1 PRODUCT OVERVIEW

S3C-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. Important CPU features include:

— 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 six CPU clocks) can be assigned to specific interrupt levels.

S3C820B MICROCONTROLLER

The S3C820B single-chip CMOS microcontroller is fabricated using a highly advanced CMOS process and is based on Samsung's newest CPU architecture.

The S3C820B is the microcontroller which has 64K-byte mask-programmable ROM and 192K-byte mask ROM for font data.

Using a proven modular design approach, Samsung engineers developed the S3C820B by integrating the following peripheral modules with the powerful SAM87 core:

— Four programmable I/O ports, excluding one BUZ pin, for a total of 32 pins.

— Eight bit-programmable pins for external interrupts.

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

— One 8-bit timer/counter and one 16-bit timer/counter with selectable operating modes.

— Watch timer for real time.

The S3C820B is a versatile microcontroller for data bank or dictionary. It is currently available in a 128-pin QFP package.

FEATURES

CPU

- SAM87 CPU core

Memory

- 64K-byte internal program memory (ROM)
- 192K-byte internal memory (ROM) for font data
- 272-byte internal register file (Excluding LCD RAM)
- 6144-byte data RAM

Instruction Set

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

Instruction Execution Time

- 1.5 μs at 4 MHz fx (minimum)
- 183 μs at 32,768 Hz fxt

Interrupts

- Five interrupt levels and 15 interrupt sources
- 15 vectors (15 sources have a dedicated vector address)
- Fast interrupt processing feature (for one selected interrupt level)

I/O Ports

- Four 8-bit I/O ports (P0–P3) for a total of 32-bit programmable pins
- Eight input pins for external interrupts
- One output only pin for BUZ

Watch Timer

- Interval time: 3.91 ms, 1s at 32,768 Hz
- Four frequency outputs to BUZ pin and BUZ pin
- Clock source generation for LCD

LCD Controller/Driver

- 65 segments and 18 common terminals
- Internal resistor circuit for LCD bias
- Voltage doubler
- All dot can be switched on/off

Timers and Timer/Counters

- One programmable 8-bit basic timer (BT) for oscillation stabilization control or watchdog timer (software reset) function
- One 8-bit timer/counter (Timer 0) with three operating modes; Interval, Capture and PWM
- One 16-bit timer/counter (Timer 1) with two 8-bit timer/counter modes; Interval

Power-Down Modes

- Idle mode (CPU clock stops)
- Stop mode (main oscillation and CPU clock stops)

Operating Temperature Range

- – 40 °C to + 85 °C

Operating Voltage Range

- 2.2 V to 4.5 V at 1 MHz fx
- 2.7 V to 4.5 V at 4 MHz fx

Package Type

- 128-pin QFP

BLOCK DIAGRAM

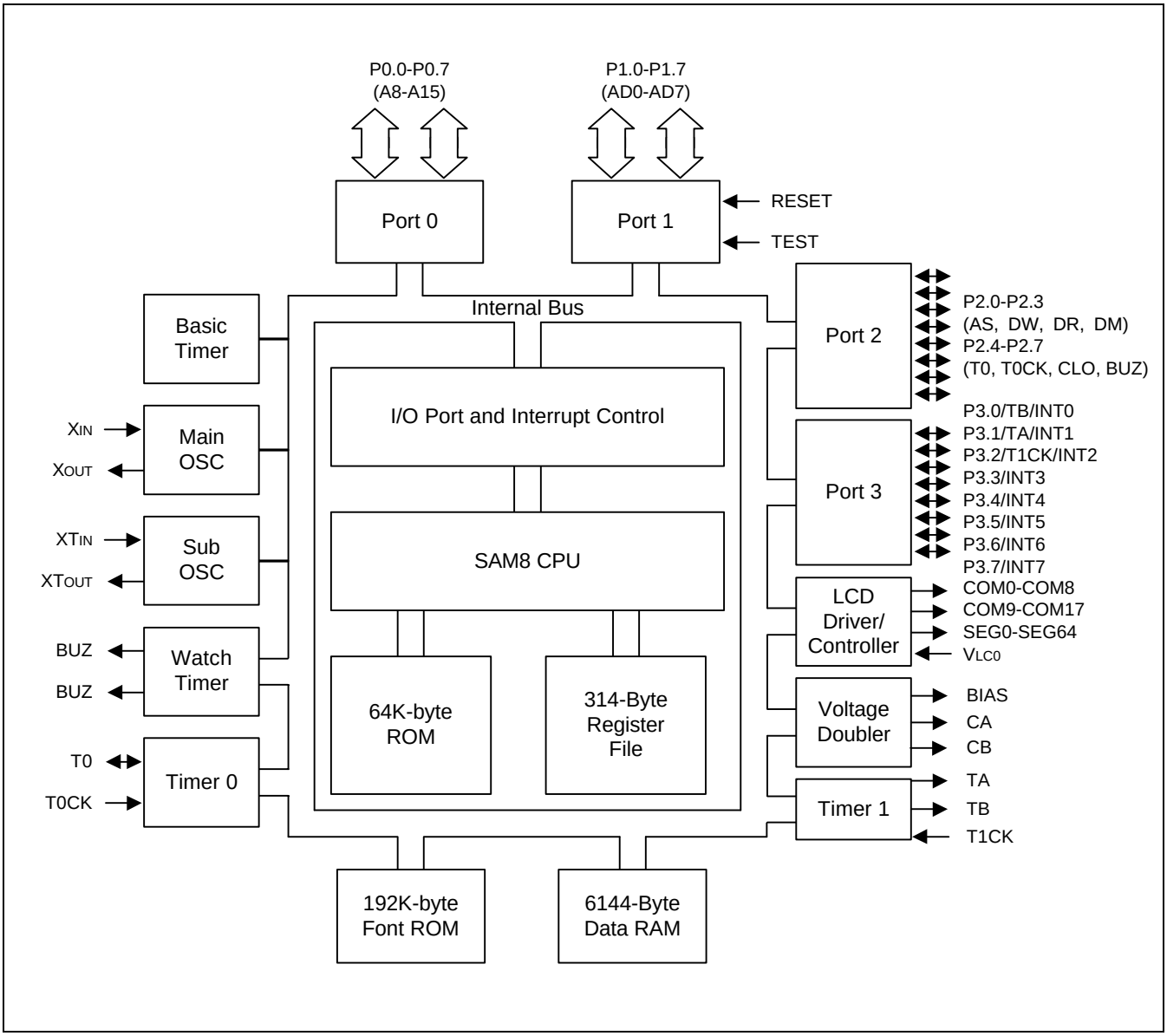


Figure 1-1. Block Diagram

PIN ASSIGNMENTS

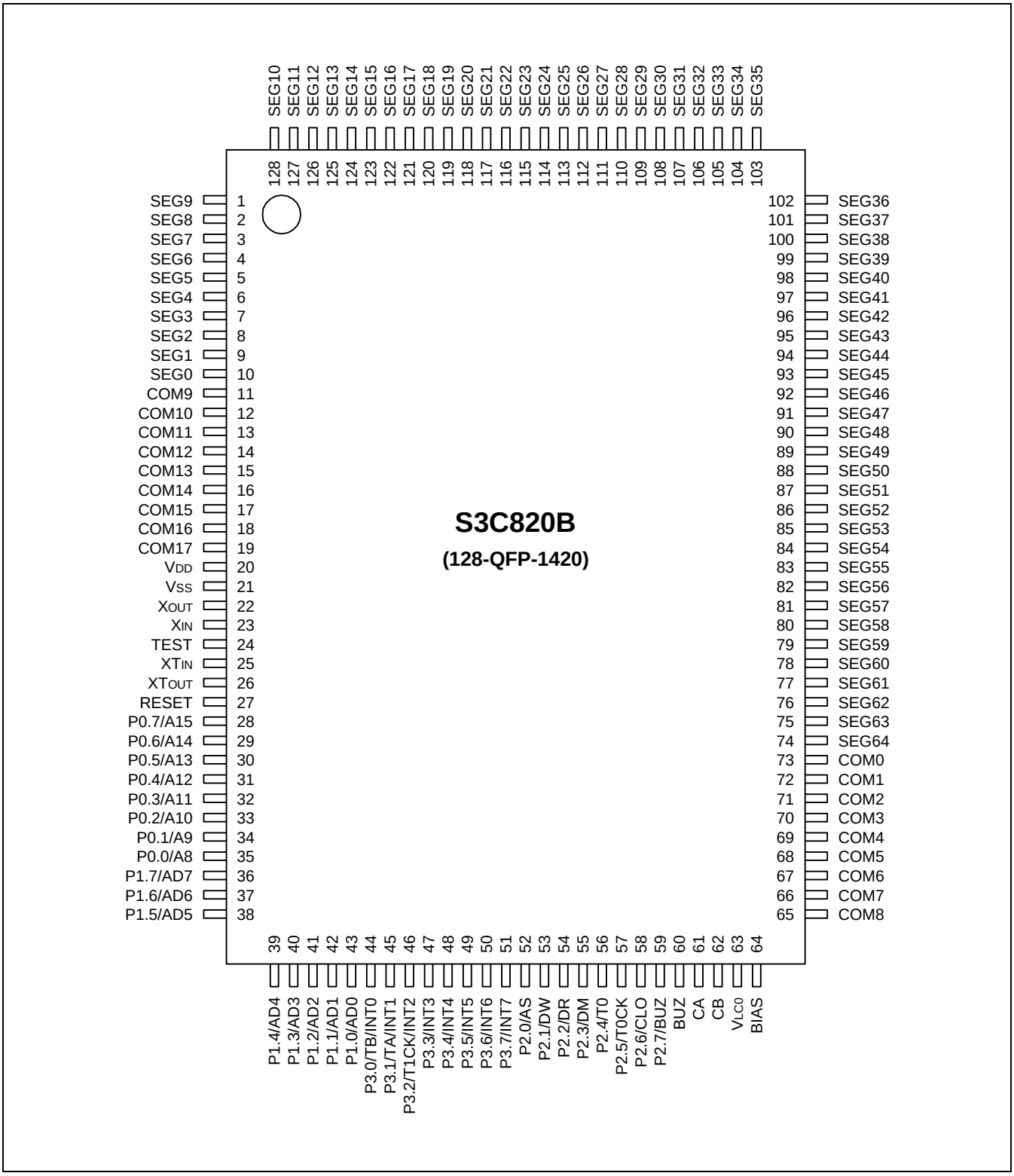


Figure 1-2. Pin Assignment (128-Pin QFP Package)

PIN DESCRIPTIONS

Table 1-1. Pin Descriptions

Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Shared Functions
P0.0–P0.7	I/O	I/O port with nibble-programmable pins; schmitt trigger input or push-pull, open-drain output and software assignable pull-up; also configurable as external interface address lines A8–A15.	3	35–28	A8–A15
P1.0–P1.7	I/O	Same general characteristics as port 0; also configurable as external interface address/data lines AD0–AD7.	3	43–36	AD0–AD7
P2.0–P2.3	I/O	I/O port with bit-programmable pins; schmitt trigger input or push-pull output and software assignable pull-ups. Lower nibble pins 0–3 are configurable for external interface signals.	5	52–55	AS, DW, DR, DM
P2.4–P2.7	I/O	P2.4/capture input, interval/PWM output (T0) P2.5/timer 0 clock input (T0CK) P2.6/system clock output (CLO) P2.7/buzzer signal output (BUZ)	6	56–59	T0, T0CK, CLO, BUZ
P3.0–P3.7	I/O	I/O port with bit-programmable pins; schmitt trigger input or push-pull output and software assignable pull-up; P3.0–P3.7 are alternately used for external interrupt input (noise filters, interrupt enable and pending control); P3.0/timer B clock output (TB)/INT0 P3.1/timer 1/A clock output (TA)/INT1 P3.2/timer 1/A clock input (T1CK)/INT2	4	44–51	TB/INT0, TA/INT1, T1CK/INT2, INT3–INT7
T1CK	I/O	Timer A external clock input pins.	4	46	P3.2/INT2
TB TA	I/O	Timer B and 1/A clock output pins.	4	44 45	P3.0/INT0 P3.1/INT1
AS, DW, DR, DM	I/O	Output pins for external interface control signals. AS: address strobe DW: data memory write DR: data memory read DM: data memory select	5	52–55	P2.0–P2.3
T0	I/O	Capture input or interval/PWM output.	6	56	P2.4
T0CK	I/O	Timer 0 clock input.	6	57	P2.5

Table 1-1. Pin Descriptions (Continued)

Pin Names	Pin Type	Pin Description	Circuit Type	Pin No.	Shared Functions
CLO	I/O	Clock output	6	58	P2.6
BUZ	I/O	Output pin for buzzer signal.	6	59	P2.7
BUZ	O	Inverted buzzer signal output.	–	60	–
INT0–INT7	I/O	External interrupt input pins.	4	44–51	P3.0/TB, P3.1/TA, P3.2/T1CK, P3.3–P3.7
AD0–AD7	I/O	Address low and data ports.	3	43–36	P1.0–P1.7
A8–A15	I/O	Address high output ports.	3	35–28	P0.0–P0.7
COM0–COM8	O	LCD common signal output.	7	73–65	–
COM9–COM17	O	LCD common signal output.	7	11–19	–
SEG0–SEG64	O	LCD seg signal output.	8	10–1 128–74	–
CA, CB	–	Capacitor terminal for voltage doubling.	–	61, 62	–
V_{LC0}	–	LCD power supply.	–	63	–
BIAS	O	Bias voltage level for LCD driving.	–	64	–
RESET	I	System reset pin	2	27	–
XT_{IN}, XT_{OUT}	–	Crystal oscillator pins for sub clock.	–	25, 26	–
TEST	I	Test signal input (must be connected to V_{DD}).	–	24	–
X_{IN}, X_{OUT}	–	Main oscillator pins	–	23, 22	–
V_{DD}, V_{SS}	–	Power input pins	–	20, 21	–

PIN CIRCUIT DIAGRAMS

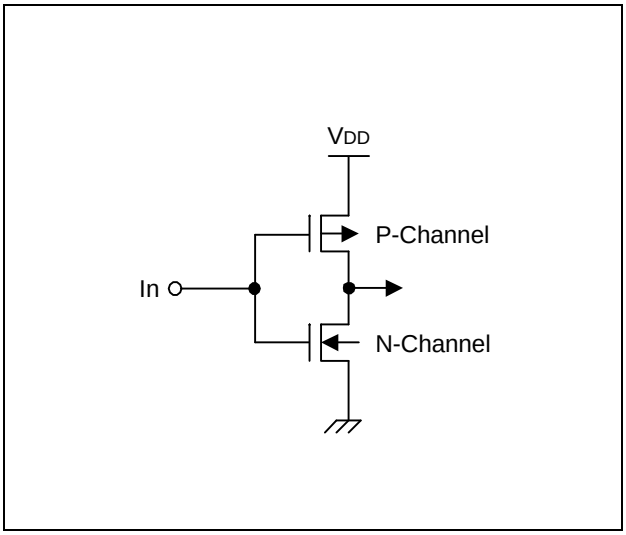


Figure 1-3. Pin Circuit Type 1

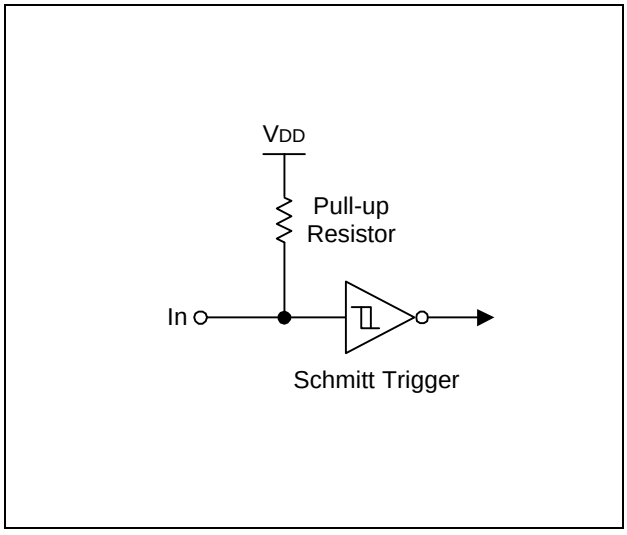


Figure 1-4. Pin Circuit Type 2 (RESET)

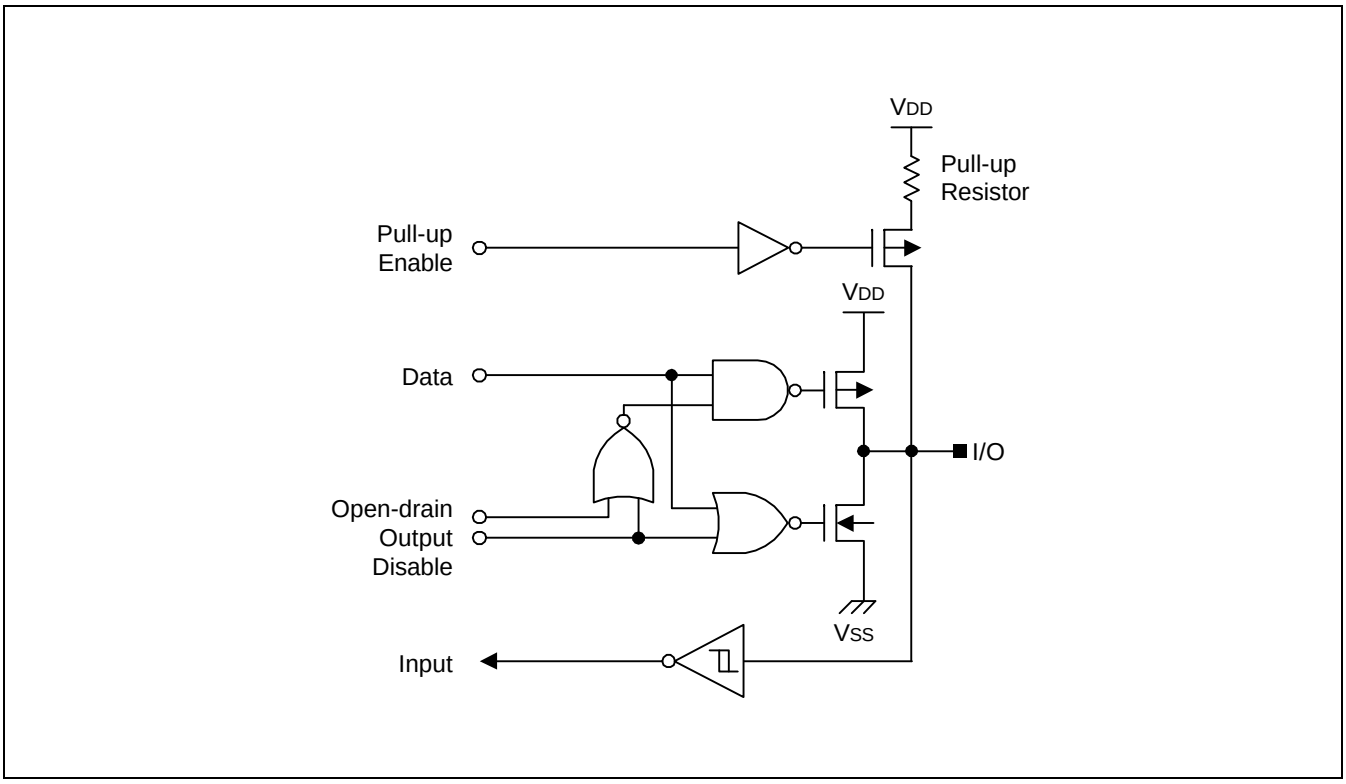


Figure 1-5. Pin Circuit Type 3 (Ports 0, 1)

PIN CIRCUIT DIAGRAMS (Continued)

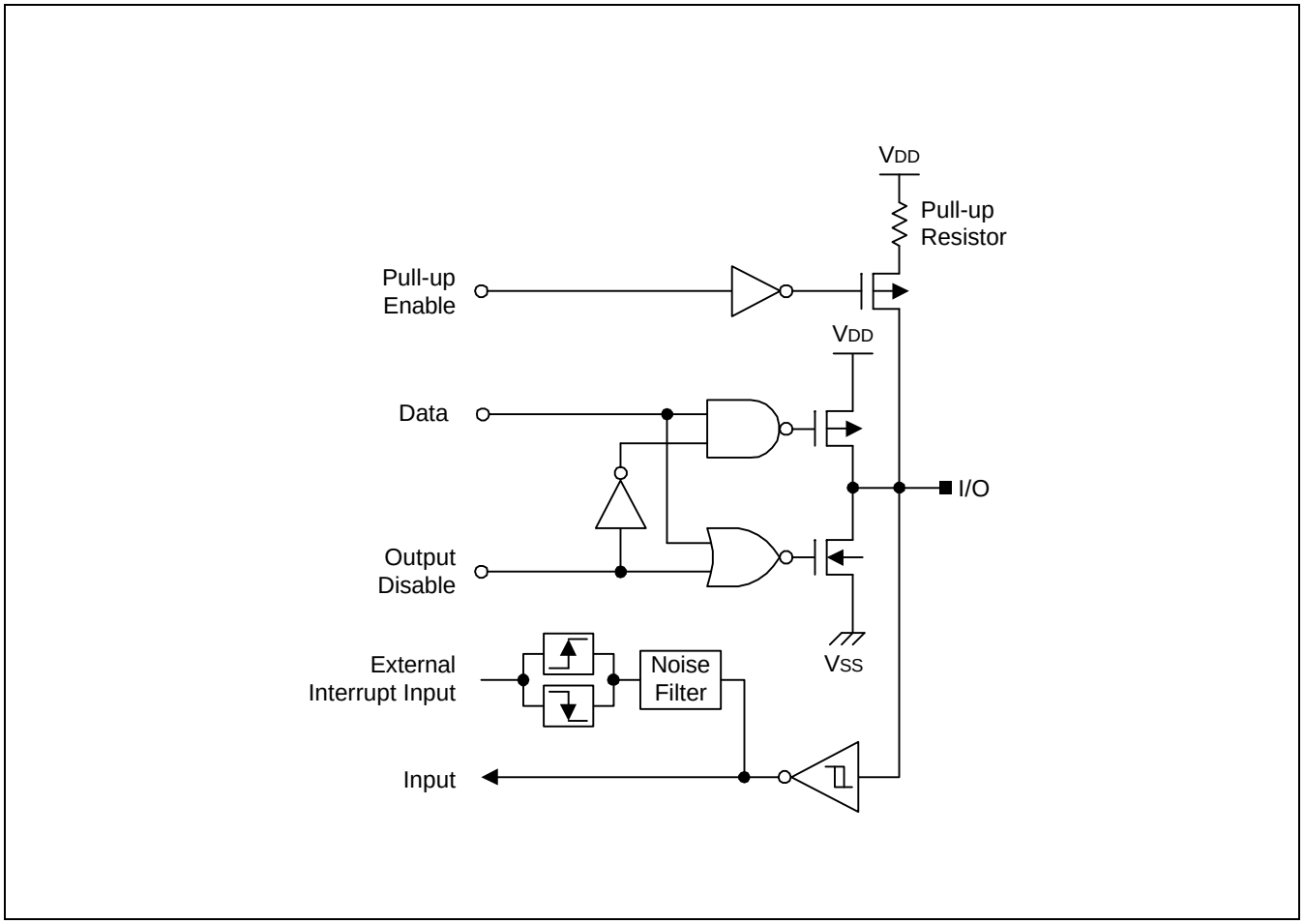


Figure 1-6. Pin Circuit Type 4 (Port 3)

PIN CIRCUIT DIAGRAMS (Continued)

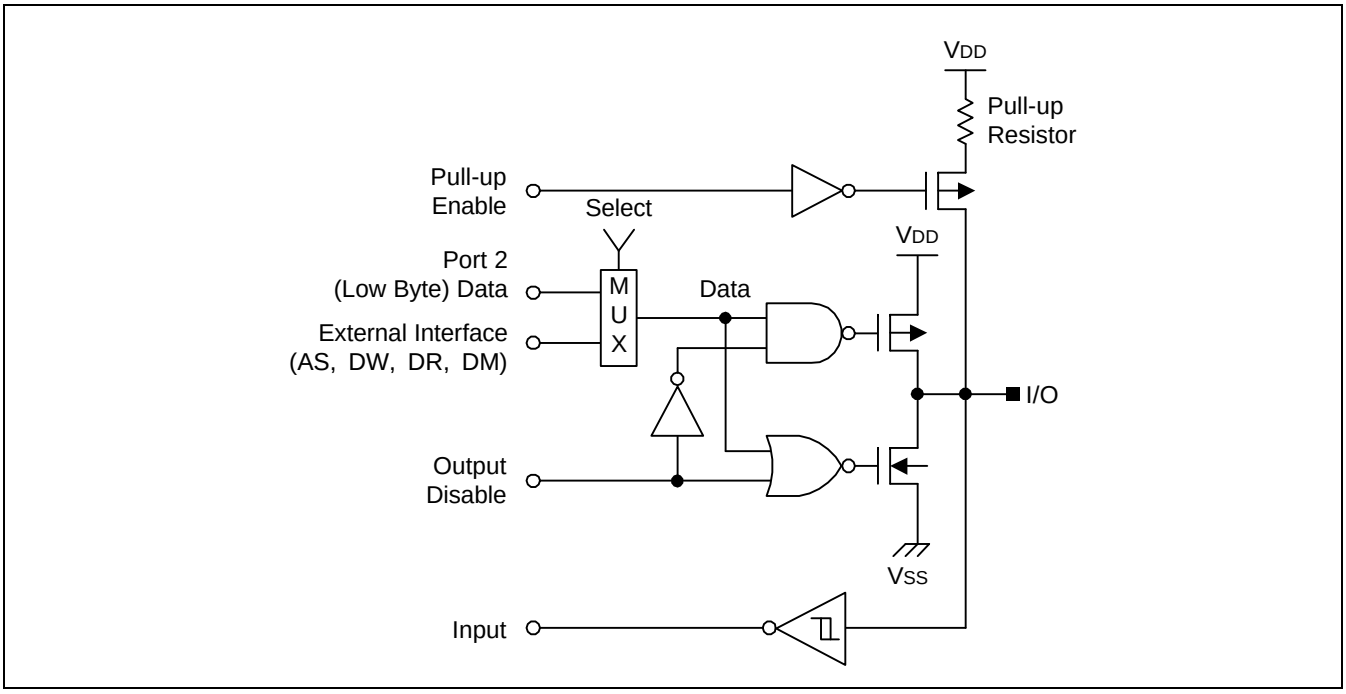


Figure 1-7. Pin Circuit Type 5 (Ports 2.0–2.3)

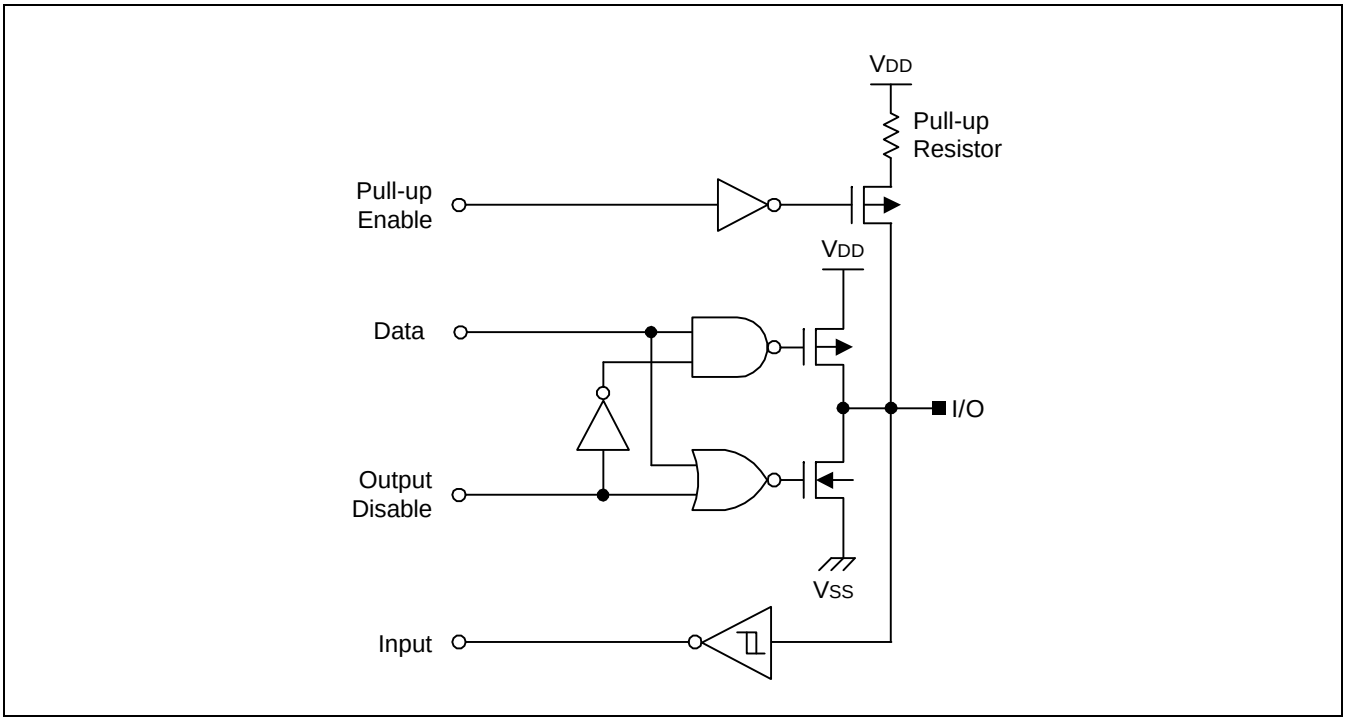


Figure 1-8. Pin Circuit Type 6 (Ports 2.4–2.7)

PIN CIRCUIT DIAGRAMS (Continued)

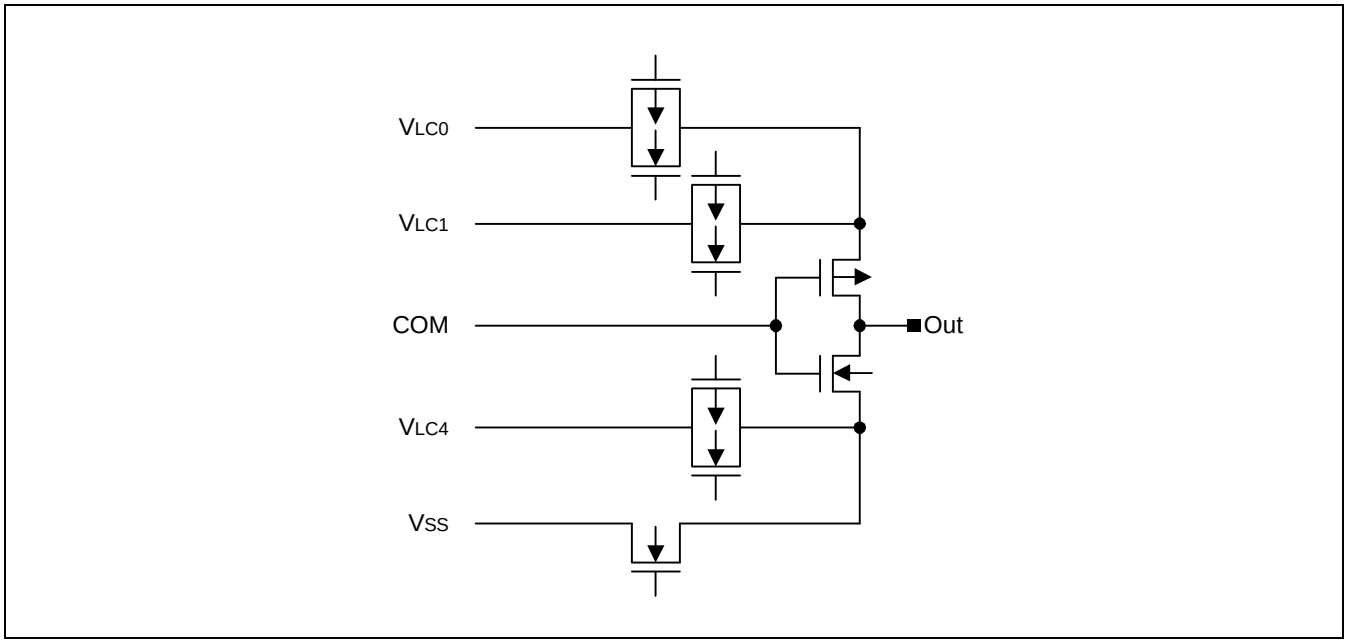


Figure 1-9. Pin Circuit Type 7 (COM0–COM17)

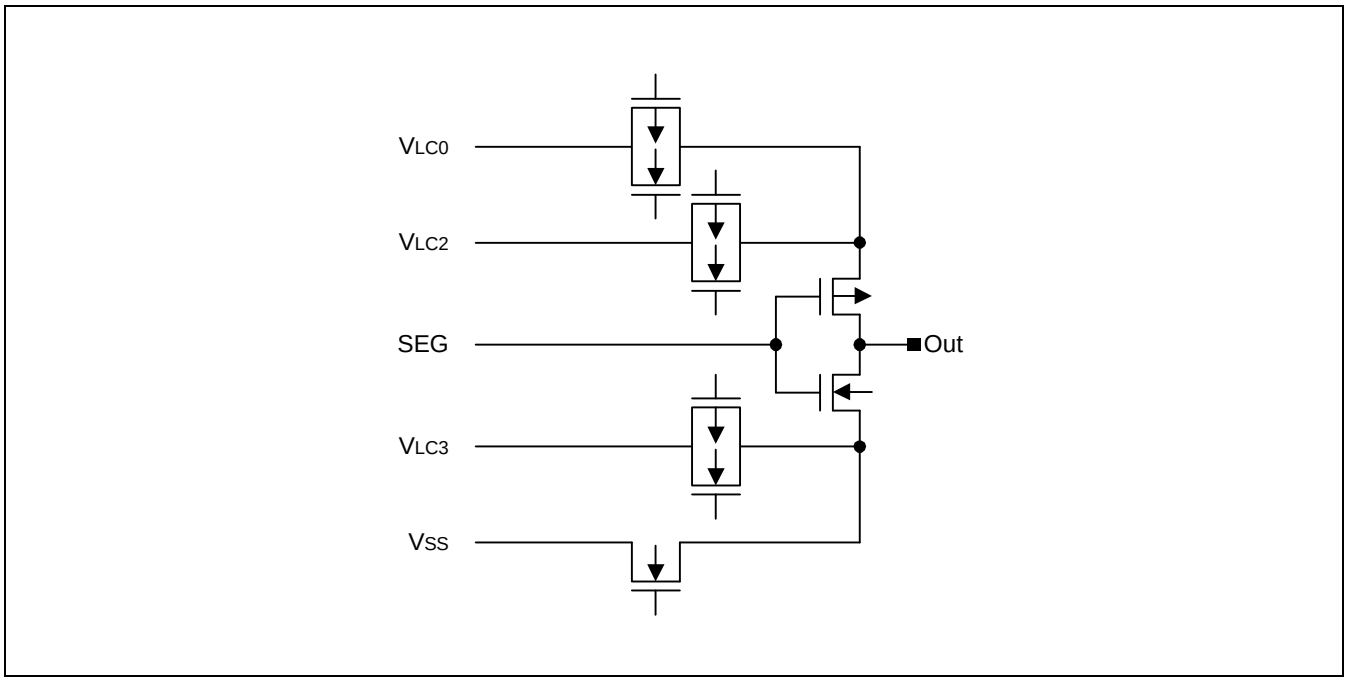


Figure 1-10. Pin Circuit Type 8 (SEG0–SEG64)

2 ADDRESS SPACES

OVERVIEW

The S3C820B microcontroller has five types of address space:

— Internal program memory (ROM)
— Internal memory (ROM) for font data
— Internal register file
— Internal data memory (RAM)
— LCD display register (RAM) file

A 16-bit address bus supports program memory operations and another 16-bit address bus and 3-bit selection bus support memory operations for font data. A separate 8-bit register bus carries addresses and data between the CPU and the register file and 13-bit address bus and 3-bit selection bus carries addresses and data between the CPU and the internal 6K-bytes data RAM.

The S3C820B has an internal 64K-byte mask-programmable ROM and 192K-byte mask ROM for font data. An external memory interface is implemented.

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

The 6,144-byte data memory (RAM) is implemented and 162-byte LCD display register file is implemented too.

There are 314 mapped registers in the internal register file. Of these, 272 are for general-purpose use. (This number includes a 16-byte working register common area that is used as a “scratch area” for data operations, a 192-byte prime register area, and a 64-byte area (Set 2) that is also used for stack operations.) Nineteen 8-bit registers are used for CPU and system control and 23 registers are mapped peripheral control and data registers. Six register locations are not mapped.

PROGRAM MEMORY (ROM)

Program memory (ROM) stores program code or table data. The S3C820B has 64K-bytes of internal mask-programmable program memory. The program memory address range is therefore 0H–FFFFH (see Figure 2-1).

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 do use the vector address area to store program code, be careful to avoid overwriting vector addresses stored in these locations.

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

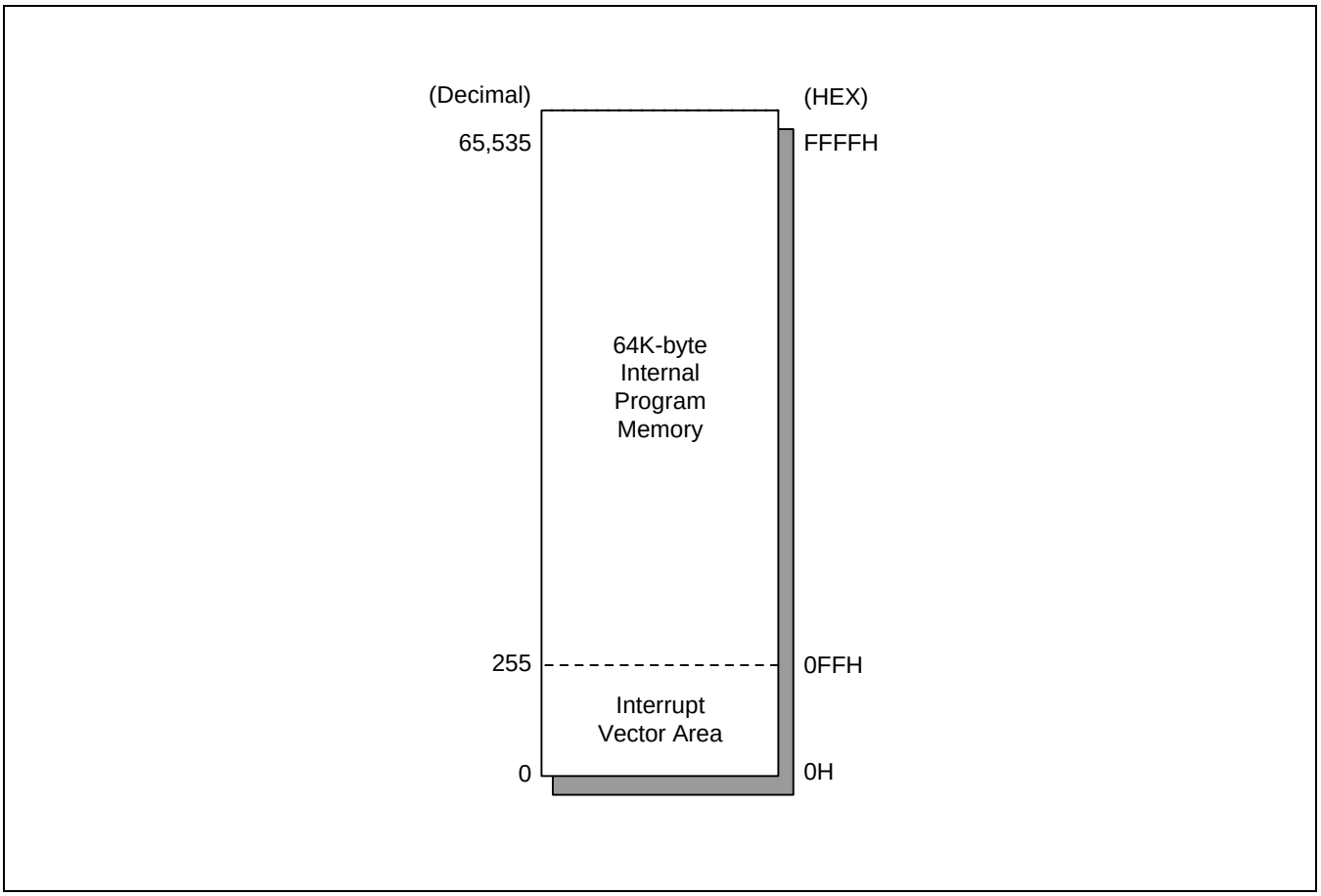


Figure 2-1. Program Memory Address Space

REGISTER ARCHITECTURE

The S3C820B register file has 314 registers excluding LCD data registers. To increase the size of the internal register file, the upper 64-byte area of the register file is expanded two 64-byte areas, *set 1* and *set 2*. The remaining 192-byte area of the physical register file contains freely-addressable, general-purpose registers called *prime registers.*

The extension of register space into separately addressable sets is supported internally by addressing mode restrictions.

LCD data registers are 16 × 8-bytes + 16 × 1-bit of page 1 and 2 × 8-bytes + 2 × 1-bit of page 2 and can be used as general-purpose registers. Please refer to Chapter 13.

Specific register types and the area (in bytes) they occupy in the S3C820B internal register space are summarized in Table 2-1.

Table 2-1. S3C820B Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16-byte common working register area, the 192-byte prime register area, and the 64-byte set 2 area)	272
LCD data registers	162
CPU and system control registers	19
Mapped clock, peripheral, and I/O control and data registers	23
Total Addressable Bytes	**461**

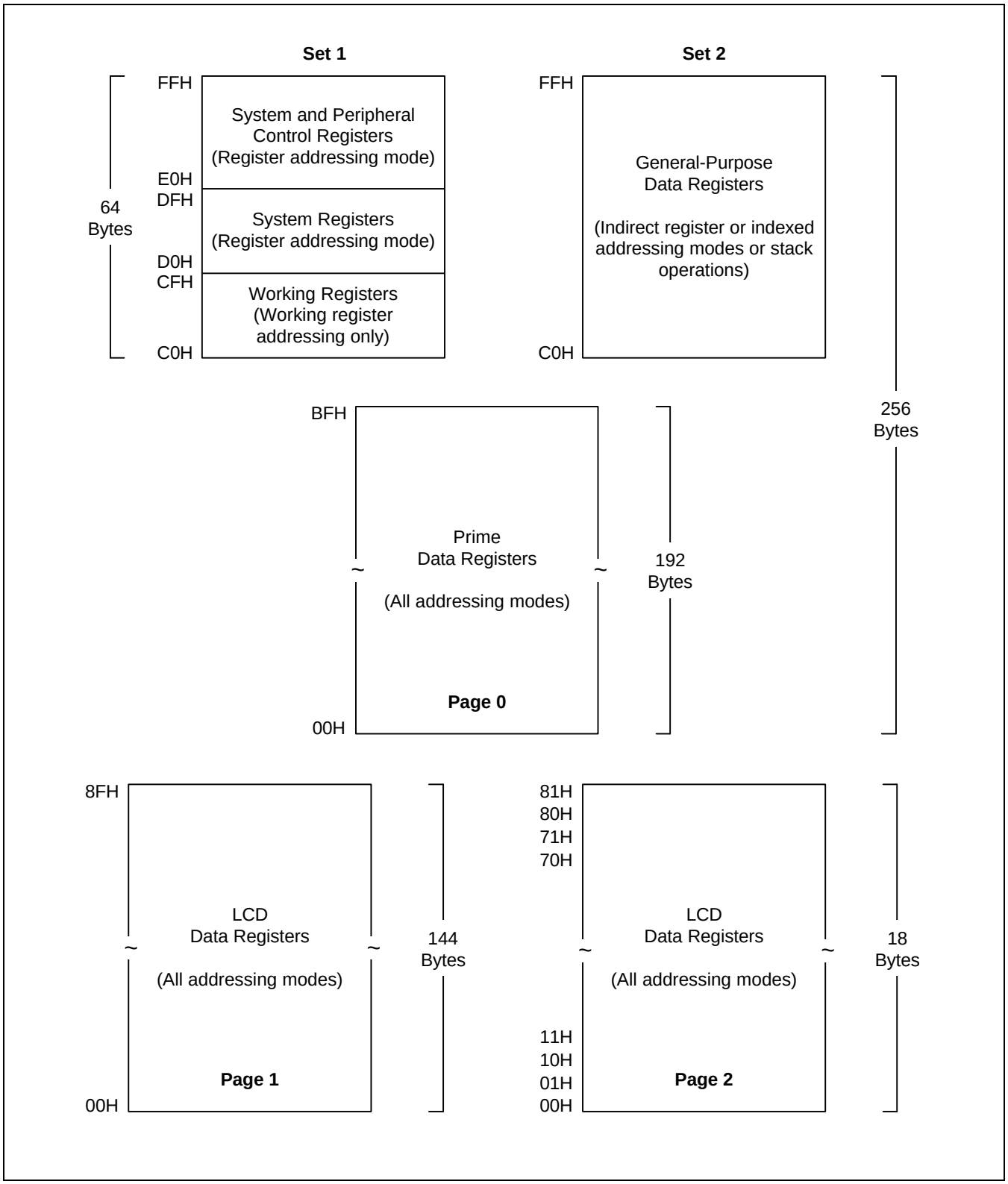
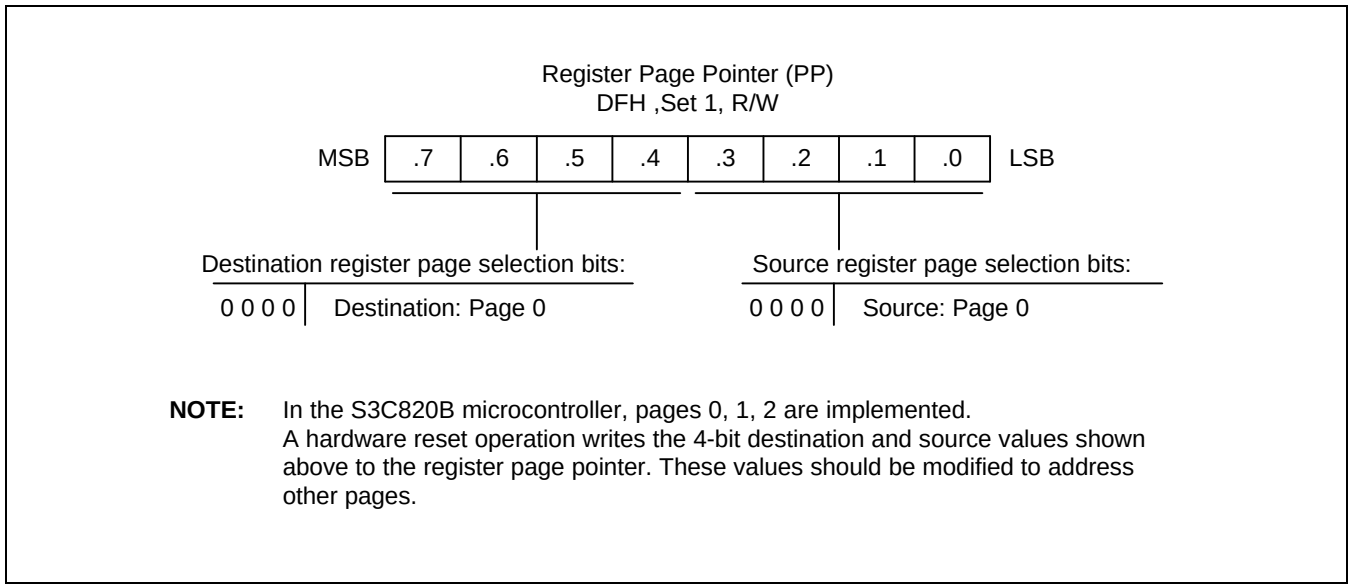


Figure 2-2. Internal Register File Organization

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 S3C820B microcontroller, a paged register file expansion is implemented for LCD data registers, and the register page pointer must be changed to address other page.

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



NOTE: In the S3C820B microcontroller, pages 0, 1, 2 are implemented. A hardware reset operation writes the 4-bit destination and source values shown above to the register page pointer. These values should be modified to address other pages.

Figure 2-3. Register Page Pointer (PP)

☞ PROGRAMMING TIP – Using the Page Pointer for RAM Clear (Page0, Page1)

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

        SRP     #0C0H

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

        LD      PP, #10H        ;  Destination ← 1, source ← 0
        LD      R0, #0FFH       ;  Page 1 RAM clear starts
RAMCL1  CLR     @R0
        DJNZ    R0, RAMCL1
        CLR     @R0             ;  R0 = 00H
```

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

REGISTER SET 1

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

In some S3C8-series microcontrollers, the upper 32-byte area of this 64-byte space (E0H–FFH) is divided into 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. In the S3C820B microcontroller, bank 1 is not implemented. A hardware reset operation therefore always selects bank 0 addressing, and the SB0 and SB1 instructions are not necessary.

The upper 32-byte area of set 1 (FFH–E0H) contains 26 mapped system and peripheral control registers. The lower 32-byte area contains 16 system registers (DFH–D0H) and a 16-byte common working register area (CFH–C0H). 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 the 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*. All set 2 locations (C0H–FFH) are addressed as part of page 0 in the S3C820B register space.

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; to access registers in set 2, you must use Register Indirect addressing mode or Indexed addressing mode.

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

PRIME REGISTER SPACE

The lower 192-bytes of the 256-byte physical internal register file (00H–BFH) is called the *prime register space* or, more simply, the *prime area*. You can access registers in this address using any addressing mode. (In other words, there is no addressing mode restriction for these registers, as is the case for set 1 and set 2 registers.) All registers in prime area locations are addressable immediately following a reset.

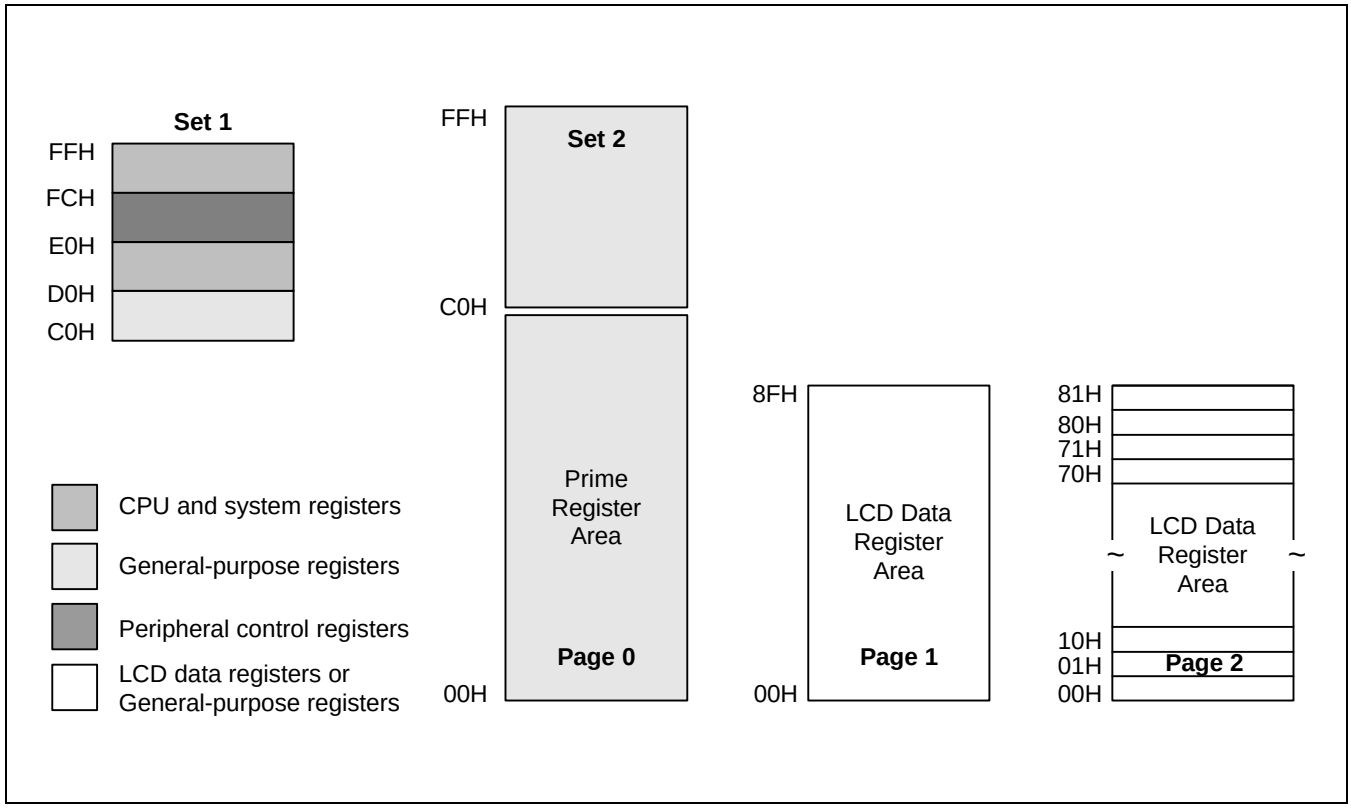


Figure 2-4. Set 1, Set 2, Prime Area Register, and LCD Data Register Map

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 consisting of 32 8-byte register groups or “slices”. Each slice consists 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 for 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 of 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).

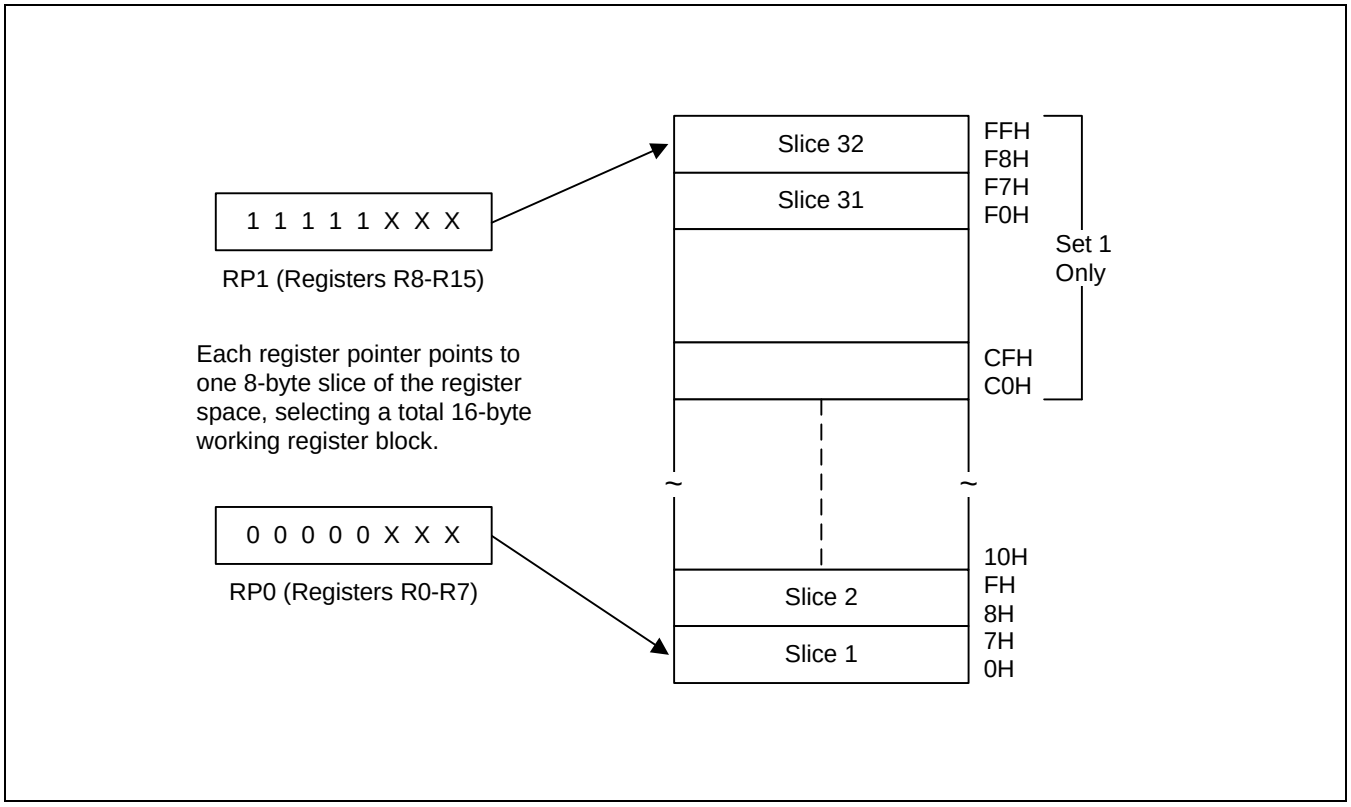


Figure 2-5. 8-Byte Working Register Areas (Slices)

USING THE REGISTER POINTERS

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-byte 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, we recommend 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 the either of the two 8-byte slices in the working register block, you can define the working register area very flexibly to support program requirements.

☞ PROGRAMMING TIP — Setting the Register Pointers

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

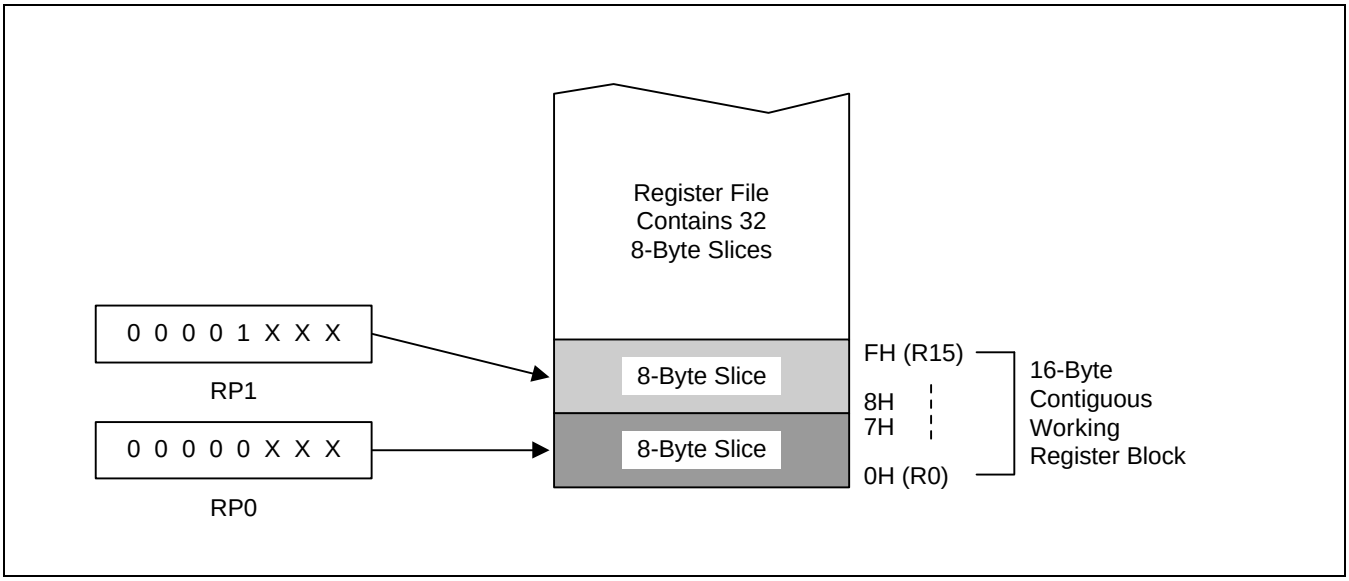


Figure 2-6. Contiguous 16-Byte Working Register Block

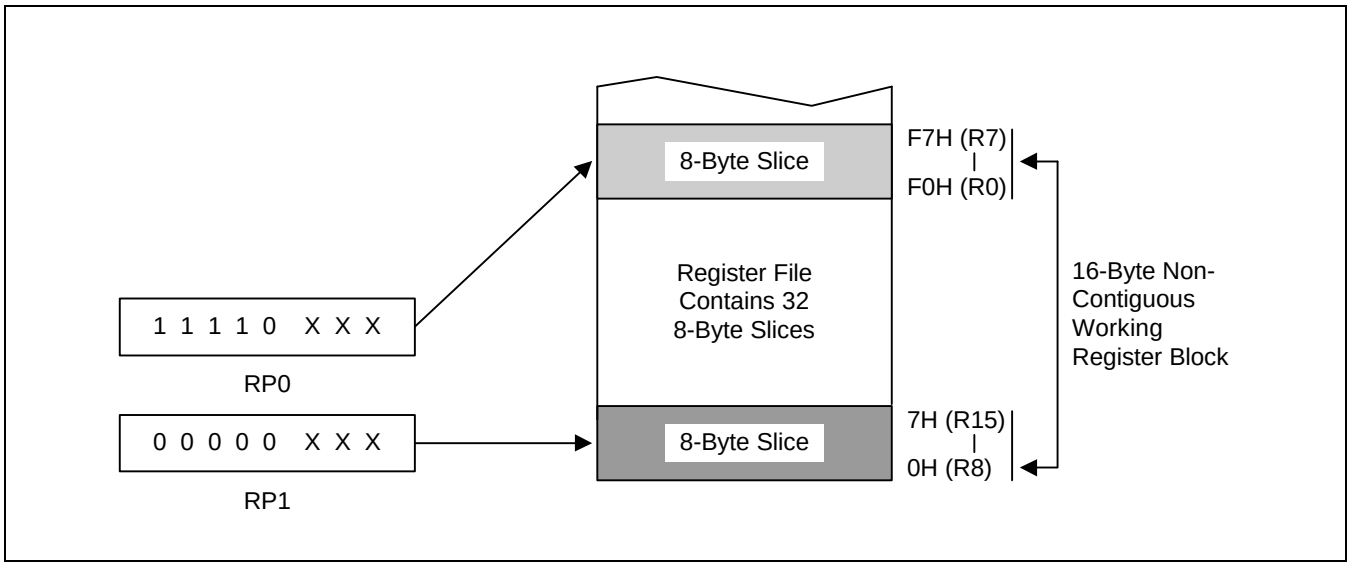


Figure 2-7. Non-Contiguous 16-Byte Working Register Block

☞ 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 80H through 85H contains the values 10H, 11H, 12H, 13H, 14H, and 15 H, respectively:

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

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

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

Now, the sum of the six registers is also located in register 80H. However, this instruction string takes 15-bytes of instruction code instead of 12-bytes, and its execution time is 50 cycles instead of 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 all locations 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; the least significant byte is always stored in the next (+ 1) odd-numbered register.

Working register addressing differs from Register addressing because 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.

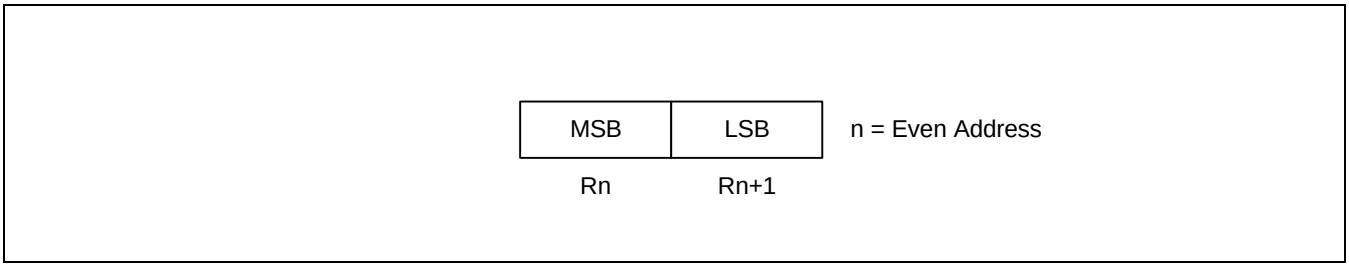


Figure 2-8. 16-Bit Register Pair

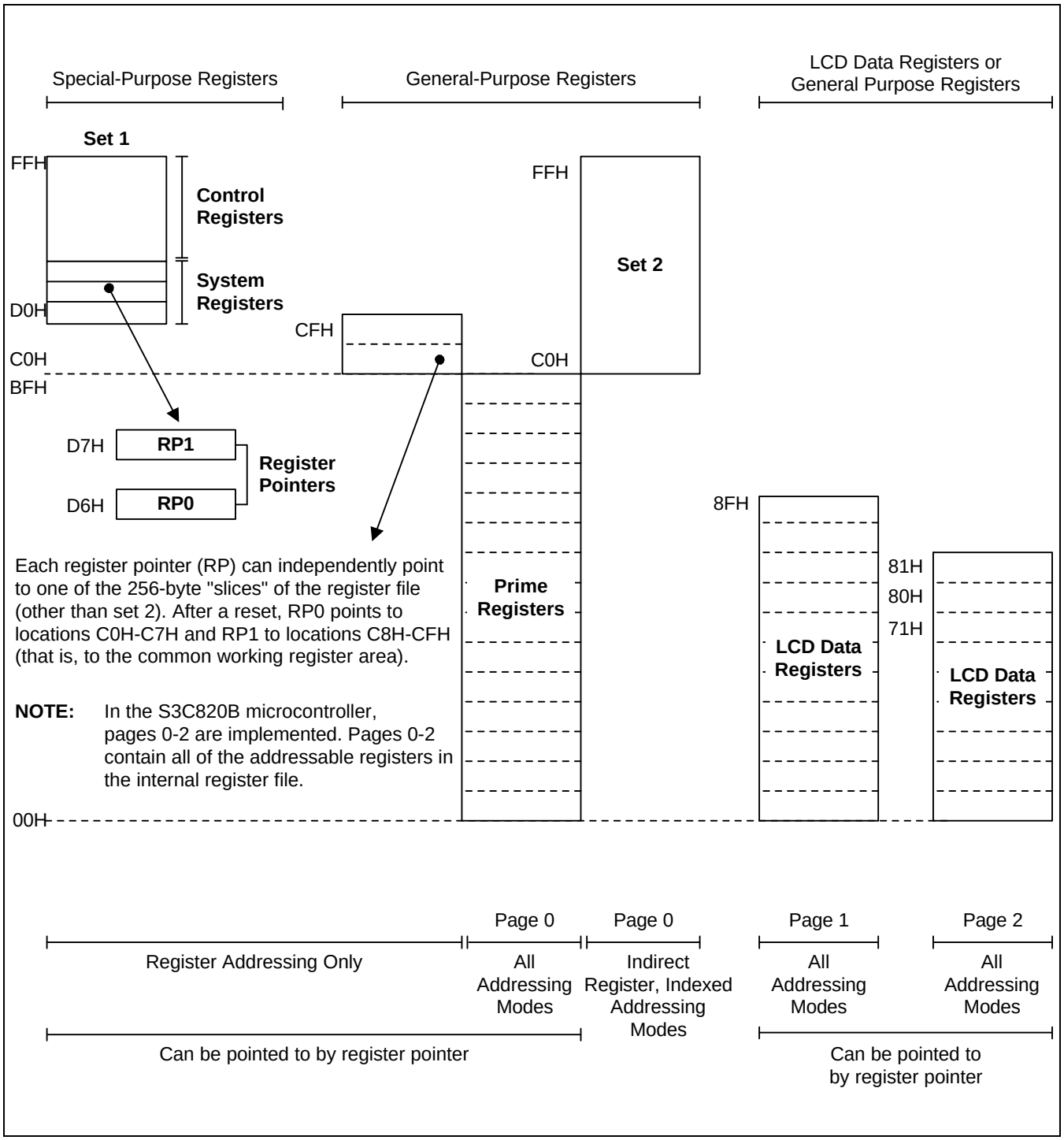


Figure 2-9. Register File Addressing

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:

RP0 → C0H–C7H

RP1 → C8H–CFH

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

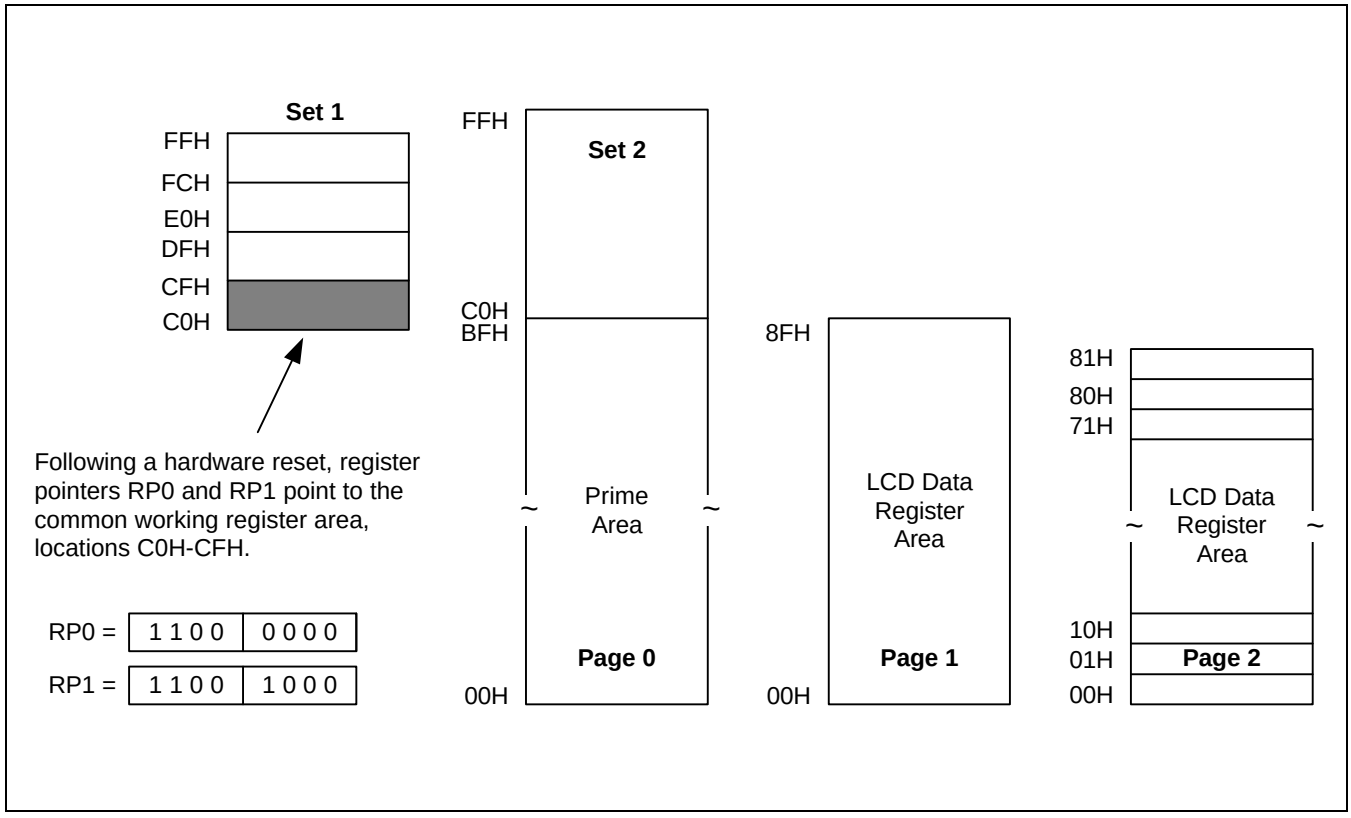


Figure 2-10. Common Working Register Area

☞ 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.

```
Example 1:  LD      0C2H, 40H       ;  Invalid addressing mode!

            Use working register addressing instead:

            SRP     #0C0H

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

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

            Use working register addressing instead:

            SRP     #0C0H

            ADD     R3, #45H        ;  R3 (C3H) ← R3 + 45H
```

4-BIT WORKING REGISTER ADDRESSING

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

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

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

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

As shown in Figure 2-11, 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).

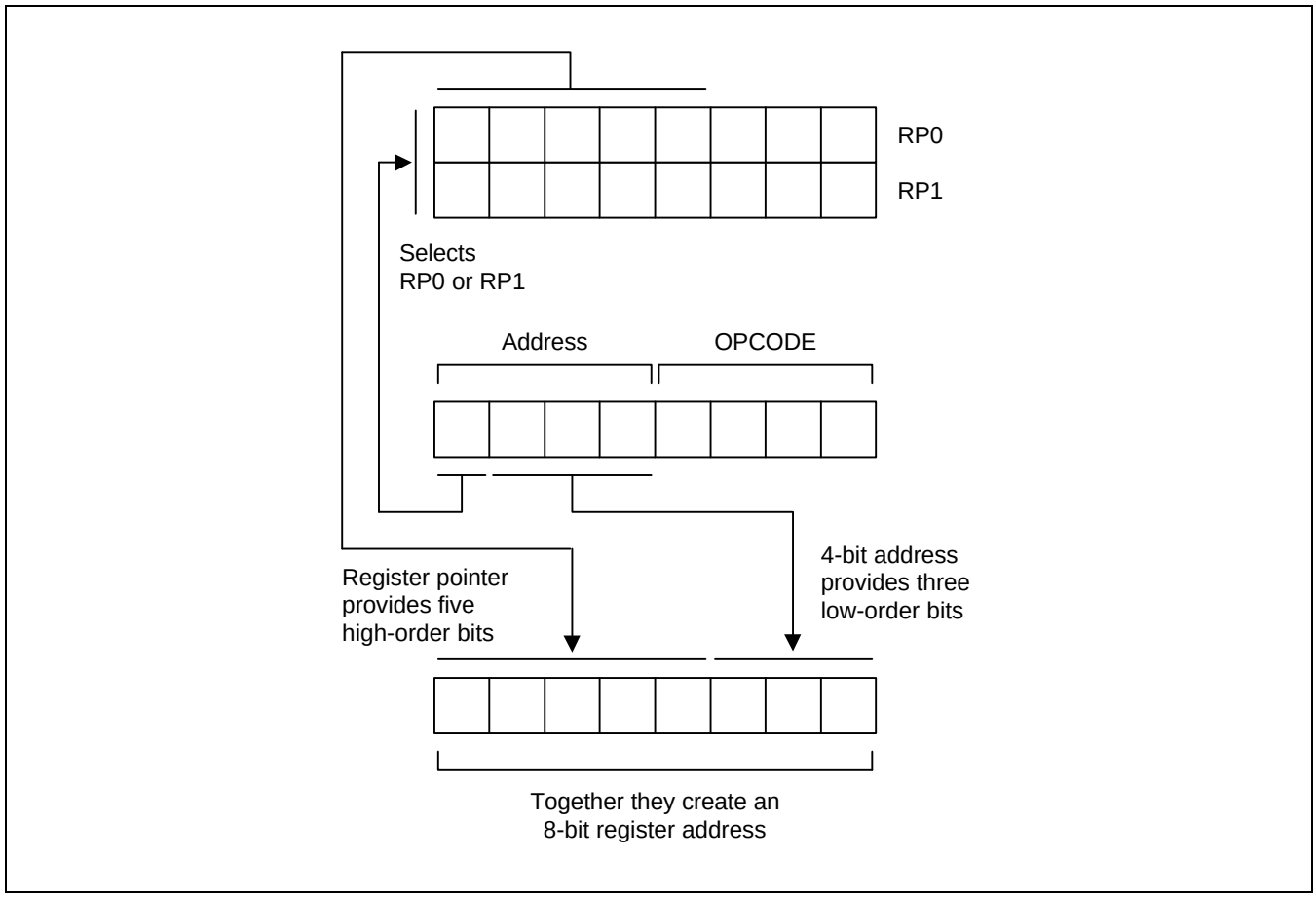


Figure 2-11. 4-Bit Working Register Addressing

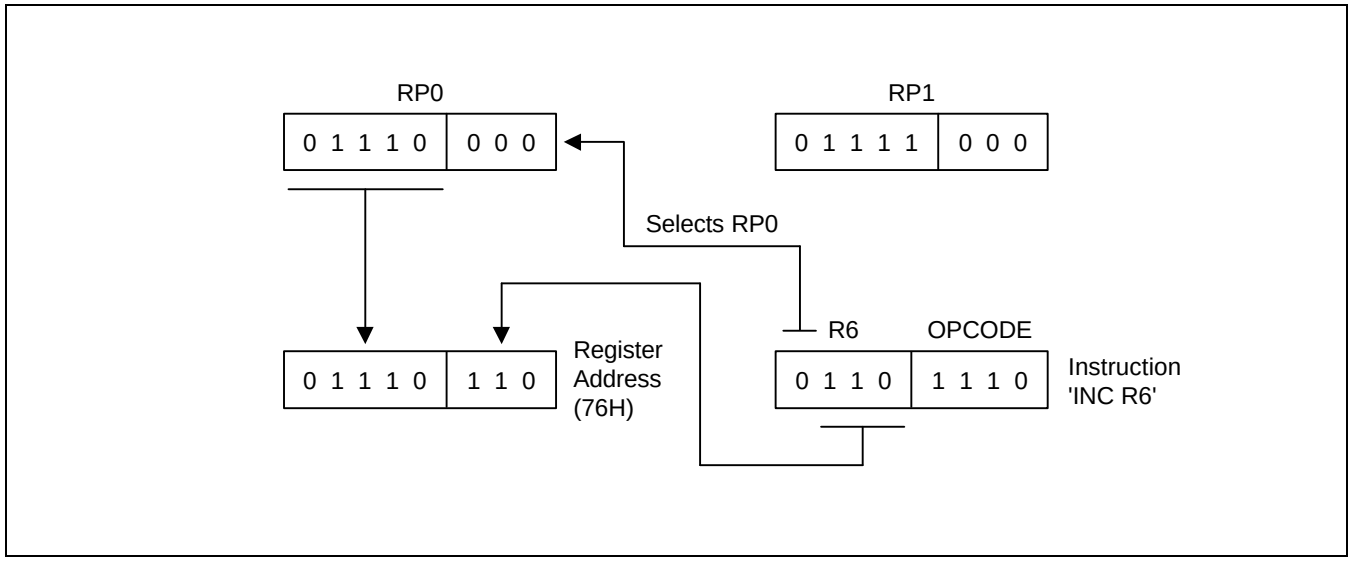


Figure 2-12. 4-Bit Working Register Addressing Example

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 (10101011B).

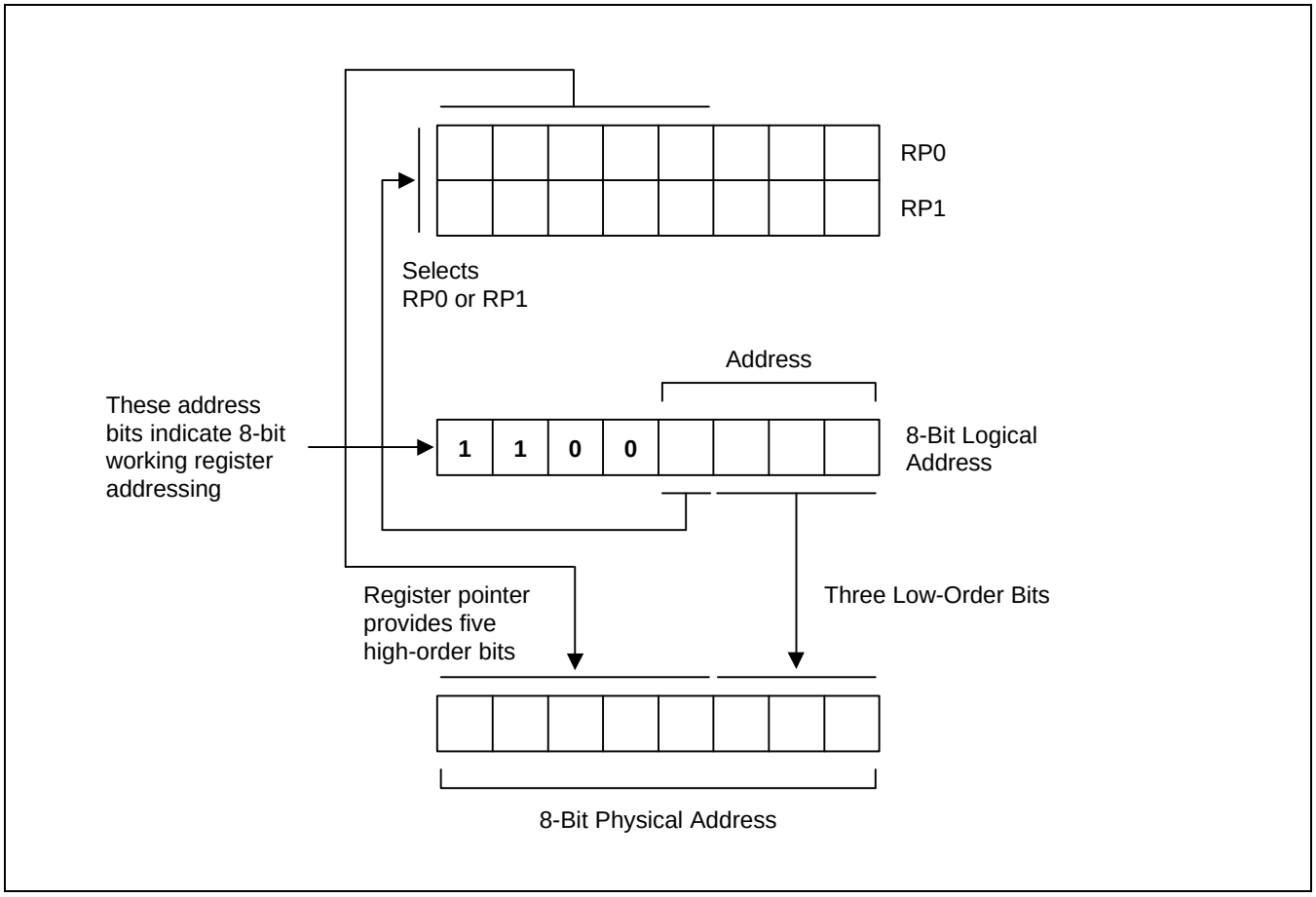


Figure 2-13. 8-Bit Working Register Addressing

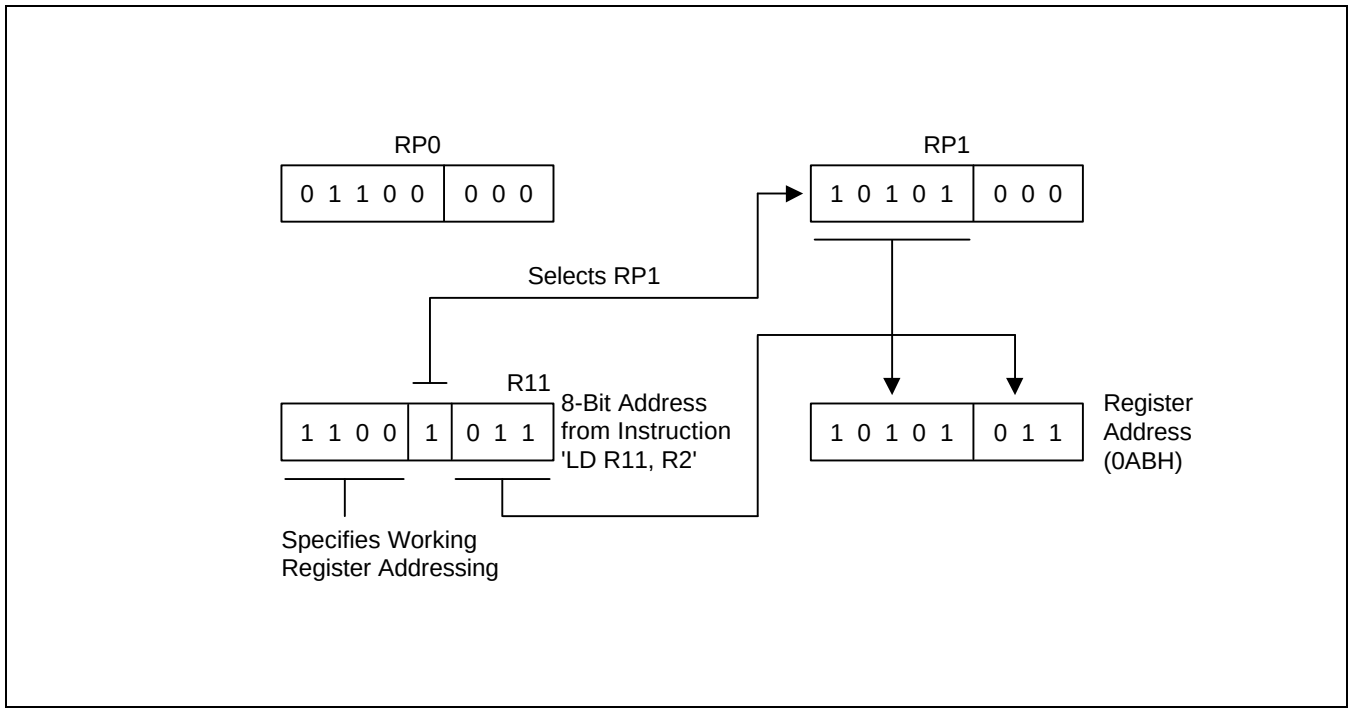


Figure 2-14. 8-Bit Working Register Addressing Example

SYSTEM AND USER STACKS

S3C8-series microcontrollers use the system stack for subroutine calls and returns and to store data. The PUSH and POP instructions are used to control system stack operations. The S3C820B architecture supports stack operations in the internal register file.

Stack Operations

Return addresses for procedure calls and interrupts and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the PC and the FLAGS register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address 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.

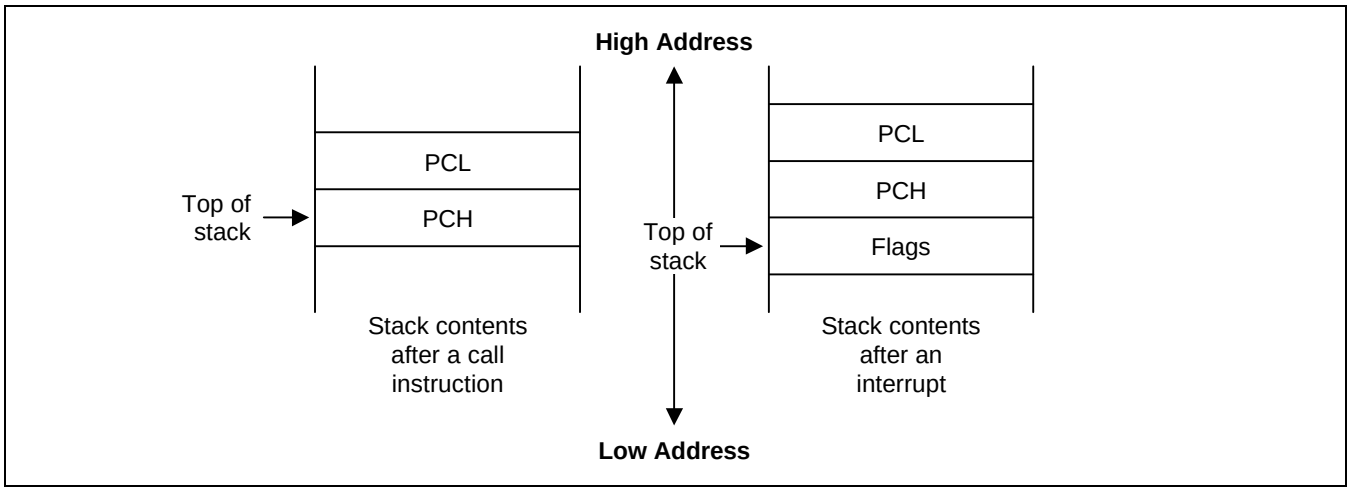


Figure 2-15. Stack Operations

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); 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 S3C820B, 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 the result of increasing or decreasing the stack address value in the SPL register during normal stack operations, the value in the SPL register will overflow (or underflow) to the SPH register, overwriting any other data that is currently stored there. To avoid overwriting data in the SPH register, you can initialize the SPL value to “FFH” instead of “00H”.

☞ PROGRAMMING TIP — Standard Stack Operations Using PUSH and POP

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

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

3 ADDRESSING MODES

OVERVIEW

The program counter is used to fetch instructions that are stored in program memory for execution. 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 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:

— 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, the operand is the content of a specified register or register pair (see Figure 3-1). Working register addressing differs from Register addressing because 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).

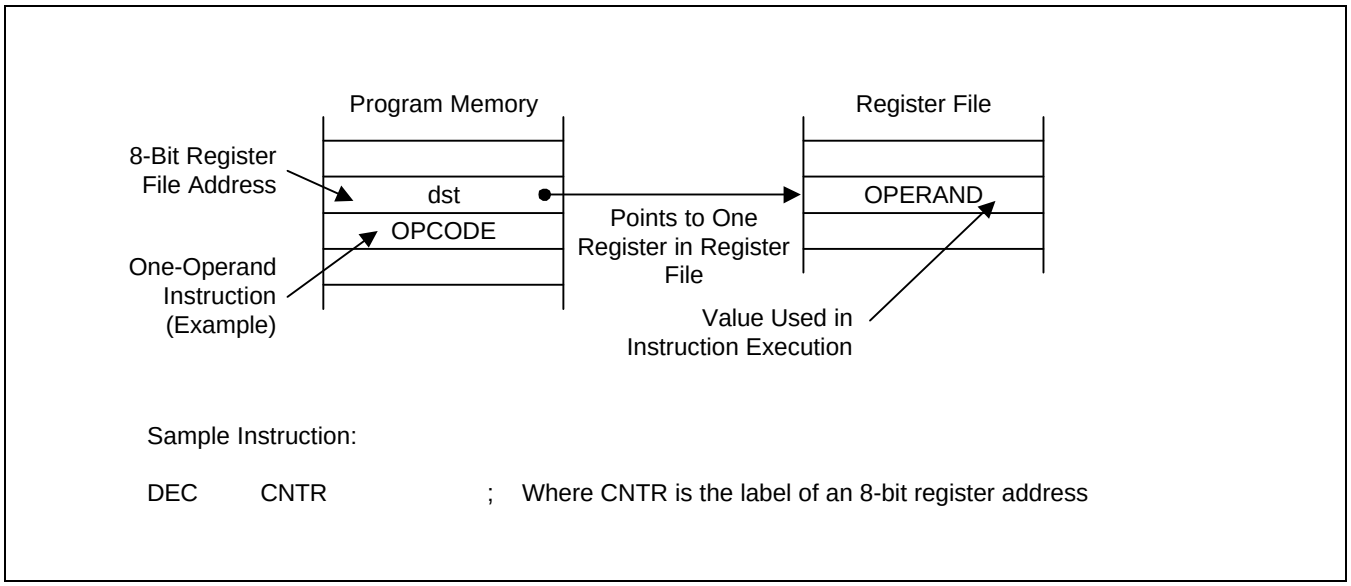


Figure 3-1. Register Addressing

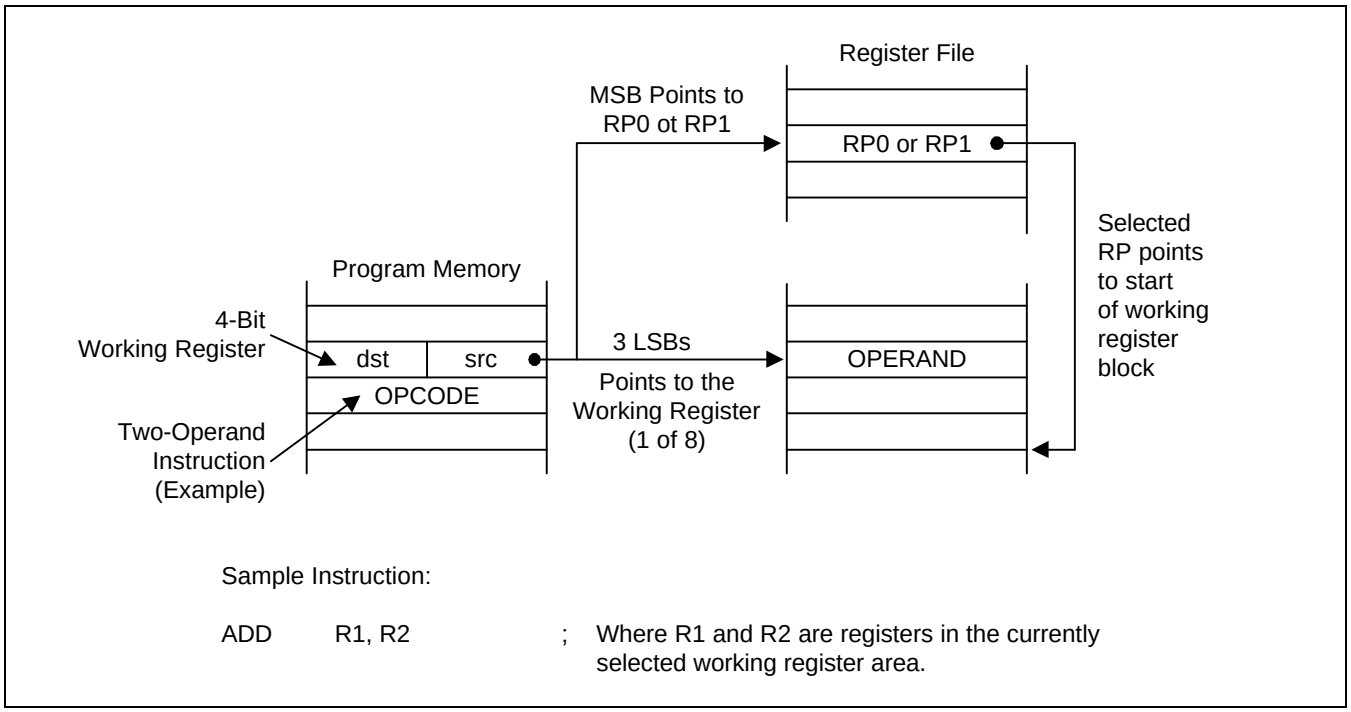
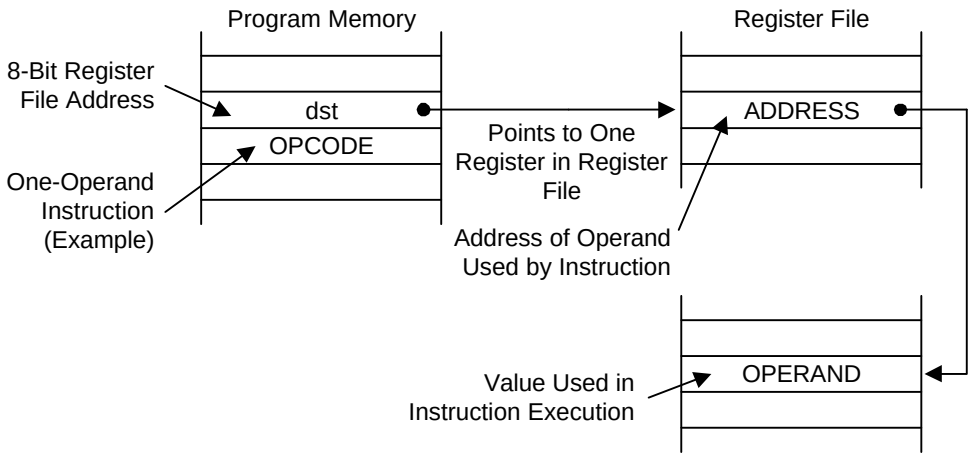


Figure 3-2. Working Register Addressing

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, if implemented (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. Remember, however, that locations C0H–FFH in set 1 cannot be accessed using Indirect Register addressing mode.



Sample Instruction:

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

Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

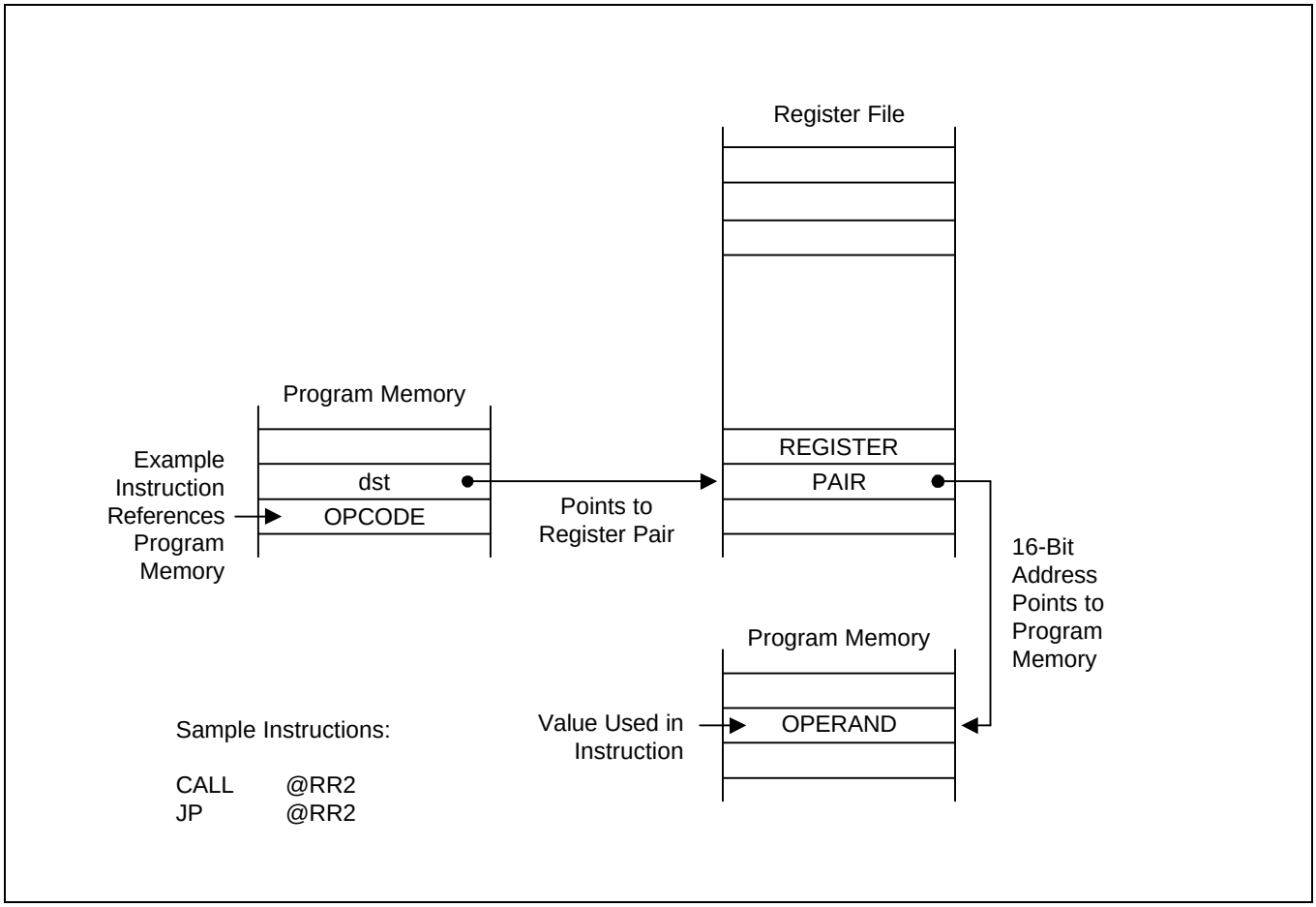


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

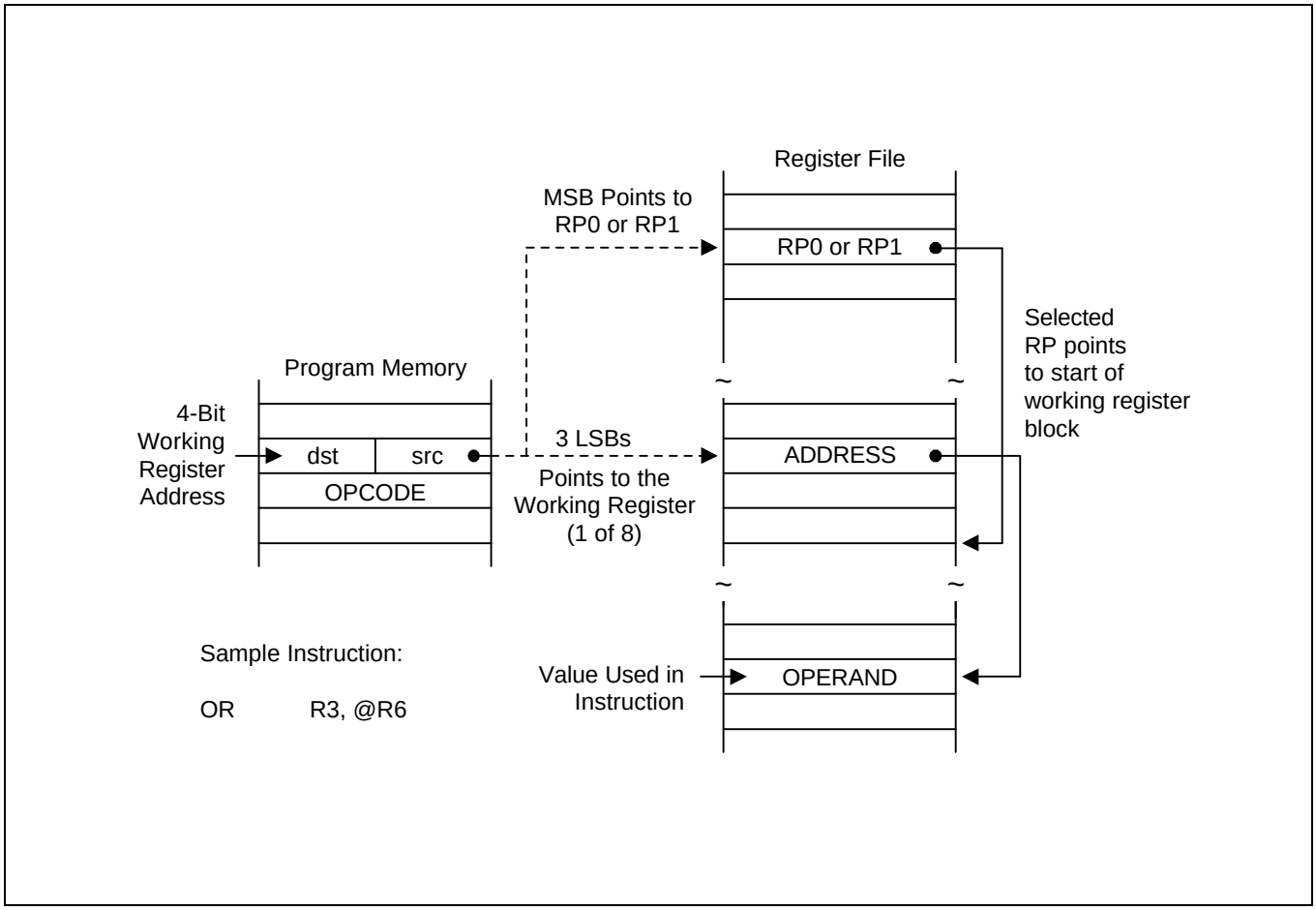
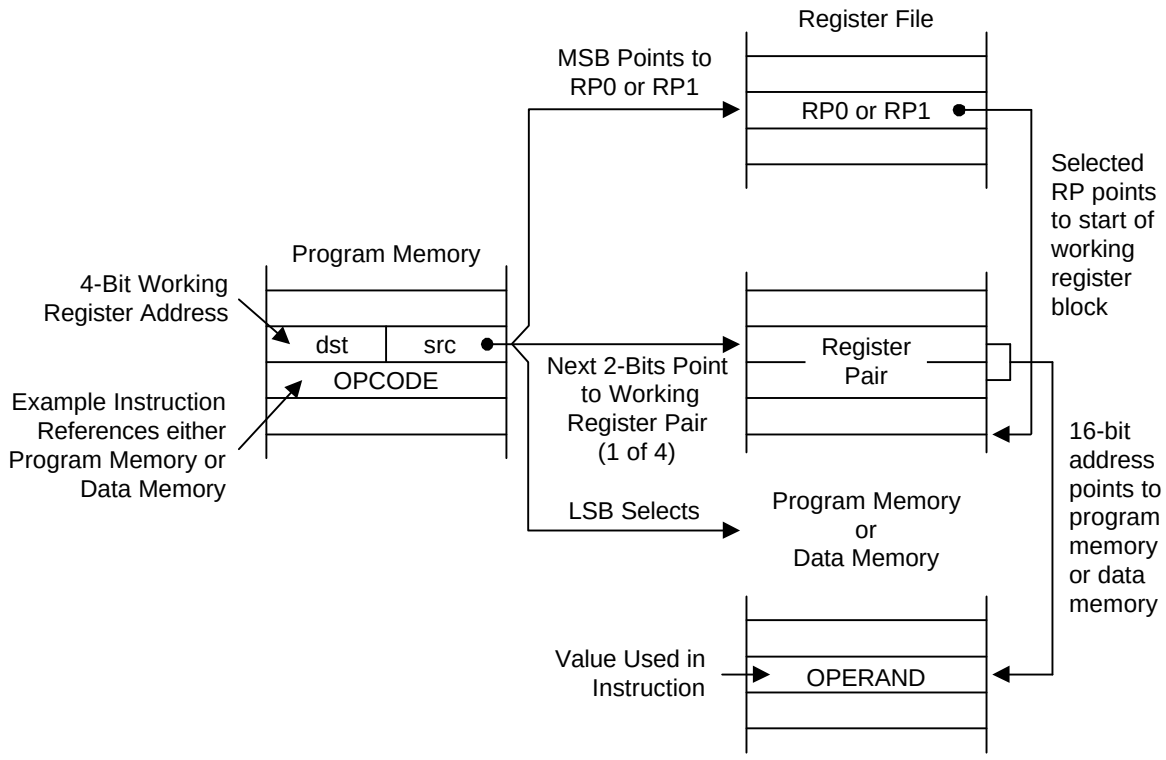


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)



Sample Instructions:

```
LDC     R5, @RR6        ;  Program memory access
LDE     R3, @RR14       ;  Internal font and data memory, and External data
                           memory access.
LDE     @RR4, R8        ;  Internal font and data memory, and External data
                           memory access.
```

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

INDEXED ADDRESSING MODE (X)

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

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

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

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

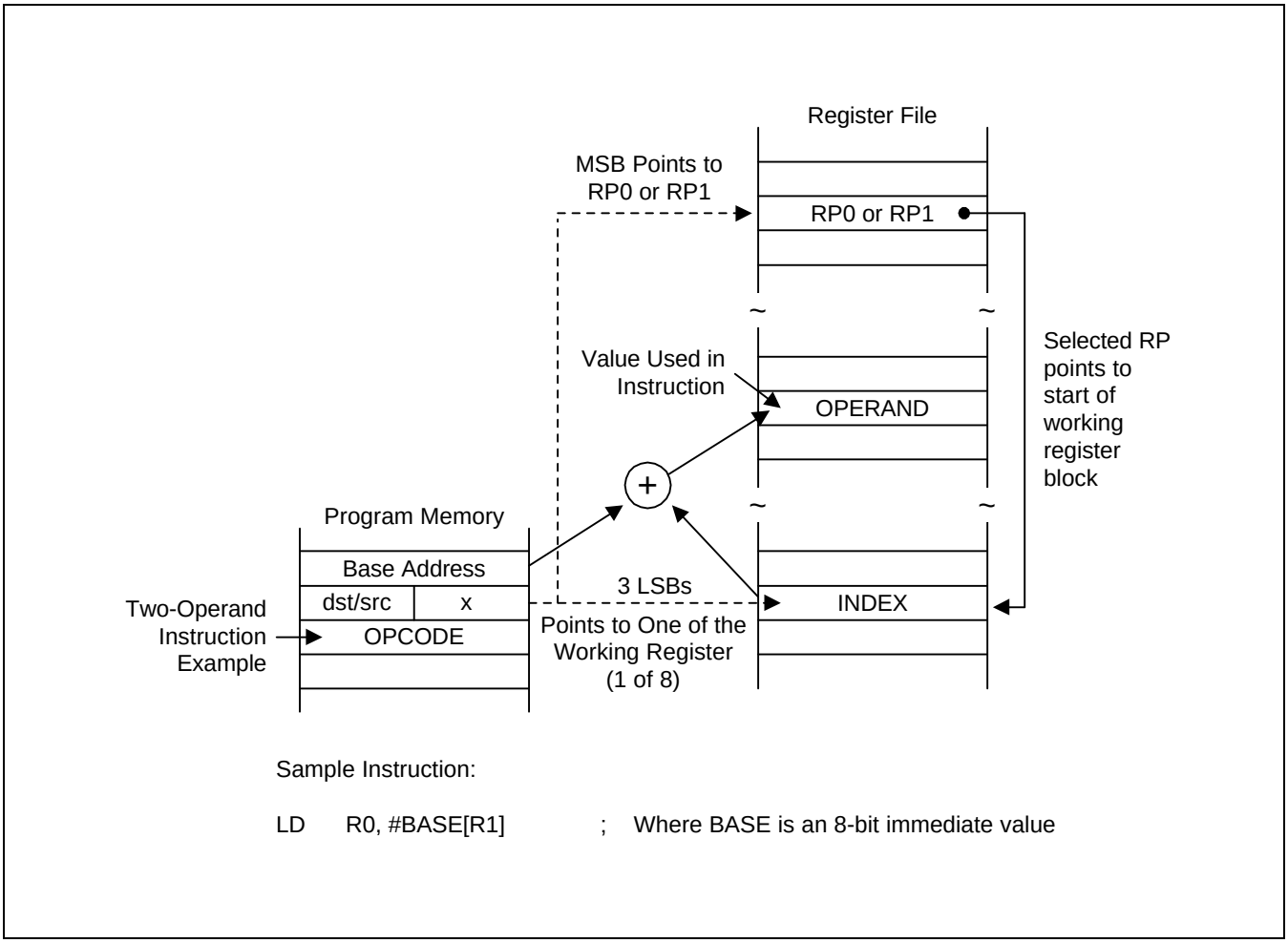
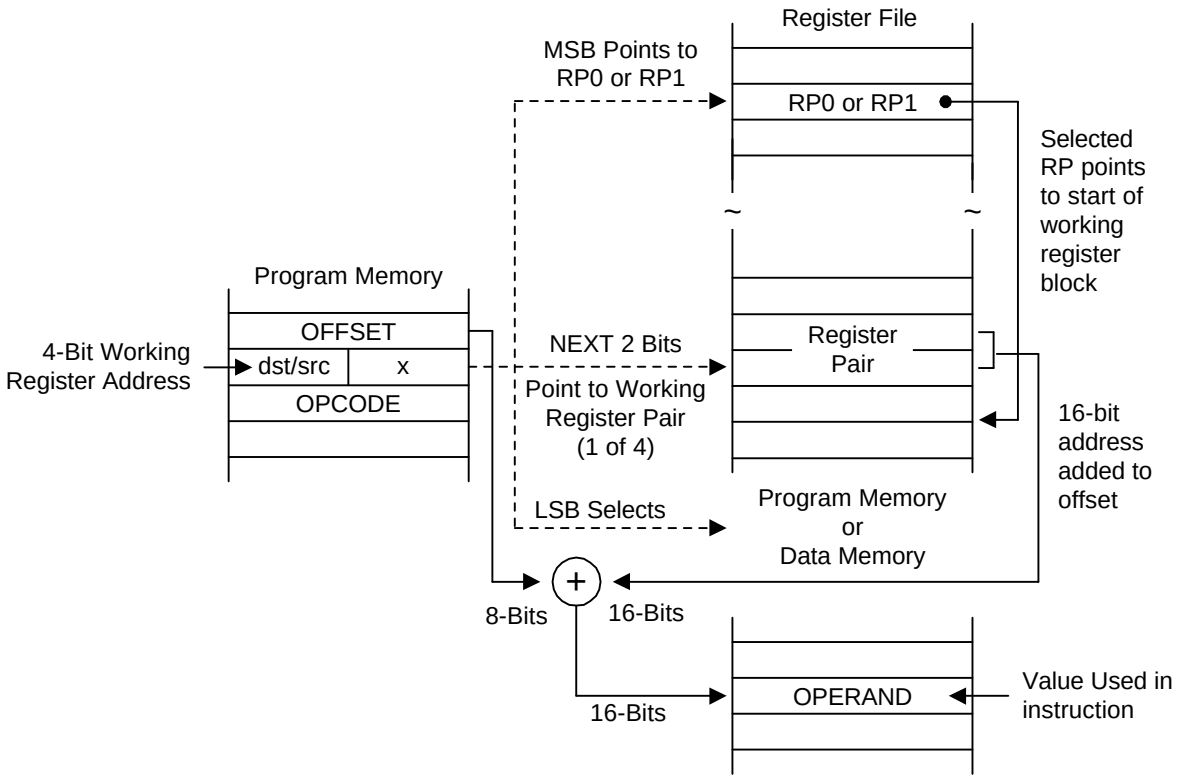


Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)

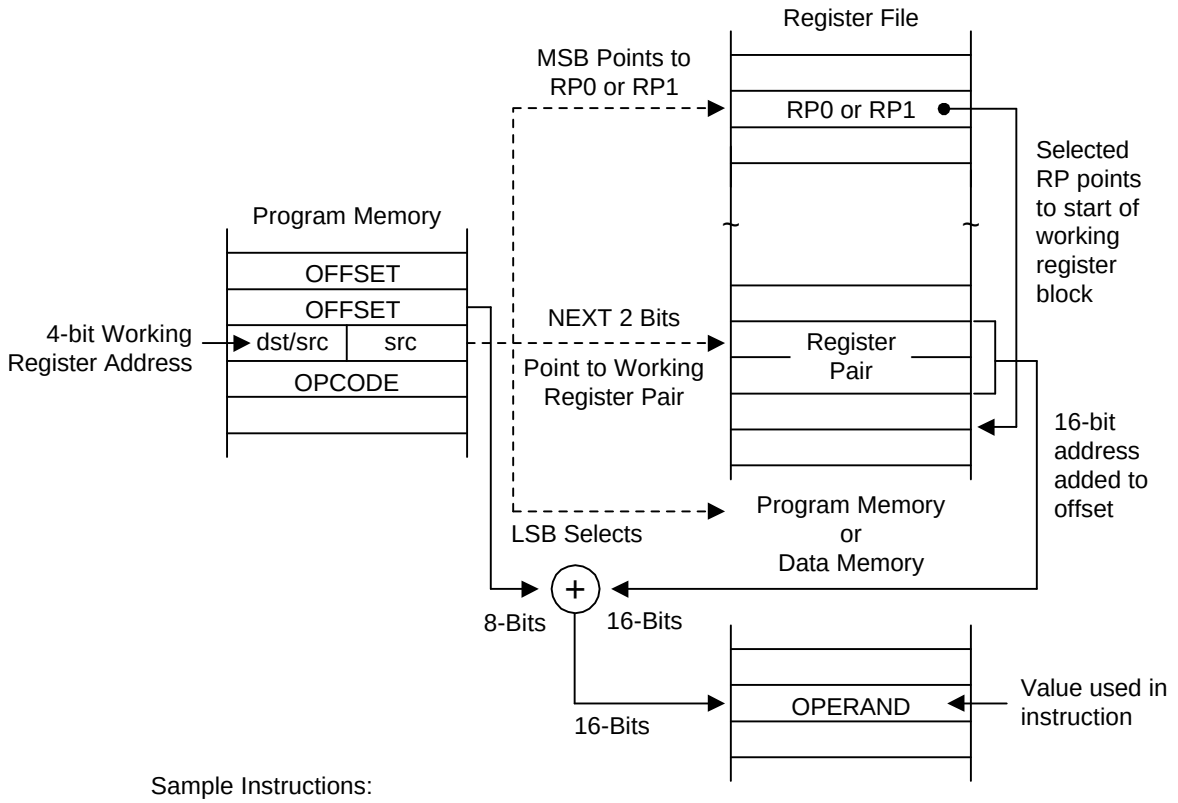


Sample Instructions:

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

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

INDEXED ADDRESSING MODE (Continued)



Sample Instructions:

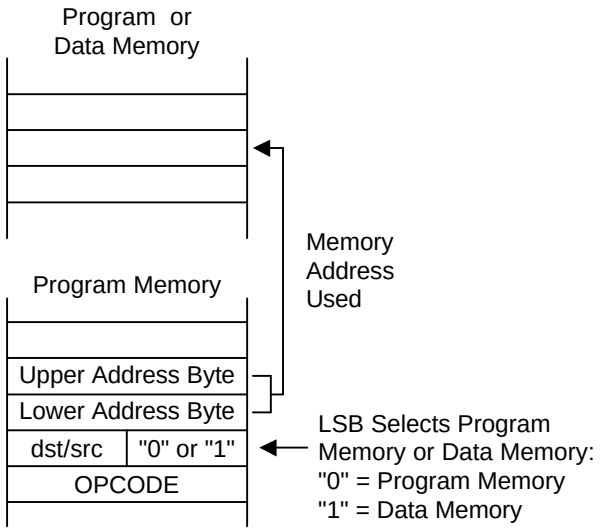
```
LDC     R4, #1000H[RR2]     ; The values in the program address (RR2 + 1000H)
                            ; are loaded into register R4.
LDE     R4, #1000H[RR2]     ; Identical operation to LDC example, except that
                            ; internal font/data memory and external data
       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

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

DIRECT ADDRESS MODE (DA)

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

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

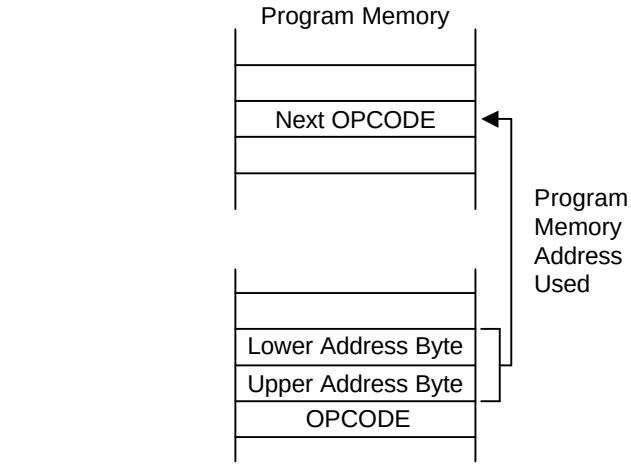


Sample Instructions:

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

Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)



Sample Instructions:

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

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

INDIRECT ADDRESS MODE (IA)

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

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

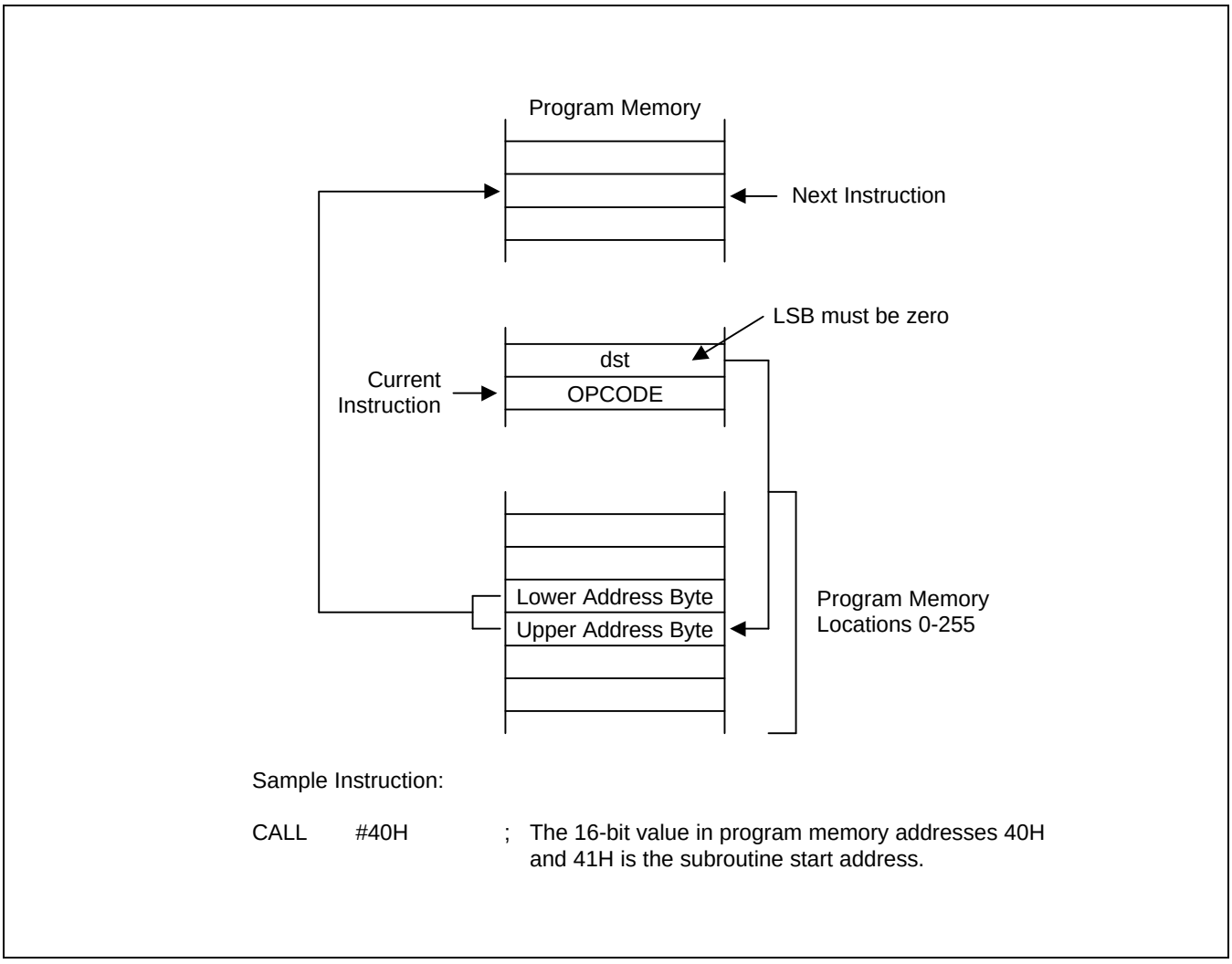


Figure 3-12. Indirect Addressing

RELATIVE ADDRESS MODE (RA)

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

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.

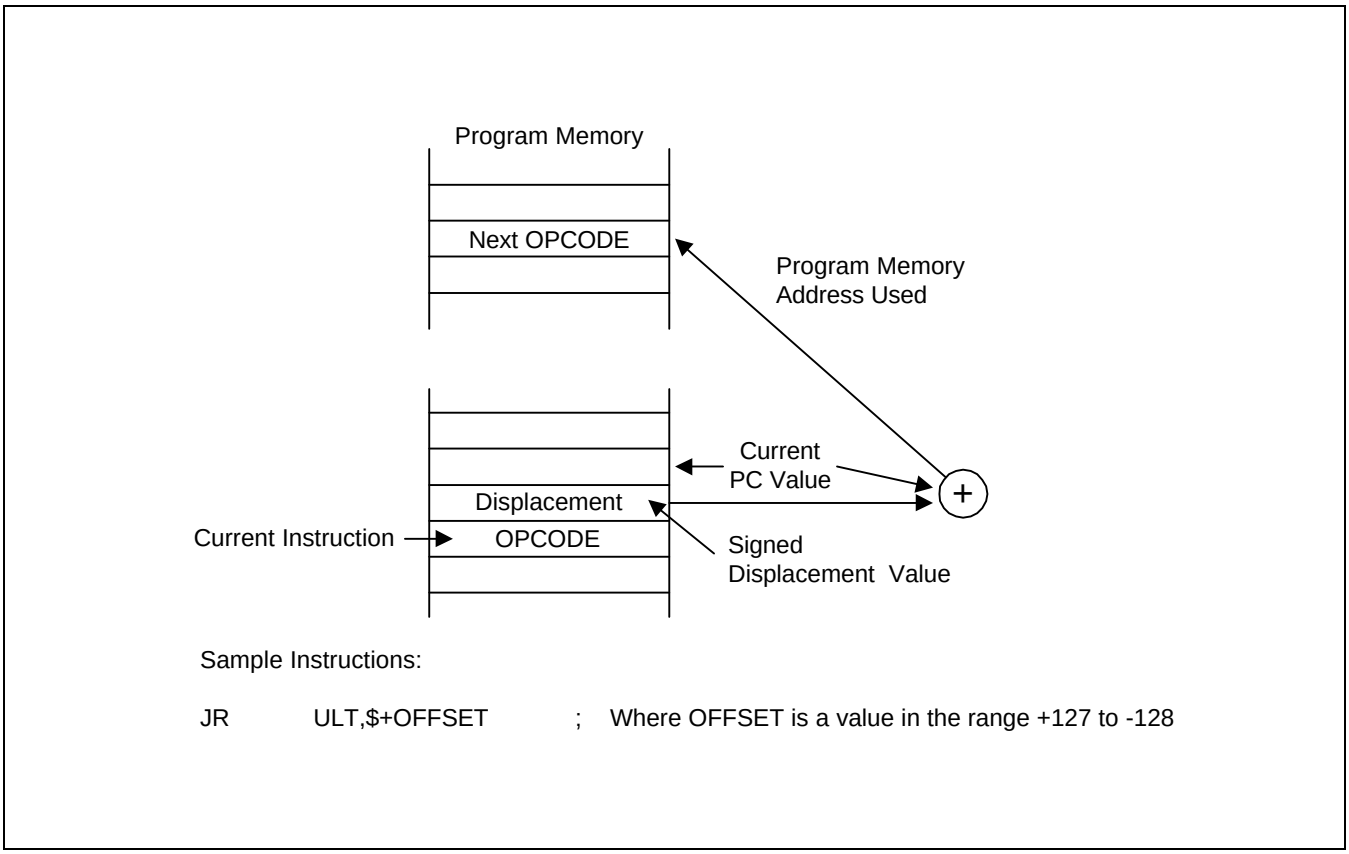


Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

In Immediate (IM) 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.

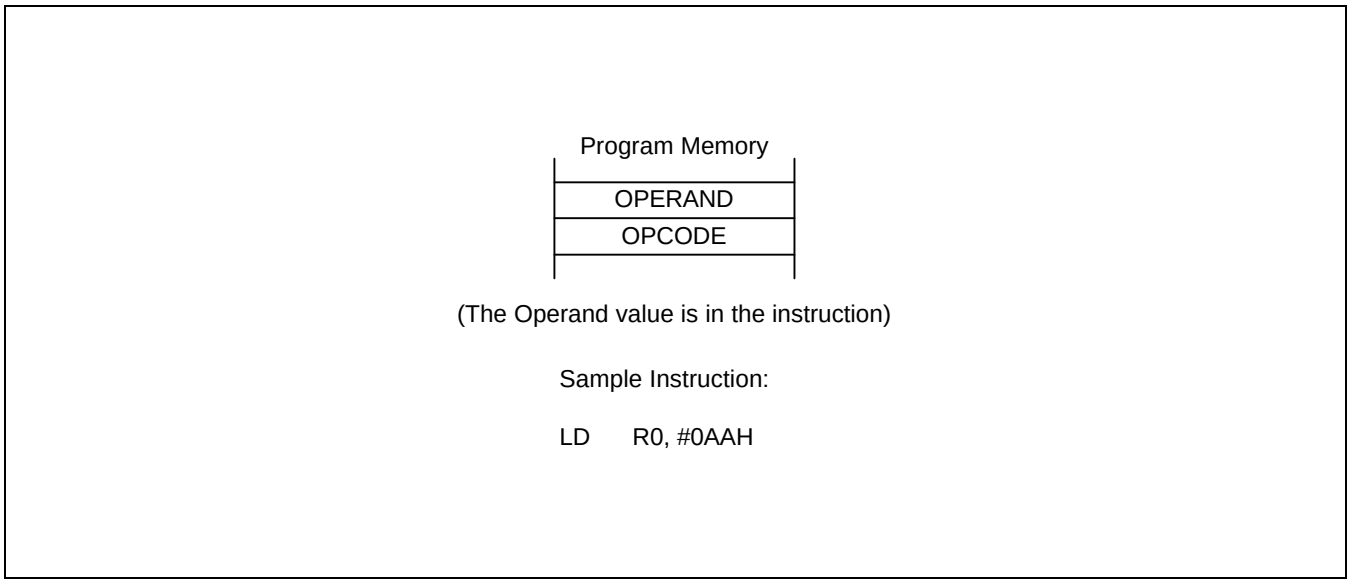


Figure 3-14. Immediate Addressing

4 CONTROL REGISTERS

OVERVIEW

In this chapter, detailed descriptions of the S3C820B 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 S3C820B register file are listed in Table 4-1. The hardware reset value for each mapped register is described in Chapter 8, “RESET and Power-Down”.

Table 4-1. Mapped Registers (Set 1)

Register Name	Mnemonic	Decimal	Hex	R/W
Timer 0 counter	T0CNT	208	D0H	R (note)
Timer 0 data register	T0DATA	209	D1H	R/W
Timer 0 control register	T0CON	210	D2H	R/W
Basic timer control register	BTCON	211	D3H	R/W
Clock control register	CLKCON	212	D4H	R/W
System flags register	FLAGS	213	D5H	R/W
Register pointer 0	RP0	214	D6H	R/W
Register pointer 1	RP1	215	D7H	R/W
Stack pointer (high-byte)	SPH	216	D8H	R/W
Stack pointer (low-byte)	SPL	217	D9H	R/W
Instruction pointer (high-byte)	IPH	218	DAH	R/W
Instruction pointer (low-byte)	IPL	219	DBH	R/W
Interrupt request register	IRQ	220	DCH	R (note)
Interrupt mask register	IMR	221	DDH	R/W
System mode register	SYM	222	DEH	R/W
Register page pointer	PP	223	DFH	R/W
Port 0 data register	P0	224	E0H	R/W
Port 1 data register	P1	225	E1H	R/W
Port 2 data register	P2	226	E2H	R/W
Port 3 data register	P3	227	E3H	R/W
Location E4H is not mapped.				
Port 0 control register	P0CON	229	E5H	R/W
Port 1 control register	P1CON	230	E6H	R/W
Location E7H is not mapped.				
Port 2 control register (high-byte)	P2CONH	232	E8H	R/W
Port 2 control register (low-byte)	P2CONL	233	E9H	R/W
Port 3 control register (high-byte)	P3CONH	234	EAH	R/W
Port 3 control register (low-byte)	P3CONL	235	EBH	R/W
Port 3 interrupt control register	P3INT	236	ECH	R/W
Port 3 interrupt pending register	P3PND	237	EDH	R/W
Port 3 interrupt state register	P3STA	238	EEH	R/W
Locations EFH–F1H are not mapped.				

Table 4-1. Mapped Registers (Continued)

Register Name	Mnemonic	Decimal	Hex	R/W
Timer A counter	TACNT	242	F2H	R (note)
Timer B counter	TBCNT	243	F3H	R (note)
Timer A data register	TADATA	244	F4H	R/W
Timer B data register	TBDATA	245	F5H	R/W
Timer A control register	TACON	246	F6H	R/W
Timer B control register	TBCON	247	F7H	R/W
Watch timer control register	WTCON	248	F8H	R/W
LCD control register	LCON	249	F9H	R/W
Memory control register	FDCON	250	FAH	R/W
Clock output mode register	CLKMOD	251	FBH	R/W
Location FCH is not mapped.				
Basic timer counter	BTCNT	253	FDH	R (note)
External memory timing register	EMT	254	FEH	R/W
Interrupt priority register	IPR	255	FFH	R/W

NOTE: You cannot use a read-only register as a destination for the instructions OR, AND, LD, or LDB.

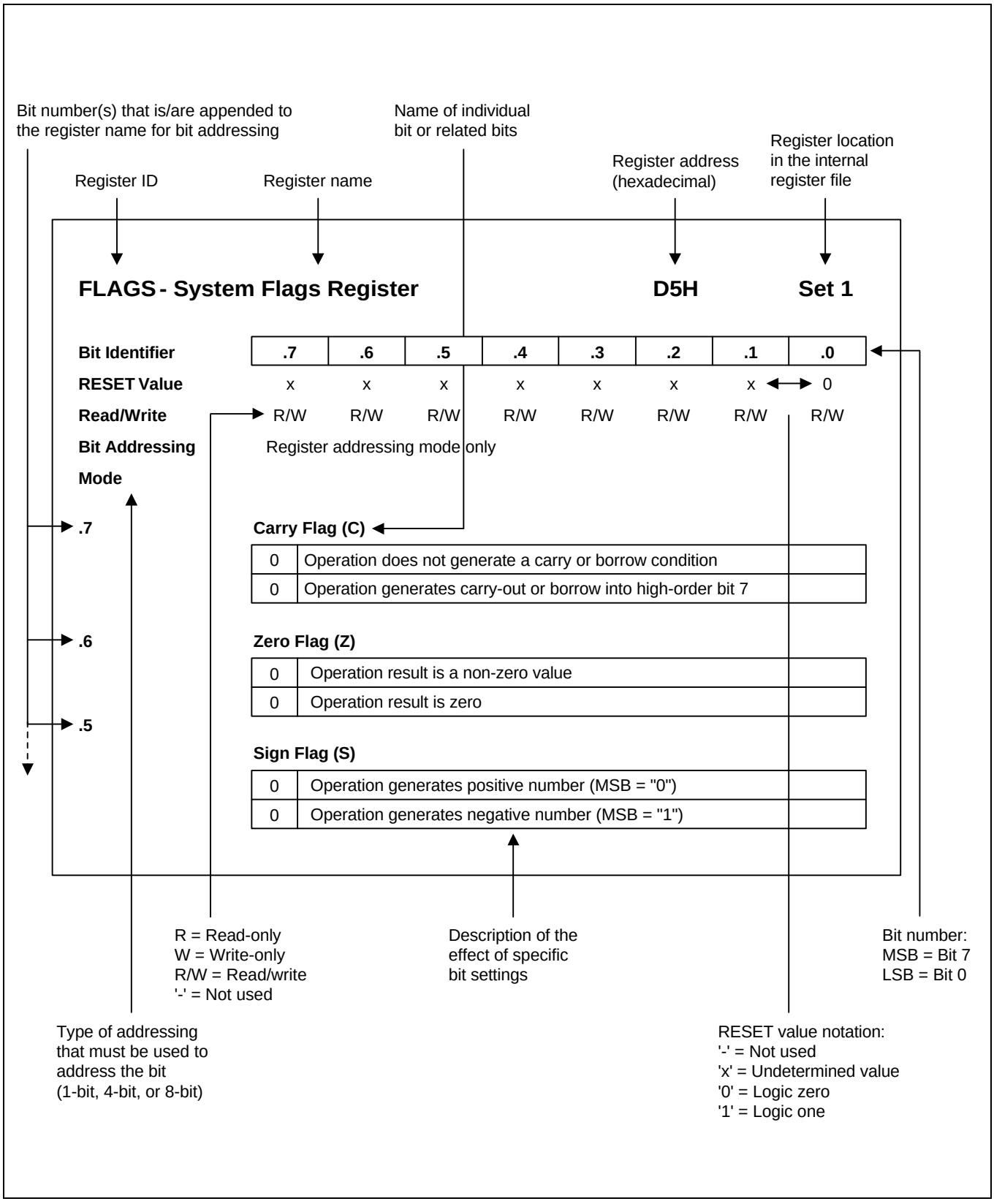


Figure 4-1. Register Description Format

BTCON — Basic Timer Control Register D3H Set 1

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

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

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

.3 and .2 Basic Timer Input Clock Selection Bits

0	0	fxx/4096 (3)
0	1	fxx/1024
1	0	fxx/128
1	1	fxx/16

.1 Basic Timer Counter Clear Bit (1)

0	No effect
1	Clear the basic timer counter value

.0 Clock Frequency Divider Clear Bit for Basic Timer and Timer 0 (2)

0	No effect
1	Clear both clock frequency dividers

NOTES:

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

CLKCON — System Clock Control Register D4H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	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 Oscillator IRQ Wake-up Function Enable Bit

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

.6 and .5 Main Oscillator Stop Control Bits

0	0	No effect
0	1	No effect
1	0	STOP main oscillator (fx)
1	1	No effect

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

0	0	fx/16, fxt (fx or fxt is selected clock for system)
0	1	fx/8, fxt (fx or fxt is selected clock for system)
1	0	fx/2, fxt (fx or fxt is selected clock for system)
1	1	fx (non-divided), fxt (fx or fxt is selected clock for system)

.2–.0 Subsystem Clock Selection Bits

1	0	1	Select sub clock (fxt) for system clock
Others			Select main clock (fx) for system clock

NOTES:

1. 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.
2. An external interrupt can be used to release stop mode and “wake up” the main oscillator.
3. When sub clock (fxt) is selected as system clock (fxx), you must use idle instruction instead of stop instruction for power down mode.
4. Although sub clock (fxt) is selected as system clock (fxx), the selected clock (fxx) of basic timer, timer counter 0/1, and clock output circuit is main clock if the value of CLKCON.6 –.5 is not “10B (value to stop main clock)”.

CLKMOD — Clock Output Mode Register FBH Set 1

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

.7 Clock Output Signal Enable Bit

0	Disable clock output signal
1	Enable clock output signal

.6–.4 Bits 6–4

0	Always logic zero

.3 and .2

Not used for S3C820B

.1 and .0 Output Clock Selection Bits

0	0	fxx
0	1	fxx/2^3
1	0	fxx/2^6
1	1	CPU clock output

EMT — External Memory Timing Register (note) FEH Set 1

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

.7–.2 Not used for S3C820B

.1 **Stack Area Selection Bit**

0	Select internal register file area
1	Select external data memory area

.0 Not used for S3C820B

NOTE: The EMT register is not used except EMT.1 register bit. The program initialization routine should clear the EMT register except the bit 1 to “00H” following a reset. Modification of EMT values during normal operation may cause a system malfunction.

FDCON — Memory Control Register FAH Set 1

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

.7–.3 **Bits 7–3**

0	Always logic zero

.2–.0 **Memory Selection Bits**

0	0	0	Font ROM 0
0	0	1	Font ROM 1
0	1	0	Font ROM 2
0	1	1	Internal data RAM
1	0	0	External data memory (DM signal active)
1	0	1	Invalid selection
1	1	0	Invalid selection
1	1	1	Invalid selection

NOTE: Refer to chapter 14 “Internal and External Memory”.

FLAGS — System Flags Register D5H Set 1

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

.7 Carry Flag (C)

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

.6 Zero Flag (Z)

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

.5 Sign Flag (S)

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

.4 Overflow Flag (V)

0	Operation result is ≤ +127 or ≥ –128
1	Operation result is > +127 or < –128

.3 Decimal Adjust Flag (D)

0	Add operation completed
1	Subtraction operation completed

.2 Half-Carry Flag (H)

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

.1 Fast Interrupt Status Flag (FIS)

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

.0 Bank Address Selection Flag (BA)

0	Bank 0 is selected (normal setting for S3C820B)
1	Invalid selection (bank 1 is not implemented)

IMR — Interrupt Mask Register DDH Set 1

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

.7 Not used for S3C820B

.6 Interrupt Level 6 (IRQ6) Enable Bit; External Interrupts P3.7–P3.4

0	Disable (mask)
1	Enable (un-mask)

.5 Interrupt Level 5 (IRQ5) Enable Bit; External Interrupts P3.3–P3.0

0	Disable (mask)
1	Enable (un-mask)

.4 Interrupt Level 4 (IRQ4) Enable Bit; Watch Timer Overflow

0	Disable (mask)
1	Enable (un-mask)

.3 and .2 Not used for S3C820B

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

0	Disable (mask)
1	Enable (un-mask)

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

0	Disable (mask)
1	Enable (un-mask)

NOTES:

1. When an interrupt level is masked, any interrupt requests that may be issued are not recognized by the CPU.
2. Interrupt levels IRQ2, IRQ3 and IRQ7 are not used in the S3C820B interrupt structure.

IPH — Instruction Pointer (High-Byte) DAH Set 1

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

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

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

IPL — Instruction Pointer (Low-Byte) DBH Set 1

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

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

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

IPR — Interrupt Priority Register FFH Set 1

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

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

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

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

0	IRQ6
1	IRQ6

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

0	IRQ5 > IRQ6
1	IRQ6 > IRQ5

.3 **Interrupt Subgroup B Priority Control Bit (note)**

0	IRQ4
1	IRQ4

.2 **Interrupt Group B Priority Control Bit (note)**

0	IRQ4
1	IRQ4

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

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

NOTE: The S3C820B interrupt structure uses only five levels: IRQ0, IRQ1, and IRQ4–IRQ6. Because IRQ2, IRQ3 and IRQ7 are not recognized, the interrupt B group and subgroup settings (IPR.2 and IPR.3) are not evaluated.

IRQ — Interrupt Request Register DCH Set 1

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

.7 Not used for S3C820B

.6 Level 6 (IRQ6) Request Pending Bit; External Interrupts P0.3–P0.0

0	Not pending
1	Pending

.5 Level 5 (IRQ5) Request Pending Bit; External Interrupts P2.3–P2.0

0	Not pending
1	Pending

.4 Level 4 (IRQ4) Request Pending Bit; Watch Timer Overflow Interrupt

0	Not pending
1	Pending

.3 and .2 Not used for S3C820B

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

0	Not pending
1	Pending

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

0	Not pending
1	Pending

NOTE: Interrupt levels IRQ2, IRQ3 and IRQ7 are not used in the S3C820B interrupt structure.

LCON — LCD Control Register F9H Set 1

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

.7 LCD Display Control Bit

0	Display off, disable voltage doubler
1	Normal display mode, enable voltage doubler

.6–.3

Not used for S3C820B

.2 and .1 LCD Clock Selection Bits

0	0	$fw/2^6$ (512 Hz when fw is 32.768 kHz)
0	1	$fw/2^5$ (1,024 Hz when fw is 32.768 kHz)
1	0	$fw/2^4$ (2,048 Hz when fw is 32.768 kHz)
1	1	$fw/2^3$ (4,096 Hz when fw is 32.768 kHz)

NOTE: fw is watch timer clock.

.0 LCD Dots Off Control Bit

0	All of the LCD dots off
1	Normal display mode

P0CON — Port 0 Control Register E5H Set 1

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

.7–.4 **Upper Nibble Port Configuration (P0.7/A15 – P0.4/A12)**

0	0	x	0	Schmitt trigger input
0	1	x	0	Schmitt trigger input, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open-drain
0	1	1	1	Output, open-drain, pull-up
1	x	x	x	External interface (A12–A15)

.3–.0 **Low Nibble Port Configuration (P0.3/A11–P0.0/A8)**

0	0	x	0	Schmitt trigger input
0	1	x	0	Schmitt trigger input, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open-drain
0	1	1	1	Output, open-drain, pull-up
1	x	x	x	External interface (A8–A11)

NOTE: “x” means don’t care.

P1CON — Port 1 Control Register E6H Set 1

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

.7–.4 Upper Nibble Port Configuration (P1.7/AD7–P1.4/AD4)

0	0	x	0	Schmitt trigger input
0	1	x	0	Schmitt trigger input, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open-drain
0	1	1	1	Output, open-drain, pull-up
1	x	x	x	External interface (AD7–AD4)

.3–.0 Low Nibble Port Configuration (P1.3/AD3–P1.0/AD0)

0	0	x	0	Schmitt trigger input
0	1	x	0	Schmitt trigger input, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open-drain
0	1	1	1	Output, open-drain, pull-up
1	x	x	x	External interface (AD3–AD0)

NOTE: “x” means don’t care.

P2CONH — Port 2 Control Register (High-Byte) E8H Set 1

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

.7 and .6 **P2.7/BUZ Mode Selection Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	0	Push-pull output mode
1	1	Select alternate function for P2.7 (BUZ signal)

.5 and .4 **P2.6/CLO Mode Selection Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	0	Push-pull output mode
1	1	Select alternate function for P2.6 (CLO clock)

.3 and .2 **P2.5/T0CK Mode Selection Bits**

0	0	Schmitt trigger input mode (T0CK clock)
0	1	Schmitt trigger input, pull-up mode (T0CK clock)
1	0	Push-pull output mode
1	1	P2.5 remains at “0” state

.1 and .0 **P2.4/T0 Mode Selection Bits**

0	0	Schmitt trigger input mode (T0CAP)
0	1	Schmitt trigger input, pull-up mode (T0CAP)
1	0	Push-pull output mode
1	1	Select alternate function for P2.4 (T0)

P2CONL — Port 2 Control Register (Low-Byte) E9H Set 1

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

.7 and .6

P2.3/DM Mode Selection Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	0	Push-pull output mode
1	1	External memory interface line (DM signal)

.5 and .4

P2.2/DR Mode Selection Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	0	Push-pull output mode
1	1	External memory interface line (DR signal)

.3 and .2

P2.1/DW Mode Selection Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	0	Push-pull output mode
1	1	External memory interface line (DW signal)

.1 and .0

P2.0/AS Mode Selection Bits

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	0	Push-pull output mode
1	1	External memory interface line (AS signal)

P3CONH — Port 3 Control Register (High-Byte) EAH Set 1

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

.7 and .6 **P3.7/INT7 Mode Selection Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	x	Push-pull output mode

.5 and .4 **P3.6/INT6 Mode Selection Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	x	Push-pull output mode

.3 and .2 **P3.5/INT5 Mode Selection Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	x	Push-pull output mode

.1 and .0 **P3.4/INT4 Mode Selection Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	x	Push-pull output mode

NOTE: "x" means don't care.

P3CONL — Port 3 Control Register (Low-Byte) EBH Set 1

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

.7 and .6 **P3.3/INT3 Mode Selection Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	0	Push-pull output mode
1	1	P3.3 remains at “0” state

.5 and .4 **P3.2/INT2/T1CK Mode Selection Bits**

0	0	Schmitt trigger input mode (T1CK)
0	1	Schmitt trigger input, pull-up mode (T1CK)
1	0	Push-pull output mode
1	1	P3.2 remains at “0” state

.3 and .2 **P3.1/INT/TA Mode Selection Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	0	Push-pull output mode
1	1	Select alternate function for P3.1 (TA clock)

.1 and .0 **P3.0/INT0/TB Mode Selection Bits**

0	0	Schmitt trigger input mode
0	1	Schmitt trigger input, pull-up mode
1	0	Push-pull output mode
1	1	Select alternate function for P3.0 (TB clock)

P3INT — Port 3 External Interrupt Enable Register ECH Set 1

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

.7 P3.7 External Interrupt (INT7) Enable Bit

0	Disable interrupt
1	Enable interrupt

.6 P3.6 External Interrupt (INT6) Enable Bit

0	Disable interrupt
1	Enable interrupt

.5 P3.5 External Interrupt (INT5) Enable Bit

0	Disable interrupt
1	Enable interrupt

.4 P3.4 External Interrupt (INT4) Enable Bit

0	Disable interrupt
1	Enable interrupt

.3 P3.3 External Interrupt (INT3) Enable Bit

0	Disable interrupt
1	Enable interrupt

.2 P3.2 External Interrupt (INT2) Enable Bit

0	Disable interrupt
1	Enable interrupt

.1 P3.1 External Interrupt (INT1) Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 P3.0 External Interrupt (INT0) Enable Bit

0	Disable interrupt
1	Enable interrupt

P3PND — Port 3 External Interrupt Pending Register EDH Set 1

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

.7 P3.7 External Interrupt (INT7) Pending Flag

0	No P3.7 external interrupt pending (when read)
1	P3.7 external interrupt is pending (when read)

.6 P3.6 External Interrupt (INT6) Pending Flag

0	No P3.6 external interrupt pending (when read)
1	P3.6 external interrupt is pending (when read)

.5 P3.5 External Interrupt (INT5) Pending Flag

0	No P3.5 external interrupt pending (when read)
1	P3.5 external interrupt is pending (when read)

.4 P3.4 External Interrupt (INT4) Pending Flag

0	No P3.4 external interrupt pending (when read)
1	P3.4 external interrupt is pending (when read)

.3 P3.3 External Interrupt (INT3) Pending Flag

0	No P3.3 external interrupt pending (when read)
1	P3.3 external interrupt is pending (when read)

.2 P3.2 External Interrupt (INT2) Pending Flag

0	No P3.2 external interrupt pending (when read)
1	P3.2 external interrupt is pending (when read)

.1 P3.1 External Interrupt (INT1) Pending Flag

0	No P3.1 external interrupt pending (when read)
1	P3.1 external interrupt is pending (when read)

.0 P3.0 External Interrupt (INT0) Pending Flag

0	No P3.0 external interrupt pending (when read)
1	P3.0 external interrupt is pending (when read)

NOTE: To clear an interrupt pending condition, write a “0” to the appropriate pending flag. Writing a “1” to an interrupt pending flag (P3PND.0–.7) has no effect.

P3STA — Interrupt State Register EEH Set 1

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

.7 P3.7 External Interrupt (INT7) State Bit

0	Falling edge detection
1	Rising edge detection

.6 P3.6 External Interrupt (INT6) State Bit

0	Falling edge detection
1	Rising edge detection

.5 P3.5 External Interrupt (INT5) State Bit

0	Falling edge detection
1	Rising edge detection

.4 P3.4 External Interrupt (INT4) State Bit

0	Falling edge detection
1	Rising edge detection

.3 P3.3 External Interrupt (INT3) State Bit

0	Falling edge detection
1	Rising edge detection

.2 P3.2 External Interrupt (INT2) State Bit

0	Falling edge detection
1	Rising edge detection

.1 P3.1 External Interrupt (INT1) State Bit

0	Falling edge detection
1	Rising edge detection

.0 P3.0 External Interrupt (INT0) State Bit

0	Falling edge detection
1	Rising edge detection

PP — Register Page Pointer DFH Set 1

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

.7–.4 Destination Register Page Selection Bits

0	0	0	0	Destination: page 0 (note)
0	0	0	1	Destination: page 1 (note)
0	0	1	0	Destination: page 2 (note)

.3–.0 Source Register Page Selection Bits

0	0	0	0	Source: page 0 (note)
0	0	0	1	Source: page 1 (note)
0	0	1	0	Source: page 2 (note)

NOTE: In the S3C820B microcontroller, the internal register file is configured as three pages (Pages 0–2). The page 0 is used for general purpose register file, and pages 1–2 are used for LCD data or general purpose registers.

RP0 — Register Pointer 0 D6H Set 1

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

.7–.3 Register Pointer 0 Address Value

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

.2–.0 Not used for S3C820B

RP1 — Register Pointer 1 D7H Set 1

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

.7–.3 Register Pointer 1 Address Value

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

.2–.0 Not used for S3C820B

SPH — Stack Pointer (High-Byte) D8H Set 1

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

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

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

SPL — Stack Pointer (Low-Byte) D9H Set 1

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

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

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

SYM — System Mode Register DEH Set 1

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

.7 Tri-State External Interface Control Bit

0	Normal operation (disable tri-state operation)
1	Set external interface lines to high impedance (enable tri-state operation)

.6 and .5

Not used for S3C820B

.4–.2 Fast Interrupt Level Selection Bits (1)

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	Not used for S3C820B
0	1	1	Not used for S3C820B
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	Not used for S3C820B

.1 Fast Interrupt Enable Bit (2)

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0 Global Interrupt Enable Bit (3)

0	Disable global interrupt processing
1	Enable global 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. Although you can manipulate SYM.0 directly to enable or disable interrupts, We recommend that you use the EI and DI instructions insteadly.

T0CON — Timer 0 Control Register D2H Set 1

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

.7 and .6 Timer 0 Input Clock Selection Bits

0	0	fxx/4,096
0	1	fxx/256
1	0	fxx/8
1	1	External clock input (at the T0CK pin, P2.5)

.5 and .4 Timer 0 Operating Mode Selection Bits

0	0	Interval timer mode (counter cleared by match signal)
0	1	Capture mode (rising edges, counter running, OVF interrupt can occur)
1	0	Capture mode (falling edges, counter running, OVF interrupt can occur)
1	1	PWM mode (OVF interrupt can occur)

.3 Timer 0 Counter Clear Bit

0	No effect (when write)
1	*Clear T0 counter, T0CNT (when write)*

.2 Timer 0 Overflow Interrupt Enable Bit (note)

0	Disable T0 overflow interrupt
1	Enable T0 overflow interrupt

.1 Timer 0 Match/Capture Interrupt Enable Bit

0	Disable T0 match/capture interrupt
1	Enable T0 match/capture interrupt

.0 Timer 0 Match/Capture Interrupt Pending Flag

0	No T0 match/capture interrupt pending (when read)
0	*Clear T0 match/capture interrupt pending condition (when write)*
1	T0 match/capture interrupt is pending (when read)
1	*No effect (when write)*

NOTE: A timer 0 overflow interrupt pending condition is automatically cleared by hardware. However, the timer 0 match/capture interrupt, IRQ0, vector FCH, must be cleared by the interrupt service routine.

TACON — Timer Counter 1/A Control Register F6H Set 1

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

.7 Timer Counter 1 Operating Mode Selection Bit

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

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

0	0	0	fxx/1,024
0	0	1	fxx/512
0	1	0	fxx/8
0	1	1	fxx
1	x	x	T1CK (External clock at P3.2)

.3 Timer 1/A Counter Clear Bit

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

.2 Timer 1/A Counter Run Enable Bit

0	Disable Counter Running
1	Enable Counter Running

.1 Timer Counter 1/A Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer Counter 1/A Interrupt Pending Bit

0	No interrupt pending (when read)
0	Clear interrupt pending condition (when write)
1	Timer counter 1/A interrupt is pending (when read)
1	No effect (when write)

TBCON — Timer Counter B Control Register F7H Set 1

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

.7 and .6

Not used for S3C820B

.5 and .4 Timer Counter B Clock Selection Bits

0	0	fxx/1,024
0	1	fxx/512
1	0	fxx/8
1	1	fxx

.3 Timer B Counter Clear Bit

0	No effect
1	Clear Timer B counter (when write)

.2 Timer B Counter Run Enable Bit

0	Disable Counter Running
1	Enable Counter Running

.1 Timer Counter B Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Timer Counter B Interrupt Pending Flag

0	No interrupt pending (when read)
0	Clear interrupt pending condition (when write)
1	Timer counter B interrupt is pending (when read)
1	No effect (when write)

WTCON — Watch Timer Control Register F8H Set 1

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

.7 Buzzer Signal Enable Bit

0	Disable buzzer signal output
1	Enable buzzer signal output

.6

0	Always logic zero

.5 and .4 Buzzer Signal Selection Bits

0	0	2 kHz
0	1	4 kHz
1	0	8 kHz
1	1	16 kHz

.3 Watch Timer Speed Selection Bit

0	1 s interval when fw is 32.768 kHz
1	3.91 ms interval when fw is 32.768 kHz

.2 Watch Timer Clock Selection Bit

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

.1 Watch Timer 1 s Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

.0 Watch Timer 1 s or 3.91 ms Interrupt Enable Bit

0	Disable interrupt
1	Enable interrupt

NOTE: When main clock is selected as system clock (fxx), watch timer 1 s/3.91 ms interrupt is used as for 3.91 ms interrupt (WTCON.3 = "1"), and sub clock is selected as watch timer clock, watch timer's 3.91 ms interval interrupt is generated two times at the same time. (If fx ≠ fxx, WTCON.3 ≠ "1", nor WTCON.2 ≠ "1", the interrupt is generated normally.)
That is, the interrupt is generated once more after executing the interrupt routine. Refer to page 17-1 for using watch timer's 3.91 ms interrupt.

5 INTERRUPT STRUCTURE

OVERVIEW

The S3C8-series interrupt structure has three basic components: levels, vectors, and sources. The SAM87 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 S3C820B interrupt structure recognizes five interrupt levels.

The interrupt level numbers 0 through 7 do not necessarily indicate the relative priority of the levels. They are simply 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. The S3C820B uses fifteen vectors.

Sources

A source is any peripheral that generates an interrupt. A source can be an external pin or a counter overflow, for example. Each vector can have several interrupt sources. In the S3C820B interrupt structure, there are fifteen possible interrupt sources.

When a service routine starts, the respective pending bit is either cleared automatically by hardware or is must be cleared “manually” by program software. The characteristics of the source’s pending mechanism determine which method is used to clear its respective pending bit.

INTERRUPT TYPES

The three components of the S3C8 interrupt structure described above — 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 S3C820B microcontroller, two interrupt types are implemented.

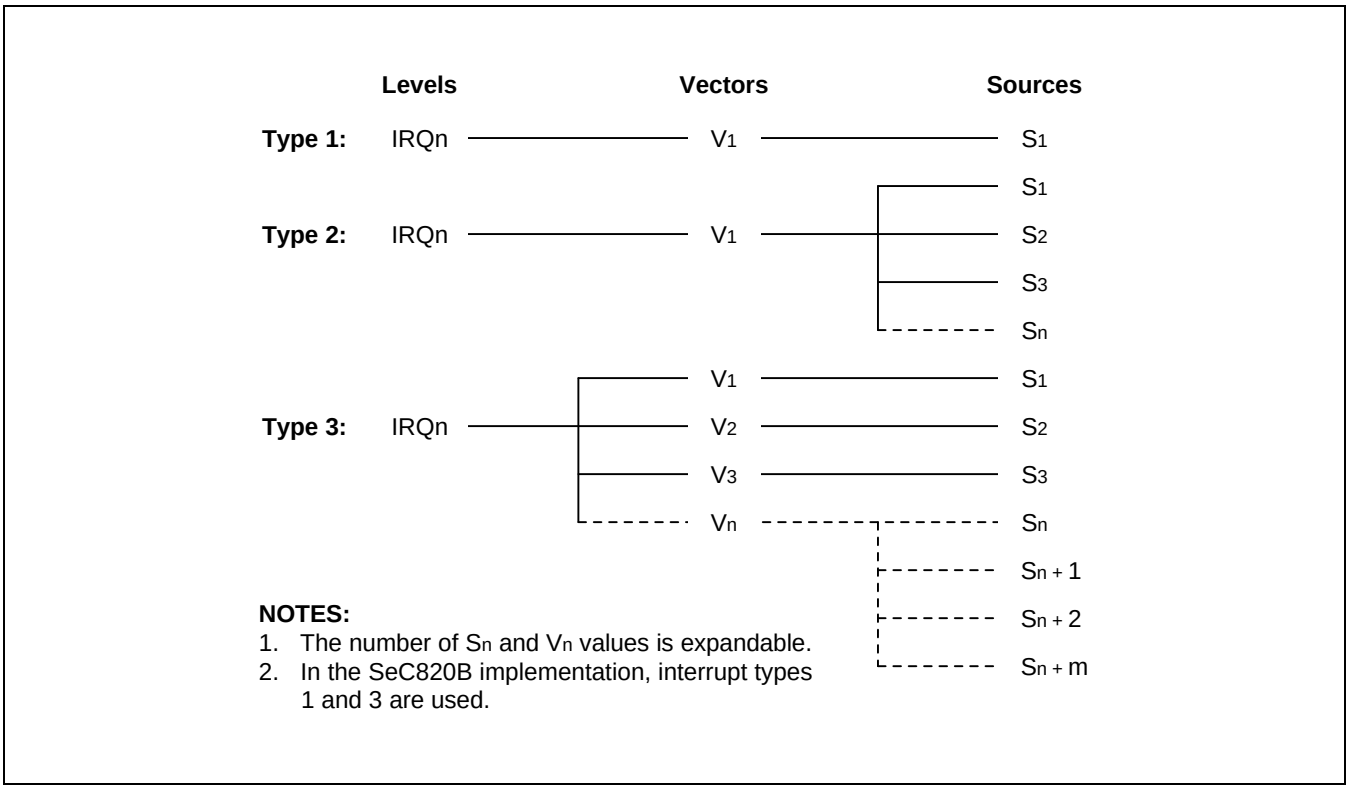


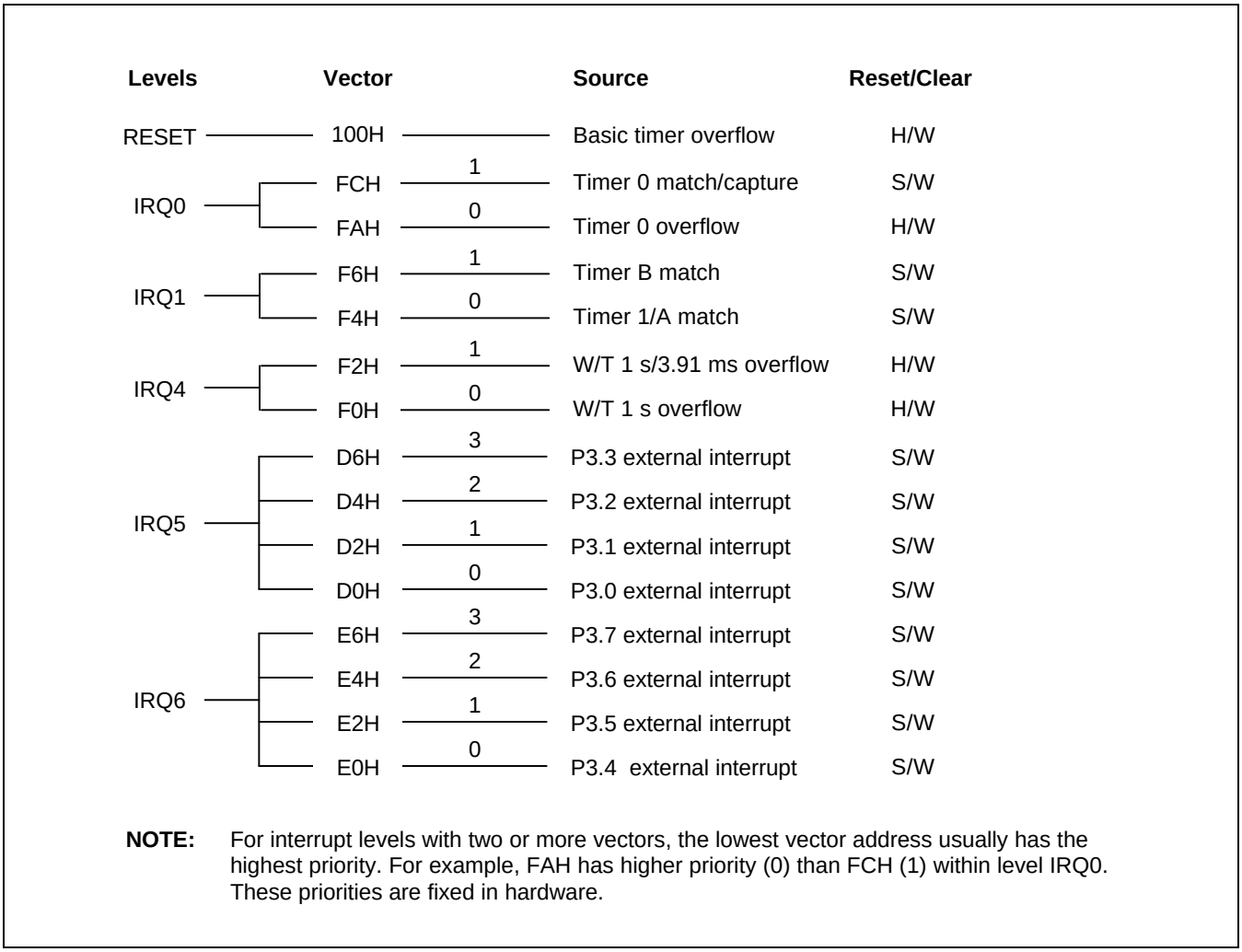
Figure 5-1. S3C8-Series Interrupt Types

S3C820B INTERRUPT STRUCTURE

The S3C820B microcontroller supports fifteen interrupt sources. All fifteen of the interrupt sources have a corresponding interrupt vector address. Five 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.



NOTE: For interrupt levels with two or more vectors, the lowest vector address usually has the highest priority. For example, FAH has higher priority (0) than FCH (1) within level IRQ0. These priorities are fixed in hardware.

Figure 5-2. S3C820B Interrupt Structure

INTERRUPT VECTOR ADDRESSES

All interrupt vector addresses for the S3C820B interrupt structure are stored in the vector address area of the internal 64K-byte ROM, 0H–FFFFH (see Figure 5-3).

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

The program reset address in the ROM is 0100H.

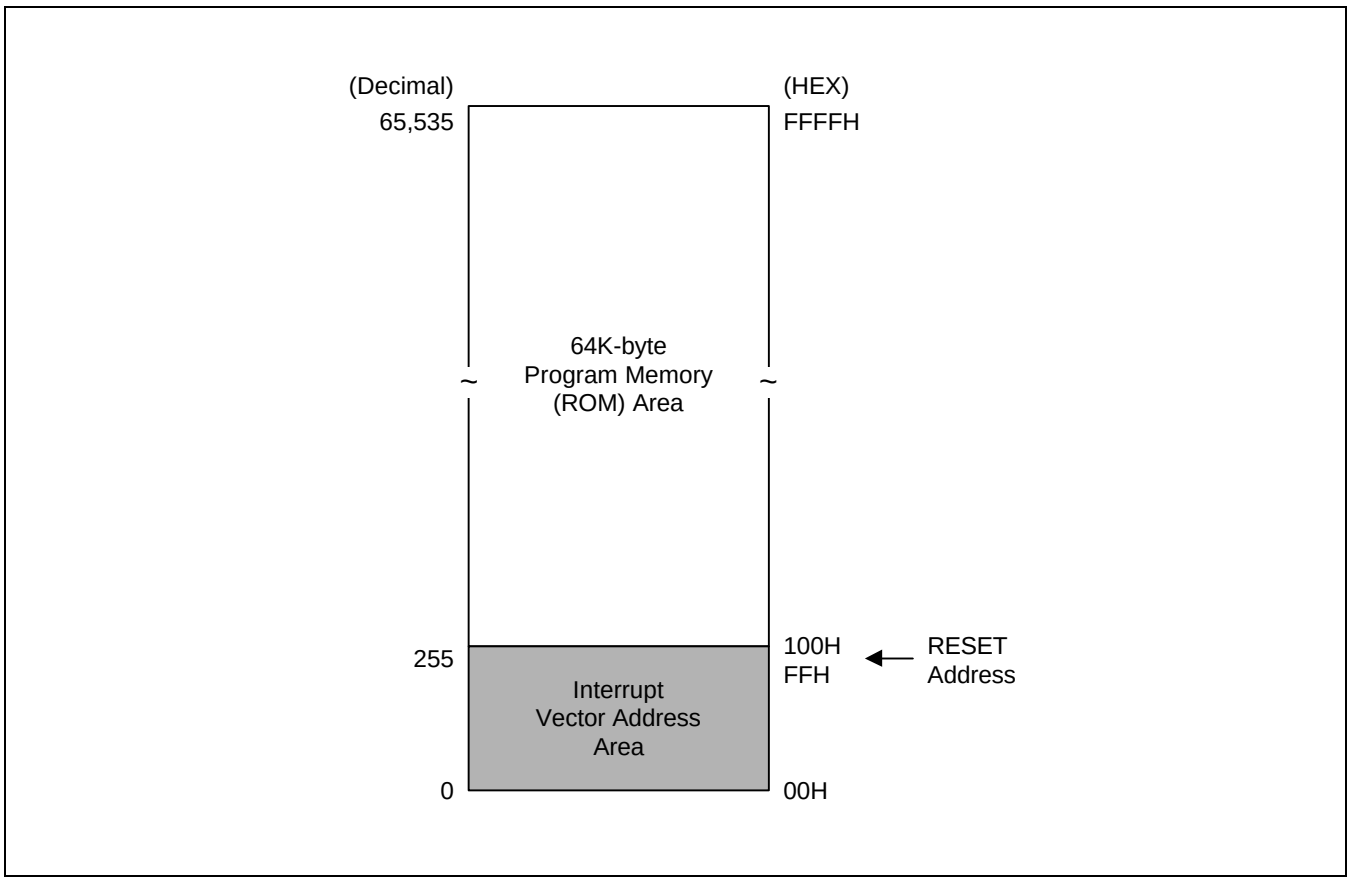


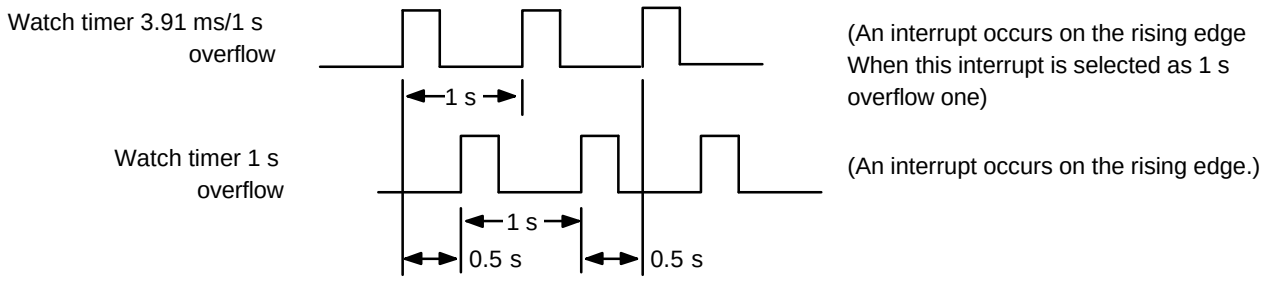
Figure 5-3. ROM Vector Address Area

Table 5-1. S3C820B Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
254	100H	Basic timer overflow	RESET	–	√	
252	FCH	Timer 0 (match/capture)	IRQ0	1		√
250	FAH	Timer 0 overflow		0	√	
246	F6H	Timer B match	IRQ1	1		√
244	F4H	Timer 1/A match		0		√
242	F2H	Watch timer 3.91 ms/1 s overflow	IRQ4	1	√	
240	F0H	Watch timer 1 s overflow		0	√	
230	E6H	P3.7 external interrupt	IRQ6	3		√
228	E4H	P3.6 external interrupt		2		√
226	E2H	P3.5 external interrupt		1		√
224	E0H	P3.4 external interrupt		0		√
214	D6H	P3.3 external interrupt	IRQ5	3		√
212	D4H	P3.2 external interrupt		2		√
210	D2H	P3.1 external interrupt		1		√
208	D0H	P3.0 external interrupt		0		√

NOTES:

1. Interrupt priorities are identified in inverse order: “0” is highest priority, “1” is the next highest, and so on.
2. If two or more interrupts within the same level contend, the interrupt with the lowest vector address usually has priority over one with a higher vector address. The priorities within a given level are fixed in hardware.
3. Two 1s-overflow interrupts of watch timer is generated in turn. One interrupt of two is generated, and the other interrupt is generated after 0.5 second. Refer to the following figure.



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, and according to the established priorities.

NOTE

Although you can manipulate SYM.0 directly to enable or disable interrupts, we recommend that you use the EI and DI instructions instead.

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

SYSTEM-LEVEL INTERRUPT CONTROL REGISTERS

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

— The interrupt mask register, IMR, enables (un-masks) or disables (masks) interrupt levels.

— The interrupt priority register, IPR, controls the relative priorities of interrupt levels.

— The interrupt request register, IRQ, contains interrupt pending flags for each interrupt level (as opposed to each interrupt source).

— The system mode register, SYM, enables or disables global interrupt processing (SYM settings also enable fast interrupts and control the activity of external interface, if implemented).

Table 5-2. Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable or disable interrupt processing for each of the five interrupt levels: IRQ0, IRQ1, and IRQ4–IRQ6.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The five levels of the S3C820B are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, group B is IRQ4, and group C is IRQ5 and IRQ6.
Interrupt request register	IRQ	R	This register contains a request pending bit for each interrupt level.
System mode register	SYM	R/W	Dynamic global interrupt processing enable/disable, fast interrupt processing, and external interface control.

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, therefore:

— 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 the part of your application program that handles interrupt processing, be sure to include the necessary register file address (register pointer) information.

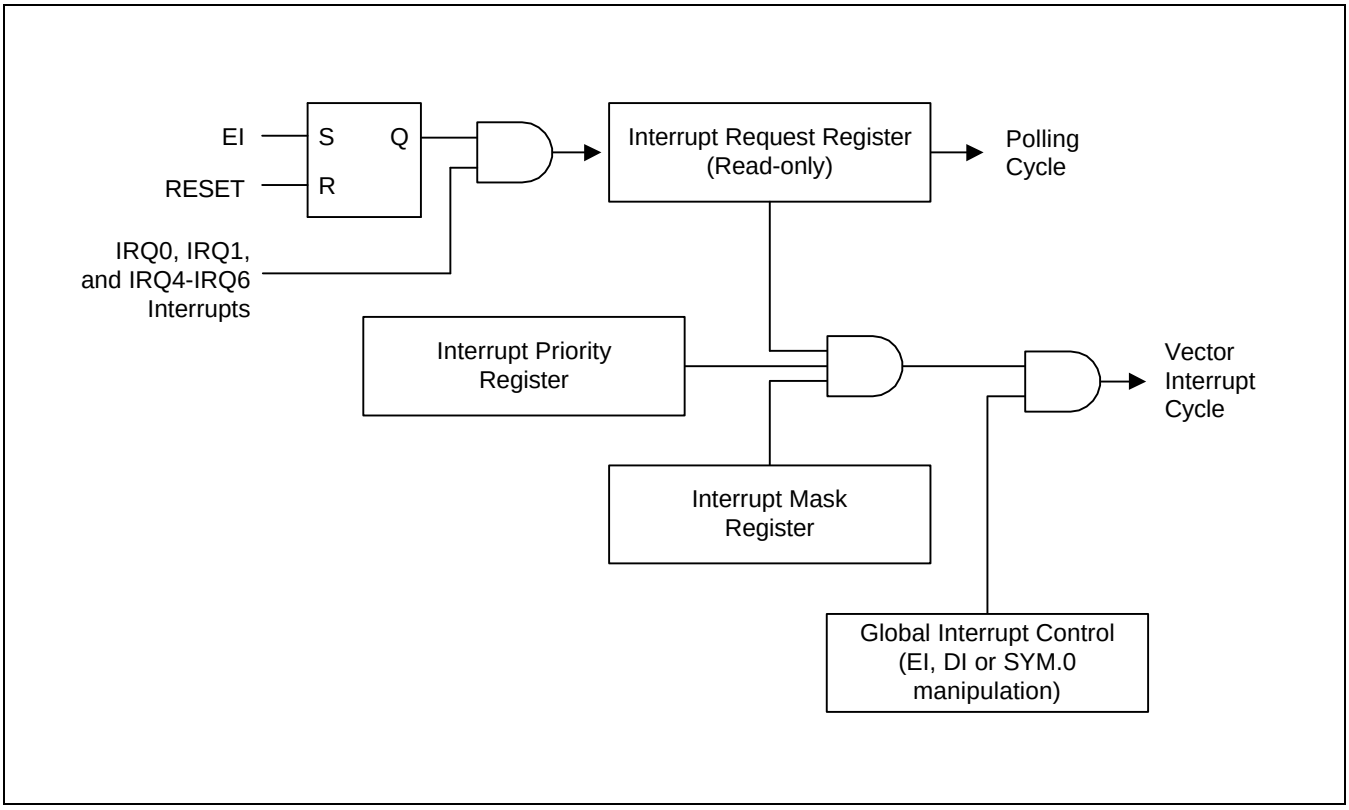


Figure 5-4. Interrupt Function Diagram

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 that peripheral (see Table 5-3).

Table 5-3. Interrupt Source Control and Data Registers

Interrupt Source	Interrupt Level	Register(s)	Location(s) in Set 1
Timer 0 match/capture or timer 0 overflow	IRQ0	T0CON (note) T0DATA	D2H D1H
Timer 1 match (Timer A, B)	IRQ1	TACON, TBCON TADATA, TBDATA	F6H, F7H F4H, F5H
Watch timer 1s overflow, 3.91 ms/1 s overflow	IRQ4	WTCON	F8H
P3.3 external interrupt P3.2 external interrupt P3.1 external interrupt P3.0 external interrupt	IRQ5	P3CONL P3INT P3PND P3STA	EBH ECH EDH EEH
P3.7 external interrupt P3.6 external interrupt P3.5 external interrupt P3.4 external interrupt	IRQ6	P3CONH P3INT P3PND P3STA	EAH ECH EDH EEH

NOTE: Because the timer 0 and watch timer overflow interrupts are cleared by hardware, the T0CON and WTCON registers control only the enable/disable functions. The T0CON register contains enable/disable and pending bit for the timer 0 match/capture interrupt.

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.7, SYM.1, and SYM.0 to “0”. The reset 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. Although you can manipulate SYM.0 directly to enable and disable interrupts, we recommend using the EI and DI instructions for this purpose.

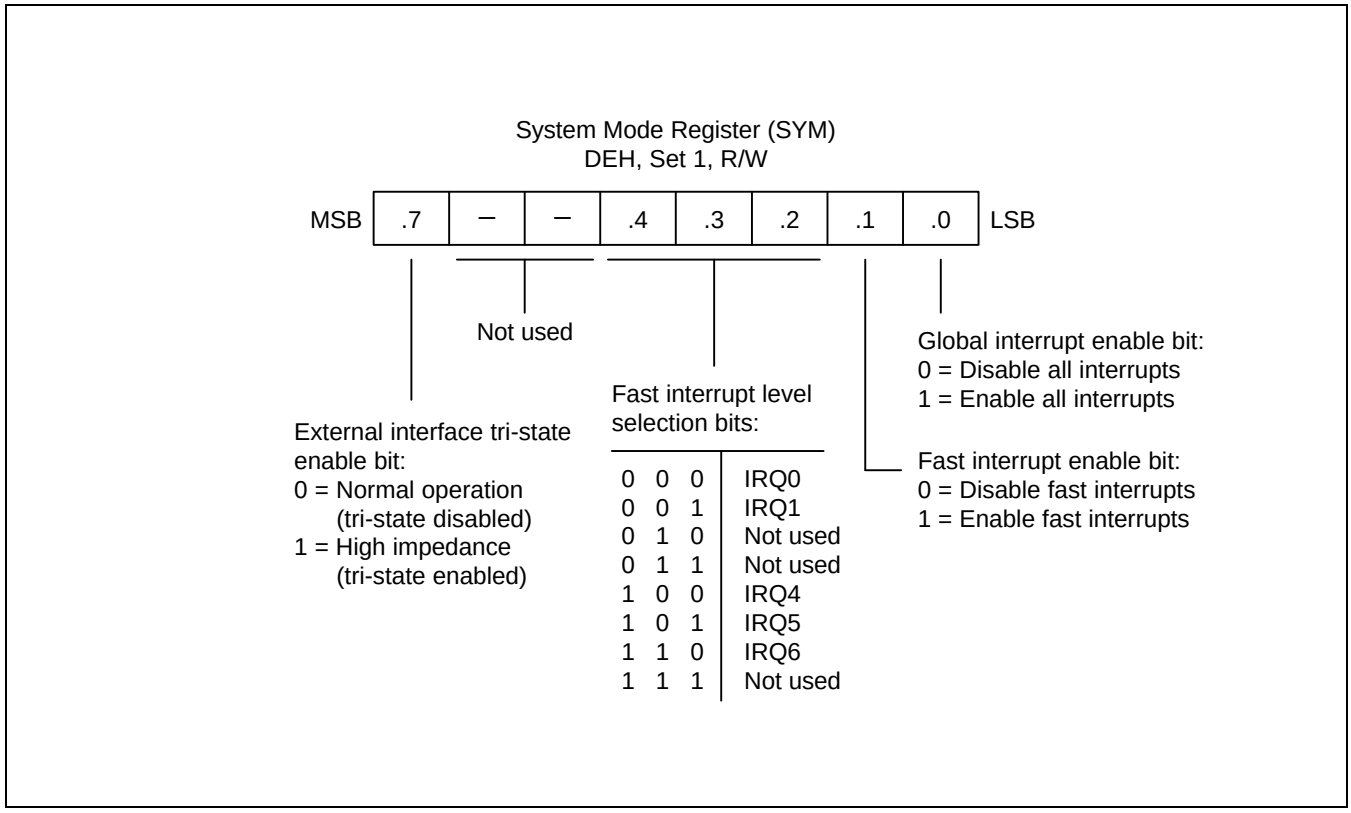


Figure 5-5. System Mode Register (SYM)

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 before enabling global interrupt processing.

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.

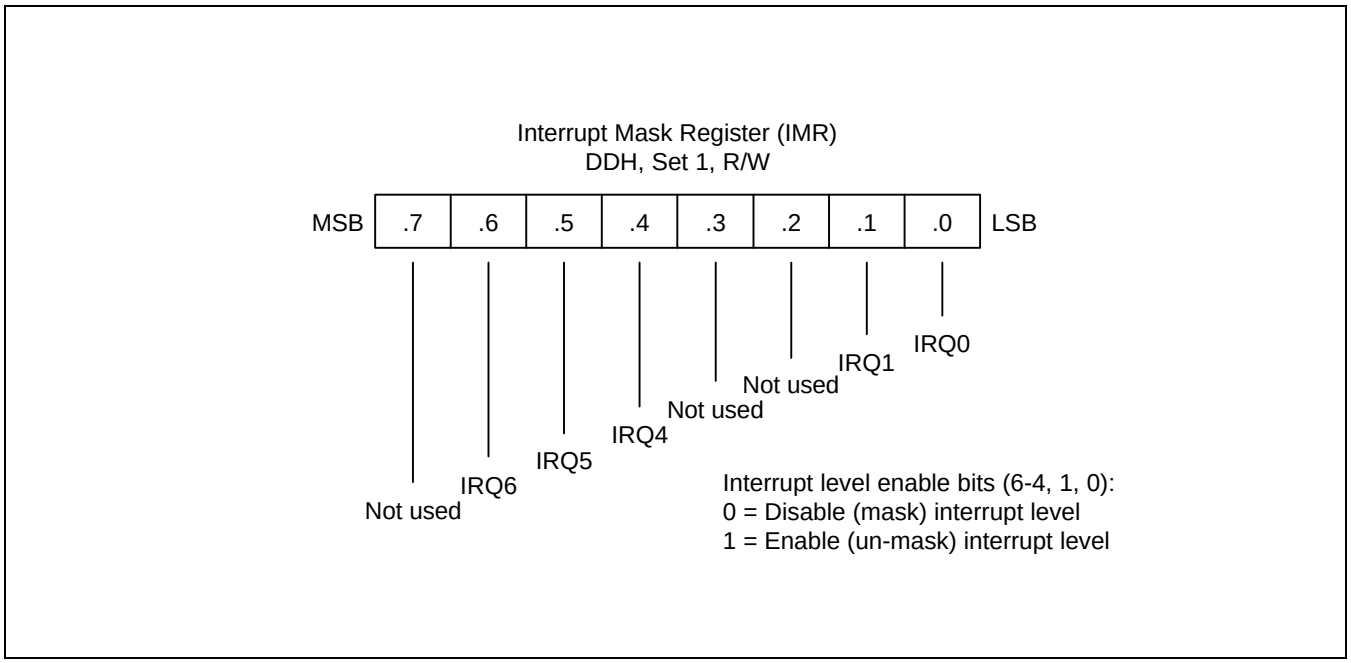


Figure 5-6. Interrupt Mask Register (IMR)

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 used in the microcontroller's interrupt structure. After a reset, all IPR bit values are undetermined and must therefore be written to their required settings before global interrupt processing.

When more than one interrupt source is active, the source with the highest priority level is serviced first. If both sources belong to the same interrupt level, the source with the lowest vector address usually has 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 IRQ4
Group C IRQ5, IRQ6

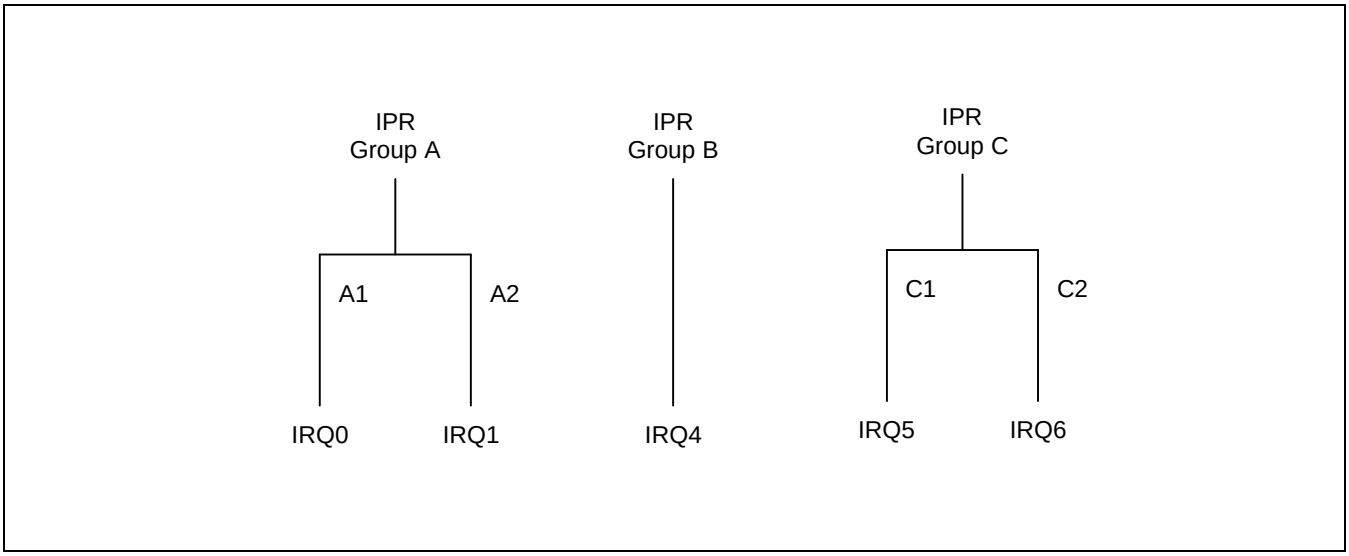


Figure 5-7. Interrupt Request Priority Groups

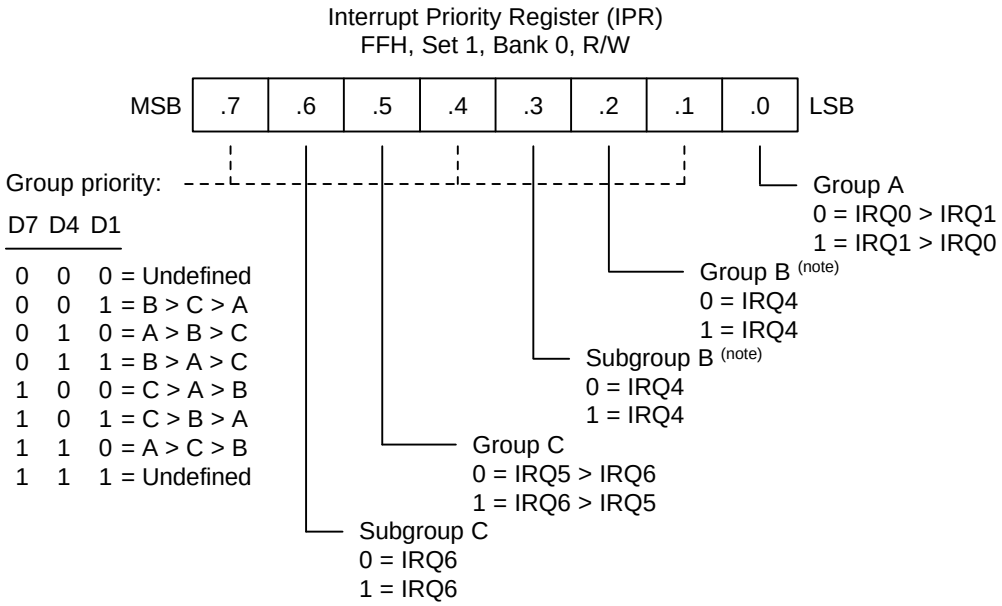
As you can see in Figure 5-8, IPR.7, IPR.4, and IPR.1 control the relative priority of interrupt groups A, B, and C. For example, the setting "001B" for these bits would select the group relationship B > C > A; the setting "101B" would select the relationship C > B > A.

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

— IPR.5 controls the relative priorities of group C interrupts.

— Interrupt group B has a subgroup to provide an additional priority relationship between for interrupt levels 2, 3, and 4. IPR.3 defines the possible subgroup B relationships. IPR.2 controls interrupt group B. In the S3C820B implementation, interrupt levels 2, 3, and 7 are not used. Therefore, IPR.2, IPR.3 and IPR.6 settings are not evaluated, as IRQ4 is the only remaining level in the group.

— IPR.0 controls the relative priority setting of IRQ0 and IRQ1 interrupts.



NOTE: In the S3C820B interrupt structure, only levels IRQ0, IRQ1 and IRQ4-IRQ6 are used. Settings for group/subgrou B and subgroup C, which control relative priorities for levels IRQ2, IRQ3 and IRQ7 are therefore not evaluated.

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.

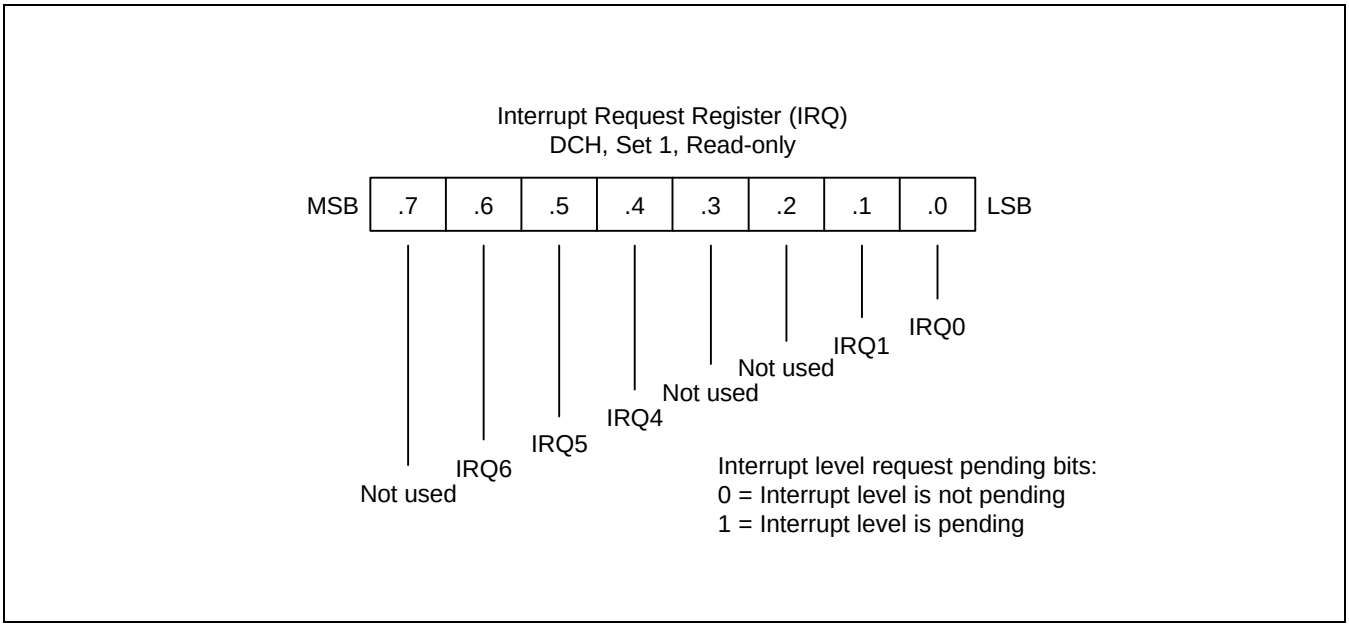


Figure 5-9. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

There are two types of interrupt pending bits: One type is automatically cleared by hardware after the interrupt service routine is acknowledged and executed; the other type must be cleared by 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 S3C820B interrupt structure, the timer 0 overflow and watch timer interrupts (IRQ0 and IRQ4) belong to this category of interrupts whose pending condition is cleared automatically by hardware.

Pending Bits Cleared by the Service Routine

The second type of pending bit must 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.

In the S3C820B interrupt structure, pending conditions for all interrupt sources *except* the timer 0 overflow and watch timer interrupts, must be cleared by the interrupt service routine.

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 can be 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 level is currently requesting service
— The interrupt must be enabled at the interrupt’s source (peripheral control register)

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

1. Reset (clear to “0”) the 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 and sets SYM.0 to “1”, allowing 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 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 above procedure to some extent.

INSTRUCTION POINTER (IP)

The instruction pointer (IP) is used by all 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 IP register names are IPH (high byte, IP15–IP8) and IPL (low byte, IP7–IP0).

FAST INTERRUPT PROCESSING

The feature called *fast interrupt processing* lets you specify that an interrupt within a given level be completed in approximately six clock cycles instead of the usual 22 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 S3C820B microcontroller, the service routine for any one of the five interrupt levels: IRQ0, IRQ1, or IRQ4–IRQ6, 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 is automatically cleared by hardware after the interrupt service routine is acknowledged and executed, and the other type 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.

7 CLOCK CIRCUITS

OVERVIEW

The S3C820B microcontroller has two oscillator circuits: a main system clock, and a subsystem clock circuit. The CPU and peripheral hardware operate on the system clock frequency supplied through these circuits. The maximum CPU clock frequency, is determined by CLKCON register settings.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

— External crystal, ceramic resonator, RC oscillation source (main clock only), or an external clock
— Oscillator stop and wake-up functions (main clock only)
— Programmable frequency divider for the CPU clock (fx divided by 1, 2, 8, or 16 or fxt)
— Clock circuit control register, CLKCON

CPU Clock Notation

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

fx main clock
fxt sub clock
fxx selected system clock

MAIN OSCILLATOR CIRCUITS

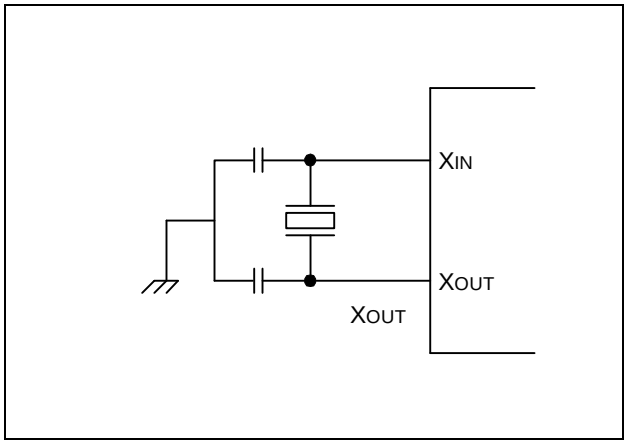


Figure 7-1. Crystal/Ceramic Oscillator

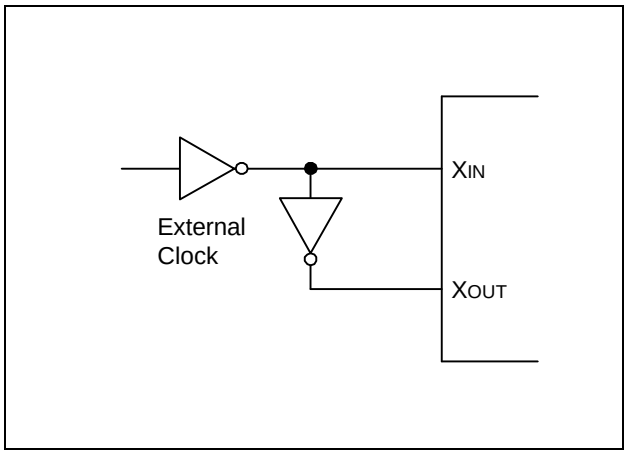


Figure 7-2. External Oscillator

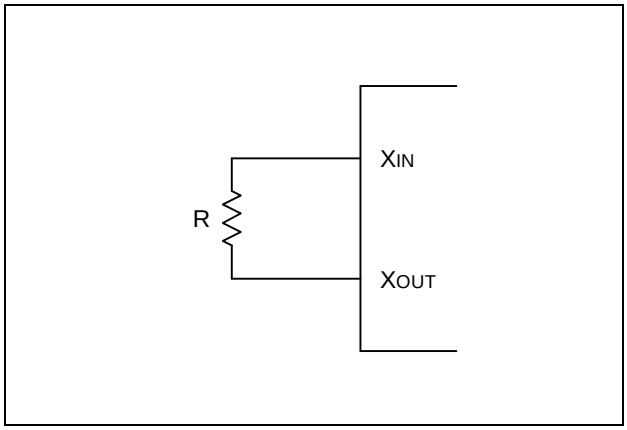


Figure 7-3. RC Oscillator

SUB OSCILLATOR CIRCUITS

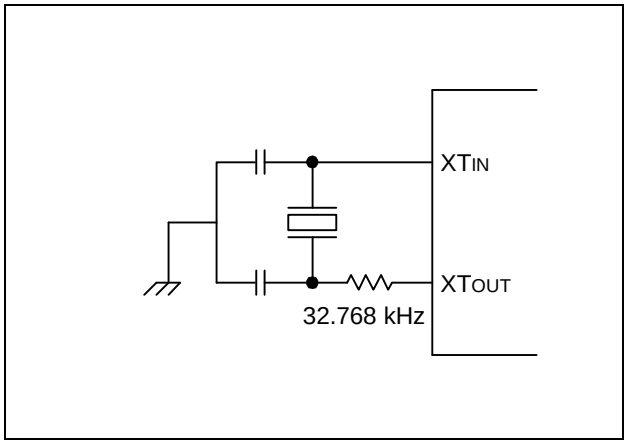


Figure 7-4. Crystal/Ceramic Oscillator

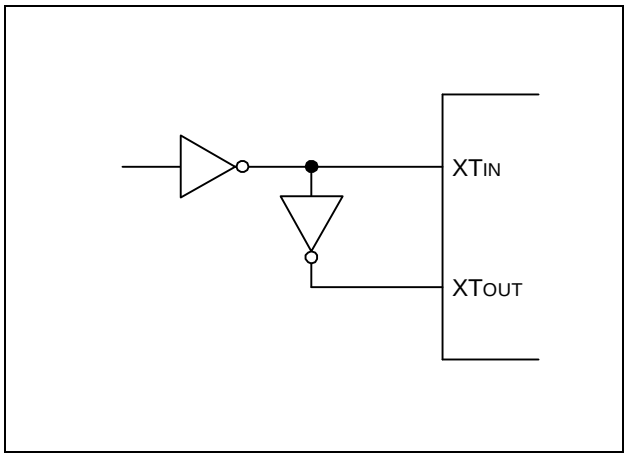


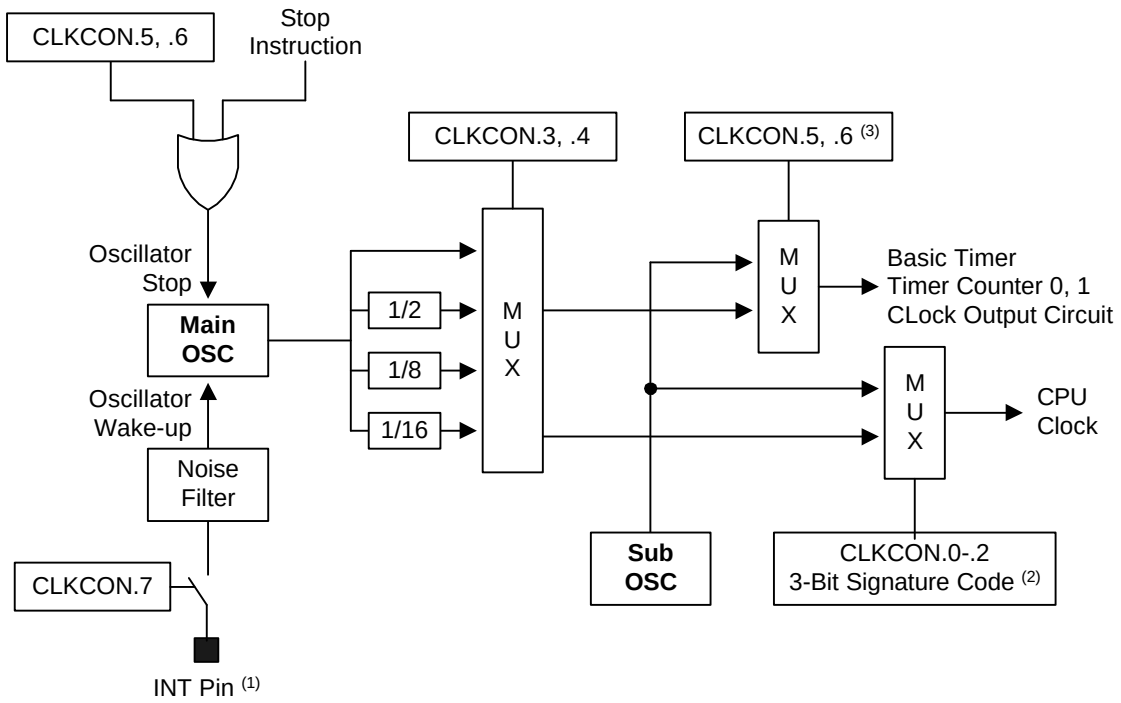
Figure 7-5. External Oscillator

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 started, by a reset operation, by an external interrupt, or by a watch timer interrupt if sub clock is selected as watch timer clock source (When the fx is selected as system clock).

— In Idle mode, the internal clock signal is gated away from the CPU, but continues to be supplied to the interrupt structure, timer 0, timer 1, and watch timer. Idle mode is released by a reset or by an interrupt (externally or internally generated).



NOTES:

1. An external interrupt can be used to release Stop mode and "wake up" the main oscillator.
2. When sub clock (fxt) is selected as system clock (fxx), you must use idle instruction instead of stop instruction for power down mode.
3. Although sub clock (fxt) is selected as sysctem clock (fxx), the selected clcok (fxx) of basic timer, timer counter 0/1, and clock output circuit is main clock if the value of CLKCON.6-.5 is not "10B (value to stop main clock)".

Figure 7-6. System Clock Circuit Diagram

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

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

— Oscillator IRQ wake-up function enable/disable

— Oscillator frequency divide-by value

— System clock signal selection

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

After a reset, the external interrupt oscillator wake-up function is enabled, the main oscillator is activated, and the fx/16 (the slowest clock speed) is selected as the CPU clock. If necessary, you can then increase the CPU clock speed to fx, fx/2, or fx/8 and you can change system clock from main clock to sub clock.

For the S3C820B microcontroller, the CLKCON.2–CLKCON.0 system clock signature code can be any value (The “101B” setting selects sub clock as system clock). The reset value for the clock signature code is “000B”.

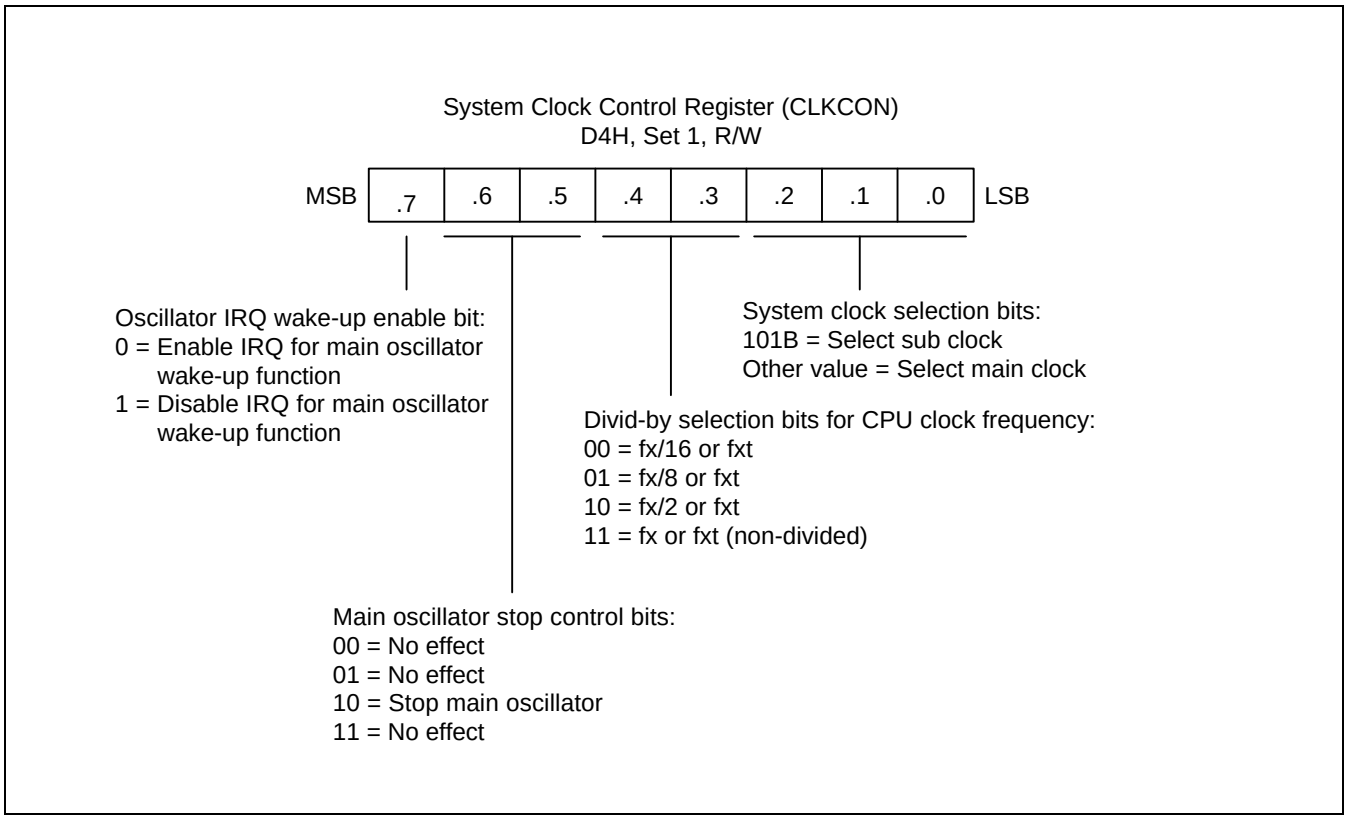


Figure 7-7. System Clock Control Register (CLKCON)

SWITCHING THE CPU CLOCK

Data loadings in the system clock control register, CLKCON, determine whether a main or a sub clock is selected as the CPU clock, and also how this frequency is to be divided. This makes it possible to switch dynamically between main and sub clocks and to modify operating frequencies.

CLKCON .2–.0 select the main clock (fx) or the sub clock (fxt) for the CPU clock and CLKCON .6–.5 start or stop main clock oscillation. CLKCON.4–.3 control the frequency divider circuit, and divide the selected fx clock by 1, 2, 8, 16, or fxt clock by 1.

For example, you are using the default CPU clock (normal operating mode and a main clock of fx/16 and you want to switch from the fx clock to a sub clock and to stop the main clock. To do this, you need to set CLKCON.2–.0 to “101B” and CLKCON.6–.5 to “10B” simultaneously. This switches the clock from fx to fxt and stops main clock oscillation.

The following steps must be taken to switch from a sub clock to the main clock: first, set CLKCON.6–.5 to any value except “10B” to enable main system clock oscillation. Then, after a certain number of machine cycles has elapsed, select the main clock by setting CLKCON.2–.0 to any value except “101B”. You must remember that the selected clock (fxx) of basic timer, timer counter0/1, and clock output circuit is main clock during the interval time. Refer to “Figure7-6”.

☞ PROGRAMMING TIP — Switching the CPU clock

1. This example shows how to change from the main clock to the sub clock:

```
MA2SUB  LD      CLKCON, #5DH        ;  Switches to the sub clock
                                    ;  Stop the main clock oscillation
        RET
```

2. This example shows how to change from sub clock to main clock:

```
SUB2MA  AND     CLKCON, #9FH        ;  Start the main clock oscillation
        CALL    DLY16               ;  Delay 16 ms
        AND     CLKCON, #98H        ;  Switch to the main clock
        RET
DLY16   SRP     #0C0H
        LD      R0, #20H
DEL     NOP
        DJNZ    R0, DEL
        RET
```

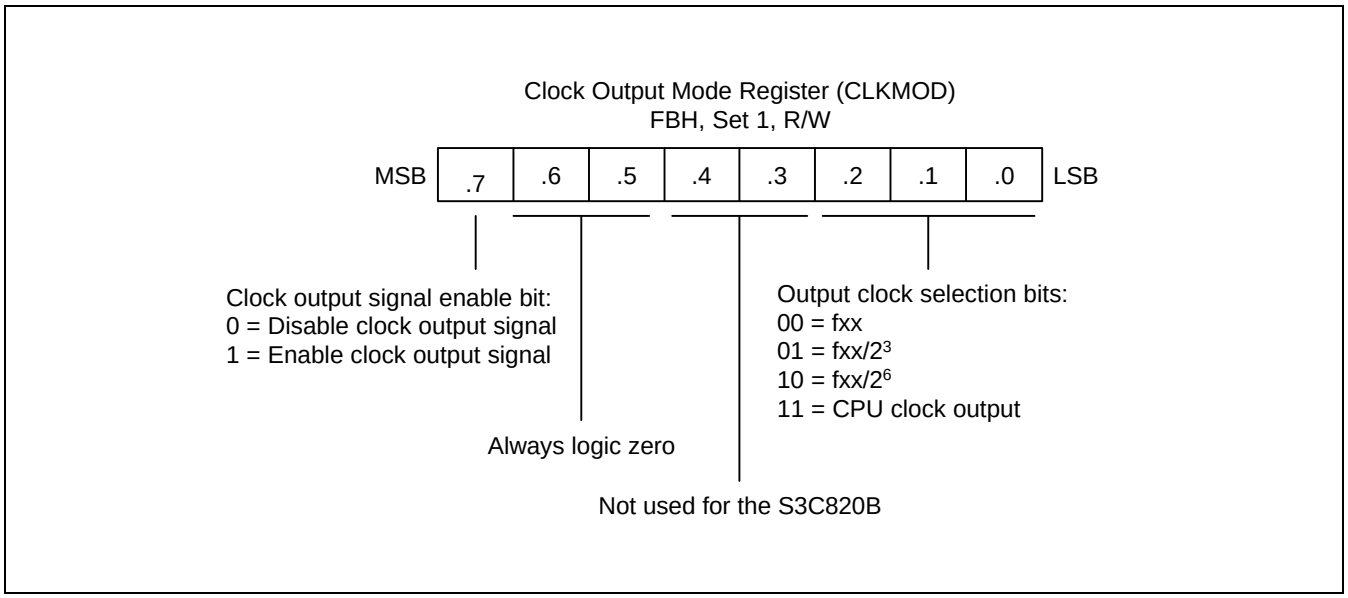


Figure 7-8. Clock Output Mode Register (CLKMOD)

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 S3C820B into a known operating status.

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

Whenever a reset occurs during normal operation (that is, when both V_{DD} and RESET are High level), the 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 (see Table 8-1).

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

— All interrupts are disabled.

— The watchdog function (basic timer) is enabled.

— Ports 0, 1, 2, and 3 are set to schmitt trigger input mode and all pull-up resistors are disabled for the I/O port pin circuits.

— Peripheral control and data register settings are disabled and reset to their default hardware values (see Table 8-1).

— 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.

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

Tables 8-1 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 a “0” is read from the bit position).

Table 8-1. Set 1 Register Values After Reset

Register Name	Mnemonic	Address		Bit Values After Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
Timer 0 counter	T0CNT	208	D0H	0	0	0	0	0	0	0	0
Timer 0 data register	T0DATA	209	D1H	1	1	1	1	1	1	1	1
Timer 0 control register	T0CON	210	D2H	0	0	0	0	0	0	0	0
Basic timer control register	BTCON	211	D3H	0	0	0	0	0	0	0	0
Clock control register	CLKCON	212	D4H	0	0	0	0	0	0	0	0
System flags register	FLAGS	213	D5H	x	x	x	x	x	x	0	0
Register pointer 0	RP0	214	D6H	1	1	0	0	0	–	–	–
Register pointer 1	RP1	215	D7H	1	1	0	0	1	–	–	–
Stack pointer (high byte)	SPH	216	D8H	x	x	x	x	x	x	x	x
Stack pointer (low byte)	SPL	217	D9H	x	x	x	x	x	x	x	x
Instruction pointer (high byte)	IPH	218	DAH	x	x	x	x	x	x	x	x
Instruction pointer (low byte)	IPL	219	DBH	x	x	x	x	x	x	x	x
Interrupt request register	IRQ	220	DCH	0	0	0	0	0	0	0	0
Interrupt mask register	IMR	221	DDH	x	x	x	x	x	x	x	x
System mode register	SYM	222	DEH	0	–	–	x	x	x	0	0
Register page pointer	PP	223	DFH	0	0	0	0	0	0	0	0
Port 0 data register	P0	224	E0H	0	0	0	0	0	0	0	0
Port 1 data register	P1	225	E1H	0	0	0	0	0	0	0	0
Port 2 data register	P2	226	E2H	0	0	0	0	0	0	0	0
Port 3 data register	P3	227	E3H	0	0	0	0	0	0	0	0
Location E4H is not mapped.											
Port 0 control register	P0CON	229	E5H	0	0	0	0	0	0	0	0
Port 1 control register	P1CON	230	E6H	0	0	0	0	0	0	0	0
Location E7H is not mapped.											
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

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

Register Name	Mnemonic	Address		Bit Values After Reset							
		Dec	Hex	7	6	5	4	3	2	1	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 interrupt control register	P3INT	236	ECH	0	0	0	0	0	0	0	0
Port 3 interrupt pending register	P3PND	237	EDH	0	0	0	0	0	0	0	0
Port 3 interrupt state register	P3STA	238	EEH	0	0	0	0	0	0	0	0
Locations EFH–F1H are not mapped.											
Timer A counter	TACNT	242	F2H	0	0	0	0	0	0	0	0
Timer B counter	TBCNT	243	F3H	0	0	0	0	0	0	0	0
Timer A data register	TADATA	244	F4H	1	1	1	1	1	1	1	1
Timer B data register	TBDATA	245	F5H	1	1	1	1	1	1	1	1
Timer A control register	TACON	246	F6H	0	0	0	0	0	0	0	0
Timer B control register	TBCON	247	F7H	–	–	0	0	0	0	0	0
Watch timer mode register	WTCON	248	F8H	0	0	0	0	0	0	0	0
LCD control register	LCON	249	F9H	0	0	0	0	0	0	0	0
Memory control register	FDCON	250	FAH	0	0	0	0	0	0	0	0
Clock output mode register	CLKMOD	251	FBH	0	0	0	0	0	0	0	0
Location FBH is not mapped.											
Basic timer counter	BTCNT	253	FDH	x	x	x	x	x	x	x	x
External memory timing register	EMT	254	FEH	0	–	–	–	–	–	0	–
Interrupt priority register	IPR	255	FFH	x	x	x	x	x	x	x	x

NOTES:

1. Although the SYM register is used for the S3C820B, SYM.5 should always be “0”. If you accidentally write a “1” to this bit during normal operation, a system malfunction may occur.
2. Except for T0CNT, IRQ, TACNT, TBCNT, and BTCNT, which are read-only, all registers in set 1 are read/write addressable.
3. You cannot use a read-only register as a destination field for the instructions OR, AND, LD, and LDB.
4. Interrupt pending flags are noted by shaded table cells.
5. Bit values of WTCON.6, FDCON.7–.3, CLKMOD.6–.4, and EMT.6–.2 should always be “0”. If you accidentally write a “1” to this bit during normal operation, a system malfunction may occur.

POWER-DOWN MODES

STOP MODE

Stop mode is invoked by the instruction STOP. In Stop mode, the operation of the CPU and main oscillator is halted. All peripherals which the main oscillator is selected as a clock source stop also because main oscillator stops. That is, the watch timer and LCD controller will not halted in stop mode if the sub clock is selected as watch timer clock source. The data stored in the internal register file are retained in stop mode. Stop mode can be released in one of three ways: by a system reset, by an internal watch timer interrupt (when sub clock is selected as clock source of watch timer), or by an external interrupt.

Example:
```
STOP
NOP
NOP
NOP
```

Using RESET to Release Stop Mode

Stop mode is released when the RESET signal goes active (Low level): all system and peripheral control registers are reset to their default hardware values and the contents of all data registers are retained. When 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 can be used to release stop mode. For the S3C820B microcontroller, we recommend using the INT0–INT7 interrupt, P3.0–P3.7.

Using an Internal Interrupt to Release Stop Mode

An internal interrupt, watch timer, can be used to release stop mode because the watch timer operates in stop mode if the clock source of watch timer is sub clock. If system clock is sub clock, you can't use any interrupts to release stop mode. That is, you had batter use the idle instruction instead of stop one when sub clock is selected as the system clock.

Please note the following conditions for Stop mode release:

— If you release stop mode using an internal or external interrupt, the current values in system and peripheral control registers are unchanged.

— 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.

— If you use an interrupt to release stop mode, the bit-pair setting for CLKCON.4/CLKCON.3 remains unchanged and the currently selected clock value is used.

The internal or 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.

NOTE

Do not use stop mode if you are using an external clock source because X_{IN} input must be cleared internally to V_{SS} to reduce current leakage, and do not configure any pins to floating node in stop mode to reduce power consumption.

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 and from all but the following peripherals, which remain active:

— Interrupt logic
— Basic timer
— Timer 0
— Timer 1 (Timer A and B)
— Watch timer
— LCD controller

I/O port pins retain the mode (input or output) they had at the time Idle mode was entered. External interface pins are halted by high or low level, in the idle mode.

Idle Mode Release

You can release Idle mode in one of two ways:

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 *slowest clock (1/16)* because of the hardware reset value for the CLKCON register. If all external interrupts are masked in the IMR register, a reset is the only way you can release Idle mode.
2. Activate any enabled interrupt — internal or external. When you use an interrupt to release Idle mode, the 2-bit CLKCON.4/CLKCON.3 value remains unchanged, and the *currently selected clock* value is used. The interrupt is then serviced. When the return-from-interrupt condition (IRET) occurs, the instruction immediately following the one which initiated Idle mode is executed.

9 I/O PORTS

OVERVIEW

The S3C820B microcontroller has two nibble-programmable and two bit-programmable I/O ports, P0–P3. All ports, P0–P3, are 8-bit ports. This gives a total of 32 I/O pins. Each port can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required.

Table 9-1 gives you a general overview of S3C820B I/O port functions.

Table 9-1. S3C820B Port Configuration Overview

Port	Configuration Options
0	8-bit general-purpose I/O port; Schmitt trigger input, schmitt trigger input with pull-up resistor, push-pull output, open-drain output, open-drain output with pull-up resistor, or external interface (A8–A15).
1	8-bit general-purpose I/O port; Schmitt trigger input, schmitt trigger input with pull-up resistor, push-pull output, open-drain output, open-drain output with pull-up resistor, or external interface (AD0–AD7).
2	8-bit general-purpose I/O port; The high byte of P2 pins, P2.7–P2.4, can be selected as schmitt trigger input, schmitt trigger input with pull-up resistor, push-pull output, or alternate function (T0, CLO, BUZ) and the low byte of P2 pins, P2.3–P2.0, can be used as schmitt trigger input, schmitt trigger input with pull-up resistor, push-pull output, or external interface (DM, DR, DW, AS)
3	8-bit general-purpose I/O port; The high byte of P3 pins, P3.7–P3.4, can be selected as schmitt trigger input, schmitt trigger input with pull-up resistor or push-pull output and the low byte of P3 pins, P3.3–P3.0, can be used as schmitt trigger input, schmitt trigger input with pull-up resistor, push-pull output, or alternate function (TA, TB); All P3 pin circuits have interrupt enable/disable (P3INT), pending control (P3PND), and rising/falling edge control (P3STA).

PORT DATA REGISTERS

Table 9-2 gives you an overview of the register locations of all four S3C820B I/O port data registers. Data registers for ports 0, 1, 2, and 3 have the general format shown in Figure 9-1.

Table 9-2. Port Data Register Summary

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

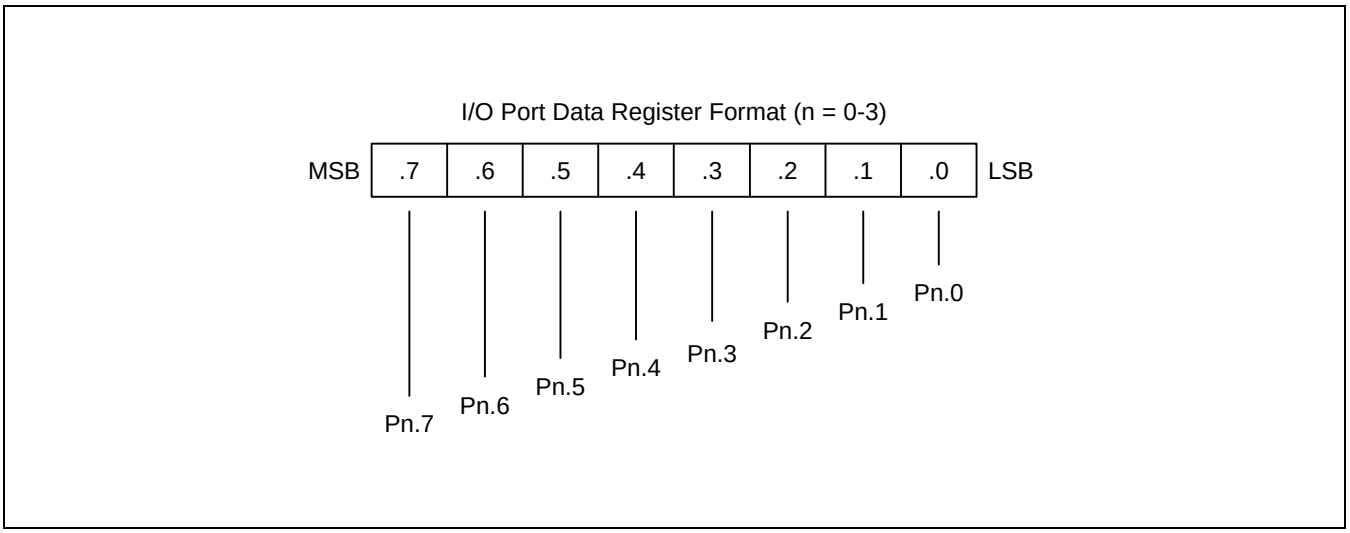


Figure 9-1. S3C820B I/O Port Data Register Format

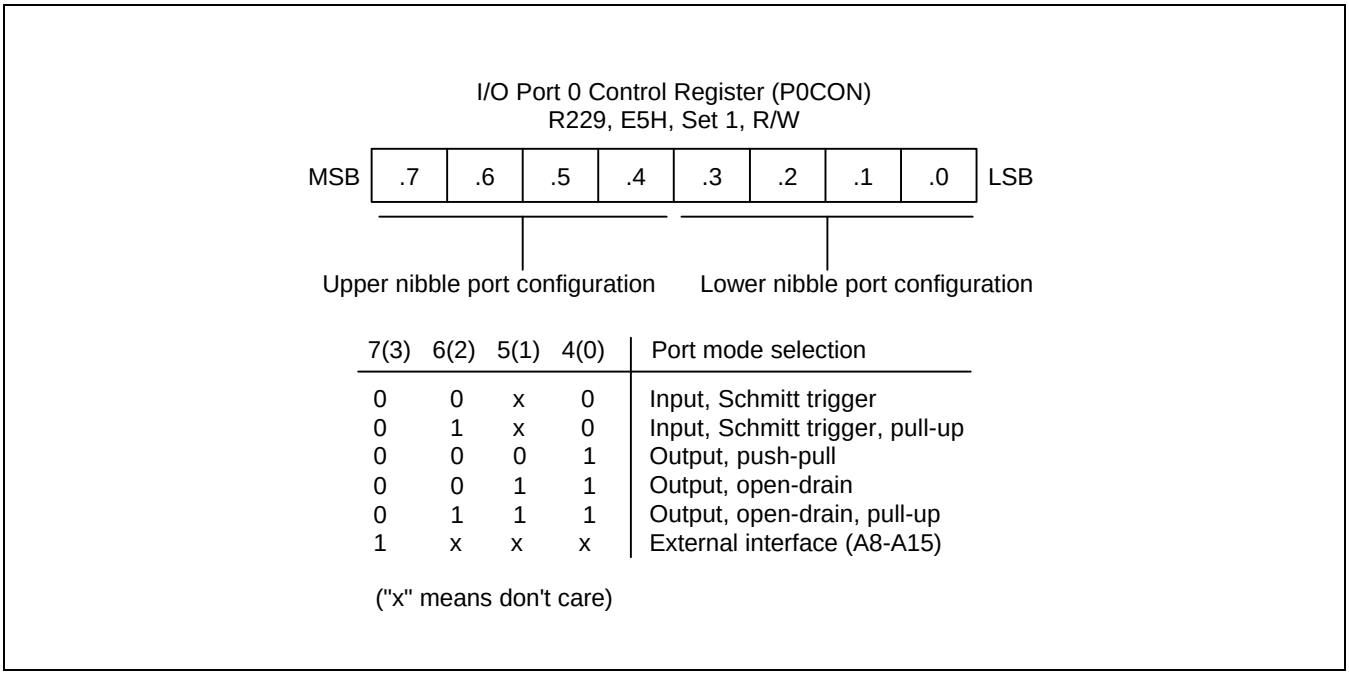
PORT 0

Port 0 pins P0.0–P0.7 can be configured on a nibble basis for general data input or output. When configured as outputs, the pins in each nibble may optionally be set to open-drain. You can alternately configure port 0 as additional address lines (A8–A15) for the external peripheral interface. It is possible to configure the lower nibble as external interface address lines A8–A11, and to use the upper nibble pins for general I/O.

To access port 0, you write or read the port 0 data register, P0 (R224, E0H) in set 1. The port 0 data register can't be written, however, when port 0 bits are configured as address lines for the external interface: writes have no effect and reads only return the state of the pin.

The port 0 control register, P0CON (R227, E5H, set 1), controls the direction of the I/O lines and allows optional selection of a pull-up resistor in input mode, and open-drain, push-pull, or pull-up selection for outputs. The P0CON setting “1xxxB” for each nibble configures the pins as external interface lines. Bits 4–7 control the upper nibble pins, P0.4–P0.7, and bits 0–3 control the lower nibble pins, P0.0–P0.3.

In normal operating mode a reset operation clears all P0CON register values to "0". If you want to configure an external memory area, you can use routine to set the P0CON value to “1xxx1xxxB”. This setting correctly configures address lines A8–A11 (lower nibble) and A12–A15 (upper nibble).



7(3)	6(2)	5(1)	4(0)	Port mode selection
0	0	x	0	Input, Schmitt trigger
0	1	x	0	Input, Schmitt trigger, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open-drain
0	1	1	1	Output, open-drain, pull-up
1	x	x	x	External interface (A8-A15)

("x" means don't care)

Figure 9-2. Port 0 Control Register (P0CON)

PORT 1

Port 1 is basically identical to port 0, except that its alternate use is as multiplexed address/data lines for the external interface. (Port 0 can alternately be configures as additional address lines A8–A15.)

Port 1 pins P1.0–P1.7 can be configured on a nibble basis for general data input or output. When configured as outputs, the pins in each nibble may optionally be set to open-drain. You can alternately configure port 0 as additional address/data lines (AD0–AD7) for the external peripheral interface.

To access port 1, you write or read the port 1 data register, P1 (R225, E1H) in set 1. The port 1 data register cannot be written, however, when port 1 bits are configured as address lines for the external interface: writes have no effect and reads only return the state of the pin.

The port 1 control register, P1CON (R230, E6H, set 1), controls the direction of the I/O lines and allows optional selection of a pull-up resistor in input mode, and open-drain, push-pull, or pull-up selection for outputs. The P1CON setting “1xxxB” for each nibble configures the pins as external interface lines. Bits 4–7 control the upper nibble pins, P1.4–P1.7, and bits 0–3 control the lower nibble pins, P1.0–P1.3.

In normal operating mode a reset operation clears all P0CON register values to "0". If you want to configure an external memory area, you can use routine to set the P0CON value to “1xxx1xxxB”. This setting correctly configures address/data lines AD0–AD7.

I/O Port 1 Control Register (P1CON)
R230, E6H, Set 1, R/W

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

Upper nibble port configuration — Lower nibble port configuration

7(3)	6(2)	5(1)	4(0)	Port mode selection
0	0	x	0	Input, Schmitt trigger
0	1	x	0	Input, Schmitt trigger, pull-up
0	0	0	1	Output, push-pull
0	0	1	1	Output, open-drain
0	1	1	1	Output, open-drain, pull-up
1	x	x	x	External interface (AD0-AD7)

("x" means don't care)

Figure 9-3. Port 1 Control Register (P1CON)

PORT 2

Port 2 is an 8-bit I/O port with individually configurable pins. It is accessed directly by writing or reading the port 2 data register, P2 (R226, E2H) in set 1. You can use port 2 for general I/O, or for the following alternative functions:

— P2.0–P2.3 can be configured as multiplexed external interface bus control lines for the DM (data memory), DR (data read), DW (data write), and AS (address strobe) signals.

— P2.4–P2.7 can be configured, respectively, as BUZ signal, CLO (Clock output), T0CK (T0 clock input), and T0 (T0 clock output or capture input.) output

The special functions that you can program using the port 2 high byte control register must also be enabled in the associated peripheral. Also, when using port 2 pins for functions other than general I/O, you must still set the corresponding port 2 control register value to configure each bit to input or output mode.

PORT 2 CONTROL REGISTERS

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

Port 2 High-Byte Control Register (P2CONH)

Four bit-pairs in the port 2 control register (P2CONH) configure port 2 pins P2.4–P2.7 to schmitt trigger input, schmitt trigger input with pull-up resistor, or push-pull output mode.

Table 9-3. Port 2 Data Register Summary (High Nibbles)

P2CONH Bit-Pair	Corresponding Port 2 Pin	Alternate Pin Function
Bits 0 and 1	P2.4	T0 output or capture input (T0)
Bits 2 and 3	P2.5	T0 clock input (T0CK)
Bits 4 and 5	P2.6	Clock output (CLO)
Bits 6 and 7	P2.7	Buzzer signal output (BUZ)

Port 2 Low-Byte Control Register (P2CONL)

The low-byte port 2 pins, P2.0–P2.3, can be configured individually as schmitt trigger inputs, schmitt trigger input with pull-up resistor, or as push-pull outputs. You can alternately configure these pins as multiplexed bus control signal lines for the external interface. To select the bus signal function, you must set the appropriate bit-pairs to “11B”.

Table 9-4. Port 2 Data Register Summary (Low Nibbles)

P2CONL Bit-Pair	Corresponding Port 2 Pin	Alternate Pin Function
Bits 0 and 1	P2.0	Address strobe (AS)
Bits 2 and 3	P2.1	Data write (DW)
Bits 4 and 5	P2.2	Data read (DR)
Bits 6 and 7	P2.3	Data memory (DM)

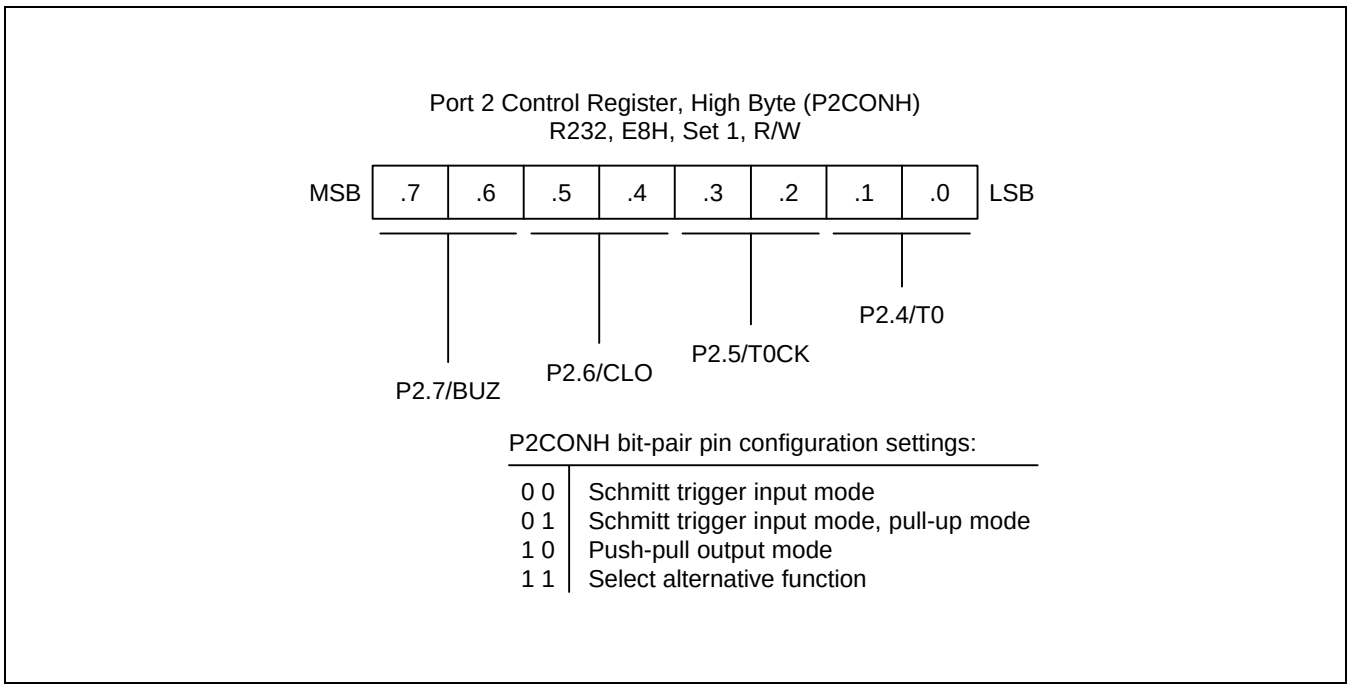


Figure 9-4. Port 2 High-Byte Control Register (P2CONH)

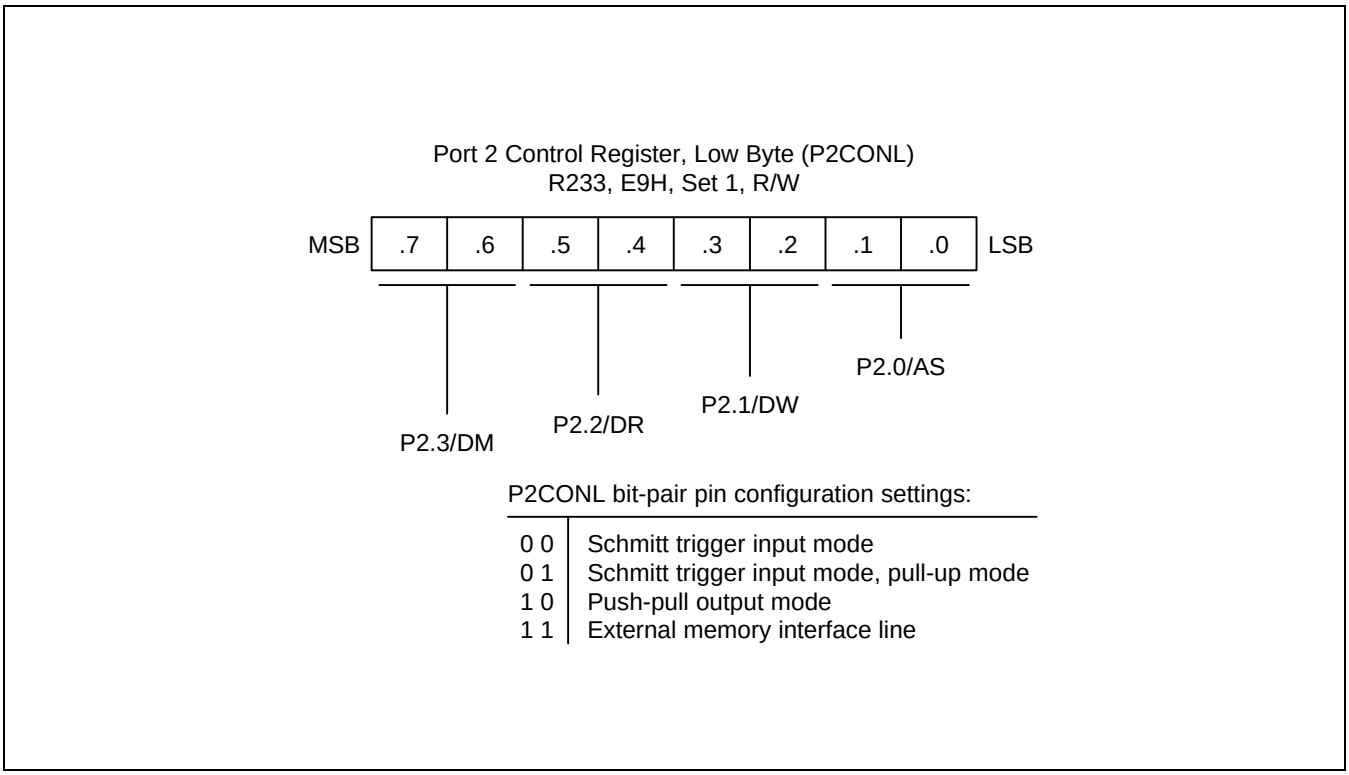


Figure 9-5. Port 2 Low-Byte Control Register (P2CONL)

PORT 3

Port 3 can serve either as a general-purpose 8-bit I/O port, alternative functions (TA for timer 1/A output and TB for timer B output), or its pins can be configured individually as external interrupt inputs. All inputs are schmitt triggered. Port 3 is accessed directly by writing or reading the Port 3 data register, P3 (R227, E3H) in set 1.

PORT 3 CONTROL REGISTERS

The direction of each port pin is configured by bit-pair settings in two control registers: P3CONH (high byte, EAH, set 1) and P3CONL (low byte, EBH, set 1). P3CONH controls pins P3.4–P3.7 (pins 48–51) and P3CONL controls pins P3.0–P3.3 (pins 44–47). Both registers are read-write addressable using 1-bit or 8-bit instructions.

When output mode is selected, a push-pull circuit is automatically configured. P3.0–P3.1 can be configured respectively, as timer B and timer 1/A output, and P3.2 can be configured as timer 1 (timer A) clock input. There are two input mode: Schmitt trigger input or schmitt trigger input with pull-up resistor.

A reset clears all P3CONH and P3CONL bits to logic zero. This configures Port 3 pins to schmitt trigger input.

Port 3 Interrupt Enable and Pending Registers (P3INT, P3PND)

To process external interrupts, two additional control registers are provided: the Port 3 interrupt enable register, P3INT (R236, ECH, set 1) and the Port 3 interrupt pending register, P3PND (R237, EDH, set 1).

By setting bits in the Port 3 interrupt enable register P3INT to "1", you can use specific Port 3 pins to generate interrupt requests when specific signal edges are detected. The interrupt names INT0–INT7 correspond to pins P3.0–P3.7. After a reset, P3INT bits are cleared to “00H”, disabling all external interrupts.

The Port 3 interrupt pending register P3PND lets you check for interrupt pending conditions and clear the pending condition when the interrupt request has been serviced. Incoming interrupt requests are detected by polling the P3PND bit values.

When the interrupt enable bit of any Port 3 pin is set to "1", a rising or falling signal edge at that pin generates an interrupt request. (Remember that the Port 3 interrupt pins must first be configured by setting them to input mode in the corresponding P3CONH or P3CONL register.)

The corresponding P3PND bit is then set to "1" and the IRQ pulse goes high to signal the CPU that an interrupt request is waiting.

When a Port 3 interrupt request has been serviced, the application program must clear the appropriate interrupt pending register bit by writing a "1" to the correct pending bit in the P3PND register. Please note that writing a "0" value has no effect.

Port 3 Interrupt State Register

P3 interrupt can be generated in falling edge or rising edge, depending on the value of the P3 interrupt state register (R238, EEH, set 1) P3STA. If the value is set to "1", P3 interrupt is generated in rising edge. If the value is set to "0", P3 interrupt is generated in falling edge.

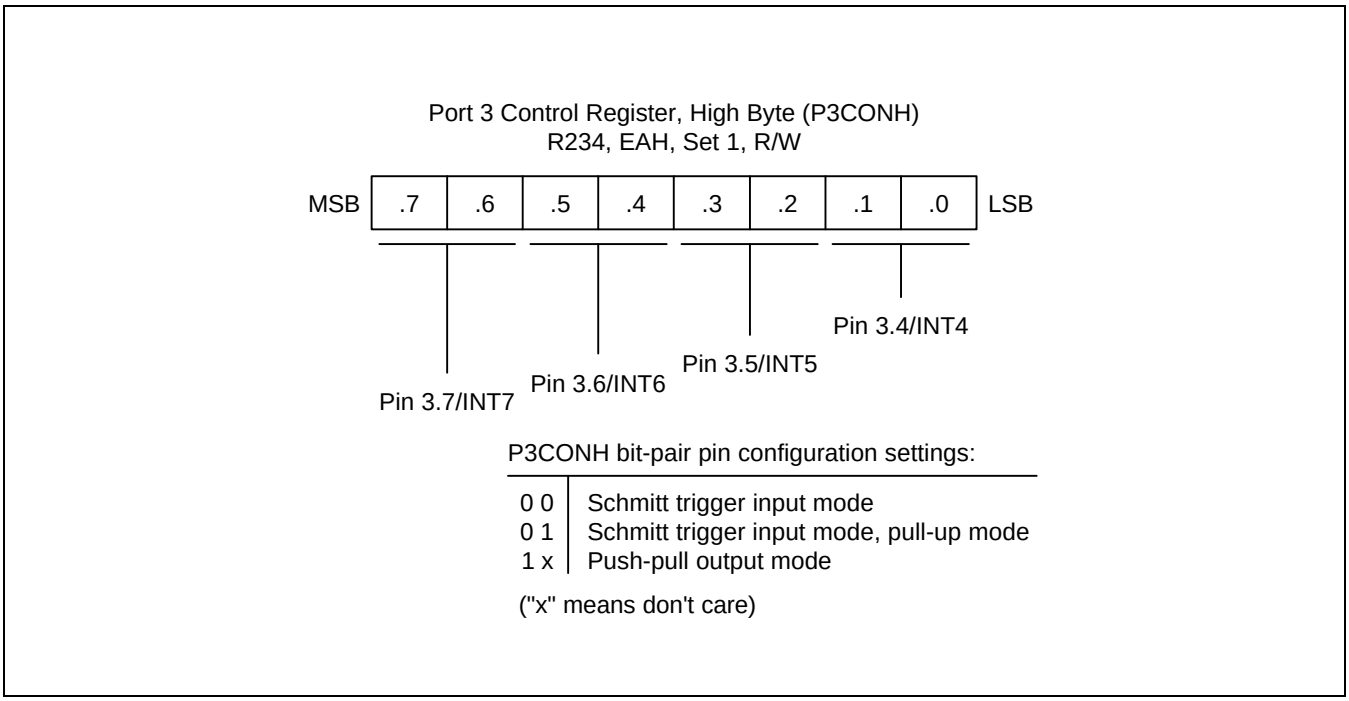


Figure 9-6. Port 3 High-Byte Control Register (P3CONH)

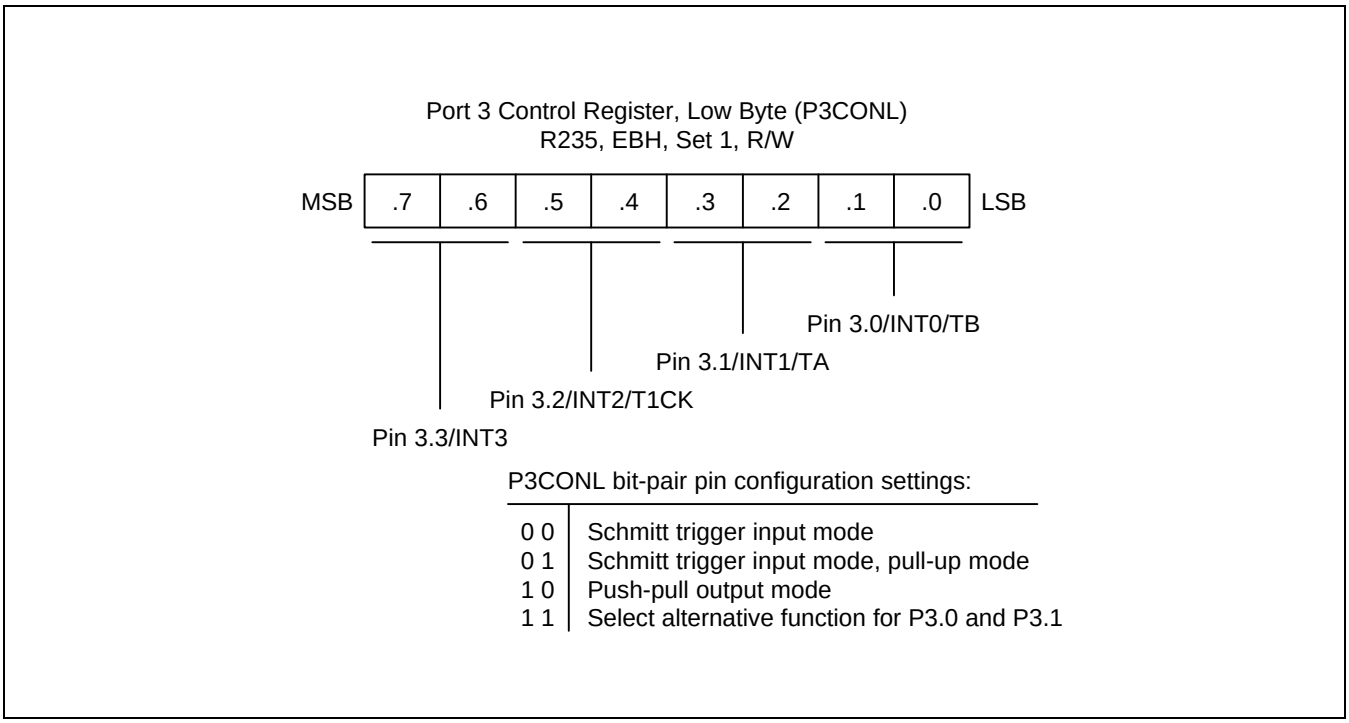


Figure 9-7. Port 3 Low-Byte Control Register (P3CONL)

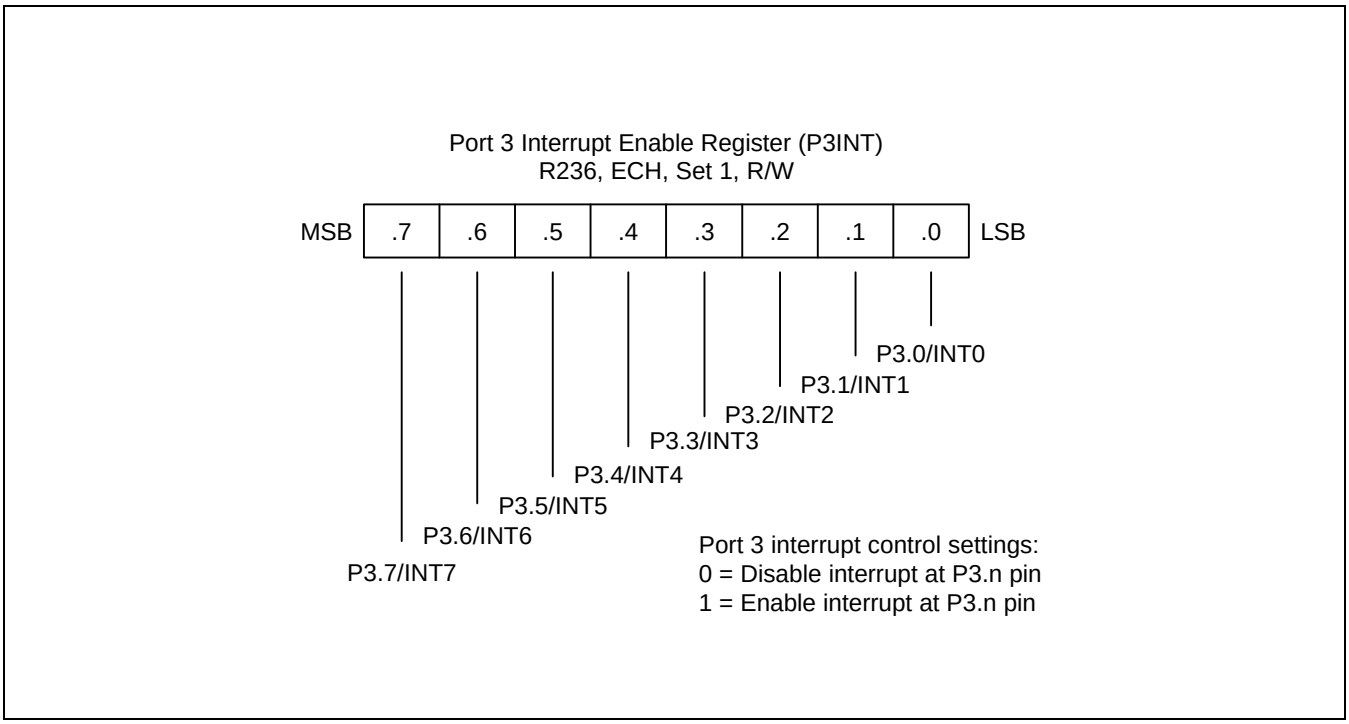


Figure 9-8. Port 3 Interrupt Enable Register (P3INT)

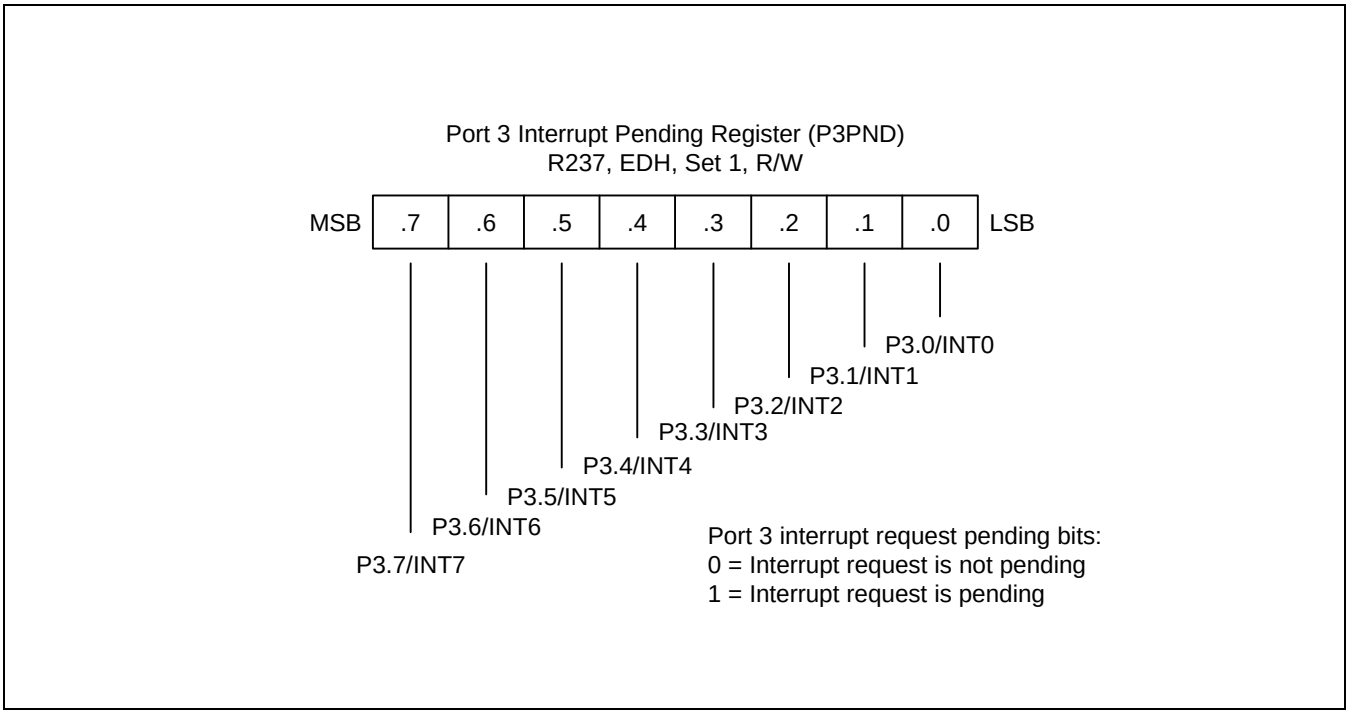


Figure 9-9. Port 3 Interrupt Pending Register (P3PND)

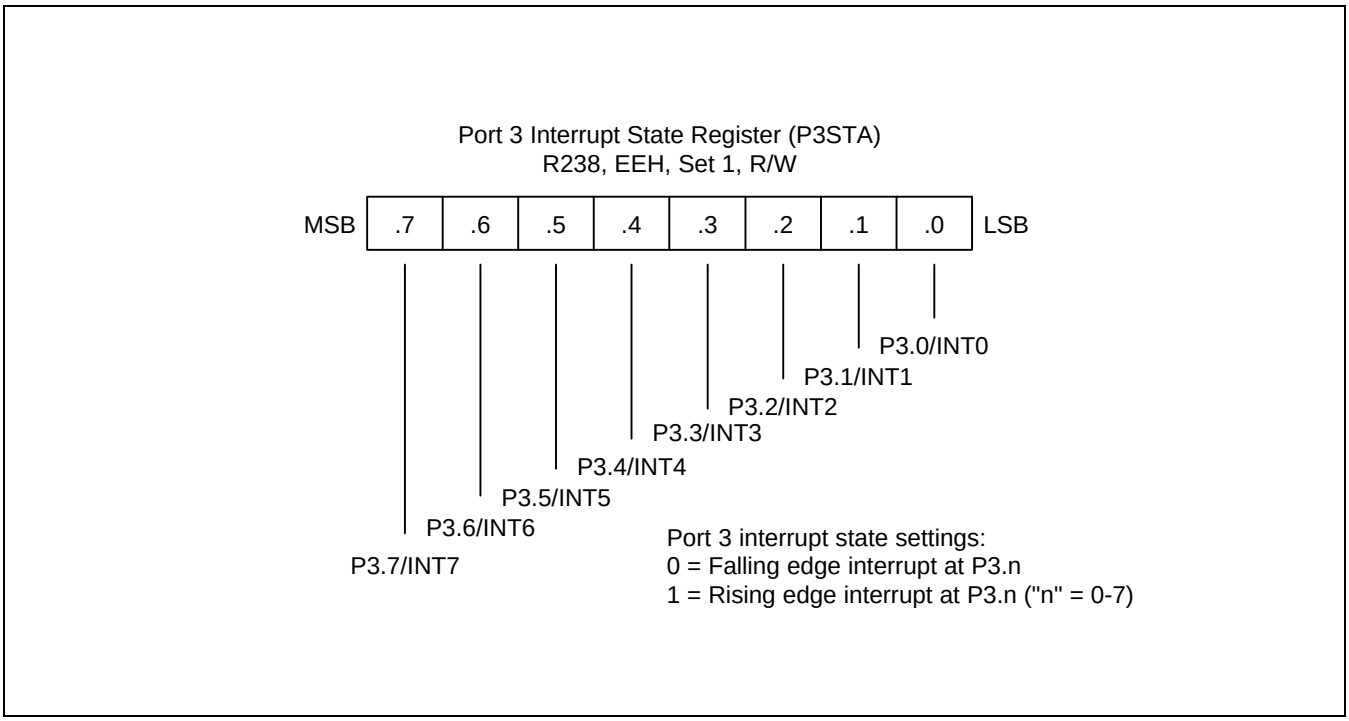


Figure 9-10. Port 3 Interrupt State Register (P3STA)

10 BASIC TIMER and TIMER 0

MODULE OVERVIEW

The S3C820B 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, or
— To signal the end of the required oscillation stabilization interval after a reset or a stop mode release.

The functional components of the basic timer block are:

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

Timer 0

Timer 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 P2.4 pin
— PWM mode

Timer 0 has the following functional components:

— Clock frequency divider (fxx divided by 4096, 256, or 8) with multiplexer
— External clock input pin (P2.5, T0CK)
— 8-bit counter (T0CNT), 8-bit comparator, and 8-bit reference data register (T0DATA)
— I/O pins for capture input (P2.4) or match output
— Timer 0 overflow interrupt (IRQ0, vector FAH) and match/capture interrupt (IRQ0, vector FCH) generation
— Timer 0 control register, T0CON (set 1, D2H, 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 fx/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 both the basic timer input clock and the timer 0 clock (unless timer 0 uses an external clock source), you write a "1" to BTCON.0.

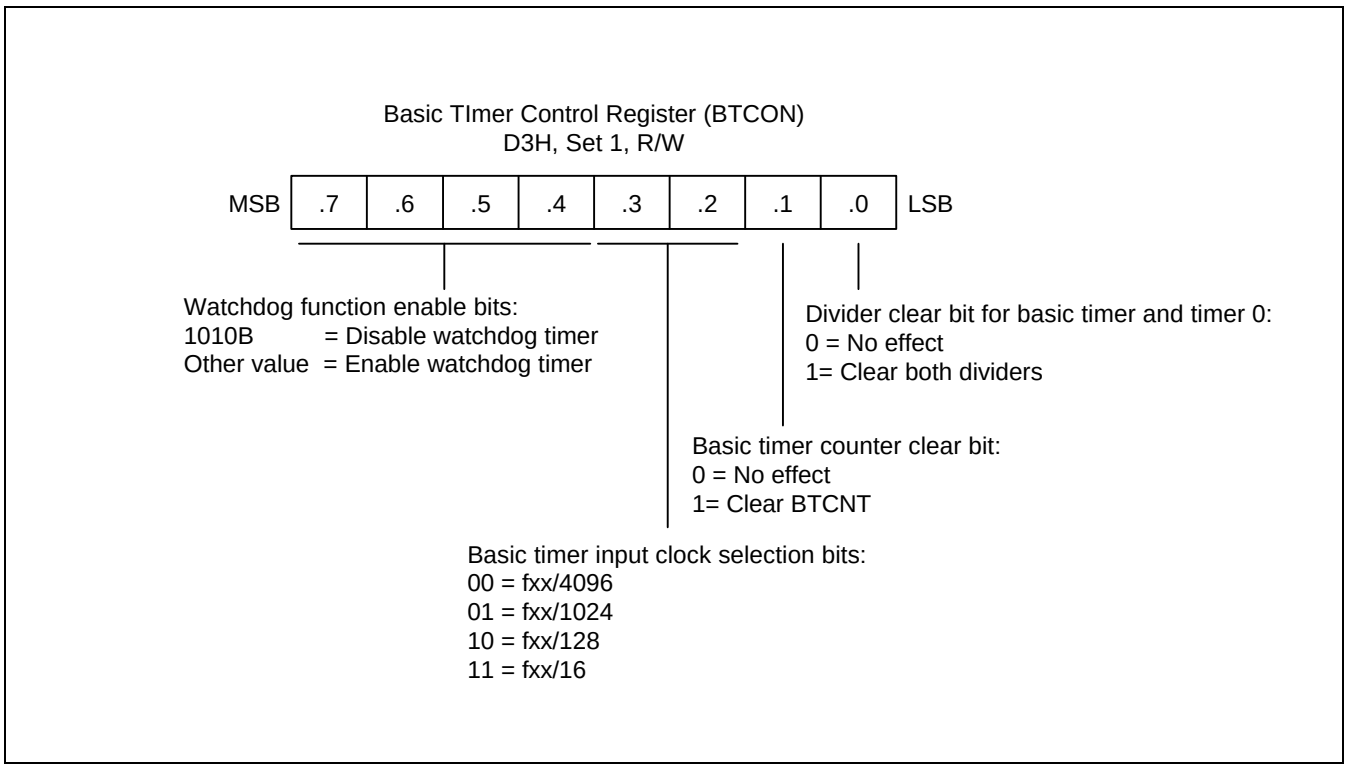


Figure 10-1. Basic Timer Control Register (BTCON)

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 fx/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.
2. If a power-on reset occurred, the basic timer counter will increase at the rate of fx/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.

TIMER 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 conditions

T0CON is located in set 1, at address D2H, 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 fx/4096, 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.3.

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

To enable the timer 0 match/capture interrupt (IRQ0, vector FCH), you must write T0CON.1 to "1". To detect a match/capture interrupt pending condition, the application program polls T0CON.0. 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 interrupt pending bit, T0CON.0.

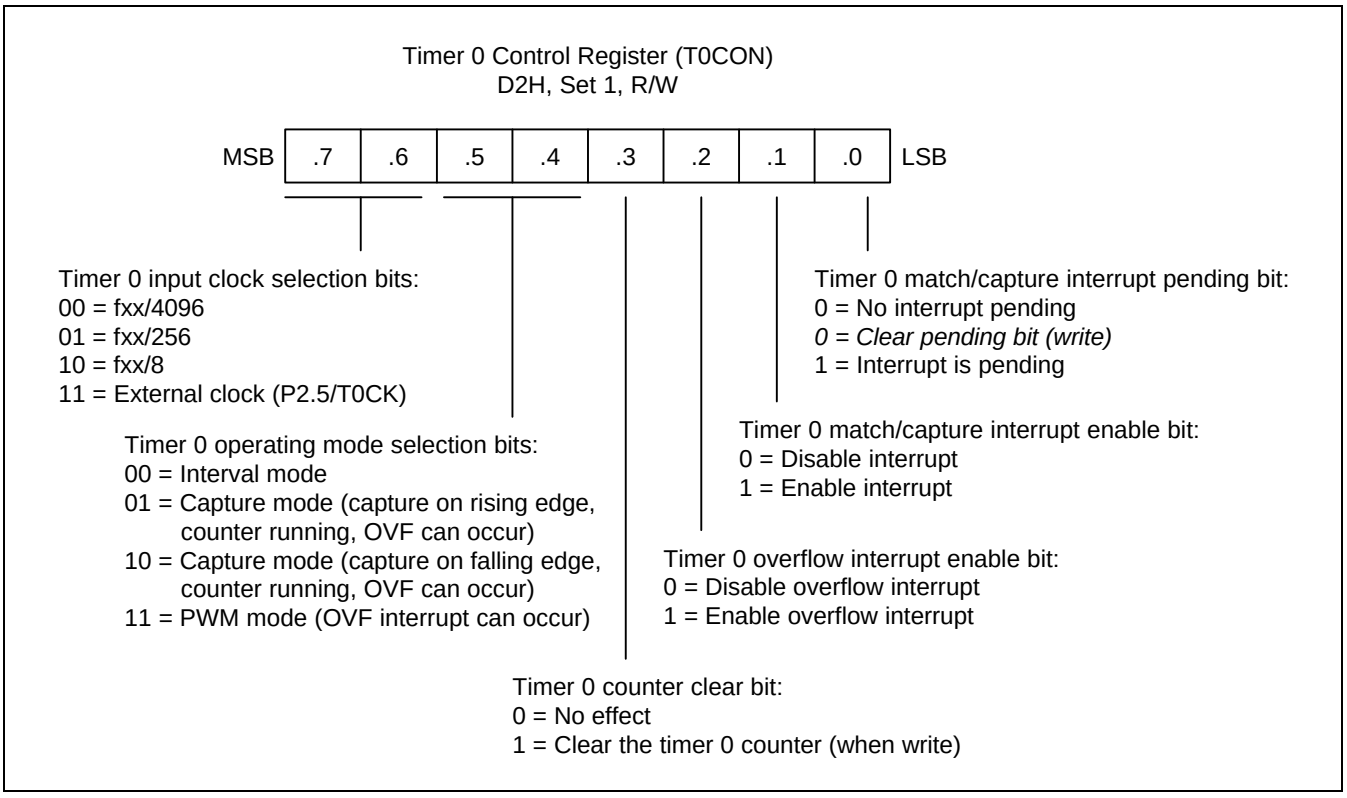


Figure 10-2. Timer 0 Control Register (T0CON)

TIMER 0 FUNCTION DESCRIPTION

Timer 0 Interrupts (IRQ0, Vectors FAH and FCH)

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

A timer 0 overflow interrupt pending condition is automatically cleared by hardware when it has been serviced. The T0INT pending condition must, however, be cleared by the application's interrupt service routine by writing a "0" to the T0CON.0 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 T0 reference data register, T0DATA. The match signal generates a timer 0 match interrupt (T0INT, vector FCH) and clears the counter.

If, for example, you write the value "10H" to T0DATA and "0BH" to T0CON, the counter will increment until it reaches "10H". At this point, the T0 interrupt request is generated, the counter value is reset, and counting resumes. With each match, the level of the signal at the timer 0 output pin is inverted (see Figure 10-3).

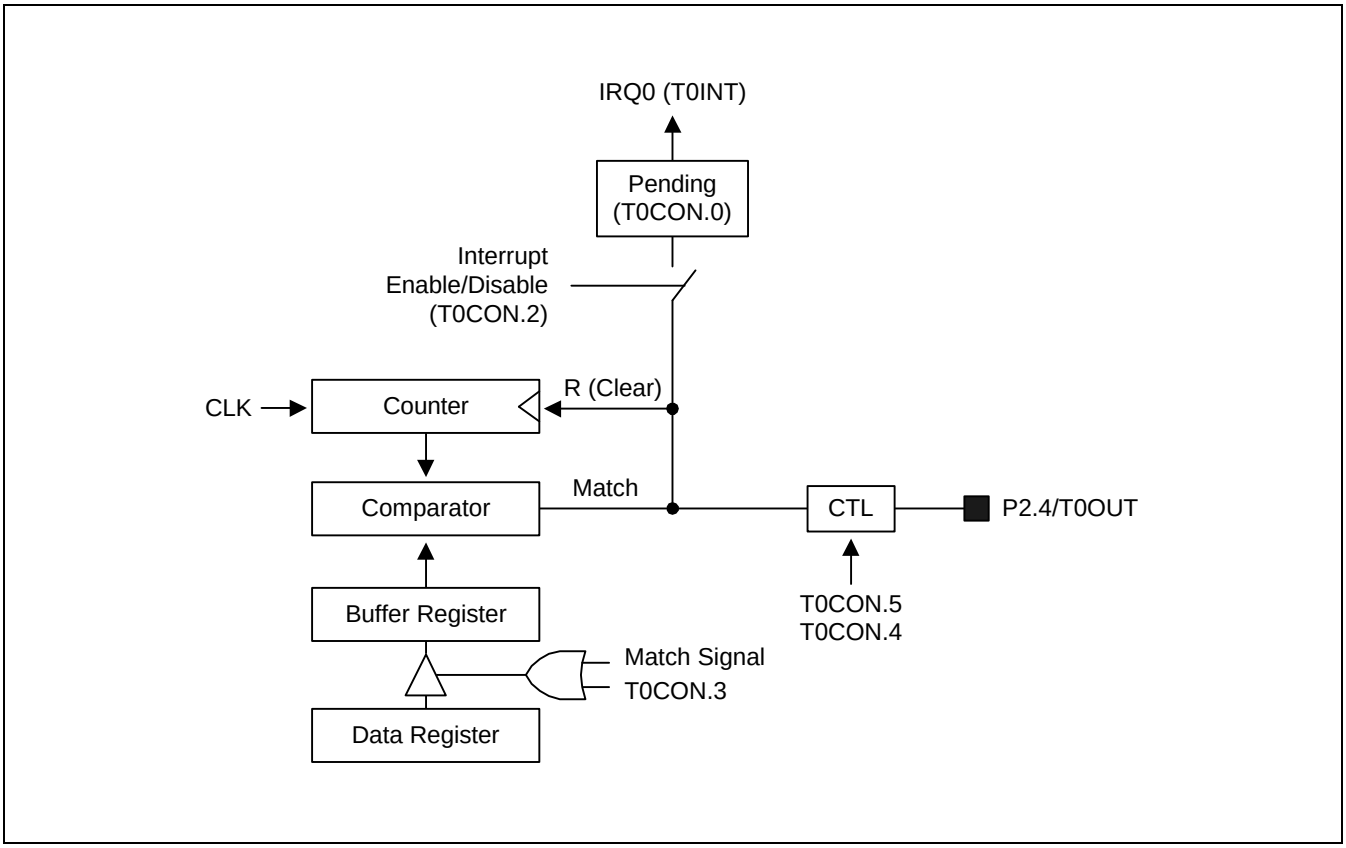
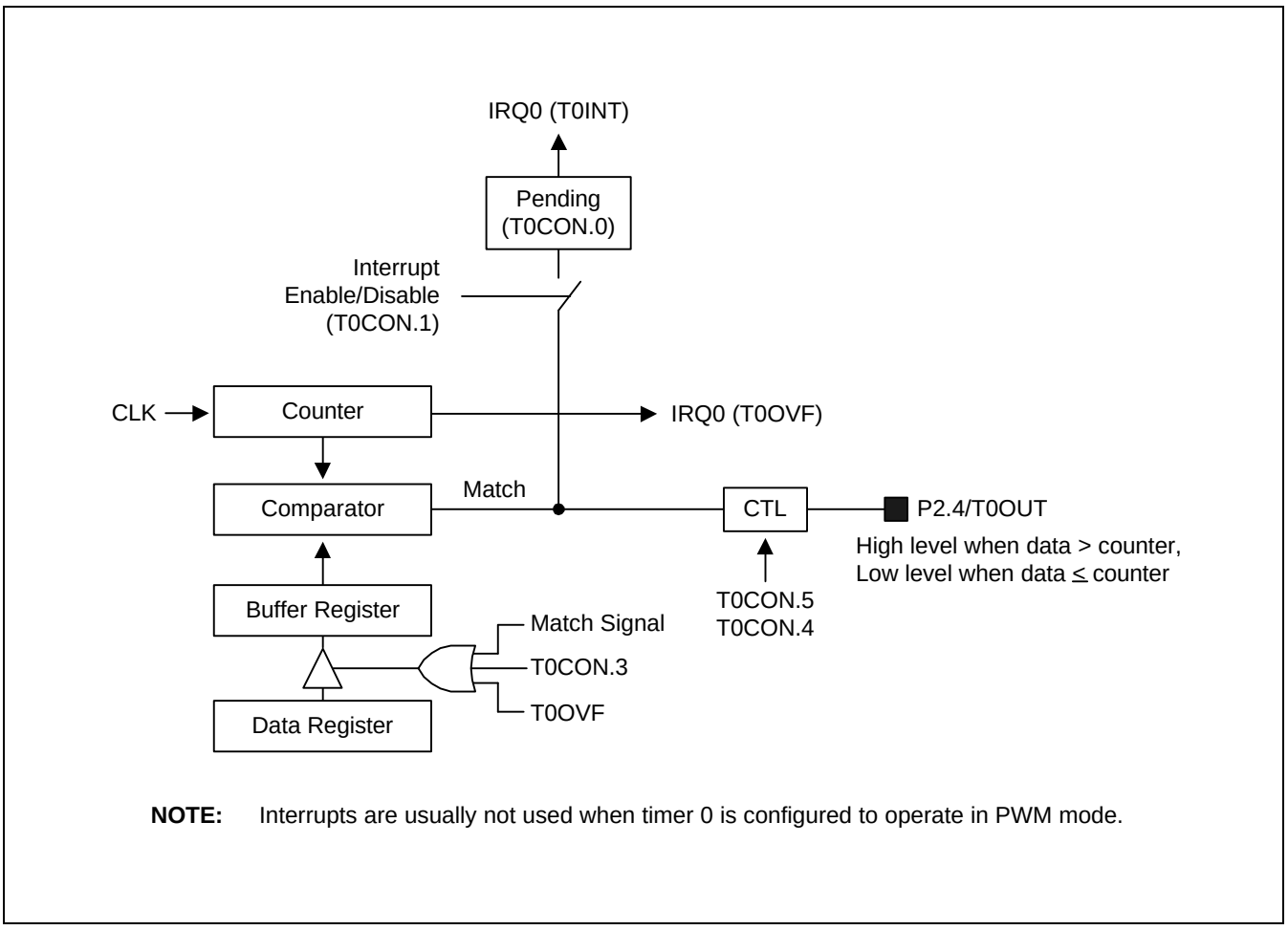


Figure 10-3. Simplified Timer 0 Function Diagram: Interval Timer Mode

Pulse Width Modulation Mode

Pulse width modulation (PWM) mode lets you program the width (duration) of the pulse that is output at the T0OUT 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 T0OUT pin is held to Low level as long as the reference data value is *less than or equal to* (≤) the counter value and then the pulse is held to High level for as long as the data value is *greater than* (>) the counter value. One pulse width is equal to $t_{CLK} \times 256$ (see Figure 10-4).



NOTE: Interrupts are usually not used when timer 0 is configured to operate in PWM mode.

Figure 10-4. Simplified Timer 0 Function Diagram: PWM Mode

Capture Mode

In capture mode, a signal edge that is detected at the T0CAP pin opens a gate and loads the current counter value into the T0 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 pin. You select the capture input by setting the values of the timer 0 capture input selection bits in the port 2 control register, P2CONH.1–.0, (set 1, E8H). When P2CONH.1–.0 is “00” or "01", 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 T0 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 10-5).

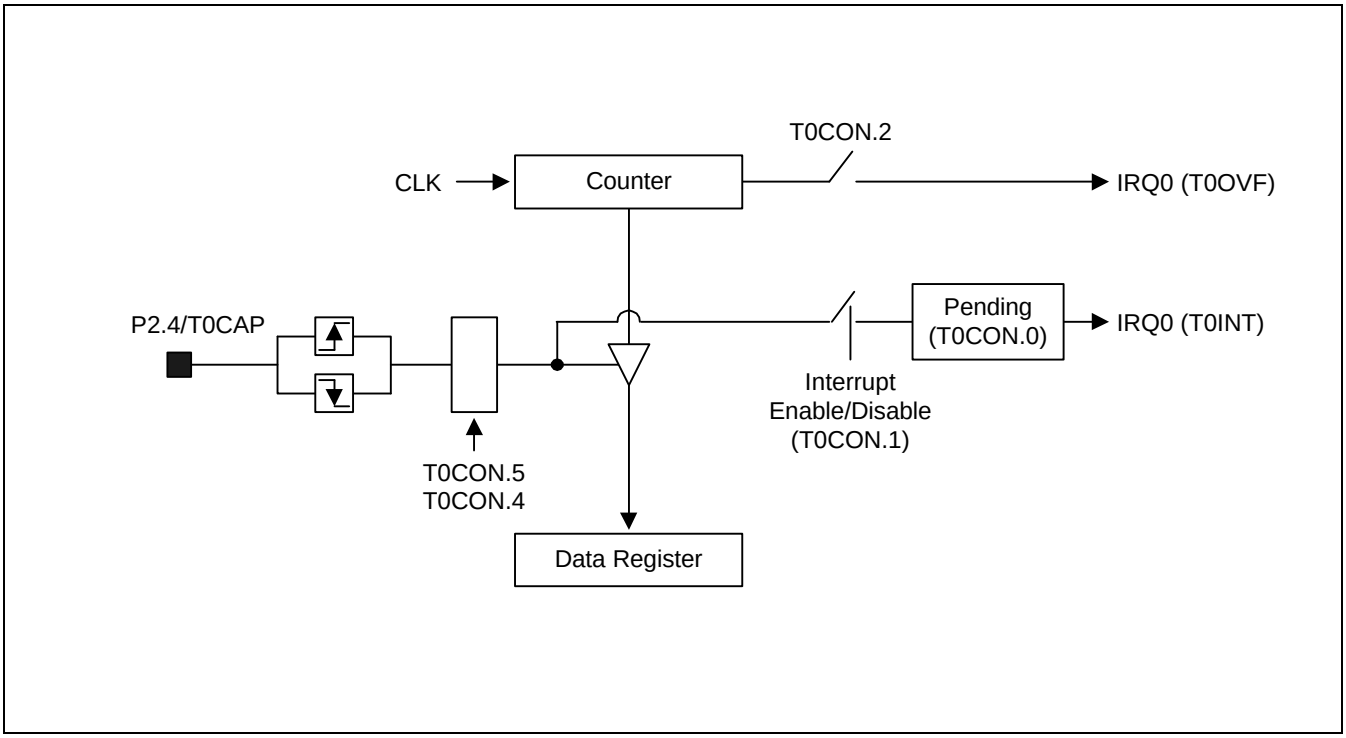
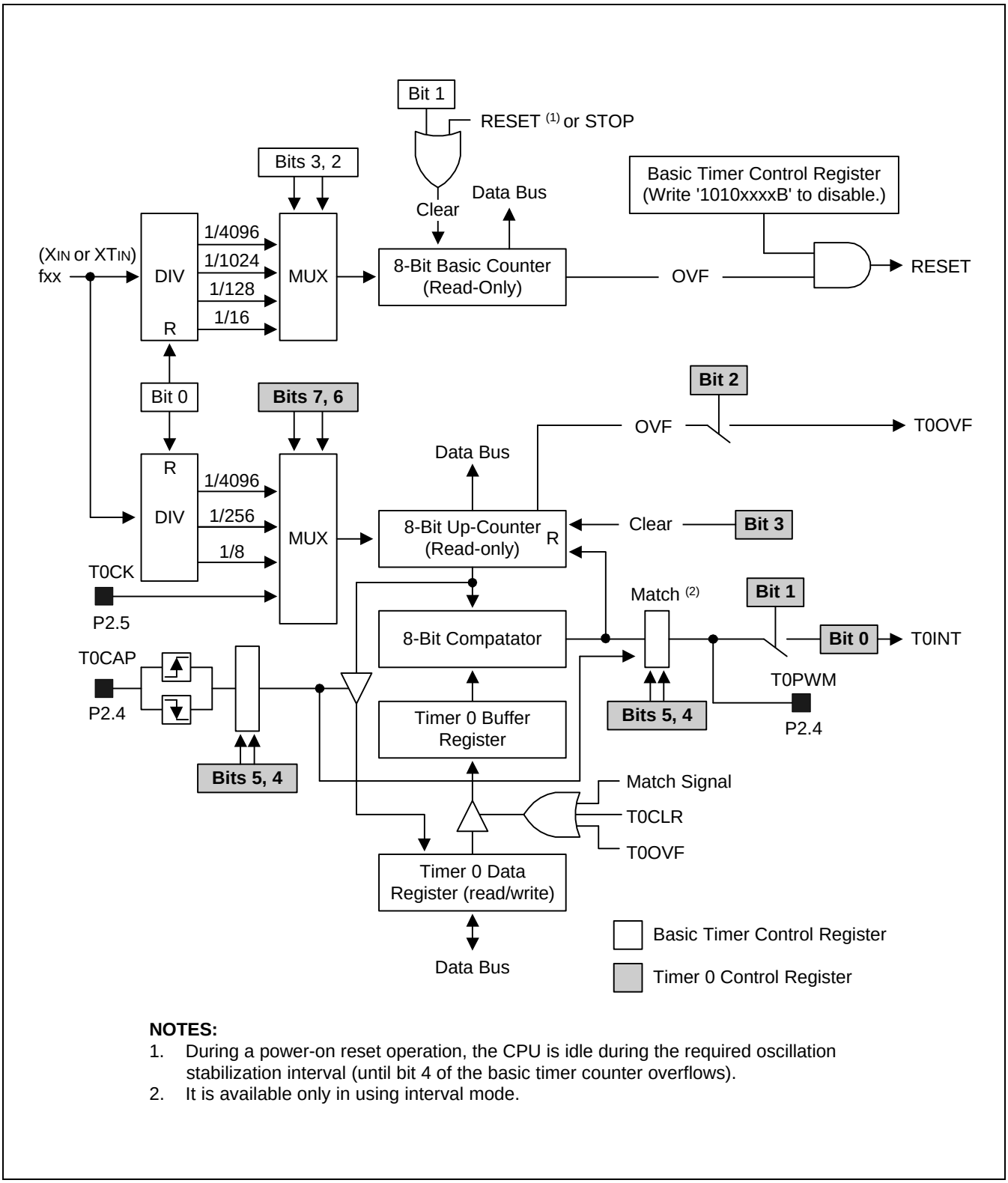


Figure 10-5. Simplified Timer 0 Function Diagram: Capture Mode



NOTES:

1. During a power-on reset operation, the CPU is idle during the required oscillation stabilization interval (until bit 4 of the basic timer counter overflows).
2. It is available only in using interval mode.

Figure 10-6. Basic Timer and Timer 0 Block Diagram

☞ PROGRAMMING TIP — Configuring the Basic Timer

This example shows how to configure the basic timer to sample specifications:

```
                ORG     0100H

RESET           DI                      ;  Disable all interrupts
                LD      BTCON, #0AAH    ;  Disable the watchdog timer
                LD      CLKCON, #18H    ;  Non-divided clock
                CLR     SYM             ;  Disable global and fast interrupts
                CLR     SPL             ;  Stack pointer low byte ← "0"
                                        ;  Stack area starts at 0FFH
                •
                •
                •
                SRP     #0C0H           ;  Set register pointer ← 0C0H
                EI                      ;  Enable interrupts
                •
                •
                •
MAIN            LD      BTCON, #52H     ;  Enable the watchdog timer
                                        ;  Basic timer clock: fxx/4096
                                        ;  Clear basic timer counter
                NOP
                NOP
                •
                •
                •
                JP      T, MAIN
                •
                •
                •
```

☞ PROGRAMMING TIP — Programming Timer 0

This sample program sets timer 0 to interval timer mode, sets the frequency of the oscillator clock, and determines the execution sequence which follows a timer 0 interrupt. The program parameters are as follows:

— Timer 0 is used in interval mode; the timer interval is set to 4 milliseconds

— Oscillation frequency is 4 MHz

— General register 64H (page 0) ← 60H + 62H + 63H + 64H (page 0) + 1H (value) is executed after a timer 0 interrupt

```
          ORG     0FAH            ;  Timer 0 overflow interrupt
          VECTOR  T0OVER
          ORG     0FCH            ;  Timer 0 match/capture interrupt
          VECTOR  T0INT
          ORG     0100H

RESET     DI                      ;  Disable all interrupts
          LD      BTCON, #0AAH    ;  Disable the watchdog timer
          LD      CLKCON, #18H    ;  Select non-divided clock
          CLR     SYM             ;  Disable global and fast interrupts
          CLR     SPL             ;  Stack pointer low byte ← "0"
                                  ;  Stack area starts at 0FFH
          •
          •
          •
          LD      T0CON, #4AH     ;  Write “01001010B”
                                  ;  Input clock is fxx/256
                                  ;  Interval timer mode
                                  ;  Enable the timer 0 interrupt
                                  ;  Disable the timer 0 overflow interrupt
          LD      T0DATA, #3FH    ;  Set timer interval to 4 milliseconds
                                  ;  (4 MHz/256) ÷ (62.5 + 1)  =  0.25 kHz (4 ms)

          SRP     #0C0H           ;  Set register pointer ← 0C0H
          EI                      ;  Enable interrupts
          •
          •
          •
T0INT     PUSH    RP0             ;  Save RP0 to stack
          SRP0    #60H            ;  RP0 ← 60H
          INC     R0              ;  R0 ← R0 + 1
          ADD     R2, R0          ;  R2 ← R2 + R0
          ADC     R3, R2          ;  R3 ← R3 + R2 + Carry
          ADC     R4, R3          ;  R4 ← R4 + R3 + Carry
```

(Continued on next page)

☞ PROGRAMMING TIP — Programming Timer 0 (Continued)

```
CP          R0, #32H                    ;  50 × 4 = 200 ms
            JR          ULT,NO_200MS_SET
            BITS        R1.2            ;  Bit setting (61.2H)

NO_200MS_SET:
            LD          T0CON,#4AH      ;  Clear pending bit
            POP         RP0             ;  Restore register pointer 0 value

T0OVER      IRET                        ;  Return from interrupt service routine
```

11 TIMER 1

ONE 16-BIT TIMER MODE (TIMER 1)

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

— One 16-bit timer mode (Timer 1)

— Two 8-bit timers mode (Timer A and B)

OVERVIEW

The 16-bit timer 1 is an 16-bit general-purpose timer. Timer 1 has the interval timer mode by using the appropriate TACON setting.

Timer 1 has the following functional components:

— Clock frequency divider (fxx divided by 1024, 512, 8, or 1 and T1CK: External clock) with multiplexer

— 16-bit counter (TACNT, TBCNT), 16-bit comparator, and 16-bit reference data register (TADATA, TBDATA)

— Timer 1 match interrupt (IRQ1, vector F4H) generation

— Timer 1 control register, TACON (set 1, F6H, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

The timer 1 module can generate an interrupt: the timer 1 match interrupt (T1INT). T1INT belongs to interrupt level IRQ1, and is assigned the separate vector address, F4H.

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 sub-routine by writing a "0" to the TACON.0 pending bit.

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the T1 reference data registers, TADATA and TBDATA. The match signal generates a timer 1 match interrupt (T1INT, vector F4H) and clears the counter.

If, for example, you write the value 10H and 32H to TADATA and TBDATA, respectively, and 8EH to TACON, the counter will increment until it reaches 3210H. At this point, the T1 interrupt request is generated, the counter value is reset, and counting resumes.

Timer 1 Control Register (TACON)

You use the timer 1 control register, TACON, to

— Enable the timer 1 operating (interval timer)

— Select the timer 1 input clock frequency

— Clear the timer 1 counter, TACNT and TBCNT

— Enable the timer 1 interrupt

— Clear timer 1 interrupt pending conditions

TACON is located in set 1, at address F6H, and is read/write addressable using register addressing mode.

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

To enable the timer 1 interrupt (IRQ1, vector F4H), you must write TACON.7, TACON.2, and TACON.1 to "1". To generate the exact time interval, you should write TACON.3 and TACON.0, which cleared counter and interrupt pending bit. To detect an interrupt pending condition when T1INT is disabled, the application program polls pending bit, TACON.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, TACON.0.

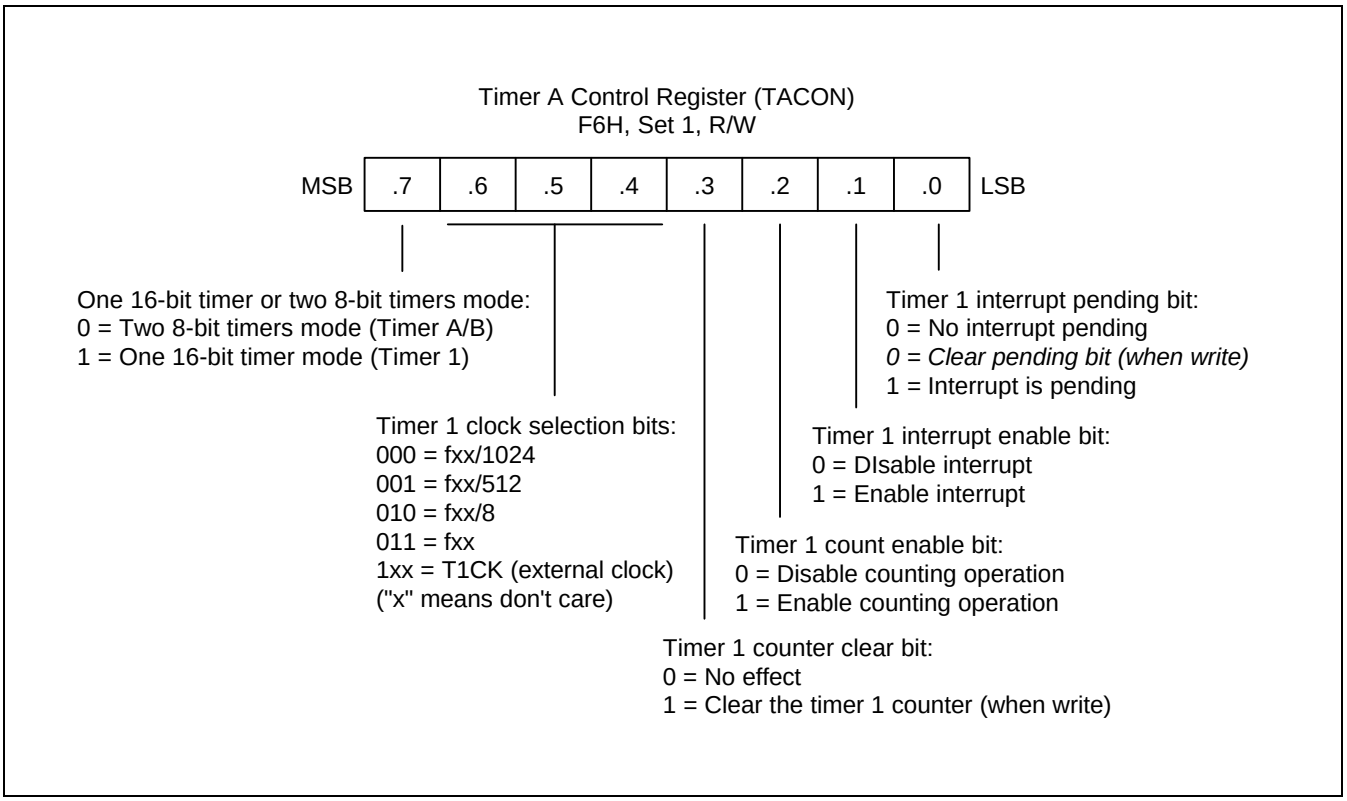


Figure 11-1. Timer 1 Control Register (TACON)

BLOCK DIAGRAM

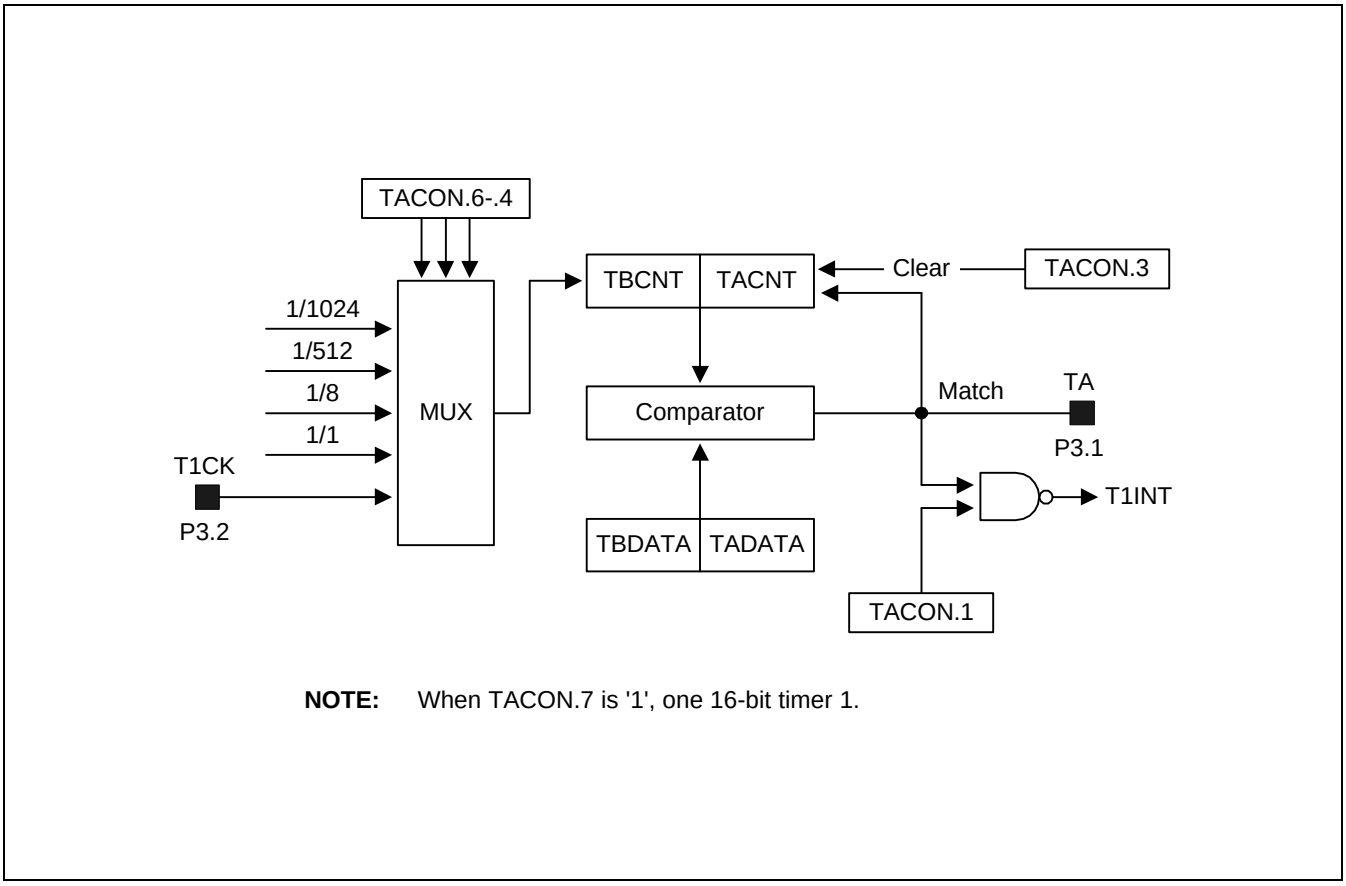


Figure 11-2. Timer 1 Functional Block Diagram

TWO 8-BIT TIMERS MODE (TIMER A and B)

OVERVIEW

The 8-bit timer A and B are the 8-bit general-purpose timers. Timer A and B have the interval timer mode by using the appropriate TACON and TBCON setting, respectively.

Timer A and B have the following functional components:

— Clock frequency divider with multiplexer
  – fxx divided by 1024, 512, 8, or 1 and T1CK (External clock) for timer A
  – fxx divided by 1024, 512, 8, or 1 for timer B

— 8-bit counter (TACNT, TBCNT), 8-bit comparator, and 8-bit reference data register (TADATA, TBDATA)

— Timer A match interrupt (IRQ1, vector F4H) generation

— Timer A control register, TACON (set 1, F6H, read/write)

— Timer B match interrupt (IRQ1, vector F6H) generation

— Timer B control register, TBCON (set 1, F7H, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

The timer A and B module can generate an interrupt: the timer A match interrupt (TAINT) and the timer B match interrupt (TBINT). TAINT belongs to interrupt level IRQ1, and is assigned the separate vector address, F4H. TBINT belongs to interrupt level IRQ1 and is assigned the separate vector address, F6H.

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

In interval timer mode, a match signal is generated when the counter value is identical to the values written to the TA or TB reference data registers, TADATA or TBDATA. The match signal generates corresponding match interrupt (TAINT, vector F4H; TBINT, vector F6H) and clears the counter.

If, for example, you write the value 10H to TBDATA, "0" to TACON.7, and 0EH to TBCON, the counter will increment until it reaches 10H. At this point, the TB interrupt request is generated, the counter value is reset, and counting resumes.

Timer A and B Control Register (TACON, TBCON)

You use the timer A and B control register, TACON and TBCON, to

— Enable the timer A and B operating (interval timer)

— Select the timer A and B input clock frequency

— Clear the timer A and B counter, TACNT and TBCNT

— Enable the timer A and B interrupt

— Clear timer A and B interrupt pending conditions

TACON and TBCON are located in set 1, at address F6H and F7H, and is read/write addressable using register addressing mode.

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

To enable the timer A and B interrupt (IRQ1, vector F4H, F6H), you must write TACON.7 to "0", TACON.2 (TBCON.2) and TACON.1 (TBCON.1) to "1". To generate the exact time interval, you should write TACON.3 (TBCON.3) and TACON.0 (TBCON.0), which cleared counter and interrupt pending bit. To detect an interrupt pending condition when TAINT and TBINT is disabled, the application program polls pending bit, TACON.0 and TBCON.0. When a "1" is detected, a timer A and B interrupt is pending. When the TAINT and TBINT sub-routine has been serviced, the pending condition must be cleared by software by writing a "0" to the timer A and B interrupt pending bit, TACON.0 and TBCON.0.

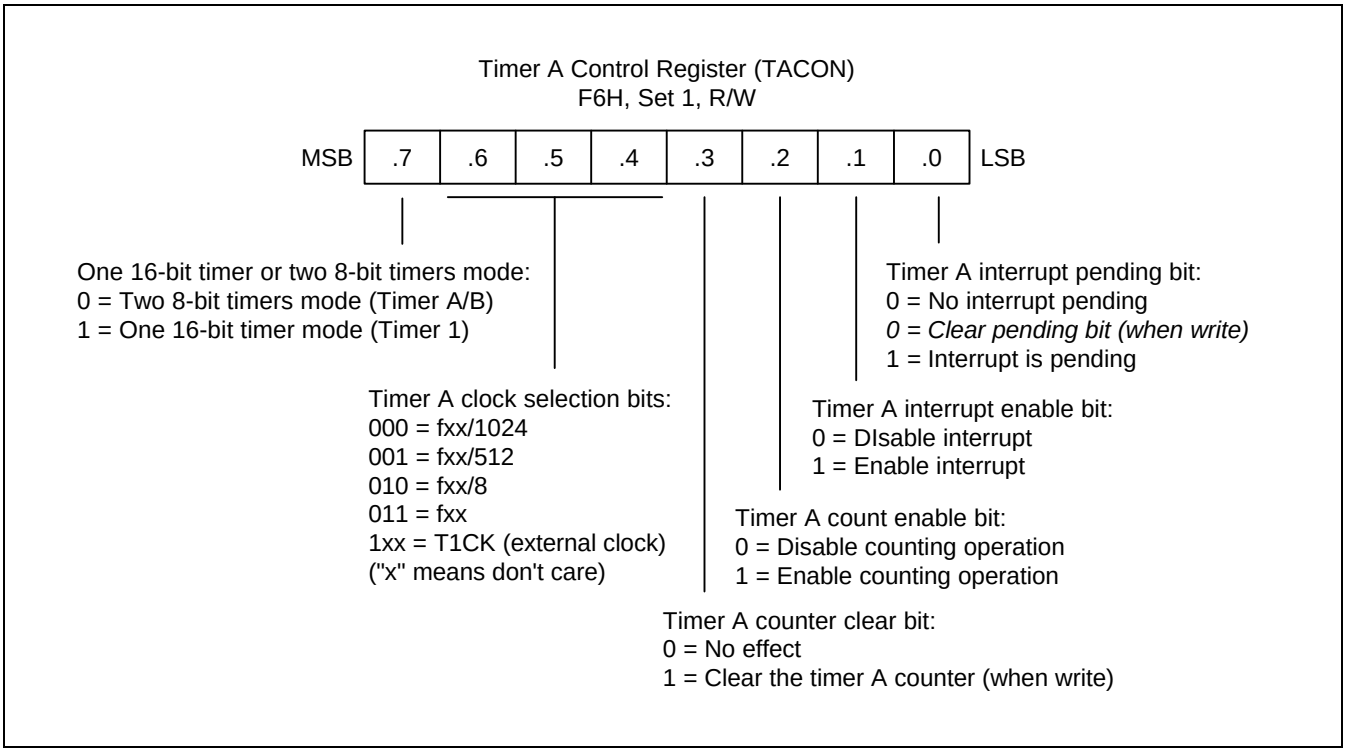


Figure 11-3. Timer A Control Register (TACON)

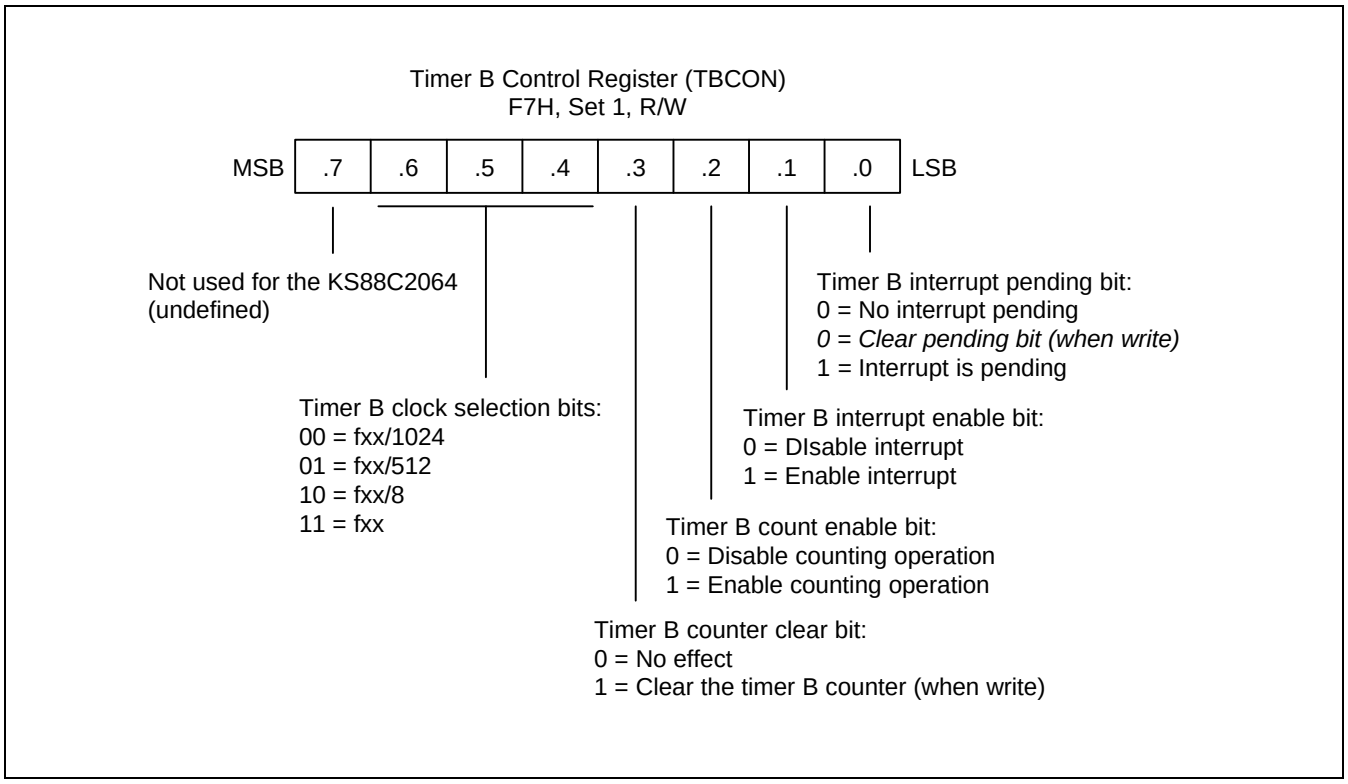
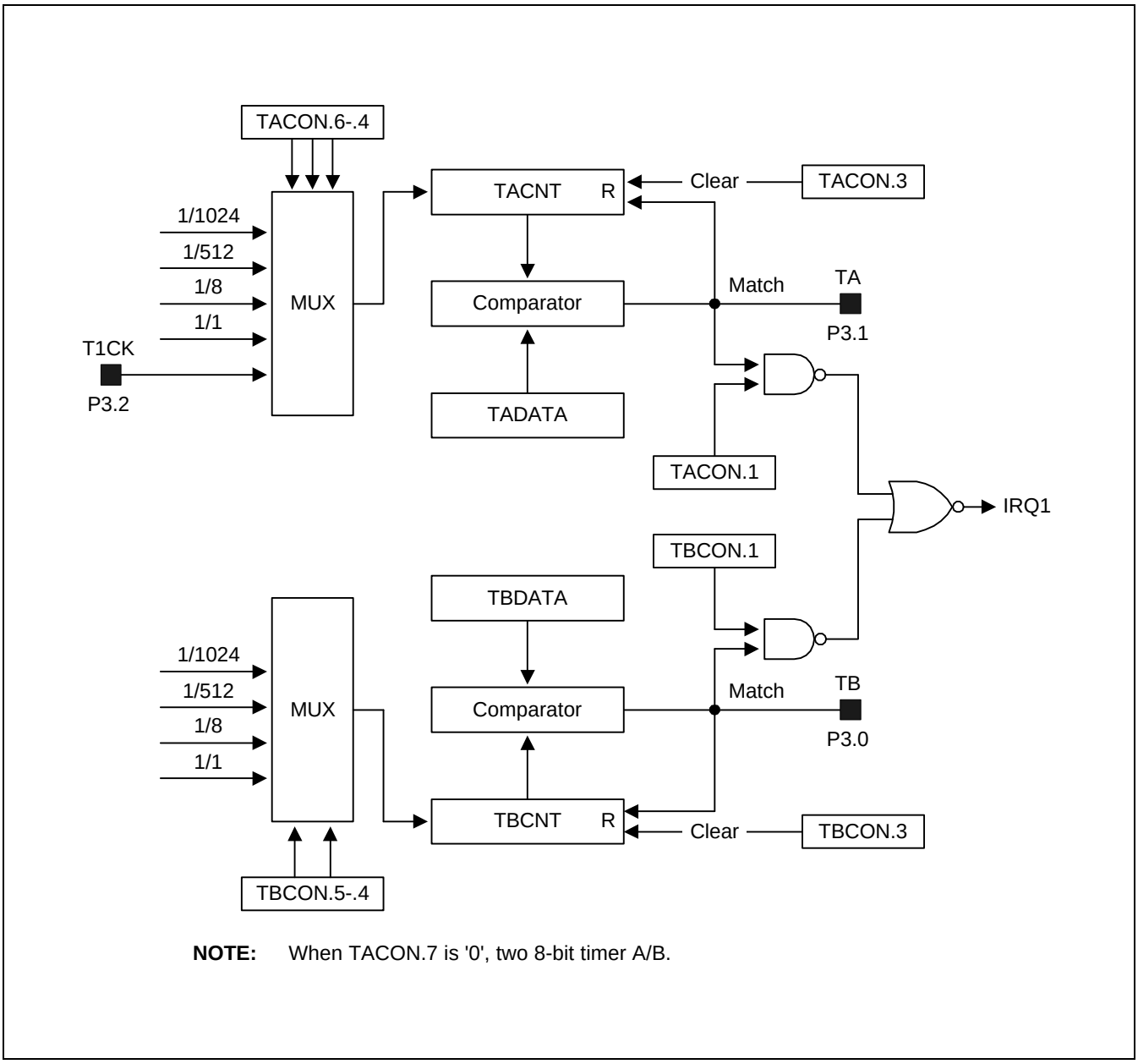


Figure 11-4. Timer B Control Register (TBCON)



NOTE: When TACON.7 is '0', two 8-bit timer A/B.

Figure 11-5. Timer A and B Function Block Diagram

12 WATCH TIMER

OVERVIEW

Watch timer functions include real-time and watch-time measurement and interval timing for the system clock. After the watch timer starts and elapses a time, the watch timer interrupt is automatically set to "1", and interrupt requests commence in 3.91 ms, or 1 second and 1 second at 32.768 kHz (fxt) clock source.

The watch timer can generate a steady 2 kHz, 4 kHz, 8 kHz, or 16 kHz signal to the BUZZER output. By setting WTCON.3 and WTCON.0 to "11b", the watch timer will function in high-speed mode, generating an interrupt every 3.91 ms. High-speed mode is useful for timing events for program debugging sequences. The watch timer supplies the clock frequency for the LCD controller (f_{LCD}). Therefore, if the watch timer is disabled, the LCD controller does not operate.

— Real-time and Watch-time measurement
— Using a main system or subsystem clock source
— Clock source generation for LCD controller
— Buzzer output frequency generator
— Timing tests in high-speed mode

WATCH TIMER CONTROL REGISTER (WTCON: R/W)

F8H	WTCON.7	WTCON.6	WTCON.5	WTCON.4	WTCON.3	WTCON.2	WTCON.1	WTCON.0
RESET	"0"	"0"	"0"	"0"	"0"	"0"	"0"	"0"

Table 12-1. Watch Timer Control Register (WTCON): 8-bit R/W

Bit Name	Values		Function	Address
WTCON.7	0		Disable buzzer (BUZ) signal output	F8H
	1		Enable buzzer (BUZ) signal output	
WTCON.6	0		Always logic "0"	
WTCON.5–.4	0	0	2 kHz buzzer (BUZ) signal output	
	0	1	4 kHz buzzer (BUZ) signal output	
	1	0	8 kHz buzzer (BUZ) signal output	
	1	1	16 kHz buzzer (BUZ) signal output	
WTCON.3	0		1 second interval (IRQ4, vector F2H)	
	1		3.91 ms interval (IRQ4, vector F2H)	
WTCON.2	0		Select (fx/128) as the watch timer clock	
	1		Select sub clock as watch timer clock	
WTCON.1	0		Disable interrupt (IRQ4, vector F0H) for 1 second	
	1		Enable interrupt (IRQ4, vector F0H) for 1 second	
WTCON.0	0		Disable interrupt (IRQ4, vector F2H) for 3.91 ms or 1 second	
	1		Enable interrupt (IRQ4, vector F2H) for 3.91 ms or 1 second	

NOTES

1. Suppose that fw is 32.768 kHz.
2. When main clock is selected as system clock (fxx), watch timer 1 s/3.91 ms interrupt is used as for 3.91 ms interrupt (WTCON.3 = "1"), and sub clock is selected as watch timer clock, watch timer's 3.91 ms interval interrupt is generated two times at the same time. (If fx ≠ fxx, WTCON.3 ≠ "1", nor WTCON.2 ≠ "1", the interrupt is generated normally.)
   That is, the interrupt is generated once more after executing the interrupt routine. Refer to page 17-1 for using watch timer's 3.91 ms interrupt.

WATCH TIMER CIRCUIT DIAGRAM

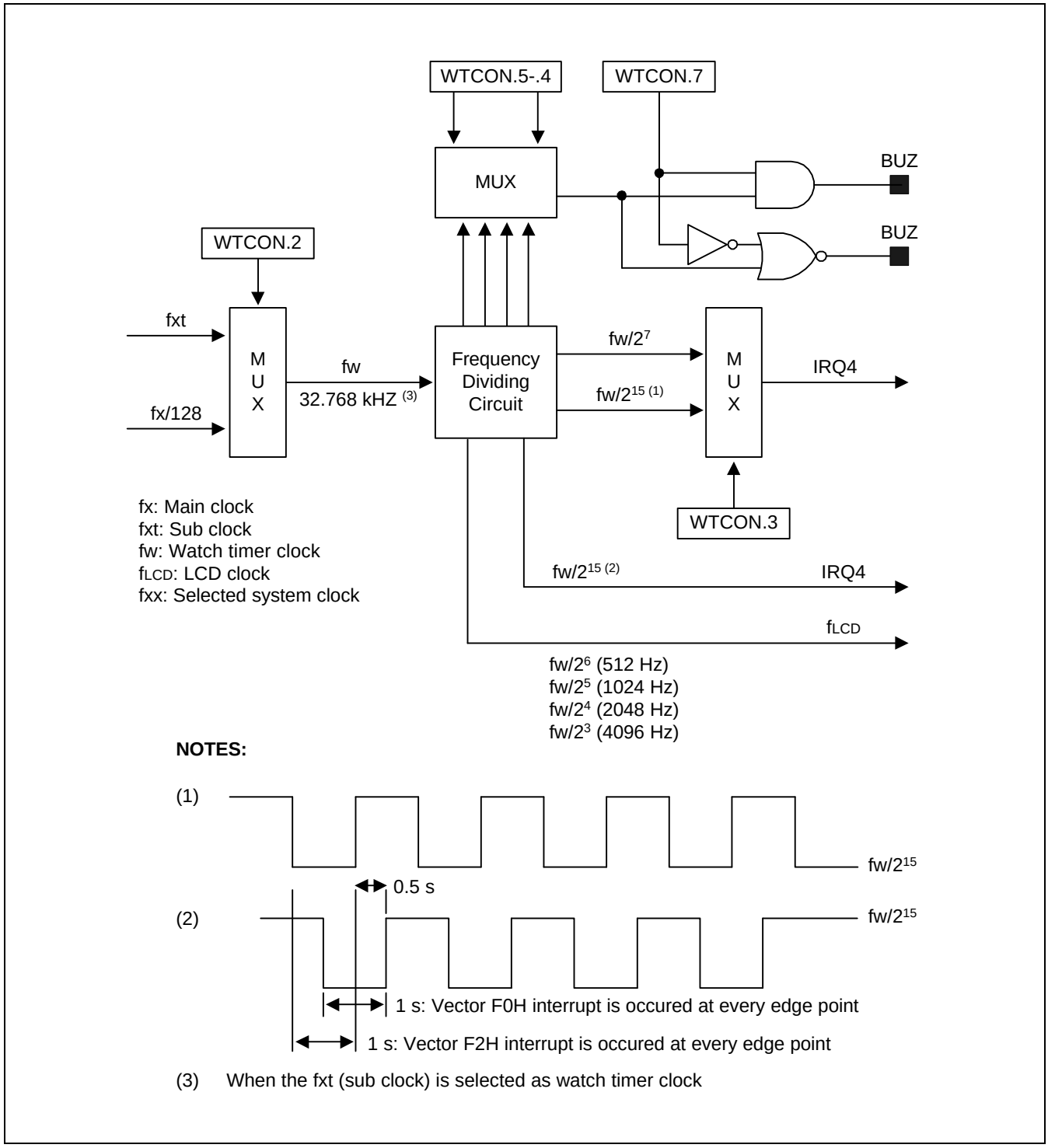


Figure 12-1. Watch Timer Circuit Diagram

13 LCD CONTROLLER/DRIVER

OVERVIEW

The S3C820B microcontroller can directly drive an up-to-1170-dot (65 segments x 18 commons) LCD panel. Its LCD block has the following components:

— LCD controller/driver
— Display RAM for storing display data
— 65 segment output pins (SEG0–SEG64)
— 18 common output pins (COM0–COM17)
— Internal resistor circuit for LCD bias
— V_{LC0} pin for controlling the driver and bias voltage
— Voltage doubler for LCD (BIAS)

The LCD control register, LCON, is used to turn the LCD display on and off, to switch current to the dividing resistors for the LCD display, frame frequency, and to turn the voltage doubler on and off for LCD bias voltage. Data written to the LCD display RAM can be transferred to the segment signal pins automatically without program control.

When a sub clock is selected as the LCD clock source, the LCD display is enabled even during main clock stop and idle modes.

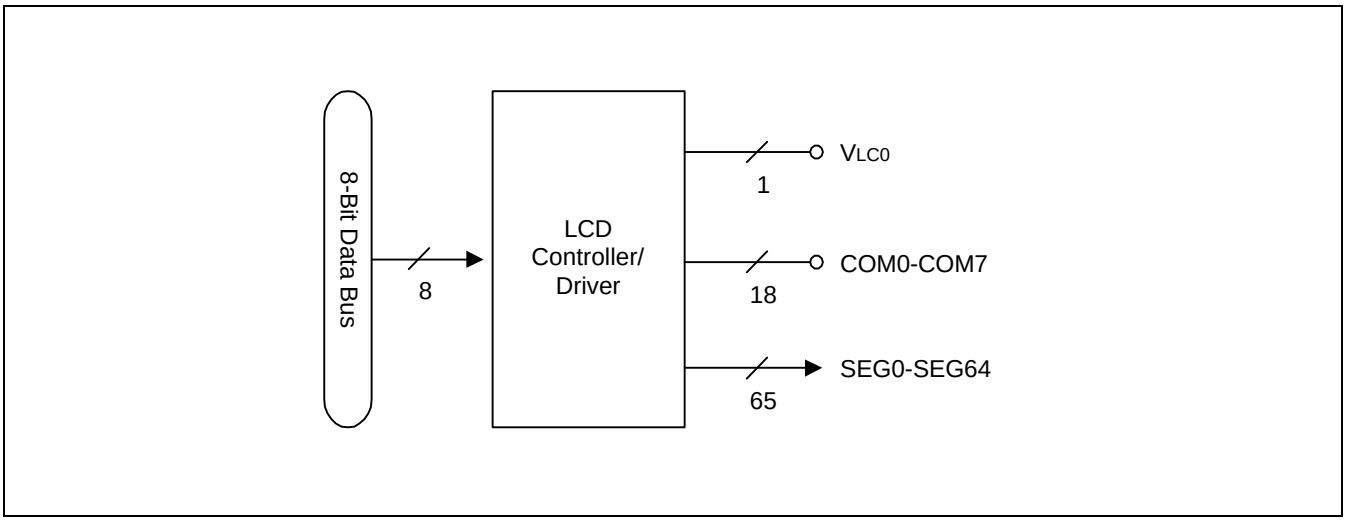


Figure 13-1. LCD Function Diagram

LCD CIRCUIT DIAGRAM

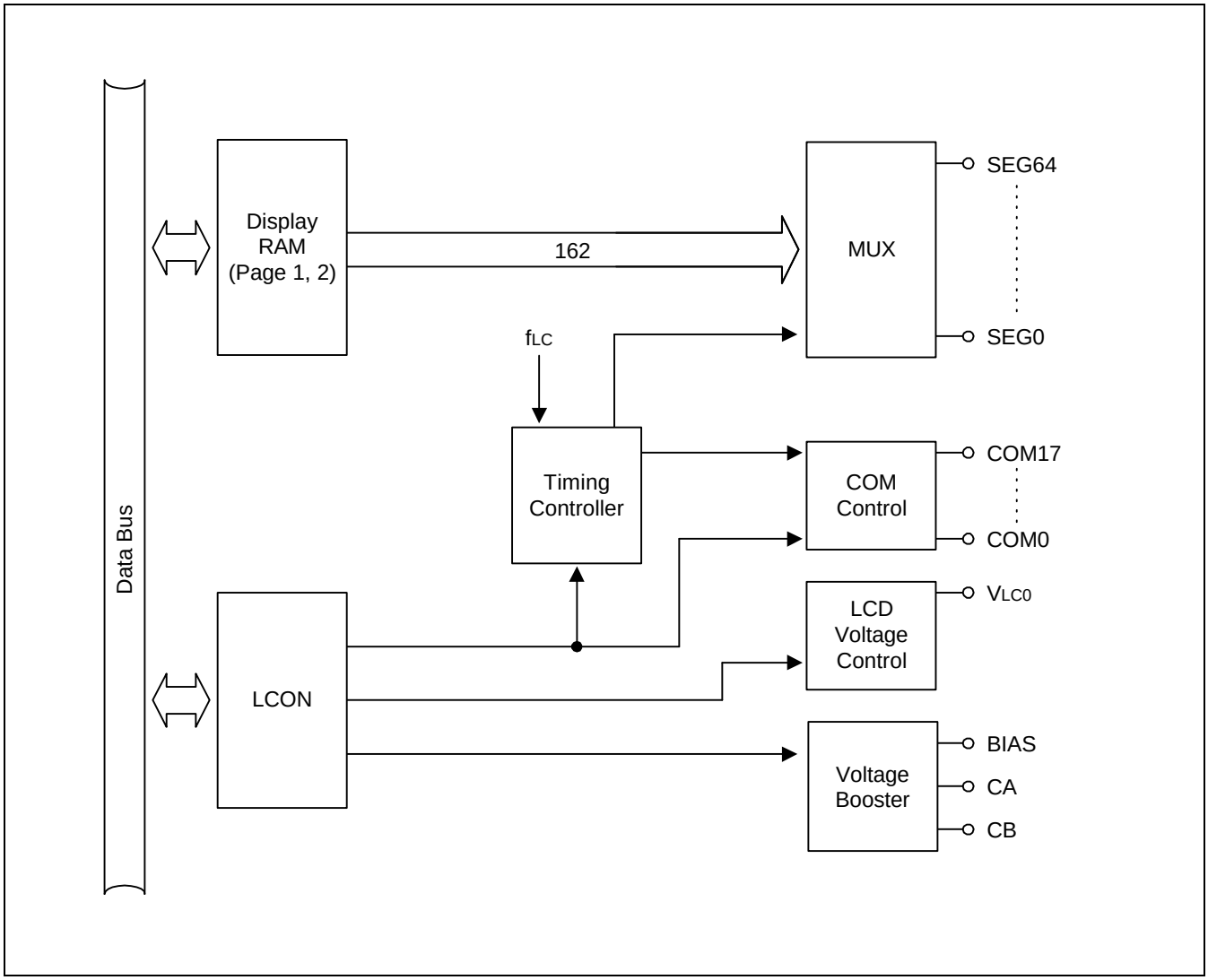


Figure 13-2. LCD Circuit Diagram

LCD RAM ADDRESS AREA

RAM addresses of pages 1, 2 are used as LCD data memory. These locations can be addressed by 1-bit or 8-bit instructions. 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–SEG64 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.

	SEG0 b0	SEG1 b1	SEG2 b2	SEG3 b3	SEG4 b4	SEG5 b5	SEG6 b6	SEG7 b7	SEG8 b0	SEG9 b1	SEG10 b2	SEG11 b3	SEG12 b4	SEG13 b5	SEG14 b6	SEG15 b7		SEG48 b0	SEG49 b1	SEG50 b2	SEG51 b3	SEG52 b4	SEG53 b5	SEG54 b6	SEG55 b7	SEG56 b0	SEG57 b1	SEG58 b2	SEG59 b3	SEG60 b4	SEG61 b5	SEG62 b6	SEG63 b7	SEG64 b0	b1-b7
COM0	100H								110H									160H								170H								180H	
COM1	101H								111H									161H								171H								181H	
COM2	102H								112H									162H								172H								182H	
COM3	103H								113H									163H								173H								183H	
COM4	104H								114H									164H								174H								184H	
COM5	105H								115H									165H								175H								185H	
COM6	106H								116H									166H								176H								186H	
COM7	107H								117H								(Page 1)	167H								177H								187H	
COM8	108H								118H									168H								178H								188H	
COM9	109H								119H									169H								179H								189H	
COM10	10AH								11AH									16AH								17AH								18AH	
COM11	10BH								11BH									16BH								17BH								18BH	
COM12	10CH								11CH									16CH								17CH								18CH	
COM13	10DH								11DH									16DH								17DH								18DH	
COM14	10EH								11EH									16EH								17EH								18EH	
COM15	10FH								11FH									16FH								17FH								18FH	
COM16	200H								210H								(Page 2)	260H								270H								280H	
COM17	201H								211H									261H								271H								281H	

Figure 13-3. LCD Display Data RAM Organization

LCD CONTROL REGISTER (LCON)

The LCD control register (LCON) is used to turn the LCD display on and off, LCD frame frequency, to turn the voltage doubler on and off for LCD bias voltage, and to control the flow of current to dividing resistors in the LCD circuit. Following a RESET, all LCON values are cleared to “0”. This turns the LCD display off and stops the flow of current to the dividing resistors.

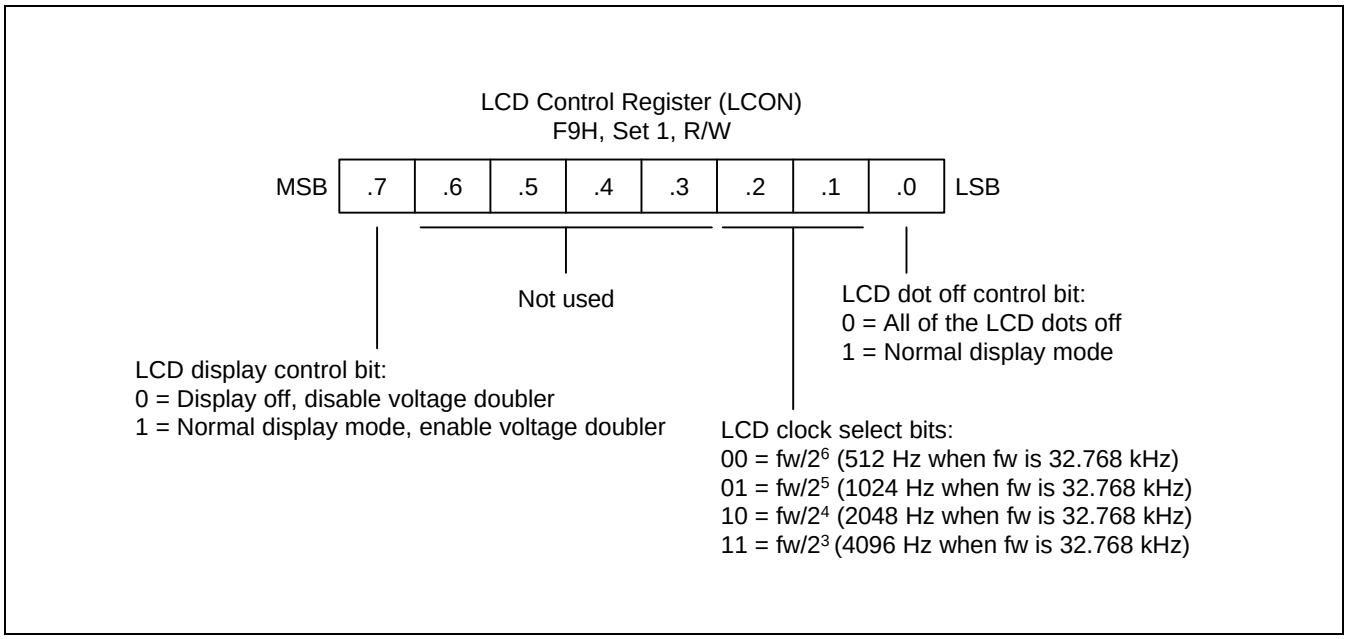
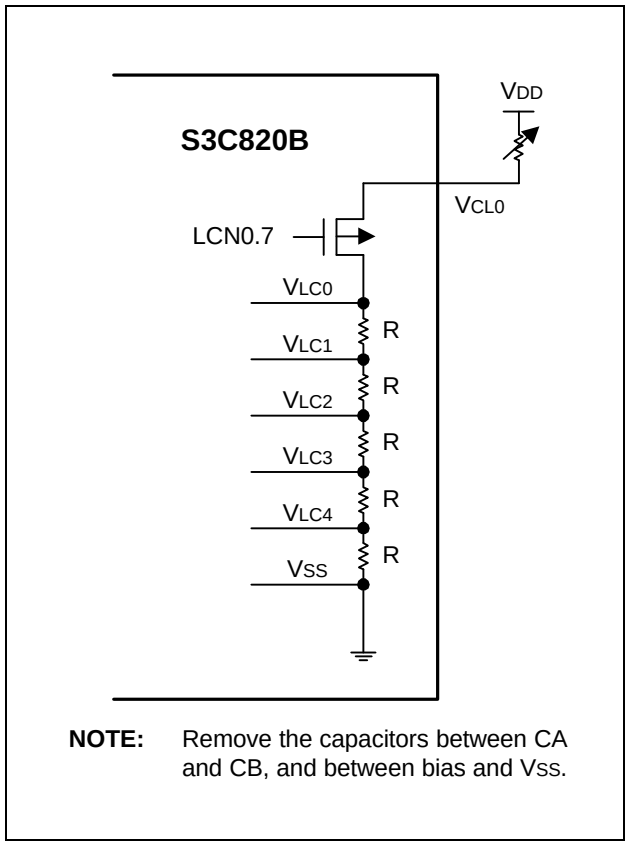


Figure 13-4. LCD Control Register (LCON)



NOTE: Remove the capacitors between CA and CB, and between bias and VSS.

Figure 13-5. Internal Voltage Dividing Resistor Connection (LCON.7 = 1)

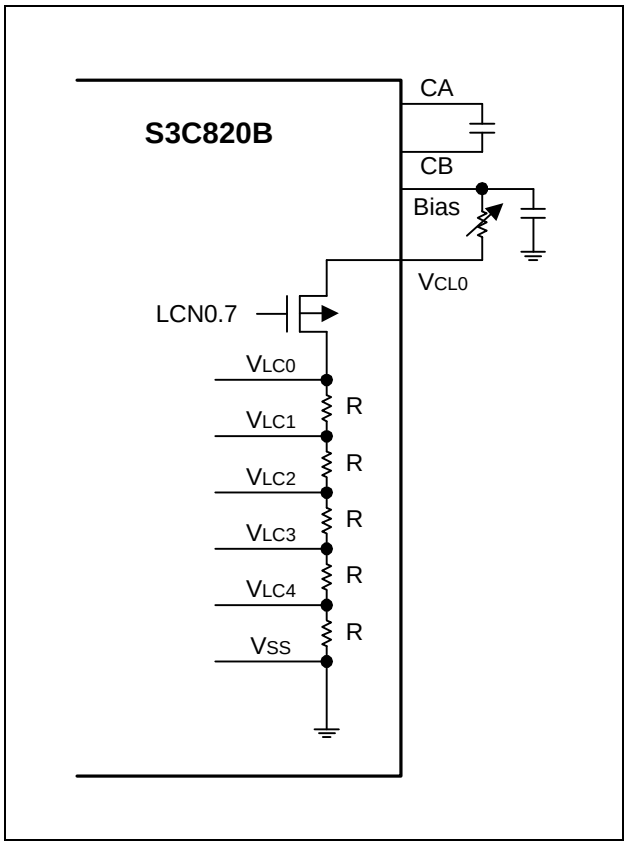


Figure 13-6. Internal Voltage Dividing Resistor Connection (LCON.7 = 1)

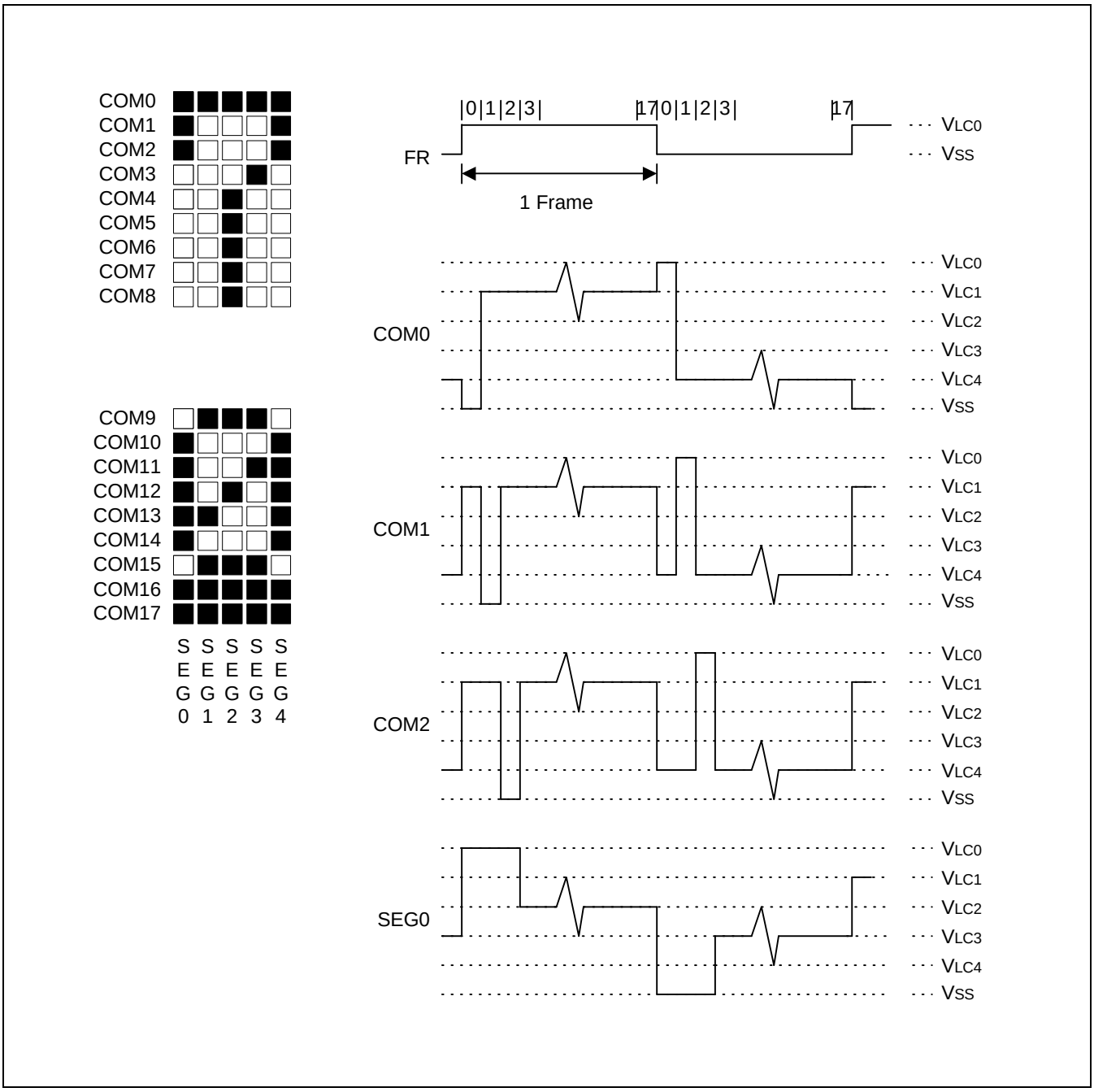


Figure 13-7. LCD Signal Waveforms (1/18 Duty, 1/5 Bias)

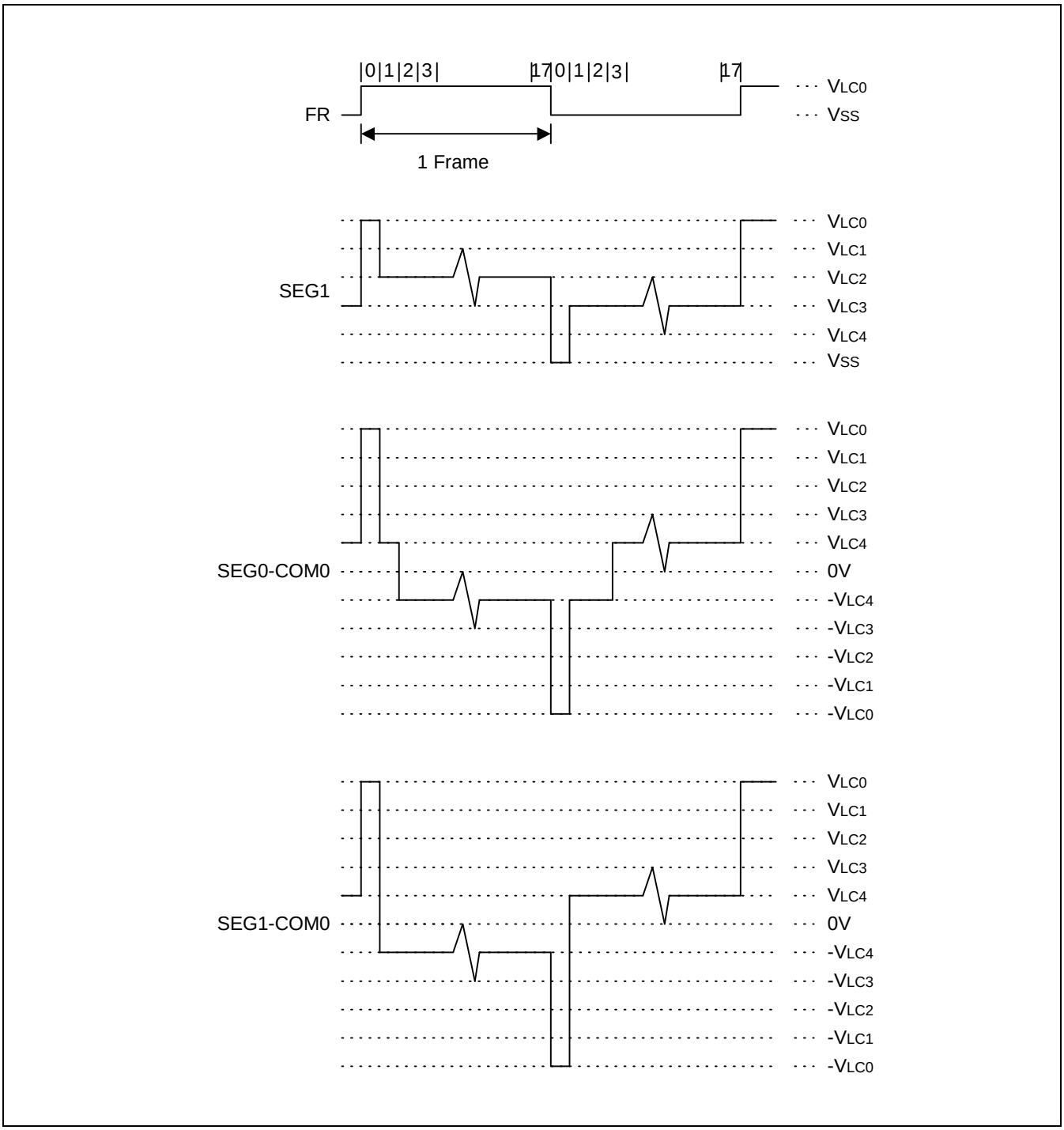


Figure 13-7. LCD Signal Waveforms (1/18 Duty, 1/5 Bias) (Continued)

14 INTERNAL AND EXTERNAL MEMORY

OVERVIEW

The S3C820B has 192K-bytes of font ROM, and 6K-bytes of data RAM, for the purpose of data storage. Also, if more data storage capacity is desired, you can connect more memory externally.

First, there are three 64K-byte font ROMs, which are named font ROM 0, 1, and 2. The 6K-bytes of data RAM is named data RAM.

When using the DM signal, the external memory can be extended up to 64K-byte, but is limited to one. Also, you can connect and use more memory if you use port instead of the DM signal to control the memory CS pin.

Memory Control Register

The memory control register (FDCON) chooses one out of font ROM 0, 1, 2, data RAM, or external memory (DM active), and turns it into active mode. For example, if you write “00000001” in the FDCON register and use LDE and/or related instructions, you can access font ROM 1. If you write “00000100” and use LDE and/or related instructions, you can access external memory.

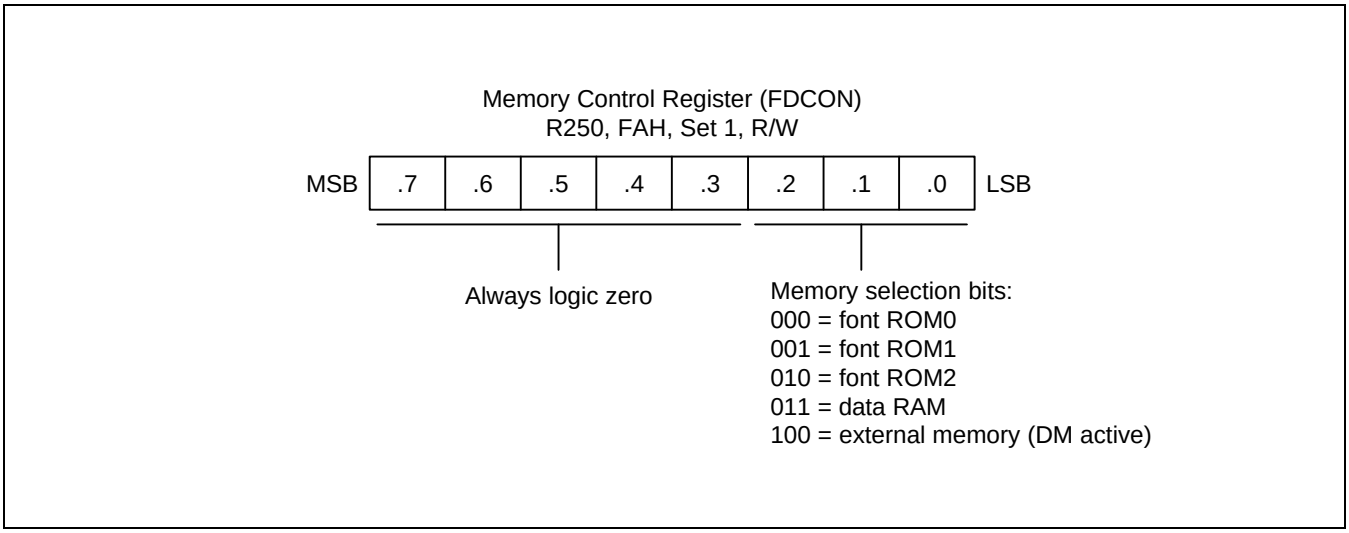


Figure 14-1. Memory Control Register (FDCON)

MEMORY STRUCTURE (FONT and DATA MEMORY)

The internal memory is composed of three 64K-byte font ROMs, and a 6K-byte data RAM. The structure of the font FOM and the data RAM are as follows.

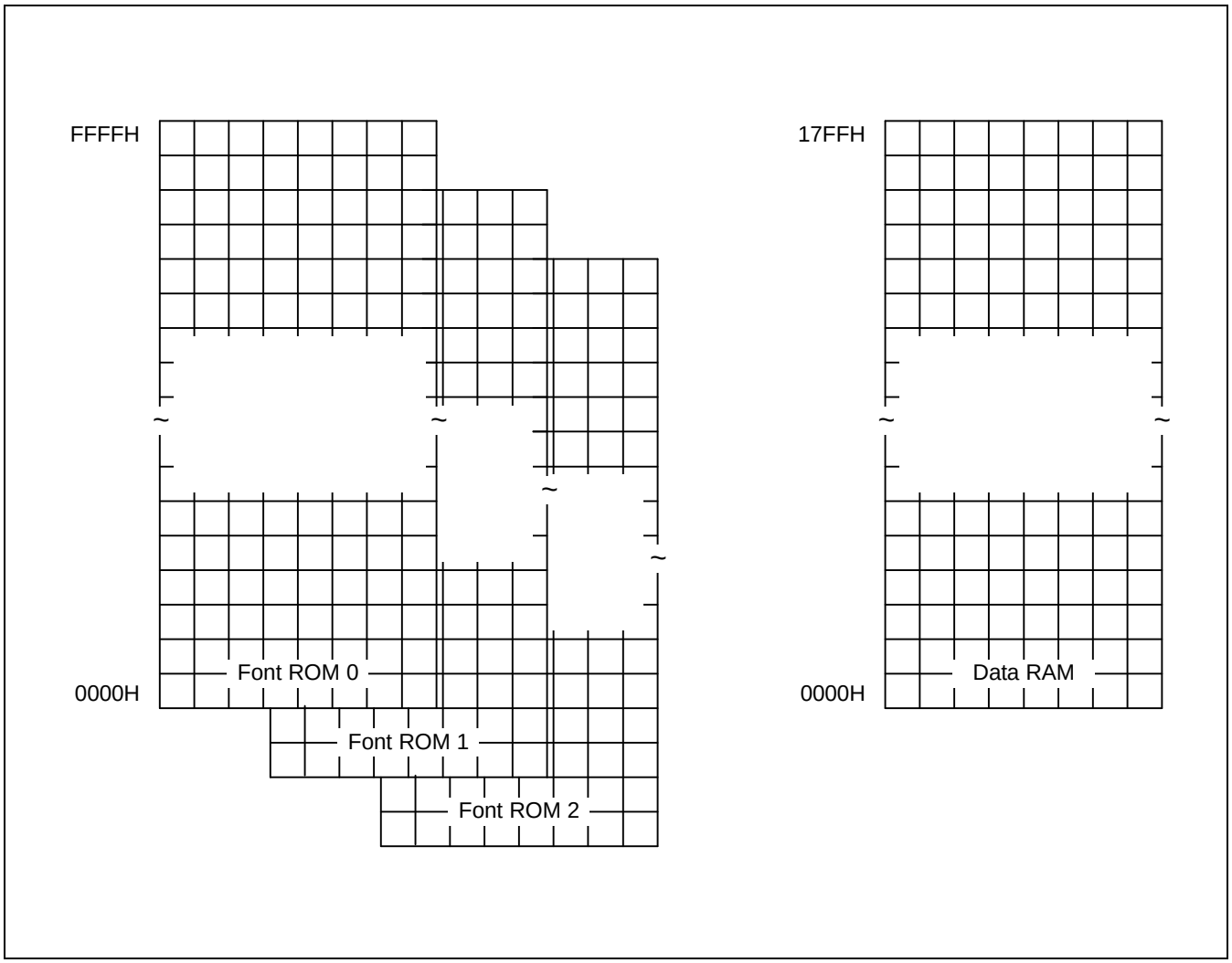


Figure 14-2. Font ROM and Data RAM Structure

EXTERNAL INTERFACE CONTROL REGISTERS

This subsection presents an overview of the S3C820B system registers which are used to configure and control the external peripheral interface:

— System mode register (SYM, R222, DEH, set 1)

— Port 0 control register (P0CON, R229, E5H, set 1)

— Port 1 control register (P1CON, R230, E6H, set 1)

— Port 2 low-byte control register (P2CONL, R233, E9H, set 1)

Detailed descriptions of each of these registers can be found in Part I, Chapter 4, “Control Registers”.

PORT 0 CONTROL REGISTER (P0CON)

The port 0 control register P0CON (E5H, set 1) controls the upper and lower nibble configuration for port 0 pins. When P0CON bit 7 = “1”, the upper nibble pins P0.7–P0.4 are configured as lines for the external memory interface. When P0CON bit 3 = “1”, the lower nibble pins P0.3–P0.0 are configured for external memory. After a reset, P0CON is cleared to 00H. Bits 7 and/or 3 must then be set to “1” by program software to enable the external memory interface function for port 0.

PORT 1 CONTROL REGISTER (P1CON)

The port 1 control register P1CON (E6H, set 1) functions identically to the P0CON register, except that it controls the upper and lower nibble configuration for port 1 pins: P1.7–P1.4 and P1.3–P1.0, respectively. P1CON is also cleared to 00H by a reset.

PORT 2 CONTROL REGISTER (P2CONL)

The pins of I/O port 2, P2.3–P2.0, can alternately be used as lines for the signal outputs that are required to control the activity of the multiplexed external memory interface bus. You manipulate the bit-pairs in the control register, P2CONL (E9H, set 1) to configure the pins individually for general-purpose use or as external memory bus control lines. If the external memory interface is implemented, you must configure all four pins as memory lines. When bit pairs 7/6, 5/4, 3/2, and 1/0 are set to “11B”, the corresponding memory signal outputs are activated:

Bit-Pair	Pin	Symbol	Function
7/6	P2.3	DM	Data memory pin
5/4	P2.2	DR	Data read pin
3/2	P2.1	DW	Data write pin
1/0	P2.0	AS	Address strobe pin

In normal operating mode, a reset operation clears P2CONL to 00H, configuring P2.3–P2.0 as normal schmitt trigger input pins.

HOW TO CONFIGURE THE EXTERNAL INTERFACE

The 3-state external memory interface is enabled or disabled by manipulating bit 7 of the system mode register SYM (R222, DEH). A reset clears SYM.7 to logic zero, disabling the high impedance levels of the interface bus lines and enabling the external interface.

HOW TO ACCESS THE INTERNAL AND EXTERNAL MEMORY

To access the font ROM and data RAM or external memory, write the value in the FDCON register and make it into active mode. Use the following instructions.

— LDE (Load internal/external data memory)
— LDED (Load internal/external data memory and decrement)
— LDEI (Load internal/external data memory and increment)
— LDEPD (Load internal/external data memory with pre-decrement)
— LDEPI (Load internal/external data memory with pre-increment)

To access the external memory, its port must be selected to external interface lines.

Table 14-1. Control Register Overview for the External Memory Interface

Register	Location	Bit(s)	Description
SYM	DEH	7	External 3-state interface enable bit
P0CON	E5H	3	If “1”, enable port 0 pins P0.0–P0.3 for external memory interface
		7	If “1”, enable port 0 pins P0.4–P0.7 for external memory interface
P1CON	E6H	3	If “1”, enable port 1 low nibble pins (P1.0–P1.3) for external memory interface
		7	If “1”, external memory interface enable for port 1 high nibble pins (P1.4–P1.7)
P2CONL	E9H	1, 0	If both “1”, address strobe (AS) enabled at P2.0
		3, 2	If both “1”, data write signal (DW) enabled at P2.1
		5, 4	If both “1”, data read signal (DR) enabled at P2.2
		7, 6	If both “1”, data memory signal (DM) enabled at P2.3

Table 14-2. External Interface Control Register Values After a RESET (Normal Mode)

Register Name	Mnemonic	Address		Bit Values After RESET (EA Pin is Low)							
		Dec	Hex	7	6	5	4	3	2	1	0
System Mode Register	SYM	R222	DEH	0	–	–	x	x	x	0	0
Port 0 Control Register	P0CON	R229	E5H	0	0	0	0	0	0	0	0
Port 1 Control Register	P1CON	R230	E6H	0	0	0	0	0	0	0	0
Port 2 Control Register (Low Byte)	P2CONL	R233	E9H	0	0	0	0	0	0	0	0

NOTE: A dash (–) indicates that the bit is not mapped; an “x” means that the value is undefined after a RESET.

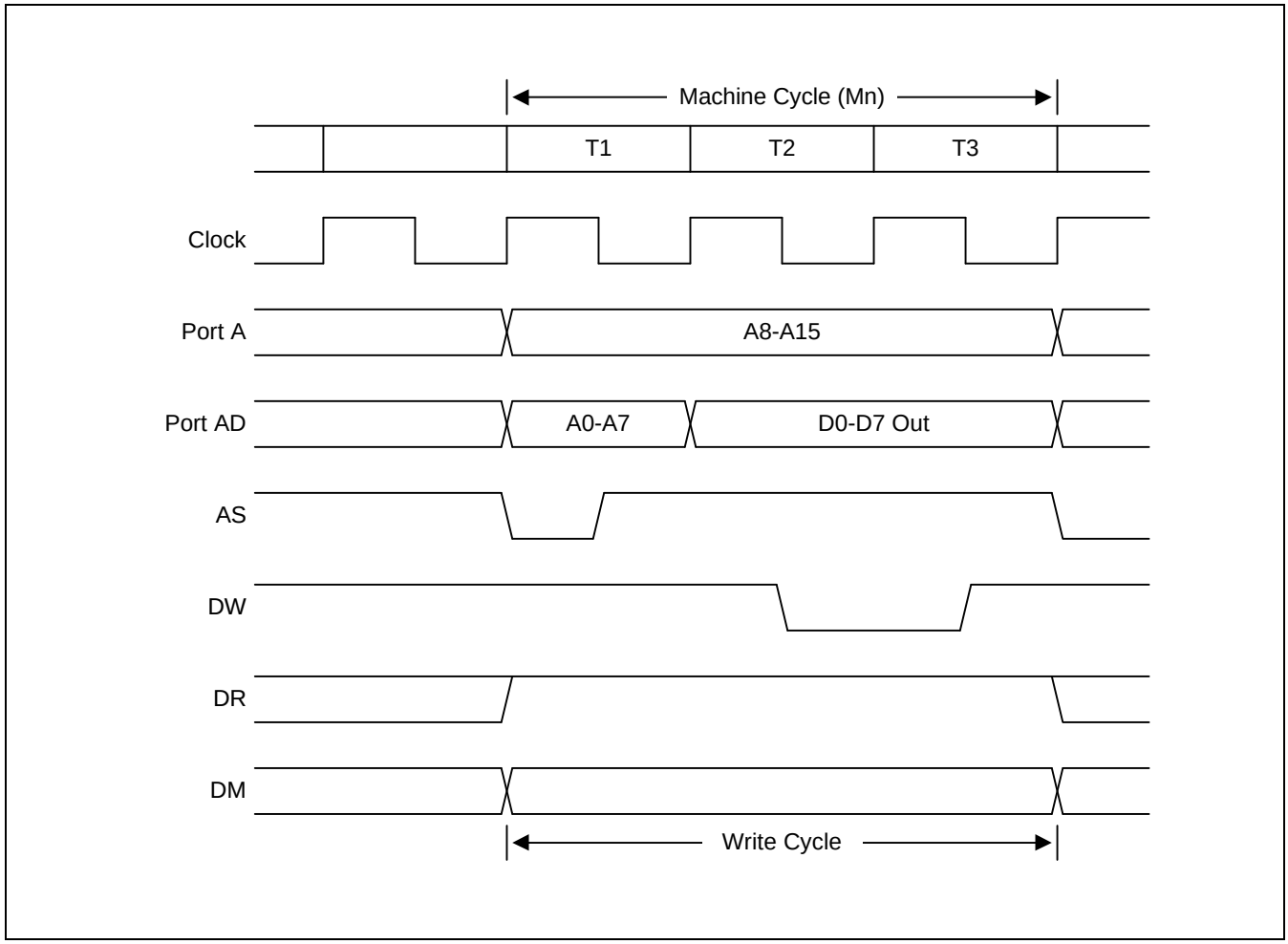


Figure 14-3. S3C820B External Bus Write Cycle Timing Diagram

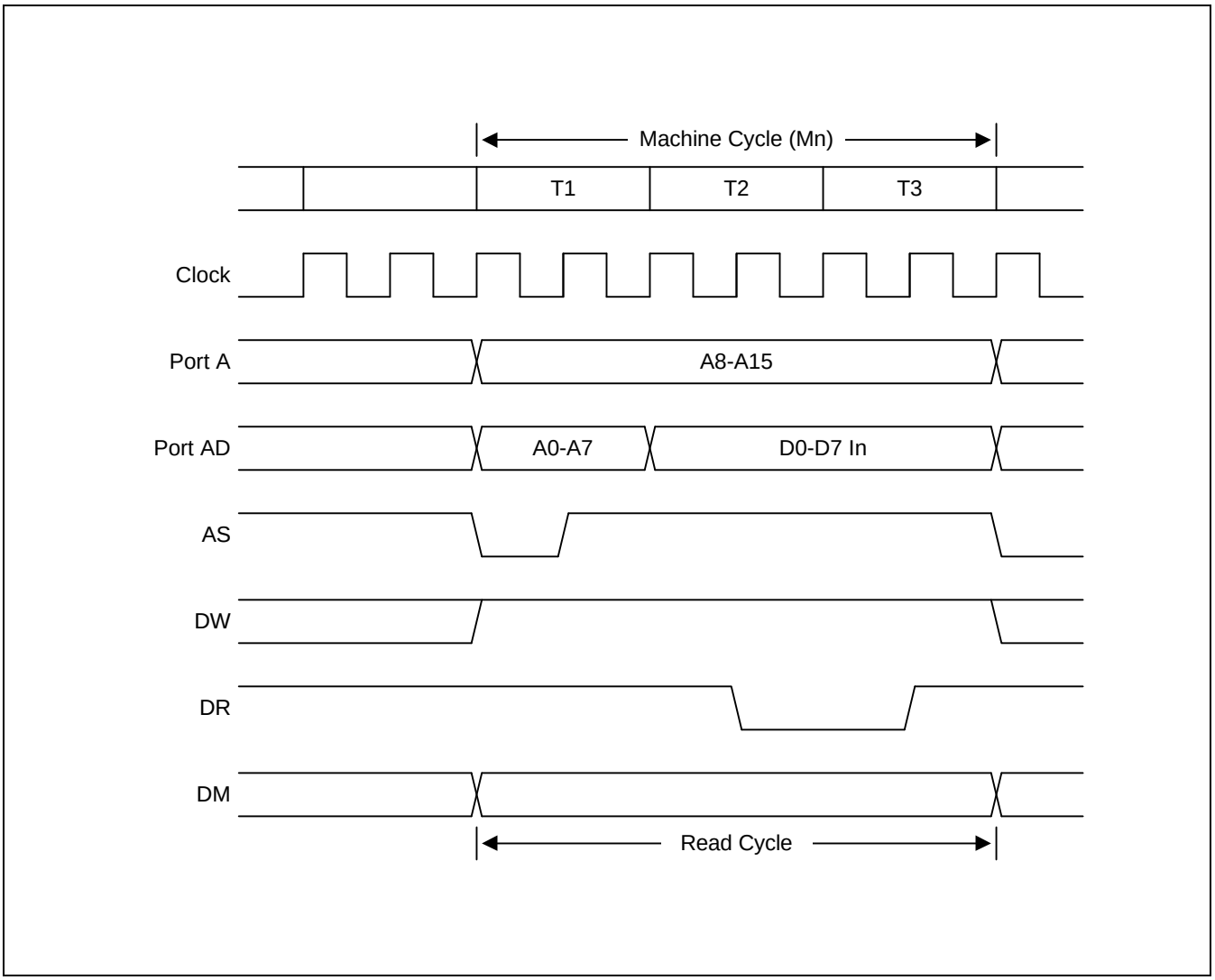


Figure 14-4. S3C820B External Bus Read Cycle Timing Diagram

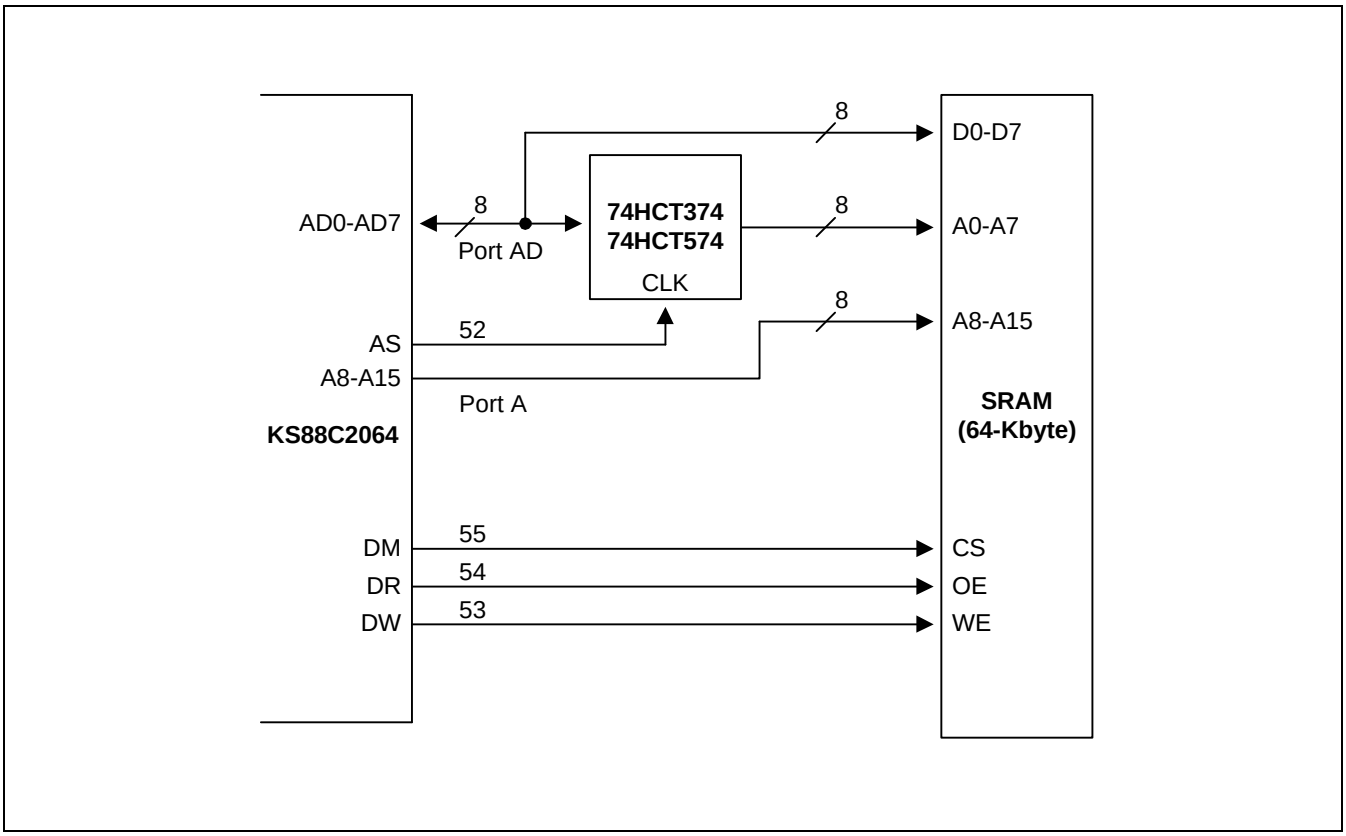


Figure 14-5. External Interface Function Diagram (S3C820B, SRAM, EPROM, EEPROM)

15 ELECTRICAL DATA

OVERVIEW

In this chapter, S3C820B electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

— Absolute maximum ratings
— D.C. electrical characteristics
— Data retention supply voltage in Stop mode
— Stop mode release timing when initiated by an external interrupt
— Stop mode release timing when initiated by a Reset
— I/O capacitance
— A.C. electrical characteristics
— Input timing for external interrupts (port 0, P2.3–P2.0)
— Input timing for RESET
— Oscillation characteristics
— Oscillation stabilization time

Table 15-1. Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 5.5	V
Input voltage	V_{IN}	Ports 0, 1, 2, and 3	– 0.3 to V_{DD} + 0.3	V
Output voltage	V_O	All output pins	– 0.3 to V_{DD} + 0.3	V
Output current High	I_{OH}	One I/O pin active	– 18	mA
		All I/O pins active	– 60	
Output current Low	I_{OL}	One I/O pin active	+ 30	mA
		Total pin current for ports 0–3	+ 100	
Operating temperature	T_A	–	– 40 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	°C

Table 15-2. D.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.2 V to 4.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High voltage	V_{IH1}	All input pins except V_{IH2} and V_{IH3}	0.8 V_{DD}	–	V_{DD}	V
	V_{IH2}	RESET	0.8 V_{DD}		V_{DD}	
	V_{IH3}	X_{IN}, XT_{IN}	V_{DD} – 0.5		V_{DD}	
Input Low voltage	V_{IL1}	All input pins except V_{IL2} and V_{IL3}	0	–	0.2 V_{DD}	
	V_{IL2}	RESET			0.2 V_{DD}	
	V_{IL3}	X_{OUT}, XT_{OUT}			0.5	
Output High voltage	V_{OH}	V_{DD} = 3.0 V; I_{OH} = – 1 mA All output pins	V_{DD} – 1.0	–	–	
Output Low voltage	V_{OL}	V_{DD} = 3.0 V; I_{OL}= 2 mA All output pins	–	–	1.0	
Input High leakage current	I_{LIH1}	V_{IN} = V_{DD}; all input pins except X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}	–	–	1	µA
	I_{LIH2}	V_{IN} = V_{DD}; X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}			20	
Input Low leakage current	I_{LIL1}	V_{IN} = 0 V; all input pins except X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}	–	–	– 1	
	I_{LIL2}	V_{IN} = 0 V; X_{IN}, X_{OUT}, XT_{IN}, and XT_{OUT}			– 20	
Output High leakage current	I_{LOH}	V_{OUT} = V_{DD} All output pins	–	–	1	
Output Low leakage current	I_{LOL}	V_{OUT} = 0 V All output pins	–	–	– 1	

Table 15-2. D.C. Electrical Characteristics (Continued)

(T_A = – 40°C to + 85°C, V_{DD} = 2.2 V to 4.5 V)

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Middle output voltage	V_{OM1}	$V_{MN} = V_{LCD} – (N/5) \times V_{LCD}$ N = 1, 2, 3, and 4	COM0–17	$V_{M1} – 0.2$	V_{M1}	$V_{M1} + 0.2$	V
	V_{OM2}		SEG0–64	$V_{M2} – 0.2$	V_{M2}	$V_{M2} + 0.2$	
	V_{OM3}		SEG0–64	$V_{M3} – 0.2$	V_{M3}	$V_{M3} + 0.2$	
	V_{OM4}		COM0–17	$V_{M4} – 0.2$	V_{M4}	$V_{M4} + 0.2$	
$\lvert V_{LCD}–V_{COMi}\rvert$ voltage drop (i = 0–17)	V_{DC}	V_{LCD} = 3.0 V to 6.0 V – 15 µA per common pin		–	–	120	mV
$\lvert V_{LCD}–V_{SEGx}\rvert$ voltage drop (x = 0–64)	V_{DS}	V_{LCD} = 3.0 V to 6.0 V – 15 µA per common pin		–	–	120	mV
LCD driving voltage	V_{LCD}	–		3.0	–	6.0	V
Pull-up resistors	R_{L1}	V_{IN} = 0 V; T_A = 25 °C; V_{DD} = 3.0 Ports 0, 1, 2, and 3		30	80	200	kΩ
	R_{L2}	V_{IN} = 0 V; T_A = 25 °C; V_{DD} = 3.0 RESET only		300	500	800	
LCD voltage dividing resistor	R_{LCD}	V_{LCD} = 3.0 V to 6.0 V T_A = 25 °C		40	60	80	kΩ
Supply current (note)	I_{DD1}	V_{DD} = 3.0 V ± 10% 2 MHz crystal		–	1.5	3.5	mA
	I_{DD2}	Idle mode; V_{DD} = 3.0 V ± 10% 2 MHz crystal			0.5	1.5	
	I_{DD3}	V_{DD} = 3.0 V ± 10% 32 kHz crystal			30	70	µA
	I_{DD4}	Idle mode; V_{DD} = 3.0 V ± 10% 32 kHz crystal			6	12	
	I_{DD5}	Stop mode; V_{DD} = 3.0 V ± 10% XT_{IN} = 0 V			0.5	1	

NOTES:

1. Supply current does not include current drawn through internal pull-up resistors, LCD voltage dividing resistors, voltage doubler, or external output current loads.
2. I_{DD1} and I_{DD2} include power consumption for subsystem clock oscillation.
3. I_{DD3} and I_{DD4} are current when main system clock oscillation stops and the subsystem clock is used.

Table 15-3. Data Retention Supply Voltage in Stop Mode

(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	–	2.2	–	4.5	V
Data retention supply current	I_{DDDR}	V_{DDDR} = 2.2 V	–	–	5	µA
Release signal set time	t_{SREL}	–	0	–	–	µs
Oscillator stabilization wait time	t_{WAIT}	Released by RESET	–	2^{16}/fx (1)	–	ms
		Released by interrupt	–	(2)	–	

NOTES:
1. fx is the main oscillator frequency.
2. The duration of the oscillation stabilization time (t_{WAIT}) when it is released by an interrupt is determined by the setting in the basic timer control register, BTCON.

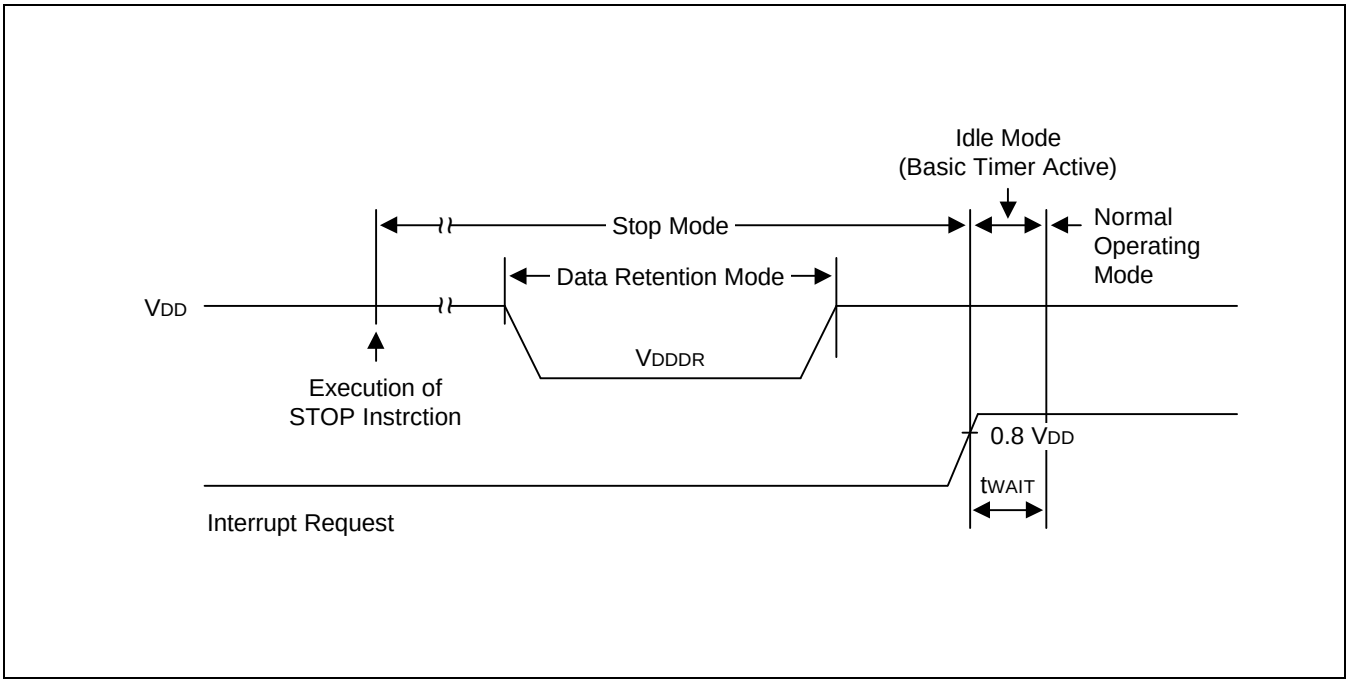


Figure 15-1. Stop Mode Release Timing When Initiated by an External Interrupt

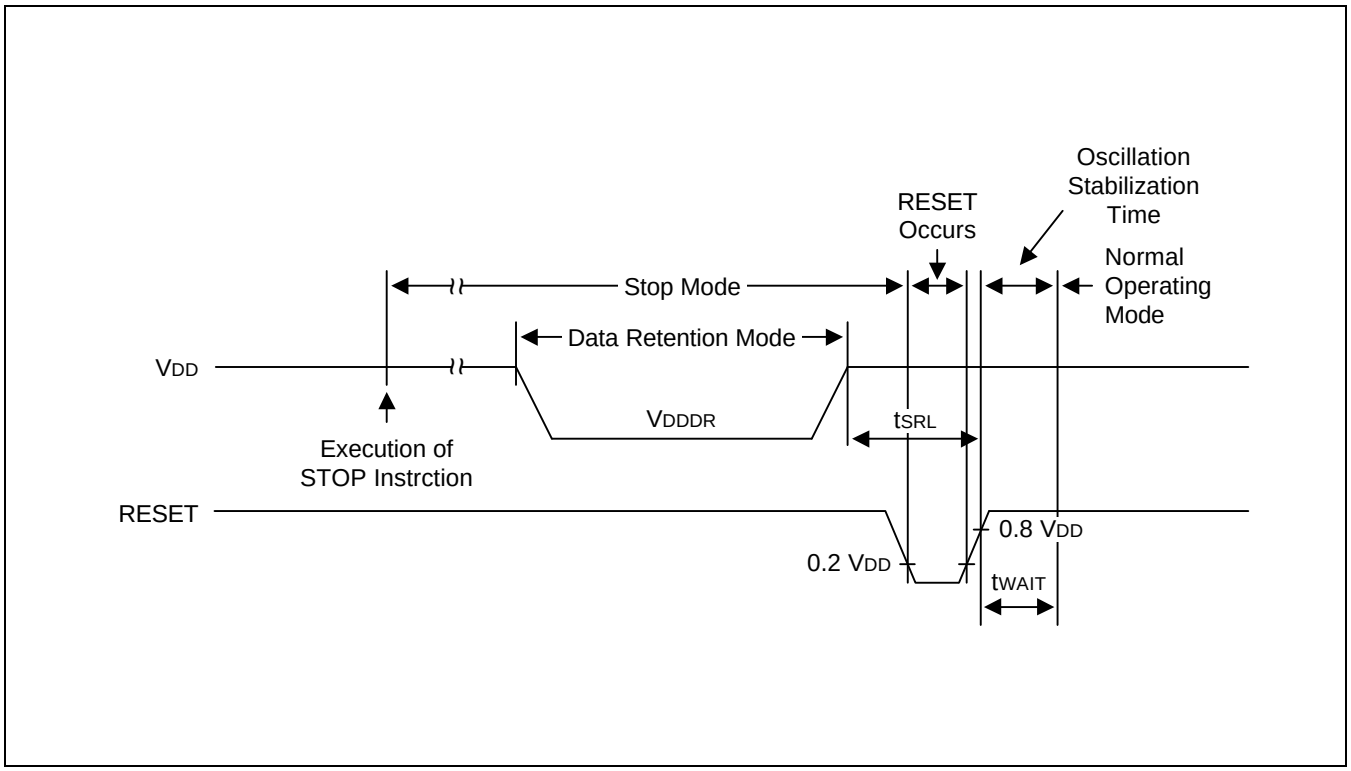


Figure 15-2. Stop Mode Release Timing When Initiated by a RESET

Table 15-4. Input/Output Capacitance

(T_A = – 40 °C to + 85 °C, V_{DD} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input capacitance	C_{IN}	f = 1 MHz; unmeasured pins are connected to V_{SS}	–	–	10	pF
Output capacitance	C_{OUT}					
I/O capacitance	C_{IO}					

Table 15-5. A.C. Electrical Characteristics

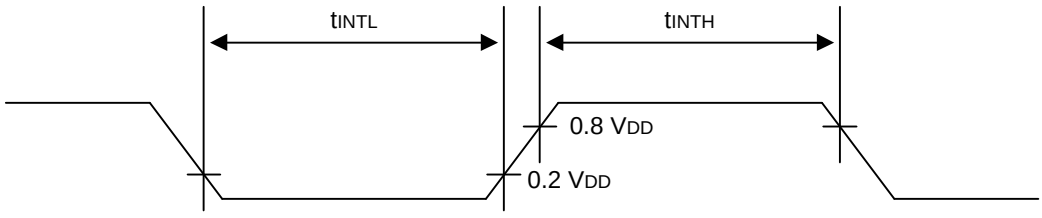
(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input, High, Low width	t_{INTH}, t_{INTL}	P3.0–P3.7 V_{DD} = 3 V	500	700	–	ns
RESET input Low width	t_{RSL}	Input V_{DD} = 3 V	2000	–	–	

Table 15-6. Voltage Doubler Output

(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Voltage Doubler Output	V_{BIAS}	V_{DD} = 3 V ± 10 % only	2 V_{DD} – 0.5	2 V_{DD}	2 V_{DD} + 0.5	V



NOTE: The unit tCPU means one CPU clock period.

Figure 15-3. Input Timing for External Interrupts (P3.0–P3.7)

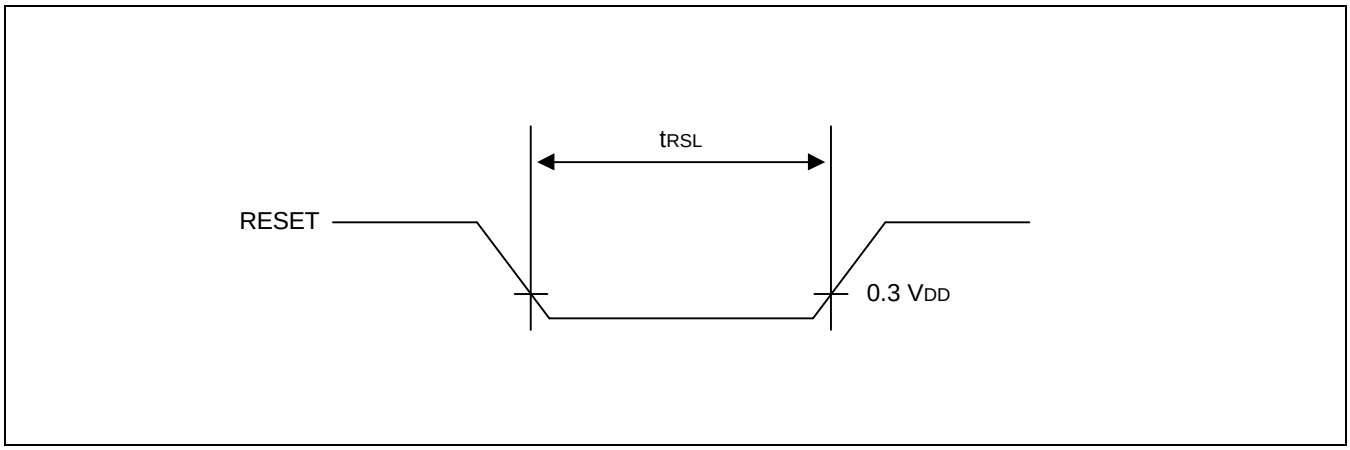


Figure 15-4. Input Timing for RESET

Table 15-7. Main Oscillation Characteristics

(T_A = – 40 °C + 85 °C, V_{DD} = 2.2 V to 4.5 V)

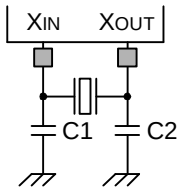
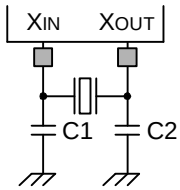
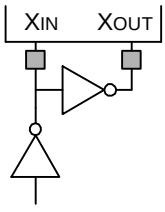
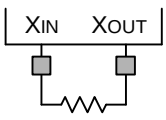
Oscillator	Clock Circuit	Conditions	Min	Typ	Max	Unit
Crystal		CPU clock oscillation frequency	0.4	–	4	MHz
Ceramic		CPU clock oscillation frequency	0.4	–	4	MHz
External clock		X_{IN} input frequency	0.4	–	4	MHz
RC		Frequency, V_{DD} = 3 V	0.4	–	2	MHz

Table 15-8. Sub Oscillation Characteristics

(T_A = – 40 °C + 85 °C, V_{DD} = 2.2 V to 4.5 V)

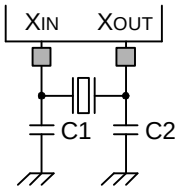
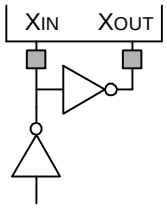
Oscillator	Clock Circuit	Conditions	Min	Typ	Max	Unit
Crystal		CPU clock oscillation frequency	32	32.768	35	kHz
External clock		XT_{IN} input frequency	32	–	500	kHz

Table 15-9. Main Oscillation Stabilization Time

(T_A = – 40 °C + 85 °C, V_{DD} = 3.0 V ± 10 %)

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	fx > 400 kHz	–	–	80	ms
Ceramic	Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	50	ms
External clock	X_{IN} input High and Low width (t_{XH}, t_{XL})	25	–	700	ns

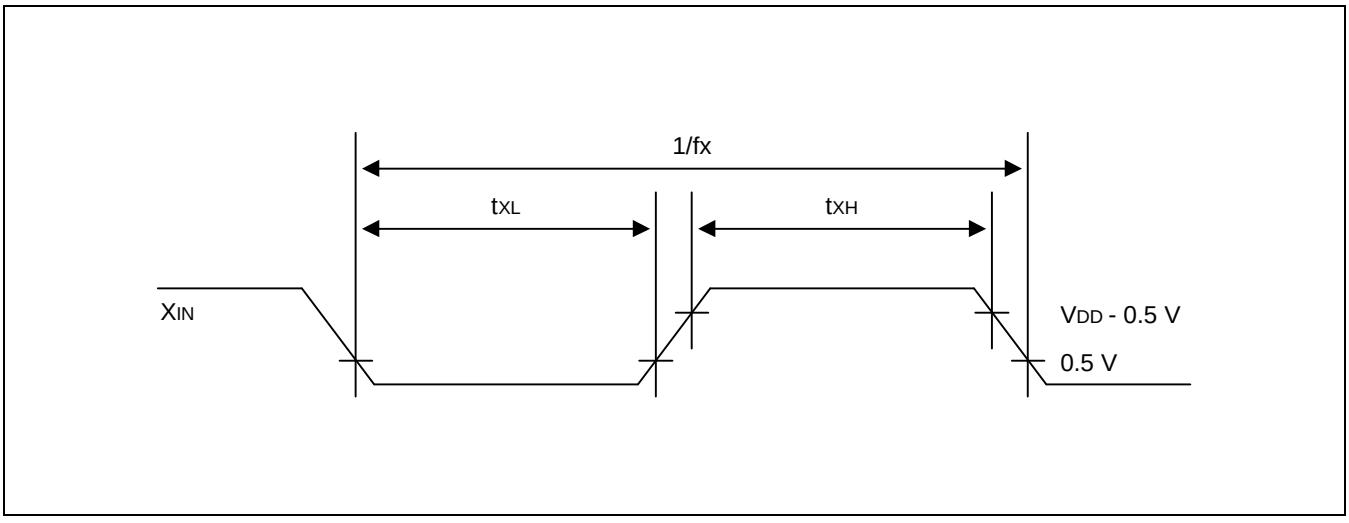


Figure 15-5. Clock Timing Measurement at X_{IN}

Table 15-10. Sub Oscillation Stabilization Time

(T_A = – 40 °C + 85 °C, V_{DD} = 3.0 V ± 10 %)

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	–	–	1.0	2	s
External clock	X_{IN} input High and Low width (t_{XH}, t_{XL})	1	–	18	µs

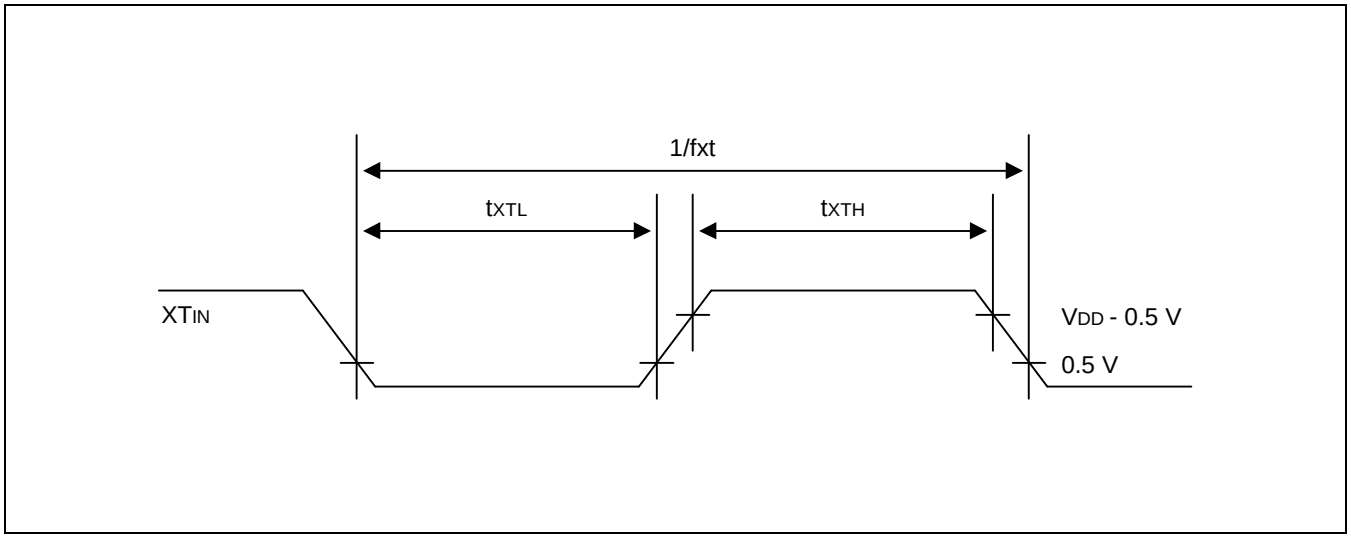
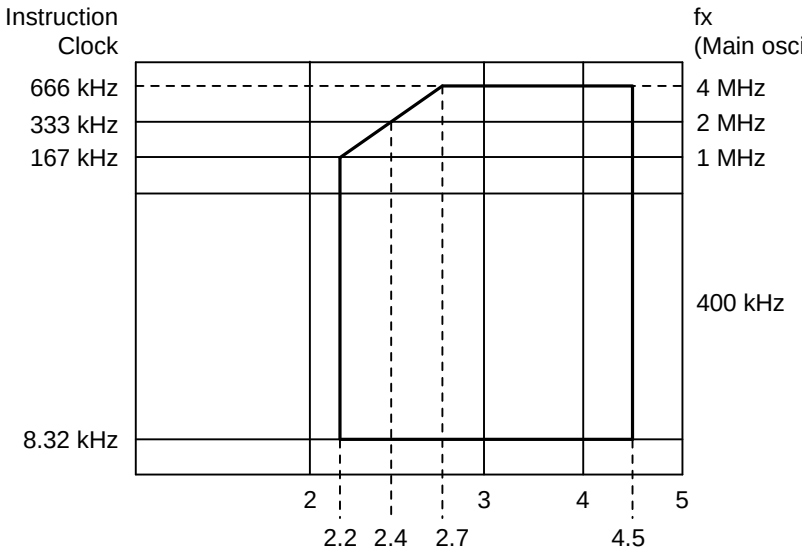


Figure 15-6. Clock Timing Measurement at XT_{IN}



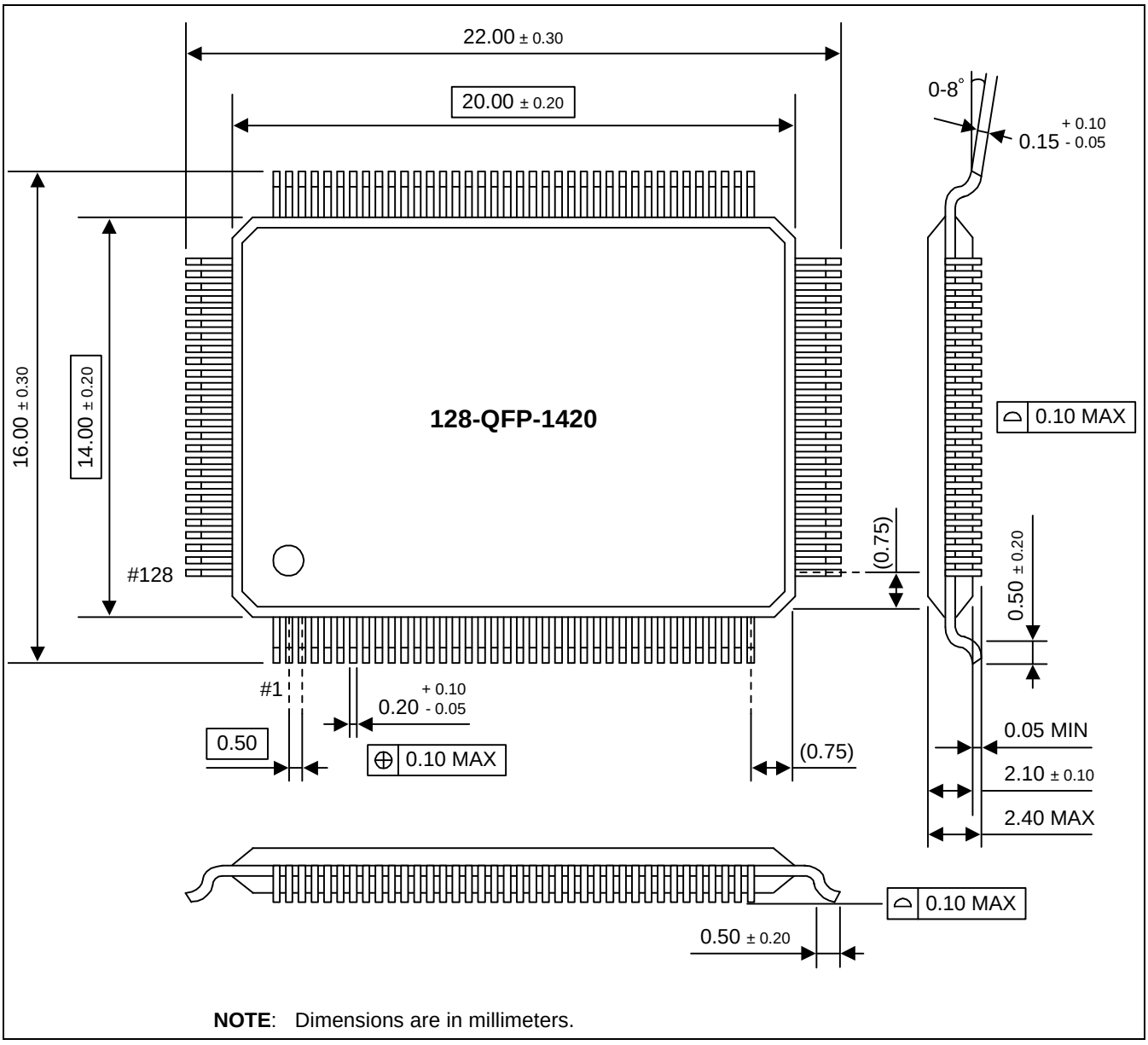
Instruction Clock = 1/6n x oscillator frequency (n = 1, 2, 8, 16)

Figure 15-7. Operating Voltage Range

16 MECHANICAL DATA

OVERVIEW

The S3C820B microcontroller is currently available in a 128-pin TQFP package.



NOTE: Dimensions are in millimeters.

Figure 16-1. 128-Pin TQFP Package Mechanical Data

17 CUSTOMER NOTICE

DJNZ INSTRUCTION

When DJNZ instruction is used in a program, 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.

☞ PROGRAMMING TIP – Using DJNZ instruction to transport RAM data from page 0 to page 1

```
        LD      PP, #10H          ;  Destination ← 1, source ← 0
        SRP     #0C0H
        LD      R0, 0FFH          ;  Transportation starts
RAMTRN  LD      R1, @R0
        LD      @R0, R1
        DJNZ    R0, RAMTRN
        LD      R1, @R0           ;  R0 = 00H
        LD      @R0, R1
```

WATCH TIMER'S 3.91 ms INTERVAL INTERRUPT

When fxx is fx (CLKCON.2–.0 ≠ "101"), fw is fxt (WTCON.2 = "1"), and WTCON.3 = "1" (3.91 ms interval), watch timer's 3.91 ms interval interrupt is generated two times at the same time. That is, the interrupt is generated once more after executing the interrupt routine. (Suppose that other interrupts allow WT's 3.91 ms interrupts.)

☞ PROGRAMMING TIP – W/T's 3.91 ms interrupt when fxx = fx, fw = fxt, and WTCON.3 ="1"

```
WT3_91 ms:      TM      WT3_91 ms_f, #00000001B  :  WT's 3.91 ms interrupt routine
                JR      Z, INC_stopwatch         ;  Check flag
                AND     WT3.91 ms_f, #11111110B  ;  Clear flag
                NOP                              :  Delay 64-cycles over (@4 MHz)
                NOP
                NOP
                NOP
                NOP
                IRET
INC_stopwatch:  OR      WT3_91 ms_f, #00000001B  ; Set flag
                 •                               ; This routine is over 64-cycles (@4 MHz)
                 •
                 •
                IRET
```

18 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 an 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.

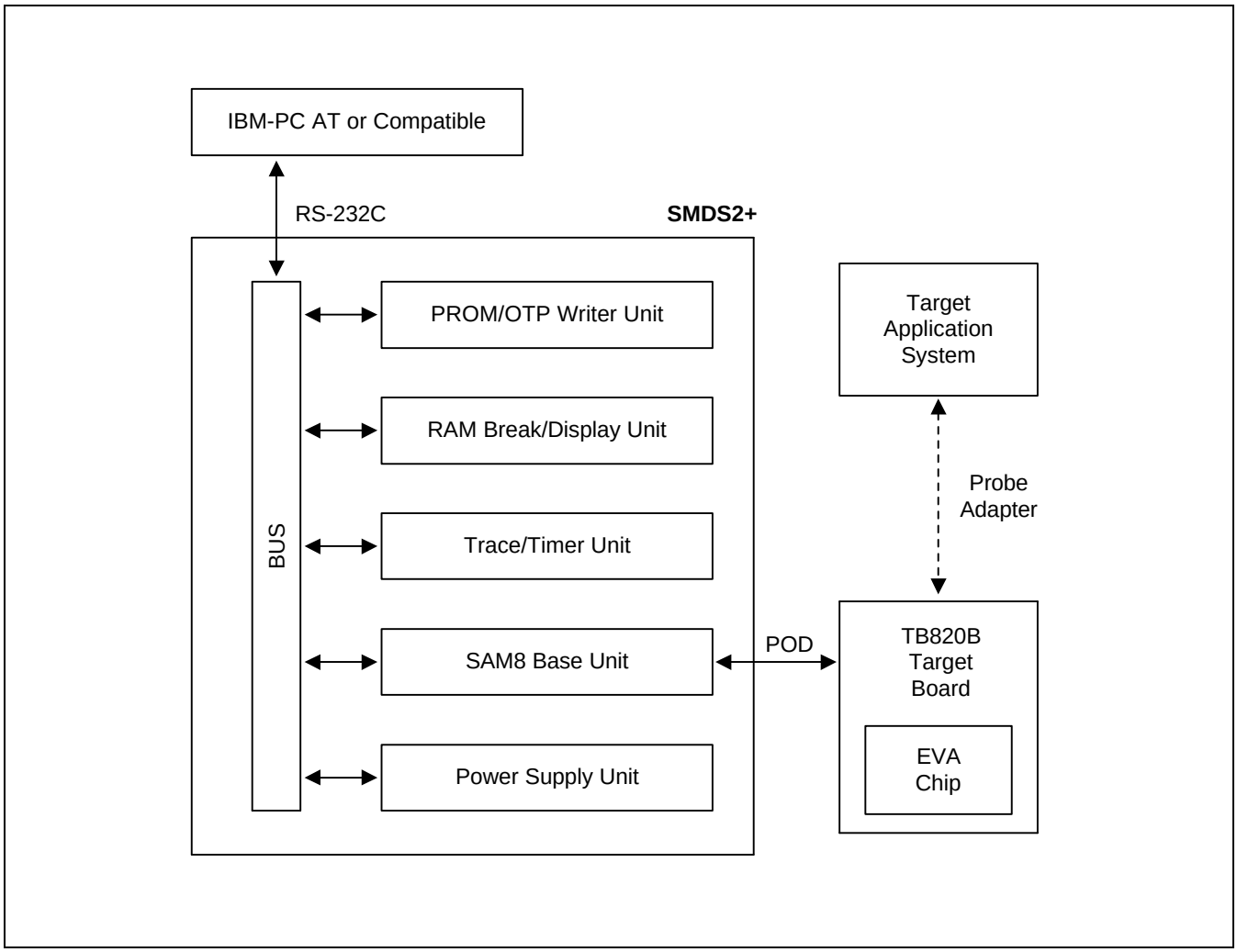


Figure 18-1. SMDS Product Configuration (SMDS2+)

TB820B TARGET BOARD

The TB820B target board is used for the S3C820B microcontroller. It is supported by the SMDS2+ development system.

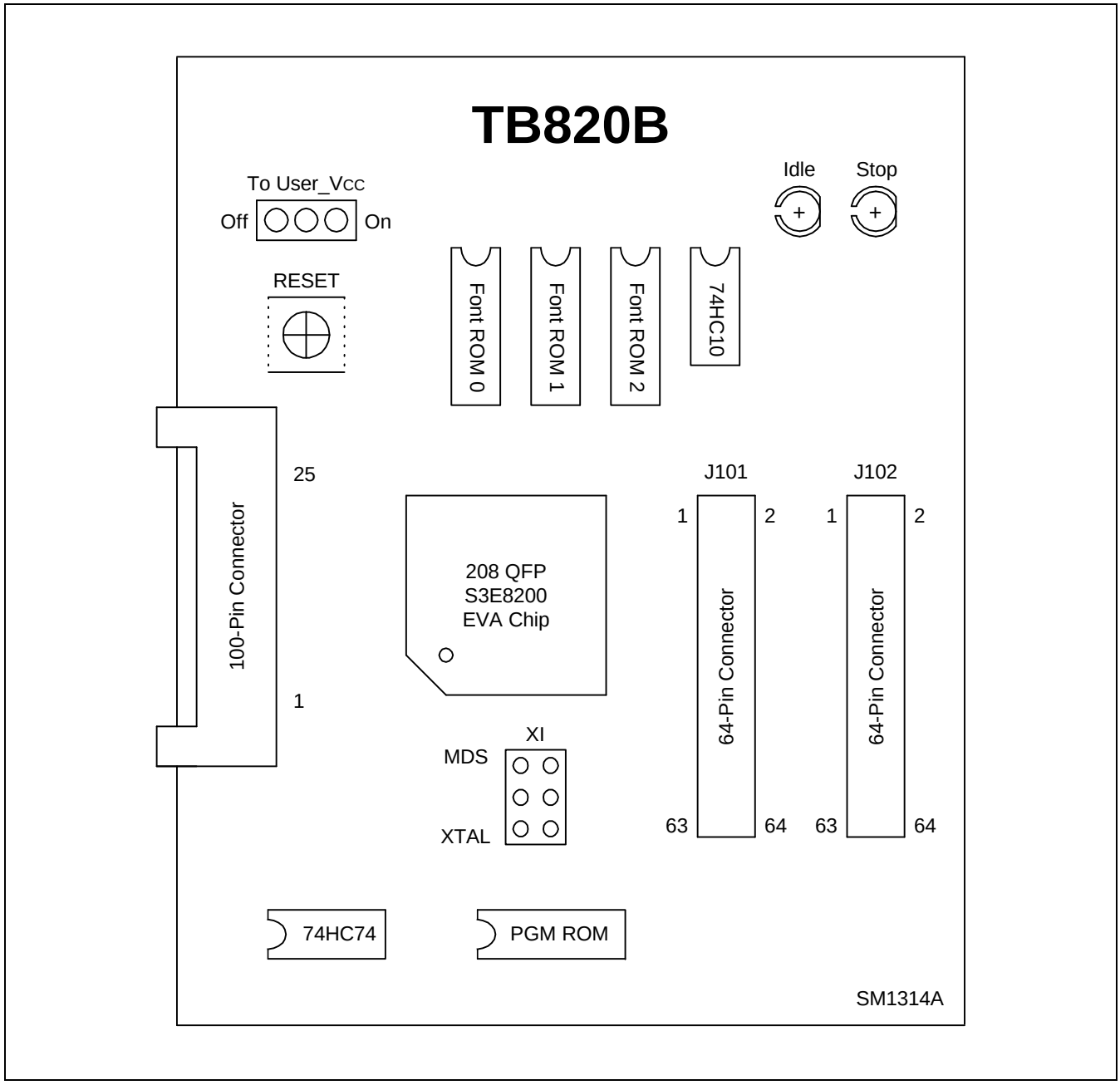
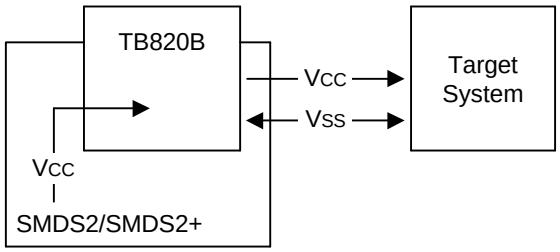
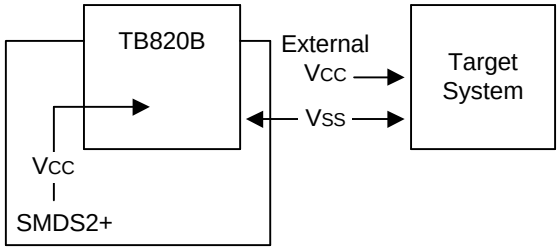


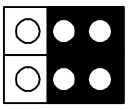
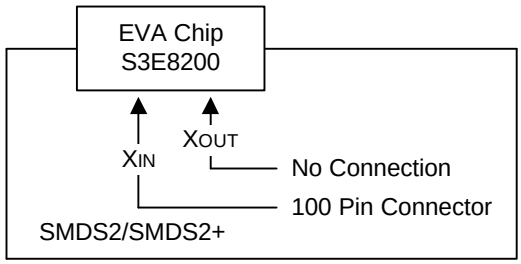
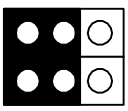
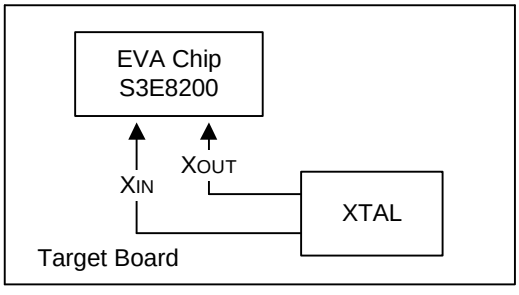
Figure 18-2. TB820B Target Board Configuration

Table 18-1. Power Selection Settings for TB820B

"To User_Vcc" Settings	Operating Mode	Comments
To User_Vcc Off  On		The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.
To User_Vcc Off  On		The SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

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

Table 18-2. Main-clock Selection Settings for TB820B

Sub Clock Setting	Operating Mode	Comments
XIN XTAL  MDS		Set the X_I switch to "MDS" when the target board is connected to the SMDS2/SMDS2+.
XIN XTAL  MDS		Set the X_I switch to "XTAL" when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+.

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 18-3. The SMDS2+ Tool Selection Setting

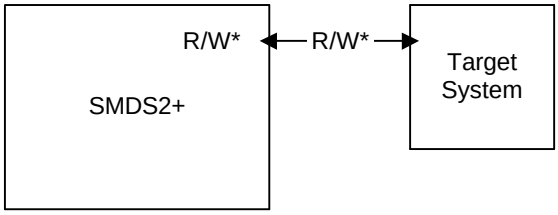
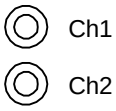
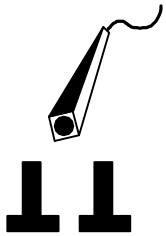
"SW1" Setting	Operating Mode
	

Table 18-4. Using Single Header Pins as the Input Path for External Trigger Sources

Target Board Part	Comments
External Triggers 	 Connector from External trigger sources of the application system You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for the SMDS2+ breakpoint and trace functions.

IDLE LED

The Green LED is ON when the evaluation chip (S3E8200) is in idle mode.

STOP LED

The Red LED is ON when the evaluation chip (S3E8200) is in stop mode.

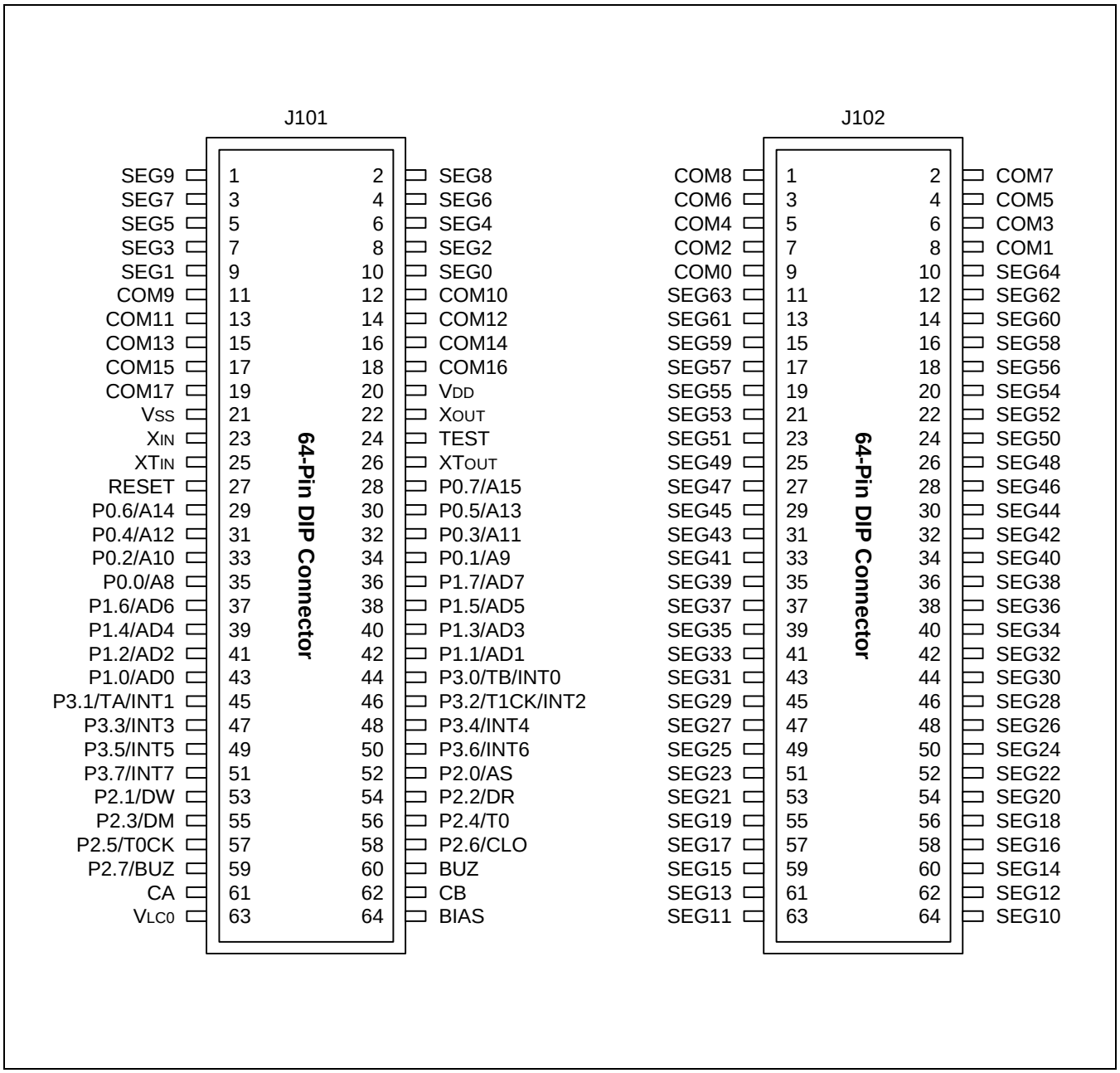


Figure 18-3. 64-Pin Connectors (J101, J102) for TB820B

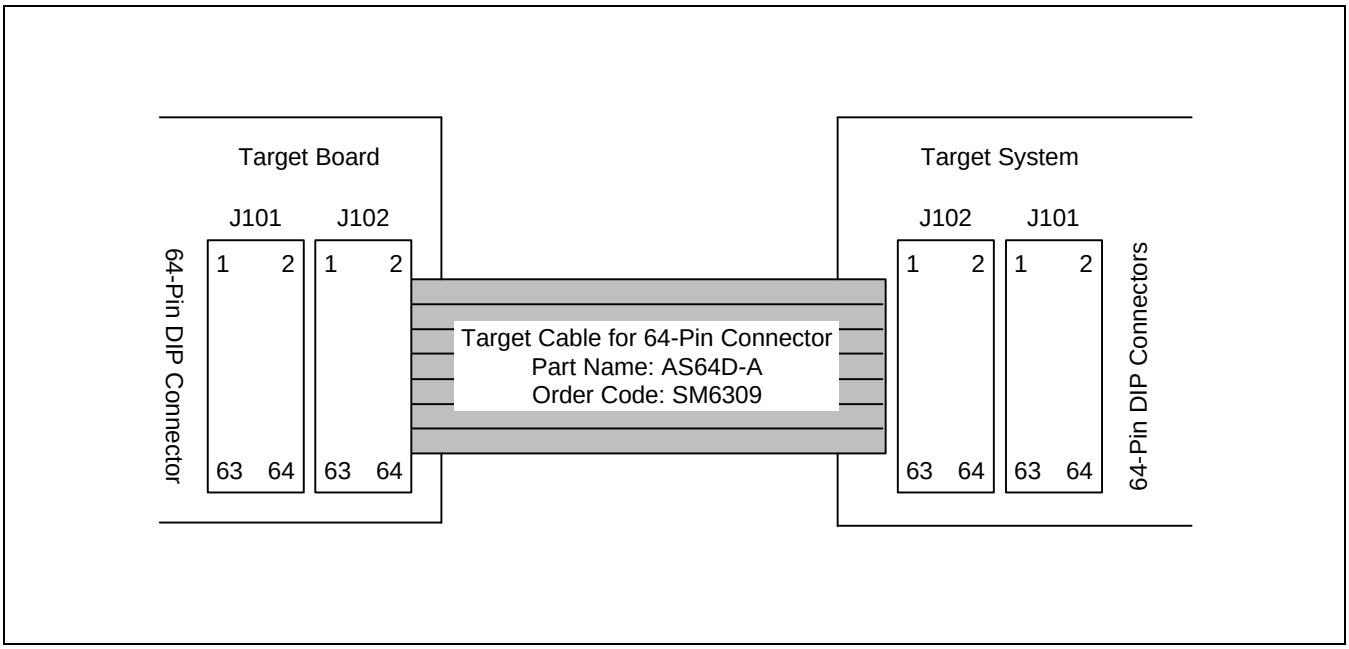


Figure 18-4. S3C820B Probe Adapter for 128-QFP Package

NOTES