S3C9454B/F9454B

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.

S3C9454B/F9454B 8-Bit CMOS Microcontroller
User's Manual, Revision 1
Publication Number: 21-S3-C9454B/F9454B-200409

Samsung Electronics' microcontroller business has been awarded full ISO-14001 certification (BVQ1 Certificate No. FM9300). All semiconductor products are designed and manufactured in accordance with the highest quality standards and objectives.

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

TEL: (0331) 209-1907
FAX: (0331) 209-1899
Home-Page URL: Http://www.samsungsemi.com/

Printed in the Republic of Korea

Preface

The S3C9454B/F9454B *Microcontroller User's Manual* is designed for application designers and programmers who are using the S3C9454B/F9454B 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 sections:

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 S3C9454B/F9454B with a general product description, and detailed information about individual pin characteristics and pin circuit types.

Chapter 2, "Address Spaces," explains the S3C9454B/F9454B 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 S3C9454B/F9454B 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 chapter 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 S3C9454B/F9454B microcontrollers. Also included in Part II are electrical, mechanical, MTP, and development tools data. It has 10 chapters:

Chapter 7	Clock Circuit	Chapter 12	A/D Converter
Chapter 8	RESET and Power-Down	Chapter 13	Electrical Data
Chapter 9	I/O Ports	Chapter 14	Mechanical Data
Chapter 10	Basic Timer and Timer 0	Chapter 15	S3F945B MTP
Chapter 11	8-bit PWM	Chapter 16	Development Tools

Two order forms are included at the back of this manual to facilitate customer order S3C9454B/F9454B microcontrollers: the Mask ROM Order Form, and the Mask Option Selection Form. You can photocopy these forms, fill them out, and then forward them to your local Samsung Sales Representative.

Table of Contents

Part I — Programming Model

Chapter 1 Product Overview

Chapter 2 Address Spaces

Chapter 3 Addressing Modes

Table of Contents (Continued)

Chapter 4 Control Registers

Chapter 5 Interrupt Structure

Chapter 6 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 and Timers

Table of Contents (Continued)

Chapter 11 8-Bit PWM

Chapter 12 A/D Converter

Chapter 13 Electrical Data

Chapter 14 Mechanical Data

Chapter 15 S3F9454B MTP

Chapter 16 Development Tools

List of Figures

List of Figures (Continued)

List of Figures (Concluded)

List of Tables

List of Programming Tips

List of Register Descriptions

List of Instruction Descriptions

List of Instruction Descriptions (Continued)

1 PRODUCT OVERVIEW

SAM88RCRI PRODUCT FAMILY

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.

S3C9454B/F9454B MICROCONTROLLER

The S3C9454B/F9454B single-chip 8-bit microcontroller is designed for useful A/D converter application field. The S3C9454B/F9454B uses powerful SAM88RCRI CPU and S3C9454B/F9454B architecture. The internal register file is logically expanded to increase the on-chip register space.
The S3C9454B/F9454B has 4K bytes of on-chip program ROM and 208 bytes of RAM. The S3C9454B/F9454B is a versatile general-purpose microcontroller that is ideal for use in a wide range of electronics applications requiring simple timer/counter, PWM. In addition, the S3C9454B/F9454's advanced CMOS technology provides for low power consumption and wide operating voltage range.

Using the SAM88RCRI design approach, the following peripherals were integrated with the SAM88RCRI core:

— Three configurable I/O ports (18 pins)
— Four interrupt sources with one vector and one interrupt level
— One 8-bit timer/counter with time interval mode
— Analog to digital converter with nine input channels(MAX) and 10-bit resolution
— One 8-bit PWM output

The S3C9454B/F9454B microcontroller is ideal for use in a wide range of electronic applications requiring simple timer/counter, PWM, ADC. S3C9454B/F9454B is available in a 20/16-pin DIP and a 20/16-pin SOP and a 20/16-pin SSOP package.

MTP

The S3F9454B is an MTP (Multi Time Programmable) version of the S3C9454B microcontroller. The S3F9454B has on-chip 4-Kbyte multi-time programmable flash ROM instead of masked ROM. The S3F9454B is fully compatible with the S3C9454B, in function, in D.C. electrical characteristics and in pin configuration.

FEATURES

CPU

- SAM88RCRI CPU core
- The SAM88RCRI core is low-end version of the current SAM87 core.

Memory

- 4-Kbyte internal program memory
- 208-byte general purpose register area

Instruction Set

- 41 instructions
- The SAM88RCRI core provides all the SAM87 core instruction except the word-oriented instruction, multiplication, division, and some one-byte instruction.

Instruction Execution Time

- 400 ns at 10 MHz f_{OSC} (minimum)

Interrupts

- 4 interrupt sources with one vector
- One interrupt level

General I/O

- Three I/O ports (Max 18 pins)
- Bit programmable ports

8-bit High-speed PWM

- 8-bit PWM 1-ch (Max: 156 kHz)
- 6-bit base + 2-bit extension

Built-in Reset Circuit

- Low voltage detector for safe Reset

Timer/Counters

- One 8-bit basic timer for watchdog function
- One 8-bit timer/counter with time interval modes

A/D Converter

- Nine analog input pins (MAX)
- 10-bit conversion resolution

Oscillation Frequency

- 1 MHz to 10 MHz external crystal oscillator
- Maximum 10 MHz CPU clock
- Internal RC: 3.2 MHz (typ.), 0.5 MHz (typ.) in V_{DD} = 5 V

Operating Temperature Range

- – 25°C to + 85°C

Operating Voltage Range

- 2.0 V to 5.5 V (LVR disable)
- LVR to 5.5V (LVR enable)

Smart Option

Package Types

- S3C9454B/F9454B:
  - 20-SSOP-225
  - 20-DIP-300A
  - 20-SOP-375
  - 16-SOP-BD300-SG
  - 16-DIP-300A
  - 16-SSOP-BD44

BLOCK DIAGRAM

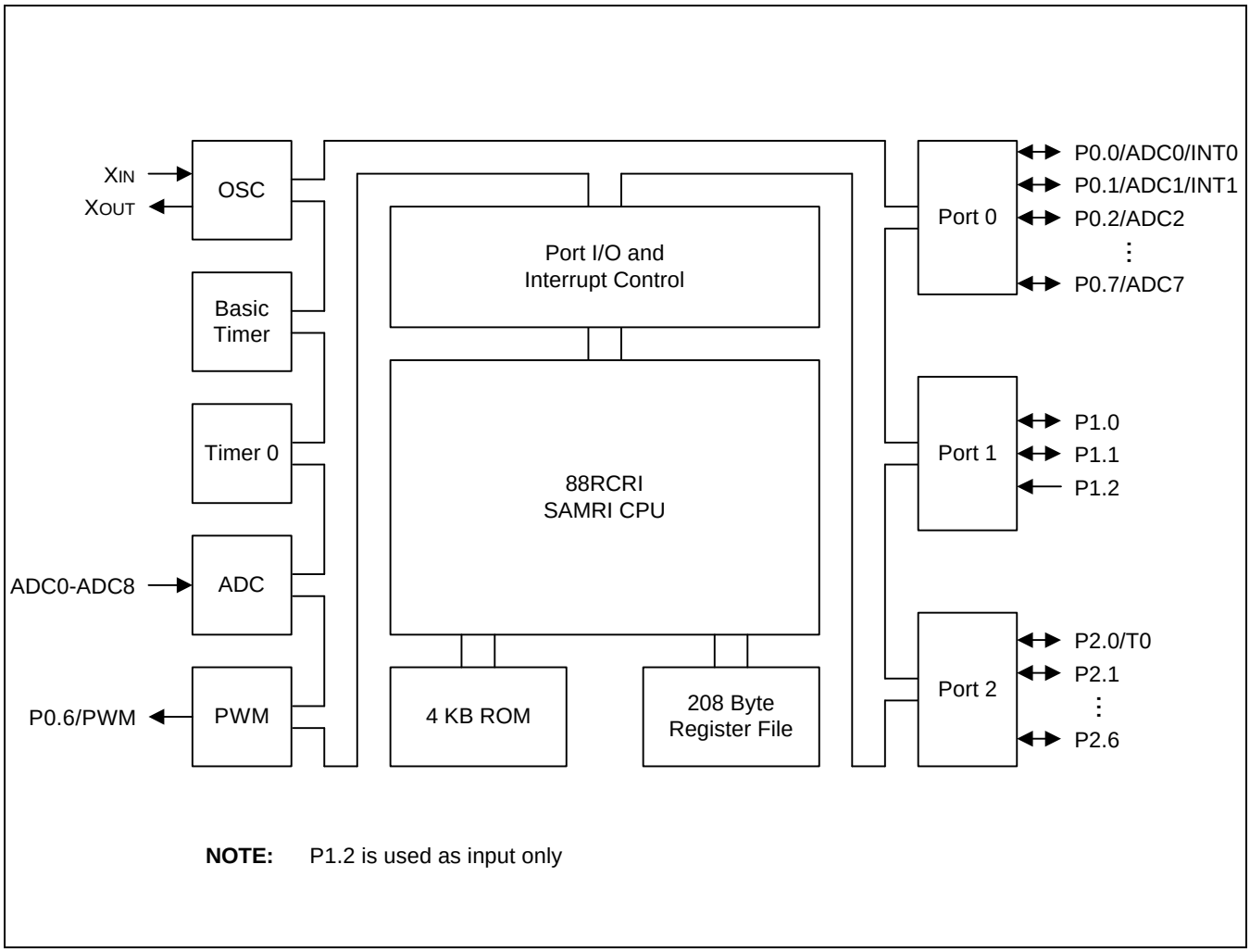


Figure 1-1. Block Diagram

PIN ASSIGNMENTS

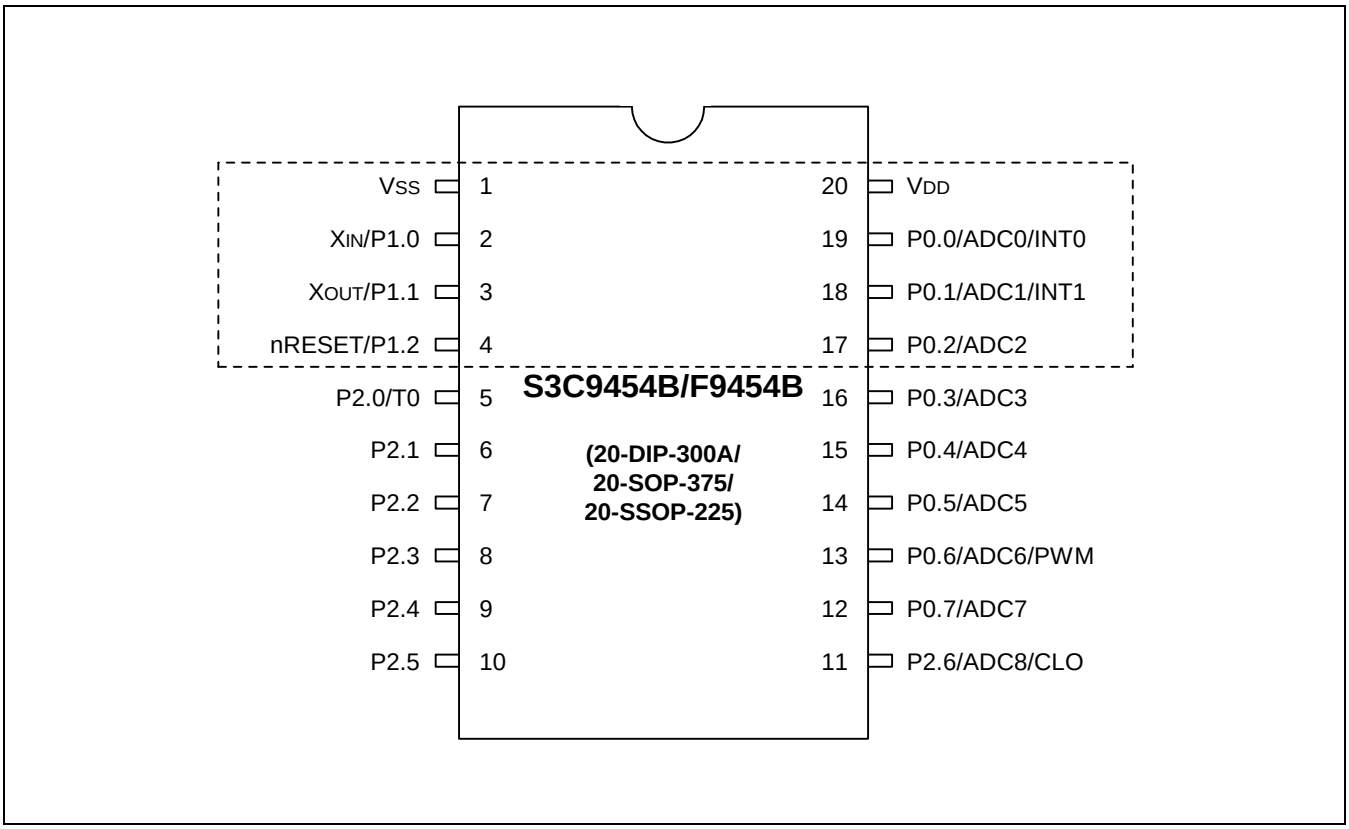


Figure 1-2. Pin Assignment Diagram (20-Pin DIP/SOP/SSOP Package)

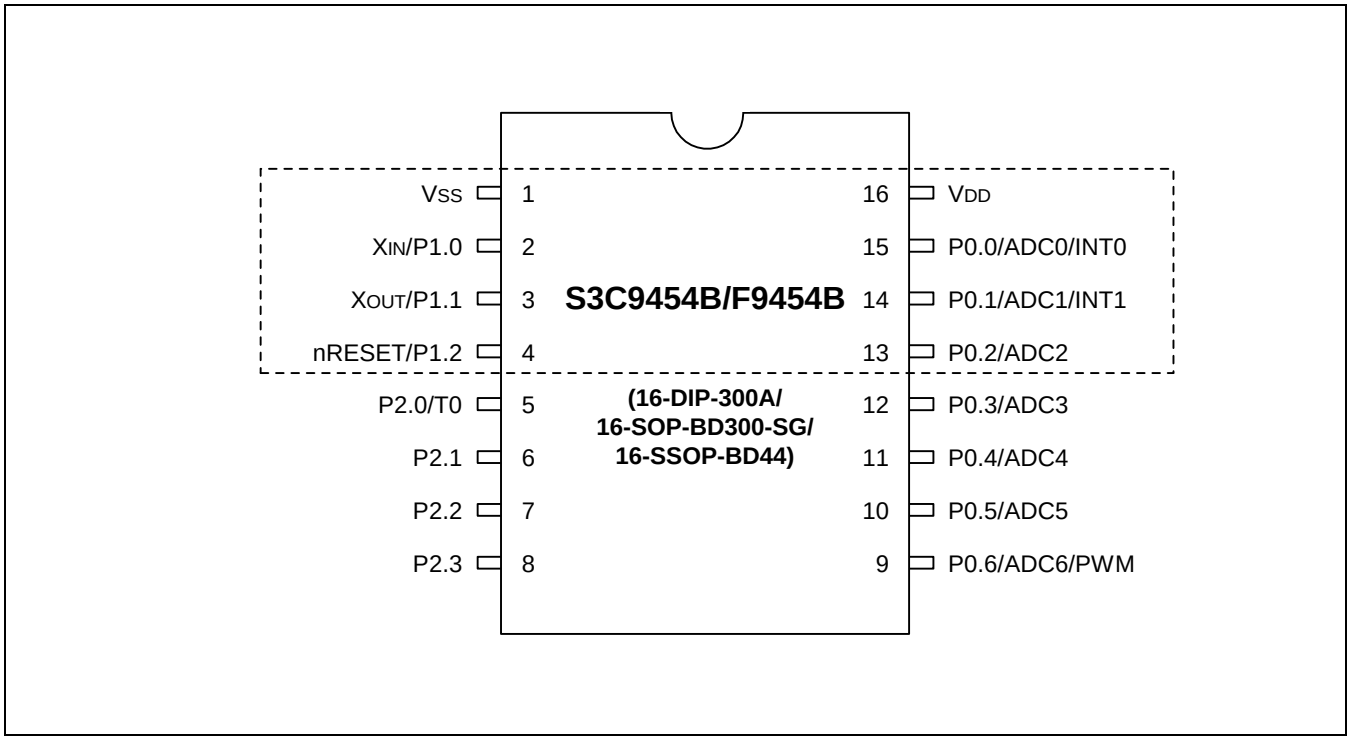


Figure 1-3. Pin Assignment Diagram (16-Pin DIP/SOP/SSOP Package)

PIN DESCRIPTIONS

Table 1-1. S3C9454B/F9454B Pin Descriptions

Pin Name	Input/ Output	Pin Description	Pin Type	Share Pins
P0.0–P0.7	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull output. Pull-up resistors are assignable by software. Port0 pins can also be used as A/D converter input, PWM output or external interrupt input.	E-1	ADC0–ADC7 INT0/INT1 PWM
P1.0–P1.1	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistors or pull-down resistors are assignable by software.	E-2	X_{IN}, X_{OUT}
P1.2	I	Schmitt trigger input port	B	RESET
P2.0–P2.6	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistors are assignable by software.	E E-1	– ADC8/CLO T0
X_{IN}, X_{OUT}	–	Crystal/Ceramic, or RC oscillator signal for system clock.		P1.0–P1.1
nRESET	I	Internal LVR or external RESET	B	P1.2
V_{DD}, V_{SS}	–	Voltage input pin and ground		–
CLO	O	System clock output port	E-1	P2.6
INT0–INT1	I	External interrupt input port	E-1	P0.0, P0.1
PWM	O	8-Bit high speed PWM output	E-1	P0.6
T0	O	Timer0 match output	E-1	P2.0
ADC0–ADC8	I	A/D converter input	E-1 E	P0.0–P0.7 P2.6

PIN CIRCUITS

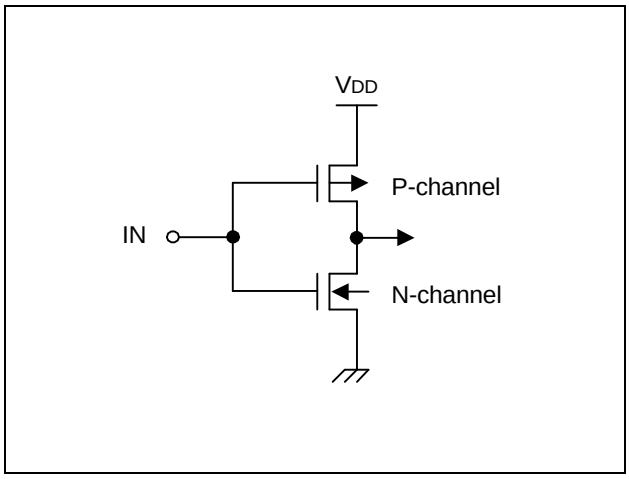


Figure 1-5. Pin Circuit Type A

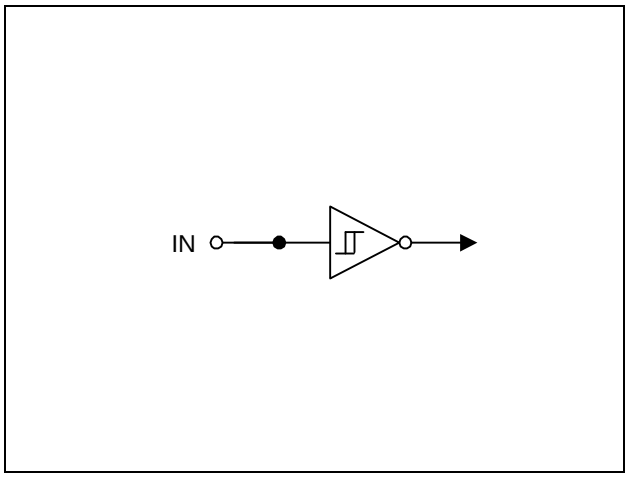


Figure 1-6. Pin Circuit Type B

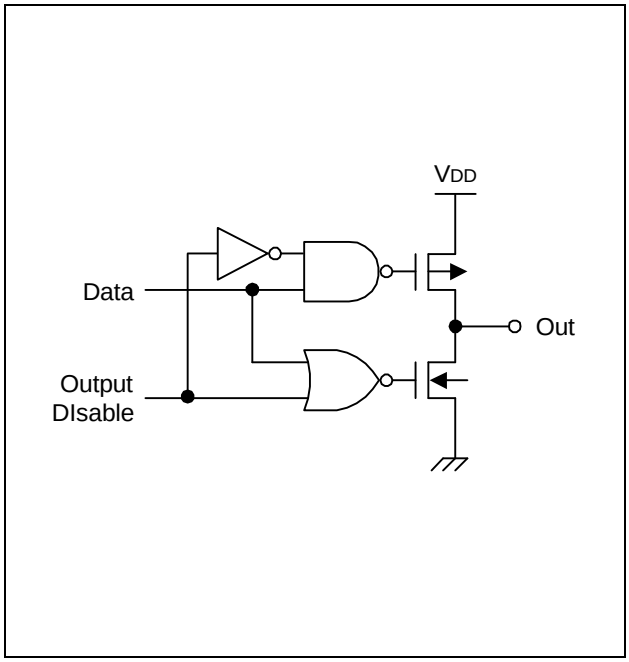


Figure 1-7. Pin Circuit Type C

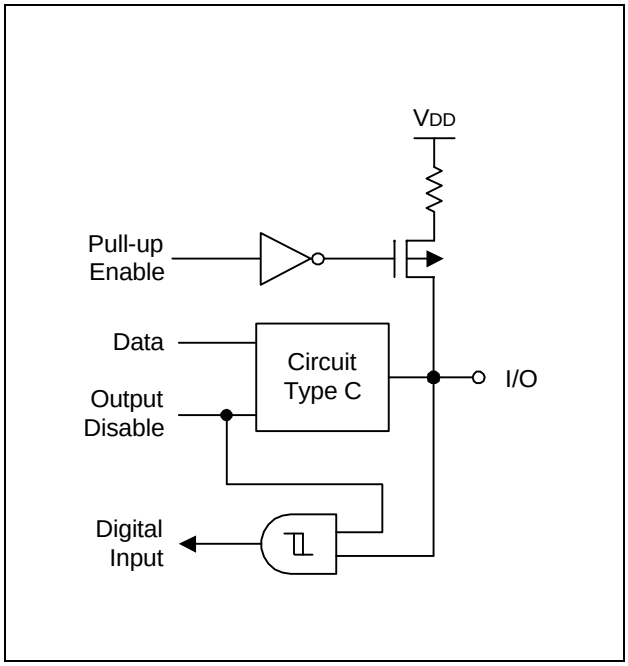


Figure 1-8. Pin Circuit Type D

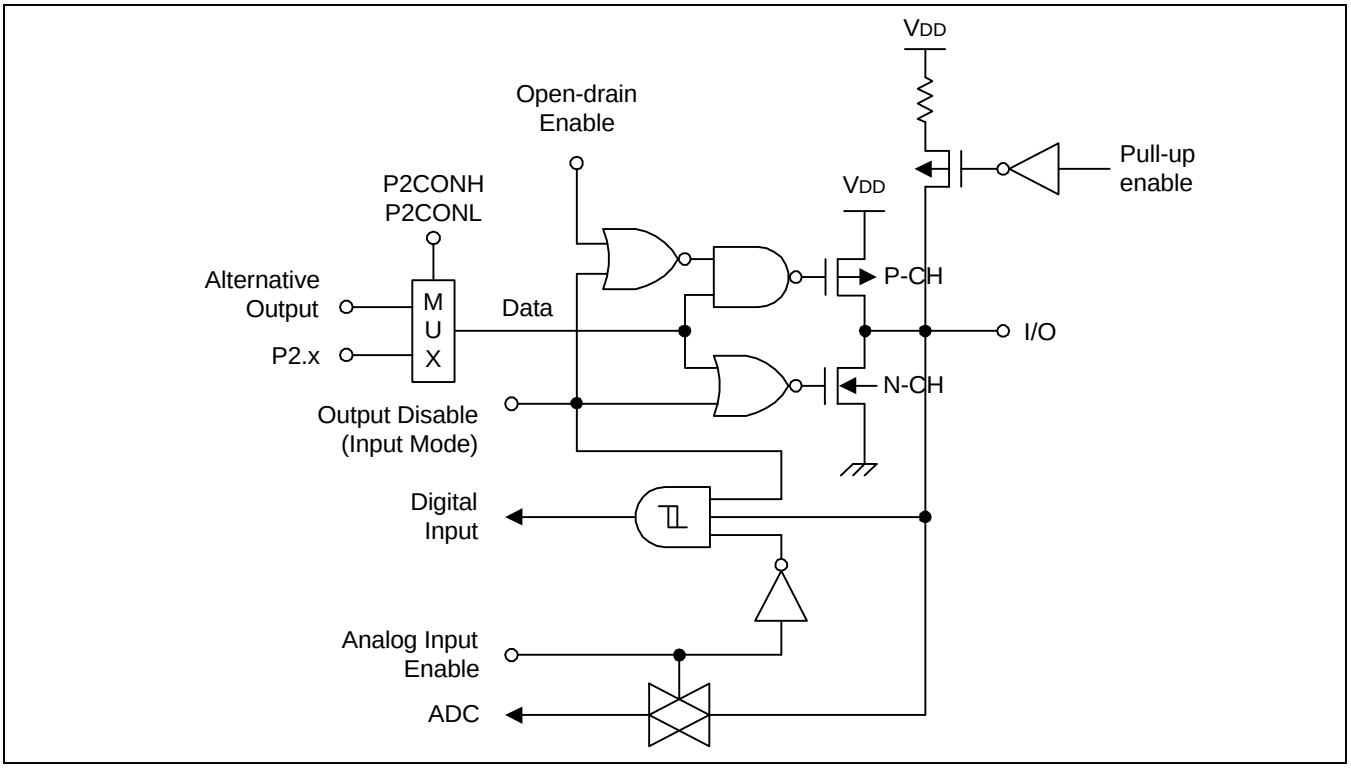


Figure 1-9. Pin Circuit Type E

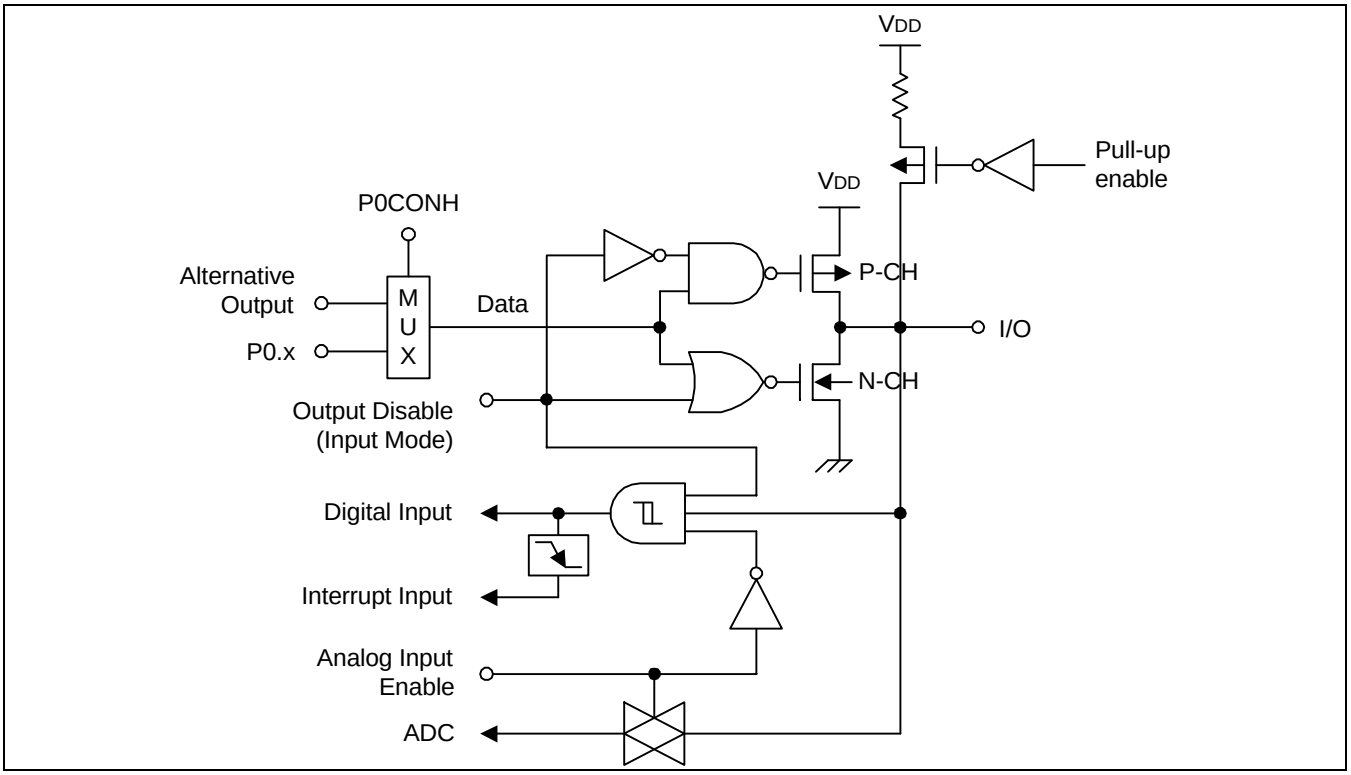


Figure 1-10. Pin Circuit Type E-1

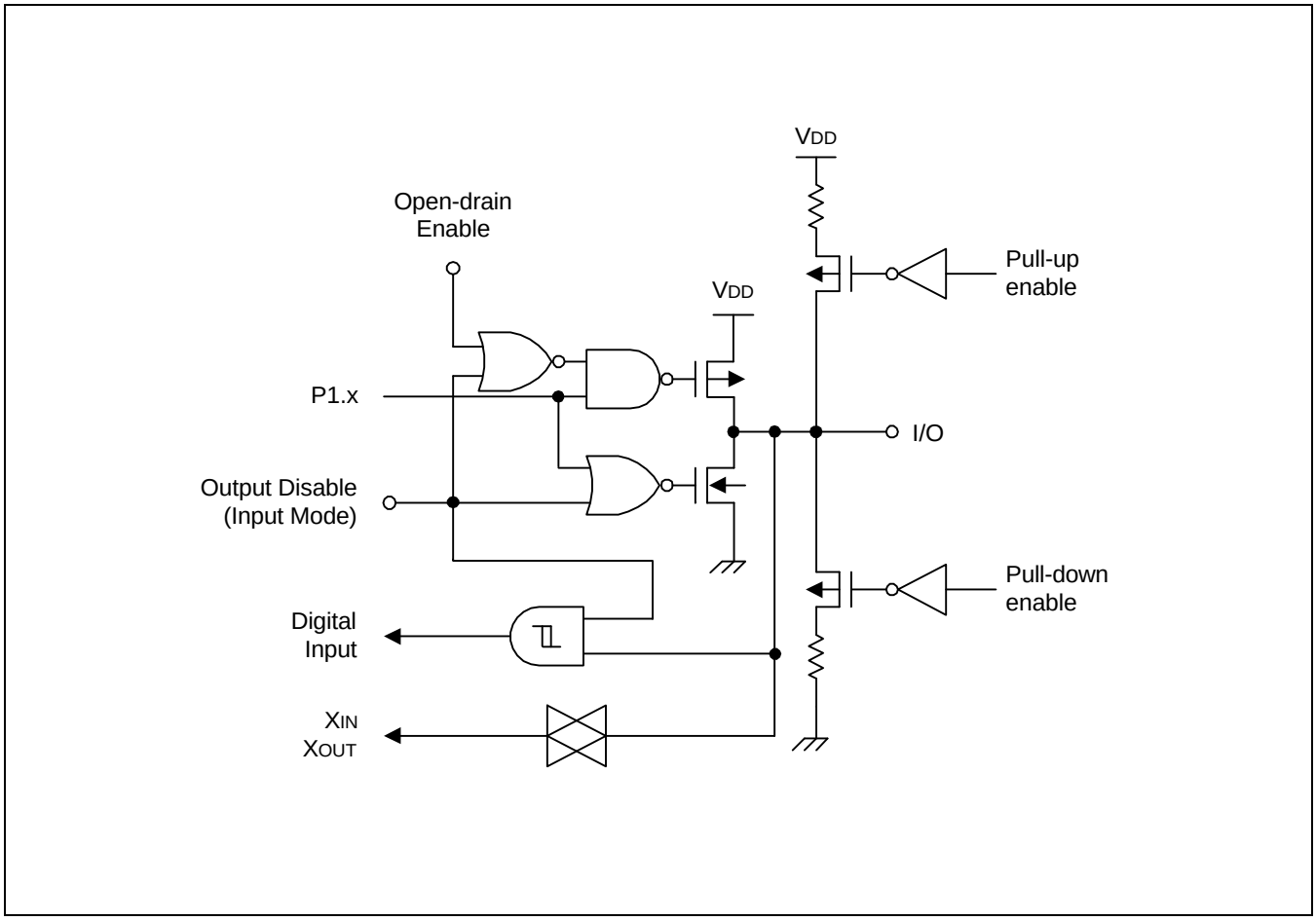


Figure 1-11. Pin Circuit Type E-2

NOTES

2 ADDRESS SPACES

OVERVIEW

The S3C9454B/F9454B microcontroller has two kinds of address space:

— Internal program memory (ROM)

— Internal register file

A 12-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 S3C9454B/F9454B have 4-Kbytes of mask-programmable on-chip program memory: which is configured as the Internal ROM mode, all of the 4-Kbyte internal program memory is used.

The S3C9454B/F9454B microcontroller has 208 general-purpose registers in its internal register file. Twenty-six bytes in the register file are mapped for system and peripheral control functions.

PROGRAM MEMORY (ROM)

Normal Operating Mode

The S3C9454B/F9454B have 4-Kbytes (locations 0H–0FFFH) of internal mask-programmable program memory.

The first 2-bytes of the ROM (0000H–0001H) are interrupt vector address.

Unused locations (0002H–00FFH except 3CH, 3DH, 3EH, 3FH) can be used as normal program memory. 3CH, 3DH, 3EH, 3FH is used smart option ROM cell.

The program Reset address in the ROM is 0100H.

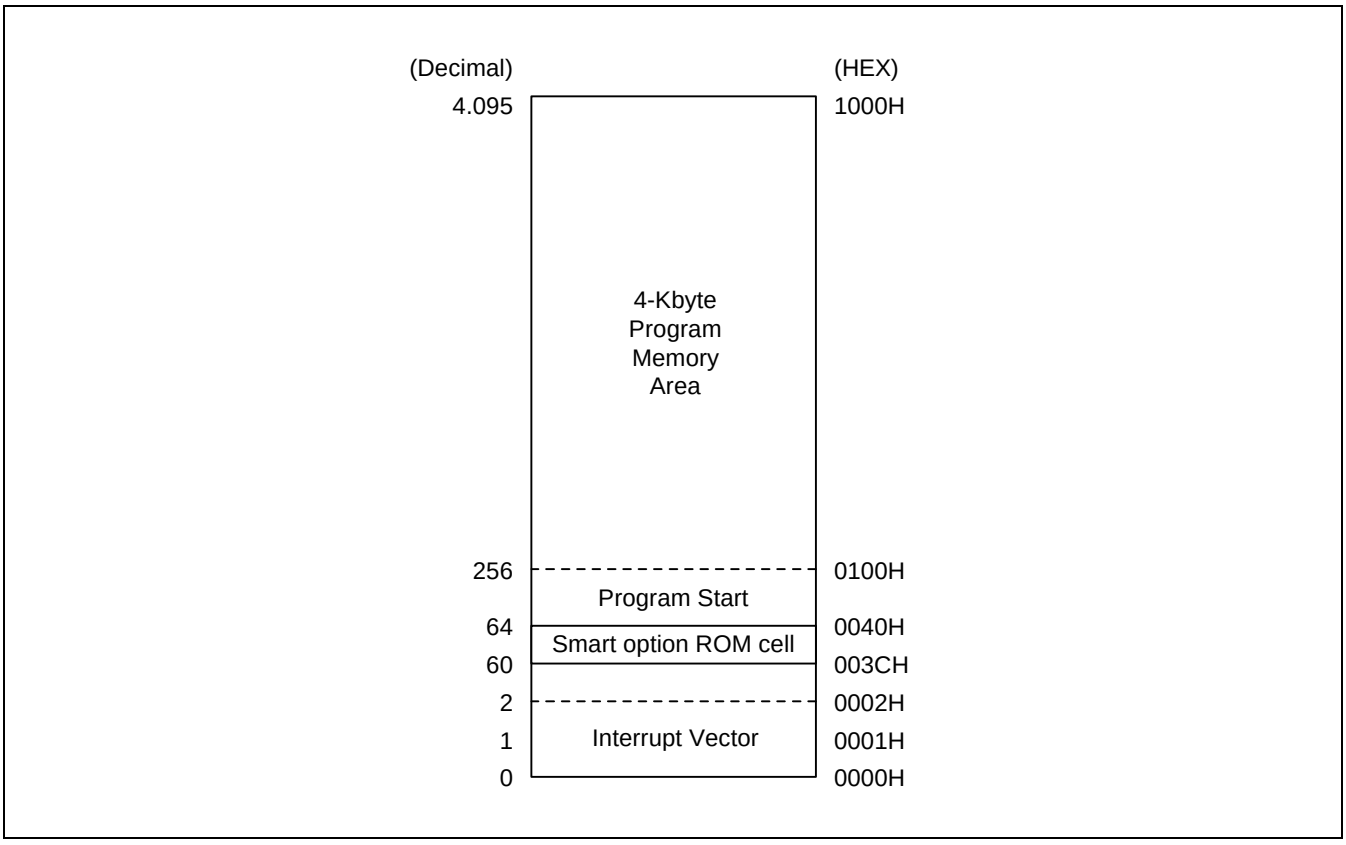
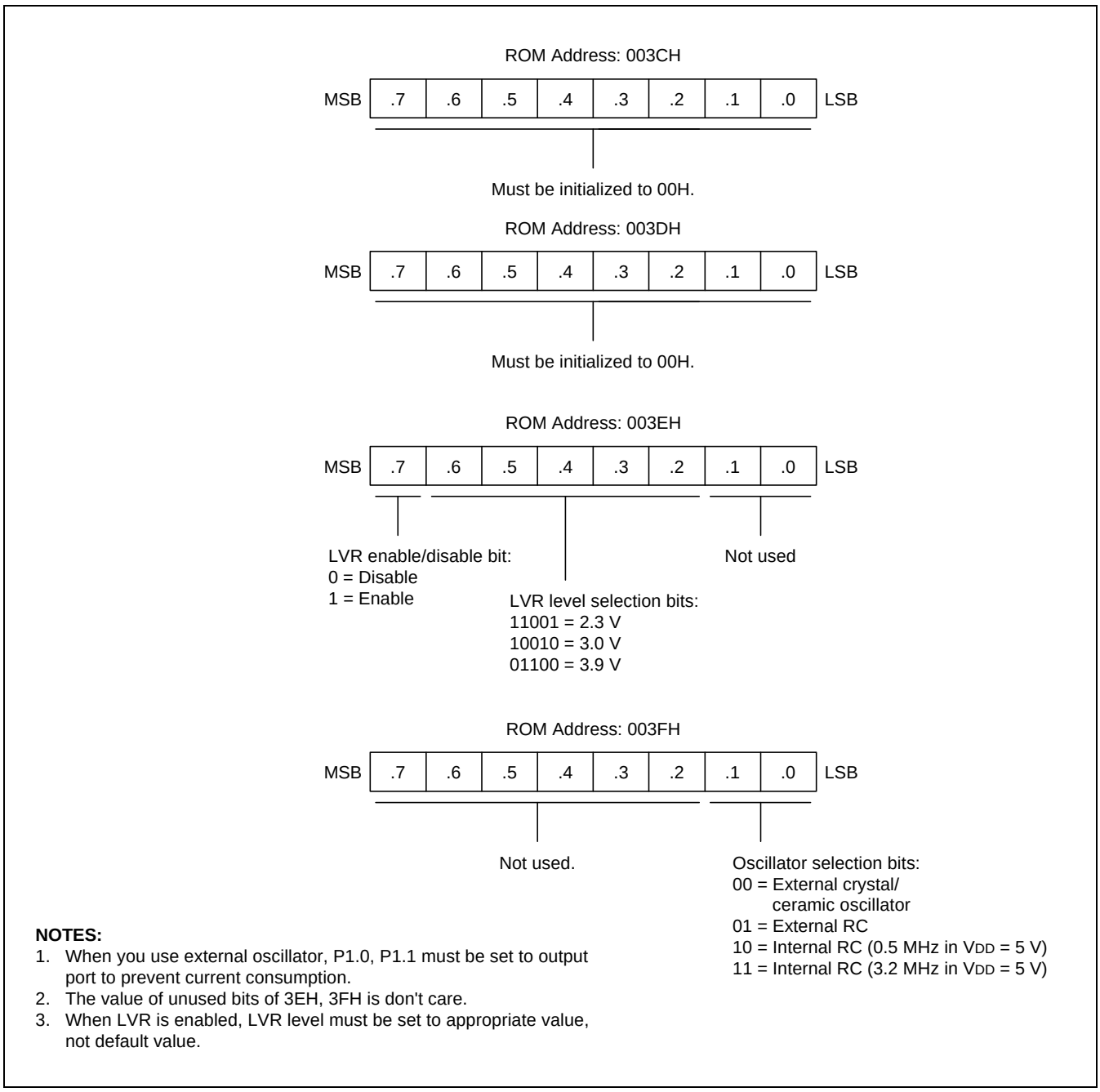


Figure 2-1. Program Memory Address Space

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 S3C9454B/F9454B only use 003EH, 003FH. Not used ROM address 003CH, 003DH should be initialized to be initialized to 00H. The default value of ROM is FFH (LVR enable, internal RC oscillator).



NOTES:
1. When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption.
2. The value of unused bits of 3EH, 3FH is don't care.
3. When LVR is enabled, LVR level must be set to appropriate value, not default value.

Figure 2-2. Smart Option

☞ PROGRAMMING TIP — Smart Option Setting

```
;           << Interrupt Vector Address >>

            ORG     0000H
            Vector  00H, INT_9454       ;  S3C9454B/F9454B has only one interrupt vector

;           << Smart Option Setting >>

            ORG     003CH
            DB      00H                 ;  003CH, must be initialized to 0.
            DB      00H                 ;  003DH, must be initialized to 0.
            DB      0E7H                ;  003EH, enable LVR (2.3 V)
            DB      03H                 ;  003FH, Internal RC (3.2 MHz in VDD = 5 V)

;           << Reset >>

            ORG     0100H
            RESET:  DI
                    •
                    •
                    •
```

REGISTER ARCHITECTURE

The upper 64-bytes of the S3C9454B/F9454B's 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*. 234 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 the general purpose register space (00H–BFH: page0). This register file expansion is not implemented in the S3C9454B/F9454B, 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
CPU and system control registers	11
Peripheral, I/O, and clock control and data registers	15
General-purpose registers (including the 16-bit common working register area)	208
Total Addressable Bytes	234

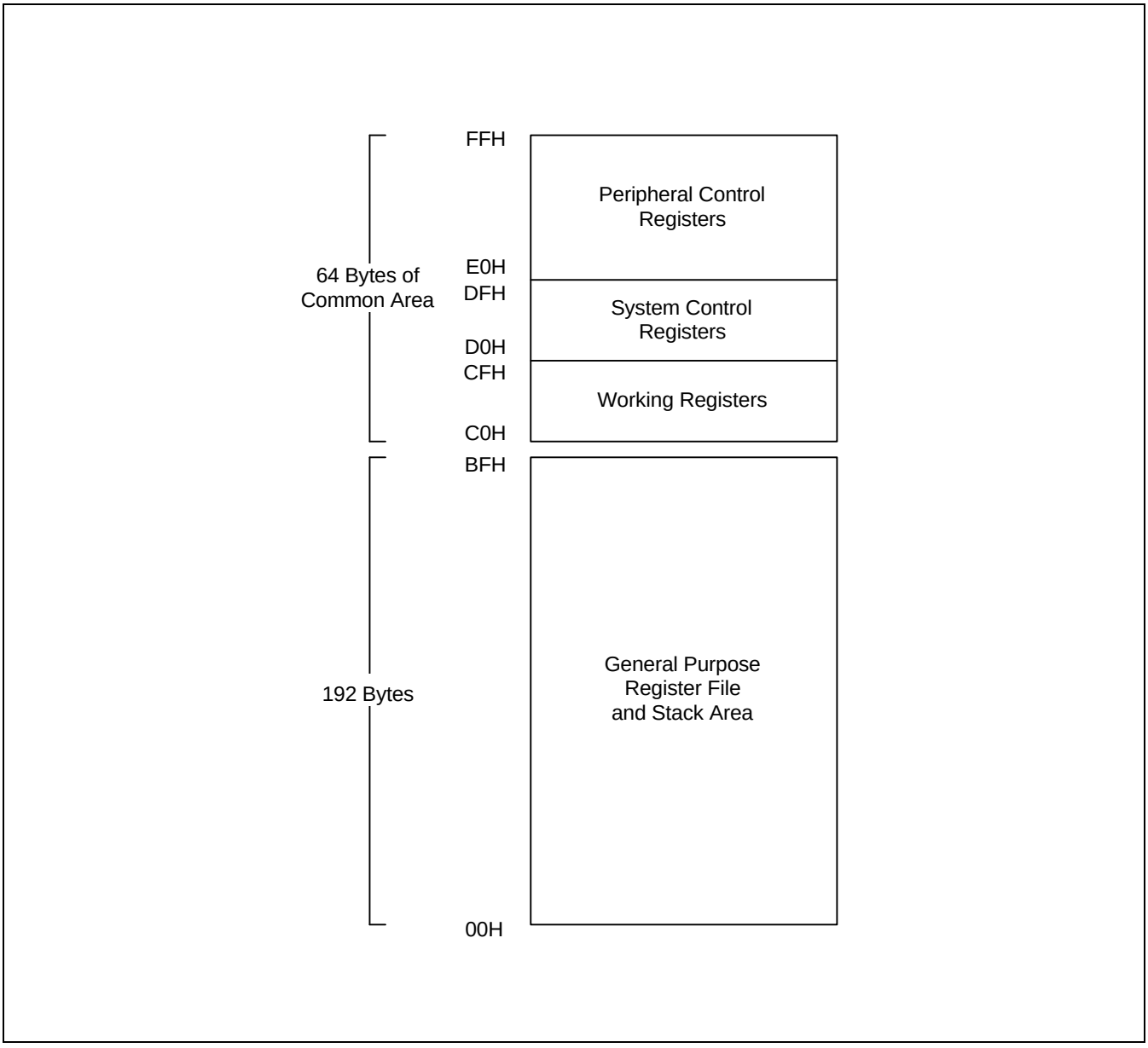


Figure 2-3. Internal Register File Organization

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.

This16-byte address range is called common area. That is, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages. However, because the S3C9454B/F9454B uses only page 0, you can use the common area for any internal data operation.

The Register (R) addressing mode can be used to access this area

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.

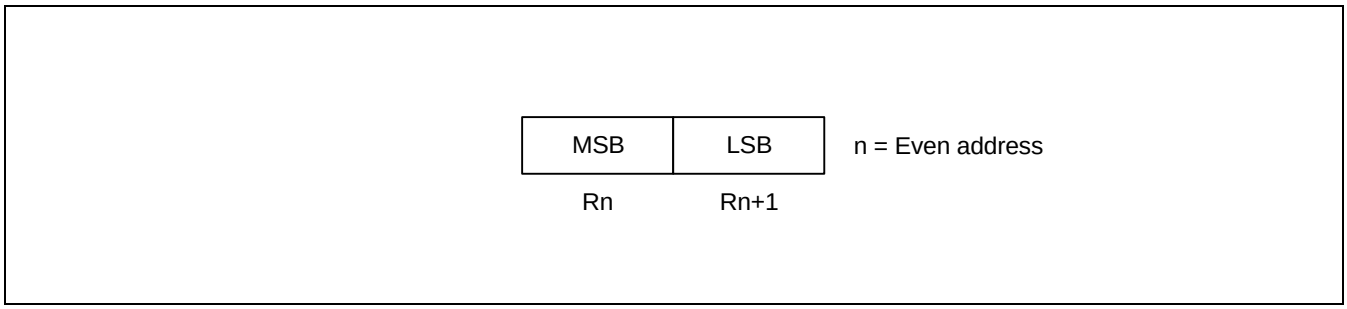


Figure 2-4. 16-Bit Register Pairs

☞ PROGRAMMING TIP — Addressing the Common Working Register Area

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

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

Use working register addressing instead:

LD R2,40H ; R2 (C2H) ← the value in location 40H

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

Use working register addressing instead:

ADD R3,#45H ; R3 (C3H) ← R3 + 45H

SYSTEM STACK

S3C9-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 S3C9454B/F9454B 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 register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address is always decremented *before* a push operation and incremented *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.

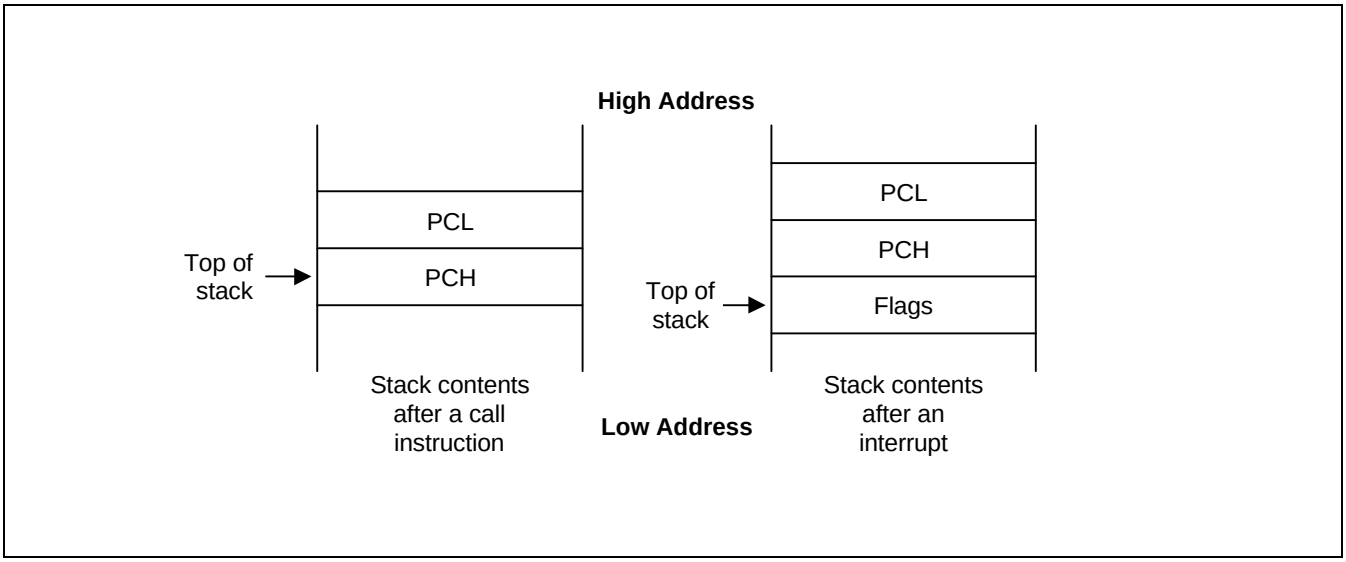


Figure 2-5. Stack Operations

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 S3C9454B/F9454B, 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
```

NOTES

3 ADDRESSING MODES

OVERVIEW

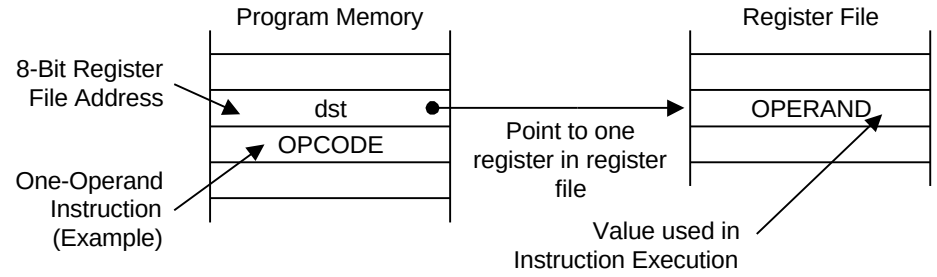
Instructions that are stored in program memory are fetched for execution using the program counter. Instructions indicate the operation to be performed and the data to be operated on. *Addressing mode* is the method used to determine the location of the data operand. The operands specified in SAM88RCRI instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

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

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

REGISTER ADDRESSING MODE (R)

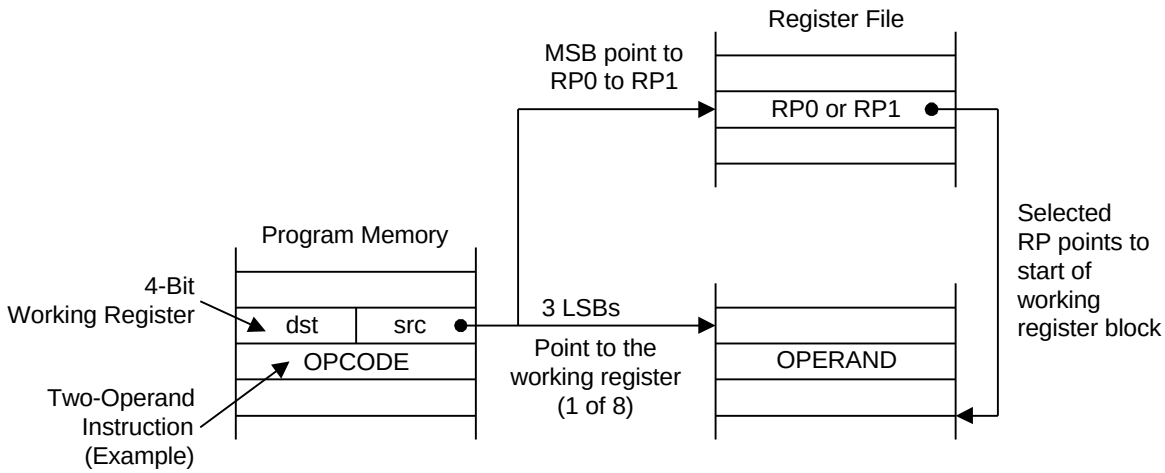
In Register addressing mode, the operand is the content of a specified register (see Figure 3-1). Working register addressing differs from Register addressing because it uses an 16-byte working register space in the register file and an 4-bit register within that space (see Figure 3-2).



Sample Instruction:

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

Figure 3-1. Register Addressing



Sample Instruction:

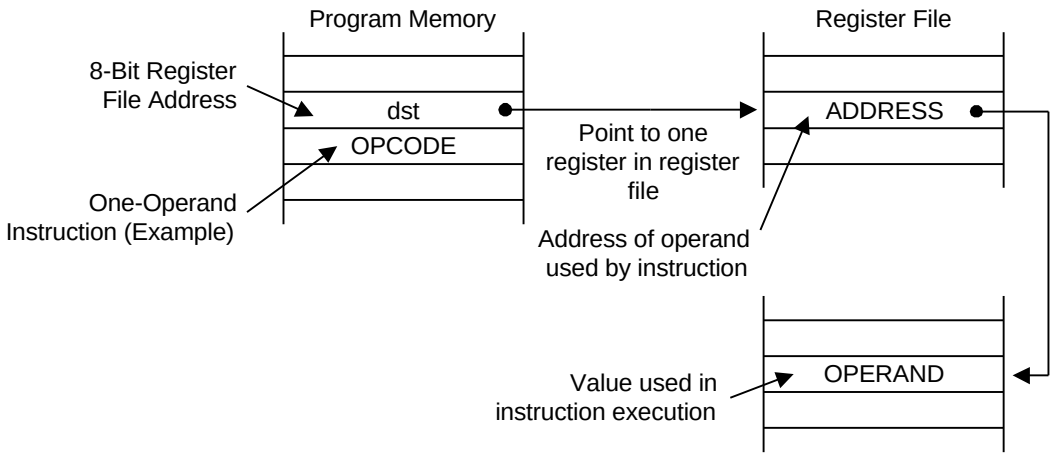
```
ADD     R1, R2          ;   Where R1 and R2 are registers in the currently selected
                            working register area.
```

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 ddress
```

Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

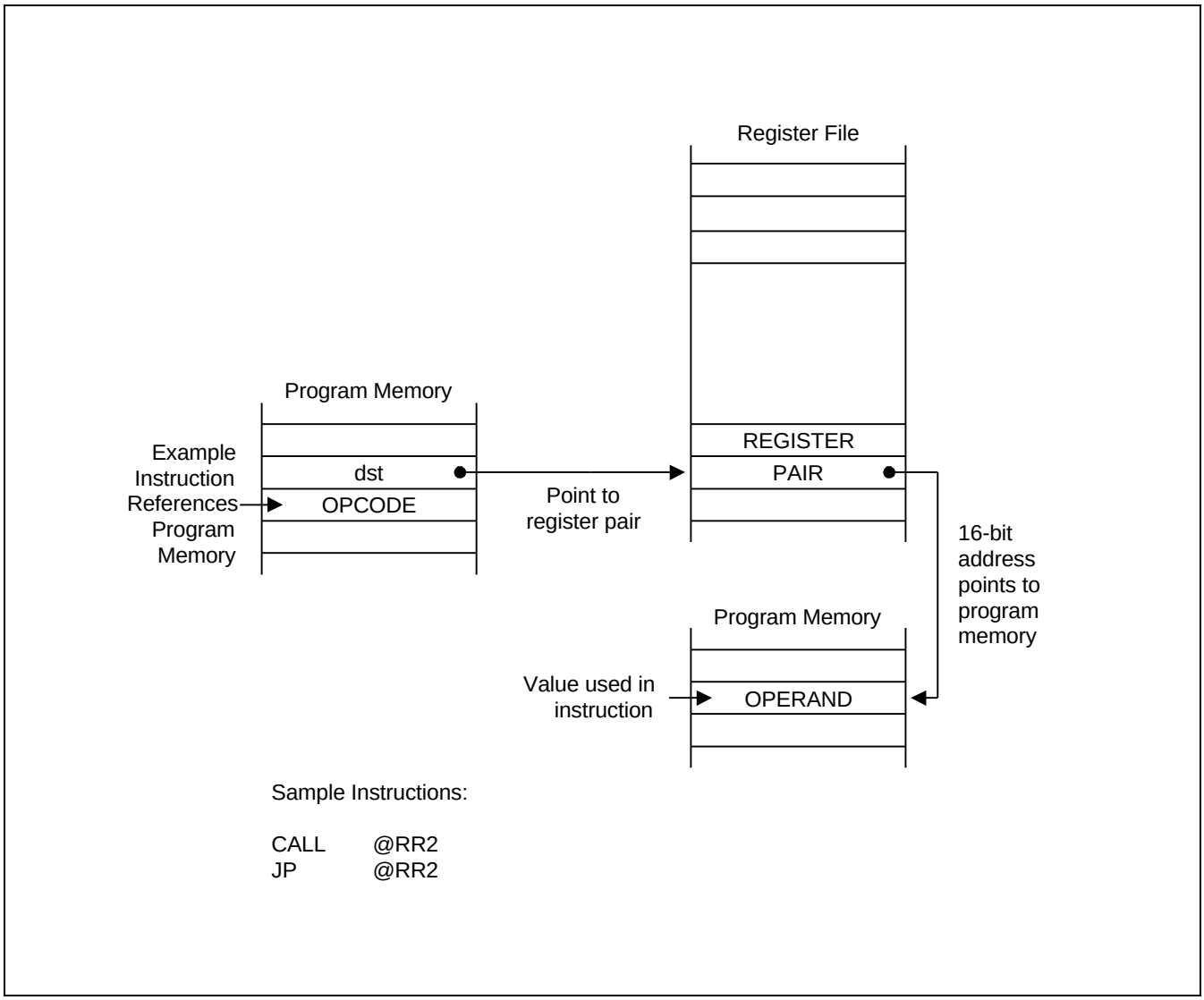


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

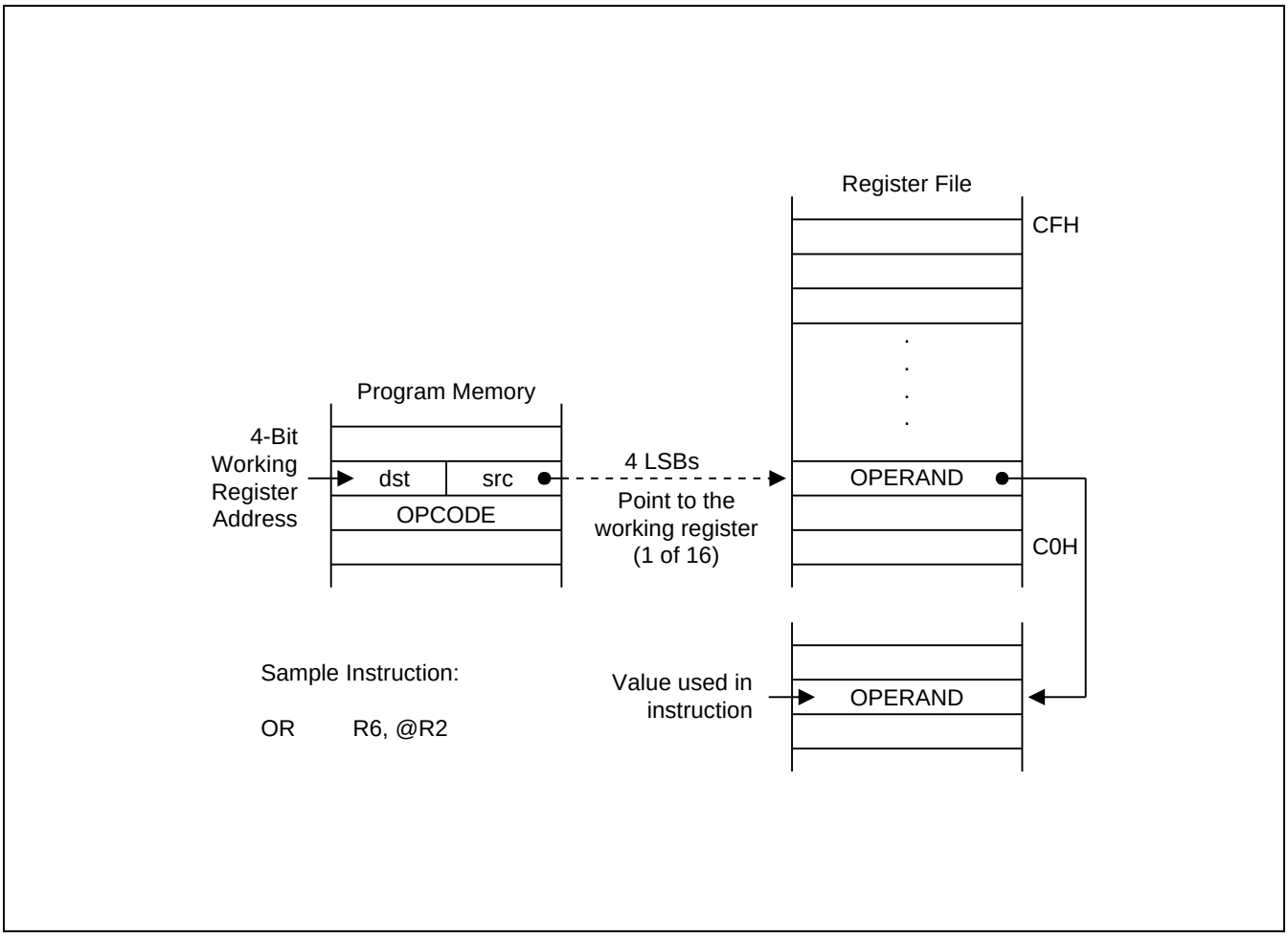


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)

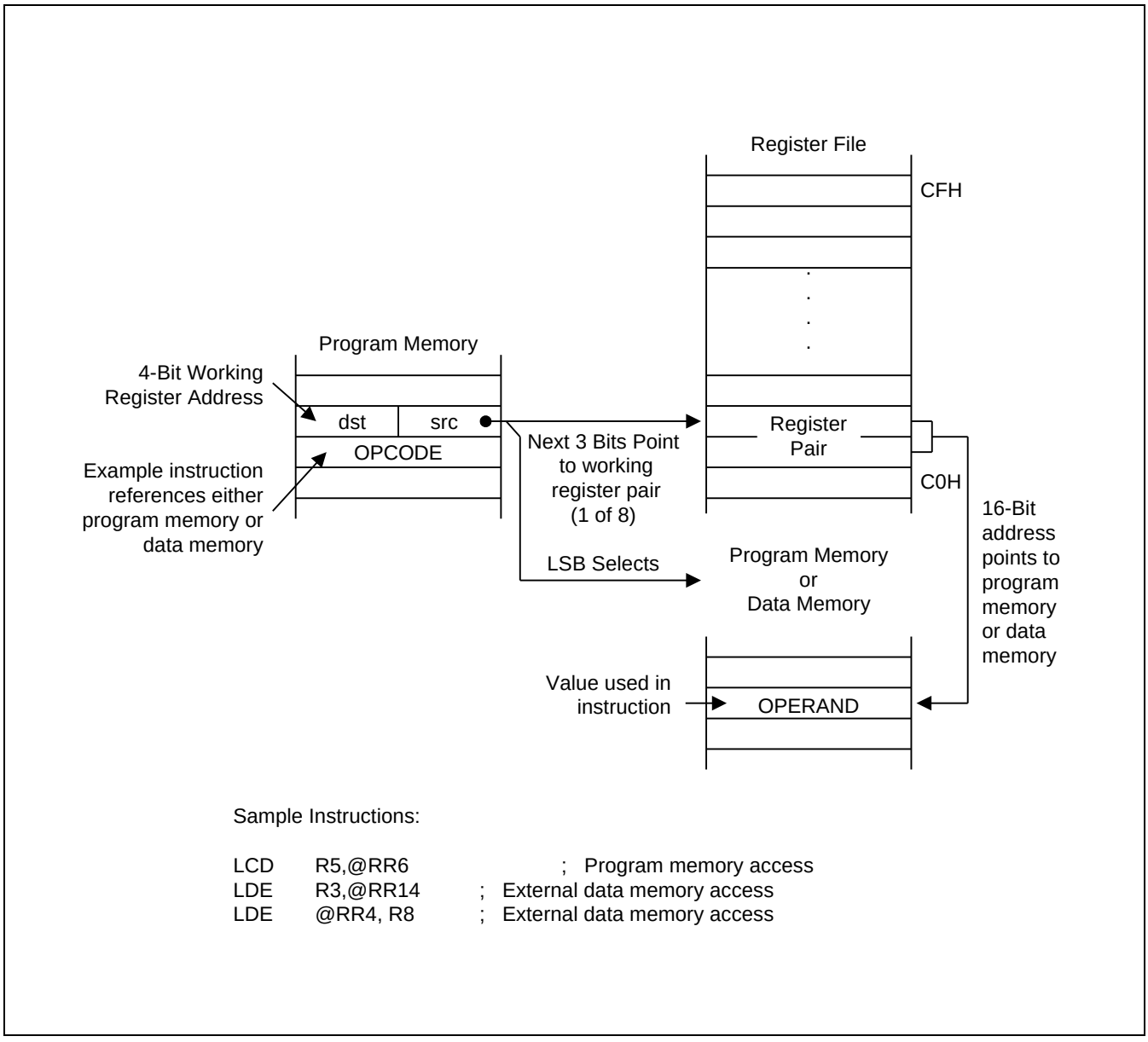


Figure 3-6. Indirect Working Register Addressing to Program or Data Memory

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 the 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, external program memory, and for external data memory, when implemented.

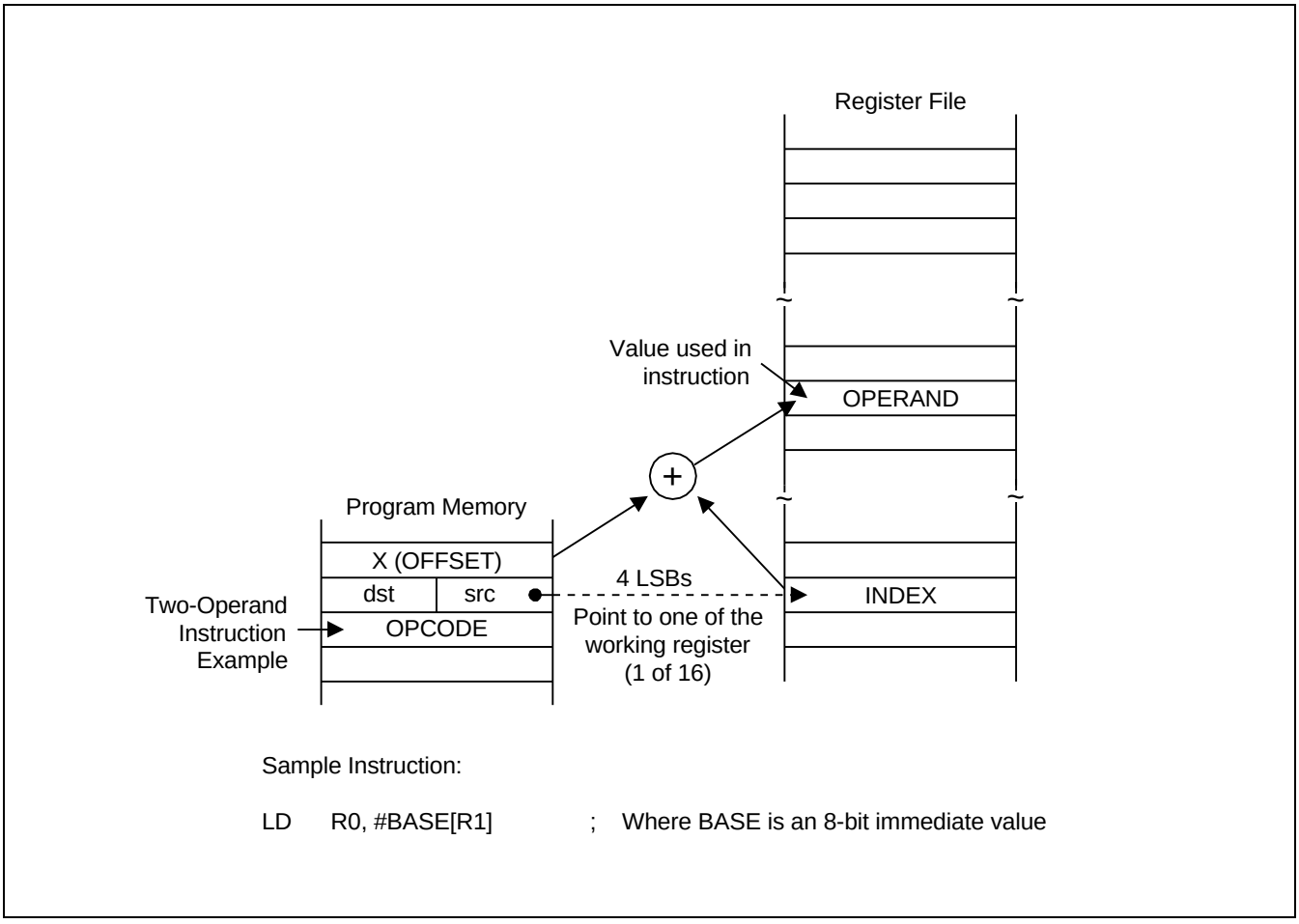
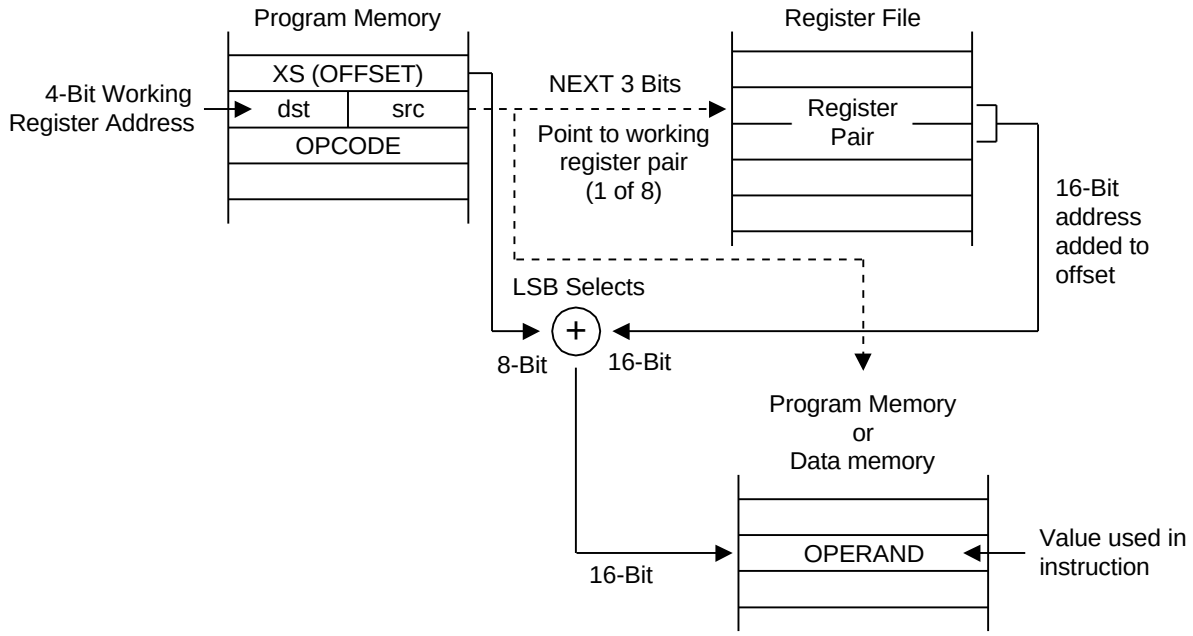


Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)

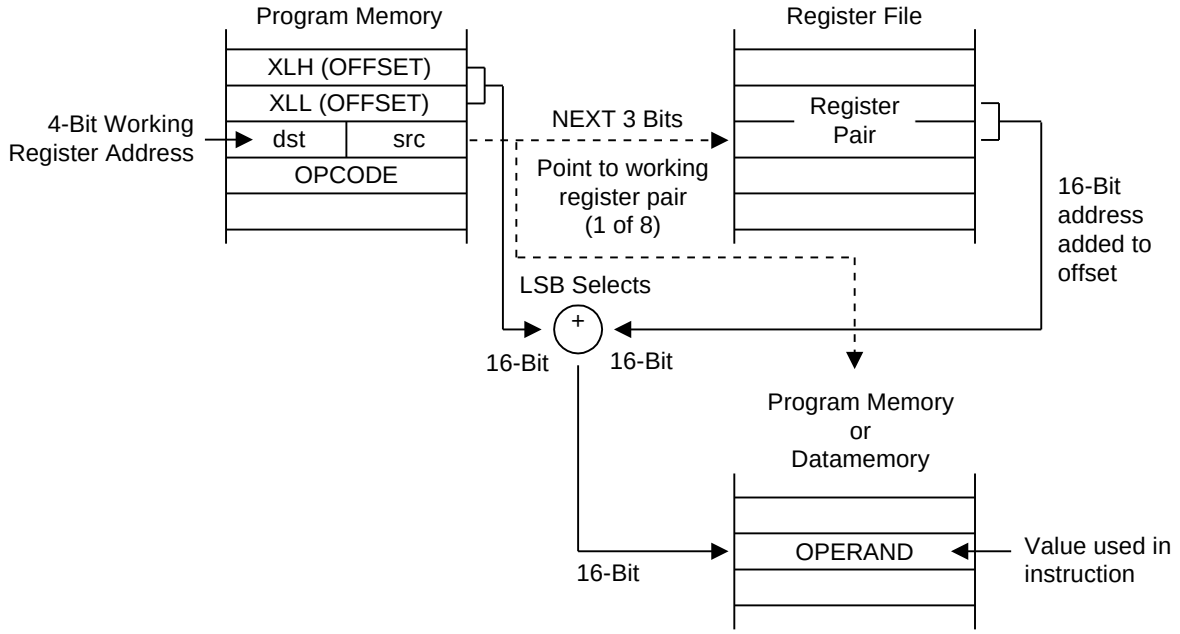


Sample Instructions:

```
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:

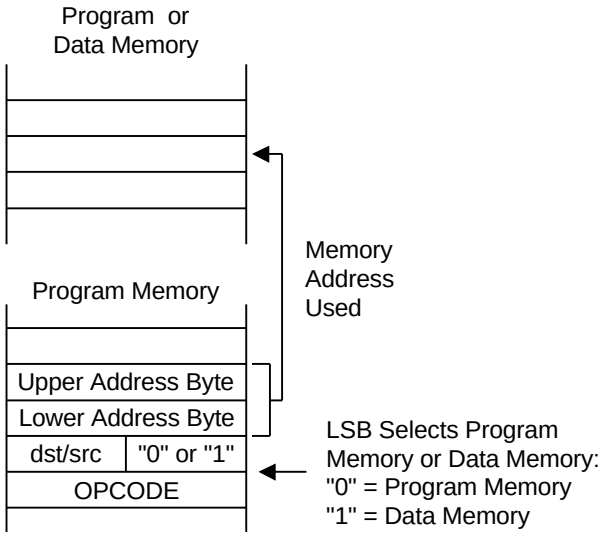
```
LDC     R4, #1000H[RR2]     ; The values in the program address (RR2 + #1000H)
                            ; are loaded into register R4.
LDE     R4, #1000H[RR2]     ; Identical operation to LDC example, except that
                            ; external program memory is accessed.
```

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

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.

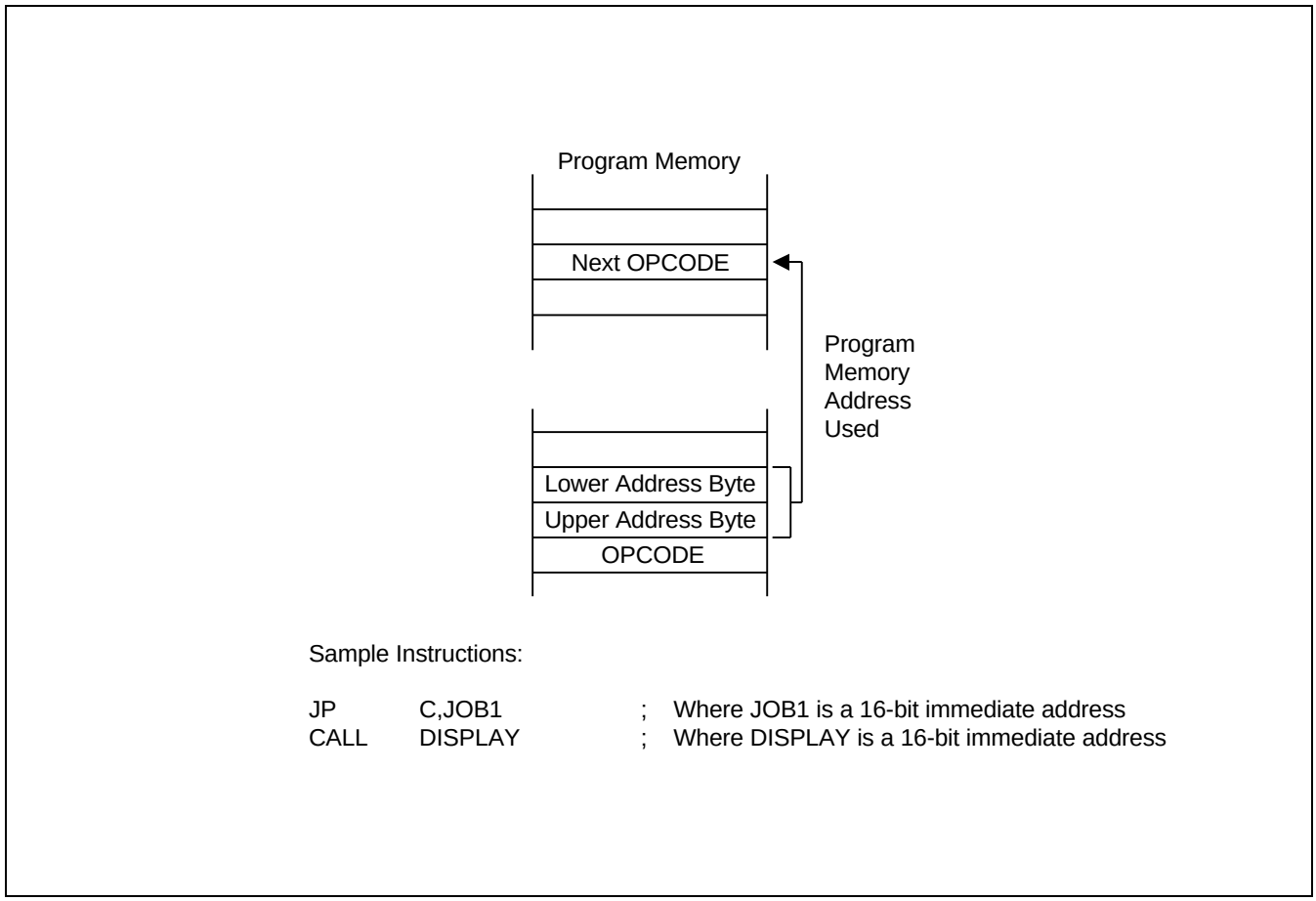


Sample Instructions:

```
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:

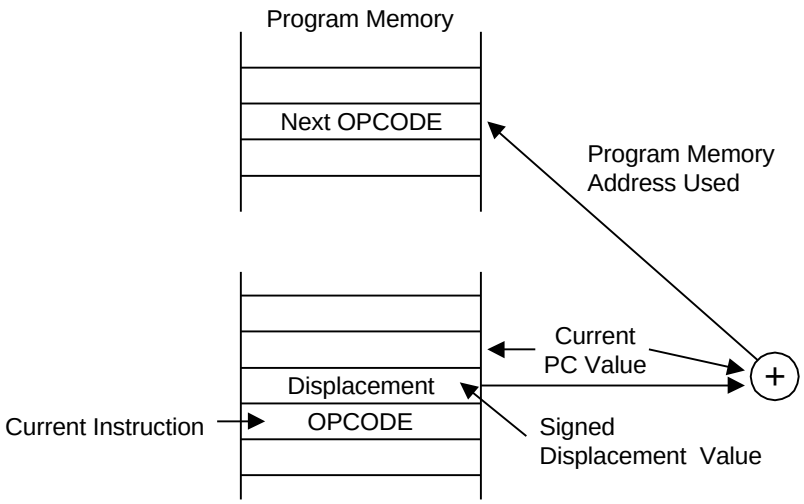
```
JP      C,JOB1      ;  Where JOB1 is a 16-bit immediate address
CALL    DISPLAY     ;  Where DISPLAY is a 16-bit immediate address
```

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

RELATIVE ADDRESS MODE (RA)

In Relative Address (RA) mode, a two's-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.



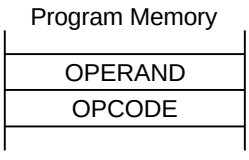
Sample Instructions:

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

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.



(The Operand value is in the instruction)

Sample Instruction: LD R0,#0AAH

Figure 3-13. Immediate Addressing

4 CONTROL REGISTERS

OVERVIEW

In this section, detailed descriptions of the S3C9454B/F9454B control registers are presented in an easy-to-read format. These descriptions will help familiarize you with the mapped locations in the register file. You can also use them as a quick-reference source when writing application programs.

System and peripheral registers are summarized in Table 4-1. Figure 4-1 illustrates the important features of the standard register description format.

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

Table 4-1. System and Peripheral Control Registers

Register name	Mnemonic	Address & Location		RESET value (Bit)							
		Address	R/W	7	6	5	4	3	2	1	0
Timer 0 counter register	T0CNT	D0H	R	0	0	0	0	0	0	0	0
Timer 0 data register	T0DATA	D1H	R/W	1	1	1	1	1	1	1	1
Timer 0 control register	T0CON	D2H	R/W	0	0	–	–	0	–	0	0
Location D3H is not mapped											
Clock control register	CLKCON	D4H	R/W	0	–	–	0	0	–	–	–
System flags register	FLAGS	D5H	R/W	x	x	x	x	–	–	–	–
Locations D6H–D8H are not mapped											
Stack pointer register	SP	D9H	R/W	x	x	x	x	x	x	x	x
Location DAH is not mapped											
MDS special register	MDSREG	DBH	R/W	0	0	0	0	0	0	0	0
Basic timer control register	BTCON	DCH	R/W	0	0	0	0	0	0	0	0
Basic timer counter	BTCNT	DDH	R	0	0	0	0	0	0	0	0
Test mode control register	FTSTCON	DEH	W	–	–	0	0	0	0	0	0
System mode register	SYM	DFH	R/W	–	–	–	–	–	0	0	0

NOTES:

1. – : Not mapped or not used, x: Undefined
2. The factory test mode register, FTSTCON, is for factory use only. Its value should always be '00H' during the normal operation.

Table 4-1. System and Peripheral Control Registers (Continued)

Register Name	Mnemonic	Address Hex	R/W	Bit Values After RESET							
				7	6	5	4	3	2	1	0
Port 0 data register	P0	E0H	R/W	0	0	0	0	0	0	0	0
Port 1 data register	P1	E1H	R/W	–	–	–	–	–	0	0	0
Port 2 data register	P2	E2H	R/W	–	0	0	0	0	0	0	0
Locations E3H–E5H are not mapped											
Port 0 control register (High byte)	P0CONH	E6H	R/W	0	0	0	0	0	0	0	0
Port 0 control register	P0CONL	E7H	R/W	0	0	0	0	0	0	0	0
Port 0 interrupt pending register	P0PND	E8H	R/W	–	–	–	–	0	0	0	0
Port 1 control register	P1CON	E9H	R/W	0	0	–	–	0	0	0	0
Port 2 control register (High byte)	P2CONH	EAH	R/W	–	0	0	0	0	0	0	0
Port 2 control register (Low byte)	P2CONL	EBH	R/W	0	0	0	0	0	0	0	0
Locations ECH–F1H are not mapped											
PWM data register	PWMDATA	F2H	R/W	0	0	0	0	0	0	0	0
PWM control register	PWMCON	F3H	R/W	0	0	–	0	0	0	0	0
STOP .control register	STOPCON	F4H	R/W	0	0	0	0	0	0	0	0
Locations F5H–F6H are not mapped											
A/D control register	ADCON	F7H	R/W	0	0	0	0	0	0	0	0
A/D converter data register (High)	ADDATAH	F8H	R	x	x	x	x	x	x	x	x
A/D converter data register (Low)	ADDATAL	F9H	R	0	0	0	0	0	0	x	x
Locations FAH–FFH are not mapped											

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

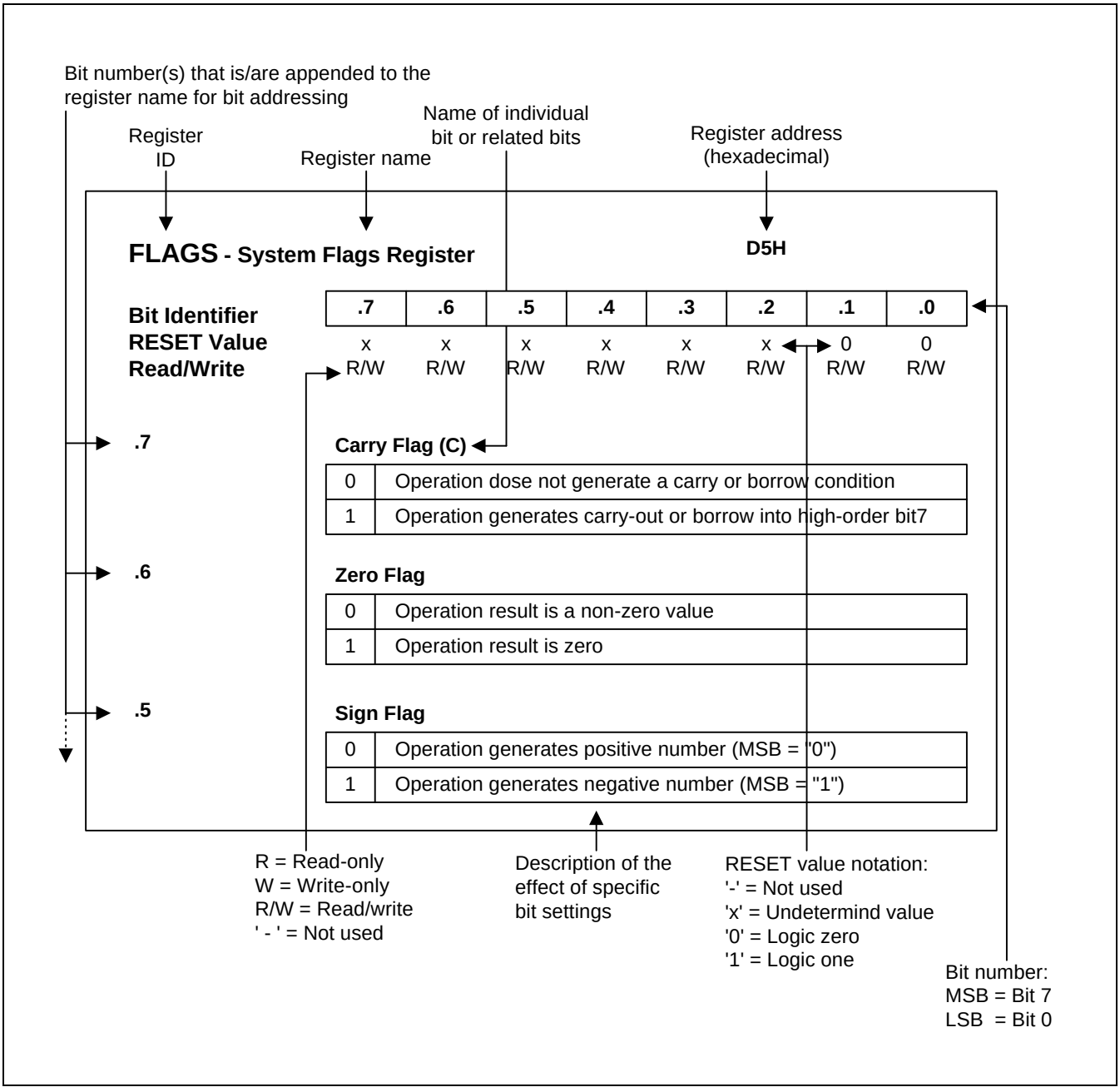


Figure 4-1. Register Description Format

ADCON — A/D Converter Control Register F7H

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 Converter Input Pin Selection Bits

0	0	0	0	ADC0 (P0.0)
0	0	0	1	ADC1 (P0.1)
0	0	1	0	ADC2 (P0.2)
0	0	1	1	ADC3 (P0.3)
0	1	0	0	ADC4 (P0.4)
0	1	0	1	ADC5 (P0.5)
0	1	1	0	ADC6 (P0.6)
0	1	1	1	ADC7 (P0.7)
1	0	0	0	ADC8 (P2.6)
1	0	0	1	Connected with GND internally
1	0	1	0	Connected with GND internally
1	0	1	1	Connected with GND internally
1	1	0	0	Connected with GND internally
1	1	0	1	Connected with GND internally
1	1	1	0	Connected with GND internally
1	1	1	1	Connected with GND internally

.3 End-of-Conversion Status Bit

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

.2–.1 Clock Source Selection Bit (note)

0	0	$f_{OSC}/16$ ($f_{OSC} \leq 10$ MHz)
0	1	$f_{OSC}/8$ ($f_{OSC} \leq 10$ MHz)
1	0	$f_{OSC}/4$ ($f_{OSC} \leq 10$ MHz)
1	1	$f_{OSC}/1$ ($f_{OSC} \leq 4$ MHz)

.0 Conversion Start Bit

0	No meaning
1	A/D conversion start

NOTE: Maximum ADC clock input = 4 MHz.

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 — 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 S3C9454B/F9454B

.4–.3 Divided by Selection Bits for CPU Clock frequency

0	0	Divide by 16 (f_{OSC}/16)
0	1	Divide by 8 (f_{OSC}/8)
1	0	Divide by 2 (f_{OSC}/2)
1	1	Non-divided clock (f_{OSC})

.2–.0 Not used for S3C9454B/F9454B

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 $\leq +127$ or ≥ -128
1	Operation result is $> +127$ or < -128

.3–.0 Not used for S3C9454B/F9454B

P0CONH — Port 0 Control Register (High Byte) E6H

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 Port 0, P0.7/INT7 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	A/D converter input (ADC7); Schmitt trigger input off

.5–.4 Port 0, P0.6/ADC6/PWM Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Alternative function (PWM output)
1	0	Push-pull output
1	1	A/D converter input (ADC6); Schmitt trigger input off

.3–.2 Port 0, P0.5/ADC5 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	A/D converter input (ADC5); Schmitt trigger input off

.1–.0 Port 0, P0.4/ADC4 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	A/D converter input (ADC4); Schmitt trigger input off

P0CONL — Port 0 Control Register (Low Byte) E7H

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 **Port 0, P0.3/INT3 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input; pull-up enable
1	0	Push-pull output
1	1	A/D converter input (ADC3); Schmitt trigger input off

.5–.4 **Port 0, P0.2/ADC2 Configuration Bits**

0	0	Schmitt trigger input
0	1	Schmitt trigger input; pull-up enable
1	0	Push-pull output
1	1	A/D converter input (ADC2); Schmitt trigger input off

.3–.2 **Port 0, P0.1/ADC1/INT1 Configuration Bits**

0	0	Schmitt trigger input/falling edge interrupt input
0	1	Schmitt trigger input; pull-up enable/falling edge interrupt input
1	0	Push-pull output
1	1	A/D converter input (ADC1); Schmitt trigger input off

.1–.0 **Port 0, P0.0/ADC0/INT0 Configuration Bits**

0	0	Schmitt trigger input/falling edge interrupt input
0	1	Schmitt trigger input; pull-up enable/falling edge interrupt input
1	0	Push-pull output
1	1	A/D converter input (ADC0); Schmitt trigger input off

P0PND — Port 0 Interrupt Pending Register E8H

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 the S3C9454B/F9454B

.3 **Port 0.1/ADC1/INT1 Interrupt Enable Bit**

0	INT1 falling edge interrupt disable
1	INT1 falling edge interrupt enable

.2 **Port 0.1/ADC1/INT1 Interrupt Pending Bit**

0	No interrupt pending (when read)
0	*Pending bit clear (when write)*
1	Interrupt is pending (when read)
1	No effect (when write)

.1 **Port 0.0/ADC0/INT0 Interrupt Enable Bit**

0	INT0 falling edge interrupt disable
1	INT0 falling edge interrupt enable

.0 **Port 0.0/ADC0/INT0 Interrupt Pending Bit**

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

P1CON — Port 1 Control Register E9H

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 Part 1.1 N-channel open-drain Enable Bit

0	Configure P1.1 as a push-pull output
1	Configure P1.1 as a n-channel open-drain output

.6 Port 1.0 N-channel open-drain Enable Bit

0	Configure P1.0 as a push-pull output
1	Configure P1.0 as a n-channel open-drain output

.5–.4

Not used for S3C9454B/F9454B

.3–.2 Port 1, P1.1 Interrupt Pending Bits

0	0	Schmitt trigger input;
0	1	Schmitt trigger input; pull-up enable
1	0	Output
1	1	Schmitt trigger input; pull-down enable

.1–.0 Port 1, P1.0 Configuration Bits

0	0	Schmitt trigger input;
0	1	Schmitt trigger input; pull-up enable
1	0	Output
1	1	Schmitt trigger input; pull-down enable

NOTE: When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption.

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
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7 Not used for the S3C9454B/F9454B

.6–.4 Port 2, P2.6/ADC8/CLO Configuration Bits

0	0	0	Schmitt trigger input; pull-up enable
0	0	1	Schmitt trigger input
0	1	x	ADC input
1	0	0	Push-pull output
1	0	1	Open-drain output; pull-up enable
1	1	0	Open-drain output
1	1	1	Alternative function; CLO output

.3–.2 Port 2, 2.5 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.1–.0 Port 2, 2.4 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

NOTE: When noise problem is important issue, you had better not use CLO 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 **Part 2, P2.3 Configuration Bits**

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.5–.4 **Port 2, P2.2 Configuration Bits**

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.3–.2 **Port 2, P2.1 Configuration Bits**

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.1–.0 **Port 2, P2.0 Configuration Bits**

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	T0 match output

PWMCON — PWM Control Register E3H

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 PWM Input Clock Selection Bits

0	0	$f_{OSC}/64$
0	1	$f_{OSC}/8$
1	0	$f_{OSC}/2$
1	1	$f_{OSC}/1$

.5

Not used for S3C9454B/F9454B

.4 PWMDATA Reload Interval Selection Bit

0	Reload from 8-bit up counter overflow
1	Reload from 6-bit up counter overflow

.3 PWM Counter Clear Bit

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

.2 PWM Counter Enable Bit

0	Stop counter
1	Start (Resume countering)

.1 PWM Overflow Interrupt Enable Bit (8-Bit Overflow)

0	Disable interrupt
1	Enable interrupt

.0 PWM Overflow Interrupt Pending Bit

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

NOTE: PWMCON.3 is not auto-cleared. You must pay attention when clear pending bit. (refer to page 11-8).

STOPCON — STOP Mode Control Register E4H

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 Watchdog Timer Function Enable Bit

10100101	Enable STOP instruction
Other value	Disable STOP instruction

NOTE: When STOPCON register is not #0A5H value, if you use STOP instruction, PC is changed to reset address.

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–.3

Not used for S3C9454B/F9454B

.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 S3C9454B/F9454B)
0	1	0	Page 2 (Not used for S3C9454B/F9454B)
0	1	1	Page 3 (Not used for S3C9454B/F9454B)

T0CON — TIMER 0 Control Register F4H

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

.7–.6 Timer 0 Input Clock Selection Bits

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/256$
1	0	$f_{OSC}/8$
1	1	$f_{OSC}/1$

.5–.4

Not used for the S3C9454B/F9454B

.3 Timer 0 Counter Clear Bit

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

.2

Not used for the S3C9454B/F9454B

.1 Timer 0 Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer 0 Interrupt Pending Bit (Capture or match interrupt)

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

NOTES:

1. T0CON.3 is not auto-cleared. You must pay attention when clear pending bit. (refer to page 10-12)
2. To use T0 match output, you set T0CON.3 to "1". (refer to page 10-7)

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.

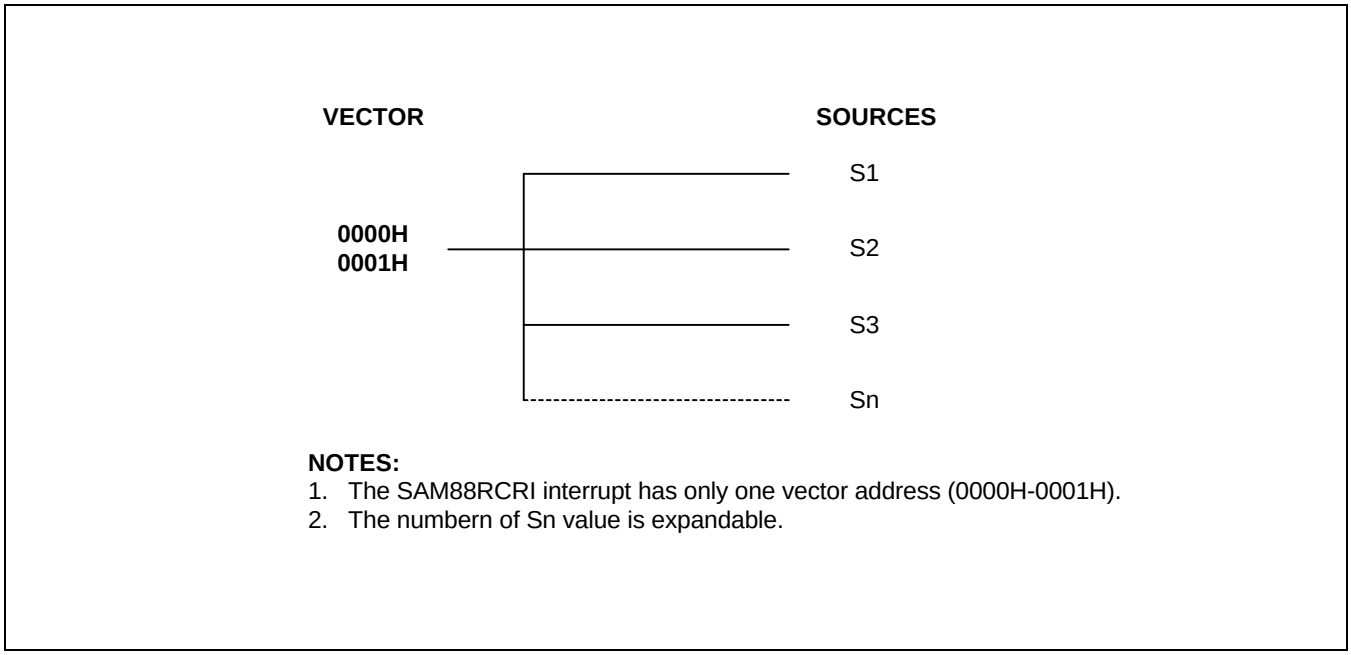


Figure 5-1. S3F9-Series Interrupt Type

INTERRUPT PROCESSING CONTROL POINTS

Interrupt processing can be controlled in two ways: either globally, or by 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.

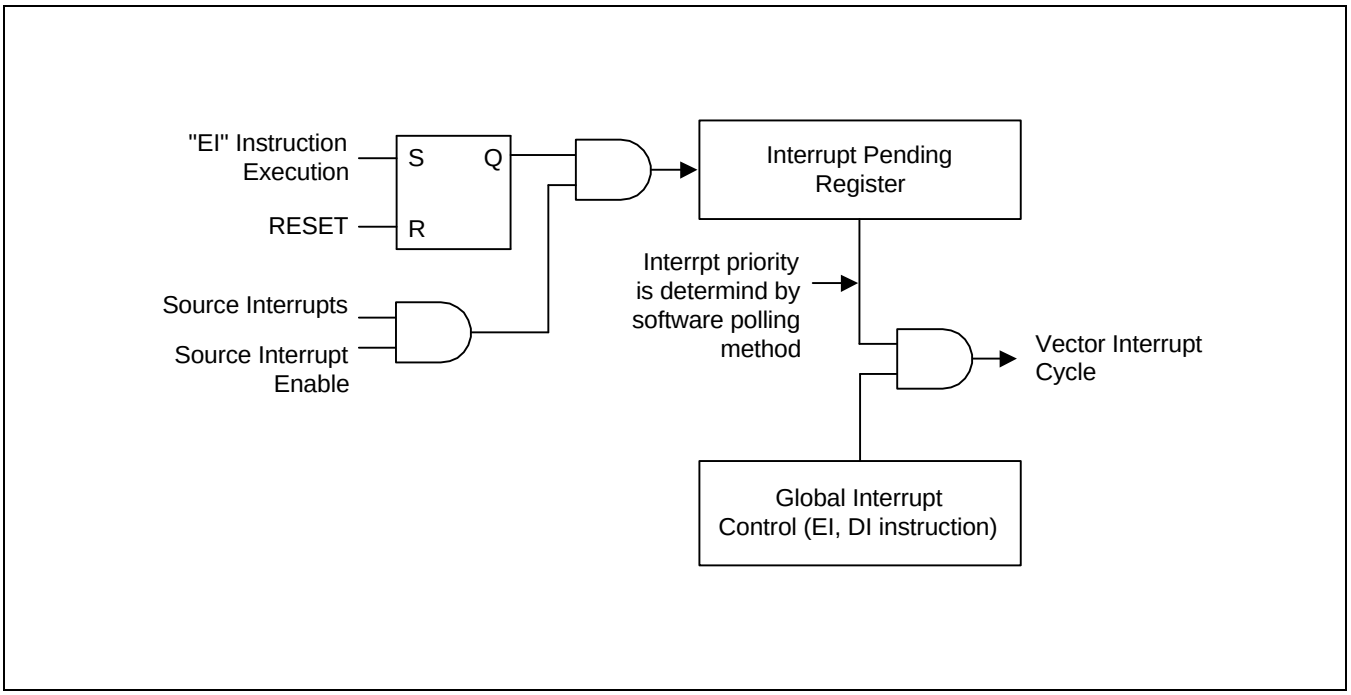


Figure 5-2. Interrupt Function Diagram

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.

S3C9454B/F9454B INTERRUPT STRUCTURE

The S3C9454B/F9454B microcontroller has four peripheral interrupt sources:

— PWM overflow
— Timer 0 match
— P0.0 external interrupt
— P0.1 external interrupt

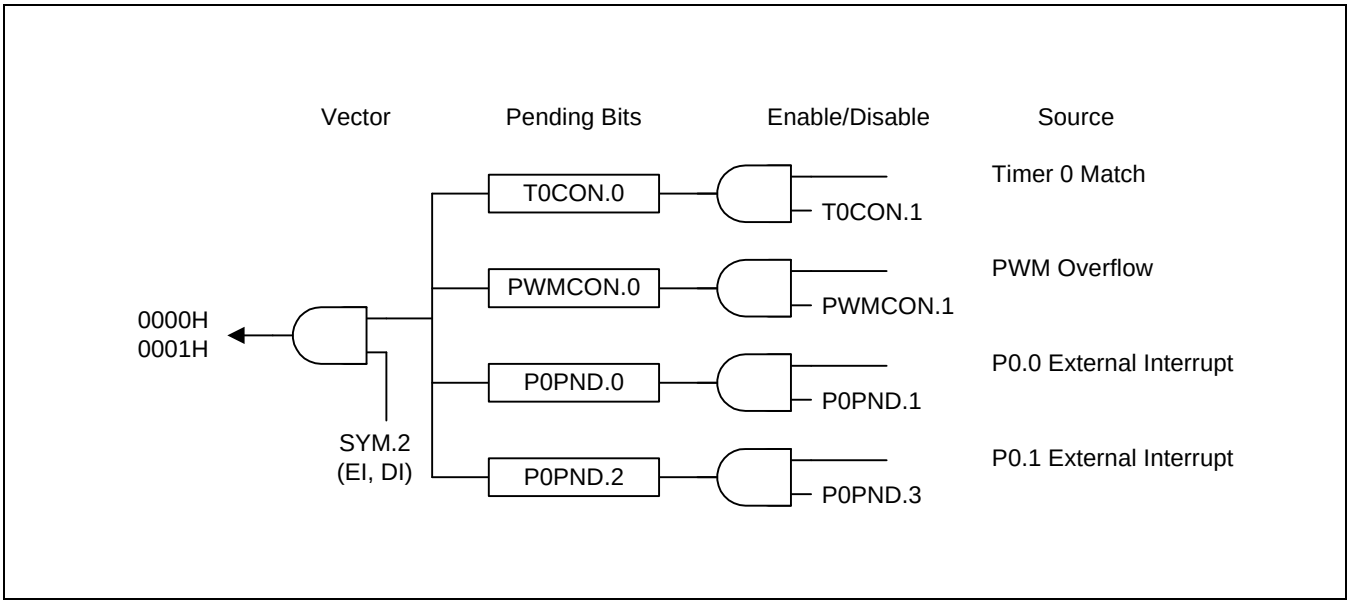


Figure 5-3. S3C9454B/F9454B Interrupt Structure

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.

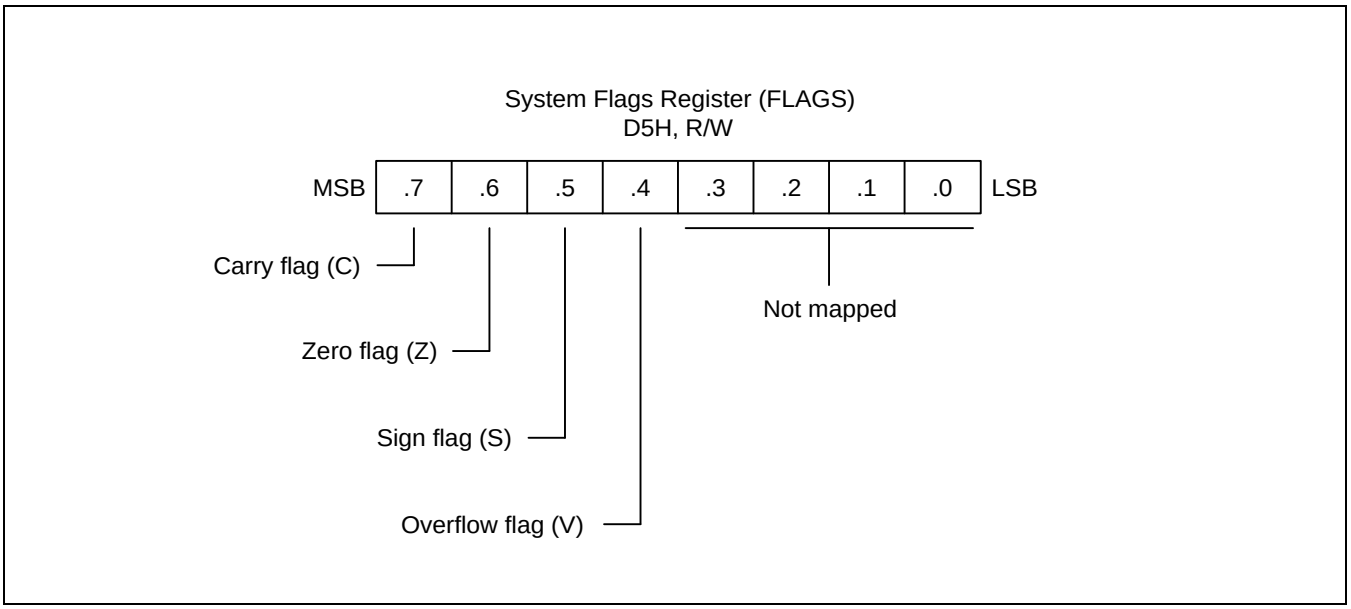


Figure 6-1. System Flags Register (FLAGS)

FLAG DESCRIPTIONS

030303Overflow 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 SAM87RI 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	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	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
opc | dst	2	4	60	R
		4	61	IR

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

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

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

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

CP — Compare

CP dst,src

Operation: dst – src

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

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

opc	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode 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 (2) ← 0

Bit zero of the system mode register, SYM.2, 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 = 04H:

DI

If the value of the SYM register is 04H, the statement "DI" leaves the new value 00H in the register and clears SYM.2 to "0", disabling interrupt processing.

EI — Enable Interrupts

EI

Operation: SYM (2) ← 1

An EI instruction sets bit 2 of the system mode register, SYM.2 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 04H, enabling all interrupts. (SYM.2 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	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	Addr Mode src
opc | dst | src	2	4	42	r	r
		6	43	r	lr
opc | src | dst	3	6	44	R	R
		6	45	R	IR
opc | dst | src	3	6	46	R	IM

Examples: Given: R0 = 15H, R1 = 2AH, R2 = 01H, register 00H = 08H, register 01H = 37H, and register 08H = 8AH:

OR R0,R1 → R0 = 3FH, R1 = 2AH
OR R0,@R2 → R0 = 37H, R2 = 01H, register 01H = 37H
OR 00H,01H → Register 00H = 3FH, register 01H = 37H
OR 01H,@00H → Register 00H = 08H, register 01H = 0BFH
OR 00H,#02H → Register 00H = 0AH

In the first example, if working register R0 contains the value 15H and register R1 the value 2AH, the statement "OR R0,R1" logical-ORs the R0 and R1 register contents and stores the result (3FH) in destination register R0.

The other examples show the use of the logical OR instruction with the various addressing modes and formats.

POP — Pop From Stack

POP dst

Operation: dst ← @SP

SP ← SP + 1

The contents of the location addressed by the stack pointer are loaded into the destination. The stack pointer is then incremented by one.

Flags: No flags affected.

Format:

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

Examples: Given: Register 00H = 01H, register 01H = 1BH, 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	AF
		10	

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

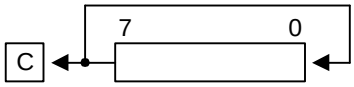
RL dst

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

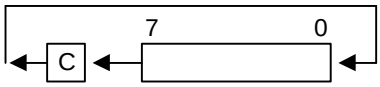
RLC dst

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

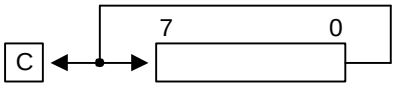
RR dst

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

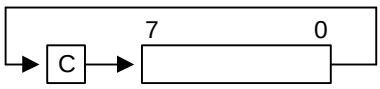
RRC dst

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	Addr Mode src
opc | dst | src	2	4	32	r	r
		6	33	r	lr
opc | src | dst	3	6	34	R	R
		6	35	R	IR
opc | dst | src	3	6	36	R	IM

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

SBC	R1,R2	→	R1 = 0CH, R2 = 03H
SBC	R1,@R2	→	R1 = 05H, R2 = 03H, register 03H = 0AH
SBC	01H,02H	→	Register 01H = 1CH, register 02H = 03H
SBC	01H,@02H	→	Register 01H = 15H,register 02H = 03H, register 03H = 0AH
SBC	01H,#8AH	→	Register 01H = 95H; C, S, and V = "1"

In the first example, if working register R1 contains the value 10H and register R2 the value 03H, the statement "SBC R1,R2" subtracts the source value (03H) and the C flag value ("1") from the destination (10H) and then stores the result (0CH) in register R1.

SCF — Set Carry Flag

SCF

Operation: C ← 1

The carry flag (C) is set to logic one, regardless of its previous value.

Flags: **C:** Set to "1".

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	DF

Example: The statement

SCF

sets the carry flag to logic one.

SRA — Shift Right Arithmetic

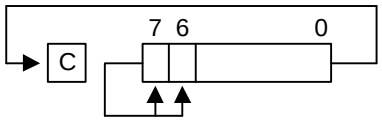
SRA dst

Operation: dst (7) ← dst (7)

C ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

An arithmetic shift-right of one bit position is performed on the destination operand. Bit zero (the LSB) replaces the carry flag. The value of bit 7 (the sign bit) is unchanged and is shifted into bit position 6.



Flags:
C: Set if the bit shifted from the LSB position (bit zero) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Always cleared to "0".

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	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	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	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

By smart option (3FH.1–.0 in ROM), user can select internal RC oscillator or external oscillator. In using internal oscillator, XIN (P1.0), XOUT (P1.1) can be used by normal I/O pins. An internal RC oscillator source provides a typical 3.2 MHz or 0.5 MHz (in V_{DD} = 5 V) depending on smart option.

An external RC oscillation source provides a typical 4 MHz clock for S3C9454B/F9454B. An internal capacitor supports the RC oscillator circuit. An external crystal or ceramic oscillation source provides a maximum 10 MHz clock. The X_{IN} and X_{OUT} pins connect the oscillation source to the on-chip clock circuit. Simplified external RC oscillator and crystal/ceramic oscillator circuits are shown in Figures 7-1 and 7-2. When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption

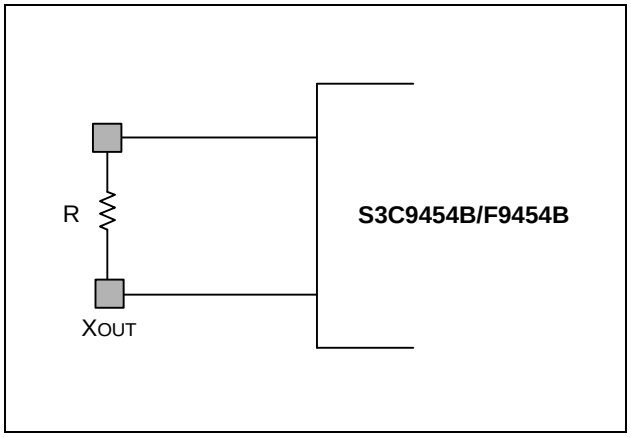


Figure 7-1. Main Oscillator Circuit (RC Oscillator with Internal Capacitor)

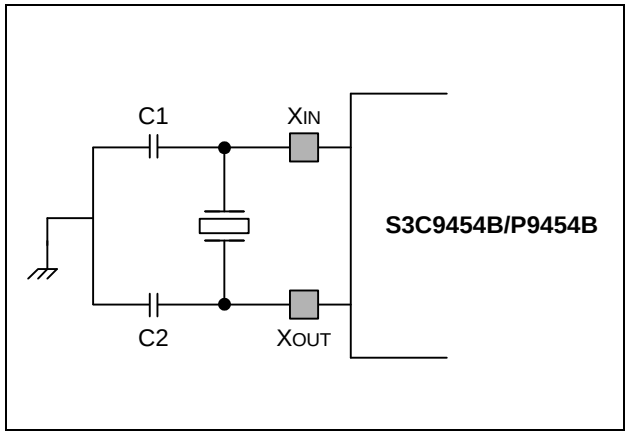


Figure 7-2. Main Oscillator Circuit (Crystal/Ceramic Oscillator)

MAIN OSCILLATOR LOGIC

To increase processing speed and to reduce clock noise, non-divided logic is implemented for the main oscillator circuit. For this reason, very high resolution waveforms (square signal edges) must be generated in order for the CPU to efficiently process logic operations.

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 S3C9454B/F9454B, 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 in location D4H. It is read/write addressable and has the following functions:

— Oscillator IRQ wake-up function enable/disable (CLKCON.7)

— Oscillator frequency divide-by value: non-divided, 2, 8, or 16 (CLKCON.4 and CLKCON.3)

The CLKCON register controls whether or not an external interrupt can be used to trigger a Stop mode release (This is called the "IRQ wake-up" function). The IRQ wake-up enable bit is CLKCON.7.

After a reset, the external interrupt oscillator wake-up function is enabled, the main oscillator is activated, and the $f_{OSC}/16$ (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed to f_{OSC}, $f_{OSC}/2$ or $f_{OSC}/8$.

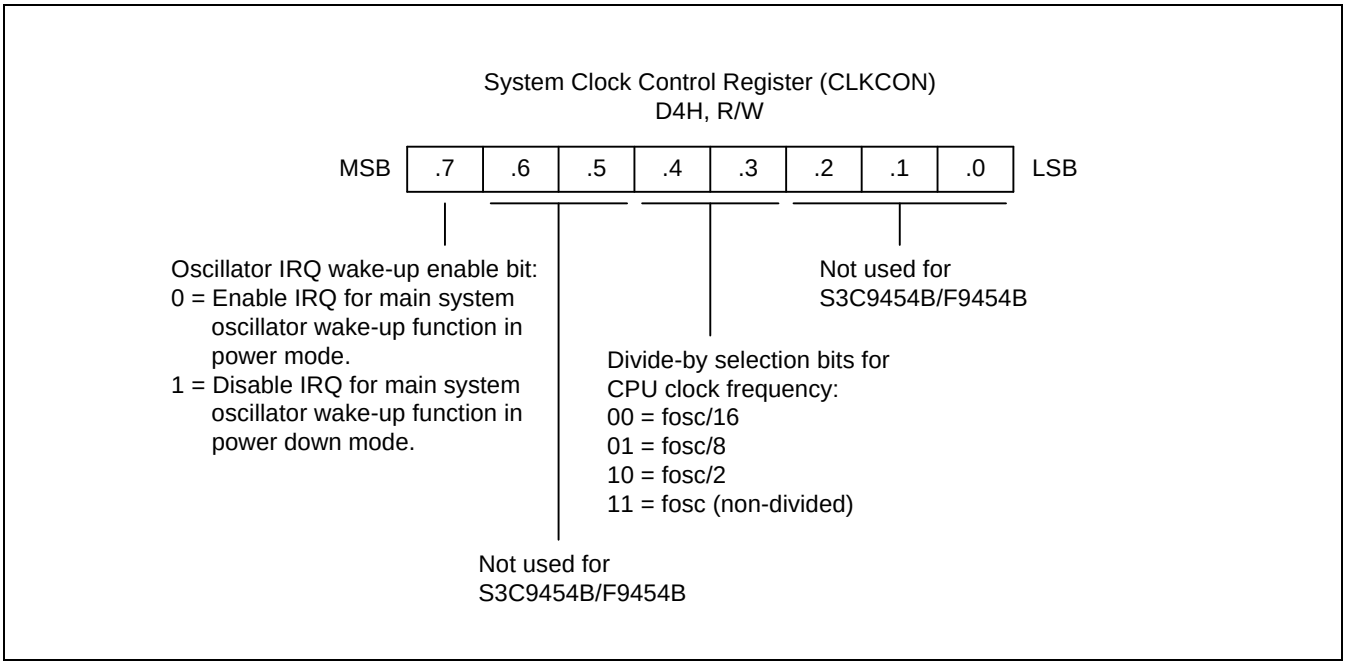
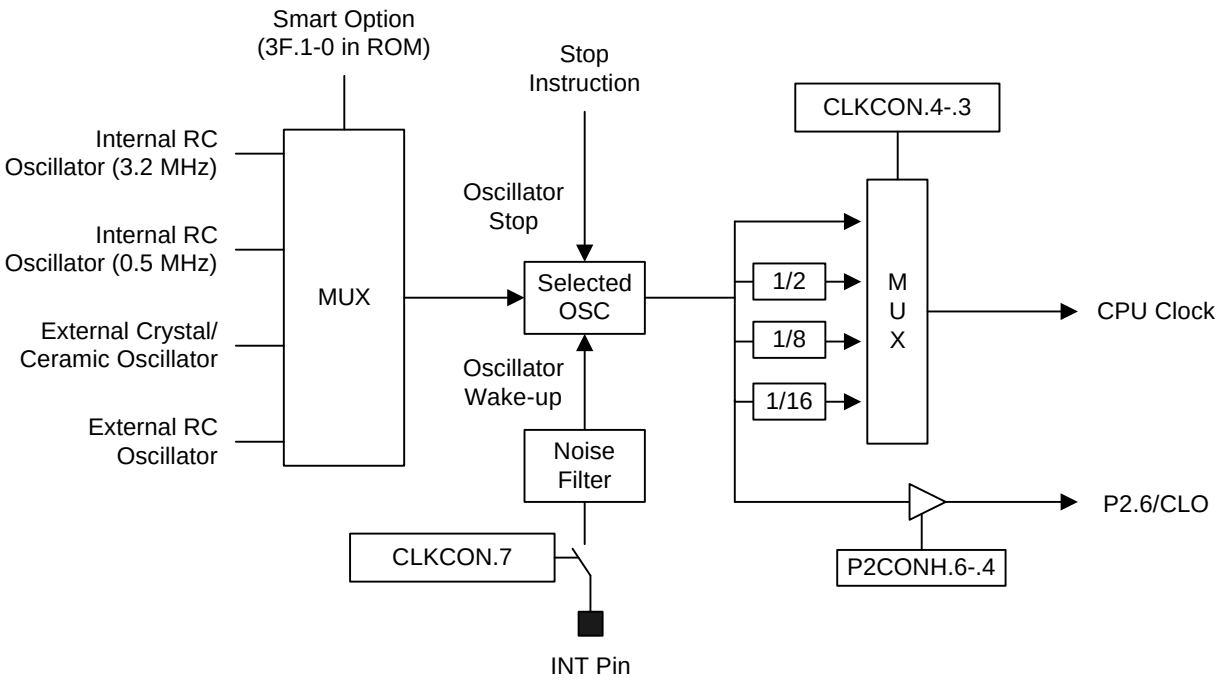


Figure 7-3. System Clock Control Register (CLKCON)



NOTE: An external interrupt (with RC-delay noise filter) can be used to release stop mode and "wake-up" the main oscillator.
In the S3C9454B/F9454B, the INT0-INT1 external interrupts are of this type.

Figure 7-4. System Clock Circuit Diagram

NOTES

8 RESET and POWER-DOWN

SYSTEM RESET

OVERVIEW

By smart option (3EH.7 in ROM), user can select internal RESET (LVR) or external RESET. In using internal RESET (LVR), nRESET pin (P1.2) can be used by normal I/O pin.

The S3C9454B/F9454B can be RESET in four ways:

— by external power-on-reset

— by the external nRESET input pin pulled low

— by the digital watchdog peripheral timing out

— by Low Voltage Reset (LVR)

During a external power-on reset, the voltage at V_{DD} is High level and the nRESET pin is forced to Low level. The nRESET signal is input through a Schmitt trigger circuit where it is then synchronized with the CPU clock. This brings the S3C9454B/F9454B into a known operating status. To ensure correct start-up, the user should take care that nRESET signal is not released before the V_{DD} level is sufficient to allow MCU operation at the chosen frequency.

The nRESET pin must be held to Low level for a minimum time interval after the power supply comes within tolerance in order to allow time for internal CPU clock oscillation to stabilize. The minimum required oscillation stabilization time for a reset is approximately 6.55 ms (@ $2^{16}/f_{OSC}$, f_{OSC} = 10 MHz).

When a reset occurs during normal operation (with both V_{DD} and nRESET at High level), the signal at the nRESET pin is forced Low and the Reset operation starts. All system and peripheral control registers are then set to their default hardware Reset values (see Table 8-1).

The MCU provides a watchdog timer function in order to ensure graceful recovery from software malfunction. If watchdog timer is not refreshed before an end-of-counter condition (overflow) is reached, the internal reset will be activated.

The on-chip Low Voltage Reset, features static Reset when supply voltage is below a reference value (Typ. 2.3, 3.0, 3.9 V). Thanks to this feature, external reset circuit can be removed while keeping the application safety. As long as the supply voltage is below the reference value, there is a internal and static RESET. The MCU can start only when the supply voltage rises over the reference value.

NOTE

To program the duration of the oscillation stabilization interval, you must make the appropriate settings to the basic timer control register, BTCON, before entering Stop mode. Also, if you do not want to use the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), you can disable it by writing "1010B" to the upper nibble of BTCON.

MCU Initialization Sequence

The following sequence of events occurs during a Reset operation:

— All interrupts are disabled.
— The watchdog function (basic timer) is enabled.
— Ports 0–2 are set to input mode
— Peripheral control and data registers are disabled and reset to their initial values (see Table 8-1).
— The program counter is loaded with the ROM reset address, 0100H.
— When the programmed oscillation stabilization time interval has elapsed, the address stored in ROM location 0100H (and 0101H) is fetched and executed.

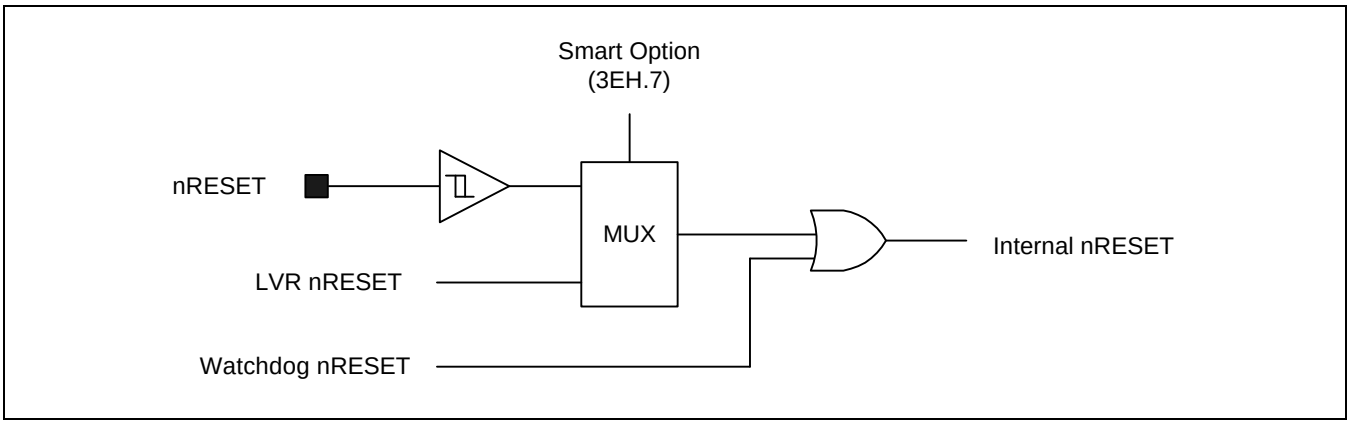


Figure 8-1. Reset Block Diagram

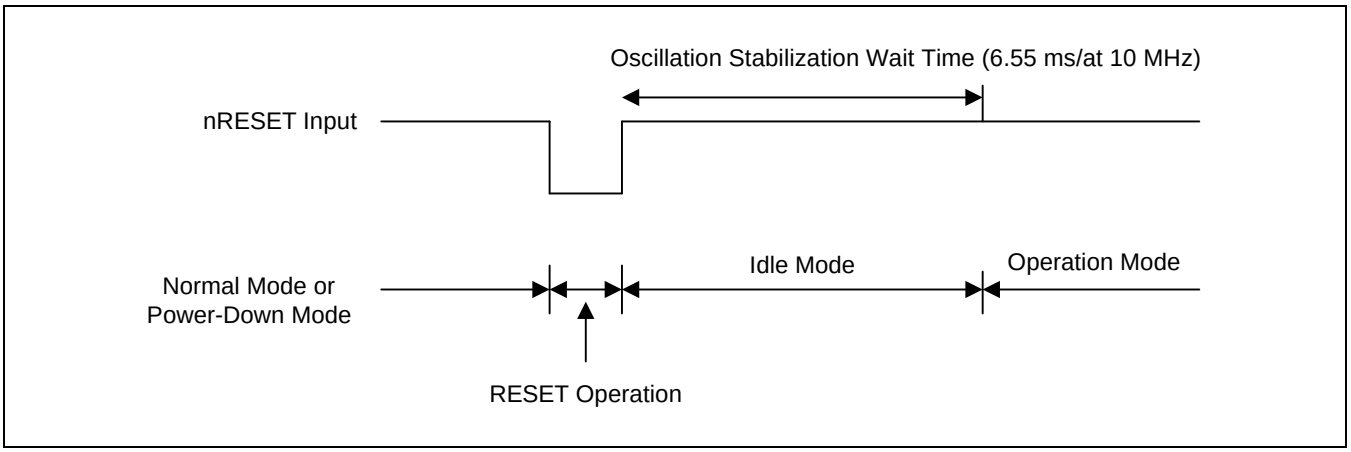


Figure 8-2. Timing for S3C9454B/F9454B After RESET

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 5 μA except that the LVR(Low Voltage Reset) is enable. All system functions are halted 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 nRESET signal or by an external interrupt.

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 then Reset to their default values and the contents of all data registers are retained. A Reset operation automatically selects a slow clock (f_{OSC}/16) because CLKCON.3 and CLKCON.4 are cleared to "00B". After the oscillation stabilization interval has elapsed, the CPU executes the system initialization routine by fetching the 16-bit address stored in ROM locations 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 (Clock-related external interrupts cannot be used). External interrupts INT0-INT1 in the S3C9454B/F9454B interrupt structure meet this criteria.

Note that when Stop mode is released by an external interrupt, the current values in system and peripheral control registers are not changed. When you use an interrupt to release Stop mode, the CLKCON.3 and CLKCON.4 register values remain unchanged, and the currently selected clock value is used. 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 put the appropriate value to BTCON register *before* entering Stop mode.

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 select peripherals remain active. During Idle mode, the internal clock signal is gated off to the CPU, but not to interrupt logic and timer/counters. 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 a slow clock (f_{OSC}/16) because CLKCON.3 and CLKCON.4 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.3 and CLKCON.4 register values remain unchanged, and the currently selected clock value is used. The interrupt is then serviced. Following the IRET from the service routine, the instruction immediately following the one that initiated Idle mode is executed.

NOTES

1. Only external interrupts that are not clock-related can be used to release stop mode. To release Idle mode, however, any type of interrupt (that is, internal or external) can be used.
2. Before enter the STOP or IDLE mode, the ADC must be disabled. Otherwise, the STOP or IDLE current will be increased significantly.

HARDWARE RESET VALUES

Table 8-1 lists the values for CPU and system registers, peripheral control registers, and peripheral data registers following a Reset operation in normal operating mode.

— 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 following a reset.

— A dash ("–") means that the bit is either not used or not mapped.

Table 8-1. Register Values After a Reset

Register Name	Mnemonic	Address & Location		RESET Value (Bit)							
		Address	R/W	7	6	5	4	3	2	1	0
Timer 0 counter register	T0CNT	D0H	R	0	0	0	0	0	0	0	0
Timer 0 data register	T0DATA	D1H	R/W	1	1	1	1	1	1	1	1
Timer 0 control register	T0CON	D2H	R/W	0	0	–	–	0	–	0	0
Location D3H is not mapped											
Clock control register	CLKCON	D4H	R/W	0	–	–	0	0	–	–	–
System flags register	FLAGS	D5H	R/W	x	x	x	x	–	–	–	–
Locations D6H–D8H are not mapped											
Stack pointer register	SP	D9H	R/W	x	x	x	x	x	x	x	x
Location DAH is not mapped											
MDS special register	MDSREG	DBH	R/W	0	0	0	0	0	0	0	0
Basic timer control register	BTCON	DCH	R/W	0	0	0	0	0	0	0	0
Basic timer counter	BTCNT	DDH	R	0	0	0	0	0	0	0	0
Test mode control register	FTSTCON	DEH	W	–	–	0	0	0	0	0	0
System mode register	SYM	DFH	R/W	–	–	–	–	0	0	0	0

NOTE: – : Not mapped or not used, x: undefined

Table 8-1. Register Values After a Reset (Continued)

Register Name	Mnemonic	Address Hex	R/W	Bit Values After RESET							
				7	6	5	4	3	2	1	0
Port 0 data register	P0	E0H	R/W	0	0	0	0	0	0	0	0
Port 1 data register	P1	E1H	R/W	–	–	–	–	–	0	0	0
Port 2 data register	P2	E2H	R/W	–	0	0	0	0	0	0	0
Locations E3H–E5H are not mapped											
Port 0 control register (High byte)	P0CONH	E6H	R/W	0	0	0	0	0	0	0	0
Port 0 control register	P0CON	E7H	R/W	0	0	0	0	0	0	0	0
Port 0 interrupt pending register	P0PND	E8H	R/W	–	–	–	–	0	0	0	0
Port 1 control register	P1CON	E9H	R/W	0	0	–	–	0	0	0	0
Port 2 control register (High byte)	P2CONH	EAH	R/W	–	0	0	0	0	0	0	0
Port 2 control register (Low byte)	P2CONL	EBH	R/W	0	0	0	0	0	0	0	0
Locations ECH–F1H are not mapped											
PWM data register	PWMDATA	F2H	R/W	0	0	0	0	0	0	0	0
PWM control register	PWMCON	F3H	R/W	0	0	–	0	0	0	0	0
STOP control register	STOPCON	F4H	R/W	0	0	0	0	0	0	0	0
Locations F5H–F6H are not mapped											
A/D control register	ADCON	F7H	R/W	0	0	0	0	0	0	0	0
A/D converter data register (High)	ADDATAH	F8H	R	x	x	x	x	x	x	x	x
A/D converter data register (Low)	ADDATAL	F9H	R	0	0	0	0	0	0	x	x
Locations FAH–FFH are not mapped											

NOTE: – : Not mapped or not used, x: undefined

☞ PROGRAMMING TIP — Sample S3C9454B/F9454B Initialization Routine

```
;--------------<< Interrupt Vector Address >>
                ORG     0000H
                VECTOR  00H,INT_9454        ;  S3C9454B/F9454B has only one interrupt vector

;--------------<< Smart Option >>
                ORG     003CH
                DB      00H                 ;  003CH, must be initialized to 0
                DB      00H                 ;  003DH, must be initialized to 0
                DB      0E7H                ;  003EH, enable LVR (2.3 V)
                DB      03H                 ;  003FH, internal RC (3.2 MHz in VDD = 5 V )

;--------------<< Initialize System and Peripherals >>
                ORG     0100H
RESET:          DI                          ;  disable interrupt
                LD      BTCON,#10100011B    ;  Watch-dog disable
                LD      CLKCON,#00011000B   ;  Select non-divided CPU clock
                LD      SP,#0C0H            ;  Stack pointer must be set


                LD      P0CONH,#10101010B   ;
                LD      P0CONL,#10101010B   ;  P0.0–P0.7 push-pull output
                LD      P1CON,#00001010B    ;  P1.0–P1.1 push-pull output
                LD      P2CONH,#01001010B   ;
                LD      P2CONL,#10101010B   ;  P2.0–P2.6 push-pull output

;--------------<< Timer 0 settings >>
                LD      T0DATA,#50H         ;  CPU = 3.2 MHz, interrupt interval = 6.4 msec
                LD      T0CON,#01001010B    ;  fOSC/256, Timer 0 interrupt enable

;--------------<< Clear all data registers from 00h to 5FH >>
                LD      R0,#0               ;  RAM clear
RAM_CLR:        CLR     @R0                 ;
                INC     R0                  ;
                CP      R0,#0BFH            ;
                JP      ULE,RAM_CLR

;--------------<< Initialize other registers >>
                •
                •
                •
                EI                          ;  Enable interrupt
```

☞ **PROGRAMMING TIP — Sample S3C9454B/F9454B Initialization Routine (Continued)**

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

MAIN:       NOP                         ;  Start main loop
            LD          BTCON,#02H      ;  Enable watchdog function
                                        ;  Basic counter (BTCNT) clear
            •
            •
            CALL        KEY_SCAN        ;
            •
            •
            •
            CALL        LED_DISPLAY     ;
            •
            •
            •
            CALL        JOB             ;
            •
            •
            •
            JR          T,MAIN          ;

;--------------<< Subroutines >>

KEY_SCAN:   NOP                         ;
            •
            •
            •
            RET

LED_DISPLAY: NOP                        ;
            •
            •
            •
            RET

JOB:        NOP                         ;
            •
            •
            •
            RET
```

☞ **PROGRAMMING TIP — Sample S3C9454B/F9454B Initialization Routine (Continued)**

```
;--------------<< Interrupt Service Routines >>          ;  Interrupt enable bit and pending bit check

INT_9454:   TM          T0CON,#00000010B    ;  Timer0 interrupt enable check
            JR          Z,NEXT_CHK1         ;
            TM          T0CON,#00000001B    ;  If timer0 interrupt was occurred,
            JP          NZ,INT_TIMER0       ;  T0CON.0 bit would be set.

NEXT_CHK1:

            TM          PWMCOM,#00000010B   ;  PWM overflow interrupt enable check
            JR          Z,NEXT_CHK2         ;
            TM          P0PND,#00000001B    ;
            JP          NZ,PWMOVF_INT       ;

NEXT_CHK2:

            TM          P0PND,#00000010B    ;  INT0 interrupt enable check
            JR          Z,NEXT_CHK3         ;
            TM          P0PND,#00000001B    ;
            JP          NZ,INT0_INT         ;

NEXT_CHK3:

            TM          P0PND,#00001000B    ; INT1 interrupt enable check
            JP          Z,END_INT           ;
            TM          P0PND,#00000100B    ;
            JP          NZ,INT1_INT         ;
            IRET                            ;  Interrupt return

END_INT     ; IRET

;--------------< Timer0 interrupt service routine >

INT_TIMER0:

            •                               ;
            •
            AND         T0CON,#1111_0_110B  ;  Pending bit clear
            IRET                            ;  Interrupt return

;--------------< PWM overflow interrupt service routine >

PWMOVF_INT:

            •
            •
            AND         PWMCON,#1111_0_110B ;  Pending bit clear
            IRET                            ;  Interrupt return
```

☞ **PROGRAMMING TIP — Sample S3C9454B/F9454B Initialization Routine (Continued)**

```
;--------------< External interrupt0 service routine >

INT0_INT:   •
            •
            AND         P0PND,#11111110B    ;  INT0 Pending bit clear
            IRET                            ;  Interrupt return

;--------------< External interrupt1 service routine >

INT1_INT:   •
            •
            AND         P0PND,#11111011B    ;  INT1 Pending bit clear
            IRET                            ;  Interrupt return
            •
            •
            END                             ;
```

NOTES

9 I/O PORTS

OVERVIEW

The S3C9454B/F9454B has three I/O ports: with 18 pins total. You access these ports directly by writing or reading port data register addresses.

All ports can be configured as LED drive. (High current output: typical 10 mA)

Table 9-1. S3C9454B/F9454B Port Configuration Overview

Port	Function Description	Programmability
0	Bit-programmable I/O port for schmitt trigger input or push-pull output. Pull-up resistors are assignable by software. Port 0 pins can also be used as alternative function. (ADC input, external interrupt input).	Bit
1	Bit-programmable I/O port for schmitt trigger input or push-pull, open-drain output. Pull-up or pull-down resistors are assignable by software. Port 1 pins can also oscillator input/output or reset input by smart option. P1.2 is input only.	Bit
2	Bit-programmable I/O port for schmitt trigger input or push-pull, open-drain output. Pull-up resistor are assignable by software. Port 2 can also be used as alternative function (ADC input, CLO, T0 clock output)	Bit

PORT DATA REGISTERS

Table 9-2 gives you an overview of the port data register names, locations, and addressing characteristics. Data registers for ports 0-2 have the structure shown in Figure 9-1.

Table 9-2. Port Data Register Summary

Register Name	Mnemonic	Hex	R/W
Port 0 data register	P0	E0H	R/W
Port 1 data register	P1	E1H	R/W
Port 2 data register	P2	E2H	R/W

NOTE: A reset operation clears the P0–P2 data register to "00H".

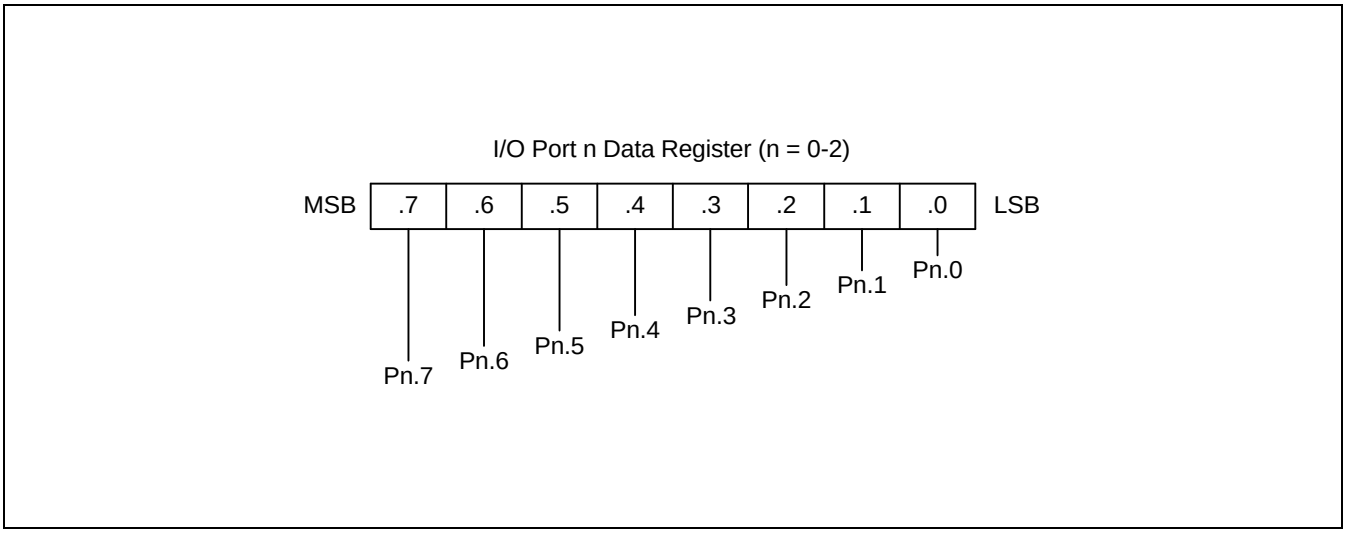


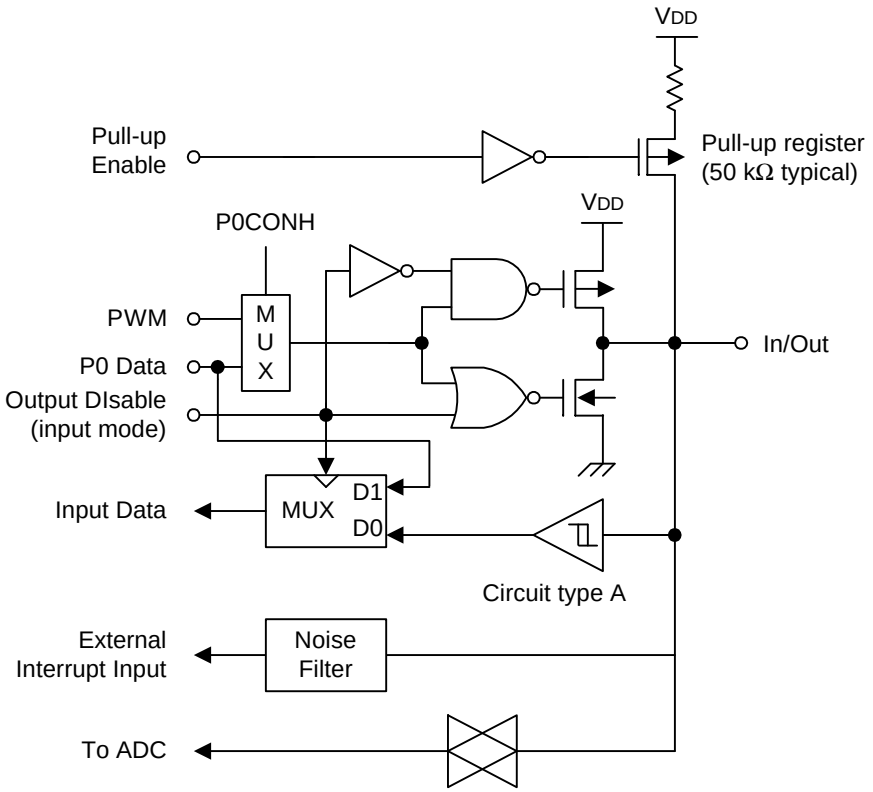
Figure 9-1. Port Data Register Format

PORT 0

Port 0 is a bit-programmable, general-purpose, I/O ports. You can select normal input or push-pull output mode. In addition, you can configure a pull-up resistor to individual pins using control register settings.
It is designed for high-current functions such as LED direct drive. Part 0 pins can also be used as alternative functions (ADC input, external interrupt input and PWM output).

Two control resisters are used to control Port 0: P0CONH (E6H) and P0CONL (E7H).

You access port 0 directly by writing or reading the corresponding port data register, P0 (E0H).



NOTE: I/O pins have protection diodes through VDD and VSS.

Mode	Input Data
Output	D0
Input	D1

Figure 9-2. Port 0 Circuit Diagram

Port 0 Control Register (High Byte)
E6H, R/W

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

[.7-.6] Port, P0.7/ADC7 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = A/D converter input (ADC7); schmitt trigger input off

[.5-.4] Port 0, P0.6/ADC6/PWM Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Alternative function (PWM output)
1 0 = Push-pull output
1 1 = A/D converter input (ADC6); schmitt trigger input off

[.3-.2] Port 0, P0.5/ADC5 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = A/D converter input (ADC5); schmitt trigger input off

[.1-.0] Port 0, P0.4/ADC4 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = A/D converter input (ADC4); schmitt trigger input off

Figure 9-3. Port 0 Control Register (P0CONH, High Byte)

Port 0 Control Register (Low Byte)
E7H, R/W

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

[.7-.6] Port 0, P0.3/ADC3 Configuration Bits
0 0 = Schmitt trigger input
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = A/D converter input (ADC3); Schmitt trigger input off

[.5-.4] Port 0, P0.2/ADC2 Configuration Bits
0 0 = Schmitt trigger input
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = A/D converter input (ADC2); Schmitt trigger input off

[.3-.2] Port 0, P0.1/ADC1/INT1 Configuration Bits
0 0 = Schmitt trigger input/falling edge interrupt input
0 1 = Schmitt trigger input; pull-up enable/falling edge interrupt input
1 0 = Push-pull output
1 1 = A/D converter input (ADC1); Schmitt trigger input off

[.1-.0] Port 0, P0.0/ADC0/INT0 Configuration Bits
0 0 = Schmitt trigger input/falling edge interrupt input
0 1 = Schmitt trigger input; pull-up enable/falling edge interrupt input
1 0 = Push-pull output
1 1 = A/D converter input (ADC0); Schmitt trigger input off

Figure 9-4. Port 0 Control Register (P0CONL, Low Byte)

Port 0 Interrupt Pending Register
E8H, R/W

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

[.7-.4] Not used for S3C9454B/F9454B

[.3] Port 0.1/ADC1/INT1, Interrupt Enable Bit
0 = INT1 falling edge interrupt disable
1 = INT1 falling edge interrupt enable

[.2] Port 0.1/ADC1/INT1, Interrupt Pending Bit
0 = No interrupt pending (when read)
0 = Pending bit clear (when write)
1 = Interrupt is pending (when read)
1 = No effect (when write)

[.1] Port 0.0/ADC0/INT0, Interrupt Enable Bit
0 = INT0 falling edge interrupt disable
1 = INT0 falling edge interrupt enable

[.0] Port 0.0/ADC0/INT0, Interrupt Pending Bit
0 = No interrupt pending (when read)
0 = Pending bit clear (when write)
1 = Interrupt is pending (when read)
1 = No effect (when write)

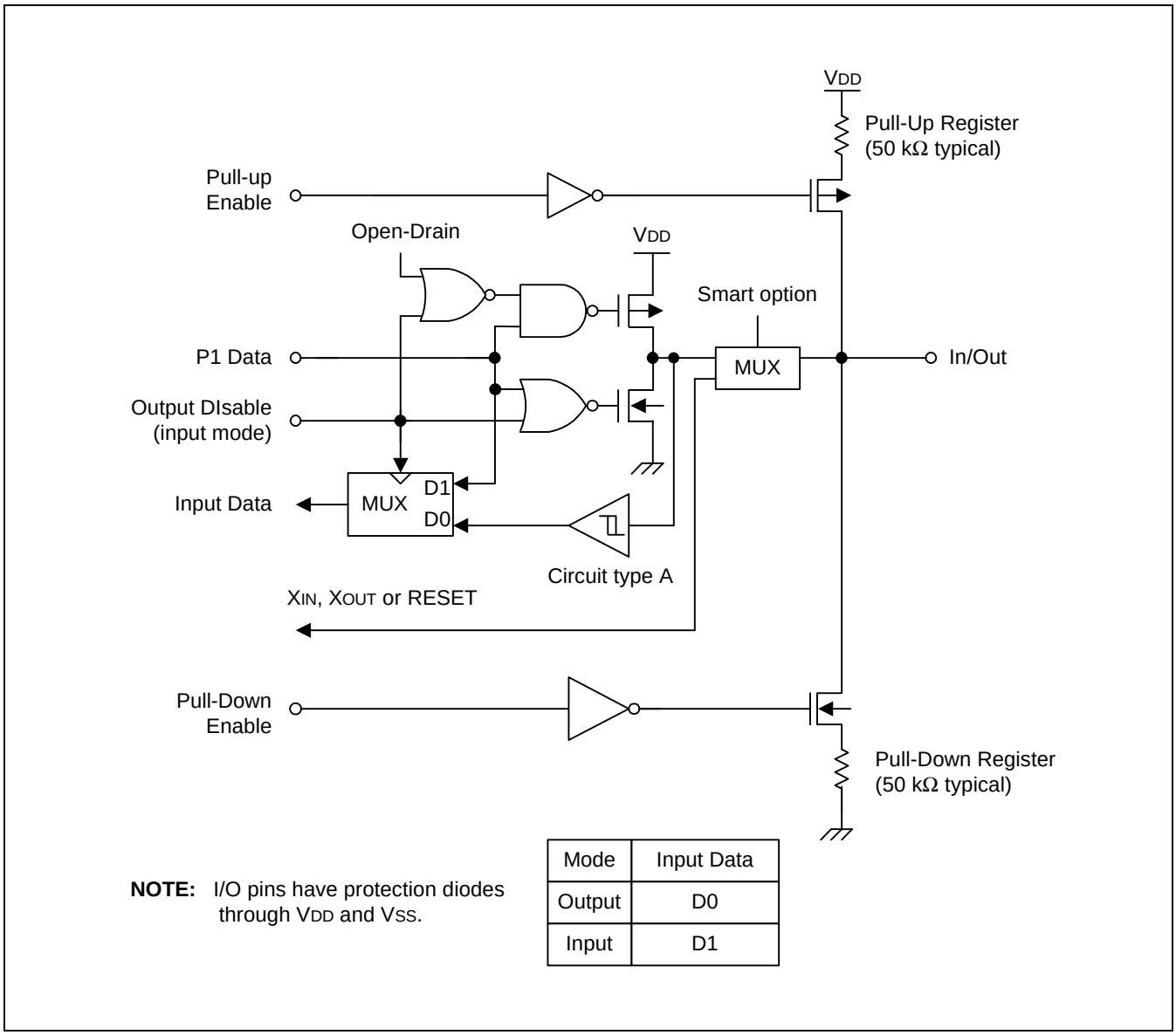
Figure 9-5. Port 0 Interrupt Pending Registers (P0PND)

PORT 1

Port 1, is a 3-bit I/O port with individually configurable pins. It can be used for general I/O port (Schmitt trigger input mode, push-pull output mode or n-channel open-drain output mode). In addition, you can configure a pull-up and pull-down resistor to individual pin using control register settings. It is designed for high-current functions such as LED direct drive.

P1.0, P1.1 are used for oscillator input/output by smart option. Also, P1.2 is used for RESET pin by smart option.

One control register is used to control port 1: P1CON (E9H).
You address port 1 bits directly by writing or reading the port 1 data register, P1 (E1H). When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption.



NOTE: I/O pins have protection diodes through VDD and VSS.

Mode	Input Data
Output	D0
Input	D1

Figure 9-6. Port 1 Circuit Diagram

Port 1 Control Register
E9H, R/W

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

[.7] Port 1.1 N-Channel Open-Drain Enable Bit
0 = Configure P1.1 as a push-pull output
1 = Configure P1.1 as a n-channel open-drain output

[.6] Port 1.0 N-Channel Open-Drain Enable Bit
0 = Configure P1.0 as a push-pull output
1 = Configure P1.0 as a N-channel open-drain output

[.5-.4] Not used for S3C9454B/F9454B

[.3-.2] Port 1, P1.1 Configuration Bits
0 0 = Schmitt trigger input;
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = Schmitt trigger input; pull-down enable

[.1-.0] Port 1, P1.0 Configuration Bits
0 0 = Schmitt trigger input;
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = Schmitt trigger input; pull-down enable

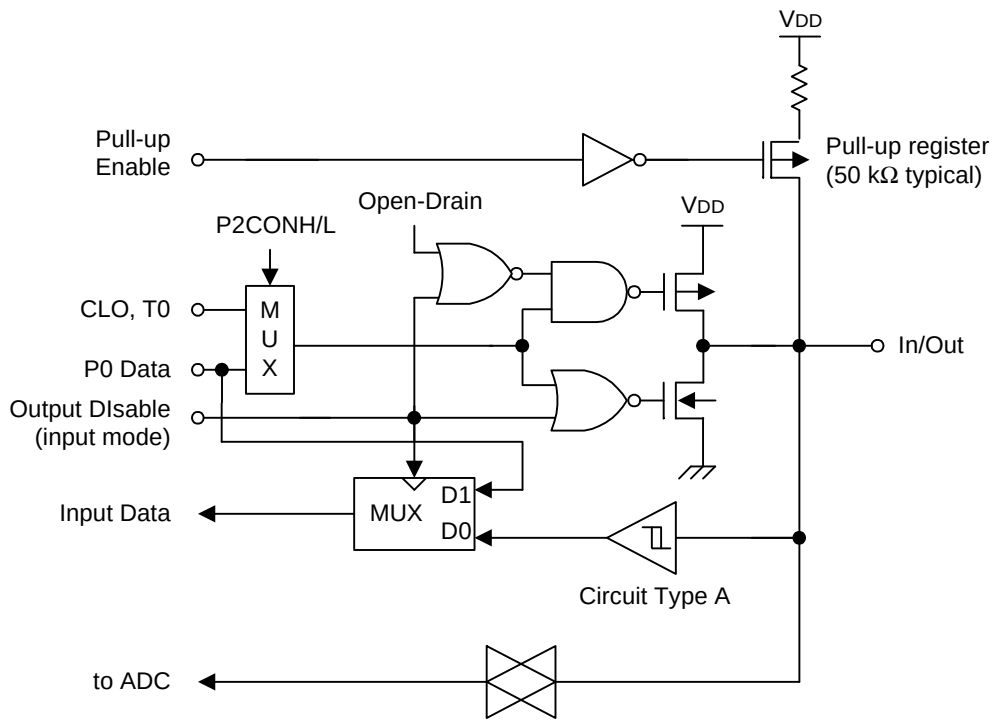
NOTE: When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption.

Figure 9-7. Port 1 Control Register (P1CON)

PORT 2

Port 2 is a 7-bit I/O port with individually configurable pins. It can be used for general I/O port (schmitt trigger input mode, push-pull output mode or N-channel open-drain output mode). You can also use some pins of port 2 ADC input, CLO output and T0 clock output. In addition, you can configure a pull-up resistor to individual pins using control register settings. It is designed for high-current functions such as LED direct drive.

You address port 2 bits directly by writing or reading the port 2 data register, P2 (E2H). The port 2 control register, P2CONH and P2CONL is located at addresses EAH, EBH respectively.



NOTE: I/O pins have protection diodes through V_{DD} and V_{SS}.

Mode	Input Data
Output	D0
Input	D1

Figure 9-8. Port 2 Circuit Diagram

Port 2 Control Register (High Byte)
EAH, R/W

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

[.7] Not sued for S3C9454B/F9454B

[.6-.4] Port 2, P2.6/ADC8/CLO Configuration Bits
0 0 0 = Schmitt trigger input; pull-up enable
0 0 1 = Schmitt trigger input
0 1 x = ADC input
1 0 0 = Push-pull output
1 0 1 = Open-drain output; pull-up enable
1 1 0 = Open-drain output
1 1 1 = Alternative function; CLO output

[.3-.2] Port 2, P2.5 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.1-.0] Port 2, P2.4 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

NOTE: When noise problem is important issue, you had better not use CLO output

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

Port 2 Control Register (Low Byte)
EBH, R/W

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

[.7-.6] Port 2, P2.3 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.5-.4] Port 2, P2.2 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.3-.2] Port 2, P2.1 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.1-.0] Port 2, P2.0 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = T0 match output

Figure 9-10. Port 2 Control Register (P2CONL, Low Byte)

NOTES

10 BASIC TIMER and TIMER 0

MODULE OVERVIEW

The S3C9454B/F9454B has two default timers: an 8-bit basic timer, one 8-bit general-purpose timer/counter, called timer 0.

Basic Timer (BT)

You can use the basic timer (BT) in two different ways:

— As a watchdog timer to provide an automatic Reset mechanism in the event of a system malfunction.

— To signal the end of the required oscillation stabilization interval after a Reset or a Stop mode release.

The functional components of the basic timer block are:

— Clock frequency divider (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)

Timer 0

Timer 0 has the following functional components:

— Clock frequency divider (f_{OSC} divided by 4096, 256, 8, or f_{OSC}) with multiplexer

— 8-bit counter (T0CNT), 8-bit comparator, and 8-bit data register (T0DATA)

— Timer 0 control register (T0CON)

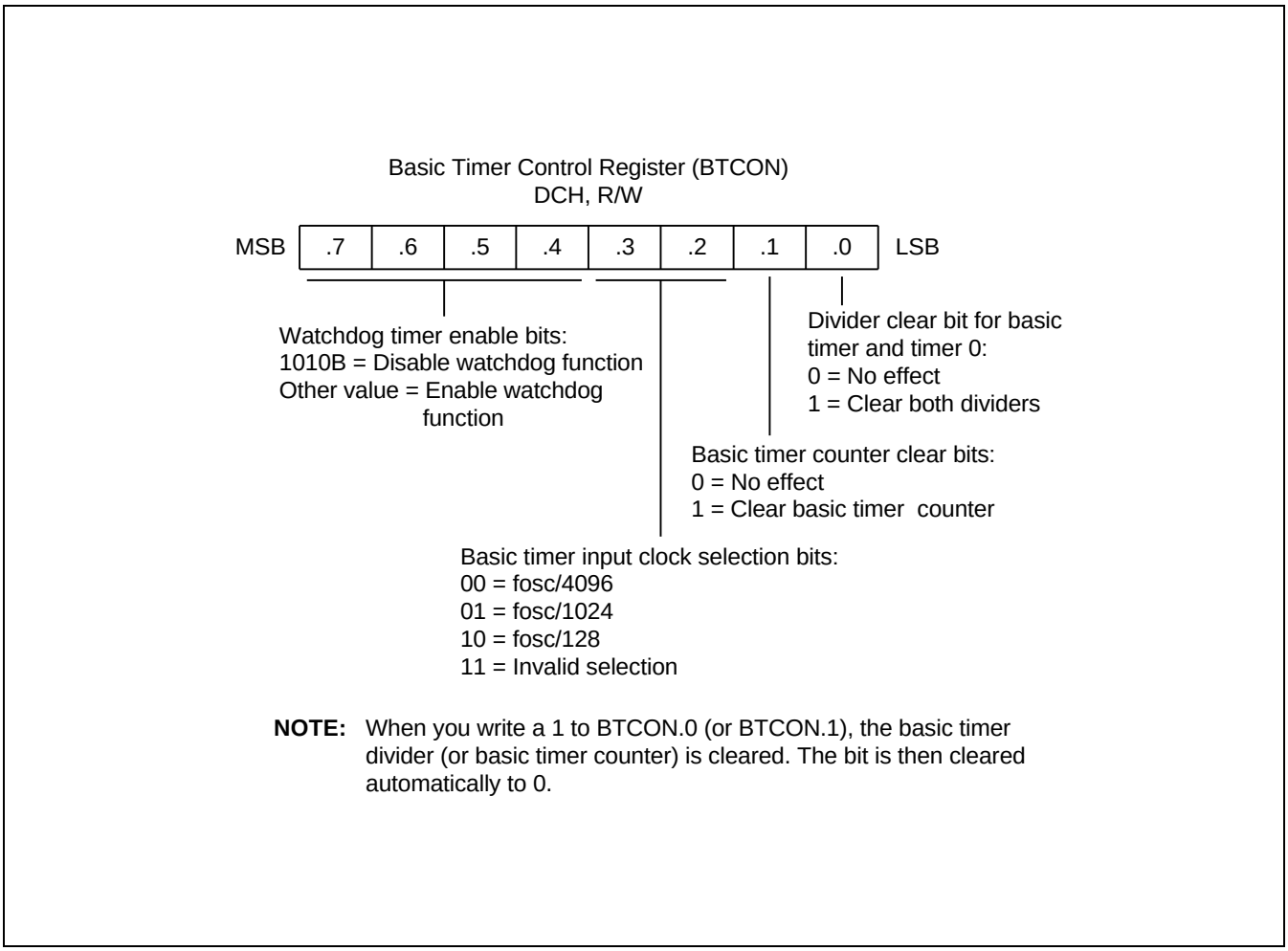
BASIC TIMER (BT)

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 both the basic timer input clock and the timer 0 clock, you write a "1" to BTCON.0.



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

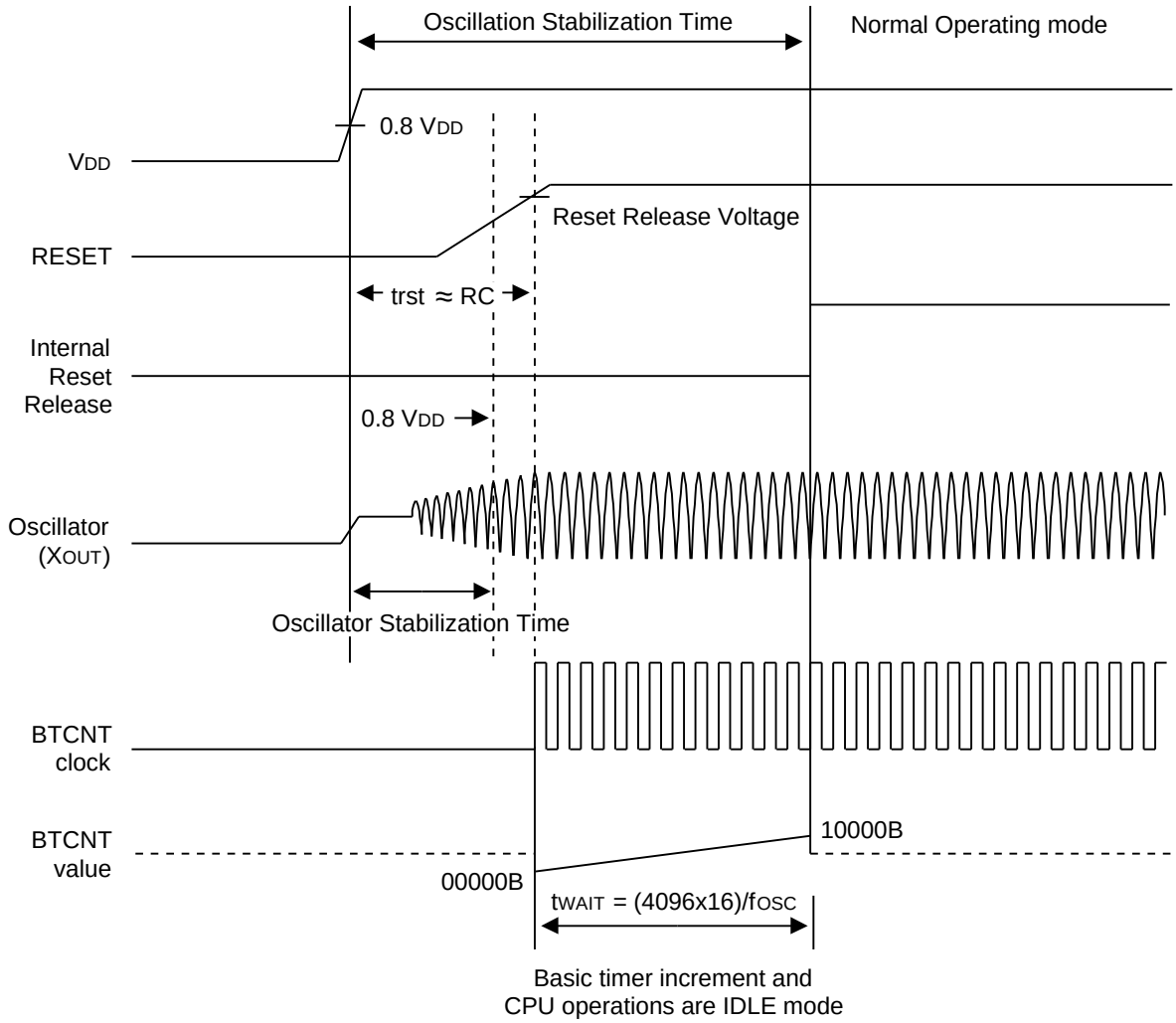
You can also use the basic timer to program a specific oscillation stabilization interval following a Reset or when Stop mode has been released by an external interrupt.

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:

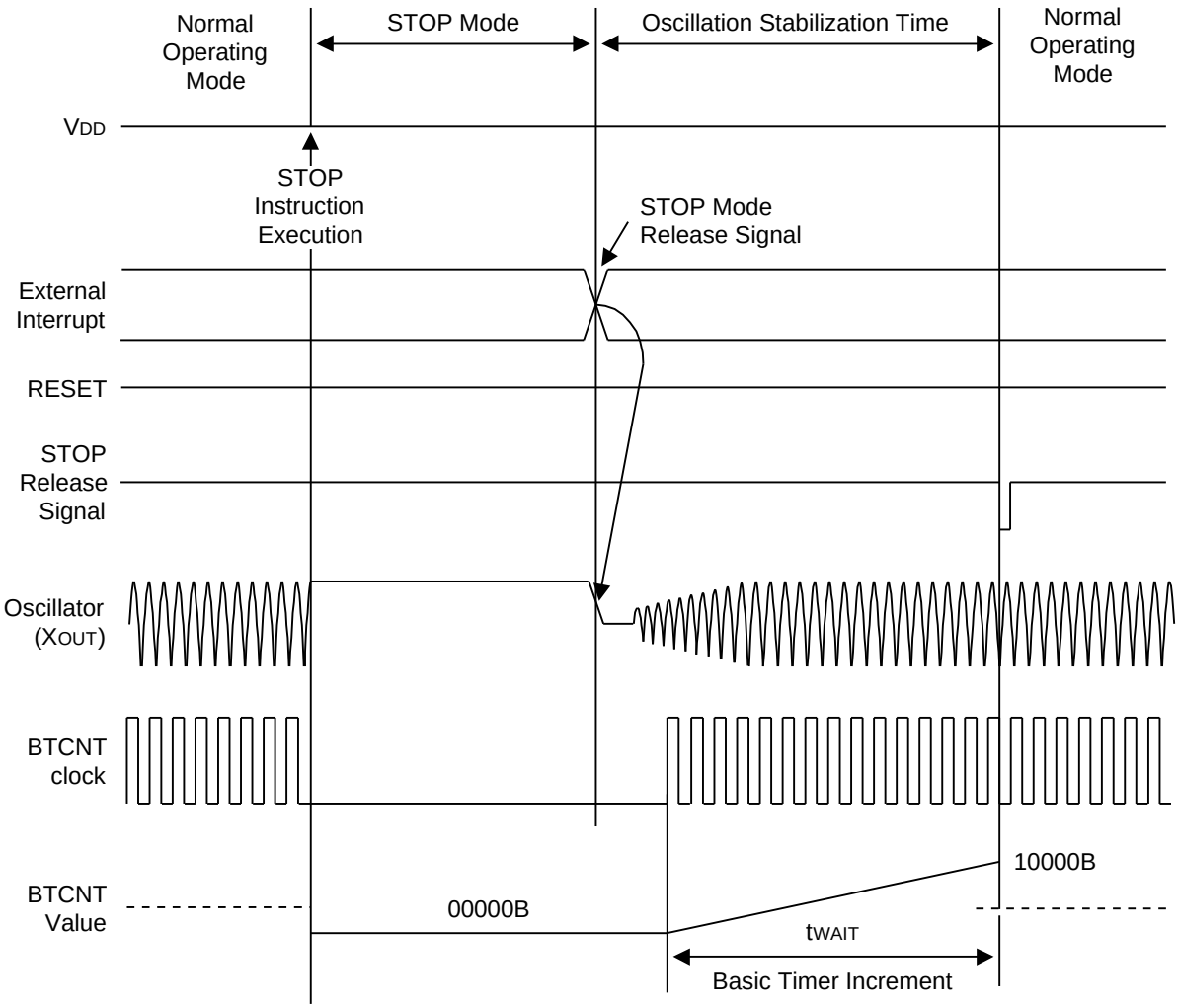
1. During Stop mode, a external power-on Reset or an external interrupt occurs to trigger the Stop mode release and oscillation starts.
2. If a external power-on Reset occurred, the basic timer counter will increase at the rate of $f_{OSC}/4096$. If an external interrupt is used to release Stop mode, 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, t WAIT, 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, t WAIT, 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 10 MHz)
0	0	(4096 x 16)/fosc	6.55 ms
0	1	(1024 x 16)/fosc	1.64 ms
1	0	(128 x 16)/fosc	0.2 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_9454        ;  S3C9454B/F9454B has only one interrupt vector

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

                ORG     003CH
                DB      00H                 ;  003CH, must be initialized to 0
                DB      00H                 ;  003DH, must be initialized to 0
                DB      0E7H                ;  003EH, enable LVR (2.3 V)
                DB      03H                 ;  003FH, internal RC (3.2 MHz in VDD = 5 V)

;--------------<< 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_9454:       •                           ;  Interrupt enable bit and pending bit check
                •                           ;
                •                           ;  Pending bit clear
                IRET                        ;
                •
                •
                END                         ;
```

TIMER 0

TIMER 0 CONTROL REGISTERS (T0CON)

The timer 0 control register, T0CON, is used to select the timer 0 operating mode (interval timer) and input clock frequency, to clear the timer 0 counter, and to enable the T0 match interrupt. It also contains a pending bit for T0 match interrupts.

A Reset clears T0CON to "00H". This sets timer 0 to normal interval timer mode, selects an input clock frequency of f_{OSC} /4096, and disables the T0 match interrupts. The T0 counter can be cleared at any time during normal operation by writing a "1" to T0CON.3.

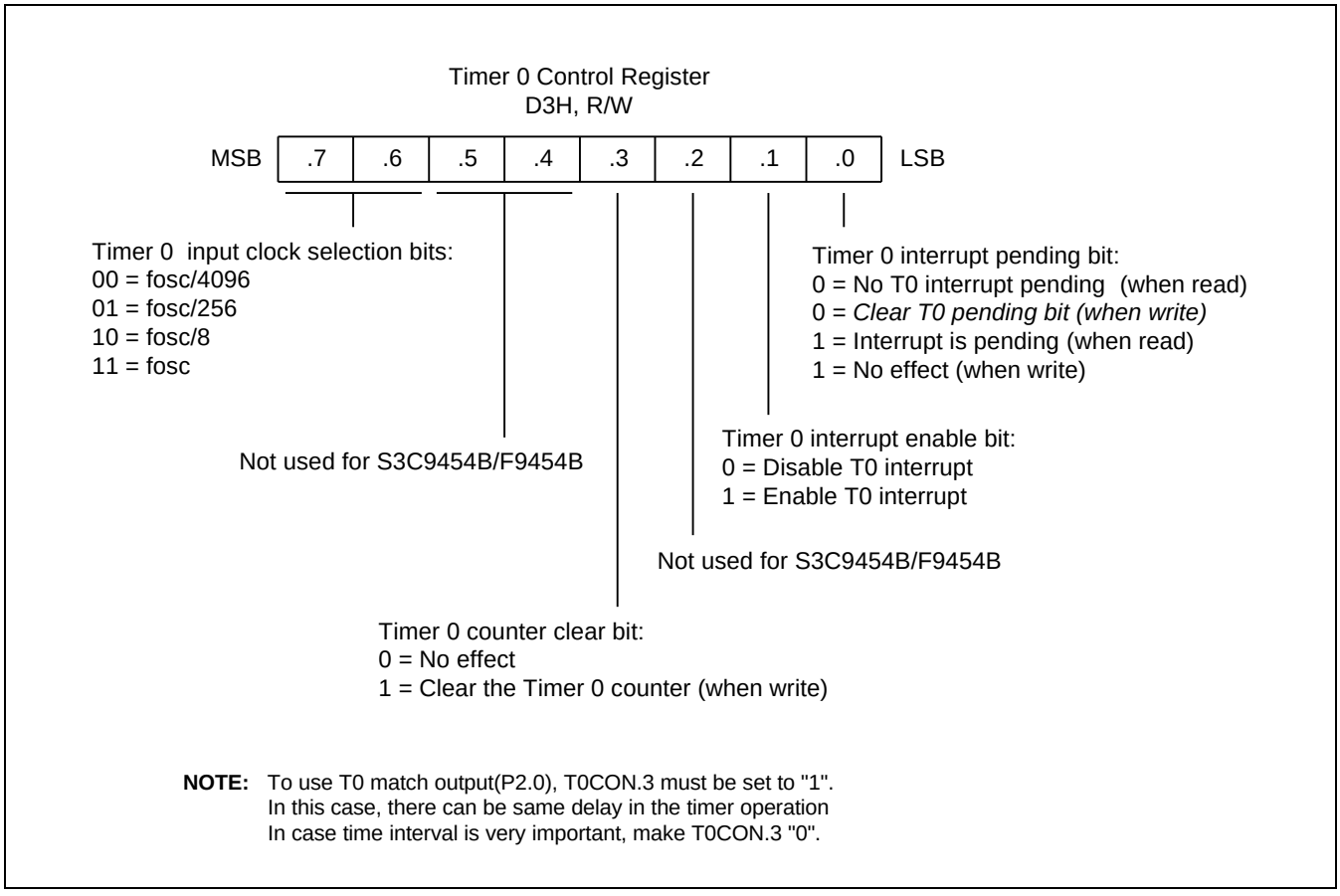
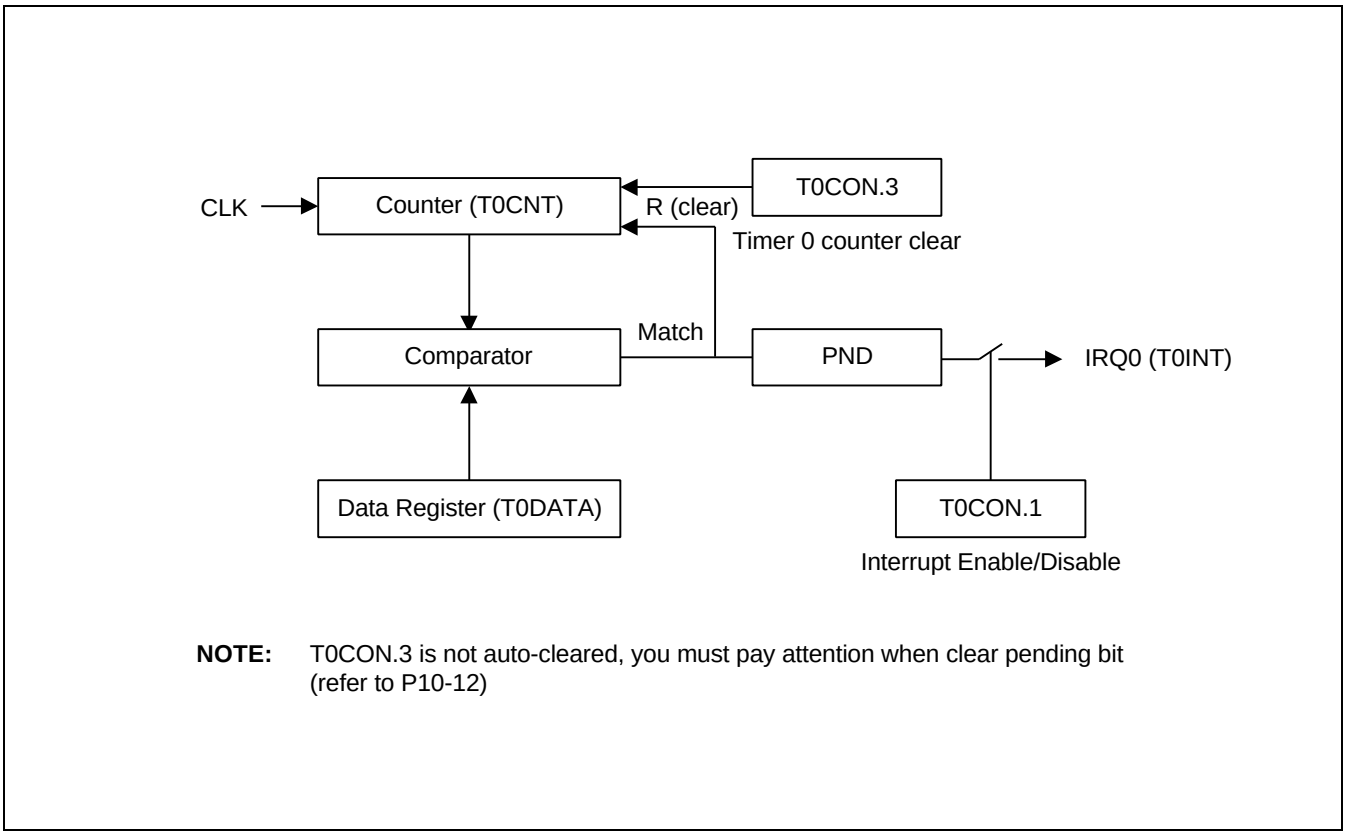


Figure 10-4. Timer 0 Control Registers (T0CON)

TIMER 0 FUNCTION DESCRIPTION

Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the Timer 0 reference data register, T0DATA. The match signal generates a Timer 0 match interrupt (T0INT, vector 00H) and then clears the counter. If, for example, you write the value "10H" to T0DATA, the counter will increment until it reaches "10H". At this point, the Timer 0 interrupt request is generated, the counter value is reset and counting resumes.



NOTE: T0CON.3 is not auto-cleared, you must pay attention when clear pending bit (refer to P10-12)

Figure 10-5. Simplified Timer 0 Function Diagram (Interval Timer Mode)

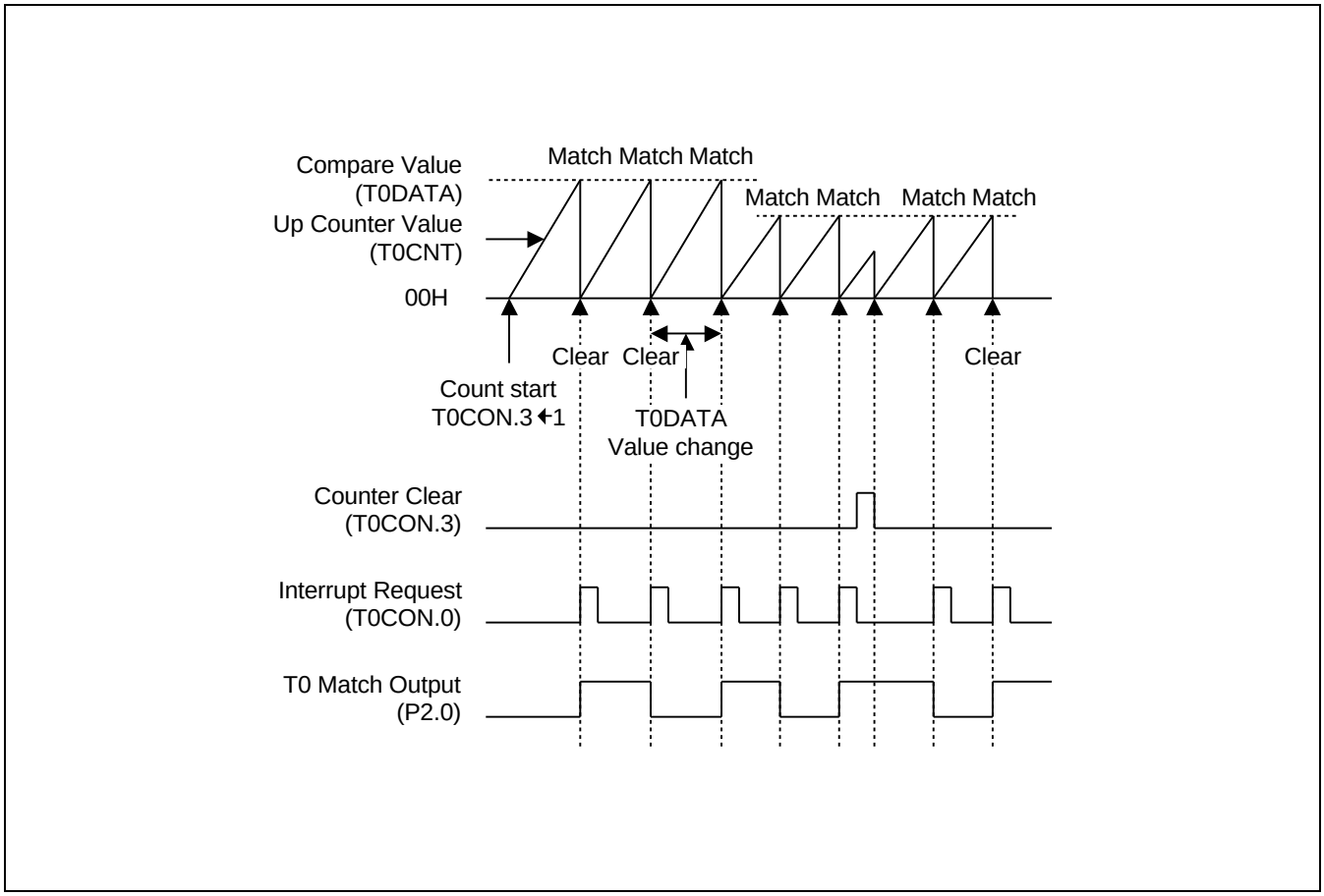
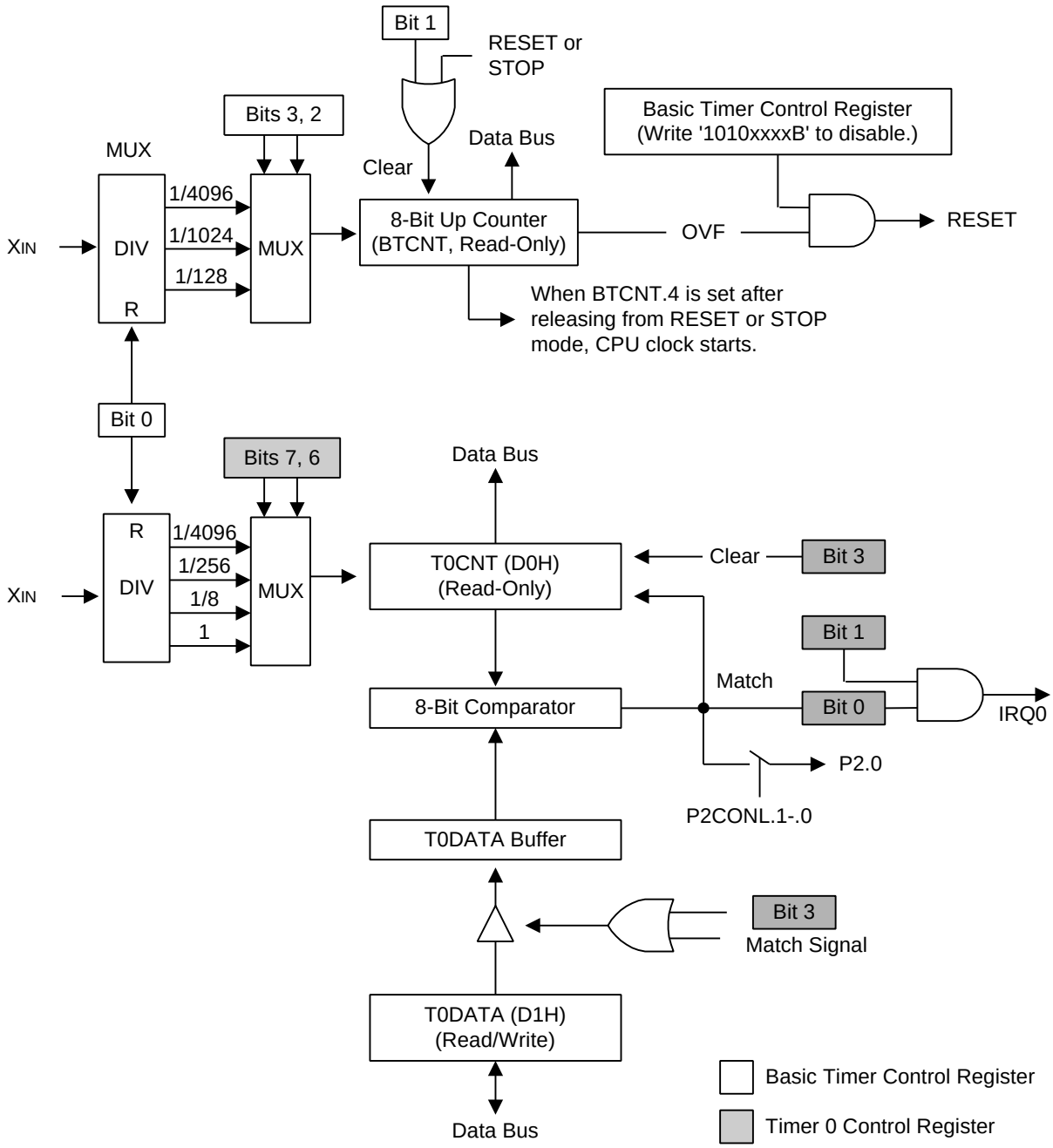


Figure 10-6. Timer 0 Timing Diagram



NOTE: During a power-on Reset operation, the CPU is idle during the required oscillation stabilization interval (until bit 4 the basic timer counter is set).

Figure 10-7. Basic Timer and Timer 0 Block Diagram

☞ PROGRAMMING TIP1 – Configuring Timer 0 (Interval Mode)

The following sample program sets Timer 0 to interval timer mode.

```
            ORG     0000H
            VECTOR  00H,INT_9454        ;  S3C9454B/F9454B has only one interrupt vector

            ORG     003CH
            DB      00H                 ;  003CH, must be initialized to 0
            DB      00H                 ;  003DH, must be initialized to 0
            DB      0E7H                ;  003EH, enable LVR (2.3 V)
            DB      03H                 ;  003FH, internal RC (3.2 MHz in VDD = 5 V)

            ORG     0100H

RESET:      DI                          ;  Disable interrupt
            LD      BTCON,#10100011B    ;  Watchdog disable
            LD      CLKCON,#00011000B   ;  Select non-divided CPU clock
            LD      SP,#0C0H            ;  Set stack pointer
            LD      P0CONH,#10101010B   ;
            LD      P0CONL,#10101010B   ;  P0.0–0.7 push-pull output
            LD      P1CON,#00001010B    ;  P1.0–P1.1 push-pull output
            LD      P2CONH,#01001010B   ;
            LD      P2CONL,#10101010B   ;  P2.0–P2.6 push-pull output

;--------------<< Timer 0 settings >>

            LD      T0DATA,#50H         ;  CPU = 3.2 MHz, interrupt interval = 4 msec
            LD      T0CON,#01001010B    ;  fOSC/256, Timer 0 interrupt enable
            •
            •
            •
            EI                          ;  Enable interrupt

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

MAIN:       NOP                         ;  Start main loop
            •
            •
            •
            CALL    LED_DISPLAY         ;  Sub-block module
            •
            •
            •
            CALL    JOB                 ;  Sub-block module
            •
            •
            •
            JR      T,MAIN              ;
```

☞ PROGRAMMING TIP1 – Configuring Timer 0 (Interval Mode) (Continued)

```
LED_DISPLAY:    NOP                          ;
                •                            ;
                •                            ;
                •                            ;
                RET                          ;

JOB:            NOP                          ;
                •                            ;
                •                            ;
                •                            ;
                RET                          ;

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

INT_9454:       TM      T0CON,#00000010B     ;  Interrupt enable check
                JR      Z,NEXT_CHK1          ;

                TM      T0CON,#00000001B     ;  If timer 0 interrupt was occurred,
                JP      NZ,INT_TIMER0        ;  T0CON.0 bit would be set.

NEXT_CHK1:      •                            ;  Interrupt enable bit and pending bit check
                •                            ;
                •                            ;
                IRET                         ;

INT_TIMER0:                                  ;  Timer 0 interrupt service routine

                •
                •
                •
                AND     T0CON,#1111_0_110B   ;   Pending bit clear
                IRET                         ;
                •
                •
                END                          ;
```

11 8-BIT PWM (PULSE WIDTH MODULATION)

OVERVIEW

This microcontroller has the 8-bit PWM circuit. The operation of all PWM circuit is controlled by a single control register, PWMCON.

The PWM counter is a 8-bit incrementing counter. It is used by the 8-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 $f_{OSC}/64$, $f_{OSC}/8$, $f_{OSC}/2$, $f_{OSC}/1$.

FUNCTION DESCRIPTION

PWM

The 8-bit PWM circuits have the following components:

— 6-bit comparator and extension cycle circuit

— 6-bit reference data register (PWMDATA.7–.2)

— 2-bit extension data register (PWMDATA.1–.0)

— PWM output pins (P0.6/PWM)

PWM Counter

To determine the PWM module's base operating frequency, the upper 6-bits of counter is compared to the PWM data (PWMDATA.7–.2). In order to achieve higher resolutions, the lower 2-bits of the counter can be used to modulate the "stretch" cycle. To control the "stretching" of the PWM output duty cycle at specific intervals, the lower 2-bits of counter value is compared with the PWMDATA.1–.0.

PWM Data and Extension Registers

PWM (duty) data registers, located in F2H, determine the output value generated by each 8-bit PWM circuit.

To program the required PWM output, you load the appropriate initialization values into the 6-bit reference data register (PWMDATA.7–.2) and the 2-bit extension data register (PWMDATA.1–.0). 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 PWM output is based on the f_{OSC} clock frequency. The PWM counter clock value is determined by the setting of PWMCON.6–.7.

Table 11-1. PWM Control and Data Registers

Register Name	Mnemonic	Address	Function
PWM data registers	PWMDATA.7–.2	F2H.7–.2	6-bit PWM basic cycle frame value
	PWMDATA.1–.0	F2H.1–.0	2-bit extension ("stretch") value
PWM control registers	PWMCON	F3H	PWM counter stop/start (resume), and PWM counter clock settings

PWM Function Description

The PWM output signal toggles to Low level whenever the lower 6-bit of counter matches the reference data register (PWMDATA.7–.2). If the value in the PWMDATA.7–.2 register is not zero, an overflow of the lower 6-bits of counter causes the PWM output to toggle to High level. In this way, the reference value written to the reference data register determines the module's base duty cycle.

The value in the upper 2-bits of counter is compared with the extension settings in the 2-bit extension data register (PWMDATA.1–.0). This lower 2-bits of counter value, together with extension logic and the PWM module's extension data 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 11-2).

If, for example, the value in the extension data register is '01B', the 2nd cycle will be one pulse longer than the other 3 cycles. If the base duty cycle is 50 %, the duty of the 2nd cycle will therefore be "stretched" to approximately 51% duty. For example, if you write 10B to the extension data register, all odd-numbered pulses will be one cycle longer. If you write 11H to the extension data register, all pulses will be stretched by one cycle except the 4th 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 11-2. PWM output "stretch" Values for Extension Data Register (PWMDATA.1–.0)

PWMDATA Bit (Bit1–Bit0)	"Stretched" Cycle Number
00	–
01	2
10	1, 3
11	1, 2, 3

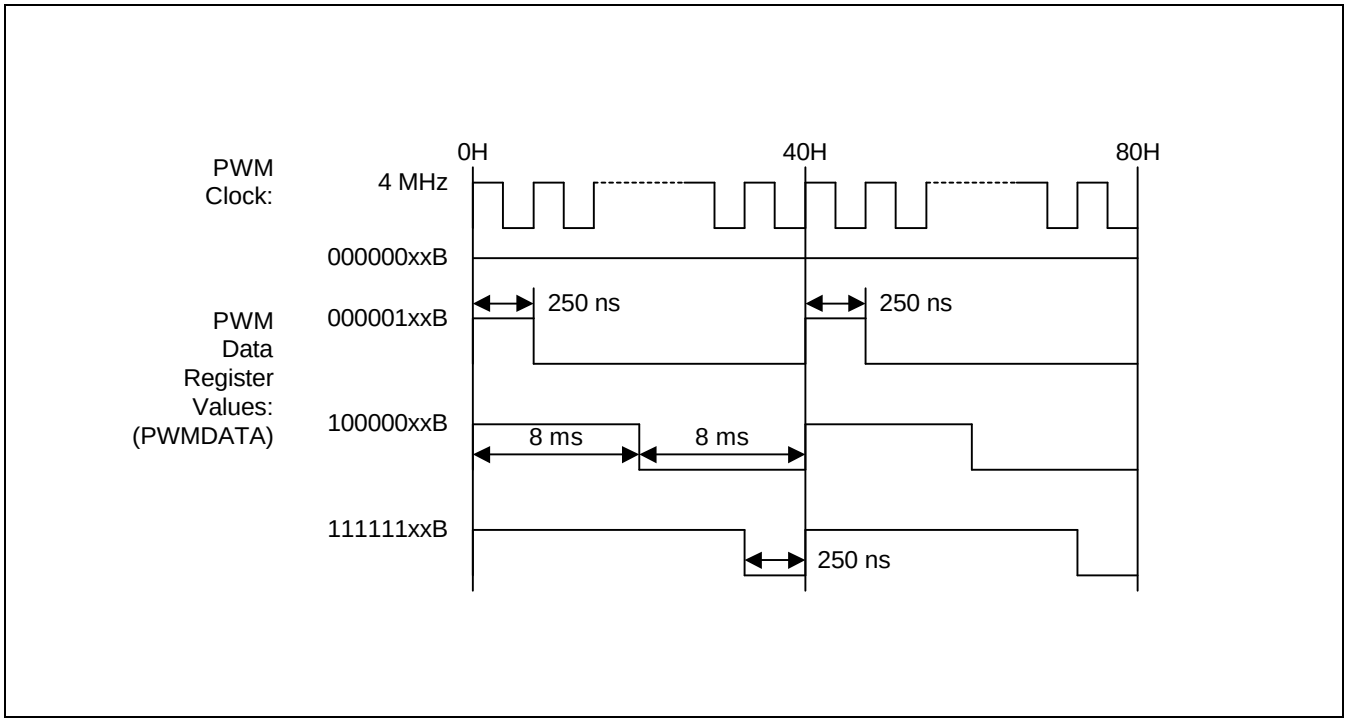


Figure 11-1. 8-Bit PWM Basic Waveform

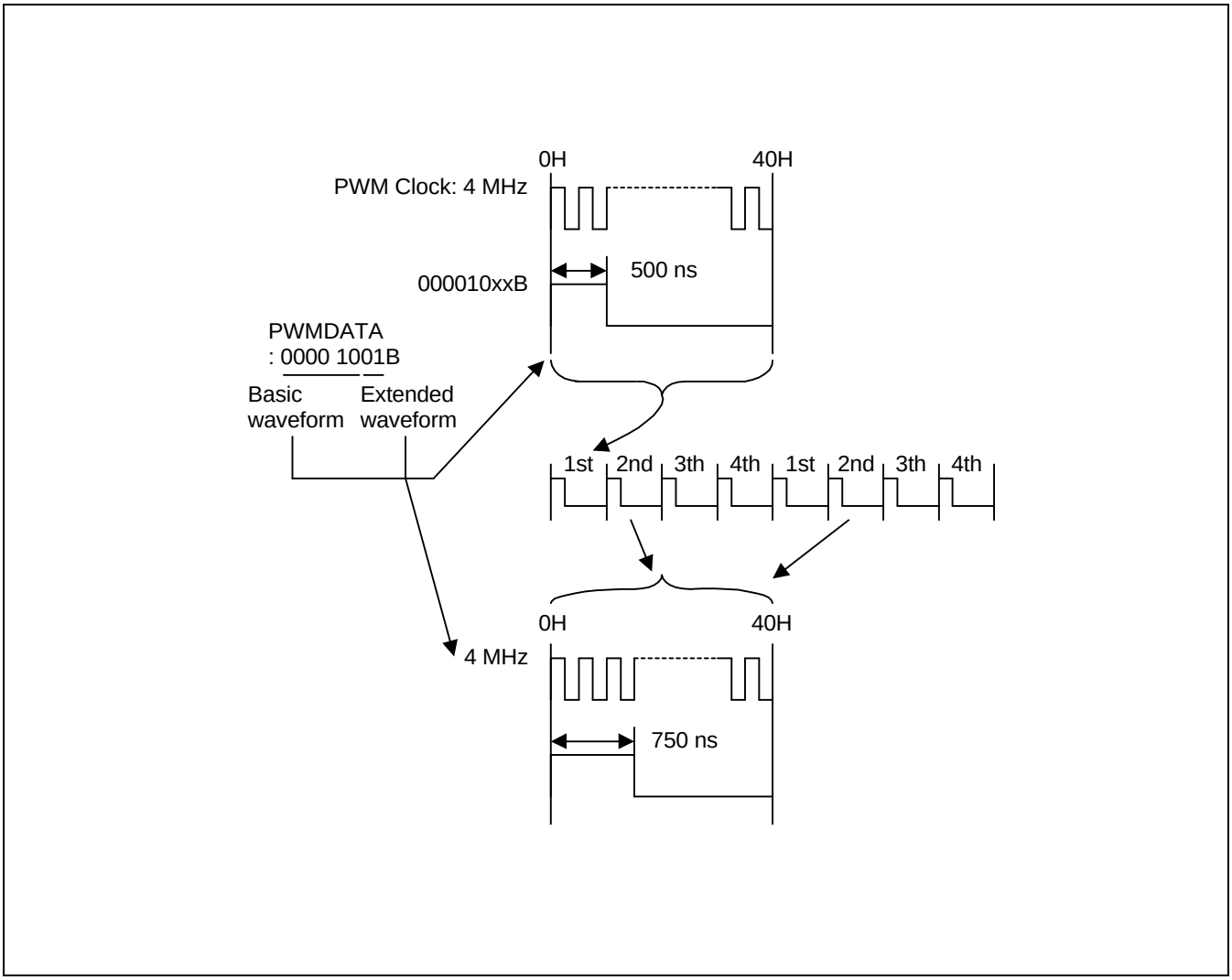


Figure 11-2. 8-Bit Extended PWM Waveform

PWM CONTROL REGISTER (PWMCON)

The control register for the PWM module, PWMCON, is located at register address F3H. PWMCON is used the 8-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 (8-bit counter overflow) interrupt control

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

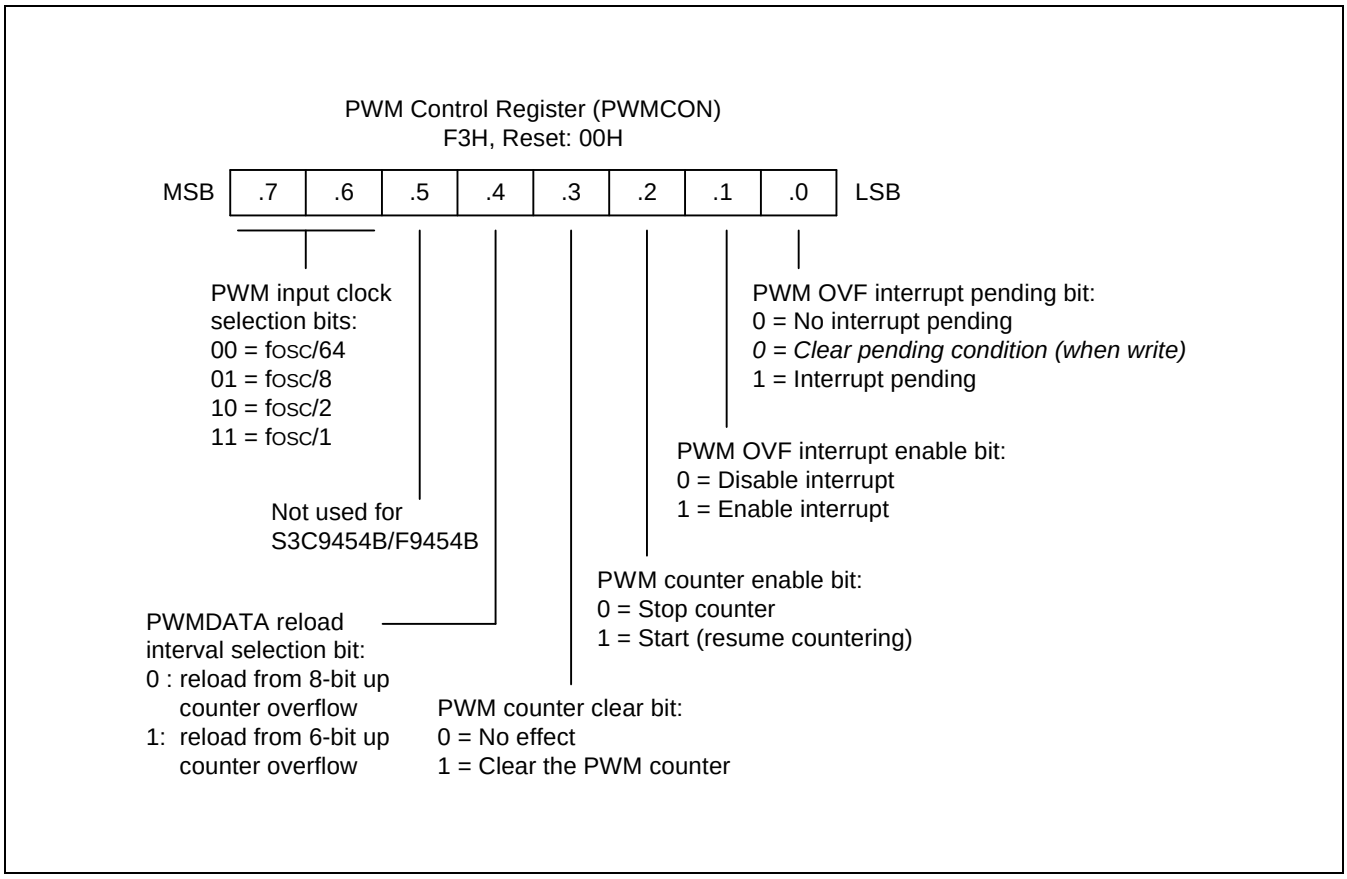


Figure 11-3. PWM Control Register (PWMCON)

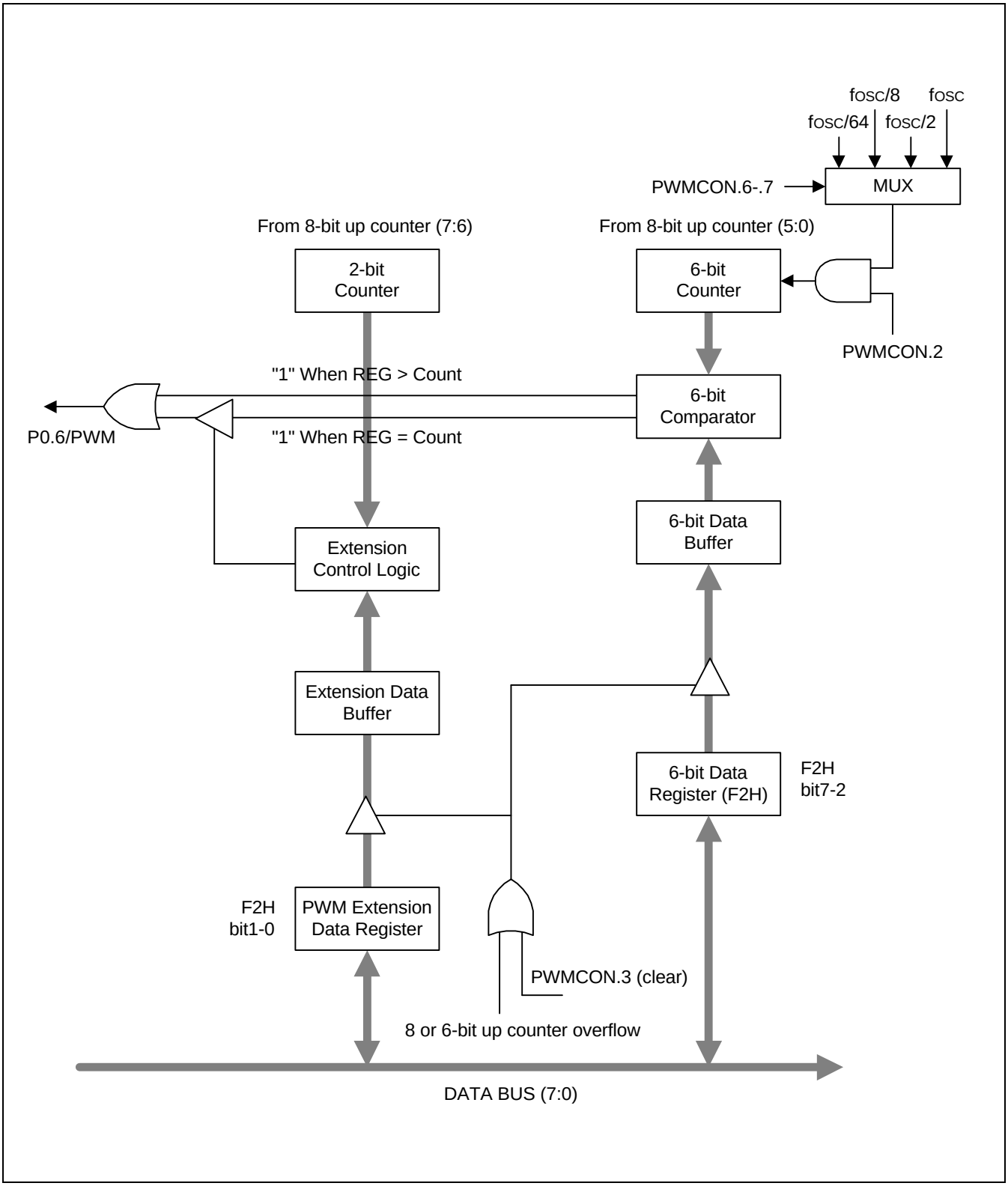


Figure 11-4. PWM Functional Block Diagram

☞ PROGRAMMING TIP — Programming the PWM Module to Sample Specifications

```
;--------------<< Interrupt Vector Address >>

ORG         0000H
            VECTOR    00H,INT_9454          ;  S3C9454/F9454 has only one interrupt vector

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

ORG         003CH
            DB        00H                   ;  003CH, must be initialized to 0.
            DB        00H                   ;  003DH, must be initialized to 0.
            DB        0E7H                  ;  003EH, enable LVR (2.3 V)
            DB        03H                   ;  003FH, internal RC (3.2 MHz in VDD = 5 V)

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

            ORG       0100H
RESET:      DI                              ;  disable interrupt
            LD        BTCON,#10100011B      ;  Watchdog disable
            •
            •
            LD        P0CONH,#10011010B     ;  Configure P0.6 PWM output
            LD        PWMCON,#00000110B     ;  fOSC/64, counter/interrupt enable
            LD        PWMDATA,#80H          ;
            •
            •
            EI                              ;  Enable interrupt

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

MAIN:                                       ;
            •                               ;
            •                               ;
            •                               ;
            •                               ;
            JR        t,MAIN                ;

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

INT_9454:   •                               ;  Interrupt enable bit and pending bit check
            TM        PWMCON,#00000010B     ;  Interrupt enable check
            JR        Z,NEXT_CHK1           ;
            TM        PWMCON,#00000001B     ;  Interrupt pending bit check
            JP        NZ,INT_PWM            ;  PWMCON's pending bit set --> PWM interrupt

NEXT_CHK1:  •
            •
            •
            IRET                            ;
```

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

```
INT_PWM:                                    ;  PWM interrupt service routine
                •
                •
                •
                AND       PWMCON,#11110110B ;  pending bit clear
                IRET                        ;
                •
                •
                END
```

12 A/D 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 V_{DD} and V_{SS} values. The A/D converter has the following components:

— Analog comparator with successive approximation logic

— D/A converter logic

— ADC control register (ADCON)

— Nine multiplexed analog data input pins (ADC0–ADC8)

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

To initiate an analog-to-digital conversion procedure, you write the channel selection data in the A/D converter control register ADCON to select one of the nine analog input pins (ADCn, n = 0–8) and set the conversion start or enable bit, ADCON.0. The read-write ADCON register is located at address F7H.

During a normal conversion, ADC logic initially sets the successive approximation register to 200H (the approximate half-way point of an 10-bit register). This register is then updated automatically during each conversion step. The successive approximation block performs 10-bit conversions for one input channel at a time. You can dynamically select different channels by manipulating the channel selection bit value (ADCON.7–4) in the ADCON register. To start the A/D conversion, you should set a the enable bit, ADCON.0. When a conversion is completed, ACON.3, the end-of-conversion (EOC) bit is automatically set to 1 and the result is dumped into the ADDATA register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATA before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

NOTE

Because the ADC does not use sample-and-hold circuitry, it is important that any fluctuations in the analog level at the ADC0–ADC8 input pins during a conversion procedure be kept to an absolute minimum. Any change in the input level, perhaps due to circuit noise, will invalidate the result.

USING A/D PINS FOR STANDARD DIGITAL INPUT

The ADC module's input pins are alternatively used as digital input in port 0 and P2.6.

A/D CONVERTER CONTROL REGISTER (ADCON)

The A/D converter control register, ADCON, is located at address F7H. ADCON has four functions:

— Bits 7-4 select an analog input pin (ADC0–ADC8).
— Bit 3 indicates the status of the A/D conversion.
— Bits 2-1 select a conversion speed.
— Bit 0 starts the A/D conversion.

Only one analog input channel can be selected at a time. You can dynamically select any one of the ten analog input pins (ADC0–ADC8) by manipulating the 4-bit value for ADCON.7–ADCON.4.

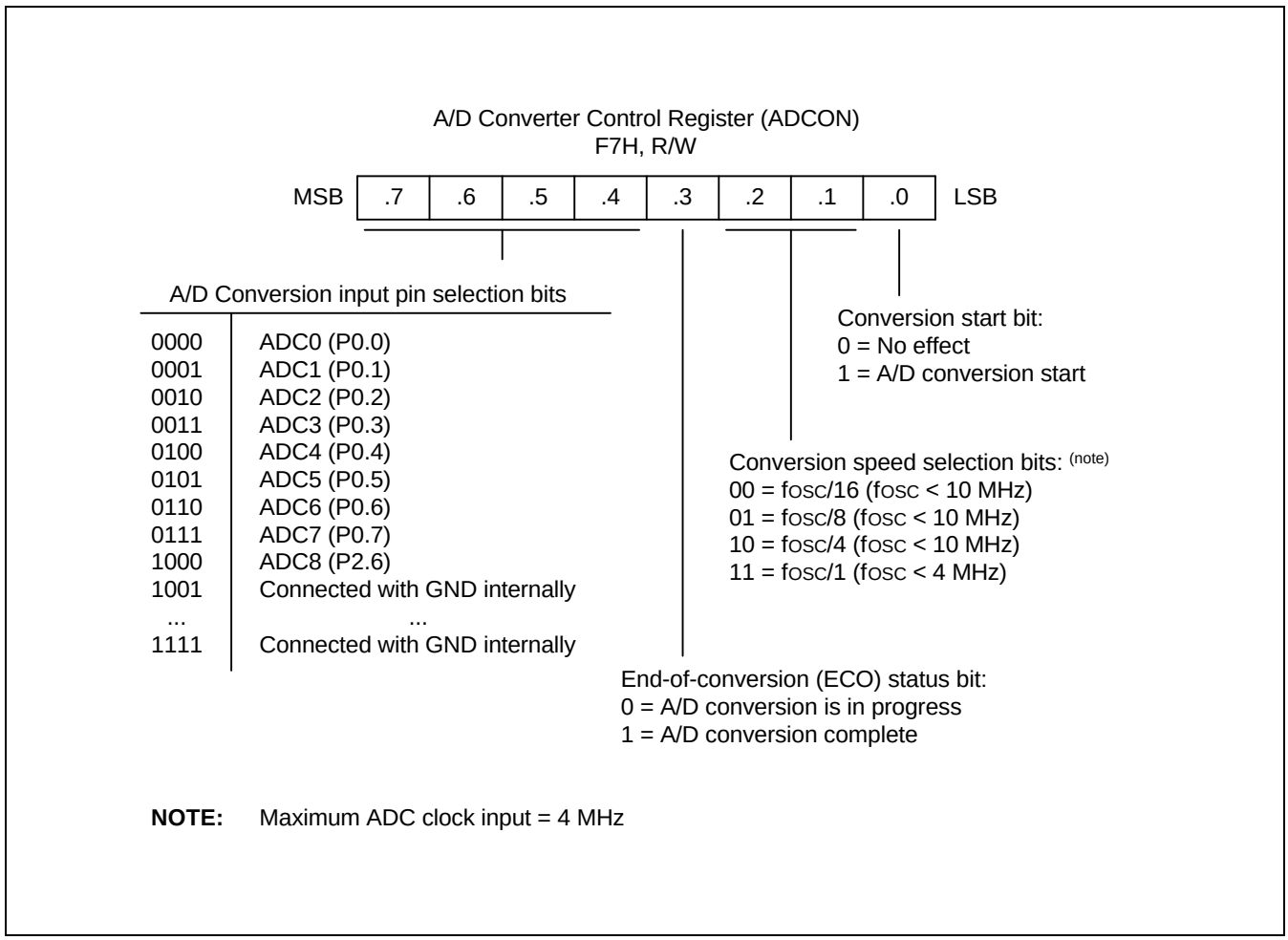


Figure 12-1. A/D Converter Control Register (ADCON)

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 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 bit conversion is always 1/2 V_{DD}.

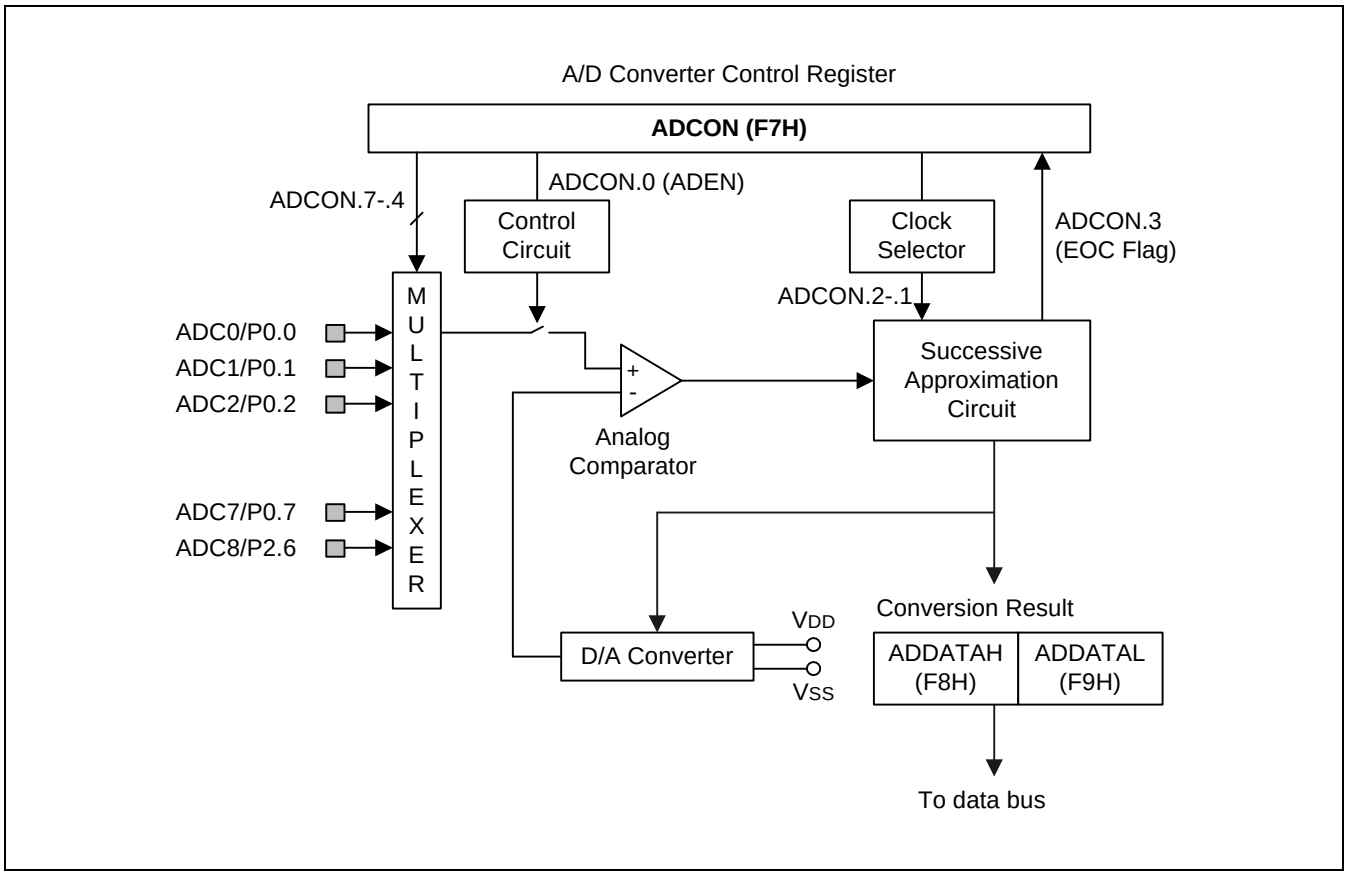


Figure 12-2. A/D Converter Circuit Diagram

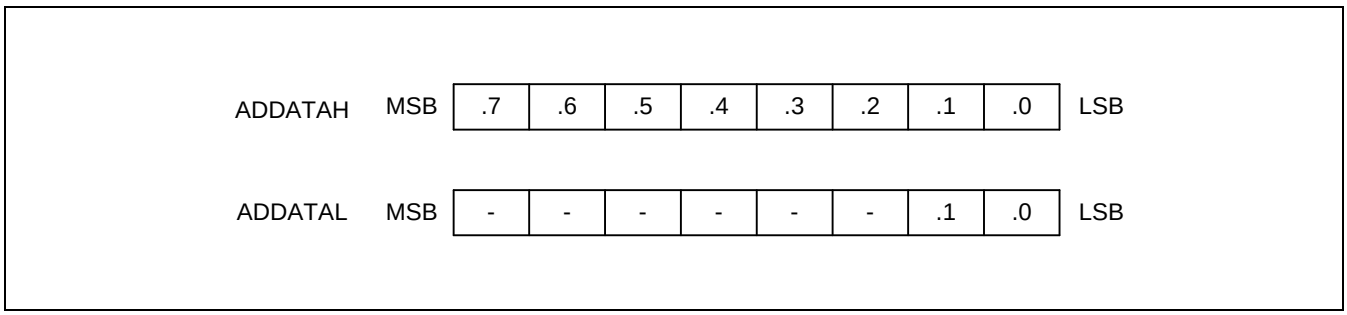


Figure 12-3. A/D Converter Data Register (ADDATAH/L)

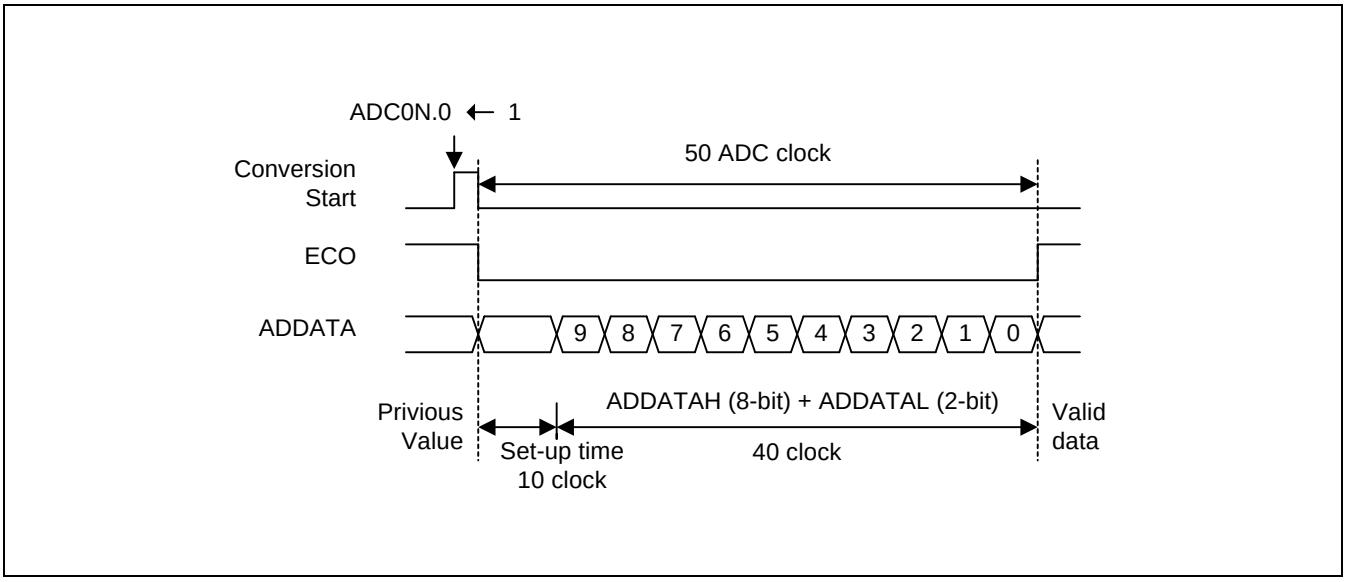


Figure 12-4. A/D Converter Timing Diagram

CONVERSION TIMING

The A/D conversion process requires 4 steps (4 clock edges) to convert each bit and 10 clocks to step-up A/D conversion. Therefore, total of 50 clocks are required to complete an 10-bit conversion: With an 10 MHz CPU clock frequency, one clock cycle is 400 ns (4/fosc). If each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

4 clocks/bit × 10-bits + step-up time (10 clock) = 50 clocks
50 clock × 400 ns = 20 μs at 10 MHz, 1 clock time = $4/f_{OSC}$ (assuming ADCON.2–.1 = 10)

INTERNAL A/D CONVERSION PROCEDURE

1. Analog input must remain between the voltage range of V_{SS} and V_{DD}.
2. Configure the analog input pins to input mode by making the appropriate settings in P0CONH, P0CONL and P2CONH registers.
3. Before the conversion operation starts, you must first select one of the nine input pins (ADC0–ADC8) by writing the appropriate value to the ADCON register.
4. When conversion has been completed, (50 clocks have elapsed), the EOC 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.

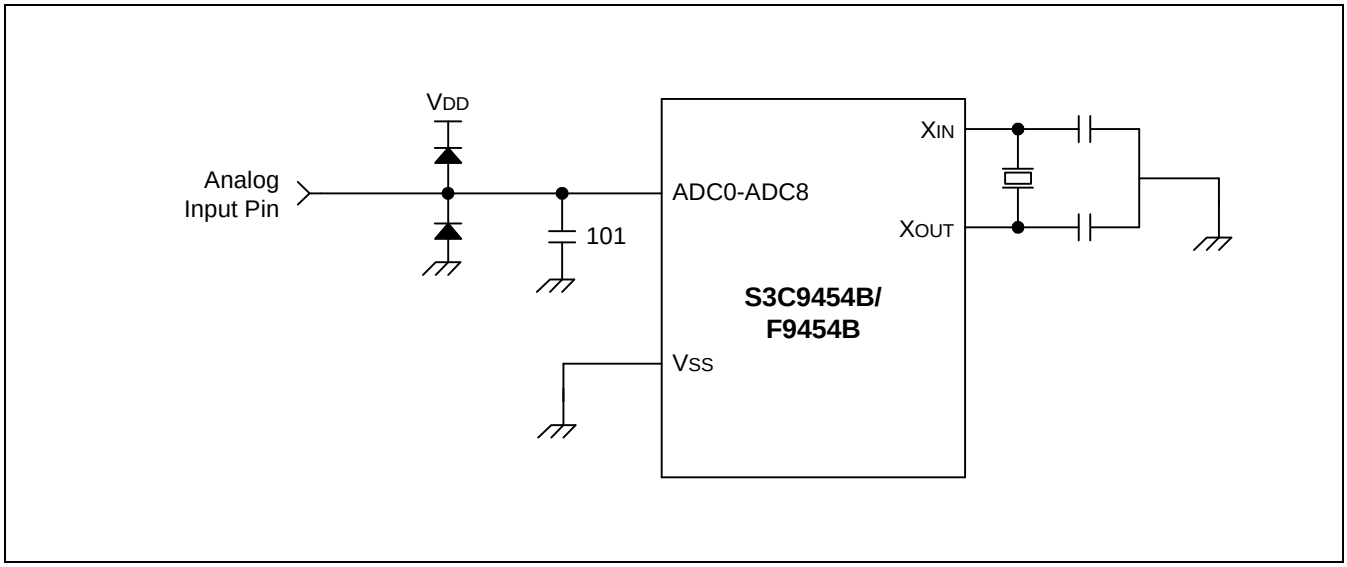


Figure 12-5. Recommended A/D Converter Circuit for Highest Absolute Accuracy

☞ PROGRAMMING TIP – Configuring A/D Converter

```
                ORG     0000H
                VECTOR  00H,INT_9454        ;  S3C9454/F9454 has only one interrupt vector

ORG             003CH
                DB      00H                 ;  003CH, must be initialized to 0
                DB      00H                 ;  003DH, must be initialized to 0
                DB      0E7H                ;  003EH, enable LVR (2.3 V)
                DB      03H                 ;  003FH, internal RC (3.2 MHz in VDD = 5 V)

                ORG     0100H
RESET:          DI                          ;  disable interrupt
                LD      BTCON,#10100011B    ;  Watchdog disable
                •
                •
                •
                LD      P0CONH,#11111111B   ;  Configure P0.4–P0.7 AD input
                LD      P0CONL,#11111111B   ;  Configure P0.0–P0.3 AD input
                LD      P2CONH,#00100000B   ;  Configure P2.6 AD input
                EI                          ;  Enable interrupt

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

MAIN:           •
                •
                •
                CALL    AD_CONV             ;  Subroutine for AD conversion
                •
                •
                •
                JR      t,MAIN              ;

AD_CONV:        LD      ADCON,#00000001B    ;  Select analog input channel → P0.0
                                            ;  select conversion speed → fOSC/16
                                            ;  set conversion start bit

                NOP
                NOP                         ;  If you select conversion speed to fOSC/16
                NOP                         ;  at least three nop must be included
```

☞ **PROGRAMMING TIP – Configuring A/D Converter (Continued)**

```
CONV_LOOP:  TM      ADCON,#00001000B    ;  Check EOC flag
            JR      Z,CONV_LOOP         ;  If EOC flag=0, jump to CONV_LOOP until EOC flag=1
            LD      R0,ADDATAH          ;  High 8 bits of conversion result are stored
                                        ;  to ADDATAH register
            LD      R1,ADDATAL          ;  Low 2 bits of conversion result are stored
                                        ;  to ADDATAL register
            LD      ADCON,#00010011B    ;  Select analog input channel → P0.1
                                        ;  Select conversion speed → fOSC/8
                                        ;  Set conversion start bit
CONV_LOOP2: TM      ADCON,#00001000B    ;  Check EOC flag
            JR      Z,CONV_LOOP2
            LD      R2,ADDATAH
            LD      R3,ADDATAL
            •
            •
            •
            RET                         ;
INT_9454:   •                           ;  Interrupt enable bit and pending bit check
            •                           ;
            •                           ;  Pending bit clear
            IRET                        ;
            •
            •
            END
```

NOTES

13 ELECTRICAL DATA

OVERVIEW

In this section, the following S3C9454B/F9454B electrical characteristics are presented in tables and graphs:

— Absolute maximum ratings
— D.C. electrical characteristics
— A.C. electrical characteristics
— Input timing measurement points
— Oscillator characteristics
— Oscillation stabilization time
— Operating voltage range
— Schmitt trigger input characteristics
— Data retention supply voltage in stop mode
— Stop mode release timing when initiated by a RESET
— A/D converter electrical characteristics
— LVR circuit characteristics
— LVR reset timing

Table 13-1. 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 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
		All I/O pins active	+ 150	
Operating temperature	T_A	–	– 25 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	°C

Table 13-2. DC Electrical Characteristics

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Input high voltage	V_{IH1}	Ports 0, 1, 2 and RESET	V_{DD}= 2.0 to 5.5 V	0.8 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN} and X_{OUT}		V_{DD}- 0.1			
Input low voltage	V_{IL1}	Ports 0, 1, 2 and RESET	V_{DD}= 2.0 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} = – 10 mA ports 0, 1, 2	V_{DD}= 4.5 to 5.5 V	V_{DD}-1.5	V_{DD}- 0.4	–	V
Output low voltage	V_{OL}	I_{OL} = 25 mA port 0, 1, and 2	V_{DD}= 4.5 to 5.5 V	–	0.4	2.0	V
Input high leakage current	I_{LIH1}	All input except I_{LIH2}	V_{IN} = V_{DD}	–	–	1	uA
	I_{LIH2}	X_{IN}, X_{OUT}	V_{IN} = V_{DD}			20	
Input low leakage current	I_{LIL1}	All input except I_{LIL2}	V_{IN} = 0 V	–	–	–1	uA
	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	uA
Output low leakage current	I_{LOL}	All output pins	V_{OUT} = 0 V	–	–	–2	uA
Pull-up resistors	R_P	V_{IN} = 0 V, T_A=25°C Ports 0, 1, 2	V_{DD} = 5 V	25	50	100	kΩ
Pull-down resistors	R_P	V_{IN} = 0 V, T_A=25°C Ports 1	V_{DD} = 5 V	25	50	100	
Supply current	I_{DD1}	Run mode 10 MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	5	10	mA
		3 MHz CPU clock	V_{DD} = 2.0 V		2	5	
	I_{DD2}	Idle mode 10 MHz CPU clock	V_{DD} = 4.5 to 5.5 V	–	2	4	
		3 MHz CPU clock	V_{DD} = 2.0 V		0.5	1.5	
	I_{DD3}	Stop mode T_A = 25°C	V_{DD} = 4.5 to 5.5 V (LVR disable)	–	0.1	5	uA
			V_{DD} = 4.5 to 5.5 V (LVR enable)		100	200	
			V_{DD} = 2.6 V (LVR enable)		30	60	

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

Table 13-3. AC Electrical Characteristics

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

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

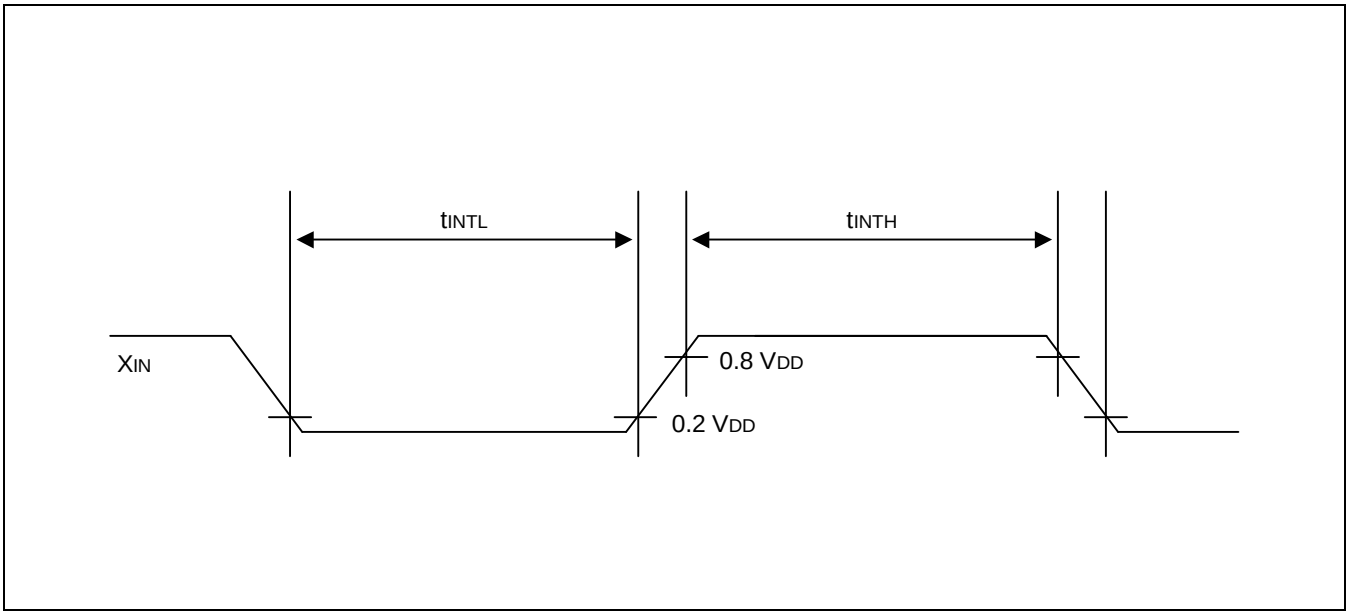


Figure 13-1. Input Timing Measurement Points

Table 13-4. Oscillator Characteristics

(T_A = – 25 °C to + 85 °C)

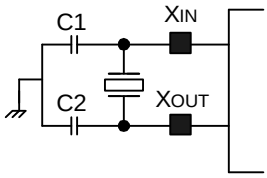
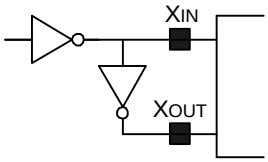
Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Main crystal or ceramic		V_{DD} = 4.5 to 5.5 V	1	–	10	MHz
		V_{DD} = 2.7 to 4.5 V	1	–	6	MHz
		V_{DD} = 2.0 to 2.7 V	1	–	3	MHz
External clock (Main System)		V_{DD} = 4.5 to 5.5 V	1	–	10	MHz
		V_{DD} = 2.7 to 4.5 V	1	–	6	MHz
		V_{DD} = 2.0 to 2.7 V	1	–	3	MHz
External RC oscillator	–	V_{DD} = 5 V	–	4	–	MHz
Internal RC oscillator	–	V_{DD} = 5 V	–	3.2		
		Tolerance:20% at T_A =25°C		0.5		

Table 13-5. Oscillation Stabilization Time

(T_A = - 25 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Main crystal	f_{OSC} > 1.0 MHz	–	–	20	ms
Main ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	10	ms
External clock (main system)	X_{IN} input high and low width (t_{XH}, t_{XL})	25	–	500	ns
Oscillator stabilization	t_{WAIT} when released by a reset (1)	–	$2^{16}/f_{OSC}$	–	ms
Wait time	t_{WAIT} when released by an interrupt (2)	–	–	–	ms

NOTES:

1. f_{OSC} 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 settings in the basic timer control register, BTCON.

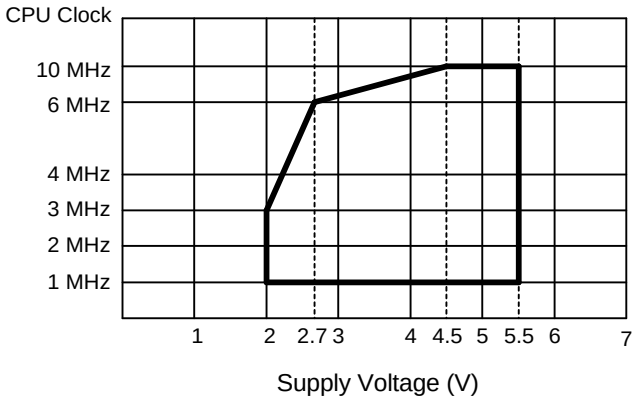


Figure 13-2. Operating Voltage Range

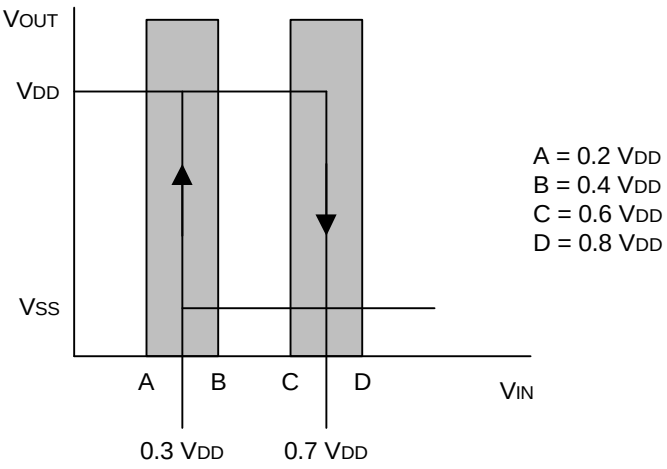


Figure 13-3. Schmitt Trigger Input Characteristics Diagram

Table 13-6. Data Retention Supply Voltage in Stop Mode

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	Stop mode	2.0	–	5.5	V
Data retention supply current	I_{DDDR}	Stop mode; V_{DDDR} = 2.0 V	–	0.1	5	uA

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

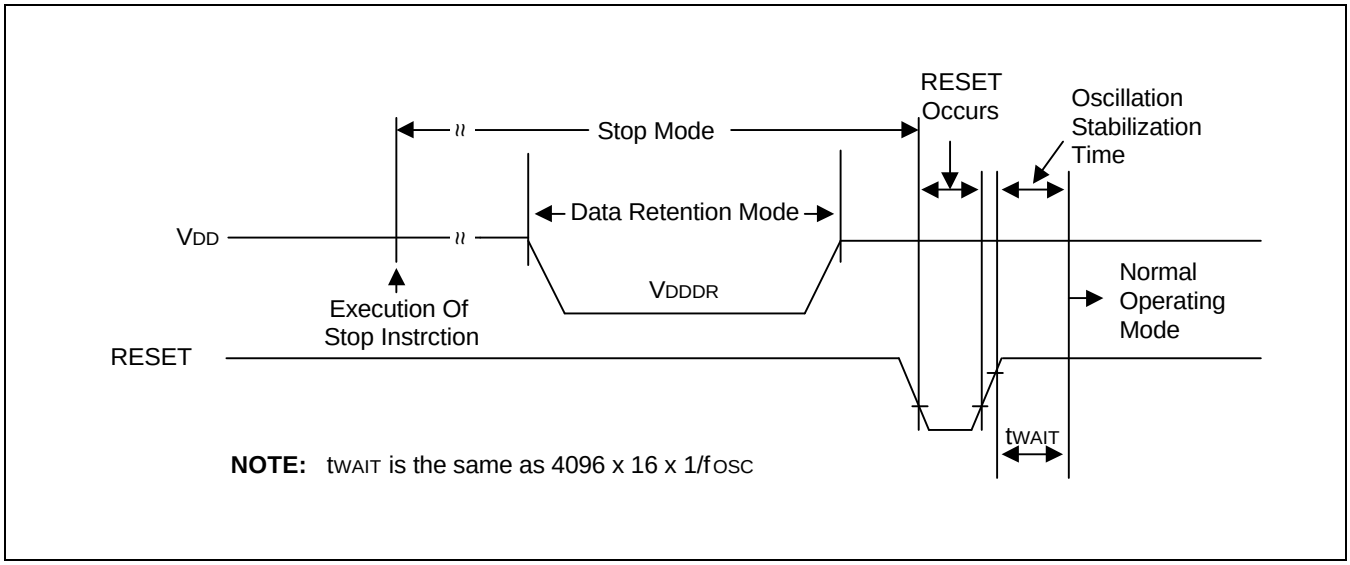


Figure 13-4. Stop Mode Release Timing When Initiated by a RESET

Table 13-7. A/D Converter Electrical Characteristics

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.0 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 = 10 MHz V_{SS} = 0 V	–	–	± 3	LSB
Integral linearity error	ILE	″	–	–	± 2	
Differential linearity error	DLE	″	–	–	± 1	
Offset error of top	EOT	″	–	± 1	± 3	
Offset error of bottom	EOB	″	–	± 1	± 2	
Conversion time (1)	t_{CON}	f_{OSC} = 10 MHz	–	20	–	µs
Analog input voltage	V_{IAN}	–	V_{SS}	–	V_{DD}	V
Analog input impedance	R_{AN}	–	2	–	–	MW
Analog input current	I_{ADIN}	V_{DD} = 5 V	–	–	10	µA
Analog block current (2)	I_{ADC}	V_{DD} = 5 V	–	1	3	mA
		V_{DD} = 3 V		0.5	1.5	
		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.

Table 13-8. LVR Circuit Characteristics

(T_A = 25 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Low voltage reset	V_{LVR}	–	2.0 3.3 3.6	2.3 3.0 3.9	2.6 3.6 4.2	V

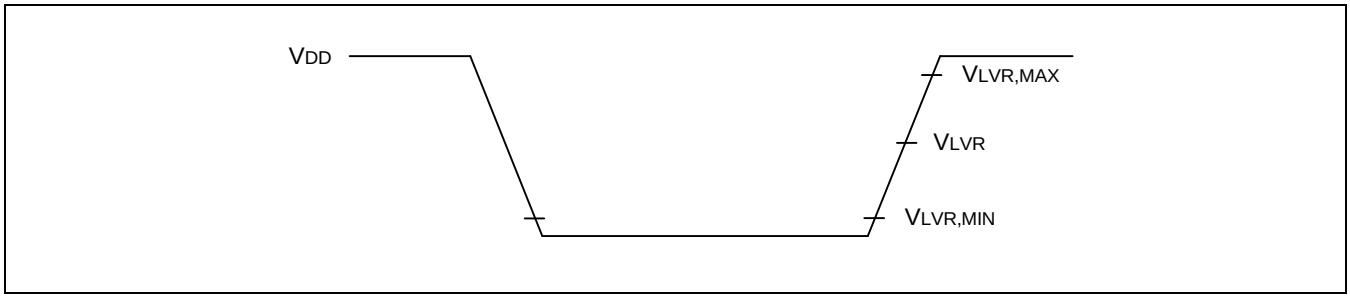


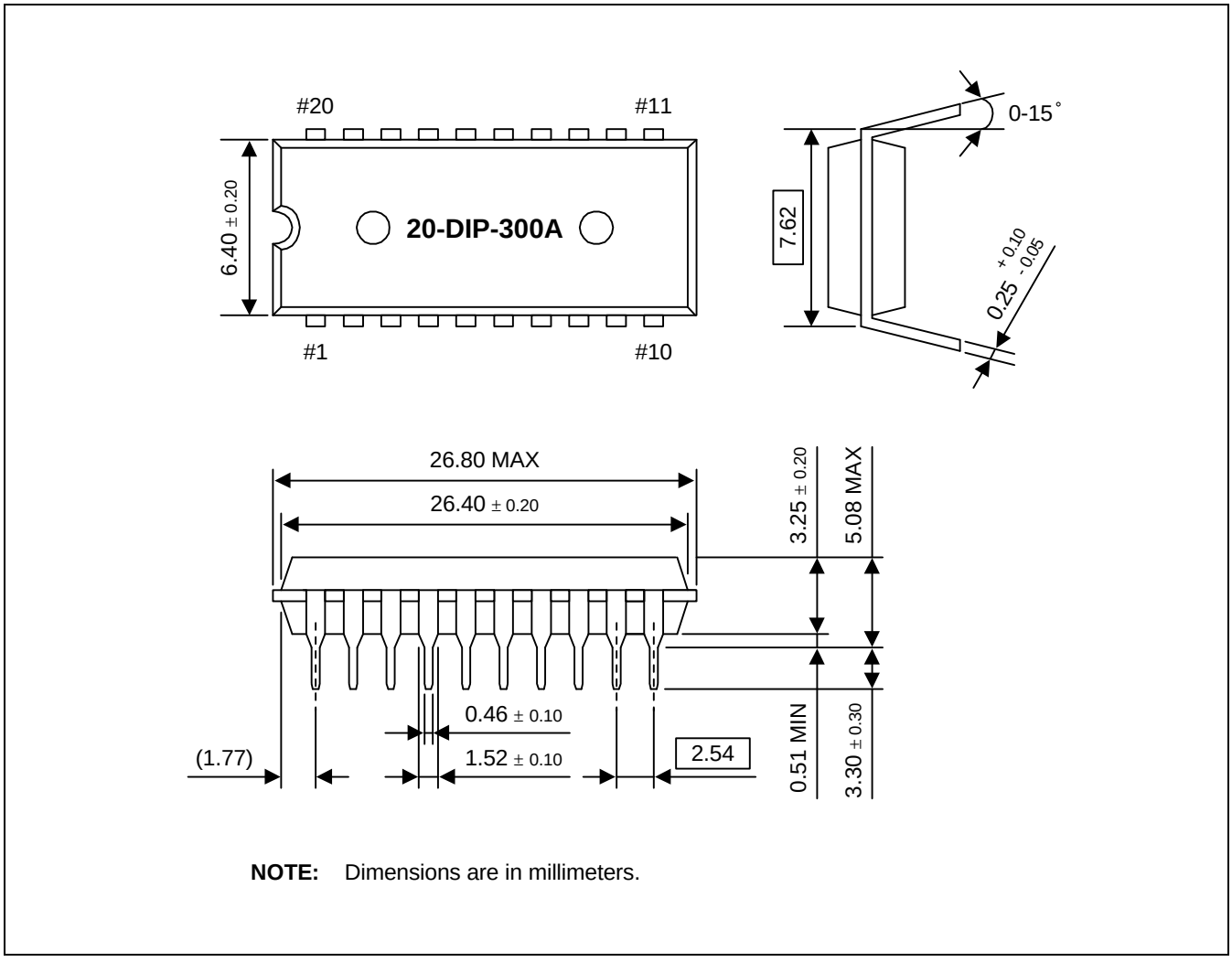
Figure 13-5. LVR Reset Timing

NOTES

14 MECHANICAL DATA

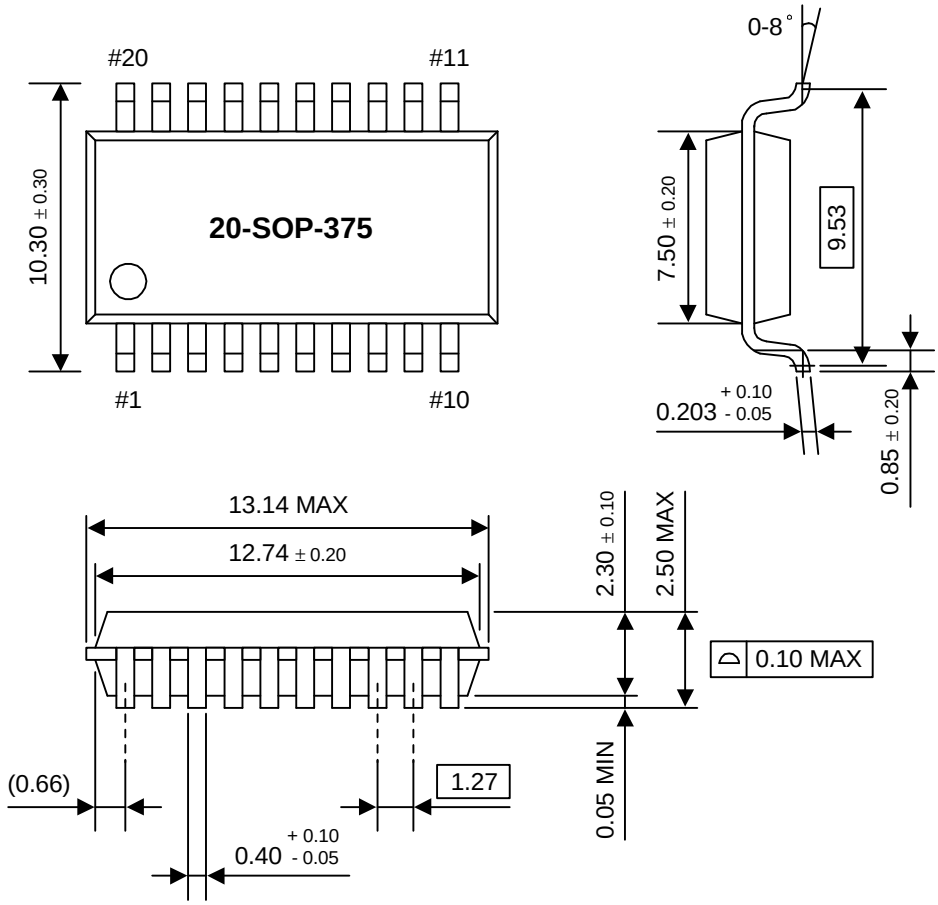
OVERVIEW

The S3C9454B/F9454B is available in a 20-pin DIP package (Samsung: 20-DIP-300A), a 20-pin SOP package (Samsung: 20-SOP-375), a 20-pin SSOP package (Samsung: 20-SSOP-225), a 16-pin DIP package (Samsung: 16-DIP-300A), a 16-pin SOP package (Samsung: 16-SOP-BD300-SG), a 16-pin SSOP package (Samsung: 16-SSOP-BD44). Package dimensions are shown in Figure 14-1, 14-2, 14-3, 14-4, 14-5 and 14-6.



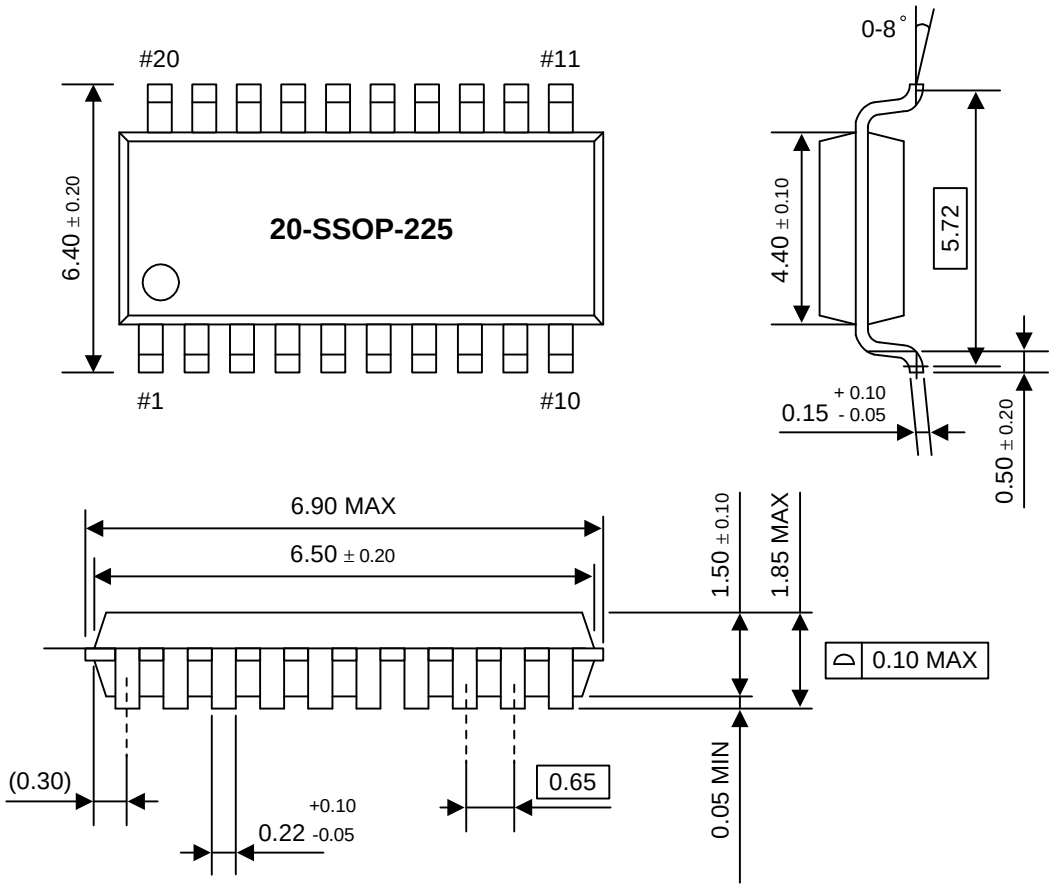
NOTE: Dimensions are in millimeters.

Figure 14-1. 20-DIP-300A Package Dimensions



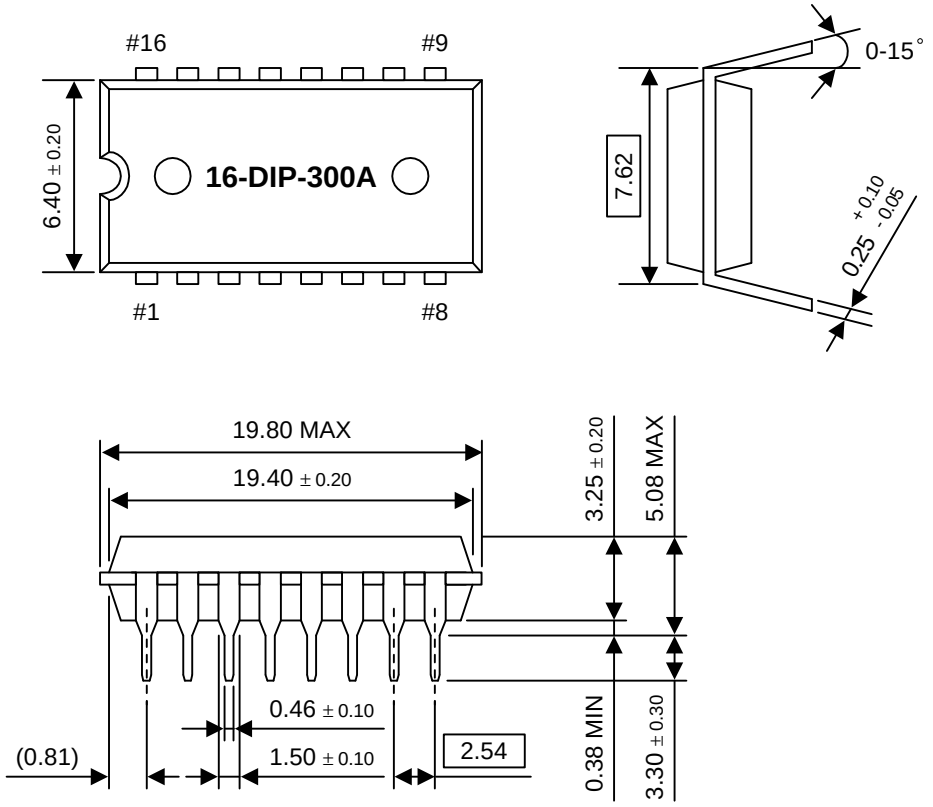
NOTE: Dimensions are in millimeters.

Figure 14-2. 20-SOP-375 Package Dimensions



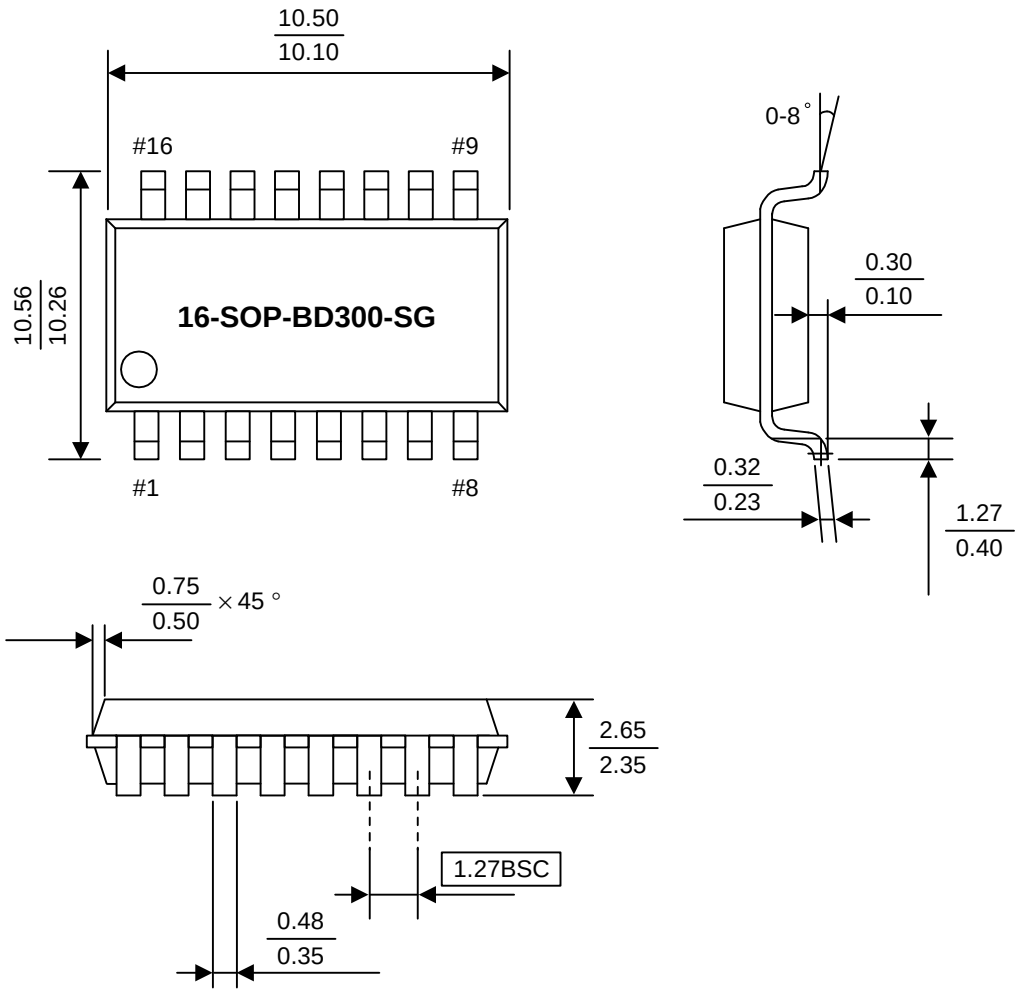
NOTE: Dimensions are in millimeters.

Figure 14-3. 20-SSOP-225 Package Dimensions



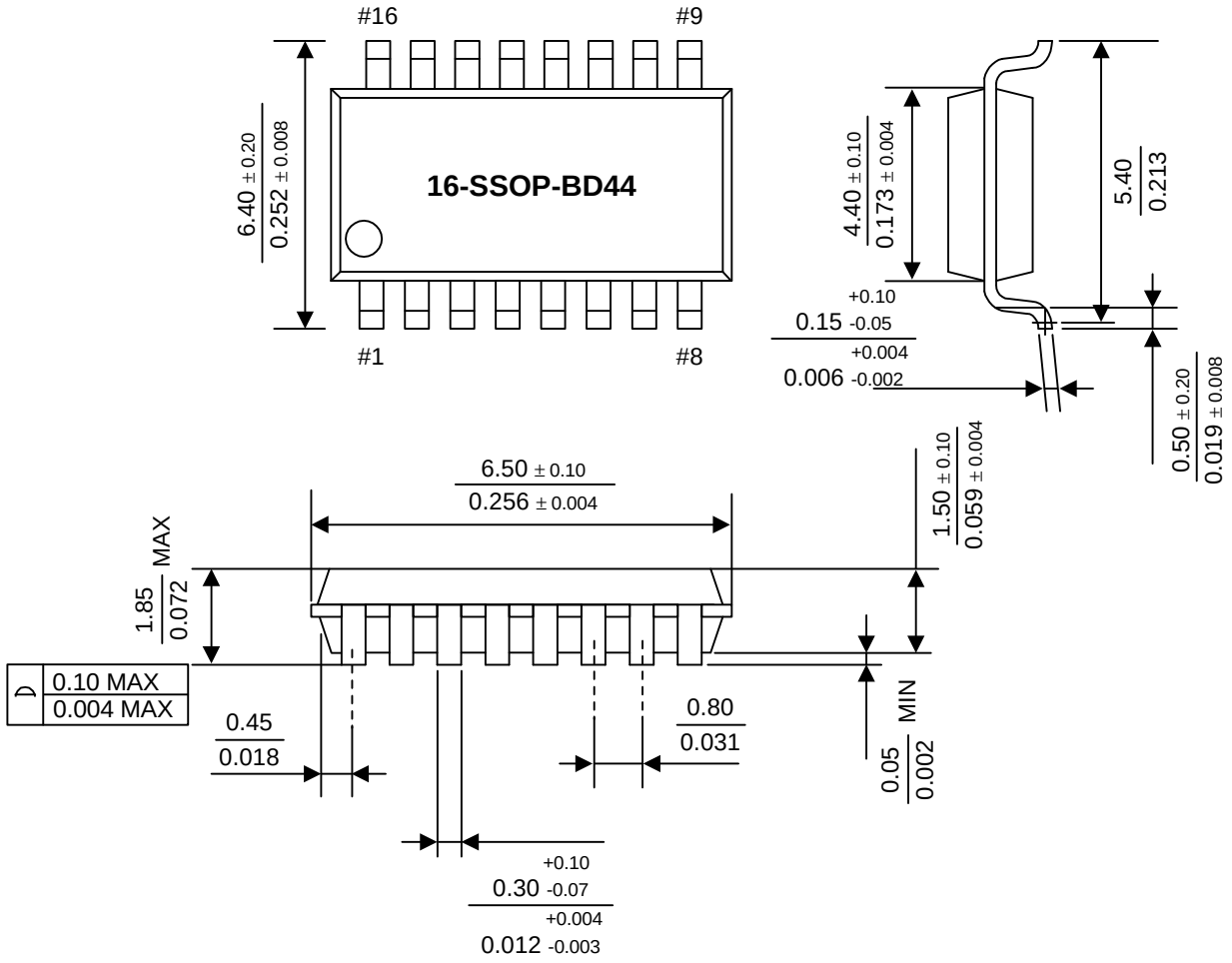
NOTE: Dimensions are in millimeters.

Figure 14-4. 16-DIP-300A Package Dimensions



NOTE: Dimensions are in millimeters.

Figure 14-5. 16-SOP-BD300-SG Package Dimensions



NOTE: Dimensions are in millimeters.

Figure 14-6. 16-SSOP-BD44 Package Dimensions

15 S3F9454B MTP

OVERVIEW

The S3F9454B single-chip CMOS microcontroller is the MTP (Multi Time Programmable) version of the S3C9454BB/F9454B microcontroller. It has an on-chip Flash ROM instead of masked ROM. The Flash ROM is accessed by serial data format.
The S3F9454B is fully compatible with the S3C9454BB/F9454B, in function, in D.C. electrical characteristics, and in pin configuration. Because of its simple programming requirements, the S3F9454B is ideal for use as an evaluation chip for the S3C9454BB/F9454B.

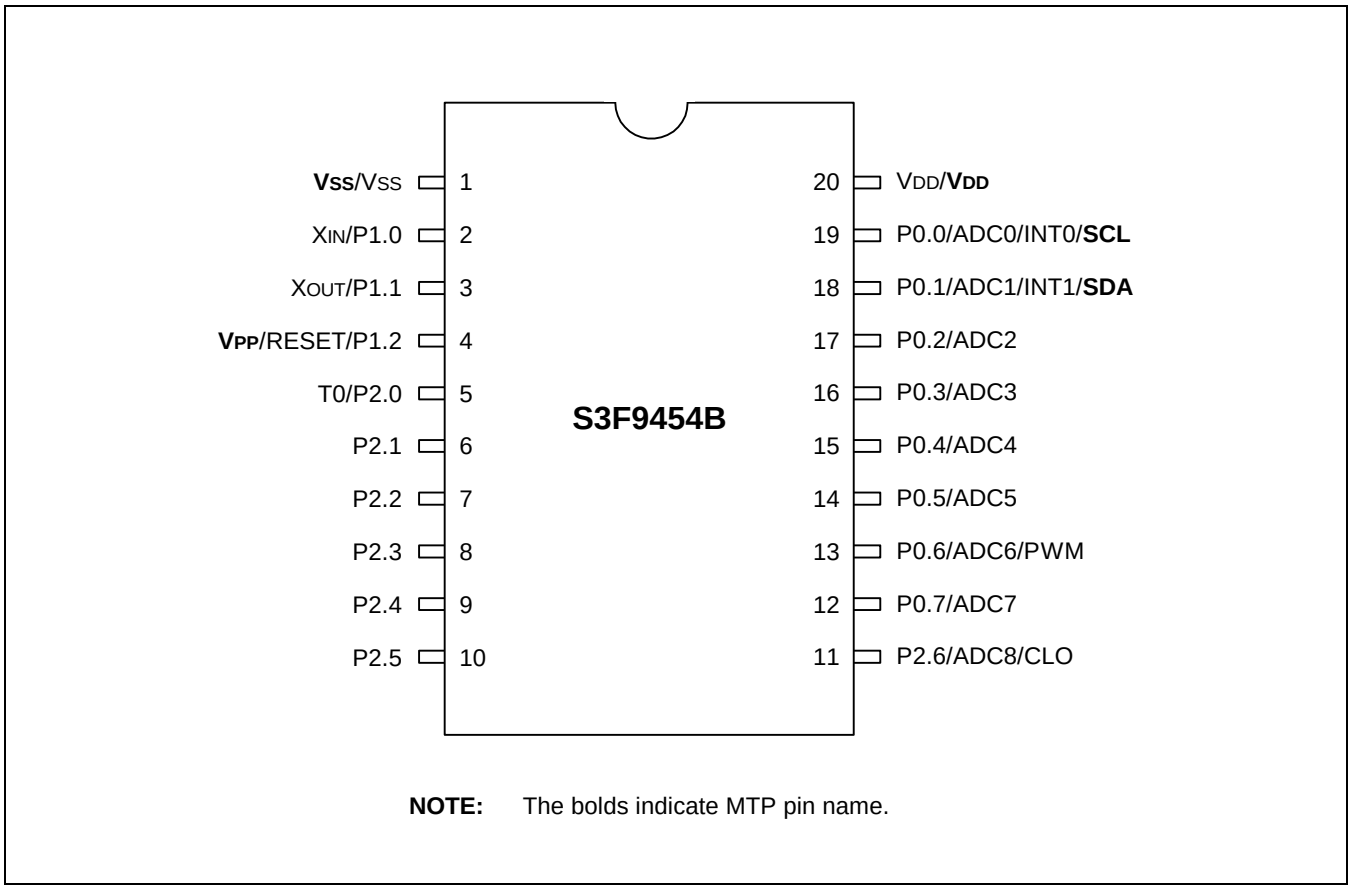


Figure 15-1. Pin Assignment Diagram (20-Pin Package)

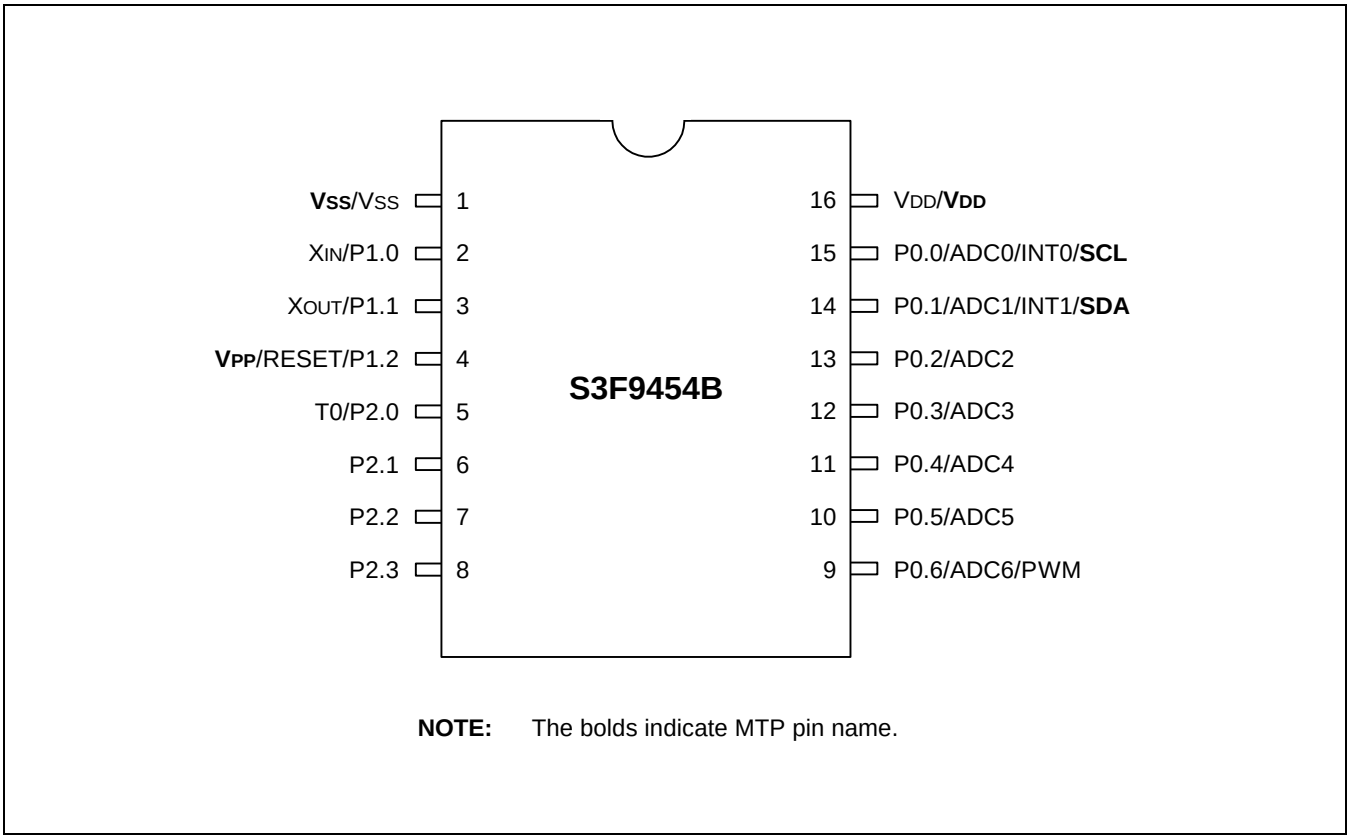


Figure 15-2. Pin Assignment Diagram (16-Pin Package)

Table 15-1. Descriptions of Pins Used to Read/Write the Flash ROM

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P0.1	SDA	18 (20-pin) 14 (16-pin)	I/O	Serial data pin (output when reading, Input when writing) Input and push-pull output port can be assigned
P0.0	SCL	19 (20-pin) 15 (16-pin)	I	Serial clock pin (input only pin)
RESET, P1.2	V_{PP}	4	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)
V_{DD}/V_{SS}	V_{DD}/V_{SS}	20 (20-pin), 16 (16-pin) 1 (20-pin), 1 (16-pin)	I	Logic power supply pin.

Table 15-2. Comparison of S3F9454B and S3C9454B Features

Characteristic	S3F9454B	S3C9454B
Program memory	4 Kbyte Flash ROM	4K byte mask ROM
Operating voltage (V_{DD})	2.0 V to 5.5 V	2.0 V to 5.5 V
MTP programming mode	V_{DD} = 5 V, V_{PP} = 12.5 V	
Pin configuration	20 DIP/20 SOP/20 SSOP/16 DIP/16SOP/16 SSOP/8 DIP/8 SOP	
EPROM programmability	User Program multi time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP} pin of the S3F9454B Flash ROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 15-3 below.

Table 15-3. Operating Mode Selection Criteria

V_{DD}	V_{PP}	REG/MEM	Address (A15–A0)	R/W	Mode
5 V	5 V	0	0000H	1	Flash ROM read
	12.5 V	0	0000H	0	Flash ROM program
	12.5 V	0	0000H	1	Flash ROM verify
	12.5 V	1	0E3FH	0	Flash ROM read protection

NOTE: "0" means Low level; "1" means High level.

NOTES

16 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system in turnkey form. The development support system is configured with a host system, debugging tools, and support software. For the host system, any standard computer that employs Win95/98/2000 as its operating system can be used. A sophisticated debugging tool is provided both hardware and software: the powerful in-circuit emulator, SMDS2+ or SK-1000, for S3C7, S3C9, S3C8 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2, and SK-1000 is supported by a third party tool vendor. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

SHINE

Samsung Host Interface for in-circuit Emulator, SHINE, is a multi-window based debugger for SMDS2+. SHINE provides pull-down and pop-up menus, mouse support, function/hot keys, and context-sensitive hyper-linked help. It has an advanced, multiple-windowed user interface that emphasizes ease of use. Each window can be sized, moved, scrolled, highlighted, added, or removed completely.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generates object code in standard hexadecimal format. Assembled program code includes the object code that is used for ROM data and required SMDS program control data. To assemble programs, SAMA requires a source file and an auxiliary definition (DEF) file with device specific information.

SASM86

The SASM86 is an relocatable assembler for Samsung's S3C9-series microcontrollers. The SASM86 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM86 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces the relocatable object code only, so the user should link object file. Object files can be linked with other object files and loaded into memory.

HEX2ROM

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

TARGET BOARDS

Target boards are available for all S3C9-series microcontrollers. All required target system cables and adapters are included with the device-specific target board.

MTPs

Multi times programmable microcontrollers (MTPs) are under development for S3C9454B/F9454B microcontroller.

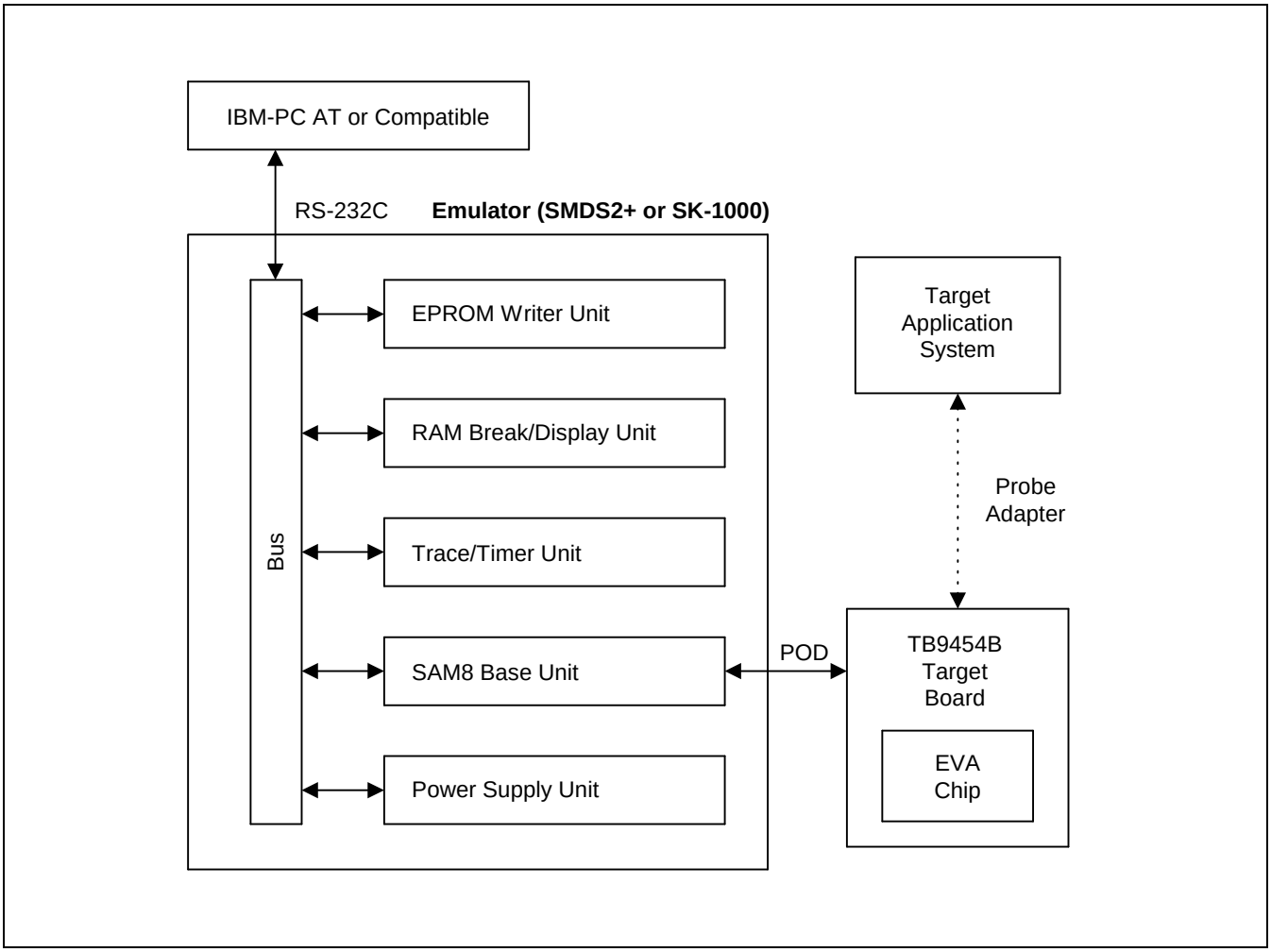


Figure 16-1. SMDS2+ or SK-1000 Product Configuration

TB9454B TARGET BOARD

The TB9454B target board is used for the S3C9454B/F9454B microcontrollers. It is supported by the SK1000/SMDS2+ development systems.

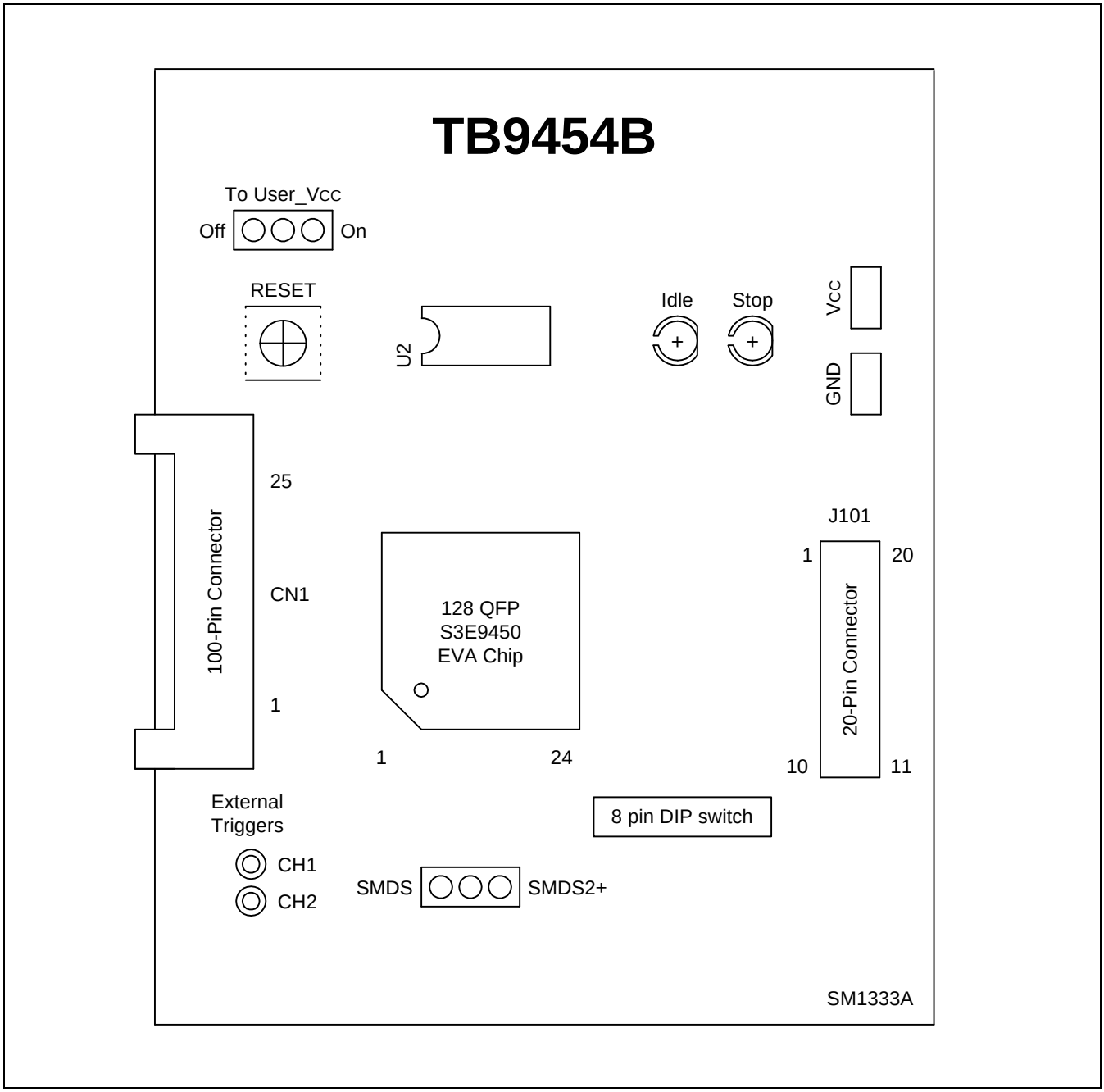
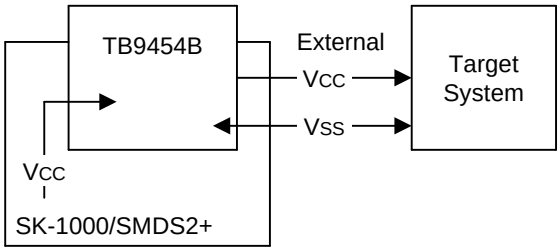
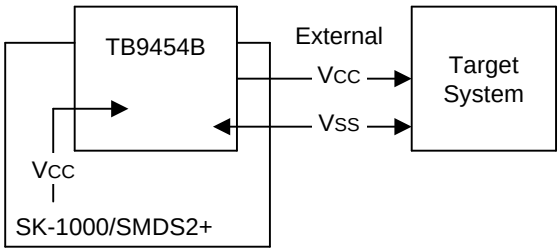


Figure 16-2. TB9454B Target Board Configuration

Table 16-1. Power Selection Settings for TB9454B

"To User_Vcc" Settings	Operating Mode	Comments
To user_Vcc off ○ ● ● on		The SK1000/SMDS2+ main board supplies V_{CC} to the target board (evaluation chip) and the target system.
To user_Vcc off ● ● ○ on		The SK1000/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 16-2. The SMDS2+ Tool Selection Setting

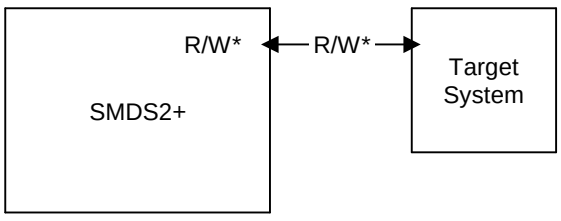
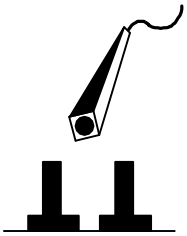
"SW1" Setting	Operating Mode
SMDS ● ● ○ SMDS2+	

Table 16-3. Using Single Header Pins as the Input Path for External Trigger Sources

Target Board Part	Comments
External Triggers Ch1 Ch2	 Connector from External Trigger Sources of the Application System You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for the SK-1000/SMDS2+ breakpoint and trace functions.

ON	Low
OFF	High

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

Figure 16-3. DIP Switch for Smart Option

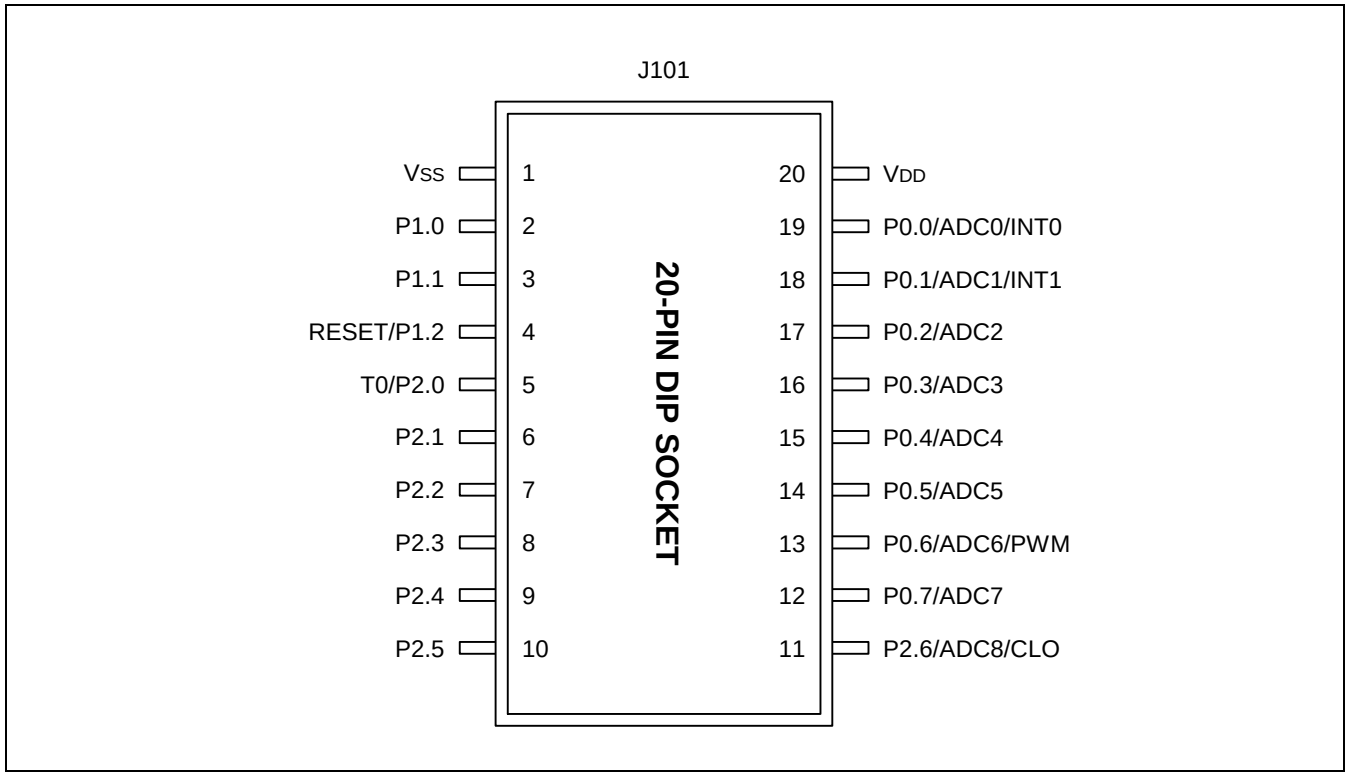


Figure 16-4. 20-Pin Connector for TB9454B

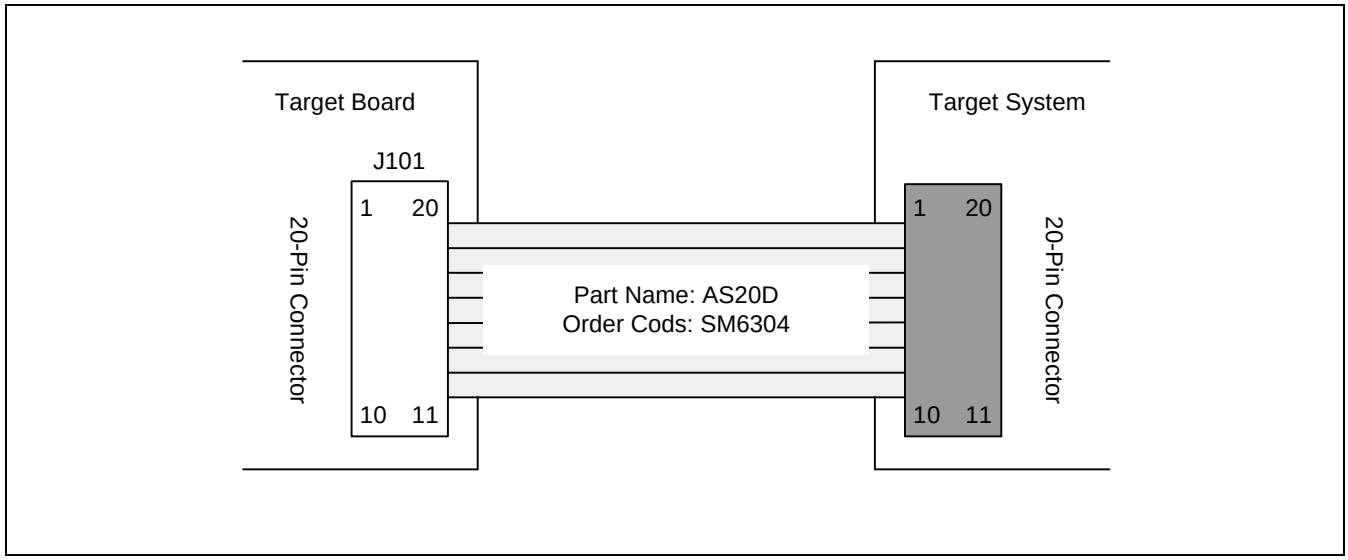


Figure 16-5. S3C9454B/F9454B Probe Adapter for 20-DIP Package