1 PRODUCT OVERVIEW

SAM8 PRODUCT FAMILY

Samsung's SAM8 family 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.

S3C8618/C8615/P8615 MICROCONTROLLERS

The S3C8618/C8615/P8615 single-chip 8-bit microcontroller is based on the powerful SAM8 CPU architecture. The internal register file is logically expanded to increase the on-chip register space. The S3C8618/C8615/P8615 have 8/16 K bytes of on-chip program ROM.

Following Samsung's modular design approach, the following peripherals were integrated with the SAM8 core:

- Four programmable I/O ports (total 28 pins)
- One 8-bit basic timer for oscillation stabilization and watchdog functions
- One 8-bit general-purpose timer/counter with selectable clock sources
- One 8-bit counter with selectable clock sources, including Hsync or Csync input
- One 8-bit timer for interval mode
- PWM block with seven 8-bit PWM circuits
- Sync processor block (for Vsync and Hsync I/O, Csync input, and Clamp signal output)
- Multi master IIC-bus with DDC support.

The S3C8618/C8615/P8615 are a versatile microcontroller that is ideal for use in multi-sync monitors or in general-purpose applications that require sophisticated timer/counter, PWM, sync signal processing, and multi-master IIC-bus support with DDC. It is available in a 42-pin SDIP or a 44-pin QFP package.

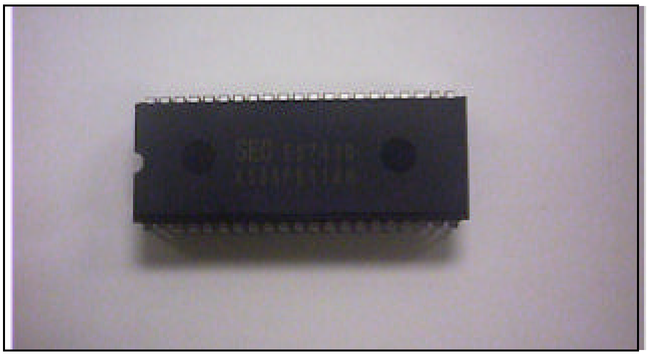


Figure 1-1. S3C8618/C8615/P8615 Microcontrollers

FEATURES

CPU

- SAM8 CPU core

Memory

- 8/16-Kbyte internal program memory (ROM)
- 272-byte general-purpose register area

Instruction Set

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

Instruction Execution Time

- 500 ns minimum (with 12 MHz CPU clock)

Interrupts

- Nine interrupt sources
- Nine interrupt vectors
- Six interrupt levels
- Fast interrupt processing for a select level

General I/O

- Four I/O ports (total 28 pins):

8-Bit Basic Timer

- Programmable timer for oscillation stabilization interval control or watchdog timer functions
- Three selectable internal clock frequencies

Timer/Counters

- One 8-bit general-purpose timer/counter with programmable operating modes and the following clock source options:
  - Two selectable internal clock frequencies
- One 8-bit timer with interval operating mode
- One 8-bit counter with the following clock source options:
  - Two selectable internal clock frequencies
  - Hsync (or Csync) input from the sync processor block
  - External clock source

Pulse Width Modulator

- Seven 8-bit PWM modules:
  - 8-bit basic frame
  - Four push-pull and three n-channel, open-drain output channels
  - Selectable clock frequencies: 46.875 kHz at 12 MHz fosc.

Sync Processor

- Detection of sync input signals (Vsync-I, Hsync-I, and Csync-I)
- Sync signal separation and output (Hsync-O, Vsync-O, and Clamp-O)
- Pseudo sync signal output
- Programmable clamp signal output

DDC and Multi-Master IIC-Bus

- Serial peripheral interface
- Support for display data channel (DDC)

Oscillator Frequency

- 6 MHz to 12 MHz external crystal oscillator
- Interval Max. 12MHz CPU clock

Operating Temperature Range

- $-40^{\circ}C$ to $+85^{\circ}C$

Operating Voltage Range

- 4.5 V to 5.5 V

Package Types

- 42-pin SDIP, 44-pin QFP

BLOCK DIAGRAM

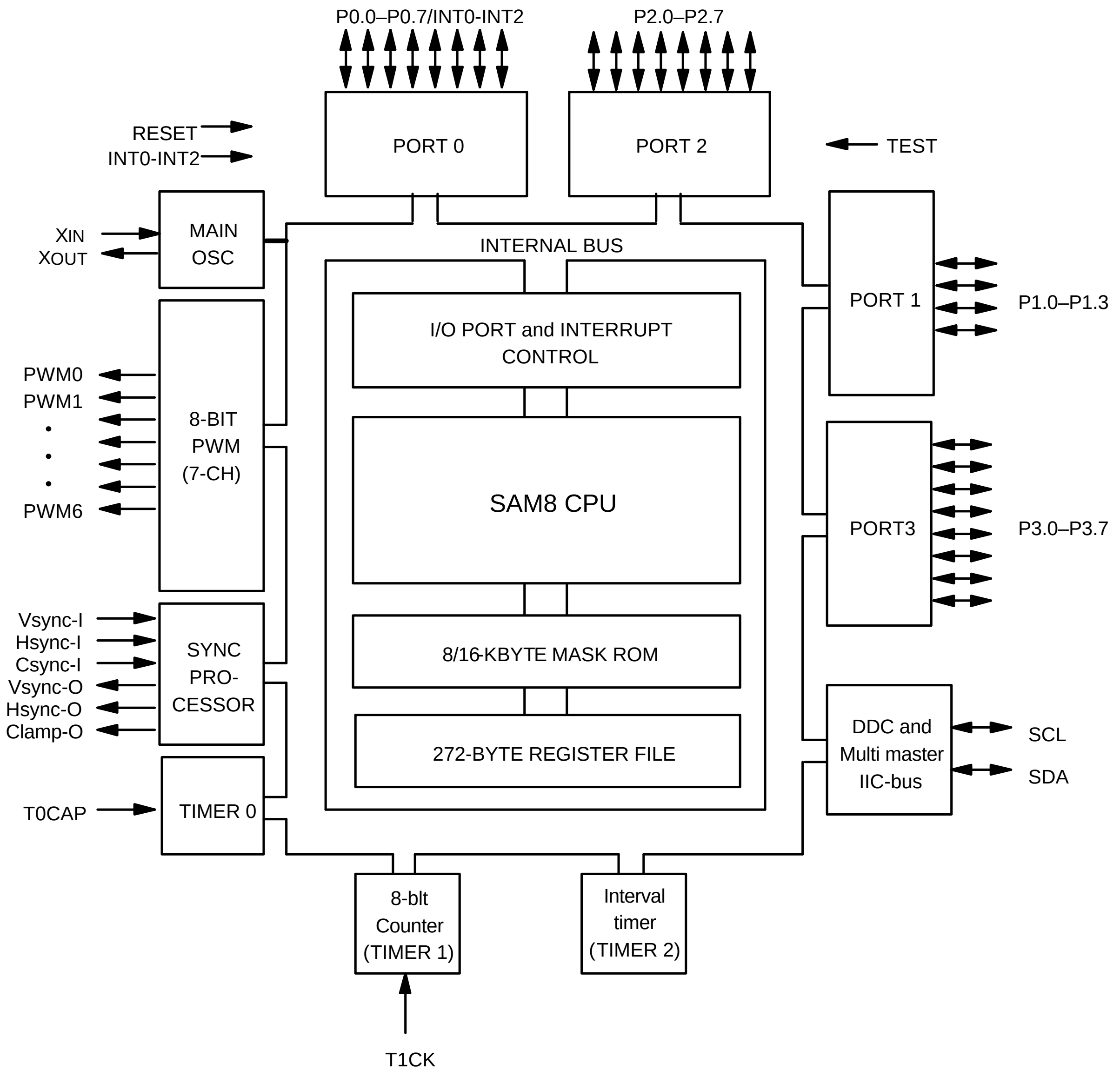


Figure 1-2. Block Diagram

PIN ASSIGNMENTS

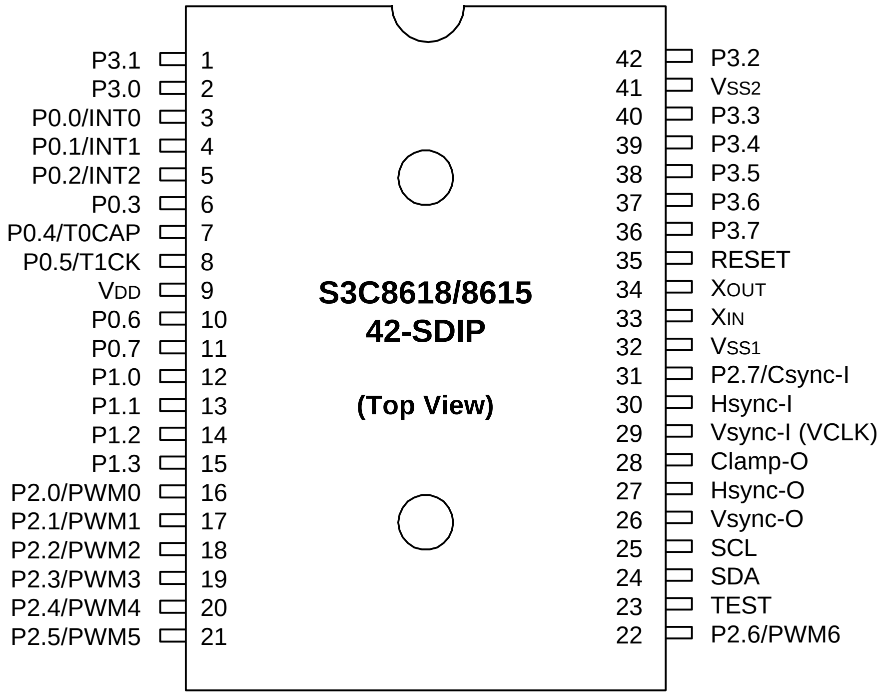


Figure 1-3. Pin Assignment Diagram (42-SDIP Package)

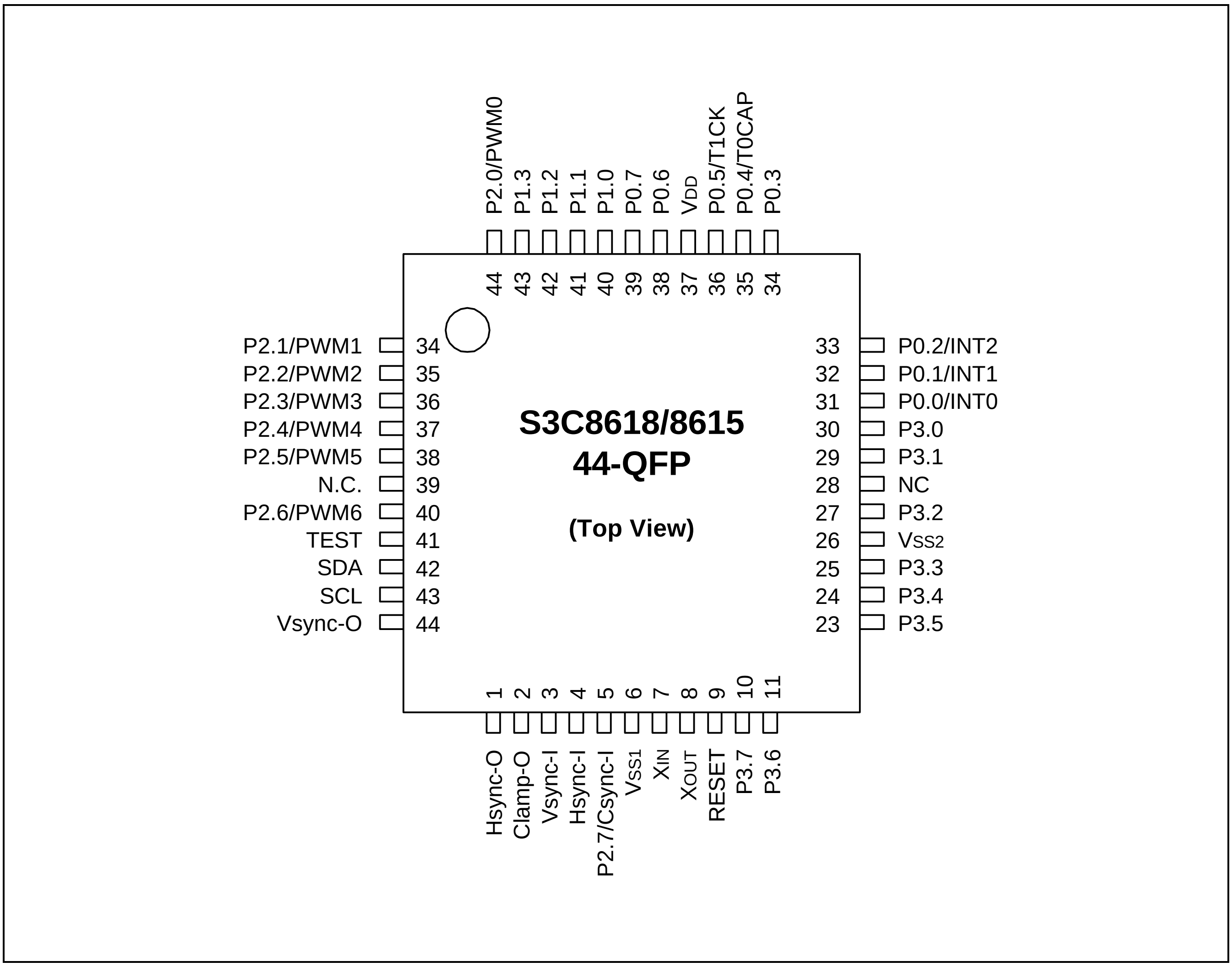


Figure 1-4. Pin Assignment Diagram (44-QFP Package)

PIN DESCRIPTIONS

Table 1-1. S3C8618/C8615/P8615 Pin Descriptions

Pin Names	Pin Type	Pin Description	Circuit Type	SDIP Pin Numbers	Shared Functions
P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	I/O	General-purpose, 8-bit I/O port. Share functions include three external interrupt inputs, I/O for timers 0 and 1. You can selectively configure port 0 pins to input or output mode.	D-1	3 4 5 6 7 8 10 11	INT0 INT1 INT2 T0CAP T1CK
P1.0–P1.3	I/O	General purpose, 8-bit I/O port. You can selectively configure port 1 pins to input or push-pull output mode.	D-1	12–15	–
P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7	I/O	General purpose, 8-bit I/O port. You can selectively configure port 2 pins to input or output mode. The port 2 pin circuit are designed to push-pull PWM output and Csync signal input.	D-1 D-1 D-1 D-1 E-1 E-1 E-1 D-1	16 17 18 19 20 21 22 31	PWM0 PWM1 PWM2 PWM3 PWM4 PWM5 PWM6 Csync-I
P3.0–P3.7	I/O	General-purpose, 8-bit I/O port. You can selectively configure port 3 pins to input or output mode.	E	2, 1, 42, 40–36	–
Hsync-I Vsync-I Clamp-O Hsync-O Vsync-O SCL SDA	I I O O O I/O I/O	The pins are sync processor signal I/O and IIC-bus clock and data I/O	A A A A A G-3 G-3	30 29 28 27 26 25 24	–
V_{DD} V_{SS1}, V_{SS2}	–	Power supply pins	–	9 32, 41	–
X_{IN}, X_{OUT}	–	System clock input and output pins	–	33, 34	–
RESET	I	System reset pin	B	35	–
TEST	I	Factory test pin input 0 V: normal operation 5 V: factory test mode	–	23	–

NOTE: See 'Pin Circuit Diagrams' on next two pages for detailed information on circuit types A, B, D-1, E, E-1,and G-3.

PIN CIRCUITS

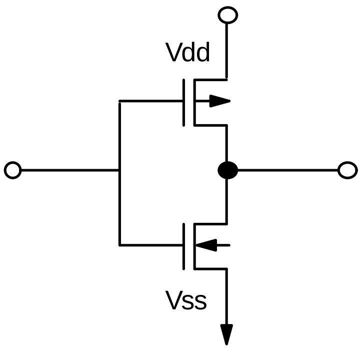


Figure 1-5. Pin Circuit Type A

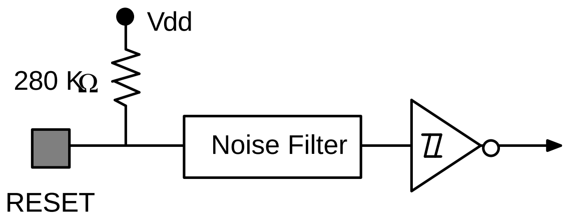


Figure 1-6. Pin Circuit Type B (RESET)

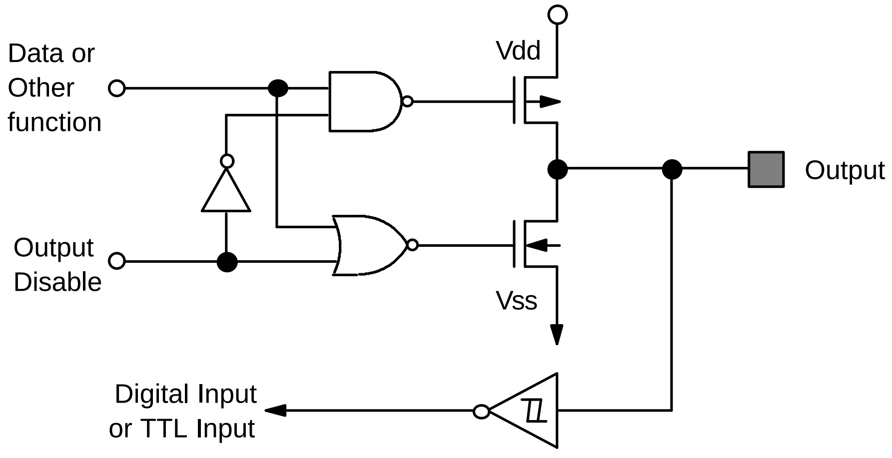


Figure 1-7. Pin Circuit Type D-1

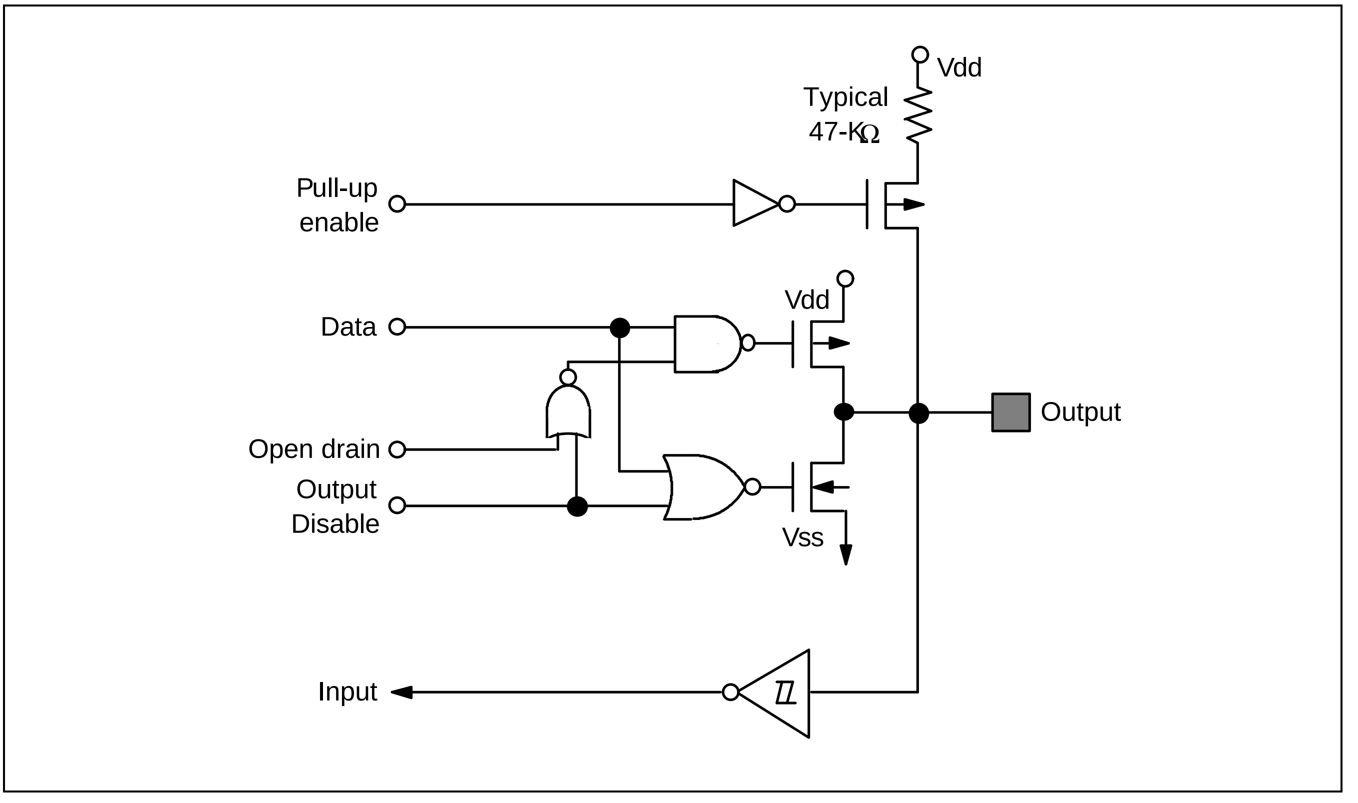


Figure 1-7. Pin Circuit Type E

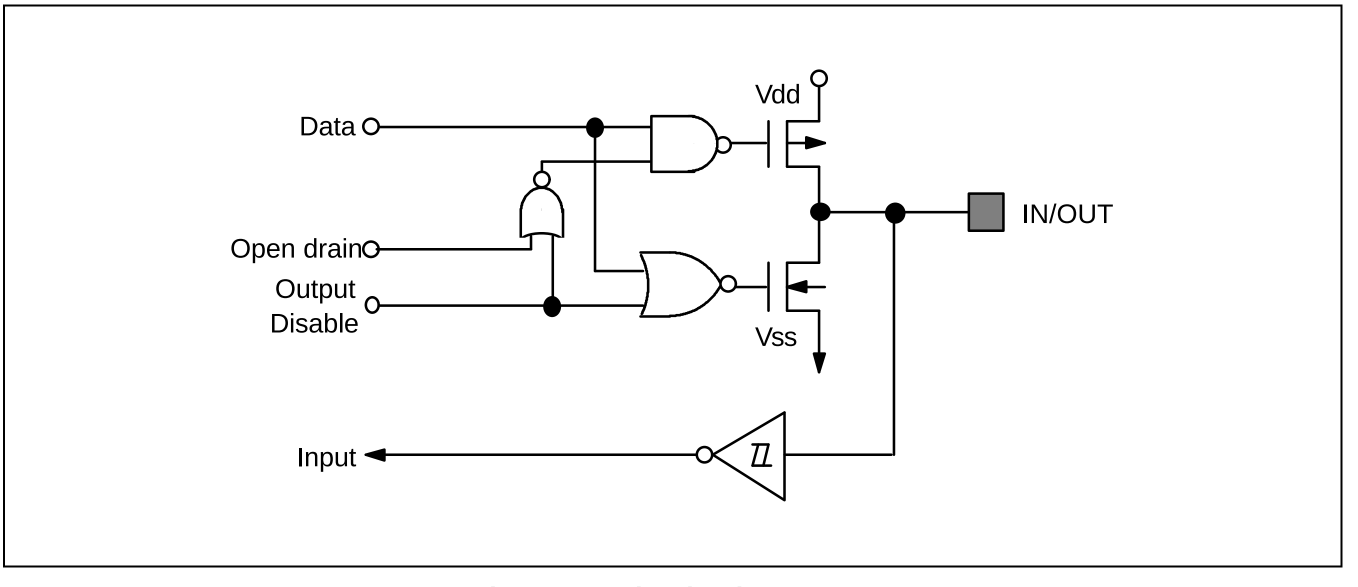


Figure 1-8. Pin Circuit Type E-1

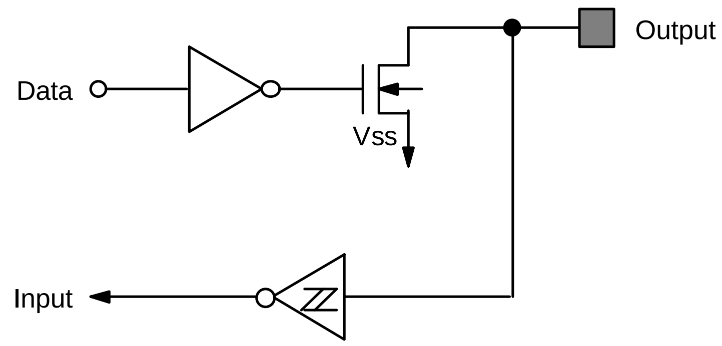


Figure 1-9. Pin Circuit Type G-3

2 ADDRESS SPACES

OVERVIEW

The S3C8618/C8615 microcontrollers have two types of address space:

— Internal program memory (ROM)

— Internal register file

A 16-bit address and data bus supports program memory operations. A separate 8-bit register bus carries addresses and data between the CPU and the internal register file. The S3C8618/C8615 have an internal 8/16-Kbyte mask-programmable ROM. An external memory interface is not implemented.

There are 325 8-bit registers in the internal register file. In this space, there are 272 registers for general-purpose use, 19 for CPU and system control, and 34 for peripheral control and data. A 16-byte common working register (scratch) area is part of the general-purpose register space. Most of these registers can serve as either a source or destination address, or as accumulators for data memory operations.

PROGRAM MEMORY (ROM)

Program memory (ROM) stores program code or table data. The S3C8618/C8615 have 8/16 K bytes of mask-programmable program memory. The memory address range is therefore 0H–3FFFH (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.

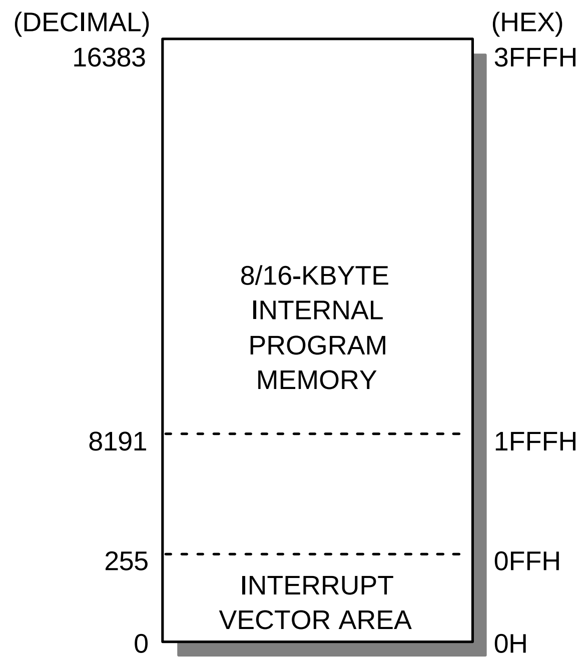


Figure 2-1. Program Memory Address Space

REGISTER ARCHITECTURE

The upper 64-byte area of the S3C8618/C8615 internal register files is logically expanded into two 64-byte areas, called *set 1* and *set 2*. The upper 32-byte area of set 1 is divided into two register banks, *bank 0* and *bank 1*. The total physical register space is thereby expanded to 352 bytes. Within this physical space, there are 352-byte registers, of which 325 are addressable.

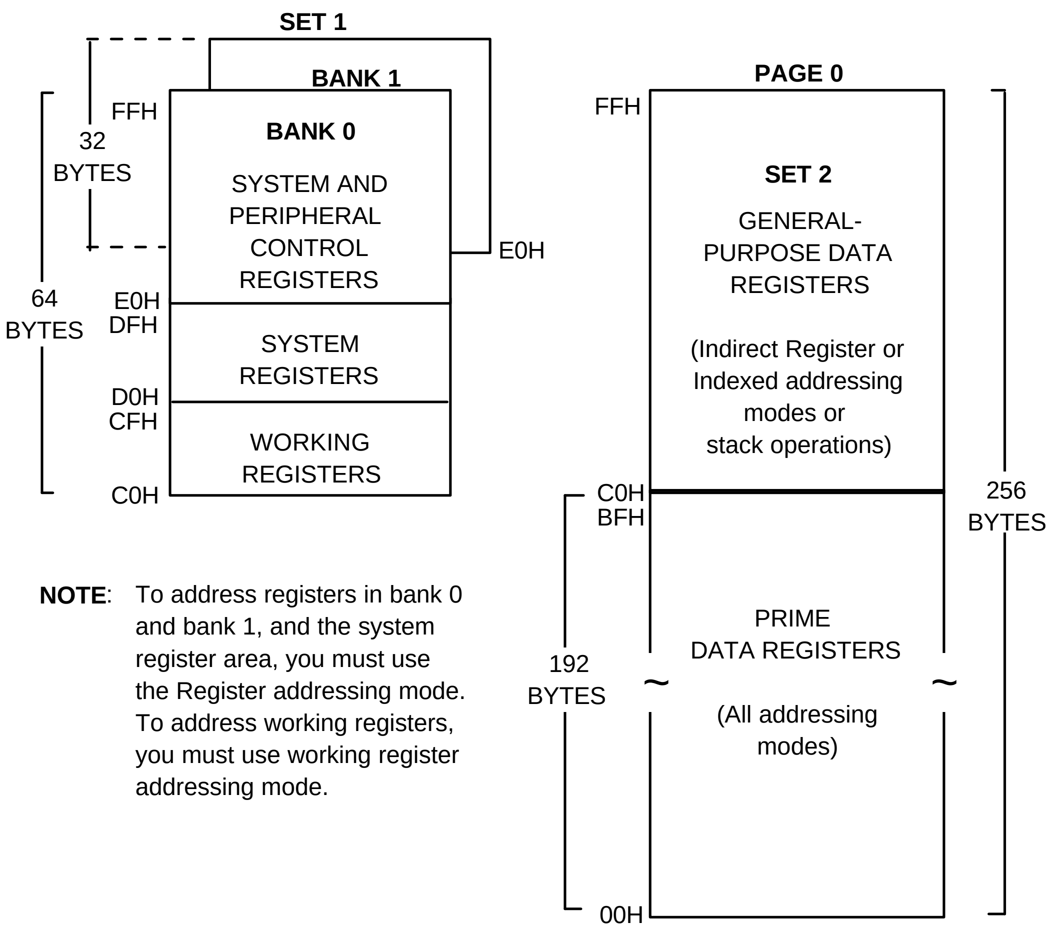
Given the microcontroller's 8-bit register bus architecture, up to 256 bytes of physical register space can be addressed as a single page. The S3C8618/C8615 register files have one page only, *page 0*. Page 0 contains 256 bytes.

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

Specific register types and the area (in bytes) they occupy in the S3C8618/C8615 internal register files are summarized in Table 2-1.

Table 2-1. Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16-byte common working register area)	272
CPU and system control registers	19
Clock, peripheral, and I/O control and data registers	34
Total Addressable Bytes	325



NOTE: To address registers in bank 0 and bank 1, and the system register area, you must use the Register addressing mode. To address working registers, you must use working register addressing mode.

Figure 2-2. Internal Register File Organization

REGISTER PAGE POINTER (PP)

The SAM8 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). This paged register file expansion is not implemented in the S3C8618/C8615. For this reason, the S3C8618/C8615 page pointer always points to page 0 (see Figure 2-3).

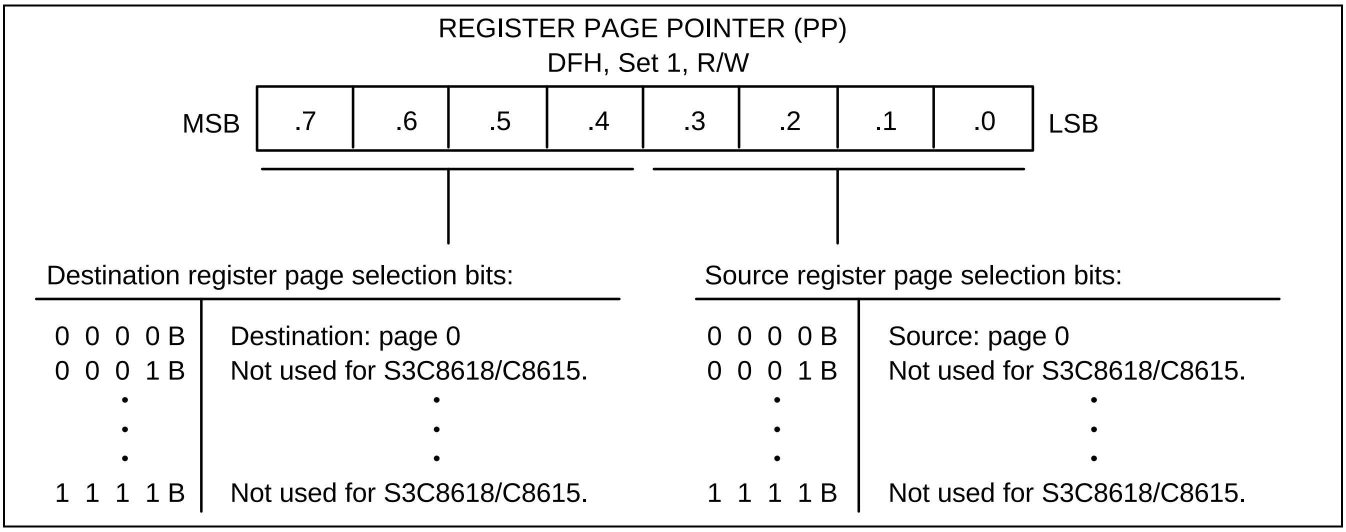


Figure 2-3. Register Page Pointer (PP)

REGISTER SET 1

The term *set 1* refers to the upper 64 bytes of the register file, locations C0H–FFH. The upper 32-byte area of this 64-byte space (E0H–FFH) is divided into two 32-byte register banks, *bank 0* and *bank 1*. You execute the set register bank instructions SB0 or SB1 to address one bank or the other. Bank 0 addressing is automatically selected by a reset operation.

For the S3C8618/C8615, only register locations E0H–ECH are addressable in the bank 1 area; the remaining locations (EDH–FFH) are not mapped. The lower 32-byte area of set 1 is not banked and can be addressed at any time. It contains 16 mapped system registers (D0H–DFH) and a 16-byte "scratch" area (C0H–CFH) for working register addressing.

Registers in set 1 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 Section 3, "Addressing Modes" .)

REGISTER SET 2

The same 64-byte physical space that is used for set 1 register locations C0H–FFH is logically duplicated to add another 64 bytes of space. This expanded area of the register file is called *set 2*. All set 2 locations (C0H–FFH) can be addressed within page 0 of the S3C8618/C8615 register space.

The logical division of set 1 and set 2 is maintained by means of addressing mode restrictions: While you can use only the Register addressing mode to access set 1 locations, you can only use Register Indirect addressing mode or Indexed addressing mode to access register locations in set 2.

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 range as page 0, and using any of the seven explicit addressing modes (see Section 3, “Addressing Modes”). All registers in the prime area are addressable immediately following a reset.

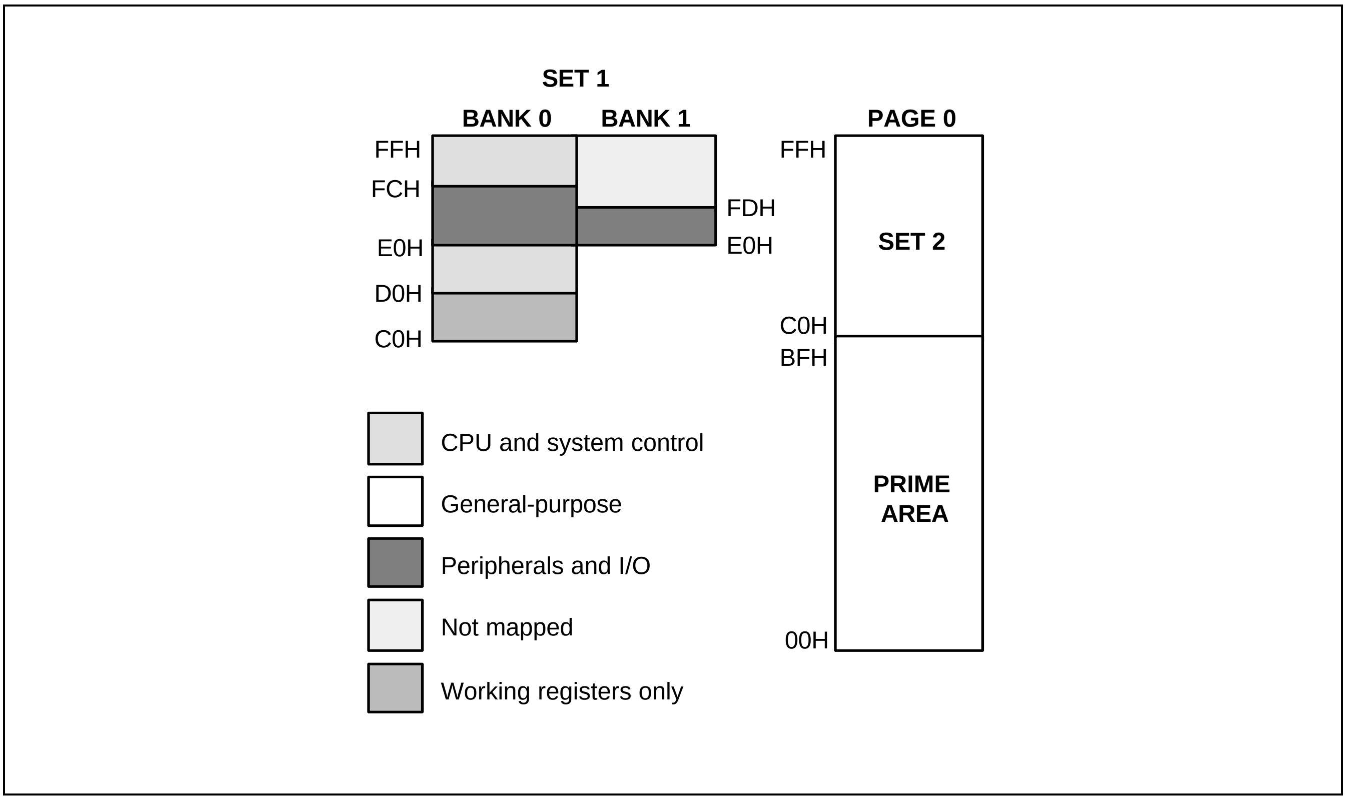


Figure 2-4. Set 1, Set 2, and Prime Area Register Map

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 viewed 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).

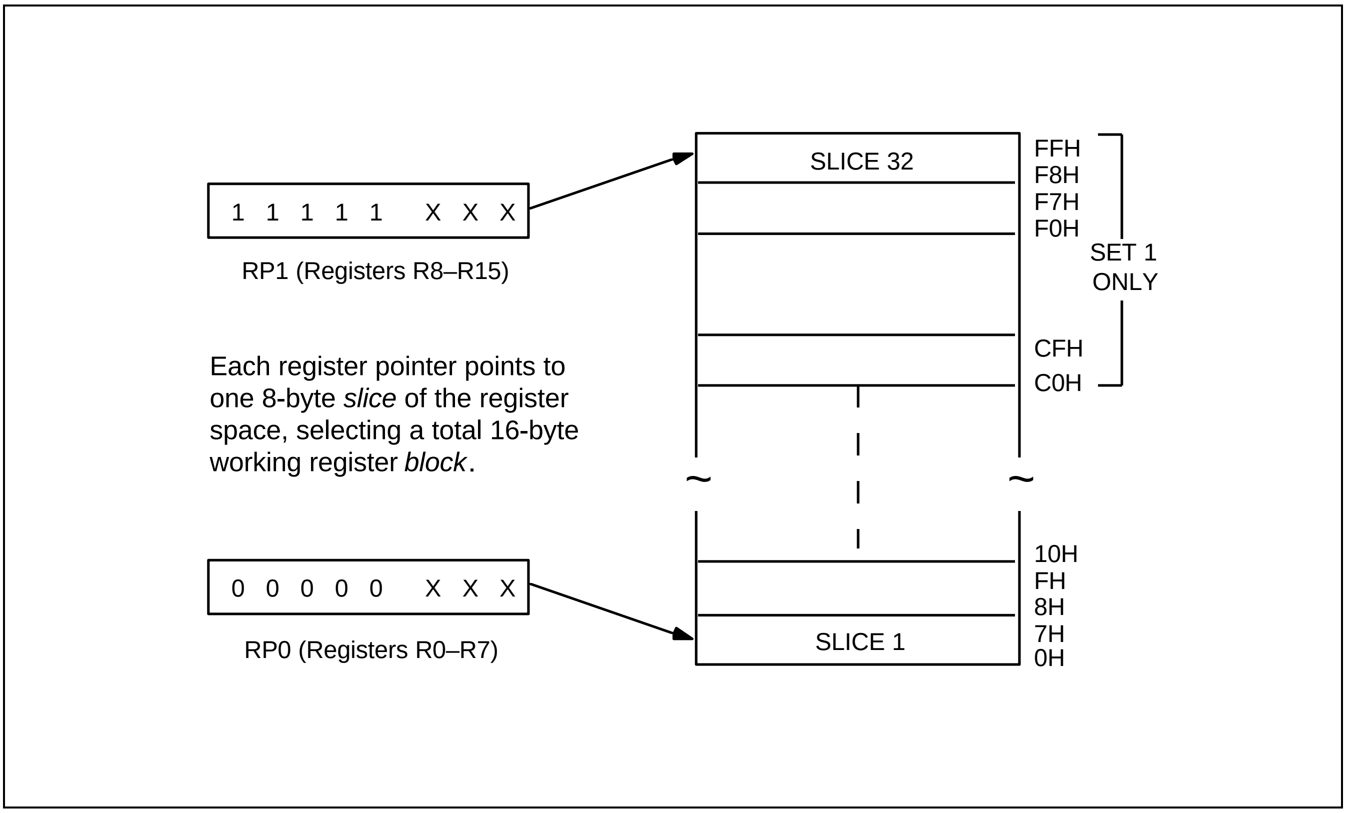


Figure 2-5. 8-Byte Working Register Areas (Slices)

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-bit slices of the register file that are currently pointed to by RP0 and RP1. You cannot, however, use the register pointers to select a working register area 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
```

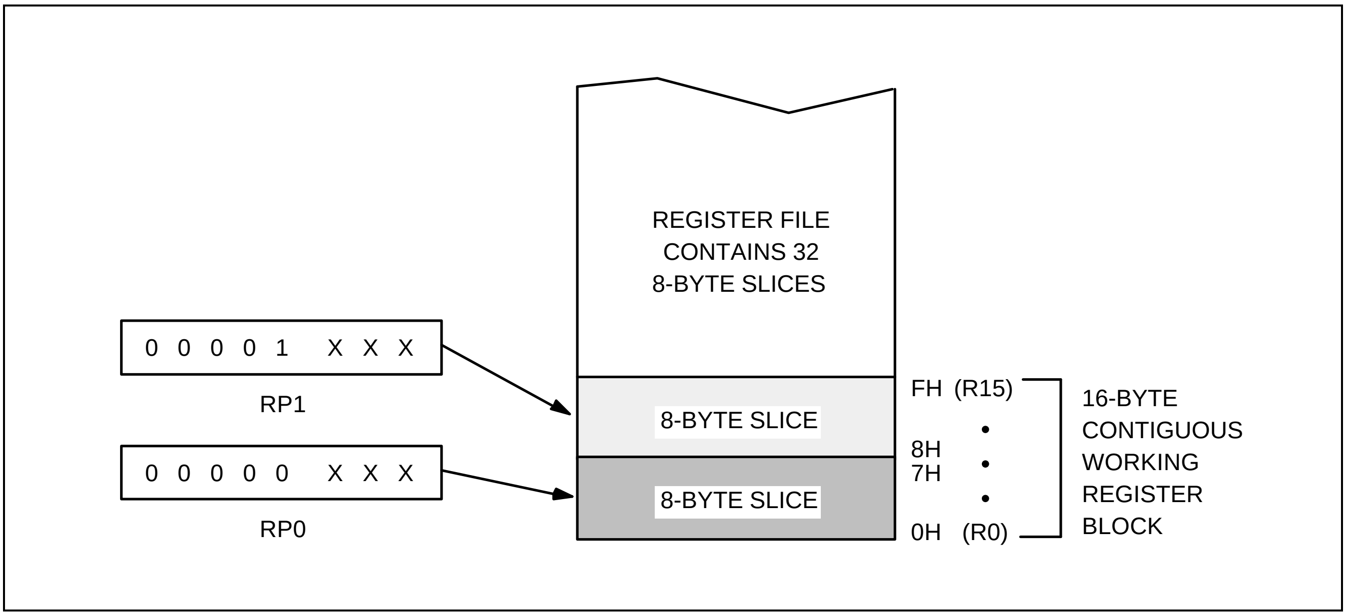


Figure 2-6. Contiguous 16-Byte Working Register Block

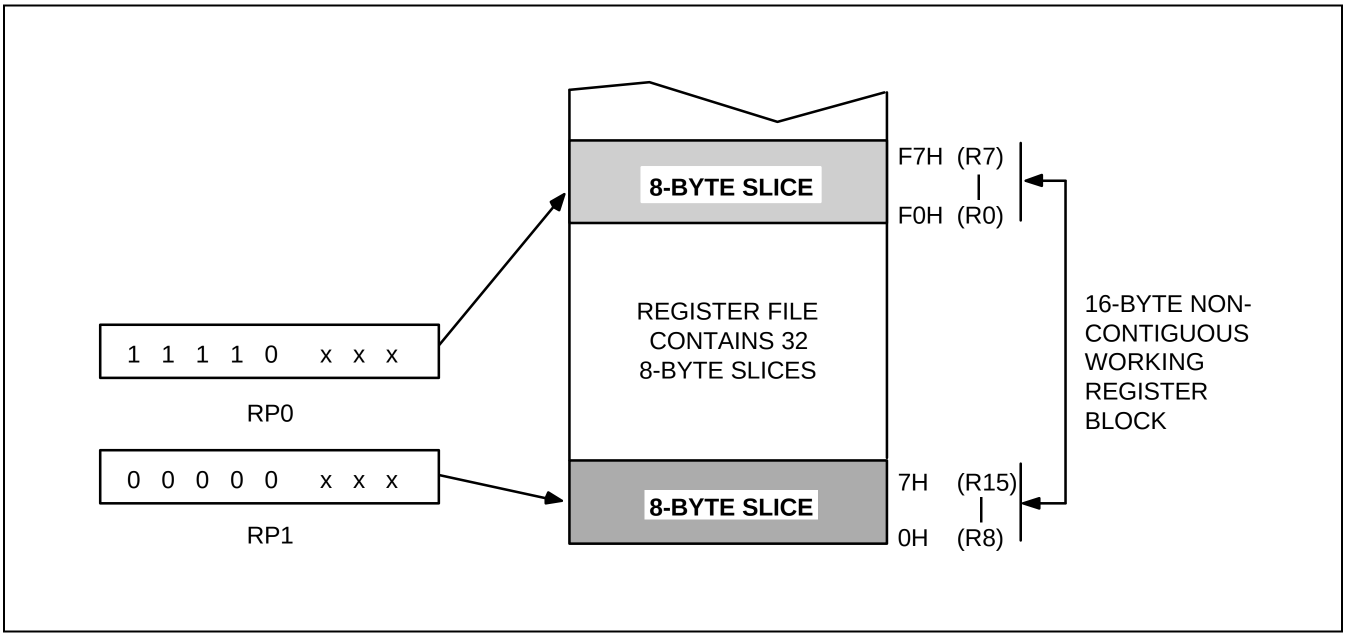


Figure 2-7. Non-Contiguous 16-Byte Working Register Block

☞ 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 and working register addressing. The register addresses 80H through 85H contain 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 SAM8 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 specify an 8-byte working register space in the register file and an 8-bit register within that space (see Figure 3-2).

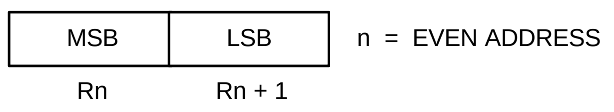


Figure 2-8. 16-Bit Register Pair

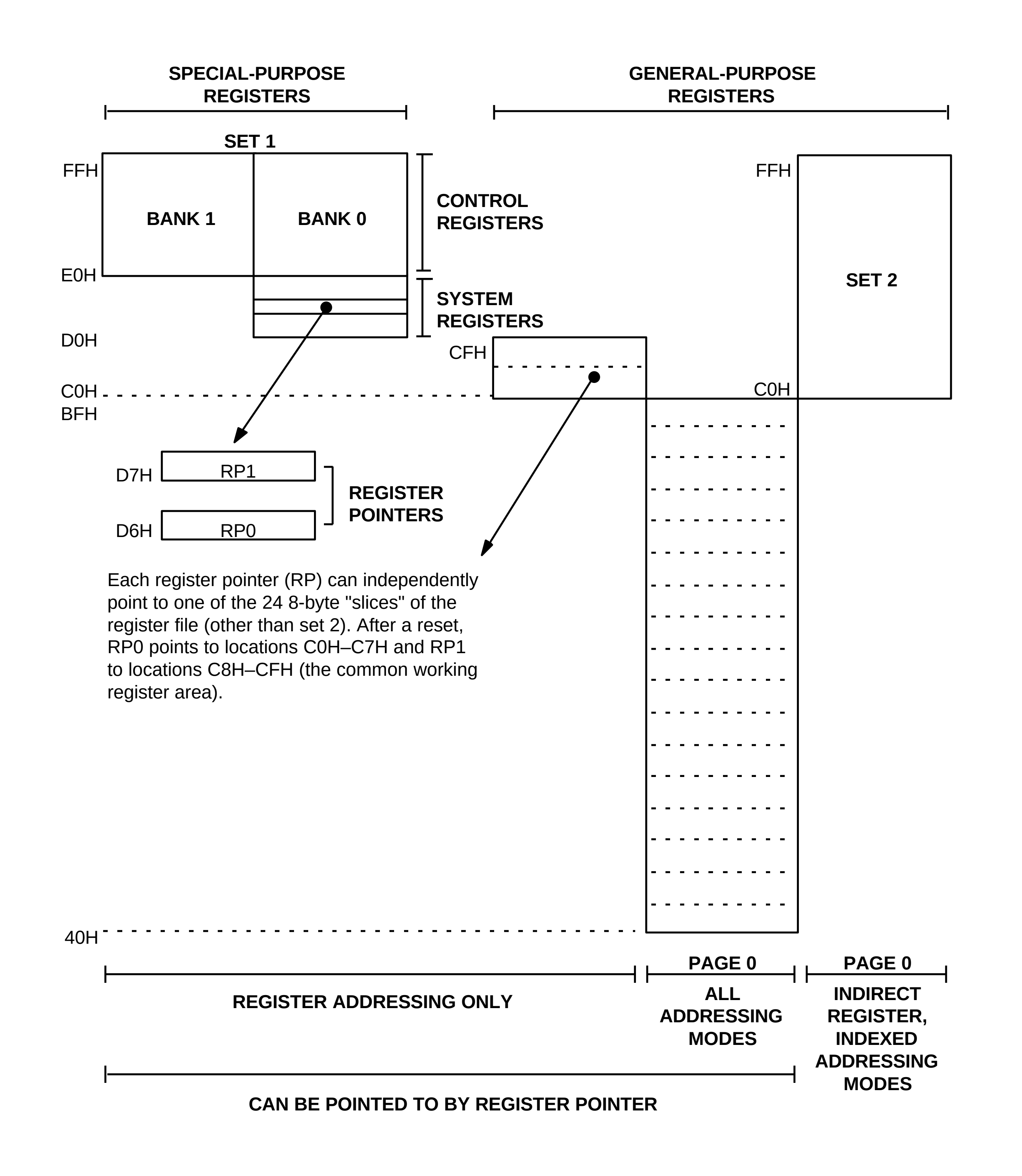


Figure 2-9. Register File Addressing

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

This16-byte address range is called *common working register area*. That is, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages. However, because the S3C8618/C8615 uses only page 0, you can use the common working register area for any internal data operation.

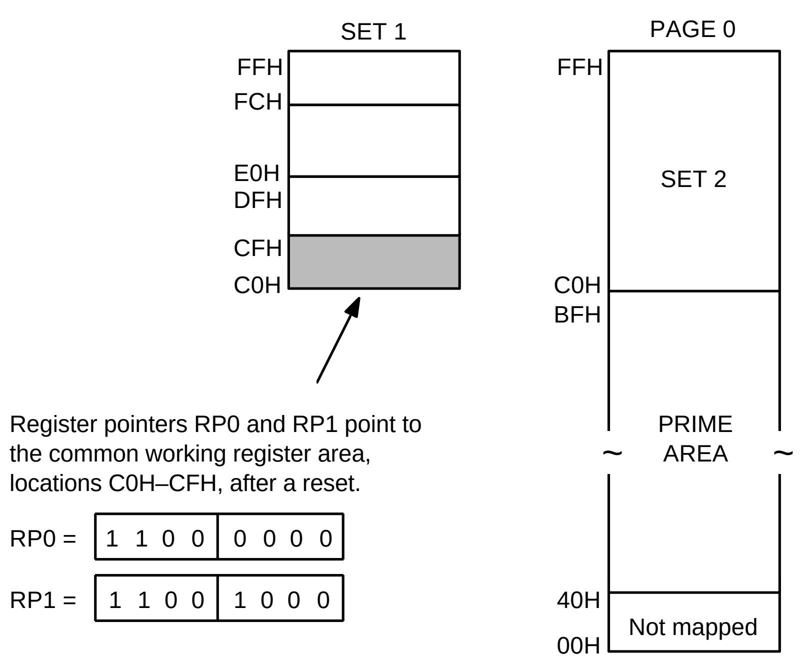


Figure 2-10. Common Working Register Area

☞ 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).

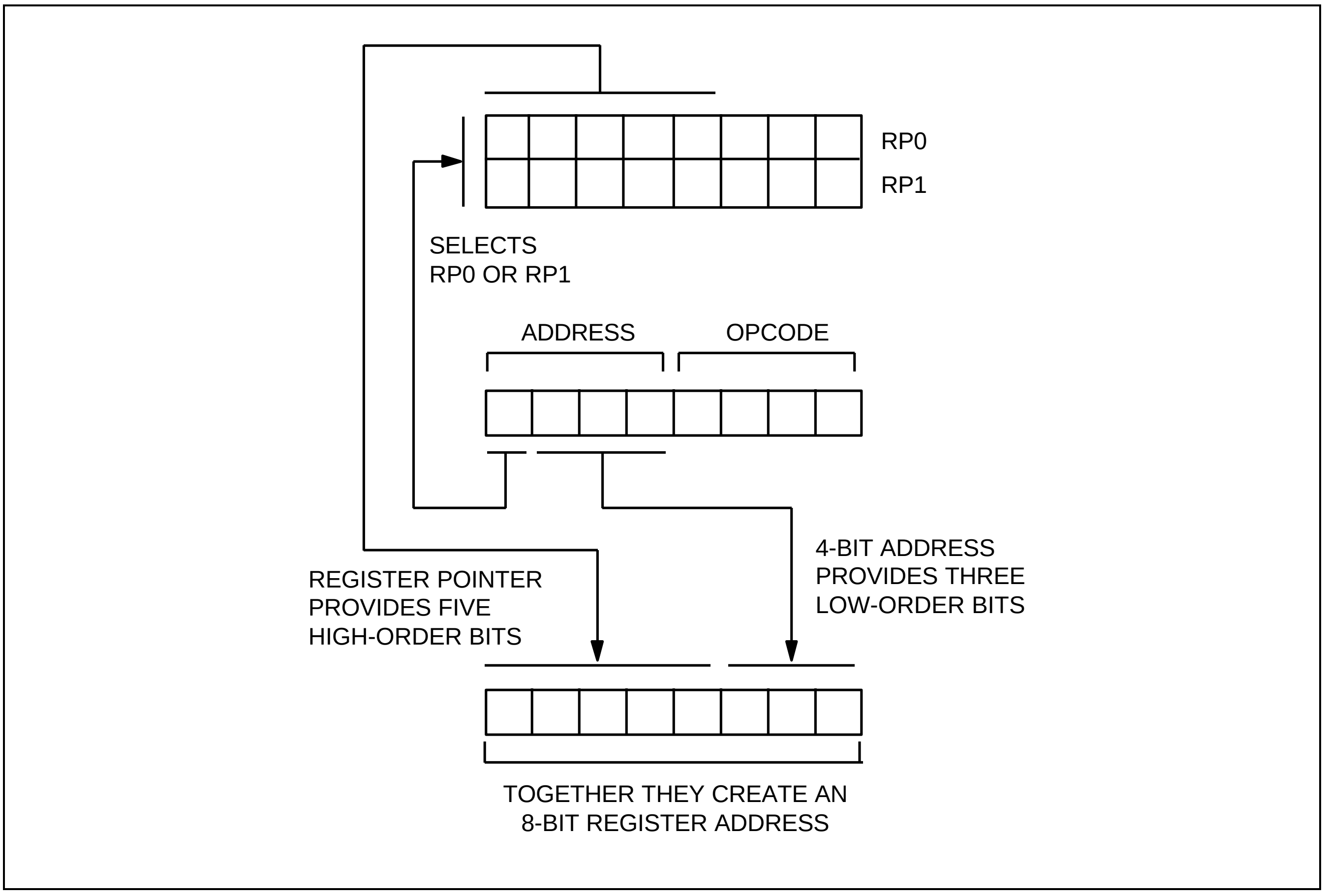


Figure 2-11. 4-Bit Working Register Addressing

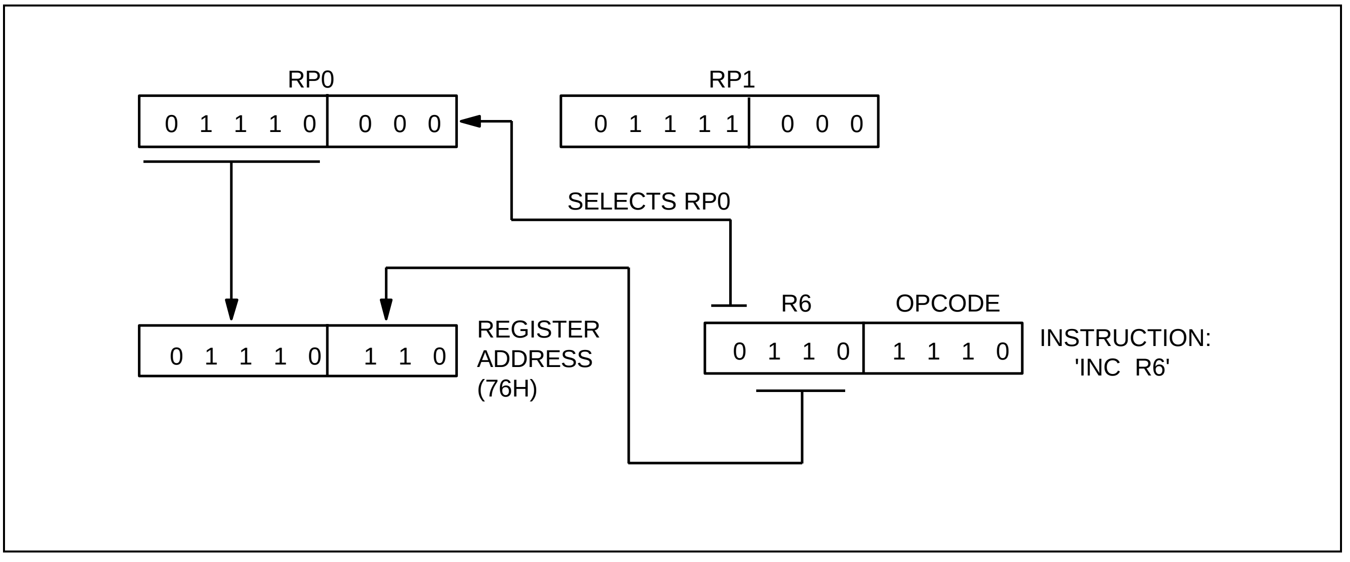


Figure 2-12. 4-Bit Working Register Addressing Example

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).

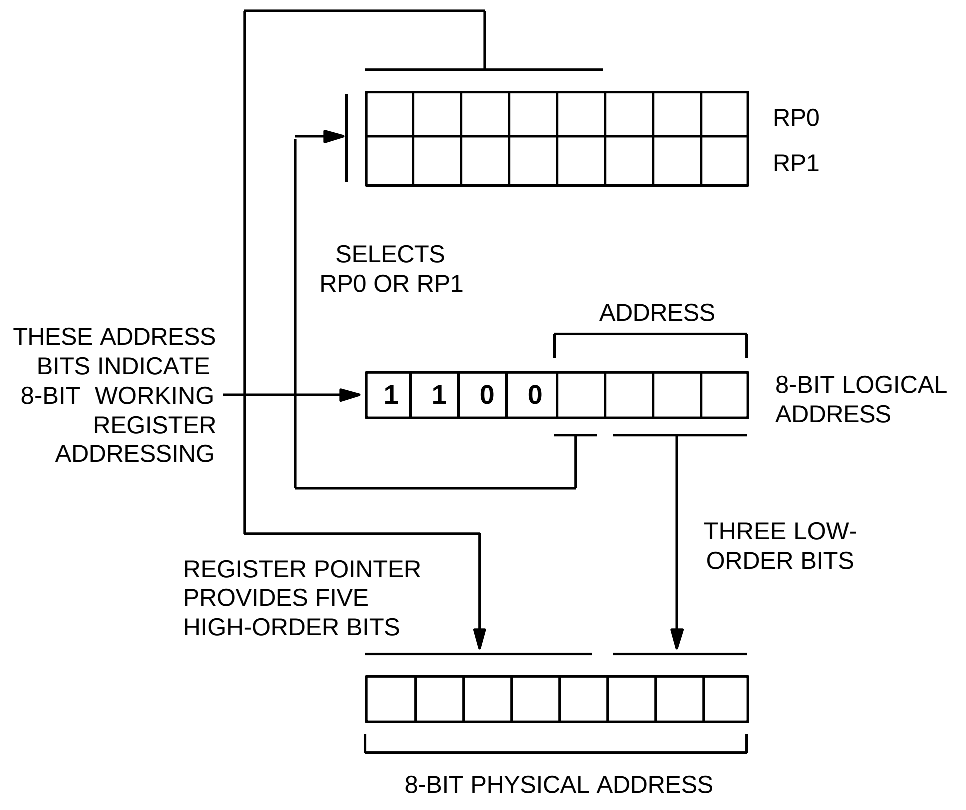


Figure 2-13. 8-Bit Working Register Addressing

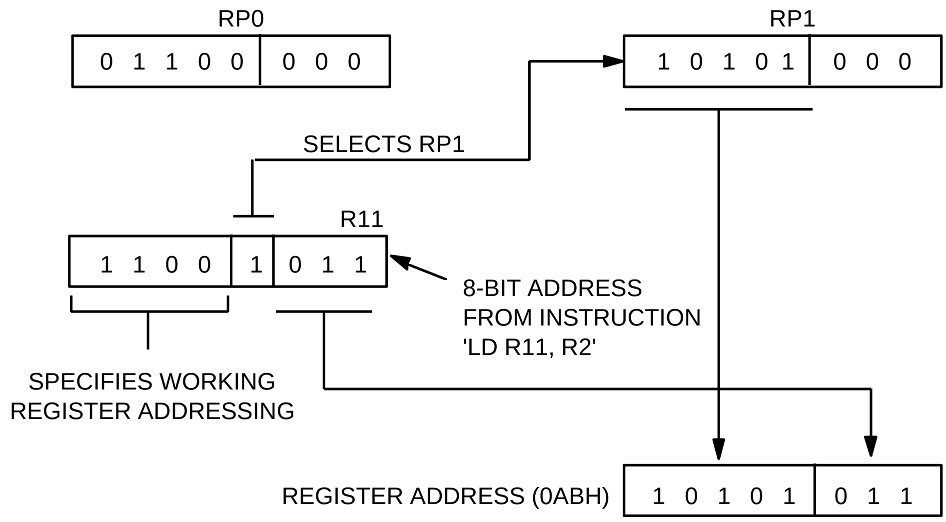


Figure 2-14. 8-Bit Working Register Addressing Example

SYSTEM AND USER STACKS

S3C8-series microcontrollers can be programmed to use the system stack for subroutine calls, returns and interrupts and to store data. The PUSH and POP instructions are used to control system stack operations. The S3C8618/C8615 architecture supports stack operations in the internal register file.

Stack Operations

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

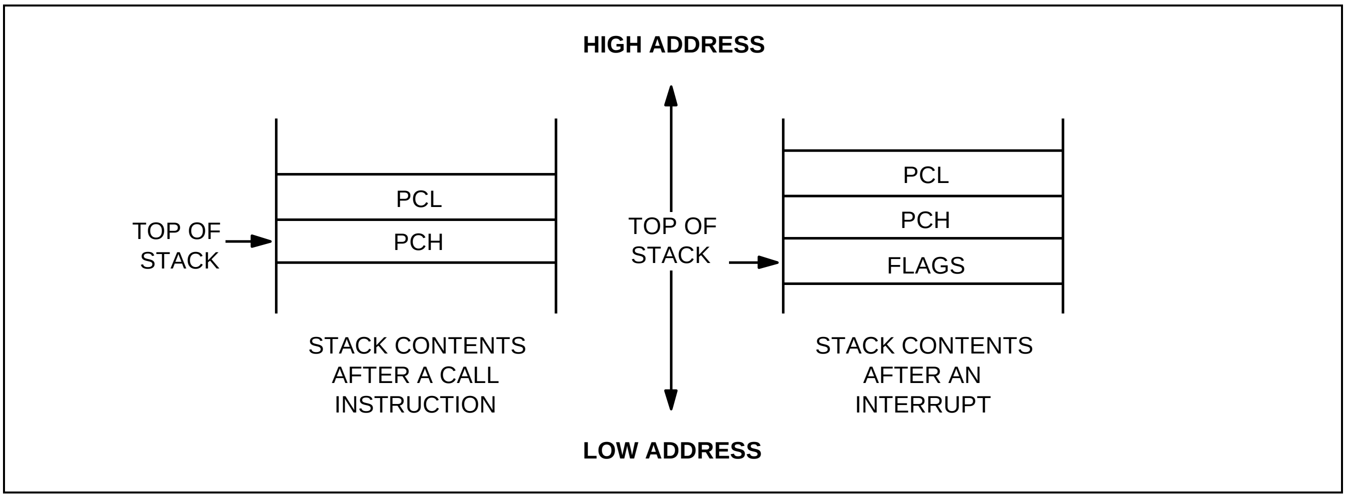


Figure 2-15. Stack Operations

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 S3C8618/C8615, 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 incrementing or decrementing the stack address 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 SAM8 instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The SAM8 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).

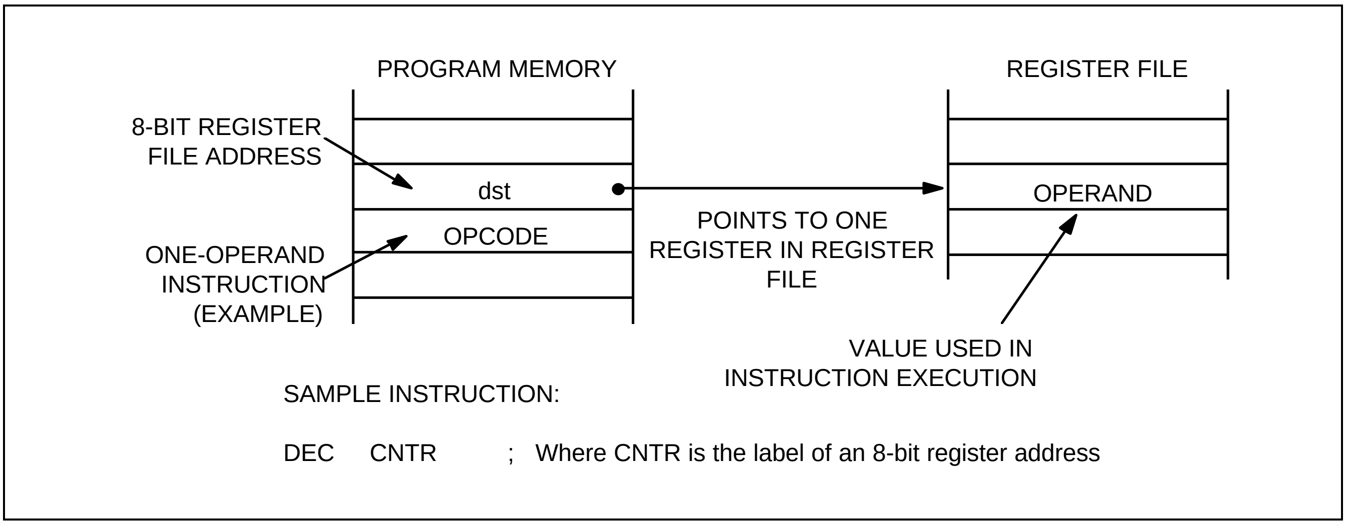


Figure 3-1. Register Addressing

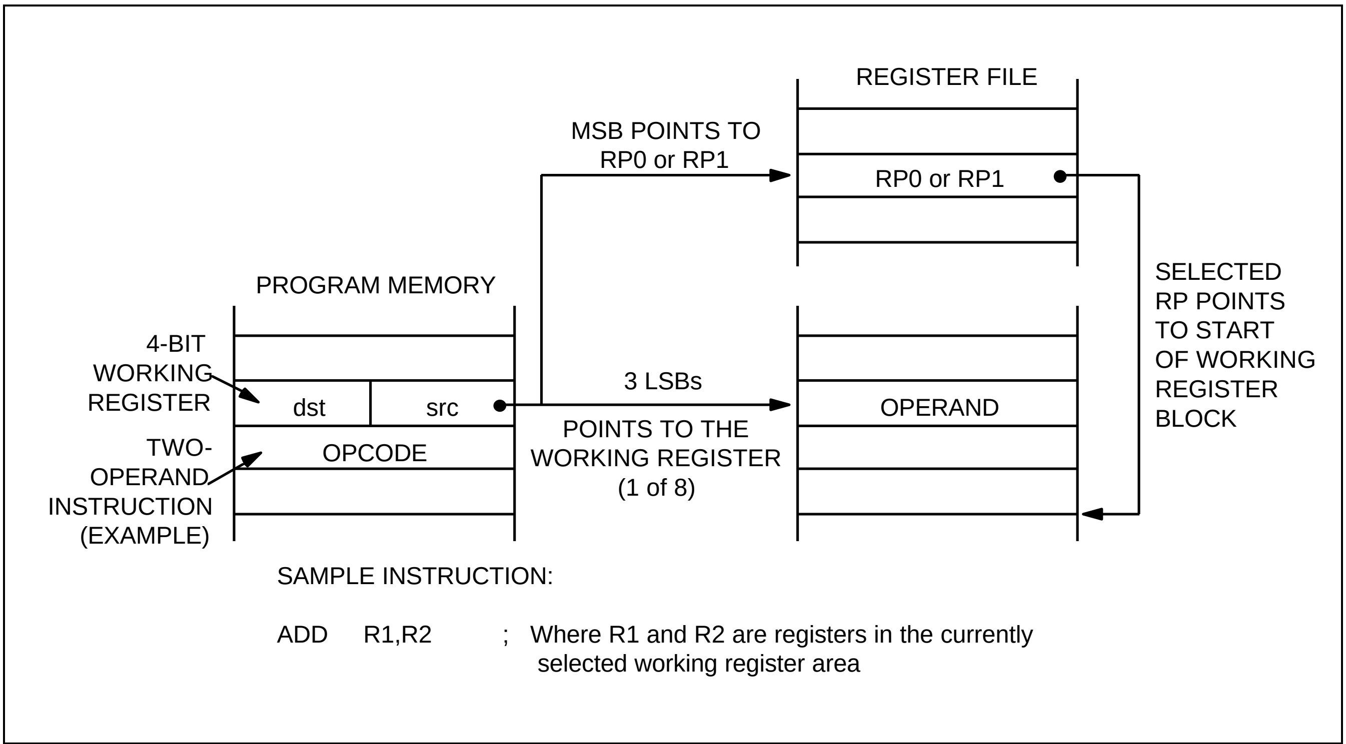


Figure 3-2. Working Register Addressing

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.

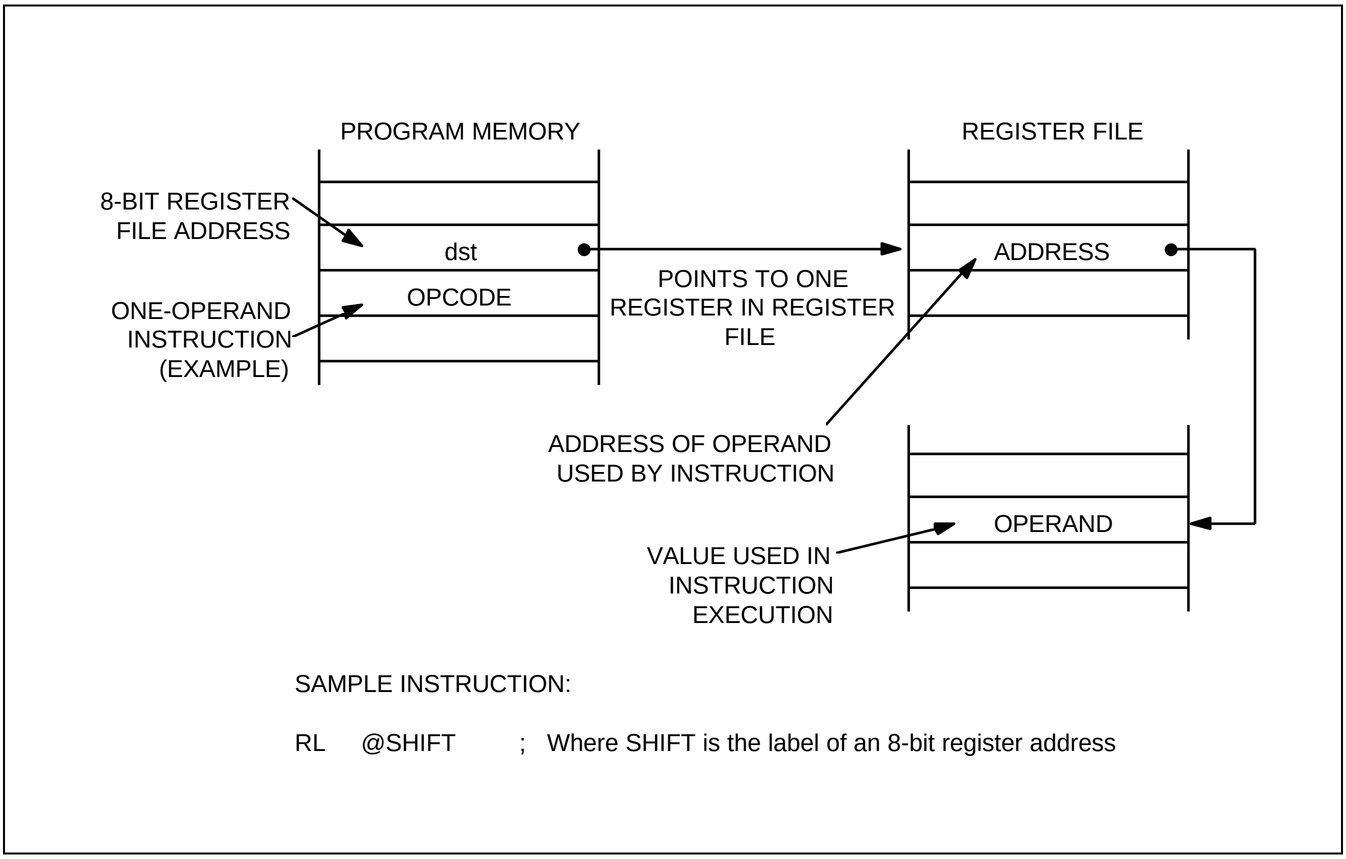


Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

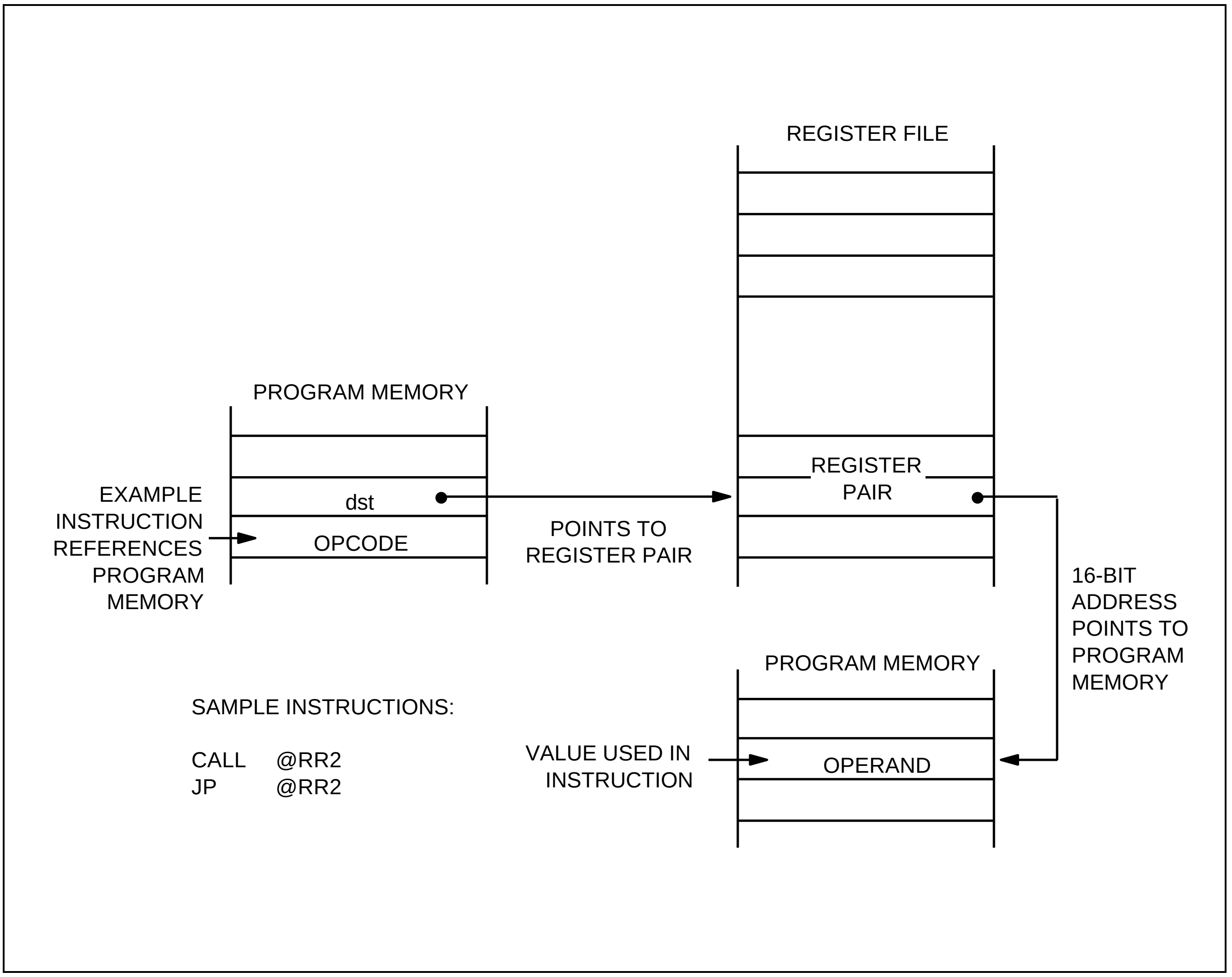


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

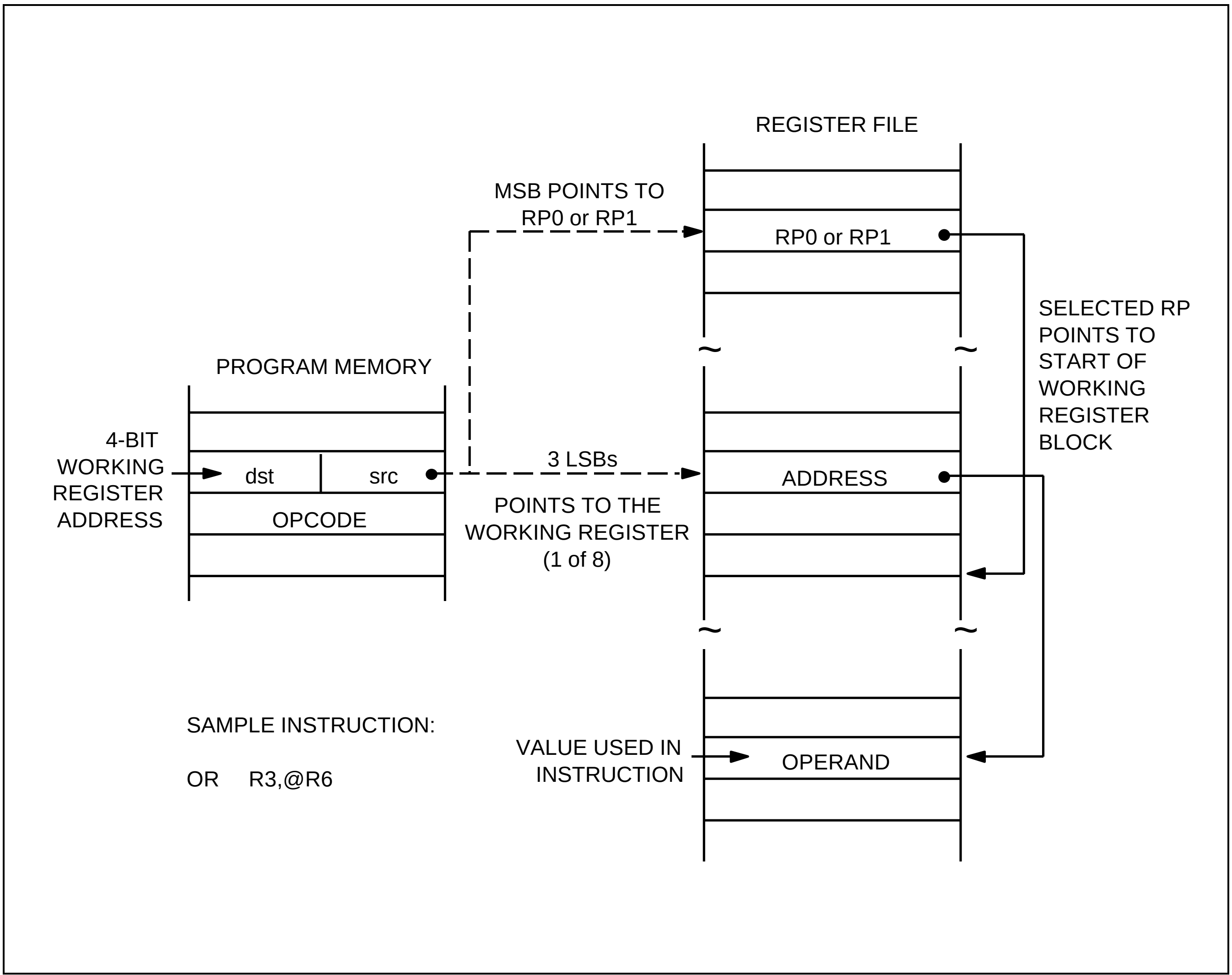
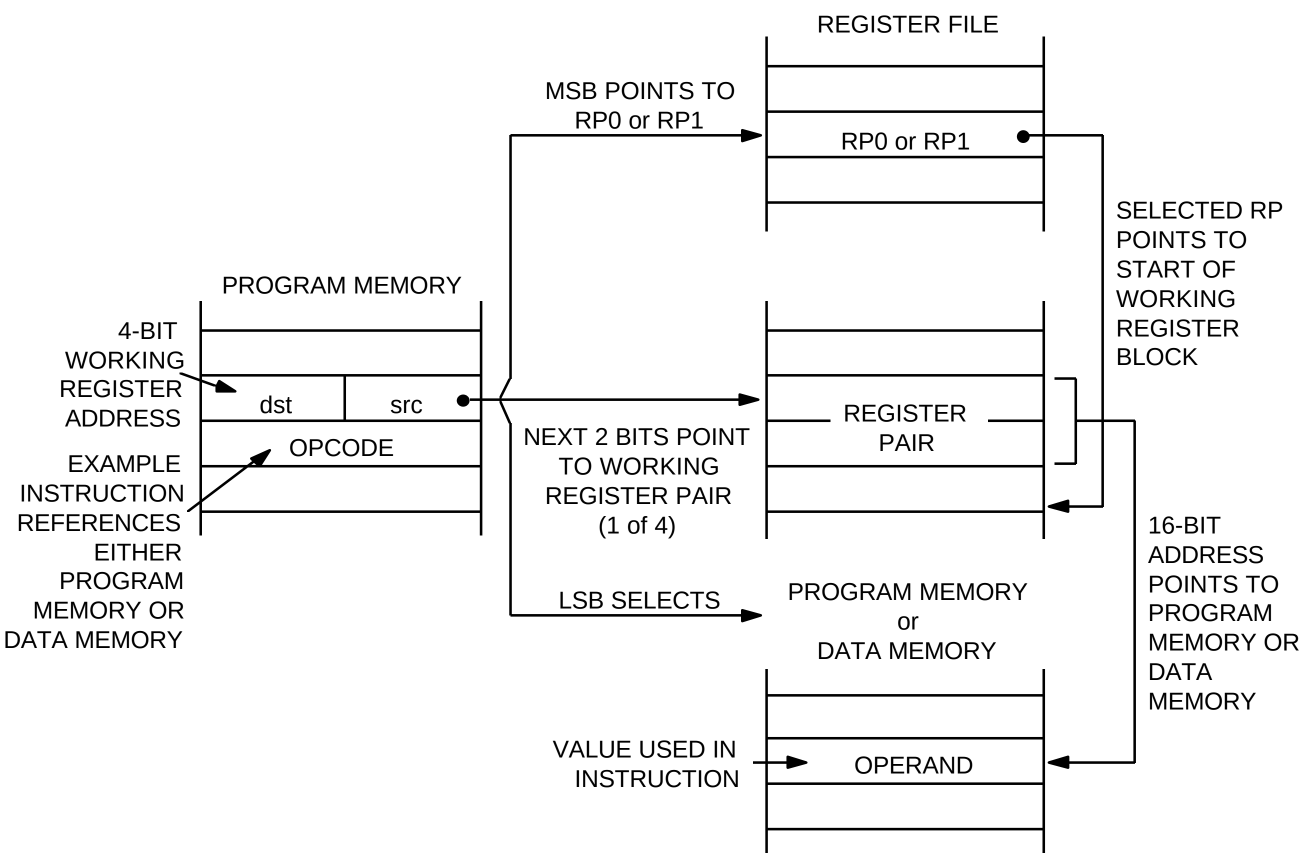


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)



SAMPLE INSTRUCTIONS:

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

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

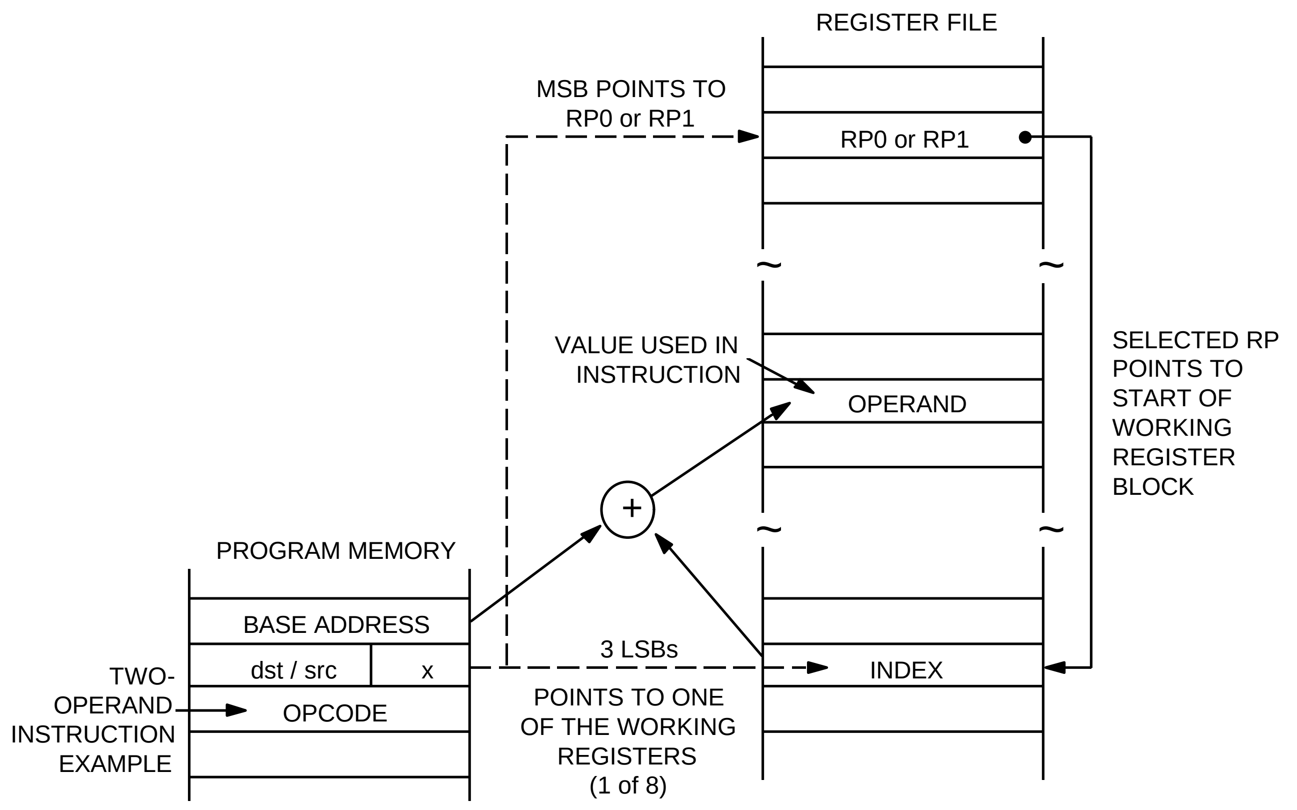
INDEXED ADDRESSING MODE (X)

Indexed (X) addressing mode adds an offset value to a base address during instruction execution in order to calculate the effective operand address (see Figure 3-7). You can use Indexed addressing mode to access locations in the internal register file or in external memory (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 memory and for external data memory (if implemented).

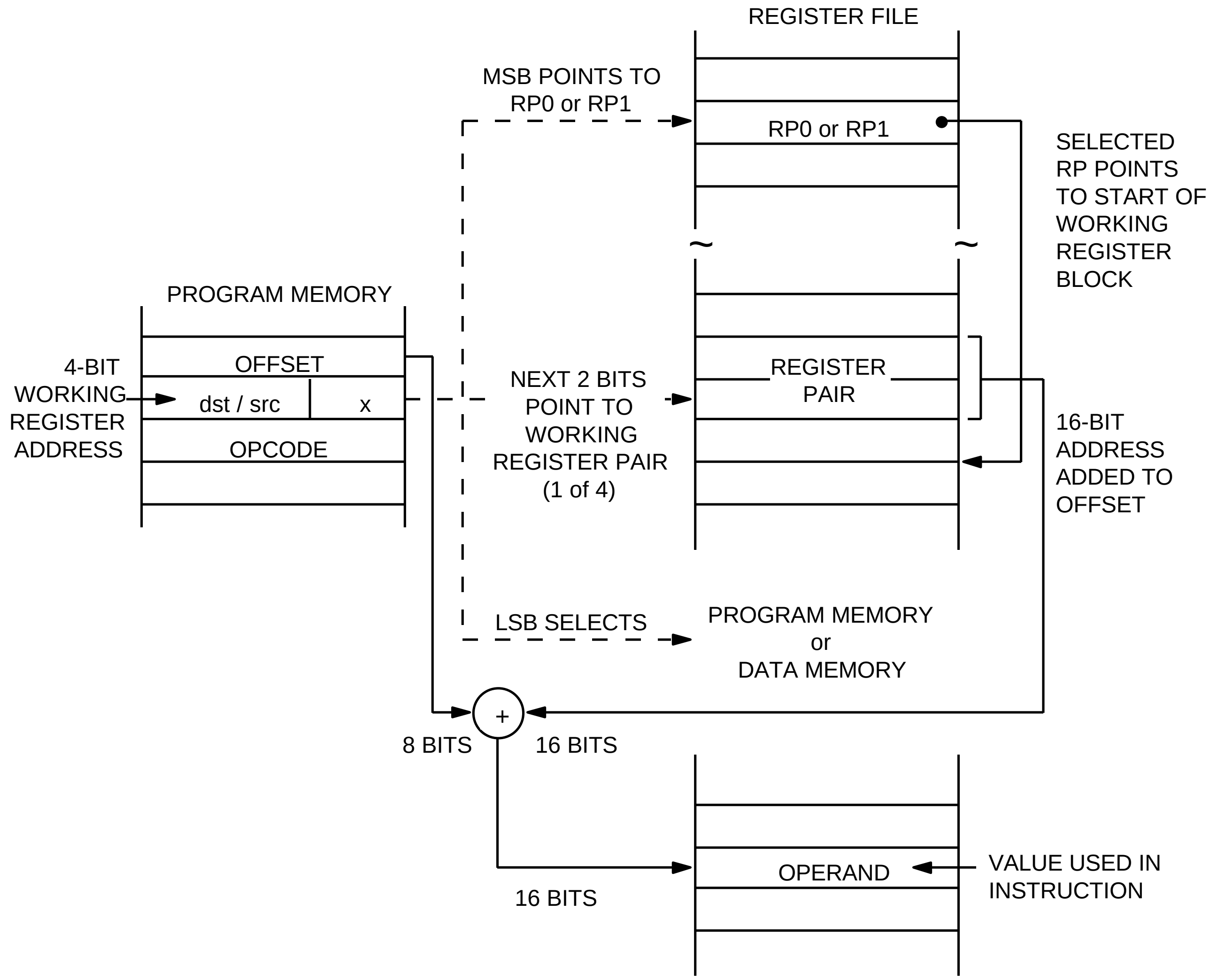


SAMPLE INSTRUCTION:

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

Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)

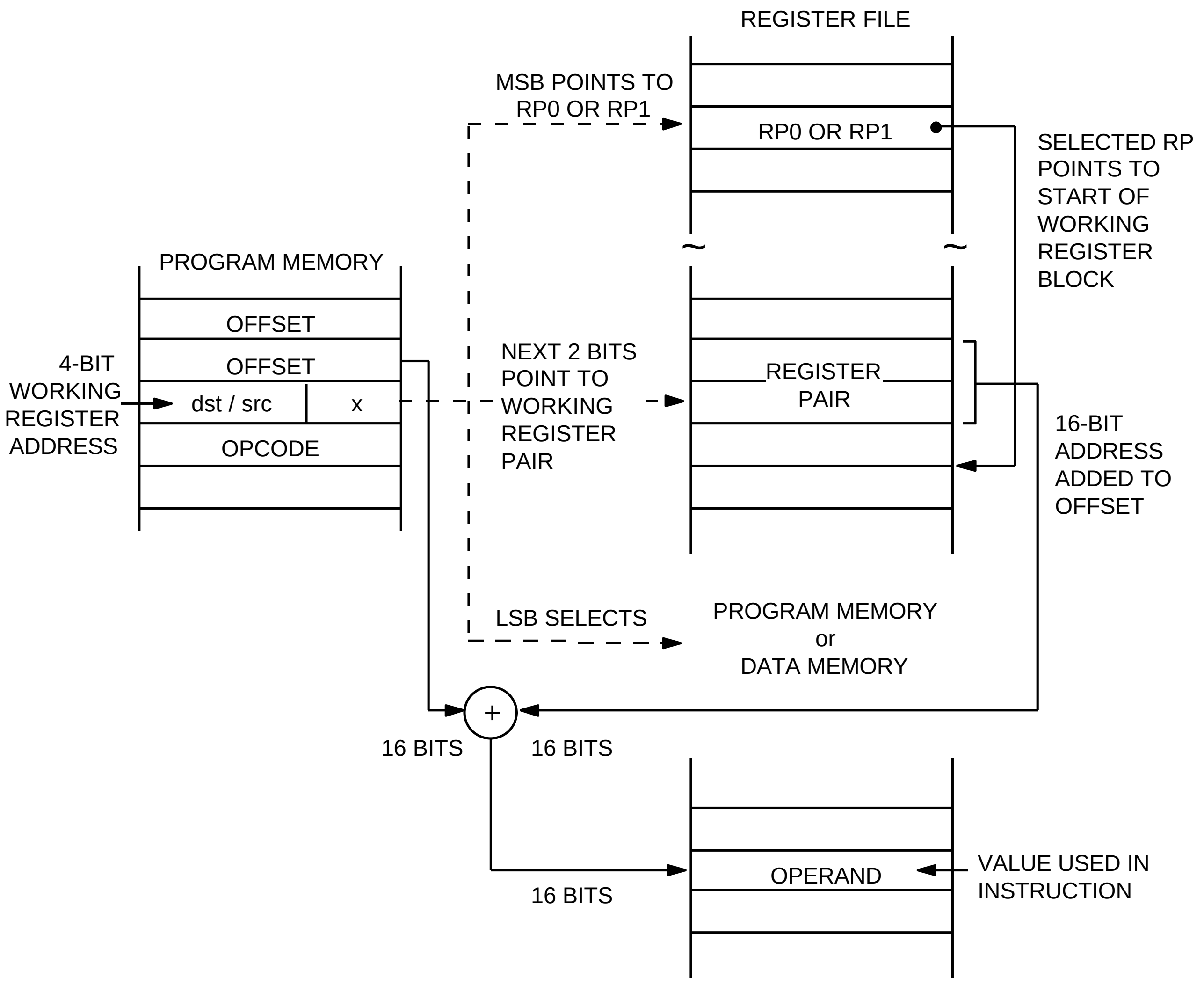


SAMPLE INSTRUCTIONS:

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

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

INDEXED ADDRESSING MODE (Concluded)



SAMPLE INSTRUCTIONS:

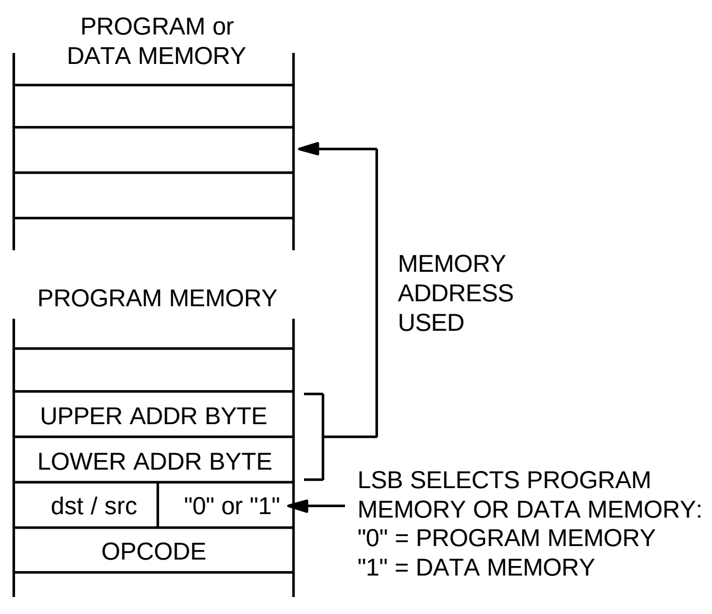
LDC	R4,#1000H[RR2]	; The values in the program address (RR2 + 1000H) are loaded into register R4.
LDE	R4,#1000H[RR2]	; Identical operation to LDC example, except that external program memory is accessed.

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

DIRECT ADDRESS MODE (DA)

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

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

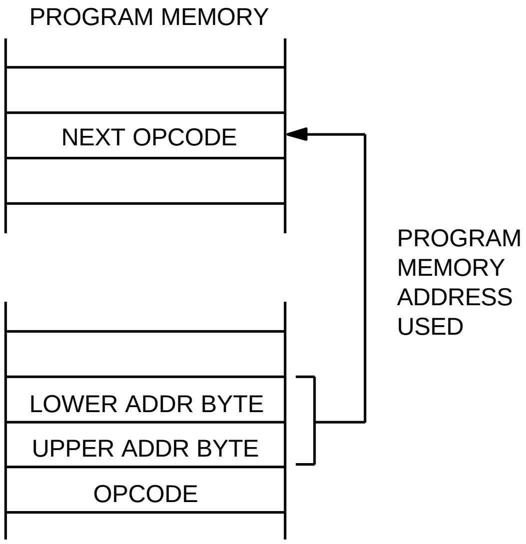


SAMPLE INSTRUCTIONS:

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

Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)



SAMPLE INSTRUCTIONS:

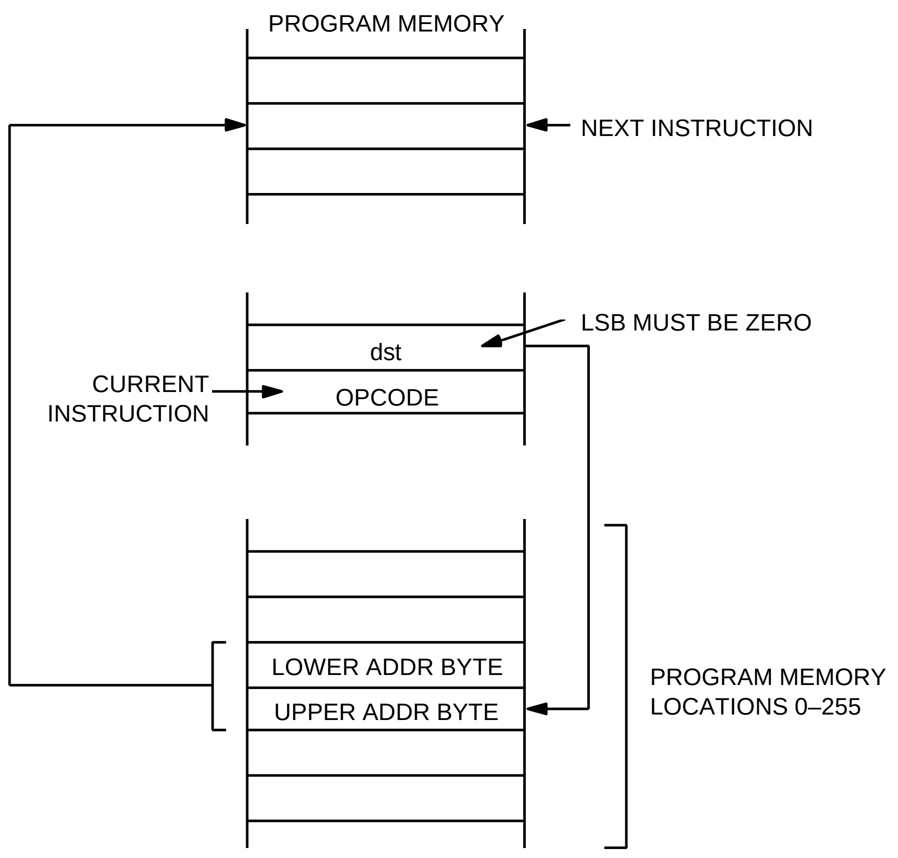
```
JP      C,JOB1      ;  Where JOB1 is a 16-bit immediate address
CALL    DISPLAY     ;  Where DISPLAY is a 16-bit immediate address
```

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

INDIRECT ADDRESS MODE (IA)

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

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



SAMPLE INSTRUCTION:

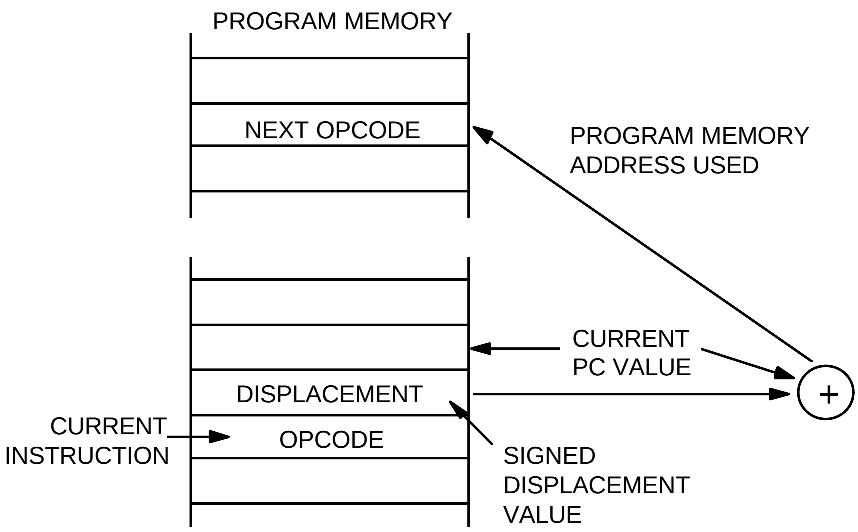
CALL #40H ; The 16-bit value in program memory addresses 40H and 41H is the subroutine start address.

Figure 3-12. Indirect Addressing

RELATIVE ADDRESS MODE (RA)

In Relative Address (RA) mode, a 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.



SAMPLE INSTRUCTION:

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

Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

In Immediate (IM) 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.

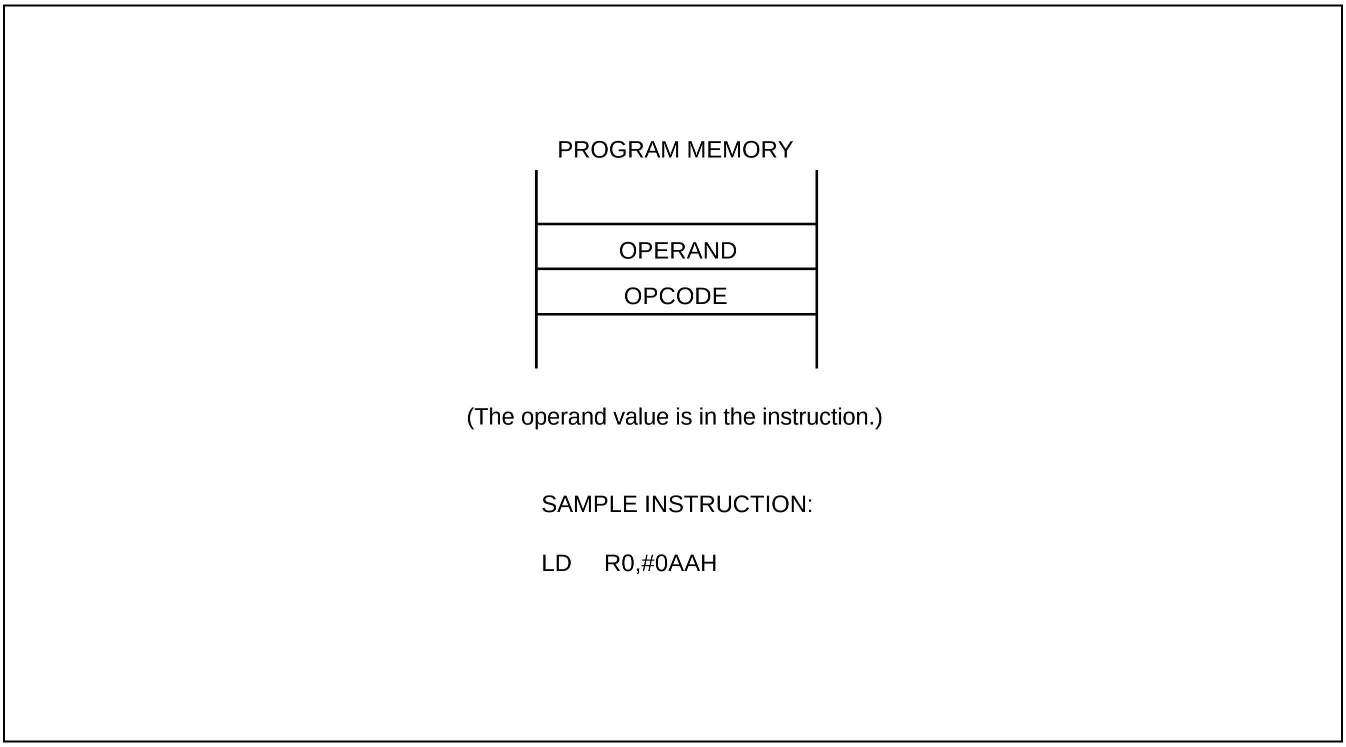


Figure 3-14. Immediate Addressing

4 CONTROL REGISTERS

In this section, detailed descriptions of the S3C8618/C8615 control registers are presented in an easy-to-read format. You can use this section as a quick-reference source when writing application programs.

The locations and read/write characteristics of all mapped registers in the S3C8618/C8615 register files are presented in Tables 4-1, 4-2, and 4-3. The hardware reset values for these registers are described in Section 8, "RESET and Power-Down."

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.

Table 4-1. Set 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Timer 0 counter register	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

NOTE: You cannot use a read-only register (T0CNT, IRQ) as a destination field for the instructions OR, AND, LD, or LDB.

Table 4-2. Set 1, Bank 0 Registers

Register Name	Mnemonic	Decimal	Hex	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
Port 0 control register (high byte)	P0CONH	228	E4H	R/W
Port 0 control register (low byte)	P0CONL	229	E5H	R/W
Port 1 control register	P1CON	230	E6H	R/W
Port 2 control register (high byte)	P2CONH	231	E7H	R/W
Port 2 control register (low byte)	P2CONL	232	E8H	R/W
Port 3 control register (high byte)	P3CONH	233	E9H	R/W
Port 3 control register (low byte)	P3CONL	234	EAH	R/W
Port 0 external interrupt control register	P0INT	235	EBH	R/W
Location ECH–EFH is not mapped				

Table 4-2. Set 1, Bank 0 Registers (Continued)

Register Name	Mnemonic	Decimal	Hex	R/W
DDC control register	DDCCON	240	F0H	R/W
Sync control register 0	SYNCON0	241	F1H	R/W
Sync control register 1	SYNCON1	242	F2H	R/W
Sync control register 2	SYNCON2	243	F3H	R/W
Sync port read data register	SYNCRD	244	F4H	R (note)
Timer 1 control register	T1CON	245	F5H	R/W
Timer 1 counter register	T1CNT	246	F6H	R/W
Timer 2 control register	T2CON	247	F7H	R/W
Timer 2 data register	T2DATA	248	F8H	R/W
Locations F9H–CH are not mapped.				
Basic timer counter register	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 (SYNCRD, BTCNT) as a destination field for the instructions OR, AND, LD, or LDB.

Table 4-3. Set 1, Bank 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
PWM 0 data register	PWM0	224	E0H	R/W
PWM 1 data register	PWM1	225	E1H	R/W
PWM 2 data register	PWM2	226	E2H	R/W
PWM 3 data register	PWM3	227	E3H	R/W
PWM 4 data register	PWM4	228	E4H	R/W
PWM 5 data register	PWM5	229	E5H	R/W
PWM 6 data register	PWM6	230	E6H	R/W
PWM counter register	PWMCNT	231	E7H	R (note)
PWM control register	PWMCON	232	E8H	R/W
IIC-bus clock control register	ICCR	233	E9H	R/W
IIC-bus control/status register	ICSR	234	EAH	R/W
IIC-bus address register	IAR	235	EBH	R/W
IIC-bus Tx/Rx data shift register	IDSR	236	ECH	R/W
Locations EDH–FFH are not mapped.				

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

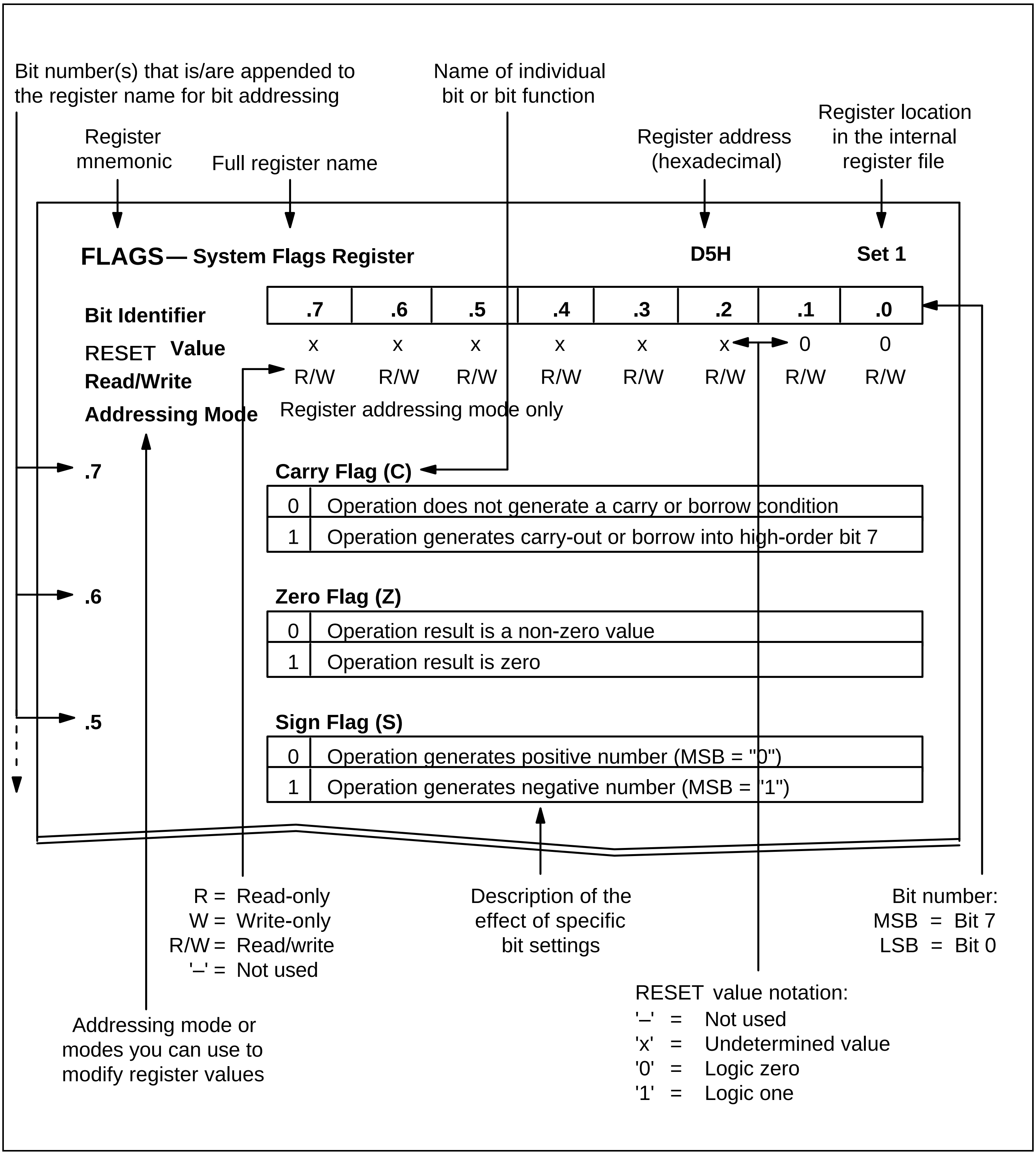


Figure 4-1. Register Description Format

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 Reset)**

1	0	1	0	Disable watchdog timer function
Any other value				Enable watchdog timer function

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

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/1024$
1	0	$f_{OSC}/128$
1	1	Invalid setting; not used for S3C8618/C8615.

.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 basic timer and timer 0 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".

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
1	1	No effect

.4 and .3 **CPU Clock (System Clock) Selection Bits** (1)

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

.2 – .0 **Subsystem Clock Selection Bits** (3)

1	0	1	Invalid setting for S3C8618/C8615.
Other value			Select main system clock (MCLK)

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. If the oscillator frequency is higher than 12 MHz, this selection is invalid.
3. These selection bits are required only for systems that have a main clock and a subsystem clock. The S3C8618/C8615 use only the main oscillator clock circuit. For this reason, the setting '101B' is invalid.

DDCCON — DDC Control Register F0H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 – .2 Not used for the S3C8618/C8615.

.1 **Normal IIC-Bus Mode/DDC1 Tx Mode Selection Bit** (1)

0	Normal IIC-bus mode; SCL pin selected
1	DDC1 transmit (Tx) mode; VCLK pin selected

.0 **SCL Pin Falling Edge Detection Flag** (2)

0	SCL pin level remains High following a reset (when read)
0	*This bit can be cleared by "0" S/W written (when write)*
1	Falling edge detected at the SCL pin following a reset or after this flag was cleared by software (when read)
1	*No effect (when write)*

NOTES:
1. DDC2B and DDC2AB (access bus) modes are supported by the multi-master IIC-bus physically and by S/W in EDID and VDIF formats.
2. DDCCON.0 is automatically set to “1” if a falling signal edge is detected at the SCL pin following a hardware reset, or after the flag has been cleared by software.

EMT — External Memory Timing Register FEH Set 1, Bank 0

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

.7 **External WAIT Input Function Enable Bit**

0	Disable WAIT input function for external device
1	Enable WAIT input function for external device

.6 **Slow Memory Timing Enable Bit**

0	Disable slow memory timing
1	Enable slow memory timing

.5 and .4 **Program Memory Automatic Wait Control Bits**

0	0	No wait (Normal Operation)
0	1	Wait one cycle
1	0	Wait two cycles
1	1	Wait three cycles

.3 and .2 **Data Memory Automatic Wait Control Bits**

0	0	No wait (Normal Operation)
0	1	Wait one cycle
1	0	Wait two cycles
1	1	Wait three cycles

.1 **Stack Area Selection Bit**

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

.0 Not used for S3C8618/C8615.

NOTE: Because an external periperal interface is not implemented in the S3C8618/C8615, all EMT register is not used. The program initialization routine should clear the EMT register to '00H' after a reset. Modification of EMT values during normal operation may cause a system malfunction.

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/W	R/W
Addressing Mode	Register addressing mode only							

.7 **Carry Flag (C)**

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

.6 **Zero Flag (Z)**

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

.5 **Sign Flag (S)**

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

.4 **Overflow Flag (V)**

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

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

0	Add operation completed
1	Subtraction operation completed

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

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

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

0	Cleared automatically during an interrupt return (IRET)
1	Automatically set to logic one during a fast interrupt service routine

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

0	Bank 0 is selected (by executing the instruction SB0)
1	Bank 1 is selected (by executing the instruction SB1)

IAR — Multi-Master IIC-Bus Address Register EBH Set 1, Bank 1

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

.7–.1 **7-Bit Slave Address Bits**

These bits are operated only when receive the slave address & general call.
Write enable when ICSR, 4 is “0”, but read enable anytime.

.0 Not mapped at S3C8618/C8615.

ICCR — Multi-Master IIC-Bus Clock Control Register E9H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	1	1	1	1
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 **Acknowledgement signal Enable/Disable Bit**

0	Disable Acknowledgement signal
1	Enable Acknowledgement signal

.6 **Transmit Clock Selection Bit**

0	$f_{OSC}/16$
1	$f_{OSC}/512$

.5 **Multi Master IIC-Bus Transmit/Receive Interrupt Enable Bit**

0	Disable interrupt
1	Enable interrupt

.4 **Multi Master IIC-Bus Transmit/Receive Interrupt Pending Bit**

0	When write "0" to this bit or ICSR.4 bit is "0"
1	When 1-byte transmit / receive is terminated, general call or slave address match occurred, or arbitration lost

.3 – 0. **ICCR3–ICCR0: Transmit Clock 4-bit Prescaler Bits**

SCL clock = IICLK/ICCR<3:0> + 1

Where, IICLK = $f_{OSC}/16$ when ICCR.6 = “0”, IICLK = $f_{OSC}/512$ when ICCR.6 = “1”

ICSR— Multi-Master IIC-Bus Control/Status Register EAH Set 1, Bank 1

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

Addressing Mode Register addressing mode only

.7 and .6 **Master/Slave, Transmit / Receive Mode Selection Bits**

0	0	Slave receive mode (Default mode)
0	1	Slave transmit mode
1	0	Master receive mode
1	1	Master transmit mode

.5 **IIC-Bus Busy Bit**

0	IIC-bus is not busy (read), IIC-bus interface stop signal generation (when write)
1	IIC-bus is busy (read), IIC-bus interface start signal generation (when write)

.4 **Serial Output Enable Bit**

0	Disable serial data transmit / receive
1	Enable serial data transmit / receive

.3 **Arbitration Lost Bit**

This bit is set by H/W when the serial I/O interface, as Master transmit mode, loses a bus arbitration procedure
As slave mode this flag is set to "1", in case of writing a data to ICSR when ICSR.5 is "1"

.2 **Address-As-Slave**

0	When START/STOP condition was generated
1	When received slave address match to IAR register

.1 **Address Zero Bit**

0	When START/STOP condition was generated
1	When received slave address is ‘0000000B’ (general call)

.0 **Last-Received Bit**

0	Last-received bit is “0” (Acknowledgment signal was received)
1	Last-received bit is “1” (Acknowledgment signal was not received)

IDSR — Multi-Master IIC-Bus Tx/Rx Data Shift Register ECH Set 1, Bank 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 **Multi Master IIC-bus Transmit/Receive Data Shift Bits**

Write enable when ICSR.4 "1", but Read enable anytime

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
Read/Write	–	–	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Not used for S3C8618/C8615.

.6 Not used for S3C8618/C8615.

.5 **Interrupt Level 5 (IRQ5) Enable Bit; Vsync Detection Interrupt**

0	Disable IRQ5 interrupt
1	Enable IRQ5 interrupt

.4 **Interrupt Level 4 (IRQ4) Enable Bit; P0.1 or P0.2 External Interrupt (INT1, INT2)**

0	Disable IRQ4 interrupt
1	Enable IRQ4 interrupt

.3 **Interrupt Level 3 (IRQ3) Enable Bit; P0.0 External Interrupt (INT0)**

0	Disable IRQ3 interrupt
1	Enable IRQ3 interrupt

.2 **Interrupt Level 2 (IRQ2) Enable Bit; Multi-Master IIC-bus Tx/Rx Interrupt**

0	Disable IRQ2 interrupt
1	Enable IRQ2 interrupt

.1 **Interrupt Level 1 (IRQ1) Enable Bit; Timer 2 match/Timer 1 overflow interrupt**

0	Disable IRQ1 interrupt
1	Enable IRQ1 interrupt

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

0	Disable IRQ0 interrupt
1	Enable IRQ0 interrupt

NOTE: Only levels IRQ0–IRQ5 are implemented in the S3C8618/C8615 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, Bank 0

Bit Identifier	**.7**	**.6**	**.5**	**.4**	**.3**	**.2**	**.1**	**.0**
RESET Value	x	–	–	x	x	x	x	x
Read/Write	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** (1)

0	0	0	Not used
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	Not used

.6 Not used for S3C8618/C8615.

.5 Not used for S3C8618/C8615.

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

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

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

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

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

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

NOTES:
1. Interrupt group A is IRQ0 and IRQ1; interrupt group B is IRQ2, IRQ3, and IRQ4; interrupt group C is IRQ5 and IRQ6.
2. Interrupt level IRQ6-7 is not used in the S3C8618/C8615 interrupt structure. For this reason, IPR.6 (subgroup C) is not used.

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
Read/Write	–	–	R	R	R	R	R	R

Addressing Mode Register addressing mode only

.7 Not used for S3C8618/C8615.

.6 Not used for S3C8618/C8615.

.5 **Level 5 (IRQ5) Request Pending Bit; Vsync Detection Interrupt**

0	No IRQ5 interrupt pending
1	IRQ5 interrupt is pending

.4 **Level 4 (IRQ4) Request Pending Bit; P0.1 or P0.2 External Interrupt**

0	No IRQ4 interrupt pending
1	IRQ4 interrupt is pending

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

0	No IRQ3 interrupt pending
1	IRQ3 interrupt is pending

.2 **Level 2 (IRQ2) Request Pending Bit; Multi-Master IIC-Bus Tx/Rx interrupt**

0	No IRQ2 interrupt pending
1	IRQ2 interrupt is pending

.1 **Level 1 (IRQ1) Request Pending Bit; Timer 2 Match/Timer 1 Overflow Interrupt**

0	No IRQ1 interrupt pending
1	IRQ1 interrupt is pending

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

0	No IRQ0 interrupt pending
1	IRQ0 interrupt is pending

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

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

.7 and .6 **P0.7 Mode Selection Bits**

0	0	Input mode
0	1	Input mode
1	X	Push-pull output mode

.5 and .4 **P0.6 Mode Selection Bits**

0	0	Input mode
0	1	Input mode
1	X	Push-pull output mode

.3 and .2 **P0.5/T0OUT Mode Selection Bits**

0	0	Input mode
0	1	Multiplexed input mode (P0.5, T1CK)
1	X	Push-pull output mode

.1 and .0 **P0.4/T0CAP Mode Selection Bits**

0	0	Input mode
0	1	Multiplexed input mode (P0.4, T0CAP)
1	X	Push-pull output mode

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

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

Addressing Mode Register addressing mode only

.7 and .6 **P0.3 Mode Selection Bits**

0	0	Input mode
0	1	Input mode, rising edge interrupt detection
1	0	Input mode, falling edge interrupt detection
1	1	Push-pull output mode

.5 and .4 **P0.2/INT2 Mode Selection Bits**

0	0	Input mode (P0.2)
0	1	Input mode, rising edge interrupt detection (INT2)
1	0	Input mode, falling edge interrupt detection (INT2)
1	1	Push-pull output mode

.3 and .2 **P0.1/INT1 Mode Selection Bits**

0	0	Input mode (P0.1)
0	1	Input mode, rising edge interrupt detection (INT1)
1	0	Input mode, falling edge interrupt detection (INT1)
1	1	Push-pull output mode

.1 and .0 **P0.0/INT0 Mode Selection Bits**

0	0	Input mode (P0.0)
0	1	Input mode, rising edge interrupt detection (INT0)
1	0	Input mode, falling edge interrupt detection (INT0)
1	1	Push-pull output mode

P0INT — Port 0 External Interrupt Control Register EBH Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 Not used for S3C8618/C8615.

.6 **P0.2 External Interrupt (IRQ4) Pending Flag** (note)

0	No P0.2 external interrupt pending (when read)
0	*Clear P0.2 interrupt pending condition (when write)*
1	P0.2 external interrupt is pending (when read)

.5 **P0.1 External Interrupt (IRQ4) Pending Flag**

0	No P0.1 external interrupt pending (when read)
0	*Clear P0.1 interrupt pending condition (when write)*
1	P0.1 external interrupt is pending (when read)

.4 **P0.0 External Interrupt (IRQ3) Pending Flag**

0	No P0.0 external interrupt pending (when read)
0	*Clear P0.0 interrupt pending condition (when write)*
1	P0.0 external interrupt is pending (when read)

.3 Not used for S3C8618/C8615.

.2 **P0.2 External Interrupt (IRQ4) Enable Bit**

0	Disable P0.2 interrupt
1	Enable P0.2 interrupt

.1 **P0.1 External Interrupt (IRQ4) Enable Bit**

0	Disable P0.1 interrupt
1	Enable P0.1 interrupt

.0 **P0.0 External Interrupt (IRQ3) Enable Bit**

0	Disable P0.0 interrupt
1	Enable P0.0 interrupt

NOTE: Writing a “1” to an interrupt pending flag (P0.2, P0.1, P0.0) has no effect.

P1CON — Port 1 Control Register E6H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 and .6 **P1.3 Mode Selection Bits**

0	x	Input mode
1	x	Push-pull output mode

.5 and .4 **P1.2 Mode Selection Bits**

0	x	Input mode
1	x	Push-pull output mode

.3 and .2 **P1.1 Mode Selection Bits**

0	x	Input mode
1	x	Push-pull output mode

.1 and .0 **P1.0 Mode Selection Bits**

0	x	Input mode
1	x	Push-pull output mode

NOTE: 'x' means don't care.

P2CONH — Port 2 Control Register (High Byte) E7H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 and .6 **P2.7/Csync-I**

0	x	TTL input mode (Csync-I)
1	x	Push-pull output mode

.5 and .4 **P2.6/PWM6 Mode Selection Bits**

0	0	Input mode
0	1	Push-pull output mode
1	0	Push-pull PWM output mode (up to 5-volt load capability)
1	1	N-channel open-drain PWM output mode (up to 5-volt load capability)

.3 and .2 **P2.5/PWM5 Mode Selection Bits**

0	0	Input mode
0	1	Push-pull output mode
1	0	Push-pull PWM output mode (up to 5-volt load capability)
1	1	N-channel open-drain PWM output mode (up to 5-volt load capability)

.1 and .0 **P2.4/PWM4 Mode Selection Bits**

0	0	Input mode
0	1	Push-pull output mode
1	0	Push-pull PWM output mode (up to 5-volt load capability)
1	1	N-channel open-drain PWM output mode (up to 5-volt load capability)

P2CONL — Port 2 Control Register (Low Byte) E8H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 and .6 **P2.3/PWM3 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	Push-pull PWM output mode (up to 5-volt load capability)

.5 and .4 **P2.2/PWM2 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	Push-pull PWM output mode (up to 5-volt load capability)

.3 and .2 **P2.1/PWM1 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	Push-pull PWM output mode (up to 5-volt load capability)

.1 and .0 **P2.0/PWM0 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	Push-pull PWM output mode (up to 5-volt load capability)

P3CONH — Port 3 Control Register (High Byte) E9H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

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

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode (5-volt load capability)

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

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode (5-volt load capability)

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

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode (5-volt load capability)

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

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode (5-volt load capability)

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

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

Addressing Mode Register addressing mode only

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

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode

.5 and .4 **P3.2 Mode Selection Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode

.3 and .2 **P3.1 Mode Selection Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode

.1 and .0 **P3.0 Mode Selection Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode

PP — Register Page Pointer DFH Set 1

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

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

0	0	0	0	Destination: page 0
0	0	0	1	Not used for S3C8618/C8615.
• • •				"
1	1	1	1	Not used for S3C8618/C8615.

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

0	0	0	0	Source: page 0
0	0	0	1	Not used for S3C8618/C8615.
• • •				"
1	1	1	1	Not used for S3C8618/C8615.

NOTE: Because only page 0 is implemented in the S3C8618/C8615 microcontrollers, the register page pointer always points to page 0 as the source and destination address.

PWMCON — PWM Control Register E7H Set 1, Bank 1

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

Addressing Mode Register addressing mode only

.7 and .6 **2-Bit Prescaler value bits for PWM Counter Input Clock**

0	0	Non-divided input clock
0	1	Divided-by-two input clock
1	0	Divided-by-three input clock
1	1	Divided-by-four input clock

.5 **PWM Counter Enable Bit**

0	Stop PWM counter operation (No Current leakage)
1	Start (or resume) PWM counter operation

.4–.0 Not used for S3C8618/C8615.

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 18 8-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 S3C8618/C8615.

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 18 8-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 S3C8618/C8615.

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** (1)

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 S3C8618/C8615.

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

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	Not used for S3C8618/C8615.
1	1	1	Not used for S3C8618/C8615.

.1 **Fast Interrupt Enable Bit** (3)

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0 **Global Interrupt Enable Bit** (4)

0	Disable global interrupt processing
1	Enable global interrupt processing

NOTES:
1. Because an external interface is not implemented for the S3C8618/C8615, SYM.7 must always be "0".
2. You can select only one interrupt level at a time for fast interrupt processing.
3. Setting SYM.1 to "1" enables fast interrupt processing for the interrupt level currently selected by SYM.2–SYM.4.
4. Following a reset, you must enable global interrupt processing by executing an EI instruction (not by writing a "1" to SYM.0).

SYNCON0 — Sync Processor Control Register 0 F1H Set 1, Bank 0

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

.7 **Sync Input Selection Bit**

0	Hsync-I input is selected
1	Csync-I input is selected

.6 **Input Edge Selection Bit for 5-Bit Capture/Compare Counter**

0	Falling edges input is selected
1	Rising edges input is selected

.5 **Mode Selection Bit for 5-Bit Capture/Compare Counter**

0	Capture mode (1)
1	Compare mode (2)

.4 – .0 **5-Bit Compare/Capture Counter**

5-bit counter increase when high level is detected, but overflow dose not occurs (Stop at detected "11111") and decrease when low level is detected, but underflow dose not occurs (Stop at detected "11111") When Capture mode: Sync polarity check mode (5-bit counter value is loaded into this register by an edge) When Compare mode: Sync separation and output mode (When counter value increase to "11111", output high through the MUX and when counter value decrease to "00000", output low Resume previous status when "11111" > counter value > "00000")

NOTES:
1. 5-bit counter value is captured when every selected edge is detected (Output to Multiplexer disabled: Vsync-O pin is low when SYNCON1.0 = 1, SYNCON1.2 = 0, SYNCON2.4, 3, 1 and 0 = 0)
2. In this mode, Output to Multiplexer is enabled. When the 5-bit counter value is '11111B', the output is High; when the counter value is '00000B', the output is low. Whenever the 5-bit counter value is less than '11111B' and greater than '00000B', the previous output level is retained.

SYNCON1 — Sync Processor Control Register 1 F2H Set 1, Bank 0

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

.7 **Hsync, Csync, Sync-on-Green (SOG) Compare Output Bit**

0	Hsync-I is not the same as Csync-I (no SOG)
1	Hsync-I is the same as Csync-I (SOG)

.6 and .5 **Clamp Signal Generator Selection Bits**

0	0	Inhibit Clamp signal output (Clamp-O)
0	1	(f_{OSC} ∞ 2) clock pulse output (250 ns at 8 MHz f_{OSC})
1	0	(f_{OSC} ∞ 4) clock pulse output (500 ns at 8 MHz f_{OSC})
1	1	(f_{OSC} ∞ 8) clock pulse output (1 µs at 8 MHz f_{OSC})

.4 **"Front Porch" / "Back Porch" Mode Selection Bit**

0	Generate Clamp-O after *falling* edge of Hsync ("back porch" mode)
1	Generate Clamp-O after *rising* edge of Hsync ("front porch" mode)

.3 **Clamp Signal Polarity Control Bit**

0	Negative polarity
1	Positive polarity

.2 **Vsync-O Status Control Bit**

0	Do not invert (by-pass)
1	Invert output signal

.1 **Hsync-O Status Control Bit**

0	Do not invert (by-pass)
1	Invert output signal

.0 **Vsync-O Source Selection Bit**

0	Vsync-I port (VCLK) input is selected
1	5-bit comparator output is selected

SYNCON2 — Sync Processor Control Register 2 F3H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 **Timer 0 Capture Input Selection Bit**

0	T0CAP input pin is selected
1	Vsync output path is selected

.6 **Factory Test Mode Selection Bit**

0	Normal operating mode
1	Factory test mode (pseudo sync generator mode)

.5 **5-Bit Counter Clock Source Selection Bit**

0	$f_{OSC}/2$
1	$f_{OSC}/3$

.4 and .3 **Pseudo Vsync Generator Output Control Bits (Positive polarity only, 5% duty)**

0	0	Normal sync processor operating mode
0	1	60.1 Hz (8-MHz f_{OSC} and SYNCON2.5 = “0”)
1	0	78.1 Hz (8-MHz f_{OSC} and SYNCON2.5 = “0”)
1	1	86.8 Hz (8-MHz f_{OSC} and SYNCON2.5 = “0”)

.2 and .1 **Pseudo Hsync Generator Output Control Bits (Positive polarity only, 25% duty)**

0	0	Normal sync processor operating mode
0	1	37.04 kHz (8-MHz f_{OSC} and SYNCON2.5 = “0”)
1	0	47.62 kHz (8-MHz f_{OSC} and SYNCON2.5 = “0”)
1	1	58.82 kHz (8-MHz f_{OSC} and SYNCON2.5 = “0”)

.0 **Vsync Detection Interrupt Enable Bit**

0	Disable interrupt
1	Enable interrupt (interrupt requests triggered by rising edge)

NOTE: Vsync detection interrupt pending conditions are automatically cleared by hardware after the CPU acknowledges the request and the interrupt service routine is initiated.

SYNCRD — Sync Processor Port Read Data Register F4H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7–.4 Not used for S3C8618/C8615.

.3 **Vertical Sync Signal Output Data Bit (Vsync-O)**

0	Low data
1	High data

.2 **Horizontal Sync Signal Output Data Bit (Hsync-O)**

0	Low data
1	High data

.1 **Vertical Sync Signal Input Data Bit (Vsync-I)**

0	Low data
1	High data

.0 **Horizontal Sync Signal Input Data Bit (Hsync-I)**

0	Low data
1	High data

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 Timer 0 Input Clock Selection Bits

0	1	CPU clock/128
1	0	CPU clock/8

.6 and .5 **2 Bit Prescaler**

0	0	No division
0	1	Divided by 2
1	0	Divided by 3
1	1	Divided by 4

.4 and .3 **Timer 0 Operating Mode Selection Bits**

0	0	Interval timer mode (counter cleared by match signal)
0	1	Capture mode (rising edges only, counter keeping, OVF interrupt can occur)
1	0	Capture mode (falling edges only, counter keeping, OVF interrupt can occur)
1	1	Not used

.2 **Timer 0 Counter Clear Bit**

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

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

0	Disable T0 overflow interrupt
1	Enable T0 overflow interrupt

.0 **Timer 0 Match/Capture Interrupt Enable Bit**

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

NOTES:
1. Both the Timer 0 overflow and the Timer 0 match/capture interrupts pending conditions are automatically cleared by hardware.
2. Do not use the Timer 0 overflow and the Timer 0 match/capture interrupts (IRQ0) at the same time. If you use them at the same time, it may occur a problem. (See Page 10-4)

T1CON — Timer 1 Control Register F5H Set 1, Bank 0

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

Addressing Mode Register addressing mode only

.7 – .5 Not used for S3C8618/C8615.

.4 **Timer 1 Counter Clear Bit**

0	No effect (when write)
1	*Clear timer 1 counter, T1CNT (when write)*

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

0	Disable T1 overflow interrupt
1	Enable T1 overflow interrupt

.2 **Timer 1 Interrupt Pending Flag**

0	No T1 interrupt pending (when read)
0	*Clear T1 interrupt pending condition (when write)*
1	T1 interrupt is pending (when read)
1	*No effect (when write)*

.1 and .0 **Timer 1 Input Clock Selection Bits**

0	0	CPU clock/4096
0	1	CPU clock/512
1	0	Hsync-I or Csync-I from sync processor
1	1	External clock input (T1CK)

T2CON — Timer 2 Control Register F7H Set 1, Bank 0

Bit Identifier	**.7**	**.6**	**.5**	**.4**	**.3**	**.2**	**.1**	**.0**
RESET Value	1	1	1	1	1	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 **Timer 1 Counter Clear Bit**

0	0	0	0	0	Non-divided
0	0	0	0	1	Divided by two
0	0	0	1	0	Divided by three
.....					Divided by four – thirty one
1	1	1	1	1	Divided by thirty two

.2 **Timer 2 Input Clock Source Selection Bit**

0	f_{OSC}/1024
1	f_{OSC} (non-divided)

.1 **Timer 2 Interrupt Enable Bit**

0	Disable interrupt
1	Enable interrupt

.0 **Timer 2 Interrupt Pending Flag**

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

5 INTERRUPT STRUCTURE

OVERVIEW

The SAM8 interrupt structure has three basic components: levels, vectors, and sources. The CPU recognizes 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. Each vector can have 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 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 S3C8618/C8615 interrupt structure recognizes seven interrupt levels, IRQ0–IRQ5.

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 will always be much smaller.) If an interrupt level has more than one vector address, the vector priorities are set in hardware. The S3C8618/C8615 have nine vectors — one corresponding to each of the nine possible interrupt sources.

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 S3C8618/C8615 interrupt structure, each source has its own vector address.

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

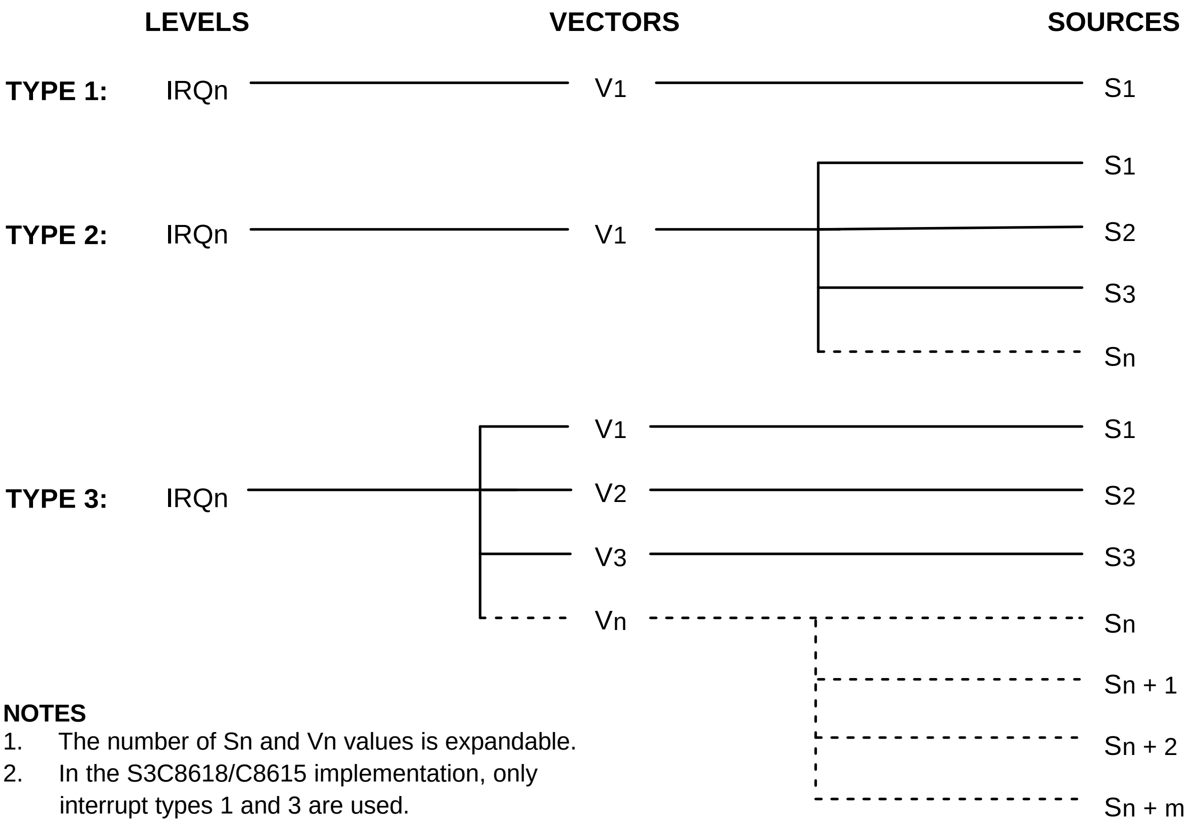
The three components of the SAM8 interrupt structure described above — levels, vectors, and sources — are combined to determine the interrupt structure of an individual device and to make full use of its available interrupt logic. There are three possible combinations of interrupt structure components, 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 S3C8618/C8615 microcontrollers, only interrupt types 1 and 3 are implemented.



NOTES
1. The number of Sn and Vn values is expandable.
2. In the S3C8618/C8615 implementation, only interrupt types 1 and 3 are used.

Figure 5-1. S3C8-Series Interrupt Types

S3C8618/C8615 INTERRUPT STRUCTURE

The S3C8618/C8615 microcontroller supports nine interrupt sources. Each interrupt source has a corresponding interrupt vector address. Six of the eight possible interrupt levels are used in the device-specific interrupt structure, which is 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.

LEVEL	VECTOR		SOURCE	RESET / CLEAR
IRQ0	FAH	0	Timer 0 overflow interrupt	H/W
	FCH	1	Timer 0 match/capture interrupt	H/W
IRQ1	EEH	0	Timer 2 match interrupt	S/W
	E6H	1	Timer 1 overflow interrupt	S/W
IRQ2	F8H		IIC-bus Tx/Rx interrupt	S/W
IRQ3	F2H		P0.0 external interrupt	S/W
IRQ4	F0H	0	P0.1 external interrupt	S/W
	FEH	1	P0.2 external interrupt	S/W
IRQ5	ECH		Vsync detection interrupt	H/W

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. S3C8618/C8615 Interrupt Structure

INTERRUPT VECTOR ADDRESSES

All interrupt vector addresses for the S3C8618/C8615 interrupt structure are stored in the vector address area of the ROM, 00H–FFH, (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.

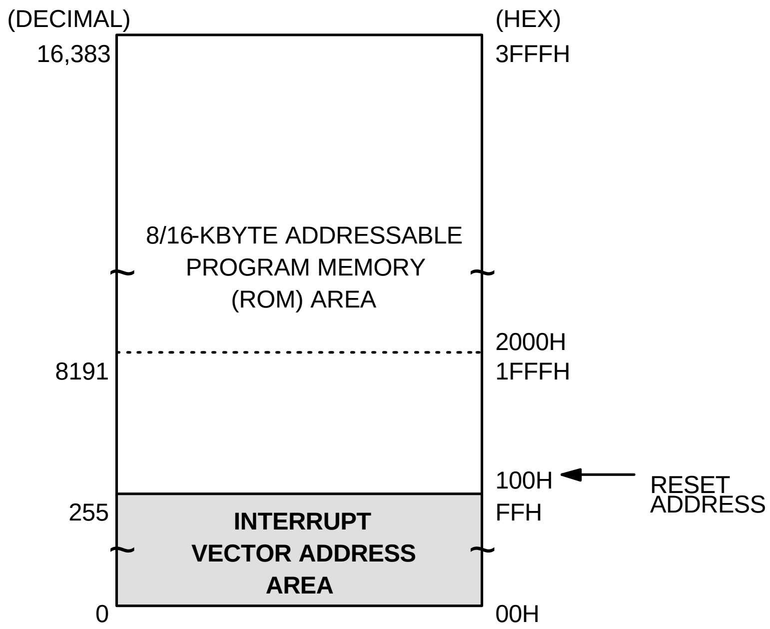


Figure 5-3. ROM Vector Address Area

Table 5-1. S3C8618/C8615 Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	**Hex Value**		**Interrupt Level**	**Priority in Level**	**H/W**	**S/W**
250	FAH	Timer 0 overflow interrupt	IRQ0	0	√	
252	FCH	Timer 0 match/capture interrupt		1	√	
238	EEH	Timer 2 match interrupt	IRQ1	0		√
246	F6H	Timer 1 overflow interrupt		1		
248	F8H	Multi-master IIC-bus Tx/Rx interrupt	IRQ2	–		√
242	F2H	P0.0 external interrupt	IRQ3	–		√
240	F0H	P0.1 external interrupt	IRQ4	0		√
254	FEH	P0.2 external interrupt		1		√
236	ECH	Vsync detection interrupt	IRQ5	–	√	

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. Do not use the Timer 0 overflow and the Timer 0 match/capture interrupts(IRQ0) at the same time. If you use them at the same time, it may occur a problem. (see Page 10-4)

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

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

NOTE

The system initialization routine that is executed following a reset must always contain an EI instruction (assuming one or more interrupts are used in the application).

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

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 six interrupt levels, IRQ0–IRQ5.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The six levels of the S3C8618/C8615 are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, group B is IRQ2–IRQ4, and group C is IRQ5.
Interrupt request register	IRQ	R	This register contains a request pending bit for each of the seven interrupt levels, IRQ0–IRQ5.
System mode register	SYM	R/W	Dynamic global interrupt processing enable and disable, fast interrupt processing.

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)

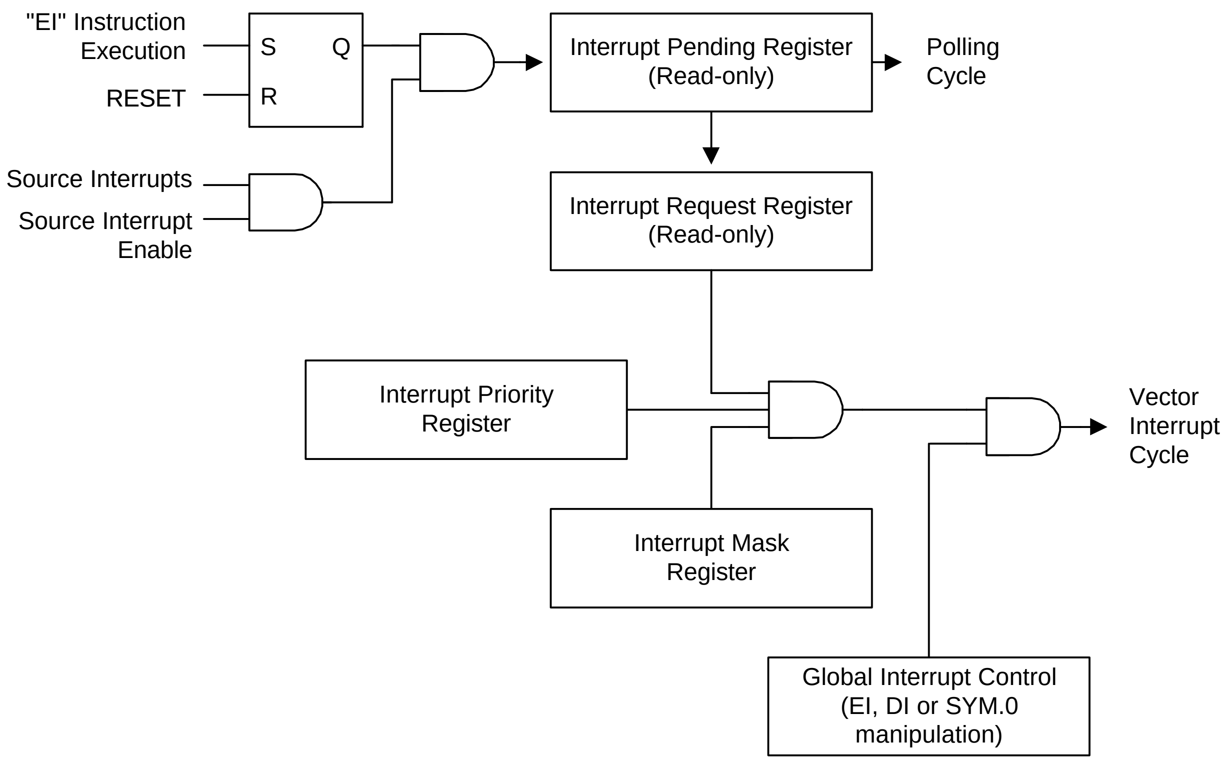
— Interrupt level enable/disable settings (IMR register)

— 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.



NOTE: In the S3C8618/C8615 microcontrollers, only six interrupt levels are recognized by CPU: IRQ0-IRQ5

Figure 5-4. Interrupt Function Diagram

PERIPHERAL INTERRUPT CONTROL REGISTERS

For each interrupt source there is a corresponding peripheral control register (or registers) to control the interrupts generated by that peripheral. These registers and their locations are listed in Table 5-3.

Table 5-3. Interrupt Source Control Registers

Interrupt Source	Interrupt Level	Control Register(s)	Register Location(s)
Timer 0 (match/capture) Timer 0 (overflow) interrupt	IRQ0	T0CON	Set 1, D2H
Timer 2 match interrupt Timer 1 overflow interrupt	IRQ1	T2CON T1CON	Set 1, bank 0, F7H Set 1, bank 0, F5H
IIC-bus Tx/Rx interrupt	IRQ2	ICCR ICSR	Set 1, bank 1, E9H Set 1, bank 1, EAH
P0.0 external interrupt	IRQ3	P0CONL, P0INT	Set 1, bank 0, E5H Set 1, bank 0, EBH
P0.1 external interrupt P0.2 external interrupt	IRQ4	P0CONL, P0INT	Set 1, bank 0, E5H Set 1, bank 0, EBH
Vsync detection interrupt	IRQ5	SYNCON2	Set 1, bank 0, F3H

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. Figure 5-5 shows the effect of the various control settings.

A reset clears SYM.7, SYM.1, and SYM.0 to "0" and the other SYM bit values (for fast interrupt level selection) are undetermined.

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

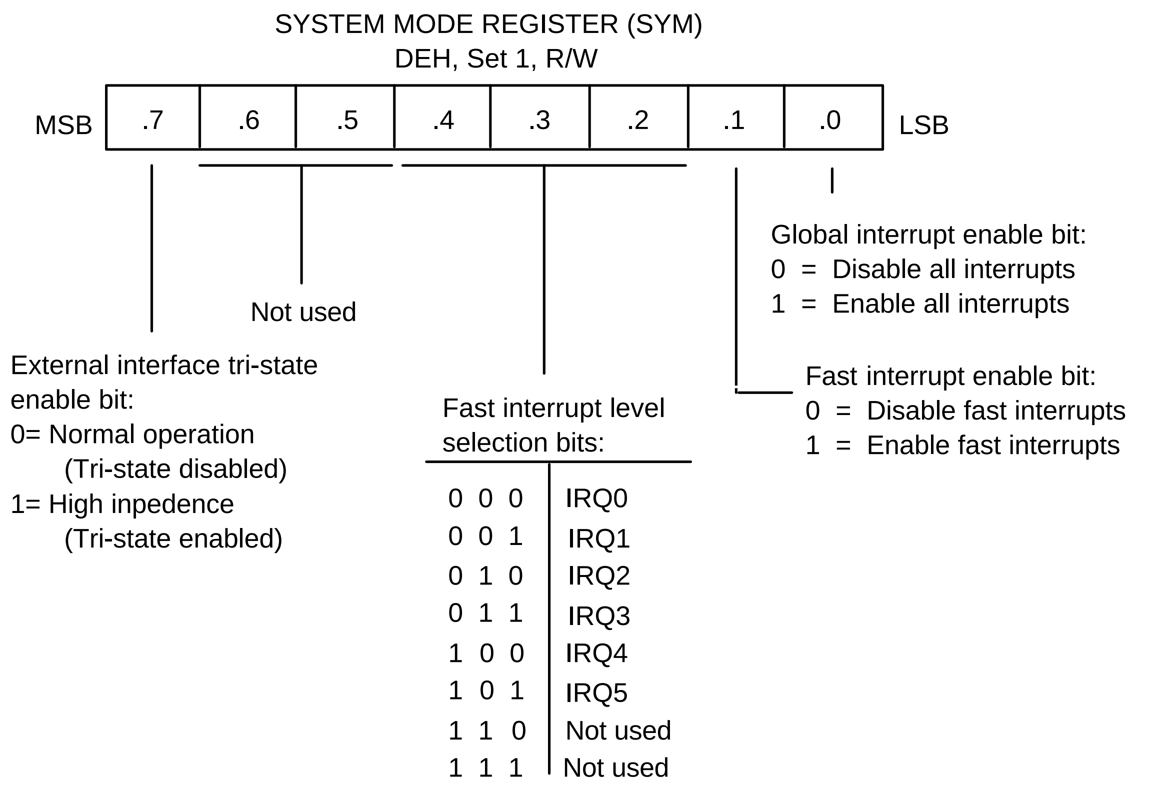


Figure 5-5. System Mode Register (SYM)

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 by the initialization routine.

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

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

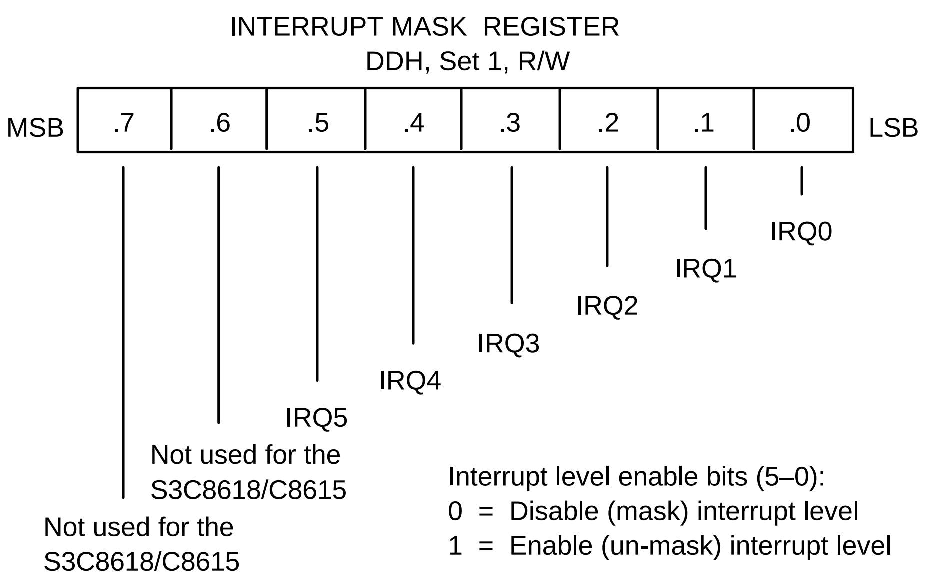


Figure 5-6. Interrupt Mask Register (IMR)

INTERRUPT PRIORITY REGISTER (IPR)

The interrupt priority register, IPR (set 1, bank 0, FFH), is used to set the relative priorities of the interrupt levels 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 by the initialization routine.

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	IRQ2, IRQ3, IRQ4
Group C	IRQ5

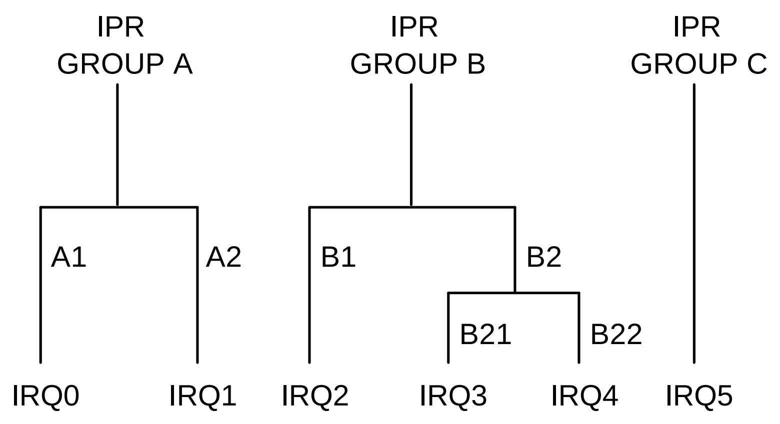


Figure 5-7. Interrupt Request Priority Groups

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.

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

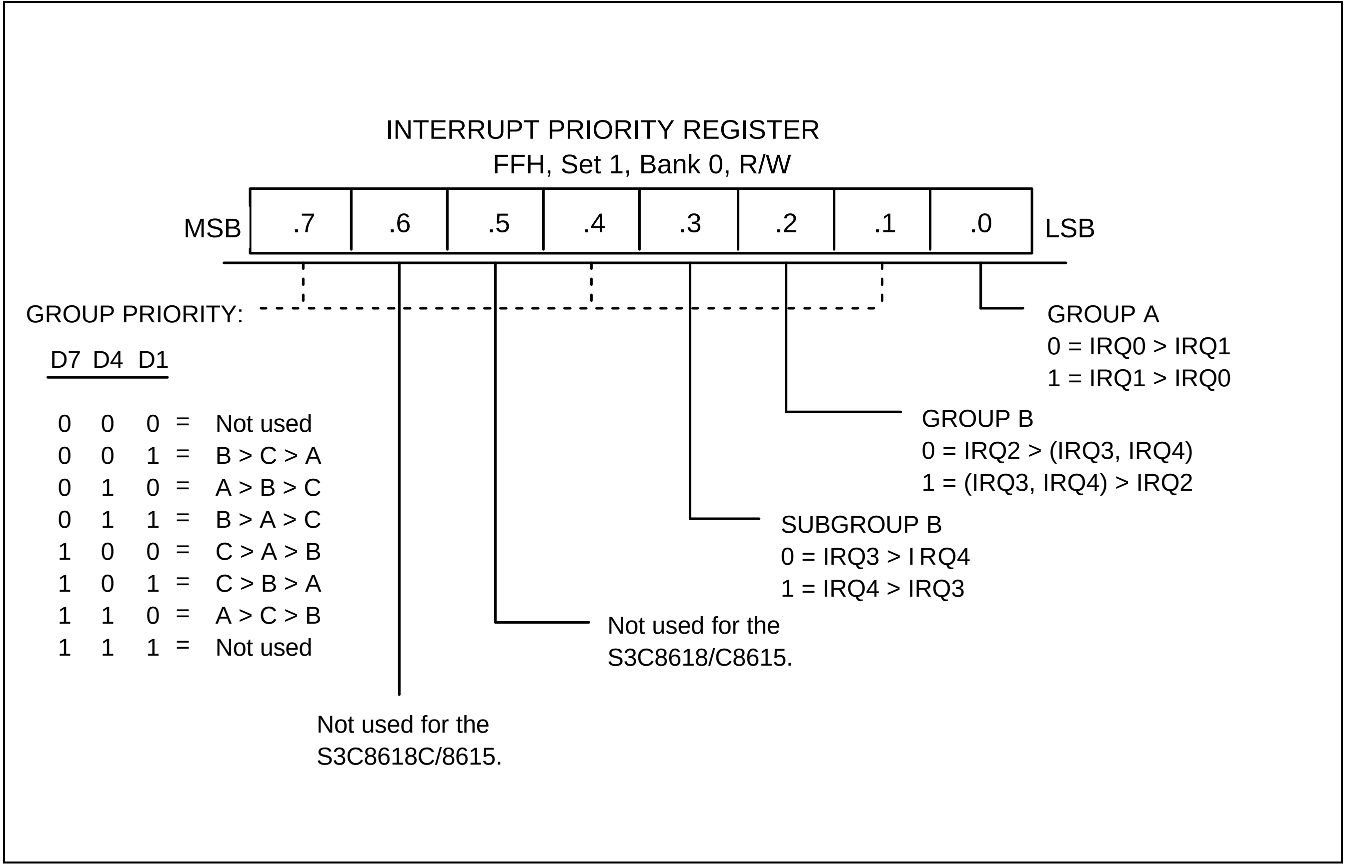


Figure 5-8. Interrupt Priority Register (IPR)

INTERRUPT REQUEST REGISTER (IRQ)

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

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

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

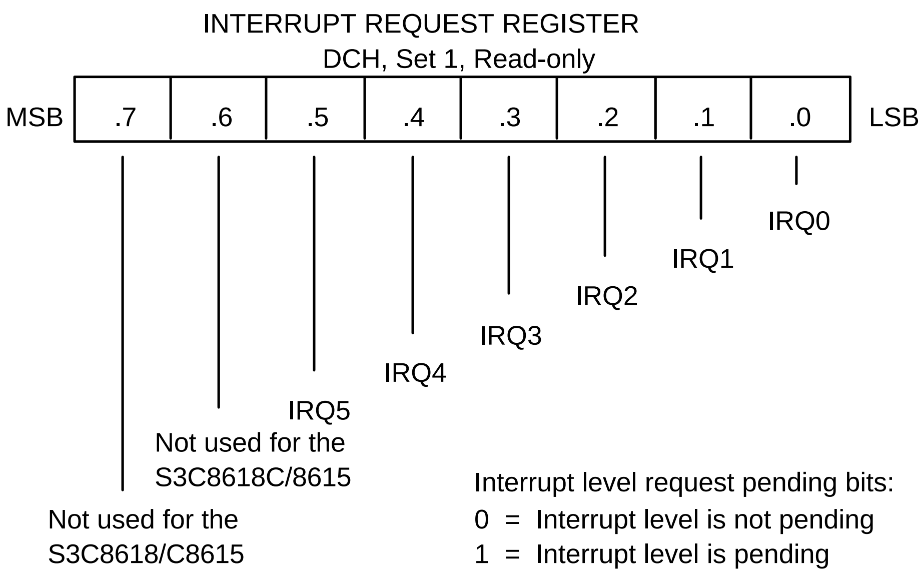


Figure 5-9. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

There are two types of interrupt pending bits: One type is automatically cleared by hardware after the interrupt service routine is acknowledged and executed; the other type must be cleared by the application program's 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, 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 S3C8618/C8615 interrupt structure, the timer 0 overflow interrupt (IRQ0, vector FAH), Timer 0 match or capture interrupt(IRQ0, vector FCH) and the Vsync detection interrupt (IRQ5, vector ECH) belong to this category of interrupts whose pending conditions are 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 S3C8618/C8615 interrupt structure, pending conditions for all interrupt sources *except* the timer 0 overflow interrupt and the Vsync detection interrupt must be cleared by the program software's 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 16 clock cycles. SYM.4–SYM.2 are used to select a specific interrupt level for fast processing and SYM.1 enables or disables fast interrupt processing.

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

1. For the S3C8618/C8615 microcontrollers, the service routine for any of the six interrupt levels (IRQ0–IRQ5) can be selected for fast interrupt processing.

2. If you want to use a fast interrupt in multi source interrupt vector, the fast interrupt may not be processed when you use two sources as interrupt vector in normal mode. But it is possible when you use only one source as interrupt vector.

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.

NOTE

If you use fast interrupts, remember to load the IP with a new start address when the fast interrupt service routine ends.

☞ PROGRAMMING TIP — Setting Up the Interrupt Control Structure

This example shows you how to enable interrupts for select interrupt sources, disable interrupts for other sources, and set interrupt priorities for the S3C8618/C8615 interrupt structure. The sample program does the following:

— Disables the watchdog function.

— Enables the following interrupts: P0.0 external interrupt, timer 0 match/capture, timer 1 overflow, timer 2 interrupt, and multi-master IIC-bus Tx/Rx interrupt.

— Disables the following interrupts: P0.1 and P0.2 external interrupts, and Vsync detection.

— Sets interrupt priorities as P0.0 > timer 2 > timer 0 > timer 1 > IIC-bus Tx/Rx.

```
        •
        •
        •
        DI                              ;  Disable interrupts globally
        LD      BTCON,#0A0H             ;  Disable watchdog function
        LD      P0CONL,#01H             ;  P0.0 ← enable rising edge interrupts
        LD      P0INT,#01H              ;  Enable P0.0 external interrupt
                                        ;  Disable P0.1 and P0.2 external interrupts
        LD      T0CON,#8FH              ;  Enable T0 match/capture interrupt
                                        ;  (capture on rising edges)
                                        ;  Enable T0 overflow interrupt
        LD      T1CON,#1AH              ;  Enable timer 1 overflow interrupt
        LD      T2CON,#0FFH             ;  Enable timer 2 match interrupt
        LD      T2DATA,#249             ;  Setting 1ms interval
        LD      ICCR,#34H               ;  Enable IIC-bus Tx/Rx interrupt, SCL clock=100kHz
        LD      SYNCON2,#00H            ;  Disable Vsync detection interrupt
        LD      IMR,#0FH                ;  Enable interrupt levels IRQ0, IRQ1, IRQ2, and IRQ3
        LD      IPR,#85                 ;  IRQ3 > IRQ1 > IRQ0 > IRQ2
                                        ;  (P0.0 > timer 0 > timer2 > timer 1 > IIC-bus)
        EI                              ;  Enable interrupts globally
        •
        •
        •
```

☞ PROGRAMMING TIP — Programming Level IRQ0 as a Fast Interrupt

This example shows you how to program fast interrupt processing for a select interrupt level — in this case, for the timer 0 match/capture interrupt:

```
            •
            •
            •
            LD      T0CON,#8FH        ;  Enable T0OVF interrupt
                                      ;  Enable T0 interrupt
                                      ;  Capture mode (on rising signal edges)
                                      ;  Select fOSC/8 as the T0 clock source
            LD      P0CONL,#01H       ;  Set P0.4 to capture input mode
            LDW     IPH,#T0_INT       ;  IPH ← high byte of interrupt service routine
                                      ;  IPL ← low byte of interrupt service routine
            LD      SYM,#02H          ;  Enable fast interrupt processing
                                      ;  Select IRQ0 for fast interrupt service
            EI                        ;  Enable interrupts
            •
            •
            •

FAST_RET:   IRET                      ;  IP ← Address of T0_INT (again)
T0_INT:
            •
            •
            •
            (Fast service routine executes)
            •
            •
            •
            LD      T0CON,#8FH        ;  Clear T0INT interrupt pending bit
            JP      T,FAST_RET
```

7 CLOCK CIRCUIT

OVERVIEW

The clock frequency generated for the S3C8618/C8615 by an external crystal can range from 6 MHz to 12 MHz. The maximum CPU clock frequency is 12 MHz. The X_{IN} and X_{OUT} pins connect the external oscillator or clock source to the on-chip clock circuit.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

— External crystal or ceramic resonator oscillation source (or an external clock source)

— Oscillator stop and wake-up functions

— Programmable frequency divider for the CPU clock (f_{OSC} divided by 1, 2, 8, or 16)

— System clock control register, CLKCON

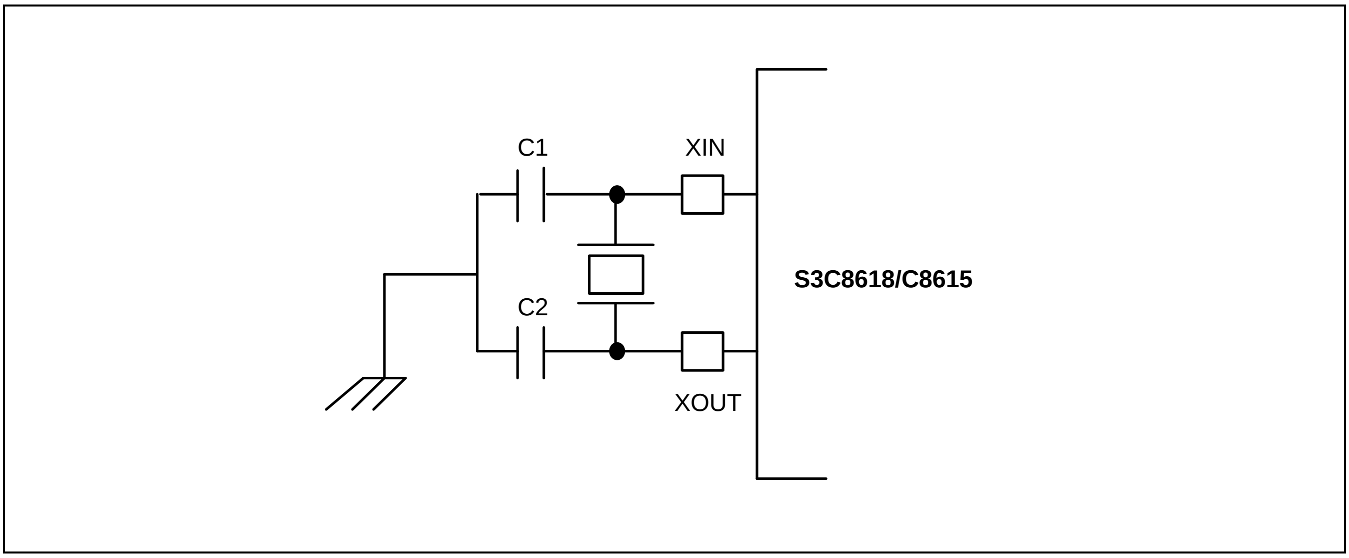


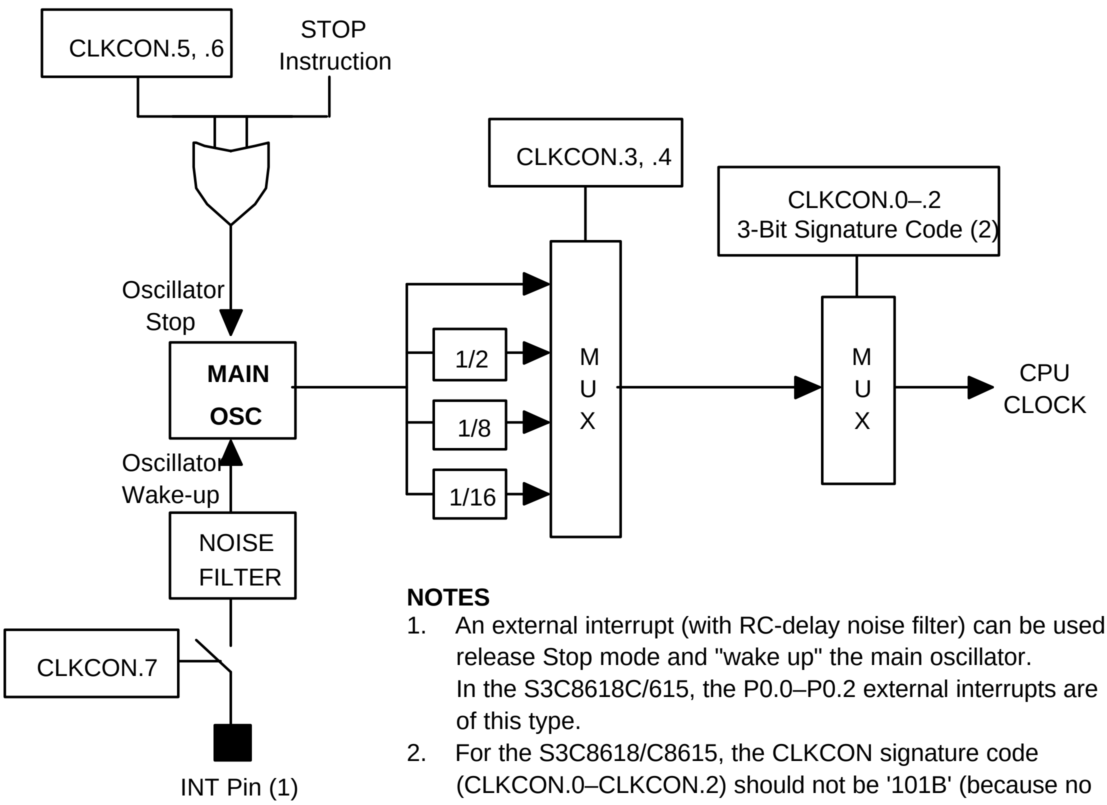
Figure 7-1. Main Oscillator Circuit (External Crystal or Ceramic Resonator)

CLOCK STATUS DURING POWER-DOWN MODES

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

— In Stop mode, the main oscillator is halted. Stop mode is released, and the oscillator started, by a reset operation or an external interrupt (with RC delay noise filter).

— In Idle mode, the internal clock signal is gated to the CPU, but not to interrupt structure, timers and timer/counters, and the IIC-bus interface functions. Idle mode is released by a reset or by an external or internal interrupt.



NOTES

1. An external interrupt (with RC-delay noise filter) can be used to release Stop mode and "wake up" the main oscillator.
   In the S3C8618C/615, the P0.0–P0.2 external interrupts are of this type.
2. For the S3C8618/C8615, the CLKCON signature code (CLKCON.0–CLKCON.2) should not be '101B' (because no subsystem clock is implemented).

Figure 7-2. 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

— Main oscillator stop control

— Oscillator frequency divide-by value

— System clock signal selection

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

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

NOTE

If you are using an external oscillator with a frequency higher than 12 MHz, you cannot use a non-divided clock (f_{OSC}) as the CPU clock (see Figure 7-3).

For the S3C8618/C8615 microcontrollers, the CLKCON.2–CLKCON.0 system clock signature code must be any value *other than* '101B'. (The ‘101B’ setting is invalid because a subsystem clock is not implemented.) The reset value for the clock signature code is '000B' and should remain so during normal operation.

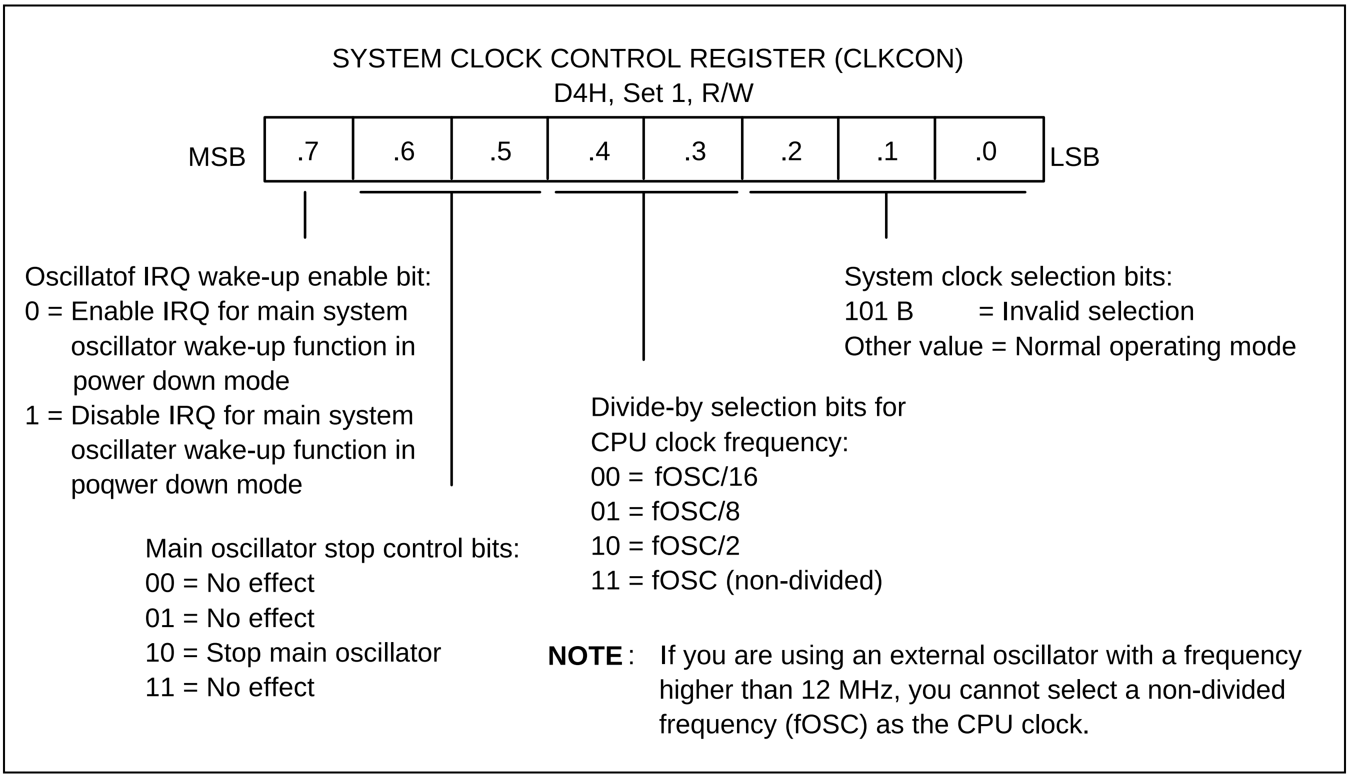


Figure 7-3. System Clock Control Register (CLKCON)

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 S3C8618/C8615 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 oscillation stabilization time for a reset operation 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 and the reset operation starts. All system and peripheral control registers are then reset to their default hardware values (see Tables 8-1, 8-2, and 8-3).

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

- All interrupts are disabled.
- The watchdog function (basic timer) is enabled.
- Ports 0–4 are set to input mode.
- Peripheral control and data registers are disabled and reset to their default hardware values.
- The program counter (PC) is loaded with the program reset address in the ROM, 0100H.
- When the programmed oscillation stabilization time interval has elapsed, the instruction stored in ROM location 0100H (and 0101H) is fetched and executed.

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

— A "1" or a "0" shows the reset bit value as logic one or logic zero, respectively.

— An 'x' means that the bit value is undefined after a reset.

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

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 register	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
Interrupt mask register	IMR	221	DDH	–	–	x	x	x	x	x	x
System mode register	SYM	222	DEH	0	–	–	x	x	x	0	0
Register page pointer	PP	223	DFH	0	0	0	0	0	0	0	0

NOTES:

1. Although the SYM register is not used for the S3C8618/C8615, 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 and IRQ, 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. The read-only registers in the S3C8618/C8615 register file are: T0CNT, IRQ, T1CNT, BTCNT, and PWMCNT.
4. Interrupt pending flags that must be cleared by software are noted by shaded table cells.

Table 8-2. Set 1, Bank 0 Register Values After Reset

Register Name	Mnemonic	Address		Bit Values After Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
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
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
Port 0 control register (high byte)	P0CONH	230	E4H	0	0	0	0	0	0	0	0
Port 0 control register (low byte)	P0CONL	231	E5H	0	0	0	0	0	0	0	0
Port 1 control register	P1CON	232	E6H	0	0	0	0	0	0	0	0
Port 2 control register (high byte)	P2CONH	233	E7H	0	0	0	0	0	0	0	0
Port 2 control register (low byte)	P2CONL	234	E8H	0	0	0	0	0	0	0	0
Port 3 control register (high byte)	P3CONH	235	E9H	0	0	0	0	0	0	0	0
Port 3 control register (low byte)	P3CONL	236	EAH	0	0	0	0	0	0	0	0
Port 0 external interrupt control register	P0INT	239	EBH	–	0	0	0	–	0	0	0
Locations ECH–EFH are not mapped											
DDC control register	DDCCON	240	F0H	–	–	–	–	–	–	0	0
Sync control register 0	SYNCON0	241	F1H	0	0	0	0	0	0	0	0
Sync control register 1	SYNCON1	242	F2H	0	0	0	0	0	0	0	0
Sync control register 2	SYNCON2	243	F3H	0	0	0	0	0	0	0	0
Sync port read data register	SYNCRD	244	F4H	–	–	–	–	x	x	x	x
Timer 1 control register	T1CON	245	F5H	–	–	–	0	0	0	0	0
Timer 1 counter register	T1CNT	246	F6H	0	0	0	0	0	0	0	0
Timer 2 control register	T2CON	247	F7H	1	1	1	1	1	0	0	0
Timer 2 counter register	T2CNT	248	F8H	1	1	1	1	1	1	1	1
Locations F9H–FCH are not mapped											
Basic timer counter register	BTCNT	253	FDH	0	0	0	0	0	0	0	0
External memory timing register	EMT	254	FEH	0	1	1	1	1	1	0	–
Interrupt priority register	IPR	255	FFH	x	x	x	x	x	x	x	x

NOTES:
1. Except for IRQ, SYNCRD, T1CNT and BTCNT, all registers in set 1, bank 0 are read/write addressable.
2. You cannot use a read-only register as a destination field for the instructions OR, AND, LD, and LDB. The read-only registers in the S3C8618/C8615 register file are: SYNCRD, IRQ, T1CNT, BTCNT, and PWMCNT.
3. Interrupt pending flags that must be cleared by software are noted by shaded table cells.

Table 8-3. Set 1, Bank 1 Register Values After Reset

Register Name	Mnemonic	Address		Bit Values After Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
PWM 0 data register	PWM0	224	E0H	0	0	0	0	0	0	0	0
PWM 1 data register	PWM1	225	E1H	0	0	0	0	0	0	0	0
PWM 2 data register	PWM2	226	E2H	0	0	0	0	0	0	0	0
PWM 3 data register	PWM3	227	E3H	0	0	0	0	0	0	0	0
PWM 4 data register	PWM4	228	E4H	0	0	0	0	0	0	0	0
PWM 5 data register	PWM5	229	E5H	0	0	0	0	0	0	0	0
PWM 6 data register	PWM6	230	E6H	0	0	0	0	0	0	0	0
PWM control register	PWMCON	231	E7H	0	0	0	–	–	–	–	–
PWM counter register	PWMCNT	232	E8H	0	0	0	0	0	0	0	0
IIC-bus clock control register	ICCR	233	E9H	0	0	0	0	1	1	1	1
IIC-bus control/status register	ICSR	234	EAH	0	0	0	0	0	0	0	0
IIC-bus address register	IAR	235	EBH	x	x	x	x	x	x	x	–
IIC-bus Tx/Rx data shift register	IDSR	236	ECH	x	x	x	x	x	x	x	x
Locations EDH–FFH are not mapped.											

NOTES:
1. You cannot use a read-only register as a destination field for the instructions OR, AND, LD, and LDB. The read-only registers in the S3C8618/C8615 register file are: SYNCRD, IRQ, T1CNT, BTCNT, and PWMCNT.
2. Interrupt pending flags that must be cleared by software are noted by shaded table cells.
3. Bits 3–0 of the IIC-bus control register, ICCR, are a 4-bit prescaler value. The other ICCR bits have control functions.

POWER-DOWN MODES

STOP MODE

Stop mode is invoked by the instruction STOP (opcode 7FH). In Stop mode, the operation of the CPU and all peripherals is halted. That is, the on-chip main oscillator stops and the supply current is reduced to less than 5 µA. All system functions stop when the clock "freezes," but data stored in the internal register file is retained. Stop mode can be released in one of two ways: by a reset or by an external interrupt (with RC delay).

NOTE

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

Using RESET to Release Stop Mode

Stop mode is released when the RESET signal goes active (High level): all system and peripheral control registers are reset to their default hardware values and the contents of all data registers are retained. A reset operation automatically selects a slow clock (1/16) because CLKCON.3 and CLKCON.4 are cleared to '00B'. After the programmed oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in ROM location 0100H (and 0101H).

Using an External Interrupt to Release Stop Mode

Only external interrupts with an RC-delay noise filter circuit can be used to release Stop mode. Which interrupt you can use to release Stop mode in a given situation depends on the microcontroller's current internal operating mode. The external interrupts in the S3C8618/C8615 interrupt structure that can be used to release Stop mode are:

— External interrupts P0.0 (INT0), P0.1 (INT1), and P0.2 (INT2)

— Timer 0 match/capture interrupt in capture mode (with rising or falling edge trigger at the T0CAP pin)

Please note the following conditions for Stop mode release:

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

— If you use an external interrupt for Stop mode release, you can also program the duration of the oscillation stabilization interval. To do this, you must make the appropriate control and clock settings *before* entering Stop mode.

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

— The external interrupt is serviced when the Stop mode release occurs. Following the IRET from the service routine, the instruction immediately following the one that initiated Stop mode is executed.

IDLE MODE

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

There are two ways to release Idle mode:

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

NOTE

Only external interrupts can be used to release Stop mode. To release Idle mode, you can use either an internally-generate or externally-generated interrupt.

☞ PROGRAMMING TIP — Sample S3C8618/C8615 Initialization Routine

The following sample program shows you how to make initial settings for the S3C8618/C8615 address space, interrupt vectors, and peripheral functions. Program comments guide you through the steps:

```
;           << Base Number Setting >>

            DECIMAL

;           << Definition >>

T0_REG      EQU         40H
            ORG         0000H

;           << Interrupt Vector Addresses >>

            ORG         00ECH
            VECTOR      VSYNC_INT           ;   IRQ5
            VECTOR      T2_MAT_INT          ;   IRQ1
;
            VECTOR      P01_INT             ;   IRQ4
            VECTOR      P00_INT             ;   IRQ3
            ORG         00F6H
            VECTOR      T1_OVF_INT          ;   IRQ1
            VECTOR      IIC_INT             ;   IRQ2
            VECTOR      T0_OVF_INT          ;   IRQ0
            VECTOR      T0_CAP_MAT_INT      ;   IRQ0
            VECTOR      P02_INT             ;   IRQ4

;           << Initialize System and Peripherals >>

            ORG         0100H               ;   Reset address
            LD          BTCON,#0A0H         ;   Disable watchdog timer
            LD          CLKCON,#10H         ;   Select divided-by-two oscillator frequency as CPU clock
                                            ;   Enable IRQ for main system oscillator wake-up

;           < System Register Settings >

            CLR         SYM                 ;   Disable fast interrupts; global interrupt disable
            CLR         EMT                 ;   No access wait time; select internal stack area
            LD          SPH,#00H            ;   Set stack pointer (stack starts from #0FFH)

;           < Interrupt Settings >

            LD          IPR,#85H            ;   Set interrupt priorities as follows:
                                            ;   IRQ3 > IRQ1 > IRQ0 > IRQ2
            LD          IMR,#0FH            ;   Enable IRQ levels 0, 1, 2, and 3

;           < Timer 0 Settings >

            LD          T0CON,#8FH          ;   Capture mode
```

(Continued on next page)

☞ PROGRAMMING TIP — Sample S3C8618/C8615 Initialization Routine (Contnued)

```
INI_PERI_SET:
          SB0                                ;  Select bank 0
          LD         P0,CONH,#0FFH           ;  Set port 0 high byte to push-pull output mode
          LD         P0CONL,#0FDH            ;  Set port 0 low byte to push-pull output mode
          LD         P0INT,#11H              ;  Enable P0.0 external interrupt
                                             ;  Disable P0.1 and P0.2 external interrupts
          LD         P1CON,#00H              ;  Set P1.0–P1.3 to input mode
          LD         P2CONH,#0FFH            ;  Set port 2 high byte to n-channel, open-drain PWM
                                             ;  output mode
          LD         P2CONL,#0FFH            ;  Set port 2 low byte to push-pull PWM output mode
          LD         P3CONH,#55H             ;  Set port 3 high byte to input mode with pull-up resistor
          LD         P3CONL,#55H             ;  Set port 3 low byte to input mode with pull-up resistor

;         < Timer 1 Settings >

          LD         T1CON,#1AH              ;  Enable timer 1 overflow interrupt, Timer 1 clock source
                                                is HsyncI from sync processor

;         < Sync Processor Settings >

          LD         SYNCON0,#40H            ;  5 bit counter capture mode
          LD         SYNCON1,#50H            ;  Set negative polarity (500ns at 8MHz) for clampO
          LD         SYNCON2,#8CH            ;  Pseudo sync output

;         < PWM Settings >

          SB1
          LD         PWMCON,#20H             ;  Start PWM counter, PWM counter clock is Fosc
          SB0

;         < DDC and IIC-Bus Tx/Rx Interface Settings >

          LD         DDCCON,#00H             ;  Select normal IIC-bus mode
          SB1                                ;  Select bank 1
          LD         ICCR,#34                ;  Enable IIC-bus interrupt, SCL clock is 100kHz

;         << Initialize Data Registers >>

          SB0                                ;  Select bank 0
          SRP        #0C0H                   ;  Set register pointer

;         < Clear all data registers from 00H to FFH >

          LD         R0,#0FFH

RAMCLR:   CLR        @R0
          DJNZ       R0,RAMCLR
```

(Continued on next page)

☞ PROGRAMMING TIP — Sample S3C8618/C8615 Initialization Routine (Continued)

```
;               < Initialize Other Registers >

                •
                •
                EI                              ; You must execute an EI instruction in this position
                                                ; in the initialization routine to enable servicing of
                                                ; external interrupts

;               << Main Loop >>

MAIN:           NOP                             ; Start main loop
                •
                •
                CALL        KEY_SCAN            ; Sub-program module
                •
                •
                CALL        LED_DISPLAY         ; Sub-program module
                •
                •
                CALL        JOB                 ; Sub-program module
                •
                •
                JR          t,MAIN              ; For main loop

;               < Subroutines >

KEY_SCAN:
                NOP
                •
                •
                •
                RET

LED_DISPLAY:
                NOP
                •
                •
                •
                RET

JOB:
                NOP
                •
                •
                •
                RET
```

(Continued on next page)

☞ PROGRAMMING TIP — Sample S3C8618/C8615 Initialization Routine (Continued)

```
;               << Interrupt Service Routines >>

P00_INT:
                PUSH    RP0                     ;  Save old RP0 value
                SRP0    #60H                    ;  Set RP0 for P0.0 interrupt service routine
                •
                •
                •
                POP     RP0                     ;  Restore the RP0 value
                IRET                            ;  Return from the interrupt

IIC_INT:
                PUSH    RP0                     ;  Save old RP0 value
                SRP0    #50H                    ;  Set RP0 for IIC-bus interrupt service routine
                •
                •
                •
                SB1
                LD      ICCR, #34H              ;  Clear IIC-bus interrupt pending bit
                SB0
                POP     RP0                     ;  Restore the RP0 value
                IRET                            ;  Return from the nterrupt

T0_INT:
                PUSH    RP0                     ;  Save old RP0 value
                SRP     #T0_REG                 ;  T0_REG value should be defined
                •
                •
                •
                POP     RP0                     ;  Restore the RP0 value
                IRET                            ;  Return from the interrupt

                END
```

9 I/O PORTS

OVERVIEW

The S3C8618/C8615 microcontrollers have four I/O ports with a total of 28 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 an overview of port functions:

Table 9-1. S3C8618/C8615 Port Configuration Overview

Port	Configuration Options	Programmability
0	8-bit general I/O port. Alternatively used for external interrupt inputs and for timer 0 and 1 input and output functions.	Bit programmable
1	8-bit I/O port for normal I/O	Bit programmable
2	8-bit I/O port for normal I/O or as PWM push-pull outputs or PWM n-channel open-drain outputs with 5-volt load capability.	Bit programmable
3	8-bit general I/O port. Alternatively used as n-channel open-drain or push-pull outputs with 5-volt load capability or for normal input with pull-up resistor.	Bit programmable

PORT DATA REGISTERS

Data registers for ports 0–3 have the format shown in Figure 9-1. Table 9-2 gives you an overview of the port data register locations:

Table 9-2. Port Data Register Summary

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

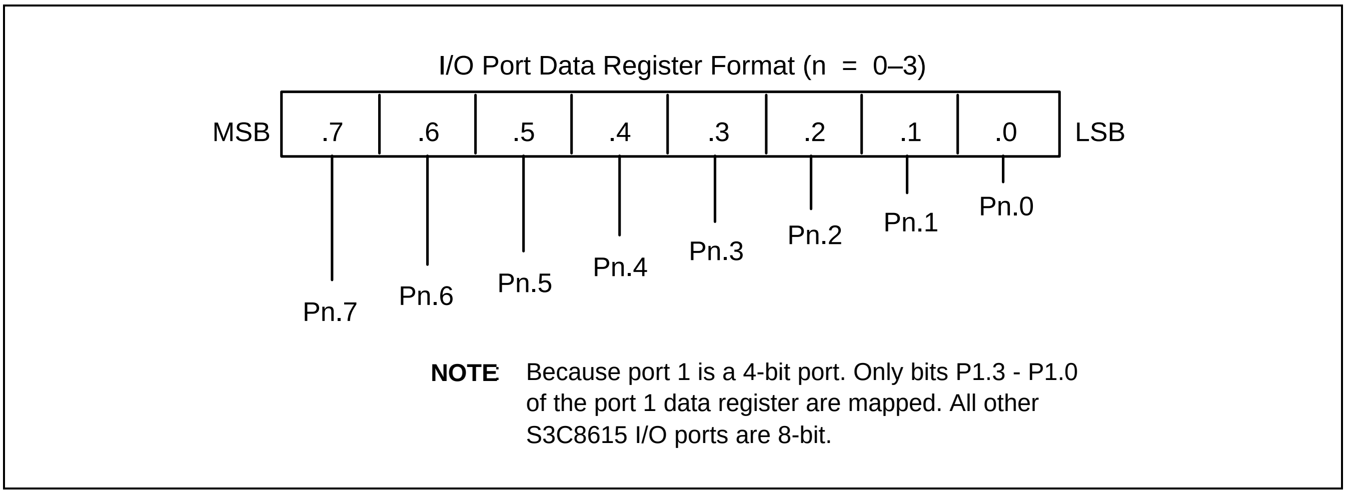


Figure 9-1. Port Data Register Format

PORT 0

Port 0 is an 8-bit I/O port with individually configurable pins. You access port 0 pins directly by writing or reading the port 0 data register, P0 (set 1, bank 0, E0H). You can use port 0 for general I/O, or for the following alternative functions:

— Low-byte pins (P0.3–P0.0) can be configured as push-pull outputs. In addition, P0.2–P0.0 as a multiplexed input pins for external interrupts INT2–INT0 with rising or falling edge detection.

— High-byte pins (P0.7–P0.4) can be configured as multiplexed inputs and push-pull outputs. In addition, P0.4 can serve as the timer 0 capture input pin (T0CAP), P0.5 as the timer 1 clock input pin (T1CLK).

Two 8-bit control registers are used to configure port 0 pins: P0CONH (set 1, bank 0, E6H) for P0.7–P0.4 and P0CONL (set 1, bank 0, E7H) for P0.3–P0.0. Each byte contains four bit-pairs; each bit-pair configures one pin. The low-byte port 0 control register, P0CONL, is also used to enable and disable the external interrupts, INT2–INT0, at pins P0.2–P0.0, respectively.

Port 0 High-Byte Control Register (P0CONH)

The four bit-pairs in the port 0 high-byte control register, P0CONH, have the following functions:

— To configure individual port 0 pins to multiplexed input mode or push-pull output mode.

— To configure alternative input or output functions for P0.7–P0.4.

Bit-pair 1/0 configures the capture signal input pin for timer 0 at P0.4. Bit-pair 3/2 controls P0.5; when it is set to ‘01B’, the timer 1 clock input is enabled.

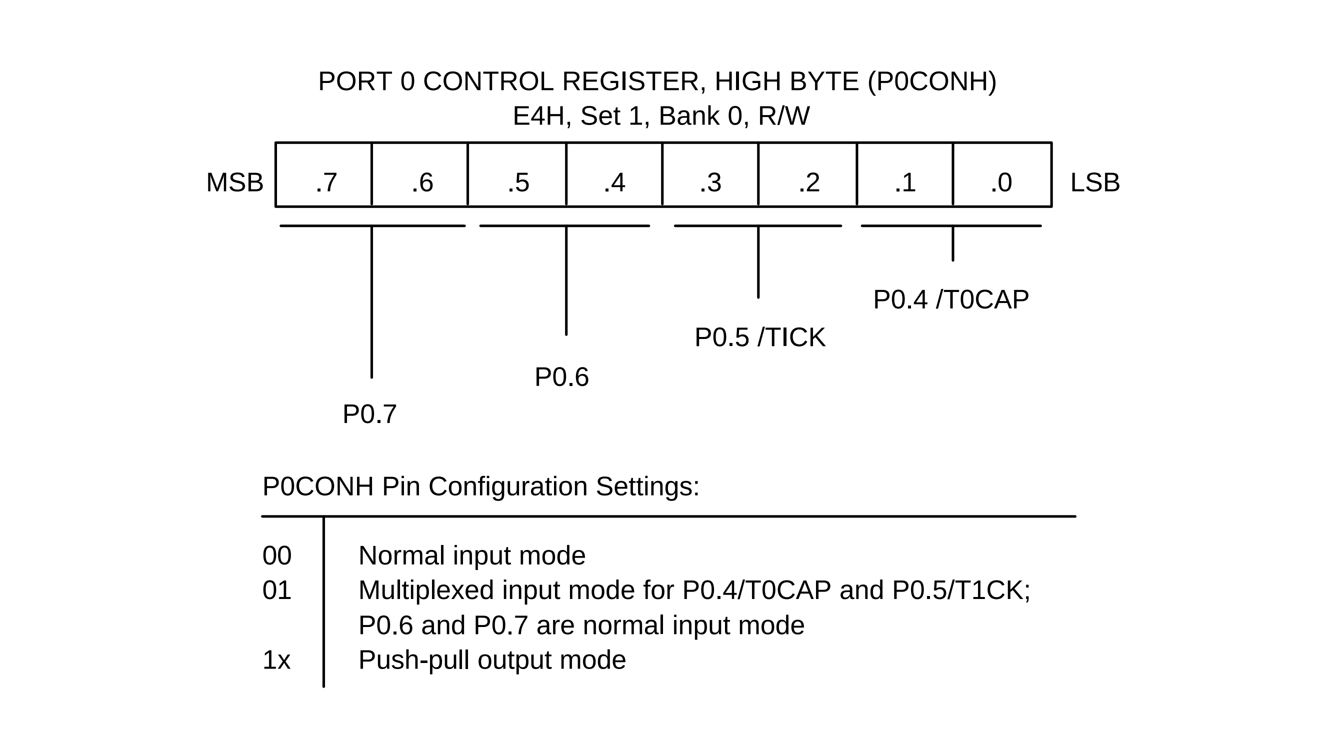


Figure 9-2. Port 0 High-Byte Control Register (P0CONH)

Port 0 Low-Byte Control Register (P0CONL)

The low-byte port 0 pins, P0.3–P0.0 can be configured individually as inputs or as push-pull outputs. You can alternatively configure pins P0.2–P0.0 as external interrupt inputs with rising or falling edge detection.

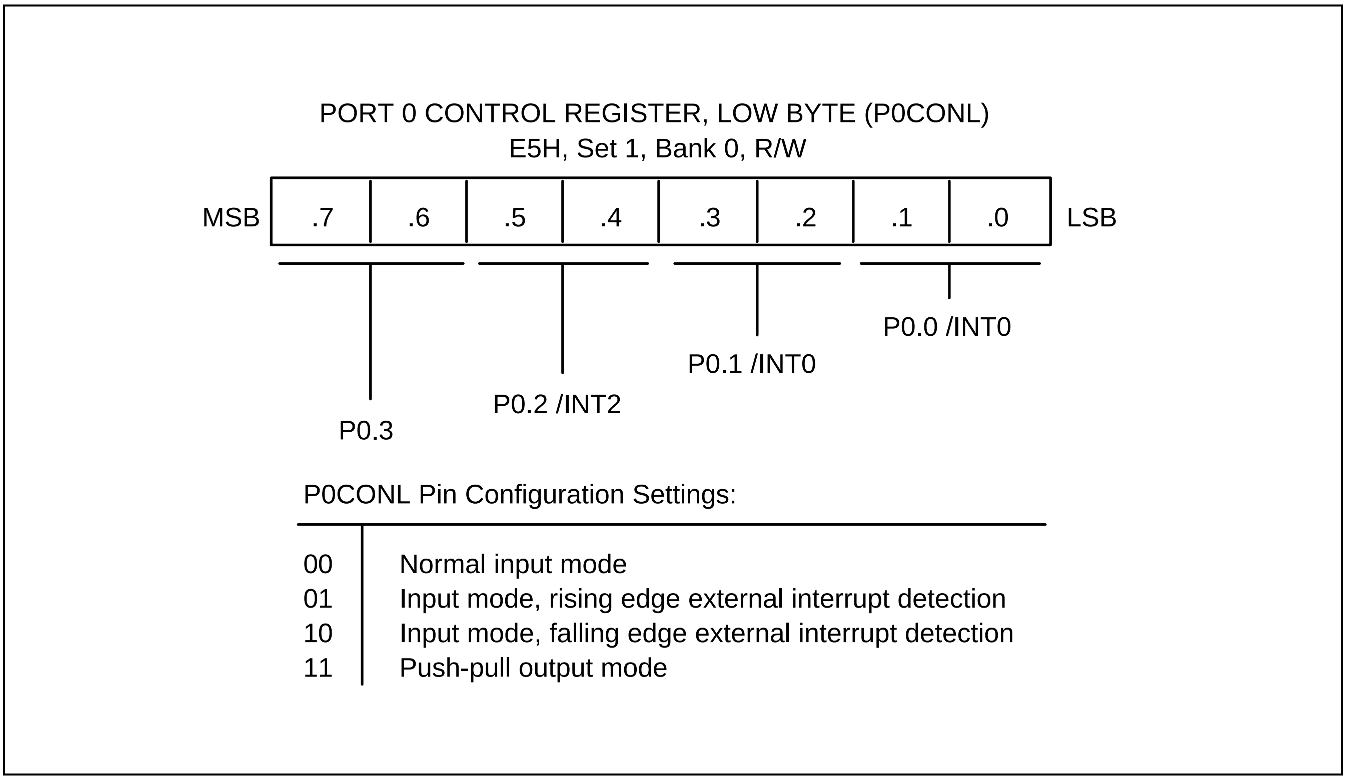


Figure 9-3. Port 0 Low-Byte Control Register (P0CONL)

Port 0 External Interrupt Control Register (P0INT)

The port 0 external interrupt control register, P0INT, is used to enable and disable the external interrupts INT2–INT0 at P0.2–P0.0, respectively, and also to detect and clear external interrupt pending conditions at these pins.

To selectively enable external interrupts INT0, INT1, and INT2, you set P0INT.0, P0INT.1, and P0INT.2 to "1", respectively. The application program can poll the corresponding interrupt pending bits — P0INT.4 for INT0, P0INT.5 for INT1, and P0INT.6 for INT2 — to detect external interrupt pending conditions.

After an external interrupt has been serviced, the service routine must clear the pending condition by writing a "0" to the appropriate pending bit. Writing a "1" to the pending bit has no effect.

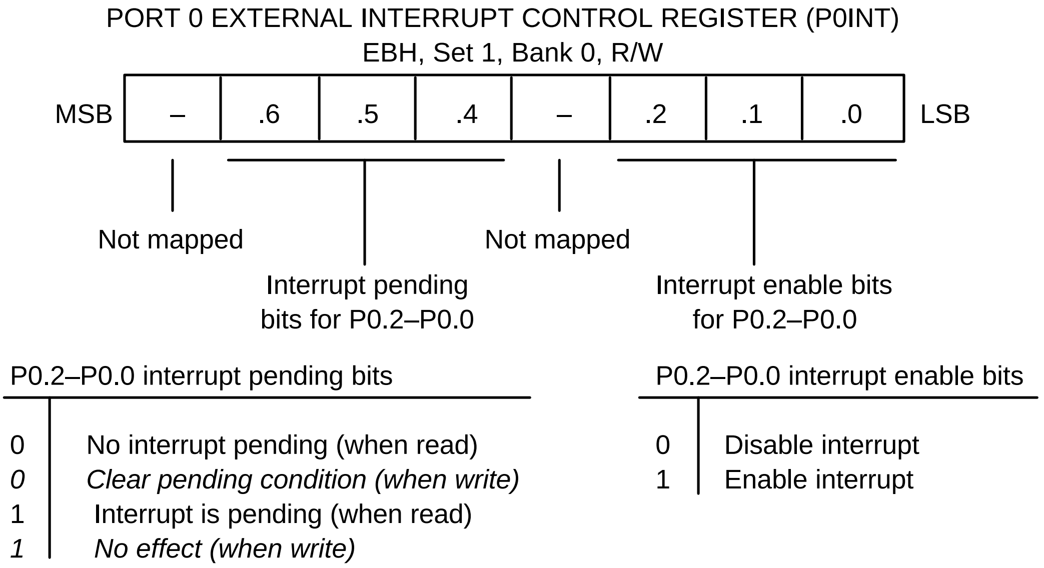


Figure 9-4. Port 0 External Interrupt Control Register (P0INT)

PORT 1

Port 1 is an 4-bit port with individually configurable pins. You access it directly by writing or reading the port 1 data register, P1 (set 1, bank 0, E1H). You can use port 1 for normal output or normal input mode.

The port 1 control register, P1CON (set 1, bank 0, E6H) is used to configure port 1 pins. Each byte contains four bit-pairs; each bit-pair configures one pin.

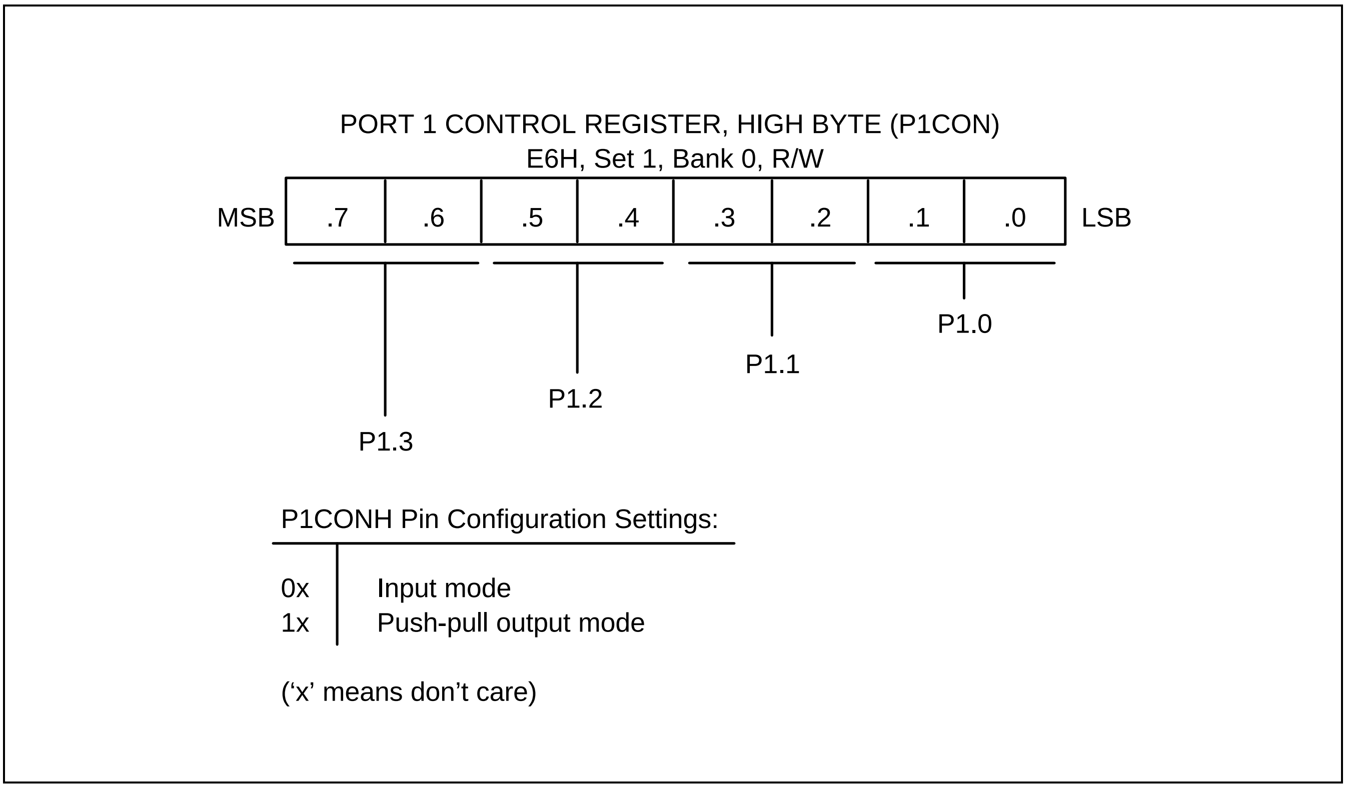
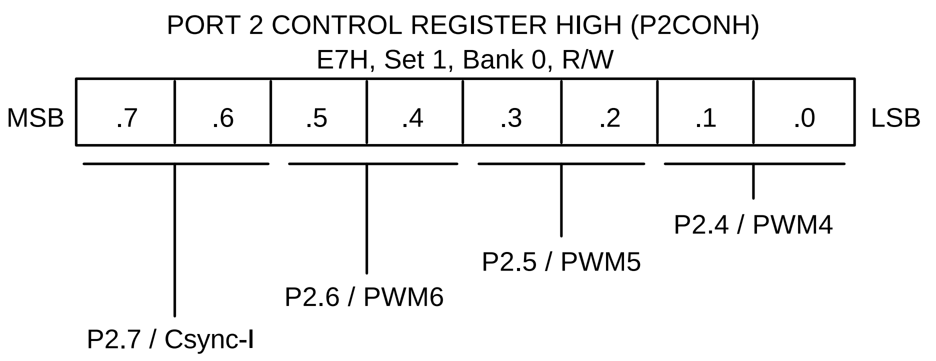


Figure 9-5. Port 1 Control Register (P1CON)

PORT 2

Port 2 is a 8-bit I/O port with individually configurable pins. You access port 2 pins directly by writing or reading the port 2 data register, P2 (set 1, bank 0, E2H).

Two 8-bit control registers are used to configure port 2 pins: P2CONH (set 1, bank 0, E7H) and P2CONL (set 1, bank 0, E8H) lets you select input mode and normal or PWM push-pull output mode or PWM n-channel open drain output mode.

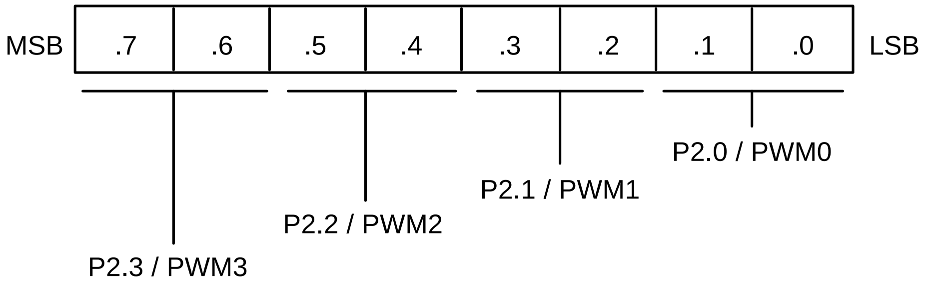


P2CONH Pin Configuration Settings:

bit 5–0	00	Input mode
	01	Push-pull output mode
	10	Push-pull PWM output mode (5-volt capability)
	11	N-channel open drain PWM output mode (5-volt capability)
bit 7, 6	0x	TTL input mode (Csync-I)
	1x	Push-pull output mode

Figure 9-6. Port 2 Control Register (P2CONH)

PORT 2 CONTROL REGISTER LOW (P2CONL)
E8H, Set 1, Bank 0, R/W



P2CONL Pin Configuration Settings:

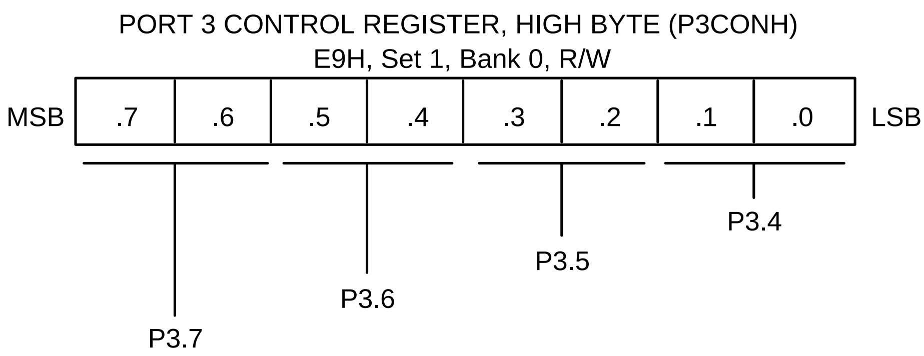
0x	Input mode
10	Push-pull output mode
11	Push-pull PWM output mode (up to 5-volt load capability)

Figure 9-7. Port 2 Control Register Low (P2CONL)

PORT 3

Port 3 is an 8-bit I/O port with individually configurable pins. You access it directly by writing or reading the port 3 data register, P3 (set 1, bank 0, E3H). You can use port 3 pins for n-channel, open-drain or push-pull outputs or you can configure them as normal inputs with or without pull-up resistor. The pin circuit used for the port 3 pins can withstand current loads up to 5 V.

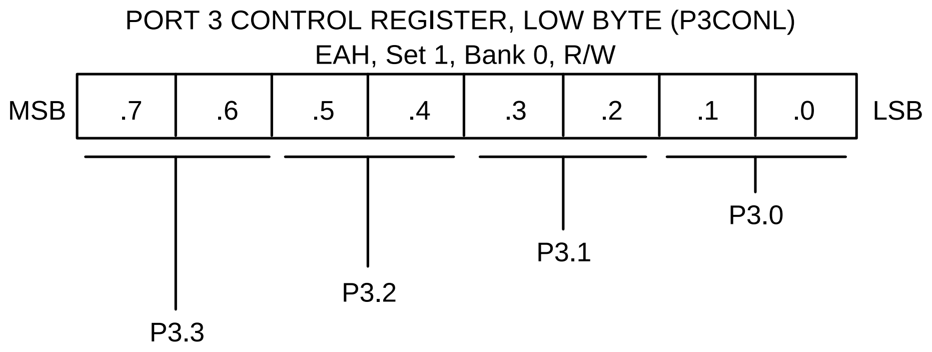
Two 8-bit control registers are used to configure port 3 pins: P3CONH (E9H, set 1, bank 0) for P3.7–P3.4 and P3CONL (set 1, bank 0, EAH) for P3.3–P3.0. Each byte contains four bit-pairs; each bit-pair configures one pin.



P3CONH Pin Configuration Settings:

00	Input mode
01	Input mode with pull-up resistor
10	Push-pull output mode
11	N-channel open-drain output mode (5-volt capability)

Figure 9-8. Port 3 High-Byte Control Register (P3CONH)



P3CONL Pin Configuration Settings:

00	Input mode
01	Input mode with pull-up resistor
10	Push-pull output mode
11	N-channel open-drain output mode (5-volt capability)

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

FUNCTION-FIXED PORT

This I/O pins are only used for the input and output of video synchronization signals to the sync processor or IIC-bus interface. Also, the horizontal and vertical sync signals may be monitored directly through the Sync Port Read Data Register (SYNCRD).

Sync signal ports

- Csync-I: Composite (SOG) synchronization input port (TTL level)
- Hsync-I: Horizontal synchronization input (TTL level)
- Vsync-I: Vertical and synchro clock for DDC
- Hsync-O: Horizontal synchronization output from the sync processor
- Vsync-O: Vertical synchronization output from the sync processor
- Clamp-O: Clamp signal output with programmable width from the sync processor

IIC-bus interface ports

- SDA: IIC-bus interface serial data
- SCL: IIC-bus interface serial clock

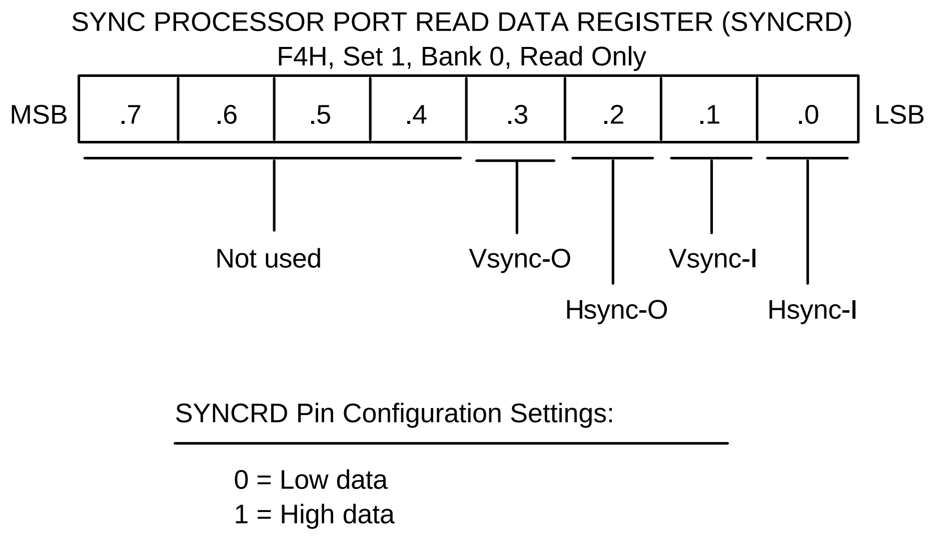


Figure 9-10. Sync Processor Port Read Data Register (SYNCRD)

☞ PROGRAMMING TIP — Configuring I/O Port Pins to Specification

The following sample program shows you how to configure the S3C8618/C8615 I/O ports to specification. The program comments explain the effect of the settings:

```
•
•
•
SB0                                ;  Select bank 0

LD          P0CONH,#0FFH           ;  Set port 0 high byte to push-pull output mode

LD          P0CONL,#0D5H           ;  Set P0.3 to push-pull output mode
                                   ;  Set P0.0–P0.2 to rising edge interrupt mode

LD          P0INT,#0FH             ;  Enable port 0 external interrupt

LD          P1CON,#00H             ;  Set port 1 to input mode

LD          P2CONH,#3FH            ;  Set port 2 high byte to PWM n-channel open-drain
                                   ;  output mode
                                   ;  (5-volt capability) and Csync input mode

LD          P2CONL,#0FFH           ;  Set port 2 low byte to PWM push-pull output mode

LD          P3CONH,#0AAH           ;  Set port 3 high byte to push-pull output mode

LD          P3CONL,#55H            ;  Set port 3 low byte to input mode with pull-up resistor
•
•
•
```

10 BASIC TIMER and TIMER 0

MODULE OVERVIEW

The S3C8618/C8615 have 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 (f_{OSC} divided by 4096, 1024, or 128) with multiplexer
— 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH, read-only)
— Basic timer control register, BTCON (set 1, D3H, read/write)

Timer 0

Timer 0 has two operating modes, one of which you select by the appropriate T0CON setting:

— Interval timer output mode
— Capture input mode with a rising or falling edge trigger

Timer 0 has the following functional components:

— Clock frequency divider (f_{OSC} divided by 128 or 8) with multiplexer
— 2 bit prescaler
— 8-bit counter (T0CNT), 8-bit comparator, and 8-bit reference data register (T0DATA)
— I/O pins for capture input (T0CAP)
— Timer 0 overflow interrupt and match/capture interrupt 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 f_{OSC}/4096. To disable the watchdog function, you must write the signature code '1010B' to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit basic timer counter, BTCNT (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.

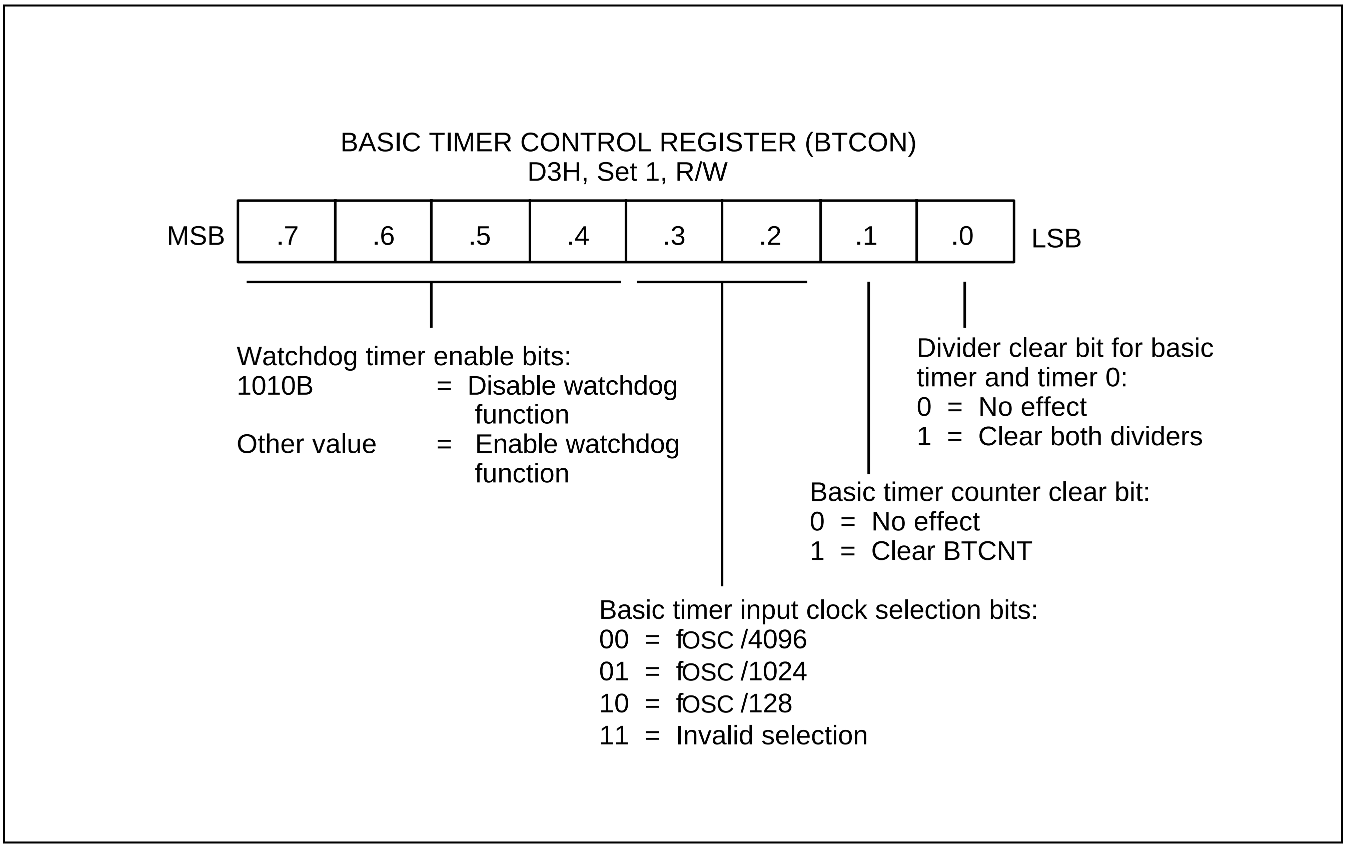


Figure 10-1. Basic Timer Control Register (BTCON)

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 external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of f_{OSC}/4096 (for reset), or at the rate of the preset clock source (for an external interrupt). When BTCNT.4 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 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 f_{OSC}/4096. If an external interrupt is used to release Stop mode, the BTCNT value increases at the rate of the preset clock source.
3. Clock oscillation stabilization interval begins and continues until bit 4 of the basic timer counter overflows.
4. When a BTCNT.4 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)

— Select the timer 0 input clock frequency

— Clear the timer 0 counter, T0CNT

— Enable the timer 0 overflow interrupt and timer 0 match/capture interrupt

— Select a 2-bit prescaler value for the timer 0 input clock

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 f_{OSC}/128, and disables the timer 0 overflow interrupt and match/capture interrupts. You can clear the timer 0 counter at any time during normal operation by writing a "1" to T0CON.2.

The timer 0 overflow interrupt (T0OVF) is interrupt level IRQ0 and has the vector address 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.0 to "1". There is no pending bit clear by software or static read bit which is H/W pending bit. When the interrupt request has been serviced, the pending condition is automatically cleared by hardware.

Do not use the Timer 0 overflow and the Timer 0 match/capture interrupts(IRQ0) at the same time. If you use them at the same time, it may occur a problem. The pending bit of the Timer 0 overflow and the Timer 0 match/capture interrupts is configured to the same interrupt request (IRQ0) by the hardware. After the Timer 0 overflow interrupt is occurred, the pending bit of the Timer 0 overflow interrupt is cleared automatically. (Where, the priority of the Timer 0 overflow interrupt is higher than the Timer 0 match/capture interrupt.) But the timer 0 match/capture interrupt is not serviced because the pending bit of the Timer 0 match/capture interrupt is already cleared when the Timer 0 overflow interrupt is serviced.

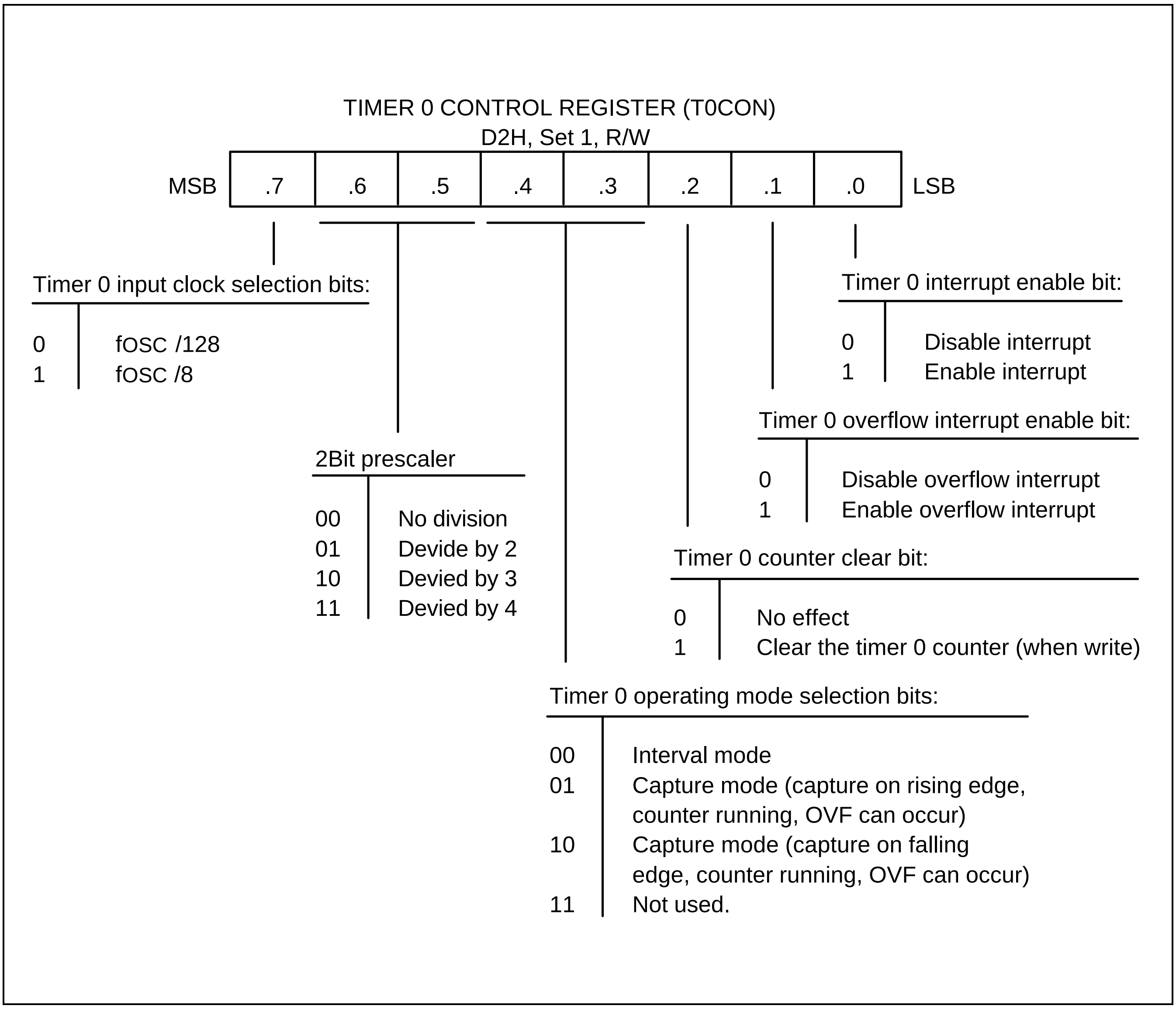


Figure 10-2. Timer 0 Control Register (T0CON)

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 and T0INT pending condition are automatically cleared by hardware when it has been serviced.

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, 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).

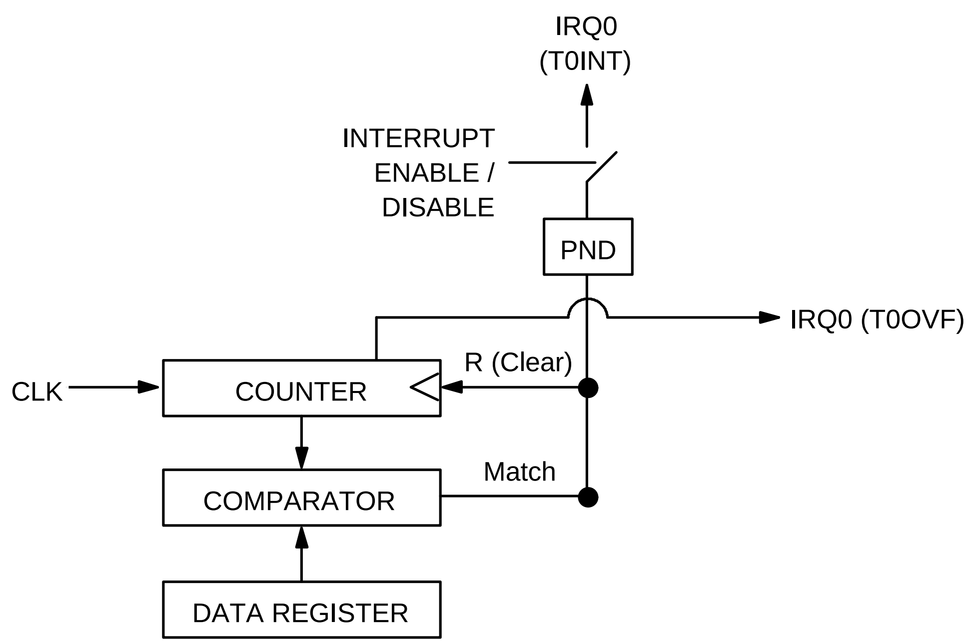


Figure 10-3. Simplified Timer 0 Function Diagram: Interval Timer Mode

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.

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 internal time of the signal that is being input at the T0CAP pin (see Figure 10-4) or the vertical sync output signal from sync processor.

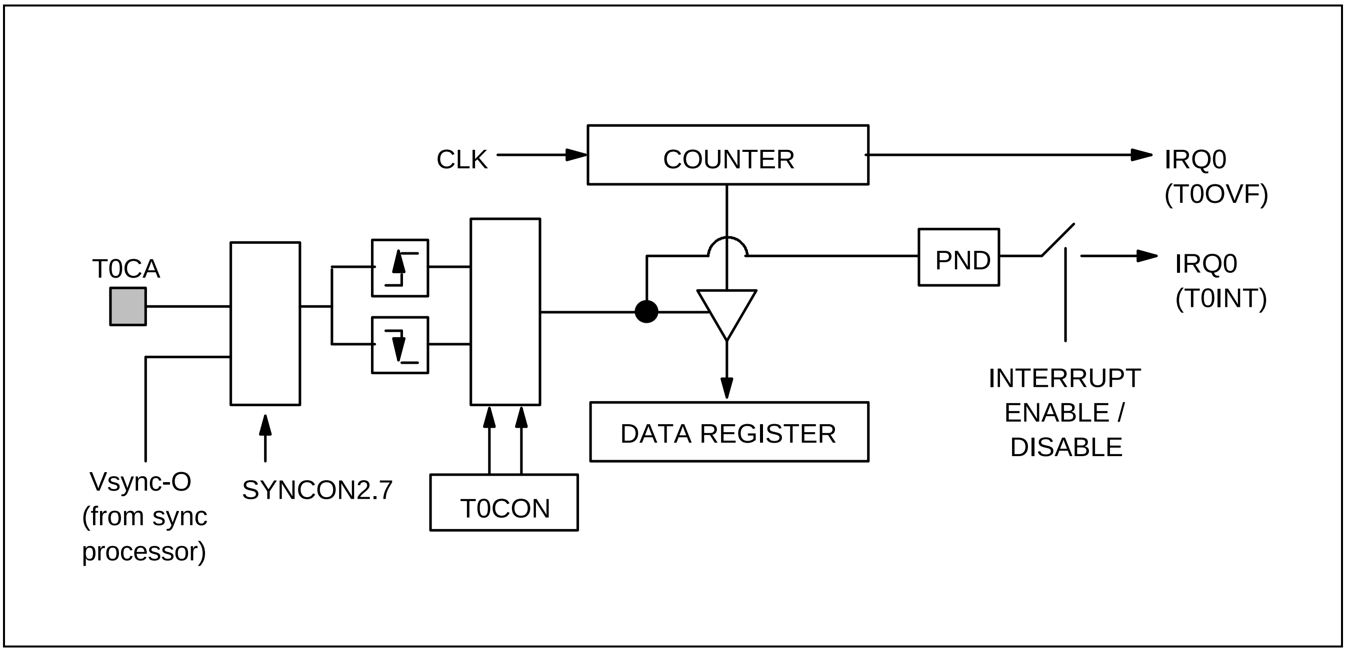
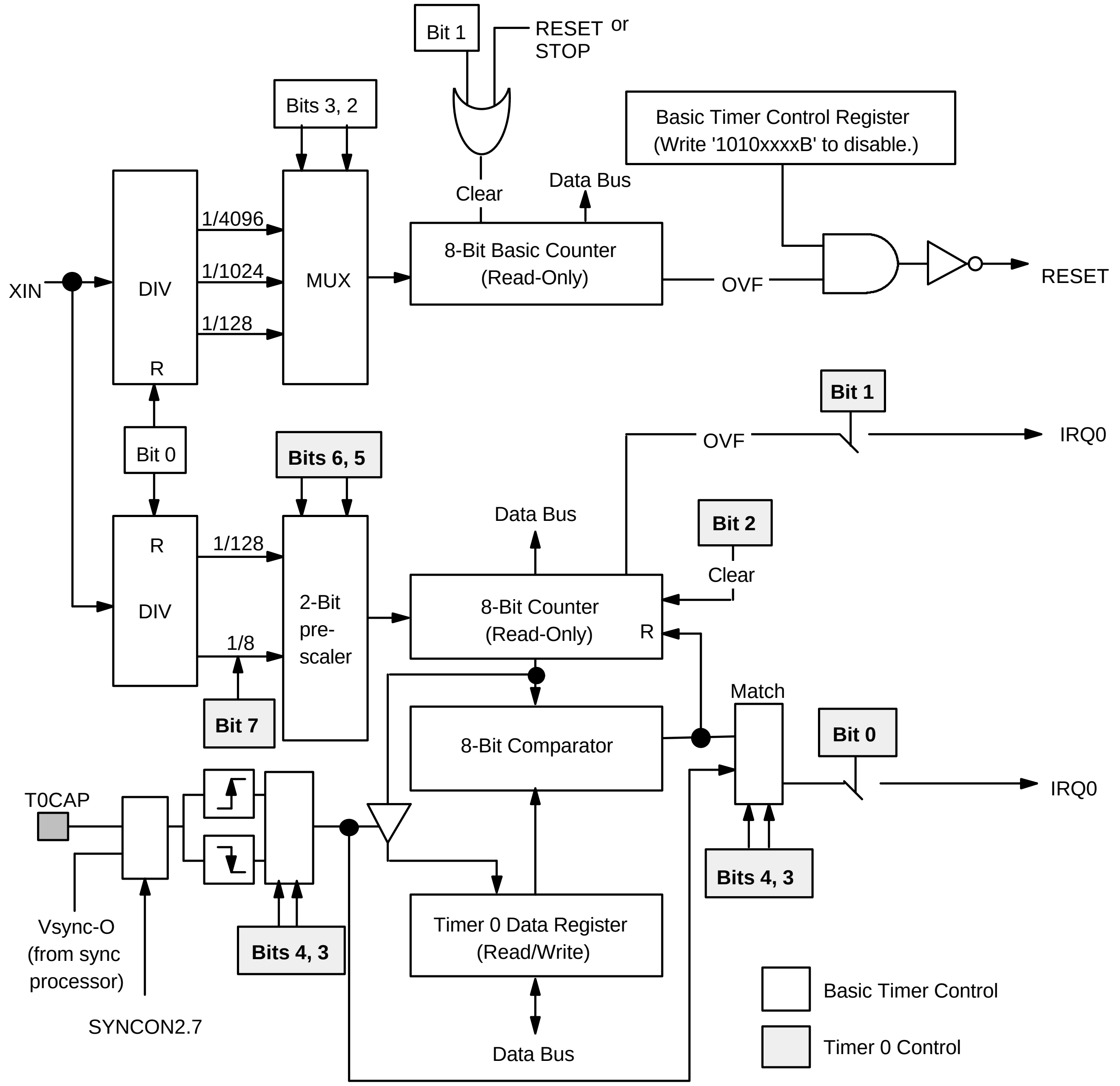


Figure 10-4. Simplified Timer 0 Function Diagram: Capture Mode



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

Figure 10-5. 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
         SB0                    ; Select bank 0
         LD     BTCON,#0AAH     ; Disable the watchdog timer
         LD     CLKCON,#98H     ; 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: fOSC/4096
                                ; Clear basic timer counter
         NOP
         NOP
         •
         •
         •
         JP     T,MAIN
         •
         •
         •
```

☞ PROGRAMMING TIP — Programming Timer 0

This sample program sets timer 0 to capture mode to calculate the input vertical sync signal. The program parameters are as follows:

— Timer 0 is used in capture mode with rising edge
— Oscillation frequency is 8 MHz
— 8 bit counter clock source is 1 MHz (= 8 MHz/8 = 1 µs)

```
            ORG       0FAH              ;  Timer 0 overflow interrupt
            VECTOR    T0OVF_INT
            ORG       0FCH              ;  Timer 0 match/capture interrupt
            VECTOR    T0INT
            ORG       0100H

RESET       DI                          ;  Disable all interrupts
            SB0                         ;  Select bank 0
            LD        BTCON,#0AAH       ;  Disable the watchdog timer
            LD        CLKCON,#98H       ;  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,#8FH        ;  Write '10001111B'
                                        ;  Input clock is fOSC/8
                                        ;  Capture mode
                                        ;  Enable the timer 0 interrupt
                                        ;  Enable the timer 0 overflow interrupt
```

☞ PROGRAMMING TIP — Programming Timer 0

```
SRP     #0C0H     ;                     Set register pointer ← 0C0H
        EI                              ; Enable interrupts
        •
        •
        •
T0INT:  PUSH      R4
        PUSH      R5
        SB0                             ; Select bank 0
        LD        R4, T0OVFCNTR         ; R4 ← Counting number of Timer 0 overflow interrupt
        CLR       R5
        MULT      RR4, #0FFH
        ADD       R5, T0DATA            ; Timer 0 capture data
        ADC       R4, #0H               ; RR4 ← (T0OVFCNTR) x 256 + T0DATA
        LD        VsyncInterval, R4     ; VsyncInterval = #xxxxx µs
        LD        VsyncInterval+1, R5
        POP       R5
        POP       R4
        IRET
```

11 TIMER 1

OVERVIEW

The S3C8618/C8615 microcontrollers have one 8-bit timer/counter, timer 1 (T1). During normal operation, timer 1 runs continuously. Timer 1 has selectable clock sources and can generate the timer 1 overflow interrupt.

Timer 1 has one control register, T1CON, and an 8-bit counter register, T1CNT. T1CON is located in set 1, bank 0, at address F5H and is read/write accessible. The timer 1 counter, T1CNT, is located in set 1, bank 0, at address F6H and is read-only addressable.

You can select one of the following clock sources as the timer 1 clock:

— External clock input (T1CLK pin)

— Hsync or Vsync input from the sync processor block

— CPU clock divided by 4096 or 512

TIMER 1 OVERFLOW INTERRUPT CONTROL

Timer 1 generates an overflow signal whenever a T1CNT overflow occurs. If you set the timer 1 overflow interrupt enable bit, T1CON.3, to “1”, an interrupt is generated each time an overflow state is detected. After the interrupt request is generated, the counter register value is cleared and counting resumes from ‘00H’.

Program software can poll the pending bit T1CON.2 to detect when a timer 1 interrupt pending condition exists (T1CON.2 = “1”). When the interrupt request is acknowledged by the CPU and the service routine starts, the interrupt service routine must then clear the interrupt pending condition by writing a “0” to T1CON.2.

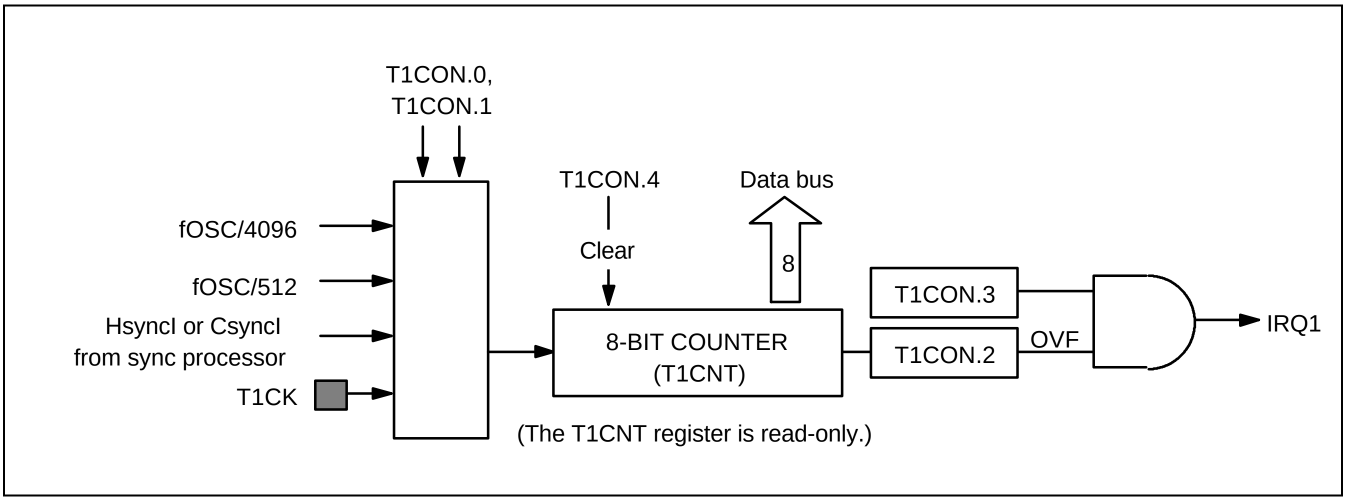


Figure 11-1. Timer 1 Block Diagram

TIMER 1 CONTROL REGISTER (T1CON)

The timer 1 control register, T1CON, is located in set 1, bank 0, at address F4H and is read/write addressable. Only bits T1CON.4–T1CON.0 are mapped in the S3C8618/C8615 implementation. T1CON contains control settings for the following functions:

— Timer 1 overflow interrupt enable/disable

— Timer 1 overflow interrupt pending control (read for status, write to clear)

— Timer 1 clock source selection

The setting for bit-pair T1CON.0 and T1CON.1 selects the timer 1 clock input. T1CON.2 is the interrupt pending flag for the timer 1 overflow interrupt (IRQ1, vector F6H). To clear a timer 1 interrupt pending condition, the interrupt service routine must write a "0" to T1CON.2 after the CPU has acknowledged the request.

A reset operation clears T1CON to '00H', selecting the CPU clock divided by 4096 as the T1 clock and disabling the timer 1 overflow interrupt.

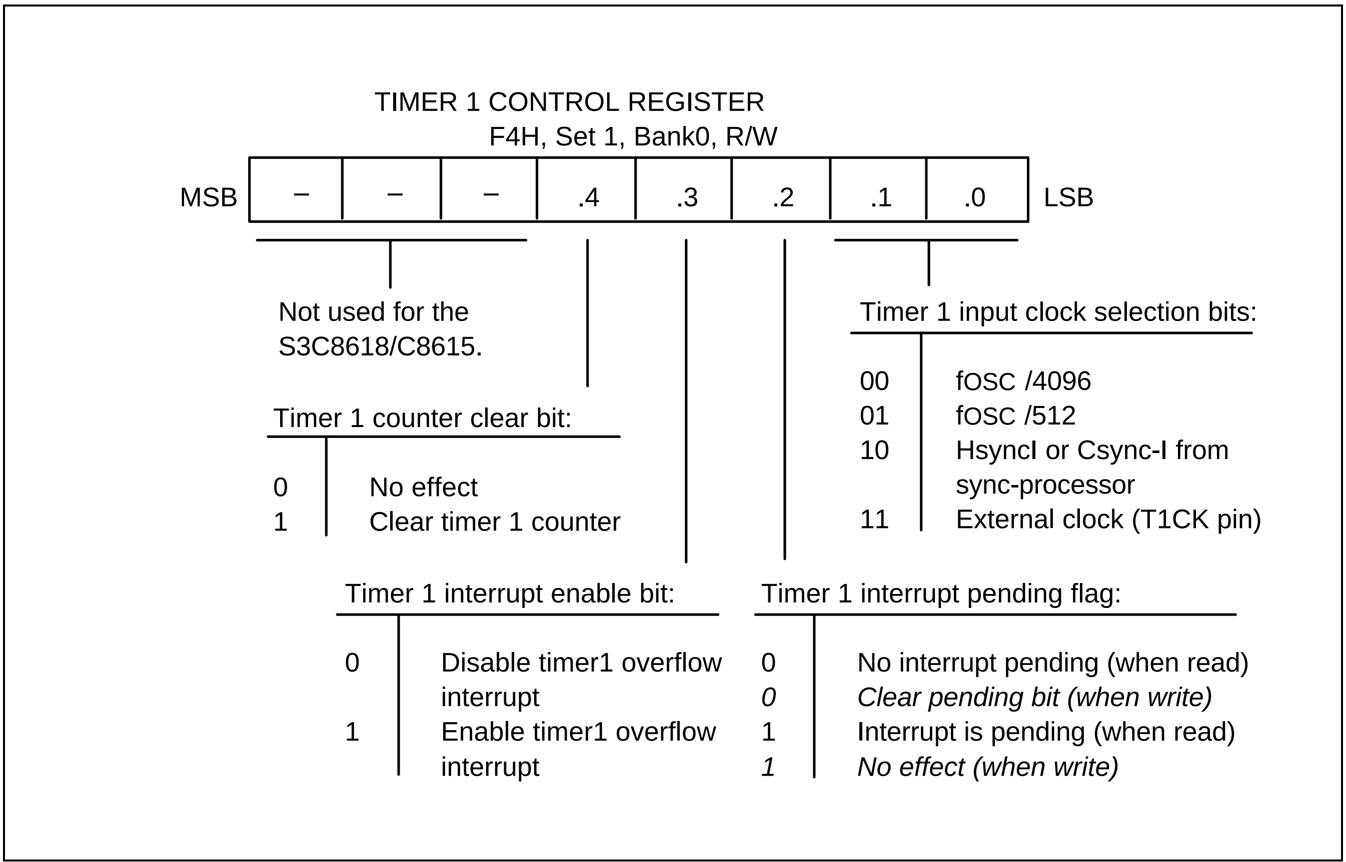


Figure 11-2. Timer 1 Control Register (T1CON)

☞ PROGRAMMING TIP — Programming Timer 1

This example sets timer 1 to event counter. The parameters of this sample program are as follows:

— Timer 1 is enabled

— Timer input clock source is Hsync-I or Csync-I from sync processor

```
            ORG       0F6H                ;  Timer 1 overflow interrupt
            VECTOR    T1OVF_INT

            ORG       0100H
RESET:
            DI                            ;  Disable all interrupts
            SB0                           ;  Select bank 0
            LD        BTCON,#0AAH         ;  Disable the watchdog timer
            LD        CLKCON,#98H         ;  Select non-divided clock
            CLR       SYM                 ;  Disable global and fast interrupts
            CLR       SPL                 ;  Stack pointer low byte ← "0"
                                          ;  Stack area starts at 0FFH
            •
            •
            •
            LD        T1CON,#1AH          ;  Write '00011010B'
                                          ;  T1 clock = Hsync-I from sync-processor
            SRP       #0C0H               ;  Set register pointer ← 0C0H
            EI                            ;  Enable interrupts
            •
            •
            •
T1OVF_INT:
            SB0
            INC       T1OVFCNTR           ;  Counting number of Hsync signal overflow interrupt
            IRET
```

12 TIMER 2

OVERVIEW

To provide additional timer/counter functionality, the S3C8618/C8615 microcontrollers have another 8-bit timer/counter called timer 2 (T2). During normal operation, timer 2 runs continuously.

Timer 2 generates the timer 2 interrupt (IRQ1, vector EEH) and has two alternative clock sources: the oscillator frequency (f_{OSC}) divided by 1024 or the non-divided oscillator frequency.

Timer 2 has one control register and one 8-bit data register. The T2 control register, T2CON, is located in set 1, bank 0, F7H, and the T2 data register, T2DATA, is in set 1, bank 0, F8H. T2CON is read/write addressable and T2DATA is read-only.

TIMER 2 INTERRUPT

Timer 2 generates an interrupt whenever a match occurs between the 8-bit counter value and the value you write to the timer 2 data register, T2DATA. To enable the timer 2 interrupt, you set the interrupt enable bit, T2CON.1, to “1”. When the interrupt request is serviced, the 8-bit counter value is cleared and counting resumes from ‘00H’.

Program software can poll the timer 2 interrupt pending flag, T2CON.0, to detect when an interrupt pending condition exists (T2CON.0 = “1”). When the interrupt request is acknowledged by the CPU and the service routine starts, the interrupt service routine must then clear the interrupt pending condition by writing a “0” to T2CON.0.

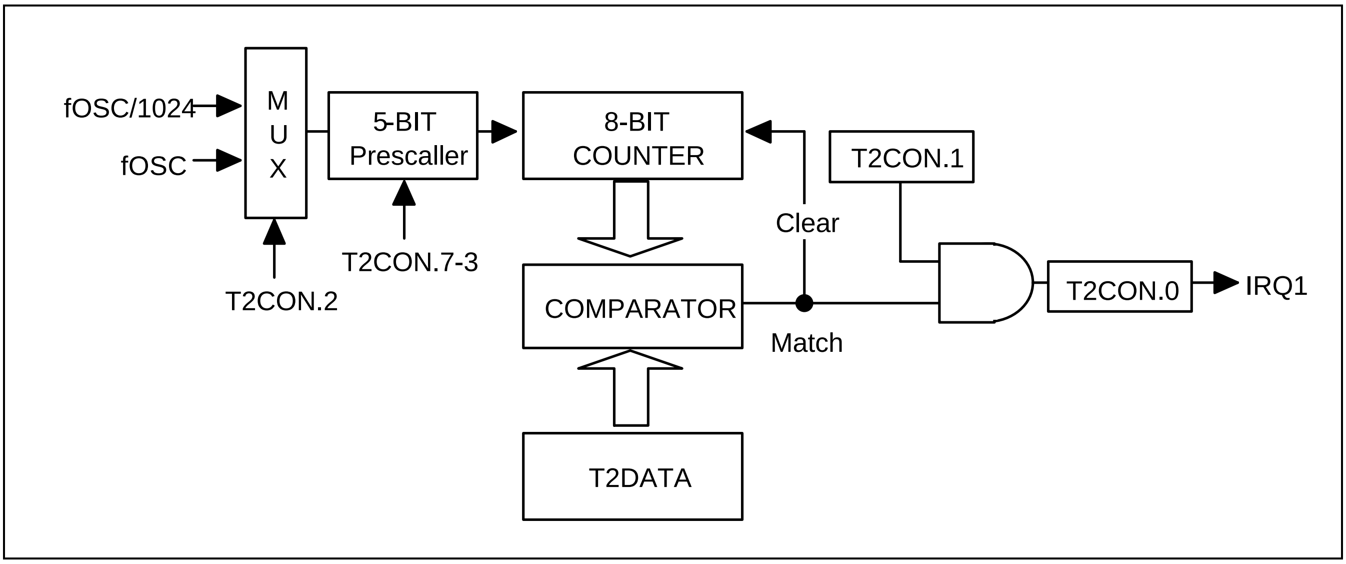


Figure 12-1. Timer 2 Block Diagram

TIMER 2 CONTROL REGISTER (T2CON)

The timer 2 control register, T2CON, is located in set 1, bank 0, F7H, and is read/write addressable. Bit 0 is not used. T2CON contains a 4-bit prescaler value for the timer 2 input clock, as well as control settings for the following functions (see Figure 12-2):

— Timer 2 clock source selection

— Timer 2 interrupt enable/disable

— Timer 2 interrupt pending control (read for status, write to clear)

A reset operation clears T2CON to 'F0H' (11111000B), thereby setting the 5-bit prescaler value to 'divide by 32.

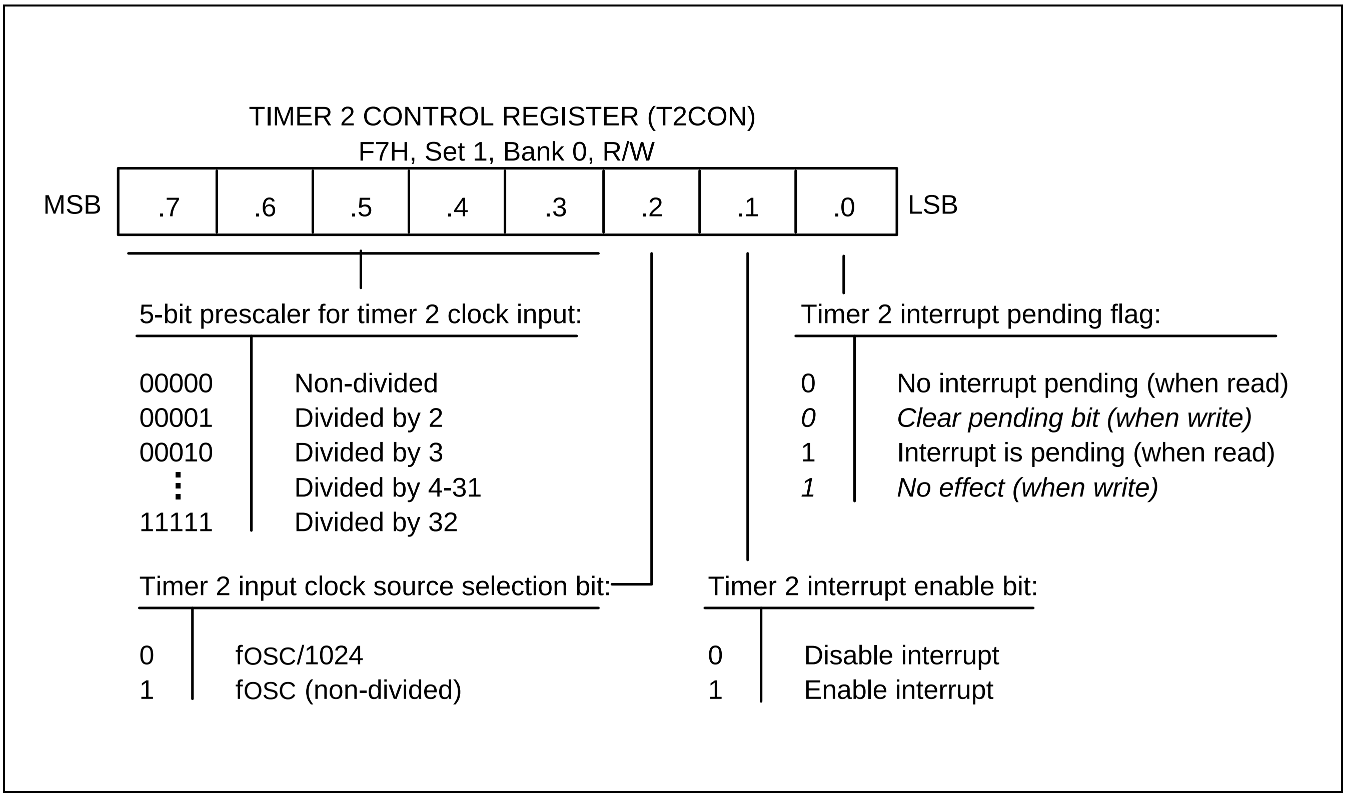


Figure 12-2. Timer 2 Control Register (T2CON)

☞ PROGRAMMING TIP — Programming Timer 2

Timer 2 is the interval mode timer. The parameters of this sample program are as follows:

— Timer 2 is enabled
— CPU clock frequency = 8MHz
— Time interval is set to 1 ms

```
            ORG       EEH
            VECTOR    T2INT

RESET:
            DI                                ;  Disable all interrupts
            •
            •
            •
            LD        T2CON,#0FAH             ;  Write '11111010B'
                                              ;  Input clock is fosc/1024
                                              ;  Enable the timer 2 interrupt
                                              ;  Prescaler value is 32
            LD        T2DATA,#249             ;  Set timer interval to 1ms
                                              ;  (8MHz/32) ÷ (249+1)=1kHz (1ms)
            EI                                ;  Enable interrupts
            •
            •
            •
T2_INT:     PUSH      RP0
            SB0
            SRP0      #60H
            INC       R0
            CP        R0,#10                  ;  1ms × 10=10ms
            JR        ULE, NO_10MS_SET
            LD        R1, T1OVFCNTR           ;  Timer1 is a event counter
                                              ;  for Hsync signal from sync processor
            LD        R2, T1CNT
            CLR       T1OVFCNTR
            AND       T1CON, #11101111B       ;  Clear Timer 1 counter
NO_10MS_SET:
            LD        T2CON, #0FAH            ;  Pending bit clear
            POP       RP0
            IRET
```

13 PULSE WIDTH MODULATION

PWM MODULE

The S3C8618/C8615 microcontrollers have seven 8-bit PWM circuits. The 8-bit circuits are PWM0–PWM6. The operation of all PWM circuits is controlled by a single control register, PWMCON.

The PWM counter, a 8-bit incrementing counter, is used by the 8-bit PWM circuits. To start the counter and enable the PWM circuits, you set PWMCON.5 to "1". If the counter is stopped, it retains its current count value; when re-started, it resumes counting from the retained count value.

By modifying the prescaler value, you can divide the input clock by one (non-divided), two, three, or four. The prescaler output is the clock frequency of the PWM counter.

The PWM counter overflows when it reaches 'FFH', and then continues counting from zero.

PWM CONTROL REGISTER (PWMCON)

The control register for the PWM module, PWMCON, is located in set 1, bank 1, at register address E7H. You use PWMCON bit settings to control the following functions in the 8-bit:

— PWM counter operation: stop/start (or resume counting)

A reset clears PWMCON to '00H', disabling all PWM functions.

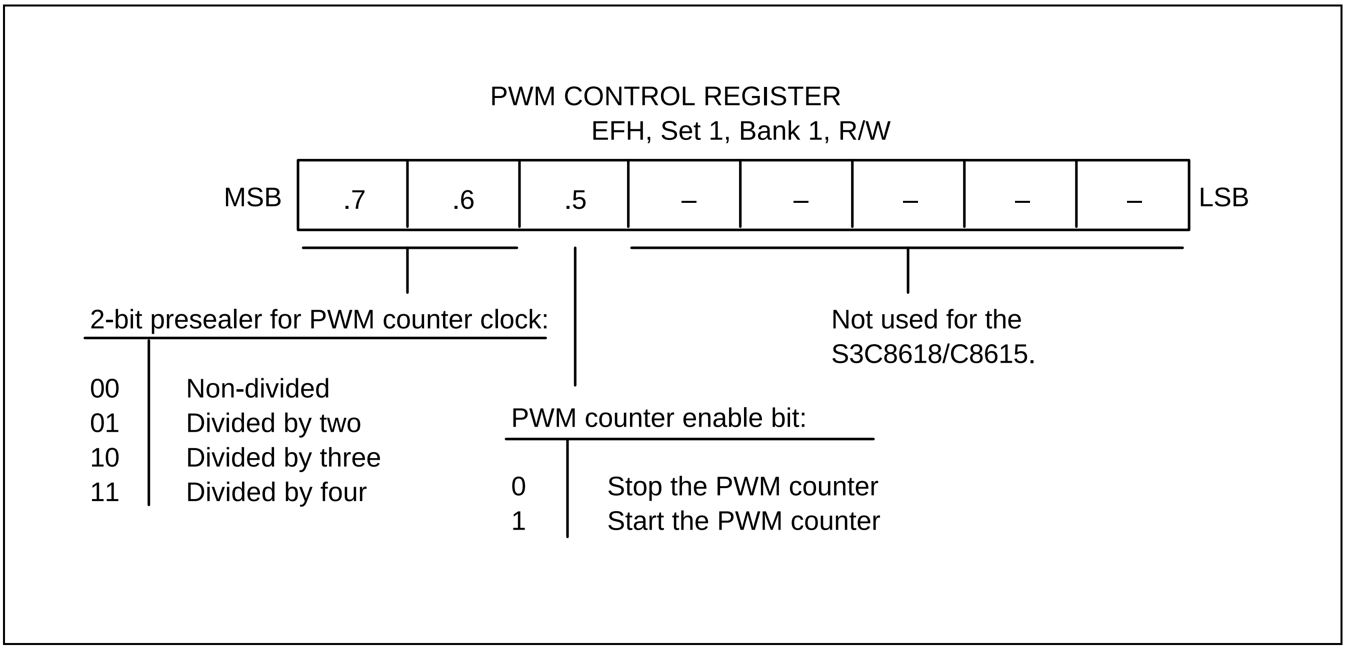


Figure 13-1. PWM Control Register (PWMCON)

PWM0–PWM6

The S3C8618/C8615 microcontrollers have seven 8-bit PWM circuits, PWM0–PWM6. Each 8-bit PWM data unit is comprised of a 8-bit basic frame. The 8-bit PWM circuits have the following components:

— 8-bit counter
— 8-bit comparators
— 8-bit PWM data registers (PWM0–PWM6)
— PWM output pins (PWM0–PWM6)

The PWM0–PWM6 circuits are controlled by the PWMCON register (set 1, bank 1, E7H).

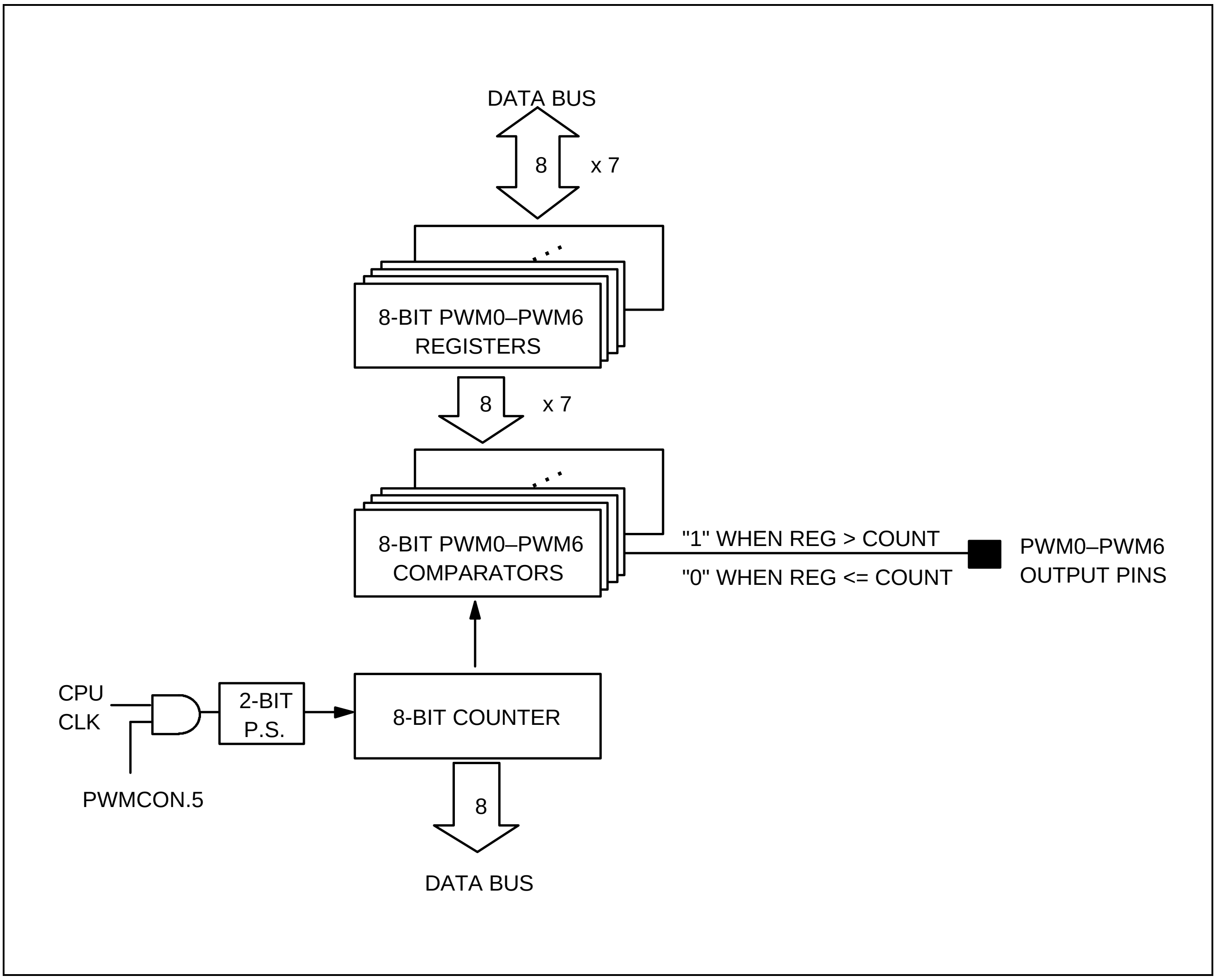


Figure 13-2. Block Diagram for PWM0–PWM6

PWM0–PWM6 FUNCTION DESCRIPTION

All seven 8-bit PWM circuits function identically and each has its own 8-bit data register and 8-bit comparator. Each circuit compares a unique data register value to the 8-bit PWM counter.

The PWM0–PWM6 data registers are located in set 1, bank 1, at locations E0H–E6H, respectively. These data registers are read/write addressable. By loading specific values into the respective data registers, you can modulate the pulse width at the corresponding PWM output pins, PWM0–PWM6. (PWM0–PWM6 correspond to port 2 pins P2.0–P2.6.

The level at the output pins toggles High and Low at a frequency equal to the counter clock, divided by 256 (2^8). The duty cycle of the 8-bit PWM pins ranges from 0% to 99.6 % duty cycle (255/256), based on the corresponding data register values.

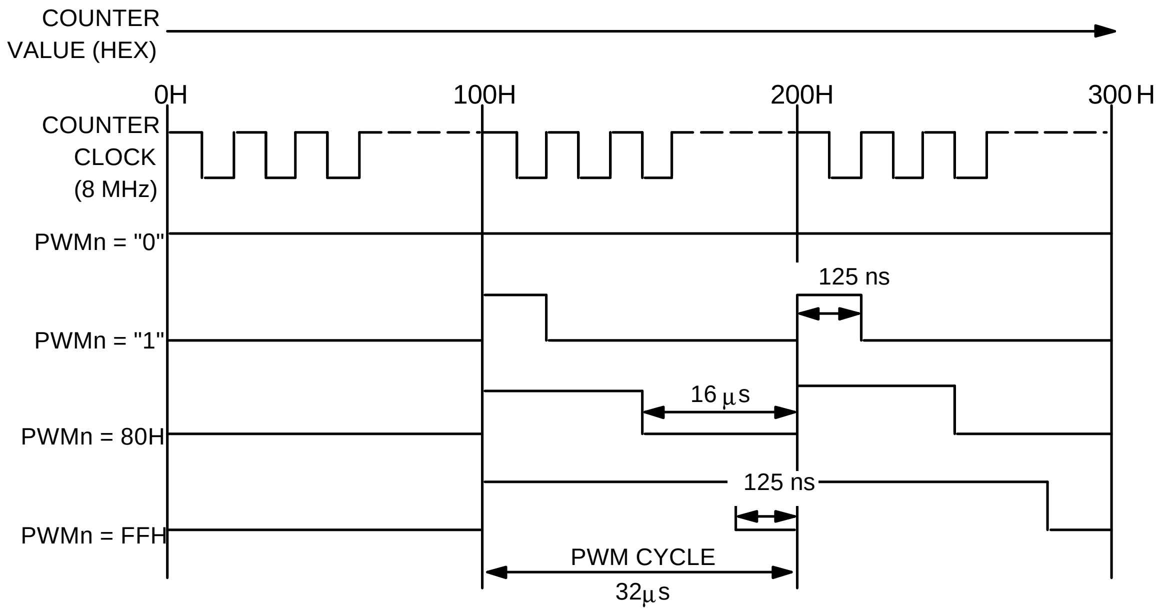
To determine the output duty cycle of an 8-bit PWM circuit, its 8-bit comparator sends the output level High when the data register value is greater than the lower 8-bit count value. The output level is Low when the data register value is less than or equal to the lower 8-bit count value. The output level at the PWM0–PWM6 pins remains at Low level for the first 256 counter clocks. Then, each PWM waveform is repeated continuously, at the same frequency and duty cycle, until one of three events occurs:

— The counter is stopped
— The counter clock frequency is changed
— A new value is written to the PWM data register

STAGGERED PWM OUTPUTS

The PWM0–PWM6 outputs are staggered to reduce the overall noise level on the pulse width modulation circuits. If you load the same value to the PWM0–PWM6 data registers, a match condition (data register value is equal to the 8-bit count value) will occur on the same clock cycle for all seven 8-bit PWM circuits.

For example, the PWM0 output is delayed by one-half of a counter clock, PWM1 output by one-half of a counter clock, PWM2 output by one-half of a counter clock, and so on for subsequent clock cycles (see Figure 13-4).



NOTES

1. A counter clock value of 8 MHz is assumed for all timing values.
2. 'n' = 0–6, for PWM0–PWM6.

Figure 13-3. PWM Waveforms for PWM0–PWM6

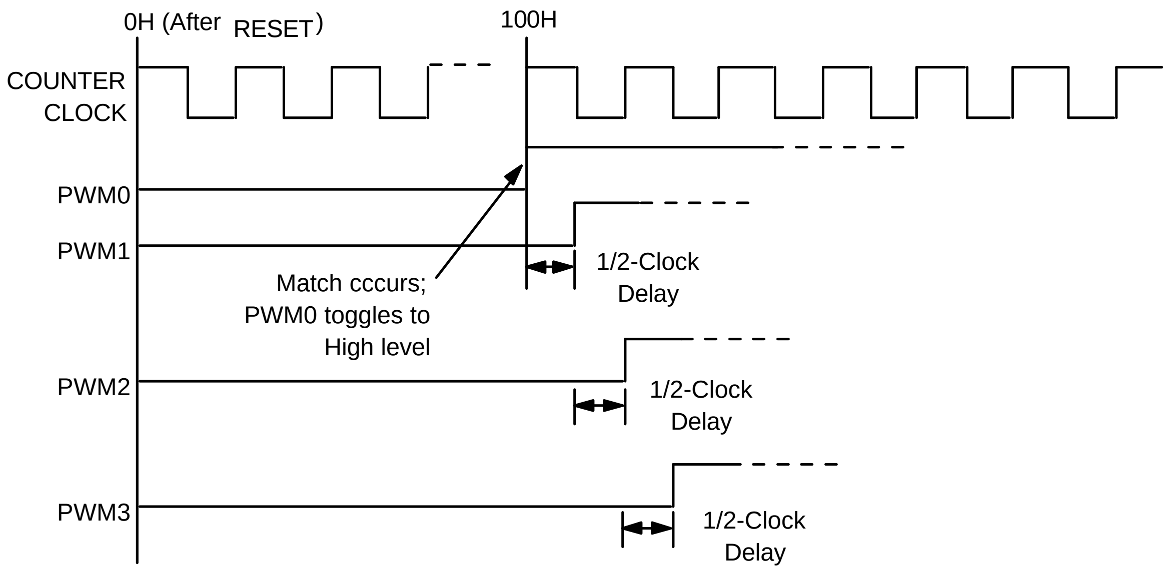


Figure 13-4. PWM Clock to PWM0–PWM6 Output Delays

PWM COUNTER

The PWM counter is a 8-bit incrementing counter. The same 8-bit counter is used by all PWM circuits. To determine the PWM module's base operating frequency, the counter is compared to the PWM data register value.

PWM DATA REGISTERS

A reset operation disables all PWM output. The current counter value is retained when the counter stops. When the counter starts, counting resumes at the retained value.

PWM CLOCK RATE

The timing of the 8-bit output channel is based on the maximum 12-MHz CPU clock frequency. The 2-bit prescaler value in the PWMCON register determines the frequency of the counter clock: You can set PWMCON.6 and PWMCON.7 to divide the CPU clock frequency by one (non-divided), two, three, or four.

Because the maximum CPU clock rate for the S3C8618/C8615 microcontrollers is 12 MHz, the maximum base PWM frequency is 46.8 kHz (12 MHz divided by 256). This assumes a non-divided CPU clock.

☞ PROGRAMMING TIP — Programming PWM0 to Sample Specifications

This sample program executes a test of the PWM block. The program parameters are as follows:

— The oscillation frequency of the main crystal is 8 MHz

— PWM frequency is 31.2 kHz

```
RESET:      DI
            SB0                                     ;   Select bank 0
            •
            •
            •
            LD      P2CONH,#11111111B               ;   Select n-channel, open-drain PWM output
            LD      P2CONL,#11111111B               ;   Select push-pull PWM output
            SB1                                     ;   Select bank 1
            OR      PWMCON,#00100000B               ;   PWMCON.5 ← 1; start the counter
                                                    ;   PWM counter clock is Fosc
            SB0                                     ;   Select bank 0
            EI
            •
            •
            •
PWMstart:
            SB1                                     ;   Select bank 1
            LD      PWM0, #80H                      ;   Loading PWM0 data
            SB0
            RET
```

14 SYNC PROCESSOR

OVERVIEW

The S3C8618/C8615 multi-sync signal processor (sync processor) is designed to process horizontal (Hsync) and vertical (Vsync) signals that are input to a multi-sync monitor. The sync processor can perform the following functions:

— Detect sync input signals (Vsync-I, Hsync-I, and Csync-I, also called Screen-On-Green, or SOG)

— Output a pseudo sync signal

— Detect the polarity of sync input signals

— Separate and output sync signals (Hsync-O, Vsync-O, and Clamp-O)

The sync processor circuits are controlled by three control registers: SYNCON0, SYNCON1, and SYNCON2.

Capture/Compare Modes

SYNCON0 register setting controls the operation of the sync processor's 5-bit counter, as well as the block's capture/compare functions. Using the 5-bit counter, the sync processor can operate in capture mode (sync polarity check mode) or in compare mode (Vsync separation mode).

The counter value increments when a High level sync signal is detected and decrements when a Low level signal is detected. No overflow or underflow can occur. That is, the 5-bit counter increments until it reaches the maximum value of 11111B and then stops or decrements until it reaches the minimum value of 00000B. You can select non-divided $f_{OSC}/2$ or $f_{OSC}/3$ as the counter's clock input source.

In capture mode, the 5-bit counter value is loaded into the capture/compare register (bits 4–0 of SYNCON0) whenever a specified signal edge is detected. In compare mode, a High signal level is output to a multiplexer whenever the counter value reaches 11111B and a Low level is output when the counter value reaches 00000B. The signal level remains constant when the counter value is less than or equal to 11111B or greater than 00000B.

Clamp Signal Output

SYNCON1 register settings control Clamp signal output and frequency. Clamp output can be completely inhibited, or it can be generated at two, four, or eight times f_{OSC}. You can specify the signal edge on which the selected Clamp frequency is to be output ("front porch" or "back porch"). SYNCON1 settings are also used to control the polarity and to monitor the status of the Clamp-O, Vsync-O, and Hsync-O signals.

Pseudo Sync Generator

SYNCON2 settings control the pseudo Vsync (60.1 Hz, 78.1 Hz or 86.8 Hz) and pseudo Hsync (37.0 kHz, 47.6kHz or 58.8 kHz) output frequencies. The polarity of these frequencies is always positive, with duty cycles of 5 % and 25 %, respectively. The pseudo sync generator supports factory testing of the sync processor block and also protects a system against the effect of unexpected signals in transition period while mode changing.

Logic for Detecting Sync-On-Green (SOG)

Special logic in the sync processor block can compare Hsync and Csync input signals to detect Sync-On-Green (SOG).

Vsync Detection Interrupt

The sync processor can generate a Vsync detection interrupt (level IRQ5, vector ECH). This interrupt is triggered by a rising signal edge at the Vsync-I pin. The interrupt pending condition is automatically cleared by hardware when the corresponding service routine is initiated.

SYNC PROCESSOR CONTROL REGISTER 0 (SYNCON0)

The sync processor control register 0, SYNCON0, is located in set 1, bank 0, at address F1H. It is read/write addressable. SYNCON0 bits 4–0 hold the 5-bit counter value which is used for capture/compare functions. SYNCON0 settings also control the following sync processor functions:

— Horizontal or composite sync input (Hsync-I or Csync-I) selection

— Input edge selection for the 5-bit counter (rising or falling edges)

— Mode selection for the 5-bit counter (capture mode or compare mode)

See Figure 14-1 for a detailed description of SYNCON0 register settings.

SYNC PROCESSOR CONTROL REGISTER 0 (SYNCON0)
F1H, Set 1, Bank 0, R/W

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

Sync input selection bit (.7):
0 = Hsync-I
1 = Csync-I

Input edge selection bit for the 5-bit capture/compare counter (.6):
0 = Falling edges only input
1 = Rising edges only input

Mode selection bit for the 5-bit capture/compare counter (.5):
0 = Capture mode
1 = Compare mode (Vsync separation mode)

5-bit capture/compare counter value bits (.4–.0):

Whenever a High signal level is detected, the count value is incremented by one until it reaches the maximum value of ‘11111B’. (No overflow occurs.)

Whenever a Low signal level is detected, the count value is decremented by one until it reaches the minimum value of ‘00000B’. (No underflow occurs.)

NOTES:

1. Capture mode is also sync polarity check mode. In this mode, the 5-bit counter value is captured (loaded into the counter register bits, SYNCON0.4-SYNCON0.0) whenever the signal edge selected by the SYNCON0.6 setting is detected. Output to the multiplexer is disabled.
2. Compare mode is also called Vsync separation mode. In this mode, output to the multiplexer is enabled. When the counter value is ‘11111B’, the output is High level; when the counter value is ‘00000B’, the output is Low level. Whenever the counter value is less than or equal to (<) ‘11111B’, or greater than (>) ‘00000B’, the previous output level is retained.

Figure 14-1. Sync Processor Control Register 0 (SYNCON0)

SYNC PROCESSOR CONTROL REGISTER 1 (SYNCON1)

The sync processor control register 1, SYNCON1, is located in set 1, bank 0, at address F2H. It is read/write addressable. Using SYNCON1 settings, you can:

— Compare Hsync and Csync input signals to detect Sync-On-Green (SOG) input

— Generate a clock pulse for Clamp signal output

— Select “front porch” or “back porch” mode for Clamp-O

— Monitor Clamp-O, Vsync-O, and Hsync-O status

— Select the clock source for Vsync-O (VCLK or 5-bit counter output)

See Figure 14-2 for a detailed description of SYNCON1 register settings.

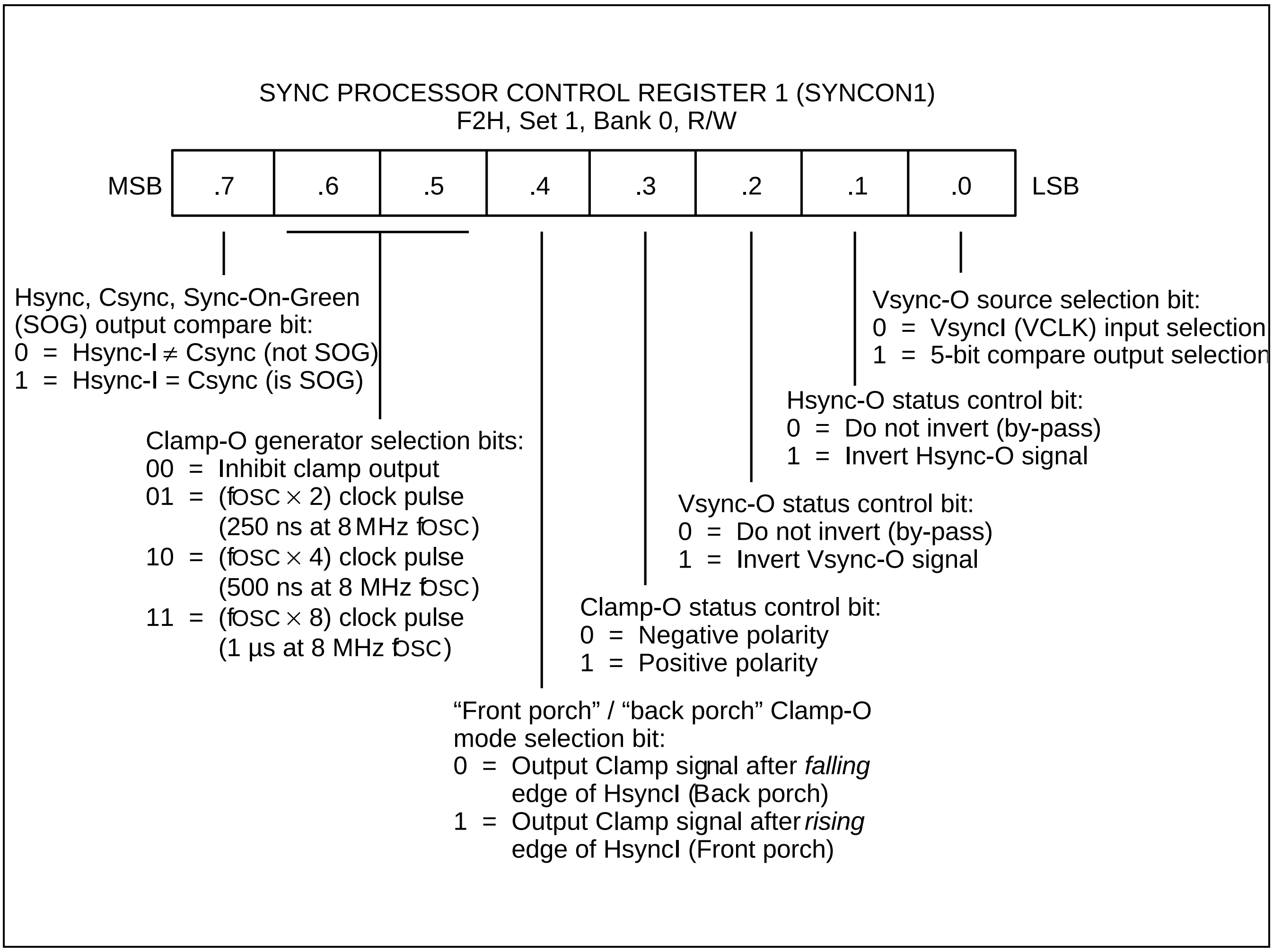


Figure 14-2. Sync Processor Control Register 1 (SYNCON1)

SYNC PROCESSOR CONTROL REGISTER 2 (SYNCON2)

The sync processor control register 2, SYNCON2, is located in set 1, bank 0, at address F3H. It is read/write addressable. Using SYNCON2 settings, you can:

— Select timer 0 capture input signal

— Select the pseudo sync processor's operating mode (normal operation or factory test mode)

— Select the clock source for the 5-bit counter

— Control output of the pseudo Vsync/Hsync generator

— Enable and disable the Vsync detection interrupt

See Figure 14-3 for a detailed description of SYNCON2 register settings.

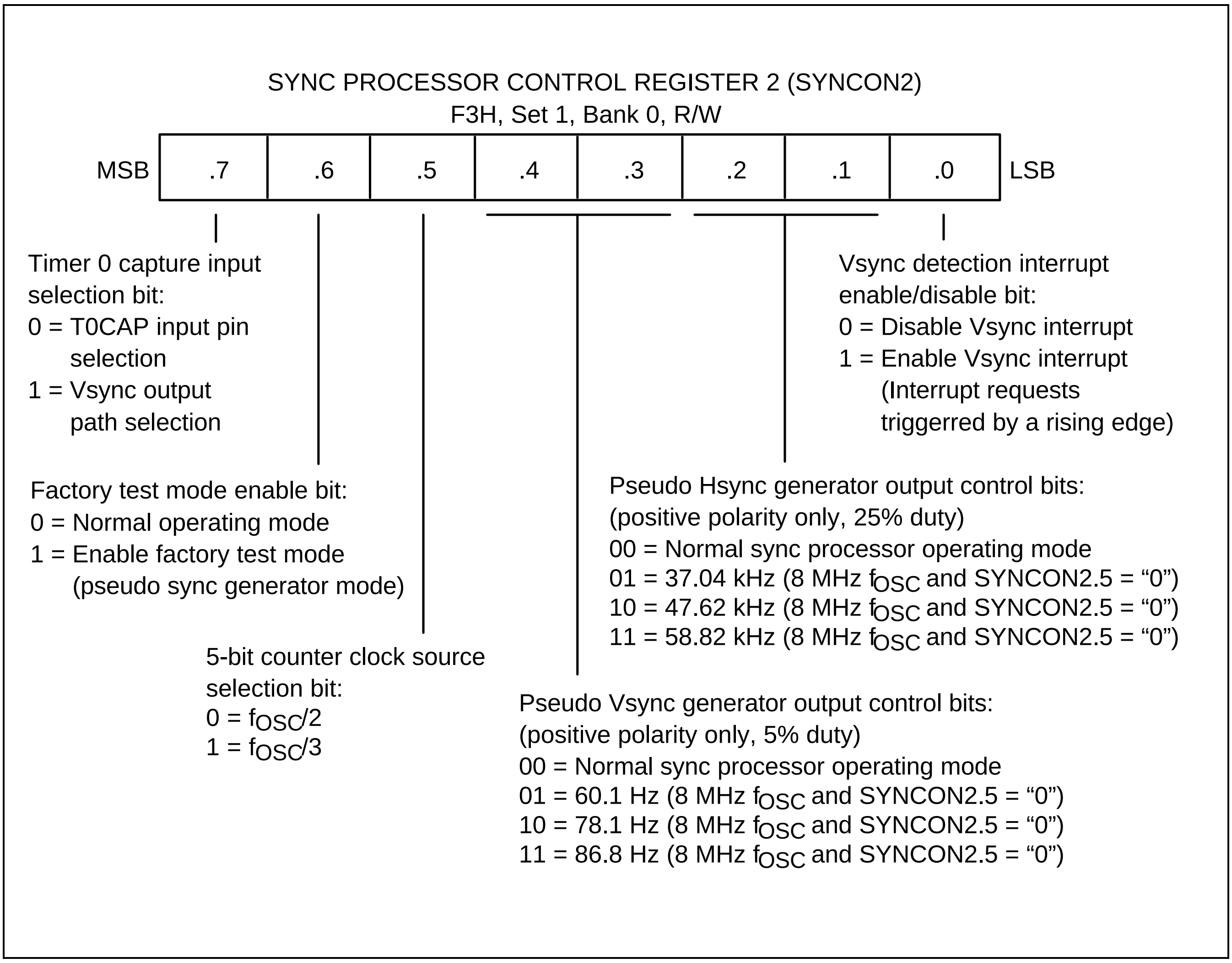


Figure 14-3. Sync Processor Control Register 2 (SYNCON2)

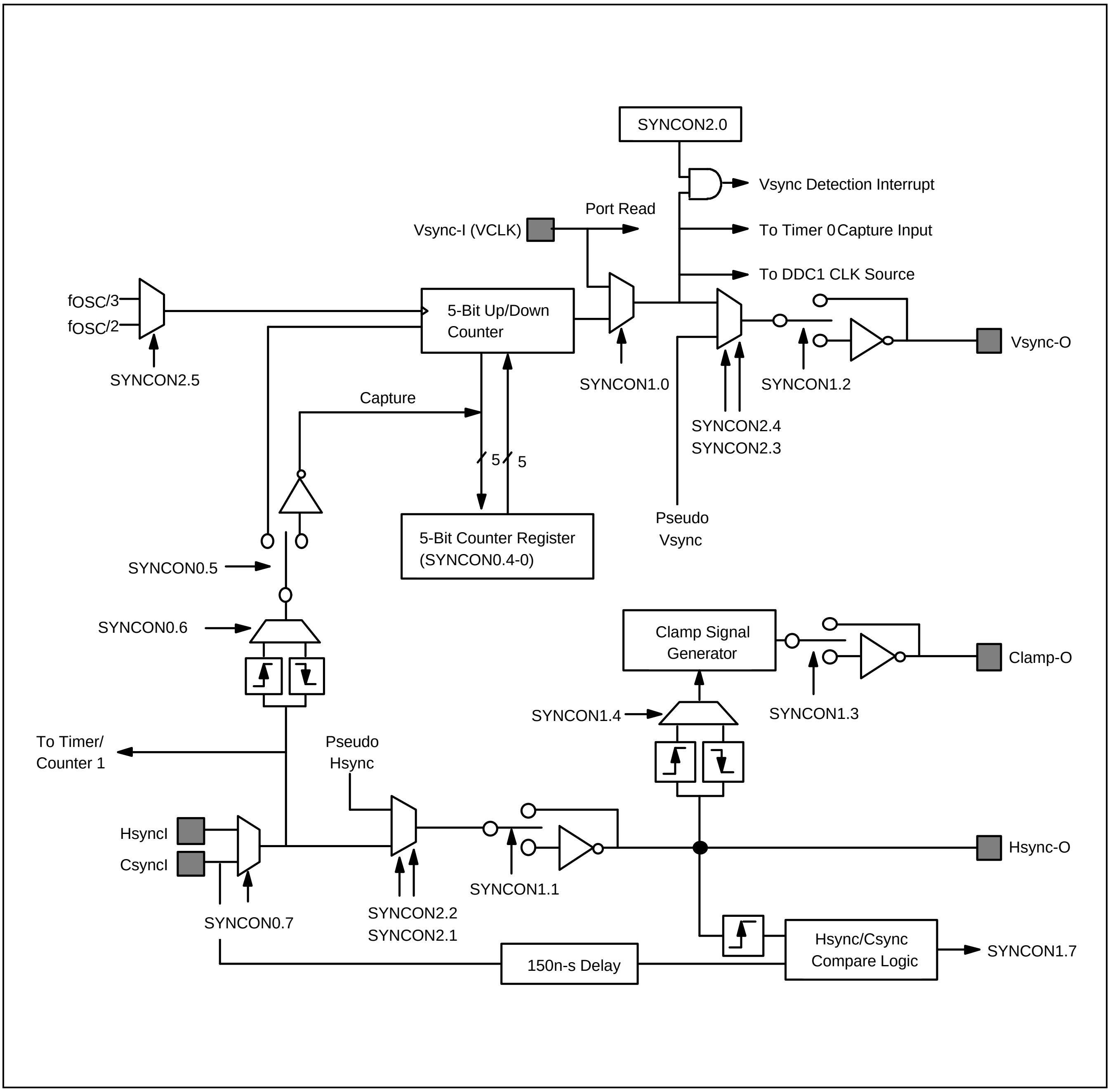


Figure 14-4. Sync Processor Block Diagram

DETECTING SYNC SIGNAL INPUT

You can detect the presence of a sync signal in two ways — directly or indirectly. The direct detection method can be implemented in read port. The indirect detection method is interrupt-driven and uses the S3C8618/C8615 sync processor hardware. These methods are explained in detail below.

Direct Detection Method

By reading the input status directly on the sync input pins Vsync-I, Hsync-I, Csync-I, you can detect the presence of the incoming sync for a corresponding output.

To enable direct sync input detection, you set SYNCON1.0 to “0” (for Vsync-I), SYNCON0.7 to “0” (for Hsync-I), and SYNCON0.7 to “1” (for Csync-I).

You then read the state of the input pin(s) over a period of time to detect transitions in the signal level(s). If a transition is detected, it can be assumed that a sync signal is present.

Indirect Detection Method

To indirectly detect vertical sync input at the Vsync-I pin, you use register settings to assign either the timer 0 capture interrupt or the Vsync detection interrupt to this pin.

For indirect detection of horizontal or composite input at the Hsync-I or Csync-I pin, you use the timer 1 input clock source to generate a timer 1 overflow interrupt when a signal level transition occurs.

When the correct settings have been made, the application software polls for the respective interrupts to determine the presence of sync input signals, as follows:

— Indirect Vsync input detection

  Check for the occurrence of a timer 0 capture interrupt (IRQ0) or a Vsync detection interrupt (IRQ5).

— Indirect Hsync input detection

  Check for the occurrence of a timer 1 overflow interrupt (IRQ1).

— Indirect Csync input detection (SOG)

  Check for the occurrence of a timer 1 overflow interrupt (IRQ1).

DETECTING SYNC SIGNAL POLARITY

The S3C8618/C8615 sync processor lets you detect the polarity of Vsync or Hsync signals in two ways: 1) using a software-intensive direct detection method, or 2) by an indirect detection method which uses the sync processor hardware.

Polarity detection is not required for Csync because the signal which passes through the Csync-I pin (SOG) is always positive polarity.

Direct Method for Polarity Detection

To use the direct polarity detection method, you read the status of the vertical or horizontal sync signals at the Vsync-I and Hsync-I pins, respectively. You then count the number of signal transitions that occur over a given number of frames, or during a given time interval, and derive an average value. If the average value is “1”, the signal’s polarity is negative. If it is “0”, the polarity is positive.

Another way to determine the polarity is to repeat a counting operation several times over a given time interval. You then compare the totals derived from the series of count operations to determine which one is greater. If the larger count value is for “1s”, the signal polarity is negative. If more “0s” were counted, the polarity is positive.

Indirect Method for Vsync Polarity Detection

To indirectly detect positive or negative Vsync polarity, you use timer 0 to capture the internal time between the rising and falling edges of the Vsync input signal.

To enable indirect Vsync polarity detection, you set SYNCON2.7 to “1”. This selects the Vsync output path as input for the timer 0 capture function. Timer 0 must also be set to operate in capture mode.

When the series of capture operations has been completed, you can read the timer 0 counter value to determine the signal’s polarity.

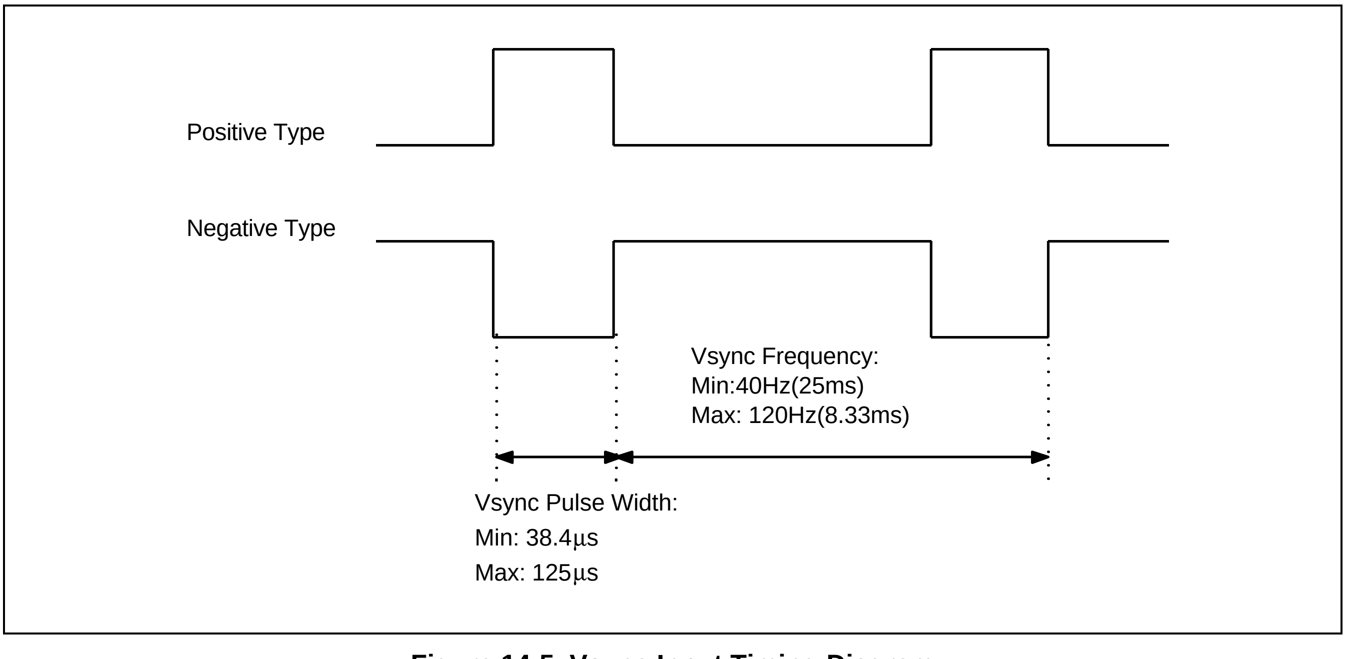


Figure 14-5. Vsync Input Timing Diagram

Indirect Hsync Polarity Detection

To indirectly detect Hsync polarity, you use the sync processor's 5-bit up/down counter. An overflow or borrow does not occur when the counter values reaches 11111B or 00000B, respectively.

To enable Hsync polarity detection, you first write 11111B or 00000B to bits 0–4 of the SYNCON0 register. Whenever a transition in the signal level at Hsync-I is detected (either a rising or a falling edge), the 5-bit counter value in SYNCON0 is updated. The counter value increments when the level of the sync signal goes High and decrements when it goes Low.

If the captured value is 00000B, the polarity is positive because the counter remains at this value after a borrow occurs. Otherwise, the capture value will be other than 00000B and indicate a negative polarity.

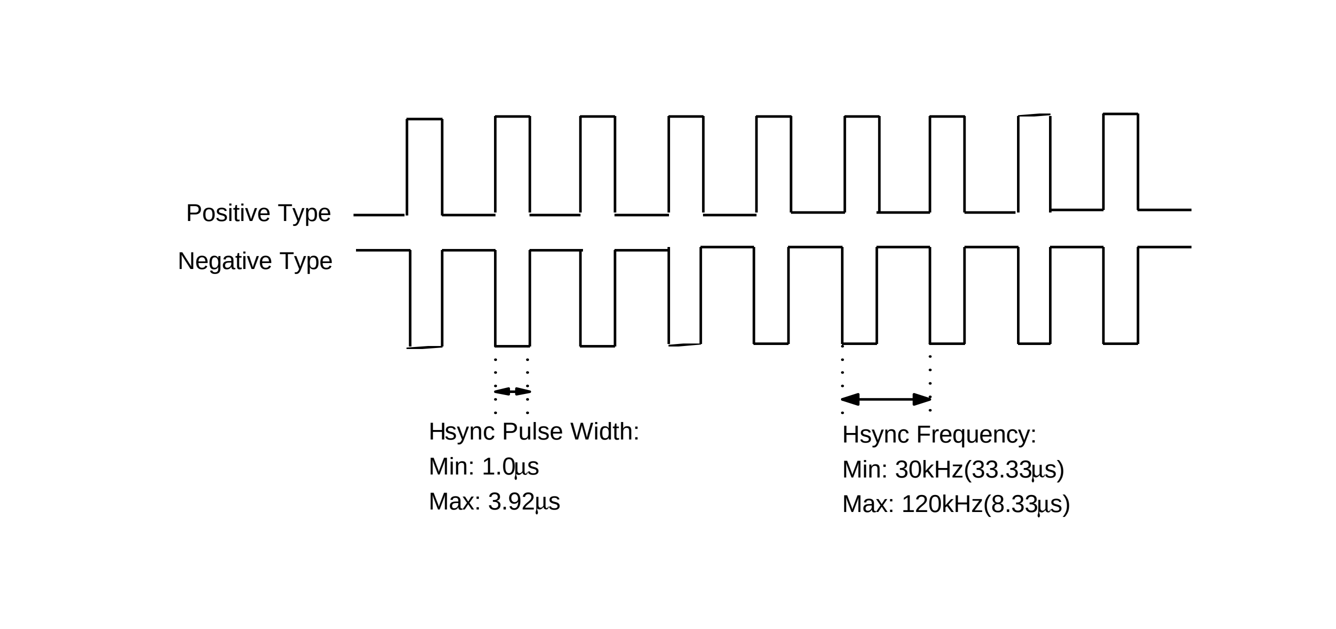


Figure 14-6. Hsync Input Timing Diagram

EXTRACTING VSYNC OUTPUT

When the Vsync input at Hsync-I or Csync-I (P2.7) also contains Hsync signals, you must extract the Vsync component from the Hsync (or Csync) input. To do this, you use the 5-bit up/down counter.

To extract the Vsync component of the input signal, you first set the 5-bit up/down counter to operate in compare mode (SYNCON0.5 = "1"). Vsync output is enabled only when the minimum or maximum threshold value is reached.

During vertical blanking, the counter decreases until it reaches a minimum value while the Hsync-I or Csync-I signal level is negative. Or, the counter value increases until it reaches a maximum value while the Hsync-I or Csync-I signal level is positive (no overflow occurs).

The Vsync detection interrupt (IRQ5) can be enabled to verify that the Vsync signal has been extracted successfully from the mixed input signal.

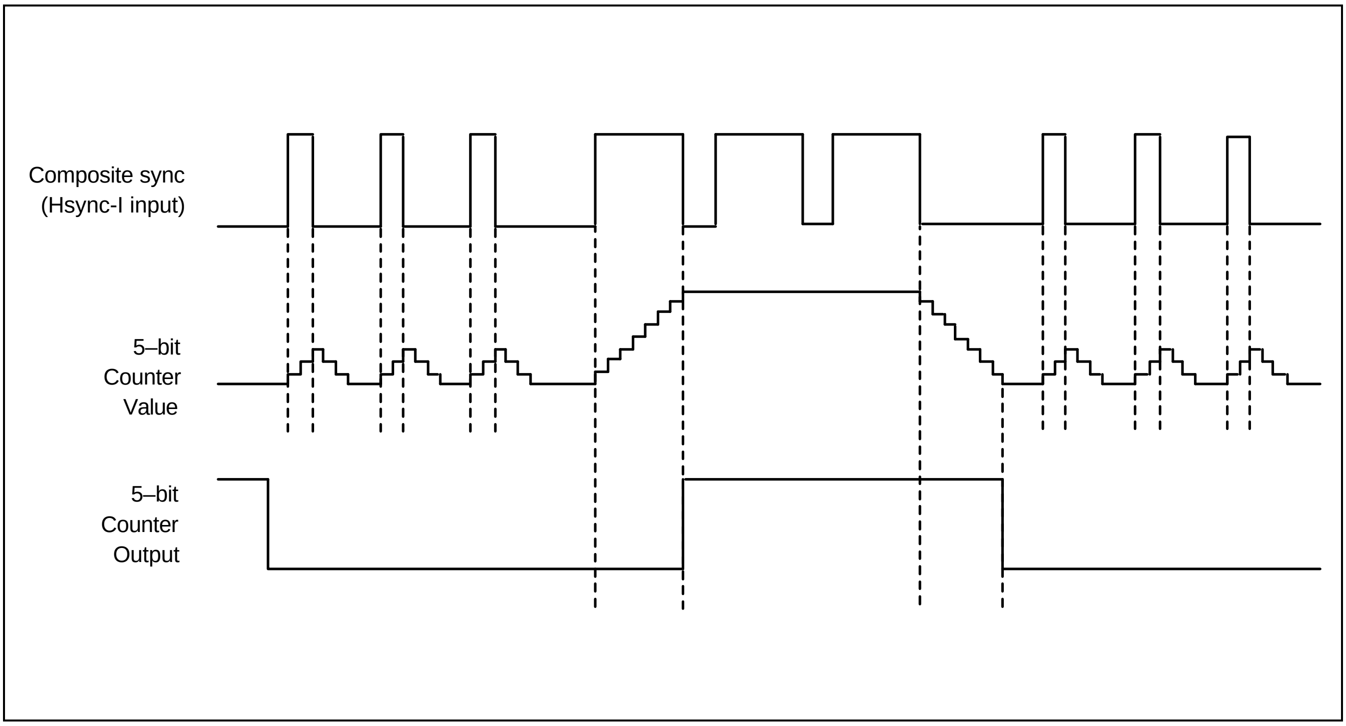


Figure 14-7. Vsync Extraction Using an Up/Down Counter

CLAMP SIGNAL OUTPUT

Clamp signal output (Clamp-O) must be synchronized with Hsync output. The Clamp-O signal can be transmitted to a vertically or horizontally driven integrated circuit to provide a pedestal level for image signals with programmable pulse width.

The Clamp signal is output on the “front porch” of an Hsync signal (NO SOG condition) or on the “back porch” of an Csync signal (SOG condition).

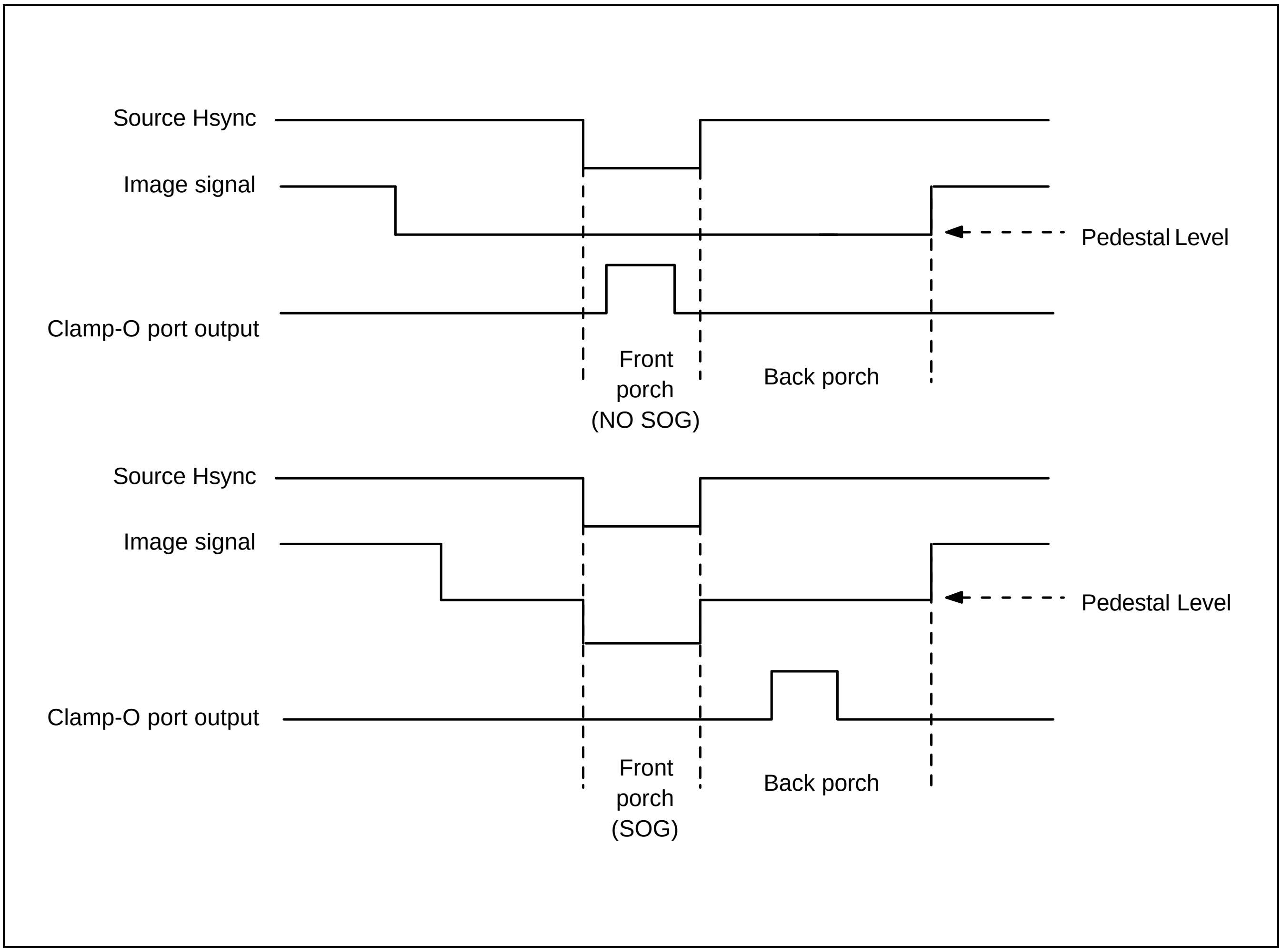


Figure 14-8. Clamp-O Signal (SOG and NO SOG Condition)

DIFFERENTIATING SOG FROM NO SOG

The pulse width at the Csync-I pin is different in SOG and NO SOG conditions. In a SOG condition, the pulse width at Csync-I and Hsync-I is identical. If a NO SOG condition exists, Csync-I has a wider pulse width than Hsync-I because the Csync-I pulse is truncated at the base of the pedestal level (see Figure 14-10).

To differentiate the Csync pulse, you must delay the Csync-I pulse for about 150 ns and then compare its phase with that of the Hsync-I pin signal.

To indicate a SOG condition, comparator logic for Hsync and Csync sets the SYNCON1.7 flag to “1” whenever Csync status differs from Hsync status more than 32 times at the rising edge of Hsync-I. To perform the comparison, first detect the polarity of the Hsync-I signal. Then configure the pin for positive output. (Csync-I is always positive and requires no special settings.) To recognize the SOG condition, you can poll the SYNCON1.7 status flag to detect when it is set to “1”.

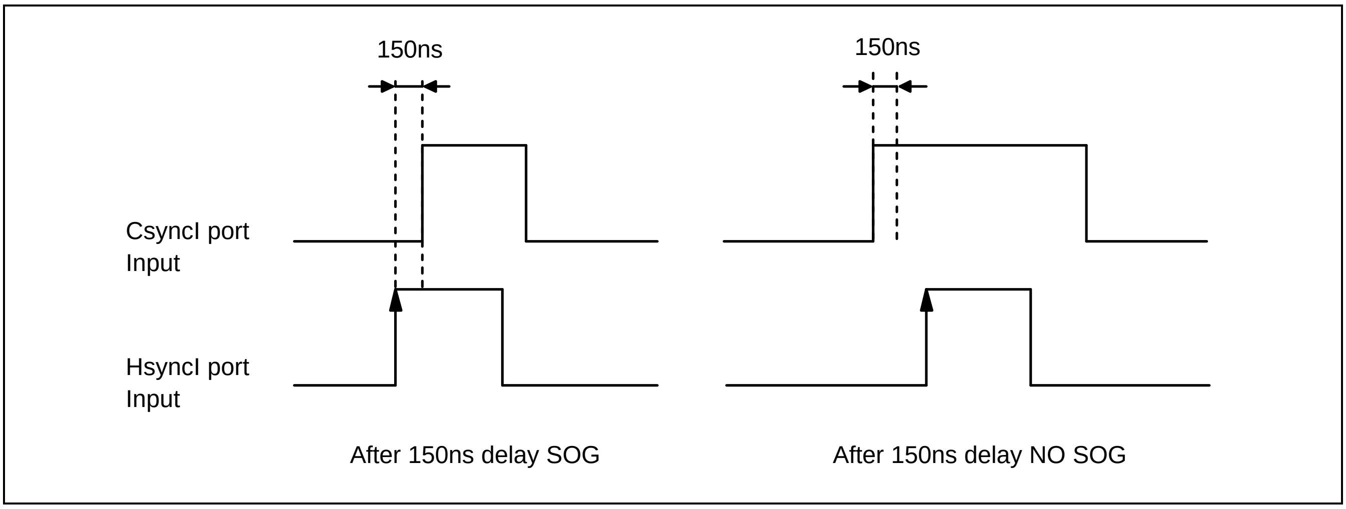


Figure 14-9. Sync Input at the Hsync-I and Csync-I Pins

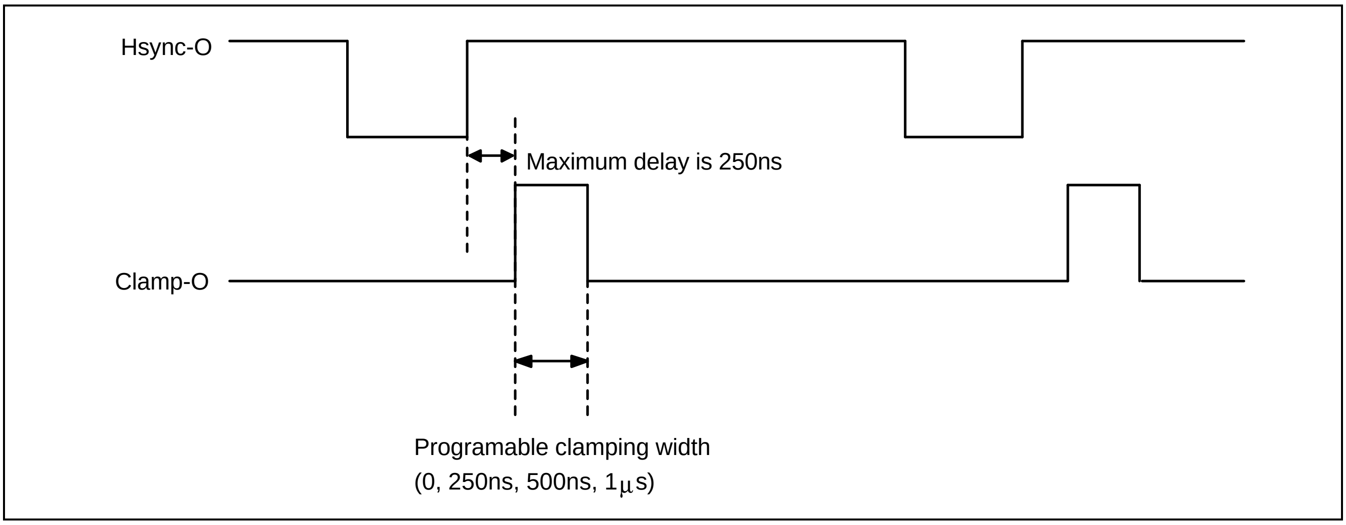


Figure 14-10. Clamp-O Signal Delay Timing

☞ PROGRAMMING TIP — Programming the Sync Processor

This example shows how to program the sync processor to sample specifications. The sample program performs the following actions:

Confirm the presence of sync signal input

Detect the polarity of Hsync or Vsync signal input

```
SYNCPRSS:
        TM      SyncP_FGR,#01<<HSyncFin_FG          ;  Set every 10 ms
        JR      Z, ChkPresnVsync                    ;  Checking presence of Hsync signal
        AND     SyncP_FGR, #0FFH-(01<<HSyncFin_FG)
                                                    ;  Every 10 ms
        CALL    TimeBase                            ;  Time count (100 ms, 1 sec, 2 sec, 3 sec …)
        CALL    ChkHNosyncRange                     ;  Compare 10 kHz (Check no sync range)
        JR      NC, ChkHsyncData
        CALL    VideoMute

ChkHsyncData:
        CALL    ChkPolaChange                       ;  Check Polarity data
        JP      C, VideoMute
        CALL    ChkHsyncChange                      ;  Check Hsync signal data
        JP      C, VideoMute

ChkPresnVsync:
        CP      NoVTime, #30                        ;  If NoVTime > 30 ms (under 33 Hz)⇒Mute
        JR      ULE, ChkVRange
        CALL    VideoMute                           ;  Mute
        TM      SyncP_FGR, #01 << Hnosync_FG        ;  No Hsync and no Vsync ?
        JR      Z, ChkSyncSrc
        OR      SyncP_FGR, #01 << NoSync_FG
```

☞ PROGRAMMING TIP — Programming the Sync Processor (Continued)

```
ChkSyncSrc:
        CALL    ChkSyncSource                  ;  Check Sync signal after Sync source changing
        CALL    ClrSyncData                    ;  Clear all Sync Control register
        RET

ChkVRange:                                     ;  Checking presence of Vsync signal
        TM      SyncP_FGR, #01 << VSyncFin_FG
        JR      Z, ChkHVPrtn
        AND     SyncP_FGR, #0FFH–(01 << VsyncFin_FG)

CountVfreg:
        CALL    ComputeVfreq                   ;  Calculate Vsync frequency
        CALL    ChkHVRange                     ;  Check range of valid sync signal
        JP      C, VideoMute                   ;  Range over !
        CALL    ChkVsyncChange                 ;  Check Vsync data
        JP      C, VideoMute                   ;  Stable sync signal input
        TM      Mute_FGR, #01 << MuteRelse_FG
        JR      Z, ChkMuteTime
```

(Continued on next page)

☞ PROGRAMMING TIP — Programming the Sync Processor (Continued)

```
ChkHVPrtn:
            RET


ChkMuteTime:
            •
            •
            •


LoadDAC:
            LD      SYNCON2, #10000000B             ;   Normal sync operation
            CALL    S_Correct                       ;   S-correction
            CALL    AdjHmodeSize                    ;   Adjust Hsync mode size
            CALL    AdjModeSize                     ;   Adjust mode size
            OR      EepRom_FGR, #01H << EepDataRd_FG
                                                    ;   Load PWM data from EEPROM
            OR      P0, #01 << Muteport             ;   Mute port release


MuteDelay:
            OR      Mute_FGR, #0FFH – (01 << MuteDelay_FG)
            RET


MuteRelease:
            LD      SYNCON2, #10000000B             ;   Normal sync operation
            AND     Mute_FGR, #0FFH – (01 << MuteDelay_FG)
            AND     Mute_FGR, #0FFH – (01 << Vmute_FG)
            OR      Mute_FGR, #01 << MuteRelease_FG
```

☞ PROGRAMMING TIP — Programming the Sync Processor (Cont.)

```
            SB1                                          ;    Return contrast/brightness value
            •
            •
            SB0
            RET

VideoMute:
            CALL    PwmInMute                            ;    Brightness, Contrast ← Minimum value
            AND     SYNCON1, #11111001B                  ;    Pseudo Sync = only positive Pol.
            LD      SYNCON2, #10001100B                  ;    74.1 Hz, 47.6 kHz (Pseudo sync)
            OR      Mute_FGR, #01 << Vmute_FG            ;    Vmute_FG ← 1
            •
            •
            RET
```

15 DDC and IIC-BUS INTERFACE

OVERVIEW

The S3C8618/C8615 microcontrollers support a multi-master IIC-bus serial interface and Display Data Channel (DDC).

A dedicated serial data line (SDA) and a serial clock line (SCL) carry information between bus masters and peripheral devices which are connected to the IIC-bus. The SDA and SCL lines are bi-directional.

The DDC1 protocol uses vertical sync input at the Vsync-I pin as its clock source (VCLK). VCLK is input-only. DDC1 is implemented physically using VCLK input and SDA output.

Protocols for the DDC2B and DDC2AB modes are supported in hardware by multi-master IIC-bus logic and in software by the EDID (Extended Display Identification) and VDIF (Video Display Interface) formats.

In multi-master IIC-bus mode, multiple S3C8618/C8615 microcontrollers can receive or transmit serial data to or from slave devices. The master S3C8618/C8615 which initiates a data transfer over the IIC-bus is responsible for terminating the transfer. Standard bus arbitration functions are supported.

To control multi-master IIC-bus operations, you write values to the following registers:

— IIC-bus control register, ICCR
— IIC-bus control/status register, ICSR
— IIC-bus Tx/Rx data shift register, IDSR
— IIC-bus address register, IAR

When the IIC-bus is free, the SDA and SCL lines are both at High level. A High-to-Low transition of SDA initiates a Start condition. A Low-to-High transition of SDA while SCL remains steady at High level initiates a Stop condition.

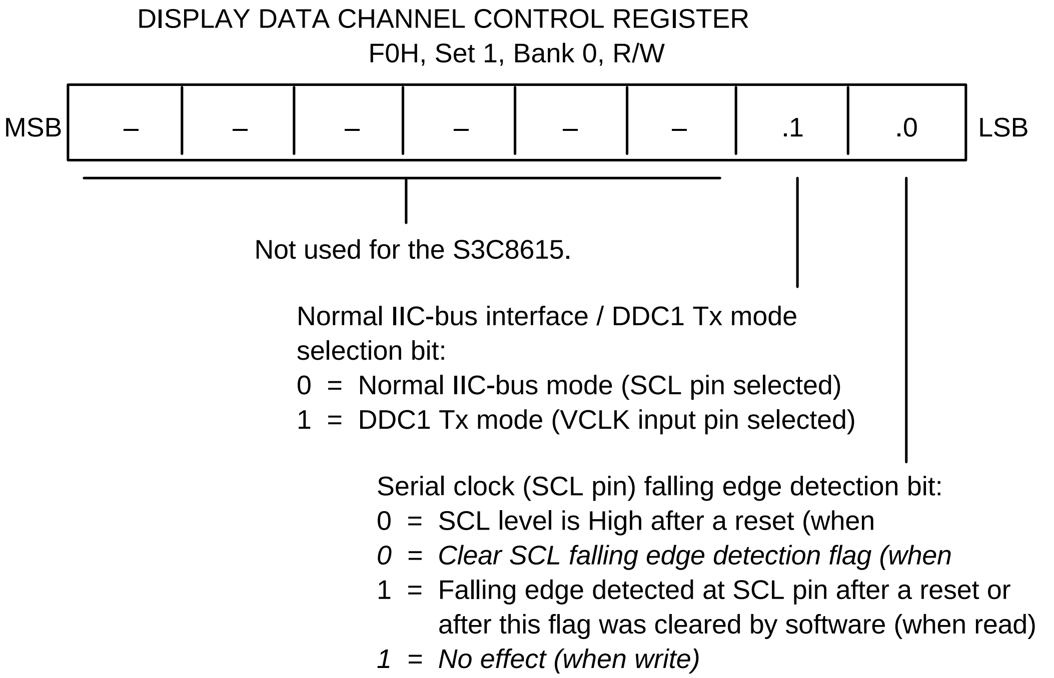
Start and Stop conditions are always generated by the bus master. A 7-bit address value in the first data byte that is put onto the bus after the Start condition is initiated determines which slave device the bus master selects. The 8th bit determines the direction of the transfer (read or write).

Every data byte that is put onto the SDA line must total eight bits. The number of bytes which can be sent or received per bus transfer operation is unlimited. Data is always sent most-significant bit (MSB) first and every byte must be immediately followed by an acknowledge (ACK) bit.

DDC CONTROL REGISTER (DDCCON)

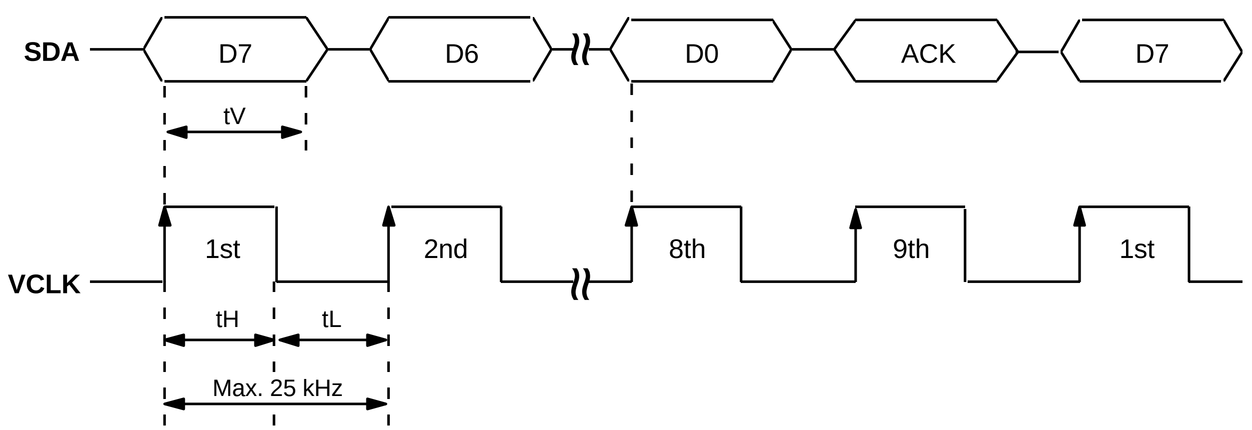
The display data channel control register, DDCCON, is located in set 1, bank 0, at address F0H. It is read/write addressable. Only two bits are mapped in this register: DDCCON.0 and DDCCON.1.

The DDCCON.0 setting lets you detect falling edges at the serial clock pin, SCL. The DDCCON.1 setting lets you select normal IIC-bus interface mode or DDC1 transmit mode (see Figure 15-1).



NOTES:
1. The IIC-bus physically supports the DDC2B and DDC2AB (data access) modes by software in EDID and VDIF formats.
2. DDCCON.1 is automatically set to "1" if a falling signal edge is detected at the SCL pin following a hardware reset or after the flag has been cleared by software.

Figure 15-1. Display Data Channel Control Register (DDCCON)



where,

tV	=	Data valid time (min. 30 µs)
tH	=	VCLK High pulse width (min. 20 µs)
tL	=	VCLK Low pulse width (min. 20 µs)
		(Max. VCLK input frequency = 25 kHz)

NOTE: In DDC1 mode, each byte of data is output most-significant bit (MSB) first.

Figure 15-2. DDC1 Mode Timing Diagram (One-Byte Transfer)

MULTI-MASTER IIC-BUS CONTROL REGISTER (ICCR)

The multi-master IIC-bus control register, ICCR, is located in set 1, bank 1, at address F0H. It is read/write addressable. ICCR settings control the following IIC-bus functions:

— CPU acknowledge signal (ACK) enable or suppress
— IIC-bus clock source selection (f_{OSC} /16 or f_{OSC} /512)
— Transmit/receive interrupt enable or disable
— Transmit/receive interrupt pending control
— 4-bit prescaler for the serial transmit clock (SCL)

In the S3C8618/C8615 interrupt structure, the IIC-bus Tx/Rx interrupt is assigned level IRQ2, vector F8H. To enable this interrupt, you set ICCR.5 to “1”. Program software can then poll the IIC-bus Tx/Rx interrupt pending bit (ICCR.4) to detect IIC-bus receive or transmit requests. When the CPU acknowledges the interrupt request from the IIC-bus, the interrupt service routine must clear the interrupt pending condition by writing a “0” to ICCR.4.

The SCL frequency is determined by the IIC-bus clock source selection (f_{OSC} /16 or f_{OSC} /512) and the 4-bit prescaler value in the ICCR register (see Figure 15-3).

MULTI-MASTER IIC-BUS CONTROL REGISTER (ICCR)
E9H, Set 1, Bank 1, R/W

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

IIC-bus acknowledge (ACK) enable bit:
0 = Disable ACK generation
1 = Enable ACK generation

IIC-bus clock source selection bit:
0 = fOSC/16
1 = fOSC/512

IIC-bus Tx/Rx interrupt enable/disable bit:
0 = Disable interrupt
1 = Enable inter rupt

IIC-bus Tx/Rx interrupt pending flag:
0 = No interrupt pending (when read)
0 = Clear pending condition (when write)
1 = Interrupt is pending (when read)
1 = No effect (when write)

IIC-bus transmit (Tx) clock prescaler:

The IIC-bus transmit clock (SCL) frequency is determined by the clock source selection (ICCR.6) and this 4-bit prescaler value, according to the following formula:

Tx clock (SCL) = IICCLK / (ICCR.3–ICCR.0) + 1

where IICCLK = fOSC/16 (ICCR.6 = “0”) or
IICCLK = fOSC/512 (ICCR.6 = “1”)

NOTE: A IIC-bus interrupt occurs 1) when a 1-byte transmit or receive operation is terminated, 2) when a general call or a slave address match occurs, or 3) if bus arbitration fails.

Figure 15-3. Multi-Master IIC-Bus Control Register (ICCR)

Table 15-1. Sample Timing Calculations for the IIC-Bus Transmit Clock (SCL)

ICCR.3–ICCR.0 Value	IICLK (ICCR.3–ICCR.0 Settings + 1)	(f_{OSC} = 8 MHz) ICCR.6 = 0 (f_{OSC} /16) IICLK = 500 kHz	(f_{OSC} = 8 MHz) ICCR.6 = 1 (f_{OSC} /512) IICLK = 15.625 kHz
0000	IICLK/1	500 kHz	15.625 kHz
0001	IICLK/2	250 kHz	7.1825 kHz
0010	IICLK/3	116.7 kHz	5.2038 kHz
0011	IICLK/4	125 kHz	3.9063 kHz
0100	IICLK/5	100 kHz	3.1250 kHz
0101	IICLK/6	83.3 kHz	2.6042 kHz
0110	IICLK/7	71.4 kHz	2.2321 kHz
0111	IICLK/8	62.5 kHz	1.9531 kHz
1000	IICLK/9	55.6 kHz	1.7361 kHz
1001	IICLK/10	50 kHz	1.5625 kHz
1010	IICLK/11	45.5 kHz	1.4205 kHz
1011	IICLK/12	41.7 kHz	1.3021 kHz
1100	IICLK/13	38.5 kHz	1.2019 kHz
1101	IICLK/14	35.7 kHz	1.1160 kHz
1110	IICLK/15	33.3 kHz	1.0417 kHz
1111	IICLK/16	31.25 kHz	0.9766 kHz

MULTI-MASTER IIC-BUS CONTROL/STATUS REGISTER (ICSR)

The multi-master IIC-bus control/status register, ICSR, is located in set 1, bank 1, at address EAH. Four bits in this register, ICSR.3–ICSR.0, are read-only status flags.

ICSR register settings are used to control or monitor the following IIC-bus functions (see Figure 15-4):

— Master/slave transmit or receive mode selection

— IIC-bus busy status flag

— Serial output enable/disable

— Failed bus arbitration procedure status flag

— Slave address/address register match or general call received status flag

— Slave address 00000000B (general call) received status flag

— Last received bit status flag (not ACK = “1”, ACK = “0”)

MULTI-MASTER IIC-BUS CONTROL/STATUS REGISTER (ICSR)
EAH, Set 1, Bank 1, R/W

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

.7–.6: IIC-bus master/slave Tx/Rx mode selection bits:
00 = Slave receive mode (Default mode)
01 = Slave transmit mode
10 = Master receive mode
11 = Master transmit mode

.5: IIC-bus busy signal status bit:
0 = IIC-bus not busy (when read), IIC interface STOP signal generation (when write)
1 = IIC-bus is busy (when read), IIC interface START signal generation (when write)

.4: IIC-bus serial output enable/disable bit:
0 = Disable serial Tx/Rx
1 = Enable serial Tx/Rx

.3: IIC-bus arbitration procedure status flag:
0 = Bus arbitration status okay
1 = Bus arbitration failed during serial I/O

.2: IIC-bus address-as-slave status flag:
0 = START/STOP condition was generated
1 = Received slave address matches the address value in the IAR

.1: IIC-bus address zero status flag:
0 = Start/stop condition was generated
1 = Received slave address is ‘00000000B’ (general call)

.0: IIC-bus last-received bit status flag:
0 = Last-received bit is “0” (ACK was received)
1 = Last-receive bit is “1” (ACK was not received)

NOTE: ICSR.3 is automatically set to “1” when a bus arbitration procedure fails over serial I/O interface, while the IIC-bus is set to ‘master transmit mode’ (ICSR.7 and ICSR.6 = ‘11B’). If slave transmit or receive mode is selected, ICSR.3 is automatically set to “1” if data is written to the shift register (IDSR) when the busy signal bit, ICSR.5 is “1”.

Figure 15-4. Multi-Master IIC-Bus Control/Status Register (ICSR)

MULTI-MASTER IIC-BUS TRANSMIT/RECEIVE DATA SHIFT REGISTER (IDSR)

The IIC-bus data shift register, IDSR, is located in set 1, bank 1, at address ECH. In a transmit operation, data that is written to the IDSR is transmitted serially, MSB first. (For receive operations, the input data is written into the IDSR register LSB first.)

The ICSR.4 setting enables or disables serial transmit/receive operations. When ICSR.4 = “1”, data can be written to the shift register. The IIC-bus shift register can, however, be read at any time, regardless of the current ICSR.4 setting.

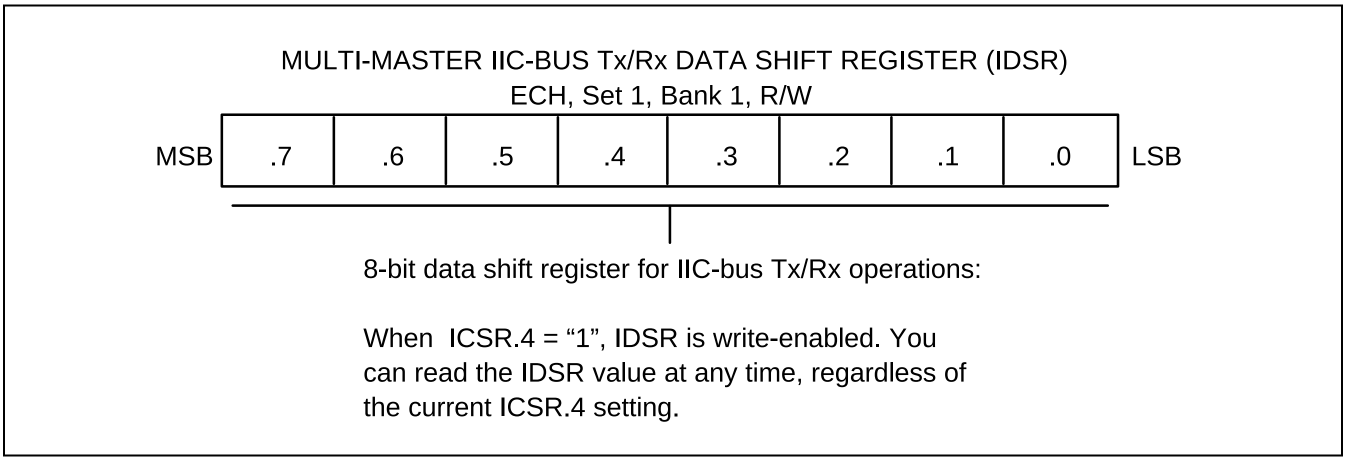


Figure 15-5. Multi-Master IIC-Bus Tx/Rx Data Shift Register (IDSR)

MULTI-MASTER IIC-BUS ADDRESS REGISTER (IAR)

The address register for the IIC-bus interface, IAR, is located in set 1, bank 1, at address EBH. It is used to store a latched 7-bit slave address. This address is mapped to IAR.7–IAR.1; bit 0 is not used (see Figure 15-6).

The latched slave address is compared to the next received slave address. If a match condition is detected, and if the latched value is 00000000B, a general call status is detected.

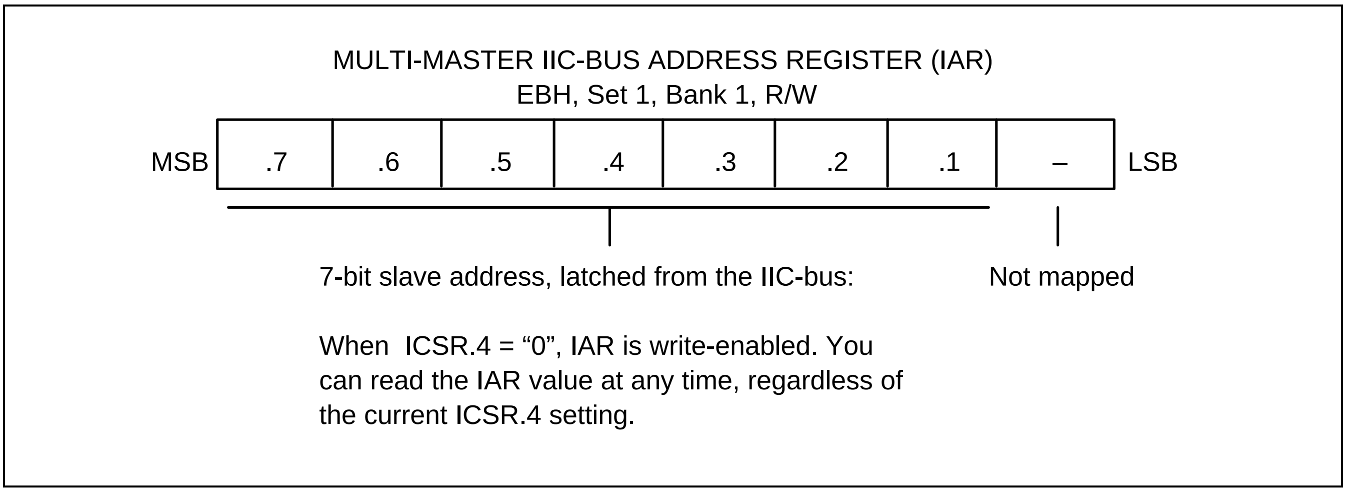
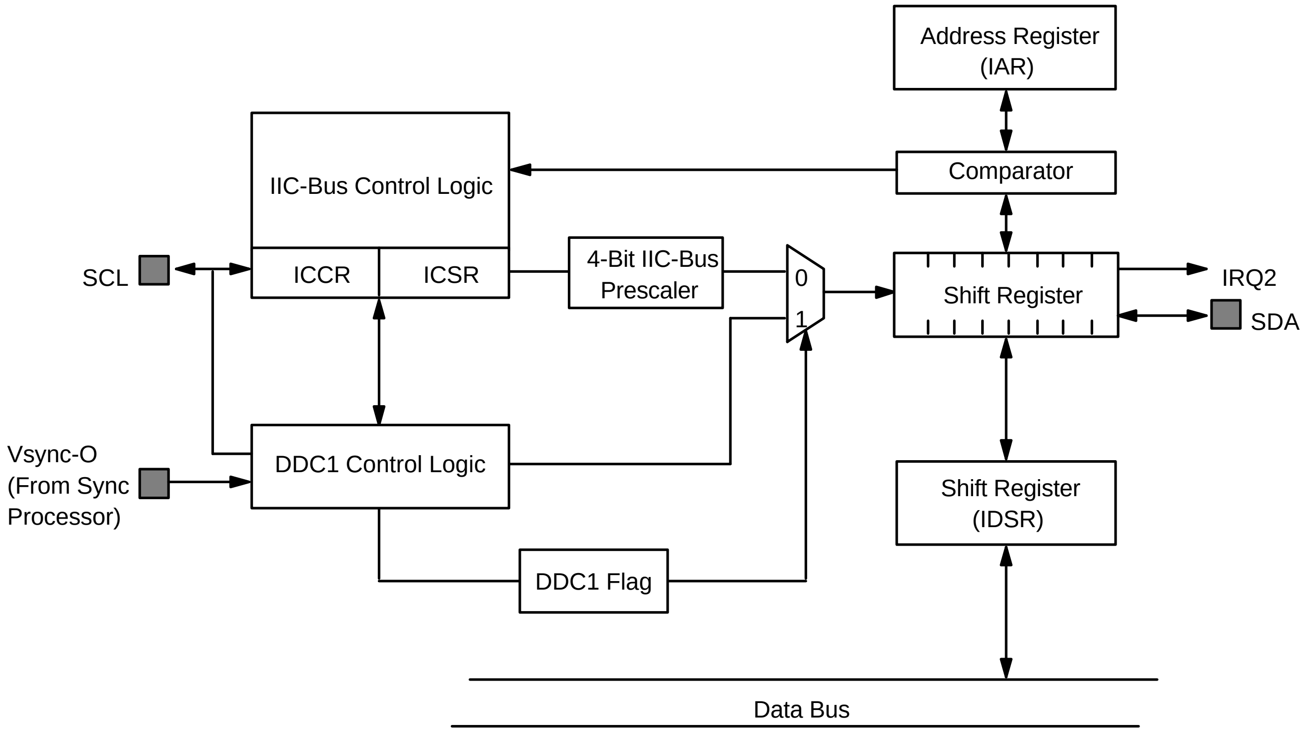


Figure 15-6. Multi-Master IIC-Bus Address Register (IAR)



NOTE: The IIC-bus interrupt (IRQ2) is generated when a 1-byte receive or transmit operation is terminated before the shift operation has been completed.

Figure 15-7. DDC and IIC-Bus Block Diagram

THE IIC-BUS INTERFACE

The S3C8618/C8615 IIC-bus interface has four operating modes:

— Master transmitter mode

— Master receive mode

— Slave transmitter mode

— Slave receive mode

Functional relationships between these operating modes are described below.

START AND STOP CONDITIONS

When the IIC-bus interface is inactive, it is in slave mode. The interface is therefore always in slave mode when a start condition is detected on the SDA line. (A start condition is a High-to-Low transition of the SDA line while the clock signal, SCL, is High level.) When the interface enters master mode, it initiates a data transfer and generates the SCL signal.

A start condition initiates a one-byte serial data transfer over the SDA line and a stop condition ends the transfer. (A stop condition is a Low-to-High transition of the SDA line while SCL is High level.) Start and stop conditions are always generated by the master. The IIC-bus is “busy” when a start condition is generated. A few clocks after a stop condition is generated, the IIC-bus is again “free”.

When a master initiates a start condition, it sends its slave address onto the bus. The address byte consists of a 7-bit address and a 1-bit transfer direction indicator (that is, write or read). If bit 8 is “0”, a transmit operation (write) is indicated; if bit 8 is “1”, a request for data (read) is indicated.

The master ends the indicated transfer operation by transmitting a stop condition. If the master wants to continue sending data over the bus, it can the generate another start condition and another slave address. In this way, read-write operations can be performed in various formats.

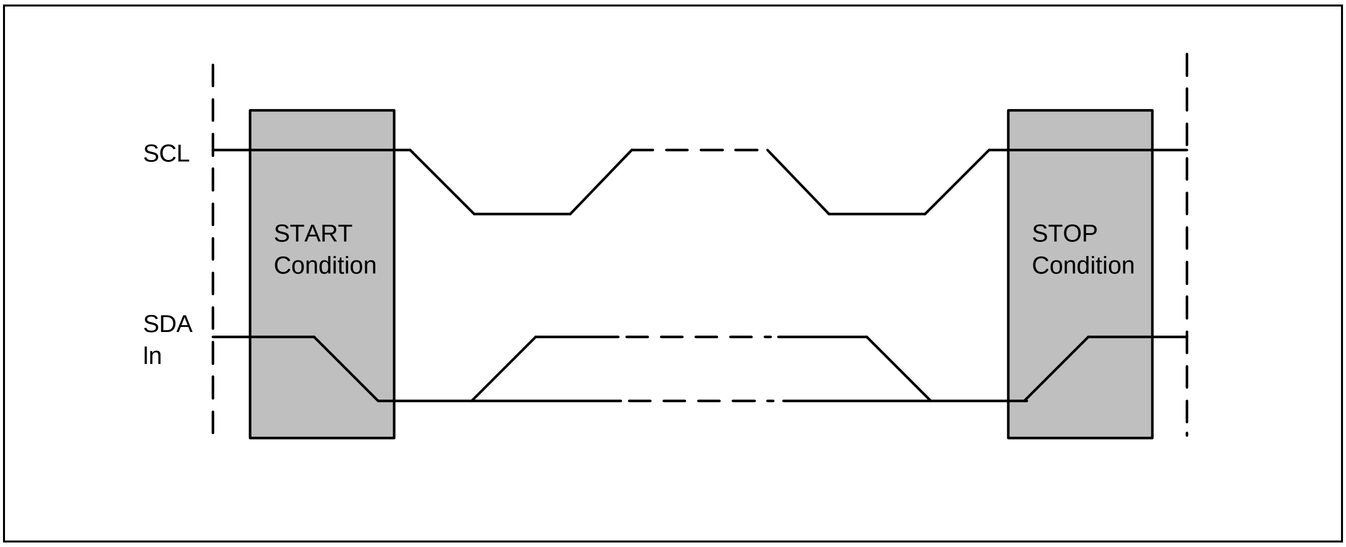


Figure 15-8. Start and Stop Conditions

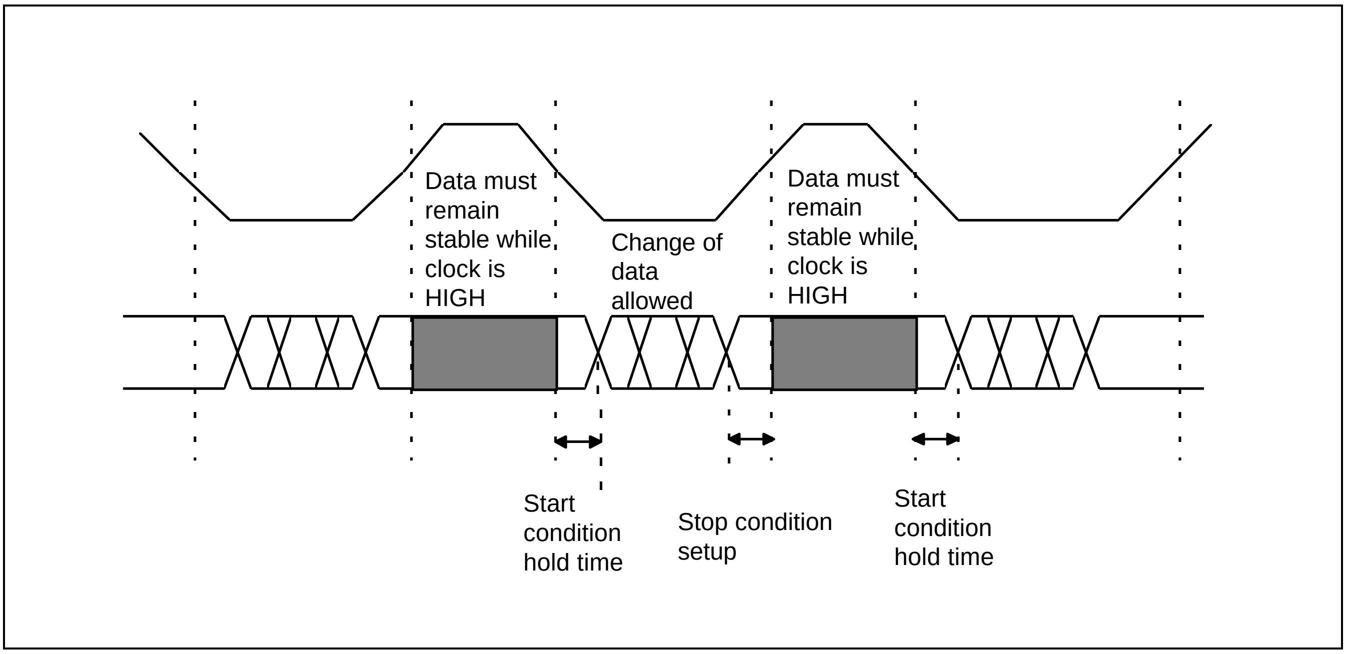
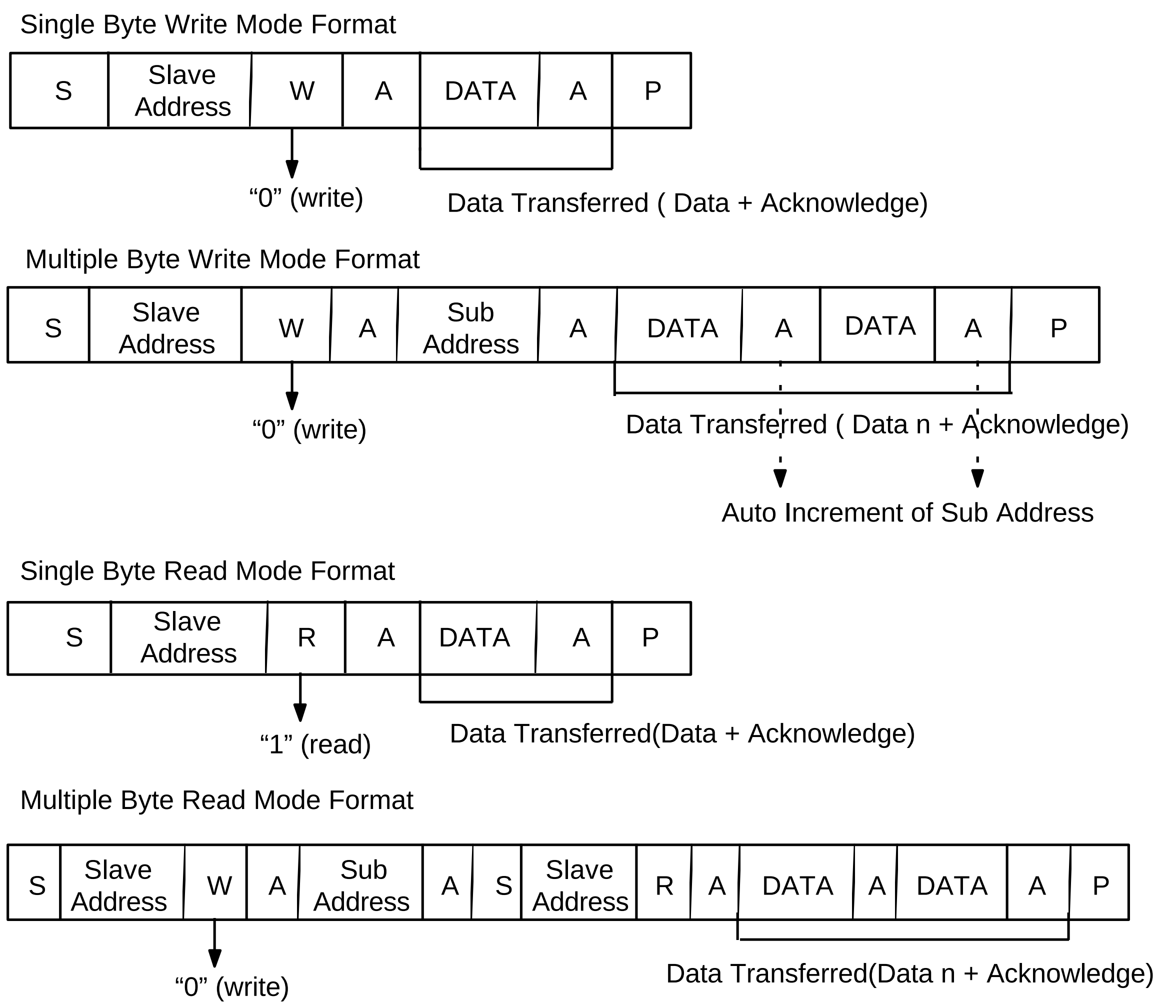


Figure 15-9. Input Data Protocol

DATA TRANSFER FORMATS

Every byte put on the SDA line must be eight bits in length. The number of bytes which can be transmitted per transfer is unlimited. The first byte following a start condition is the address byte. This address byte is transmitted by the master when the IIC-bus is operating in master mode. Each byte must be followed by an acknowledge (ACK) bit. Serial data and addresses are always sent MSB first.



NOTES
1. S: start, A: acknowledge, P: stop
2. The "Sub Address" indicates the internal address of the slave device.

Figure 15-10. IIC-Bus Interface Data Formats

ACK SIGNAL TRANSMISSION

To complete a one-byte transfer operation, the receiver must send an ACK bit to the transmitter. The ACK pulse occurs at the ninth clock of the SCL line (eight clocks are required to complete the one-byte transfer). The clock pulse required for the transmission of the ACK bit is always generated by the master.

The transmitter releases the SDA line (that is, it sends the SDA line High) when the ACK clock pulse is received. The receiver must drive the SDA line Low during the ACK clock pulse so that SDA is Low during the High period of the ninth SCL pulse.

The ACK bit transmit function can be enabled and disabled by software (ICCR.7). However, the ACK pulse on the ninth clock of SCL is required to complete a one-byte data transfer operation.

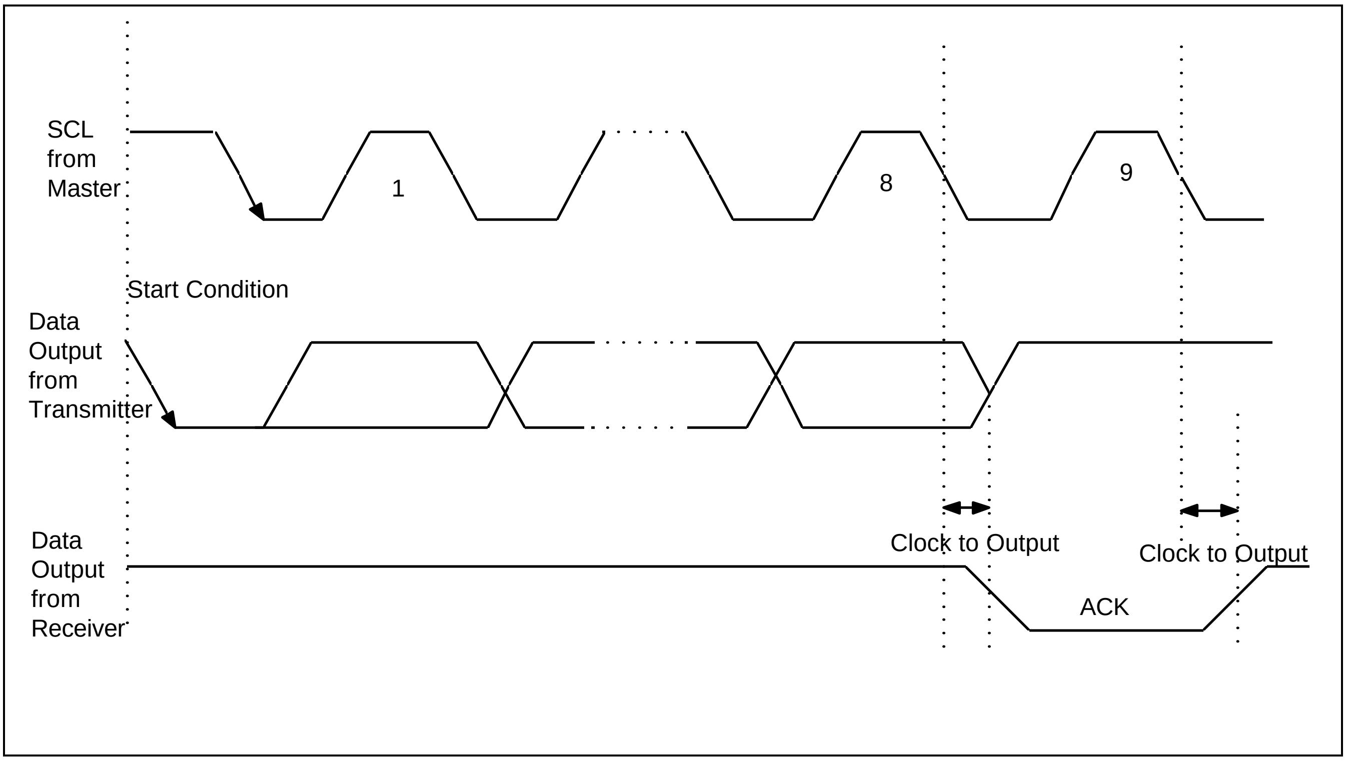


Figure 15-11. Acknowledge Response from Receiver

READ-WRITE OPERATIONS

When operating in transmitter mode, the IIC-bus interface interrupt routine waits for the master (the S3C8618/C8615) to write a data byte into the IIC-bus data shift register (IDSR). To do this, it holds the SCL line Low prior to transmission.

In receive mode, the IIC-bus interface waits for the master to read the byte from the IIC-bus data shift register (IDSR). It does this by holding the SCL line Low following the complete reception of a data byte.

BUS ARBITRATION PROCEDURES

Arbitration takes place on the SDA line to prevent contention on the bus between two masters. If a master with a SDA High level detects another master with an SDA active Low level, it will not initiate a data transfer because the current level on the bus does not correspond to its own. The master which loses the arbitration can generate SCL pulses only until the end of the last-transmitted data byte. The arbitration procedure can continue while data continues to be transferred over the bus.

The first stage of arbitration is the comparison of address bits. If a master loses the arbitration during the addressing stage of a data transfer, it is possible that the master which won the arbitration is attempting to address the master which lost. In this case, the losing master must immediately switch to slave receiver mode.

ABORT CONDITIONS

If a slave receiver does not acknowledge the slave address, it must hold the level of the SDA line High. This signals the master to generate a stop condition and to abort the transfer.

If a master receiver is involved in the aborted transfer, it must also signal the end of the slave transmit operation. It does this by not generating an ACK after the last data byte received from the slave. The slave transmitter must then release the SDA to allow a master to generate a stop condition.

CONFIGURING THE IIC-BUS

To control the frequency of the serial clock (SCL), you program the 4-bit prescaler value in the ICCR register. The IIC-bus interface address is stored in IIC-bus address register, IAR. (By default, the IIC-bus interface address is an unknown value.)

THE DDC INTERFACE

OVERVIEW

To enable data communications based on the display data channel (DDC) protocol, the S3C8618/C8615 microcontrollers have a DDC interface. The DDC interface supports the DDC1, DDC2, and DDC2AB (Address Bus) modes. DDC2B mode may be not perfectly supported by IIC specifications because the memory (RAM) is somewhat insufficient.

The S3C8618/C8615 are connected to the DDC bus by one data pin (SDA) and two clock pins (SCL, Vsync-O from sync processor). The SDA and SCL pins are connected directly to the IIC-bus.

COMPARISON OF DDC1 AND DDC2 MODE

DDC1 is a uni-directional data channel between a display device (monitor) and a host computer. To enable the host to correctly configure the monitor, the monitor transmits basic information about its operating characteristics to the host over the DDC1 channel during a system setup operation.

DDC2 mode is a bi-directional data channel for communications between a display device and a host computer, and is based on the IIC-bus protocol.

The DDC2 mode is based on the IIC-bus protocol which uses nine clock pulses to transfer a data byte, the same data format should be used in DDC1 mode to conserve the limited space available for software code in the monitor. Also, EEPROM integrated circuits that is based on the IIC-bus protocol also use the 9-bit data format.

In DDC2 mode, the 9th bit (ACK) is used for all data transfers and in DDC1 mode, the ACK bit is ignored. For monitors that support both DDC1 and DDC2 modes, the DDC2 channel always has the highest priority.

In DDC1 mode, the speed of data transfers is determined by the VCLK signal frequency. When data is detected on the IIC-bus, the VCLK frequency can then be increased to a maximum rate of 25 kHz (for S3C8618/C8615 applications). If data cannot be received at the normal video frame frequency, the monitor display may be of old type which is a older display without DDC capabilities. Please note that older monitor types can be damaged if a higher than normal vertical frequency (VCLK) is applied to them.

FUNCTIONAL DESCRIPTION FOR DDC1/DDC2 MODE

During startup, a DDC1-capable host system will try to read the DDC1 data stream that is sent from the monitor. If there are repeated read errors, the host system will assume that the monitor is of an older type which does not support DDC. The host may then attempt to read the monitor ID pins using earlier ID schemes.

After the host and monitor are powered-on, the host outputs a VCLK frequency at Vsync-I while holding the SCL line High. When it detects SCL High, the monitor sends the DDC data to the host in DDC1 mode. This transmission is synchronized with the clock input at Vsync-O from sync processor. To receive the DDC1 data stream, the host must hold the SCL level High.

Initially, the VCLK frequency should be equal to the Vsync frequency. When data has been put onto the bus, the frequency can be increased to 25 kHz (maximum) after a monitor sufficiently recognize the VCLK signal.

The S3C8618/C8615 enter DDC1 mode by executing the following steps:

1. A “1” is output at the SCL and SDA pins.
2. DDCCON.1 is set to “1” to activate DDC1 mode.

After these settings have been made, when a power-on reset occurs (and assuming that SCL remains High level), DDC1 mode is validated.

When DDC1 mode is validated, a continuous data stream is output in 128-byte EDID format from an external EEPROM in the display device. The transmission is synchronized with the VCLK frequency and is repeated continuously until SCL goes Low. If the monitor does the DDC1 data transmission, the host can terminate it by clearing DDCCON.0 to “0” at any time.

If, during the transmission, the host detects a High-to-Low transition at the SCL pin, it must switch immediately from DDC1 mode to DDC2 mode.

DDC2AB (ACCESS BUS) MODE

Overview

The higher-level DDC2AB protocol, which is based on the access bus protocol specification, is optional for monitors and host systems. Using this protocol, additional information about device features and extensions can be transmitted in bi-directional mode.

Access Bus (AB) Commands

There are two layers of access bus commands: 1) base level commands, and 2) application-specific commands.

Base level commands manage common AB features such as address assignment, capability strings, and power management. Application-specific commands are unique to a particular device.

The base level protocol is additionally subdivided into interface and application layers. The interface layer implements the AB commands required to configure the device. The application layer transfers specific device information to the host.

Monitor Response to Base AB Commands

Most monitors must support both AB and DDC1-type communications. Because of this requirement, additional information about how monitors respond to certain base AB commands is provided below.

During power-up (when the monitor detects host VCC), the monitor sends an attention message to the host to check whether it is AB-compliant. If the check fails, the monitor defaults to DDC1 mode and emulates a slave-only IIC-bus memory device at address A0H/A1H.

Interpretation of Reset Requests

AB-compliant host systems are required to reset all device addresses as part of the configuration process. Different monitors may, however, interpret a reset request differently.

AB-compliant monitors decode a host's reset signal as address 6EH and store this information. When the monitor has detected and validated the AB reset message, it becomes a fully compliant access bus device. It then remains in AB mode as long as the power supply to the access bus is maintained.

If the power level to the access bus drops, the monitor enters its default power-up mode, DDC1. To switch the monitor back to AB mode, the host must output a reset message (6EH).

Assignment of Access Bus Device Addresses

During the transfer of AB device identification data, the host detects each device on the access bus and assigns it a unique address. When the host has assigned an address to a display device, it issues an Assign Address command to replace the device’s default AB address. The monitor continues to use the host-assigned address until it receives another Assign Address command.

As mentioned above, if the access bus loses power, the monitor reverts to DDC1 mode. When power is restored, the monitor’s AB address is reset to its default value and sends the host an attention message. If the host acknowledges the message, the monitor remains in AB mode. If not, it reverts again to DDC1 mode (address A0H/A1H).

☞ PROGRAMMING TIP — Programming the IIC-Bus Interface

This example shows one way to program the IIC-bus module to sample specifications. The program parameters are as follows:

— Main crystal oscillation frequency is 8 MHz

— IIC-bus clock frequency (SCL) is 100 kHz

— The following registers are assigned for the program values:

Register 70H	Tx_DATA	;	Transmitted data
Register 71H	Rx_Data	;	Received data
Register 72H	SubAddr	;	Sub-address
Register 73	eepromf_ref	;	Register for external EEPROM
0	restartf		
1	eepdataRxf		
2	eepdataTxf		
3	eepreadRxf		
4	eepreadf		
5	eepsubAddrf		

Additional address information:

— Slave address: #0A0H (external EEPROM-type)

— Sub-address: #10H (data address pointer)

— IIC-bus data: #20H

☞ PROGRAMMING TIP — Programming the IIC-Bus Interface (Continued)

```
        •
        •
        •
IICunitTest:
        SB1                                   ;   Select bank 1
        TM      eepromf_ref,#01<<eepdataRxf   ;   eepreadRxf - "1"?
        JR      NZ,DataCompare
ChkACKbit:
        TM      ICSR,#01<<bit0                ;   LRB (ACK bit ) = "1"?
        JR      Z,EEPsubAddr
        LD      eepromf_reg,#00H
        OR      eepromf_reg,#01<<restartf     ;   restartf ← "1"
        AND     ICSR,#11011111B               ;   Stop
        JP      IICrtn
EEPsubAddr:
        TM      eepromf_reg,#01<<eepsubAddrf  ;   IIC1 → EEPROM (write)
        JR      NZ,EEPsubAddr
        LD      IDSR,#SubADDR                 ;   Call EEPROM sub-address
        OR      eepromf_reg,#01<<eepsubAddrf  ;   eepsubAddrf ← "1"
        JP      IICrtn
EEPdataTx:
        TM      eeprom_reg,#01<<eepreadf
        JR      NZ,EEPdataRx
        TM      eepromf_reg,#01<<eepdataTxf
        JR      NZ,Tx1Byte
        LD      IDSR,TxDATA                   ;   Byte Tx (IIC1 → EEPROM)
        OR      eepromf_reg,#01<<eepdataTxf
```

☞ PROGRAMMING TIP — Programming the IIC-Bus Interface (Continued)

```
            JP      IICrtn
Tx1Byte:
            AND     IDSR,#11011111B                     ;  Stop
            AND     eepromf_reg,#0FFH-(01<<eepdataTxf)
            OR      eepromf_reg,#01<<restartf           ;  restartf ← "1"
            OR      eepromf_reg,#01<<eepreadf           ;  eepreadf ← "1"
            AND     eepromf_reg,#0FFH-(01<<eepsubAddrf
            JP      IICrtn
EEPreadRx:
            TM      eepromf_reg,#01<<eepreadRxf;        ;  eepreadRxf = "1"?
            JR      NZ,EEPdataRx
            LD      IDSR,#0A9H                          ;  Read mode
            LD      ICSR,#9FH                           ;  IIC1 = Master Rx
            OR      eeprom_reg,#01<<eepreadRxf          ;  eepreadRxf ← "1"
            OR      ICSR,#00100000B                     ;  Repeat start
            JP      IICrtn
EEPdataRx:
            TM      eepromf_reg,#01<<eepdataRxf         ;  eepdataRxf = "1"?
            JR      NZ,DataCompare
            OR      eepromf_reg,#01<<eepdataRxf         ;  eepdataRxf ← "1"
            AND     ICCR,#01111111B                     ;  ACK = High
            JP      IICrtn
```

(Continued on next page)

☞ PROGRAMMING TIP — Programming the IIC-Bus Interface (Continued)

```
DataCompare:
            LD      RxDATA,IDSR
            CP      TxDATA,RxDATA               ;   TxDATA = RxDATA?
            JP      NE,IICrtn
EEPdataDisp:
            INC     SubADDR                     ;   00H–FFH
            DEC     TxDATA                      ;   FFH–00H
            LD      eepromf_reg,#00H            ;   eepromf_reg clear
            OR      eepromf_reg,#01<<restartf   ;   restartf ← "1"
            AND     ICSR,#11011111B             ;   Stop
            JP      IICrtn
WaitIIC:
            LD      R4,#0FFH
IICwait0:
            LD      R5,#0FH
IICwait1:
            NOP
            NOP
            DJNZ    R5,IICwait1
            DJNZ    R4,IICwait0
IICrtn:
            RET
            •
            •
            •
```

16 ELECTRICAL DATA

OVERVIEW

In this section, S3C8618/C8615 electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

— Absolute maximum ratings
— D.C. electrical characteristics
— I/O capacitance
— A.C. electrical characteristics
— Oscillation characteristics
— Oscillation stabilization time
— Schmitt trigger characteristics

Table 16-1. Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 7.0	V
Input voltage	V_{I1}	Type C (n-channel, open-drain)	– 0.3 to + 10	V
	V_{I2}	All port pins except V_{I1}	– 0.3 to V_{DD} + 0.3	
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	– 10	mA
		All I/O pins active	– 60	
Output current Low	I_{OL1}	One I/O pin active	+ 30	mA
	I_{OL2}	Total pin current except port 3	+ 100	
	I_{OL3}	Sync-processor I/O pins and IIC-bus clock and data pins	+ 150	
Operating temperature	T_A	–	– 40 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	°C

Table 16-2. D.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.5 V to 5.5 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}	X_{IN}, X_{OUT}	V_{DD} – 0.5		V_{DD}	
	V_{IH3}	TTL input (HsyncI, VsyncI and CsyncI)	2.0		V_{DD}	
Input Low voltage	V_{IL1}	All input pins except V_{IL2} and V_{IL3}	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN}, X_{OUT}			0.4	
	V_{IL3}	TTL input (HsyncI, VsyncI and CsyncI)			0.8	
Output High voltage	V_{OH1}	V_{DD}= 4.5 V to 5.5 V I_{OH} = – 8 mA Port 1 only	V_{DD} – 1.0	–	–	V
	V_{OH2}	V_{DD} = 4.5 V to 5.5 V I_{OH} = – 2 mA Ports 0, 2, ClampO, H and VsyncO	V_{DD} – 1.0			
	V_{OH3}	V_{DD} = 4.5 V to 5.5 V I_{OH} = – 6 mA, Port 3	V_{DD} – 1.0			

Table 16-2. D.C. Electrical Characteristics (Continued)

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.5 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Output Low voltage	V_{OL1}	V_{DD} = 4.5 V to 5.5 V I_{OL} = 8 mA, port 1 only	–	–	0.4	V
	V_{OL2}	I_{OL} = 2 mA Port 0, 2, ClampO, HsyncO and VsyncO			0.4	
	V_{OL3}	I_{OL} = 6 mA Port 3, SCL and SDA			0.4	
Input High leakage current	I_{LIH1}	$V_{IN} = V_{DD}$ All input pins except X_{IN}, X_{OUT}	–	–	3	µA
	I_{LIH2}	$V_{IN} = V_{DD}$ X_{OUT} only	–	–	20	
	I_{LIH3}	$V_{IN} = V_{DD}$ X_{IN} only	2.5	6	20	
Input Low leakage current	I_{LIL1}	V_{IN} = 0 V All input pins except X_{IN}, X_{OUT} and RESET	–	–	– 3	µA
	I_{LIL2}	V_{IN} = 0 V; X_{OUT} only	–	–	– 20	
	I_{LIL3}	V_{IN} = 0 V; X_{IN} only	– 2.5	– 6	– 20	
Output High leakage current	I_{LOHL}	$V_{OUT} = V_{DD}$ All output pins except port 1	–	–	3	µA
Output Low leakage current	I_{LOL}	V_{OUT} = 0 V	–	–	– 3	µA
Pull-up resistor	R_{L1}	V_{IN} = 0 V; V_{DD} = 4.5 V to 5.5 V Port 3	20	47	80	kΩ
	R_{L2}	V_{IN} = 0 V; V_{DD} = 4.5 V to 5.5 V RESET only	150	280	480	
Supply current (note)	I_{DD1}	V_{DD} = 4.5 V to 5.5 V 12 MHz CPU clock	–	15	30	mA
	I_{DD2}	Idle mode; V_{DD} = 4.5 V to 5.5 V 12 MHz CPU clock		5	10	
	I_{DD3}	Stop mode; V_{DD} = 5.0 V		1	10	µA

NOTE: Supply current does not include drawn internal pull–up resistors and external loads of output.

Table 16-3. 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}	Stop mode	2	–	6	V
Data retention supply current	I_{DDDR}	Stop mode, V_{DDDR} = 2.0 V	–	–	5	μA

NOTES:
1. During the oscillator stabilization wait time (t_{WAIT}), all CPU operations must be stopped.
2. Supply current does not include drawn through internal pull–up resistors and external output current loads.

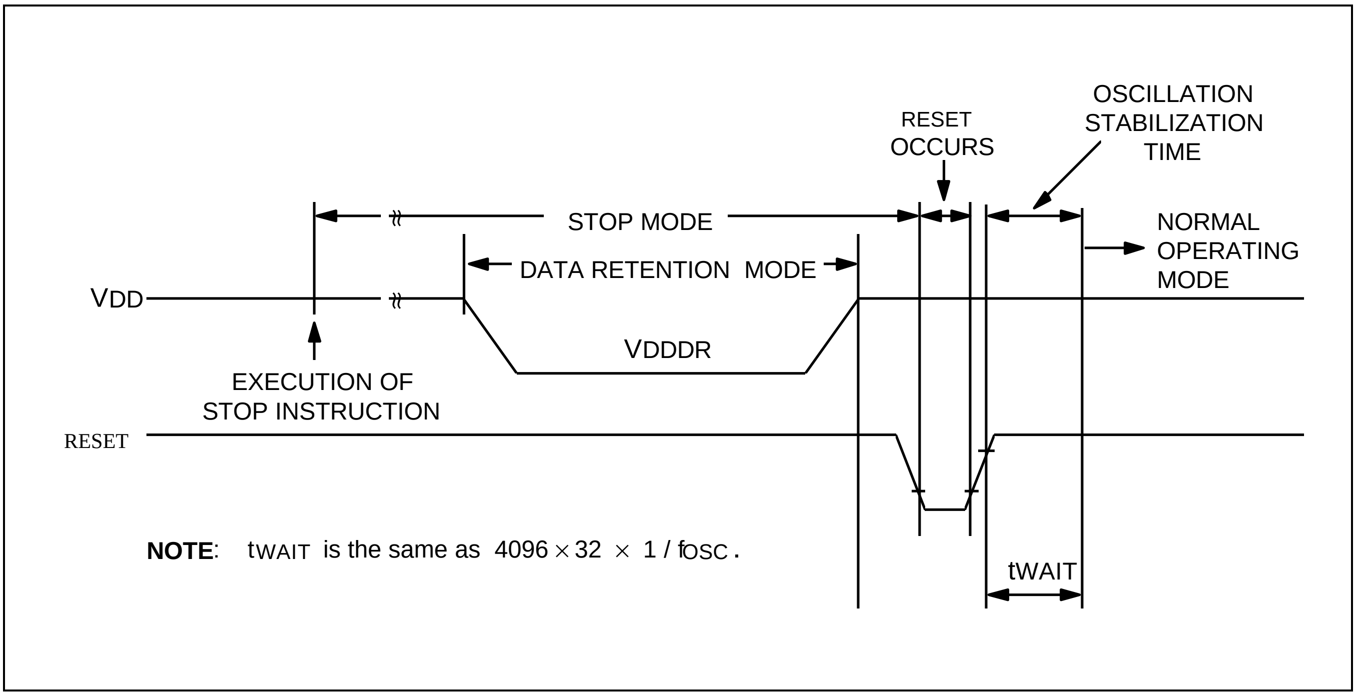


Figure 16-1. Stop Mode Release Timing When Initiated by a Reset

Table 16-4. Input/Output Capacitance

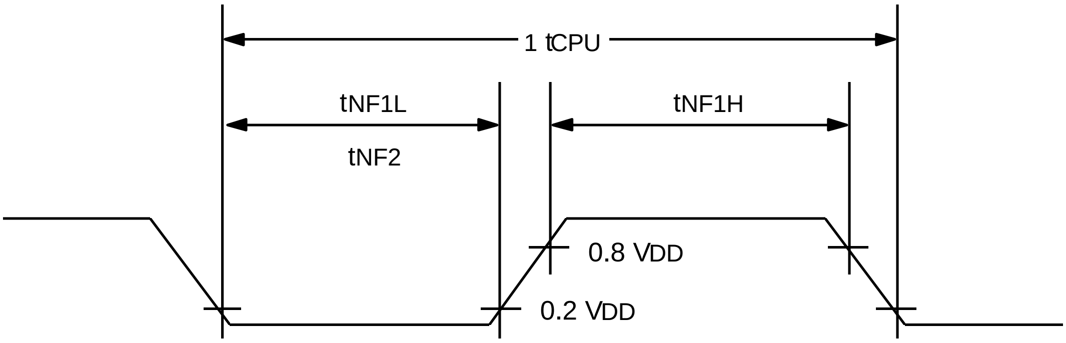
(T_A = – 40 °C to + 85 °C, V_{DD} = 0 V)

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 16-5. A.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.5 V to 5.5V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Noise Filter	t_{NF1H}, t_{NF1L}	P0.2–P0.0, T0CAP and T1CK (RC delay)	300	–	–	ns
	t_{NF2}	RESET only (RC delay)	800	–	–	

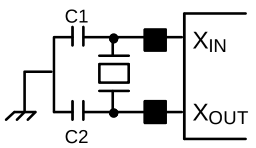
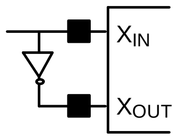


NOTE: The unit tCPU means one CPU clock period.

Figure 16-2. Input Timing Measurement Points for P0.0–P0.2, T0CAP and T1CK

Table 16-6. Oscillation Characteristics

(T_A = – 40 °C + 85 °C)

Oscillator	Clock Circuit	Conditions	Min	Typ	Max	Unit
Main crystal or ceramic		V_{DD} = 4.5 V to 5.5 V	6	–	12	MHz
External clock (main)		V_{DD} = 4.5 V to 5.5 V	6	–	12	MHz

NOTE: The maximum oscillator frequency is 12 MHz. If you use an oscillator frequency higher than 12 MHz, you cannot select a non-divided CPU clock using CLKCON settings. That is, you must select one of the divide-by values.

Table 16-7. Recommended Oscillator Constants

(T_A = – 40 °C + 85 °C, V_{DD} = 4.5 V to 5.5 V)

Manufacturer	Product Name	Load Cap (pF)		Oscillator Voltage Range (V)		Remarks
		C1	C2	MIN	MAX	
TDK	FCR8.0MC5 (note)	–	–	4.5	5.5	On-chip C Leaded Type
	FCR8.0M5	33	33	4.5	5.5	Leaded Type
	CCR8.0MC5 (note)	–	–	4.5	5.5	On-chip C SMD Type

NOTE: On-chip C: 30 pF ± 20 % built in.

Table 16-8. Oscillation Stabilization Time

(T_A = – 40 °C + 85 °C, V_{DD} = 4.5 V to 5.5 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	V_{DD} = 4.5 V to 5.5 V	–	–	20	ms
Ceramic	V_{DD} = 4.5 V to 5.5V	–	–	10	
External clock	X_{IN} input High and Low level width (t_{XH}, t_{XL})	25	–	500	ns

NOTE: Oscillation stabilization time is the time required for the CPU clock to return to its normal oscillation frequency after a power-on occurs, or when Stop mode is released.

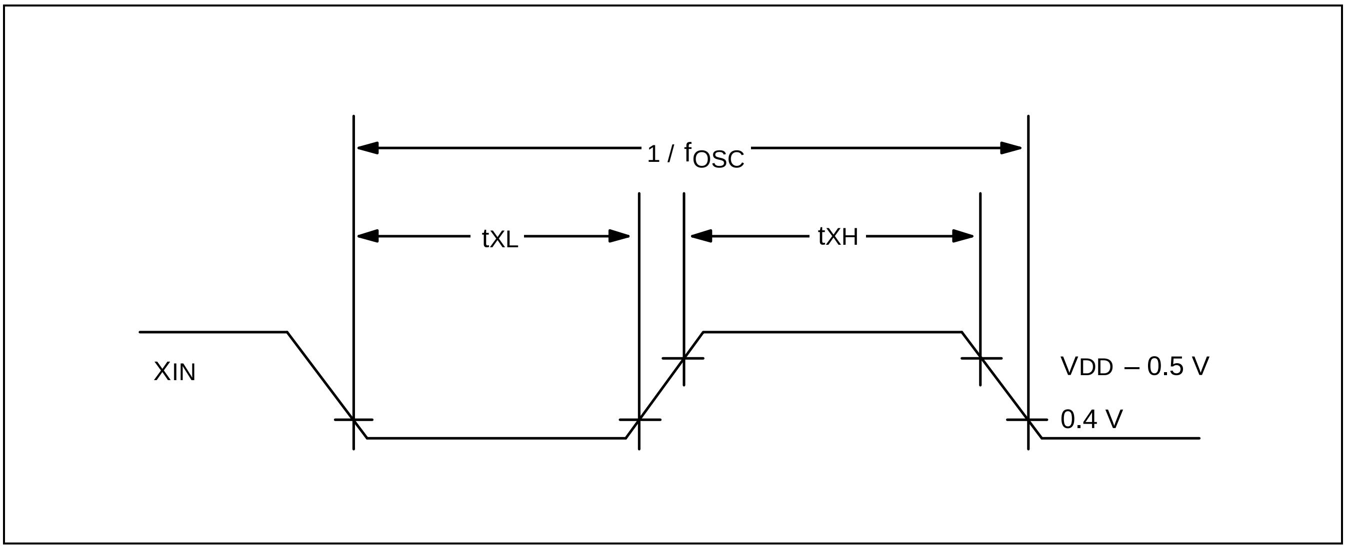


Figure 16-3. Clock Timing Measurement Points for X_{IN}

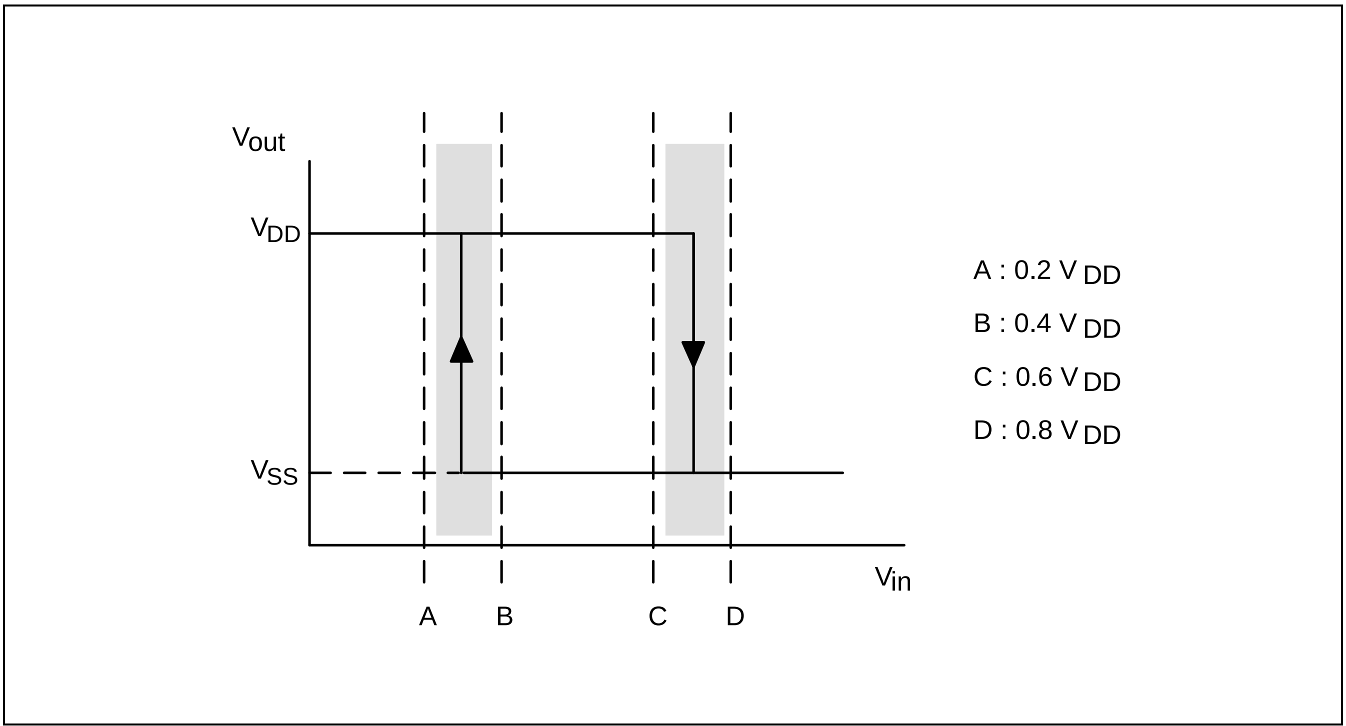
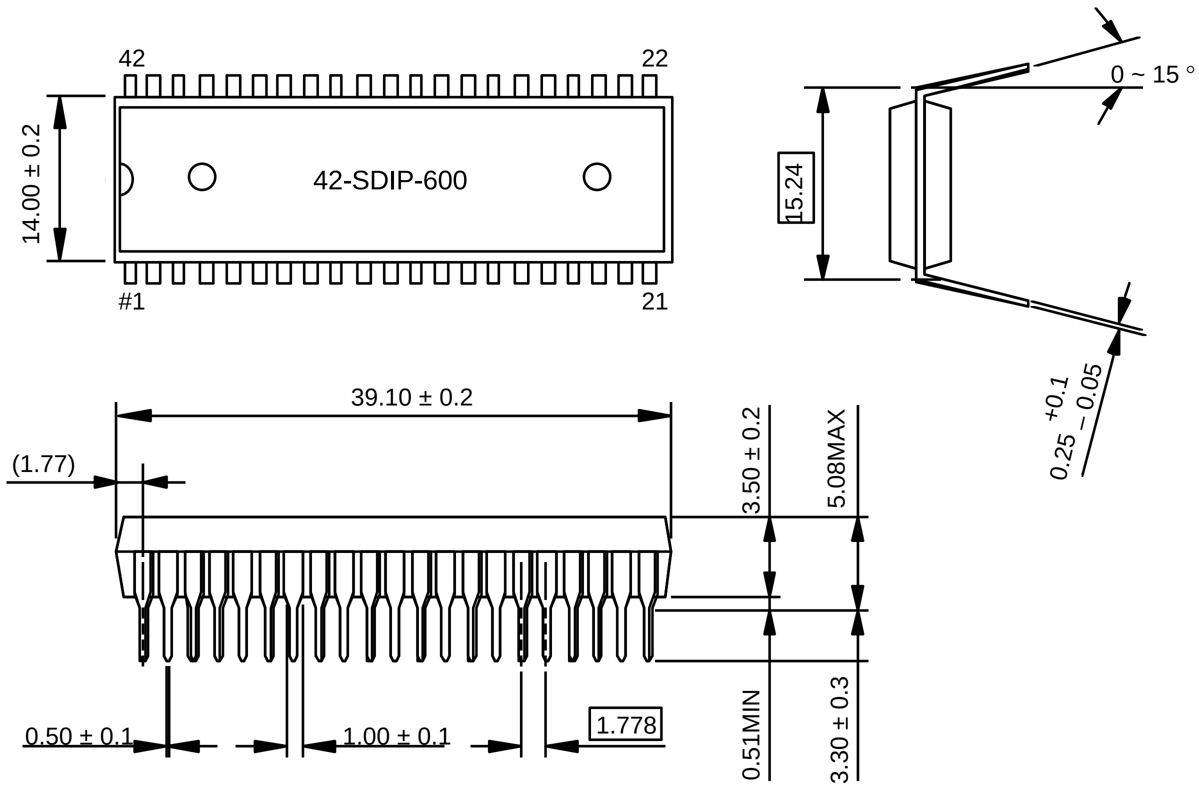


Figure 16-4. Schmitt Trigger Characteristics (Normal Port; except TTL Input)

17 MECHANICAL DATA

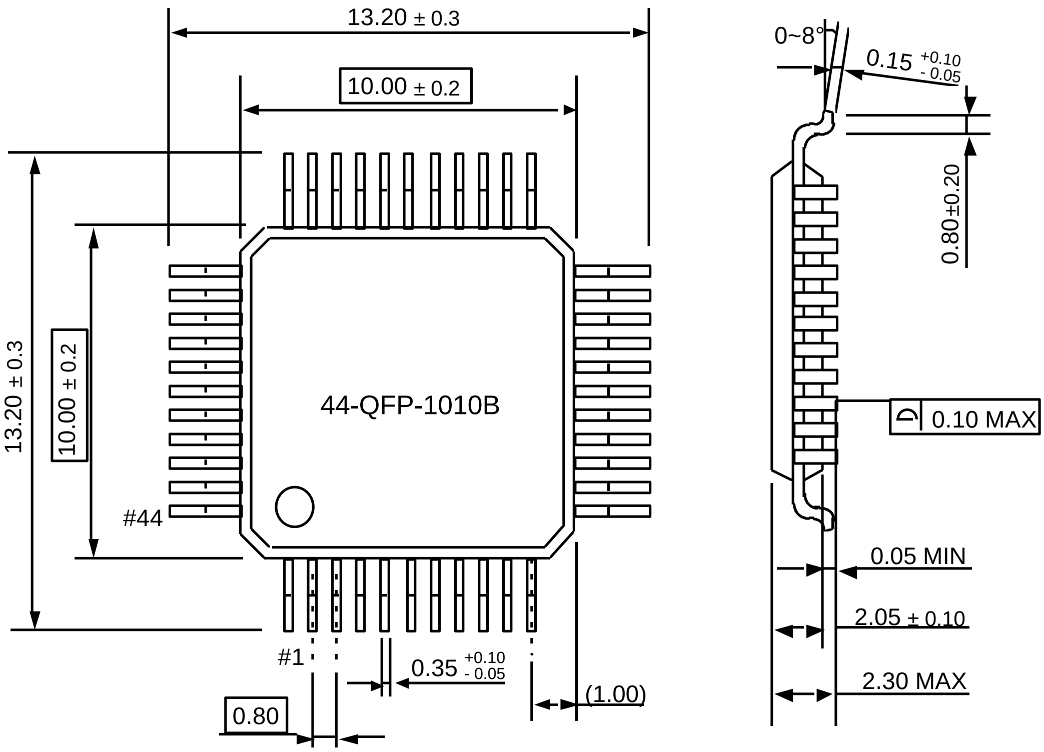
OVERVIEW

The S3C8615 microcontroller is available in a 42-pin SDIP package (Samsung part number 42-SDIP-600) and a 44-QFP package (Samsung part number 44-QFP-1010B).



NOTE: Dimensions are in millimeters.

Figure 17-1. 42-Pin SDIP Package Mechanical Data (42-SDIP-600)



NOTE: Dimensions are in millimeters.

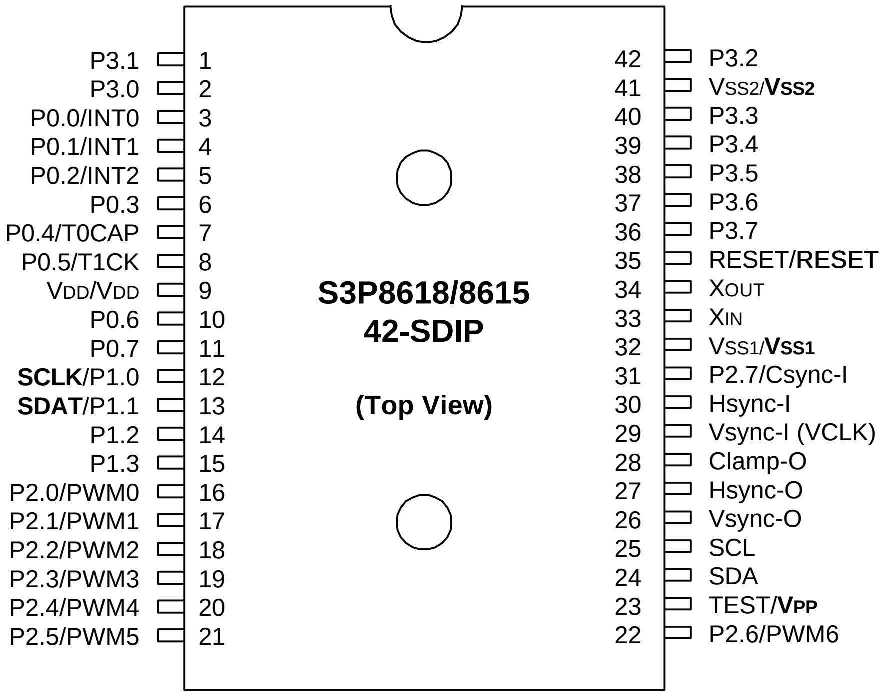
Figure 17-2. 44-Pin QFP Package Mechanical Data (44-QFP-1010B)

18 S3P8615 OTP

OVERVIEW

The S3P8615 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the S3P8615 microcontrollers. It has an on-chip EPROM instead of masked ROM. The EPROM is accessed by serial data format.

The S3P8615 is fully compatible with the S3C8618/C8615, both in function and in pin configuration. Because of its simple programming requirements, the S3P8615 is ideal for use as an evaluation chip for the S3C8618/C8615.



NOTE: The bolds indicate an OTP pin name.

Figure 18-1. S3P8615 Pin Assignments (42-SDIP Package)

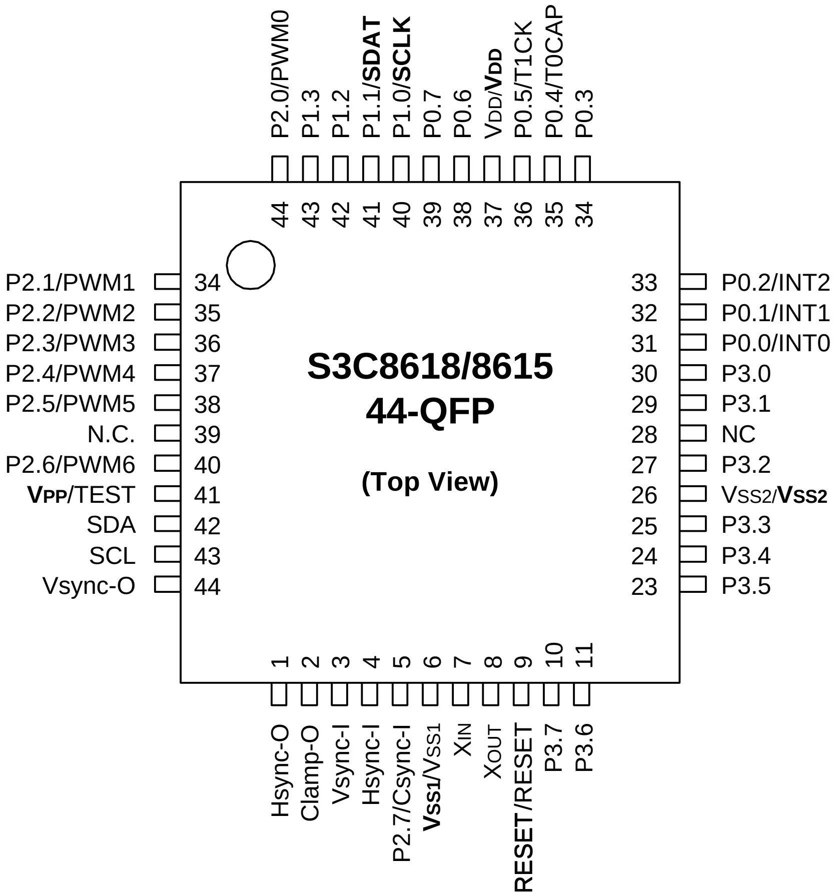


Figure 18-2. S3P8615 Pin Assignments (44-QFP Package)

Table 18-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip	During Programming			
Pin Name	Pin Name	Pin No.	I/O	Function
P1.1	SDAT	13 (*30)	I/O	Serial DATa Pin (Output when reading, Input when writing) Input & Push-pull Output Port can be assigned
P1.0	SCLK	12 (*29)	I	Serial CLocK Pin (Input Only Pin)
TEST	V_{PP} (TEST)	23 (*41)	I	EPROM Cell Writing Power Supply Pin (Indicates OTP Mode Entering) When writing 12.5 V is applied and when reading 5V is applied.(Option)
RESET	RESET	35 (*9)	I	Chip Initialization
$V_{DD}/V_{SS1}/V_{SS2}$	$V_{DD}/V_{SS}/V_{SS}$	9 / 32 / 41 (*26 / 6 / 15)	I	Logic Power Supply Pin. V_{DD} should be tied to 5 V during programming.

NOTE: * means the 44-QFP OTP pin number.

Table 18-2. Comparison of S3P8615 and S3C8618/C8615 Features

Characteristic	S3P8615	S3C8618/C8615
Program Memory	16 K byte EPROM	16 K byte mask ROM
Operating Voltage (V_{DD})	4.5 V to 5.5 V	4.5 V to 5.5V
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (TEST)=12.5V	
Pin Configuration	42-SDIP, 44-QFP	42-SDIP, 44-QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP} (TEST) pin of the S3P8615, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 16-3 below.

Table 18-3. Operating Mode Selection Criteria

V_{DD}	V_{PP} (TEST)	REG/ MEM	ADDRESS (A15–A0)	R/W	MODE
5 V	5 V	0	0000H	1	EPROM read
	12.5 V	0	0000H	0	EPROM program
	12.5 V	0	0000H	1	EPROM verify
	12.5 V	1	0E3FH	0	EPROM read protection

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

D.C. ELECTRICAL CHARACTERISTICS

Table 18-4. D.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.5 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High leakage current	I_{LIH1}	$V_{IN} = V_{DD}$; All input pins except X_{IN}, X_{OUT}	–	–	3	µA
	I_{LIH2}	$V_{IN} = V_{DD}$; X_{OUT} only			20	
	I_{LIH3}	$V_{IN} = V_{DD}$; X_{IN} only	2.5	6	20	
Input Low leakage current	I_{LIL1}	V_{IN} = 0 V; All input pins except X_{IN}, X_{OUT} and RESET	–	–	– 3	µA
	I_{LIL2}	V_{IN} = 0 V; X_{OUT} only	–	–	– 20	
	I_{LIL3}	V_{IN} = 0 V; X_{IN} only	– 2.5	– 6	– 20	
Output High leakage current	I_{LOH1}	$V_{OUT} = V_{DD}$	–	–	3	µA
Output Low leakage current	I_{LOL1}	V_{OUT} = 0 V	–	–	– 3	µA
Supply current	I_{DD1}	Normal operating mode; 12 MHz CPU clock	–	15	30	mA
	I_{DD2}	IDLE mode; 12 MHz CPU clock		5	10	
	I_{DD3}	Stop mode; V_{DD} = 5.0 V	–	1	10	µA
Data retention supply voltage	V_{DDDR}	Stop mode	2	–	6	V
Data retention supply voltage	I_{DDDR}	Stop mode; V_{DDDR} = 2V	–	–	5	µA

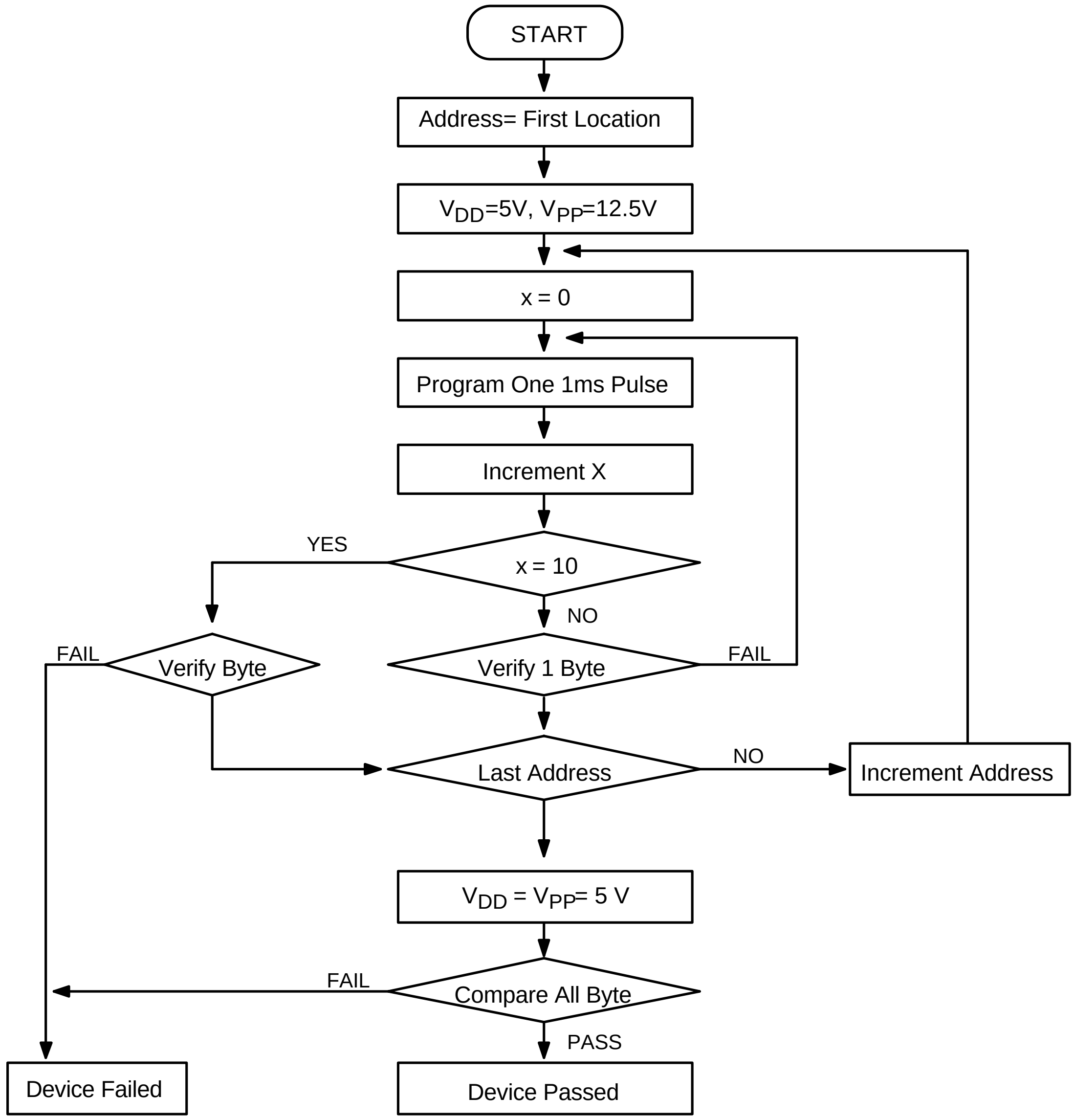


Figure 18-3. OTP Programming Algorithm

19 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, S3C6, 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 upto 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.

OTPs

One time programmable microcontroller (OTP) for the S3C8618/C8615 microcontroller and OTP programmer (Gang) are now available.

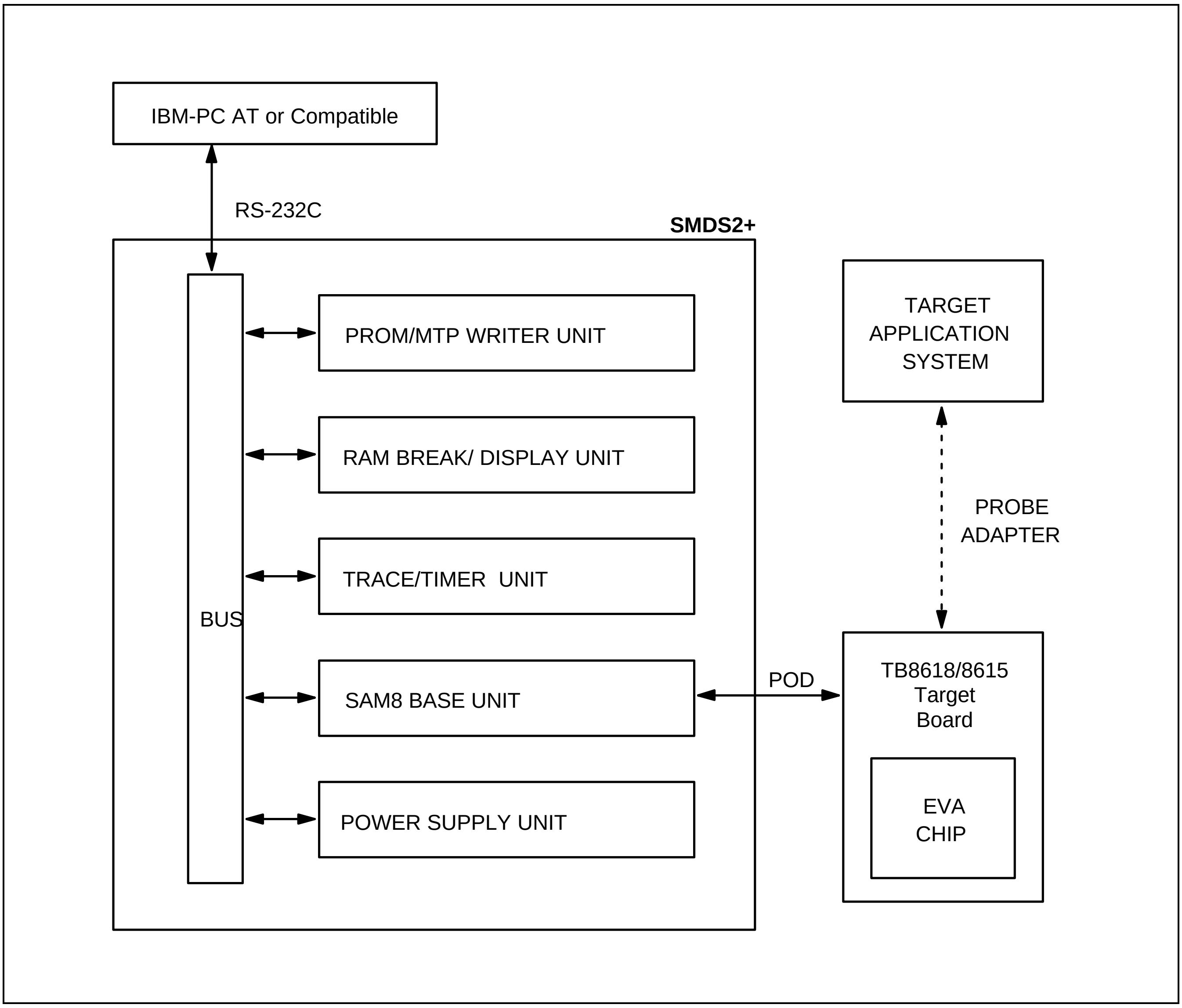


Figure 19-1. SMDS Product Configuration (SMDS2+)

TB8618/8615 TARGET BOARD

The TB8618/8615 target board is used for the S3C8618/C8615/P8615 microcontroller. It is supported by the SMDS2+ development system.

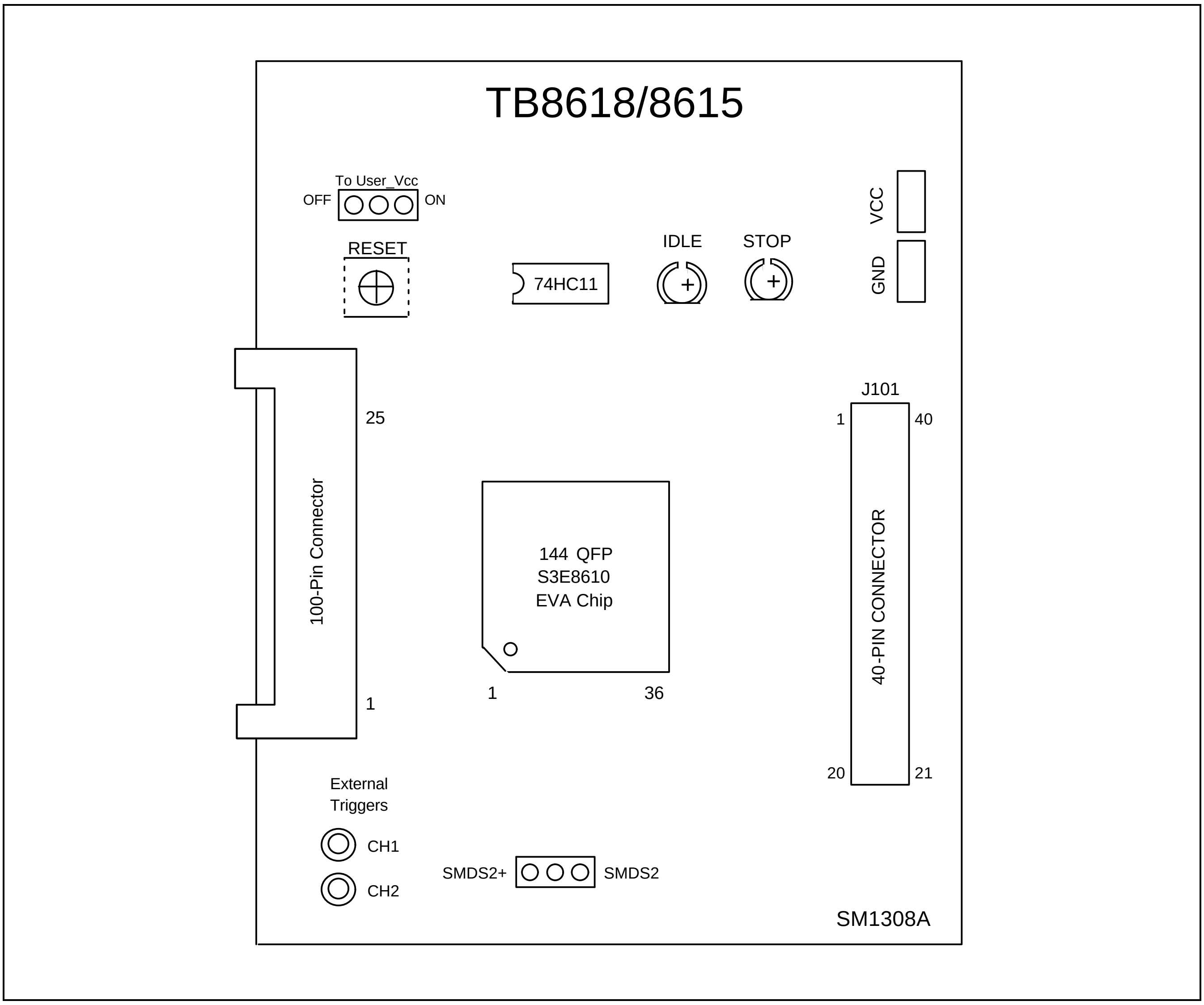
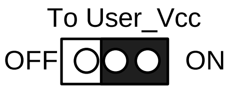
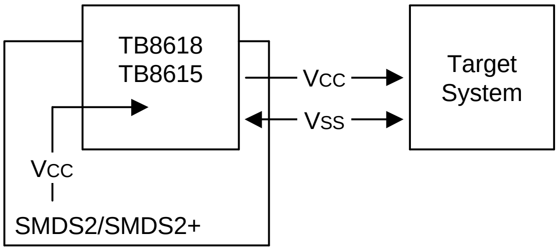
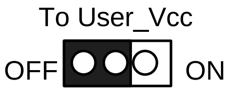
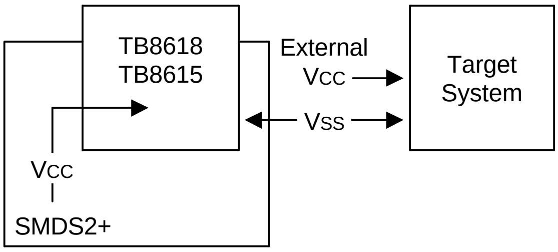


Figure 19-2. TB8618/8615 Target Board Configuration

Table 19-1. Power Selection Settings for TB886108A/TB886116A

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

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 19-2. The SMDS2+ Tool Selection Setting

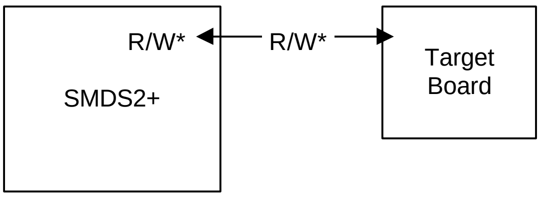
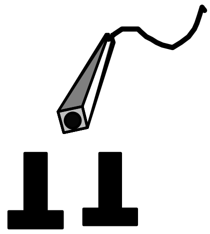
'SW1' Setting	Operating Mode
	

Table 19-3. Using Single Header Pins as the Input Path for External Trigger Sources

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

IDLE LED

This LED is ON when the evaluation chip (S3E8610) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (S3E8610) is in stop mode.

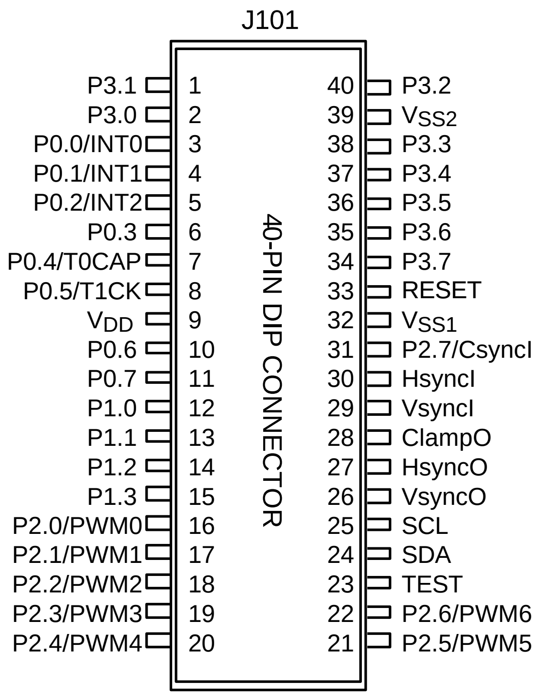


Figure 19-3. 40-Pin Connector for TB8618/8615

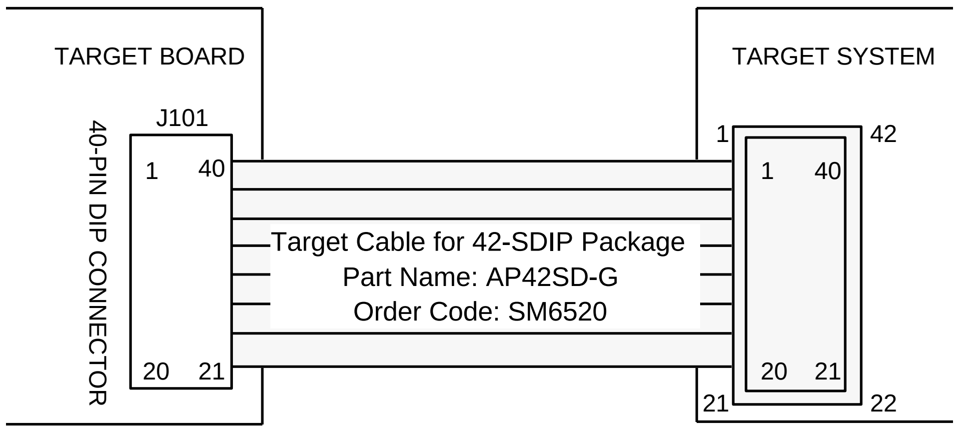


Figure 19-4. TB8618/8615 Adapter Cable for 42-SDIP Package (S3C8618/C8615/P8615)