1 PRODUCT OVERVIEW

The S3C7524/C7528/C7534/C7538 single-chip CMOS microcontroller has been designed for high-performance using SAM 47 (Samsung Arrangeable Microcontrollers). SAM 47, Samsung's newest 4-bit CPU core is notable for its low energy consumption and low operating voltage.

You can select from two ROM sizes: 4K or 8K bytes
Except for the difference in ROM size, the features and functions of the S3C7524 and the S3C7528, the S3C7534 and the S3C7538 are identical.

With it's DTMF generator, watchdog timer function, and versatile 8-bit timer/counters, theS3C7524/C7528 /C5304/C5308 offers an excellent design solution for a wide variety of telecommunication applications.

Up to 35 pins of the available 42-pin SDIP or 44-pin QFP package for the S3C7524/C7528, and up to 23 pins of the available 30-pin SDIP or 32-pin SOP package for the S3C7534/C7538 can be assign to I/O. Six vectored interrupts for S3C7524/C7528 and four vectored interrupts for S3C7534/C7538 provide fast response to internal and external events. In addition, the S3C7524/C7528/C7534/C7538 's advanced CMOS technology provides for low power consumption and a wide operating voltage range.

OTP

The S3C7524/C7528 microcontroller is also available in OTP (One Time Programmable) version, S3P7528. The S3C7534/C7538 microcontroller is also available in OTP (One Time Programmable) version, S3P7538. The S3P7528/P7538 microcontroller has an on-chip 8K-byte one-time-programable EPROM instead of masked ROM. The S3P7528 is comparable to S3C7524/C7528, both in function and in pin configuration. Also, the S3P7538 is comparable to the S3C7534/C7538, both in function and in pin configuration.

FEATURES SUMMARY

Memory

- 768 × 4-bit RAM
- 4,096 × 8-bit ROM (S3C7524/C7534)
  8,192 × 8-bit ROM (S3C7528/C7538)

35 I/O Pins

- Input only: 4 pins (S3C7524/C7528)
  1 pins (S3C7534/C7538)
- I/O: 23 pins (S3C7524/C7528)
  14 pins (S3C7534/C7538)
- N-channel open-drain I/O: 8 pins

Memory-Mapped I/O Structure

- Data memory bank 15

DTMF Generator

- 16 dual-tone frequencies for tone dialing

8-Bit Basic Timer

- Programmable interval timer
- Watchdog timer

Two 8-Bit Timer/Counters

- Programmable 8-bit timer
- External event counter function
- Arbitrary clock frequency output

Watch Timer

- Real-time and interval time measurement
- Four frequency outputs to the BUZ pin

Bit Sequential Carrier

- Supports 8-bit serial data transfer in arbitrary format

Interrupts

- 3 external interrupt vectors (S3C7524/C7528)
  1 external interrupt vectors (S3C7534/C7538)
- 3 internal interrupt vectors
- 2 quasi-interrupts

Power-Down Modes

- Idle: Only CPU clock stops
- Stop: System clock stops

Oscillation Sources

- Crystal, or ceramic for main system clock
- Main system clock frequency: 0.4–6.0 MHz (typical)
- CPU clock divider circuit (by 4, 8, or 64)

Instruction Execution Times

- 0.95, 1.91, and 15.3 µs at 4.19 MHz
- 1.12, 2.23, 17.88 µs at 3.58 MHz
- 0.67, 1.33, 10.7 µs at 6.0 MHz

Operating Temperature

- – 40 °C to 85 °C

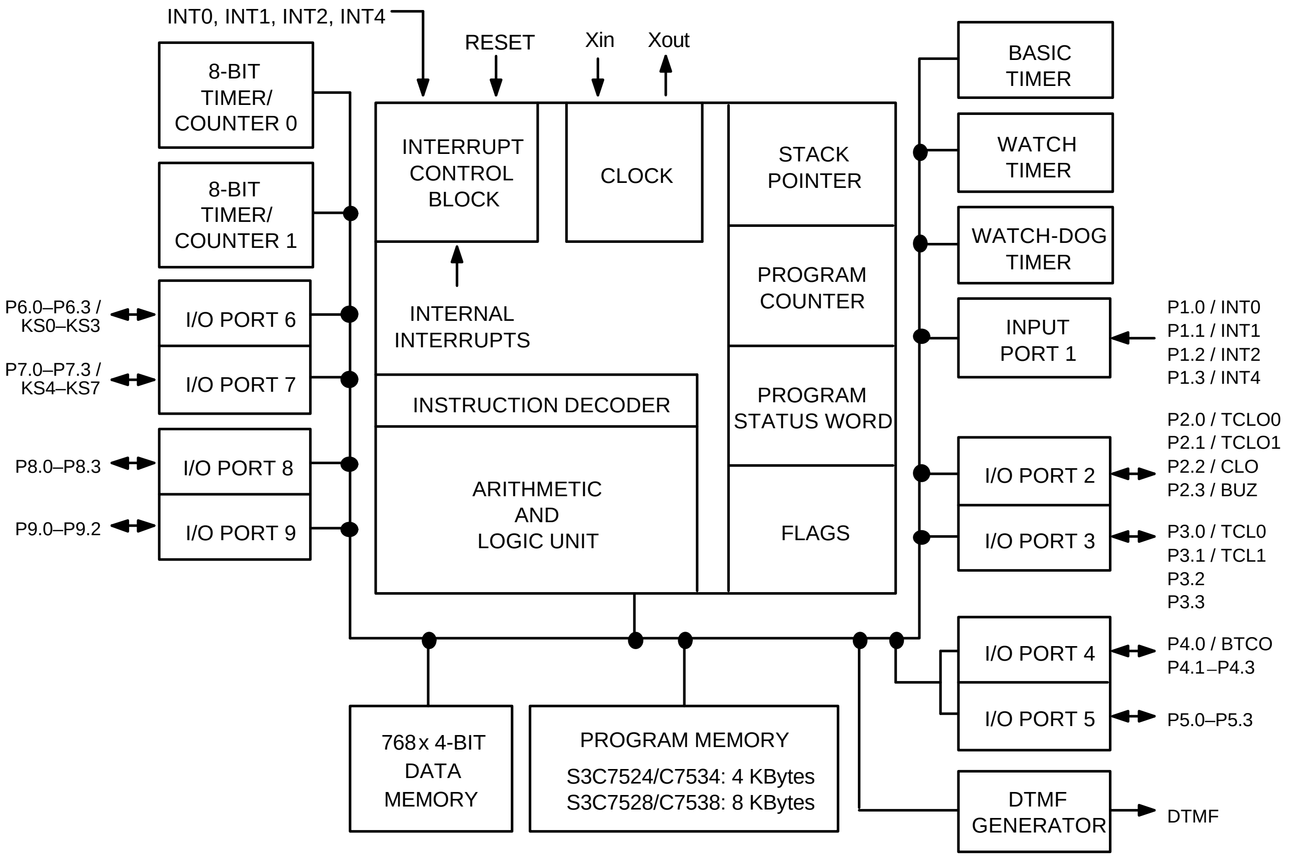
Operating Voltage Range

- 2.0 V to 5.5 V

Package Types

- 42 SDIP, 44 QFP (S3C7524/C7528)
- 30 SDIP, 32 SOP (S3C7534/C7538)

BLOCK DIAGRAM



NOTE: S3C7534/C7538 does not use P1.1/INT1, P1.2/INT2, P1.3/INT3, P3.2, P3.3, INT1, INT2, INT4, P8.0-P8.3, and P9.0-P9.2.

Figure 1–1. S3C7524/C7528 Simplified Block Diagram

PIN ASSIGNMENTS

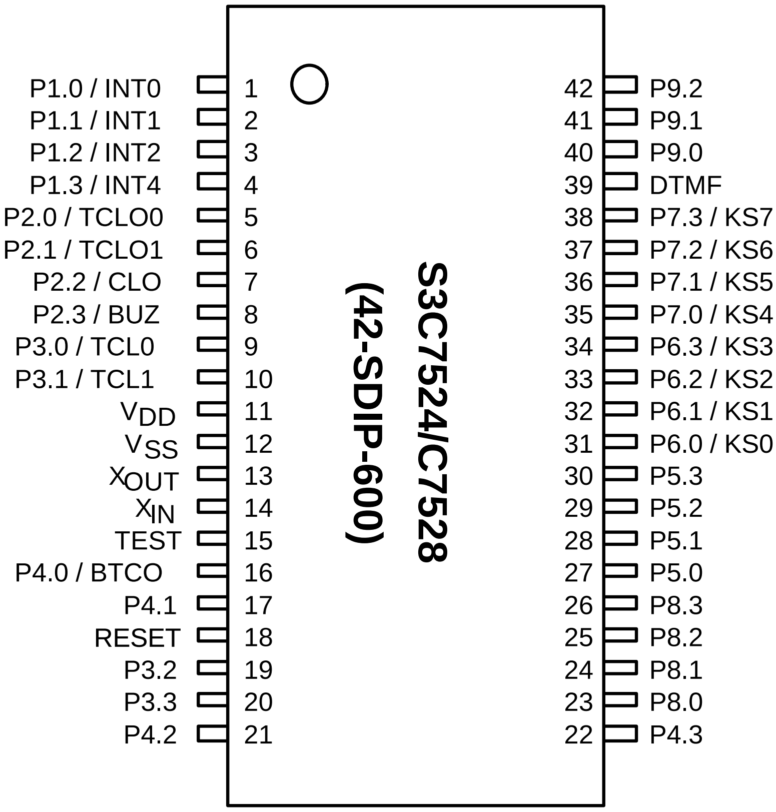


Figure 1–2. S3C7524/C7528 Pin Assignment Diagrams (42–SDIP)

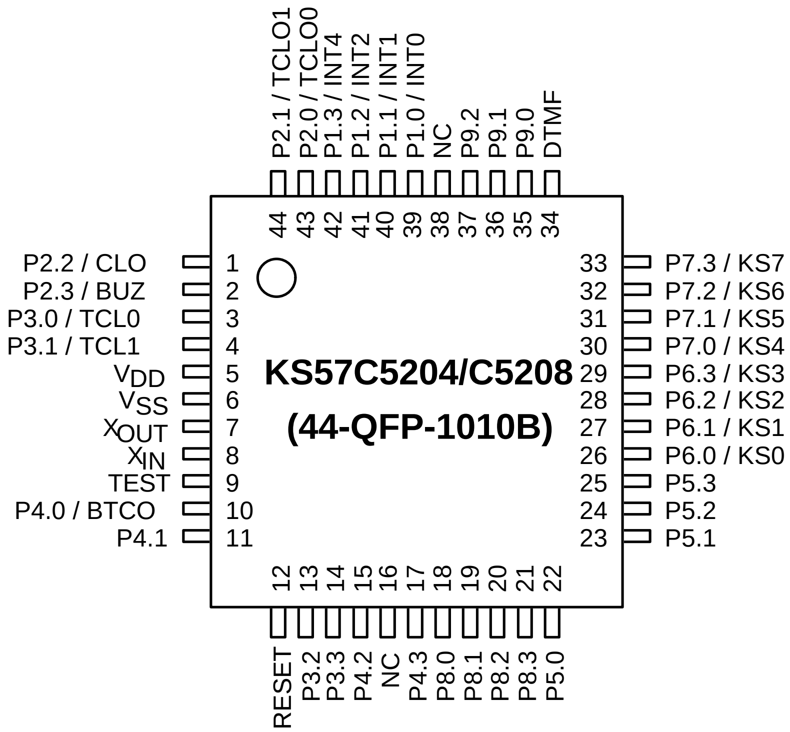


Figure 1–3. S3C7524/C7528 Pin Assignment Diagrams (44–QFP)

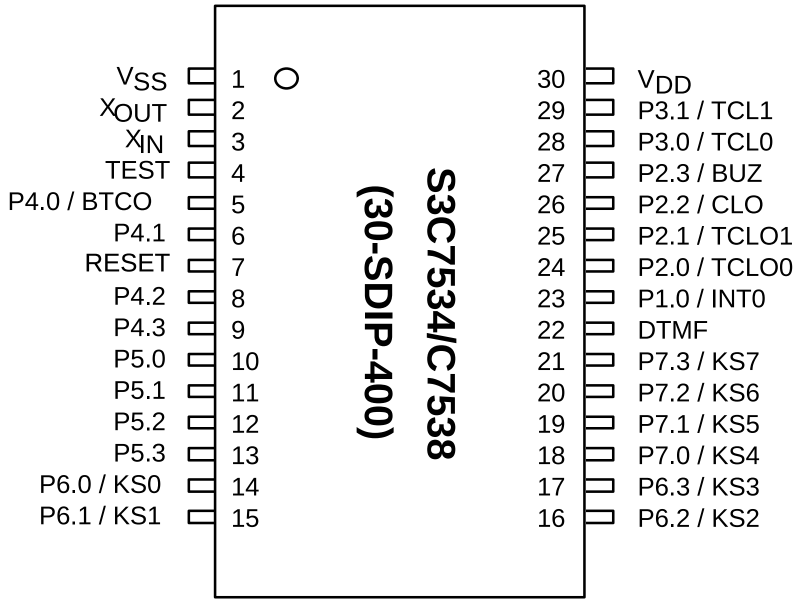


Figure 1–4. S3C7534/C7538 Pin Assignment Diagrams (30–SDIP)

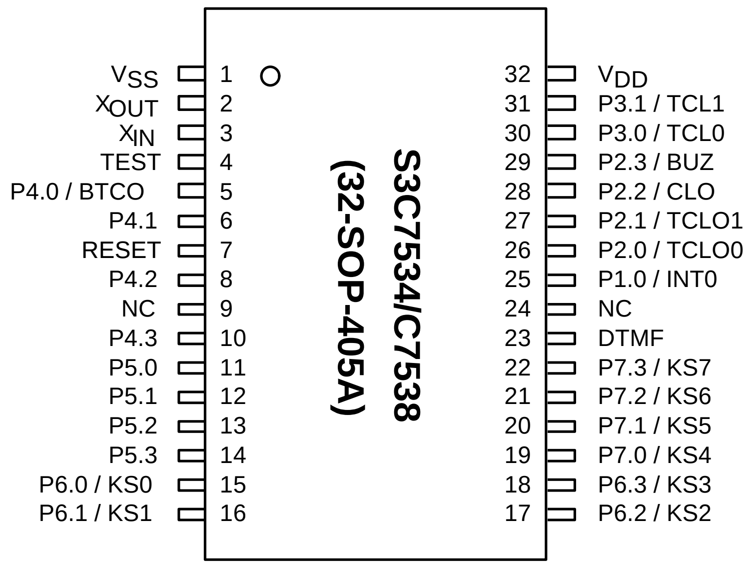


Figure 1–5. S3C7534/C7538 Pin Assignment Diagrams (32–SOP)

PIN DESCRIPTIONS

Table 1-1. S3C7524/C7528 Pin Descriptions

Pin Name	Pin Type	Reset Value	Description	Pin Number	Share Pin	Circuit Type
P1.0 P1.1 P1.2 P1.3	I	I	4-bit input port. 1-bit and 4-bit read and test is possible. Each pull-up resistors are assignable by software.	1 (39) 2 (40) 3 (41) 4 (42)	INT0 INT1 INT2 INT4	A-4
P2.0 P2.1 P2.2 P2.3	I/O	I	4-bit I/O port. 1-bit and 4-bit read/write and test is possible. Individual pins are software configurable as input or output.	5 (43) 6 (44) 7 (1) 8 (2)	TCLO0 TCLO1 CLO BUZ	D-2
P3.0 P3.1 P3.2 P3.3			4-bit pull-up resistors are software assignable to input pins and are automatically disabled for output pins. Ports 2 and 3 can be paired to enable 8-bit data transfer.	9 (3) 10 (4) 19 (13) 20 (14)	TCL0 TCL1	D-4
P4.0 P4.1 P4.2 P4.3 P5.0–P5.3	I/O	I	4-bit I/O ports. 1-bit and 4-bit read/write and test is possible. Individual pins are software configurable as input or output. 4-bit pull-up resistors are software assignable to input pins and are automatically disabled for output pins. N-channel open-drain or push-pull output can be selected by software (1-bit unit) Ports 4 and 5 can be paired to support 8-bit data transfer.	16 (10) 17 (11) 21 (15) 22 (17) 27–30 (22–25)	BTCO	E-2
P6.0–P6.3 P7.0–P7.3	I/O	I	4-bit I/O ports. 1-bit or 4-bit read/write and test is possible. Individual pins are software configurable as input or output. 4-bit pull-up resistors are software assignable to input pins and are automatically disabled for output pins. Ports 6 and 7 can be paired to enable 8-bit data transfer.	31–34 (26–29) 35–38 (30–33)	KS0–KS3 KS4–KS7	D-4
P8.0–P8.3 P9.0–P9.2	I/O	I	4-bit I/O port. 1-bit or 4-bit read/write and test is possible. Individual pins are software configurable as input or output. 4-bit pull-up resistors are software assignable to input pins and are automatically disabled for output pins. Ports 8 and 9 can be paired to enable 8-bit data transfer.	23–26 (18–21) 40–42 (35–37)	–	D-2

Table 1-1. S3C7524/C7528 Pin Descriptions (Continued)

Pin Name	Pin Type	Reset Value	Description	Pin Number	Share Pin	Circuit Type
DTMF	O	–	DTMF output.	39 (34)	–	G-6
BTCO	I/O	I	Basic timer clock output	16 (10)	P4.0	E-2
INT0 INT1	I	I	External interrupts. The triggering edge for INT0 and INT1 is selectable.	1 (39) 2 (40)	P1.0 P1.1	A-3
INT2	I	I	Quasi-interrupt with detection of rising edges	3 (41)	P1.2	A-3
INT4	I	I	External interrupt with detection of rising and falling edges.	4 (42)	P1.3	A-3
TCLO0	I/O	I	Timer/counter 0 clock output	5 (43)	P2.0	D-2
TCLO1	I/O	I	Timer/counter 1 clock output	6 (44)	P2.1	D-2
CLO	I/O	I	Clock output	7 (1)	P2.2	D-2
BUZ	I/O	I	2 kHz, 4 kHz, 8 kHz, or 16 kHz frequency output at the watch timer clock frequency of 4.19 MHz for buzzer sound	8 (2)	P2.3	D-2
TCL0	I/O	I	External clock input for timer/counter 0	9 (3)	P3.0	D-4
TCL1	I/O	I	External clock input for timer/counter 1	10 (4)	P3.1	D-4
KS0–KS3 KS4–KS7	I/O	I	Quasi-interrupt inputs with falling edge detection	31–34 (26–29) 35–38 (30–33)	P6.0– P6.3 P7.0– P7.3	D-4
V_{DD}	–	–	Power supply	11 (5)	–	–
V_{SS}	–	–	Ground	12 (6)	–	–
RESET	–	–	RESET signal	18 (12)	–	B
X_{in} X_{out}	–	–	Crystal, or ceramic oscillator signal for main system clock. (For external clock input, use X_{in} and input X_{in}'s reverse phase to X_{out})	14 (8) 13 (7)	–	–
TEST	–	–	Test signal input	15 (9)	–	–
NC	–	–	No connection	(16, 38)	–	–

NOTE: Parentheses indicate pin number for 44 QFP package.

Table 1-2. S3C7534/C7538 Pin Descriptions

Pin Name	Pin Type	Description	Pin Number	Share Pin	Circuit Type
P1.0	I	1-bit input port. 1-bit and 4-bit read and test is possible. Each bit pull-up resistors are assignable.	23 (25)	INT0	A-4
P2.0 P2.1 P2.2 P2.3	I/O	4-bit I/O port. 1-bit and 4-bit read/write and test is possible. Each individual pin can be assignable as input or output. 4-bit pull-up resisters are software assignable to input pins and are automatically disabled for output pins. Ports 2 and 3 can be paired to enable 8-bit data transfer.	24 (26) 25 (27) 26 (28) 27 (29)	TCLO0 TCLO1 CLO BUZ	D-2
P3.0 P3.1			28 (30) 29 (31)	TCL0 TCL1	D-4
P4.0 P4.1 P4.2 P4.3 P5.0–P5.3	I/O	4-bit I/O ports. 1-bit and 4-bit read/write and test is possible. Each individual pin can be assignable as input or output. 4-bit pull-up resisters are software assignable to input pins and are automatically disabled for output pins. The N-channel open-drain or push-pull output can be selected by software (1-bit unit). Ports 4 and 5 can be paired to enable 8-bit data transfer.	5 (5) 6 (6) 8 (8) 9 (10) 10–13 (11–14)	BTCO	E-2
P6.0–P6.3 P7.0–P7.3	I/O	4-bit I/O ports. 1-bit and 4-bit read/write and test is possible. Each individual pin can be assignable as input or output. 4-bit pull-up resisters are software assignable to input pins and are automatically disabled for output pins. Ports 6 and 7 can be paired to enable 8-bit data transfer.	14–17 (15–18) 18–21 (19–22)	KS0–KS3 KS4–KS7	D-4

Table 1-1. S3C7534/C7538 Pin Descriptions (Continued)

Pin Name	I/O Type	Description	Pin Number	Share Pin	Circuit Type
DTMF	O	DTMF output.	22 (23)	–	G-6
INT0	I	External interrupt input. The triggering edge for INT0 is selectable.	23 (25)	P1.0	A-3
TCLO0	I/O	Timer/counter 0 clock output	24 (26)	P2.0	D-2
TCLO1	I/O	Timer/counter 1 clock output	25 (27)	P2.1	D-2
CLO	I/O	Clock output	26 (28)	P2.2	D-2
BUZ	I/O	2 kHz, 4 kHz, 8 kHz, or 16 kHz frequency output at the watch timer clock frequency of 4.19 MHz for buzzer sound	27 (29)	P2.3	D-2
TCL0	I/O	External clock input for timer/counter 0	28 (30)	P3.0	D-4
TCL1	I/O	External clock input for timer/counter 1	29 (31)	P3.1	D-4
BTCO	I/O	Basic timer clock output	5 (5)	P4.0	E-2
V_{DD}	–	Power supply	30 (32)	–	–
V_{SS}	–	Ground	1 (1)	–	–
X_{in} X_{out}	–	Crystal, or ceramic oscillator signal for main system clock. (For external clock input, use X_{in} and input X_{in}'s reverse phase to X_{out})	3 (3) 2 (2)	–	–
NC	–	No connection	(9, 24)	–	–
TEST	–	Test signal input	4 (4)	–	–
RESET	–	RESET signal	7 (7)	–	B
KS0–KS3 KS4–KS7	I/O	Quasi-interrupt inputs with falling edge detection	14–17 (15–18) 18–21 (19–22)	P6.0– P6.3 P7.0– P7.3	D-4

NOTE: Parentheses indicate the pin number for 32-SOP package.

PIN CIRCUIT DIAGRAMS

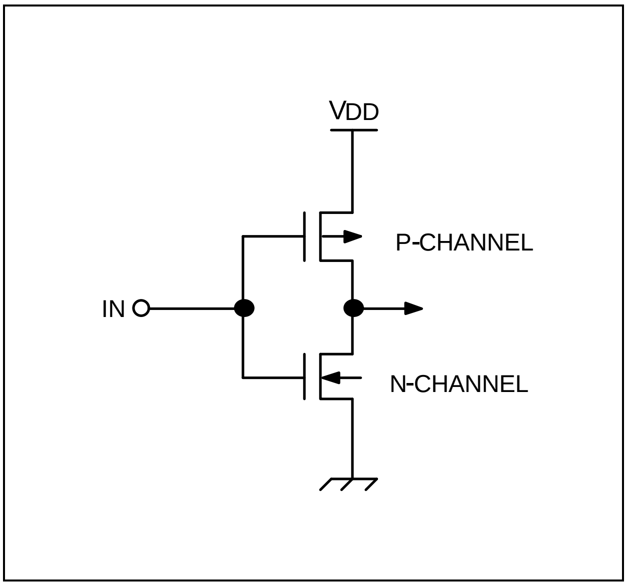


Figure 1–6. Pin Circuit Type A

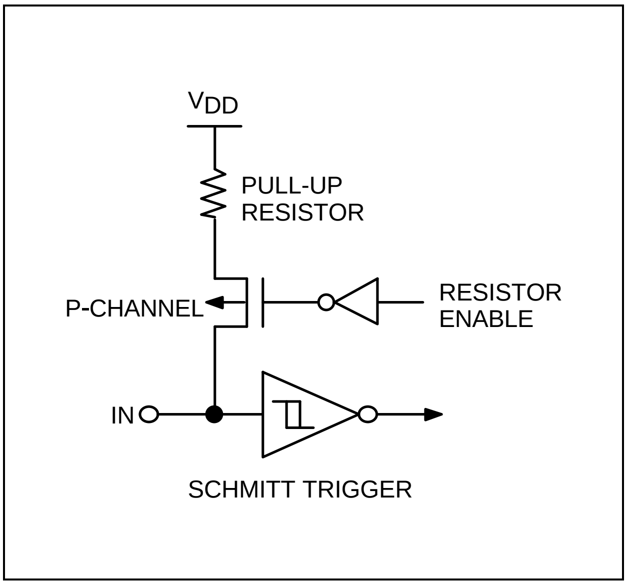


Figure 1–8. Pin Circuit Type A-4

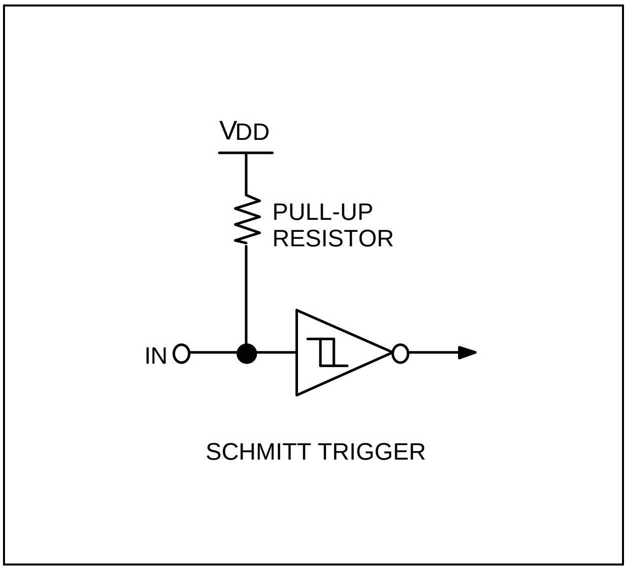


Figure 1–7. Pin Circuit Type B

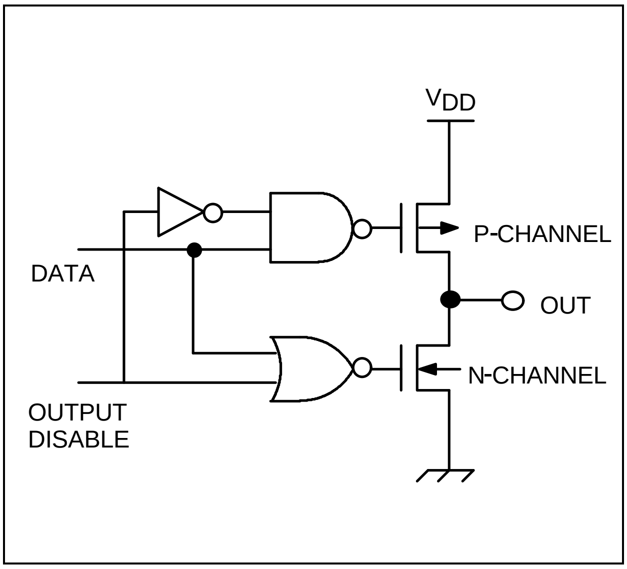


Figure 1–9. Pin Circuit Type C

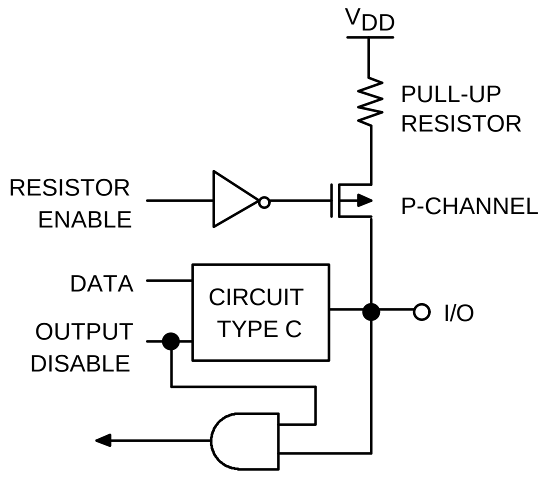


Figure 1–10. Pin Circuit Type D-2

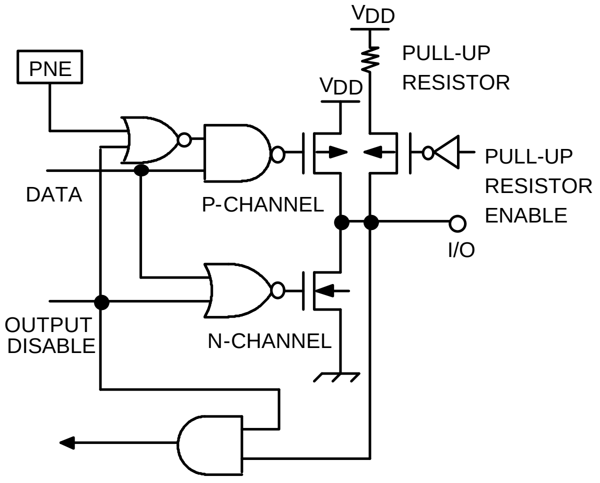


Figure 1–12. Pin Circuit Type E-2

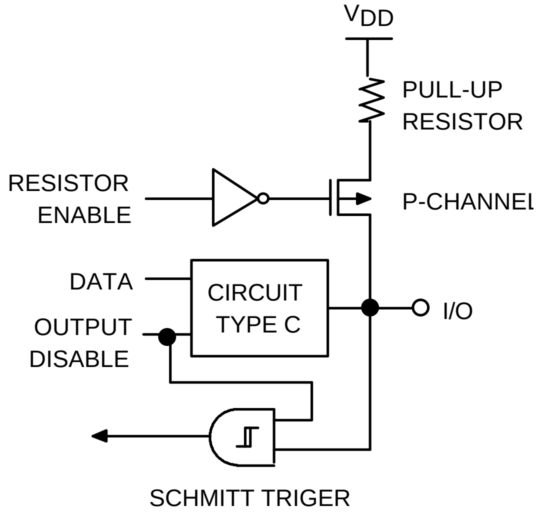


Figure 1–11. Pin Circuit Type D-4

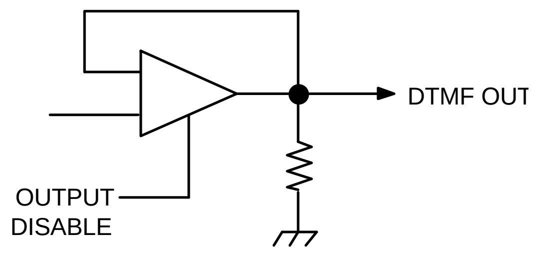


Figure 1–13. Pin Circuit Type G-6

2 ADDRESS SPACES

PROGRAM MEMORY (ROM)

OVERVIEW

ROM maps for the S3C7524/C7528/C7534/C7538 are mask programmable at the factory. In its standard configuration, the device's 4,096 x 8-bit (S3C7524/C7534) or 8,192 × 8-bit (S3C7528/C7538) program memory have three areas that are directly addressable by the program counter (PC):

— 16-byte area for vector addresses

— 16-byte general-purpose area

— 96-byte instruction reference area

— 3,968-byte general-purpose area (S3C7524/C7534)
8,064-byte general-purpose area (S3C7528/C7538)

General-Purpose Program Memory

Two program memory areas are allocated for general-purpose use: One area is 16 bytes in size and the other is 3,968 bytes(S3C7524/C7534) or 8,064 bytes(S3C7528/C7538).

Vector Addresses

A 16-byte vector address area is used to store the vector addresses required to execute system resets and interrupts. Start addresses for interrupt service routines are stored in this area, along with the values of the enable memory bank (EMB) and enable register bank (ERB) flags that are used to initialize the corresponding service routines. The 16-byte area can be used alternately as general-purpose ROM.

REF Instructions

Locations 0020H–007FH are used as a reference area (look-up table) for 1-byte REF instructions. The REF instruction reduces the byte size of instruction operands. REF can reference one 2-byte instruction, two 1-byte instructions, and three-byte instructions which are stored in the look-up table. Unused look-up table addresses can be used as general-purpose ROM.

Table 2–1. Program Memory Address Ranges

ROM Area Function	Address Ranges	Area Size (in Bytes)
Vector address area	0000H–000FH	16
General-purpose program memory	0010H–001FH	16
REF instruction look-up table area	0020H–007FH	96
General-purpose program memory	0080H–0FFFH 0080H–1FFFH	3,968 (S3C7524/C7534) 8,064 (S3C7528/C7538)

GENERAL-PURPOSE MEMORY AREAS

The 16-byte area at ROM locations 0010H–001FH and 3,968-byte (S3C7524/C7534) area at ROM locations 0080H–0FFFH or the 8,064-byte (S3C7528/C7538) area at ROM locations 0080H–1FFFH are used as general-purpose program memory. Unused locations in the vector address area and REF instruction look-up table areas can be used as general-purpose program memory. However, care must be taken not to overwrite live data when writing programs that use special-purpose areas of the ROM.

VECTOR ADDRESS AREA

The 16-byte vector address area of the ROM is used to store the vector addresses for executing system resets and interrupts. The starting addresses of interrupt service routines are stored in this area, along with the enable memory bank (EMB) and enable register bank (ERB) flag values that are needed to initialize the service routines. 16-byte vector addresses are organized as follows:

EMB	ERB	0	PC12	PC11	PC10	PC9	PC8
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

To set up the vector address area for specific programs, use the instruction VENTn. The programming tips on the next page explain how to do this.

NOTE: PC12 is not used in the S3C7524/C7534 and the value of PC 12 is always “0”.

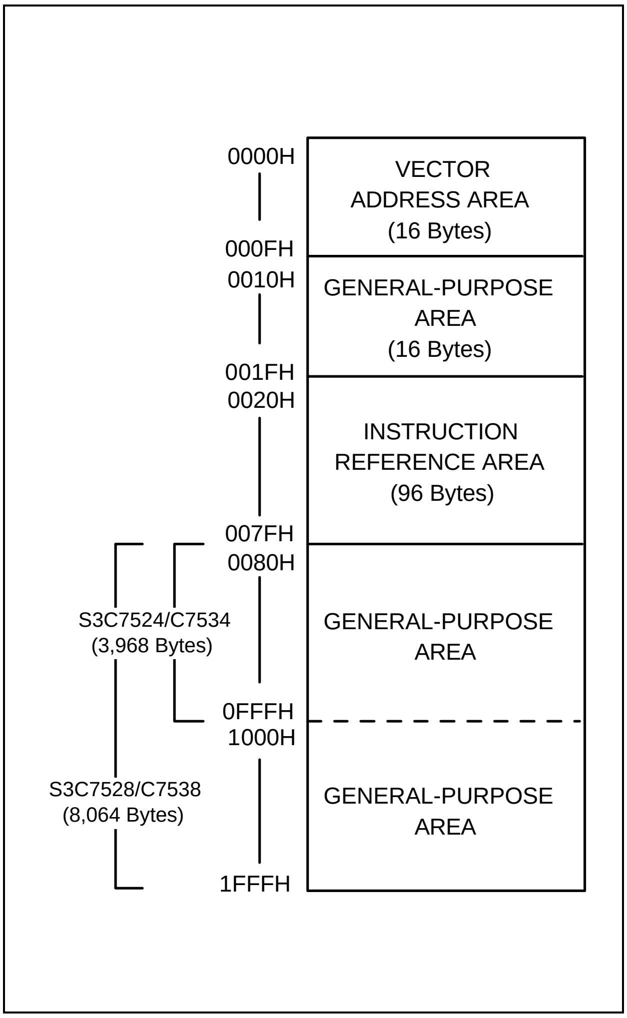
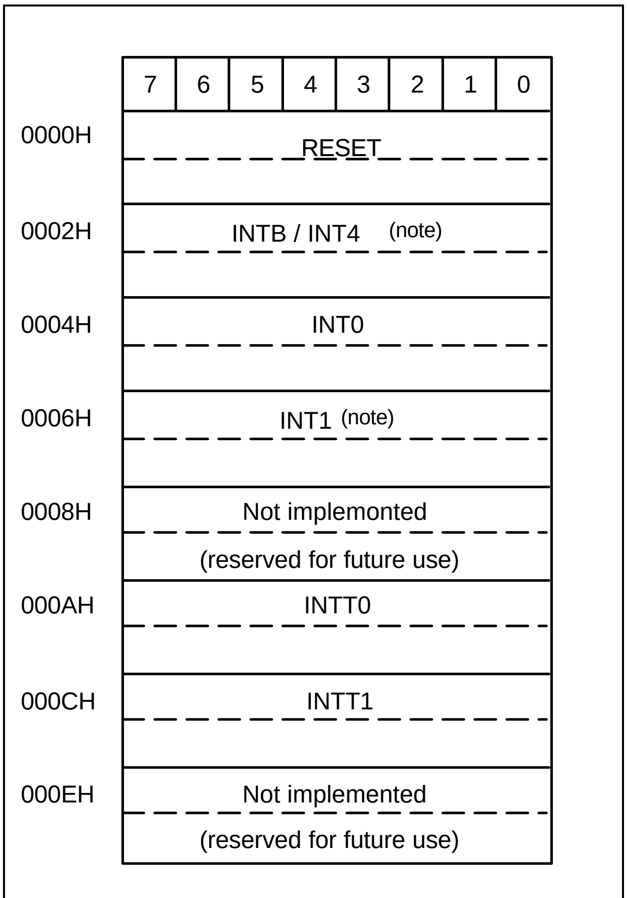


Figure 2–1. ROM Address Structure



NOTE: S3C7534/C7538 does not use the INT1 and INT4 interrupts.

Figure 2–2. Vector Address Map

☞ PROGRAMMING TIP — Defining Vectored Interrupts

The following examples show you several ways you can define the vectored interrupt and instruction reference areas in program memory:

1. When all vector interrupts are used:

```
ORG     0000H

VENT0   1,0,RESET           ;   EMB ← 1, ERB ← 0; Jump to RESET address
VENT1   0,0,INTB            ;   EMB ← 0, ERB ← 0; Jump to INTB address
VENT2   0,0,INT0            ;   EMB ← 0, ERB ← 0; Jump to INT0 address
VENT3   0,0,INT1 (note)     ;   EMB ← 0, ERB ← 0; Jump to INT1 address
NOP
NOP
VENT5   0,0,INTT0           ;   EMB ← 0, ERB ← 0; Jump to INTT0 address
VENT6   0,0,INTT1           ;   EMB ← 0, ERB ← 0; Jump to INTT1 address
```

2. When a specific vectored interrupt such as INT0, and INTT0 is not used, the unused vector interrupt locations must be skipped with the assembly instruction ORG so that jumps will address the correct locations:

```
ORG     0000H

VENT0   1,0,RESET           ;   EMB ← 1, ERB ← 0; Jump to RESET address
VENT1   0,0,INTB            ;   EMB ← 0, ERB ← 0; Jump to INTB address

ORG     0006H               ;   INT0 interrupt not used

VENT3   0,0,INT1 (note)     ;   EMB ← 0, ERB ← 0; Jump to INT1 address

ORG     000CH               ;   INTT0 interrupt not used

VENT6   0,0,INTT1           ;   EMB ← 0, ERB ← 0; Jump to INTT1 address
ORG     0010H
```

3. If an INT0 interrupt is not used and if its corresponding vector interrupt area is not fully utilized, or if it is not written by a ORG instruction as in Example 2, a CPU malfunction will occur:

```
ORG     0000H

VENT0   1,0,RESET           ;   EMB ← 1, ERB ← 0; Jump to RESET address
VENT1   0,0,INTB            ;   EMB ← 0, ERB ← 0; Jump to INTB address
VENT3   0,0,INT1 (note)     ;   EMB ← 0, ERB ← 0; Jump to INT0 address
NOP
NOP
VENT5   0,0,INTT0           ;   EMB ← 0, ERB ← 0; Jump to INT1 address
VENT6   0,0,INTT1           ;   EMB ← 0, ERB ← 0; Jump to INTT0 address

ORG     0010H

General-purpose ROM area
```

In this example, when an INTT0 interrupt is generated, the corresponding vector area is not VENT5 INTT0, but VENT6 INTT1. This causes an INTT0 interrupt to jump incorrectly to the INTT1 address and causes a CPU malfunction to occur.

NOTE: S3C7534/C7538 does not use the INT1 interrupt.

INSTRUCTION REFERENCE AREA

Using 1-byte REF instructions, you can easily reference instructions with larger byte sizes that are stored in addresses 0020H–007FH of program memory. This 96-byte area is called the REF instruction reference area, or look-up table. Locations in the REF look-up table may contain two one-byte instructions, a single two-byte instruction, or three-byte instruction such as a JP (jump) or CALL. The starting address of the instruction you are referencing must always be an even number. To reference a JP or CALL instruction, it must be written to the reference area in a two-byte format: for JP, this format is TJP; for CALL, it is TCALL.

You can use REF instructions to execute instructions larger than one byte. There are tree ways you can use REF instruction:

— Using the 1-byte REF instruction to execute one 2-byte or two 1-byte instructions,

— Branching to any location by referencing a branch instruction stored in the look-up table,

— Calling subroutines at any location by referencing a call instruction stored in the look-up table.

☞ Programming Tip — Using the REF Look-Up Table

Here is one example of how to use the REF instruction look-up table:

```
                ORG      0020H

JMAIN           TJP      MAIN               ;    0, MAIN
KEYCK           BTSF     KEYFG              ;    1, KEYFG check
WATCH           TCALL    CLOCK              ;    2, call CLOCK
INCHL           LD       @HL,A              ;    3, (HL) ← A
                INCS     HL
                •
                •
                •
ABC             LD       EA,#00H            ;    47, EA ← #00H
                ORG      0080

MAIN            NOP
                NOP
                •
                •
                •
                REF      KEYCK              ;    BTSF   KEYFG (1-byte instruction)
                REF      JMAIN              ;    KEYFG  =  1, jump to MAIN (1-byte instruction)
                REF      WATCH              ;    KEYFG  =  0, call CLOCK (1-byte instruction)
                REF      INCHL              ;    LD   @HL,A
                                            ;    INCS   HL
                REF      ABC                ;    LD   EA,#00H (1-byte instruction)
                •
                •
                •
```

DATA MEMORY (RAM)

OVERVIEW

In its standard configuration, the 896 × 4-bit data memory has five areas:

- 32 × 4-bit working register area
- 224 × 4-bit general-purpose area (also used as stack area)
- 2 × 256 × 4-bit general-purpose area
- 128 × 4-bit area for peripheral hardware

To make it easier to reference, the data memory area has four memory banks — bank 0, bank 1, bank 2, and bank 15. The select memory bank instruction (SMB) is used to select the bank you want to select as working data memory. Data stored in RAM locations are 1-, 4-, and 8-bit addressable.

Initialization values for the data memory area are not defined by hardware and must therefore be initialized by program software following RESET. However, when RESET signal is generated in power-down mode, the data memory contents are held.

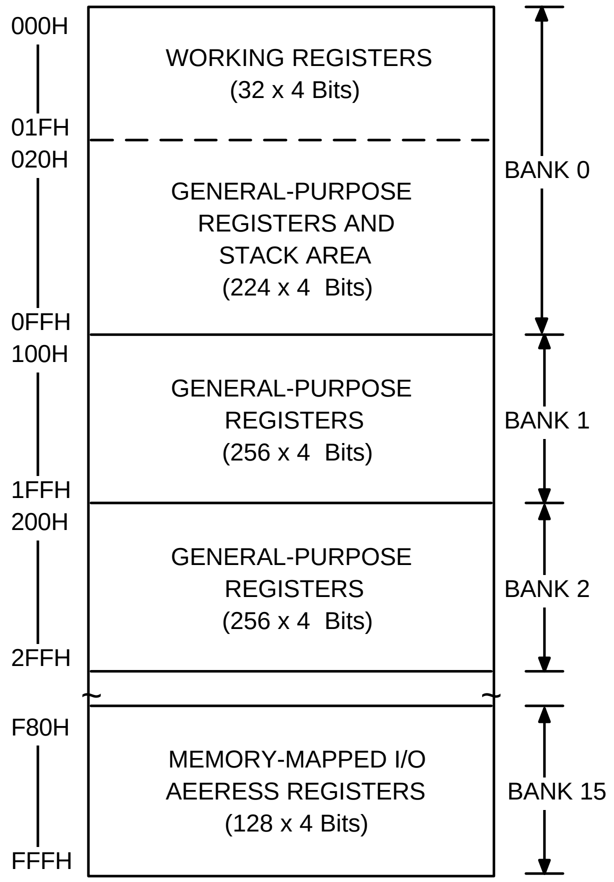


Figure 2–3. Data Memory (RAM) Map

Memory Banks 0, 1, 2, and 15

Bank 0	(000H–0FFH)	The lowest 32 nibbles of bank 0 (000H–01FH) are used as working registers; the next 224 nibbles (020H–0FFH) can be used both as stack area and as general-purpose data memory. Use the stack area for implementing subroutine calls and returns, and for interrupt processing.
Bank 1	(100H–1FFH)	The 256 nibbles of bank 1 (100H–1FFH) are for general-purpose use.
Bank 2	(200H–2FFH)	The 256 nibbles of bank 2 (200H–2FFH) are for general-purpose use
Bank 15	(F80H–FFFH)	The microcontroller uses bank 15 for memory-mapped peripheral I/O. Fixed RAM locations for each peripheral hardware register: the port latches, timers, peripherals controls, etc. are mapped into this area.

Data Memory Addressing Modes

The enable memory bank (EMB) flag controls the addressing mode for data memory banks 0, 1, 2 or 15. When the EMB flag is logic zero, the addressable area is restricted to specific locations, depending on whether direct or indirect addressing is used. With direct addressing, you can access locations 000H–07FH of bank 0 and bank 15. With indirect addressing, only bank 0 (000H–0FFH) can be accessed. When the EMB flag is set to logic one, all four data memory banks can be accessed according to the current SMB value.

For 8-bit addressing, two 4-bit registers are addressed as a register pair. Also, when using 8-bit instructions to address RAM locations, remember to use the even-numbered register address as the instruction operand.

Working Registers

The RAM working register area in data memory bank 0 is further divided into four *register* banks (bank 0, 1, 2, and 3). Each register bank has eight 4-bit registers and paired 4-bit registers are 8-bit addressable.

Register A is used as a 4-bit accumulator and register pair EA as an 8-bit extended accumulator. The carry flag bit can also be used as a 1-bit accumulator. Register pairs WX, WL, and HL are used as address pointers for indirect addressing. To limit the possibility of data corruption due to incorrect register addressing, it is advisable to use register bank 0 for the main program and banks 1, 2, and 3 for interrupt service routines.

Table 2–2. Data Memory Organization and Addressing

Addresses	Register Areas	Bank	EMB Value	SMB Value
000H–01FH	Working registers	0	0, 1	0
020H–0FFH	Stack and general-purpose registers			
100H–1FFH	General-purpose registers	1	1	1
200H–2FFH	General-purpose registers	2	1	2
F80H–FFFH	Peripheral hardware registers	15	0, 1	15

☞ PROGRAMMING TIP — Clearing Data Memory Banks 0 and 1

Clear banks 0 and 1 of the data memory area:

```
RAMCLR   SMB    1          ;   RAM (100H–1FFH) clear
         LD     HL,#00H
         LD     A,#0H
RMCL1    LD     @HL,A
         INCS   HL
         JR     RMCL1

         SMB    0          ;   RAM (010H–0FFH) clear
         LD     HL,#10H
RMCL0    LD     @HL,A
         INCS   HL
         JR     RMCL0
```

WORKING REGISTERS

Working registers, mapped to RAM address 000H–01FH in data memory bank 0, are used to temporarily store intermediate results during program execution, as well as pointer values used for indirect addressing. Unused registers may be used as general-purpose memory. Working register data can be manipulated as 1-bit units, 4-bit units or, using paired registers, as 8-bit units.

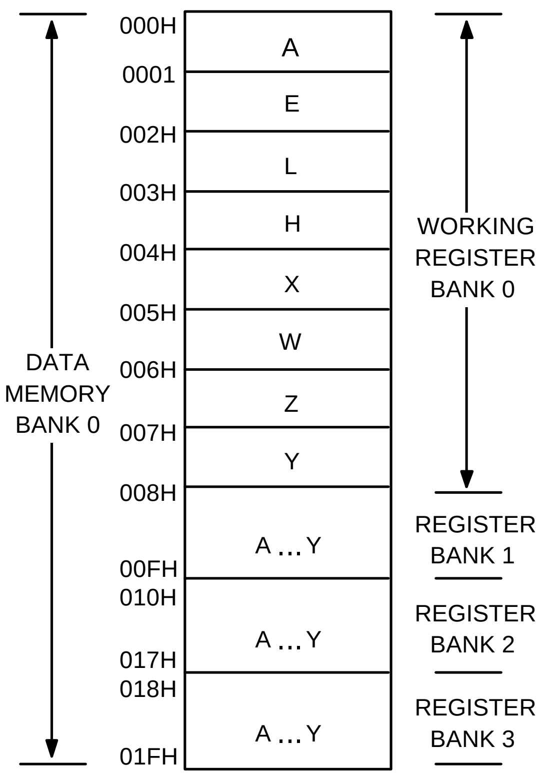


Figure 2–4. Working Register Map

Working Register Banks

For addressing purposes, the working register area is divided into four register banks — bank 0, bank 1, bank 2, and bank 3. Any one of these banks can be selected as the working register bank by the register bank selection instruction (SRB n) and by setting the status of the register bank enable flag (ERB).

Generally, working register bank 0 is used for the main program, and banks 1, 2, and 3 for interrupt service routines. Following this convention helps to prevent possible data corruption during program execution due to contention in register bank addressing.

Table 2–3. Working Register Organization and Addressing

ERB Setting	SRB Settings				Selected Register Bank
	3	2	1	0	
0	0	0	x	x	Always set to bank 0
1	0	0	0	0	Bank 0
			0	1	Bank 1
			1	0	Bank 2
			1	1	Bank 3

NOTE: x = not applicable.

Paired Working Registers

Each of the register banks is subdivided into eight 4-bit registers. These registers, named Y, Z, W, X, H, L, E and A, can either be manipulated individually using 4-bit instructions, or together as register pairs for 8-bit data manipulation.

The names of the 8-bit register pairs in each register bank are EA, HL, WX, YZ and WL. Registers A, L, X and Z always become the lower nibble when registers are addressed as 8-bit pairs. This makes a total of eight 4-bit registers or four 8-bit double registers in each of the four working register banks.

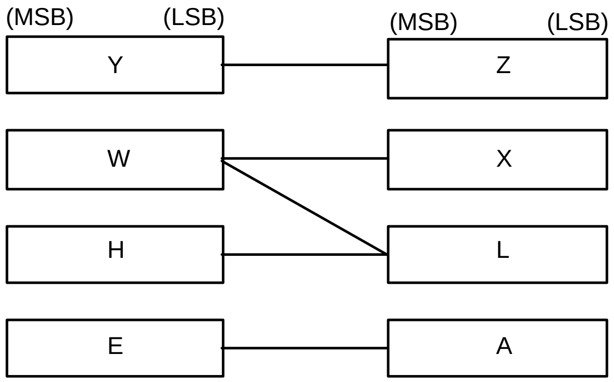


Figure 2–5. Register Pair Configuration

Special-Purpose Working Registers

Register A is used as a 4-bit accumulator and double register EA as an 8-bit accumulator. The carry flag can also be used as a 1-bit accumulator.

8-bit double registers WX, WL and HL are used as data pointers for indirect addressing. When the HL register serves as a data pointer, the instructions LDI, LDD, XCHI, and XCHD can make very efficient use of working registers as program loop counters by letting you transfer a value to the L register and increment or decrement it using a single instruction.

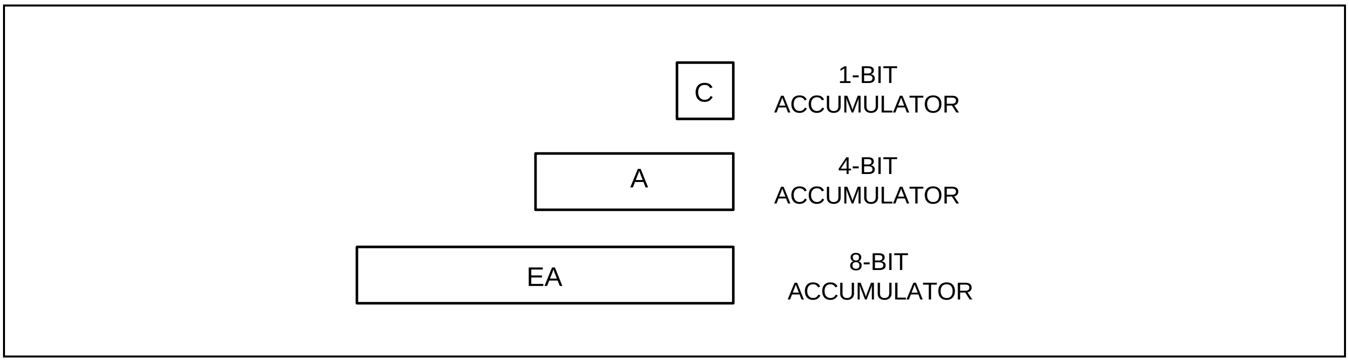


Figure 2–6. 1-Bit, 4-Bit, and 8-Bit Accumulator

Recommendation for Multiple Interrupt Processing

If more than four interrupts are being processed at one time, you can avoid possible loss of working register data by using the PUSH RR instruction to save register contents to the stack before the service routines are executed in the same register bank. When the routines have executed successfully, you can restore the register contents from the stack to working memory using the POP instruction.

☞ PROGRAMMING TIP — Selecting the Working Register Area

The following examples show the correct programming method for selecting working register area:

1. When ERB = "0":

```
VENT2       1,0,INT0                ;  EMB ← 1, ERB ← 0, Jump to INT0 address
;
INT0        PUSH    SB              ;  PUSH current SMB, SRB
            SRB     2               ;  Instruction does not execute because ERB = "0"
            PUSH    HL              ;  PUSH HL register contents to stack
            PUSH    WX              ;  PUSH WX register contents to stack
            PUSH    YZ              ;  PUSH YZ register contents to stack
            PUSH    EA              ;  PUSH EA register contents to stack
            SMB     0
            LD      EA,#00H
            LD      80H,EA
            LD      HL,#40H
            INCS    HL
            LD      WX,EA
            LD      YZ,EA
            POP     EA              ;  POP EA register contents from stack
            POP     YZ              ;  POP YZ register contents from stack
            POP     WX              ;  POP WX register contents from stack
            POP     HL              ;  POP HL register contents from stack
            POP     SB              ;  POP current SMB, SRB
            IRET
```

The POP instructions execute alternately with the PUSH instructions. If an SMB n instruction is used in an interrupt service routine, a PUSH and POP SB instruction must be used to store and restore the current SMB and SRB values, as shown in Example 2 below.

2. When ERB = "1":

```
VENT2       1,1,INT0                ;  EMB ← 1, ERB ← 1, Jump to INT0 address
;
INT0        PUSH    SB              ;  Store current SMB, SRB
            SRB     2               ;  Select register bank 2 because of ERB = "1"
            SMB     0
            LD      EA,#00H
            LD      80H,EA
            LD      HL,#40H
            INCS    HL
            LD      WX,EA
            LD      YZ,EA
            POP     SB              ;  Restore SMB, SRB
            IRET
```

☞ PROGRAMMING TIP — Selecting the Working Register Area

The following examples show the correct programming method for selecting working register area:

1. When ERB = "0":

```
VENT2   1,0,INT0          ; EMB ← 1, ERB ← 0, Jump to INT0 address

INT0    PUSH    SB        ; PUSH current SMB, SRB
        SRB     2         ; Instruction does not execute because ERB = "0"
        PUSH    HL        ; PUSH HL register contents to stack
        PUSH    WX        ; PUSH WX register contents to stack
        PUSH    YZ        ; PUSH YZ register contents to stack
        PUSH    EA        ; PUSH EA register contents to stack
        SMB     0
        LD      EA,#00H
        LD      80H,EA
        LD      HL,#40H
        INCS    HL|
        LD      WX,EA
        LD      YZ,EA
        POP     EA        ; POP EA register contents from stack
        POP     YZ        ; POP YZ register contents from stack
        POP     WX        ; POP WX register contents from stack
        POP     HL        ; POP HL register contents from stack
        POP     SB        ; POP current SMB, SRB
        IRET
```

The POP instructions execute alternately with the PUSH instructions. If an SMB n instruction is used in an interrupt service routine, a PUSH and POP SB instruction must be used to store and restore the current SMB and SRB values, as shown in Example 2 below.

2. When ERB = "1":

```
VENT2   1,1,INT0          ; EMB ← 1, ERB ← 1, Jump to INT0 address

INT0    PUSH    SB        ; Store current SMB, SRB
        SRB     2         ; Select register bank 2 because of ERB = "1"
        SMB     0
        LD      EA,#00H
        LD      80H,EA
        LD      HL,#40H
        INCS    HL
        LD      WX,EA
        LD      YZ,EA
        POP     SB        ; Restore SMB, SRB
        IRET
```

STACK OPERATIONS

STACK POINTER (SP)

The stack pointer (SP) is an 8-bit register that stores the address used to access the stack, an area of data memory set aside for temporary storage of stack addresses. The SP can be read or written by 8-bit control instructions. When addressing the SP, bit 0 must always remain cleared to logic zero.

F80H	SP3	SP2	SP1	"0"
F81H	SP7	SP6	SP5	SP4

There are two basic stack operations: writing to the top of the stack (push), and reading from the top of the stack (pop). A push decrements the SP and a pop increments it so that the SP always points to the top address of the last data to be written to the stack.

The program counter contents and program status word are stored in the stack area prior to the execution of a CALL or a PUSH instruction, or during interrupt service routines. Stack operation is a LIFO (Last In-First Out) type. The stack area is located in general-purpose data memory bank 0.

During an interrupt or a subroutine, the PC value and the PSW are saved to the stack area. When the routine has completed, the stack pointer is referenced to restore the PC and PSW, and the next instruction is executed.

The SP can address stack registers in bank 0 (addresses 000H-0FFH) regardless of the current value of the enable memory bank (EMB) flag and the select memory bank (SMB) flag. Although general-purpose register areas can be used for stack operations, be careful to avoid data loss due to simultaneous use of the same register(s).

Since the reset value of the stack pointer is not defined in firmware, we recommend that you initialize the stack pointer by program code to location 00H. This sets the first register of the stack area to 0FFH.

NOTE

A subroutine call occupies six nibbles in the stack; an interrupt requires six. When subroutine nesting or interrupt routines are used continuously, the stack area should be set in accordance with the maximum number of subroutine levels. To do this, estimate the number of nibbles that will be used for the subroutines or interrupts and set the stack area correspondingly.

☞ PROGRAMMING TIP — Initializing the Stack Pointer

To initialize the stack pointer (SP):

1. When EMB = "1":

```
SMB     15          ;   Select memory bank 15
LD      EA,#00H     ;   Bit 0 of accumulator A is always cleared to "0"
LD      SP,EA       ;   Stack area initial address (0FFH) ← (SP) – 1
```

2. When EMB = "0":

```
LD      EA,#00H
LD      SP,EA       ;   Memory addressing area  (00H–7FH, F80H–FFFH)
```

PUSH OPERATIONS

Three kinds of push operations reference the stack pointer (SP) to write data from the source register to the stack: PUSH instructions, CALL instructions, and interrupts. In each case, the SP is *decreased* by a number determined by the type of push operation and then points to the next available stack location.

PUSH Instructions

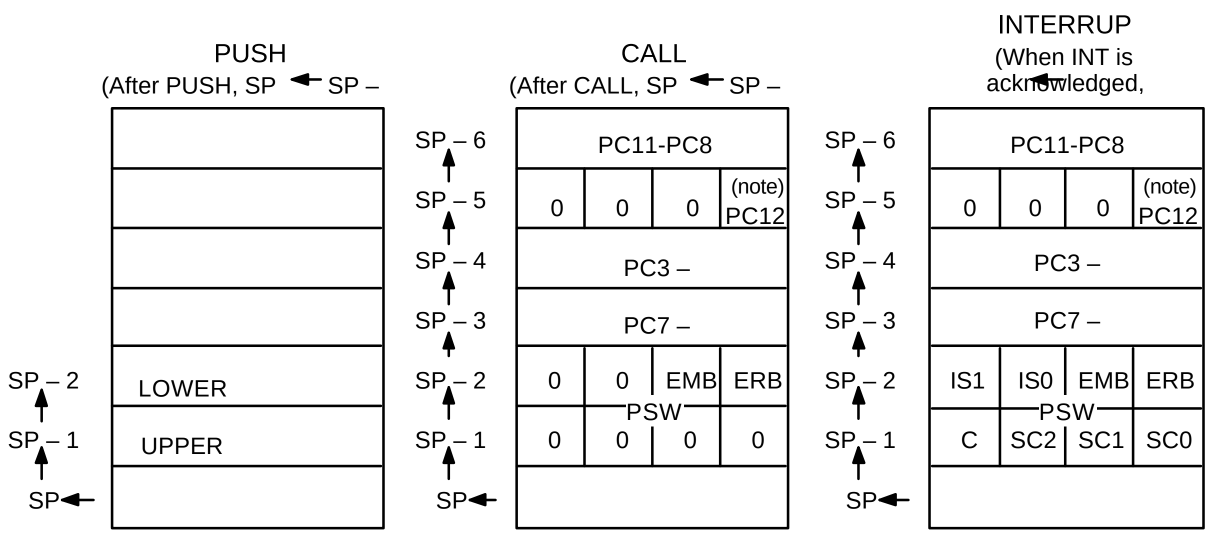
A PUSH instruction references the SP to write two 4-bit data nibbles to the stack. Two 4-bit stack addresses are referenced by the stack pointer: one for the upper register value and another for the lower register. After the PUSH has executed, the SP is decreased *by two* and points to the next available stack location.

CALL Instructions

When a subroutine call is issued, the CALL instruction references the SP to write the PC's contents to six 4-bit stack locations. Current values for the enable memory bank (EMB) flag and the enable register bank (ERB) flag are also pushed to the stack. Since six 4-bit stack locations are used per CALL, you may nest subroutine calls up to the number of levels permitted in the stack.

Interrupt Routines

An interrupt routine references the SP to push the contents of the PC and the program status word (PSW) to the stack. Six 4-bit stack locations are used to store this data. After the interrupt has executed, the SP is decreased *by six* and points to the next available stack location. During an interrupt sequence, subroutines may be nested up to the number of levels which are permitted in the stack area.



NOTE: S3C7524/C7534 does not use PC12.

Figure 2–7. Push-Type Stack Operations

POP OPERATIONS

For each push operation there is a corresponding pop operation to write data from the stack back to the source register or registers: for the PUSH instruction it is the POP instruction; for CALL, the instruction RET or SRET; for interrupts, the instruction IRET. When a pop operation occurs, the SP is *incremented* by a number determined by the type of operation and points to the next free stack location.

POP Instructions

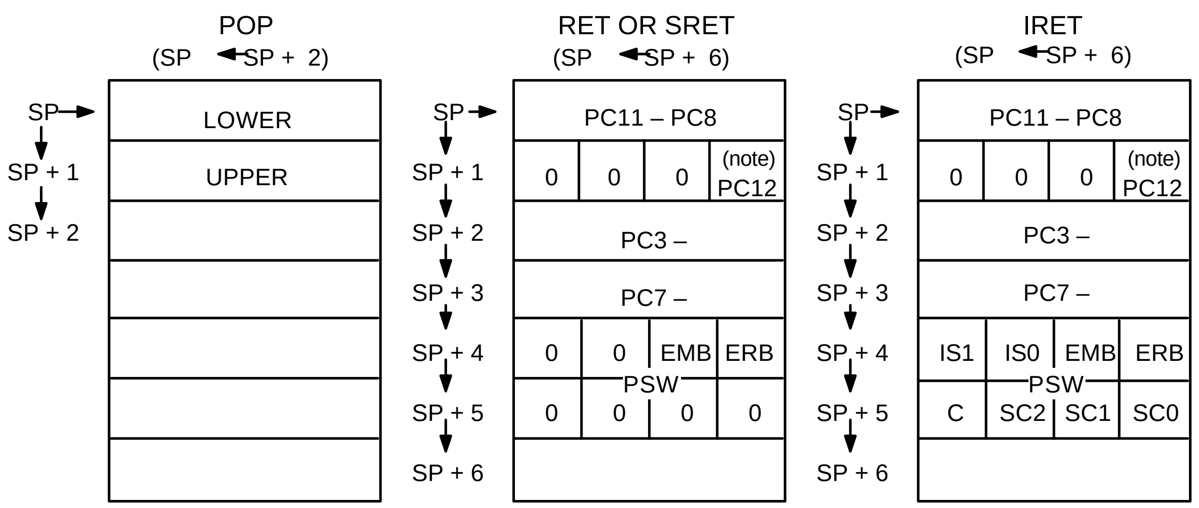
A POP instruction references the SP to write data stored in two 4-bit stack locations back to the register pairs and SB register. The value of the lower 4-bit register is popped first, followed by the value of the upper 4-bit register. After the POP has executed, the SP is incremented *by two* and points to the next free stack location.

RET and SRET Instructions

The end of a subroutine call is signaled by the return instruction, RET or SRET. The RET or SRET uses the SP to reference the six 4-bit stack locations used for the CALL and to write this data back to the PC, the EMB, and the ERB. After the RET or SRET has executed, the SP is incremented *by six* and points to the next free stack location.

IRET Instructions

The end of an interrupt sequence is signaled by the instruction IRET. IRET references the SP to locate the six 4-bit stack addresses used for the interrupt and to write this data back to the PC and the PSW. After the IRET has executed, the SP is incremented *by six* and points to the next free stack location.



NOTE: S3C7524/C7534 does not use PC12.

Figure 2–8. Pop-Type Stack Operations

BIT SEQUENTIAL CARRIER (BSC)

The bit sequential carrier (BSC) is a 8-bit general register that can be manipulated using 1-, 4-, and 8-bit RAM control instructions. RESET clears all BSC bit values to logic zero.

Using the BSC, you can specify sequential addresses and bit locations using 1-bit indirect addressing (memb.@L). (Bit addressing is independent of the current EMB value.) This way, programs can process 8-bit data by moving the bit location sequentially and then incrementing or decreasing the value of the L register.

BSC data can also be manipulated using direct addressing.

If the values of the L register are 0H at BSC2.@L, the address and bit location assignment is FC2H.0. If the L register content is 8H at BSC2.@L, the address and bit location assignment is FC3H.3.

Table 2–4. BSC Register Organization

Name	Address	Bit 3	Bit 2	Bit 1	Bit 0
BSC0	FC0H	BSC0.3	BSC0.2	BSC0.1	BSC0.0
BSC1	FC1H	BSC1.3	BSC1.2	BSC1.1	BSC1.0
BSC2	FC2H	BSC2.3	BSC2.2	BSC2.1	BSC2.0
BSC3	FC3H	BSC3.3	BSC3.2	BSC3.1	BSC3.0

☞ PROGRAMMING TIP — Using the BSC Register to Output 16-Bit Data

To use the bit sequential carrier (BSC) register to output 8-bit data (59H) to the P2.3 pin:

```
            BITS    EMB
            SMB     15
            LD      EA,#59H         ;
            LD      BSC2,EA         ;   BSC2 ← A, BSC3 ← E
            SMB     0
            LD      L,#8H           ;
AGN         LDB     C,BSC2.@L       ;
            LDB     P2.3,C          ;   P2.3 ← C
            INCS    L

            JR      AGN

            RET
```

PROGRAM COUNTER (PC)

A 13-bit program counter (PC) stores addresses for instruction fetches during program execution. Whenever a reset operation or an interrupt occurs, bits PC11 (S3C7524/C7534) or bits PC12 (S3C7528/C7538) through PC0 are set to the vector address.

Usually, the PC is incremented by the number of bytes of the instruction being fetched. One exception is the 1-byte REF instruction which is used to reference instructions stored in the ROM.

PROGRAM STATUS WORD (PSW)

The program status word (PSW) is an 8-bit word that defines system status and program execution status and which permits an interrupted process to resume operation after an interrupt request has been serviced. PSW values are mapped as follows:

FB0H	IS1	IS0	EMB	ERB
FB1H	C	SC2	SC1	SC0

The PSW can be manipulated by 1-bit or 4-bit read/write and by 8-bit read instructions, depending on the specific bit or bits being addressed. The PSW can be addressed during program execution regardless of the current value of the enable memory bank (EMB) flag.

Part or all of the PSW is saved to stack prior to execution of a subroutine call or hardware interrupt. After the interrupt has been processed, the PSW values are popped from the stack back to the PSW address.

When a RESET is generated, the EMB and ERB values are set according to the RESET vector address, and the carry flag is left undefined (or the current value is retained). PSW bits IS0, IS1, SC0, SC1, and SC2 are all cleared to logical zero.

Table 2–5. Program Status Word Bit Descriptions

PSW Bit Identifier	Description	Bit Addressing	Read/Write
IS1, IS0	Interrupt status flags	1, 4	R/W
EMB	Enable memory bank flag	1	R/W
ERB	Enable register bank flag	1	R/W
C	Carry flag	1	R/W
SC2, SC1, SC0	Program skip flags	8	R

INTERRUPT STATUS FLAGS (IS0, IS1)

PSW bits IS0 and IS1 contain the current interrupt execution status values. You can manipulate IS0 and IS1 flags directly using 1-bit RAM control instructions

By manipulating interrupt status flags in conjunction with the interrupt priority register (IPR), you can process multiple interrupts by anticipating the next interrupt in an execution sequence. The interrupt priority control circuit determines the IS0 and IS1 settings in order to control multiple interrupt processing. When both interrupt status flags are set to "0", all interrupts are allowed. The priority with which interrupts are processed is then determined by the IPR.

When an interrupt occurs, IS0 and IS1 are pushed to the stack as part of the PSW and are automatically incremented to the next status. Then, when the interrupt service routine ends with an IRET instruction, IS0 and IS1 values are restored to the PSW. Table 2–6 shows the effects of IS0 and IS1 flag settings.

Table 2–6. Interrupt Status Flag Bit Settings

IS1 Value	IS0 Value	Status of Currently Executing Process	Effect of IS0 and IS1 Settings on Interrupt Request Control
0	0	0	All interrupt requests are serviced
0	1	1	Only high-priority interrupt as determined in the interrupt priority register (IPR) is serviced
1	0	2	No more interrupt requests are serviced
1	1	–	Not applicable; these bit settings are undefined

Since interrupt status flags can be addressed by write instructions, programs can exert direct control over interrupt processing status. Before interrupt status flags can be addressed, however, you must first execute a DI instruction to inhibit additional interrupt routines. When the bit manipulation has been completed, execute an EI instruction to re-enable interrupt processing.

☞ PROGRAMMING TIP — Setting ISx Flags for Interrupt Processing

The following instruction sequence shows how to use the IS0 and IS1 flags to control interrupt processing:

```
INTB    DI                ;   Disable interrupt
        BITR   IS1        ;   IS1 ← 0
        BITS   IS0        ;   Allow interrupts according to IPR priority level
        EI                ;   Enable interrupt
```

EMB FLAG (EMB)

The EMB flag is used to enable whether the memory bank selected by SMB register is to be valid or not. In this way, it controls the addressing mode for data memory banks 0, 1, or 15.

When the EMB flag is "0", the data memory address space is restricted to bank 15 and addresses 000H–07FH of memory bank 0, regardless of the SMB register contents. When the EMB flag is set to "1", the general-purpose areas of bank 0, 1, and 15 can be accessed by using the appropriate SMB value.

☞ PROGRAMMING TIP — Using the EMB Flag to Select Memory Banks

EMB flag settings for memory bank selection:

1. When EMB = "0":

```
SMB   1        ;  Non-essential instruction since EMB = "0"
LD    A,#9H
LD    90H,A    ;  (F90H) ← A, bank 15 is selected
LD    34H,A    ;  (034H) ← A, bank 0 is selected
SMB   0        ;  Non-essential instruction since EMB = "0"
LD    90H,A    ;  (F90H) ← A, bank 15 is selected
LD    34H,A    ;  (034H) ← A, bank 0 is selected
SMB   15       ;  Non-essential instruction, since EMB = "0"
LD    20H,A    ;  (020H) ← A, bank 0 is selected
LD    90H,A    ;  (F90H) ← A, bank 15 is selected
```

2. When EMB = "1":

```
SMB   1        ;  Select memory bank 1
LD    A,#9H
LD    90H,A    ;  (190H) ← A, bank 1 is selected
LD    34H,A    ;  (134H) ← A, bank 1 is selected
SMB   0        ;  Select memory bank 0
LD    90H,A    ;  (090H) ← A, bank 0 is selected
LD    34H,A    ;  (034H) ← A, bank 0 is selected
SMB   15       ;  Select memory bank 15
LD    20H,A    ;  Program error, but assembler does not detect it
LD    90H,A    ;  (F90H) ← A, bank 15 is selected
```

ERB FLAG (ERB)

The 1-bit register bank enable flag (ERB) determines the range of addressable working register area. When the ERB flag is "1", the working register area from register banks 0 to 3 is selected according to the register bank selection register (SRB). When the ERB flag is "0", register bank 0 is the selected working register area, regardless of the current value of the register bank selection register (SRB).

When an internal RESET is generated, bit 6 of program memory address 0000H is written to the ERB flag. This automatically initializes the flag. When a vectored interrupt is generated, bit 6 of the respective address table in program memory is written to the ERB flag, setting the correct flag status before the interrupt service routine is executed.

During the interrupt routine, the ERB value is automatically pushed to the stack area along with the other PSW bits. Afterwards, it is popped back to the FB0H.0 bit location in the PSW. The initial ERB flag settings for each vectored interrupt are defined using VENTn instructions.

☞ PROGRAMMING TIP — Using the ERB Flag to Select Register Banks

ERB flag settings for register bank selection:

1. When ERB = "0":

```
SRB    1          ;   Register bank 0 is selected (since ERB = "0", the
                  ;   SRB is configured to bank 0)
LD     EA,#34H    ;   Bank 0 EA ← #34H
LD     HL,EA      ;   Bank 0 HL ← EA
SRB    2          ;   Register bank 0 is selected
LD     YZ,EA      ;   Bank 0 YZ ← EA
SRB    3          ;   Register bank 0 is selected
LD     WX,EA      ;   Bank 0 WX ← EA
```

2. When ERB = "1":

```
SRB    1          ;   Register bank 1 is selected
LD     EA,#34H    ;   Bank 1 EA ← #34H
LD     HL,EA      ;   Bank 1 HL ← Bank 1 EA
SRB    2          ;   Register bank 2 is selected
LD     YZ,EA      ;   Bank 2 YZ ← BANK2 EA
SRB    3          ;   Register bank 3 is selected
LD     WX,EA      ;   Bank 3 WX ← Bank 3 EA
```

SKIP CONDITION FLAGS (SC2, SC1, SC0)

The skip condition flags SC2, SC1, and SC0 indicate the current program skip conditions and are set and reset automatically during program execution. Skip condition flags can only be addressed by 8-bit read instructions. Direct manipulation of the SC2, SC1, and SC0 bits is not allowed.

CARRY FLAG (C)

The carry flag is used to save the result of an overflow or borrow when executing arithmetic instructions involving a carry (ADC, SBC). The carry flag can also be used as a 1-bit accumulator for performing Boolean operations involving bit-addressed data memory.

If an overflow or borrow condition occurs when executing arithmetic instructions with carry (ADC, SBC), the carry flag is set to "1". Otherwise, its value is "0". When a RESET occurs, the current value of the carry flag is retained during power-down mode, but when normal operating mode resumes, its value is undefined.

The carry flag can be directly manipulated by predefined set of 1-bit read/write instructions, independent of other bits in the PSW. Only the ADC and SBC instructions, and the instructions listed in Table 2–7, affect the carry flag.

Table 2–7. Valid Carry Flag Manipulation Instructions

Operation Type	Instructions	Carry Flag Manipulation
Direct manipulation	SCF	Set carry flag to "1"
	RCF	Clear carry flag to "0" (reset carry flag)
	CCF	Invert carry flag value (complement carry flag)
	BTST C	Test carry and skip if C = "1"
Bit transfer	LDB (operand) [(1)],C	Load carry flag value to the specified bit
	LDB C,(operand) [(1)]	Load contents of the specified bit to carry flag
Data transfer	RRC A	Rotate Right through carry flag
Boolean manipulation	BAND C,(operand) [(1)]	AND the specified bit with contents of carry flag and save the result to the carry flag
	BOR C,(operand) [(1)]	OR the specified bit with contents of carry flag and save the result to the carry flag
	BXOR C,(operand) [(1)]	XOR the specified bit with contents of carry flag and save the result to the carry flag
Interrupt routine	INTn [(2)]	Save carry flag to stack with other PSW bits
Return from interrupt	IRET	Restore carry flag from stack with other PSW bits

NOTES
1. The operand has three bit addressing formats: mema.a, memb.@L, and @H + DA.b.
2. 'INTn' refers to the specific interrupt being executed and is not an instruction.

☞ PROGRAMMING TIP — Using the Carry Flag as a 1-Bit Accumulator

1. Set the carry flag to logic one:

```
SCF                      ;   C ← 1
LD      EA,#0C3H         ;   EA ← #0C3H
LD      HL,#0AAH         ;   HL ← #0AAH
ADC     EA,HL            ;   EA ← #0C3H + #0AAH + #1H, C ← 1
```

2. Logical-AND bit 3 of address 3FH with P3.3 and output the result to P5.0:

```
LD      H,#3H            ;   Set the upper four bits of the address to the H register value
LDB     C,@H+0FH.3       ;   C ← bit 3 of 3FH
BAND    C,P3.3           ;   C ← C AND P3.3
LDB     P5.0,C           ;   Output result from carry flag to P5.0
```

3 ADDRESSING MODES

OVERVIEW

The enable memory bank flag, EMB, controls the two addressing modes for data memory. When the EMB flag is set to logic one, you can address the entire RAM area; when the EMB flag is cleared to logic zero, the addressable area in the RAM is restricted to specific locations.

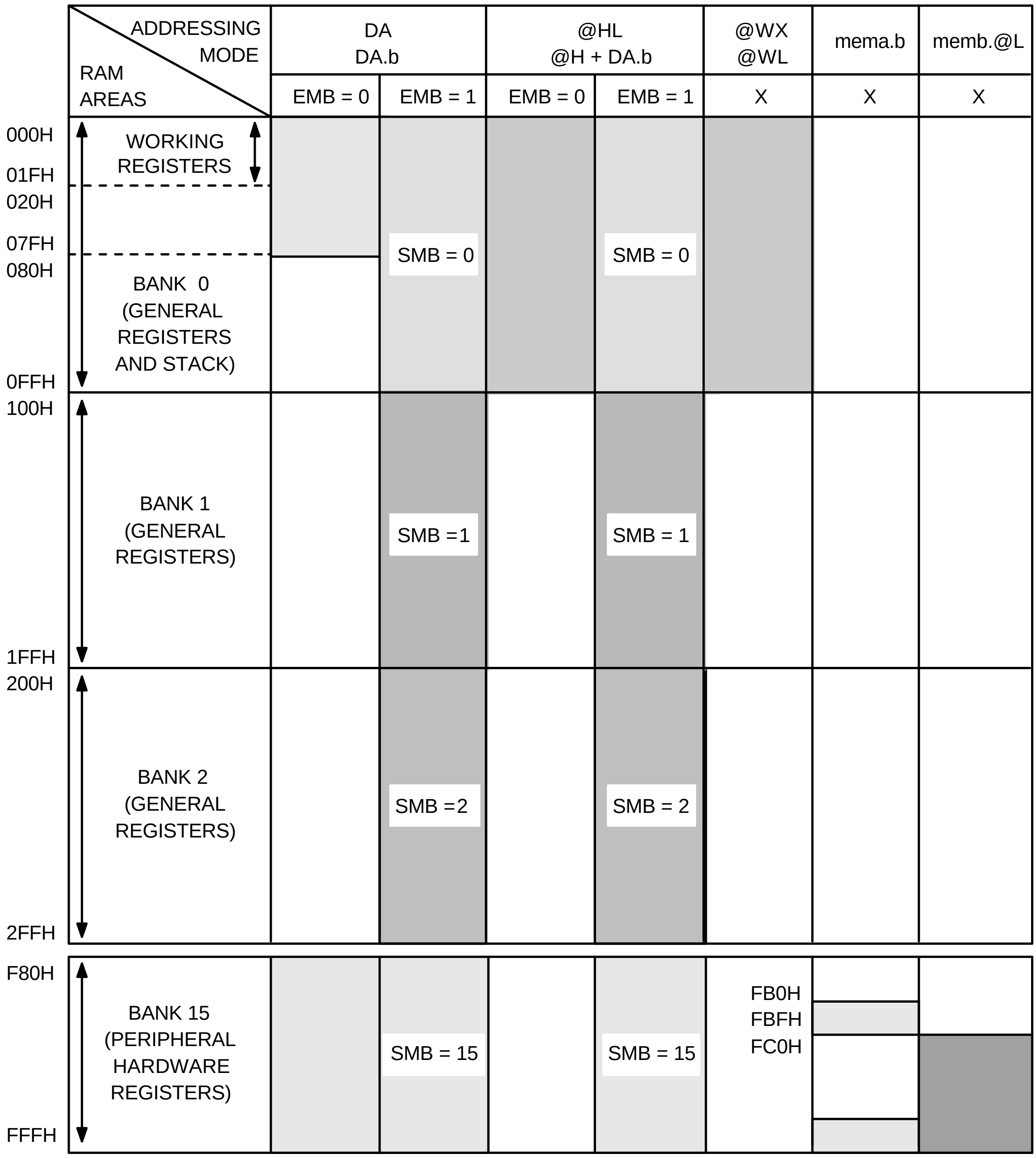
The EMB flag works in connection with the select memory bank instruction, SMBn. You will recall that the SMBn instruction is used to select RAM bank 0, 1, 2 or 15. The SMB setting is always contained in the upper four bits of a 12-bit RAM address. For this reason, both addressing modes (EMB = "0" and EMB = "1") apply specifically to the memory bank indicated by the SMB instruction, and any restrictions to the addressable area within banks 0, 1, 2 or 15. Direct and indirect 1-bit, 4-bit, and 8-bit addressing methods can be used. Several RAM locations are addressable at all times, regardless of the current EMB flag setting.

Here are a few guidelines to keep in mind regarding data memory addressing:

— When you address peripheral hardware locations in bank 15, the mnemonic for the memory-mapped hardware component can be used as the operand in place of the actual address location.

— Always use an even-numbered RAM address as the operand in 8-bit direct and indirect addressing.

— With direct addressing, use the RAM address as the instruction operand; with indirect addressing, the instruction specifies a register which contains the operand's address.



NOTES
1. 'X' means don't care.
2. Blank columns indicate RAM areas that are not addressable, given the addressing method and enable memory bank (EMB) flag setting shown in the column headers.

Figure 3–1. RAM Address Structure

EMB AND ERB INITIALIZATION VALUES

The EMB and ERB flag bits are set automatically by the values of the RESET vector address and the interrupt vector address. When a RESET is generated internally, bit 7 of program memory address 0000H is written to the EMB flag, initializing it automatically. When a vectored interrupt is generated, bit 7 of the respective vector address table is written to the EMB. This automatically sets the EMB flag status for the interrupt service routine. When the interrupt is serviced, the EMB value is automatically saved to stack and then restored when the interrupt routine has completed.

At the beginning of a program, the initial EMB and ERB flag values for each vectored interrupt must be set by using VENT instruction. The EMB and ERB can be set or reset by bit manipulation instructions (BITS, BITR) despite the current SMB setting.

☞ PROGRAMMING TIP — Initializing the EMB and ERB Flags

The following assembly instructions show how to initialize the EMB and ERB flag settings:

```
        ORG     0000H           ;  ROM address assignment
        VENT0   1,0,RESET       ;  EMB ← 1, ERB ← 0, branch RESET
        VENT1   0,1,INTB        ;  EMB ← 0, ERB ← 1, branch INTB
        VENT2   0,1,INT0        ;  EMB ← 0, ERB ← 1, branch INT0
        VENT3   0,1,INT1 (note) ;  EMB ← 0, ERB ← 1, branch INT1
        NOP
        NOP
        VENT5   0,1,INTT0       ;  EMB ← 0, ERB ← 1, branch INTT0
        VENT6   0,1,INTT1       ;  EMB ← 0, ERB ← 1, branch INTT1
          •
          •
          •
RESET   BITR    EMB
```

NOTE: S3C7534/C7538 does not use the INT1 interrupt.

ENABLE MEMORY BANK SETTINGS

EMB = "1"

When the enable memory bank flag EMB is set to logic one, you can address the data memory bank specified by the select memory bank (SMB) value (0, 1, 2 or 15) using 1-, 4-, or 8-bit instructions. You can use both direct and indirect addressing modes. The addressable RAM areas when EMB = "1" are as follows:

- If SMB = 0, 000H–0FFH
- If SMB = 1, 100H–1FFH
- If SMB = 2, 200H–2FFH
- If SMB = 15, F80H–FFFH

EMB = "0"

When the enable memory bank flag EMB is set to logic zero, the addressable area is defined independently of the SMB value, and is restricted to specific locations depending on whether a direct or indirect address mode is used.

If EMB = "0", the addressable area is restricted to locations 000H–07FH in bank 0 and to locations F80H–FFFH in bank 15 for direct addressing. For indirect addressing, only locations 000H–0FFH in bank 0 are addressable, regardless of SMB value.

To address the peripheral hardware register (bank 15) using indirect addressing, the EMB flag must first be set to "1" and the SMB value to "15". When a RESET occurs, the EMB flag is set to the value contained in bit 7 of ROM address 0000H.

EMB-Independent Addressing

At any time, several areas of the data memory can be addressed independently of the current status of the EMB flag. These exceptions are described in Table 3–1.

Table 3–1. RAM Addressing Not Affected by the EMB Value

Address	Addressing Method	Affected Hardware	Program Examples
000H–0FFH	4-bit indirect addressing using WX and WL register pairs; 8-bit indirect addressing using SP	Not applicable	LD A,@WX PUSH POP
FB0H–FBFH FF0H–FFFH	1-bit direct addressing	PSW, IEx, IRQx, I/O	BITS EMB BITR IE4
FC0H–FFFH	1-bit indirect addressing using the L register	I/O	BAND C,P3.@L

SELECT BANK REGISTER (SB)

The select bank register (SB) is used to assign the memory bank and register bank. The 8-bit SB register consists of the 4-bit select register bank register (SRB) and the 4-bit select memory bank register (SMB), as shown in Figure 3–2.

During interrupts and subroutine calls, SB register contents can be saved to stack in 8-bit units by the PUSH SB instruction. You later restore the value to the SB using the POP SB instruction.

	SMB (F83H)				SRB (F82H)			
SB REGISTER	SMB 3	SMB 2	SMB 1	SMB 0	0	0	SRB 1	SRB 0

Figure 3–2. SMB and SRB Values in the SB Register

Select Register Bank (SRB) Instruction

The select register bank (SRB) value specifies which register bank is to be used as a working register bank. The SRB value is set by the 'SRB n' instruction, where n = 0, 1, 2, 3.

One of the four register banks is selected by the combination of ERB flag status and the SRB value that is set using the 'SRB n' instruction. The current SRB value is retained until another register is requested by program software. PUSH SB and POP SB instructions are used to save and restore the contents of SRB during interrupts and subroutine calls. RESET clears the 4-bit SRB value to logic zero.

Select Memory Bank (SMB) Instruction

To select one of the four available data memory banks, you must execute an SMB n instruction specifying the number of the memory bank you want (0, 1, 2 or 15). For example, the instruction 'SMB 1' selects bank 1 and 'SMB 15' selects bank 15. (And remember to enable the selected memory bank by making the appropriate EMB flag setting.

The upper four bits of the 12-bit data memory address are stored in the SMB register. If the SMB value is not specified by software (or if a RESET does not occur) the current value is retained. RESET clears the 4-bit SMB value to logic zero.

The PUSH SB and POP SB instructions save and restore the contents of the SMB register to and from the stack area during interrupts and subroutine calls.

DIRECT AND INDIRECT ADDRESSING

1-bit, 4-bit, and 8-bit data stored in data memory locations can be addressed directly using a specific register or bit address as the instruction operand.

Indirect addressing specifies a memory location that contains the required direct address. The S3C7 instruction set supports 1-bit, 4-bit, and 8-bit indirect addressing. For 8-bit indirect addressing, an even-numbered RAM address must always be used as the instruction operand.

1-BIT ADDRESSING

Table 3–2. 1-Bit Direct and Indirect RAM Addressing

Operand Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA.b	Direct: bit is indicated by the RAM address (DA), memory bank selection, and specified bit number (b).	0	000H–07FH	Bank 0	–
			F80H–FFFH	Bank 15	All 1-bit addressable peripherals (SMB = 15)
		1	000H–FFFH	SMB = 0, 1, 2, 15	
mema.b	Direct: bit is indicated by addressable area (mema) and bit number (b).	x	FB0H–FBFH FF0H–FFFH	Bank 15	IS0, IS1, EMB, ERB, IEx, IRQx, Pn.n
memb.@L	Indirect: lower two bits of register L as indicated by the upper 10 bits of RAM area (memb) and the upper two bits of register L.	x	FC0H–FFFH	Bank 15	Pn.n
@H + DA.b	Indirect: bit indicated by the lower four bits of the address (DA), memory bank selection, and the H register identifier.	0	000H–0FFH	Bank 0	All 1-bit addressable peripherals (SMB = 15)
		1	000H–FFFH	SMB = 0, 1, 2, 15	

NOTE: x = not applicable.

☞ PROGRAMMING TIP — 1-Bit Addressing Modes

1-Bit Direct Addressing

1. If EMB = "0":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0
        BITS    AFLAG       ; 34H.3 ← 1
        BITS    BFLAG       ; F85H.3 (BMOD.3) ← 1
        BTST    CFLAG       ; If FBAH.0 (IRQW) = 1, skip
        BITS    BFLAG       ; Else if, FBAH.0 (IRQW) = 0, F85H.3 (BMOD.3) ← 1
        BITS    P3.0        ; FF3H.0 (P3.0) ← 1
```

2. If EMB = "1":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0
        BITS    AFLAG       ; 34H.3 ← 1
        BITS    BFLAG       ; 85H.3 ← 1
        BTST    CFLAG       ; If 0BAH.0 = 1, skip
        BITS    BFLAG       ; Else if 0BAH.0 = 0, 085H.3 ← 1
        BITS    P3.0        ; FF3H.0 (P3.0) ← 1
```

1-Bit Indirect Addressing

1. If EMB = "0":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0
        LD      H,#0BH      ; H ← #0BH
        BTSTZ   @H+CFLAG    ; If 0BAH.0 = 1, 0BAH.0 ← 0 and skip
        BITS    CFLAG       ; Else if 0BAH.0 = 0, FBAH.0 (IRQW) ← 1
```

2. If EMB = "1":

```
AFLAG   EQU     34H.3
BFLAG   EQU     85H.3
CFLAG   EQU     0BAH.0
        SMB     0
        LD      H,#0BH      ; H ← #0BH
        BTSTZ   @H+CFLAG    ; If 0BAH.0 = 1, 0BAH.0 ← 0 and skip
        BITS    CFLAG       ; Else if 0BAH.0 = 0, 0BAH.0 ← 1
```

4-BIT ADDRESSING

Table 3–3. 4-Bit Direct and Indirect RAM Addressing

Operand Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA	Direct: 4-bit address indicated by the RAM address (DA) and the memory bank selection	0	000H–07FH	Bank 0	–
			F80H–FFFH	Bank 15	All 4-bit addressable peripherals (SMB = 15)
		1	000H–FFFH	SMB = 0, 1, 2 15	
@HL	Indirect: 4-bit address indicated by the memory bank selection and register HL	0	000H–0FFH	Bank 0	–
		1	000H–FFFH	SMB = 0, 1, 2 15	All 4-bit addressable peripherals (SMB = 15)
@WX	Indirect: 4-bit address indicated by register WX	x	000H–0FFH	Bank 0	–
@WL	Indirect: 4-bit address indicated by register WL	x	000H–0FFH	Bank 0	

NOTE: x = not applicable.

PROGRAMMING TIP — 4-Bit Addressing Modes

4-Bit Direct Addressing

1. If EMB = "0":

```
ADATA   EQU   46H
BDATA   EQU   8EH
        SMB   15          ;  Non-essential instruction, since EMB = "0"
        LD    A,P3        ;  A ← (P3)
        SMB   0           ;  Non-essential instruction, since EMB = "0"
        LD    ADATA,A     ;  (046H) ← A
        LD    BDATA,A     ;  (F8EH) ← A
```

2. If EMB = "1":

```
ADATA   EQU   46H
BDATA   EQU   8EH
        SMB   15
        LD    A,P3        ;  A ← (P3)
        SMB   0
        LD    ADATA,A     ;  (046H) ← A
        LD    BDATA,A     ;  (08EH) ← A
```

☞ PROGRAMMING TIP — 4-Bit Addressing Modes (Continued)

4-Bit Indirect Addressing (Example 1)

1. If EMB = "0", compare bank 0 locations 040H–046H with bank 0 locations 060H–066H:

```
ADATA     EQU     46H
BDATA     EQU     66H
          SMB     1              ;   Non-essential instruction, since EMB = "0"
          LD      HL,#BDATA
          LD      WX,#ADATA
COMP      LD      A,@WL          ;   A ← bank 0 (040H–046H)
          CPSE    A,@HL          ;   If bank 0 (060H–066H) = A, skip
          SRET
          DECS    L
          JR      COMP
          RET
```

2. If EMB = "1", compare bank 0 locations 040H–046H to bank 1 locations 160H–166H:

```
ADATA     EQU     46H
BDATA     EQU     66H
          SMB     1
          LD      HL,#BDATA
          LD      WX,#ADATA
COMP      LD      A,@WL          ;   A ← bank 0 (040H–046H)
          CPSE    A,@HL          ;   If bank 1 (160H–166H) = A, skip
          SRET
          DECS    L
          JR      COMP
          RET
```

☞ PROGRAMMING TIP — 4-Bit Addressing Modes (Concluded)

4-Bit Indirect Addressing (Example 2)

1. If EMB = "0", exchange bank 0 locations 040H–046H with bank 0 locations 060H–066H:

```
ADATA       EQU     46H
BDATA       EQU     66H
        SMB     1       ;               Non-essential instruction, since EMB = "0"
        LD      HL,#BDATA
        LD      WX,#ADATA
TRANS       LD      A,@WL           ;   A ← bank 0 (040H–046MH)
        XCHD    A,@HL   ;               Bank 0 (060H–066H) ← A
        JR      TRANS
```

2. If EMB = "1", exchange bank 0 locations 040H–046H to bank 1 locations 160H–166H:

```
ADATA       EQU     46H
BDATA       EQU     66H
            SMB     1
            LD      HL,#BDATA
            LD      WX,#ADATA
TRANS       LD      A,@WL           ;   A ← bank 0 (040H–046H)
            XCHD    A,@HL           ;   Bank 1 (160H–166H) ← A
            JR      TRANS
```

8-BIT ADDRESSING

Table 3–4. 8-Bit Direct and Indirect RAM Addressing

Instruction Notation	Addressing Mode Description	EMB Flag Setting	Addressable Area	Memory Bank	Hardware I/O Mapping
DA	Direct: 8-bit address indicated by the RAM address (*DA = even number*) and memory bank selection	0	000H–07FH	Bank 0	–
			F80H–FFFH	Bank 15	All 8-bit addressable peripherals (SMB = 15)
		1	000H–FFFH	SMB = 0, 1, 2, 15	
@HL	Indirect: the 8-bit address indicated by the memory bank selection and register HL; (the 4-bit L register value must be an even number)	0	000H–0FFH	Bank 0	–
		1	000H–FFFH	SMB = 0, 1, 2, 15	All 8-bit addressable peripherals (SMB = 15)

☞ PROGRAMMING TIP — 8-Bit Addressing Modes

8-Bit Direct Addressing

1. If EMB = "0":

```
ADATA   EQU   46H
BDATA   EQU   8EH
        SMB   15          ;  Non-essential instruction, since EMB = "0"
        LD    EA,P4       ;  E ← (P5), A ← (P4)
        SMB   0
        LD    ADATA,EA    ;  (046H) ← A, (047H) ← E
        LD    BDATA,EA    ;  (F8EH) ← A, (F8FH) ← E
```

2. If EMB = "1":

```
ADATA   EQU   46H
BDATA   EQU   8EH
        SMB   15
        LD    EA,P4       ;  E ← (P5), A ← (P4)
        SMB   0
        LD    ADATA,EA    ;  (046H) ← A, (047H) ← E
        LD    BDATA,EA    ;  (08EH) ← A, (08FH) ← E
```

8-Bit Indirect Addressing

1. If EMB = "0":

```
ADATA   EQU   146H
        SMB   1           ;  Non-essential instruction, since EMB = "0"
        LD    HL,#ADATA
        LD    EA,@HL      ;  A ← (046H), E ← (047H)
```

2. If EMB = "1":

```
ADATA   EQU   146H
        SMB   1
        LD    HL,#ADATA
        LD    EA,@HL      ;  A ← (146H), E ← (147H)
```

4 MEMORY MAP

OVERVIEW

To support program control of peripheral hardware, I/O addresses for peripherals are memory-mapped to bank 15 of the RAM. Memory mapping lets you use a mnemonic as the operand of an instruction in place of the specific memory location.

Access to bank 15 is controlled by the select memory bank (SMB) instruction and by the enable memory bank flag (EMB) setting. If the EMB flag is “0”, bank 15 can be addressed using direct addressing, regardless of the current SMB value. 1-bit direct and indirect addressing can be used for specific locations in bank 15, regardless of the current EMB value.

I/O MAP FOR HARDWARE REGISTERS

Table 4–1 contains detailed information about I/O mapping for peripheral hardware in bank 15 (register locations F80H–FFFH). Use the I/O map as a quick-reference source when writing application programs. The I/O map gives you the following information:

- Register address
- Register name (mnemonic for program addressing)
- Bit values (both addressable and non-manipulable)
- Read-only, write-only, or read and write addressability
- 1-bit, 4-bit, or 8-bit data manipulation characteristics

Table 4–1. I/O Map for Memory Bank 15

Memory Bank 15							Addressing Mode		
Address	**Register**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**R/W**	**1-Bit**	**4-Bit**	**8-Bit**
F80H	SP	.3	.2	.1	"0"	R/W	No	No	Yes
F81H		.7	.6	.5	.4				
Locations F82H–F84H are not mapped.									
F85H	BMOD	.3	.2	.1	.0	W	.3	Yes	No
F86H	BCNT					R	No	No	Yes
F87H									
F88H	WMOD	"0"	.2	.1	"0" (1)	W	No	No	Yes
F89H		.7	"0"	.5	.4				
Locations F8AH–F8FH are not mapped.									
F90H	TMOD0	.3	.2	"0"	"0"	W	.3	No	Yes
F91H		"0"	.6	.5	.4				
F92H		TOE1	TOE0	BOE	"0"	R/W	Yes	Yes	No
F93H		"0"	TOL1	TOL0	"0"				
F94H	TCNT0					R	No	No	Yes
F95H									
F96H	TREF0					W	No	No	Yes
F97H									
F98H	WDMOD	.3	.2	.1	.0	W	No	No	Yes
F99H		.7	.6	.5	.4				
F9AH	WDFLAG	WDTCF	"0"	"0"	"0"	W	Yes	Yes	No
Locations F9BH–F9FH are not mapped.									
FA0H	TMOD1	.3	.2	"0"	"0"	W	.3	No	Yes
FA1H		"0"	.6	.5	.4				
Locations FA2H–FA3H are not mapped.									
FA4H	TCNT1					R	No	No	Yes
FA5H									
Locations FA6H–FA7H are not mapped.									
FA8H	TREF1					W	No	No	Yes
FA9H									
Locations FAAH–FAFH are not mapped.									
FB0H	PSW	IS1	IS0	EMB	ERB	R/W	Yes	Yes	Yes
FB1H		C (2)	SC2	SC1	SC0	R	No	No	
FB2H	IPR	IME	.2	.1	.0	W	IME	Yes	No

Table 4–1. I/O Map for Memory Bank 15 (Continued)

FB3H	PCON	.3	.2	.1	.0	W	.3, .2	Yes	No
FB4H	IMOD0	"0"	"0"	.1	.0	W	No	Yes	No
FB5H	IMOD1 (3)	"0"	"0"	"0"	.0				
FB6H	IMOD2	"0"	"0"	.1	.0				
Locations FB7H is not mapped.									
FB8H		IE4 (3)	IRQ4 (3)	IEB	IRQB	R/W	Yes	Yes	No
Locations FB9H is not mapped.									
FBAH		"0"	"0"	IEW	IRQW	R/W	Yes	Yes	No
FBBH		"0"	"0"	IET1	IRQT1				
FBCH		"0"	"0"	IET0	IRQT0				
Locations FBDH is not mapped.									
FBEH		IE1 (3)	IRQ1 (3)	IE0	IRQ0	R/W	Yes	No	No
FBFH		"0"	"0"	IE2	IRQ2				
FC0H	BSC0					R/W	Yes	No	Yes
FC1H	BSC1								
FC2H	BSC2								
FC3H	BSC3								
Locations FC4H–FCFH are not mapped.									
FD0H	CLMOD	.3	"0"	.1	.0	W	No	Yes	No
Locations FD1H is not mapped.									
FD2H	DTMR	"0"	.2	.1	.0	W	No	No	Yes
FD3H		.7	.6	.5	.4				
Locations FD4H–FD9H are not mapped.									
FDAH	PNE1	PNE4.3	PNE4.2	PNE4.1	PNE4.0	W	No	No	Yes
FDBH		PNE5.3	PNE5.2	PNE5.1	PNE5.0				
FDCH	PUMOD1	PUR1.3 (4)	PUR1.2 (4)	PUR1.1 (4)	PUR1.0	W	No	No	Yes
FDDH		PUR5	PUR4	PUR3	PUR2				
FDEH	PUMOD2	PUR9 (4)	PUR8 (4)	PUR7	PUR6	W	No	Yes	No
Locations FDFH–FE7H are not mapped.									
FE8H	PMG1	PM2.3	PM2.2	PM2.1	PM2.0	W	No	No	Yes
FE9H		PM3.3 (4)	PM3.2 (4)	PM3.1	PM3.0				
FEAH	PMG2	PM4.3	PM4.2	PM4.1	PM4.0				
FEBH		PM5.3	PM5.2	PM5.1	PM5.0				
FECH	PMG3	PM6.3	PM6.2	PM6.1	PM6.0				
FEDH		PM7.3	PM7.2	PM7.1	PM7.0				

Table 4–1. I/O Map for Memory Bank 15 (Continued)

FEEH	PMG4 [4]	PM8.3 [4]	PM8.2 [4]	PM8.1 [4]	PM8.0 [4]	W	No	No	Yes
FEFH		“0”	PM9.2 [4]	PM9.1 [4]	PM9.0 [4]				
Locations FF0H is not mapped.									
FF1H	Port 1	.3	.2	.1	.0	R	Yes	Yes	No
FF2H	Port 2	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FF3H	Port 3	.3 / .7	.2 / .6	.1 / .5	.0 / .4				
FF4H	Port 4	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FF5H	Port 5	.3 / .7	.2 / .6	.1 / .5	.0 / .4				
FF6H	Port 6	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FF7H	Port 7	.3 / .7	.2 / .6	.1 / .5	.0 / .4				
FF8H	Port 8	.3	.2	.1	.0	R/W	Yes	Yes	Yes
FF9H	Port 9	“0”	.2 / .6	.1 / .5	.0 / .4				

NOTES

1. Bit 0 in the WMOD register must be set to logic “0”
2. The carry flag can be read or written by specific bit manipulation instructions only.
3. S3C7534/C7538 does not use the INT1 and INT4 interrupts.
4. S3C7534/C7538 does not use the PMG4, P1.1–P1.3, P3.2–P3.3, P8, P9.

REGISTER DESCRIPTIONS

In this section, register descriptions are presented in a consistent format to familiarize you with the memory-mapped I/O locations in bank 15 of the RAM. Figure 4–1 describes features of the register description format. Register descriptions are arranged in alphabetical order. Programmers can use this section as a quick-reference source when writing application programs.

Counter registers, buffer registers, and reference registers, as well as the stack pointer and port I/O latches, are not included in these descriptions. More detailed information about how these registers are used is included in Part II of this manual, “Hardware Descriptions,” in the context of the corresponding peripheral hardware module descriptions.

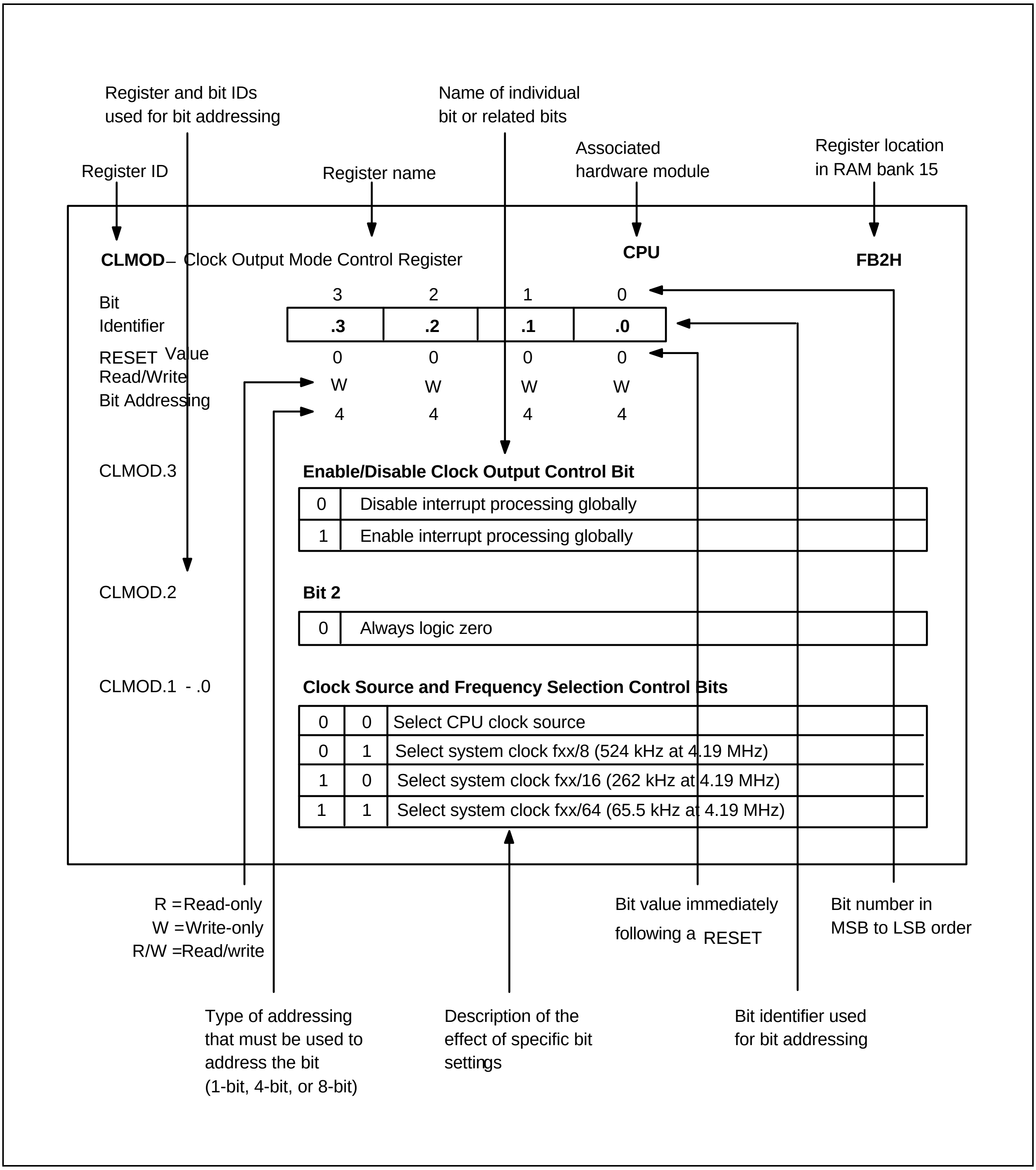


Figure 4–1. Register Description Format

BMOD — Basic Timer Mode Register BT F85H

Bit	3	2	1	0
Identifier	**.3**	**.2**	**.1**	**.0**
RESET **Value**	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	4	4	4

BMOD.3 **Basic Timer Restart Bit**

1	Restart basic timer, then clear IRQB flag, BCNT and BMOD.3 to logic zero

BMOD.2 – .0 **Input Clock Frequency and Signal Stabilization Interval Control Bits**

0	0	0	Input clock frequency: Signal stabilization interval:	fx / 2^{12} (1.02 kHz) 2^{20} / fx (250 ms)
0	1	1	Input clock frequency: Signal stabilization interval:	fx / 2^{9} (8.18 kHz) 2^{17} / fx (31.3 ms)
1	0	1	Input clock frequency: Signal stabilization interval:	fx / 2^{7} (32.7 kHz) 2^{15} / fx (7.82 ms)
1	1	1	Input clock frequency: Signal stabilization interval:	fx / 2^{5} (131 kHz) 2^{13} / fx (1.95 ms)

NOTES

1. Signal stabilization interval is the time required to stabilize clock signal oscillation after stop mode is terminated by an interrupt. The stabilization interval can also be interpreted as “Interrupt Interval Time”.
2. When a RESET occurs, the oscillation stabilization time is 31.3 ms (2^{17}/fx) at 4.19 MHz.
3. ‘fx’ is the system clock rate, given a clock frequency of 4.19 MHz.

CLMOD — Clock Output Mode Register CPU FD0H

Bit	3	2	1	0
Identifier	**.3**	**"0"**	**.1**	**.0**
RESET **Value**	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

CLMOD.3 **Enable/Disable Clock Output Control Bit**

0	Disable clock output
1	Enable clock output

CLMOD.2 **Bit 2**

0	Always logic zero

CLMOD.1 – .0 **Clock Source and Frequency Selection Control Bits**

0	0	Select CPU clock source fx/4, fx/8 or fx/64 (1.05 MHz, 524 kHz or 65.5 kHz)
0	1	Select system clock fx/8 (524 kHz)
1	0	Select system clock fx/16 (262 kHz)
1	1	Select system clock fx/64 (65.5kHz)

NOTE: 'fx' is the system clock, given a clock frequency of 4.19 MHz.

DTMR — DTMF Mode Register

DTMF **FD3H, FD2H**

Bit	3	2	1	0	3	2	1	0
Identifier	**.7**	**.6**	**.5**	**.4**	**“0”**	**.2**	**.1**	**.0**
RESET **Value**	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

DTMR.7 – .4 **DTMR Bit Values For Keyboard Inputs**

0	0	0	0	Function key D
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	0
1	0	1	1	*
1	1	0	0	#
1	1	0	1	Function key A
1	1	1	0	Function key B
1	1	1	1	Function key C

DTMR.3 **Bit 3**

0	Always logic zero

DTMR.2 – .1 **Tone Selection Bits**

0	0	Dual-tone enable
1	0	Dual-tone enable (alternate setting)
0	1	Single-column tone enable
1	1	Single-low tone enable

DTMR.0 **DTMF Operation Enable/Disable Bit**

0	Disable DTMF operation
1	Enable DTMF operation

IMOD0 — External Interrupt 0 (INT0) Mode Register CPU FB4H

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**.1**	**.0**
RESET **Value**	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

IMOD0.3 – .2 **Bits 3–2**

0	Always logic zero

IMOD0.1 – .0 **External Interrupt Mode Control Bits**

0	0	Interrupt requests are triggered by a rising signal edge
0	1	Interrupt requests are triggered by a falling signal edge
1	0	Interrupt requests are triggered by both rising and falling signal edges
1	1	Interrupt request flag (IRQx) cannot be set to logic one

IMOD1— External Interrupt 1 (INT1) Mode Register (note) CPU FB5H

Bit	3	2	1	0
Identifier	**“0”**	**“0”**	**“0”**	**.0**
RESET **Value**	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

IMOD1.3 – .1 **Bits 3–1**

0	Always logic zero

IMOD1.0 **External Interrupt 1 Edge Detection Control Bit**

0	Rising edge detection
1	Falling edge detection

NOTE: S3C7534/C7538 does not use the IMOD1.

IMOD2 — External Interrupt 2 (INT2) Mode Register CPU FB6H

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**.1**	**.0**
RESET **Value**	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

IMOD2.3 – .2 **Bits 3–2**

0	Always logic zero

IMOD2.1 – .0 **External Interrupt 2 Edge Detection Selection Bit**

0	0	Interrupt request at INT2 pin triggered by rising edge (S3C7524/7528 only)
0	1	Interrupt request at KS4–KS7 triggered by falling edge
1	0	Interrupt request at KS2–KS7 triggered by falling edge
1	1	Interrupt request at KS0–KS7 triggered by falling edge

IE0, 1, IRQ0, 1 — INT0, 1 Interrupt Enable/Request Flags CPU FBEH

Bit	3	2	1	0
Identifier	IE1 (note)	IRQ1 (note)	IE0	IRQ0
RESET Value	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

IE1 (note) **INT1 Interrupt Enable Flag**

0	Disable interrupt requests at the INT1 pin
1	Enable interrupt requests at the INT1 pin

IRQ1 (note) **INT1 Interrupt Request Flag**

–	Generate INT1 interrupt (This bit is set and cleared by hardware when rising or falling edge detected at INT1 pin.)

IE0 **INT0 Interrupt Enable Flag**

0	Disable interrupt requests at the INT0 pin
1	Enable interrupt requests at the INT0 pin

IRQ0 **INT0 Interrupt Request Flag**

–	Generate INT0 interrupt (This bit is set and cleared automatically by hardware when rising or falling edge detected at INT0 pin.)

NOTE: S3C7534/C7538 does not use IE1, IRQ1, and INT1 pin.

IE2, IRQ2 — INT2 Interrupt Enable/Request Flags

CPU FBFH

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**IE2**	**IRQ2**
RESET **Value**	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3 – .2 **Bits 3–2**

0	Always logic zero

IE2 **INT2 Interrupt Enable Flag**

0	Disable INT2 interrupt requests at the INT2 pin (note) or KS0–KS7 pins
1	Enable INT2 interrupt requests at the INT2 pin (note) or KS0–KS7 pins

IRQ2 **INT2 Interrupt Request Flag**

–	Generate INT2 quasi-interrupt (This bit is set and is not cleared automatically by hardware when a rising edge is detected at INT2 pin (note) or when a falling edge is detected at one of the KS0–KS7 pins. Since INT2 is a quasi-interrupt, IRQ2 flag must be cleared by software.)

NOTE: S3C7534/C7538 does not use INT2 pin.

IE4, IRQ4 — INT4 Interrupt Enable/Request Flags (note) CPU FB8H
IEB, IRQB — INTB Interrupt Enable/Request Flags CPU FB8H

Bit	3	2	1	0
Identifier	**IE4** (note)	**IRQ4** (note)	**IEB**	**IRQB**
RESET **Value**	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

IE4 (note) **INT4 Interrupt Enable Flag**

0	Disable interrupt requests at the INT4 pin
1	Enable interrupt requests at the INT4 pin

IRQ4 (note) **INT4 Interrupt Request Flag**

–	Generate INT4 interrupt (This bit is set and cleared automatically by hardware when rising and falling signal edge detected at INT4 pin.)

IEB **INTB Interrupt Enable Flag**

0	Disable INTB interrupt requests
1	Enable INTB interrupt requests

IRQB **INTB Interrupt Request Flag**

–	Generate INTB interrupt (This bit is set and cleared automatically by hardware when reference interval signal received from basic timer.)

NOTE: S3C7534/C7538 does not use IE4, IRQ4, and INT4 pin.

IET0, IRQT0 — INTT0 Interrupt Enable/Request Flags CPU FBCH

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**IET0**	**IRQT0**
RESET **Value**	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3 – .2 **Bits 3–2**

0	Always logic zero

IET0 **INTT0 Interrupt Enable Flag**

0	Disable INTT0 interrupt requests
1	Enable INTT0 interrupt requests

IRQT0 **INTT0 Interrupt Request Flag**

–	Generate INTT0 interrupt (This bit is set and cleared automatically by hardware when contents of TCNT0 and TREF0 registers match.)

IET1, IRQT1 — INTT1 Interrupt Enable/Request Flags CPU FBBH

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**IET1**	**IRQT1**
RESET **Value**	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.2 – .3 **Bits 2–3**

0	Always logic 0

IET1 **INTT1 Interrupt Enable Flag**

0	Disable INTT1 interrupt requests
1	Enable INTT1 interrupt requests

IRQT1 **INTT1 Interrupt Request Flag**

–	Generate INTT1 interrupt (This bit is set and cleared automatically by hardware when contents of TCNT1 and TREF1 registers match.)

IEW, IRQW — Intw Interrupt Enable/Request Flags CPU FBAH

Bit	3	2	1	0
Identifier	**"0"**	**"0"**	**IEW**	**IRQW**
RESET **Value**	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W
Bit Addressing	1/4	1/4	1/4	1/4

.3 – .2 **Bits 3–2**

0	Always logic zero

IEW **INTW Interrupt Enable Flag**

0	Disable INTW interrupt requests
1	Enable INTW interrupt requests

IRQW **INTW Interrupt Request Flag**

–	Generate INTW interrupt (This bit is set when the timer interval is set to 0.5 seconds or 3.19 milliseconds at the watch timer frequency of 32.768 kHz.)

NOTE: Since INTW is a quasi-interrupt, the IRQW flag must be cleared by software.

IPR — Interrupt Priority Register CPU FB2H

Bit	3	2	1	0
Identifier	**IME**	**.2**	**.1**	**.0**
RESET **Value**	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	4	4	4

IME **Interrupt Master Enable Bit (MSB)**

0	Disable all interrupt processing
1	Enable processing of all interrupt service requests

IPR.2 – .0 **Interrupt Priority Assignment Bits**

0	0	0	Normal interrupt processing according to default priority settings
0	0	1	Process INTB and INT4 (note) interrupts at highest priority
0	1	0	Process INT0 interrupts at highest priority
0	1	1	Process INT1 (note) interrupts at highest priority
1	0	1	Process INTT0 interrupts at highest priority
1	1	0	Process INTT1 interrupts at highest priority

NOTES

1. S3C7534/C7538 does not use the INT1 and INT4 interrupts.
2. During normal interrupt processing, interrupts are processed in the order in which they occur. If two or more interrupts occur simultaneously, the processing order is determined by the default interrupt priority settings shown below. Using the IPR settings, you can select specific interrupts for high-priority processing in the event of contention. When the high-priority (IPR) interrupt has been processed, waiting interrupts are handled according to their default priorities. The default priorities are as follows ('1' is highest priority; '5' is lowest priority):

INTB, INT4	1
INT0	2
INT1	3
INTT0	4
INTT1	5

PCON — Power Control Register   CPU   FB3H

	3	2	1	0
Bit	3	2	1	0
Identifier	**.3**	**.2**	**.1**	**.0**
RESET **Value**	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	1/4	4	4

PCON.3 – .2 **CPU Operating Mode Control Bits**

0	0	Enable normal CPU operating mode
0	1	Initiate idle power-down mode
1	0	Initiate stop power-down mode

PCON.1 – .0 **CPU Clock Frequency Selection Bits**

0	0	Select fx/64
1	0	Select fx/8
1	1	Select fx/4

NOTE: ‘fx’ is the system clock.

PSW — Program Status Word

CPU FB1H, FB0H

Bit	7	6	5	4	3	2	1	0
Identifier	**C**	**SC2**	**SC1**	**SC0**	**IS1**	**IS0**	**EMB**	**ERB**
RESET **Value**	(1)	0	0	0	0	0	0	0
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W
Bit Addressing	(2)	8	8	8	1/4	1/4	1	1

C **Carry Flag**

0	No overflow or borrow condition exists
1	An overflow or borrow condition does exist

SC2 – SC0 **Skip Condition Flags**

0	No skip condition exists; no direct manipulation of these bits is allowed
1	A skip condition exists; no direct manipulation of these bits is allowed

IS1, IS0 **Interrupt Status Flags**

0	0	Service all interrupt requests
0	1	Service only the high-priority interrupt(s) as determined in the interrupt priority register (IPR)
1	0	Do not service any more interrupt requests
1	1	Undefined

EMB **Enable Data Memory Bank Flag**

0	Restrict program access to data memory to bank 15 (F80H–FFFH) and to the locations 000H–07FH in the bank 0 only
1	Enable full access to data memory banks 0, 1, and 15

ERB **Enable Register Bank Flag**

0	Select register bank 0 as working register area
1	Select register banks 0, 1, 2 or 3 as working register area in accordance with the select register bank (SRB) instruction operand

NOTES

1. The value of the carry flag after a RESET occurs during normal operation is undefined. If a RESET occurs during power-down mode (IDLE or STOP), the current value of the carry flag is retained.
2. The carry flag can only be addressed by a specific set of 1-bit manipulation instructions. See Section 2 for detailed information.

PMG1 — Port I/O Mode Flags (Group 1: Ports 2, 3) I/O FE9H, FE8H

Bit	7	6	5	4	3	2	1	0
Identifier	**PM3.3** (note)	**PM3.2** (note)	**PM3.1**	**PM3.0**	**PM2.3**	**PM2.2**	**PM2.1**	**PM2.0**
RESET **Value**	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PM3.3 (note) **P3.3 I/O Mode Selection Flag**

0	Set P3.3 to input mode
1	Set P3.3 to output mode

PM3.2 (note) **P3.2 I/O Mode Selection Flag**

0	Set P3.2 to input mode
1	Set P3.2 to input mode

PM3.1 **P3.1 I/O Mode Selection Flag**

0	Set P3.1 to input mode
1	Set P3.1 to output mode

PM3.0 **P3.0 I/O Mode Selection Flag**

0	Set P3.0 to input mode
1	Set P3.0 to output mode

PM2.3 **P2.3 I/O Mode Selection Flag**

0	Set P2.3 to input mode
1	Set P2.3 to output mode

PM2.2 **P2.2 I/O Mode Selection Flag**

0	Set P2.2 to input mode
1	Set P2.2 to output mode

PM2.1 **P0.1 I/O Mode Selection Flag**

0	Set P2.1 to input mode
1	Set P2.1 to output mode

PM2.0 **P2.0 I/O Mode Selection Flag**

0	Set P2.0 to input mode
1	Set P2.0 to output mode

NOTE: S3C7534/C7538 does not use the PM3.2–PM3.3.

PMG2 — Port I/O Mode Flags (Group 2: Ports 4, 5) I/O FEBH, FEAH

Bit	7	6	5	4	3	2	1	0
Identifier	**PM5.3**	**PM5.2**	**PM5.1**	**PM5.0**	**PM4.3**	**PM4.2**	**PM4.1**	**PM4.0**
RESET **Value**	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PM5.3 **P5.3 I/O Mode Selection Flag**

0	Set P5.3 to input mode
1	Set P5.3 to output mode

PM5.2 **P5.2 I/O Mode Selection Flag**

0	Set P5.2 to input mode
1	Set P5.2 to output mode

PM5.1 **P5.1 I/O Mode Selection Flag**

0	Set P5.1 to input mode
1	Set P5.1 to output mode

PM5.0 **P5.0 I/O Mode Selection Flag**

0	Set P5.0 to input mode
1	Set P5.0 to output mode

PM4.3 **P4.3 I/O Mode Selection Flag**

0	Set P4.3 to input mode
1	Set P4.3 to output mode

PM4.2 **P4.2 I/O Mode Selection Flag**

0	Set P4.2 to input mode
1	Set P4.2 to output mode

PM4.1 **P4.1 I/O Mode Selection Flag**

0	Set P4.1 to input mode
1	Set P4.1 to output mode

PM4.0 **P4.0 I/O Mode Selection Flag**

0	Set P4.0 to input mode
1	Set P4.0 to output mode

PMG3 — Port I/O Mode Flags (Group 3: Ports 6, 7) I/O FEDH, FECH

Bit	7	6	5	4	3	2	1	0
Identifier	**PM7.3**	**PM7.2**	**PM7.1**	**PM7.0**	**PM6.3**	**PM6.2**	**PM6.1**	**PM6.0**
RESET **Value**	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PM7.3 **P7.3 I/O Mode Selection Flag**

0	Set P7.3 to input mode
1	Set P7.3 to output mode

PM7.2 **P7.2 I/O Mode Selection Flag**

0	Set P7.2 to input mode
1	Set P7.2 to output mode

PM7.1 **P7.1 I/O Mode Selection Flag**

0	Set P7.1 to input mode
1	Set P7.1 to output mode

PM7.0 **P7.0 I/O Mode Selection Flag**

0	Set P7.0 to input mode
1	Set P7.0 to output mode

PM6.3 **P6.3 I/O Mode Selection Flag**

0	Set P6.3 to input mode
1	Set P6.3 to output mode

PM6.2 **P6.2 I/O Mode Selection Flag**

0	Set P6.2 to input mode
1	Set P6.2 to output mode

PM6.1 **P6.1 I/O Mode Selection Flag**

0	Set P6.1 to input mode
1	Set P6.1 to output mode

PM6.0 **P6.0 I/O Mode Selection Flag**

0	Set P6.0 to input mode
1	Set P6.0 to output mode

PMG4 — Port I/O Mode Flags (Group 3: Ports 8, 9) (note) I/O FEFH, FEEH

Bit	7	6	5	4	3	2	1	0
Identifier	**"0"**	**PM9.2**	**PM9.1**	**PM9.0**	**PM8.3**	**PM8.2**	**PM8.1**	**PM8.0**
RESET **Value**	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

.7 **Bit 7**

0	Always logic zero

PM9.2 **P9.2 I/O Mode Selection Flag**

0	Set P9.2 to input mode
1	Set P9.2 to output mode

PM9.1 **P9.1 I/O Mode Selection Flag**

0	Set P9.1 to input mode
1	Set P9.1 to output mode

PM9.0 **P9.0 I/O Mode Selection Flag**

0	Set P 9.0 to input mode
1	Set P 9.0 to output mode

PM8.3 **P8.3 I/O Mode Selection Flag**

0	Set P8.3 to input mode
1	Set P8.3 to output mode

PM8.2 **P8.2 I/O Mode Selection Flag**

0	Set P8.2 to input mode
1	Set P8.2 to output mode

PM8.1 **P8.1 I/O Mode Selection Flag**

0	Set P8.1 to input mode
1	Set P8.1 to output mode

PM8.0 **P8.0 I/O Mode Selection Flag**

0	Set P8.0 to input mode
1	Set P8.0 to output mode

NOTE: S3C7534/C7538 does not use the PMG4.

PNE 1 — Port Open-Drain Enable Register FDBH, FDAH

Bit	7	6	5	4	3	2	1	0
Identifier	PNE5.3	PNE5.2	PNE5.1	PNE5.0	PNE4.3	PNE4.2	PNE4.1	PNE4.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PNE5.3 **P5.3 Output mode Control Bit**

0	Push-pull output
1	N-channel open-drain output

PNE5.2 **P5.2 Output mode Control Bit**

0	Push-pull output
1	N-channel open-drain output

PNE5.1 **P5.1 Output mode Control Bit**

0	Push-pull output
1	N-channel open-drain output

PNE5.0 **P5.0 Output mode Control Bit**

0	Push-pull output
1	N-channel open-drain output

PNE4.3 **P4.3 Output mode Control Bit**

0	Push-pull output
1	N-channel open-drain output

PNE4.2 **P4.2 Output mode Control Bit**

0	Push-pull output
1	N-channel open-drain output

PNE4.1 **P4.1 Output mode Control Bit**

0	Push-pull output
1	N-channel open-drain output

PNE4.0 **P4.0 Output mode Control Bit**

0	Push-pull output
1	N-channel open-drain output

PUMOD1 — Pull-Up Resistor Mode Register 1 I/O FDDH, FDCH

Bit	7	6	5	4	3	2	1	0
Identifier	**PUR5**	**PUR4**	**PUR3**	**PUR2**	**PUR1.3** (note)	**PUR1.2** (note)	**PUR1.1** (note)	**PUR1.0**
RESET **Value**	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

PUR5 **Connect/Disconnect Port 5 Pull-Up Resistor Control Bit**

0	Disconnect port 5 pull-up resistor
1	Connect port 5 pull-up resistor

PUR4 **Connect/Disconnect Port 4 Pull-Up Resistor Control Bit**

0	Disconnect port 4 pull-up resistor
1	Connect port 4 pull-up resistor

PUR3 **Connect/Disconnect Port 3 Pull-Up Resistor Control Bit**

0	Disconnect port 3 pull-up resistor
1	Connect port 3 pull-up resistor

PUR2 **Connect/Disconnect Port 2 Pull-Up Resistor Control Bit**

0	Disconnect port 2 pull-up resistor
1	Connect port 2 pull-up resistor

PUR1.3 (note) **Connect/Disconnect P1.3 Pull-Up Resistor Control Bit**

0	Disconnect P1.3 pull-up resistor
1	Connect P1.3 pull-up resistor

PUR1.2 (note) **Connect/Disconnect P1.2 Pull-Up Resistor Control Bit**

0	Disconnect P1.2 pull-up resistor
1	Connect P1.2 pull-up resistor

PUR1.1 (note) **Connect/Disconnect P1.1 Pull-Up Resistor Control Bit**

0	Disconnect P1.1 pull-up resistor
1	Connect P1.1 pull-up resistor

PUR1.0 **Connect/Disconnect P1.0 Pull-Up Resistor Control Bit**

0	Disconnect P1.0 pull-up resistor
1	Connect P1.0 pull-up resistor

NOTE: S3C7534/C7538 does not use the PUR1.1–PUR1.3.

PUMOD2 — PULL-UP RESISTOR MODE REGISTER 2 I/O FDEH

Bit	3	2	1	0
Identifier	PUR9 (note)	PUR8 (note)	PUR7	PUR6
RESET Value	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	4	4	4	4

PUR9 (note) — **Connect/Disconnect Port 9 Pull-Up Resistor Control Bit**

0	Disconnect port 9 pull-up resistor
1	Connect port 9 pull-up resistor

PUR8 (note) — **Connect/Disconnect Port 8 Pull-Down Resistor Control Bit**

0	Disconnect port 8 pull-down resistor
1	Connect port 8 pull-down resistor

PUR7 — **Connect/Disconnect Port 7 Pull-Up Resistor Control Bit**

0	Disconnect port 7 pull-up resistor
1	Connect port 7 pull-up resistor

PUR6 — **Connect/Disconnect Port 6 Pull-Up Resistor Control Bit**

0	Disconnect port 6 pull-up resistor
1	Connect port 6 pull-up resistor

NOTE: S3C7534/C7538 does not use the PUR8 and PUR9.

TMOD0 — Timer/Counter 0 Mode Register   T/C0   F91H, F90H

Bit	3	2	1	0	3	2	1	0
Identifier	**"0"**	**.6**	**.5**	**.4**	**.3**	**.2**	**"0"**	**"0"**
RESET **Value**	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	1/8	8	8	8

TMOD0.7 **Bit 7**

0	Always logic zero

TMOD0.6 – .4 **Timer/Counter 0 Input Clock Selection Bits**

0	0	0	External clock input at TCL0 pin on rising edge
0	0	1	External clock input at TCL0 pin on falling edge
1	0	0	Internal system clock $fx/2^{10}$ (4.09 kHz)
1	0	1	Select clock: $fx/2^{6}$ (65.5 kHz)
1	1	0	Select clock: $fx/2^{4}$ (262 kHz)
1	1	1	Select clock: fx (4.19 MHz)

TMOD0.3 **Clear Counter and Resume Counting Control Bit**

1	Clears TCNT0 and IRQT0. TOL0 is remained and resume counting immediately (This bit is cleared automatically when counting starts.)

TMOD0.2 **Enable/Disable Timer/Counter 0 Bit**

0	Disable timer/counter 0; retain TCNT0 contents
1	Enable timer/counter 0

TMOD0.1 **Bit 1**

0	Always logic zero

TMOD0.0 **Bit 0**

0	Always logic zero

TMOD1 — Timer/Counter 1 Mode Register T/C1 FA1H, FA0H

Bit	3	2	1	0	3	2	1	0
Identifier	**"0"**	**.6**	**.5**	**.4**	**.3**	**.2**	**"0"**	**"0"**
RESET **Value**	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	1/8	8	8	8

TMOD1.7 **Bit 7**

0	Always logic zero

TMOD1.6 – .4 **Timer/Counter 0 Input Clock Selection Bits**

0	0	0	External clock input at TCL1 pin on rising edge
0	0	1	External clock input at TCL1 pin on falling edge
1	0	0	Internal system clock fx/2^{12} (1.02 kHz)
1	0	1	Select clock: fx/2^{10} (4.09 kHz)
1	1	0	Select clock: fx/2^{8} (16.4 kHz)
1	1	1	Select clock: fx/2^{6} (65.5 kHz)

TMOD1.3 **Clear Counter and Resume Counting Control Bit**

1	Clears TCNT1 and IRQT1. TOL1 is remained and resume counting immediately (This bit is cleared automatically when counting starts.)

TMOD1.2 **Enable/Disable Timer/Counter 0 Bit**

0	Disable timer/counter 1; retain TCNT1 contents
1	Enable timer/counter 1

TMOD1.1 **Bit 1**

0	Always logic zero

TMOD1.0 **Bit 0**

0	Always logic zero

TOE — Timer Output Enable Flag Register T/C F92H

Bit	3	2	1	0
Identifier	**TOE1**	**TOE0**	**BOE**	**“0”**
RESET **Value**	0	0	0	0
Read/Write	R/W	R/W	R/W	W
Bit Addressing	1/4	1/4	1/4	1/4

TOE1 **Timer/Counter 1 Output Enable Flag**

0	Disable timer/counter 1 output to the TCLO1 pin
1	Enable timer/counter 1 output to the TCLO1 pin

TOE0 **Timer/Counter 0 Output Enable Flag**

0	Disable timer/counter 0 output at the TCLO0 pin
1	Enable timer/counter 0 output at the TCLO0 pin

BOE **Basic Timer Output Enable Flag**

0	Disable basic timer output at the BTCO pin
1	Enable basic timer output at the BTCO pin

.0 **Bit 0**

0	Always logic zero

WDMOD — Watchdog Timer Mode Register F99H, F98H

Bit	7	6	5	4	3	2	1	0
Identifier	**.7**	**.6**	**.5**	**.4**	**.3**	**.2**	**.1**	**.0**
RESET **Value**	1	0	1	0	0	1	0	1
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

WDMOD **Watchdog Timer Enable/Disable Control**

5AH	Disable watchdog timer function
Any other value	Enable watchdog timer function

WDFLAG — Watchdog Timer Counter Clear Flag Register F9AH

Bit	3	2	1	0
Identifier	**WDTCF**	**"0"**	**"0"**	**"0"**
RESET **Value**	0	0	0	0
Read/Write	W	W	W	W
Bit Addressing	1/4	1/4	1/4	1/4

WDTCF **Watchdog Timer Counter Clear Flag**

1	Clears the watchdog timer counter

.2–.0 **Bits 2-0**

0	Always logic zero

NOTE: After watchdog timer is cleared by writing "1", this bit is cleared to "0" automatically.

WMOD — Watch Timer Mode Register WT F89H, F88H

Bit	3	2	1	0	3	2	1	0
Identifier	**.7**	**"0"**	**.5**	**.4**	**"0"**	**.2**	**.1**	**"0"**
RESET **Value**	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W
Bit Addressing	8	8	8	8	8	8	8	8

WMOD.7 **Enable/Disable Buzzer Output Bit**

0	Disable buzzer (BUZ) signal output
1	Enable buzzer (BUZ) signal output

WMOD.6 **Bit 6**

0	Always logic zero

WMOD.5 – .4 **Output Buzzer Frequency Selection Bits**

0	0	fw/16 buzzer (BUZ) signal output (2 kHz)
0	1	fw/8 buzzer (BUZ) signal output (4 kHz)
1	0	fw/4 buzzer (BUZ) signal output (8 kHz)
1	1	fw/2 buzzer (BUZ) signal output (16 kHz)

WMOD.3 **Bit 3**

0	Always logic zero

WMOD.2 **Enable/Disable Watch Timer Bit**

0	Disable watch timer and clear frequency dividing circuits
1	Enable watch timer

WMOD.1 **Watch Timer Speed Control Bit**

0	Normal speed; set IRQW to 0.5 seconds
1	High-speed operation; set IRQW to 3.91 ms

WMOD.0 **Bit 0**

0	Always logic zero

NOTE: System clock of 4.19 MHz is assumed.

6 OSCILLATOR CIRCUITS

OVERVIEW

The S3C7524/C7528/C7534/C7538 microcontrollers have one oscillator circuit, the system clock circuit, (fx). The CPU and peripheral hardware operates on the system clock frequency supplied through this circuit. Specifically, a clock pulse is required by the following peripheral modules:

— Basic timer

— Timer/counters 0

— Watch timer

— Clock output circuit

Clock Control Registers

The power control register, PCON, is used to select normal CPU operating mode or one of two power-down modes — stop or idle. Bits 3 and 2 of the PCON register can be manipulated by a STOP or IDLE instruction to engage stop or idle power-down mode.

The system clock frequencies can be divided by 4, 8, or 64. By manipulating PCON bits 1 and 0, you select one of the following frequencies as the selected system clock.

$\frac{fx}{4}, \frac{fx}{8}, \frac{fx}{64}$

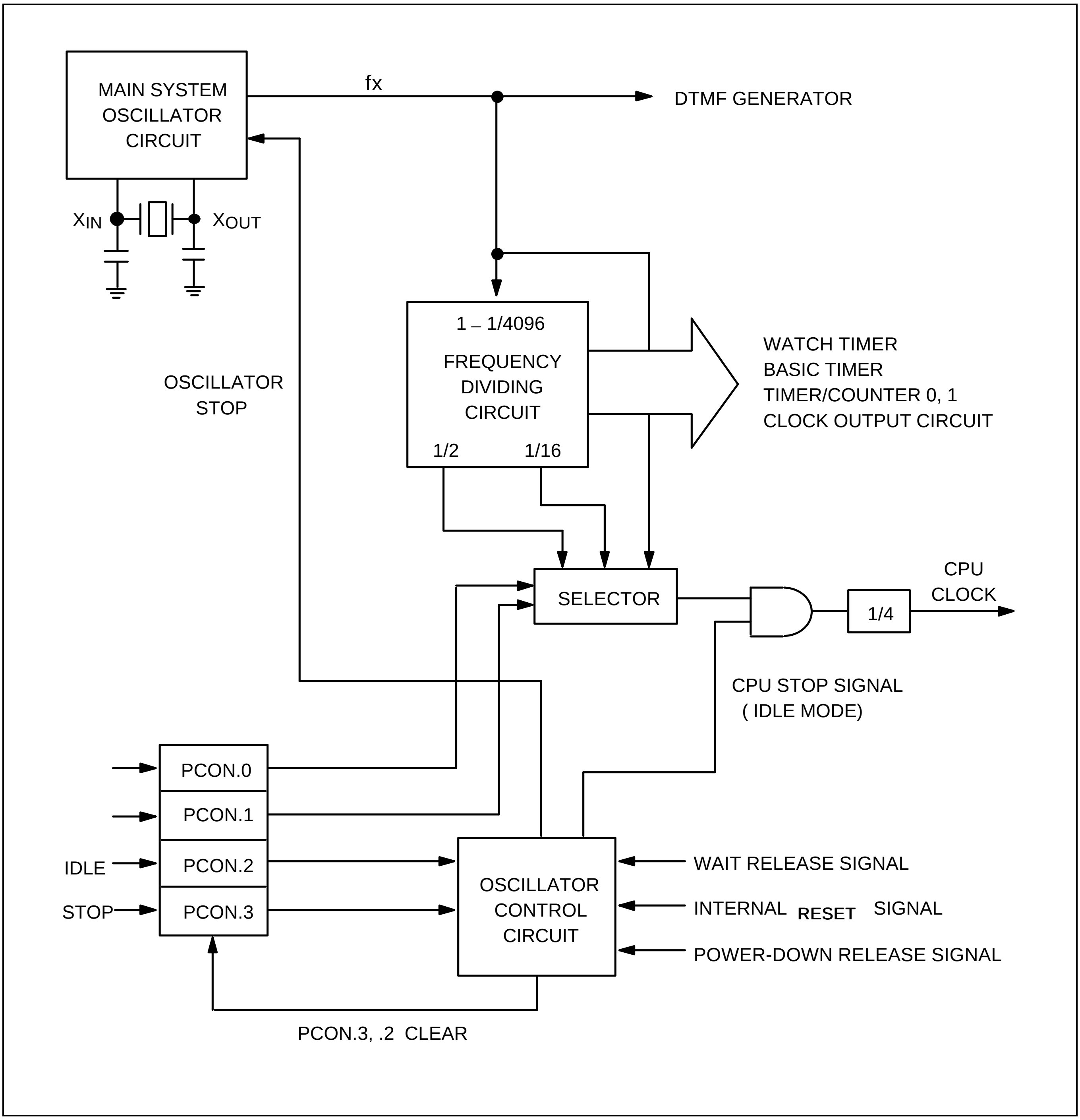


Figure 6–1. Clock Circuit Diagram

System Oscillator Circuits

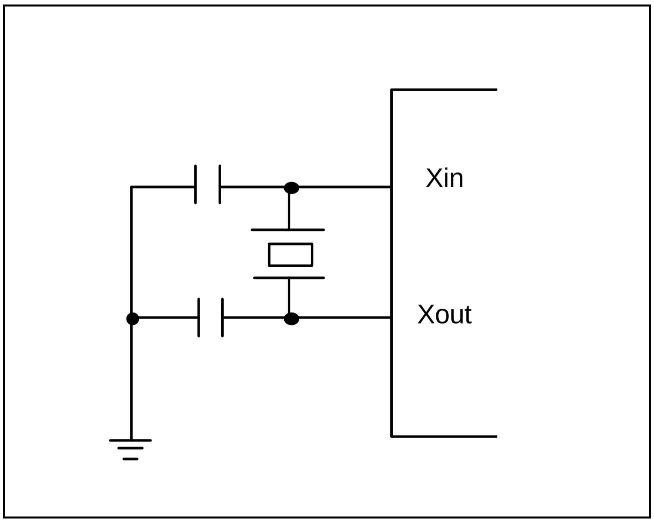


Figure 6–2. Crystal/Ceramic Oscillator

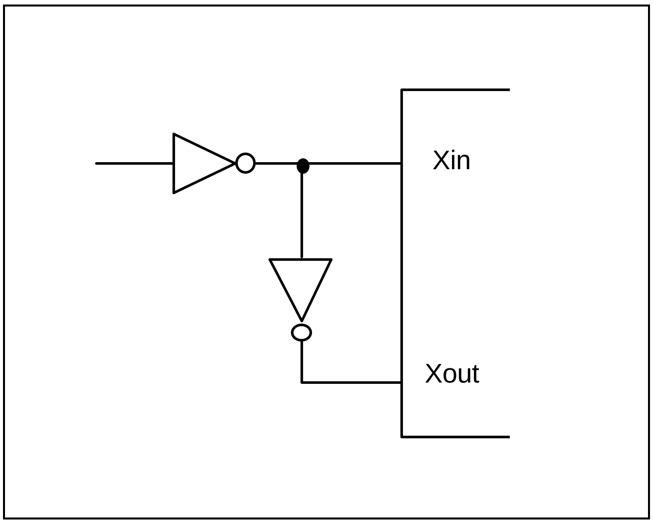


Figure 6–3. External Oscillator

POWER CONTROL REGISTER (PCON)

The power control register, PCON, is a 4-bit register that is used to select the CPU clock frequency and to control CPU operating and power-down modes. PCON can be addressed directly by 4-bit write instructions or indirectly by the instructions IDLE and STOP.

FB3H	PCON.3	PCON.2	PCON.1	PCON.0

PCON bits 3 and 2 are addressed by the STOP and IDLE instructions, respectively, to engage the IDLE and STOP power-down modes. IDLE and STOP modes can be initiated by these instruction despite the current value of the enable memory bank flag (EMB). By manipulating bits 1 and 0 of the PCON register, the system clock frequency can be divided by 4, 8, or 64.

RESET sets PCON register values to logic zero: PCON.1 and PCON.0 divide the fx frequency by 64, and PCON.3 and PCON.2 enable normal CPU operating mode.

Table 6–1. Power Control Register (PCON) Organization

PCON Bit Settings		Resulting CPU Operating Mode
PCON.3	PCON.2	
0	0	Normal CPU operating mode
0	1	Idle power-down mode
1	0	Stop power-down mode

PCON Bit Settings		Resulting CPU Clock Frequency
PCON.1	PCON.0	
0	0	fx/64
1	0	fx/8
1	1	fx/4

☞ PROGRAMMINGNG TIP — Setting the CPU Clock

To set the CPU clock to 1.05 MHz at 4.19 MHz:

```
BITS    EMB
SMB     15
LD      A,#3H
LD      PCON,A
```

INSTRUCTION CYCLE TIMES

The unit of time that equals one machine cycle varies depending on how the oscillator clock signal is divided (by 4, 8, or 64) by the system clock. Table 6–2 shows corresponding cycle times in microseconds.

Selected CPU Clock	Resulting Frequency	Oscillation Source	Cycle Time (µsec)
fx/64	65.5 kHz	fx = 4.19 MHz	15.3
fx/8	524.0 kHz		1.91
fx/4	1.05 MHz		0.95

Table 6–2. Instruction Cycle Times for CPU Clock Rates

CLOCK OUTPUT MODE REGISTER (CLMOD)

The clock output mode register, CLMOD, is a 4-bit register that is used to enable or disable clock output to the CLO pin and to select the CPU clock source and frequency. CLMOD is addressable by 4-bit write instruction only.

FD0H	CLMOD.3	"0"	CLMOD.1	CLMOD.0

RESET clears CLMOD to logic zero, which automatically selects the CPU clock as the clock source (without initiating clock oscillation), and disable clock output.

CLOMD.3 is the enable/disable clock output control bit; CLOMD.1 and CLOMD.0 are used to select one of four possible clock sources and frequencies: normal CPU clock, fx/8, fx/16, or fx/64.

Table 6–3. Clock Output Mode Register (CLMOD) Organization

CLMOD Bit Setting		Resulting Clock Output	
CLMOD.1	CLMOD.0	Clock Source	Frequency
0	0	CPU clock (fx/4, fx/8, fx/64)	1.05 MHz,524 kHz, 65.5 kHz
0	1	fx/8	524 kHz
1	0	fx/16	262 kHz
1	1	fx/64	65.5 kHz

CLMOD.3	Result of CLMOD.3 Setting
0	Clock output is disable
1	Clock output is enable

NOTE: Frequencies assume that fx = 4.19 MHz.

CLOCK OUTPUT CIRCUIT

The clock output circuit, used to output clock pulses to the CLO pin, has the following components:

- 4-bit clock output mode register (CLMOD)
- Clock selector
- output latch
- Port mode flag
- CLO output pin (P2.2)

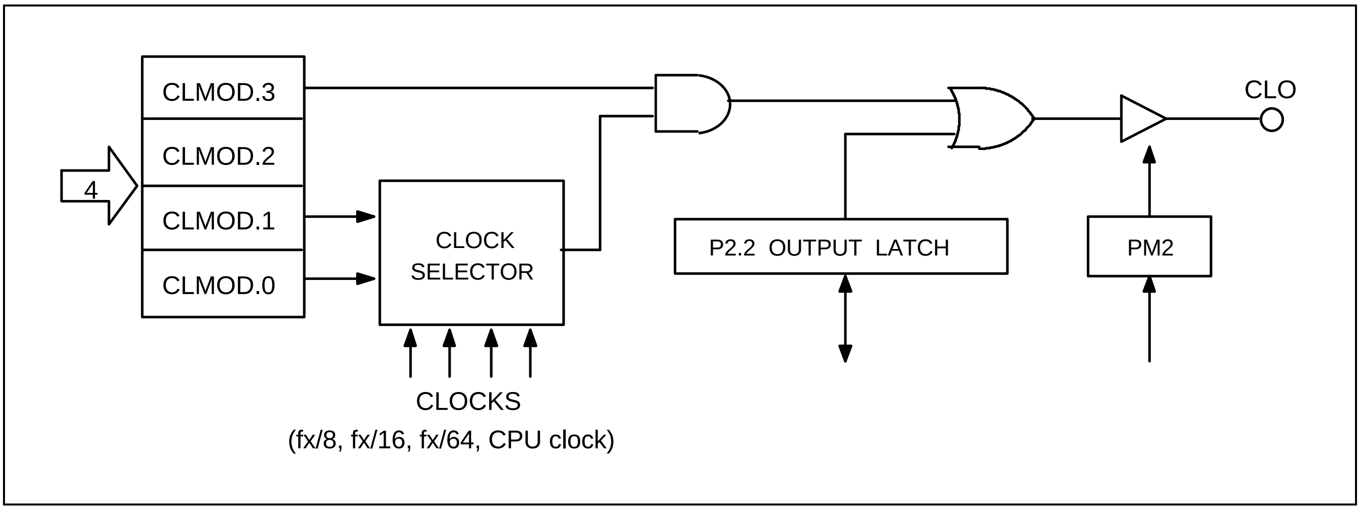


Figure 6–4. CLO Output Pin Circuit Diagram

CLOCK OUTPUT PROCEDURE

The procedure for outputting clock pulses to the CLO pin may be summarized as follows:

1. Disable clock output by clearing CLMOD.3 to logic zero.
2. Set the clock output frequency (CLMOD.1, CLMOD.0).
3. Load a “0” to the output latch of the CLO pin (P2.2).
4. Set the P2.2 mode flag (PM 2) to output mode.
5. Enable clock output by setting CLMOD.3 to logic one.

☞ **PROGRAMMING TIP — CPU Clock Output to the CLO Pin**

To output the CPU clock to the CLO pin

```
BITS    EMB
SMB     15
LD      EA,#04H
LD      PMG1,EA          ;  P 2.2 ← Output mode
BITR    P2.2             ;  Clear P2.2 output latch
LD      A,#8H
LD      CLMOD,A
```

7 INTERRUPTS

OVERVIEW

The S3C7524/C7528/C7534/C7538 interrupt control circuit has five functional components:

— Interrupt enable flags (IEx)
— Interrupt request flags (IRQx)
— Interrupt mask enable register (IME)
— Interrupt priority register (IPR)
— Power-down release signal circuit

Three kinds of interrupts are supported:

— Internal interrupts generated by on-chip processes
— External interrupts generated by external peripheral devices
— Quasi-interrupts used for edge detection and as clock sources

Table 7–1. Interrupt Types and Corresponding Port Pin(s)

Interrupt Type	Interrupt Name	Corresponding Port Pin
External interrupts	INT0, INT1 (note), INT4 (note)	P1.0, P1.1 (note), P1.3 (note)
Internal interrupts	INTB, INTT0, INTT1	Not applicable
Quasi-interrupts	INT2	P1.2 (note), KS0–KS7
	INTW	Not applicable

NOTE: S3C7534/C7538 does not use the interrupts of INT1, INT4, and INT2 pin.

VECTORED INTERRUPTS

Interrupt requests may be processed as vectored interrupts in hardware, or they can be generated by program software. A vectored interrupt is generated when the following flags and register settings, corresponding to the specific interrupt (INTn) are set to logic one:

- Interrupt enable flag (IEx)
- Interrupt master enable flag (IME)
- Interrupt request flag (IRQx)
- Interrupt status flags (IS0, IS1)
- Interrupt priority register (IPR)

If all conditions are satisfied for the execution of a requested service routine, the start address of the interrupt is loaded into the program counter and the program starts executing the service routine from this address.

EMB and ERB flags for RAM memory banks and registers are stored in the vector address area of the ROM during interrupt service routines. The flags are stored at the beginning of the program with the VENT instruction. The initial flag values determine the vectors for resets and interrupts. Enable flag values are saved during the main routine, as well as during service routines. Any changes that are made to enable flag values during a service routine are not stored in the vector address.

When an interrupt occurs, the enable flag values before the interrupt is initiated are saved along with the program status word (PSW), and the enable flag values for the interrupt is fetched from the respective vector address. Then, if necessary, you can modify the enable flags during the interrupt service routine. When the interrupt service routine is returned to the main routine by the IRET instruction, the original values saved in the stack are restored and the main program continues program execution with these values.

Software-Generated Interrupts

To generate an interrupt request from software, the program manipulates the appropriate IRQx flag. When the interrupt request flag value is set, it is retained until all other conditions for the vectored interrupt have been met, and the service routine can be initiated.

Multiple Interrupts

By manipulating the two interrupt status flags (IS0 and IS1), you can control service routine initialization and thereby process multiple interrupts simultaneously.

If more than four interrupts are being processed at one time, you can avoid possible loss of working register data by using the PUSH RR instruction to save register contents to the stack before the service routines are executed in the same register bank. When the routines have executed successfully, you can restore the register contents from the stack to working memory using the POP instruction.

Power-Down Mode Release

An interrupt can be used to release power-down mode (stop or idle). Interrupts for power-down mode release are initiated by setting the corresponding interrupt enable flag. Even if the IME flag is cleared to zero, power-down mode will be released by an interrupt request signal when the interrupt enable flag has been set. In such cases, the interrupt routine will not be executed since IME = "0".

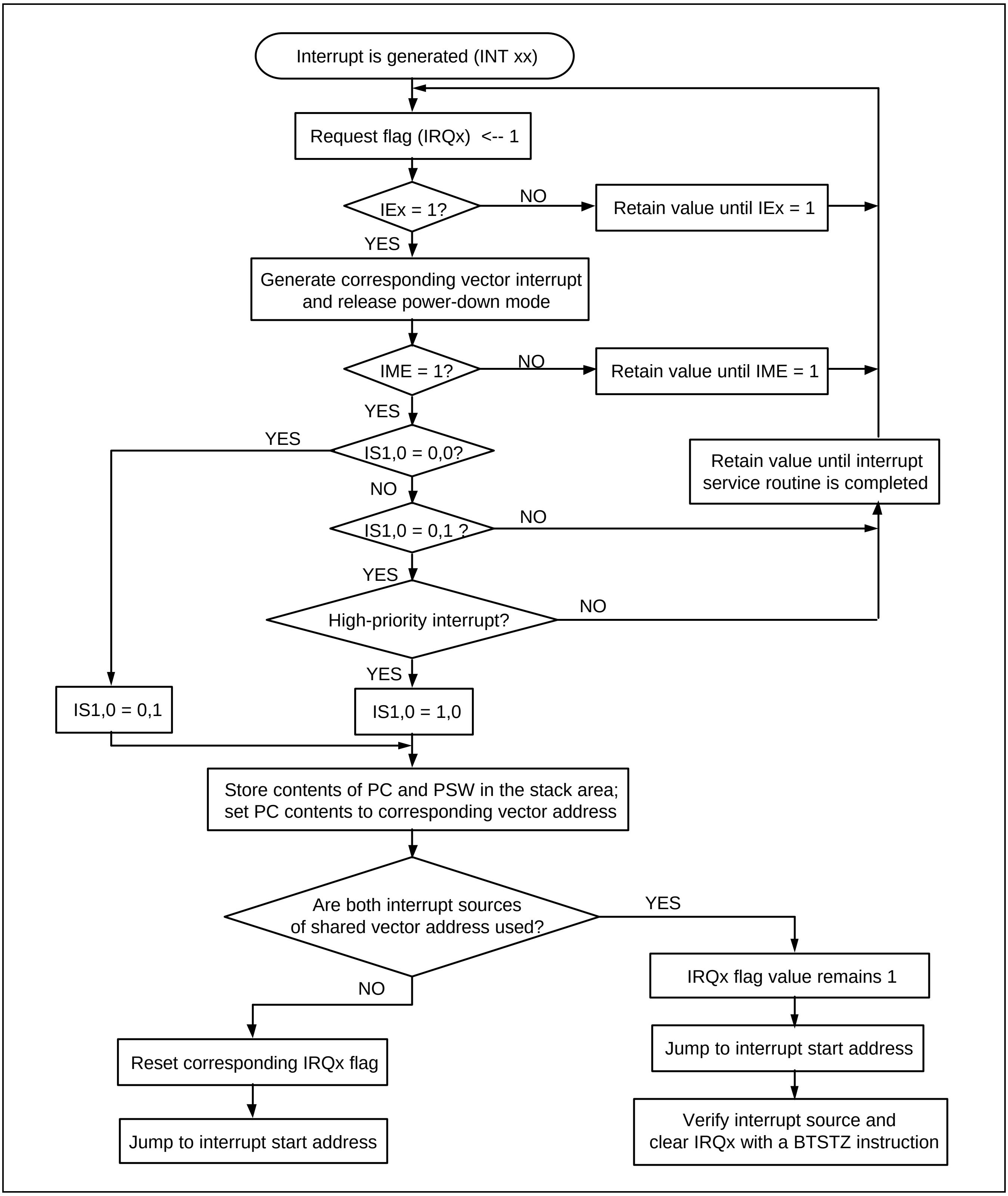


Figure 7-1. Interrupt Execution Flowchart

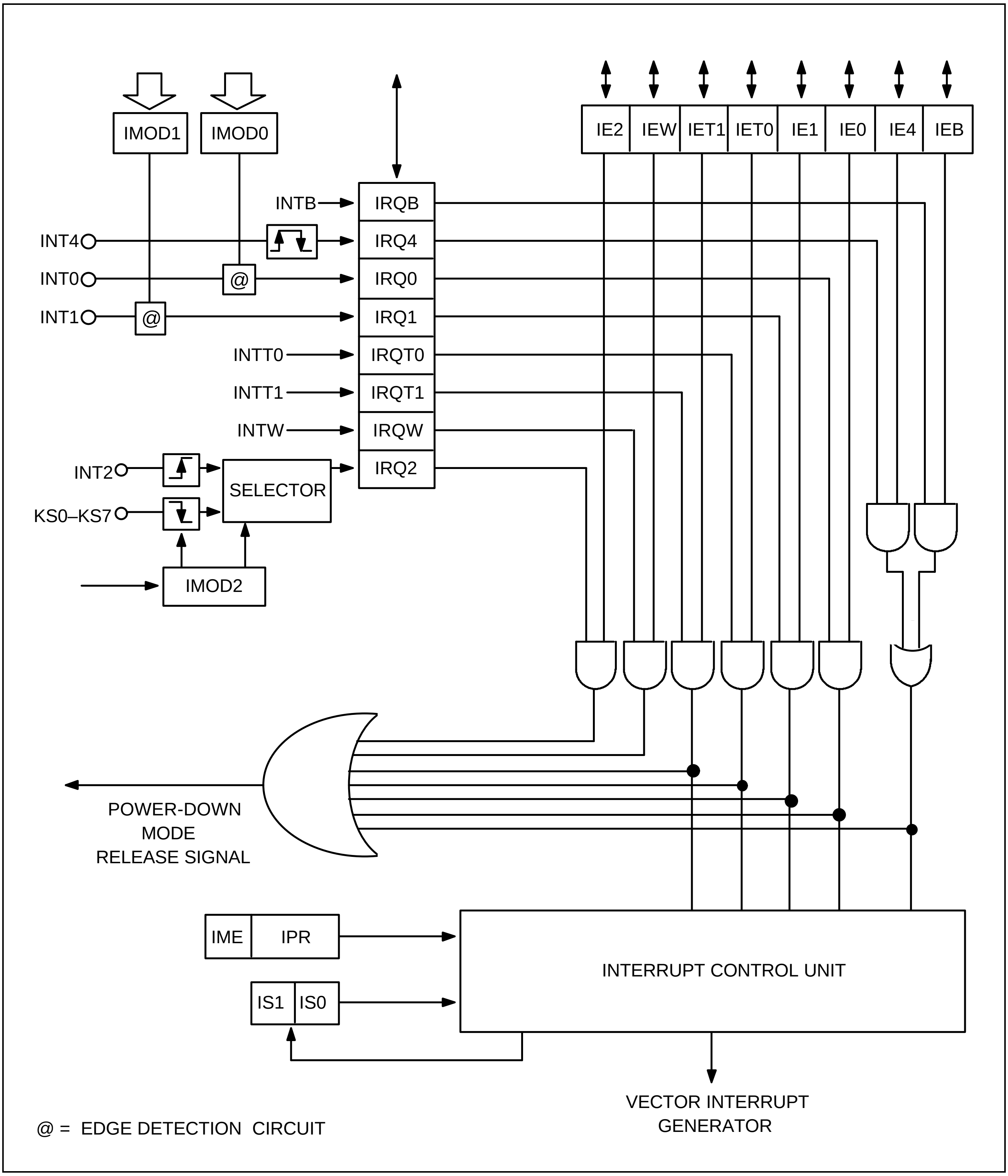


Figure 7-2. Interrupt Control Circuit Diagram (S3C7524/C7528)

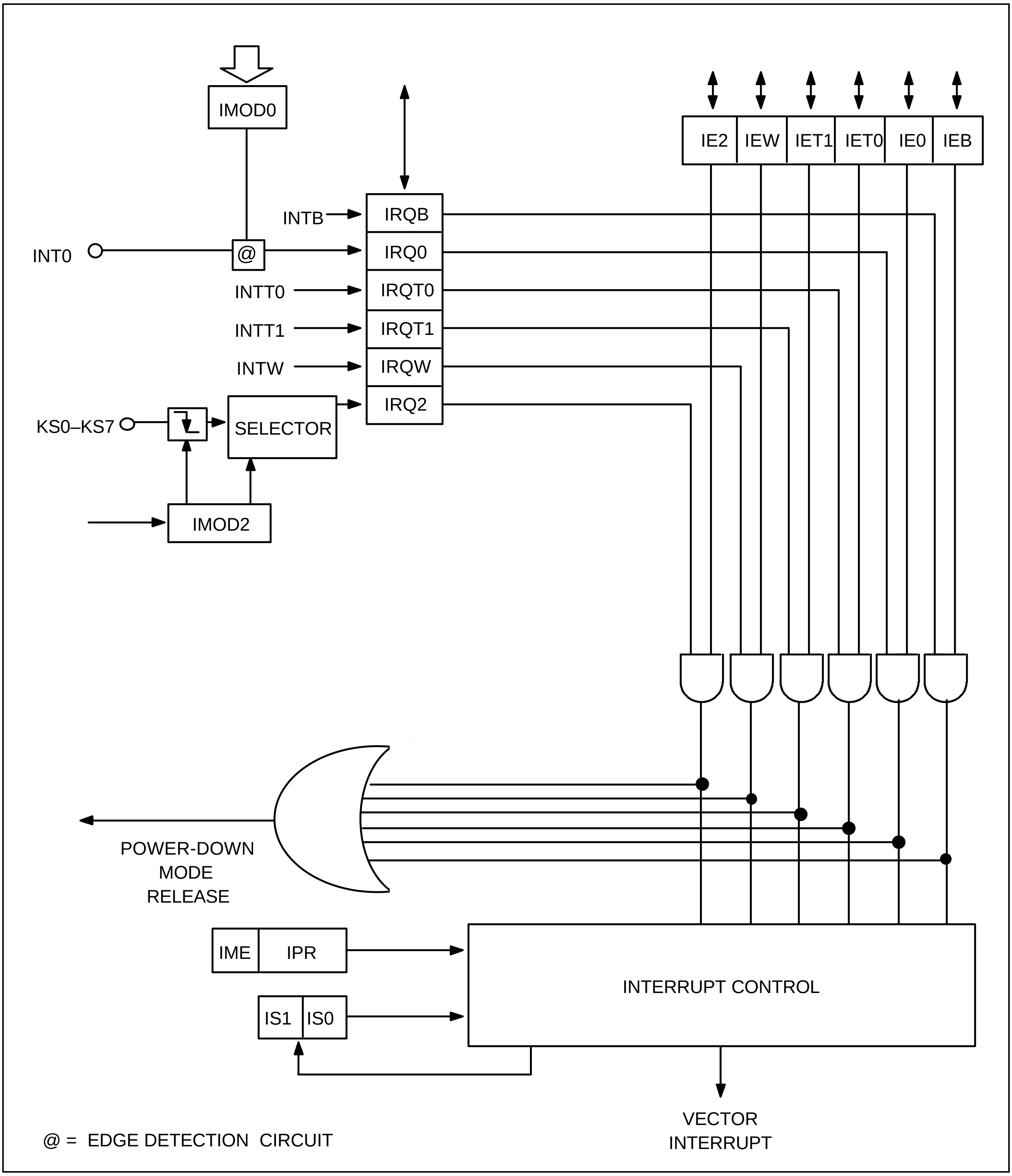


Figure 7-3. Interrupt Control Circuit Diagram (S3C7534/C7538)

Multiple Interrupts

The interrupt controller can service multiple interrupts in two ways: as two-level interrupts, where either all interrupt requests or only those of highest priority are serviced, or as multi-level interrupts, when the interrupt service routine for a lower-priority request is accepted during the execution of a higher priority routine.

Two-Level Interrupt Handling

Two-level interrupt handling is the standard method for processing multiple interrupts. When the IS1 and IS0 bits of the PSW (FB0H.3 and FB0H.2, respectively) are both logic zero, program execution mode is normal and all interrupt requests are serviced (see Figure 7–3).

Whenever an interrupt request is accepted, IS1 and IS0 are incremented by one ("0" → "1" or "1" → "0"), and the values are stored in the stack along with the other PSW bits. After the interrupt routine has been serviced, the modified IS1 and IS0 values are automatically restored from the stack by an IRET instruction.

IS0 and IS1 can be manipulated directly by 1-bit write instructions, regardless of the current value of the enable memory bank flag (EMB). Before you can modify an interrupt status flag, however, you must first disable interrupt processing with a DI instruction.

When IS1 = "0" and IS0 = "1", all interrupt service routines are inhibited except for the highest priority interrupt currently defined by the interrupt priority register (IPR).

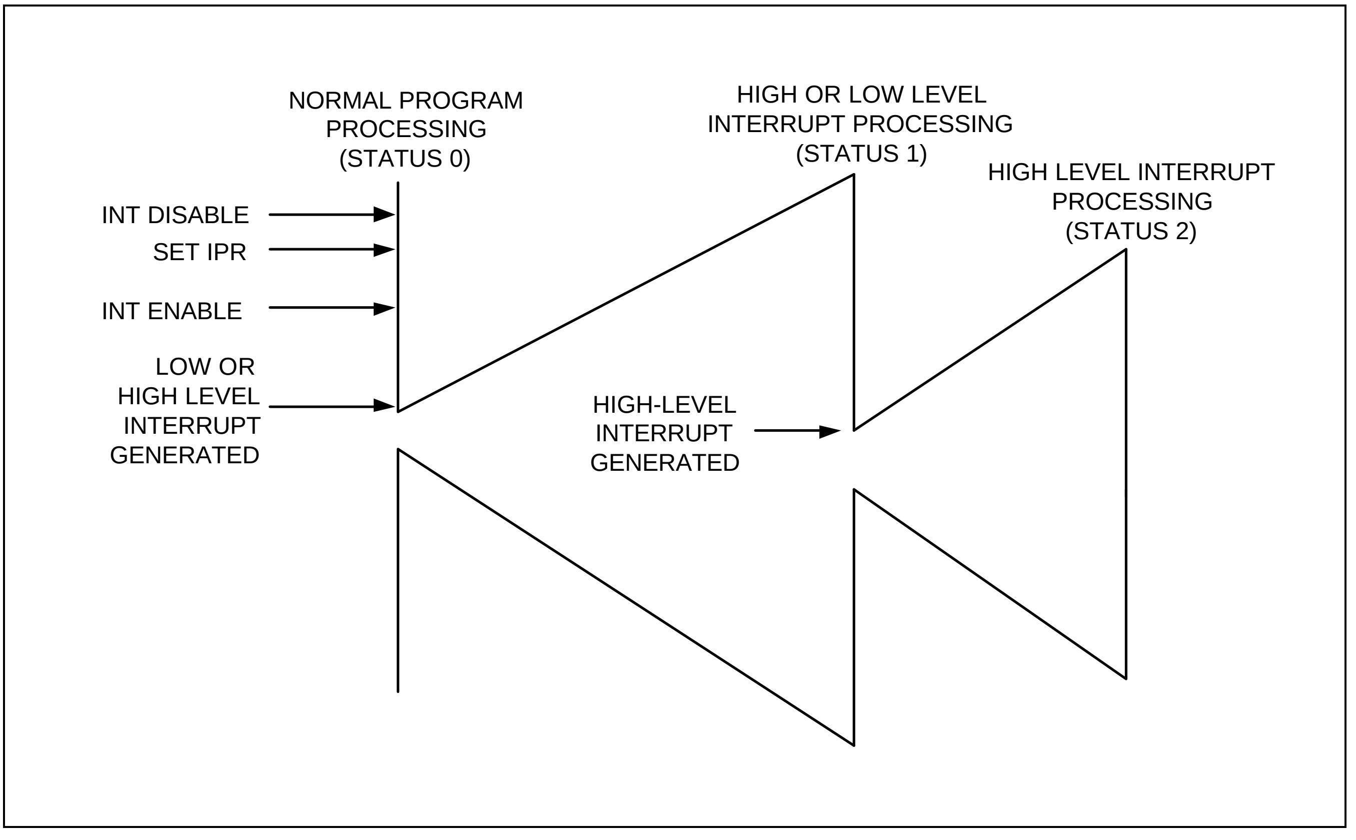


Figure 7-4. Two-Level Interrupt Handling

Multi-Level Interrupt Handling

With multi-level interrupt handling, a lower-priority interrupt request can be executed while a high-priority interrupt is being serviced. This is done by manipulating the interrupt status flags, IS0 and IS1 (see Table 7–2).

When an interrupt is requested during normal program execution, interrupt status flags IS0 and IS1 are set to "1" and "0", respectively. This setting allows only highest-priority interrupts to be serviced. When a high-priority request is accepted, both interrupt status flags are then cleared to "0" by software so that a request of any priority level can be serviced. In this way, the high- and low-priority requests can be serviced in parallel (see Figure 7–4).

Table 7-2. IS1 and IS0 Bit Manipulation for Multi-Level Interrupt Handling

Process Status	Before INT		Effect of ISx Bit Setting	After INT ACK	
	IS1	IS0		IS1	IS0
0	0	0	All interrupt requests are serviced.	0	1
1	0	1	Only high-priority interrupts as determined by the current settings in the IPR register are serviced.	1	0
2	1	0	No additional interrupt requests will be serviced.	–	–
–	1	1	Value undefined	–	–

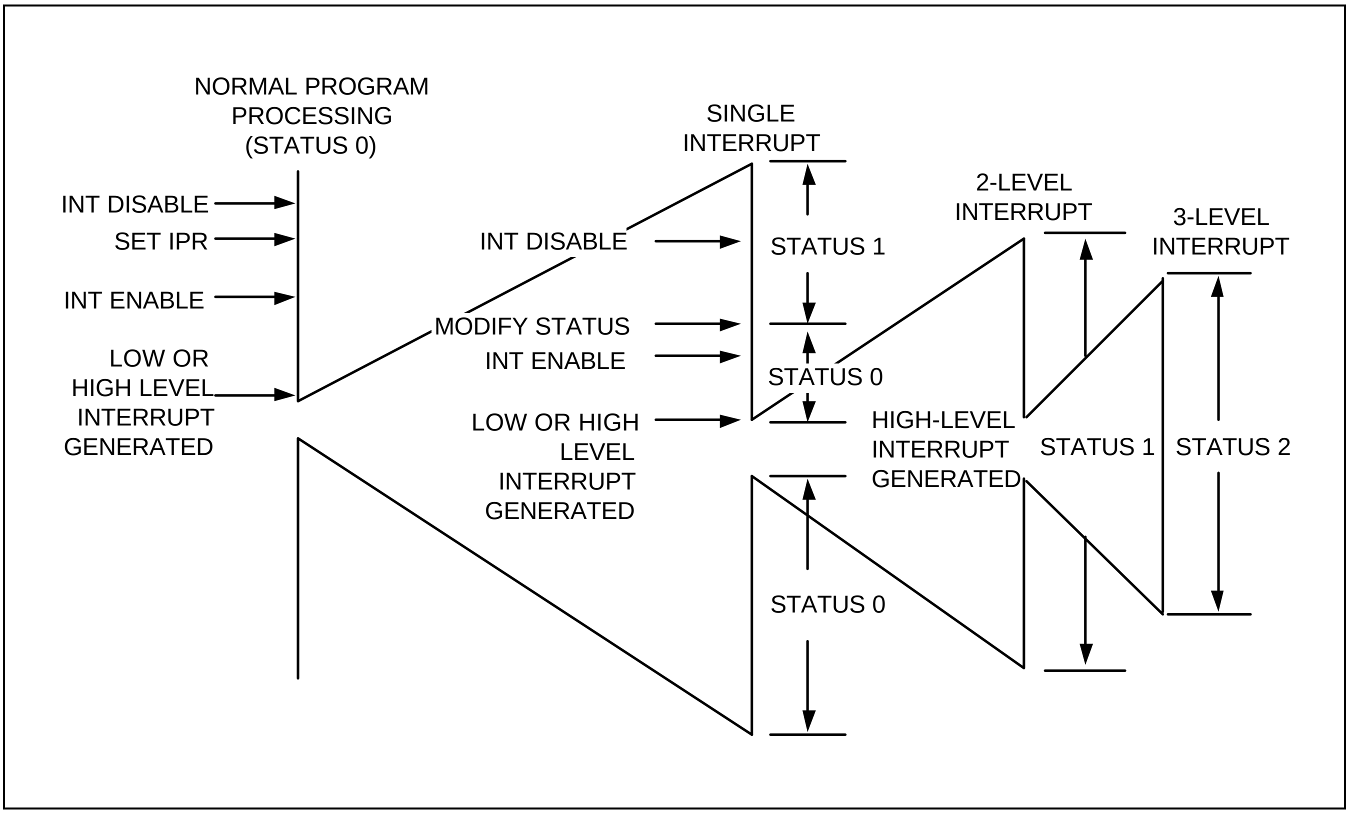


Figure 7-5. Multi-Level Interrupt Handling

INTERRUPT PRIORITY REGISTER (IPR)

The 4-bit interrupt priority register (IPR) is used to control multi-level interrupt handling. Its reset value is logic zero. Before the IPR can be modified by 4-bit write instructions, all interrupts must first be disabled by a DI instruction.

FB2H	IME	IPR.2	IPR.1	IPR.0

By manipulating the IPR settings, you can choose to process all interrupt requests with the same priority level, or you can select one type of interrupt for high-priority processing. A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt cannot be interrupted by any other interrupt source.

Table 7-3. Standard Interrupt Priorities

Interrupt	Default Priority
INTB, INT4 (note)	1
INT0	2
INT1 (note)	3
INTT0	4
INTT1	5

The MSB of the IPR, the interrupt master enable flag (IME), enables and disables all interrupt processing. Even if an interrupt request flag and its corresponding enable flag are set, a service routine cannot be executed until the IME flag is set to logic one. The IME flag can be directly manipulated by EI and DI instructions, regardless of the current enable memory bank (EMB) value.

Table 7-4. Interrupt Priority Register Settings

IPR.2	IPR.1	IPR.0	Result of IPR Bit Setting
0	0	0	Normal interrupt handling according to default priority settings
0	0	1	Process INTB and INT4 (note) interrupts at highest priority
0	1	0	Process INT0 interrupts at highest priority
0	1	1	Process INT1 (note) interrupts at highest priority
1	0	1	Process INTT0 interrupts at highest priority
1	1	0	Process INTT1 interrupts at highest priority

NOTES

1. During normal interrupt processing, interrupts are processed in the order in which they occur. If two or more interrupts occur simultaneously, the processing order is determined by the default interrupt priority settings shown in Table 7-3. Using the IPR settings, you can select specific interrupts for high-priority processing in the event of contention. When the high-priority (IPR) interrupt has been processed, waiting interrupts are handled according to their default priorities.
2. S3C7534/C7538 does not use the INT1 and INT4 interrupts.

☞ PROGRAMMING TIP — Setting the INT Interrupt Priority

The following instruction sequence sets the INT1 interrupt to high priority:

```
BITS    EMB
SMB     15
DI              ;       IPR.3 (IME) ← 0
LD      A,#3H
LD      IPR,A
EI              ;       IPR.3 (IME) ← 1
```

EXTERNAL INTERRUPT 0 and 1 MODE REGISTERS (IMOD0, IMOD1)

The following components are used to process external interrupts at the INT0 and INT1 (note) pin:

- Edge detection circuit
- Two mode registers, IMOD0 and IMOD1 (note)

The mode registers are used to control the triggering edge of the input signal. IMOD0 and IMOD1 (note) settings let you choose either the rising or falling edge of the incoming signal as the interrupt request trigger.
The INT4 (note) interrupt is an exception since its input signal generates an interrupt request on both rising and falling edges.

FB4H	"0"	"0"	IMOD0.1	IMOD0.0
FB5H	"0"	"0"	"0"	IMOD1.0 (NOTE)

IMOD0 and IMOD1 (note) bits are addressable by 4-bit write instructions. RESET clears all IMOD values to logic zero, selecting rising edges as the trigger for incoming interrupt requests.

Table 7-5. IMOD0 and IMOD1 Register Organization

IMOD0	**0**	**0**	**IMOD0.1**	**IMOD0.0**	**Effect of IMOD0 Settings**
			0	0	Rising edge detection
			0	1	Falling edge detection
			1	0	Both rising and falling edge detection
			1	1	IRQ0 flag cannot be set to "1"

IMOD1 (note)	**0**	**0**	**0**	**IMOD1.0**	**Effect of IMOD1 Settings**
				0	Rising edge detection
				1	Falling edge detection

NOTE: S3C7534/C7538 does not use the INT1 and INT4 interrupts.

EXTERNAL INTERRUPT 0 AND 1 MODE REGISTERS (CONTINUED)

When a sampling clock rate of fx/64 is used for INT0, an interrupt request flag must be cleared before 16 machine cycles have elapsed.

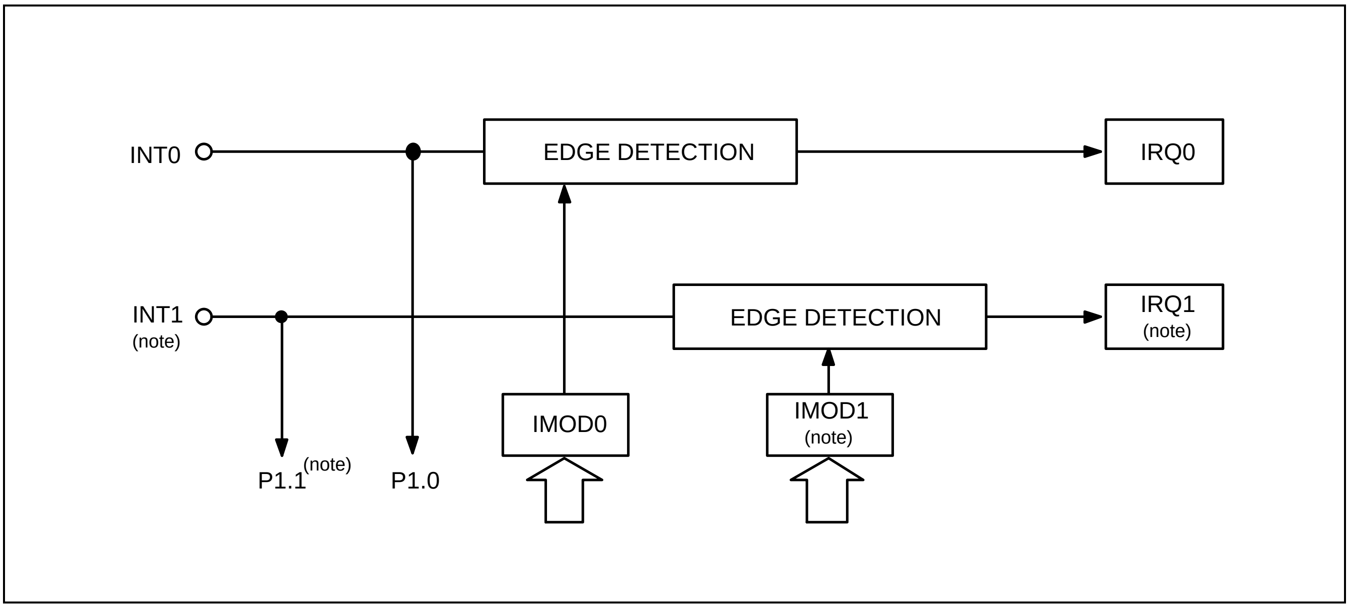


Figure 7-6. Circuit Diagram for INT0 and INT1 Pins

When modifying the IMOD0 and IMOD1 registers, it is possible to accidentally set an interrupt request flag. To avoid unwanted interrupts, take these precautions when writing your programs:

1. Disable all interrupts with a DI instruction.
2. Modify the IMOD0 or IMOD1 register.
3. Clear all relevant interrupt request flags.
4. Enable the interrupt by setting the appropriate IEx flag.
5. Enable all interrupts with an EI instructions.

NOTE: S3C7534/C7538 does not use the INT1 and P1.1.

EXTERNAL INTERRUPT 2 MODE REGISTER (IMOD2)

The mode register for external interrupts at the KS0–KKS7 pins, IMOD2, is addressable only by 4-bit write instructions. RESET clears all IMOD2 bits to logic zero.

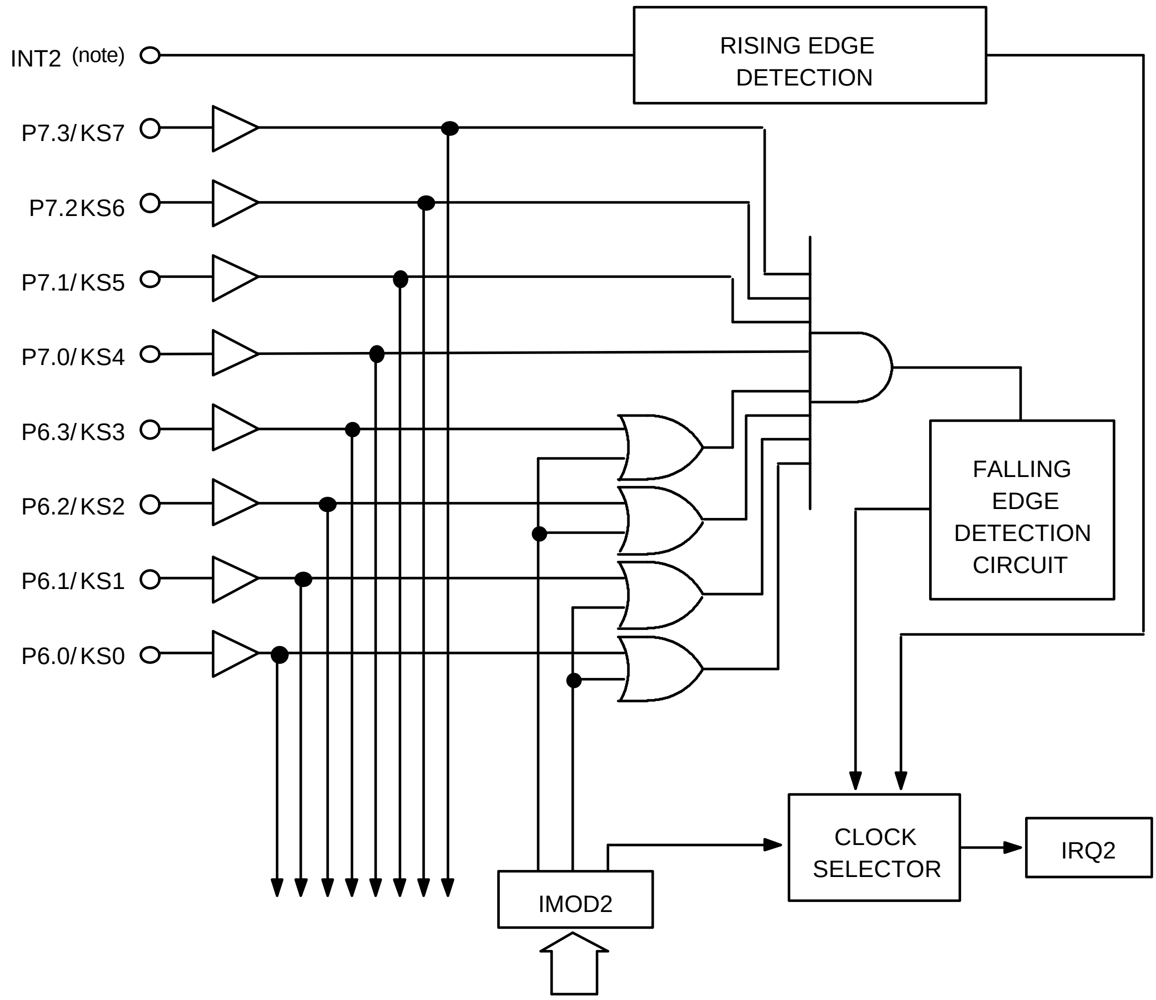
FB6H	"0"	"0"	IMOD2.1	IMOD2.0

When IMOD2 is cleared to logic zero, INT2 uses the rising edge of an incoming signal as the interrupt request trigger. If a rising edge is detected at the INT2 pin, or when a falling edge is detected at any one of the pins KS0–KS7, the IRQ2 flag is set to logic one and a release signal for power-down mode is generated.

Table 7-6. IMOD2 Register Bit Settings

IMOD2	**0**	**0**	**IMOD2.1**	**IMOD2.0**	**Effect of IMOD2 Settings**
			0	0	Select rising edge at INT2 pin (note)
			0	1	Select falling edge at KS4–KS7
			1	0	Select falling edge at KS2–KS7
			1	1	Select falling edge at KS0–KS7

NOTE: S3C7534/C7538 does not use the INT2 pin.



NOTES

1. To generate a key interrupt on a falling edge at KS0-KS7, all KS0-KS7 pins must be configured to the input mode. Particularly, the KS4-KS7 must always be configured to the input mode.
2. S3C7534/C7538 does not use the INT2 pin.

Figure 7-7. Circuit Diagram for INT2 and KS0–KS7 Pins

☞ PROGRAMMING TIP — Using INT2 as a Key Input Interrupt

When the INT2 interrupt is used as a key interrupt, the selected key interrupt source pin must be set to input:

1. When KS0–KS7 are selected (eight pins):

```
BITS    EMB
SMB     15
LD      A, #3H
LD      IMOD2, A        ;   (IMOD2) ← #3H, KS0–KS7 falling edge select
LD      EA, #00H
LD      PMG3, EA        ;   P6, 7 ← input mode
LD      A, #3H
LD      PUMOD2, A       ;   Enable P6 and P7 pull-up resistors
```

2. When KS2–KS7 are selected (six pins):

```
BITS    EMB
SMB     15
LD      A, #2H
LD      IMOD2, A        ;   (IMOD2) ← #2H, KS2–KS7 falling edge select
LD      EA, #03H
LD      PMG3, EA        ;   P7, P6.2–P6.3 ← input mode
LD      A, #3H
LD      PUMOD2, A       ;   Enable P6 and P7 pull-up resistors
```

3. When KS4–KS7 are selected (four pins), P7 must be specified as a key strobe signal input:

```
BITS    EMB
SMB     15
LD      A, #1H
LD      IMOD2, A        ;   (IMOD2) ← #1H, KS4–KS7 falling edge select
LD      EA, #0FH
LD      PMG3, EA
LD      A, #2
LD      PUMOD2, A       ;   Enable P7 pull-up resistor
```

INTERRUPT FLAGS

There are three types of interrupt flags: interrupt request and interrupt enable flags that correspond to each interrupt, the interrupt master enable flag, which enables or disables all interrupt processing.

Interrupt Master Enable Flag (IME)

The interrupt master enable flag, IME, enables or disables all interrupt processing. Therefore, even when an IRQx flag is set and its corresponding IEx flag is enabled, the interrupt service routine is not executed until the IME flag is set to logic one.

The IME flag is located in the IPR register (IPR.3). It can be directly manipulated by EI and DI instructions, regardless of the current value of the enable memory bank flag (EMB).

IME	IPR.2	IPR.1	IPR.0	Effect of Bit Settings
0				Inhibit all interrupts
1				Enable all interrupts

Interrupt Enable Flags (IEx)

IEx flags, when set to logical one, enable specific interrupt requests to be serviced. When the interrupt request flag is set to logic one, an interrupt will not be serviced until its corresponding IEx flag is also enabled.

Interrupt enable flags can be read, written, or tested directly by 1-bit instructions (BITS and BITR) or 4-bit instructions. IEx flags can be addressed directly at their specific RAM addresses, despite the current value of the enable memory bank (EMB) flag.

Table 7-7. Interrupt Enable and Interrupt Request Flag Addresses

Address	Bit 3	Bit 2	Bit 1	Bit 0
FB8H	IE4 (5)	IRQ4 (5)	IEB	IRQB
FBAH	0	0	IEW	IRQW
FBBH	0	0	IET1	IRQT1
FBCH	0	0	IET0	IRQT0
FBEH	IE1 (5)	IRQ1 (5)	IE0	IRQ0
FBFH	0	0	IE2	IRQ2

NOTES
1. Iex refers generically to all interrupt enable flags.
2. IRQx refers generically to all interrupt request flags.
3. IEx = 0 is interrupt disable mode.
4. IEx = 1 is interrupt enable mode.
5. S3C7534/C7538 does not use the IE1, IE4, IRQ1, and IRQ4 interrupts.

Interrupt Request Flags (IRQx)

Interrupt request flags, are read/write addressable by 1-bit or 4-bit instructions. IRQx flags can be addressed directly at their specific RAM addresses, regardless of the current value of the enable memory bank (EMB) flag.

When a specific IRQx flag is set to logic one, the corresponding interrupt request is generated. The flag is then automatically cleared to logic zero when the interrupt has been serviced. Exceptions are the watch timer interrupt request flags, IRQW, and the external interrupt 2 flag IRQ2, which must be cleared by software after the interrupt service routine has executed. IRQx flags are also used to execute interrupt requests from software. In summary, follow these guidelines for using IRQx flags:

1. IRQx is set to request an interrupt when an interrupt meets the set condition for interrupt generation.
2. IRQx is set to "1" by hardware and then cleared by hardware when the interrupt has been serviced (with the exception of IRQW and IRQ2).
3. If IRQx is set to "1" by software, an interrupt is also generated.

When two interrupts share the same service routine start address, interrupt processing may occur in one of two ways:

— When only one interrupt is enabled, the IRQx flag is cleared automatically when the interrupt has been serviced.

— When two interrupts are enabled, the request flag is not automatically cleared so that the user has an opportunity to locate the source of the interrupt request. In this case, the IRQx setting must be cleared manually using a BTSTZ instruction.

Table 7-8. Interrupt Request Flag Conditions and Priorities

Interrupt Source	Internal / External	Pre-condition for IRQx Flag Setting	Interrupt Priority	IRQ Flag Name
INTB	I	Reference time interval signal from basic timer	1	IRQB
INT4 (2)	E	Both rising and falling edges detected at INT4	1	IRQ4
INT0	E	Rising or falling edge detected at INT0 pin	2	IRQ0
INT1 (2)	E	Rising or falling edge detected at INT1 pin	3	IRQ1
INTT0	I	Signals for TCNT0 and TREF0 registers match	5	IRQT0
INTT1	I	Signals for TCNT1 and TREF1 registers match	6	IRQT1
INT2 (2)	E	Rising edge detected at INT2 pin or else a falling edge is detected at any of the KS0–KS7 pins	–	IRQ2
INTW	I	Time interval of 0.5 secs or 3.19 msecs	–	IRQW

NOTES

1. The quasi-interrupt INT2 is only used for testing incoming signals.
2. S3C7534/C7538 does not use the INT2 pin, INT1, and INT4 interrupts.

☞ PROGRAMMING TIP — Enabling the INTB and INT4 Interrupts

To simultaneously enable INTB and INT4 interrupts:

```
INTB    DI
        BTSTZ   IRQB        ;   IRQB = 1 ?
        JR      INT4        ;   If no, INT4 interrupt; if yes, INTB interrupt is processed
        •
        •
        •
        EI
        IRET

INT4    BITR    IRQ4        ;   INT4 is processed
        •
        •
        •
        EI
        IRET
```

8 POWER-DOWN

OVERVIEW

The S3C7524/C7528/C7534/C7538 microcontroller has two power-down modes to reduce power consumption: idle and stop. Idle mode is initiated by the IDLE instruction and stop mode by the instruction STOP. (Several NOP instructions must always follow an IDLE or STOP instruction in a program.) In idle mode, the CPU clock stops while peripherals and the oscillation source continue to operate normally.

When RESET occurs during normal operation or during a power-down mode, a reset operation is initiated and the CPU enters idle mode. When the standard oscillation stabilization time interval (31.3 ms at 4.19 MHz) has elapsed, normal CPU operation resumes.

In stop mode, system clock oscillation is halted (assuming it is currently operating), and peripheral hardware components are powered-down. The effect of stop mode on specific peripheral hardware components — CPU, basic timer, timer/counters, and watch-timer — and on external interrupt requests, is detailed in Table 8–1.

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.

Idle or stop modes are terminated either by a RESET, or by an interrupt with the exception of INT0, which are enabled by the corresponding interrupt enable flag, IEx. When power-down mode is terminated by RESET input, a normal reset operation is executed. Assuming that both the interrupt enable flag and the interrupt request flag are set to "1", power-down mode is released immediately upon entering power-down mode.

When an interrupt is used to release power-down mode, the operation differs depending on the value of the interrupt master enable flag (IME):

— If the IME flag = "0", program execution is started immediately after the instruction which issues the request to enter power-down mode. The interrupt request flag remains set to logic one.

— If the IME flag = "1", two instructions are executed after the power-down mode release. Then, the vectored interrupt is initiated. However, when the release signal is caused by INT2 or INTW, the operation is identical to the IME = 0 condition. That is, a vector interrupt is not generated.

Table 8–1. Hardware Operation During Power-Down Modes

Operation	Stop Mode (STOP)	Idle Mode (IDLE)
Clock oscillator	System clock oscillation stops	CPU clock oscillation stops (system clock oscillation continues)
Basic timer	Basic timer stops	Basic timer operates (with IRQB set at each reference interval)
Timer/counter 0	Operates only if TCL0 is selected as the counter clock	Timer/counter 0 operates
Timer/counter 1	Operates only if TCL1 is selected as the counter clock	Timer/counter 1 operates
Watch timer	Watch timer operation is stopped	Watch timer operates
External interrupts	INT0, INT1, INT2, and INT4 are acknowledged	INT0, INT1, INT2, and INT4 are acknowledged
CPU	All CPU operations are disabled	All CPU operations are disabled
Power-down mode release signal	Interrupt request signals are enabled by an interrupt enable flag or by RESET input	Interrupt request signals are enabled by an interrupt enable flag or by RESET input

NOTE: S3C7534/C7538 does not use the INT1, INT2, and INT4 interrupts.

IDLE MODE TIMING DIAGRAMS

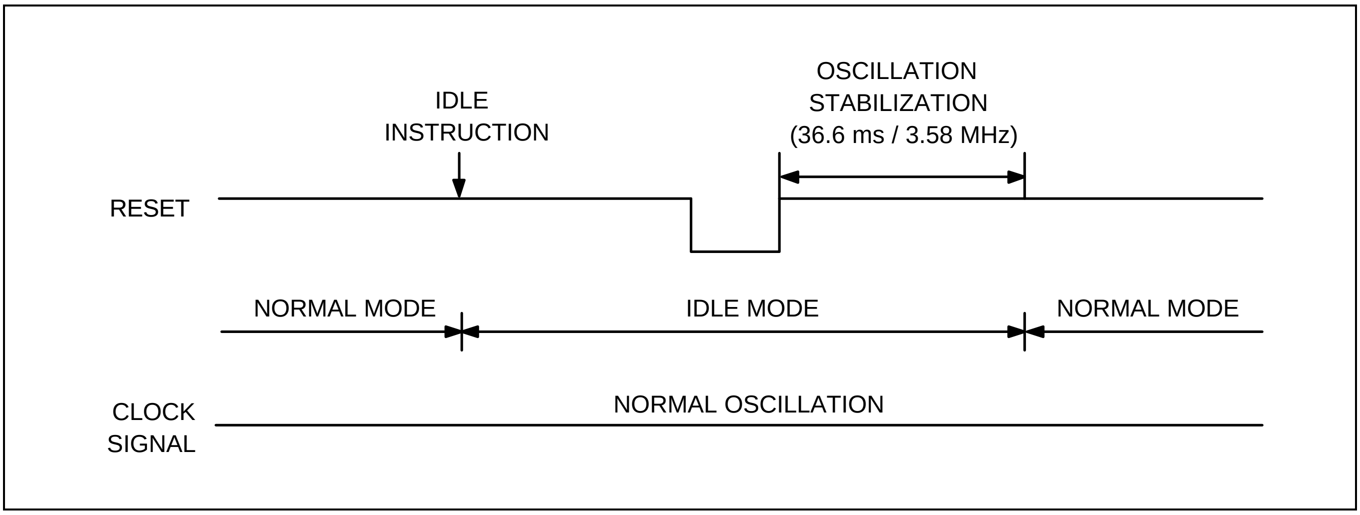


Figure 8–1. Timing When Idle Mode is Released by RESET

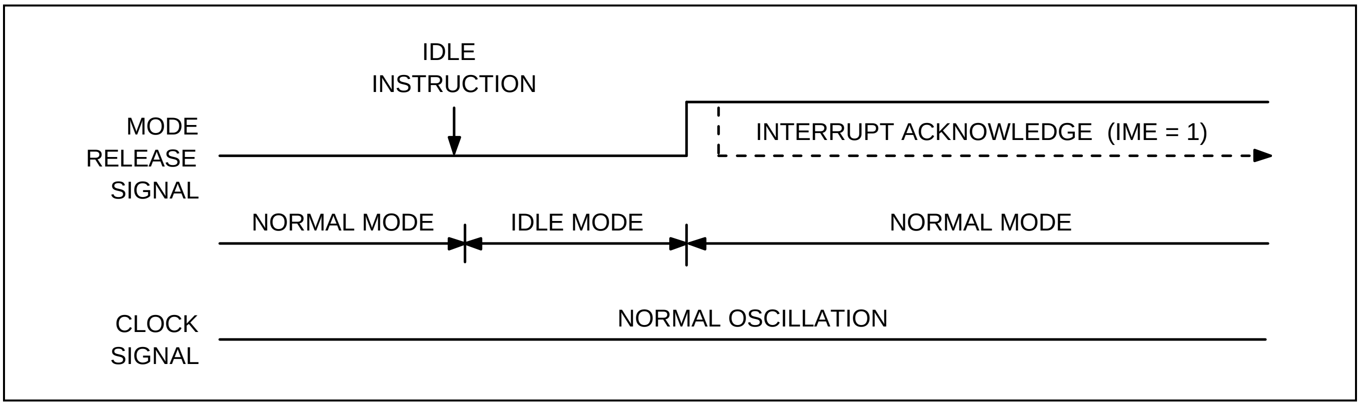


Figure 8–2. Timing When Idle Mode is Released by an Interrupt

STOP MODE TIMING DIAGRAMS

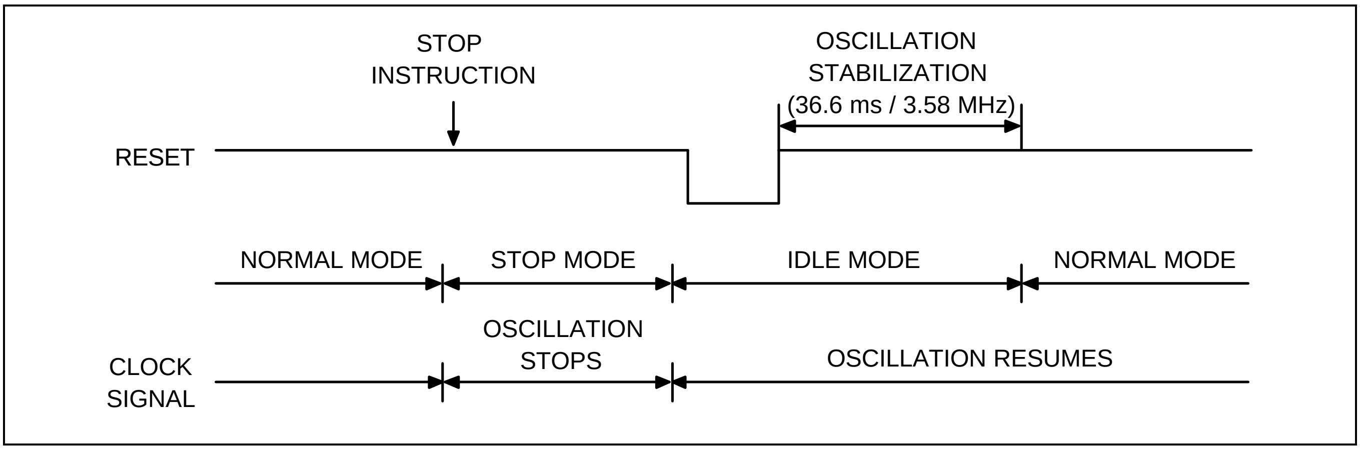


Figure 8–3. Timing When Stop Mode is Released by RESET

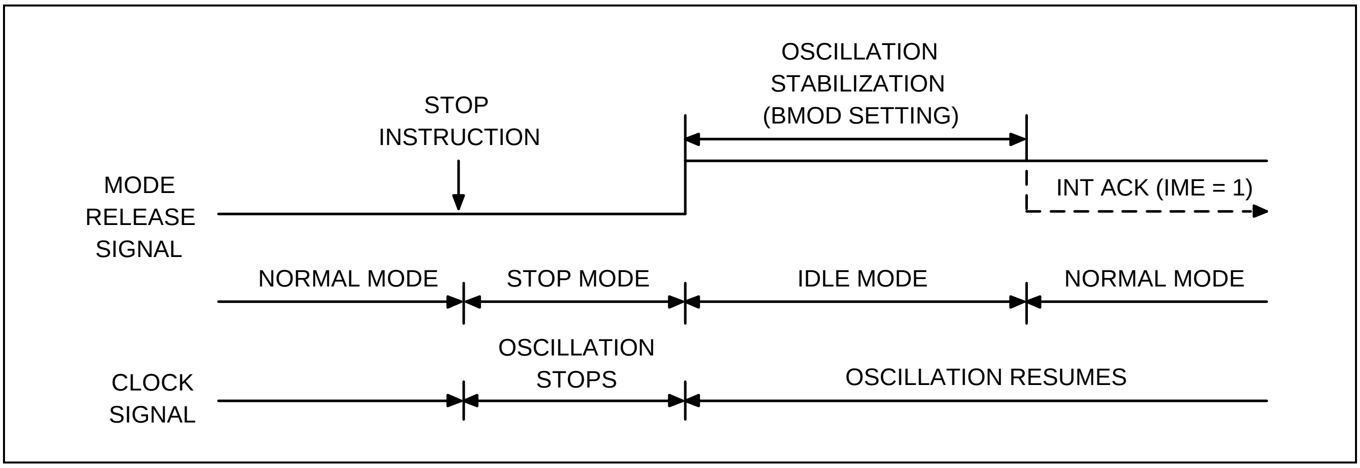


Figure 8–4. Timing When Stop Mode is Release by an Interrupt

PORT PIN CONFIGURATION FOR POWER-DOWN

The following method describes how to configure I/O port pins to reduce power consumption during power-down modes (stop, idle):

Condition 1: If the microcontroller is not configured to an external device:

1. Connect unused port pins according to the information in Table 8–2.
2. Disable all pull-up resistors for output pins by making the appropriate modifications to the pull-up resistor mode register, PUMOD. Reason: If output goes low when the pull-up resistor is enabled, there may be unexpected surges of current through the pull-up.
3. Disable pull-up resistors for input pins configured to V_{DD} or V_{SS} levels in order to check the current input option. Reason: If the input level of a port pin is set to V_{SS} when a pull-up resistor is enabled, it will draw an unnecessarily large current.

Condition 2: If the microcontroller is configured to an external device and the external device's V_{DD} source is turned off in power-down mode.

1. Connect unused port pins according to the information in Table 8–2.
2. Disable the pull-up resistors of output pins by making the appropriate modifications to the pull-up resistor mode register, PUMOD. Reason: If output goes low when the pull-up resistor is enabled, there may be unexpected surges of current through the pull-up.
3. Disable pull-up resistors for input pins configured to V_{DD} or V_{SS} levels in order to check the current input option. Reason: If the input level of a port pin is set to V_{SS} when a pull-up resistor is enabled, it will draw an unnecessarily large current.
4. Disable the pull-up resistors of input pins connected to the external device by making the necessary modifications to the PUMOD register.
5. Configure the output pins that are connected to the external device to low level. Reason: When the external device's V_{DD} source is turned off, and if the microcontroller's output pins are set to high level, V_{DD} – 0.7 V is supplied to the V_{DD} of the external device through its input pin. This causes the device to operate at the level V_{DD} – 0.7 V. In this case, total current consumption would not be reduced.
6. Determine the correct output pin state necessary to block current pass in according with the external transistors (PNP, NPN).

RECOMMENDED CONNECTIONS FOR UNUSED PINS

To reduce overall power consumption, please configure unused pins according to the guidelines described in Table 8–2.

Table 8–2. Unused Pin Connections for Reduced Power Consumption

Pin/Share Pin Names	Recommended Connection
P1.0 / INT0 –P 1.2 / INT2	Connect to V_{DD}
P1.3 / INT4	
P2.0 / TCLO0 P2.1 / TCLO1 P2.2 / CLO P2.3 / BUZ P3.0 / TCL0 P3.1 / TCL1 P3.2 P3.3 P4.0 / BTCO P4.1–P4.3 P5.0–P5.3 P6.0 / KS0–P6.3 / KS3 P7.0 / KS4–P7.3 / KS7 P8.0–P8.3 P9.0–P9.2	Input mode: Connect to V_{DD} Output mode: No connection
DTMF	No connection
NC	Connect to V_{SS}

NOTE: P1.1, P1.2, P1.3, P3.2, P3.3, P8, P9, INT1, INT2, and INT4 are not used in the S3C7534/C7538.

9 RESET

OVERVIEW

When a RESET signal is input during normal operation or power-down mode, a hardware reset operation is initiated and the CPU enters idle mode. Then, when the standard oscillation stabilization interval of 36.6 ms at 3.58 MHz has elapsed, normal system operation resumes.

Regardless of when the RESET occurs — during normal operating mode or during a power-down mode — most hardware register values are set to the reset values described in Table 9–1 below. The current status of several register values is, however, always retained when a RESET occurs during idle or stop mode; If a RESET occurs during normal operating mode, their values are undefined. Current values that are retained in this case are as follows:

— Carry flag

— General-purpose registers E, A, L, H, X, W, Z, and Y

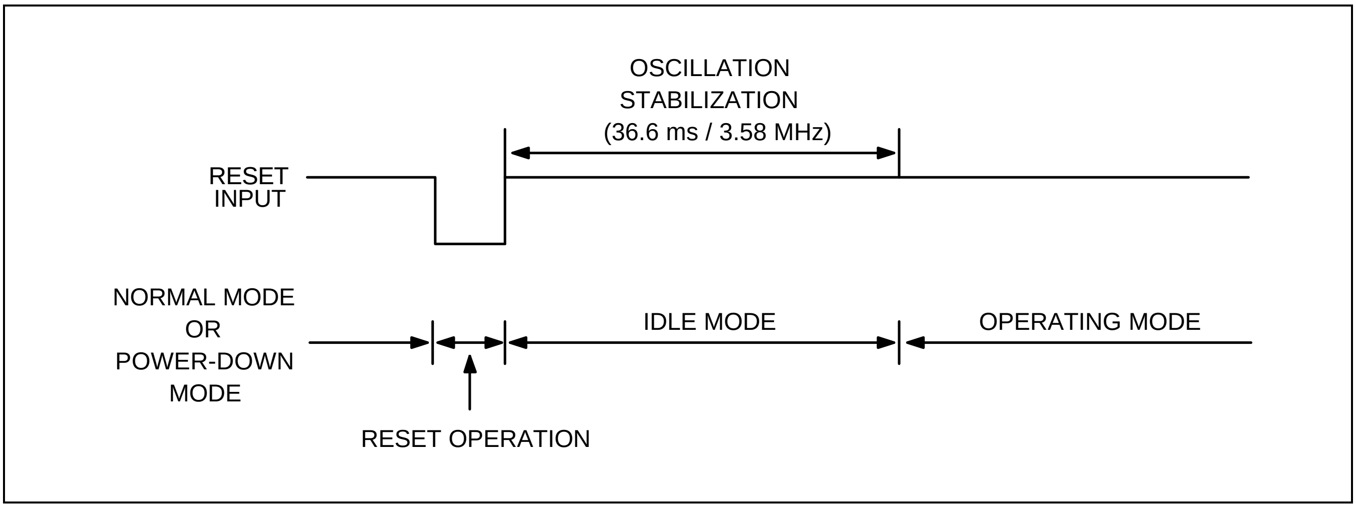


Figure 9-1. Timing for Oscillation Stabilization After RESET

HARDWARE RESET VALUES AFTER RESET

Table 9–1 gives you detailed information about hardware register values after a RESET occurs during power-down mode or during normal operation.

Table 9-1. Hardware Register Values After RESET

Hardware Component or Subcomponent	If RESET Occurs During Power-Down Mode	If RESET Occurs During Normal Operation
Program counter (PC)	Lower five bits of address 0000H are transferred to PC12–8, and the contents of 0001H to PC7–0.	Lower five bits of address 0000H are transferred to PC12–8, and the contents of 0001H to PC7–0.
Program Status Word (PSW):		
Carry flag (C)	Values retained	Undefined
Skip flag (SC0–SC2)	0	0
Interrupt status flags (IS0, IS1)	0	0
Bank enable flags (EMB, ERB)	Bit 6 of address 0000H in program memory is transferred to the ERB flag, and bit 7 of the address to the EMB flag.	Bit 6 of address 0000H in program memory is transferred to the ERB flag, and bit 7 of the address to the EMB flag.
Stack pointer (SP)	Undefined	Undefined
Data Memory (RAM):		
General registers E, A, L, H, X, W, Z, Y	Values retained	Undefined
General-purpose registers	Values retained (1)	Undefined
Bank selection registers (SMB, SRB)	0, 0	0, 0
BSC register (BSC0–BSC)	0	0
Clocks:		
Power control register (PCON)	0	0
Clock output mode register (CLMOD)	0	0
Interrupts:		
Interrupt request flags (IRQx)	0	0
Interrupt enable flags (IEx)	0	0
Interrupt priority flag (IPR)	0	0
Interrupt master enable flag (IME)	0	0
INT0 mode register (IMOD0)	0	0
INT1 mode register (IMOD1) (2)	0	0
INT2 mode register (IMOD2)	0	0

NOTES

1. The value of the 0F8H–0FDH are not retained when a RESET signal is input.
2. S3C7534/C7538 does not use the IMOD1.

Table 9-1. Hardware Register Values After RESET (Continued)

Hardware Component or Subcomponent	If RESET Occurs During Power-Down Mode	If RESET Occurs During Normal Operation
I/O Ports:		
Output buffers	Off	Off
Output latches	0	0
Port mode flags (PMG)	0	0
Pull-up resistor mode reg (PUMOD1/2)	0	0
Port open-drain enable register (PNE1)	0	0
Basic Timer:		
Count register (BCNT)	Undefined	Undefined
Mode register (BMOD)	0	0
Output enable flag (BOE)	0	0
Timer/Counters 0 and 1:		
Count registers (TCNT0/1)	0	0
Reference registers (TREF0/1)	FFH/FFH	FFH/FFH
Mode registers (TMOD0/1)	0	0
T/C output enable flags (TOE0/1)	0	0
T/C output latch (TOL0/1)	0	0
Watch Timer:		
Watch timer mode register (WMOD)	0	0
Watchdog Timer		
WDT mode register (WDMOD)	A5H	A5H
WDT clear flag (WDTCF)	0	0
DTMF Generator:		
DTMF mode register (DTMR)	0	0

10 I/O PORTS

OVERVIEW

The S3C7524/C7528 has one input port and eight I/O ports. The S3C7534/C7538 has one input port and six I/O ports. Pin addresses for all I/O ports are mapped in bank 15 of the RAM. The contents of I/O port pin latches can be read, written, or tested at the corresponding address using bit manipulation instructions.

The S3C7524/C7528 has four input pins and 31 configurable I/O pins for a maximum number of 35 I/O pins. The S3C7534/C7538 has one input pin and 22 configurable I/O pins for a maximum number of 23 I/O pins.

Port Mode Flags

Port mode flags (PM) are used to configure I/O ports 2 and 3 (port mode group 1), ports 4 and 5 (port mode group 2), ports 6 and 7 (port mode group 3), and port 8 and 9 (port mode group 4, S3C7524/C7528 only) to input or output mode by setting or clearing the corresponding I/O buffer. PMG flags are grouped in four 8-bit registers, and are addressable by 8-bit write instructions only.

PUMOD Control Register

The pull-up mode registers, PUMOD1 and 2 are 8-bit and 4-bit registers, respectively, used to assign internal pull-up resistors by software to specific I/O ports.

When configurable I/O ports 2 through 9 serves as an output pin, its assigned pull-up resistor is automatically disabled, even though the pin's pull-up resistor is enabled by a corresponding bit setting in the pull-up resistor mode register (PUMOD).

PUMOD1 is addressable by 8-bit write instructions only, PUMOD2 is addressable by 4-bit write instructions only. RESET clears PUMOD register values to logic zero, automatically disconnecting all software-assignable port pull-up resistors.

Table 10-1. I/O Port Overview

Port	I/O	Pins	Pin Names	Address	Function Description
1	I	4	P1.0 P1.1–P1.3 (note)	FF1H	4-bit input port. 1-bit and 4-bit read and test is possible. 1-bit pull-up resistors are software assignable
2, 3	I/O	8	P2.0–P2.3 P3.0–P3.1 P3.2–P3.3 (note)	FF2H FF3H	4-bit I/O ports. 1-bit and 4-bit read/write/test is possible. Ports 2 and 3 pins are individually software configurable as input or output. 4-bit Pull-up resistors are software assignable; pull-up registers are automatically disabled for output pins. Ports 2 and 3 can be paired for 8-bit data transfer.

NOTE: S3C7534/C7538 does not use the P1.1–P1.3, and P3.2–P3.3 pins.

Table 10-1. I/O Port Overview (Continued)

Port	I/O	Pins	Pin Names	Address	Function Description
4, 5	I/O	8	P4.0–P4.3 P5.0–P5.3	FF4H FF5H	4-bit I/O ports. 1-bit and 4-bit read/write/test is possible. Port 4 and 5 pins are individually software configurable as input or output. 4-bit pull-up resistors are software assignable; pull-up registers are automatically disable for output pins. N-Ch open drain or push-pull output may be selected by software. Ports 4 and 5 can be paired for 8-bit data transfer.
6, 7	I/O	8	P6.0–P6.3 P7.0–P7.3	FF6H FF7H	4-bit I/O ports. 1-bit and 4-bit read/write/test is possible. Port 6 and 7 pins are individually software configurable as input or output. 4-bit pull-up resistors are software assignable; pull-up registers are automatically disabled for output pins. Ports 6 and 7 can be paired for 8-bit data transfer.
8, 9 (note)	I/O	8	P8.0–P8.3 P9.0–P9.2	FF8H FF9H	4-bit I/O port. 1-bit and 4-bit read/write and test is possible. Ports 8 and 9 pins are individually software configurable as input or output. 4-bit pull-up resistors are software assignable; pull-up registers are automatically disable for output pins. Ports 8 and 9 can be paired for 8-bit data transfer.

NOTE: S3C7534/C7538 does not use the 8, 9 ports.

Table 10-2. Port Pin Status During Instruction Execution

Instruction Type	Example	Input Mode Status	Output Mode Status

1-bit test
1-bit input
4-bit input
8-bit input
	BTST P2.3
	LDB C,P1.0
	LD A,P7
	LD EA,P4
		Input or test data at each pin	Input or test data at output latch

1-bit output	BITR P2.3	Output latch contents undefined	Output pin status is modified
4-bit output 8-bit output	LD P2,A LD P6,EA	Transfer accumulator data to the output latch	Transfer accumulator data to the output pin

PORT MODE FLAGS (PM FLAGS)

Port mode flags (PM) are used to configure I/O ports 2–9 to input or output mode by setting or clearing the corresponding I/O buffer.

For convenient program reference, PM flags are organized into four groups — PMG1, PMG2, PMG3, and PMG4 as shown in Table 10-3. PM flags are addressable by 8-bit write instructions only.

When a PM flag is "0", the port is set to input mode; when it is "1", the port is enabled for output. RESET clears all port mode flags to logic zero, automatically configuring the corresponding I/O ports to input mode.

Table 10-3. Port Mode Group Flags

PM Group ID	Address	Bit 3	Bit 2	Bit 1	Bit 0
PMG1	FE8H	PM2.3	PM2.2	PM2.1	PM2.0
	FE9H	PM3.3 (2)	PM3.2 (2)	PM3.1	PM3.0
PMG2	FEAH	PM4.3	PM4.2	PM4.1	PM4.0
	FEBH	PM5.3	PM5.2	PM5.1	PM5.0
PMG3	FECH	PM6.3	PM6.2	PM6.1	PM6.0
	FEDH	PM7.3	PM7.2	PM7.1	PM7.0
PMG4 (2)	FEEH	PM8.3	PM8.2	PM8.1	PM8.0
	FEFH	"0"	PM9.2	PM9.1	PM9.0

NOTES
1. If bit = "0", the corresponding I/O pin is set to input mode. If bit = "1", the pin is set to output mode: PM4 for port 4 and so on. All flags are cleared to "0" following RESET.
2. S3C7534/C7538 does not use the PM3.2, PM3.3, and PMG4.

☞ PROGRAMMING TIP — Configuring I/O Ports to Input or Output

Configure P2.3 and P3 as an output port and the other ports as input ports:

```
BITS    EMB
SMB     15
LD      EA,#0F8H
LD      PMG1,EA      ;   P2.3, P3 ← Output, P2.0–P2.2 ← Input
LD      EA,#00H
LD      PMG2,EA      ;   P4, P5 ← Input,
LD      PMG3,EA      ;   P6, P7 ← Input
LD      PMG4,EA      ;   P8, P9 ← Input
```

NOTE: S3C7534/C7538 does not use the P1.1–P1.3, P3.2–P3.3, and P8.9.

PULL-UP RESISTOR MODE REGISTER (PUMOD)

The pull-up resistor mode registers (PUMOD1 and 2) are 8-bit registers used to assign internal pull-up resistors by software to specific I/O ports.

When configurable I/O ports 2 through 9 are used as an output pin, its assigned pull-up resistor is automatically disabled, even though the pin's pull-up is enabled by a corresponding PUMOD bit setting.

PUMOD1 is addressable by 8-bit write instructions only. PUMOD2 is addressable by 4bit write instructions only. RESET clears PUMOD register values to logic zero, automatically disconnecting all software-assignable port pull-up resistors.

Table 10-4. Pull-Up Resistor Mode Register (PUMOD) Organization

PUMOD ID	Address	Bit 3	Bit 2	Bit 1	Bit 0
PUMOD1	FDCH	PUR1.3	PUR1.2	PUR1.1	PUR1.0
	FDDH	PUR5	PUR4	PUR3	PUR2
PUMOD2	FDEH	PUR9 (2)	PUR8 (2)	PUR7	PUR6

NOTES
1. When bit = "1", pull-up resistors are assigned to the corresponding I/O port: PUR3 for port 3, PUR2 for port 2, and so on.
2. S3C7534/C7538 does not use the PUR1.1–PUR1.3, PUR8, and PUR9.

☞ PROGRAMMING TIP — Enabling and Disabling I/O Port Pull-Up Resistors

P2–P5 enable pull-up resistors, P1 disable pull-up resistors.

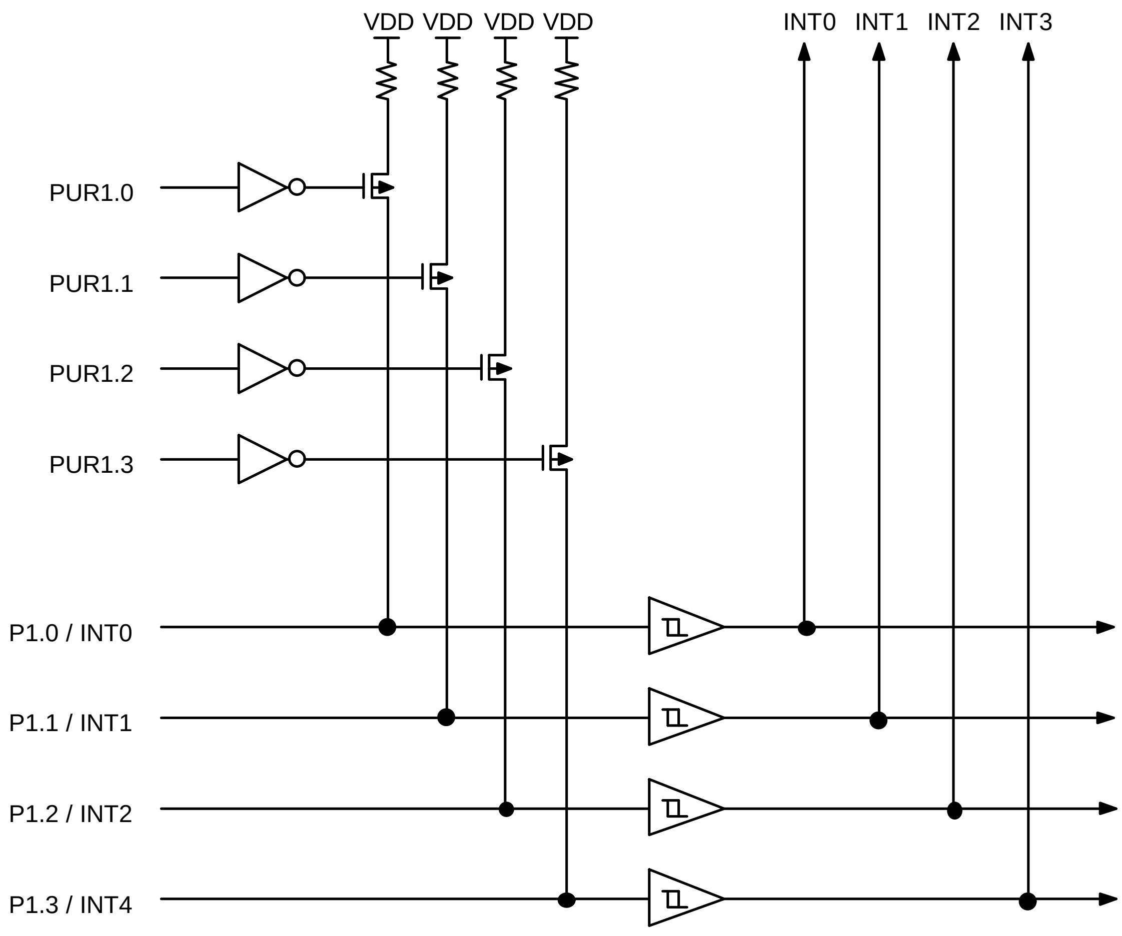
```
BITS    EMB
SMB     15
LD      EA,#0F0H
LD      PUMOD1,EA         ;    P2–P5 enable
```

N-CHANNEL OPEN-DRAIN MODE REGISTER (PNE)

The n-channel, open-drain mode register (PNE) is used to configure ports 4 and 5 to n-channel open-drain or as push-pull outputs. When a bit in the PNE register is set to "1", the corresponding output pin is configured to n-channel open-drain; when set to "0", the output pin is configured to push-pull. The PNE register consists of an 8-bit register; PNE1 can be addressed by 8-bit write instructions only.

FDAH	PNE4.3	PNE4.2	PNE4.1	PNE4.0	PNE1
FDBH	PNE5.3	PNE5.2	PNE5.1	PNE5.0	

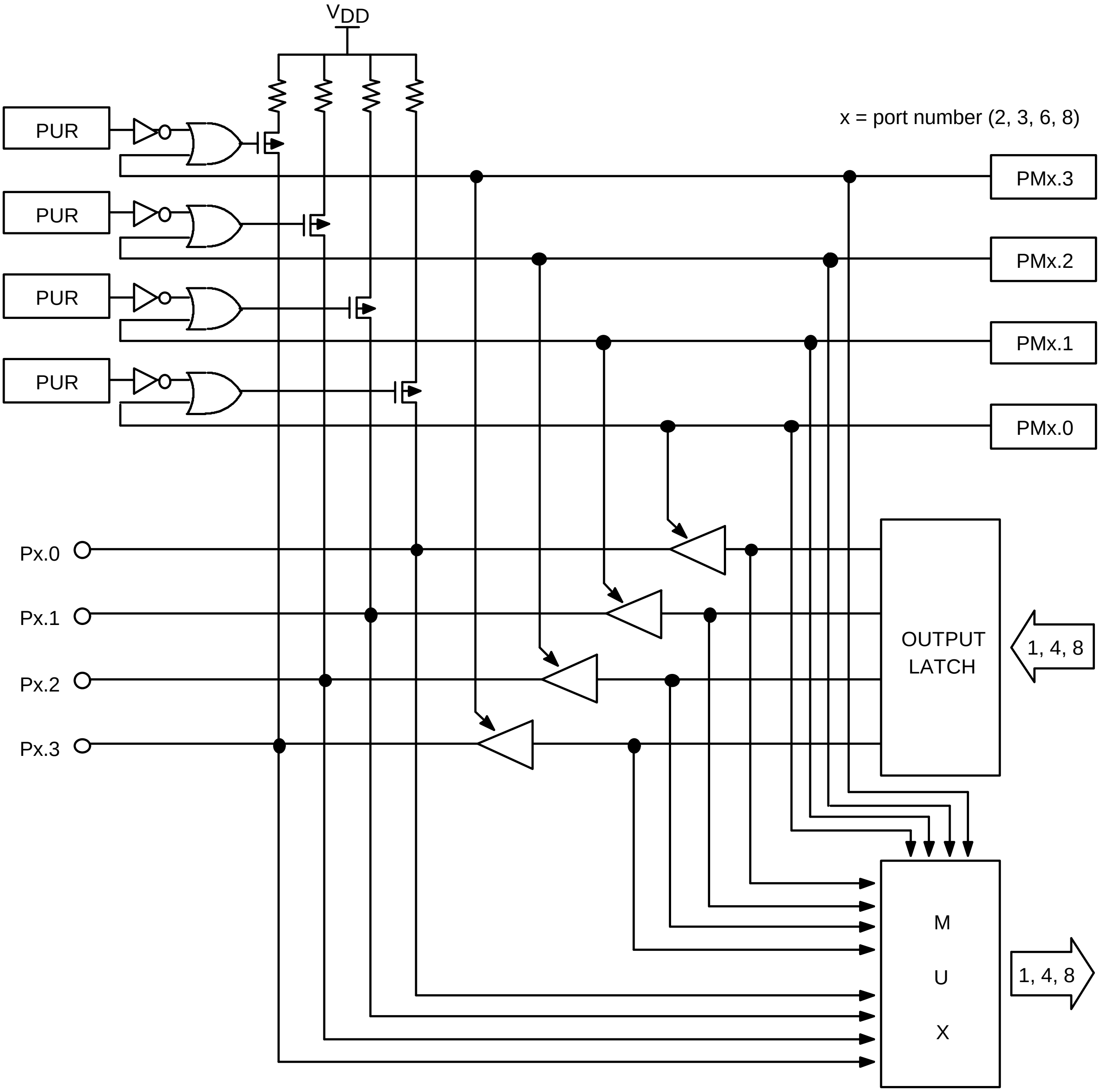
PORT 1 CIRCUIT DIAGRAM



NOTE: S3C7534/C7538 does not use the P1.1-P1.3.

Figure 10-1. Port 1 Circuit Diagram

PORT 2, 3, 6, 7, 8, and 9 CIRCUIT DIAGRAM



NOTES

1. When a port pin acts as an output, its pull-up resistor is automatically disabled, even though the port's pull-up resistor is enabled by bit settings to the pull-up resistor mode register (PUMOD).
2. S3C7534/C7538 does not use the P3.2, P3.3, P8, and P9.

Figure 10-2. Port 2, 3, 6, 7, 8, and 9 Circuit Diagram

PORT 4, 5 CIRCUIT DIAGRAM

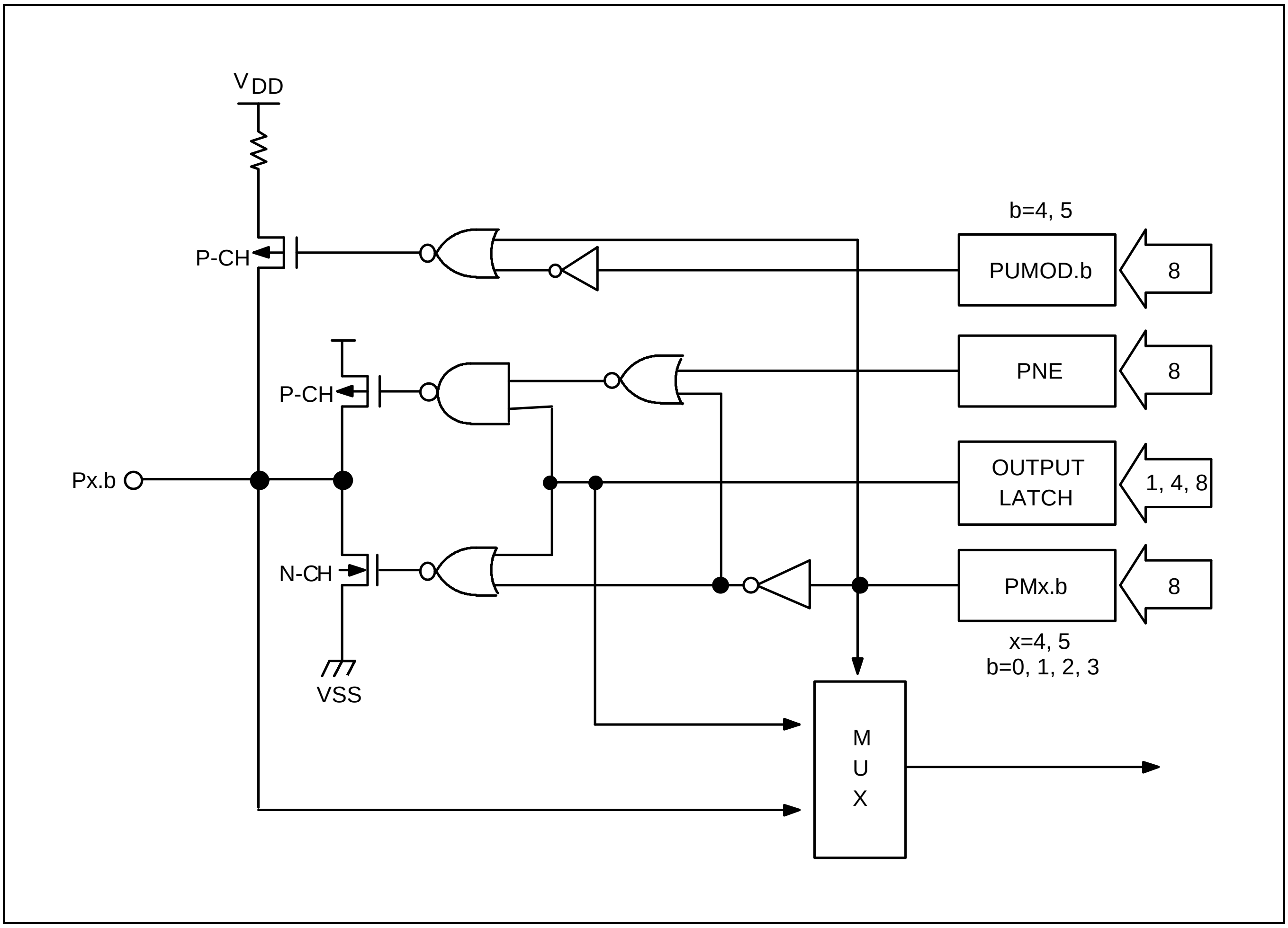


Figure 10-3. Port 4 and 5 Circuit Diagram

11 TIMERS and TIMER/COUNTERS

OVERVIEW

The S3C7524/C7528/C7534/C7538 microcontroller has four timer and timer/counter modules:

— 8-bit basic timer (BT)

— 8-bit timer/counters (TC0, TC1)

— Watch timer (WT)

The 8-bit basic timer (BT) is the microcontroller's main interval timer. It generates an interrupt request at a fixed time interval when the appropriate modification is made to its mode register. When the contents of the basic timer counter register BCNT overflows, a pulse is output to the basic timer output pin, BTCO. The basic timer also functions as a 'watchdog' timer and is used to determine clock oscillation stabilization time when stop mode is released by an interrupt and after a RESET.

The 8-bit timer/counters (TC0, TC1) are programmable timer/counters that are used primarily for event counting and for clock frequency modification and output.

The watch timer (WT) module consists of an 8-bit watch timer mode register, a clock selector, and a frequency divider circuit. Watch timer functions include real-time and watch-time measurement, system clock interval timing, buzzer output generation.

BASIC TIMER (BT)

OVERVIEW

The 8-bit basic timer (BT) has six functional components:

— Clock selector logic

— 4-bit mode register (BMOD)

— 8-bit counter register (BCNT)

— Output enable flag (BOE)

— 8-bit watchdog timer mode register (WDMOD)

— Watchdog timer counter clear flag (WDTCF)

The basic timer generates interrupt requests at precise intervals, based on the frequency of the system clock. Timer pulses are output from the basic timer's counter register BCNT to the output pin BTCO when an overflow occurs in the counter register BCNT. You can use the basic timer as a "watchdog" timer for monitoring system events or use BT output to stabilize clock oscillation when stop mode is released by an interrupt and following RESET. Bit settings in the basic timer mode register BMOD turns the BT module on and off, selects the input clock frequency, and controls interrupt or stabilization intervals.

Interval Timer Function

The basic timer's primary function is to measure elapsed time intervals. The standard time interval is equal to 256 basic timer clock pulses.

To restart the basic timer, one bit setting is required: bit 3 of the mode register BMOD should be set to logic one. The input clock frequency and the interrupt and stabilization interval are selected by loading the appropriate bit values to BMOD.2–BMOD.0.

The 8-bit counter register, BCNT, is incremented each time a clock signal is detected that corresponds to the frequency selected by BMOD. BCNT continues incrementing as it counts BT clocks until an overflow occurs (≥ 255). An overflow causes the BT interrupt request flag (IRQB) to be set to logic one to signal that the designated time interval has elapsed. An interrupt request is than generated, BCNT is cleared to logic zero, and counting continues from 00H.

Watchdog Timer Function

The basic timer can also be used as a "watchdog" timer to signal the occurrence of system or program operation error. For this purpose, instruction that clear the watchdog timer (BITS WDTCF) should be executed at proper points in a program within given period. If an instruction that clears the watchdog timer is not executed within the given period and the watchdog timer overflows, reset signal is generated and the system restarts with reset status. An operation of watchdog timer is as follows:

— Write some values (except #5AH) to watchdog timer mode register, WDMOD

— If WDCNT overflows, system reset is generated.

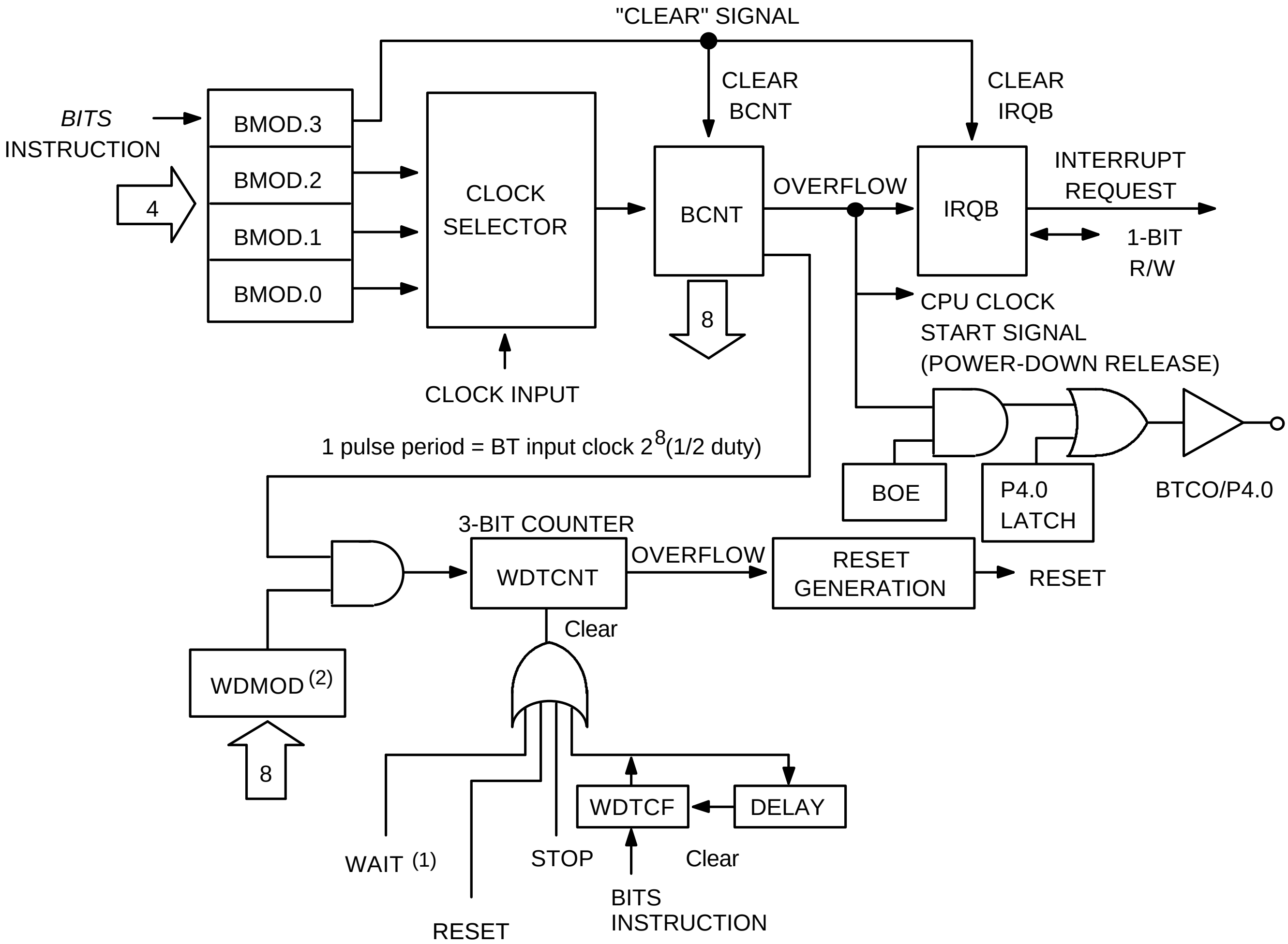
Oscillation Stabilization Interval Control

Bits 2–0 of the BMOD register are used to select the input clock frequency for the basic timer. This setting also determines the time interval (also referred to as 'wait time') required to stabilize clock signal oscillation when stop mode is released by an interrupt. When a RESET signal is inputted, the standard stabilization interval for system clock oscillation following the RESET is 31.3 ms at 4.19 MHz.

Table 11-1. Basic Timer Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
BMOD	Control	Controls the clock frequency (mode) of the basic timer; also, the oscillation stabilization interval after stop mode release or RESET	4-bit	F85H	4-bit write-only; BMOD.3: 1-bit writeable	"0"
BCNT	Counter	Counts clock pulses matching the BMOD frequency setting	8-bit	F86H–F87H	8-bit read-only	U (note)
BOE	Flag	Controls output of basic timer output latch to the BTCO pin	1-bit	F92H.1	1-, 4-bit read/write	"0"
WDMOD	Control	Controls watchdog timer operation.	8-bit	F98H–F99H	8-bit write-only	A5H
WDTCF	Control	Clears the watchdog timer's counter.	1-bit	F9AH.3	1-, 4-bit write-only	"0"

NOTE: 'U' means the value is undetermined after a RESET.



NOTES:

1. WAIT means the stabilization time after RESET or the stabilization time after STOP mode release.
2. The RESET signal can be generated, if the WDMOD is toggled eignt times. “Toggle” means to change the value of WDMOD from 5AH to another value, or vice versa.

Figure 11-1. Basic Timer Circuit Diagram

BASIC TIMER MODE REGISTER (BMOD)

The basic timer mode register, BMOD, is a 4-bit write-only register. Bit 3, the basic timer start control bit, is also 1-bit addressable. All BMOD values are set to logic zero following RESET and interrupt request signal generation is set to the longest interval. (BT counter operation cannot be stopped.) BMOD settings have the following effects:

— Restart the basic timer;

— Control the frequency of clock signal input to the basic timer;

— Determine time interval required for clock oscillation to stabilize following the release of stop mode by an interrupt.

By loading different values into the BMOD register, you can dynamically modify the basic timer clock frequency during program execution. Four BT frequencies, ranging from $fx/2^{12}$ to $fx/2^{5}$, are selectable. Since BMOD's reset value is logic zero, the default clock frequency setting is $fx/2^{12}$.

The most significant bit of the BMOD register, BMOD.3, is used to restart the basic timer. When BMOD.3 is set to logic one by a 1-bit write instruction, the contents of the BT counter register (BCNT) and the BT interrupt request flag (IRQB) are both cleared to logic zero, and timer operation restarts.

The combination of bit settings in the remaining three registers — BMOD.2, BMOD.1, and BMOD.0 — determine the clock input frequency and oscillation stabilization interval.

Table 11-2. Basic Timer Mode Register (BMOD) Organization

BMOD.3	Basic Timer Start Control Bit
1	Start basic timer; clear IRQB, BCNT, and BMOD.3 to "0"

BMOD.2	BMOD.1	BMOD.0	Basic Timer Input Clock	Oscillation Stabilization
0	0	0	$fx/2^{12}$ (1.02 kHz)	$2^{20}/fx$ (250 ms)
0	1	1	$fx/2^{9}$ (8.18 kHz)	$2^{17}/fx$ (31.3 ms)
1	0	1	$fx/2^{7}$ (32.7 kHz)	$2^{15}/fx$ (7.82 ms)
1	1	1	$fx/2^{5}$ (131 kHz)	$2^{13}/fx$ (1.95 ms)

NOTES

1. Clock frequencies and oscillation stabilization assume a system oscillator clock frequency (fx) of 4.19 MHz.
2. fx = system clock frequency.
3. Oscillation stabilization time is the time required to stabilize clock signal oscillation after stop mode is released. The data in the table column 'Oscillation Stabilization' can also be interpreted as "Interrupt Interval Time."
4. The standard stabilization time for system clock oscillation following a RESET is 31.3 ms at 4.19 MHz.

BASIC TIMER COUNTER (BCNT)

BCNT is an 8-bit counter for the basic timer. It can be addressed by 8-bit read instructions. RESET leaves the BCNT counter value undetermined. BCNT is automatically cleared to logic zero whenever the BMOD register control bit (BMOD.3) is set to "1" to restart the basic timer. It is incremented each time a clock pulse of the frequency determined by the current BMOD bit settings is detected.

When BCNT has incrementing to hexadecimal ‘FFH’ (≥ 255 clock pulses), it is cleared to ‘00H’ and an overflow is generated. The overflow causes the interrupt request flag, IRQB, to be set to logic one. When the interrupt request is generated, BCNT immediately resumes counting incoming clock signals.

NOTE

Always execute a BCNT read operation twice to eliminate the possibility of reading unstable data while the counter is incrementing. If, after two consecutive reads, the BCNT values match, you can select the latter value as valid data. Until the results of the consecutive reads match, however, the read operation must be repeated until the validation condition is met.

BASIC TIMER OUTPUT ENABLE FLAG (BOE)

The basic timer output enable flag (BOE) enables and disables basic timer output to the BTCO pin at I/O port 4 (P4.0). When BOE is logic zero, basic timer output to the BTCO pin is disabled; when it is logic one, BT output to the BTCO pin is enabled. A RESET clears the BOE flag to "0", disabling basic timer output to the BTCO pin. When the BOE flag is set to "1" and the BCNT register overflows, the overflow signal is sent to the BTCO pin. BOE can be addressed by 1-bit read and write instructions.

	Bit 3	Bit 2	**Bit 1**	Bit 0
F92H	TOE1	TOE0	**BOE**	0

BASIC TIMER OPERATION SEQUENCE

The basic timer's sequence of operations may be summarized as follows:

1. Set BMOD.3 to logic one to restart the basic timer
2. BCNT is then incremented by one after each clock pulse corresponding to BMOD selection
3. BCNT overflows if BCNT = 255 (BCNT = FFH)
4. When an overflow occurs, the IRQB flag is set by hardware to logic one
5. The interrupt request is generated
6. BCNT is then cleared by hardware to logic zero
7. Basic timer resumes counting clock pulses

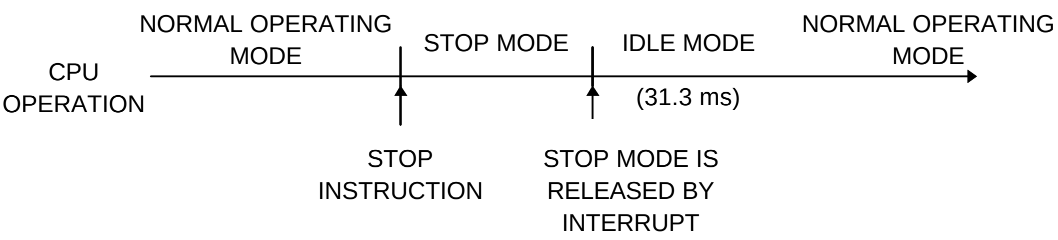
☞ PROGRAMMING TIP — Using the Basic Timer

1. To read the basic timer count register (BCNT):

```
        BITS    EMB
        SMB     15
BCNTR   LD      EA,BCNT
        LD      YZ,EA
        LD      EA,BCNT
        CPSE    EA,YZ
        JR      BCNTR
```

2. When stop mode is released by an interrupt, set the oscillation stabilization interval to 31.3 ms:

```
        BITS    EMB
        SMB     15
        LD      A,#0BH
        LD      BMOD,A          ;   Wait time is 31.3 ms
        STOP                    ;   Set stop power-down mode
        NOP
        NOP
        NOP
```



3. To set the basic timer interrupt interval time to 1.95 ms (at 4.19 MHz):

```
        BITS    EMB
        SMB     15
        LD      A,#0FH
        LD      BMOD,A
        EI
        BITS    IEB             ;   Basic timer interrupt enable flag is set to "1"
```

4. Clear BCNT and the IRQB flag and restart the basic timer:

```
        BITS    EMB
        SMB     15
        BITS    BMOD.3
```

WATCHDOG TIMER MODE REGISTER (WDMOD)

The watchdog timer mode register, WDMOD, is a 8-bit write-only register. WDMOD register controls to enable or disable the watchdog function. WDMOD values are set to logic "A5H" following RESET and this value enables the watchdog timer. Watchdog timer is set to the longest interval because BT overflow signal is generated with the longest interval.

WDMOD	Watchdog Timer Enable/Disable Control
5AH	Disable watchdog timer function
Any other value	Enable watchdog timer function

WATCHDOG TIMER COUNTER (WDCNT)

The watchdog timer counter, WDCNT, is a 3-bit counter. WDCNT is automatically cleared to logic zero, and restarts whenever the WDTCF register control bit is set to "1". RESET, stop, and wait signal clears the WDCNT to logic zero also.

WDCNT increments each time a clock pulse of the overflow frequency determined by the current BMOD bit setting is generated. When WDCNT has incremented to hexadecimal '07H', it is cleared to '00H' and an overflow is generated. The overflow causes the system RESET. When the interrupt request is generated, BCNT immediately resumes counting incoming clock signals.

WATCHDOG TIMER COUNTER CLEAR FLAG (WDTCF)

The watchdog timer counter clear flag, WDTCF, is a 1-bit write instruction. When WDTCF is set to one, it clears the WDCNT to zero and restarts the WDCNT. WDTCF register bits 2–0 are always logic zero.

Table 11-3. Watchdog Timer Interval Time

BMOD	BT Input Clock	WDCNT Input Clock	WDT Interval Time	Main Clock
x000b	2^{12}/fx	2^{12}/fx $\times$ 2^8	2^{12}/fx $\times$ 2^8 $\times$ 2^3	2 second
x011b	2^9/fx	2^9/fx $\times$ 2^8	2^9/fx $\times$ 2^8 $\times$ 2^3	250 ms
x101b	2^7/fx	2^7/fx $\times$ 2^8	2^7/fx $\times$ 2^8 $\times$ 2^3	62.5 ms
x111b	2^5/fx	2^5/fx $\times$ 2^8	2^5/fx $\times$ 2^8 $\times$ 2^3	15.6 ms

NOTES
1. Clock frequencies assume a system oscillator clock frequency (fx) of 4.19 MHz
2. fx = system clock frequency.

☞ PROGRAMMING TIP — Using the Watchdog Timer

```
RESET     DI
          LD        EA,#00H
          LD        SP,EA
                •
                •
                •
          LD        A,#0DH          ;  WDCNT input clock is 7.82 ms
          LD        BMOD,A
                •
                •
                •
MAIN      BITS      WDTCF           ;  Main routine operation period must be shorter than watchdog
                •                   ;  timer's period
                •
                •
          JP        MAIN
```

8-BIT TIMER/COUNTERS 0 AND 1 (TC0, TC1)

OVERVIEW

The S3C7524/C7528/C7534/C7538 TC0 and TC1 are identical except that they have different counter clock sources, which are controlled by the TMODn register. Timer/counters 0 and 1 (TC0, TC1) are used to count system 'events' by identifying the transition (high-to-low or low-to-high) of incoming square wave signals. To indicate that an event has occurred, or that a specified time interval has elapsed, TC generates an interrupt request. By counting signal transitions and comparing the current counter value with the reference register value, TC can be used to measure specific time intervals.

TC has a reloadable counter that consists of two parts: an 8-bit reference register, TREFn (n = 0, 1) into which you write the counter reference value, and an 8-bit counter register ,TCNTn (n = 0, 1) whose value is automatically incremented by counter logic.

8-bit mode register, TMODn (n = 0, 1), is used to activate the timer/counter and to select the basic clock frequency to be used for timer/counter operations. To dynamically modify the basic frequency, new values can be loaded into the TMODn register during program execution.

TC FUNCTION SUMMARY

8-bit programmable timer	Generates interrupts at specific time intervals based on the selected clock frequency.
External event counter	Counts various system "events" based on edge detection of external clock signals at the TC input pin, TCLn (n = 0, 1).
Arbitrary frequency output	Outputs clock frequencies to the TC output pin, TCLOn (n = 0, 1).
External signal divider	Divides the frequency of an incoming external clock signal according to a modifiable reference value (TREFn), and outputs the modified frequency to the TCLOn pin.

TC COMPONENT SUMMARY

Mode register (TMODn)	Activates the timer/counter and selects the internal clock frequency or the external clock source at the TCLn pin.
Reference register (TREFn)	Stores the reference value for the desired number of clock pulses between interrupt requests.
Counter register (TCNTn)	Counts internal or external clock pulses based on the bit settings in TMODn and TREFn.
Clock selector circuit	Together with the mode register (TMODn), lets you select one of four internal clock frequencies or an external clock.
8-bit comparator	Determines when to generate an interrupt by comparing the current value of the counter register (TCNTn) with the reference value previously programmed into the reference register (TREFn).
Output latch (TOLn)	When the contents of the TCNTn and TREFn registers coincide, the timer/counter interrupt request flag (IRQTn) is set to "1", the status of TOLn is inverted, and an interrupt is generated.
Output enable flag (TOEn)	Must be set to logic one before the contents of the TOLn latch can be output to TCLOn.
Interrupt request flag (IRQTn)	Cleared when TC operation starts and the TC interrupt service routine is executed and set to one whenever the counter value and reference value coincide.
Interrupt enable flag (IETn)	Must be set to logic one before the interrupt requests generated by timer/counters can be processed.

Table 11-4. TC Register Overview

Register Name	Type	Description	Size	RAM Address	Addressing Mode	Reset Value
TMOD0 TMOD1	Control	Controls TC0 and TC1 enable/disable (bit 2); clears and resumes counting operation (bit 3); sets input clock and clock frequency (bits 6–4)	8-bit	F90H–F91H FA0H–FA1H	8-bit write only; (TMODn.3 is also 1-bit writeable)	"0"
TCNT0 TCNT1	Counter	Counts clock pulses matching the TMODn frequency setting	8-bit	F94H–F95H FA4H–FA5H	8-bit read-only	"0"
TREF0 TREF1	Reference	Stores reference value for the timer/counters interval setting	8-bit	F96H–F97H FA8H–FA9H	8-bit write-only	FFH
TOE0 TOE1	Flag	Controls timer/counters output to the TCLOn pin	1-bit	F92H.2 F92H.3	1-, 4-bit read/write	"0"

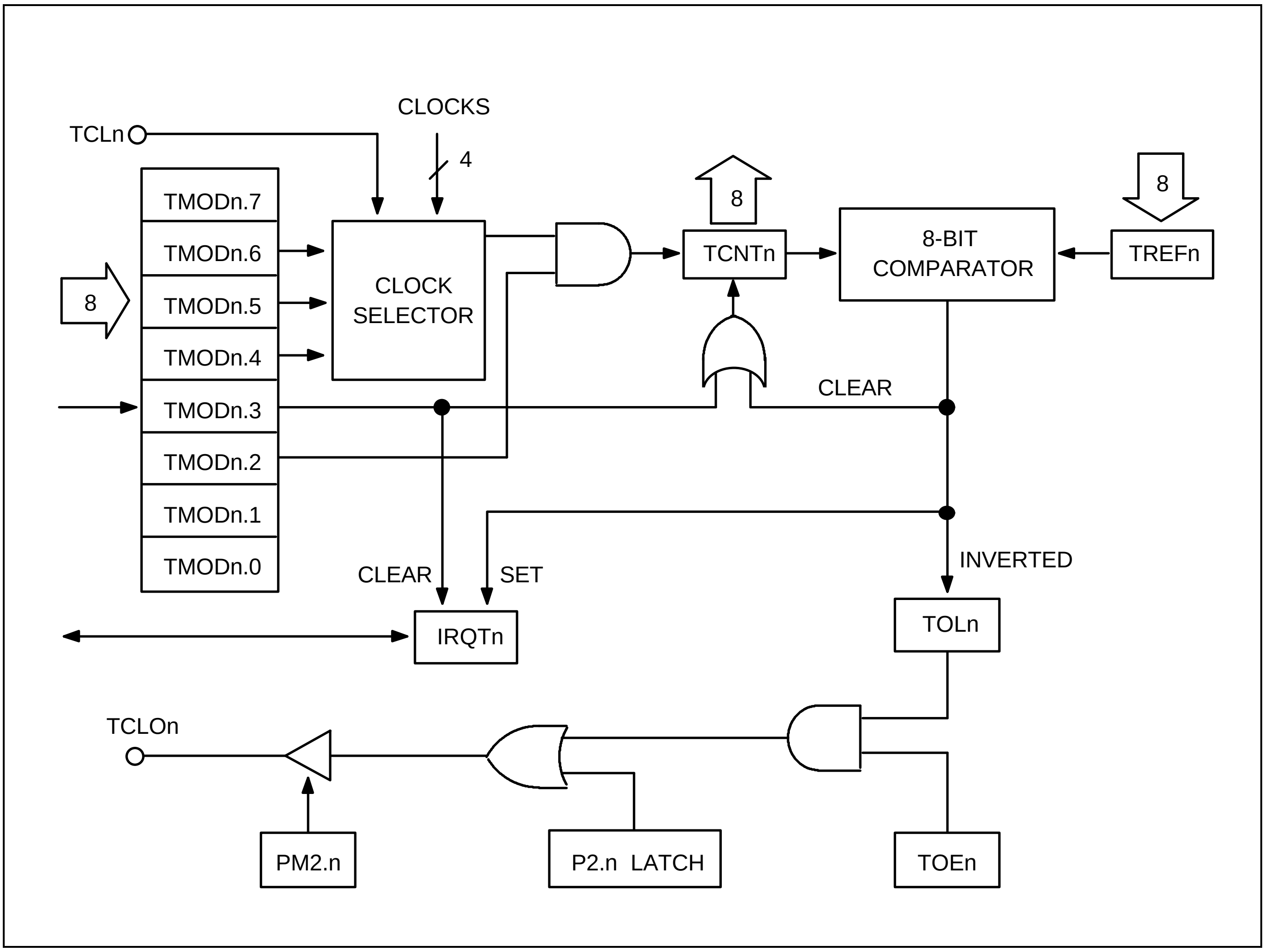


Figure 11-2. TC Circuit Diagram

TC ENABLE/DISABLE PROCEDURE

Enable Timer/Counter

— Set TMODn.2 to logic one

— Set the TC interrupt enable flag IETn to logic one

— Set TMODn.3 to logic one

TCNTn and IRQTn are cleared to logic zero, and timer/counter operation starts.

Disable Timer/Counter

— Set TMODn.2 to logic zero

Clock signal input to the counter register TCNTn is halted. The current TCNTn value is retained and can be read if necessary.

TC PROGRAMMABLE TIMER/COUNTER FUNCTION

Timer/counters can be programmed to generate interrupt requests at various intervals based on the selected system clock frequency. Its 8-bit TC mode register TMODn is used to activate the timer/counter and to select the clock frequency. The reference register TREFn stores the value for the number of clock pulses to be generated between interrupt requests. The counter register, TCNTn, counts the incoming clock pulses, which are compared to the TREFn value as TCNTn is incremented. When there is a match (TREFn = TCNTn), an interrupt request is generated.

To program timer/counter to generate interrupt requests at specific intervals, choose one of four internal clock frequencies (divisions of the system clock, fx) and load a counter reference value into the reference register. The count register is incremented each time an internal counter pulse is detected with the reference clock frequency specified by TMODn.4–TMODn.6 settings. To generate an interrupt request, the TC interrupt request flag (IRQTn) should be set to logic one, the status of TOLn is inverted, and the interrupt is generated. The content of the counter register is then cleared to 00H and TC continues counting. The interrupt request mechanism for TC includes an interrupt enable flag (IETn) and an interrupt request flag (IRQTn).

TC OPERATION SEQUENCE

The general sequence of operations for using TC can be summarized as follows:

1. Set TMODn.2 to "1" to enable TC0 and TC1
2. Set TMODn.6 to "1" to enable the system clock (fx) input
3. Set TMODn.5 and TMODn.4 bits to desired internal frequency ($fx/2^n$)
4. Load a value to TREFn to specify the interval between interrupt requests
5. Set the TC interrupt enable flag (IETn) to "1"
6. Set TMODn.3 bit to "1" to clear TCNTn and IRQTn, and start counting
7. TCNTn increments with each internal clock pulse
8. When the comparator shows TCNTn = TREFn, the IRQTn flag is set to "1"
9. Output latch (TOLn) logic toggles high or low
10. Interrupt request is generated
11. TCNTn is cleared to 00H and counting resumes
12. Programmable timer/counter operation continues until TMODn.2 is cleared to "0".

TC EVENT COUNTER FUNCTION

Timer/counters can monitor or detect system 'events' by using the external clock input at the TCLn pin as the counter source. The TC mode register selects rising or falling edge detection for incoming clock signals. The counter register is incremented each time the selected state transition of the external clock signal occurs.

With the exception of the different TMODn.4–TMODn.6 settings, the operation sequence for TC's event counter function is identical to its programmable timer/counter function. To activate the TC event counter function,

— Set TMODn.2 to "1" to enable TC;

— Clear TMODn.6 to "0" to select the external clock source at the TCLn pin;

— Select TCLn edge detection for rising or falling signal edges by loading the appropriate values to TMODn.5 and TMODn.4.

— P3.0 and P3.1 must be set to input mode.

Table 11-5. TMODn Settings for TCLn Edge Detection

TMODn.5	TMODn.4	TCLn Edge Detection
0	0	Rising edges
0	1	Falling edges

TC CLOCK FREQUENCY OUTPUT

Using timer/counters, a modifiable clock frequency can be output to the TC clock output pin, TCLOn. To select the clock frequency, load the appropriate values to the TC mode register, TMODn. The clock interval is selected by loading the desired reference value into the reference register TREFn. In summary, the operational sequence required to output a TC-generated clock signal to the TCLOn pin is as follows:

1. Load a reference value to TREFn.
2. Set the internal clock frequency in TMODn.
3. Initiate TCn clock output to TCLOn (TMODn.2 = "1").
4. Set port 2 mode flag (PM2.0 and PM 2.1) to "1".
5. Set P2.0 and P2.1 output latches to "0".
6. Set TOEn flag to "1".

Each time the contents of TCNTn and TREFn coincide and an interrupt request is generated, the state of the output latch TOLn is inverted and the TC-generated clock signal is output to the TCLOn pin.

☞ PROGRAMMING TIP — TC0 Signal Output to the TCLO0 Pin

Output a 30 ms pulse width signal to the TCLO0 pin:

```
BITS    EMB
SMB     15
LD      EA,#68H
LD      TREF0,EA
LD      EA,#4CH
LD      TMOD0,EA
LD      EA,#01H
LD      PMG1,EA         ;   P2.0 ← output mode
BITR    P2.0            ;   P2.0 clear
BITS    TOE0
```

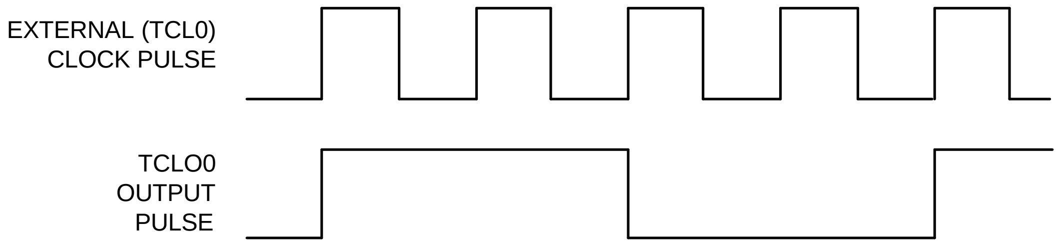
TC EXTERNAL INPUT SIGNAL DIVIDER

By selecting an external clock source and loading a reference value into the TC reference register, TREFn, you can divide the incoming clock signal by the TREFn value and then output this modified clock frequency to the TCLOn pin. The sequence of operations used to divide external clock input can be summarized as follows:

1. Load a signal divider value to the TREFn register
2. Clear TMODn.6 to "0" to enable external clock input at the TCLn pin
3. Set TMODn.5 and TMODn.4 to desired TCLn signal edge detection
4. Set port 2 mode flag (PM2.0, PM2.1) to output ("1")
5. Set P2.0 and P2.1 output latches to "0"
6. Set TOEn flag to "1" to enable output of the divided frequency to the TCLOn pin

☞ PROGRAMMING TIP — External TCL0 Clock Output to the TCLO0 Pin

Output external TCL0 clock pulse to the TCLO0 pin (divide by four):




```
BITS    EMB
SMB     15
LD      EA,#01H
LD      TREF0,EA
LD      EA,#0CH
LD      TMOD0,EA
LD      EA,#01H
LD      PMG1,EA         ;       P2.0 ← output mode
BITR    P2.0            ;       P2.0 clear
BITS    TOE0
```

TC MODE REGISTER (TMODn)

TMODn are the 8-bit mode control registers for timer/counter 0 and 1. They are addressable by 8-bit write instructions. One bit, TMODn.3, is also 1-bit writeable. RESET clears all TMODn bits to logic zero and disables TC operations.

F90H	TMOD0.3	TMOD0.2	"0"	"0"	TMOD0
F91H	"0"	TMOD0.6	TMOD0.5	TMOD0.4	

FA0H	TMOD1.3	TMOD1.2	"0"	"0"	TMOD1
FA1H	"0"	TMOD1.6	TMOD1.5	TMOD1.4	

TMODn.2 is the enable/disable bit for timer/counter 0 and 1. When TMODn.3 is set to "1", the contents of TCNTn and IRQTn are cleared, counting starts from 00H, and TMODn.3 is automatically reset to "0" for normal TC operation. When TC operation stops (TMODn.2 = "0"), the contents of the counter register TCNTn are retained until TC is re-enabled.

The TMODn.6, TMODn.5, and TMODn.4 bit settings are used together to select the TC clock source. This selection involves two variables:

— Synchronization of timer/counter operations with either the rising edge or the falling edge of the clock signal input at the TCLn pin, and

— Selection of one of four frequencies, based on division of the incoming system clock frequency, for use in internal TC operation.

Table 11-6. TC Mode Register (TMODn) Organization

Bit Name	Setting	Resulting TC0 Function	Address
TMODn.7	0	Always logic zero	F91H (TMOD0) FA1H (TMOD1)
TMODn.6 TMODn.5 TMODn.4	0,1	Specify input clock edge and internal frequency	
TMODn.3	1	Clear TCNTn and IRQTn. TOLn is remained and resume counting immediately (This bit is automatically cleared to logic zero immediately after counting resumes.)	F90H (TMOD0) FA0H (TMOD1)
TMODn.2	0	Disable timer/counter; retain TCNTn contents	
	1	Enable timer/counter	
TMODn.1	0	Always logic zero	
TMODn.0	0	Always logic zero	

Table 11-7. TMODn.6, TMODn.5, and TMODn.4 Bit Settings

TMODn.6	TMODn.5	TMODn.4	TC0 Counter Source	TC1 Counter Source
0	0	0	External clock input (TCL0) on rising edges	External clock input (TCL1) on rising edges
0	0	1	External clock input (TCL0) on falling edges	External clock input (TCL1) on falling edges
1	0	0	$fx/2^{10}$ (4.09 kHz)	$fx/2^{12}$ (1.02 kHz)
1	0	1	$fx/2^{6}$ (65.5 kHz)	$fx/2^{10}$ (4.09 kHz)
1	1	0	$fx/2^{4}$ (262 kHz)	$fx/2^{8}$ (16.4 kHz)
1	1	1	fx = 4.19 MHz	$fx/2^{6}$ (65.5 kHz)

NOTE: 'fx' = system clock of 4.19 MHz.

☞ PROGRAMMING TIP — Restarting TC0 Counting Operation

1. Set TC0 timer interval to 4.09 kHz:

```
BITS    EMB
SMB     15
LD      EA,#4CH
LD      TMOD0,EA
EI
BITS    IET0
```

2. Clear TCNT0 and IRQT0, TOL0 is remained and restart TC0 counting operation:

```
BITS    EMB
SMB     15
BITS    TMOD0.3
```

TC COUNTER REGISTER (TCNTn)

The 8-bit counter register for TC, TCNTn, is read-only and can be addressed by 8-bit RAM control instructions. RESET sets all counter register values to logic zero (00H).

Whenever TMODn.3 is enabled, TCNTn is cleared to logic zero and counting resumes. The TCNTn register value is incremented each time an incoming clock signal is detected that matches the signal edge and frequency setting of the TMODn register (specifically, TMODn.6–TMODn.4).

Each time TCNTn is incremented, the new value is compared to the reference value stored in the reference register, TREFn. When TCNTn = TREFn, an overflow occurs in the counter register, the interrupt request flag, IRQTn, is set to logic one, and an interrupt request is generated to indicate that the specified timer/counter interval has elapsed.

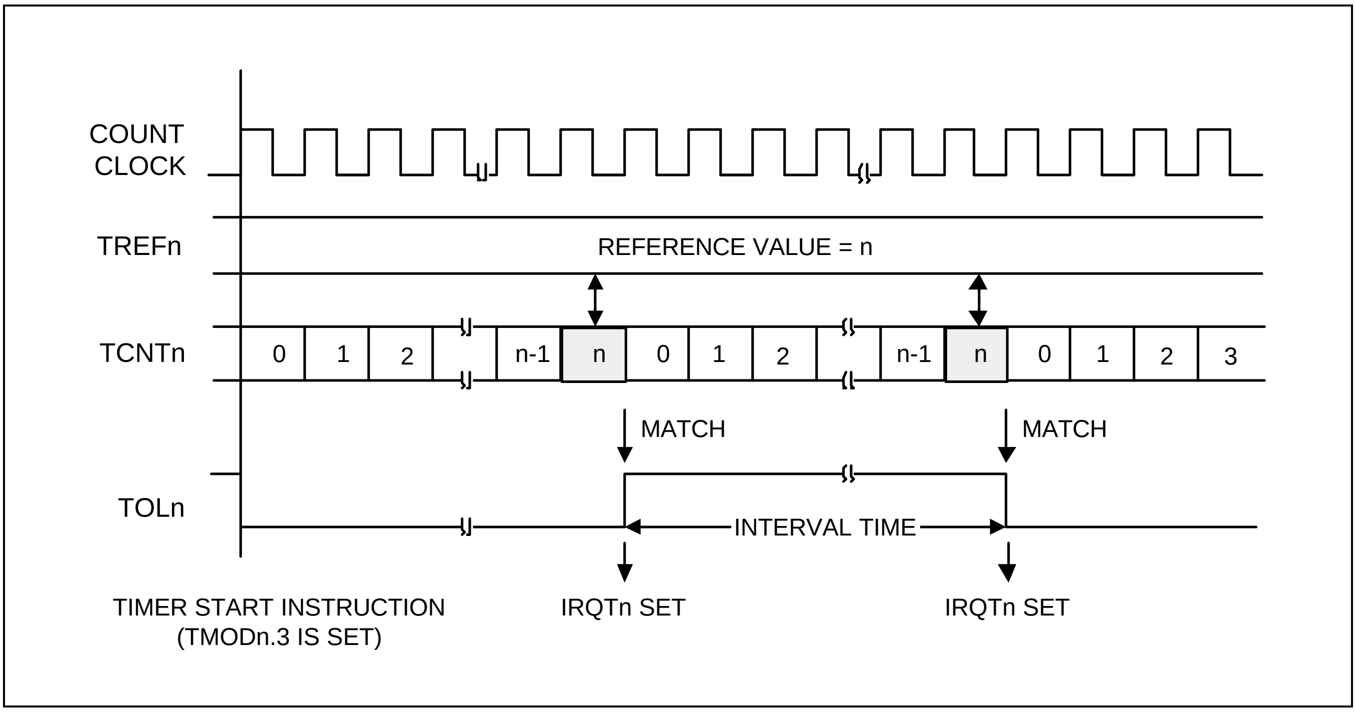


Figure 11-3. TC Timing Diagram

TC REFERENCE REGISTER (TREFn)

The TC reference register, TREFn, is an 8-bit write-only register. RESET initializes the TREFn value to 'FFH'.

TREFn is used to store a reference value to be compared to the incrementing TCNTn register in order to identify an elapsed time interval. Reference values will differ depending upon the specific function that TC is being used to perform — as a programmable timer/counter, event counter, clock signal divider, or arbitrary frequency output source.

During timer/counter operation, the value loaded into the reference register is compared to the counter value. When TCNTn = TREFn, the TC output latch (TOLn) is inverted and an interrupt request is generated to signal the interval or event. The TREFn value, together with the TMODn clock frequency selection, determines the specific TC timer interval. Use the following formula to calculate the correct value to load to the TREFn reference register:

$$\text{TC timer interval} = (\text{TREFn value} + 1) \times \frac{1}{\text{TMODn frequency setting}}$$

$$(\text{ assuming a TREFn value } \neq 0)$$

TC OUTPUT ENABLE FLAG (TOEn)

The 1-bit timer/counter output enable flag TOEn controls output from timer/counter to the TCLOn pin. TOEn is addressable by 1-bit read and write instructions.

	Bit 3	Bit 2	Bit 1	Bit 0
F92H	TOE1	TOE0	BOE	0

When you set the TOEn flag to "1", the contents of TOLn can be output to the TCLOn pin. Whenever a RESET occurs, TOEn is automatically set to logic zero, disabling all TC output.

TC OUTPUT LATCH (TOLn)

TOLn is the output latch for timer/counter 0 and 1. When the 8-bit comparator detects a correspondence between the value of the counter register TCNTn and the reference value stored in the TREFn register, the TOLn value is inverted — the latch toggles high-to-low or low-to-high. Whenever the state of TOLn is switched, the TC signal is output. TC output is directed to the TCLOn pin.

Assuming TC is enabled, when bit 3 of the TMODn register is set to "1", the TOLn latch is remained, the counter register, TCNTn and the interrupt request flag, IRQTn are cleared, and counting resumes immediately. When TCn is disabled (TMODn.2 = "0"), the contents of the TOLn latch are retained and can be read, if necessary.

☞ PROGRAMMING TIP — Setting a TC0 Timer Interval

To set a 30 ms timer interval for TC0, given fx = 3.58 MHz, follow these steps.

1. Select the timer/counter 0 mode register with a maximum setup time of 73.3 ms (assume the TC0 counter clock = $fx/2^{10}$, and TREF0 is set to FFH):

2. Calculate the TREF0 value:

$$30 \text{ ms} = \frac{\text{TREF0 value} + 1}{3.49 \text{ kHz}}$$

$$\text{TREF0} + 1 = \frac{30 \text{ ms}}{286\ \mu\text{s}} = 104.8 = 69\text{H}$$

$$\text{TREF0 value} = 69\text{H} - 1 = 68\text{H}$$

3. Load the value 68H to the TREF0 register:

```
BITS    EMB
SMB     15
LD      EA,#68H
LD      TREF0,EA
LD      EA,#4CH
LD      TMOD0,EA
```

WATCH TIMER

OVERVIEW

The watch timer is a multi-purpose timer consisting of three basic components:

- 8-bit watch timer mode register (WMOD)
- Clock selector
- Frequency divider circuit

Watch timer functions include real-time and watch-time measurement and interval timing for the system clock. It is also used as a clock source for generating buzzer output.

Real-Time and Watch-Time Measurement

To start watch timer operation, set bit 2 of the watch timer mode register, WMOD.2, to logic one. The watch timer starts, the interrupt request flag IRQW is automatically set to logic one, and interrupt requests commence in 0.5-second intervals.

Since the watch timer functions as a quasi-interrupt instead of a vectored interrupt, the IRQW flag should be cleared to logic zero by program software as soon as a requested interrupt service routine has been executed.

Using a System Clock Source

The watch timer can generate interrupts based on the system clock frequency. The system clock (fx) is used as the signal source, according to the following formula:

$$\text{Watch timer clock (fw)} = \frac{\text{Main system clock (fx)}}{128} = 32.768\text{kHz}$$

(assuming fx = 4.19 MHz)

Buzzer Output Frequency Generator

The watch timer can generate a steady 2 kHz, 4 kHz, 8 kHz, or 16 kHz signal at 4.19 MHz to the BUZ pin. To select the BUZ frequency you want, load the appropriate value to the WMOD register. This output can then be used to actuate an external buzzer sound. To generate a BUZ signal, three conditions must be met:

- The WMOD.7 register bit is set to "1"
- The output latch for I/O port 2.3 is cleared to "0"
- The port 2.3 output mode flag (PM2.3) set to 'output' mode

Timing Tests in High-Speed Mode

By setting WMOD.1 to "1", the watch timer will function in high-speed mode, generating an interrupt every 3.91 ms at 4.19 MHz. At its normal speed (WMOD.1 = '0'), the watch timer generates an interrupt request every 0.5 seconds. High-speed mode is useful for timing events for program debugging sequences.

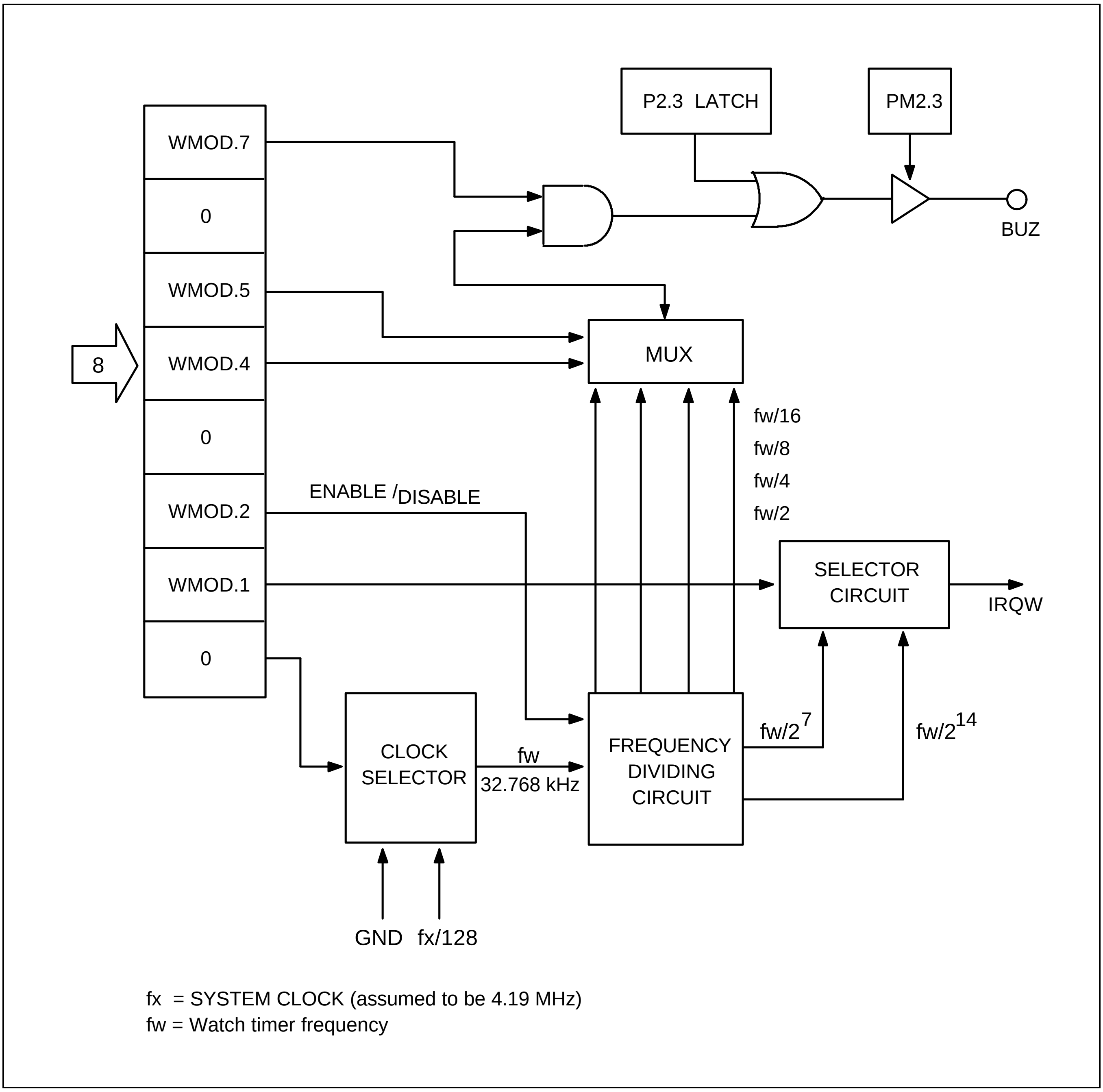


Figure 11-4. Watch Timer Circuit Diagram

WATCH TIMER MODE REGISTER (WMOD)

The watch timer mode register WMOD is used to select specific watch timer operations. It is 8-bit write-only addressable.

F88H	"0"	WMOD.2	WMOD.1	"0"
F89H	WMOD.7	"0"	WMOD.5	WMOD.4

WMOD settings control the following watch timer functions:

— Watch timer speed control (WMOD.1)
— Enable/disable watch timer (WMOD.2)
— Buzzer frequency selection (WMOD.4 and WMOD.5)
— Enable/disable buzzer output (WMOD.7)

Table 11-8. Watch Timer Mode Register (WMOD) Organization

Bit Name	Values		Function	Address
WMOD.7	0		Disable buzzer (BUZ) signal output	F89H
	1		Enable buzzer (BUZ) signal output	
WMOD.6	0		Always logic zero	
WMOD.5 – .4	0	0	fw/16 buzzer (BUZ) signal output (2 kHz)	
	0	1	fw/8 buzzer (BUZ) signal output (4 kHz)	
	1	0	fw/4 buzzer (BUZ) signal output (8 kHz)	
	1	1	fw/2 buzzer (BUZ) signal output (16 kHz)	
WMOD.3	0		Always logic zero	F88H
WMOD.2	0		Disable watch timer; clear frequency dividing circuits	
	1		Enable watch timer	
WMOD.1	0		Normal mode; sets IRQW to 0.5 seconds	
	1		High-speed mode; sets IRQW to 3.91 ms	
WMOD.0	0		Always logic zero	

NOTE: System clock frequency (fx) is assumed to be 4.19 MHz. 'fw' = watch timer clock frequency.

☞ PROGRAMMING TIP — Using the Watch Timer

1. Select a 0.5 second interrupt, and 2 kHz buzzer enable:

```
        BITS    EMB
        SMB     15
        LD      EA,#08H
        LD      PMG1,EA          ;   P2.3 ← output mode
        BITR    P2.3             ;   Clear P2.3 output latch
        LD      EA,#84H
        LD      WMOD,EA
        BITS    IEW
```

2. Sample real-time clock processing method:

```
CLOCK   BTSTZ   IRQW             ;   0.5 second check
        RET                      ;   No, return
        •                        ;   Yes, 0.5 second interrupt generation
        •
        •                        ;   Increment HOUR, MINUTE, SECOND
```

12 DTMF GENERATOR

OVERVIEW

The dual-tone multi-frequency (DTMF) output circuit is used to generate 16 dual-tone multiple frequency signals for tone dialing. This function is controlled by the DTMF mode register. By writing the contents of the output latch for DTMF circuit with output instructions, 16 dual or single tones can be output to the DTMF output pin. The tone output frequency is selected by the DTMF mode register. Figure 12–1 shows the DTMF block diagram. A frequency of 3.58 MHz is used for DTMF generator. Clock output is inhibited when DTMR.0 (DTMF Enable Bit) goes low.

The decoder receives data from the data latch and outputs the result to the row and column tone counter. The row and column tone counter are incremented until new data is latched. When DTMR.0 is logic one, data is latched, and the tone output is changed. Table 12–2 shows the 16 available keyboard frequencies.

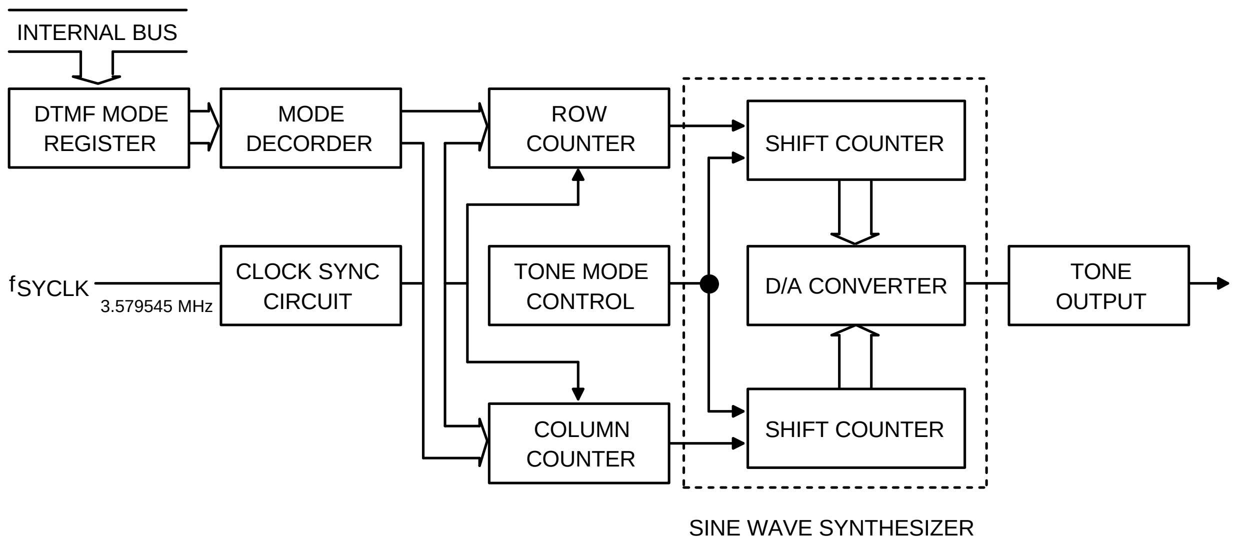


Figure 12-1. Block Diagram of DTMF Generator

Table 12-1. Keyboard Arrangement

COLUMN 1	COLUMN 2	COLUMN 3	COLUMN 4	
1	2	3	A	ROW1
4	5	6	B	ROW2
7	8	9	C	ROW3
*	0	#	D	ROW4

Table 12-2. Tone Output Frequencies

Input	Specified Frequency (Hz)	Actual Frequency (Hz)	% Error
Row1	697	699.1	+ 0.31
Row2	770	766.2	– 0.49
Row3	852	847.4	– 0.54
Row4	941	948.0	+ 0.74
Column 1	1209	1215.7	+ 0.57
Column 2	1336	1331.7	– 0.32
Column 3	1477	1471.7	– 0.35
Column 4	1633	1645.0	+ 0.73

DTMF MODE REGISTER

DTMF output is controlled by the DTMF mode register. Bit position DTMR.0 enables or disables DTMF operation. If DTMR.0 = 1, DTMF operation is enabled.

Programmers should write zeros or ones to bit positions DTMR.4–DTMR.7 according to the keyboard input specification. Writing the data in a look-up table is useful for program efficiency. The DTMR register is a write-only register, and is manipulated using 8-bit RAM control instructions.

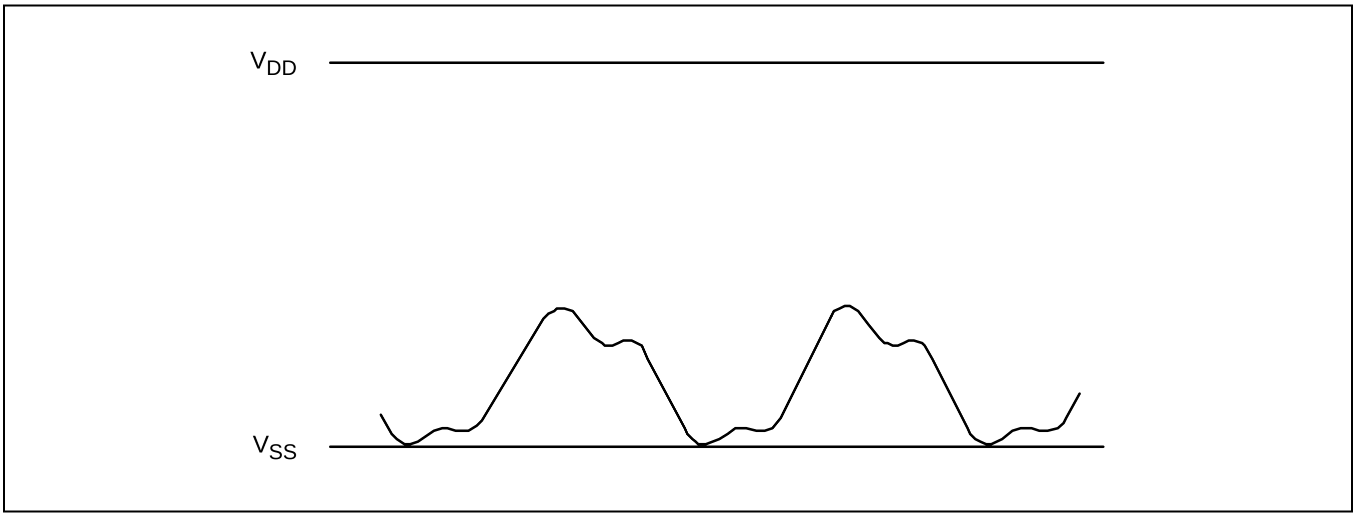


Figure 12-2. DTMF Output Wave Form

Table 12-3. DTMF Mode Register (DTMR) Organization

<table>
<tr><th>Bit Name</th><th colspan="2">Setting</th><th>Resulting DTMF Function</th><th>Address</th></tr>
<tr><td>DTMR.7–.4</td><td colspan="2">0,1</td><td>Specify according to keyboard</td><td>FD3H</td></tr>
<tr><td>DTMR.3</td><td colspan="2">–</td><td>Not Applicable</td><td rowspan="7">FD2H</td></tr>
<tr><td rowspan="4">DTMR.2–.1</td><td>0</td><td>0</td><td rowspan="2">Dual-tone enable</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>Single-column tone enable</td></tr>
<tr><td>1</td><td>1</td><td>Single-low tone enable</td></tr>
<tr><td rowspan="2">DTMR.0</td><td colspan="2">0</td><td>Disable DTMF operation</td></tr>
<tr><td colspan="2">1</td><td>Enable DTMF operation</td></tr>
</table>

Table 12-4. DTMR.7–DTMR.4 key Input Control Settings

DTMR.7	DTMR.6	DTMR.5	DTMR.4	Keyboard
0	0	0	0	D
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	0
1	0	1	1	*
1	1	0	0	#
1	1	0	1	A
1	1	1	0	B
1	1	1	1	C

13 ELECTRICAL DATA

In this section, information on S3C7524/C7528 electrical characteristics is presented as tables and graphics. The information is arranged in the following order:

Standard Electrical Characteristics

- Absolute maximum ratings
- D.C. electrical characteristics
- System clock oscillator characteristics
- I/O capacitance
- A.C. electrical characteristics
- Operating voltage range

Miscellaneous Timing Waveforms

- A.C timing measurement point
- Clock timing measurement at X_{in} and X_{out}
- TCL timing
- Input timing for RESET
- Input timing for external interrupts
- Serial data transfer timing

Stop Mode Characteristics and Timing Waveforms

- RAM data retention supply voltage in stop mode
- Stop mode release timing when initiated by RESET
- Stop mode release timing when initiated by an interrupt request

Table 13-1. Absolute Maximum Ratings

(T_A = 25 °C)

Parameter	Symbol	Conditions	Rating	Units
Supply Voltage	V_{DD}	–	– 0.3 to + 6.5	V
Input Voltage	V_{I1}	All I/O ports	– 0.3 to V_{DD} + 0.3	V
Output Voltage	V_O	–	– 0.3 to V_{DD} + 0.3	V
Output Current High	I_{OH}	One I/O port active	– 15	mA
		All I/O ports active	– 35	
Output Current Low	I_{OL}	One I/O port active	+ 30 (Peak value) + 15 (note)	mA
		All I/O ports active	+ 100 (Peak value) + 60 (note)	
Operating Temperature	T_A	–	– 40 to + 85	°C
Storage Temperature	T_{stg}	–	– 65 to + 150	°C

NOTE: The values for output current low (I_{OL}) are calculated as peak value $\times \sqrt{Duty}$.

Table 13-2. D.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Input high voltage	V_{IH1}	All input pins except those specified below for V_{IH2} – V_{IH3}	0.7 V_{DD}	–	V_{DD}	V
	V_{IH2}	Ports 1, 3, 6, 7, and RESET	0.8 V_{DD}		V_{DD}	
	V_{IH3}	X_{in} and X_{out}	V_{DD} – 0.1		V_{DD}	
Input low voltage	V_{IL1}	All input pins except those specified below for V_{IL2}–V_{IL3}	–	–	0.3 V_{DD}	V
	V_{IL2}	Ports 1, 3, 6, 7, and RESET			0.2 V_{DD}	
	V_{IL3}	X_{in} and X_{out}			0.1	

Table 13-2. D.C. Electrical Characteristics (Continued)

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Output high voltage	V_{OH}	I_{OH} = – 1 mA Ports except 1	V_{DD} – 1.0	–	–	V
Output low voltage	V_{OL1}	V_{DD} = 4.5 V to 5.5 V I_{OL} = 15 mA, Ports 4 and 5 only	–	0.4	2	V
		V_{DD} = 2.0 to 5.5 V, I_{OL} = 1.6mA	–	–	0.4	
	V_{OL2}	V_{DD} = 4.5 V to 5.5 V I_{OL}= 4 mA, all out ports except 4,5	–	–	2	V
		V_{DD} = 2.0 to 5.5 V, I_{OL} = 1.6mA	–	–	0.4	
Input high leakage current	I_{LIH1}	$V_I = V_{DD}$ All input pins except those specified below	–	–	3	μA
	I_{LIH2}	$V_I = V_{DD}$ X_{in} and X_{out}			20	
Input low leakage current	I_{LIL1}	V_I = 0 V All input pins except below and RESET	–	–	– 3	μA
	I_{LIL2}	V_I = 0 V X_{in} and X_{out} only			– 20	
Output high leakage current	I_{LOH}	$V_O = V_{DD}$ All out pins	–	–	3	μA
Output low leakage current	I_{LOL}	V_O = 0 V X_{in} and X_{out} only	–	–	– 3	μA
Pull-up resistor	R_{L1}	V_{DD} = 5 V; V_I = 0 V except RESET	25	47	100	kΩ
		V_{DD} = 3 V	50	95	200	
	R_{L2}	V_{DD} = 5 V; V_I = 0 V; RESET	100	220	400	
		V_{DD} = 3 V	200	450	800	

Table 13-2. D.C. Electrical Characteristics (Concluded)

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

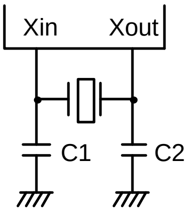
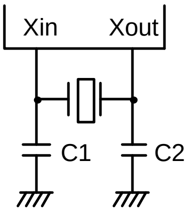
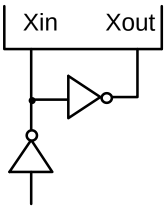
Parameter	Symbol	Conditions		Min	Typ	Max	Units
Supply current [(1)]	I_{DD1} (DTMF on)	Run mode; V_{DD} = 5 V ± 10% [(2)] 3.58 MHz crystal oscillator, C1 = C2 = 22 pF		–	2.9	5.0	mA
		V_{DD} = 3 V ± 10%			1.6	3.0	
	I_{DD2} (DTMF off)	Run mode; V_{DD} = 5 V ± 10%	6.0 MHz	–	2.6	8.0	mA
		crystal oscillator, C1 = C2 = 22 pF	3.58 MHz		1.8	4.0	
		V_{DD} = 3 V ± 10%	6.0 MHz		1.8	4.0	
			3.58 MHz		1.2	2.3	
	$I_{DD}3$	Idle mode; = V_{DD} = 5 V ± 10%	6.0 MHz	–	0.7	2.5	mA
		crystal oscillator, C1 = C2 = 22 pF	3.58 MHz		0.6	1.8	
		V_{DD} = 3 V ± 10%	6.0 MHz		0.3	1.5	
			3.58 MHz		0.2	1.0	
	I_{DD4}	Stop mode; V_{DD} = 5 V ± 10%		–	0.01	3	μA
		Stop mode; V_{DD} = 3 V ± 10%			0.01	2	
Row tone level	V_{ROW}	V_{DD} = 5 V ± 10% V_{DD} = 3 V ± 10% V_{DD} = 2 V RL = 5kΩ		– 16.0	– 14.0	– 11.0	dBV
Ratio of column to row tone	dB_{CR}	V_{DD} = 5 V ± 10% V_{DD} = 3 V ± 10% V_{DD} = 2 V RL = 5kΩ		1	2	3	
Distortion (Dual tone)	THD	V_{DD} = 5 V ± 10% V_{DD} = 3 V ± 10% V_{DD} = 2 V RL = 5kΩ, 1MHz band		–	–	5	%

NOTES

1. D.C. electrical values for Supply Current (I_{DD1} to I_{DD3}) do not include current drawn through internal pull-up registers.
2. For D.C. electrical values, the power control register (PCON) must be set to 0011B.

Table 13-3. Main System Clock Oscillator Characteristics

(T_A = – 40 °C + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Oscillator	Clock Configuration	Parameter	Test Condition	Min	Typ	Max	Units
Ceramic Oscillator		Oscillation frequency (1)	V_{DD} = 2.7 V to 5.5 V	0.4	–	6.0	MHz
			V_{DD} = 2.0 V to 5.5 V	0.4	–	4.2	
		Stabilization time (2)	V_{DD} = 3 V	–	–	4	ms
Crystal Oscillator		Oscillation frequency (1)	V_{DD} = 2.7 V to 5.5 V	0.4	–	6.0	MHz
			V_{DD} = 2.0 V to 5.5 V	0.4	–	4.2	
		Stabilization time (2)	V_{DD} = 3 V	–	–	10	ms
External Clock		X_{in} input frequency (1)	V_{DD} = 2.7 V to 5.5 V	0.4	–	6.0	MHz
			V_{DD} = 2.0 V to 5.5V	0.4	–	4.2	
		X_{in} input high and low level width (t_{XH}, t_{XL})	–	83.3	–	1250	ns

NOTES

1. Oscillation frequency and X_{in} input frequency data are for oscillator characteristics only.
2. Stabilization time is the interval required for oscillating stabilization after a power-on occurs, or when stop mode is terminated.

Table 13-4. Input/Output Capacitance

(T_A = 25 °C, V_{DD} = 0 V)

Parameter	Symbol	Condition	Min	Typ	Max	Units
Input Capacitance	C_{IN}	f = 1 MHz; Unmeasured pins are returned to V_{SS}	–	–	15	pF
Output Capacitance	C_{OUT}		–	–	15	pF
I/O Capacitance	C_{IO}		–	–	15	pF

Table 13-5. A.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 2.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (1)	t_{CY}	V_{DD} = 2.7 V to 5.5 V	0.67	–	64	µs
		V_{DD} = 2.0 V to 5.5 V	0.95			
TCL0, TCL1 Input Frequency	f_{TI0}, f_{TI1}	V_{DD} = 2.7 V to 5.5 V	0	–	1.5	MHz
		V_{DD} = 2.0 V to 5.5V			1	MHz
TCL0, TCL1 Input High, Low Width	t_{TIH0}, t_{TIL0} t_{TIH1}, t_{TIL1}	V_{DD} = 2.7 V to 5.5 V	0.48	–	–	µs
		V_{DD} = 2.0 V to 5.5 V	1.8			
Interrupt Input High, Low Width	t_{INTH}, t_{INTL}	INT0, INT1, INT2, INT4, KS0–KS7	10	–	–	µs
RESET Input Low Width	t_{RSL}	Input	10	–	–	µs

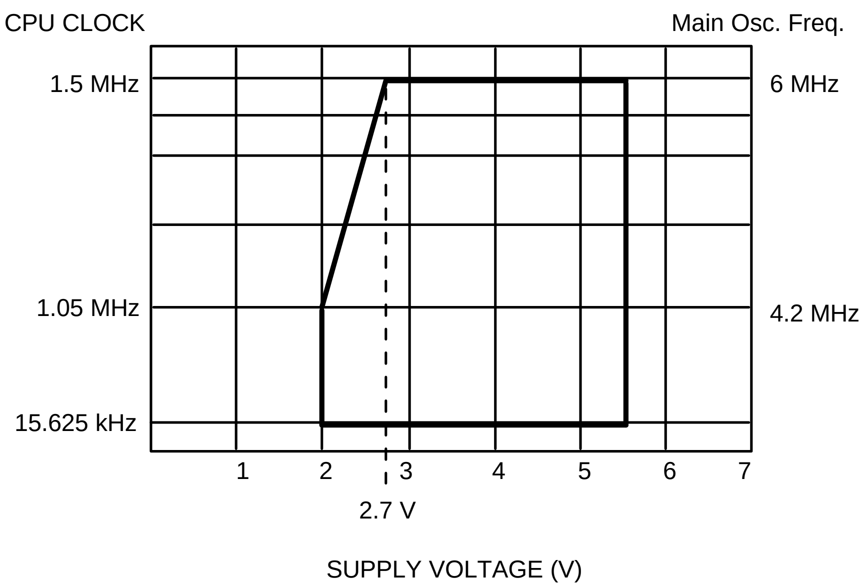


Figure 13-1. Standard Operating Voltage Range

Table 13-6. RAM 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}	–	1.5	–	5.5	V
Data retention supply current	I_{DDDR}	V_{DDDR} = 1.5 V	–	0.1	10	µA
Release signal set time	t_{SREL}	–	0	–	–	µs
Oscillator stabilization wait time (1)	t_{WAIT}	Released by RESET Released by interrupt	–	2^{17}/fx (2)	–	ms ms

NOTES

1. During oscillator stabilization wait time, all CPU operations must be stopped to avoid instability during oscillator start-up.
2. Use the basic timer mode register (BMOD) interval timer to delay execution of CPU instructions during the wait time.

TIMING WAVEFORMS

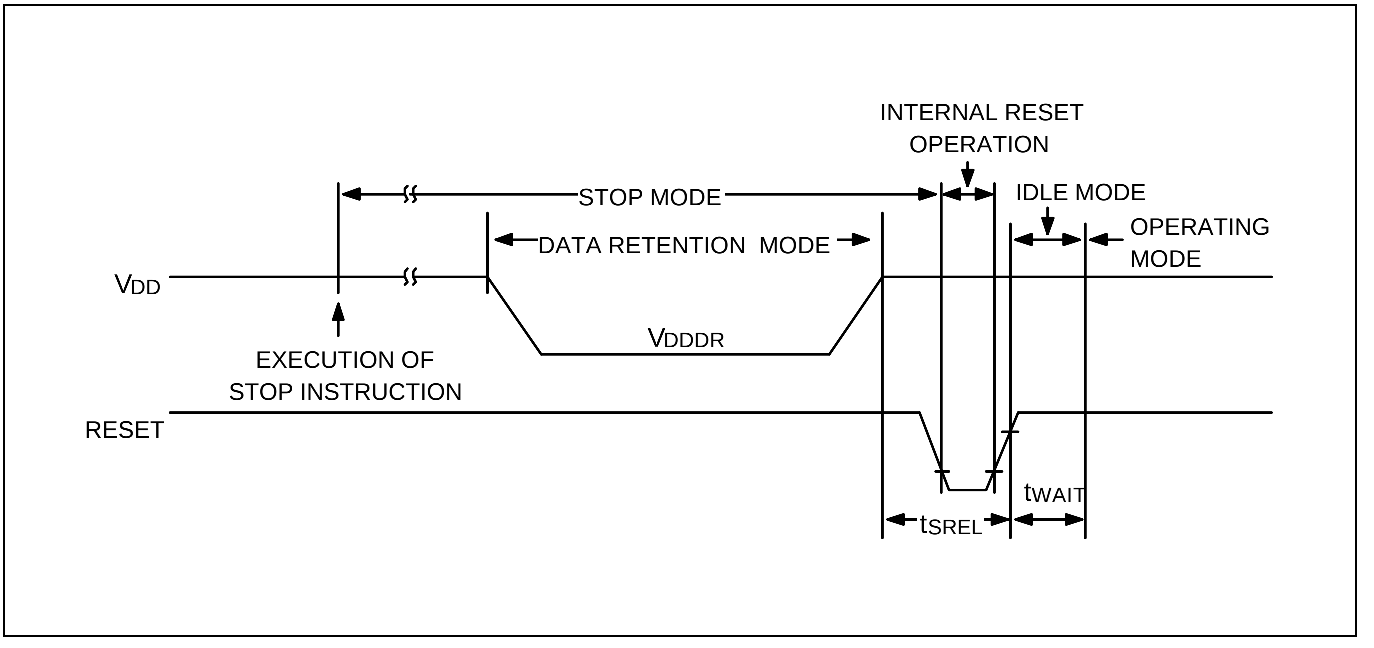


Figure 13-2. Stop Mode Release Timing When Initiated By RESET

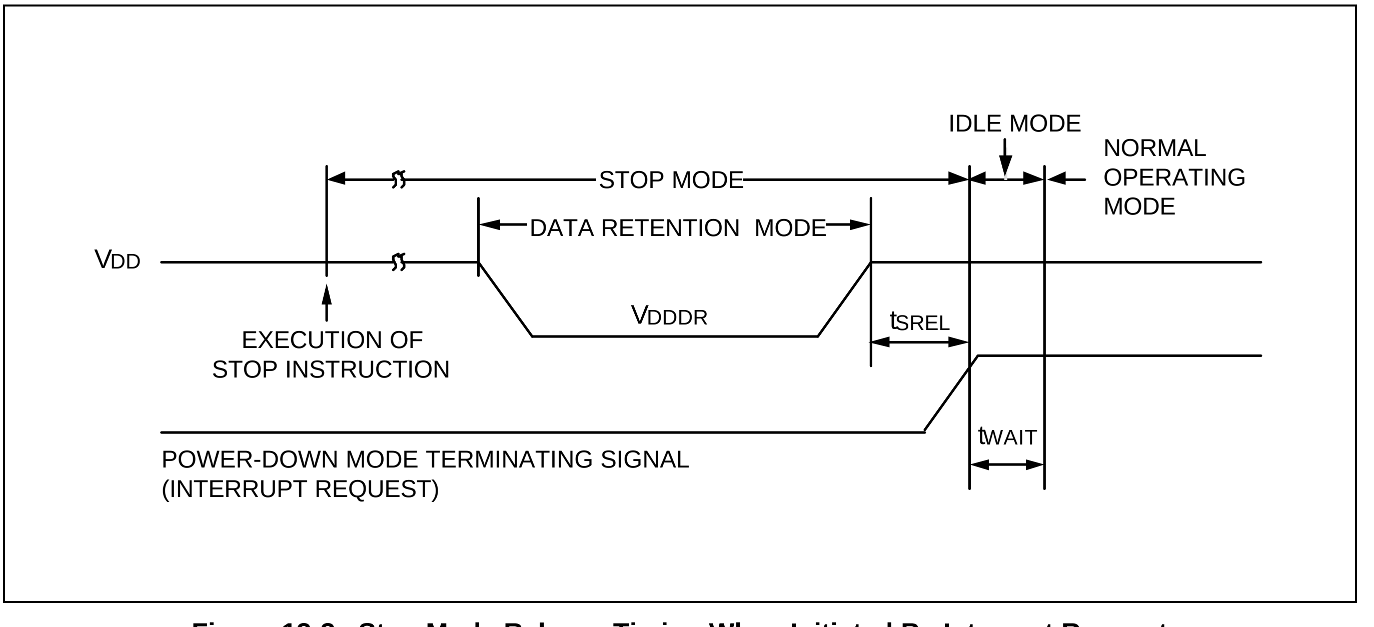


Figure 13-3. Stop Mode Release Timing When Initiated By Interrupt Request

Timing Waveforms (continued)

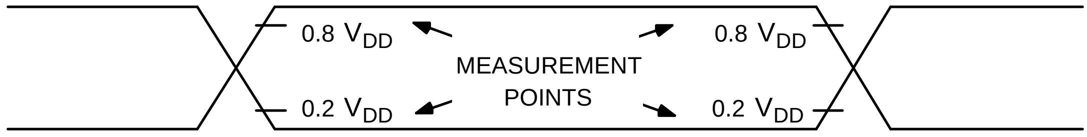


Figure 13-4. A.C. Timing Measurement Points (Except for X_{in})

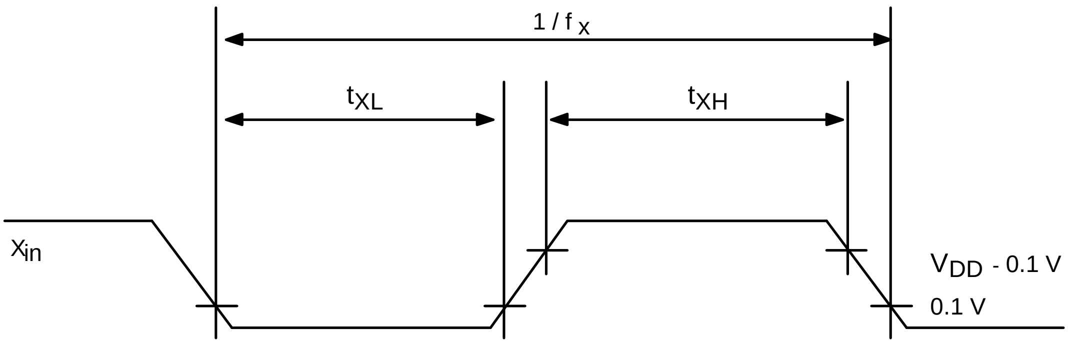


Figure 13-5. Clock Timing Measurement at X_{in}

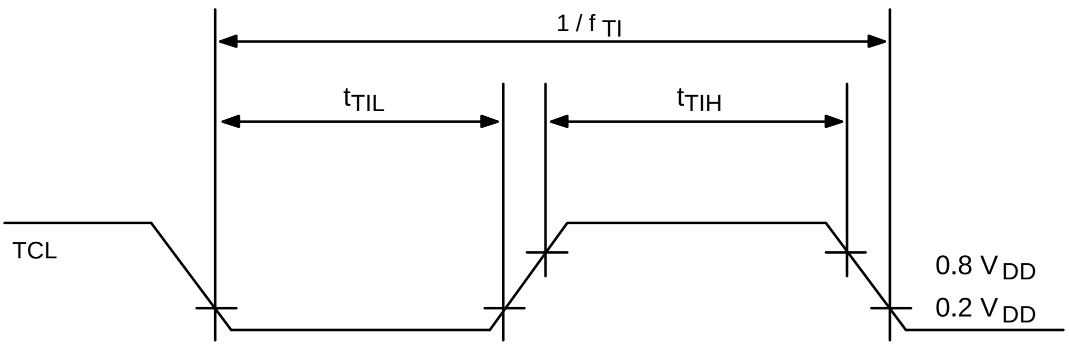


Figure 13-6. TCL Timing

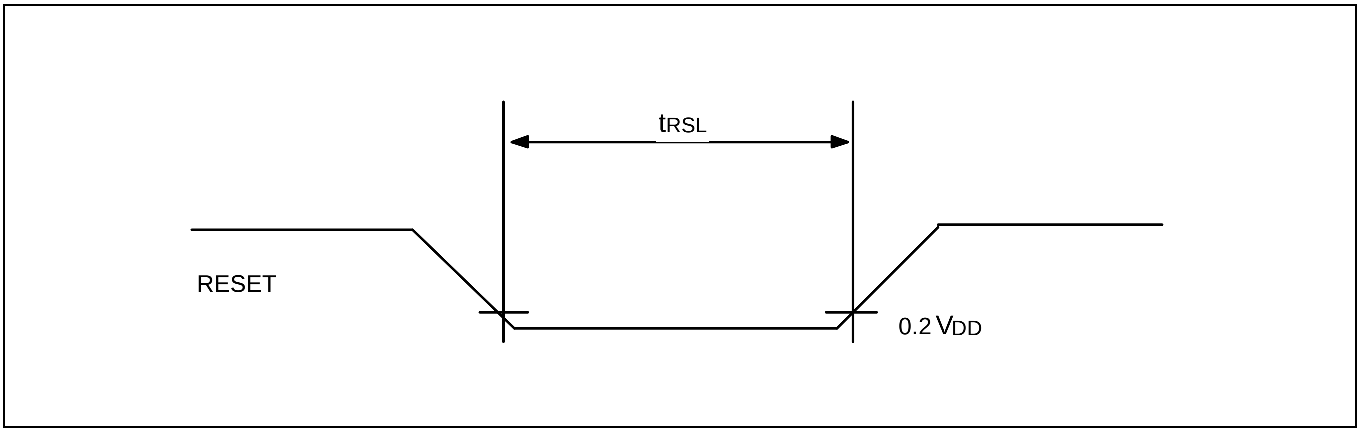


Figure 13-7. Input Timing for RESET Signal

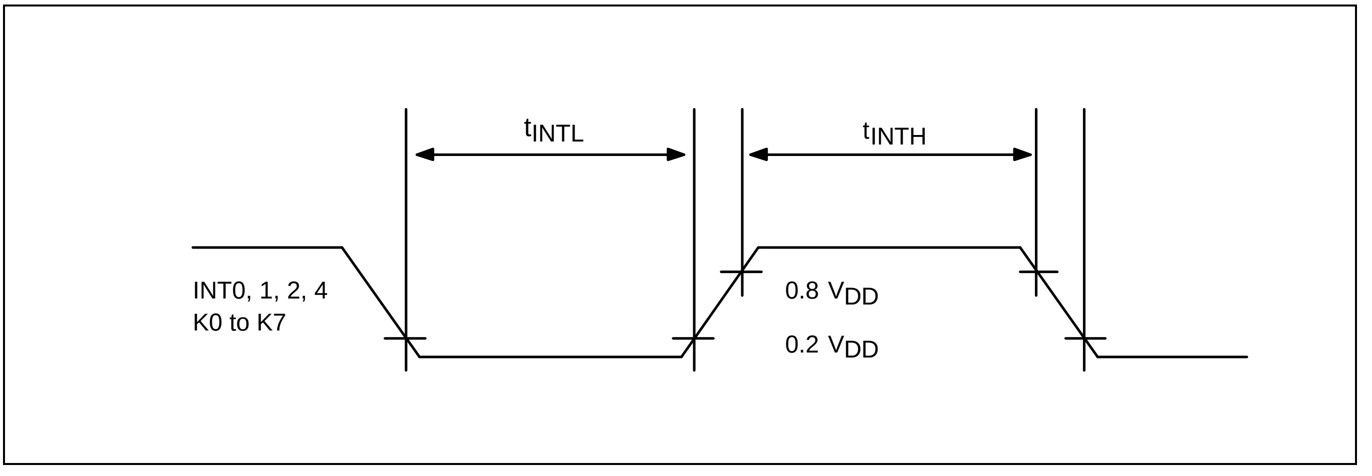


Figure 13-8. Input Timing for External Interrupts and Quasi-Interrupts

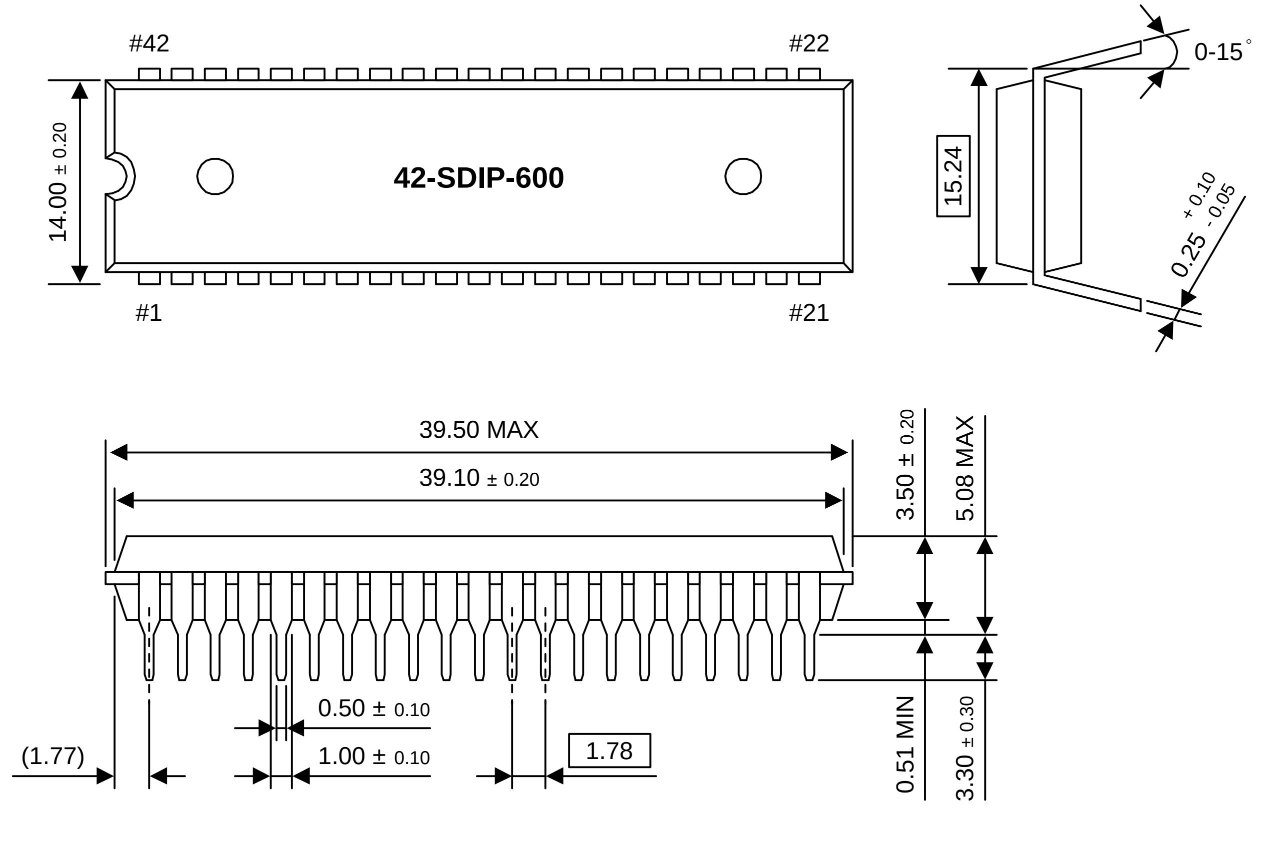
14 MECHANICAL DATA

This section contains the following information about the device package:

— Package dimensions in millimeters

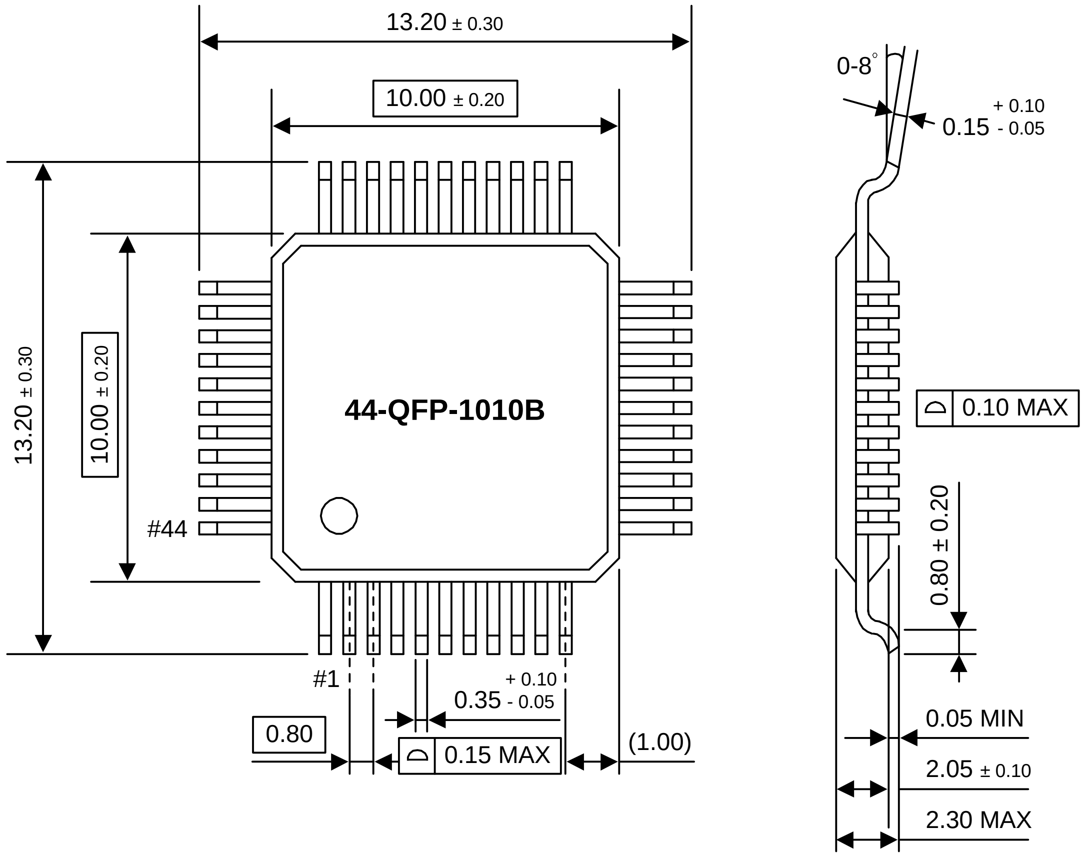
— Pad diagram

— Pad/pin coordinate data table



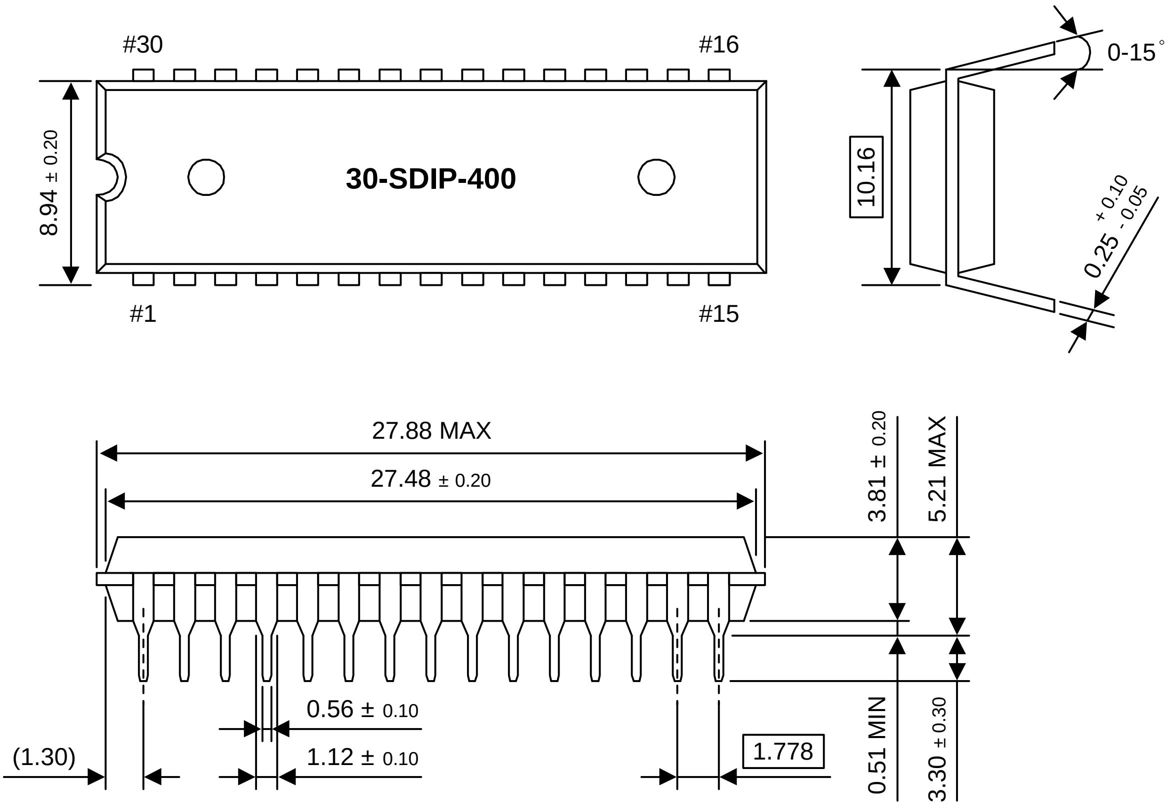
NOTE: Dimensions are in millimeters.

Figure 14-1. 42-SDIP-600 Package Dimensions



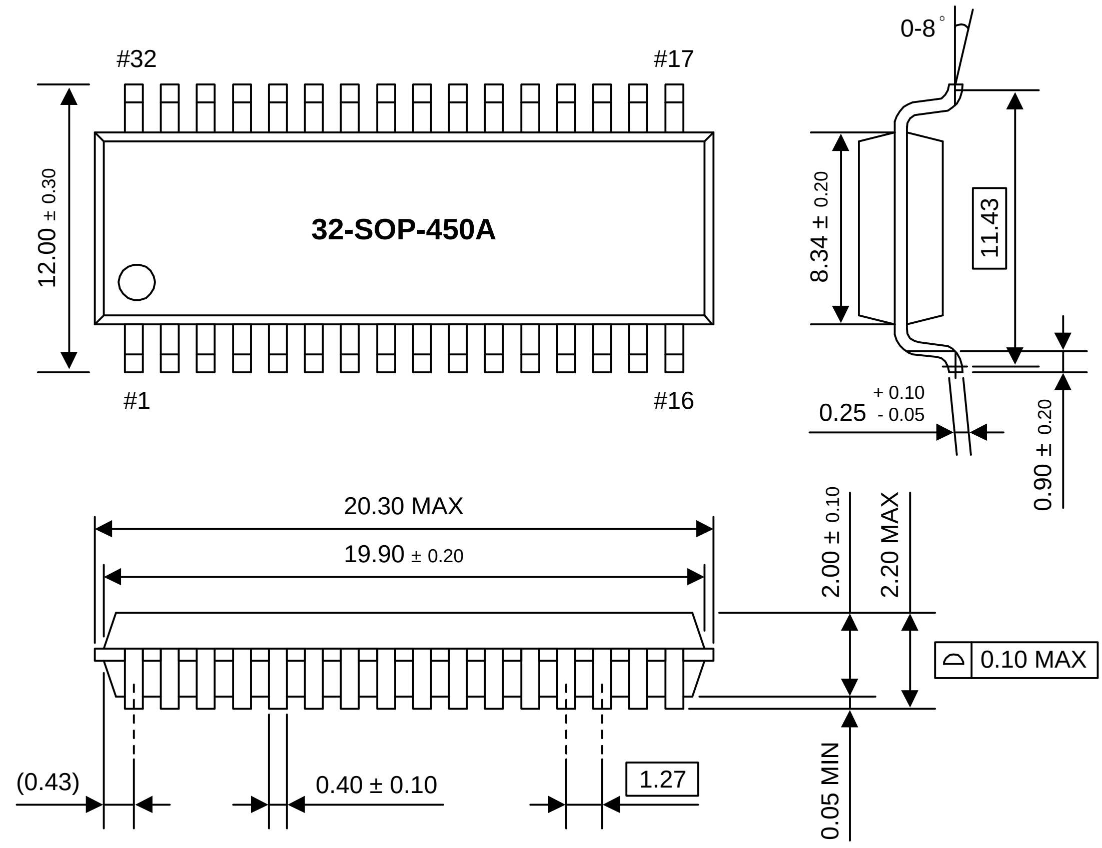
NOTE: Dimensions are in millimeters.

Figure 14-2. 44-QFP-1010B Package Dimensions



NOTE: Dimensions are in millimeters.

Figure 14-3. 30-SDIP-400 Package Dimensions



NOTE: Dimensions are in millimeters.

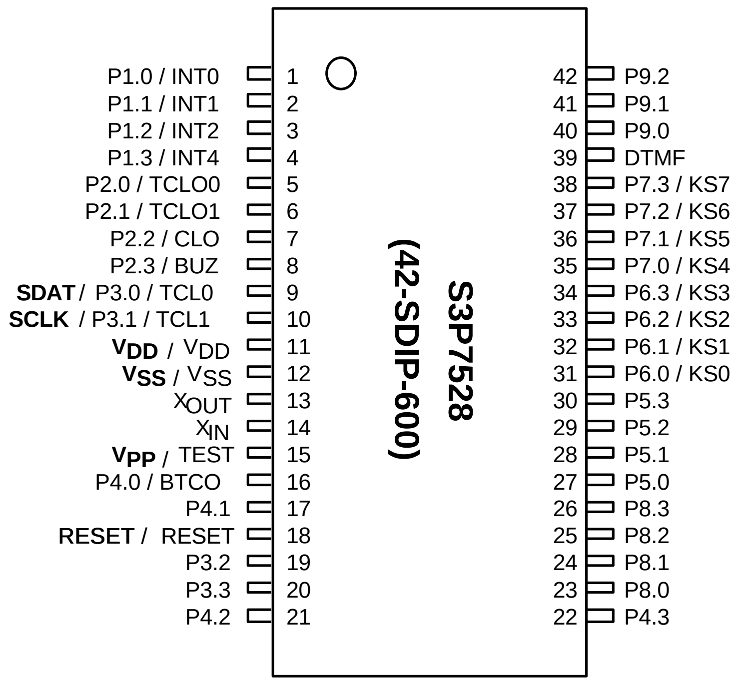
Figure 14-4. 32-SOP-450A Package Dimensions

15 S3P7528/P7538 OTP

OVERVIEW

The S3P7528/P7538 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the S3C7524/C7528/C7534/C7538 microcontroller. It has an on-chip EPROM instead of masked ROM. The EPROM is accessed by a serial data format.

The S3P7528/P7538 is fully compatible with the S3C7528/C7538, both in function and in pin configuration. Because of its simple programming requirements, the S3P7528/P7538 is ideal for use as an evaluation chip for the S3C7528/C7538.



NOTE: The bold words indicate OTP pin names.

Figure 15-1. S3P7528 Pin Assignments (42-SDIP)

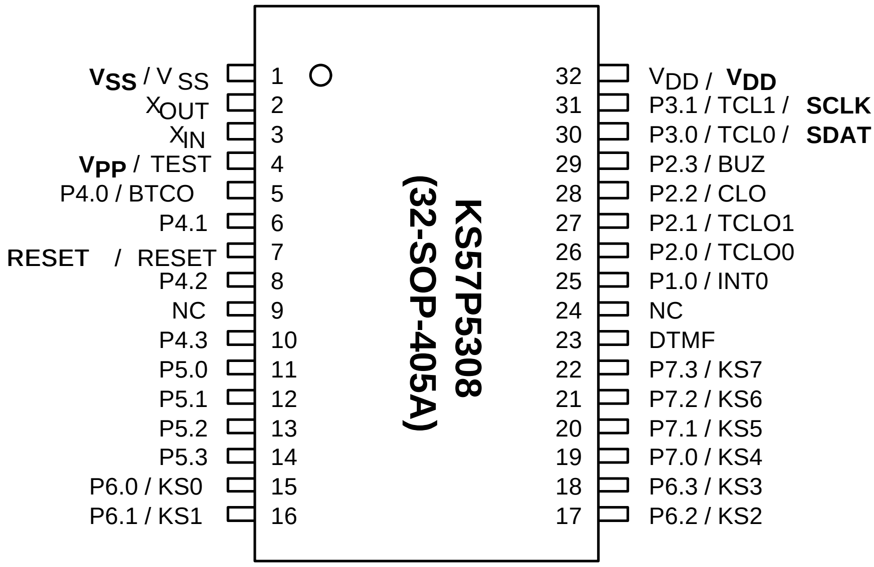


Figure 15-2. S3P7528 Pin Assignments (44-QFP)

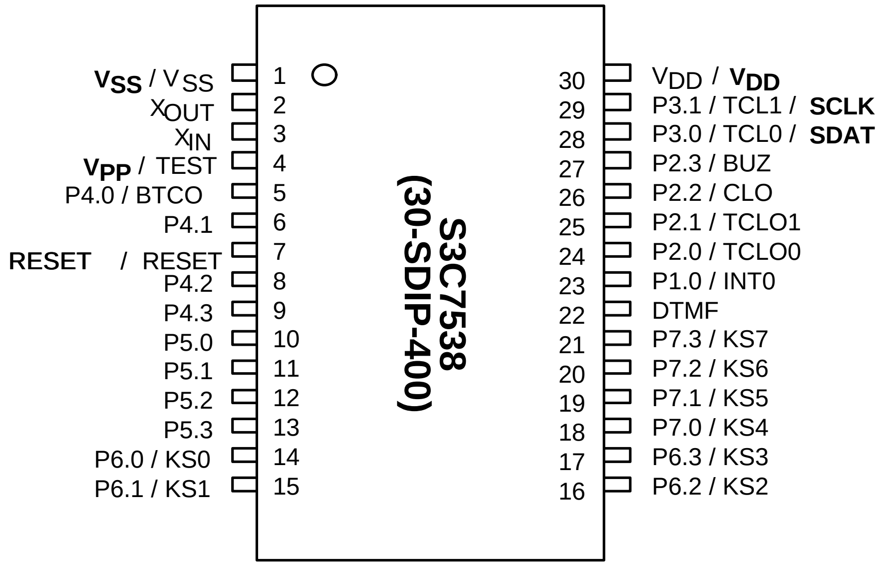


Figure 15-3. S3P7538 Pin Assignments (30-SDIP)

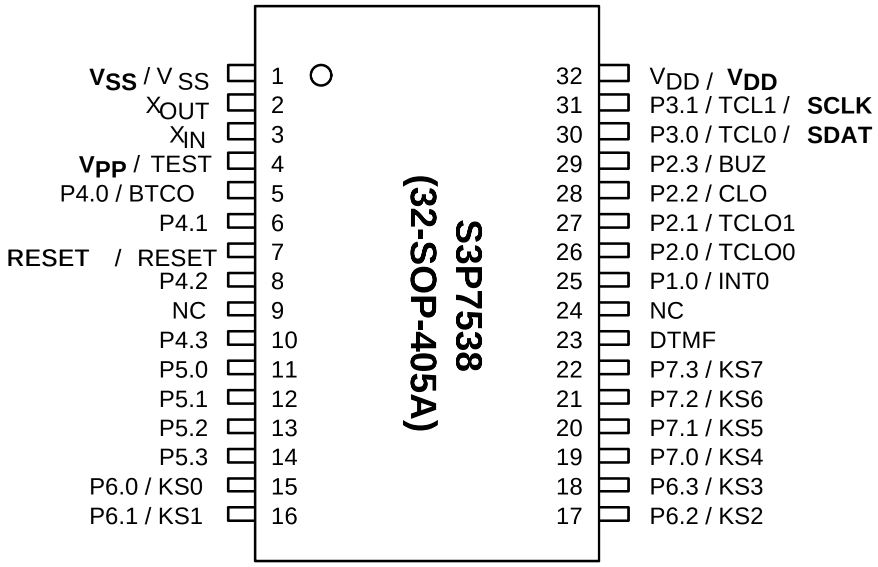


Figure 15-4. S3P7538 Pin Assignments (32-SOP)

Table 15-1. S3P7528 Pin Descriptions Used to Read/Write the EPROM

Main Chip	During Programming			
Pin Name	**Pin Name**	**Pin No.**	**I/O**	**Function**
P3.0	SDAT	9 (3)	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input / push-pull output port.
P3.1	SCLK	10 (4)	I/O	Serial clock pin. Input only pin.
TEST	V_{PP} (TEST)	15 (9)	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is applied, OTP is in reading mode. (Option)
RESET	RESET	18 (12)	I	Chip initialization
V_{DD} / V_{SS}	V_{DD} / V_{SS}	11/12 (5/6)	I	Logic power supply pin. V_{DD} should be tied to +5 V during programming.

NOTE: Parentheses indicate pin numbers of 44 QFP package.

Table 15-2. S3P7538 Pin Descriptions Used to Read/Write the EPROM

Main Chip	During Programming			
Pin Name	**Pin Name**	**Pin No.**	**I/O**	**Function**
P3.0	SDAT	28 (30)	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input / push-pull output port.
P3.1	SCLK	29 (31)	I/O	Serial clock pin. Input only pin.
TEST	V_{PP} (TEST)	4 (4)	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is applied, OTP is in reading mode. (Option)
RESET	RESET	7 (7)	I	Chip initialization
V_{DD} / V_{SS}	V_{DD} / V_{SS}	30/1 (32/1)	I	Logic power supply pin. V_{DD} should be tied to +5 V during programming.

NOTE: Parentheses indicate pin numbers of 32 SDIP package.

Table 15-3. Comparison of S3P7528 and S3C7528 Features

Characteristic	S3P7528	S3C7528
Program Memory	8 K byte EPROM	8 K byte mask ROM
Operating Voltage (V_{DD})	2.0 V to 5.5 V	2.0 V to 5.5 V
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (TEST) = 12.5 V	–
Pin Configuration	42 SDIP / 44 QFP	42 SDIP / 44 QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

Table 15-4. Comparison of S3P7538 and S3C7538 Features

Characteristic	S3P7538	S3C7538
Program Memory	8 K byte EPROM	8 K byte mask ROM
Operating Voltage (V_{DD})	2.0 V to 5.5 V	2.0 V to 5.5 V
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (TEST) = 12.5 V	–
Pin Configuration	30 SOP / 32 SOP	30 SOP / 32 SOP
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 S3P7528, 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 15-3 below.

Table 15-5. Operating Mode Selection Criteria

V_{DD}	Vpp (TEST)	REG/ MEM	Address (A15-A0)	R/W	Mode
5 V	5 V	0	0000H	1	EPROM read
	12.5V	0	0000H	0	EPROM program
	12.5V	0	0000H	1	EPROM verify
	12.5V	1	0E3FH	0	EPROM read protection

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

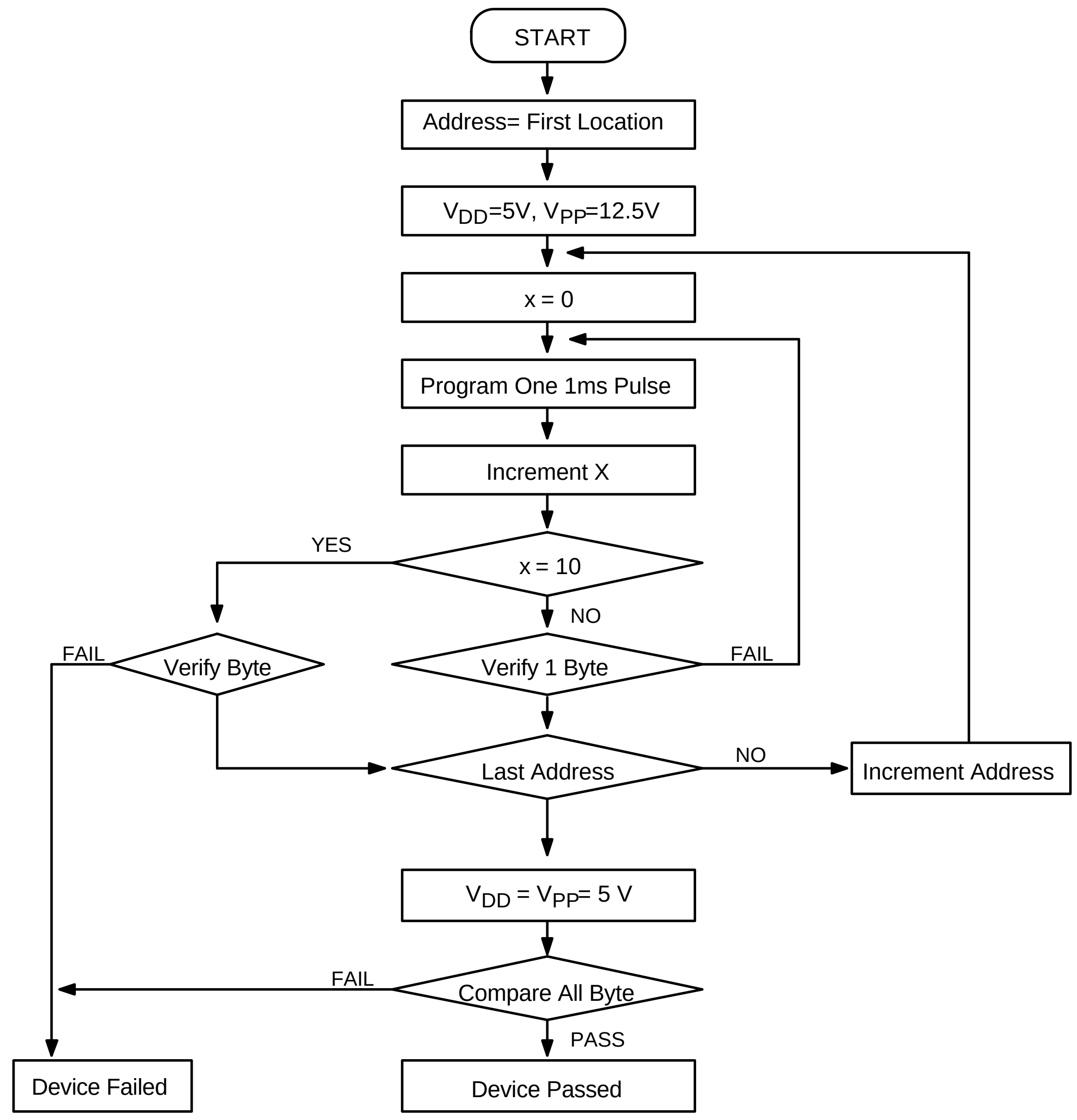


Figure 15-5. OTP Programming Algorithm

16 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system in turnkey form. The development support system is configured with a host system, debugging tools, and support software. For the host system, any standard computer that operates with MS-DOS as its operating system can be used. One type of debugging tool including hardware and software is provided: the sophisticated and powerful in-circuit emulator, SMDS2+, for S3C7, S3C9, S3C8 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

SHINE

Samsung Host Interface for In-Circuit Emulator, SHINE, is a multi-window based debugger for SMDS2+. SHINE provides pull-down and pop-up menus, mouse support, function/hot keys, and context-sensitive hyper-linked help. It has an advanced, multiple-windowed user interface that emphasizes ease of use. Each window can be sized, moved, scrolled, highlighted, added, or removed completely.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generates object code in standard hexadecimal format. Assembled program code includes the object code that is used for ROM data and required SMDS program control data. To assemble programs, SAMA requires a source file and an auxiliary definition (DEF) file with device specific information.

SASM57

The SASM57 is an relocatable assembler for Samsung's S3C7-series microcontrollers. The SASM57 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM57 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 S3C7-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 S3C7524/C7528/C7534/C7538 microcontroller and OTP programmer (Gang) are now available.

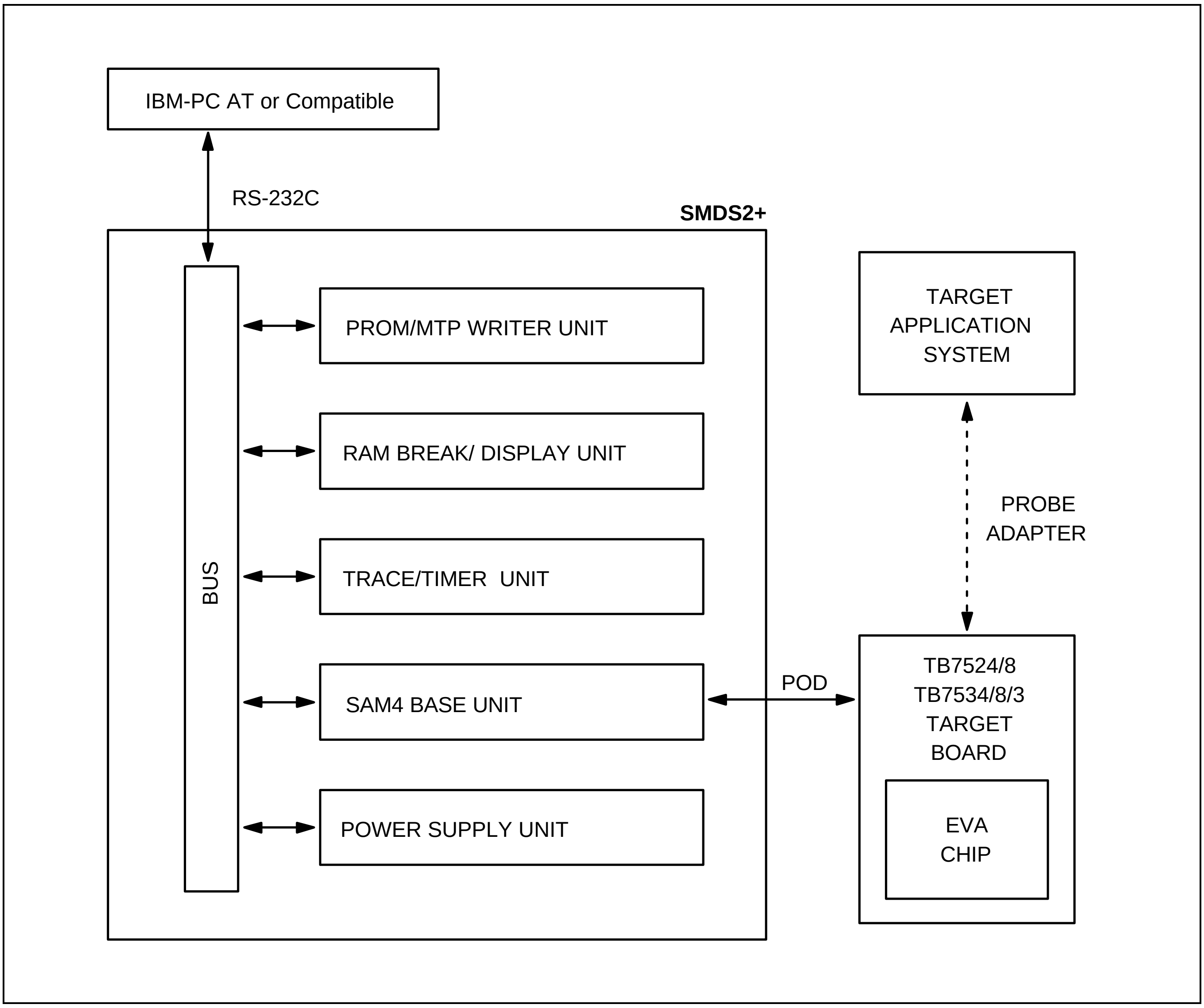


Figure 16-1. SMDS Product Configuration (SMDS2+)

TB7524/528/534/538/533 TARGET BOARD

The TB7524/528/534/538/533 target board is used for the S3C7524/C7528/P7528/C7534/C7538/P7538 microcontroller. It is supported by the SMDS2+ development system.

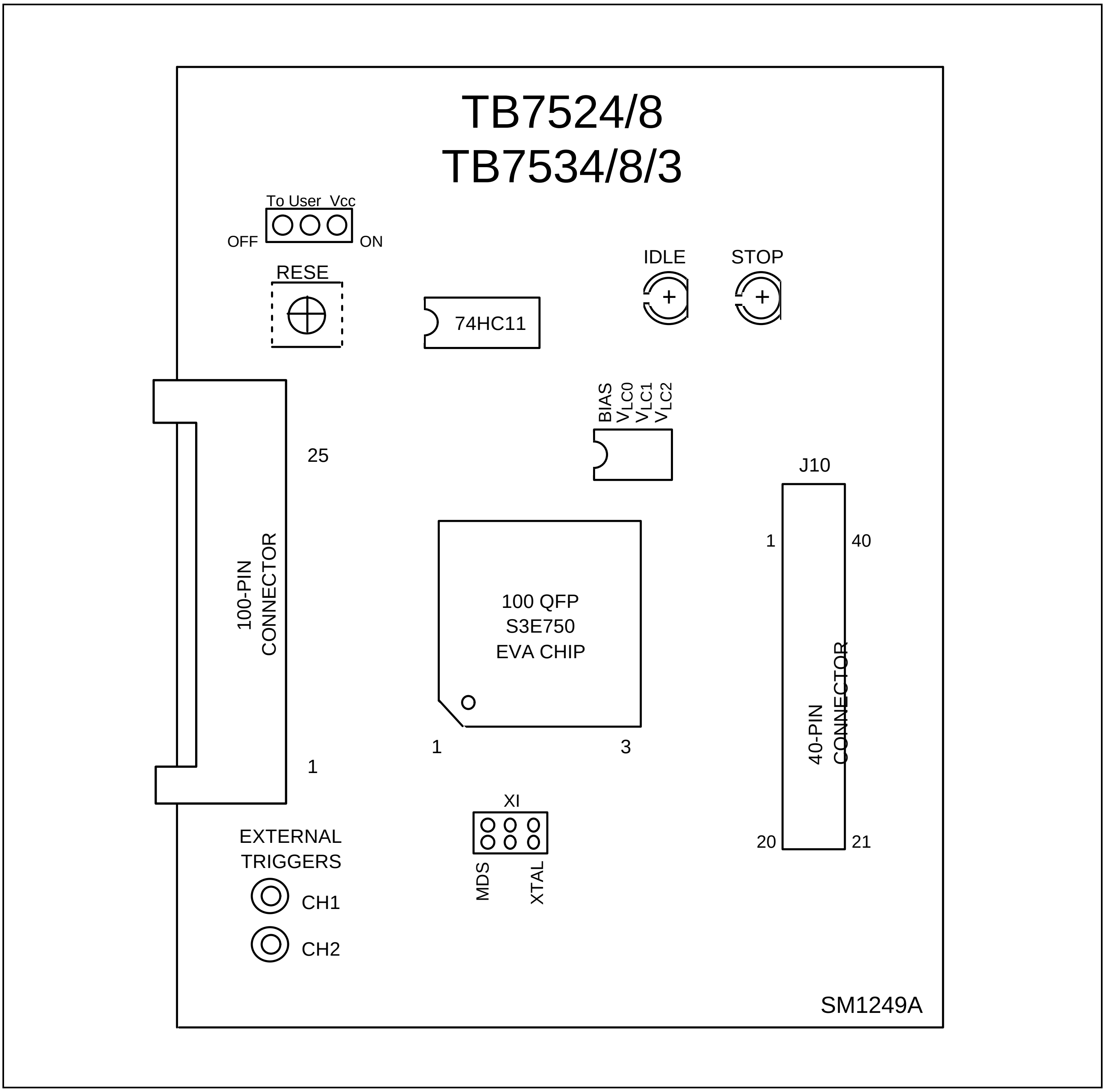


Figure 16-2. TB7524/528/534/538/533 Target Board Configuration

Table 16-1. Power Selection Settings for TB7524/528/534/538/533

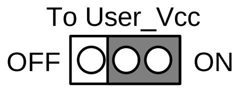
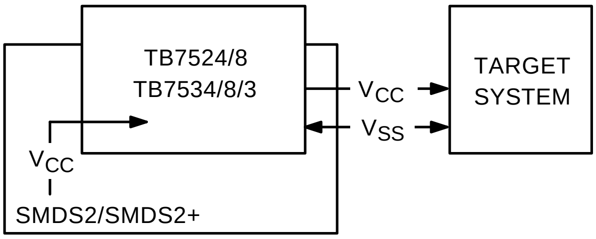
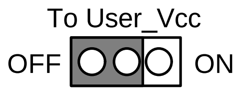
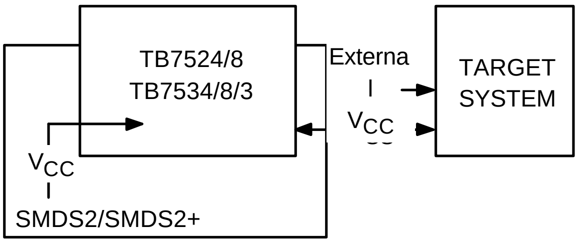
<table>
<tr><th>'To User_Vcc' Settings</th><th>Operating Mode</th><th>Comments</th></tr>
<tr><td></td><td></td><td>The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.</td></tr>
<tr><td></td><td></td><td>The SMDS2/SMDS2+ supplies V_{CC} only to the target board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

Table 16-2. Main-clock Selection Settings for TB7524/528/534/538/533

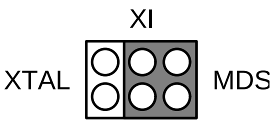
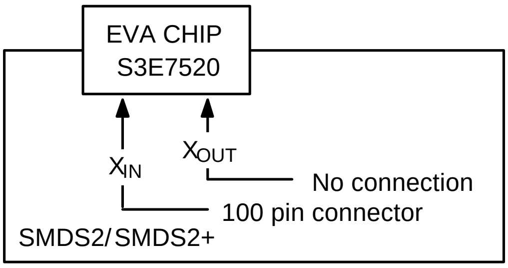
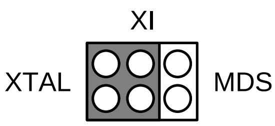
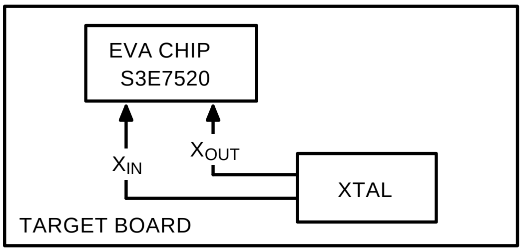
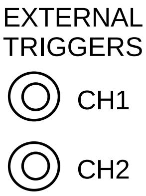
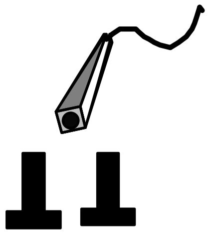
<table>
<tr><th>Sub Clock Setting</th><th>Operating Mode</th><th>Comments</th></tr>
<tr><td></td><td></td><td>Set the XI switch to “MDS” when the target board is connected to the SMDS2/SMDS2+.</td></tr>
<tr><td></td><td></td><td>Set the XI switch to “XTAL” when the target board is used as a standalone unit, and is not connected to the SMDS2/SMDS2+.</td></tr>
</table>

Table 16-4. Using Single Header Pins as the Input Path for External Trigger Sources

Target Board Part	Comments
 	 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 (S3E7520) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (S3E7520) is in stop mode.

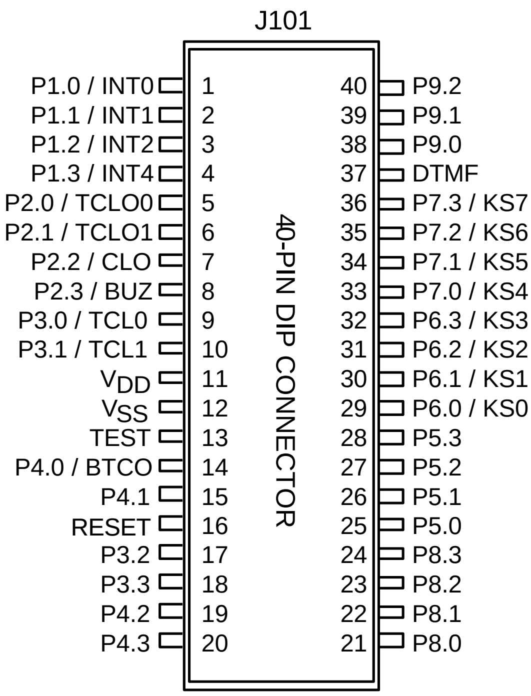


Figure 16-3. 40-Pin Connectors for TB7524/528/534/538/533

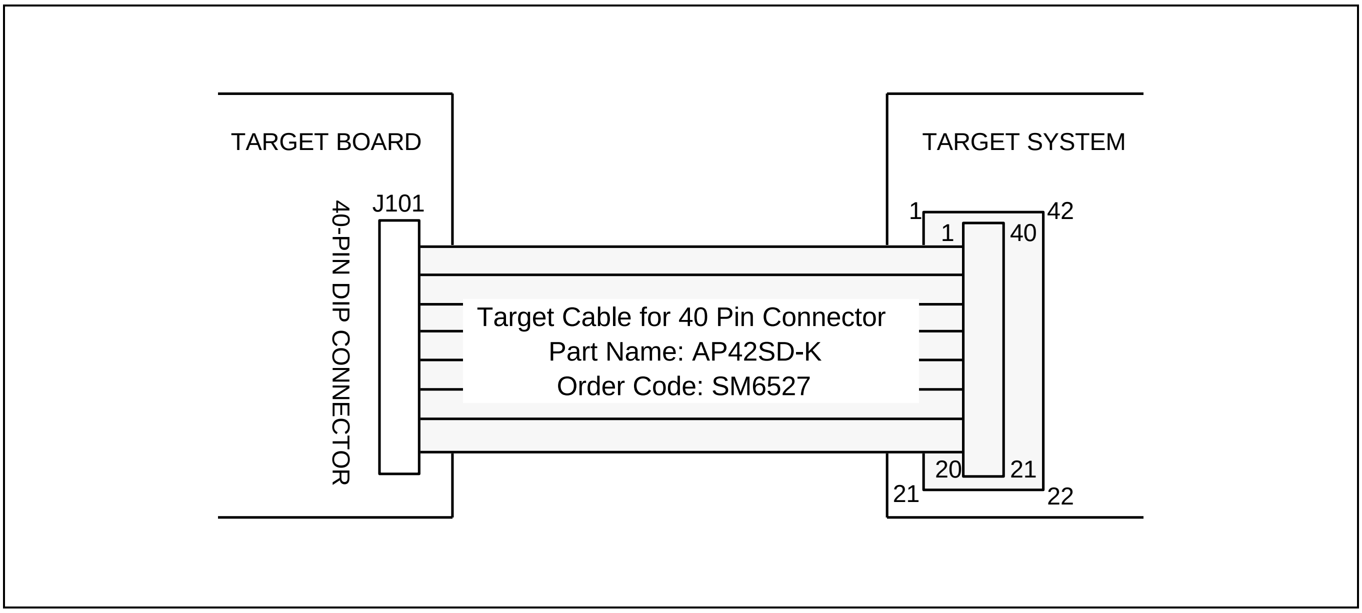


Figure 16-4. TB575204A/5208A Adapter Cable for 42-SDIP Package (S3C7524/C7528/P7528)

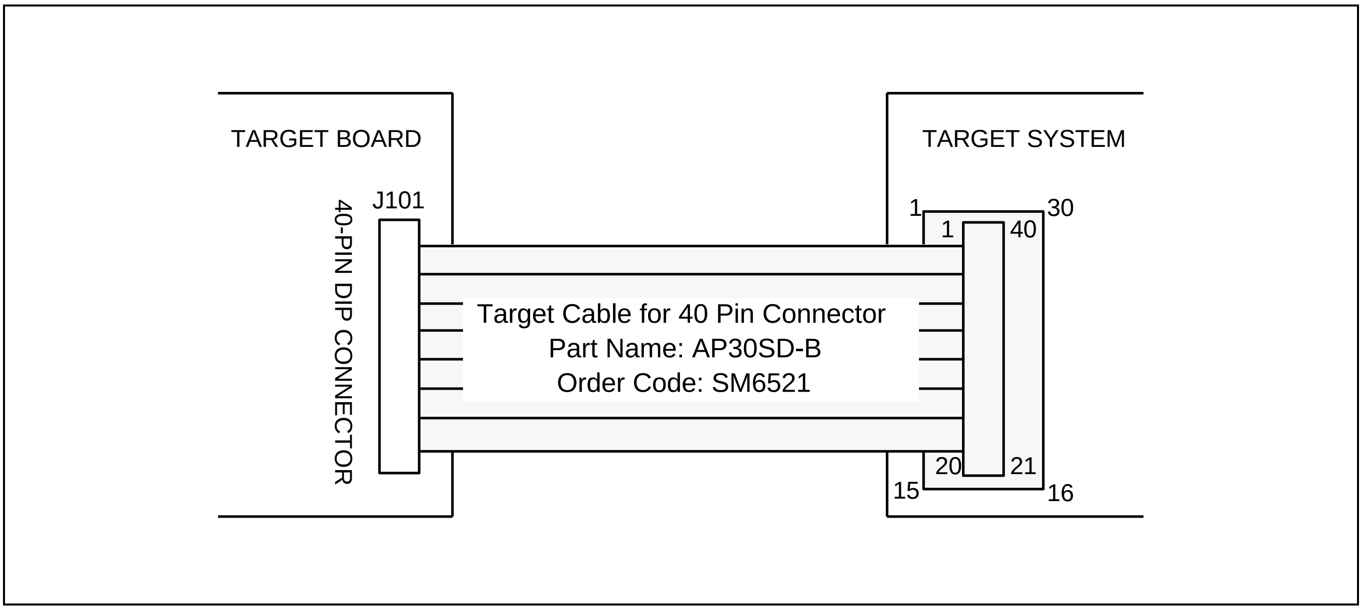


Figure 16-5. TB575304A/5308A Adapter Cable for 30-SDIP Package (S3C7534/C7538/P7538)