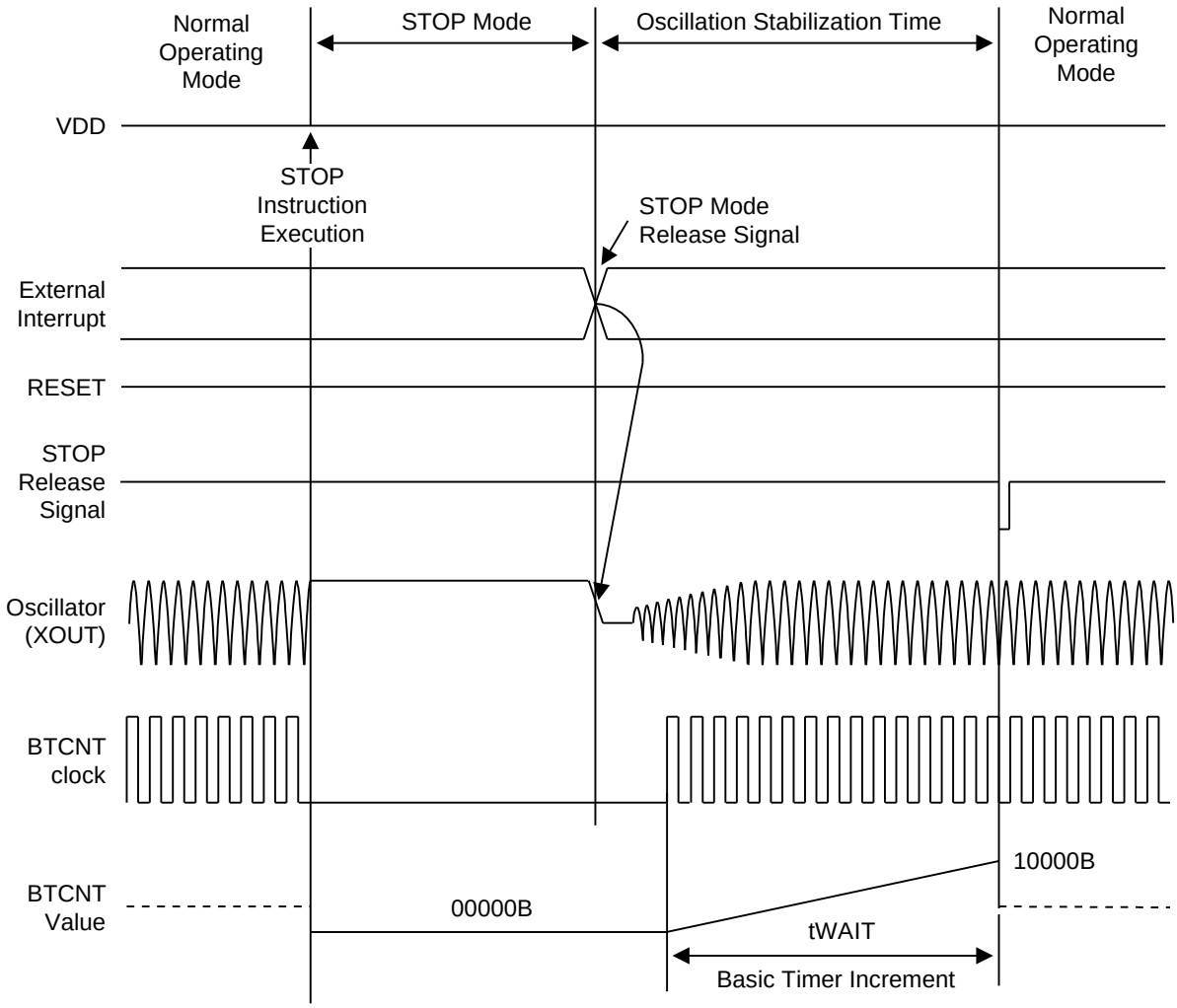
1. During Stop mode, an external power-on reset or an external interrupt occurs to trigger the Stop mode release and oscillation starts.
2. If an external power-on reset occurred, the basic timer counter will increase at the rate of $f_{OSC}/4096$. If an external interrupt is used to release Stop mode, the BTCNT value increases at the rate of the preset clock source.
3. Clock oscillation stabilization interval begins and continues until bit 4 of the basic timer counter is set.
4. When a BTCNT.4 is set, normal CPU operation resumes.

Figure 10-2 and 10-3 shows the oscillation stabilization time on RESET and STOP mode release



NOTE: Duration of the oscillator stabilization wait time, tWAIT, when it is released by a Power-on-reset is 4096 x 16/fOSC.
tRST ≈RC (R and C are value of external power on reset)

Figure 10-2. Oscillation Stabilization Time on RESET



NOTE: Duration of the oscillator stabilzation wait time, tWAIT, it is released by an interrupt is determined by the setting in basic timer control register, BTCON.

BTCON.3	BTCON.2	tWAIT	tWAIT (When fOSC is 8 MHz)
0	0	(4096 x 16)/fosc	8.19 ms
0	1	(1024 x 16)/fosc	2.05 ms
1	0	(128 x 16)/fosc	0.25 ms
1	1	Invalid setting	—

Figure 10-3. Oscillation Stabilization Time on STOP Mode Release

☞ PROGRAMMING TIP — Configuring the Basic Timer

This example shows how to configure the basic timer to sample specification.

```
                ORG     0000H
                VECTOR  00H,INT_9498            ;  S3C9498/F9498 has only one interrupt vector

;--------------<< Smart Option >>

                ORG     003CH
                DB      0FFH                    ;  003CH, must be initialized to 1
                DB      0FFH                    ;  003DH, must be initialized to 1
                DB      0C7H                    ;  003EH, enable LVR (3.0 V)
                DB      0FFH                    ;  003FH, RESET pin enable

;--------------<< Initialize System and Peripherals >>

                ORG     0100H

RESET:          DI                              ;  Disable interrupt
                LD      CLKCON,#00011000B       ;  Select non-divided CPU clock
                LD      SP,#0C0H                ;  Stack pointer must be set
                •
                •

                LD      BTCON,#02H              ;  Enable watchdog function
                                                ;  Basic timer clock: fOSC/4096
                                                ;  Basic counter (BTCNT) clear
                •
                •
                •
                EI                              ;  Enable interrupt

;--------------<< Main loop >>

MAIN:           •
                LD      BTCON,#02H              ;  Enable watchdog function
                                                ;  Basic counter (BTCNT) clear
                •
                •
                •
                JR      T,MAIN                  ;

;--------------<< Interrupt Service Routines >>

INT_9498:       •                               ;  Interrupt enable bit and pending bit check
                •                               ;
                •                               ;  Pending bit clear
                IRET                            ;
                •
                •
                END                             ;
```

11 8-BIT TIMER A/B

8-BIT TIMER A

OVERVIEW

The 8-bit timer A is an 8-bit general-purpose timer/counter. Timer A has three operating modes, you can select one of them using the appropriate TACON setting:

— Interval timer mode (Toggle output at TAOUT pin)
— Capture input mode with a rising or falling edge trigger at the TACAP pin
— PWM mode (TAOUT)

Timer A has the following functional components:

— Clock frequency divider (fxx divided by 1024, 256, or 64) with multiplexer
— External clock input pin (TACK)
— 8-bit counter (TACNT), 8-bit comparator, and 8-bit reference data register (TADATA)
— I/O pins for capture input (TACAP) or PWM or match output (TAOUT)
— Timer A overflow interrupt and match/capture interrupt generation
— Timer A control register, TACON (F3H, read/write)

FUNCTION DESCRIPTION

Timer A Interrupts

The timer A module can generate two interrupts: the timer A overflow interrupt (TAOVF), and the timer A match/ capture interrupt (TAINT).

Timer A overflow interrupt and match/capture interrupt pending conditions are cleared by software when it has been serviced.

Interval Timer Function

The timer A module can generate an interrupt: the timer A match interrupt (TAINT).

When timer A interrupt occurs and is serviced by the CPU, the pending condition is cleared by software.

In interval timer mode, a match signal is generated and TAOUT is toggled when the counter value is identical to the value written to the Timer A reference data register, TADATA. The match signal generates a timer A match interrupt and clears the counter.

If, for example, you write the value 10H to TADATA and 0AH to TACON, the counter will increment until it reaches 10H. At this point, the TA interrupt request is generated, the counter value is reset, and counting resumes.

Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the TAOUT pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer A 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 A overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TAOUT 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 $t_{CLK} \bullet 256$.

Capture Mode

In capture mode, a signal edge that is detected at the TACAP pin opens a gate and loads the current counter value into the Timer A data register. You can select rising or falling edges to trigger this operation.

Timer A also gives you capture input source: the signal edge at the TACAP pin. You select the capture input by setting the value of the timer A capture input selection bit in the Port 2 low–byte control register, P2CONL, (EBH). When P2CONL.5.4 is 00 and 01, the TACAP input or normal input is selected. When P2CONL.5.4 is set to 10 and 11, output is selected.

Both kinds of timer A interrupts can be used in capture mode: the timer A overflow interrupt is generated whenever a counter overflow occurs; the timer A match/capture interrupt is generated whenever the counter value is loaded into the Timer A data register.

By reading the captured data value in TADATA, and assuming a specific value for the timer A clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the TACAP pin.

TIMER A CONTROL REGISTER (TACON)

You use the timer A control register, TACON

— Select the timer A operating mode (interval timer, capture mode and PWM mode)

— Select the timer A input clock frequency

— Clear the timer A counter, TACNT

— Enable the timer A overflow interrupt or timer A match/capture interrupt

— Timer A start/stop

— Clear Timer A match/capture interrupt pending conditions

TACON is located at address F3H, and is read/write addressable using Register addressing mode.

A reset clears TACON to '00H'. This sets timer A to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all Timer A interrupts. You can clear the timer A counter at any time during normal operation by writing a "1" to TACON.3. You can start Timer A counter by writing a “1” to TACON.0.

The timer A overflow interrupt (TAOVF) has the vector address 00H-01H. When a timer A overflow interrupt occurs and is serviced by the CPU, but the pending condition must clear by software.

To enable the timer A match/capture interrupt , you must write TACON.1 to "1". To generate the exact time interval, you should write TACON.3 and TINTPND .0, which cleared counter and interrupt pending bit. When interrupt service routine is served, the pending condition must be cleared by software by writing a ‘0’ to the interrupt pending bit.

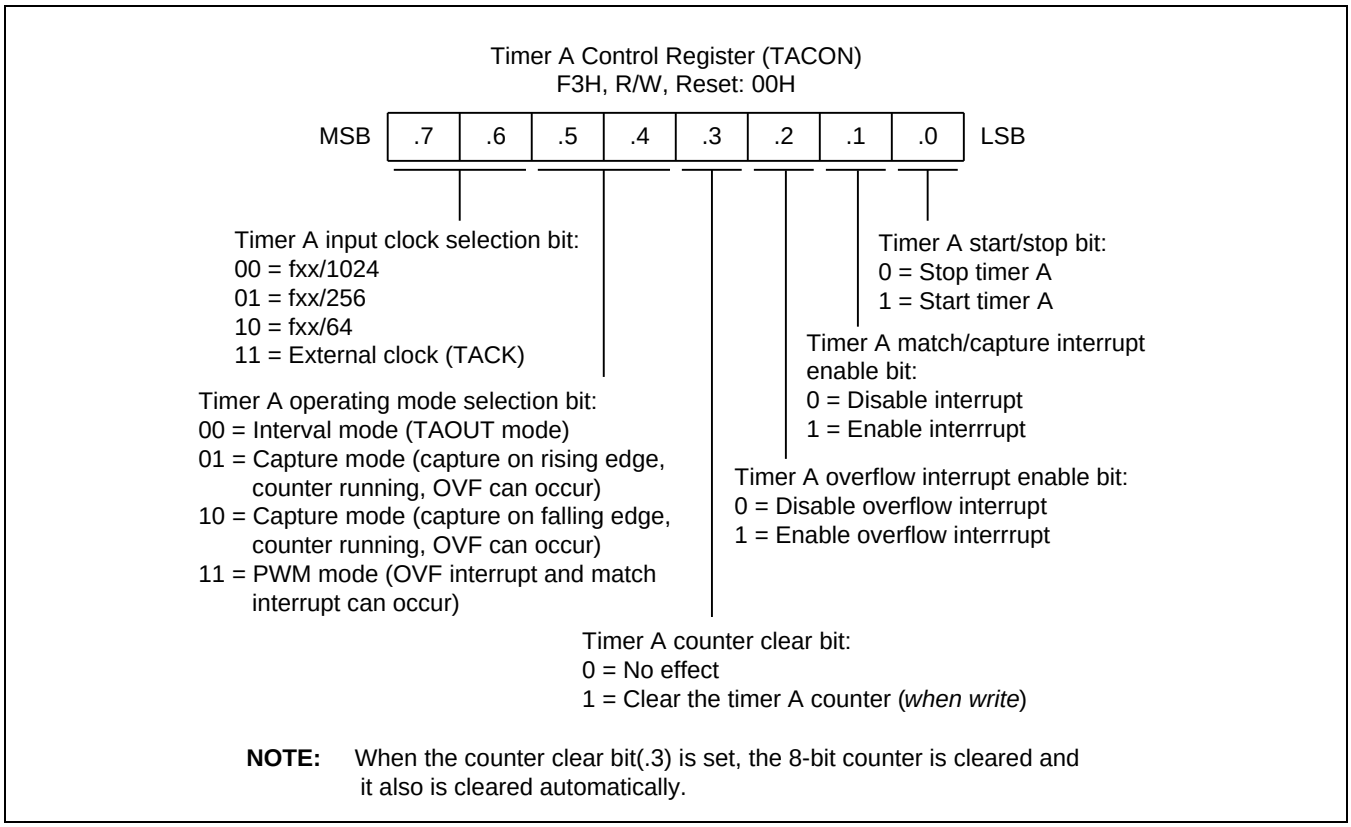


Figure 11-1. Timer A Control Register (TACON)

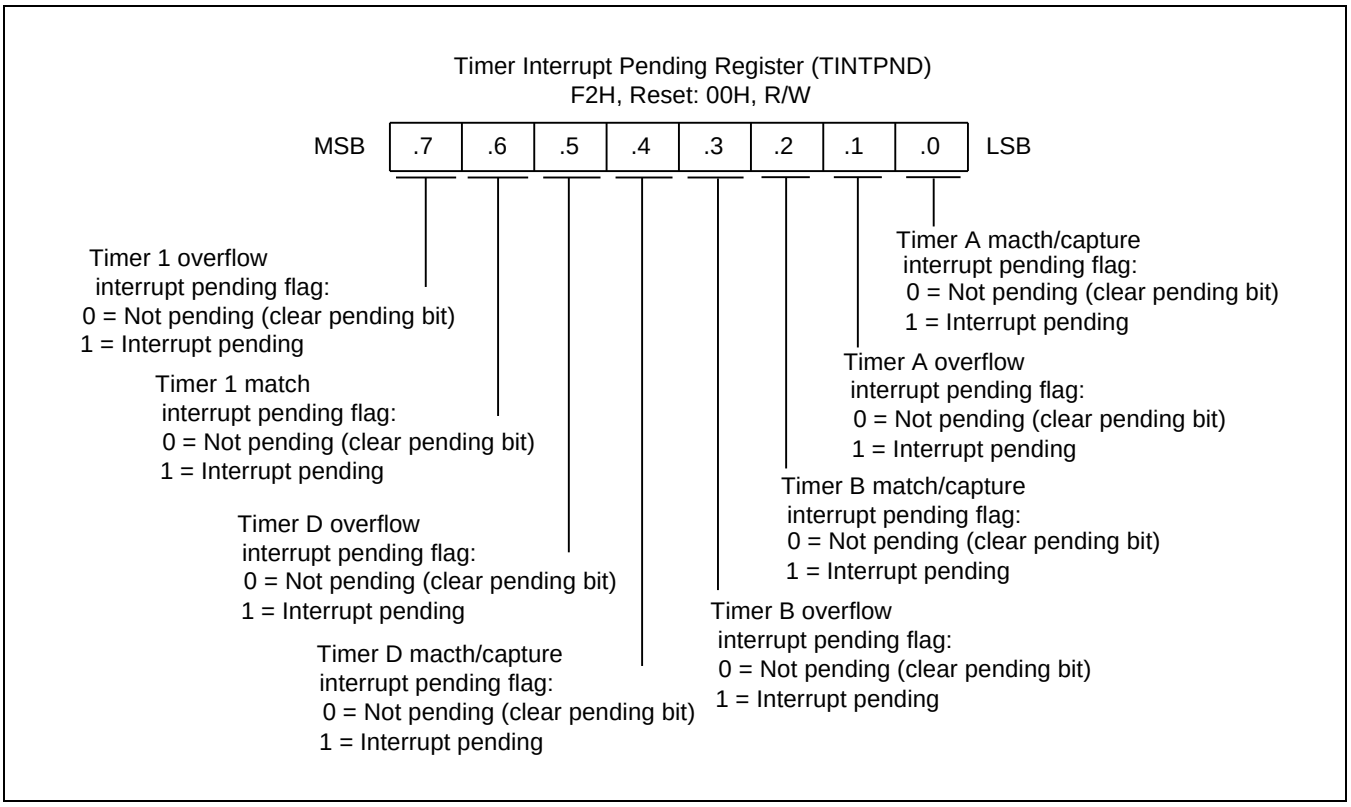


Figure 11-2. Timer interrupts Pending Register (TINTPND)

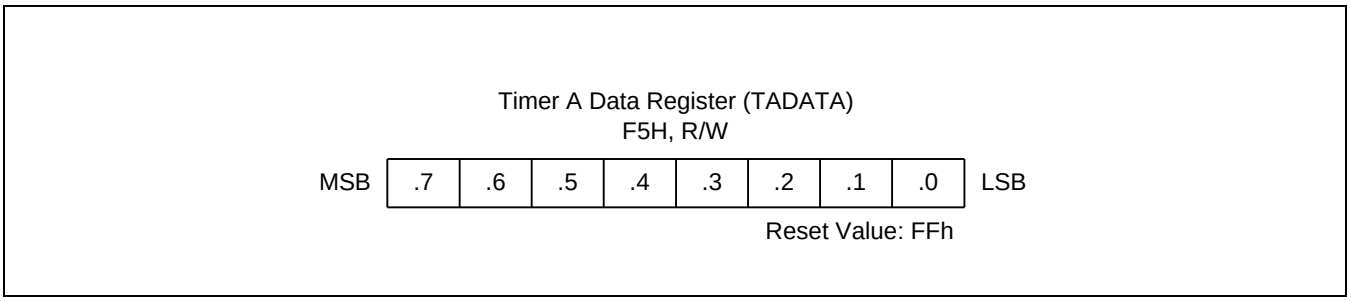


Figure 11-3. Timer A DATA Register (TADATA)

BLOCK DIAGRAM

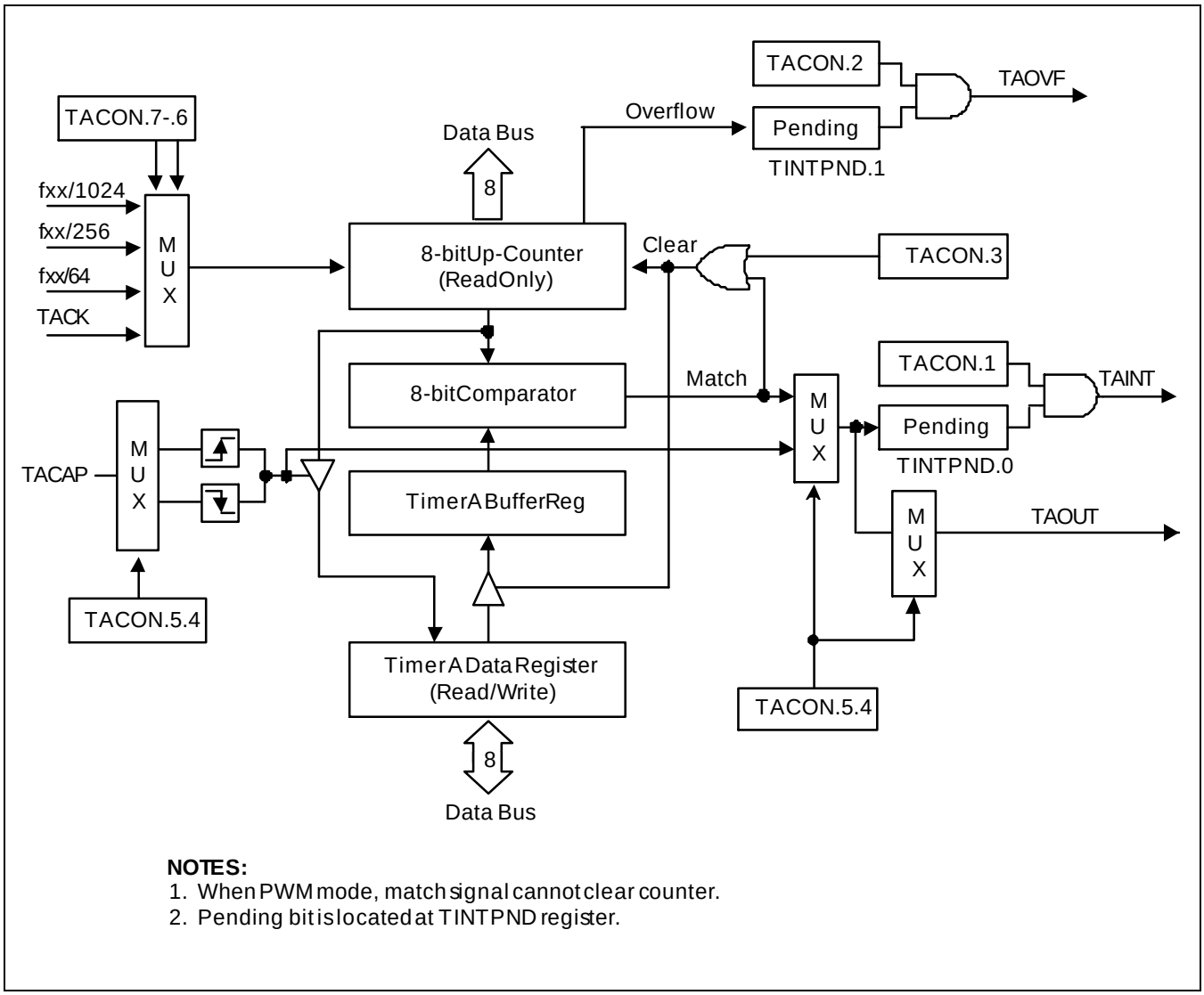
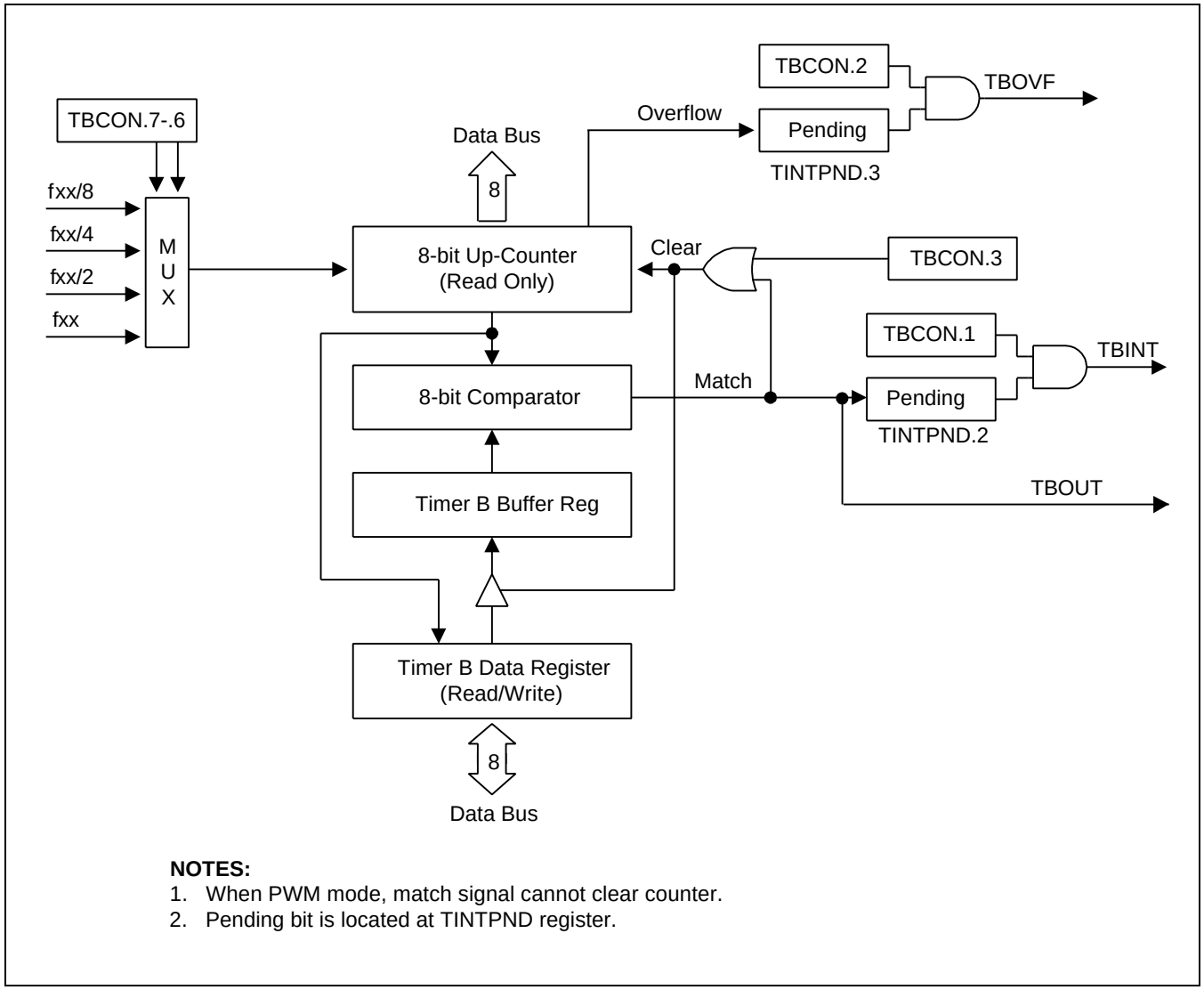


Figure 11-4. Timer A Functional Block Diagram

8-BIT TIMER B

OVERVIEW

The S3C9498/F9498 micro-controller has an 8-bit counter called timer B.



NOTES:
1. When PWM mode, match signal cannot clear counter.
2. Pending bit is located at TINTPND register.

Figure 11-5. Timer B Functional Block Diagram

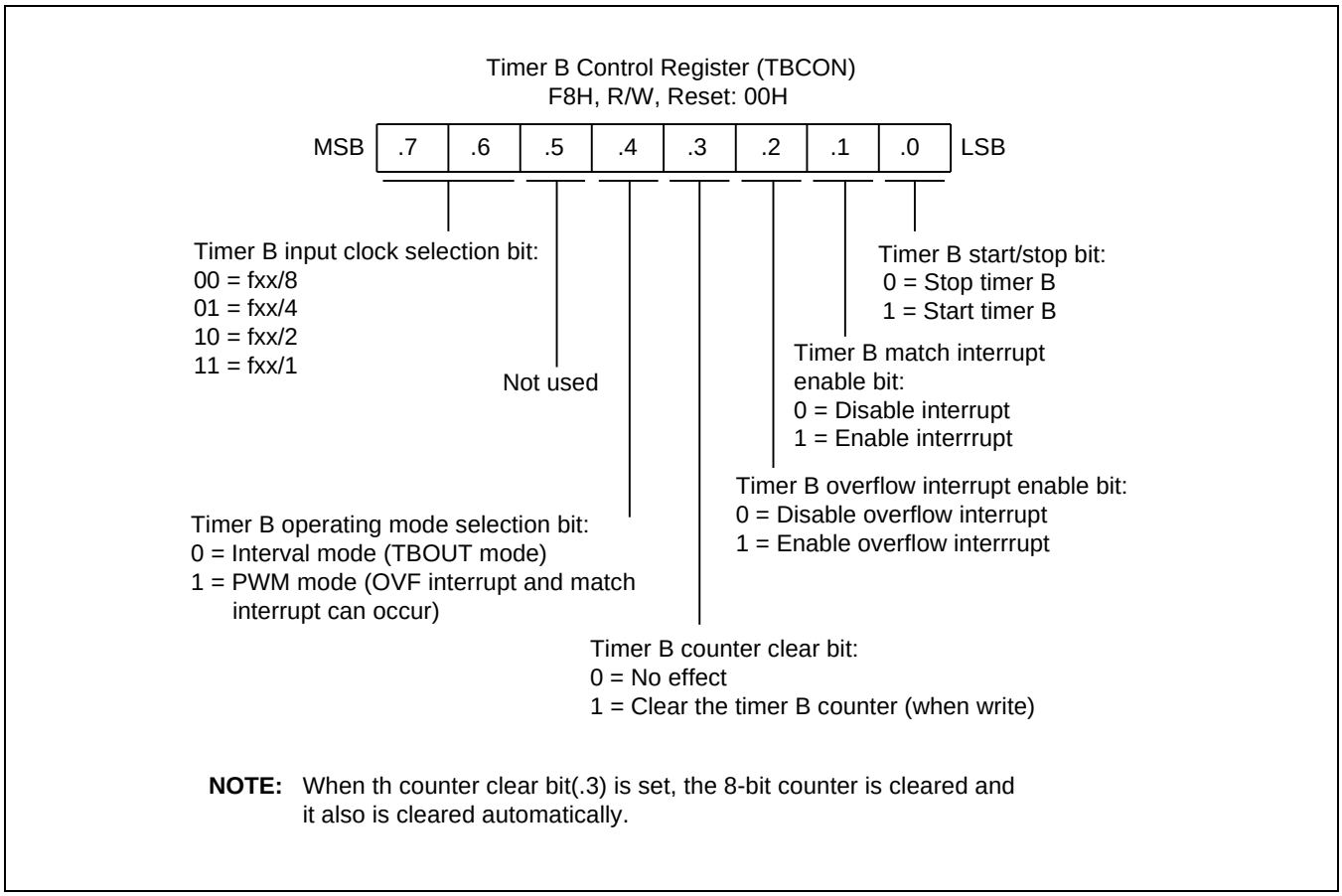


Figure 11-6. Timer B Control Register (TBCON)

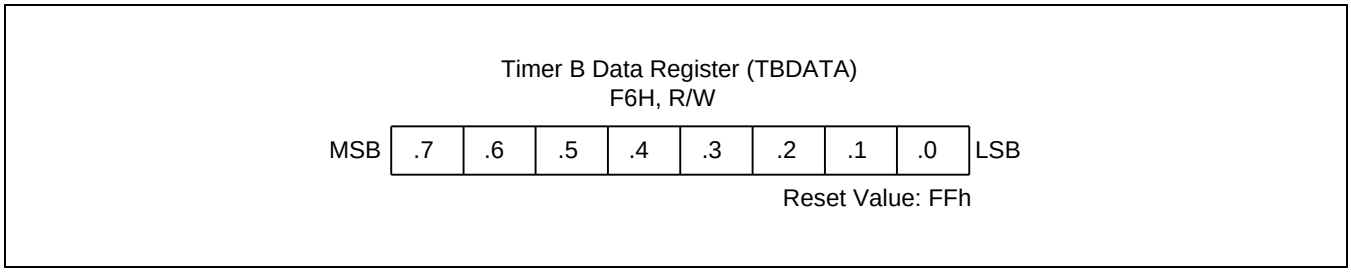


Figure 11-7. Timer B DATA Registers (TBDATA)

NOTES

12 16-BIT TIMER 1

OVERVIEW

The S3C9498/F9498 has two 16-bit timer/counters. The 16-bit timer 1 is a 16-bit general-purpose timer/counter. Timer 1 has three operating modes, one of which you select using the appropriate T1CON setting is:

— Interval timer mode (Toggle output at T1OUTpin)
— Capture input mode with a rising or falling edge trigger at the T1CAP pin
— PWM mode (T1PWM); PWM output shares their output port with T1OUT pin

Timer 1 has the following functional components:

— Clock frequency divider (fxx divided by 1024, 256, 64, 8, 1 or T1CK: External clock) with multiplexer
— External clock input pin (T1CK)
— A 16-bit counter , 16-bit comparator, and two 16-bit reference data register (T1DATAH/L)
— I/O pins for capture input (T1CAP), or match output (T1OUT)
— Timer 1 overflow interrupt and match/capture interrupt generation
— Timer 1 control register, T1CON

FUNCTION DESCRIPTION

Timer 1 Interrupts

The timer 1 module can generate two interrupts, the timer 1 overflow interrupt (T1OVF), and the timer 1 match/capture interrupt (T1INT).
A timer 1 overflow interrupt pending condition is cleared by software when it has been serviced.
A timer 1 match/capture interrupt, T1INT pending condition is also cleared by software when it has been serviced.

Interval Mode (match)

Timer 1 module can generate an interrupt: Timer 1 match interrupt (T1INT).
In interval timer mode, a match signal is generated and T1OUT is toggled when the counter value is identical to the value written to the T1 reference data register, T1DATAH/L. The match signal generates a timer 1 match interrupt (T1INT) and clears the counter.

Capture Mode

In capture mode for Timer 1, a signal edge that is detected at the T1CAP pin opens a gate and loads the current counter value into the T1 data register (T1DATAH/L for rising edge, or falling edge). You can select rising or falling edges to trigger this operation.
Timer 1 also gives you capture input source, the signal edge at the T1CAP pin. You select the capture input by setting the capture input selection bit in the port 2 control register, P2CONH,.
Both kinds of timer 1 interrupts (T1OVF, T1INT) can be used in capture mode, the timer 1 overflow interrupt is generated whenever a counter overflow occurs, the timer 1 capture interrupt is generated whenever the counter value is loaded into the T1 data register (T1DATAH/L).
By reading the captured data value in T1DATAH/L, and assuming a specific value for the timer 1 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the T1CAP pin.

PWM Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the T1OUT pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 1 data register. In PWM mode, however, the match signal does not clear the counter but can generate a match interrupt. The counter runs continuously, overflowing at FFFFH, and then continuous increasing from 0000H. Whenever an overflow is occurred, an overflow (OVF1) interrupt can be generated.

Although you can use the match or the overflow interrupt in the PWM mode, these interrupts are not typically used in PWM-type applications. Instead, the pulse at the T1OUT 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} .

TIMER 1 CONTROL REGISTER (T1CON)

You use the TIMER 1 control register, T1CON, to

— Select the TIMER 1 operating mode (interval timer, capture mode, or PWM mode)
— Select the TIMER 1 input clock frequency
— Clear the TIMER 1 counter.
— Enable the TIMER 1 overflow interrupt
— Enable the TIMER 1 match/capture interrupt

T1CON is located at address F0H, and is read/write addressable using Register addressing mode.

A reset clears T1CON to '00H'. This sets TIMER 1 to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all TIMER 1 interrupts.
You can clear the TIMER 1 counter at any time during normal operation by writing a "1" to T1CON.2. To generate the exact time interval, you should write "1" to T1CON.2 and clear appropriate pending bits of the TINTPND.6 register.

To detect a match/capture or overflow interrupt pending condition when T1INT or T1OVF is disabled, the application program should poll the pending bit T1CON and INTPND register. When a "1" is detected, a TIMER 1 match/capture or overflow interrupt is pending.
When the sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the interrupt pending bit.

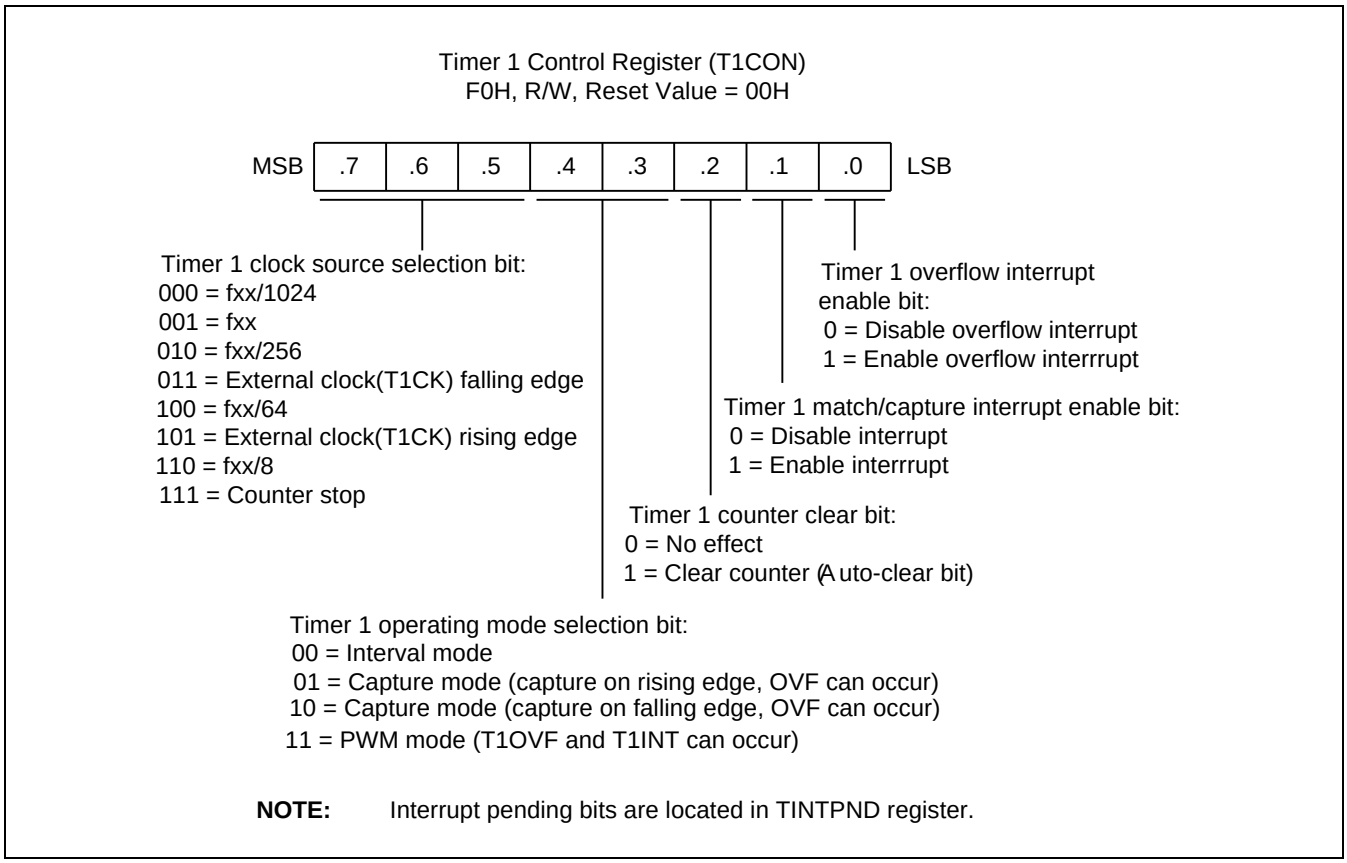


Figure 12-1. TIMER 1 Control Register (T1CON)

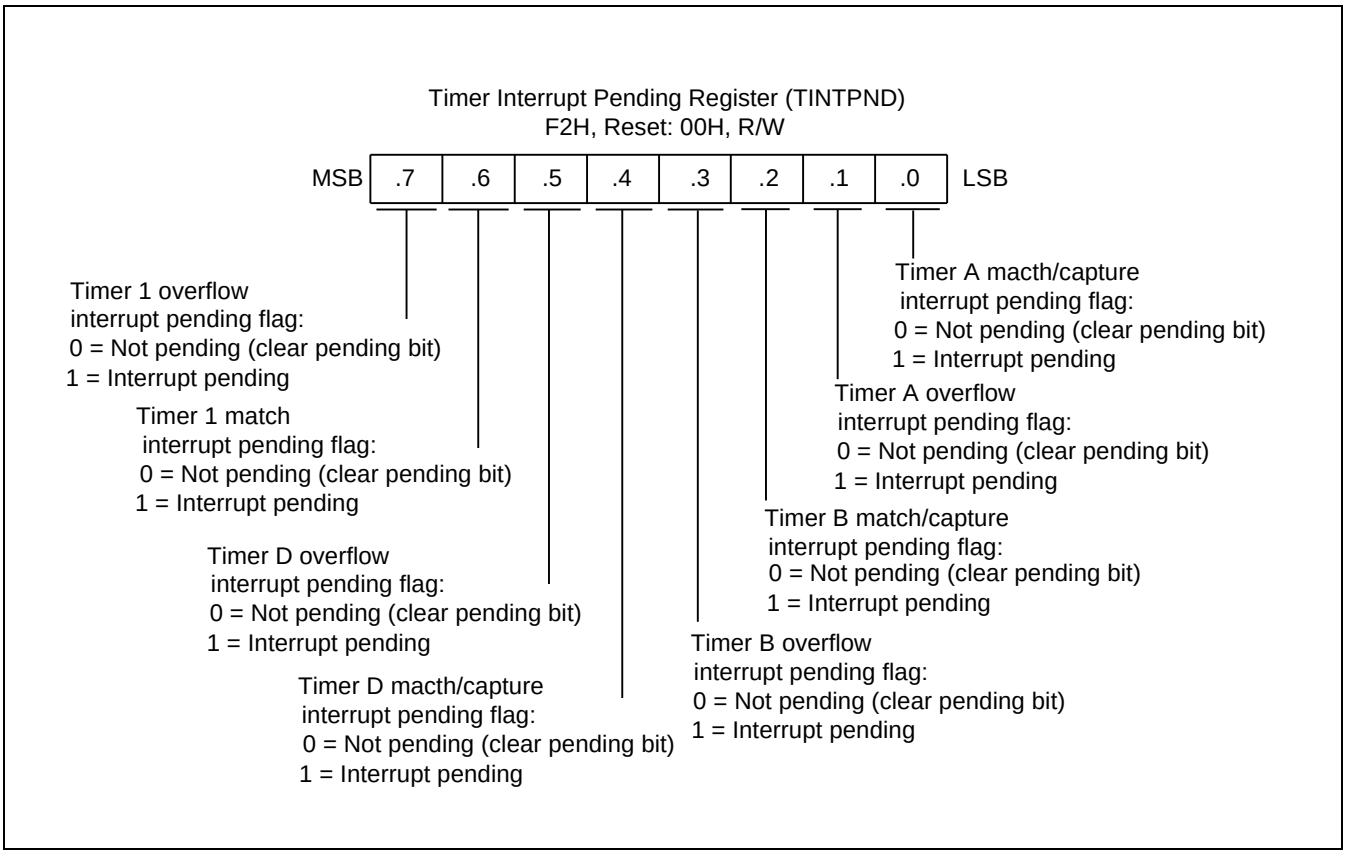
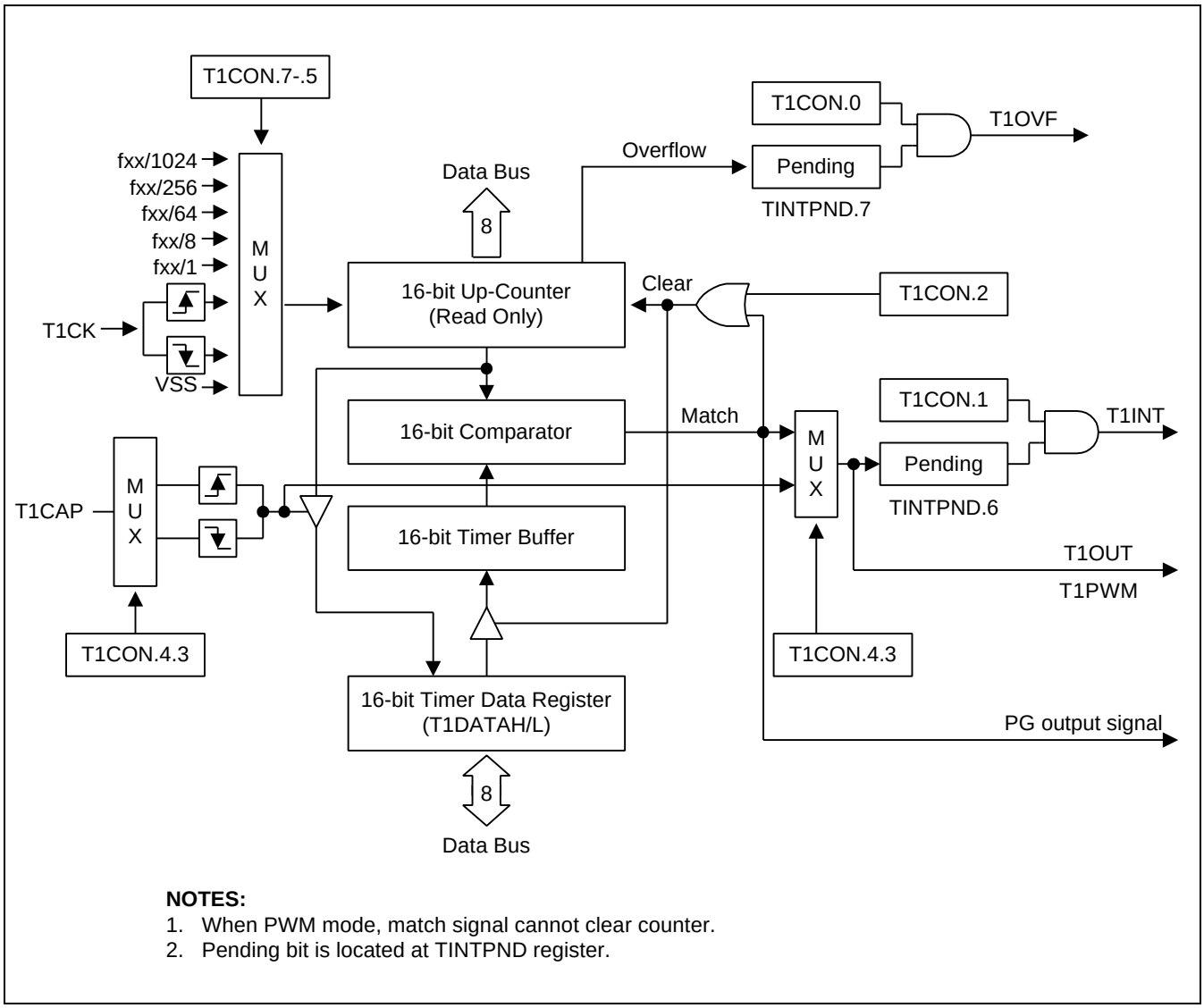


Figure 12-2. Timer A/B/D and TIMER 1 Pending Register (TINTPND)

BLOCK DIAGRAM



NOTES:
1. When PWM mode, match signal cannot clear counter.
2. Pending bit is located at TINTPND register.

Figure 12-3. TIMER 1 Functional Block Diagram

☞ PROGRAMMING TIP — Using the Timer 1

```
                ORG         0000h

                VECTOR      00h,INT_9498

                ORG         0100h

INITIAL:
                LD          SYM,#00h            ;  Disable Global/Fast interrupt
                LD          SP,#0C0H            ;  Set stack area
                LD          BTCON,#10100011b    ;  Disable Watch-dog

                LD          T1DATAH,#00H
                LD          T1DATAL,#0F0H

                LD          T1CON,#01001110b    ;  fxx/256, interval, clear counter, Enable interrupt
                                                ;  Duration 7.68ms (8 MHz x'tal)

                            EI

MAIN:
                •
                •
                •
                MAIN ROUTINE
                •
                •
                •

                JR          T,MAIN

INT_9498:
                •
                •
                •
                Interrupt service routine
                •
                •
                •
                IRET

                .END
```

13 TIMER 0

ONE 16-BIT TIMER MODE (TIMER 0)

The 16-bit timer 0 is used in one 16-bit timer or two 8-bit timers mode. If TCCON.7 is set to "1", Timer 0 is used as a 16-bit timer. If TCCON.7 is set to "0", timer 0 is used as two 8-bit timers.

— One 16-bit timer mode (Timer 0)

— Two 8-bit timers mode (Timer C and D)

OVERVIEW

The 16-bit timer 0 is an 16-bit general-purpose timer. Timer 0 has the interval timer mode by using the appropriate TCCON setting.

Timer 0 has the following functional components:

— Clock frequency divider (fxx divided by 1024, 512, 8, or 1) with multiplexer

— 16-bit comparator, and 16-bit reference data register (TCDATA, TDDATA)

— Timer 0 match interrupt generation

— Timer 0 control register, TCCON (D0H, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

The timer 0 module can generate an interrupt: the timer 0 match interrupt (T0INT). The T0INT pending condition should be cleared by software when it has been serviced. Even though T0INT is disabled, the application's service routine can detect a pending condition of T0INT by the software and execute it's sub-routine. When this case is used, the T0INT pending bit must be cleared by the application sub-routine by writing a "0" to the TCCON.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 0 reference data registers, TCDATA and TDDATA. The match signal generates a timer 0 match interrupt and clears the counter.

If, for example, you write the value 32H and 10H to TCDATA and TDDATA, respectively, and 8EH to TCCON, the counter will increment until it reaches 3210H. At this point, the timer 0 interrupt request is generated, the counter value is reset, and counting resumes.

Timer 0 Control Register (TCCON)

You use the timer 0 control register, TCCON, to

— Enable the timer 0 operating (interval timer)
— Select the timer 0 input clock frequency
— Clear the timer 0 counter
— Enable the timer 0 interrupt
— Clear timer 0 interrupt pending conditions

TCCON is located at address D0H, and is read/write addressable using register addressing mode.

A reset clears TCCON to "00H". This sets timer 0 to disable interval timer mode, selects an input clock frequency of fxx/1024, and disables timer 0 interrupt. You can clear the timer 0 counter at any time during normal operation by writing a "1" to TCCON.3.

To enable the timer 0 interrupt , you must write TCCON.7, TCCON.2, and TCCON.1 to "1".
To generate the exact time interval, you should write TCCON.3 and TCCON.0, which cleared counter and interrupt pending bit. To detect an interrupt pending condition when T0INT is disabled, the application program polls pending bit, TCCON.0. When a "1" is detected, a timer 0 interrupt is pending. When the T0INT sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 0 interrupt pending bit, TCCON.0.

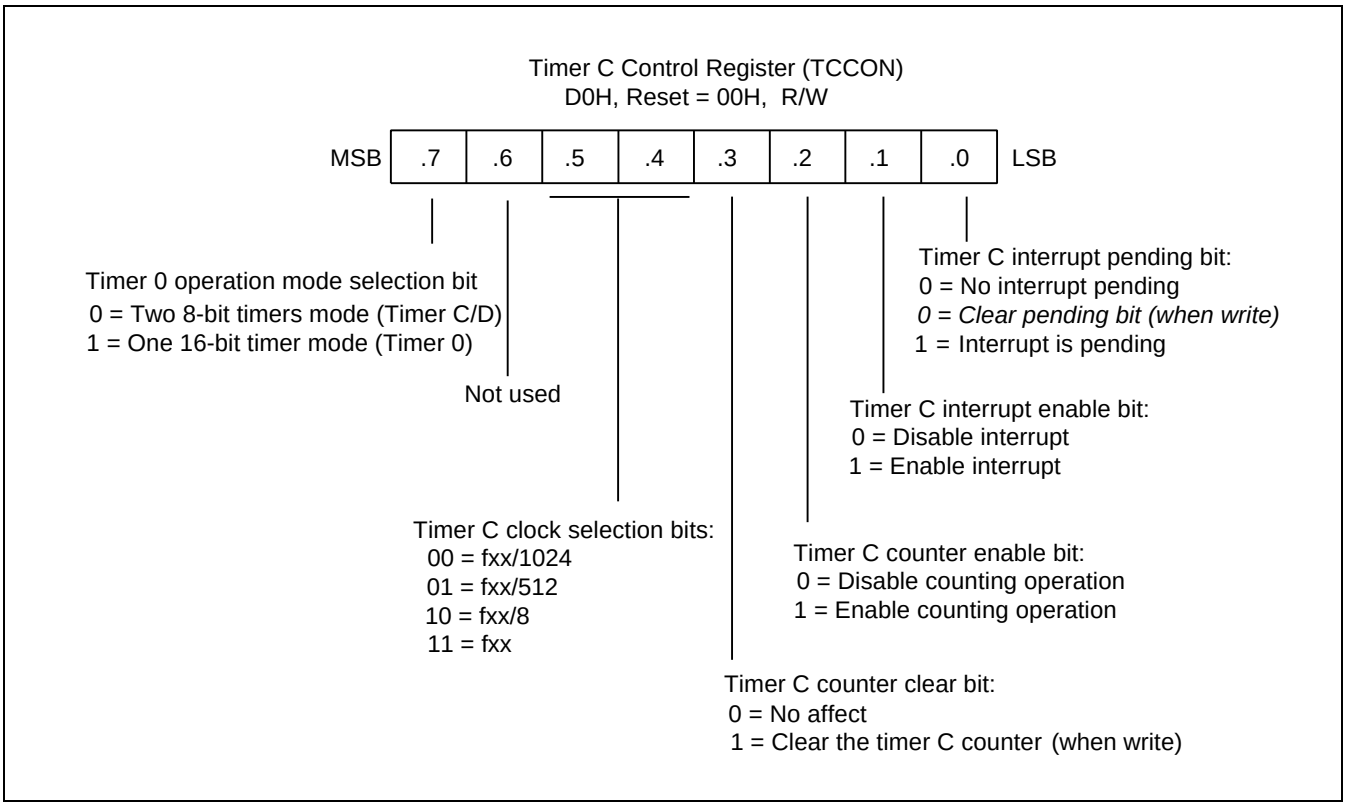
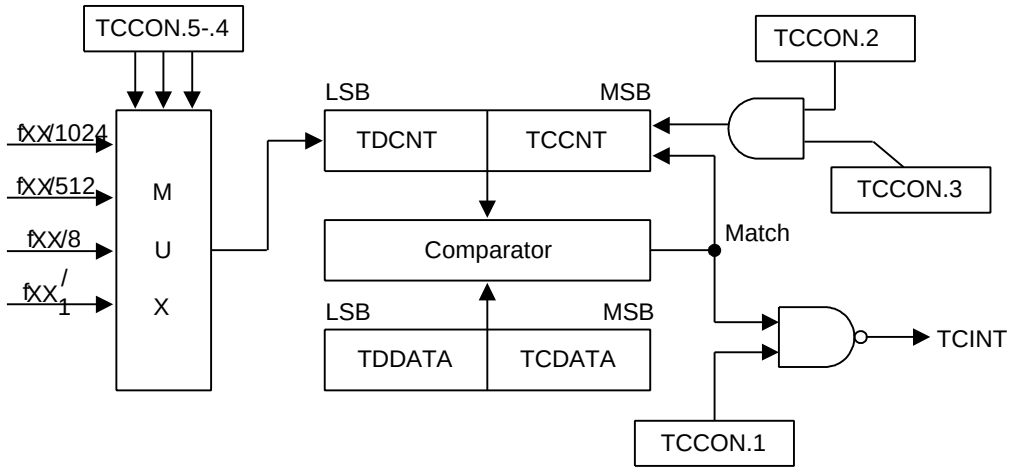


Figure 13-1. Timer 0 Control Register (TCCON)

BLOCK DIAGRAM



NOTE: When TCCON.7 is '1', one 16-bit Timer 0.

Figure 13-2. Timer 0 Functional Block Diagram

TWO 8-BIT TIMERS MODE (TIMER C and D)

OVERVIEW

The 8-bit Timer C and D are the 8-bit general-purpose timers. Timer C have the interval timer mode, and the Timer D have the interval timer mode and PWM mode by using the appropriate TCCON and TDCON setting, respectively.

Timer C and D have the following functional components:

— Clock frequency divider with multiplexer
  – fxx divided by 1024, 512, 8 and 1 for Timer C
  – fxx divided by 8, 4, 2, or 1 for Timer D
— 8-bit counter (TCCNT, TDCNT), 8-bit comparator, and 8-bit reference data register (TCDATA, TDDATA))
— Timer C match interrupt generation
— Timer C control register, TCCON (D0H, read/write)
— Timer D have I/O pin for match and PWM output (P1.6, TDOUT)
— Timer D overflow interrupt generation
— Timer D match interrupt generation
— Timer D control register, TDCON (D1H, read/write)

Timer C and D Control Register (TCCON, TDCON)

You can use the Timer C and D control register, TCCON and TDCON to

— Enable the Timer C (interval timer mode) and D operating (interval timer mode and PWM mode)
— Select the Timer C and D input clock frequency
— Clear the Timer C and D counter, TCCNT and TDCNT
— Enable the Timer C and D interrupt
— Clear Timer C and D interrupt pending conditions

TCCON and TDCON are located in address D0H and D1H, and is read/write addressable using register addressing mode.

A reset clears TCCON to "00H". This sets Timer C to disable interval timer mode, selects an input clock frequency of fxx/1024, and disables Timer C interrupt. You can clear the Timer C counter at any time during normal operation by writing a "1" to TCCON.3.

A reset clears TDCON to "00H". This sets Timer D to enable interval timer mode and disable PWM mode, selects an input clock frequency of fxx/8, and disables Timer C interrupt. You can clear the Timer D counter at any time during normal operation by writing a "1" to TDCON.3.

To enable the Timer C interrupt (TCINT) and Timer D interrupt (TDINT) you must write TCCON.7 to "0", TCCON.2 (TDCON.2) and TCCON.1 (TDCON.1) to "1". To generate the exact time interval, you should write TCCON.3 (TDCON.3) and TCCON.0 (TINTPND.4), which cleared counter and interrupt pending bit. To detect an interrupt pending condition when TCINT and TDINT is disabled, the application program polls pending bit, TCCON.0 and TINTPND.4. When a "1" is detected, a Timer C interrupt (TCINT) and Timer D interrupt (TDINT) is pending. When the TCINT and TDINT sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the Timer C and D interrupt pending bit, TCCON.0 and TINTPND.4.

Also, to enable Timer D overflow interrupt (TDOVF), you must write TCCON.7 to "0", TDCON.2 and TDCON.0 to "1". To generate the exact time interval, you should write TDCON.3 and TINTPND.5, witch cleared counter and interrupt pending bit.

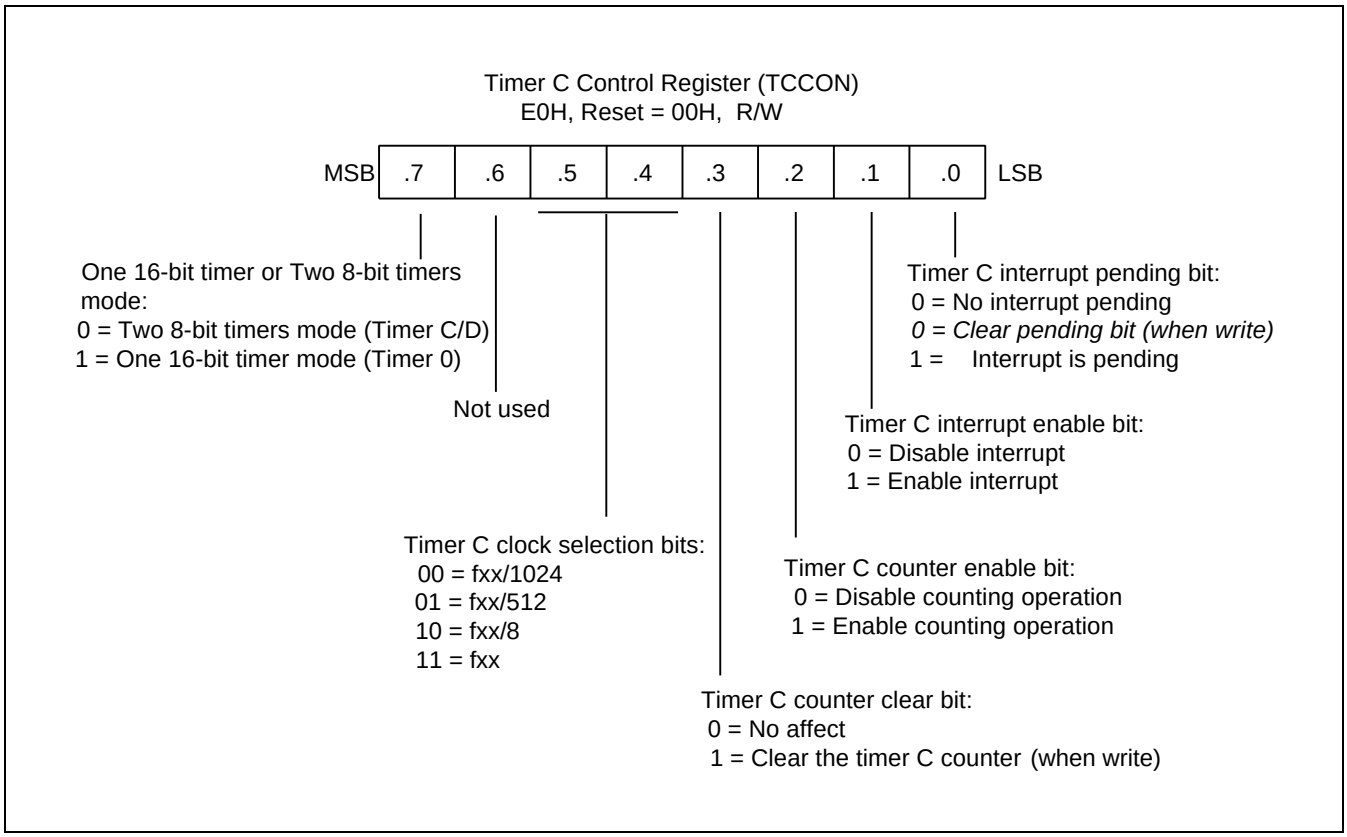


Figure 13-3. Timer C Control Register (TCCON)

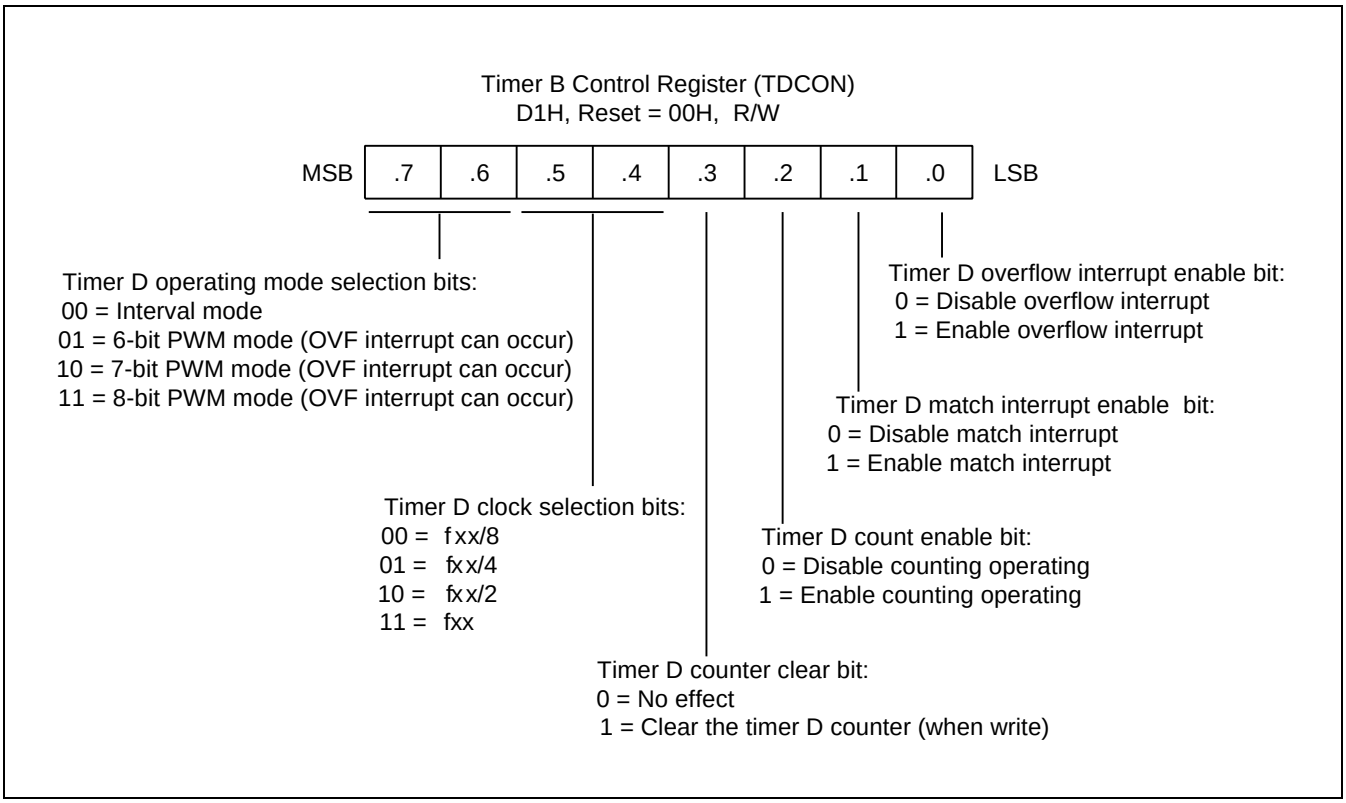


Figure 13-4. Timer D Control Register (TDCON)

FUNCTION DESCRIPTION

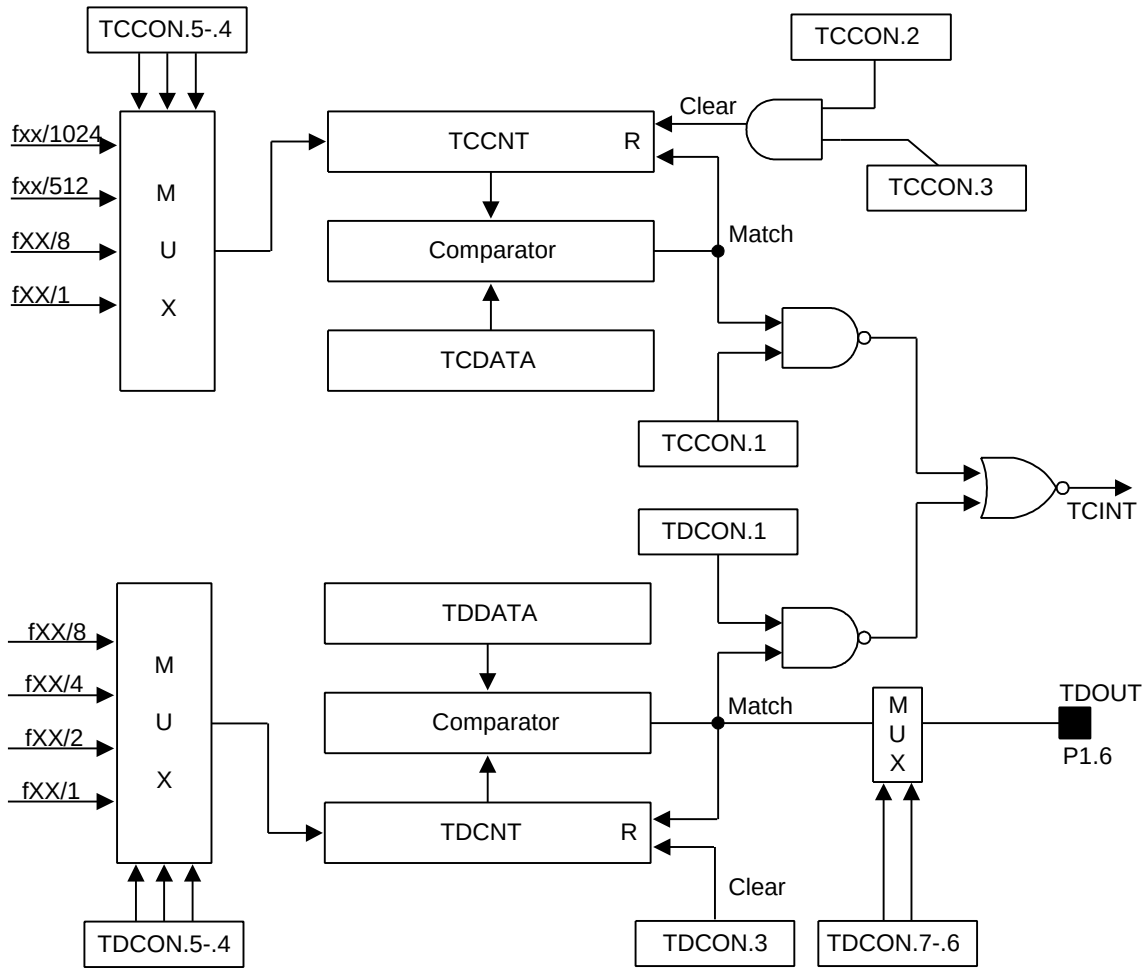
Interval Timer Function (Timer C and Timer D)

The Timer C and D module can generate an interrupt: the Timer C match interrupt (TCINT) and the Timer D match interrupt (TDINT). The Timer C match interrupt pending condition (TCCON.0) and the Timer D match interrupt pending condition (TINTPND.4) must be cleared by software in the application's interrupt service by means of writing a "0" to the TCCON.0 and TINTPND.4 interrupt pending bit.

Even though TCINT and TDINT are disabled, the application's service routine can detect a pending condition of TCINT and TDINT by the software and execute it's sub-routine. When this case is used, the TCINT and TDINT pending bit must be cleared by the application sub-routine by writing a "0" to the corresponding pending bit TCCON.0 and TINTPND.6.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the Timer C or Timer D reference data registers, TCDATA or TDDATA. The match signal generates corresponding match interrupt (TCINT, TDINT) and clears the counter.

If, for example, you write the value 20H to TCDATA and 0EH to TCCON, the counter will increment until it reaches 20H. At this point, the Timer C interrupt request is generated, the counter value is cleared, and counting resumes and you write the value 10H to TDDATA, "0" to TCCON.6, and 0EH to TDCON, the counter will increment until it reaches 10H. At this point, TB interrupt request is generated, the counter value is cleared and counting resumes.



NOTE: When TCCON.7 is '0', two 8-bit timer C/D (Interval mode).

Figure 13-5. Timer C and B Function Block Diagram

Pulse Width Modulation Mode (Timer D)

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the TDOUT (P1.6) pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the Timer D 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 D overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TDOUT 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 13-6).

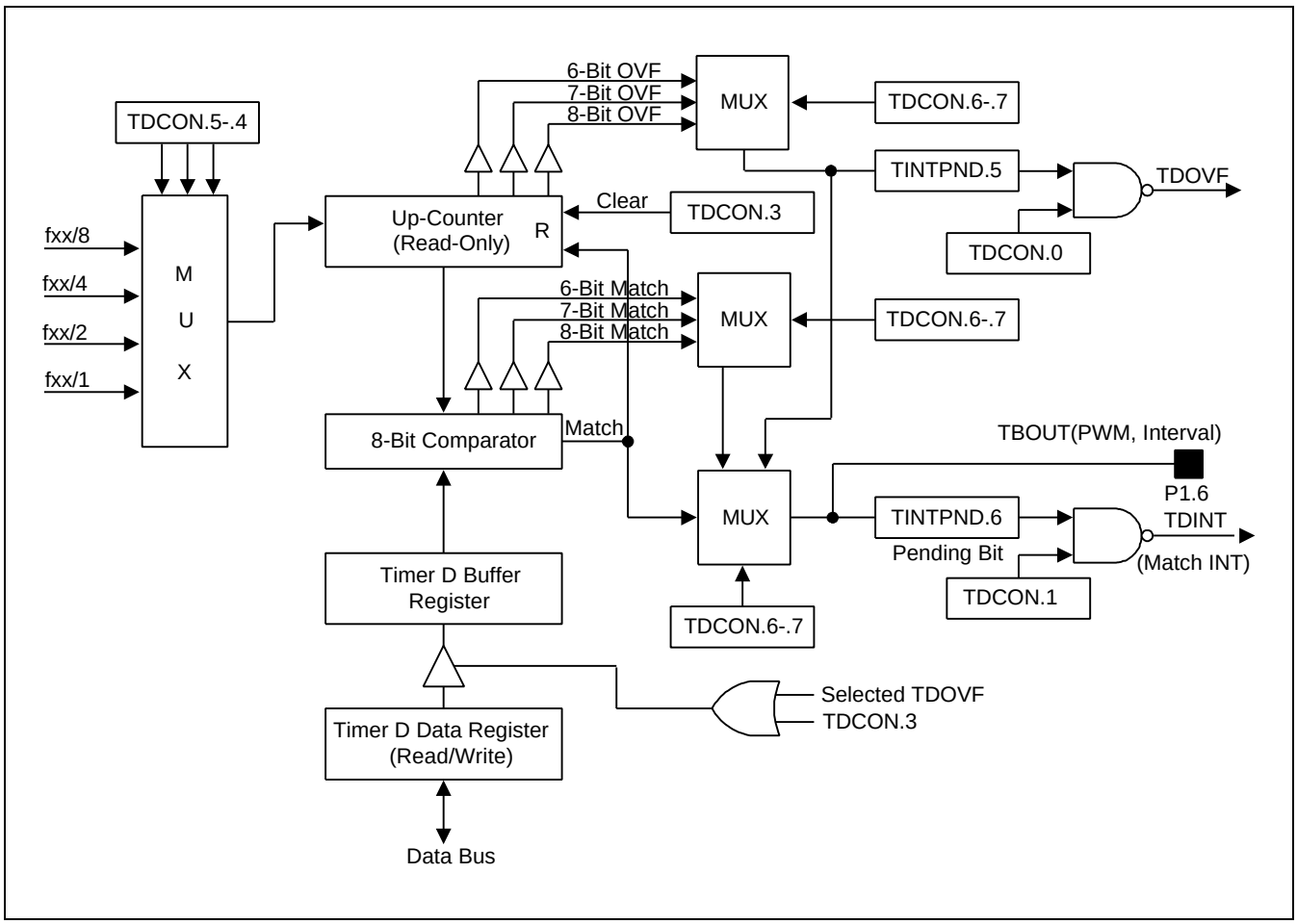


Figure 13- 6. Timer D PWM Function Block Diagram

NOTES

14 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:

— Shift Register I/O with baud rate of fxx/(16 × (8bit BRDATA+1))

— 8-bit UART mode; variable baud rate, fxx/(16 × (8bit BRDATA+1))

— 9-bit UART mode; fxx/16

— 9-bit UART mode; variable baud rate, fxx/(16 × (8bit BRDATA+1))

UART receive and transmit buffers are both accessed via the data register, UDATA, is at address FFH. 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, the first data byte will be lost (Overrun error).

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 (UARTPND.1) is "0" and the receive enable bit (UARTCON.4) is "1". In mode 1 and 2, 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:

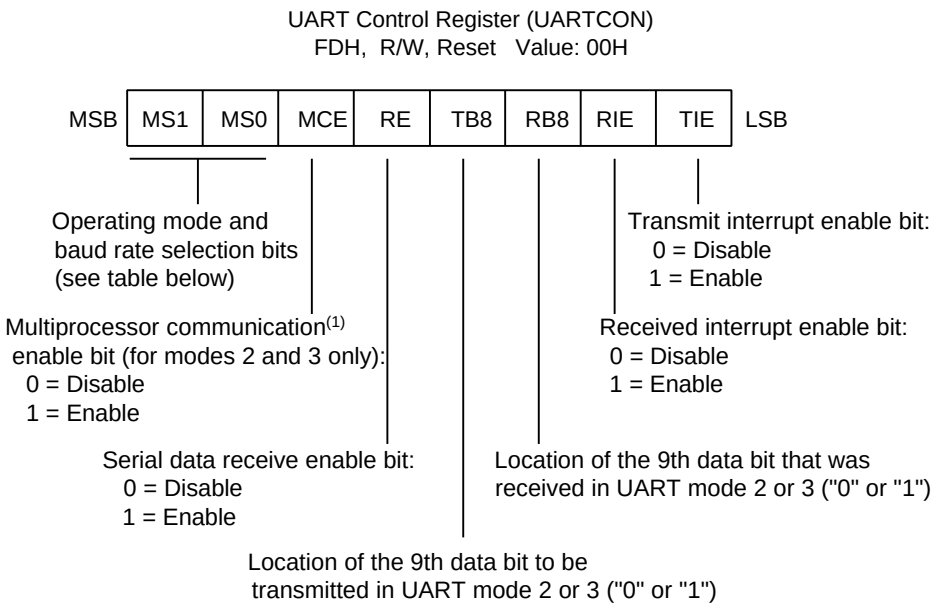
1. Configure P0.0 and P0.1 to alternative function (RXD (P0.0), TXD (P0.1)) for UART module by setting the P0CON register to appropriatly value.
2. Load an 8-bit value to the UARTCON control register to properly configure the UART I/O module.
3. For interrupt generation, set the UART interrupt enable bit (UARTCON.1 or UARTCON.0) to "1".
4. When you transmit data to the UART buffer, write transmit data to UDATA, the shift operation starts.
5. When the shift operation (transmit/receive) is completed, UART pending bit (UARTPND.1 or UARTPND.0) is set to "1" and an UART interrupt request is generated.

UART CONTROL REGISTER (UARTCON)

The control register for the UART is called UARTCON at address FDH. 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 (mode 2)
— 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.

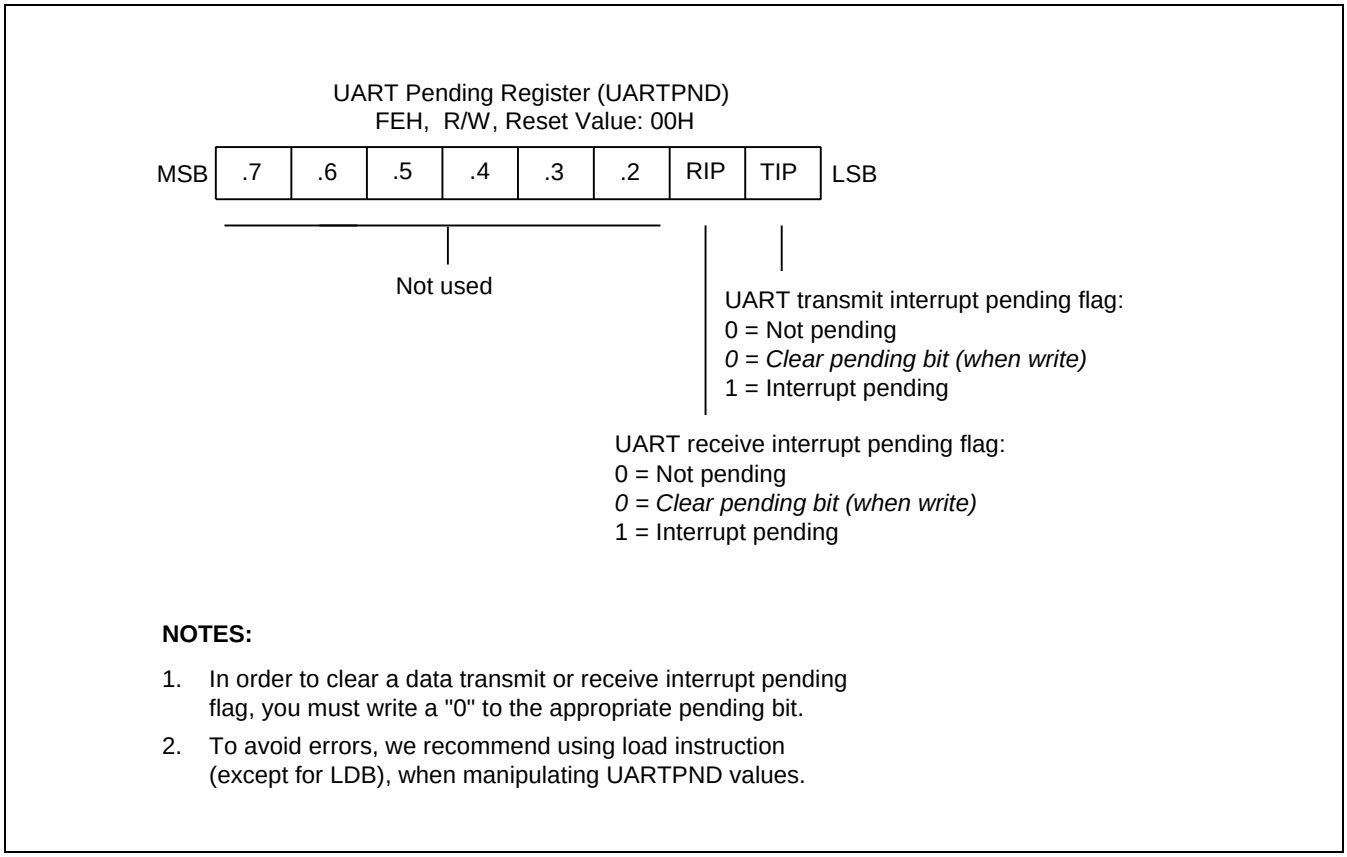
Figure 14-1. UART Control Register (UARTCON)

UART INTERRUPT PENDING REGISTER (UARTPND)

The UART interrupt pending register, UARTPND is located at address FEH. It contains the UART data transmit interrupt pending bit (UARTPND.0) and the receive interrupt pending bit (UARTPND.1).

In mode 0 of the UART module, the receive interrupt pending flag UARTPND.1 is set to "1" when the 8th receive data bit has been shifted. In mode 1 or 2, the UARTPND.1 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 UARTPND.1 flag must be cleared by software in the interrupt service routine.

In mode 0 of the UART module, the transmit interrupt pending flag UARTPND.0 is set to "1" when the 8th transmit data bit has been shifted. In mode 1 or 2, the UARTPND.0 bit is set at the start of the stop bit. When the CPU has acknowledged the transmit interrupt pending condition, the UARTPND.0 flag must be cleared by software in the interrupt service routine.



NOTES:

1. In order to clear a data transmit or receive interrupt pending flag, you must write a "0" to the appropriate pending bit.
2. To avoid errors, we recommend using load instruction (except for LDB), when manipulating UARTPND values.

Figure 14-2. UART Interrupt Pending Register (UARTPND)

UART DATA REGISTER (UDATA)

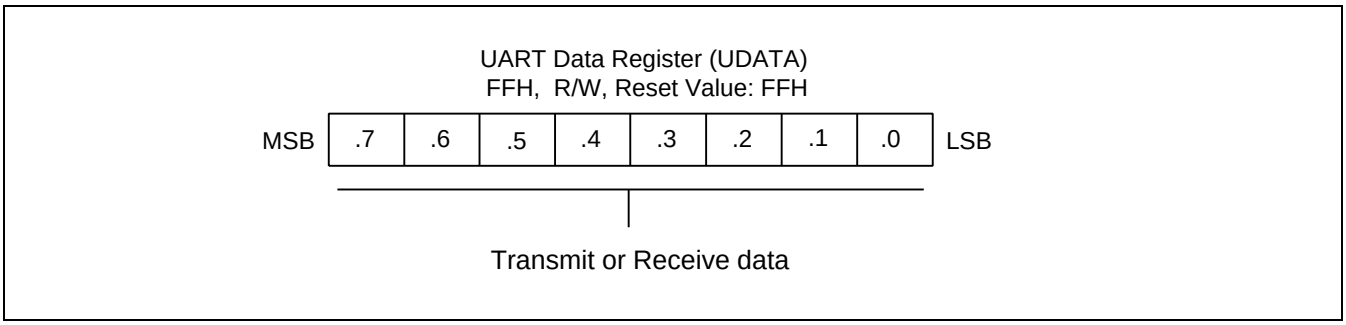


Figure 14-3. UART Data Register (UDATA)

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).

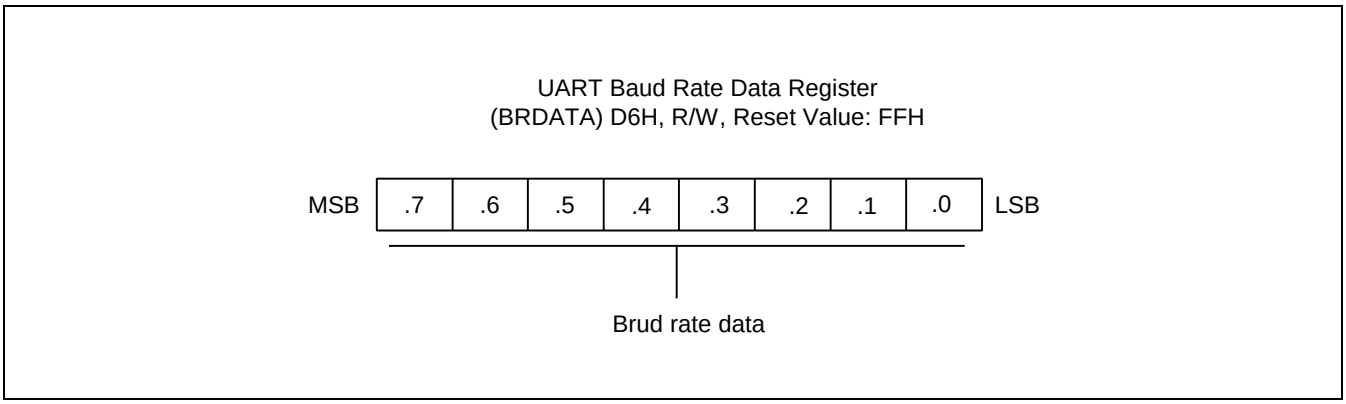


Figure 14-4. UART Baud Rate Data Register (BRDATA)

BAUD RATE CALCULATIONS

The baud rate is determined by the baud rate data register, 8bit BRDATA

Mode 0 baud rate = fxx/(16 × (8Bit BRDATA + 1))
Mode 1 baud rate = fxx/(16 × (8Bit BRDATA + 1))
Mode 2 baud rate = fxx/16
Mode 3 baud rate = fxx/(16 × (8Bit BRDATA + 1))

Table 14-1. Commonly Used Baud Rates Generated by 8-bit BRDATA

Mode	Baud Rate	Oscillation Clock	BRDATA	
			Decimal	Hex
Mode 2	0.5 MHz	8 MHz	x	x
Mode 0	62,500 Hz	10 MHz	09	09H
Mode 1	9,615 Hz	10 MHz	64	40H
Mode 3	38,461 Hz	8 MHz	12	0CH
	12,500 Hz	8 MHz	39	27H
	19,230 Hz	4 MHz	12	0CH
	9,615 Hz	4 MHz	25	19H

BLOCK DIAGRAM

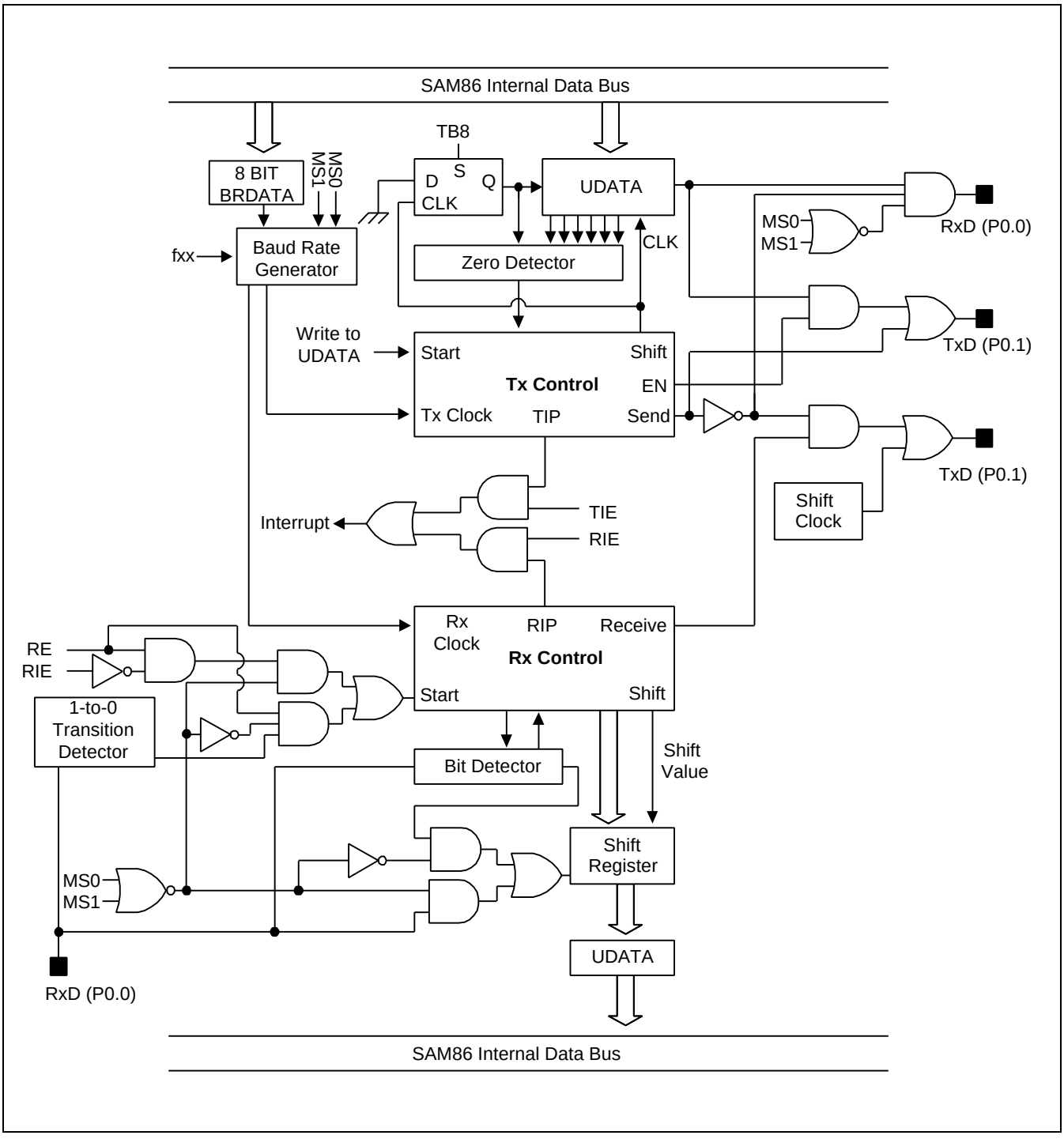


Figure 14-5. UART Functional Block Diagram

UART MODE 0 FUNCTION DESCRIPTION

In mode 0, UART is input and output through the RxD (P0.0) pin and TxD (P0.1) 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 (FFH) to start the transmission operation.

Mode 0 Receive Procedure

1. Select mode 0 by setting UATCON.6 and .7 to "00B".
2. Clear the receive interrupt pending bit (UARTPND.1) by writing a "0" to UARTPND.1.
3. Set the UART receive enable bit (UARTCON.4) to "1".
4. The shift clock will now be output to the TxD (P0.1) pin and will read the data at the RxD (P0.0) pin. A UART receive interrupt (vector 00H-01H) occurs when UARTCON.1 is set to "1".

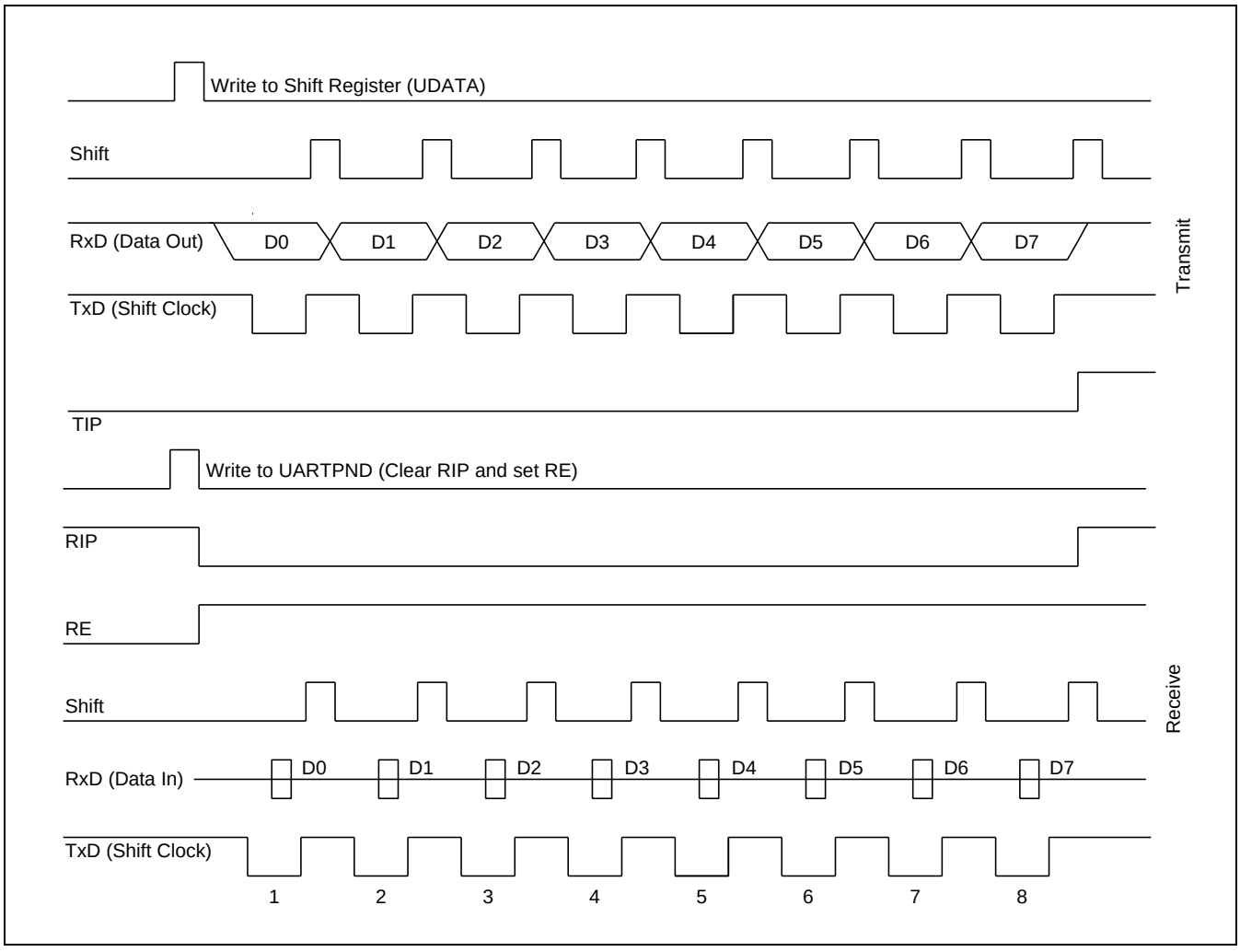


Figure 14-6. Timing Diagram for UART Mode 0 Operation

UART MODE 1 FUNCTION DESCRIPTION

In mode 1, 10-bits are transmitted (through the TxD (P0.1) pin) or received (through the RxD (P0.0) 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 8bit 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 (FFH). The start and stop bits are generated automatically by hardware.

Mode 1 Receive Procedure

1. Select the baud rate to be generated by 8bit 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 (P0.0) pin will cause the UART module to start the serial data receive operation.

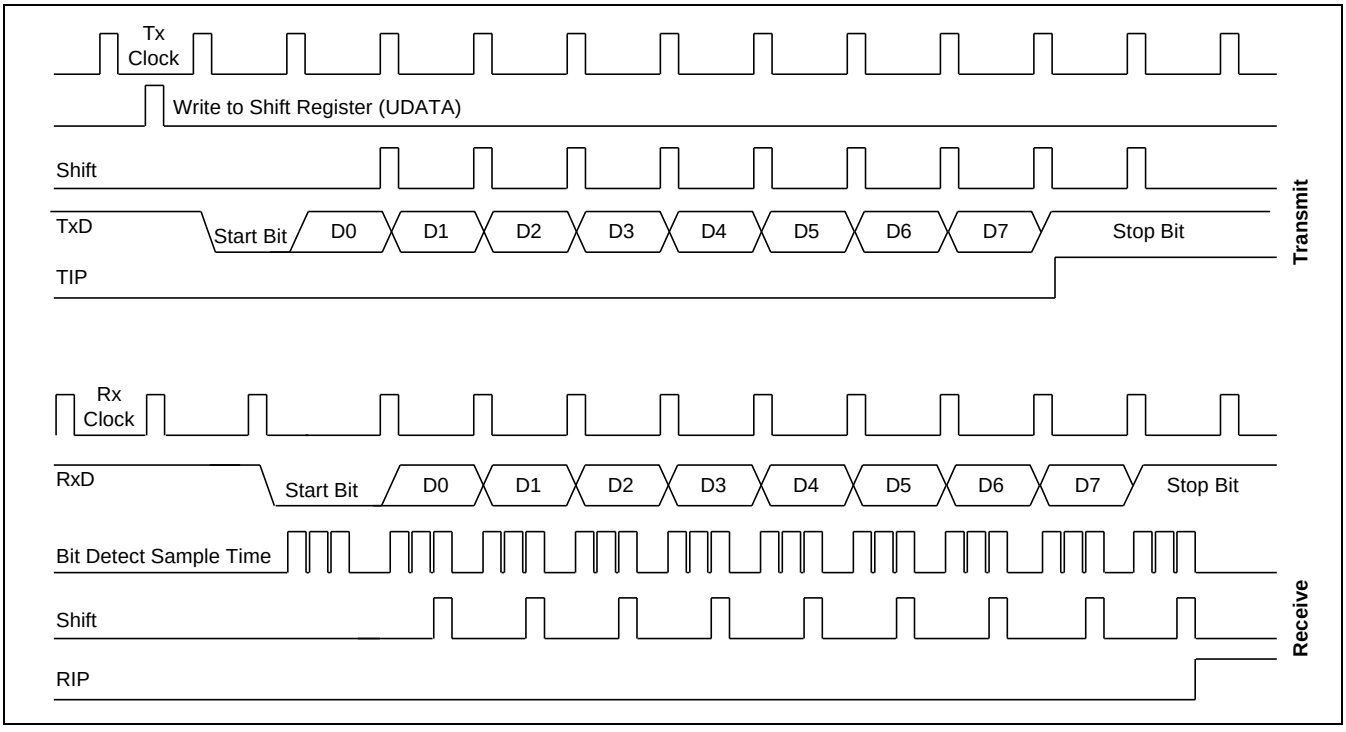


Figure 14-7. Timing Diagram for UART Mode 1 Operation

UART MODE 2 FUNCTION DESCRIPTION

In mode 2, 10-bits are transmitted through the TxD pin or received through the RxD pin. Each data frame has three 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 (UARTCON0.3). When receiving, the 9th data bit that is received is written to the RB8 bit (UARTCON0.2), 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 UART0) 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 (FFH), 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.

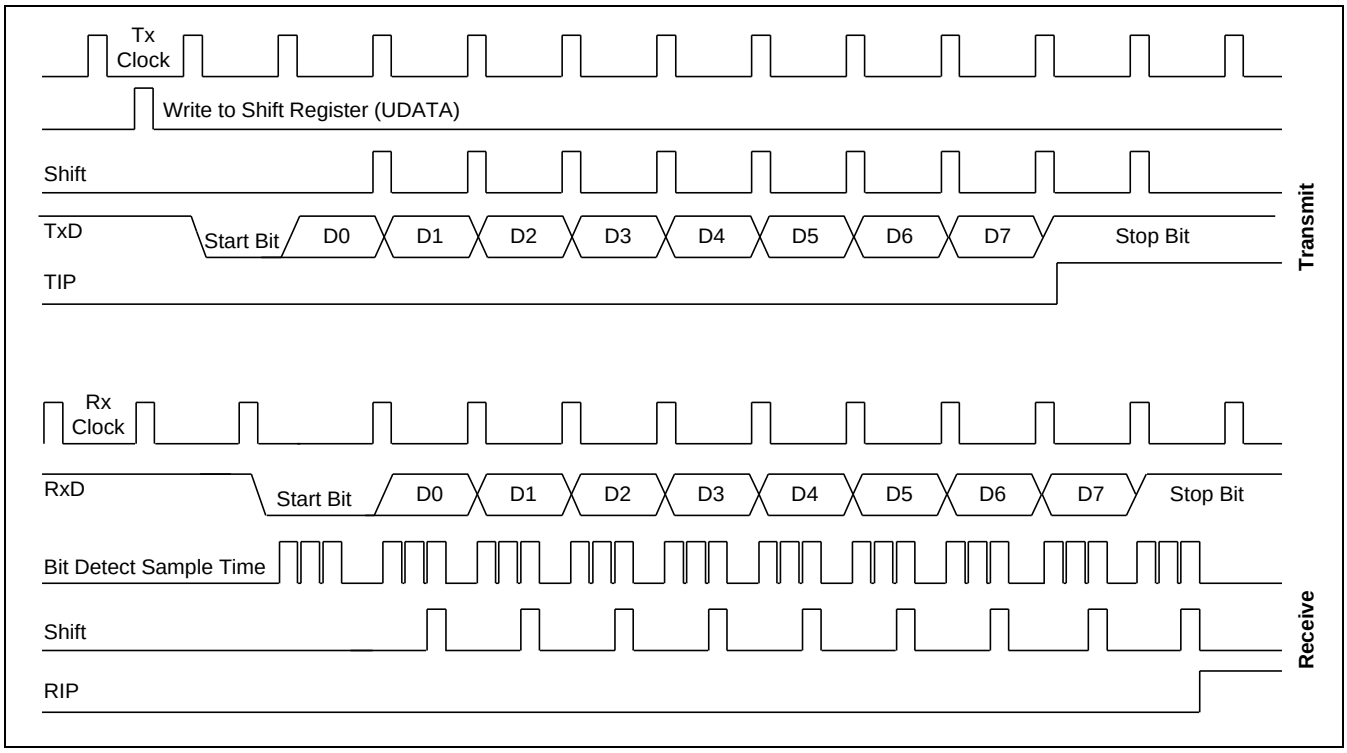


Figure 14-8. Timing Diagram for UART Mode 2 Operation

UART Mode 3 Function Description

In mode 3, 11-bits are transmitted (through the TxD) or received (through the RxD). 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 setting BRDATA.
2. Select mode 3 (9-bit UART) by setting UARTCON bits 6 and 7 to '11B'. Also, select the 9th data bit to be transmitted by writing TB8 to "0" or "1"
3. Write transmission data to the shift register, UDATA (FFH), to start the transmit operation.

Mode 3 Receive Procedure

1. Select the baud rate to be generated by setting BRDATA.
2. Select mode 3 and set the receive enable bit (RE) in the UARTCON register to "1".
3. The receive operation starts when the signal at the RxD pin goes to low level.

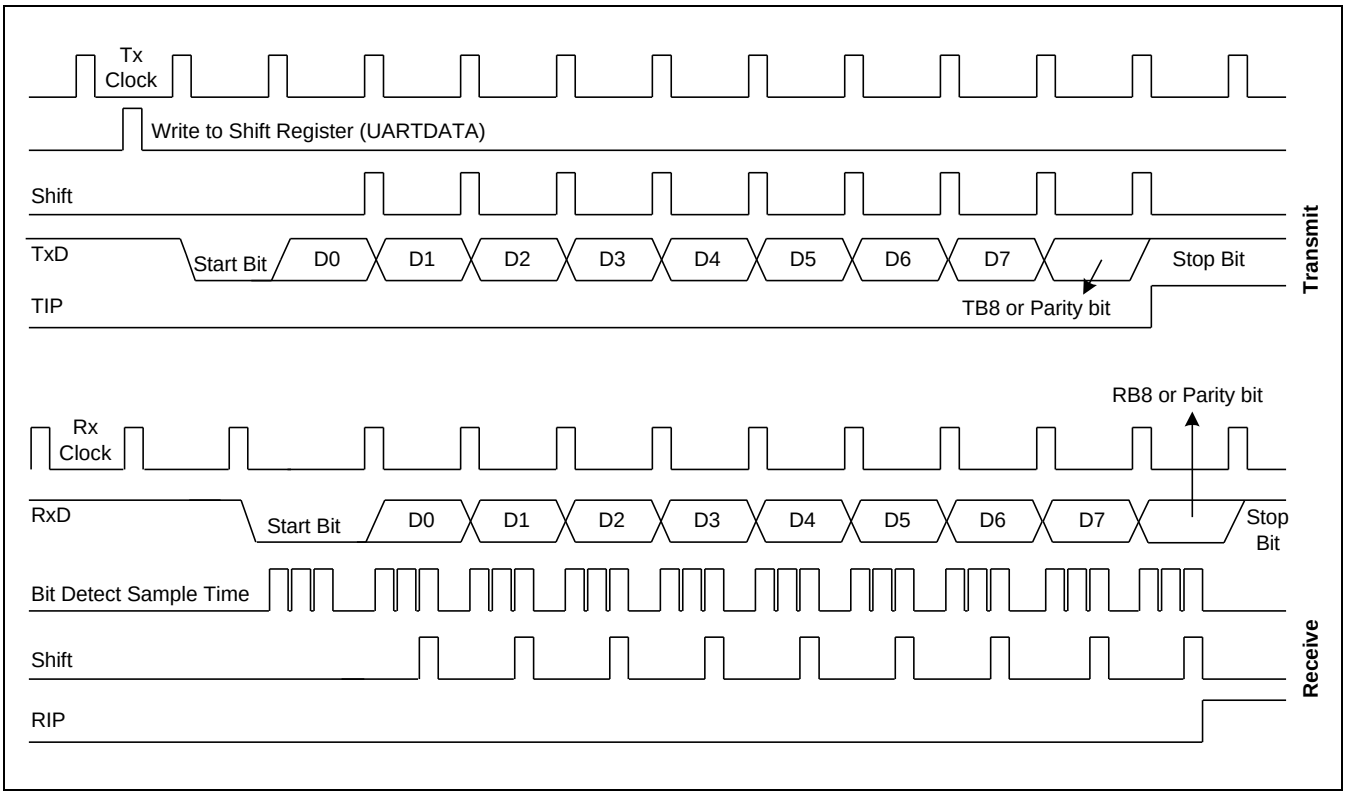


Figure 14-9. Timing Diagram for UART Mode 3 Operation

SERIAL COMMUNICATION FOR MULTIPROCESSOR CONFIGURATIONS

The S3C9-series multiprocessor communication features let a "master" S3C9498/F9498 send a multiple-frame serial message to a "slave" device in a multi- S3C9498/F9498 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 mode 2 or 3 with the parity disable mode. In mode 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 registers. 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 S3C9498/F9498 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.

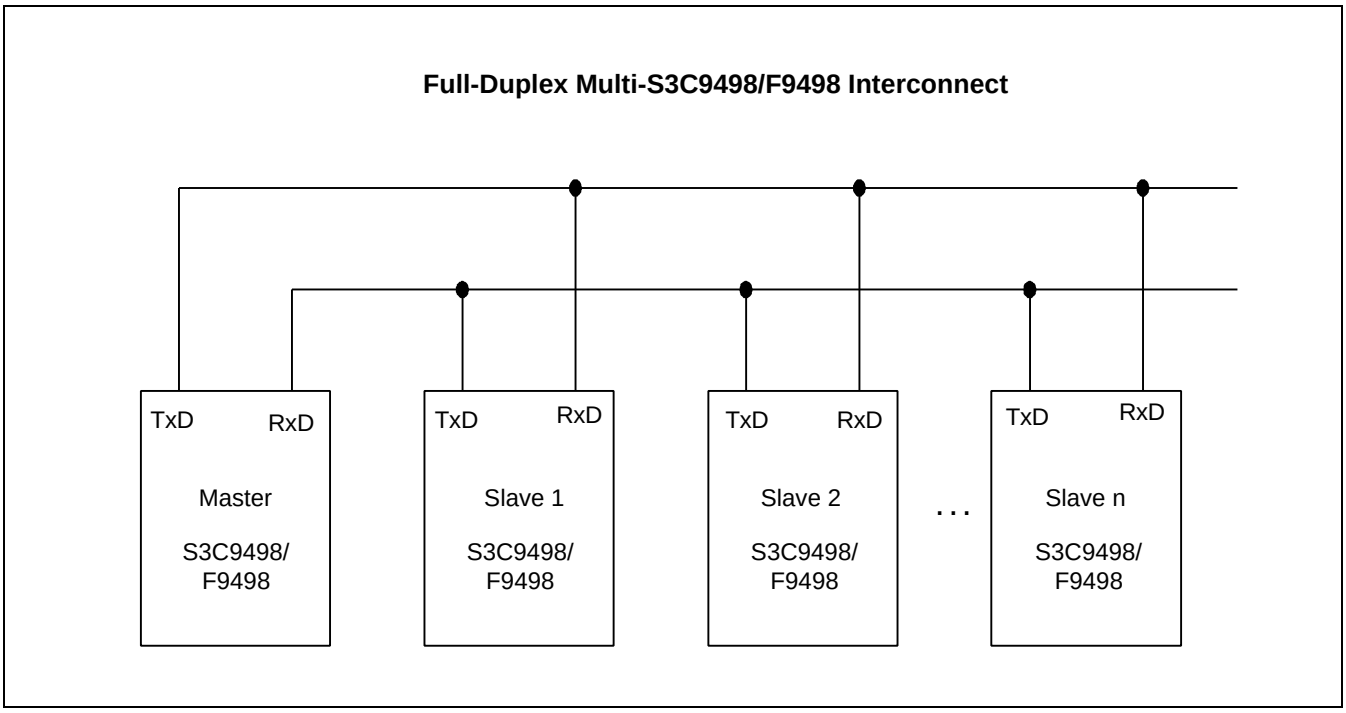


Figure 14-10. Connection Example for Multiprocessor Serial Data Communications

15 SERIAL I/O INTERFACE

OVERVIEW

Serial I/O module, SIO can interface with various types of external devices that require serial data transfer. The components of each SIO function block are:

— 8-bit control register (SIOCON)
— Clock selection logic
— 8-bit data buffer (SIODATA)
— 8-bit presale (SIOPS)
— 3-bit serial clock counter
— Serial data I/O pins (SI, SO)
— External clock input pin (SCK)

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 3 (SO, SCK, SI) by loading the appropriate value to the P3CON Register.
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.1) to "1".
4. When you the 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) is set to "1" and an SIO interrupt request is generated.

SERIAL I/O CONTROL REGISTERS (SIOCON)

The control registers for serial I/O interface, SIOCON, is located at F1H. It has the control settings 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.

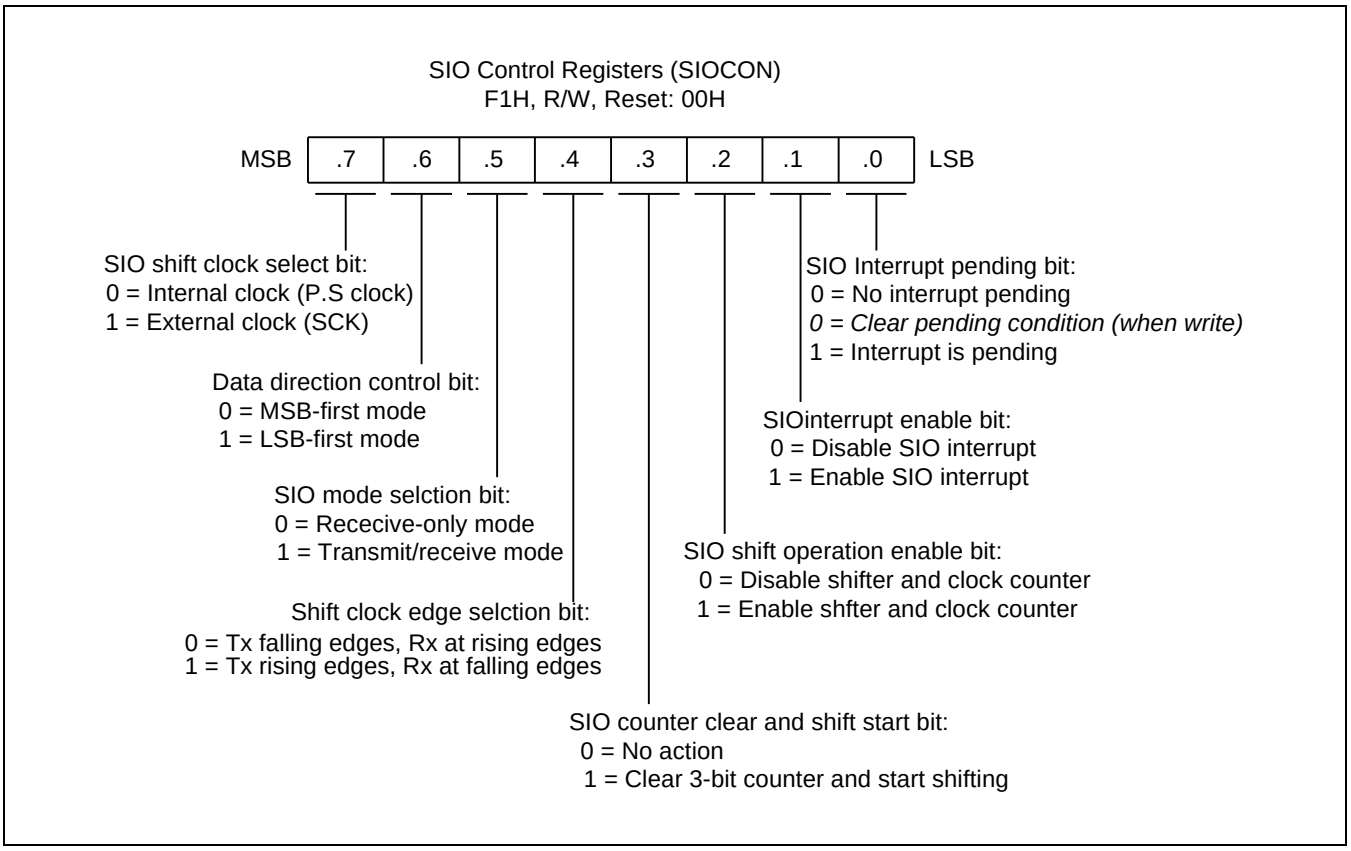


Figure 15-1. Serial I/O Interface Control Register (SIOCON)

SIO PRESCALER REGISTER (SIOPS)

The control register for serial I/O interface module, SIOPS is located at F4H.
The value stored in the SIO prescaler registers, SIOPS, lets you determine the SIO clock rate (baud rate) as follows:

Baud rate = Input clock(Xin/4) / (SIOP+ 1), or external SCK input clock

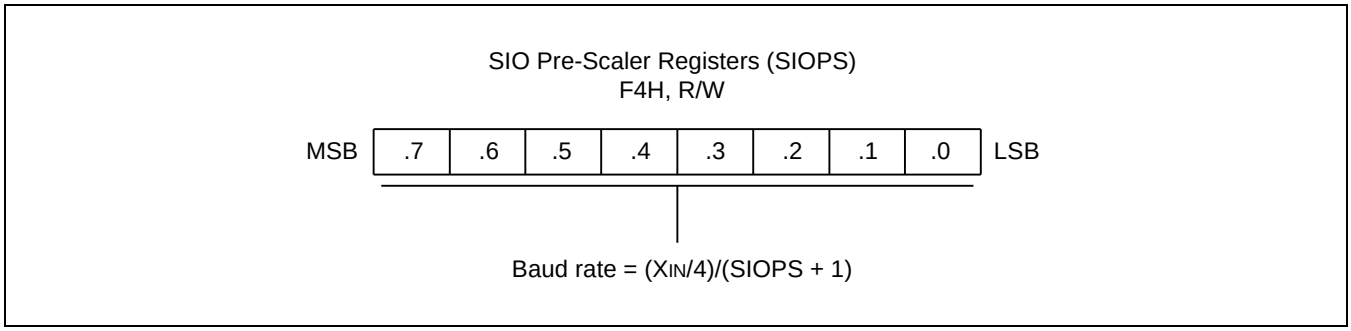


Figure 15-2. SIO Pre-Scaler Register (SIOPS)

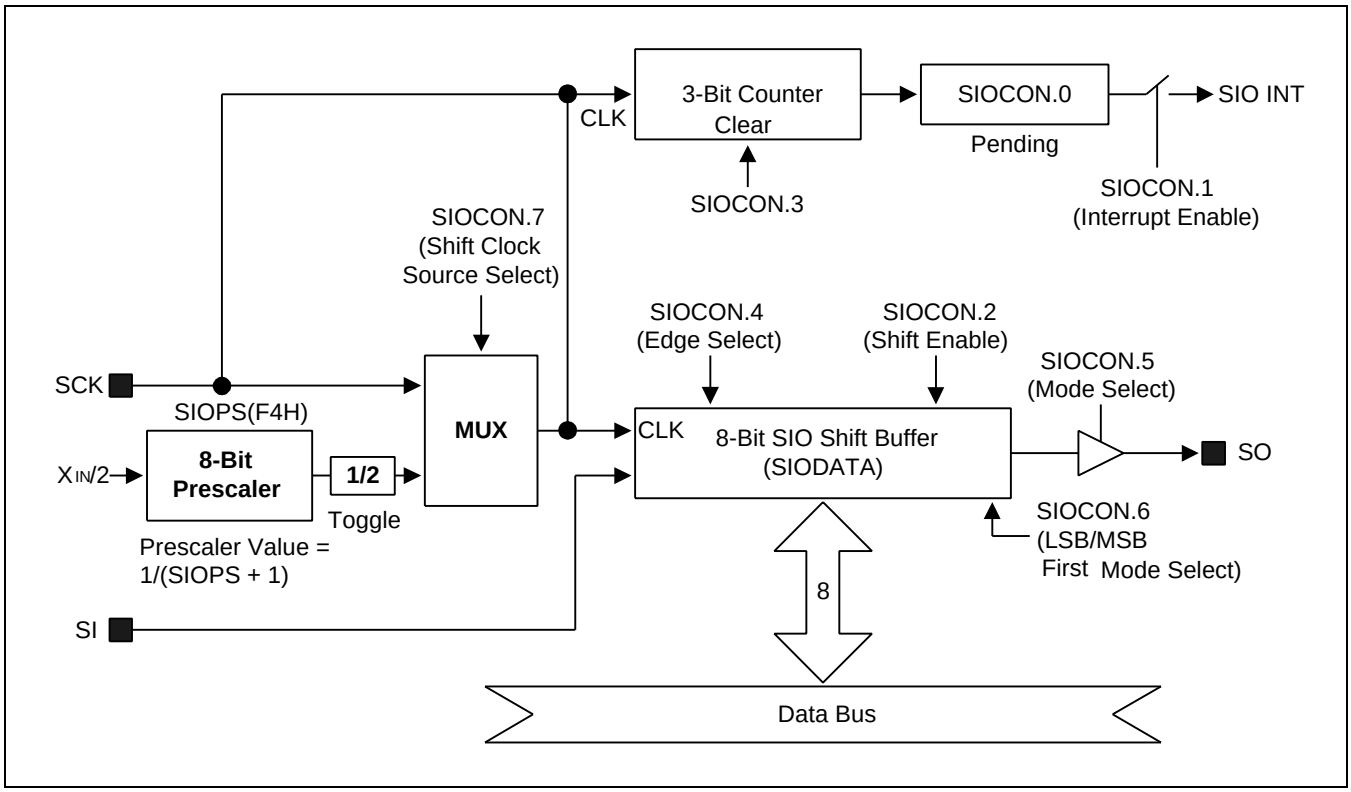


Figure 15-3. SIO Functional Block Diagram

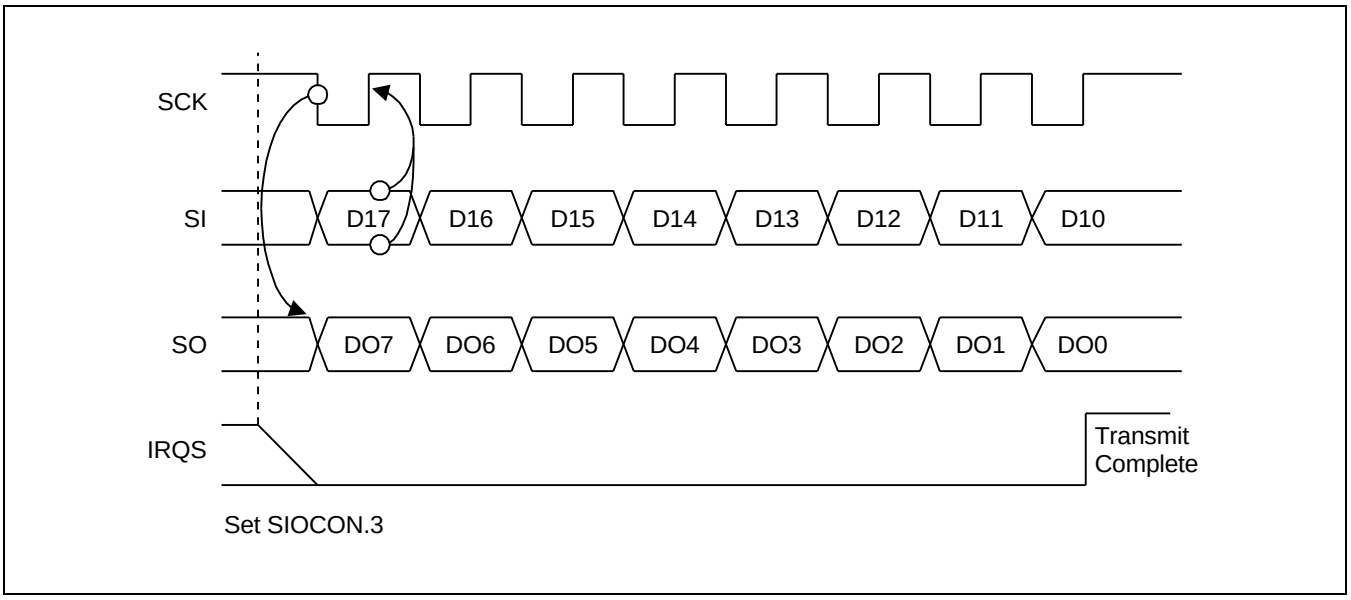


Figure 15-4. Serial I/O Timing in Transmit-Receive Mode (Tx at falling, SIOCON.4 = 0)

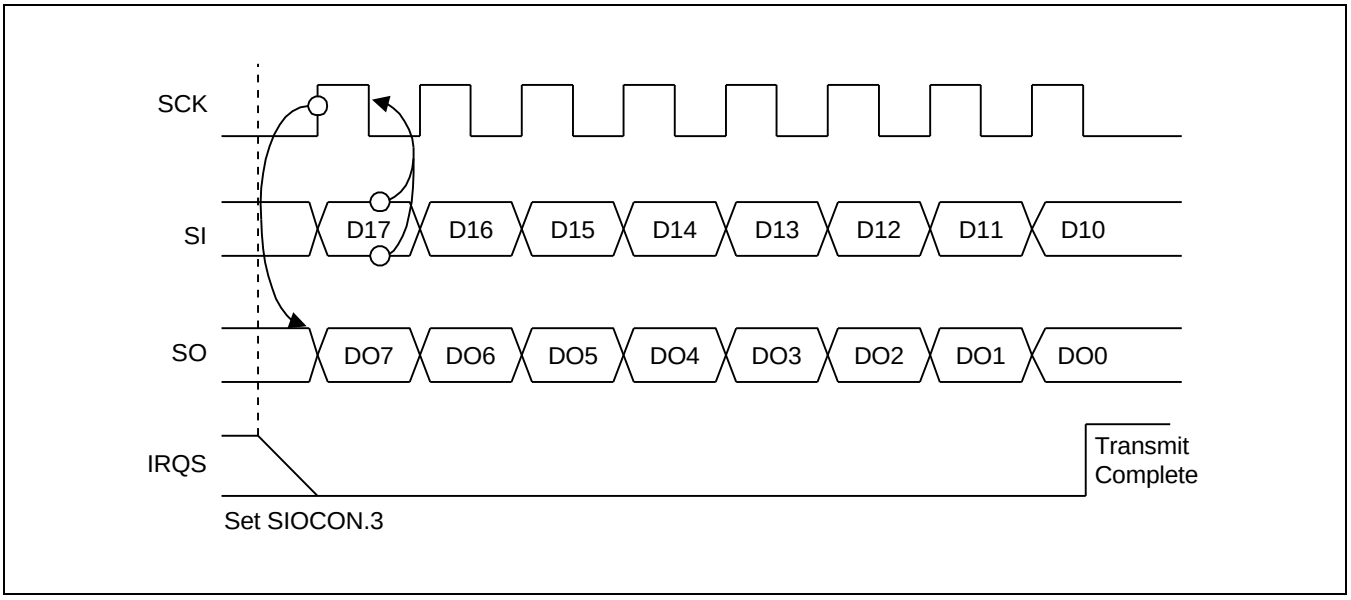


Figure 15-5. Serial I/O Timing in Transmit-Receive Mode (Tx at rising, SIOCON.4 = 1)

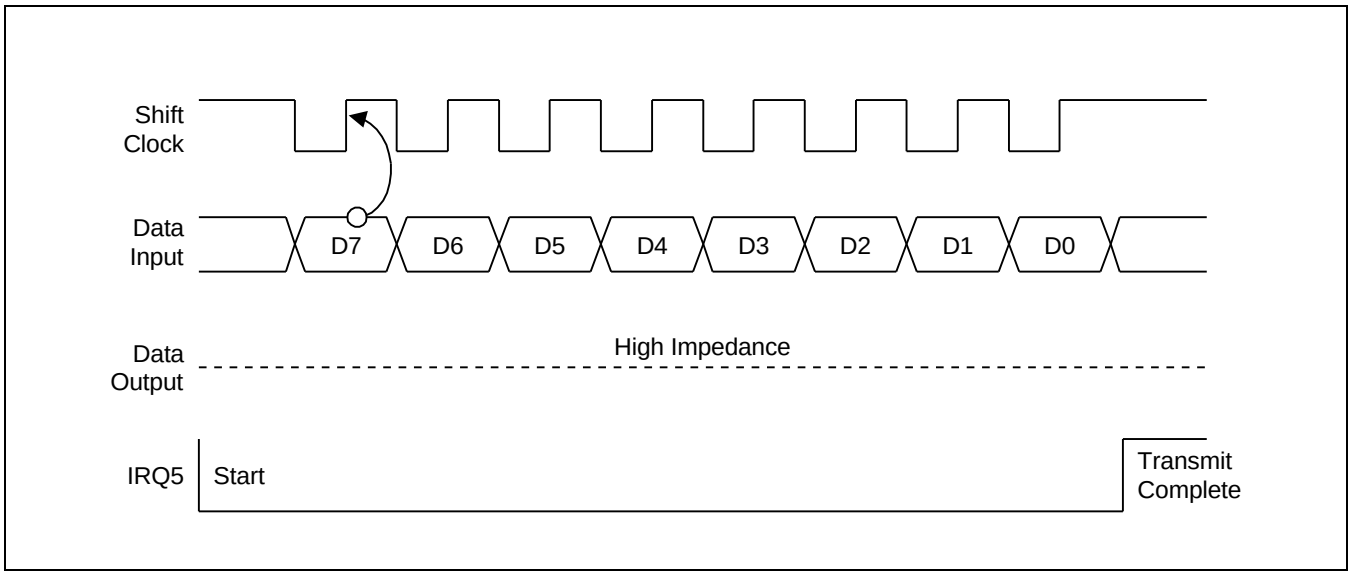


Figure 15-6. Serial I/O Timing in Receive-Only Mode

☞ **PROGRAMMING TIP — SIO**

```
                ORG         0000H

                VECTOR      00H, INT_9498       ;  S3C9498/F9498 has only one interrupt vector

                ORG         0100H
INITIAL:

                LD          SYM, #00H           ;  Global/Fast interrupt disable -> SYM
                LD          BTCON, #10100010B   ;  Watch-dog disable
                LD          CLKCON, #00011000B  ;  non-divided CPU clock
                LD          SP, #0C0H           ;  9498 → 00~BF (After decrease, push data)

                •
                •
                LD          P3CON, #10111100B   ;  SIO setting
                •
                •
                LD          SIOCON, #00100110B  ;  Enable SIO/Interrupt
                LD          SIOPS, #20          ;  setting baud rate
                •
                •
                EI
```

☞ PROGRAMMING TIP — SIO (Continued)

```
MAIN:
            •
            •
            •
            CALL    SUB_SIO             ;  Data transmit routine
            •
            •
            •
            JP      MAIN
SUB_SIO:
            LD      SIODATA, TRANSBUF   ;  1-byte transmission
            OR      SIOCON, #00001000B  ;  Shift start (8-bit transmit)
            •
            •
            RET
INT_9498:                               ;  S3C9498/F9498 has just one interrupt vector
            LD      R0, SIOCON
            AND     R0, #00000011B
            CP      R0, #00000011B
            JP      EQ, INT_SIO         ;  SIOCON's pending bit & INT. enable bit check
INT_SIO:
            AND     SIOCON, #11111110   ;  Pending bit clear
            •
            •
            •
            IRET
```

16 12-BIT PWM (PULSE WIDTH MODULATION)

OVERVIEW

This microcontroller has the 12-bit PWM circuit. The operation of all PWM circuit is controlled by a single control register, PWMCON.

The PWM counter is a 12-bit incrementing counter. It is used by the 12-bit PWM circuits. To start the counter and enable the PWM circuits, you set PWMCON.2 to "1". If the counter is stopped, it retains its current count value; when re-started, it resumes counting from the retained count value. When there is a need to clear the counter you set PWMCON.3 to "1".

You can select a clock for the PWM counter by set PWMCON.6-.7. Clocks which you can select are Fosc/256, Fosc/64, Fosc/8, Fosc/1.

FUNCTION DESCRIPTION

PWM

The 12-bit PWM circuits have the following components:

— 6-bit comparator and extension cycle circuit

— 6-bit reference data registers (PWMDATA)

— 6-bit extension data registers (PWMEX)

— PWM output pins (P2.7/PWM)

PWM counter

The PWM counter is a 12-bit incrementing counter comprised of a lower 6-bit counter and an upper 6-bit counter.

To determine the PWM module's base operating frequency, the lower byte counter is compared to the PWM data register value. In order to achieve higher resolutions, the six bits of the upper counter can be used to modulate the "stretch" cycle. To control the "stretching" of the PWM output duty cycle at specific intervals, the 6-bit extended counter value is compared with the 6-bit value (bits 7-2) that you write to the module's extension register.

PWM data and extension registers

PWM (duty) data registers, located in E4H , determine the output value generated by each 12-bit PWM circuit. These registers, PWM is read/write addressable.

— 8-bit data register PWMDATA, of which only bits 5-0 are used.

— 8-bit extension registers PWMEX (E5H), of which only bits 7-2 are used

To program the required PWM output, you load the appropriate initialization values into the 6-bit data registers (PWMDATA) and the 6-bit extension registers (PWMEX). To start the PWM counter, or to resume counting, you set PWMCON.2 to "1".

A reset operation disables all PWM output. The current counter value is retained when the counter stops. When the counter starts, counting resumes at the retained value.

PWM clock rate

The timing characteristics of both 12-bit output channels are identical, and are based on the Fosc clock frequency. The counter clock value is determined by the setting of PWMCON.6-.7.

Table 16-1. PWM Control and Data Registers

Register Name	Mnemonic	Address	Function
PWM data registers	PWMDATA	E4H	6-bit PWM basic cycle frame value
	PWMEX	E5H	6-bit extension ("stretch") value
PWM control registers	PWMCON	EDH	PWM counter stop/start (resume), and Fosc clock settings

PWM function Description

The PWM output signal toggles to Low level whenever the lower 6-bit counter matches the reference value stored in the module's data register (PWMDATA). If the value in the PWMDATA register is not zero, an overflow of the lower counter causes the PWM output to toggle to High level. In this way, the reference value written to the data register determines the module's base duty cycle.

The value in the 6-bit extension counter is compared with the extension settings in the 6-bit extension data register (PWMEX). This 6-bit extension counter value, together with extension logic and the PWM module's extension register , is then used to "stretch" the duty cycle of the PWM output. The "stretch" value is one extra clock period at specific intervals, or cycles (see Table 16-2).

If, for example, the value in the extension register is '04H', the 32nd cycle will be one pulse longer than the other 63 cycles. If the base duty cycle is 50 %, the duty of the 32nd cycle will therefore be "stretched" to approximately 51% duty. For example, if you write 80H to the extension register, all odd-numbered pulses will be one cycle longer. If you write FCH to the extension register, all pulses will be stretched by one cycle except the 64th pulse. PWM output goes to an output buffer and then to the corresponding PWM output pin. In this way, you can obtain high output resolution at high frequencies.

Table 16-2. PWM output "stretch" Values for Extension Registers PWMEX

PWMEX Bit	"Stretched" Cycle Number
7	1, 3, 5, 7, 9, . . . , 55, 57, 59, 61, 63
6	2, 6, 10, 14, . . . , 50, 54, 58, 62
5	4, 12, 20, . . . , 44, 52, 60
4	8, 24, 40, 56
3	16, 48
2	32
1	Not used
0	Not used

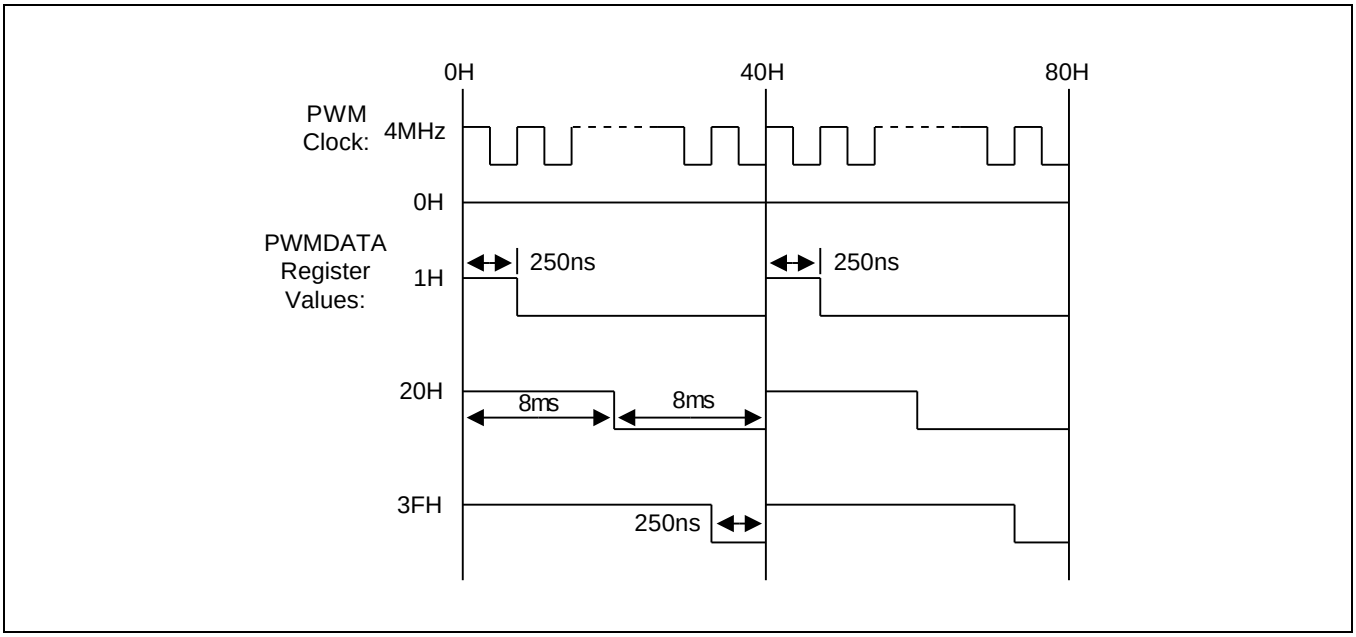


Figure 16-1. 12-Bit PWM Basic Waveform

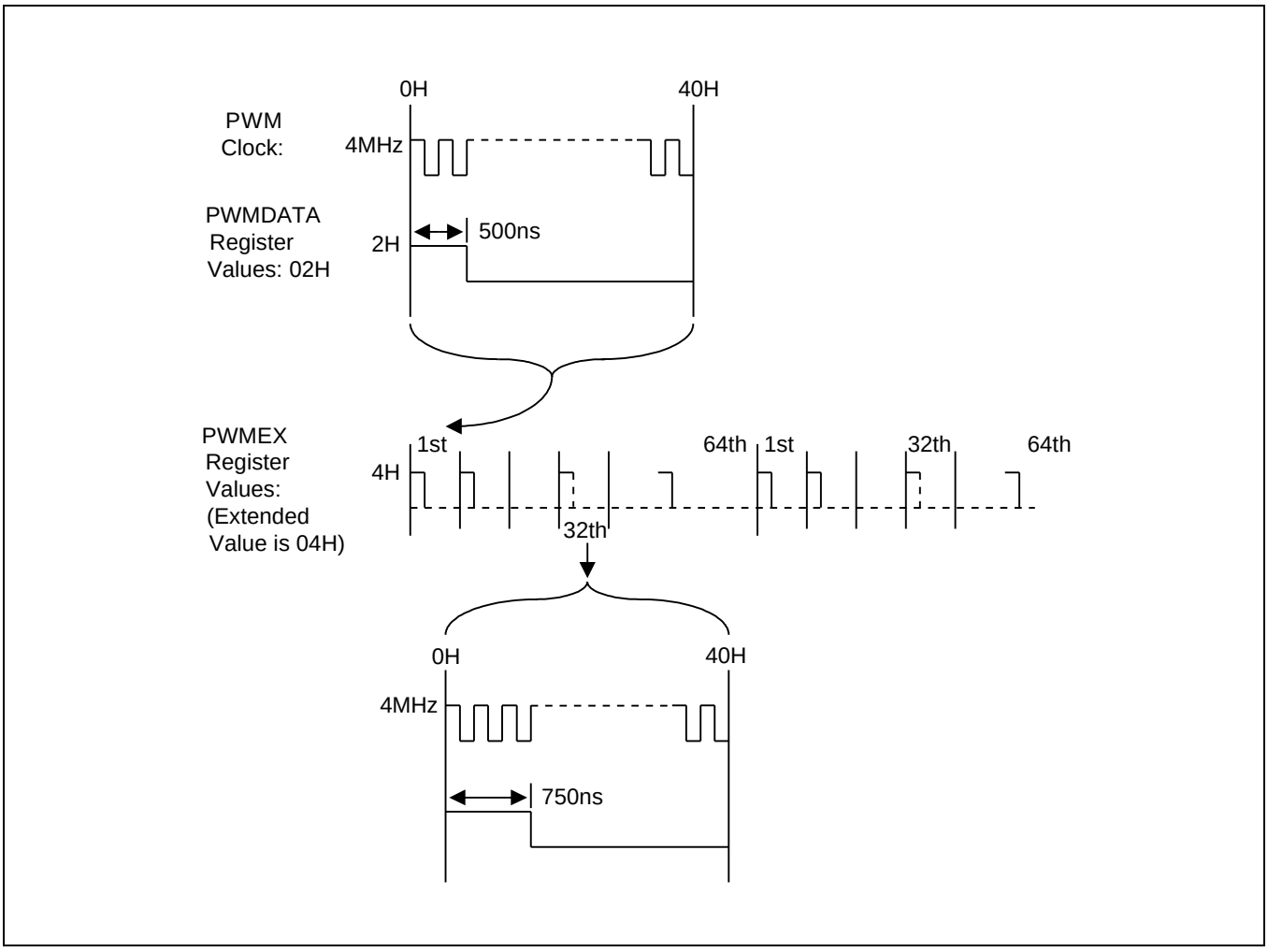


Figure 16-2. 12-Bit Extended PWM Waveform

PWM CONTROL REGISTER (PWMCON)

The control register for the PWM module, PWMCON, is located at register address EDH. PWMCON is used the 12-bit PWM modules. Bit settings in the PWMCON register control the following functions:

— PWM counter clock selection
— PWM data reload interval selection
— PWM counter clear
— PWM counter stop/start (or resume) operation
— PWM counter overflow (upper 6-bit counter overflow) interrupt control

A reset clears all PWMCON bits to logic zero, disabling the entire PWM module.

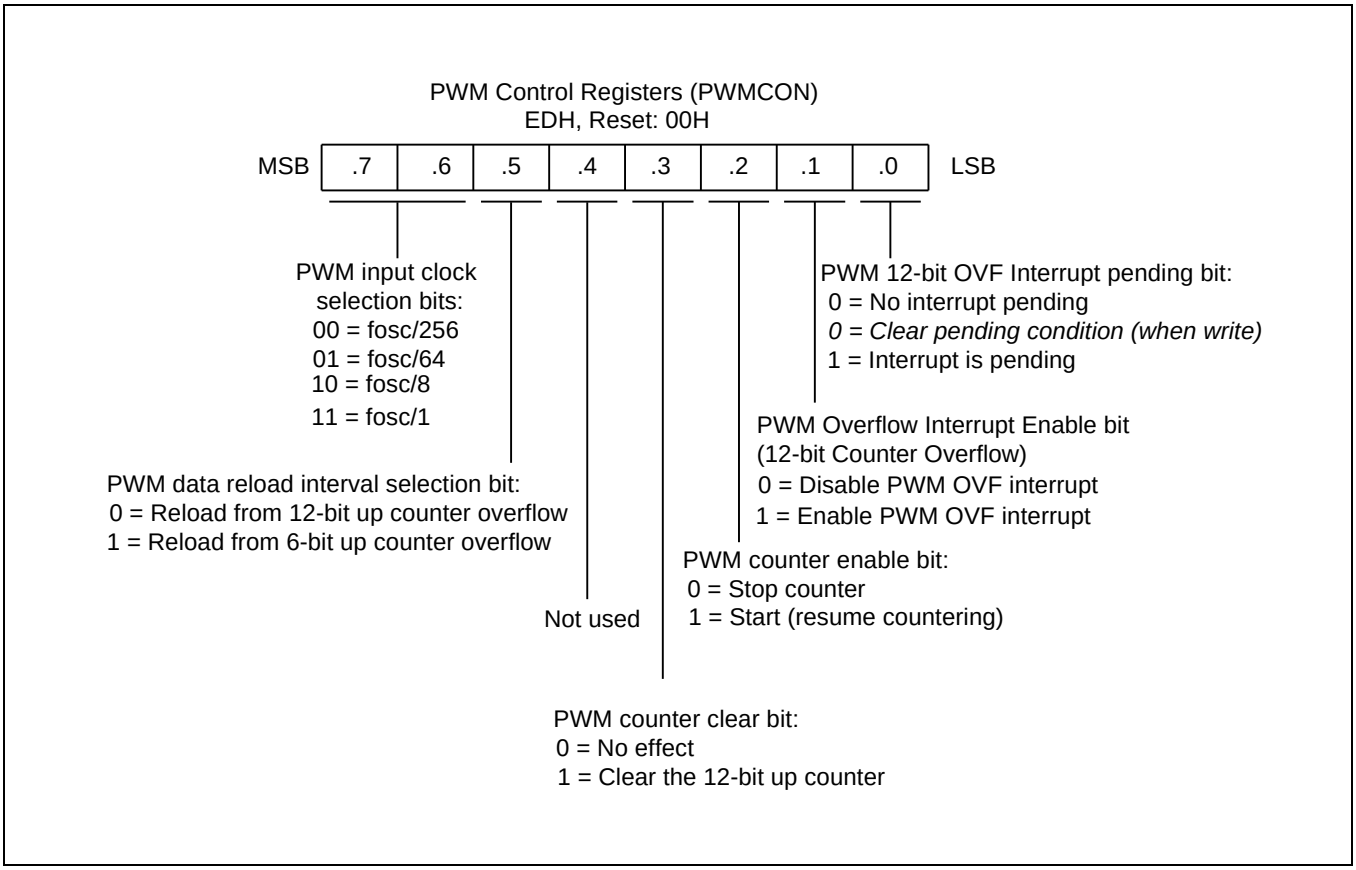


Figure 16-3. PWM/Capture Module Control Register (PWMCON)

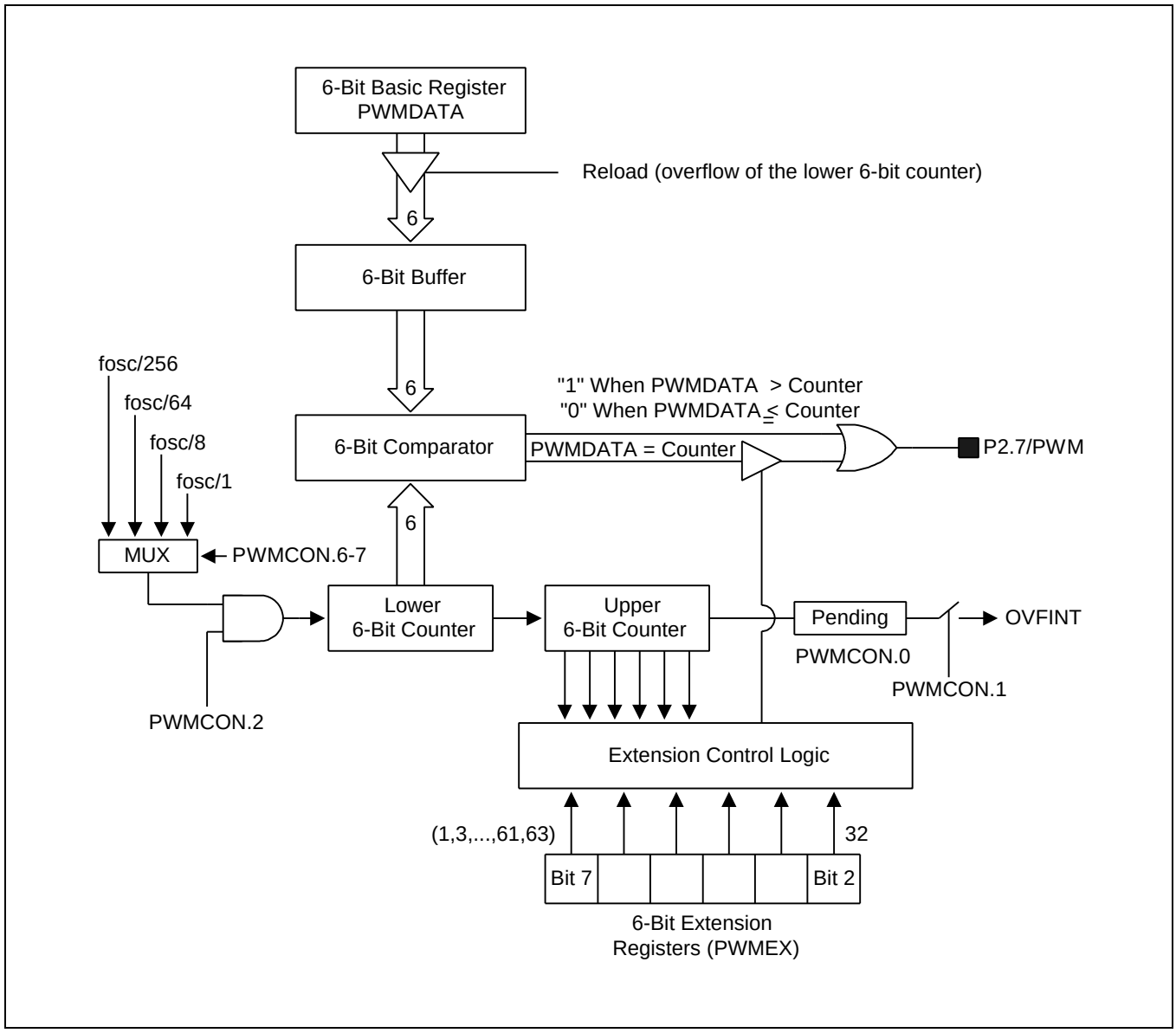


Figure 16-4. PWM/Capture Module Functional Block Diagram

☞ **PROGRAMMING TIP — Programming the PWM Module to Sample Specifications**

```
                ORG         0000H

                VECTOR      00H, INT_9498           ;  S3C9498/F9498 has only one interrupt vector

                ORG         0100H
INITIAL:

                LD          SYM, #00H               ;  Global/Fast interrupt disable -> SYM
                LD          BTCON, #10100010B       ;  Watch-dog disable
                LD          CLKCON, #00011000B      ;  non-divided CPU clock
                LD          SP, #0C0H               ;  9498 → 00–BF (After decrease, push data)

                •
                •
                LD          P2CONH, #0C0H           ;  P0.7 PWM0 output

                LD          PWMEX, #0               ;  Extension register setting
                LD          PWMDATA, #20H           ;  Data register setting
                LD          PWMCON, #00000100B      ;  Start counting
                                                    ;  Half duty PWM wave out to P0.7
                •
                •
                •
                EI

MAIN:

                •
                •
                •
                CALL        SUB_ROUTINE
                •
                •
                •
                JP          MAIN
SUB_ROUTINE:

                NOP
                •
                •
                •
                RET
```

NOTES

17 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 nine input channels to equivalent 10-bit digital values. The analog input level must lie between the AV_{REF} and V_{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)

— Eight multiplexed analog data input pins (AD0 – AD7), alternately digital data I/O port

— 10-bit A/D conversion data output register (ADDATAH/L)

— $AV_{REF,}$ AV_{SS} (AV_{SS} is internally connected to V_{SS})

FUNCTION DESCRIPTION

To initiate an analog-to-digital conversion procedure, at the first you must set port control register(P1CONH/L) for AD analog input. And you write the channel selection data in the A/D converter control register ADCON.4-.7 to select one of the eight analog input pins (AD0-7) and set the conversion start bit, ADCON.0. The read-write ADCON register is located at address FCH. The unused pin can be used for normal I/O.

During a normal conversion, ADC logic initially sets the successive approximation register to 200H (the approximate half-way point of a 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.4 - 7) in the ADCON register. To start the A/D conversion, you should set the enable bit, ADCON.0. When a conversion is completed, the end-of-conversion (EOC) bit is automatically set to 1 and the result is dumped into the ADDATAH/L register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATAH/L 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-AD7input 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 a 10-bit conversion: When Fxx/8 is selected for conversion clock with a 8 MHz fxx clock frequency, one clock cycle is 1 us. Each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

4 clocks/bit × 10 bits + set-up time = 50 clocks, 50 clock × 1us = 50 us at 8 MHz

A/D CONVERTER CONTROL REGISTER (ADCON)

The A/D converter control register, ADCON, is located at address FCH. It has three functions:

— Analog input pin selection (bits 4, 5, 6, and 7)

— A/D conversion End-of-conversion (ECO) status (bit 3)

— A/D conversion speed selection (bits 1,2)

— A/D operation start (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 (ADC0–ADC7)can be selected dynamically by manipulating the ADCON.4–7 bits. And the pins not used for analog input can be used for normal I/O function.

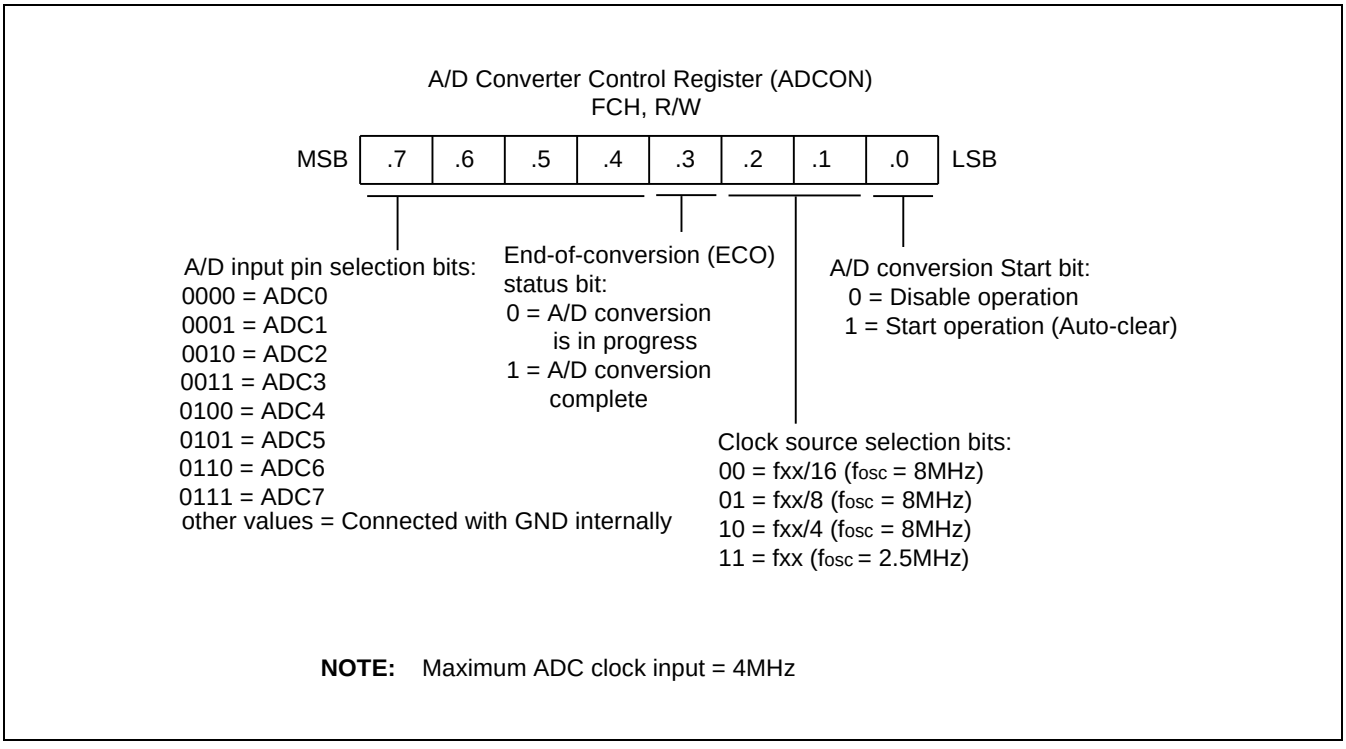


Figure 17-1. A/D Converter Control Register (ADCON)

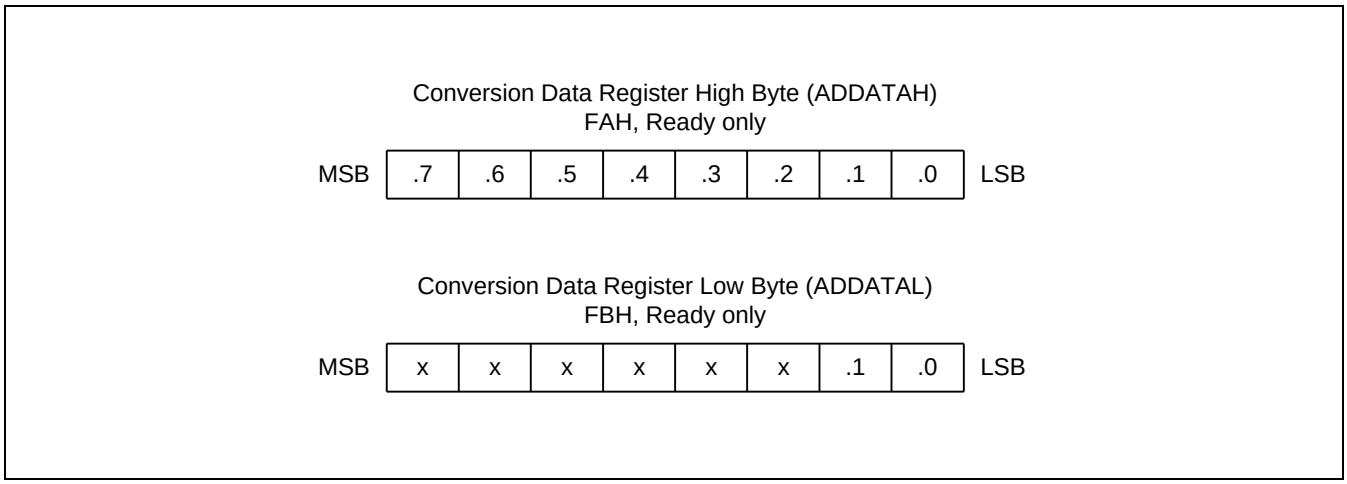


Figure 17-2. A/D Converter Data Register (ADDATAH/L)

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 V_{SS} to AV_{REF} (usually, AV_{REF} = V_{DD}).

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

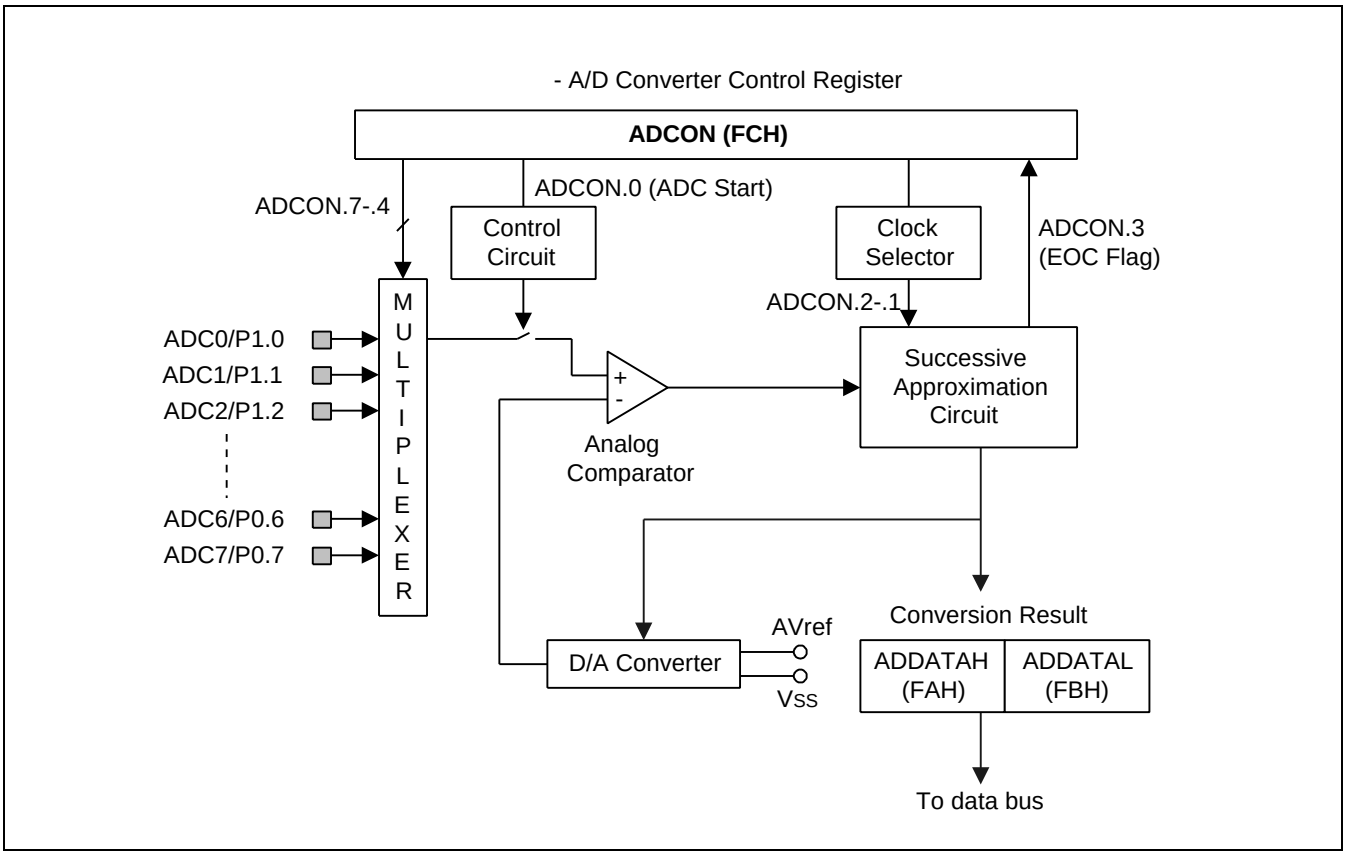
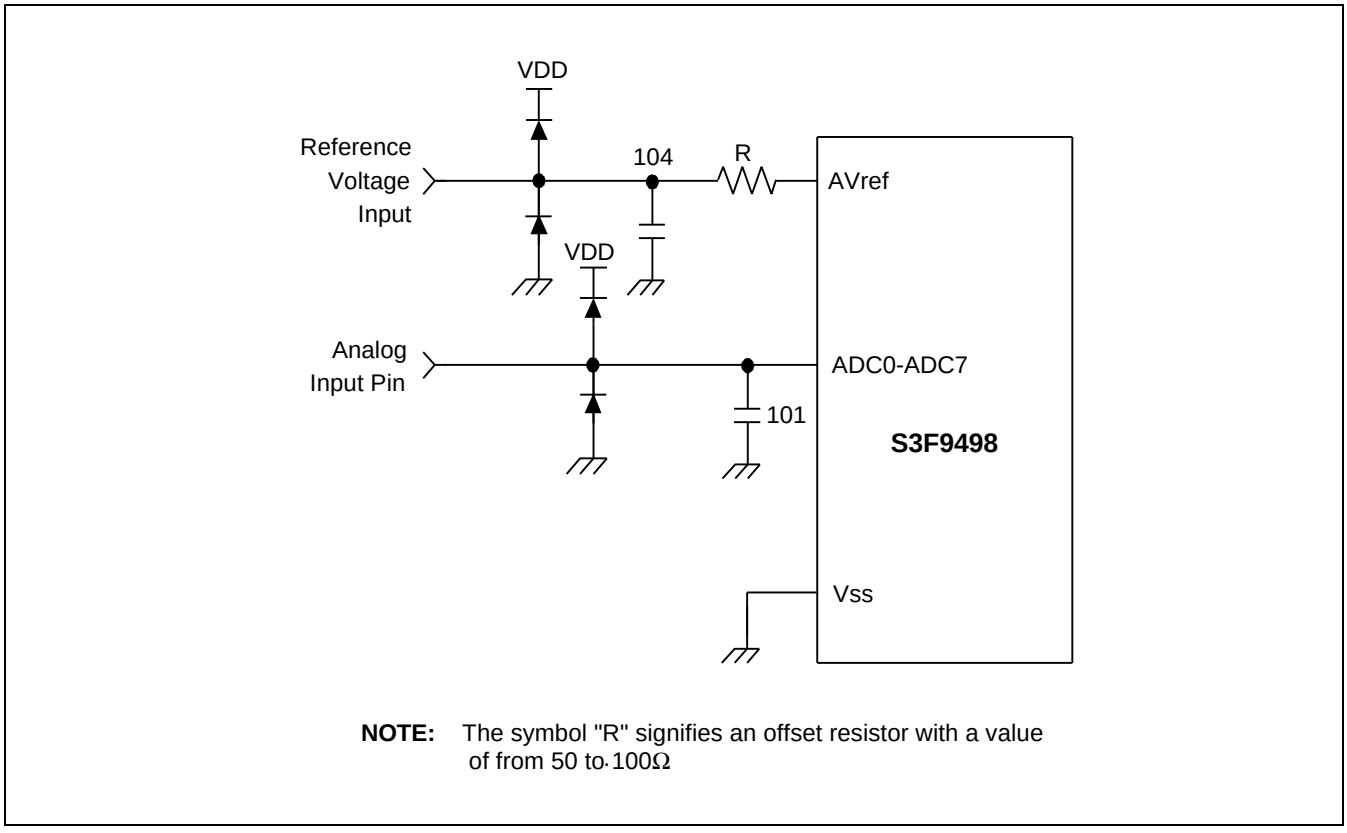


Figure 17-3. A/D Converter Functional Block Diagram

INTERNAL A/D CONVERSION PROCEDURE

1. Analog input must remain between the voltage range of V_{SS} and AV_{REF}.
2. Configure P1.0–P1.7 for analog input before A/D conversions. To do this, you load the appropriate value to the P1CONH and P1CONL (for ADC0–ADC7) registers.
3. Before the conversion operation starts, you must first select one of the eight input pins (ADC0–ADC7) by writing the appropriate value to the ADCON register.
4. When conversion has been completed, (50 clocks have elapsed), the EOC, ADCON.3 flag is set to "1", so that a check can be made to verify that the conversion was successful.
5. The converted digital value is loaded to the output register, ADDATAH (8-bit) and ADDATAL (2-bit), then the ADC module enters an idle state.
6. The digital conversion result can now be read from the ADDATAH and ADDATAL register.



NOTE: The symbol "R" signifies an offset resistor with a value of from 50 to 100Ω

Figure 17-4. Recommended A/D Converter Circuit for Highest Absolute Accuracy

NOTES

18 ELECTRICAL DATA

OVERVIEW

In this section, the following S3C9498/F9498 electrical characteristics are presented in tables and graphs:

— Absolute maximum ratings
— D.C. electrical characteristics
— A.C. electrical characteristics
— Operating Voltage Range
— Schmitt trigger input characteristics
— Oscillator characteristics
— Oscillation stabilization time
— Data retention supply voltage in Stop mode
— Stop mode release timing when initiated by a RESET
— Power-on RESET circuit characteristics
— A/D converter electrical characteristics

Table 18-1. 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	All input ports	– 0.3 to V_{DD} + 0.3	V
Output voltage	V_O	All output ports	– 0.3 to V_{DD} + 0.3	V
Output current high	I_{OH}	One I/O pin active	– 25	mA
		All I/O pins active	– 80	
Output current low	I_{OL}	One I/O pin active	+ 30	mA
		Total pin current for ports 1, 2, 3	+ 100	
		Total pin current for ports 0	+ 200	
Operating temperature	T_A	–	– 25 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	°C

Table 18-2. D.C. Electrical Characteristics

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.2 V to 5.5 V)

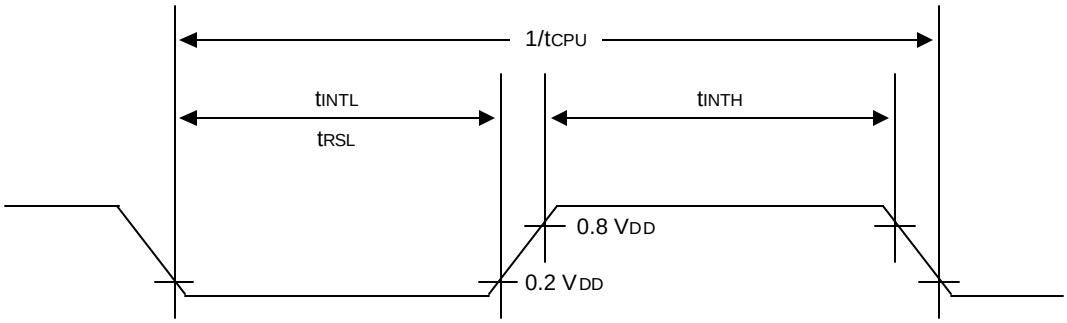
Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Input high voltage	V_{IH1}	Ports 0, 1, 2, 3 and nRESET	V_{DD}= 2.2 to 5.5 V	0.8 V_{DD}	–	V_{DD}	V
	V_{IH3}	X_{IN} and X_{OUT}		V_{DD} – 0.1			
Input low voltage	V_{IL1}	Ports 0, 1, 2, 3 and nRESET	V_{DD}= 2.2 to 5.5 V	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN} and X_{OUT}				0.1	
Output high voltage	V_{OH}	I_{OH} = – 3 mA ports 0-3	V_{DD}= 4.5 to 5.5 V	V_{DD} – 1.0	V_{DD} – 0.4	–	V
Output low voltage	V_{OL}	I_{OL} = 8 mA port 0-3	V_{DD}= 4.5 to 5.5 V	–	0.4	2.0	V
Input high leakage current	I_{LIH1}	All input pins except I_{LIH2}	V_{IN} = V_{DD}	–	–	1	µA
	I_{LIH2}	X_{IN}, X_{OUT}	V_{IN} = V_{DD}			20	
Input low leakage current	I_{LIL1}	All input pins except I_{LIL2}	V_{IN} = 0 V	–	–	– 1	µA
	I_{LIL2}	X_{IN}, X_{OUT}	V_{IN} = 0 V			– 20	
Output high leakage current	I_{LOH}	All output pins	V_{OUT} = V_{DD}	–	–	2	µA
Output low leakage current	I_{LOL}	All output pins	V_{OUT} = 0 V	–	–	– 2	µA
Pull-up resistor	R_P	V_{IN} = 0 V Port 0-3	V_{DD} = 5V, T_A = 25 °C	25	50	100	kΩ
Supply current	I_{DD1}	RUN mode 8-MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	6	12	mA
		3-MHz CPU clock	V_{DD} = 2.2 to 3 V		2	4	
	I_{DD2}	Idle mode 8-MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	1.5	3	
		3-MHz CPU clock	V_{DD} = 2.2 to 3 V		1	2	
	I_{DD3}	Stop mode, LVR disable	V_{DD} = 4.5 to 5.5 V T_A = 25 °C	–	1	3	µA
			V_{DD} = 2.2 to 3 V T_A = 25 °C		0.5	2	

NOTE: D.C. electrical values for Supply current (I_{DD1} to I_{DD3}) do not include current drawn through internal pull-up resisters, output port drive current, LVR, and ADC.

Table 18-3. A.C. Electrical Characteristics

(T_A = –25 °C to + 85 °C, V_{DD} = 2.2 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input high, low width	t_{INTH}, t_{INTL}	Port 1(INT0, INT1) V_{DD} = 5V ± 10%	–	200	–	ns
nRESET input low width	t_{RSL} –	Input V_{DD} = 5V ± 10%	–	1	–	us



NOTE: The unit tcpu means one CPU clock period.

Figure 18-1. Input Timing Measurement Points

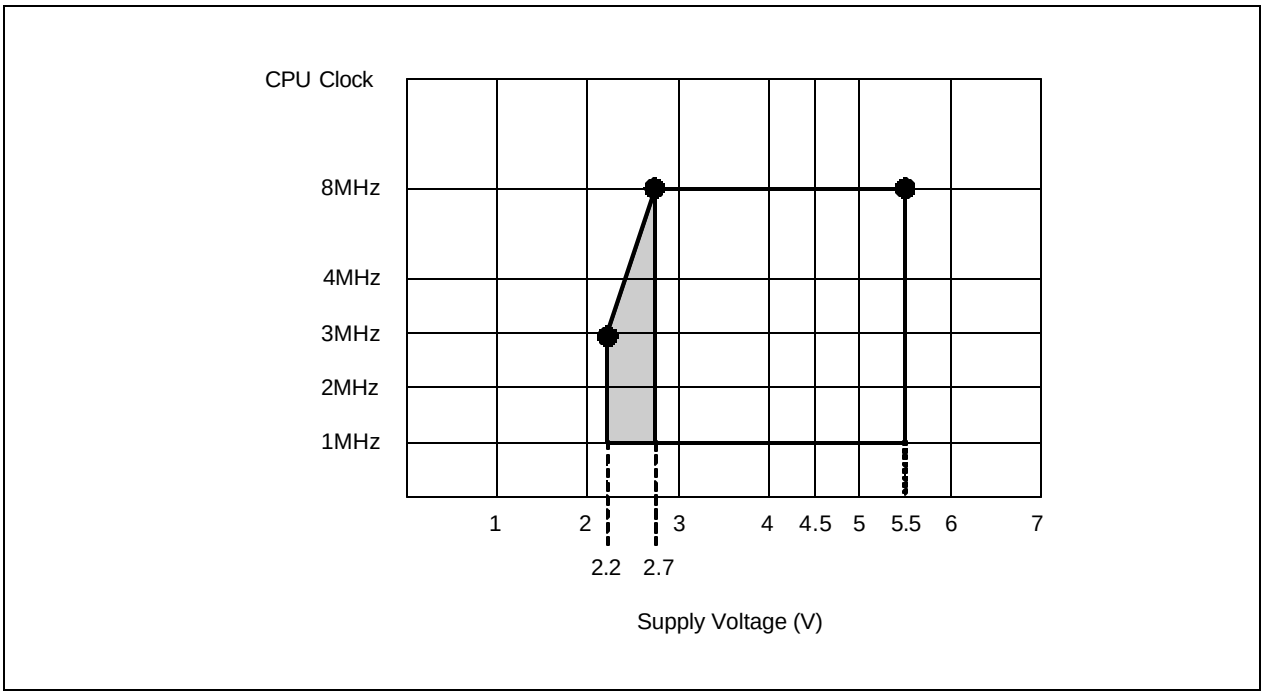


Figure 18-2. Operating Voltage Range (S3C9498/F9498)

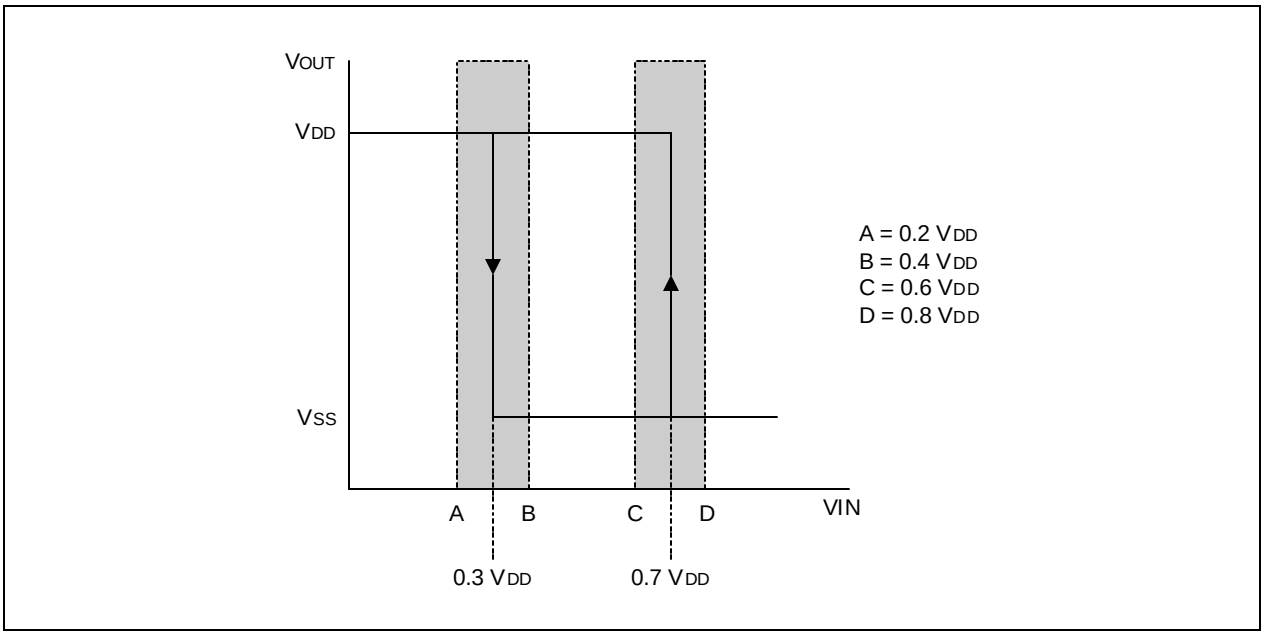


Figure 18-3. Schimtt Trigger Input Characteristic Diagram

Table 18-4. Oscillator Characteristics

(T_A = – 25 °C to + 85 °C)

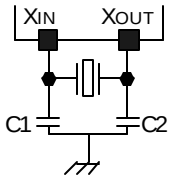
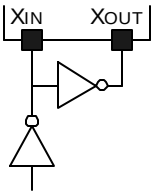
Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Main crystal or ceramic		V_{DD} = 4.5 to 5.5 V V_{DD} = 3.0 to 4.5 V	1	–	8	MHz
External clock (Main system)		V_{DD} = 4.5 to 5.5 V V_{DD} = 3.0 to 4.5 V	1	–	8	

Table 18-5. Oscillation Stabilization Time

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.2 V to 5.5 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Main crystal	fosc > 1.0 MHz	–	–	20	ms
Main ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	10	
External clock (main system)	X_{IN} input high and low width (t_{XH}, t_{XL})	50	–	–	ns
Oscillator stabilization wait time	t_{WAIT} when released by a reset (1)	–	2^{16}/fosc	–	ms
	t_{WAIT} when released by an interrupt (2)	–	–	–	

NOTES:

1. fosc is the oscillator frequency.
2. The duration of the oscillator stabilization wait time, t_{WAIT}, when it is released by an interrupt is determined by the setting in the basic timer control register, BTCON.

Table 18-6. Data Retention Supply Voltage in Stop Mode

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.2 V to 5.5V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	Stop mode	2.2	–	5.5	V
Data retention supply current	I_{DDDR}	Stop mode; V_{DDDR} = 2.2 V	–	0.1	5	µA

NOTE: Supply current does not include current drawn through internal pull-up resistors or external output current loads.

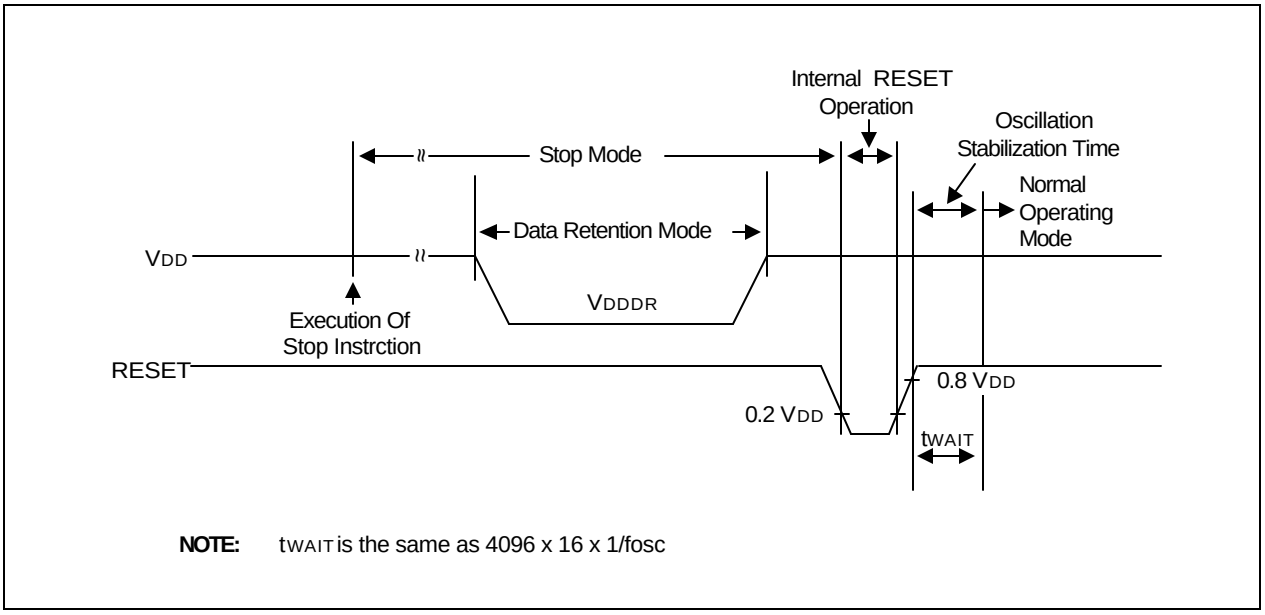


Figure 18-4. Stop Mode Release Timing When Initiated by a RESET

Table 18-7. LVR(Low Voltage Reset) Circuit Characteristics

(T_A = 25 °C)

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
LVR voltage level	V_{LVR}	T_A = 25 °C	2.7	3.0	3.3	V

Table 18-8. A/D Converter Electrical Characteristics

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.2 V to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Total accuracy		V_{DD} = 5.12 V CPU clock = 8 MHz AV_{REF} = 5.12 V AV_{SS} = 0 V	–	–	± 3	LSB
Integral linearity error	ILE	"	–	–	± 3	LSB
Differential linearity error	DLE	"	–	–	± 1	
Offset error of top	EOT	"	–	±1	± 3	
Offset error of bottom	EOB	"	–	±1	± 3	
Conversion time(1)	t_{CON}	fosc = 8 MHz	25	–	–	µs
Analog input voltage	V_{IAN}	–	AV_{SS}	–	AV_{REF}	V
Analog input impedance	R_{AN}	–	2	–	–	MΩ
ADC reference voltage	AV_{REF}	–	2.5	–	V_{DD}	V
ADC reference ground	AV_{SS}	–	V_{SS}	–	V_{SS} + 0.3	V
Analog input current	I_{ADIN}	AV_{REF} = V_{DD} = 5 V	–	–	10	µA
ADC block current (2)	I_{ADC}	AV_{REF} = V_{DD} = 5 V	–	1	3	mA
		AV_{REF} = V_{DD} = 3 V		0.5	1.5	
		AV_{REF} = V_{DD} = 5 V 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 operating current during A/D conversion.

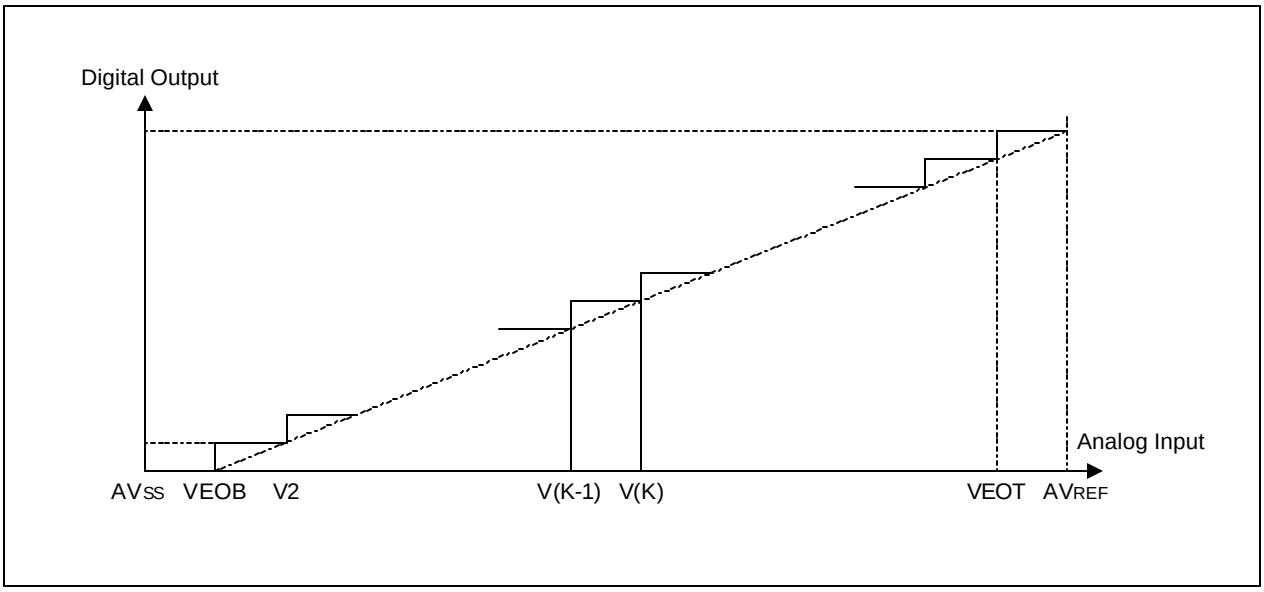


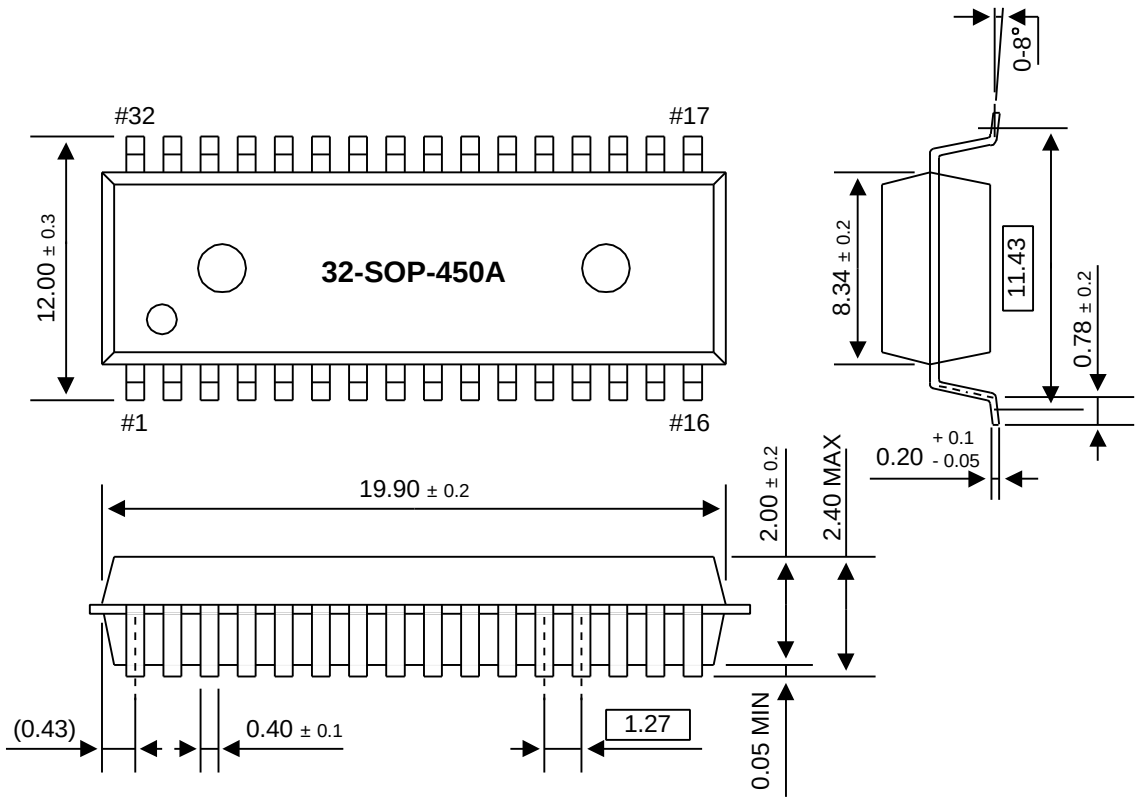
Figure 18-5. Definition of DLE and ILE

NOTES

19 MECHANICAL DATA

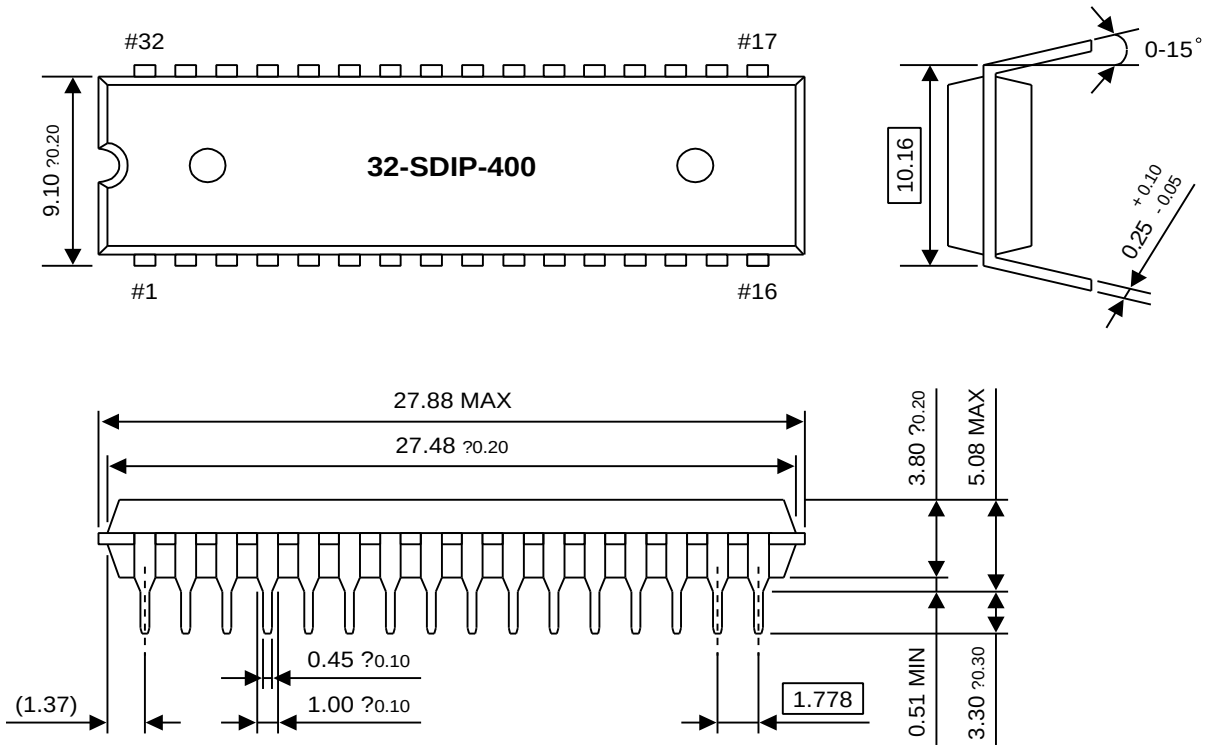
OVERVIEW

The S3C9498/F9498 is available in a 32-pin SDIP package (Samsung: 32-SDIP-400) and a 32-pin SOP package (32-SOP-450A) and 30-pin package(30-SDIP-400) and a 28-pin SOP package (28-SOP-375). Package dimensions are shown in Figures19-1, 19-2, 19-3 and 19-4



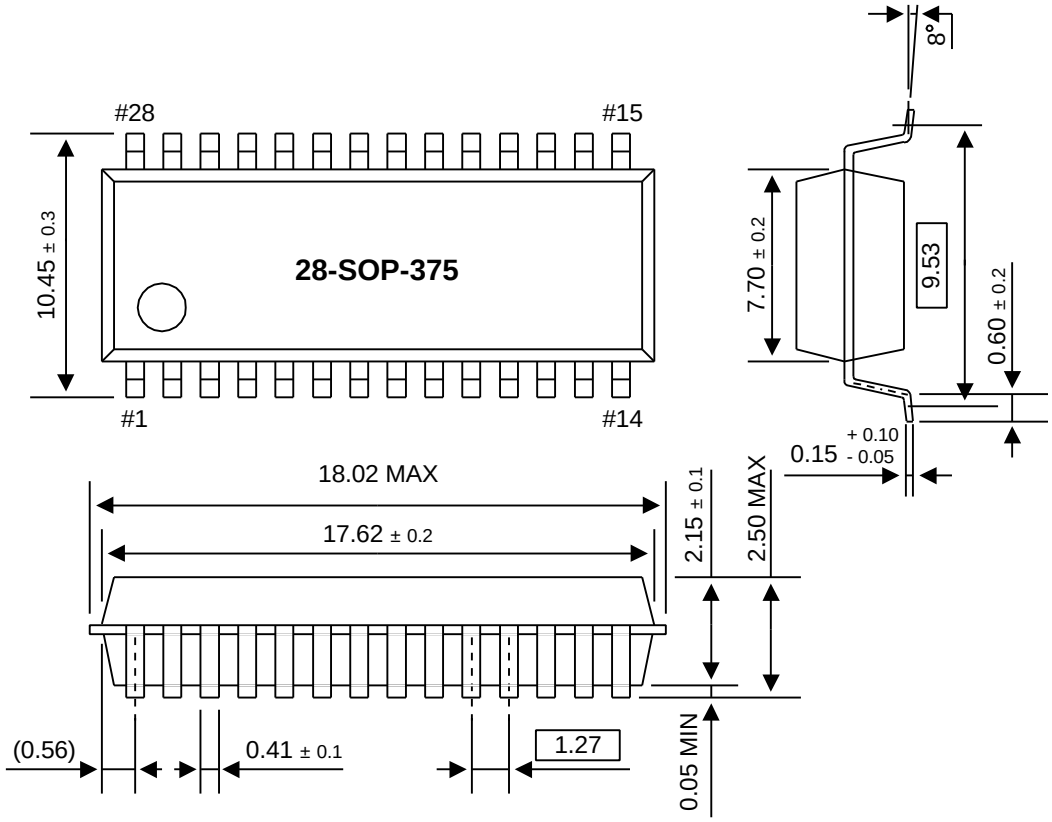
NOTE: Dimensions are in millimeters

Figure 19-1. 32-SOP-450A Package Dimensions



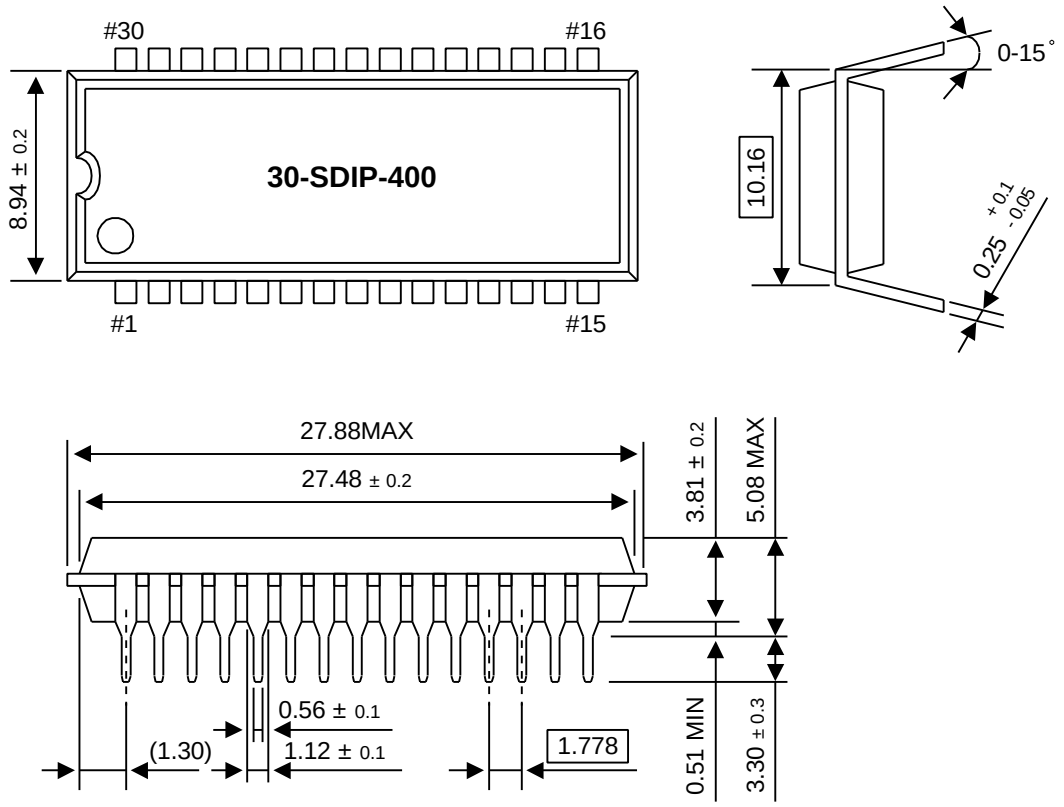
NOTE: Dimensions are in millimeters.

Figure 19-2. 32-SDIP-400 Package Dimensions



NOTE: Dimensions are in millimeters

Figure 19-3. 28-SOP-375 Package Dimensions



NOTE: Dimensions are in millimeters.

Figure 19-4. 30-Pin SDIP Package Dimensions

20 MTP

OVERVIEW

The S3C9498/F9498 single-chip CMOS microcontroller is the MTP (Multi Time Programmable) version of the S3C9498 microcontroller. It has an on-chip Flash ROM instead of masked ROM. The Flash ROM is accessed by serial data format.

The S3C9498/F9498 is fully compatible with the S3C9498, in function, in D.C. electrical characteristics, and in pin configuration. Because of its simple programming requirements, the S3F9488 is ideal for use as an evaluation chip for the S3C9498.

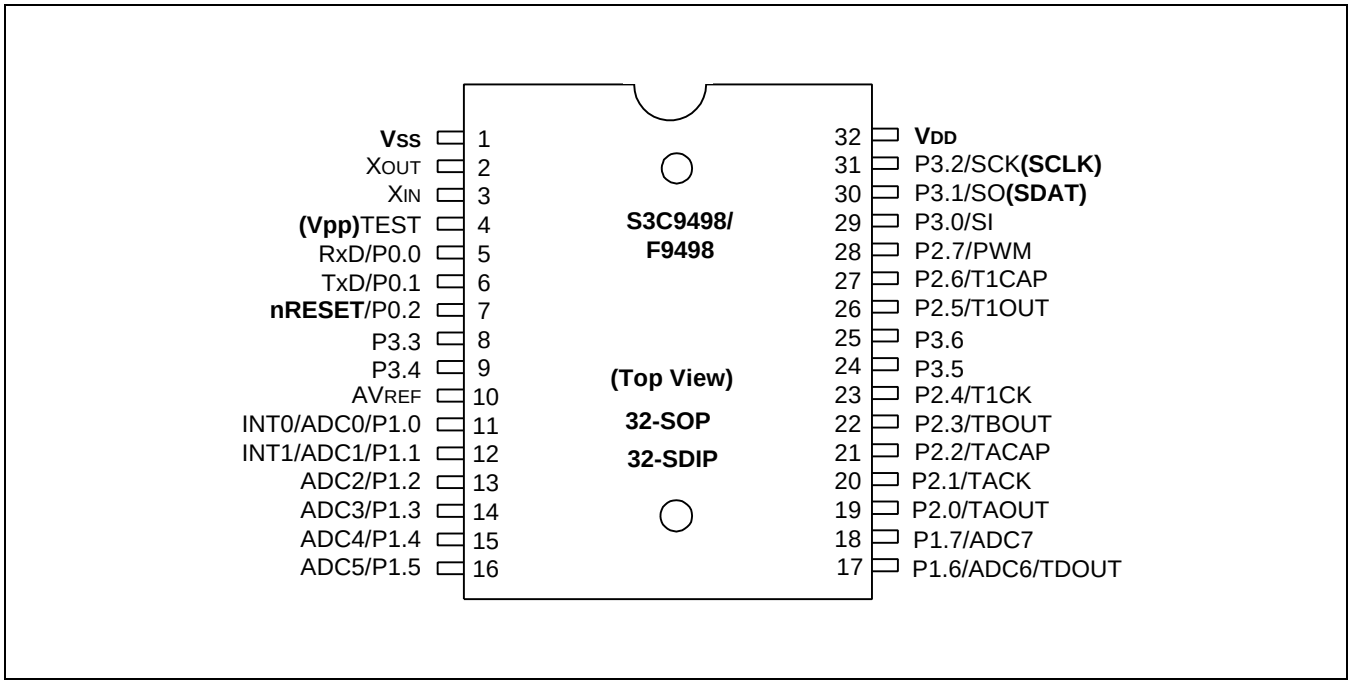


Figure 20-1. Pin Assignment Diagram (32-Pin Package)

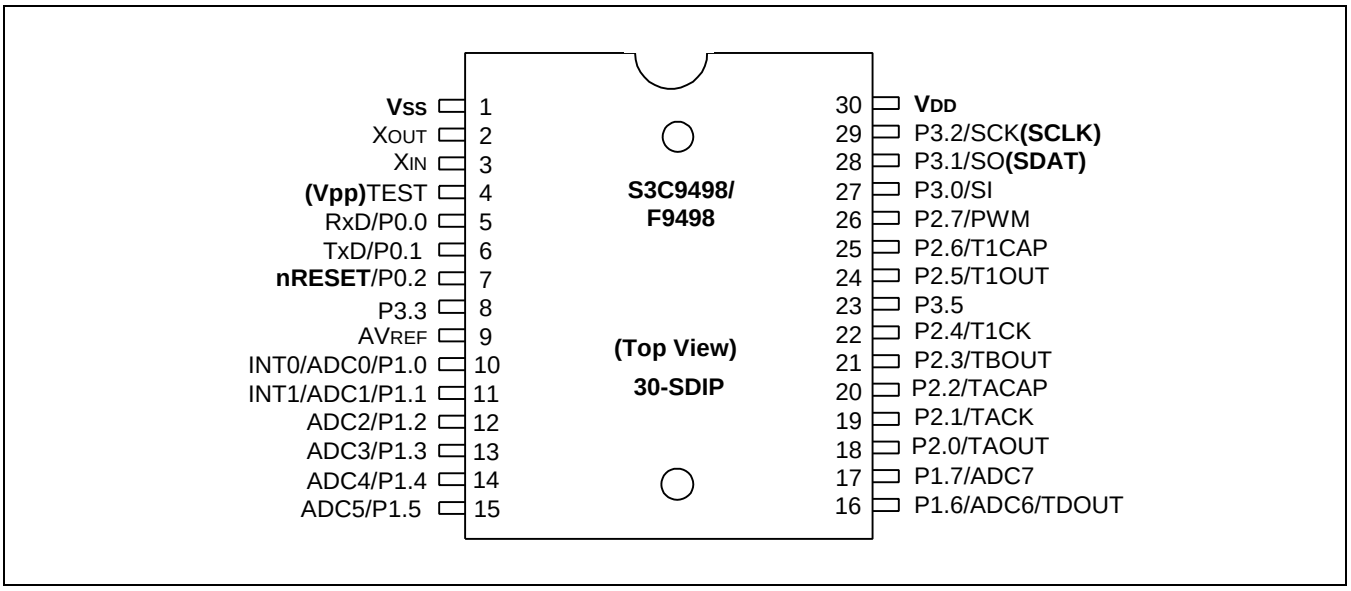


Figure 20-2. Pin Assignment Diagram (30-Pin Package)

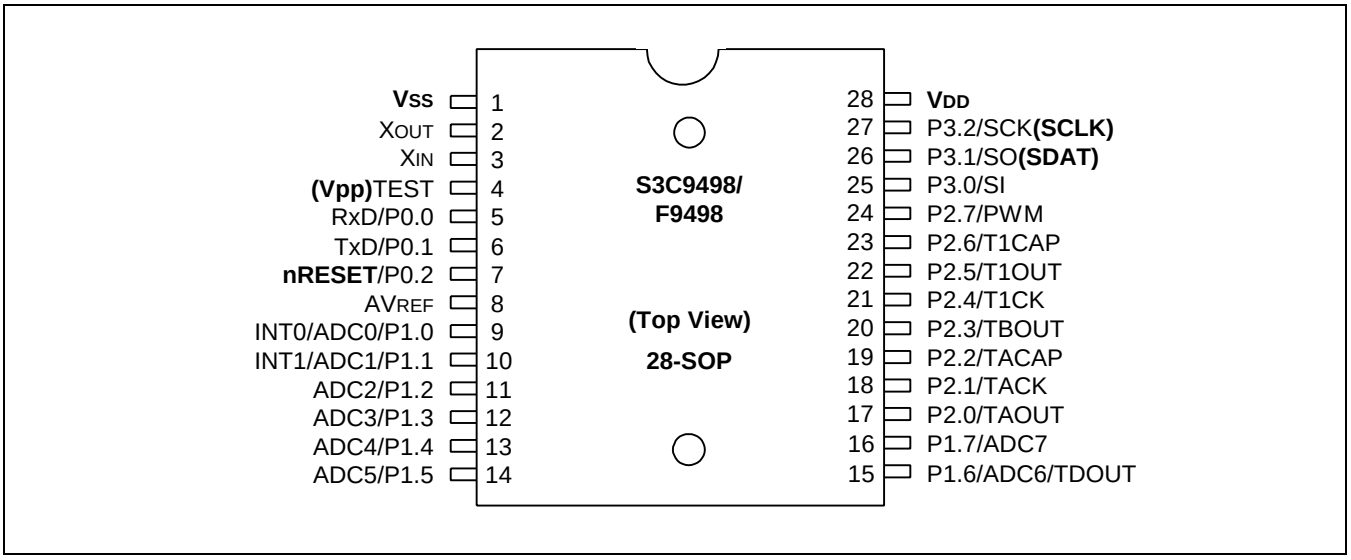


Figure 20-3. Pin Assignment Diagram (28-Pin Package)

Table 20-1. Descriptions of Pins Used to Read/Write the Flash ROM

Main Chip	During Programming			
Pin Name	Pin Name	Pin No.	I/O	Function
P3.1	SDAT	30 (32-pin) 28 (30-pin) 26 (28-pin)	I/O	Serial data pin (output when reading, Input when writing) Input and push-pull output port can be assigned
P3.2	SCLK	31 (32-pin) 29 (30-pin) 27 (28-pin)	I	Serial clock pin (input only pin)
TEST	VPP	4 (32-pin) 4 (30-pin) 4 (28-pin)	I	Power supply pin for flash ROM cell writing (indicates that MTP enters into the writing mode). When 12.5 V is applied, MTP is in writing mode and when 5 V is applied, MTP is in reading mode. (Option)
P0.2	RESETB	7 (32-pin) 7 (30-pin) 7 (28-pin)	I	
V_{DD}/V_{SS}	V_{DD}/V_{SS}	32/1 (32-pin) 30/1 (30-pin) 28/1 (28-pin)	I	Logic power supply pin.

NOTES

21 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system on a turnkey basis. The development support system is composed of a host system, debugging tools, and supporting software. For a host system, any standard computer that employs Win95/98/2000 as its operating system can be used. A sophisticated debugging tool is provided both in hardware and software: the powerful in-circuit emulator, SMDS2+ or SK-1000, for the S3C7-, S3C9-, and S3C8- microcontroller families. SMDS2+ is a newly improved version of SMDS2, and SK-1000 is supported by a third party tool vendor. Samsung also offers supporting software that includes, debugger, an 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 easily sized, moved, scrolled, highlighted, added, or removed.

SASM

The SASM takes a source file containing assembly language statements and translates them into a corresponding source code, an object code and comments. The SASM supports macros and conditional assembly. It runs on the MS-DOS operating system. As it produces the re-locatable object codes only, the user should link object files. Object files can be linked with other object files and loaded into memory. SASM requires a source file and an auxiliary register file (device_name.reg) with device specific information.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generating an object code in the standard hexadecimal format. Assembled program codes include the object code used for ROM data and required In-circuit emulators program control data. To assemble programs, SAMA requires a source file and an auxiliary definition (device_name.def) file with device specific information.

HEX2ROM

HEX2ROM file generates a ROM code from a HEX file which is produced by the assembler. A ROM code is needed to fabricate a microcontroller which has a mask ROM. When generating a ROM code (.OBJ file) by HEX2ROM, the value "FF" is automatically filled into the unused ROM area, up to the maximum ROM size of the target device.

TARGET BOARDS

Target boards are available for all the S3C9-series microcontrollers. All the required target system cables and adapters are included with the device-specific target board. TB9498 is a specific target board for the S3C9498/F9498 development

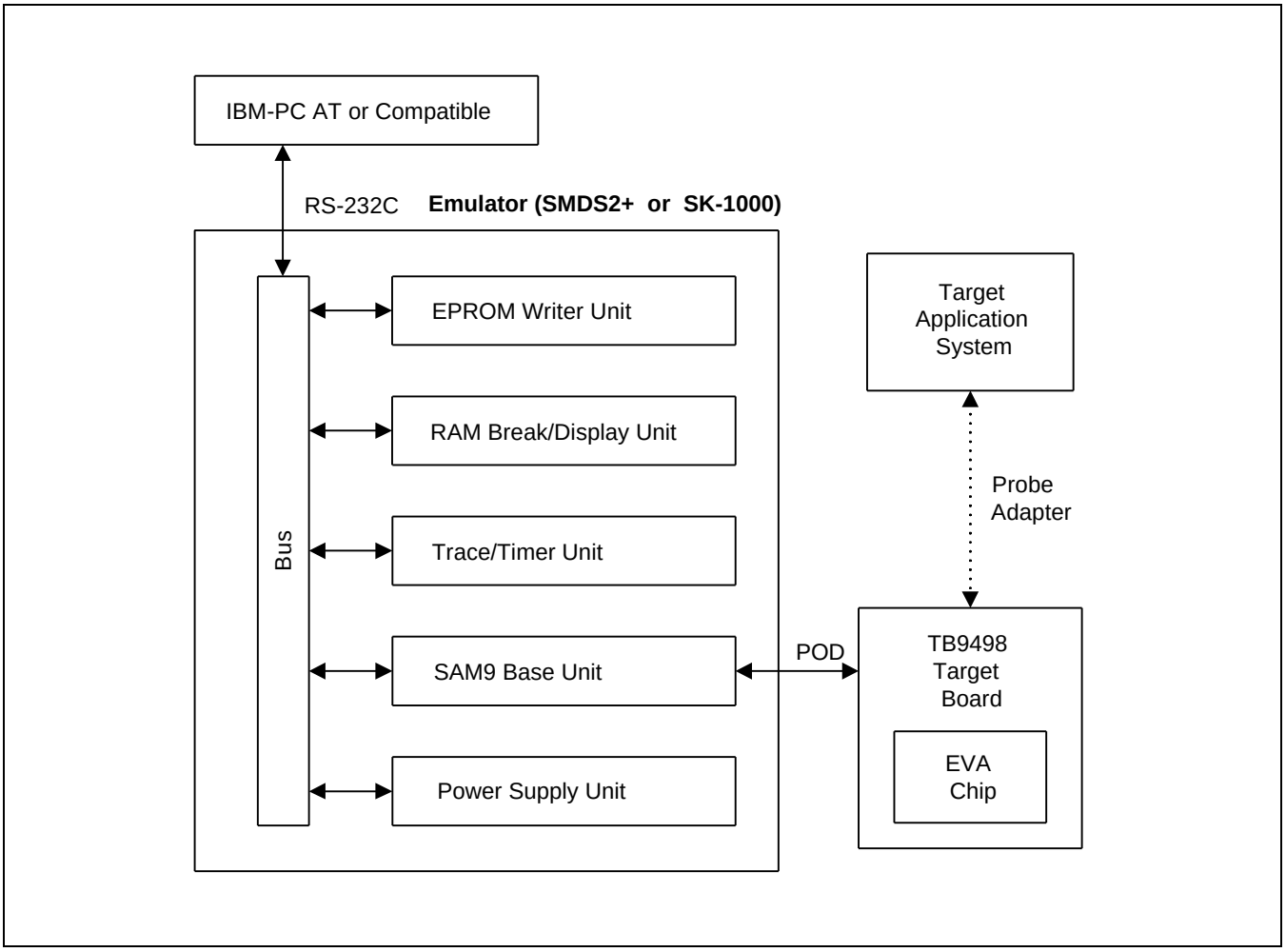


Figure 21-1. SMDS+ or SK-1000 Product Configuration

TB9498 TARGET BOARD

The TB9498 target board is used for the S3C9498/F9498 microcontrollers. It is supported by the SK-1000/SMDS2+ development systems.

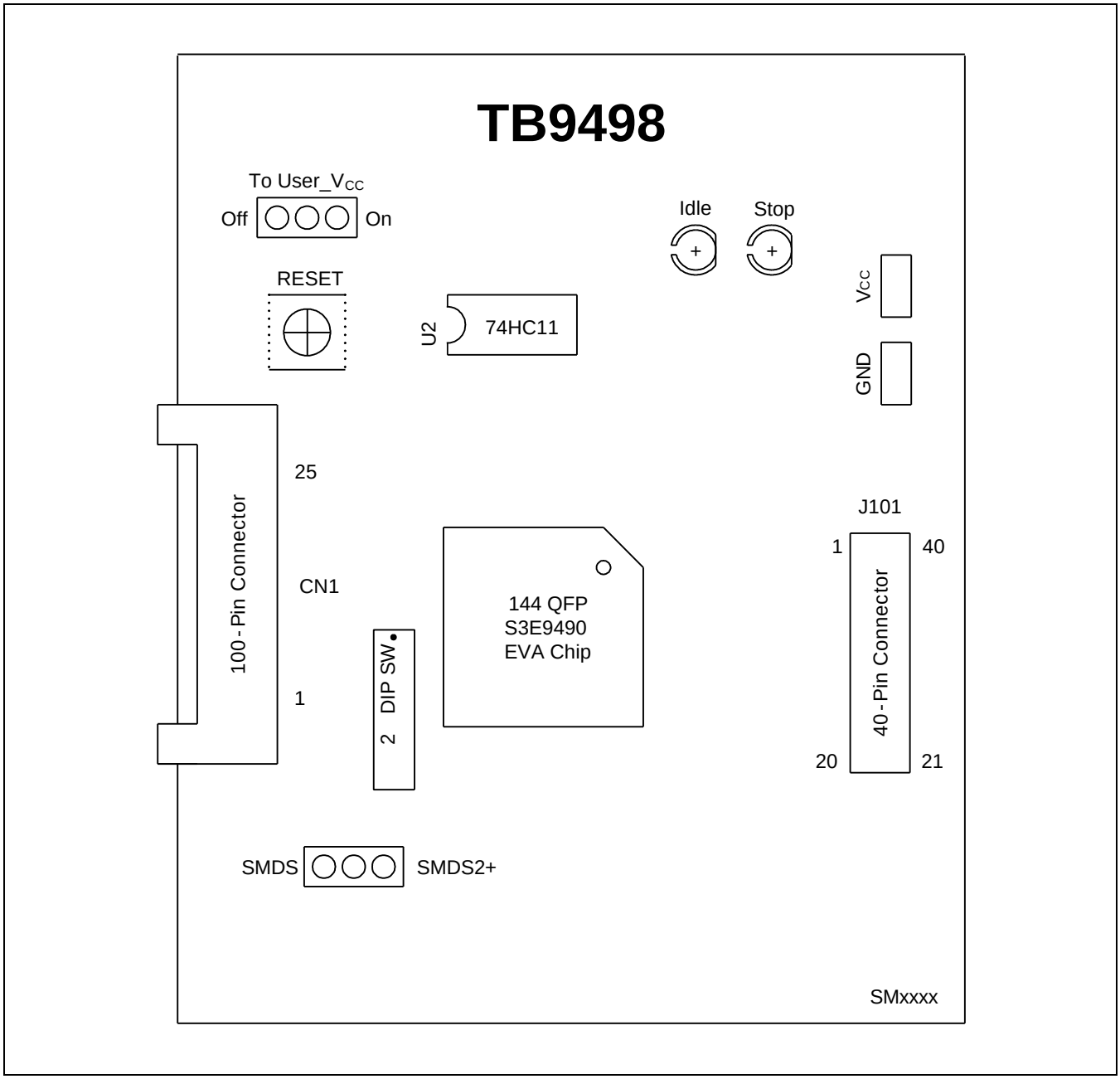
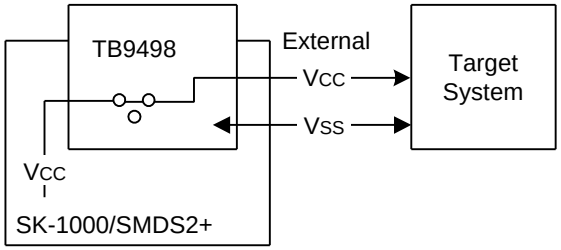
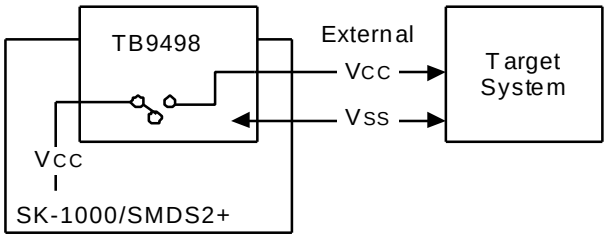


Figure 21-2. TB9498 Target Board Configuration

Table 21-1. Power Selection Settings for TB9498

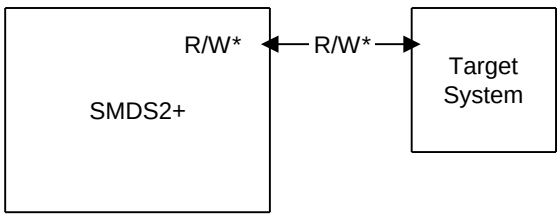
"To User_Vcc" Settings	Operating Mode	Comments
To user_Vcc off  on		The SK-1000/SMDS2+ main board supplies V_{CC} to the target board (evaluation chip) and the target system.
To user_Vcc off  on		The SK-1000/SMDS2+ main board supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

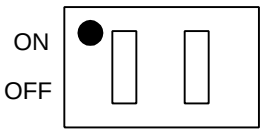
NOTE: The following symbol in the "To User_Vcc" Setting column indicates the electrical short (off) configuration:

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 21-2. The SMDS2+ Tool Selection Setting

"SW1" Setting	Operating Mode
SMDS  SMDS2+	



3EH.7 3FH.0

ON	Low
OFF	High

NOTE: About EVA chip, smart option is determined by DIP switch not software.

Figure 21-3. DIP Switch for Smart Option

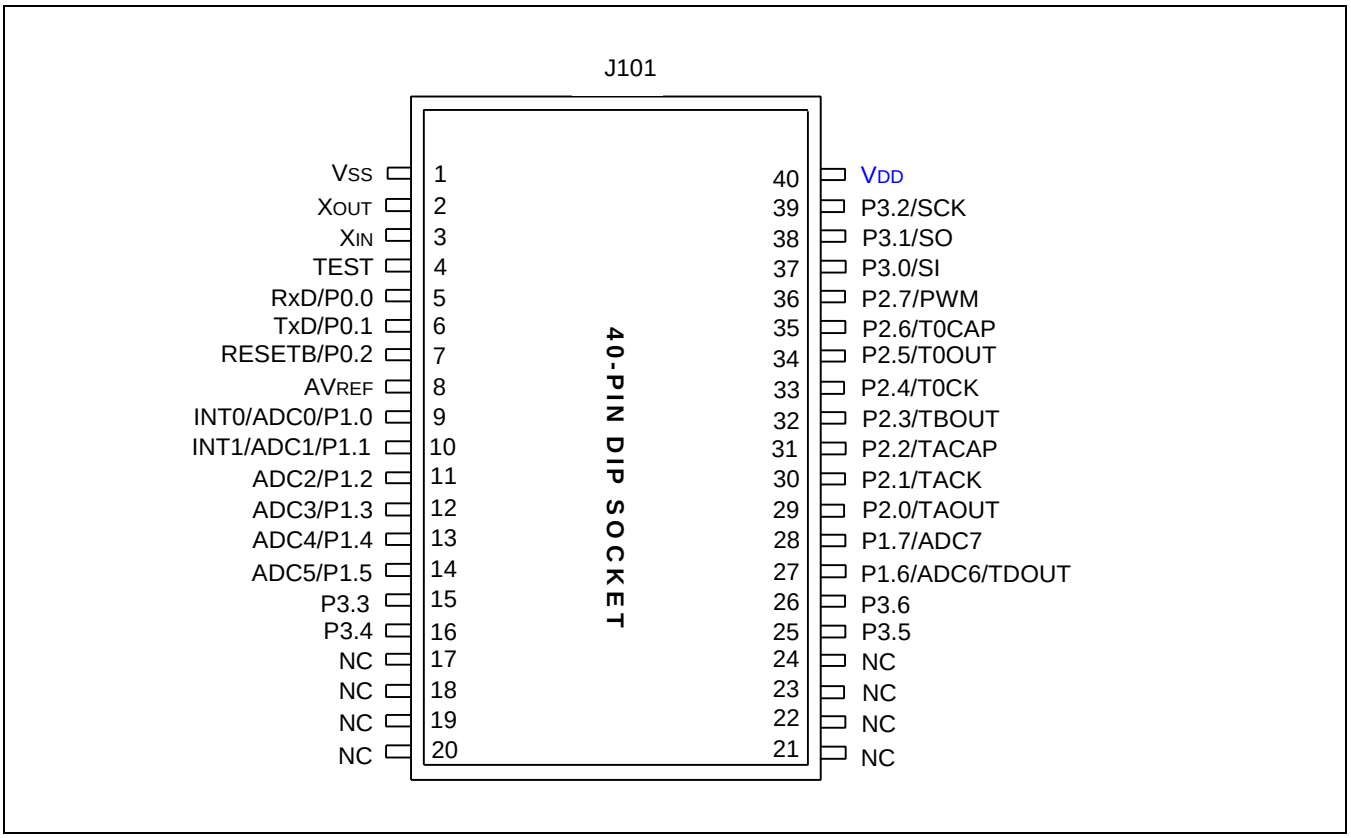


Figure 21-4. 44-Pin Connector for TB9498

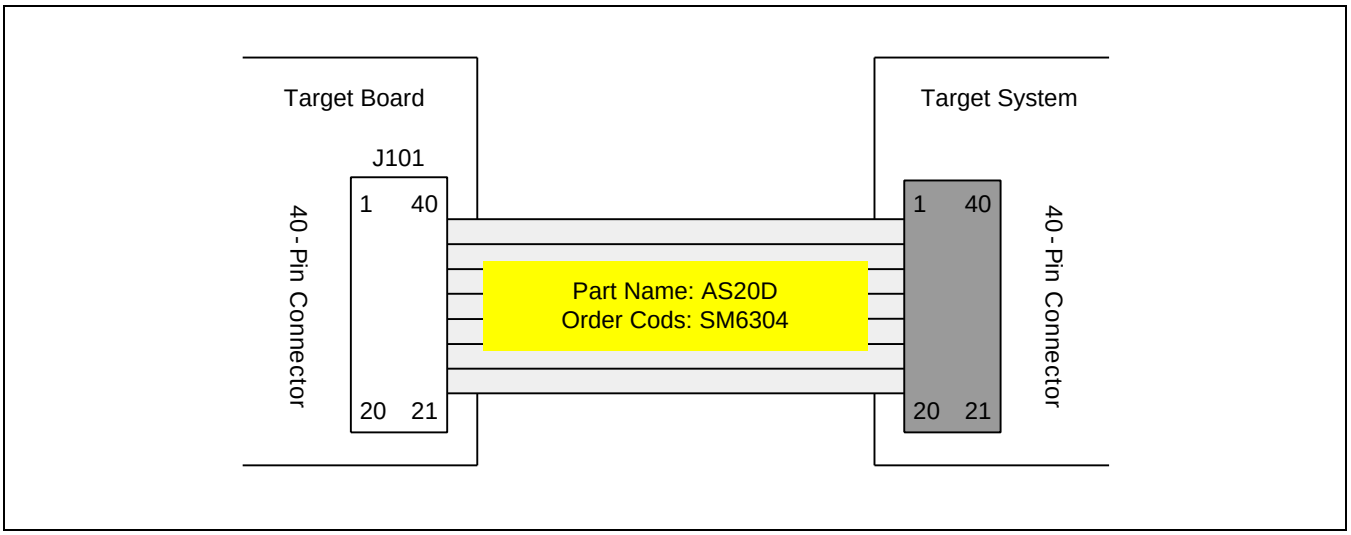


Figure 21-5. S3C9498/F9498 Probe Adapter for 40pin Connector Package