21-S3-C831B/P831B-062003

# **USER'S MANUAL**

S3C831B/P831B 8-Bit CMOS Microcontroller Revision 1



www.DataSheet4U.com

# 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.

# S3C831B MICROCONTROLLER

The S3C831B single-chip microcontroller are fabricated using the highly advanced CMOS process. Its design is based on the powerful SAM88RC CPU core. Stop and idle (power-down) modes were implemented to reduce power consumption.

The S3C831B is a microcontroller with a 64K-byte mask-programmable ROM embedded. The S3P831B is a microcontroller with a 64K-byte one-time-programmable ROM embedded.

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

- Large number of programable I/O ports (Total 72 pins)
- PLL frequency synthesizer
- 16-bits intermediate frequency counter
- Two synchronous SIO modules
- Two 8-bit timer/counters
- One 16-bit timer/counter
- Low voltage reset
- A/D converter with 8 selectable input pins

# OTP

The S3C831B microcontroller is also available in OTP (One Time Programmable) version, S3P831B. The S3P831B microcontroller has an on-chip 64K-byte one-time-programmable EPROM instead of masked ROM. The S3P831B is comparable to S3C831B, both in function and in pin configuration.



# FEATURES

### CPU

• SAM88RC CPU core

### Memory

- 2576-byte internal register file (including LCD display RAM)
- 64K-byte internal program memory area

# **Instruction Set**

- 78 instructions
- Idle and Stop instructions

# 72 I/O Pins

- 32 normal I/O pins
- 40 pins sharing with LCD segment signals

#### Interrupts

- 8 interrupt levels and 17 internal sources
- Fast interrupt processing feature

#### 8-Bit Basic Timer

- Watchdog timer function
- 4 kinds of clock source

#### **Timer/Counter 0**

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

# **Timer/Counter 1**

- Programmable 8-bit interval timer
- External event counter function

# Timer/Counter 2

- Programmable 16-bit interval timer
- External event counter function

#### Watch Timer

- Interval Time: 50ms, 0.5s, 1.0s at 4.5 MHz
- 1/1.5/3/6 kHz buzzer output selectable

# Analog to Digital Converter

- 8-channel analog input
- 8-bit conversion resolution

# Two 8-bit Serial I/O Interface

- 8-bit transmit/receive mode
- 8-bit receive mode
- Selectable baud rate or external clock source

# PLL Frequency Synthesizer

- V<sub>IN</sub> level: 300mVpp (minimum)
- AMVCO range: 0.5 MHz–30 MHz (3-bit counter added)
- FMVCO range: 30 MHz-150 MHz

# 16-Bit Intermediate Frequency (IF) Counter

- V<sub>IN</sub> level: 300mV<sub>PP</sub> (minimum)
- AMIF range: 100 kHz-1 MHz
- FMIF range: 5 MHz-15 MHz

# LCD Controller/Driver

- 40 segments and 4 common terminals
- 4/3/2 common and static selectable
- Internal or external resistor circuit for LCD bias

# Low Voltage Reset (LVR)

- · Low voltage check to make system reset
- V<sub>LVR</sub>: 2.4V, 3.7 V selectable

#### **Two Power-Down Modes**

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

#### **Oscillation Source**

Crystal or ceramic for system clock (fx)

#### Instruction Execution Time

• 444 ns at 9.0 MHz (minimum)

#### Operating Temperature Range

• -25 °C to +85 °C

# **Operating Voltage Range**

- 2.2V to 5.5V at 0.4 MHz 4.5 MHz
- 4.0 V to 5.5 V at 0.4 MHz–9.0 MHz
- 2.5V to 3.5V, 4.5 V to 5.5 V in PLL/IFC block

#### Package Type

100-QFP-1420C, 100-TQFP-1414



# **BLOCK DIAGRAM**

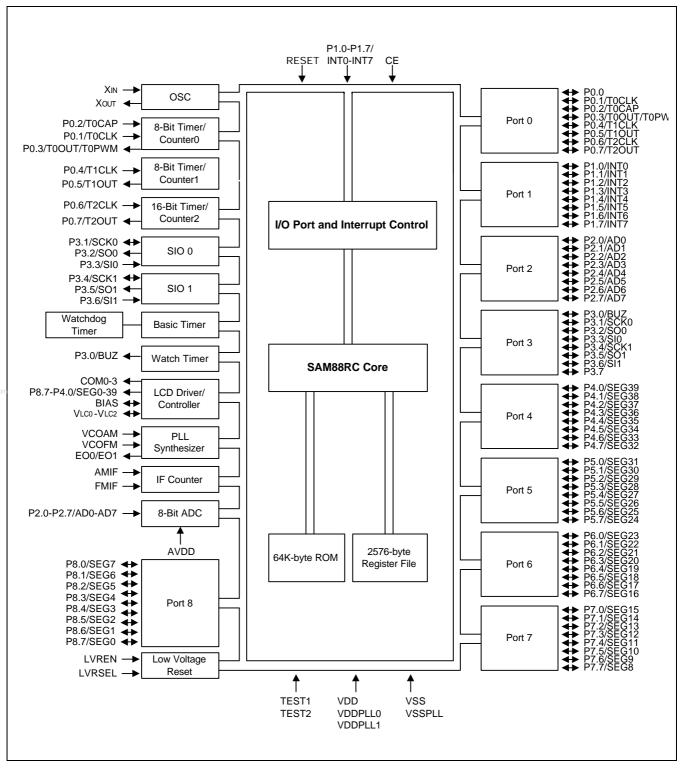


Figure 1-1. Block Diagram



# **PIN ASSIGNMENT**

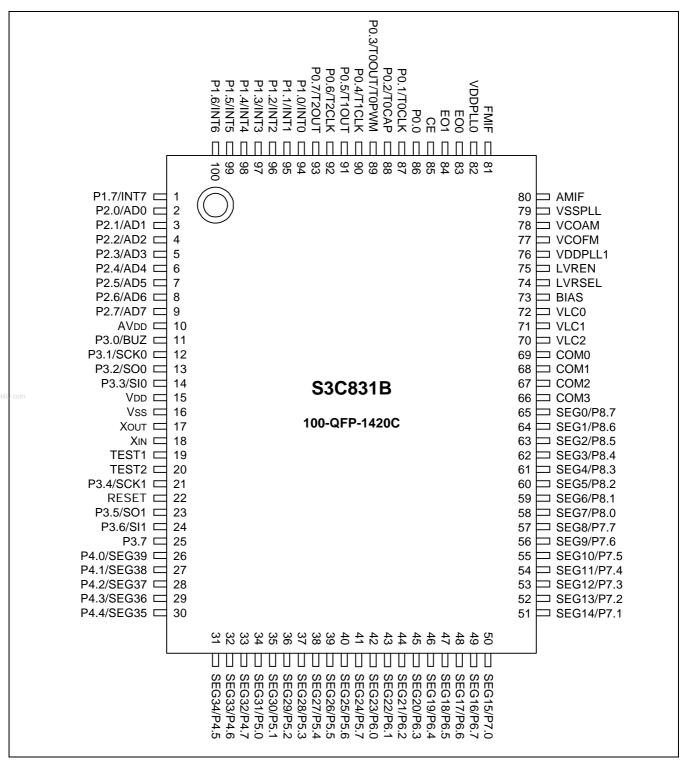


Figure 1-2. S3C831B Pin Assignments (100-QFP-1420C)



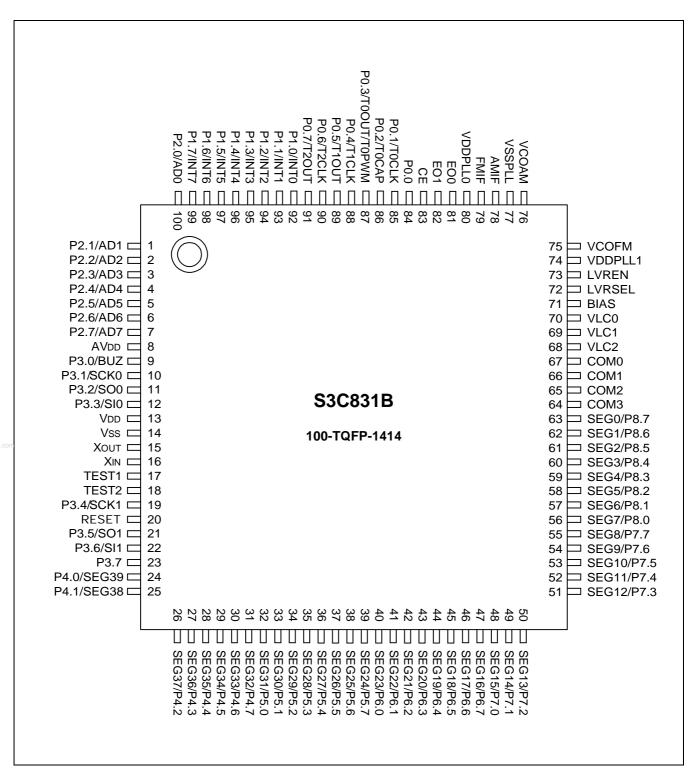


Figure 1-3. S3C831B Pin Assignments (100-TQFP-1414)



# **PIN DESCRIPTIONS**

Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Share Pins
P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	I/O	I/O port with bit programmable pins; Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups.	E-4	86(84) 87(85) 88(86) 89(87) 90(88) 91(89) 92(90) 93(91)	– TOCLK TOCAP TOOUT/TOPWM T1CLK TOUT T2CLK T2OUT
P1.0-P1.3	I/O	I/O port with bit programmable pins; Schmitt trigger Input or push-pull output and software assignable pull-ups; Alternately used for external interrupt input (noise filters, interrupt enable and pending control).	D-7	94-97 (92-95)	INTO-INT3
P1.4-P1.7	I/O	I/O port with bit programmable pins; Input or push-pull and software assignable pull-ups; Alternately used for external interrupt input (Noise filters, interrupt enable and pending control)	D-8	98-1 (96-99)	INT4-INT7
P2.0-P2.7	I/O	I/O port with bit programmable pins; Schmitt trigger input or push-pull output and software assignable pull-ups.	F-16	2-9 (100-7)	AD0-AD7
P3.0 P3.1 P3.2 P3.3 P3.4 P3.5 P3.6 P3.7	I/O	I/O port with bit programmable pins; Input or push- pull, open-drain output and software assignable pull-ups.	E-2	11(9) 12(10) 13(11) 14(12) 21(19) 23(21) 24(22) 25(23)	BUZ SCK0 SO0 SI0 SCK1 SO1 SI1 -
P4.0-P4.7	I/O	I/O port with nibble programmable pins; Input or push-pull, open-drain output and software assignable pull-ups.	H-42	26-33 (24-31)	SEG39-SEG32
P5.0-P5.7	I/O	Same as Port 4	H-42	34-41 (32-39)	SEG31-SEG24
P6.0-P6.7	I/O	I/O port with nibble programmable pins; Schmitt trigger input or push-pull, open-drain output and software assignable pull-ups.	H-41	42-49 (40-47)	SEG23-SEG16
P7.0-P7.7	I/O	Same as Port 6	H-41	50-57 (48-55)	SEG15-SEG8
P8.0-P8.7	I/O	Same as Port 6	H-41	58-65 (56-63)	SEG7-SEG0

# Table 1-1. S3C831B Pin Descriptions

**NOTE:** The parentheses indicate pin number for 100-TQFP-1414 package.



Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Share Pins
COM0-COM3	0	Common signal output for LCD display	н	69-66 (67-64)	_
SEG0-SEG23	I/O	LCD segment signal output	H-41	65-42 (63-40)	P8-P6
SEG24-SEG39	I/O	LCD segment signal output	H-42	41-26 (39-24)	P5-P4
BIAS	Ι	LCD power control	_	73(71)	_
VLC0 VLC1 VLC2	I	LCD power supply Voltage dividing resistors are assignable by software	_	72-70 (70-68)	_
V <sub>DD</sub>	-	Main power supply	-	15(13)	_
V <sub>SS</sub>	-	Main ground	-	16(14)	_
VDDPLL0-1	-	PLL/IFC power supply	-	82, 76 (80,74)	_
VSSPLL	-	PLL/IFC ground	-	79(77)	_
AV <sub>DD</sub>	-	A/D converter power supply	-	10(8)	_
X <sub>OUT</sub> , X <sub>IN</sub>	_	Main oscillator pins for CPU oscillation	-	17, 18 (15,16)	_
TEST1, TEST2	I	Test signal input pin (Must be connected to V <sub>SS</sub> )	_	19, 20 (17,18)	_
LVRSEL	I	LVR criterion voltage selection pin (Must be connected to $V_{DD}$ or $V_{SS}$ )	A	74(72)	_
LVREN	I	LVR enable pin (Must be connected to $V_{DD}$ or $V_{SS}$ )	A	75(73)	_
RESET	I	System reset pin	В	22(20)	_
CE	I	Input pin for checking device power Normal operation is high level and PLL/IFC Operation is stopped at low power	B-5	85(83)	_
EO0	0	PLL's phase error output0	A-2	83(81)	_
EO1	0	PLL's phase error output1	A-2	84(82)	_
VCOAM VCOFM	Ι	External VCOAM/VCOFM signal inputs	B-4	78, 77 (76,75)	_

Table 1-1.	S3C831B	Pin	Descriptions	(Continued)
------------	---------	-----	--------------	-------------

**NOTE:** The parentheses indicate pin number for 100-TQFP-1414 package.



Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Share Pins
FMIF, AMIF	I	FM/AM intermediate frequency signal inputs	B-4	81, 80 (79,78)	-
AD0-AD7	I/O	ADC input pins	F-16	2-9 (100-7)	P2.0-P2.7
BUZ	I/O	1, 1.5, 3 or 6 kHz frequency output for buzzer sound at 4.5 MHz clock	E-2	11(9)	P3.0
SCK0	I/O	SIO0 interface signal	E-2	12(10)	P3.1
SO0	I/O	SIO0 interface data output signal	E-2	13(11)	P3.2
SI0	I/O	SIO0 interface data input signal	E-2	14(12)	P3.3
SCK1	I/O	SIO1 interface signal	E-2	21(19)	P3.4
SO1	I/O	SIO1 interface data output signal	E-2	23(21)	P3.5
SI1	I/O	SIO1 interface data input signal	E-2	24(22)	P3.6
TOCLK	I/O	Timer 0 clock input	E-4	87(85)	P0.1
T0CAP	I/O	Timer 0 capture input	E-4	88(86)	P0.2
TOOUT	I/O	Timer 0 clock output	E-4	89(87)	P0.3
TOPWM	I/O	Timer 0 PWM output	E-4	89(87)	P0.3
T1CLK	I/O	Timer 1 clock input	E-4	90(88)	P0.4
T1OUT	I/O	Timer 1 clock output	E-4	91(89)	P0.5
T2CLK	I/O	Timer 2 clock input	E-4	92(90)	P0.6
T2OUT	I/O	Timer 2 clock output	E-4	93(91)	P0.7
INT0-INT3	I/O	External interrupt input pins	D-7	94-97 (92-95)	P1.0-P1.3
INT4-INT7	I/O	External interrupt input pins	D-8	98-1 (96-99)	P1.4-P1.7

**NOTE:** The parentheses indicate pin number for 100-TQFP-1414 package.



# **PIN CIRCUITS**

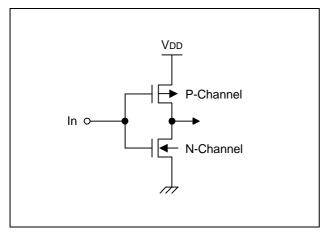


Figure 1-4. Pin Circuit Type A

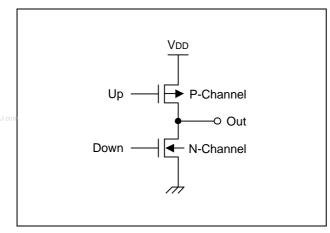


Figure 1-5. Pin Circuit Type A-2 (EO)

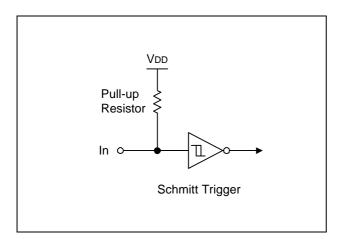


Figure 1-6. Pin Circuit Type B (RESET)



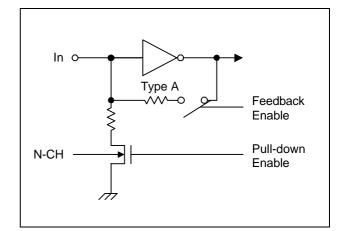


Figure 1-7. Pin Circuit Type B-4

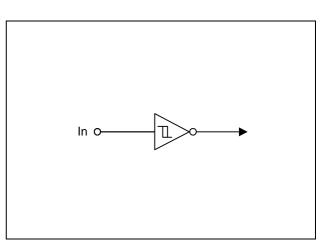


Figure 1-8. Pin Circuit Type B-5 (CE)

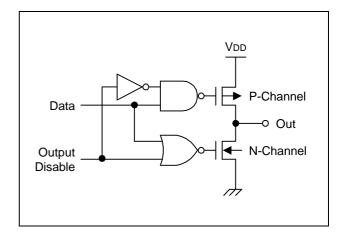


Figure 1-9. Pin Circuit Type C

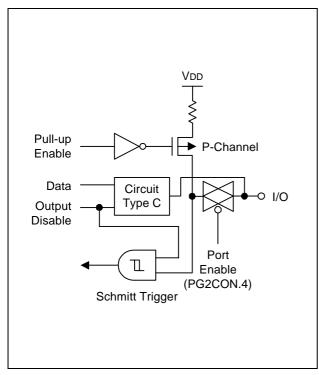


Figure 1-10. Pin Circuit Type D-7 (P1.0-P1.3)

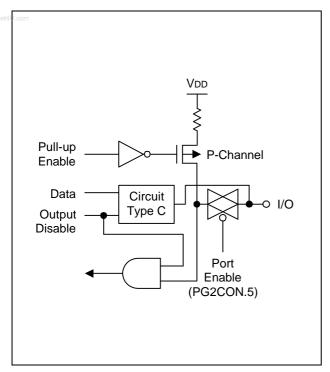


Figure 1-11. Pin Circuit Type D-8 (P1.4-P1.7)

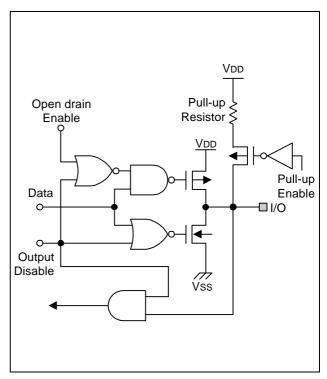


Figure 1-12. Pin Circuit Type E-2 (P3)

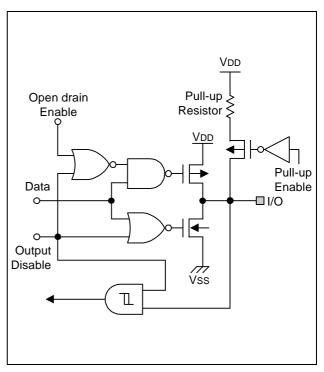


Figure 1-13. Pin Circuit Type E-4 (P0)



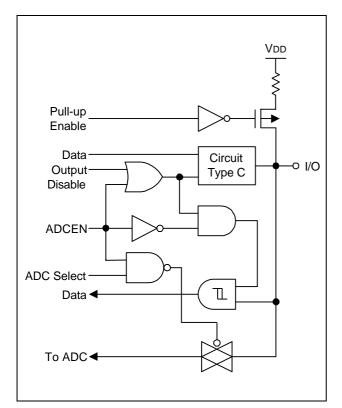


Figure 1-14. Pin Circuit Type F-16 (P2)

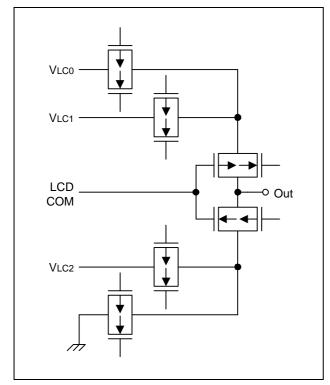


Figure 1-15. Pin Circuit Type H (COM0-COM3)



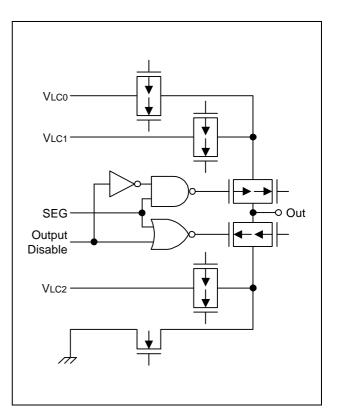


Figure 1-16. Pin Circuit Type H-39

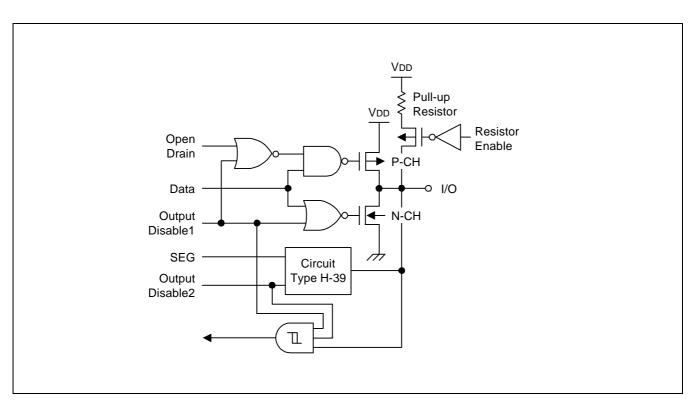


Figure 1-17. Pin Circuit Type H-41 (P6-P8)

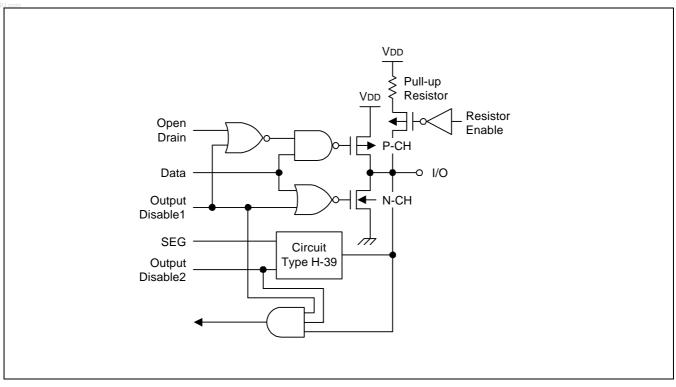


Figure 1-18. Pin Circuit Type H-42 (P4, P5)



# **2** ADDRESS SPACES

# OVERVIEW

The S3C831B 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 S3C831B has an internal 64-Kbyte mask-programmable ROM.

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

A 20-byte LCD display register file is implemented.

There are 2,646 mapped registers in the internal register file. Of these, 2,576 are for general-purpose. (This number includes a 16-byte working register common area used as a "scratch area" for data operations, ten 192-byte prime register areas, and ten 64-byte areas (Set 2)). Thirteen 8-bit registers are used for the CPU and the system control, and 57 registers are mapped for peripheral controls and data registers. Ten register locations are not mapped.



# **PROGRAM MEMORY (ROM)**

Program memory (ROM) stores program codes or table data. The S3C831B has 64K bytes internal mask-programmable program memory.

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

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

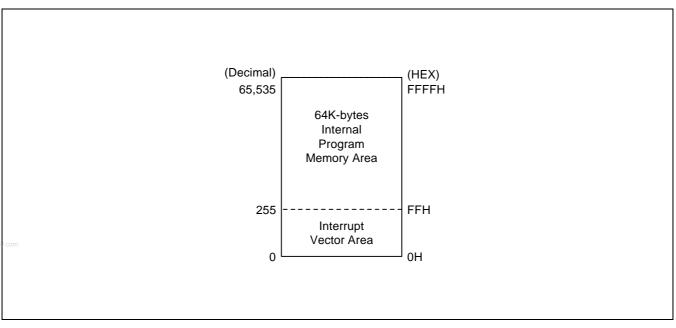


Figure 2-1. Program Memory Address Space



# **REGISTER ARCHITECTURE**

In the S3C831B 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 S3C831B the total number of addressable 8-bit registers is 2,646. Of these 2,646 registers, 13 bytes are for CPU and system control registers, 57 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 (including 20 bytes for LCD display registers).

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.

Register Type	Number of Bytes
General-purpose registers (including the 16-byte common working register area, ten 192-byte prime register area (including LCD data registers), and ten 64-byte set 2 area).	2,576
CPU and system control registers	13
Mapped clock, peripheral, I/O control, and data registers	57
Total Addressable Bytes	2,646

#### Table 2-1. S3C831B Register Type Summary



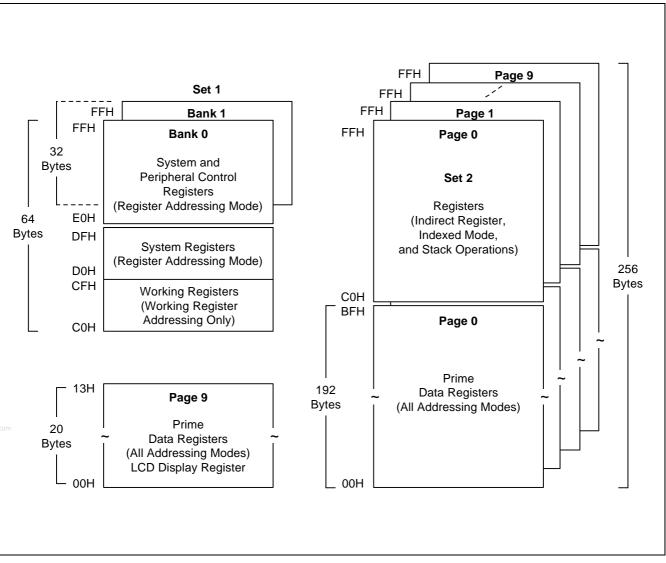


Figure 2-2. Internal Register File Organization



# **REGISTER PAGE POINTER (PP)**

The S3C8-series architecture supports the logical expansion of the physical 256-byte internal register file (using an 8-bit data bus) into as many as 16 separately addressable register pages. Page addressing is controlled by the register page pointer (PP, DFH). In the S3C831B microcontroller, a paged register file expansion is implemented for LCD data registers, and the register page pointer must be changed to address other pages.

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

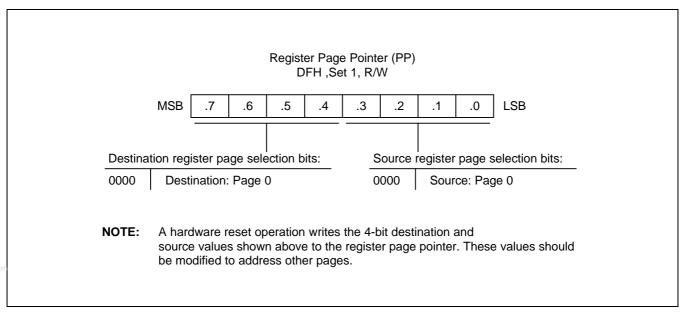


Figure 2-3. Register Page Pointer (PP)

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

	LD SRP	PP,#00H #0C0H	;	Destination $\leftarrow$ 0, Source $\leftarrow$ 0
RAMCL0	LD CLR DJNZ	R0,#0FFH @R0 R0,RAMCL0	;	Page 0 RAM clear starts
	CLR	@R0	;	R0 = 00H
RAMCL1	LD LD CLR DJNZ	PP,#10H R0,#0FFH @R0 R0,RAMCL1	;	Destination $\leftarrow$ 1, Source $\leftarrow$ 0 Page 1 RAM clear starts
	CLR	@R0	;	R0 = 00H

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



#### **REGISTER SET 1**

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

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

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

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

# **REGISTER SET 2**

The same 64-byte physical space that is used for set 1 locations C0H–FFH is logically duplicated to add another 64 bytes of register space. This expanded area of the register file is called set 2. For the S3C831B, the set 2 address range (C0H–FFH) is accessible on pages 0-9.

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

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



### PRIME REGISTER SPACE

The lower 192 bytes (00H–BFH) of the S3C831B's ten 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, or 9 you must set the register page pointer (PP) to the appropriate source and destination values.

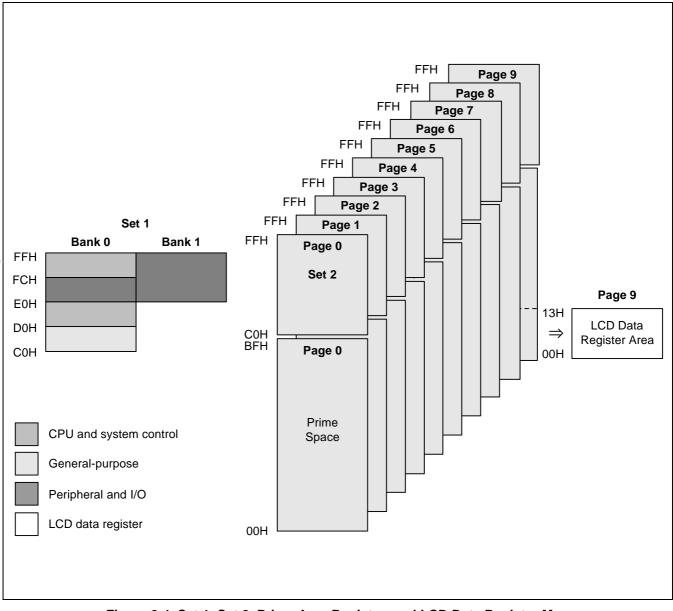


Figure 2-4. Set 1, Set 2, Prime Area Register, and LCD Data Register Map



#### WORKING REGISTERS

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

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

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

- One working register *slice* is 8 bytes (eight 8-bit working registers, R0–R7 or R8–R15)
- One working register *block* is 16 bytes (sixteen 8-bit working registers, R0–R15)

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

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

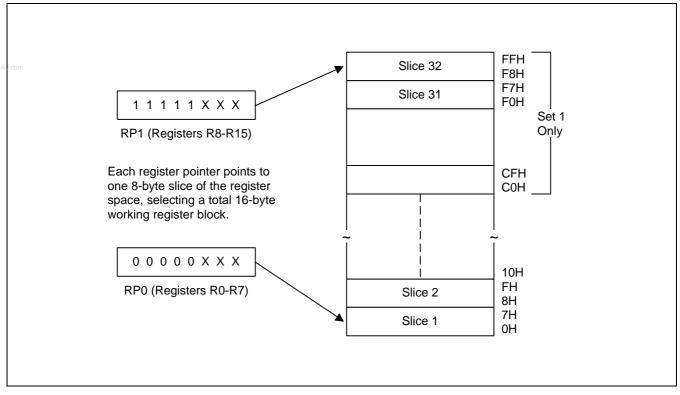


Figure 2-5. 8-Byte Working Register Areas (Slices)



### **USING THE REGISTER POINTS**

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

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

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

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

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

# PROGRAMMING TIP — Setting the Register Pointers

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

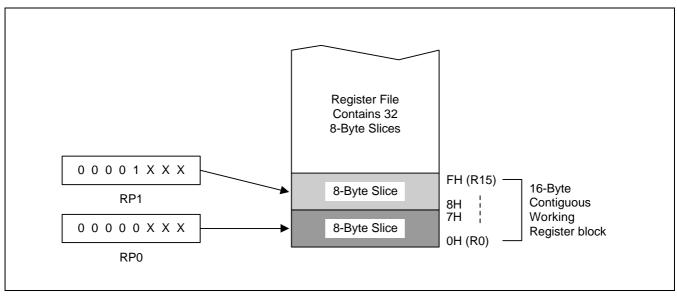


Figure 2-6. Contiguous 16-Byte Working Register Block



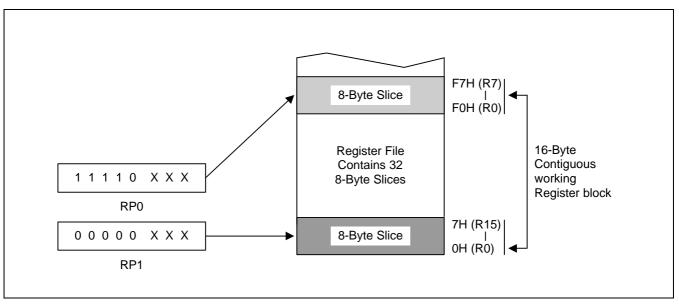


Figure 2-7. Non-Contiguous 16-Byte Working Register Block

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

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

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

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 $\leftarrow$ (80H) + (82H) + C
ADC	80H,83H	; 80H $\leftarrow$ (80H) + (83H) + C
ADC	80H,84H	; 80H ← (80H) + (84H) + C
ADC	80H,85H	; 80H $\leftarrow$ (80H) + (85H) + C

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



# **REGISTER ADDRESSING**

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

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

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

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

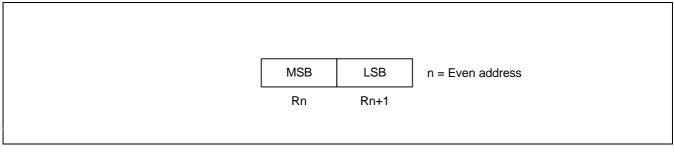


Figure 2-8. 16-Bit Register Pair



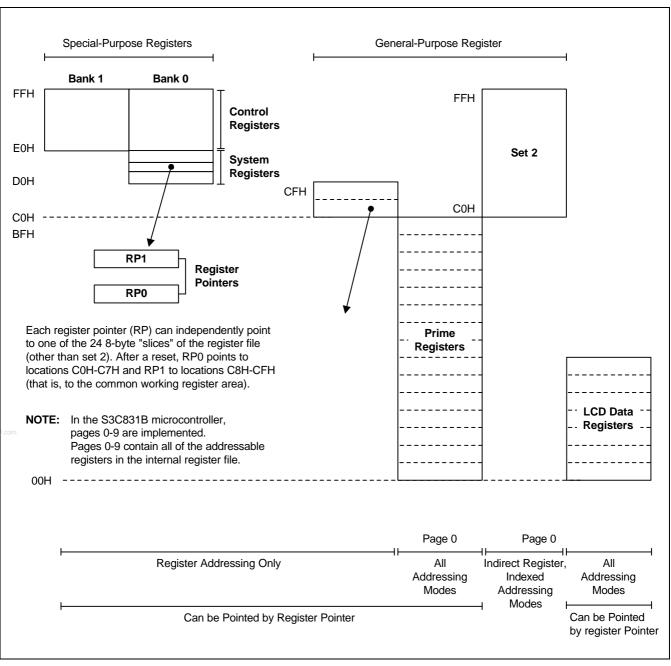


Figure 2-9. Register File Addressing



#### COMMON WORKING REGISTER AREA (C0H-CFH)

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

 $\mathsf{RP0} \ \rightarrow \ \mathsf{C0H-C7H}$ 

 $\text{RP1} \ \rightarrow \ \text{C8H-CFH}$ 

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

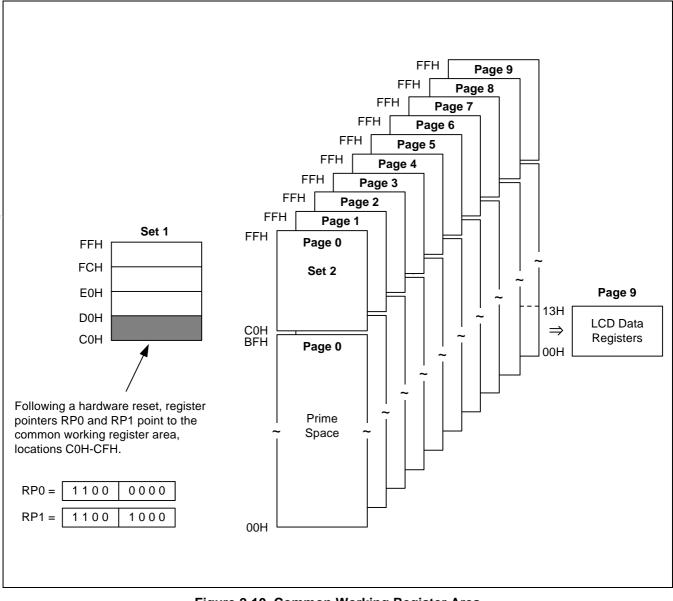


Figure 2-10. Common Working Register Area



# PROGRAMMING TIP — Addressing the Common Working Register Area

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

Examples	1. LD	0C2H,40H	;	Invalid addressing mode!
	Use worl	king register addressing inst	tead	d:
	SRP LD	#0C0H R2,40H	;	R2 (C2H) $\leftarrow$ the value in location 40H
	2. ADD Use worl	0C3H,#45H king register addressing inst	; tead	Invalid addressing mode! d:
	SRP ADD	#0C0H R3,#45H	;	R3 (C3H) ← R3 + 45H

#### **4-BIT WORKING REGISTER ADDRESSING**

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

- The high-order bit of the 4-bit address selects one of the register pointers ("0" selects RP0, "1" selects RP1).
- The five high-order bits in the register pointer select an 8-byte slice of the register space.
- The three low-order bits of the 4-bit address select one of the eight registers in the slice.

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

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



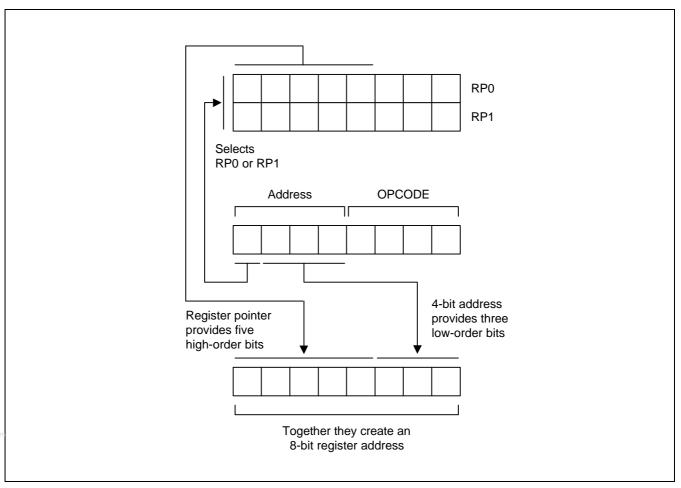


Figure 2-11. 4-Bit Working Register Addressing

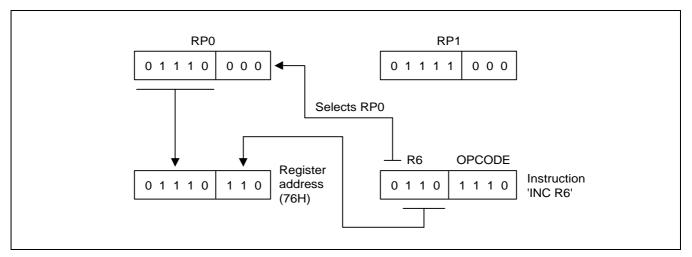


Figure 2-12. 4-Bit Working Register Addressing Example



#### 8-BIT WORKING REGISTER ADDRESSING

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

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

Figure 2-14 shows an example of 8-bit working register addressing. The four high-order bits of the instruction address (1100B) specify 8-bit working register addressing. Bit 4 ("1") selects RP1 and the five high-order bits in RP1 (10101B) become the five high-order bits of the register address. The three low-order bits of the register address (011) are provided by the three low-order bits of the 8-bit instruction address. The five address bits from RP1 and the three address bits from the instruction are concatenated to form the complete register address, 0ABH (101011B).

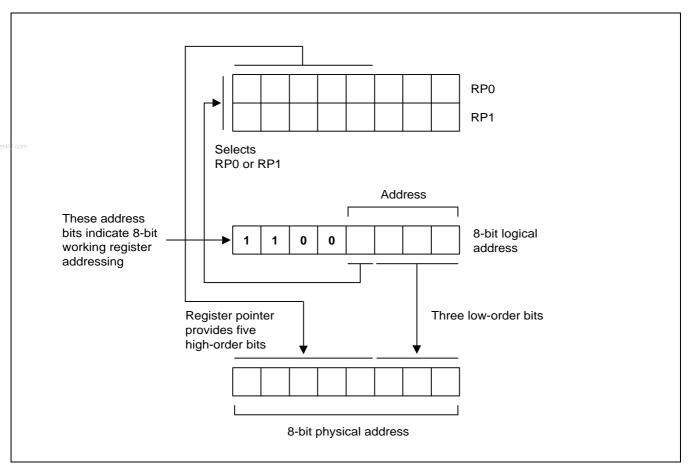


Figure 2-13. 8-Bit Working Register Addressing



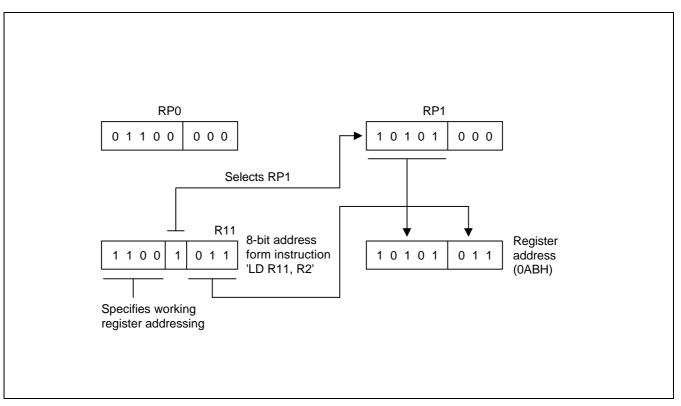


Figure 2-14. 8-Bit Working Register Addressing Example



# SYSTEM AND USER STACK

The S3C8-series microcontrollers use the system stack for data storage, subroutine calls and returns. The PUSH and POP instructions are used to control system stack operations. The S3C831B architecture supports stack operations in the internal register file.

### **Stack Operations**

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

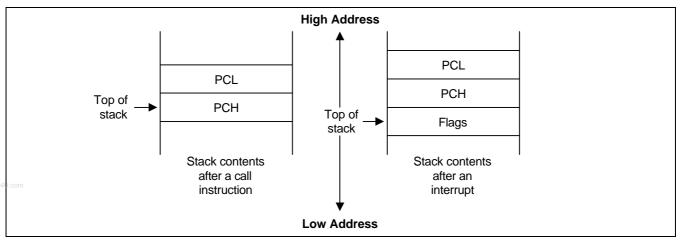


Figure 2-15. Stack Operations

# **User-Defined Stacks**

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

# Stack Pointers (SPL, SPH)

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

Because only internal memory space is implemented in the S3C831B, 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	<ul> <li>SPL ← FFH</li> <li>(Normally, the SPL is set to 0FFH by the initialization</li> <li>routine)</li> </ul>
• PUSH PUSH PUSH • •	PP RP0 RP1 R3	<ul> <li>; Stack address 0FEH ← PP</li> <li>; Stack address 0FDH ← RP0</li> <li>; Stack address 0FCH ← RP1</li> <li>; Stack address 0FBH ← R3</li> </ul>
• POP POP POP POP	R3 RP1 RP0 PP	; R3 $\leftarrow$ Stack address 0FBH ; RP1 $\leftarrow$ Stack address 0FCH ; RP0 $\leftarrow$ Stack address 0FDH ; PP $\leftarrow$ Stack address 0FEH



NOTES



# **3** ADDRESSING MODES

# OVERVIEW

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

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

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



# **REGISTER ADDRESSING MODE (R)**

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

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

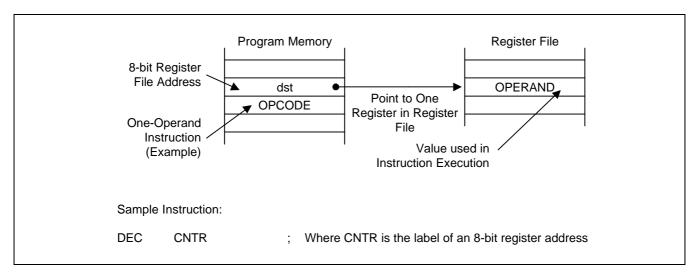


Figure 3-1. Register Addressing

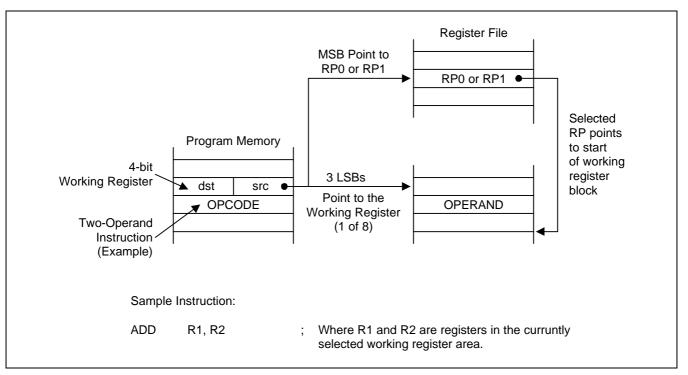


Figure 3-2. Working Register Addressing



# **INDIRECT REGISTER ADDRESSING MODE (IR)**

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

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

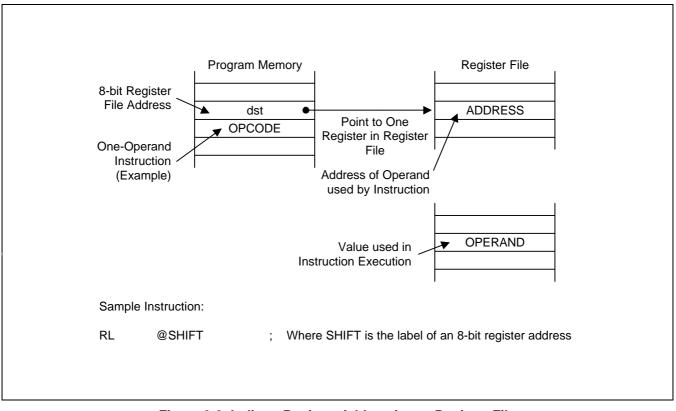
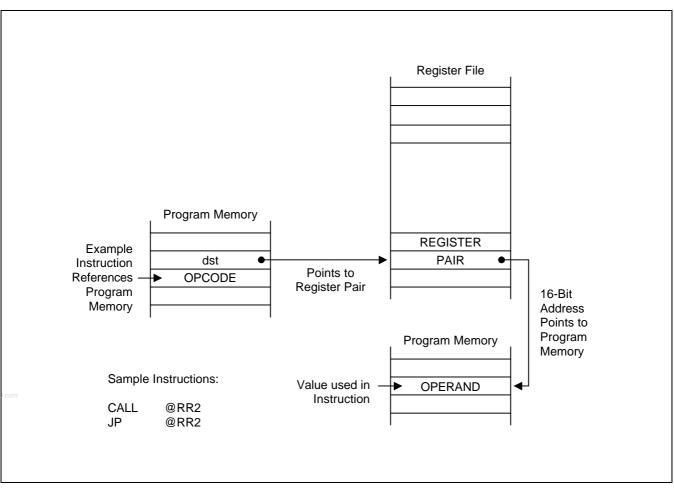


Figure 3-3. Indirect Register Addressing to Register File

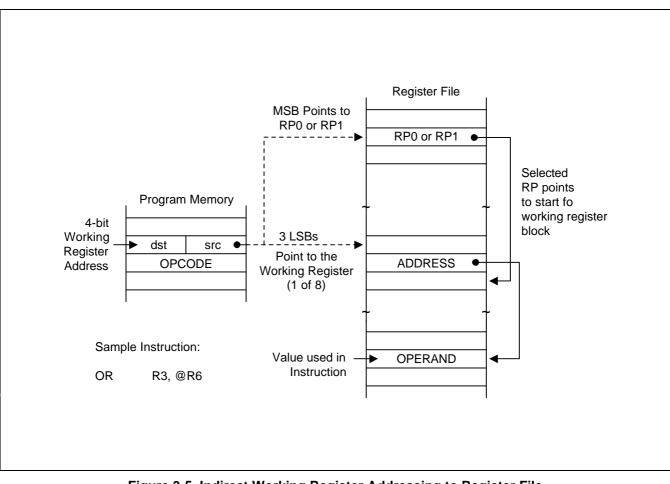




## INDIRECT REGISTER ADDRESSING MODE (Continued)

Figure 3-4. Indirect Register Addressing to Program Memory

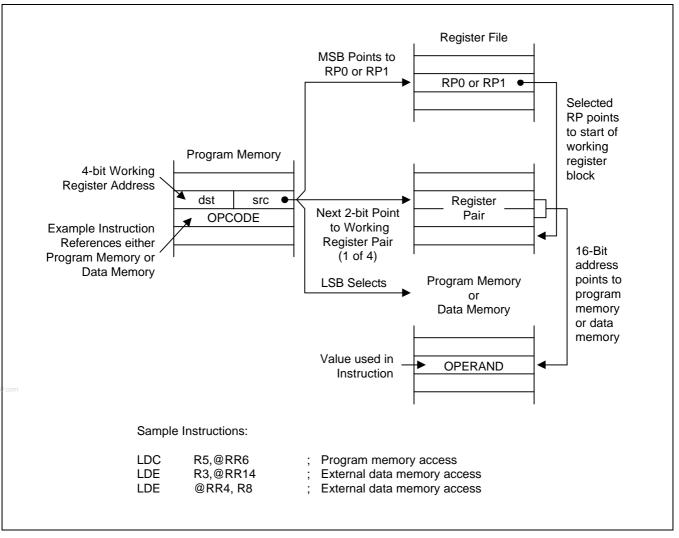




## INDIRECT REGISTER ADDRESSING MODE (Continued)

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.

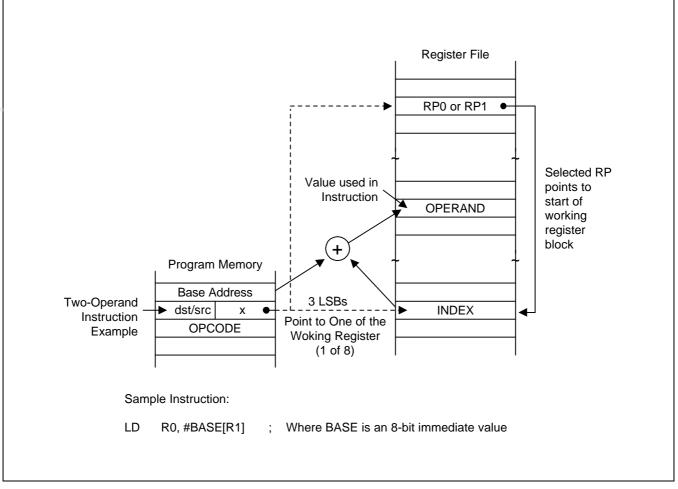


Figure 3-7. Indexed Addressing to Register File



# INDEXED ADDRESSING MODE (Continued)

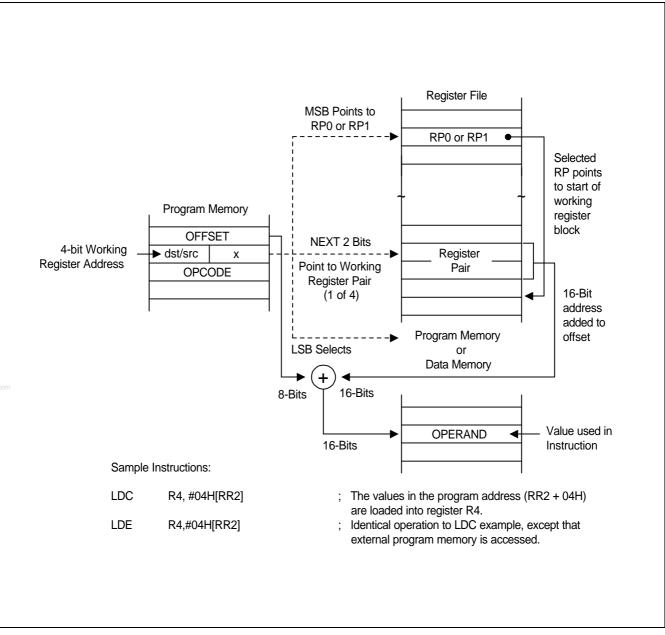


Figure 3-8. Indexed Addressing to Program or Data Memory with Short Offset



# INDEXED ADDRESSING MODE (Concluded)

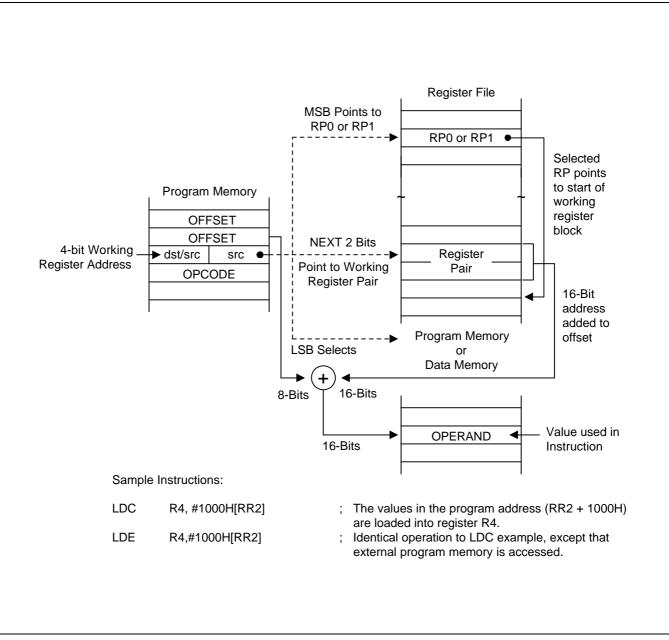


Figure 3-9. Indexed Addressing to Program or Data Memory



### **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.

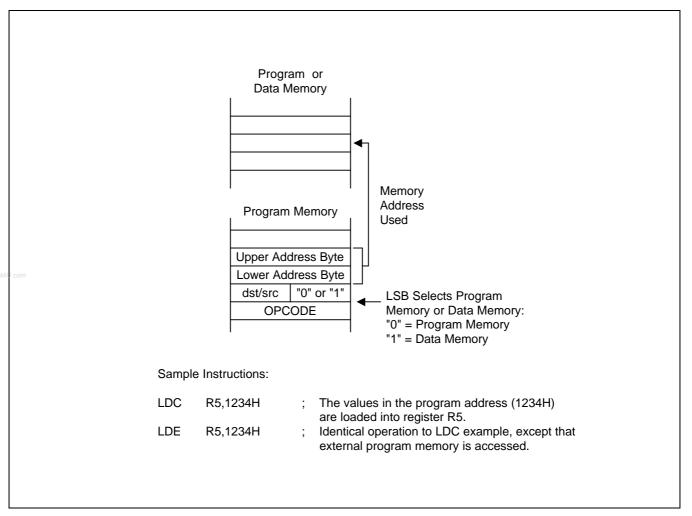


Figure 3-10. Direct Addressing for Load Instructions



# **DIRECT ADDRESS MODE (Continued)**

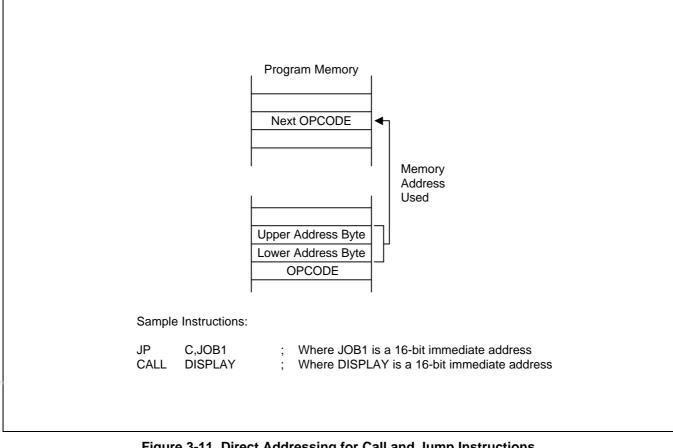


Figure 3-11. Direct Addressing for Call and Jump Instructions



### 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.

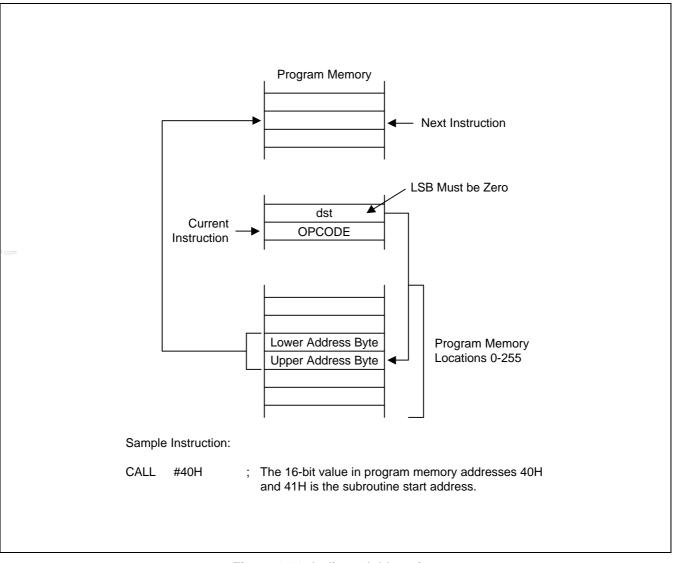


Figure 3-12. Indirect Addressing



## **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.

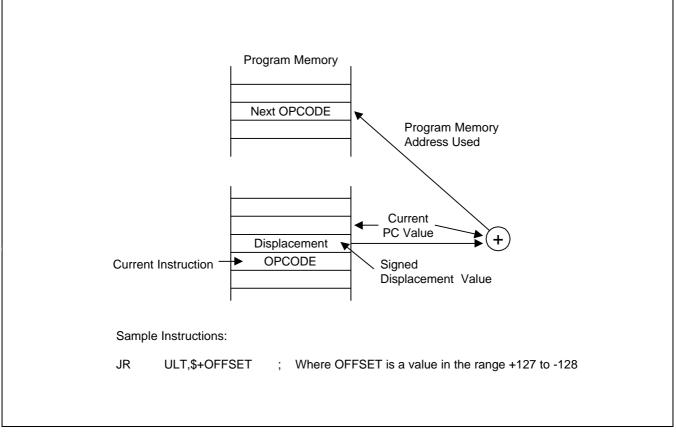
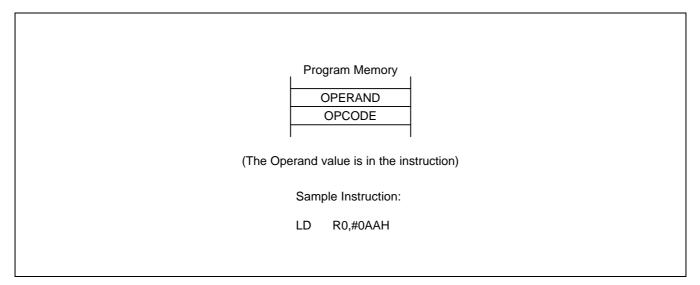


Figure 3-13. Relative Addressing



## **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.







# 4 CONTROL REGISTERS

## OVERVIEW

In this chapter, detailed descriptions of the S3C831B 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 S3C831B 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."

Register Name	Mnemonic	Address		R/W	RES			SET Values (bit)				
		Decimal	Hex		7	6	5	4	3	2	1	0
	Locations D	)H-D2H are	not mapped	d.								
Basic timer control register	BTCON	211	D3H	R/W	0	0	0	0	0	0	0	0
System clock control register	CLKCON	212	D4H	R/W	0	Ι	Ι	0	0	-	-	—
System flags register	FLAGS	213	D5H	R/W	х	х	х	х	х	х	0	0
Register pointer 0	RP0	214	D6H	R/W	1	1	0	0	0	-	-	—
Register pointer 1	RP1	215	D7H	R/W	1	1	0	0	1	Ι	—	—
Stack pointer (high byte)	SPH	216	D8H	R/W	x	х	х	х	х	х	х	х
Stack pointer (low byte)	SPL	217	D9H	R/W	x	х	х	х	х	х	х	х
Instruction pointer (high byte)	IPH	218	DAH	R/W	х	х	х	х	х	х	х	х
Instruction pointer (low byte)	IPL	219	DBH	R/W	х	х	х	х	х	х	х	х
Interrupt request register	IRQ	220	DCH	R	0	0	0	0	0	0	0	0
Interrupt mask register	IMR	221	DDH	R/W	х	х	х	х	х	х	х	х
System mode register	SYM	222	DEH	R/W	0	_	-	х	х	х	0	0
Register page pointer	PP	223	DFH	R/W	0	0	0	0	0	0	0	0

#### Table 4-1. Set 1 Registers



4-1

Register Name	Mnemonic	Add	ress	R/W		RE	SE	τν	alue	es (	bit)	
_		Decimal	Hex		7	6	5	4	3	2	1	0
Timer 0 counter register	TOCNT	224	E0H	R	0	0	0	0	0	0	0	0
Timer 0 data register	TODATA	225	E1H	R/W	1	1	1	1	1	1	1	1
Timer 0 control register	TOCON	226	E2H	R/W	0	0	0	0	0	0	0	0
Timer 1 counter register	T1CNT	227	E3H	R	0	0	0	0	0	0	0	0
Timer 1 data register	T1DATA	228	E4H	R/W	1	1	1	1	1	1	1	1
Timer 1 control register	T1CON	229	E5H	R/W	0	0	0	0	0	0	0	0
Interrupt pending register	INTPND	230	E6H	R/W	-	-	-	-	-	-	0	0
	Location	E7H is not	mapped.		•	•						
Watch timer control register	WTCON	232	E8H	R/W	Ι	0	0	0	0	0	0	0
SIO 0 control register	SIO0CON	233	E9H	R/W	0	0	0	0	0	0	0	0
SIO 0 data register	SIO0DATA	234	EAH	R/W	0	0	0	0	0	0	0	0
SIO 0 prescaler register	SIO0PS	235	EBH	R/W	0	0	0	0	0	0	0	0
SIO 1 control register	SIO1CON	236	ECH	R/W	0	0	0	0	0	0	0	0
SIO 1 data register	SIO1DATA	237	EDH	R/W	0	0	0	0	0	0	0	0
SIO 1 prescaler register	SIO1PS	238	EEH	R/W	0	0	0	0	0	0	0	0
A/D converter control register	ADCON	239	EFH	R/W	Ι	0	0	0	0	0	0	0
A/D converter data register	ADDATA	240	F0H	R	х	х	х	х	х	х	x	х
LCD control register	LCON	241	F1H	R/W	0	-	-	Ι	0	0	0	0
LCD mode register	LMOD	242	F2H	R/W	0	0	0	0	0	0	0	0
IF counter mode register	IFMOD	243	F3H	R/W	0	-	-	0	0	0	0	0
IF counter 1	IFCNT1	244	F4H	R	0	0	0	0	0	0	0	0
IF counter 0	IFCNT0	245	F5H	R	0	0	0	0	0	0	0	0
PLL data register 1	PLLD1	246	F6H	R/W	х	х	х	х	х	х	x	х
PLL data register 0	PLLD0	247	F7H	R/W	х	х	х	х	х	х	x	х
PLL mode register	PLLMOD	248	F8H	(note)				(nc	ote)			
PLL reference frequency register	PLLREF	249	F9H	(note)				(nc	ote)			
	Location FAH is not mapped.											
STOP control register	STPCON	251	FBH	R/W	0	0	0	0	0	0	0	0
	Location	FCH is not	mapped.	1			r					
Basic timer counter	BTCNT	253	FDH	R/W	0	0	0	0	0	0	0	0
	Location	FEH is not	mapped.	1			I				r	
Interrupt priority register	IPR	255	FFH	R/W	х	х	х	х	х	х	x	х

Table 4-2. Set 1, Bank 0 Registers

**NOTE:** Refer to the corresponding register in this chapter.



Devict. N	Mnemonic Address R				V RES			SET Values (bit)				
Register Name	Mnemonic		ress	R/W		1	r	TV		· `	bit)	
		Decimal	Hex		7	6	5	4	3	2	1	0
Port 0 control register (high byte)	P0CONH	224	E0H	R/W	0	0	0	0	0	0	0	0
Port 0 control register (low byte)	P0CONL	225	E1H	R/W	0	0	0	0	0	0	0	0
Port 0 pull-up resistors enable register	P0PUR	226	E2H	R/W	0	0	0	0	0	0	0	0
	Location	E3H is not	mapped.									
Port 1 control register (high byte)	P1CONH	228	E4H	R/W	0	0	0	0	0	0	0	0
Port 1 control register (low byte)	P1CONL	229	E5H	R/W	0	0	0	0	0	0	0	0
Port 1 interrupt control register	P1INT	230	E6H	R/W	0	0	0	0	0	0	0	0
Port 1 interrupt pending register	P1PND	231	E7H	R/W	0	0	0	0	0	0	0	0
Port 2 control register (high byte)	P2CONH	232	E8H	R/W	0	0	0	0	0	0	0	0
Port 2 control register (low byte)	P2CONL	233	E9H	R/W	0	0	0	0	0	0	0	0
Port 3 control register (high byte)	P3CONH	234	EAH	R/W	0	0	0	0	0	0	0	0
Port 3 control register (low byte)	P3CONL	235	EBH	R/W	0	0	0	0	0	0	0	0
Port 3 pull-up resistors enable register	P3PUR	236	ECH	R/W	0	0	0	0	0	0	0	0
Port group 0 control register	PG0CON	237	EDH	R/W	0	0	0	0	0	0	0	0
Port group 1 control register	PG1CON	238	EEH	R/W	0	0	0	0	0	0	0	0
Port group 2 control register	PG2CON	239	EFH	R/W	0	0	0	0	0	0	0	0
Port 0 data register	P0	240	F0H	R/W	0	0	0	0	0	0	0	0
Port 1 data register	P1	241	F1H	R/W	0	0	0	0	0	0	0	0
Port 2 data register	P2	242	F2H	R/W	0	0	0	0	0	0	0	0
Port 3 data register	P3	243	F3H	R/W	0	0	0	0	0	0	0	0
Port 4 data register	P4	244	F4H	R/W	0	0	0	0	0	0	0	0
Port 5 data register	P5	245	F5H	R/W	0	0	0	0	0	0	0	0
Port 6 data register	P6	246	F6H	R/W	0	0	0	0	0	0	0	0
Port 7 data register	P7	247	F7H	R/W	0	0	0	0	0	0	0	0
Port 8 data register	P8	248	F8H	R/W	0	0	0	0	0	0	0	0
	Location	F9H is not	mapped.		•	•						
Timer 2 counter (high byte)	T2CNTH	250	FAH	R	0	0	0	0	0	0	0	0
Timer 2 counter (low byte)	T2CNTL	251	FBH	R	0	0	0	0	0	0	0	0
Timer 2 data register (high byte)	T2DATAH	252	FCH	R/W	1	1	1	1	1	1	1	1
Timer 2 data register (low byte)	T2DATAL	253	FDH	R/W	1	1	1	1	1	1	1	1
Timer 2 control register	T2CON	254	FEH	R/W	0	0	0	0	0	0	0	0
	Location	FFH is not	mapped.									

Table 4-3. Set 1, Bank 1 Registers



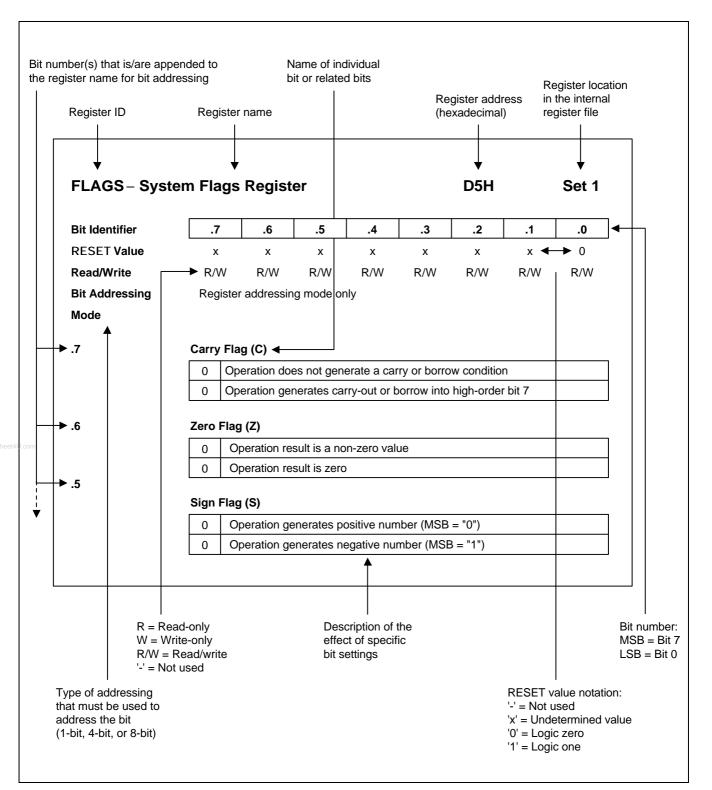


Figure 4-1. Register Description Format



ADCON — A/C	) Conv	erte	r Contro	l Registe	r		EFH	Set	1, Bank (
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	Reg	jister a	addressing	g mode only					
.7	Not	used	for the S3	C831B					
.6–.4	A/D	Inpu	t Pin Sele	ction Bits					
	0	0	0 AD	0 (P2.0)					
	0	0	1 AD	1 (P2.1)					
	0	1	0 AD	2 (P2.2)					
	0	1	1 AD	3 (P2.3)					
	1	0	0 AD	4 (P2.4)					
	1	0	1 AD	5 (P2.5)					
	1	1	0 AD	6 (P2.6)					
	1	1	1 AD	7 (P2.7)					
.3	End		onvorcio	n Bit (road	only)				
.0				n Bit (read- ot complete	oniy)				
	1		version co						
		Con		Inplete					
.2–.1	Clo	ck So	ource Sele	ection Bits					
	0	0	fxx/16						
	0	1	fxx/8						
	1	0	fxx/4						
	1	1	fxx						
.0	Sta	rt or I	Enable Bi	t					
	0	Disa	able opera	tion					
	1	Star	t operatio	n (automatic	ally disable	operation	n after conve	arcian com	nlete)



BTCON — Basic	Tim	er C	ontr	ol R	egister			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	Reg	ister a	addres	ssing	mode only					
.7–.4		1	Ē		unction Dis					
	1	0	1	0			ner functio			
		Otł	ners		Enable wa	atchdog tin	ner functior	1		
.3–.2	Bas	ic Tir	ner Ir	nput (	Clock Sele	ction Bits				
	0	0	fxx/4	4096	(3)					
	0	1	fxx/	1024						
	1	0	fxx/	128						
	1	1	fxx/1	16						
.1	Bas	ic Tir	ner C	ount	er Clear Bi	t <sup>(1)</sup>				
	0	No e	effect							
	1	Clea	ar the	basic	timer cour	nter value				
.0	Clo	ck Fre	equer	ncy D	ivider Clea	ar Bit for a	II timers <sup>(2</sup>	:)		1
	0	No e	effect							
	1	Clea	ar botł	n cloc	k frequency	y dividers				
NOTEO										

#### 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. only for S3C831B).



	ystem	Clo	ck Contro	ol Regist	ter		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	Reg	ister a	addressing	mode only					
.7			r IRQ Wak	•					
	0					/er down m			
	1	Disa	ble IRQ for	r main wak	e-up in pov	ver down m	ode		
	<b></b>								
.6–.5	Not	used	for the S3C	C831B					
.4–.3	CPL	J Clo	ck (System	n Clock) Se	election B	its <sup>(note)</sup>			
	0	0	fxx/16						
	0	1	fxx/8						
	1	0	fxx/2						
	1	1	fxx						
.2–.0	Not	used	for the S30	C831B					

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



FLAGS — Syste	em Fla	ags F	Register				D5H		Set 1
Bit Identifier		7	.6	.5	.4	.3	.2	.1	.0
RESET Value	)	(	х	х	х	х	х	0	0
Read/Write	R/	W	R/W	R/W	R/W	R/W	R/W	R	R/W
Addressing Mode	Regi	ster a	ddressing	mode only					
.7	Carr	y Flag	a (C)						
		-		s not genera	ate a carry	or borrow	condition		
	1			-			high-order t	oit 7	
•			_		.,				
.6		Flag	• •						
	0		ation resu	Ilt is a non-z	zero value				
		Oper	allon resu	III IS Zero					
.5	Sigr	Flag	(S)						
	0	Oper	ation gene	erates a po	sitive numb	per (MSB =	"0")		
	1	Oper	ation gene	erates a ne	gative num	ber (MSB :	= "1")		
<b>4</b>	Ove	rflow	Flag (V)						
	0	Oper	ation resu	It is $\leq$ +12	7 or $\geq -1$	28			
	1	Oper	ation resu	lt is > +12	7 or < -1	28			
.3	Deci	imal A	Adjust Fla	g (D)					
	0	Add	operation	completed					
	1	Subt	raction op	eration com	pleted				
.2	Half	-Carry	/ Flag (H)						
	0	No c	arry-out of	f bit 3 or no	borrow int	o bit 3 by a	ddition or s	ubtraction	
	1	Addi	tion gener	ated carry-o	out of bit 3	or subtract	ion generat	ed borrow	into bit 3
.1	Fast	Inter	rupt Stati	us Flag (Fl	S)				
	0		-	(IRET) in p	•	/hen read)			
	1	Fast	interrupt s	service routi	ine in prog	ress (when	read)		
.0	Ban	k Add	Iress Sele	ection Flag	(BA)				
-	0		0 is selec	-	<u> </u>				
	1		1 is selec						



	ounter	Mod	le Regist	er			F3H	Set	1, Bank 0
Bit Identifier		7	.6	.5	.4	.3	.2	.1	.0
RESET Value	(	0	_	_	0	0	0	0	0
Read/Write	R	/W	_	_	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Reg	ister a	addressing	mode only					
.7	Sys <sup>a</sup> Tim		Clock Cont	rol Bit for	PLL Freq	uency Syn	thesizer, II	- Counter,	Watch
	0	The	supplied cl	ocks are n	ot divided				
	1	The	supplied cl	ocks are d	livided by 2				
.6–.5	Not	used	for the S3C	C831B					
.4	Sele	1	e PLL/IFC	-		e as 4.5V to	o 5.5V.		
	1		ect the PLL/						
<b>.3–.2</b>	Inte 0	rrupt	Sampling IFC is disa	abled; FMI			wn and FMI	F/AMIF's f	eed-back
	0	1	Enable IF	C operatio	n; AMIF pir sistor is off		d; FMIF is p	oulled dow	n and
	1	0			n; FMIF pir sistor is off		d; AMIF is <sub>l</sub>	oulled dow	n and
	1	1	Enable IF	C operatio	n; Both AN	IIF and FM	IF are seled	cted.	
.1–.0	Gate	e Tim	e Selectio	n Bits (fxx	c = 4.5 MHz	<b>z)</b> (note)			
	0	0	Gate oper	ns in 2-mill	isecond int	ervals			
	0	1	Gate oper	ns in 8-mill	isecond int	ervals			
	1	0	Gate oper	ns in 16-m	illisecond ir	ntervals			

Gate remains open continuously

NOTE: If the main clock is 9MHz, IFMOD.7 should be set to "1".

1

1



IMR — Interrupt M	ask	Regist	ter				DDH		Set 1
Bit Identifier		7	.6	.5	.4	.3	.2	.1	.0
RESET Value		x	х	х	х	х	х	х	х
Read/Write	R/	/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Reg	ister ad	dressing	mode only					
.7	Inte	rrupt Lo	evel 7 (IF	RQ7) Enab	le Bit; IF l	nterrupt			
	0	Disabl	e (mask)						
	1	Enable	e (unmas	k)					
.6	Inte	rrupt L	evel 6 (IF	RQ6) Enab	le Bit; CE	Interrupt			
	0	Disabl	e (mask)						
	1	Enable	e (unmas	k)					
.5	Inte	rrupt L	evel 5 (IF	RQ5) Enab	le Bit; P1.	4-P1.7			
	0	Disabl	e (mask)						
	1	Enable	e (unmas	k)					
<b>.4</b>	Inte	rrupt L	evel 4 (IF	RQ4) Enab	le Bit; P1.	0-P1.3			
	0	Disabl	e (mask)						
				k)					
	1	Enable	e (unmas						
.3		I		RQ3) Enab	le Bit; Wa	tch Timer			
.3		rrupt Lo		RQ3) Enab	le Bit; Wa	tch Timer			
.3	Inte	<b>rrupt L</b> o Disabl	evel 3 (IF	RQ3) Enab	le Bit; Wa	tch Timer			
.3	<b>Inte</b> 0 1	rrupt Lo Disabl Enable	<b>evel 3 (IF</b> le (mask) e (unmas	RQ3) Enab k)		tch Timer	nterrupt		
	<b>Inte</b> 0 1	rrupt Lo Disabl Enable rrupt Lo	<b>evel 3 (IF</b> le (mask) e (unmas	RQ3) Enab k) RQ2) Enab			nterrupt		
	Inte 0 1 Inte	rrupt Lo Disabl Enable rrupt Lo Disabl	evel 3 (IF le (mask) e (unmas evel 2 (IF	RQ3) Enab k) RQ2) Enab			nterrupt		
	Inte 0 1 Inte 0 1	rrupt Lo Disabl Enable rrupt Lo Disabl Enable	evel 3 (IF le (mask) e (unmas evel 2 (IF le (mask) e (unmas	RQ3) Enab k) RQ2) Enab k)	le Bit; SIC			pt	
.2	Inte 0 1 Inte 0 1	rrupt Lo Disabl Enable rrupt Lo Disabl Enable	evel 3 (IF le (mask) e (unmas evel 2 (IF le (mask) e (unmas	RQ3) Enab k) RQ2) Enab k) RQ1) Enab	le Bit; SIC	0, SIO 1 II		pt	
.2	Inte 0 1 Inte 0 1 Inte	rrupt Lo Disabl Enable Disabl Enable rrupt Lo Disabl	evel 3 (IF le (mask) e (unmas evel 2 (IF le (mask) e (unmas evel 1 (IF	RQ3) Enab k) RQ2) Enab k) RQ1) Enab	le Bit; SIC	0, SIO 1 II		pt	
.2	Inter           0           1           Inter           0           1           Inter           0           1	rrupt Lo Disabl Enable Disabl Enable rrupt Lo Disabl Enable	evel 3 (IF le (mask) e (unmas evel 2 (IF le (mask) e (unmas evel 1 (IF le (mask) e (unmas	RQ3) Enab k) RQ2) Enab k) RQ1) Enab	le Bit; SIC le Bit; Tim	0, SIO 1 II	r 2 Interru		
.2 .1	Inter           0           1           Inter           0           1           Inter           0           1	rrupt Lo Disable Frupt Lo Disable Frupt Lo Disable Frupt Lo Enable	evel 3 (IF le (mask) e (unmas evel 2 (IF le (mask) e (unmas evel 1 (IF le (mask) e (unmas	RQ3) Enab k) RQ2) Enab k) RQ1) Enab k) RQ0) Enab	le Bit; SIC le Bit; Tim	0, SIO 1 li ner 1, Time	r 2 Interru		

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



	upt l	Pend	ing Regi	ster			E6H	Set	1, Bank 0
Bit Identifier		7	.6	.5	.4	.3	.2	.1	.0
RESET Value		_	_	_	-	_	-	0	0
Read/Write		_	_	_	_	_	_	R/W	R/W
Addressing Mode	Reg	ister a	ddressing	mode only					
.7–.2 .1			for the S3C	C831B	upt Pendii	ng Bit			
	0	Inter	rupt reques	st is not pei	nding (whe	en read), pe	nding bit cl	ear (when	write 0)
	1	Inter	rupt reques	st is pendin	g				
.0	Tim 0	1		nterrupt Pe st is not per		en read), pe	nding bit cl	ear (when	write 0)
	1	Inter	rupt reques	st is pendin	g				

<b>IPH</b> — Instructio	IPH — Instruction Pointer (High Byte)							Set 1	
Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0	
RESET Value	х	х	х	х	х	х	х	х	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Addressing Mode	Register a	addressing	mode only						
.7–.0	The high-		ction pointe	er value is t	he upper e	•	the 16-bit i located in t		

IPL — Instructio	n Pointer (	Low Byte	e)			DBH		Set 1
Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	х	х	х	х	х	х	х	х
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register a	addressing	mode only					

.7–.0

\_

#### Instruction Pointer Address (Low Byte)

register (DBH).

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 Pr	iority	y Re	giste	er				FFH	Set	1, Bank 0
Bit Identifier	-	7		6	.5	.4	.3	.2	.1	.0
RESET Value		x	2	x	х	х	х	х	х	х
Read/Write	R	/W	R/	/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Reg	ister a	addres	ssing	mode only					
.7, .4, and .1	Prio	ority C	ontro	ol Bit	s for Interr	rupt Grou	ps A, B, ar	nd C <sup>(note)</sup>		
	0	0	0	Grou	μp priority ι	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	Grou	μp priority ι	undefined				
.6	Inte	rrupt	Subo	group	C Priority	Control I	Bit			
	0		6 > 1							
	1	IRQ	7 > 1	RQ6						
.5	r			-	Priority Co	ntrol Bit				
	0	-			, IRQ7)					
	1	(IRC	8, IR	Q7) >	> IRQ5					
.3	Inte	rrupt	Subç	group	B Priority	Control I	Bit			
	0	IRQ	3 > II	RQ4						
	1	IRQ	4 >	RQ3						
.2	Inte	rrupt	Grou	ıp B F	Priority Co	ntrol Bit				
	0	IRQ	2 > (	(IRQ3	, IRQ4)					
	1	(IRC	3, IR	Q4) >	> IRQ2					
.0	Inte	rrunt	Grou		Priority Co	ntrol Bit				
.0	0	-	0 > 1	•						
	1		1 > 1							
NOTE: Interrupt Group A - Interrupt Group B - Interrupt Group C -	IRQ2,	IRQ3								



IRQ — Interrupt	Reques	st 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	Regis	ter addressing	mode only	,				
.7	l evel	l 7 (IRQ7) Rec	west Pend	ina Bit <sup>.</sup> IF	Interrupt			
		Not pending						
		Pending						
.6		I 6 (IRQ6) Rec	juest Pend	ing Bit; Ci	= Interrupt			
		Not pending Pending						
		rending						
.5	Level	l 5 (IRQ5) Rec	uest Pend	ing Bit; P1	I.4-P1.7			
	0	Not pending						
	1	Pending						
t41 <b>_4</b> .m	Level	I 4 (IRQ4) Rec	uest Pend	ina Bit: P1	.0-P1.3			
		Not pending						
		Pending						
.3		l 3 (IRQ3) Rec	uest Pend	ing Bit; W	atch Time	r		
		Not pending						
	1	Pending						
.2	Level	l 2 (IRQ2) Rec	juest Pend	ing Bit; SI	O 0, SIO 1	Interrupt		
	0	Not pending						
	1	Pending						
.1	Level	l 1 (IRQ1) Rec	uest Pend	ing Bit; Ti	mer 1, Tim	er 2 Interr	upt	
	0	Not pending						
	1	Pending						
.0	Level	l 0 (IRQ0) Rec	uest Pend	ina Bit: Ti	mer 0 Mate	ch/Capture	e or Overflo	w
		Not pending		<u> </u>				
		Pending						
		i chung						



	ntro	l Reg	giste	er				F1H	Set	1, Bank 0		
Bit Identifier		7		6	.5	.4	.3	.2	.1	.0		
RESET Value	0		-	-	_	_	0	0	0	0		
Read/Write	R/	/W	-	-	_	_	R/W	R/W	R/W	R/W		
Addressing Mode	Reg	ister a	addres	ssing	mode only							
.7	LCD Output Control Bit         0       LCD output is low and current to dividing resistors is cut off											
	1       IF LMOD.3 = "0", LCD display is turned off         IF LMOD.3 = "1", output COM and SEG signals in display mode											
.6–.4	Not	used	for the	e S30	C831B.							
.3–.0	LCD	Port	Sele									
	0	0	0	0	Select LC	D SEG0-3	9					
	0	0	0	1			5/P4.0-4.3					
	0	0	1	0			1/P4 as I/O	•				
	0	0	1	1			7/P4, P5.0-		port			
	0	1	0	0			3/P4, P5 as	•				
	0	1	0	1			9/P4, P5, P		•			
	0	1	1	0	Select LC	D SEG0-1	5/P4, P5, P	6 as I/O po	ort			
	0	1	1	1	Select LCD SEG0-11/P4, P5, P6, P7.0-7.3 as I/O port							
	1	0	0	0			/P4, P5, P6		•			
	1	0	0	1	Select LC	D SEG0-3	/P4, P5, P6	, P7, P8.0-	8.3 as I/O	port		
	1	0	1	0	All I/O por	t (P4-P8)						



	ode(	Cont	rol R	legi	ster			F2H	Set	1, Bank 0
Bit Identifier	<b>.</b>	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	R/W
Addressing Mode	Reg	ister a	addres	ssing	mode only					
.7	<b>COI</b>	COM Signal Enable/Disable Bit         0       Enable COM signal								
	1		able C		-					
		0156			siyilal					
.6	LCD	) Volt	age D	ividi	ng Resisto	or Control	Bit			
	0	Inte	rnal vo	oltage	e dividing re	esistors				
	1	Exte	ernal v	oltag	e dividing i	esistors; ir	nternal volta	age dividing	g resistors	are off
.5–.4	-	1			() Frequen	-	on Bits (W	hen fxx = $\frac{1}{2}$	4.5MHz)	
	0	0			at fxx = $4.5$					
	0	1	-		fxx = 4.5					
	1	0			t fxx = 4.5					
	1	1	500	Hz at	t fxx = 4.5 l	VIHZ				
	NOT	E:	If the n	nain c	lock is 9MH	z, IFMOD.7	should be s	et to "1".		
.3–.0	Dut	Duty and Bias Selection for LCD Display								
	0	х	х	Х	LCD displ	ay off (CO	M and SEG	6 output lov	v)	
	1	0	0	0	1/4 duty,	1/3 bias				
	1	0	0	1	1/3 duty,	1/3 bias				
	1	0	1	1	1/3 duty,	1/2 bias				

1/2 duty, 1/2 bias

0

1

1

0

0

0

Static

1

1



P0CONH-P	ort 0 C	ontr	ol Regis	ter (High	Byte)		E0H	Set	1, Bank		
Bit Identifier	· ·	7	.6	.5	.4	.3	.2	.1	.0		
RESET Value		0	0	0	0	0	0	0	0		
ead/Write	R	/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		
ddressing Mode	Reg	ister	addressing	mode only							
6	P0.7	7/T2C	UT								
	0	0	Input mod	de							
	0	1	Output m	ode, open-	drain						
	1	0	Alternativ	e function	(T2OUT)						
	1	1	Output m	ode, push-	pull						
5–.4	P0.6	6/T2C	1								
	0	0									
	0	1	Output mode, open-drain								
	1	0	Not availa								
	1	1	Output mode, push-pull								
3–.2	P0.5	5/T1C	UT								
	0	0	Input mod	de							
	0	1	Output m	ode, open-	drain						
	1	0	Alternativ	e function	(T1OUT)						
	1	1	Output m	ode, push-	pull						
		•									
I <b>—.0</b>	P0.4	I/T1C	LK								
	0	0	Input mod	de (T1CLK)							
	0	1	Output m	ode, open-	drain						
	1	0	Not availa	able							
	1	1	Output mode, push-pull								



POCONL - Port	t 0 C	ontr	ol Regist	er (Low I	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	Reg	ister a	addressing	mode only								
.7–.6	P0.3/T0OUT/T0PWM											
	0	0	Input mod	le								
	0	1	Output me	ode, open-o	drain							
	1											
	1	1	Output mode, push-pull									
.5–.4	4 P0.2/T0CAP											
	0	0	Input mode (T0CAP)									
	0	1	Output mode, open-drain									
	1	0	Not available									
	1	1	Output mode, push-pull									
.3–.2	<b>P0.</b> 1	/T0C	LK									
	0	0	Input mod	le (T0CLK)								
	0	1		ode, open-o	drain							
	1	0	Not availa									
	1	1	Output me	ode, push-p	bull							
.1–.0	P0.0	)										
	0	0	Input mod	le								
	0	1	Output me	ode, open-o	drain							
	1	0	Not availa	able								
	1	1	Output me	ode, push-p	bull							



POPUR - Port	0 Pull-	up (	Control F	Register			E2H	Set	1, Bank 1
Bit Identifier		7	.6	.5	.4	.3	.2	.1	.0
RESET Value	(	)	0	0	0	0	0	0	0
Read/Write	R/	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Regi	ster a	addressing	mode only					
_									
.7	-			or Enable	Bit				
	0		up disable						
	1	Pull-	up enable						
.6	P0.6	Pull	up Resist	or Enable	Bit				
	0	Pull-	up disable						
	1	Pull-	up enable						
.5			•	or Enable	Bit				
	0		up disable						
	1	Pull-	up enable						
om <b>4</b>	P0.4	Pull	-up Resist	or Enable	Bit				
	0		up disable						
	1	Pull-	up enable						
.3				or Enable	Bit				
	0		up disable						
	1	Pull-	up enable						
.2	P0.2	Pull	-up Resist	or Enable	Bit				
	0	Pull-	up disable						
	1	Pull-	up enable						
.1	P0.1		-	or Enable	Bit				
	0		up disable						
	1	Pull-	up enable						
.0	P0 0	Pull	un Resist	or Enable	Bit				
	0		up disable						
	1		up enable						



	rt 1 Contr	ol Regis	ter (High	Byte)		E4H	Set	1, Bank 1
Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write Addressing Mode	R/W Register a	R/W addressing	R/W mode only	R/W	R/W	R/W	R/W	R/W

.7–.6

### P1.7/INT7

0	0	Input mode; pull-up; interrupt on falling edge
0	1	Input mode; interrupt on rising edge
1	0	Input mode; interrupt on rising or falling edge
1	1	Output mode, push-pull

.5–.4

#### P1.6/INT6

0	0	Input mode; pull-up; interrupt on falling edge
0	1	Input mode; interrupt on rising edge
1	0	Input mode; interrupt on rising or falling edge
1	1	Output mode, push-pull

.3–.2

#### P1.5/INT5

0	0	Input mode; pull-up; interrupt on falling edge
0	1	Input mode; interrupt on rising edge
1	0	Input mode; interrupt on rising or falling edge
1	1	Output mode, push-pull

.1–.0

#### P1.4/INT4

0	0	Input mode; pull-up; interrupt on falling edge
0	1	Input mode; interrupt on rising edge
1	0	Input mode; interrupt on rising or falling edge
1	1	Output mode, push-pull



P1CONL - Po	ort 1 C	ontr	ol Regist	ter (Low	Byte)	E5H	Set	1, Bank 1			
Bit Identifier	<u> </u>	7	.6	.5	.4	.3	.2	.1	.0		
RESET Value		0	0	0	0	0	0	0	0		
Read/Write	R	/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		
Addressing Mode	Register addressing mode only										
.7–.6	P1.3/INT3										
	0	0 Schmitt trigger input mode; pull-up; interrupt on falling edge									
	0	1	Schmitt tr	igger input	mode; inte	rrupt on ris	sing edge				
	1	0	Schmitt tr	igger input	mode; inte	errupt on ris	sing or fallir	ng edge			
	1	1	Output mo	ode, push-p	oull						
.5–.4	P1.2/INT2										
	0	0	Schmitt trigger input mode; pull-up; interrupt on falling edge								
	0	1	Schmitt trigger input mode; interrupt on rising edge								
	1	0	Schmitt trigger input mode; interrupt on rising or falling edge								
	1	1	Output mode, push-pull								
···	P1.1/INT1										
	0	0	0 Schmitt trigger input mode; pull-up; interrupt on falling edge								
	0	1	Schmitt trigger input mode; interrupt on rising edge								
	1	0	Schmitt trigger input mode; interrupt on rising or falling edge								
	1	1 Output mode, push-pull									
.1–.0	P1.0/INT0										
	0	0	Schmitt trigger input mode; pull-up; interrupt on falling edge								
	0	1	Schmitt tr	igger input	mode; inte	errupt on ris	sing edge				
	1	0	Schmitt tr	igger input	mode; inte	errupt on ris	sing or fallir	ng edge			
	1	1	Output mode, push-pull								



4-21

P1INT — Port 1	E6H	Set 1, Bank 1										
Bit Identifier		7	.6	.5	.4	.3	.2	.1	.0			
RESET Value	0		0	0	0	0	0	0	0			
Read/Write	R	/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			
Addressing Mode	Reg	ister a	ddressing									
.7	P1.7 External Interrupt (INT7) Enable Bit											
	0	Disal	ole interrup									
	1	Enab	le interrup	ot								
.6	P1.6 External Interrupt (INT6) Enable Bit											
	0		ole interrup									
	1	Enab	ole interrup	ot								
.5	P1.5 External Interrupt (INT5) Enable Bit											
	0 Disable interrupt											
	1	Enab	ole interrup	ot								
t4 <b>∎4</b> 0m	P1.4 External Interrupt (INT4) Enable Bit											
	0 Disable interrupt											
	1	Enab	le interrup	ot								
.3	P1.3 External Interrupt (INT3) Enable Bit											
	0	Disal	ole interrup	ot								
	1 Enable interrupt											
		LIIAL	ole interrup	ot								
.2		1			Enable Bi	it						
.2		2 Exte		upt (INT2)	Enable Bi	it						
.2	P1.2	2 Exte	rnal Interr	r <b>upt (INT2)</b> ot	Enable Bi	it						
.2 .1	<b>P1.2</b> 0 1	2 Exte Disal Enab	rnal Interr ole interrup le interrup	r <b>upt (INT2)</b> ot ot	) Enable Bi							
	<b>P1.2</b> 0 1	2 Exte Disal Enab	rnal Interr ole interrup le interrup	t <b>upt (INT2)</b> ot ot <b>upt (INT1)</b>								
	P1.2 0 1 P1.1	2 Exter Disal Enab Exter Disal	rnal Interr ole interrup ole interrup	t <b>upt (INT2)</b> ot ot <b>rupt (INT1)</b> ot								
	<b>P1.2</b> 0 1 <b>P1.1</b> 0 1	2 Exter Disal Enab Exter Disal Enab	rnal Interrup ble interrup rnal Interr ble interrup ble interrup	rupt (INT2) ot ot rupt (INT1) ot ot		it						
.1	<b>P1.2</b> 0 1 <b>P1.1</b> 0 1	2 Exter Disal Enab Exter Disal Enab	rnal Interrup ble interrup rnal Interr ble interrup ble interrup	rupt (INT2) ot t <b>upt (INT1)</b> ot ot <b>upt (INT0)</b>	Enable Bi	it						



P1PND — Port	E7H	Set 1, Bank 1										
Bit Identifier		.7	.6	.5	.4	.3	.2	.1	.0			
RESET Value		0	0	0	0	0	0	0	0			
Read/Write	R	/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			
Addressing Mode	Register addressing mode only											
.7	P1.7/INT7 Interrupt Pending Bit											
	0 Interrupt request is not pending, pending bit clear when write 0											
	1	Inter	rupt reques	st is pendin	g							
.6	D1 (		Intorrunt	Ponding P	2;+							
.0	P1.6/INT6 Interrupt Pending Bit         0       Interrupt request is not pending, pending bit clear when write 0											
	1			st is pendin								
					3							
.5	P1.5/INT5 Interrupt Pending Bit											
	0 Interrupt request is not pending, pending bit clear when write 0											
	1 Interrupt request is pending											
on <b>.4</b>	P1.4/INT4 Interrupt Pending Bit											
	0 Interrupt request is not pending, pending bit clear when write 0											
	1 Interrupt request is pending											
.3	P1.3/INT3 Interrupt Pending Bit											
	0 Interrupt request is not pending, pending bit clear when write 0											
	1 Interrupt request is pending											
2	D4 (		Internet	Donalina (	5:4							
.2	P1.2/INT2 Interrupt Pending Bit         0       Interrupt request is not pending, pending bit clear when write 0											
	1 Interrupt request is pending											
			-1		5							
.1	P1.1/INT1 Interrupt Pending Bit											
	0	Inter	rupt reque:	st is not pe	nding, pen	ding bit clea	ar when wr	ite 0				
	1	Inter	rupt reque	st is pendin	g							
.0	P1.(	D/INTO	Interrupt	Pending E	Bit							
	0	1		st is not pe		ding bit clea	ar when wri	ite 0				
	1	Inter	rupt reque	st is pendin	g							

P2CONH-P	E8H	Set 1, Bank 1									
Bit Identifier	· ·	7	.6	.5	.4	.3	.2	.1	.0		
RESET Value		0	0	0	0	0	0	0	0		
Read/Write	R	/W	R/W	R/W	R/W	R/W					
Addressing Mode	Reg	ister	addressing	mode only							
.7–.6	P2.7/AD7										
	0	0	O Schmitt trigger input mode								
	0	1	Schmitt trigger input mode, pull-up								
	1	0	Alternative function (ADC mode)								
	1	1	Output mode, push-pull								
.54	<b>P2.6</b> 0 1 1	0 1 0 1	Schmitt trigger input mode Schmitt trigger input mode, pull-up Alternative function (ADC mode) Output mode, push-pull								
	P2.5	5/AD5	5								
	0	0	Schmitt trigger input mode								
	0	1	Schmitt trigger input mode, pull-up								
	1	0	Alternativ	e function (	ADC mode	e)					
	1	1	Output me	ode, push-p	bull						
.1–.0	P2.4	I/AD4	L								
	0	0	Schmitt tr	igger input	mode						
	0	1	Schmitt tr	igger input	mode, pull	l-up					
	4	~	Alternative function (ADC mode)								

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input mode, pull-up
1	0	Alternative function (ADC mode)
1	1	Output mode, push-pull



P2CONL - Po	E9H	Set 1, Bank 1									
Bit Identifier		7	.6	.5	.4	.3	.2	.1	.0		
RESET Value		0	0	0	0	0	0	0	0		
Read/Write	R	/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		
Addressing Mode	Register addressing mode only										
.7–.6	P2.3/AD3										
	0	0	Schmitt tr	igger input							
	0	1	Schmitt trigger input mode, pull-up								
	1	0	Alternative function (ADC mode)								
	1	1	Output mode, push-pull								
.5–.4	P2.2/AD2										
	0	0	Schmitt trigger input mode								
	0	1	Schmitt trigger input mode, pull-up								
	1	0	Alternative function (ADC mode)								
	1	1	Output mode, push-pull								
·····	P2.1/AD1										
	0	0	Schmitt trigger input mode								
	0	1	Schmitt trigger input mode, pull-up								
	1	0	Alternative function (ADC mode)								
	1	1	Output me	ode, push-p	bull						
1 0	<b>D</b> 2 (										
.1–.0		P2.0/AD0									
	0	0	Schmitt trigger input mode Schmitt trigger input mode, pull-up								
	1	0		e function (		-					
	1	1				=)					
	I		Output mode, push-pull								



P3CONH – P	ort 3 C	ontr	ol Regist	ter (High	Byte)		EAH	Set	1, Bank 1		
Bit Identifier		7	.6	.5	.4	.3	.2	.1	.0		
RESET Value		0	0	0	0	0	0	0	0		
Read/Write	R	/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		
Addressing Mode	Reg	ister a	addressing	mode only							
.7–.6	P3.7	P3.7									
	0										
	0	1	Output me	ode, open-o	drain						
	1	0	Not availa	able							
	1	1	Output me	ode, push-p	oull						
.5–.4	P3.6	6/SI1									
	0	0	Input mod	le (SI1)							
	0	1	Output me	ode, open-o	drain						
	1	0	Not availa	able							
	1	1	Output me	ode, push-p	bull						
<sup>t4U.com</sup> .3–.2	P3.5	5/SO1									
	0	0	Input mod	le							
	0	1	Output me	ode, open-o	drain						
	1	0	Alternativ	e function (	(SO1)						
	1	1	Output me	ode, push-p	bull						
.1–.0	P3.4	I/SCK	(1								
	0	0	1	le (SCK1)							
	0	1		ode, pull-up	)						
		<u> </u>	1								

NOTE: The SO1 and SCK1 outputs are selected as push-pull or open-drain by "PG2CON".

Alternative function (SCK1 out)

Output mode, push-pull

1

1

0

1



RESET Value 0 0 0 0 0 0 0										
Read/WriteR/W <th>.0</th>	.0									
	0									
Addressing Mode Register addressing mode only	/W									
.7–.6 P3.3/SIO										
0 0 Input mode (SI0)										
0 1 Output mode, open-drain										
1 0 Not available										
1 1 Output mode, push-pull										
.5–.4 P3.2/SO0										
0 0 Input mode										
0 1 Output mode, open-drain										
1 0 Alternative function (SO0)										
1 1 Output mode, push-pull										
.3–.2 P3.1/SCK0										
0 0 Input mode (SCK0)										
0 1 Output mode, open-drain										
1 0 Alternative function (SCK0 out)										
1 1 Output mode, push-pull										
.1–.0 P3.0/BUZ										
0 0 Input mode										
0 1 Output mode, open-drain										
1 0 Alternative function (BUZ)										
1 1 Output mode, push-pull										

NOTE: The SO0 and SCK0 outputs are selected as push-pull or open-drain by "PG2CON".



<b>P3PUR</b> — Port 3	3 Pull-ւ	ip Control F	Register			ECH	Set	1, Bank 1
Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/V	V R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Regis	ter addressing	mode only					
.7	P3.7 I	Pull-up Resist	or Enable	Bit				
	0 I	Pull-up disable	1					
	1	Pull-up enable						
.6	P3.6 I	Pull-up Resist	or Enable	Bit				
	0 1	Pull-up disable	1					
	1	Pull-up enable						
.5	P3.5 I	Pull-up Resist	or Enable	Bit				
	0	Pull-up disable						
	1 I	Pull-up enable						
4 <sup>U</sup> <b>4</b>	P3.4 I	Pull-up Resist	or Enable	Bit				
	1	Pull-up enable						
.3	P3.3 I	Pull-up Resist	or Enable	Bit				
-		Pull-up disable						
	-	· Pull-up enable						
.2	P3.2 I	Pull-up Resist	or Enable	Bit				
		Pull-up disable						
	1	Pull-up enable						
4	D2 4 1	Pull-up Resist	or Enchla	D:4				
.1		Pull-up disable		ы				
		Pull-up enable						
.0		Pull-up Resist		Bit				1
		Pull-up disable						
	1	Pull-up enable						



Bank	Set 1	EDH		r	Registe	0 Control	oup (	rt Gro	PG0CON - Por
.0	.1	.2	.3	.4	.5	.6	.7		Bit Identifier
0	0	0	0	0	0	0	0	(	RESET Value
R/W	R/W	R/W	R/W	R/W	R/W	R/W	/W	R/	Read/Write
					mode only	addressing	ister a	Reg	Addressing Mode
		)-P4.3	P4.0	.7–.6					
put mode									
put mode, pull-up									
			Open-drai	0	1				
			Push-pull	1	1				
		4-P4.7	P4 4	.5–.4					
						Input mod	0	0	
			Open-drai	0	1				
					output mod		1	1	
			s	ode	e e, pull-up n output m		0 1 0	0 0 1	.3–.2
			s			7/SEG27-24	1		.1–.0
						Input mod			
								0	
							0	1	
				le	output mod	Push-pull	1	1	
-					e, pull-up n output m output moo		1 0 1		



PG1CON - Por	t Gro	oup	1 Contro	l Registe	r		EEH	Set	1, Bank 1			
Bit Identifier	<b>.</b>	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	Reg	ister a	addressing	mode only								
.7–.6	P6.0	P6.0-P6.3/SEG23-20 Mode Selection Bits										
	0											
	0	1     Schmitt trigger input mode, pull-up										
	1	0		in output m	· · ·							
	1	1	Push-pull	output mod	de							
			•									
.5–.4	P6.4-P6.7/SEG19-16 Mode Selection Bits											
	0	0	Schmitt tr	igger input	mode							
	0	1	Schmitt tr	igger input	mode, pul	l-up						
	1	0	Open-dra	in output m	ode							
	1	1	Push-pull	output mod	de							
.3 <b>–</b> .2	P7.0	)-P7.3	3/SEG15-12	2 Mode Se	lection Bit	S						
	0	0		igger input								
	0	1		igger input		l-up						
	1	0	· ·	in output m								
	1	1	Push-pull	output mod	de							
.1–.0	P7.4	I-P7.7	7/SEG11-8	Mode Sele	ection Bits	5						
	0	0	Schmitt tr	igger input	mode							
	0	1		igger input		l-up						
	1	0		in output m								
	1	1	Push-pull	output mod	de							



PG2CON-P	ort Gro	oup	2 Contro	l Registe	er		EFH	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	Reg	ister a	addressing	mode only							
.7	SIO	1 Out	put Contr	ol Bit							
	0	SO1	, SCK1 ou	tput is sele	cted as pu	sh-pull.					
	1	SO1	, SCK1 ou	tput is sele	cted as op	en-drain.					
.6	SIO	0 Out	put Contre	ol Bit							
	0	SOC	), SCK0 ou	tput is sele	cted as pu	sh-pull.					
	1	SOC	), SCK0 ou	tput is sele	cted as op	en-drain.					
.5	<b>P1.4</b>	P1.4-P1.7 Input Enable Bits         0       Port 1.4-1.7 input enable									
	1	Port	1.4-1.7 inp	out disable							
<sup>om</sup> .4	P1.0	)-P1.3	3 Input Ena	able Bits							
	0	Port	1.0-1.3 inp	out enable							
	1	Port	1.0-1.3 inp	out disable							
.3–.2	P8.0	)-P8.3	8/SEG7-4 N	/lode Sele	ction Bits						
	0	0	Schmitt tr	igger input	mode						
	0	1	Schmitt tr	igger input	mode, pul	l-up					
	1	0	Open-dra	in output m	node						
	1	1	Push-pull	output mo	de						
.1–.0	P8 4	L-P8 7	//SEG3-0 M	/lode Sele	tion Bits						
	0	0	1	igger input							
	0	1		igger input		l-up					
	1	0		in output m							
		<u> </u>	+ •	•							

Push-pull output mode

1

1



PLLMOD – P	LL Mo	de R	legister				F8H	Set	1, Bank 0		
Bit Identifier	· ·	.7	.6	.5	.4	.3	.2	.1	.0		
RESET Value	(n	ote)	(note)	(note)	(note)	0	0	0	0		
Read/Write	R	/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		
Addressing Mode	Reg	jister	addressing	mode only							
.7 and .4	PLL	. Frec	quency Div	vision Meth	nod Select	ion Bits					
	0	0	Direct me	thod for VC	COAM inpu	it (0.5 to 30	MHz)				
	0	1	Enable 3-	bit counter	for VCOA	M input (0.	5 to 30MHz	)			
	1	0	Pulse swa	allow metho	od for VCO	AM input (	0.5 to 30MH	Hz)			
	1	1	Pulse swa	allow metho	od for VCO	FM input (	30 to 150M	Hz)			
.6	PLL	PLL Enable/Disable Bit									
	0										
	1										
.5		Bit Value to be Loaded into PLLD0 Register NF bit is loaded into the LSB of swallow counter									
2		15 14		bla Dit							
.3		-	<b>errupt Ena</b> able INTIF i								
	1		ble INTIF i								
.2	INT	IF Int	errupt Pen	ding Bit							
	0	Inte	rrupt is not	pending (w	hen read)						
	0	Clea	ar pending	bit (when w	vrite)						
	1	Inte	rrupt is pen	ding (when	read)						
.1	INT	CE In	terrupt En	able Bit							
	0	1		interrupt re	equests at	the CE pin					
	1	-		interrupt re							
0	INT	CE In	terrunt Pe	ndina Rit							
.0		1	terrupt Pe	-	hen read)				]		
.0	<b>INT</b> 0 0	Inte	rrupt is not	nding Bit pending (w bit (when w	,						

**NOTE:** If a system reset occurs during operation mode, the current value contained is retained. If a system reset occurs after power-on, the value is undefined.



	Refe	erenc	e Fr	equ	ency Sel	ection R	egister	F9H	Set	1, Bank 0		
Bit Identifier	-	7	-	6	.5	.4	.3	.2	.1	.0		
RESET Value	(	1)	(*	1)	(1)	(2)	(1)	(1)	(1)	(1)		
Read/Write	I	R	F	२	R	R/W	R/W	R/W	R/W	R/W		
Addressing Mode	Reg	ister a	addres	ssing	mode only							
.7	PLL	Freq	uenc	y Syı	nthesizer L	.ocked/Un	locked Sta	atus Flag				
	0											
	1 PLL is currently in unlocked state											
.6	CE Pin level Status Flag											
	0	1			ntly low lev	el						
	1											
	1 CE pin is currently high level											
.5	IF Counter Gate Open/Close Status Flag											
	0	Gate	e is cu	irrent	ly open							
	1	Gate	e is cu	irrent	ly close							
.4	Ром	ver or	n Flag	(3)								
	0	Clea	r pow	/er-or	n flag bit (w	hen write)						
	1	Pow	er-on	occu	rred (when	read)						
.3 – .0	Def		- <b>-</b>			D:40 //M	lhon furr	(4)				
.5 – .0		1		-	cy Selectio	•	nen txx= 4	+.3IVINZ) (*)				
	0	0	0	0 1	1-kHz sigi							
	0	0	1	0	3-kHz sigi 5-kHz sigi							
	0	0	1	1	6.25-kHz							
	0	1	0	0	9-kHz sigi							
	0	1	0	1	10-kHz sig							
	0	1	1	0	12.5-kHz							
	0	1	1	1	25-kHz sig	•						
	1	0	0	0	50-kHz sig	-						
	1	0	0	1	100-kHz s	-						
NOTES:	L											

#### NOTES:

- 1. If a system reset occurs during operation mode, the current value contained is retained. If a system reset occurs after power-on, the value is undefined.
- 2. If a system reset occurs during operation mode, the current value contained is retained. If a system reset occurs after power-on, the value is "1".
- 3. The POF bit is read initially to check whether or not power has been turned on.
- 4. If the main clock is 9MHz, IFMOD.7 should be set to "1".



<b>PP</b> — Register Pa	ge Pointe		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 a	addressing	mode only					

.7–.4

# **Destination Register Page Selection Bits**

0	0	0	0	Destination: page 0
0	0	0	1	Destination: page 1
0	0	1	0	Destination: page 2
0	0	1	1	Destination: page 3
0	1	0	0	Destination: page 4
0	1	0	1	Destination: page 5
0	1	1	0	Destination: page 6
0	1	1	1	Destination: page 7
1	0	0	0	Destination: page 8
1	0	0	1	Destination: page 9

.3 – .0

# Source Register Page Selection Bits

		-		
0	0	0	0	Source: page 0
0	0	0	1	Source: page 1
0	0	1	0	Source: page 2
0	0	1	1	Source: page 3
0	1	0	0	Source: page 4
0	1	0	1	Source: page 5
0	1	1	0	Source: page 6
0	1	1	1	Source: page 7
1	0	0	0	Source: page 8
1	0	0	1	Source: page 9

**NOTE:** In the S3C831B microcontroller, the internal register file is configured as ten pages (Pages 0-9). The pages 0-8 are used for general purpose register file, and page 9 is used for LCD data register or general purpose registers.



<b>RP0</b> — Register	Pointer 0		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 a	Register addressing only									
.7–.3	Register p areas in th two 8-byte	pointer 0 ca ne register e register s s to addres	file. Using lices at one	dently poin the registe e time as a	t to one of t r pointers R ctive workin 1, selecting	P0 and RF	P1, you car space. Afte	n select er a reset,			
.2–.0	Not used	for the S30	C831B								

RP1 — Register I	Pointer 1		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 a	addressing	only					
.7 – .3	Register p areas in th two 8-byte	pointer 1 ca ne register e register s s to addres	file. Using t lices at one	dently point the register time as ac	to one of t pointers R tive workin 1, selecting	P0 and RP g register s	1, you can space. Afte	select r a reset,
.2 – .0	Not used	for the S3C	C831B					



SIO0CON - sid	C 0 C	ontr	ol Regis	ster			E9H	Set	1, Bank 0
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	R/W
Addressing Mode	Reg	ister a	addressing	mode only	,				
.7	SIO	0 Shi	ft Clock S	election B	Bit				
	0	Inter	nal clock (	P.S clock)					
	1	Exte	rnal clock	(SCK0)					
.6	Data	a Dire	ction Con	trol Bit					
	0	MSE	B-first mode	e					
	1	LSB	-first mode						
.5	SIO	0 Mo	de Selecti	on Bit					
	0	Rece	eive-only m	node					
	1	Tran	smit/receiv	ve mode					
.4	Shif	t Cloc	ck Edge S	election B	it				
	0	Tx a	t falling ed	ges, Rx at	rising edge	S			
	1	Tx a	t rising edg	ges, Rx at f	falling edge	S			
.3	SIO	0 Coi	unter Clea	r and Shif	t Start Bit				
	0	No a	iction						
	1	Clea	r 3-bit cou	nter and st	art shifting				
.2	SIO	0 Shi	ft Operati	on Enable	Bit				
	0	Disa	ble shifter	and clock (	counter				
	1	Enat	ole shifter a	and clock c	counter				
.1	SIO	0 Inte	errupt Ena	ble Bit					
	0	Disa	ble SIO 0 l	Interrupt					
	1	Enat	ole SIO 0 I	nterrupt					
.0	SIO	0 Inte	errupt Pen	ding Bit					
	0	No ir	nterrupt pe	nding					
	0	Clea	r pending (	condition (v	when write)				
	1	Inter	rupt is pen	ding					



SIO1CON-s	10 1 0	Contr	ol Regis	ster			ECH	Set	1, Bank 0
Bit Identifier	· ·	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value		0	0	0	0	0	0	0	0
Read/Write	R	/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Reg	ister a	addressing	mode only					
.7	SIO	1 Shi	ft Clock S	election B	it				
	0	Inter	nal clock (	P.S clock)					
	1	Exte	rnal clock	(SCK1)					
.6	Data	a Dire	ction Con	trol Bit					
	0	MSE	B-first mode	e					
	1	LSB	-first mode	1					
.5	SIO	1 Mo	de Selecti	on Bit					
	0	Rece	eive-only m	node					
	1	Tran	smit/recei	ve mode					
.4	Shif	ft Clo	ck Edge S	election B	it				
	0	Tx a	t falling ed	ges, Rx at	rising edge	S			
	1	Tx a	t rising edg	ges, Rx at f	alling edge	S			
.3	SIO	1 Co	unter Clea	r and Shif	t Start Bit				
	0	No a	action						
	1	Clea	r 3-bit cou	nter and sta	art shifting				
.2	SIO	1 Shi	ift Operati	on Enable	Bit				
	0	Disa	ble shifter	and clock o	counter				
	1	Enal	ole shifter a	and clock c	ounter				
.1	SIO	1 Inte	errupt Ena	ble Bit					
	0	Disa	ble SIO 1 I	Interrupt					
	1	Enal	ole SIO 1 li	nterrupt					
.0	SIO	1 Inte	errupt Pen	ding Bit					
	0	1	nterrupt pe						
	0	Clea	r pending (	condition (w	vhen write)				
	1	Inter	rupt is pen	ding					



Set 1

SPH — Stack Po	ointer (High	D8H			Set 1			
Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	х	х	х	х	х	х	х	х
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register a	addressing	mode only					
.7–.0	Stack Po	inter Addr	ess (High	Byte)				
	address (	SP15–SP8	•	er byte of th	ne stack po	inter value	6-bit stack is located i	

# **SPL** — Stack Pointer (Low Byte)

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	х	х	х	х	х	х	х	х
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register a	ddressing	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.

D9H



STPCON – Sto	FBH	Set	1, Bank 0					
Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register a	ddressir	ng mode only					
.7–.0	STOP Co	ntrol Bi	ts					
	10100	101	Enable stop					
	Other va	alues	Disable stop					

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



SYM — System I	Mode	Regi	ster					DEH		Set 1	
Bit Identifier	-	7	-	6	.5	.4	.3	.2	.1	.0	
RESET Value	(	0	-	-	_	х	х	х	0	0	
Read/Write	R	/W	-	-	_	R/W	R/W	R/W	R/W	R/W	
Addressing Mode	Reg	ister a	addres	ssing n	node only						
.7	Not	used,	But y	ou mu	st keep "(	)"					
.6–.5	Not	used	for the	e S3C8	331B.						
.4–.2	Fas		-	1	Selectior	n Bits <sup>(1)</sup>					
	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	Fas	t Inte	rrupt	Enable	e Bit <sup>(2)</sup>						
	0	Disa	ble fa	ist inte	rrupt proc	essing					
	1	Ena	ble fa	st inter	rupt proce	essing					
.0	Glo	bal In	terru	pt Ena	ble Bit <sup>(3</sup>	)					
	0	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).



TOCON — Timer	0 Co	ontro	ol Re	giste	er			E2H	Set	1, Bank 0
Bit Identifier		7		6	.5	.4	.3	.2	.1	.0
RESET Value	(	C	(	)	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	Reg	ister a	addres	ssing	mode only					
.7–.5	Time	er O lı	nput	Clock	Selectior	n Bits				
	0	0	0	fxx/1	024					
	0	0	1	fxx/2	256					
	0	1	0	fxx/6	64					
	0	1	1	fxx/8	3					
	1	0	0	fxx						
	1	0	1		rnal clock	· /	<u> </u>			
	1	1	0		rnal clock (	(T0CLK) ris	sing edge			
	1	1	1	Cour	nter stop					
.4–.3	Time 0	er O C	- -	t <b>ing I</b> val m	Mode Sele	ction Bits				
	0	1				ire on risin	g edge, co	unter runnir		
	1	0	· ·		· ·		ng edge, co		-	
	1	1					errupt can o			
		•	1 001				inupt can o			
.2	Time	er 0 C	Count	er Cle	ear Bit <sup>(not</sup>	e)				
	0		effect							
	1			timer	0 counter (	when write	e)			
						<u> </u>	,			
.1	Time	er O N	<b>/</b> atch	/Capt	ure Interro	upt Enable	e Bit			
	0	Disa	ble in	terrup	ot					
	1	Ena	ble int	terrup	t					
.0	Time	er 0 C	Overfl	low In	nterrupt Er	nable				
	0	Disa	ble o	verflo	w interrupt					
	1	Ena	ble ov	verflov	v interrupt					
NOTE: When you write a " operation, the T0C0							red to "00H"	. Immediate	ly following t	he write

T1CON — Timer	T1CON — Timer 1 Control Register										
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	R/W	
Addressing Mode	Reg	ister a	addres	ssing i	mode only						
.7–.5	Tim	er 1 lı	nput	Clock	Selectior	Bits					
	0	0	0	fxx/2	56						
	0	0	1	fxx/6	4						
	0	1	0	fxx/8	6						
	0	1	1	fxx							
	1	1	1	Exte	rnal clock (	(T1CLK) in	put				
.4 .3 ::4U.com .2	<b>Tim</b> 0 1	er 1 C No e Clea	<b>Count</b> effect ar the	timer	831B ear Bit <sup>(Not</sup> 1 counter ( able Bit		e)				
	0	Disa	ble co	ountin	g operatior	า					
	1	Enal	ble co	ounting	g operation	1					
.1	<b>Tim</b> 0 1	Disa	ble tir	mer 1	nable Bit interrupt interrupt						
.0	Tim	er 1 l	nterru	upt Pe	ending Bit						
	0				rupt pendir	• •	,				
	0					nding condi	ition (when	write)			
	1	T1 ir	nterru	pt is p	ending						

**NOTE:** When you write a "1" to T1CON.3, the timer 1 counter value is cleared to "00H". Immediately following the write operation, the T1CON.3 value is automatically cleared to "0".



T2CON — Time	T2CON — Timer 2 Control Register										
Bit Identifier		.7	.6	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	Reg	jister a	addres	sing ı	mode only						
.7–.5	Tim	er 2 l	nput (	Clock	Selection	n Bits					
	0	0	0	fxx/2	56						
	0	0	1	fxx/6	4						
	0	1	0	fxx/8							
	0	1	1	fxx							
	1	1	1	Exte	rnal clock	(T2CLK) in	nput				
.4 .3	<b>Tim</b> 0 1	er 2 C No e Clea	Count effect ar the f	er Cle	831B. ear Bit <sup>(Not</sup> 2 counter able Bit	e) (when write	e)				
.2	0	1			g operation	n					
	1				g operation						
.1					nable Bit	ı <u> </u>				]	
	0	Disa	ble tin	ner 2	interrupt						
	1	Ena	ble tin	ner 2 i	interrupt						
.0	Tim	er 2 l	nterru	ipt Pe	ending Bit	:					
	0	No t	imer 2	inter	rupt pendi	ng (when r	ead)				
	0	Clea	ar time	er 2 in	terrupt per	nding bit (w	/hen write)				
	1	T2 ir	nterrup	ot is p	ending						

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



WTCON — Wat	ch Ti	mer	Control	Register			E8H	Set	1, Bank 0			
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	Reg	gister addressing mode only										
.7	Not	Not used for the S3C831B.										
.6	Wat	tch Ti	mer Enabl	le Bit								
	0	Disa	able watch	timer; clea	r frequency	dividing ci	rcuits					
	1	Ena	ble watch t	imer								
	_	_										
.5–.4			- -		(When fxx	-						
	0	0		· · /	signal outp							
	0	1			Z) signal ou	-						
	1	0		. ,	signal outp							
					Signal Outp	Jul						
.t <sup>4U.com</sup> .2	Wat	tch Ti	mer Speed	d Selectio	n Bits (Wh	en fxx=4.5	MHz)					
	0	0	1.0 s Inte	rval								
	0	1	0.5 s Inte	rval								
	1	1	50 ms Int	erval								
.1	Wat	1	mer Interr	-								
	0	Disa	able watch	timer interr	upt							
	1	Ena	ble watch t	imer interro	upt							
0		ah T	marluta		na Bit							
.0		1	mer Interr	•	-				]			
	1	-	ar pending		,							
				•	,							
		1 Interrupt is pending (when read)										

NOTE: If the main clock is 9MHz, IFMOD.7 should be set to "1".



# 5 INTERRUPT STRUCTURE

# OVERVIEW

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

#### Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight possible interrupt levels: IRQ0–IRQ7, also called level 0–level 7. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. The S3C831B 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. S3C831B uses seventeen 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 S3C831B interrupt structure, there are seventeen 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 S3C831B microcontroller, two interrupt types are implemented.

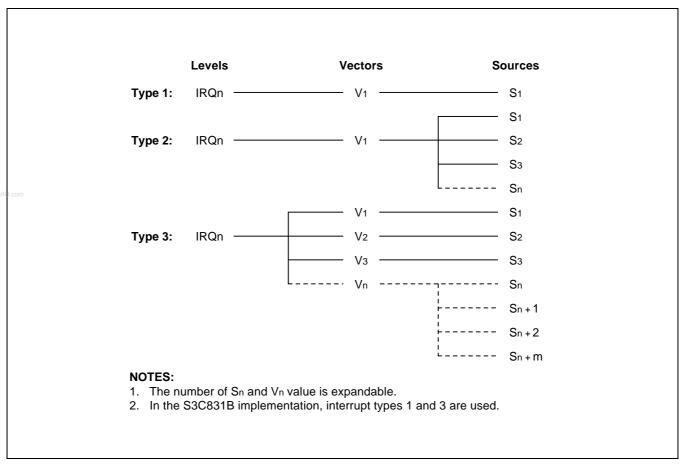


Figure 5-1. S3C8-Series Interrupt Types



#### S3C831B INTERRUPT STRUCTURE

The S3C831B microcontroller supports seventeen interrupt sources. All seventeen 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
	— E0H ———	Timer 0 match/capture	S/W
IRQ0	— E2H ———	Timer 0 overflow	H/W,S/W
	— E4H ———	Timer 2 match	S/W
IRQ1	— E6H ———	Timer 1 match	S/W
IRQ2 —	— E8H ———	SIO1 interrupt	S/W
	— EAH ———	SIO0 interrupt	S/W
IRQ3	— F2H ———	Watch timer	S/W
Γ_	— D0H ———	P1.0 external interrupt	S/W
IRQ4	— D2H ———	P1.1 external interrupt	S/W
	— D4H ———	P1.2 external interrupt	S/W
	— D6H ———	P1.3 external interrupt	S/W
<b></b>	— D8H ———	P1.4 external interrupt	S/W
	— DAH ———	P1.5 external interrupt	S/W
IRQ5 —	— DCH ———	P1.6 external interrupt	S/W
	— DEH ———	P1.7 external interrupt	S/W
IRQ6	— СОН ———	CE interrupt	S/W
IRQ7	— C2H ———	IF interrupt	S/W
For example The priorities 2. External inte	, E0H has higher priority the within each level are set a	sing or falling edge, dependir	Ĵ.

corresponding control register setting.

Figure 5-2. S3C831B Interrupt Structure



#### INTERRUPT VECTOR ADDRESSES

All interrupt vector addresses for the S3C831B interrupt structure are stored in the vector address area of the first 256 bytes of the program memory (ROM).

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

The program reset address in the ROM is 0100H.

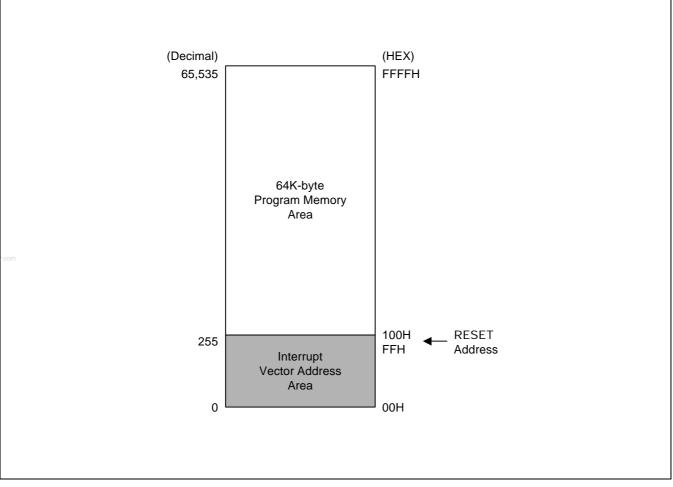


Figure 5-3. ROM Vector Address Area



Vector Address		Interrupt Source	Req	uest	Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
256	100H	Basic timer overflow	RESET	-	$\checkmark$	
226	E2H	Timer 0 overflow	IRQ0	1	$\checkmark$	$\checkmark$
224	E0H	Timer 0 match/capture		0		$\checkmark$
230	E6H	Timer 1 match	IRQ1	1		$\checkmark$
228	E4H	Timer 2 match		0		$\checkmark$
234	EAH	SIO0 interrupt	IRQ2	1		$\checkmark$
232	E8H	SIO1 interrupt	0		$\checkmark$	
242	F2H	Watch timer	IRQ3	_		$\checkmark$
214	D6H	P1.3 external interrupt	IRQ4	3		$\checkmark$
212	D4H	P1.2 external interrupt		2		$\checkmark$
210	D2H	P1.1 external interrupt 1		1		$\checkmark$
208	D0H	P1.0 external interrupt 0		0		$\checkmark$
222	DEH	P1.7 external interrupt IRQ5 3			$\checkmark$	
220	DCH	P1.6 external interrupt 2			$\checkmark$	
218	DAH	P1.5 external interrupt 1			$\checkmark$	
216	D8H	P1.4 external interrupt 0			$\checkmark$	
192	C0H	CE interrupt IRQ6 –			$\checkmark$	
194	C2H	IF interrupt	IRQ7	-		$\checkmark$

# Table 5-1. Interrupt Vectors

#### NOTES:

1. Interrupt priorities are identified in inverse order: "0" is the highest priority, "1" is the next highest, and so on.

2. If two or more interrupts within the same level contend, the interrupt with the lowest vector address usually has priority over one with a higher vector address. The priorities within a given level are fixed in hardware.



# **ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)**

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

# NOTE

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

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

# SYSTEM-LEVEL INTERRUPT CONTROL REGISTERS

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

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

Control Register	ID	R/W	Function Description	
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable or disable interrupt processing for each of the eight interrupt levels: IRQ0–IRQ7.	
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The eight levels of S3C831B 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		R/W	This register enables/disables fast interrupt processing, and dynamic global interrupt processing.	

#### Table 5-2. Interrupt Control Register Overview

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.

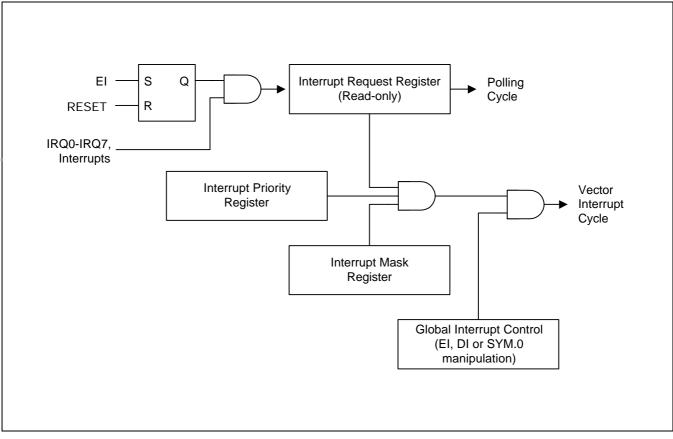


Figure 5-4. Interrupt Function Diagram



# 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).

Interrupt Source	Interrupt Level	Register(s)	Location(s) in Set 1
Timer 0 overflow Timer 0 match/capture	IRQ0	T0CON T0CNT T0DATA INTPND	E2H, bank 0 E0H, bank 0 E1H, bank 0 E6H, bank 0
Timer 1 match	IRQ1	T1CON T1CNT T1DATA	E5H, bank 0 E3H, bank 0 E4H, bank0
Timer 2 match		T2CON T2CNTH, T2CNTL T2DATAH, T2DATAL	FEH, bank 1 FAH, FBH, bank 1 FCH, FDH, bank 1
SIO0 interrupt SIO1 interrupt	IRQ2	SIO0CON SIO0DATA SIO0PS SIO1CON SIO1DATA SIO1PS	E9H, bank 0 EAH, bank 0 EBH, bank 0 ECH, bank 0 EDH, bank 0 EEH, bank 0
Watch timer	IRQ3	WTCON	E8H, bank 0
P1.3 external interrupt P1.2 external interrupt P1.1 external interrupt P1.0 external interrupt	IRQ4	P1CONL P1INT P1PND	E5H, bank 1 E6H, bank 1 E7H, bank 1
P1.7 external interrupt P1.6 external interrupt P1.5 external interrupt P1.4 external interrupt	IRQ5	P1CONH P1INT P1PND	E4H, bank 1 E6H, bank 1 E7H, bank 1
CE interrupt IRQ6		PLLMOD PLLREF PLLD1, PLLD0	F8H, bank 0 F9H, bank 0 F6H, F7H, bank 0
IF interrupt	IRQ7	IFMOD IFCNT1, IFCNT0 PLLMOD PLLREF	F3H, bank 0 F4H, F5H, bank 0 F8H, bank 0 F9H, bank 0

# Table 5-3. Interrupt Source Control and Data Registers



#### SYSTEM MODE REGISTER (SYM)

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

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

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

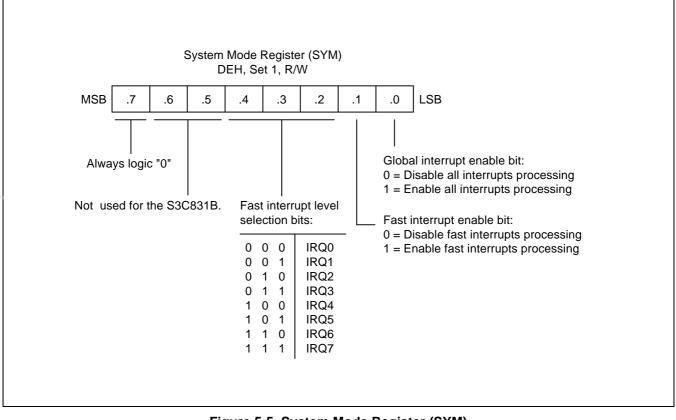


Figure 5-5. System Mode Register (SYM)



#### **INTERRUPT MASK REGISTER (IMR)**

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

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

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

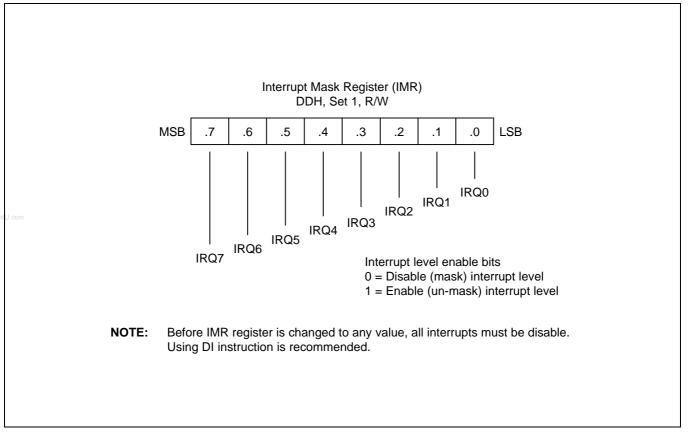


Figure 5-6. Interrupt Mask Register (IMR)



# **INTERRUPT PRIORITY REGISTER (IPR)**

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

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

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

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

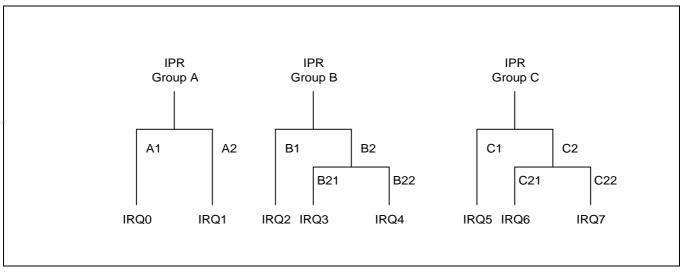


Figure 5-7. Interrupt Request Priority Groups

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

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

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



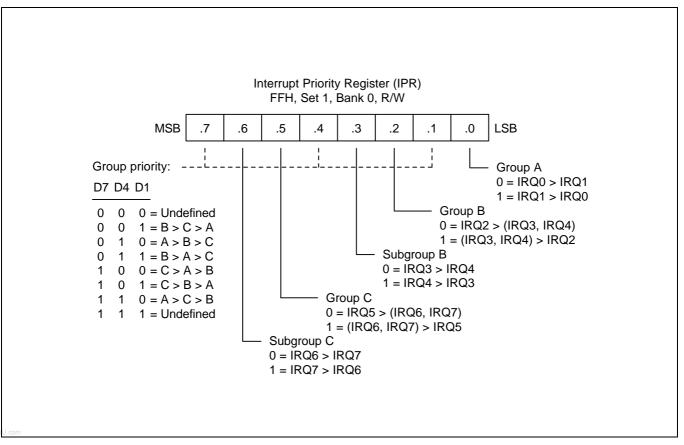


Figure 5-8. Interrupt Priority Register (IPR)



# **INTERRUPT REQUEST REGISTER (IRQ)**

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

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

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

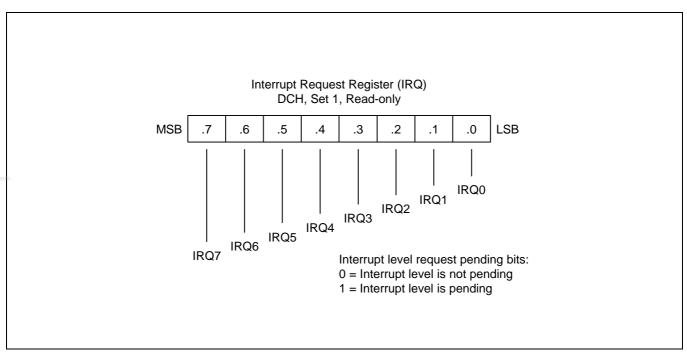


Figure 5-9. Interrupt Request Register (IRQ)



#### INTERRUPT PENDING FUNCTION TYPES

#### Overview

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

#### Pending Bits Cleared Automatically by Hardware

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

In the S3C831B interrupt structure, the timer 0 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 S3C831B 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.



NOTES



# 6 INSTRUCTION SET

### OVERVIEW

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

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

### DATA TYPES

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

### **REGISTER ADDRESSING**

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

### ADDRESSING MODES

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



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
	dst,src	Pop user stack (decrementing)
POPUI	dst,src	Pop user stack (incrementing)
PUSH	src	Push to stack
PUSHUD	dst,src	Push user stack (decrementing)
PUSHUI	dst,src	Push user stack (incrementing)

Table 6-1. Instruction Group Summary



Mnemonic	Operands	Instruction	
Arithmetic Instruct	ions		
ADC	dst,src	Add with carry	
ADD	dst,src	Add	
СР	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	
СОМ	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 In	structions	
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 Ins	structions	
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
ТСМ	dst,src	Test complement under mask
ТМ	dst,src	Test under mask

### Table 6-1. Instruction Group Summary (Continued)



Mnemonic	Operands	Instruction	
Rotate and Shift In	structions		
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 Instru	ctions		
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	

Table 6-1. Instruction Group Summary (Concluded)



### 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.

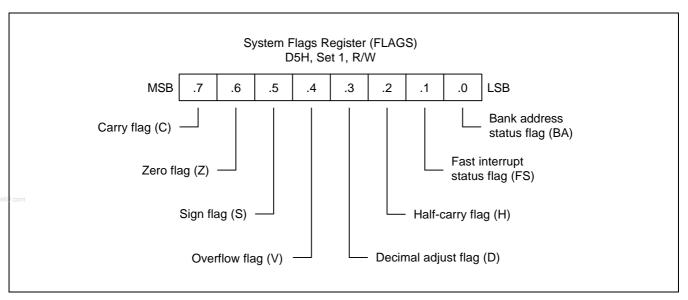


Figure 6-1. System Flags Register (FLAGS)



### 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.

### 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.

### 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

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

### Table 6-2. Flag Notation Conventions

### 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
Н	Hexadecimal number suffix
D	Decimal number suffix
В	Binary number suffix
орс	Opcode



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)
rO	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)
lr	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 $p = 0, 2,, 14$ )
Х	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 6-4. Instruction Notation Conventions



				OP	CODE MAP				
				LOWER	NIBBLE (H	IEX)			
	-	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
Р	1	RLC R1	RLC IR1	ADC r1,r2	ADC r1,Ir2	ADC R2,R1	ADC IR2,R1	ADC R1,IM	BCP r1.b, R2
Р	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>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
В	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	А	INCW RR1	INCW IR1	CP r1,r2	CP r1,Ir2	CP R2,R1	CP IR2,R1	CP R1,IM	LDC r1, Irr2, xL
E	В	CLR R1	CLR IR1	XOR r1,r2	XOR r1,Ir2	XOR R2,R1	XOR IR2,R1	XOR R1,IM	LDC r2, Irr2, xL
	С	RRC R1	RRC IR1	CPIJE Ir,r2,RA	LDC r1,Irr2	LDW RR2,RR1	LDW IR2,RR1	LDW RR1,IML	LD r1, lr2
н	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



				OP	CODE MAP				
				LOWER	NIBBLE (H	IEX)			
	-	8	9	А	В	С	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
Р	1	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	ENTER
Р	2								EXIT
Е	3								WFI
R	4								SB0
	5								SB1
N	6								IDLE
T	7	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	STOP
в	8								DI
в	9								EI
L	A								RET
E	В								IRET
	С								RCF
н	D	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	SCF
E	E								CCF
x	F	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NOP



### **CONDITION CODES**

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

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

Binary	Mnemonic	Description	Flags Set
0000	F	Always false	-
1000	Т	Always true	-
0111 <sup>(note)</sup>	С	Carry	C = 1
1111 <sup>(note)</sup>	NC	No carry	C = 0
0110 <sup>(note)</sup>	Z	Zero	Z = 1
1110 <sup>(note)</sup>	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 <sup>(note)</sup>	EQ	Equal	Z = 1
1110 <sup>(note)</sup>	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 <sup>(note)</sup>	UGE	Unsigned greater than or equal	C = 0
0111 <sup>(note)</sup>	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

Table 6-6. Condition Code
---------------------------

### NOTES:

 It indicates condition codes that are related to two different mnemonics but which test the same flag. For example, Z and EQ are both true if the zero flag (Z) is set, but after an ADD instruction, Z would probably be used; after a CP instruction, however, EQ would probably be used.

2. For operations involving unsigned numbers, the special condition codes UGE, ULT, UGT, and ULE must be used.



### INSTRUCTION DESCRIPTIONS

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

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



# **ADC** — Add with carry

ADC dst,src

**Operation:** dst  $\leftarrow$  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'scomplement 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 <u>dst</u>	Mode <u>src</u>
opc dst   src	2	4	12	r	r
		6	13	r	lr
opc src dst	3	6	14	R	R
		6	15	R	IR
opc dst src	3	6	16	R	IM

Examples:

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

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

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



### ADD - Add

ADD dst,src

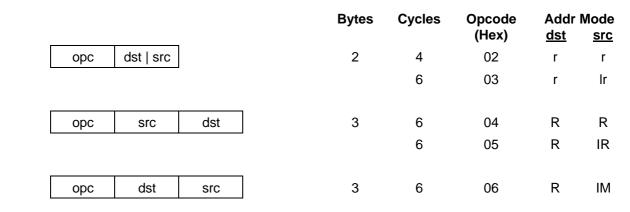
**Operation:** dst  $\leftarrow$  dst + src

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:



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

ADD	R1,R2	$\rightarrow$	R1 = 15H, R2 = 03H
ADD	R1,@R2	$\rightarrow$	R1 = 1CH, R2 = 03H
ADD	01H,02H	$\rightarrow$	Register 01H = 24H, register 02H = 03H
ADD	01H,@02H	$\rightarrow$	Register 01H = 2BH, register 02H = 03H
ADD	01H,#25H	$\rightarrow$	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  $\leftarrow$  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   <u>dst</u>	Mode <u>src</u>
орс	dst   src			2	4	52	r	r
					6	53	r	lr
			1					
орс	src	dst		3	6	54	R	R
					6	55	R	IR
орс	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	$\rightarrow$	R1 = 02H, R2 = 03H
AND	R1,@R2	$\rightarrow$	R1 = 02H, R2 = 03H
AND	01H,02H	$\rightarrow$	Register 01H = 01H, register 02H = 03H
AND	01H,@02H	$\rightarrow$	Register 01H = 00H, register 02H = 03H
AND	01H,#25H	$\rightarrow$	Register 01H = 21H

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



### **BAND** — Bit AND

- BAND dst,src.b
- BAND dst.b,src
- **Operation:**  $dst(0) \leftarrow dst(0)$  AND src(b)

or

 $dst(b) \leftarrow dst(b) AND src(0)$ 

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:

- Z: Set if the result is "0"; cleared otherwise.
- S: Cleared to "0".

C: Unaffected.

- V: Undefined.
- D: Unaffected.
- H: Unaffected.

#### Format:

			Bytes	Cycles	Opcode (Hex)	Addr <u>dst</u>	Mode <u>src</u>
орс	dst   b   0	src	3	6	67	rO	Rb
орс	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	$\rightarrow$	R1 = 06H, register 01H = 05H		
	BAND 01H.1,R1	$\rightarrow$	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	Addr	Mode
					(Hex)	<u>dst</u>	<u>src</u>
орс	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  $\rightarrow$  R1 = 07H, register 01H = 01H

If destination working register R1 contains the value 07H (00000111B) and the source register 01H contains the value 01H (0000001B), the statement "BCP R1,01H.1" compares bit one of the source register (01H) and bit zero of the destination register (R1). Because the bit values are not identical, the zero flag bit (Z) is cleared in the FLAGS register (0D5H).



# BITC — Bit Complement

BITC dst.b

**Operation:** dst(b)  $\leftarrow$  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>
орс	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  $\rightarrow$  R1 = 05H

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



### BITR — Bit Reset

BITR dst.b

**Operation:** dst(b)  $\leftarrow$  0

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

Flags: No flags are affected.

Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
орс	dst   b   0	2	4	77	rb

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

Example:	Given:	R1	=	07H:	
----------	--------	----	---	------	--

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

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



# BITS - Bit Set

BITS dst.b

**Operation:** dst(b)  $\leftarrow$  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>
орс	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  $\rightarrow$  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:**  $dst(0) \leftarrow dst(0)$  OR src(b)

or

 $dst(b) \leftarrow dst(b) OR src(0)$ 

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 <u>dst</u>	Mode <u>src</u>
орс	dst   b   0	src	3	6	07	r0	Rb
орс	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.

03H:

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

BOR	R1, 01H.1	$\rightarrow$	R1 = 07H, register 01H = 03H
BOR	01H.2, R1	$\rightarrow$	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.



Addr Mada

# 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	(Hex)	dst	src <u>src</u>
орс	src   b   0	dst	3	10	37	RA	rb

Dutes

Cycles

Oneede

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

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

BTJRF SKIP,R1.3

PC jumps to SKIP location

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

 $\rightarrow$ 



6-23

### BTJRT — Bit Test, Jump Relative on True

BTJRT dst,src.b

**Operation:** If src(b) is a "1", then PC  $\leftarrow$  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 <u>dst</u>	Mode <u>src</u>
орс	src   b   1	dst	3	10	37	RA	rb

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

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

BTJRT SKIP,R1.1

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



### **BXOR** — Bit XOR

BXOR dst,src.b

BXOR dst.b,src

**Operation:**  $dst(0) \leftarrow dst(0) \text{ XOR } src(b)$ 

or

 $dst(b) \leftarrow dst(b) XOR src(0)$ 

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 <u>dst</u>	Mode <u>src</u>
орс	dst   b   0	src	3	6	27	rO	Rb
орс	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	$\rightarrow$	R1 = 06H, register $01H = 03H$
BXOR	01H.2,R1	$\rightarrow$	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	$\leftarrow$	SP – 1
	@SP	$\leftarrow$	PCL
	SP	$\leftarrow$	SP –1
	@SP	$\leftarrow$	PCH
	PC	$\leftarrow$	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:

			E	Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
орс	d	st		3	14	F6	DA
орс	dst			2	12	F4	IRR
орс	dst			2	14	D4	IA

Examples:

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

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

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



# **CCF** — Complement Carry Flag

CCF

- $\mathsf{C}\ \leftarrow\ \mathsf{NOT}\ \mathsf{C}$ **Operation:** 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) 1 4 EF opc
- **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.



# $\mathbf{CLR} - \mathbf{Clear}$

CLR

 $@01H \rightarrow$ 

CLR	dst						
Operation:	dst $\leftarrow$ "0" The destin		on is cleared to "0".				
Flags:	No flags a	re affected.					
Format:							
				Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
	орс	dst		2	4	B0	R
					4	B1	IR
Examples:	Given: Re	egister 00H	= 4FH, register 01H =	02H, and	register 02l	H = 5EH:	
	CLR 00	$H \rightarrow$	Register 00H = 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.

Register 01H = 02H, register 02H = 00H



# COM - Complement

COM dst

Operation: dst 

NOT dst

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

#### Flags: C: Unaffected.

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

#### Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
орс	dst	2	4	60	R
			4	61	IR

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

 $COM R1 \rightarrow R1 = 0F8H$ 

COM @R1  $\rightarrow$  R1 = 07H, register 07H = 0EH

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

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



### **CP** – Compare

CP dst,src

#### **Operation:** dst - src

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

#### Flags:

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

#### Format:

				Bytes	Cycles	Opcode (Hex)	Addr <u>dst</u>	Mode <u>src</u>
	орс	dst   src		2	4	A2	r	r
_					6	A3	r	lr
	орс	src	dst	3	6	A4	R	R
					6	A5	R	IR
	орс	dst	src	3	6	A6	R	IM

#### **Examples:**

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

 $\label{eq:CP} \mathsf{CP} \qquad \mathsf{R1},\mathsf{R2} \ \rightarrow \qquad \mathsf{Set the } \mathsf{C} \ \mathsf{and} \ \mathsf{S} \ \mathsf{flags}$ 

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

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

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

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



. . . . . .

# **CPIJE** — Compare, Increment, and Jump on Equal

CPIJE dst,src,RA

**Operation:** If dst - src = "0", PC  $\leftarrow$  PC + RA

 $Ir \leftarrow Ir + 1$ 

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	Addr	Mode
						(Hex)	<u>dst</u>	<u>src</u>
орс	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  $\rightarrow$  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 (0000010B) 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.)



A . I . I ... N.M. ... I ..

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

**CPIJNE** dst,src,RA

**Operation:** If dst – src "0", PC  $\leftarrow$  PC + RA

 $lr \leftarrow lr + 1$ 

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	Addr	Mode
						(Hex)	<u>dst</u>	<u>src</u>
орс	src	dst	RA	3	12	D2	r	Ir

. . . .

**A** 

**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  $\rightarrow$  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 (0000010B). Because the result of the comparison is *non-equal*, the relative address is added to the PC and the PC then jumps to the memory location pointed to by SKIP. The source pointer register (R2) is also incremented by one, leaving a value of 04H. (Remember that the memory location must be within the allowed range of + 127 to -128.)



### **DA** — Decimal Adjust

DA

**Operation:** dst  $\leftarrow$  DA dst

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:

		Byt	es	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
орс	dst	2	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	;	$C \leftarrow$ "0", $H \leftarrow$ "0", Bits 4–7 = 3, bits 0–3 = C, R1 $\leftarrow$ 3CH
DA	R1	;	$R1 \leftarrow 3CH + 06$

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:

	0001	0101		15
+	0010	0111		27
	0011	1100	=	3CH

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

Assuming the same values given above, the statements

SUB 27H,R0;  $C \leftarrow "0", H \leftarrow "0", Bits 4-7 = 3, bits 0-3 = 1$ DA @R1 ; @R1  $\leftarrow 31-0$ 

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



# DEC - Decrement

DEC dst

**Operation:** dst  $\leftarrow$  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 <u>dst</u>
орс	dst	2	4	00	R
			4	01	IR

Examples:	Given:	R1 =	03H and	register 03H = 10H:
	DEC	R1	$\rightarrow$	R1 = 02H
	DEC	@R1	$\rightarrow$	Register 03H = 0FH

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



## **DECW** — Decrement Word

DECW dst

**Operation:** dst  $\leftarrow$  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 <u>dst</u>
орс	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



## **DI** — Disable Interrupts

DI

**Operation:** SYM (0)  $\leftarrow$  0

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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
орс	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					
<b>Operation:</b>	dst ÷ src					
	dst (UPPER) $\leftarrow$ REMAINDER					
	dst (LOWER) $\leftarrow$ 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:	<ul> <li>C: Set if the V flag is set and quotient is between 2<sup>8</sup> and 2<sup>9</sup> −1; cleared otherwise.</li> <li>Z: Set if divisor or quotient = "0"; cleared otherwise.</li> <li>S: Set if MSB of quotient = "1"; cleared otherwise.</li> <li>V: Set if quotient is ≥ 2<sup>8</sup> or if divisor = "0"; cleared otherwise.</li> <li>D: Unaffected.</li> <li>H: Unaffected.</li> </ul>					
Format:						
		Bytes	Cycles	Opcode (Hex)	Addr <u>dst</u>	Mode <u>src</u>
	opc src dst	3	26/10	94	RR	R
			26/10	95	RR	IR
			26/10	96	RR	IM
NOTE: Execut	ion takes 10 cycles if the divide-by-zero is attempte	ed; otherwise i	it takes 26 cy	cles.		
Examples:	Given: R0 = 10H, R1 = 03H, R2 = 40H,	register 40H	I = 80H:			
	DIV RR0,R2 $\rightarrow$ R0 = 03H,	R1 = 40H				

DIV	RR0,R2	$\rightarrow$	R0 = 03H, R1 = 40H
DIV	RR0,@R2	$\rightarrow$	R0 = 03H, R1 = 20H
DIV	RR0,#20H	$\rightarrow$	R0 = 03H, R1 = 80H

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



# DJNZ — Decrement and Jump if Non-Zero

DJNZ	r,dst				
Operation:	<ul> <li>r ← r - 1</li> <li>If r ≠ 0, PC ← PC + dst</li> <li>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.</li> <li>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.</li> </ul>				
Flags:	No flags are affected.				
Format:					
		Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
	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 rela	ative address:		
	SRP #0C0H				
	DJNZ R1,LOOP				
	DJNZ is typically used to control a "		ctions. In many cas		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.



<b>El</b> — Enab	ble Interrupts			
EI				
Operation:	SYM (0) $\leftarrow$ 1 An EI instruction sets bit zero of the system mode be serviced as they occur (assuming they have hi set while interrupt processing was disabled (by ex- when you execute the EI instruction.	ighest p	riority). If ar	n interrupt's pending bit was
Flags:	No flags are affected.			
Format:				
	E	Bytes	Cycles	Opcode (Hex)
	орс	1	4	9F
Example:	Given: SYM = 00H:			
	EI			
	If the SYM register contains the value 00H, that is			

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



## ENTER — Enter

### ENTER

SP	$\leftarrow$	SP – 2
@SP	$\leftarrow$	IP
IP	$\leftarrow$	PC
PC	$\leftarrow$	@IP
IP	$\leftarrow$	IP + 2
	@SP IP PC	@SP ← IP ← PC ←

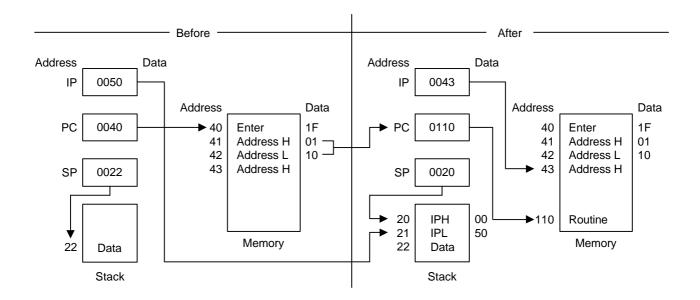
This instruction is useful when implementing threaded-code languages. The contents of the instruction pointer are pushed to the stack. The program counter (PC) value is then written to the instruction pointer. The program memory word that is pointed to by the instruction pointer is loaded into the PC,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

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
орс	1	14	1F

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





# EXIT-Exit

### EXIT

Operation:	IP	$\leftarrow$	@SP
	SP	$\leftarrow$	SP + 2
	PC	$\leftarrow$	@IP
	IP	$\leftarrow$	IP + 2

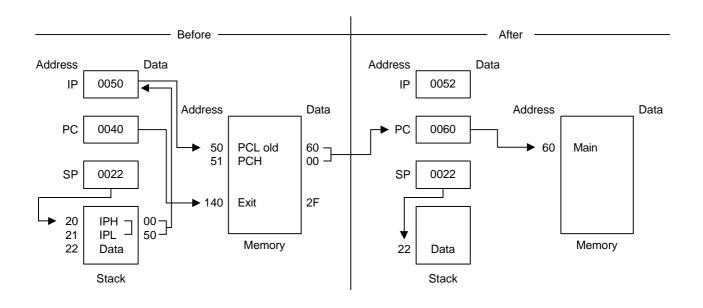
This instruction is useful when implementing threaded-code languages. The stack value is popped and loaded into the instruction pointer. The program memory word that is pointed to by the instruction pointer is then loaded into the program counter, and the instruction pointer is incremented by two.

Flags:	No flags are affected.
i lugo.	no nugo uro uncolcu.

### Format:

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

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





# IDLE — Idle Operation

### IDLE

### **Operation:**

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

Flags: No flags are affected.

#### Format:

	Bytes	Cycles	Opcode	Addr	Mode
			(Hex)	<u>dst</u>	<u>src</u>
орс	1	4	6F	-	-

**Example:** The instruction

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



## INC - Increment

INC

**Operation:** dst  $\leftarrow$  dst + 1

dst

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:

		E	Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
dst   opc			1	4	rE	r
					r = 0 to F	
орс	dst	]	2	4	20	R
				4	21	IR

### Examples:

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

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

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

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

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



## **INCW** — Increment Word

INCW dst

**Operation:** dst  $\leftarrow$  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:

		Byte	s Cycles	Gorean Opcode (Hex)	Addr Mode <u>dst</u>
орс	dst	2	8	A0	RR
			8	A1	IR

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

INCW RR0  $\rightarrow$  R0 = 1AH, R1 = 03H

INCW @R1  $\rightarrow$  Register 02H = 10H, register 03H = 00H

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

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

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



## IRET — Interrupt Return

 $\begin{array}{cccc} \textbf{IRET} & \underline{IRET (Normal)} & \underline{IRET (Fast)} \\ \textbf{Operation:} & FLAGS \leftarrow @SP & PC \leftrightarrow IP \\ & SP \leftarrow SP + 1 & FLAGS \leftarrow FLAGS' \\ & PC \leftarrow @SP & FIS \leftarrow 0 \\ & SP \leftarrow SP + 2 \\ & SYM(0) \leftarrow 1 \\ & This instruction is used at the end of an interval of an inte$ 

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

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

#### Format:

IRET (Normal)	Bytes	Cycles	Opcode (Hex)
орс	1	10 (internal stack)	BF
		12 (internal stack)	
IRET (Fast)	Bytes	Cycles	Opcode (Hex)
орс	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.

ОH	
FFH	IRET
100H	Interrupt Service Routine
	JP to FFH
FFFH	

**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).

F



## JP — Jump

- JP cc,dst (Conditional)
- JP dst (Unconditional)
- **Operation:** If cc is true, PC  $\leftarrow$  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: <sup>(1)</sup>

(2)		Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
cc   opc	dst	3	8	ccD	DA
				cc = 0 to F	
орс	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	$\rightarrow$	$LABEL_W = 1000H, PC = 1000H$
JP	@00H	$\rightarrow$	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 $\leftarrow$ PC + dst

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

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

Flags: No flags are affected.

### Format:

(1)		Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
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  $\rightarrow$  PC = 1FF7H

If the carry flag is set (that is, if the condition code is true), the statement "JR C,LABEL\_X" will pass control to the statement whose address is now in the PC. Otherwise, the program instruction following the JR would be executed.



## LD — Load

dst,src

**Operation:** dst  $\leftarrow$  src

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

Flags: No flags are affected.

### Format:

LD

			Bytes	Cycles	Opcode (Hex)	Addr <u>dst</u>	Mode <u>src</u>
dst   opc	src		2	4	rC	r	IM
				4	r8	r	R
src   opc	dst		2	4	r9	R	r
					r = 0 to F		
орс	dst   src		2	4	C7	r	lr
				4	D7	lr	r
орс	src	dst	3	6	E4	R	R
				6	E5	R	IR
орс	dst	src	3	6	E6	R	IM
				6	D6	IR	IM
орс	SrC	dst	3	6	F5	IR	R
орс	dst   src	x	3	6	87	r	x [r]
орс	src   dst	х	3	6	97	x [r]	r



# LD - Load

LD	(Contir	nued)						
Examples:		Given: $R0 = 01H$ , $R1 = 0AH$ , register $00H = 01H$ , register $01H = 20H$ , register $02H = 02H$ , LOOP = 30H, and register $3AH = 0FFH$ :						
	LD	R0,#10H	$\rightarrow$	R0 = 10H				
	LD	R0,01H	$\rightarrow$	R0 = 20H, register $01H = 20H$				
	LD	01H,R0	$\rightarrow$	Register 01H = 01H, R0 = 01H				
	LD	R1,@R0	$\rightarrow$	R1 = 20H, R0 = 01H				
	LD	@R0,R1	$\rightarrow$	R0 = 01H, R1 = 0AH, register 01H = 0AH				
	LD	00H,01H	$\rightarrow$	Register 00H = 20H, register 01H = 20H				
	LD	02H,@00H	$\rightarrow$	Register 02H = 20H, register 00H = 01H				
	LD	00H,#0AH	$\rightarrow$	Register $00H = 0AH$				
	LD	@00H,#10H	$\rightarrow$	Register 00H = 01H, register 01H = $10H$				
	LD	@00H,02H	$\rightarrow$	Register 00H = 01H, register 01H = 02, register 02H = 02H				
	LD	R0,#LOOP[R1	$] \rightarrow$	R0 = 0FFH, R1 = 0AH				
	LD	#LOOP[R0],R1	$\rightarrow$	Register 31H = 0AH, R0 = 01H, R1 = 0AH				



## LDB — Load Bit

LDB dst,src.b

LDB dst.b,src

**Operation:** dst(0)  $\leftarrow$  src(b)

or

 $dst(b) \leftarrow src(0)$ 

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	Addr	Mode
_						(Hex)	<u>dst</u>	src
	орс	dst   b   0	src	3	6	47	r0	Rb
ſ	орс	src   b   1	dst	3	6	47	Rb	rO

**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	$\rightarrow$	R0 = 07H, register $00H = 05H$
LDB	00H.0,R0	$\rightarrow$	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  $\leftarrow$  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 <u>dst</u>	Mode <u>src</u>
1.	орс	dst   src			2	10	C3	r	Irr
2.	орс	src   dst			2	10	D3	Irr	r
3.	орс	dst   src	XS	]	3	12	E7	r	XS [rr]
4.	орс	src   dst	XS	]	3	12	F7	XS [rr]	r
5.	орс	dst   src	XLL	XL <sub>H</sub>	4	14	A7	r	XL [rr]
6.	орс	src   dst	XLL	XL <sub>H</sub>	4	14	B7	XL [rr]	r
7.	орс	dst   0000	DAL	DA <sub>H</sub>	4	14	A7	r	DA
8.	орс	src   0000	DAL	DA <sub>H</sub>	4	14	B7	DA	r
9.	орс	dst   0001	DA <sub>L</sub>	DA <sub>H</sub>	4	14	A7	r	DA
10.	орс	src   0001	DAL	DA <sub>H</sub>	4	14	B7	DA	r

#### NOTES:

- 1. The source (src) or working register pair [rr] for formats 5 and 6 cannot use register pair 0–1.
- 2. For formats 3 and 4, the destination address 'XS [rr]' and the source address 'XS [rr]' are each one byte.
- 3. For formats 5 and 6, the destination address 'XL [rr] and the source address 'XL [rr]' are each two bytes.
- 4. The DA and r source values for formats 7 and 8 are used to address program memory; the second set of values, used in formats 9 and 10, are used to address data memory.



## LDC/LDE — Load Memory

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

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

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



# LDCD/LDED — Load Memory and Decrement

LDCD/LDED	dst,src								
Operation:	memory to the pair. The cont address is the	$rr \leftarrow rr - 1$ 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							
								mblei	
Flags:	No flags are a	affected.							
Format:									
	opc d	dst   src		Bytes 2	Cycles 10	Opcode (Hex) E2	Addr M <u>dst</u> r	<b>lode</b> <u>src</u> Irr	
rr ← rr − 1         These instructions are used for user stacks or block transfers of data from memory to the register file. The address of the memory location is specifie pair. The contents of the source location are loaded into the destination loc address is then decremented. The contents of the source are unaffected.         LDCD references program memory and LDED references external data m makes 'Irr' an even number for program memory and an odd number for d         Flags:       No flags are affected.         Format:       Bytes Cycles Option (H)	on 1033H =	0CDH, ar	۱d						
	LDCD R8	3,@RR6 ; ; ;	er file. The address of the memory locat the source location are loaded into the operation of the source are gram memory and LDED references ex- umber for program memory and an odd Bytes C 2 R7 = 33H, R8 = 12H, program memory location 1033H = 0DDH: ; 0CDH (contents of program me ; into R8 and RR6 is decremente ; R8 = 0CDH, R6 = 10H, R7 = ; 0DDH (contents of data memory ; into R8 and RR6 is decremente ; into R8 and RR6 is decremente	ented by on	e		d		
	LDED R8	3,@RR6 ; ; ;	into R8 and RR6	is decreme	ented by on	,			



# LDCI/LDEI — Load Memory and Increment

LDCI/LDEI	dst,src							
Operation:	rr ← rr These in: memory pair. The	dst $\leftarrow$ src rr $\leftarrow$ 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:								
	орс	dst   src		Bytes 2	Cycles 10	Opcode (Hex) E3	Addr Moc <u>dst sr</u> r Ir	<u>rc</u>
Examples:			= 33H, R8 = 12H, p al data memory locati ; 0CDH (contents ; into R8 and RR6 ; R8 = 0CDH, R	ons 1033H of program 5 is increme	= 0DDH a memory lo ented by one	nd 1034H =	0D5H: I) is loaded	
	LDEI	R8,@RR6	; 0DDH (contents ; into R8 and RR6 ; R8 = 0DDH, R	3 is increme	ented by one	,		



A . I . I ... N.M. ... I ..

## LDCPD/LDEPD — Load Memory with Pre-Decrement

#### LDCPD/ LDEPD

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

dst  $\leftarrow$  src

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	Addr	woae
				(Hex)	<u>dst</u>	<u>src</u>
орс	src   dst	2	14	F2	Irr	r

.....

**A** 

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

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



## LDCPI/LDEPI — Load Memory with Pre-Increment

#### LDCPI/ LDEPI

dst,src

**Operation:**  $rr \leftarrow rr + 1$ 

dst  $\leftarrow$  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 <u>dst</u>	
орс	src   dst	2	14	F3	Irr	r

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

LDCPI	@RR6,R0	; (RR6 $\leftarrow$ 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 $\leftarrow$ 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  $\leftarrow$  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 <u>dst</u>	Mode <u>src</u>
орс	src	dst		3	8	C4	RR	RR
					8	C5	RR	IR
орс	dst	S	C	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	$\rightarrow$	R6 = 06H, R7 = 1CH, R4 = 06H, R5 = 1CH
LDW	00H,02H	$\rightarrow$	Register 00H = 03H, register 01H = 0FH, register 02H = 03H, register 03H = 0FH
LDW	RR2,@R7	$\rightarrow$	R2 = 03H, R3 = 0FH,
LDW	04H,@01H	$\rightarrow$	Register 04H = 03H, register 05H = 0FH
LDW	RR6,#1234H	$\rightarrow$	R6 = 12H, R7 = 34H
LDW	02H,#0FEDH	$\rightarrow$	Register 02H = 0FH, register 03H = 0EDH

In the second example, please note that the statement "LDW 00H,02H" loads the contents of the source word 02H, 03H into the destination word 00H, 01H. This leaves the value 03H in general register 00H and the value 0FH in register 01H.

The other examples show how to use the LDW instruction with various addressing modes and formats.



# MULT — Multiply (Unsigned)

MULT dst,src

**Operation:** dst  $\leftarrow$  dst  $\times$  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 <u>dst</u>	Mode <u>src</u>
орс	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	$\rightarrow$	Register 00H = 01H, register 01H = 20H, register 02H = 09H
MULT	00H, @01H	$\rightarrow$	Register 00H = 00H, register 01H = $0C0H$
MULT	00H, #30H	$\rightarrow$	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:**  $PC \leftarrow @ IP$ 

 $IP \leftarrow IP + 2$ 

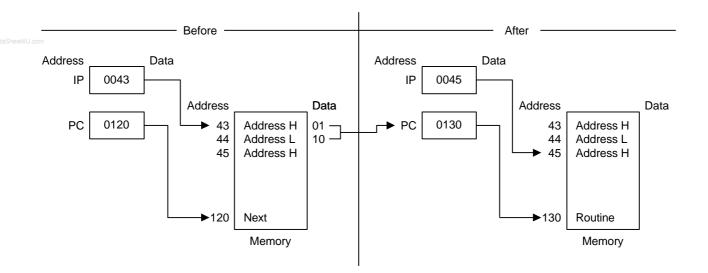
The NEXT instruction is useful when implementing threaded-code languages. The program memory word that is pointed to by the instruction pointer is loaded into the program counter. The instruction pointer is then incremented by two.

Flags: No flags are affected.

#### Format:

	Byte	s Cycles	s Opcode (Hex)
орс	1	10	0F

**Example:** The following diagram shows one example of how to use the NEXT instruction.





# $\mathbf{NOP}-\mathbf{No}$ 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)
орс	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.



## $\mathbf{OR}$ — Logical OR

OR dst,src

**Operation:** dst  $\leftarrow$  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:

			Ву	tes	Cycles	Opcode (Hex)	Addr <u>dst</u>	Mode <u>src</u>
орс	dst   src		2	2	4	42	r	r
					6	43	r	lr
орс	src	dst	3	3	6	44	R	R
					6	45	R	IR
орс	dst	src	3	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 $\rightarrow$	R0 = 3FH, R1 = 2AH
OR	$R0,@R2 \rightarrow$	R0 = 37H, R2 = 01H, register 01H = 37H
OR	00H,01H $\rightarrow$	Register 00H = 3FH, register 01H = 37H
OR	01H,@00H $\rightarrow$	Register 00H = 08H, register 01H = 0BFH
OR	00H,#02H $\rightarrow$	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.



# $\mathbf{POP} - \mathbf{Pop}$ From Stack

POP	dst							
Operation:		P + 1		dressed by the ed by one.	stack point	er are load	ed into the de	estination. The
Flags:	No flags a	affected.						
Format:								
					Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
	орс	dst			2	8	50	R
						8	51	IR
Examples:		egister 00H register 0FB		•	1BH, SPH	(0D8H) =	00H, SPL (0	D9H) = 0FBH,
	POP	00H	$\rightarrow$	Register 00H	= 55H, SF	P = 00FCH	1	
	POP	@00H	$\rightarrow$	Register 00H	= 01H, re	gister 01H	= 55H, SP =	= 00FCH
				gister 00H conta 0FBH (55H) int				"POP 00H" increments the

loads the contents of location 00FBH (55H) into destination register 00H and then increments 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:		tion is use dressed by	the user s	defined stacks tack pointer ar	•			•	
Flags:	No flags are affected.								
Format:									
					Bytes	Cycles	Opcode (Hex)	Addr <u>dst</u>	Mode <u>src</u>
	орс	SIC	dst	]	3	8	92	R	IR
Example:	Given: Reg register 02H		= 42H (us	ser stack pointe	er register	), register 4	2H = 6FH, a	and	

POPUD 02H,@00H  $\rightarrow$  Register 00H = 41H, register 02H = 6FH, register 42H = 6FH

If general register 00H contains the value 42H and register 42H the value 6FH, the statement "POPUD 02H,@00H" loads the contents of register 42H into the destination register 02H. The user stack pointer is then decremented by one, leaving the value 41H.



# **POPUI** — Pop User Stack (Incrementing)

POPUI	dst,src					
Operation:	dst $\leftarrow$ src IR $\leftarrow$ IR + 1 The POPUI instruction is used for user-defined register file location addressed by the user state stack pointer is then incremented.		-			
Flags:	No flags are affected.					
Format:						
		Bytes	Cycles	Opcode (Hex)	Addr M <u>dst</u>	/lode <u>src</u>
	opc src dst	3	8	93	R	IR
Example:	Given: Register 00H = 01H and register 01H	l = 70H:				
	POPUI 02H,@00H $\rightarrow$ Register 00H	= 02H, re	gister 01H	= 70H, regist	ter 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  $\leftarrow$  SP - 1

 $@SP \leftarrow src$ 

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>
орс	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 = 0AA	H, SPH = 00H, and SPL = 00H:
-----------	---------------------------	----------------------	------------------------------

PUSH	40H	$\rightarrow$	Register 40H = 4FH, stack register 0FFH = 4FH, SPH = 0FFH, SPL = 0FFH
PUSH	@40H	$\rightarrow$	Register 40H = 4FH, register 4FH = 0AAH, stack register 0FFH = 0AAH, SPH = 0FFH, SPL = 0FFH

In the first example, if the stack pointer contains the value 0000H, and general register 40H the value 4FH, the statement "PUSH 40H" decrements the stack pointer from 0000 to 0FFFFH. It then loads the contents of register 40H into location 0FFFFH and adds this new value to the top of the stack.



## **PUSHUD** — Push User Stack (Decrementing)

PUSHUD	dst,src					
Operation:	$IR \leftarrow IR - 1$					
	dst $\leftarrow$ 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 I <u>dst</u>	Node <u>src</u>
	opc dst src	3	8	82	IR	R
Example:	Given: Register 00H = 03H, register 01H =	05H, and	register 02F	H = 1AH:		
	PUSHUD @00H,01H $\rightarrow$ Register 00H	= 02H, re	gister 01H	= 05H, regis	ter 02H	= 05H
	If the upper steely pointer (register 00H) for ever	nla) aante	sing the valu	10 02LL the e	totomont	

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:	$IR \leftarrow IR + 1$					
	dst $\leftarrow$ src					
	This instruction is used for user-defined stacks stack pointer and then loads the contents of the the incremented user stack pointer.	0				
Flags:	No flags are affected.					
Format:						
		Bytes	Cycles	Opcode (Hex)	Addr <u>dst</u>	Mode <u>src</u>
	opc dst src	3	8	83	IR	R
Example:	Given: Register 00H = 03H, register 01H =	05H, and	register 04F	H = 2AH:		
	PUSHUI @00H,01H $\rightarrow$ Register 00H	= 04H, re	gister 01H	= 05H, regis	ster 04H	= 05H
	If the user stack pointer (register 00H, for exam	nple) conta	ains the valu	ue 03H, the s	tatement	[

"PUSHUI @00H,01H" increments the user stack pointer by one, leaving the value 04H. The 01H register value, 05H, is then loaded into the location addressed by the incremented user stack pointer.



# RCF — Reset Carry Flag

RCF	RCF					
Operation:	$C \leftarrow 0$ The ca	) rry flag is cleared to logic ze	ero, regardles	s of its pr	evious valu	е.
Flags:	C:	Cleared to "0".				
	No oth	er flags are affected.				
Format:						
				Bytes	Cycles	Opcode (Hex)
	ор			1	4	CF
Example:	Given:	C = "1" or "0":				

The instruction RCF clears the carry flag (C) to logic zero.



# **RET** — Return

### RET

**Operation:**  $PC \leftarrow @SP$ 

 $SP \leftarrow SP + 2$ 

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)
op	C	1 8 (internal stack)	AF
		10 (internal stack)	

### **Example:** Given: SP = 00FCH, (SP) = 101AH, and PC = 1234:

RET  $\rightarrow$  PC = 101AH, SP = 00FEH

The statement "RET" pops the contents of stack pointer location 00FCH (10H) into the high byte of the program counter. The stack pointer then pops the value in location 00FEH (1AH) into the PC's low byte and the instruction at location 101AH is executed. The stack pointer now points to memory location 00FEH.

## RL — Rotate Left

RL

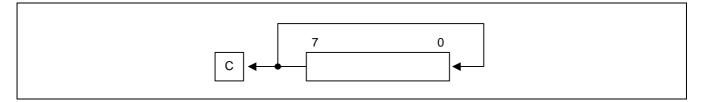
dst

**Operation:**  $C \leftarrow dst(7)$ 

dst (0)  $\leftarrow$  dst (7)

dst (n + 1)  $\leftarrow$  dst (n), n = 0-6

The contents of the destination operand are rotated left one bit position. The initial value of bit 7 is moved to the bit zero (LSB) position and also replaces the carry flag.



#### Flags:

- C: Set if the bit rotated from the most significant bit position (bit 7) was "1".
- Z: Set if the result is "0"; cleared otherwise.
  - **S:** Set if the result bit 7 is set; cleared otherwise.
  - V: Set if arithmetic overflow occurred; cleared otherwise.
  - D: Unaffected.
  - H: Unaffected.

### Format:

		Bytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
орс	dst	2	4	90	R
			4	91	IR

Examples:	Given:	Register 00H =	0AAH, register 01H =	02H and register 02H = $17H$ :
-----------	--------	----------------	----------------------	--------------------------------

RL00H $\rightarrow$ Register 00H = 55H, C = "1"RL@01H $\rightarrow$ 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

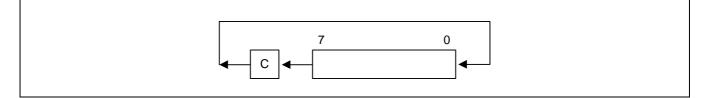
RLC dst

**Operation:** dst (0)  $\leftarrow$  C

 $C \leftarrow dst(7)$ 

dst (n + 1)  $\leftarrow$  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 <u>dst</u>
	орс	dst	2	4	10	R
_				4	11	IR

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

RLC 00H  $\rightarrow$  Register 00H = 54H, C = "1"

RLC @01H  $\rightarrow$  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.



# $\mathbf{RR}$ — Rotate Right

dst

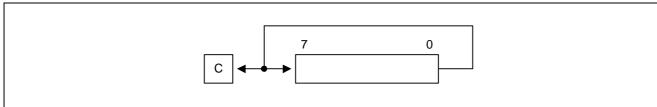
RR

**Operation:**  $C \leftarrow dst(0)$ 

dst (7) ← dst (0)

dst (n)  $\leftarrow$  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 <u>dst</u>
орс	dst	2	4	E0	R
			4	E1	IR

**Examples:** Given: Register 00H = 31H, register 01H = 02H, and register 02H = 17H:

RR 00H  $\rightarrow$  Register 00H = 98H, C = "1"

RR @01H  $\rightarrow$  Register 01H = 02H, register 02H = 8BH, C = "1"

In the first example, if general register 00H contains the value 31H (00110001B), the statement "RR 00H" rotates this value one bit position to the right. The initial value of bit zero is moved to bit 7, leaving the new value 98H (10011000B) in the destination register. The initial bit zero also resets the C flag to "1" and the sign flag and overflow flag are also set to "1".



# **RRC** — Rotate Right Through Carry

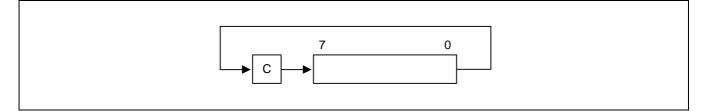
RRC dst

**Operation:** dst (7)  $\leftarrow$  C

 $C \leftarrow dst(0)$ 

dst (n)  $\leftarrow$  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 <u>dst</u>
	орс	dst	2	4	C0	R
-				4	C1	IR

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

RRC	00H	$\rightarrow$	Register 00H = 2AH, C = "1"
	-		

```
\label{eq:RRC} \ensuremath{\mathbb{R}}\ensuremath{\mathbb{O}}\ensuremath{\mathbb{O}}\ensuremath{\mathbb{H}}\ensuremath{\mathbb{O}}\ensuremath{\mathbb{O}}\ensuremath{\mathbb{H}}\ensuremath{\mathbb{H}}\ensuremath{\mathbb{O}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{H}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensuremath{\mathbb{C}}\ensur
```

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 fla selecting bank 0 register addressing in the set 7						
Flags:	No flags are affected.						
Format:							
		Bytes	Cycles	Opcode (Hex)			
	орс	1	4	4F			
Example:	The statement						
	SB0						
	clears FLAGS.0 to "0", selecting bank 0 register addressing.						



# SB1 — Select Bank 1

## SB1 **Operation:** $\mathsf{BANK} \leftarrow 1$ 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) 5F 1 4 opc 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  $\leftarrow$  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 <u>dst</u>	Mode <u>src</u>
орс	dst   src			2	4	32	r	r
					6	33	r	lr
<b></b>			l					
орс	src	dst		3	6	34	R	R
					6	35	R	IR
[			I					
орс	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	$\rightarrow$	R1 = 0CH, R2 = 03H
SBC	R1,@R2	$\rightarrow$	R1 = 05H, R2 = 03H, register 03H = 0AH
SBC	01H,02H	$\rightarrow$	Register 01H = 1CH, register 02H = $03H$
SBC	01H,@02H	$\rightarrow$	Register 01H = 15H, register 02H = 03H, register 03H = 0AH
SBC	01H,#8AH	$\rightarrow$	Register 01H = 95H; C, S, and V = "1"

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



# $\mathbf{SCF}-\mathbf{Set}$ Carry Flag

SCF

Operation:	$C \leftarrow 1$ The carry flag (C) is set to logic one, regardless	s of its pre	vious value.	
Flags:	<b>C:</b> Set to "1".			
	No other flags are affected.			
Format:				
		Bytes	Cycles	Opcode (Hex)
	орс	1	4	DF
Example:	The statement			
	SCF			
	sets the carry flag to logic one.			



# SRA — Shift Right Arithmetic

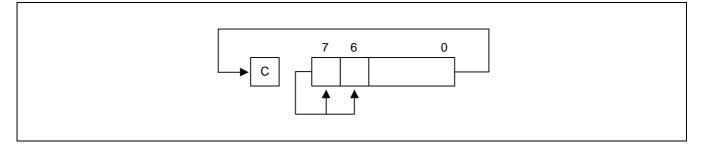
SRA dst

**Operation:** dst (7)  $\leftarrow$  dst (7)

 $C \leftarrow dst(0)$ 

dst (n)  $\leftarrow$  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 <u>dst</u>
орс	dst	2	4	D0	R
			4	D1	IR

Examples:	Given: Register 00H	= 9AH, register 02H =	03H, register 03H =	0BCH, and $C = "1"$ :
-----------	---------------------	-----------------------	---------------------	-----------------------

SRA	00H	$\rightarrow$	Register 00H = 0CD, C = "0"
SRA	@02H	$\rightarrow$	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)	$\leftarrow$	src (3–7)
	If src (1) = 0 and src (0) = 1 then:	RP1 (3–7)	$\leftarrow$	src (3–7)
	If src (1) = 0 and src (0) = 0 then:	RP0 (4–7)	$\leftarrow$	src (4–7),
		RP0 (3)	$\leftarrow$	0
		RP1 (4–7)	$\leftarrow$	src (4–7),
		RP1 (3)	$\leftarrow$	1
	The source data bits one and zero (L	.SB) determin	e whe	ether to write one

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 Addr Mode (Hex) src 2 4 31 IM

**Examples:** The statement

SRP #40H

sets register pointer 0 (RP0) at location 0D6H to 40H and register pointer 1 (RP1) at location 0D7H to 48H.

The statement "SRP0 #50H" sets RP0 to 50H, and the statement "SRP1 #68H" sets RP1 to 68H.



# **STOP** — Stop Operation

### STOP

## **Operation:**

The STOP instruction stops the both the CPU clock and system clock and causes the microcontroller to enter Stop mode. During Stop mode, the contents of on-chip CPU registers, peripheral registers, and I/O port control and data registers are retained. Stop mode can be released by an external reset operation or by external interrupts. For the reset operation, the RESET pin must be held to Low level until the required oscillation stabilization interval has elapsed.

In application programs, a STOP instruction must be immediately followed by at least three NOP instructions. This ensures an adeguate time interval for the clock to stabilize before the next instruction is executed. If three or more NOP instructons are not used after STOP instruction, leakage current could be flown because of the floating state in the internal bus.

**Flags:** No flags are affected.

Format:

	Bytes	Cycles	Opcode	Addr Mode		
			(Hex)	<u>dst</u>	<u>src</u>	
орс	1	4	7F	-	_	

### **Example:** The statement

STOP NOP NOP NOP ; halts all microcontroller operations



. . . . . .

# SUB - Subtract

SUB dst,src

**Operation:**  $dst \leftarrow dst - src$ 

> 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 <u>dst</u>	Mode <u>src</u>
орс	dst   src		2	4	22	r	r
				6	23	r	lr
орс	src	dst	3	6	24	R	R
				6	25	R	IR
орс	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	$\rightarrow$	R1 = 0FH, R2 = 03H
SUB	R1,@R2	$\rightarrow$	R1 = 08H, R2 = 03H
SUB	01H,02H	$\rightarrow$	Register 01H = 1EH, register 02H = 03H
SUB	01H,@02H	$\rightarrow$	Register 01H = 17H, register 02H = 03H
SUB	01H,#90H	$\rightarrow$	Register 01H = 91H; C, S, and V = "1"
SUB	01H,#65H	$\rightarrow$	Register 01H = 0BCH; C and S = "1", V = "0"

In the first example, if working register R1 contains the value 12H and if register R2 contains the value 03H, the statement "SUB R1,R2" subtracts the source value (03H) from the destination value (12H) and stores the result (0FH) in destination register R1.



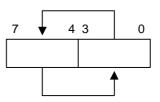
. . . . . .

# **SWAP** — Swap Nibbles

SWAP dst

**Operation:** dst  $(0 - 3) \leftrightarrow dst (4 - 7)$ 

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:

		В	ytes	Cycles	Opcode (Hex)	Addr Mode <u>dst</u>
орс	dst		2	4	F0	R
				4	F1	IR

**Examples:** Given: Register 00H = 3EH, register 02H = 03H, and register 03H = 0A4H:

SWAP	00H	$\rightarrow$	Register 00H = 0E3H
SWAP	@02H	$\rightarrow$	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 <u>dst</u>	Mode <u>src</u>
c	рс	dst   src		2	4	62	r	r
					6	63	r	lr
C	рс	SrC	dst	3	6	64	R	R
					6	65	R	IR
C	рс	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:

ТСМ	R0,R1	$\rightarrow$	R0 = 0C7H, R1 = 02H, Z = "1"
ТСМ	R0,@R1	$\rightarrow$	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = "0"
ТСМ	00H,01H	$\rightarrow$	Register 00H = 2BH, register 01H = 02H, Z = "1"
ТСМ	00H,@01H	$\rightarrow$	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = "1"
ТСМ	00H,#34	$\rightarrow$	Register 00H = 2BH, Z = "0"

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (0000010B), the statement "TCM R0,R1" tests bit one in the destination register for a "1" value. Because the mask value corresponds to the test bit, the Z flag is set to logic one and can be tested to determine the result of the TCM operation.



# **TM** — Test Under Mask

TM dst,src

Operation: dst AND src

This instruction tests selected bits in the destination operand for a logic zero value. The bits to be tested are specified by setting a "1" bit in the corresponding position of the source operand (mask), which is ANDed with the destination operand. The zero (Z) flag can then be checked to determine the result. The destination and source operands are unaffected.

#### Flags:

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

#### Format:

				Bytes	Cycles	Opcode (Hex)	Addr <u>dst</u>	Mode <u>src</u>
орс	dst   src			2	4	72	r	r
					6	73	r	lr
орс	src	dst		3	6	74	R	R
					6	75	R	IR
орс	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:

ТМ	R0,R1	$\rightarrow$	R0 = 0C7H, R1 = 02H, Z = "0"
ТМ	R0,@R1	$\rightarrow$	R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = "0"
ТМ	00H,01H	$\rightarrow$	Register 00H = 2BH, register 01H = 02H, Z = "0"
ТМ	00H,@01H	$\rightarrow$	Register 00H = 2BH, register 01H = 02H, register 02H = 23H, Z = "0"
ТМ	00H,#54H	$\rightarrow$	Register 00H = 2BH, Z = "1"

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement "TM R0,R1" tests bit one in the destination register for a "0" value. Because the mask value does not match the test bit, the Z flag is cleared to logic zero and can be tested to determine the result of the TM operation.



# WFI — Wait for Interrupt

## WFI

## Operation:

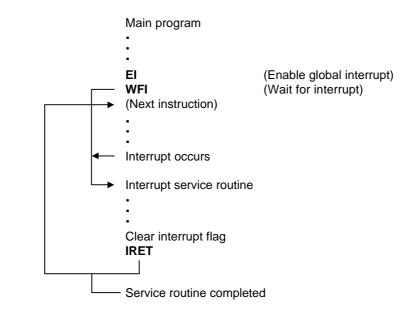
The CPU is effectively halted until an interrupt occurs, except that DMA transfers can still take place during this wait state. The WFI status can be released by an internal interrupt, including a fast interrupt.

Flags: No flags are affected.

## Format:

	Bytes	Cycles	Opcode (Hex)
орс	1	4n	3F
		(n = 1, 2, 3	3, )

**Example:** The following sample program structure shows the sequence of operations that follow a "WFI" statement:





# XOR — Logical Exclusive OR

XOR dst,src

**Operation:** dst 

dst 

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.

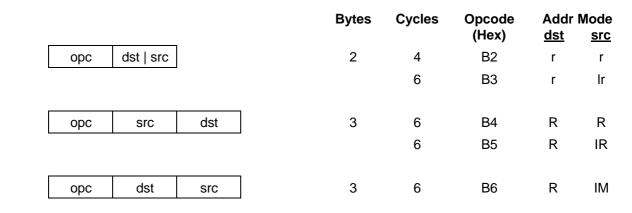
#### Flags:

- 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.

C: Unaffected.

H: Unaffected.

#### Format:



**Examples:** Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

XOR	R0,R1	$\rightarrow$	R0 = 0C5H, R1 = 02H
XOR	R0,@R1	$\rightarrow$	R0 = 0E4H, R1 = 02H, register 02H = 23H
XOR	00H,01H	$\rightarrow$	Register 00H = 29H, register 01H = 02H
XOR	00H,@01H	$\rightarrow$	Register 00H = 08H, register 01H = 02H, register 02H = 23H
XOR	00H,#54H	$\rightarrow$	Register 00H = 7FH

In the first example, if working register R0 contains the value 0C7H and if register R1 contains the value 02H, the statement "XOR R0,R1" logically exclusive-ORs the R1 value with the R0 value and stores the result (0C5H) in the destination register R0.



NOTES



# CLOCK CIRCUIT

# OVERVIEW

The clock frequency generated for the S3C831B by an external crystal can range from 0.4 MHz to 9.0 MHz. The maximum CPU clock frequency is 9.0 MHz. The  $X_{IN}$  and  $X_{OUT}$  pins connect the external oscillator or clock source to the on-chip clock circuit.

# SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

- External crystal or ceramic resonator oscillation source (or an external clock source)
- Oscillator stop and wake-up functions
- Programmable frequency divider for the CPU clock (fxx divided by 1, 2, 8, or 16)
- System clock control register, CLKCON
- STOP control register, STPCON

## **CPU Clock Notation**

In this document, the following notation is used for descriptions of the CPU clock;

fx: main clock fxt: sub clock (the fxt is not implemented in the S3C831B) fxx: selected system clock

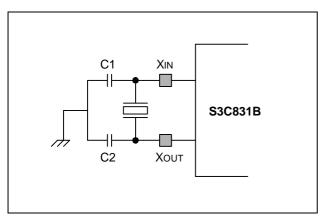


Figure 7-1. Main Oscillator Circuit (Crystal or Ceramic Oscillator)

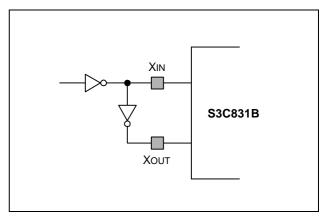


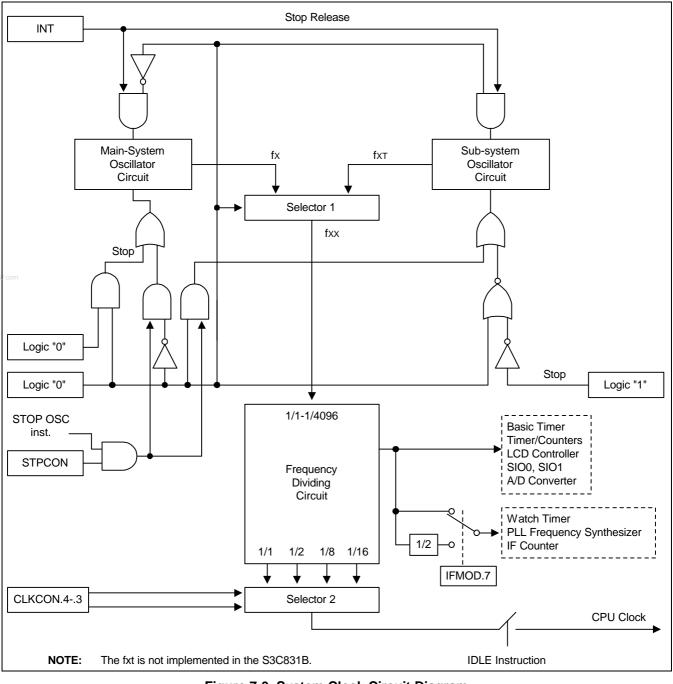
Figure 7-2. Main Oscillator Circuit (External Oscillator)

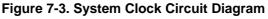


## **CLOCK STATUS DURING POWER-DOWN MODES**

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

- In Stop mode, the main oscillator is halted. Stop mode is released, and the oscillator is started, by a reset
  operation or an external interrupt (with RC delay noise filter).
- In Idle mode, the internal clock signal is gated to the CPU, but not to interrupt structure, timers, timer/ counters, and watch timer. Idle mode is released by a reset or by an external or internal interrupt.







## SYSTEM CLOCK CONTROL REGISTER (CLKCON)

The system clock control register, CLKCON, is located in the set 1, address D4H. It is read/write addressable and has the following functions:

- Oscillator frequency divide-by value

After the main oscillator is activated, and the fxx/16 (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed fxx/8, fxx/2, or fxx/1.

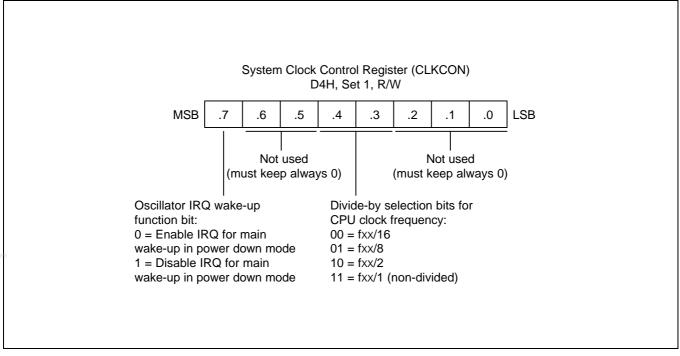


Figure 7-4. System Clock Control Register (CLKCON)



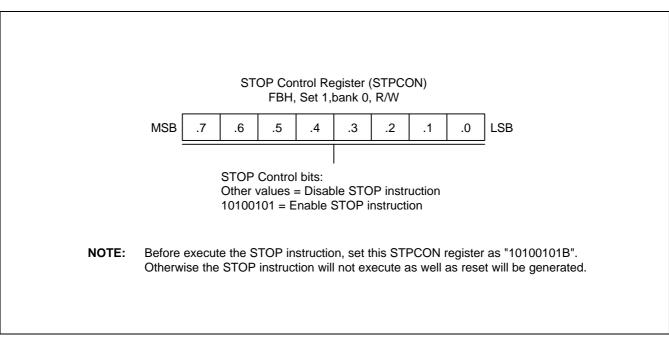


Figure 7-5. STOP Control Register (STPCON)



# 8 RESET and POWER-DOWN

# SYSTEM RESET

# OVERVIEW

During a power-on reset, the voltage at  $V_{DD}$  goes to High level and the 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 S3C831B 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<sub>DD</sub> and RESET are High level), the

RESET pin is forced Low level and the reset operation starts. All system and peripheral control registers are then reset to their default hardware values

In summary, the following sequence of events occurs during a reset operation:

- All interrupt is disabled.
- The watchdog function (basic timer) is enabled.
- Ports 0-3 are set to input mode, and all pull-up resistors are disabled for the I/O port.
- Peripheral control and data register settings are disabled and reset to their default hardware values.
- The program counter (PC) is loaded with the program reset address in the ROM, 0100H.
- When the programmed oscillation stabilization time interval has elapsed, the instruction stored in ROM location 0100H (and 0101H) is fetched and executed.

## NORMAL MODE RESET OPERATION

In normal (masked ROM) mode, the Test pin is tied to V<sub>SS</sub>. A reset enables access to the 64-Kbyte on-chip ROM.

# NOTE

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



# HARDWARE RESET VALUES

Table 8-1, 8-2, 8-3 list the reset values for CPU and system registers, peripheral control registers, and peripheral data registers following a reset operation. The following notation is used to represent reset values:

- A "1" or a "0" shows the reset bit value as logic one or logic zero, respectively.
- An "x" means that the bit value is undefined after a reset.
- A dash ("-") means that the bit is either not used or not mapped, but read 0 is the bit value.

Register Name	Mnemonic	Add	ress	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
Clock Control Register	CLKCON	212	D4H	0	-	-	0	0	I	-	-
System Flags Register	FLAGS	213	D5H	х	х	х	х	х	х	0	0
Register Pointer (High Byte)	RP0	214	D6H	1	1	0	0	0	-	-	_
Register Pointer (Low Byte)	RP1	215	D7H	1	1	0	0	1	-	-	-
Stack Pointer (High Byte)	SPH	216	D8H	х	х	х	х	х	х	х	х
Stack Pointer (Low Byte)	SPL	217	D9H	х	х	х	х	х	х	х	х
Instruction Pointer (High Byte)	IPH	218	DAH	х	х	х	х	х	х	х	х
Instruction Pointer (Low Byte)	IPL	219	DBH	х	х	х	х	х	х	х	х
Interrupt Request Register	IRQ	220	DCH	0	0	0	0	0	0	0	0
Interrupt Mask Register	IMR	221	DDH	х	х	х	х	х	х	х	х
System Mode Register	SYM	222	DEH	0	-	_	х	х	х	0	0
Register Page Pointer	PP	223	DFH	0	0	0	0	0	0	0	0

# Table 8-1. S3C831B Set 1 Register and Values after RESET



Register Name	Mnemonic	Add	ress		Bit Values after RESET						
		Dec	Hex	7	6	5	4	3	2	1	0
Timer 0 Counter Register	TOCNT	224	E0H	0	0	0	0	0	0	0	0
Timer 0 Data Register	TODATA	225	E1H	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	226	E2H	0	0	0	0	0	0	0	0
Timer 1 Counter Register	T1CNT	227	E3H	0	0	0	0	0	0	0	0
Timer 1 Data Register	T1DATA	228	E4H	1	1	1	1	1	1	1	1
Timer 1 Control Register	T1CON	229	E5H	0	0	0	0	0	0	0	0
Interrupt Pending Register	INTPND	230	E6H	_	-	_	_	_	_	0	0
	Location E7	'H is not	mapped	d.	•	•	•	•	•	•	
Watch Timer Control Register	WTCON	232	E8H	-	0	0	0	0	0	0	0
SIO 0 Control Register	SIO0CON	233	E9H	0	0	0	0	0	0	0	0
SIO 0 Data Register	SIO0DATA	234	EAH	0	0	0	0	0	0	0	0
SIO 0 Prescaler Register	SIO0PS	235	EBH	0	0	0	0	0	0	0	0
SIO 1 Control Register	SIO1CON	236	ECH	0	0	0	0	0	0	0	0
SIO 1 Data Register	SIO1DATA	237	EDH	0	0	0	0	0	0	0	0
SIO 1 Prescaler Register	SIO1PS	238	EEH	0	0	0	0	0	0	0	0
A/D Converter Control Register	ADCON	239	EFH	-	0	0	0	0	0	0	0
A/D Converter Data Register	ADDATA	240	F0H	х	х	х	х	х	х	x	х
LCD Control Register	LCON	241	F1H	0	-	—	—	0	0	0	0
LCD Mode Register	LMOD	242	F2H	0	0	0	0	0	0	0	0
IF Counter Mode Register	IFMOD	243	F3H	0	-	-	0	0	0	0	0
IF Counter 1	IFCNT1	244	F4H	0	0	0	0	0	0	0	0
IF Counter 0	IFCNT0	245	F5H	0	0	0	0	0	0	0	0
PLL Data Register 1	PLLD1	246	F6H	х	х	х	х	х	х	x	х
PLL Data Register 0	PLLD0	247	F7H	х	х	х	х	х	х	х	х
PLL Mode Register	PLLMOD	248	F8H				(no	ote)			
PLL Reference Frequency Register	PLLREF	249	F9H				(no	ote)			
	Location FA	H is not	mapped	d.		-	-	-	-		
STOP Control Register	STPCON	251	FBH	0	0	0	0	0	0	0	0
	Location FC	CH is not	mapped	d.							
Basic Timer Data Register	BTCNT	253	FDH	0	0	0	0	0	0	0	0
	Location FE	H is not	mapped	d.		-	-	-	-		
Interrupt Priority Register	IPR	255	FFH	х	х	х	х	х	х	х	х

Table 8-2, S3C831B Set 1,	Bank 0 Register Values after RESET

**NOTE:** Refer to the corresponding register in the chapter 4.



Table 8-3, S3C831B Set 1, Ba	nk 1 Register Values after RESET

Register Name	Mnemonic	Add	ress		Bit Values after RESET						
		Dec	Hex	7	6	5	4	3	2	1	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 Pull-up Resistors Enable Register	POPUR	226	E2H	0	0	0	0	0	0	0	0
	Locations E3	3H is no	t mappe	d.							
Port 1 Control Register (High Byte)	P1CONH	228	E4H	0	0	0	0	0	0	0	0
Port 1 Control Register (Low Byte)	P1CONL	229	E5H	0	0	0	0	0	0	0	0
Port 1 Interrupt Control Register	P1INT	230	E6H	0	0	0	0	0	0	0	0
Port 1 Interrupt Pending Register	P1PND	231	E7H	0	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	0	0
Port 3 Control Register (Low Byte)	P3CONL	235	EBH	0	0	0	0	0	0	0	0
Port 3 Pull-up Resistors Enable Register	P3PUR	236	ECH	0	0	0	0	0	0	0	0
Port Group 0 Control Register	PG0CON	237	EDH	0	0	0	0	0	0	0	0
Port Group 1 Control Register	PG1CON	238	EEH	0	0	0	0	0	0	0	0
Port Group 2 Control Register	PG2CON	239	EFH	0	0	0	0	0	0	0	0
Port 0 Data Register	P0	240	F0H	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	241	F1H	0	0	0	0	0	0	0	0
Port 2 Data Register	P2	242	F2H	0	0	0	0	0	0	0	0
Port 3 Data Register	P3	243	F3H	0	0	0	0	0	0	0	0
Port 4 Data Register	P4	244	F4H	0	0	0	0	0	0	0	0
Port 5 Data Register	P5	245	F5H	0	0	0	0	0	0	0	0
Port 6 Data Register	P6	246	F6H	0	0	0	0	0	0	0	0
Port 7 Data Register	P7	247	F7H	0	0	0	0	0	0	0	0
Port 8 Data Register	P8	248	F8H	0	0	0	0	0	0	0	0
	Location F9	H is not	mapped	ł.							
Timer 2 Counter (High Byte)	T2CNTH	250	F9H	0	0	0	0	0	0	0	0
Timer 2 Counter (Low Byte)	T2CNTL	251	FBH	0	0	0	0	0	0	0	0
Timer 2 Data Register (High Byte)	T2DATAH	252	FCH	1	1	1	1	1	1	1	1
Timer 2 Data Register (Low Byte)	T2DATAL	253	FDH	1	1	1	1	1	1	1	1
Timer 2 Control Register	Timer 2 Control Register         T2CON         254         FEH         0         <						0				
	Location FF	H is not	mapped	J.							



# **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 external interrupts, for more details see Figure 7-3.

## NOTE

Do not use stop mode if you are using an external clock source because  $X_{IN}$  input must be restricted internally to  $V_{SS}$  to reduce current leakage.

### Using RESET to Release Stop Mode

Stop mode is released when the RESET signal is released and returns to high level: all system and peripheral control registers are reset to their default hardware values and the contents of all data registers are retained. A reset operation automatically selects a slow clock fxx/16 because CLKCON.3 and CLKCON.4 are cleared to '00B'. After the programmed oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in ROM location 0100H.

## Using an External Interrupt to Release Stop Mode

External interrupts with an RC-delay noise filter circuit can be used to release Stop mode. Which interrupt you can use to release Stop mode in a given situation depends on the microcontroller's current internal operating mode. The external interrupts in the S3C831B interrupt structure that can be used to release Stop mode are:

External interrupts P1.0–P1.7 (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.

### How to Enter into Stop Mode

Handling STPCON register then writing Stop instruction (keep the order).

LD STPCON, #10100101B STOP NOP NOP NOP



## **IDLE MODE**

Idle mode is invoked by the instruction IDLE (opcode 6FH). In idle mode, CPU operations are halted while some peripherals remain active. During idle mode, the internal clock signal is gated away from the CPU, but all peripherals remain active. Port pins retain the mode (input or output) they had at the time idle mode was entered.

There are two ways to release idle mode:

- 1. Execute a reset. All system and peripheral control registers are reset to their default values and the contents of all data registers are retained. The reset automatically selects the slow clock fxx/16 because CLKCON.4 and CLKCON.3 are cleared to '00B'. If interrupts are masked, a reset is the only way to release idle mode.
- 2. Activate any enabled interrupt, causing idle mode to be released. When you use an interrupt to release idle mode, the CLKCON.4 and CLKCON.3 register values remain unchanged, and the currently selected clock value is used. The interrupt is then serviced. When the return-from-interrupt (IRET) occurs, the instruction immediately following the one that initiated idle mode is executed.



# 9 I/O PORTS

# OVERVIEW

The S3C831B microcontroller has four bit-programmable and five nibble-programmable I/O ports, P0–P8. The port 0–8 are all 8-bit ports. This gives a total of 72 I/O pins. Each port can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required. All ports of the S3C831B can be configured to input or output mode and P4– P8 are shared with LCD segment signals.

Table 9-1 gives you a general overview of the S3C831B I/O port functions.



Port	Configuration Options
0	1-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output mode selected by software; software assignable pull-up. Alternately P0.1–P0.7 can be used as T0CLK, T0CAP, T0OUT/T0PWM, T1CLK, T1OUT, T2CLK, T2OUT.
1	1-bit programmable I/O port. Input or push-pull output mode selected by software; software assignable pull-up (P1.03: Shmitt trigger input, P1.47: Input). P1.0–P1.7 can be used as inputs for external interrupts INT0–INT7 (with noise filter and interrupt control).
2	1-bit programmable I/O port. Schmitt trigger input or push-pull output mode selected by software; software assignable pull- up. Alternately P2.0–P2.7 can be used as AD0–AD7.
3	1-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-up. Alternately P3.0–P3.6 can be used as BUZ, SCK0, SO0, SI0, SCK1, SO1, SI1.
4	4-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-up. P4.0–P4.7 can alternately be used as outputs for LCD segment signals.
5 Il.com	4-bit programmable I/O port. Input or push-pull, open-drain output mode selected by software; software assignable pull-up. P5.0–P5.7 can alternately be used as outputs for LCD segment signals.
6	4-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output mode selected by software; software assignable pull-up. P6.0–P6.7 can alternately be used as outputs for LCD segment signals.
7	4-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output mode selected by software; software assignable pull-up. P7.0–P7.7 can alternately be used as outputs for LCD segment signals.
8	4-bit programmable I/O port. Schmitt trigger input or push-pull, open-drain output mode selected by software; software assignable pull-up. P8.0–P8.7 can alternately be used as outputs for LCD segment signals.

Table 9-1. S3C831B Port Configuration Overview



# PORT DATA REGISTERS

Table 9-2 gives you an overview of the register locations of all nine S3C831B 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.

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

Table 9-2. Port Data Register Summary



9-3

# PORT 0

Port 0 is an 8-bit I/O port with individually configurable pins. Port 0 pins are accessed directly by writing or reading the port 0 data register, P0 at location F0H in set 1, bank 1. P0.0–P0.7 can serve inputs, as outputs (push pull or open-drain) or you can configure the following alternative functions:

- Low-nibble pins (P0.1-P0.3): T0CLK, T0CAP, T0OUT/T0PWM
- High-nibble pins (P0.4-P0.7): T1CLK, T1OUT, T2CLK, T2OUT

# Port 0 Control Register

Port 0 has two 8-bit control registers: P0CONH for P0.4–P0.7 and P0CONL for P0.0–P0.3. A reset clears the P0CONH and P0CONL registers to "00H", configuring 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 0 control registers must also be enabled in the associated peripheral module.

## Port 0 Pull-up Resistor Enable Register (P0PUR)

Using the port 0 pull-up resistor enable register, P0PUR (E2H, set 1, bank 1), you can configure pull-up resistors to individual port 0 pins.

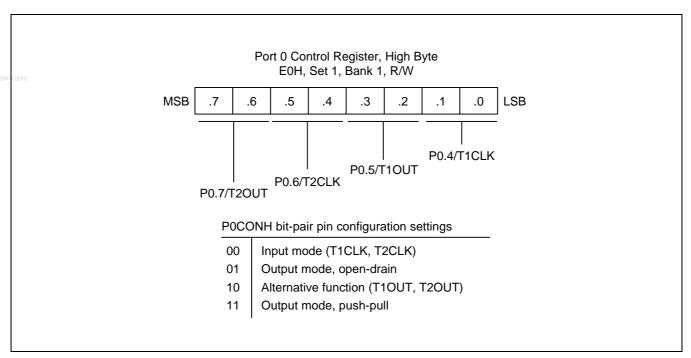
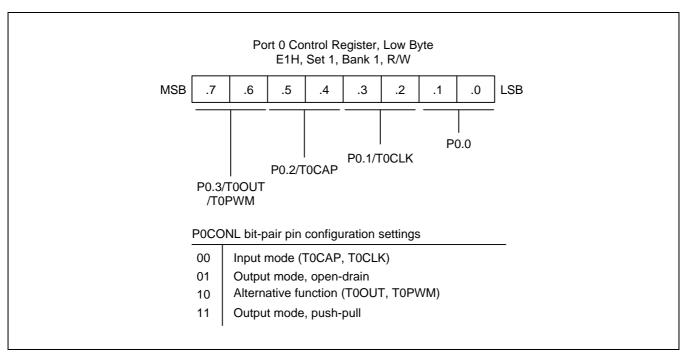


Figure 9-1. Port 0 High-Byte Control Register (P0CONH)







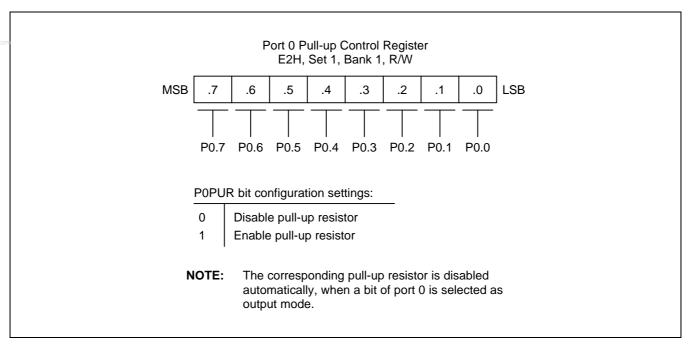


Figure 9-3. Port 0 Pull-up Control Register (P0PUR)



## PORT 1

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

- General-purpose I/O
- External interrupt inputs for INT0–INT7

Port 1 is accessed directly by writing or reading the port 1 data register, P1 at location F1H in set 1, bank 1.

## NOTE

The port 1 inputs can be disabled by PG2CON.5–.4 when the port is selected as input mode. Refer to the PG2CON register.

# Port 1 Control Register (P1CONH, P1CONL)

Port 1 pins are configured individually by bit-pair settings in two control registers located in set 1, bank 1: P1CONL (low byte, E5H) and P1CONH (high byte, E4H).

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

- Input with interrupt generation on falling edges (P1.0-.3: Schmitt trigger input).
- Input with interrupt generation on rising edges (P1.0-.3: Schmitt trigger input).
- Input with interrupt generation on falling/rising edges (P1.0-.3: Schmitt trigger input).

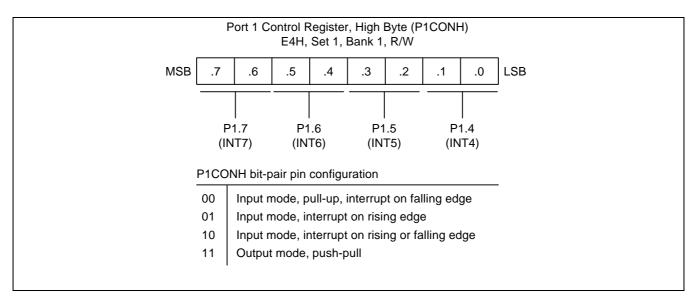
## Port 1 Interrupt Enable and Pending Registers (P1INT, P1PND)

To process external interrupts at the port 1 pins, two additional control registers are provided: the port 1 interrupt enable register P1INT (E6H, set 1, bank 1) and the port 1 interrupt pending register P1PND (E7H, set 1, bank 1).

The port 1 interrupt pending register P1PND 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 P1PND register at regular intervals.

When the interrupt enable bit of any port 1 pin is "1", a rising or falling edge at that pin will generate an interrupt request. The corresponding P1PND 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 P1PND bit.







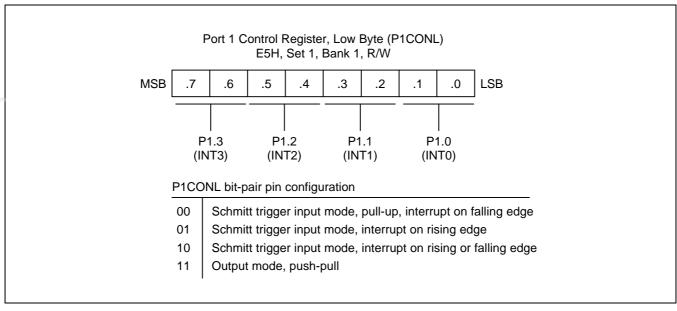
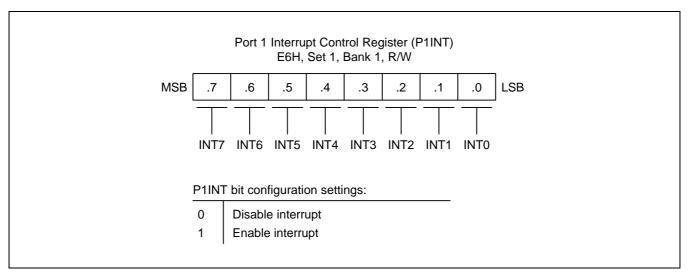


Figure 9-5. Port 1 Low-Byte Control Register (P1CONL)







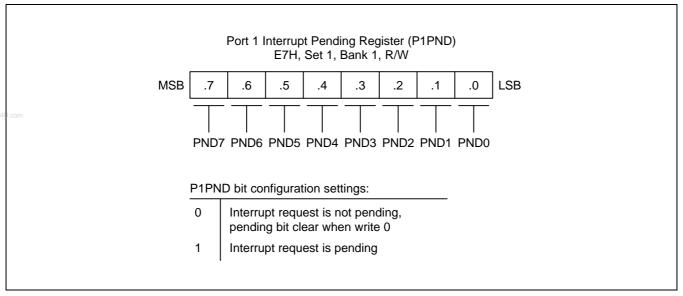


Figure 9-7. Port 1 Interrupt Pending Register (P1PND)



# 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, AD0–AD7. The pins are accessed directly by writing or reading the port 2 data register, P2 at location F2H in set 1, bank 1.

To individually configure the port 2 pins P2.0–P2.7, you make bit-pair settings in two control registers located in set 1, bank 1: P2CONL (low byte, E9H) and P2CONH (high byte, E8H). In input mode, ADC voltage input are also available.

## **Port 2 Control Registers**

Two 8-bit control registers are used to configure port 2 pins: P2CONL (E9H, set 1, Bank 1) for pins P2.0–P2.3 and P2CONH (E8H, set 1, Bank 1) for pins P2.4–P2.7. Each byte contains four bit-pairs and each bit-pair configures one port 2 pin. The P2CONH and the P2CONL registers also control the alternative functions.

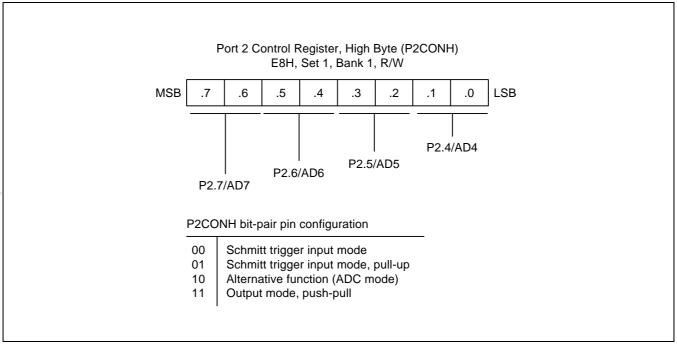


Figure 9-8. Port 2 High-Byte Control Register (P2CONH)



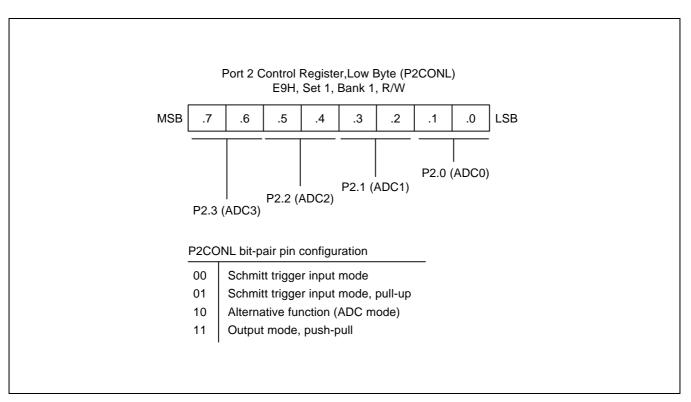


Figure 9-9. Port 2 Low-Byte Control Register (P2CONL)



Port 3 is an 8-bit I/O port with individually configurable pins. Port 3 pins are accessed directly by writing or reading the port 3 data register, P3 at location F3H in set 1, bank 1. P3.0–P3.7 can serve as inputs or as push-pull, open-drain outputs. You can configure the following alternative functions:

- BUZ, SCK0, SO0, SI0, SCK1, SO1, and SI1

## **Port 3 Control Registers**

Port 3 has two 8-bit control registers: P3CONH for P3.4–P3.7 and P3CONL for P3.0–P3.3. A reset clears the P3CONH and P3CONL registers to "00H", configuring all pins to input mode. You use control registers settings to select input or output mode, 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.

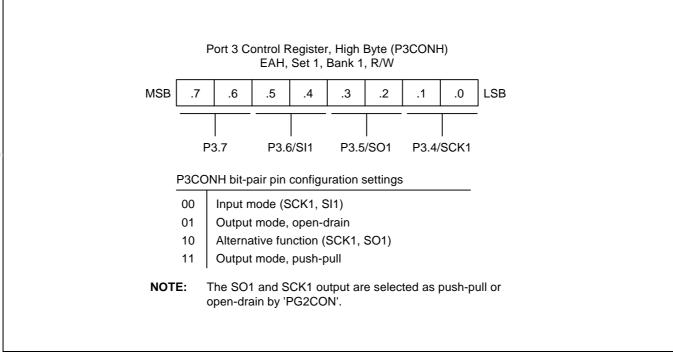


Figure 9-10. Port 3 High-Byte Control Register (P3CONH)



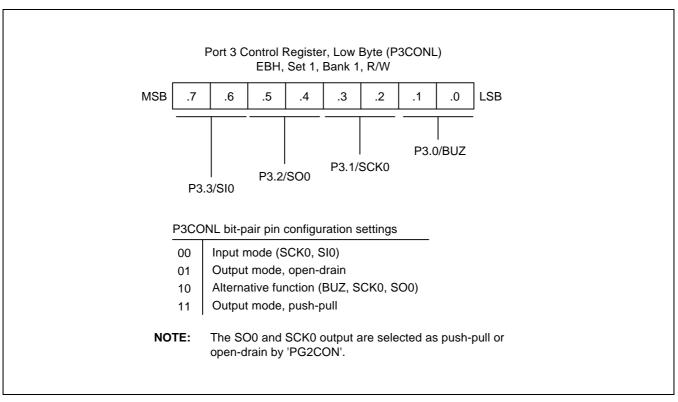


Figure 9-11. Port 3 Low-Byte Control Register (P3CONL)

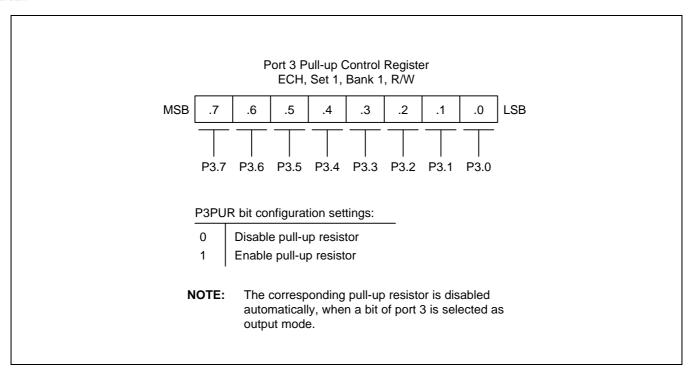


Figure 9-12. Port 3 Pull-up Control Register (P3PUR)



# **PORT 4, 5**

Port 4 and 5 are 8-bit I/O ports with nibble configurable pins, respectively. Port 4 and 5 pins are accessed directly by writing or reading the port 4 and 5 data registers, P4 at location F4H and P5 at location F5H in set 1, bank 1. P4.0–P4.7 and P5.0–P5.7 can serve as inputs (with or without pull-ups), as output (open drain or push-pull). And they can serve as segment pins for LCD, also.

## Port Group 0 Control Register

Port 4 and 5 have a 8-bit control register: PG0CON.4–.7 for P4.0–P4.7 and PG0CON.0–.3 for P5.0–P5.7. A reset clears the PG0CON register to "00H", configuring all pins to input mode.

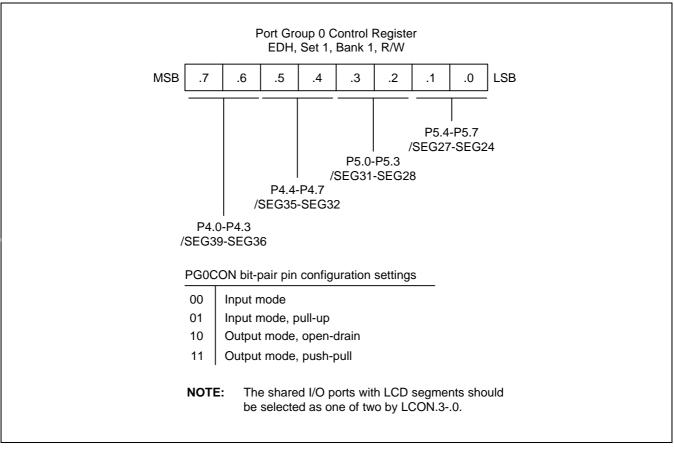


Figure 9-13. Port Group 0 Control Register (PG0CON)



# PORT 6, 7

Port 6 and 7 are 8-bit I/O port with nibble configurable pins, respectively. Port 6 and 7 pins are accessed directly by writing or reading the port 6 and 7 data registers, P6 at location F6H and P7 at location F7H in set 1, bank 1. P6.0–P6.7 and P7.0–P7.7 can serve as inputs (with or without pull-ups), as output (open drain or push-pull). And they can serve as segment pins for LCD also.

## Port Group 1 Control Register

Port 6 and 7 have a 8-bit control register: PG1CON.4–.7 for P6.0–P6.7 and PG1CON.0–.3 for P7.0–P7.7. A reset clears the PG1CON register to "00H", configuring all pins to input mode.

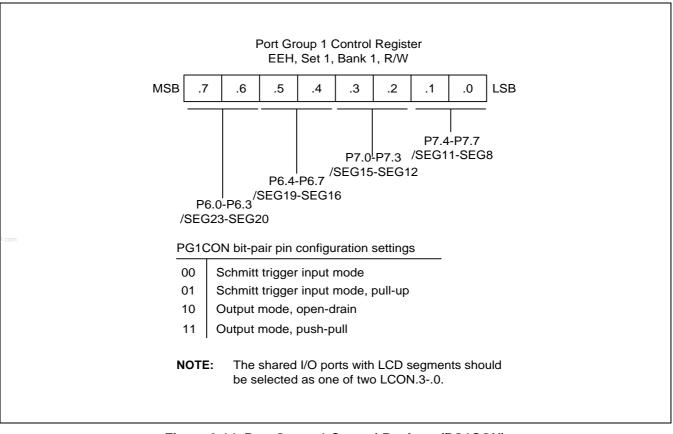


Figure 9-14. Port Group 1 Control Register (PG1CON)



# PORT 8

Port 8 is an 8-bit I/O port with nibble 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 or without pull-ups), as output (open drain or push-pull). And they can serve as segment pins for LCD also.

## Port Group 2 Control Register

Port 8 has a 8-bit control register: PG2CON for P8.0–P8.7. A reset clears the PG2CON register to "00H", configuring all pins to input mode.

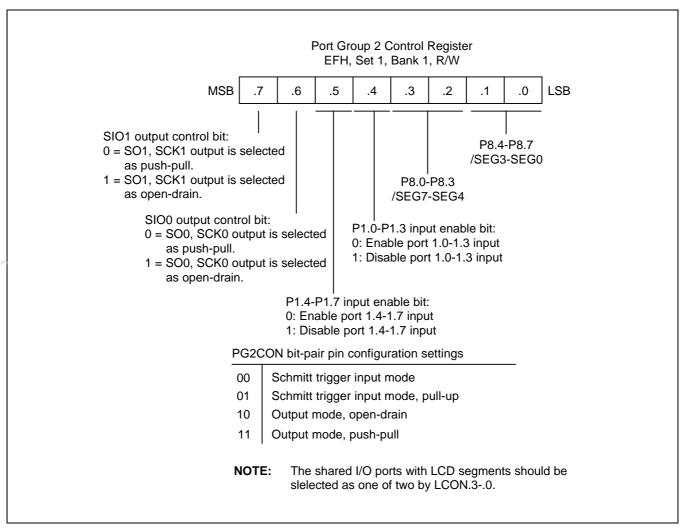


Figure 9-15. Port Group 2 Control Register (PG2CON)



NOTES



# **10** BASIC TIMER and TIMER 0

# OVERVIEW

The S3C831B has two default timers: an 8-bit *basic timer* and one 8-bit general-purpose timer/counter. The 8-bit timer/counter is called *timer 0*.

# **BASIC TIMER (BT)**

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

- As a watchdog timer to provide an automatic reset mechanism in the event of a system malfunction.
- To signal the end of the required oscillation stabilization interval after a reset or a stop mode release.

The functional components of the basic timer block are:

- Clock frequency divider (fxx divided by 4096, 1024, 128, or 16) with multiplexer
- 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH, read-only)
- Basic timer control register, BTCON (set 1, D3H, read/write)



## **BASIC TIMER CONTROL REGISTER (BTCON)**

The basic timer control register, BTCON, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function. It is located in set 1, address D3H, and is read/write addressable using Register addressing mode.

A reset clears BTCON to "00H". This enables the watchdog function and selects a basic timer clock frequency of fxx/4096. To disable the watchdog function, you must write the signature code "1010B" to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH), can be cleared at any time during normal operation by writing a "1" to BTCON.1. To clear the frequency dividers for all timers input clock, you write a "1" to BTCON.0.

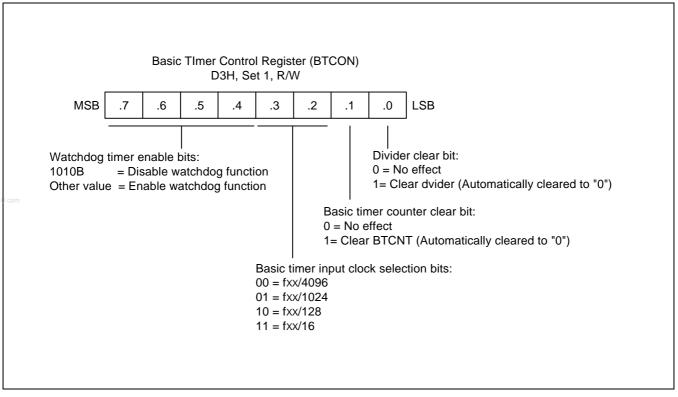


Figure 10-1. Basic Timer Control Register (BTCON)



#### **BASIC TIMER FUNCTION DESCRIPTION**

#### Watchdog Timer Function

You can program the basic timer overflow signal (BTOVF) to generate a reset by setting BTCON.7–BTCON.4 to any value other than "1010B". (The "1010B" value disables the watchdog function.) A reset clears BTCON to "00H", automatically enabling the watchdog timer function. A reset also selects the CPU clock (as determined by the current CLKCON register setting), divided by 4096, as the BT clock.

A reset whenever a basic timer counter overflow occurs. During normal operation, the application program must prevent the overflow, and the accompanying reset operation, from occurring. To do this, the BTCNT value must be cleared (by writing a "1" to BTCON.1) at regular intervals.

If a system malfunction occurs due to circuit noise or some other error condition, the BT counter clear operation will not be executed and a basic timer overflow will occur, initiating a reset. In other words, during normal operation, the basic timer overflow loop (a bit 7 overflow of the 8-bit basic timer counter, BTCNT) is always broken by a BTCNT clear instruction. If a malfunction does occur, a reset is triggered automatically.

#### **Oscillation Stabilization Interval Timer Function**

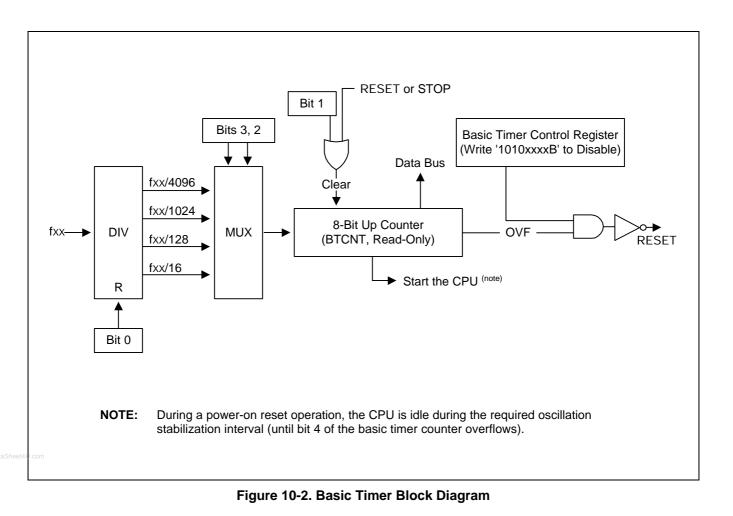
You can also use the basic timer to program a specific oscillation stabilization interval following a reset or when stop mode has been released by an external interrupt.

In stop mode, whenever a reset or an internal and an external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of fxx/4096 (for reset), or at the rate of the preset clock source (for an internal and an external interrupt). When BTCNT.3 overflows, a signal is generated to indicate that the stabilization interval has elapsed and to gate the clock signal off to the CPU so that it can resume normal operation.

In summary, the following events occur when stop mode is released:

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





# 8-BIT TIMER/COUNTER 0

Timer/counter 0 has three operating modes, one of which you select using the appropriate T0CON setting:

- Interval timer mode
- Capture input mode with a rising or falling edge trigger at the P0.2 pin
- PWM mode

Timer/counter 0 has the following functional components:

- Clock frequency divider (fxx divided by 1024, 256, 64, 8, or 1) with multiplexer
- External clock input (P0.1, T0CLK)
- 8-bit counter (T0CNT), 8-bit comparator, and 8-bit reference data register (T0DATA)
- I/O pins for capture input, match output, or PWM output (P0.2/T0CAP, P0.3/T0OUT, P0.3/T0PWM)
- Timer 0 overflow interrupt (IRQ0, vector E2H) and match/capture interrupt (IRQ0, vector E0H) generation
- Timer 0 control register, T0CON (set 1, E2H, bank 0, read/write)

# TIMER/COUNTER 0 CONTROL REGISTER (T0CON)

You use the timer 0 control register, T0CON, to

- Select the timer 0 operating mode (interval timer, capture mode, or PWM mode)
- Select the timer 0 input clock frequency
- Clear the timer 0 counter, T0CNT
- Enable the timer 0 overflow interrupt or timer 0 match/capture interrupt
- Clear timer 0 match/capture interrupt pending condition



T0CON is located in set 1, bank 0, at address E2H, and is read/write addressable using Register addressing mode.

A reset clears T0CON to "00H". This sets timer 0 to normal interval timer mode, selects an input clock frequency of fxx/1024, and disables all timer 0 interrupts. You can clear the timer 0 counter at any time during normal operation by writing a "1" to T0CON.2.

The timer 0 overflow interrupt (T0OVF) is interrupt level IRQ0 and has the vector address E2H. When a timer 0 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware or must be cleared by software.

To enable the timer 0 match/capture interrupt (IRQ0, vector E0H), you must write T0CON.1 to "1". To detect a match/capture interrupt pending condition, the application program polls INTPND.1. When a "1" is detected, a timer 0 match or capture interrupt is pending. When the interrupt request has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 0 match/capture interrupt pending bit, INTPND.1.

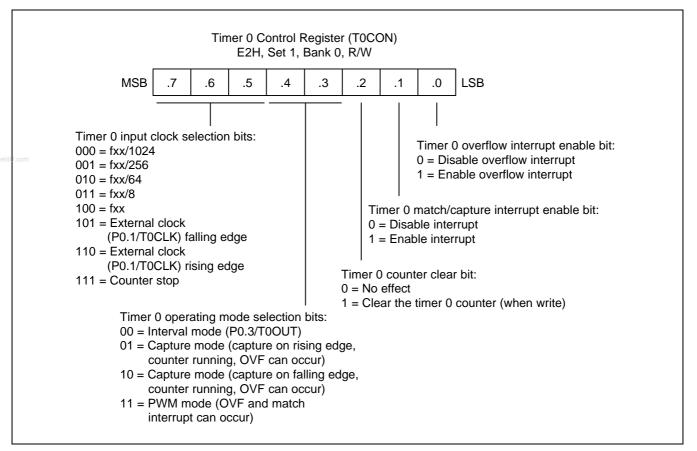


Figure 10-3. Timer 0 Control Register (T0CON)



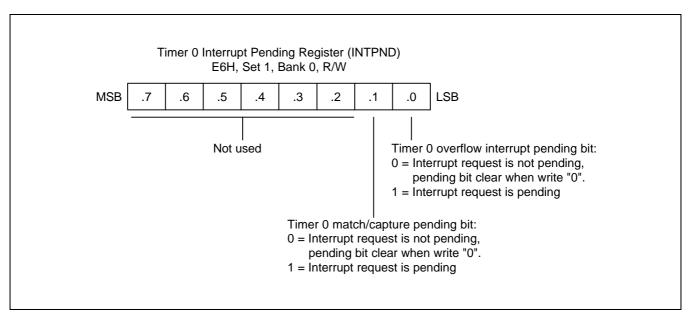


Figure 10-4. Timer 0 Interrupt Pending Register (INTPND)



#### **TIMER 0 FUNCTION DESCRIPTION**

#### Timer 0 Interrupts (IRQ0, Vectors E0H and E2H)

The timer 0 can generate two interrupts: the timer 0 overflow interrupt (T0OVF), and the timer 0 match/ capture interrupt (T0INT). T0OVF is belongs to interrupt level IRQ0, vector E2H. T0INT also belongs to interrupt level IRQ0, but is assigned the separate vector address, E0H.

A timer 0 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced or should be cleared by software in the interrupt service routine by writing a "0" to the INTPND.0 interrupt pending bit. However, the timer 0 match/capture interrupt pending condition must be cleared by the application's interrupt service routine by writing a "0" to the INTPND.1 interrupt pending bit.

#### **Interval Timer Mode**

In interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 0 reference data register, T0DATA. The match signal generates a timer 0 match interrupt (T0INT, vector E0H) and clears the counter.

If, for example, you write the value "10H" to T0DATA, the counter will increment until it reaches "10H". At this point, the timer 0 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer 0 output pin is inverted (see Figure 10-5).

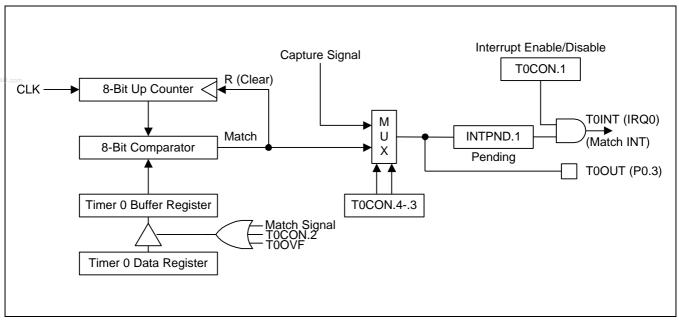


Figure 10-5. Simplified Timer 0 Function Diagram: Interval Timer Mode



## **Pulse Width Modulation Mode**

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the T0PWM (P0.3) pin. As in interval timer mode, a match signal is generated when the counter value is identical to the value written to the timer 0 data register. In PWM mode, however, the match signal does not clear the counter. Instead, it runs continuously, overflowing at "FFH", and then continues incrementing from "00H".

Although you can use the match signal to generate a timer 0 overflow interrupt, interrupts are not typically used in PWM-type applications. Instead, the pulse at the T0PWM (P0.3) 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<sub>CLK</sub> × 256 (see Figure 10-6).

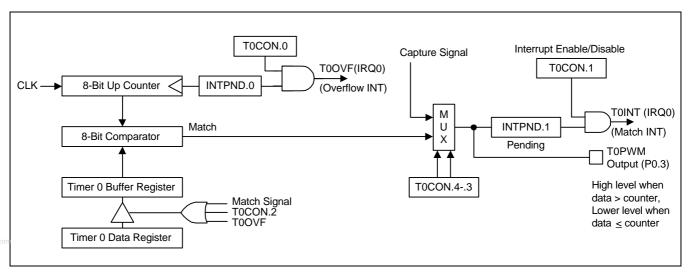


Figure 10-6. Simplified Timer 0 Function Diagram: PWM Mode



## **Capture Mode**

In capture mode, a signal edge that is detected at the T0CAP (P0.2) pin opens a gate and loads the current counter value into the timer 0 data register. You can select rising or falling edges to trigger this operation.

Timer 0 also gives you capture input source: the signal edge at the T0CAP (P0.2) pin. You select the capture input by setting the values of the timer 0 capture input selection bits in the port 0 control register, P0CONL.5–.4, (set 1, bank 1, E1H). When P0CONL.5–.4 is "00", the T0CAP input is selected.

Both kinds of timer 0 interrupts can be used in capture mode: the timer 0 overflow interrupt is generated whenever a counter overflow occurs; the timer 0 match/capture interrupt is generated whenever the counter value is loaded into the timer 0 data register.

By reading the captured data value in T0DATA, and assuming a specific value for the timer 0 clock frequency, you can calculate the pulse width (duration) of the signal that is being input at the T0CAP pin (see Figure 10-7).

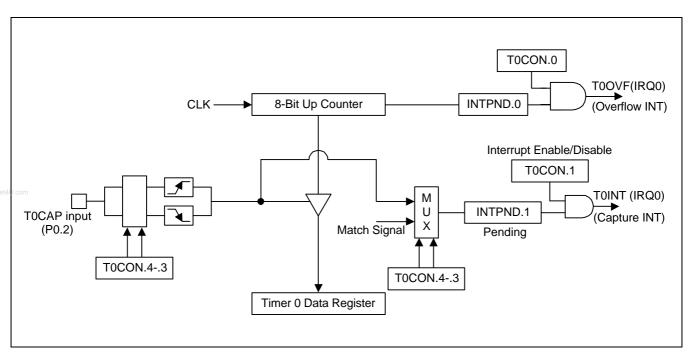


Figure 10-7. Simplified Timer 0 Function Diagram: Capture Mode



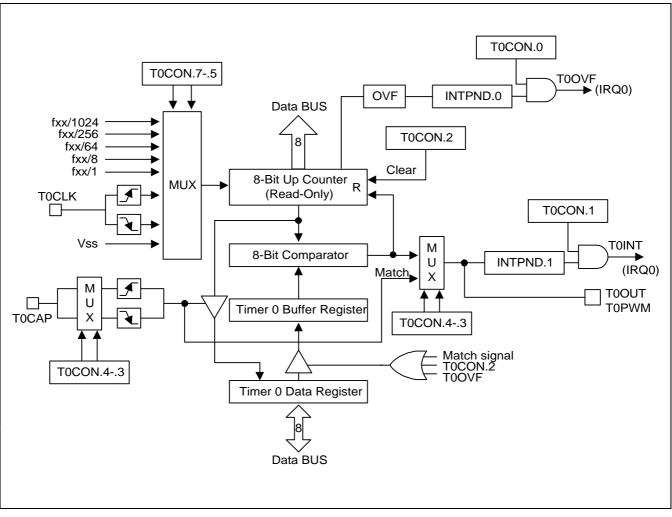


Figure 10-8. Timer 0 Block Diagram



NOTES



# 8-BIT TIMER 1

# OVERVIEW

The 8-bit timer 1 is an 8-bit general-purpose timer. Timer 1 has the interval timer mode by using the appropriate T1CON setting.

Timer 1 has the following functional components:

- Clock frequency divider (fxx divided by 256, 64, 8 or 1) with multiplexer
- External clock input (P0.4/T1CLK)
- 8-bit counter (T1CNT), 8-bit comparator, and 8-bit reference data register (T1DATA)
- Timer 1 interrupt (IRQ1, vector E6H) generation
- Timer 1 control register, T1CON (set 1, Bank 0, E5H, read/write)

## **FUNCTION DESCRIPTION**

#### **Interval Timer Function**

The timer 1 can generate an interrupt, the timer 1 match interrupt (T1INT). T1INT belongs to interrupt level IRQ1, and is assigned the separate vector address, E6H.

The T1INT pending condition should be cleared by software when it has been serviced. Even though T1INT is disabled, the application's service routine can detect a pending condition of T1INT by the software and execute it's sub-routine. When this case is used, the T1INT pending bit must be cleared by the application subroutine by writing a "0" to the T1CON.0 pending bit.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the Timer 1 reference data registers, T1DATA. The match signal generates a timer 1 match interrupt (T1INT, vector E6H) and clears the counter.

If, for example, you write the value 10H to T1DATA and 0EH to T1CON, the counter will increment until it reaches 10H. At this point, the Timer 1 interrupt request is generated, the counter value is reset, and counting resumes.



# TIMER 1 CONTROL REGISTER (T1CON)

You use the timer 1 control register, T1CON, to

- Enable the timer 1 operating (interval timer)
- Select the timer 1 input clock frequency
- Clear the timer 1 counter, T1CNT
- Enable the timer 1 interrupt and clear timer 1 interrupt pending condition

T1CON is located in set 1, bank 0, at address E5H, and is read/write addressable using register addressing mode.

A reset clears T1CON to "00H". This sets timer 1 to disable interval timer mode, and disables timer 1 interrupt. You can clear the timer 1 counter at any time during normal operation by writing a "1" to T1CON.3

To enable the timer 1 interrupt (IRQ1, vector E6H), you must write T1CON.2, and T1CON.1 to "1". To detect an interrupt pending condition when T1INT is disabled, the application program polls pending bit, T1CON.0. When a "1" is detected, a timer 1 interrupt is pending. When the T1INT sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 1 interrupt pending bit, T1CON.0.

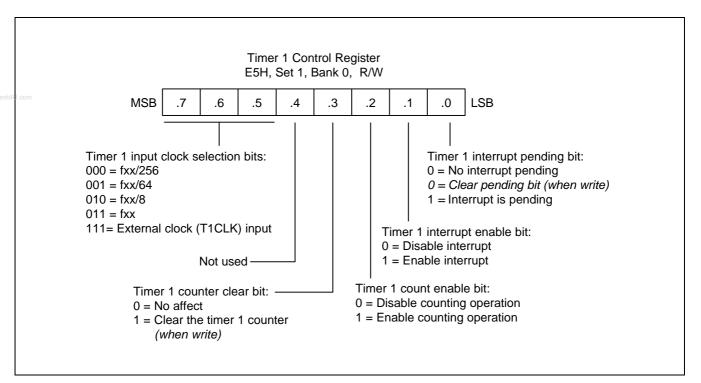


Figure 11-1. Timer 1 Control Register (T1CON)



# **BLOCK DIAGRAM**

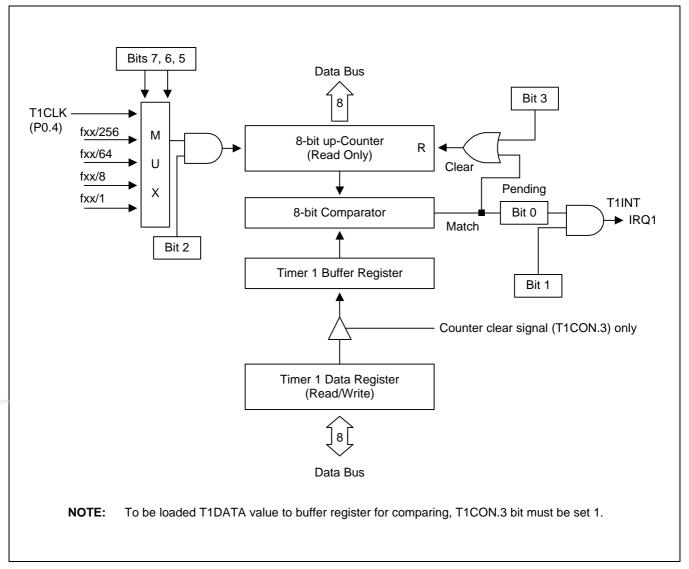


Figure 11-2. Timer 1 Functional Block Diagram



NOTES



# **12** 16-BIT TIMER 2

# OVERVIEW

The 16-bit timer 2 is an 16-bit general-purpose timer. Timer 2 has the interval timer mode by using the appropriate T2CON setting.

Timer 2 has the following functional components:

- Clock frequency divider (fxx divided by 256, 64, 8 or 1) with multiplexer
- External clock input (T2CLK)
- 16-bit counter (T2CNTH/L), 16-bit comparator, and 16-bit reference data register (T2DATAH/L)
- Timer 2 interrupt (IRQ1, vector E4H) generation
- Timer 2 control register, T2CON (set 1, Bank 1, FEH, read/write)

# **FUNCTION DESCRIPTION**

#### **Interval Timer Function**

The timer 2 can generate an interrupt, the timer 2 match interrupt (T2INT). T2INT belongs to interrupt level IRQ1, and is assigned the separate vector address, E4H.

The T2INT pending condition should be cleared by software when it has been serviced. Even though T2INT is disabled, the application's service routine can detect a pending condition of T2INT by the software and execute it's sub-routine. When this case is used, the T2INT pending bit must be cleared by the application subroutine by writing a "0" to the T2CON.0 pending bit.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the Timer 2 reference data registers, T2DATA. The match signal generates a timer 2 match interrupt (T2INT, vector E4H) and clears the counter.

If, for example, you write the value 0010H to T2DATAH/L and 0EH to T2CON, the counter will increment until it reaches 10H. At this point, the Timer 2 interrupt request is generated, the counter value is reset, and counting resumes.



# **TIMER 2 CONTROL REGISTER (T2CON)**

You use the timer 2 control register, T2CON, to

- Enable the timer 2 operating (interval timer)
- Select the timer 2 input clock frequency
- Clear the timer 2 counter, T2CNTH/L
- Enable the timer 2 interrupt and clear timer 2 interrupt pending condition

T2CON is located in set 1, bank 1, at address FEH, and is read/write addressable using register addressing mode.

A reset clears T2CON to "00H". This sets timer 2 to disable interval timer mode, and disables timer 2 interrupt. You can clear the timer 2 counter at any time during normal operation by writing a "1" to T2CON.3

To enable the timer 2 interrupt (IRQ1, vector E4H), you must write T2CON.2, and T2CON.1 to "1". To detect an interrupt pending condition when T2INT is disabled, the application program polls pending bit, T2CON.0. When a "1" is detected, a timer 2 interrupt is pending. When the T2INT sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the timer 2 interrupt pending bit, T2CON.0.

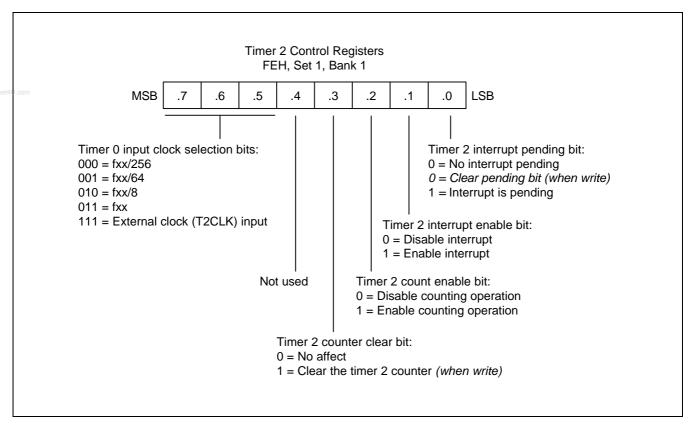


Figure 12-1. Timer 2 Control Register (T2CON)



# **BLOCK DIAGRAM**

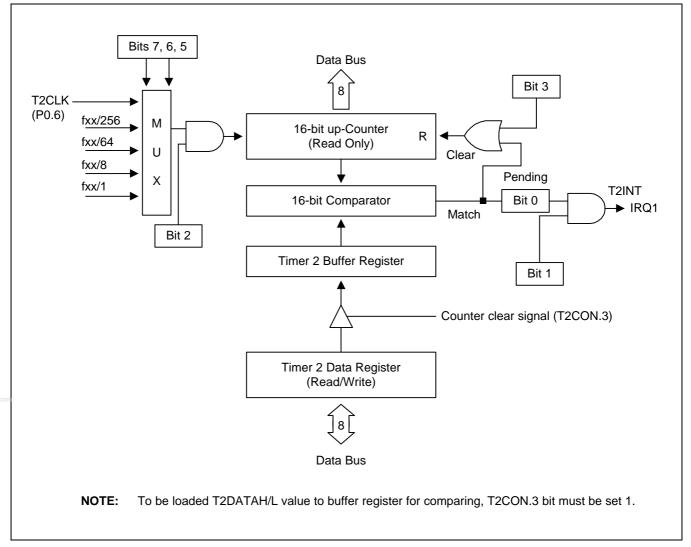


Figure 12-2. Timer 2 Functional Block Diagram



NOTES



# 13 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 6 of the watch timer control register, WTCON.6 to "1". And if you want to service watch timer overflow interrupt (IRQ3, vector F2H), then set the WTCON.1 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 50 ms, 0.5 and 1-second intervals by setting Watch timer speed selection bits (WTCON.3 – .2).

The watch timer can generate a steady 1 kHz, 1.5 kHz, 3 kHz, or 6 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 50 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 System Clock Source only
- Clock Source Generation for LCD Controller (f<sub>LCD</sub>)
- I/O pin for Buzzer Output Frequency Generator (P3.0, BUZ)
- Timing Tests in High-Speed Mode
- Watch timer overflow interrupt (IRQ3, vector F2H) generation
- Watch timer control register, WTCON (set 1, bank 0, E8H, read/write)



## WATCH TIMER CONTROL REGISTER (WTCON)

The watch timer control register, WTCON is used to select the watch timer interrupt time and Buzzer signal, to enable or disable the watch timer function. It is located in set 1, bank 0 at address E8H, and is read/write addressable using register addressing mode.

A reset clears WTCON to "00H". This disable the watch timer.

So, if you want to use the watch timer, you must write appropriate value to WTCON.

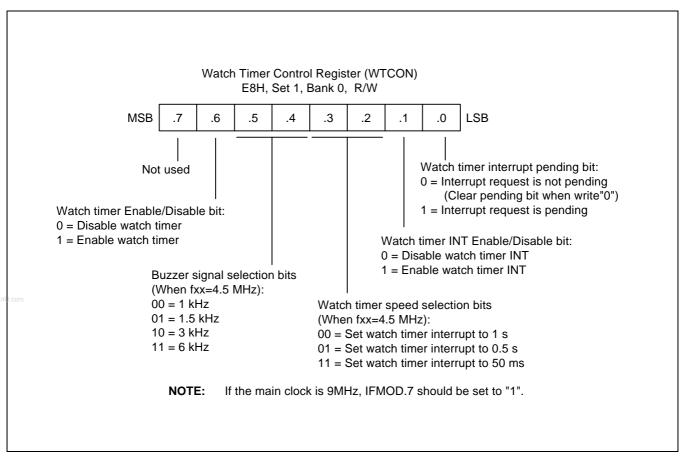


Figure 13-1. Watch Timer Control Register (WTCON)



# WATCH TIMER CIRCUIT DIAGRAM

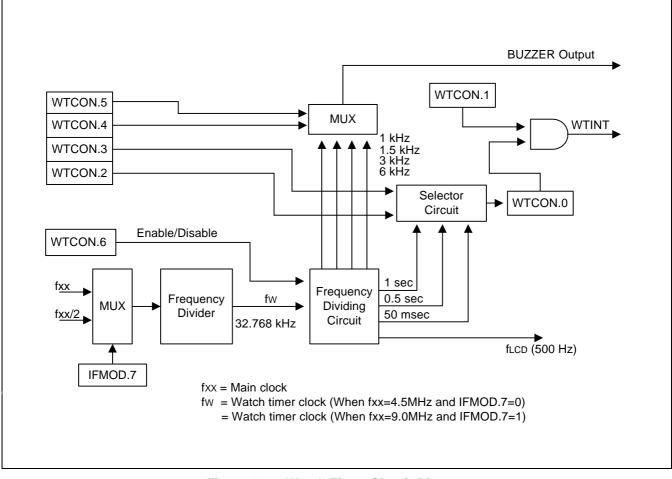


Figure 13-2. Watch Timer Circuit Diagram



NOTES



# 14 LCD CONTROLLER/DRIVER

# OVERVIEW

The S3C831B microcontroller can directly drive an up-to-20-digit (40-segment) LCD panel. The LCD block has the following components:

- LCD controller/driver
- Display RAM (00H–13H) for storing display data in page 9
- 40 segment output pins (SEG0–SEG39)
- Four common output pins (COM0-COM3)
- Three LCD operating power supply pins ( $V_{LC0-}$   $V_{LC2}$ ) and bias pin for LCD driving voltage ( $V_{LCD}$ )
- LCD voltage dividing resistors

Bit settings in the LCD mode register, LMOD, determine the LCD frame frequency, duty and bias, and LCD voltage dividing resistors.

The LCD control register LCON turns the LCD display on and off and switches current to the LCD voltage dividing resistors for the display. LCD data stored in the display RAM locations are transferred to the segment signal pins automatically without program control.

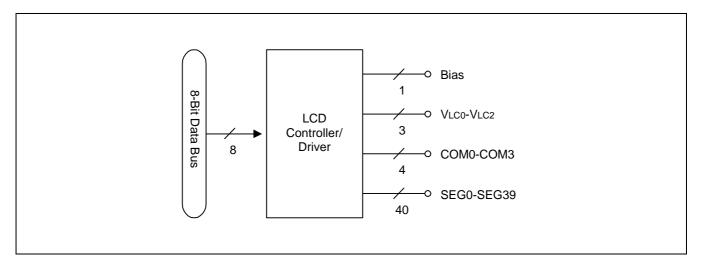


Figure 14-1. LCD Function Diagram



# LCD CIRCUIT DIAGRAM

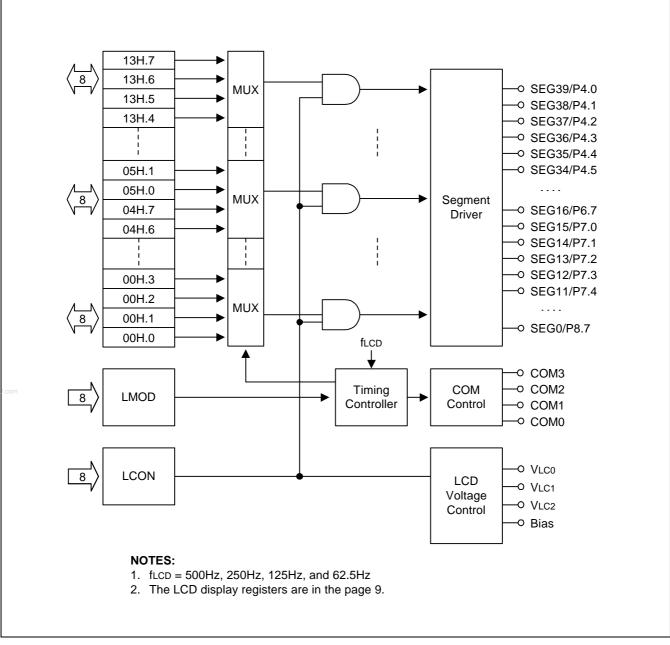


Figure 14-2. LCD Circuit Diagram



## LCD RAM ADDRESS AREA

RAM addresses 00H–13H of page 9 are used as LCD data memory. When the bit value of a display segment is "1", the LCD display is turned on; when the bit value is "0", the display is turned off.

Display RAM data are sent out through segment pins SEG0–SEG39 using a direct memory access (DMA) method that is synchronized with the  $f_{LCD}$  signal. RAM addresses in this location that are not used for LCD display can be allocated to general-purpose use.

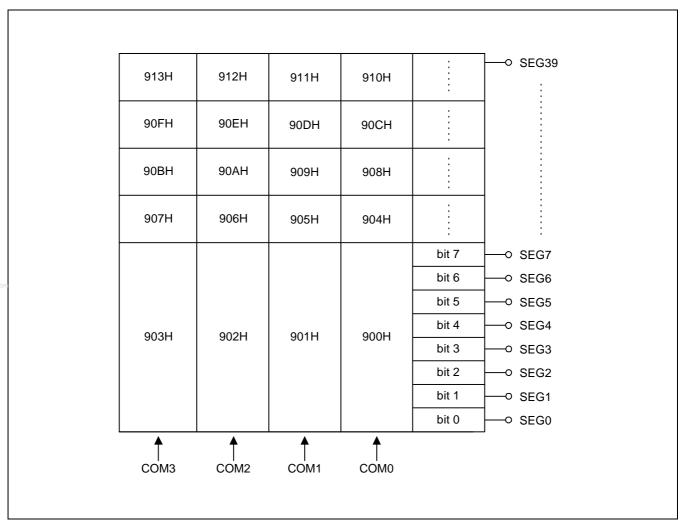


Figure 14-3. LCD Display Data RAM Organization



# LCD CONTROL REGISTER (LCON), F1H at BANK 0 of SET 1

# Table 14-1. LCD Control Register (LCON) Organization

LCON Bit	Setting	Description				
LCON.7	0	LCD output is low and the current for dividing the resistors is cut off.				
	1	If LMOD.3 = "0", LCD display is turned off. (All LCD segments are off signal output.)				
		If LMOD.3 = "1", output COM and SEG signals in display mode.				
LCON.64	Not used for	the S3C831B.				
LCON.30	0000	Select LCD SEG0–39.				
	0001	Select LCD SEG0–35/P4.0–4.3 as I/O port.				
	0010	Select LCD SEG0–31/P4 as I/O port.				
	0011	Select LCD SEG0–27/P4, P5.0–P5.3 as I/O port.				
	0100	Select LCD SEG0–23/P4, P5 as I/O port.				
	0101	Select LCD SEG0–19/P4, P5, P6.0–P6.3 as I/O port.				
	0110	Select LCD SEG0–15/P4, P5, P6 as I/O port.				
	0111	Select LCD SEG0–11/P4, P5, P6, P7.0–P7.3 as I/O port.				
	1000	Select LCD SEG0–7/P4, P5, P6, P7 as I/O port.				
	1001	Select LCD SEG0–3/P4, P5, P6, P7, P8.0–P8.3 as I/O port.				
	1010	All I/O port (P4–P8)				



## LCD MODE REGISTER (LMOD)

The LCD mode control register LMOD is mapped to RAM address F2H at bank 0 of set 1.

LMOD controls these LCD functions:

- Duty and bias selection (LMOD.3–LMOD.0)
- LCDCK clock frequency selection (LMOD.5–LMOD.4)
- LCD voltage dividing resistors selection (LMOD.6)
- LCD common signal enable or disable selection (LMOD.7)

The LCD clock signal, LCDCK, determines the frequency of COM signal scanning of each segment output. This is also referred to as the 'frame frequency.' Since LCDCK is generated by dividing the watch timer clock (fw), the watch timer must be enabled when the LCD display is turned on. RESET clears the LMOD register values to logic zero. This produces the following LCD control settings:

- Display is turned off
- LCDCK frequency is 62.5 Hz (at fx = 4.5 MHz) from the watch timer clock.

The LCD display can continue to operate during idle mode.

LCDCK Frequency	Static	1/2 Duty	1/3 Duty	1/4 Duty
62.5 Hz	62.5 Hz	31.3 Hz	20.8 Hz	15.6 Hz
125 Hz	125 Hz	62.5 Hz	41.7 Hz	31.3 Hz
250 Hz	250 Hz	125 Hz	83.3 Hz	62.5 Hz
500 Hz	500 Hz	250 Hz	166.7 Hz	125 Hz

#### Table 14-2. LCD Clock Signal (LCDCK) Frame Frequency

**NOTE:** fx = 4.5 MHz



 Table 14-3. LCD Mode Control Register (LMOD) Organization, F2H at Bank 0 of Set 1

LMOD.7	COM Signal Enable/Disable Bit
0	Enable COM signal
1	Disable COM signal

LMOD.6	LCD Voltage Dividing Resistors Control Bit						
0	Internal voltage dividing resistors						
1	External voltage dividing resistors; Internal voltage dividing resistors are off.						

LMOD.5	LMOD.4	LCD Clock (LCDCK) Frequency (When fxx = 4.5MHz)					
0	0	62.5 Hz at fx = 4.5 MHz					
0	1	125 Hz at fx = 4.5 MHz					
1	0	250 Hz at fx = 4.5 MHz					
1	1	500 Hz at fx = 4.5 MHz					

NOTE: If the main clock is 9.0MHz, IFMOD.7 should be set to "1".

LMOD.3	LMOD.2	LMOD.1	LMOD.0	Duty and Bias Selection for LCD Display
0	х	х	х	LCD display off (LCD off signal output)
1	0	0	0	1/4 duty, 1/3 bias
4U.com 1	0	0	1	1/3 duty, 1/3 bias
1	0	1	1	1/3 duty, 1/2 bias
1	0	1	0	1/2 duty, 1/2 bias
1	1	0	0	Static

#### Table 14-4. Maximum Number of Display Digits per Duty Cycle

LCD Duty	LCD Bias	COM Output Pins	Maximum Seg Display
Static	Static	COM0	40
1/2	1/2	COM0–COM1	40 x 2
1/3	1/2	COM0–COM2	40 x 3
1/3	1/3	COM0–COM2	40 x 3
1/4	1/3	COM0–COM3	40 x 4



#### LCD DRIVE VOLTAGE

The LCD display is turned on only when the voltage difference between the common and segment signals is greater than  $V_{LCD}$ . The LCD display is turned off when the difference between the common and segment signal voltages is less than  $V_{LCD}$ . The turn-on voltage, +  $V_{LCD}$  or -  $V_{LCD}$ , is generated only when both signals are the selected signals of the bias. Table 14-5 shows LCD drive voltages for static mode, 1/2 bias, and 1/3 bias.

LCD Power Supply	Static Mode	1/2 Bias	1/3 Bias
V <sub>LC0</sub>	V <sub>LCD</sub>	V <sub>LCD</sub>	V <sub>LCD</sub>
V <sub>LC1</sub>	_	1/2 V <sub>LCD</sub>	2/3 V <sub>LCD</sub>
V <sub>LC2</sub>	_	1/2 V <sub>LCD</sub>	1/3 V <sub>LCD</sub>
V <sub>ss</sub>	0 V	0 V	0 V

Table 14-5. LCD Drive Voltage Values

**NOTE:** The LCD panel display may deteriorate if a DC voltage is applied that lies between the common and segment signal voltage. Therefore, always drive the LCD panel with AC voltage.

#### LCD COM/SEG SIGNALS

The 40 LCD segment signal pins are connected to corresponding display RAM locations at 00H–13H at page 7. The corresponding bits of the display RAM are synchronized with the common signal output pins COM0, COM1, COM2, and COM3.

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 is sent to the corresponding segment pin. Each bias has select and no-select signals.

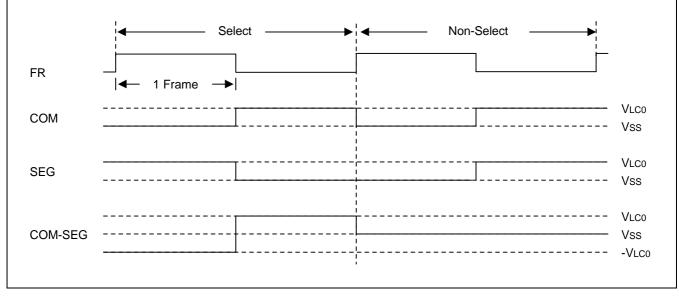


Figure 14-4. Select/No-Select Bias Signals in Static Display Mode



#### LCD CONTROLLER/DRIVER

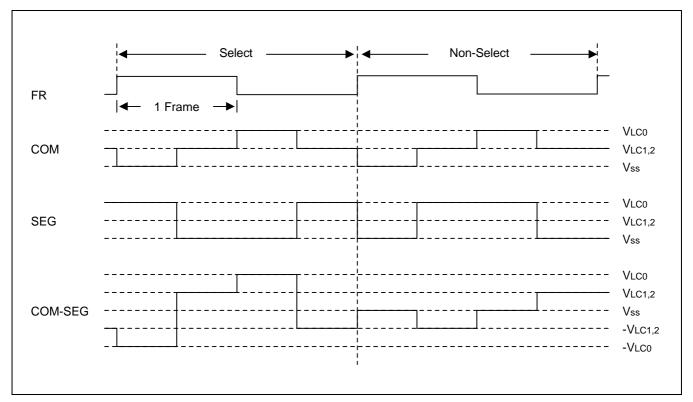


Figure 14-5. Select/No-Select Bias Signals in 1/2 Duty, 1/2 Bias Display Mode

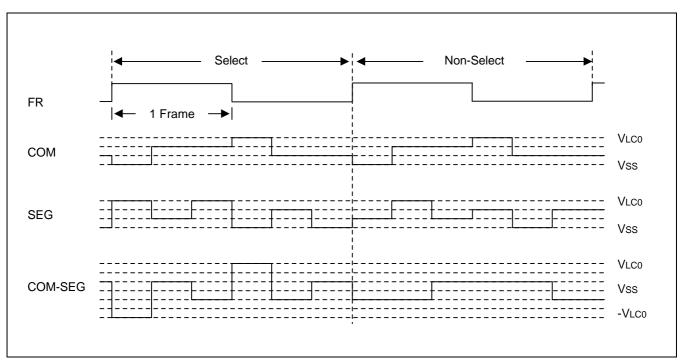
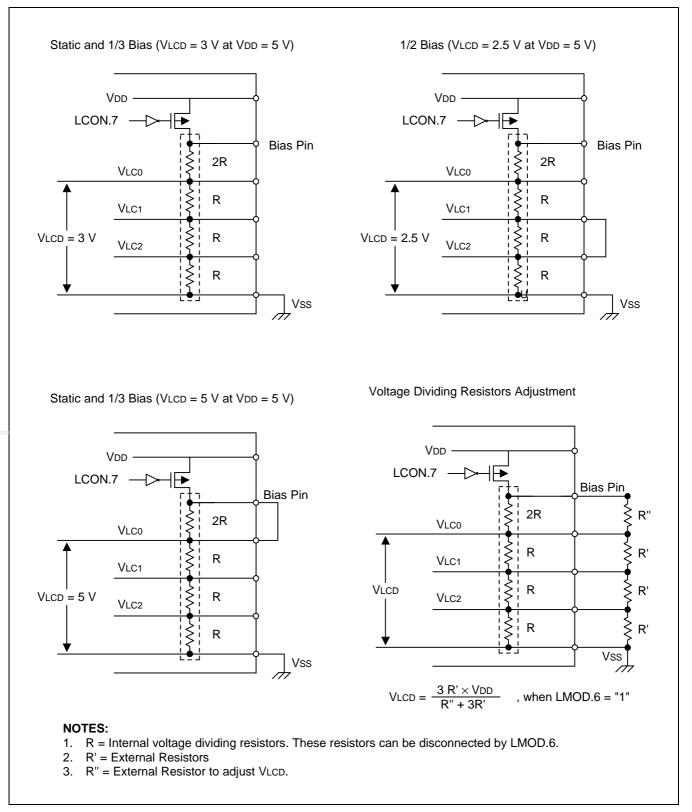


Figure 14-6. Select/No-Select Bias Signals in 1/3 Duty, 1/3 Bias Display Mode





#### Figure 14-7. Voltage Dividing Resistor Circuit Diagram



NOTES



www.DataSheet4U.com

## 15 8-BIT ANALOG-TO-DIGITAL CONVERTER

## OVERVIEW

The 8-bit A/D converter (ADC) module uses successive approximation logic to convert analog levels entering at one of the eight input channels to equivalent 8-bit digital values. The analog input level must lie between the  $AV_{DD}$  and  $V_{SS}$  values. The A/D converter has the following components:

- Analog comparator with successive approximation logic
- D/A converter logic (resistor string type)
- ADC control register (ADCON)
- Eight multiplexed analog data input pins (AD0–AD7)
- 8-bit A/D conversion data output register (ADDATA)
- 8-bit digital input port (Alternately, I/O port.)
- AV<sub>DD</sub> pin is internally connected to V<sub>DD</sub>.

## **FUNCTION DESCRIPTION**

To initiate an analog-to-digital conversion procedure, at first you must set with alternative function for ADC input enable at port 2, the pin set with alternative function can be used for ADC analog input. And you write the channel selection data in the A/D converter control register ADCON.4–.6 to select one of the eight analog input pins (AD0–7) and set the conversion start or enable bit, ADCON.0. The read-write ADCON register is located in set 1, bank 0, at address EFH. The pins which 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 80H (the approximate half-way point of an 8-bit register). This register is then updated automatically during each conversion step. The successive approximation block performs 8-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 ADDATA register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATA before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

#### NOTE

Because the 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 5 steps (5 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 8-bit conversion: When fxx/8 is selected for conversion clock with an 4.5 MHz fxx clock frequency, one clock cycle is 1.78 us. Each bit conversion requires 5 clocks, the conversion rate is calculated as follows:

5 clocks/bit  $\times$  8 bits + set-up time = 50 clocks, 50 clock  $\times$  1.78 us = 89 us at 0.56 MHz (4.5 MHz/8)

Note that A/D converter needs at least  $25\mu s$  for conversion time.

#### A/D CONVERTER CONTROL REGISTER (ADCON)

The A/D converter control register, ADCON, is located at address EFH in set 1, bank 0. It has three functions:

- Analog input pin selection (bits 4, 5, and 6)
- End-of-conversion status detection (bit 3)
- ADC clock selection (bits 2 and 1)
- A/D operation start or enable (bit 0)

After a reset, the start bit is turned off. You can select only one analog input channel at a time. Other analog input pins (AD0–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.

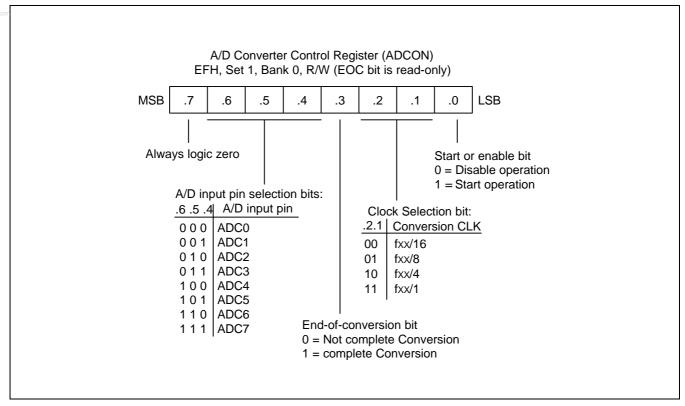


Figure 15-1. A/D Converter Control Register (ADCON)



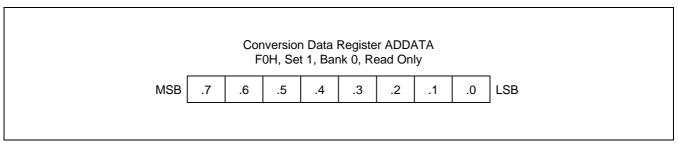


Figure 15-2. A/D Converter Data Register (ADDATA)

#### INTERNAL REFERENCE VOLTAGE LEVELS

In the ADC function block, the analog input voltage level is compared to the reference voltage. The analog input level must remain within the range  $V_{SS}$  to  $AV_{DD}$  (The  $AV_{DD}$  pin is internally connected with  $V_{DD}$ ).

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  $1/2 \text{ AV}_{DD}$ .

### **BLOCK DIAGRAM**

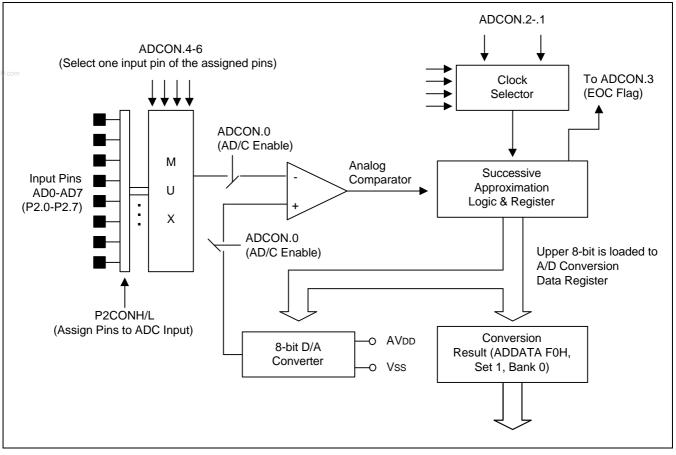


Figure 15-3. A/D Converter Functional Block Diagram



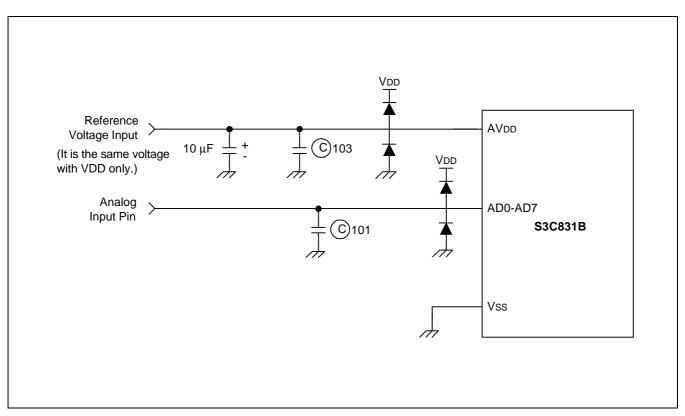


Figure 15-4. Recommended A/D Converter Circuit for Highest Absolute Accuracy



## 16 SERIAL I/O INTERFACE

## OVERVIEW

Serial I/O modules, SIO0 and SIO1 can interface with various types of external device that require serial data transfer. The components of SIO0 and SIO1 function block are:

- 8-bit control register (SIO0CON, SIO1CON)
- Clock selector logic
- 8-bit data buffer (SIO0DATA, SIO1DATA)
- 8-bit prescaler (SIO0PS, SIO1PS)
- 3-bit serial clock counter
- Serial data I/O pins (SI0, SO0, SI1, SO1)
- Serial clock input/output pins (SCK0, SCK1)
- Serial data and clock output type selection (PG2CON.7–.6)

The SIO modules 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 (SCK0/SI0/SO0, SCK1/SI1/SO1) by loading the appropriate value to the P3CONH and P3CONL register if necessary.
- 2. Configure the output type (SCK0/SO0, SCK1/SO1) by manipulating PG2CON.7-.6 if necessary.
- Load an 8-bit value to the SIO0CON and SIO1CON control registers to properly configure the serial I/O modules. In this operation, SIO0CON.2 and SIO1CON.2 must be set to "1" to enable the data shifters, respectively.
- 4. For interrupt generation, set the serial I/O interrupt enable bits (SIO0CON.1, SIO1CON.1) to "1", respectively.
- 5. When you transmit data to the serial buffer, write data to SIO0DATA or SIO1DATA and set SIO0CON.3 or SIO1CON.3 to 1, the shift operation starts.
- 6. When the shift operation (transmit/receive) is completed, the SIO0 and SIO1 pending bits (SIO0CON.0 and SIO1CON.0) are set to "1" and SIO interrupt requests are generated, respectively.



#### SIO0 AND SIO1 CONTROL REGISTERS (SIO0CON, SIO1CON)

The control registers for serial I/O interface modules, SIO0CON, is located at E9H and SIO1CON, is located at ECH in set 1, bank 0. They have the control settings for SIO modules, respectively.

- 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 SIO0CON and SIO1CON values to "00H". This configures the corresponding modules with an internal clock source at the SCK0 and SCK1, selects receive-only operating mode, and clears the 3-bit counter, respectively. The data shift operation and the interrupt are disabled. The selected data direction is MSB-first.

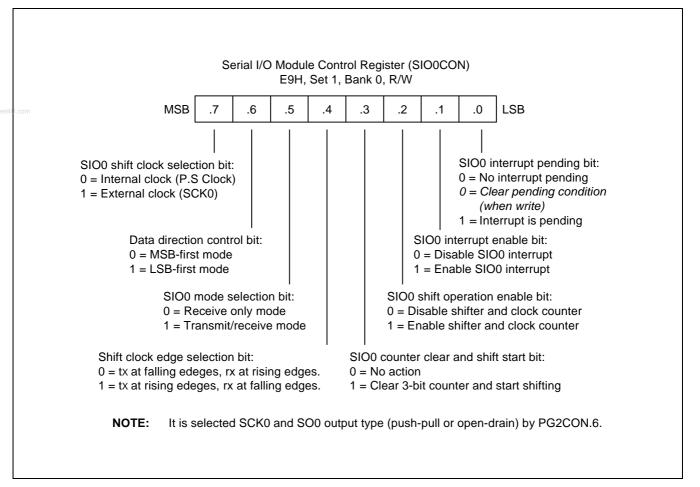


Figure 16-1. Serial I/O Module Control Register (SIO0CON)



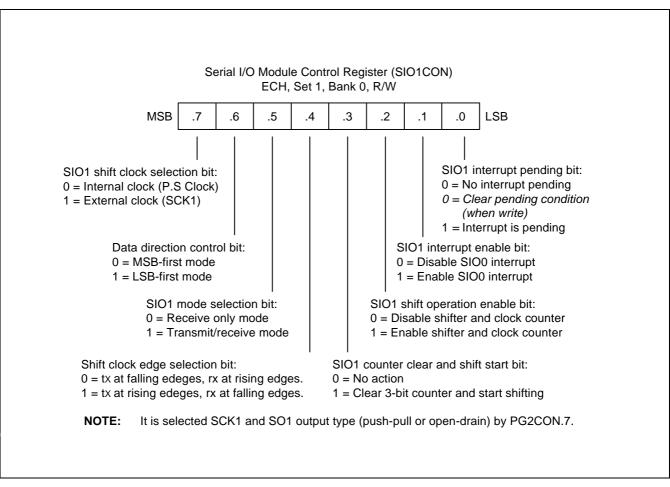


Figure 16-2. Serial I/O Module Control Register (SIO1CON)



#### SIO0 AND SIO1 PRE-SCALER REGISTER (SIO0PS, SIO1PS)

The prescaler registers for serial I/O interface modules, SIO0PS and SIO1PS, are located at EBH and EEH in set 1, bank 0, respectively.

The values stored in the SIO0 and SIO1 pre-scale registers, SIO0PS and SIO1PS, lets you determine the SIO0 and SIO1 clock rate (baud rate) as follows, respectively:

Baud rate = Input clock (fxx/4)/(Pre-scaler value + 1), or SCK0 and SCK1 input clock.

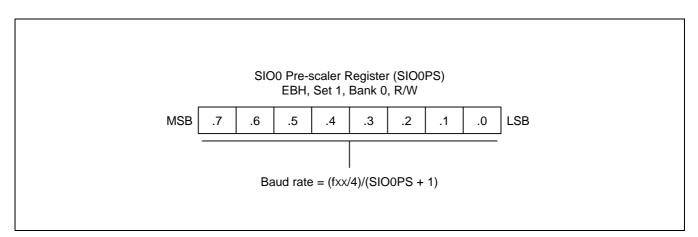


Figure 16-3. SIO0 Pre-scaler Register (SIO0PS)

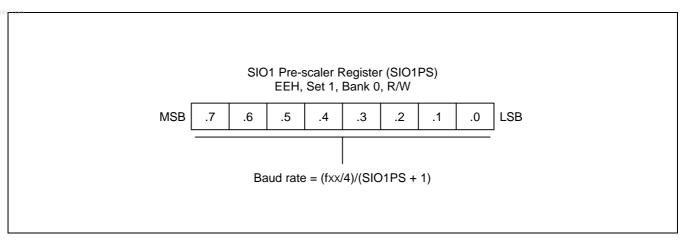


Figure 16-4. SIO1 Pre-scaler Register (SIO1PS)



## SIO0 BLOCK DIAGRAM

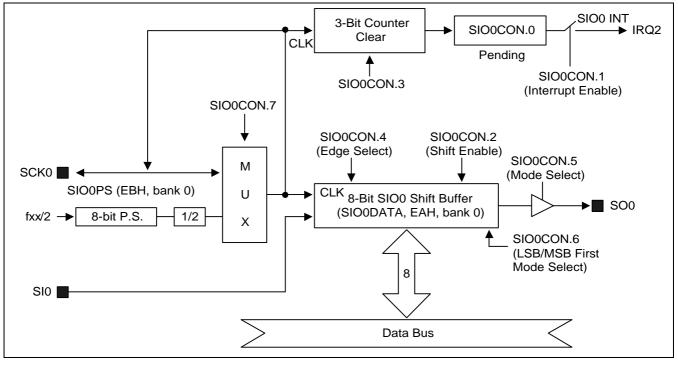


Figure 16-5. SIO0 Functional Block Diagram

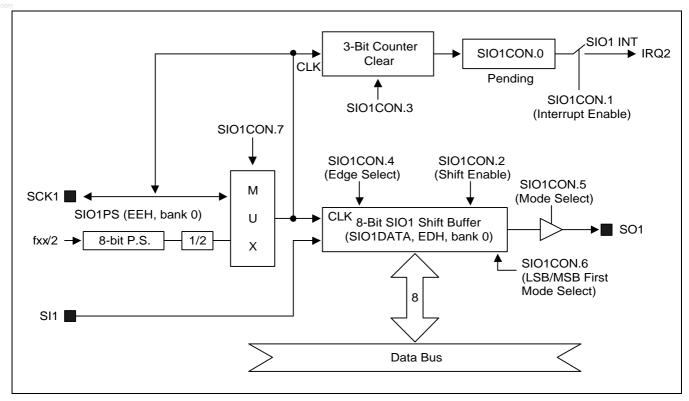


Figure 16-6. SIO1 Functional Block Diagram



#### SERIAL I/O TIMING DIAGRAM (SIO0, SIO1)

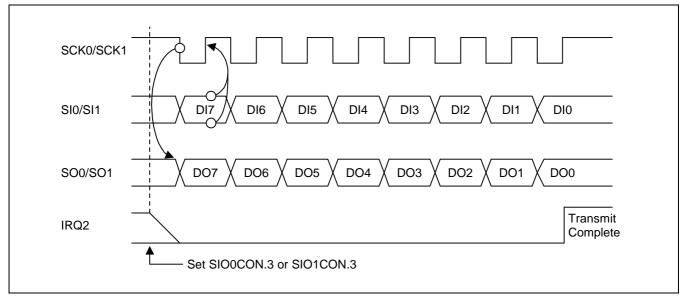


Figure 16-7. Serial I/O Timing in Transmit/Receive Mode (Tx at falling, SIO0CON.4 or SIO1CON.4 = 0)

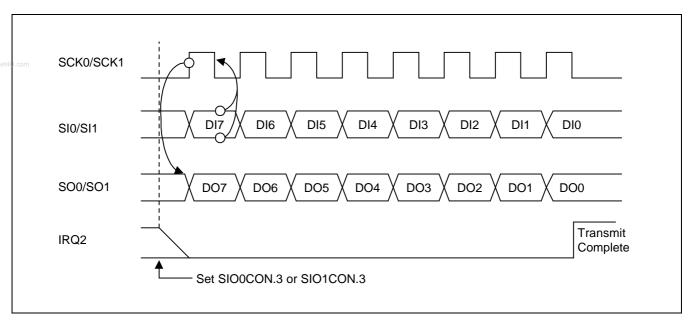


Figure 16-8. Serial I/O Timing in Transmit/Receive Mode (Tx at rising, SIO0CON.4 or SIO1CON.4 = 1)



## 17 LOW VOLTAGE RESET

## OVERVIEW

The low voltage reset block is useful for an system reset under the specific voltage of system. The components of LVR block are:

- LVREN pin
- LVRSEL pin
- Reference voltage generator
- Voltage divider
- Comparator
- Glitch filter

#### LVREN PIN

A LVREN pin is used to enable or disable LVR function.

The LVR function is disabled when the LVREN pin is connected to  $V_{SS}$  and is enabled when the LVREN pin is connected to  $V_{DD}$ .

#### LVRSEL PIN

A LVRSEL pin is used to select the criterion voltage of Low Voltage reset. The criterion voltage is typical 3.7V for LVR when the pin is connected to  $V_{SS}$  and is typical 2.4V for LVR when the pin is connected to  $V_{DD}$ .

#### **BLOCK DIAGRAM**

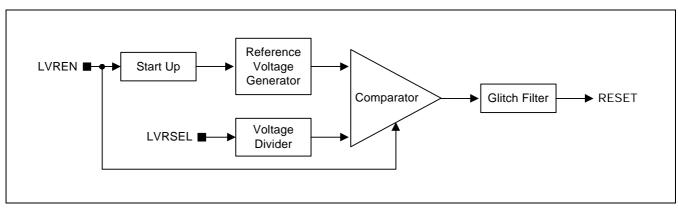


Figure 17-1. Low Voltage Reset Block Diagram



NOTES



# 18 PLL FREQUENCY SYNTHESIZER

## OVERVIEW

The phase locked loop (PLL) frequency synthesizer locks medium frequency (MF), high frequency (HF), and very high frequency (VHF) signals to a fixed frequency using a phase difference comparison system. As shown in Figure 18-1, the PLL frequency synthesizer consists of an input selection circuit, programmable divider, phase detector, reference frequency generator, and a charge pump.

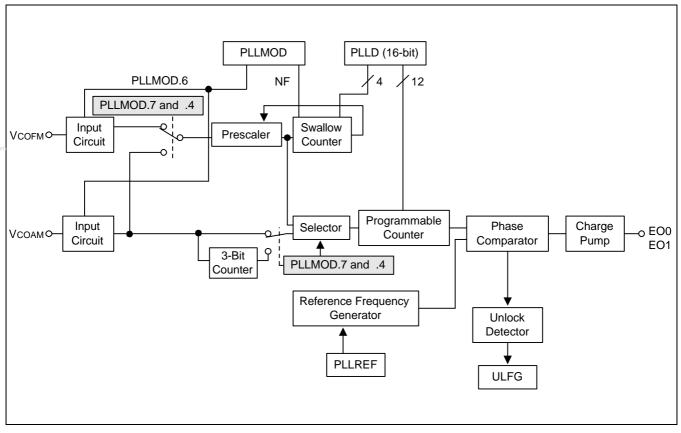


Figure 18-1. Block Diagram of the PLL Frequency Synthesizer



### PLL FREQUENCY SYNTHESIZER FUNCTION

The PLL frequency synthesizer divides the signal frequency at the  $V_{COAM}$  or  $V_{COFM}$  pin using the programmable divider. It then outputs the phase difference between the divided frequency and reference frequency at the EO0 and EO1 pin.

#### NOTE

The PLL frequency synthesizer operates only when the CE pin is High level. When the CE pin is Low level, the synthesizer is disable.

#### Input Selection Circuit

The input selection circuit consists of the  $V_{COAM}$  pin and  $V_{COFM}$  pins, an FM/AM selector, and two amplifiers. The input selection circuit selects the frequency division method and the input pin of the PLL frequency.

You can choose one of two frequency division methods using the PLL mode register: 1) direct frequency division method, or 2) pulse swallow method. The PLL mode register is also used to select the  $V_{COAM}$  or  $V_{COFM}$  pin as the frequency input pin.

#### **Programmable Divider**

The programmable divider divides the frequency of the signal from the  $V_{COAM}$  and  $V_{COFM}$  pins in accordance with the values contained in the swallow counter and programmable counter. The programmable divider consists of prescalers, a swallow counter, and a programmable counter.

When the PLL operation starts, the contents of the PLL data registers (PLLD0–PLLD1) and the NF bit in the PLLMOD register are automatically loaded into the 12-bit programmable counter and the 5-bit swallow counter.

When the 12-bit programmable down counter reaches zero, the contents of the data register are automatically reloaded into the programmable counter and the swallow counter for the next counting operation.

If you modify the data register value while the PLL is operating, the new values are not immediately loaded into the two counters; the new data are loaded into the two counters when the current count operation has been completed.

The contents of the data register undetermined after initial power-on. However, the data register retains its current value when the reset operation is initiated by an external reset or a change in level at the CE pin.

The swallow counter is a 5-bit binary down counter; the programmable counter is a 12-bit binary down counter. The swallow counter is for FM mode only. The swallow counter and programmable counter start counting down simultaneously. When the swallow counter starts counting down, the 1/33 prescaler is selected. When the swallow counter reaches zero, it stop operation and selects the 1/32 prescaler.



## PLL DATA REGISTER (PLLD)

The frequency division value of the swallow counter and programmable counter is set in the PLL data register (PLLD0-PLLD1). PLL data register configuration is shown in Figure 18-2.

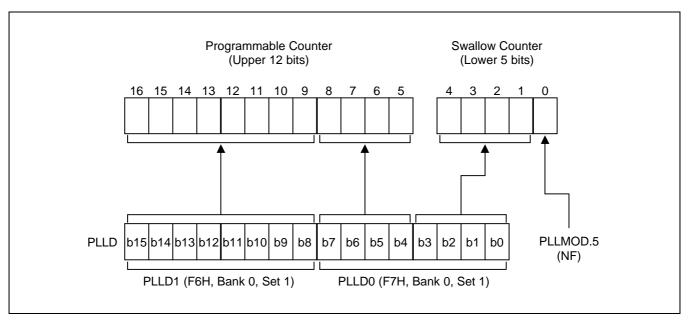


Figure 18-2. PLL Register Configuration

#### Direct Frequency Division and Pulse Swallow Formulas

In the direct frequency division method, the upper 12 bits are valid. In the pulse swallow method, all 16 bits are valid. The upper 12 bit are set in the programmable counter and the lower 4 bits and the NF bit are set in the swallow counter. The frequency division formulas for both methods, as set in the PLL data register, are shown below:

- Direct frequency division (AM) is

$$f_{R} = \frac{fV_{COAM}}{N}$$
 (When PLLMOD.7 and PLLMOD.4 are set to logic "00".)  
$$f_{R} = \frac{fV_{COAM}}{8xN}$$
 (When PLLMOD.7 and PLLMOD.4 are set to logic "01".)

Where the frequency division value (N) is 12 bits;  $fV_{COAM}$  = input frequency at the V<sub>COAM</sub> pin

- Pulse swallow system is

(AM) 
$$f_R = \frac{fV_{COAM}}{(N\times32+M)}$$
 (When PLLMOD.7 and PLLMOD.4 are set to logic "10".)  
(FM)  $f_R = \frac{fV_{COFM}}{(N\times32+M)}$  (When PLLMOD.7 and PLLMOD.4 are set to logic "11".)

where the frequency division values (N and M) are 12 bits and 5 bits, respectively;  $fV_{COFM}$  = input frequency at the  $V_{COFM}$  pin.



## **REFERENCE FREQUENCY GENERATOR**

The reference frequency generator produce reference frequency which are then compared by the phase comparator. As shown in Figure 18-3, the reference frequency generator divides a crystal oscillation frequency of 4.5 MHz and generates the reference frequency ( $f_R$ ) for the PLL frequency synthesizer. Using the PLLREF register, you can select from ten different reference frequencies.

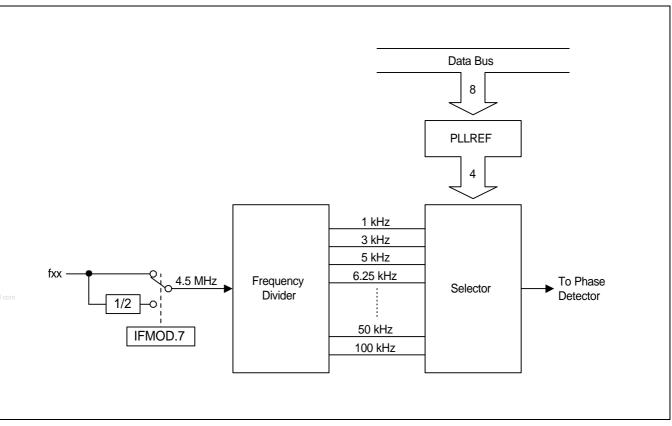


Figure 18-3. Reference Frequency Generator



## PLL MODE REGISTER (PLLMOD)

The PLL mode register (PLLMOD) is used to start and stop PLL operation and to enable or disable 3-bit counter for F<sub>VCOAM</sub>. PLLMOD values also determine the frequency dividing method.

PLLMOD PLLMOD.7 PLLMOD.6 N	IF PLLM	OD.4 PLLMOD.3	PLLMOD.2	PLLMOD.1	PLLMOD.0
----------------------------	---------	---------------	----------	----------	----------

PLLMOD.7 selects the frequency dividing method. The basic configuration for the two frequency dividing methods are as follows:

#### **Direct Method**

- Used for AM mode
- Swallow counter is not used
- V<sub>COAM</sub> pin is selected for input
- Selectable 3-bit counter (PLLMOD.7 and .4)

#### **Pulse Swallow Method**

- Used for AM, FM mode
- Swallow counter is used
- V<sub>COFM</sub> pin is selected for input

The input frequency at the V<sub>COAM</sub> or V<sub>COFM</sub> pin is divided by the programmable divider. The frequency division value of the programmable divider is written to the PLL data register.

When the pulse swallow method is selected by setting PLLMOD.7 and PLLMOD.4, the input signal is first divided by a 1/32 or 1/33 prescaler and the divided frequency is input to the programmable divider. Table 18-1 shows PLLMOD organization.



#### Table 18-1. PLLMOD Organization

#### PLL Enable and INTIF/INTCE Interrupt Control Bits

PLLMOD.6	0	Disable PLL.
	1	Enable PLL.
PLLMOD.3	0	Disable INTIF interrupt.
	1	Enable INTIF interrupt.
PLLMOD.2	0	INTIF interrupt is not pending (when read).; Clear INTIF pending bit (when write).
	1	INTIF interrupt is pending (when read).
PLLMOD.1	0	Disable INTCE interrupt requests at CE pin.
	1	Enable INTCE interrupt requests at CE pin.
PLLMOD.0	0	INTCE interrupt is not pending (when read).; Clear INTCE pending bit (when write).
	1	INTCE interrupt is pending (when read).

#### **Frequency Division Method Selection Bit**

	PLLMOD.7 and PLLMOD.4		Frequency Division Method	Selected Pin	Input Voltage	Input Frequency	Division Value	
t4U	com	0	Direct method for AM	$V_{COAM}$ selected; $V_{COFM}$ pulled Low	300mV <sub>PP</sub>	0.5–30 MHz	16 to (2 <sup>16</sup> –1)	
	0	1	Enable 3-bit counter for AM	$V_{COAM}$ selected; $V_{COFM}$ pulled Low	300mV <sub>PP</sub>	0.5–30 MHz	16 to (2 <sup>16</sup> –2)	
	1	0	Pulse swallow method for AM	$V_{COAM}$ selected; $V_{COFM}$ pulled Low	300mV <sub>PP</sub>	0.5–30 MHz	2 <sup>10</sup> to (2 <sup>18</sup> –2)	
	1	1	Pulse swallow method for FM	$V_{COFM}$ selected; $V_{COAM}$ pulled Low	300mV <sub>PP</sub>	30–150MHz	2 <sup>10</sup> to (2 <sup>18</sup> –2)	

**NOTE:** The NF bit, a one-bit frequency division value, is written to bit 0 in the swallow counter.



## PLL REFERENCE FREQUENCY SELECTION REGISTER (PLLREF)

The PLL reference frequency selection register (PLLREF) used to determine the reference frequency. You can select one of ten reference frequencies by setting bits PLLREF.3-PLLREF.0 to the appropriate value.

PLLREF	PLLREF.7	PLLREF.6	PLLREF.5	PLLREF.4	PLLREF.3	PLLREF.2	PLLREF.1	PLLREF.0
--------	----------	----------	----------	----------	----------	----------	----------	----------

You can select one of the reference frequencies by setting bits PLLREF.3-PLLREF.0.

PLLREF.3	PLLREF.2	PLLREF.1	PLLREF.0	Reference Frequency Selection
0	0	0	0	Select 1 kHz as reference frequency
0	0	0	1	Select 3 kHz as reference frequency
0	0	1	0	Select 5 kHz as reference frequency
0	0	1	1	Select 6.25 kHz as reference frequency
0	1	0	0	Select 9 kHz as reference frequency
0	1	0	1	Select 10 kHz as reference frequency
0	1	1	0	Select 12.5 kHz as reference frequency
0	1	1	1	Select 25 kHz as reference frequency
1	0	0	0	Select 50 kHz as reference frequency
1	0	0	1	Select 100 kHz as reference frequency

Table 18-2. PLLREF Register Organization (When fxx = 4.5 MHz)

NOTE: If the main clock is 9MHz, IFMOD.7 should be set to "1".



## PHASE DETECTOR, CHARGE PUMP, AND UNLOCK DETECTOR

The phase comparator compare the phase difference between divided frequency ( $f_N$ ) output from the programmable divider and the reference frequency ( $f_R$ ) output from the reference frequency generator.

The charge pump outputs the phase comparator's output from error output pins EO0 and EO1. The relation between the error output pin, divided frequency  $f_N$ , and reference frequency  $f_R$  is shown below:

 $f_R > f_N =$  Low level output

 $f_R < f_N =$  High level output

 $f_R = f_N =$  Floating level

A PLL operation starts when a value is loaded to the PLLMOD register, The PLL unlock flag (ULFG) in the PLL reference register, PLLREF, provides status information regarding the reference frequency and divided frequency.

The unlock detector detects the unlock state of the PLL frequency synthesizer. The unlock flag in the PLLREF register is set to "1" in an unlock state. When ULFG = "0", the PLL locked state is selected.

PLLREF.74	ULFG	CEFG	IFCFG	POFG	F9H at bank 0 of set 1
					-

The ULFG flag is set continuously at a period of reference frequency  $f_R$  by the unlock detector. You must therefore read the ULFG flag in the PLLREF register at periods longer than  $1/f_R$  of the reference frequency. ULFG is reset wherever it is read.

PLL operation is controlled by the state of the CE (chip enable) pin. The PLL frequency synthesizer is disabled and the error output pin is set to floating state whenever the CE pin is Low. When CE pin is High level, the PLL operates normally.

The chip enable flag in the PLLREF register, CEFG, provides the status of the current level of the CE pin. Whenever the state of the CE pin goes from Low to High, the CEFG flag is set to "1" and a CE reset operation occurs. When the CE pin goes from High to Low, the CEFG flag is cleared to "0" and a CE interrupt is generated.

The power on flag in the PLLREF register, POFG, is set by initiated power-on reset, but it is not set when a reset occurs on the normal operation. The POFG flag is cleared to "0" by writing "0" to POFG flag bit in PLLREF.



## USING THE PLL FREQUENCY SYNTHESIZER

This section describes the steps you should follow when using the PLL direct frequency division method and the pulse swallow method. In each case, you must make the following selections in this order:

- 1. Frequency division method: Direct frequency division and enable 3-bit counter (AM) or pulse swallow (AM, FM)
- 2. Input pin: VCOAM or VCOFM
- 3. Reference frequency: f<sub>R</sub>
- 4. Frequency division value: N

#### **Direct Frequency Division Method**

Select the direct frequency division method by writing a "0" to PLLMOD.7 and PLLMOD.4.

The VCOAM pin is configured for input when you select the direct frequency division method.

Select the reference frequency by writing the appropriate values to the PLLREF register.

The frequency division value is

$$N = \frac{fV_{COAM}}{f_R}$$
 (When PLLMOD.7 and PLLMOD.4 = 00),  $N = \frac{fV_{COAM}}{8 x f_R}$  (When PLLMOD.7 and PLLMOD.4 = 01)

where  $fV_{COAM}$  is the input frequency at the  $V_{COAM}$  pin, and  $f_R$  is the reference frequency.

#### Example (When PLLMOD.7 and PLLMOD.4 = 00):

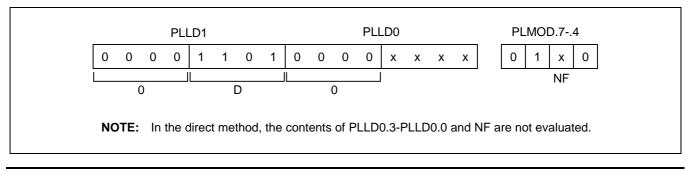
The following data are used to receive an AM-band broadcasting station:

Receive frequency:	1422 kHz
Reference frequency:	9 kHz
Intermediate frequency:	+ 450 kHz

The frequency division value N is calculated as follows:

$$N = \frac{fV_{COAM}}{f_{R}} = \frac{(1422+450)\times10^{3}}{9\times10^{3}} = 208 \text{ (decimal)}$$
  
= 0D0H (hexadecimal)

You would modify the PLL data register and PLLMOD.7-.4 register as follows:





#### **Pulse Swallow Method**

- 1. Select the pulse swallow method by writing a "1" to PLLMOD.7 and PLLMOD.4.
- 2. The VCOFM pin is configured for input when you select the pulse swallow method.
- 3. Select the reference frequency by writing the appropriate value to the PLLREF register.
- 4. Calculate the frequency division value as follows:

$$32N + M = \frac{fV_{COFM}}{f_R} \text{ (When PLLMOD.7 and PLLMOD.4 = 11),}$$
$$32N + M = \frac{fV_{COAM}}{f_R} \text{ (When PLLMOD.7 and PLLMOD.4 = 10)}$$

where  $fV_{COFM}$  is the input frequency at the  $V_{COFM}$  pin, and  $f_R$  is the reference frequency, N is the quotient of  $\frac{fV_{COFM}}{32f_R}$  and M is the remainder of  $\frac{fV_{COFM}}{32f_R}$ .

#### Example (When PLLMOD.7 and PLLMOD.4 = 11):

The following data are used to receive an FM-band broadcasting station:

Receive frequency:	100.0 MHz
Reference frequency:	25 kHz
Intermediate frequency:	10.7 MHz

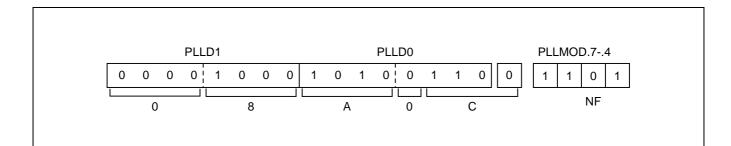
The frequency division value N and M are calculated as follows:

 $\frac{\text{fV}_{\text{COFM}}}{\text{f}_{\text{R}}} = \frac{(100.0 + 10.7) \times 10^6}{25 \times 10^3} = 4428 = 138 \times 32 + 12$ 

N = 138 (decimal) = 8AH (hexadecimal)

M = 12 (decimal) = 0C (hexadecimal)

You would modify the PLL data register and PLLMOD.7-.4 register as follows:





## **19** INTERMEDIATE FREQUENCY COUNTER

## **OVERVIEW**

The S3C831B uses an intermediate frequency counter (IFC) to counter the frequency of the AM or FM signal at FMIF or AMIF pin. The IFC block consists of a 1/2 divider, gate control circuit, IFC mode register (IFMOD) and a 16-bit binary counter. The gate control circuit, which controls the frequency counting time, is programmed using the IFMOD register. Four different gate times can be selected using IFMOD register settings.

During gate time, the 16-bit IFC counts the input frequency at the FMIF or AMIF pins. The FMIF or AMIF pin input signal for the 16-bit counter is selected using IFMOD register settings.

The 16-bit binary counter (IFCNT1-IFCNT0) can be read by 8-bit register addressing mode only. When the FMIF pin input signal is selected, the signal is divided by two. When the AMIF pin input signal is directly connected to the IFC, it is not divided.

By setting IFMOD register, the gate is opened for 2-ms, 8-ms, or 16-ms periods. During the open period of the gate, input frequency is counted by the 16-bit counter. When the gate is closed, the counting operation is complete, and an interrupt is generated.

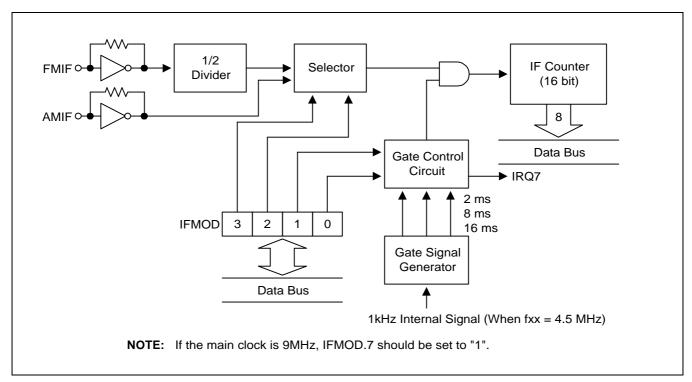


Figure 19-1. IF Counter Block Diagram



## IFC MODE REGISTER (IFMOD)

The IFC mode register (IFMOD) is a 8-bit register that is used to select the input pin, divider for  $V_{COFM}$  input frequency, PLL/IPC operation voltage, clock divider for PLL/IFC/WT, and gate time. Setting IFMOD register reset IFC value and IFC gate flag value, and starts IFC operation.

IFMOD	IFMOD.7	IFMOD.6	IFMOD.5	IFMOD.4	IFMOD.3	IFMOD.2	IFMOD.1	IFMOD.0	F3H at bank 0 of set 1

IFC operation starts when you select AMIF or FMIF as the IFC input pin. A reset operation clears all IFMOD values to "0".

#### Table 19-1. IFMOD Organization

#### PLL Frequency Synthesizer, If Counter, Watch Timer Clock Control Bit

IFMOD.7	PLL, IFC, Watch Timer Clock Setting						
0	The fxx is not divided for PLL/IFC/WT clock.						
1	The fxx is divided by 2 for PLL/IFC/WT clock.						

#### The PLL/IFC Operation Voltage Selection Bit

IFMOD.4 Operation Voltage Selection for PLL/IFC							
0	Select the PLL/IFC operation voltage as 4.5V to 5,5V						
1	Select the PLL/IFC operation voltage as 2.5V to 3,5V						



Table 19-1. IFMOD Organization (Continued)

#### **Pin Selection Bits**

IFMOD.3	IFMOD.2	Effect of Control Setting					
0	0	IFC is disabled; FMIF/AMIF are pulled down and FMIF/AMIF's feed-back resistor are off.					
0	1	Enable IFC operation; AMIF pin is selected; FMIF is pulled down and FMIF's feed-back resistor is off.					
1	0	Enable IFC operation; FMIF is selected; AMIF is pulled down and AMIF's feed- back resistor is off.					
1	1	Enable IFC operation; Both AMIF and FMIF are selected.					

#### Gate Time Select Bits

IFMOD.1	IFMOD.0	Select Gate Time				
0	0	Gate time is 2 ms.				
0	1	Gate time is 8 ms.				
1	0	Gate time is 16 ms.				
1	1	Gate is open				

## **IFC GATE FLAG REGISTER (PLLREF.5)**

PLLREF.74	ULFG	CEFG	IFCFG	POFG	F9H at bank 0 of set 1
-----------	------	------	-------	------	------------------------

When IFC operation is started by setting IFMOD, the IFC gate flag (IFCFG) is cleared to "0". After a specified gate time has elapsed, the IFCFG bit is automatically set to "1". This lets you check whether a IFC counting operation has been completed or not.

The IFC interrupt can also be used to check whether or not a IFC counting operation is complete.



## **GATE TIMES**

When you write a value to IFMOD, the IFC gate is opened for a 1-millisecond, 4-millisecond, or 8-millisecond interval, setting with a rising clock edge. When the gate is open, the frequency at the AMIF or FMIF pin is counted by the 16-bit counter. When the gate closes, the IFC gate flag (IFCFG) is set to "1". An interrupt is then generated and the IFC interrupt pending bit (PLLMOD.2) is set.

Figure 19-2 shows gate timings with a 1-kHz internal clock.

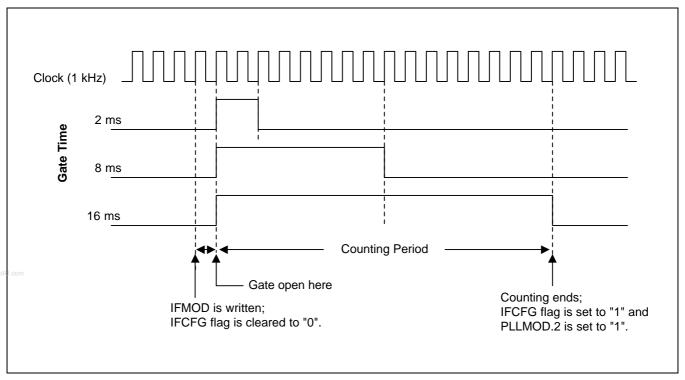


Figure 19-2. Gate Timing (2,8, or 16 ms)



#### Selecting "Gate Remains Open"

If you select "gate remain open" (IFMOD.0 and IFMOD.1 = "1"), the IFC counts the input signal during the open period of the gate. The gate closes the next time a value is written to IFMOD.

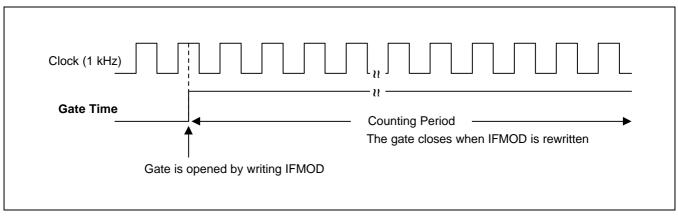
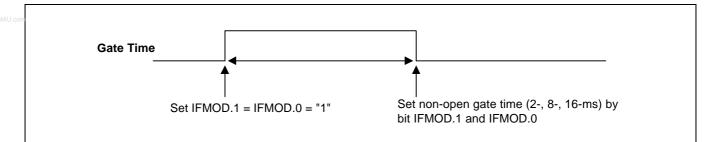


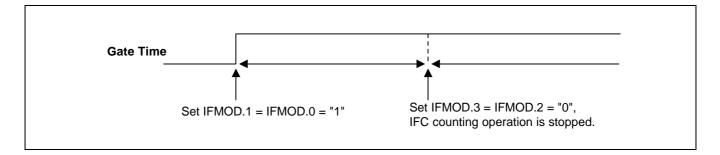
Figure 19-3. Gate Timing (When Open)

When you select "gate remains open" as the gating time, you can control the opening and closing of the gate in one of two ways:

— Set the gate time to a specific interval (2-ms, 8-ms, or 16-ms) by setting bits IFMOD.1 and IFMOD.0.



 Disable IFC operation by clearing bits IFMOD.3 and IFMOD.2 to "0". This method lets the gate remain open, and stops the counting operation.





#### **Gate Time Errors**

A gate time error occurs whenever the gate signals are not synchronized to the interval instruction clock. That is, the IFC does not start counter operation until a rising edge of the gate signal is detected, even though the counter start instruction (setting bits IFMOD.3 and IFMOD.2) has been executed. Therefore, there is a maximum 1-ms timing error (see Figure 19-4).

After you have executed the IFC start instruction, you can check the gate state at any time. Please note, however that the IFC does not actually start its counting operation until stabilization time for the gate control signal has elapsed.

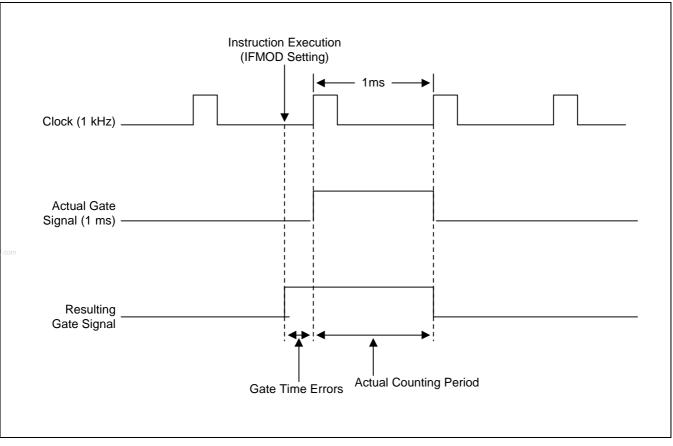


Figure 19-4. Gate Timing (1-ms Error)

#### **Counting Errors**

The IF counter counts the rising edges of the input signal in order to determine the frequency. If the input signal is High level when the gate is open, one additional pulse is counted. When the gate is close, however, counting is not affected by the input signal status. In other words, the counting error is "+1, 0".



## IF COUNTER (IFC) OPERATION

IFMOD register bits 2 and 3 are used to select the input pin and to start or stop IFC counting operation. You stop the counting operation by clearing IFMOD.2 and IFMOD.3 to "0". The IFC retains its previous value until IFMOD register values are specified.

Setting bits IFMOD.3 and IFMOD.2 starts the frequency counting operation. Counting continues as long as the gate is open. The 16-bit counter value is automatically cleared to 0000H after it overflows (at FFFFH), and continues counting from zero. The 16-bit count value (IFCNT1-IFCNT0) can be read by register addressing mode. A reset operation clears the counter to zero.

IFCNT0	IFCNT0.7	IFCNT0.6	IFCNT0.5	IFCNT0.4	IFCNT0.3	IFCNT0.2	IFCNT0.1	IFCNT0.0
IFCNT1	IFCNT1.7	IFCNT1.6	IFCNT1.5	IFCNT1.4	IFCNT1.3	IFCNT1.2	IFCNT1.1	IFCNT1.0

When the specified gate open time has elapsed, the gate closes in order to complete the counter operation. At this time, the IFC interrupt pending bit (PLLMOD.2) is automatically set to "1" and an interrupt is generated. The pending bit must be cleared to "0" by software when the interrupt is serviced. The IFC gate flag (IFCFG) is set to "1" at the same time the gate is closed. Since the IFCFG flag is cleared to "0" when IFC operation start, you can check the IFCFG flag to determine when IFC operation stops (that is, when the specified gate open time has elapsed).

The frequency applied to FMIF or AMIF pin is counted while the gate is open. The frequency applied to FMIF pin is divided by 2 before counting. The relationship between the count value (N) and input frequencies  $f_{AMIF}$  and  $f_{FMIF}$  is shown below.

#### - FMIF pin input frequency is

$$fFMIF = \frac{N(DEC)}{TG} \times 2$$

when TG = gate time (2 ms, 8 ms, 16 ms)

— AMIF pin input frequency is

$$f_{AMIF} = \frac{N (DEC)}{T_G}$$

when TG = gate time (2 ms, 8 ms, 16 ms)

Table 19-2 shows the range of frequency that you can apply to the AMIF and FMIF pins.

#### **Table 19-2. IF Counter Frequency Characteristics**

Pin	Voltage Level	Frequency Range
AMIF	300 m V <sub>PP</sub> (min)	0.1 MHz to 1 MHz
FMIF	300 m V <sub>PP</sub> (min)	5 MHz to 15 MHz



### INPUT PIN CONFIGURATION

The AMIF and FMIF pins have built-in AC amplifiers (see Figure 19-5). The DC component of the input signal must be stripped off by the external capacitor.

When the AMIF or FMIF pin is selected for the IFC function and the switch is turned on voltage of each pin increases to approximately 1/2  $V_{DD}$  after a sufficiently long time. If the pin voltage does not increase to approximately 1/2  $V_{DD}$ , the AC amplifier exceeds its operating range, possibly causing an IFC malfunction. To prevent this from occurring, you should program a sufficiently long time delay interval before starting the count operation.

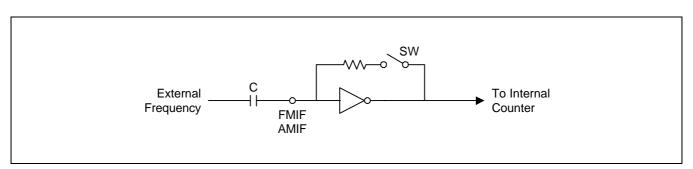


Figure 19-5. AMIF and FMIF Pin Configuration



## **IFC DATA CALCULATION**

## Selecting the FMIF pin for IFC Input

First, divide the signal at the FMIF pin by 2, and then apply this value to the IF counter. This means that the IF counter value is equal to one-half of the input signal frequency.

FMIF input frequency ( $f_{FMIF}$ ): 10.7 MHz Gate time ( $T_G$ ): 8 ms IFC counter value (N):

Bin	1	0	1	0	0	1	1	1	0	0	1	1	0	0	0	0
Dec		A	A			-	7			3	3		0			
IFCNT				IFC	NT1				IFCNT0							

## Selecting the AMIF Pin for IFC Input

The signal at AMIF pin is directly input to the IF counter.

AMIF input frequency ( $f_{AMIF}$ ): 450 kHz Gate time ( $T_G$ ): 8 ms IFC counter value (N):

Bin	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0
Dec		(	)			E	Ξ			1			0			
IFCNT				IFC	NT1				IFCNT0							



NOTES



# 20 ELECTRICAL DATA

## OVERVIEW

In this chapter, S3C831B electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

- Absolute maximum ratings
- D.C. electrical characteristics
- A.C. electrical characteristics
- Input/output capacitance
- Data retention supply voltage in stop mode
- A/D converter electrical characteristics
- PLL electrical characteristics
- Low voltage reset electrical characteristics
- Serial I/O timing characteristics
- Oscillation characteristics
- Oscillation stabilization time



Table 20-1. Absolute	Maximum Ratings
----------------------	-----------------

$(T_A =$	25	°C)
----------	----	-----

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V <sub>DD</sub>	_	-0.3 to +6.5	V
Input voltage	VI	Ports 0–8	-0.3 to V <sub>DD</sub> + 0.3	
Output voltage	V <sub>O</sub>	_	-0.3 to V <sub>DD</sub> + 0.3	
Output current high	I <sub>ОН</sub>	One I/O pin active	– 15	mA
		All I/O pins active	- 60	
Output current low	I <sub>OL</sub>	One I/O pin active	+ 30	
		Total pin current for port	+ 100	
Operating temperature	Τ <sub>Α</sub>		- 25 to + 85	°C
Storage temperature	T <sub>STG</sub>		– 65 to + 150	

#### Table 20-2. D.C. Electrical Characteristics

 $(T_A = -25 \degree C \text{ to } + 85 \degree C, V_{DD} = 2.2 \text{ V} \text{ to } 5.5 \text{ V})$ 

Parameter	Symbol	Conditio	ns	Min.	Тур.	Max.	Unit
Operating	V <sub>DD</sub>	fx = 0.4–4.5 MHz		2.2	-	5.5	V
Voltage		fx = 4.5–9 MHz		4.0	_	5.5	
		PLL, IFC operating	IFMOD.4 = 0	4.5	_	5.5	
			IFMOD.4 = 1	2.5	-	3.5	
Input High	V <sub>IH1</sub>	P1.4–P1.7, Ports 3, 4,	5	0.7 V <sub>DD</sub>		V <sub>DD</sub>	V
Voltage	V <sub>IH2</sub>	P1.0-P1.3, Ports 0, 2,	6, 7, 8	0.8 V <sub>DD</sub>	-	V <sub>DD</sub>	
	V <sub>IH3</sub>	RESET, CE		0.8 V <sub>DD</sub>		V <sub>DD</sub>	
	V <sub>IH4</sub>	X <sub>IN</sub> , X <sub>OUT</sub>		V <sub>DD</sub> -0.1	).1 V <sub>DD</sub>		
Input Low	V <sub>IL1</sub>	P1.4–P1.7, Ports 3, 4,			0.3 V <sub>DD</sub>	V	
Voltage	V <sub>IL2</sub>	P1.0–P1.3, Ports 0, 2, 6, 7, 8		_	-	0.2 V <sub>DD</sub>	
	V <sub>IL3</sub>	RESET, CE			0.2 V <sub>DD</sub>		
	V <sub>IL4</sub>	X <sub>IN</sub> , X <sub>OUT</sub>				0.1	
Output High	V <sub>OH1</sub>	$V_{DD} = 4.5 \text{ V to } 5.5 \text{ V}$		V <sub>DD</sub> – 2.0		V <sub>DD</sub>	V
Voltage		EO0, EO1; I <sub>OH</sub> = -1 m/	A				
	V <sub>OH2</sub>	$V_{DD} = 4.5 \text{ V to } 5.5 \text{ V}$		V <sub>DD</sub> – 1.0	_	$V_{DD}$	
		Other output ports; I <sub>OH</sub>	= –1 mA				
Output Low	V <sub>OL1</sub>	$V_{DD} = 4.5 V \text{ to } 5.5 V$		-		2.0	V
Voltage		EO0, EO1; I <sub>OL</sub> = 1 mA					
	V <sub>OL2</sub>	$V_{DD} = 4.5 \text{ V to } 5.5 \text{ V}$		-	-	2.0	
		Other output ports; $I_{OL}$	= 10 mA				



## Table 20-2. D.C. Electrical Characteristics (Continued)

(T<sub>A</sub> = -25 °C to + 85 °C, V<sub>DD</sub> = 2.2 V to 5.5 V)

Parameter	Symbol	Cone	ditions	Min.	Тур.	Max.	Unit
Input High Leakage Current	I <sub>LIH1</sub>	$V_{IN} = V_{DD}$ All input pins $X_{IN}, X_{OUT}$	except	-	_	3	uA
	I <sub>LIH2</sub>	$V_{IN} = V_{DD,} X_{I}$	<sub>N</sub> , X <sub>OUT</sub>			20	
Input Low Leakage Current	I <sub>LIL1</sub>	V <sub>IN</sub> = 0 V All input pins RESET, X <sub>IN</sub>	•	-	_	-3	
	I <sub>LIL2</sub>	V <sub>IN</sub> = 0 V, X <sub>II</sub>	<sub>N</sub> , X <sub>OUT</sub>			-20	
Output High Leakage Current	I <sub>LOH</sub>	V <sub>OUT</sub> = V <sub>DD</sub> All output pin	S	-	-	3	
Output Low Leakage Current	ILOL	V <sub>OUT</sub> = 0 V All output pin	S	-	-	-3	
Pull-Up Resistor	R <sub>L1</sub>	V <sub>IN</sub> = 0 V; Port 0–8	V <sub>DD</sub> = 5 V	25	47	100	kΩ
		T <sub>A</sub> = 25°C	$V_{DD} = 3 V$	50	95	200	
	R <sub>L2</sub>	V <sub>IN</sub> = 0 V RESET	V <sub>DD</sub> = 5 V	150	250	400	
		800					
Pull-Down Resistor	R <sub>D</sub>	$V_{IN} = 0 V$ $V_{DD} = 5 V$ RESET		15	30	45	kΩ
Oscillator Feed Back Resistors	R <sub>OSC</sub>			300	600	1500	kΩ
LCD Voltage Dividing Resistor	R <sub>LCD</sub>	T <sub>A</sub> = 25 °C		70	100	150	kΩ
$ V_{LCD} - COMi $ Voltage Drop (I = 0-3)	V <sub>DC</sub>	–15 μA per c	ommon pin	-	45	120	mV
$ V_{LCD} - SEGx $ Voltage Drop (x = 0-39)	V <sub>DS</sub>	–15 μA per c	–15 μA per common pin		45	120	mV
Middle Output	V <sub>LC0</sub>	V <sub>DD</sub> = 2.7 V	to 5.5 V	0.6V <sub>DD</sub> - 0.2	0.6V <sub>DD</sub>	0.6V <sub>DD</sub> + 0.2	V
Voltage	V <sub>LC1</sub>			0.4V <sub>DD</sub> - 0.2	0.4V <sub>DD</sub>	0.4V <sub>DD</sub> + 0.2	
	V <sub>LC2</sub>			0.2V <sub>DD</sub> -0.2	$0.2V_{DD}$	0.2V <sub>DD</sub> + 0.2	



## Table 20-2. D.C. Electrical Characteristics (Concluded)

(T<sub>A</sub> = -25 °C to + 85 °C, V<sub>DD</sub> = 2.2 V to 5.5 V)

Parameter	Symbol	Conditions		Min.	Тур.	Max.	Unit
Supply current <sup>(1)</sup>	I <sub>DD1</sub>	Run mode: 4.5 MHz crystal oscillator	9.0 MHz	-	12.0	25.0	mA
		$CE = V_{DD,} V_{DD} = 5 V \pm 10 \%$ Crystal Oscillator C1 = C2 = 22pF	4.5 MHz		6.5	15.0	
		V <sub>DD</sub> = 3 V ± 10 %	4.5 MHz		4.0	9.0	
	I <sub>DD2</sub>	Run mode: CE = 0 V, V <sub>DD</sub> = 5 V ± 10 %	9.0 MHz		5.0	12.0	
		Crystal Oscillator C1 = C2 = 22pF	4.5 MHz		2.5	5.5	
		V <sub>DD</sub> = 3 V ± 10 %	4.5 MHz		1.5	3.5	
	I <sub>DD3</sub>	Idle mode: CE = 0 V, V <sub>DD</sub> = 5 V ± 10 %	9.0 MHz		1.5	4.0	
		Crystal Oscillator C1 = C2 = 22pF	4.5 MHz		1.0	2.0	
		V <sub>DD</sub> = 3 V ± 10 %	4.5 MHz		0.5	1.2	
U.com	I <sub>DD4</sub> <sup>(2)</sup>	Stop mode (in LVR disable): CE = 0 V, T <sub>A</sub> = 25 °C, V <sub>DD</sub> = 5	V ± 10 %		0.5	3	μΑ
		V <sub>DD</sub> = 3 V ± 10 %			0.5	2	

### NOTES:

1. Supply current does not include current drawn through internal pull-up resistors, PWM, or external output current loads.

2. I<sub>DD4</sub> is current when the main clock oscillation stops.

3. Every values in this table is measured when bits 4–3 of the system clock control register (CLKCON.4–.3) is set to 11B.



 $(T_A = -25 \degree C \text{ to } +85 \degree C, V_{DD} = 2.2 \text{ V} \text{ to } 5.5 \text{ V})$ 

Parameter	Symbol	Conditions	Min	Тур	Max	Unit
Interrupt input high, low width (P1.0–P1.7)	tINTH, tINTL	P1.0–P1.7, V <sub>DD</sub> = 5 V	10	_		μs
RESET input low width	tRSL	$V_{DD} = 5 V$	10	_	-	us

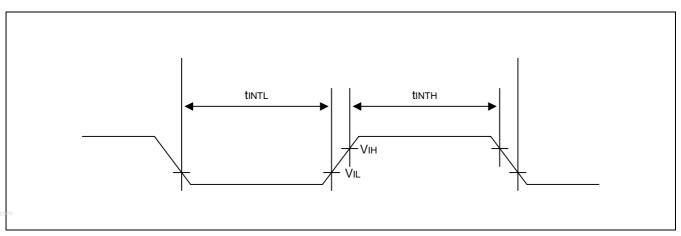


Figure 20-1. Input Timing for External Interrupts (Ports 1)

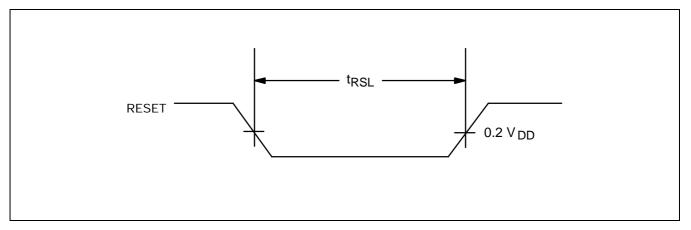


Figure 20-2. Input Timing for RESET



## Table 20-4. Input/Output Capacitance

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

Parameter	Symbol	Conditions	Min	Тур	Max	Unit
Input capacitance	C <sub>IN</sub>	f = 1 MHz; unmeasured pins are returned to $V_{SS}$	-	_	10	pF
Output capacitance	C <sub>OUT</sub>					
I/O capacitance	C <sub>IO</sub>					

## Table 20-5. Data Retention Supply Voltage in Stop Mode

 $(T_A = -25 \ ^{\circ}C \ to + 85 \ ^{\circ}C)$ 

Parameter	Symbol	Conditions	Min	Тур	Max	Unit
Data retention supply voltage	V <sub>DDDR</sub>		2.2	-	5.5	V
Data retention supply current	I <sub>DDDR</sub>	$V_{DDDR} = 2.2 V (TA = 25 °C)$ Stop mode (in LVR disable)	_	-	1	uA

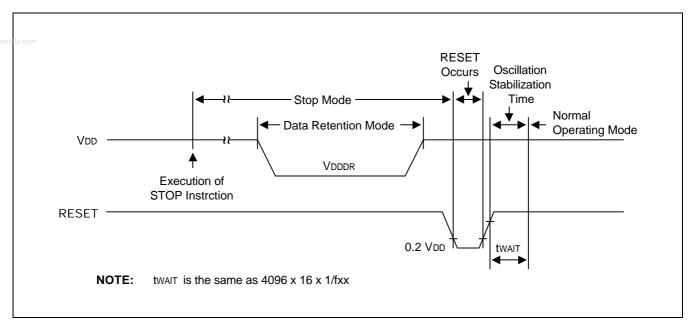


Figure 20-3. Stop Mode Release Timing Initiated by RESET



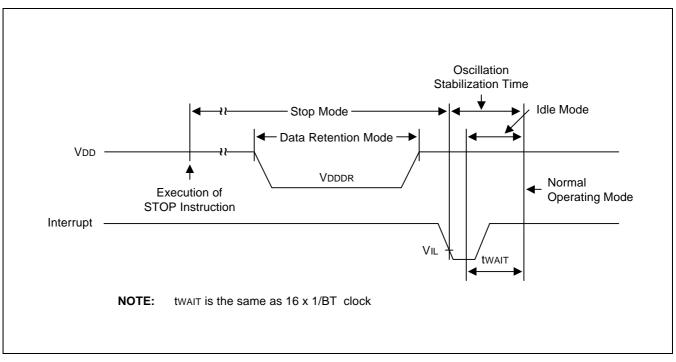


Figure 20-4. Stop Mode Release Timing Initiated by Interrupts

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

 $(T_A = -25 \text{ °C to } +85 \text{ °C}, V_{DD} = 2.7 \text{ V to } 5.5 \text{ V}, V_{SS} = 0 \text{ V})$ 

Parameter	Symbol	Conditions	Min	Тур	Max	Unit
A/D converting resolution	_	_	-	8	-	bits
Absolute accuracy	-	V <sub>DD</sub> = 5.12 V	-	-	±2	LSB
A/D conversion time (NOTE)	t <sub>CON</sub>	Conversion clock = fxx	50/fxx		-	μs
Analog input voltage	V <sub>IAN</sub>	-	V <sub>SS</sub>	-	V <sub>DD</sub>	V
Analog input impedance	R <sub>AN</sub>	$V_{DD} = 5 V$	2	1000	_	MΩ

**NOTE:** A/D Converter needs at least  $25\mu s$  for conversion time.



## Table 20-7. PLL Electrical Characteristics

$(T_A = -25 \ ^\circ C \text{ to } +85 \ ^\circ C, V_{DD} =$	2.5 V to 3.5 V, 4.5 V to 5.5 V)
--	---------------------------------

Parameter	Symbol	Conditions	Min	Тур	Max	Unit
VCOFM, VCOAM, FMIF and AMIF input voltage (peak to peak)	V <sub>IN</sub>	Sine wave input 0.3		Ι	V <sub>DD</sub>	V
Frequency	fV <sub>COAM</sub>	VCOAM mode, sine wave input; $V_{IN} = 0.3V_{P-P}$ 0.5		-	30	MHz
	fV <sub>COFM</sub>	VCOFM mode, sine wave input; $V_{IN} = 0.3V_{P-P}$			150	
7 (1911)		AMIF mode, sine wave input; V <sub>IN</sub> = 0.3V <sub>P-P</sub>	0.1		1.0	
	f <sub>FMIF</sub>	FMIF mode, sine wave input; V <sub>IN</sub> = 0.3V <sub>P-P</sub>	5		15	

## Table 20-8. Low Voltage Reset Electrical Characteristics

 $(T_A = -25 \ ^\circ C \text{ to } +85 \ ^\circ C)$ 

Parameter	Symbol	Conditions	Min	Тур	Max	Unit
Detect voltage range	$V_{DET}$	LVRSEL = V <sub>SS</sub>	3.2	3.7	4.2	V
		LVRSEL = V <sub>DD</sub>	2.2	2.4	2.7	V
LVR operating current	I <sub>LVR</sub>	_	_	10	25	μΑ



## Table 20-9. Synchronous SIO Electrical Characteristics

(T<sub>A</sub> = -25 °C to +85 °C, V<sub>DD</sub> = 2.2 V to 5.5 V)

Parameter	Symbol	Conditions Min		Тур	Max	Unit
SCK0/SCK1 cycle time	t <sub>CKY</sub>	External SCK0/SCK1 source	1000	-	-	ns
		Internal SCK0/SCK1 source	1000			
SCK0/SCK1 high, low	t <sub>KH</sub> , t <sub>KL</sub>	External SCK0/SCK1 source	500	-	-	
width		Internal SCK0/SCK1 source	t <sub>KCY</sub> /2–50			
SI setup time to	t <sub>SIK</sub>	External SCK0/SCK1 source	250	_	-	
SCK0/SCK1 high		Internal SCK0/SCK1 source	250			
SI hold time to	t <sub>KSI</sub>	External SCK0/SCK1 source	400	-	-	
SCK0/SCK1 high		Internal SCK0/SCK1 source	400	400		
Output delay for	t <sub>KSO</sub>	External SCK0/SCK1 source	-	_	300	
SCK0/SCK1 to SO		Internal SCK0/SCK1 source			250	

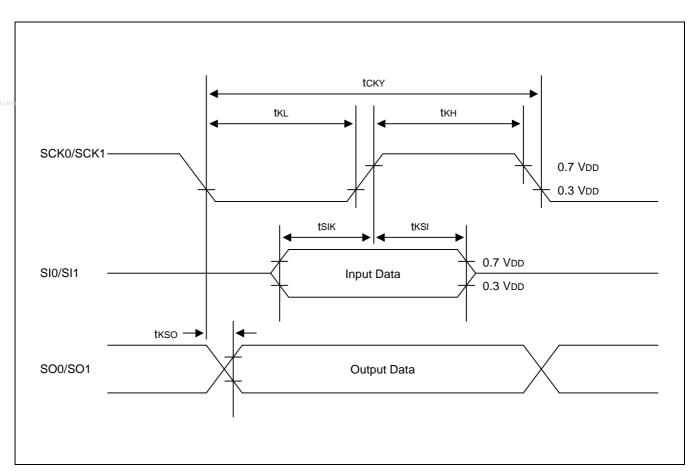


Figure 20-5. Serial Data Transfer Timing



## Table 20-10. Main Oscillator Characteristics (fx)

$(T_A = -$	25 °C to +85 °C, V	$V_{DD} = 2.2 \text{ V to}$	5.5 V)
------------	--------------------	-----------------------------	--------

Oscillator	Clock Circuit	Parameter	Test Condition	Min	Тур	Мах	Unit
Crystal	XIN XOUT	Crystal oscillation	$V_{DD} = 2.2V - 5.5V$	0.4	-	4.5	MHz
		frequency <sup>(1)</sup>	$V_{DD} = 4.0V - 5.5V$	0.4	-	9	
		Stablilization time <sup>(2)</sup>	Stabilization occurs when V <sub>DD</sub> is equal to the minimum oscillator voltage range.	-	_	40	ms
Ceramic	XIN XOUT	Crystal oscillation	V <sub>DD</sub> = 2.2V – 5.5V	0.4	_	4.5	MHz
	$\begin{array}{c} \bullet - \parallel \\ - \\ - \\ - \\ - \\ - \\ - \\ - \\ - \\ -$	frequency <sup>(1)</sup>	V <sub>DD</sub> = 4.0V - 5.5V	0.4	_	9	
	m m	Stablilization time <sup>(2)</sup>	_	_	Ι	10	ms
External	Xin Xout	X <sub>IN</sub> input frequency <sup>(1)</sup>	$V_{DD} = 2.2V - 5.5V$	0.4	-	4.5	MHz
Clock			$V_{DD} = 4.0V - 5.5V$	0.4	_	9	
	×	X <sub>IN</sub> input high and low	$V_{DD} = 2.2V - 5.5V$	110	_	1250	ns
U.com	$\overset{\frown}{\rightarrow}$	level width ( $t_{XH}$ , $t_{XL}$ )	V <sub>DD</sub> = 4.0V - 5.5V	55	-	1250	

#### NOTES:

1. Oscillation frequency and  $X_{IN}$  input frequency data are for oscillator characteristics only.

2. Stabilization time is the interval required for oscillating stabilization after a power-on occurs, or when stop mode is terminated.



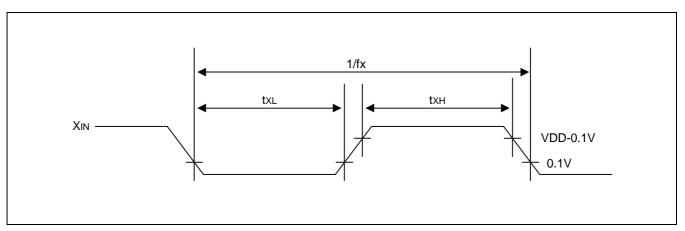


Figure 20-6. Clock Timing Measurement at X<sub>IN</sub>

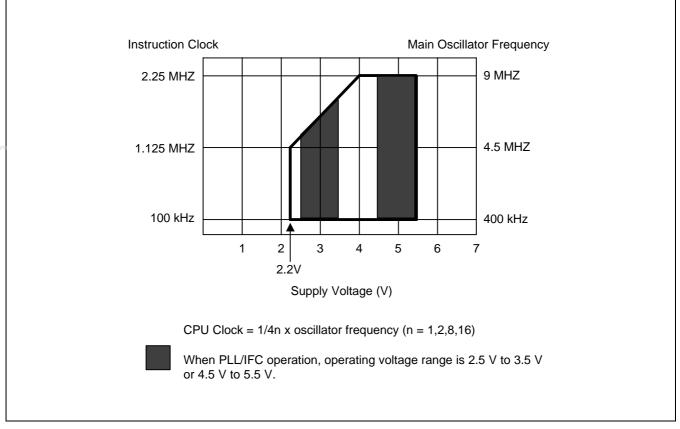


Figure 20-7. Operating Voltage Range



NOTES



## 21 MECHANICAL DATA

## **OVERVIEW**

The S3C831B microcontroller is currently available in 100-pin-QFP or 100-pin-TQFP package.

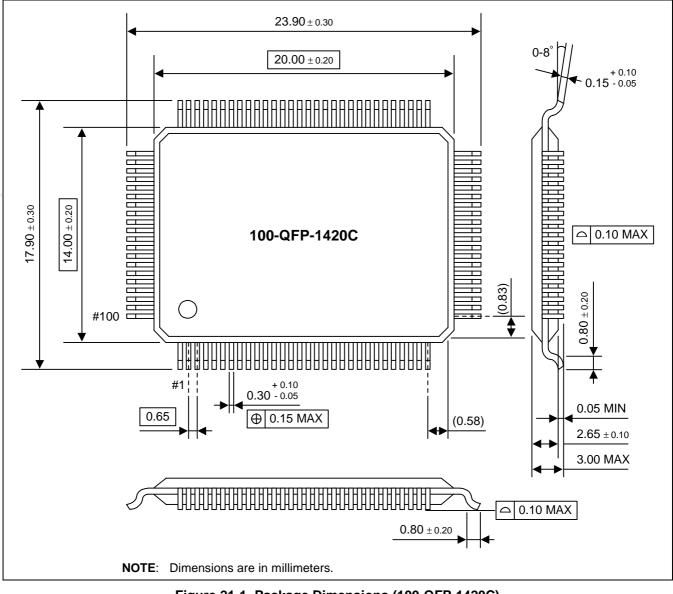


Figure 21-1. Package Dimensions (100-QFP-1420C)



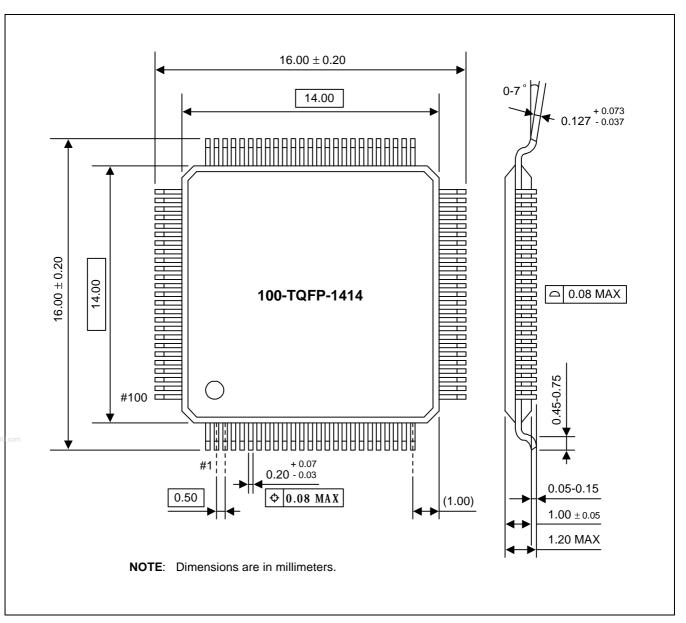


Figure 21-2. Package Dimensions (100-TQFP-1414)



## 22 S3P831B OTP

## **OVERVIEW**

The S3P831B single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the S3C831B microcontroller. It has an on-chip OTP ROM instead of a masked ROM. The EPROM is accessed by serial data format.

The S3P831B is fully compatible with the S3C831B, both in function in D.C. electrical characteristics and in pin configuration. Because of its simple programming requirements, the S3P831B is ideal as an evaluation chip for the S3C831B.



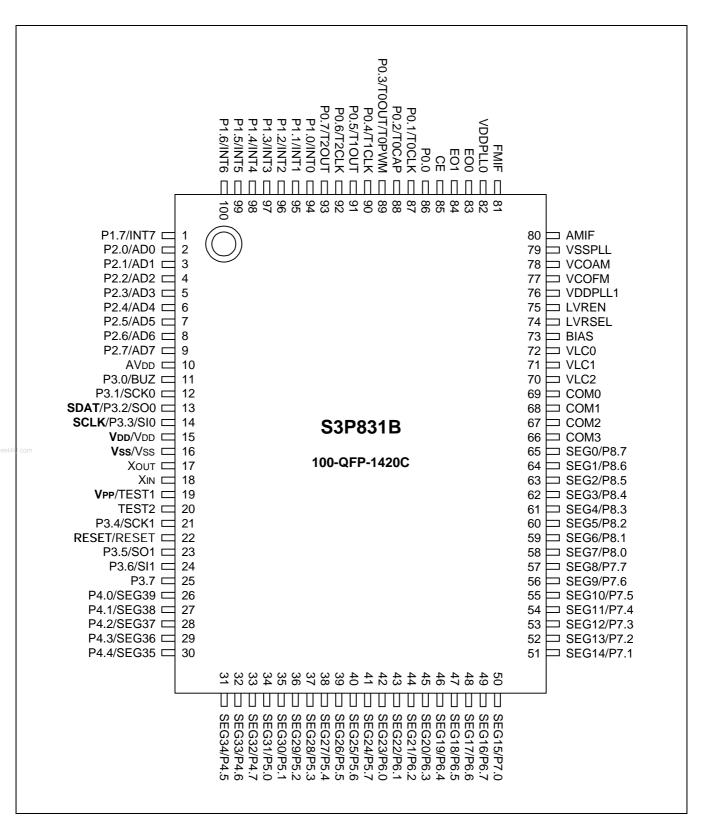


Figure 22-1. S3P831B Pin Assignments (100-Pin QFP Package)



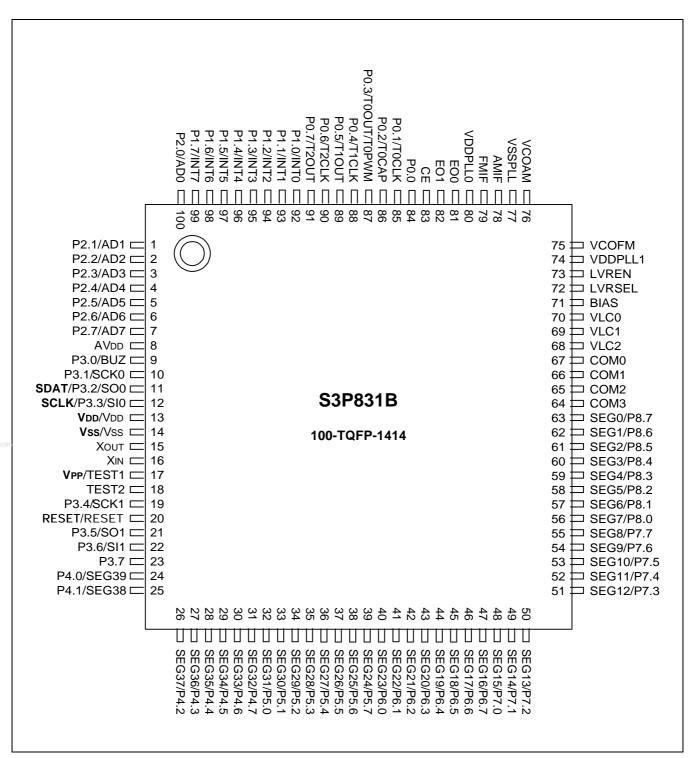


Figure 22-2. S3P831B Pin Assignments (100-Pin TQFP Package)



Main Chip	During Programming						
Pin Name	Pin Name	Pin No.	I/O	Function			
P3.2/SO0	SDAT	13(11)	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input/push-pull output port.			
P3.3/SI0	SCLK	14(12)	I	Serial clock pin. Input only pin.			
TEST1	V <sub>PP</sub>	19(17)	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is applied, OTP is in reading mode. (Option)			
RESET	RESET	22(20)	I	Chip Initialization			
V <sub>DD</sub> /V <sub>SS</sub>	V <sub>DD</sub> /V <sub>SS</sub>	15/16(13/14)	-	Logic power supply pin. VDD should be tied to +5 V during programming.			

Table 22-1. Descriptions of Pins Used to Read/Write the EPROM

**NOTE:** Parentheses indicate pin number for 100-TQFP-1414 package.

Table 22-2.	Comparison	of S3P831B	and S3C831B Features
-------------	------------	------------	----------------------

Characteristic	S3P831B	S3C831B
Program Memory	64-Kbyte EPROM	64-Kbyte mask ROM
Operating Voltage (V <sub>DD</sub> )	2.2 V to 5.5 V	2.2 V to 5.5 V
OTP Programming Mode	V <sub>DD</sub> = 5 V, V <sub>PP</sub> (TEST1) = 12.5 V	
Pin Configuration	100 QFP, 100 TQFP	100 QFP, 100 TQFP
EPROM Programmability	User Program 1 time	Programmed at the factory

## **OPERATING MODE CHARACTERISTICS**

When 12.5 V is supplied to the  $V_{PP}$  (TEST1) pin of the S3P831B, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 22-3 below.

VDD	VPP (TEST1)	REG/MEM	Address(A15–A0)	R/W	Mode
5 V	5 V	0	0000H	1	EPROM read
	12.5 V	0	0000H	0	EPROM program
	12.5 V	0	0000H	1	EPROM verify
	12.5 V	1	0E3FH	0	EPROM read protection

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



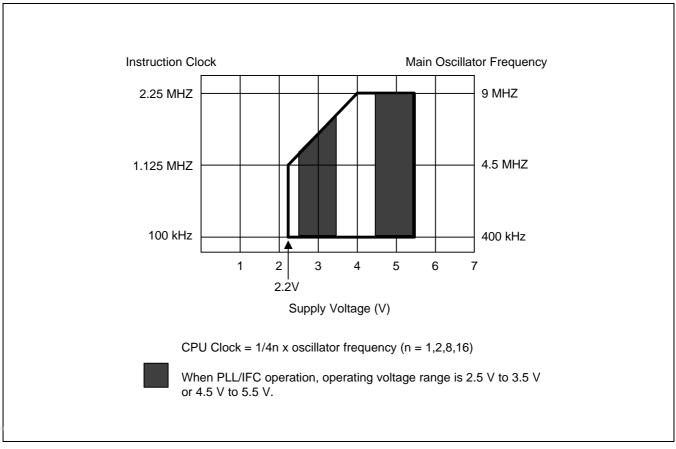


Figure 22-3. Operating Voltage Range



NOTES



## 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 as its operating system can be used. One type of debugging tool including hardware and software is provided: the sophisticated and powerful in-circuit emulator, SMDS2+, for S3C7, S3C9, S3C8 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

#### SHINE

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

#### SAMA ASSEMBLER

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

#### SASM88

The SASM88 is a relocatable assembler for Samsung's S3C8-series microcontrollers. The SASM88 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM88 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces the relocatable object code only, so the user should link object file. Object files can be linked with other object files and loaded into memory.

#### HEX2ROM

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

#### TARGET BOARDS

Target boards are available for all S3C8-series microcontrollers. All required target system cables and adapters are included with the device-specific target board.



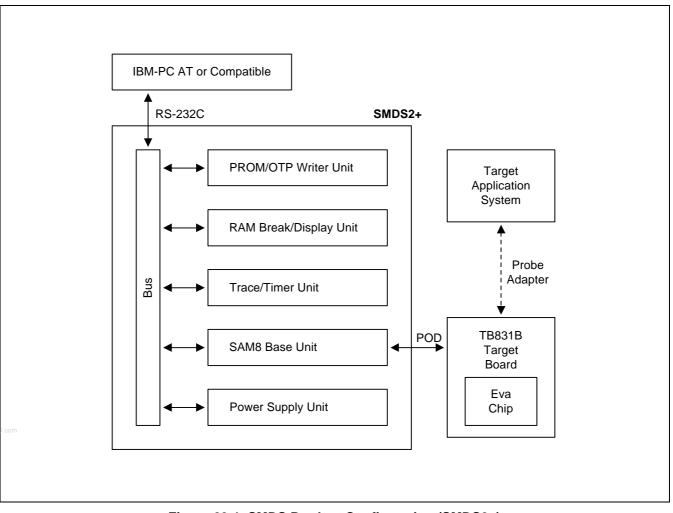


Figure 23-1. SMDS Product Configuration (SMDS2+)



## **TB831B TARGET BOARD**

The TB831B target board is used for the S3C831B/P831B microcontroller. It is supported with the SMDS2+, Smart Kit and OPENice.

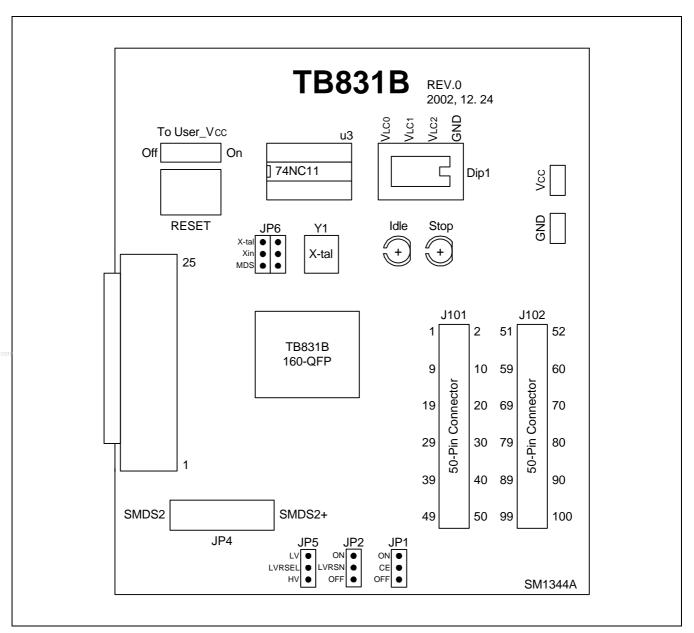


Figure 23-2. TB831B Target Board Configuration



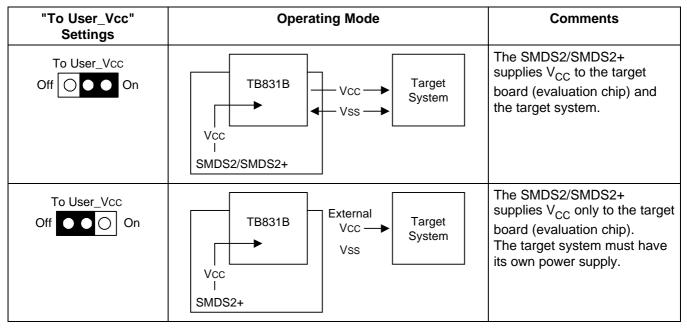


Table 23-1. Power Selection Settings for TB831B

**NOTE:** The following symbol in the "To User\_Vcc" Setting column indicates the electrical short (off) configuration:



## SMDS2+ SELECTION (SAM8)

In order to write data into program memory that is available in SMDS2+, the target board should be selected to be for SMDS2+ through a switch as follows. Otherwise, the program memory writing function is not available.

Table 23-2. The SMDS2+ Tool Selection Setting

"SW1" Setting	Operating Mode
SMDS2 SMDS2+	R/W ◀ R/W ► Target SMDS2+

#### IDLE LED

The Yellow LED is ON when the evaluation chip (S3E8310) is in idle mode.

## STOP LED

The Red LED is ON when the evaluation chip (S3E8310) is in stop mode.



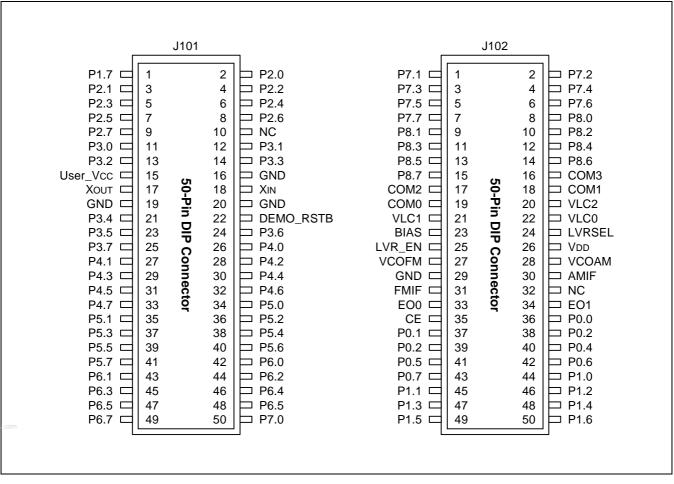


Figure 23-3. 50-Pin Connectors (J101, J102) for TB831B

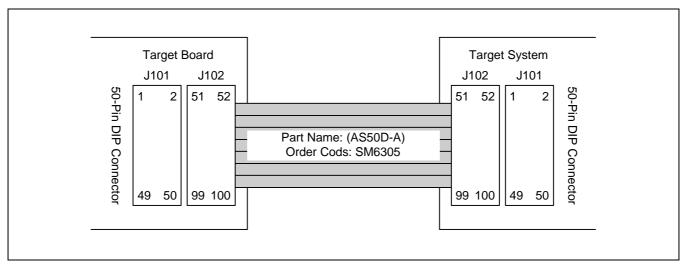


Figure 23-4. S3C831B/P831B Probe Adapter Cables for 100-QFP Package

