# S3C828B/F828B /C8289/F8289 /C8285/F8285

## 8-BIT CMOS MICROCONTROLLERS USER'S MANUAL

### Revision 1.3

**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.

**S3C828B/F828B/C8289/F8289/C8285/F8285 8-Bit CMOS Microcontrollers**
**User's Manual, Revision 1.3**
**Publication Number: 21.3-S3-C828B/F828B/C8289/F8289/C8285/F8285-092005**

© 2005 Samsung Electronics

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

Samsung Electronics Co., Ltd.
San #24 Nongseo-Ri, Giheung- Eup
Yongin-City, Gyeonggi-Do, Korea
C.P.O. Box #37, Suwon 449-900

TEL: (82)-(031)-209-5238
FAX: (82)-(031)-209-6494

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

# NOTIFICATION OF REVISIONS

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

**PRODUCT NAME:** S3C828B/F828B/C8289/F8289/C8285/F8285 8-bit CMOS Microcontroller

**DOCUMENT NAME:** S3C828B/F828B/C8289/F8289/C8285/F8285 User's Manual, Revision 1.3

**DOCUMENT NUMBER:** 21.3-S3-C828B/F828B/C8289/F8289/C8285/F8285-092005

**EFFECTIVE DATE:** September, 2005

**SUMMARY:** As a result of additional product testing and evaluation, some specifications published in S3C828B/F828B/C8289/F8289/C8285/F8285 User's Manual, Revision 1.3, have been changed. These changes for in S3C828B/F828B/C8289/F8289/C8285/F8285 microcontroller, which are described in detail in the *Revision Descriptions* section below, are related to the followings:
— Chapter 19. Embedded flash memory interface
— Chapter 20. Electrical Data

**DIRECTIONS:** Please note the changes in your copy (copies) of the S3C828B/F828B/C8289 /F8289/C8285/F8285 User's Manual, Revision 1.3. Or, simply attach the *Revision Descriptions* of the next page to S3C828B/F828B/C8289/F8289/C8285/F8285 User's Manual, Revision 1.3.

# REVISION HISTORY

| Revision | Date | Remark |
|---|---|---|
| 0 | February, 2005 | Preliminary Spec for internal release only. |
| 1 | April, 2005 | First edition. |
| 1.1 | June, 2005 | Second edition. |
| 1.2 | July, 2005 | Third edition. |
| 1.3 | September, 2005 | Fourth edition. |

# REVISION DESCRIPTIONS

## 1. CHAPTER 20 ELECTRICAL DATA

**Table 20-15. Internal Flash ROM Electrical Characteristics (page 20-15)**

$(T_A = -25\ ^{\circ}C$ to $+ 85\ ^{\circ}C$, $V_{DD} = 2.0$ 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 | FN$_{WE}$ | – | – | – | 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 64K 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(S3F828B) and 100 times for half-flash (S3F8289/F8285).
5. The Chip Erasing is available in Tool Program Mode only.

**Table 20-6. A/D Converter Electrical Characteristics**

$(T_A = -25\ ^{\circ}C$ to $+ 85\ ^{\circ}C$, $V_{DD} = 2.7$ V to 3.6 V, $V_{SS} = 0$ V)

| Parameter | Symbol | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Analog input voltage | V$_{IAN}$ | – | V$_{SS}$ | – | AV$_{REF}$ | V |
| Analog input impedance | R$_{AN}$ | – | 2 | 1000 | – | MΩ |
| Analog reference voltage | AV$_{REF}$ | – | 2.0 | – | V$_{DD}$ | V |

## 2. CHAPTER 19. EMBEDDED FLASH MEMORY INTERFACE

This chapter is modified for only S3F828B.

## 3. CHAPTER 12. 16-BIT TIMER 0/1

The Figure12-2 condition 'Match signal' should be moved in the page 12-3.

# Preface

The S3C828B/F828B/C8289/F8289/C8285/F8285 *Microcontroller User's Manual* is designed for application designers and programmers who are using the S3C828B/F828B/C8289/F8289/C8285/F8285 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 S3C828B/F828B/C8289/F8289/C8285/F8285 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 S3C828B/F828B/C8289/F8289/C8285/F8285 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 S3C828B/F828B/C8289/F8289/C8285/F8285 microcontroller. Also included in Part II are electrical, mechanical, Flash, and development tools data. It has 17 chapters:

| | | | |
|---|---|---|---|
| Chapter 7 | Clock Circuit | Chapter 16 | Serial I/O Interface |
| Chapter 8 | RESET and Power-Down | Chapter 17 | UART |
| Chapter 9 | I/O Ports | Chapter 18 | Battery Level Detector |
| Chapter 10 | Basic Timer | Chapter 19 | Embedded Flash Memory |
| Chapter 11 | 8-bit Timer A/B | Chapter 20 | Electrical Data |
| Chapter 12 | 16-bit Timer 0/1 | Chapter 21 | Mechanical Data |
| Chapter 13 | Watch Timer | Chapter 22 | S3F828B/F8289/F8285 Flash MCU |
| Chapter 14 | LCD Controller/Driver | Chapter 23 | Development Tools |
| Chapter 15 | 10-bit-to-Digital Converter | | |

Two order forms are included at the back of this manual to facilitate customer order for S3C828B/F828B/C8289/ F8289/C8285/F8285 microcontrollers: the Mask ROM Order Form, and the Mask Option Selection Form. You can photocopy these forms, fill them out, and then forward them to your local Samsung Sales Representative.

# Table of Contents

# Part I — Programming Model

## Chapter 1        Product Overview

## Chapter 2        Address Spaces

# Table of Contents (Continued)

# 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)

# Table of Contents (Continued)

# Table of Contents (Concluded)

# List of Figures

# List of Figures (Continued)

# List of Figures (Continued)

# List of Figures (Concluded)

# List of Tables

# List of Tables (Continued)

# List of Programming Tips

# List of Register Descriptions

# List of Register Descriptions (Continued)

# List of Instruction Descriptions

# List of Instruction Descriptions (Continued)

# 1   PRODUCT OVERVIEW

## S3C8-SERIES MICROCONTROLLERS

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

— Efficient register-oriented architecture

— Selectable CPU clock sources

— Idle and Stop power-down mode release by interrupt

— Built-in basic timer with watchdog function

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

## S3C828B/F828B/C8289/F8289/C8285/F8285 MICROCONTROLLER

The S3C828B/F828B/C8289/F8289/C8285/F8285 single-chip CMOS microcontroller are fabricated using the highly advanced CMOS process, based on Samsung's newest CPU architecture.

The S3C828B, S3C8289, S3C8285 are a microcontroller with a 64K-byte, 32K-byte, 16K-byte mask-programmable ROM embedded respectively.

The S3F828B is a microcontroller with a 64K-byte Flash ROM embedded.
The S3F8289 is a microcontroller with a 32K-byte Flash ROM embedded.
The S3F8285 is a microcontroller with a 16K-byte Flash ROM embedded.

Using a proven modular design approach, Samsung engineers have successfully developed the S3C828B/F828B/C8289/F8289/C8285/F8285 by integrating the following peripheral modules with the powerful SAM8 core:

— Nine programmable I/O ports, including six 8-bit ports, and one 7-bit port, one 6-bit port, one 4-bit port, for a total of 65 pins.

— Eight bit-programmable pins for external interrupts.

— One 8-bit basic timer for oscillation stabilization and watchdog functions (system reset).

— Two 8-bit timer/counter and two 16-bit timer/counter with selectable operating modes.

— Watch timer for real time.

— LCD Controller/driver

— A/D converter with 8 selectable input pins

— Synchronous SIO modules

The S3C828B/F828B/C8289/F8289/C8285/F8285 is versatile microcontroller for camera, LCD and ADC application, etc. They are currently available in 80-pin TQFP and 80-pin QFP package

## FLASH

The S3F828B/F8289/F8285 are FLASH version of the S3C828B/C8289/C8285 microcontroller. The S3F828B microcontroller has an on-chip FLASH ROM instead of a masked ROM. The S3F828B/F8289/F8285 is comparable to the S3C828B/C8289/C8285, both in function and in pin configuration. The S3F828B only is a full flash. The full flash means that data can be written into the program ROM by an instruction.

**SAMSUNG**
ELECTRONICS

# FEATURES

## CPU

- SAM88 RC CPU core

## Memory

- Program Memory (ROM)
  - 64K × 8 bits program memory
    (S3C828B/F828B)
  - 32K × 8 bits program memory
    (S3C8289/F8289)
  - 16K × 8 bits program memory
    (S3C8285/F8285)
  - Internal flash memory (program memory)
    - √ Sector size: 128 bytes
    - √ 10 years data retention
    - √ Fast programming time:
      + chip erase: 50ms
      + sector erase: 10ms
      + byte program: 30μs

    - √ User programmable by 'LDC' instruction
    - √ Endurance: 10,000 erase/program cycles
    - √ Sector(128 bytes) erase available
    - √ Byte programmable
    - √ External serial programming support
    - √ Expandable OBPTM(on board program)
      sector
- Data Memory (RAM)
  - Including LCD display data memory
  - 2614 × 8 bits data memory (S3C828B/F828B)
  - 1078 × 8 bits data memory (S3C8289/F8289)
  - 566 × 8 bits data memory (S3C8285/F8285)

## Instruction Set

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

## 65 I/O Pins

- I/O: 25 pins
- I/O: 40 pins(Sharing with LCD signal outputs)

## Interrupts

- 8 interrupt levels and 18 interrupt sources
- Fast interrupt processing feature

## 8-Bit Basic Timer

- Watchdog timer function
- 4 kinds of clock source

## 8-Bit Timer/Counter A

- Programmable 8-bit internal timer
- External event counter function
- PWM and capture function

## 8-Bit Timer/Counter B

- Programmable 8-bit internal timer
- Carrier frequency generator

## 16-Bit Timer/Counter 0

- Programmable 16-bit internal timer

## 16-Bit Timer/Counter 1

- Programmable 16-bit internal timer
- External event counter function
- PWM and capture function

## 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 8 common terminals
- 1/2, 1/3, 1/4, and 1/8 duty selectable
- Internal resistor circuit for LCD bias

## Analog to Digital Converter

- 8-channel analog input
- 10-bit conversion resolution
- 25uS conversion time

## UART

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

## 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

**SAMSUNG**
ELECTRONICS

## FEATURES (Continued)

**Battery Level Detector**

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

**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 – 11.1MHz
- 32.768 kHz crystal oscillation circuit for
  sub clock

**Instruction Execution Times**

- 360nS at 11.1 MHz fx(minimum)

**Operating Voltage Range**

- 2.0 V to 3.6 V at 0.4 – 4.2MHz
- 2.7 V to 3.6 V at 0.4 – 10.0MHz
- 3.0 V to 3.6 V at 0.4 – 11.1MHz

**Operating Temperature Range**

- -25°C to +85°C

**Package Type**

- 80-QFP-1420C, 80-TQFP-1212

**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)

SAMSUNG
ELECTRONICS
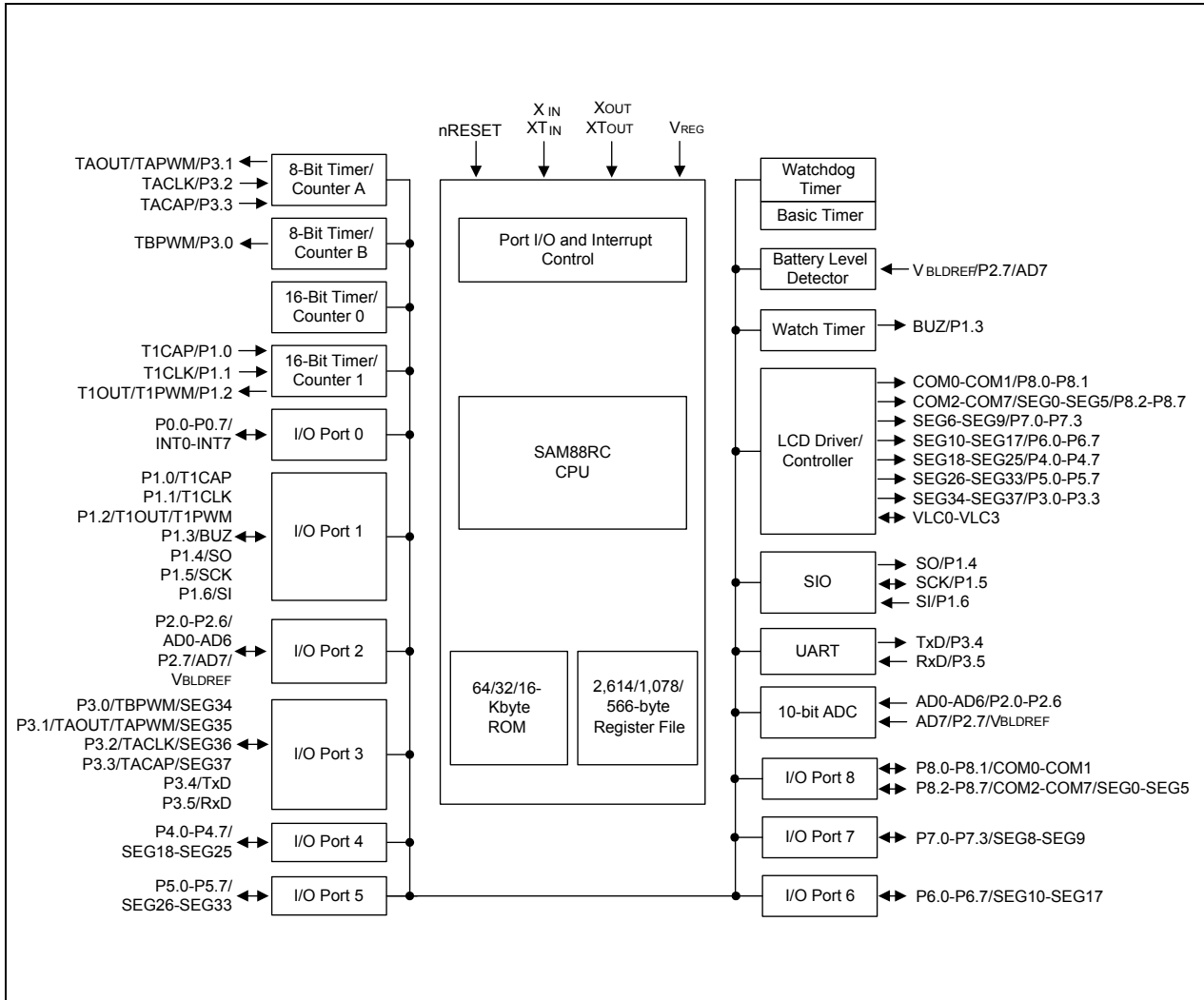
# BLOCK DIAGRAM



**Figure 1-1. Block Diagram**

**SAMSUNG**
**ELECTRONICS**
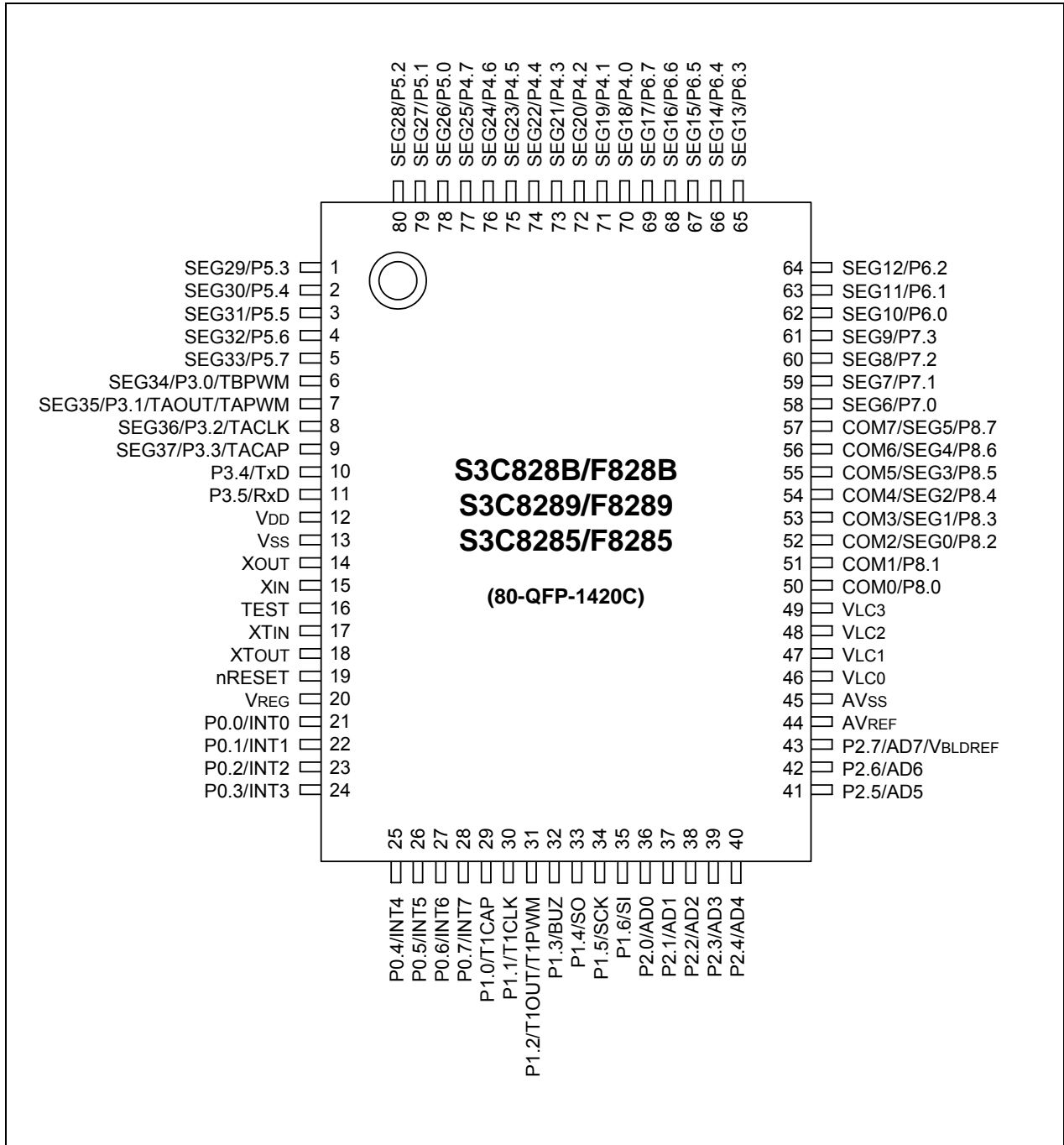
## PIN ASSIGNMENT



Figure 1-2. S3C828B/F828B/C8289/F8289/C8285/F8285 Pin Assignments (80-QFP-1420C)
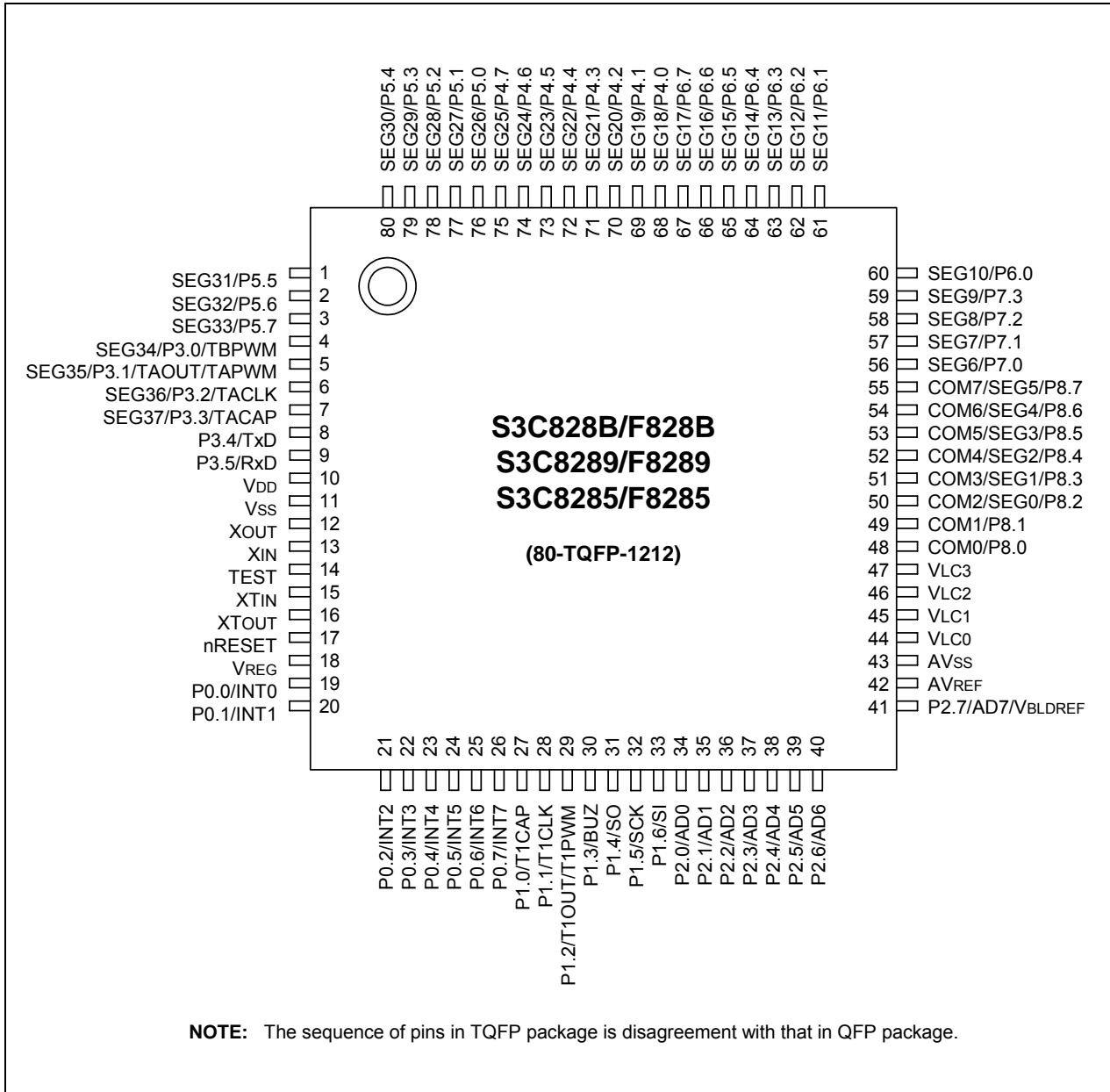
**Figure 1-3. S3C828B/F828B/C8289/F8289/C8285/F8285 Pin Assignments (80-TQFP-1212)**

## PIN DESCRIPTIONS

**Table 1-1. S3C828B/F828B/C8289/F8289/C8285/F8285 Pin Descriptions**

| Pin Names | Pin Type | Pin Description | Circuit Type | Pin Numbers [note] | Share Pins |
|---|---|---|---|---|---|
| P0.0–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–P0.7 are alternately used for external interrupt input(noise filters, interrupt enable and pending control). | E-4 | 21–28 (19–26) | INT0–INT7 |
| P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 | I/O | I/O port with bit-programmable pins; Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups. | E-4 | 29(27) 30(28) 31(29) 32(30) 33(31) 34(32) 35(33) | T1CAP T1CLK T1OUT/T1PWM BUZ SO SCK SI |
| P2.0–P2.6 <br><br> P2.7 | I/O | I/O port with bit-programmable pins; Input or push-pull output and software assignable pull-ups. | F-1 <br><br> F-2 | 36–42 (34–40) <br> 43(41) | AD0–AD6 <br><br> AD7/$V_{BLDREF}$ |
| P3.0 P3.1 <br><br> P3.2 P3.3 P3.4 P3.5 | I/O | I/O port with bit-programmable pins; Input or push-pull output and software assignable pull-ups. | H-9 <br><br><br><br> D-1 | 6(4) 7(5) <br><br> 8(6) 9(7) 10(8) 11(9) | TBPWM/SEG34 TAOUT/TAPWM /SEG35 TACLK/SEG36 TACAP/SEG37 TxD RxD |
| P4.0–P4.7 | I/O | I/O port with bit-programmable pins; Input or push-pull, open-drain output and software assignable pull-ups. | H-8 | 70– 77 (68–75) | SEG18–SEG25 |
| P5.0–P5.7 | I/O | I/O port with bit-programmable pins; Input or push-pull, open-drain output and software assignable pull-ups. | H-8 | 78–80,1–5 (76–80,1–3) | SEG26–SEG33 |
| P6.0–P6.7 | I/O | I/O port with bit-programmable pins; Input or push-pull output and software assignable pull-ups. | H-9 | 62–69 (60–67) | SEG10–SEG17 |
| P7.0–P7.3 | I/O | I/O port with bit-programmable pins; Input or push-pull output and software assignable pull-ups. | H-9 | 58–61 (56–59) | SEG6–SEG9 |
| P8.0–P8.1 P8.2–P8.7 | I/O | I/O port with bit-programmable pins; Input or push-pull output and software assignable pull-ups. | H-9 | 50–51(48–49) 52–57(50–55) | COM0–COM1 COM2–COM7/ SEG0–SEG5 |
| $V_{LC0}$–$V_{LC3}$ | – | LCD power supply pins. | – | 46–49 (44–47) | – |

**Table 1-1. S3C828B/F828B/C8289/F8289/C8285/F8285 Pin Descriptions (Continued)**

| Pin Names | Pin Type | Pin Description | Circuit Type | Pin Numbers | Share Pins |
|---|---|---|---|---|---|
| INT0–INT7 | I/O | External interrupts input pins. | E-4 | 21–28(19–26) | P0.0–P0.7 |
| T1CAP | I/O | Timer 1 capture input. | E-4 | 29(27) | P1.0 |
| T1CLK | I/O | Timer 1 external clock input. | E-4 | 30(28) | P1.1 |
| T1OUT/T1PWM | I/O | Timer 1 clock output and PWM output. | E-4 | 31(29) | P1.2 |
| BUZ | I/O | Output pin for buzzer signal. | E-4 | 32(30) | P1.3 |
| SO, SCK, SI | I/O | Serial clock, serial data output, and serial data input. | E-4 | 33–35 (31–33) | P1.4, P1.5, P1.6 |
| AD0–AD6 | I/O | A/D converter analog input channels. | F-1 | 36–42 (34–40) | P2.0–P2.6 |
| AD7 | | | F-2 | 43(41) | P2.7/$V_{BLDREF}$ |
| AVREF | – | A/D converter reference voltage. | – | 44(42) | – |
| AVSS | – | A/D converter ground. | – | 45(43) | – |
| VBLDREF | – | Battery level detector reference voltage. | F-2 | 43(41) | P2.7/AD7 |
| TACAP | I/O | Timer A capture input. | D-1 | 9(7) | P3.3 |
| TACLK | I/O | Timer A external clock input. | D-1 | 8(6) | P3.2 |
| TAOUT/TAPWM | I/O | Timer A clock output and PWM output. | D-1 | 7(5) | P3.1 |
| TBPWM | I/O | Timer B PWM output. | D-1 | 6(4) | P3.0 |
| TxD, RxD | I/O | Uart data output, input | D-1 | 10,11(8,9) | P3.4, P3.5 |
| COM0–COM1 COM2–COM7 | I/O | LCD common signal outputs. | H-9 | 50–51(48–49) 52–57(50–55) | P8.0–P8.1 P8.2–P8.7/ SEG0-SEG5 |
| SEG0–SEG5 | I/O | LCD segment signal outputs. | H-9 | 52–57(50–55) | COM2–COM7/ P8.2–P8.7 |
| SEG6–SEG9 SEG10–SEG17 SEG18–SEG25 SEG26–SEG33 SEG34 SEG35 SEG36 SEG37 | | | H-8 H-9 | 58–61(56–59) 62–69(60–67) 70–77(68–75) 78-80,1-5 (76-80,1-3) 6(4) 7(5) 8(6) 9(7) | P7.0–P7.3 P6.0–P6.7 P4.0–P4.7 P5.0–P5.7 P3.0/TBPWM P3.1/TAOUT/TAPWM P3.2/TACLK P3.3/TACAP |
| $V_{REG}$ | O | Regulator voltage output for sub clock (needed 0.1μF) | – | 20(18) | – |
| nRESET | I | System reset pin | B | 19(17) | – |
| $XT_{IN}$, $XT_{OUT}$ | – | Crystal oscillator pins for sub clock. | – | 17,18(15,16) | – |
| $X_{IN}$, $X_{OUT}$ | – | Main oscillator pins. | – | 15,14(13,12) | – |
| TEST | I | Test input: it must be connected to $V_{SS}$ | – | 16(14) | – |
| $V_{DD}$, $V_{SS}$ | – | Power input pins. A capacitor must be connected between $V_{DD}$ and $V_{SS}$. | – | 12,13(10,11) | – |

**NOTE:** Parentheses indicate pin number for 80-TQFP-1212 package.

SAMSUNG
ELECTRONICS
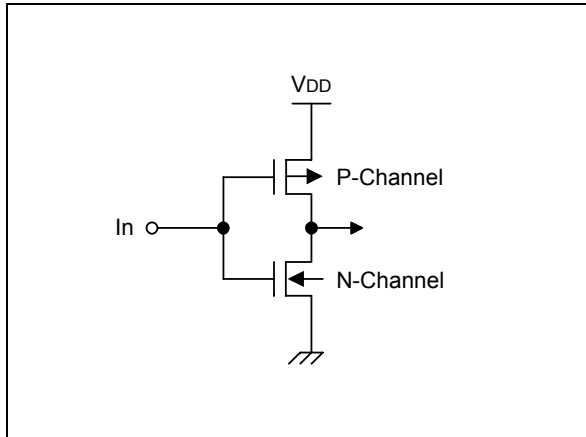
## PIN CIRCUITS



**Figure 1-4. Pin Circuit Type A**
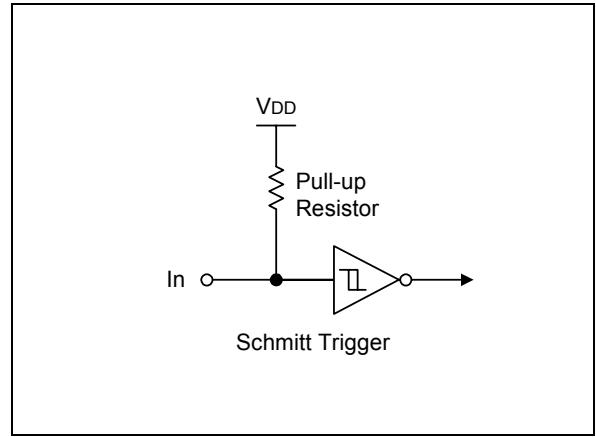


**Figure 1-5. Pin Circuit Type B**
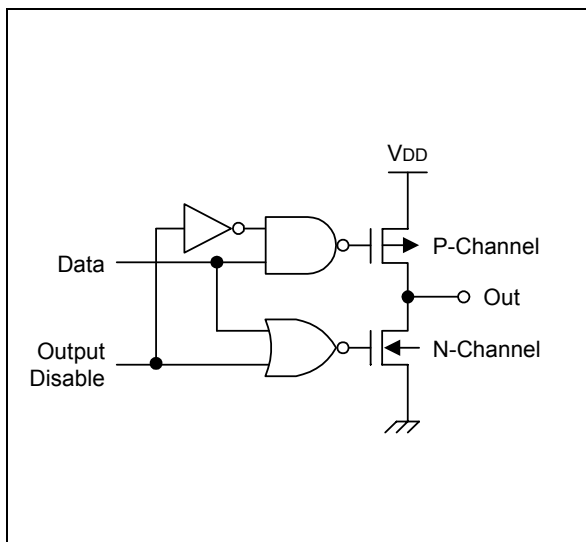


**Figure 1-6. Pin Circuit Type C**
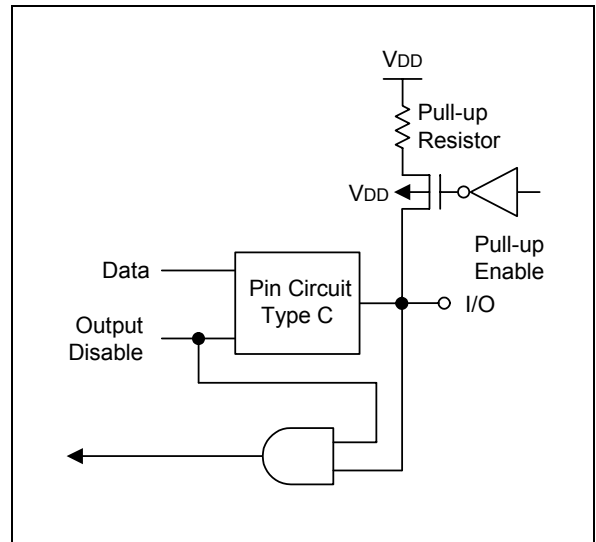


**Figure 1-7. Pin Circuit Type D-1 (P3.4, P3.5)**
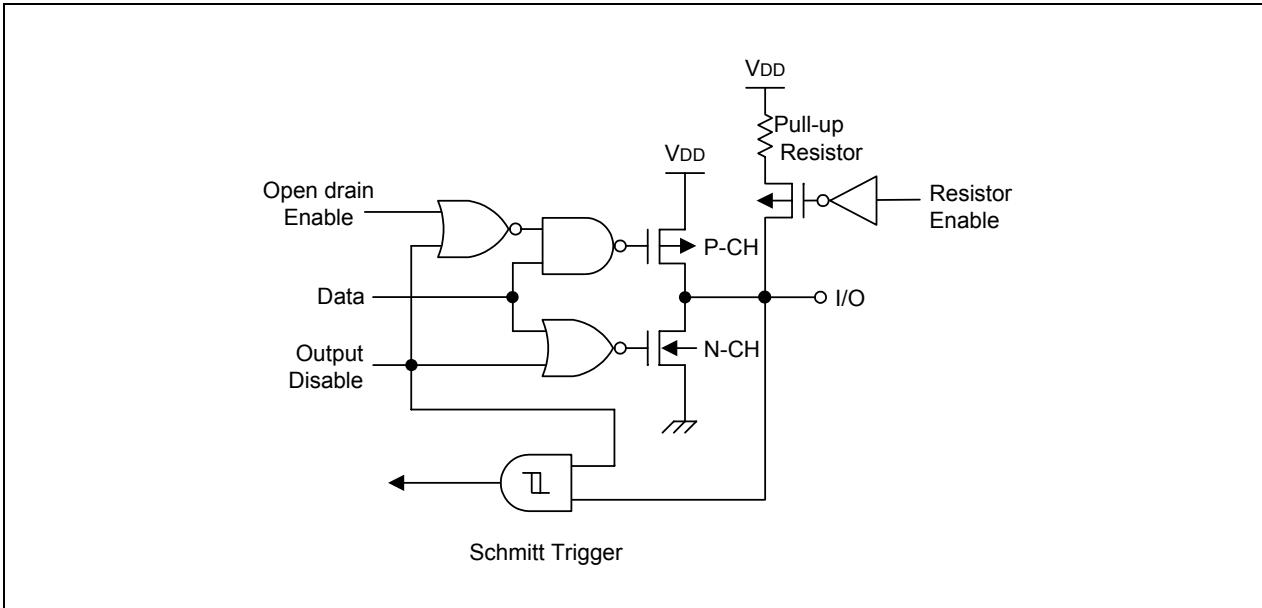
**Figure 1-8. Pin Circuit Type E-4 (P0, P1)**
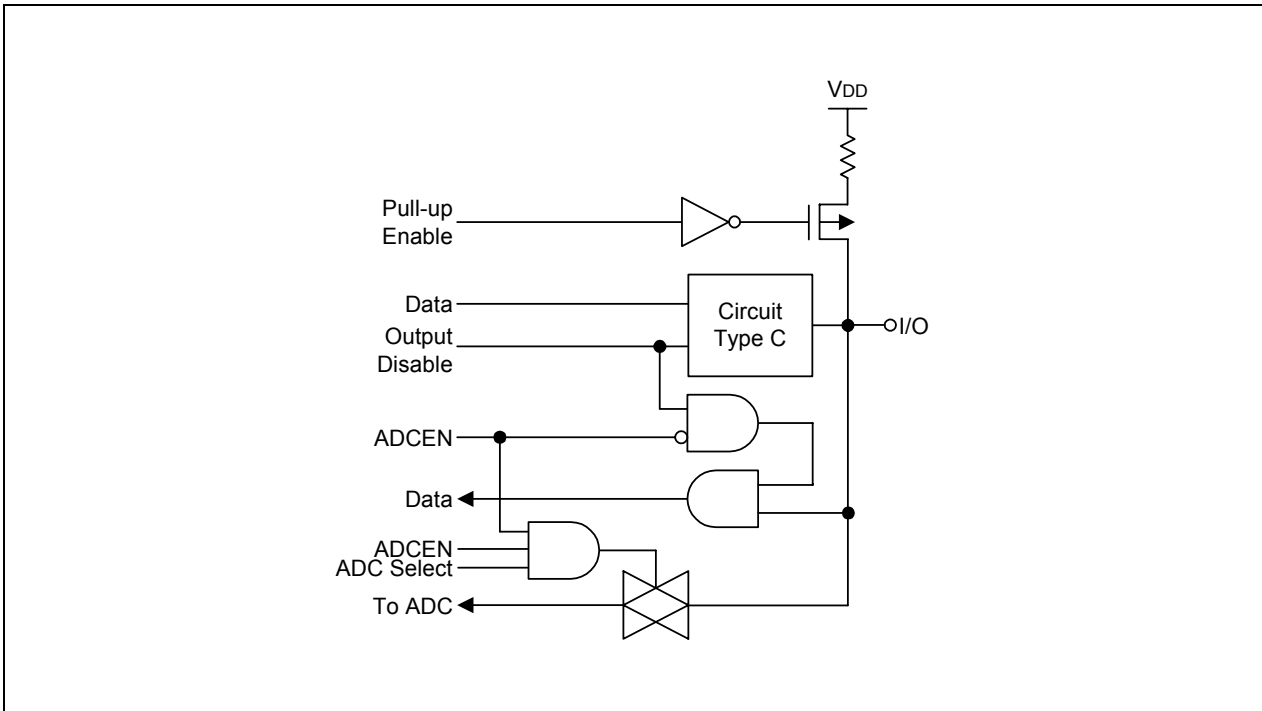


**Figure 1-9. Pin Circuit Type F-1 (P2.0–P2.6)**
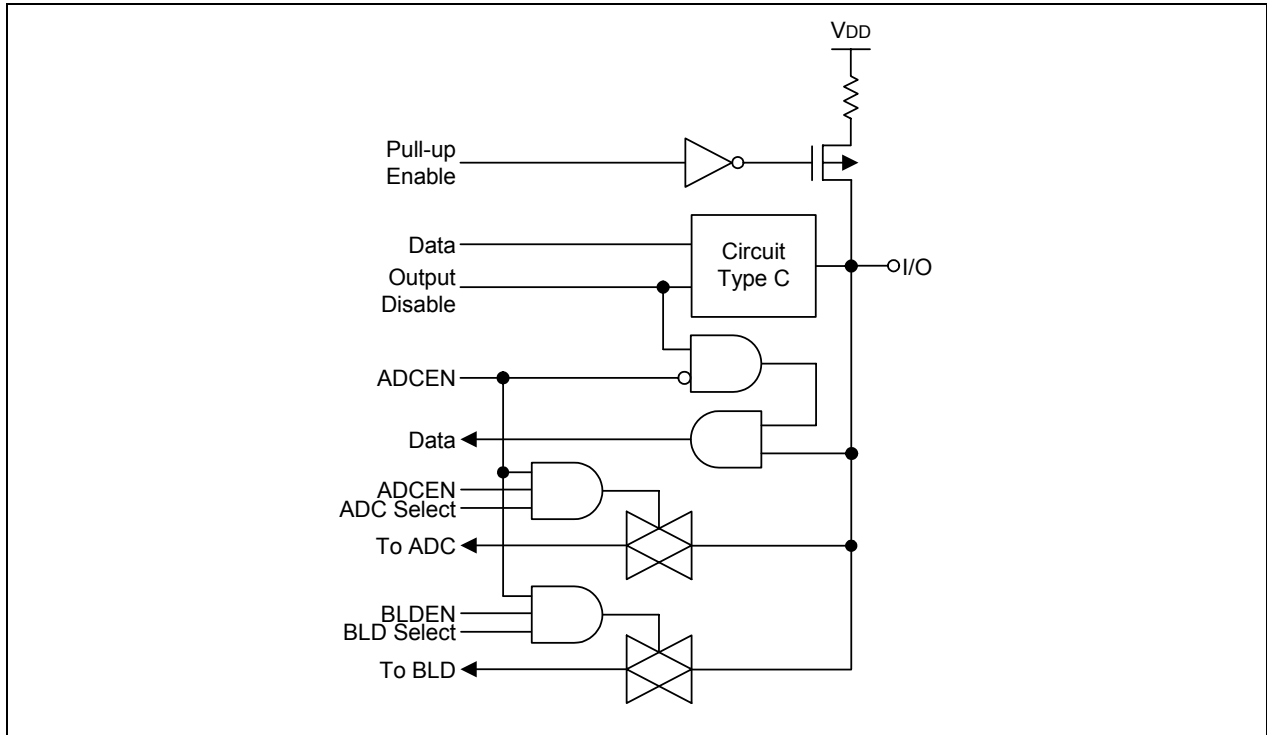
Figure 1-10. Pin Circuit Type F-2 (P2.7)
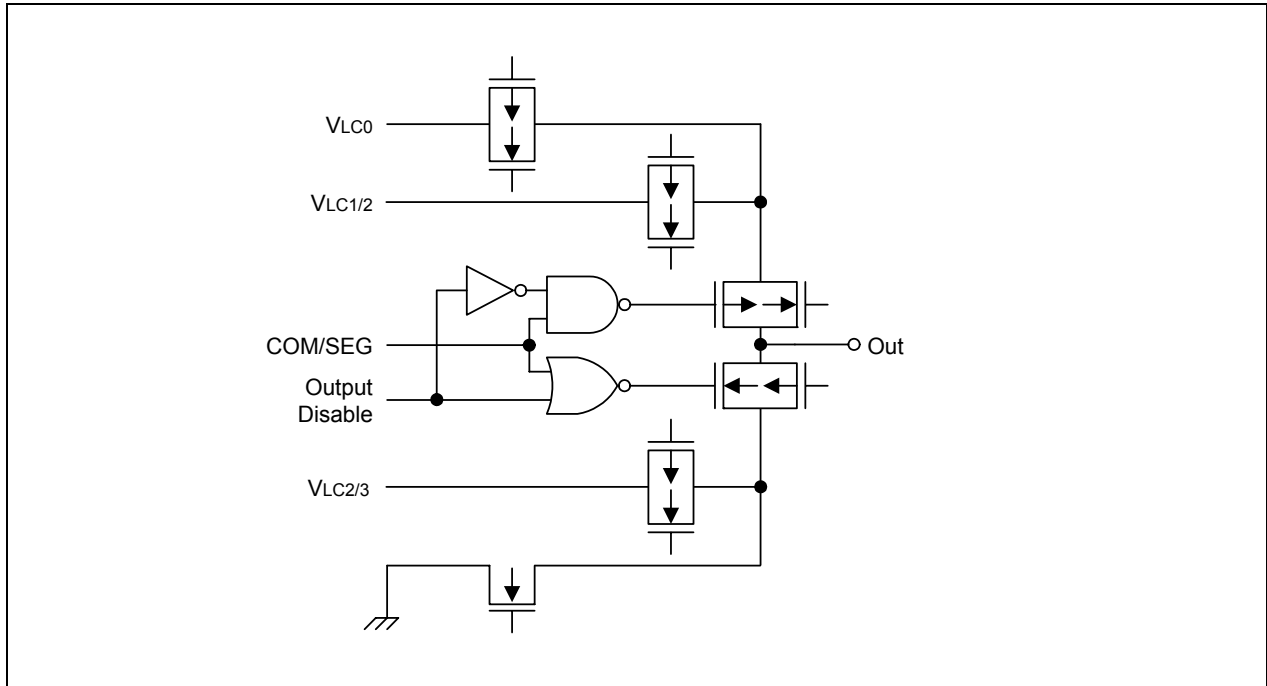


Figure 1-11. Pin Circuit Type H-4
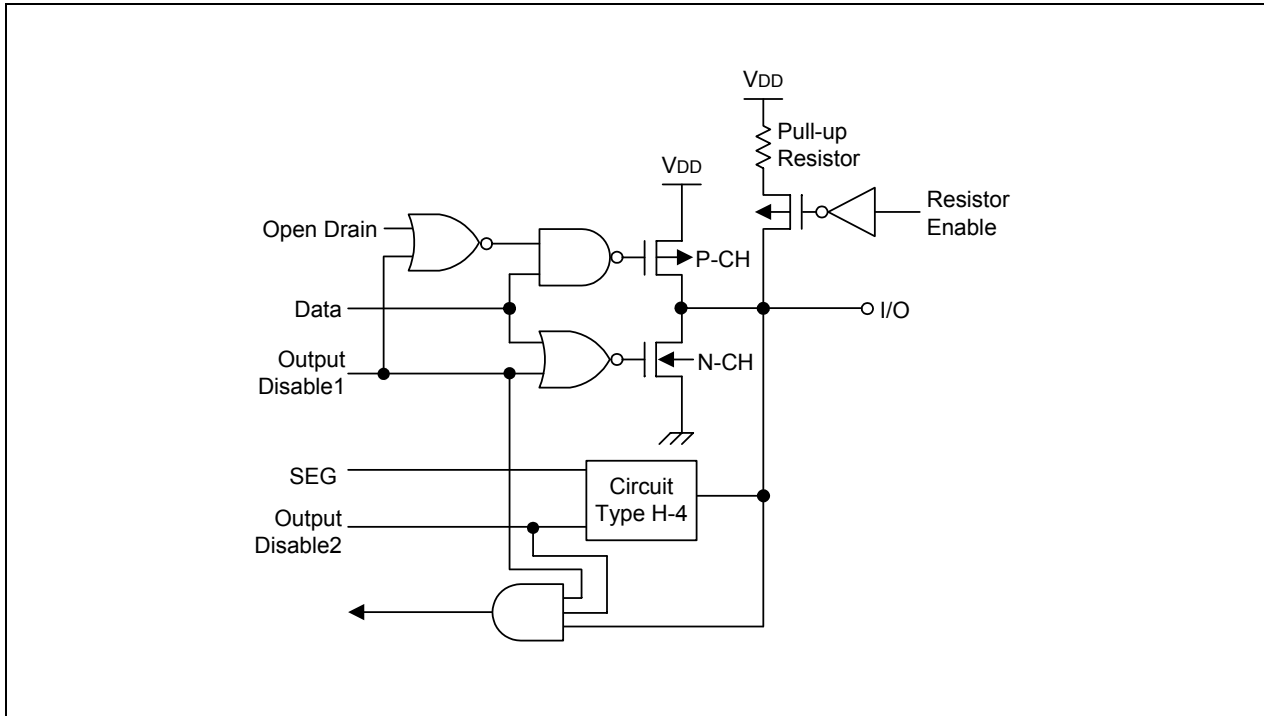
**Figure 1-12. Pin Circuit Type H-8 (P4, P5)**
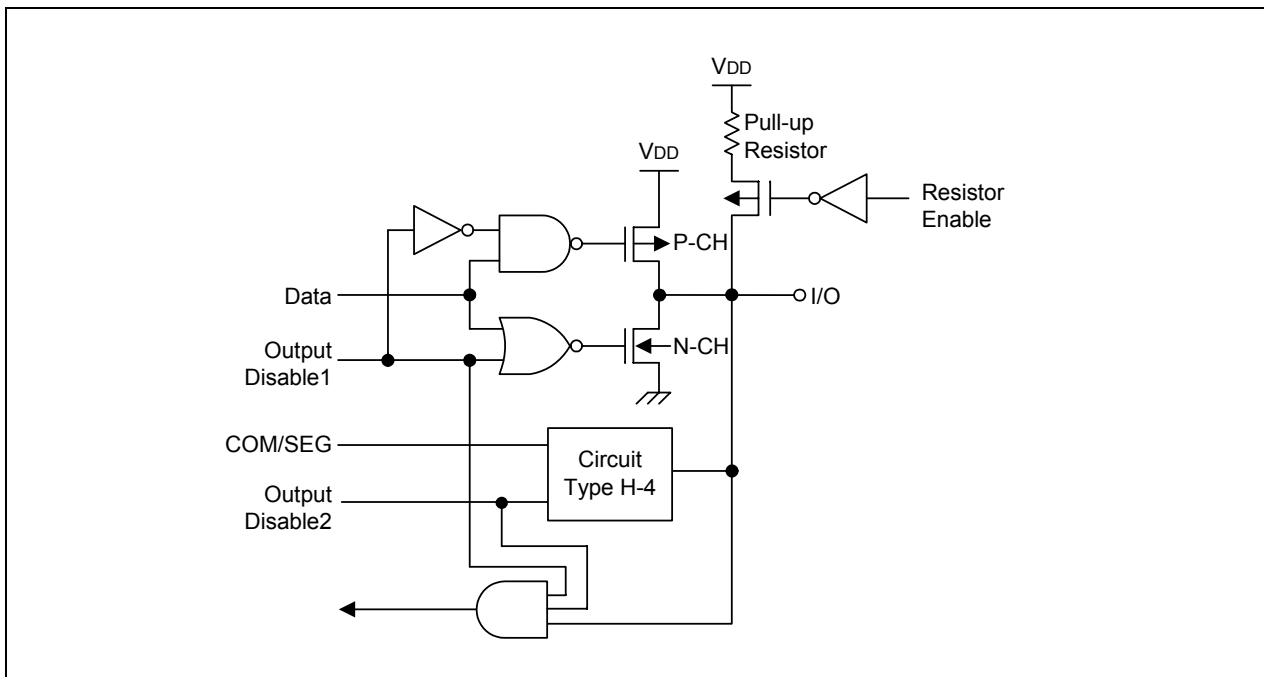


**Figure 1-13. Pin Circuit Type H-9 (P3.0-P3.3, P6, P7, P8)**

# 2 ADDRESS SPACES

## OVERVIEW

The S3C828B/C8289/C8285 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 S3C828B has an internal 64-Kbyte mask-programmable ROM. The S3C8289 has an internal 32-Kbyte mask-programmable ROM. The S3C8285 has an internal 16-Kbyte mask-programmable ROM.

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

A 38-byte LCD display register file is implemented.

## PROGRAM MEMORY (ROM)

Program memory (ROM) stores program codes or table data. The S3C828B/F828B has 64K bytes internal mask-programmable program memory, the S3C8289/F8289 has 32K bytes and the S3C8285/F8285 has 16K 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 in the S3C828B/C8289/C8285.

The reset address of ROM can be changed by a smart option only in the S3F828B(Full-Flash Device). Refer to the chapter 19. Embedded Flash Memory Interface for more detail contents.
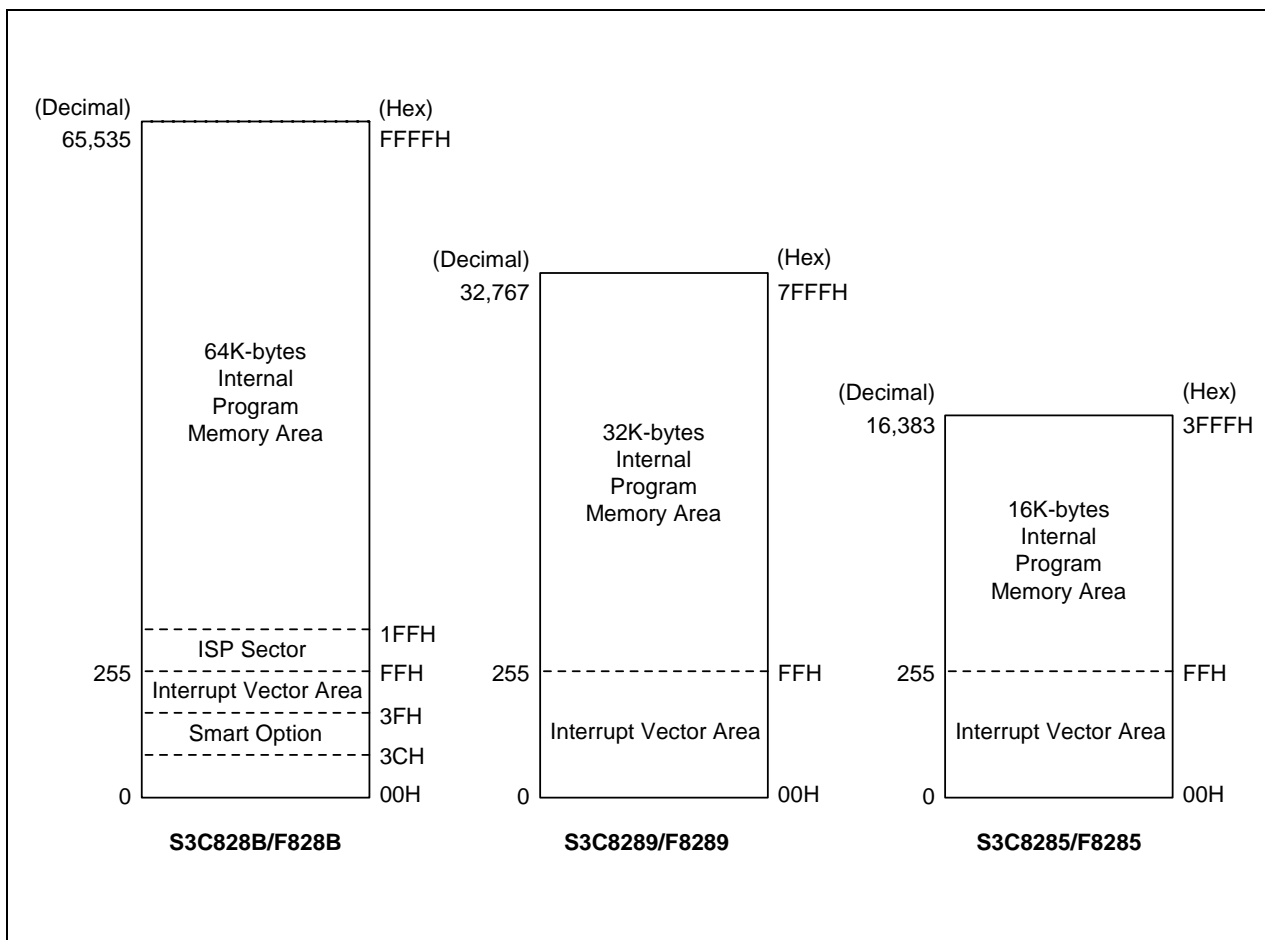


**Figure 2-1. Program Memory Address Space**

## 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 bits:(note)**
00 = 256 bytes
01 = 512 bytes
10 = 1024 bytes
11 = 2048 bytes

**ISP reset vector address selection bits:**
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

**NOTE:**
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**

# REGISTER ARCHITECTURE

In the S3C828B/F828B/C8289/F8289/C8285/F8285 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 S3C828B/F828B the total number of addressable 8-bit registers is 2,695. Of these 2,695 registers, 13 bytes are for CPU and system control registers, 38 bytes are for LCD data registers, 68 bytes are for peripheral control and data registers, 16 bytes are used as a shared working registers, and 2,560 registers are for general-purpose use, page 0-page 9 (in case of S3C8289/F8289, page 0-page 3 and S3C8285/F8285, page0-page1).

You can always address set 1 register locations, regardless of which of the ten 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. S3C828B/F828B Register Type Summary**

| Register Type | Number of Bytes |
|---|---|
| General-purpose registers (including the 16-byte common working register area, ten 192-byte prime register area, and ten 64-byte set 2 area) | 2,576 |
| LCD data registers | 38 |
| CPU and system control registers | 13 |
| Mapped clock, peripheral, I/O control, and data registers | 68 |
| **Total Addressable Bytes** | 2,695 |

**SAMSUNG**
**ELECTRONICS**

**Table 2-2. S3C8289/F8289 Register Type Summary**

| Register Type | Number of Bytes |
|---|---|
| General-purpose registers (including the 16-byte common working register area, four 192-byte prime register area, and four 64-byte set 2 area)<br>LCD data registers<br>CPU and system control registers<br>Mapped clock, peripheral, I/O control, and data registers | 1,040<br><br><br>38<br>13<br>68 |
| **Total Addressable Bytes** | 1,159 |

**Table 2-3. S3C8285/F8285 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)<br>LCD data registers<br>CPU and system control registers<br>Mapped clock, peripheral, I/O control, and data registers | 528<br><br><br>38<br>13<br>68 |
| **Total Addressable Bytes** | 647 |
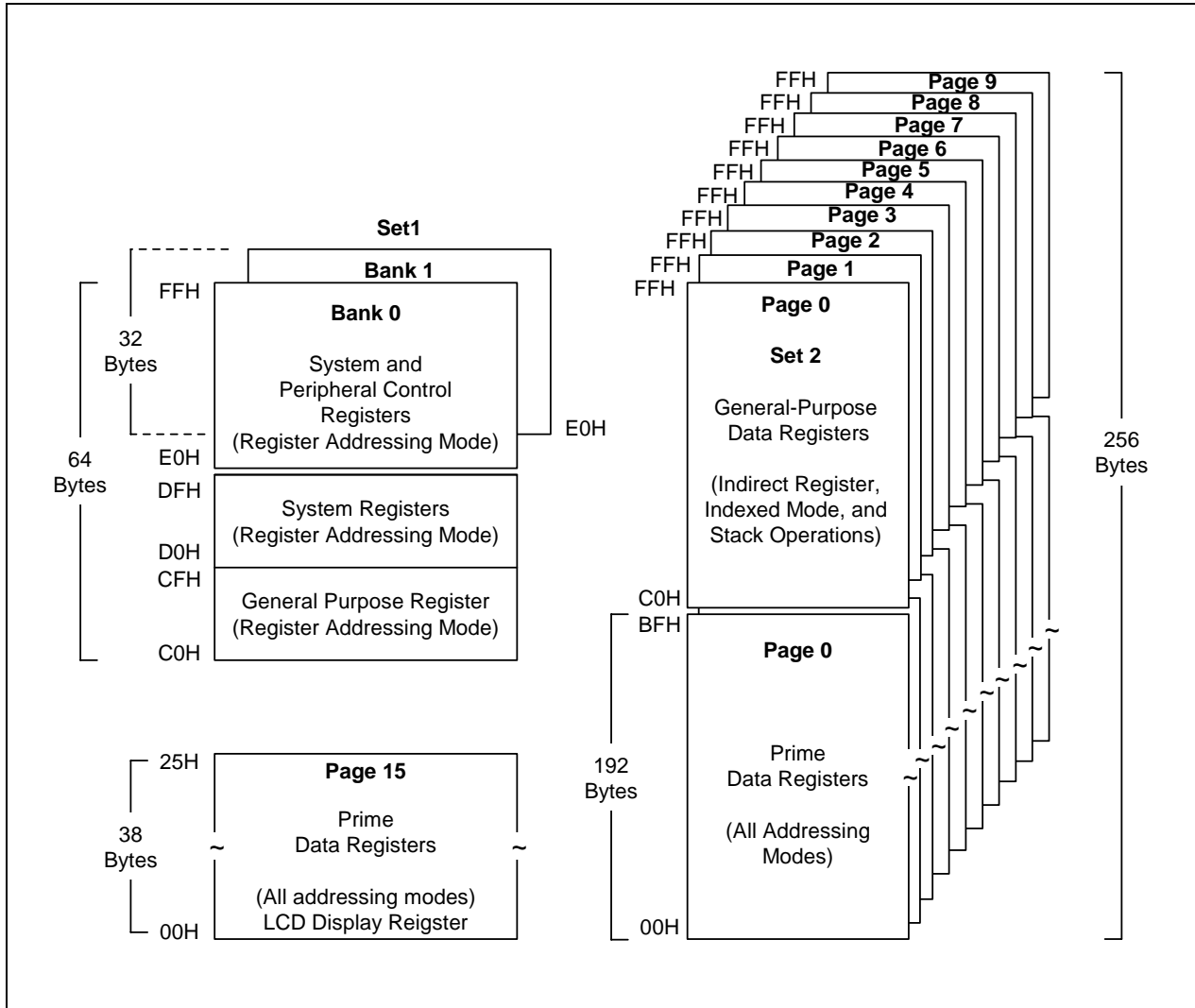
**Figure 2-3. Internal Register File Organization (S3C828B/F828B)**
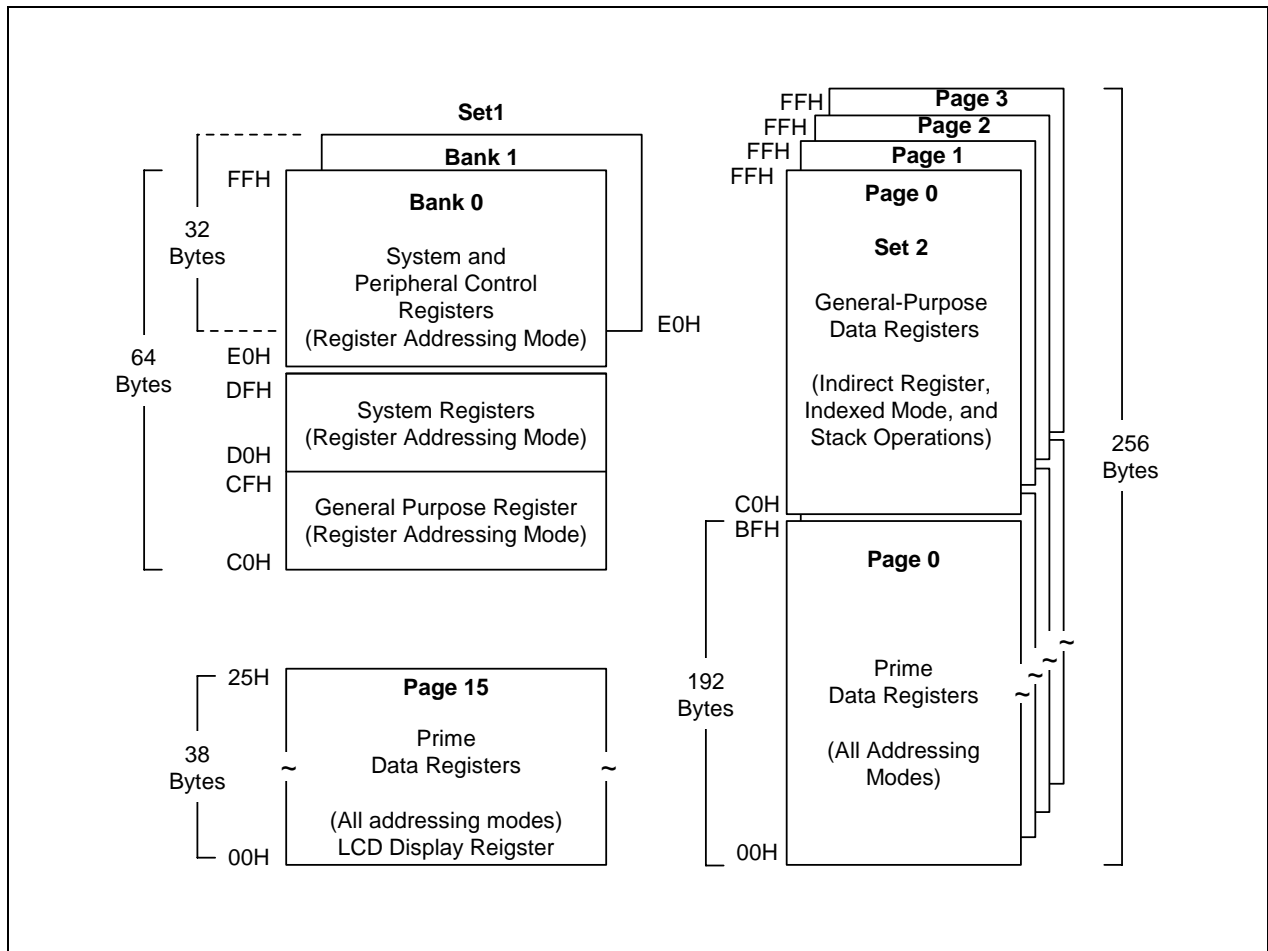
**Figure 2-4. Internal Register File Organization (S3C8289/F8289)**
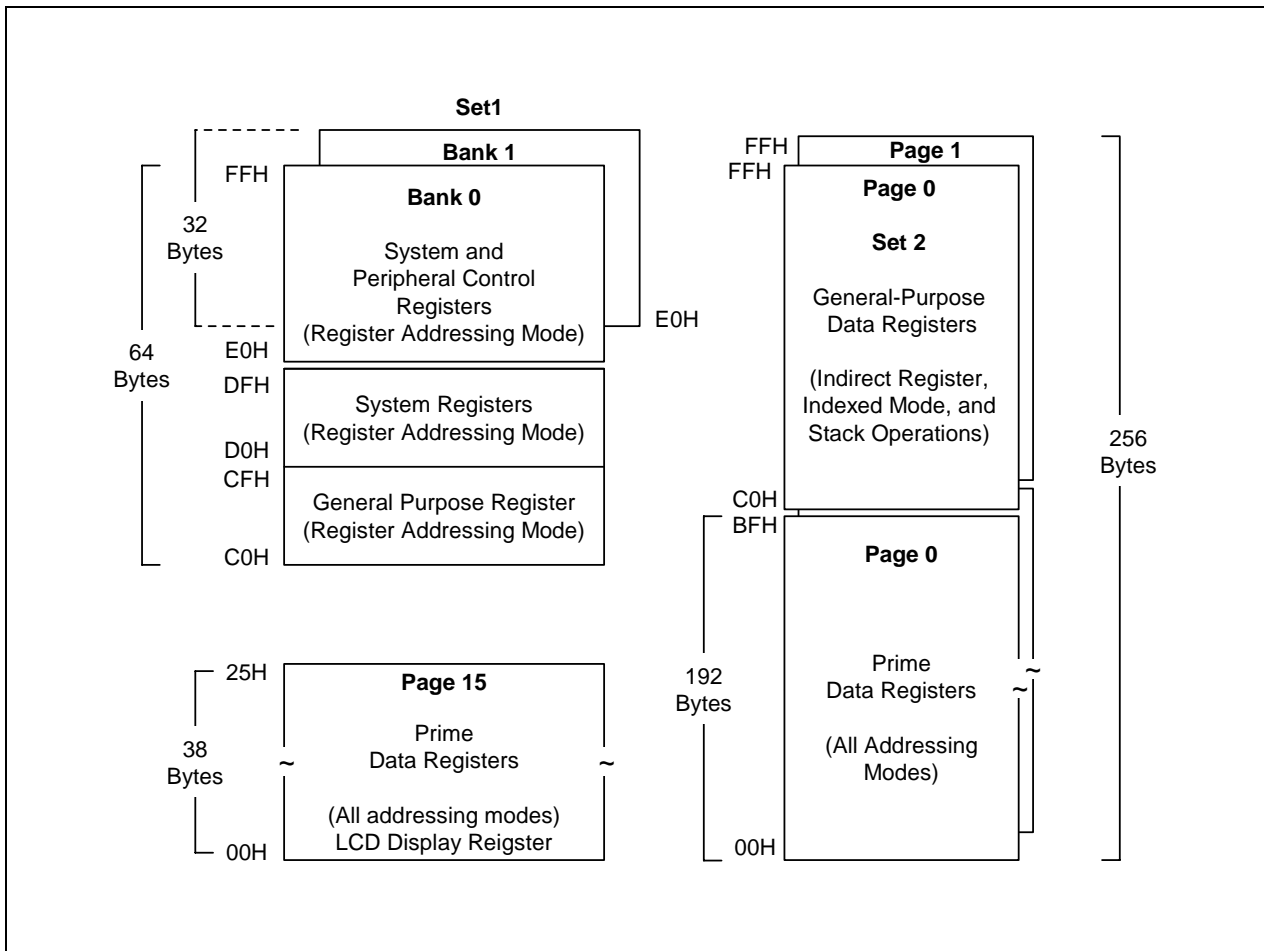
**Figure 2-5. Internal Register File Organization (S3C8285/F8285)**

## 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 S3C828B/C8289/C8285 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

Destination register page selection bits:

| 0000 | Destination: Page 0 |
| 0001 | Destination: Page 1 |
| 0010 | Destination: Page 2 (Not used for the S3C8285) |
| 0011 | Destination: Page 3 (Not used for the S3C8285) |
| 0100 | Destination: Page 4 (Not used for the S3C8289/5) |
| 0101 | Destination: Page 5 (Not used for the S3C8289/5) |
| 0110 | Destination: Page 6 (Not used for the S3C8289/5) |
| 0111 | Destination: Page 7 (Not used for the S3C8289/5) |
| 1000 | Destination: Page 8 (Not used for the S3C8289/5) |
| 1001 | Destination: Page 9 (Not used for the S3C8289/5) |
| 1111 | Destination: Page 15 |
| Others | Not used for the S3C828B/9/5 |

Source register page selection bits:

| 0000 | Source: page 0 |
| 0001 | Source: page 1 |
| 0010 | Source: page 2 (not used for the S3C8285) |
| 0011 | Source: page 3 (not used for the S3C8285) |
| 0100 | Source: page 4 (not used for the S3C8289/5) |
| 0101 | Source: page 5 (not used for the S3C8289/5) |
| 0110 | Source: page 6 (not used for the S3C8289/5) |
| 0111 | Source: page 7 (not used for the S3C8289/5) |
| 1000 | Source: page 8 (not used for the S3C8289/5) |
| 1001 | Source: page 9 (not used for the S3C8289/5) |
| 1111 | Source: page 15 |
| Others | Not used for the S3C828B/9/5 |

**NOTES:**
1. In the S3C828B microcontroller, the internal register file is configured as eleven pages (Pages 0-9, 15). The pages 0-9 are used for general purpose register file.
2. In the S3C8289 microcontroller, the internal register file is configured as five pages (Pages 0-3, 15).
3. In the S3C8285 microcontroller, the internal register file is configured as three pages (Pages 0-1, 15) The page 0-1 is used for general purpose register file.
4. The page 15 of S3C828B/9/5 is used for LCD data register or general purpose regiser.

**Figure 2-6. 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 68 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 S3C828B,
the set 2 address range (C0H–FFH) is accessible on pages 0-9.
S3C8289, the set 2 address range (C0H–FFH) is accessible on pages 0-3.
S3C8285, the set 2 address range (C0H–FFH) is accessible on pages 0-1.

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 S3C828B/C8289/C8285's ten or four or two 256-byte register pages is called *prime register area.* Prime registers can be accessed using any of the seven addressing modes (see Chapter 3, "Addressing Modes.")

The prime register area on page 0 is immediately addressable following a reset. In order to address prime registers on pages 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, or 15 you must set the register page pointer (PP) to the appropriate source and destination values.
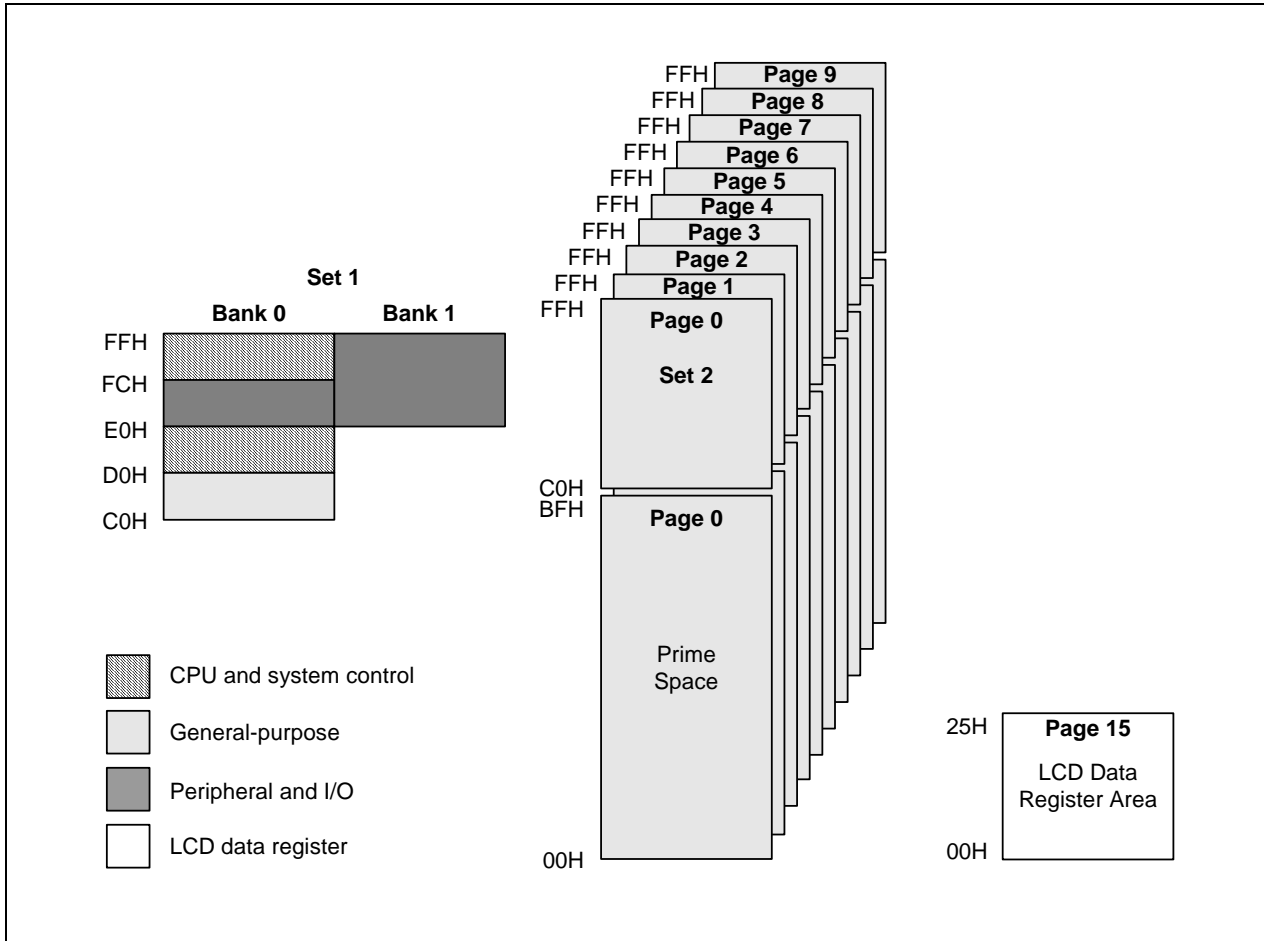


**Figure 2-7. Set 1, Set 2, Prime Area Register, and LCD Data Register Map**

### 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).
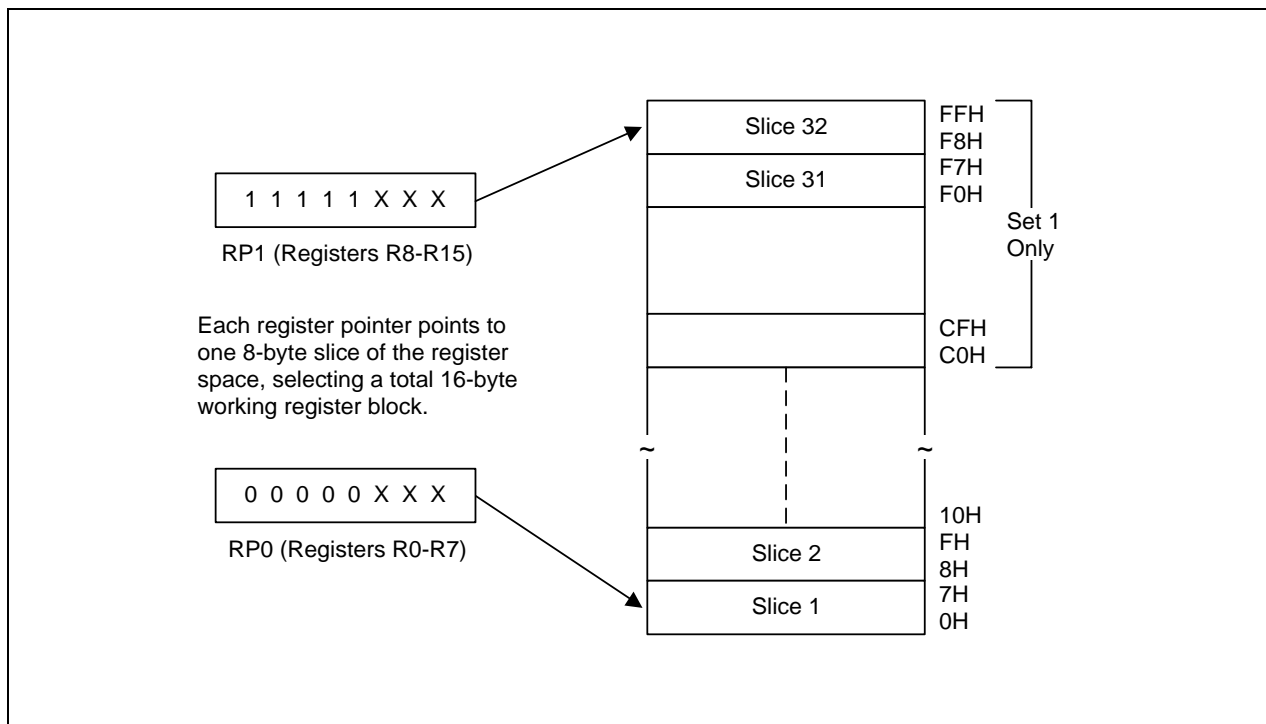


**Figure 2-8. 8-Byte Working Register Areas (Slices)**

## USING THE REGISTER POINTS

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

To change a register pointer value, you load a new value to RP0 and/or RP1 using an SRP or LD instruction. (see Figures 2-9 and 2-10).

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 2-9). In some cases, it may be necessary to define working register areas in different (non-contiguous) areas of the register file. In Figure 2-10, 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
```
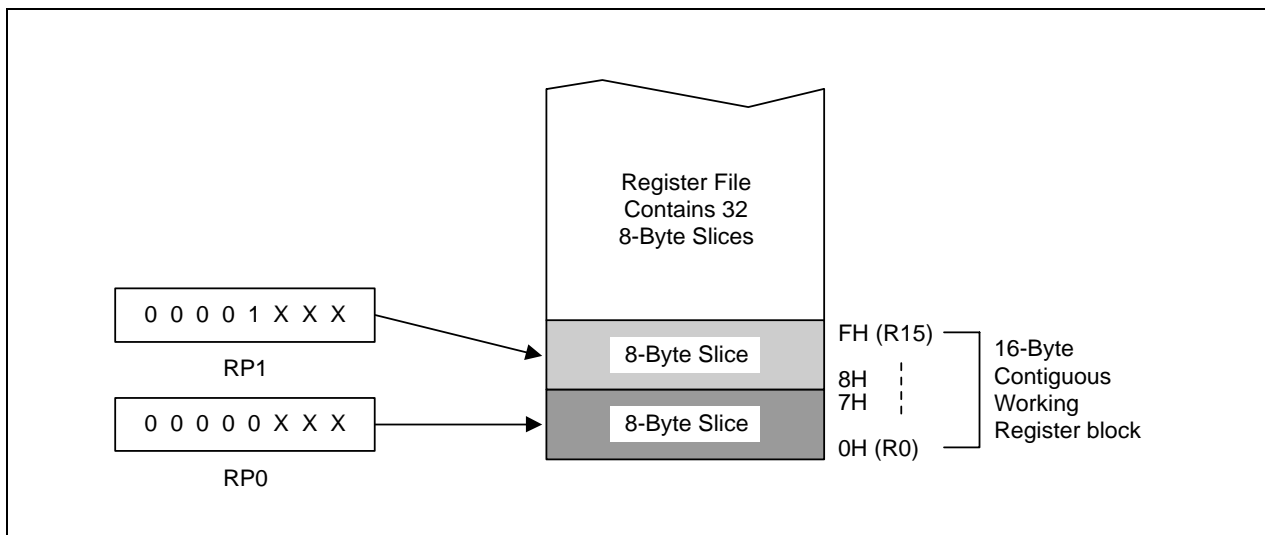


**Figure 2-9. Contiguous 16-Byte Working Register Block**
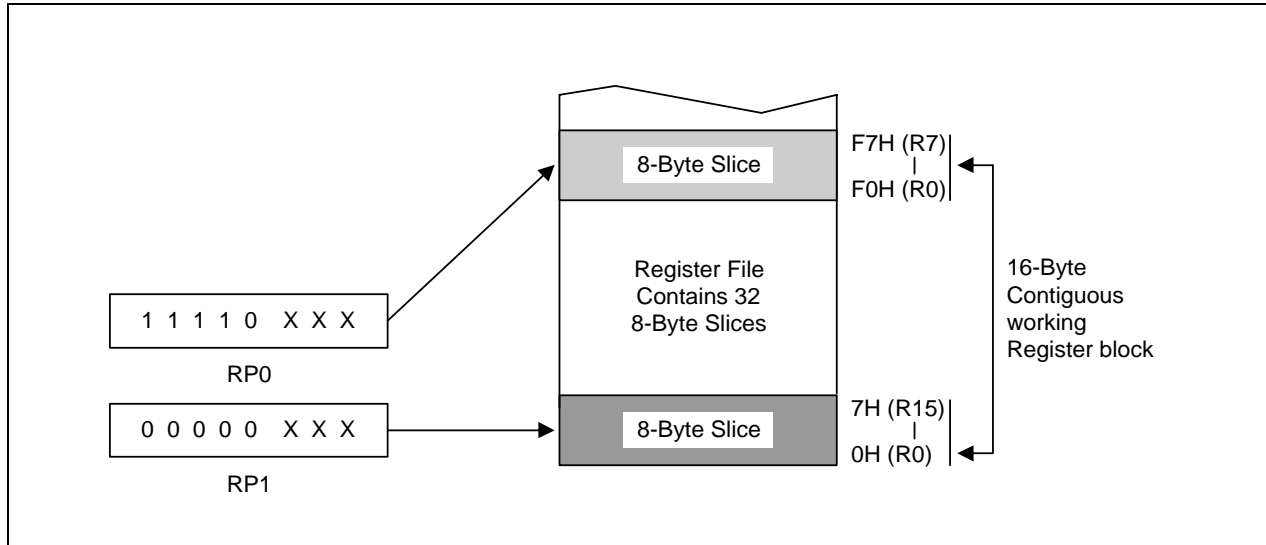
**Figure 2-10. Non-Contiguous 16-Byte Working Register Block**

☞ **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.

| MSB | LSB | n = Even address |
|-----|-----|------------------|
| Rn  | Rn+1 |                 |
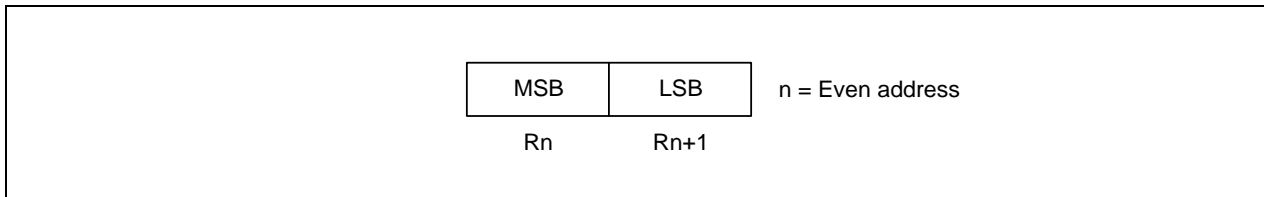
**Figure 2-11. 16-Bit Register Pair**
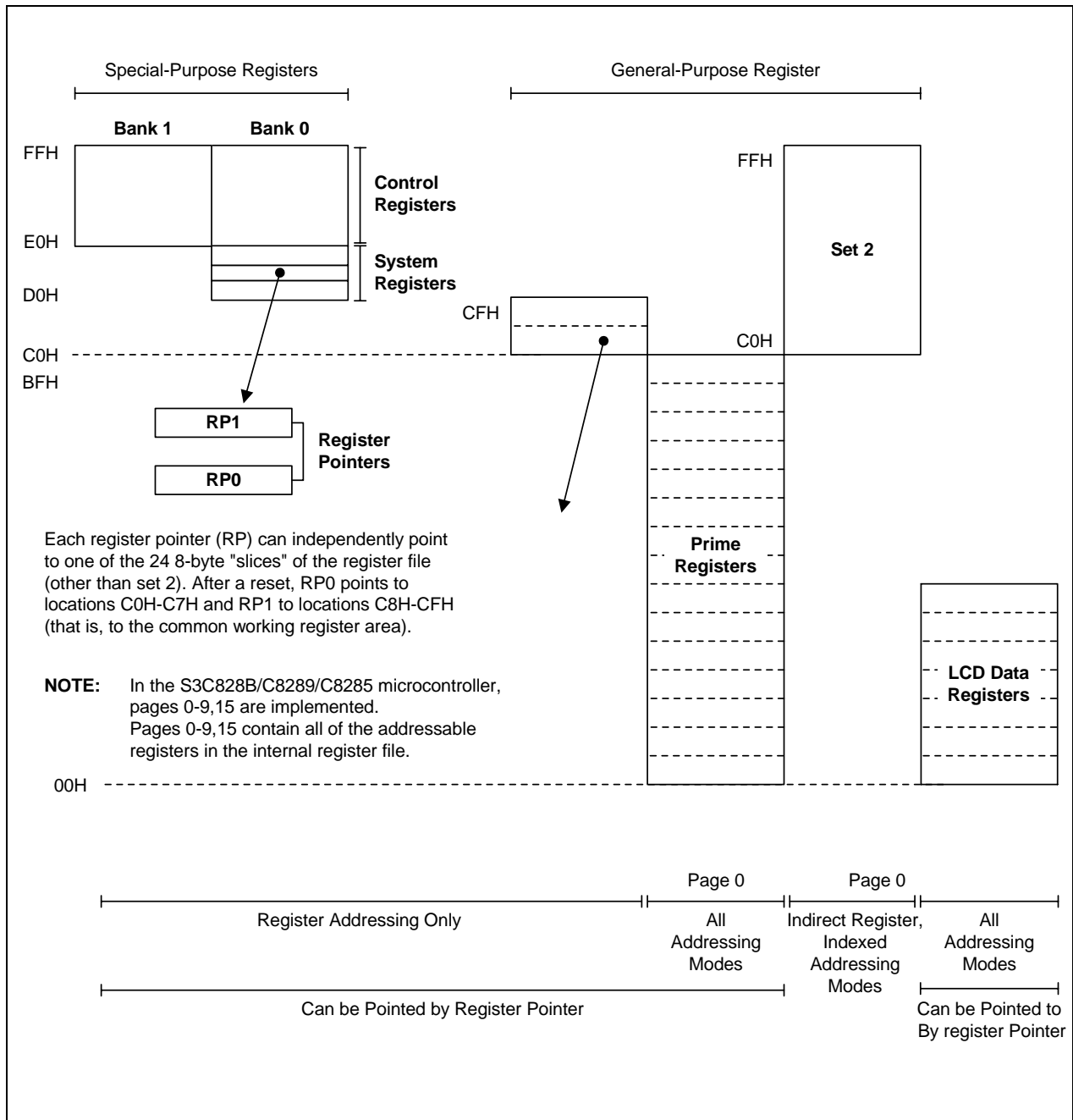
**Figure 2-12. Register File Addressing**

## COMMON WORKING REGISTER AREA (C0H–CFH)

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

RP0   →   C0H–C7H

RP1   →   C8H–CFH

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



**NOTE:**   In case of S3C8289/F8289, page0-page3 and S3C8285/F8285, page0-page1.
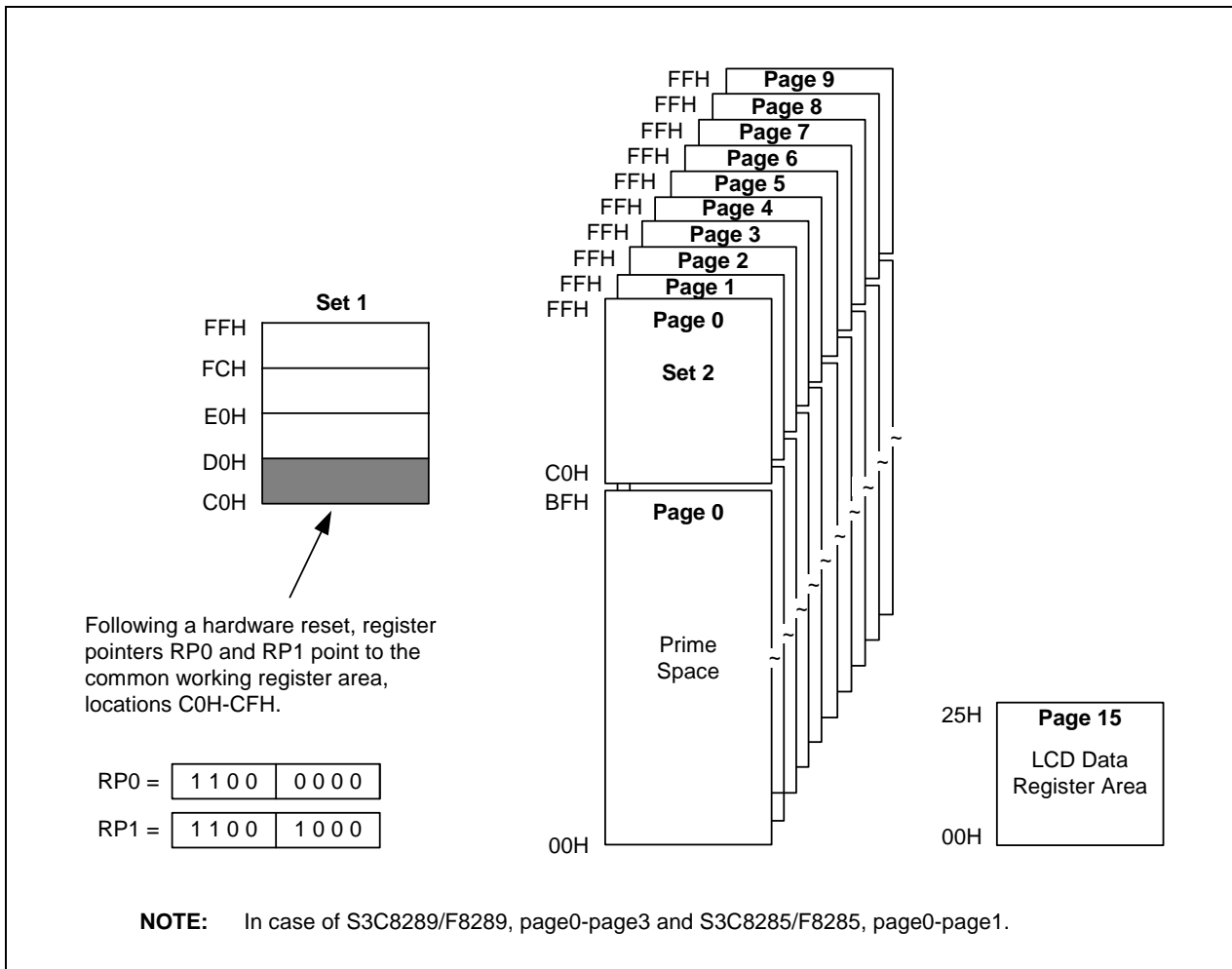
**Figure 2-13. Common Working Register Area**

☞ **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 2-14, 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 2-15 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).
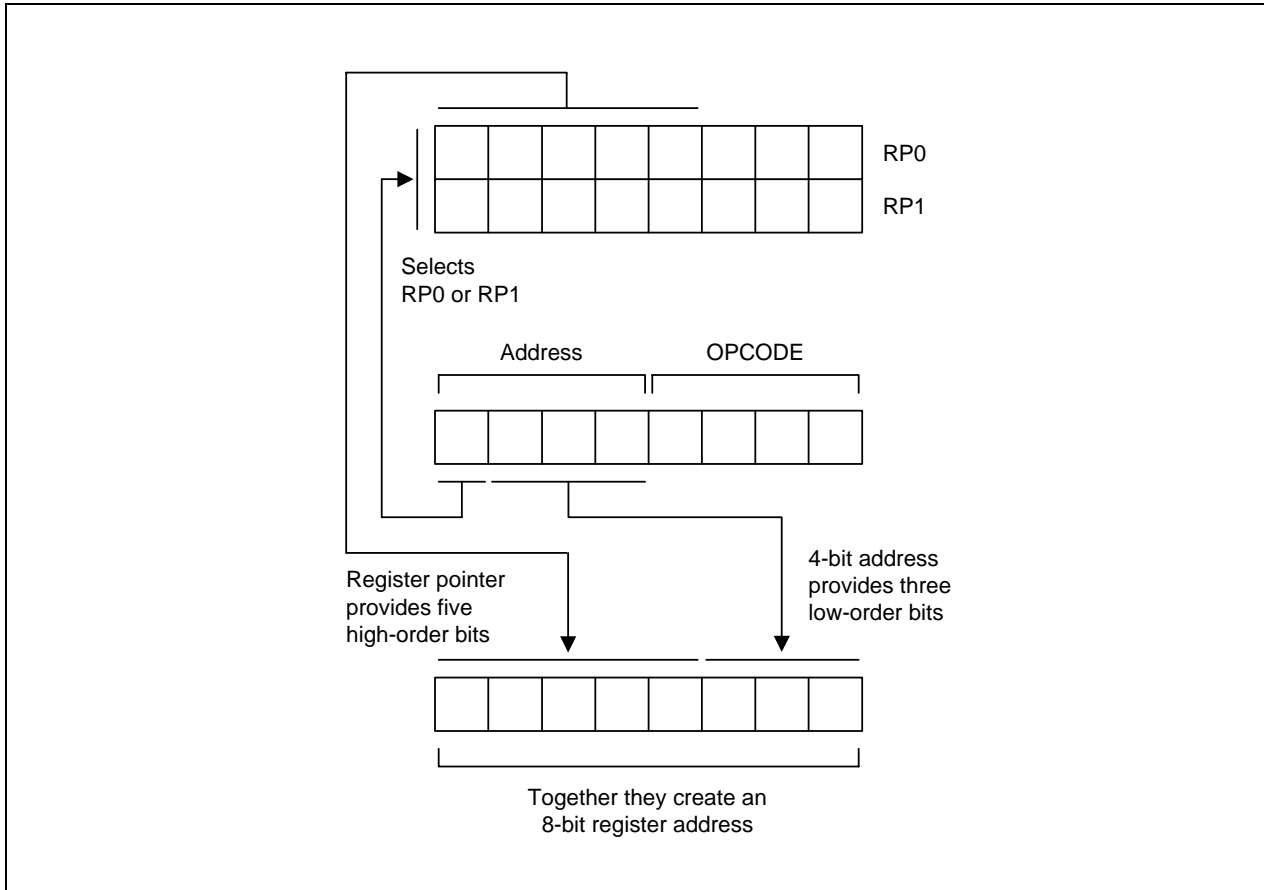
**Figure 2-14. 4-Bit Working Register Addressing**
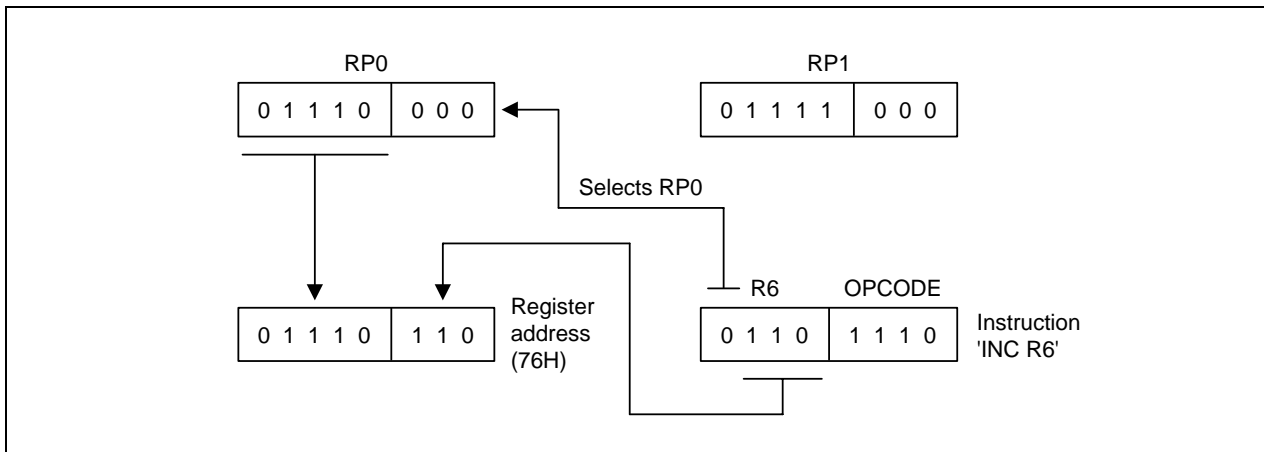


**Figure 2-15. 4-Bit Working Register Addressing Example**

## 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 2-16, 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 2-17 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).
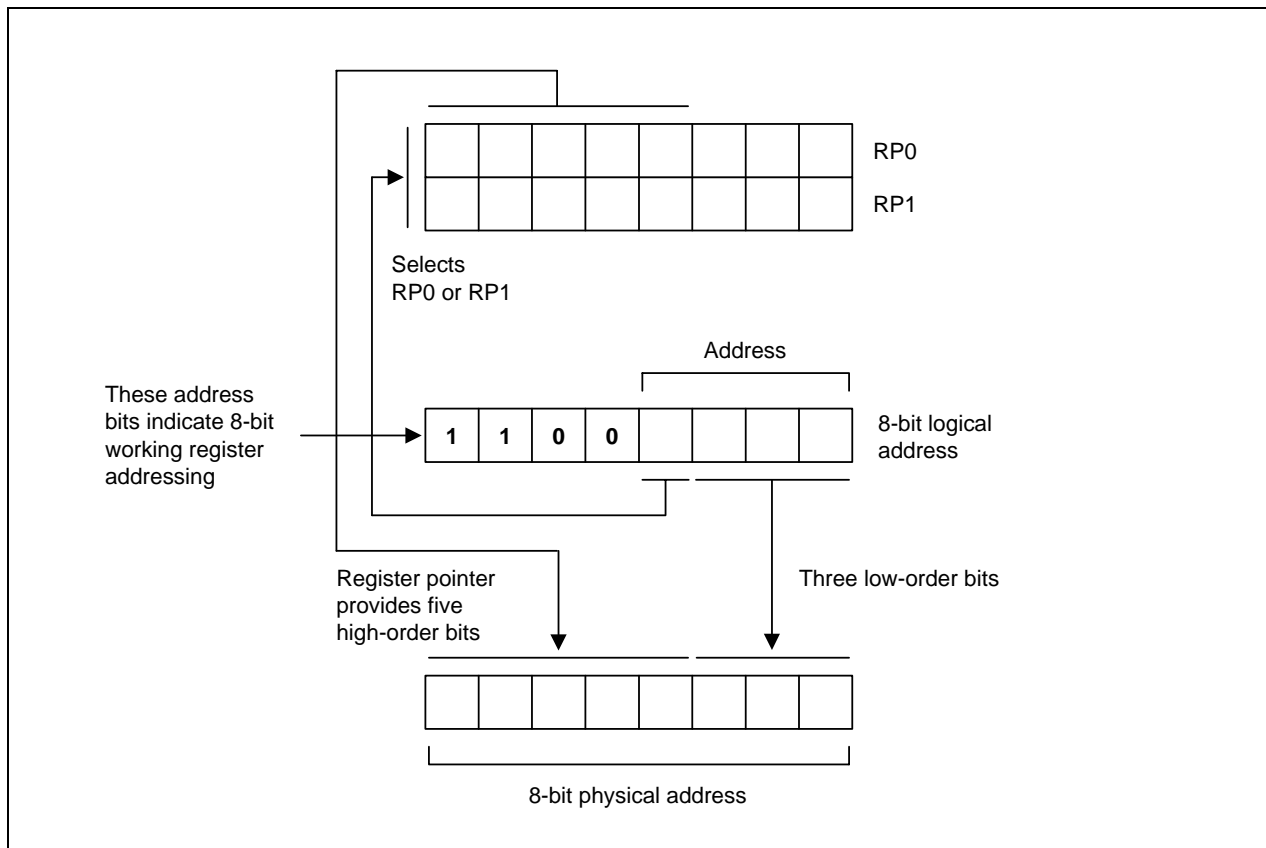


**Figure 2-16. 8-Bit Working Register Addressing**
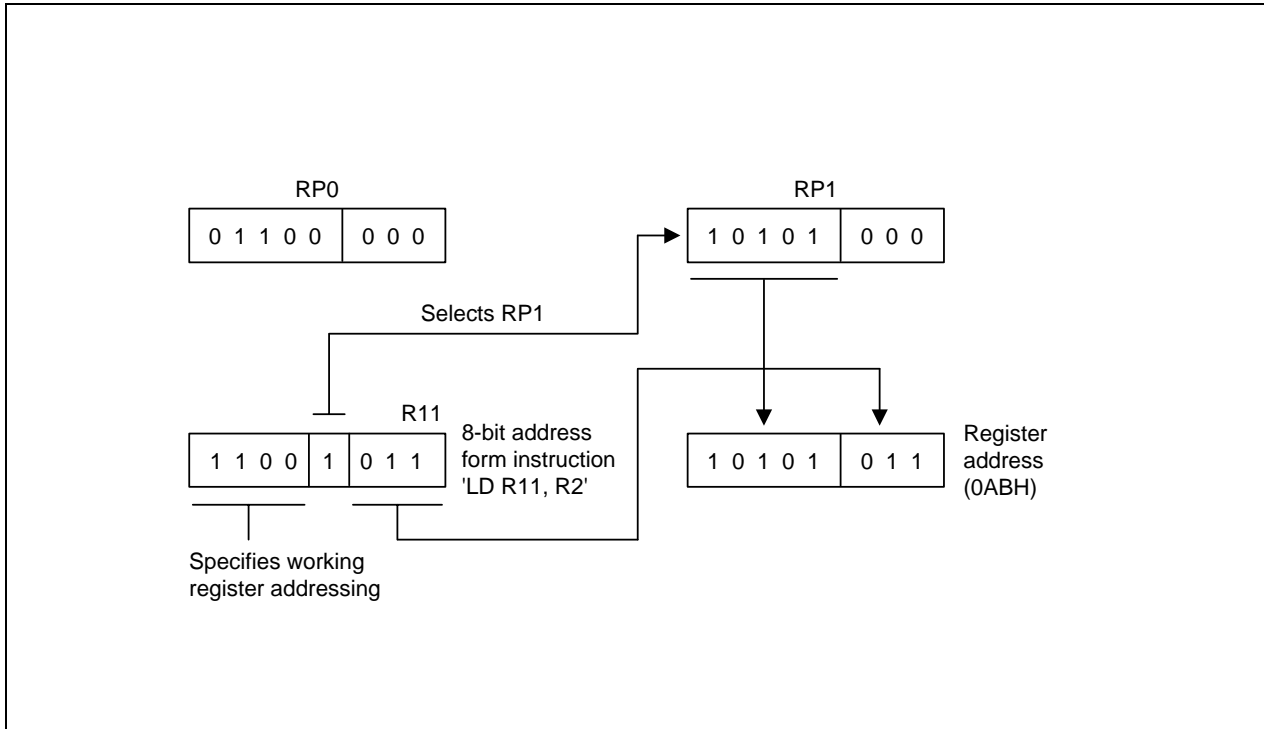
**Figure 2-17. 8-Bit Working Register Addressing Example**

## 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 S3C828B/C8289/C8285 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 2-18.
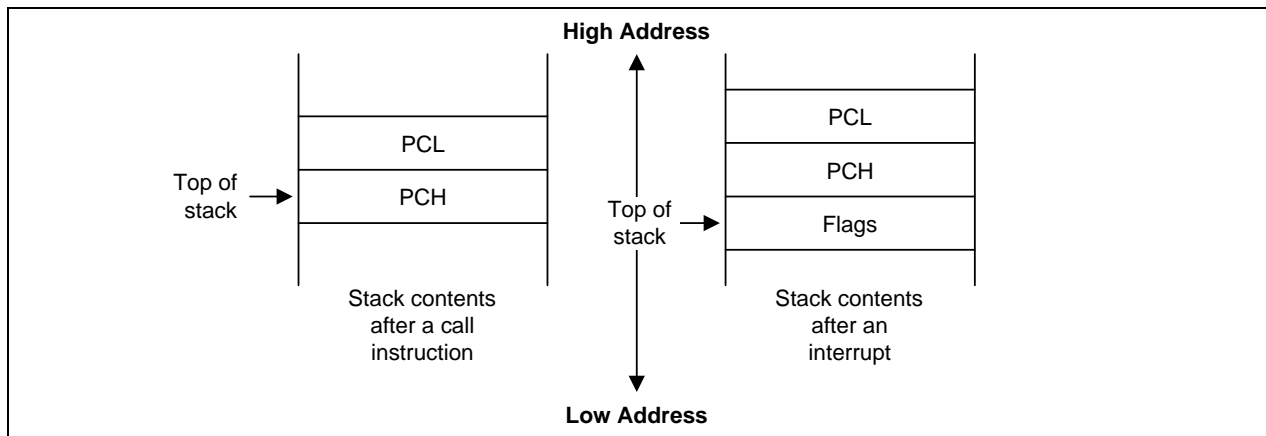


**Figure 2-18. Stack Operations**

### 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 S3C828B/C8289/C8285, 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).
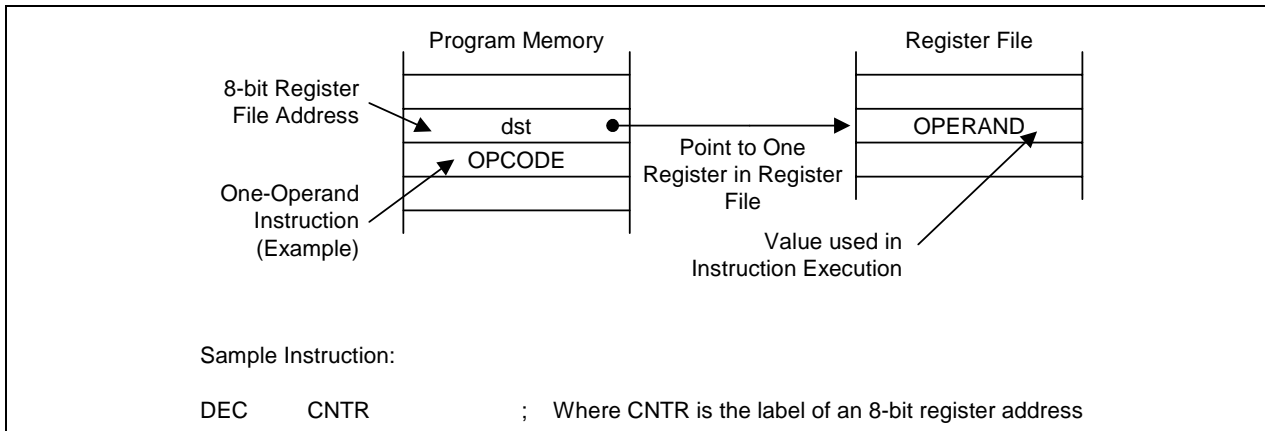


**Figure 3-1. Register Addressing**
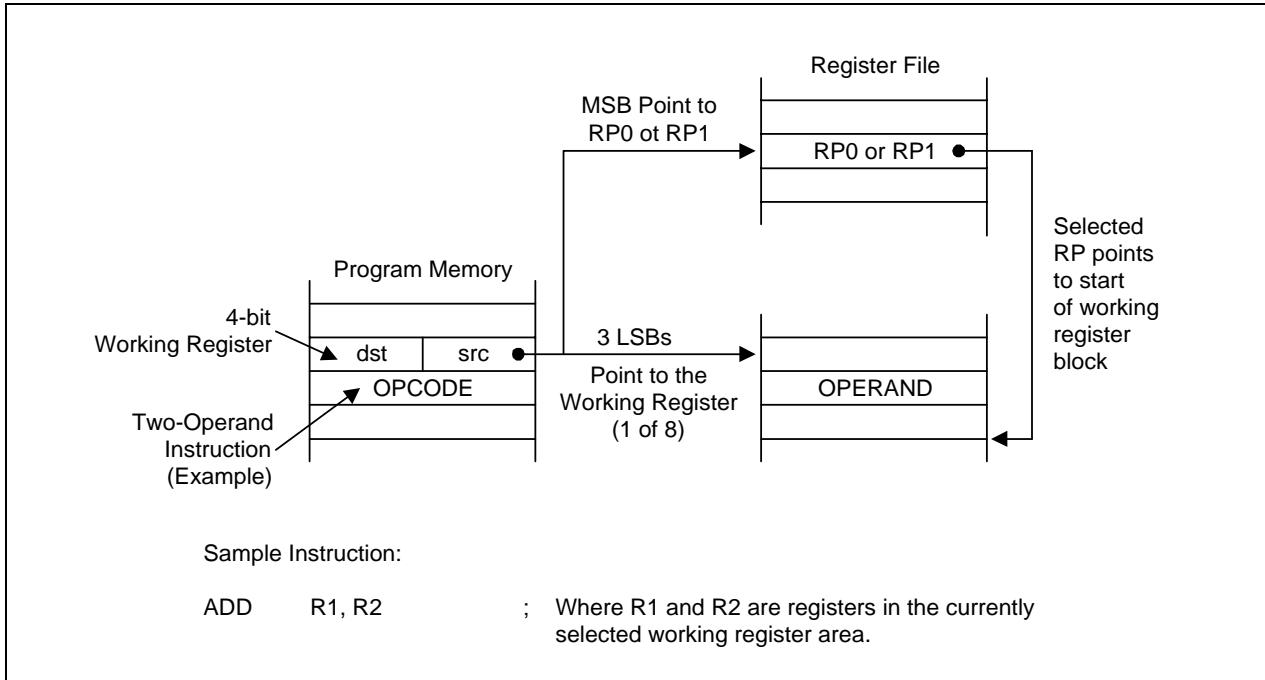


**Figure 3-2. Working Register Addressing**

## INDIRECT REGISTER ADDRESSING MODE (IR)

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

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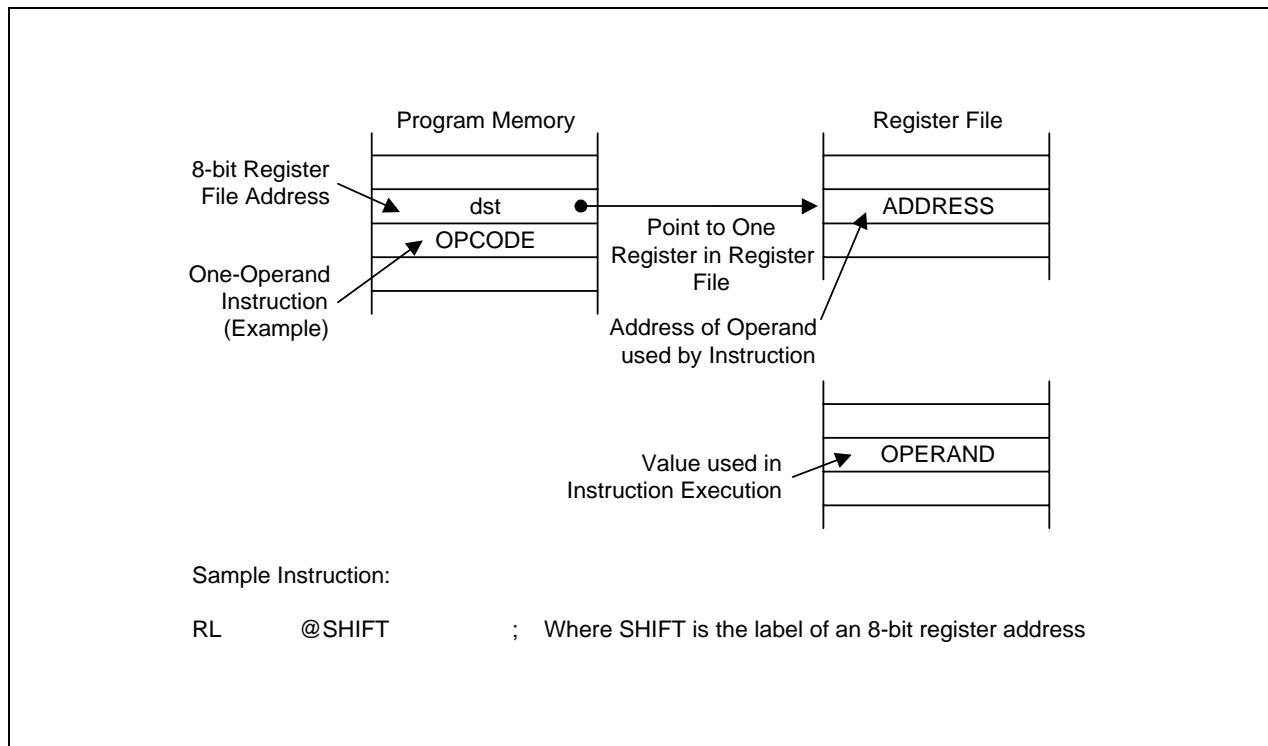


**Figure 3-3. Indirect Register Addressing to Register File**

## INDIRECT REGISTER ADDRESSING MODE (Continued)
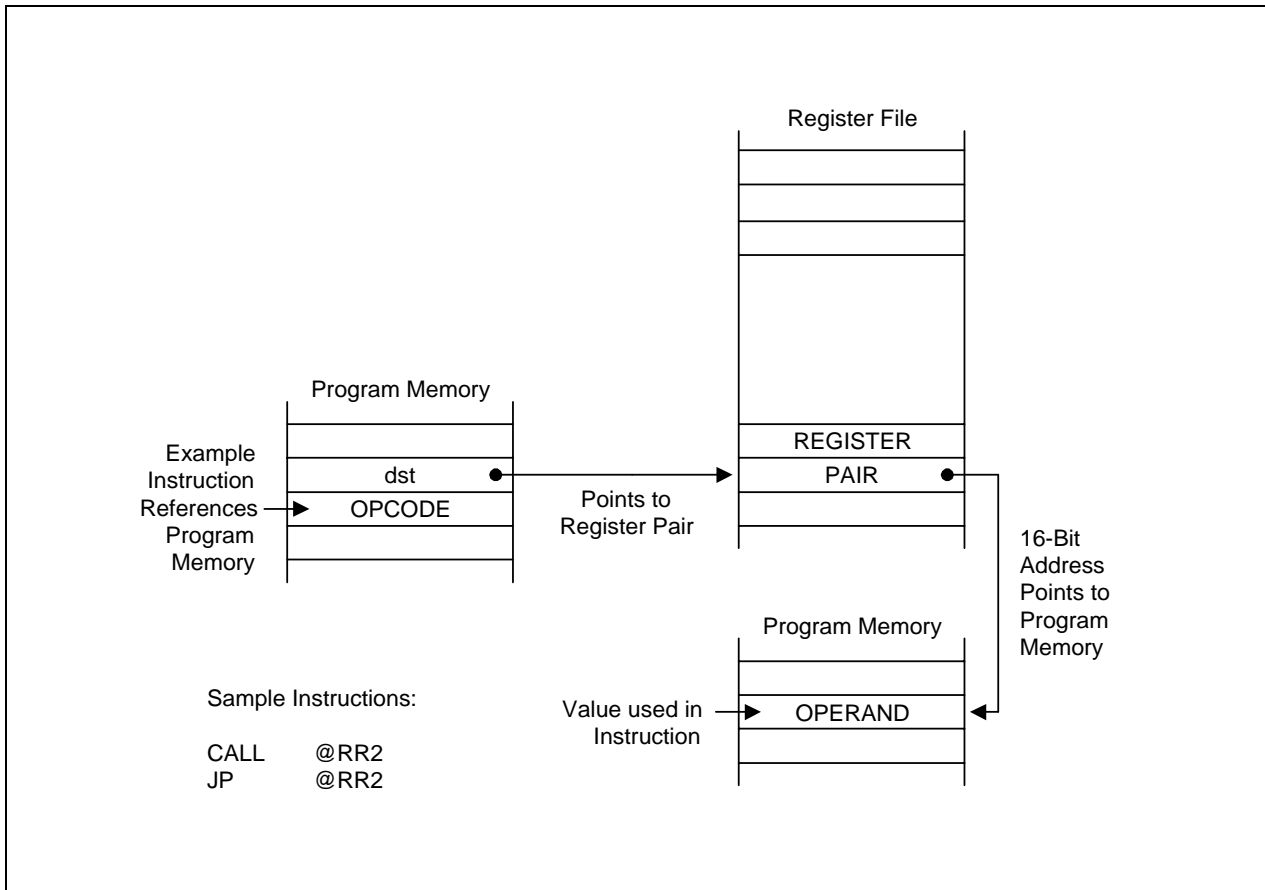


**Figure 3-4. Indirect Register Addressing to Program Memory**

## INDIRECT REGISTER ADDRESSING MODE (Continued)
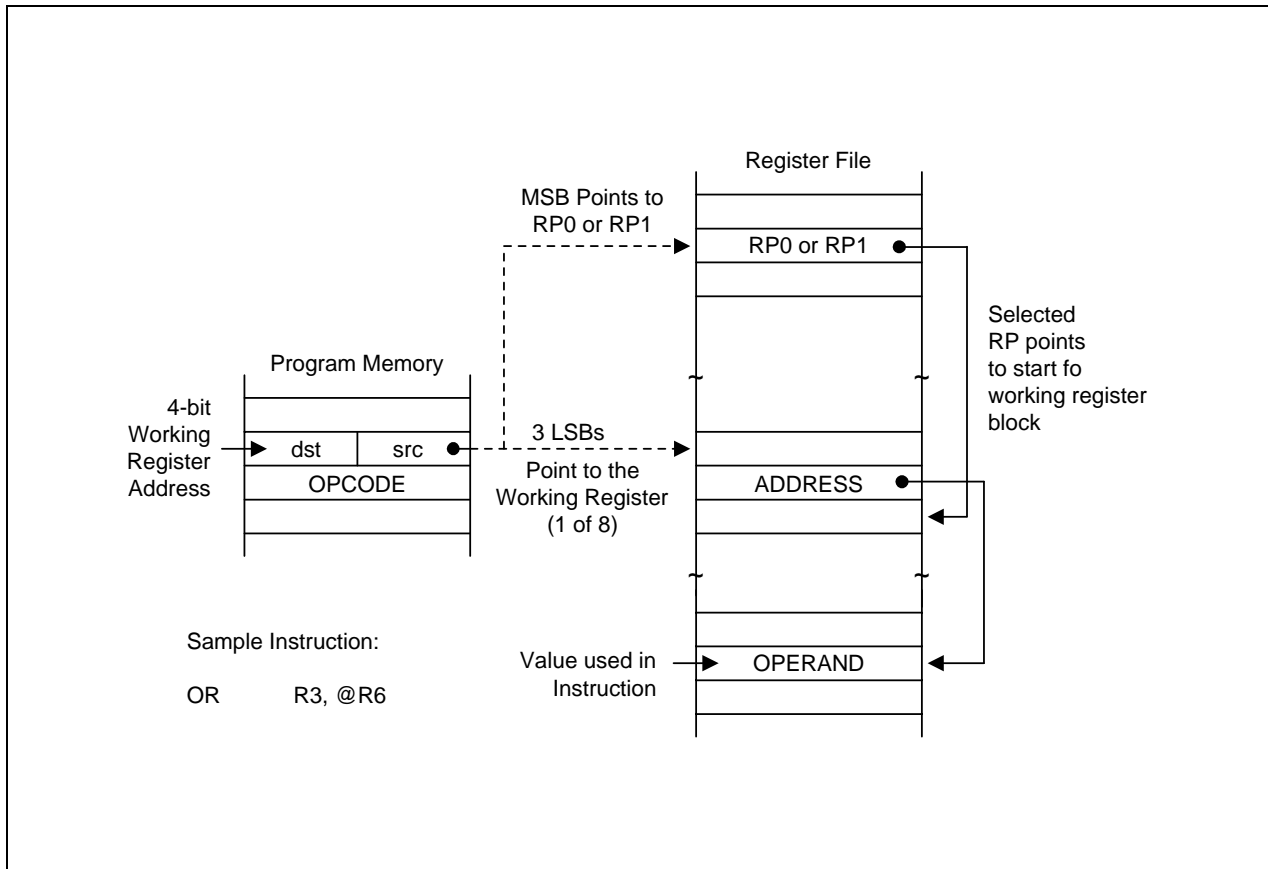


**Figure 3-5. Indirect Working Register Addressing to Register File**

## INDIRECT REGISTER ADDRESSING MODE (Concluded)
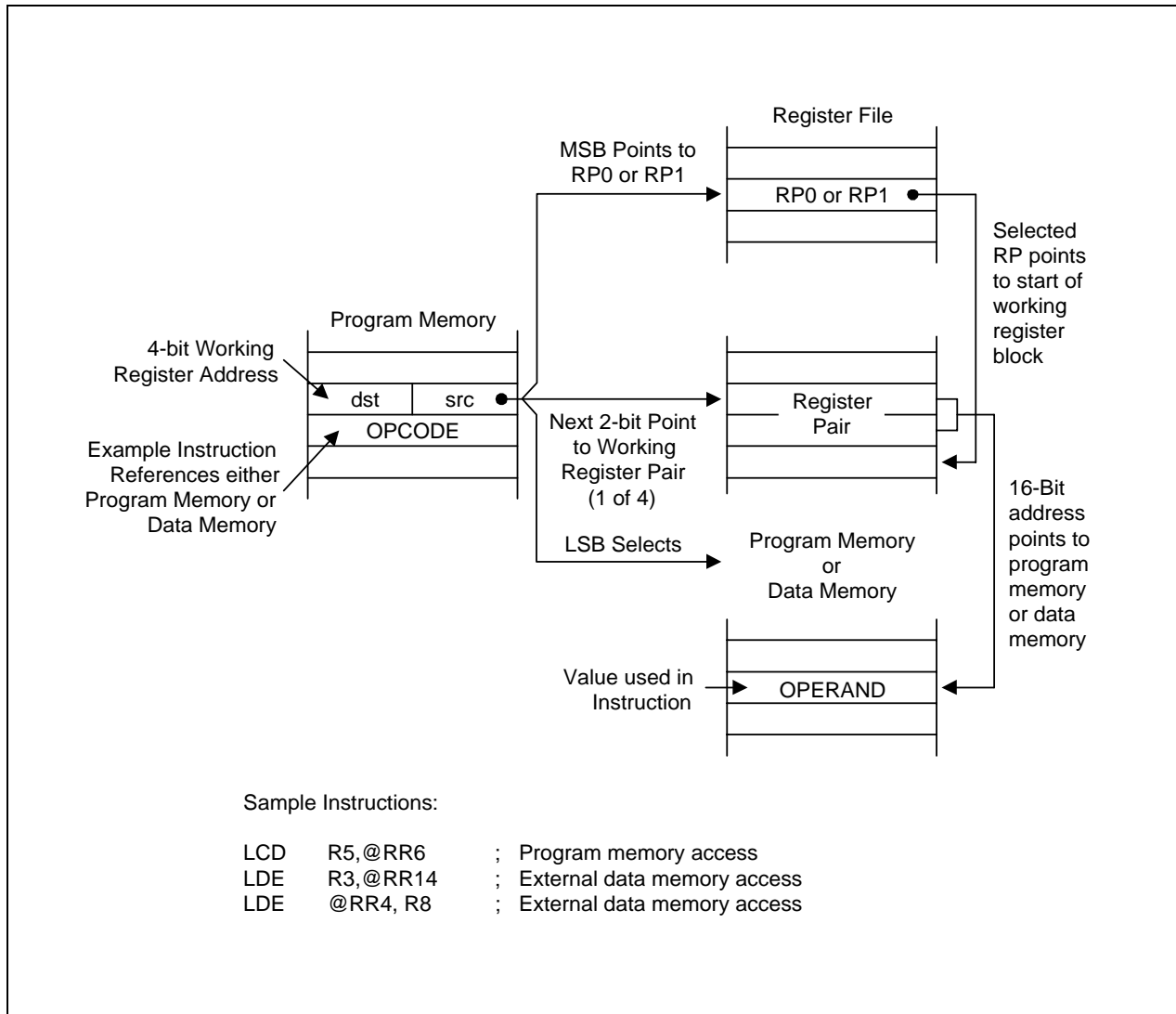


**Figure 3-6. Indirect Working Register Addressing to Program or Data Memory**

## INDEXED ADDRESSING MODE (X)

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

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

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

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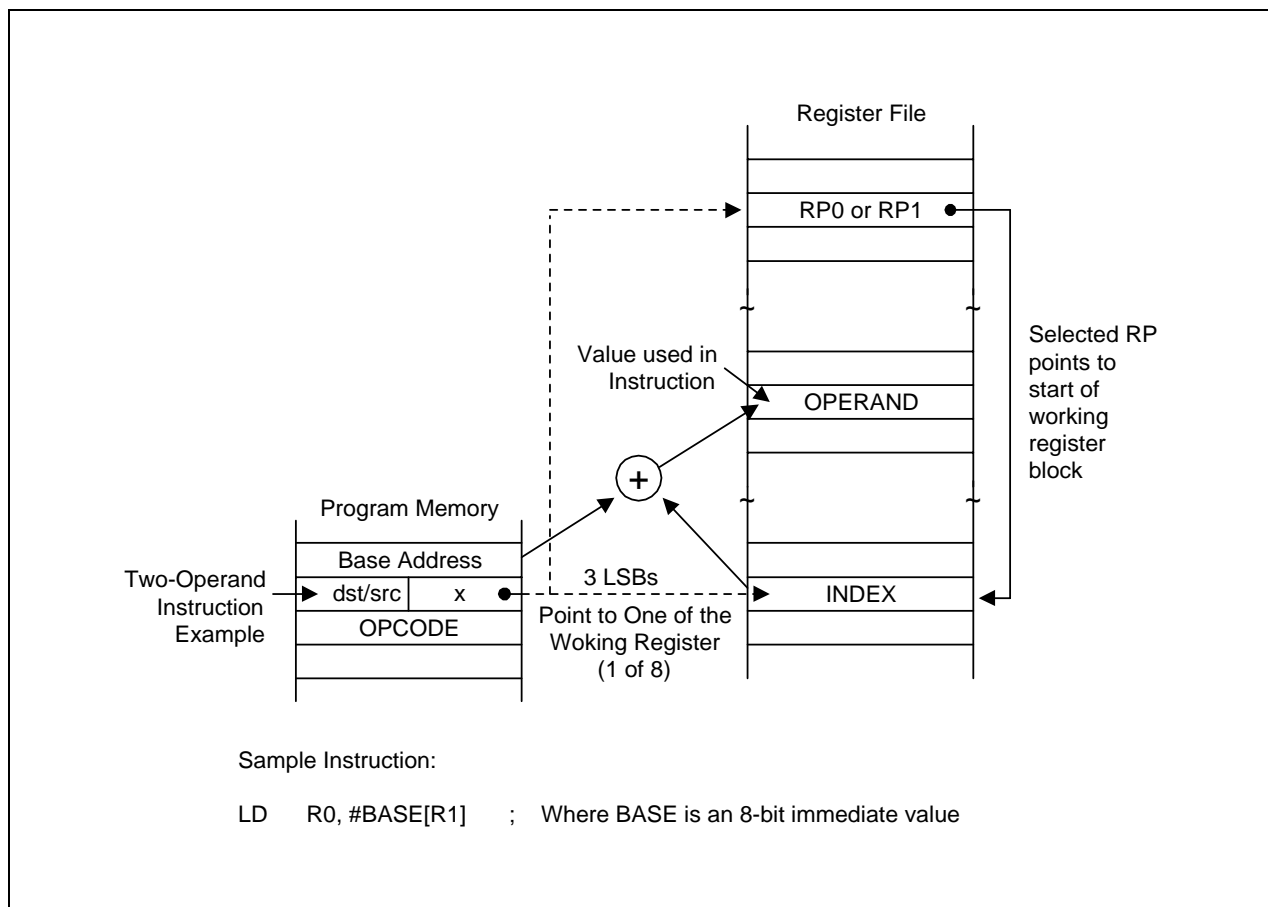


**Figure 3-7. Indexed Addressing to Register File**

## INDEXED ADDRESSING MODE (Continued)
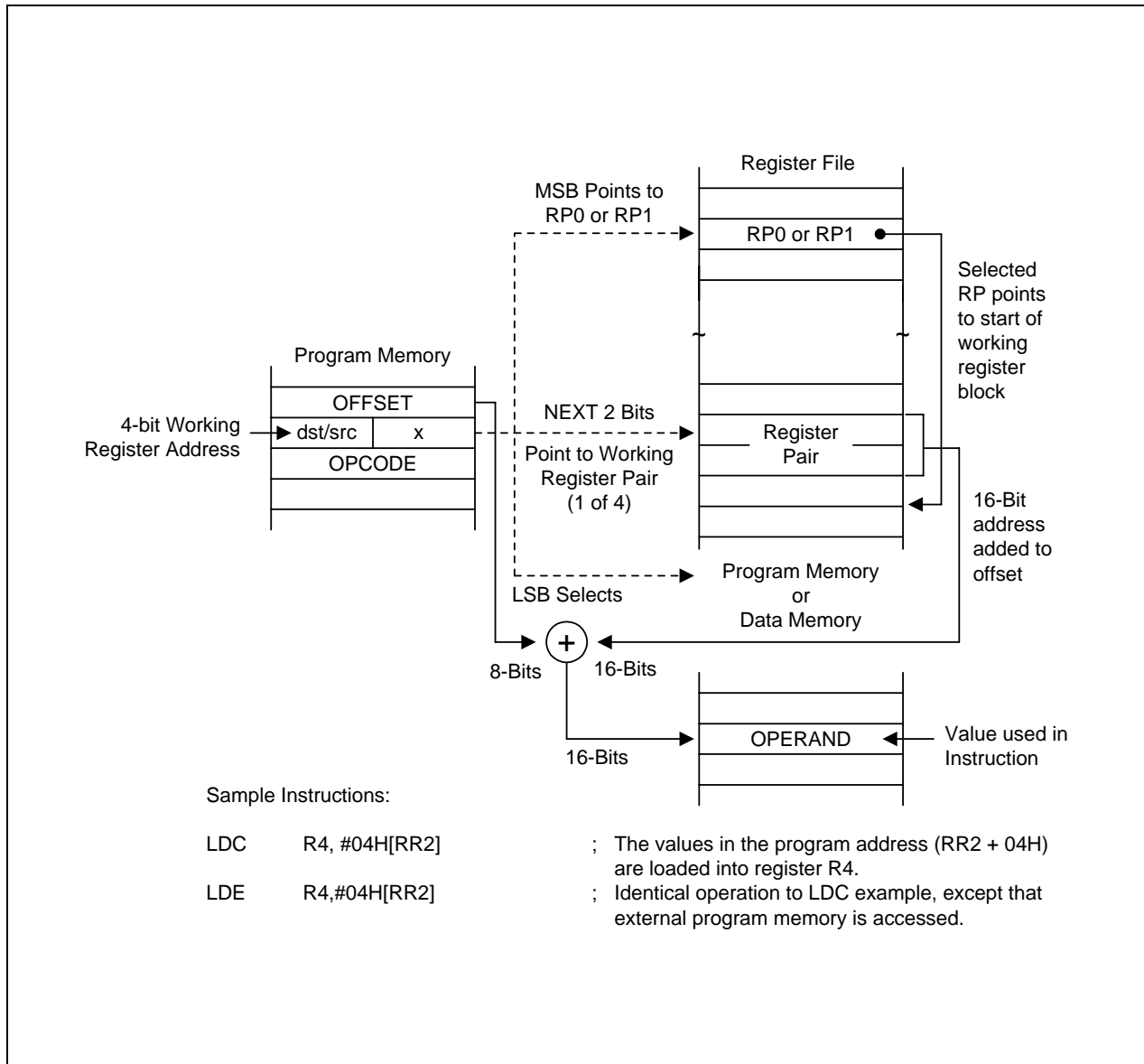


**Figure 3-8. Indexed Addressing to Program or Data Memory with Short Offset**

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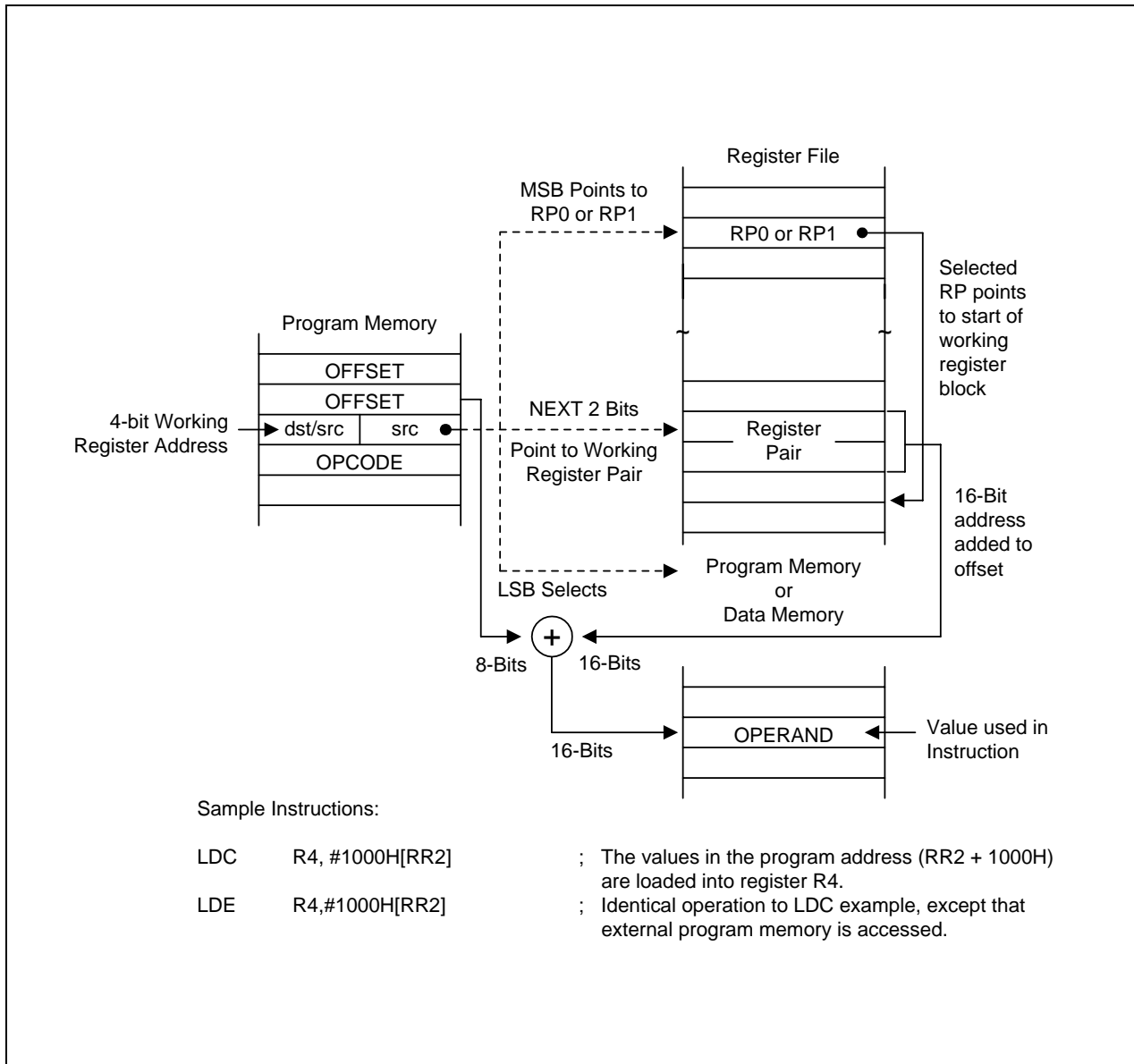
**Figure 3-9. Indexed Addressing to Program or Data Memory**

## 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.
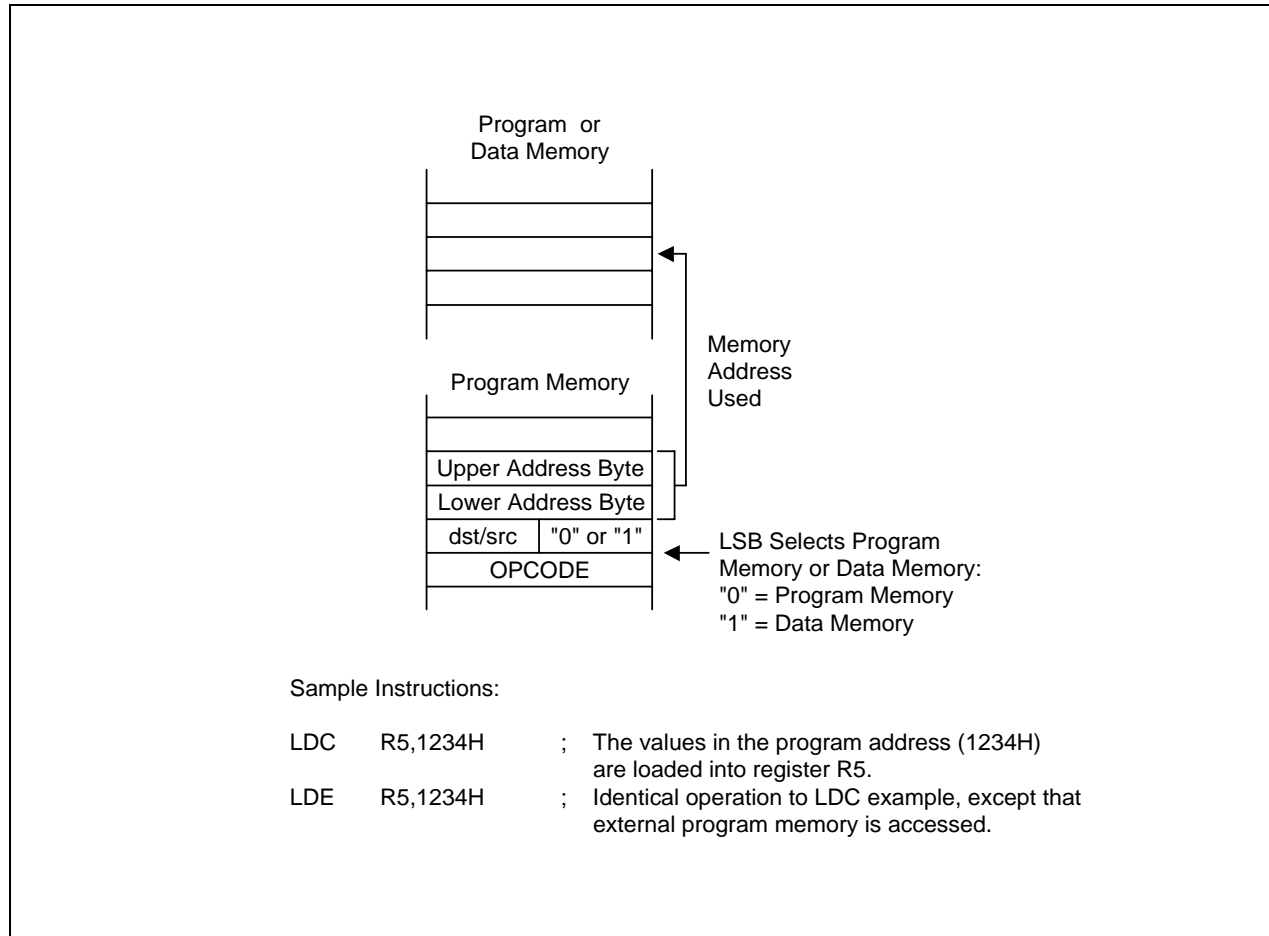
**Figure 3-10. Direct Addressing for Load Instructions**

## DIRECT ADDRESS MODE (Continued)

Program Memory

| Next OPCODE |

Memory
Address
Used

| Upper Address Byte |
| Lower Address Byte |
| OPCODE |

Sample Instructions:

JP      C,JOB1          ;    Where JOB1 is a 16-bit immediate address
CALL    DISPLAY         ;    Where DISPLAY is a 16-bit immediate address
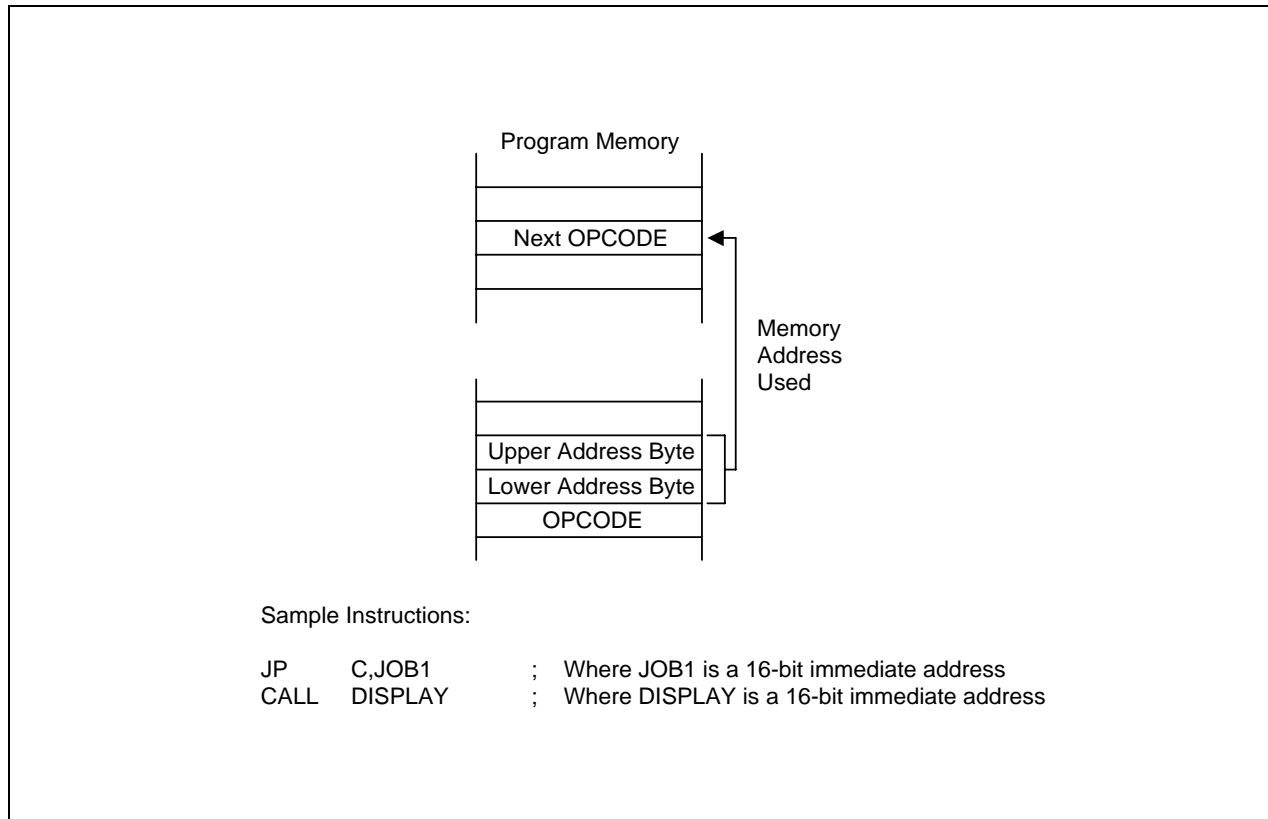
**Figure 3-11. Direct Addressing for Call and Jump Instructions**

## 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.

Program Memory

Next Instruction

LSB Must be Zero

dst

Current
Instruction                OPCODE

Lower Address Byte          Program Memory
Upper Address Byte          Locations 0-255

Sample Instruction:

CALL    #40H            ;   The 16-bit value in program memory addresses 40H
                            and 41H is the subroutine start address.
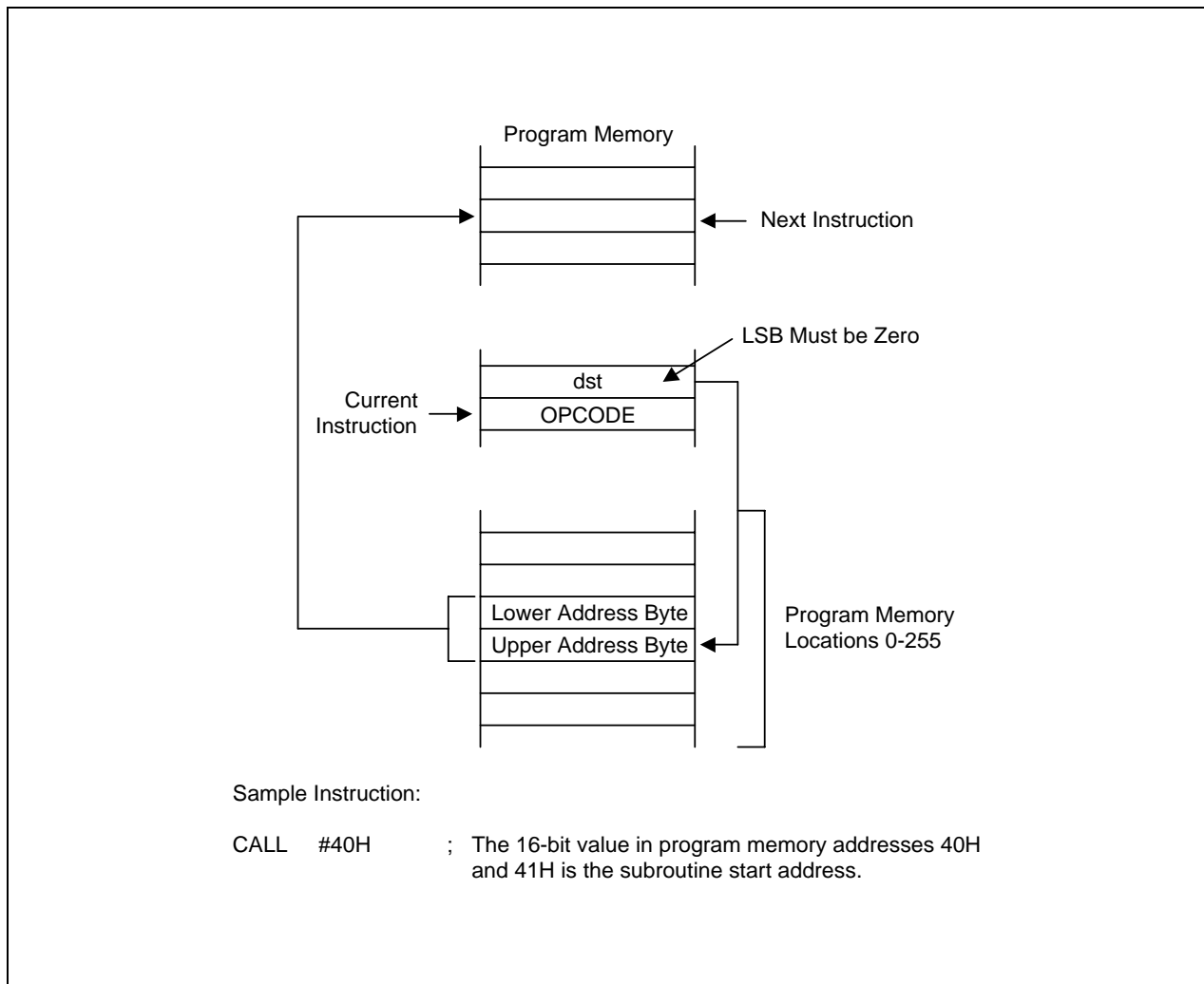
**Figure 3-12. Indirect Addressing**

## 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.
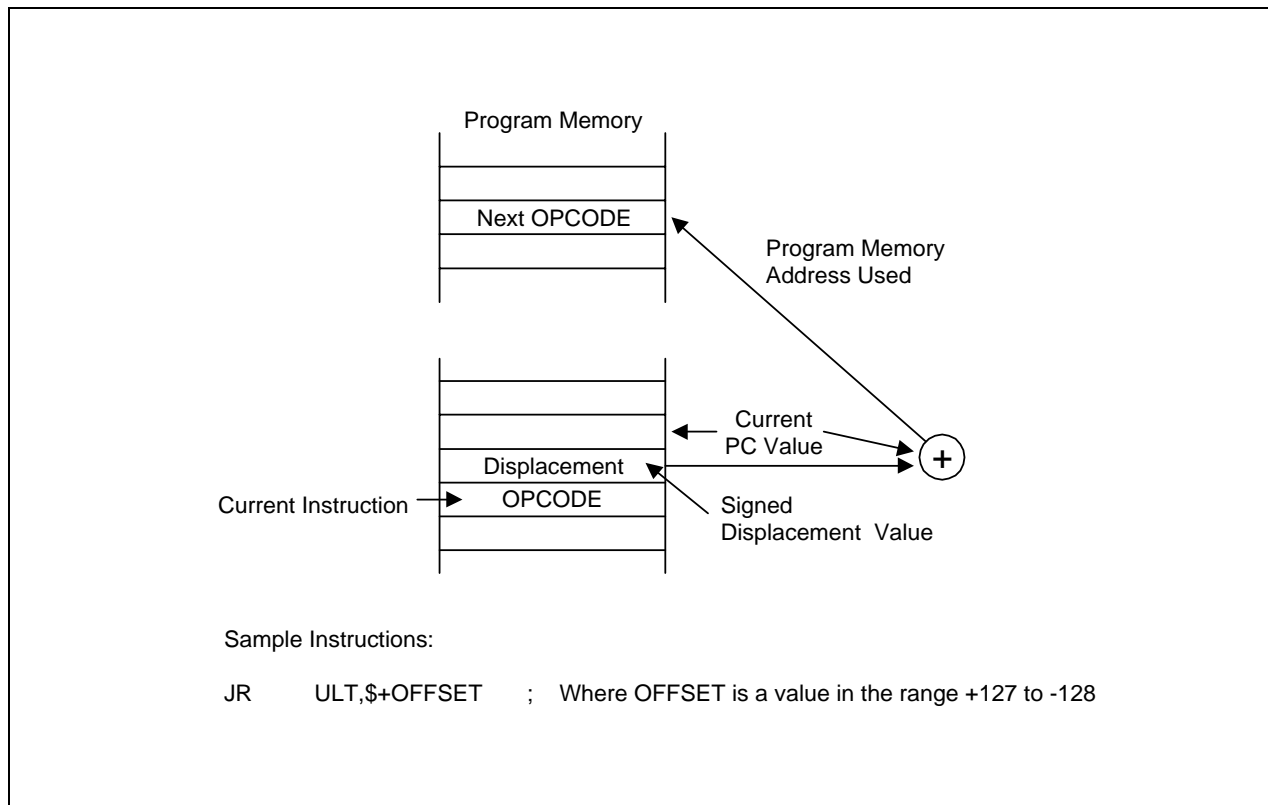


**Figure 3-13. Relative Addressing**

## 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.

Program Memory

| OPERAND |
| OPCODE |

(The Operand value is in the instruction)

Sample Instruction:

LD      R0,#0AAH
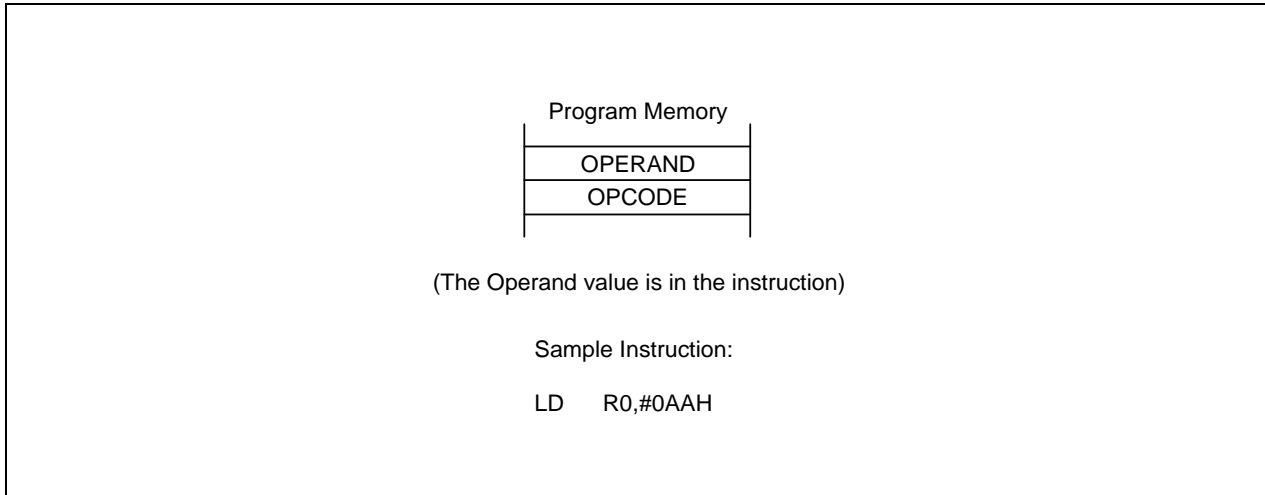
**Figure 3-14. Immediate Addressing**

# 4 CONTROL REGISTERS

## OVERVIEW

In this chapter, detailed descriptions of the S3C828B/C8289/C8285 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 S3C828B/C8289/C8285 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 | Decimal | Hex | R/W |
|---|---|---|---|---|
| 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 |

SAMSUNG
ELECTRONICS

**Table 4-2. Set 1, Bank 0 Registers**

| Register Name | Mnemonic | Decimal | Hex | R/W |
|---|---|---|---|---|
| LCD Control Register | LCON | 208 | D0H | R/W |
| Watch Timer Control Register | WTCON | 209 | D1H | R/W |
| Battery Level Detector Control Register | BLDCON | 210 | D2H | R/W |
| SIO Control Register | SIOCON | 224 | E0H | R/W |
| SIO Data Register | SIODATA | 225 | E1H | R/W |
| SIO Pre-Scaler Register | SIOPS | 226 | E2H | R/W |
| Timer 0 Control Register | T0CON | 227 | E3H | R/W |
| Timer 0 Counter Register(High Byte) | T0CNTH | 228 | E4H | R |
| Timer 0 Counter Register(Low Byte) | T0CNTL | 229 | E5H | R |
| Timer 0 Data Register(High Byte) | T0DATAH | 230 | E6H | R/W |
| Timer 0 Data Register(Low Byte) | T0DATAL | 231 | E7H | R/W |
| Timer A Control Register | TACON | 232 | E8H | R/W |
| Timer A Counter Register | TACNT | 233 | E9H | R |
| Timer A Data Register | TADATA | 234 | EAH | R/W |
| Timer 1 Control Register | T1CON | 235 | EBH | R/W |
| Timer 1 Counter Register(High Byte) | T1CNTH | 236 | ECH | R |
| Timer 1 Counter Register(Low Byte) | T1CNTL | 237 | EDH | R |
| Timer 1 Data Register(High Byte) | T1DATAH | 238 | EEH | R/W |
| Timer 1 Data Register(Low Byte) | T1DATAL | 239 | EFH | R/W |
| Timer B Data Register(High Byte) | TBDATAH | 240 | F0H | R/W |
| Timer B Data Register(Low Byte) | TBDATAL | 241 | F1H | R/W |
| Timer B Control Register | TBCON | 242 | F2H | R/W |
| A/D Converter Control Register | ADCON | 243 | F3H | R/W |
| A/D Converter Data Register(High Byte) | ADDATAH | 244 | F4H | R |
| A/D Converter Data Register(Low Byte) | ADDATAL | 245 | F5H | R |
| UART Control Register | UARTCON | 246 | F6H | R/W |
| UART Data Register | UDATA | 247 | F7H | R/W |
| UART Baud Rate Data Register | BRDATA | 248 | F8H | R/W |
| Interrupt Pending Register | INTPND | 249 | F9H | R/W |
| Oscillator Control Register | OSCCON | 250 | FAH | R/W |
| STOP Control Register | STPCON | 251 | FBH | R/W |
| Location FCH is not mapped. | | | | |
| Basic Timer Counter | BTCNT | 253 | FDH | R |
| Location FEH is not mapped. | | | | |
| Interrupt Priority Register | IPR | 255 | FFH | R/W |

SAMSUNG
ELECTRONICS

**Table 4-3. Set 1, Bank 1 Registers**

| Register Name | Mnemonic | Decimal | Hex | R/W |
|---|---|---|---|---|
| Flash Memory Sector Address Register(High Byte) | FMSECH | 208 | D0H | R/W |
| Flash Memory Sector Address Register(Low Byte) | FMSECL | 209 | D1H | R/W |
| Flash Memory Control Register | FMCON | 210 | D2H | R/W |
| Port 0 Control Register(High Byte) | P0CONH | 224 | E0H | R/W |
| Port 0 Control Register(Low Byte) | P0CONL | 225 | E1H | R/W |
| Port 0 Interrupt Control Register(High Byte) | P0INTH | 226 | E2H | R/W |
| Port 0 Interrupt Control Register(Low Byte) | P0INTL | 227 | E3H | R/W |
| Port 0 Interrupt Pending Register | P0PND | 228 | E4H | R/W |
| Port 1 Control Register(High Byte) | P1CONH | 229 | E5H | R/W |
| Port 1 Control Register(Low Byte) | P1CONL | 230 | E6H | R/W |
| Port 1 Pull-up Resistor Enable Register | P1PUR | 231 | E7H | R/W |
| Port 2 Control Register(High Byte) | P2CONH | 232 | E8H | R/W |
| Port 2 Control Register(Low Byte) | P2CONL | 233 | E9H | R/W |
| Port 3 Control Register(High Byte) | P3CONH | 234 | EAH | R/W |
| Port 3 Control Register(Low Byte) | P3CONL | 235 | EBH | R/W |
| Port 4 Control Register(High Byte) | P4CONH | 236 | ECH | R/W |
| Port 4 Control Register(Low Byte) | P4CONL | 237 | EDH | R/W |
| Port 4 Pull-up Resistor Enable Register | P4PUR | 238 | EEH | R/W |
| Port 5 Pull-up Resistor Enable Register | P5PUR | 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 |
| Port 7 Data Register | P7 | 247 | F7H | R/W |
| Port 8 Data Register | P8 | 248 | F8H | R/W |
| Port 5 Control Register(High Byte) | P5CONH | 249 | F9H | R/W |
| Port 5 Control Register(Low Byte) | P5CONL | 250 | FAH | R/W |
| Port 6 Control Register(High Byte) | P6CONH | 251 | FBH | R/W |
| Port 6 Control Register(Low Byte) | P6CONL | 252 | FCH | R/W |
| Port 7 Control Register | P7CON | 253 | FDH | R/W |
| Port 8 Control Register | P8CON | 254 | FEH | R/W |
| Flash Memory User Programming Enable Register | FMUSR | 255 | FFH | R/W |

**SAMSUNG**
**ELECTRONICS**

Bit number(s) that is/are appended to the register name for bit addressing

Name of individual bit or related bits

Register address (hexadecimal)

Register location in the internal register file

Register ID

Full Register name

**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 | x | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bit Addressing Mode     Register addressing mode only

.7

.6

.5

**Carry Flag (C)**

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

**Zero Flag (Z)**

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

**Sign Flag (S)**

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

R = Read-only
W = Write-only
R/W = Read/write
'-' = Not used

Description of the effect of specific bit settings

Bit number:
MSB = Bit 7
LSB = Bit 0

Type of addressing that must be used to address the bit (1-bit, 4-bit, or 8-bit)

RESET value notation:
'-' = Not used
'x' = Undetermined value
'0' = Logic zero
'1' = Logic one

**Figure 4-1. Register Description Format**

SAMSUNG
ELECTRONICS

# ADCON — A/D Converter Control Register          F3H          Set 1, Bank0

| 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 | R/W | R/W | R/W |
| Addressing Mode | Register addressing mode only |

.7

| Not used for the S3C828B/C8289/C8285 |
|---|

.6–.4          **A/D Input Pin Selection Bits**

| 0 | 0 | 0 | AD0 |
|---|---|---|---|
| 0 | 0 | 1 | AD1 |
| 0 | 1 | 0 | AD2 |
| 0 | 1 | 1 | AD3 |
| 1 | 0 | 0 | AD4 |
| 1 | 0 | 1 | AD5 |
| 1 | 1 | 0 | AD6 |
| 1 | 1 | 1 | AD7 |

.3          **End-of-Conversion Bit (Read-only)**

| 0 | Conversion not complete |
|---|---|
| 1 | Conversion complete |

.2–.1          **Clock Source Selection Bits**

| 0 | 0 | fxx/16 |
|---|---|---|
| 0 | 1 | fxx/8 |
| 1 | 0 | fxx/4 |
| 1 | 1 | fxx/1 |

.0          **Start or Enable Bit**

| 0 | Disable operation |
|---|---|
| 1 | Start operation |

SAMSUNG
ELECTRONICS

# BLDCON — Battery Level Detector Control Register          D2H          Set 1, Bank0

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

**Addressing Mode**          Register addressing mode only

**.7–.6**

| Not used for the S3C828B/C8289/C8285 |
|---|

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

| 0 | Internal source |
|---|---|
| 1 | External source |

**.4**          **BLD Output Bit (Read-only)**

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

**.3**          **BLD Enable/Disable Bit**

| 0 | Disable BLD |
|---|---|
| 1 | Enable BLD |

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

| 0 | 0 | 0 | $V_{BLD}$ = 2.2 V |
|---|---|---|---|
| 1 | 0 | 1 | $V_{BLD}$ = 2.4 V |
| 0 | 1 | 1 | $V_{BLD}$ = 2.8 V |

SAMSUNG
ELECTRONICS

# BTCON — Basic Timer Control Register                    D3H              Set 1

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

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

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

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

| 0 | 0 | fxx/4096 |
|---|---|---|
| 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 S3C828B/C8289/C8285 |
|---|

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

| 0 | 0 | fxx/16 |
|---|---|---|
| 0 | 1 | fxx/8 |
| 1 | 0 | fxx/2 |
| 1 | 1 | fxx/1 |

.2–.0

| Not used for the S3C828B/C8289/C8285 |
|---|

**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.

SAMSUNG
ELECTRONICS

# 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 |

SAMSUNG
ELECTRONICS

# FMCON — Flash Memory Control Register          D2H          Set 1, Bank1

| 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 |
| Others | | | | Not available |

.3          **Sector Erase Status Bit (Read-only)**

| 0 | Success sector erase |
|---|---|
| 1 | Fail sector erase |

.2–.1          | Not used for the S3C828B/C8289/C8285 |
|---|

.0          **Flash Operation Start Bit**

| 0 | Operation stop bit |
|---|---|
| 1 | Operation start bit |

**NOTE:** The FMCON.0 will be cleared automatically just after the corresponding operation completed.

# FMSECH — Flash Memory Sector Address Register (High Byte)  D0H    Set 1, Bank1

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

| .7–.0 | Flash Memory Sector Address Bits (High Byte) |
|---|---|
| | The 15$^{th}$-8$^{th}$ to select a sector of Flash ROM |

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

# FMSECL — Flash Memory Sector Address Register (Low Byte)  D1H      Set 1, Bank1

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

| .7 | Flash Memory Sector Address Bit (Low Byte) |
|---|---|
| | The 7$^{th}$ bit to select a sector of Flash ROM |

| .6–.0 | Not used for the S3C828B/C8289/C8285 |
|---|---|

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

# FMUSR — Flash Memory User Programming Enable Register  FFH          Set 1, Bank1

| Bit Identifier | .7 | .6 | .5 | .4 | .3 | .2 | .1 | .0 |
|---|---|---|---|---|---|---|---|---|
| RESET Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 |
|---|---|---|---|---|---|---|---|---|
| Others | | | | | | | | 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 P0.4–0.7**

| 0 | Disable (mask) |
|---|---|
| 1 | Enable (unmask) |

.6                        **Interrupt Level 6 (IRQ6) Enable Bit; External Interrupts P0.0–0.3**

| 0 | Disable (mask) |
|---|---|
| 1 | Enable (unmask) |

.5                        **Interrupt Level 5 (IRQ5) Enable Bit;** UART Transmit, UART Receive, Watch Timer

| 0 | Disable (mask) |
|---|---|
| 1 | Enable (unmask) |

.4                        **Interrupt Level 4 (IRQ4) Enable Bit; SIO**

| 0 | Disable (mask) |
|---|---|
| 1 | Enable (unmask) |

.3                        **Interrupt Level 3 (IRQ3) Enable Bit; Timer 1 Match/Capture or Overflow**

| 0 | Disable (mask) |
|---|---|
| 1 | Enable (unmask) |

.2                        **Interrupt Level 2 (IRQ2) Enable Bit; Timer 0 Match**

| 0 | Disable (mask) |
|---|---|
| 1 | Enable (unmask) |

.1                        **Interrupt Level 1 (IRQ1) Enable Bit; Timer B Match**

| 0 | Disable (mask) |
|---|---|
| 1 | Enable (unmask) |

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

| 0 | Disable (mask) |
|---|---|
| 1 | Enable (unmask) |

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

SAMSUNG
ELECTRONICS

# INTPND — Interrupt Pending Register                          F9H        Set 1, Bank0

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

**Addressing Mode**    Register addressing mode only

| .7–.6 | Not used for the S3C828B/C8289/C8285 |
|---|---|

**.5**    **Rx Interrupt Pending Bit (for UART)**

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

**.4**    **Tx Interrupt Pending Bit (for UART)**

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

**.3**    **Timer 1 Match/Capture Interrupt Pending Bit**

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

**.2**    **Timer 1 Overflow Interrupt Pending Bit**

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

**.1**    **Timer A Match/Capture Interrupt Pending Bit**

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

**.0**    **Timer A Overflow Interrupt Pending Bit**

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

SAMSUNG
ELECTRONICS

# 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, Bank0

| 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**

| | | | |
|---|---|---|---|
| 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

SAMSUNG
ELECTRONICS

# 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 Interrupts P0.4–0.7**

| 0 | Not pending |
|---|---|
| 1 | Pending |

**.6**                    **Level 6 (IRQ6) Request Pending Bit; External Interrupts P0.0–0.3**

| 0 | Not pending |
|---|---|
| 1 | Pending |

**.5**                    **Level 5 (IRQ5) Request Pending Bit; UART Transmit, UART Receive, Watch Timer**

| 0 | Not pending |
|---|---|
| 1 | Pending |

**.4**                    **Level 4 (IRQ4) Request Pending Bit; SIO**

| 0 | Not pending |
|---|---|
| 1 | Pending |

**.3**                    **Level 3 (IRQ3) Request Pending Bit; Timer 1 Match/Capture or Overflow**

| 0 | Not pending |
|---|---|
| 1 | Pending |

**.2**                    **Level 2 (IRQ2) Request Pending Bit; Timer 0 Match**

| 0 | Not pending |
|---|---|
| 1 | Pending |

**.1**                    **Level 1 (IRQ1) Request Pending Bit; Timer B Match**

| 0 | Not pending |
|---|---|
| 1 | Pending |

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

| 0 | Not pending |
|---|---|
| 1 | Pending |

# LCON — LCD Control Register                         D0H        Set 1, Bank0

| 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 | $fw/2^8$ (128 Hz) |
|---|---|---|
| 0 | 1 | $fw/2^7$ (256 Hz) |
| 1 | 0 | $fw/2^6$ (512 Hz) |
| 1 | 1 | $fw/2^5$ (1024 Hz) |

**.4–.2**           **LCD Duty and Bias Selection Bits** (note)

| 0 | 0 | 0 | 1/8duty, 1/4 bias |
|---|---|---|---|
| 0 | 0 | 1 | 1/4duty, 1/3 bias |
| 0 | 1 | 0 | 1/3duty, 1/3 bias |
| 0 | 1 | 1 | 1/3duty, 1/2 bias |
| 1 | x | x | 1/2duty, 1/2 bias |

**.1**              | Not used for the S3C828B/C8289/C8285 |
|---|

**.0**              **LCD Display Control Bits**

| 0 | All LCD signals are low (Turn off the P-Tr) |
|---|---|
| 1 | Turn display on (Turn on the P-Tr) |

**NOTES:**
1.  "x" means don't care.
2.  When 1/3 bias is selected, the bias levels are set as $V_{LC0}$, $V_{LC1}$, $V_{LC2}$ ($V_{LC3}$), and $V_{SS}$.
3.  When 1/2bias is selected, the bias levels are set as $V_{LC0}$, $V_{LC1}$ ($V_{LC2}$, $V_{LC3}$), and $V_{SS}$.

**SAMSUNG**
**ELECTRONICS**

# OSCCON — Oscillator Control Register                    FAH        Set 1, Bank0

| 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 | Select normal circuit for sub oscillator |
|---|---|
| 1 | Select power saving circuit for sub oscillator (note)<br>(Automatically cleared to "0" when the sub oscillator is stopped by OSCCON.2 or the CPU is entered into stop mode in sub operating mode) |

**.6–.4**              

| Not used for the S3C828B/C8289/C8285 |
|---|

**.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 S3C828B/C8289/C8285 |
|---|

**.0**                 **System Clock Selection Bit**

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

**NOTE:**   A capacitor (0.1μF) should be connected between VREG and GND when the sub-oscillator is used to power saving mode (OSCCON.7 = "1").

# P0CONH — Port 0 Control Register (High Byte)                E0H        Set 1, Bank 1

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

**.7–.6**          **P0.7/INT7**

| | | |
|---|---|---|
| 0 | 0 | Schmitt trigger input mode |
| 0 | 1 | Schmitt trigger input mode with pull-up resistor |
| 1 | 0 | Output mode, open-drain |
| 1 | 1 | Output mode, push-pull |

**.5–.4**          **P0.6/INT6**

| | | |
|---|---|---|
| 0 | 0 | Schmitt trigger input mode |
| 0 | 1 | Schmitt trigger input mode with pull-up resistor |
| 1 | 0 | Output mode, open-drain |
| 1 | 1 | Output mode, push-pull |

**.3–.2**          **P0.5/INT5**

| | | |
|---|---|---|
| 0 | 0 | Schmitt trigger input mode |
| 0 | 1 | Schmitt trigger input mode with pull-up resistor |
| 1 | 0 | Output mode, open-drain |
| 1 | 1 | Output mode, push-pull |

**.1–.0**          **P0.4/INT4**

| | | |
|---|---|---|
| 0 | 0 | Schmitt trigger input mode |
| 0 | 1 | Schmitt trigger input mode with pull-up resistor |
| 1 | 0 | Output mode, open-drain |
| 1 | 1 | Output mode, push-pull |

SAMSUNG
ELECTRONICS

# P0CONL — Port 0 Control Register (Low Byte)          E1H          Set 1, Bank 1

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

**.7–.6**      **P0.3/INT3**

| 0 | 0 | Schmitt trigger input mode |
|---|---|---|
| 0 | 1 | Schmitt trigger input mode with pull-up resistor |
| 1 | 0 | Output mode, open-drain |
| 1 | 1 | Output mode, push-pull |

**.5–.4**      **P0.2/INT2**

| 0 | 0 | Schmitt trigger input mode |
|---|---|---|
| 0 | 1 | Schmitt trigger input mode with pull-up resistor |
| 1 | 0 | Output mode, open-drain |
| 1 | 1 | Output mode, push-pull |

**.3–.2**      **P0.1/INT1**

| 0 | 0 | Schmitt trigger input mode |
|---|---|---|
| 0 | 1 | Schmitt trigger input mode with pull-up resistor |
| 1 | 0 | Output mode, open-drain |
| 1 | 1 | Output mode, push-pull |

**.1–.0**      **P0.0/INT0**

| 0 | 0 | Schmitt trigger input mode |
|---|---|---|
| 0 | 1 | Schmitt trigger input mode with pull-up resistor |
| 1 | 0 | Output mode, open-drain |
| 1 | 1 | Output mode, push-pull |

SAMSUNG
ELECTRONICS

# P0INTH — Port 0 Interrupt Control Register (High Byte)        E2H        Set 1, Bank1

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

**Addressing Mode**    Register addressing mode only

**.7–.6**              **P0.7/External interrupt (INT7) Enable 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**              **P0.6/External interrupt (INT6) Enable 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**              **P0.5/External interrupt (INT5) Enable 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**              **P0.4/External interrupt (INT4) Enable 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 |

**SAMSUNG**
**ELECTRONICS**

# P0INTL — Port 0 Interrupt Control Register (Low Byte)          E3H          Set 1, Bank1

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

**Addressing Mode**   Register addressing mode only

**.7–.6**          **P0.3/External interrupt (INT3) Enable 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) Enable 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) Enable 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) Enable 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 |

**SAMSUNG**
**ELECTRONICS**

# P0PND — Port 0 Interrupt Pending Register          E4H        Set 1, Bank1

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

**Addressing Mode**　　Register addressing mode only

**.7**　　　　**P0.7/External Interrupt (INT7) Pending Bit**

| 0 | Clear pending bit (when write) |
|---|---|
| 1 | P0.7/INT7 interrupt request is pending (when read) |

**.6**　　　　**P0.6/External Interrupt (INT6) Pending Bit**

| 0 | Clear pending bit (when write) |
|---|---|
| 1 | P0.6/INT6 interrupt request is pending (when read) |

**.5**　　　　**P0.5/External Interrupt (INT5) Pending Bit**

| 0 | Clear pending bit (when write) |
|---|---|
| 1 | P0.5/INT5 interrupt request is pending (when read) |

**.4**　　　　**P0.4/External Interrupt (INT4) Pending Bit**

| 0 | Clear pending bit (when write) |
|---|---|
| 1 | P0.4/INT4 interrupt request is pending (when read) |

**.3**　　　　**P0.3/External Interrupt (INT3) Pending Bit**

| 0 | Clear pending bit (when write) |
|---|---|
| 1 | P0.3/INT3 interrupt request is pending (when read) |

**.2**　　　　**P0.2/External Interrupt (INT2) Pending Bit**

| 0 | Clear pending bit (when write) |
|---|---|
| 1 | P0.2/INT2 interrupt request is pending (when read) |

**.1**　　　　**P0.1/External Interrupt (INT1) Pending Bit**

| 0 | Clear pending bit (when write) |
|---|---|
| 1 | P0.1/INT1 interrupt request is pending (when read) |

**.0**　　　　**P0.0/External Interrupt (INT0) Pending Bit**

| 0 | Clear pending bit (when write) |
|---|---|
| 1 | P0.0/INT0 interrupt request is pending (when read) |

# P1CONH — Port 1 Control Register (High Byte)          E5H          Set 1, Bank1

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

| .7–.6 | Not used for the S3C828B/C8289/C8285 |
|---|---|

**.5–.4**    **P1.6/SI**

| 0 | 0 | Schmitt trigger input mode (SI) |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Not used for the S3C828B/C8289/C8285 |

**.3–.2**    **P1.5/SCK**

| 0 | 0 | Schmitt trigger input mode (SCK input) |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SCK output) |

**.1–.0**    **P1.4/SO**

| 0 | 0 | Schmitt trigger input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SO) |

SAMSUNG
ELECTRONICS

# P1CONL — Port 1 Control Register (Low Byte)          E6H          Set 1, Bank1

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

**.7–.6**  **P1.3/BUZ**

| 0 | 0 | Schmitt trigger input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (BUZ) |

**.5–.4**  **P1.2/T1OUT/T1PWM**

| 0 | 0 | Schmitt trigger input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (T1OUT/T1PWM) |

**.3–.2**  **P1.1/T1CLK**

| 0 | 0 | Schmitt trigger input mode (T1CLK) |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Not used for the S3C828B/C8289/C8285 |

**.1–.0**  **P1.0/T1CAP**

| 0 | 0 | Schmitt trigger input mode (T1CAP) |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Not used for the S3C828B/C8289/C8285 |

**SAMSUNG**
**ELECTRONICS**

# P1PUR — Port 1 Pull-up Resistor Enable Register          E7H          Set 1, Bank1

| 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 S3C828B/C8289/C8285 |
|---|

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

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

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

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

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

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

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

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

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

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

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

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

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

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**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.

**SAMSUNG**
**ELECTRONICS**

# P2CONH — Port 2 Control Register (High Byte) E8H Set 1, Bank1

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

.7–.6 **P2.7/AD7/V$_{BLDREF}$**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (AD7/V$_{BLDREF}$) |

.5-.4 **P2.6/AD6**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (AD6) |

.3–.2 **P2.5/AD5**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (AD5) |

.1–.0 **P2.4/AD4**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (AD4) |

SAMSUNG
ELECTRONICS

# P2CONL — Port 2 Control Register (Low Byte)          E9H          Set 1, Bank1

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

**.7–.6**          **P2.3/AD3**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (AD3) |

**.5–.4**          **P2.2/AD2**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (AD2) |

**.3–.2**          **P2.1/AD1**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (AD1) |

**.1–.0**          **P2.0/AD0**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (AD0) |

SAMSUNG
ELECTRONICS

# P3CONH — Port 3 Control Register (High Byte)          EAH       Set 1, Bank1

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

**Addressing Mode**     Register addressing mode only

**.7–.6**

| Not used for the S3C828B/C8289/C8285 |
|---|

**.5**     **P3.1/TAOUT/TAPWM/SEG35 (P3CONL.3-.2 = "11" only)**

| 0 | TAOUT/TAPWM out |
|---|---|
| 1 | SEG35 out |

**.4**     **P3.0/TBPWM/SEG34 (P3CONL.3-.2 = "11" only)**

| 0 | TBPWM out |
|---|---|
| 1 | SEG34 out |

**.3–.2**     **P3.5/RxD**

| 0 | 0 | Input mode (RxD) |
|---|---|---|
| 0 | 1 | Input mode, pull-up (RxD) |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (RxD out) |

**.1–.0**     **P3.4/TxD**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (TxD) |

SAMSUNG
ELECTRONICS

# P3CONL — Port 3 Control Register (Low Byte)          EBH          Set 1, Bank1

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

**.7–.6**          **P3.3/TACAP/SEG37**

| 0 | 0 | Input mode (TACAP) |
|---|---|---|
| 0 | 1 | Input mode, pull-up (TACAP) |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG37) |

**.5–.4**          **P3.2/TACLK/SEG36**

| 0 | 0 | Input mode (TACLK) |
|---|---|---|
| 0 | 1 | Input mode, pull-up (TACLK) |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG36) |

**.3–.2**          **P3.1/TAOUT/TAPWM/SEG35**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (TAOUT/TAPWM/SEG35) |

**.1–.0**          **P3.0/TBPWM/SEG34**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (TBPWM/SEG34) |

SAMSUNG
ELECTRONICS

# P4CONH — **Port 4 Control Register (High Byte)**          **ECH**      **Set 1, Bank1**

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

**.7–.6**          **P4.7/SEG25**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG25) |

**.5–.4**          **P4.6/SEG24**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG24) |

**.3–.2**          **P4.5/SEG23**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG23) |

**.1–.0**          **P4.4/SEG22**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG22) |

**SAMSUNG**
**ELECTRONICS**

# P4CONL — Port 4 Control Register (Low Byte) EDH Set 1, Bank1

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

.7–.6 **P4.3/SEG21**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG21) |

.5–.4 **P4.2/SEG20**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG20) |

.3–.2 **P4.1/SEG19**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG19) |

.1–.0 **P4.0/SEG18**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG18) |

**SAMSUNG ELECTRONICS**

# P4PUR — Port 4 Pull-up Resistor Enable Register          EEH          Set 1, Bank1

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

.7          **P4.7 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

.6          **P4.6 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

.5          **P4.5 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

.4          **P4.4 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

.3          **P4.3 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

.2          **P4.2 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

.1          **P4.1 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

.0          **P4.0 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**NOTE:**  A pull-up resistor of port 4 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

SAMSUNG
ELECTRONICS

# P5CONH — Port 5 Control Register (High Byte)              F9H        Set 1, Bank1

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

**.7–.6**          **P5.7/SEG33**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG33) |

**.5–.4**          **P5.6/SEG32**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG32) |

**.3–.2**          **P5.5/SEG31**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG31) |

**.1–.0**          **P5.4/SEG30**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG30) |

**SAMSUNG**
**ELECTRONICS**

# P5CONL — Port 5 Control Register (Low Byte)          FAH        Set 1, Bank1

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

.7–.6          **P5.3/SEG29**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG29) |

.5–.4          **P5.2/SEG28**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG28) |

.3–.2          **P5.1/SEG27**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG27) |

.1–.0          **P5.0/SEG26**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Output mode, N-channel open-drain |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG26) |

**SAMSUNG**
ELECTRONICS

# P5PUR — Port 5 Pull-up Resistor Enable Register          EFH          Set 1, Bank1

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

**.7**                     **P5.7 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**.6**                     **P5.6 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**.5**                     **P5.5 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**.4**                     **P5.4 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**.3**                     **P5.3 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**.2**                     **P5.2 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**.1**                     **P5.1 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**.0**                     **P5.0 Pull-up Resistor Enable Bit**

| 0 | Pull-up disable |
|---|---|
| 1 | Pull-up enable |

**NOTE:**  A pull-up resistor of port 5 is automatically disabled only when the corresponding pin is selected as push-pull output or alternative function.

# P6CONH — Port 6 Control Register (High Byte)          FBH       Set 1, Bank1

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

.7–.6          **P6.7/SEG17**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG17) |

.5–.4          **P6.6/SEG16**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG16) |

.3–.2          **P6.5/SEG15**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG15) |

.1–.0          **P6.4/SEG14**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG14) |

SAMSUNG
ELECTRONICS

# P6CONL — Port 6 Control Register (Low Byte)          FCH        Set 1, Bank1

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

.7–.6          **P6.3/SEG13**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG13) |

.5–.4          **P6.2/SEG12**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG12) |

.3–.2          **P6.1/SEG11**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG11) |

.1–.0          **P6.0/SEG10**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG10) |

**SAMSUNG**
**ELECTRONICS**

# P7CON — **Port 7 Control Register**                          **FDH         Set 1, Bank1**

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

**.7–.6**        **P7.3/SEG9**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG9) |

**.5–.4**        **P7.2/SEG8**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG8) |

**.3–.2**        **P7.1/SEG7**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG7) |

**.1–.0**        **P7.0/SEG6**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (SEG6) |

**SAMSUNG**
**ELECTRONICS**

# P8CON — Port 8 Control Register              FEH       Set 1, Bank1

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

.7–.6                    **P8.7-P8.4/COM7-COM4/SEG5-SEG2**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (COM7-COM4/SEG5-SEG2) |

.5–.4                    **P8.3/COM3/SEG1**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (COM3/SEG1) |

.3–.2                    **P8.2/COM2/SEG0**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (COM2/SEG0) |

.1–.0                    **P8.1-P8.0/COM1-COM0**

| 0 | 0 | Input mode |
|---|---|---|
| 0 | 1 | Input mode, pull-up |
| 1 | 0 | Output mode, push-pull |
| 1 | 1 | Alternative function (COM1-COM0) |

**SAMSUNG**
**ELECTRONICS**

# PP — Register Page Pointer                    DFH            Set 1

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

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

| 0 | 0 | 0 | 0 | Destination: page 0 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | Destination: page 1 |
| 0 | 0 | 1 | 0 | Destination: page 2 (not used for the S3C8285) |
| 0 | 0 | 1 | 1 | Destination: page 3 (not used for the S3C8285) |
| 0 | 1 | 0 | 0 | Destination: page 4 (not used for the S3C8289/C8285) |
| 0 | 1 | 0 | 1 | Destination: page 5 (not used for the S3C8289/C8285) |
| 0 | 1 | 1 | 0 | Destination: page 6 (not used for the S3C8289/C8285) |
| 0 | 1 | 1 | 1 | Destination: page 7 (not used for the S3C8289/C8285) |
| 1 | 0 | 0 | 0 | Destination: page 8 (not used for the S3C8289/C8285) |
| 1 | 0 | 0 | 1 | Destination: page 9 (not used for the S3C8289/C8285) |
| 1 | 1 | 1 | 1 | Destination: page 15 |
| Others | | | | Not used for the S3C828B/C8289/C8285 |

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

| 0 | 0 | 0 | 0 | Source: page 0 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | Source: page 1 |
| 0 | 0 | 1 | 0 | Source: page 2 (not used for the S3C8285) |
| 0 | 0 | 1 | 1 | Source: page 3 (not used for the S3C8285) |
| 0 | 1 | 0 | 0 | Source: page 4 (not used for the S3C8289/C8285) |
| 0 | 1 | 0 | 1 | Source: page 5 (not used for the S3C8289/C8285) |
| 0 | 1 | 1 | 0 | Source: page 6 (not used for the S3C8289/C8285) |
| 0 | 1 | 1 | 1 | Source: page 7 (not used for the S3C8289/C8285) |
| 1 | 0 | 0 | 0 | Source: page 8 (not used for the S3C8289/C8285) |
| 1 | 0 | 0 | 1 | Source: page 9 (not used for the S3C8289/C8285) |
| 1 | 1 | 1 | 1 | Source: page 15 |
| Others | | | | Not used for the S3C828B/C8289/C8285 |

**NOTES:**
1. In the S3C828B microcontroller, the internal register file is configured as eleven pages (pages 0-9,15). The pages 0-9 are used for general purpose register file.
2. In the S3C8289 microcontroller, the internal register file is configured as eleven pages (pages 0-3,15). The pages 0-3 are used for general purpose register file.
3. In the S3C8285 microcontroller, the internal register file is configured as eleven pages (pages 0-1,15). The pages 0-1 are used for general purpose register file.
4. The page 15 of S3C828B/C8289/C8285 is used for LCD data register or general purpose register.

**SAMSUNG**
**ELECTRONICS**

# 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 S3C828B/C8289/C8285

# 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 S3C828B/C8289/C8285

# SIOCON — SIO Control Register                    E0H        Set 1, Bank0

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

.7                **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 condition (when write) |
|---|---|
| 1 | Interrupt is pending |

SAMSUNG
ELECTRONICS

# 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, Bank0

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

.7–.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, You must set this STPCON register as "10100101b". Otherwise the STOP instruction will not execute as well as reset will be generated.

SAMSUNG
ELECTRONICS

# SYM — System Mode Register                              DEH                Set 1

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

**.7**

| Not used, But you must keep "0" |
|---|

**.6–.5**

| Not used for the S3C828B/C8289/C8285 |
|---|

**.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).

**SAMSUNG**
**ELECTRONICS**

# T0CON — Timer 0 Control Register                    E3H        Set 1, Bank0

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

**Addressing Mode**    Register addressing mode only

**.7–.5**              **Timer 0 Input Clock Selection Bits**

| 0 | 0 | 0 | TBOF |
|---|---|---|---|
| 0 | 0 | 1 | fxx/256 |
| 0 | 1 | 0 | fxx/64 |
| 0 | 1 | 1 | fxx/8 |
| 1 | x | x | fxx |

**.4**

| Not used for the S3C828B/C8289/C8285 |
|---|

**.3**                 **Timer 0 Counter Clear Bit**

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

**.2**                 **Timer 0 Counter Enable Bit**

| 0 | Disable counting operation |
|---|---|
| 1 | Enable counting operation |

**.1**                 **Timer 0 Match Interrupt Enable Bit**

| 0 | Disable interrupt |
|---|---|
| 1 | Enable interrupt |

**.0**                 **Timer 0 Interrupt Pending Bit**

| 0 | No timer 0 interrupt pending (when read),<br>Clear timer 0 interrupt pending bit (when write) |
|---|---|
| 1 | T0 interrupt is pending |

**NOTE:** The T0CON.3 value is automatically cleared to "0" after being cleared counter.

SAMSUNG
ELECTRONICS

# T1CON — Timer 1 Control Register          EBH      Set 1, Bank0

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

**.7–.5**                     **Timer 1 Input Clock Selection Bits**

| 0 | 0 | 0 | fxx/1024 |
|---|---|---|---|
| 0 | 0 | 1 | fxx/256 |
| 0 | 1 | 0 | fxx/64 |
| 0 | 1 | 1 | fxx/8 |
| 1 | 0 | 0 | fxx/1 |
| 1 | 0 | 1 | External clock (T1CLK) falling edge |
| 1 | 1 | 0 | External clock (T1CLK) rising edge |
| 1 | 1 | 1 | Counter stop |

**.4–.3**                     **Timer 1 Operating Mode Selection Bits**

| 0 | 0 | Interval mode (T1OUT) |
|---|---|---|
| 0 | 1 | Capture mode (Capture on rising edge, counter running, OVF can occur) |
| 1 | 0 | Capture mode (Capture on falling edge, counter running, OVF can occur) |
| 1 | 1 | PWM mode (OVF and match interrupt can occur) |

**.2**                     **Timer 1 Counter Enable Bit**

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

**.1**                     **Timer 1 Match/Capture Interrupt Enable Bit**

| 0 | Disable interrupt |
|---|---|
| 1 | Enable interrupt |

**.0**                     **Timer 1 Overflow Interrupt Enable Bit**

| 0 | Disable overflow interrupt |
|---|---|
| 1 | Enable overflow interrupt |

**NOTE:** The T1CON.2 value is automatically cleared to "0" after being cleared counter.

# TACON — Timer A Control Register                    **E8H**      **Set 1, Bank0**

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

**.7–.5**                    **Timer A Input Clock Selection Bits**

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | fxx/1024 |
| 0 | 0 | 1 | fxx/256 |
| 0 | 1 | 0 | fxx/64 |
| 0 | 1 | 1 | fxx/8 |
| 1 | 0 | 0 | fxx(system clock) |
| 1 | 0 | 1 | External clock (TACLK) falling edge |
| 1 | 1 | 0 | External clock (TACLK) rising edge |
| 1 | 1 | 1 | Counter stop |

**.4–.3**                    **Timer A Operating Mode Selection Bits**

| | | |
|---|---|---|
| 0 | 0 | Internal mode (TAOUT) |
| 0 | 1 | Capture mode (capture on rising edge, counter running, OVF can occur) |
| 1 | 0 | Capture mode (capture on falling edge, counter running, OVF can occur) |
| 1 | 1 | PWM mode (OVF interrupt can occur) |

**.2**                       **Timer A Overflow Interrupt Enable Bit**

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

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

| | |
|---|---|
| 0 | Disable interrupt |
| 1 | Enable interrupt |

**.0**                       **Timer A Overflow Interrupt Enable Bit**

| | |
|---|---|
| 0 | Disable overflow interrupt |
| 1 | Enable overflow interrupt |

**NOTE:**  The TACON.2 value is automatically cleared to "0" after being cleared the counter.

SAMSUNG
ELECTRONICS

# TBCON — Timer B Control Register                                    F2H        Set 1, Bank0

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

**.7–.6**                 **Timer B Input Clock Selection Bits**

| 0 | 0 | fxx/1 |
|---|---|---|
| 0 | 1 | fxx/2 |
| 1 | 0 | fxx/4 |
| 1 | 1 | fxx/8 |

**.5–.4**                 **Timer B Interrupt Time Selection Bits**

| 0 | 0 | Generating after low data is borrowed. |
|---|---|---|
| 0 | 1 | Generating after high data is borrowed. |
| 1 | 0 | Generating after low and high data are borrowed. |
| 1 | 1 | Not available |

**.3**                 **Timer B Interrupt Enable Bit**

| 0 | Disable Interrupt |
|---|---|
| 1 | Enable Interrupt |

**.2**                 **Timer B Start/Stop Bit**

| 0 | Stop timer B |
|---|---|
| 1 | Start timer B |

**.1**                 **Timer B Mode Selection Bit**

| 0 | One-shot mode |
|---|---|
| 1 | Repeating mode |

**.0**                 **Timer B Output flip-flop Control Bit**

| 0 | TBOF is low (TBPWM: low level for low data, high level for high data) |
|---|---|
| 1 | TBOF is high (TBPWM: high level for low data, low level for high data) |

# UARTCON — UART Control Register                                        F6H        Set 1, Bank0

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

**.7–.6**  **UART Mode Selection Bits**

| 0 | 0 | Mode 0: shift register (fxx/(16 × (BRDATA+1))) |
|---|---|---|
| 0 | 1 | Mode 1: 8-bit UART (fxx/(16 × (BRDATA+1))) |
| 1 | 0 | Mode 2: 9-bit UART (fxx/16) |
| 1 | 1 | Mode 3: 9-bit UART (fxx/(16 × (BRDATA+1))) |

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

| 0 | Disable |
|---|---|
| 1 | Enable |

**.4**  **Serial Data Receive Enable Bit**

| 0 | Disable |
|---|---|
| 1 | Enable |

**.3**  **TB8**

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

**.2**  **RB8**

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

**.1**  **Receive Interrupt Enable Bit**

| 0 | Disable Rx interrupt |
|---|---|
| 1 | Enable Rx interrupt |

**.0**  **Transmit Interrupt Enable Bit**

| 0 | Disable Tx interrupt |
|---|---|
| 1 | Enable Tx interrupt |

**NOTES:**

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

SAMSUNG
ELECTRONICS

# WTCON — Watch Timer Control Register                     D1H        Set 1, Bank0

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

**.7** **Watch Timer Clock Selection Bit**

| 0 | Main system clock divided by $2^7$ (fxx/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 1.0s |
|---|---|---|
| 0 | 1 | Set watch timer interrupt to 0.5s |
| 1 | 0 | Set watch timer interrupt to 0.25s |
| 1 | 1 | Set watch timer interrupt to 3.91ms |

**.1** **Watch Timer Enable Bit**

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

**.0** **Watch Timer Interrupt Pending Bit**

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

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

**SAMSUNG**
**ELECTRONICS**

# 5 INTERRUPT STRUCTURE

## OVERVIEW

The S3C8-series interrupt structure has three basic components: levels, vectors, and sources. The SAM8 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 S3C828B/C8289/C8285 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. S3C828B/C8289/C8285 uses eighteen 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 S3C828B/C8289/C8285 interrupt structure, there are eighteen 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 S3C828B/C8289/C8285 microcontroller, two interrupt types are implemented.
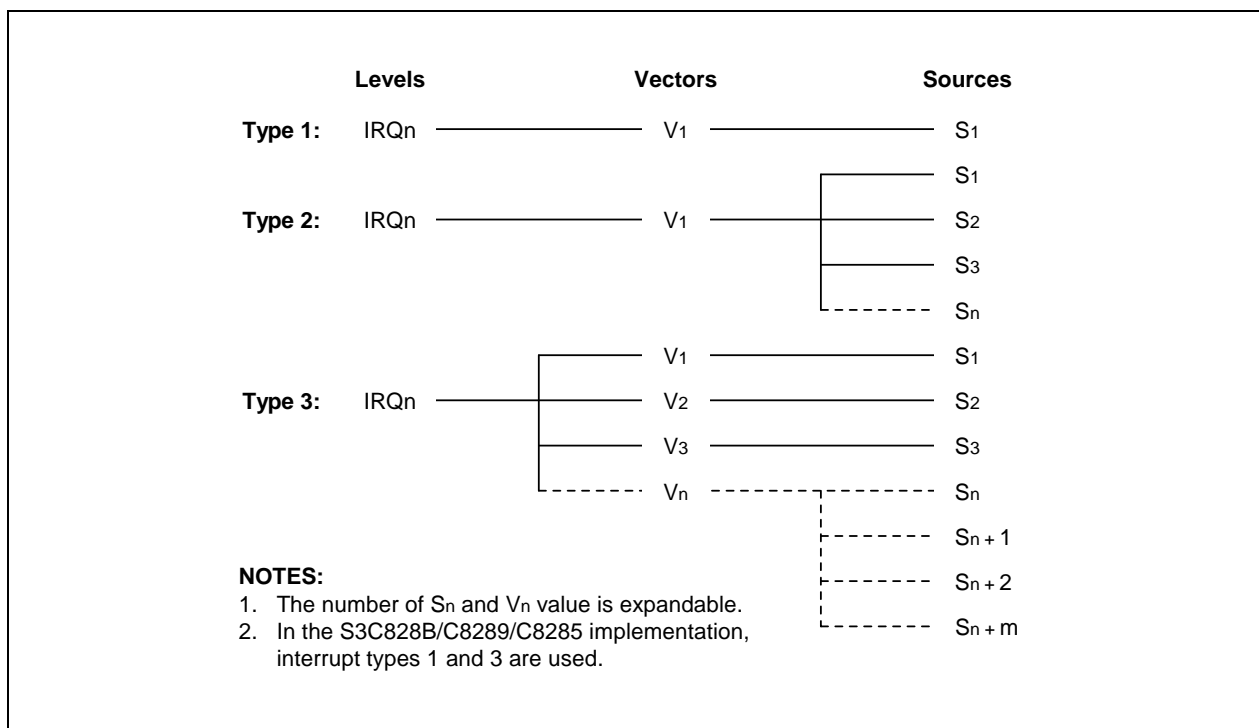


**Figure 5-1. S3C8-Series Interrupt Types**

## S3C828B/C8289/C8285 INTERRUPT STRUCTURE

The S3C828B/F828B/C8289/F8289/C8285/F8285 microcontroller supports nineteen interrupt sources. All nineteen 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 | DCH | Timer A match/capture | S/W |
| | DEH | Timer A overflow | H/W,S/W |
| IRQ1 | E0H | Timer B match | H/W |
| IRQ2 | E2H | Timer 0 match | S/W |
| IRQ3 | E4H | Timer 1 match/capture | S/W |
| | E6H | Timer 1 overflow | H/W,S/W |
| IRQ4 | E8H | SIO interrupt | S/W |
| IRQ5 | EAH | UART data transmit | S/W |
| | ECH | UART data receive | S/W |
| | EEH | Watch timer overflow | S/W |
| IRQ6 | F0H | P0.0 External interrupt | S/W |
| | F2H | P0.1 External interrupt | S/W |
| | F4H | P0.2 External interrupt | S/W |
| | F6H | P0.3 External interrupt | S/W |
| IRQ7 | F8H | P0.4 External interrupt | S/W |
| | FAH | P0.5 External interrupt | S/W |
| | FCH | P0.6 External interrupt | S/W |
| | FEH | P0.7 External interrupt | S/W |

**NOTES:**
1. Within a given interrupt level, the low vector address has high priority.
   For example, E0H has higher priority than E2H within the level IRQ.0 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. S3C828B/C8289/C8285 Interrupt Structure**

SAMSUNG
ELECTRONICS

**INTERRUPT VECTOR ADDRESSES**

All interrupt vector addresses for the S3C828B/F828B/C8289/F8289/C8285/F8285 interrupt structure are stored in the vector address area of the internal 64-Kbyte ROM, 0H–FFFFH, or 16,32-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.
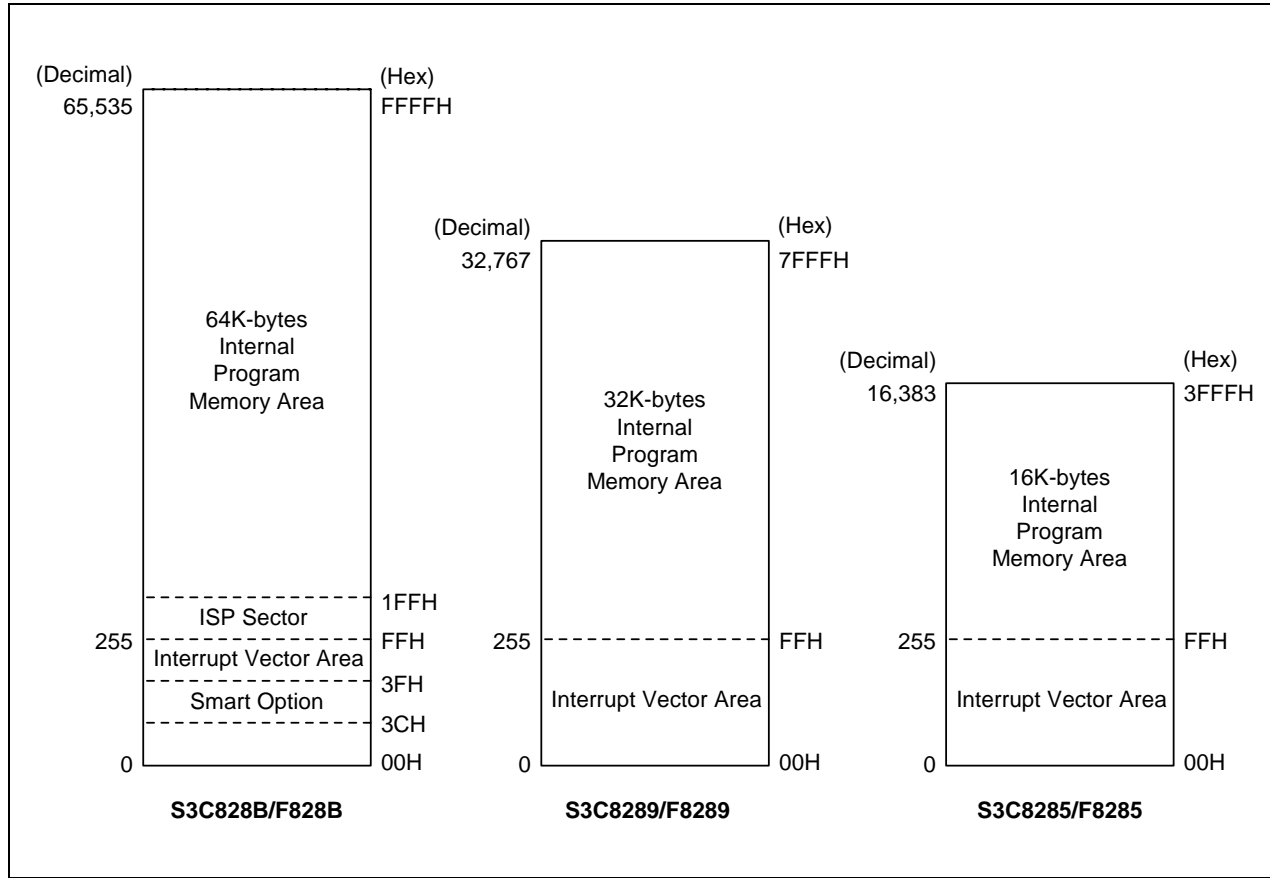


**Figure 5-3. ROM Vector Address Area**

**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 | – | √ | |
| 220 | DCH | Timer A match/capture | IRQ0 | 0 | | √ |
| 222 | DEH | Timer A overflow | | 1 | √ | √ |
| 224 | E0H | Timer B match | IRQ1 | – | √ | |
| 226 | E2H | Timer 0 match | IRQ2 | – | | √ |
| 228 | E4H | Timer 1 match/capture | IRQ3 | 0 | | √ |
| 230 | E6H | Timer 1 overflow | | 1 | √ | √ |
| 232 | E8H | SIO interrupt | IRQ4 | – | | √ |
| 234 | EAH | UART data transmit | IRQ5 | 0 | | √ |
| 236 | ECH | UART data receive | | 1 | | √ |
| 238 | EEH | Watch timer overflow | | 2 | | √ |
| 240 | F0H | P0.0 external interrupt | IRQ6 | 0 | | √ |
| 242 | F2H | P0.1 external interrupt | | 1 | | √ |
| 244 | F4H | P0.2 external interrupt | | 2 | | √ |
| 246 | F6H | P0.3 external interrupt | | 3 | | √ |
| 248 | F8H | P0.4 external interrupt | IRQ7 | 0 | | √ |
| 250 | FAH | P0.5 external interrupt | | 1 | | √ |
| 252 | FCH | P0.6 external interrupt | | 2 | | √ |
| 254 | FEH | P0.7 external interrupt | | 3 | | √ |

**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.
3. Timer A or Timer 1 can not service two interrupt sources simultaneously, then only one interrupt source have to be used.

**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 S3C828B/F828B/C8289/F8289/C8285/F8285 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, dynamic global interrupt processing, and external interface control (An external memory interface is implemented in the S3C828B/F828B/C8289/F8289/C8285/F8285 microcontroller). |

**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.
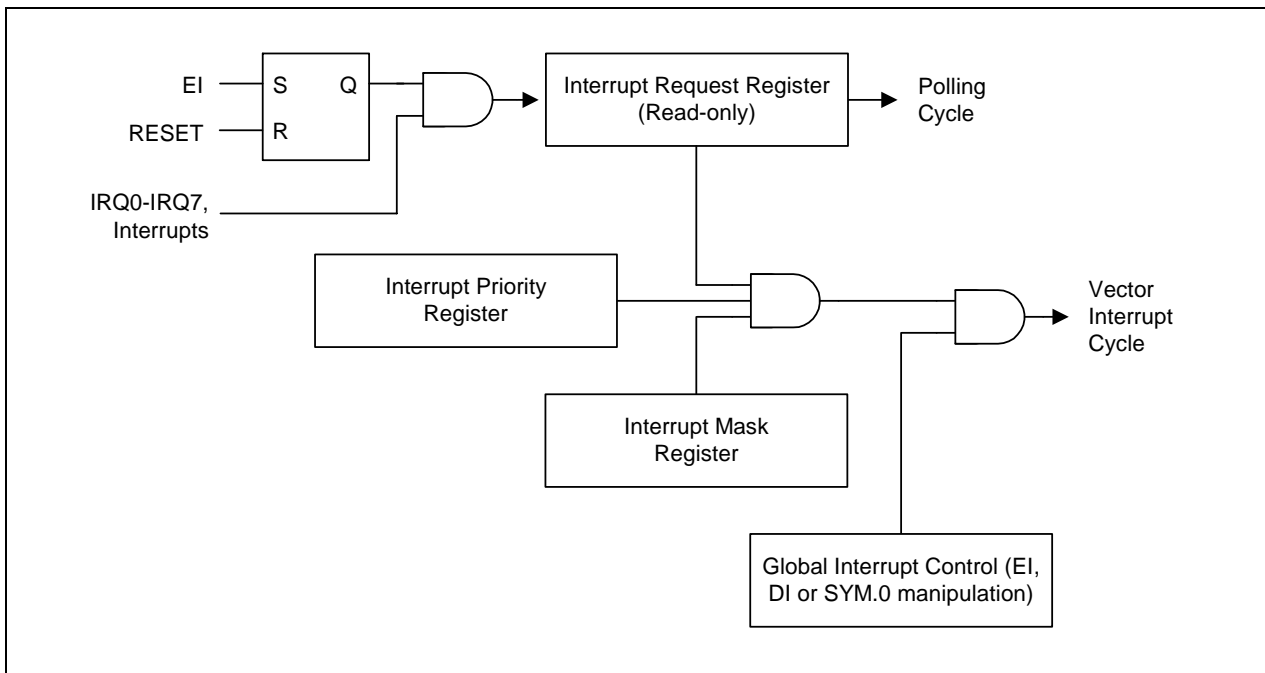


**Figure 5-4. Interrupt Function Diagram**

## 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 A match/capture<br>Timer A overflow | IRQ0 | TACON<br>TACNT<br>TADATA | E8H, bank 0<br>F9H, bank 0<br>EAH, bank 0 |
| Timer B match | IRQ1 | TBCON<br>TBDATAH, TBDATAL | F2H, bank 0<br>F0H, F1H, bank 0 |
| Timer 0 match | IRQ2 | T0CON<br>T0CNTH, T0CNTL,<br>T0DATAH, T0DATAL | E3H, bank 0<br>E4H, E5H, bank 0<br>E6H, E7H, bank 0 |
| Timer 1 match/capture<br>Timer 1 overflow | IRQ3 | T1CON<br>T1CNTH, T1CNTL<br>T1DATAH, T1DATAL | EBH, bank 0<br>ECH, EDH, bank 0<br>EEH, EFH, bank 0 |
| SIO interrupt | IRQ4 | SIOCON<br>SIODATA<br>SIOPS | E0H, bank 0<br>E1H, bank 0<br>E2H, bank 0 |
| UART data transmit<br>UART data receive<br><br>Watch timer overflow | IRQ5 | UARTCON<br>UDATA<br>BRDATA<br>WTCON | F6H, bank 0<br>F7H, bank 0<br>F8H, bank 0<br>D1H, bank 0 |
| P0.0 external interrupt<br>P0.1 external interrupt<br>P0.2 external interrupt<br>P0.3 external interrupt | IRQ6 | P0CONL<br>P0INTL<br>P0PND | E1H, bank 1<br>E3H, bank 1<br>E4H, bank 1 |
| P0.4 external interrupt<br>P0.5 external interrupt<br>P0.6 external interrupt<br>P0.7 external interrupt | IRQ7 | P0CONH<br>P0INTH<br>P0PND | E0H, bank 1<br>E2H, bank 1<br>E4H, bank 1 |

## SYSTEM MODE REGISTER (SYM)

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

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

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

```
                        System Mode Register (SYM)
                             DEH, Set 1, R/W

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


              Always logic "0"                    Global interrupt enable bit: (3)
                                                  0 = Disable all interrupts processing
        Not  used for the S3C828B/C8249/C8245     1 = Enable all interrupts processing

              Fast interrupt level
              selection bits: (1)                 Fast interrupt enable bit: (2)
                                                  0 = Disable fast interrupts processing
              0  0  0 = IRQ0                       1 = Enable fast interrupts processing
              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

         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.
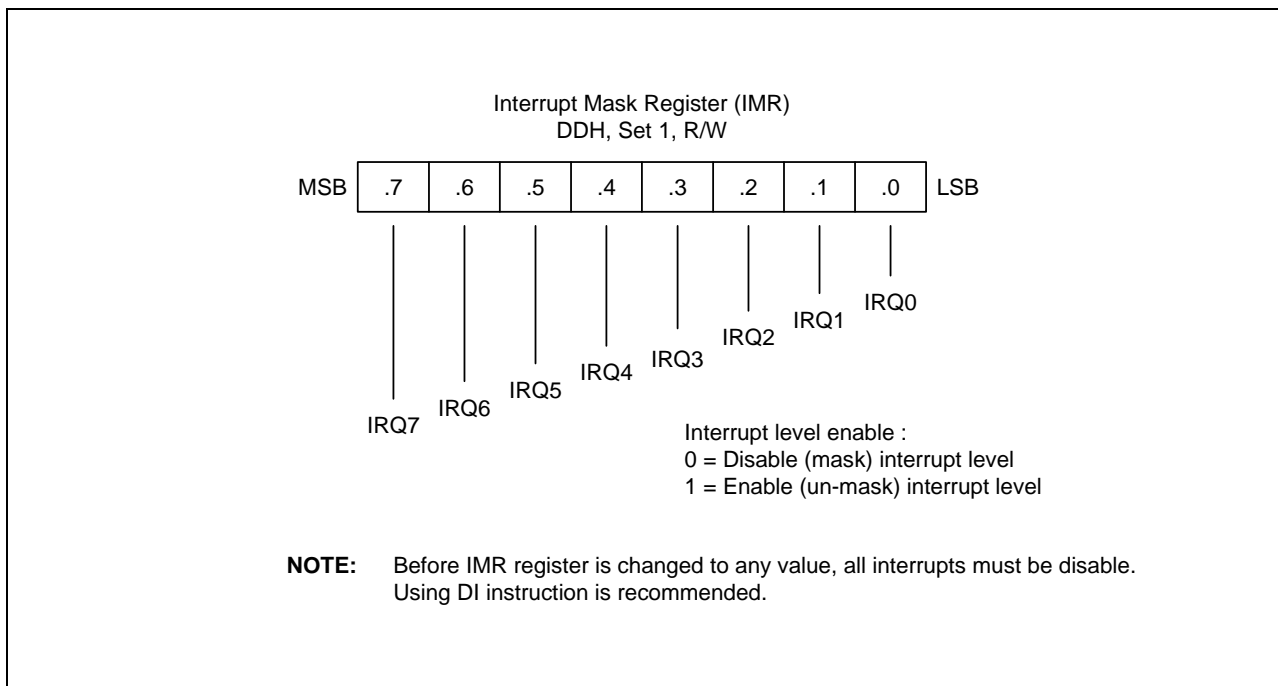


**Figure 5-6. Interrupt Mask Register (IMR)**

## 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
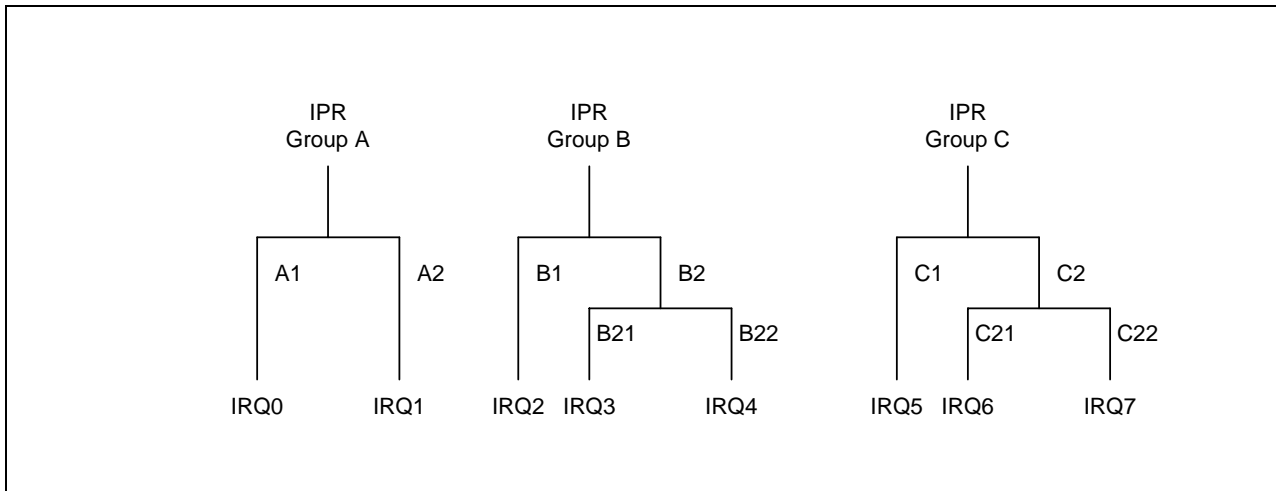


**Figure 5-7. Interrupt Request Priority Groups**

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.

Interrupt Priority Register (IPR)
FFH, Set 1, Bank 0, R/W

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

Group priority:

D7  D4  D1

0   0   0  = 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  = Undefined

Group A:
0 = IRQ0 > IRQ1
1 = IRQ1 > IRQ0

Group B:
0 = IRQ2 > (IRQ3, IRQ4)
1 = (IRQ3, IRQ4) > IRQ2

Subgroup B:
0 = IRQ3 > IRQ4
1 = IRQ4 > IRQ3

Group C:
0 = IRQ5 > (IRQ6, IRQ7)
1 = (IRQ6, IRQ7) > IRQ5

Subgroup C:
0 = IRQ6 > IRQ7
1 = IRQ7 > IRQ6
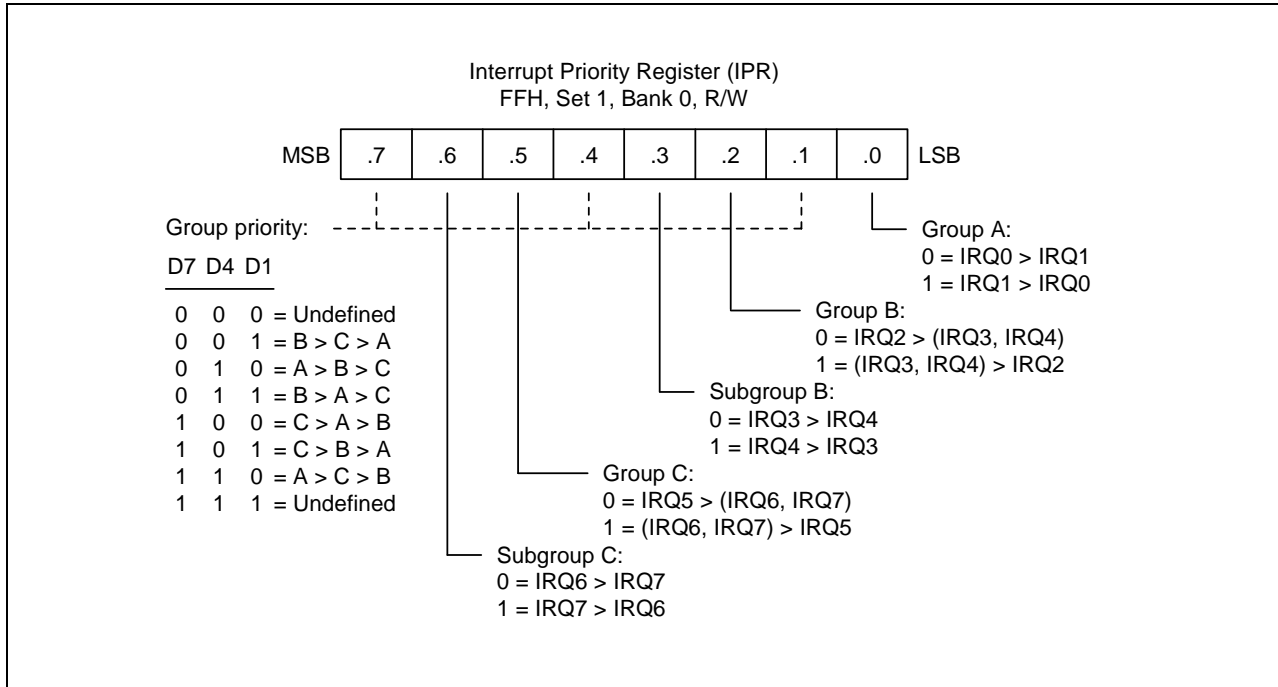
**Figure 5-8. Interrupt Priority Register (IPR)**

SAMSUNG
ELECTRONICS

## INTERRUPT REQUEST REGISTER (IRQ)

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

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

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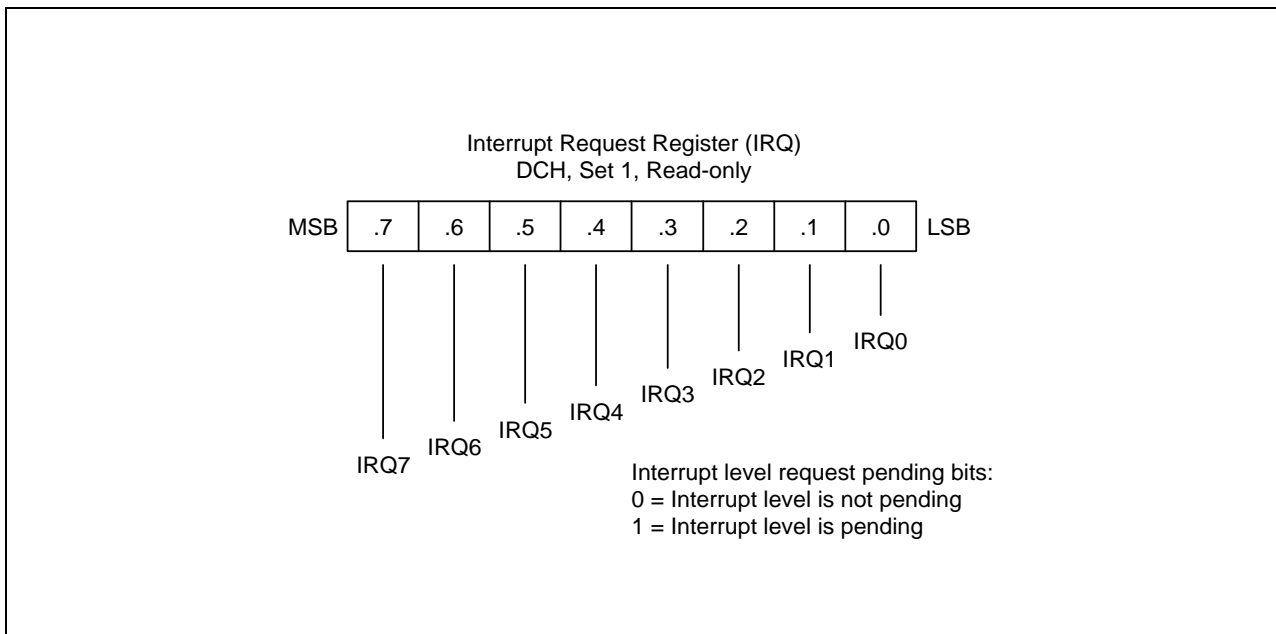


**Figure 5-9. Interrupt Request Register (IRQ)**

## INTERRUPT PENDING FUNCTION TYPES

### Overview

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

### Pending Bits Cleared Automatically by Hardware

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

In the S3C828B/C8289/C8285 interrupt structure, the timer A overflow interrupt (IRQ0) belongs to this category of interrupts in which pending condition is cleared automatically by hardware.

### Pending Bits Cleared by the Service Routine

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

## INTERRUPT SOURCE POLLING SEQUENCE

The interrupt request polling and servicing sequence is as follows:

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

## INTERRUPT SERVICE ROUTINES

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

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

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

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

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

## GENERATING INTERRUPT VECTOR ADDRESSES

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

1.  Push the program counter's low-byte value to the stack.

2.  Push the program counter's high-byte value to the stack.

3.  Push the FLAG register values to the stack.

4.  Fetch the service routine's high-byte address from the vector location.

5.  Fetch the service routine's low-byte address from the vector location.

6.  Branch to the service routine specified by the concatenated 16-bit vector address.

### NOTE

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

## NESTING OF VECTORED INTERRUPTS

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

1.  Push the current 8-bit interrupt mask register (IMR) value to the stack (PUSH IMR).

2.  Load the IMR register with a new mask value that enables only the higher priority interrupt.

3.  Execute an EI instruction to enable interrupt processing (a higher priority interrupt will be processed if it occurs).

4.  When the lower-priority interrupt service routine ends, restore the IMR to its original value by returning the previous mask value from the stack (POP IMR).

5.  Execute an IRET.

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

## INSTRUCTION POINTER (IP)

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

## FAST INTERRUPT PROCESSING

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

## FAST INTERRUPT PROCESSING (Continued)

Two other system registers support fast interrupt processing:

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

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

### NOTE

For the S3C828B/F828B/C8289/F8289/C8285/F8285 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.

**SAMSUNG**
**ELECTRONICS**

# 6 INSTRUCTION SET

## OVERVIEW

The SAM8 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) |

SAMSUNG
ELECTRONICS

**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**

| Mnemonic | Operands | Instruction |
|----------|----------|-------------|
| 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**

| Mnemonic | Operands | Instruction |
|----------|----------|-------------|
| 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.*
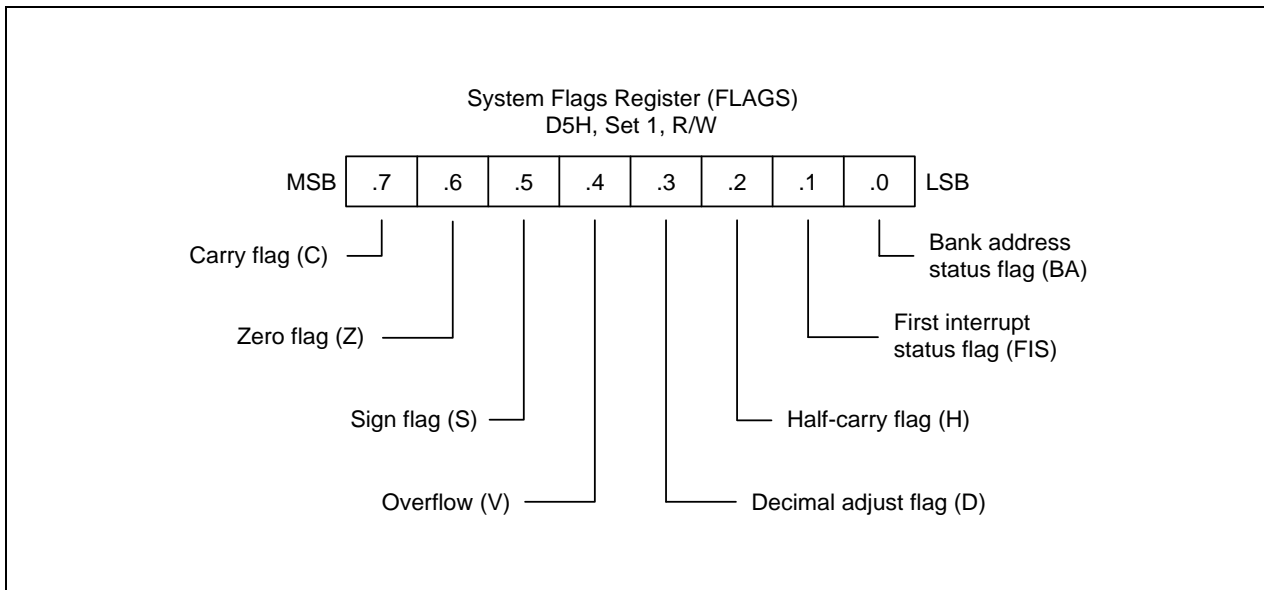


**Figure 6-1. System Flags Register (FLAGS)**

## 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)**

| | | − | 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 |

*OPCODE MAP — LOWER NIBBLE (HEX); left vertical label: UPPER NIBBLE (HEX)*

## 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.

**SAMSUNG**
**ELECTRONICS**

## INSTRUCTION DESCRIPTIONS

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

— Instruction name (mnemonic)

— Full instruction name

— Source/destination format of the instruction operand

— Shorthand notation of the instruction's operation

— Textual description of the instruction's effect

— Specific flag settings affected by the instruction

— Detailed description of the instruction's format, execution time, and addressing mode(s)

— Programming example(s) explaining how to use the instruction

# ADC — Add with carry

**ADC**          dst,src

**Operation:**          dst ← dst + src + c

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

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

**Format:**

| | | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|
| opc | dst \| src | 2 | 4 | 12 | r | r |
| | | | 6 | 13 | r | Ir |
| 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.

**SAMSUNG**
**ELECTRONICS**

# ADD — Add

**ADD**          dst,src

**Operation:**     dst  ←  dst  +  src

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

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

**Format:**

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

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

ADD    R1,R2         →     R1   =   15H, R2   =   03H

ADD    R1,@R2        →     R1   =   1CH, R2   =   03H

ADD    01H,02H       →     Register 01H   =   24H, register 02H   =   03H

ADD    01H,@02H      →     Register 01H   =   2BH, register 02H   =   03H

ADD    01H,#25H      →     Register 01H   =   46H

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

# AND — Logical AND

**AND** dst,src

**Operation:** dst ← dst AND src

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

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

**Format:**

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

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

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

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

**SAMSUNG**
**ELECTRONICS**

# BAND — Bit AND

**BAND**        dst,src.b

**BAND**        dst.b,src

**Operation:**        dst(0)  ←  dst(0)   AND   src(b)

                         or

dst(b)  ←  dst(b)   AND   src(0)

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

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

**Format:**

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

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

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

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

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

# BCP — Bit Compare

**BCP**             dst,src.b

**Operation:**     dst(0) – src(b)

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

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

**Format:**

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

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

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

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

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

SAMSUNG
ELECTRONICS

# 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 <u>dst</u> |
|---|---|:---:|:---:|:---:|:---:|
| opc | dst \| b \| 0 | 2 | 4 | 57 | rb |

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

**Example:**    Given:  R1  =  07H

BITC   R1.1   →    R1  =  05H

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

# BITR — Bit Reset

**BITR**          dst.b

**Operation:**    dst(b)  ←  0

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

**Flags:**        No flags are affected.

**Format:**

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

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

**Example:**    Given:  R1  =  07H:

BITR    R1.1    →      R1   =   05H

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

# BITS — Bit Set

**BITS**          dst.b

**Operation:**     dst(b)  ←  1

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

**Flags:**       No flags are affected.

**Format:**

|  |  | Bytes | Cycles | Opcode (Hex) | Addr Mode <u>dst</u> |
|---|---|:---:|:---:|:---:|:---:|
| opc | dst \| b \| 1 | 2 | 4 | 77 | rb |

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

**Example:**      Given:  R1  =  07H:

BITS    R1.3    →       R1  =  0FH

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

# BOR — Bit OR

**BOR**          dst,src.b

**BOR**          dst.b,src

**Operation:**   dst(0)  ←  dst(0)  OR  src(b)

                     or

                 dst(b)  ←  dst(b)  OR  src(0)

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

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

**Format:**

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

**SAMSUNG**
**ELECTRONICS**

# BTJRF — Bit Test, Jump Relative on False

**BTJRF**          dst,src.b

**Operation:**     If src(b) is a "0", then PC  $\leftarrow$  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) | | | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|---|
| opc | src \| b \| 0 | 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:

BTJRF  SKIP,R1.3                $\rightarrow$        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 | 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.)

**SAMSUNG**
**ELECTRONICS**

# BXOR — Bit XOR

**BXOR**   dst,src.b

**BXOR**   dst.b,src

**Operation:**  dst(0) &larr; dst(0) XOR src(b)

        or

    dst(b) &larr; dst(b) XOR src(0)

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

**Flags:**   **C:** Unaffected.

     **Z:** Set if the result is "0"; cleared otherwise.

     **S:** Cleared to "0".

     **V:** Undefined.

     **D:** Unaffected.

     **H:** Unaffected.

**Format:**

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

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

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

BXOR R1,01H.1   &rarr;   R1 = 06H, register 01H = 03H

BXOR 01H.2,R1   &rarr;   Register 01H = 07H, R1 = 07H

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

# CALL — Call Procedure

**CALL**        dst

**Operation:**   SP   ←   SP – 1
@SP   ←   PCL
SP   ←   SP –1
@SP   ←   PCH
PC   ←   dst

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

**Flags:**       No flags are affected.

**Format:**

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

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

CALL   3521H  →       SP  =  0000H

(Memory locations 0000H  =  1AH, 0001H  =  4AH, where

4AH is the address that follows the instruction.)

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

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

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

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

**SAMSUNG**
**ELECTRONICS**

# CCF — Complement Carry Flag

**CCF**

**Operation:**     C  ←  NOT  C

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

**Flags:**      **C:**  Complemented.

No other flags are affected.

**Format:**

|  |  | Bytes | Cycles | Opcode (Hex) |
|---|---|---|---|---|
| opc |  | 1 | 4 | EF |

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

CCF

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

# CLR — Clear

**CLR**          dst

**Operation:**     dst  ←  "0"

The destination location is cleared to "0".

**Flags:**         No flags are affected.

**Format:**

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

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

CLR    00H    →      Register 00H  =  00H

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

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

SAMSUNG
ELECTRONICS

# COM — Complement

**COM**          dst

**Operation:**      dst  ←  NOT  dst

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

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

**Format:**

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

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

COM    R1      →      R1  =  0F8H

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

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

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

**SAMSUNG**
**ELECTRONICS**

# CP — Compare

**CP**            dst,src

**Operation:**    dst – src

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

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

**Format:**

|  | | | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|---|
| opc | dst \| src | | 2 | 4 | A2 | r | r |
|  | | | | 6 | A3 | r | 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.

SAMSUNG
ELECTRONICS

# CPIJE — Compare, Increment, and Jump on Equal

**CPIJE**            dst,src,RA

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

Ir  ←  Ir  +  1

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

**Flags:**          No flags are affected.

**Format:**

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

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

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

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

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

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

**CPIJNE** dst,src,RA

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

Ir ← Ir + 1

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

**Flags:** No flags are affected.

**Format:**

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

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

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

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

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

**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:**

| | | Bytes | Cycles | Opcode (Hex) | Addr Mode dst |
|---|---|---|---|---|---|
| opc | 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).

**SAMSUNG**
**ELECTRONICS**

# 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.

**SAMSUNG**
**ELECTRONICS**

# DECW — Decrement Word

**DECW**          dst

**Operation:**      dst $\leftarrow$ 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    $\rightarrow$      R0   =   12H, R1   =   33H

DECW   @R2    $\rightarrow$      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

**SAMSUNG**
**ELECTRONICS**

# DI — Disable Interrupts

**DI**

**Operation:**    SYM (0)  ←  0

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

**Flags:**    No flags are affected.

**Format:**

|  | Bytes | Cycles | Opcode (Hex) |
|---|---|---|---|
| opc | 1 | 4 | 8F |

**Example:**    Given:  SYM  =  01H:

DI

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

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

# DIV — Divide (Unsigned)

**DIV**           dst,src

**Operation:**    dst  ÷  src

dst  (UPPER)  ←  REMAINDER

dst  (LOWER)  ←  QUOTIENT

The destination operand (16 bits) is divided by the source operand (8 bits). The quotient (8 bits) is stored in the lower half of the destination. The remainder (8 bits) is stored in the upper half of the destination. When the quotient is $\geq 2^8$, 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).

**SAMSUNG**
**ELECTRONICS**

# DJNZ — Decrement and Jump if Non-Zero

**DJNZ**          r,dst

**Operation:**    $r \leftarrow r - 1$

If $r \neq 0$, $PC \leftarrow PC + dst$

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.

**SAMSUNG**

**ELECTRONICS**

# 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.)

**SAMSUNG**
**ELECTRONICS**

# 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.
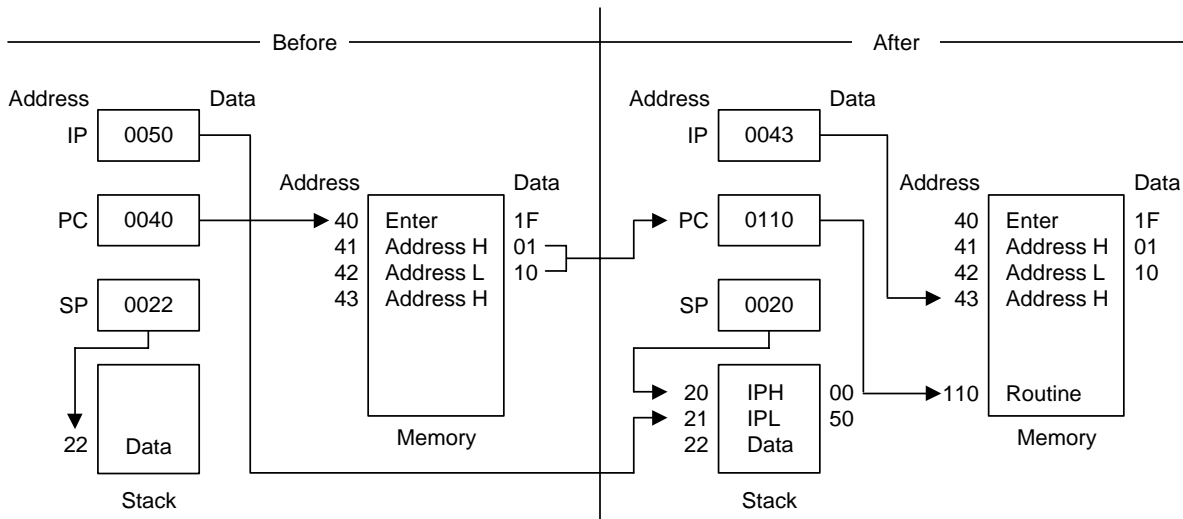
# EXIT — Exit

**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.

SAMSUNG
ELECTRONICS
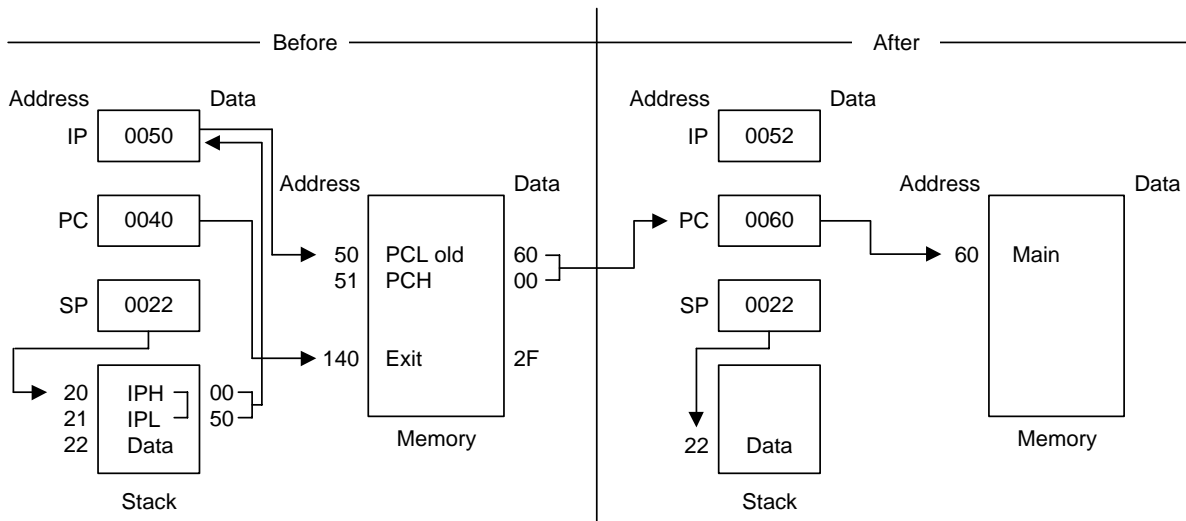
# IDLE — Idle Operation

**IDLE**

**Operation:**

The IDLE instruction stops the CPU clock while allowing system clock oscillation to continue. Idle mode can be released by an interrupt request (IRQ) or an external reset operation.

**Flags:** No flags are affected.

**Format:**

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

**Example:** The instruction

IDLE

stops the CPU clock but not the system clock.

# INC — Increment

| | |
|---|---|
| **INC** | dst |

**Operation:**     dst  ←  dst  +  1

The contents of the destination operand are incremented by one.

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

**Format:**

|  | Bytes | Cycles | Opcode (Hex) | Addr Mode dst |
|---|---|---|---|---|
| dst \| opc | 1 | 4 | rE | r |
|  |  |  | r = 0 to F |  |
| 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.

**SAMSUNG**
**ELECTRONICS**

# 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
```

**SAMSUNG**
**ELECTRONICS**

**6-45**

# IRET — Interrupt Return

**IRET**  <u>IRET (Normal)</u>  <u>IRET (Fast)</u>

**Operation:**
FLAGS $\leftarrow$ @SP  PC $\leftrightarrow$ IP
SP $\leftarrow$ SP + 1  FLAGS $\leftarrow$ FLAGS'
PC $\leftarrow$ @SP  FIS $\leftarrow$ 0
SP $\leftarrow$ SP + 2
SYM(0) $\leftarrow$ 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<br>(Normal) | **Bytes** | **Cycles** | **Opcode (Hex)** |
|---|---|---|---|
| opc | 1 | 10 (internal stack) | BF |
|  |  | 12 (internal stack) |  |

| IRET<br>(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.

```
0H
FFH      IRET
100H     Interrupt
         Service
         Routine
         JP to FFH
FFFFH
```
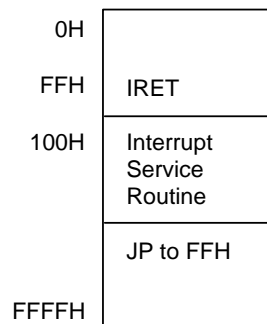
**NOTE:** In the fast interrupt example above, if the last instruction is not a jump to IRET, you must pay attention to the order of the last two instructions. The IRET cannot be immediately proceeded by a clearing of the interrupt status (as with a reset of the IPR register).

**SAMSUNG**
**ELECTRONICS**

# 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]

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

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:**

|  |  |  |  | **Bytes** | **Cycles** | **Opcode (Hex)** | **Addr Mode dst** |
|---|---|---|---|---|---|---|---|
| (1) |  |  |  |  |  |  |  |
| 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.

SAMSUNG
ELECTRONICS

# LD — Load

**LD**            dst,src

**Operation:**    dst  ←  src

The contents of the source are loaded into the destination. The source's contents are unaffected.

**Flags:**        No flags are affected.

**Format:**

| | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|
| dst \| opc    src | 2 | 4 | rC | r | IM |
| | | 4 | r8 | r | R |
| src \| opc    dst | 2 | 4 | r9 | R | r |
| r = 0 to F | | | | | |
| opc    dst \| src | 2 | 4 | C7 | r | lr |
| | | 4 | D7 | lr | r |
| opc    src    dst | 3 | 6 | E4 | R | R |
| | | 6 | E5 | R | IR |
| opc    dst    src | 3 | 6 | E6 | R | IM |
| | | 6 | D6 | IR | IM |
| opc    src    dst | 3 | 6 | F5 | IR | R |
| opc    dst \| src    x | 3 | 6 | 87 | r | x [r] |
| opc    src \| dst    x | 3 | 6 | 97 | x [r] | r |

# LD — Load

**LD**              (Continued)

**Examples:**    Given:  R0   =   01H, R1   =   0AH, register 00H   =   01H, register 01H   =   20H,
register 02H   =   02H, LOOP   =   30H, and register 3AH   =   0FFH:

| | | | |
|---|---|---|---|
| LD | R0,#10H | → | R0 = 10H |
| LD | R0,01H | → | R0 = 20H, register 01H = 20H |
| LD | 01H,R0 | → | Register 01H = 01H, R0 = 01H |
| LD | R1,@R0 | → | R1 = 20H, R0 = 01H |
| LD | @R0,R1 | → | R0 = 01H, R1 = 0AH, register 01H = 0AH |
| LD | 00H,01H | → | Register 00H = 20H, register 01H = 20H |
| LD | 02H,@00H | → | Register 02H = 20H, register 00H = 01H |
| LD | 00H,#0AH | → | Register 00H = 0AH |
| LD | @00H,#10H | → | Register 00H = 01H, register 01H = 10H |
| LD | @00H,02H | → | Register 00H = 01H, register 01H = 02, register 02H = 02H |
| LD | R0,#LOOP[R1] | → | R0 = 0FFH, R1 = 0AH |
| LD | #LOOP[R0],R1 | → | Register 31H = 0AH, R0 = 01H, R1 = 0AH |

SAMSUNG
ELECTRONICS

# LDB — Load Bit

| **LDB** | dst,src.b |
|---|---|
| **LDB** | dst.b,src |

**Operation:**     dst(0)  ←  src(b)

                              or

dst(b)  ←  src(0)

The specified bit of the source is loaded into bit zero (LSB) of the destination, or bit zero of the source is loaded into the specified bit of the destination. No other bits of the destination are affected. The source is unaffected.

**Flags:**         No flags are affected.

**Format:**

|  |  | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|
| opc \| dst \| b \| 0 \| src | | 3 | 6 | 47 | r0 | Rb |
| opc \| src \| b \| 1 \| dst | | 3 | 6 | 47 | Rb | r0 |

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

**Examples:**    Given:  R0   =   06H and general register 00H   =   05H:

LDB        R0,00H.2    →        R0   =   07H, register 00H   =   05H

LDB        00H.0,R0    →        R0   =   06H, register 00H   =   04H

In the first example, destination working register R0 contains the value 06H and the source general register 00H the value 05H. The statement "LD   R0,00H.2" loads the bit two value of the 00H register into bit zero of the R0 register, leaving the value 07H in register R0.

In the second example, 00H is the destination register. The statement "LD   00H.0,R0" loads bit zero of register R0 to the specified bit (bit zero) of the destination register, leaving 04H in general register 00H.

# LDC/LDE — **Load Memory**

**LDC/LDE**      dst,src

**Operation:**      dst  ←  src

This instruction loads a byte from program or data memory into a working register or vice-versa. The source values are unaffected. LDC refers to program memory and LDE to data memory. The assembler makes 'Irr' or 'rr' values an even number for program memory and odd an odd number for data memory.

**Flags:**      No flags are affected.

**Format:**

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

SAMSUNG
ELECTRONICS

# LDC/LDE — Load Memory

**LDC/LDE**      (Continued)

**Examples:**   Given: R0 = 11H, R1 = 34H, R2 = 01H, R3 = 04H; Program memory locations
0103H = 4FH, 0104H = 1A, 0105H = 6DH, and 1104H = 88H. External data memory
locations 0103H = 5FH, 0104H = 2AH, 0105H = 7DH, and 1104H = 98H:

| | | | |
|---|---|---|---|
| LDC | R0,@RR2 | ; | R0 ← contents of program memory location 0104H |
| | | ; | R0 = 1AH, R2 = 01H, R3 = 04H |
| | | | |
| LDE | R0,@RR2 | ; | R0 ← contents of external data memory location 0104H |
| | | ; | R0 = 2AH, R2 = 01H, R3 = 04H |
| | | | |
| LDC (note) | @RR2,R0 | ; | 11H (contents of R0) is loaded into program memory |
| | | ; | location 0104H (RR2), |
| | | ; | working registers R0, R2, R3 → no change |
| | | | |
| LDE | @RR2,R0 | ; | 11H (contents of R0) is loaded into external data memory |
| | | ; | location 0104H (RR2), |
| | | ; | working registers R0, R2, R3 → no change |
| | | | |
| LDC | R0,#01H[RR2] | ; | R0 ← contents of program memory location 0105H |
| | | ; | (01H + RR2), |
| | | ; | R0 = 6DH, R2 = 01H, R3 = 04H |
| | | | |
| LDE | R0,#01H[RR2] | ; | R0 ← contents of external data memory location 0105H |
| | | ; | (01H + RR2), R0 = 7DH, R2 = 01H, R3 = 04H |
| | | | |
| LDC (note) | #01H[RR2],R0 | ; | 11H (contents of R0) is loaded into program memory location |
| | | ; | 0105H (01H + 0104H) |
| | | | |
| LDE | #01H[RR2],R0 | ; | 11H (contents of R0) is loaded into external data memory |
| | | ; | location 0105H (01H + 0104H) |
| | | | |
| LDC | R0,#1000H[RR2] | ; | R0 ← contents of program memory location 1104H |
| | | ; | (1000H + 0104H), R0 = 88H, R2 = 01H, R3 = 04H |
| | | | |
| LDE | R0,#1000H[RR2] | ; | R0 ← contents of external data memory location 1104H |
| | | ; | (1000H + 0104H), R0 = 98H, R2 = 01H, R3 = 04H |
| | | | |
| LDC | R0,1104H | ; | R0 ← contents of program memory location 1104H, R0 = 88H |
| | | | |
| LDE | R0,1104H | ; | R0 ← contents of external data memory location 1104H, |
| | | ; | R0 = 98H |
| | | | |
| LDC (note) | 1105H,R0 | ; | 11H (contents of R0) is loaded into program memory location |
| | | ; | 1105H, (1105H) ← 11H |
| | | | |
| LDE | 1105H,R0 | ; | 11H (contents of R0) is loaded into external data memory |
| | | ; | location 1105H, (1105H) ← 11H |

**NOTE:**  These instructions are not supported by masked ROM type devices.

# LDCD/LDED — **Load Memory and Decrement**

**LDCD/LDED**   dst,src

**Operation:**   dst ← src

rr ← rr − 1

These instructions are used for user stacks or block transfers of data from program or data memory to the register file. The address of the memory location is specified by a working register pair. The contents of the source location are loaded into the destination location. The memory address is then decremented. The contents of the source are unaffected.

LDCD references program memory and LDED references external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for data memory.

**Flags:**   No flags are affected.

**Format:**

|          |           | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|----------|-----------|-------|--------|--------------|---------------|-----|
| opc | dst \| src | 2 | 10 | E2 | r | Irr |

**Examples:**   Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory location 1033H = 0CDH, and external data memory location 1033H = 0DDH:

LDCD   R8,@RR6   ;  0CDH (contents of program memory location 1033H) is loaded

;  into R8 and RR6 is decremented by one

;  R8 = 0CDH, R6 = 10H, R7 = 32H (RR6 ← RR6 − 1)

LDED   R8,@RR6   ;  0DDH (contents of data memory location 1033H) is loaded

;  into R8 and RR6 is decremented by one (RR6 ← RR6 − 1)

;  R8 = 0DDH, R6 = 10H, R7 = 32H

**SAMSUNG**
**ELECTRONICS**

# LDCI/LDEI — **Load Memory and Increment**

**LDCI/LDEI**　　　dst,src

**Operation:**　　　dst　←　src

rr　←　rr　+　1

These instructions are used for user stacks or block transfers of data from program or data memory to the register file. The address of the memory location is specified by a working register pair. The contents of the source location are loaded into the destination location. The memory address is then incremented automatically. The contents of the source are unaffected.

LDCI refers to program memory and LDEI refers to external data memory. The assembler makes 'Irr' even for program memory and odd for data memory.

**Flags:**　　　No flags are affected.

**Format:**

|  |  | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|
| opc | dst \| src | 2 | 10 | E3 | r | Irr |

**Examples:**　　　Given:  R6 = 10H, R7 = 33H, R8 = 12H, program memory locations 1033H = 0CDH and 1034H = 0C5H; external data memory locations 1033H = 0DDH and 1034H = 0D5H:

LDCI　　　R8,@RR6　　　;　0CDH (contents of program memory location 1033H) is loaded

　　　　　　　　　　　　　;　into R8 and RR6 is incremented by one (RR6　←　RR6 + 1)

　　　　　　　　　　　　　;　R8　=　0CDH, R6　=　10H, R7　=　34H

LDEI　　　R8,@RR6　　　;　0DDH (contents of data memory location 1033H) is loaded

　　　　　　　　　　　　　;　into R8 and RR6 is incremented by one (RR6　←　RR6 + 1)

　　　　　　　　　　　　　;　R8　=　0DDH, R6　=　10H, R7　=　34H

# LDCPD/LDEPD — Load Memory with Pre-Decrement

**LDCPD/**
**LDEPD**        dst,src

**Operation:**    rr  ←  rr  −  1

dst  ←  src

These instructions are used for block transfers of data from program or data memory from the register file. The address of the memory location is specified by a working register pair and is first decremented. The contents of the source location are then loaded into the destination location. The contents of the source are unaffected.

LDCPD refers to program memory and LDEPD refers to external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for external data memory.

**Flags:**       No flags are affected.

**Format:**

|  |  | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|
| opc | src \| dst | 2 | 14 | F2 | Irr | r |

**Examples:**    Given:  R0  =  77H, R6  =  30H, and R7  =  00H:

LDCPD    @RR6,R0        ;  (RR6  ←  RR6 – 1)
                        ;  77H (contents of R0) is loaded into program memory location
                        ;  2FFFH (3000H – 1H)
                        ;  R0  =  77H, R6  =  2FH, R7  =  0FFH

LDEPD    @RR6,R0        ;  (RR6  ←  RR6 – 1)
                        ;  77H (contents of R0) is loaded into external data memory
                        ;  location 2FFFH (3000H – 1H)
                        ;  R0  =  77H, R6  =  2FH, R7  =  0FFH

**SAMSUNG**
**ELECTRONICS**

# LDCPI/LDEPI — Load Memory with Pre-Increment

**LDCPI/**
**LDEPI**        dst,src

**Operation:**    rr  ←  rr  +  1

dst  ←  src

These instructions are used for block transfers of data from program or data memory from the register file. The address of the memory location is specified by a working register pair and is first incremented. The contents of the source location are loaded into the destination location. The contents of the source are unaffected.

LDCPI refers to program memory and LDEPI refers to external data memory. The assembler makes 'Irr' an even number for program memory and an odd number for data memory.

**Flags:**        No flags are affected.

**Format:**

| | | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|
| opc | src \| dst | 2 | 14 | F3 | Irr | r |

**Examples:**   Given:  R0  =  7FH, R6  =  21H, and R7  =  0FFH:

LDCPI    @RR6,R0         ;  (RR6  ←  RR6 + 1)
                        ;  7FH (contents of R0) is loaded into program memory
                        ;  location 2200H (21FFH + 1H)
                        ;  R0  =  7FH, R6  =  22H, R7  =  00H

LDEPI    @RR6,R0         ;  (RR6  ←  RR6 + 1)
                        ;  7FH (contents of R0) is loaded into external data memory
                        ;  location 2200H (21FFH + 1H)
                        ;  R0  =  7FH, R6  =  22H, R7  =  00H

# LDW — Load Word

**LDW**        dst,src

**Operation:**        dst ← src

The contents of the source (a word) are loaded into the destination. The contents of the source are unaffected.

**Flags:**        No flags are affected.

**Format:**

| | | | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | 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:**

| | | | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | 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
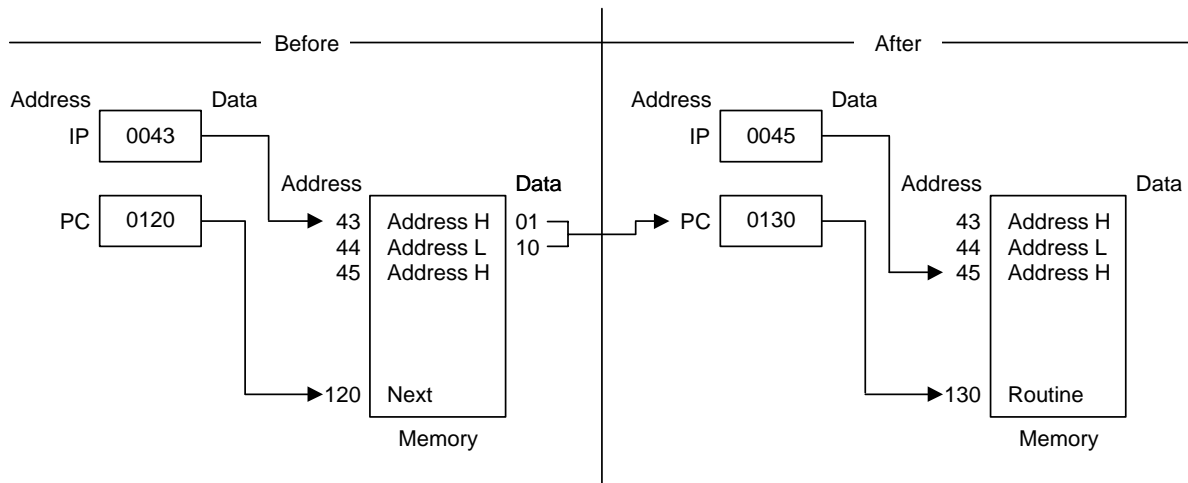
# NOP — No Operation

**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:**

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

**SAMSUNG**
**ELECTRONICS**

# 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 | 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.

**SAMSUNG**
**ELECTRONICS**

# 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 | 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 <u>dst</u> |
|---|---|---|---|---|---|
| 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.

**SAMSUNG**

**ELECTRONICS**

# PUSHUD — Push User Stack (Decrementing)

**PUSHUD**        dst,src

**Operation:**    IR  ←  IR  − 1

dst  ←  src

This instruction is used to address user-defined stacks in the register file. PUSHUD decrements the user stack pointer and loads the contents of the source into the register addressed by the decremented stack pointer.

**Flags:**        No flags are affected.

**Format:**

|  |  |  | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|---|
| opc | dst | src | 3 | 8 | 82 | IR | R |

**Example:**      Given:  Register 00H  =  03H, register 01H  =  05H, and register 02H  =  1AH:

PUSHUD  @00H,01H  →        Register 00H = 02H, register 01H = 05H, register 02H = 05H

If the user stack pointer (register 00H, for example) contains the value 03H, the statement "PUSHUD @00H,01H" decrements the user stack pointer by one, leaving the value 02H. The 01H register value, 05H, is then loaded into the register addressed by the decremented user stack pointer.

# PUSHUI — Push User Stack (Incrementing)

**PUSHUI**        dst,src

**Operation:**     IR  ←  IR  +  1

dst  ←  src

This instruction is used for user-defined stacks in the register file. PUSHUI increments the user stack pointer and then loads the contents of the source into the register location addressed by the incremented user stack pointer.

**Flags:**        No flags are affected.

**Format:**

|  |  |  | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|---|
| opc | dst | src | 3 | 8 | 83 | IR | R |

**Example:**      Given:  Register 00H  =  03H, register 01H  =  05H, and register 04H  =  2AH:

PUSHUI   @00H,01H   →        Register 00H = 04H, register 01H = 05H, register 04H = 05H

If the user stack pointer (register 00H, for example) contains the value 03H, the statement "PUSHUI @00H,01H" increments the user stack pointer by one, leaving the value 04H. The 01H register value, 05H, is then loaded into the location addressed by the incremented user stack pointer.

SAMSUNG
ELECTRONICS

# 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.

**SAMSUNG**
**ELECTRONICS**

# 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.

```
          ┌──────────7──────────────0─────────┐
          │                                   │
   C ◄────●──────┌──────────────────────┐◄────┘
   └───┘         └──────────────────────┘
```

**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.
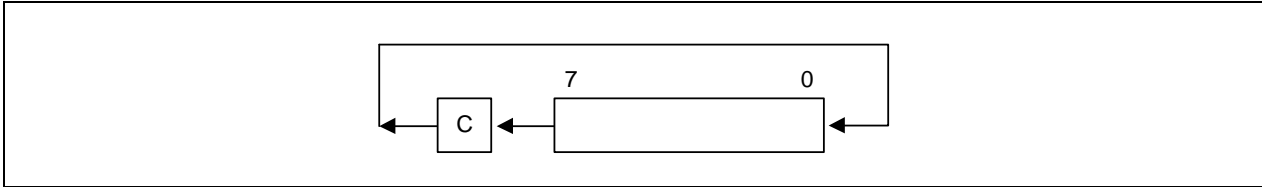


**Flags:**       **C:** Set if the bit rotated from the most significant bit position (bit 7) was "1".
**Z:** Set if the result is "0"; cleared otherwise.
**S:** Set if the result bit 7 is set; cleared otherwise.
**V:** Set if arithmetic overflow occurred, that is, if the sign of the destination changed 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.

SAMSUNG
ELECTRONICS
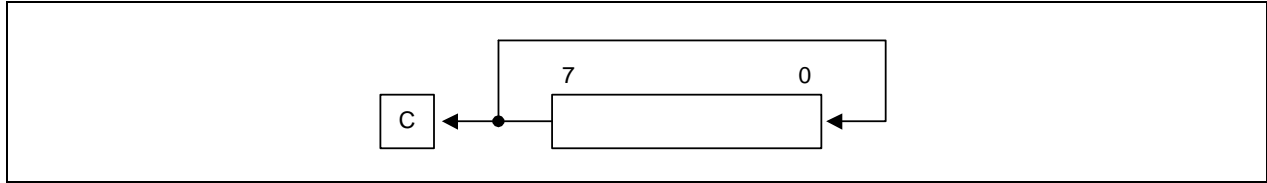
# RR — Rotate Right

**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).
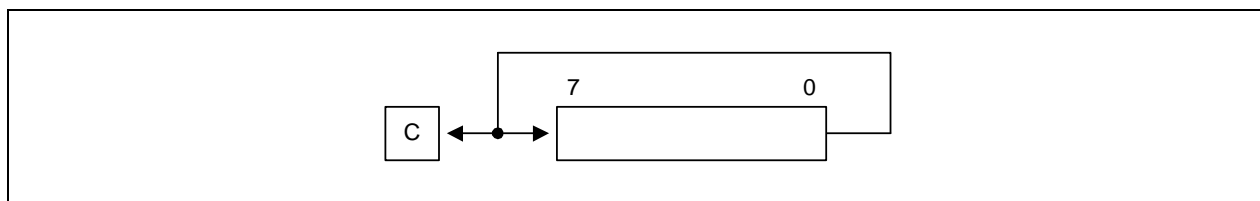


**Flags:**    **C:**  Set if the bit rotated from the least significant bit position (bit zero) was "1".
**Z:**  Set if the result is "0"; cleared otherwise.
**S:**  Set if the result bit 7 is set; cleared otherwise.
**V:**  Set if arithmetic overflow occurred, that is, if the sign of the destination 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".

**SAMSUNG**

**ELECTRONICS**

# RRC — Rotate Right Through Carry

**RRC**            dst

**Operation:**     dst (7)  ←  C

C  ←  dst (0)

dst (n)  ←  dst (n  +  1), n  =  0–6

The contents of the destination operand and the carry flag are rotated right one bit position. The initial value of bit zero (LSB) replaces the carry flag; the initial value of the carry flag replaces bit 7 (MSB).
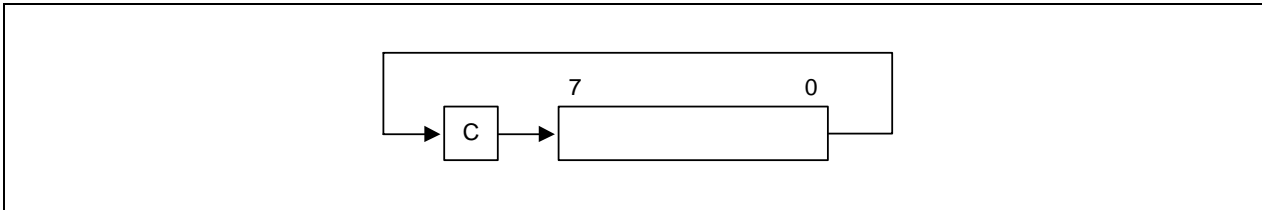


**Flags:**     **C:** Set if the bit rotated from the least significant bit position (bit zero) was "1".
**Z:** Set if the result is "0" cleared otherwise.
**S:** Set if the result bit 7 is set; cleared otherwise.
**V:** Set if arithmetic overflow occurred, that is, if the sign of the destination changed during rotation; cleared otherwise.
**D:** Unaffected.
**H:** Unaffected.

**Format:**

|  |  | Bytes | Cycles | Opcode (Hex) | Addr Mode dst |
|---|---|---|---|---|---|
| opc | dst | 2 | 4 | C0 | R |
|  |  |  | 4 | C1 | IR |

**Examples:**     Given:  Register 00H  =  55H, register 01H  =  02H, register 02H  =  17H, and C  =  "0":

RRC       00H          →       Register 00H  =  2AH, C  =  "1"

RRC       @01H         →       Register 01H  =  02H, register 02H  =  0BH, C  =  "1"

In the first example, if general register 00H contains the value 55H (01010101B), the statement "RRC   00H" rotates this value one bit position to the right. The initial value of bit zero ("1") replaces the carry flag and the initial value of the C flag ("1") replaces bit 7. This   leaves the new value 2AH (00101010B) in destination register 00H. The sign flag and overflow flag are both cleared to "0".

# SB0 — Select Bank 0

**SB0**

**Operation:**    BANK $\leftarrow$ 0

The SB0 instruction clears the bank address flag in the FLAGS register (FLAGS.0) to logic zero, selecting bank 0 register addressing in the set 1 area of the register file.

**Flags:**    No flags are affected.

**Format:**

|  | Bytes | Cycles | Opcode (Hex) |
|---|---|---|---|
| opc | 1 | 4 | 4F |

**Example:**    The statement

SB0

clears FLAGS.0 to "0", selecting bank 0 register addressing.

# SB1 — Select Bank 1

**SB1**

**Operation:**        BANK  ← 1

The SB1 instruction sets the bank address flag in the FLAGS register (FLAGS.0) to logic one, selecting bank 1 register addressing in the set 1 area of the register file. (Bank 1 is not implemented in some S3C8-series microcontrollers.)

**Flags:**        No flags are affected.

**Format:**

|  | Bytes | Cycles | Opcode (Hex) |
|---|---|---|---|
| opc | 1 | 4 | 5F |

**Example:**        The statement

SB1

sets FLAGS.0 to "1", selecting bank 1 register addressing, if implemented.

# SBC — Subtract with Carry

**SBC**          dst,src

**Operation:**      dst ← dst − src − c

The source operand, along with the current value of the carry flag, is subtracted from the destination operand and the result is stored in the destination. The contents of the source are unaffected. Subtraction is performed by adding the two's-complement of the source operand to the destination operand. In multiple precision arithmetic, this instruction permits the carry ("borrow") from the subtraction of the low-order operands to be subtracted from the subtraction of high-order operands.

**Flags:**       **C:** Set if a borrow occurred (src > dst); cleared otherwise.
             **Z:** Set if the result is "0"; cleared otherwise.
             **S:** Set if the result is negative; cleared otherwise.
             **V:** Set if arithmetic overflow occurred, that is, if the operands were of opposite sign and the sign of the result is the same as the sign of the source; cleared otherwise.
             **D:** Always set to "1".
             **H:** Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise, indicating a "borrow".

**Format:**

|                            | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|----------------------------|-------|--------|--------------|---------------|-----|
| opc │ dst │ src            | 2     | 4      | 32           | r             | r   |
|                            |       | 6      | 33           | r             | Ir  |
| 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 = 5H; 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.

**SAMSUNG**
**ELECTRONICS**

# 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.

SAMSUNG
ELECTRONICS

# 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.
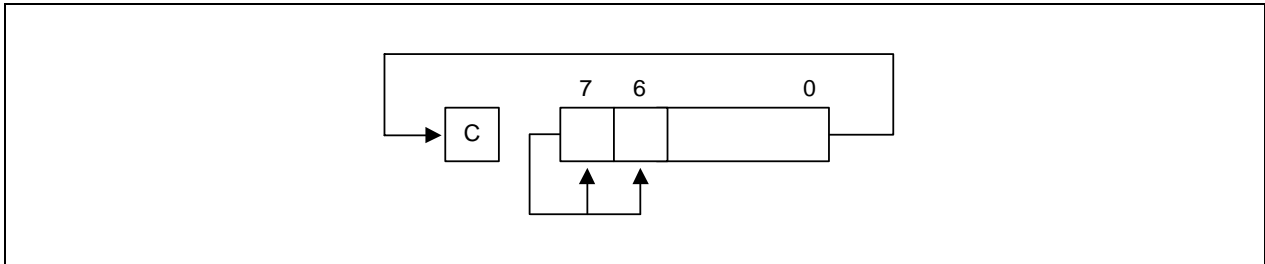


**Flags:**      **C:** Set if the bit shifted from the LSB position (bit zero) was "1".
**Z:** Set if the result is "0"; cleared otherwise.
**S:** Set if the result is negative; cleared otherwise.
**V:** Always cleared to "0".
**D:** Unaffected.
**H:** Unaffected.

**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 <u>src</u> |
|---------|---------|-------|--------|--------------|----------------------|
| 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.

**Flags:**       No flags are affected.

**Format:**

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

**Example:**       The statement

STOP

halts all microcontroller operations.

# SUB — Subtract

**SUB**          dst,src

**Operation:**      dst  ←  dst  − src

The source operand is subtracted from the destination operand and the result is stored in the destination. The contents of the source are unaffected. Subtraction is performed by adding the two's complement of the source operand to the destination operand.

**Flags:**       **C:** Set if a "borrow" occurred; cleared otherwise.
**Z:** Set if the result is "0"; cleared otherwise.
**S:** Set if the result is negative; cleared otherwise.
**V:** Set if arithmetic overflow occurred, that is, if the operands were of opposite signs and the sign of the result is of the same as the sign of the source operand; cleared otherwise.
**D:** Always set to "1".
**H:** Cleared if there is a carry from the most significant bit of the low-order four bits of the result; set otherwise indicating a "borrow".

**Format:**

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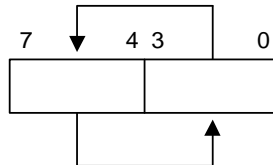


**Flags:**     **C:** Undefined.
**Z:** Set if the result is "0"; cleared otherwise.
**S:** Set if the result bit 7 is set; cleared otherwise.
**V:** Undefined.
**D:** Unaffected.
**H:** Unaffected.

**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:**

|  |  | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|
| opc | dst \| src | 2 | 4 | 62 | r | r |
|  |  |  | 6 | 63 | r | lr |
| opc | src | dst | 3 | 6 | 64 | R | R |
|  |  |  | 6 | 65 | R | IR |
| opc | dst | src | 3 | 6 | 66 | R | IM |

**Examples:**        Given: R0 = 0C7H, R1 = 02H, R2 = 12H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

TCM        R0,R1        →        R0 = 0C7H, R1 = 02H, Z = "1"

TCM        R0,@R1        →        R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = "0"

TCM        00H,01H        →        Register 00H = 2BH, register 01H = 02H, Z = "1"

TCM        00H,@01H        →        Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = "1"

TCM        00H,#34        →        Register 00H = 2BH, Z = "0"

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TCM   R0,R1" tests bit one in the destination register for a "1" value. Because the mask value corresponds to the test bit, the Z flag is set to logic one and can be tested to determine the result of the TCM operation.

**SAMSUNG**
**ELECTRONICS**

# TM — Test Under Mask

**TM**            dst,src

**Operation:**    dst   AND   src

This instruction tests selected bits in the destination operand for a logic zero value. The bits to be tested are specified by setting a "1" bit in the corresponding position of the source operand (mask), which is ANDed with the destination operand. The zero (Z) flag can then be checked to determine the result. The destination and source operands are unaffected.

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

**Format:**

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

**SAMSUNG**
**ELECTRONICS**

                                                                                                  **6-85**

# WFI — Wait for Interrupt
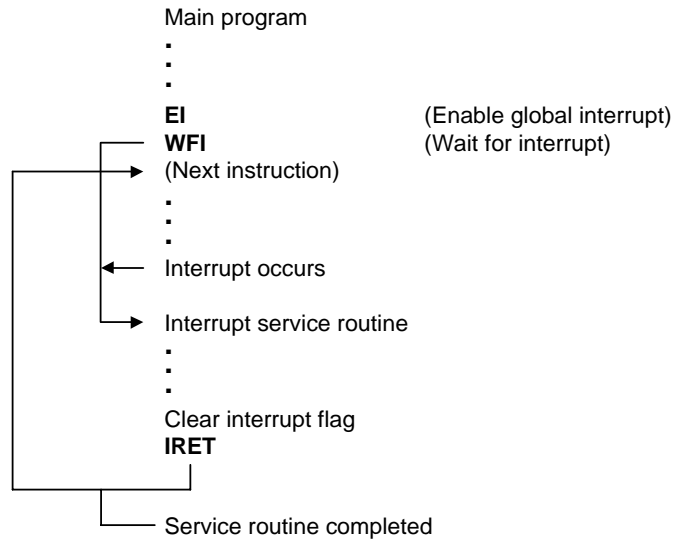
**WFI**

**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
```

SAMSUNG
ELECTRONICS

# XOR — Logical Exclusive OR

**XOR**          dst,src

**Operation:**   dst ← dst XOR src

The source operand is logically exclusive-ORed with the destination operand and the result is stored in the destination. The exclusive-OR operation results in a "1" bit being stored whenever the corresponding bits in the operands are different; otherwise, a "0" bit is stored.

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

**Format:**

|  |  | Bytes | Cycles | Opcode (Hex) | Addr Mode dst | src |
|---|---|---|---|---|---|---|
| opc | dst \| src | 2 | 4 | B2 | r | r |
|  |  |  | 6 | B3 | r | 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 clock frequency generated for the S3C828B/F828B/C8289/F8289/C8285/F8285 by an external crystal can range from 0.4 MHz to 11.1 MHz. The maximum CPU clock frequency is 11.1 MHz. The $X_{IN}$ and $X_{OUT}$ pins connect the external oscillator or clock source to the on-chip clock circuit.

### SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

— External crystal or ceramic resonator oscillation source (or an external clock source)
— Oscillator stop and wake-up functions
— Programmable frequency divider for the CPU clock (fxx divided by 1, 2, 8, or 16)
— System clock control register, CLKCON
— Oscillator control register, OSCCON and STOP control register, STPCON

### CPU CLOCK NOTATION

In this document, the following notation is used for descriptions of the CPU clock;

fx: main clock

fxT: sub clock
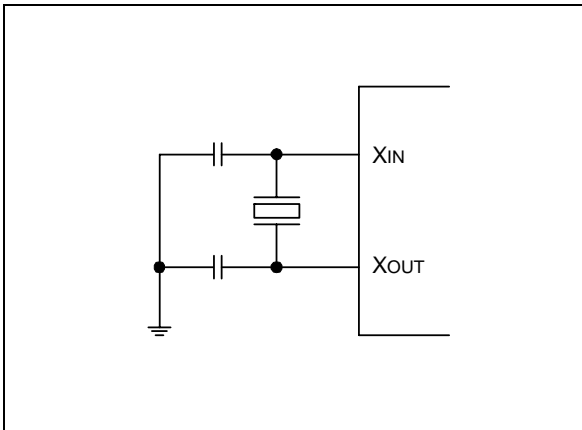
fxx: selected system clock

## MAIN OSCILLATOR CIRCUITS



**Figure 7-1. Crystal/Ceramic Oscillator (f$_X$)**



**Figure 7-2. External Oscillator (f$_X$)**



**Figure 7-3. RC Oscillator (f$_X$)**

## SUB OSCILLATOR CIRCUITS



**Figure 7-4. Crystal Oscillator (f$_{XT}$, Normal)**



**Figure 7-5. Crystal Oscillator (f$_{XT}$, for Low Current)**
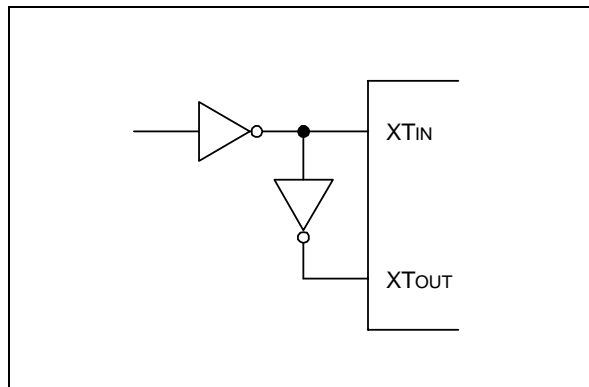


**Figure 7-6. External Oscillator (f$_{XT}$)**

SAMSUNG
ELECTRONICS

## CLOCK STATUS DURING POWER-DOWN MODES

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

— In Stop mode, the main oscillator is halted. Stop mode is released, and the oscillator is started, by a reset operation or an external interrupt (with RC delay noise filter), and can be released by internal interrupt too when the sub-system oscillator is running and watch timer is operating with sub-system clock.

— 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.
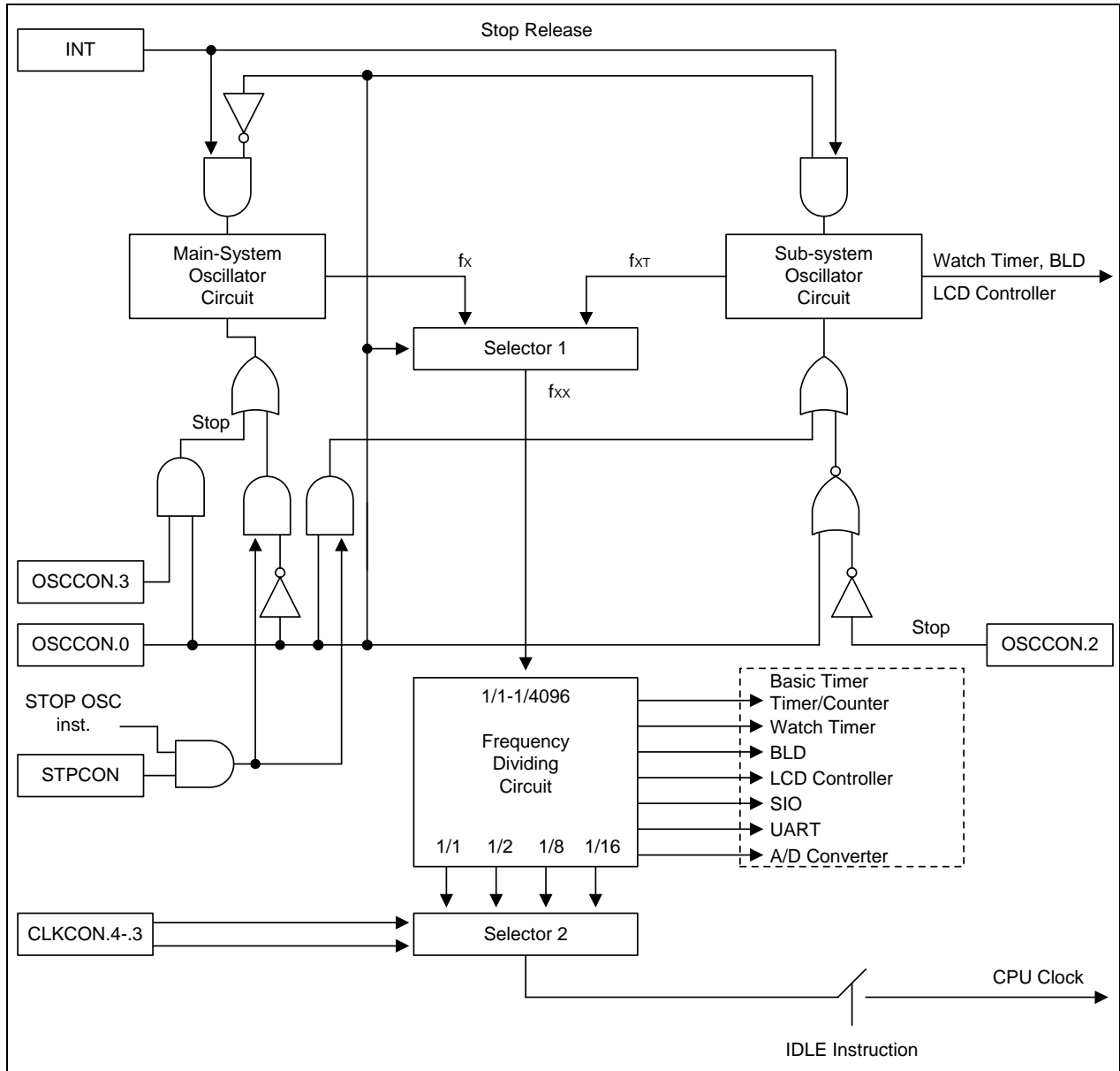


**Figure 7-7. System Clock Circuit Diagram**

**SYSTEM CLOCK CONTROL REGISTER (CLKCON)**

The system clock control register, CLKCON, is located in the set 1, address D4H. It is read/write addressable and has the following functions:

— Oscillator frequency divide-by value

After the main oscillator is activated, and the fxx/16 (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed fxx/8, fxx/2, or fxx/1.



System Clock Control Register (CLKCON)
D4H, Set 1, R/W

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

Not used
(must keep always 0)

Not used
(must keep always 0)

Oscillator IRQ wake-up function bit:
0 = Enable IRQ for main wake-up in
     power down mode
1 = Diable IRQ for main wake-up
     in power down mode

Divide-by selection bits for
CPU clock frequency:
00 = fxx/16
01 = fxx/8
10 = fxx/2
11 = fxx/1 (non-divided)

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

**Figure 7-8. System Clock Control Register (CLKCON)**

Oscillator Control Register (OSCCON)
FAH, Set 1, bank 0, R/W

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

Not used for
S3C828B/C8289/C8285

System clock selection bit:
0 = Main oscillator select
1 = Subsystem oscillator select

Subsystem oscillator circuit selection bit: (1)
0 = Normal circuit for sub oscillator
1 = Power saving circuit for sub oscillator

Not used for S3C828B/C8289/C8285

Subsystem oscillator control bit:
0 = Subsystem oscillator RUN
1 = Subsystem oscillator STOP

Mainsystem oscillator control bit:
0 = Mainsystem oscillator RUN
1 = Mainsystem oscillator STOP

**NOTES:**
1.   A capacitor (0.1uF) should be connected between VREG and GND when the
     sub-oscillator is used to power saving mode (OSCCON.7 = "1")
2.   The OSCCON.7 automatically cleared to "0" when the suboscillator is stopped by
     OSCCON.2 or the CPU is entered into stop mode in sub operating mode.

**Figure 7-9. Oscillator Control Register (OSCCON)**

STOP Control Register (STPCON)
FBH, Set 1,bank 0, R/W

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

STOP Control bits:
Other values = Disable STOP instruction
10100101 = Enable STOP instruction

**NOTE:**   Before executing the STOP instruction, set the STPCON
register as "10100101b". Otherwise the STOP instruction
will not be executed and reset will be generated.
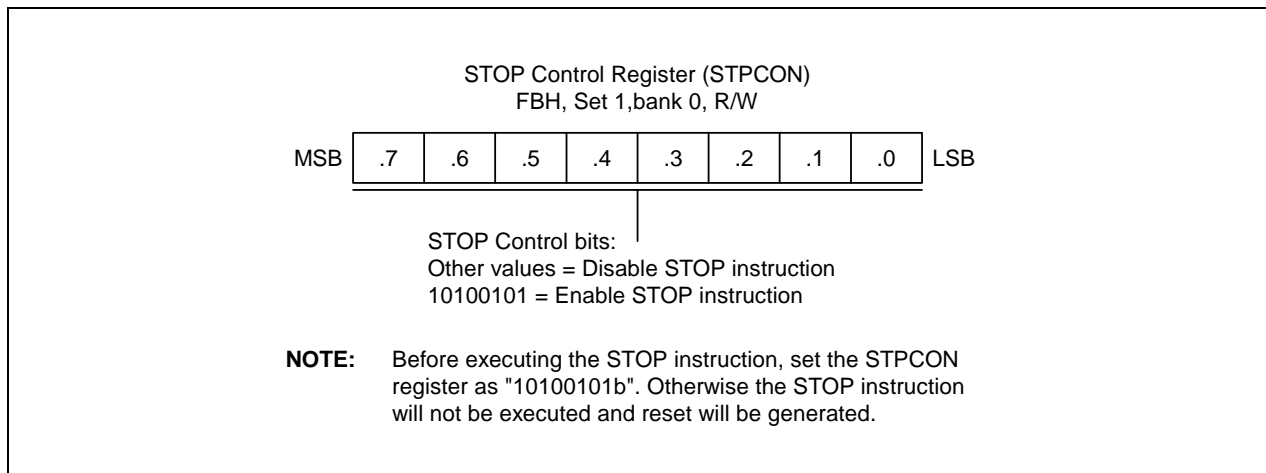
**Figure 7-10. STOP Control Register (STPCON)**

**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
```

# 8 RESET and POWER-DOWN

## SYSTEM RESET

### OVERVIEW

During a power-on reset, the voltage at $V_{DD}$ goes to High level and the RESET pin is forced to Low level. The RESET signal is input through a schmitt trigger circuit where it is then synchronized with the CPU clock. This procedure brings the S3C828B/F828B/C8289/F8289/C8285/F8285 into a known operating status.

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

Whenever a reset occurs during normal operation (that is, when both $V_{DD}$ and RESET are High level), the 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 0-8 and 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 at ROM can be changed by Smart Option in the S3F828B (full-flash device). Refer to "The Chapter 19. Embedded Flash Memory Interface" for more detailed contents.

### NORMAL MODE RESET OPERATION

In normal (masked ROM) mode, the Test pin is tied to $V_{SS}$. A reset enables access to the S3C828B(64Kbyte), S3C8289(32-Kbyte), and S3C8285(16-Kbyte) on-chip ROM. (The external interface is not automatically configured).

<div align="center">

**NOTE**

</div>

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. S3C828B/F828B/C8289/F8289/C8285/F8285 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 |

**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".

SAMSUNG
ELECTRONICS

**Table 8-2. S3C828B/F828B/C8289/F8289/C8285/F8285 Set 1, Bank 0 Register and 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 | 208 | D0H | 0 | 0 | 0 | 0 | 0 | 0 | – | 0 |
| Watch Timer Control Register | WTCON | 209 | D1H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Battery Level Detector Control Register | BLDCON | 210 | D2H | – | – | 0 | 0 | 0 | 0 | 0 | 0 |
| SIO Control Register | SIOCON | 224 | E0H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIO Data Register | SIODATA | 225 | E1H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SIO Pre-Scaler Register | SIOPS | 226 | E2H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Timer 0 Control Register | T0CON | 227 | E3H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Timer 0 Counter Register(High Byte) | T0CNTH | 228 | E4H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Timer 0 Counter Register(Low Byte) | T0CNTL | 229 | E5H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Timer 0 Data Register(High Byte) | T0DATAH | 230 | E6H | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Timer 0 Data Register(Low Byte) | T0DATAL | 231 | E7H | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Timer A Control Register | TACON | 232 | E8H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Timer A Counter Register | TACNT | 233 | E9H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Timer A Data Register | TADATA | 234 | EAH | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Timer 1 Control Register | T1CON | 235 | EBH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Timer 1 Counter Register(High Byte) | T1CNTH | 236 | ECH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Timer 1 Counter Register(Low Byte) | T1CNTL | 237 | EDH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Timer 1 Data Register(High Byte) | T1DATAH | 238 | EEH | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Timer 1 Data Register(Low Byte) | T1DATAL | 239 | EFH | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Timer B Data Register(High Byte) | TBDATAH | 240 | F0H | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Timer B Data Register(Low Byte) | TBDATAL | 241 | F1H | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Timer B Control Register | TBCON | 242 | F2H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A/D Converter Control Register | ADCON | 243 | F3H | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A/D Converter Data Register(High Byte) | ADDATAH | 244 | F4H | x | x | x | x | x | x | x | x |
| A/D Converter Data Register(Low Byte) | ADDATAL | 245 | F5H | – | – | – | – | – | – | x | x |
| UART Control Register | UARTCON | 246 | F6H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| UART Data Register | UDATA | 247 | F7H | x | x | x | x | x | x | x | x |
| UART Baud Rate Data Register | BRDATA | 248 | F8H | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Interrupt Pending Register | INTPND | 249 | F9H | – | – | 0 | 0 | 0 | 0 | 0 | 0 |
| Oscillator Control Register | OSCCON | 250 | FAH | 0 | – | – | – | 0 | 0 | – | 0 |
| 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 |

**SAMSUNG**
ELECTRONICS

**Table 8-3. S3C828B/F828B/C8289/F8289/C8285/F8285 Set 1, Bank 1 Register and Values after RESET**

| Register Name | Mnemonic | Address | | Bit Values after RESET | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Dec | Hex | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Flash Memory Sector Address Register (High Byte) | FMSECH | 208 | D0H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Flash Memory Sector Address Register (Low Byte) | FMSECL | 209 | D1H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Flash Memory Control Register | FMCON | 210 | D2H | 0 | 0 | 0 | 0 | 0 | – | – | 0 |
| Port 0 Control Register (High Byte) | P0CONH | 224 | E0H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 0 Control Register (Low Byte) | P0CONL | 225 | E1H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 0 Interrupt Control Register (High Byte) | P0INTH | 226 | E2H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 0 Interrupt Control Register(Low Byte) | P0INTL | 227 | E3H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 0 Interrupt Pending Register | P0PND | 228 | E4H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 1 Control Register (High Byte) | P1CONH | 229 | E5H | – | – | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 1 Control Register (Low Byte) | P1CONL | 230 | E6H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 1 Pull-up Resistor Enable Register | P1PUR | 231 | E7H | – | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 2 Control Register (High Byte) | P2CONH | 232 | E8H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 2 Control Register (Low Byte) | P2CONL | 233 | E9H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 3 Control Register (High Byte) | P3CONH | 234 | EAH | – | – | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 3 Control Register (Low Byte) | P3CONL | 235 | EBH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 4 Control Register (High Byte) | P4CONH | 236 | ECH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 4 Control Register (Low Byte) | P4CONL | 237 | EDH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 4 Pull-up Resistor Enable Register | P4PUR | 238 | EEH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 5 Pull-up Resistor Enable Register | P5PUR | 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 |
| 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 |
| Port 4 Data Register | P4 | 244 | F4H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 5 Data Register | P5 | 245 | F5H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 6 Data Register | P6 | 246 | F6H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 7 Data Register | P7 | 247 | F7H | – | – | – | – | 0 | 0 | 0 | 0 |
| Port 8 Data Register | P8 | 248 | F8H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 5 Control Register (High Byte) | P5CONH | 249 | F9H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 5 Control Register (Low Byte) | P5CONL | 250 | FAH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 6 Control Register (High Byte) | P6CONH | 251 | FBH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 6 Control Register (Low Byte) | P6CONL | 252 | FCH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 7 Control Register | P7CON | 253 | FDH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Port 8 Control Register | P8CON | 254 | FEH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Flash Memory User Programming Enable Register | FMUSR | 255 | FFH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 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 $3\mu A$. 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 7-3.

### NOTE

Do not use stop mode if you are using an external clock source because $X_{IN}$ input must be restricted internally to $V_{SS}$ to reduce current leakage.

### Using 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 S3C828B/F828B/C8289/F8289/C8285/F8285 interrupt structure that can be used to release Stop mode are:

— External interrupts P0.0–P0.7 (INT0–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.

### Using an Internal Interrupt to Release Stop Mode

Activate any enabled interrupt, causing Stop mode to be released. Other things are same as using external interrupt.

### 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 timers remain active. Port pins retain the mode (input or output) they had at the time idle mode was entered.

There are two ways to release idle mode:

1.  Execute a reset. All system and peripheral control registers are reset to their default values and the contents of all data registers are retained. The reset automatically selects the slow clock fxx/16 because CLKCON.4 and CLKCON.3 are cleared to '00B'. If interrupts are masked, a reset is the only way to release idle mode.

2.  Activate any enabled interrupt, causing idle mode to be released. When you use an interrupt to release idle mode, the CLKCON.4 and CLKCON.3 register values remain unchanged, and the currently selected clock value is used. The interrupt is then serviced. When the return-from-interrupt (IRET) occurs, the instruction immediately following the one that initiated idle mode is executed.

# 9 I/O PORTS

## OVERVIEW

The S3C828B/F828B/C8289/F8289/C8285/F8285 microcontroller has nine bit-programmable I/O ports, P0–P8. The port 1 is a 7-bit port, the port 3 is a 6-bit port, the port 7 is a 4-bit port, and the others are 8-bit ports. This gives a total of 65 I/O pins. Each port can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required.

Table 9-1 gives you a general overview of the S3C828B/F828B/C8289/F8289/C8285/F8285 I/O port functions.

**Table 9-1. S3C828B/F828B/C8289/F8289/C8285/F8285 Port Configuration Overview**

| Port | Configuration Options |
|------|----------------------|
| 0 | 1-bit programmable I/O port.<br>Schmitt trigger input or push-pull open-drain output mode selected by software; software assignable pull-ups.<br>P0.0–P0.7 can be used as inputs for external interrupts INT0–INT7<br>(with noise filter, interrupt enable and pending control). |
| 1 | 1-bit programmable I/O port.<br>Schmitt trigger input or push-pull, open-drain output mode selected by software; software assignable pull-ups.<br>Alternately P1.0–P1.6 can be used as T1CAP, T1CLK, T1OUT, T1PWM, BUZ, SO, SCK, SI. |
| 2 | 1-bit programmable I/O port.<br>Input or push-pull output mode selected by software; software assignable pull-ups. Alternatively P2.0-P2.7 can be used as AD0–AD7/$V_{BLDREF}$. |
| 3 | 1-bit programmable I/O port.<br>Input or push-pull output mode selected by software; software assignable pull-ups.<br>Alternately P3.0–P3.5 can be used as TBPWM, TAOUT/TAPWM, TACLK, TACAP, TxD, RxD or LCD SEG. |
| 4 | 1-bit programmable I/O port.<br>Input or push-pull, open drain output mode selected by software; software assignable pull-ups.<br>P4.0–P4.7 can alternately be used as outputs for LCD SEG. |
| 5 | 1-bit programmable I/O port.<br>Input or push-pull, open drain output mode selected by software; software assignable pull-ups.<br>P5.0–P5.7 can alternately be used as outputs for LCD SEG. |
| 6 | 1-bit programmable I/O port.<br>Input or push-pull output mode selected by software; software assignable pull-ups.<br>P6.0–P6.7 can alternately be used as outputs for LCD SEG. |
| 7 | 1-bit programmable I/O port.<br>Input or push-pull output mode selected by software; software assignable pull-ups.<br>P7.0–P7.3 can alternately be used as outputs for LCD SEG. |
| 8 | 1-bit or 2-bit or 4-bit programmable I/O port.<br>Input or push-pull, open drain output mode selected by software; software assignable pull-ups.<br>P8.0–P8.7 can alternately be used as outputs for LCD COM/SEG. |

SAMSUNG
ELECTRONICS

9-1

**PORT DATA REGISTERS**

Table 9-2 gives you an overview of the register locations of all four S3C828B/F828B/C8289/F8289/C8285/F8285 I/O port data registers. Data registers for ports 0, 1, 2, 3, 4, 5, 6, 7 and 8 have the general format shown in Figure 9-1.

**Table 9-2. Port Data Register Summary**

| Register Name | Mnemonic | Decimal | Hex | Location | R/W |
|---|---|---|---|---|---|
| Port 0 data register | P0 | 240 | F0H | Set 1, Bank 1 | R/W |
| Port 1 data register | P1 | 241 | F1H | Set 1, Bank 1 | R/W |
| Port 2 data register | P2 | 242 | F2H | Set 1, Bank 1 | R/W |
| Port 3 data register | P3 | 243 | F3H | Set 1, Bank 1 | R/W |
| Port 4 data register | P4 | 244 | F4H | Set 1, Bank 1 | R/W |
| Port 5 data register | P5 | 245 | F5H | Set 1, Bank 1 | R/W |
| Port 6 data register | P6 | 246 | F6H | Set 1, Bank 1 | R/W |
| Port 7 data register | P7 | 247 | F7H | Set 1, Bank 1 | R/W |
| Port 8 data register | P8 | 248 | F8H | Set 1, Bank 1 | R/W |

SAMSUNG
ELECTRONICS

### PORT 0

Port 0 is an 8-bit I/O Port that you can use two ways:

— General-purpose I/O
— External interrupt inputs for INT0–INT7

Port 0 is accessed directly by writing or reading the port 0 data register, P0 at location F0H in set 1, bank 1.

### Port 0 Control Register (P0CONH, P0CONL)

Port 0 has two 8-bit control registers: P0CONH for P0.4-P0.7 andP0CONL for P0.0-P0.3. A reset clears the P0CONH and P0CONL registers to "00H", configuring all pins to input mode. In input mode, three different selections are available:

— Schmitt trigger input with interrupt generation on falling signal edges.
— Schmitt trigger input with interrupt generation on rising signal edges.
— Schmitt trigger input with interrupt generation on falling/rising signal edges.

### Port 0 Interrupt Enable and Pending Registers (P0INTH, P0INTL)

To process external interrupts at the port 0 pins, the additional control registers are provided: the port 0 interrupt enable register P0INTH (high byte, E2H, set 1, bank 1), P0INTL (Low byte, E3H, set1, bank1) and the port 0 interrupt pending register P0PND (E4H, set 1, bank 1).

The port 0 interrupt pending register P0PND 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 P0PND register at regular intervals.

When the interrupt enable bit of any port 0 pin is "1", a rising or falling signal edge at that pin will generate an interrupt request. The corresponding P0PND 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 P0PND bit.
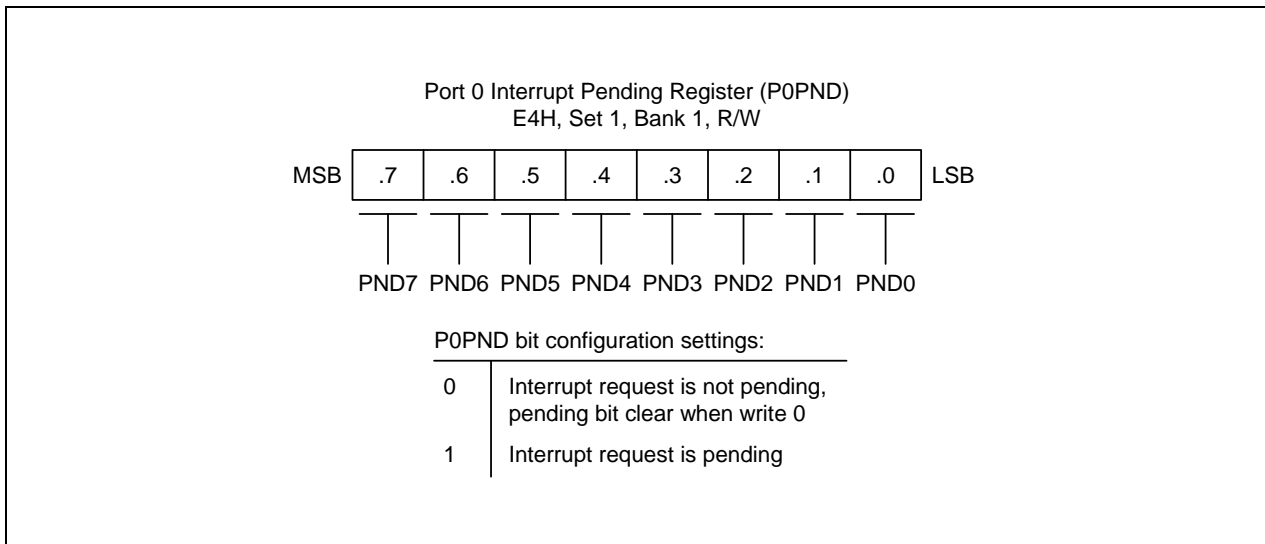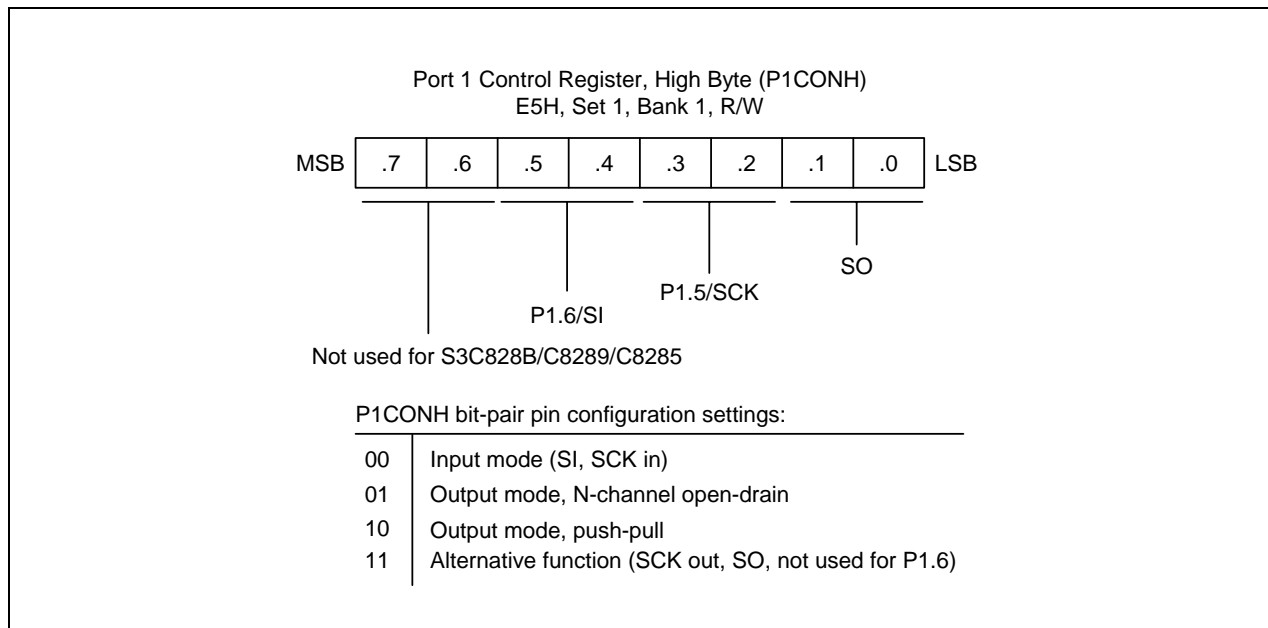
Port 0 Control Register, High Byte (P0CONH)
E0H, Set 1, Bank 1, R/W

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

P0.7         P0.6         P0.5         P0.4
(INT7)       (INT6)       (INT5)       (INT4)

P0CONH bit-pair pin configuration settings:

| 00 | Schmitt trigger input mode |
| 01 | Schmitt trigger input mode, pull-up |
| 10 | Output mode, open-drain |
| 11 | Output mode, push-pull |
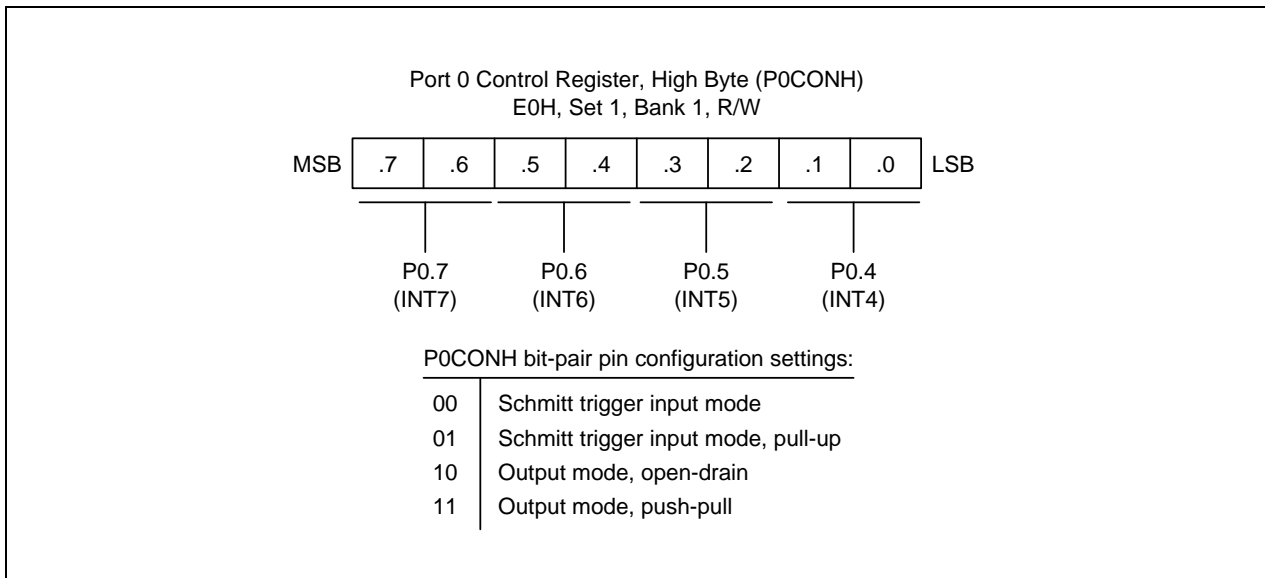
**Figure 9-1. Port 0 High-Byte Control Register (P0CONH)**

Port 0 Control Register, Low Byte (P0CONL)
E1H, Set 1, Bank 1, R/W

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

P0.3         P0.2         P0.1         P0.0
(INT3)       (INT2)       (INT1)       (INT0)

P0CONL bit-pair pin configuration settings:

| 00 | Schmitt trigger input mode |
| 01 | Schmitt trigger input mode, pull-up |
| 10 | Output mode, open-drain |
| 11 | Output mode, push-pull |
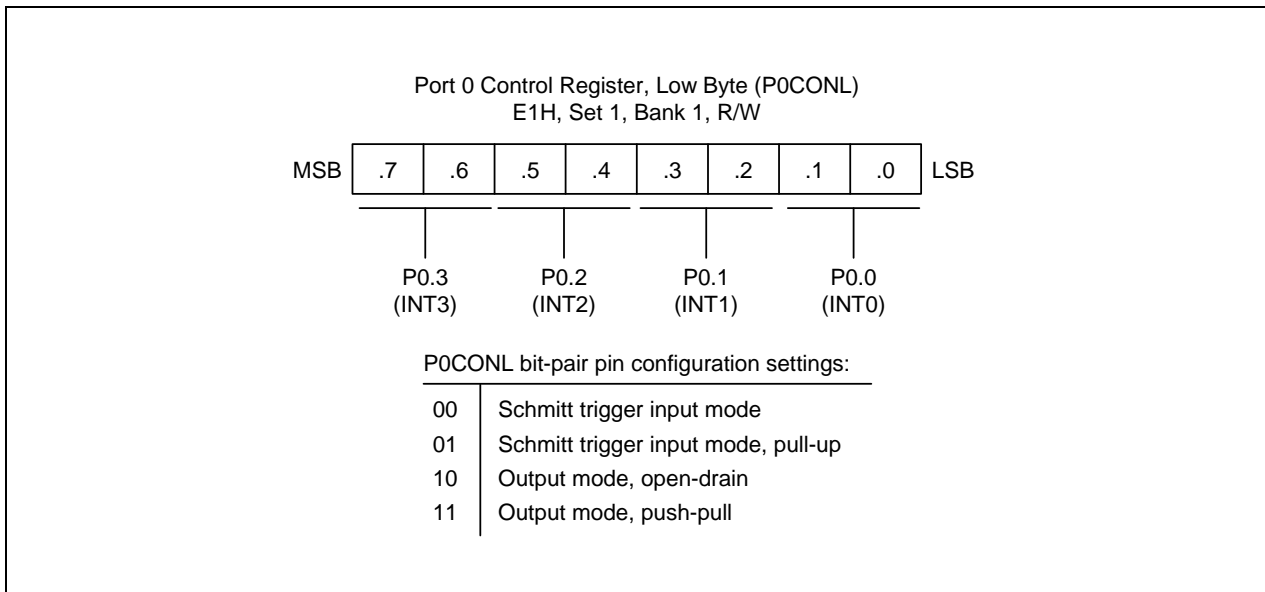
**Figure 9-2. Port 0 Low-Byte Control Register (P0CONL)**

SAMSUNG
ELECTRONICS

Port 0 Interrupt Control Register, High Byte (P0INTH)
E2H, Set 1, Bank 1, R/W

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

INT7     INT6     INT5     INT4

P0INTH 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 |
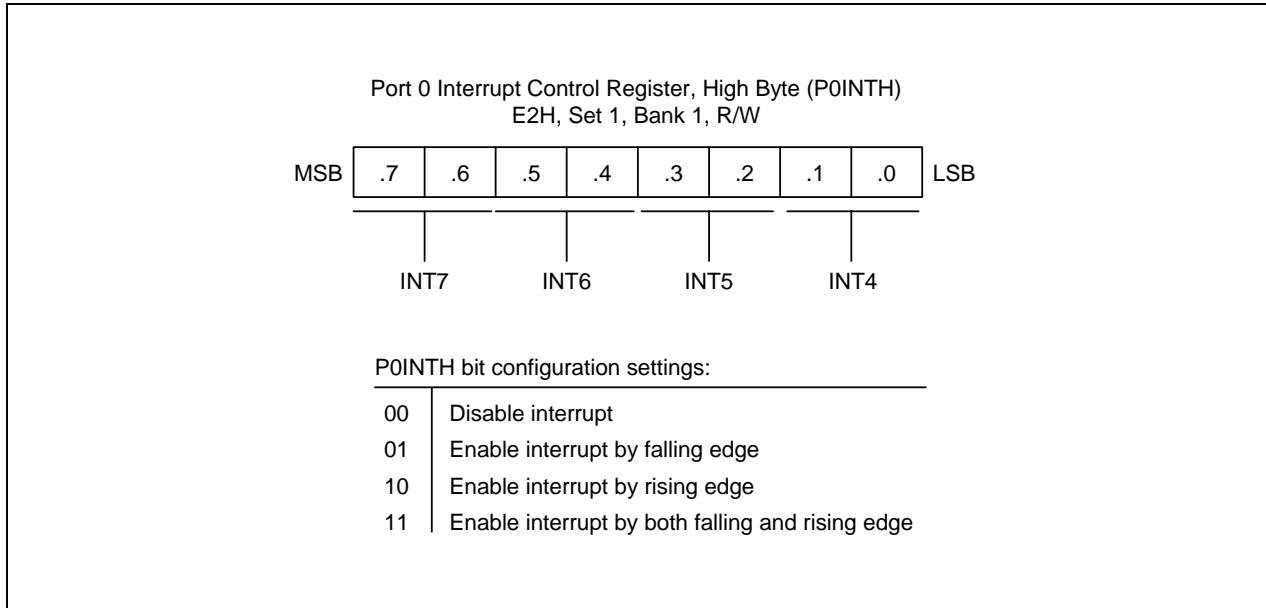
**Figure 9-3. Port 0 High-Byte Interrupt Control Register (P0INTH)**

Port 0 Interrupt Control Register, Low Byte (P0INTL)
E3H, Set 1, Bank 1, R/W

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

INT3     INT2     INT1     INT0

P0INTL 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 |
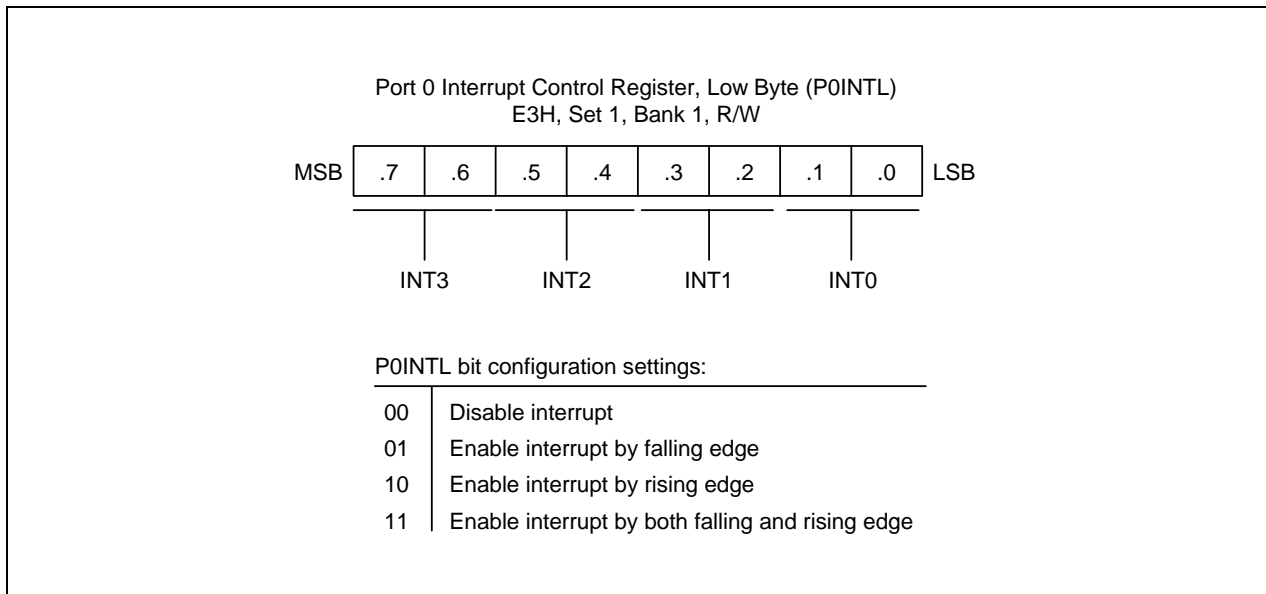
**Figure 9-4. Port 0 Low-Byte Interrupt Control Register(P0INTL)**

SAMSUNG
ELECTRONICS

Port 0 Interrupt Pending Register (P0PND)
E4H, Set 1, Bank 1, R/W

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

PND7 PND6 PND5 PND4 PND3 PND2 PND1 PND0

P0PND bit configuration settings:

| 0 | Interrupt request is not pending, pending bit clear when write 0 |
|---|---|
| 1 | Interrupt request is pending |

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

SAMSUNG
ELECTRONICS

**PORT 1**

Port 1 is an 7-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 1. P1.0–P1.6 can serve inputs, as outputs (push pull or open-drain) or you can configure the following alternative functions:

— Low-byte pins (P1.0-P1.3): T1CAP, T1CLK, T1OUT, T1PWM, BUZ
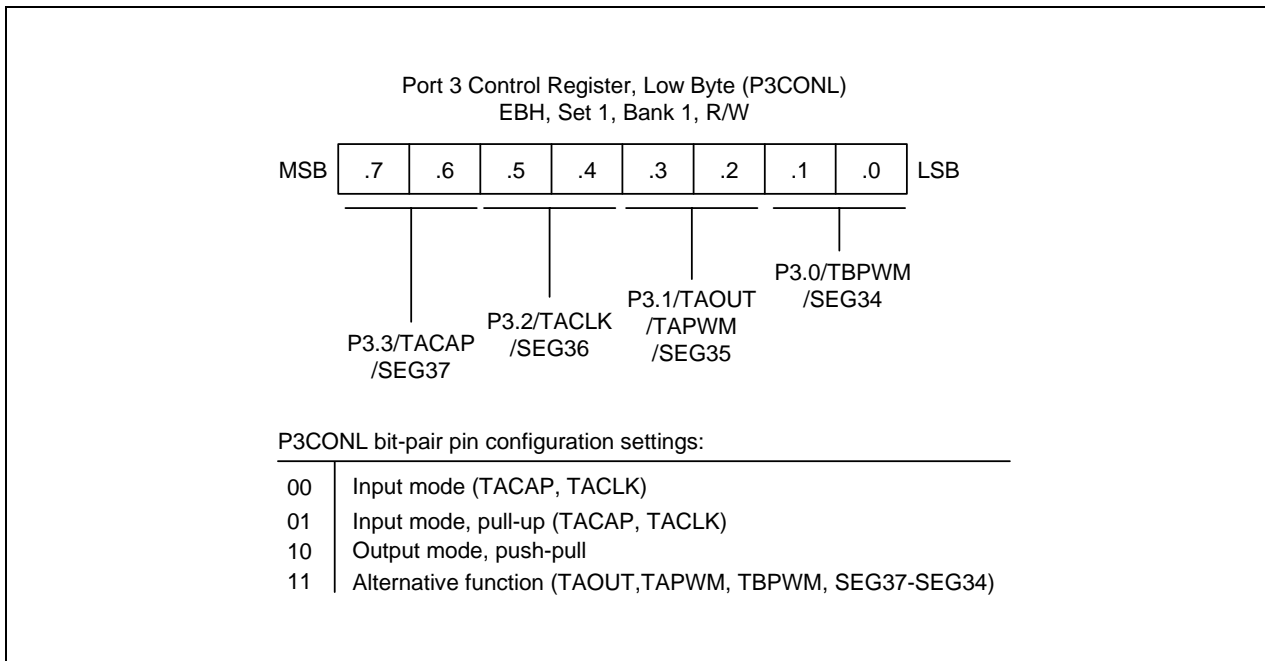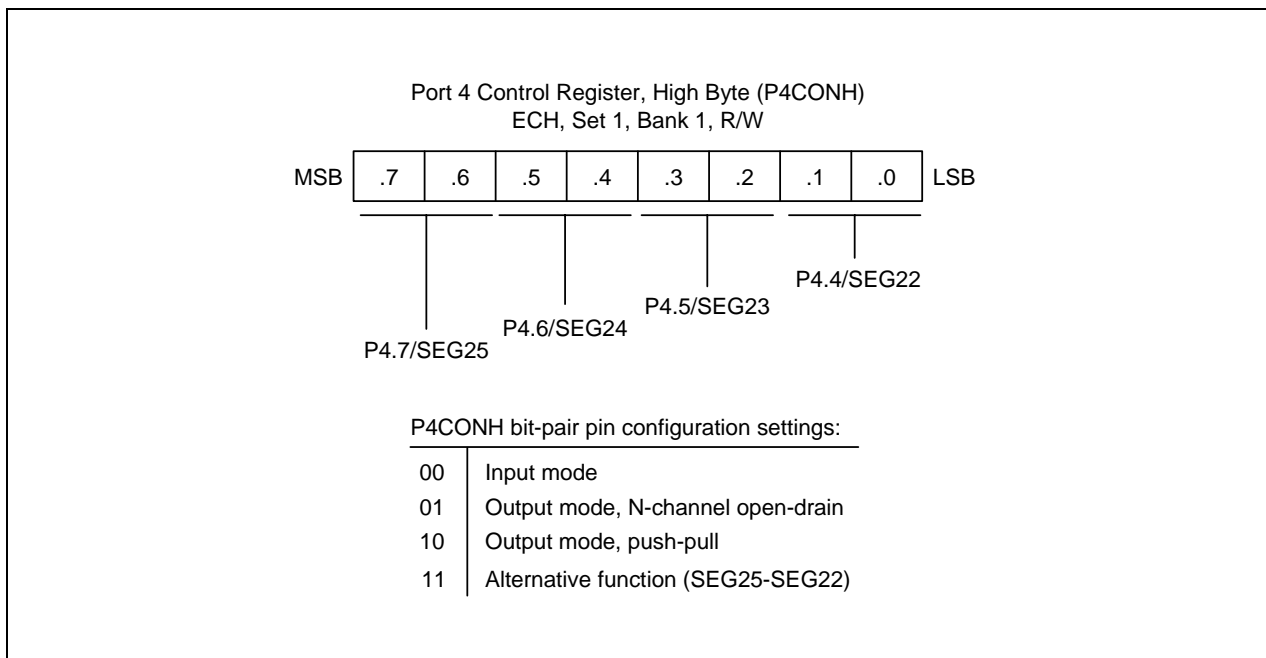
— High-byte pins (P1.4-P1.6): SO, SCK, SI

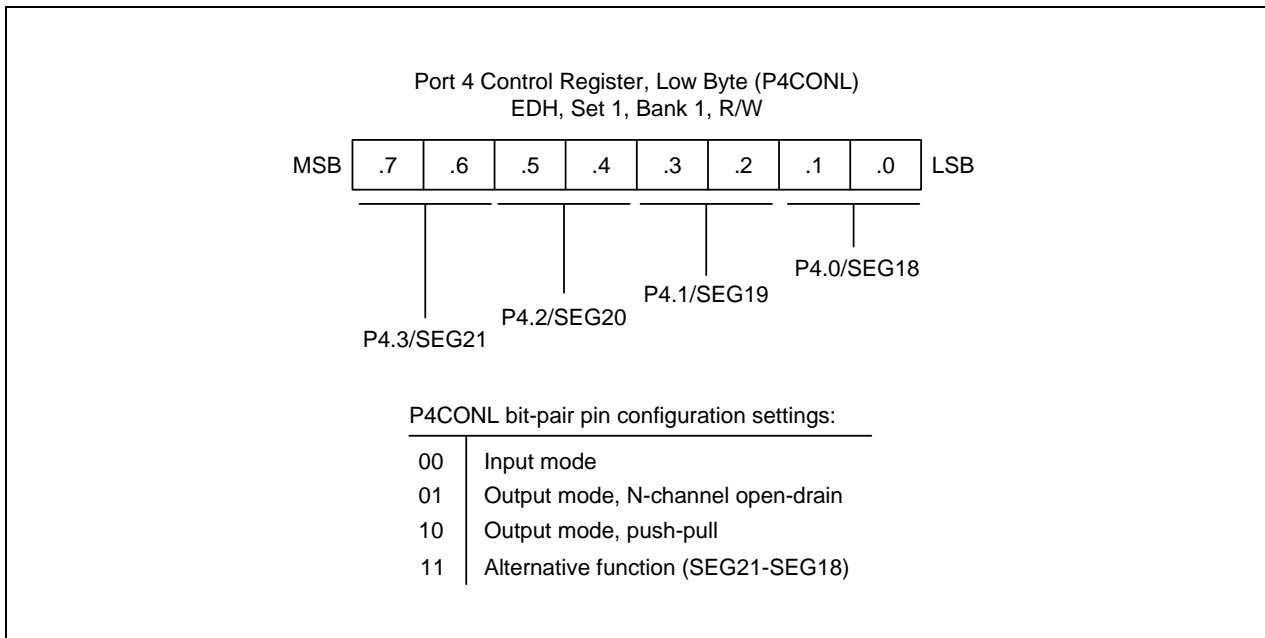**Port 1 Control Register (P1CONH, P1CONL)**

Port 1 has two 8-bit control registers: P1CONH for P1.4–P1.6 and P1CONL for P1.0–P1.3. A reset clears the P1CONH and P1CONL registers to "00H", configuring all pins to input mode. You use control registers settings to select input or output mode (push-pull or open drain) and enable the alternative functions.

When programming the port, please remember that any alternative peripheral I/O function you configure using the port 1 control registers must also be enabled in the associated peripheral module.

**Port 1 Pull-up Resistor Enable Register (P1PUR)**

Using the port 1 pull-up resistor enable register, P1PUR (E7H, set 1, bank 1), you can configure pull-up resistors to individual port 1 pins.

Port 1 Control Register, High Byte (P1CONH)
E5H, Set 1, Bank 1, R/W

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

SO

P1.5/SCK

P1.6/SI

Not used for S3C828B/C8289/C8285

P1CONH bit-pair pin configuration settings:

| | |
|---|---|
| 00 | Input mode (SI, SCK in) |
| 01 | Output mode, N-channel open-drain |
| 10 | Output mode, push-pull |
| 11 | Alternative function (SCK out, SO, not used for P1.6) |

**Figure 9-6. Port 1 High-Byte Control Register (P1CONH)**

Port 1 Control Register, Low Byte (P1CONL)
E6H, Set 1, Bank 1, R/W

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

P1.0/T1CAP

P1.1/T1CLK

P1.2/T1OUT
/T1PWM

P1.3/
BUZ

P1CONL bit-pair pin configuration settings:

| 00 | Input mode (T1CAP, T1CLK) |
|----|---------------------------|
| 01 | Output mode, N-channel open-drain |
| 10 | Output mode, push-pull |
| 11 | Alternative function (BUZ, T1OUT, T1PWM, not used for P1.0-P1.1) |

**NOTE:**   When use this port 1, user must be care of the pull-up resistance status.
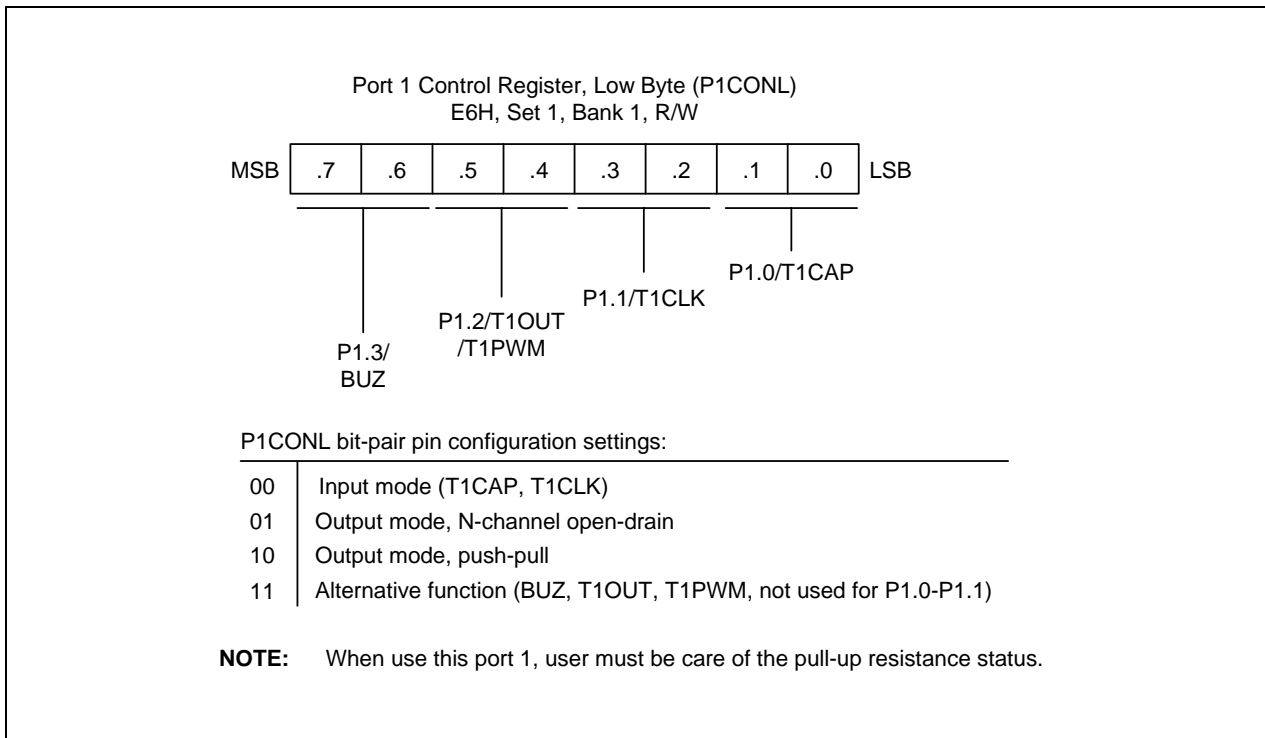
**Figure 9-7. Port 1 Low-Byte Control Register (P1CONL)**

Port 1 Pull-up Resistor Enable Register (P1PUR)
E7H, Set 1, Bank 1, R/W

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

P1.6  P1.5  P1.4  P1.3  P1.2  P1.1  P1.0

Not used for the S3C828B/C8289/C8285

P1PUR bit configuration settings:

| 0 | Pull-up Disable |
|---|-----------------|
| 1 | Pull-up Enable |
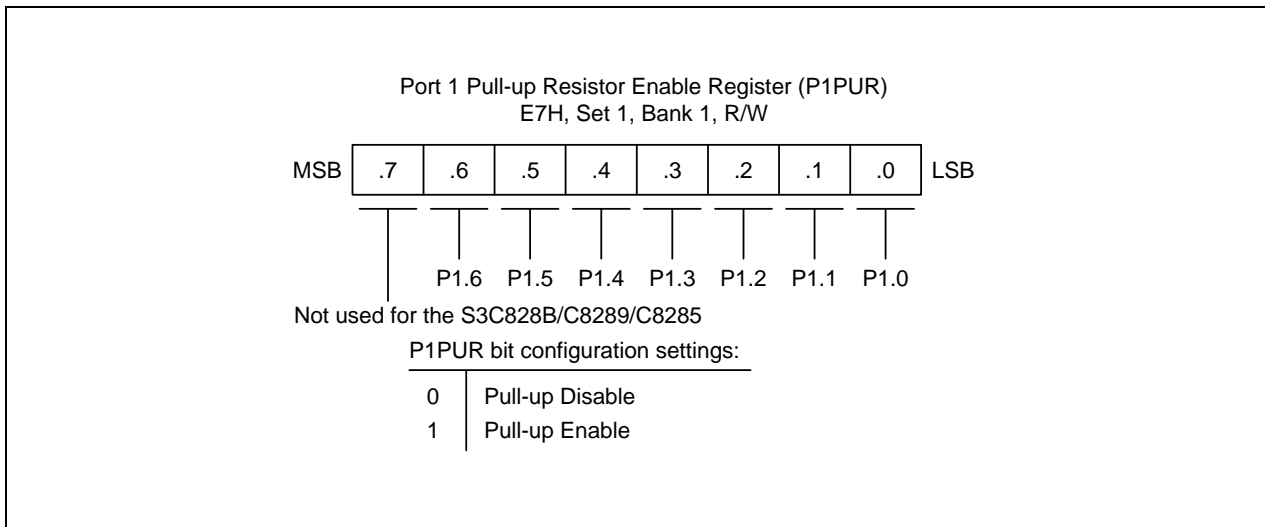
**Figure 9-8. Port 1 Pull-up Resistor Enable Register (P1PUR)**

SAMSUNG
ELECTRONICS

### PORT 2

Port 2 is an 8-bit I/O port that can be used for general-purpose I/O as A/D converter inputs, ADC0–ADC7. The pins are accessed directly by writing or reading the port 2 data register, P2 at location F2H in set 1, bank 1.

P2.0–P2.7 can serve as inputs, as outputs(push-pull) or you can configure the following alternative functions. In input mode, ADC or external reference voltage input are also available.

— Low byte pins (P2.0–P2.3): AD0–AD3

— High byte pins (P2.4–P2.7): AD4–AD7, $V_{BLDREF}$

### Port 2 Control Registers (P2CONH, P2CONL)
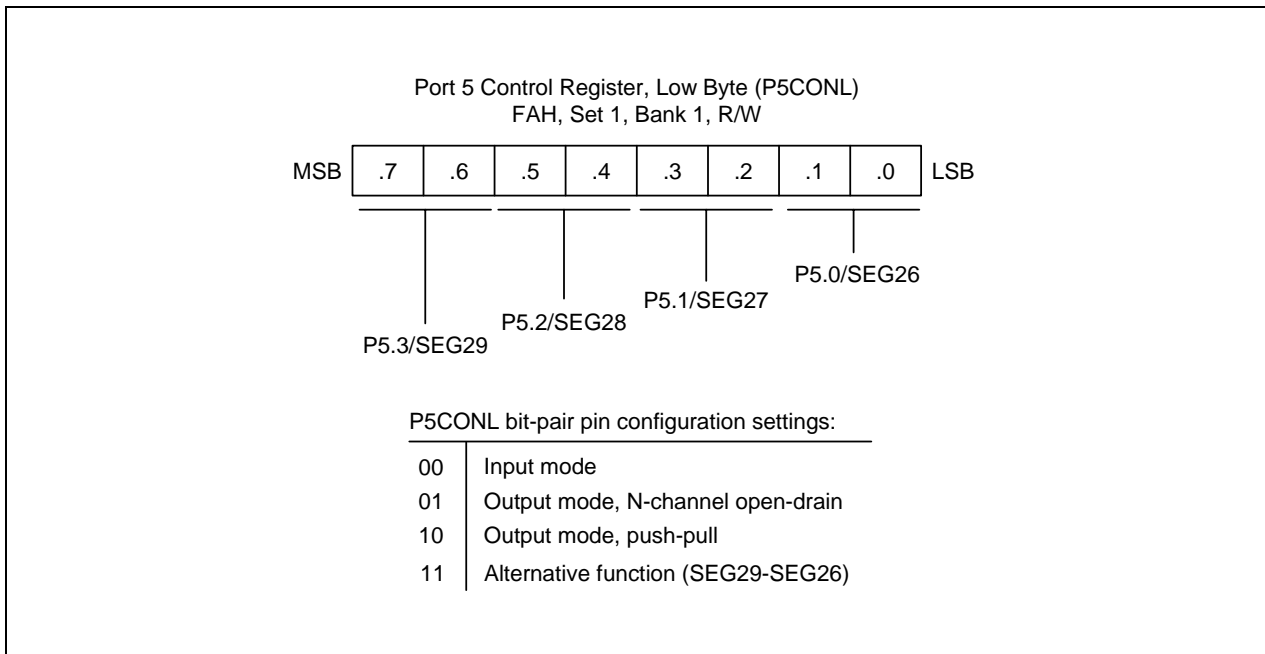
Port 2 has two 8-bit control registers: P2CONH for P2.4–P2.7 and P2CONL for P2.0-P2.3. A reset clears the P2CONH and P2CONL registers also control the alternative functions.

Port 2 Control Register, High Byte (P2CONH)
E8H, Set 1, Bank 1, R/W

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

P2.4 (AD4)

P2.5 (AD5)

P2.6 (AD6)

P2.7
(AD7/$V_{BLDREF}$)

P2CONH bit-pair pin configuration settings:

| | |
|---|---|
| 00 | Input mode |
| 01 | Input mode, pull-up |
| 10 | Output mode, push-pull |
| 11 | Alternative function (AD4-AD7, BLD external input enable) |
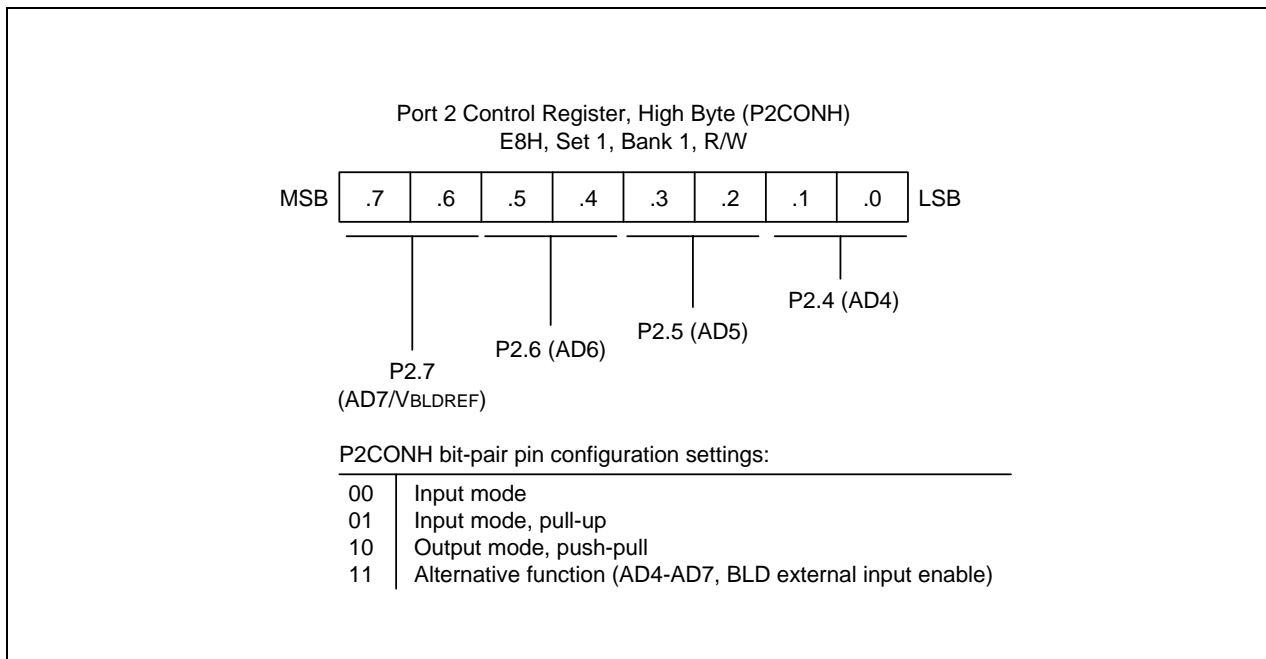
**Figure 9-9. Port 2 High-Byte Control Register (P2CONH)**

Port 2 Control Register,Low Byte (P2CONL)
E9H, Set 1, Bank 1, R/W

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

P2.0 (AD0)

P2.1 (AD1)

P2.2 (AD2)

P2.3 (AD3)

P2CONL bit-pair pin configuration settings:

| 00 | Input mode |
| 01 | Input mode, pull-up |
| 10 | Output mode, push-pull |
| 11 | Alternative function (AD0-AD3) |
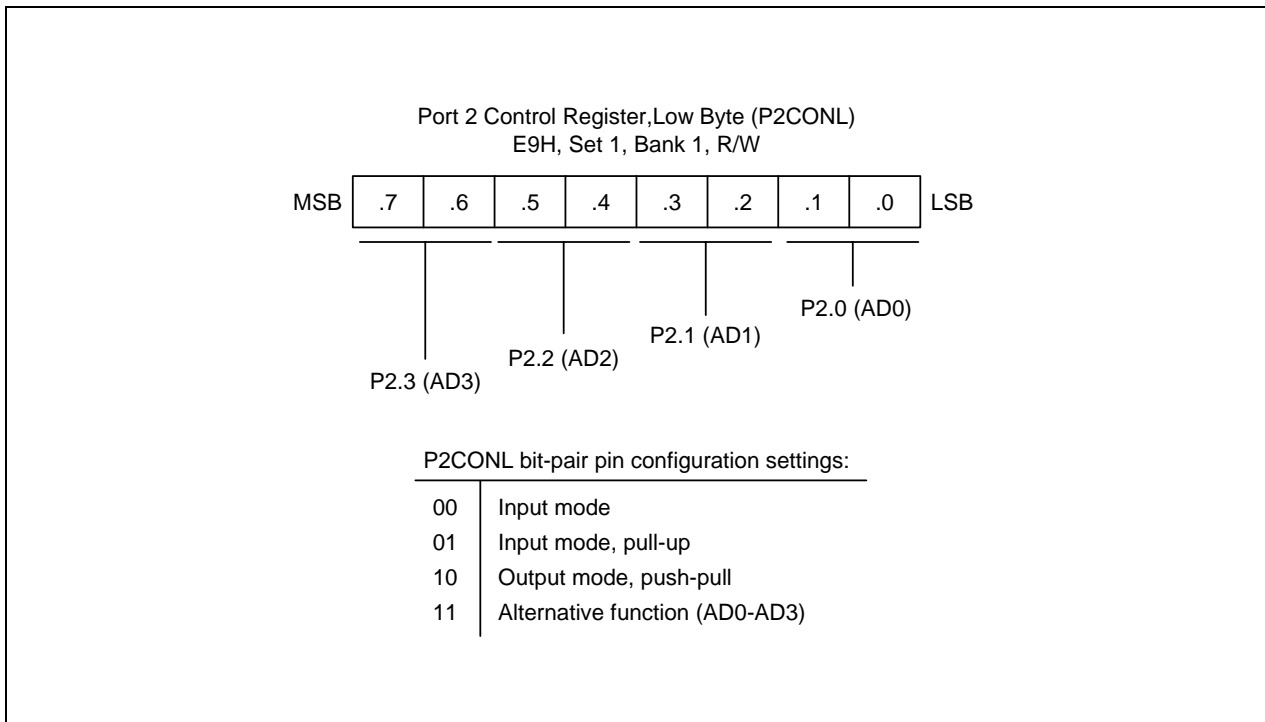
**Figure 9-10. Port 2 Low-Byte Control Register (P2CONL)**
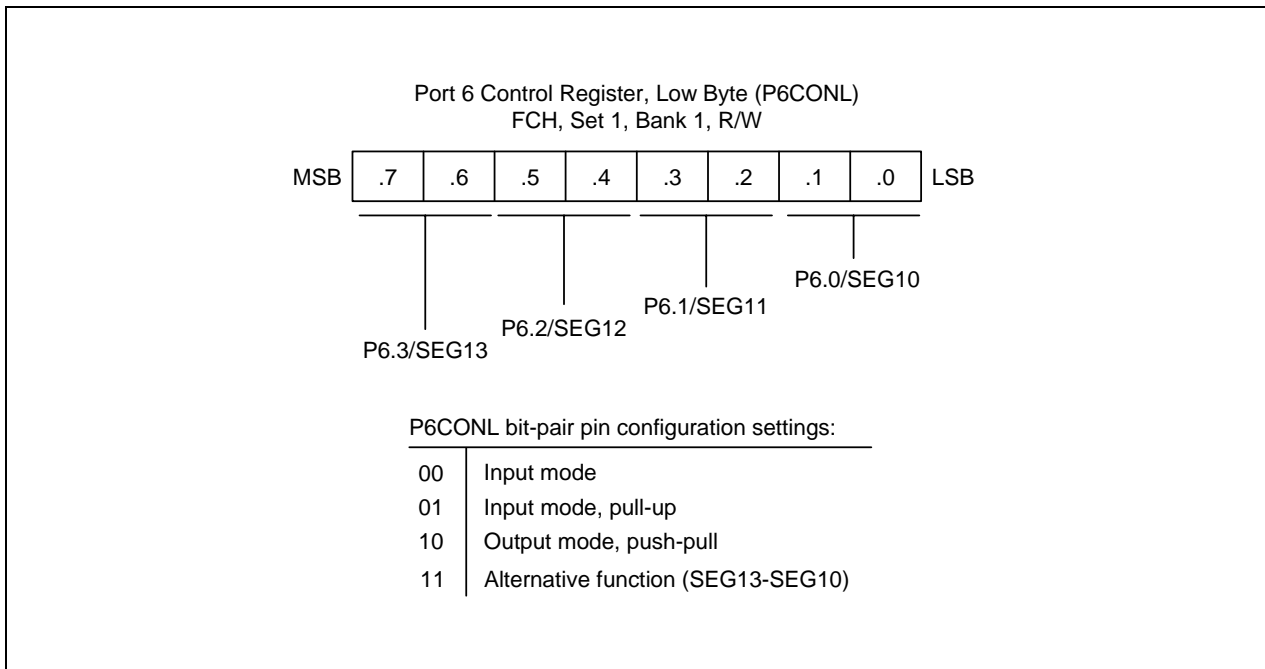
SAMSUNG
ELECTRONICS

## PORT 3

Port 3 is an 6-bit I/O port with individually configurable pins. Port 3 pins are accessed directly by writing or reading the port 3 data register, P3 at location F3H in set 1, bank 1. P3.0–P3.5 can serve as inputs (with or without pull-ups), as push-pull outputs. And the P3.0–P3.3 can serve as segment pins for LCD or you can configure the following alternative functions:

— Low-byte pins (P3.0–P3.3): TBPWM, TAOUT, TAPWM, TACLK, TACAP

— High-byte pins (P3.4–P3.6): TxD, RxD

### Port 3 Control Registers (P3CONH, P3CONL)

Port 3 has two 8-bit control registers: P3CONH for P3.4–P3.5 and P3CONL for P3.0–P3.3. A reset clears the P3CONH and P3CONL registers to "00H", configuring all pins to input mode. You use control registers settings to select input or output mode, enable pull-up resistors, 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.
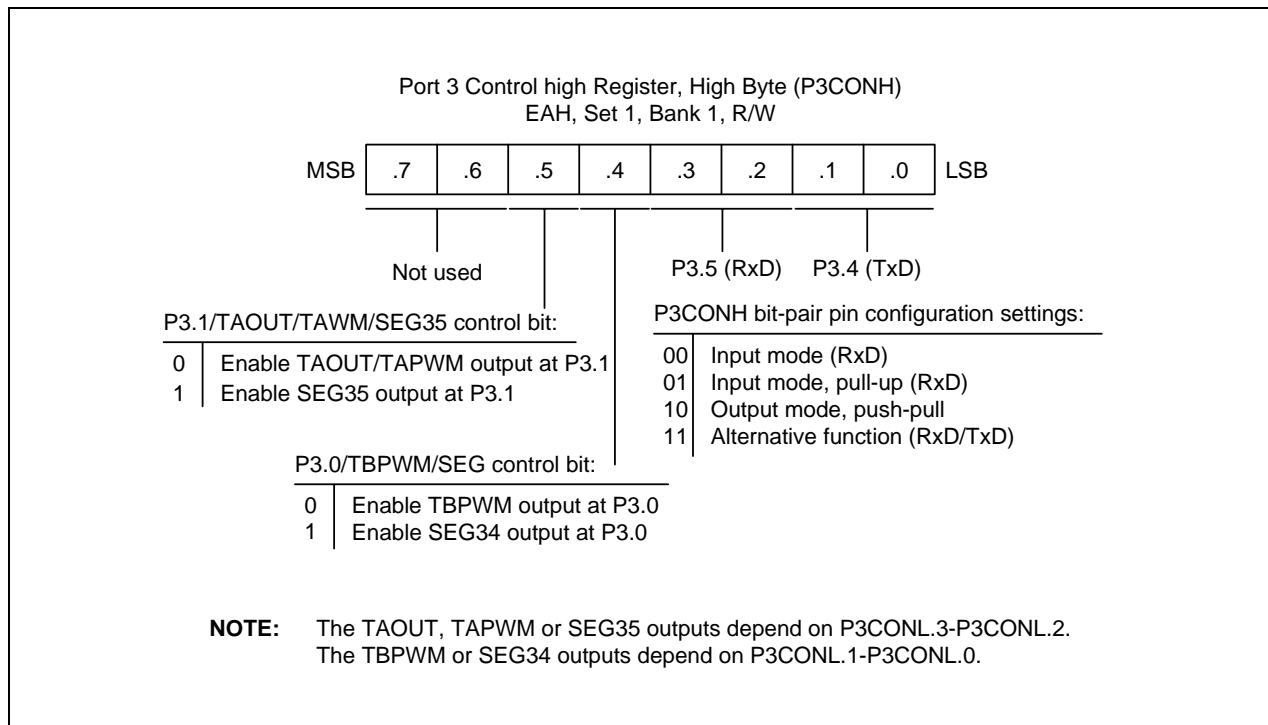


**Figure 9-11. Port 3 High-Byte Control Register (P3CONH)**

Port 3 Control Register, Low Byte (P3CONL)
EBH, Set 1, Bank 1, R/W

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

P3.0/TBPWM
/SEG34

P3.1/TAOUT
/TAPWM
/SEG35

P3.2/TACLK
/SEG36

P3.3/TACAP
/SEG37

P3CONL bit-pair pin configuration settings:

| 00 | Input mode (TACAP, TACLK) |
| 01 | Input mode, pull-up (TACAP, TACLK) |
| 10 | Output mode, push-pull |
| 11 | Alternative function (TAOUT,TAPWM, TBPWM, SEG37-SEG34) |

**Figure 9-12. Port 3 Low-Byte Control Register (P3CONL)**

## PORT 4

Port 4 is an 8-bit I/O port with individually configurable pins. Port 4 pins are accessed directly by writing or reading the port 4 data register, P4 at location F4H in set 1, bank 1. P4.0–P4.7 can serve as inputs (with or without pull-ups), as output (open drain or push-pull). And, they can serve as segment pins for LCD also.

### 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.

### Port 4 Pull-up Resistor Enable Register (P4PUR)

Using the Port 4 pull-up resistor enable register, P4PUR (EEH, set 1, bank 1), you can configure pull-up resistors to individual port 4 pins.

Port 4 Control Register, High Byte (P4CONH)
ECH, Set 1, Bank 1, R/W

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

P4.4/SEG22

P4.5/SEG23

P4.6/SEG24

P4.7/SEG25

P4CONH bit-pair pin configuration settings:

| 00 | Input mode |
|----|------------|
| 01 | Output mode, N-channel open-drain |
| 10 | Output mode, push-pull |
| 11 | Alternative function (SEG25-SEG22) |

**Figure 9-13. Port 4 High-Byte Control Register (P4CONH)**

Port 4 Control Register, Low Byte (P4CONL)
EDH, Set 1, Bank 1, R/W

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

P4.0/SEG18

P4.1/SEG19

P4.2/SEG20

P4.3/SEG21

P4CONL bit-pair pin configuration settings:

| 00 | Input mode |
| 01 | Output mode, N-channel open-drain |
| 10 | Output mode, push-pull |
| 11 | Alternative function (SEG21-SEG18) |

**Figure 9-14. Port 4 Low-Byte Control Register (P4CONL)**

Port 4 Pull-up Resistor Enable Register (P4PUR)
EEH, Set 1, Bank 1, R/W

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

P4.7  P4.6  P4.5  P4.4  P4.3  P4.2  P4.1  P4.0

P4PUR bit configuration settings:

| 0 | Pull-up Disable |
| 1 | Pull-up Enable |
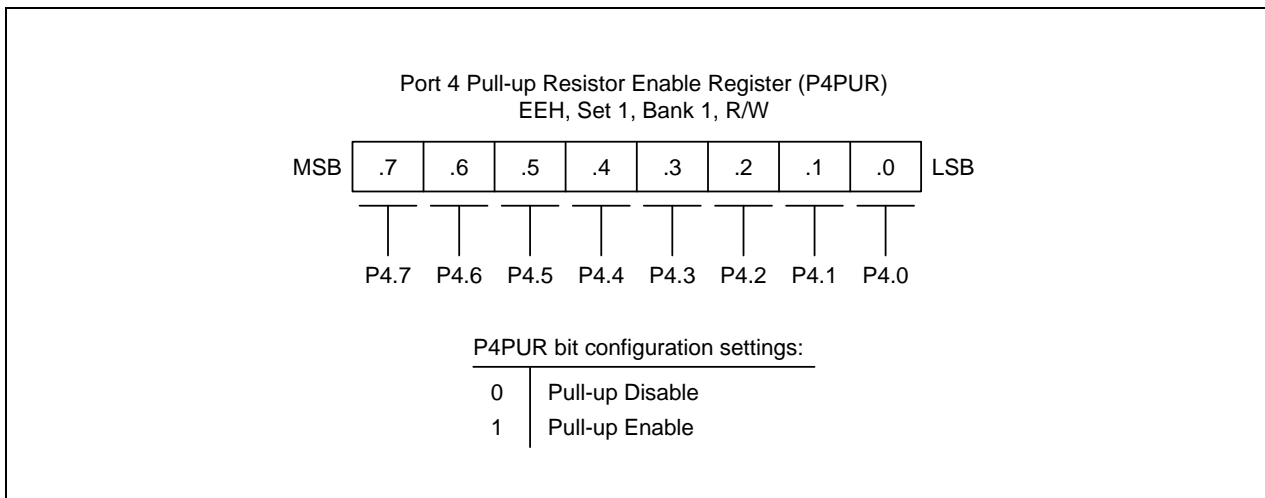
**Figure 9-15. Port 4 Pull-up Resistor Enable Register (P4PUR)**

SAMSUNG
ELECTRONICS

## 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 1. P5.0–P5.7 can serve as inputs (with without pull-ups), as output (open drain or push-pull). And, they can serve as segment pins for LCD also.

### 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.

### Port 5 Pull-up Resistor Enable Register (P5PUR)

Using the port 5 pull-up resistor enable register, P5PUR (EFH, set1, bank1), you can configure pull-up resistors to individual port 5 pins.

Port 5 Control Register, High Byte (P5CONH)
F9H, Set 1, Bank 1, R/W

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

P5.4/SEG30

P5.5/SEG31

P5.6/SEG32

P5.7/SEG33

P5CONH bit-pair pin configuration settings:

| | |
|----|----|
| 00 | Input mode |
| 01 | Output mode, N-channel open-drain |
| 10 | Output mode, push-pull |
| 11 | Alternative function (SEG33-SEG30) |
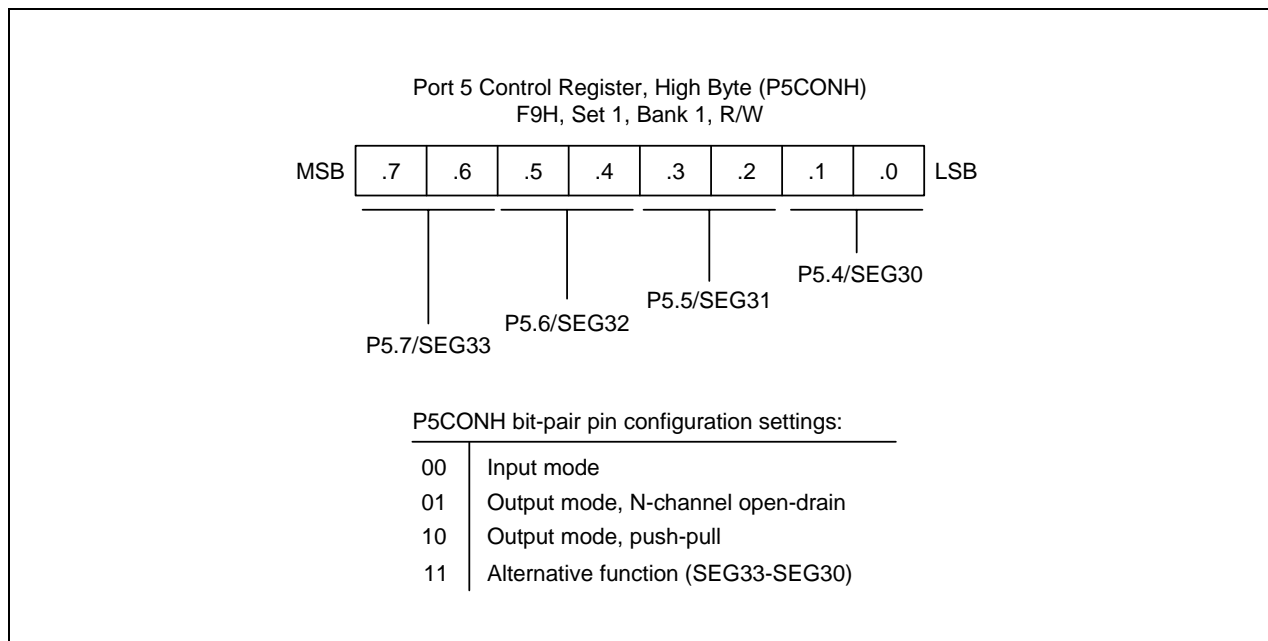
**Figure 9-16. Port 5 High-Byte Control Register (P5CONH)**

Port 5 Control Register, Low Byte (P5CONL)
FAH, Set 1, Bank 1, R/W

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

P5.0/SEG26

P5.1/SEG27

P5.2/SEG28

P5.3/SEG29

P5CONL bit-pair pin configuration settings:

| 00 | Input mode |
|----|------------|
| 01 | Output mode, N-channel open-drain |
| 10 | Output mode, push-pull |
| 11 | Alternative function (SEG29-SEG26) |

**Figure 9-17. Port 5 Low-Byte Control Register (P5CONL)**

Port 5 Pull-up Resistor Enable Register (P5PUR)
EFH, Set 1, Bank 1, R/W

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

P5.7  P5.6  P5.5  P5.4  P5.3  P5.2  P5.1  P5.0

P5PUR bit configuration settings:

| 0 | Pull-up Disable |
|---|-----------------|
| 1 | Pull-up Enable |
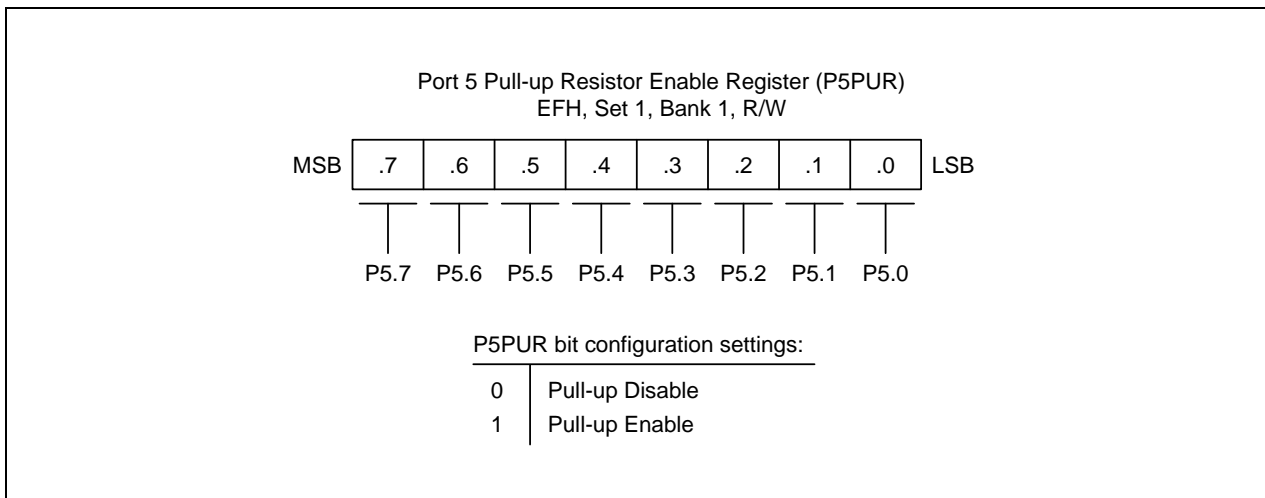
**Figure 9-18. Port 5 Pull-up Resistor Enable Register (P5PUR)**

SAMSUNG
ELECTRONICS

**PORT 6**

Port 6 is an 8-bit I/O port with individually configurable pins. Port 6 pins are accessed directly by writing or reading the port 5 data register, P6 at location F6H in set 1, bank 1. P6.0–P6.7 can serve as inputs (with without pull-ups), as push-pull outputs. And, they can serve as segment pins for LCD also.

**Port 6 Control Registers (P6CONH, P6CONL)**

Port 6 has two 8-bit control registers: P6CONH for P6.4–P6.7 and P6CONL for P6.0–P6.3. A reset clears the P6CONH and P6CONL registers to "00H", configuring all pins to input mode.

Port 6 Control Register, High Byte (P6CONH)
FBH, Set 1, Bank 1, R/W

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

P6.4/SEG14

P6.5/SEG15

P6.6/SEG16

P6.7/SEG17

P6CONH bit-pair pin configuration settings:

| | |
|---|---|
| 00 | Input mode |
| 01 | Input mode, pull-up |
| 10 | Output mode, push-pull |
| 11 | Alternative function (SEG17-SEG14) |
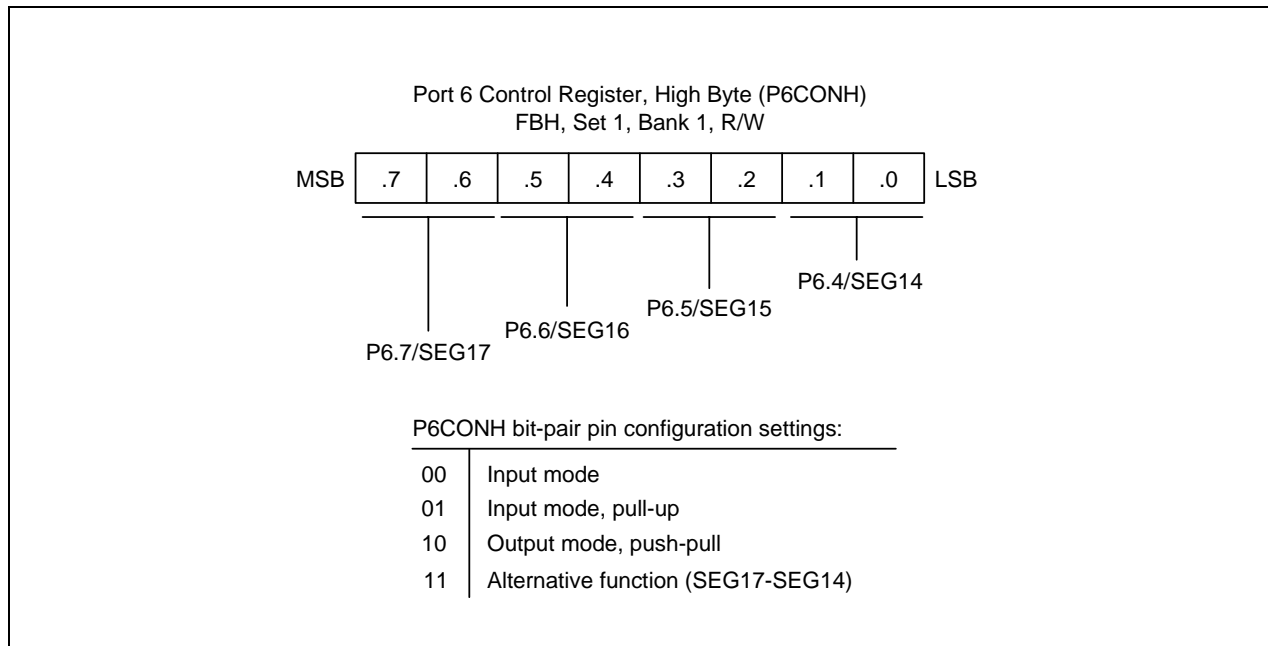
**Figure 9-19. Port 6 High-byte Control Register (P6CONH)**

Port 6 Control Register, Low Byte (P6CONL)
FCH, Set 1, Bank 1, R/W

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

P6.0/SEG10

P6.1/SEG11

P6.2/SEG12

P6.3/SEG13

P6CONL bit-pair pin configuration settings:

| 00 | Input mode |
| 01 | Input mode, pull-up |
| 10 | Output mode, push-pull |
| 11 | Alternative function (SEG13-SEG10) |

**Figure 9-20. Port 6 Low-byte Control Register (P6CONL)**

**PORT 7**

Port 7 is an 4-bit I/O port with individually configurable pins. Port 7 pins are accessed directly by writing or reading the port 7 data register, P7 at location F7H in set 1, bank 1. P7.0–P7.3 can serve as inputs (with without pull-ups), as push-pull outputs. And, they can serve as segment pins for LCD also.

**Port 7 Control Registers (P7CON)**

Port 7 has a 8-bit control registers: P7CON for P7.0–P7.3. A reset clears the P7CON register to "00H", configuring all pins to input mode.

Port 7 Control Register (P7CON)
FDH, Set 1, Bank 1, R/W

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

P7.0/SEG6

P7.1/SEG7

P7.2/SEG8

P7.3/SEG9

P7CONH bit-pair pin configuration settings:

| 00 | Input mode |
|----|------------|
| 01 | Input mode, pull-up |
| 10 | Output mode, push-pull |
| 11 | Alternative function (SEG9-SEG6) |
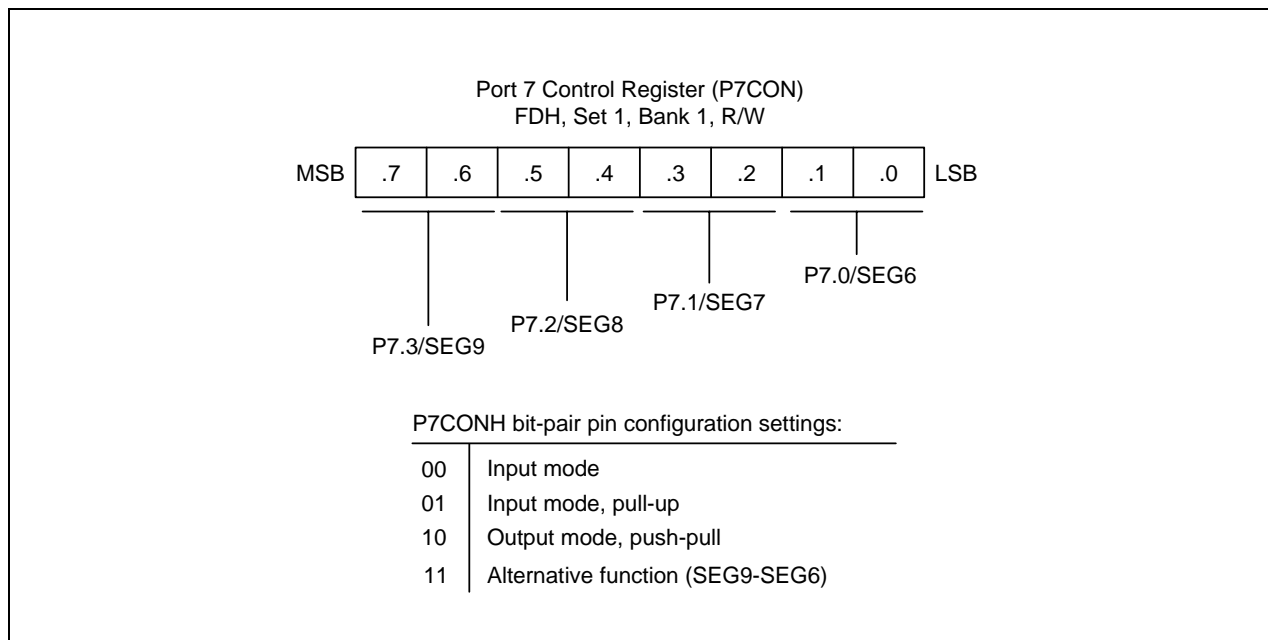
**Figure 9-21. Port 7 Control Register (P7CON)**

## PORT 8

Port 8 is an 8-bit I/O port with individually configurable pins. Port 8 pins are accessed directly by writing or reading the port 8 data register, P8 at location F8H in set 1, bank 1. P8.0–P8.7 can serve as inputs (with without pull-ups), as push-pull outputs. And, they can serve as segment pins for LCD also.

### Port 8 Control Registers (P8CON)

Port 8 has a 8-bit control registers: P8CON for P8.0–P8.7. A reset clears the P8CON register to "00H", configuring all pins to input mode.
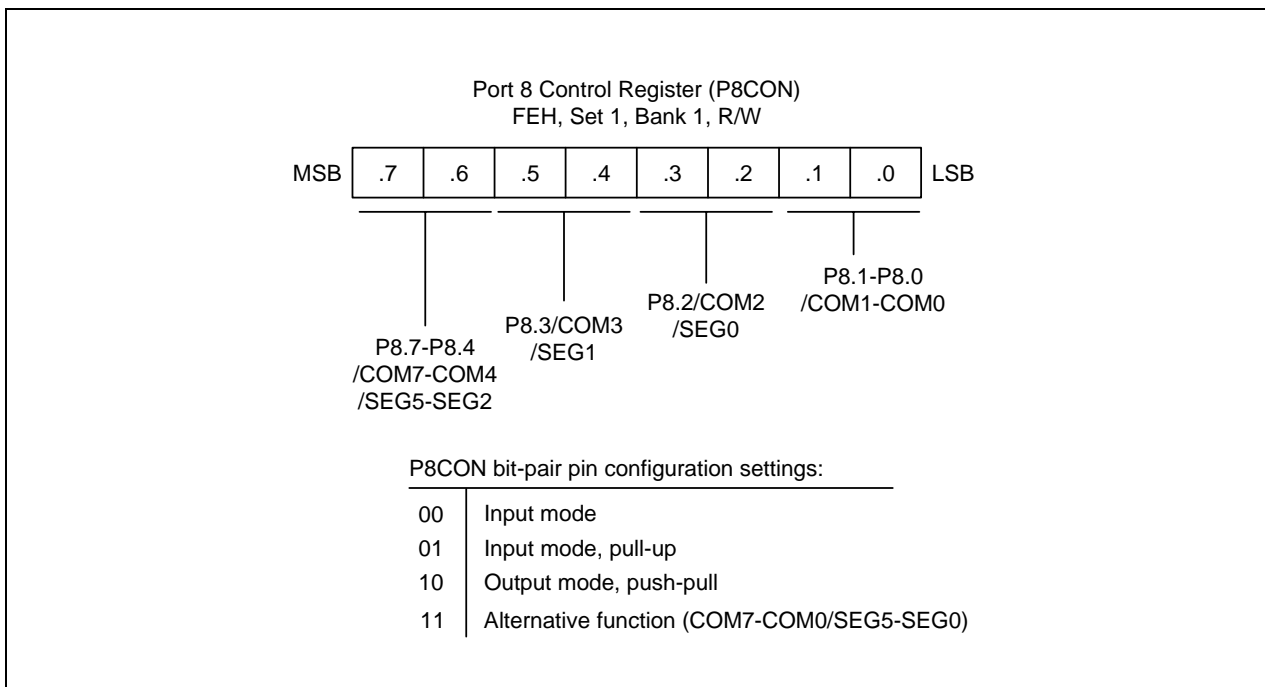


**Figure 9-22. Port 8 Control Register (P8CON)**

# 10 BASIC TIMER

## OVERVIEW

S3C828B/F828B/C8289/F8289/C8285/F8285 has an 8-bit basic timer .

### BASIC TIMER (BT)

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

— As a watchdog timer to provide an automatic reset mechanism in the event of a system malfunction, or

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

The functional components of the basic timer block are:

— Clock frequency divider (fxx divided by 4096, 1024, 128, or 16) with multiplexer

— 8-bit basic timer counter, BTCNT (set 1, Bank 0, FDH, read-only)

— Basic timer control register, BTCON (set 1, D3H, read/write)

### BASIC TIMER CONTROL REGISTER (BTCON)

The basic timer control register, BTCON, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function. It is located in set 1, address D3H, and is read/write addressable using Register addressing mode.

A reset clears BTCON to "00H". This enables the watchdog function and selects a basic timer clock frequency of fxx/4096. To disable the watchdog function, you must write the signature code "1010B" to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH), can be cleared at any time during the normal operation by writing a "1" to BTCON.1. To clear the frequency dividers, write a "1" to BTCON.0.

Basic TImer Control Register (BTCON)
D3H, Set 1, R/W

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

Watchdog timer enable bits:
1010B        = Disable watchdog function
Other value  = Enable watchdog function

Divider clear bit:
0 = No effect
1= Clear dvider

Basic timer counter clear bit:
0 = No effect
1= Clear BTCNT

Basic timer input clock selection bits:
00 = fxx/4096
01 = fxx/1024
10 = fxx/128
11 = fxx/16
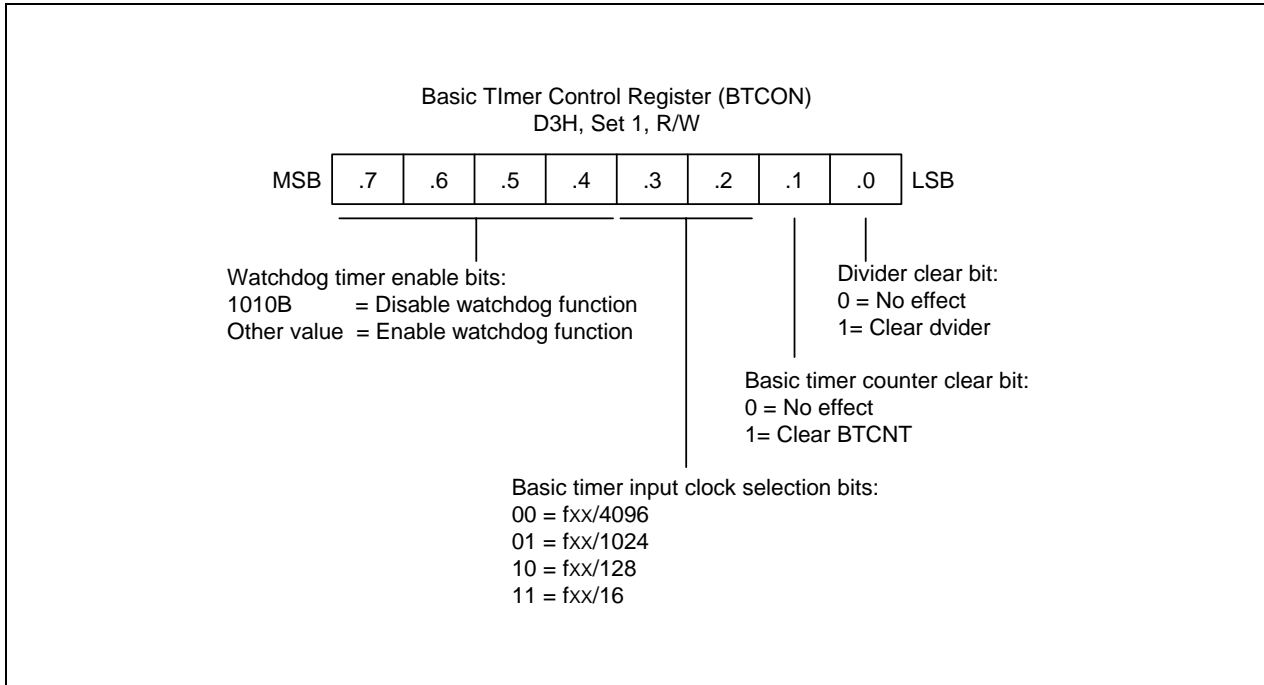
**Figure 10-1. Basic Timer Control Register (BTCON)**

SAMSUNG
ELECTRONICS

**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.

The MCU is resented 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 the 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 after a reset or when stop mode has been released by an external interrupt.

In stop mode, whenever a reset or an external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of fxx/4096 (for reset), or at the rate of the preset clock source (for 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 the normal operation.

In summary, the following events occur when stop mode is released:

1. During the stop mode, a power-on reset or 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 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, the normal CPU operation resumes.
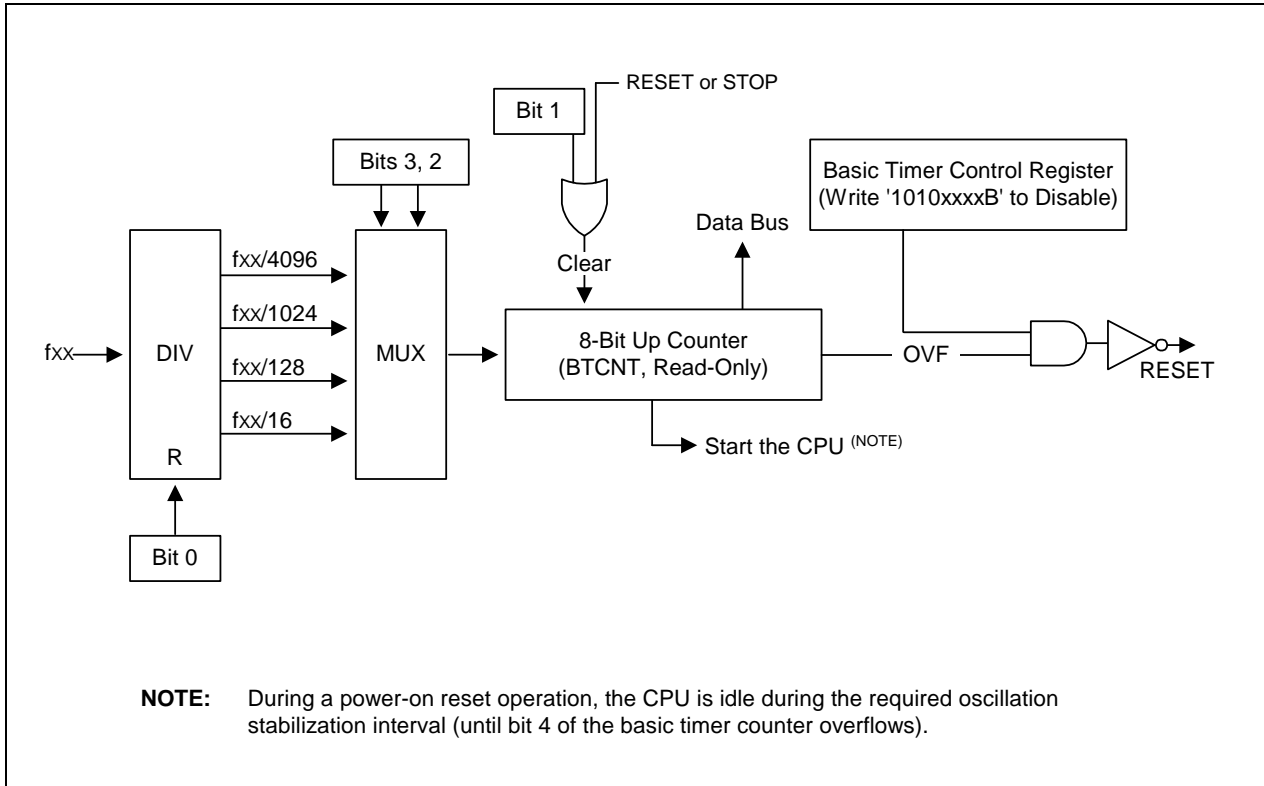
**Figure 10-2. Basic Timer Block Diagram**

# 11 8-BIT TIMER A/B

## 8-BIT TIMER A

### OVERVIEW

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

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

Timer A has the following functional components:

— Clock frequency divider (fxx divided by 1024, 256, 64, 8 or 1) with multiplexer
— External clock input pin (TACLK)
— 8-bit counter (TACNT), 8-bit comparator, and 8-bit reference data register (TADATA)
— I/O pins for capture input (TACAP) or PWM or match output (TAPWM, TAOUT)
— Timer A overflow interrupt (IRQ0, vector DEH) and match/capture interrupt (IRQ0, vector DCH) generation
— Timer A control register, TACON (set 1, Bank 0, E8H, read/write)

## TIMER A CONTROL REGISTER (TACON)

You use the timer A control register, TACON, to

— Select the timer A operating mode (interval timer, capture mode, or PWM mode)
— Select the timer A input clock frequency
— Clear the timer A counter, TACNT
— Enable the timer A overflow interrupt or timer A match/capture interrupt
— Clear timer A match/capture interrupt pending condition

TACON is located in set 1, Bank 0 at address E8H, and is read/write addressable using Register addressing mode.

A reset clears TACON to '00H'. This sets timer A to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all timer A interrupts. You can clear the timer A counter at any time during normal operation by writing a "1" to TACON.2.

The timer A overflow interrupt (TAOVF) is interrupt level IRQ0 and has the vector address DEH. When a timer A overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the timer A match/capture interrupt (IRQ0, vector DCH), you must write TACON.1 to "1". To detect a match/capture interrupt pending condition, the application program polls INTPND.1. When a "1" is detected, a timer A match or capture interrupt is pending. When the interrupt request has been serviced, the pending condition must be cleared by software by writing a "0" to the timer A match/capture interrupt pending bit, INTPND.1.

Timer A Control Register (TACON)
E8H, Set 1, Bank 0, R/W

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

Timer A input clock selection bits:
000 = fxx/1024
001 = fxx/256
010 = fxx/64
011 = fxx/8
100 = fxx
101 = External clock (TACLK) falling edge
110 = External clock (TACLK) rising edge
111 = Counter stop

Timer A operating mode selection bits:
00 = Interval mode (TAOUT)
01 = Capture mode (capture on rising edge,
       Counter running, OVF can occur)
10 = Capture mode (Capture on falling edge,
       Counter running, OVF can occur)
11 = PWM mode (OVF interrupt can occur)

Timer A overflow interrupt enable bit:
0 = Disable oveflow interrupt
1 = Enable overflow interrupt

Timer A match/capture interrupt enable bit:
0 = DIsable interrupt
1 = Enable interrupt

Timer A counter clear bit:
0 = No effect
1 = Clear the timer A counter (when write)

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

SAMSUNG
ELECTRONICS

## TIMER A FUNCTION DESCRIPTION

### Timer A Interrupts (IRQ0, Vectors DCH and DEH)

The timer A can generate two interrupts: the timer A overflow interrupt (TAOVF), and the timer A match/ capture interrupt (TAINT). TAOVF is interrupt level IRQ0, vector DEH. TAINT also belongs to interrupt level IRQ0, but is assigned the separate vector address, DCH.

A timer A overflow interrupt pending condition is automatically cleared by hardware when it has been serviced or should be cleared by software in the interrupt service routine by writing a "0" to the INTPND.0 interrupt pending bit. However, the timer A match/capture interrupt pending condition must be cleared by the application's interrupt service routine by writing a "0" to the INTPND.1 interrupt pending bit.

### Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer A reference data register, TADATA. The match signal generates a timer A match interrupt (TAINT, vector DCH) and clears the counter.

If, for example, you write the value "10H" to TADATA, the counter will increment until it reaches "10H". At this point, the timer A interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer A output pin is inverted (see Figure 11-2).
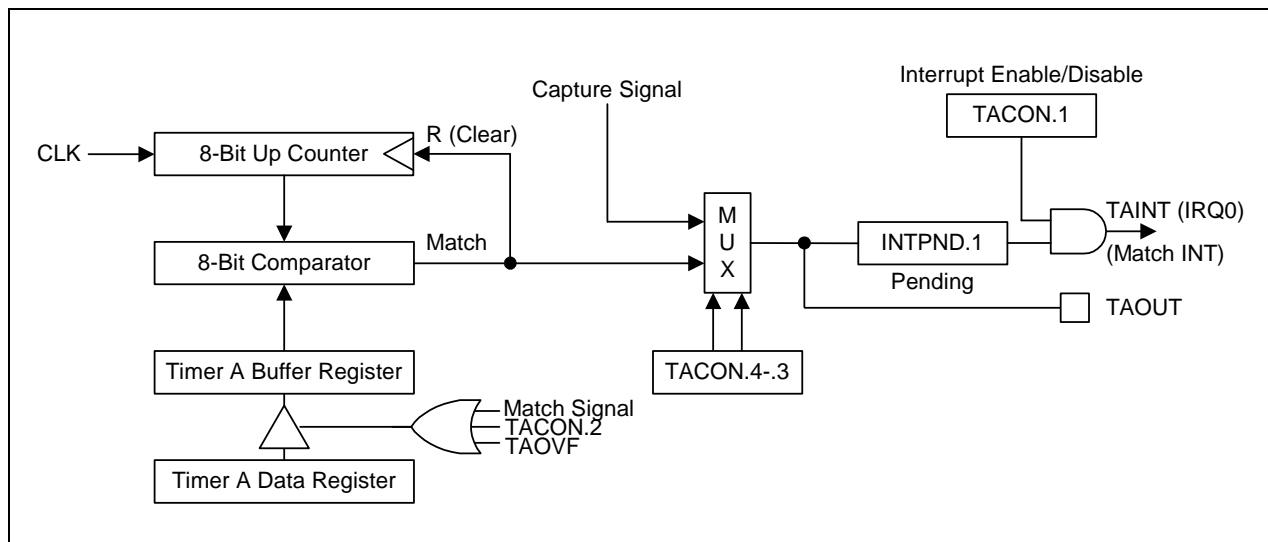


**Figure 11-2 Simplified Timer A Function Diagram: Interval Timer Mode**

**Pulse Width Modulation Mode**

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the TAPWM pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer A data register. In PWM mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at "FFH", and then continues incrementing from "00H".

Although you can use the match signal to generate a timer A overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the TAPWM pin is held to Low level as long as the reference data value is *less than or equal to* ( $\leq$ ) the counter value and then the pulse is held to High level for as long as the data value is *greater than* ( > ) the counter value. One pulse width is equal to $t_{CLK} \times 256$ (see Figure 11-3).
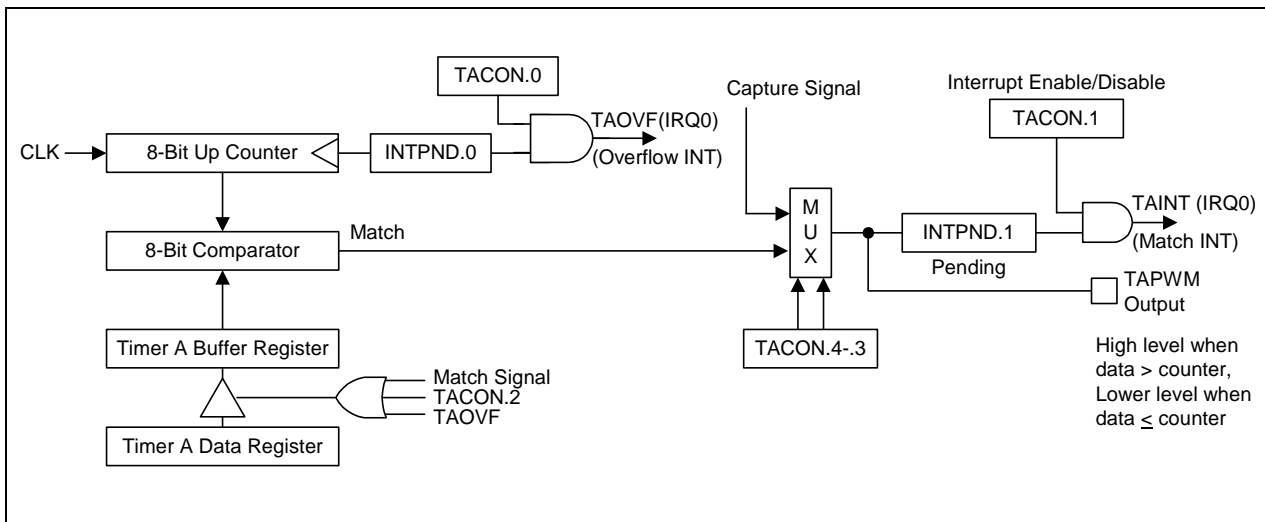


**Figure 11-3. Simplified Timer A Function Diagram: PWM Mode**

SAMSUNG
ELECTRONICS

## Capture Mode

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

Timer A also gives you capture input source: the signal edge at the TACAP pin. You select the capture input by setting the values of the timer A capture input selection bits in the port 3 control register, P3CONL.7–.6, (set 1, bank 1, EBH). When P3CONL.7–.6 is "00", the TACAP input is selected.

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

By reading the captured data value in TADATA, and assuming a specific value for the timer A clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the TACAP pin (see Figure 11-4).
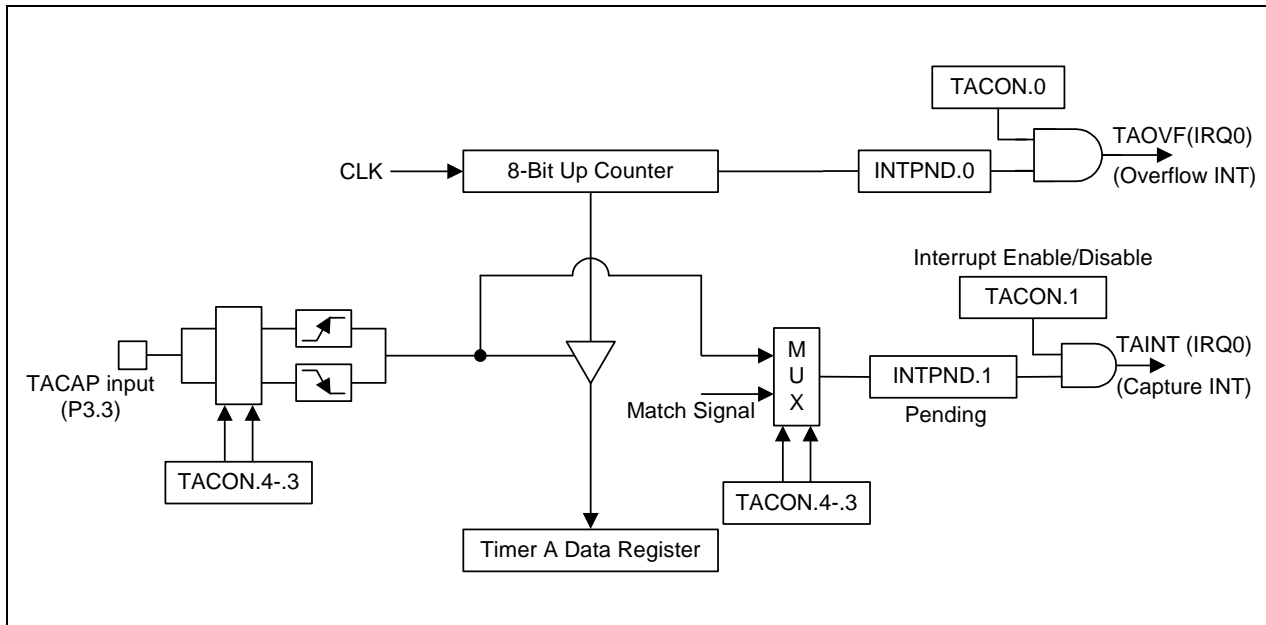


**Figure 11-4. Simplified Timer A Function Diagram: Capture Mode**
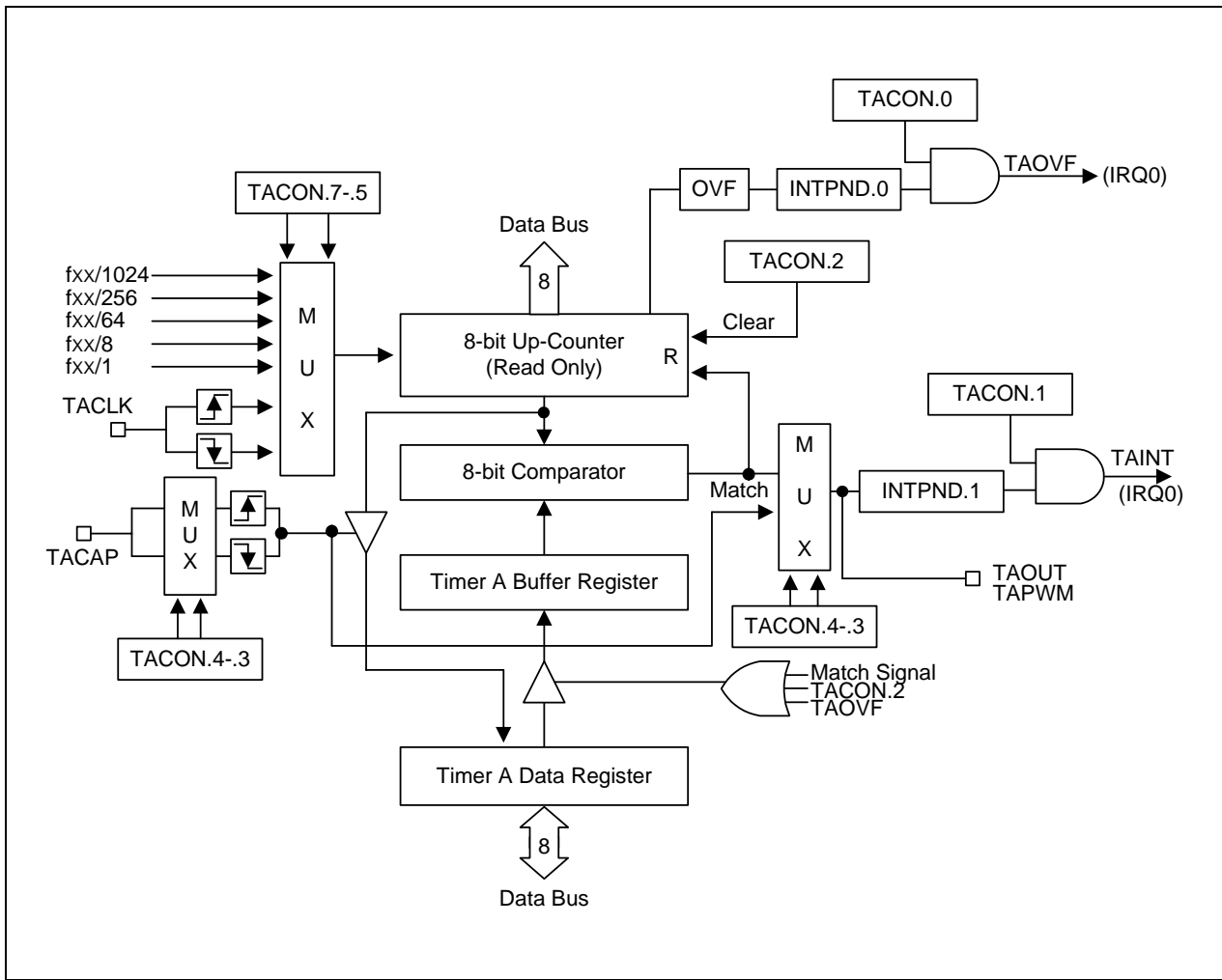
## BLOCK DIAGRAM



**Figure 11-5. Timer A Functional Block Diagram**

## 8-BIT TIMER B

**OVERVIEW**

The S3C828B/F828B/C8289/F8289/C8285/F8285 micro-controller has an 8-bit counter called timer B. Timer B, which can be used to generate the carrier frequency of a remote controller signal.
Pending condition of timer B is cleared automatically by hardware.

Timer B has two functions:

— As a normal interval timer, generating a timer B interrupt at programmed time intervals.

— To supply a clock source to the 8-bit timer/counter module, timer B, for generating the timer B overflow interrupt.

Timer B Control Register (TBCON)
F2H, Set 1, Bank 0,  R/W

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

Timer B input clock selection bits:
00 = fxx
01 = fxx/2
10 = fxx/4
11 = fxx/8

Timer B interrupt time selection bits:
00 = Elapsed time for low data value
01 = Elapsed time for high data value
10 = Elapsed time for low and high data values
11 = Not available

Timer B interrupt enable bit:
0 = Disable interrupt
1 = Enable interrupt

Timer B start/stop bit:
0 = Stop timer B
1 = Start timer B

Timer B mode selection bit:
0 = One-shot mode
1 = Repeating mode

Timer B output flip-flop control bit:
0 = TBOF is low
1 = TBOF is high

**Figure 11-6. Timer B Control Register**

## BLOCK DIAGRAM
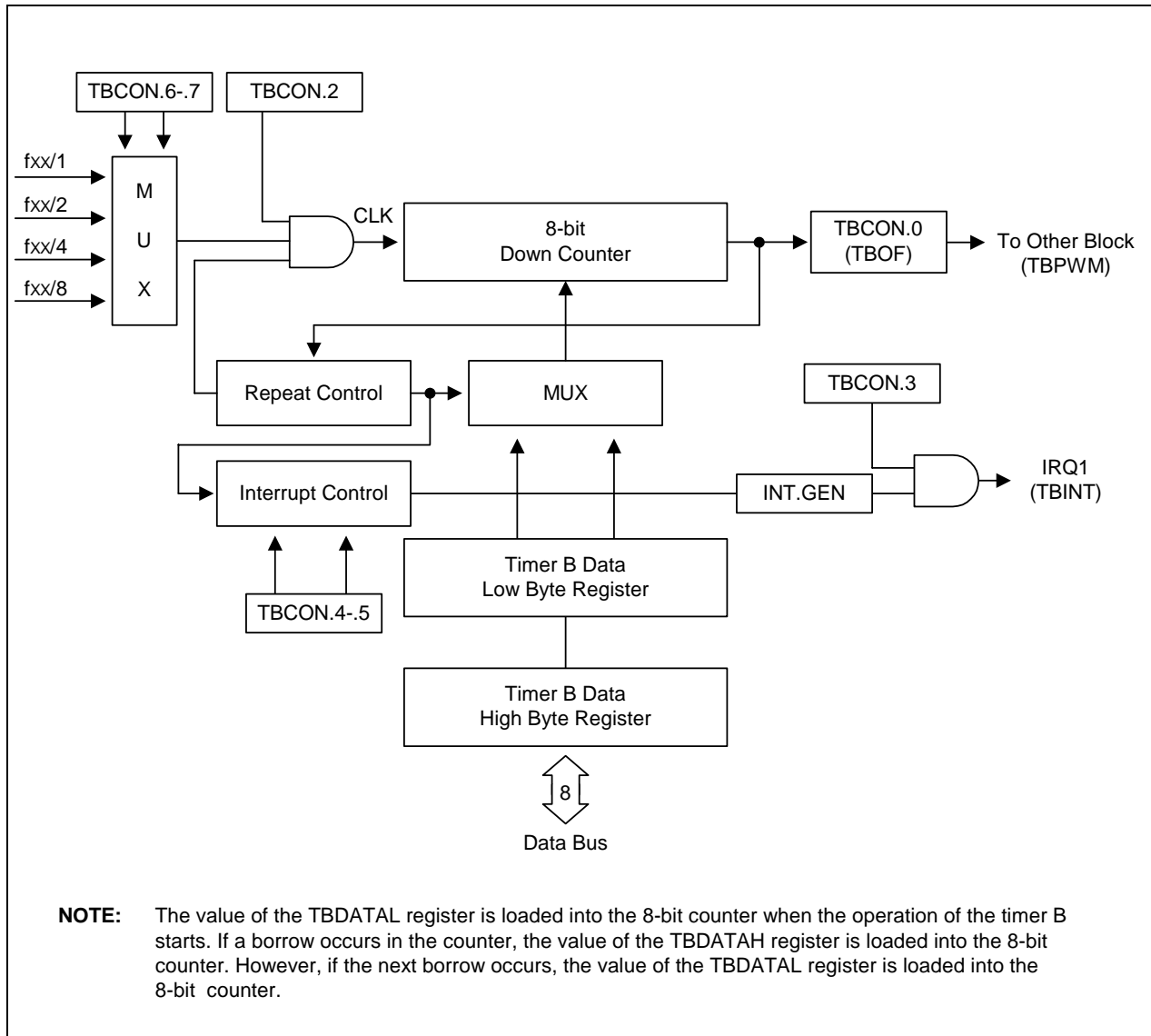


**Figure 11-7. Timer B Functional Block Diagram**

## TIMER B PULSE WIDTH CALCULATIONS
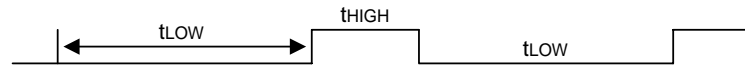


To generate the above repeated waveform consisted of low period time, $t_{LOW}$, and high period time, $t_{HIGH}$.

When TBOF = 0,

$t_{LOW}$ = (TBDATAL + 2) x 1/fx, 0H < TBDATAL < 100H, where fx = The selected clock.

$t_{HIGH}$ = (TBDATAH + 2) x 1/fx, 0H < TBDATAH < 100H, where fx = The selected clock.

When TBOF = 1,

$t_{LOW}$ = (TBDATAH + 2) x 1/fx, 0H < TBDATAH < 100H, where fx = The selected clock.

$t_{HIGH}$ = (TBDATAL + 2) x 1/fx, 0H < TBDATAL < 100H, where fx = The selected clock.

To make $t_{LOW}$ = 24 us and $t_{HIGH}$ = 15 us. $f_{OSC}$ = 4 MHz, fx = 4 MHz/4 = 1 MHz

When TBOF = 0,

$t_{LOW}$ = 24 us = (TBDATAL + 2) /fx = (TBDATAL + 2) x 1us, TBDATAL = 22.

$t_{HIGH}$ = 15 us = (TBDATAH + 2) /fx = (TBDATAH + 2) x 1us, TBDATAH = 13.

When TBOF = 1,

$t_{HIGH}$ = 15 us = (TBDATAL + 2) /fx = (TBDATAL + 2) x 1us, TBDATAL = 13.

$t_{LOW}$ = 24 us = (TBDATAH + 2) /fx = (TBDATAH + 2) x 1us, TBDATAH = 22.
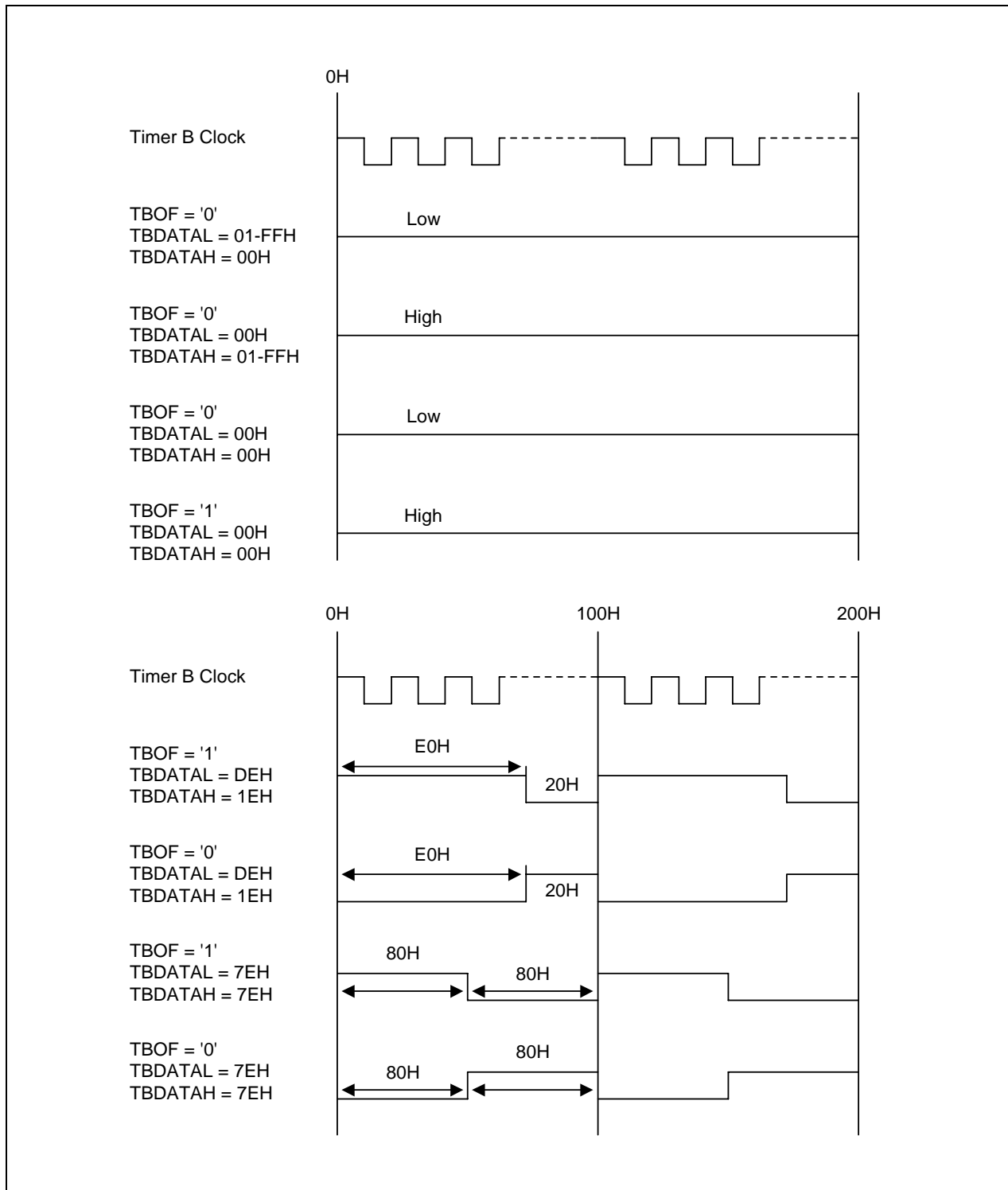
**Figure 11-8. Timer B Output Flip-Flop Waveforms in Repeat Mode**

☞  **PROGRAMMING TIP — To generate 38 kHz, 1/3duty signal through P3.0**

This example sets Timer B to the repeat mode, sets the oscillation frequency as the Timer B clock source, and TBDATAH and TBDATAL to make a 38 kHz, 1/3 Duty carrier frequency. The program parameters are:

```
        8.795 μs
                    ┌───── 17.59 μs ─────┐         ┌──────────┐          ┌──
────────┐           │                    │         │          │          │
        │           ←────────────────────→         │          │          │
        └───────────┘                    └─────────┘          └──────────┘
                    37.9 kHz 1/3 Duty
```
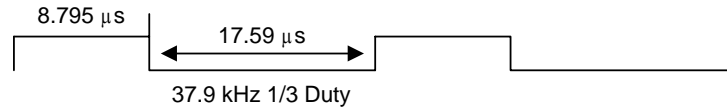
— Timer B is used in repeat mode

— Oscillation frequency is 4 MHz (0.25 μs)

— TBDATAH = 8.795 μs/0.25 μs = 35.18, TBDATAL = 17.59 μs/0.25 μs = 70.36

— Set P3.0 to TBPWM mode.


```
            ORG       0100H                   ;  Reset address
START       DI
            •
            •
            •
            LD        TBDATAL,#(70-2)         ;  Set 17.5 μs
            LD        TBDATAH,#(35-2)         ;  Set 8.75 μs
            LD        TBCON,#00000110B        ;  Clock Source  ←  fxx
                                              ;  Disable Timer B interrupt.
                                              ;  Select repeat mode for Timer B.
                                              ;  Start Timer B operation.
                                              ;  Set Timer B Output flip-flop (TBOF) high.

            LD        P3CONL,#02H             ;  Set P3.0 to TBPWM mode.
                                              ;  This command generates 38 kHz, 1/3 duty pulse signal
                                              ;     through P3.0.
            •
            •
            •
```

**SAMSUNG**
**ELECTRONICS**

**11-11**

☞ **PROGRAMMING TIP — To generate a one pulse signal through P3.0**

This example sets Timer B to the one shot mode, sets the oscillation frequency as the Timer B clock source, and TBDATAH and TBDATAL to make a 40μs width pulse. The program parameters are:

```
                         ┌──────────────┤  40 μs  ├──────────────┐
                         │◄─────────────────────────────────────►│
                                                                  └──────────
```
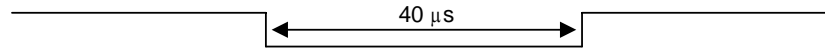
— Timer B is used in one shot mode

— Oscillation frequency is 4 MHz (1 clock = 0.25 μs)

— TBDATAH = 40 μs / 0.25 μs = 160, TBDATAL = 1

— Set P3.0 to TBPWM mode

```
          ORG       0100H                  ;  Reset address
START     DI
          •
          •
          •
          LD        TBDATAH,# (160-2)      ;  Set 40 μs
          LD        TBDATAL,# 1            ;  Set any value except   00H
          LD        TBCON,#00000001B       ;  Clock Source  ←  f_OSC
                                           ;  Disable Timer B interrupt.
                                           ;  Select one shot mode for Timer B.
                                           ;  Stop Timer B operation.
                                           ;  Set Timer B output flip-flop (TBOF) high
          LD        P3CONL, #02H           ;  Set P3.0 to TBPWM mode.
          •
          •
Pulse_out: LD       TBCON,#00000101B       ;  Start Timer B operation
                                           ;  to make the pulse at   this point.
          •                                ;  After the instruction is executed, 0.75 μs is required
          •                                ;  before the falling edge of the pulse starts.
          •
```

**SAMSUNG**
**ELECTRONICS**

# 12  16-BIT TIMER 0/1

## 16-BIT TIMER 0

### OVERVIEW

The 16-bit timer 0 is an 16-bit general-purpose timer. Timer 0 has the interval timer mode by using the appropriate T0CON setting.

Timer 0 has the following functional components:

— Clock frequency divider (fxx divided by 256, 64, 8, or 1) with multiplexer

— TBOF (from timer B) is one of the clock frequencies.

— 16-bit counter (T0CNTH/L), 16-bit comparator, and 16-bit reference data register (T0DATAH/L)

— Timer 0 interrupt (IRQ2, vector E2H) generation

— Timer 0 control register, T0CON (set 1, Bank 0, E3H, read/write)

### FUNCTION DESCRIPTION

**Interval Timer Function**

The timer 0 module can generate an interrupt, the timer 0 match interrupt (T0INT). T0INT belongs to interrupt level IRQ2, and is assigned the separate vector address, E2H.

The T0INT pending condition is automatically cleared by hardware when it has been serviced. Even though T0INT is disabled, the application's service routine can detect a pending condition of T0INT by the software and execute it's sub-routine. When this case is used, the T0INT pending bit must be cleared by the application subroutine by writing a "0" to the T0CON.0 pending bit.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the T0 reference data registers, T0DATAH/L. The match signal generates a timer 0 match interrupt (T0INT, vector E2H) and clears the counter.

If, for example, you write the value 0010H to T0DATAH/L and 0FH to T0CON, the counter will increment until it reaches 10H. At this point, the T0 interrupt request is generated, the counter value is reset, and counting resumes.

**SAMSUNG**
**ELECTRONICS**

## TIMER 0 CONTROL REGISTER (T0CON)

You use the timer 0 control register, T0CON, to

— Enable the timer 0 operating (interval timer)

— Select the timer 0 input clock frequency

— Clear the timer 0 counter, T0CNT

— Enable the timer 0 interrupt and clear timer 0 interrupt pending condition

T0CON is located in set 1, bank 0, at address E3H, and is read/write addressable using register addressing mode.

A reset clears T0CON to "00H". This sets timer 0 to disable interval timer mode, selects the TBOF, and disables timer 0 interrupt. You can clear the timer 0 counter at any time during normal operation by writing a "1" to T0CON.3

To enable the timer 0 interrupt (IRQ2, vector E2H), you must write T0CON.2, and T0CON.1 to "1". To generate the exact time interval, you should write T0CON.3 and 0, which cleared counter and interrupt pending bit. To detect an interrupt pending condition when T0INT is disabled, the application program polls pending bit, T0CON.0. When a "1" is detected, a timer 0 interrupt is pending. When the T0INT sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 0 interrupt pending bit, T0CON.0.

Timer 0 Control Registers (T0CON)
E3H, Set 1, Bank 0,  R/W

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

Timer 0 input clock selection bits:
000 = TBOF
001 = fxx/256
010 = fxx/64
011 = fxx/8
1xx = fxx

Timer 0 interrupt pending bit:
0 = No interrupt pending
0 = Clear pending bit (when write)
1 = Interrupt is pending

Timer 0 interrupt enable bit:
0 = Disable interrupt
1 = Enable interrupt

Not used

Timer 0 count enable bit:
0 = Disable counting operation
1 = Enable counting operation

Timer 0 counter clear bit:
0 = No affect
1 = Clear the timer 0 counter (when write)

**Figure 12-1. Timer 0 Control Register (T0CON)**

SAMSUNG
ELECTRONICS
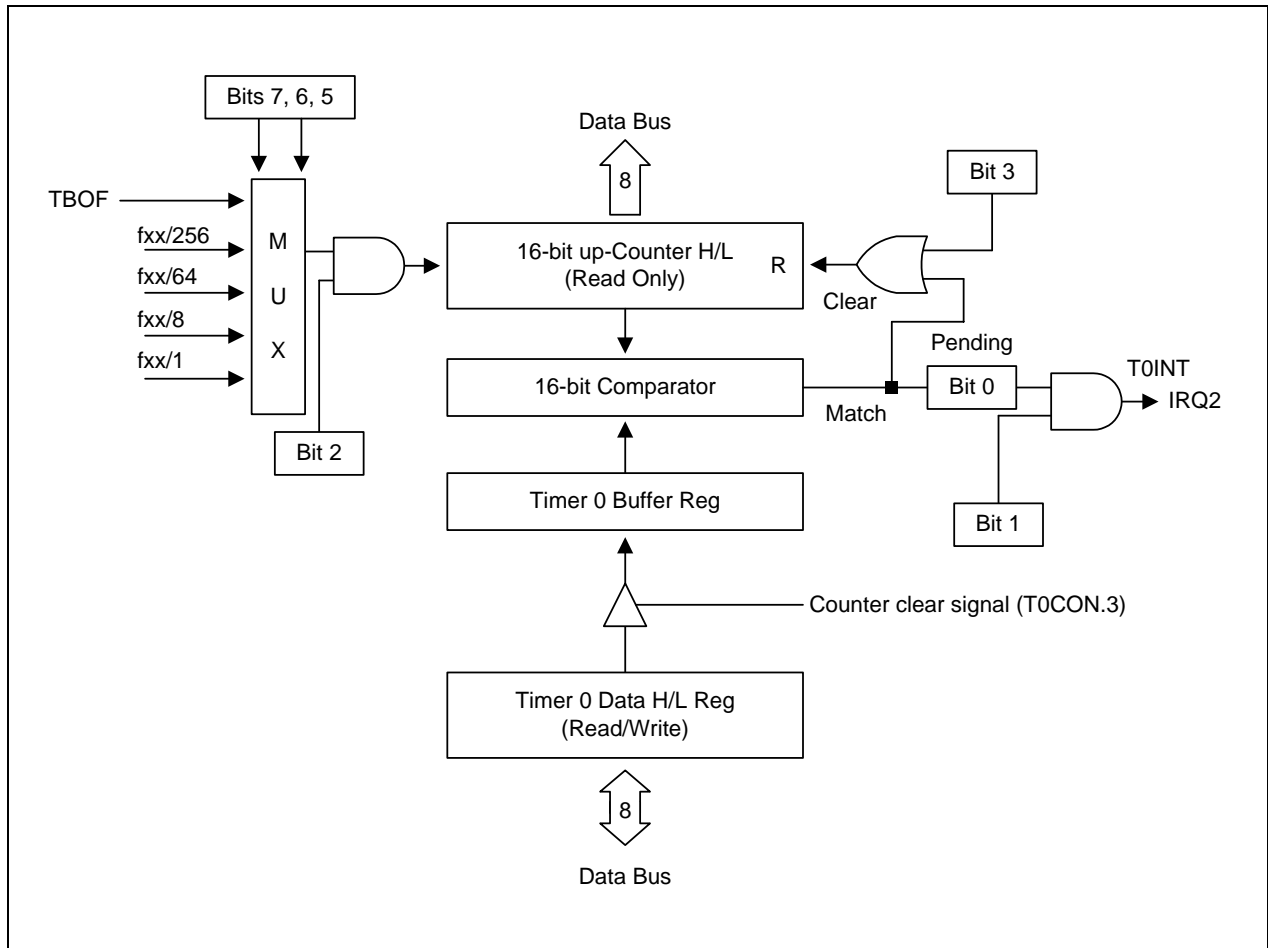
BLOCK DIAGRAM



**Figure 12-2. Timer 0 Functional Block Diagram**

## 16-BIT TIMER 1

**OVERVIEW**

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

— Interval timer mode (Toggle output at T1OUT pin)

— Capture input mode with a rising or falling edge trigger at the T1CAP pin

— PWM mode (T1PWM)

Timer 1 has the following functional components:

— Clock frequency divider ($f_{xx}$ divided by 1024, 256, 64, 8, or 1) with multiplexer

— External clock input pin (T1CLK)

— 16-bit counter (T1CNTH/L), 16-bit comparator, and 16-bit reference data register (T1DATAH/L)

— I/O pins for capture input (T1CAP), or PWM or match output (T1PWM, T1OUT)

— Timer 1 overflow interrupt (IRQ3, vector E6H) and match/capture interrupt (IRQ3, vector E4H) generation

— Timer 1 control register, T1CON (set 1, Bank 0, EBH, read/write)

**TIMER 1 CONTROL REGISTER (T1CON)**

You use the timer 1 control register, T1CON, to

— Select the timer 1 operating mode (interval timer, capture mode, or PWM mode)

— Select the timer 1 input clock frequency

— Clear the timer 1 counter, T1CNTH/T1CNTL

— Enable the timer 1 overflow interrupt or timer 1 match/capture interrupt

— Clear timer 1 match/capture interrupt pending conditions

T1CON is located in set 1 and Bank 0 at address EBH, and is read/write addressable using Register addressing mode.

A reset clears T1CON to '00H'. This sets timer 1 to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all timer 1 interrupts. To disable the counter operation, please set T1CON.7-.5 to 111B. You can clear the timer 1 counter at any time during normal operation by writing a "1" to T1CON.3.
The timer 1 overflow interrupt (T1OVF) is interrupt level IRQ3 and has the vector address E6H. When a timer 1 overflow interrupt occurs and is serviced interrupt (IRQ3, vector E4H), you must write T1CON.1 to "1". To generate the exact time interval, you should write T1CON by the CPU, the pending condition is cleared automatically by hardware.

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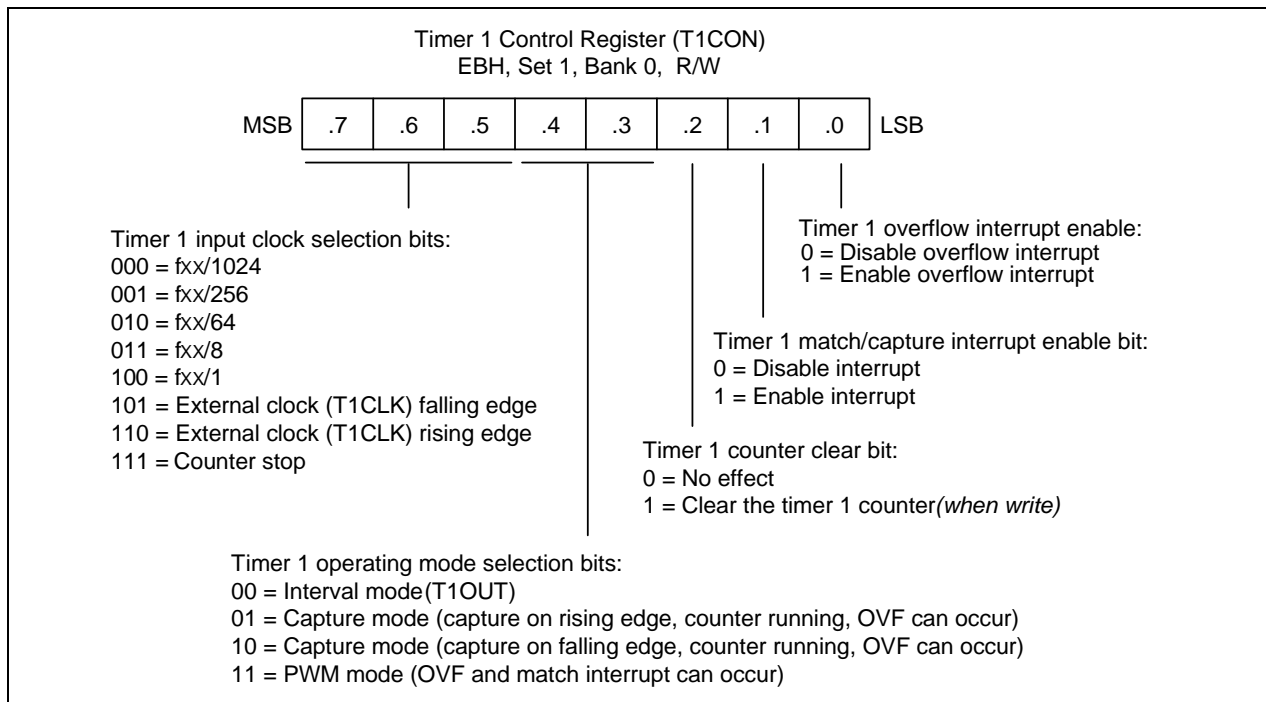


**Figure 12-3. Timer 1 Control Register (T1CON)**

## TIMER 1 FUNCTION DESCRIPTION

### Timer 1 Interrupts (IRQ2, Vectors E4H and E6H)

The timer 1 can generate two interrupts: the timer 1 overflow interrupt (T1OVF), and the timer 1 match/ capture interrupt (T3INT). T3OVF is belongs to interrupt level IRQ3, vector E6H. T1INT also belongs to interrupt level IRQ3, but is assigned the separate vector address, E4H.

A timer 1 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced or should be cleared by software in the interrupt service routine by writing a "0" to the INTPND.2 interrupt pending bit. However, the timer 1 match/capture interrupt pending condition must be cleared by the application's interrupt service routine by writing a "0" to the INTPND.3 interrupt pending bit.

### Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 1 reference data register, T1DATAH/T1DATAL. The match signal generates a timer 1 match interrupt (T1INT, vector E4H) and clears the counter.

If, for example, you write the value "1087H" to T1DATAH/T1DATAL, the counter will increment until it reaches "1087H". At this point, the timer 1 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer 1 output pin is inverted (see Figure 12-4).
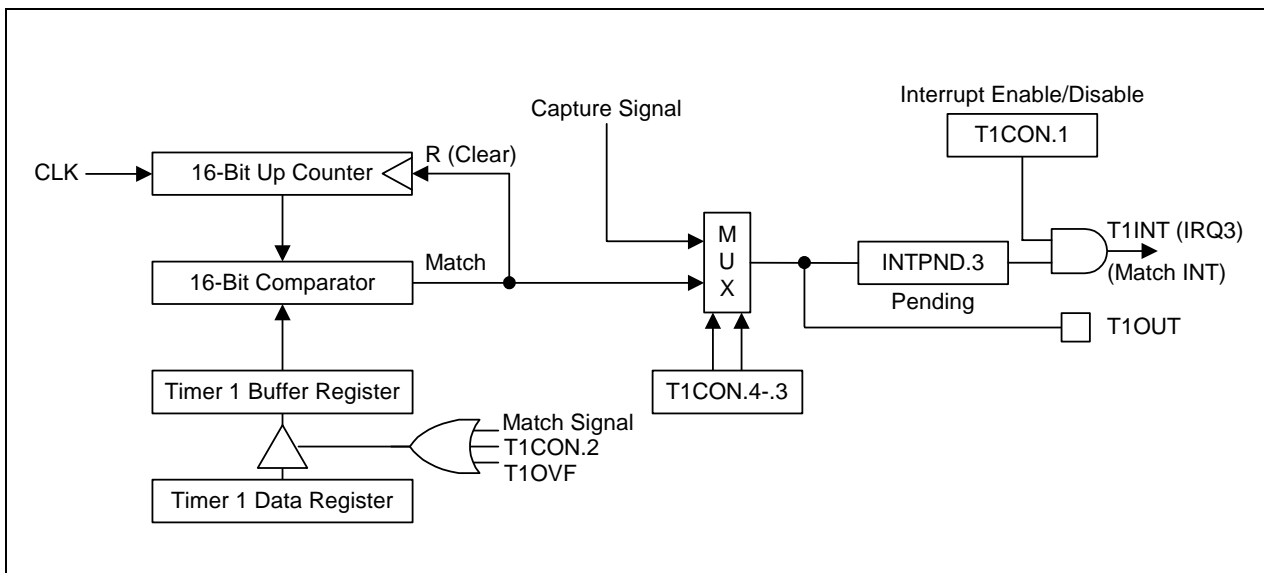


**Figure 12-4. Simplified Timer 1 Function Diagram: Interval Timer Mode**

**Pulse Width Modulation Mode**

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the T1PWM pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 1 data register. In PWM mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at "FFFFH", and then continues incrementing from "0000H".

Although you can use the match signal to generate a timer 1 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the T1PWM pin is held to Low level as long as the reference data value is *less than or equal to* ( $\leq$ ) the counter value and then the pulse is held to High level for as long as the data value is *greater than* ( > ) the counter value. One pulse width is equal to $t_{CLK} \times 65536$ (see Figure 12-5).
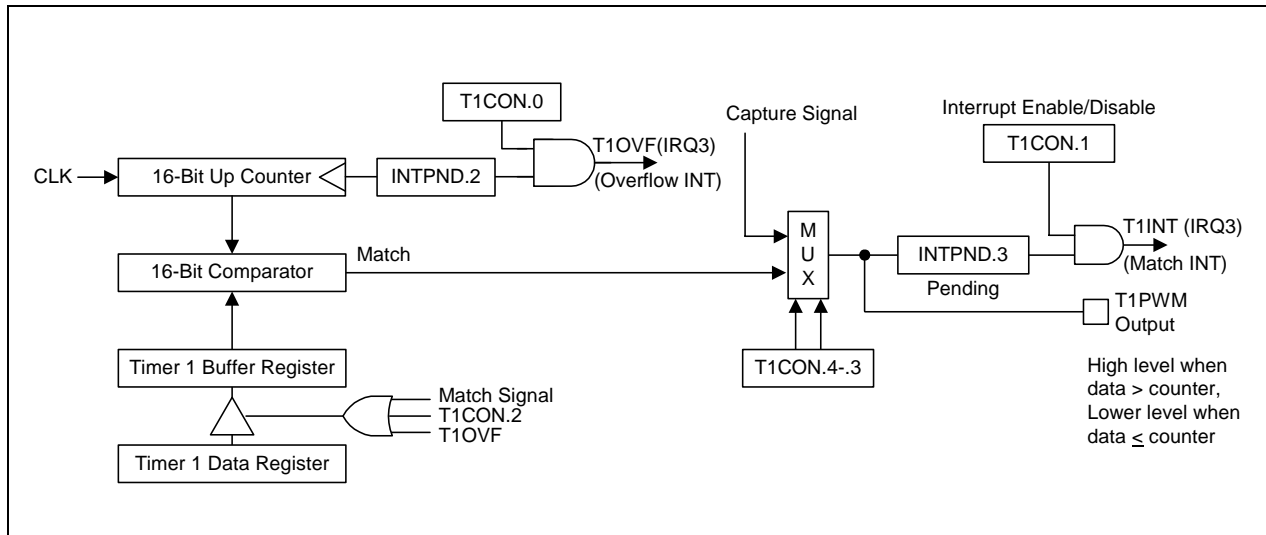


**Figure 12-5. Simplified Timer 1 Function Diagram: PWM Mode**

**Capture Mode**

In capture mode, a signal edge that is detected at the T1CAP pin opens a gate and loads the current counter value into the timer 1 data register. You can select rising or falling edges to trigger this operation.

Timer 1 also gives you capture input source: the signal edge at the T1CAP pin. You select the capture input by setting the values of the timer 1 capture input selection bits in the port 1 control register, P1CONH.1–.0, (set 1, bank 1, E6H). When P1CONH.1–.0 is "00", the T1CAP input is selected.

Both kinds of timer 1 interrupts can be used in capture mode: the timer 1 overflow interrupt is generated whenever a counter overflow occurs; the timer 1 match/capture interrupt is generated whenever the counter value is loaded into the timer 1 data register.

By reading the captured data value in T1DATAH/T1DATAL, and assuming a specific value for the timer 1 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the T1CAP pin (see Figure 12-6).
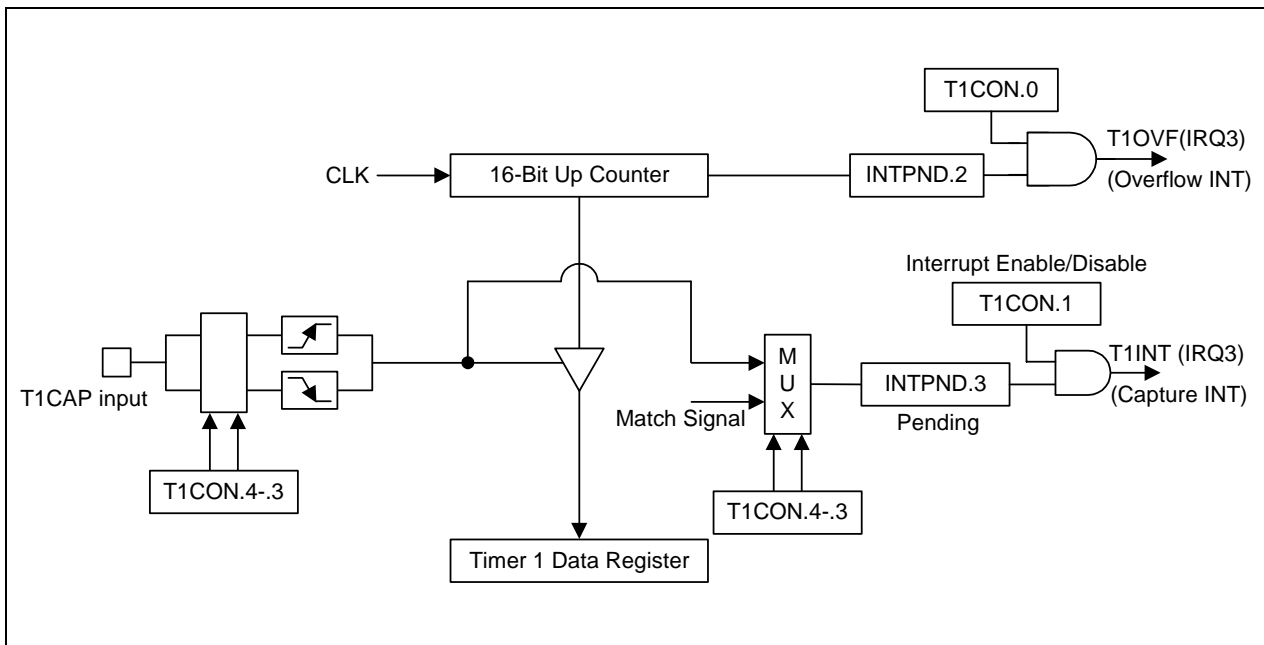


**Figure 12-6. Simplified Timer 1 Function Diagram: Capture Mode**
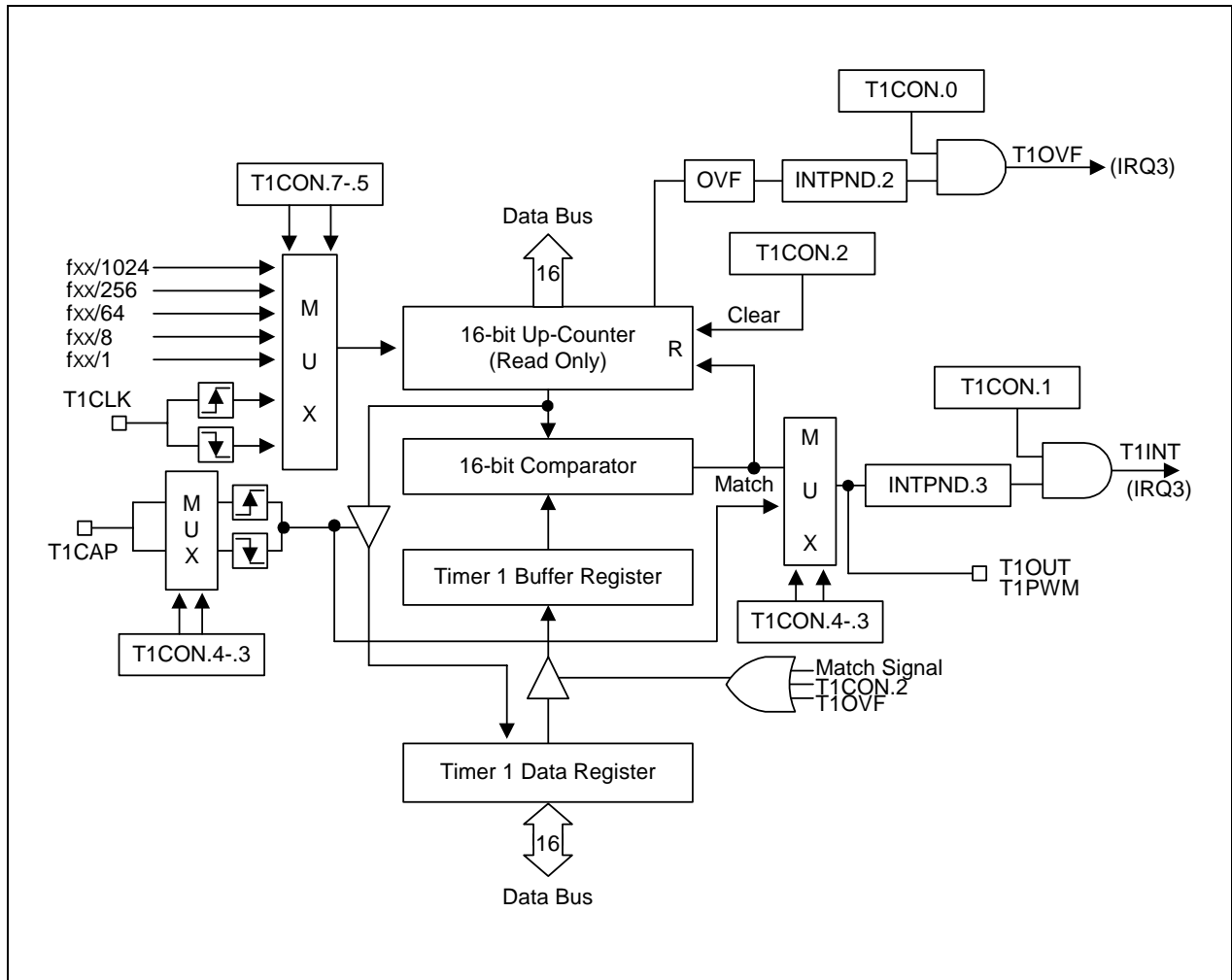
**BLOCK DIAGRAM**



**Figure 12-7. Timer 1 Functional Block Diagram**

# 13 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 (IRQ5, vector EEH), then set the WTCON.6 to "1".
The watch timer overflow interrupt pending condition (WTCON.0) must be cleared by software in the application's interrupt service routine by means of writing a "0" to the WTCON.0 interrupt pending bit.
After the watch timer starts and elapses a time, the watch timer interrupt pending bit (WTCON.0) is automatically set to "1", and interrupt requests commence in 3.91 ms, 0.25, 0.5 and 1-second intervals by setting Watch timer speed selection bits (WTCON.3–.2).

The watch timer can generate a steady 0.5 kHz, 1 kHz, 2 kHz, or 4 kHz signal to BUZ output pin for Buzzer. By setting WTCON.3 and WTCON.2 to "11b", the watch timer will function in high-speed mode, generating an interrupt every 3.91 ms. High-speed mode is useful for timing events for program debugging sequences.

The watch timer supplies the clock frequency for the LCD controller ($f_{LCD}$). Therefore, if the watch timer is disabled, the LCD controller does not operate.

Watch timer has the following functional components:

— Real Time and Watch-Time Measurement

— Using a Main Clock Source or Sub clock

— Clock Source Generation for LCD Controller ($f_{LCD}$)

— I/O pin for Buzzer Output Frequency Generator (BUZ)

— Timing Tests in High-Speed Mode

— Watch timer overflow interrupt (IRQ5, vector EEH) generation

— Watch timer control register, WTCON (set 1, bank 0, D1H, read/write)

**SAMSUNG**
**ELECTRONICS**

13-1

**WATCH TIMER CONTROL REGISTER (WTCON)**

The watch timer control register, WTCON is used to select the watch timer interrupt time and Buzzer signal, to enable or disable the watch timer function. It is located in set 1, bank 0 at address D1H, and is read/write addressable using register addressing mode.
A reset clears WTCON to "00H". This disable the watch timer.
So, if you want to use the watch timer, you must write appropriate value to WTCON.



Watch Timer Control Register (WTCON)
D1H, Set 1, Bank 0,  R/W

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

Watch timer clock selection bit:
0 = Select main clock divided by $2^7$ (fx/128)
1 = Select sub clock (fxt)

Watch timer INT Enable/Disable bit:
0 = Disable watch timer INT
1 = Enable watch timer INT

Buzzer signal selection bits:
00 = 0.5 kHz
01 = 1 kHz
10 = 2 kHz
11 = 4 kHz

Watch timer interrupt pending bit:
0 = Interrupt request is not pending
     (Clear pending bit when write"0")
1 = Interrupt request is pending

Watch timer Enable/Disable bit:
0 = Disable watch timer
1 = Enable watch timer

Watch timer speed selection bits:
00 = Set watch timer interrupt to 1 s
01 = Set watch timer interrupt to 0.5 s
10 = Set watch timer interrupt to 0.25 s
11 = Set watch timer interrupt to 3.91 ms

**Figure 13-1. Watch Timer Control Register (WTCON)**

## WATCH TIMER CIRCUIT DIAGRAM
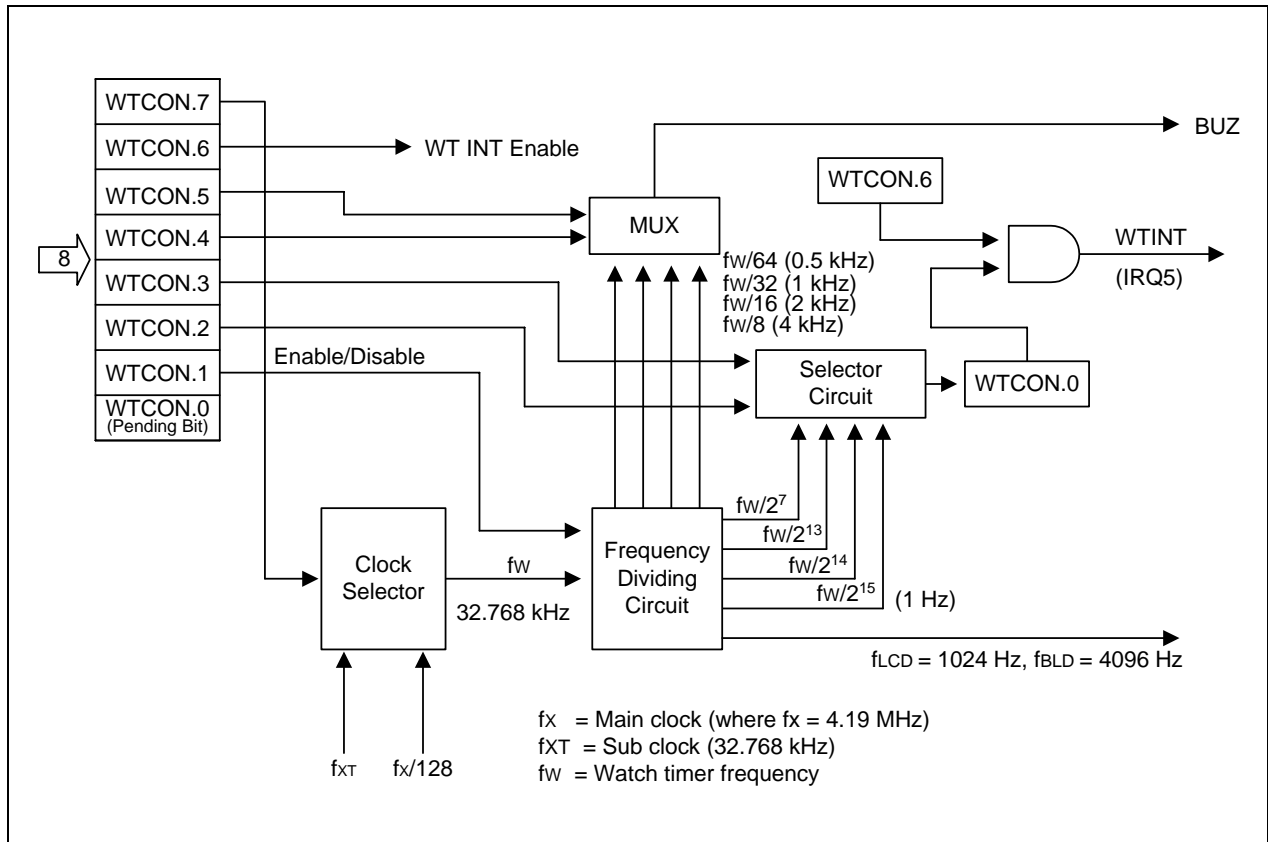


**Figure 13-2. Watch Timer Circuit Diagram**

# 14 LCD CONTROLLER/DRIVER

## OVERVIEW

The S3C828B/F828B/C8289/F8289/C8285/F8285 microcontroller can directly drive an up-to-256-dot (32 segments x 8 commons) LCD panel. Its LCD block has the following components:

— LCD controller/driver

— Display RAM for storing display data

— 6 common/segment output pins (COM2/SEG0–COM7/SEG5)

— 32 segment output pins (SEG6–SEG37)

— 2 common output pins (COM0–COM1)

— Four LCD operating power supply pins ($V_{LC0}$–$V_{LC3}$)

— LCD bias by internal/external register

The LCD control register, LCON, is used to turn the LCD display on and off, switch the current to the dividing resistors for the LCD display, and frame frequency. Data written to the LCD display RAM can be automatically transferred to the segment signal pins without any program control.

When a subsystem clock is selected as the LCD clock source, the LCD display is enabled even in the main clock stop or idle mode.
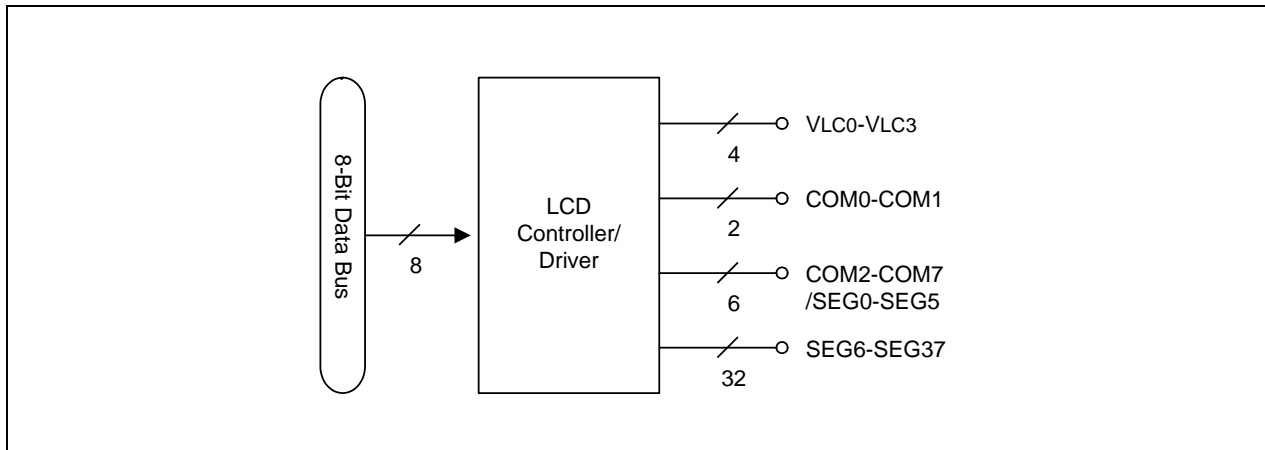


**Figure 14-1. LCD Function Diagram**

## LCD CIRCUIT DIAGRAM
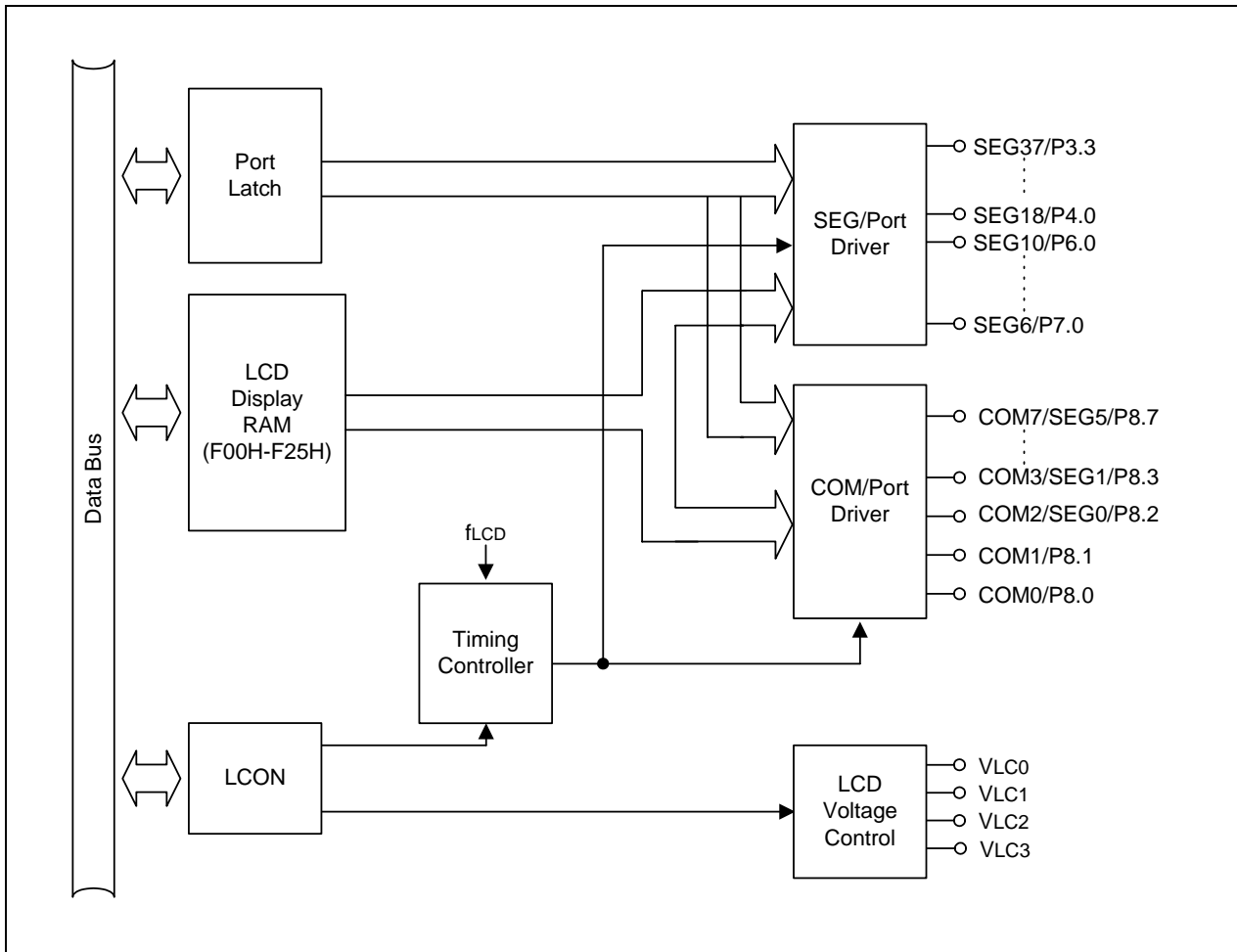


**Figure 14-2. LCD Circuit Diagram**

**LCD RAM ADDRESS AREA**

RAM addresses of page 15 are used as LCD data memory. These locations can be addressed by 1-bit or 8-bit instructions. If the bit value of a display segment is "1", the LCD display is turned on. If the bit value is "0", the display is turned off.

Display RAM data are sent out through the segment pins, SEG0–SEG37, using the direct memory access (DMA) method that is synchronized with the $f_{LCD}$ signal. RAM addresses in this location that are not used for LCD display can be allocated to general-purpose use.

| COM | Bit | SEG0 | SEG1 | SEG2 | SEG3 | SEG4 | ------ | SEG36 | SEG37 |
|---|---|---|---|---|---|---|---|---|---|
| COM0<br>COM1<br>COM2<br>COM3<br>COM4<br>COM5<br>COM6<br>COM7 | .0<br>.1<br>.2<br>.3<br>.4<br>.5<br>.6<br>.7 | F00H | F01H | F02H | F03H | F04H | ------ | F24H | F25H |

**Figure 14-3. LCD Display Data RAM Organization**

## LCD CONTROL REGISTER (LCON)

A LCON is located in page 15 of set1, bank0 at address D0H, 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. A reset clears the LCON registers to "00H", configuring turns off the LCD display, select 1/8 duty and 1/4 bias, select 128Hz 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)
D0H, Set 1, Bank 0, R/W

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

Internal LCD dividing register enable bit:
0 = Enable internal LCD dividing resistors
1 = Disable internal LCD dividing resistors

LCD display control bit:
0 = All LCD signals are low
(Turn off the P-Tr)
1 = Turn display on
(Turn on the P-Tr)

LCD clock selection bits:
00 = fw/$2^8$ (128 Hz)
01 = fw/$2^7$ (256 Hz)
10 = fw/$2^6$ (512 Hz)
11 = fw/$2^5$ (1024 Hz)

Not used

LCD duty and bias selection bits:
000 = 1/8 duty, 1/4 bias
001 = 1/4 duty, 1/3 bias
010 = 1/3 duty, 1/3 bias
011 = 1/3 duty, 1/2 bias
1xx = 1/2 duty, 1/2 bias

NOTES:
1.  "x" means don't care.
2.  When 1/3 bias is selected, the bias levels are set as VLC0, VLC1, VLC2(VLC3), and VSS.
3.  When 1/2 bias is selected, the bias levels are set as VLC0, VLC1(VLC2, VLC3), and VSS.

**Figure 14-4. LCD Control Register (LCON)**

## LCD VOLTAGE DIVIDING RESISTOR
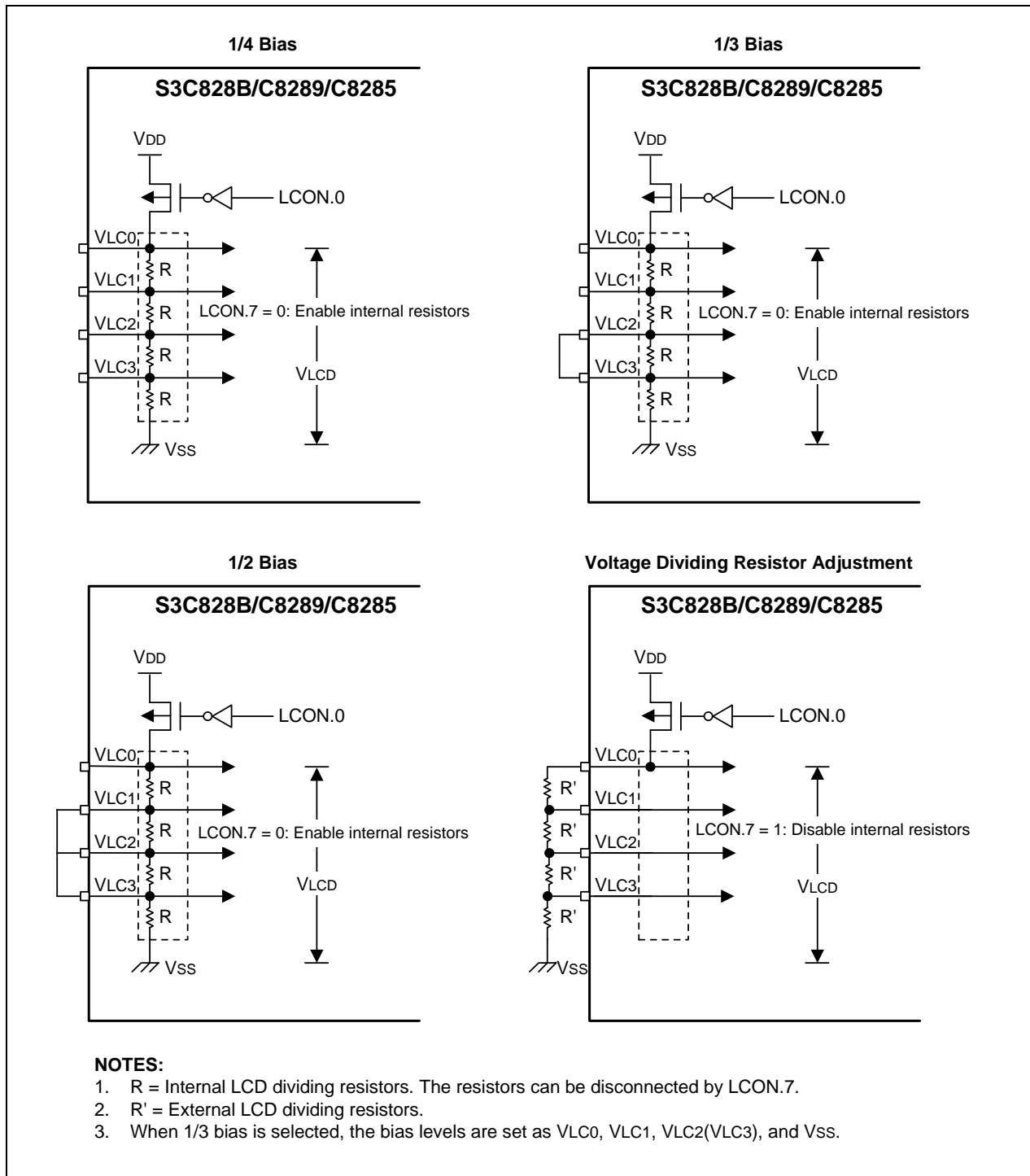


**Figure 14-5. LCD Voltage Dividing Resistor Connection**

NOTES:
1.  R = Internal LCD dividing resistors. The resistors can be disconnected by LCON.7.
2.  R' = External LCD dividing resistors.
3.  When 1/3 bias is selected, the bias levels are set as VLC0, VLC1, VLC2(VLC3), and VSS.

## COMMON (COM) SIGNALS

The common signal output pin selection (COM pin selection) varies according to the selected duty cycle.

— In 1/8 duty mode, COM0-COM7 (SEG6–SEG37) pins are selected.

— In 1/4 duty mode, COM0-COM3 (SEG2–SEG37) pins are selected.

— In 1/3 duty mode, COM0-COM2 (SEG1–SEG37) pins are selected.

— In 1/2 duty mode, COM0-COM1 (SEG0–SEG37) pins are selected.

## SEGMENT (SEG) SIGNALS

The 38 LCD segment signal pins are connected to corresponding display RAM locations at page 15. 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.
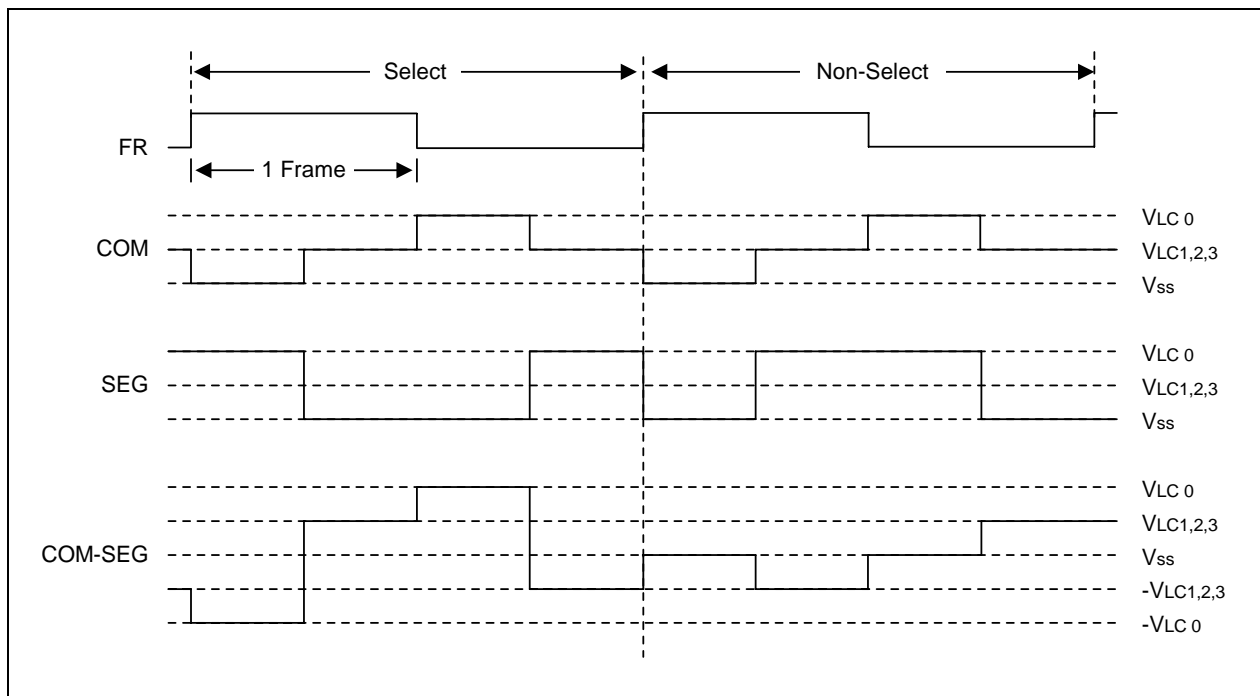
**Figure 14-6. Select/No-Select Signal in 1/2 Duty, 1/2 Bias Display Mode**
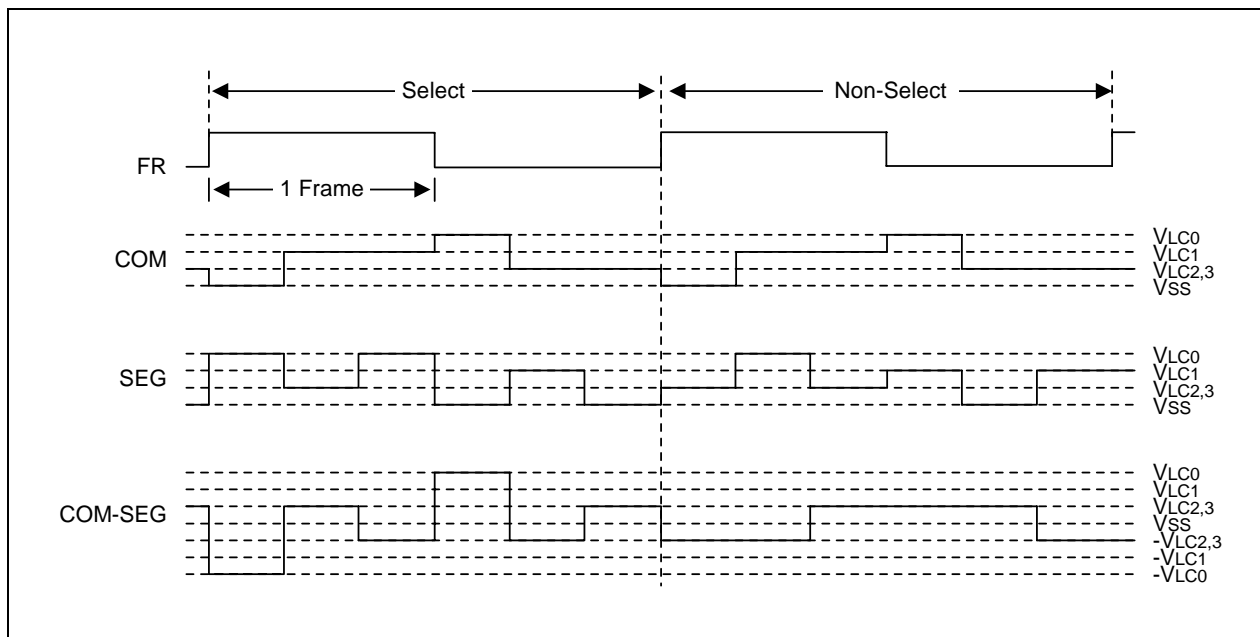


**Figure 14-7. Select/No-Select Signal in 1/3 Duty, 1/3 Bias Display Mode**
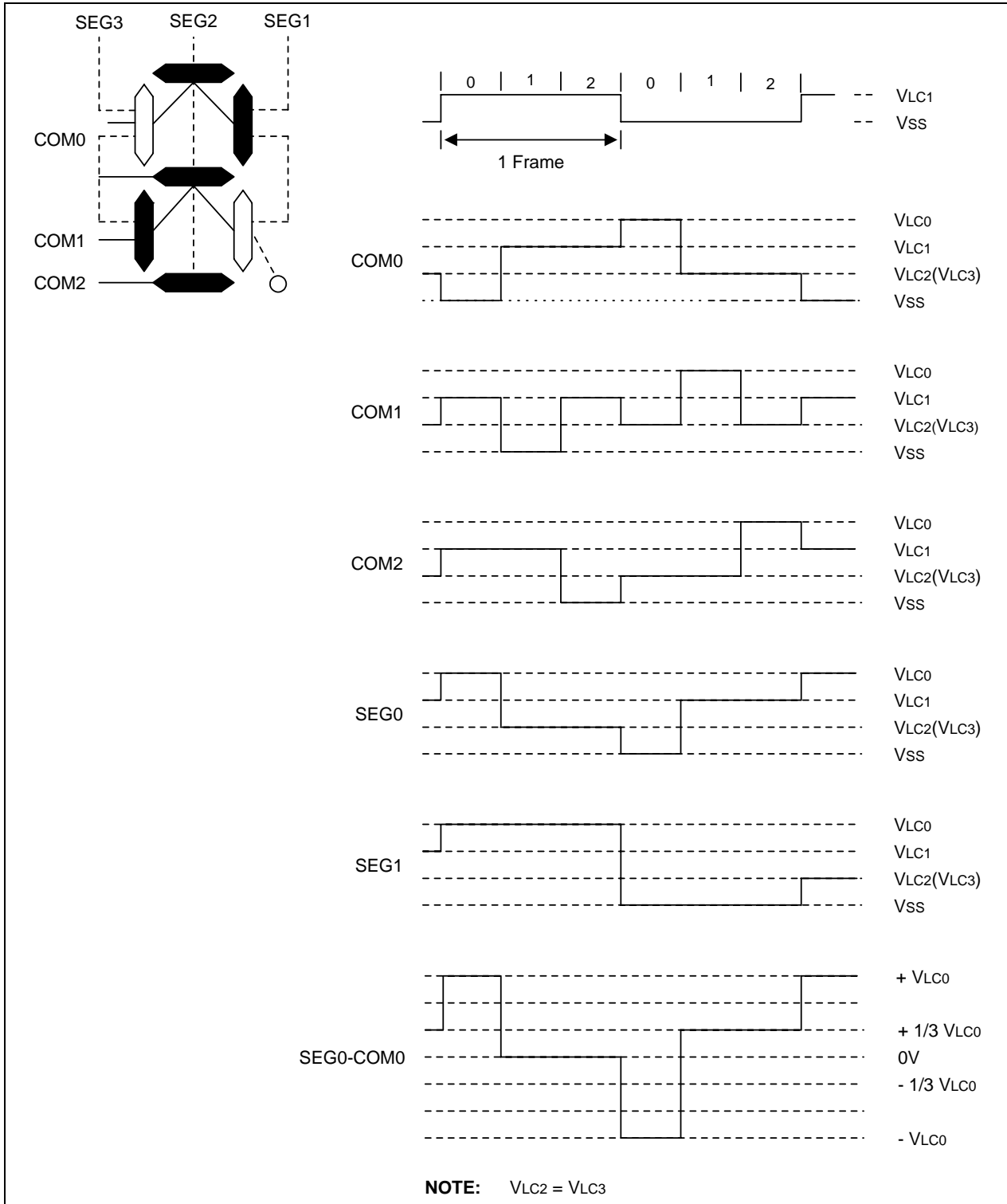
**Figure 14-8. LCD Signal Waveforms (1/3 Duty, 1/3 Bias)**
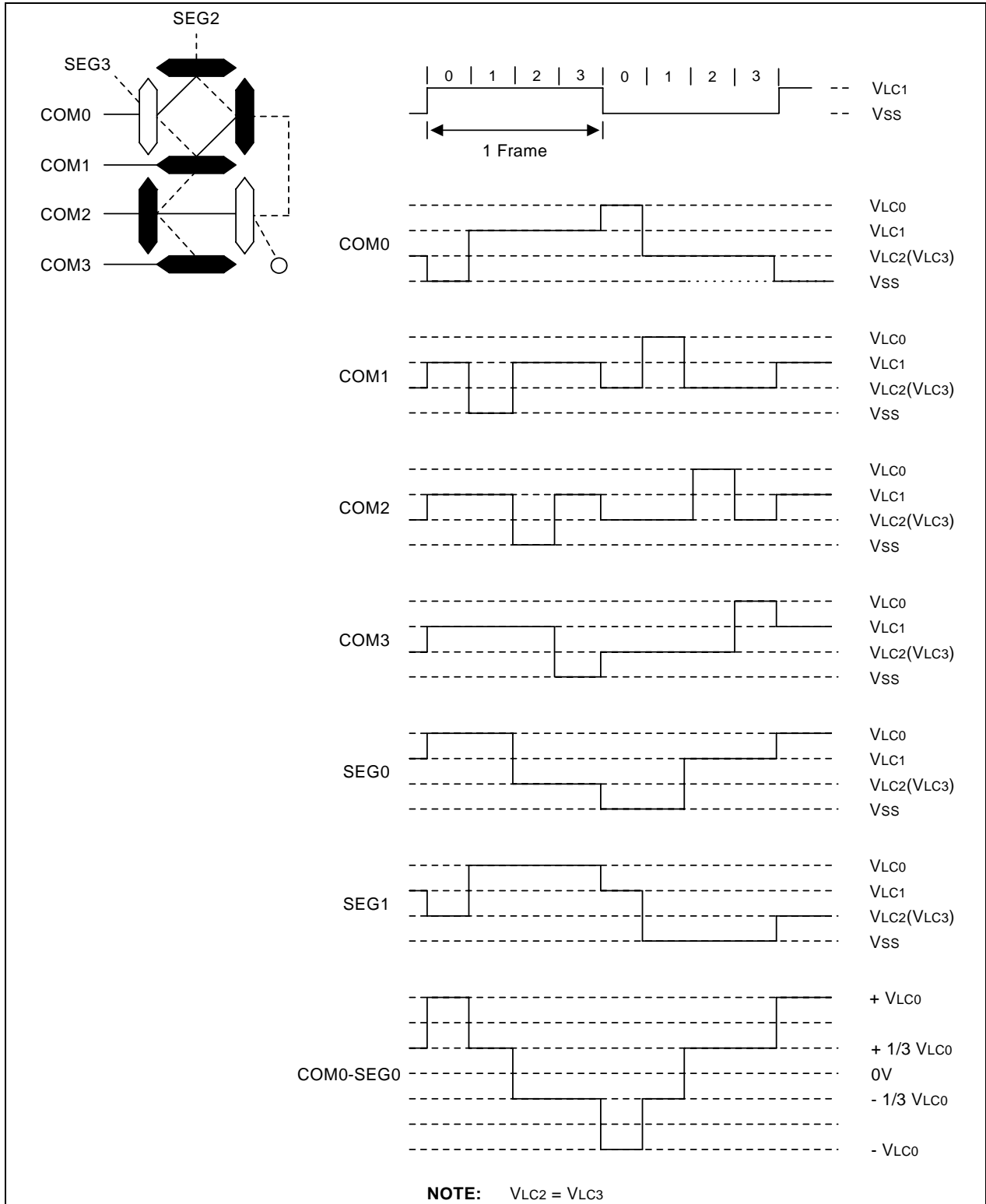
**Figure 14-9. LCD Signal Waveforms (1/4 Duty, 1/3 Bias)**
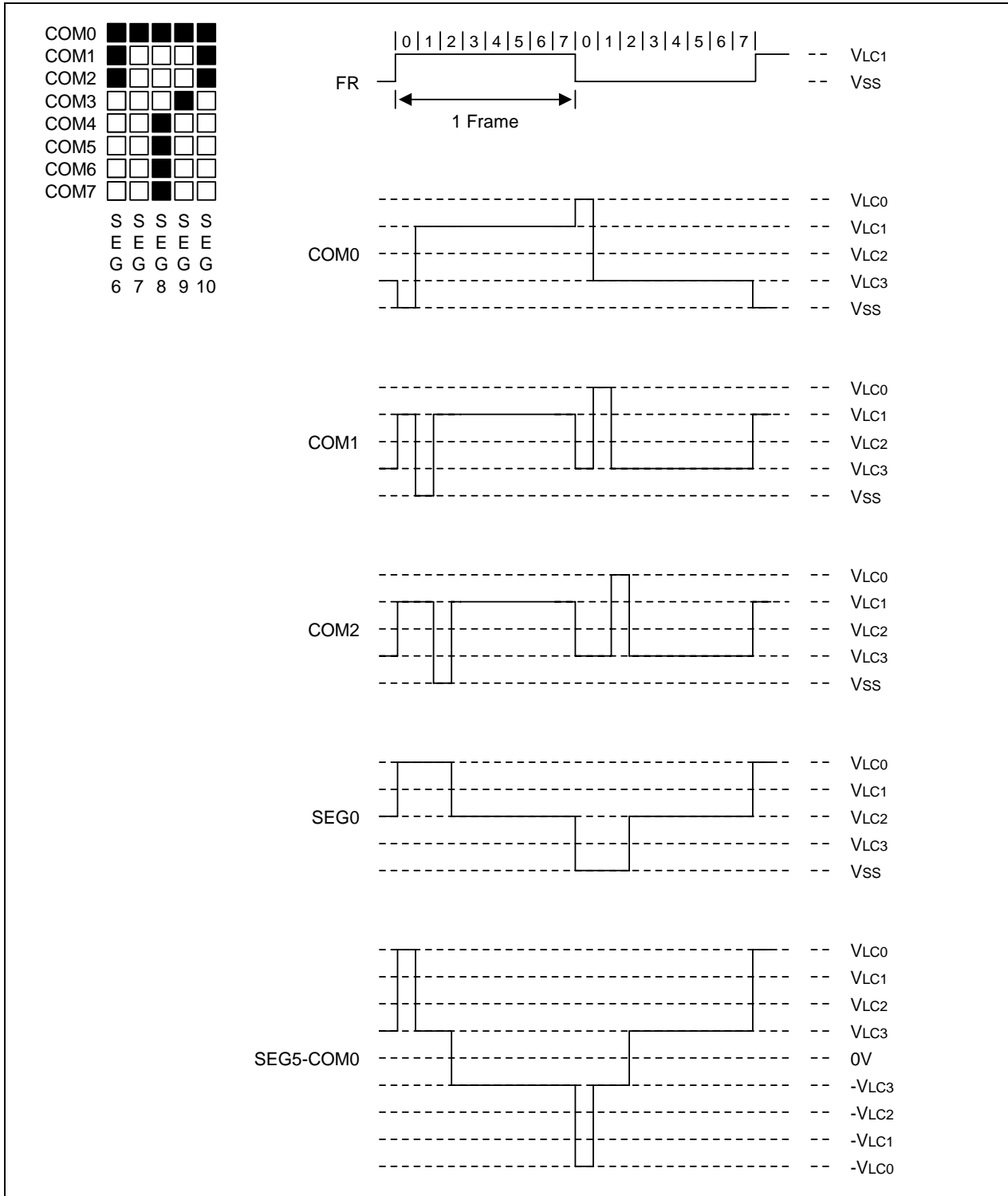
**Figure 14-10. LCD Signal Waveforms (1/8 Duty, 1/4 Bias)**

SAMSUNG
ELECTRONICS
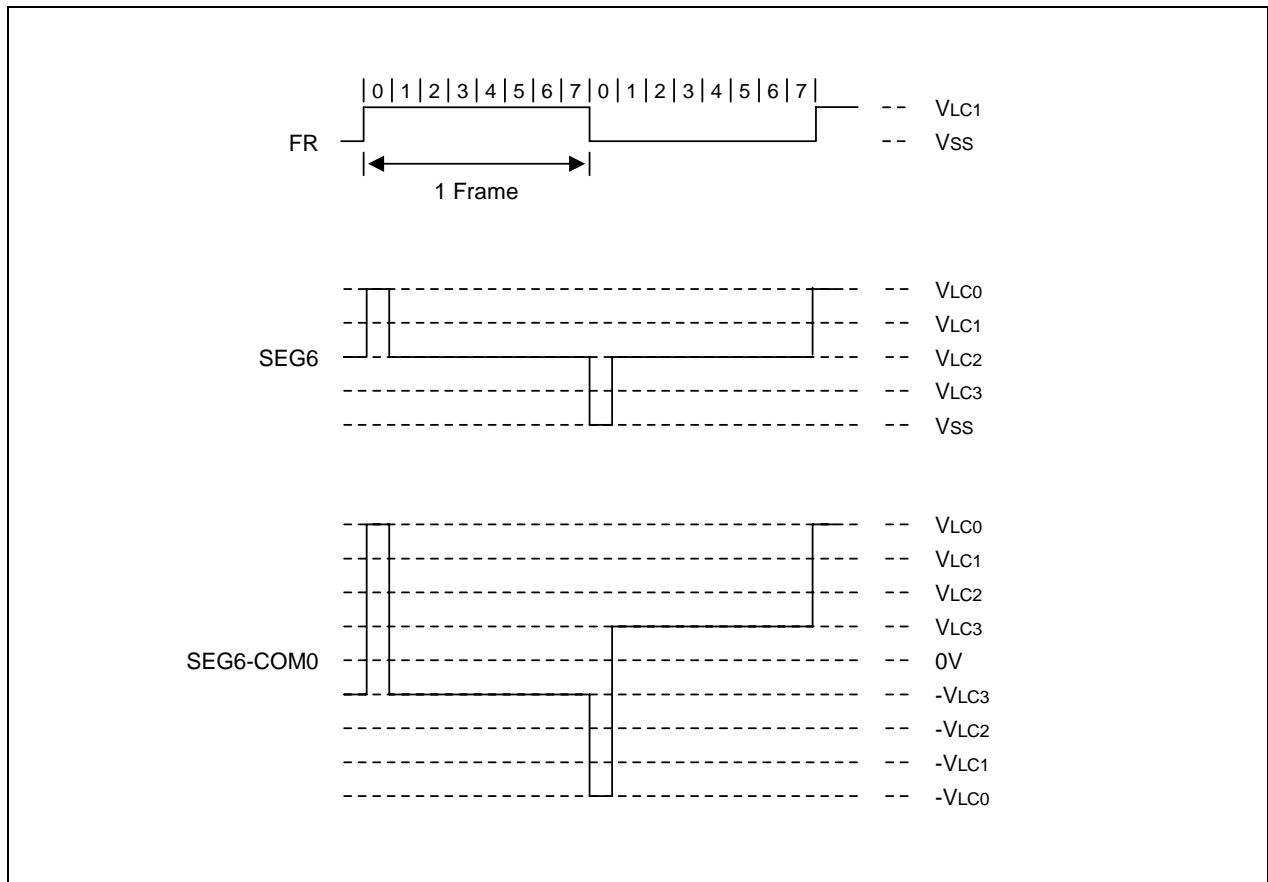
**Figure 14-10. LCD Signal Waveforms (1/8 Duty, 1/4 Bias) (Continued)**

# 15 10-BIT ANALOG-TO-DIGITAL CONVERTER

## OVERVIEW

The 10-bit A/D converter (ADC) module uses successive approximation logic to convert analog levels entering at one of the eight input channels to equivalent 10-bit digital values. The analog input level must lie between the $AV_{REF}$ and $AV_{SS}$ values. The A/D converter has the following components:

— Analog comparator with successive approximation logic

— D/A converter logic (resistor string type)

— ADC control register (ADCON)

— Eight multiplexed analog data input pins (AD0–AD7)

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

— 8-bit digital input port (Alternately, I/O port.)

— $AV_{REF}$ and $AV_{SS}$ pins, $AV_{SS}$ is internally connected to $V_{SS}$

## FUNCTION DESCRIPTION

To initiate an analog-to-digital conversion procedure, at the first you must set ADCEN signal for ADC input enable at port 2, the pin set with 1 can be used for ADC analog input. And you write the channel selection data in the A/D converter control register ADCON.4–.6 to select one of the eight analog input pins (ADC0–7) and set the conversion start or enable bit, ADCON.0. The read-write ADCON register is located in set 1, bank 0, at address F3H. The pins witch are not used for ADC can be used for normal I/O.

During a normal conversion, ADC logic initially sets the successive approximation register to 200H (the approximate half-way point of an 10-bit register). This register is then updated automatically during each conversion step. The successive approximation block performs 10-bit conversions for one input channel at a time. You can dynamically select different channels by manipulating the channel selection bit value (ADCON.6–4) in the ADCON register. To start the A/D conversion, you should set the enable bit, ADCON.0. When a conversion is completed, ADCON.3, the end-of-conversion (EOC) bit is automatically set to 1 and the result is dumped into the ADDATAH/L register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATAH/L before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

**NOTE**

Because the A/D converter has no sample-and-hold circuitry, it is very important that fluctuation in the analog level at the AD0–AD7 input pins during a conversion procedure be kept to an absolute minimum. Any change in the input level, perhaps due to noise, will invalidate the result. If the chip enters to STOP or IDLE mode in conversion process, there will be a leakage current path in A/D block. You must use STOP or IDLE mode after ADC operation is finished.

**CONVERSION TIMING**

The A/D conversion process requires 4 steps (4 clock edges) to convert each bit and 10 clocks to set-up A/D conversion. Therefore, total of 50 clocks are required to complete an 10-bit conversion: When fxx/8 is selected for conversion clock with an 8 MHz fxx clock frequency, one clock cycle is 1 us. Each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

4 clocks/bit $\times$ 10 bits + set-up time = 50 clocks, 50 clock $\times$ 1us = 50 $\mu$s at 1 MHz

**A/D CONVERTER CONTROL REGISTER (ADCON)**

The A/D converter control register, ADCON, is located at address F3H in set 1, bank 0. It has three functions:

— Analog input pin selection (ADCON.6–.4)

— End-of-conversion status detection (ADCON.3)

— ADC clock selection (ADCON.2–.1)

— A/D operation start or enable (ADCCON.0)

After a reset, the start bit is turned off. You can select only one analog input channel at a time. Other analog input pins (AD0–AD7) can be selected dynamically by manipulating the ADCON.4–6 bits. And the pins not used for analog input can be used for normal I/O function.
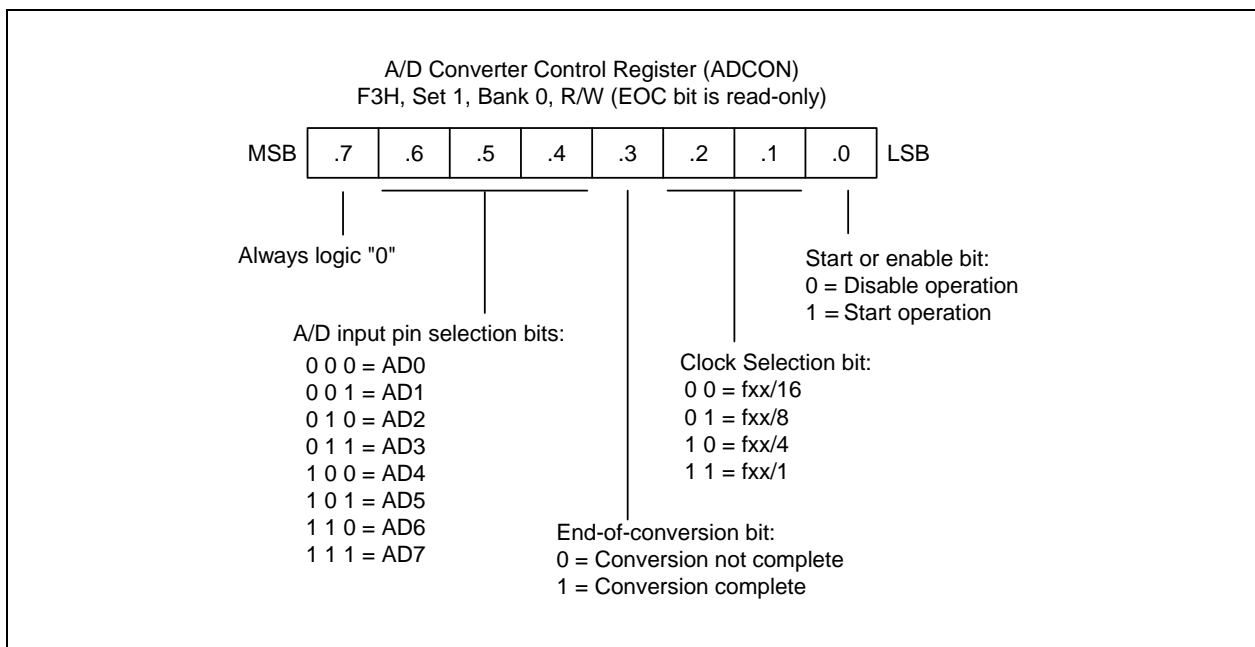


**Figure 15-1. A/D Converter Control Register (ADCON)**

A/D Converter Data Register, High Byte (ADDATAH)
F4H, Set 1, Bank 0, Read Only

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

A/D Converter Data Register, Low Byte (ADDATAL)
F5H, Set 1, Bank 0, Read Only

| MSB | — | — | — | — | — | — | .1 | .0 | LSB |
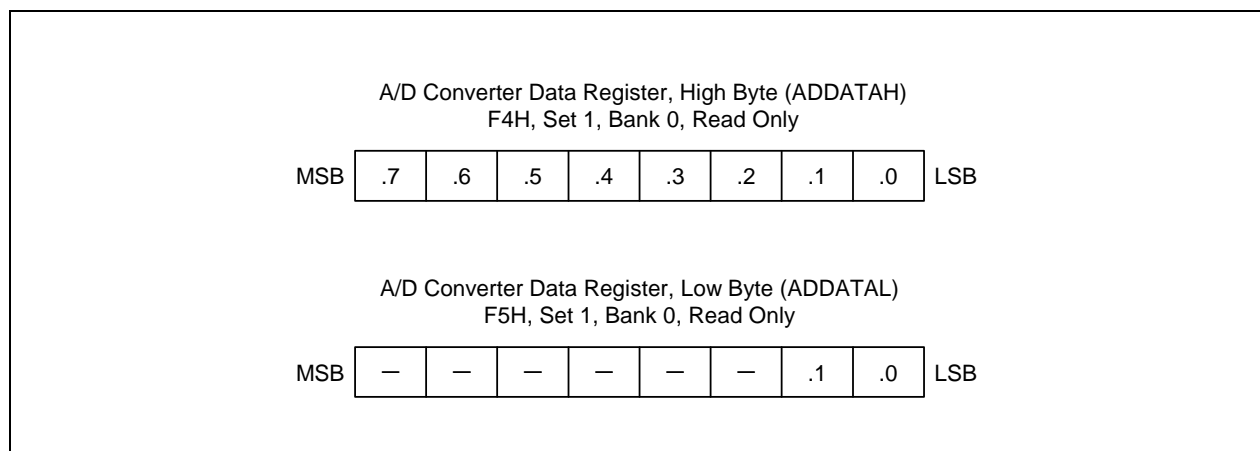
**Figure 15-2. A/D Converter Data Register (ADDATAH/L)**

**INTERNAL REFERENCE VOLTAGE LEVELS**

In the ADC function block, the analog input voltage level is compared to the reference voltage. The analog input level must remain within the range $AV_{SS}$ to $AV_{REF}$ (usually, $AV_{REF} \leq V_{DD}$).

Different reference voltage levels are generated internally along the resistor tree during the analog conversion process for each conversion step. The reference voltage level for the first conversion bit is always 1/2 $AV_{REF}$.

## BLOCK DIAGRAM
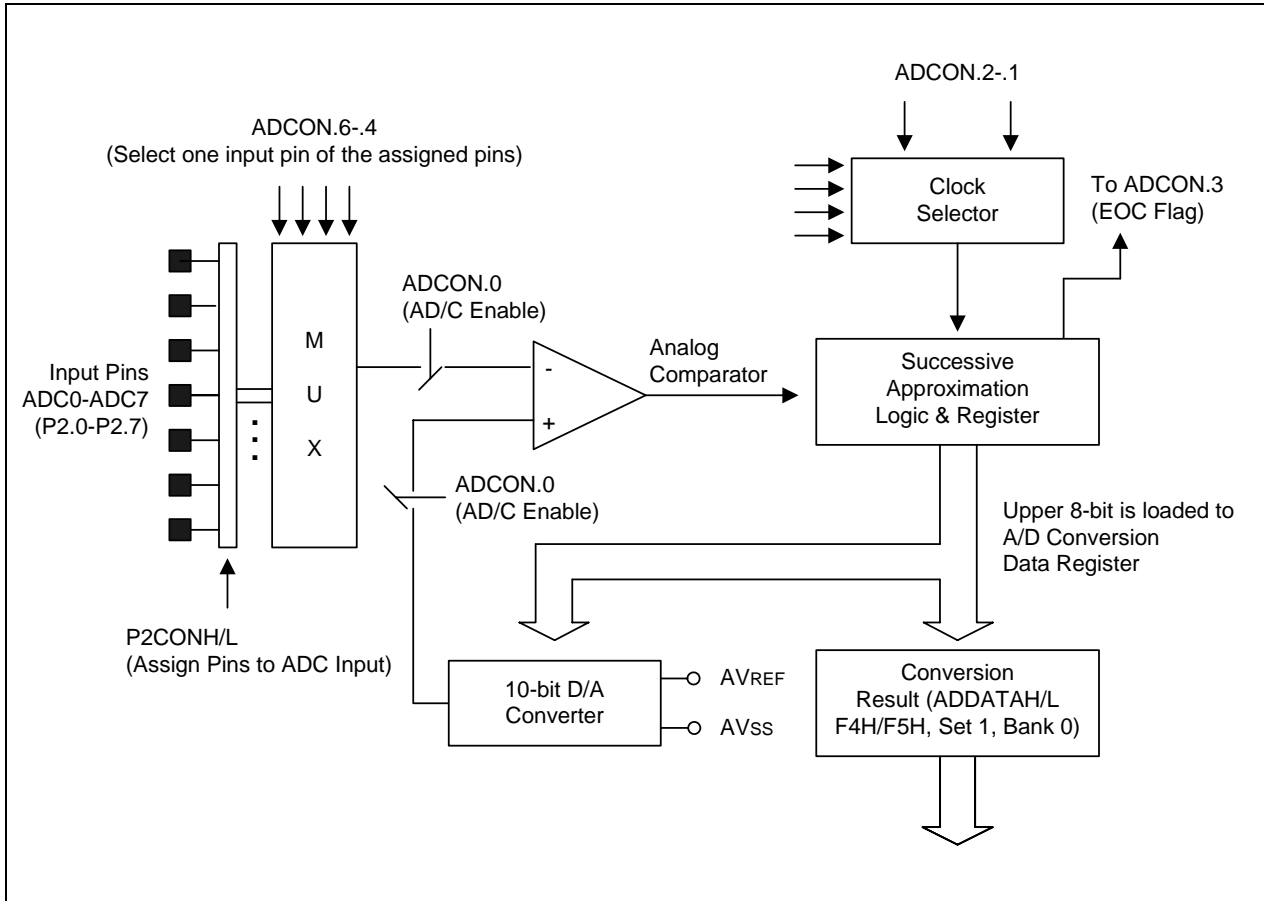


**Figure 15-3. A/D Converter Functional Block Diagram**
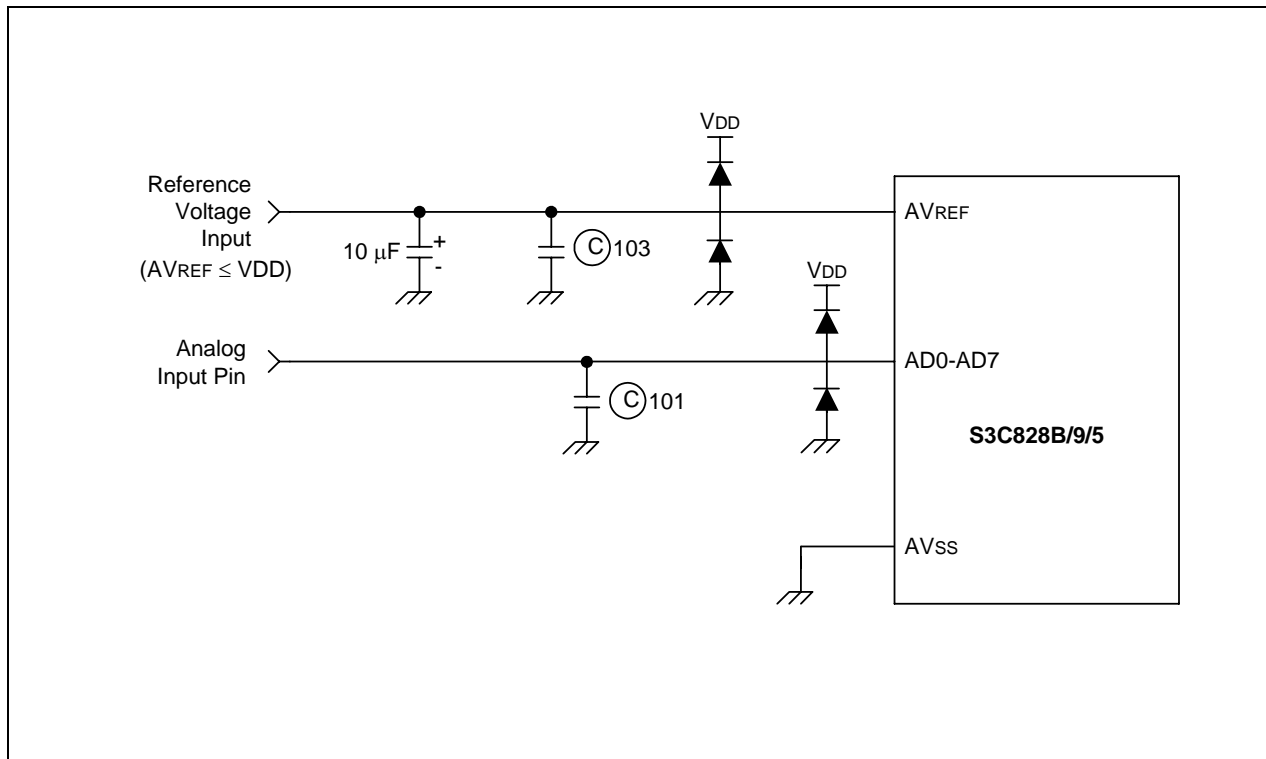
**Figure 15-4. Recommended A/D Converter Circuit for Highest Absolute Accuracy**

# 16 SERIAL I/O INTERFACE

## OVERVIEW

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

— 8-bit control register (SIOCON)

— Clock selector logic

— 8-bit data buffer (SIODATA)

— 8-bit pre-scaler (SIOPS)

— 3-bit serial clock counter

— Serial data I/O pins (SI, SO)

— External clock input/output pins (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 modules, follow these basic steps:

1. Configure the I/O pins at port (SO, SCK, SI) by loading the appropriate value to the P1CONH 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.1) 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) is set to "1" and an SIO interrupt request is generated.

SAMSUNG
ELECTRONICS

## SIO CONTROL REGISTER (SIOCON)

The control register for serial I/O interface module, SIOCON, is located at E0H in set 1, bank 0. It has the control settings for SIO module.

— Clock source selection (internal or external) for shift clock

— Interrupt enable

— Edge selection for shift operation

— Clear 3-bit counter and start shift operation

— Shift operation (transmit) enable

— Mode selection (transmit/receive or receive-only)

— Data direction selection (MSB first or LSB first)

A reset clears the SIOCON value to "00H". This configures the corresponding module with an internal clock source at the SCK, selects receive-only operating mode, and clears the 3-bit counter. The data shift operation and the interrupt are disabled. The selected data direction is MSB-first.

Serial I/O Module Control Register (SIOCON)
E0H, Set 1, Bank 0, R/W

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

SIO shift clock selection bit:
0 = Internal clock (P.S Clock)
1 = External clock (SCK)

Data direction control bit:
0 = MSB-first mode
1 = LSB-first mode

SIO mode selection bit:
0 = Receive only mode
1 = Transmit/receive mode

Shift clock edge selection bit:
0 = Tx at falling edges, Rx at rising edges
1 = Tx at rising edges, Rx at falling edges

SIO counter clear and shift start bit:
0 = No action
1 = Clear 3-bit counter and start shifting

SIO shift operation enable bit:
0 = Disable shifter and clock counter
1 = Enable shifter and clock counter

SIO interrupt enable bit:
0 = Disable SIO interrupt
1 = Enable SIO interrupt

SIO interrupt pending bit:
0 = No interrupt pending
0 = Clear pending condition
     (when write)
1 = Interrupt is pending

**Figure 16-1. Serial I/O Module Control Registers (SIOCON)**

## SIO PRE-SCALER REGISTER (SIOPS)

The control register for serial I/O interface module, SIOPS, is located at E2H 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)/(Pre-scaler value + 1), or SCK input clock, where the input clock is fxx/4
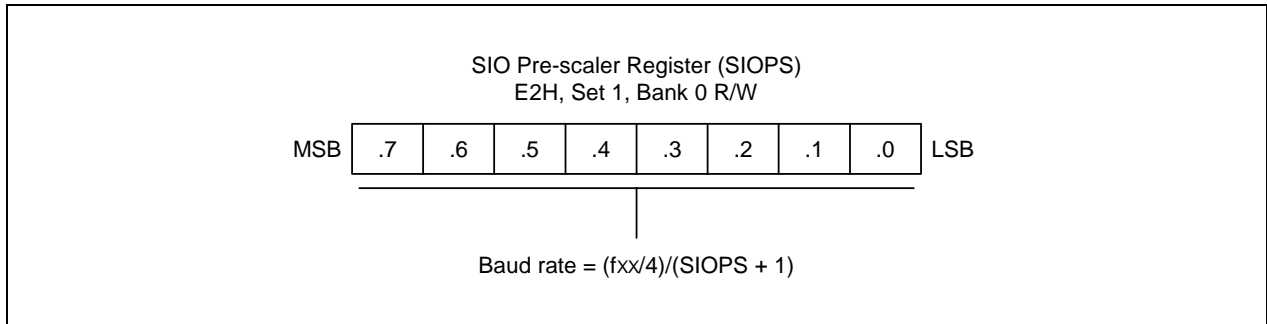


**Figure 16-2. SIO Pre-scaler Register (SIOPS)**

## BLOCK DIAGRAM
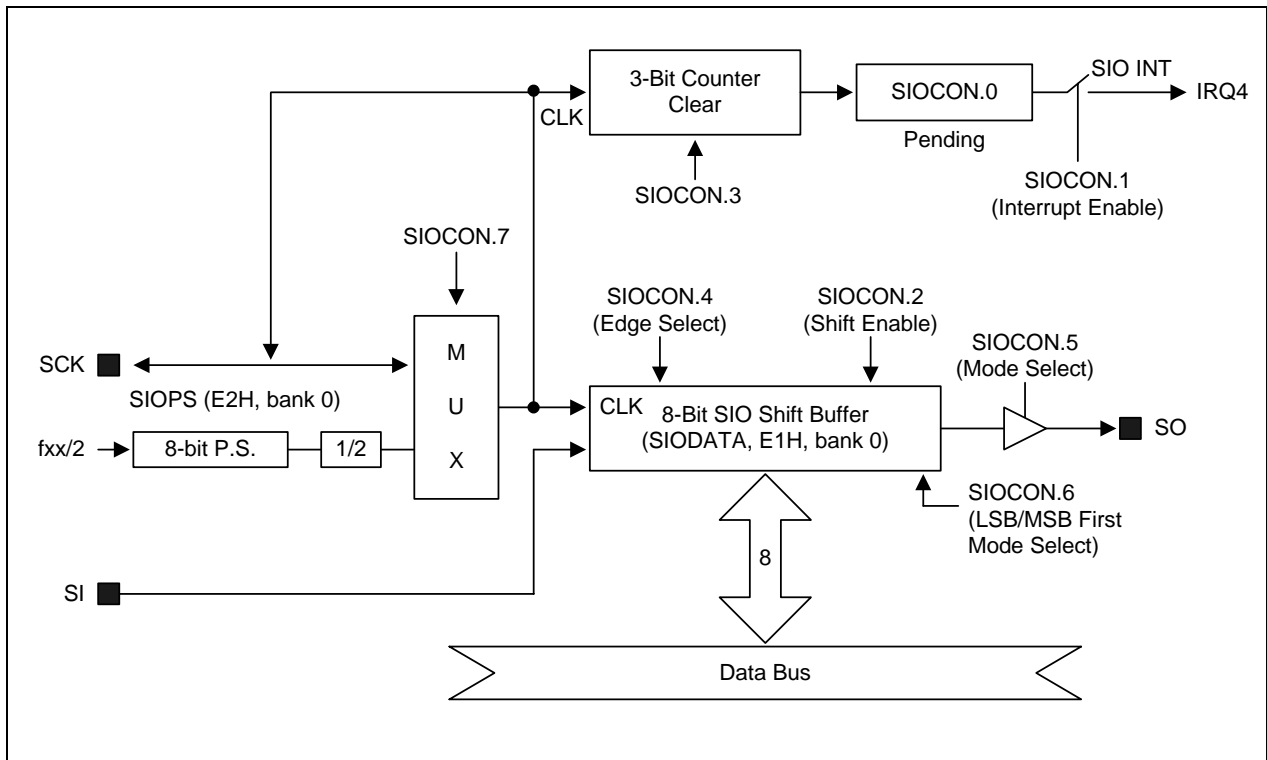


**Figure 16-3. SIO Functional Block Diagram**

SERIAL I/O TIMING DIAGRAM
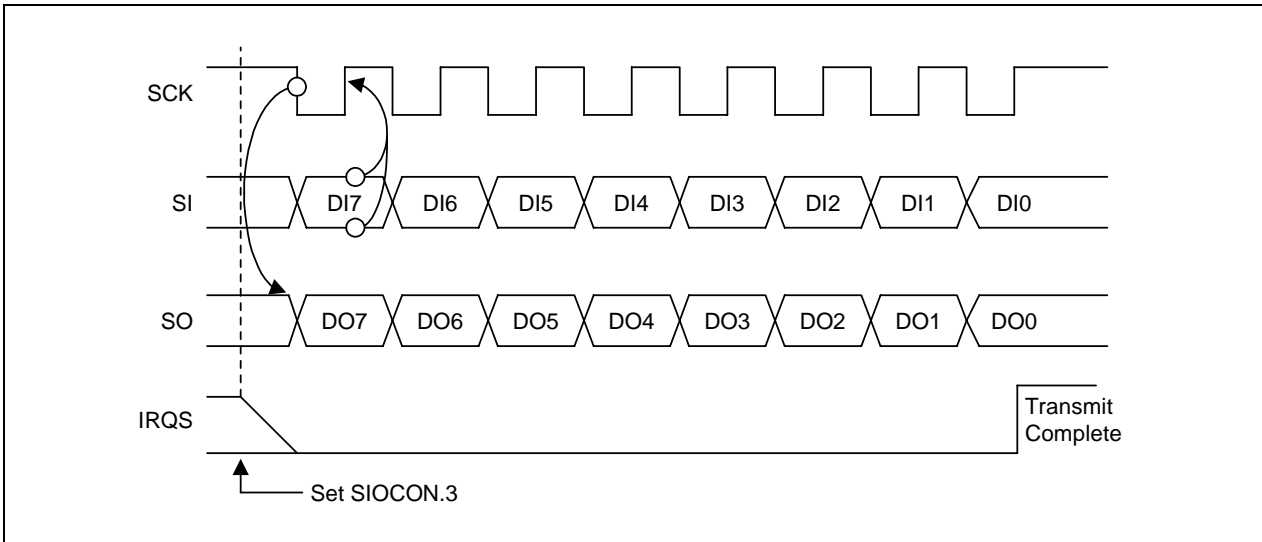


**Figure 16-4. Serial I/O Timing in Transmit/Receive Mode (Tx at falling, SIOCON.4 = 0)**
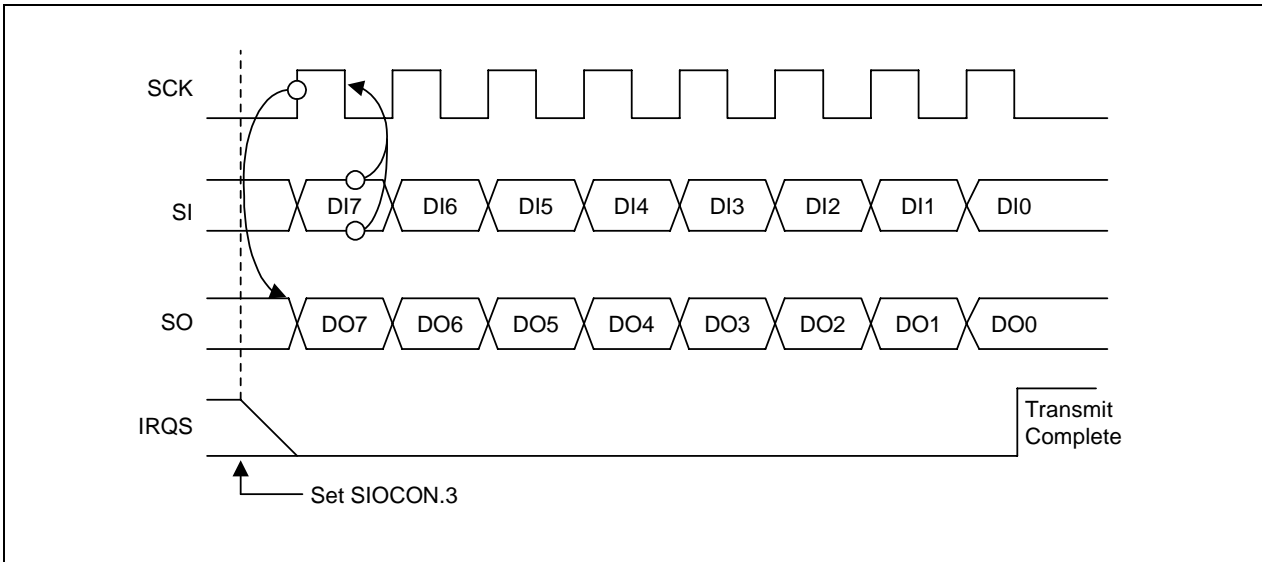


**Figure 16-5. Serial I/O Timing in Transmit/Receive Mode (Tx at rising, SIOCON.4 = 1)**

# 17 UART

## OVERVIEW

The UART block has a full-duplex serial port with programmable operating modes: There is one synchronous mode and three UART (Universal Asynchronous Receiver/Transmitter) modes:

— Serial I/O with baud rate of $fxx/(16 \times (BRDATA+1))$
— 8-bit UART mode; variable baud rate
— 9-bit UART mode; $fxx/16$
— 9-bit UART mode, variable baud rate

UART receive and transmit buffers are both accessed via the data register, UDATA, is set 1, bank 0 at address F7H. Writing to the UART data register loads the transmit buffer; reading the UART data register accesses a physically separate receive buffer.

When accessing a receive data buffer (shift register), reception of the next byte can begin before the previously received byte has been read from the receive register. However, if the first byte has not been read by the time the next byte has been completely received, one of the bytes will be lost.

In all operating modes, transmission is started when any instruction (usually a write operation) uses the UDATA register as its destination address. In mode 0, serial data reception starts when the receive interrupt pending bit (INTPND.5) is "0" and the receive enable bit (UARTCON.4) is "1". In mode 1, 2, and 3, reception starts whenever an incoming start bit ("0") is received and the receive enable bit (UARTCON.4) is set to "1".

### PROGRAMMING PROCEDURE

To program the UART modules, follow these basic steps:

1. Configure P3.5 and P3.4 to alternative function (RxD (P3.5), TxD (P3.4)) for UART module by setting the P3CONH register to appropriately value.

2. Load an 8-bit value to the UARTCON control register to properly configure the UART I/O module.

3. For interrupt generation, set the UART I/O interrupt enable bit (UARTCON.1 or UARTCON.0) to "1".

4. When you transmit data to the UART buffer, write data to UDATA, the shift operation starts.

5. When the shift operation (transmit/receive) is completed, UART pending bit (INTPND.4 or INTPND.5) is set to "1" and an UART interrupt request is generated.

**SAMSUNG**
ELECTRONICS

## UART CONTROL REGISTER (UARTCON)

The control register for the UART is called UARTCON in set 1, bank 0 at address F6H. It has the following control functions:

— Operating mode and baud rate selection

— Multiprocessor communication and interrupt control

— Serial receive enable/disable control

— 9th data bit location for transmit and receive operations (modes 2 and 3 only)

— UART transmit and receive interrupt control

A reset clears the UARTCON value to "00H". So, if you want to use UART module, you must write appropriate value to UARTCON.

```
                    UART Control Register (UARTCON)
                        F6H, Set 1, Bank 0,  R/W

        MSB  | MS1 | MS0 | MCE | RE | TB8 | RB8 | RIE | TIE |  LSB


          Operating mode and                      Transmit interrupt enable bit:
          baud rate selection bits:               0 = Disable
          (see table below)                       1 = Enable

        Multiprocessor communication(1)          Received interrupt enable bit:
        enable bit (for modes 2 and 3 only):      0 = Disable
        0 = Disable                               1 = Enable
        1 = Enable

              Serial data receive enable bit:    Location of the 9th data bit that was
              0 = Disable                        received in UART mode 2 or 3 ("0" or "1")
              1 = Enable

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

| MS1 | MS0 | Mode | Description[2] | Baud Rate |
|-----|-----|------|-------------|-----------|
| 0 | 0 | 0 | Shift register | $(f_{xx}/(16 \times (BRDATA + 1)))$ |
| 0 | 1 | 1 | 8-bit UART | $(f_{xx}/(16 \times (BRDATA + 1)))$ |
| 1 | 0 | 2 | 9-bit UART | $(f_{xx}/16)$ |
| 1 | 1 | 3 | 9-bit UART | $(f_{xx}/(16 \times (BRDATA + 1)))$ |

**NOTES:**
1.  In mode 2 or 3, if the UARTCON.5 bit is set to "1" then the receive interrupt will not be activated if the received 9th data bit is "0". In mode 1, if UARTCON.5 = "1" then the receive interrut will not be activated if a valid stop bit was not received.
    In mode 0, the UARTCON.5 bit should be "0"
2.  The descriptions for 8-bit and 9-bit UART mode do not include start and stop bits for serial data receive and transmit.
3.  The interrupt pending bits of Rx and Tx are in the INTPND register.

**Figure 17-1. UART Control Register (UARTCON)**

**SAMSUNG**
**ELECTRONICS**

## UART INTERRUPT PENDING BITS

The UART interrupt pending bits, INTPND.5–.4, are located in set 1, bank 0 at address F9H, it contains the UART data transmit interrupt pending bit (INTPND.4) and the receive interrupt pending bit (INTPND.5).

In mode 0, the receive interrupt pending bit INTPND.5 is set to "1" when the 8th receive data bit has been shifted. In mode 1, 2, and 3, the INTPND.5 bit is set to "1" at the halfway point of the stop bit's shift time. When the CPU has acknowledged the receive interrupt pending condition, the INTPND.5 bit must then be cleared by software in the interrupt service routine.

In mode 0, the transmit interrupt pending bit INTPND.4 is set to "1" when the 8th transmit data bit has been shifted. In mode 1, 2, or 3, the INTPND.4 bit is set at the start of the stop bit. When the CPU has acknowledged the transmit interrupt pending condition, the INTPND.4 bit must then be cleared by software in the interrupt service routine.
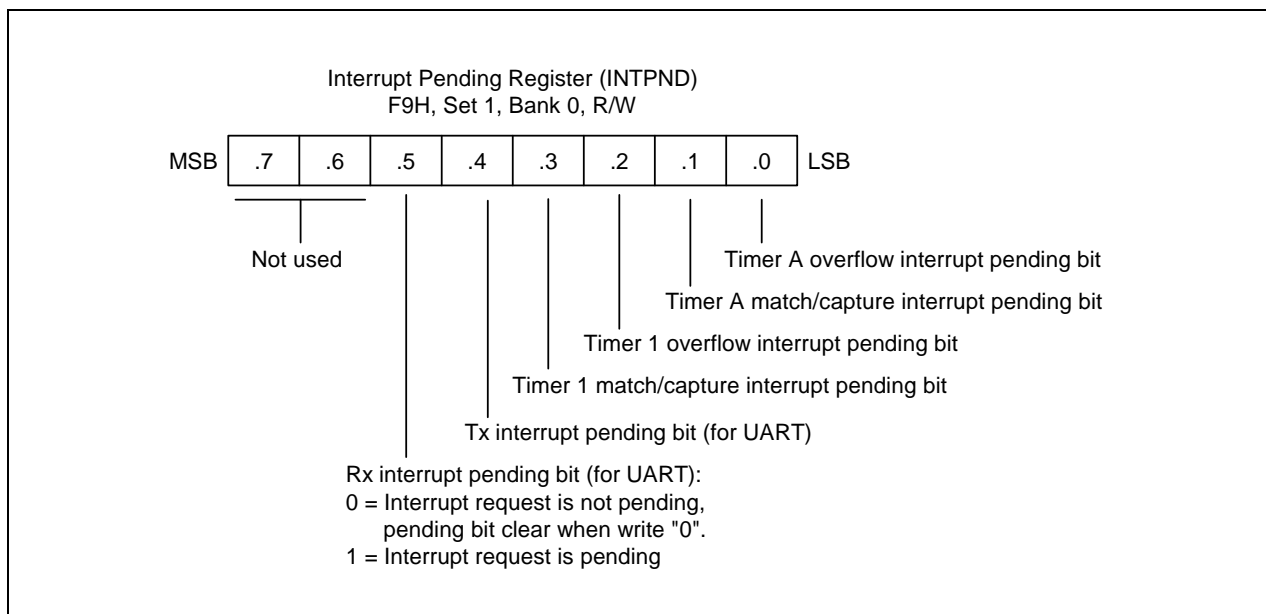


**Figure 17-2. UART Interrupt Pending Bits (INTPND.5–.4)**

**UART DATA REGISTER (UDATA)**



UART Data Register (UDATA)
F7H, Set 1, Bank 0, R/W

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

Transmit or receive data
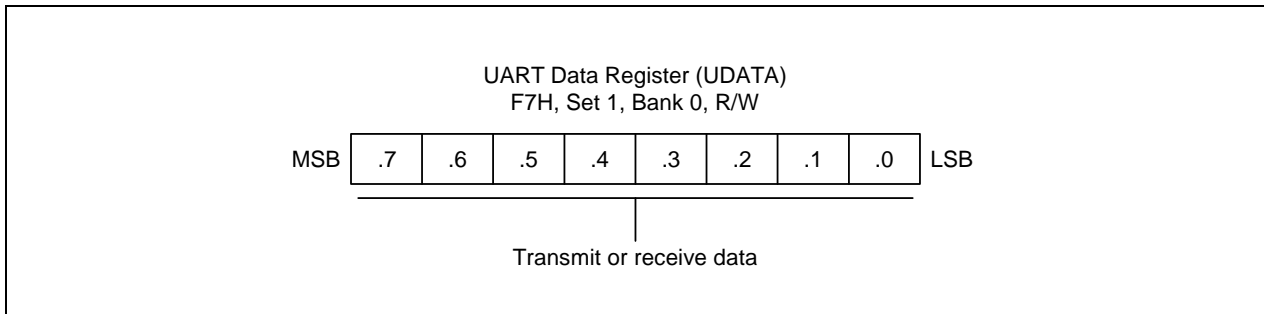
**Figure 17-3. UART Data Register (UDATA)**

**UART BAUD RATE DATA REGISTER (BRDATA)**

The value stored in the UART baud rate register, BRDATA, lets you determine the UART clock rate (baud rate).



UART Baud Rate Data Register (BRDATA)
F8H, Set 1, Bank 0, R/W

MSB | .7 | .6 | .5 | .4 | .3 | .2 | .1 | .0 | LSB
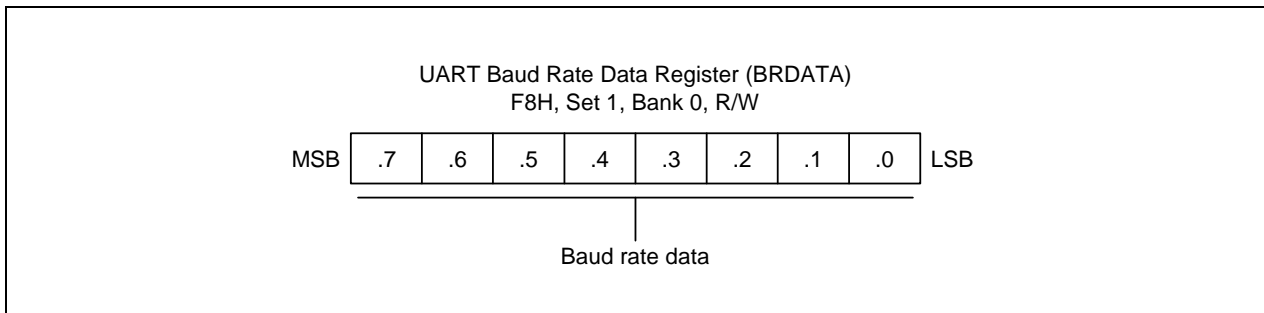
Baud rate data

**Figure 17-4. UART Baud Rate Data Register (BRDATA)**

**BAUD RATE CALCULATIONS**

**Mode 0 Baud Rate Calculation**

In mode 0, the baud rate is determined by the UART baud rate data register, BRDATA in set 1, bank 0 at address F8H: Mode 0 baud rate = $f_{xx}/(16 \times (BRDATA + 1))$.

**Mode 2 Baud Rate Calculation**

The baud rate in mode 2 is fixed at the $f_{OSC}$ clock frequency divided by 16: Mode 2 baud rate = $f_{xx}/16$

**Modes 1 and 3 Baud Rate Calculation**

In modes 1 and 3, the baud rate is determined by the UART baud rate data register, BRDATA in set 1, bank 0 at address F8H: Mode 1 and 3 baud rate = $f_{xx}/(16 \times (BRDATA + 1))$

SAMSUNG
ELECTRONICS

Table 17-1. Commonly Used Baud Rates Generated by BRDATA

| Mode | Baud Rate | Oscillation Clock | BRDATA | |
|---|---|---|---|---|
| | | | **Decimal** | **Hexdecimal** |
| Mode 2 | 0.5 MHz | 8 MHz | x | x |
| Mode 0 Mode 1 Mode 3 | 230.400 Hz | 11.0592 MHz | 02 | 02H |
| | 115.200 Hz | 11.0592 MHz | 05 | 05H |
| | 57.600 Hz | 11.0592 MHz | 11 | 0BH |
| | 38.400 Hz | 11.0592 MHz | 17 | 11H |
| | 19.200 Hz | 11.0592 MHz | 35 | 23H |
| | 9.600 Hz | 11.0592 MHz | 71 | 47H |
| | 4.800 Hz | 11.0592 MHz | 143 | 8FH |
| | 62.500 Hz | 10 MHz | 09 | 09H |
| | 9.615 Hz | 10 MHz | 64 | 40H |
| | 38.461 Hz | 8 MHz | 12 | 0CH |
| | 12.500 Hz | 8 MHz | 39 | 27H |
| | 19.230 Hz | 4 MHz | 12 | 0CH |
| | 9.615 Hz | 4 MHz | 25 | 19H |

# BLOCK DIAGRAM
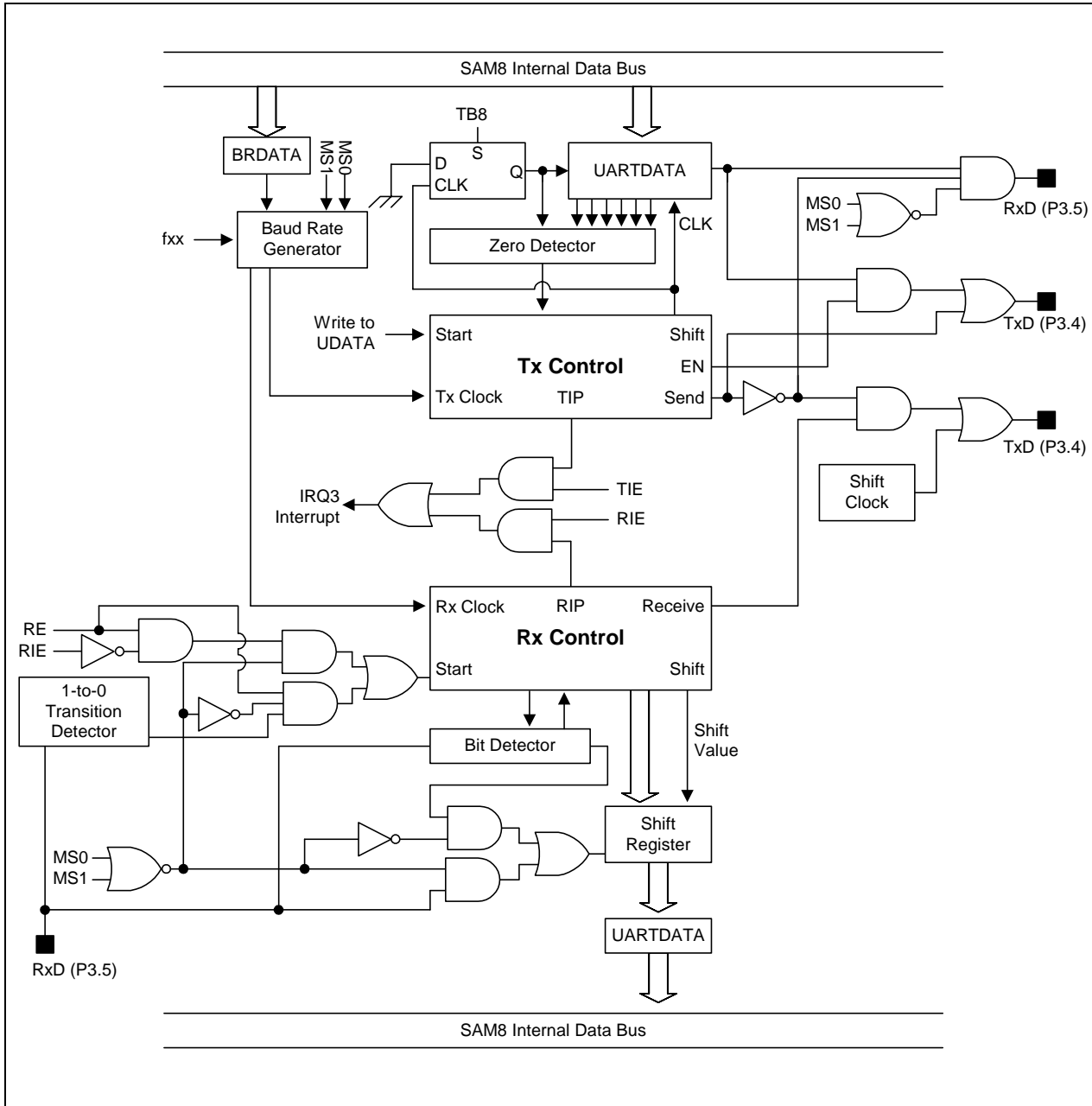


**Figure 17-5. UART Functional Block Diagram**

## UART MODE 0 FUNCTION DESCRIPTION

In mode 0, UART is input and output through the RxD (P3.5) pin and TxD (P3.4) pin outputs the shift clock. Data is transmitted or received in 8-bit units only. The LSB of the 8-bit value is transmitted (or received) first.

### Mode 0 Transmit Procedure

1.  Select mode 0 by setting UARTCON.6 and .7 to "00B".

2.  Write transmission data to the shift register UDATA (F7H, set 1, bank 0) to start the transmission operation.

### Mode 0 Receive Procedure

1.  Select mode 0 by setting UARTCON.6 and .7 to "00B".

2.  Clear the receive interrupt pending bit (INTPND.5) by writing a "0" to INTPND.5.

3.  Set the UART receive enable bit (UARTCON.4) to "1".

4.  The shift clock will now be output to the TxD (P3.4) pin and will read the data at the RxD (P3.5) pin. A UART receive interrupt (IRQ5, vector ECH) occurs when UARTCON.1 is set to "1".
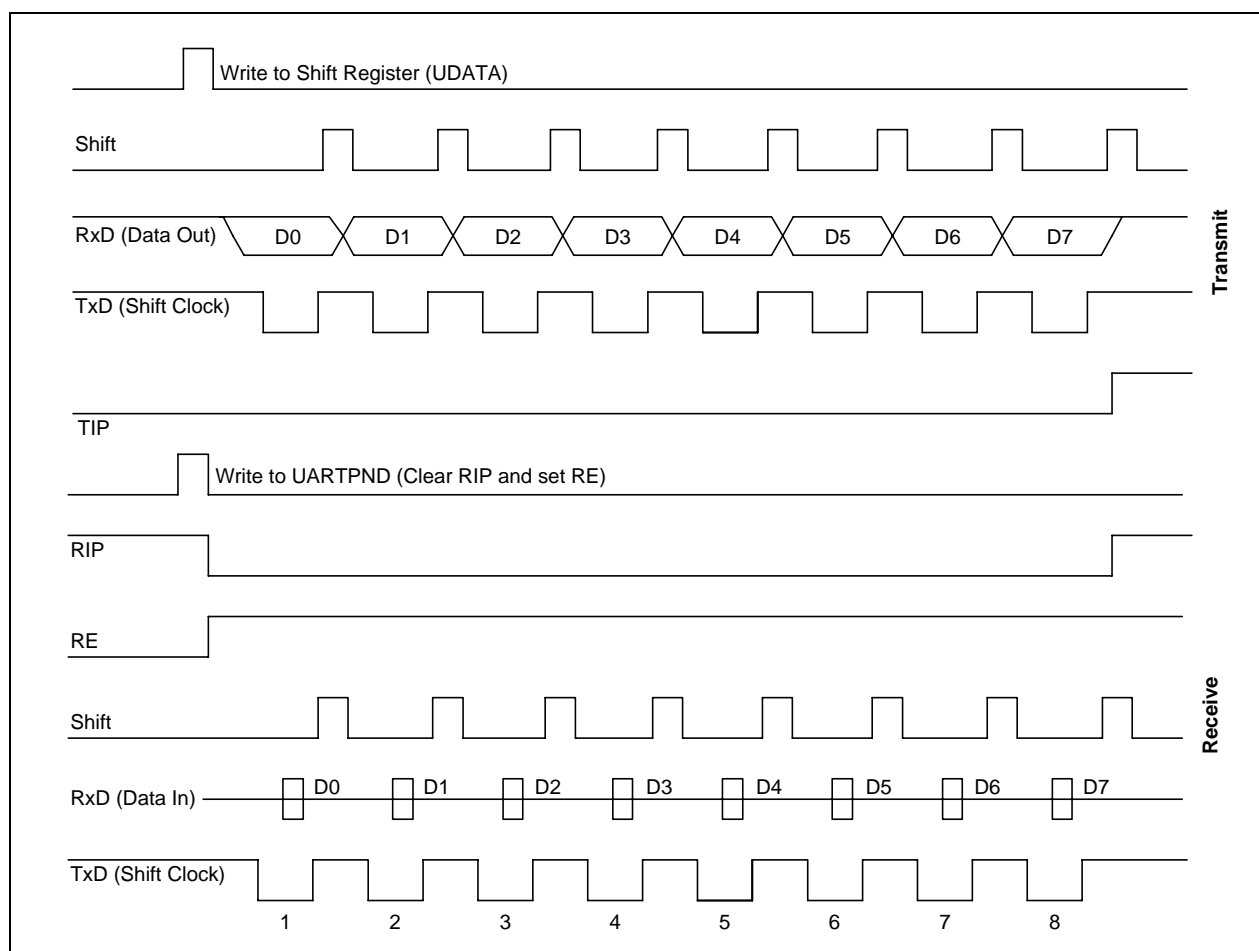


**Figure 17-6. Timing Diagram for Serial Port Mode 0 Operation**

**SERIAL PORT MODE 1 FUNCTION DESCRIPTION**

In mode 1, 10-bits are transmitted (through the TxD (P3.4) pin) or received (through the RxD (P3.5) pin). Each data frame has three components:

— Start bit ("0")

— 8 data bits (LSB first)

— Stop bit ("1")

When receiving, the stop bit is written to the RB8 bit in the UARTCON register. The baud rate for mode 1 is variable.

**Mode 1 Transmit Procedure**

1.  Select the baud rate generated by BRDATA.

2.  Select mode 1 (8-bit UART) by setting UARTCON bits 7 and 6 to '01B'.

3.  Write transmission data to the shift register UDATA (F7H, set 1, bank 0). The start and stop bits are generated automatically by hardware.

**Mode 1 Receive Procedure**

1.  Select the baud rate to be generated by BRDATA.

2.  Select mode 1 and set the RE (Receive Enable) bit in the UARTCON register to "1".

3.  The start bit low ("0") condition at the RxD (P3.5) pin will cause the UART module to start the serial data receive operation.
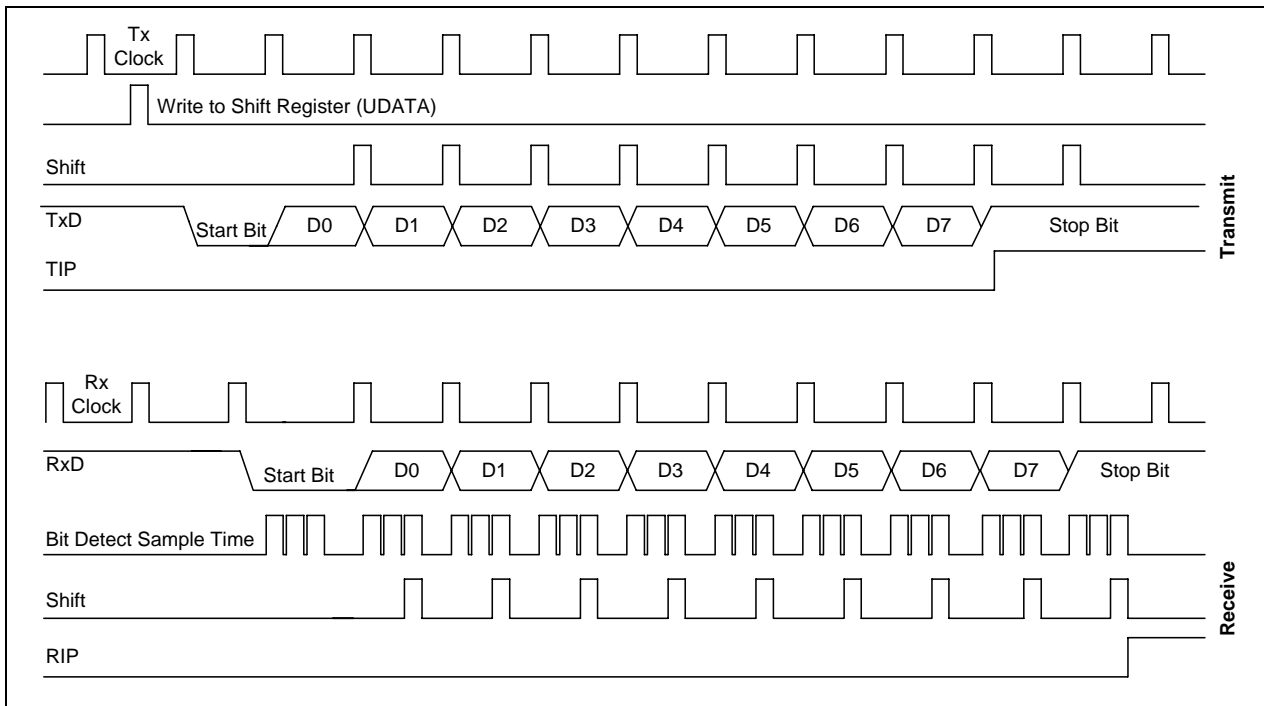


**Figure 17-7. Timing Diagram for Serial Port Mode 1 Operation**

**SERIAL PORT MODE 2 FUNCTION DESCRIPTION**

In mode 2, 11-bits are transmitted (through the TxD (P3.4) pin) or received (through the RxD (P3.5) pin). Each data frame has four components:

— Start bit ("0")

— 8 data bits (LSB first)

— Programmable 9th data bit

— Stop bit ("1")

The 9th data bit to be transmitted can be assigned a value of "0" or "1" by writing the TB8 bit (UARTCON.3). When receiving, the 9th data bit that is received is written to the RB8 bit (UARTCON.2), while the stop bit is ignored. The baud rate for mode 2 is fosc/16 clock frequency.

**Mode 2 Transmit Procedure**

1.  Select mode 2 (9-bit UART) by setting UARTCON bits 6 and 7 to '10B'. Also, select the 9th data bit to be transmitted by writing TB8 to "0" or "1".

2.  Write transmission data to the shift register, UDATA (F7H, set 1, bank 0), to start the transmit operation.

**Mode 2 Receive Procedure**

1.  Select mode 2 and set the receive enable bit (RE) in the UARTCON register to "1".

2.  The receive operation starts when the signal at the RxD (P3.5) pin goes to low level.
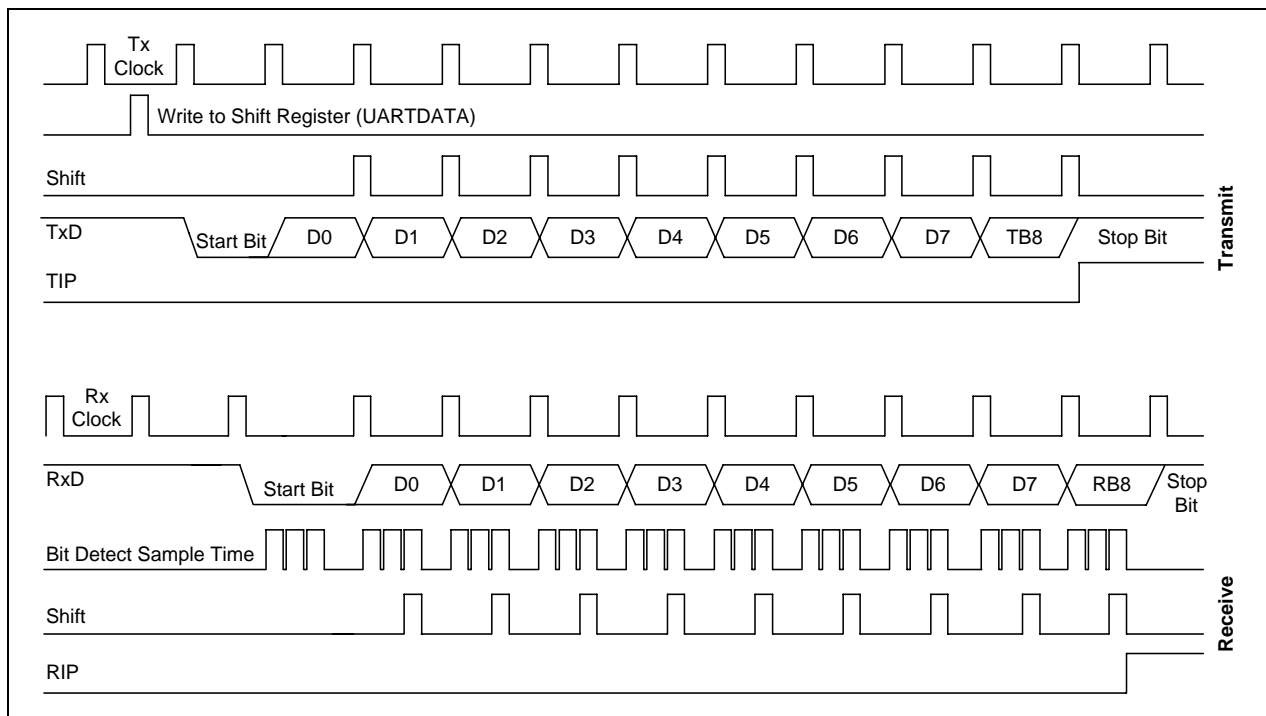


**Figure 17-8. Timing Diagram for Serial Port Mode 2 Operation**

## SERIAL PORT MODE 3 FUNCTION DESCRIPTION

In mode 3, 11-bits are transmitted (through the TxD (P3.4) pin) or received (through the RxD (P3.5) pin). Mode 3 is identical to mode 2 except for baud rate, which is variable. Each data frame has four components:

— Start bit ("0")

— 8 data bits (LSB first)

— Programmable 9th data bit

— Stop bit ("1")

### Mode 3 Transmit Procedure

1.  Select the baud rate generated by BRDATA.

2.  Select mode 3 operation (9-bit UART) by setting UARTCON bits 6 and 7 to '11B'. Also, select the 9th data bit to be transmitted by writing UARTCON.3 (TB8) to "0" or "1".

3.  Write transmission data to the shift register, UDATA (F7H, set 1, bank 0), to start the transmit operation.

### Mode 3 Receive Procedure

1.  Select the baud rate to be generated by BRDATA.

2.  Select mode 3 and set the RE (Receive Enable) bit in the UARTCON register to "1".

3.  The receive operation will be started when the signal at the RxD (P3.5) pin goes to low level.
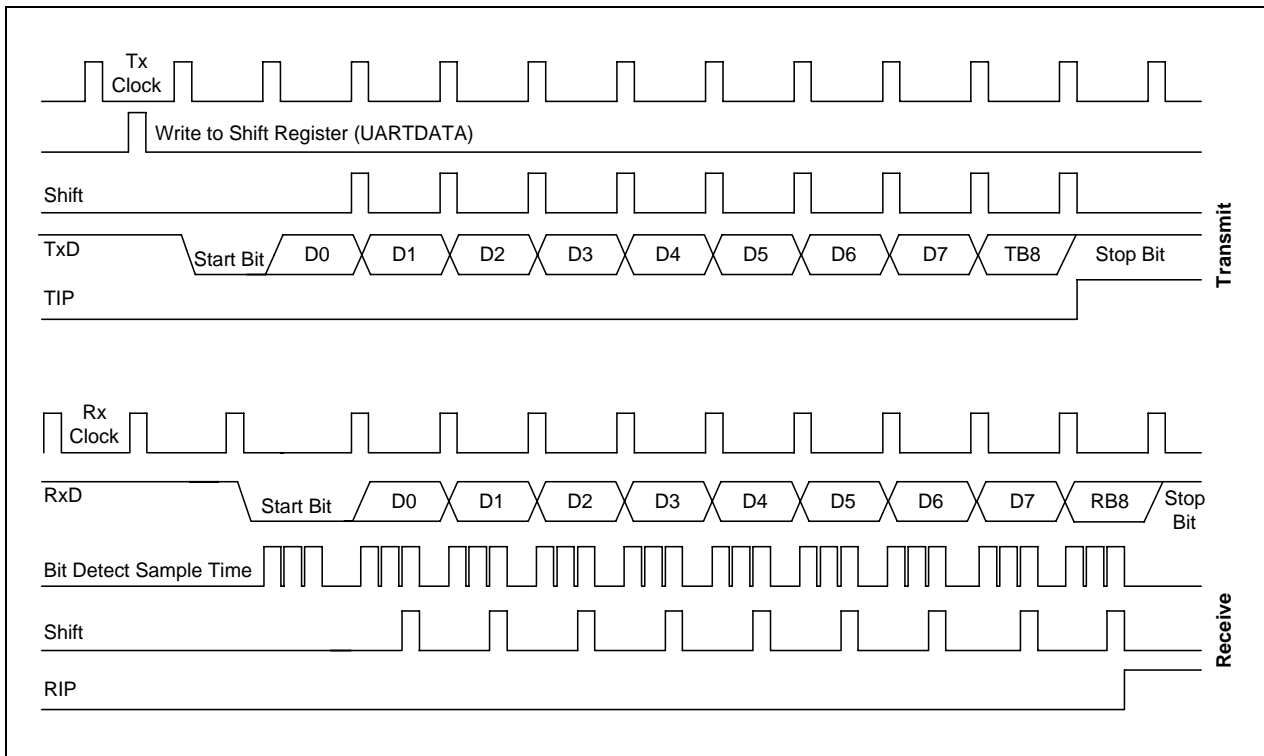


**Figure 17-9. Timing Diagram for Serial Port Mode 3 Operation**

## SERIAL COMMUNICATION FOR MULTIPROCESSOR CONFIGURATIONS

The S3C8-series multiprocessor communication features lets a "master" S3C828B/F828B/C8289/F8289/C8285/F8285 send a multiple-frame serial message to a "slave" device in a multi- S3C828B/F828B/C8289/F8289/C8285/F8285 configuration. It does this without interrupting other slave devices that may be on the same serial line.

This feature can be used only in UART modes 2 or 3. In these modes 2 and 3, 9 data bits are received. The 9th bit value is written to RB8 (UARTCON.2). The data receive operation is concluded with a stop bit. You can program this function so that when the stop bit is received, the serial interrupt will be generated only if RB8 = "1".

To enable this feature, you set the MCE bit in the UARTCON register. When the MCE bit is "1", serial data frames that are received with the 9th bit = "0" do not generate an interrupt. In this case, the 9th bit simply separates the address from the serial data.

### Sample Protocol for Master/Slave Interaction

When the master device wants to transmit a block of data to one of several slaves on a serial line, it first sends out an address byte to identify the target slave. Note that in this case, an address byte differs from a data byte: In an address byte, the 9th bit is "1" and in a data byte, it is "0".

The address byte interrupts all slaves so that each slave can examine the received byte and see if it is being addressed. The addressed slave then clears its MCE bit and prepares to receive incoming data bytes.

The MCE bits of slaves that were not addressed remain set, and they continue operating normally while ignoring the incoming data bytes.

While the MCE bit setting has no effect in mode 0, it can be used in mode 1 to check the validity of the stop bit. For mode 1 reception, if MCE is "1", the receive interrupt will be issue unless a valid stop bit is received.

**Setup Procedure for Multiprocessor Communications**

Follow these steps to configure multiprocessor communications:

1. Set all S3C828B/F828B/C8289/F8289/C8285/F8285 devices (masters and slaves) to UART mode 2 or 3.

2. Write the MCE bit of all the slave devices to "1".

3. The master device's transmission protocol is:

   — First byte: the address
     identifying the target
     slave device (9th bit = "1")

   — Next bytes: data
     (9th bit = "0")

4. When the target slave receives the first byte, all of the slaves are interrupted because the 9th data bit is "1". The targeted slave compares the address byte to its own address and then clears its MCE bit in order to receive incoming data. The other slaves continue operating normally.
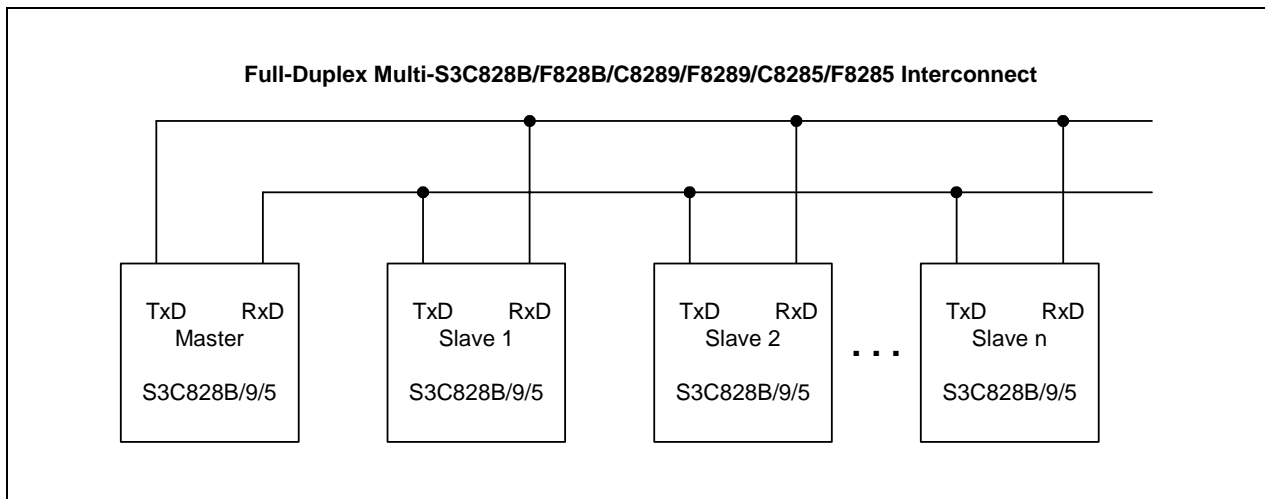


**Figure 17-10. Connection Example for Multiprocessor Serial Data Communications**

# 18 BATTERY LEVEL DETECTOR

## OVERVIEW

The S3C828B/F828B/C8289/F8289/C8285/F8285 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 $B_{LD}$ 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.
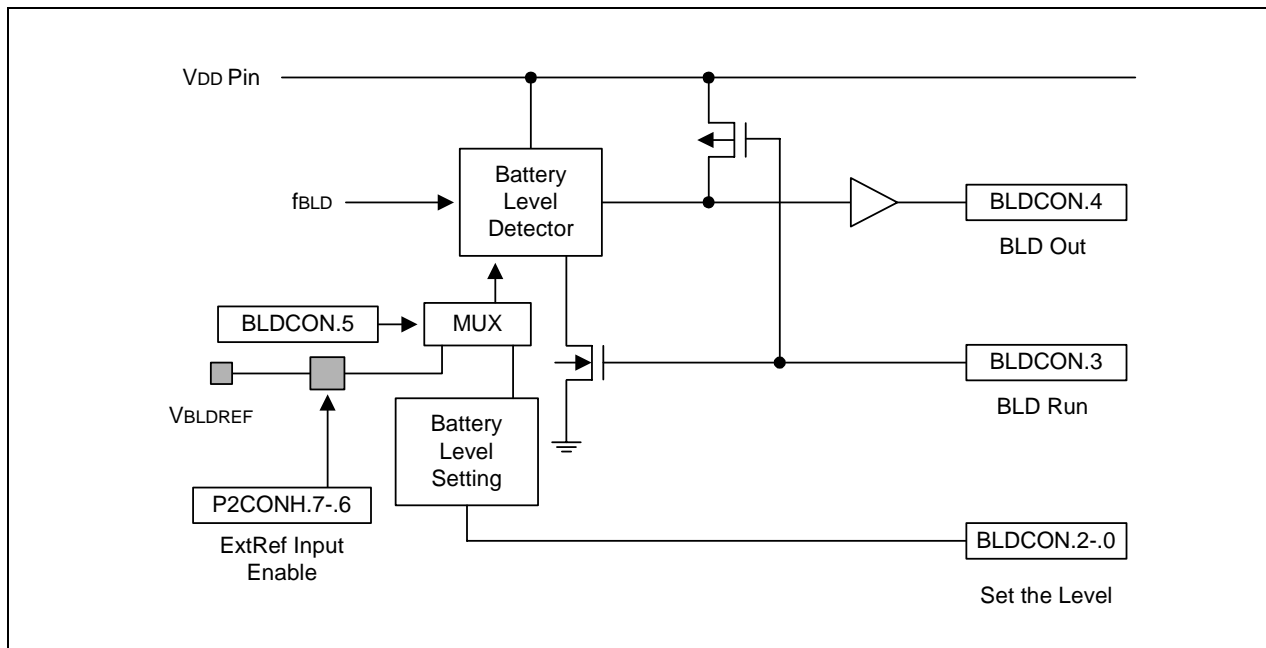


Figure 18-1. Block Diagram for Battery Level Detect

## BATTERY LEVEL DETECTOR CONTROL REGISTER (BLDCON)

The bit 3 of BLDCON controls to run or disable the operation of Battery level detect. Basically this $V_{BLD}$ is set as 2.2 V by system reset and it can be changed in 3 kinds voltages by selecting Battery Level Detect 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. Figure 18-2 shows specific $V_{BLD}$ of 3 levels.

Battery Level Detector Control Register (BLDCON)
D2H, Set 1, Bank 0, R/W

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

Not used

VIN source bit:
0 = Internal source
1 = External source

Detection voltage selection bits:
000 = ($V_{BLD}$ = 2.2V)
101 = ($V_{BLD}$ = 2.4V)
011 = ($V_{BLD}$ = 2.8V)

BLD Enable/Disable bit:
0 = Disable BLD
1 = Enable BLD

BLD Output bit:
0 = $V_{IN}$ > $V_{REF}$ (When BLD is enable)
1 = $V_{IN}$ < $V_{REF}$ (When BLD is enable)
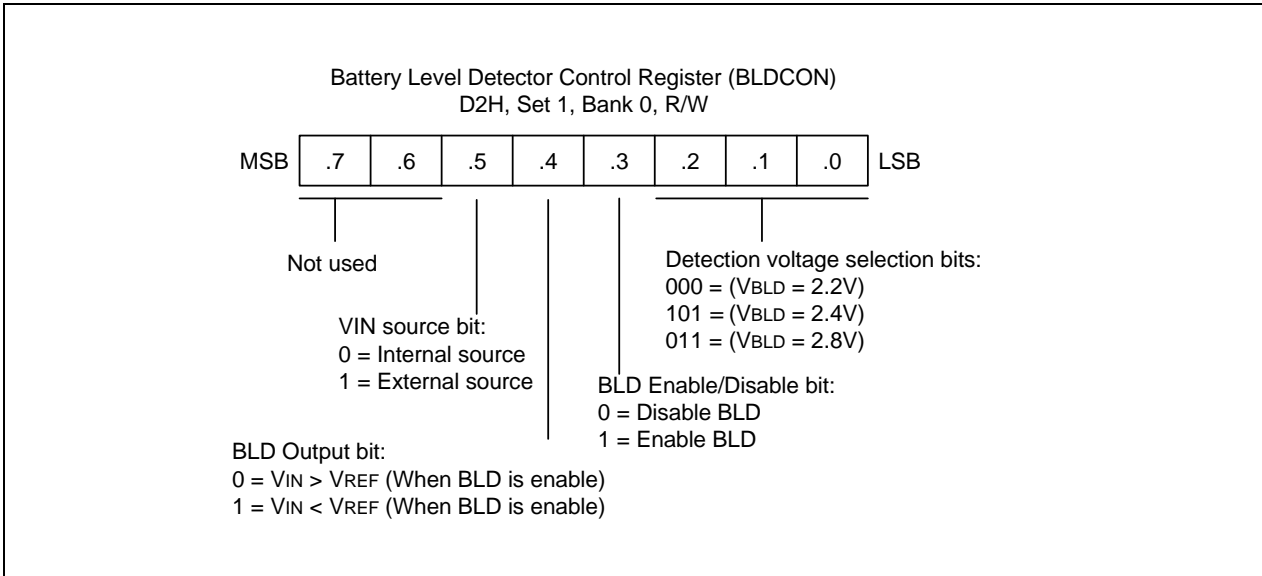
**Figure 18-2. Battery Level Detector Control Register (BLDCON)**

SAMSUNG
ELECTRONICS
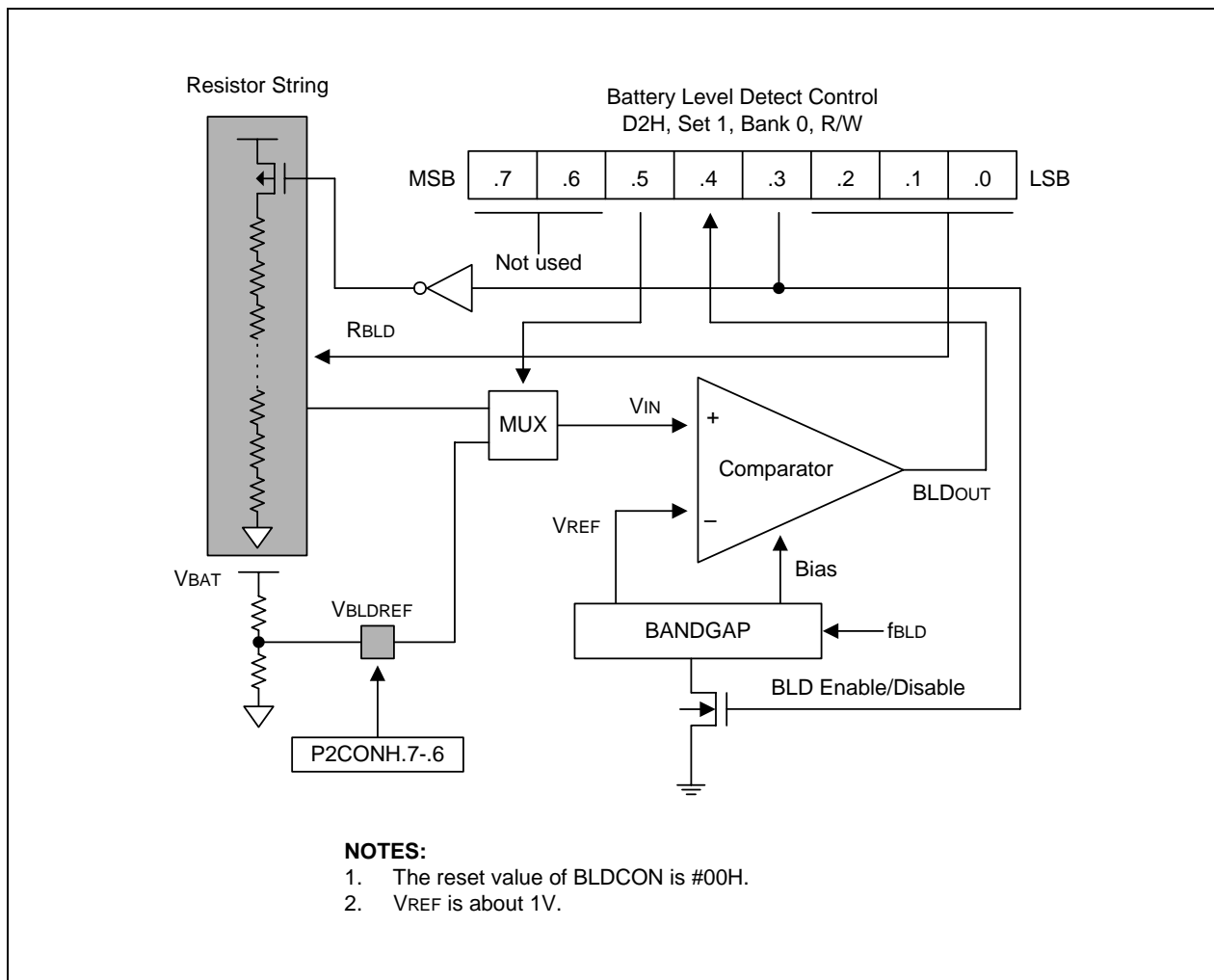
**Figure 18-3. Battery Level Detector Circuit and Block Diagram**

**Table 18-1. 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 |

# 19

# EMBEDDED FLASH MEMEORY INTERFACE

## OVERVIEW

This chapter is only for the S3F828B. The S3F828B has an on-chip 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 a flash memory area any time you want. The S3F828B's embedded 64K-byte memory has two operating features:

— User Program Mode: S3F828B Only

— Tool Program Mode: Refer to the chapter 22. S3F828B/F8289/F8285 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 a initial your code to a chip by using Tool Program mode by using a programming tool.

The S3F828B has the pumping circuit internally, therefore, 12.5V into $V_{PP}$ (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.

```
                    Flash Memory Control Register (FMCON)
                          D2H, Set 1, Bank 1, R/W

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


            Flash memory mode selection bits:        Not used
            0101 = Programming mode
            1010 = Sector erase mode                 Flash operation start bit:
            0110 = Hard lock mode                    0 = Operation stop bit
            Others = Not available                   1 = Operation start bit
                                                     (This bit will be cleared automatically
                                                      just after the corresponding
                              Sector erase status bit:   operation completed.)
                              0 = Success sector erase
                              1 = Fail sector erase
```

**Figure 19-1. Flash Memory Control Register (FMCON)**

The bit0 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.

**SAMSUNG**
**ELECTRONICS**

**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.

Flash Memory User Programming Enable Register (FMUSR)
FFH, Set 1, Bank 1, R/W

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

Flash memory user programming enable bits:
10100101: Enable user programming mode
Other values: Disable user programming mode
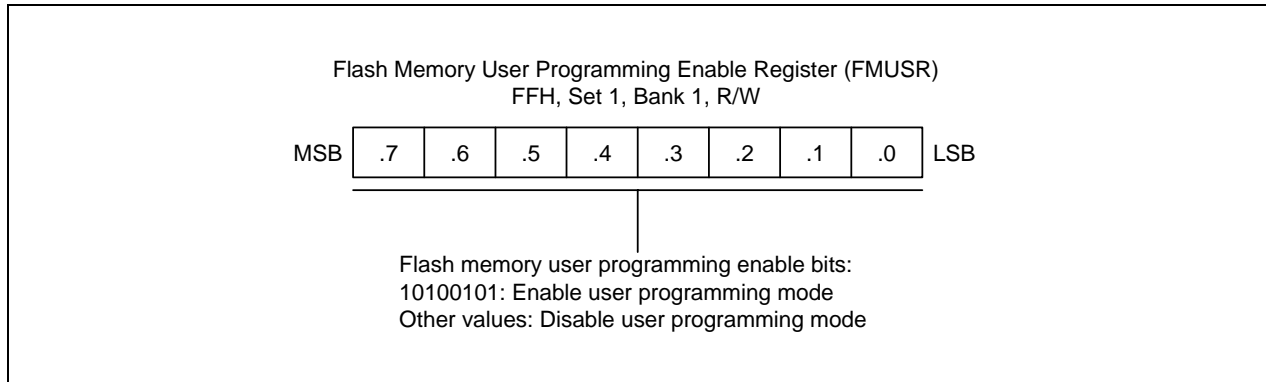
**Figure 19-2. Flash Memory User Programming Enable Register (FMUSR)**

## 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 S3F828B because it has 512 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. In case of other sectors, you must load sector address to FMSECH and FMSECL register according to the sector.



Flash Memory Sector Address Register (FMSECH)
D0H, Set 1, Bank 1, R/W

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

Flash Memory Setor Address (High Byte)

NOTE:    The high-byte flash memory sector address pointer
         value is the higher eight bits of the 16-bit pointer address.
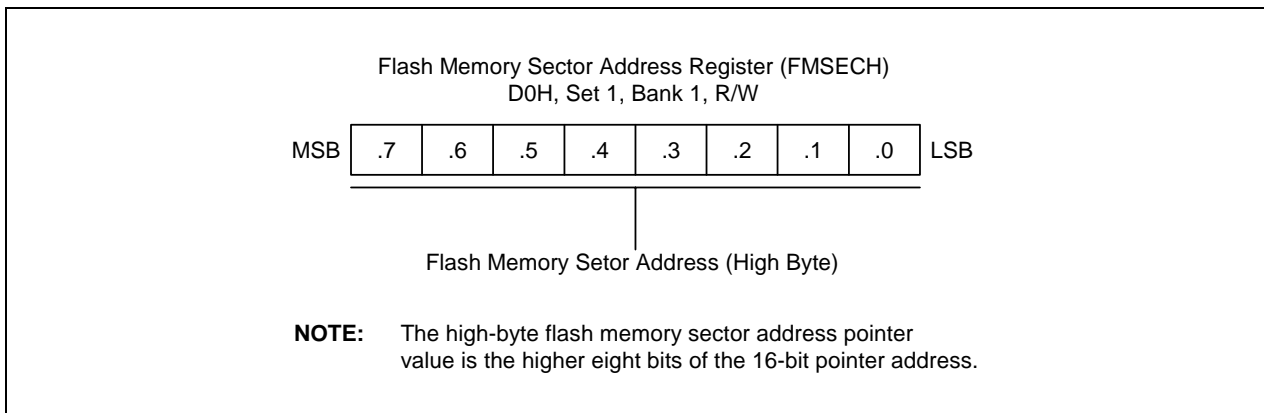
**Figure 19-3. Flash Memory Sector Address Register High Byte (FMSECH)**



Flash Memory Sector Address Register (FMSECL)
D1H, Set 1, Bank 1, R/W

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

Don't care

Flash Memory Sector Address (Low Byte)

NOTE:    The low-byte flash memory sector address pointer
         value is the lower eight bits of the 16-bit pointer address.
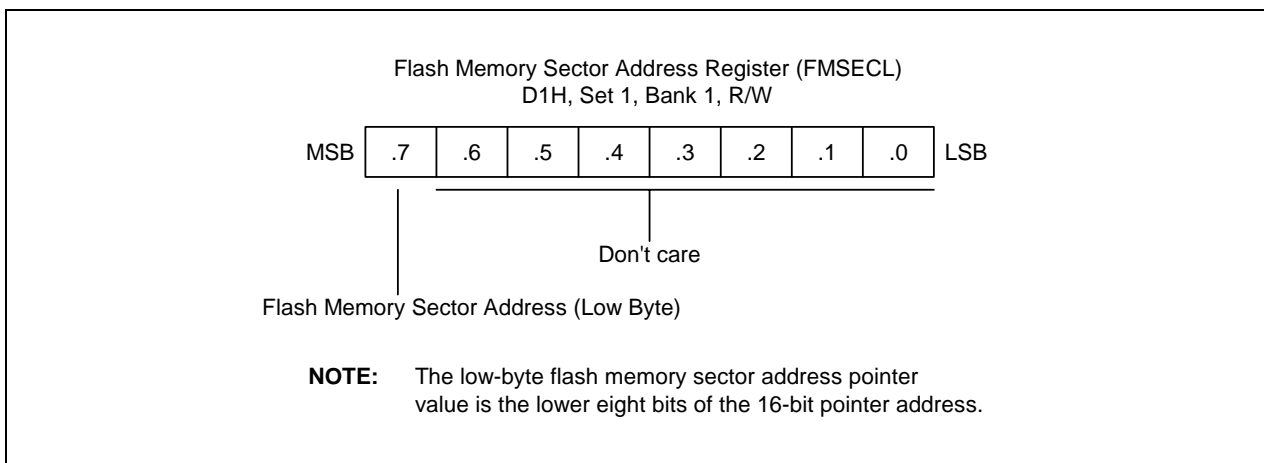
**Figure 19-4. Flash Memory Sector Address Register Low Byte (FMSECL)**

SAMSUNG
ELECTRONICS

## ISP™ (ON-BOARD PROGRAMMING) SECTOR

ISP™ sectors located in program memory area can store On Board Program software (Boot program code for upgrading application code by interfacing with I/O port pin). The ISP™ 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.
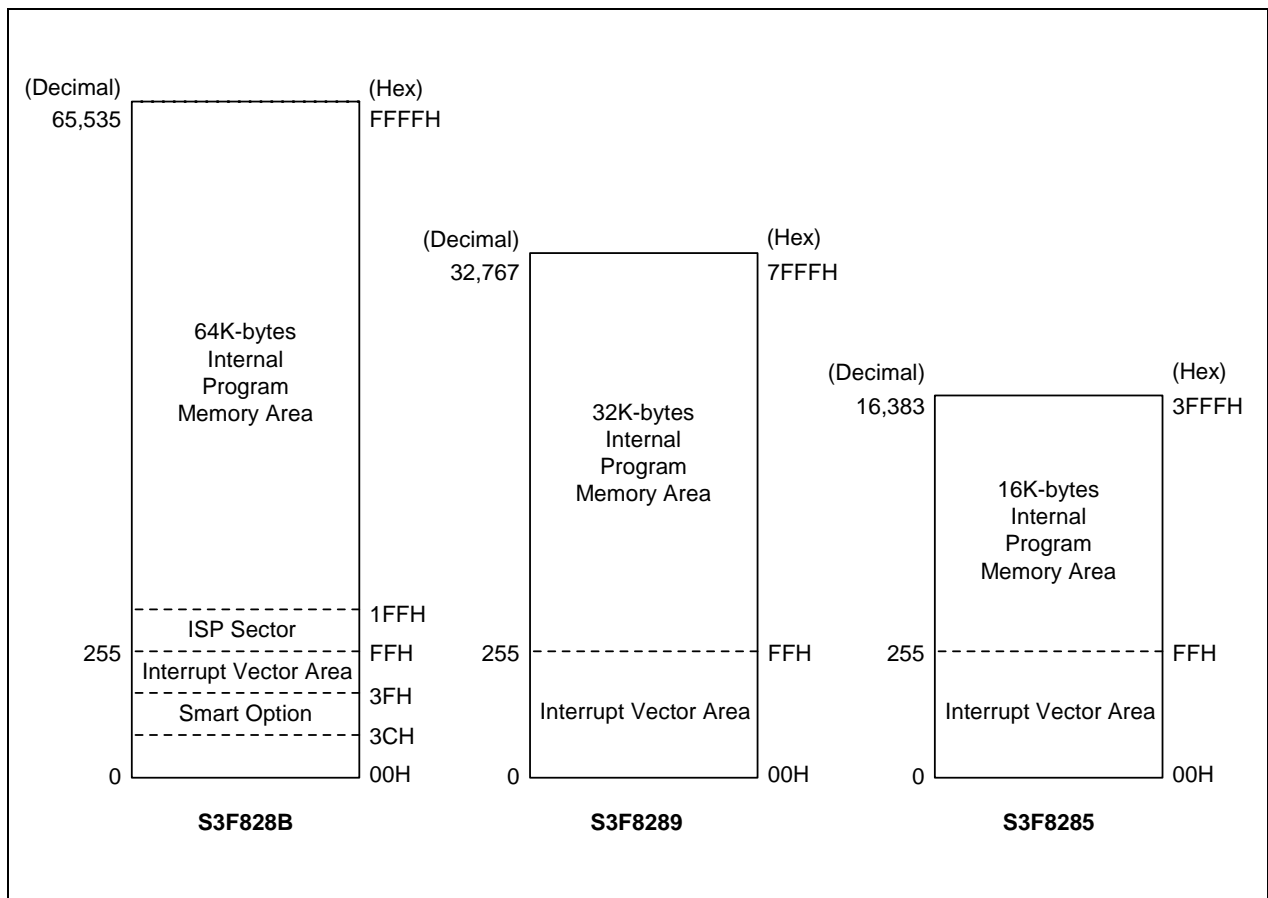


**Figure 19-5. Program Memory Address Space**

**Table 19-2. ISP Sector Size**

| Smart Option(003CH) 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 (003CH.2 – 003CH.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 19-3 by setting the ISP Reset Vector Address Selection bits.

**Table 19-3. Reset Vector Address**

| Smart Option (003CH) 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 (003CH.7 – 003CH.5) is not dependent of the selection of ISP sector size by Smart Option (003CH.2 – 003CH.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 S3F828B is divided into 512 sectors for unit of erase and programming, respectively. 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).
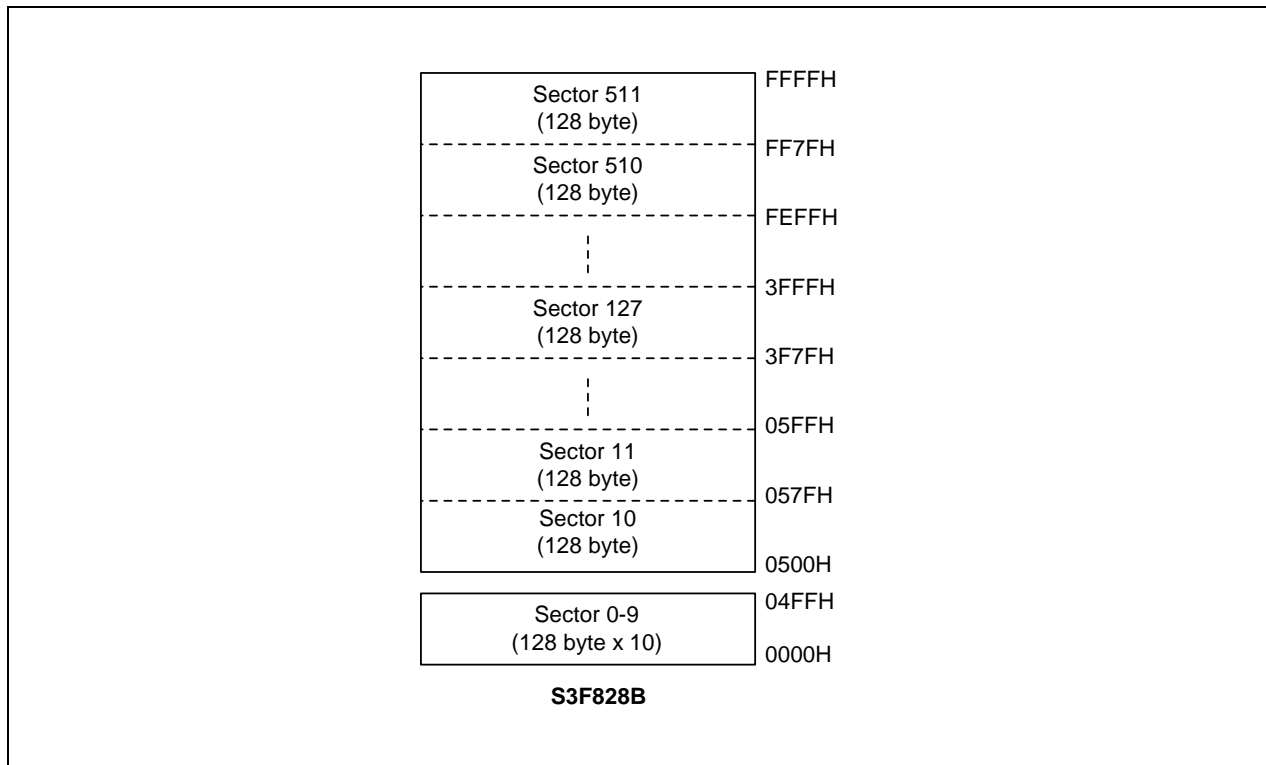


**Figure 19-6. Sector Configurations in User Program Mode**

**The Sector ERASE program 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.    Set Flash Memory Control Register (FMCON) to "10100001B".

4.    Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".

5.    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)
              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
```

SAMSUNG
ELECTRONICS

## 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 Control Register (FMCON) to "01010000B".

4.  Set Flash Memory Sector Register (FMSECH, FMSECL) to sector value of write address.

5.  Load a transmission data into a working register.

6.  Load a flash memory upper address into upper register of pair working register.

7.  Load a flash memory lower address into lower register of pair working register.

8.  Load transmission data to flash memory location area on 'LDC' instruction by indirectly addressing mode

9.  Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".


☞  **PROGRAMMING TIP — Program**


```
              •
              •
              SB1
              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
              LD        FMUSR,#0A5H          ; User Program mode enable
              LD        FMCON,#01010000B     ; 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
```

## READING

The read operation of programming starts by 'LDC' instruction.

**The program 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
              •
              •
              •
              •
```

SAMSUNG
ELECTRONICS

## HARD LOCK PROTECTION

User can set Hard Lock Protection by write '0110' in FMCON7-4. 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 program procedure in User program Mode**

1.  Set Flash Memory User Programming Enable Register (FMUSR) to "10100101B".

2.  Set Flash Memory Control Register (FMCON) to "01100001B".

3.  Set Flash Memory User Programming Enable Register (FMUSR) to "00000000B".


☞ **PROGRAMMING TIP — Hard Lock Protection**


```
        •
        •

        SB1
        LD        FMUSR,#0A5H         ; User Program mode enable
        LD        FMCON,#01100001B    ; Hard Lock mode set & start
        NOP                           ; Dummy Instruction, This instruction must be needed
        LD        FMUSR,#0            ; User Program mode disable

        •
        •
```

# 20 ELECTRICAL DATA

## OVERVIEW

In this chapter, S3C828B/F828B/C8289/F8289/C8285/F8285 electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

— Absolute maximum ratings

— Input/output capacitance

— D.C. electrical characteristics

— A.C. electrical characteristics

— Oscillation characteristics

— Oscillation stabilization time

— Data retention supply voltage in stop mode

— LVR timing characteristics

— BLD electrical characteristics

— Serial I/O timing characteristics

— A/D converter electrical characteristics

— UART timing characteristics

— Internal Flash ROM electrical characteristics

— Operating voltage range

**SAMSUNG**
**ELECTRONICS**

**Table 20-1. Absolute Maximum Ratings**

($T_A$= 25 $^\circ$C)

| Parameter | Symbol | Conditions | Rating | Unit |
|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | – | $-$ 0.3 to + 4.6 | V |
| Input voltage | $V_I$ | Ports 0-8 | $-$ 0.3 to $V_{DD}$ + 0.3 | |
| Output voltage | $V_O$ | – | $-$ 0.3 to $V_{DD}$ + 0.3 | |
| Output current high | $I_{OH}$ | One I/O pin active | $-$ 15 | mA |
| | | All I/O pins active | $-$ 60 | |
| Output current low | $I_{OL}$ | One I/O pin active | + 30 | |
| | | Total pin current for ports | + 100 | |
| Operating temperature | $T_A$ | – | $-$ 25 to + 85 | $^\circ$C |
| Storage temperature | $T_{STG}$ | – | $-$ 65 to + 150 | |

**Table 20-2. D.C. Electrical Characteristics**

($T_A$ = $-$25 $^\circ$C to + 85 $^\circ$C, $V_{DD}$ = 2.0 V to 3.6 V)

| Parameter | Symbol | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Operating voltage | $V_{DD}$ | $f_X$ = 0.4–4.2 MHz, $f_{XT}$ = 32.8kHz | 2.0 | – | 3.6 | V |
| | | $f_X$ = 0.4–10MHz | 2.7 | – | 3.6 | |
| | | $f_X$ = 0.4–11.1 MHz | 3.0 | – | 3.6 | |
| Input high voltage | $V_{IH1}$ | All input pins except $V_{IH2}$, $V_{IH3}$ | $0.7V_{DD}$ | – | $V_{DD}$ | |
| | $V_{IH2}$ | Ports0–1, nRESET | $0.8V_{DD}$ | | $V_{DD}$ | |
| | $V_{IH3}$ | $X_{IN}$, $X_{OUT}$ and $XT_{IN}$, $XT_{OUT}$ | $V_{DD}$–0.1 | | $V_{DD}$ | |
| Input low voltage | $V_{IL1}$ | All input pins except $V_{IL2}$,$V_{IL3}$ | – | – | $0.3V_{DD}$ | |
| | $V_{IL2}$ | Ports0–1, nRESET | | | $0.2V_{DD}$ | |
| | $V_{IL3}$ | $X_{IN}$, $X_{OUT}$ and $XT_{IN}$, $XT_{OUT}$ | | | 0.1 | |

SAMSUNG
ELECTRONICS

**Table 20-2. D.C. Electrical Characteristics (Continued)**

($T_A$ = −25 $^{\circ}$C to + 85 $^{\circ}$C, $V_{DD}$ = 2.0 V to 3.6 V)

| Parameter | Symbol | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Output high voltage | $V_{OH}$ | $V_{DD}$ = 2.7V to 3.6V<br>$I_{OH}$ = −1 mA<br>All output pins | $V_{DD}$−1.0 | – | – | V |
| Output low voltage | $V_{OL1}$ | $V_{DD}$ = 2.7V to 3.6V<br>$I_{OL}$ = 15 mA<br>Ports1−2 | – | – | 1.0 | |
| | $V_{OL2}$ | $V_{DD}$ = 2.7V to 3.6V<br>$I_{OL}$ = 10 mA<br>All output ports except $V_{OL1}$ | – | – | 1.0 | |
| Input high leakage current | $I_{LIH1}$ | $V_{IN}$ = $V_{DD}$<br>All input pins except $I_{LIH2}$ | – | – | 3 | uA |
| | $I_{LIH2}$ | $V_{IN}$ = $V_{DD}$<br>$X_{IN}$, $X_{OUT}$, $XT_{IN}$, $XT_{OUT}$ | | | 20 | |
| Input low leakage current | $I_{LIL1}$ | $V_{IN}$ = 0 V<br>All input pins except for nRESET, $I_{LIL2}$ | – | – | −3 | |
| | $I_{LIL2}$ | $V_{IN}$ = 0 V<br>$X_{IN}$, $X_{OUT}$, $XT_{IN}$, $XT_{OUT}$ | | | −20 | |
| Output high leakage current | $I_{LOH}$ | $V_{OUT}$ = $V_{DD}$<br>All output pins | – | – | 3 | |
| Output low leakage current | $I_{LOL}$ | $V_{OUT}$ = 0 V<br>All output pins | – | – | −3 | |
| LCD voltage dividing resistor | $R_{LCD}$ | $T_A$ = 25 $^{\circ}$C | 25 | 50 | 80 | kΩ |
| Oscillator feed back resistors | $R_{OSC1}$ | $V_{DD}$ = 3 V, $T_A$=25 $^{\circ}$C<br>$X_{IN}$ = $V_{DD}$, $X_{OUT}$ = 0V | 600 | 1600 | 3000 | |
| | $R_{OSC2}$ | $V_{DD}$ = 3 V, $T_A$=25 $^{\circ}$C<br>$XT_{IN}$ = $V_{DD}$, $XT_{OUT}$ = 0V | 2000 | 4000 | 8000 | |
| Pull-up resistor | $R_{L1}$ | $V_{IN}$ = 0 V; $V_{DD}$ = 3 V<br>Ports 0−8, $T_A$ = 25$^{\circ}$C | 40 | 70 | 100 | |
| | $R_{L2}$ | $V_{IN}$ = 0 V; $V_{DD}$ = 3 V<br>$T_A$ = 25 $^{\circ}$C, nRESET | 220 | 360 | 500 | |

SAMSUNG
ELECTRONICS

**20-3**

**Table 20-2. D.C. Electrical Characteristics (Continued)**

($T_A$ = −25 $^\circ$C to + 85 $^\circ$C, $V_{DD}$ = 2.0 V to 3.6 V)

| Parameter | Symbol | Conditions | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| Middle output voltage [1] | $V_{LC1}$ | $V_{DD}$ = 2.7V to 3.6V, 1/4 bias LCD clock = 0Hz, $V_{LC0}$ = $V_{DD}$ | | $0.75V_{DD}$−0.2 | $0.75V_{DD}$ | $0.75V_{DD}$+0.2 | V |
| | $V_{LC2}$ | | | $0.5V_{DD}$−0.2 | $0.5V_{DD}$ | $0.5V_{DD}$+0.2 | |
| | $V_{LC3}$ | | | $0.25V_{DD}$−0.2 | $0.25V_{DD}$ | $0.25V_{DD}$+0.2 | |
| $|V_{LCD}$ − COMi$|$ Voltage drop (i = 0 − 7) | $V_{DC}$ | −15 µA per common pin | | – | – | 120 | mV |
| $|V_{LCD}$ − SEGx$|$ Voltage drop (x = 0 − 34) | $V_{DS}$ | −15 µA per common pin | | – | – | 120 | |
| Supply current [2] | $I_{DD1}$ [3] | Run mode: $V_{DD}$ = 3.3V ± 0.3V Crystal oscillator C1 = C2 = 22pF | 11.1 MHz | – | 4.0 | 8.0 | mA |
| | | | 4.0 MHz | – | 1.8 | 3.6 | |
| | $I_{DD2}$ [3] | Idle mode: $V_{DD}$ = 3.3V ± 0.3V Crystal oscillator C1 = C2 = 22pF | 11.1 MHz | – | 1.0 | 2.0 | |
| | | | 4.0 MHz | | 0.5 | 1.0 | |
| | $I_{DD3}$ [4] | Run mode: $V_{DD}$ = 3.3V ± 0.3V, $T_A$ = 25$^\circ$C, OSCON.7 = 1 32kHz crystal oscillator | | – | 14.0 | 28.0 | µA |
| | $I_{DD4}$ [4] | Idle mode: $V_{DD}$ = 3.3V ± 0.3V, $T_A$ = 25$^\circ$C, OSCON.7 = 1 32kHz crystal oscillator | | – | 2.0 | 4.0 | |
| | $I_{DD5}$ [5] | Stop mode: $V_{DD}$ = 3.3V ± 0.3V | $T_A$=25$^\circ$C | – | 0.2 | 2.0 | |
| | | | $T_A$=-25$^\circ$C to 85$^\circ$C | – | – | 10 | |

**NOTES:**
1. It is middle output voltage when the $V_{DD}$ and $V_{LC0}$ pin are connected.
2. Supply current does not include current drawn through internal pull-up resistors, LCD voltage dividing resistors, the LVR block, and external output current loads.
3. $I_{DD1}$ and $I_{DD2}$ include a power consumption of subsystem oscillator.
4. $I_{DD3}$ and $I_{DD4}$ are the current when the main system clock oscillation stop and the subsystem clock is used. (OSCCON.7 = 1)
5. $I_{DD5}$ is the current when the main and subsystem clock oscillation stops.
6. 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

**Table 20-3. A.C. Electrical Characteristics**

($T_A$ = $-25\ ^\circ$C to + 85 $^\circ$C, $V_{DD}$ = 2.0 V to 3.6 V)

| Parameter | Symbol | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Interrupt input high, low width (P0.0–P0.7) | $t_{INTH}$, $t_{INTL}$ | All interrupt, $V_{DD}$ = 3 V | 500 | 700 | – | ns |
| nRESET input low width | $t_{RSL}$ | $V_{DD}$ = 3 V | 10 | – | – | μs |
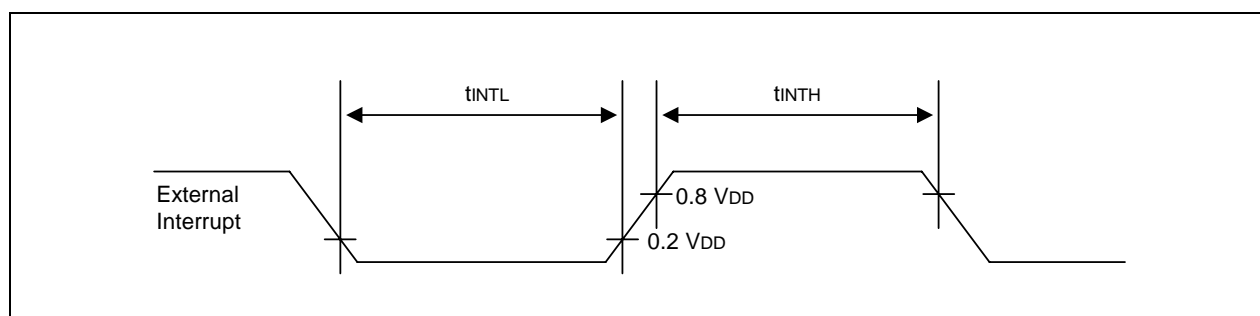


**Figure 20-1. Input Timing for External Interrupts**
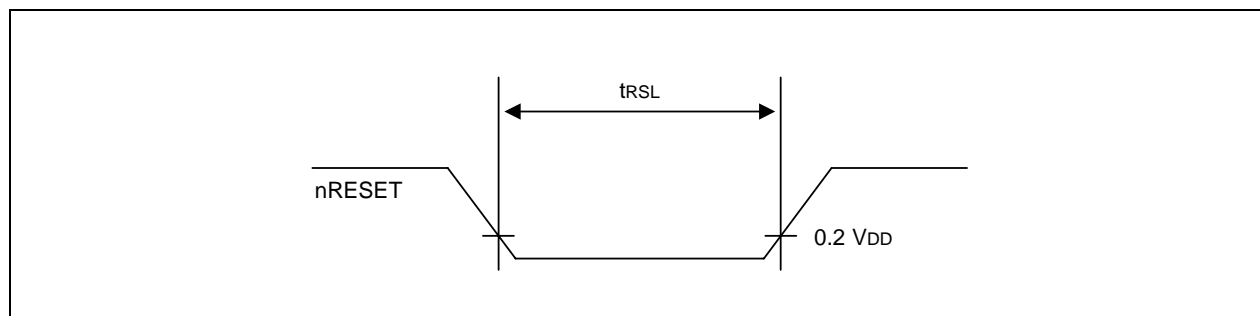


**Figure 20-2. Input Timing for nRESET**

**SAMSUNG**

**ELECTRONICS**

**Table 20-4. Input/Output Capacitance**

($T_A$ = −25 $°$C to + 85 $°$C, $V_{DD}$ = 0 V )

| Parameter | Symbol | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | $C_{IN}$ | f = 1 MHz; unmeasured pins are returned to $V_{SS}$ | – | – | 10 | pF |
| Output capacitance | $C_{OUT}$ | | | | | |
| I/O capacitance | $C_{IO}$ | | | | | |

**Table 20-5. Data Retention Supply Voltage in Stop Mode**

($T_A$ = −25 $°$C to + 85 $°$C)

| Parameter | Symbol | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Data retention supply voltage | $V_{DDDR}$ | | 2.0 | – | 3.6 | V |
| Data retention supply current | $I_{DDDR}$ | $V_{DDDR}$ = 2V Stop mode, $T_A$ = 25 $°$C Disable LVR block | – | – | 1 | uA |
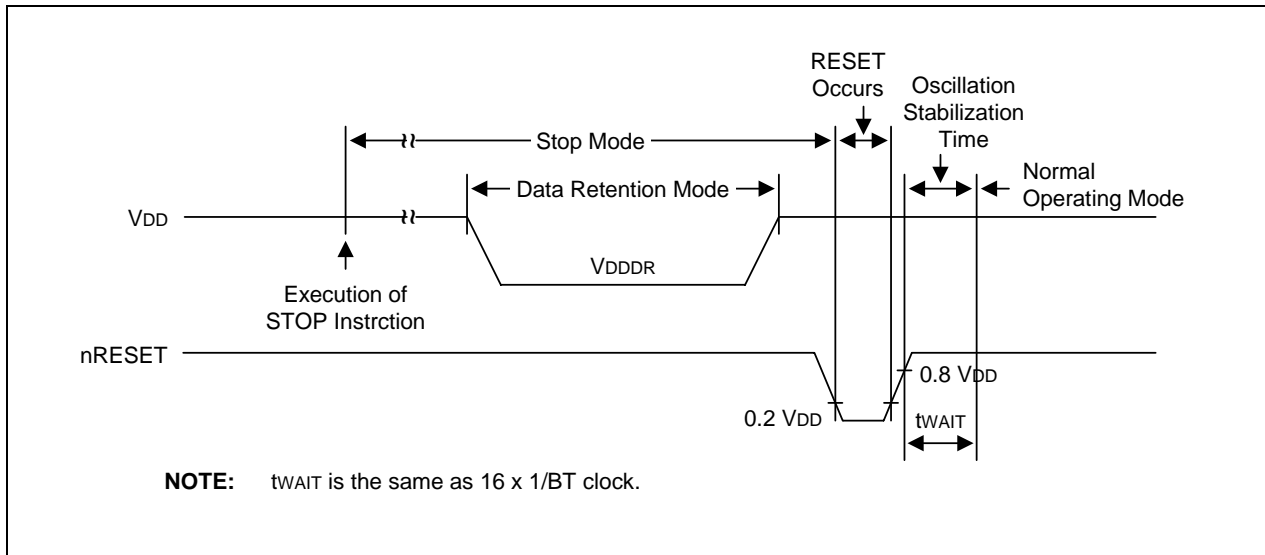


**Figure 20-3. Stop Mode Release Timing Initiated by RESET**
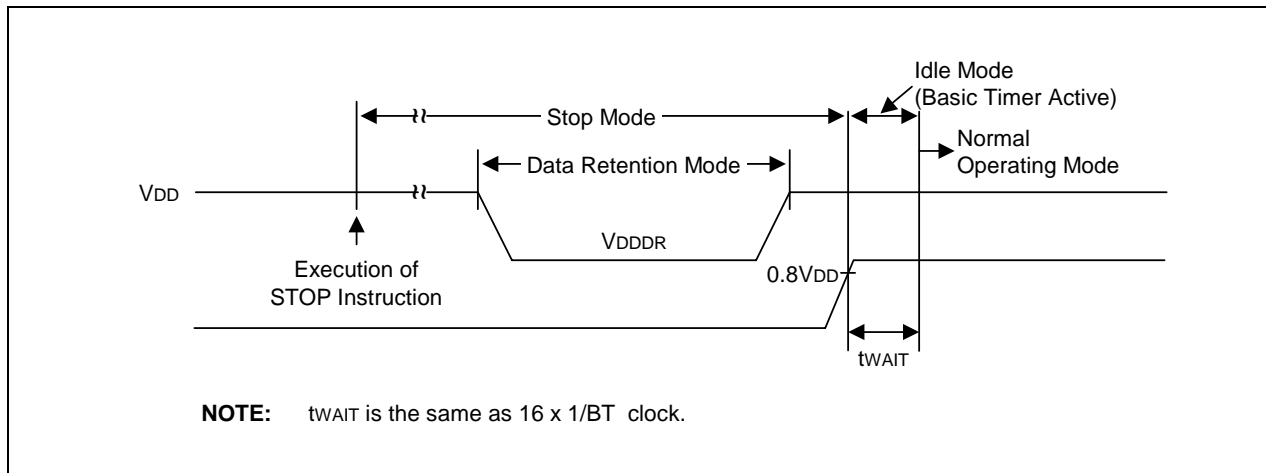
**Figure 20-4. Stop Mode Release Timing Initiated by Interrupts**

**Table 20-6. A/D Converter Electrical Characteristics**

($T_A$ = −25 °C to + 85 °C, $V_{DD}$ = 2.7 V to 3.6 V, $V_{SS}$ = 0 V)

| Parameter | Symbol | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Resolution | | – | – | 10 | – | bit |
| Total accuracy | | – | – | – | ±3 | LSB |
| Integral linearity error | ILE | $V_{DD}$ = 3.072 V | – | – | ±2 | |
| Differential linearity error | DLE | $V_{SS}$ = 0 V | | – | ±1 | |
| Offset error of top | EOT | CPU clock = 11.1 MHz | | ±1 | ±3 | |
| Offset error of bottom | EOB | | | ±1 | ±3 | |
| Conversion time [1] | $T_{CON}$ | 10-bit resolution 50 x fxx/4, fxx = 8 MHz | 25 | – | – | µS |
| Analog input voltage | $V_{IAN}$ | – | $V_{SS}$ | – | $AV_{REF}$ | V |
| Analog input impedance | $R_{AN}$ | – | 2 | 1000 | – | MΩ |
| Analog reference voltage | $AV_{REF}$ | – | 2.0 | – | $V_{DD}$ | V |
| Analog ground | $AV_{SS}$ | – | $V_{SS}$ | – | $V_{SS}$ + 0.3 | |
| Analog input current | $I_{ADIN}$ | $V_{DD}$ = 3.3 V | – | – | 5 | µA |
| Analog block current [2] | $I_{ADC}$ | $V_{DD}$ = 3.3 V | – | 0.5 | 1.5 | mA |
| | | $V_{DD}$ = 3.3 V When power down mode | | 100 | 500 | nA |

**NOTES**:
1.  'Conversion time' is the time required from the moment a conversion operation starts until it ends.
2.  $I_{ADC}$ is an operating current during A/D converter.

SAMSUNG
ELECTRONICS

**Table 20-7. Low Voltage Reset Electrical Characteristics**

($T_A$ = 25 °C)

| Parameter | Symbol | Test Condition | 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 | $\Delta V$ | – | – | 10 | 100 | mV |
| Current consumption | $I_{DDPR}$ | $V_{DD}$ = 3.3 V | – | 70 | 120 | µA |

**NOTE:** The current of LVR circuit is consumed when LVR is enabled by "Smart Option".
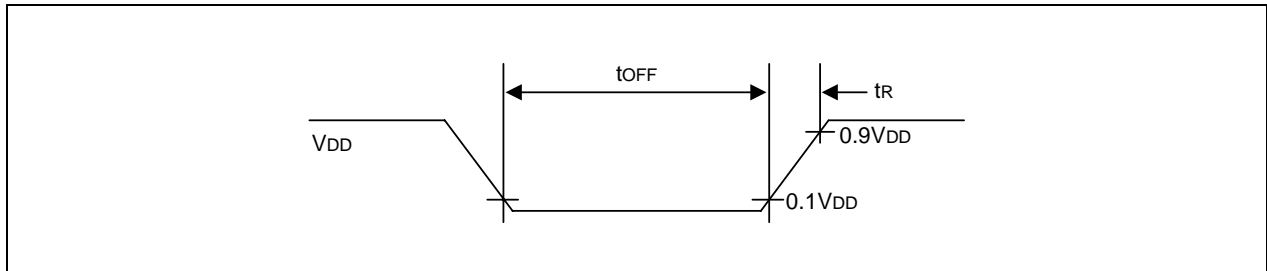


**Figure 20-5. LVR (Low Voltage Reset) Timing**

**Table 20-8. 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 | V |
| | | BLDCON.2–.0 = 101b | 2.15 | 2.4 | 2.65 | |
| | | BLDCON.2–.0 = 011b | 2.5 | 2.8 | 3.1 | |
| Hysteresis Voltage of BLD | $\Delta V$ | BLDCON.2–.0 = 000, 101, 011b | – | 10 | 100 | mV |
| Current Consumption | $I_{BLD}$ | $V_{DD}$ = 3.3 V | – | 70 | 120 | µA |
| | | $V_{DD}$ = 2.2 V | – | 50 | 100 | |
| BLD Circuit Response Time | $T_B$ | fw = 32.768 kHz | – | – | 1 | ms |

SAMSUNG
ELECTRONICS

**Table 20-9. Synchronous SIO 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 | |
| | | Internal SCK source | | | 250 | |
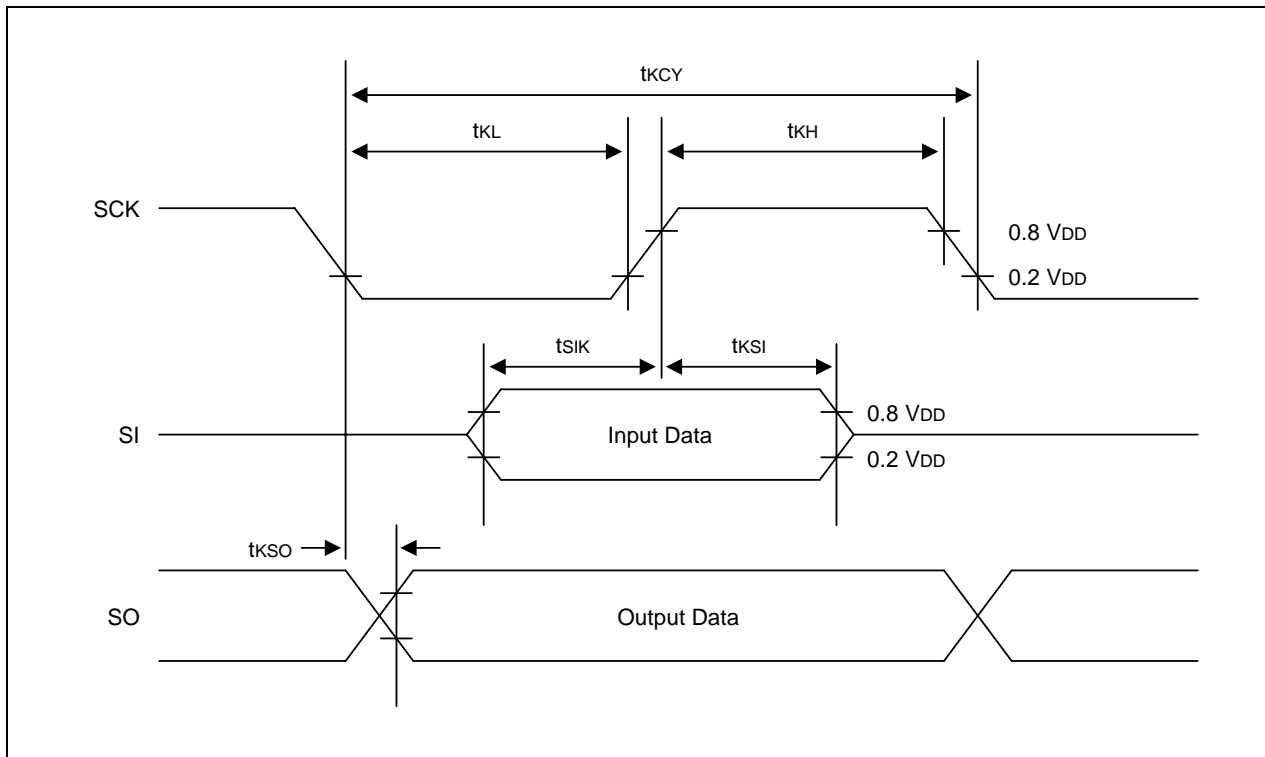


**Figure 20-6. Serial Data Transfer Timing**

SAMSUNG
ELECTRONICS

**Table 20-10. UART Timing Characteristics in Mode 0 (11.1MHz)**

($T_A = -25°C$ to $+ 85°C$, $V_{DD}$ = 2.0 V to 3.6 V, Load Capacitance = 80pF)

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Serial port clock cycle time | $t_{SCK}$ | 460 | $t_{CPU} \times 6$ | 620 | ns |
| Output data setup to clock rising edge | $t_{S1}$ | 220 | $t_{CPU} \times 5$ | – | |
| Clock rising edge to input data valid | $t_{S2}$ | – | – | 220 | |
| Output data hold after clock rising edge | $t_{H1}$ | $t_{CPU} - 50$ | $t_{CPU}$ | – | |
| Input data hold after clock rising edge | $t_{H2}$ | 0 | – | – | |
| Serial port clock High, Low level width | $t_{HIGH}, t_{LOW}$ | 180 | $t_{CPU} \times 3$ | 360 | |

**NOTES:**
1.  All timings are in nanoseconds (ns) and assume a 11.1-MHz CPU clock frequency.
2.  The unit $t_{CPU}$ means one CPU clock period.
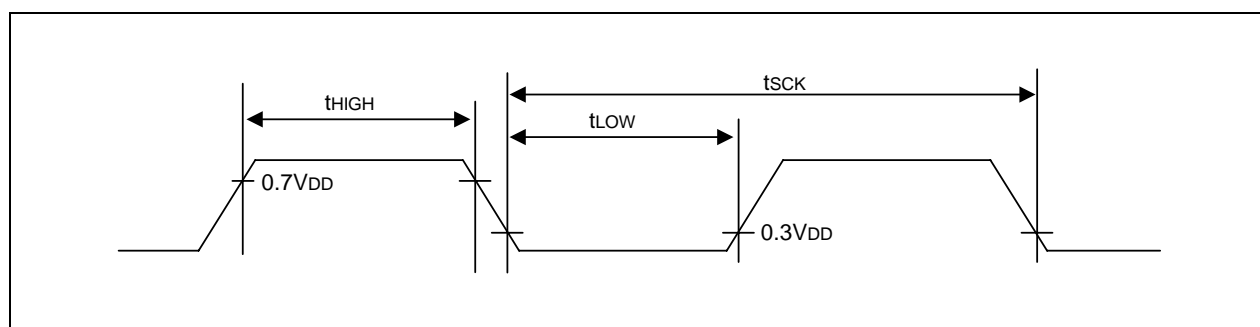


**Figure 20-7. Waveform for UART Timing Characteristics**

NOTE:   The symbols shown in this diagram are defined as follows:

$f_{SCK}$      Serial port clock cycle time
$t_{S1}$      Output data setup to clock rising edge
$t_{S2}$      Clock rising edge to input data valid
$t_{H1}$      Output data hold after clock rising edge
$t_{H2}$      Input data hold after clock rising edge
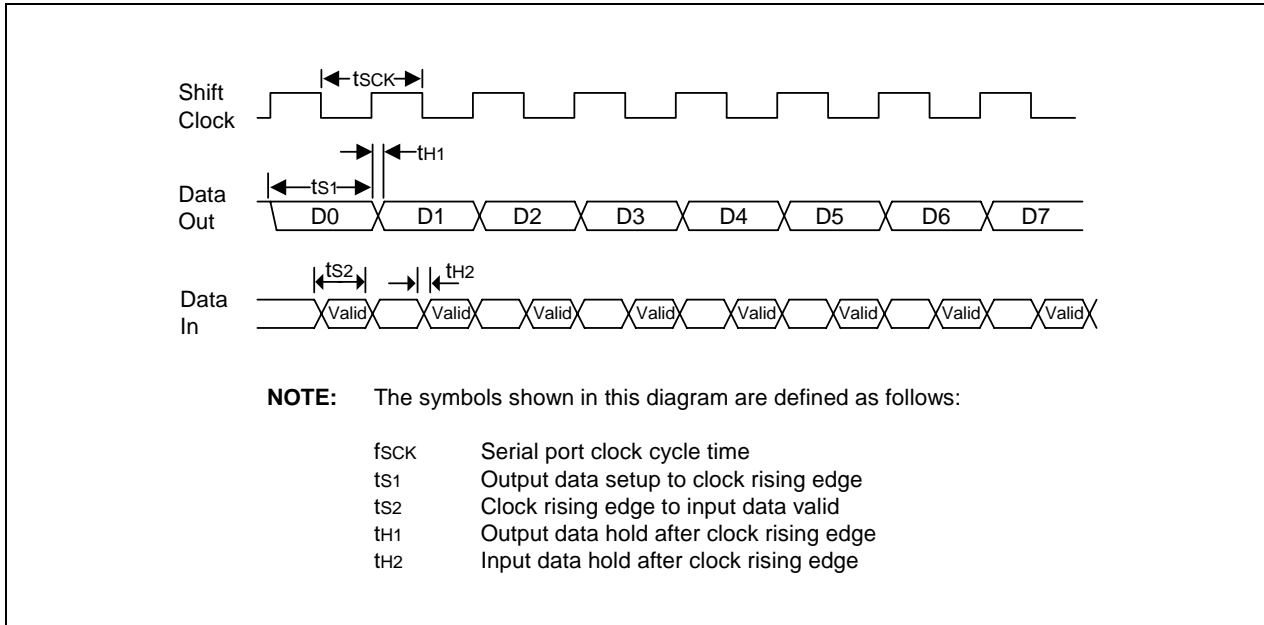
**Figure 20-8. Timing Waveform for the UART Module**

SAMSUNG
ELECTRONICS

**Table 20-11. Main Oscillator Characteristics**

$(T_A = -25\ {}^{\circ}C\ to + 85\ {}^{\circ}C)$
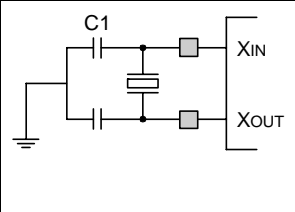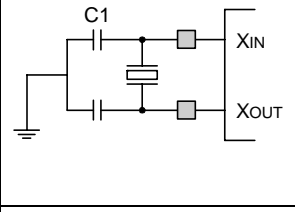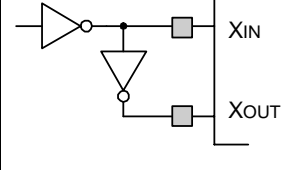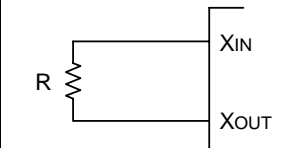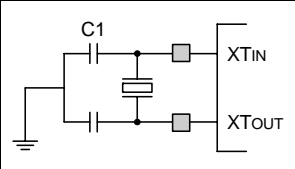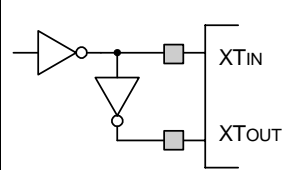
| Oscillator | Clock Configuration | Parameter | Test Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|---|
| Crystal |  | Main oscillation frequency | 3.0 V – 3.6 V | 0.4 | – | 11.1 | MHz |
| | | | 2.7 V – 3.6 V | 0.4 | – | 10 | |
| | | | 2.0 V – 3.6 V | 0.4 | – | 4.2 | |
| Ceramic Oscillator |  | Main oscillation frequency | 3.0 V – 3.6 V | 0.4 | – | 11.1 | |
| | | | 2.7 V – 3.6 V | 0.4 | – | 10 | |
| | | | 2.0 V – 3.6 V | 0.4 | – | 4.2 | |
| External Clock |  | $X_{IN}$ input frequency | 3.0 V – 3.6 V | 0.4 | – | 11.1 | |
| | | | 2.7 V – 3.6 V | 0.4 | – | 10 | |
| | | | 2.0 V – 3.6 V | 0.4 | – | 4.2 | |
| RC Oscillator |  | Frequency | 3.3 V | 0.4 | – | 1 | MHz |

**Table 20-12. Sub Oscillation Characteristics**

$(T_A = -25^{\circ}C\ to + 85^{\circ}C)$

| 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**

**20-13**

**Table 20-13. Main Oscillation Stabilization Time**

($T_A$ = −25 $^{\circ}$C to + 85 $^{\circ}$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 |
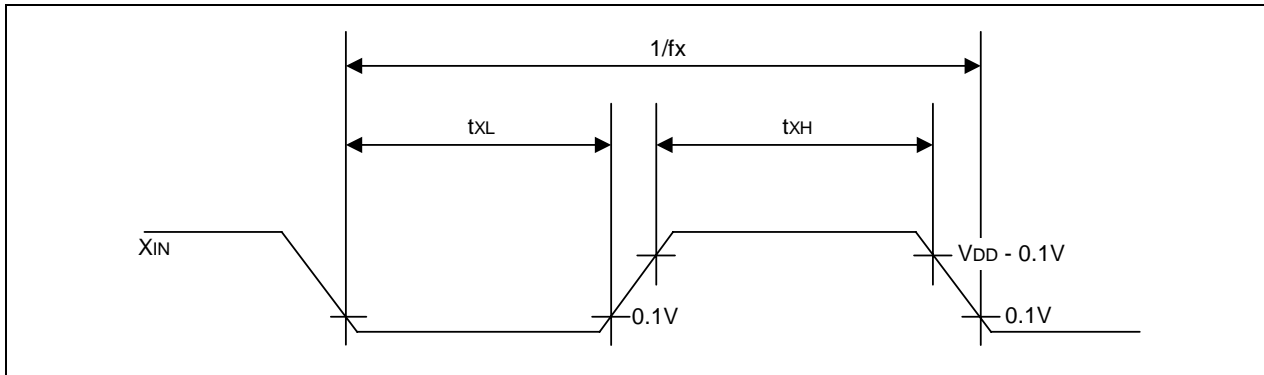


**Figure 20-9. Clock Timing Measurement at $X_{IN}$**

**Table 20-14. Sub Oscillation Stabilization Time**

($T_A$ = −25 $^{\circ}$C to + 85 $^{\circ}$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 |
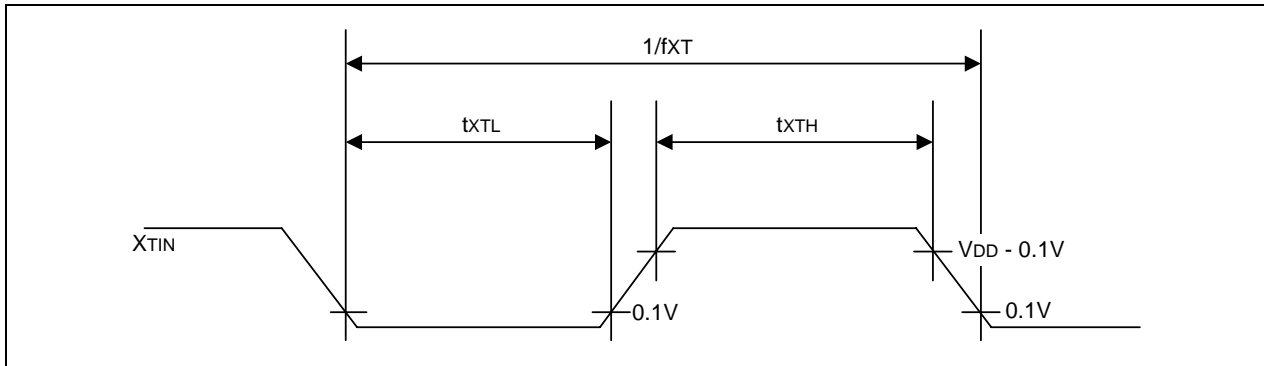


**Figure 20-10. Clock Timing Measurement at $XT_{IN}$**

SAMSUNG
ELECTRONICS
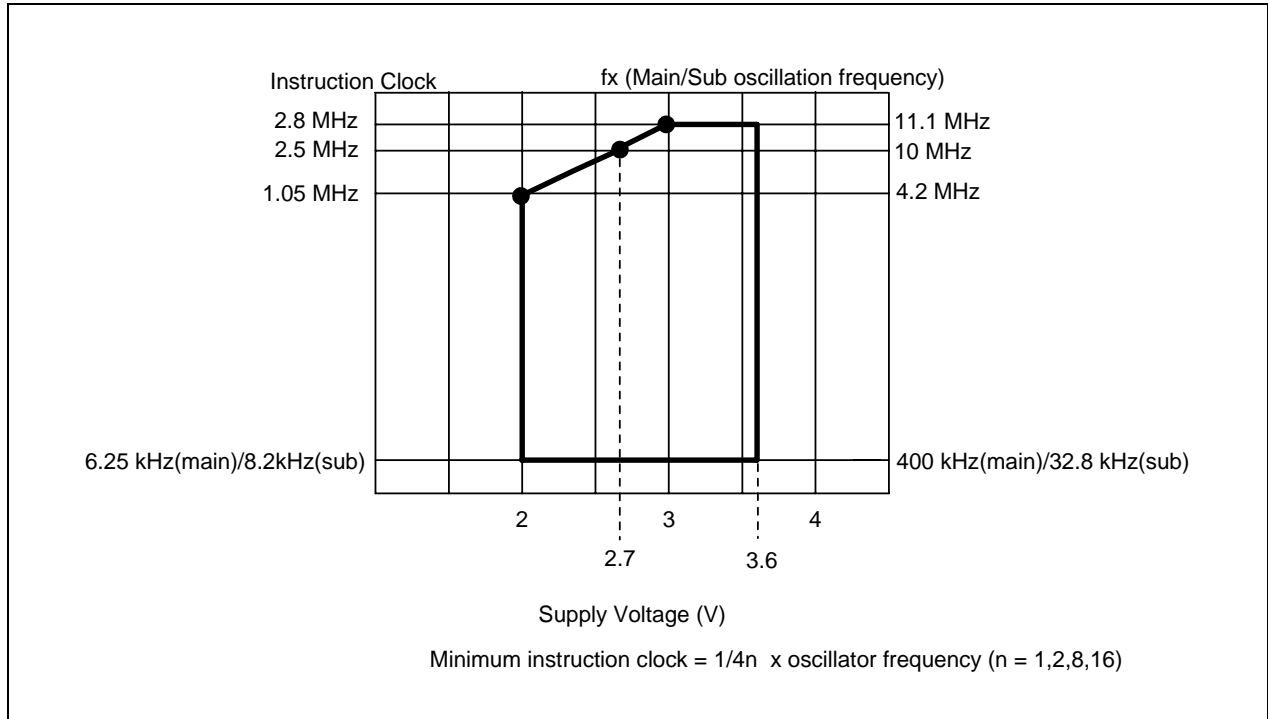
**Figure 20-11. Operating Voltage Range**

**Table 20-15. Internal Flash ROM Electrical Characteristics**

($T_A$ = −25 $^\circ$C to + 85 $^\circ$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 | $FN_{WE}$ | – | – | – | 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 64K 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(S3F828B) and 100 times for half flash (S3F8289/F8285).
5. The Chip Erasing is available in Tool Program Mode only.

# 21 MECHANICAL DATA

## OVERVIEW

The S3C828B/F828B/C8289/F8289/C8285/F8285 microcontroller is currently available in 80-pin-QFP/TQFP package.



**NOTE**:  Dimensions are in millimeters.
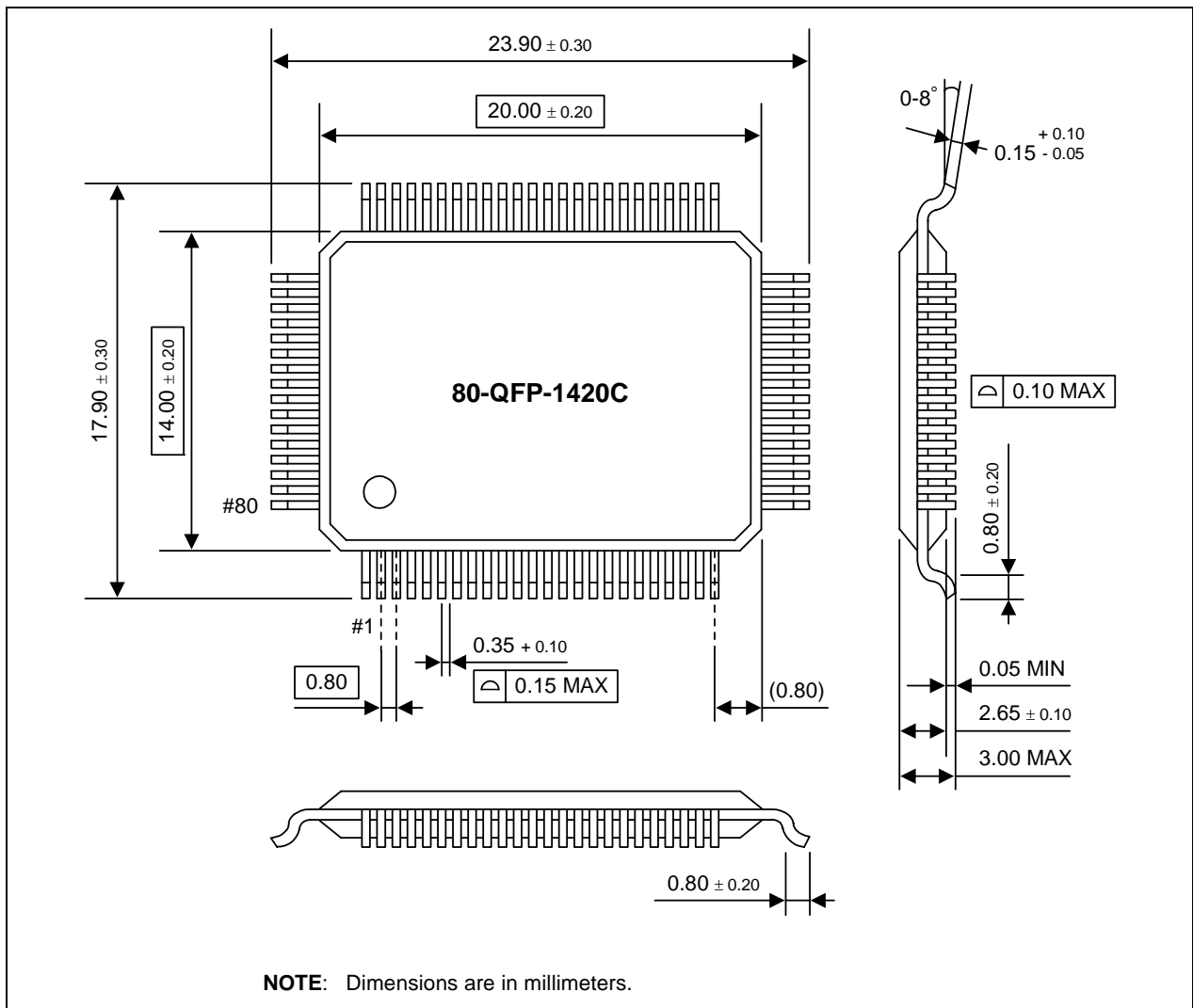
**Figure 21-1. Package Dimensions (80-QFP-1420C)**
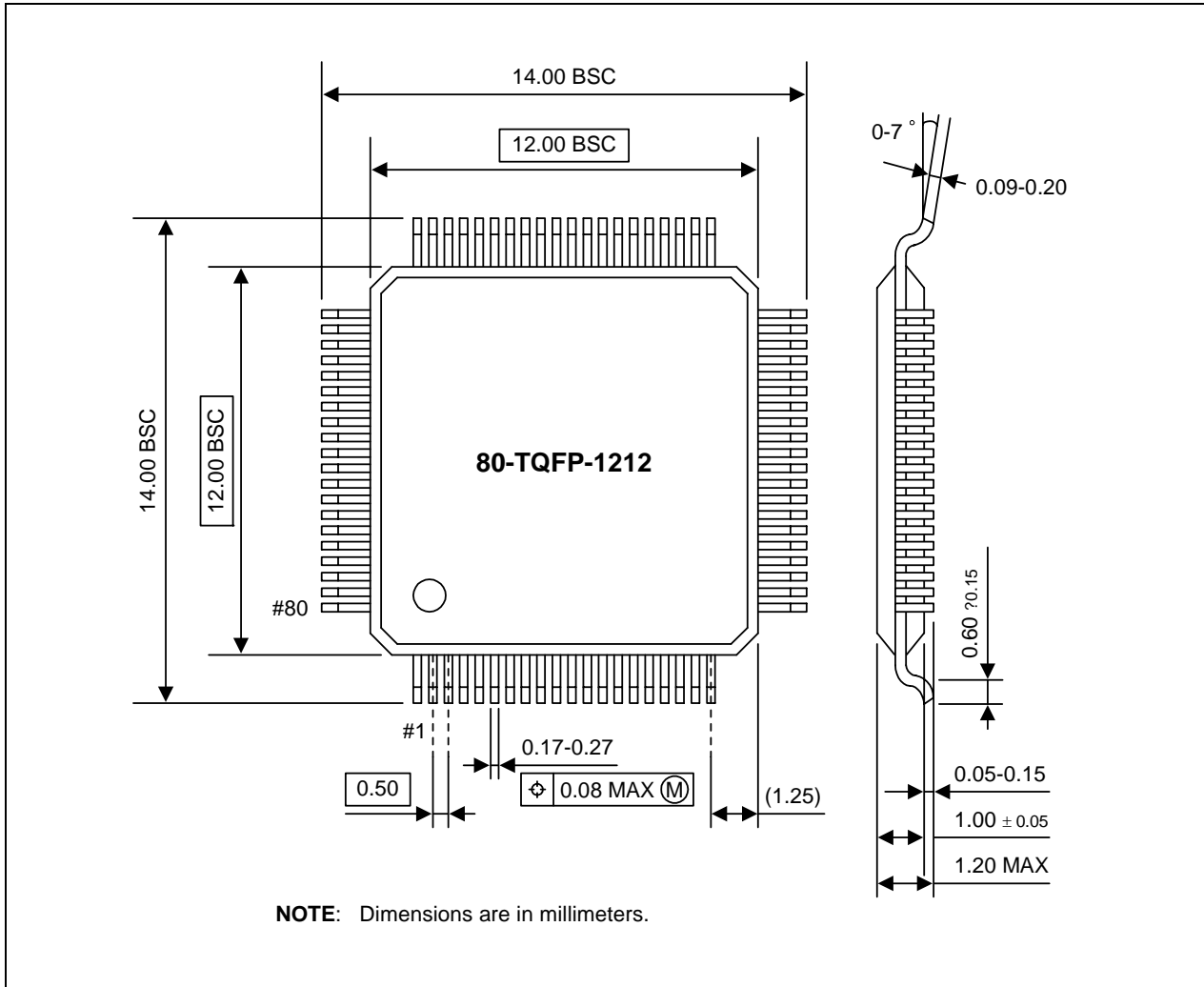
**Figure 21-2. Package Dimensions (80-TQFP-1212)**

# 22 S3F828B/F8289/F8285 FLASH MCU

## OVERVIEW

The S3F828B/F8289/F8285 single-chip CMOS microcontroller is the Flash MCU version of the S3C828B/C8289/C8285 microcontroller. It has an on-chip Flash MCU ROM instead of a masked ROM. The Flash ROM is accessed by serial data format.

The S3F828B/F8289/F8285 is fully compatible with the S3C828B/C8289/C8285, both in function and in pin configuration. Because of its simple programming requirements, the S3F828B/F8289/F8285 is ideal as an evaluation chip for the S3C828B/C8289/C8285.

**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 19. Embedded Flash Memory Interface.
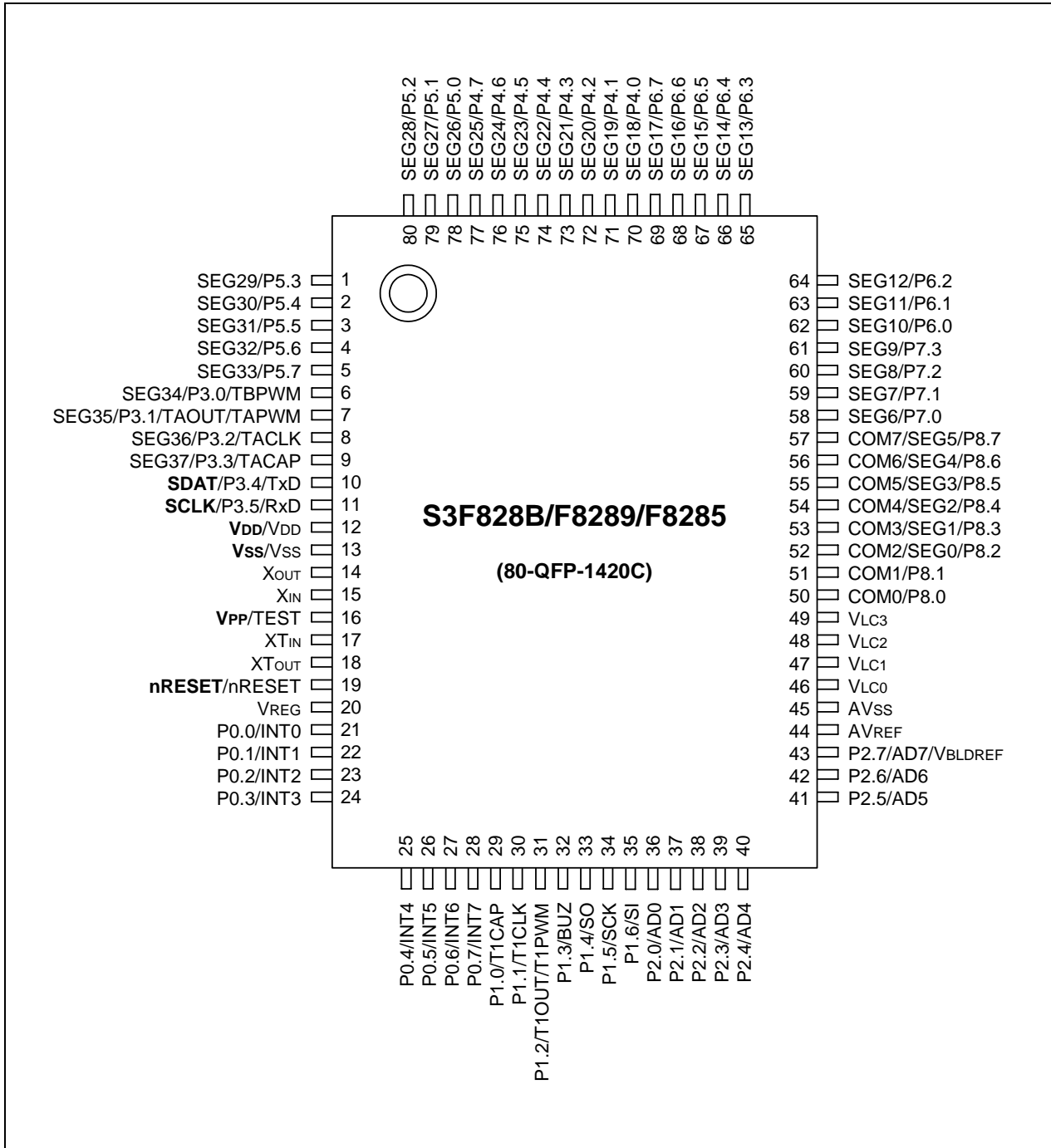
**Figure 22-1. S3F828B/F8289/F8285 Pin Assignments (80-QFP-1420C)**
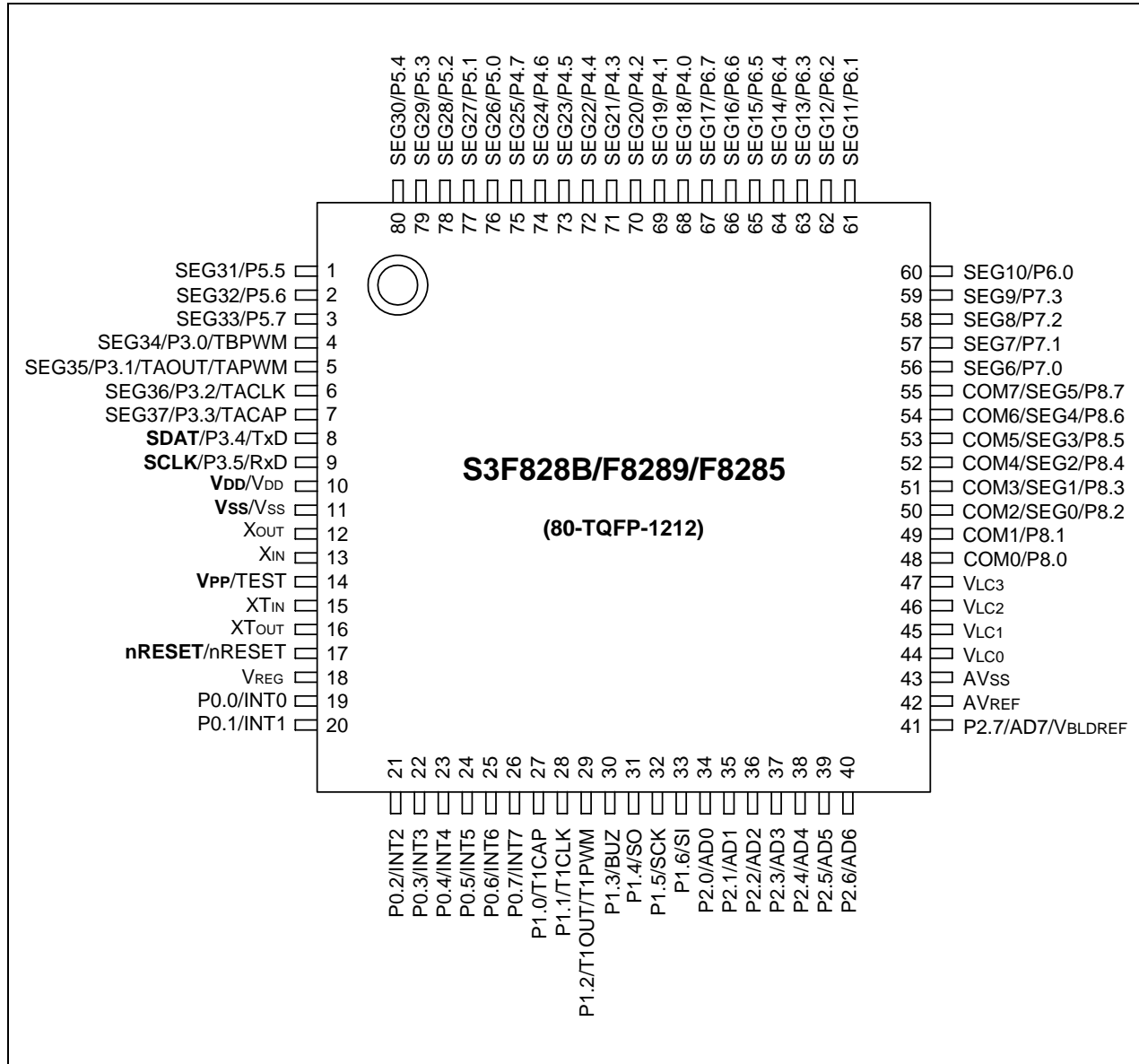
**Figure 22-2. S3F828B/F8289/F8285 Pin Assignments (80-TQFP-1212)**

**Table 22-1. Descriptions of Pins Used to Read/Write the Flash ROM**

| Main Chip | During Programming | | | |
|-----------|-----------|---------|-----|----------|
| Pin Name | Pin Name | Pin No. | I/O | Function |
| P3.4 | SDAT | 10(8) | I/O | Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input/push-pull output port. |
| P3.5 | SCLK | 11(9) | I/O | Serial clock pin. Input only pin. |
| TEST | $V_{PP}$ | 16(14) | I | Power supply pin for Flash ROM cell writing (indicates that FLASH MCU enters into the writing mode). When 12.5 V is applied, FLASH MCU is in writing mode and when 3.3 V is applied, FLASH MCU is in reading mode. |
| nRESET | nRESET | 19(17) | I | Chip Initialization |
| $V_{DD}$, $V_{SS}$ | $V_{DD}$, $V_{SS}$ | 12(10) 13(11) | – | Power supply pin for logic circuit. VDD should be tied to +3.3V during programming. |

**NOTES:**
1. Parentheses indicate pin number for 80-TQFP-1212 package.
2. The $V_{PP}$ (Test) pin had better connect to $V_{DD}$ (S3F828B only).

**Table 22-2. Comparison of S3F828B/F8289/F8285 and S3C828B/C8289/C8285 Features**

| Characteristic | S3F828B/9/5 | S3C828B/9/5 |
|----------------|-------------|-------------|
| Program memory | 64K/32K/16K-byte Flash ROM | 64K/32K/16K-byte mask ROM |
| Operating voltage ($V_{DD}$) | 2.0 V to 3.6 V | 2.0 V to 3.6 V |
| Flash MCU programming mode | $V_{DD}$ = 3.3 V, $V_{PP}$ (TEST) = 12.5 V | – |
| Programmability | User program multi time | Programmed at the factory |

## OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the $V_{PP}$ (TEST) pin of the S3C828B/C8289/C8285, 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 22-3 below.

**Table 22-3. Operating Mode Selection Criteria**

| $V_{DD}$ | $V_{PP}$(TEST) | REG/nMEM | 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 | 0 | 0000H | 1 | Flash ROM verify |
| | 12.5 V | 1 | 0E3FH | 0 | Flash ROM read protection |

**NOTES:**
1. The $V_{PP}$ (Test) pin had better connect to $V_{DD}$ (S3F828B only).
2. "0" means Low level; "1" means High level.

**Table 22-4. 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: $V_{DD} = 3.3V \pm 0.3V$ Crystal oscillator C1 = C2 = 22pF | 11.1 MHz | – | 4.0 | 8.0 | mA |
| | | | 4.0 MHz | | 1.8 | 3.6 | |
| | $I_{DD2}$[2] | Idle mode: $V_{DD} = 3.3V \pm 0.3V$ Crystal oscillator C1 = C2 = 22pF | 11.1MHz | – | 1.0 | 2.0 | |
| | | | 4.0 MHz | | 0.5 | 1.0 | |
| | $I_{DD3}$[3] | Run mode: $V_{DD} = 3.3V \pm 0.3V$, $T_A = 25°C$, OSCCON.7 = 1 32kHz crystal oscillator | | – | 14.0 | 28.0 | μA |
| | $I_{DD4}$[3] | Idle mode: $V_{DD} = 3.3V \pm 0.3V$, $T_A = 25°C$, OSCCON.7 = 1 32kHz crystal oscillator | | – | 2.0 | 4.0 | |
| | $I_{DD5}$[4] | Stop mode: $V_{DD} = 3V \pm 0.3V$ | $T_A = 25°C$ | – | 0.2 | 2.0 | |
| | | | $T_A = -25°C$ to $+85°C$ | – | – | 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 a power consumption of subsystem oscillator.
3. $I_{DD3}$ and $I_{DD4}$ are the current when the main system clock oscillation stops and the subsystem clock is used. (OSCCON.7=1)
4. $I_{DD5}$ is the current when the main and subsystem clock oscillation stops.
5. Every values in this table is measured when bits 4-3 of the system clock control register (CLKCON.4-.3) is set to 11B.

SAMSUNG
ELECTRONICS
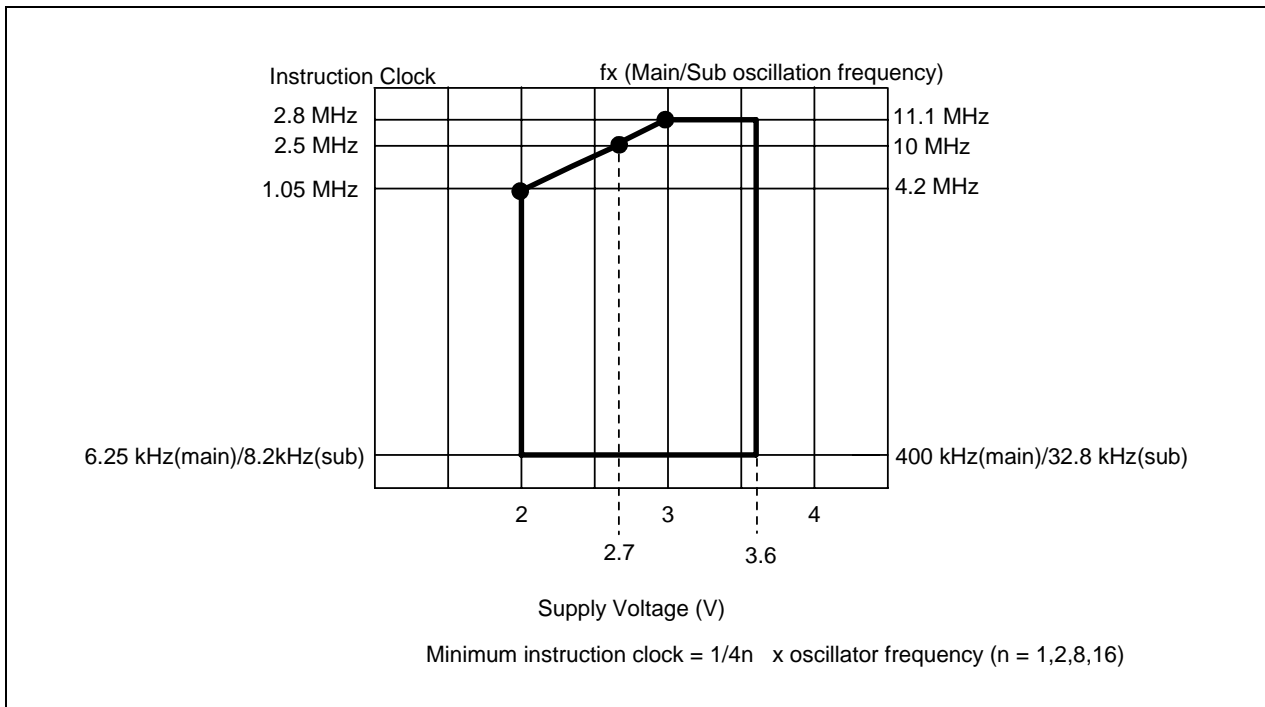
**Figure 22-3. Operating Voltage Range**

# 23 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.

**SAMSUNG**
ELECTRONICS

**23-1**
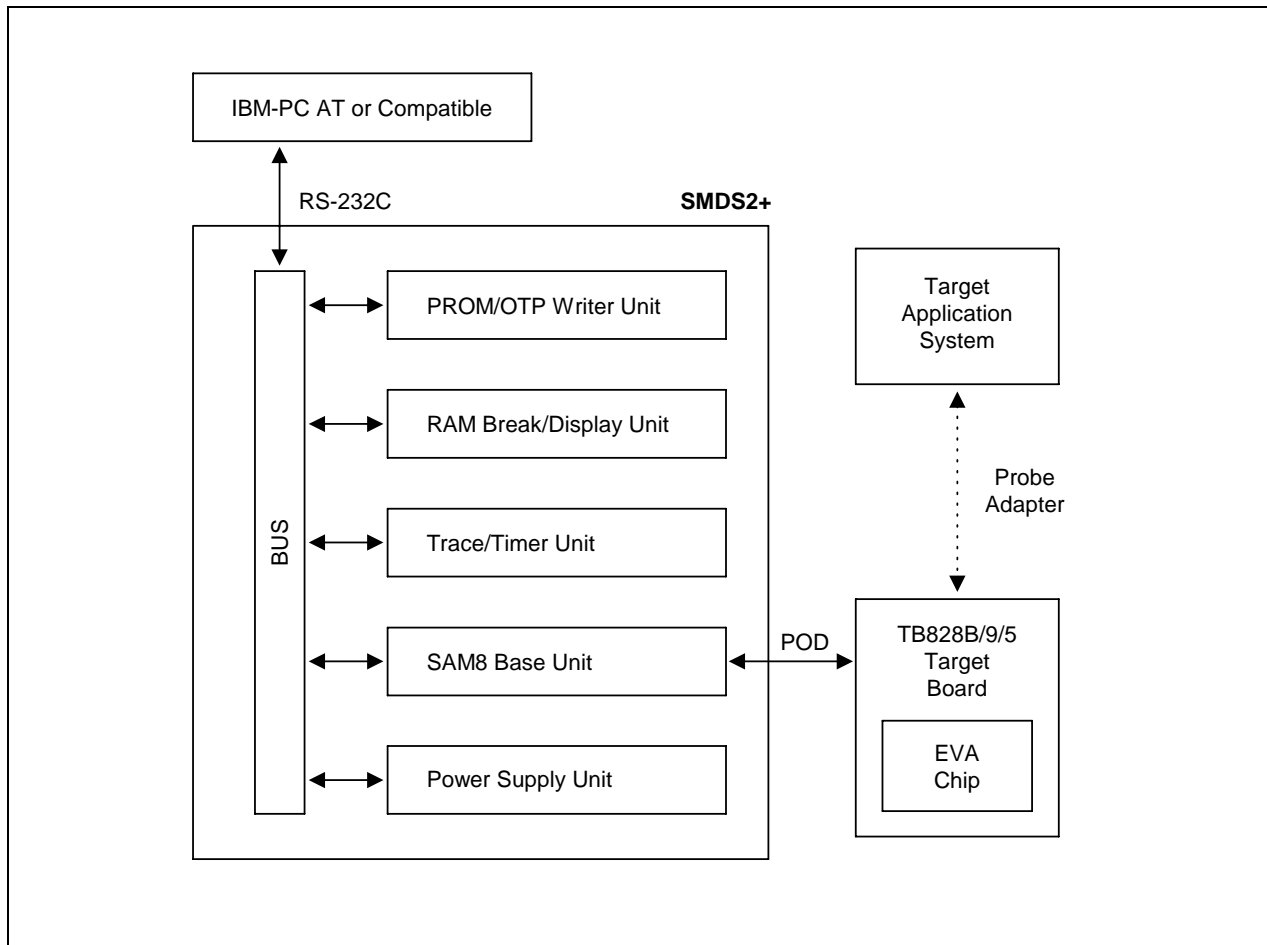
**Figure 23-1. SMDS Product Configuration (SMDS2+)**
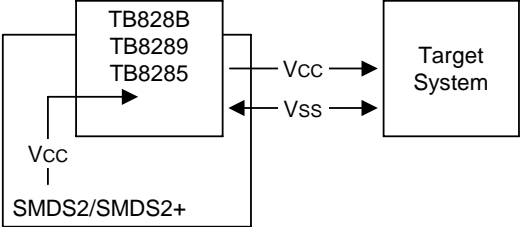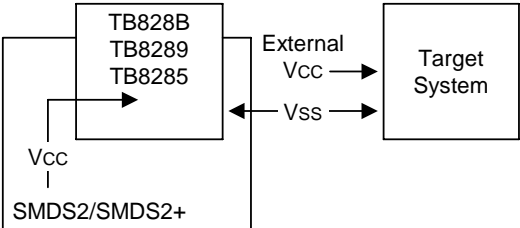
## TB828B/9/5 TARGET BOARD

The TB828B/9/5 target board is used for the S3C828B/F828B/C8289/F8289/C8285/F8285 microcontroller. It is supported with the SMDS2+.
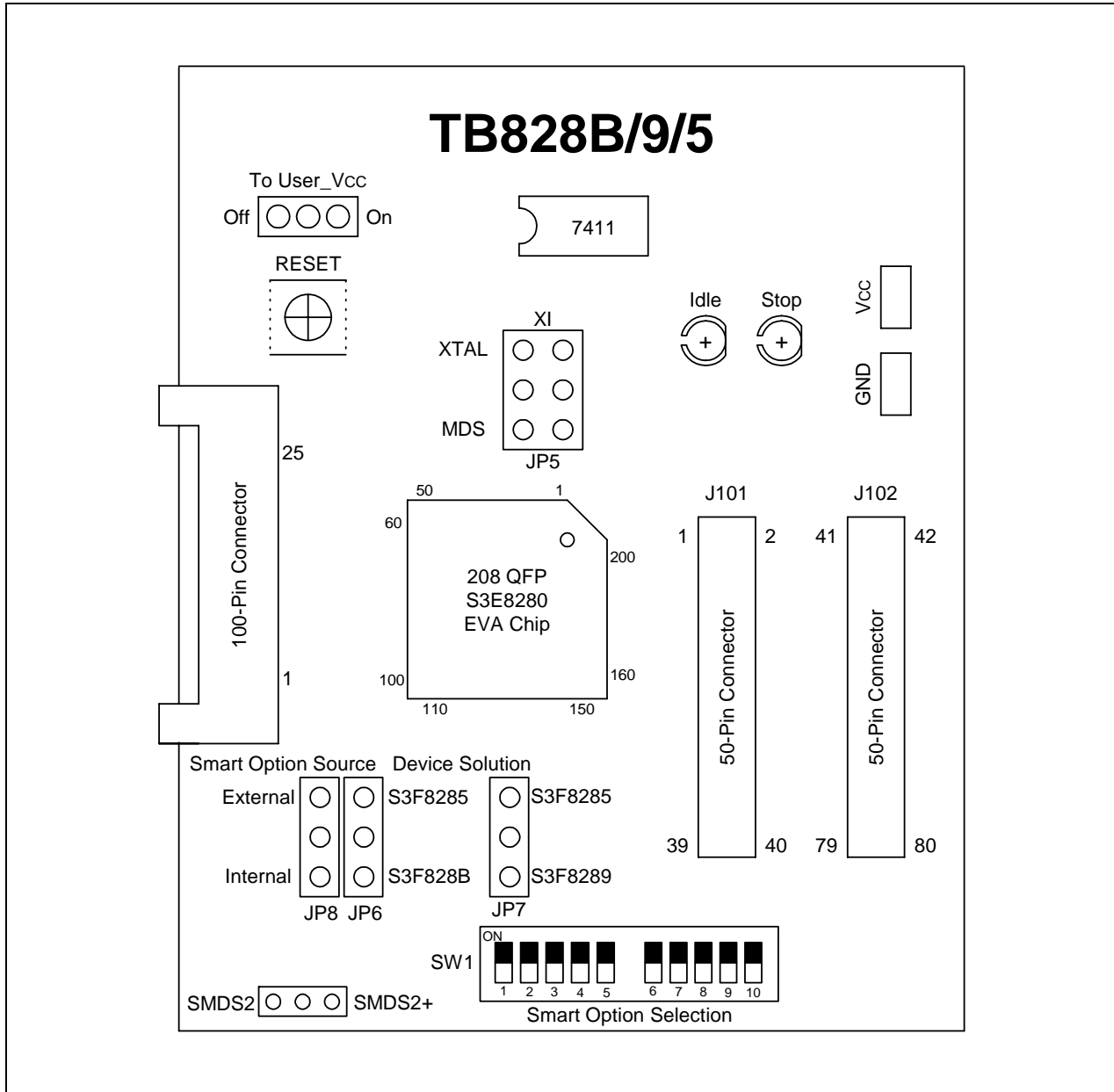

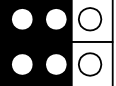
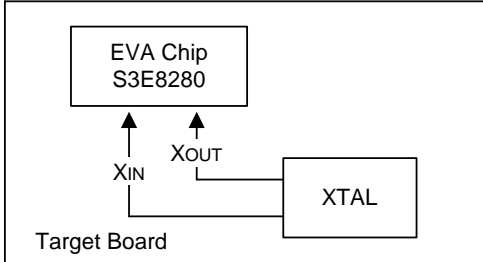**Figure 23-2. TB828B/9/5 Target Board Configuration**
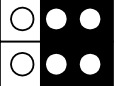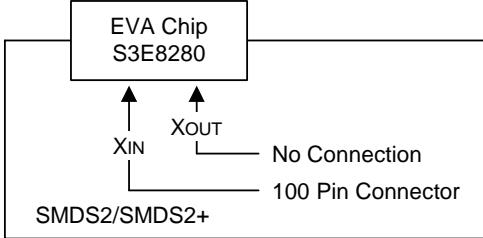
**Table 23-1. Power Selection Settings for TB828B/9/5**

| "To User_Vcc" Settings | Operating Mode | Comments |
|---|---|---|
| To User_VCC<br>Off ⦿ ●● On |  | The SMDS2/SMDS2+ supplies $V_{CC}$ to the target board (evaluation chip) and the target system. |
| To User_VCC<br>Off ●● ⦿ On |  | The SMDS2/SMDS2+ supplies $V_{CC}$ only to the target board (evaluation chip). The target system must have its own power supply. |
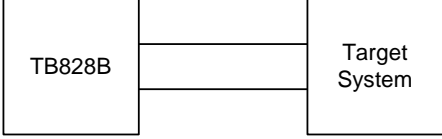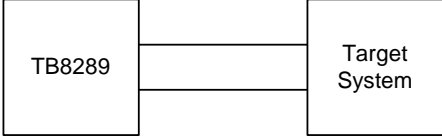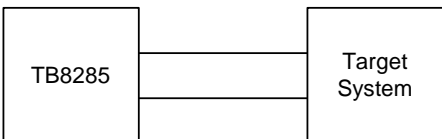
**NOTE:** The following symbol in the "To User_Vcc" Setting column indicates the electrical short (off) configuration:



**Table 23-2. Main-clock Selection Settings for TB828B/9/5**

| Main Clock Settings | Operating Mode | Comments |
|---|---|---|
| XIN<br>XTAL ⦿ ●● MDS<br>⦿ ●● |  | Set the XI switch to "MDS" when the target board is connected to the SMDS2/SMDS2+. |
| XIN<br>XTAL ●● ⦿ MDS<br>●● ⦿ |  | Set the XI switch to "XTAL" when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+. |
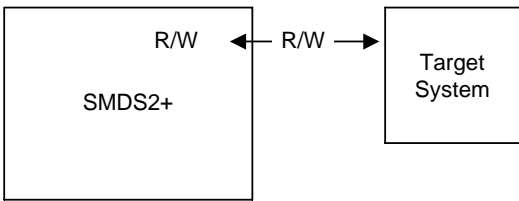
**Table 23-3. Device Selection Settings for TB828B/9/5**

| "Device Selection" Settings | Operating Mode | Comments |
|---|---|---|
| Device Selection<br><br>8249/5 ○●● 828B | TB828B —— Target System | Operate with TB828B |
| Device Selection<br><br>8289/5 ●●○ 828B<br><br>8285 ○●● 8289 | TB8289 —— Target System | Operate with TB8289 |
| Device Selection<br><br>8289/5 ●●○ 828B<br><br>8285 ●●○ 8289 | TB8285 —— Target System | Operate with TB8285 |

## 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 23-4. The SMDS2+ Tool Selection Setting**

| "SMDS2+" Setting | Operating Mode |
|---|---|
| SMDS2 ○●● SMDS2+ | SMDS2+ R/W ← R/W → Target System |
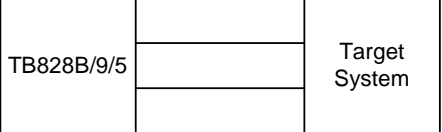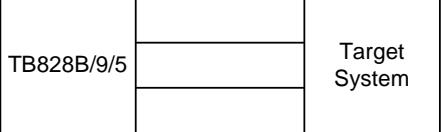
## IDLE LED

The Yellow LED is ON when the evaluation chip (S3E8280) is in idle mode.
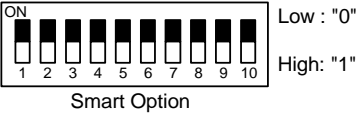
## STOP LED

The Red LED is ON when the evaluation chip (S3E8280) is in stop mode.

**Table 23-5. Smart Option Source Settings for TB828B/9/5**

| "Smart Option Source" Settings | Operating Mode | | Comments |
|---|---|---|---|
| Select Smart Option Source<br><br>Internal ○●● External | TB828B/9/5 | Target System | The Smart Option is selected by external smart option switch (SW1) |
| Select Smart Option Source<br><br>Internal ●●○ External | TB828B/9/5 | Target System | The Smart Option is selected by internal smart option area (003EH–0003FH of ROM). But this selection is not available. |

**Table 23-6. Smart Option Switch Setting for TB828B/9/5**

| "Smart Option" Setting | Comments |
|---|---|
| ON<br>1 2 3 4 5 6 7 8 9 10<br>Smart Option<br><br>Low : "0"<br>High: "1" | The Smart Option can be selected by this switch when the Smart Option source is selected by external. The SW1.3–SW1.1 are comparable to the 003EH.2–.0. The SW1.8-SW1.6 are comparable to the 003EH.7–.5. The SW1.9 is comparable to the 003FH.0. The SW1.5–1.4 is not connected. The SW1.10 is not used. |

SAMSUNG
ELECTRONICS
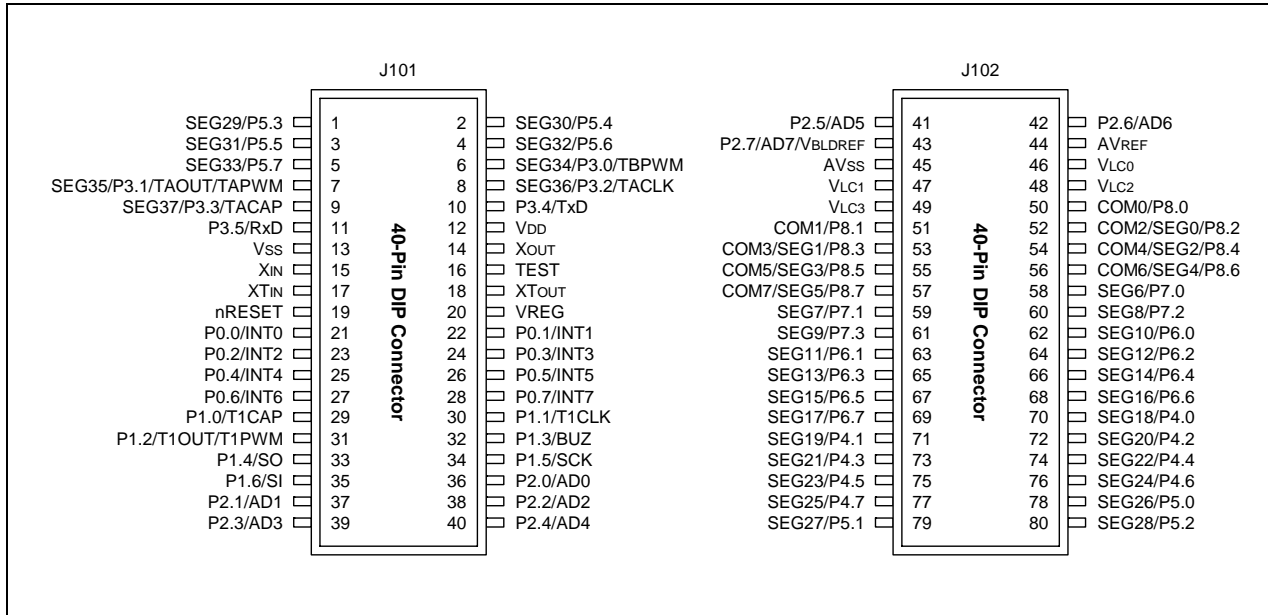
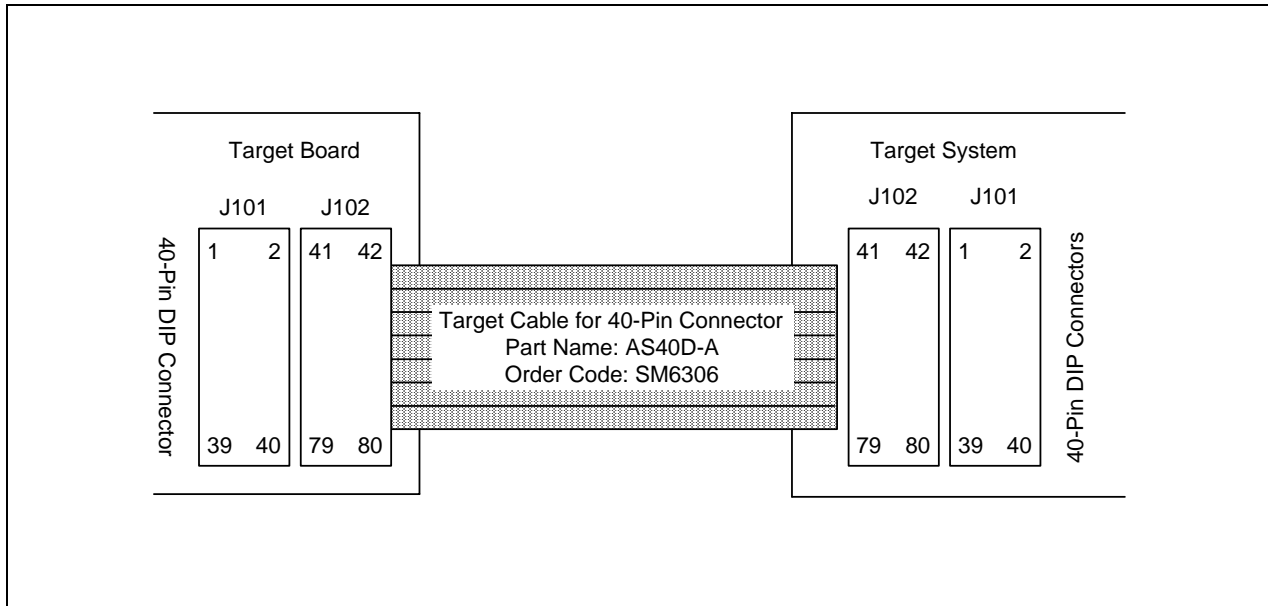**Figure 23-3. 40-Pin Connectors (J101, J102) for TB828B/9/5**



**Figure 23-4. S3E8280 Cables for 80-QFP Package**

# S3C8 SERIES MASK ROM ORDER FORM

**Product description:**

Device Number: S3C8_____- _____(write down the ROM code number)

Product Order Form: ☐ Package ☐ Pellet ☐ Wafer Package Type: _____

**Package Marking (Check One):**

☐ Standard ☐ Custom A ☐ Custom B
(Max 10 chars) (Max 10 chars each line)

**SEC** @ YWW
Device Name

@ YWW
Device Name

@ YWW

@ : Assembly site code, Y : Last number of assembly year, WW : Week of assembly

**Delivery Dates and Quantities:**

| Deliverable | Required Delivery Date | Quantity | Comments |
|---|---|---|---|
| ROM code | – | Not applicable | See ROM Selection Form |
| Customer sample | | | |
| Risk order | | | See Risk Order Sheet |

Please answer the following questions:

☞ **For what kind of product will you be using this order?**

☐ New product ☐ Upgrade of an existing product

☐ Replacement of an existing product ☐ Other

If you are replacing an existing product, please indicate the former product name
( )

☞ **What are the main reasons you decided to use a Samsung microcontroller in your product?**

**Please check all that apply.**

☐ Price ☐ Product quality ☐ Features and functions

☐ Development system ☐ Technical support ☐ Delivery on time

☐ Used same micom before ☐ Quality of documentation ☐ Samsung reputation

**Mask Charge (US$ / Won):** _____

**Customer Information:**

Company Name: _____ Telephone number _____

**Signatures:** _____ _____
(Person placing the order) (Technical Manager)

(For duplicate copies of this form, and for additional ordering information, please contact your local
Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

# S3C8 SERIES
# REQUEST FOR PRODUCTION AT CUSTOMER RISK

**Customer Information:**

Company Name:                    _____

Department:                      _____

Telephone Number:        _____        Fax: _____

Date:                    _____

**Risk Order Information:**

Device Number:           S3C8_____- _____ (write down the ROM code number)

Package:                 Number of Pins: _____        Package Type: _____

Intended Application:    _____

Product Model Number:    _____

**Customer Risk Order Agreement:**

We hereby request SEC to produce the above named product in the quantity stated below. We believe our risk order product to be in full compliance with all SEC production specifications and, to this extent, agree to assume responsibility for any and all production risks involved.

**Order Quantity and Delivery Schedule:**

Risk Order Quantity:             _____ PCS

Delivery Schedule:

| Delivery Date (s) | Quantity | Comments |
|---|---|---|
| | | |
| | | |
| | | |

**Signatures:**      _____        _____

                    (Person Placing the Risk Order)        (SEC Sales Representative)

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

# S3C828B/C8289/C8285
# MASK OPTION SELECTION FORM

**Device Number:**        S3C8_____-_____(write down the ROM code number)

**Attachment (Check one):**    ☐   Diskette          ☐   PROM

**Customer Checksum:**     _____

**Company Name:**        _____

**Signature (Engineer):**    _____

Please answer the following questions:

☞   **Application** (Product Model ID: _____)

| | | |
|---|---|---|
| ☐ Audio | ☐ Video | ☐ Telecom |
| ☐ LCD Databank | ☐ Caller ID | ☐ LCD Game |
| ☐ Industrials | ☐ Home Appliance | ☐ Office Automation |
| ☐ Remocon | ☐ Other | |

Please describe in detail its application

---

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)