S3C8275X/F8275X/C8278X /F8278X/C8274X/F8274X

8-BIT CMOS MICROCONTROLLERS USER'S MANUAL

Revision 1.4

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.

S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X 8-Bit CMOS Microcontrollers
User's Manual, Revision 1.4
Publication Number: 21.4-S3-C8275X/F8275X/C8278X/F8278X/C8274X/F8274X-042007

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

Samsung Electronics Co., Ltd.
San #24 Nongseo-Dong, Giheung-Gu
Yongin-City, Gyeonggi-Do, Korea
C.P.O. Box #37, Suwon 446-711

TEL: (82)-(31)-209-5238
FAX: (82)-(31)-209-6494

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

NOTIFICATION OF REVISIONS

ORIGINATOR: Samsung Electronics, LSI Development Group, Gi-Heung, South Korea

PRODUCT NAME: S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X 8-bit CMOS Microcontroller

DOCUMENT NAME: S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X User's Manual, Revision1.4

DOCUMENT NUMBER: 21.4-S3-C8275X/F8275X/C8278X/F8278X/C8274X/F8274X-042007

EFFECTIVE DATE: April, 2007

SUMMARY: As a result of additional product testing and evaluation, some specifications published in the S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X User's Manual, Revision 1, have been changed. These changes for S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X microcontroller, which are described in detail in the *Revision Descriptions* section below, are related to the followings:

— Chapter 16. Embedded flash memory interface
— Chapter 17. Electrical Data
— Chapter 7. Clock Circuit
— Chapter 2. Address Spaces

DIRECTIONS: Please note the changes in your copy (copies) of the S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X User's Manual, Revision 1. Or, simply attach the *Revision Descriptions* of the next page to S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X User's Manual, Revision 1.

REVISION HISTORY

Revision	Date	Remark
0	February, 2005	Preliminary spec for internal release only.
1	April, 2005	First edition. Reviewed by Finechips.
1.1	July, 2005	Second edition. Reviewed by Finechips.
1.2	August, 2005	Third edition. Reviewed by Finechips.
1.3	May, 2006	Fourth edition. Reviewed by Finechips
1.4	April, 2007	Fifth edition. Reviewed by Finechips

REVISION DESCRIPTIONS

1. Electrical Data

Table 17-12. A.C. Electrical Characteristics for Internal Flash ROM

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.0 V to 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Programming time (1)	Ftp	–	30	–	–	μs
Chip erasing time (2)	Ftp1	–	**50**	–	–	ms
Sector erasing time (3)	Ftp2	–	10	–	–	ms
Data access time	Ft_{RS}	–	–	25	–	ns
Number of writing/erasing	FNwe	–	–	–	**10,000(4)**	Times

NOTES:
1. The programming time is the time during which one byte (8-bit) is programmed.
2. The chip erasing time is the time during which all 16K byte block is erased.
3. The sector erasing time is the time during which all 128 byte block is erased.
4. **Maximum number of writing/erasing is 10,000 times for full-flash(S3F8275) and 100 times for half-flash (S3F8278X/F8274X).**
5. The chip erasing is available in Tool Program Mode only.

2. Condition of Operating Voltage

Condition of operating voltage is modified “fx = 0 – 4.2MHz” to “fx = 0.4 – 4.2MHz” at 2.0V – 3.6V and “fx = 0 – 8MHz” to “fx = 0.4 – 8MHz” at 2.5V – 3.6V in the page 17-2.

3. CHAPTHER 16. Embedded Flash Memory Interface

This chapter is modified for only S3F8275X.

4. CHAPTHER 7. Clock Circuit

The contents of OSCCON.7 should be changed “ 0 = Select normal circuit for sub oscillator” into “ 0 = Initial state” in the page 4-21 and Figure 7-10.

It is added “NOTE: The OSCCON.7 should be maintained to “1”, during the sub oscillator operation.” In the page 4-21 and Figure 7-10.

The figure 7-7 is modified partly.

Descriptions of Revision 1.4

1. Smart Option Area

The Figures are modified about smart option area. Those are “Figure 2-1. Program Memory Address Space” and “Figure 5-3. ROM Vector Address Area”.

2. CHAPTHER 17. Electrical Data

It is changed “V_{DD} = 2.0 V to 3.6 V” into “V_{DD} = 2.2 V to 3.6 V” in the Table 17-12.

3. DEVICE NAME

The device name is changed S3C8275/F8275/C8278/F8278/C8274/F8274 to S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X. The ‘X’ means ‘Commercial type’.

Preface

The S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X *Microcontroller User's Manual* is designed for application designers and programmers who are using the S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X microcontroller for application development. It is organized in two main parts:

Part I	Programming Model	Part II	Hardware Descriptions

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

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

Chapter 1, "Product Overview," is a high-level introduction to S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X with general product descriptions, as well as detailed information about individual pin characteristics and pin circuit types.

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

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

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

Chapter 5, "Interrupt Structure," describes the S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X interrupt structure in detail and further prepares you for additional information presented in the individual hardware module descriptions in Part II.

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

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

Part II "hardware Descriptions," has detailed information about specific hardware components of the S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X microcontroller. Also included in Part II are electrical, mechanical, Flash MCU, and development tools data. It has 14 chapters:

Chapter 7	Clock Circuit	Chapter 15	Battery Level Detector
Chapter 8	RESET and Power-Down	Chapter 16	Embedded Flash Memory Interface
Chapter 9	I/O Ports	Chapter 17	Electrical Data
Chapter 10	Basic Timer	Chapter 18	Mechanical Data
Chapter 11	Timer 1	Chapter 19	S3F8275X/F8278X/F8274X Flash MCU
Chapter 12	Watch Timer	Chapter 20	Development Tools
Chapter 13	LCD Controller/Driver		
Chapter 14	Serial I/O Interface		

Table of Contents

Part I — Programming Model

Chapter 1 Product Overview

Chapter 2 Address Spaces

Chapter 3 Addressing Modes

Table of Contents (Continued)

Chapter 4 Control Registers

Chapter 5 Interrupt Structure

Chapter 6 Instruction Set

Table of Contents (Continued)

Part II Hardware Descriptions

Chapter 7 Clock Circuit

Chapter 8 RESET and Power-Down

Chapter 9 I/O Ports

Chapter 10 Basic Timer

Table of Contents (Continued)

Chapter 11 Timer 1

Chapter 12 Watch Timer

Chapter 13 LCD Controller/Driver

Chapter 14 Serial I/O Interface

Chapter 15 Battery Level Detector

Table of Contents (Continued)

Chapter 16 Embedded Flash Memory Interface

Chapter 17 Electrical Data

Chapter 18 Mechanical Data

Chapter 19 S3F8275X/F8278X/F8274X Flash MCU

Chapter 20 Development Tools

List of Figures

List of Figures

List of Figures (Continued)

List of Figures (Concluded)

List of Tables

List of Tables (Continued)

List of Programming Tips

Description **Page Number**

List of Register Descriptions

List of Instruction Descriptions

List of Instruction Descriptions (Continued)

1 PRODUCT OVERVIEW

S3C8-SERIES MICROCONTROLLERS

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

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

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

S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X MICROCONTROLLER

The S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X single-chip CMOS microcontrollers are fabricated using the highly advanced CMOS process, based on Samsung's latest CPU architecture.

The S3C8275X/C8278X/C8274X is a microcontroller with a 16/8/4K-byte mask-programmable ROM embedded.

The S3F8275X/F8278X/F8274X is a microcontroller with a 16/8/4K-byte flash ROM embedded.

Using a proven modular design approach, Samsung engineers have successfully developed the S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X by integrating the following peripheral modules with the powerful SAM8 core:

- Seven programmable I/O ports, including six 8-bit ports and one 4-bit port, for a total of 52 pins.
- Eight bit-programmable pins for external interrupts.
- One 8-bit basic timer for oscillation stabilization and watchdog function (system reset).
- Two 8-bit timer/counter with selectable operating modes.
- Watch timer for real time

FLASH

The S3F8275X/F8278X/F8274X are FLASH version of the S3C8275X/C8278X/C8274X microcontroller. The S3F8275X/F8278X/F8274X microcontroller has an on-chip FLASH ROM instead of a masked ROM. The S3F8275X/F8278X/F8274X is comparable to the S3C8275X/C8278X/C8274X, both in function and in pin configuration. The S3F8275X only is a full flash. The full flash means that data can be written into the program ROM by an instruction.

FEATURES

CPU

- SAM88RC CPU core

Memory

- Program Memory(ROM)
  - 16K×8 bits program memory(S3C8275X/F8275X)
  - 8K×8 bits program memory(S3C8278X/F8278X)
  - 4K×8 bits program memory(S3C8274X/F8274X)
  - Internal flash memory(Program memory)
    - √ Sector size: 128 Bytes
    - √ 10 Years data retention
    - √ Fast programming time:
      - + Chip erase: 50ms
      - + Sector erase: 10ms
      - + Byte program: 30us
    - √ User programmable by 'LDC' instruction
    - √ Endurance: 10,000 erase/program cycles
    - √ Sector(128-bytes) erase available
    - √ Byte programmable
    - √ External serial programming support
    - √ Expandable OBP™(On board program) sector
- Data Memory (RAM)
  - Including LCD display data memory
  - 544 × 8 bits data memory(S3C8275X/F8275X)
  - 288 × 8 bits data memory(S3C8278X/F8278X)
  - 288 × 8 bits data memory(S3C8274X/F8274X)

Instruction Set

- 78 instructions
- Idle and Stop instructions added for power-down modes

52 I/O Pins

- I/O: 16 pins
- I/O: 36 pins (Sharing with LCD signal outputs)

Interrupts

- 8 interrupt levels and 12 interrupt sources
- Fast interrupt processing feature

8-Bit Basic Timer

- Watchdog timer function
- 4 kinds of clock source

Two 8-Bit Timer/Counters

- Programmable interval timer
- External event counter function
- Configurable as one 16-bit timer/counters

Watch Timer

- Interval time: 3.91mS, 0.25S, 0.5S, and 1S at 32.768 kHz
- 0.5/1/2/4 kHz Selectable buzzer output

LCD Controller/Driver

- 32 segments and 4 common terminals
- Static, 1/2 duty, 1/3 duty, and 1/4 duty selectable
- Internal resistor circuit for LCD bias

8-bit Serial I/O Interface

- 8-bit transmit/receive mode
- 8-bit receive mode
- LSB-first or MSB-first transmission selectable
- Internal or External clock source

Battery Level Detector

- 3-criteria voltage selectable (2.2V, 2.4V, 2.8V)
- En/Disable by software for current consumption source

Low Voltage Reset (LVR)

- Criteria voltage: 2.2V
- En/Disable by smart option (ROM address: 3FH)

Two Power-Down Modes

- Idle: only CPU clock stops
- Stop: selected system clock and CPU clock stop

Oscillation Sources

- Crystal, ceramic, or RC for main clock
- Main clock frequency: 0.4 MHz – 8 MHz
- 32.768 kHz crystal for sub clock

Instruction Execution Times

- 500nS at 8 MHz fx(minimum)

Operating Voltage Range

- 2.0 V to 3.6 V at 0.4 – 4.2 MHz
- 2.5 V to 3.6 V at 0.4 – 8.0 MHz

Operating Temperature Range

- –25 °C to +85 °C

Package Type

- 64-QFP-1420F, 64-LQFP-1010

Smart Option

- Low Voltage Reset(LVR) level and enable/disable are at your hardwired option (ROM address 3FH)
- ISP related option selectable (ROM address 3EH)

BLOCK DIAGRAM

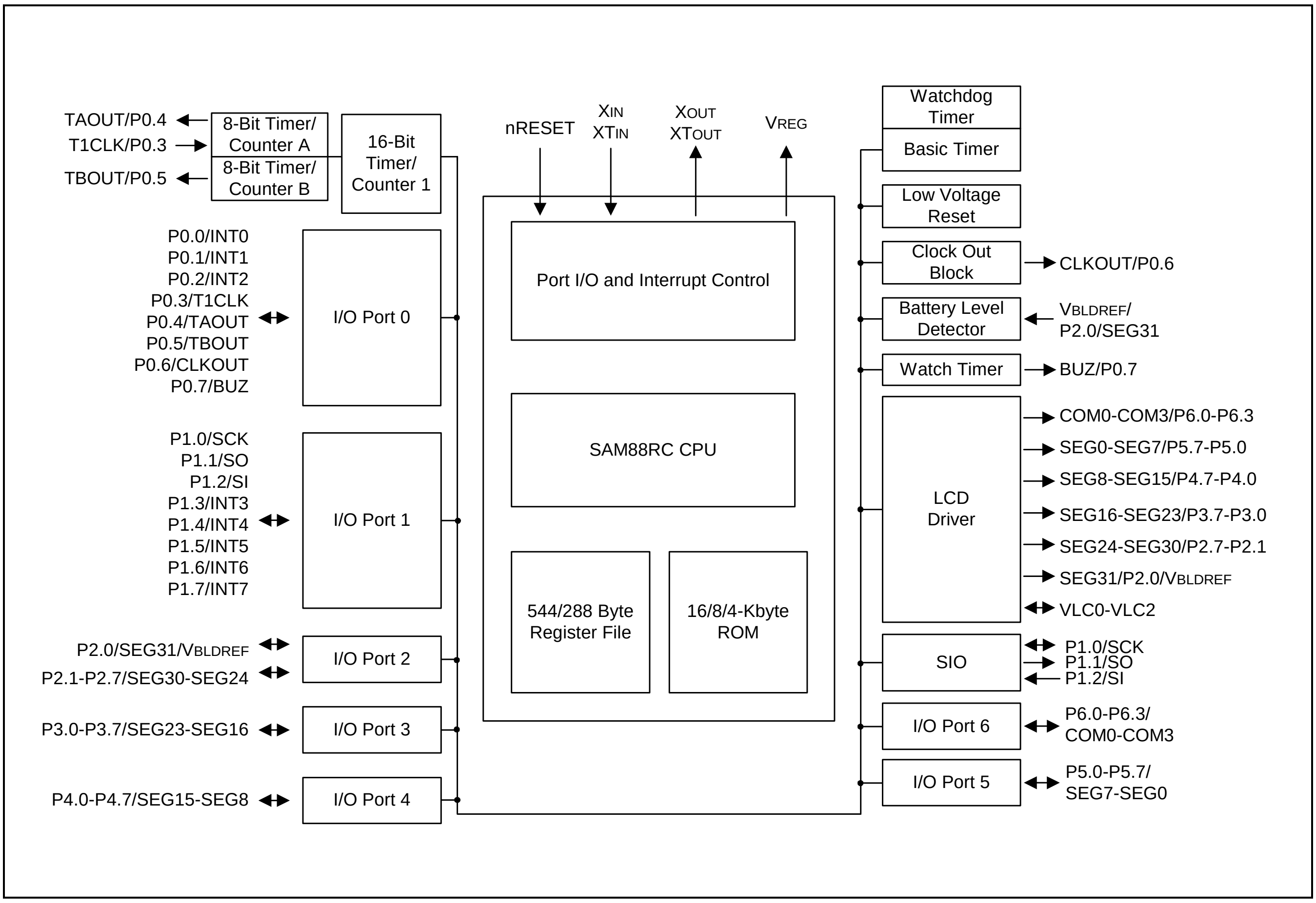


Figure 1-1. Block Diagram

PIN ASSIGNMENT

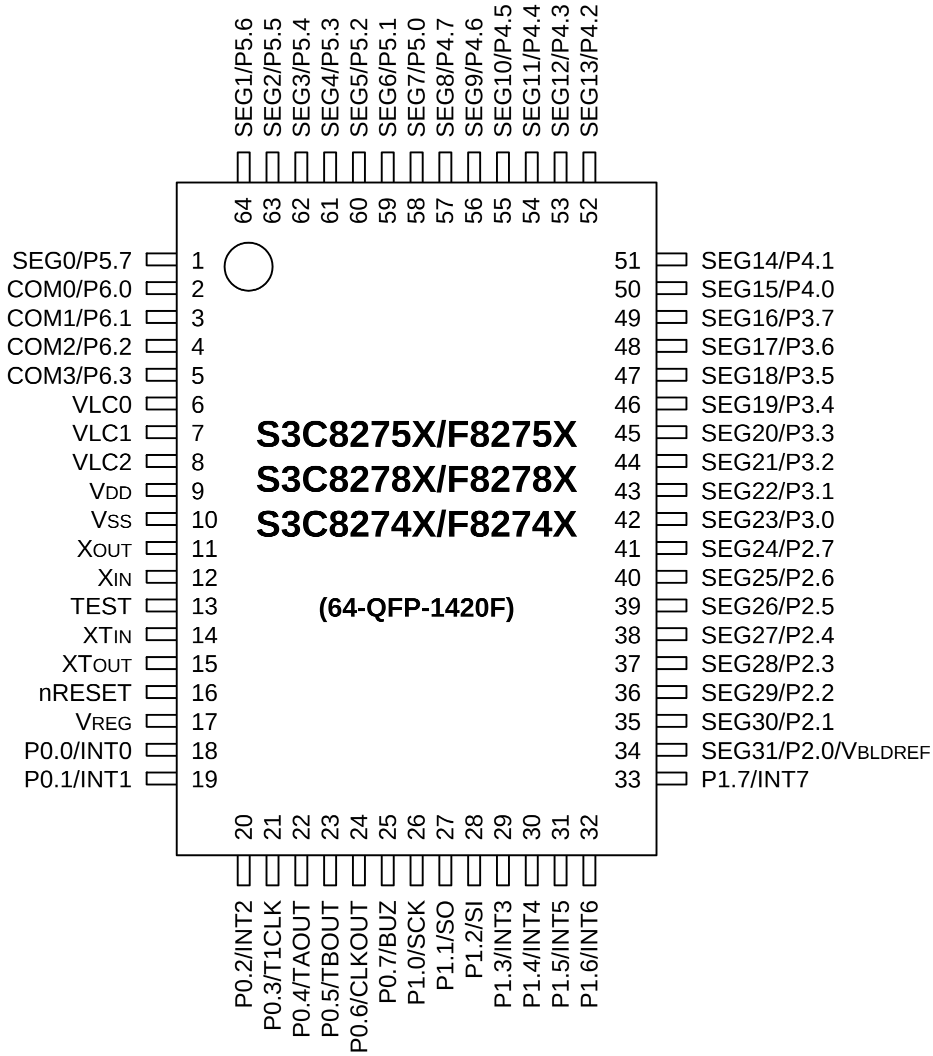


Figure 1-2. S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X Pin Assignments (64-QFP-1420F)

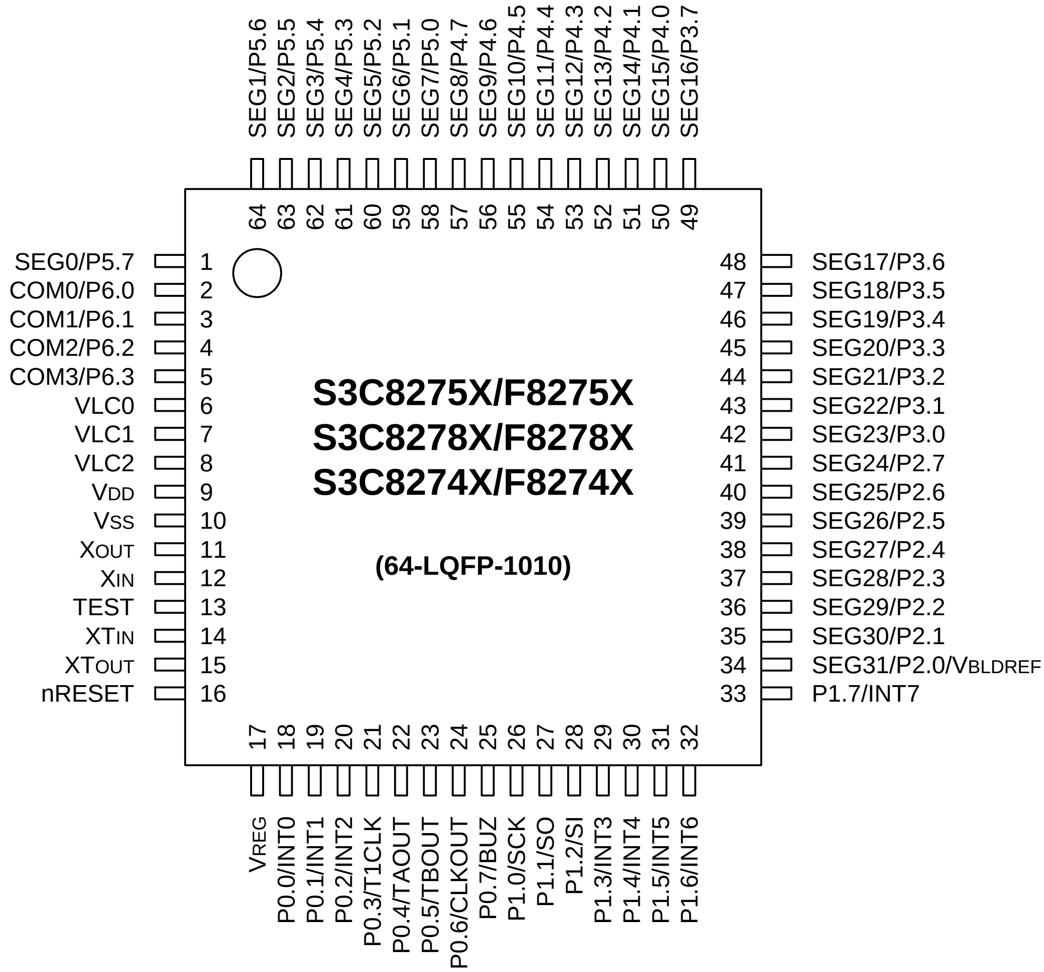


Figure 1-3. S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X Pin Assignments (64-LQFP-1010)

PIN DESCRIPTIONS

Table 1-1. S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X Pin Descriptions

Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Shared Functions
P0.0–P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	I/O	I/O port with bit-programmable pins; Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups; P0.0–P0.2 are alternately used for external interrupt input(noise filters, interrupt enable and pending control).	E-4	18–20 21 22 23 24 25	INT0–INT2 T1CLK TAOUT TBOUT CLKOUT BUZ
P1.0 P1.1 P1.2 P1.3–P1.7	I/O	I/O port with bit-programmable pins; Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups; P1.3–P1.7 are alternately used for external interrupt input(noise filters, interrupt enable and pending control).	E-4	26 27 28 29–33	SCK SO SI INT3–INT7
P2.0 P2.1–P2.7	I/O	I/O port with bit-programmable pins; Input or push-pull, open-drain output and software assignable pull-ups.	H-10 H-8	34 35–41	SEG31/V_{BLDREF} SEG30–SEG24
P3.0–P3.7	I/O	I/O port with bit-programmable pins; Input or push-pull, open-drain output and software assignable pull-ups.	H-8	42–49	SEG23–SEG16
P4.0–P4.7 P5.0–P5.7 P6.0–P6.3	I/O	I/O port with bit-programmable pins; Input or push-pull output and software assignable pull-ups.	H-9	50–57 58–64, 1 2–5	SEG15–SEG8 SEG7–SEG0 COM0–COM3

Table 1-1. S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X Pin Descriptions (Continued)

Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Shared Functions
VLC0–VLC2	–	LCD power supply pins.	–	6–8	–
INT0–INT2 INT3–INT7	I/O	External interrupts input pins.	E-4	18–20 29–33	P0.0–P0.2 P1.3–P1.7
T1CLK	I/O	Timer 1/A external clock input.	E-4	21	P0.3
TAOUT	I/O	Timer 1/A clock output.	E-4	22	P0.4
TBOUT	I/O	Timer B clock output.	E-4	23	P0.5
CLKOUT	I/O	System clock output.	E-4	24	P0.6
BUZ	I/O	Output pin for buzzer signal.	E-4	25	P0.7
SCK, SO, SI	I/O	Serial clock, data output, and data input.	E-4	26,27,28	P1.0, P1.1, P1.2
COM0–COM3	I/O	LCD common signal outputs.	H-9	2–5	P6.0–P6.3
SEG0–SEG15 SEG16–SEG30 SEG31	I/O	LCD segment signal outputs.	H-9 H-8 H-10	1,64– 50 49–35 34	P5.7–P4.0 P3.7–P2.1 P2.0/V_{BLDREF}
V_{BLDREF}	I/O	Battery level detector reference voltage	H-10	34	P2.0/SEG31
V_{REG}	O	Regulator voltage output for sub clock (needed 0.1uF)	–	17	–
nRESET	I	System reset pin	B	16	–
XT_{IN}, XT_{OUT}	–	Sub oscillator pins	–	14, 15	–
X_{IN}, X_{OUT}	–	Main oscillator pins.	–	12, 11	–
TEST	I	Test input: it must be connected to V_{SS}	–	13	–
V_{DD}, V_{SS}	–	Power input pins	–	9, 10	–

PIN CIRCUITS

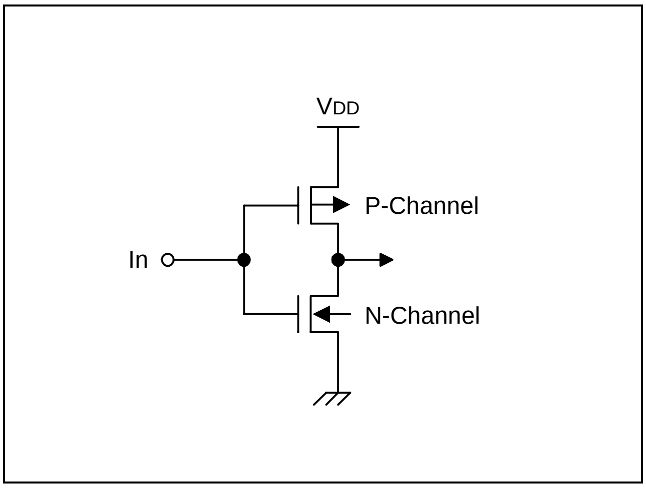


Figure 1-4. Pin Circuit Type A

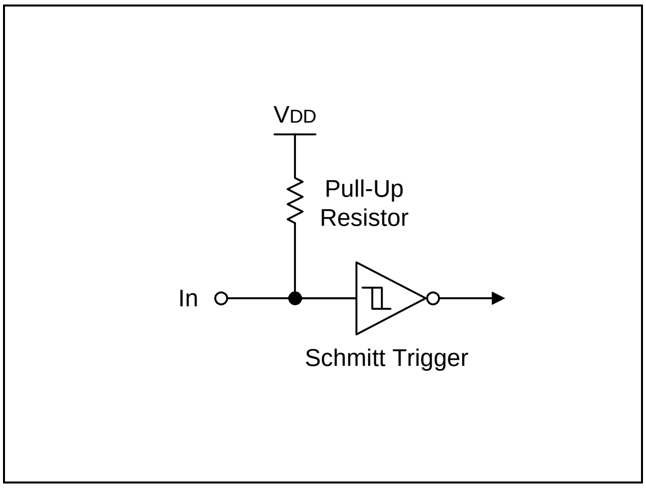


Figure 1-5. Pin Circuit Type B (nRESET)

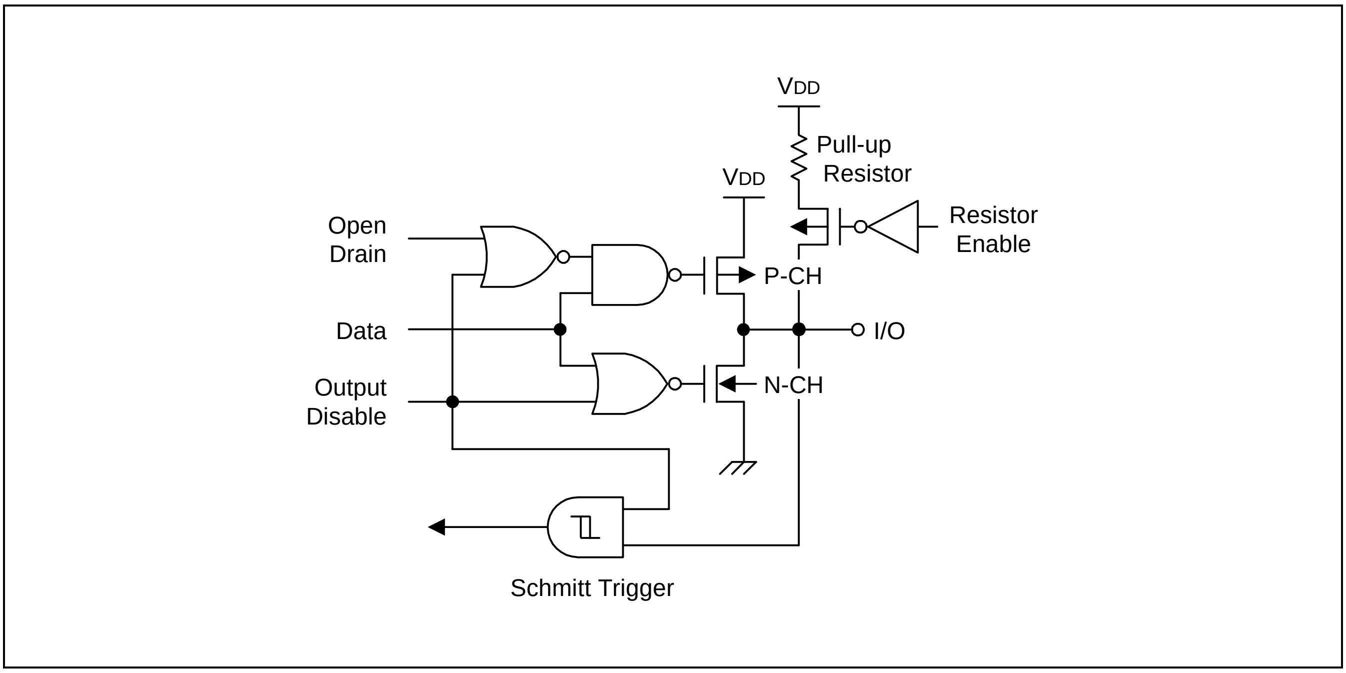


Figure 1-6. Pin Circuit Type E-4 (P0, P1)

SAMSUNG ELECTRONICS

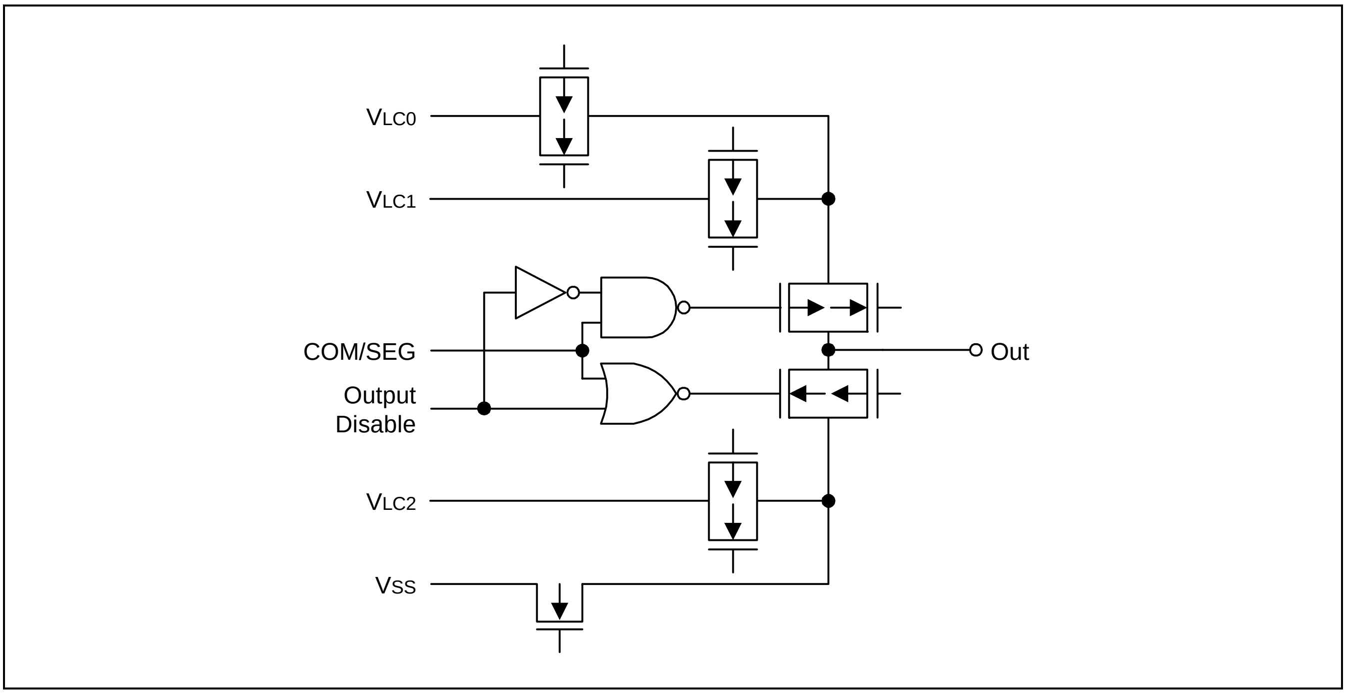


Figure 1-7. Pin Circuit Type H-4

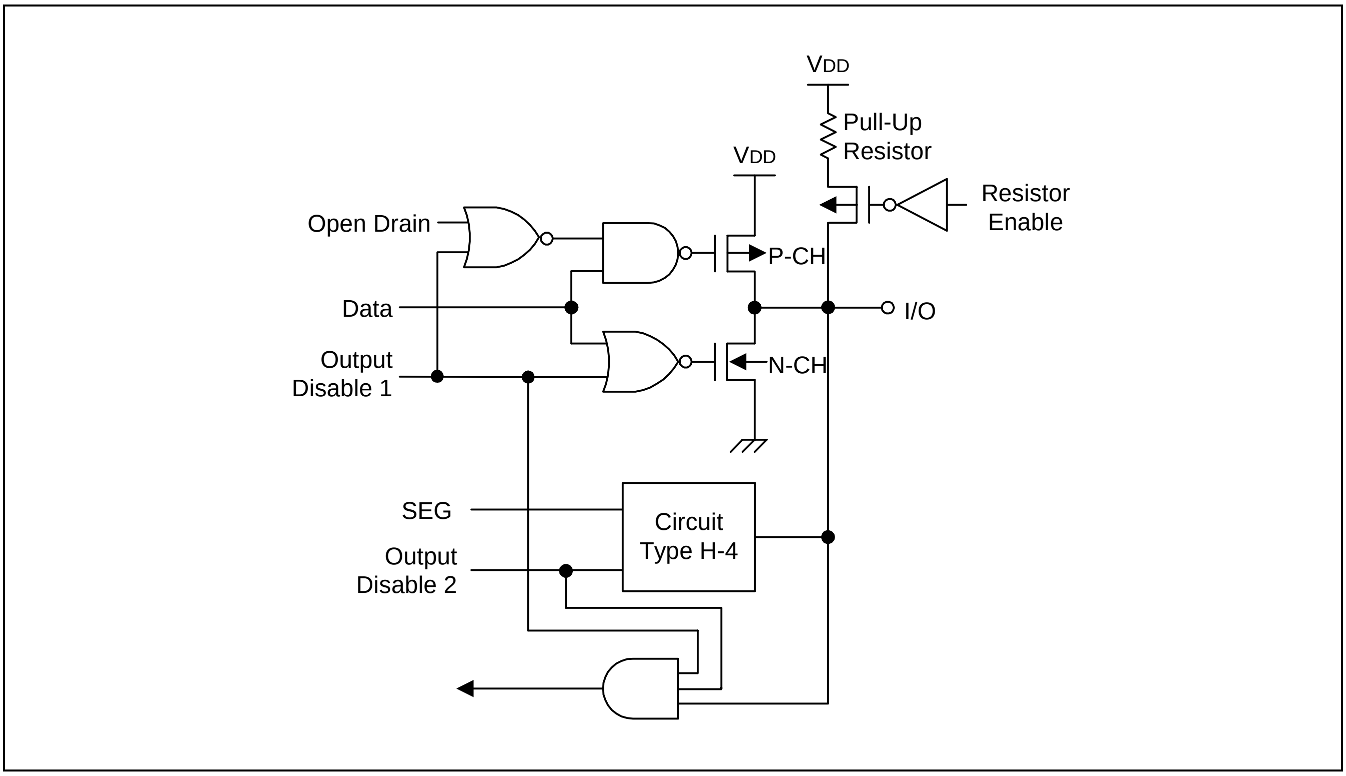


Figure 1-8. Pin Circuit Type H-8 (P2.1– P2.7, P3)

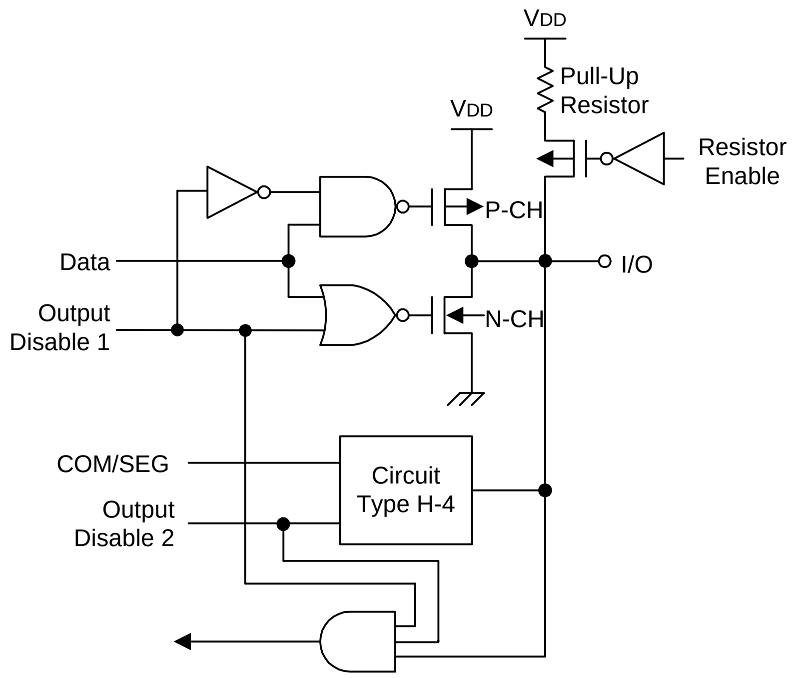


Figure 1-9. Pin Circuit Type H-9 (P4, P5, P6)

SAMSUNG ELECTRONICS

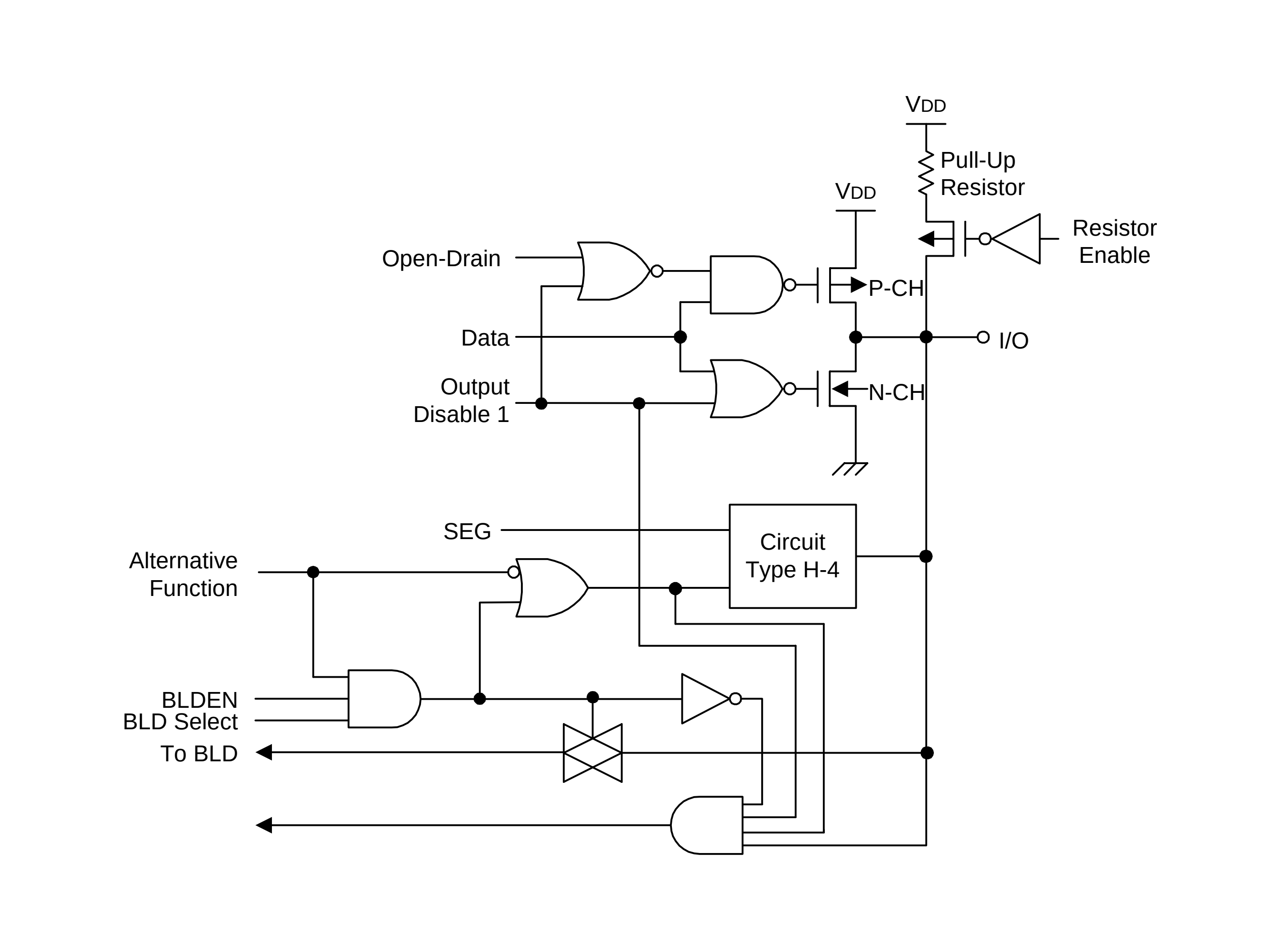


Figure 1-10. Pin Circuit Type H-10 (P2.0)

2 ADDRESS SPACES

OVERVIEW

The S3C8275X/C8278X/C8274X microcontroller has two types of address space:

- Internal program memory (ROM)
- Internal register file

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

The S3C8275X has an internal 16-Kbyte mask-programmable ROM. The S3C8278X has an internal 8-Kbyte mask-programmable ROM. The S3C8274X has an internal 4-Kbyte mask-programmable ROM.

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

A 16-byte LCD display register file is implemented.

There are 605 mapped registers in the internal register file. Of these, 528 are for general-purpose.
(This number includes a 16-byte working register common area used as a “scratch area” for data operations, two 192-byte prime register areas, and two 64-byte areas (Set 2)). Thirteen 8-bit registers are used for the CPU and the system control, and 48 registers are mapped for peripheral controls and data registers. Nineteen register locations are not mapped.

PROGRAM MEMORY (ROM)

Program memory (ROM) stores program codes or table data. The S3C8275X has 16K bytes internal mask-programmable program memory, the S3C8278X has 8K bytes, the S3C8274X has 4K bytes.

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

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

The reset address of ROM can be changed by a smart option only in the S3F8275X (Full-Flash Device). Refer to the chapter 16. Embedded Flash Memory Interface for more detail contents.

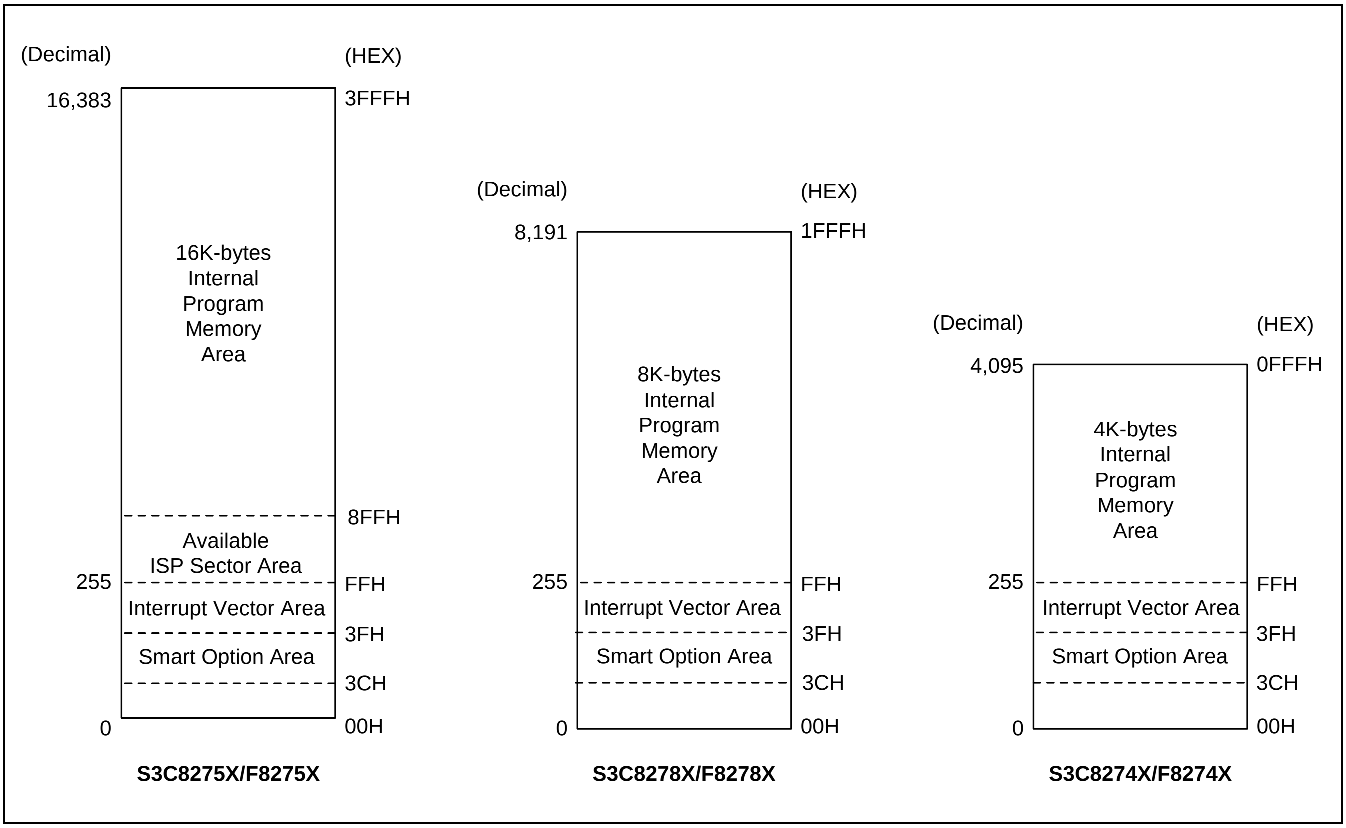


Figure 2-1. Program Memory Address Space

SAMSUNG ELECTRONICS

SMART OPTION

ROM Address: 003EH

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

ISP reset vector change enable/disable bit:
0 = OBP reset vector address
1 = Normal vector (address 0100H)

Not used

ISP protection size selection: (note)
00 = 256 bytes
01 = 512 bytes
10 = 1024 bytes
11 = 2048 bytes

ISP reset vector address selection bit:
00 = 200H (ISP area size: 256 byte)
01 = 300H (ISP area size: 512 byte)
10 = 500H (ISP area size: 1024 byte)
11 = 900H (ISP area size: 2048 byte)

ISP protection enable/disable bit:
0 = Enable (not erasable by LDC)
1 = Disable (erasable by LDC)

ROM Address: 003FH

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

Not used

These bits should be always logic "110b".

LVR enable/disable bit (criteria voltage: 2.2V):
0 = Disable LVR
1 = Enable LVR

ROM Address: 003CH

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

Not used

ROM Address: 003DH

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

Not used

NOTES:
1. After selecting ISP reset vector address in selecting ISP protection size, don't select upper than ISP area size.
2. When any values are written in the Smart Option area (003CH-003FH) by LDC instruction, the data of the area may be changed but the Smart Option is not affected. The data for Smart Option should be written in the Smart Option area (003CH-003FH) by OTP/MTP tools (SPW2 plus single programmer, or GW-PRO2 gang programmer).

Figure 2-2. Smart Option

Smart option is the ROM option for start condition of the chip. The ROM address used by smart option is from 003CH to 003FH.

The ISP of smart option (003EH) is available in the S3F8275X only. The default value of ROM address 003EH is FFH. And ROM address 003EH should be kept FFH when used the S3C8275X/C8278X/F8278X/C8274X/F8274X.

The LVR of smart option (003FH) is available in all the device, S3C8275X/F8275X/C8278X/F8278X/C8274X/F8274X. The default value of ROM address 003FH is FFH.

REGISTER ARCHITECTURE

In the S3C8275X/C8278X/C8274X implementation, the upper 64-byte area of register files is expanded two 64-byte areas, called *set 1* and *set 2*. The upper 32-byte area of set 1 is further expanded two 32-byte register banks (bank 0 and bank 1), and the lower 32-byte area is a single 32-byte common area.

In case of S3C8275X the total number of addressable 8-bit registers is 605. Of these 605 registers, 13 bytes are for CPU and system control registers, 16 bytes are for LCD data registers, 48 bytes are for peripheral control and data registers, 16 bytes are used as a shared working registers, and 512 registers are for general-purpose use, page 0-page 1 (in case of S3C8278X/C8274X, page 0).

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

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

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

Table 2-1. S3C8275X Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16-byte common working register area, two 192-byte prime register area, and two 64-byte set 2 area)	528
LCD data registers	16
CPU and system control registers	13
Mapped clock, peripheral, I/O control, and data registers	48
Total Addressable Bytes	605

Table 2-2. S3C8278X/C8274X Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16-byte common working register area, one 192-byte prime register area, and one 64-byte set 2 area)	272
LCD data registers	16
CPU and system control registers	13
Mapped clock, peripheral, I/O control, and data registers	48
Total Addressable Bytes	349

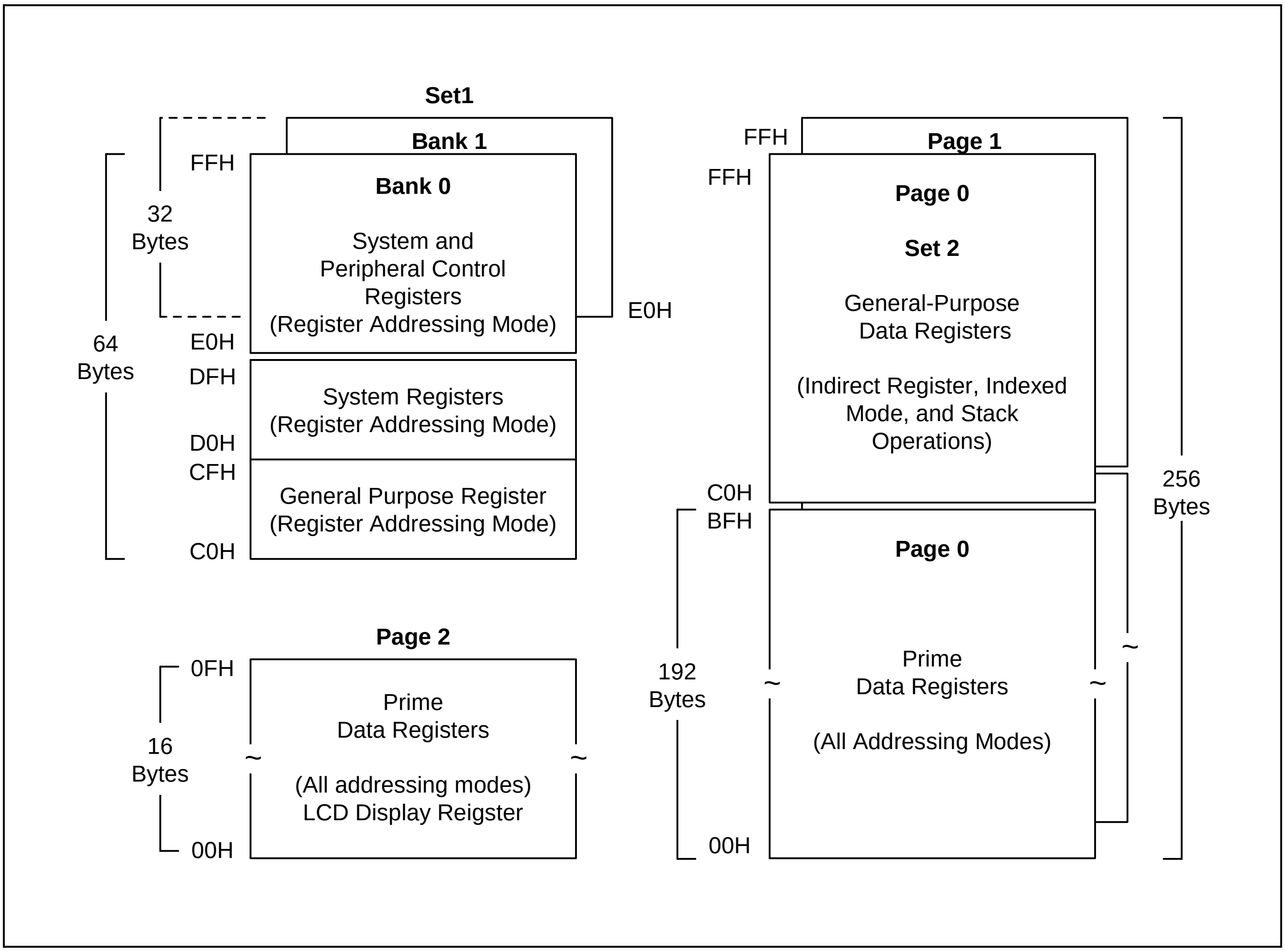
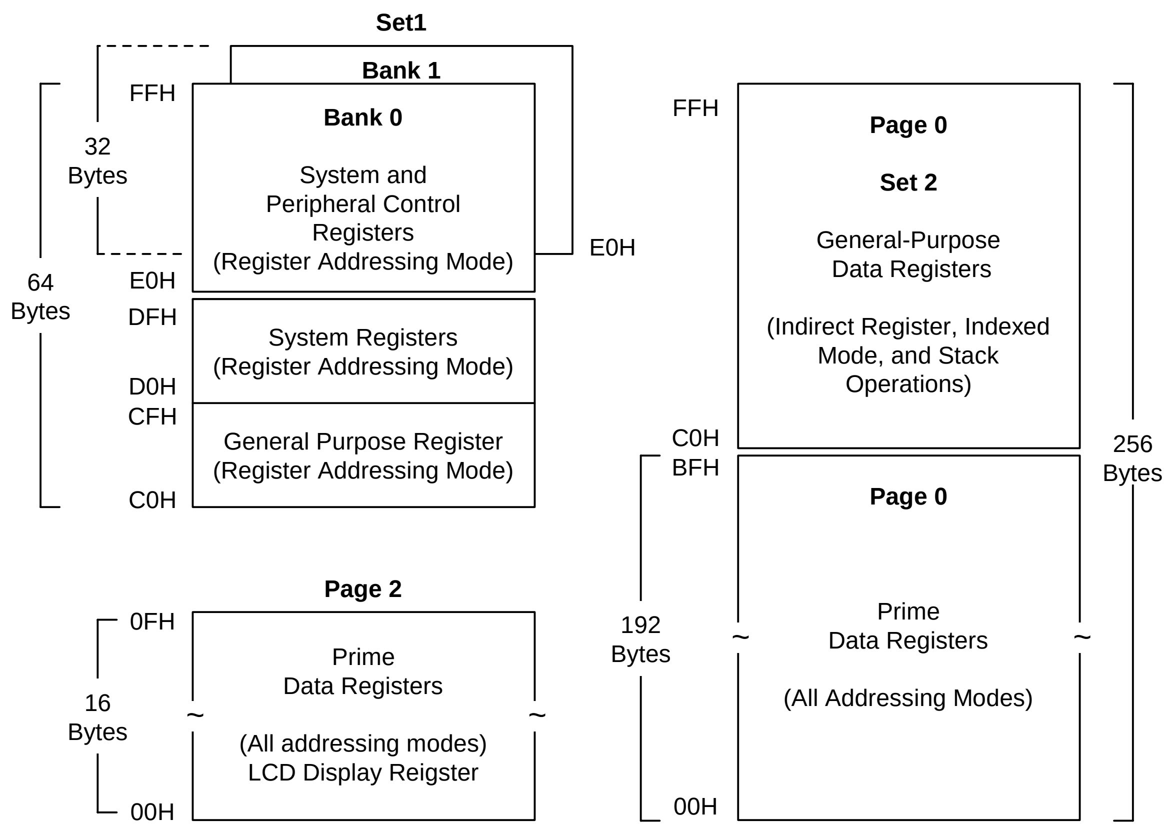


Figure 2-3. Internal Register File Organization (S3C8275X)

SAMSUNG ELECTRONICS



NOTE: In case of S3C8278X/C8274X, there are page 0 and page 2.
Page 2 is for LCD display register, 16 bytes.

Figure 2-4. Internal Register File Organization (S3C8278X/C8274X)

REGISTER PAGE POINTER (PP)

The S3C8-series architecture supports the logical expansion of the physical 256-byte internal register file (using an 8-bit data bus) into as many as 16 separately addressable register pages. Page addressing is controlled by the register page pointer (PP, DFH). In the S3C8275X/C8278X/C8274X microcontroller, a paged register file expansion is implemented for LCD data registers, and the register page pointer must be changed to address other pages.

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

Register Page Pointer (PP)
DFH, Set 1, R/W

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

Source register page selection bits:

0000	Source: Page 0
0001	Source: Page 1 (Not used for the S3C8278X/C8274X)
0010	Source: Page 2
others	Not used for the S3C8275X/C8278X/C8274X

Destination register page selection bits:

0000	Destination: Page 0
0001	Destination: Page 1 (Not used for the S3C8278X/C8274X)
0010	Destination: Page 2
others	Not used for the S3C8275X/C8278X/C8274X

NOTES:

1. In the S3C8275X microcontroller, the internal register file is configured as three pages (Pages 0-2). The pages 0-1 are used for general purpose register file, and page 2 is used for LCD data register or general purpose register.
2. In the S3C8278X/C8274X microcontroller, the internal register file is configured as two pages (Pages 0, 2). The page 0 is used for general purpose register file, and page 2 is used for LCD data register or general purpose register.
3. A hardware reset operation writes the 4-bit destination and source values shown above to the register page pointer. These values should be modified to address other pages.

Figure 2-5. Register Page Pointer (PP)

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

```
        LD      PP,#00H         ;  Destination ← 0, Source ← 0

        SRP     #0C0H

        LD      R0,#0FFH        ;  Page 0 RAM clear starts
RAMCL0  CLR     @R0
        DJNZ    R0,RAMCL0
        CLR     @R0             ;  R0 = 00H

        LD      PP,#10H         ;  Destination ← 1, Source ← 0

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

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

REGISTER SET 1

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

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

The upper two 32-byte areas (bank 0 and bank 1) of set 1 (E0H–FFH) contains 48 mapped system and peripheral control registers. The lower 32-byte area contains 16 system registers (D0H–DFH) and a 16-byte common working register area (C0H–CFH). You can use the common working register area as a “scratch” area for data operations being performed in other areas of the register file.

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

REGISTER SET 2

The same 64-byte physical space that is used for set 1 locations C0H–FFH is logically duplicated to add another 64 bytes of register space. This expanded area of the register file is called set *2*. For the S3C8275X,
the set 2 address range (C0H–FFH) is accessible on pages 0–1.
S3C8278X/C8274X, the set 2 address range (C0H–FFH) is accessible on page 0.

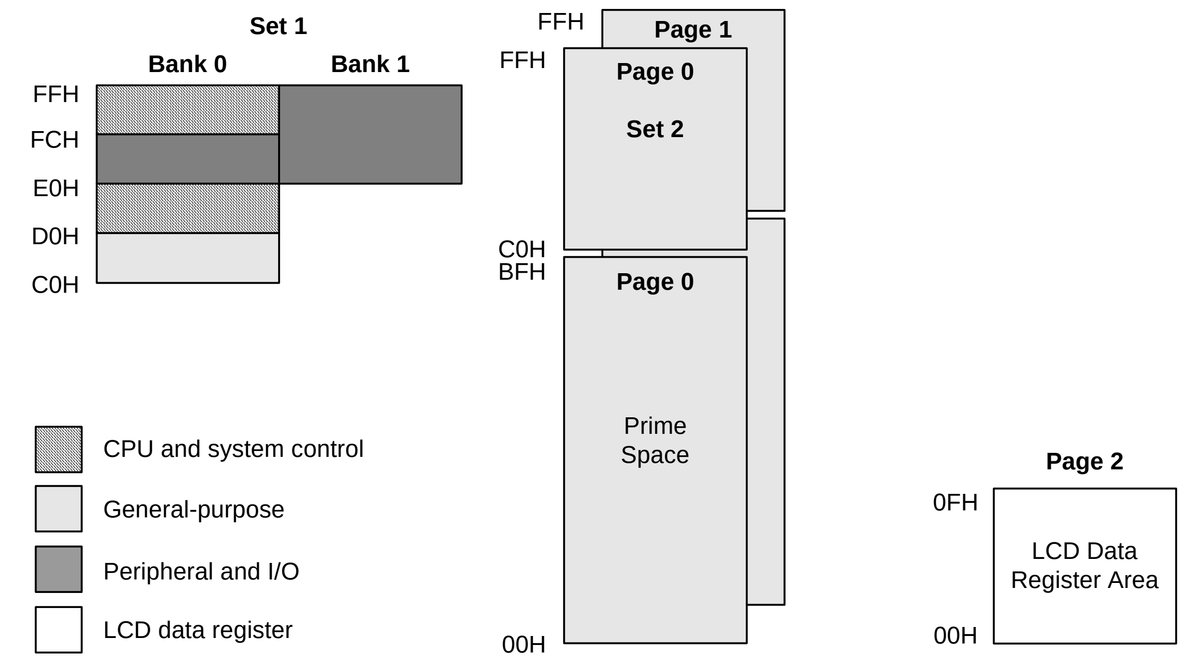
The logical division of set 1 and set 2 is maintained by means of addressing mode restrictions. You can use only Register addressing mode to access set 1 locations. In order to access registers in set 2, you must use Register Indirect addressing mode or Indexed addressing mode.

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

PRIME REGISTER SPACE

The lower 192 bytes (00H–BFH) of the S3C8275X/C8278X/C8274X's two or one 256-byte register pages is called *prime register area.* Prime registers can be accessed using any of the seven addressing modes (see Chapter 3, "Addressing Modes.")

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



NOTE: In case of S3C8278X/C8274X, there are page 0 and page 2. Page 2 is for LCD display register, 16 bytes.

Figure 2-6. Set 1, Set 2, Prime Area Register, and LCD Data Register Map

WORKING REGISTERS

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

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

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

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

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

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

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

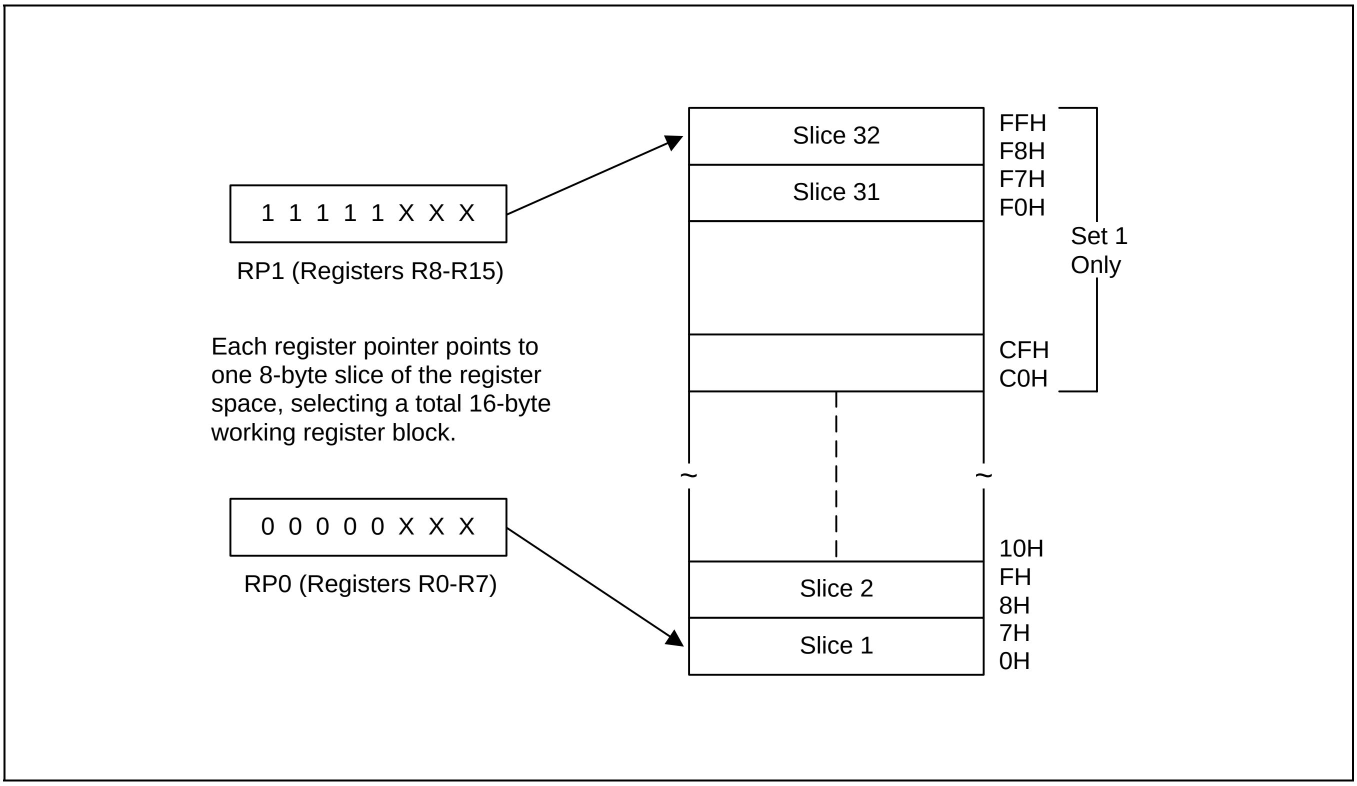


Figure 2-7. 8-Byte Working Register Areas (Slices)

SAMSUNG ELECTRONICS

USING THE REGISTER POINTS

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

To change a register pointer value, you load a new value to RP0 and/or RP1 using an SRP or LD instruction. (see Figures 2-8 and 2-9).

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

The selected 16-byte working register block usually consists of two contiguous 8-byte slices. As a general programming guideline, it is recommended that RP0 point to the "lower" slice and RP1 point to the "upper" slice (see Figure 2-8). In some cases, it may be necessary to define working register areas in different (non-contiguous) areas of the register file. In Figure 2-9, RP0 points to the "upper" slice and RP1 to the "lower" slice.

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

☞ PROGRAMMING TIP — Setting the Register Pointers

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

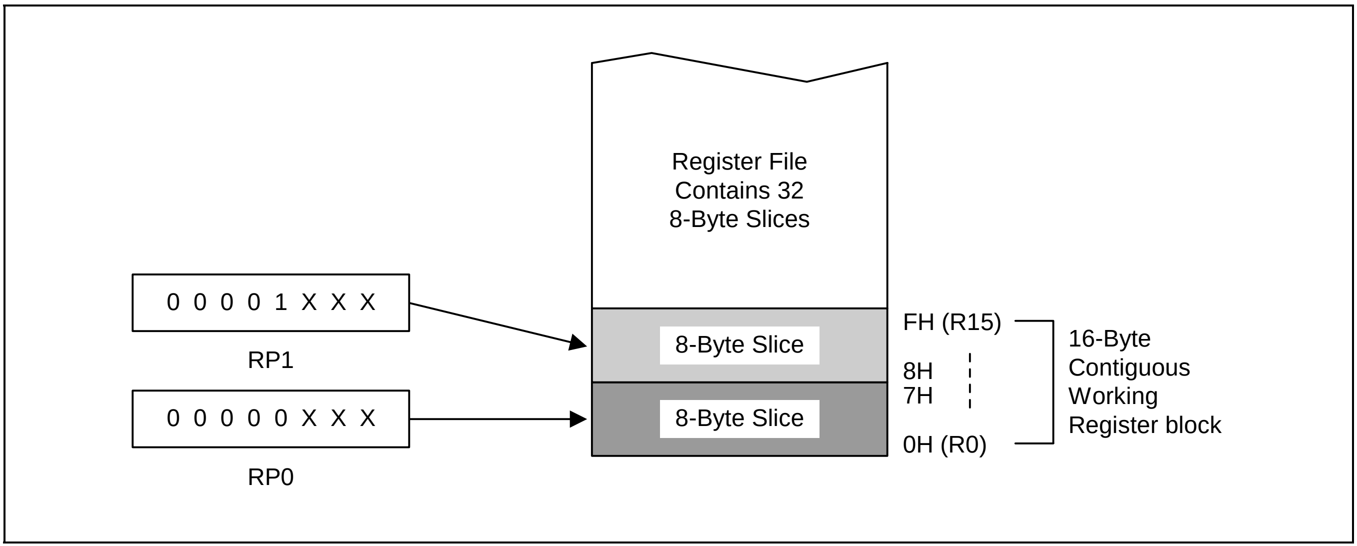


Figure 2-8. Contiguous 16-Byte Working Register Block

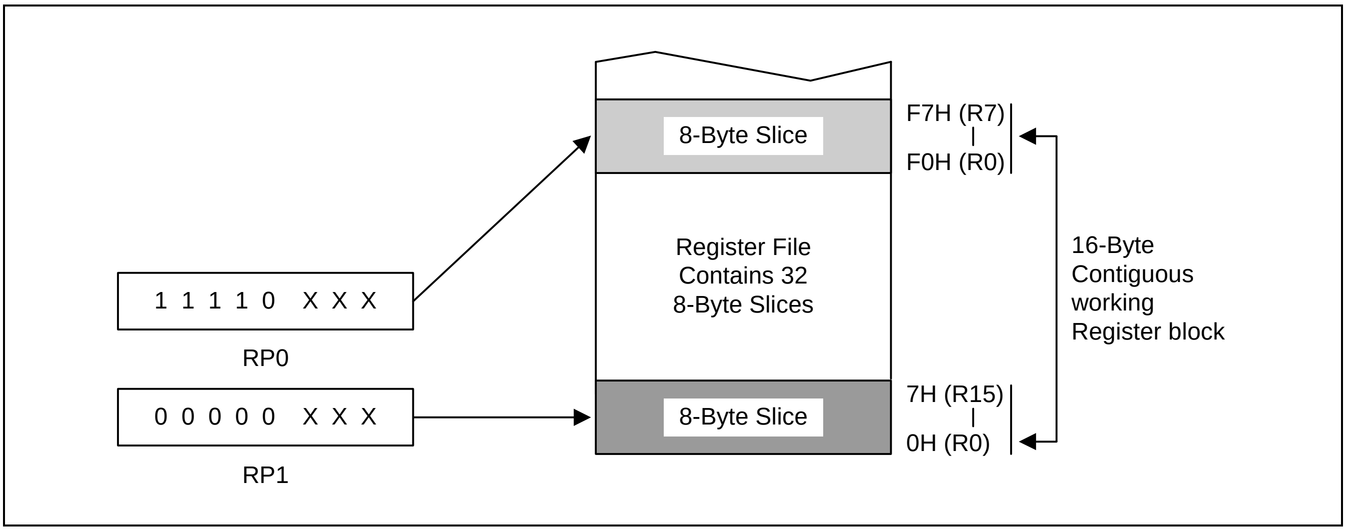


Figure 2-9. Non-Contiguous 16-Byte Working Register Block

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

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

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

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

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

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

REGISTER ADDRESSING

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

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

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

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

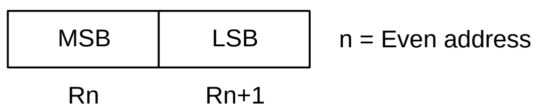


Figure 2-10. 16-Bit Register Pair

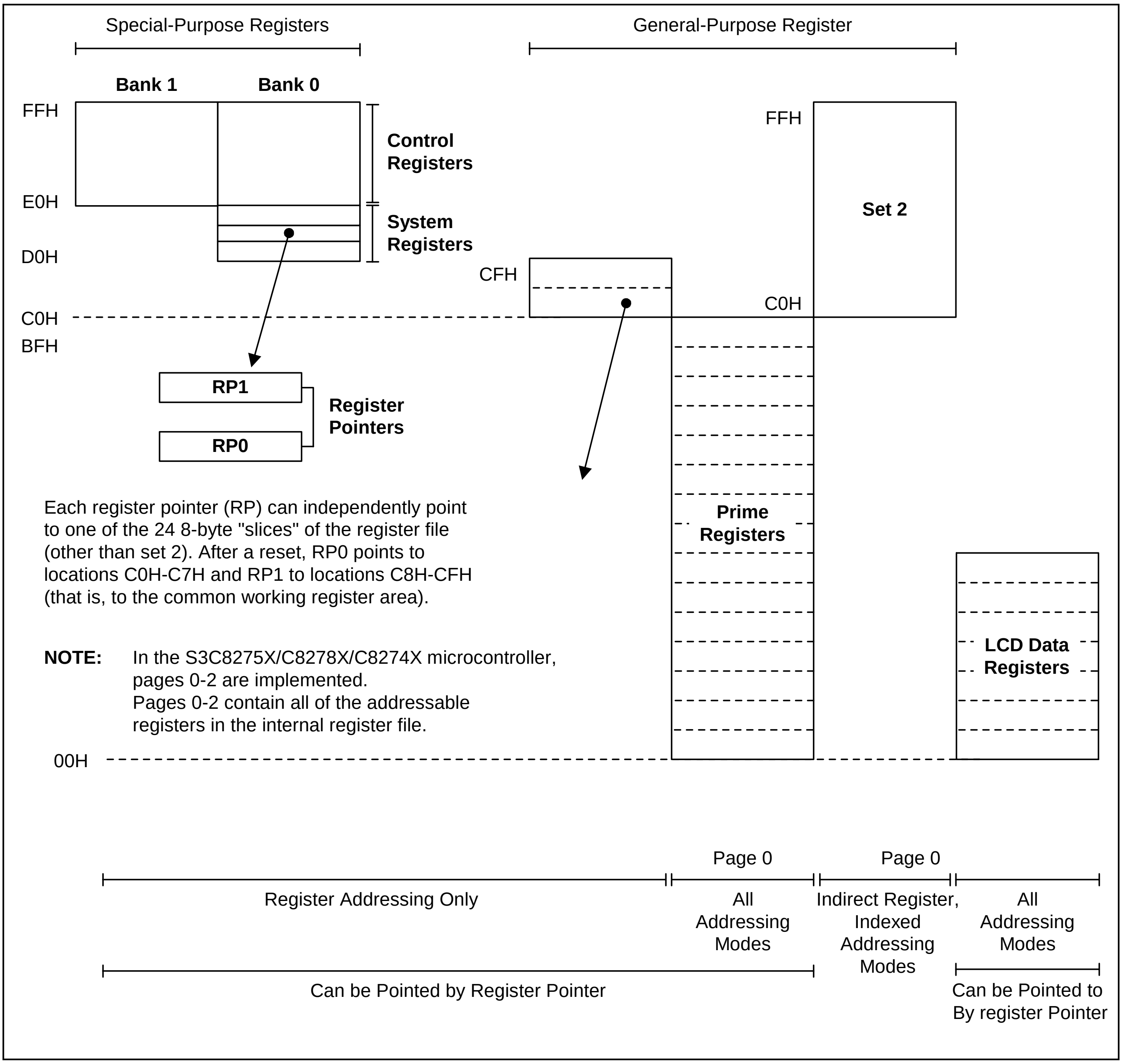


Figure 2-11. Register File Addressing

SAMSUNG ELECTRONICS

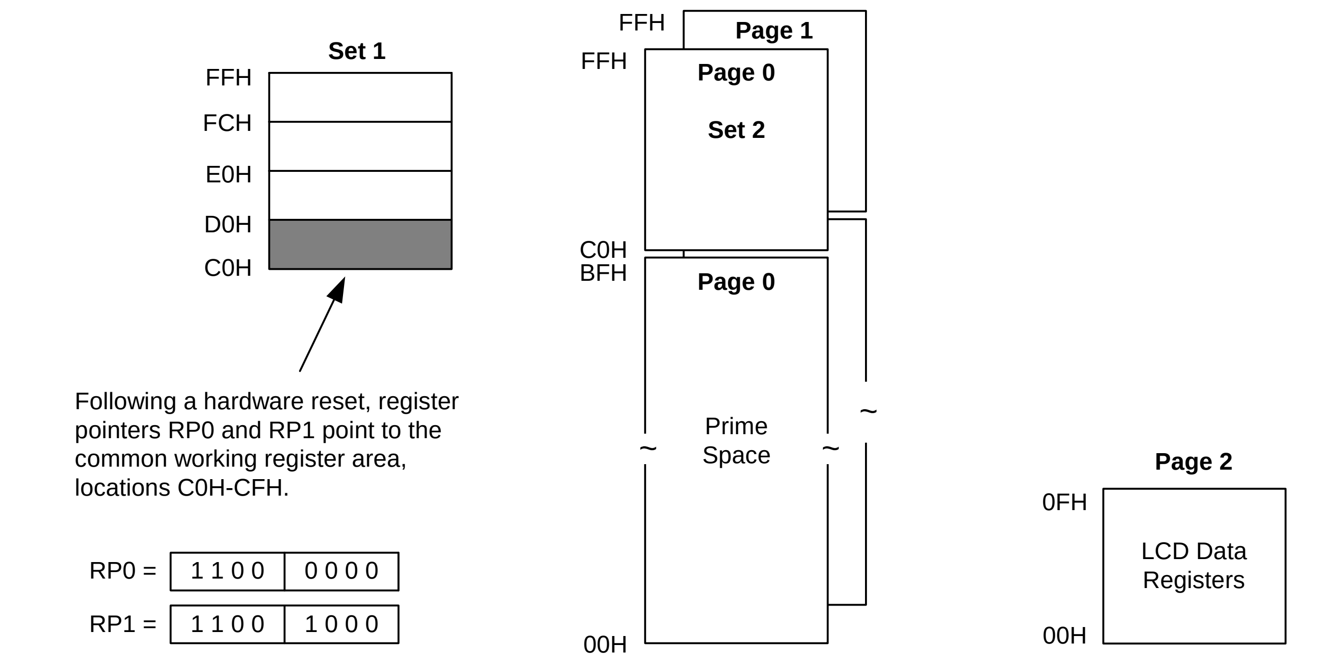
COMMON WORKING REGISTER AREA (C0H–CFH)

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

RP0 → C0H–C7H

RP1 → C8H–CFH

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



NOTE: In case of S3C8278X/C8274X, there are page 0 and page 2. Page 2 is for LCD display register, 16 bytes.

Figure 2-12. Common Working Register Area

☞ PROGRAMMING TIP — Addressing the Common Working Register Area

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

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

Use working register addressing instead:

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

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

Use working register addressing instead:

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

4-BIT WORKING REGISTER ADDRESSING

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

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

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

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

As shown in Figure 2-13, the result of this operation is that the five high-order bits from the register pointer are concatenated with the three low-order bits from the instruction address to form the complete address. As long as the address stored in the register pointer remains unchanged, the three bits from the address will always point to an address in the same 8-byte register slice.

Figure 2-14 shows a typical example of 4-bit working register addressing. The high-order bit of the instruction "INC R6" is "0", which selects RP0. The five high-order bits stored in RP0 (01110B) are concatenated with the three low-order bits of the instruction's 4-bit address (110B) to produce the register address 76H (01110110B).

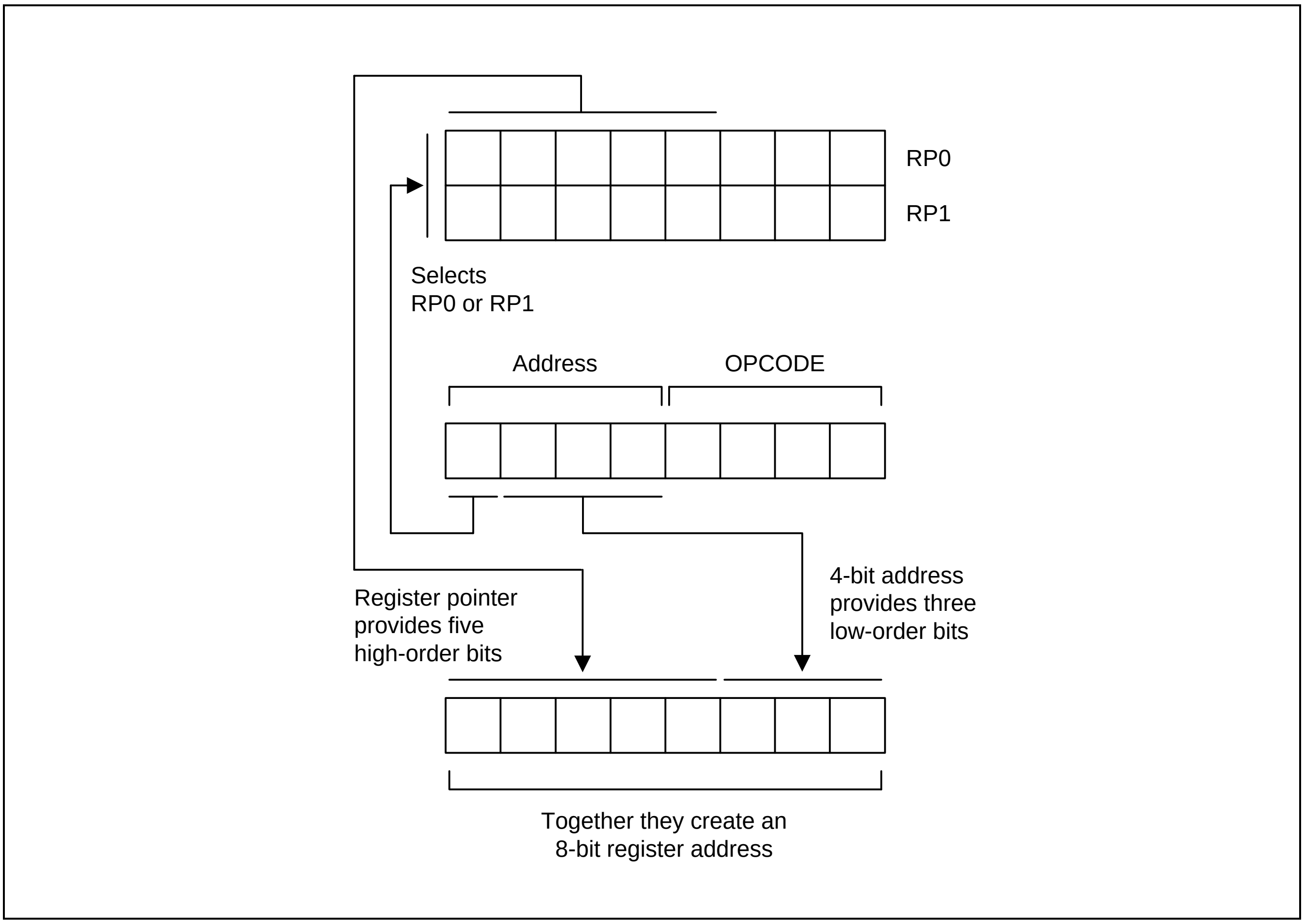


Figure 2-13. 4-Bit Working Register Addressing

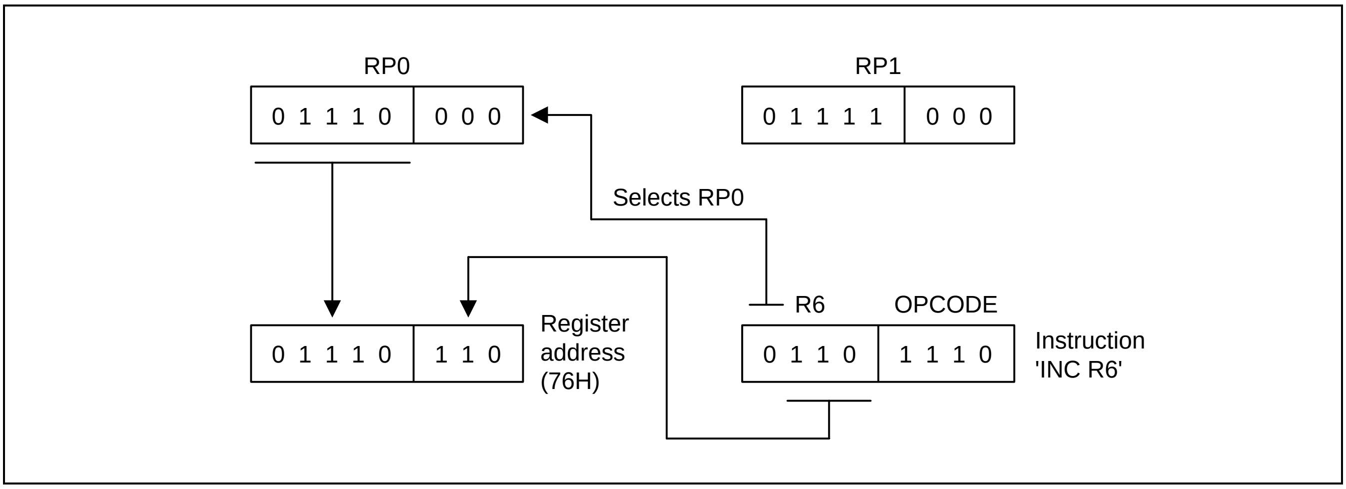


Figure 2-14. 4-Bit Working Register Addressing Example

8-BIT WORKING REGISTER ADDRESSING

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

As shown in Figure 2-15, the lower nibble of the 8-bit address is concatenated in much the same way as for 4-bit addressing: Bit 3 selects either RP0 or RP1, which then supplies the five high-order bits of the final address; the three low-order bits of the complete address are provided by the original instruction.

Figure 2-16 shows an example of 8-bit working register addressing. The four high-order bits of the instruction address (1100B) specify 8-bit working register addressing. Bit 4 ("1") selects RP1 and the five high-order bits in RP1 (10101B) become the five high-order bits of the register address. The three low-order bits of the register address (011) are provided by the three low-order bits of the 8-bit instruction address. The five address bits from RP1 and the three address bits from the instruction are concatenated to form the complete register address, 0ABH (10101011B).

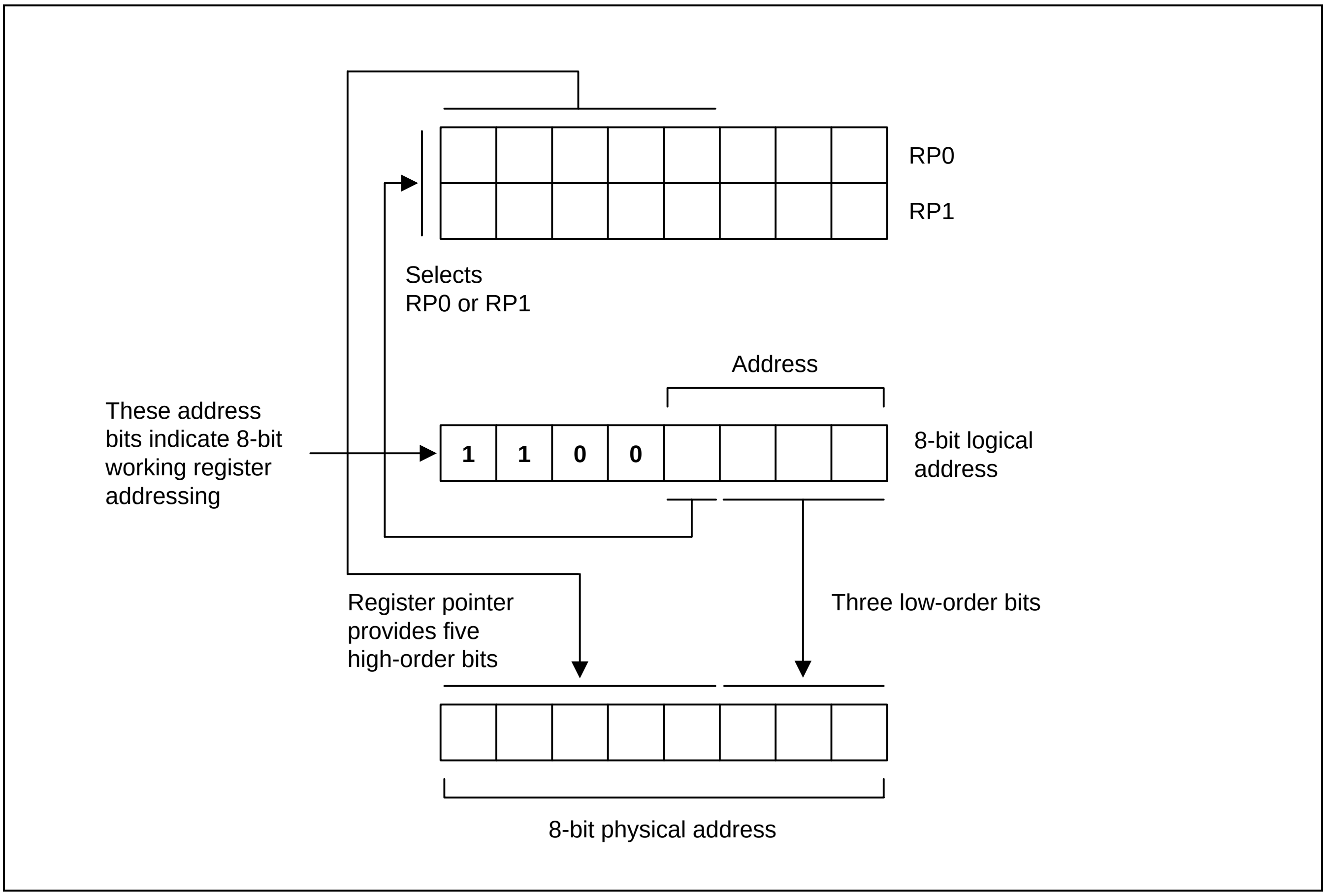


Figure 2-15. 8-Bit Working Register Addressing

SAMSUNG ELECTRONICS

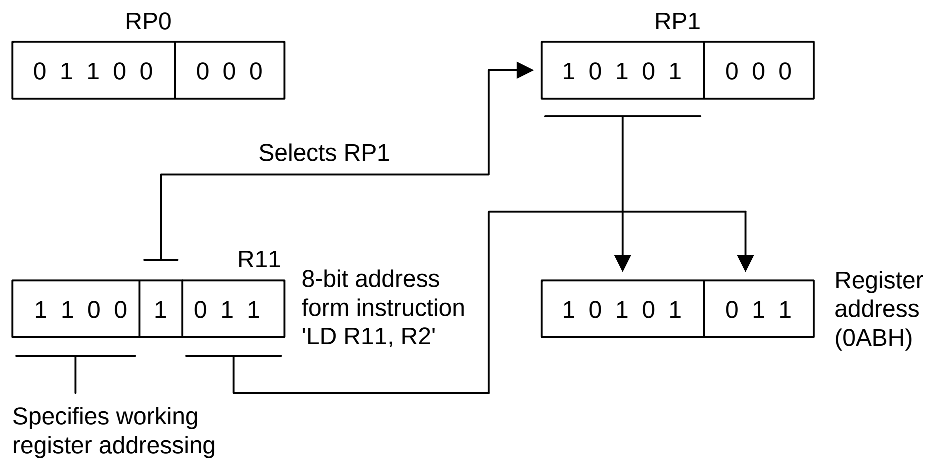


Figure 2-16. 8-Bit Working Register Addressing Example

SYSTEM AND USER STACK

The S3C8-series microcontrollers use the system stack for data storage, subroutine calls and returns. The PUSH and POP instructions are used to control system stack operations. The S3C8275X/C8278X/C8274X architecture supports stack operations in the internal register file.

Stack Operations

Return addresses for procedure calls, interrupts, and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the PC and the FLAGS register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address value is always decreased by one before a push operation and increased by one *after* a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of the stack, as shown in Figure 2-17.

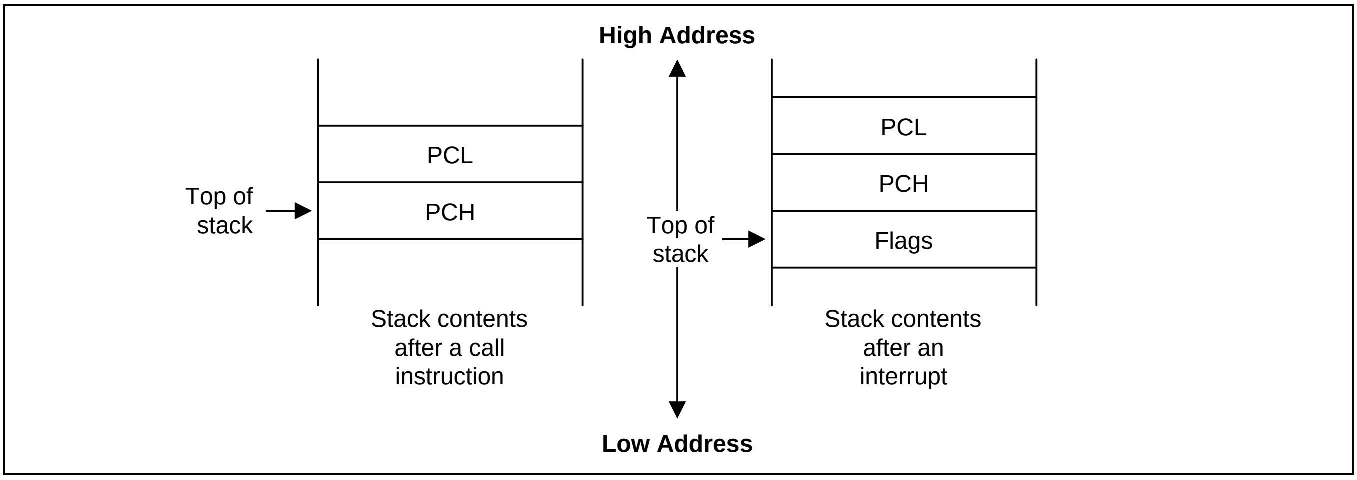


Figure 2-17. Stack Operations

User-Defined Stacks

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

Stack Pointers (SPL, SPH)

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

Because only internal memory space is implemented in the S3C8275X/C8278X/C8274X, the SPL must be initialized to an 8-bit value in the range 00H–FFH. The SPH register is not needed and can be used as a general-purpose register, if necessary.

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

☞ Programming TIP — Standard Stack Operations Using PUSH and POP

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

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

3 ADDRESSING MODES

OVERVIEW

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

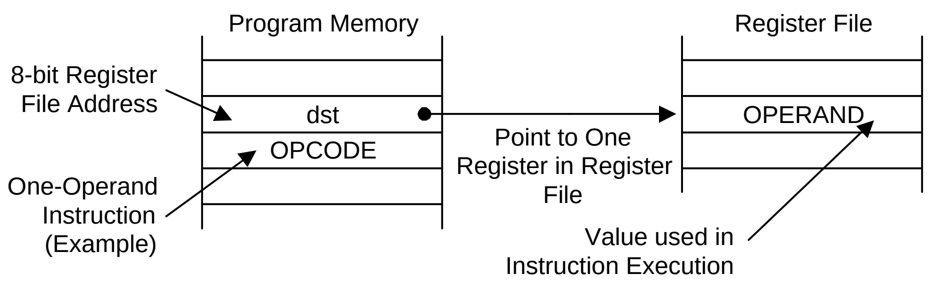
The S3C8-series instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction. The seven addressing modes and their symbols are:

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

REGISTER ADDRESSING MODE (R)

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

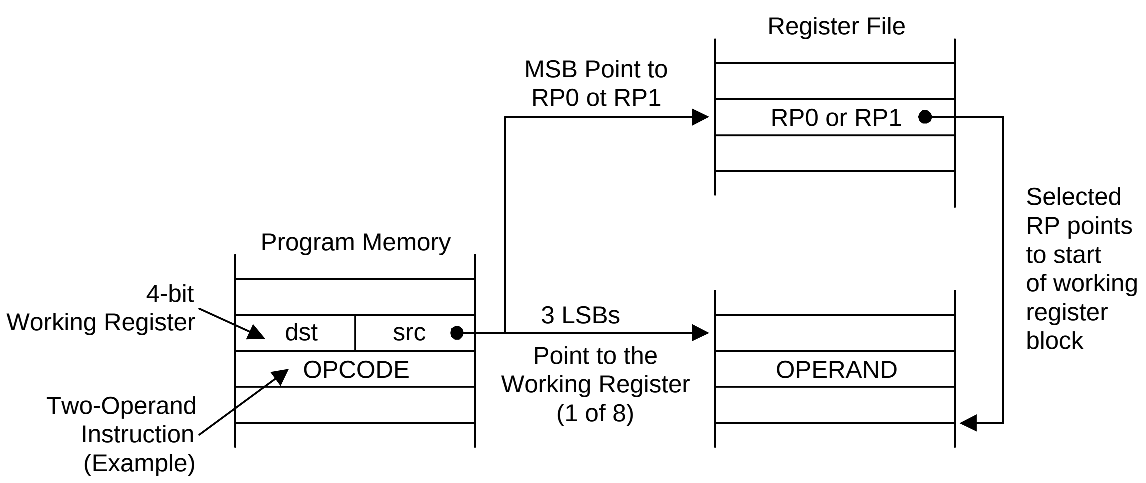
Working register addressing differs from Register addressing in that it uses a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space (see Figure 3-2).



Sample Instruction:

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

Figure 3-1. Register Addressing



Sample Instruction:

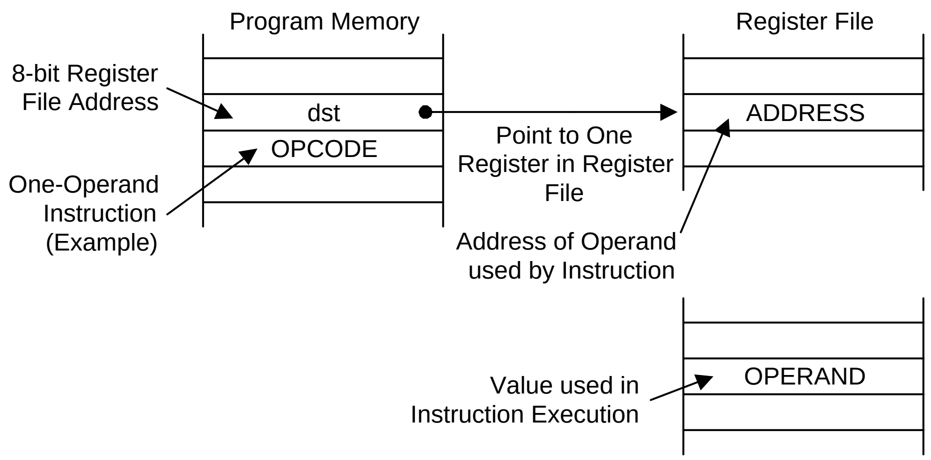
ADD R1, R2 ; Where R1 and R2 are registers in the currently selected working register area.

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. Please note, however, that you cannot access locations C0H–FFH in set 1 using the Indirect Register addressing mode.



Sample Instruction:

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

Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

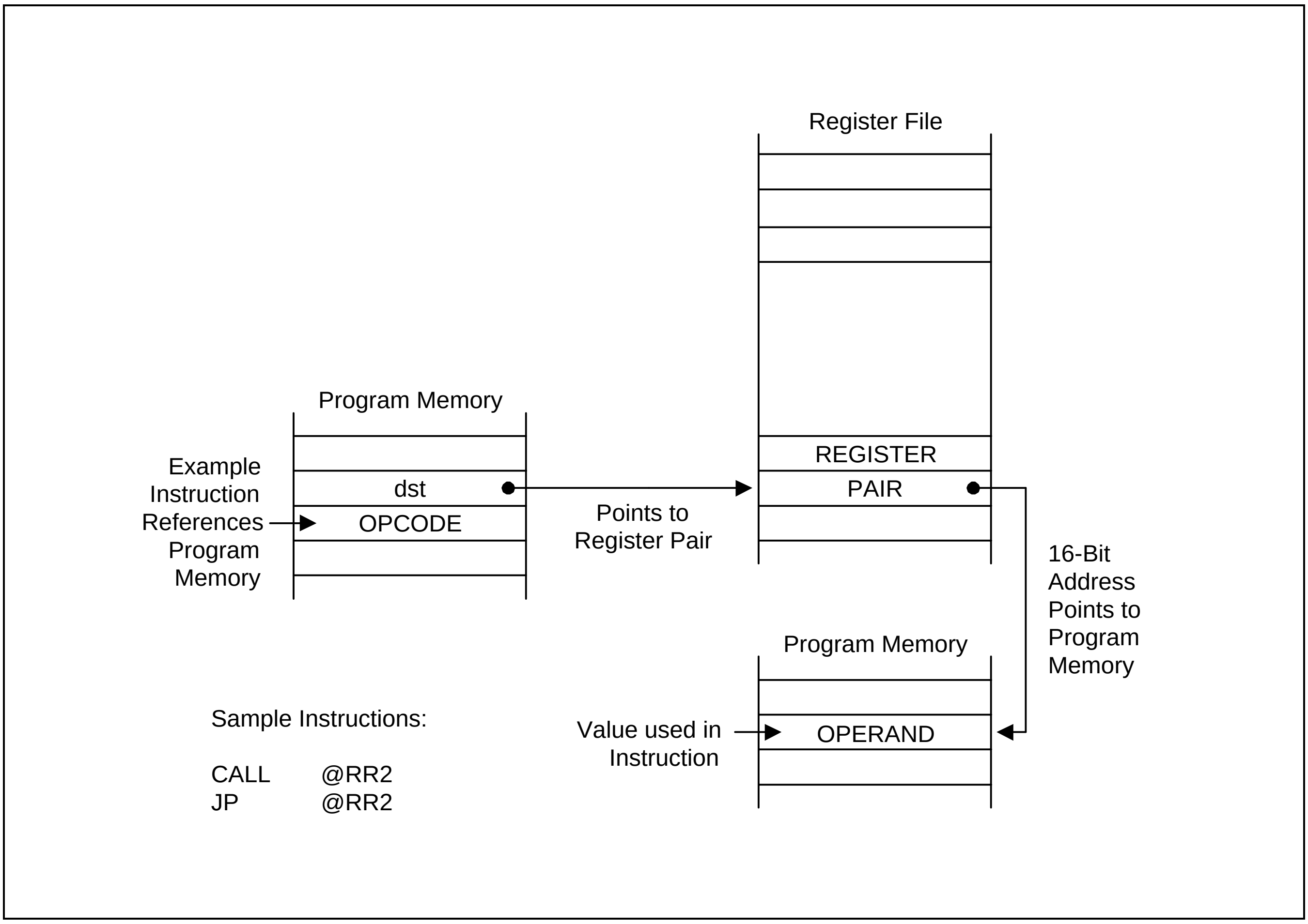


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

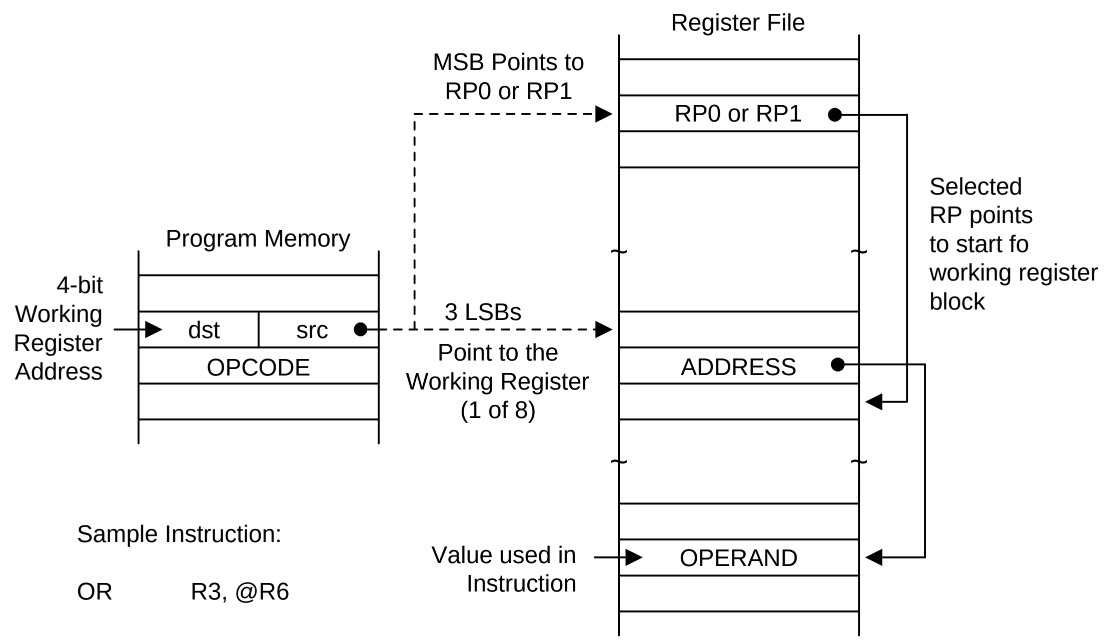
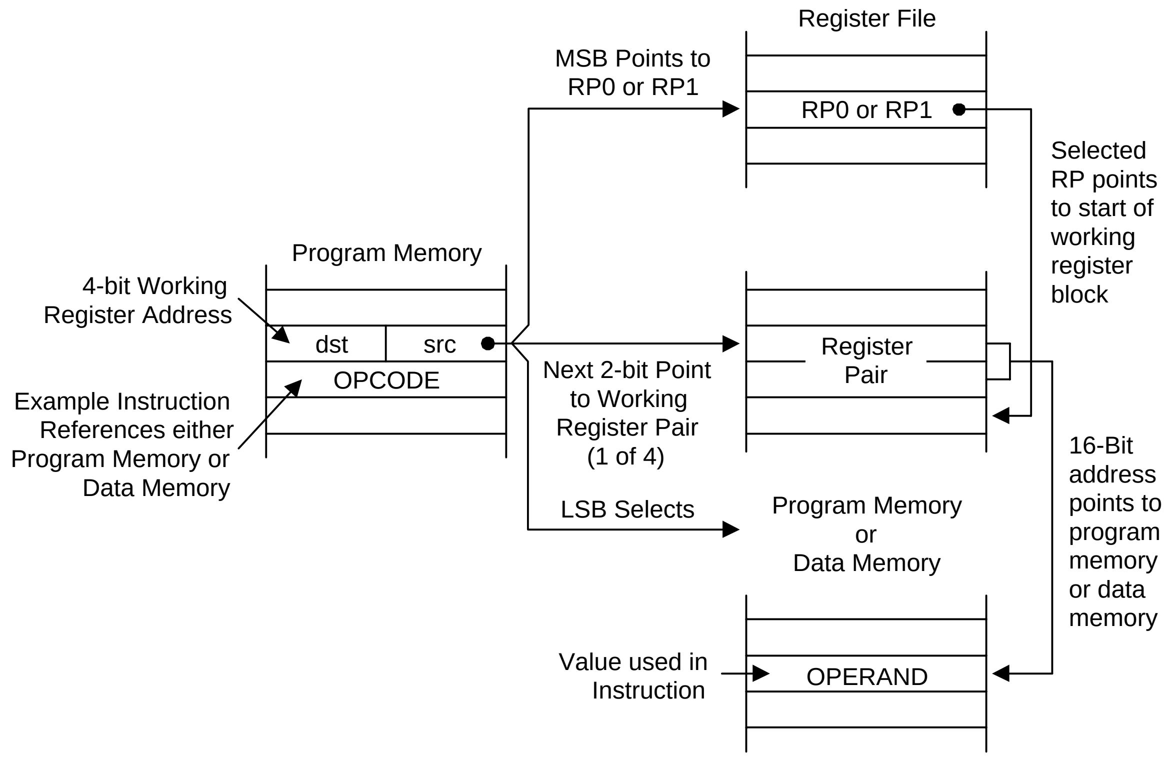


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)



Sample Instructions:

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

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

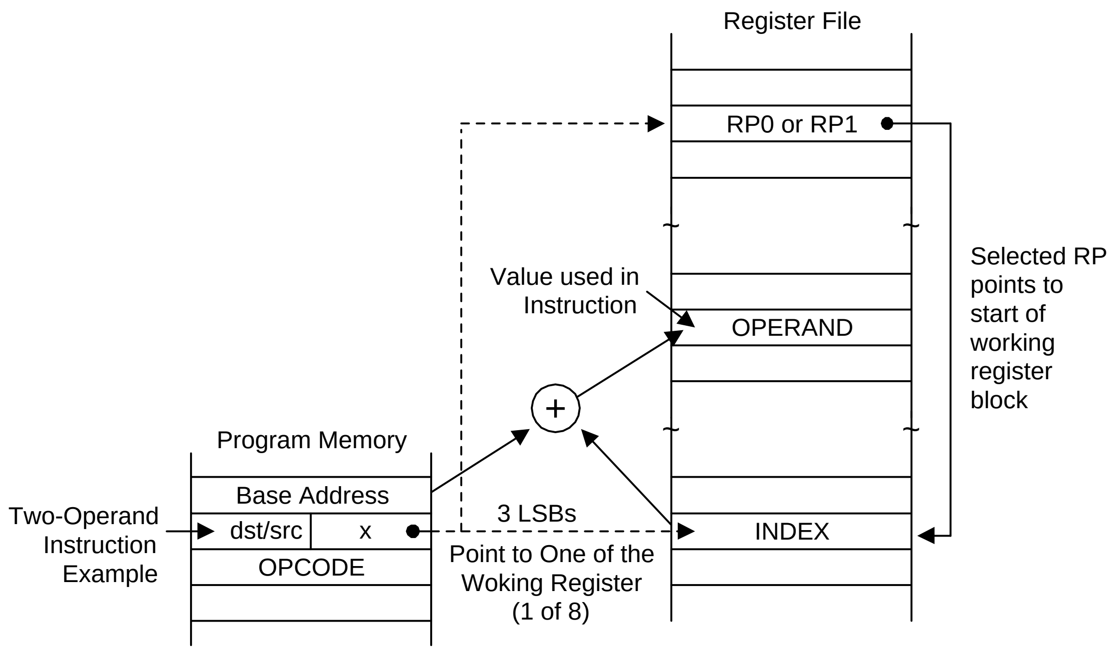
INDEXED ADDRESSING MODE (X)

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

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

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

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

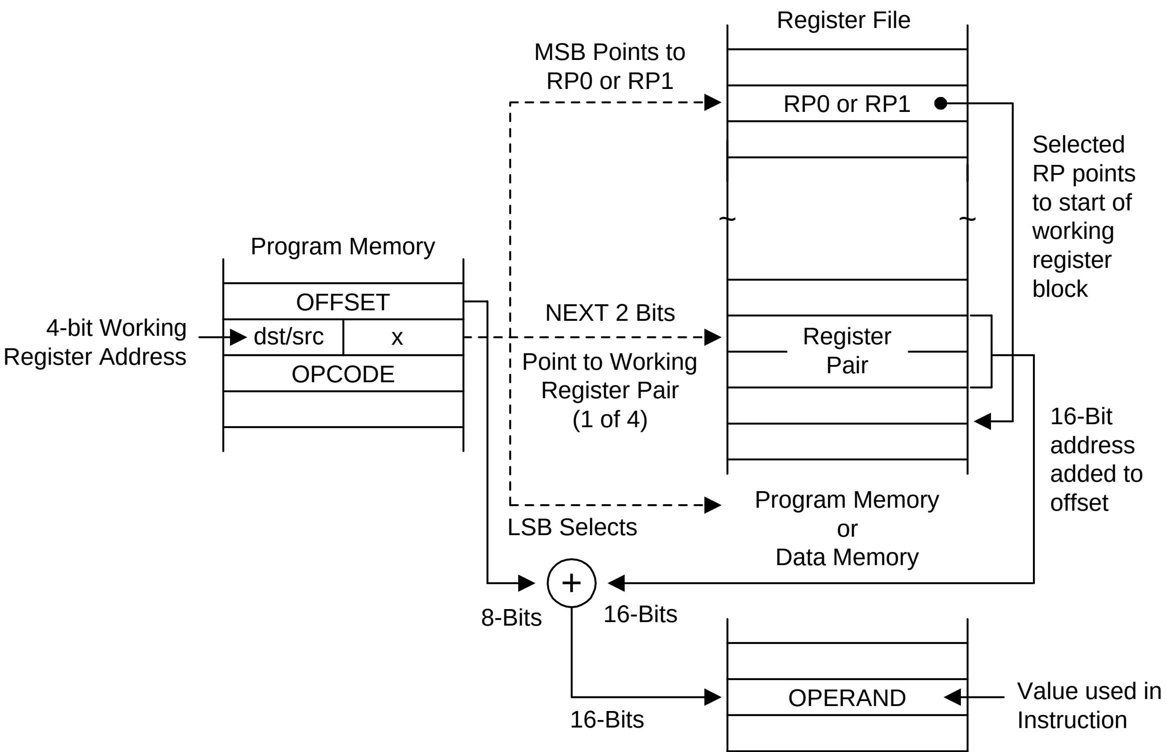


Sample Instruction:

LD R0, #BASE[R1] ; Where BASE is an 8-bit immediate value

Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)

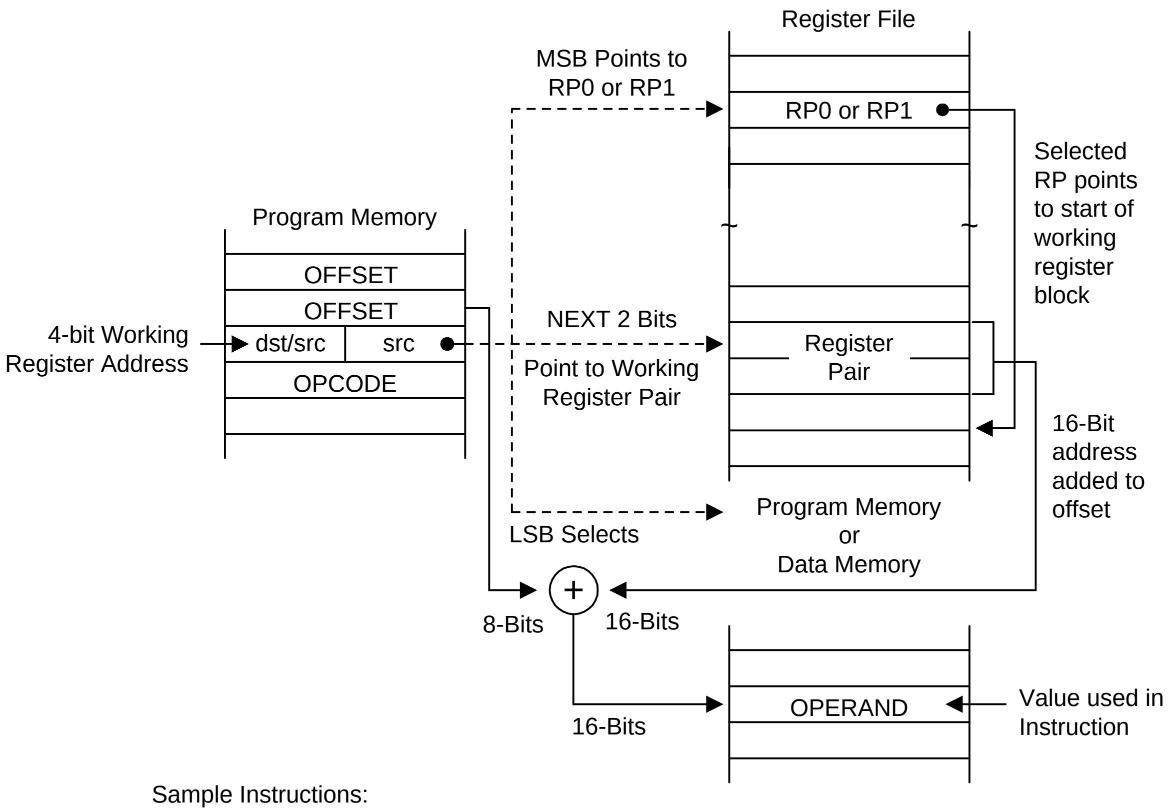


Sample Instructions:

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

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

INDEXED ADDRESSING MODE (Concluded)



Sample Instructions:

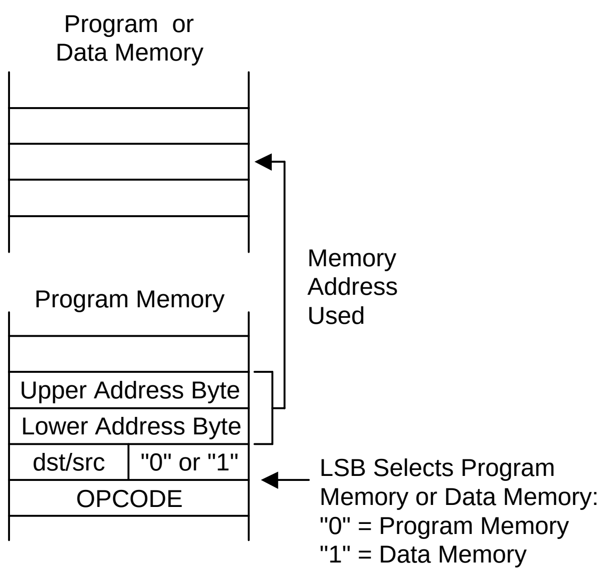
LDC	R4, #1000H[RR2]	; The values in the program address (RR2 + 1000H) are loaded into register R4.
LDE	R4,#1000H[RR2]	; Identical operation to LDC example, except that external program memory is accessed.

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

DIRECT ADDRESS MODE (DA)

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

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

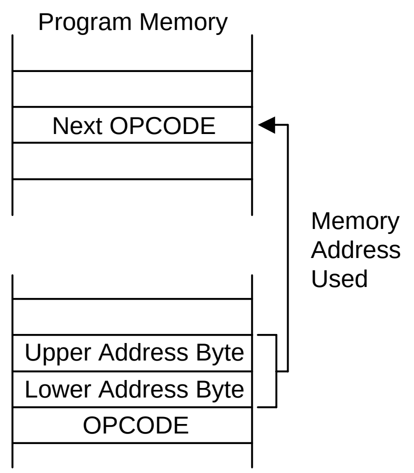


Sample Instructions:

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

Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)



Sample Instructions:

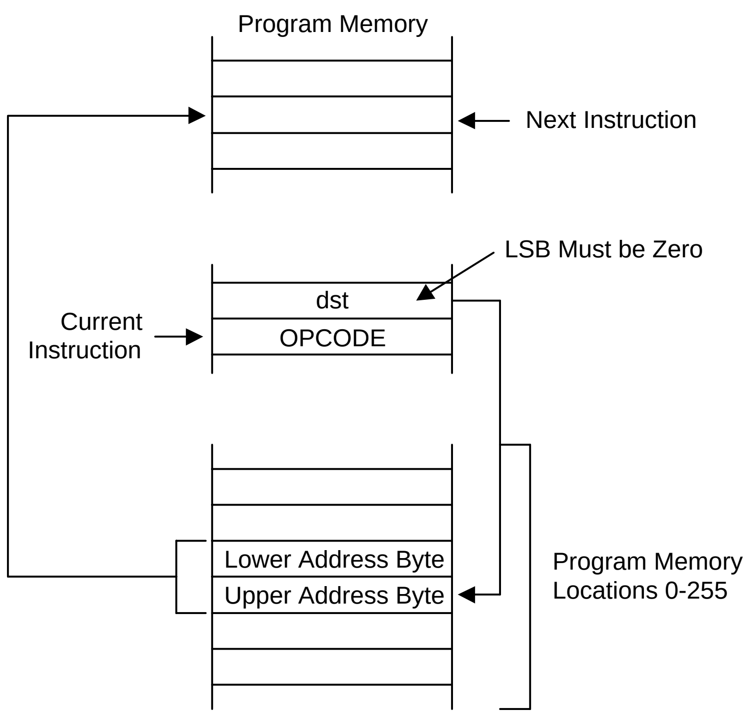
```
JP      C,JOB1          ;   Where JOB1 is a 16-bit immediate address
CALL    DISPLAY         ;   Where DISPLAY is a 16-bit immediate address
```

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

INDIRECT ADDRESS MODE (IA)

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

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



Sample Instruction:

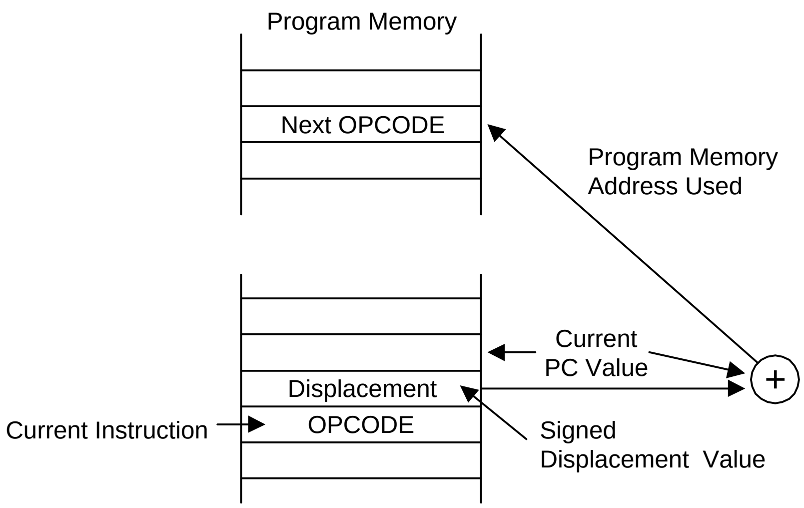
CALL	#40H	;	The 16-bit value in program memory addresses 40H and 41H is the subroutine start address.

Figure 3-12. Indirect Addressing

RELATIVE ADDRESS MODE (RA)

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

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



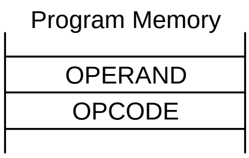
Sample Instructions:

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

Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

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



(The Operand value is in the instruction)

Sample Instruction:

LD R0,#0AAH

Figure 3-14. Immediate Addressing

SAMSUNG ELECTRONICS

4 CONTROL REGISTERS

OVERVIEW

In this chapter, detailed descriptions of the S3C8275X/C8278X/C8274X control registers are presented in an easy-to-read format. You can use this chapter as a quick-reference source when writing application programs. Figure 4-1 illustrates the important features of the standard register description format.

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

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

The locations and read/write characteristics of all mapped registers in the S3C8275X/C8278X/C8274X register file are listed in Table 4-1. The hardware reset value for each mapped register is described in Chapter 8, "RESET and Power-Down."

Table 4-1. Set 1 Registers

Register Name	Mnemonic	Address		R/W
		Decimal	Hex	
Locations D0H – D2H are not mapped.				
Basic timer control register	BTCON	211	D3H	R/W
System clock control register	CLKCON	212	D4H	R/W
System flags register	FLAGS	213	D5H	R/W
Register pointer 0	RP0	214	D6H	R/W
Register pointer 1	RP1	215	D7H	R/W
Stack pointer (high byte)	SPH	216	D8H	R/W
Stack pointer (low byte)	SPL	217	D9H	R/W
Instruction pointer (high byte)	IPH	218	DAH	R/W
Instruction pointer (low byte)	IPL	219	DBH	R/W
Interrupt request register	IRQ	220	DCH	R
Interrupt mask register	IMR	221	DDH	R/W
System mode register	SYM	222	DEH	R/W
Register page pointer	PP	223	DFH	R/W

Table 4-2. Set 1, Bank 0 Registers

Register Name	Mnemonic	Address		R/W
		Decimal	Hex	
Oscillator control register	OSCCON	224	E0H	R/W
SIO control register	SIOCON	225	E1H	R/W
SIO data register	SIODATA	226	E2H	R/W
SIO pre-scaler register	SIOPS	227	E3H	R/W
Port 0 control register (high byte)	P0CONH	228	E4H	R/W
Port 0 control register (low byte)	P0CONL	229	E5H	R/W
Port 0 pull-up resistor enable register	P0PUR	230	E6H	R/W
Port 1 control register (high byte)	P1CONH	231	E7H	R/W
Port 1 control register (low byte)	P1CONL	232	E8H	R/W
Port 1 pull-up resistor enable register	P1PUR	233	E9H	R/W
Port 2 control register (high byte)	P2CONH	234	EAH	R/W
Port 2 control register (low byte)	P2CONL	235	EBH	R/W
Port 2 pull-up resistor enable register	P2PUR	236	ECH	R/W
Port 3 control register (high byte)	P3CONH	237	EDH	R/W
Port 3 control register (low byte)	P3CONL	238	EEH	R/W
Port 3 Pull-up resistor enable register	P3PUR	239	EFH	R/W
Port 0 data register	P0	240	F0H	R/W
Port 1 data register	P1	241	F1H	R/W
Port 2 data register	P2	242	F2H	R/W
Port 3 data register	P3	243	F3H	R/W
Port 4 data register	P4	244	F4H	R/W
Port 5 data register	P5	245	F5H	R/W
Port 6 data register	P6	246	F6H	R/W
External interrupt pending register	EXTIPND	247	F7H	R/W
External interrupt control register (high byte)	EXTICONH	248	F8H	R/W
External interrupt control register (low byte)	EXTICONL	249	F9H	R/W
Locations FAH are not mapped.				
STOP control register	STPCON	251	FBH	R/W
Locations FCH are not mapped.				
Basic timer counter	BTCNT	253	FDH	R
Locations FEH are not mapped.				
Interrupt priority register	IPR	255	FFH	R/W

Table 4-3. Set 1, Bank 1 Registers

Register Name	Mnemonic	Address		R/W
		Decimal	Hex	
LCD control Register	LCON	224	E0H	R/W
Watch timer control register	WTCON	225	E1H	R/W
Timer A counter	TACNT	226	E2H	R
Timer B counter	TBCNT	227	E3H	R
Timer A data register	TADATA	228	E4H	R/W
Timer B data register	TBDATA	229	E5H	R/W
Timer 1/A control register	TACON	230	E6H	R/W
Timer B control register	TBCON	231	E7H	R/W
Clock output control register	CLOCON	232	E8H	R/W
Port 4 control register (high byte)	P4CONH	233	E9H	R/W
Port 4 control register (low byte)	P4CONL	234	EAH	R/W
Port 5 control register (high byte)	P5CONH	235	EBH	R/W
Port 5 control register (low byte)	P5CONL	236	ECH	R/W
Port 6 control register	P6CON	237	EDH	R/W
Locations EEH – EFH are not mapped.				
Flash memory control register	FMCON	240	F0H	R/W
Flash memory user programming enable register	FMUSR	241	F1H	R/W
Flash memory sector address register (high byte)	FMSECH	242	F2H	R/W
Flash memory sector address register (low byte)	FMSECL	243	F3H	R/W
Battery level detector control register	BLDCON	244	F4H	R/W
Locations E5H – FFH are not mapped.				

NOTES:
1. An “x” means that the bit value is undefined following reset.
2. A dash(“–“) means that the bit is neither used nor mapped, but the bit is read as “0”.

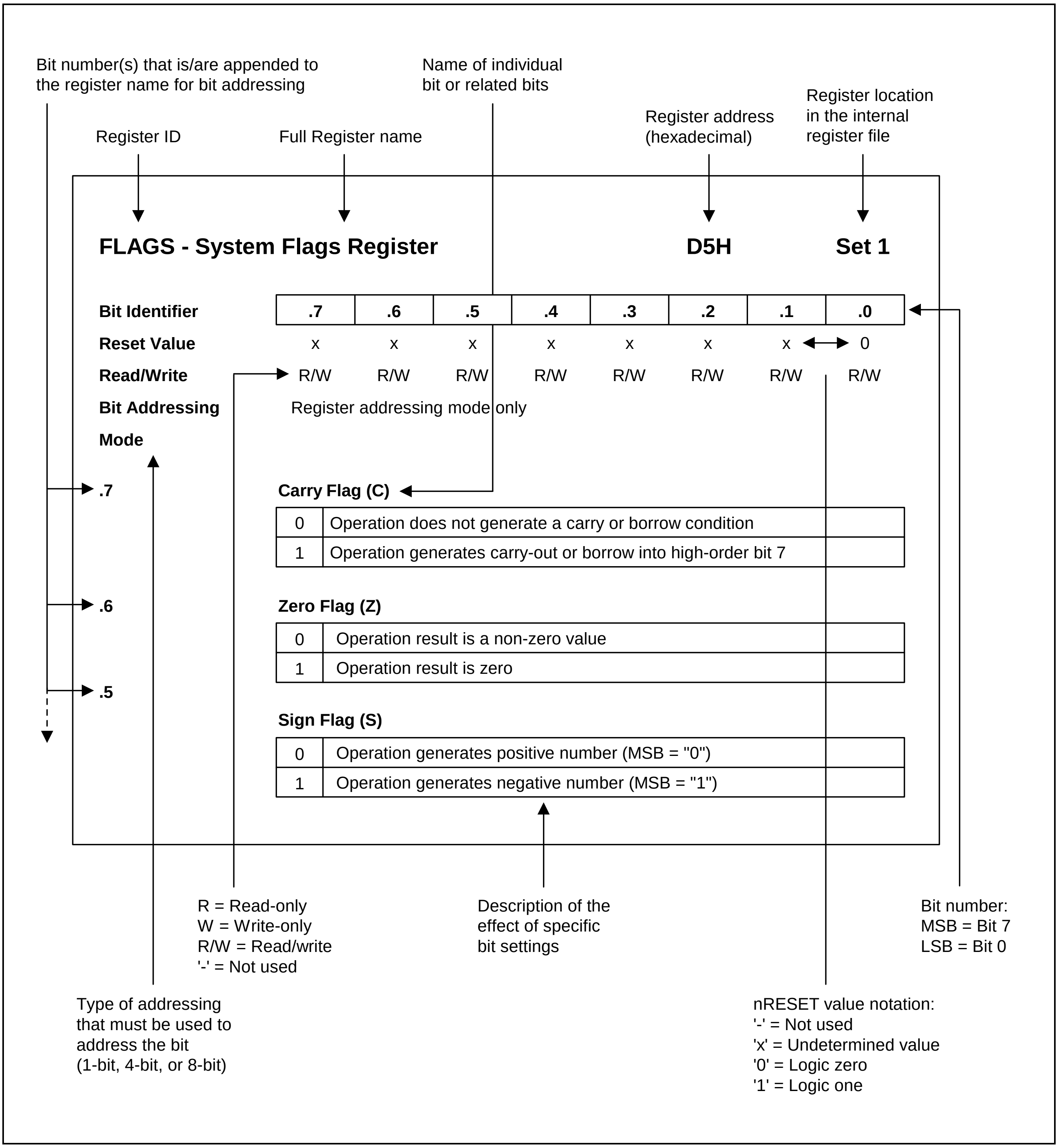


Figure 4-1. Register Description Format

SAMSUNG ELECTRONICS

BLDCON — Battery Level Detector Control Register F4H Set 1, Bank 1

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

Addressing Mode Register addressing mode only

.7–.6 Not used for the S3C8275X/C8278X/C8274X

.5 **V_{IN} Source Bit**

0	Internal source
1	External source

.4 **Battery Level Detector Output Bit**

0	$V_{IN} > V_{REF}$ (when BLD is enabled)
1	$V_{IN} < V_{REF}$ (when BLD is enabled)

.3 **Battery Level Detector Enable/Disable Bit**

0	Disable BLD
1	Enable BLD

.2–.0 **Detection Voltage Selection Bits**

0	0	0	V_{BLD} = 2.2V
1	0	1	V_{BLD} = 2.4V
0	1	1	V_{BLD} = 2.8V
Other values			Not available

BTCON — Basic Timer Control Register D3H Set 1

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

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

1	0	1	0	Disable watchdog timer function
Other values				Enable watchdog timer function

.3–.2 **Basic Timer Input Clock Selection Bits**

0	0	fxx/4096 [3]
0	1	fxx/1024
1	0	fxx/128
1	1	fxx/16

.1 **Basic Timer Counter Clear Bit [1]**

0	No effect
1	Clear the basic timer counter value

.0 **Clock Frequency Divider Clear Bit for Basic Timer and Timer/Counters [2]**

0	No effect
1	Clear both clock frequency dividers

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

CLKCON — System Clock Control Register D4H Set 1

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

.7 **Oscillator IRQ Wake-up Function Bit**

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

.6–.5 Not used for the S3C8275X/C8278X/C8274X (must keep always “0”)

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

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

.2–.0 Not used for the S3C8275X/C8278X/C8274X (must keep always “0”)

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

CLOCON — Clock Output Control Register E8H Set 1, Bank 1

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

Addressing Mode Register addressing mode only

.7–.2 Not used for the S3C8275X/C8278X/C8274X (must keep always "0")

.1–.0 **Clock Output Frequency Selection Bits**

0	0	Select fxx/64
0	1	Select fxx/16
1	0	Select fxx/8
1	1	Select fxx/4

EXTICONH — External Interrupt Control Register (High Byte) F8H Set 1, Bank 0

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

.7–.6 **P1.7 External Interrupt (INT7) Configuration Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.5–.4 **P1.6 External Interrupt (INT6) Configuration Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.3–.2 **P1.5 External Interrupt (INT5) Configuration Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.1–.0 **P1.4 External Interrupt (INT4) Configuration Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

EXTICONL — External Interrupt Control Register (Low Byte) F9H Set 1, Bank 0

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

.7–.6 **P1.3 External Interrupt (INT3) Configuration Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.5–.4 **P0.2 External Interrupt (INT2) Configuration Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.3–.2 **P0.1 External Interrupt (INT1) Configuration Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

.1–.0 **P0.0 External Interrupt (INT0) Configuration Bits**

0	0	Disable interrupt
0	1	Enable interrupt by falling edge
1	0	Enable interrupt by rising edge
1	1	Enable interrupt by both falling and rising edge

EXTIPND — External Interrupt Pending Register F7H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 **P1.7/INT7 Interrupt Pending Bit**

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

.6 **P1.6/INT6 Interrupt Pending Bit**

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

.5 **P1.5/INT5 Interrupt Pending Bit**

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

.4 **P1.4/INT4 Interrupt Pending Bit**

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

.3 **P1.3/INT3 Interrupt Pending Bit**

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

.2 **P0.2/INT2 Interrupt Pending Bit**

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

.1 **P0.1/INT1 Interrupt Pending Bit**

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

.0 **P0.0/INT0 Interrupt Pending Bit**

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

FLAGS — System Flags Register D5H Set 1

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

Addressing Mode Register addressing mode only

.7 **Carry Flag (C)**

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

.6 **Zero Flag (Z)**

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

.5 **Sign Flag (S)**

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

.4 **Overflow Flag (V)**

0	Operation result is $\leq$ +127 or $\geq$ –128
1	Operation result is $>$ +127 or $<$ –128

.3 **Decimal Adjust Flag (D)**

0	Add operation completed
1	Subtraction operation completed

.2 **Half-Carry Flag (H)**

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

.1 **Fast Interrupt Status Flag (FIS)**

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

.0 **Bank Address Selection Flag (BA)**

0	Bank 0 is selected
1	Bank 1 is selected

FMCON — Flash Memory Control Register F0H Set 1, Bank 1

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	–	–	R/W
Addressing Mode	Register addressing mode only							

.7–.4 **Flash Memory Mode Selection Bits**

0	1	0	1	Programming mode
1	0	1	0	Sector erase mode
0	1	1	0	Hard lock mode
Other values				Not available

.3 **Sector Erase Status Bit**

0	Success sector erase
1	Fail sector erase

.2–.1 Not used for the S3F8275X/F8278X/F8274X

.0 **Flash Operation Start Bit**

0	Operation stop
1	Operation start (This bit will be cleared automatically just after the corresponding operator completed).

FMSECH — Flash Memory Sector Address Register (High Byte) F2H Set 1, Bank 1

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

.7–.0 **Flash Memory Sector Address Bits (High Byte)**

The 15th - 8th bits to select a sector of flash ROM

NOTE: The high-byte flash memory sector address pointer value is the higher eight bits of the 16-bit pointer address.

FMSECL — Flash Memory Sector Address Register (Low Byte) F3H Set 1, Bank 1

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

.7 **Flash Memory Sector Address Bit (Low Byte)**

The 7th bit to select a sector of flash ROM

.6–.0 **Bits 6–0**

Don't care

NOTE: The low-byte flash memory sector address pointer value is the lower eight bits of the 16-bit pointer address.

FMUSR — Flash Memory User Programming Enable Register F1H Set 1, Bank 1

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

.7–.0 **Flash Memory User Programming Enable Bits**

1 0 1 0 0 1 0 1	Enable user programming mode
Other values	Disable user programming mode

IMR — Interrupt Mask Register DDH Set 1

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

Addressing Mode Register addressing mode only

.7 **Interrupt Level 7 (IRQ7) Enable Bit; External Interrupts P1.4–1.7**

0	Disable (mask)
1	Enable (unmask)

.6 **Interrupt Level 6 (IRQ6) Enable Bit; External Interrupts P1.3**

0	Disable (mask)
1	Enable (unmask)

.5 **Interrupt Level 5 (IRQ5) Enable Bit; External Interrupt P0.2**

0	Disable (mask)
1	Enable (unmask)

.4 **Interrupt Level 4 (IRQ4) Enable Bit; External Interrupt P0.1**

0	Disable (mask)
1	Enable (unmask)

.3 **Interrupt Level 3 (IRQ3) Enable Bit; External Interrupt P0.0**

0	Disable (mask)
1	Enable (unmask)

.2 **Interrupt Level 2 (IRQ2) Enable Bit; Watch Timer Overflow**

0	Disable (mask)
1	Enable (unmask)

.1 **Interrupt Level 1 (IRQ1) Enable Bit; SIO Interrupt**

0	Disable (mask)
1	Enable (unmask)

.0 **Interrupt Level 0 (IRQ0) Enable Bit; Timer 1/A Match, Timer B Match**

0	Disable (mask)
1	Enable (unmask)

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

IPH — Instruction Pointer (High Byte) DAH Set 1

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

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

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

IPL — Instruction Pointer (Low Byte) DBH Set 1

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

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

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

IPR — Interrupt Priority Register FFH Set 1, Bank 0

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

Addressing Mode Register addressing mode only

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

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

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

0	IRQ6 > IRQ7
1	IRQ7 > IRQ6

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

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

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

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

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

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

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

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

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

IRQ — Interrupt Request Register DCH Set 1

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

Addressing Mode Register addressing mode only

.7 **Level 7 (IRQ7) Request Pending Bit; External Interrupt P1.4–1.7**

0	Not pending
1	Pending

.6 **Level 6 (IRQ6) Request Pending Bit; External Interrupt P1.3**

0	Not pending
1	Pending

.5 **Level 5 (IRQ5) Request Pending Bit; External Interrupt P0.2**

0	Not pending
1	Pending

.4 **Level 4 (IRQ4) Request Pending Bit; External Interrupt P0.1**

0	Not pending
1	Pending

.3 **Level 3 (IRQ3) Request Pending Bit; External Interrupt P0.0**

0	Not pending
1	Pending

.2 **Level 2 (IRQ2) Request Pending Bit; Watch Timer Overflow**

0	Not pending
1	Pending

.1 **Level 1 (IRQ1) Request Pending Bit; SIO Interrupt**

0	Not pending
1	Pending

.0 **Level 0 (IRQ0) Request Pending Bit; Timer 1/A Match, Timer B Match**

0	Not pending
1	Pending

LCON — LCD Control Register E0H Set 1, Bank 1

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

Addressing Mode Register addressing mode only

.7 **Internal LCD Dividing Resistors Enable Bit**

0	Enable internal LCD dividing resistors
1	Disable internal LCD dividing resistors

.6–.5 **LCD Clock Selection Bits**

0	0	$fw/2^9$ (64 Hz)
0	1	$fw/2^8$ (128 Hz)
1	0	$fw/2^7$ (256 Hz)
1	1	$fw/2^6$ (512 Hz)

.4–.2 **LCD Duty and Bias Selection Bits**

0	0	0	1/4duty, 1/3bias
0	0	1	1/3duty, 1/3bias
0	1	0	1/3duty, 1/2bias
0	1	1	1/2duty, 1/2bias
1	x	x	Static

NOTES:
1. "x" means don't care.
2. When 1/2 bias is selected, the bias levels are set as V_{LC0}, V_{LC1} (V_{LC2}), and V_{SS}.

.1

Not used for the S3C8275X/C8278X/C8274X

.0 **LCD Display Control Bit**

0	Turn display off (Turn off the P-Tr)
1	Turn display on (Turn on the P-Tr)

SAMSUNG ELECTRONICS

OSCCON — Oscillator Control Register E0H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 **Sub Oscillator Circuit Selection Bit**

0	Initial state
1	Power saving circuit for sub oscillator (Automatically cleared to "0" when the sub oscillator is stopped by OSCCON.2).

NOTES:
1. The OSCCON.7 must be maintained to “1”, during the sub oscillator operation.
2. A capacitor (0.1uF) should be connected between V_{REG} and GND.

.6–.4 Not used for the S3C8275X/C8278X/C8274X

.3 **Main Oscillator Control Bit**

0	Main oscillator RUN
1	Main oscillator STOP

.2 **Sub Oscillator Control Bit**

0	Sub oscillator RUN
1	Sub oscillator STOP

.1 Not used for the S3C8275X/C8278X/C8274X

.0 **System Clock Selection Bit**

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

P0CONH — Port 0 Control Register (High Byte) E4H Set 1, Bank 0

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

.7–.6 **P0.7/BUZ Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (BUZ)

.5–.4 **P0.6/CLKOUT Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (CLKOUT)

.3–.2 **P0.5/TBOUT Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (TBOUT)

.1–.0 **P0.4/TAOUT Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (TAOUT)

P0CONL — Port 0 Control Register (Low Byte) E5H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7–.6 **P0.3/T1CLK Configuration Bits**

0	0	Schmitt trigger input mode (T1CLK)
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

.5–.4 **P0.2/INT2 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

.3–.2 **P0.1/INT1 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

.1–.0 **P0.0/INT0 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

P0PUR — Port 0 Pull-Up Control Register E6H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 **P0.7's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.6 **P0.6's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.5 **P0.5's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.4 **P0.4's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.3 **P0.3's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.2 **P0.2's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.1 **P0.1's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.0 **P0.0's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: A pull-up resistor of port 0 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

P1CONH — Port 1 Control Register (High Byte) E7H Set 1, Bank 0

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

.7–.6 **P1.7/INT7 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

.5–.4 **P1.6/INT6 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

.3–.2 **P1.5/INT5 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

.1–.0 **P1.4/INT4 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

P1CONL — Port 1 Control Register (Low Byte) E8H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7–.6 **P1.3/INT3 Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

.5–.4 **P1.2/SI Configuration Bits**

0	0	Schmitt trigger input mode (SI)
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Not used for the S3C8275X/C8278X/C8274X

.3–.2 **P1.1/SO Configuration Bits**

0	0	Schmitt trigger input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SO)

.1–.0 **P1.0/SCK Configuration Bits**

0	0	Schmitt trigger input mode (SCK)
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SCK)

P1PUR — Port 1 Pull-up Control Register F9H Set 1, Bank 0

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

.7 **P1.7's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.6 **P1.6's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.5 **P1.5's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.4 **P1.4's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.3 **P1.3's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.2 **P1.2's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.1 **P1.1's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.0 **P1.0's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: A pull-up resistor of port 1 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

P2CONH — Port 2 Control Register (High Byte) EAH Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7–.6 **P2.7/SEG24 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG24)

.5-.4 **P2.6/SEG25 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG25)

.3–.2 **P2.5/SEG26 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG26)

.1–.0 **P2.4/SEG27 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG27)

P2CONL — Port 2 Control Register (Low Byte) EBH Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7–.6 **P2.3/SEG28 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG28)

.5–.4 **P2.2/SEG29 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG29)

.3–.2 **P2.1/SEG30 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG30)

.1–.0 **P2.0/SEG31/V_{BLDREF} Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG31 or V_{BLDREF})

P2PUR — Port 2 Pull-up Control Register ECH Set 1, Bank 0

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

.7 **P2.7's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.6 **P2.6's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.5 **P2.5's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.4 **P2.4's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.3 **P2.3's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.2 **P2.2's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.1 **P2.1's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.0 **P2.0's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: A pull-up resistor of port 2 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

P3CONH — Port 3 Control Register (High Byte) EDH Set 1, Bank 0

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

.7–.6 **P3.7/SEG16 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG16)

.5–.4 **P3.6/SEG17 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG17)

.3–.2 **P3.5/SEG18 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG18)

.1–.0 **P3.4/SEG19 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG19)

P3CONL — Port 3 Control Register (Low Byte) EEH Set 1, Bank 0

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

.7–.6 **P3.3/SEG20 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG20)

.5–.4 **P3.2/SEG21 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG21)

.3–.2 **P3.1/SEG22 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG22)

.1–.0 **P3.0/SEG23 Configuration Bits**

0	0	Input mode
0	1	N-channel open-drain output mode
1	0	Push-pull output mode
1	1	Alternative function (SEG23)

P3PUR — Port 3 Pull-up Control Register EFH Set 1, Bank 0

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

.7 **P3.7's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.6 **P3.6's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.5 **P3.5's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.4 **P3.4's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.3 **P3.3's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.2 **P3.2's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.1 **P3.1's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

.0 **P3.0's Pull-up Resistor Enable Bit**

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: A pull-up resistor of port 3 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

P4CONH — Port 4 Control Register (High Byte) E9H Set 1, Bank 1

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

Addressing Mode Register addressing mode only

.7–.6 **P4.7/SEG8 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG8)

.5–.4 **P4.6/SEG9 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG9)

.3–.2 **P4.5/SEG10 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG10)

.1–.0 **P4.4/SEG11 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG11)

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

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

Addressing Mode Register addressing mode only

.7–.6 **P4.3/SEG12 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG12)

.5–.4 **P4.2/SEG13 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG13)

.3–.2 **P4.1/SEG14 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG14)

.1–.0 **P4.0/SEG15 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG15)

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

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

.7–.6 **P5.7/SEG0 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG0)

.5–.4 **P5.6/SEG1 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG1)

.3–.2 **P5.5/SEG2 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG2)

.1–.0 **P5.4/SEG3 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG3)

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

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

.7–.6 **P5.3/SEG4 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG4)

.5–.4 **P5.2/SEG5 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG5)

.3–.2 **P5.1/SEG6 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG6)

.1–.0 **P5.0/SEG7 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (SEG7)

P6CON — Port 6 Control Register EDH Set 1, Bank 1

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

.7–.6 **P6.3/COM3 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (COM3)

.5–.4 **P6.2/COM2 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (COM2)

.3–.2 **P6.1/COM1 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (COM1)

.1–.0 **P6.0/COM0 Configuration Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	Alternative function (COM0)

PP — Register Page Pointer DFH Set 1

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

.7–.4 **Destination Register Page Selection Bits**

0	0	0	0	Destination: page 0
0	0	0	1	Destination: page 1 (Not used for the S3C8278X/C8274X)
0	0	1	0	Destination: page 2
Others				Not used for the S3C8275X/C8278X/C8274X

.3– .0 **Source Register Page Selection Bits**

0	0	0	0	Source: page 0
0	0	0	1	Source: page 1 (Not used for the S3C8278X/C8274X)
0	0	1	0	Source: page 2
Others				Not used for the S3C8275X/C8278X/C8274X

NOTES:
1. In the S3C8275X microcontroller, the internal register file is configured as three pages (Pages 0-2). The pages 0-1 are used for general purpose register file, and page 2 is used for LCD data register or general purpose registers.
2. In the S3C8278X/C8274X microcontroller, the internal register file is configured as two pages (Pages 0, 2). The page 0 is used for general purpose register file, and page 2 is used for LCD data register or general purpose registers.

RP0 — Register Pointer 0 D6H Set 1

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

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

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

.2–.0 Not used for the S3C8275X/C8278X/C8274X

RP1 — Register Pointer 1 D7H Set 1

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

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

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

.2– .0 Not used for the S3C8275X/C8278X/C8274X

 SAMSUNG ELECTRONICS

SIOCON — SIO Control Register E1H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 **SIO Shift Clock Selection Bit**

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

.6 **Data Direction Control Bit**

0	MSB-first mode
1	LSB-first mode

.5 **SIO Mode Selection Bit**

0	Receive-only mode
1	Transmit/receive mode

.4 **Shift Clock Edge Selection Bit**

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

.3 **SIO Counter Clear and Shift Start Bit**

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

.2 **SIO Shift Operation Enable Bit**

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

.1 **SIO Interrupt Enable Bit**

0	Disable SIO interrupt
1	Enable SIO interrupt

.0 **SIO Interrupt Pending Bit**

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

SPH — Stack Pointer (High Byte) D8H Set 1

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

Addressing Mode Register addressing mode only

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

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

SPL — Stack Pointer (Low Byte) D9H Set 1

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

Addressing Mode Register addressing mode only

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

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

STPCON — Stop Control Register FBH Set 1, Bank 0

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

.7–.0 **STOP Control Bits**

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

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

SYM — System Mode Register DEH Set 1

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

Addressing Mode Register addressing mode only

.7

This bit must remain logic "0"

.6–.5

Not used for the S3C8275X/C8278X/C8274X

.4–.2

Fast Interrupt Level Selection Bits (1)

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

.1

Fast Interrupt Enable Bit (2)

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0

Global Interrupt Enable Bit (3)

0	Disable all interrupt processing
1	Enable all interrupt processing

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

TACON — Timer 1/A Control Register E6H Set 1, Bank 1

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

.7 **Timer 1 Operating Mode Selection Bit**

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

.6–.4 **Timer 1/A Clock Selection Bits**

0	0	0	fxx/512
0	0	1	fxx/256
0	1	0	fxx/64
0	1	1	fxx/8
1	0	0	fxx (system clock)
1	0	1	fxt (sub clock)
1	1	0	T1CLK (external clock)
1	1	1	Not available

.3 **Timer 1/A Counter Clear Bit**

0	No effect
1	Clear the timer 1/A counter (when write, automatically cleared to "0" after being cleared basic timer counter)

.2 **Timer 1/A Counter Operating Enable Bit**

0	Disable counting operation
1	Enable counting operation

.1 **Timer 1/A Interrupt Enable Bit**

0	Disable interrupt
1	Enable interrupt

.0 **Timer 1/A Interrupt Pending Bit**

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

TBCON — Timer B Control Register E7H Set 1, Bank 1

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

Addressing Mode Register addressing mode only

.7 Not used for the S3C8275X/C8278X/C8274X

.6–.4 **Timer B Clock Selection Bits**

0	0	0	fxx/512
0	0	1	fxx/256
0	1	0	fxx/64
0	1	1	fxx/8
1	0	0	fxt (sub clock)
Others			Not available

.3 **Timer B Counter Clear Bit**

0	No effect
1	Clear the timer B counter (when write, automatically cleared to "0" after being cleared basic timer counter)

.2 **Timer B Counter Operating Enable Bit**

0	Disable counting operation
1	Enable counting operation

.1 **Timer B Interrupt Enable Bit**

0	Disable interrupt
1	Enable interrupt

.0 **Timer B Interrupt Pending Bit**

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

WTCON — Watch Timer Control Register E1H Set 1, Bank 1

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

Addressing Mode Register addressing mode only

.7 **Watch Timer Clock Selection Bit**

0	Main system clock divided by 2^7 (fx/128)
1	Sub system clock (fxt)

.6 **Watch Timer Interrupt Enable Bit**

0	Disable watch timer interrupt
1	Enable watch timer interrupt

.5–.4 **Buzzer Signal Selection Bits**

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

.3–.2 **Watch Timer Speed Selection Bits**

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

.1 **Watch Timer Enable Bit**

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

.0 **Watch Timer Interrupt Pending Bit**

0	Interrupt is not pending, clear pending bit when write
1	Interrupt is pending

NOTE: Watch timer clock frequency (fw) is assumed to be 32.768 kHz.

5 INTERRUPT STRUCTURE

OVERVIEW

The S3C8-series interrupt structure has three basic components: levels, vectors, and sources. The SAM8RC CPU recognizes up to eight interrupt levels and supports up to 128 interrupt vectors. When a specific interrupt level has more than one vector address, the vector priorities are established in hardware. A vector address can be assigned to one or more sources.

Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight possible interrupt levels: IRQ0–IRQ7, also called level 0–level 7. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. The S3C8275X/C8278X/C8274X interrupt structure recognizes eight interrupt levels.

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

Vectors

Each interrupt level can have one or more interrupt vectors, or it may have no vector address assigned at all. The maximum number of vectors that can be supported for a given level is 128 (The actual number of vectors used for S3C8-series devices is always much smaller). If an interrupt level has more than one vector address, the vector priorities are set in hardware. S3C8275X/C8278X/C8274X uses twelve vectors.

Sources

A source is any peripheral that generates an interrupt. A source can be an external pin or a counter overflow. Each vector can have several interrupt sources. In the S3C8275X/C8278X/C8274X interrupt structure, there are twelve possible interrupt sources.

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

INTERRUPT TYPES

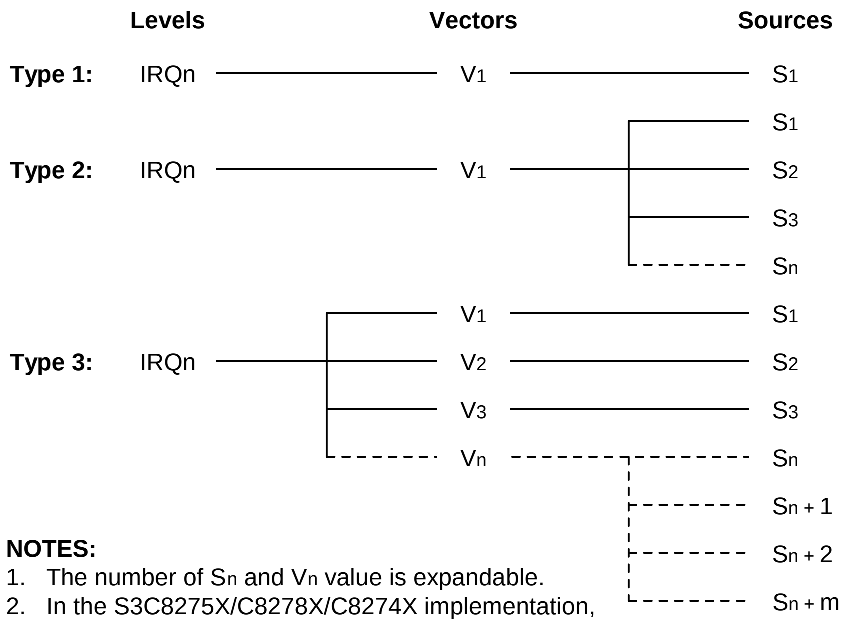
The three components of the S3C8 interrupt structure described before — levels, vectors, and sources — are combined to determine the interrupt structure of an individual device and to make full use of its available interrupt logic. There are three possible combinations of interrupt structure components,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

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

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

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

In the S3C8275X/C8278X/C8274X microcontroller, two interrupt types are implemented.



NOTES:
1. The number of S_n and V_n value is expandable.
2. In the S3C8275X/C8278X/C8274X implementation, interrupt types 1 and 3 are used.

Figure 5-1. S3C8-Series Interrupt Types

SAMSUNG ELECTRONICS

S3C8275X/C8278X/C8274X INTERRUPT STRUCTURE

The S3C8275X/C8278X/C8274X microcontroller supports twelve interrupt sources. All twelve of the interrupt sources have a corresponding interrupt vector address. Eight interrupt levels are recognized by the CPU in this device-specific interrupt structure, as shown in Figure 5-2.

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

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

Levels	Vectors	Sources	Reset/Clear
RESET	100H	Basic timer overflow	H/W
IRQ0	F0H	Timer 1/A match	S/W
	F2H	Timer B match	S/W
IRQ1	F4H	SIO interrupt	S/W
IRQ2	F6H	Watch timer overflow	S/W
IRQ3	E0H	P0.0 external interrupt	S/W
IRQ4	E2H	P0.1 external interrupt	S/W
IRQ5	E4H	P0.2 external interrupt	S/W
IRQ6	E6H	P1.3 external interrupt	S/W
IRQ7	E8H	P1.4 external interrupt	S/W
	EAH	P1.5 external interrupt	S/W
	ECH	P1.6 external interrupt	S/W
	EEH	P1.7 external interrupt	S/W

NOTES:

1. Within a given interrupt level, the low vector address has high priority.
   For example, F0H has higher priority than F2H within the level IRQ0 the priorities within each level are set at the factory.
2. External interrupts are triggered by a rising or falling edge, depending on the corresponding control register setting.

Figure 5-2. S3C8275X/C8278X/C8274X Interrupt Structure

INTERRUPT VECTOR ADDRESSES

All interrupt vector addresses for the S3C8275X/C8278X/C8274X interrupt structure are stored in the vector address area of the internal 16-Kbyte ROM, 0H–3FFFH, or 8, 4-Kbyte (see Figure 5-3).

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

The program reset address in the ROM is 0100H.

The reset address of ROM can be changed by a smart option only in the S3F8275X (Full-Flash Device). Refer to the chapter 16. Embedded Flash Memory Interface for more detail contents.

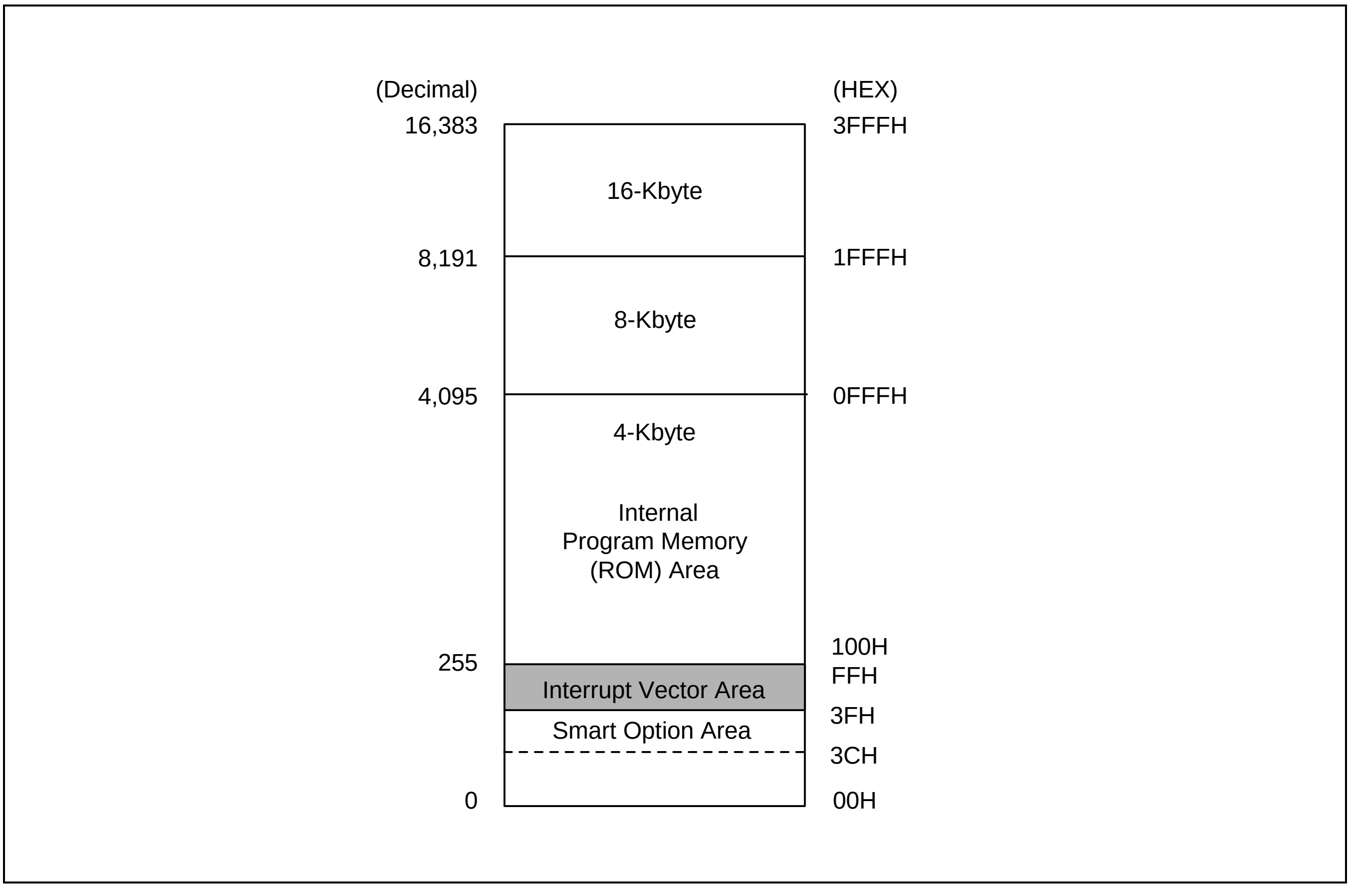


Figure 5-3. ROM Vector Address Area

Table 5-1. Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
256	100H	Basic timer overflow	Reset	–	√	
242	F2H	Timer B match	IRQ0	1		√
240	F0H	Timer 1/A match		0		√
244	F4H	SIO interrupt	IRQ1	–		√
246	F6H	Watch timer overflow	IRQ2	–		√
224	E0H	P0.0 external interrupt	IRQ3	–		√
226	E2H	P0.1 external interrupt	IRQ4	–		√
228	E4H	P0.2 external interrupt	IRQ5	–		√
230	E6H	P1.3 external interrupt	IRQ6	–		√
238	EEH	P1.7 external interrupt	IRQ7	3		√
236	ECH	P1.6 external interrupt		2		√
234	EAH	P1.5 external interrupt		1		√
232	E8H	P1.4 external interrupt		0		√

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

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

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

NOTE

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

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

SYSTEM-LEVEL INTERRUPT CONTROL REGISTERS

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

- The interrupt mask register, IMR, enables (un-masks) or disables (masks) interrupt levels.
- The interrupt priority register, IPR, controls the relative priorities of interrupt levels.
- The interrupt request register, IRQ, contains interrupt pending flags for each interrupt level (as opposed to each interrupt source).
- The system mode register, SYM, enables or disables global interrupt processing (SYM settings also enable fast interrupts and control the activity of external interface, if implemented).

Table 5-2. Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable or disable interrupt processing for each of the eight interrupt levels: IRQ0–IRQ7.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The seven levels of S3C8275X/C8278X/C8274X are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, group B is IRQ2, IRQ3 and IRQ4, and group C is IRQ5, IRQ6, and IRQ7.
Interrupt request register	IRQ	R	This register contains a request pending bit for each interrupt level.
System mode register	SYM	R/W	This register enables/disables fast interrupt processing and dynamic global interrupt processing.

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

INTERRUPT PROCESSING CONTROL POINTS

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

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

— Interrupt level enable/disable settings (IMR register)

— Interrupt level priority settings (IPR register)

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

NOTE

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

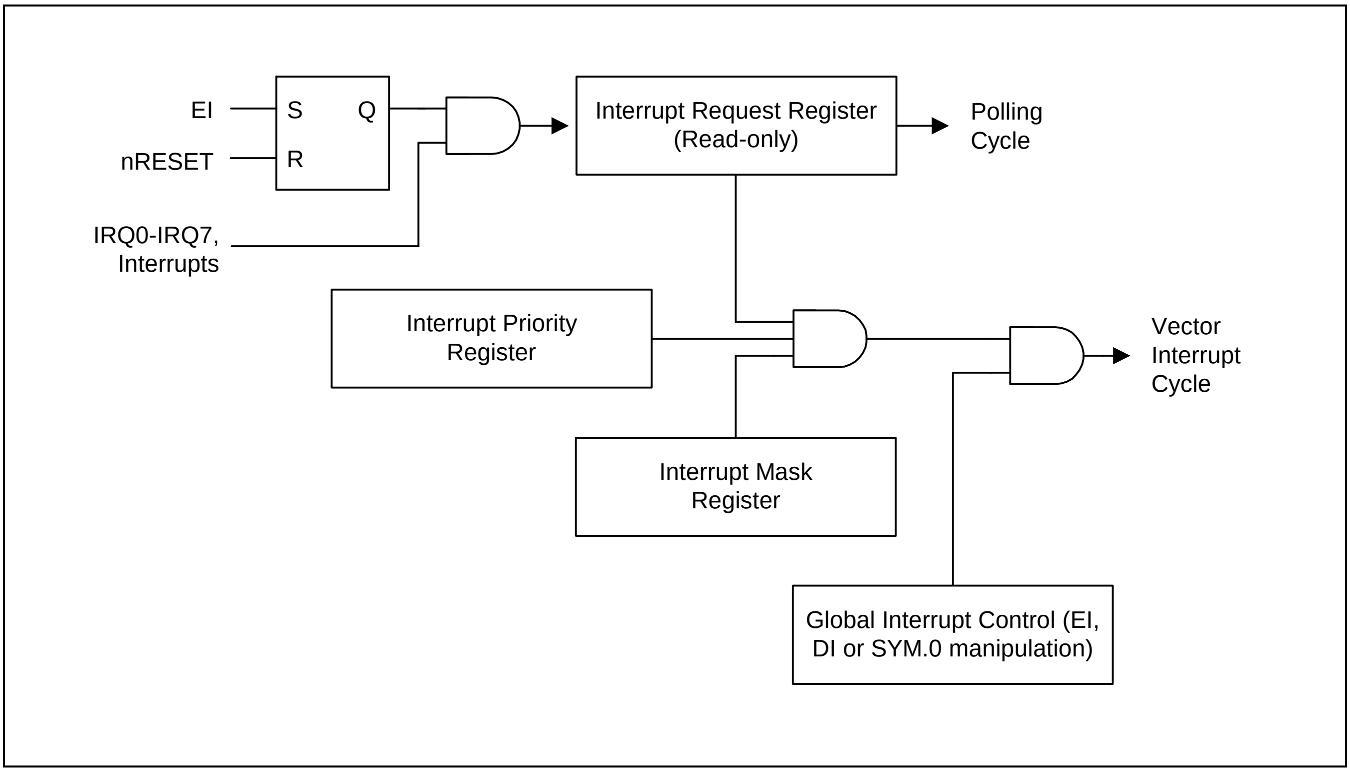


Figure 5-4. Interrupt Function Diagram

PERIPHERAL INTERRUPT CONTROL REGISTERS

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

Table 5-3. Interrupt Source Control and Data Registers

Interrupt Source	Interrupt Level	Register(s)	Location(s) in Set 1
Timer B match Timer 1/A match	IRQ0	TBCON, TBDATA, TBCNT TACON, TADATA, TACNT	E7H, E5H, E3H, bank 1 E6H, E4H, E2H, bank 1
SIO interrupt	IRQ1	SIOCON SIODATA SIOPS	E1H, bank 0 E2H, bank 0 E3H, bank 0
Watch timer overflow	IRQ2	WTCON	E1H, bank 1
P0.0 external interrupt	IRQ3	P0CONL EXTICONL EXTIPND	E5H, bank 0 F9H, bank 0 F7H, bank 0
P0.1 external interrupt	IRQ4	P0CONL EXTICONL EXTIPND	E5H, bank 0 F9H, bank 0 F7H, bank 0
P0.2 external interrupt	IRQ5	P0CONL EXTICONL EXTIPND	E5H, bank 0 F9H, bank 0 F7H, bank 0
P1.3 external interrupt	IRQ6	P1CONL EXTICONL EXTIPND	E8H, bank 0 F9H, bank 0 F7H, bank 0
P1.7 external interrupt P1.6 external interrupt P1.5 external interrupt P1.4 external interrupt	IRQ7	P1CONH EXTICONH EXTIPND	E7H, bank 0 F8H, bank 0 F7H, bank 0

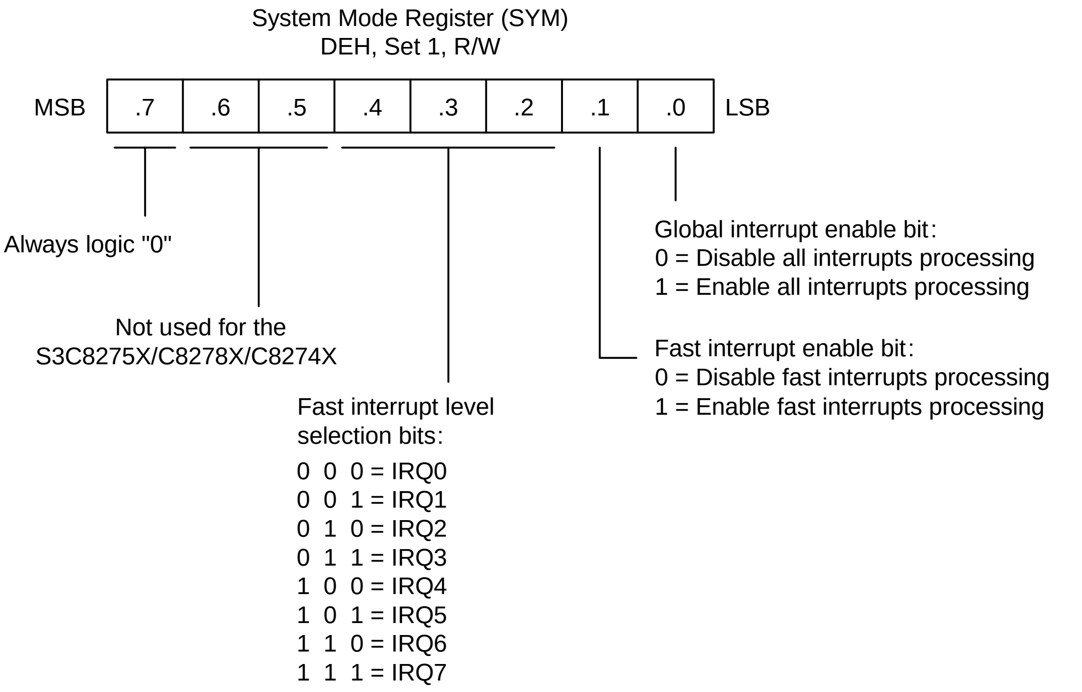
NOTE: If an interrupt is un-mask (Enable interrupt level) in the IMR register, the pending bit and enable bit of the interrupt should be written after a DI instruction is executed.

SYSTEM MODE REGISTER (SYM)

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

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

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



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

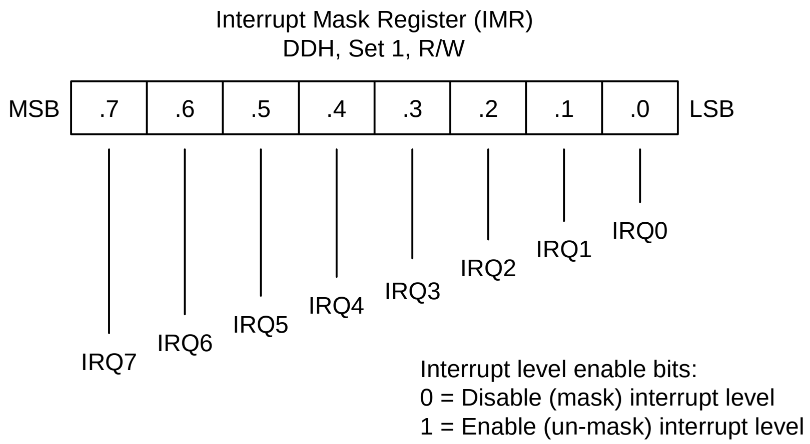
Figure 5-5. System Mode Register (SYM)

INTERRUPT MASK REGISTER (IMR)

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

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

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



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

Figure 5-6. Interrupt Mask Register (IMR)

INTERRUPT PRIORITY REGISTER (IPR)

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

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

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

Group A	IRQ0, IRQ1
Group B	IRQ2, IRQ3, IRQ4
Group C	IRQ5, IRQ6, IRQ7

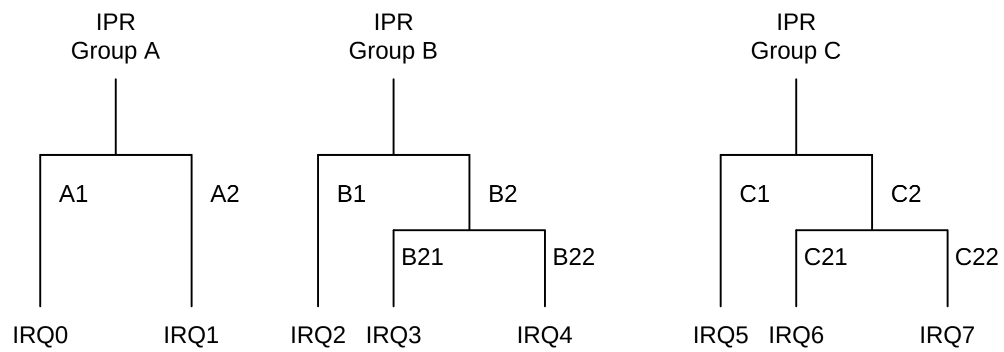


Figure 5-7. Interrupt Request Priority Groups

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

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

- IPR.5 controls the relative priorities of group C interrupts.
- Interrupt group C includes a subgroup that has an additional priority relationship among the interrupt levels 5, 6, and 7. IPR.6 defines the subgroup C relationship. IPR.5 controls the interrupt group C.
- IPR.0 controls the relative priority setting of IRQ0 and IRQ1 interrupts.

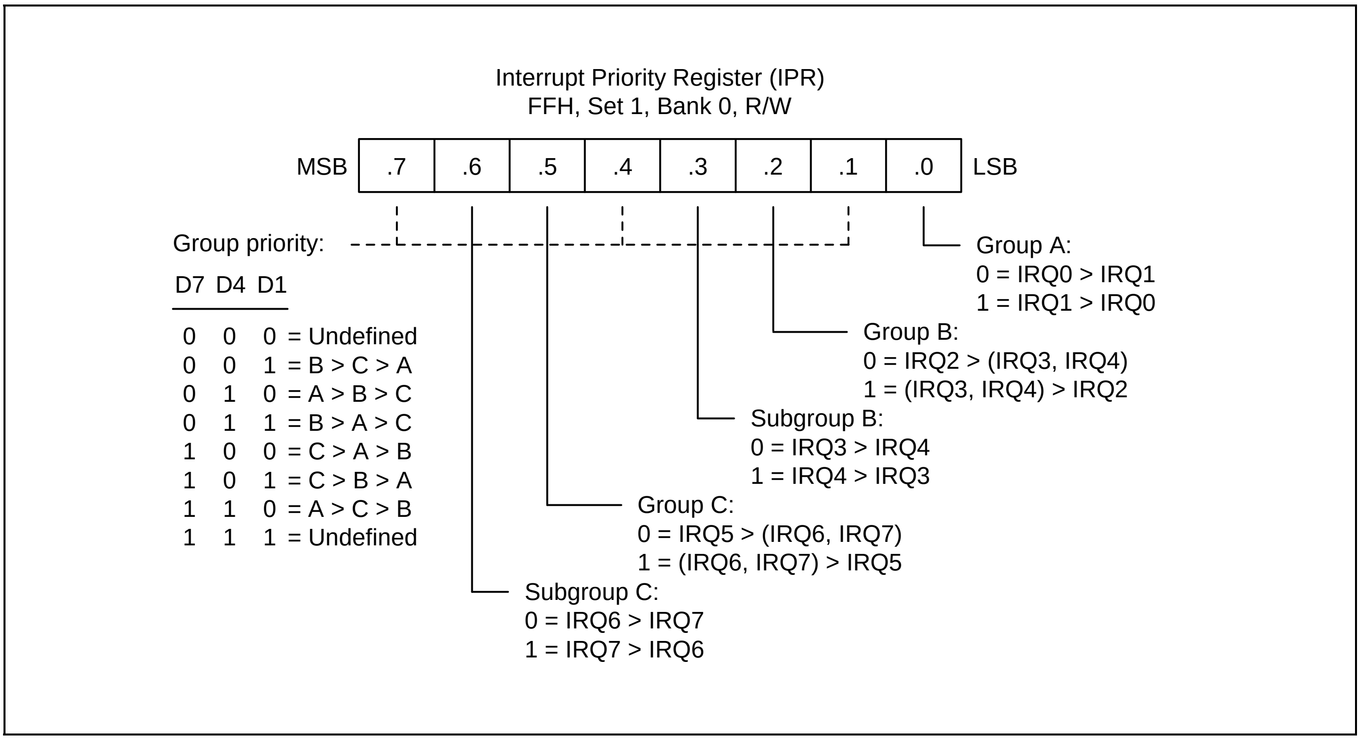


Figure 5-8. Interrupt Priority Register (IPR)

INTERRUPT REQUEST REGISTER (IRQ)

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

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

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

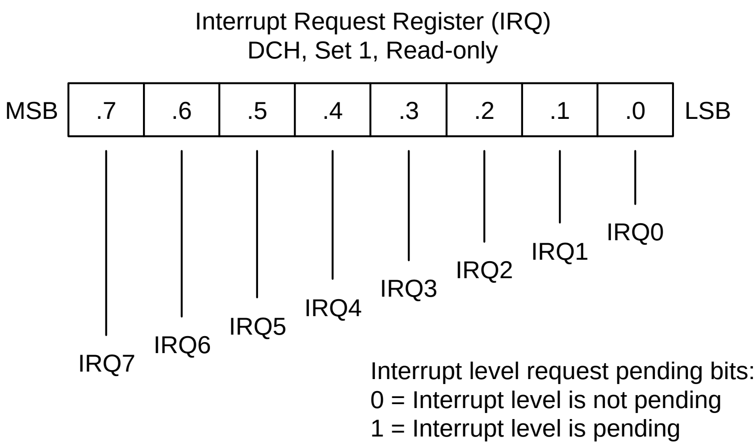


Figure 5-9. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

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

Pending Bits Cleared Automatically by Hardware

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

Pending Bits Cleared by the Service Routine

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

☞ Programming Tip — How to clear an interrupt pending bit

As the following examples are shown, a load instruction should be used to clear an interrupt pending bit.

Examples:

1.
```
        SB0
        LD      EXTIPND, #11111011B     ; Clear P0.2's interrupt pending bit
        •
        •
        •
        IRET
```

INTERRUPT SOURCE POLLING SEQUENCE

The interrupt request polling and servicing sequence is as follows:

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

INTERRUPT SERVICE ROUTINES

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

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

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

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

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

GENERATING INTERRUPT VECTOR ADDRESSES

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

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

NOTE

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

NESTING OF VECTORED INTERRUPTS

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

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

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

INSTRUCTION POINTER (IP)

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

FAST INTERRUPT PROCESSING

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

FAST INTERRUPT PROCESSING (Continued)

Two other system registers support fast interrupt processing:

- The instruction pointer (IP) contains the starting address of the service routine (and is later used to swap the program counter values), and
- When a fast interrupt occurs, the contents of the FLAGS register is stored in an unmapped, dedicated register called FLAGS' (“FLAGS prime”).

NOTE

For the S3C8275X/C8278X/C8274X microcontroller, the service routine for any one of the eight interrupt levels: IRQ0–IRQ7, can be selected for fast interrupt processing.

Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

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

Fast Interrupt Service Routine

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

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

Relationship to Interrupt Pending Bit Types

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

Programming Guidelines

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

6 INSTRUCTION SET

OVERVIEW

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

- A full complement of 8-bit arithmetic and logic operations, including multiply and divide
- No special I/O instructions (I/O control/data registers are mapped directly into the register file)
- Decimal adjustment included in binary-coded decimal (BCD) operations
- 16-bit (word) data can be incremented and decremented
- Flexible instructions for bit addressing, rotate, and shift operations

DATA TYPES

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

REGISTER ADDRESSING

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

ADDRESSING MODES

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

Table 6-1. Instruction Group Summary

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

Table 6-1. Instruction Group Summary (Continued)

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

Table 6-1. Instruction Group Summary (Continued)

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

Table 6-1. Instruction Group Summary (Concluded)

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

FLAGS REGISTER (FLAGS)

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

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

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

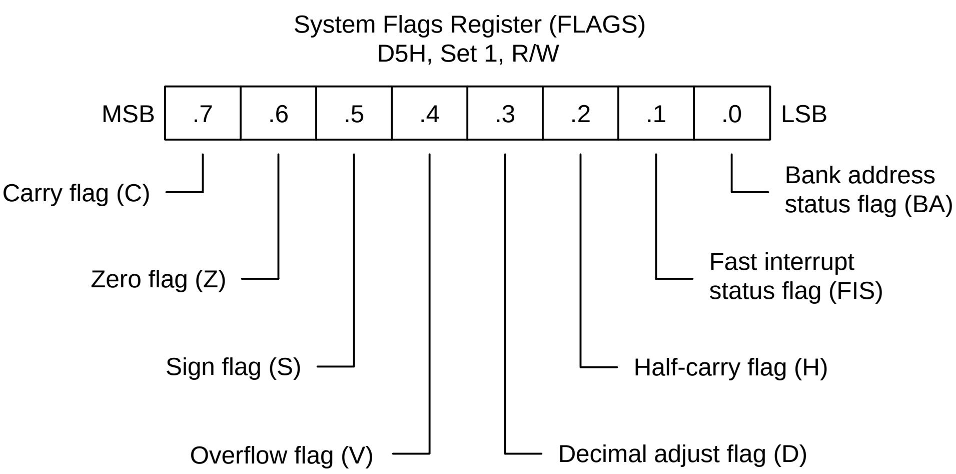


Figure 6-1. System Flags Register (FLAGS)

FLAG DESCRIPTIONS

C Carry Flag (FLAGS.7)

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

Z Zero Flag (FLAGS.6)

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

S Sign Flag (FLAGS.5)

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

V Overflow Flag (FLAGS.4)

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

D Decimal Adjust Flag (FLAGS.3)

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

H Half-Carry Flag (FLAGS.2)

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

FIS Fast Interrupt Status Flag (FLAGS.1)

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

BA Bank Address Flag (FLAGS.0)

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

INSTRUCTION SET NOTATION

Table 6-2. Flag Notation Conventions

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

Table 6-3. Instruction Set Symbols

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

SAMSUNG ELECTRONICS

Table 6-4. Instruction Notation Conventions

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

Table 6-5. Opcode Quick Reference

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

Table 6-5. Opcode Quick Reference (Continued)

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

CONDITION CODES

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

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

Table 6-6. Condition Codes

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

NOTES:

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

INSTRUCTION DESCRIPTIONS

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

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

ADC — Add with carry

ADC dst,src

Operation: dst ← dst + src + c

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

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

Format:

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.
D: Always cleared to "0".
H: Set if a carry from the low-order nibble occurred.

Format:

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

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

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

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

AND — Logical AND

AND dst,src

Operation: dst ← dst AND src

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

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

Format:

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

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

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

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

BAND — Bit AND

BAND dst,src.b

BAND dst.b,src

Operation: dst(0) ← dst(0) AND src(b)

or

dst(b) ← dst(b) AND src(0)

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

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

Format:

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

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

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

BAND R1,01H.1 → R1 = 06H, register 01H = 05H
BAND 01H.1,R1 → Register 01H = 05H, R1 = 07H

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

BCP — Bit Compare

BCP dst,src.b

Operation: dst(0) – src(b)

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

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

Format:

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

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

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

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

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

BITC — Bit Complement

BITC dst.b

Operation: dst(b) ← NOT dst(b)

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

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

Format:

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

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

Example: Given: R1 = 07H

BITC R1.1 → R1 = 05H

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

BITR — Bit Reset

BITR dst.b

Operation: dst(b) $\leftarrow$ 0

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

Flags: No flags are affected.

Format:

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

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

Example: Given: R1 = 07H:

BITR R1.1 $\rightarrow$ R1 = 05H

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

BITS — Bit Set

BITS dst.b

Operation: dst(b) ← 1

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

Flags: No flags are affected.

Format:

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

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

Example: Given: R1 = 07H:

BITS R1.3 → R1 = 0FH

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

BOR — Bit OR

BOR dst,src.b

BOR dst.b,src

Operation: dst(0) ← dst(0) OR src(b)

or

dst(b) ← dst(b) OR src(0)

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

Flags:

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

Format:

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

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

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

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

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

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

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

BTJRF — Bit Test, Jump Relative on False

BTJRF dst,src.b

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

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

Flags: No flags are affected.

Format:

(Note 1)

opc	src | b | 0	dst

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

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

Example: Given: R1 = 07H:

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

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

BTJRT — Bit Test, Jump Relative on True

BTJRT dst,src.b

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

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

Flags: No flags are affected.

Format:

(Note 1)			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	src | b | 1	dst	3	10	37	RA	rb

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

Example: Given: R1 = 07H:

BTJRT SKIP,R1.1

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

BXOR — Bit XOR

BXOR dst,src.b

BXOR dst.b,src

Operation: dst(0) ← dst(0) XOR src(b)

or

dst(b) ← dst(b) XOR src(0)

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

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

Format:

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

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

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

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

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

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

CALL — Call Procedure

CALL dst

Operation:

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

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

Flags: No flags are affected.

Format:

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

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

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

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

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

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

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

CCF — Complement Carry Flag

CCF

Operation: C ← NOT C

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

Flags: **C:** Complemented.

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	EF

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

CCF

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

CLR — Clear

CLR dst

Operation: dst ← "0"

The destination location is cleared to "0".

Flags: No flags are affected.

Format:

opc	dst

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

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

CLR 00H → Register 00H = 00H

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

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

COM — Complement

COM dst

Operation: dst ← NOT dst

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

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

Format:

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

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

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

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

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

CP — Compare

CP dst,src

Operation: dst – src

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

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

Format:

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

Examples:

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

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

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

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

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

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

CPIJE — Compare, Increment, and Jump on Equal

CPIJE dst,src,RA

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

Ir ← Ir + 1

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

Flags: No flags are affected.

Format:

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

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

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

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

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

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

CPIJNE dst,src,RA

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

Ir ← Ir + 1

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

Flags: No flags are affected.

Format:

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

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

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

CPIJNER1,@R2,SKIP → R2 = 04H, PC jumps to SKIP location

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

SAMSUNG ELECTRONICS

DA — Decimal Adjust

DA dst

Operation: dst ← DA dst

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

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

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

Format:

opc	dst

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

DA — Decimal Adjust

DA (Continued)

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

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

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

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

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

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

Assuming the same values given above, the statements

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

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

DEC — Decrement

DEC dst

Operation: dst ← dst – 1

The contents of the destination operand are decremented by one.

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

Format:

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

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

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

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

DECW — Decrement Word

DECW dst

Operation: dst ← dst – 1

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

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

Format:

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

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

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

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

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

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

DI — Disable Interrupts

DI

Operation: SYM (0) $\leftarrow$ 0

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

Flags: No flags are affected.

Format:

	Bytes	**Cycles**	**Opcode (Hex)**
opc	1	4	8F

Example: Given: SYM = 01H:

DI

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

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

DIV — Divide (Unsigned)

DIV dst,src

Operation: dst ÷ src

dst (UPPER) ← REMAINDER

dst (LOWER) ← QUOTIENT

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

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

Format:

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

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

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

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

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

DJNZ — Decrement and Jump if Non-Zero

DJNZ r,dst

Operation: r ← r – 1

If r ≠ 0, PC ← PC + dst

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

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

Flags: No flags are affected.

Format:

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

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

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

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

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

EI — Enable Interrupts

EI

Operation: SYM (0) ← 1

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	9F

Example: Given: SYM = 00H:

EI

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

ENTER — Enter

ENTER

Operation:

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

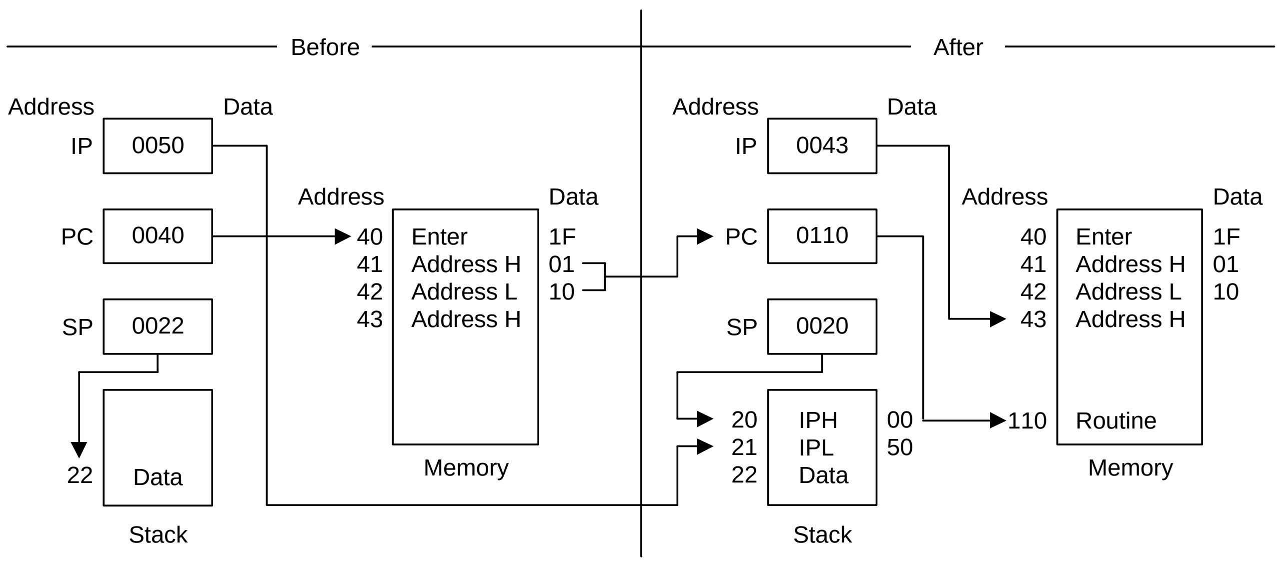
This instruction is useful when implementing threaded-code languages. The contents of the instruction pointer are pushed to the stack. The program counter (PC) value is then written to the instruction pointer. The program memory word that is pointed to by the instruction pointer is loaded into the PC,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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	14	1F

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



EXIT — Exit

EXIT

Operation:

IP ← @SP

SP ← SP + 2

PC ← @IP

IP ← IP + 2

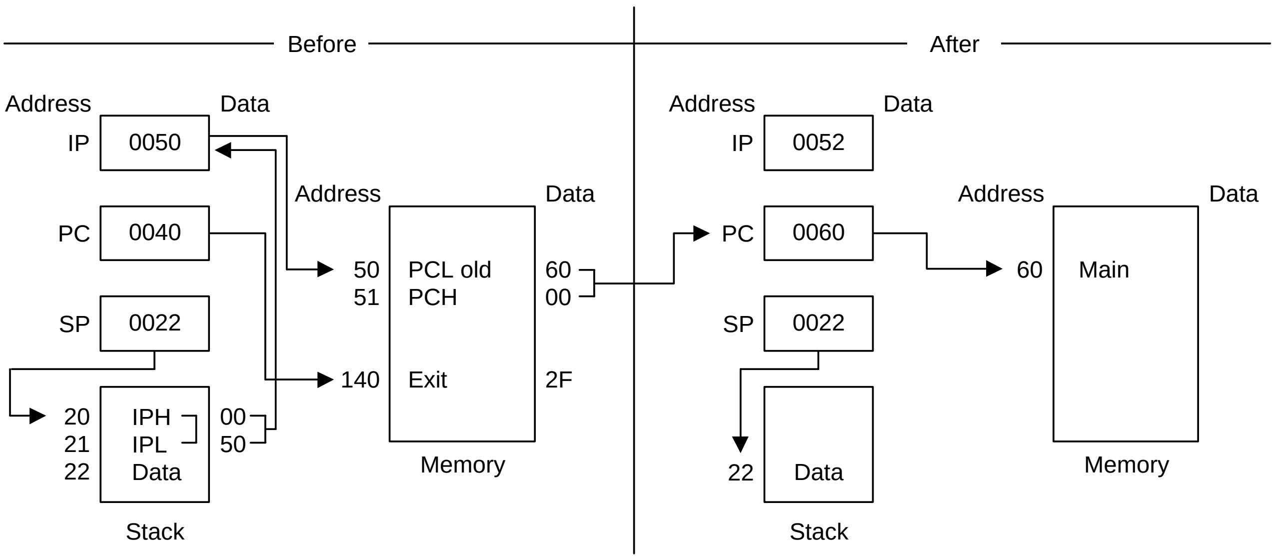
This instruction is useful when implementing threaded-code languages. The stack value is popped and loaded into the instruction pointer. The program memory word that is pointed to by the instruction pointer is then loaded into the program counter, and the instruction pointer is incremented by two.

Flags: No flags are affected.

Format:

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

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



IDLE — Idle Operation

IDLE

Operation:

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

Flags: No flags are affected.

Format:

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

Example: The instruction

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

INC — Increment

INC dst

Operation: dst ← dst + 1

The contents of the destination operand are incremented by one.

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

Format:

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

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

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

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

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

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

INCW — Increment Word

INCW dst

Operation: dst ← dst + 1

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

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

Format:

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

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

INCW RR0 → R0 = 1AH, R1 = 03H

INCW @R1 → Register 02H = 10H, register 03H = 00H

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

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

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

IRET — Interrupt Return

IRET IRET (Normal) IRET (Fast)

Operation:

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

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

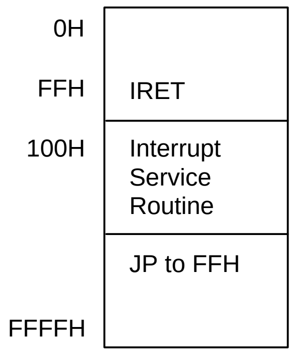
Flags: All flags are restored to their original settings (that is, the settings before the interrupt occurred).

Format:

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

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

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



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

JP — Jump

JP cc,dst (Conditional)

JP dst (Unconditional)

Operation: If cc is true, PC ← dst

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

Flags: No flags are affected.

Format: (1)

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

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

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

```
JP    C,LABEL_W    →    LABEL_W = 1000H, PC = 1000H
JP    @00H         →    PC = 0120H
```

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

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

JR — Jump Relative

JR cc,dst

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

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

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

Flags: No flags are affected.

Format:

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

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

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

JR C,LABEL_X → PC = 1FF7H

If the carry flag is set (that is, if the condition code is true), the statement "JR C,LABEL_X" will pass control to the statement whose address is now in the PC. Otherwise, the program instruction following the JR would be executed.

LD — Load

LD dst,src

Operation: dst ← src

The contents of the source are loaded into the destination. The source's contents are unaffected.

Flags: No flags are affected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
dst | opc, src	2	4	rC	r	IM
		4	r8	r	R
src | opc, dst	2	4	r9	R	r
			r = 0 to F		
opc, dst | src	2	4	C7	r	Ir
		4	D7	Ir	r
opc, src, dst	3	6	E4	R	R
		6	E5	R	IR
opc, dst, src	3	6	E6	R	IM
		6	D6	IR	IM
opc, src, dst	3	6	F5	IR	R
opc, dst | src, x	3	6	87	r	x [r]
opc, src | dst, x	3	6	97	x [r]	r

LD — Load

LD (Continued)

Examples: Given: R0 = 01H, R1 = 0AH, register 00H = 01H, register 01H = 20H, register 02H = 02H, LOOP = 30H, and register 3AH = 0FFH:

LD	R0,#10H	→	R0 = 10H
LD	R0,01H	→	R0 = 20H, register 01H = 20H
LD	01H,R0	→	Register 01H = 01H, R0 = 01H
LD	R1,@R0	→	R1 = 20H, R0 = 01H
LD	@R0,R1	→	R0 = 01H, R1 = 0AH, register 01H = 0AH
LD	00H,01H	→	Register 00H = 20H, register 01H = 20H
LD	02H,@00H	→	Register 02H = 20H, register 00H = 01H
LD	00H,#0AH	→	Register 00H = 0AH
LD	@00H,#10H	→	Register 00H = 01H, register 01H = 10H
LD	@00H,02H	→	Register 00H = 01H, register 01H = 02, register 02H = 02H
LD	R0,#LOOP[R1]	→	R0 = 0FFH, R1 = 0AH
LD	#LOOP[R0],R1	→	Register 31H = 0AH, R0 = 01H, R1 = 0AH

LDB — Load Bit

LDB dst,src.b

LDB dst.b,src

Operation: dst(0) ← src(b)

or

dst(b) ← src(0)

The specified bit of the source is loaded into bit zero (LSB) of the destination, or bit zero of the source is loaded into the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>	Addr Mode <u>src</u>
opc | dst | b | 0 | src	3	6	47	r0	Rb
opc | src | b | 1 | dst	3	6	47	Rb	r0

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

Examples: Given: R0 = 06H and general register 00H = 05H:

LDB	R0,00H.2	→	R0 = 07H, register 00H = 05H
LDB	00H.0,R0	→	R0 = 06H, register 00H = 04H

In the first example, destination working register R0 contains the value 06H and the source general register 00H the value 05H. The statement "LD R0,00H.2" loads the bit two value of the 00H register into bit zero of the R0 register, leaving the value 07H in register R0.

In the second example, 00H is the destination register. The statement "LD 00H.0,R0" loads bit zero of register R0 to the specified bit (bit zero) of the destination register, leaving 04H in general register 00H.

LDC/LDE — Load Memory

LDC/LDE dst,src

Operation: dst ← src

This instruction loads a byte from program or data memory into a working register or vice-versa. The source values are unaffected. LDC refers to program memory and LDE to data memory. The assembler makes 'Irr' or 'rr' values an even number for program memory and odd an odd number for data memory.

Flags: No flags are affected.

Format:

	Format				Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
1.	opc	dst | src			2	10	C3	r	Irr
2.	opc	src | dst			2	10	D3	Irr	r
3.	opc	dst | src	XS		3	12	E7	r	XS [rr]
4.	opc	src | dst	XS		3	12	F7	XS [rr]	r
5.	opc	dst | src	XL_L	XL_H	4	14	A7	r	XL [rr]
6.	opc	src | dst	XL_L	XL_H	4	14	B7	XL [rr]	r
7.	opc	dst | 0000	DA_L	DA_H	4	14	A7	r	DA
8.	opc	src | 0000	DA_L	DA_H	4	14	B7	DA	r
9.	opc	dst | 0001	DA_L	DA_H	4	14	A7	r	DA
10.	opc	src | 0001	DA_L	DA_H	4	14	B7	DA	r

NOTES:

1. The source (src) or working register pair [rr] for formats 5 and 6 cannot use register pair 0–1.
2. For formats 3 and 4, the destination address 'XS [rr]' and the source address 'XS [rr]' are each one byte.
3. For formats 5 and 6, the destination address 'XL [rr] and the source address 'XL [rr]' are each two bytes.
4. The DA and r source values for formats 7 and 8 are used to address program memory; the second set of values, used in formats 9 and 10, are used to address data memory.

LDC/LDE — Load Memory

LDC/LDE (Continued)

Examples: Given: R0 = 11H, R1 = 34H, R2 = 01H, R3 = 04H; Program memory locations 0103H = 4FH, 0104H = 1A, 0105H = 6DH, and 1104H = 88H. External data memory locations 0103H = 5FH, 0104H = 2AH, 0105H = 7DH, and 1104H = 98H:

```
LDC        R0,@RR2          ; R0 ← contents of program memory location 0104H
                            ; R0 = 1AH, R2 = 01H, R3 = 04H

LDE        R0,@RR2          ; R0 ← contents of external data memory location 0104H
                            ; R0 = 2AH, R2 = 01H, R3 = 04H

LDC (note) @RR2,R0          ; 11H (contents of R0) is loaded into program memory
                            ; location 0104H (RR2),
                            ; working registers R0, R2, R3 → no change

LDE        @RR2,R0          ; 11H (contents of R0) is loaded into external data memory
                            ; location 0104H (RR2),
                            ; working registers R0, R2, R3 → no change

LDC        R0,#01H[RR2]     ; R0 ← contents of program memory location 0105H
                            ; (01H + RR2),
                            ; R0 = 6DH, R2 = 01H, R3 = 04H

LDE        R0,#01H[RR2]     ; R0 ← contents of external data memory location 0105H
                            ; (01H + RR2), R0 = 7DH, R2 = 01H, R3 = 04H

LDC (note) #01H[RR2],R0     ; 11H (contents of R0) is loaded into program memory location
                            ; 0105H (01H + 0104H)

LDE        #01H[RR2],R0     ; 11H (contents of R0) is loaded into external data memory
                            ; location 0105H (01H + 0104H)

LDC        R0,#1000H[RR2]   ; R0 ← contents of program memory location 1104H
                            ; (1000H + 0104H), R0 = 88H, R2 = 01H, R3 = 04H

LDE        R0,#1000H[RR2]   ; R0 ← contents of external data memory location 1104H
                            ; (1000H + 0104H), R0 = 98H, R2 = 01H, R3 = 04H

LDC        R0,1104H         ; R0 ← contents of program memory location 1104H, R0=88H

LDE        R0,1104H         ; R0 ← contents of external data memory location 1104H,
                            ; R0 = 98H

LDC (note) 1105H,R0         ; 11H (contents of R0) is loaded into program memory location
                            ; 1105H, (1105H) ← 11H

LDE        1105H,R0         ; 11H (contents of R0) is loaded into external data memory
                            ; location 1105H, (1105H) ← 11H
```

NOTE: These instructions are not supported by masked ROM type devices.

LDCD/LDED — Load Memory and Decrement

LDCD/LDED dst,src

Operation: dst ← src

rr ← rr – 1

These instructions are used for user stacks or block transfers of data from program or data memory to the register file. The address of the memory location is specified by a working register pair. The contents of the source location are loaded into the destination location. The memory address is then decremented. The contents of the source are unaffected.

LDCD references program memory and LDED references external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for data memory.

Flags: No flags are affected.

Format:

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

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	10	E3	r	Irr

Examples: Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory locations 1033H = 0CDH and 1034H = 0C5H; external data memory locations 1033H = 0DDH and 1034H = 0D5H:

```
LDCI   R8,@RR6   ; 0CDH (contents of program memory location 1033H) is loaded
                 ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                 ; R8 = 0CDH, R6 = 10H, R7 = 34H

LDEI   R8,@RR6   ; 0DDH (contents of data memory location 1033H) is loaded
                 ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                 ; R8 = 0DDH, R6 = 10H, R7 = 34H
```

LDCPD/LDEPD — Load Memory with Pre-Decrement

LDCPD/
LDEPD dst,src

Operation: rr ← rr – 1

dst ← src

These instructions are used for block transfers of data from program or data memory from the register file. The address of the memory location is specified by a working register pair and is first decremented. The contents of the source location are then loaded into the destination location. The contents of the source are unaffected.

LDCPD refers to program memory and LDEPD refers to external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for external data memory.

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	src | dst	2	14	F2	Irr	r

Examples: Given: R0 = 77H, R6 = 30H, and R7 = 00H:

```
LDCPD   @RR6,R0      ; (RR6 ← RR6 – 1)
                     ; 77H (contents of R0) is loaded into program memory location
                     ; 2FFFH (3000H – 1H)
                     ; R0 = 77H, R6 = 2FH, R7 = 0FFH

LDEPD   @RR6,R0      ; (RR6 ← RR6 – 1)
                     ; 77H (contents of R0) is loaded into external data memory
                     ; location 2FFFH (3000H – 1H)
                     ; R0 = 77H, R6 = 2FH, R7 = 0FFH
```

LDCPI/LDEPI — Load Memory with Pre-Increment

LDCPI/
LDEPI dst,src

Operation: rr ← rr + 1

dst ← src

These instructions are used for block transfers of data from program or data memory from the register file. The address of the memory location is specified by a working register pair and is first incremented. The contents of the source location are loaded into the destination location. The contents of the source are unaffected.

LDCPI refers to program memory and LDEPI refers to external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for data memory.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | src | dst	2	14	F3	Irr	r

Examples: Given: R0 = 7FH, R6 = 21H, and R7 = 0FFH:

```
LDCPI   @RR6,R0     ; (RR6 ← RR6 + 1)
                    ; 7FH (contents of R0) is loaded into program memory
                    ; location 2200H (21FFH + 1H)
                    ; R0 = 7FH, R6 = 22H, R7 = 00H

LDEPI   @RR6,R0     ; (RR6 ← RR6 + 1)
                    ; 7FH (contents of R0) is loaded into external data memory
                    ; location 2200H (21FFH + 1H)
                    ; R0 = 7FH, R6 = 22H, R7 = 00H
```

LDW — Load Word

LDW dst,src

Operation: dst ← src

The contents of the source (a word) are loaded into the destination. The contents of the source are unaffected.

Flags: No flags are affected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | src | dst	3	8	C4	RR	RR
		8	C5	RR	IR
opc | dst | src	4	8	C6	RR	IML

Examples: Given: R4 = 06H, R5 = 1CH, R6 = 05H, R7 = 02H, register 00H = 1AH, register 01H = 02H, register 02H = 03H, and register 03H = 0FH:

LDW	RR6,RR4	→	R6 = 06H, R7 = 1CH, R4 = 06H, R5 = 1CH
LDW	00H,02H	→	Register 00H = 03H, register 01H = 0FH, register 02H = 03H, register 03H = 0FH
LDW	RR2,@R7	→	R2 = 03H, R3 = 0FH,
LDW	04H,@01H	→	Register 04H = 03H, register 05H = 0FH
LDW	RR6,#1234H	→	R6 = 12H, R7 = 34H
LDW	02H,#0FEDH	→	Register 02H = 0FH, register 03H = 0EDH

In the second example, please note that the statement "LDW 00H,02H" loads the contents of the source word 02H, 03H into the destination word 00H, 01H. This leaves the value 03H in general register 00H and the value 0FH in register 01H.

The other examples show how to use the LDW instruction with various addressing modes and formats.

SAMSUNG ELECTRONICS

MULT — Multiply (Unsigned)

MULT dst,src

Operation: dst ← dst × src

The 8-bit destination operand (even register of the register pair) is multiplied by the source operand (8 bits) and the product (16 bits) is stored in the register pair specified by the destination address. Both operands are treated as unsigned integers.

Flags:
C: Set if result is > 255; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if MSB of the result is a "1"; cleared otherwise.
V: Cleared.
D: Unaffected.
H: Unaffected.

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | src | dst	3	22	84	RR	R
		22	85	RR	IR
		22	86	RR	IM

Examples: Given: Register 00H = 20H, register 01H = 03H, register 02H = 09H, register 03H = 06H:

MULT 00H, 02H → Register 00H = 01H, register 01H = 20H, register 02H = 09H
MULT 00H, @01H → Register 00H = 00H, register 01H = 0C0H
MULT 00H, #30H → Register 00H = 06H, register 01H = 00H

In the first example, the statement "MULT 00H,02H" multiplies the 8-bit destination operand (in the register 00H of the register pair 00H, 01H) by the source register 02H operand (09H). The 16-bit product, 0120H, is stored in the register pair 00H, 01H.

NEXT — Next

NEXT

Operation: PC ← @ IP

IP ← IP + 2

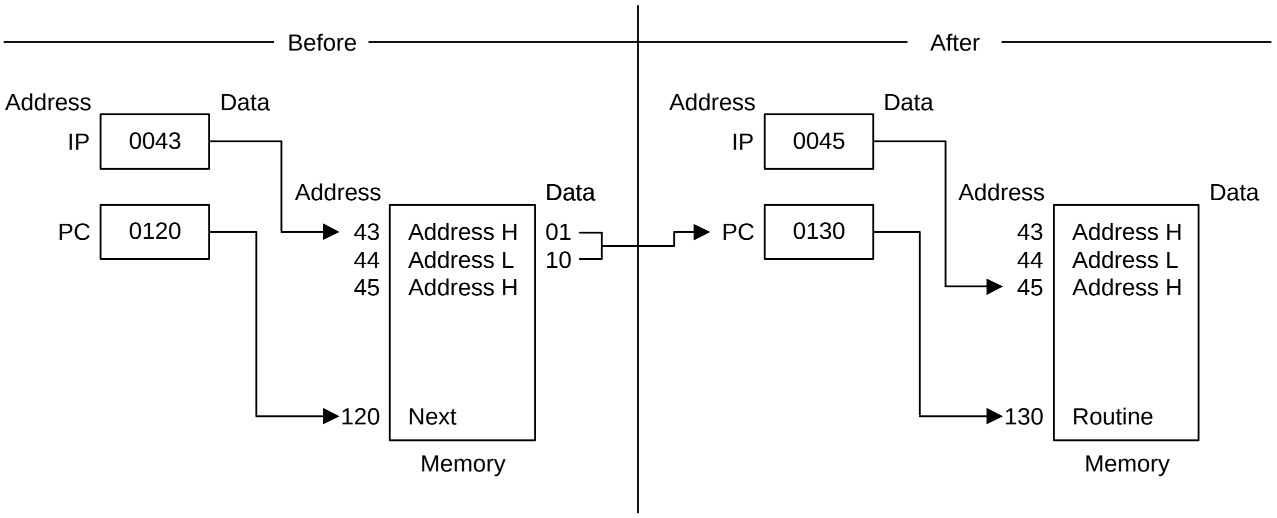
The NEXT instruction is useful when implementing threaded-code languages. The program memory word that is pointed to by the instruction pointer is loaded into the program counter. The instruction pointer is then incremented by two.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	10	0F

Example: The following diagram shows one example of how to use the NEXT instruction.



SAMSUNG ELECTRONICS

NOP — No Operation

NOP

Operation: No action is performed when the CPU executes this instruction. Typically, one or more NOPs are executed in sequence in order to effect a timing delay of variable duration.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	FF

Example: When the instruction

NOP

is encountered in a program, no operation occurs. Instead, there is a delay in instruction execution time.

OR — Logical OR

OR dst,src

Operation: dst ← dst OR src

The source operand is logically ORed with the destination operand and the result is stored in the destination. The contents of the source are unaffected. The OR operation results in a "1" being stored whenever either of the corresponding bits in the two operands is a "1"; otherwise a "0" is stored.

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

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	42	r	r
		6	43	r	Ir
opc | src | dst	3	6	44	R	R
		6	45	R	IR
opc | dst | src	3	6	46	R	IM

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

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

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

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

POP — Pop From Stack

POP dst

Operation: dst ← @SP

SP ← SP + 1

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

Flags: No flags affected.

Format:

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

Examples: Given: Register 00H = 01H, register 01H = 1BH, SPH (0D8H) = 00H, SPL (0D9H) = 0FBH, and stack register 0FBH = 55H:

POP 00H → Register 00H = 55H, SP = 00FCH

POP @00H → Register 00H = 01H, register 01H = 55H, SP = 00FCH

In the first example, general register 00H contains the value 01H. The statement "POP 00H" loads the contents of location 00FBH (55H) into destination register 00H and then increments the stack pointer by one. Register 00H then contains the value 55H and the SP points to location 00FCH.

POPUD — Pop User Stack (Decrementing)

POPUD dst,src

Operation: dst ← src

IR ← IR – 1

This instruction is used for user-defined stacks in the register file. The contents of the register file location addressed by the user stack pointer are loaded into the destination. The user stack pointer is then decremented.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | src | dst	3	8	92	R	IR

Example: Given: Register 00H = 42H (user stack pointer register), register 42H = 6FH, and register 02H = 70H:

POPUD 02H,@00H → Register 00H = 41H, register 02H = 6FH, register 42H = 6FH

If general register 00H contains the value 42H and register 42H the value 6FH, the statement "POPUD 02H,@00H" loads the contents of register 42H into the destination register 02H. The user stack pointer is then decremented by one, leaving the value 41H.

POPUI — Pop User Stack (Incrementing)

POPUI dst,src

Operation: dst ← src

IR ← IR + 1

The POPUI instruction is used for user-defined stacks in the register file. The contents of the register file location addressed by the user stack pointer are loaded into the destination. The user stack pointer is then incremented.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | src | dst	3	8	93	R	IR

Example: Given: Register 00H = 01H and register 01H = 70H:

POPUI 02H,@00H → Register 00H = 02H, register 01H = 70H, register 02H = 70H

If general register 00H contains the value 01H and register 01H the value 70H, the statement "POPUI 02H,@00H" loads the value 70H into the destination general register 02H. The user stack pointer (register 00H) is then incremented by one, changing its value from 01H to 02H.

PUSH — Push To Stack

PUSH src

Operation: SP ← SP – 1

@SP ← src

A PUSH instruction decrements the stack pointer value and loads the contents of the source (src) into the location addressed by the decremented stack pointer. The operation then adds the new value to the top of the stack.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | src	2	8 (internal clock)	70	R
		8 (external clock)		
		8 (internal clock)		
		8 (external clock)	71	IR

Examples: Given: Register 40H = 4FH, register 4FH = 0AAH, SPH = 00H, and SPL = 00H:

PUSH 40H → Register 40H = 4FH, stack register 0FFH = 4FH, SPH = 0FFH, SPL = 0FFH

PUSH @40H → Register 40H = 4FH, register 4FH = 0AAH, stack register 0FFH = 0AAH, SPH = 0FFH, SPL = 0FFH

In the first example, if the stack pointer contains the value 0000H, and general register 40H the value 4FH, the statement "PUSH 40H" decrements the stack pointer from 0000 to 0FFFFH. It then loads the contents of register 40H into location 0FFFFH and adds this new value to the top of the stack.

PUSHUD — Push User Stack (Decrementing)

PUSHUD dst,src

Operation: IR ← IR – 1

dst ← src

This instruction is used to address user-defined stacks in the register file. PUSHUD decrements the user stack pointer and loads the contents of the source into the register addressed by the decremented stack pointer.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	3	8	82	IR	R

Example: Given: Register 00H = 03H, register 01H = 05H, and register 02H = 1AH:

PUSHUD @00H,01H → Register 00H = 02H, register 01H = 05H, register 02H = 05H

If the user stack pointer (register 00H, for example) contains the value 03H, the statement "PUSHUD @00H,01H" decrements the user stack pointer by one, leaving the value 02H. The 01H register value, 05H, is then loaded into the register addressed by the decremented user stack pointer.

PUSHUI — Push User Stack (Incrementing)

PUSHUI dst,src

Operation: IR ← IR + 1

dst ← src

This instruction is used for user-defined stacks in the register file. PUSHUI increments the user stack pointer and then loads the contents of the source into the register location addressed by the incremented user stack pointer.

Flags: No flags are affected.

Format:

Format			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst	src	3	8	83	IR	R

Example: Given: Register 00H = 03H, register 01H = 05H, and register 04H = 2AH:

PUSHUI @00H,01H → Register 00H = 04H, register 01H = 05H, register 04H = 05H

If the user stack pointer (register 00H, for example) contains the value 03H, the statement "PUSHUI @00H,01H" increments the user stack pointer by one, leaving the value 04H. The 01H register value, 05H, is then loaded into the location addressed by the incremented user stack pointer.

RCF — Reset Carry Flag

RCF RCF

Operation: C ← 0

The carry flag is cleared to logic zero, regardless of its previous value.

Flags: **C:** Cleared to "0".

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	CF

Example: Given: C = "1" or "0":

The instruction RCF clears the carry flag (C) to logic zero.

RET — Return

RET

Operation: PC ← @SP

SP ← SP + 2

The RET instruction is normally used to return to the previously executing procedure at the end of a procedure entered by a CALL instruction. The contents of the location addressed by the stack pointer are popped into the program counter. The next statement that is executed is the one that is addressed by the new program counter value.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	8 (internal stack)	AF
		10 (internal stack)	

Example: Given: SP = 00FCH, (SP) = 101AH, and PC = 1234:

RET → PC = 101AH, SP = 00FEH

The statement "RET" pops the contents of stack pointer location 00FCH (10H) into the high byte of the program counter. The stack pointer then pops the value in location 00FEH (1AH) into the PC's low byte and the instruction at location 101AH is executed. The stack pointer now points to memory location 00FEH.

RL — Rotate Left

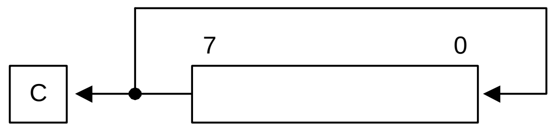
RL dst

Operation: C ← dst (7)

dst (0) ← dst (7)

dst (n + 1) ← dst (n), n = 0–6

The contents of the destination operand are rotated left one bit position. The initial value of bit 7 is moved to the bit zero (LSB) position and also replaces the carry flag.



Flags:

C: Set if the bit rotated from the most significant bit position (bit 7) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	90	R
		4	91	IR

Examples: Given: Register 00H = 0AAH, register 01H = 02H and register 02H = 17H:

RL 00H → Register 00H = 55H, C = "1"

RL @01H → Register 01H = 02H, register 02H = 2EH, C = "0"

In the first example, if general register 00H contains the value 0AAH (10101010B), the statement "RL 00H" rotates the 0AAH value left one bit position, leaving the new value 55H (01010101B) and setting the carry and overflow flags.

RLC — Rotate Left Through Carry

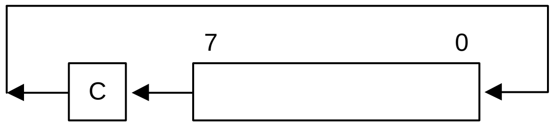
RLC dst

Operation: dst (0) ← C

C ← dst (7)

dst (n + 1) ← dst (n), n = 0–6

The contents of the destination operand with the carry flag are rotated left one bit position. The initial value of bit 7 replaces the carry flag (C); the initial value of the carry flag replaces bit zero.



Flags:
C: Set if the bit rotated from the most significant bit position (bit 7) was "1".
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	10	R
		4	11	IR

Examples: Given: Register 00H = 0AAH, register 01H = 02H, and register 02H = 17H, C = "0":

RLC 00H → Register 00H = 54H, C = "1"

RLC @01H → Register 01H = 02H, register 02H = 2EH, C = "0"

In the first example, if general register 00H has the value 0AAH (10101010B), the statement "RLC 00H" rotates 0AAH one bit position to the left. The initial value of bit 7 sets the carry flag and the initial value of the C flag replaces bit zero of register 00H, leaving the value 55H (01010101B). The MSB of register 00H resets the carry flag to "1" and sets the overflow flag.

RR — Rotate Right

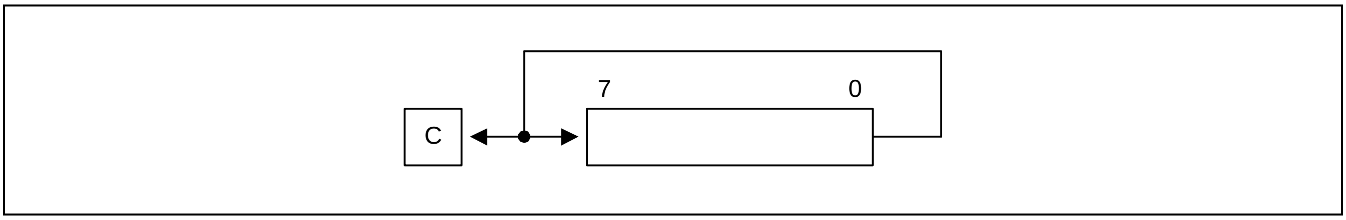
RR dst

Operation: C ← dst (0)

dst (7) ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

The contents of the destination operand are rotated right one bit position. The initial value of bit zero (LSB) is moved to bit 7 (MSB) and also replaces the carry flag (C).



Flags:

- **C:** Set if the bit rotated from the least significant bit position (bit zero) was "1".
- **Z:** Set if the result is "0"; cleared otherwise.
- **S:** Set if the result bit 7 is set; cleared otherwise.
- **V:** Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
- **D:** Unaffected.
- **H:** Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	E0	R
		4	E1	IR

Examples: Given: Register 00H = 31H, register 01H = 02H, and register 02H = 17H:

RR 00H → Register 00H = 98H, C = "1"

RR @01H → Register 01H = 02H, register 02H = 8BH, C = "1"

In the first example, if general register 00H contains the value 31H (00110001B), the statement "RR 00H" rotates this value one bit position to the right. The initial value of bit zero is moved to bit 7, leaving the new value 98H (10011000B) in the destination register. The initial bit zero also resets the C flag to "1" and the sign flag and overflow flag are also set to "1".

RRC — Rotate Right Through Carry

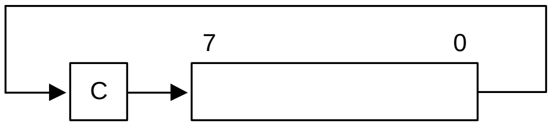
RRC dst

Operation: dst (7) ← C

C ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

The contents of the destination operand and the carry flag are rotated right one bit position. The initial value of bit zero (LSB) replaces the carry flag; the initial value of the carry flag replaces bit 7 (MSB).



Flags:

C: Set if the bit rotated from the least significant bit position (bit zero) was "1".
Z: Set if the result is "0" cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
D: Unaffected.
H: Unaffected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	C0	R
		4	C1	IR

Examples: Given: Register 00H = 55H, register 01H = 02H, register 02H = 17H, and C = "0":

RRC 00H → Register 00H = 2AH, C = "1"

RRC @01H → Register 01H = 02H, register 02H = 0BH, C = "1"

In the first example, if general register 00H contains the value 55H (01010101B), the statement "RRC 00H" rotates this value one bit position to the right. The initial value of bit zero ("1") replaces the carry flag and the initial value of the C flag ("1") replaces bit 7. This leaves the new value 2AH (00101010B) in destination register 00H. The sign flag and overflow flag are both cleared to "0".

SAMSUNG ELECTRONICS

SB0 — Select Bank 0

SB0

Operation: BANK ← 0

The SB0 instruction clears the bank address flag in the FLAGS register (FLAGS.0) to logic zero, selecting bank 0 register addressing in the set 1 area of the register file.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	4F

Example: The statement

SB0

clears FLAGS.0 to "0", selecting bank 0 register addressing.

SB1 — Select Bank 1

SB1

Operation: BANK ← 1

The SB1 instruction sets the bank address flag in the FLAGS register (FLAGS.0) to logic one, selecting bank 1 register addressing in the set 1 area of the register file. (Bank 1 is not implemented in some S3C8-series microcontrollers.)

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	5F

Example: The statement

SB1

sets FLAGS.0 to "1", selecting bank 1 register addressing, if implemented.

SBC — Subtract with Carry

SBC dst,src

Operation: dst ← dst – src – c

The source operand, along with the current value of the carry flag, is subtracted from the destination operand and the result is stored in the destination. The contents of the source are unaffected. Subtraction is performed by adding the two's-complement of the source operand to the destination operand. In multiple precision arithmetic, this instruction permits the carry ("borrow") from the subtraction of the low-order operands to be subtracted from the subtraction of high-order operands.

Flags:

C: Set if a borrow occurred (src > dst); cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite sign and the sign of the result is the same as the sign of the source; cleared otherwise.
D: Always set to "1".
H: Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise, indicating a "borrow".

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	32	r	r
		6	33	r	lr
opc | src | dst	3	6	34	R	R
		6	35	R	IR
opc | dst | src	3	6	36	R	IM

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

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

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

SCF — Set Carry Flag

SCF

Operation: C ← 1

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

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

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	DF

Example: The statement

SCF

sets the carry flag to logic one.

SRA — Shift Right Arithmetic

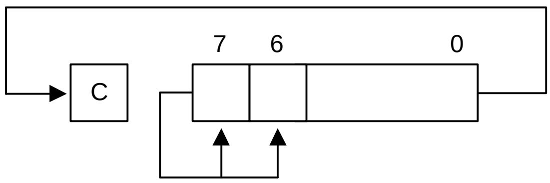
SRA dst

Operation: dst (7) ← dst (7)

C ← dst (0)

dst (n) ← dst (n + 1), n = 0–6

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



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

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	2	4	D0	R
			4	D1	IR

Examples: Given: Register 00H = 9AH, register 02H = 03H, register 03H = 0BCH, and C = "1":

SRA 00H → Register 00H = 0CD, C = "0"

SRA @02H → Register 02H = 03H, register 03H = 0DEH, C = "0"

In the first example, if general register 00H contains the value 9AH (10011010B), the statement "SRA 00H" shifts the bit values in register 00H right one bit position. Bit zero ("0") clears the C flag and bit 7 ("1") is then shifted into the bit 6 position (bit 7 remains unchanged). This leaves the value 0CDH (11001101B) in destination register 00H.

SRP/SRP0/SRP1 — Set Register Pointer

SRP src

SRP0 src

SRP1 src

Operation:

If src (1)	=	1 and src (0)	=	0 then:	RP0 (3–7)	←	src (3–7)
If src (1)	=	0 and src (0)	=	1 then:	RP1 (3–7)	←	src (3–7)
If src (1)	=	0 and src (0)	=	0 then:	RP0 (4–7)	←	src (4–7),
					RP0 (3)	←	0
					RP1 (4–7)	←	src (4–7),
					RP1 (3)	←	1

The source data bits one and zero (LSB) determine whether to write one or both of the register pointers, RP0 and RP1. Bits 3–7 of the selected register pointer are written unless both register pointers are selected. RP0.3 is then cleared to logic zero and RP1.3 is set to logic one.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode src
opc | src	2	4	31	IM

Examples: The statement

SRP #40H

sets register pointer 0 (RP0) at location 0D6H to 40H and register pointer 1 (RP1) at location 0D7H to 48H.

The statement "SRP0 #50H" sets RP0 to 50H, and the statement "SRP1 #68H" sets RP1 to 68H.

SAMSUNG ELECTRONICS

STOP — Stop Operation

STOP

Operation:

The STOP instruction stops the both the CPU clock and system clock and causes the microcontroller to enter Stop mode. During Stop mode, the contents of on-chip CPU registers, peripheral registers, and I/O port control and data registers are retained. Stop mode can be released by an external reset operation or by external interrupts. For the reset operation, the RESET pin must be held to Low level until the required oscillation stabilization interval has elapsed.

In application programs, a STOP instruction must be immediately followed by at least three NOP instructions. This ensures an adeguate time interval for the clock to stabilize before the next instruction is executed. If three or more NOP instructons are not used after STOP instruction, leakage current could be flown because of the floating state in the internal bus.

Flags: No flags are affected.

Format:

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

Example: The statement

```
STOP          ;   halts all microcontroller operations
NOP
NOP
NOP
```

SUB — Subtract

SUB dst,src

Operation: dst ← dst – src

The source operand is subtracted from the destination operand and the result is stored in the destination. The contents of the source are unaffected. Subtraction is performed by adding the two's complement of the source operand to the destination operand.

Flags:

C: Set if a "borrow" occurred; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite signs and the sign of the result is of the same as the sign of the source operand; cleared otherwise.
D: Always set to "1".
H: Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise indicating a "borrow".

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	22	r	r
		6	23	r	Ir
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.

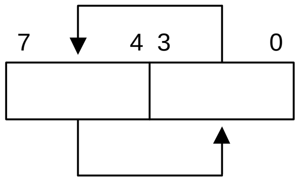
SAMSUNG ELECTRONICS

SWAP — Swap Nibbles

SWAP dst

Operation: dst (0 – 3) ↔ dst (4 – 7)

The contents of the lower four bits and upper four bits of the destination operand are swapped.



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

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc	dst	2	4	F0	R
			4	F1	IR

Examples: Given: Register 00H = 3EH, register 02H = 03H, and register 03H = 0A4H:

SWAP	00H	→	Register 00H = 0E3H
SWAP	@02H	→	Register 02H = 03H, register 03H = 4AH

In the first example, if general register 00H contains the value 3EH (00111110B), the statement "SWAP 00H" swaps the lower and upper four bits (nibbles) in the 00H register, leaving the value 0E3H (11100011B).

TCM — Test Complement Under Mask

TCM dst,src

Operation: (NOT dst) AND src

This instruction tests selected bits in the destination operand for a logic one value. The bits to be tested are specified by setting a "1" bit in the corresponding position of the source operand (mask). The TCM statement complements the destination operand, which is then ANDed with the source mask. The zero (Z) flag can then be checked to determine the result. The destination and source operands are unaffected.

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

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	62	r	r
		6	63	r	lr
opc | src | dst	3	6	64	R	R
		6	65	R	IR
opc | dst | src	3	6	66	R	IM

Examples: Given: R0 = 0C7H, R1 = 02H, R2 = 12H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

TCM	R0,R1	→	R0 = 0C7H, R1 = 02H, Z = "1"
TCM	R0,@R1	→	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = "0"
TCM	00H,01H	→	Register 00H = 2BH, register 01H = 02H, Z = "1"
TCM	00H,@01H	→	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = "1"
TCM	00H,#34	→	Register 00H = 2BH, Z = "0"

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TCM R0,R1" tests bit one in the destination register for a "1" value. Because the mask value corresponds to the test bit, the Z flag is set to logic one and can be tested to determine the result of the TCM operation.

TM — Test Under Mask

TM dst,src

Operation: dst AND src

This instruction tests selected bits in the destination operand for a logic zero value. The bits to be tested are specified by setting a "1" bit in the corresponding position of the source operand (mask), which is ANDed with the destination operand. The zero (Z) flag can then be checked to determine the result. The destination and source operands are unaffected.

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

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	72	r	r
		6	73	r	Ir
opc | src | dst	3	6	74	R	R
		6	75	R	IR
opc | dst | src	3	6	76	R	IM

Examples: Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

TM	R0,R1	→	R0 = 0C7H, R1 = 02H, Z = "0"
TM	R0,@R1	→	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = "0"
TM	00H,01H	→	Register 00H = 2BH, register 01H = 02H, Z = "0"
TM	00H,@01H	→	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = "0"
TM	00H,#54H	→	Register 00H = 2BH, Z = "1"

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TM R0,R1" tests bit one in the destination register for a "0" value. Because the mask value does not match the test bit, the Z flag is cleared to logic zero and can be tested to determine the result of the TM operation.

WFI — Wait for Interrupt

WFI

Operation:

The CPU is effectively halted until an interrupt occurs, except that DMA transfers can still take place during this wait state. The WFI status can be released by an internal interrupt, including a fast interrupt.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4n	3F

(n = 1, 2, 3, …)

Example: The following sample program structure shows the sequence of operations that follow a "WFI" statement:

Main program
.
.
.
EI (Enable global interrupt)
WFI (Wait for interrupt)
(Next instruction)
.
.
.
Interrupt occurs
Interrupt service routine
.
.
.
Clear interrupt flag
IRET
Service routine completed

XOR — Logical Exclusive OR

XOR dst,src

Operation: dst ← dst XOR src

The source operand is logically exclusive-ORed with the destination operand and the result is stored in the destination. The exclusive-OR operation results in a "1" bit being stored whenever the corresponding bits in the operands are different; otherwise, a "0" bit is stored.

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

Format:

Format	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	B2	r	r
		6	B3	r	Ir
opc | src | dst	3	6	B4	R	R
		6	B5	R	IR
opc | dst | src	3	6	B6	R	IM

Examples: Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

XOR	R0,R1	→	R0 = 0C5H, R1 = 02H
XOR	R0,@R1	→	R0 = 0E4H, R1 = 02H, register 02H = 23H
XOR	00H,01H	→	Register 00H = 29H, register 01H = 02H
XOR	00H,@01H	→	Register 00H = 08H, register 01H = 02H, register 02H = 23H
XOR	00H,#54H	→	Register 00H = 7FH

In the first example, if working register R0 contains the value 0C7H and if register R1 contains the value 02H, the statement "XOR R0,R1" logically exclusive-ORs the R1 value with the R0 value and stores the result (0C5H) in the destination register R0.

7 CLOCK CIRCUIT

OVERVIEW

The S3C8275X/C8278X/C8274X microcontroller has two oscillator circuits: a main clock and a sub clock circuit. The CPU and peripheral hardware operate on the system clock frequency supplied through these circuits. The maximum CPU clock frequency of S3C8275X/C8278X/C8274X is determined by CLKCON register settings.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

- External crystal, ceramic resonator, RC oscillation source, or an external clock source
- Oscillator stop and wake-up functions
- Programmable frequency divider for the CPU clock (fxx divided by 1, 2, 8, or 16)
- System clock control register, CLKCON
- Oscillator control register, OSCCON and STOP control register, STPCON
- Clock output control register, CLOCON

CPU Clock Notation

In this document, the following notation is used for descriptions of the CPU clock;

fx: main clock
fxt: sub clock
fxx: selected system clock

MAIN OSCILLATOR CIRCUITS

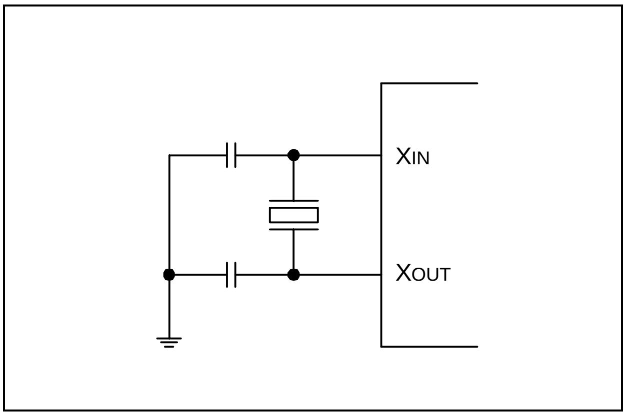


Figure 7-1. Crystal/Ceramic Oscillator (fx)

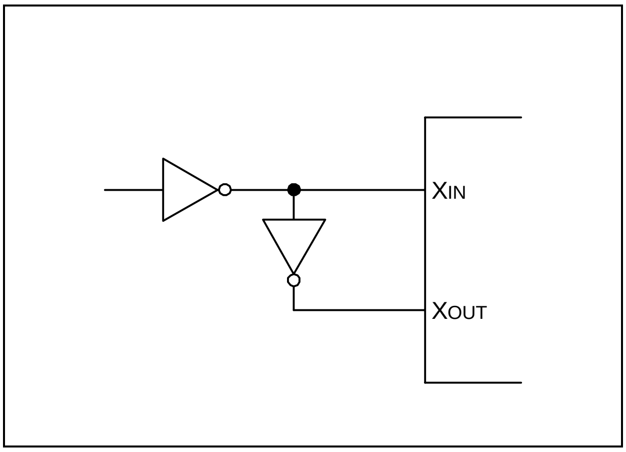


Figure 7-2. External Oscillator (fx)

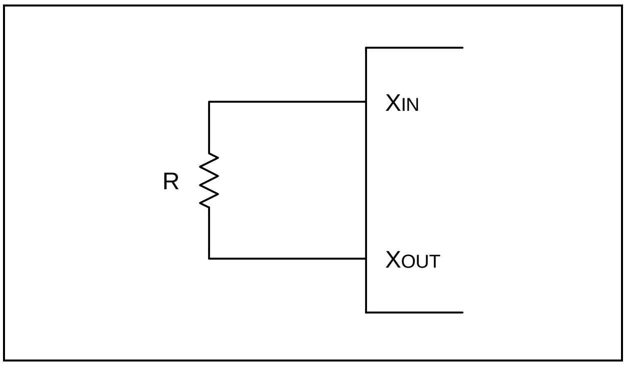


Figure 7-3. RC Oscillator (fx)

SUB OSCILLATOR CIRCUITS

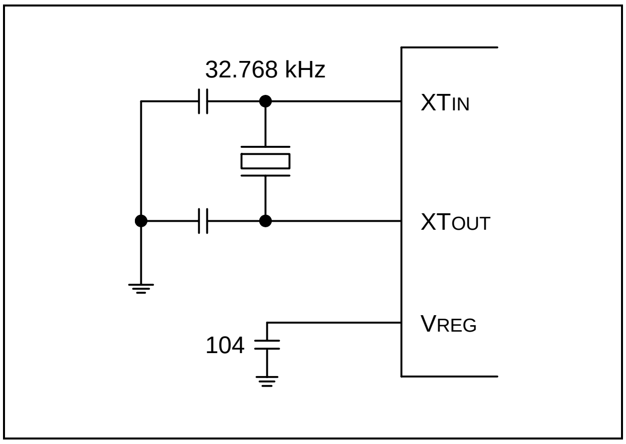


Figure 7-4. Crystal Oscillator (fxt)

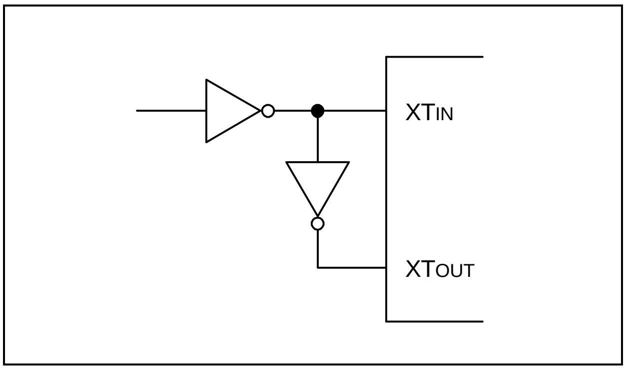


Figure 7-5. External Oscillator (fxt)

CLOCK STATUS DURING POWER-DOWN MODES

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

- In stop mode, the main oscillator is halted. Stop mode is released, and the oscillator started, by a reset operation or an external interrupt (with RC delay noise filter).
- In Idle mode, the internal clock signal is gated to the CPU, but not to interrupt structure, timers and timer/ counters. Idle mode is released by a reset or by an external or internal interrupt.

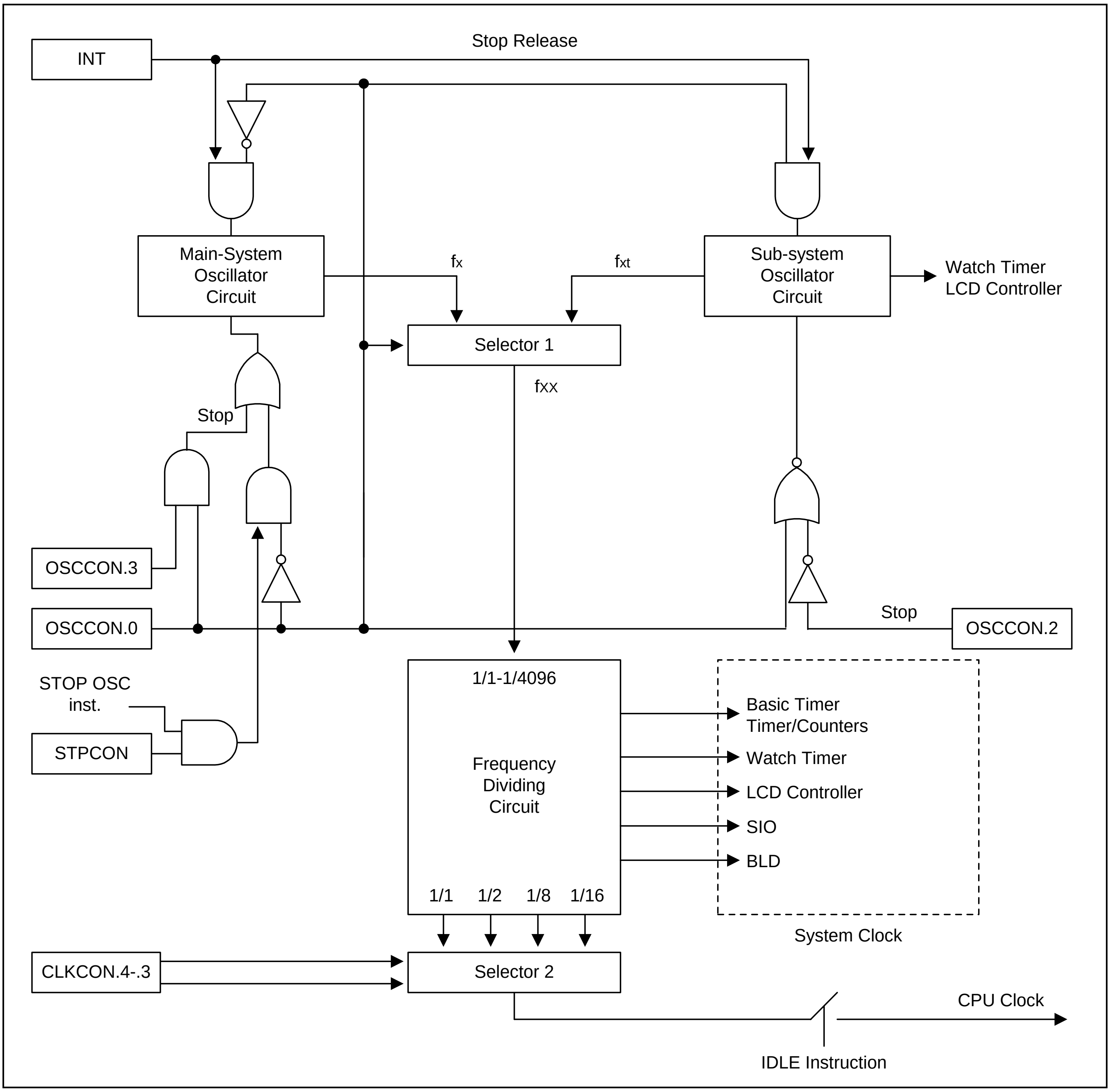


Figure 7-6. System Clock Circuit Diagram

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

The system clock control register, CLKCON, is located in the set 1, at address D4H. It is read/write addressable and has the following functions:

- Oscillator IRQ wake up function enable/disable
- Oscillator frequency divide-by value

CLKCON register settings control 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 the main oscillator is activated, and the fxx/16 (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed to fxx/8, fxx/2, or fxx/1.

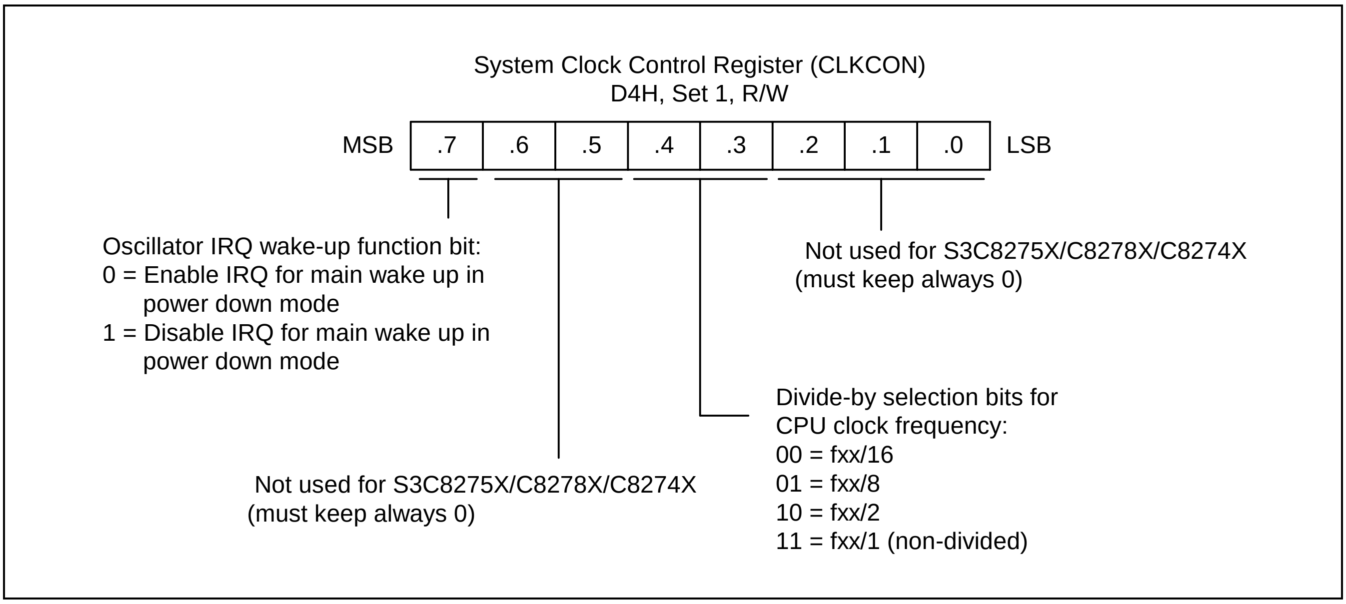


Figure 7-7. System Clock Control Register (CLKCON)

CLOCK OUTPUT CONTROL REGISTER (CLOCON)

The clock output control register, CLOCON, is located in set 1 bank 1, at address E8H. It is read/write addressable and has the following functions:

- Clock output frequency selection

After a reset, fxx/64 is select for clock output frequency because the reset value of CLOCON.1–.0 is "00b".

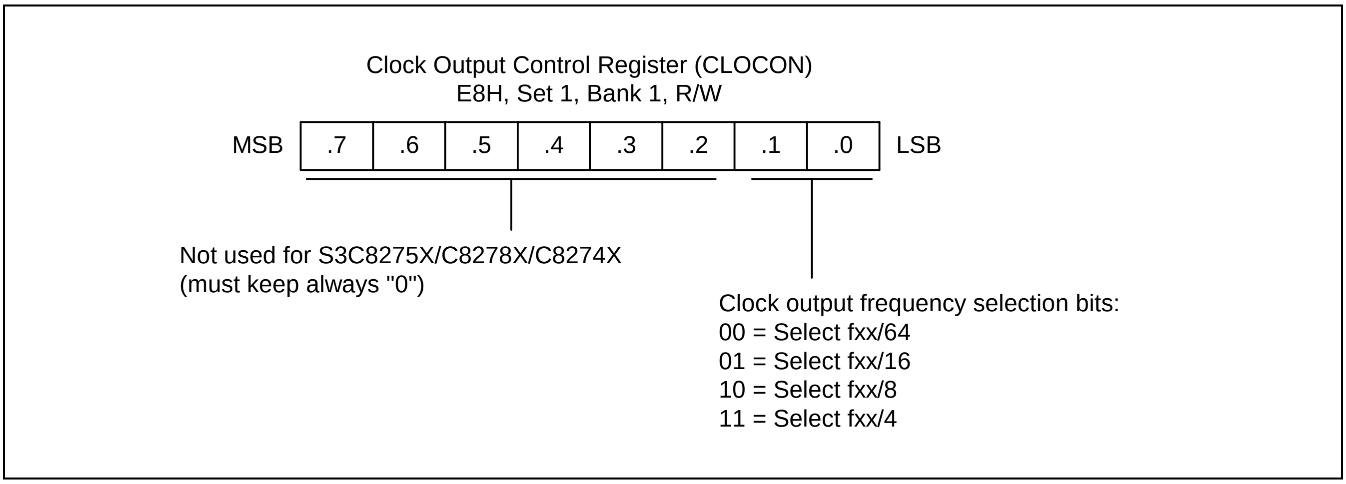


Figure 7-8. Clock Output Control Register (CLOCON)

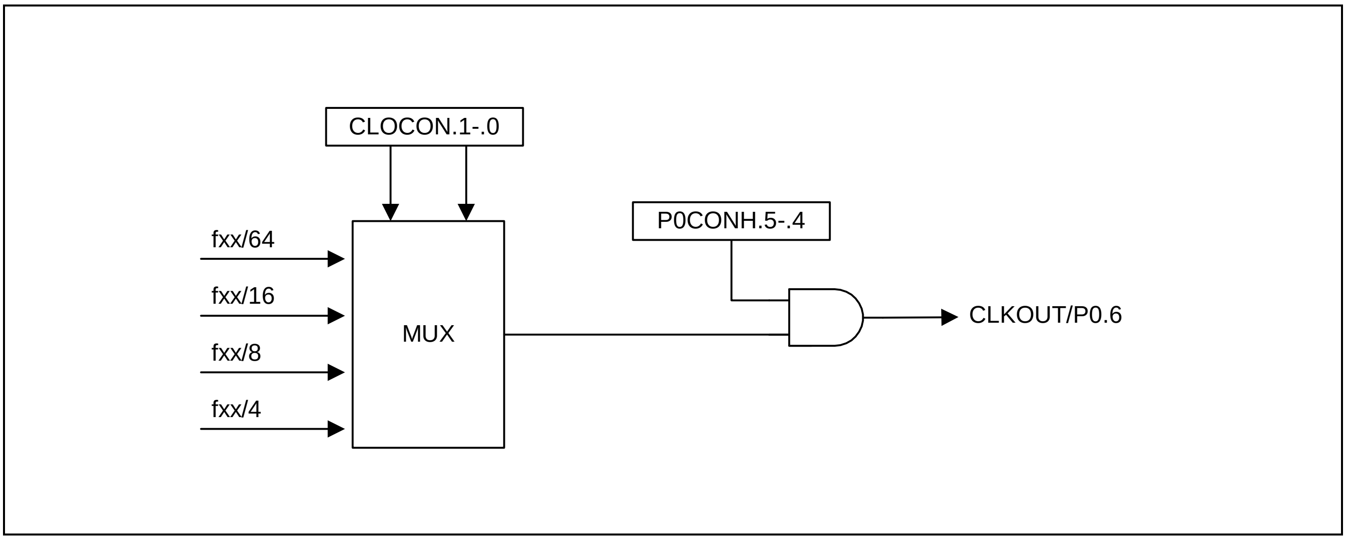


Figure 7-9. Clock Output Block Diagram

OSCILLATOR CONTROL REGISTER (OSCCON)

The oscillator control register, OSCCON, is located in set 1, bank 0, at address E0H. It is read/write addressable and has the following functions:

- System clock selection
- Main oscillator control
- Sub oscillator control
- Sub oscillator circuit selection

OSCCON.0 register settings select Main clock or Sub clock as system clock.
After a reset, Main clock is selected for system clock because the reset value of OSCCON.0 is "0".

The main oscillator can be stopped or run by setting OSCCON.3.

The sub oscillator can be stopped or run by setting OSCCON.2.

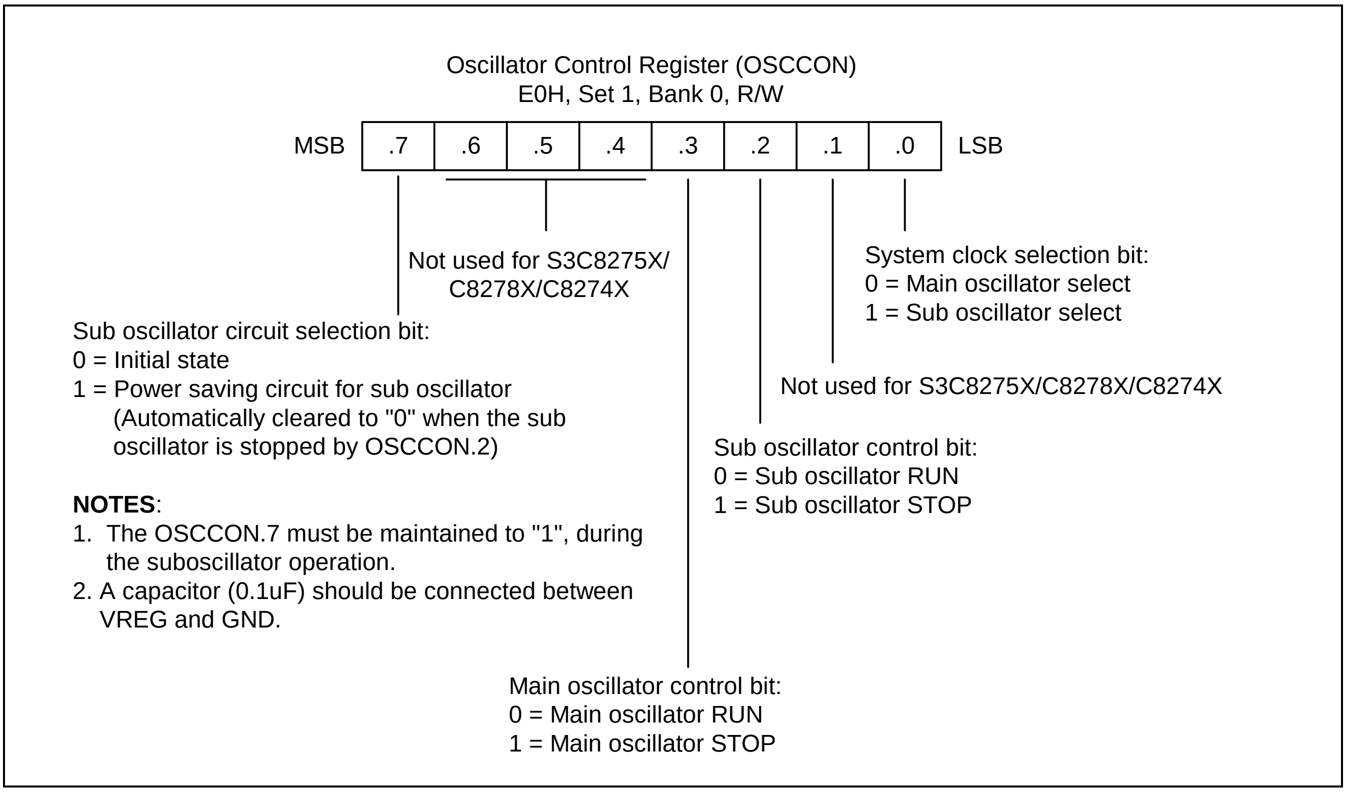


Figure 7-10. Oscillator Control Register (OSCCON)

SAMSUNG ELECTRONICS

SWITCHING THE CPU CLOCK

Data loading in the oscillator control register, OSCCON, determine whether a main or a sub clock is selected as the CPU clock, and also how this frequency is to be divided by setting CLKCON. This makes it possible to switch dynamically between main and sub clocks and to modify operating frequencies.

OSCCON.0 select the main clock (fx) or the sub clock (fxt) for the CPU clock. OSCCON.3 start or stop main clock oscillation, and OSCCON.2 start or stop sub clock oscillation. CLKCON.4–.3 control the frequency divider circuit, and divide the selected fxx clock by 1, 2, 8, 16.

For example, you are using the default CPU clock (normal operating mode and a main clock of fx/16) and you want to switch from the fx clock to a sub clock and to stop the main clock. To do this, you need to set CLKCON.4-.3 to "11", OSCCON.0 to “1”, and OSCCON.3 to “1” simultaneously. This switches the clock from fx to fxt and stops main clock oscillation.

The following steps must be taken to switch from a sub clock to the main clock: first, set OSCCON.3 to “0” to enable main clock oscillation. Then, after a certain number of machine cycles has elapsed, select the main clock by setting OSCCON.0 to “0”.

☞ PROGRAMMING TIP — Switching the CPU clock

1. This example shows how to change from the main clock to the sub clock:

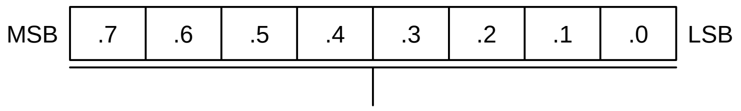
```
MA2SUB   LD      OSCCON,#01H        ;  Switches to the sub clock
                                    ;  Stop the main clock oscillation

         RET
```

2. This example shows how to change from sub clock to main clock:

```
SUB2MA   AND     OSCCON,#07H        ;  Start the main clock oscillation
         CALL    DLY16              ;  Delay 16 ms
         AND     OSCCON,#06H        ;  Switch to the main clock
         RET
DLY16    SRP     #0C0H
         LD      R0,#20H
DEL      NOP
         DJNZ    R0,DEL
         RET
```

STOP Control Register (STPCON)
FBH, Set 1, Bank 0, R/W



STOP control bits:
Other values = Disable STOP instruction
10100101 = Enable STOP instruction

NOTE: Before execute the STOP instruction, set this STPCON register as "10100101B". Otherwise the STOP instuction will not execute as well as reset will be generated.

Figure 7-11. STOP Control Register (STPCON)

8 RESET and POWER-DOWN

SYSTEM RESET

OVERVIEW

During a power-on reset, the voltage at V_{DD} goes to High level and the nRESET pin is forced to Low level. The nRESET signal is input through a schmitt trigger circuit where it is then synchronized with the CPU clock. This procedure brings the S3C8275X/C8278X/C8274X into a known operating status.

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

Whenever a reset occurs during normal operation (that is, when both V_{DD} and nRESET are High level), the nRESET pin is forced Low level and the reset operation starts. All system and peripheral control registers are then reset to their default hardware values

In summary, the following sequence of events occurs during a reset operation:

- All interrupt is disabled.
- The watchdog function (basic timer) is enabled.
- Ports 0-6 are set to input mode, and all pull-up resistors are disabled for the I/O port.
- Peripheral control and data register settings are disabled and reset to their default hardware values.
- The program counter (PC) is loaded with the program reset address in the ROM, 0100H.
- When the programmed oscillation stabilization time interval has elapsed, the instruction stored in ROM location 0100H (and 0101H) is fetched and executed at normal mode by smart option.
- The reset address of ROM can be changed by a smart option only in the S3F8275X (Full-Flash Device). Refer to the chapter 16. Embedded Flash Memory Interface for more detail contents.

NORMAL MODE RESET OPERATION

In normal (masked ROM) mode, the Test pin is tied to V_{SS}. A reset enables access to the 16/8/4-Kbyte on-chip ROM (The external interface is not automatically configured).

NOTE

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

HARDWARE RESET VALUES

Table 8-1, 8-2, 8-3 list the reset values for CPU and system registers, peripheral control registers, and peripheral data registers following a reset operation. The following notation is used to represent reset values:

- A "1" or a "0" shows the reset bit value as logic one or logic zero, respectively.
- An "x" means that the bit value is undefined after a reset.
- A dash ("–") means that the bit is either not used or not mapped, but read 0 is the bit value.

Table 8-1. S3C8275X/C8278X/C8274X Set 1 Register and Values After RESET

Register Name	Mnemonic	Address		Bit Values After RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Locations D0H – D2H are not mapped.											
Basic timer control register	BTCON	211	D3H	0	0	0	0	0	0	0	0
System clock control register	CLKCON	212	D4H	0	–	–	0	0	–	–	–
System flags register	FLAGS	213	D5H	x	x	x	x	x	x	0	0
Register pointer 0	RP0	214	D6H	1	1	0	0	0	–	–	–
Register pointer 1	RP1	215	D7H	1	1	0	0	1	–	–	–
Stack pointer (high byte)	SPH	216	D8H	x	x	x	x	x	x	x	x
Stack pointer (low byte)	SPL	217	D9H	x	x	x	x	x	x	x	x
Instruction pointer (high byte)	IPH	218	DAH	x	x	x	x	x	x	x	x
Instruction pointer (low byte)	IPL	219	DBH	x	x	x	x	x	x	x	x
Interrupt request register	IRQ	220	DCH	0	0	0	0	0	0	0	0
Interrupt mask register	IMR	221	DDH	x	x	x	x	x	x	x	x
System mode register	SYM	222	DEH	0	–	–	x	x	x	0	0
Register page pointer	PP	223	DFH	0	0	0	0	0	0	0	0

Table 8-2. S3C8275X/C8278X/C8274X Set 1, Bank 0 Register Values After RESET

Register Name	Mnemonic	Address		Bit Values After RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
Oscillator control register	OSCCON	224	E0H	0	–	–	–	0	0	–	0
SIO control register	SIOCON	225	E1H	0	0	0	0	0	0	0	0
SIO data register	SIODATA	226	E2H	0	0	0	0	0	0	0	0
SIO pre-scaler register	SIOPS	227	E3H	0	0	0	0	0	0	0	0
Port 0 control register (high byte)	P0CONH	228	E4H	0	0	0	0	0	0	0	0
Port 0 control register (low byte)	P0CONL	229	E5H	0	0	0	0	0	0	0	0
Port 0 pull-up resistor enable register	P0PUR	230	E6H	0	0	0	0	0	0	0	0
Port 1 control register (high byte)	P1CONH	231	E7H	0	0	0	0	0	0	0	0
Port 1 control register (low byte)	P1CONL	232	E8H	0	0	0	0	0	0	0	0
Port 1 pull-up resistor enable register	P1PUR	233	E9H	0	0	0	0	0	0	0	0
Port 2 control register (high byte)	P2CONH	234	EAH	0	0	0	0	0	0	0	0
Port 2 control register (low byte)	P2CONL	235	EBH	0	0	0	0	0	0	0	0
Port 2 pull-up resistor enable register	P2PUR	236	ECH	0	0	0	0	0	0	0	0
Port 3 control register (high byte)	P3CONH	237	EDH	0	0	0	0	0	0	0	0
Port 3 control register (low byte)	P3CONL	238	EEH	0	0	0	0	0	0	0	0
Port 3 pull-up resistor enable register	P3PUR	239	EFH	0	0	0	0	0	0	0	0
Port 0 data register	P0	240	F0H	0	0	0	0	0	0	0	0
Port 1 data register	P1	241	F1H	0	0	0	0	0	0	0	0
Port 2 data register	P2	242	F2H	0	0	0	0	0	0	0	0
Port 3 data register	P3	243	F3H	0	0	0	0	0	0	0	0
Port 4 data register	P4	244	F4H	0	0	0	0	0	0	0	0
Port 5 data register	P5	245	F5H	0	0	0	0	0	0	0	0
Port 6 data register	P6	246	F6H	–	–	–	–	0	0	0	0
External interrupt pending register	EXTIPND	247	F7H	0	0	0	0	0	0	0	0
External interrupt control register (high byte)	EXTICONH	248	F8H	0	0	0	0	0	0	0	0
External interrupt control register (low byte)	EXTICONL	249	F9H	0	0	0	0	0	0	0	0
Location FAH is not mapped.											
STOP control register	STPCON	251	FBH	0	0	0	0	0	0	0	0
Location FCH is not mapped.											
Basic timer counter	BTCNT	253	FDH	0	0	0	0	0	0	0	0
Location FEH is not mapped.											
Interrupt priority register	IPR	255	FFH	x	x	x	x	x	x	x	x

Table 8-3. S3C8275X/C8278X/C8274X Set 1, Bank 1 Register Values After RESET

Register Name	Mnemonic	Address		Bit Values After RESET							
		Dec	Hex	7	6	5	4	3	2	1	0
LCD control register	LCON	224	E0H	0	0	0	0	0	0	–	0
Watch timer control register	WTCON	225	E1H	0	0	0	0	0	0	0	0
Timer A counter	TACNT	226	E2H	0	0	0	0	0	0	0	0
Timer B counter	TBCNT	227	E3H	0	0	0	0	0	0	0	0
Timer A data register	TADATA	228	E4H	1	1	1	1	1	1	1	1
Timer B data register	TBDATA	229	E5H	1	1	1	1	1	1	1	1
Timer 1/A control register	TACON	230	E6H	0	0	0	0	0	0	0	0
Timer B control register	TBCON	231	E7H	–	0	0	0	0	0	0	0
Clock output control register	CLOCON	232	E8H	–	–	–	–	–	–	0	0
Port 4 control register (high byte)	P4CONH	233	E9H	0	0	0	0	0	0	0	0
Port 4 control register (low byte)	P4CONL	234	EAH	0	0	0	0	0	0	0	0
Port 5 control register (high byte)	P5CONH	235	EBH	0	0	0	0	0	0	0	0
Port 5 control register (low byte)	P5CONL	236	ECH	0	0	0	0	0	0	0	0
Port 6 control register	P6CON	237	EDH	0	0	0	0	0	0	0	0
Locations EEH – EFH are not mapped.											
Flash memory control register	FMCON	240	F0H	0	0	0	0	0	–	–	0
Flash memory user programming enable register	FMUSR	241	F1H	0	0	0	0	0	0	0	0
Flash memory sector address register (high byte)	FMSECH	242	F2H	0	0	0	0	0	0	0	0
Flash memory sector address register (low byte)	FMSECL	243	F3H	0	0	0	0	0	0	0	0
Battery level detector control register	BLDCON	244	F4H	–	–	0	0	0	0	0	0
Locations F5H – FFH are not mapped.											

POWER-DOWN MODES

STOP MODE

Stop mode is invoked by the instruction STOP (opcode 7FH). In Stop mode, the operation of the CPU and all peripherals is halted. That is, the on-chip main oscillator stops and the supply current is reduced to less than 3 μA. All system functions stop when the clock "freezes", but data stored in the internal register file is retained. Stop mode can be released in one of two ways: by a reset or by interrupts, for more details see Figure 7-7.

NOTE

Do not use stop mode if you are using an external clock source because X_{IN} or XT_{IN} input must be restricted internally to V_{SS} to reduce current leakage.

Using nRESET to Release Stop Mode

Stop mode is released when the nRESET signal is released and returns to high level: all system and peripheral control registers are reset to their default hardware values and the contents of all data registers are retained. A reset operation automatically selects a slow clock fxx/16 because CLKCON.3 and CLKCON.4 are cleared to '00B'. After the programmed oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in ROM location 0100H (and 0101H)

Using an External Interrupt to Release Stop Mode

External interrupts with an RC-delay noise filter circuit can be used to release Stop mode. Which interrupt you can use to release Stop mode in a given situation depends on the microcontroller's current internal operating mode. The external interrupts in the S3C8275X/C8278X/C8274X interrupt structure that can be used to release Stop mode are:

- External interrupts P0.0–P0.2 (INT0–INT2) and P1.3–P1.7 (INT3–INT7)

Please note the following conditions for Stop mode release:

- If you release Stop mode using an external interrupt, the current values in system and peripheral control registers are unchanged except STPCON register.
- If you use an internal or external interrupt for Stop mode release, you can also program the duration of the oscillation stabilization interval. To do this, you must make the appropriate control and clock settings *before* entering Stop mode.
- When the Stop mode is released by external interrupt, the CLKCON.4 and CLKCON.3 bit-pair setting remains unchanged and the currently selected clock value is used.
- The external interrupt is serviced when the Stop mode release occurs. Following the IRET from the service routine, the instruction immediately following the one that initiated Stop mode is executed.

How to Enter into Stop Mode

Handling STPCON register then writing Stop instruction (keep the order).

```
LD      STPCON, #10100101B
STOP
NOP
NOP
NOP
```

IDLE MODE

Idle mode is invoked by the instruction IDLE (opcode 6FH). In idle mode, CPU operations are halted while some peripherals remain active. During idle mode, the internal clock signal is gated away from the CPU, but all peripherals remain active. Port pins retain the mode (input or output) they had at the time idle mode was entered.

There are two ways to release idle mode:

1. Execute a reset. All system and peripheral control registers are reset to their default values and the contents of all data registers are retained. The reset automatically selects the slow clock fxx/16 because CLKCON.4 and CLKCON.3 are cleared to ‘00B’. If interrupts are masked, a reset is the only way to release idle mode.
2. Activate any enabled interrupt, causing idle mode to be released. When you use an interrupt to release idle mode, the CLKCON.4 and CLKCON.3 register values remain unchanged, and the currently selected clock value is used. The interrupt is then serviced. When the return-from-interrupt (IRET) occurs, the instruction immediately following the one that initiated idle mode is executed.

9 I/O PORTS

OVERVIEW

The S3C8275X/C8278X/C8274X microcontroller has seven bit-programmable I/O ports, P0–P6. Port 0–port 5 are 8-bit ports, port 6 is 4-bit. This gives a total of 52 I/O pins. Each port can be flexibly configured to meet application design requirements.

The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required. All ports of the S3C8275X/C8278X/C8274X can be configured to input or output mode. P2–P6 are shared with LCD signals.

Table 9-1 gives you a general overview of S3C8275X/C8278X/C8274X I/O port functions.

Table 9-1. S3C8275X/C8278X/C8274X Port Configuration Overview

Port	Configuration Options
0	1-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups. Alternatively P0.0–P0.2 can be used as input for external interrupts INT and P0.3-P0.7 can be used as T1CLK, TAOUT, TBOUT, CLKOUT, and BUZ.
1	1-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups. Alternatively P1.3–P1.7 can be used as input for external interrupts INT and P1.0–P1.2 can be used as SCK, SO, and SI.
2	1-bit programmable I/O port. Input or push-pull, open-drain output and software assignable pull-ups. Alternatively P2 can be used as outputs for LCD segment signals.
3	1-bit programmable I/O port. Input or push-pull, open-drain output and software assignable pull-ups. Alternatively P3 can be used as outputs for LCD segment signals.
4	1-bit programmable I/O port. Input or push-pull output and software assignable pull-ups. Alternatively P4 can be used as outputs for LCD segment signals.
5	1-bit programmable I/O port. Input or push-pull output and software assignable pull-ups. Alternatively P5 can be used as outputs for LCD segment signals.
6	1-bit programmable I/O port. Input or push-pull output and software assignable pull-ups. Alternately P6.0–P6.3 can be used as outputs for LCD common signals.

PORT DATA REGISTERS

Table 9-2 gives you an overview of the register locations of all seven S3C8275X/C8278X/C8274X I/O port data registers. Data registers for ports 0, 1, 2, 3, 4, 5, and 6 have the general format shown in Figure 9-1.

Table 9-2. Port Data Register Summary

Register Name	Mnemonic	Decimal	Hex	Location	R/W
Port 0 data register	P0	240	F0H	Set 1, Bank 0	R/W
Port 1 data register	P1	241	F1H	Set 1, Bank 0	R/W
Port 2 data register	P2	242	F2H	Set 1, Bank 0	R/W
Port 3 data register	P3	243	F3H	Set 1, Bank 0	R/W
Port 4 data register	P4	244	F4H	Set 1, Bank 0	R/W
Port 5 data register	P5	245	F5H	Set 1, Bank 0	R/W
Port 6 data register	P6	246	F6H	Set 1, Bank 0	R/W

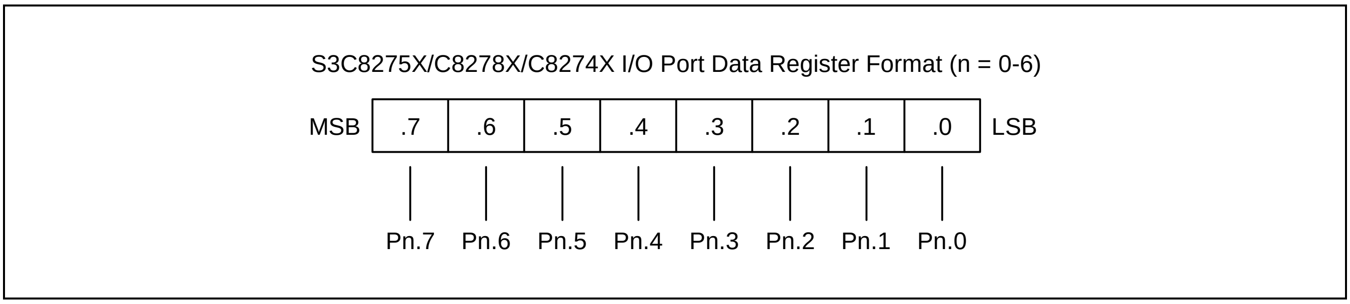


Figure 9-1. S3C8275X/C8278X/C8274X I/O Port Data Register Format

PORT 0

Port 0 is an 8-bit I/O port with individually configurable pins. Port 0 pins are accessed directly by writing or reading the port 0 data register, P0 at location F0H in set 1, bank 0. P0.0-P0.7 can serve as inputs (with or without pull-up), as outputs (push-pull or open-drain) or you can be configured the following functions.

- Low-nibble pins (P0.0–P0.3): INT0–INT2, T1CLK
- High-nibble pins (P0.4–P0.7): TAOUT, TBOUT, CLKOUT, BUZ

Port 0 Control Registers (P0CONH, P0CONL)

Port 0 has two 8-bit control registers: P0CONH for P0.4–P0.7 and P0CONL for P0.0–P0.3. A reset clears the P0CONH and P0CONL registers to "00H", configuring P0.0–P0.2 pins to input mode with interrupt and P0.3–P0.7 pins to input mode. You use control registers setting to select input or output mode (push-pull or open-drain) and enable the alternative functions.

When programming this port, please remember that any alternative peripheral I/O function you configure using the port 0 control registers must also be enabled in the associated peripheral module.

Port 0 Pull-up Resistor Control Register (P0PUR)

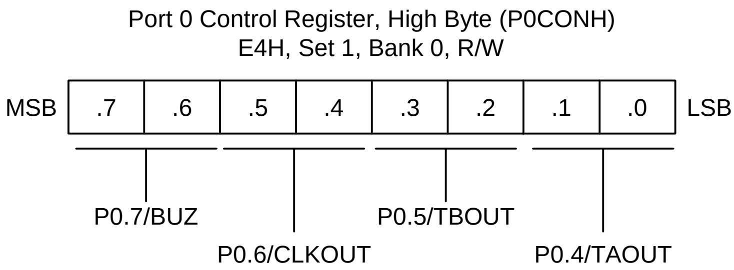
Using the port 0 pull-up resistor control register, P0PUR (E6H, set 1, bank 0) you can configure pull-up resistors to individual port 0 pins.

Port 0 Interrupt Control Registers (EXTICONL.5–.0, EXTIPND.2–.0)

To process external interrupts at the port 0 pins, two additional control registers are provided: the external interrupt control register EXTICONL.5–.0 (F9H, set 1, bank 0) and the external interrupt pending register EXTIPND.2–.0 (F7H, set 1, bank 0)

The external interrupt pending register EXTIPND.2–.0 lets you check for interrupt pending conditions and clear the pending condition when the interrupt service routine has been initiated. The application program detects interrupt requests by polling the EXTIPND.2–.0 register at regular intervals.

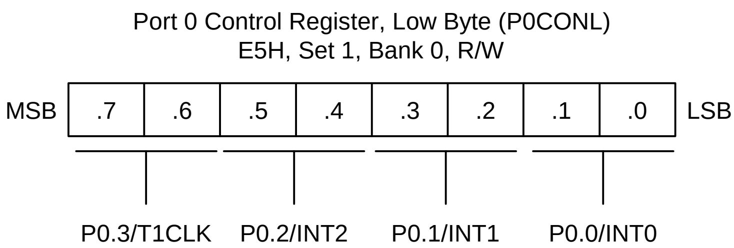
When the interrupt enable bit of any port 0 pin is "1", a rising or falling edge at that pin will generate an interrupt request. The corresponding pending bit is then automatically set to "1" and the IRQ level goes low to signal the CPU that an interrupt request is waiting. When the CPU acknowledges the interrupt request, application software must the clear the pending condition by writing a "0" to the corresponding EXTIPND bit.



P0CONH bit-pair pin configuration settings:

00	Schmitt trigger input mode
01	N-channel open-drain output mode
10	Push-pull output mode
11	Alternative function (BUZ, CLKOUT, TBOUT, TAOUT)

Figure 9-2. Port 0 High-Byte Control Register (P0CONH)



P0CONL bit-pair pin configuration settings:

00	Schmitt trigger input mode (T1CLK)
01	N-channel open-drain output mode
10	Push-pull output mode
11	Not available

Figure 9-3. Port 0 Low-Byte Control Register (P0CONL)

Port 0 Pull-up Control Register (P0PUR)
E6H, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	

P0PUR bit configuration settings:

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: A pull-up resistor of port 0 is automatically disabled when the corresponding pin is selected as push-pull output or alternative function.

Figure 9-4. Port 0 Pull-up Control Register (P0PUR)

External Interrupt Control Register, Low Byte (EXTICONL)
E9H, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P1.3/INT3		P0.2/INT2		P0.1/INT1		P0.0/INT0		

EXTICONL bit configuration settings:

00	Disable interrupt
01	Enable interrupt by falling edge
10	Enable interrupt by rising edge
11	Enable interrupt by both falling and rising edge

Figure 9-5. External Interrupt Control Register, Low Byte (EXTICONL)

External Interrupt Pending Register (EXTIPND)
F7H, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P1.7 (INT7)	P1.6 (INT6)	P1.5 (INT5)	P1.4 (INT4)	P1.3 (INT3)	P0.2 (INT2)	P0.1 (INT1)	P0.0 (INT0)	

EXTIPND bit configuration settings:

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

Figure 9-6. External Interrupt Pending Register (EXTIPND)

SAMSUNG ELECTRONICS

PORT 1

Port 1 is an 8-bit I/O port with individually configurable pins. Port 1 pins are accessed directly by writing or reading the port 1 data register, P1 at location F1H in set 1, bank 0. P1.0–P1.7 can serve as inputs (with or without pull-up), as outputs (push-pull or open-drain) or you can be configured the following functions.

- Low-nibble pins (P1.0–P1.3): SCK, SO, SI, INT3
- High-nibble pins (P1.4–P1.7): INT4–INT7

Port 1 Control Registers (P1CONH, P1CONL)

Port 1 has two 8-bit control registers: P1CONH for P1.4–P1.7 and P1CONL for P1.0–P1.3. A reset clears the P1CONH and P1CONL registers to "00H", configuring P1.3–P1.7 pins to input mode with interrupt and P1.0–P1.2 pins to input mode. You use control registers setting to select input or output mode (push-pull or open-drain) and enable the alternative functions.

When programming this port, please remember that any alternative peripheral I/O function you configure using the port 1 control registers must also be enabled in the associated peripheral module.

Port 1 Pull-up Resistor Control Register (P1PUR)

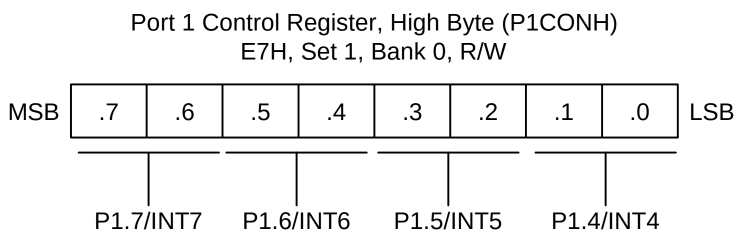
Using the port 1 pull-up resistor control register, P1PUR (E9H, set 1, bank 0), you can configure pull-up resistors to individual port 1 pins.

Port 1 Interrupt Control Registers (EXTICONH, EXTICONL.7–.6, EXTIPND.7–.3)

To process external interrupts at the port 1 pins, three additional control registers are provided: the external interrupt control registers EXTICONH/EXTICONL.7–.6 (F8H/F9H, set 1, bank 0) and the external interrupt pending register EXTIPND.7–.3 (F7H, set 1, bank 0).

The external interrupt pending register EXTIPND.7–.3 lets you check for interrupt pending conditions and clear the pending condition when the interrupt service routine has been initiated. The application program detects interrupt requests by polling the EXTIPND.7–.3 register at regular intervals.

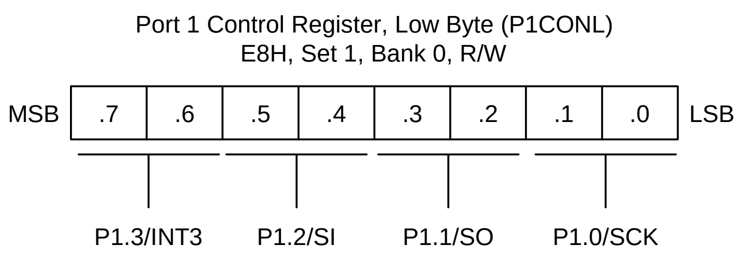
When the interrupt enable bit of any port 1 pin is "1", a rising or falling edge at that pin will generate an interrupt request. The corresponding pending bit is then automatically set to "1" and the IRQ level goes low to signal the CPU that an interrupt request is waiting. When the CPU acknowledges the interrupt request, application software must the clear the pending condition by writing a "0" to the corresponding EXTIPND bit.



P1CONH bit-pair pin configuration settings:

00	Schmitt trigger input mode
01	N-channel open-drain output mode
10	Push-pull output mode
11	Not available

Figure 9-7. Port 1 High-Byte Control Register (P1CONH)

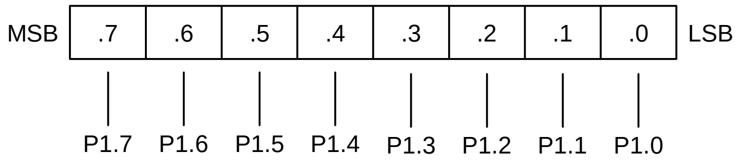


P1CONL bit-pair pin configuration settings:

00	Schmitt trigger input mode (SI, SCK)
01	N-channel open-drain output mode
10	Push-pull output mode
11	Alternative function (SCK, SO)

Figure 9-8. Port 1 Low-Byte Control Register (P1CONL)

Port 1 Pull-up Control Register (P1PUR)
E9H, Set 1, Bank 0, R/W



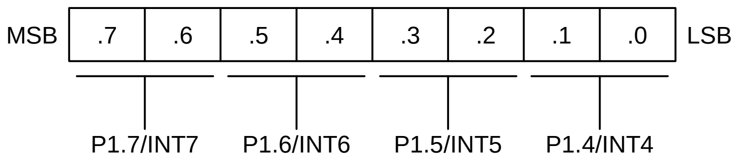
P1PUR bit configuration settings:

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: A pull-up resistor of port 1 is automatically disabled when the corresponding pin is selected as push-pull output or alternative function.

Figure 9-9. Port 1 Pull-up Control Register (P1PUR)

External Interrupt Control Register, High Byte (EXTICONH)
F8H, Set 1, Bank 0, R/W

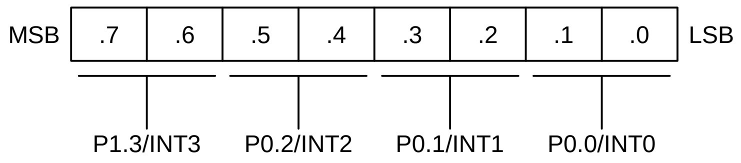


EXTICONH bit configuration settings:

00	Disable interrupt
01	Enable interrupt by falling edge
10	Enable interrupt by rising edge
11	Enable interrupt by both falling and rising edge

Figure 9-10. External Interrupt Control Register, High Byte (EXTICONH)

External Interrupt Control Register, Low Byte (EXTICONL)
F9H, Set 1, Bank 0, R/W

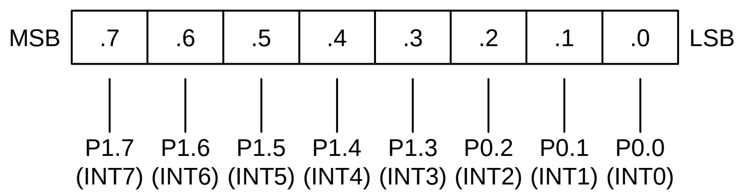


EXTICONL bit configuration settings:

00	Disable interrupt
01	Enable interrupt by falling edge
10	Enable interrupt by rising edge
11	Enable interrupt by both falling and rising edge

Figure 9-11. External Interrupt Control Register, Low Byte (EXTICONL)

External Interrupt Pending Register (EXTIPND)
F7H, Set 1, Bank 0, R/W



EXTIPND bit configuration settings:

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

Figure 9-12. External Interrupt Pending Register (EXTIPND)

PORT 2

Port 2 is an 8-bit I/O port with individually configurable pins. Port 2 pins are accessed directly by writing or reading the port 2 data register, P2 at location F2H in set 1, Bank 0. P2.0-P2.7 can serve as inputs (with or without pull-up), as outputs (push-pull or open-drain) or you can be configured the following functions.

- Low-nibble pins (P2.0-P2.3): SEG31–SEG28, V_{BLDREF}
- High-nibble pins (P2.4-P2.7): SEG27–SEG24

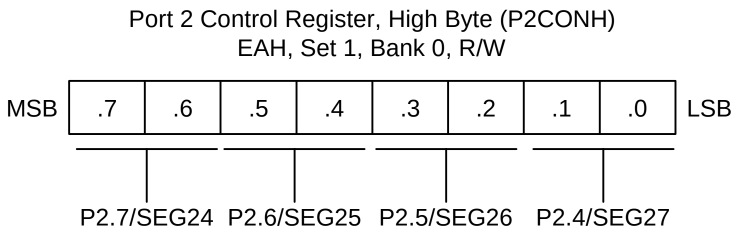
Port 2 Control Registers (P2CONH, P2CONL)

Port 2 has two 8-bit control registers: P2CONH for P2.4-P2.7 and P2CONL for P2.0-P2.3. A reset clears the P2CONH and P2CONL registers to "00H", configuring all pins to input mode. You use control registers setting to select input or output mode (push-pull or open-drain) and enable the alternative functions.

When programming this port, please remember that any alternative peripheral I/O function you configure using the port 2 control registers must also be enabled in the associated peripheral module.

Port 2 Pull-up Resistor Control Register (P2PUR)

Using the port 2 pull-up resistor control register, P2PUR (ECH, set 1, bank 0), you can configure pull-up resistors to individual port 2 pins.



P2CONH bit-pair pin configuration settings:

00	Input mode
01	N-channel open-drain output mode
10	Push-pull output mode
11	Alternative function (SEG24-SEG27)

Figure 9-13. Port 2 High-byte Control Register (P2CONH)

Port 2 Control Register, Low Byte (P2CONL)
EBH, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P2.3/SEG28		P2.2/SEG29		P2.1/SEG30		P2.0/SEG31/V_{BLDREF}		

P2CONL bit-pair pin configuration settings:

00	Input mode
01	N-channel open-drain output mode
10	Push-pull output mode
11	Alternative function (SEG28-SEG31/V_{BLDREF})

Figure 9-14. Port 2 Low-byte Control Register (P2CONL)

Port 2 Pull-up Control Register (P2PUR)
ECH, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	

P2PUR bit configuration settings:

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: A pull-up resistor of port 2 is automatically disabled when the corresponding pin is selected as push-pull output or alternative function.

Figure 9-15. Port 2 Pull-up Control Register (P2PUR)

PORT 3

Port 3 is an 8-bit I/O port with individually configurable pins. Port 3 pins are accessed directly by writing or reading the port 3 data register, P3 at location F3H in set 1, bank 0. P3.0-P3.7 can serve as inputs (with or without pull-up), as outputs (push-pull or open-drain) or you can be configured the following functions.

- Low-nibble pins (P3.0-P3.3): SEG23-SEG20
- High-nibble pins (P3.4-P3.7): SEG19-SEG16

Port 3 Control Registers (P3CONH, P3CONL)

Port 3 has two 8-bit control registers: P3CONH for P3.4-P3.7 and P3CONL for P3.0-P3.3. A reset clears the P3CONH and P3CONL registers to "00H", configuring all pins to input mode. You use control registers setting to select input or output mode (push-pull or open-drain) and enable the alternative functions.

When programming this port, please remember that any alternative peripheral I/O function you configure using the port 3 control registers must also be enabled in the associated peripheral module.

Port 3 Pull-up Resistor Control Register (P3PUR)

Using the port 3 pull-up resistor control register, P3PUR (EFH, set 1, bank 0), you can configure pull-up resistors to individually port 3 pins.

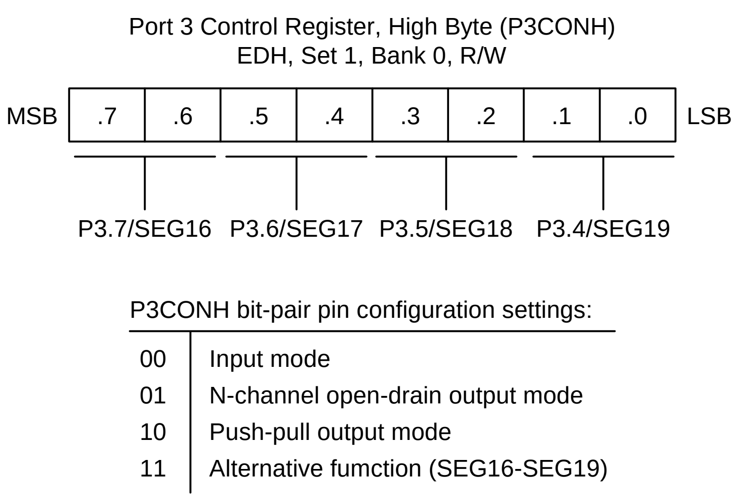


Figure 9-16. Port 3 High Byte Control Register (P3CONH)

Port 3 Control Register, Low Byte (P3CONL)
EEH, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P3.3/SEG20		P3.2/SEG21		P3.1/SEG22		P3.0/SEG23		

P3CONL bit-pair pin configuration settings:

00	Input mode
01	N-channel open-drain output mode
10	Push-pull output mode
11	Alternative function (SEG20-SEG23)

Figure 9-17. Port 3 Low Byte Control Register (P3CONL)

Port 3 Pull-up Control Register (P3PUR)
EFH, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	

P3PUR bit configuration settings:

0	Disable pull-up resistor
1	Enable pull-up resistor

NOTE: A pull-up resistor of port 3 is automatically disabled when the corresponding pin is selected as push-pull output or alternative function.

Figure 9-18. Port 3 Pull-up Control Register (P3PUR)

SAMSUNG ELECTRONICS

PORT 4

Port 4 is an 8-bit I/O port with individually configurable pins. Port 4 pins are accessed directly by writing or reading the port 4 data register, P4 at location F4H in set 1, bank 0. P4.0-P4.7 can serve as inputs (with or without pull-up), as push-pull output or you can be configured the following functions.

- Low-nibble pins (P4.0-P4.3): SEG15-SEG12
- High-nibble pins (P4.4-P4.7): SEG11-SEG8

Port 4 Control Registers (P4CONH, P4CONL)

Port 4 has two 8-bit control registers: P4CONH for P4.4-P4.7 and P4CONL for P4.0-P4.3. A reset clears the P4CONH and P4CONL registers to "00H", configuring all pins to input mode. You use control registers setting to select input (with or without pull-up) or push-pull output mode and enable the alternative functions.

When programming this port, please remember that any alternative peripheral I/O function you configure using the port 4 control registers must also be enabled in the associated peripheral module.

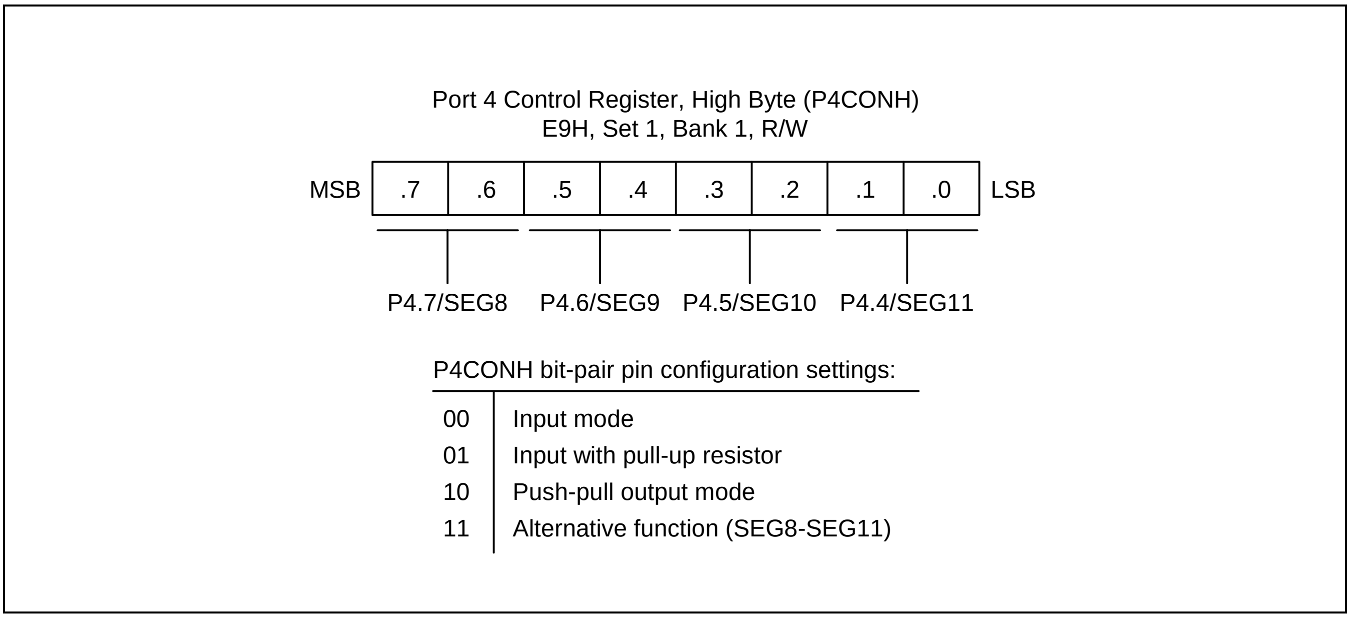
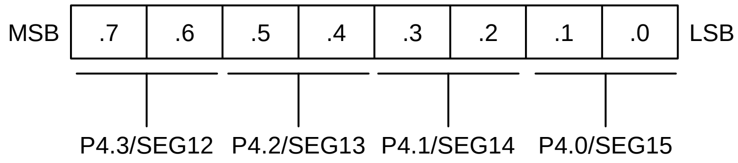


Figure 9-19. Port 4 High-Byte Control Register (P4CONH)

Port 4 Control Register, Low Byte (P4CONL)
EAH, Set 1, Bank 1, R/W



P4CONH bit-pair pin configuration settings:

00	Input mode
01	Input with pull-up resistor
10	Push-pull output mode
11	Alternative function (SEG12-SEG15)

Figure 9-20. Port 4 Low-Byte Control Register (P4CONL)

PORT 5

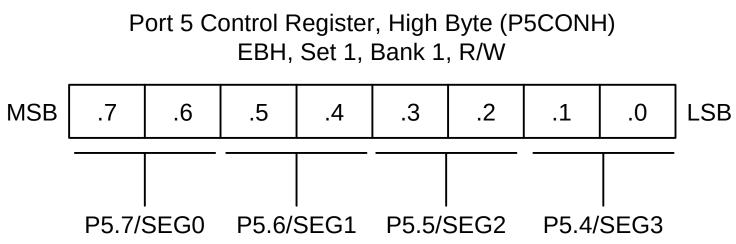
Port 5 is an 8-bit I/O port with individually configurable pins. Port 5 pins are accessed directly by writing or reading the port 5 data register, P5 at location F5H in set 1, bank 0. P5.0-P5.7 can serve as inputs (with or without pull-up), as push-pull output or you can be configured the following functions.

- Low-nibble pins (P5.0-P5.3): SEG7–SEG4
- High-nibble pins (P5.4-P5.7): SEG3–SEG0

Port 5 Control Registers (P5CONH, P5CONL)

Port 5 has two 8-bit control registers: P5CONH for P5.4-P5.7 and P5CONL for P5.0-P5.3. A reset clears the P5CONH and P5CONL registers to "00H", configuring all pins to input mode. You use control registers setting to select input (with or without pull-up) or push-pull output mode and enable the alternative functions.

When programming this port, please remember that any alternative peripheral I/O function you configure using the port 5 control registers must also be enabled in the associated peripheral module.



P5CONH bit-pair pin configuration settings:

00	Input mode
01	Input with pull-up resistor
10	Push-pull output mode
11	Alternative function (SEG0-SEG3)

Figure 9-21. Port 5 High-Byte Control Register (P5CONH)

Port 5 Control Register, Low Byte (P5CONL)
ECH, Set 1, Bank 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P5.3/SEG4		P5.2/SEG5		P5.1/SEG6		P5.0/SEG7		

P5CONL bit-pair pin configuration settings:

00	Input mode
01	Input with pull-up resistor
10	Push-pull output mode
11	Alternative function (SEG4-SEG7)

Figure 9-22. Port 5 Low-Byte Control Register (P5CONL)

PORT 6

Port 6 is a 4-bit I/O port with individually configurable pins. Port 6 pins are accessed directly by writing or reading the port 6 data register, P6 at location F6H in set 1, bank 0. P6.0-P6.3 can serve as inputs (with or without pull-up), as push-pull output or you can be configured the following functions.

- Low-nibble pins (P6.0-P6.3): COM0–COM3

Port 6 Control Register (P6CON)

Port 6 has an 8-bit control register: P6CON for P6.0-P6.3. A reset clears the P6CON register to "00H", configuring all pins to input mode. You use control register setting to select input (with or without pull-up) or push-pull output mode and enable the alternative functions.

When programming this port, please remember that any alternative peripheral I/O function you configure using the port 6 control register must also be enabled in the associated peripheral module.

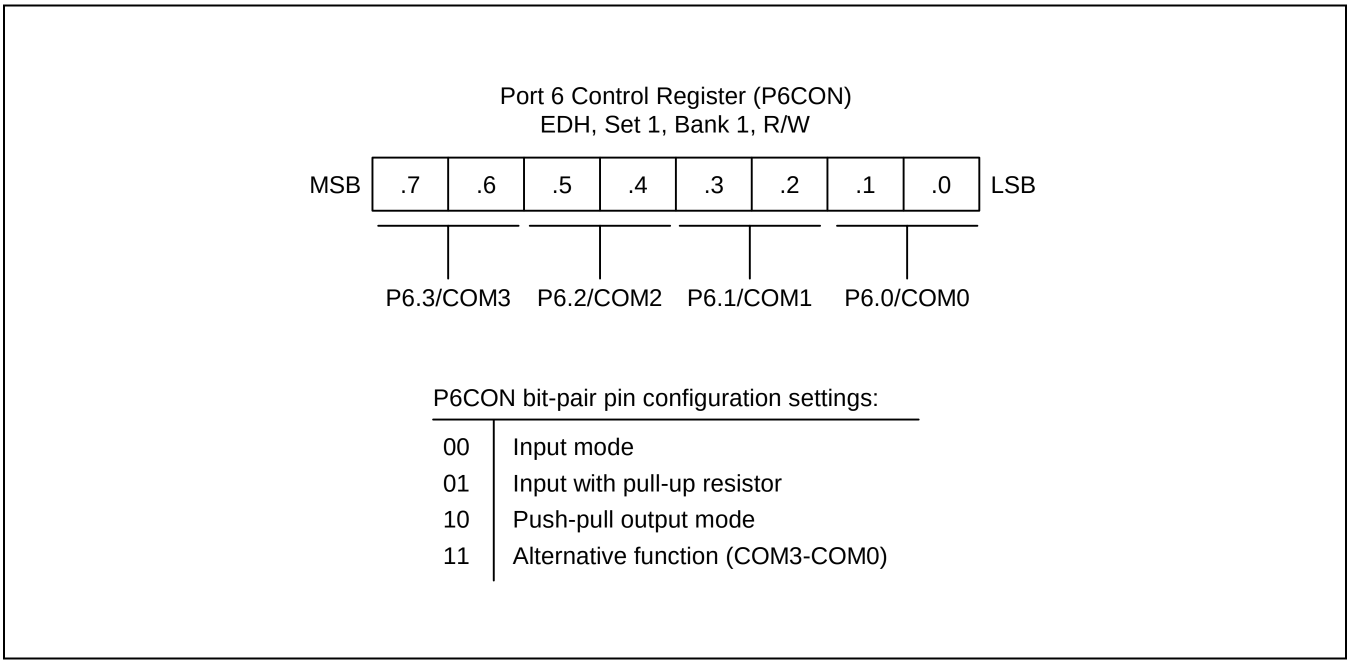


Figure 9-23. Port 6 Control Register (P6CON)

10 BASIC TIMER

OVERVIEW

Basic timer (BT) can be used 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_{XX} divided by 4096, 1024, 128, or 16) with multiplexer
- 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH, read-only)
- Basic timer control register, BTCON (set 1, D3H, read/write)

BASIC TIMER CONTROL REGISTER (BTCON)

The basic timer control register, BTCON, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function. It is located in set 1, address D3H, and is read/write addressable using Register addressing mode.

A reset clears BTCON to "00H". This enables the watchdog function and selects a basic timer clock frequency of fxx/4096. To disable the watchdog function, you must write the signature code "1010B" to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH), can be cleared at any time during normal operation by writing a "1" to BTCON.1. To clear the frequency dividers for all timers input clock, you write a "1" to BTCON.0.

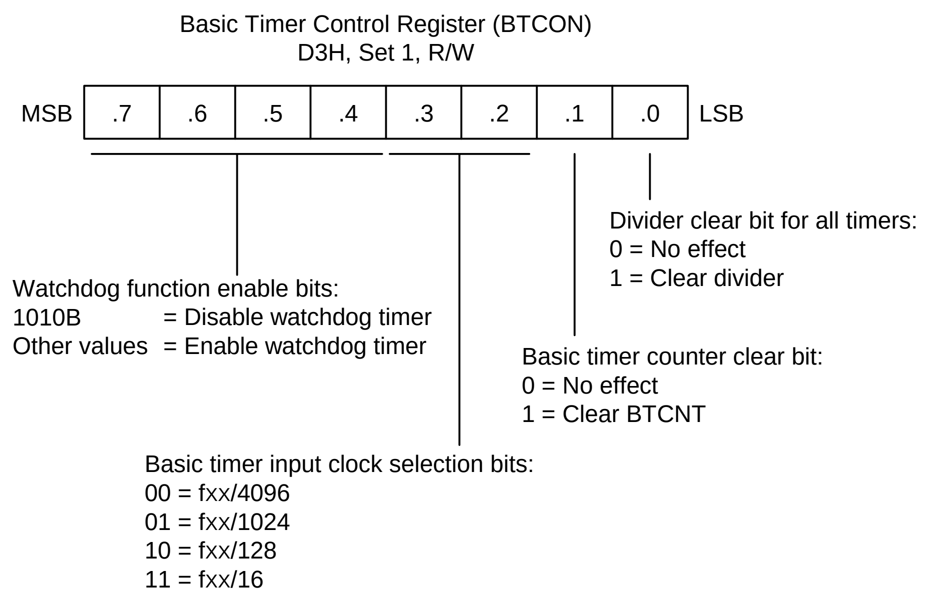


Figure 10-1. Basic Timer Control Register (BTCON)

BASIC TIMER FUNCTION DESCRIPTION

Watchdog Timer Function

You can program the basic timer overflow signal (BTOVF) to generate a reset by setting BTCON.7–BTCON.4 to any value other than "1010B". (The "1010B" value disables the watchdog function.) A reset clears BTCON to "00H", automatically enabling the watchdog timer function. A reset also selects the CPU clock (as determined by the current CLKCON register setting), divided by 4096, as the BT clock.

A reset whenever a basic timer counter overflow occurs. During normal operation, the application program must prevent the overflow, and the accompanying reset operation, from occurring. To do this, the BTCNT value must be cleared (by writing a "1" to BTCON.1) at regular intervals.

If a system malfunction occurs due to circuit noise or some other error condition, the BT counter clear operation will not be executed and a basic timer overflow will occur, initiating a reset. In other words, during normal operation, the basic timer overflow loop (a bit 7 overflow of the 8-bit basic timer counter, BTCNT) is always broken by a BTCNT clear instruction. If a malfunction does occur, a reset is triggered automatically.

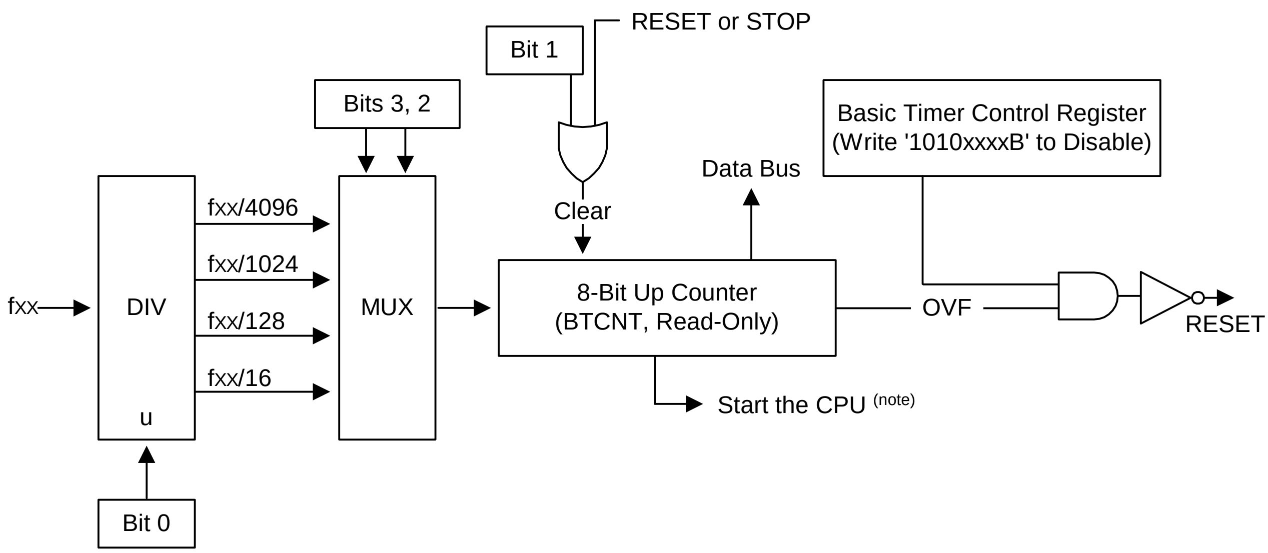
Oscillation Stabilization Interval Timer Function

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

In stop mode, whenever a reset or an internal and an external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of fxx/4096 (for reset), or at the rate of the preset clock source (for an internal and an external interrupt). When BTCNT.4 overflows, a signal is generated to indicate that the stabilization interval has elapsed and to gate the clock signal off to the CPU so that it can resume normal operation.

In summary, the following events occur when stop mode is released:

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



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

Figure 10-2. Basic Timer Block Diagram

11 TIMER 1

ONE 16-BIT TIMER MODE (TIMER 1)

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

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

OVERVIEW

The 16-bit timer 1 is a 16-bit general-purpose timer. Timer 1 has the interval timer mode by using the appropriate TACON setting.

Timer 1 has the following functional components:

- Clock frequency divider (fxx divided by 512, 256, 64, 8, or 1, fxt, and T1CLK: External clock) with multiplexer
- 16-bit counter (TACNT, TBCNT), 16-bit comparator, and 16-bit reference data register (TADATA, TBDATA)
- Timer 1 match interrupt (IRQ 0, vector F0H) generation
- Timer 1 control register, TACON (set 1, bank 1, E6H, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

The timer 1 module can generate an interrupt, the timer 1 match interrupt (T1INT). T1INT belongs to the interrupt level IRQ 0, and is assigned a separate vector address, F0H.

The T1INT pending condition should be cleared by software after IRQ 0 is serviced. The T1INT pending bit must be cleared by the application sub-routine by writing a "0" to the TACON.0 pending bit.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the timer 1 reference data registers, TADATA and TBDATA. The match signal generates a timer 1 match interrupt and clears the counter.

If, for example, you write the value 32H and 10H to TADATA and TBDATA, respectively, and 8EH to TACON, the counter will increment until it reaches 3210H. At this point, the timer 1 interrupt request is generated, the counter value is reset, and counting resumes.

Timer 1 Control Register (TACON)

You use the timer 1 control register, TACON, to

- Enable the timer 1 operating (interval timer)
- Select the timer 1 input clock frequency
- Clear the timer 1 counter, TACNT and TBCNT
- Enable the timer 1 interrupt
- Clear timer 1 interrupt pending conditions

TACON is located in set 1, bank 1, at address E6H, and is read/write addressable using Register addressing mode.

A reset clears TACON to "00H". This sets timer 1 to disable interval timer mode, selects an input clock frequency of fxx/512, and disables timer 1 interrupt. You can clear the timer 1 counter at any time during the normal operation by writing a "1" to TACON.3.

To enable the timer 1 interrupt (IRQ 0, vector F0H), you must write TACON.7, TACON.2, and TACON.1 to "1". To generate the exact time interval, you should write TACON.3 and TACON.0 to "10B", which cleared counter and interrupt pending bit. When the T1INT sub-routine is serviced, the pending condition must be cleared by software by writing a "0" to the timer 1 interrupt pending bit, TACON.0.

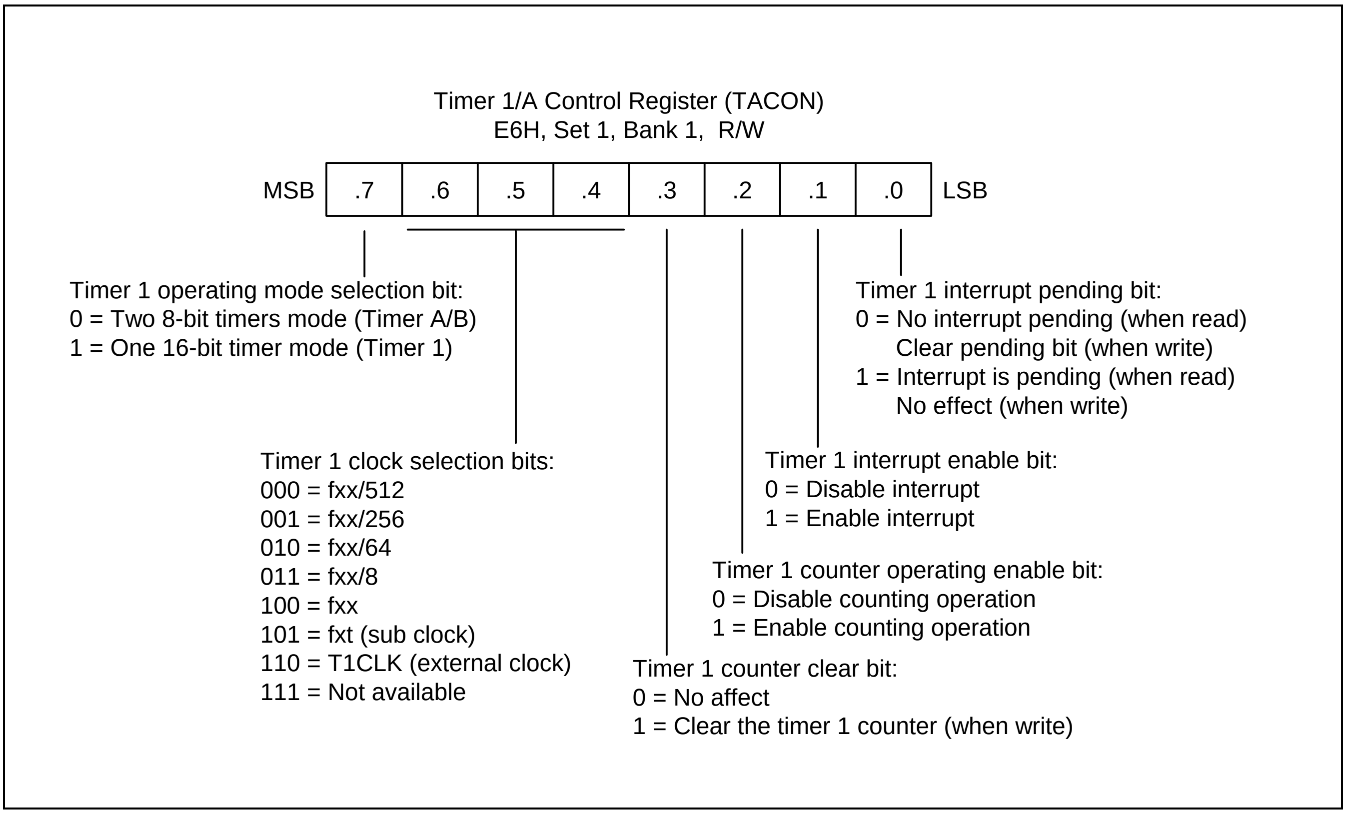
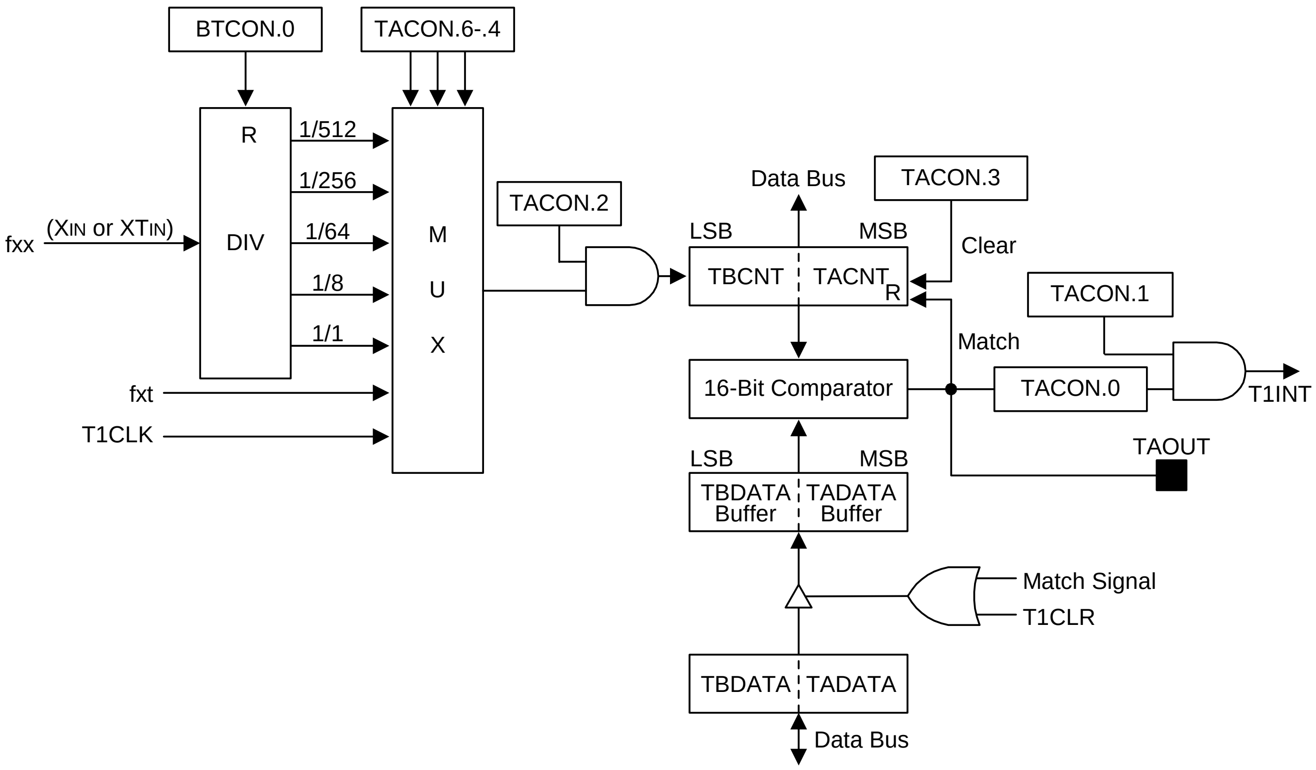


Figure 11-1. Timer 1/A Control Register (TACON)



NOTE: When one 16-bit timer mode (TACON.7 <- "1": Timer 1)

Figure 11-2. Timer 1 Block Diagram (One 16-bit Mode)

TWO 8-BIT TIMERS MODE (TIMER A and B)

OVERVIEW

The 8-bit timer A and B are the 8-bit general-purpose timers. Timer A and B have the interval timer mode by using the appropriate TACON and TBCON setting, respectively.

Timer A and B have the following functional components:

- Clock frequency divider with multiplexer
  - fxx divided by 512, 256, 64, 8 or 1, fxt, and T1CLK (External clock) for timer A
  - fxx divided by 512, 256, 64 or 8 and fxt for timer B
- 8-bit counter (TACNT, TBCNT), 8-bit comparator, and 8-bit reference data register (TADATA, TBDATA)
- Timer A have I/O pin for match output (TAOUT)
- Timer A match interrupt (IRQ 0, vector F0H) generation
- Timer A control register, TACON (set 1, bank 1, E6H, read/write)
- Timer B have I/O pin for match output (TBOUT)
- Timer B match interrupt (IRQ 0, vector F2H) generation
- Timer B control register, TBCON (set 1, bank 1, E7H, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

The timer A and B module can generate an interrupt: the timer A match interrupt (TAINT) and the timer B match interrupt (TBINT). TAINT belongs to the interrupt level IRQ 0, and is assigned a separate vector address, F0H. TBINT belongs to the interrupt level IRQ 0 and is assigned a separate vector address, F2H.

The TAINT and TBINT pending condition should be cleared by software after they are serviced.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the TA or TB reference data registers, TADATA and TBDATA. The match signal generates corresponding match interrupt (TAINT, vector F0H; TBINT, vector F2H) and clears the counter.

If, for example, you write the value 10H to TBDATA, "0" to TACON.7, and 0EH to TBCON, the counter will increment until it reaches 10H. At this point, the TB interrupt request is generated, the counter value is reset, and counting resumes.

Timer A and B Control Register (TACON, TBCON)

You use the timer A and B control register, TACON and TBCON, to

- Enable the timer A (interval timer mode) and B operating (interval timer mode)
- Select the timer A and B input clock frequency
- Clear the timer A and B counter, TACNT and TBCNT
- Enable the timer A and B interrupts
- Clear timer A and B interrupt pending conditions

TACON and TBCON are located in set 1, bank 1, at address E6H and E7H, and is read/write addressable using Register addressing mode.

A reset clears TACON to "00H". This sets timer A to disable interval timer mode, selects an input clock frequency of fxx/512, and disables timer A interrupt. You can clear the timer A counter at any time during normal operation by writing a "1" to TACON.3.

A reset clears TBCON to "00H". This sets timer B to disable interval timer mode, selects an input clock frequency of fxx/512, and disables timer B interrupt. You can clear the timer B counter at any time during normal operation by writing a "1" to TBCON.3.

To enable the timer A interrupt (TAINT) and timer B interrupt (TBINT), you must write TACON.7 to "0", TACON.2 (TBCON.2) and TACON.1 (TBCON.1) to "1". To generate the exact time interval, you should write TACON.3 (TBCON.3) and TACON.0 (TBCON.0), which cleared counter and interrupt pending bit. When the TAINT and TBINT sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the timer A and B interrupt pending bits, TACON.0 or TBCON.0.

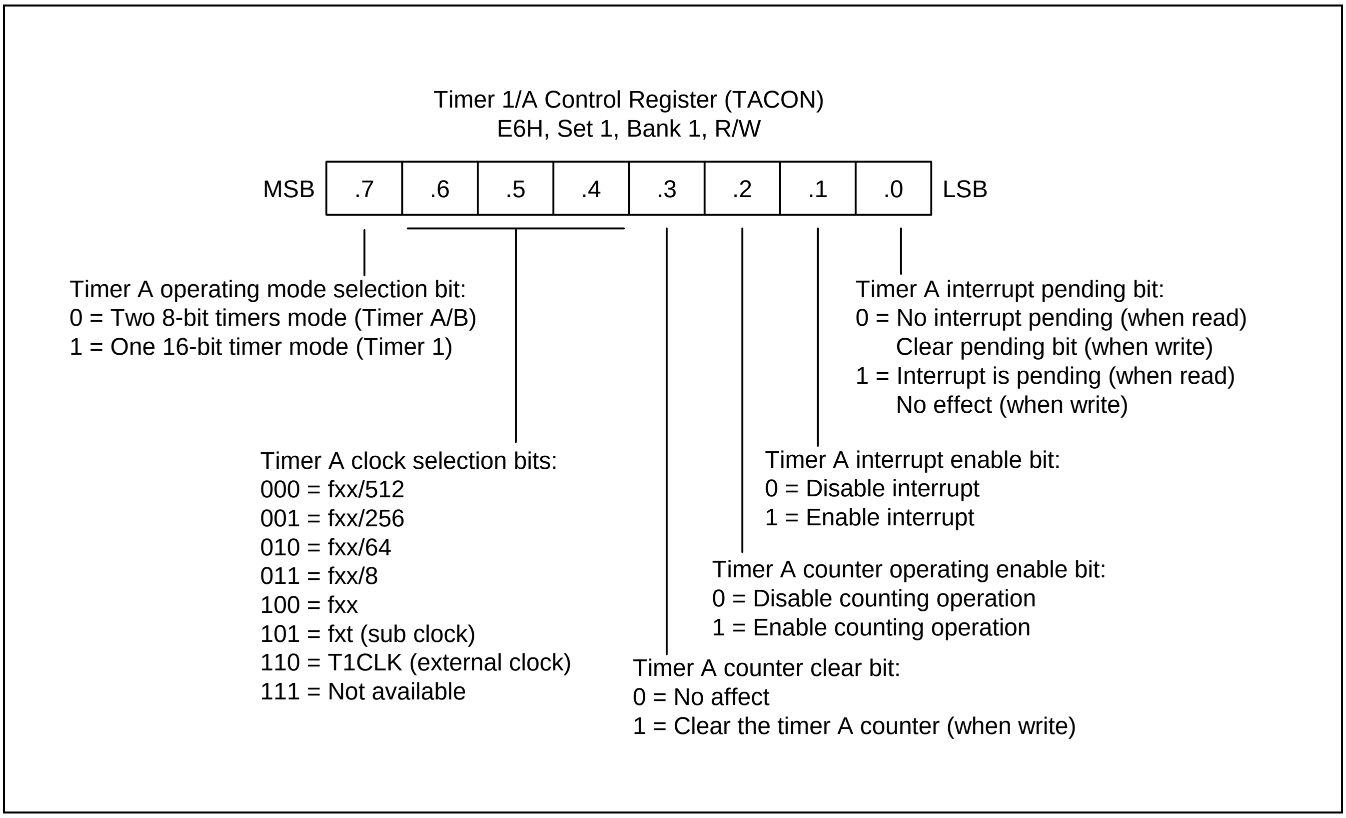


Figure 11-3. Timer 1/A Control Register (TACON)

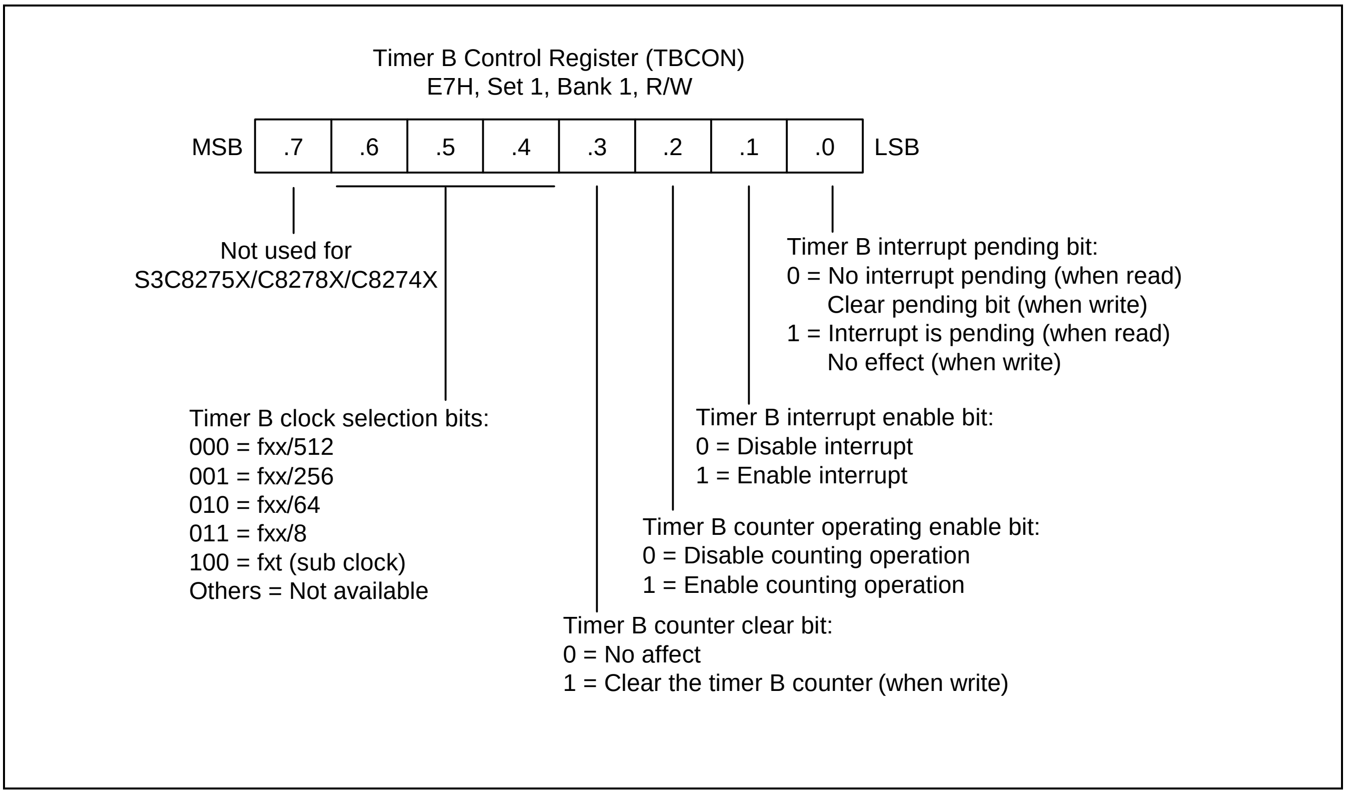
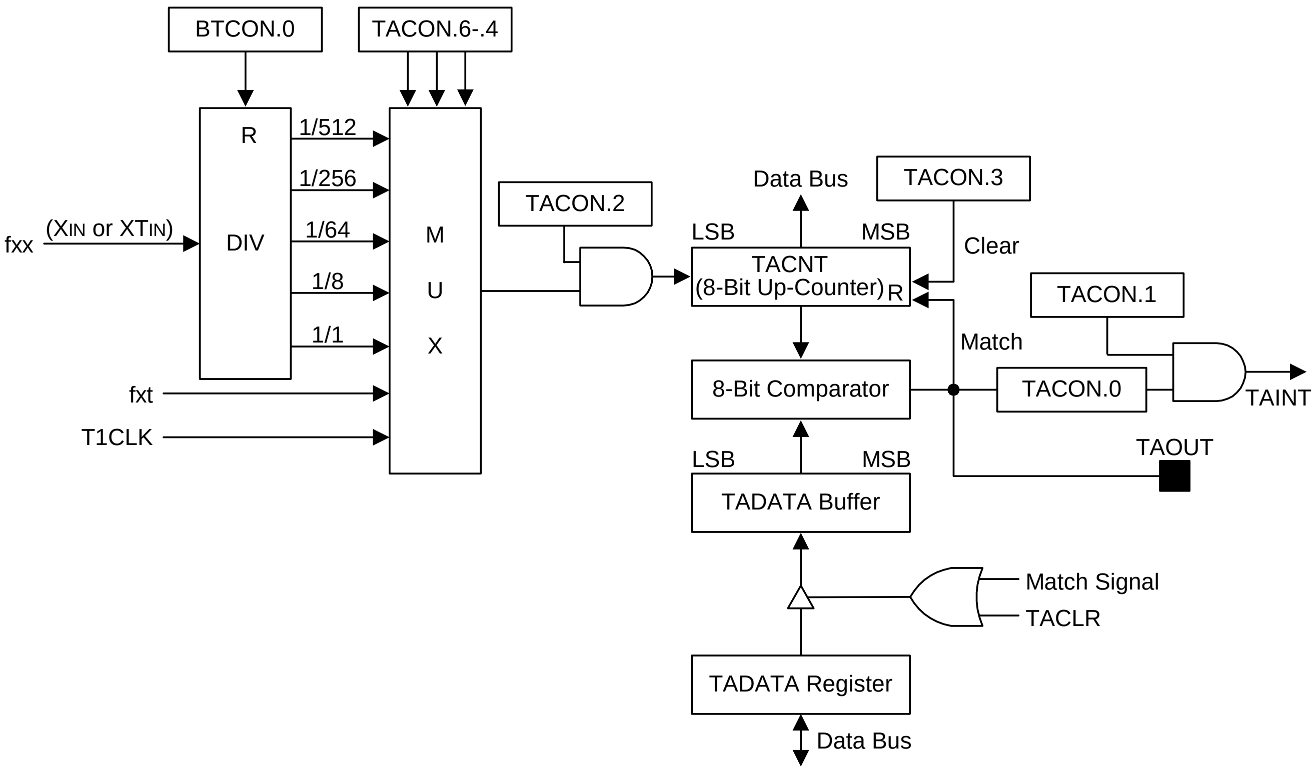


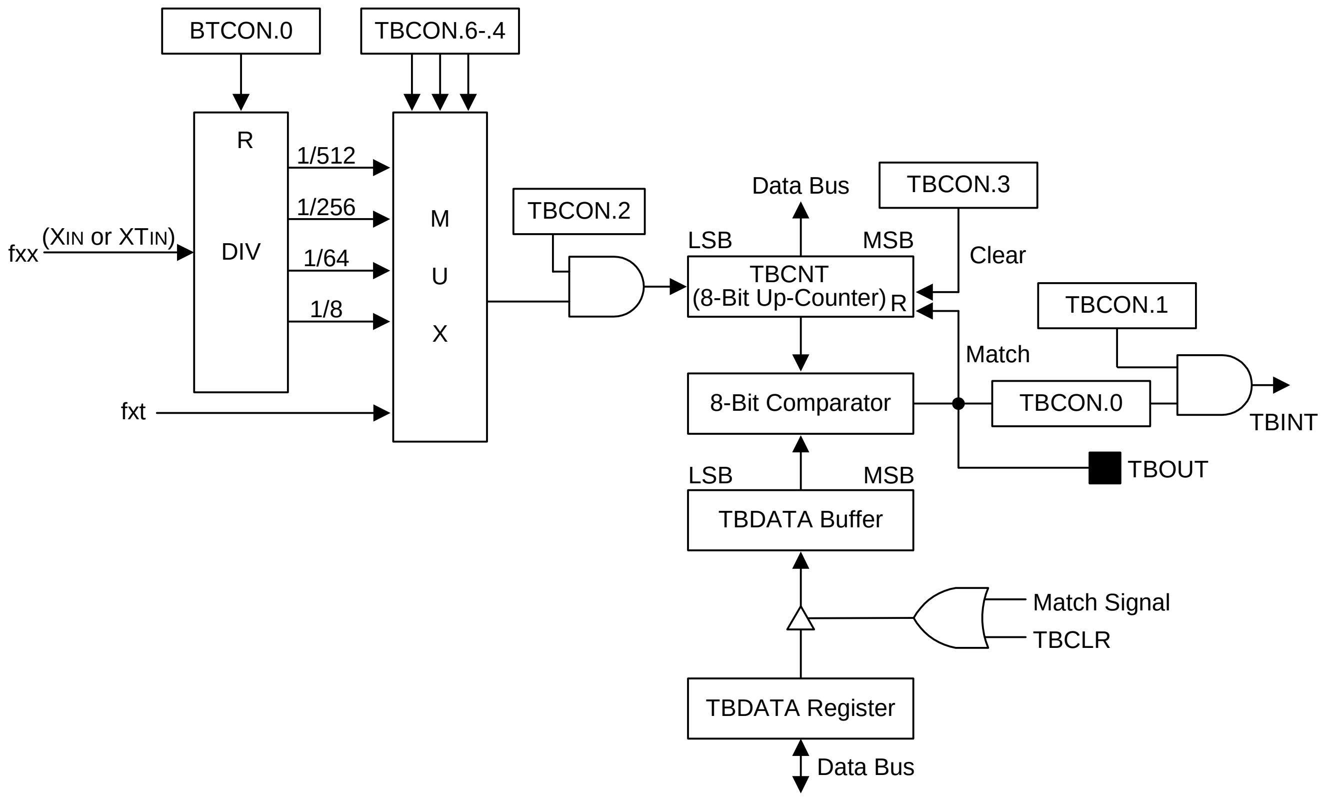
Figure 11-4. Timer B Control Register (TBCON)

SAMSUNG ELECTRONICS



NOTE: When two 8-bit timers mode (TACON.7 <- "0": Timer A)

Figure 11-5. Timer A Block Diagram (Two 8-bit Timers Mode)



NOTE: When two 8-bit timers mode (TACON.7 <- "0": Timer B)

Figure 11-6. Timer B Block Diagram (Two 8-bit Timers Mode)

12 WATCH TIMER

OVERVIEW

Watch timer functions include real-time and watch-time measurement and interval timing for the system clock. To start watch timer operation, set bit 1 of the watch timer control register, WTCON.1 to "1".
And if you want to service watch timer overflow interrupt (IRQ 2, vector F6H), then set the WTCON.6 to "1".
The watch timer overflow interrupt pending condition (WTCON.0) must be cleared by software in the application's interrupt service routine by means of writing a "0" to the WTCON.0 interrupt pending bit.
After the watch timer starts and elapses a time, the watch timer interrupt pending bit (WTCON.0) is automatically set to "1", and interrupt requests commence in 3.91ms, 0.25, 0.5 and 1-second intervals by setting Watch timer speed selection bits (WTCON.3– .2).

The watch timer can generate a steady 0.5 kHz, 1 kHz, 2 kHz, or 4 kHz signal to BUZ output pin for Buzzer. By setting WTCON.3 and WTCON.2 to "11b", the watch timer will function in high-speed mode, generating an interrupt every 3.91 ms. High-speed mode is useful for timing events for program debugging sequences.

Also, you can select watch timer clock source by setting the WTCON.7 appropriately value.

The watch timer supplies the clock frequency for the LCD controller (f_{LCD}). Therefore, if the watch timer is disabled, the LCD controller does not operate.

Watch timer has the following functional components:

- Real Time and Watch-Time Measurement
- Using a Main or Sub Clock Source (Main clock divided by 2^7(fx/128) or Sub clock(fxt))
- Clock Source Generation for LCD Controller (f_{LCD})
- I/O pin for Buzzer Output Frequency Generator (P0.7, BUZ)
- Timing Tests in High-Speed Mode
- Watch timer overflow interrupt (IRQ 2, vector F6H) generation
- Watch timer control register, WTCON (set 1, bank 1, E1H, read/write)

WATCH TIMER CONTROL REGISTER (WTCON)

The watch timer control register, WTCON is used to select the input clock source, the watch timer interrupt time and Buzzer signal, to enable or disable the watch timer function. It is located in set 1, bank 1 at address E1H, and is read/write addressable using Register addressing mode.
A reset clears WTCON to "00H". This disable the watch timer and select fx/128 as the watch timer clock.
So, if you want to use the watch timer, you must write appropriate value to WTCON.

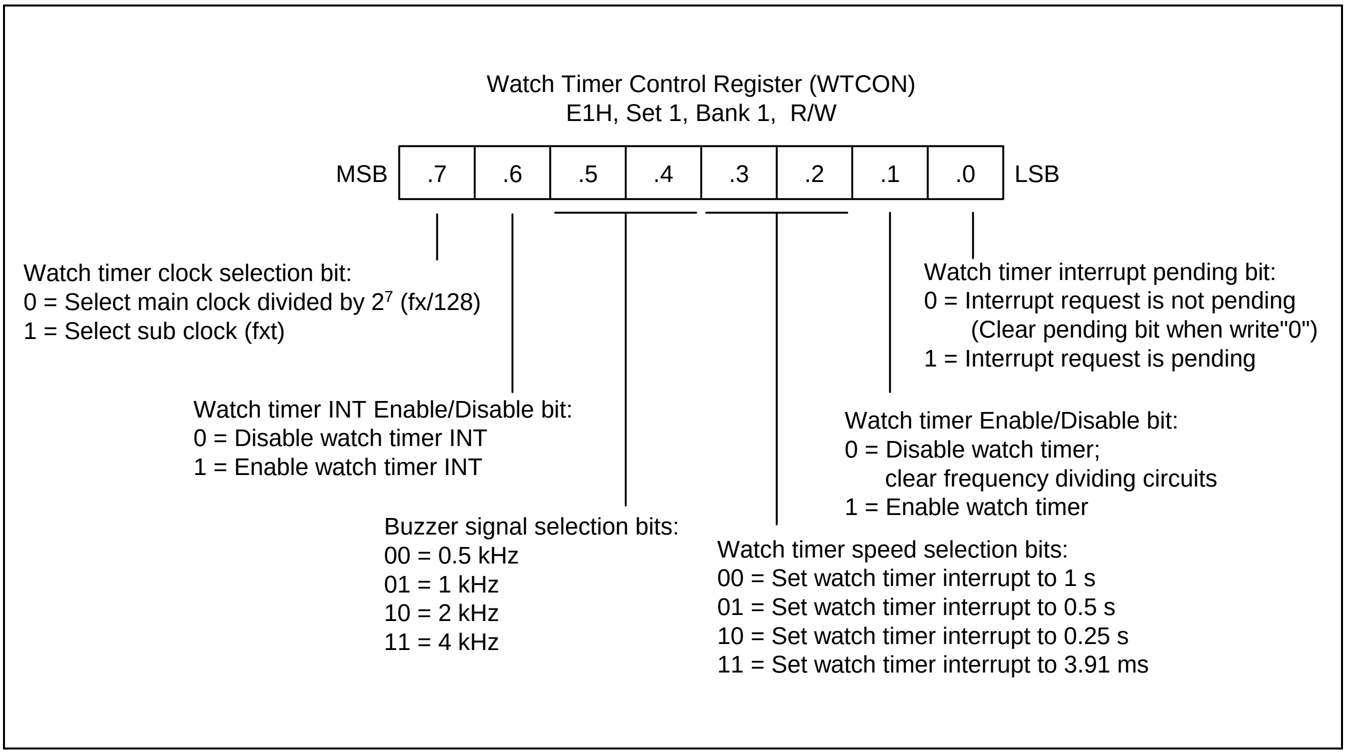


Figure 12-1. Watch Timer Control Register (WTCON)

SAMSUNG ELECTRONICS

WATCH TIMER CIRCUIT DIASGRAM

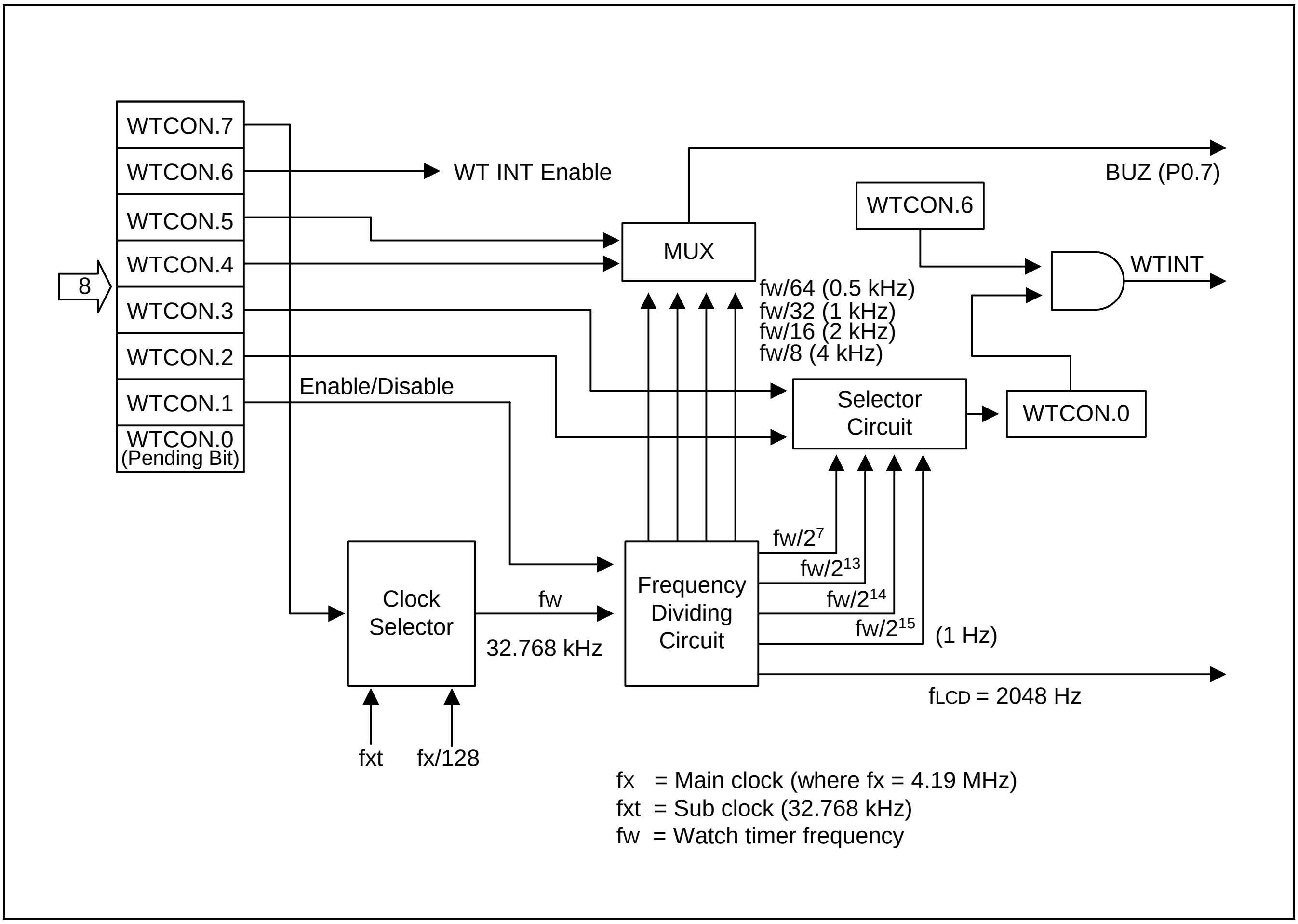


Figure 12-2. Watch Timer Circuit Diagram

13 LCD CONTROLLER/DRIVER

OVERVIEW

The S3C8275X/C8278X/C8274X microcontroller can directly drive an up-to-128-dot (32 segments x 4 commons) LCD panel. Its LCD block has the following components:

- LCD controller/driver
- Display RAM (00H–0FH of page 2) for storing display data
- 32 segment output pins (SEG0–SEG31)
- 4 common output pins (COM0–COM3)
- Three LCD operating power supply pins (V_{LC0}–V_{LC2})
- LCD bias by Internal/External register

The LCD control register, LCON, is used to turn the LCD display on or off, to select LCD clock frequency, to select bias and duty, and switch the current to the dividing resistor for the LCD display. Data written to the LCD display RAM can be automatically transferred to the segment signal pins without program control.

When a sub clock is selected as the LCD clock source, the LCD display is enabled even in main clock stop or idle mode.

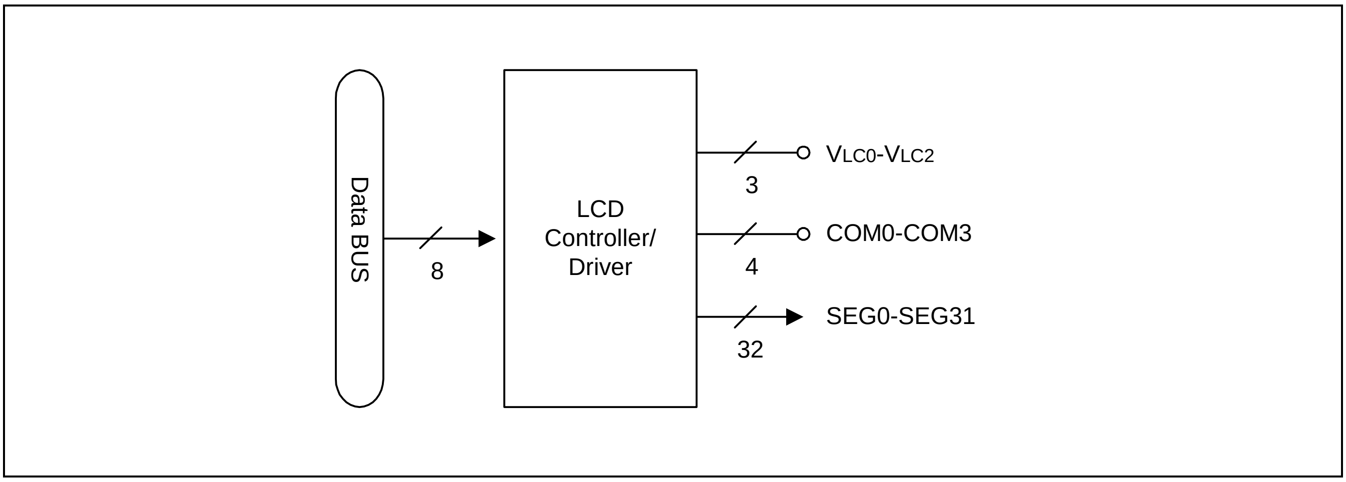


Figure 13-1. LCD Function Diagram

LCD CIRCUIT DIAGRAM

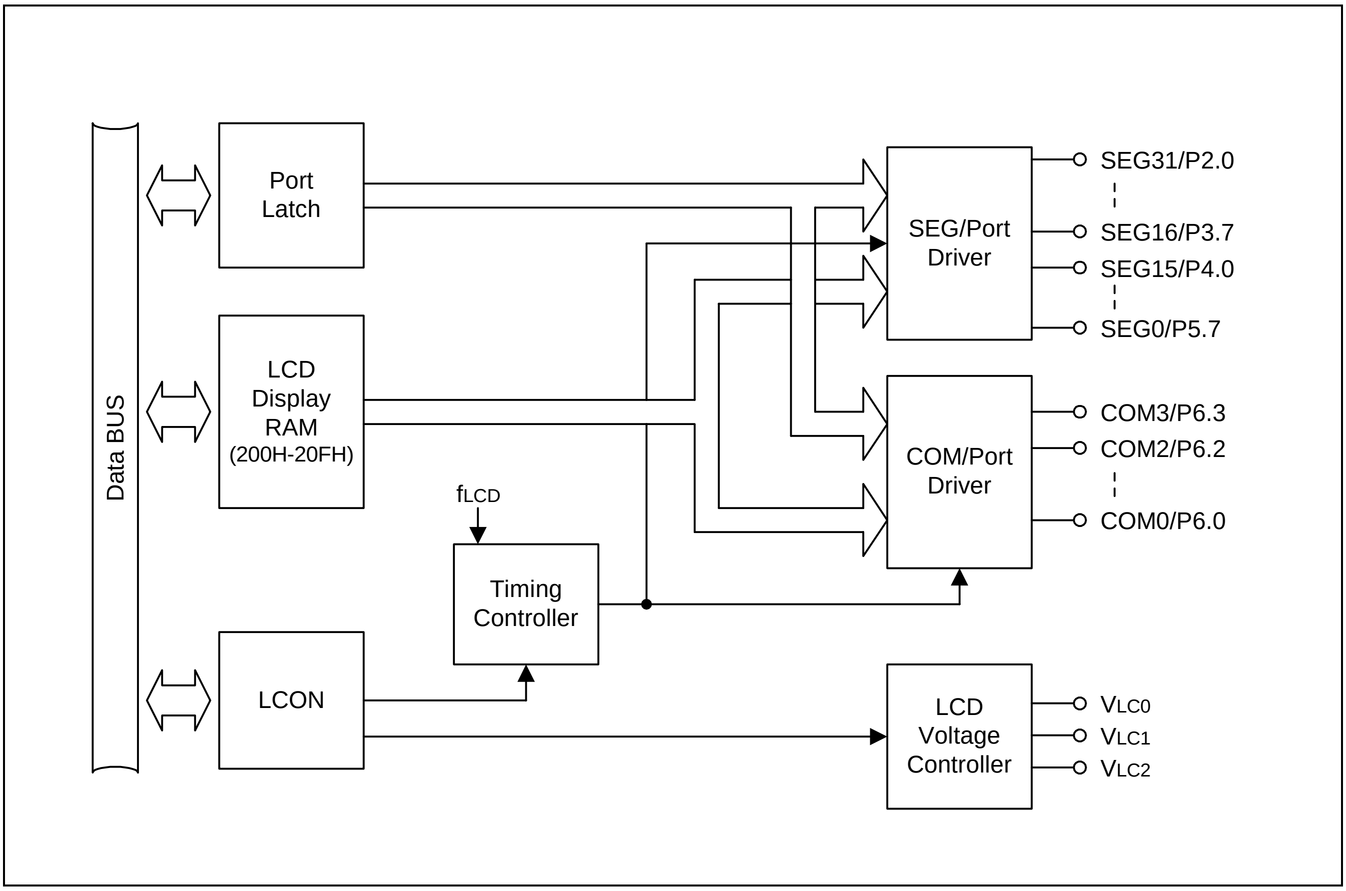


Figure 13-2. LCD Circuit Diagram

SAMSUNG ELECTRONICS

LCD RAM ADDRESS AREA

RAM addresses of page 2 are used as LCD data memory. When the bit value of a display segment is "1", the LCD display is turned on; when the bit value is "0", the display is turned off.

Display RAM data are sent out through segment pins SEG0–SEG31 using a direct memory access (DMA) method that is synchronized with the f_{LCD} signal. RAM addresses in this location that are not used for LCD display can be allocated to general-purpose use.

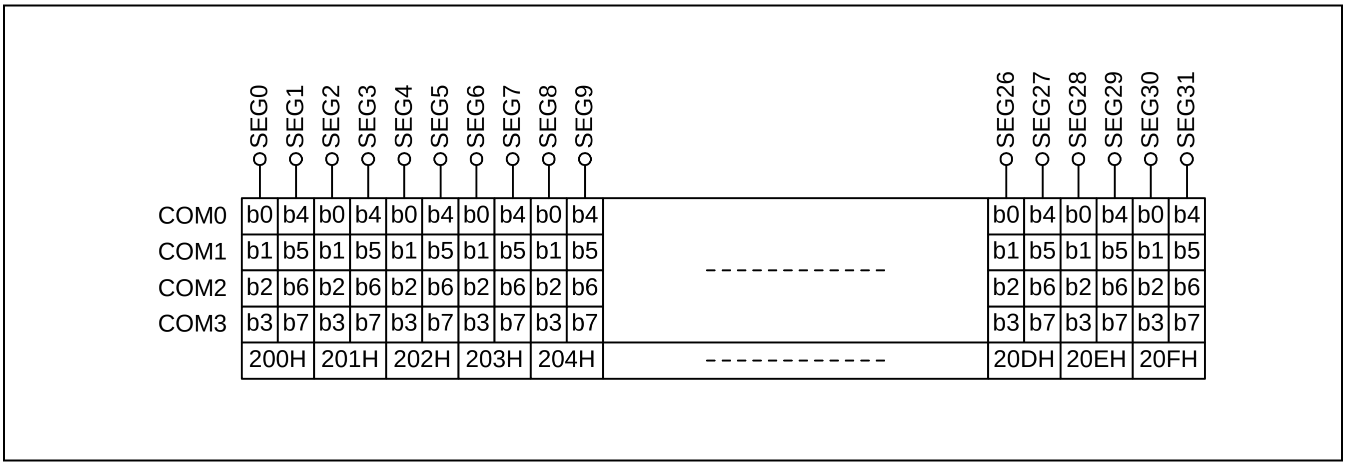


Figure 13-3. LCD Display Data RAM Organization

Table 13-1. LCD Clock Signal Frame Frequency

LCDCK Frequency (f_{LCD})	Static	1/2 Duty	1/3 Duty	1/4 Duty
64 Hz	64	32	21	16
128 Hz	128	64	43	32
256Hz	256	128	85	64
512 Hz	512	256	171	128

LCD CONTROL REGISTER (LCON)

A LCON is located in set 1, bank 1, at address E0H, and is read/write addressable using Register addressing mode. It has the following control functions.

- LCD duty and bias selection
- LCD clock selection
- LCD display control
- Internal/External LCD dividing resistors selection

The LCON register is used to turn the LCD display on/off, to select duty and bias, to select LCD clock and control the flow of the current to the dividing in the LCD circuit. Following a RESET, all LCON values are cleared to "0". This turns off the LCD display, select 1/4 duty and 1/3 bias, select 64Hz for LCD clock, and Enable internal LCD dividing resistors.

The LCD clock signal determines the frequency of COM signal scanning of each segment output. This is also referred as the LCD frame frequency. Since the LCD clock is generated by watch timer clock (fw). The watch timer should be enabled when the LCD display is turned on.

LCD Control Register (LCON)
E0H, Set 1, Bank 1, R/W

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

.7 — Internal LCD dividing register enable bit:
0 = Enable internal LCD dividing resistors
1 = Disable internal LCD dividing resistors

.6–.5 — LCD clock selection bits:
00 = fw/2^9 (64 Hz)
01 = fw/2^8 (128 Hz)
10 = fw/2^7 (256 Hz)
11 = fw/2^6 (512 Hz)

.4–.2 — LCD duty and bias selection bits:
000 = 1/4 duty, 1/3 bias
001 = 1/3 duty, 1/3 bias
010 = 1/3 duty, 1/2 bias
011 = 1/2 duty, 1/2 bias
1xx = Static

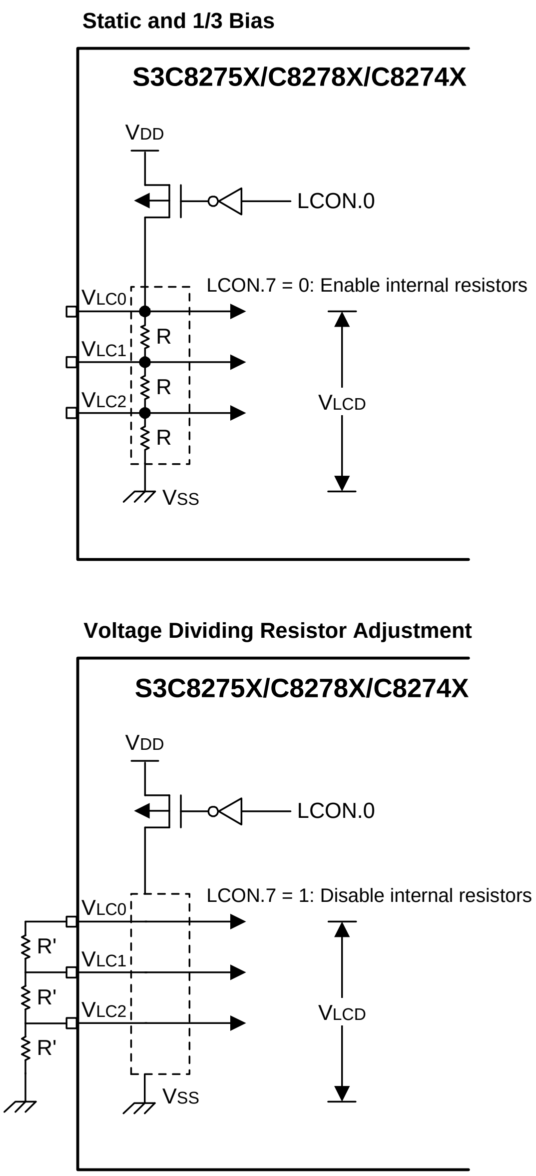
.1 — Not used for S3C8275X/C8278X/C8274X

.0 — LCD display control bit:
0 = Turn display off
(Turn off the P-Tr)
1 = Turn display on
(Turn on the P-Tr)

NOTES:
1. "x" means don't care.
2. When 1/2 bias is selected, the bias levels are set as V_{LC0}, V_{LC1}(V_{LC2}), and V_{SS}.

Figure 13-4. LCD Control Register (LCON)

LCD VOLTAGE DIVIDING RESISTOR



NOTES:
1. R = Internal LCD dividing resistors. The resistors can be disconnected by LCON.7.
2. R' = External LCD dividing resistors.

Figure 13-5. Internal Voltage Dividing Resistor Connection

COMMON (COM) SIGNALS

The common signal output pin selection (COM pin selection) varies according to the selected duty cycle.

- In 1/4 duty mode, COM0-COM3 pins are selected
- In 1/3 duty mode, COM0-COM2 pins are selected
- In 1/2 duty mode, COM0-COM1 pins are selected

SEGMENT (SEG) SIGNALS

The 32 LCD segment signal pins are connected to corresponding display RAM locations at page 2. Bits of the display RAM are synchronized with the common signal output pins.

When the bit value of a display RAM location is "1", a select signal is sent to the corresponding segment pin. When the display bit is "0", a 'no-select' signal to the corresponding segment pin.

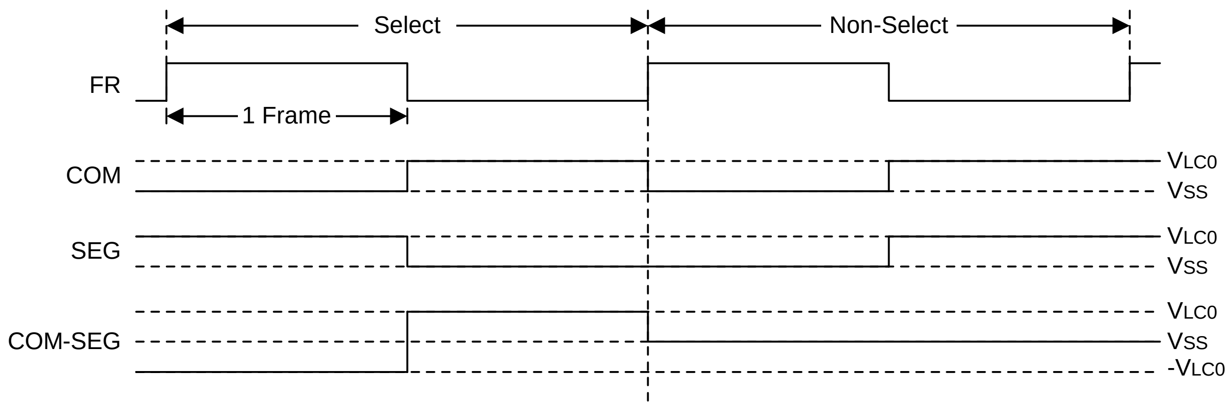


Figure 13-6. Select/No-Select Signals in Static Display Mode

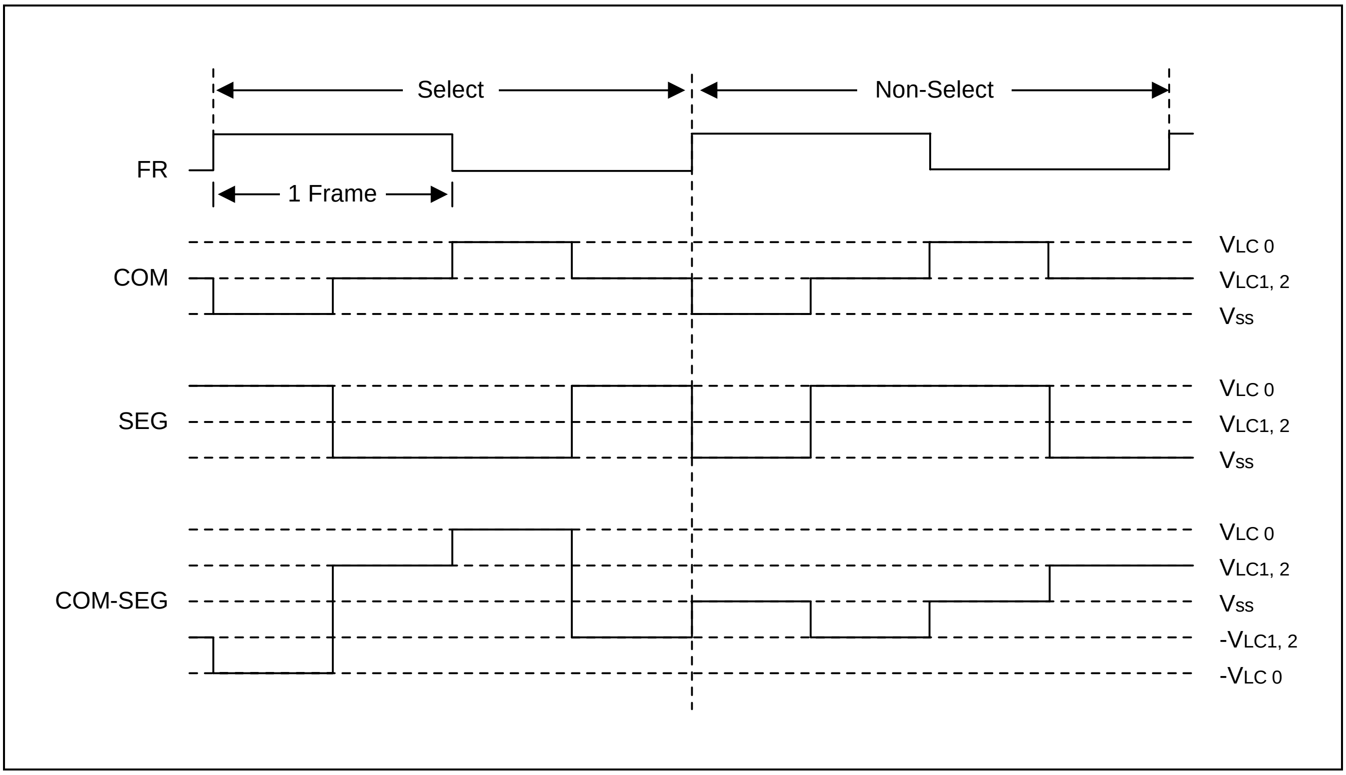


Figure 13-7. Select/No-Select Signal in 1/2 Duty, 1/2 Bias Display Mode

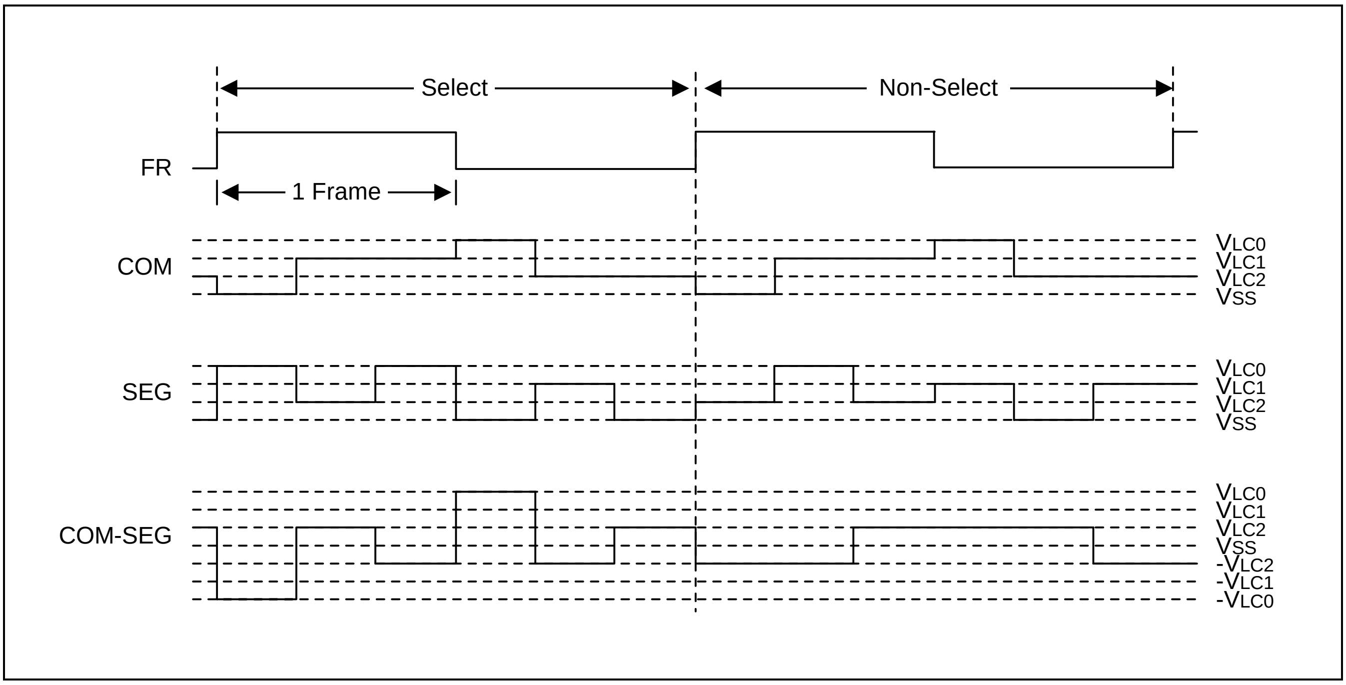


Figure 13-8. Select/No-Select Signal in 1/3 Duty, 1/3 Bias Display Mode

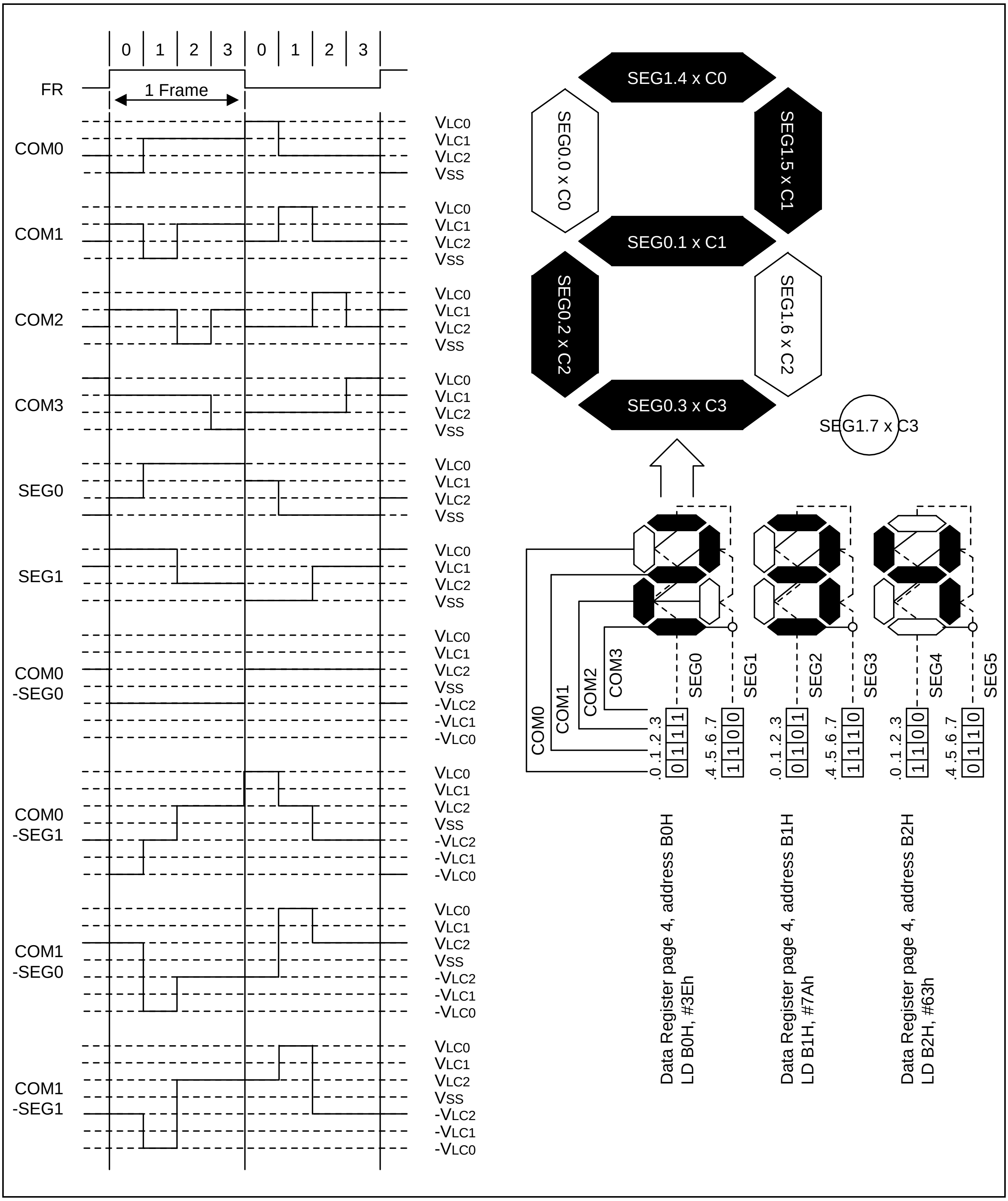


Figure 13-9. LCD Signals and Wave Forms Example in 1/4 Duty, 1/3 Bias Display Mode

SAMSUNG ELECTRONICS

14 SERIAL I/O INTERFACE

OVERVIEW

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

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

The SIO module can transmit or receive 8-bit serial data at a frequency determined by its corresponding control register settings. To ensure flexible data transmission rates, you can select an internal or external clock source.

PROGRAMMING PROCEDURE

To program the SIO module, follow these basic steps:

1. Configure the I/O pins at port (SCK/SI/SO) by loading the appropriate value to the P1CONL register if necessary.
2. Load an 8-bit value to the SIOCON control register to properly configure the serial I/O module. In this operation, SIOCON.2 must be set to "1" to enable the data shifter.
3. For interrupt generation, set the serial I/O interrupt enable bit (SIOCON) to "1".
4. When you transmit data to the serial buffer, write data to SIODATA and set SIOCON.3 to 1, the shift operation starts.
5. When the shift operation (transmit/receive) is completed, the SIO pending bit (SIOCON.0) are set to "1" and SIO interrupt request is generated.

SIO CONTROL REGISTERS (SIOCON)

The control register for serial I/O interface module, SIOCON, is located at E1H in set 1, bank 0. It has the control setting for SIO module.

- Clock source selection (internal or external) for shift clock
- Interrupt enable
- Edge selection for shift operation
- Clear 3-bit counter and start shift operation
- Shift operation (transmit) enable
- Mode selection (transmit/receive or receive-only)
- Data direction selection (MSB first or LSB first)

A reset clears the SIOCON value to "00H". This configures the corresponding module with an internal clock source at the SCK, selects receive-only operating mode, and clears the 3-bit counter. The data shift operation and the interrupt are disabled. The selected data direction is MSB-first.

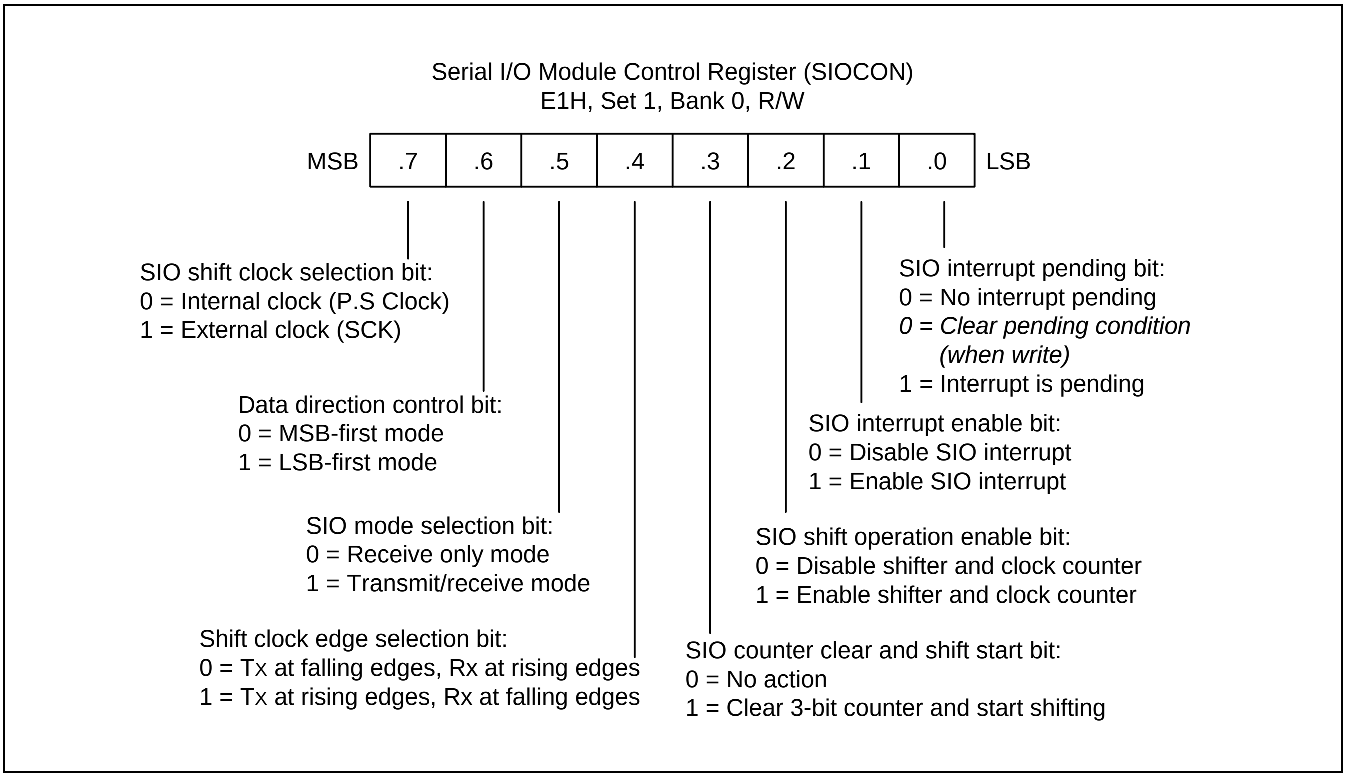


Figure 14-1. Serial I/O Module Control Register (SIOCON)

SIO PRE-SCALER REGISTER (SIOPS)

The prescaler register for serial I/O interface module, SIOPS, is located at E3H in set 1, bank 0.
The value stored in the SIO pre-scaler register, SIOPS, lets you determine the SIO clock rate (baud rate) as follows:

Baud rate = Input clock (fxx/4)/(Prescaler value + 1), or SCK input clock.

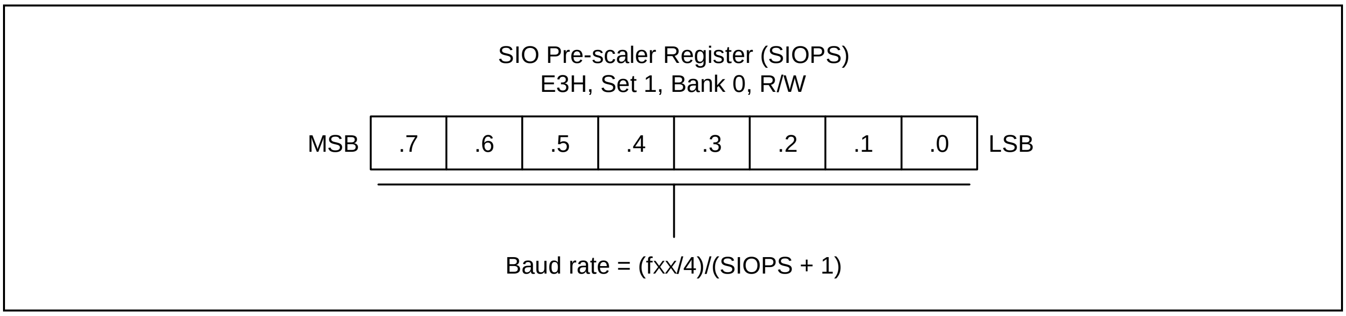


Figure 14-2. SIO Prescaler Register (SIOPS)

SIO BLOCK DIAGRAM

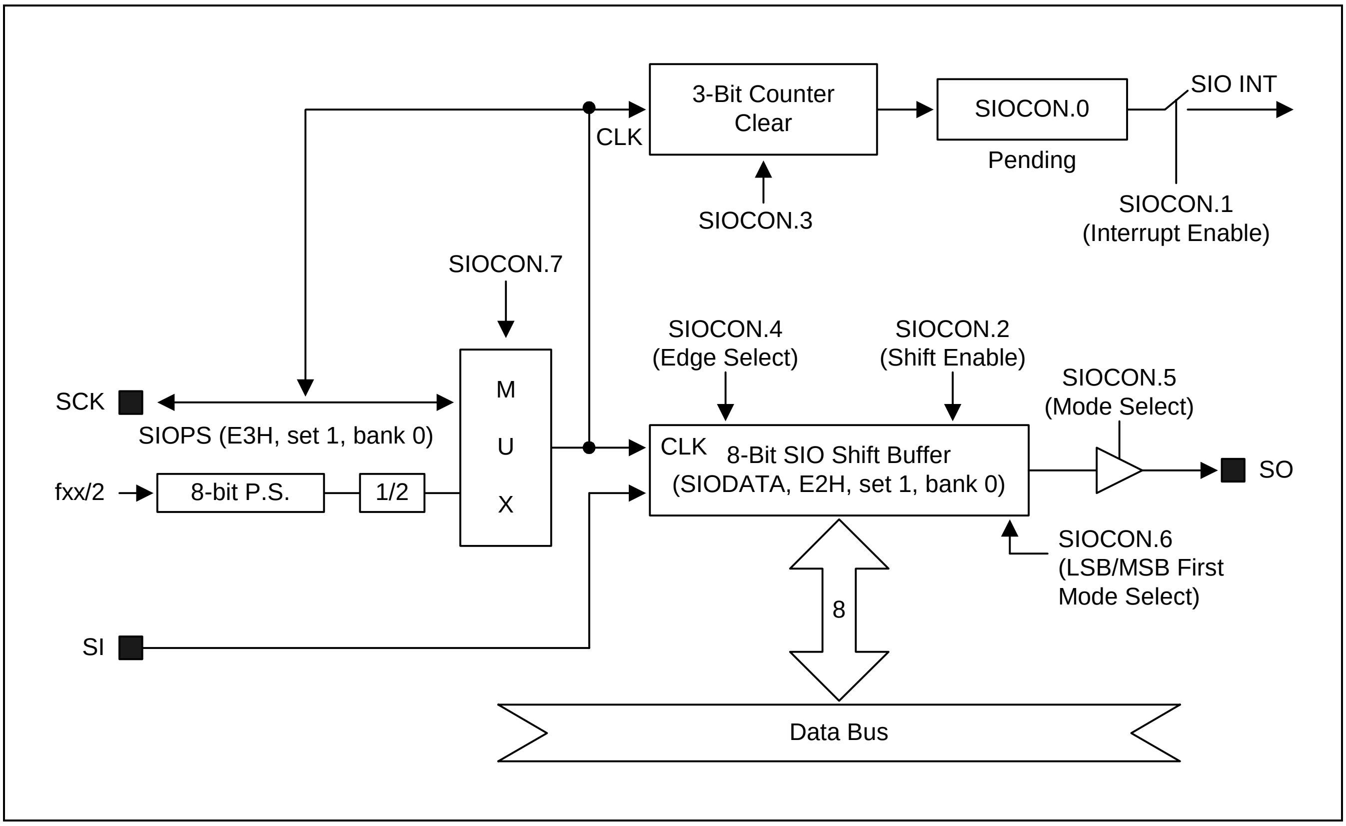


Figure 14-3. SIO Functional Block Diagram

SERIAL I/O TIMING DIAGRAM (SIO)

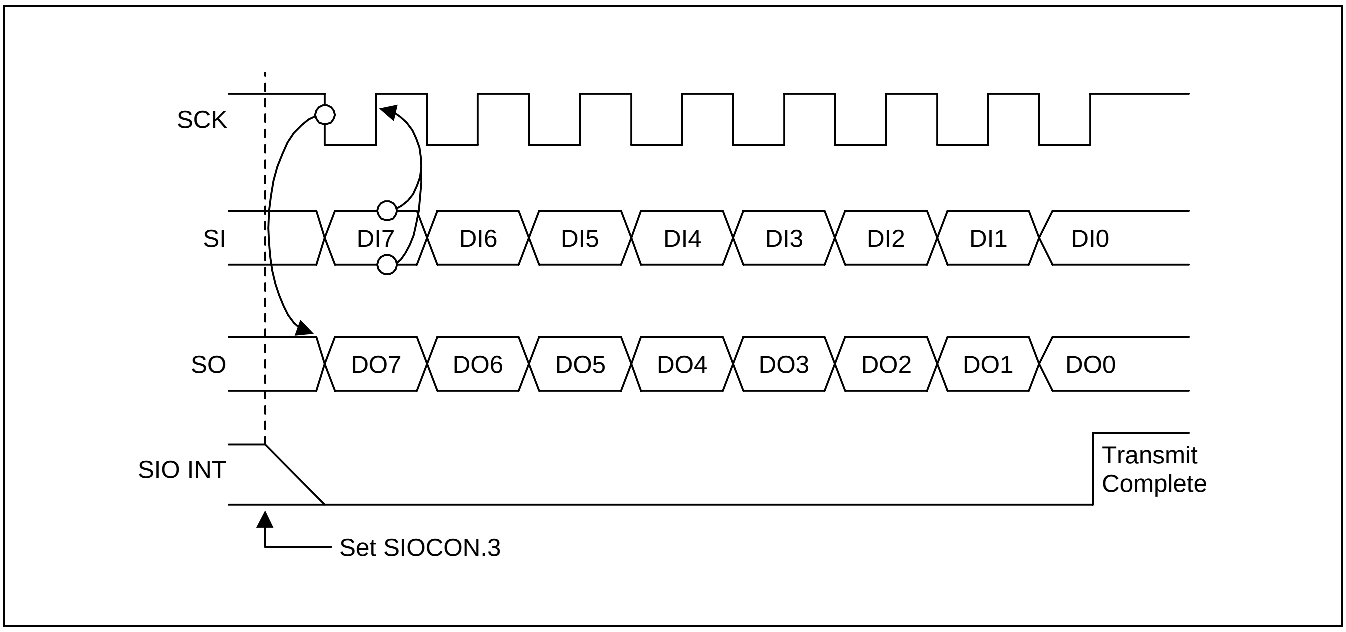


Figure 14-4. Serial I/O Timing in Transmit/Receive Mode (Tx at falling, SIOCON.4 = 0)

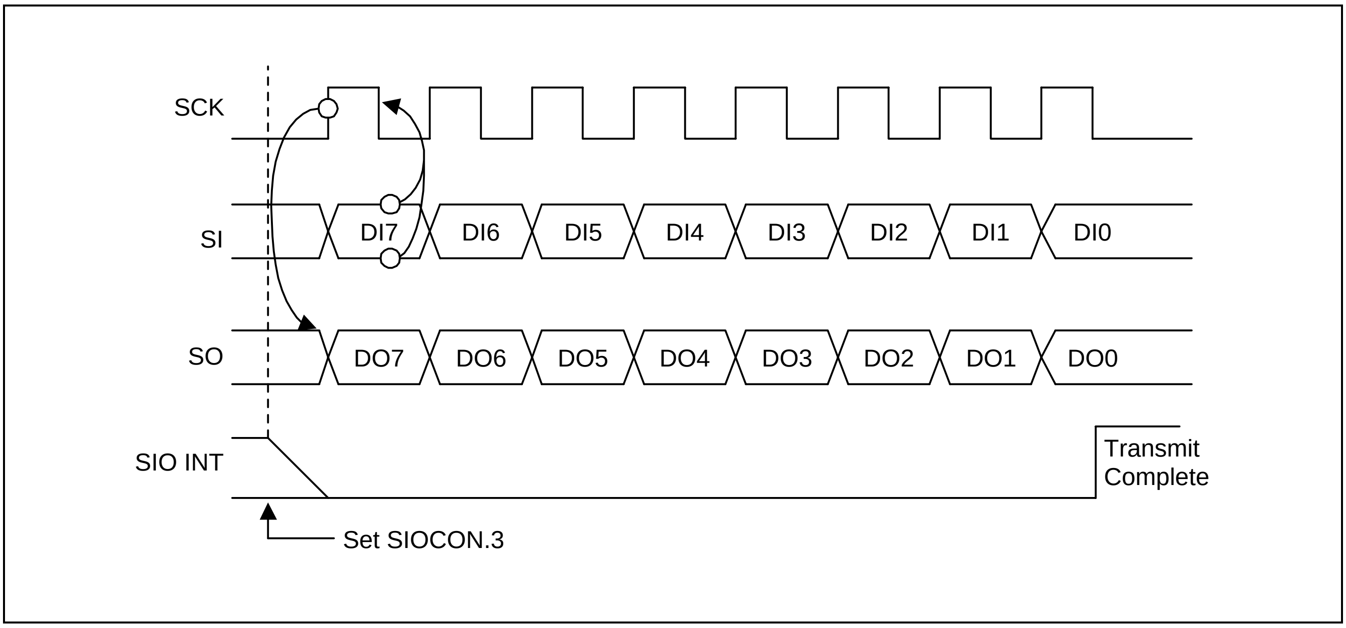


Figure 14-5. Serial I/O Timing in Transmit/Receive Mode (Tx at rising, SIOCON.4 = 1)

15 BATTERY LEVEL DETECTOR

OVERVIEW

The S3C8275X/C8278X/C8274X micro-controller has a built-in BLD (Battery Level Detector) circuit which allows detection of power voltage drop or external input level through software. Turning the BLD operation on and off can be controlled by software. Because the IC consumes a large amount of current during BLD operation. It is recommended that the BLD operation should be kept OFF unless it is necessary. Also the BLD criteria voltage can be set by the software. The criteria voltage can be set by matching to one of the 3 kinds of voltage below that can be used.

2.2 V, 2.4 V or 2.8 V (V_{DD} reference voltage), or external input level (External reference voltage)

The BLD block works only when BLDCON.3 is set. If V_{DD} level is lower than the reference voltage selected with BLDCON.2–.0, BLDCON.4 will be set. If V_{DD} level is higher, BLDCON.4 will be cleared. When users need to minimize current consumption, do not operate the BLD block.

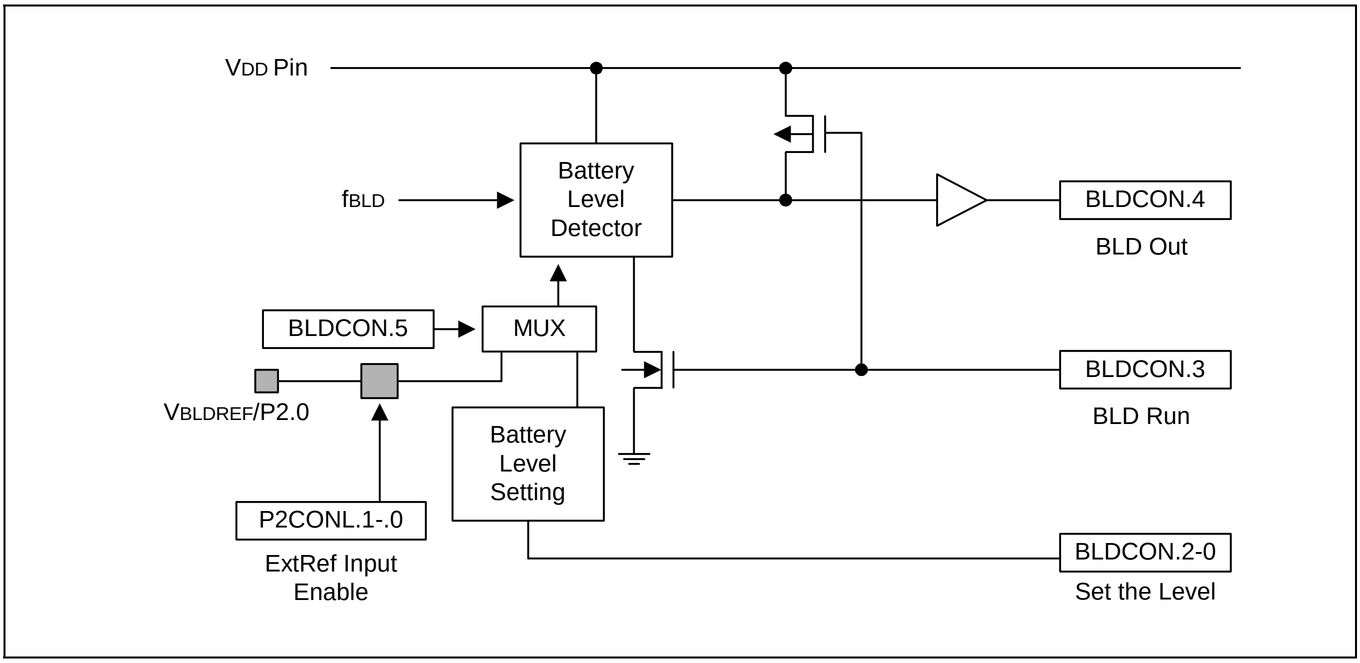
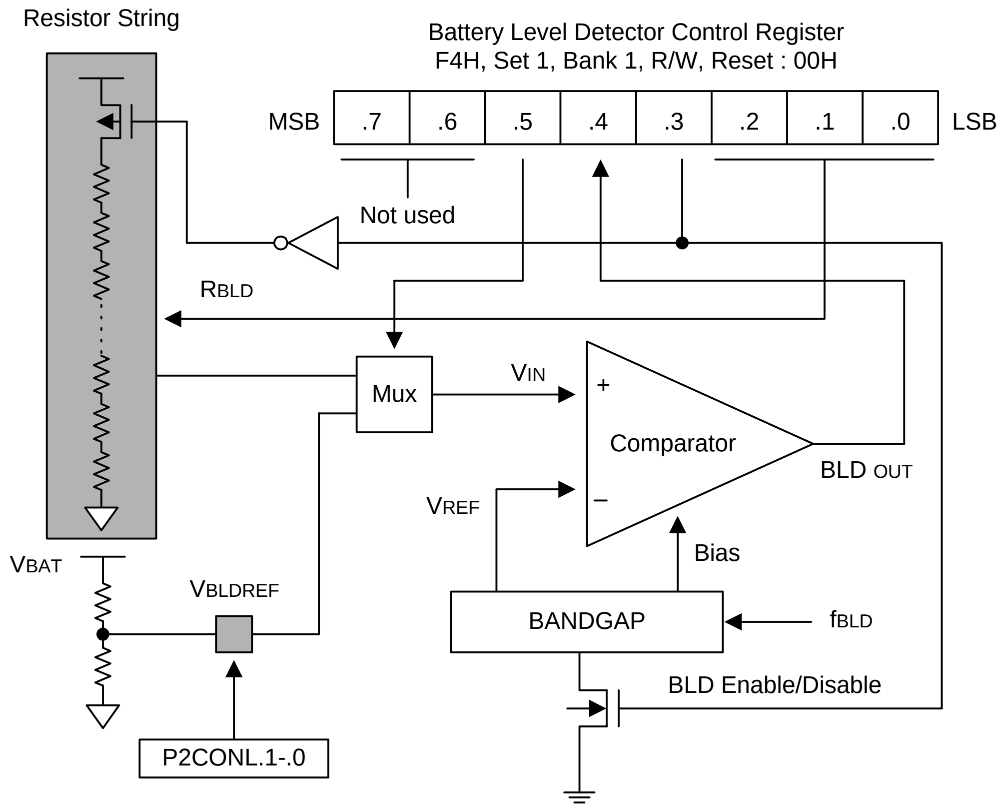


Figure 15-1. Block Diagram for Voltage Level Detect

BATTERY LEVEL DETECTOR CONTROL REGISTER (BLDCON)

The bit 3 of BLDCON controls to run or disable the operation of Battery Level Detector. Basically this V_{BLD} is set as 2.2V by system reset and it can be changed in 3 kinds voltages by selecting Battery Level Detector Control Register (BLDCON). When you write 3-bit data value to BLDCON, an established resistor string is selected and the V_{BLD} is fixed in accordance with this resistor. Table 15-1 shows specific V_{BLD} of 3 levels.



NOTES:
1. The reset value of BLDCON is #00H.
2. VREF is about 1 volt.

Figure 15-2. Battery Level Detect Circuit and Control Register

Table 15-1. BLDCON Value and Detection Level

BLDCON .2–.0	V_{BLD}
0 0 0	2.2 V
1 0 1	2.4 V
0 1 1	2.8 V
Other values	Not available

16 EMBEDDED FLASH MEMORY INTERFACE

OVERVIEW

This chapter is only for the S3F8275X. The S3F8275X has an on-chip full-flash memory internally instead of masked ROM. The flash memory is accessed by "LDC" instruction and the type of sector erase and a byte programmable flash, a user can program the data in the flash memory area any time you want. The S3F8275X's embedded 16K-byte memory has two operating features:

- User program mode: S3F8275X only
- Tool program mode: Refer to the chapter 19. S3F8275X/F8278X/F8274X FLASH MCU.

USER PROGRAM MODE

This mode supports sector erase, byte programming, byte read and one protection mode (Hard lock protection). The read protection mode is available only in Tool Program mode. So in order to make a chip into read protection, you need to select a read protection option when you program an initial your code to a chip by using Tool Program mode by using a programming tool.

The S3F8275X has the pumping circuit internally. Therefore, 12.5V into VPP (test) pin is not needed. To program a flash memory in this mode several control registers will be used. There are four kind functions– programming, reading, sector erase, hard lock protection.

FLASH MEMORY CONTROL REGISTERS (USER PROGRAM MODE)

Flash Memory Control Register

FMCON register is available only in user program mode to select the Flash Memory operation mode; sector erase, byte programming, and to make the flash memory into a hard lock protection.

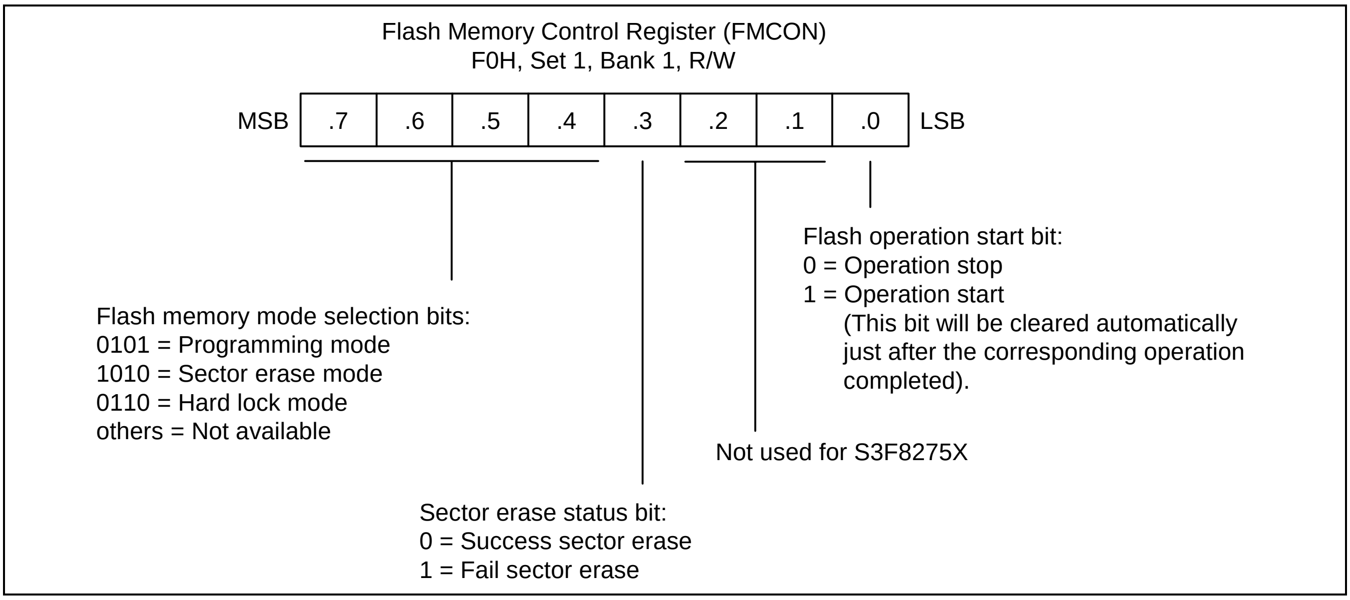


Figure 16-1. Flash Memory Control Register (FMCON)

The bit 0 of FMCON register (FMCON.0) is a start bit for Erase and Hard Lock operation mode. Therefore, operation of Erase and Hard Lock mode is activated when you set FMCON.0 to "1". Also you should wait a time of Erase (Sector erase) or Hard lock to complete it's operation before a byte programming or a byte read of same sector area by using "LDC" instruction. When you read or program a byte data from or into flash memory, this bit is not needed to manipulate.

The sector erase status bit is read only. If an interrupt is requested during the operation of "Sector erase", the operation of "Sector Erase" is discontinued, and the interrupt is served by CPU. Therefore, the sector erase status bit should be checked after executing "Sector Erase". The "Sector Erase" operation is success if the bit is logic "0", and is failure if the bit is logic "1".

NOTE: When the ID code, "A5H", is written to the FMUSR register. A mode of sector erase, user program, and hard lock may be executed unfortunately. So, It should be careful of the above situation.

Flash Memory User Programming Enable Register

The FMUSR register is used for a safety operation of the flash memory. This register will protect undesired erase or program operation from malfunctioning of CPU caused by an electrical noise. After reset, the user-programming mode is disabled, because the value of FMUSR is "00000000B" by reset operation. If necessary to operate the flash memory, you can use the user programming mode by setting the value of FMUSR to "10100101B". The other value of "10100101b", user program mode is disabled.

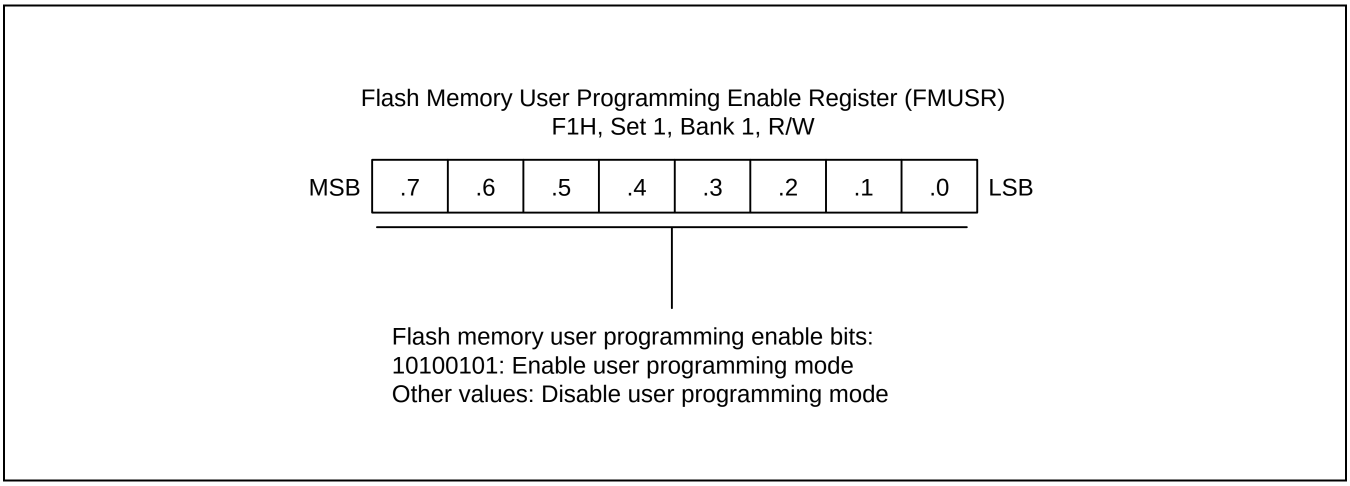


Figure 16-2. Flash Memory User Programming Enable Register (FMUSR)

Flash Memory Sector Address Registers

There are two sector address registers for addressing a sector to be erased. The FMSECL (Flash Memory Sector Address Register Low Byte) indicates the low byte of sector address and FMSECH (Flash Memory Sector Address Register High Byte) indicates the high byte of sector address.

The FMSECH is needed for S3F8275X because it has 128 sectors, respectively. One sector consist of 128-bytes. Each sector's address starts XX00H or XX80H, that is, a base address of sector is XX00H or XX80H. So FMSECL register 6-0 don't mean whether the value is ‘1' or '0'. We recommend that the simplest way is to load sector base address into FMSECH and FMSECL register.

When programming the flash memory, you should write data after loading sector base address located in the target address to write data into FMSECH and FMSECL register. If the next operation is also to write data, you should check whether next address is located in the same sector or not. It case of other sectors, you must load sector address to FMSECH and FMSECL register according to the sector.

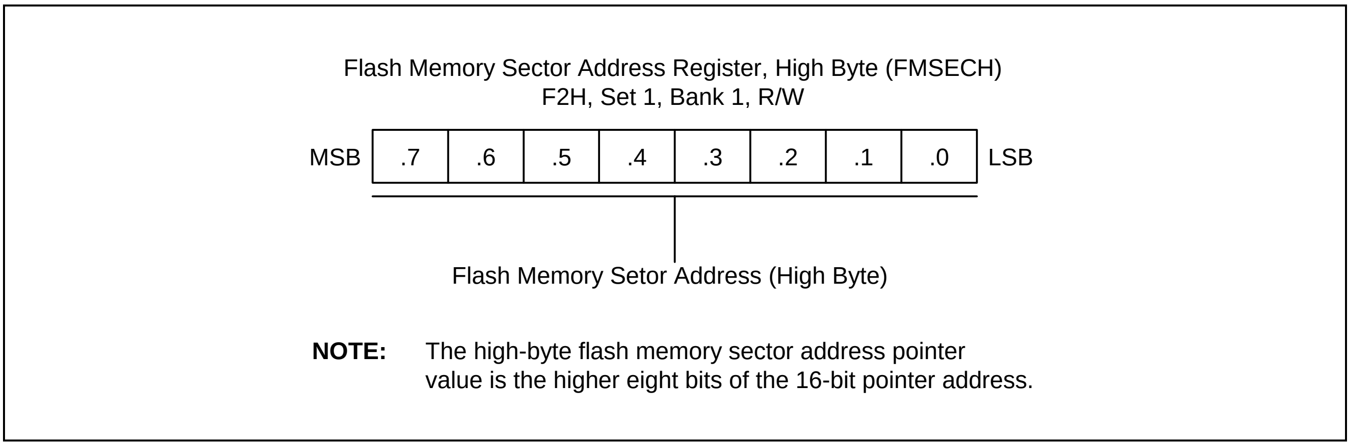


Figure 16-3. Flash Memory Sector Address Register, High Byte (FMSECH)

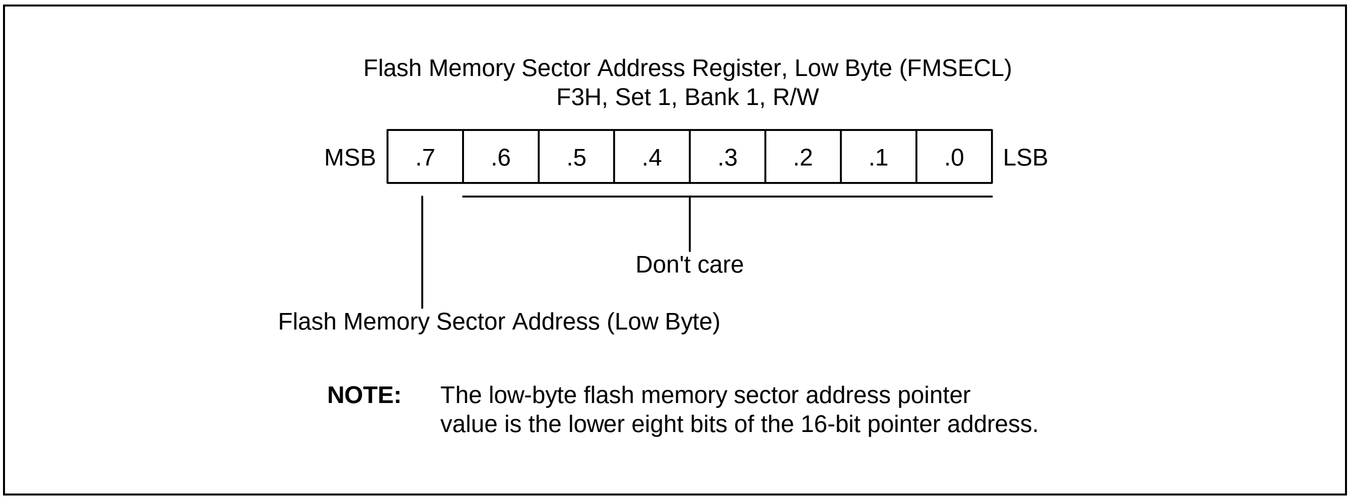


Figure 16-4. Flash Memory Sector Address Register, Low Byte (FMSECL)

SAMSUNG ELECTRONICS

ISPTM (ON-BOARD PROGRAMMING) SECTOR

ISPTM sectors located in program memory area can store on board program software (boot program code for upgrading application code by interfacing with I/O pin). The ISPTM sectors can not be erased or programmed by LDC instruction for the safety of On Board Program software.

The ISP sectors are available only when the ISP enable/disable bit is set 0, that is, enable ISP at the Smart Option. If you don't like to use ISP sector, this area can be used as a normal program memory (can be erased or programmed by LDC instruction) by setting ISP disable bit ("1") at the Smart Option. Even if ISP sector is selected, ISP sector can be erased or programmed in the Tool Program mode, by Serial programming tools. The size of ISP sector can be varied by settings of Smart Option. You can choose appropriate ISP sector size according to the size of On Board Program software.

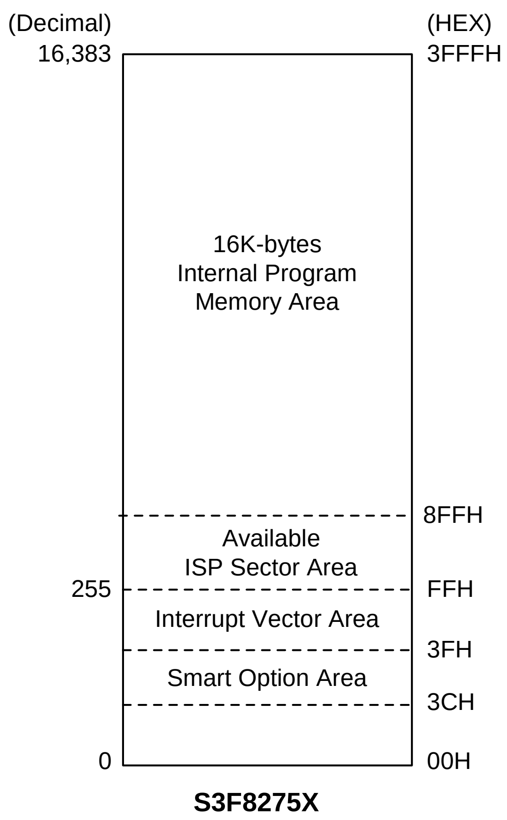


Figure 16-5. Program Memory Address Space

Table 16-1. ISP Sector Size

Smart Option(003EH) ISP Size Selection Bit			Area of ISP Sector	ISP Sector Size
Bit 2	Bit 1	Bit 0		
1	x	x	–	0
0	0	0	100H – 1FFH (256 byte)	256 Bytes
0	0	1	100H – 2FFH (512 byte)	512 Bytes
0	1	0	100H – 4FFH (1024 byte)	1024 Bytes
0	1	1	100H – 8FFH (2048 byte)	2048 Bytes

NOTE: The area of the ISP sector selected by Smart Option bit (003EH.2 – 003EH.0) can not be erased and programmed by LDC instruction in user program mode.

ISP Reset Vector and ISP Sector Size

If you use ISP sectors by setting the ISP enable/disable bit to "0" and the Reset Vector Selection bit to "0" at the Smart Option, you can choose the reset vector address of CPU as shown in Table 16-3 by setting the ISP Reset Vector Address Selection bits.

Table 16-2. Reset Vector Address

Smart Option (003EH) ISP Reset Vector Address Selection Bit			Reset Vector Address After POR	Usable Area for ISP Sector	ISP Sector Size
Bit 7	Bit 6	Bit 5			
1	x	x	0100H	–	–
0	0	0	0200H	100H – 1FFH	256 Bytes
0	0	1	0300H	100H – 2FFH	512 Bytes
0	1	0	0500H	100H – 4FFH	1024 Bytes
0	1	1	0900H	100H – 8FFH	2048 Bytes

NOTE: The selection of the ISP reset vector address by smart option (003EH.7 – 003EH.5) is not dependent of the selection of ISP sector size by smart option (003EH.2 – 003EH.0).

SAMSUNG ELECTRONICS

SECTOR ERASE

User can erase a flash memory partially by using sector erase function only in User Program Mode. The only unit of flash memory to be erased and programmed in User Program Mode is called sector.

The program memory of S3F8275X is divided into 128 sectors for unit of erase and programming. Every sector has all 128-byte sizes of program memory areas. So each sector should be erased first to program a new data (byte) into a sector. Minimum 10ms delay time for erase is required after setting sector address and triggering erase start bit (FMCON.0). Sector Erase is not supported in Tool Program Modes (MDS mode tool or Programming tool).

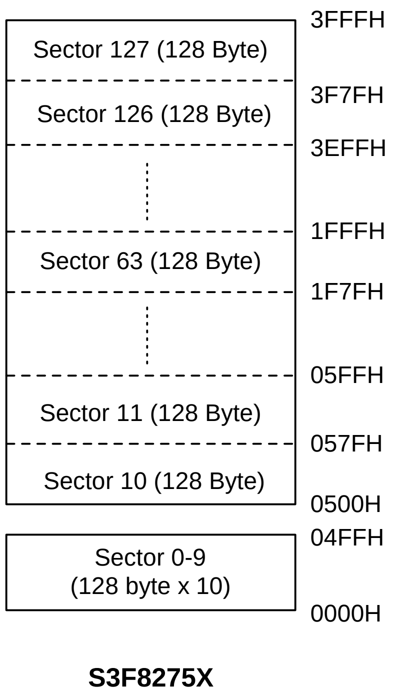


Figure 16-6. Sector Configurations in User Program Mode

The Sector Erase Procedure in User Program Mode

1. Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".
2. Set Flash Memory Sector Address Register (FMSECH/FMSECL).
3. Check user's ID code (written by user).
4. Set Flash Memory Control Register (FMCON) to "10100001B".
5. Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B"
6. Check the "sector erase status bit" whether "sector erase" is success or not.

☞ PROGRAMMING TIP — Sector Erase

```
            •
            •
            SB1
reErase:    LD      FMUSR,#0A5H             ; User program mode enable
            LD      FMSECH,#10H
            LD      FMSECL,#00H             ; Set sector address (1000H – 107FH)
            CP      UserID_Code,#User_value ; Check user's ID code (written by user)
                                            ; User_value is any value by user
            JR      NE,Not_ID_Code          ; If not equal, jump to Not_ID_Code
            LD      FMCON,#10100001B        ; Start sector erase
            NOP                             ; Dummy instruction, this instruction must be needed
            NOP                             ; Dummy instruction, this instruction must be needed
            LD      FMUSR,#0                ; User program mode disable
            TM      FMCON,#00001000B        ; Check "Sector erase status bit"
            JR      NZ,reErase              ; Jump to reErase if fail

            •
            •
            •
            •

Not_ID_Code:
            SB1
            LD      FMUSR,#0                ; User Program mode disable

            •
            •
            •
            •
```

PROGRAMMING

A flash memory is programmed in one byte unit after sector erase. And for programming safety's sake, must set FMSECH and FMSECL to flash memory sector value.

The write operation of programming starts by 'LDC' instruction.
You can write until 128byte, because this flash sector's limits is 128byte.
So if you written 128byte, must reset FMSECH and FMSECL.

The Program Procedure in User program Mode

1. Must erase sector before programming.
2. Set Flash Memory User Programming Enable Register (FMUSR) to “10100101B”.
3. Set Flash Memory Sector Register (FMSECH, FMSECL) to sector value of write address.
4. Load a flash memory upper address into upper register of pair working register.
5. Load a flash memory lower address into lower register of pair working register.
6. Load a transmission data into a working register.
7. Check user’s ID code (written by user)
8. Set Flash Memory Control Register (FMCON) to “01010001B”.
9. Load transmission data to flash memory location area on ‘LDC’ instruction by indirectly addressing mode
10. Set Flash Memory User Programming Enable Register (FMUSR) to “00000000B”.

☞ PROGRAMMING TIP — Program

```
        •
        •

        SB1
        LD      FMUSR,#0A5H           ; User Program mode enable
        LD      FMSECH,#17H
        LD      FMSECL,#80H           ; Set sector address (1780H–17FFH)
        LD      R2,#17H               ; Set a ROM address in the same sector 1780H–17FFH
        LD      R3,#84H
        LD      R4,#78H               ; Temporary data
        CP      UserID_Code,#User_value  ; Check user's ID code (written by user)
                                         ; User_value is any value by user
        JR      NE,Not_ID_Code           ; If not equal, jump to Not_ID_Code
        LD      FMCON,#01010001B      ; Start program
        LDC     @RR2,R4               ; Write the data to a address of same sector(1784H)
        NOP                           ; Dummy Instruction, This instruction must be needed
        LD      FMUSR,#0              ; User Program mode disable
        •
        •
        •
        •

Not_ID_Code:
        SB1
        LD      FMUSR,#0              ; User Program mode disable
        •
        •
        •
        •
```


SAMSUNG ELECTRONICS

READING

The read operation of programming starts by 'LDC' instruction.

The Reading Procedure in User Program Mode

1. Load a flash memory upper address into upper register of pair working register.
2. Load a flash memory lower address into lower register of pair working register.
3. Load receive data from flash memory location area on 'LDC' instruction by indirectly addressing mode.

☞ PROGRAMMING TIP — Reading

```
            •
            •

            LD      R2,#3H          ; Load flash memory upper address
                                    ; To upper of pair working register
            LD      R3,#0           ; Load flash memory lower address
                                    ; To lower pair working register
LOOP:       LDC     R0,@RR2         ; Read data from flash memory location
                                    ; (Between 300H and 3FFH)
            INC     R3
            CP      R3,#0H
            JP      NZ,LOOP

            •
            •
            •
            •
```

HARD LOCK PROTECTION

User can set Hard Lock Protection by write '0110' in FMCON.7–4. If this function is enabled, the user cannot write or erase the data in a flash memory area. This protection can be released by the chip erase execution (in the tool program mode).

In terms of user program mode, the procedure of setting Hard Lock Protection is following that. Whereas in tool mode the manufacturer of serial tool writer could support Hardware Protection. Please refer to the manual of serial program writer tool provided by the manufacturer.

The Hard Lock Protection Procedure in User program Mode

1. Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".
2. Check user's ID code (written by user)
3. Set Flash Memory Control Register (FMCON) to "01100001B".
4. Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".

☞ **PROGRAMMING TIP — Hard Lock Protection**

```
              •
              •
              SB1
              LD        FMUSR,#0A5H              ; User Program mode enable
              CP        UserID_Code,#User_value  ; Check user's ID code (written by user)
                                                 ; User_value is any value by user
              JR        NE,Not_ID_Code           ; If not equal, jump to Not_ID_Code
              LD        FMCON,#01100001B         ; Hard Lock mode set & start
              NOP                                ; Dummy Instruction, This instruction must be needed
              LD        FMUSR,#0                 ; User Program mode disable
              •
              •
              •
              •
Not_ID_Code:
              SB1
              LD        FMUSR,#0                 ; User Program mode disable
              •
              •
              •
              •
```

17 ELECTRICAL DATA

OVERVIEW

In this chapter, S3C8275X/C8278X/C8274X electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

- Absolute maximum ratings
- D.C. electrical characteristics
- Data retention supply voltage in Stop mode
- Stop mode release timing when initiated by an external interrupt
- Stop mode release timing when initiated by a RESET
- I/O capacitance
- A.C. electrical characteristics
- Input timing for external interrupts
- Input timing for RESET
- Serial data transfer timing
- BLD electrical characteristics
- LVR electrical characteristics
- Oscillation characteristics
- Oscillation stabilization time
- Operating voltage range
- A.C. electrical characteristics for Internal flash ROM

Table 17-1. Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 4.6	V
Input voltage	V_I	Ports 0–6	– 0.3 to V_{DD} + 0.3	V
Output voltage	V_O	–	– 0.3 to V_{DD} + 0.3	V
Output current High	I_{OH}	One I/O pin active	– 15	mA
		All I/O pins active	– 60	
Output current Low	I_{OL}	One I/O pin active	+ 30 (Peak value)	mA
		Total pin current for ports	+ 100 (Peak value)	
Operating temperature	T_A	–	– 25 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	°C

Table 17-2. D.C. Electrical Characteristics

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.0 V to 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating voltage	V_{DD}	fx = 0.4 – 4.2MHz, fxt = 32.8kHz	2.0	–	3.6	V
		fx = 0.4 – 8.0MHz	2.5	–	3.6	
Input high voltage	V_{IH1}	All input pins except for V_{IH2}, V_{IH3}	0.7 V_{DD}	–	V_{DD}	V
	V_{IH2}	Ports 0-1, nRESET	0.8 V_{DD}		V_{DD}	
	V_{IH3}	X_{IN}, X_{OUT} and XT_{IN}, XT_{OUT}	V_{DD} – 0.1		V_{DD}	
Input low voltage	V_{IL1}	All input pins except for V_{IL2}, V_{IL3}	–	–	0.3 V_{DD}	V
	V_{IL2}	Ports 0-1, nRESET	–	–	0.2 V_{DD}	
	V_{IL3}	X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}	–	–	0.1	
Output high voltage	V_{OH}	V_{DD} = 2.7 to 3.6 V; All output ports; I_{OH} = –1 mA	V_{DD} – 1.0	–	–	V
Output low voltage	V_{OL1}	V_{DD} = 2.7 to 3.6 V I_{OL} = 15mA Ports 0-1	–	–	1.0	V
	V_{OL2}	V_{DD} = 2.7 to 3.6 V I_{OL} = 10mA All output ports except for V_{OL1}	–	–	1.0	V
Input high leakage current	I_{LIH1}	V_I = V_{DD} All input pins except for I_{LIH2}	–	–	3	μA
	I_{LIH2}	V_I = V_{DD} X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}			20	

Table 17-2. D.C. Electrical Characteristics (Continued)

(T_A = – 25°C to + 85°C, V_{DD} = 2.0 V to 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input low leakage current	I_{LIL1}	V_I = 0 V; All input pins except nRESET, I_{LIL2}	–	–	–3	μA
	I_{LIL2}	V_I = 0 V; X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}	–	–	–20	
Output high leakage current	I_{LOH}	V_O = V_{DD} All output pins	–	–	3	
Output low leakage current	I_{LOL}	V_O = 0 V All output pins	–	–	–3	
Pull-up resistors	R_{L1}	V_I = 0 V; V_{DD} = 3V, T_A = 25°C Ports 0–6	40	70	100	kΩ
	R_{L2}	V_I = 0 V; V_{DD} = 3V, T_A = 25°C nRESET	220	360	500	
Oscillator feed back resistors	R_{OSC1}	V_{DD} = 3 V, T_A = 25 °C X_{IN} = V_{DD}, X_{OUT} = 0V	600	1700	3000	kΩ
	R_{OSC2}	V_{DD} = 3 V, T_A = 25 °C XT_{IN} = V_{DD}, XT_{OUT} = 0 V	2000	4000	8000	
LCD voltage dividing resistor	R_{LCD}	T_A = 25 °C	60	110	160	kΩ
$\lvert V_{LCD}\text{-}COMi \rvert$ voltage drop (i = 0-3)	V_{DC}	– 15 μA per common pin	–	–	120	mV
$\lvert V_{LCD}\text{-}SEGx \rvert$ voltage drop (x = 0–31)	V_{DS}	– 15 μA per common pin	–	–	120	
Middle output voltage (1)	V_{LC1}	V_{DD} = 2.7 V to 3.6 V, 1/3 bias LCD clock = 0Hz, V_{LC0} = V_{DD}	2/3V_{DD}–0.2	2/3V_{DD}	2/3V_{DD}+ 0.2	V
	V_{LC2}		1/3V_{DD}–0.2	1/3V_{DD}	1/3V_{DD}+ 0.2	

NOTE: It is middle output voltage when the V_{LC0} pin is opened.

Table 17-2. D.C. Electrical Characteristics (Concluded)

(T_A = – 25°C to + 85°C, V_{DD} = 2.0 V to 3.6 V)

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Supply current [1]	I_{DD1} [2]	Run mode: V_{DD} = 3.3 V ± 0.3 V	8.0 MHz	–	3.0	6.0	mA
		Crystal oscillator C1 = C2 = 22pF	4.0 MHz		1.5	3.0	
	I_{DD2} [2]	Idle mode: V_{DD} = 3.3 V ± 0.3 V	8.0 MHz		0.5	1.6	
		Crystal oscillator C1 = C2 = 22pF	4.0 MHz		0.4	1.2	
	I_{DD3} [3]	Run mode: V_{DD} = 3.3 V ± 0.3 V, 32 kHz crystal oscillator T_A = 25 °C, OSCCON.7=1			12.0	25.0	µA
	I_{DD4} [3]	Idle mode: V_{DD} = 3.3 V ± 0.3 V, 32 kHz crystal oscillator T_A = 25 °C, OSCCON.7=1			2.0	4.0	
	I_{DD5} [4]	Stop mode; V_{DD} = 3.3 V ± 0.3 V	T_A = 25 °C		0.2	2.0	
			T_A = –25 °C ~ +85 °C		–	10	

NOTES:

1. Supply current does not include current drawn through internal pull-up resistors, LCD voltage dividing resistors, the LVR block, and external output current loads.
2. I_{DD1} and I_{DD2} include power consumption for sub clock oscillation.
3. I_{DD3} and I_{DD4} are current when main clock oscillation stops and the sub clock is used (OSCCON.7=1).
4. I_{DD5} is current when main clock and sub clock oscillation stops.
5. Every values in this table is measured when bits 4-3 of the system clock control register (CLKCON.4-.3) is set to 11B.

Table 17-3. Data Retention Supply Voltage in Stop Mode

(T_A = – 25 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	–	2.0	–	3.6	V
Data retention supply current	I_{DDDR}	Stop mode, T_A = 25 °C V_{DDDR} = 2.0 V Disable LVR block	–	–	1	μA

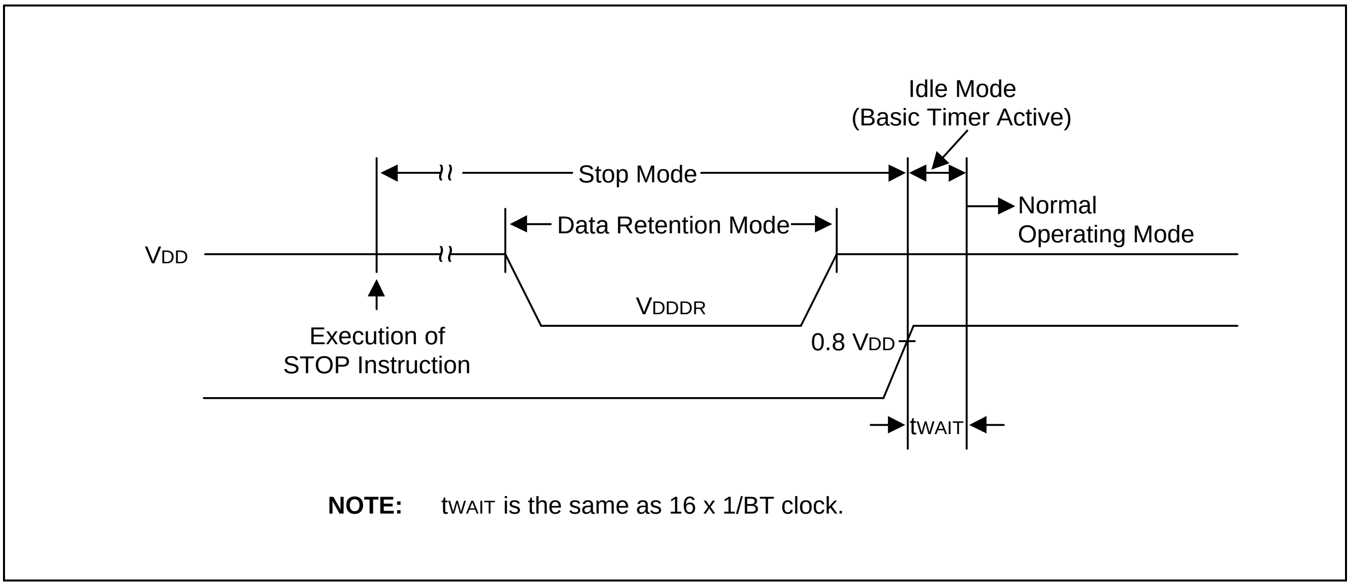
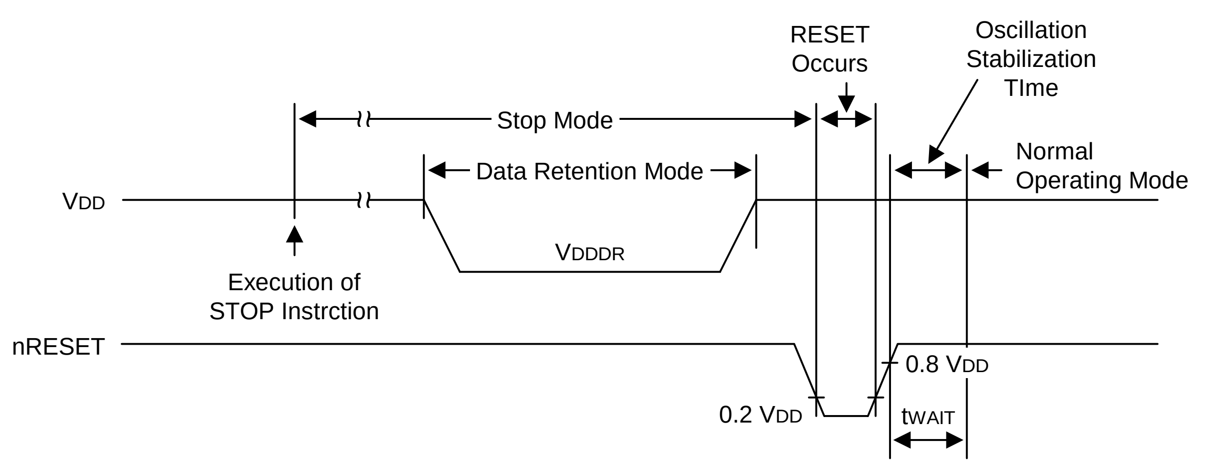


Figure 17-1. Stop Mode Release Timing When Initiated by an External Interrupt



NOTE: tWAIT is the same as 16 × 1/BT clock.

Figure 17-2. Stop Mode Release Timing When Initiated by a RESET

Table 17-4. Input/Output Capacitance

(T_A = –25 °C ~ + 85 °C, V_{DD} = 0 V)

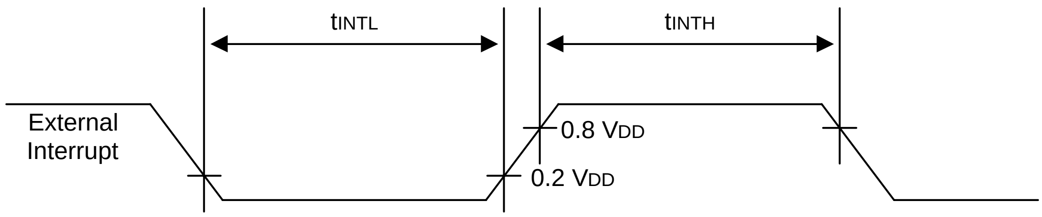
Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input capacitance	C_{IN}	f = 1 MHz; unmeasured pins are connected to V_{SS}	–	–	10	pF
Output capacitance	C_{OUT}					
I/O capacitance	C_{IO}					

SAMSUNG ELECTRONICS

Table 17-5. A.C. Electrical Characteristics

(T_A = – 25°C to + 85°C, V_{DD} = 2.0 V to 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
SCK cycle time	t_{KCY}	External SCK source	1,000	–	–	ns
		Internal SCK source	1,000			
SCK high, low width	t_{KH}, t_{KL}	External SCK source	500			
		Internal SCK source	t_{KCY}/2–50			
SI setup time to SCK high	t_{SIK}	External SCK source	250			
		Internal SCK source	250			
SI hold time to SCK high	t_{KSI}	External SCK source	400			
		Internal SCK source	400			
Output delay for SCK to SO	t_{KSO}	External SCK source	–	–	300	ns
		Internal SCK source			250	
Interrupt input, High, Low width	t_{INTH}, t_{INTL}	All interrupt V_{DD} = 3 V	500	700	–	ns
nRESET input Low width	t_{RSL}	Input V_{DD} = 3 V	10	–	–	µs



NOTE: The unit tCPU means one CPU clock period.

Figure 17-3. Input Timing for External Interrupts

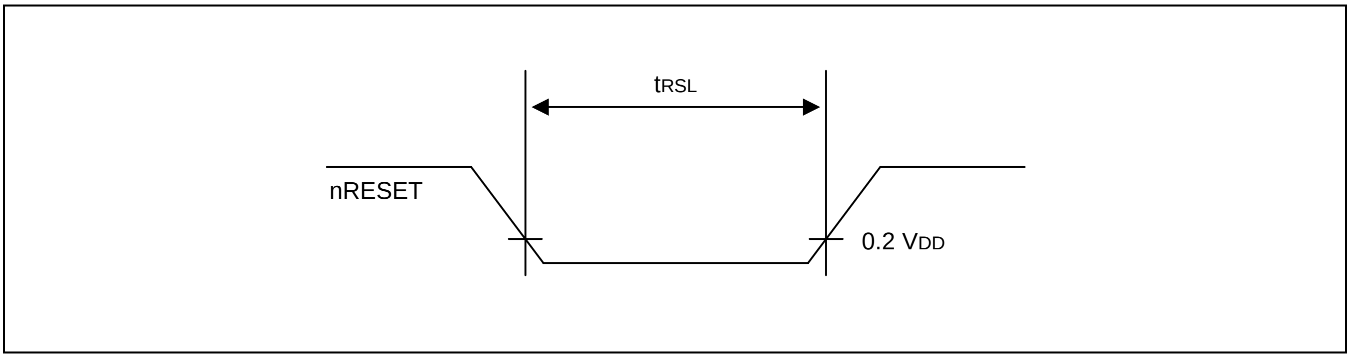


Figure 17-4. Input Timing for RESET

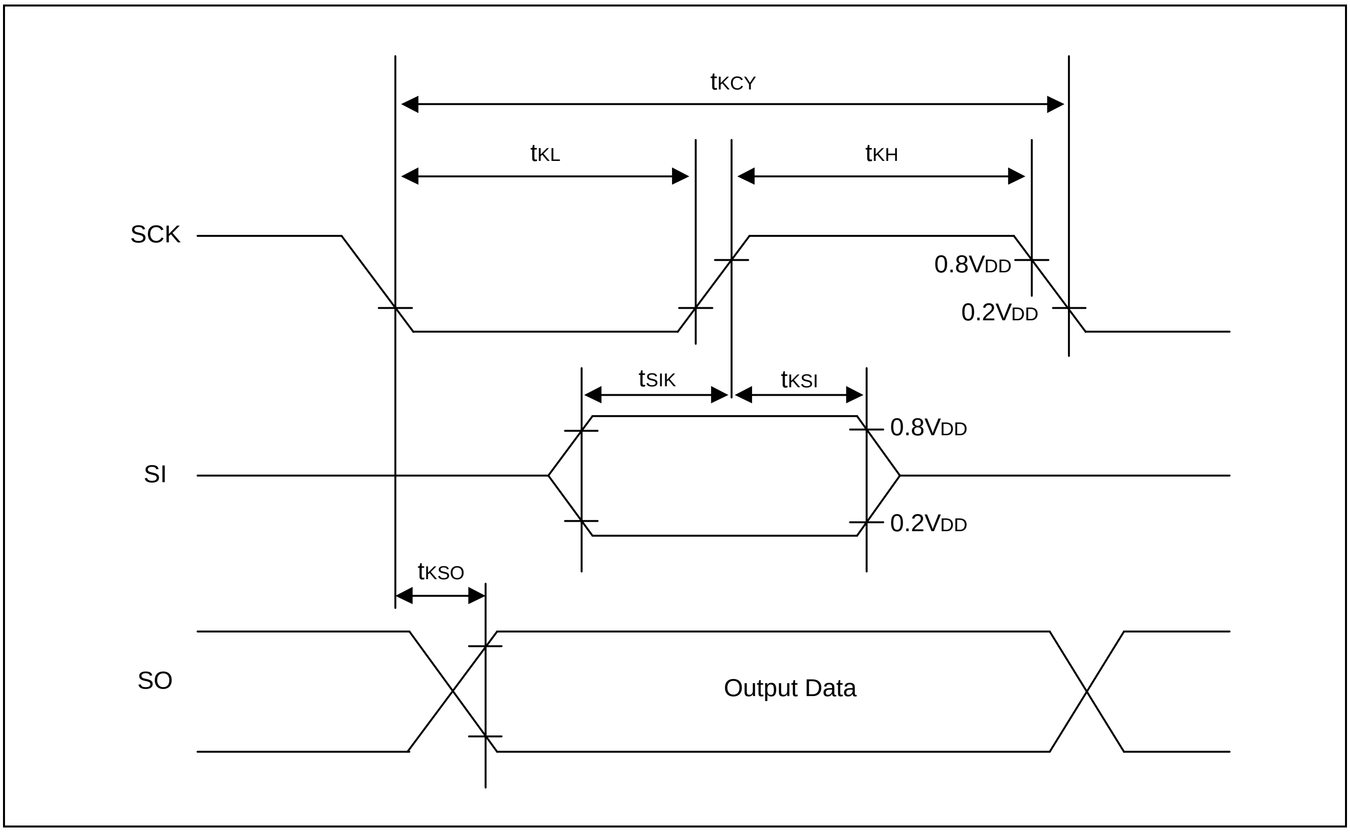


Figure 17-5. Serial Data Transfer Timing

Table 17-6. Battery Level Detector Electrical Characteristics

(T_A = 25°C, V_{DD} = 2.0 V to 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating voltage of BLD	V_{DDBLD}	–	2.0	–	3.6	V
Voltage of BLD	V_{BLD}	BLDCON.2-.0 = 000b	2.0	2.2	2.4	
		BLDCON.2-.0 = 101b	2.15	2.4	2.65	
		BLDCON.2-.0 = 011b	2.5	2.8	3.1	
Current consumption	I_{BLD}	V_{DD} = 3.3 V	–	70	120	μA
		V_{DD} = 2.2 V	–	50	100	
Hysteresis voltage of BLD	ΔV	BLDCON.2-0 = 000b, 101b, 011b	–	10	100	mV
BLD circuit response time	T_B	Fw = 32.768 kHz	–	–	1	ms

Table 17-7. LVR (Low Voltage Reset) Electrical Characteristics

(T_A = 25°C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Voltage of LVR	V_{LVR}	T_A = 25 °C	2.0	2.2	2.4	V
V_{DD} voltage rising time	t_R	–	10	–	–	μs
V_{DD} voltage off time	t_{OFF}	–	0.5	–	–	s
Hysteresis voltage of LVR	ΔV	–	–	10	100	mV
Current consumption	I_{DDPR}	V_{DD} = 3.3 V	–	70	120	μA

NOTES:
1. The current of LVR circuit is consumed when LVR is enabled by "Smart Option"
2. Current consumed when low voltage reset circuit is provided internally.

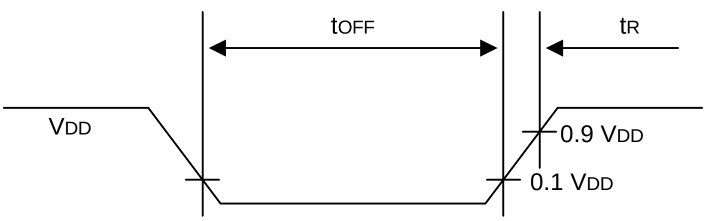


Figure 17-6. LVR (Low Voltage Reset) Timing

Table 17-8. Main Oscillation Characteristics

(T_A = – 25°C to + 85°C)

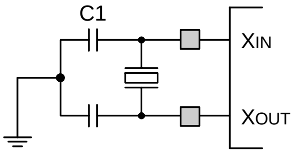
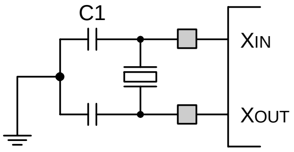
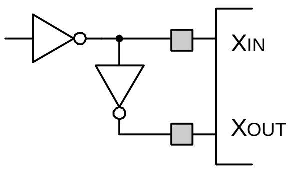
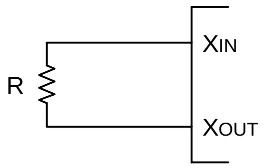
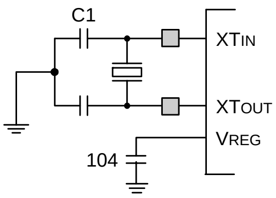
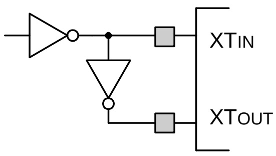
Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Crystal		Main oscillation frequency	2.5 V – 3.6 V	0.4	–	8	MHz
			2.0 V – 3.6 V	0.4	–	4.2	
Ceramic oscillator		Main oscillation frequency	2.5 V – 3.6 V	0.4	–	8	
			2.0 V – 3.6 V	0.4	–	4.2	
External clock		X_{IN} input frequency	2.5 V – 3.6 V	0.4	–	8	
			2.0 V – 3.6 V	0.4	–	4.2	
RC oscillator		Frequency	3.3 V	0.4	–	1	MHz

Table 17-9. Sub Oscillation Characteristics

(T_A = – 25°C to + 85°C)

Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Crystal		Sub oscillation frequency	2.0 V – 3.6 V	32	32.768	35	kHz
External clock		XT_{IN} input frequency	2.0 V – 3.6 V	32	–	100	

SAMSUNG ELECTRONICS

Table 17-10. Main Oscillation Stabilization Time

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.0 V to 3.6 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	fx > 1 MHz	–	–	40	ms
Ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	10	ms
External clock	X_{IN} input high and low width (t_{XH}, t_{XL})	62.5	–	1250	ns

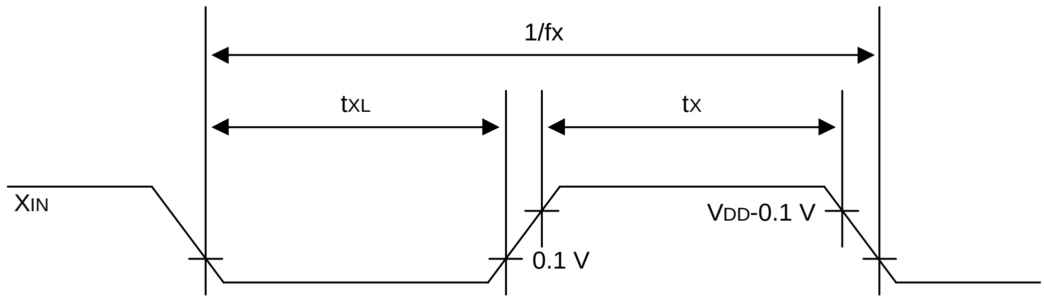


Figure 17-7. Clock Timing Measurement at X_{IN}

Table 17-11. Sub Oscillation Stabilization Time

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.0 V to 3.6 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	–	–	–	10	s
External clock	XT_{IN} input high and low width (t_{XH}, t_{XL})	5	–	15	μs

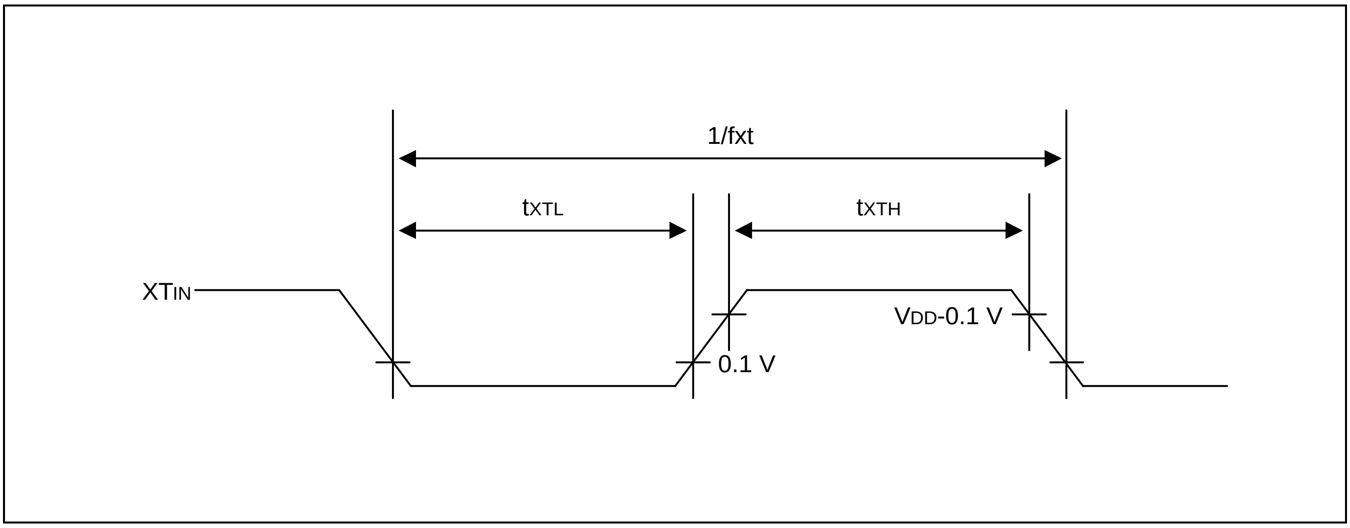


Figure 17-8. Clock Timing Measurement at XT_{IN}

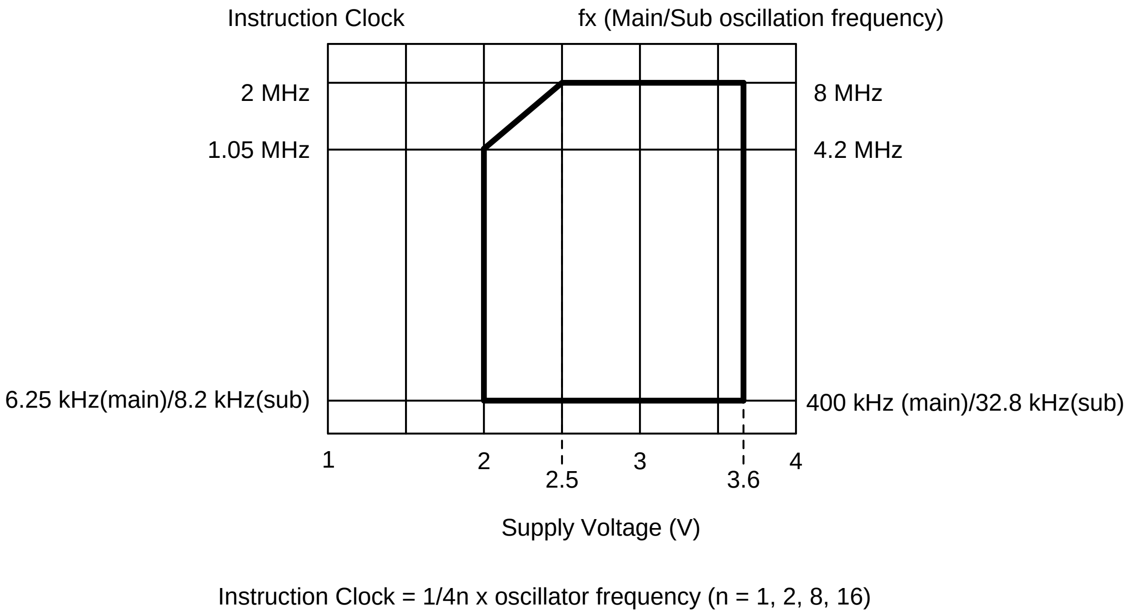


Figure 17-9. Operating Voltage Range

Table 17-12. A.C. Electrical Characteristics for Internal Flash ROM

(T_A = – 25 °C to + 85 °C, V_{DD} = 2.2 V to 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Programming time (1)	Ftp	–	30	–	–	µs
Chip erasing time (2)	Ftp1	–	50	–	–	ms
Sector erasing time (3)	Ftp2	–	10	–	–	ms
Data access time	Ft_{RS}	–	–	25	–	ns
Number of writing/erasing	FNwe	–	–	–	10,000(4)	Times

NOTES:
1. The programming time is the time during which one byte (8-bit) is programmed.
2. The chip erasing time is the time during which all 16K byte block is erased.
3. The sector erasing time is the time during which all 128 byte block is erased.
4. **Maximum number of writing/erasing is 10,000 times for full-flash(S3F8275X) and 100 times for half-flash (S3F8278X/F8274X).**
5. The chip erasing is available in Tool Program Mode only.

18 MECHANICAL DATA

OVERVIEW

The S3C8275X/C8278X/C8274X microcontroller is currently available in a 64-pin QFP and LQFP package.

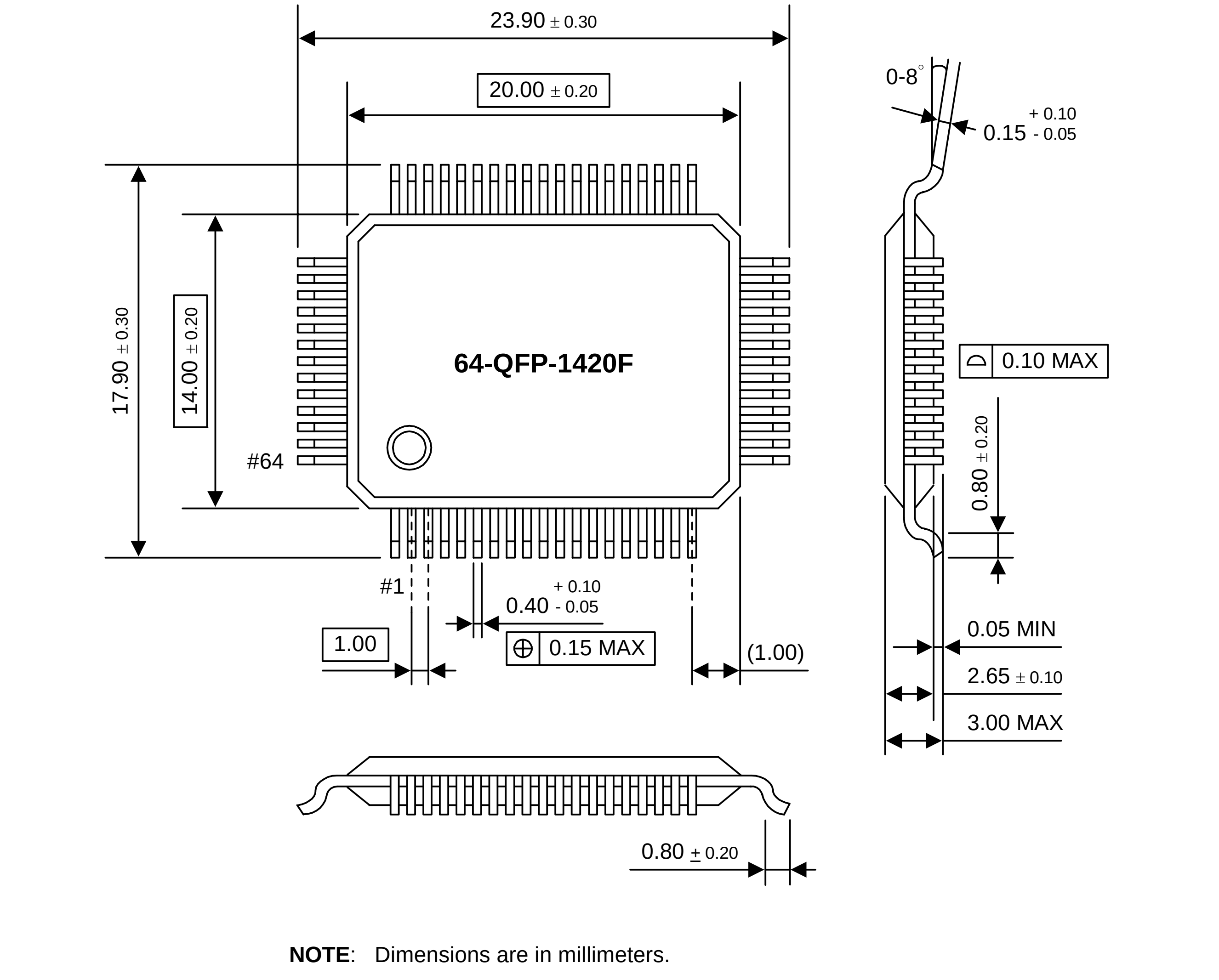
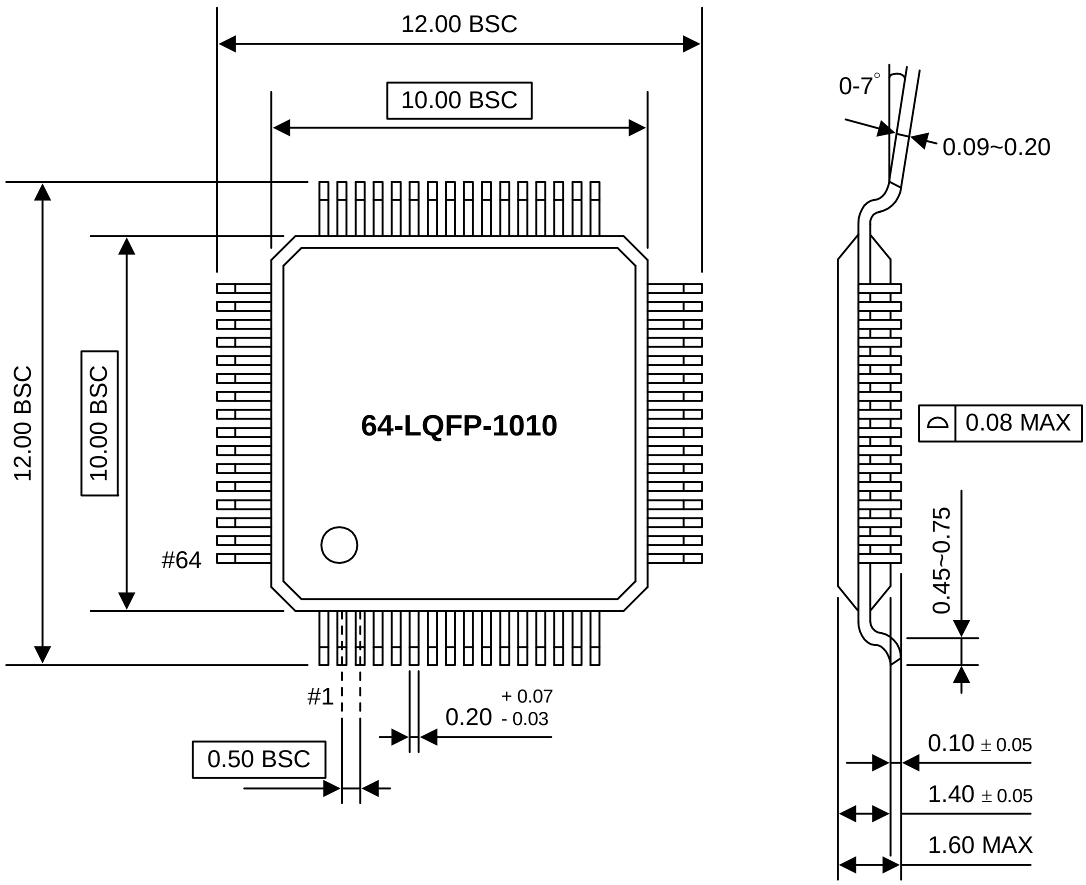


Figure 18-1. 64-Pin QFP Package Dimensions (64-QFP-1420F)



NOTE: Dimensions are in millimeters.

Figure 18-2. 64-Pin LQFP Package Dimensions (64-LQFP-1010)

19 S3F8275X/F8278X/F8274X FLASH MCU

OVERVIEW

The S3F8275X/F8278X/F8274X single-chip CMOS microcontroller is the Flash MCU version of the S3C8275X/C8278X/C8274X microcontroller. It has an on-chip Flash ROM instead of masked ROM. The Flash ROM is accessed by serial data format.

The S3F8275X/F8278X/F8274X is fully compatible with the S3C8275X/C8278X/C8274X, both in function and in pin configuration. Because of its simple programming requirements, the S3F8275X/F8278X/F8274X is ideal for use as an evaluation chip for the S3C8275X/C8278X/C8274X.

NOTE: This chapter is about the Tool Program Mode of Flash MCU. If you want to know the User Program Mode, refer to the chapter 16. Embedded Flash Memory Interface.

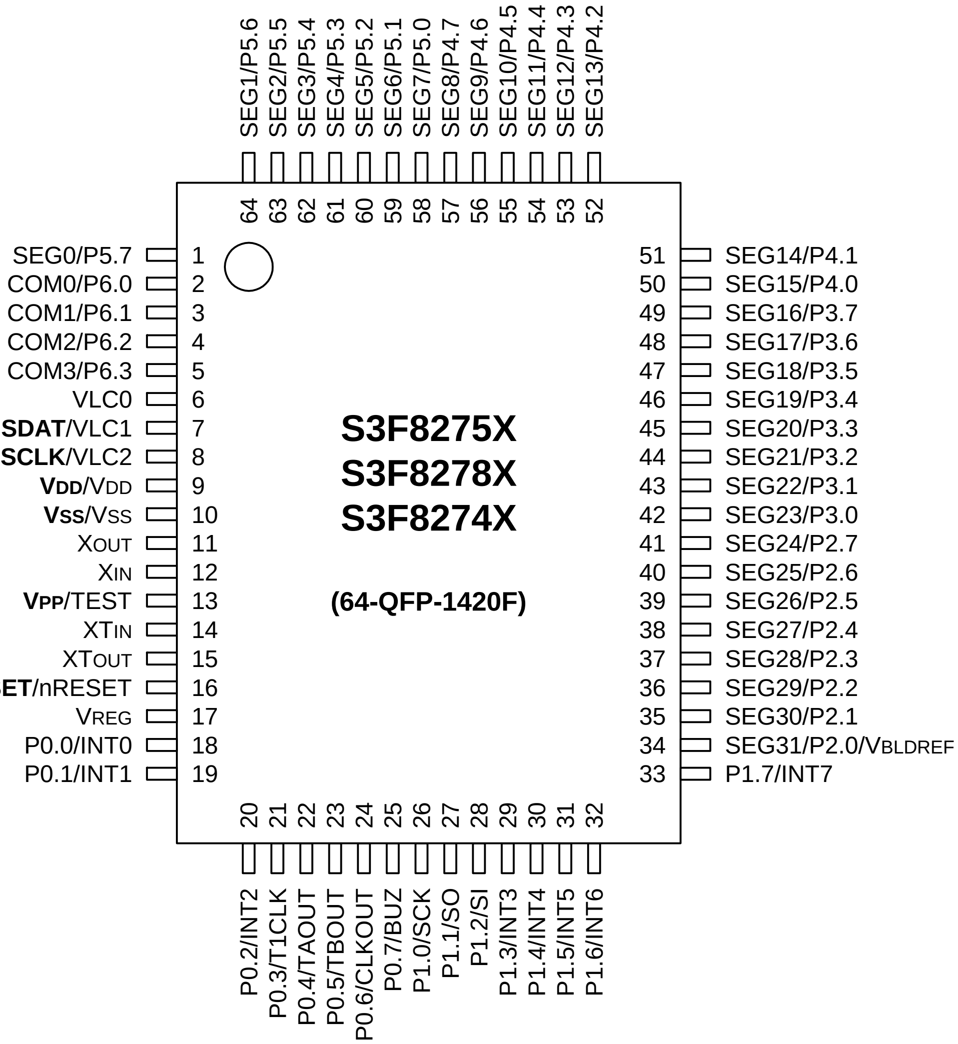


Figure 19-1. S3F8275X/F8278X/F8274X Pin Assignments (64-QFP-1420F)

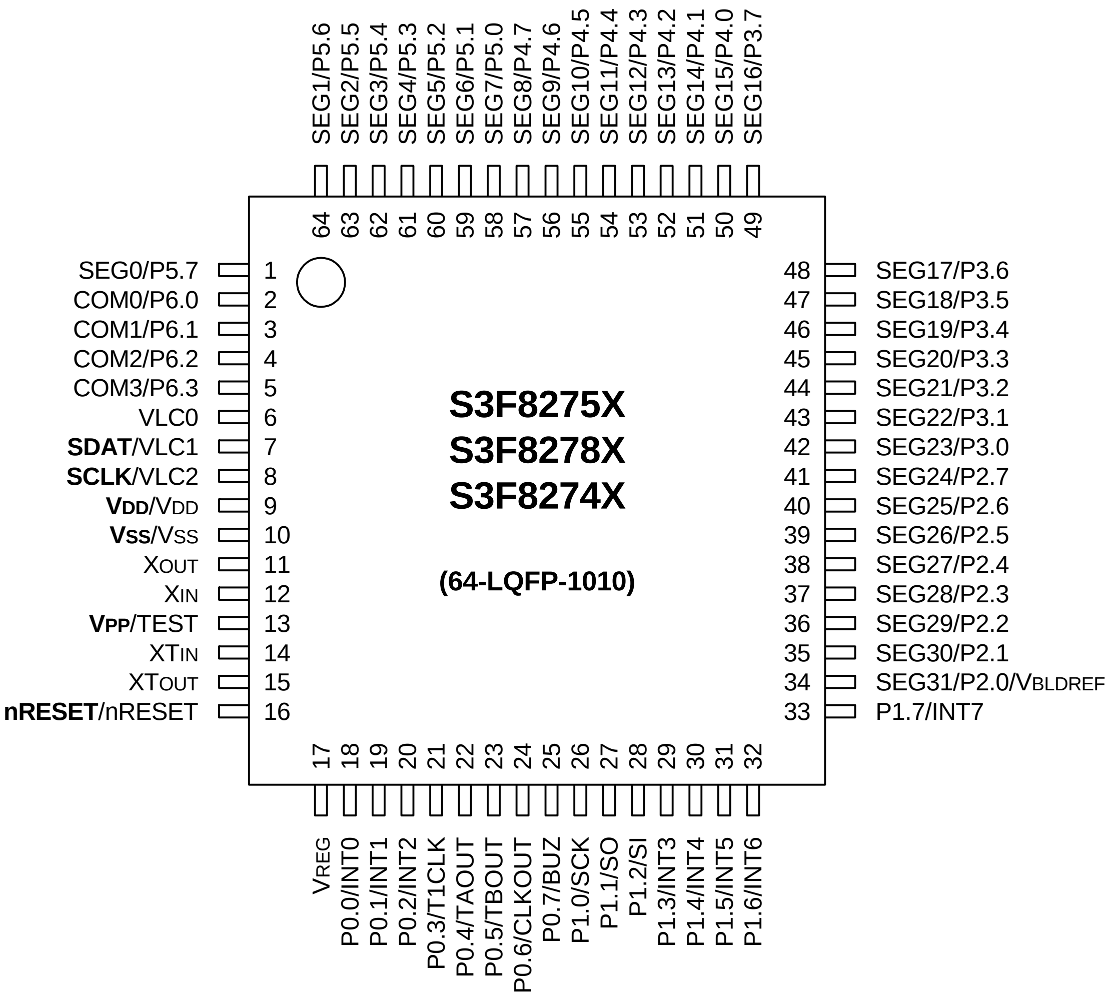


Figure 19-2. S3F8275X/F8278X/F8274X Pin Assignments (64-LQFP-1010)

Table 19-1. Descriptions of Pins Used to Read/Write the Flash ROM

Main Chip	During Programming			
Pin Name	**Pin Name**	**Pin No.**	**I/O**	**Function**
VLC1	SDAT	7	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as an Input or push-pull output port.
VLC2	SCLK	8	I/O	Serial clock pin. Input only pin.
TEST	V_{PP}	13	I	S3F8278X/F8274X: Power supply pin for Flash ROM cell reading/writing. 12.5V is applied in Flash writing mode and 3.3V is applied in Flash reading mode. S3F8275X: Power supply pin for Flash ROM cell reading/writing. 3.3V is applied in Flash reading/writing mode because internal block makes 12.5V. So, TEST pin must be connected to V_{DD}.
nRESET	nRESET	16	I	Chip initialization
V_{DD}/V_{SS}	V_{DD}/V_{SS}	9 / 10	I	Power supply pin for logic circuit. V_{DD} should be tied to +3.3 V during programming.

Table 19-2. Comparison of S3F8275X/F8278X/F8274X and S3C8275X/C8278X/C8274X Features

Characteristic	S3F8275X/F8278X/F8274X	S3C8275X/C8278X/C8274X
Program memory	16/8/4-Kbyte Flash ROM	16/8/4-Kbyte mask ROM
Operating voltage (V_{DD})	2.0 V to 3.6 V	2.0 V to 3.6 V
Flash ROM programming mode	V_{DD} = 3.3 V, V_{PP}(TEST)=12.5V	–
Pin configuration	64-QFP, 64-LQFP	64-QFP, 64-LQFP
Flash ROM programmability	User Program multi time	Programmed at the factory

NOTE: The V_{PP} (Test) pin must be connected to V_{DD} (S3F8275X only).

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP}(TEST) pin of the S3F8275X/F8278X/F8274X, the 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 19-3 below.

Table 19-3. Operating Mode Selection Criteria

V_{DD}	V_{PP} (TEST)	REG/MEM	Address (A15-A0)	R/W	Mode
3.3 V	3.3 V	0	0000H	1	Flash ROM read
	12.5 V	0	0000H	0	Flash ROM program
	12.5 V	1	0E3FH	0	Flash ROM read protection

NOTES:
1. The V_{PP} (Test) pin must be connected to V_{DD} (S3F8275X only).
2. "0" means Low level; "1" means High level.

Table 19-4. D.C. Electrical Characteristics

(T_A = – 25°C to + 85°C, V_{DD} = 2.0 V to 3.6 V)

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Supply current (1)	I_{DD1} (2)	Run mode: V_{DD} = 3.3 V ± 0.3 V Crystal oscillator C1 = C2 = 22pF	8.0 MHz	–	3.0	6.0	mA
			4.0 MHz		1.5	3.0	
	I_{DD2} (2)	Idle mode: V_{DD} = 3.3 V ± 0.3 V Crystal oscillator C1 = C2 = 22pF	8.0 MHz		0.5	1.6	
			4.0 MHz		0.4	1.2	
	I_{DD3} (3)	Run mode: V_{DD} = 3.3 V ± 0.3 V, 32 kHz crystal oscillator T_A = 25 °C, OSCCON.7=1			12.0	25.0	μA
	I_{DD4} (3)	Idle mode: V_{DD} = 3.3 V ± 0.3 V 32 kHz crystal oscillator T_A = 25 °C, OSCCON.7=1			2.0	4.0	
	I_{DD5} (4)	Stop mode; V_{DD} = 3.3V ± 0.3 V	T_A = 25 °C		0.2	2.0	
			T_A = –25 °C ~ +85 °C		–	10	

NOTES:

1. Supply current does not include current drawn through internal pull-up resistors, LCD voltage dividing resistors, the LVR block and external output current loads.
2. I_{DD1} and I_{DD2} include power consumption for sub clock oscillation.
3. I_{DD3} and I_{DD4} are current when main clock oscillation stops and the sub clock is used (OSCCON.7=1).
4. I_{DD5} is current when main clock and sub clock oscillation stops.
5. Every values in this table is measured when bits 4-3 of the system clock control register (CLKCON.4-.3) is set to 11B.

SAMSUNG ELECTRONICS

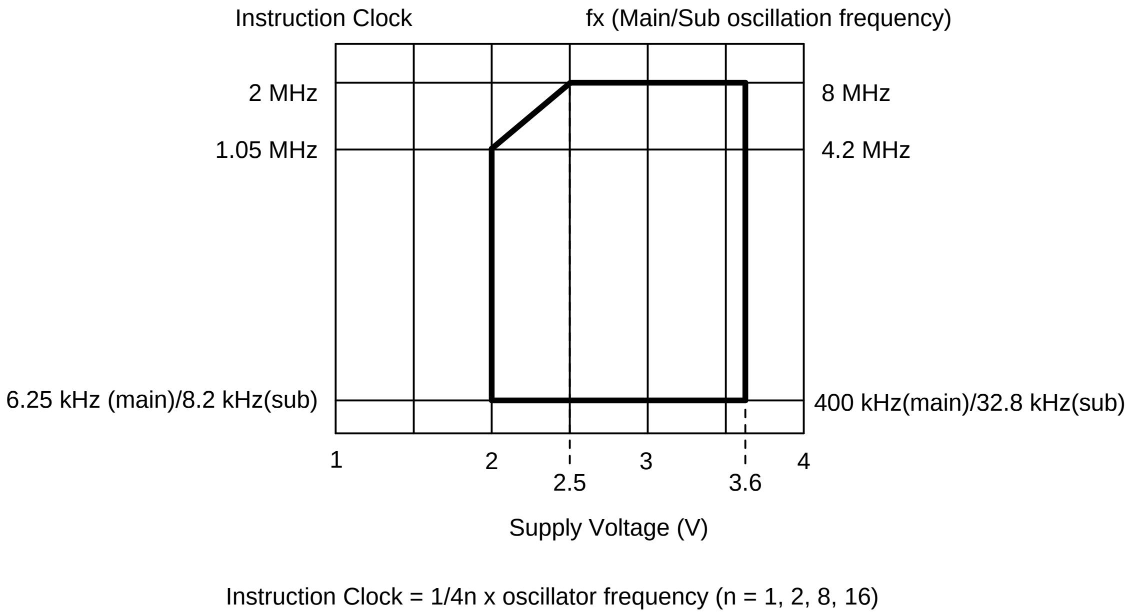


Figure 19-3. Operating Voltage Range

20 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system in turnkey form. The development support system is configured with a host system, debugging tools, and support software. For the host system, any standard computer that operates with MS-DOS, Windows 95, and 98 as its operating system can be used. One type of debugging tool including hardware and software is provided: the sophisticated and powerful in-circuit emulator, SMDS2+, and OPENice for S3C7, S3C9, S3C8 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

SHINE

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

SAMA ASSEMBLER

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

SASM88

The SASM88 is a relocatable assembler for Samsung's S3C8-series microcontrollers. The SASM88 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM88 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces the relocatable object code only, so the user should link object file. Object files can be linked with other object files and loaded into memory.

HEX2ROM

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

TARGET BOARDS

Target boards are available for all S3C8-series microcontrollers. All required target system cables and adapters are included with the device-specific target board.

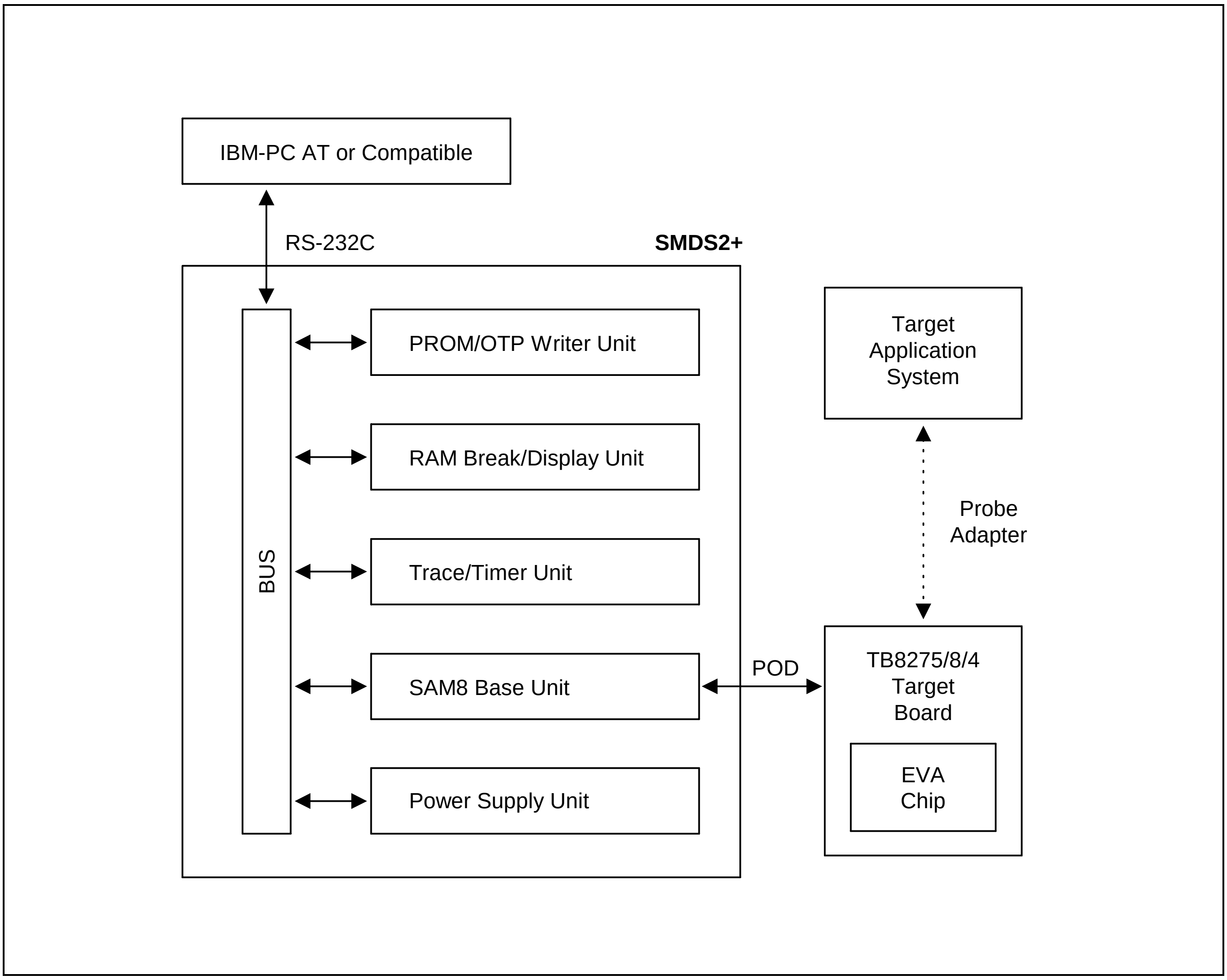


Figure 20-1. SMDS Product Configuration (SMDS2+)

TB8275/8/4 TARGET BOARD

The TB8275/8/4 target board is used for the S3C8275X/C8278X/C8274X microcontroller. It is supported with the SMDS2+.

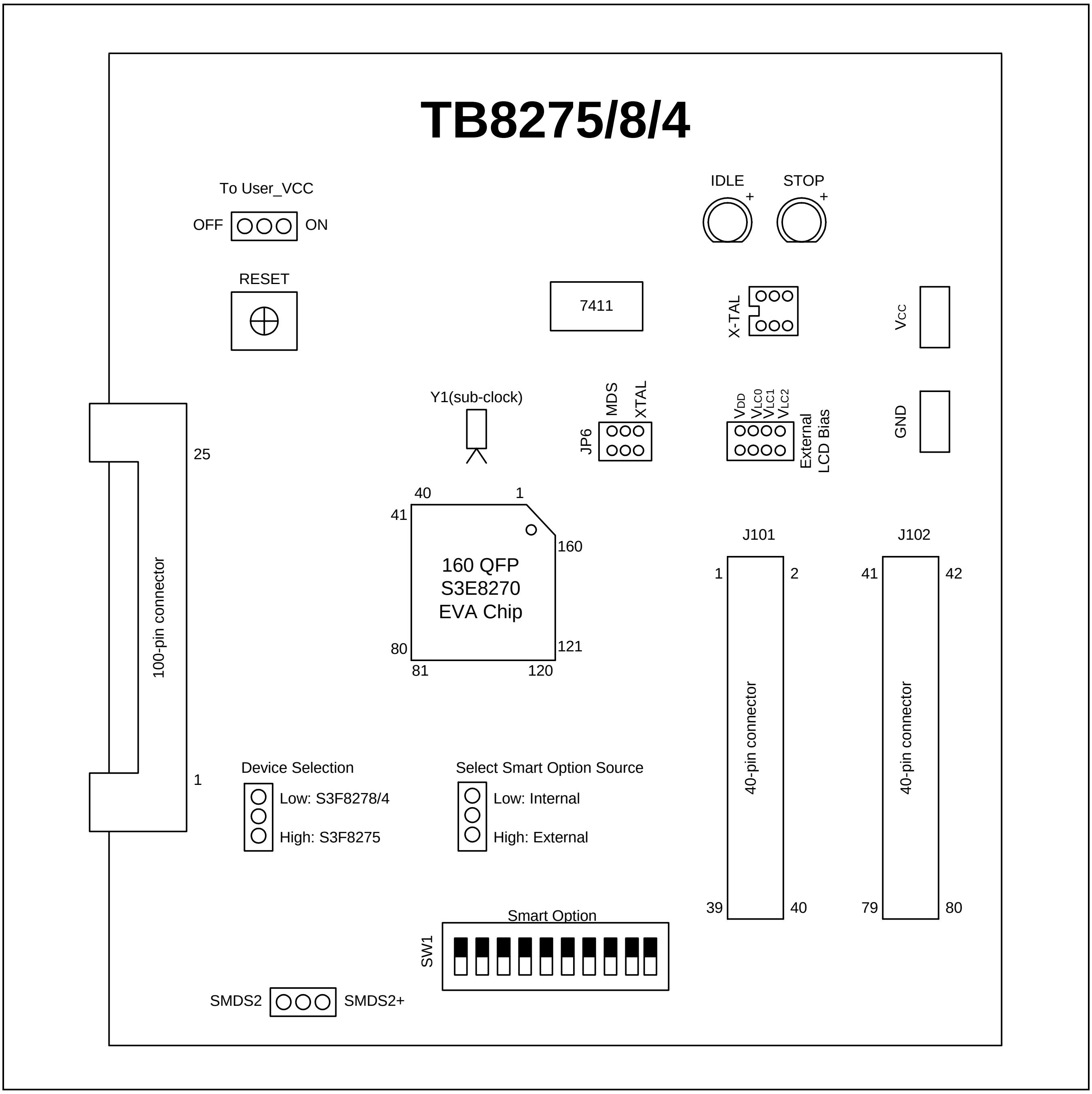
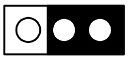
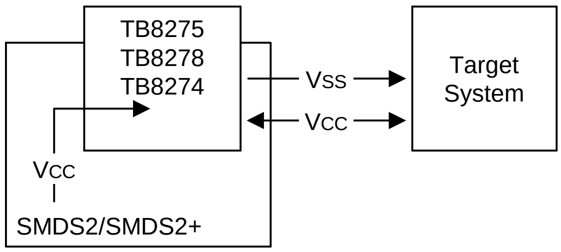
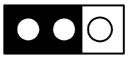
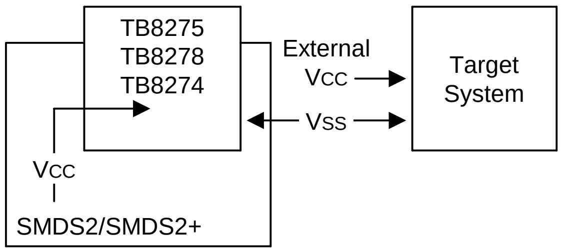


Figure 20-2. TB8275/8/4 Target Board Configuration

Table 20-1. Power Selection Settings for TB8275/8/4

<table>
<tr><th>"To User_Vcc" Settings</th><th>Operating Mode</th><th>Comments</th></tr>
<tr><td>To User_Vcc
Off  On</td><td>
</td><td>The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.</td></tr>
<tr><td>To User_Vcc
Off  On</td><td>
</td><td>The SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.</td></tr>
</table>

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

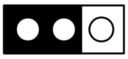


Table 20-2. Main-clock Selection Settings for TB8275/8/4

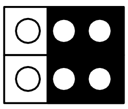
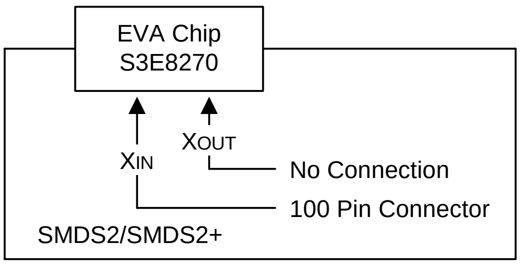
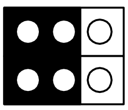
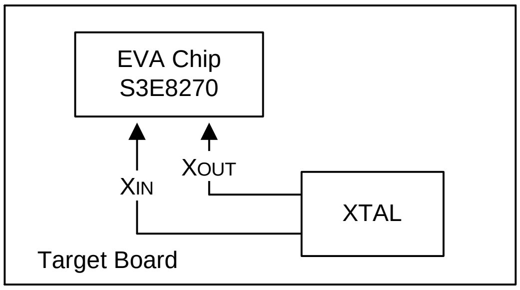
<table>
<tr><th>Main Clock Settings</th><th>Operating Mode</th><th>Comments</th></tr>
<tr><td>XIN
XTAL  MDS</td><td>
</td><td>Set the XI switch to "MDS" when the target board is connected to the SMDS2/SMDS2+.</td></tr>
<tr><td>XIN
XTAL  MDS</td><td>
</td><td>Set the XI switch to "XTAL" when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+.</td></tr>
</table>

Table 20-3. Select Smart Option Source Setting for TB8275/8/4

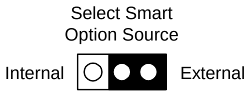
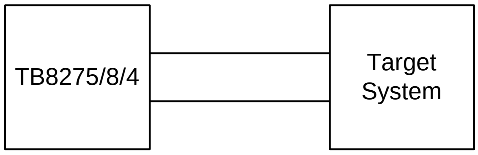
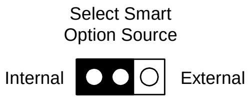
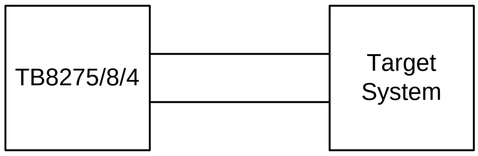
"Smart Option Source" Settings	Operating Mode	Comments
		The Smart Option is selected by external smart option switch (SW1)
		The Smart Option is selected by internal smart option area (003EH–003FH of ROM). But this selection is not available.

Table 20-4. Smart Option Switch Settings for TB8275/8/4

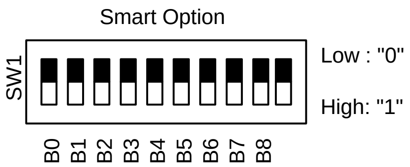
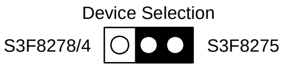
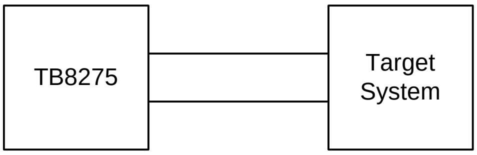
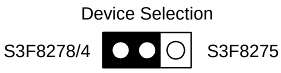
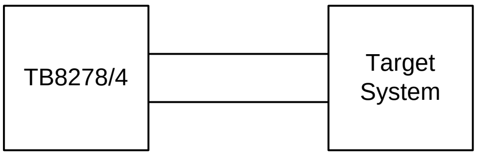
"Smart Option" Settings	Comments
	The Smart Option is selected by this switch when the Smart Option source is selected by external. The B2–B0 are comparable to the 003EH.2–.0. The B7–B5 are comparable to the 003EH.7–.5. The B8 is comparable to the 003FH.0. The B4–B3 is not connected.

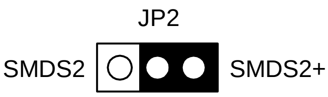
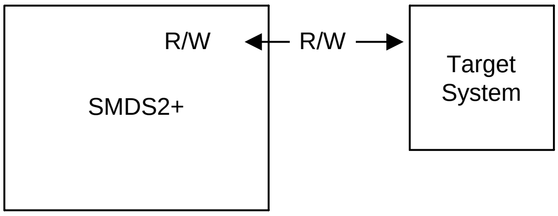
Table 20-5. Device Selection Settings for TB8275/8/4

"Device Selection" Settings	Operating Mode	Comments
		Operate with TB8275
		Operate with TB8278/4

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 20-6. The SMDS2+ Tool Selection Setting

"JP2" Setting	Operating Mode
	

IDLE LED

The Yellow LED is ON when the evaluation chip (S3E8270) is in idle mode.

STOP LED

The Red LED is ON when the evaluation chip (S3E8270) is in stop mode.

SAMSUNG ELECTRONICS

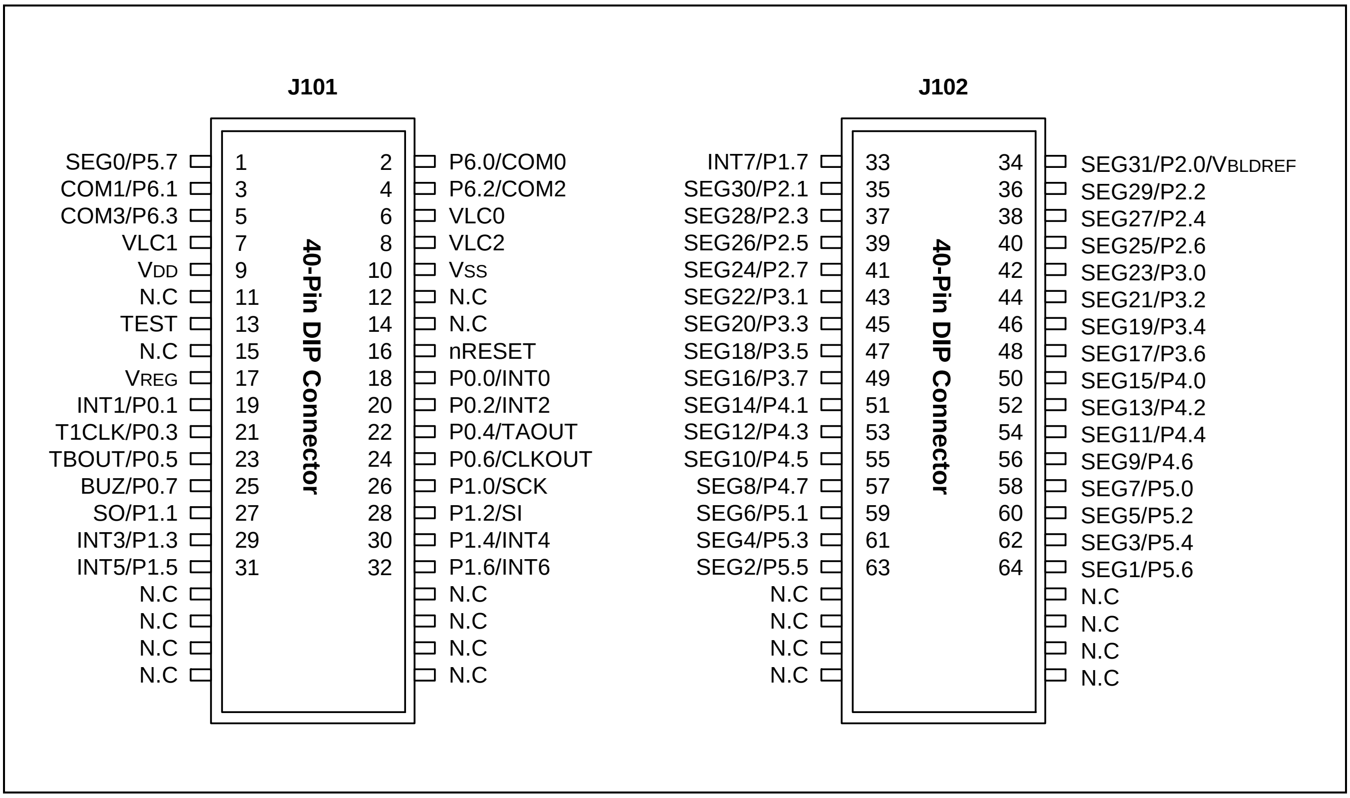


Figure 20-3. 40-Pin Connectors (J101, J102) for TB8275/8/4

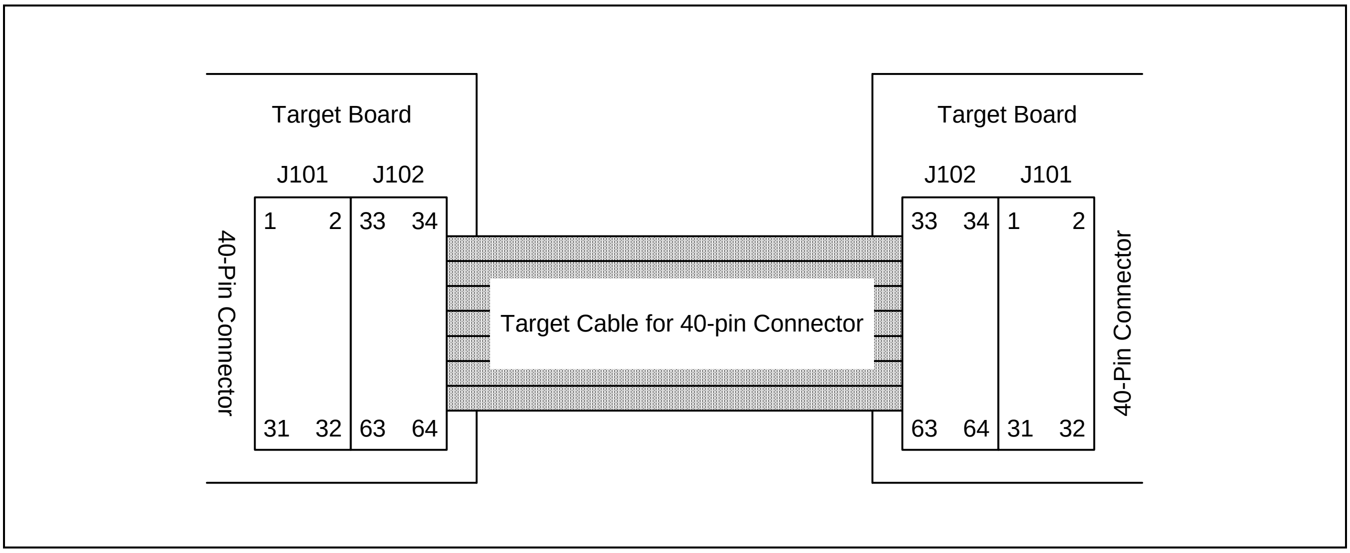


Figure 20-4. S3E8270 Cables for 64-QFP Package