Chapter 1 General

Creative Core Components

Sharp electronic components create new value

Single-chip 16-bit Microcomputer

SM6010 Series

User's Manual

SHARP

Preliminary
Ref. No. SCX031704E

○ Handle this document carefully for it contains material protected by international copyright law. Any reproduction, full or in part, without the express written permission of the company, of this material is prohibited.

○ The application circuit examples in this document are provided to explaining the representative application of SHARP devices. They are not to guarantee the exercise of industrial property right or other rights, nor to permit a license to use patented applications/technology. SHARP takes no responsibility for any problems related to industrial property right or other rights of a third party resulting from the use of SHARP's devices.

○ SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structures, and other contents described herein at any time without notice in order to improve design or reliability. Contact SHARP in order to obtain the latest document before using the devices shown in this document.

○ In no event shall the company be liable for any damages resulting from misuse or inappropriate use of the products and failure to strictly adhere to the following conditions and precautions.

(1) The products covered herein are, in principle, designed and manufactured for the following purposes. When using the products covered herein for the equipments listed in Paragraph (2), even for the following purposes, be sure to observe the precautions given in Paragraph (2). Never use the products for the equipments listed in paragraph (3) even for the following purposes.

• Office electronics • Instrumentation and measuring equipments • Machine tools

• Audiovisual equipments • Home appliances • Communication equipments other than for trunk lines

(2) When using the products covered herein for the following equipments, which demand very high reliability, customers should consider to ensure the reliability and the safety of the equipments and the overall system on their responsibility by talking in the fail-safe design, redundant-design and other appropriate measures.

• Control and safety devices for airplanes, trains, automobiles, and other transportation equipments

• Mainframe computers • Traffic control systems • Gas leak detectors and automatic cutoff devices

• Disaster prevention and security equipments • Other safety devices and safety equipments, etc.

(3) Do not use the products covered herein for the following equipments which demands extremely high performance in terms of functionality, reliability, or accuracy.

• Aerospace equipments • Communications equipments for trunk lines

• Control equipments for the nuclear power industry • Medical equipments related to life support, etc.

(4) Please contact a sales representative of the company for all queries and comments regarding the interpretation of the above three Paragraphs.

○ Among the products covered herein, those come under the strategic materials provided by Foreign Exchange Control Law need to be approved based on the same law when exported.

○ Please contact a sales representative of the company for all queries regarding the products covered herein before taking further actions.

INTRODUCTION

The SM6010 series of single chip 16 bit microcomputer utilizes Sharp proprietary CPU SM6000CPU which allows use of same mnemonics by all programs written for the series microcomputers: easier and effective program transport.

This manual is divided into the following sections.

- General description
- Functional description
- Instruction set (common to SM6000 series devices)

Also refer to separate document **SM6000 C Compiler Software User's Manual.**

Contents

General specifications of SM6010

(1) CPU

- General purpose registers: 16-bit x 16
- 62 basic instructions, bit manipulation instructions suitable for controlling, bit transfer instructions, bit branch instructions

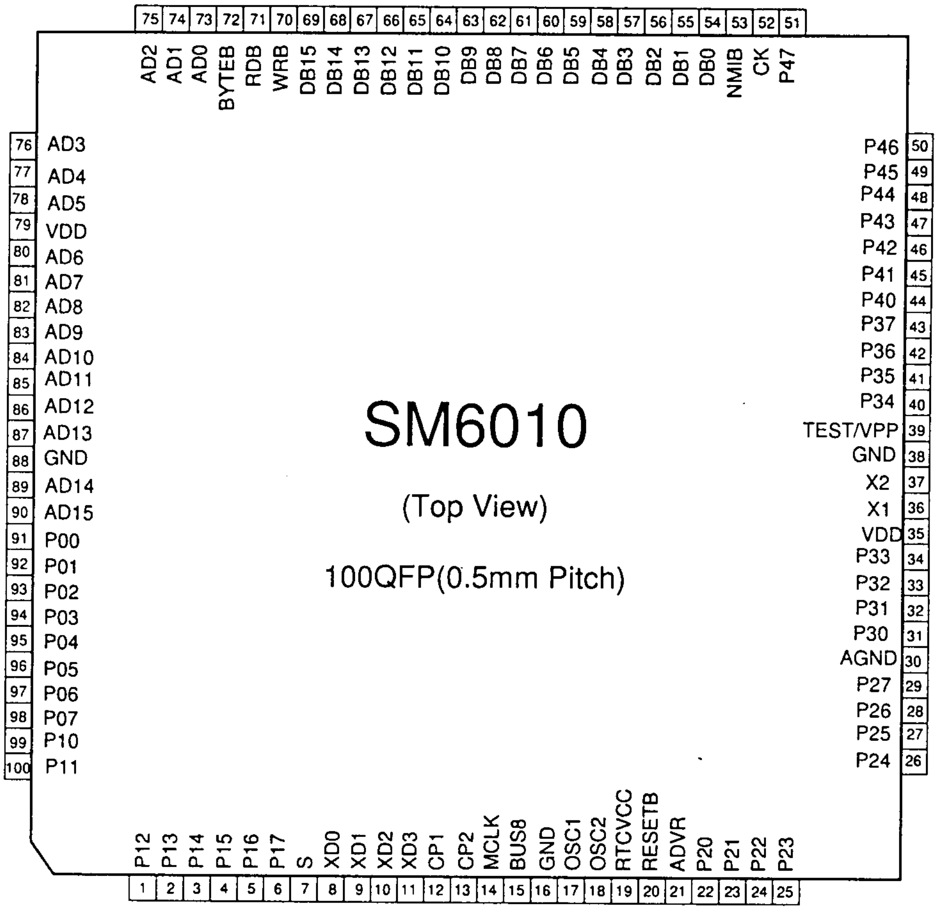
  High speed multiplication and division instructions (16 bits x 16 bits, 16 bits ÷ 16 bits, 32 bits ÷ 16 bits)
- 10 addressing modes
- 16 Mbytes of address space
- An interrupt request starts a high performance automatic data transfer (DTS). Appropriate settings of interrupts and registers enable hardware automatic data transfer. Various functions can be operated successively and the resultant data can also successively be stored.
- Selectable system clocks divided by 2 up to 16 main clock for low power operation.

Pin Assignment

RESETB	IN	Reset
AD[15:0]	OUT	Address Bus[15:0]
DB[15:0]	I/O	Data Bus[15:0]
WRB	OUT	Write Enable: Low Active
RDB	OUT	Read Enable: Low Active
BYTEB	OUT	Byte Access: Low Active
X1	-	System Clock In
X2	-	System Clock Out
OSC1	-	Sub Clock In
OSC2	-	Sub Clock Out
TEST	IN	Test Mode
ADVR	-	Reference VDD for A/D converter
AGND	-	GND for A/D Converter
NMI	IN	Non Maskable Interrupt
BUS8	IN	Byte Boot Selection
MCLK	OUT	LCD display frame signal
CP1	OUT	Display Data Latch Pulse and Scan Signal Transfer Clock Signal
CP2	OUT	Display Data Shift Clock to LCD Display
S	OUT	Scan Line Start Pulse to LCD Display
XD[3:0]	OUT	Data to LCD Display
P0[7:0]/ADDR[23:16]	I/O	Port 0 or Address Bus[23:16]
P1[7]/ADDR[24]	I/O	Port1[7]or Address Bus[24]
P1[6]/LCDCTL	I/O	Port1[6]or LCDCTL
P1[5]/LCDCTL	I/O	Port1[5]or PWMOUT
P1[4:3]	I/O	Port1[4:3]
P1[2]/SCK	I/O	Port1[2] or SCI Clock

P1[1]/RXD	I/O	Port1[1] or SCI Recciving Port
P1[0]/TXD	I/O	Port1[0] or SCI Transmitting Port
P2[7:0]/AIN[7:0]	IN	Port2 or Analog Inputs
P3[7:0]	I/O	Port3[7:0]
P4[7:0]/INT[7:0]	I/O	Port4 or External Interrupt Inputs
GND:3	-	Ground
VCC:3	-	Power
RTCVCC:1	-	Power for Real Time Clock

Pin layout


SM6010
(Top View)
100QFP(0.5mm Pitch)
1 P12
2 P13
3 P14
4 P15
5 P16
6 P17
7 S
8 XD0
9 XD1
10 XD2
11 XD3
12 CP1
13 CP2
14 MCLK
15 BUS8
16 GND
17 OSC1
18 OSC2
19 RTCVCC
20 RESETB
21 ADVR
22 P20
23 P21
24 P22
25 P23
26 P24
27 P25
28 P26
29 P27
30 AGND
31 P30
32 P31
33 P32
34 P33
35 VDD
36 X1
37 X2
38 GND
39 TEST/VPP
40 P34
41 P35
42 P36
43 P37
44 P40
45 P41
46 P42
47 P43
48 P44
49 P45
50 P46
51 P47
52 CK
53 NMIB
54 DB0
55 DB1
56 DB2
57 DB3
58 DB4
59 DB5
60 DB6
61 DB7
62 DB8
63 DB9
64 DB10
65 DB11
66 DB12
67 DB13
68 DB14
69 DB15
70 WRB
71 RDB
72 BYTEB
73 AD0
74 AD1
75 AD2
76 AD3
77 AD4
78 AD5
79 VDD
80 AD6
81 AD7
82 AD8
83 AD9
84 AD10
85 AD11
86 AD12
87 AD13
88 GND
89 AD14
90 AD15
91 P00
92 P01
93 P02
94 P03
95 P04
96 P05
97 P06
98 P07
99 P10
100 P11

Dimensions

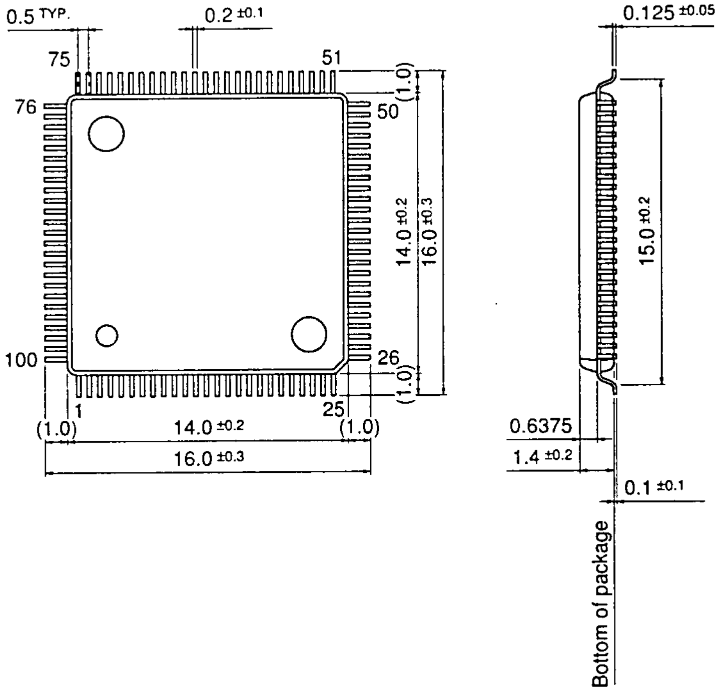


Fig. 1-5 Dimensions (0.5 mm pitch, LQFP100-P-1414)

1 Electrical characteristics

Absolute maximum ratings

Parameter	Symbol	Condition	Limits	Unit
Supply voltage	V_{DD}		-0.3~+6.5	V
Input voltage	V_I		-0.3~V_{DD}+0.3	V
Output voltage	V_O		-0.3~V_{DD}+0.3	V
Maximum output current	I_{OH}	Output pins except for DA0 and DA1	-2	mA
	I_{OL1}	$P3_0$, $P3_3$, $P4_{11}$	2	mA
	I_{OL2}	Output pins except for $P3_0$, $P3_3$, $P4_{11}$, DA0 and DA1		mA
Total output current	ΣI_{OH}	Output pins except for DA0 and DA1	-30	mA
	ΣI_{OL1}	$P3_0$, $P3_3$, $P4_{11}$	30	mA
Operating temperature	T_{OPR}	Output pins except for $P3_0$, $P3_3$, $P4_{11}$, DA0 and DA1	-20~+70	℃
Storage temperature	T_{STG}		-50~+150	℃

Operating conditions

Parameter	Symbol	Condition	Limits	Unit	Remarks
Supply voltage	V_{DD}		2.5~5.5	V	
			2.5~V_{DD}	V	
A/D supply voltage	ADVR		2.5~V_{DD}	V	
Analog GND level	AGND		2.5~V_{DD}	V	
RTC supply voltage	RTCCVcc	V_{DD}=4.5V~5.5V	1.8~V_{DD}	V	
		V_{DD}=4.5V~5.5V	2.7~V_{DD}	V	
Minimum instruction execution time	t_{SYS}	V_{DD}=4.5V~5.5V	133	ns	
		V_{DD}=2.5V~5.5V	200	ns	
Maximum system clock frequency	f_{SMAX}	V_{DD}=4.5V~5.5V	15	MHz	1
		V_{DD}=2.5V~5.5V	10	MHz	1
Maximum main clock frequency	f_{MAX}	V_{DD}=4.5V~5.5V	30	MHz	
		V_{DD}=2.5V~5.5V	20	MHz	
Operating temperature	T_{OPR}		-20~+70	℃	

Note 1: When using divided-by-1/2 main clock as the system clock.

DC characteristics

(1) DC characteristics

（ＶＤＤ＝２．７Ｖ～３．３Ｖ、 Ｔａ＝－２０～＋７０℃)

Parameter	Symbol	Test Condition	Limits			Unit	Remark
			MIN.	TYP.	MAX.		
Input Voltage	VIH1		0.7VDD		VDD	V	1
	VIL1		0		0.3VDD		1
	VIH2		VDD-0.5		VDD		2
	VIL2		0		0.5		2
Input Current	IH1	VIN=VDD,VDD=3V			10	μA	3
	IL1	VIN=0V,VDD=3V			-10		3
Output Voltage	VOH1	IOH=-1.0mA,VDD=3V	VDD-0.5			V	4
	VOL1	IOL= 1.0mA,VDD=3V			0.5		5
A/DConverter Resolution		ADVR=VDD=3V		10		bits	
Differential Linear Error				±1	±2.5	LSB	
Linear Error				±3	±5	LSB	
Total Error				±4	±6	LSB	
Operating Current	IDD			25	50	mA	6
	ICC			200	400	μA	7
	IHLT1			5	10	mA	8
	IHLT2			10	20	μA	9
	ISTOP			1.0	10	μA	10

Note 1 Applicable Pins P00-P07,P10,P13-P17,P20-P27,P30-P37,DB0-DB15,AD0-AD15,X1,OSC1

Note 2 Applicable Pins P11,P12,P40-P47,RESETB,TESTB,BUS8,NMIB

Note 3 Applicable Pins P00-P07,P10-P17,P20-P27,P30-P37,P40-P47,DB0-DB15,AD0-AD15,RESETB,NMIB,BUS8

Note 4 Applicable Pins P00-P07,P10-P17,P30-P37,P40-P47,DB0-DB15,AD0-AD15,S,XD0-XD3,CP1,CP2,MCLK
CKOUT,RDB,WRB,BYTEB

Note 5 Applicable Pins P00-P07,P10-P17,P30-P37,P40-P47,DB0-DB15,AD0-AD15,S,XD0-XD3,CP1,CP2,MCLK
CKOUT,RDB,WRB,BYTEB

Note 6 No Load VDD=3V, Main Clock 20MHz

Note 7 No Load VDD=3V, RTCVCC=3V, Sub Clock 32kHZ

Note 8 No Load VDD=3V, Main Clock 20MHz with peripheral blocks inactive

Note 9 No Load VDD=3V, RTCVCC=3V,Sub Clock32kHZ with RTC inactive

Note10 No Load VDD=3V, ADVR=DRVR=GND,RTCVCC=1.8V with constant input level(Inc.OSC0)、25℃

(2)DC characteristics

(ＶＤＤ＝４．５Ｖ～５．５Ｖ、　Ｔａ＝－２０～＋７０℃)

Parameter	Symbol	Test Condition	Limits			Unit	Remark
			MIN.	TYP.	MAX.		
Input Voltage	VIH1		0.7VDD		VDD	V	1
	VIL1		0		0.3VDD		1
	VIH2		VDD-0.5		VDD		2
	VIL2		0		0.5		2
Input Current	IH1	VIN=VDD,VDD=5V			10	μA	3
	IL1	VIN=0V,VDD=5V			-10		3
Output Voltage	VOH1	IOH=-2.0mA,VDD=3V	VDD-0.5			V	4
	VOL1	IOL= 2.0mA,VDD=3V			0.5		5
A/DConverter Resolution		ADVR=VDD=5V		10		bits	
Differential Linear Error				±1	±2.5	LSB	
Linear Error				±3	±5	LSB	
Total Error				±4	±6	LSB	
Operating Current	IDD			70	130	mA	6
	ICC			250	500	μA	7
	IHLT1			15	30	mA	8
	IHLT2			10	20	μA	9
	ISTOP			1.0	20	μA	10

Note 1 Applicable Pins P00-P07,P10,P13-P17,P20-P27,P30-P37,DB0-DB15,AD0-AD15,X1,OSC1

Note 2 Applicable Pins P11,P12,P40-P47,RESETB,TESTB,BUS8,NMIB

Note 3 Applicable Pins P00-P07,P10-P17,P20-P27,P30-P37,P40-P47,DB0-DB15,AD0-AD15,RESETB,NMIB,BUS8

Note 4 Applicable Pins P00-P07,P10-P17,P30-P37,P40-P47,DB0-DB15,AD0-AD15,S,XD0-XD3,CP1,CP2,MCLK CKOUT,RDB,WRB,BYTEB

Note 5 Applicable Pins P00-P07,P10-P17,P30-P37,P40-P47,DB0-DB15,AD0-AD15,S,XD0-XD3,CP1,CP2,MCLK CKOUT,RDB,WRB,BYTEB

Note 6 No Load VDD=5V,Main Clock 30MHz

Note 7 No Load VDD=5V,RTCVCC=3V, Sub Clock32kHZ

Note 8 No Load VDD=5V, Main Clock 30MHz with peripheral blocks inactive

Note 9 No Load VDD=5V,RTCVCC=3V, Sub Clock 32kHZ with RTC inactive

Note10 No Load VDD=5V,ADVR=DRVR=GND,RTCVCC=3V with constant input level(Inc.OSC0)、25℃

AC Characteristics(External Memory Access)

VDD=4.5～5.5v、Ta=-20～70℃

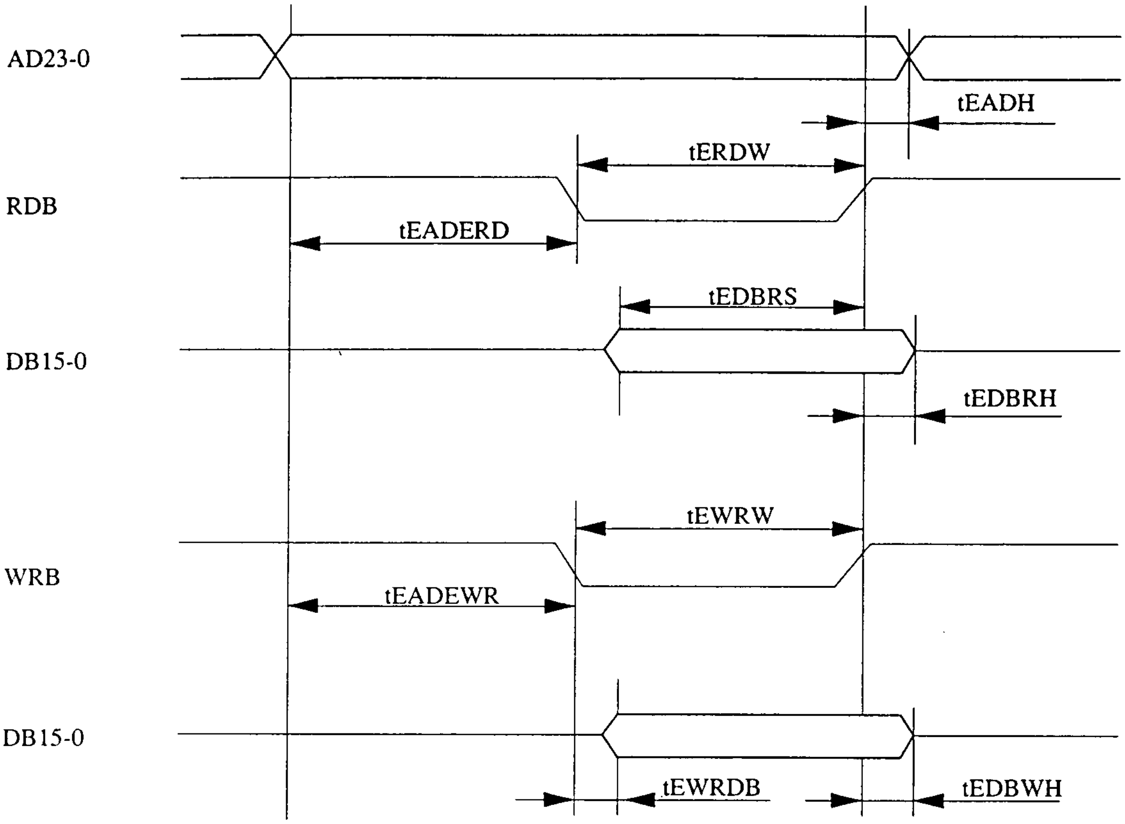
SYMBOL	DESCRIPTION	Min	Typ	Max	Units	Notes
tEADERD	ADDRESS to ERDB LOW		tCKCYC	tCKCYC+20	nS	1
tERDW	ERDB LOW period	tCKCYC-20		tCKCYC	nS	2
tEDBRS	REDA DATA setup			tCKCYC/2-20	nS	
tEADH	ERDB to ADRESS Hold	0	10		nS	
tEDBRH	ERDB to EDB Hold	0			nS	
tEADEWR	ADDRESS to EWRB LOW		tCKCYC	tCKCYC+10	nS	
tEWR	EWRB LOW Period	tCKCYC-20		tCKCYC	nS	2
tEWRDB	EWRB LOW to DATA Valid			20	nS	
tDBWH	EWRB HIGH to DATA Hold	10			nS	
					nS	
					nS	
					nS	

VDD=2.7～5.5v、Ta=-20～70℃

SYMBOL	DESCRIPTION	Min	Typ	Max	Units	Notes
tEADERD	ADDRESS to ERDB LOW		tCKCYC	tCKCYC+40	nS	1
tERDW	ERDB LOW period	tCKCYC-20		tCKCYC	nS	2
tEDBRS	REDA DATA setup			tCKCYC/2-40	nS	
tEADH	ERDB to ADRESS Hold	0	10		nS	
tEDBRH	ERDB to EDB Hold	0			nS	
tEADEWR	ADDRESS to EWRB LOW		tCKCYC	tCKCYC+20	nS	
tEWR	EWRB LOW Period	tCKCYC-20		tCKCYC	nS	2
tEWRDB	EWRB LOW to DATA Valid			40	nS	
tDBWH	EWRB HIGH to DATA Hold	5			nS	
					nS	
					nS	
					nS	

1.tCKCYC: 1/(Main Clock/2)
2.tCKCYC will be required when 3-cycle bus mode

AC Characteristics



Chapter 2 SM6000CPU

SM6000CPU is a 16 bit CPU core. Its architecture inclues 16 bits by 16 channels general purpose register. This architecture, unlike accumulator architecture, easily compatible with high-level language like C. Furthermore it is provided with simple instruction set which facilitates high speed operation.

SM6000CPU has the following features.

■ General purpose register

The CPU core contains 16 bits by 16 channels general purpose register. It can be accessed in unit of byte (8 bits) and long word (32 bits) as well as in unit of 16 bit-word . For further information, refer to **2.2 General purpose registers and SP and FP.**

■ 62 basic instructions

- High speed multiplication and division instructions (signed/unsigned, 16 × 16, 16÷16, 32÷16).
- Bit manipulation instruction, bit transfer instruction, bit operation instruction and bit branch instruction, suitable for controlling application.
- Interrupt operation (software interrupt) upon execution of an instruction
- 10 addressing modes suitable for simple instruction set

For further information, refer to **Chapter 8 Instruction set.**

■ 16 Mbyte address space

SM6000CPU core has 16 Mbyte memory space to be divided into segments of 65 Kbytes. The segments can be accessed with three data pointers which can be used for generating working address and other purposes. For further information, refer to **2.1 Address space** and **2.3 Dedicated register.**

■ High-speed operation

The operation speed at 10 MHz (main clock 20 MHz) is as shown below:

- Operation between 8 bit registers, and 16 bit registers 0.2μs
- 16 bits × 16 bits multiplication 1.0μs
- 16 bits÷16 bits division 1.8μs
- 32 bits÷16 bits division 3.9μs

■ High performance interrupt using DTS

When an interrupt request is generated, DTS can be operated. DTS is one of automatic transfer functions. When the data required for the desired process is stored in the DTS address, the data is automatically transferred upon interrupt request. DTS is also called automatic transfer server. In addition to data transfer between memories by using DTS, by specifying the control register or the like as the transfer destination, various functional blocks can be automatically transferred. For further information, refer to **3.6 DTS.**

■ Easy generation/release of automatic variable area

Automatic variable area necessary to use high-level language like C can be easily generated/released by executing instructions (LINK/UNLINK). The generated automatic variable area can be effectively accessed thruogh indirect addressing using frame pointer FP. For further information, refer to **2.2 General purpose registers SP and FP.**

■ Efficient access with SFR

SFR is abbreviation of Special Function Register. The SFR area consists of a 256-word (512 byte) addressable space which can be most efficiently accessed. Therefore, with SM6010, the I/O registers and control registers to be repeatedly accessed are located in this area, allowing the user to program effectively. Portions (equivalent of 128 words) of memory space can also be used as SFR field. This memory space assignment can be changed from the user program to keep RAM and external memory to be efficiently accessed. For further information, refer to **2.3 Dedicated register.**

■ Data type

Various types of data are available: some types are suitable for data processing and the others for controllers. For further information, refer to **2.4 Data type.**

■ Low power consumption

To achieve high speed operation of the microcomputer with minimum power consumption, the settings of the microcomputer can be fine tuned by taking into considerations the processing conditions. SM6000CPU features 2 standby modes, variable system clocks and main/sub clock selection.

For further information on standby mode, refer to **3.7 Standby fucntion** and for variable system clock, **3.2 Clock system.**

Note:

- *SM6010 does not use main/sub clock select function because it has no built-in sub clock generator.*

This chapter focuses on the architecture, data type and instruction system together with its addressing modes, of the SM6000CPU. For description on control of the CPU and CPU peripherals, refer to Chapter 3; and on I/O ports and function controls, Chapters 4 to 10.

2.1 Address space

SM6000CPU supports up to 16 Mbytes of address space. The address space is divided into 256 segments (segment 0 - segment 255), each 64 kbytes.

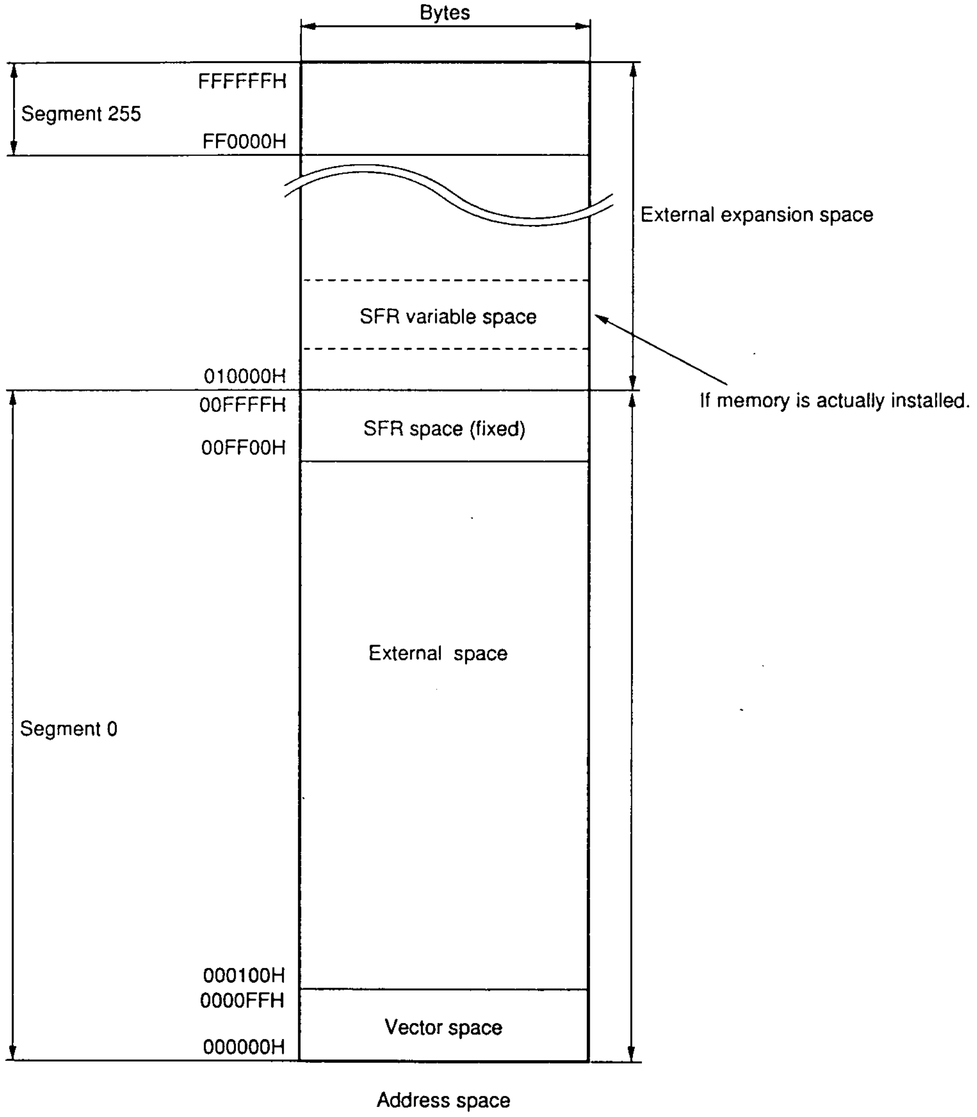


Fig. 2-1 SM6010 address space

Note:

- *In the memory maps like this, unless otherwise noted, the highest address is always appears at the top or leftmost of the map.*

2.2 General purpose registers and SP and FP

General purpose registers are total 16 (R0-R15) each 16 bits, and support both byte (8 bits) access and long word access (32 bits) as corresponding instructions are used. The register R14 can also serve as a frame pointer FP, and R15 as stack pointer SP. **Fig. 2-2** shows the arrangement of general purpose registers.

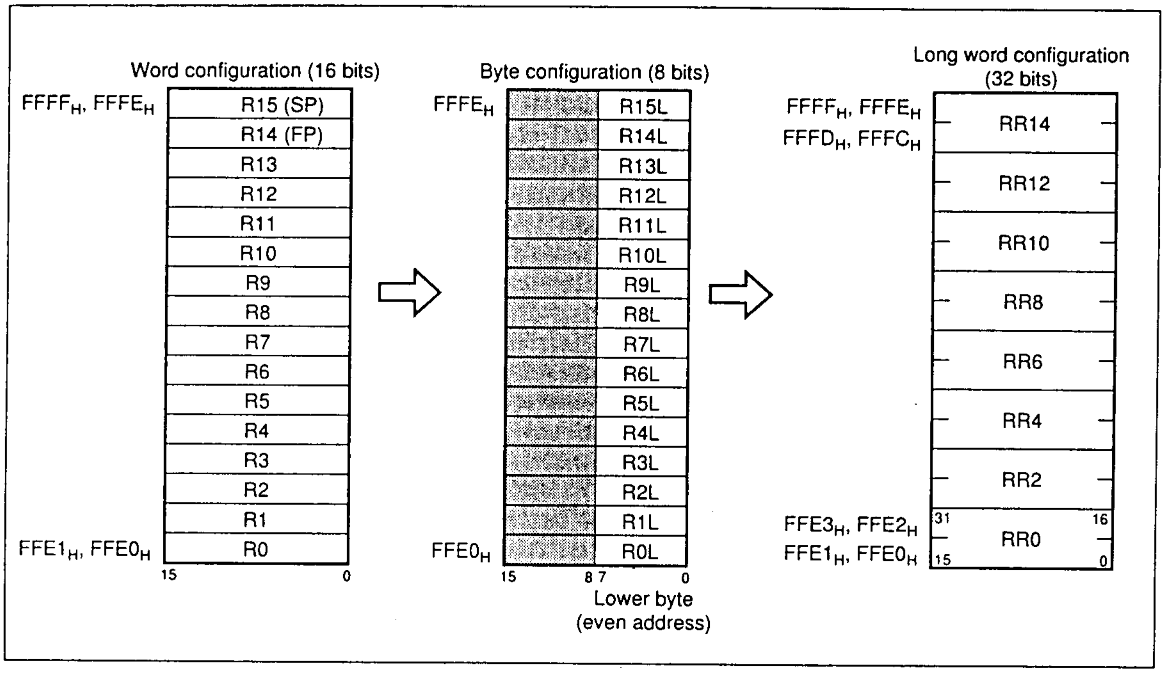


Fig. 2-2 General purpose register

2.2.1 Stack area and stack pointer SP

The stack pointer SP is a memory pointer into which the content of the program counter PC is saved upon subroutine call or generation of an interrupt. The stack pointer SP is a 16 bit register and also serves as the general purpose register R15. Before the user program can use a stack area, it must set the desired value in the stack pointer SP. The data pointer DP2 also has the setting to specify in which segment the stack frame is located. (Refer to **2.3 Dedicated register**)

When a subroutine call or push/pop instruction is associated with stacking operation, the stack pointer SP and data pointer DP2 together generates a 24 bit effective address which is the start address of the stack frame.

Since a 24 bit effective address is generated upon a stacking operation, the stack area can be located anywhere in 16 Mbyte address space within an area where memory is installed. However, only even address can be set in the stack pointer SP.

Note:

The data set into the stack pointer SP must always be even.

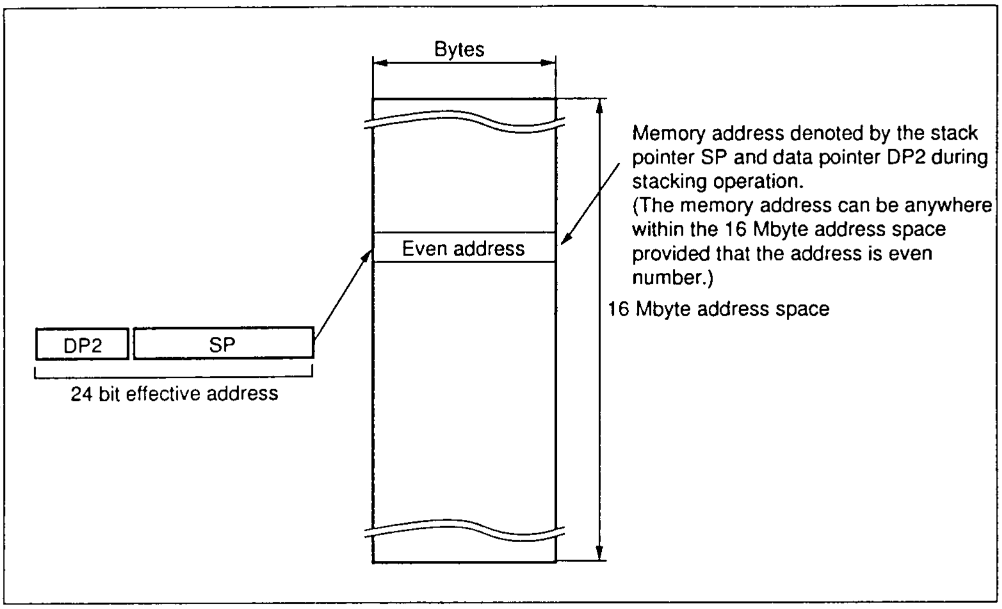
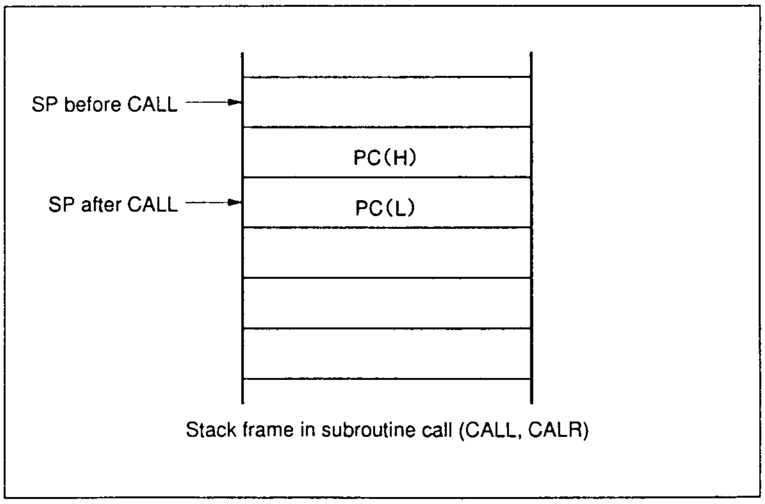

Bytes
Memory address denoted by the stack pointer SP and data pointer DP2 during stacking operation.
(The memory address can be anywhere within the 16 Mbyte address space provided that the address is even number.)
Even address
16 Mbyte address space
DP2
SP
24 bit effective address

Fig. 2-3 Stack pointer SP and data pointer DP2

A group of data to be saved (stored) simultaneously into the stack is called a stack frame. The type and quantity of a stack frame depends on processing. There are 4 stack frame types:

- Subroutine call within a segment

The data of the program counter PC (equivalent of 1 word) is saved into the stack area. Execution of CALL or CALR instruction leads to subroutine call within a segment: the content of the program counter PC will be saved into the stack area, and the contents of the stack pointer SP is decremented by 2 (see **Fig. 2-4**). When RET instruction is executed and specified condition is met, the content saved in the stack area is returned back to the program counter PC and the content of the stack pointer SP is incremented by 2.



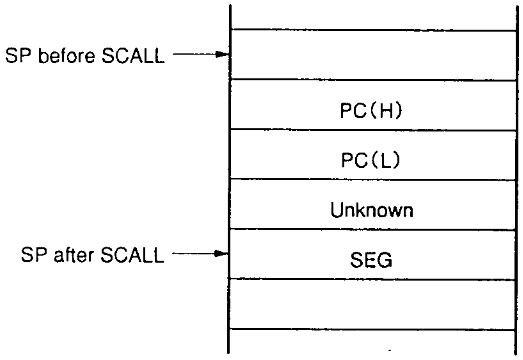
Stack frame in subroutine call (CALL, CALR)

Fig. 2-4 Subroutine call within a segment

- Subroutine call exceeding a segment

Into the stack area the data of program counter PC and register SEG (27 words) are saved. (For further information on program counter PC and register SEG, refer to **2.3.1 Program counter PC and segment register SEG.**)

When SCALL instruction is executed, subroutine call across segments takes place: the content of the program counter PC and register SEG are saved into the stack area and the contents of the stack pointer SP is decremented by 4 (see **Fig. 2-5**). When SRET instruction is executed and specified condition is met, the contents saved in the stack area are returned back to the registers and the content of the stack pointer SP is incremented by 4.



Stack frame in subroutine call (SCAL)

Fig. 2-5 Subroutine call across segments

- Upon occurrence of interrupt

Data of the program counter PC and register SEG, and data (3 words) of processor status word (register PSW) are saved into the stack area (see **Fig. 2-6**).

Upon occurrene of an interrupt, the contents (whether the size is larger than a segment or not) of program counter PC, register SEG and register PSW are saved into the stack area and the content of the stack pointer SP is decremented by 6.

When IRET instruction is executed and the specified condition is met, the contents saved in the stack area are returned back to the registers and the content of the stack pointer SP is incremented by 6.

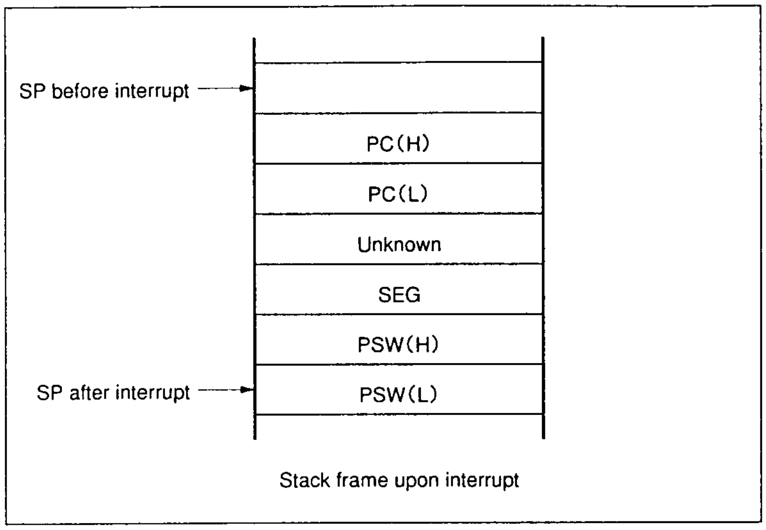


Fig. 2-6 Stack frame upon occurrence of an interrupt

- Upon execution of instruction (PUSH/POP)

Data of 1 word is saved into the stack area.

Upon execution of PUSH instruction, the word data specified by the operand (can be used only by SFR addressing) is stored in the stack area. At the same time, the content of stack pointer SP is decremented by 2 (see Fig. 2-7).

Upon execution of POP instruction, the word data stored in the stack area is stored in the SFR specified by the operand. At the same time, the content of stack pointer SP is inremented by 2.

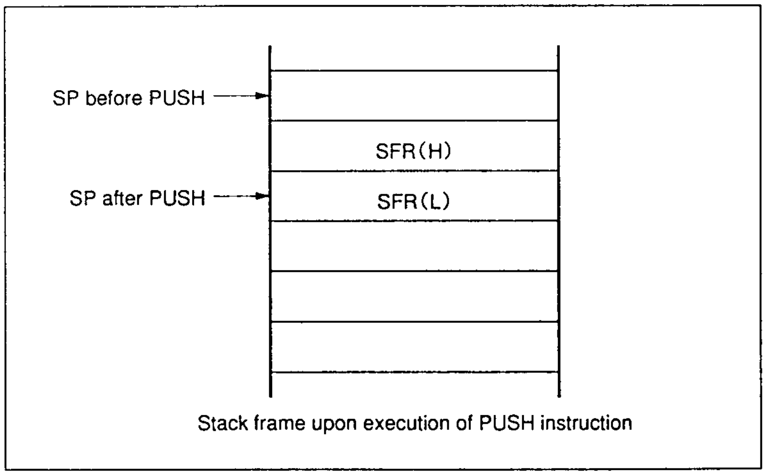


Fig. 2-7 Stack frame upon execution of instructions (PUSH/POP)

2.2.2 Frame pointer FP

SM6010 has LINK/UNLINK instructions to generate/release the automatic variable area. It also has the frame pointer FP to effectively access the generated automatic variable area. This configuration helps to effectively generate the automatic variable area by using high-level language like C.

LINK instruction secures the specified number of bytes on the stack frame, which can be set as variable area. Stack frames are secured in units of word and up to 254 bytes of variable area can be generated with a single LINK instruction. Upon execution of LINK instruction, the microcomputer generates the variable area in the order shown below.

(1) Saves the current content of the frame pointer FP to the memory addressed by the stack pointer SP.

(2) Decrements the contents of the stack pointer SP by 2.

(3) Stores the decremented content of the stack pointer SP into the frame pointer FP. This operation sets the new frame address in the frame pointer FP.

(4) The size specified upon execution of LINK instruction is subtracted from the content of stack pointer SP, securing the stack area (variable area) whose size is specified by LINK instruction.

The variable area secured by the LINK instruction can be accessed by the user program through the register (R14) indirect addressing mode.

UNLINK instruction releases the variable area. UNLINK instruction causes the microcomputer to follow the following steps to release the variable area.

(1) Stores the current content of the frame pointer FP into the stack pointer SP to release the variable area. Now the stack pointer SP indicates the address at which the content of the frame pointer FP is stored.

(2) Stores the content at the address denoted by the stack pointer SP into the frame pointer FP, restoring the previous FP content.

(3) Increments the content of the stack pointer SP by 2.

Now the status of the microcomputer is the same as that before securing the variable area. That is, releases the variable area.

Fig. 2-8 shows status of each pointer when the following instruction is executed.

```
LINK #10
```

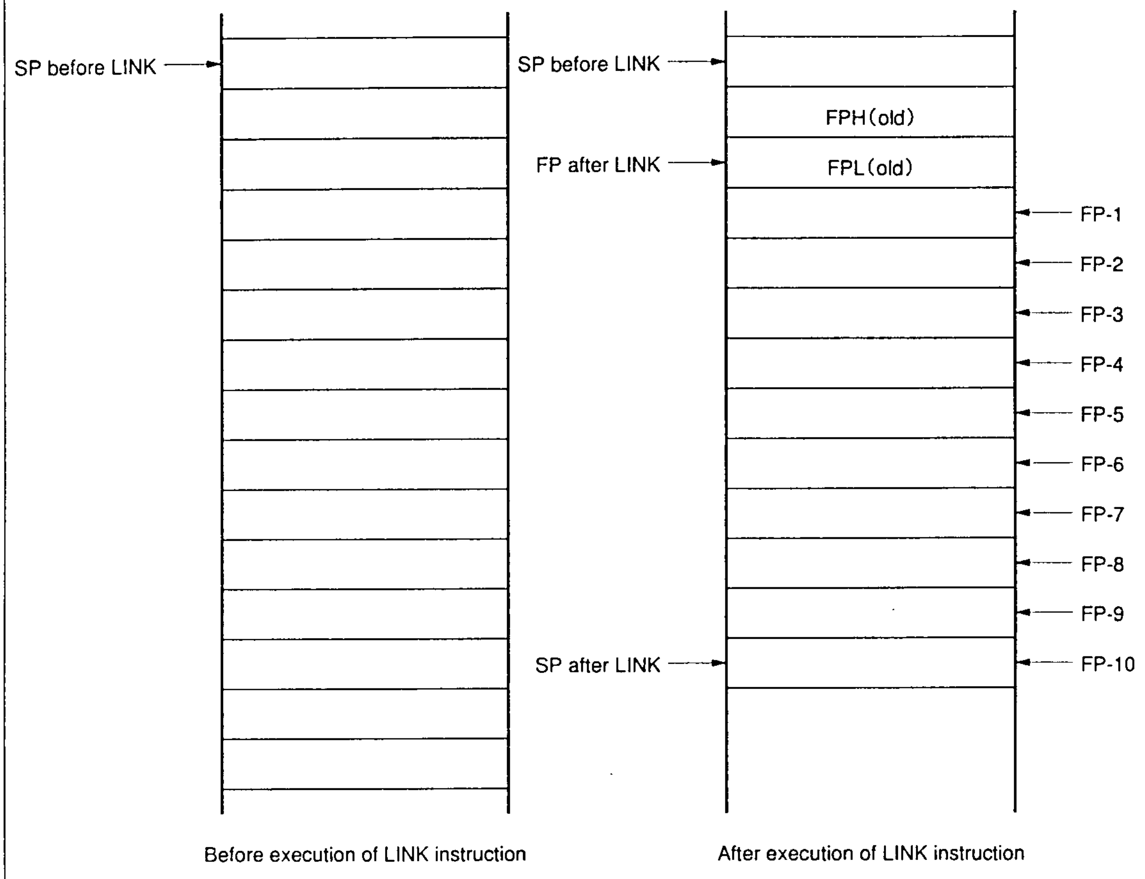


Fig. 2-8 Status of pointers upon execution of LINK instruction

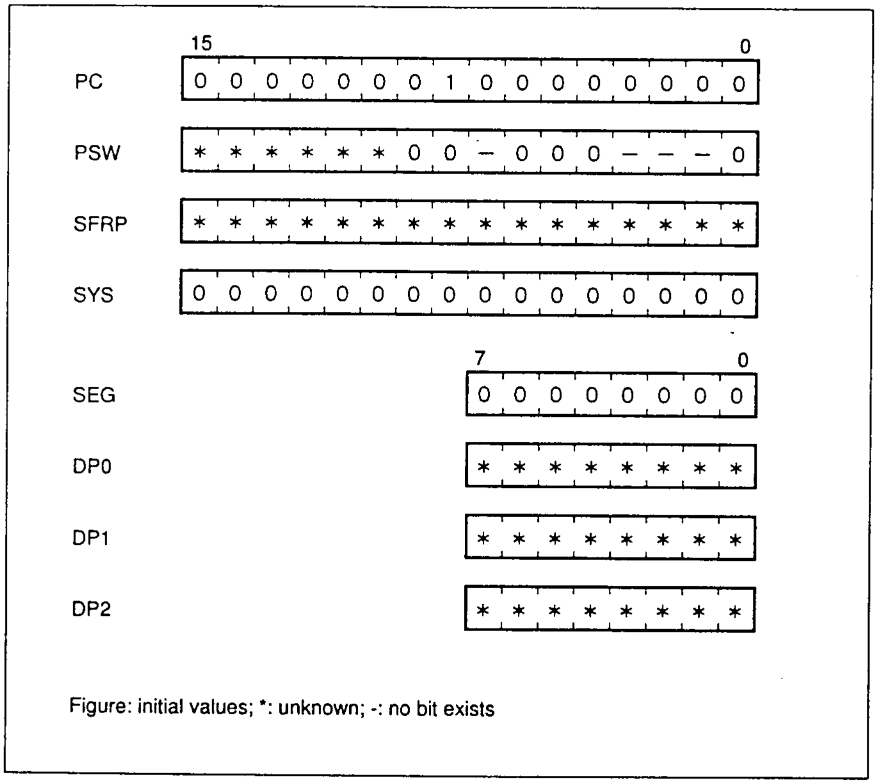
The general purpose register R14 is not used as a frame pointer and may be used as a data storage device if the program does not use LINK and UNLINK instructions.

2.3Dedicated register

With SM6010, the following dedicated registers are available.

- Program counter PC
- Segment register SEG
- Data pointers DP0, DP1 and DP2
- Processor status word PSW
- System configuration register SYS
- SFP pointer (register SFRP)
- Special function register SFR (SFR area)

All these registers except for the program counter PC are actually located in the fixed SFR area and can be accessed in the same way as for other registers in the SFR area. Figure 2-9 shows the configuration of the dedicated registers. For further information on SFR area, refer to **2.3.5 SFR area and SFR pointer.**



	15															0
PC	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
PSW	*	*	*	*	*	*	0	0	-	0	0	0	-	-	-	0
SFRP	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
SYS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									7							0
SEG									0	0	0	0	0	0	0	0
DP0									*	*	*	*	*	*	*	*
DP1									*	*	*	*	*	*	*	*
DP2									*	*	*	*	*	*	*	*

Figure: initial values; *: unknown; -: no bit exists

Fig. 2-9 Configuration of dedicated register

2.3.1 Program counter PC and segment register SEG

The program counter PC is a 16 bit register. It indicates the address of the instruction which the CPU executes next. All CPU instructions are word with LSB set as 0. Therefore, LSB set at 1 will cause an address error interrupt, indicating CPU error (jumping address of JMP instruction is odd number address).

The segment register SEG is a pointer which manages the 16 Mbyte address space in units of 64 Kbytes. The content of the segment register SEG and that of the program counter PC together generates 24 bit effective address: SEG upper and PC lower. After the system reset, the SM6010 will start the program at address 00100H.

A carry from program counter PC is not reflected on the segment value. That is, the next segment cannot be processed unless the instruction for that segment is executed. Therefore, unless the segment is changed, the user program loops within the current segment. The content of the segment register SEG is overwritten with current segment data by new instruction (SCALL, SJMP, SRET).

Fig. 2-10 shows configuration of program counter PC and segment register SEG.

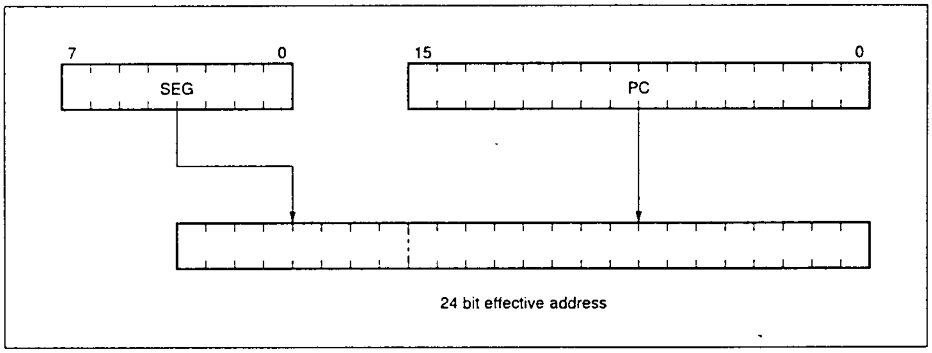


Fig. 2-10 Program counter PC and segment register SEG

The segment register SEG is located in the SFR area and can be read and written through a program. To change segments, do not directly update the content of the segment register SEG by using MOV instruction, but use SCALL, SJMP or SRET instruction.

The segment register SEG is a pointer dedicated for instruction code. To access data spanning segments, use data pointers DP0, DP1 and DP2 described below.

2.3.2 Data pointers DP0-DP2

As described previously the segment register SEG is a segment pointer used in conjunction with the program counter PC. In contrast, data pointers DP0-DP2 are used as segment pointer for accessing data.

- Data pointer DP0

Used to specify a segment to access data in the 16 Mbyte address space.In the direct accessing mode (DA addressing), DP0 is referenced as segment value.

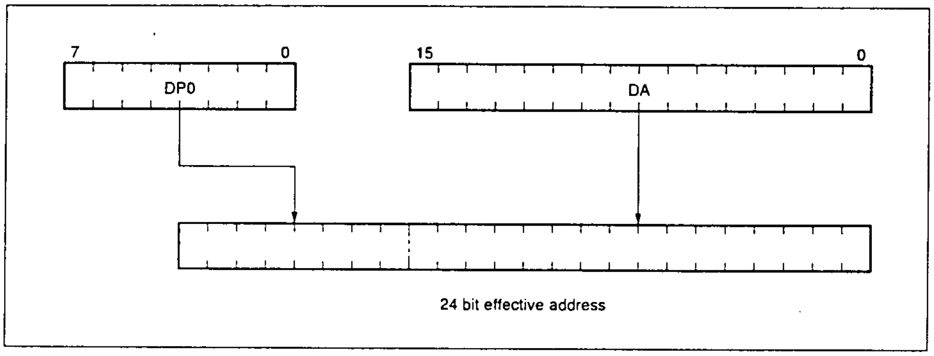


Fig. 2-11 Data pointer DP0

- Data pointer DP1

Used to specify a segment to access data in the 16 Mbyte address space. When indirect accessing is made through register (including predecrement, post-increment) using registers R0-R7, or through index addressing, DP1 is used as the segment value.

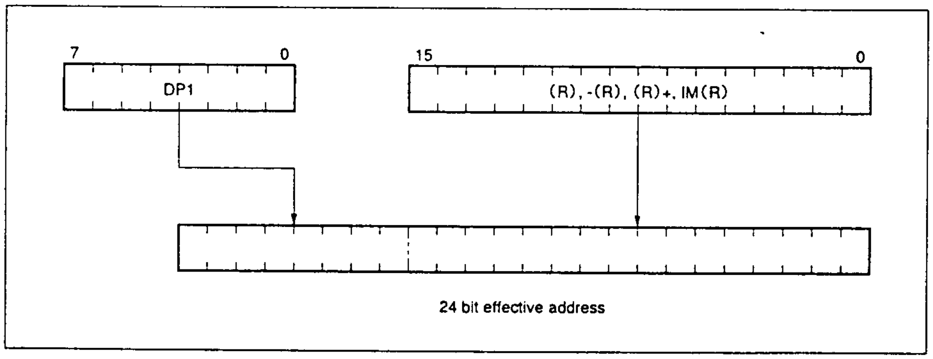


Fig. 2-12 Data pointer DP1

- Data pointer DP2

Used to specify a segment to access data in the 16 Mbyte address space. When indirect accessing is made through register (including predecrement, post-increment) using registers R8-R15, or through index addressing, DP2 is used as the segment value. DP2 is used as the segment pointer when stacking operation (at interrupt, at execution of call/push/pop instructions) is tried.

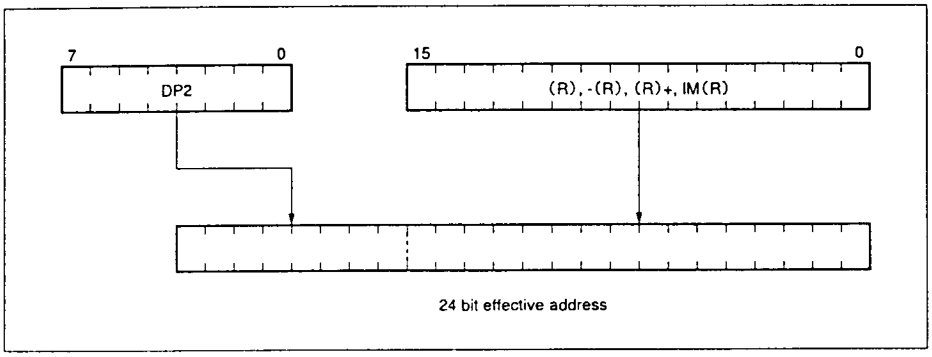


Fig. 2-13 Data pointer DP2

2.3.3 PSW: processor status word

The processor status word (register PSW) is a 16 bit register containing the flag which shows the processing status of the CPU, and other information. The PSW is located in the SFR area. The PSW can be directly accessed and a particular bit in the PSW can be accessed by executing the dedicated instruction.

Register	Symbol	Bit length	Address	SFR
Processor status word	PSW	Word	00FFDEH	EFH

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W	-	-	-	R/W
Initial value	*	*	*	*	*	*	0	0	-	0	0	0	-	-	-	0
Bit symbol	C	Z	N	V	D11	U	I	D	-	PRIO			X	O	A	B

■ Bit 15: carry flag (C)

Set if an operation results in carry. Otherwise, cleared.

■ Bit 14: zero flag (Z)

Set if an operation results in zero. Otherwise, cleared.

■ Bit 13: negative flag (N)

This is the MSB of the result of an operation. The MSB is considered as a symbol.

■ Bit 12: overflow flag (V)

Set if an operation produces overflow. Otherwise, cleared.

■ Bit 11: reserved bit (D11)

Read/write bit and set according to operation result. The user program cannot use this bit for data storage.

■ Bit 10: bus request enable flag (U).

This is a bus request enable flag. Setting this bit to 1 enables to accept a bus release request from other bus master devices.

■ Bit 9: interrupt enable flag (I)

This is a maskable interrupt enable flag. Setting this bit to 1, by executing EI instruction or by writing into PSW *, accepts a maskable interrupt. Set at 0 (by DI instruction), this bit ignores a maskable interrupt request. The DI or EI instruction controls bit 9 more effectively than BMOV instruction.

■ Bit 8: DTS master enable flag (D)

This is a master enable flag for DTS operation. Setting this bit to 1 enables DTS operation. When set at 0, this bit inhibits DTS even if DTS selection bit is set at 1, allowing normal interrupt to be accepted.

■ Bits 6-4: interrupt priority set (PRIO)

These bits set the priority level of acceptable interrupts. Interrupts lower than the set level are not accepted.

■ Bit 3: BXOR instruction bit (X)

Data transferred by BXOR or BMOV instruction to this bit place is exclusive-ORed with bit B and the result is stored in the bit B place. Use of this instruction makes the user program simpler than when using BMOV which needs the same instruction bytes/cycles. Writing to register PSW * will not cause XOR. Reading this phantom register will be returned with 1.

■ Bit 2: BOR instruction bit (O)

Data transferred by BOR or BMOV instruction to this bit place is ORed with bit B and the result is stored in the bit B place. Use of this instruction makes the user program simpler than when using BMOV which needs the same instruction bytes/cycles. Writing to register PSW * will not cause XOR. Reading this phantom register will be returned with 1.

■ Bit 1: BAND instruction bit (A)

Data transferred by BAND or BMOV instruction to this bit place is ANDed with bit B and the result is stored in the bit B place. Use of this instruction makes the user program simpler than when using BMOV which needs the same instruction bytes/cycles. Writing to register PSW * will not cause AND operation. Reading this phantom register will be returned with 1.

■ Bit 0: bit flag (B)

This is a flag used for bit operation. This bit stored the result of bit operation performed with BXOR, BOR or BAND instruction.

***: Data transfer to register PSW by using MOV instruction.**

2.3.4 SYS: system configuration register

This register sets the operation mode of the CPU, bus timing of external access and bus width. SYS is located in SFR area and can be directly accessed.

Register	Symbol	Bit length	Address	SFR
System configuration register	SYS	Word	00FFDAH	EDH

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit symbol	M		BC	BS	CS	DIVSEL		MC	D7	D6	D5	D4	D3	D2	D1	D0

- Bits 15-14: CPU operation mode set (M) Not Used

Bit M		CPU operation mode	
0	0	Single chip mode (no external expansion)	
0	1	External expansion mode	Memory expansion mode (external access except for internal ROM, RAM and I/O)
1	0		Microprocessor mode (external access except for internal RAM and I/O)
1	1	Do not set	

SM6010 starts in the External expansion mode since the bit M is initialized to 10. The following two modes in which external expansion is possible are collectively referred to as "external expansion mode" in this manual.

- Memory expansion mode
- Microprocessor mode

An expanded area is called "external expansion area".

- Bit 13: bus cycle select (BC)

This bit selects the number of clocks used to access memory.(reset value:1)

Bit BC	Operation
0	Sets the external access bus to 2 clock cycles.
1	Sets the external access bus to 3 clock cycles.

■ Bit 11: operation clock select (CS)

This bit is used to change the system clock from the main clock to subclock, if subclock is available.

Bit CS	Operation
0	Selects the main clock.
1	Selects the subclock.

■ Bit 10-9: system clock division select (DIVSEL)

Bit DIVSEL		System clock selection
0	0	Divided-by-2 of the main clock
0	1	Divided-by-4 of the main clock
1	0	Divided-by-8 of the main clock
1	1	Divided-by-16 of the main clock

This bit is ignored if a subclock is selected as the system clock.

■ Bit 8: main clock stop (MC)

Bit MC	Operation
0	Enables main clock oscillator.
1	Stops main clock (active only when CS is set at 1.)

If the subclock is available and subclock is selected (bit SYS.CS is set at 1), setting bit MC at 1 stops the main clock.

■ Bit 7: reserved bits (D7-D0) Always set at "0" with SM6010.

Reserved for future expansion. Always set these bits at 0.

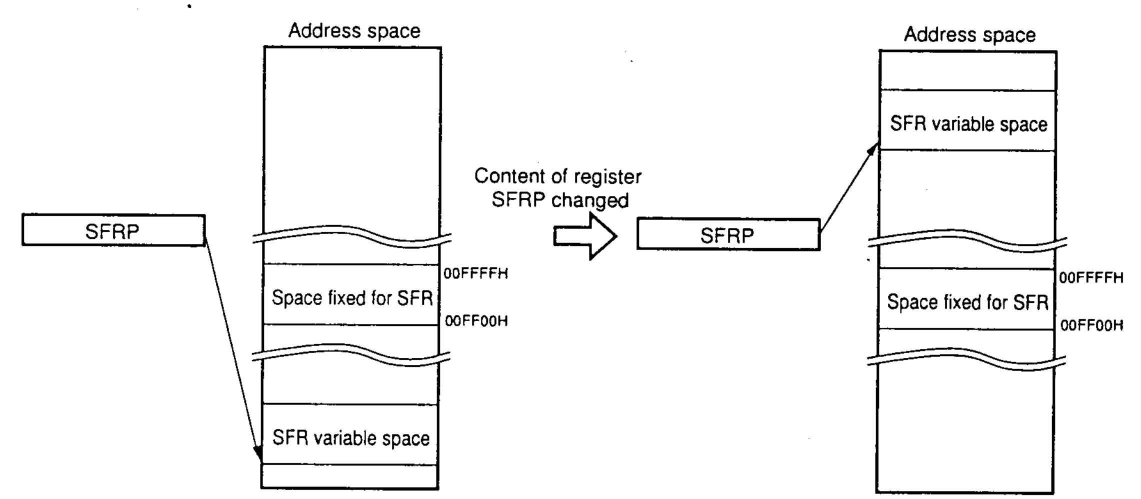
2.3.5 SFR area and SFR pointer (register SFRP)

SFR stands for special function register. The register occupies 256 words (512 bytes) location in the SFR area which is most effectively accessed. Therefore, frequently accessed memory, I/O and control registers are located in this area.

SFR addressing for data access lightens the ROM load and reduces the instruction execution cycles. With SM6010, 128 word space (80H-0FFH) in SFR area is assigned to I/O and control register as fixed SFR area. The remaining 128 word space (00H-7FH) is a variable SFR area and opened to the user program. SFR area can be efficiently accessed when 8 bit address is used in the SFR addressing mode. Many instructions support the SFR addressing.

Variable SFR area can be located at any space of 128 words within the address space, when memory is installed. Space allocation is through setting of SFR pointer (register SFRP).

By effectively using the SFR addressing, mode I/O and control register in the fixed SFR area can be accessed without considering in what segment the user program is running. By relocating the variable SFR area to segments, memory can be efficiently accessed.



SFR variable space can be located anywhere in the address space denoted by the content of register SFRP, except for the address space fixed for SFR. The space fixed for SFR cannot be changed by the content of SFRP.

Fig. 2-14 SFR area and address space

In SFR addressing mode, SFR area is accessed in units of word, up to 256 words (512 bytes). Accessing SFR area in units of byte is also possible but up to 256 bytes, even addresses only.

The microcomputer accesses the variable SFR area in different way than that when it accesses fixed SFR area.

● Variable SFR area access

With SFR addressing, the range that can be set with operand is 00H to 0FFH. The variable SFR area range from 00H to 7FH. To first gain access to this area, initial value must be stored in the register SFRP.

The set value (00H-7FH) is represented by the value of register SFRP (upper 16 bits) and lower 7 bits specified in SFR addressing, total 24 bits which indicates the effective address with the 25B always 0 (see **Fig. 2-15**).

By changing the content of register SFRP, any location (memory is installed) in the address space can be allocated to the variable SFR area.

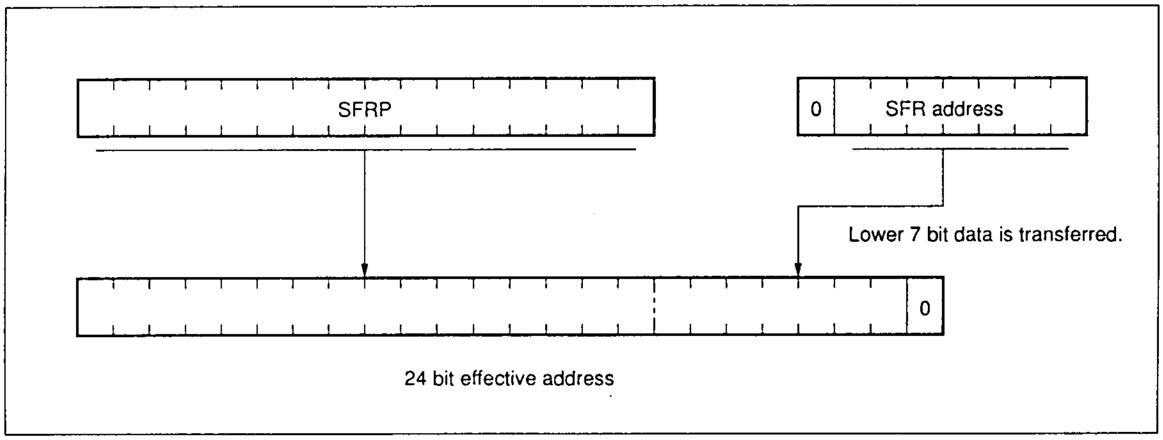


Fig. 2-15 24 bit address generated by the register SFRP

● Fixed SFR area access

When a value in the range between 80H and 0FFH is set in the SFR addressing, the register located at absolute address 00FF00H-00FFFHH is accessed. This location is called fixed SFR area.

Onto this space, the processor status word PSW, system configuration register SYS and the like, the registers necessary to control the CPU, are mapped. Control registers and I/Os associated with SM6010 functional controlling are also mapped onto this space. Because these control registers are located at the fixed SFR area, various functions can be controlled with effective accessing through the SFR addressing. During this procedure, the user program can be at any segment.

Even addresses in the fixed SFR area can be accessed when byte accessing is made. When the general purpose register R10 is accessed, only lower 8 bit data is accessed while the upper 8 bits are ignored.

Of the fixed SFR area, space 0E0H-0FFH (absolute address 00FFC0H-00FFFEH) is solely occupied by the CPU. The registers located in this space are common to models having SM6000CPU core (excluding reserved space).

Table 2-1 shows the registers common to SM6000CPU and **Table 2-2** shows the all registers in the fixed SFR area.

Table 2-1 CPU registers

SFR-ADDR	DA-ADDR	Word access	Byte access
FFH	00FFFEH	R15	R15L
FEH	00FFFCH	R14	R14L
FDH	00FFFAH	R13	R13L
FCH	00FFF8H	R12	R12L
FBH	00FFF6H	R11	R11L
FAH	00FFF4H	R10	R10L
F9H	00FFF2H	R9	R9L
F8H	00FFF0H	R8	R8L
F7H	00FFEEH	R7	R7L
F6H	00FFECH	R6	R6L
F5H	00FFEAH	R5	R5L
F4H	00FFE8H	R4	R4L
F3H	00FFE6H	R3	R3L
F2H	00FFE4H	R2	R2L
F1H	00FFE2H	R1	R1L
F0H	00FFE0H	R0	R0L
EFH	00FFDEH	PSW	PSW (L)
EEH	00FFDCH	SFRP	SFRP (L)
EDH	00FFDAH	SYS	SYS (L)
ECH	00FFD8H	Reserve	Reserve
EBH	00FFD6H	DP2	DP2
EAH	00FFD4H	DP1	DP1
E9H	00FFD2H	DP0	DP0
E8H	00FFD0H	SEG	SEG
E7H	00FFCEH	INTREQ1	INTREQ1 (L)
E6H	00FFCCH	INTREQ0	INTREQ0 (L)
E5H	00FFCAH	INTPRIO1	INTPRIO1 (L)
E4H	00FFC8H	INTPRIO0	INTPRIO0 (L)
E3H	00FFC6H	DTSSEL1	DTSSEL1 (L)
E2H	00FFC4H	DTSSEL0	DTSSEL0 (L)
E1H	00FFC2H	Reserve	Reserve
E0H	00FFC0H	Reserve	Reserve

Table 2-2 Registers in the space fixed for SFR (in the order of address) (1/2)

Address	SFR	Bit length*1	Symbol	Register	Page
00FFFEH	FFH	–	R15 (SP)	General purpose register R15 (stack pointer)	24
00FFFCH	FEH	–	R14 (FP)	General purpose register R14 (frame pointer)	24, 28
00FFFAH	FDH	–	R13	General purpose register R13	24
00FFF8H	FCH	–	R12	General purpose register R12	24
00FFF6H	FBH	–	R11	General purpose register R11	24
00FFF4H	FAH	–	R10	General purpose register R10	24
00FFF2H	F9H	–	R9	General purpose register R9	24
00FFF0H	F8H	–	R8	General purpose register R8	24
00FFEEH	F7H	–	R7	General purpose register R7	24
00FFECH	F6H	–	R6	General purpose register R6	24
00FFEAH	F5H	–	R5	General purpose register R5	24
00FFE8H	F4H	–	R4	General purpose register R4	24
00FFE6H	F3H	–	R3	General purpose register R3	24
00FFE4H	F2H	–	R2	General purpose register R2	24
00FFE2H	F1H	–	R1	General purpose register R1	24
00FFE0H	F0H	–	R0	General purpose register R0	24
00FFDEH	EFH	–	PSW	Processor status word	34, 30, 76
00FFDCH	EEH	–	SFRP	SFR (special function register) pointer	38, 30
00FFDAH	EDH	–	SYS	System configuration register	36, 30, 62, 107
00FFD6H	EBH	Byte	DP2	Data pointer 2	33
00FFD4H	EAH	Byte	DP1	Data pointer 1	32
00FFD2H	E9H	Byte	DP0	Data pointer 0	32
00FFD0H	E8H	Byte	SEG	Segment register	31
00FFCEH	E7H	–	INTREQ1	Interrupt request register 1	77
00FFCCH	E6H	–	INTREQ0	Interrupt request register 0	77
00FFCAH	E5H	–	INTPRIO1	Interrupt priority register 1	78
00FFC8H	E4H	–	INTPRIO0	Interrupt priority register 0	78
00FFC6H	E3H	–	DTSSEL1	DTS select register 1	79
00FFC4H	E2H	–	DTSSEL0	DTS select register 0	79

*1 "-" = word

*2 Cannot be accessed through the byte access instruction SFR addressing. Both upper and lower bytes will be accessed at the same time when SFR addressing with a word access instruction.

Table 2-2 also appears in Appendix B-1.

Address	SFR	Bit Length	Symbol	Register
00FFB8H	DCH	Byte	LCD_DMA	DMA Control
00FFB6H	DBH	Byte	LCD_MCLKW	MCLK Width
00FFB4H	DAH	Byte	LCD_CLKDIV	Clock Frequency Divider
00FFB2H	D9H	Byte	LCD_GRAY2	Gray Shade for 4-level
00FFB0H	D8H	Byte	LCD_GRAY1	Gray Shade for 4-level
00FFAEH	D7H	Byte	LCD_VDLT	Virtual Display Delta
00FFACH	D6H	Byte	LCD_SADRH	Start Address[23:16]
00FFAAH	D5H	Byte	LCD_SADRM	Start Address[15:8]
00FFA8H	D4H	Byte	LCD_SADRL	Start Address[7:0]
00FFA6H	D3H	Byte	LCD_DUTY	Duty Cycle
00FFA4H	D2H	Byte	LCD_CP1W	Line Pulse Width
00FFA2H	D1H	Byte	LCD_BC	Line Display Byte Count
00FFA0H	D0H	Byte	LCD_MODE	Mode
00FF8EH	C7H	Byte	SCI0C	Control
00FF8CH	C6H	Word	SCI0BR	Bit Rate
00FF8BH	C5H	Higher byte	SCI0S	Status
00FF8AH	C5H	Lower byte	SCI0RD	Receiver Data Register
00FF89H	C4H	Higher byte	SCI0M	Mode
00FF88H	C4H	Lower byte	SCI0TD	Transmitter Data Register
00FF82H	C1H	Byte	ADCC	Control
00FF80H	C0H	Byte	ADCD	Data
00FF6AH	B5H	Byte	PH_CKSEL	Peripherals' Clock Select
00FF68H	B4H	Byte	PWM_DC	Duty Cycle
00FF66H	B3H	Byte	PWM_TC	Terminal Count
00FF64H	B2H	Byte	PWM_CTL	Clock Divider
00FF62H	B1H	Byte	WDTC	Control
00FF58H	ACH	Byte	RTCTL	Control
00FF56H	ABH	Word	RTA	Alarm
00FF54H	AAH	Word	RTTD	Day
00FF52H	A9H	Word	RTHM	Hour-Minute
00FF50H	A8H	Byte	RTTS	Second
00FF44H	A2H	Byte	EXTDM	Extension Data Memory
00FF42H	A1H	Byte	SIRTM	500Khz timer for DASK

00FF40H	A0H	Byte	SIRCTL	Control
00FF1CH	8EH	Byte	P2	Data Register
00FF1AH	8DH	Byte	P4M1	Mode Register
00FF16H	8BH	Byte	P3D	Direction Register
00FF14H	8AH	Byte	P3	Data Register
00FF12H	89H	Byte	P4M0	Mode Register
00FF10H	88H	Byte	P4D	Direction register
00FF0EH	87H	Byte	P4	Data Register
00FF0AH	85H	Byte	P1M	Mode Register
00FF08H	84H	Byte	P0M	Mode Register
00FF06H	83H	Byte	P1D	Direction Register
00FF04H	82H	Byte	P1	Data Register
00FF02H	81H	Byte	P0D	Direction Register
00FF00H	80H	Byte	P0	Data Register

2.4 Data type

SM6000CPU supports the data type capable of data processing applications and other various data types suitable for controller, total 7 types shown below.

- Bit
- Unsigned byte
- Signed byte
- Unsigned word
- Signed word
- Unsigned long word
- Signed long word

These data types are described below.

(1) Bit data

Bit is a single bit operand representing only 0 or 1. The bit type has no sign. The bit type can specify only 1 bit in a word operand. Any bit in a byte or long word cannot be specified.

By specifying the bit operand during instruction execution, any bit in the general purpose register and the SFR can be set, cleared, operated, transferred or tested.

(2) Unsigned byte data

Unsigned byte has 8 bit value and can represent a value 0 to 255. The byte can be used with arithmetic operation, logical operation and data transfer instructions. Since the unsigned byte data is not restricted by a word boundary, it can be located anywhere in the address space. When accessing unsigned byte data in SFR addressing, the data must be located at even address.

(3) Signed byte data

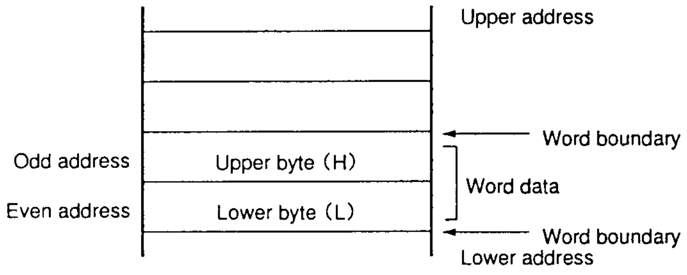
Signed byte has 8 bit value and can represent a value -128 to +127. The byte can be used with arithmetic operation, logical operation and data transfer instructions. Since the signed byte data is not restricted by a word boundary, it can be located anywhere in the address space. When accessing signed byte data in SFR addressing, the data must be located at even address.

(4) Unsigned word data

Unsigned word data has 16 bit value and can represent a value 0 to 65,535. Unsigned word data can be used with arithmetic operation, logical operation and data transfer instructions. Any single bit in the unsigned word data can be specified by bit access.

Unsigned word data must be placed between word boundaries. *

* ***The terminology "Word boundary" is used in relation to a word data to be stored in memory. The lower byte of some type of data must be stored into memory location having even address and the upper byte into the odd address location (the even address + 1). As a rule, all SM6010 word data should be stored within word boundaries. A long word data (32 bits) can be stored by repeating the procedure. Addressing an even address upon execution of an instruction automatically select word boundaries. (See figure below.)***



Word data located between word boundaries

(5) Signed word data

Signed word data has 16 bit value and can represent a value -32,768 to +32,767. Signed word data can be used with arithmetic operation, logical operation and data transfer instructions. Any single bit in the signed word data can be specified by bit access. Any signed word data must be located on a word boundary.

(6) Unsigned long word data

Unsigned long word data has 32 bit value and can represent a value 0 to +4,294,967,295. Unsigned long word data can be used only with MULU and DIVLU instructions. Unsigned long word data cannot be transferred by a single instruction. To store an unsigned long word to a general purpose register or to transfer a generated unsigned long word to memory, two word instructions are required.

(7) Signed long word data

Signed long word data has 32 bit value and can represent a value -2,147,483,648 to +2,147,483,647. Signed long word data can be used only with MULS and DIVLS instructions. Signed long word data cannot be transferred by a single instruction. To store a signed long word to a general purpose register or to transfer a generated signed long word to memory, two word instructions are required.

2.5 Instruction set and addressing mode

2.5.1 Instruction set

SM6010 has the following 7 instruction sets.

(1) Data transfer instruction

(2) Arithmetic operation instruction

(3) Logical operation instruction

(4) Rotate and shift instruction

(5) Bit manipulation instruction

(6) Branch instruction

(7) CPU control instruction

The writing format and fuction of these instructions are desribed below group by group.

(1) Data transfer instructions

MOV	DST , SRC	Move data
LDM	DST , SRC	Move multiple data
STM	DST , SRC	Move multiple data
CLR	DST	Clear
PUSH	SRC	Push data
POP	DST	Pop data
SWAP	DST	Swap byte
MOVSE	DST , SRC	Move data with sign extend
MOVZE	DST , SRC	Move data with zero extend

(2) Arithmetic operation instruction

ADD	DST , SRC	Add
ADC	DST , SRC	Add with carry
SUB	DST , SRC	Subtract
SBC	DST , SRC	Subtract with carry
CMP	DST . SRC	Compare
MULU	DST , SRC	Multiply unsigned

MULS	DST , SRC	Multiply signed
DIVU	DST , SRC	Divide unsigned
DIVS	DST , SRC	Divide signed
DIVLU	DST , SRC	Divide 32 bit unsigned
DIVLS	DST , SRC	Divide 32 bit signed
DADD	DST , SRC	Decimal add
DSUB	DST , SRC	Decimal subtract
NEG	DST	2's complement

(3) Logical operation instruction

AND	DST , SRC	Logical AND
OR	DST , SRC	Logical OR
XOR	DST , SRC	Logical exclusive OR
COM	DST	1's complement
BTST	DST , SRC	Bit Test

(4) Rotate and shift instruction

SLL	DST , SRC	Shift left logical
SRL	DST , SRC	Shift right logical
SRA	DST , SRC	Shift right arithmetic
ROR	DST , SRC	Rotate right
ROL	DST , SRC	Rotate left

(5) Bit manipulation instruction

BSET	DST	Bit set
BCLR	DST	Bit clear
BMOV	DST , SRC	Bit move
BAND	BF, SRC	Bit logical AND
BOR	BF , SRC	Bit logical OR
BXOR	BF, SRC	Bit logical exclusive OR

(6) Branch instruction

BR	cc , RA	Relative jump
JMP	cc , DST	Jump
SJMP	DST	Segment jump
CALR	DST	Relative call
CALL	cc , DST	Call
SCALL	cc , DST	Segment call
RET	cc	Return
IRET	cc	Interrupt return
SRET	cc	Segment return
BBC	SRC , RA	Branch on bit clear
BBS	SRC , RA	Branch on bit set
BBCS	SRC , RA	Branch on bit clear with set
BBSC	SRC , RA	Branch on bit set with clear
DBNZ	R , DST	Loop control

(7) CPU control instruction

TRAP		Software interrupt
TRAPV		Overflow software interrupt
LINK	SRC	Generate stack frame
UNLINK		Unlink stack frame
STOP		Stop CPU
HALT		Halt CPU
EI		Enable interrupt (Bit I ← 1)
DI		Disable interrupt (Bit I ← 0)
RST		Reset software
NOP		No operation

2.5.2 Addressing mode

SM6000CPU supports the following 10 addressing modes. These modes are sometimes described in the user's manual in brackes [ ]. Available addressing modes differ from instruction to instruction.

Table 2-3 Addressing modes

Addressing mode		Symbol
Register direct	: R addressing	Rn
Register pair direct	: RR addressing	RRn
Register indirect	: (R) addressing	(Rn)
Register indirect with displacement	: disp (R) addressing	disp (Rn)
Post increment register indirect	: (R) + addressing	(Rn) +
Pre decrement register indirect	: - (R) addressing	- (Rn)
Absolute address	: DA addressing	DA
Immediate	: IM addressing	IM
Program counter relative	: RA addressing	RA
Special function register	: SFR addressing	SFR

Rn : R0-R15
RRn : RR0-RR14

(1) Register direct Rn [R addressing]

The 16 general purpose registers (R0-R15) denoted in the register field of an instruction code are operands. Even if an operand is 8 bit data instruction, R addressing can be used: write R0-R15, and only the lower 8 bit data of 16 bit general purpose register functions as an operand. To avoid confusing on data type, when using the byte instruction in R addressing, write R0L-R15L; and do not use R0L-R15L in word instructions.

(2) Register pair direct RRn [RR addressing]

The 8 general purpose register pairs (RR0-RR14) denoted in the register field of an instruction code are operands. These register pairs can be used only in multiplication and division instructions (MULU, DIVLU, MULS, DIVLS, DIVS, DIVU). When these instructions are executed, operation result (long word) is stored into a general register pair.

(3) Register indirect (Rn) [(R) addressing]

The content of general purpose registers (R0-R15) denoted by the register field of an instruction code is used as the address to specify an operand in memory.

The referenced segments depend on the specified register. When indirect addressing through general purpose registers R0-R7, the data pointer DP1 is referenced; and when R8-R15, data pointer DP2. When accessing word data indirectly through registers, the registers must specify even address.

Table 2-4 Data pointer and referenced segment

Register	Referenced segment
General purpose register R0-R7	Segment denoted by DP1
General purpose register R8-R15	Segment denoted by DP2

(4) Register indirect with displacement disp (Rn) [disp (R) addressing]

The 16 bit displacement of the second word of an instruction code is added to the content of the register denoted in the register field of the instruction code to express the address of an operand on memory.

The referenced data pointer depend on the specified register. When indirect addressing through general purpose registers R0-R7, the data pointer DP1 is referenced; and when R8-R15, data pointer DP2. The disp value is signed 15 bit data added to the register. Data can be specified to the disp in a range of -32,768 and +32,767.

When accessing word data indirectly through registers with displacement, the address must be even.

(5) Post increment register indirect (Rn)+ [(R) addressing]

The content specified in the register field of an instruction code is used as the address of an operand on memory. 1 (byte instruction) or 2 (word instruction) is added to the content of the register and the result stored in the register. The referenced segments depend on the specified register. When indirect addressing through general purpose registers R0-R7, the data pointer DP1 is referenced; and when R8-R15, data pointer DP2.

When accessing word data indirectly through post increment registers, the address must be even.

(6) Predecrement register indirect -(Rn) [-(R) addressing]

The content of the register specified by the register field of an instruction code subtracted by 1 (byte instruction) or 2 (word instruction) is the address of an operand in memory. The result of subtraction is stored in the register.

The referenced segments depend on the specified register. When indirect addressing through general purpose registers R0-R7, the data pointer DP1 is referenced; and when R8-R15, data pointer DP2. To access word data indirectly through predecrement registers, the address must be even.

(7) Absolute address DA [DA addressing]

The value specified by the second word of an instruction code is used as the address of an operand in memory. The data pointer DP0 is referenced as the segment data.

(8) Immediate IM [IM addressing]

The value specified by the second word of an instruction code is used as an immediate operand.

The 8 bit value or 16 bit value is used depending on the size specified by an instruction: when a byte instruction specifies word data, lower byte of the second word is used and upper byte is ignored; if word instruction, the second word is used as 16 bit word data.

(9) Program counter relative RA [RA addressing]

In this addressing, 8 bit displacement (BR, BBC, BBS, BBCS and BBS instructions) or 12 bit displacement (CALR, BBC, BBS, BBCS and BBSC instructions) is used. A signed data specified by the disp field of an instruction code is added to the current program counter PC and the result is used as an operand. RA should be in the range from -256 to +254 (8 bit) or -4,096 to +4,094 (12 bit).

(10) Special function register SFR [SFR addressing]

Data in the address range FF00H-FFFFH (fixed SFR area) and data in the 128 word space (variable SFR area) starting at the address specified by the register SFRP can be specified with 8 bit address: 00H-07F is variable SFR area and 0F0H-0FFH fixed SFR area. Usually word accessing is used. Using SFR addressing with a byte instruction accesses lower 8 bits (even address). The upper 8 bits are ignored.

2.5.3 Instruction format

SM6000CPU has simplified instruction set for use with high speed CPU. These instructions are shown in **Table 2-5** and **Appendix B-3.**

Table 2-5 Instruction format

OP	OPR	BYTE	FORMAT				
			15 12	11 8	7 4	3	0
SLL, SRL, SRA, ROR	Rd, Rs	2	Rd	Rs	OP		
SLL, SRL, SRA, ROR	Rd, IM4	2	Rd	IM4	OP		
ADD, ADC, SUB, SBC, AND, OR, XOR, MOV, CMP	Rd, Rs	2	Rd	Rs	OP	b/w	OP
ADD, ADC, SUB, SBC, AND, OR, XOR, MOV, CMP	Rd, IM4	2	Rd	IM4	OP	b/w	OP
ADD, ADC, SUB, SBC, AND, OR, XOR MOV, BTST, CMP	SFR, IM	4	SFR		OP	b/w	OP
			IMH		IML		
ADD, ADC, SUB, SBC, AND, OR, XOR MOV, CMP	SFR, DA	4	SFR		OP	b/w	OP
			DAH		DAL		
ADD, ADC, SUB, SBC, AND, OR, XOR MOV, CMP	DA, SFR	4	SFR		OP	b/w	OP
			DAH		DAL		
ADD, ADC, SUB, SBC, AND, OR, XOR MOV, CMP	SFR, SFR	4	11111111		OP	b/w	OP
			SFR		SFR		
DADD, DSUB	Rbd, Rbs	2	Rbd	Rbs	OP		
MULS, MULU, DIVS	RRd, Rs	2	RRd	Rs	OP	b/w	OP
DIVUS, DIVLU	RRd, RRs	2	RRd	RRs	OP	b/w	OP
BCLR, BSET	SFR.b	2	SFR		OP	bit	
BCLR, BSET	IM(R).b	4	bit	R	OP		
			IMH		IML		
DBNZ	R, DA	4	R	1111	OP		
			DAH		DAL		
BR	cc, RA	2	RA		OP	cc	
CALR	RA	2	RAL		OP	RAH	
JMP, CALL	cc, (R)	2	cc	R	OP		
JMP, CALL	cc, DA	4	cc	1111	OP		
			IMH		IML		
BBC, BBS, BBCS, BBSC	SFR.b, RA	4	bit	RAH	OP		
			RAL		SFR		
BBC, BBS, BBCS, BBSC	IM(R).b, RA	4	bit	R	OP		
			RAL		SFR		
LDM, STM	(R), IM	4	R	1111	OP		
			IMH		IML		
BMOV	SFR.b, SFR.b	4	bit.d	bit.s	OP		
			SFR		SFR		
BAND, BOR, BXOR	BF, SFR.b	4	bit.d	bit.s	OP		
			PSW		SFR		
MOV	R, (R)	2	R	(R)	OP		
MOV	R, (R)+	2	R	(R)+	OP		
MOV	(R), R	2	(R)	R	OP		
MOV	−(R), R	2	(R)	R	OP		
MOV	Rd, IM(R)	4	Rd	R	OP		
			IMH		IML		
MOV	IM(R), Rs	4	R	Rs	OP		
			IMH		IML		
SJMP, SCALL	seg: DA	4	seg		OP		
			DAH		DAL		
NEG, COM, SWAP, PUSH, POP, CLR	SFR	2	SFR		OP		
LINK	IM8	2	IM8		OP		
RET, RETI, SRET	cc	2	cc	1111	OP		
UNLINK, STOP, HALT, EI, DI NOP, TRAP, TRAPV	-	2	11111111		OP		

2.6 Precautions

● Registers need the initial value settings

The initial value of the following registers are unknown.

- Register SFRP
- Data pointers DP0-DP2

The user program must first set the initial value of the SFRP and DP0-DP2. The content of the registers SFRP and the location of variable SFR area are unknown at the end of the hardware reset sequence. If a subroutine call is made while the content of data pointers DP0-DP2 are unknown, the user program cannot run as designed. For the registers that must be set at the end of the hardware reset sequence, refer to **3.4 Hardware reset**.

● Store only even data into SP and FP

Store only even data into the stack pointer SP to allow it represent an even address. If contains odd data, the operation of the microcomputer becomes unstable during stacking operation. Also store only even data into the frame pointer FP. Otherwise, the SP contains odd data when stack frame is set, causing unstable microcomputer operation.

● No subclock on SM6010

Even though the SM6000CPU can use a subclock to stop generation of the main clock or as the system clock, it cannot use a subclock since it is not provided on the SM6010 (SM6003/6004/6005/6006).

3

Chapter 3 System control

3.1 Oscillator circuit

3.1.1 Connection of crystal

The SM6010 has a main clock oscillator to which a crystal of up to 30 MHz can be connected. **Fig. 3-1** shows a typical main clock oscillator configuration.

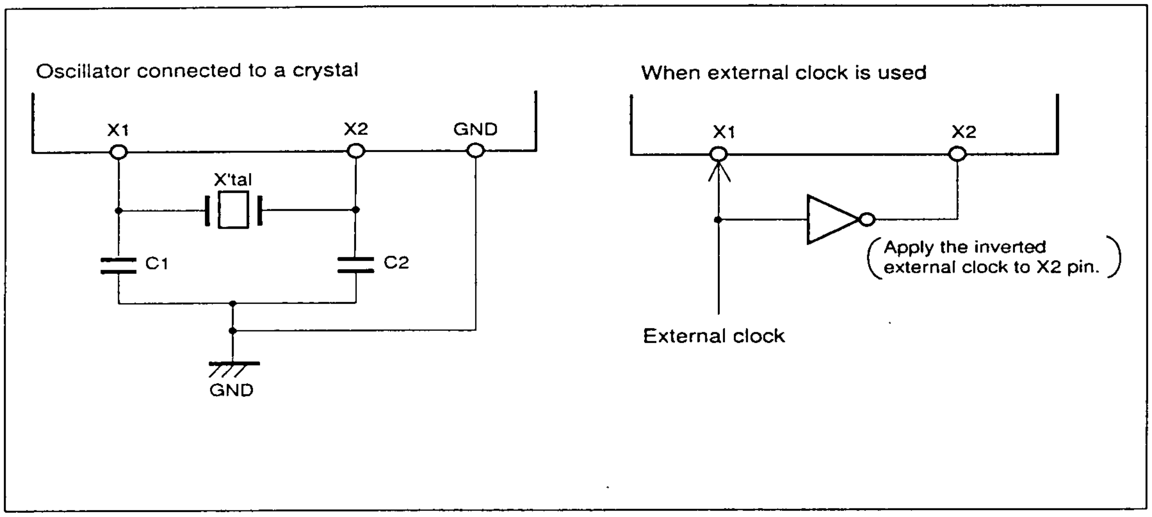
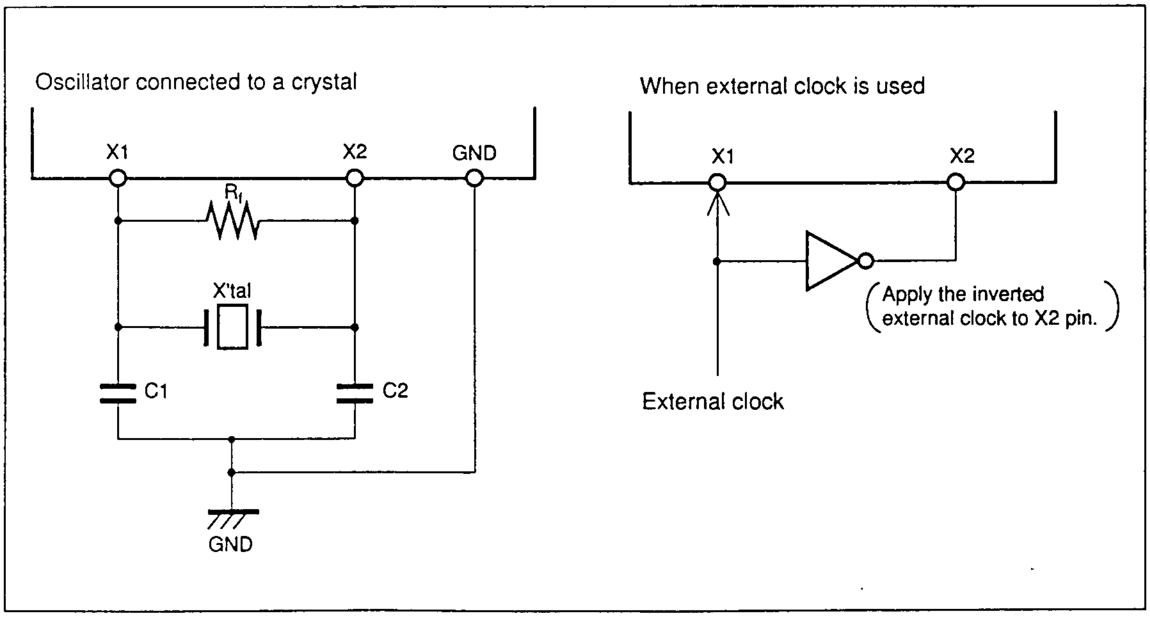

Oscillator connected to a crystal
X1
X2
GND
X'tal
C1
C2
GND
When external clock is used
X1
X2
Apply the inverted external clock to X2 pin.
External clock

Fig. 3-1 Main clock oscillator (reference only)

The oscillator circuit shown in **Fig. 3-1** is an example and may not be applied to the product because the circuit may differ from those in **Fig. 3-1** depending on the type and make of crystal.

The configuration, constant and components (e.g. capacitor) of the circuit and components are determined according to the PCB pattern layout.


Oscillator connected to a crystal
X1
X2
GND
Rf
X'tal
C1
C2
GND
When external clock is used
X1
X2
Apply the inverted external clock to X2 pin.
External clock

3.1.2 Mounting considerations

When installing LSI and oscillator circuits (main clock and subclock) on a PCB , observe the following precautions to minimize effects of stray capacitance of wiring and noises.

(1) Wirings connected to the oscillator circuit must be as short as possible. The capacitors and crystal must be located closely to the X pins to minimize stray capacitance (see **Fig. 3-2** (1)).

(2) Do not connect the return path of the oscillator to a GND of high current (see **Fig. 3-2** (2)).

(3) Do not derive the clock signal directly from the oscillator circuit wiring (see **Fig. 3-2** (3)).

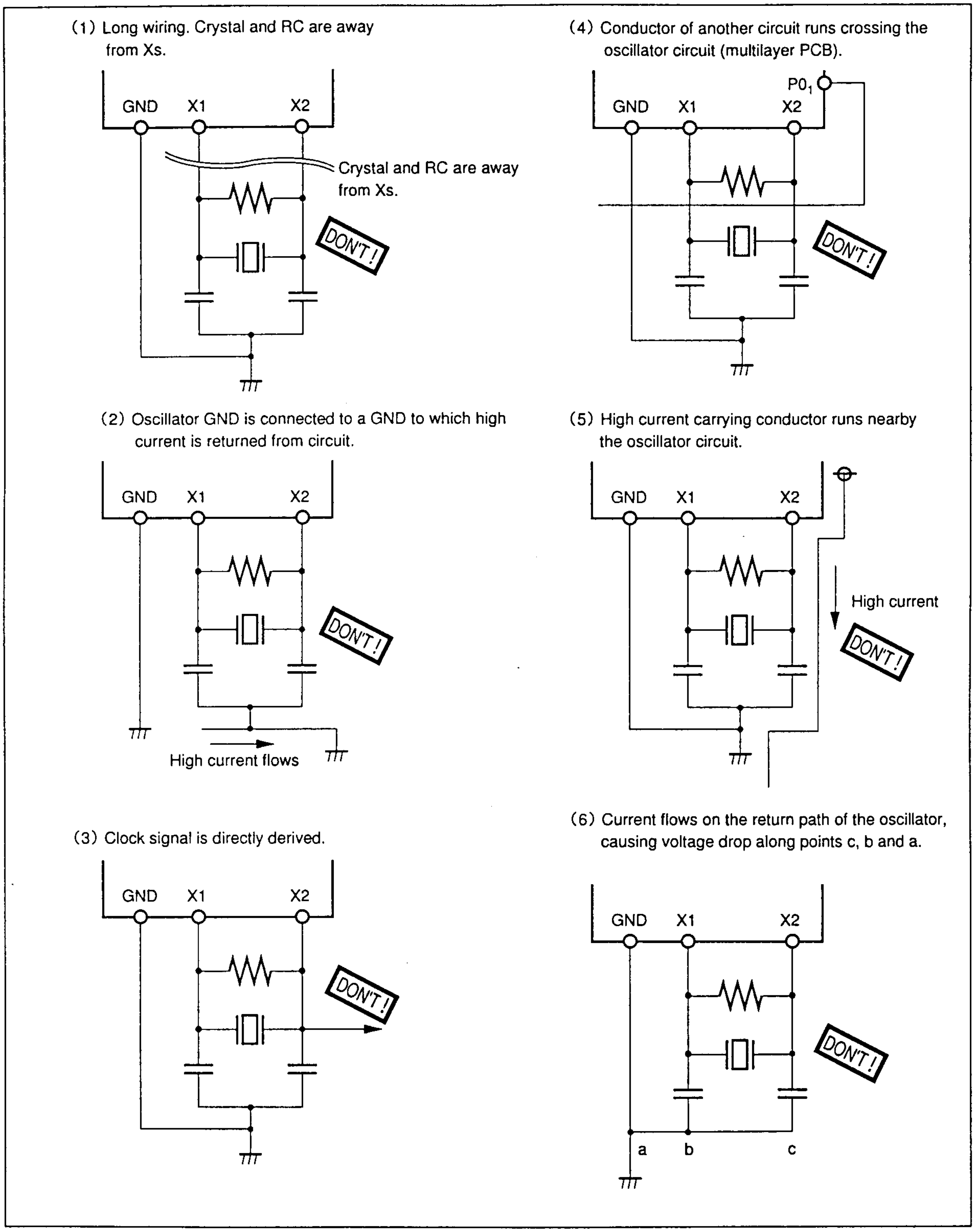
(4) When the oscillator circuit is mounted on a multilayer PCB, do not cross the wiring plane or wirings of the oscillator circuit and other circuit wiring plane or wirings (see **Fig. 3-2** (4)).

(5) Do not run high current conductor close to the oscillator circuit (see **Fig. 3-2** (5)).

(6) Directly connect the negative side of the capacitors to the GND of LSI for no potential difference (see **Fig. 3-2** (6)). This also minimize stray capacitance.

(7) Run GND of LSI and V_{DD} lines so that they offer minimum stray capacitance.

Fig. 3-2 shows examples of poor oscillator circuit connections.



(1) Long wiring. Crystal and RC are away from Xs.

(2) Oscillator GND is connected to a GND to which high current is returned from circuit.

(3) Clock signal is directly derived.

(4) Conductor of another circuit runs crossing the oscillator circuit (multilayer PCB).

(5) High current carrying conductor runs nearby the oscillator circuit.

(6) Current flows on the return path of the oscillator, causing voltage drop along points c, b and a.

Fig. 3-2 Examples of wrong oscillator configuration

3.2 Clock system

The SM6010 operates on the main clock whose frequency is determined by the inputs to the X1 and X2 pins. See **Fig. 3-3.**

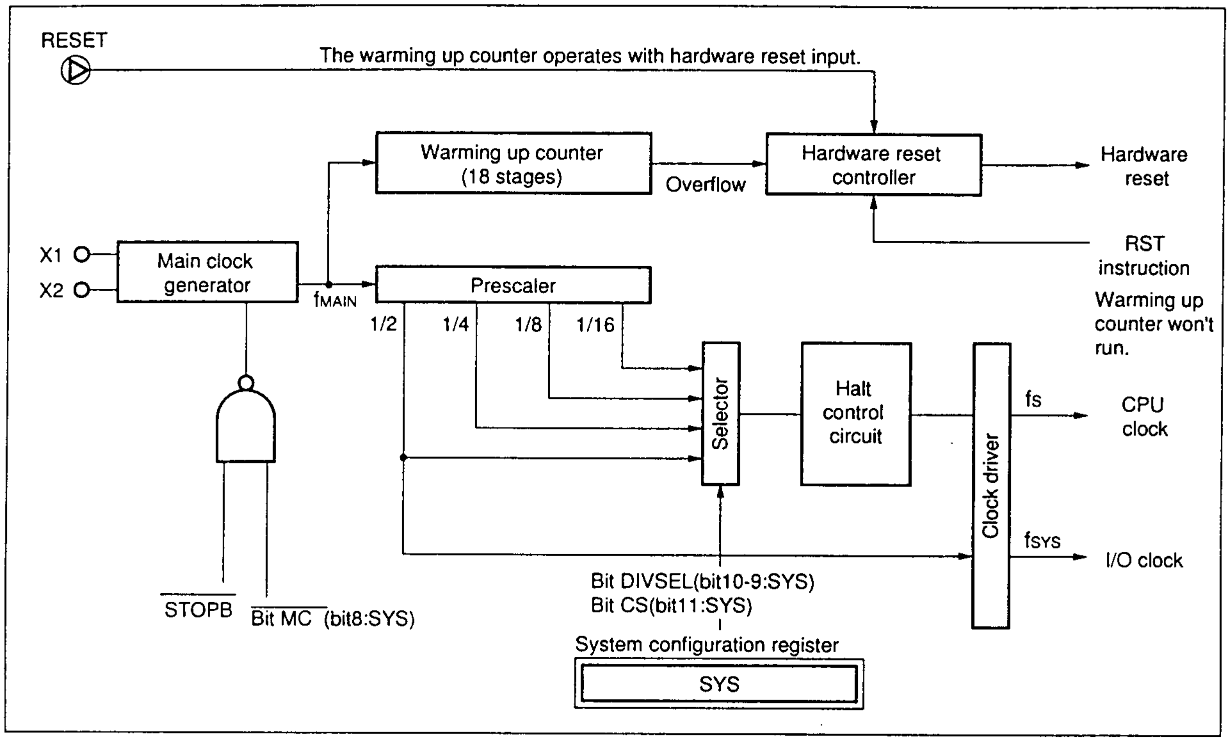


Fig. 3-3 Clock system

The CPU operates on the system clock fs which is a divided-by-n main clock (n = 2, 4, 8, 16). Selection can by made through the user program. The hardware reset sets the system clock to the default divided-by-2 main clock.

Peripheral blocks operate on the divided-by-2 main clock (fSYS). Changing the system clock fs does not change the frequency of fSYS delivered to these peripheral blocks. Also, the halt mode does not stop the peripheral blocks.

3.2.1 Registers in the clock system

(1) SYS: system configuration register

The SYS sets the operation mode of the CPU. This section describes the functions related to the clock control (bit 8-11). For full description of all bits, refer to **2.3.4 SYS: system configuration register.**

Bit description is as follows.

■ Bit 11: operation clock select (CS) .

Bit 11 selects either the system clock or subclock.

Bit CS	Operation
0	Select main clock
1	Select subclock

■ Bits 10-9: divided system clock select (DIVSEL)

Bit DIVSEL		System clock
0	0	Divided-by-2 main clock
0	1	Divided-by-4 main clock
1	0	Divided-by-8 main clock
1	1	Divided-by-16 main clock

DIVSEL does not affect the frequency of clocks supplied to peripheral blocks.

■ Bit 8: main clock stop (MC) .

Bit MC	Operation
0	Enables main clock oscillator
1	Stops main clock (active only when CS is set at 1)

If the subclock is available and subclock is selected (bits SYS.CS is set at 1), setting bit MC at 1 stops the main clock.

(2) Warming up counter

This 18-bit counter counts the time required for the main clock oscillator to stabilize. This counter starts counting at the end of hardware reset sequence or upon returning from the stop mode. When its count reaches 2^{18} and causes oveflow, the microcomputer returns to the normal mode.

Since the warming up counter is not mapped on the address space, it cannot be read/written.

(3) Subclock Consideration

DIVSEL does not work while CPU is running with subclcok. Do not change CS and DIVSEL at the same time. During running with main clock, MC bit doesn't affect main clock operation. Do not execute STOP instruction while CPU is running with subclock. You need to set MC bit to stop main clock.

Watchdog Timer does mot work with subclock operation.

3.2.2 System clock selection and operation

The system clock can be selected by following procedure described below:

(1) Disable all the interrupts. Use DI instruction. Do not attempt to set the interrupt enable bits to 0 (of the functional block control registers) for the purpose of disabling interrupts.

(2) Select the desired frequency divider by setting bit DIVSEL (bits 10-9: SYS). While the instruction to write the set value into the bit DIVSEL is executed, the system clock change process takes place, the time required for this process depends on the frequency of the sytems clock before and after the selection. When the DVSEL is written, the system clock is turned to high level at the rising edge of the next system clock and kept high until the internal divider counts 0FH and then 0H (up to fMAIN/ 2 x 15 clocks). Upon reaching 0H, the selected system clock begins to tick the microcomputer. The length of the high duration depends on the timing of the write operation into the bit DIVSEL.

- The high duration of the system clock during clock changing period = [(fMAIN/2) x 1]-[(fMAIN/2) x 15]

Fig. 3-4 shows a timing when the system clock is changed from the divided-by-2 main clock to divided-by-4 main clock.

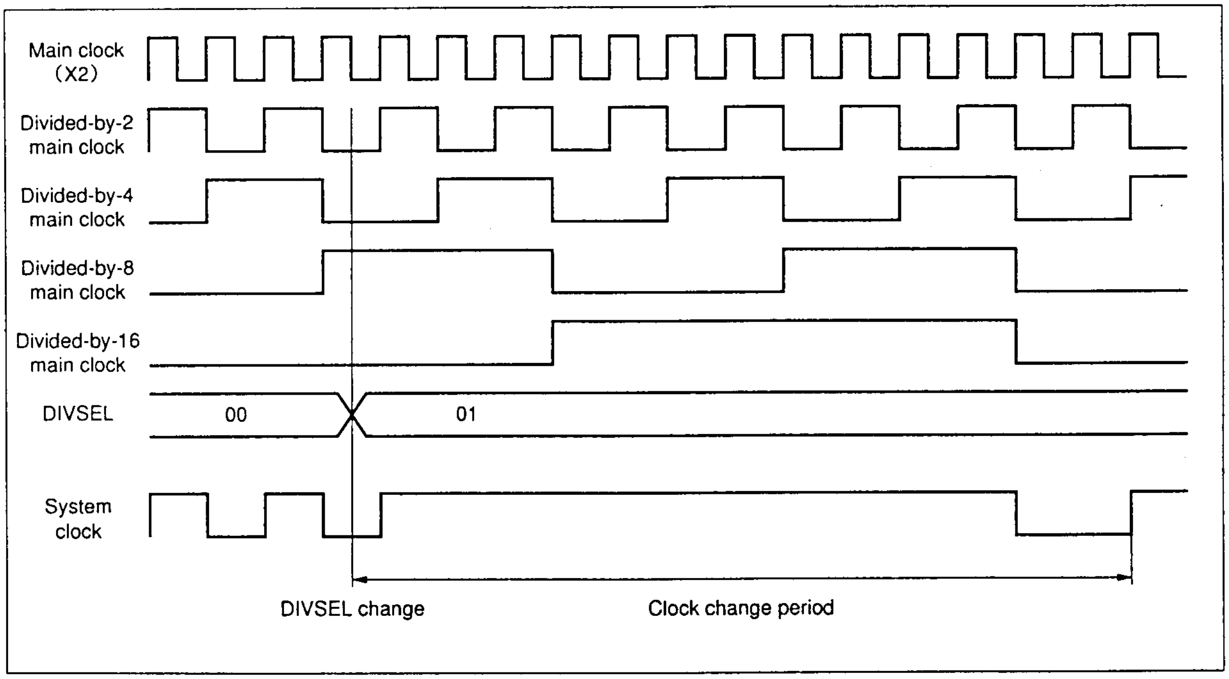


Fig. 3-4 System clock changing timing chart (example)

3.3 Operations

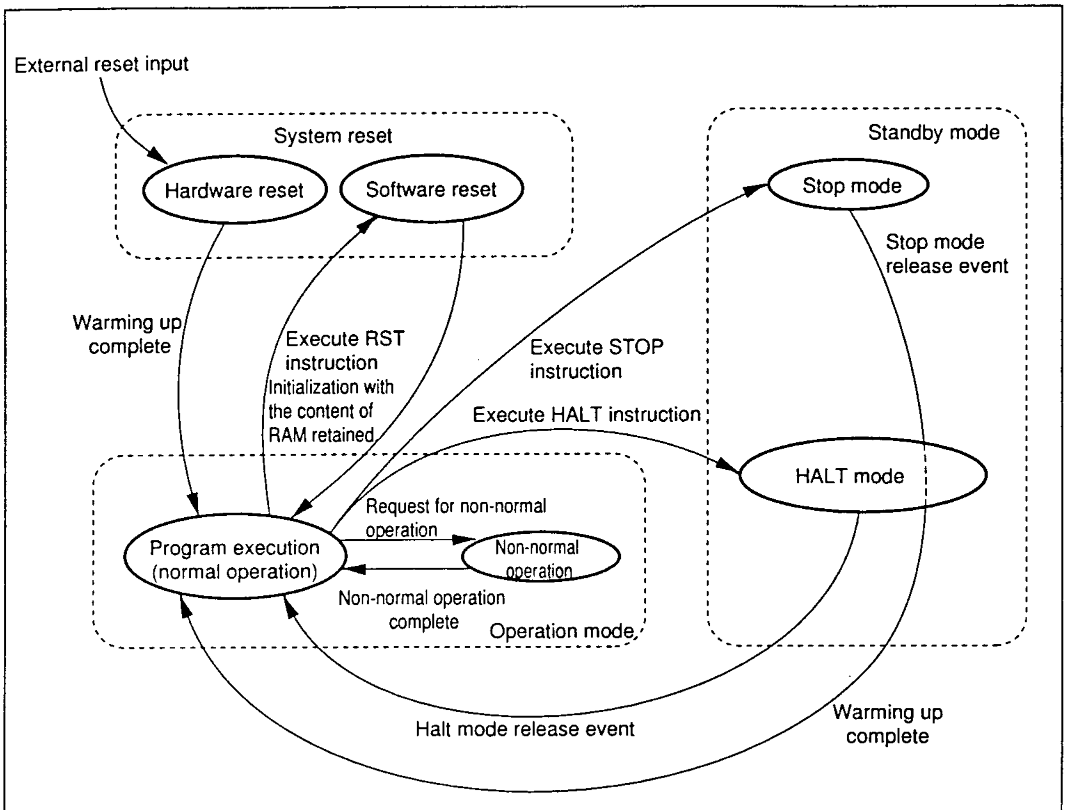
The SM6000 CPU falls into one of the following 4 operation states, as shown in **Table 3-2.**

Table 3-2 Operating status of SM6000 CPU

Status	Description	Section
Program execution (normal operation)	The CPU is executing the user program.	–
Operations other than program execution	The CPU is executing transient process such as the acknowledgement cycle caused by an interrupt request and DTS. When the CPU enters interrupt handling routine after the acknowledgement cycle, the CPU is said executing the program. While the CPU is not handling the program but operating for other purposes, peripheral blocks such as timer continue operation.	3.5 Interrupt 3.6 DTS
System reset	The CPU is initialized by the hardware or software reset sequence. This includes the status where the RESET pin is at low level.	3.4 System reset
Standby state (standby mode)	The CPU is in stop. Standby mode is further classified into two modes: Halt mode in which CPU clock is stopped, and Stop mode in which the main clock is stopped.	3.7 Standby function

For each CPU status, refer to the section listed in the Section column.

Fig. 3-5 below shows the operation shift of the CPU.



Note 1: When Stop mode release event occurs, the CPU returns to the normal operation mode after the warming up period.
Note 2: External reset input is accepted regardless of the state of the microcomputer.

Fig. 3-5 Shift among operations

3.4 System reset

System reset is a function to initialize the system of the microcomputer. The system reset is classified into two, hardware and software resets. The trigger sources of the system reset are as shown below.

(1) External reset (hardware reset)

The hardware reset sequence starts when the low level is placed on RESET pin of the SM6010.

(2) RST instruction is executed (software reset)

RST instruction starts the software reset sequence.

Table 3-3 shows the differences between hardware and software reset.

Table 3-3 Differences between hardware and software reset

Item	Hardware reset	Software reset
Start	Low level placed on RESETB pin	RST instruction executed
Warming up counter	Starts when RESETB pin is pulled	No operation. No warming up.
Initialization	high. Both resets have the same functions	
Start address	000100H	

3.4.1 Description of operation

(1) Hardware reset

When RESETB pin is pulled low, e.g. through an external reset IC, hardware reset sequence starts at the next 2 instruction cycles. To assure the hardware reset, RESETB pin should be kept low for more than 2 system clock period.

When RESETB returns from low to high level, microcomputer starts the internal warming up counter. The hardware reset sequence completes after the warming up period of approx. 2^{18} main clocks and the program starts at address 000100H (see **Fig. 3-6**). During the warming up period, microcomputer is in the same state as in the halt mode.

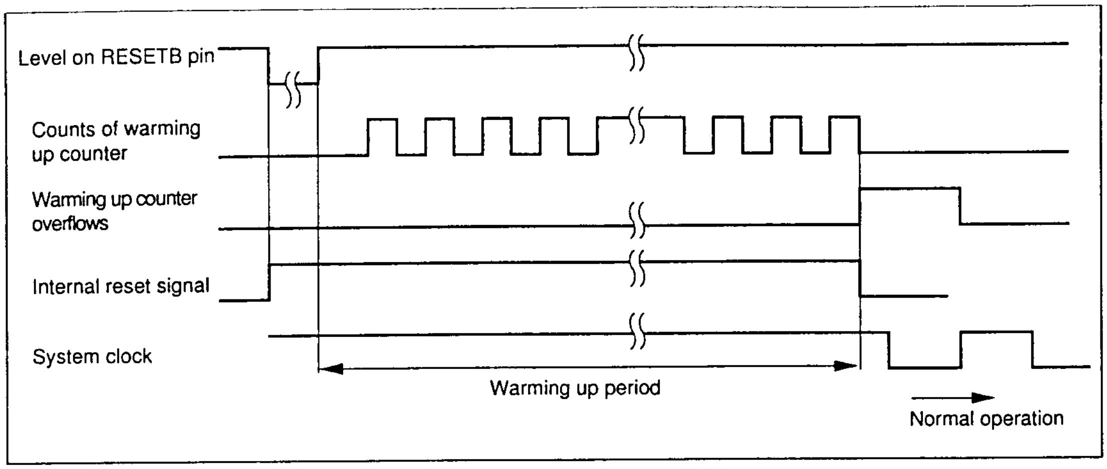


Fig. 3-6 Hardware reset timing

(2) Software reset

The microcomputer retains the previous content of RAM if the software reset causes initialization. The warming up counter will not operate and return to the normal operation. Other initializations are same as in the case of hardware reset sequence.

Execute RST instruction (in nonmaskable interrupt vector upon overflowing of the watch dog timer) from the user program to force the the microcomputer to start the software reset. The software reset will be used as initializing process when the microconmputer overruns.

Initialization by software reset includes initialization of the I/Os and registers in the same way as in the case of hardware reset. The initial values must be set by the user program.

(3) System status at the end of system reset

When the system reset sequence completes, the system is initialized as follows:

● Operation mode

- Single chip mode
- System clock speed is 1/2 the main clock.

● CPU control related registers and memories whose contents are unkown

- General purpose registers R0-R15
- Stack pointer SP
- Data pointers DP0-DP2
- Some bits of processor status word PSW
- The values read through input pins (value of input data registers)

For the initial value of the registers on the fixed SFR area, refer to description of these registers.

- State of input and output pins
  - Set as normal I/O port and not a functional pin.
  - I/O ports are set to input.

- Interrupt functions
  - Bit I (bit 9: PSW) of the processor status PSW is 0, inhibiting all maskable interrupts.
  - Interrupt enable bit in the functional block control registers are set at 0, assuring disabling of individual maskable interrupts.

- Internal peripheral functions

The timer 0 runs as a free-run counter. fsys/2 is selected as the count clock. All other internal peripheral functions are in stop.

(4) Register setting procedure after system reset sequence

At the end of the system reset sequence, the following registers should be set to the initial value, as necessary, by using the user program.

- Setting stack pointer SP and data pointers DP0-DP2

Store the start address of the stack into the stack pointer SP. Set the segment of the stack by storing the desired value into the data pointer DP2. Do not use the stack (interrupt, call instruction) before conducting this setting. Since the nonmaskable interrupts cannot be disabled even after the hardware/software reset sequence, the stack must be set first.

- Setting system configuration register SYS

Set the operation mode of the CPU and system clock speed. If external expansion mode is used, set the bus size and bus cycle necessary for external accessing.

- Setting processor status word PSW (register PSW)

Set the necessary interrupt related settings: enable/disable maskable interrupts, enable/disable DTS operation, and interrupt priority.

All maskable interrupts are inhibited after hardware reset since bit I (bit 9: PSW) is set at 0.

- Setting I/O ports

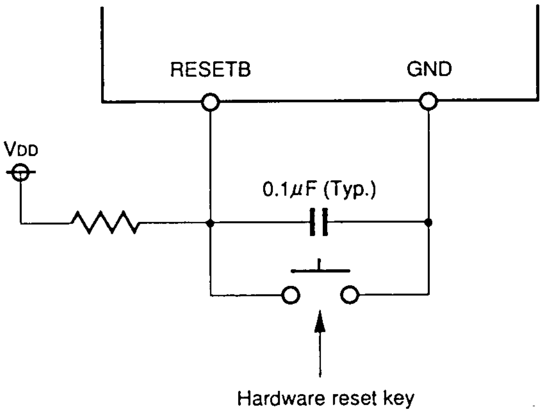
When necessary, set the direction of I/O ports, set pins to peripheral function pins, select the edge of the external interrupt signal necessary to detect the interrupt. For further information, refer to **Chapter 4 I/O port.**

- Setting functional blocks

When necessary, set the internal peripheral functions.

3.4.2 Connections

Fig. 3-7 shows typical RESETB pin connection. Connect a bypass capacitor across RESETB and GND as shown in **Fig. 3-7** to suppress noises. Install an external reset input key and connect a pull-up resister across RESETB pin and VDD.



Note: Standard value of the capacitor across RESETB and GND pins is 0.1μF and must be tailored to the actual circuit condition.

Fig. 3-7 RESETB pin connection (example)

3.4.3 System reset considerations

- Precautions on system configuration

  - Instantaneous, excessive supply voltage variation may start hardware reset sequence automatically.
  - Status of all ports are unknown during power-up and the start of the hardware reset sequence.

The above conditions depend on user system design and should be checked during installation.

- Precautions on power-up sequence

Connect the noise capacitor to RESETB pin (refer to **Fig. 3-7 RESETB pin connection**). On power-up, fully charged capacitor will simulate the external reset signal. To avoid this, place a low level on RESETB pin through a reset IC. The values of the capacitor showing in **Fig. 3-7** are standard ones.

The actual value should be selected by the user for stable system operation.

3.5 Interrupt

The SM6010 has 24 maskable interrupts and 5 nonmaskable interrupts. Maskable interrupt can be used for automatical data transfer through DTS (Data Transfer Server). Nonmaskable interrupt events include watch dog timer interrupt and software trap instruction.

This section describes DTS from the time an interrupt request is issued to the acceptance of DTS for processing. For further information on DTS processing and control procedure, refer to **3.6 DTS**.

This section describes the registers related to interrupt, types of interrupt and interrupt operation. For description of maskable interrupt events, refer to the description on that particular interrupt.

Fig. 3-8 shows a block diagram of interrupt controller, **Table 3-4** lists the interrupts and **Table 3-5** shows maskable interrupt events and interrupt enable bits.

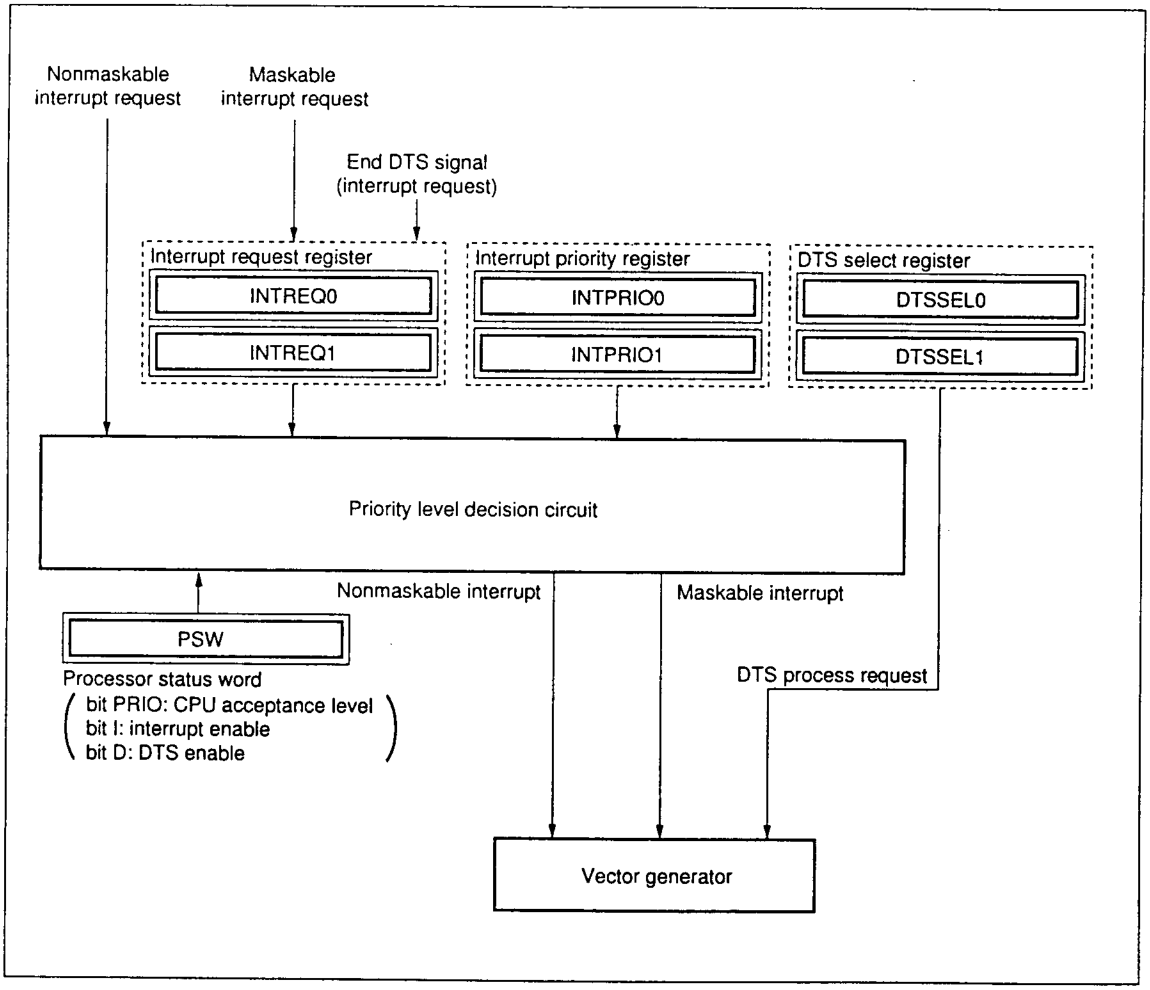

Nonmaskable interrupt request
Maskable interrupt request
End DTS signal (interrupt request)
Interrupt request register
INTREQ0
INTREQ1
Interrupt priority register
INTPRIO0
INTPRIO1
DTS select register
DTSSEL0
DTSSEL1
Priority level decision circuit
Nonmaskable interrupt
Maskable interrupt
PSW
Processor status word
bit PRIO: CPU acceptance level
bit I: interrupt enable
bit D: DTS enable
DTS process request
Vector generator

Fig. 3-8 Interrupt controller

Table 3-4 List of Interrupts

Interrupt	Interrupt events	Interrupt from SM6010			Interrupt	DTS	Prioirty
	from CPU	Group	Block	Event	vector	vector	level
Nonmaskable (software trap)	Detect undefined code	-	CPU	Illegal instruction	0CH	-	-
	TRAP instruction	-	CPU	Execute TRAP instruction	10H	-	-
	TRAPV instruction	-	CPU	Execute TRAPV instruction (bit V=1)	14H	-	-
Nonmaskable (hardware trap)	NMI	-	-	Non Maskable Interrupt	00H	-	-
	WATCHDOG	-	Watch dog	Overflow	04H	-	25
	ADERR	-	Address error	Odd address access	08H	-	24
Maskable	INT23	5	RTC	Alarm	20H	A0H	23
	INT22		RTC	Second	24H	A4H	22
	INT21		RTC	Minute	28H	A8H	21
	INT20		RTC	Hour	2CH	ACH	20
	INT19	4	EXINT0	External interrupt input ($P4_0$ pin)	30H	B0H	19
	INT18		EXINT1	External interrupt input ($P4_1$ pin)	34H	B4H	18
	INT17		EXINT2	External interrupt input ($P4_2$ pin)	38H	B8H	17
	INT16		EXINT3	External interrupt input ($P4_3$ pin)	3CH	BCH	16
	INT15	3	EXINT4	External interrupt input ($P4_4$ pin)	40H	C0H	15
	INT14		EXINT5	External interrupt input ($P4_5$ pin)	44H	C4H	14
	INT13		EXINT6	External interrupt input ($P4_6$ pin)	48H	C8H	13
	INT12		EXINT7	External interrupt input ($P4_7$ pin)	4CH	CCH	12
	INT11	2	RTC	Day	50H	D0H	11
	INT6	1	SCI0	Receive	64H	E4H	6
	INT5		SCI0	Transmit	68H	E8H	5
	INT0	0	A/D converter	End of conversion	7CH	FCH	0

3.5.1 Interrupt register

(1) Bits I, D and PRIO

Bits I and D are located on the processor status word (register PSW). Bit I (bit 9: PSW) is the interrupt master enable bit which enables or disables all the maskable interrupts at the same time. Bit D (bit 8: PSW) is the master enable bit of the DTS operation which enables or disables all DTS operations.

Bit PRIO (bits 6-4: PSW) sets the priority order of maskable interrupts.

When an interrupt request is issued, the microcomputer compares PRIO bits with corresponding 3 bits of priority registers INTPRIO0 and INTPRIO1 which contain 3 bits x 5 groups of priority bits. (See **Fig. 3-9 Interrupt process flow** on page 85.)If the value of 3 bits of the registers is equal to or larger than bits PRIO, the interrupt is accepted. For details of INTPRIO0 and INTPRIO1, refer to **3.5.1 (3) INTPRIO 0,1: interrupt priority register 0 and 1**. For details of priorities, refer to **3.5.4 Interrupt priority.**

- Bit I = 0: disable all maskable interrupts at the same time

  = 1: enable all maskable interrupts at the same time (individual setting is also required)
- Bit D = 0: disable all DTS operations at the same time

  = 1: enable all DTS operations at the same time (individual setting is also required)
- If priority bits (bits PRIO0-PRIO5) bits PRIO (bits 6-4: PSW), interrupt is enabled.

Instructions DI and EI are bit I manipulation instructions. DI sets bit I and EI clears the bit. These instructions must be associated with some settings as described in **3.5.3 Interrupt operation.**

When necessary to disable all maskable interrupts or DTS operations, clear bit I or bit D, respectively. Other disabling means cannot guarantee the operation of the microcomputer.

PSW bits are described in **2.3.3 PSW: processor status word.**

(2) INTREQ 0, 1: interrupt request registers 0 and 1

These registers latch maskable interrupt request.

Register	Symbol	Bit length	Address	SFR
Interrupt request register 0	INTREQ0	Word	00FFCCH	E6H

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit symbol	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Register	Symbol	Bit length	Address	SFR
Interrupt request register 1	INTREQ1	Word	00FFCEH	E7H

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
Bit symbol	-	-	-	-	-	-	-	-	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16

Bits in these registers are set as the corresponding 24 maskable interrupt request occurs. Bits INT0-INT23 correspond to interrupts INT0-INT23, respectively.

A bit set upon an interrupt request will be cleared as the request is accepted and the acknowledgement cycle starts. (Refer to **3.5.3 Interrupt operation ● Acknowledge cycle.**) Any bit can be cleared by writing 0 into that bit place from the user program. This cancels the corresponding interrupt being requested. In contrast, setting a bit to 1 simulates the corresponding interrupt.

When DTS process takes place, the corresponding interrupt request bit will be set at the end of the process (see **3.6 DTS**, *2) generating the DTS end interrupt. The function of this interrupt is same as other interrupt requests.

CAUTION

When the register is written (transfer instruction, e.g. MOV, or BCLR instruction), any coincidental interrupt request may be ignored.

(3) INTPRIO 0, 1: interrupt priority registers 0 and 1

These registers set the priority level of groups.

Register	Symbol	Bit length	Address	SFR
Interrupt priority register 0	INTPRIO0	Word	00FFC8H	E4H

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Initial value	-	0	0	0	-	0	0	0	-	0	0	0	-	0	0	0
Bit symbol	-	PRIO3			-	PRIO2			-	PRIO1			-	PRIO0		

Register	Symbol	Bit length	Address	SFR
Interrupt priority register 1	INTPRIO1	Word	00FFCAH	E5H

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	-	-	-	-	-	-	-	-	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Initial value	-	-	-	-	-	-	-	-	-	0	0	0	-	0	0	0
Bit symbol	-	-	-	-	-	-	-	-	-	PRIO5			-	PRIO4		

Interrupts are categorized into groups as shown in the table below. The interrupts in a group are given the same priority level. A set of three bits(PRIOx) in INTRPRIO 0 and 1 can have a value between 0 (lowest priority) to 7 (highest priority). Interrupts in a group having a value (represented by bit PRIOx) equal to or larger than that of bit PRIO (bits 6-4: PSW) can be accepted.

Bit PRIOx	Interrupts in group	
PRIO5	INT23, INT22, INT21, INT20	(interrupt group 5)
PRIO4	INT19, INT18, INT17, INT16	(interrupt group 4)
PRIO3	INT15, INT14, INT13, INT12	(interrupt group 3)
PRIO2	INT11, INT10, INT9, INT8	(interrupt group 2)
PRIO1	INT7, INT6, INT5, INT4	(interrupt group 1)
PRIO0	INT3, INT2, INT1, INT0	(interrupt group 0)

CAUTION

When setting the interrupt priority register (INTPRIO0 and INTPRIO1), first disable all maskable interrupts by using DI instruction (do not disable interrupt by clearing the interrupt request bit in the control registers of function block). Othewise, the interrupt controller will not operate as designed.

(4) DTSSEL 0, 1: DTS select registers 0 and 1

These registers select between normal interrupt process or DTS process upon an interrupt request.

Register	Symbol	Bit length	Address	SFR
DTS select register 0	DTSSEL0	Word	00FFC4H	E2H

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit symbol	DTS15	DTS14	DTS13	DTS12	DTS11	DTS10	DTS9	DTS8	DTS7	DTS6	DTS5	DTS4	DTS3	DTS2	DTS1	DTS0

Register	Symbol	Bit length	Address	SFR
DTS select register 1	DTSSEL1	Word	00FFC6H	E3H

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	-	-	-	-	-	-	-	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
Bit symbol	-	-	-	-	-	-	-	-	DTS23	DTS22	DTS21	DTS20	DTS19	DTS18	DTS17	DTS16

The bits in these registers correspond to the bits in the interrupt request registers INTREQ0 and INTREQ1. Bits DTS23-DTS0 correspond to bits INT23-INT0. When the master enable bit, bit D (bit 8: PSW) for DTS is set at 1, the value of the bits set in register DTSSEL0 and DTSSEL1 become valid.

When an interrupt request occurs while interrupts are enabled and bit D = 1, the interrupt controller checks the corresponding bit in register DTSSEL0 or DTSSEL1. When the bit is set at 1, the microcomputer starts the DTS.

When DTS completes the operation and the microcomputer returns to the normal operation, the corresponding bit in register DTSSEL0 or DTSSEL1 is cleared. At the same time, the corresponding bit in the interrupt request register INTREQ0 or INTREQ1 is set. This generates the DTS end interrupt which allows the microcomputer to process interrupt.

CAUTION

When setting the DTS select registers DTSSEL0 and DTSSEL1, first disable all maskable interrupts by using DI instruction (do not disable interrupts by clearing the interrupt request bits in the control registers of function block). Othewise, the interrupt controller will not operate as designed.

3.5.2 Types of interrupts

SM6010 interrupts are classified into the following types:

- Nonmaskable interrupt

  Software trap (undefined code detect, TRAP and TRAPV instructions)

  Hardware trap (watch dog timer interrupt, address error)

- Maskable interrupts (normal interrupts and DTS)

  Normal interrupt process (by software)

  DTS operation (automatic data transfer by DTS controller)

The user program can use the interrupts (except DTS) in the same manner.

When suitable user program is executed after the control goes to the interrupt vector and IRET instruction is executed at the end of the routine, the control returns to the normal routine.

Below details the interrupts.

● NMI(Non maskable Interrupt)

When NMIB signal is detected by its edge the control goes to the interrupt directly.

● Undefined code detect (nonmaskable interrupt, software trap)

When undefined code is detected upon instruction code fetch, the control goes to the interrupt service routine. Unlike normal interrupt process, no interrupt reqeust is generated. The codes shown below are undefined.

65H–67H, 6DH–6FH, D4H–D5H, DCH–DDH

The code 6FH is reserved and not a user code.

● TRAP and TRAPV instructions (nonmaskable interrupt, software trap)

Executing TRAP instruction generates a software like interrupt. Executing TRAPV instruction also generates software like interrupt if the register PSW V flag is set at 1. The microcomputer clears V flag before saving register PSW in the interrupt sequence. TRAPV instruction does not cause an interrupt if the V flag is not set.

In the same way as upon detecting an undefined code, the microcomputer moves the control directly to the interrupt service routine, when TRAP/TRAPV is used to generate an interrupt. **3.5.3 Interrupt operation** shows the operation flow.

- ### Watch dog timer interrupt (nonmaskable interrupt, hardware trap)

The watch dog timer generates a nonmaskable interrupt upon overflowing.

- ### Address error (nonmaskable interrupt, hardware trap)

If an odd address is used to access a word data, a nonmaskable interrupt is generated to indicate the address error.

● Normal interrupt process (maskable interrupt)

These 24 maskable interrupts can be individually enabled and disabled from the user program by setting interrupt enable bits of the control register in the priority registers INTRIO0 and INTRIO1. For setting of these registers, refer to **3.5.4 Interrupt priority.**

The bit I disables or enables all the maskable interrupts (bit I = 0, by DI instruction: disable; 1: by EI instruction, enable).

A maskable interrupt sets the corresponding bit of the interrupt request register INTREQ0 or INTREQ1 to 1. The bit is kept unchanged until the interrupt is handled or cleared by the user program.

● DTS process (maskable interrupt)

DTS handles 24 maskable interrupt requests. Each DTS operation can be enabled or disabled by setting bit D (bit 8: PSW) and DTS select registers DTSSEL0 and DTSSEL1, in additon to enable/diable setting for normal interrupts. The bit D diables or enables all DTS processes (D = 0: disable; 1: enable). When both interrupt enable and DTS enable are set, the microcomputer starts DTS upon occurrence of the corresponding interrupt request. For DTS operation, refer to **3.6 DTS**.

3.5.3 Interrupt operation

This section describes procedure to enable maskable interrupt and DTS, acknowledge cycle, and interrupt flow. Before or at least at the same time of enabling an interrupt, start the related functional blocks. The interrupt routine start address must be stored into the interrupt vector address and DTS vector address.

● Setting

The user program must perform the following settings.

- Interrupt priority (reference level for comparison)

  Set the priority order (0-7) at bit PRIO (bits 6-4: PSW).

- Interrupt priority (for each interrupt request)

  Set the priority bits INTRIO0 or INTRIO1 whichever correspond to the current interrupt group.

  For example, external interrupt EXINT2 (INT13) is in group 3, set the priority bit PRIO3 (bits 14-12: INTPRIO0) to the value equal to or larger than the value of PRIO (bits 6-4: PSW).

- Enable individual interrupts

  Set the interrupt enable bit in the control register of the desired functional block to 1. More than one bit is required to enable some interrupts. For example, external interrupt EXINT2 (INT13) requires setting of bits 7 and 6 of port 4 external interrupt register P41.

- Interrupt master enable bit (bit I)

  Executing EI instruction sets bit I to 1, simultaneously enabling all interrupts enabled individually.

When an interrupt request is issued, the microcomputer enters acknowledge cycle to start handling the interrupt. Before executing EI instruction, set bit I to 0. Interrupt related bits (bit I, bit PRIO, etc.) must be set by taking into considerations the effects of these bits.

Execute IRET instruction at the end of interrupt routine (specified condition match) to return the control to the normal operation.

Do not use other return instruction.

To use DTS from the user program, the following settings are required in addition to enable setting. Bit I must be set at 0.

- Set DTS master enable bit (bit D)

  Set bit D (bit 8: PSW) to 1.

- Set individual DTS

  Set the desired DTS bit in the DTS register DTSSEL0 or DTSSEL1 to 1. For example, if external interrupt EXINT2 (INT13), set bit DTS13 (bit 13: DTSSEL0) to 1.

With the above mentioned settings, when the interrupt reqeust is issued, the microcomputer starts DTS after the aknowledge cycles. While DTS is operating, the program counter PC stops and the CPU looks idling. Although bit I remains 1 while DTS is operating, no other interrupts are not detected but the corresponding interrupt request register bits are set.

When DTS completes the operation, the microcomputer clears the bit (in DTSSEL0 or DTSSEL1) corresponding to that DTS. Then, the microcomputer sets the bit (of the interrupt request register INTREQ0 or INTREQ1) corresponding to that DTS to generate the interrupt request. If necessary, write the process upon completion of DTS operation into an interrupt routine. If interrupt requests having higher priority than DTS interrupt have been issued, these interrupts will be handled in the order of priority.

● Acknowledge cycle

The CPU detects interrupt requests every instruction code fetch. The CPU detects interrupt requests every instruction code fetch. When it detects an interrupt request, the microcomputer follows the predetermined flow to judge what process it must take. When detected interrupt requests are legal, it selects the highest priority interrupt request and enters the acknowledge cycle which is the time delay from the issue of interrupt request to interrupt vector address fetch. During the cycle the microcomputer proceeds to the following processes.

(1) Checks whether the interrupt is to be handled as standard interrupt or DTS. If non-DTS interrupt request, the microcomputer proceeds to steps (2) to (5). Otherwise, steps (6)-(8).

(2) Saves (pushes) the content of the program counter PC, segment register SEG and register PSW into the stack.

(3) Reads the interrupt service routine address from the vector area corresponding to the interrupt and updates the content of program counter PC and segment register SEG. That is, if the user program has stored the interrupt routine start address into the interrupt vector address, the program starts at this address after the acknowledge cycles elapsed.

(4) Clears the interrupt request register bit corresponding to the processed interrupt and bit I (bit 9: PSW).

(5) Moves the control to the interrupt handling routine.

(6) Reads the DTS control block from the corresponding DTS vector area.

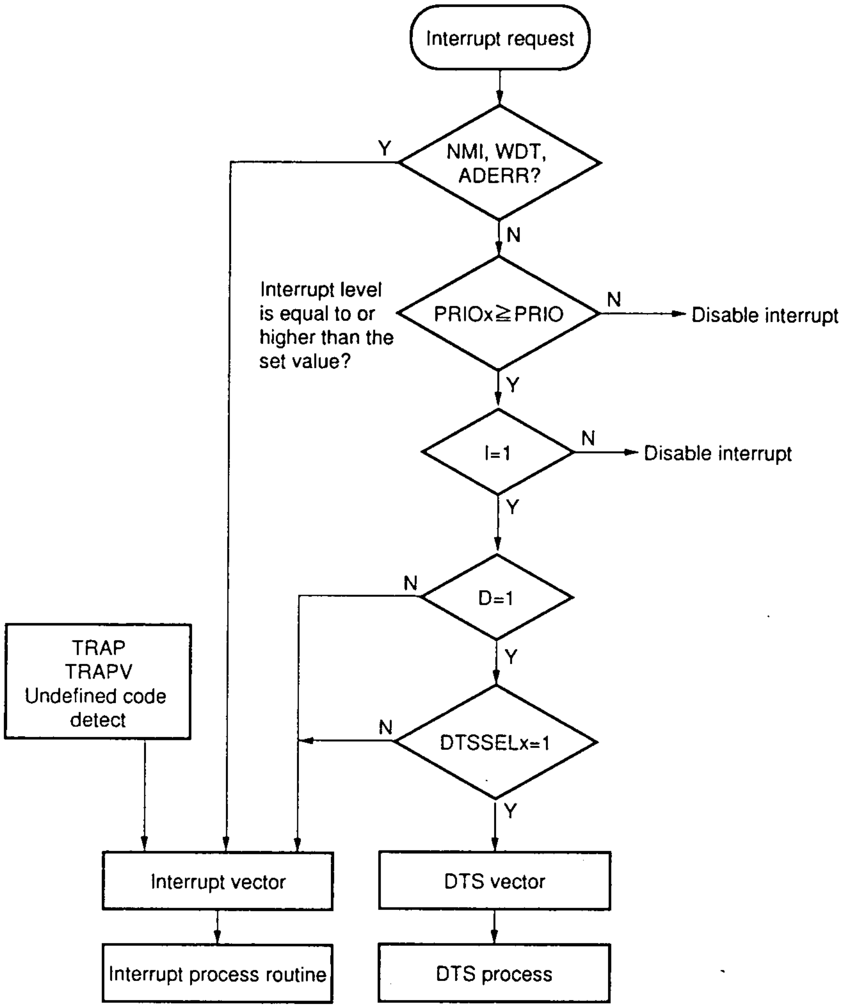
(7) Clears the interrupt request register bit corresponding to that interrupt request.

(8) Starts data transfer process according to the content of the read DTS control block. During the transfer, the program stops. Any coming interrupt requests are not detected but set the corresponding bits of the interrupt request registers. These interrupt requests are suspended until DTS completes the operation.

To return from an interrupt routine other than DTS, use IRET instruction. If the specified conditions are met, pop operation takes place which is just the reversal of push operation in step (2) above.

- Operation flow to interrupt handling

Fig. 3-9 shows the operation flow from interrupt request to the start of the interrupt handling process.



■Symbol
NMI: external NMI input (not supported by SM6000)
WDT: watch dog timer interrupt
ADERR: address error (odd illegal address access)
PRIO: bit PRIO (bits 6-4: PSW)
PRIOx: setting of interrupt priority register INTPRIO 0 or 1, which corresponds to the generated interrupt
DTSSELx: setting of DTS select register DTSSEL 0 or 1, which corresponds to the generated interrupt
I: bit I (bit 9: PSW)
D: bit D (bit 8: PSW)

Fig. 3-9 Interrupt process flow

When a software trap occurs due to detection of undefined code or TRAP or TRAPV instruction, the microcomputer immediately moves the control to the interrupt vector without checking the type of the interrupt.

The flow chart shown in **Fig. 3-9** is to illustrate interrupt handling sequence as a reference for writing the user program. In the actual process, unlike the flow chart, the internal hardware gives DTS process the highest priority for branching.

3.5.4 Interrupt priority

Software traps such as undefined code detect and TRAP and TRAPV instruction have no priority order. When the CPU fetches these instruction codes, it moves the control to the interrupt handling routine.

Hardware traps such as watch dog timer interrupt and address error have higher priority over the maskable interrupts.

INT0-INT23 are 24 maskable interrupt events which are classified by the CPU into the following 6 groups, each 4 events.

- INTs23-20, INTs19-16, INTs15-12, INTs11-8, INTs7-4, INTs3-0

Each group can be given one of 7 priority levels which are set at the bits located in the setting registers INTPRIO0 and INTPRIO1. For example, bit PRIO5 (bits 6-4: INTPRIO1) sets the priority level of INTs 23-20.

The priority level reference is set at bit PRIO (bits 6-4: PSW) which can have a value from 0 to 7. When a maskable interrupt request is issued while it is enable, the microcomputer compares the bit PRIO with the priority bit for that interrupt and goes to the interrupt handling sequence if the value of bit PRIO meets the following conditions.

- Priority bit (bits PRIO0-PRIO5) ≥ bit PRIO (bits 6-4: PSW)

If all groups are at the same priority level, then priority levels are set as follows:

- INT23-20 > INT19-16 > INT15-12 > INT11-8 > INT7-4 > INT3-0

This is the case when all bits in registers INTPRIO0 and INTPRIO1 are cleared to 0 after the hardware/software reset sequence. Priority levels in a group are fixed as shown below, and cannot be changed.

Group 5	INT23 > INT22 > INT21 > INT20
Group 4	INT19 > INT18 > INT17 > INT16
Group 3	INT15 > INT14 > INT13 > INT12
Group 2	INT11 > INT10 > INT9 > INT8
Group 1	INT7 > INT6 > INT5 > INT4
Group 0	INT3 > INT2 > INT1 > INT0

When the CPU detects interrupts, the interrupt controller accepts the highest interrupt first.

3.5.5 Interrupt operation considerations

- Inhibit maskable interrupts during clock change

Before changing the system clock, disable all maskable interrupts by executing DI instruction to set bit I to 0. Do not disalbe the interrupts by clearing the interrupt request bits in the control registers of functional block.

- Registers INTPRIO0 and INTPRIO1 writing guideline

Before changing the content of interrupt priority registers INTPRIO0 and INTPRIO1, disable all maskable interrupts by executing DI instruction to set bit I to 0. Do not disalbe the interrupts by clearing the interrupt request bits in the control registers of functional block.

- Interrupt request registers INTREQ0 and INTREQ1 writing guideline

Any interrupt request coming during the interrupt request register INTREQ0 or INTREQ1 writing cycles (transfer instructions like MOV or BCLR instruction) is ignored and the corresponding interrupt register bit may not be set. Write into the interrupt request registers INTREQ0 and INTREQ1 only when any coming interrupt can be ignored.

3.6 DTS

DTS (Data Transfer Server, also referred to as automatic transfer server) transfers various data upon specific interrupt.

The DTS automatically transfers data according to the data stored in the DTS control block *1 (Refer to **3.6.2 Operation of DTS**). DTS transfers data between memory and I/O, between memories, between I/Os.

These I/Os are registers (control registers and port registers) located in the address space and in the SFR area. Up to 255 words of block can be transferred at a DTS cycle *2.

The DTS is triggered by an interrupt request. The user program can select the interrupts which trigger the DTS. Any maskable interrupt event can be used for DTS transfer. At the end of a set of transfers, DTS generates the end interrupt.The DTS can be operated to function like DMA data transfer between memories, but more precisely controlled than DMA if the destination of data is set into the control register of the function block or data register. Fore example, DTS can be used to transfer continuous A/D converter data and SCI data.

This section describes the data to be set in the DTS control block and DTS operation. For DTS enable setting and operation flow to DTS operation, refer to **3.5 Interrupt**.

Fig. 3-10 shows the DTS control block.

Offset value		
+8H	BLK-COUNTER※	Upper address
+7H	SRC-SEG (segment)	↑
+6H	DST-SEG (segment)	
+5H	SRC-ADRS (upper address)	
+4H	SRC-ADRS (lower address)	
+3H	DST-ADRS (upper address)	
+2H	DST-ADRS (lower address)	
+1H	CONTROL	
Address denoted by DTS vector (set only even addresses) → +0H	COUNTER	Lower address

※ Use only in the block transfer mode.

Fig. 3-10 DTS control block

Symbols representing DTS control block may be correctively expressed in this manual as follows:

- SRC-SFG, SRC-ADRS → SRC or SRC area
- DST-SFG, DST-ADRS → DST or DST area

*1 *DTS control data is stored in the DTS control block, 8-9 byte memory area. The user can allocate the DTS control block in any virtual memory area e.g. RAM. The start address of the memory block is written in the DTS vector. Data is stored in the DTS control block in the same way as RAM writing.*

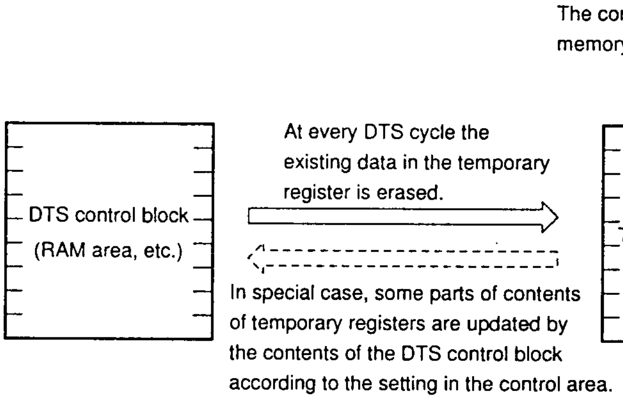
*2 *The time required for the control to start DTS operation and return to the normal operation is defined as DTS cycle. A set of DTS cycles required to generate the end DTS interrupt request is called DTS set. The end DTS interrupt request operates in the same way as normal interrupt requests.*

3.6.1 DTS control block

The DTS area is a virtual space allocated on RAM. It functions as a control register, address pointer or counter. Below describes the DTS control block.

In 3.6 **DTS**, the words "source address" and "destination address" are used. These words represent the data read from the DTS control block into temporary registers (non-user registers) but not the data stored in the DTS control block.

The DTS controller reads the contents of the DTS control block into the temporary register every DTS cycle. The DTS controller uses these temporary registers as the memory pointer of the source and destination. The DTS controller does not directly use the data in the DTS control block as memory pointers. The user program (by setting data in the control area) determines whether the contents of the temporary registers, which have changed during a DTS cycle, are to be returned back to the DTS control block.


The contents are used as
memory pointer and control data.
At every DTS cycle the
existing data in the temporary
register is erased.
DTS control block
(RAM area, etc.)
Temporary register
In special case, some parts of contents
of temporary registers are updated by
the contents of the DTS control block
according to the setting in the control area.

(1) COUNTER area: DTS count

The COUNTER area is an 8-bit memory space whose content determines the number of DTS cycles. Then content is decremented by one at the end of every DTS cycle. When the content reaches 00H, the corresponding bit in DTS select register DTSSEL0 or DTSSEL1 is cleared. At the same time, the corresponding bit in the interrupt request register INTREQ0 or INTREQ1 is set, indicating the end of DTS operation and causing the user program to start the interrupt handling process.

(2) CONTROL area: DTS control

CONTROL area is an 8-bit memory space whose content controls the DTS.

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	-	-	-	-	-	-	-	-
Bit symbol	MODE			B/W	SU	DU	SI	DI

These bits have the following functions.

■ Bits 7-5: DTS mode set (bit MODE)

These bits set the DTS operation mode.

Bit MODE			Operation mode
*	0	0	Single transfer (byte/word)
*	0	1	Block transfer (byte/word)
*	1	0	Bidirectional transfer (byte/word)
*	1	1	Don't set.

■ Bit 4: byte/word transfer select (bit B/W)

Bit B/W	No. of transferred bits
0	Byte transfer
1	Word transfer

This bit affects the unit of address increments set by the bits SI and DI described later. If bits SI and DI are set for increment, increment is by 1 (1 byte) in byte transer mode, and by 2 (2 bytes) in word transfer.

■ Bit 3: update source memory (bit SU)

Bit SU	Operation
0	Not update SRC-ADRS area at the end of DTS cycle
1	Update SRC-ADRS area at the end of DTS cycle (Store the source address into SRC-ADRS area)

When DTS is to operate, the microcomputer stores the content of SRC-ADRS area into the DTS controller temporary register. The content of the temporary register is used as the memory pointer of the source address.

When bit SU is set at 1, the microcomputer returns the working temporary register data to SRC-ADRS area at the end of DTS cycle, updating the data in this area.

In contrast, if SU = 0, data in the SRC-ADRS area set by the user program is used repeatedly in every DTS cycle as the source address.

Bit 2: update destination memory (bit DU)

Bit DU	Operation
0	Not update DST-ADRS area at the end of DTS cycle
1	Update DST-ADRS area at the end of DTS cycle (Store the destination address into DST-ADRS area)

In the same way as bit SU, the microcomputer stores the content of DST-ADRS area into the temporary register of DTS controller. The temporary register is used as the memory pointer of the destination address during DTS operation. When bit DU is set at 1, the microcomputer returns the working temporary register data to DST -ADRS area, updating the data in this area.

In contrast, if DU = 0, data in the DST-ADRS area set by the user program is used repeatedly in every DTS cycle as the destination address.

Bit 1: source address increment (bit SI)

Bit SI	Operation
0	Not increment the source address
1	Increment the source address at the end of transfer (by 1 in byte transfer, 2 in word transfer)

With SI = 1, the microcomputer increments the source address by 1 at the end of DTS transfer: in block transfer, the number of transfers and in bidirectional transfer, by 2.

Bit SU determines whether the SRC-ADRS area is updated or not at the end of DTS cycle.

Bit 0: destination increment (bit DI)

Bit DI	Operation
0	Not increment the destination address
1	Increment the destination address at the end of transfer (by 1 in byte transfer, 2 in word transfer)

With DI = 1, the microcomputer increments the destination address by 1 at the end of DTS transfer: in block transfer, the number of transfers and in bidirectional transfer, by 2.

Bit DU determines whether the DST-ADRS area is updated or not at the end of DTS cycle.

(3) DST-ADRS area: DTS destination (2 bytes)

DST-ADRS area is a 16-bit memory space whose content is read into the temporary register of the DTS controller before DTS starts operation and used as the destination address.

(4) SRC-ADRS area: DTS source (2 bytes)

SRC-ADRS area is a 16-bit memory space whose content is read into the temporary register of the DTS controller before DTS starts operation and used as the source address.

(5) DST-SEG area: DTS destination (segment data)

DST-SEG area is an 8-bit memory space whose content is the destination address (segment data). Overflow caused by increments in DTS-ADRS area will not change the data in the DST-SEG area. DTS operation will not change the content of DST-SEG area.

Do not increase the source address until it exceeds the size of the segment.

(6) SRC-SEG area: DTS source (segment data)

SRC-SEG area is an 8-bit memory space whose content is the source address (segment data). Overflow caused by increments in SRC-ADRS area will not change the data in the SRC-SEG area. DTS operation will not change the content of SRC-SEG area.

Do not increase the source address until it exceeds the size of the segment.

(7) BLK-COUNTER area: DTS block transfer count

BLK-COUNTER area is an 8-bit memory space to which the number of block transfers is stored. BLK-COUNTER area is used only in the block transfer mode. In the other modes than the transfer mode, BLK-COUNTER area can be used for other purpose (e.g. the start address of other DTS control blocks).

3.6.2 Operation of DTS

- Status of microcomputer during DTS process

When an interrupt request enabled for DTS operation occurs, the microcomputer starts DTS process instead of transferring the control to the interrupt vector. See **3.5.3 Interrupt operation.**

The duration from the starting of DTS process to the returning to the normal operation is equal to one DTS cycle.

During the DTS cycle the microcomputer performs special process without running the program. Since peripheral function blocks are operating, they will start operation as the corresponding functional block operation data is transferred to the control register, etc. by DTS process.

Clocks are kept supplied to these functional blocks to assure correct count operation of all functional blocks. If the user program uses the instruction execution cycle as operating timing, the DTS cycle causes operation error.

If an interrupt request other than DTS interrupt is issued during DTS processing, the bit in the corresponding interrupt request register INTREQ0 or INTREQ1 is set to 1. But this interrupt request is not handled during the DTS cycle.

After one DTS cycle the microcomputer returns back to the normal operation. If enabled interrupt requests have issued during the DTS cycle, the microcomputer processes these interrupt requests in the order of priority. When the DTS enabling interrupt occurs again, the microcomputer repeats the DTS cycle. The number of DTS cycles (1-255) are preset in the DTS control block counter area. At the end of set number of DTS cycles, corresponding bit of the interrupt request register (INTREQ0 or INTREQ1) is set to 1 indicating the end of 1 DTS set. The waiting interrupt request(s) is recognized and the program control moves to the vector corresponding to the interrupt request. The user program corresponding to the end of 1 DTS set can start with this vector.

At the end of 1 DTS set, the highest priority interrupt is processed first. Other interrupt must wait until the higher priority interrupt(s) has been processed.

● Transfer mode

The DTS operates in the following three transfer modes.

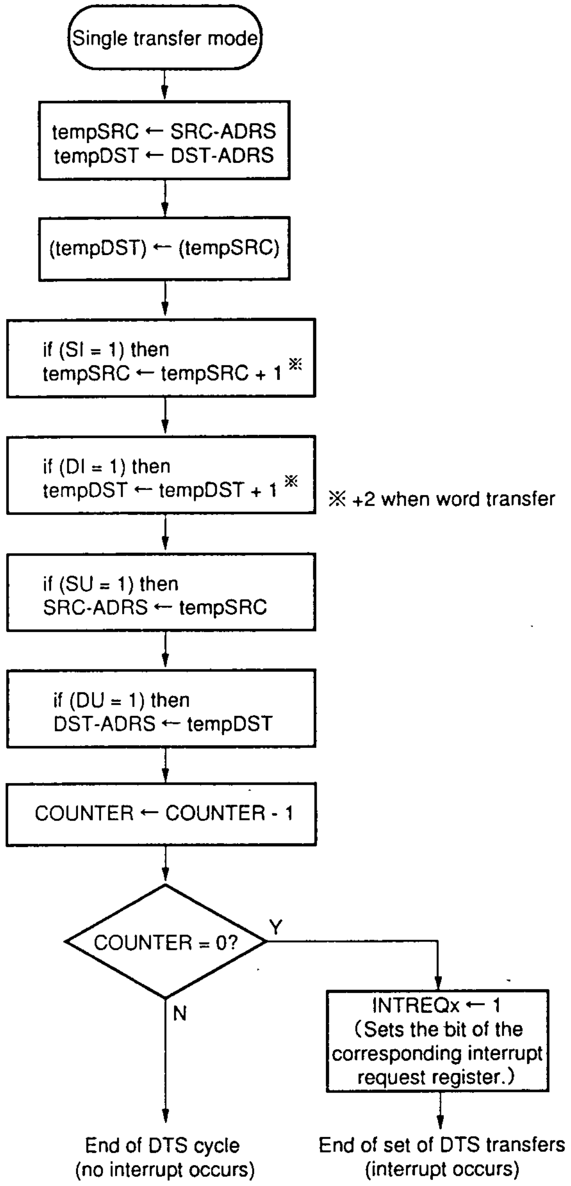
- Single transfer mode
- Block transfer mode
- Bidirectional transfer mode

Set the desired mode by storing the suitable data into the control area. Data transfer between memory I/Os, between memories and between I/Os are possible in any mode.

When reading operation mode descriptions (a), (b) and (c) below, also refer to **3.6.1 DTS control block** which will help to understand the flowcharts.

(a) Single transfer mode

In the single transfer mode, DTS performs one transfer per DTS cycle and issues DTS end interrupt request upon end of one set of DTS transfer cycles (the number of transfers set in COUNTER area).



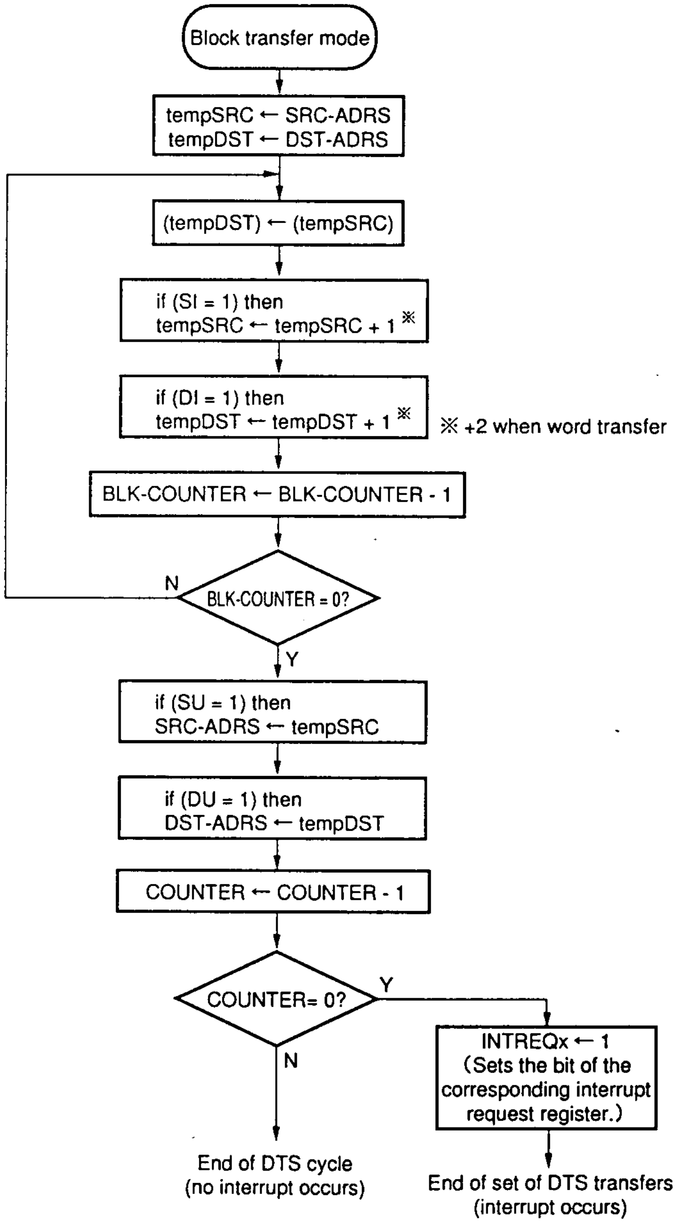
■ Symbol

tempSRC : temporary register (source address memory pointer)
tempDST : temporary register (destination address memory pointer)
(tempSRC) : memory denoted by tempSRC
(tempDST) : memory denoted by tempDST
SI (DI) : source (destination) increment flag (resides in CONTROL area)
SU (DU) : source (destination) update flag (resides in CONTROL area)

Fig. 3-11 Single transfer flow chart

(b) Block transfer mode

In the block transfer mode, DTS performs number of transfers (the number set in BLK-COUNTER area) per DTS cycle and issues DTS end interrupt request upon end of one set of DTS transfer cycles (the number of transfers set in COUNTER area).



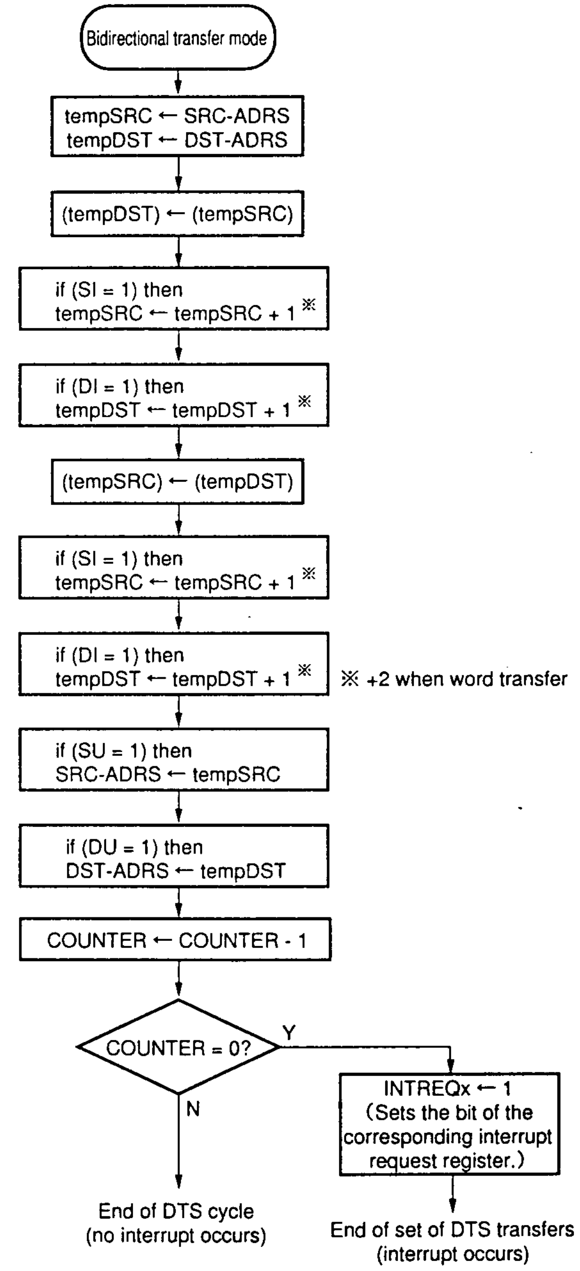
■ Symbol

tempSRC : temporary register (source address memory pointer)
tempDST : temporary register (destination address memory pointer)
(tempSRC) : memory denoted by tempSRC
(tempDST) : memory denoted by tempDST
SI (DI) : source (destination) increment flag (resides in CONTROL area)
SU (DU) : source (destination) update flag (resides in CONTROL area)

Fig. 3-12 Block transfer flow chart

(c) Bidirectional transfer mode

In the bidirectional transfer mode, DTS performs two transfers per DTS cycle (from the source to the destination, and from the destination to the source) and issues DTS end interrupt request upon completion of the number of DTS cycles (the number of transfers set in COUNTER area).

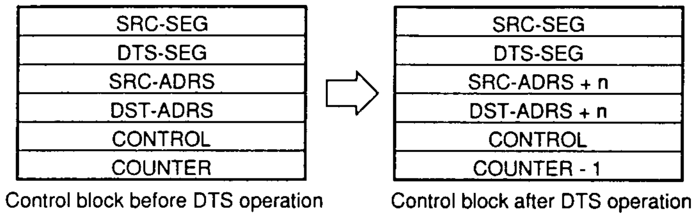


■ Symbol

tempSRC : temporary register (source address memory pointer)
tempDST : temporary register (destination address memory pointer)
(tempSRC) : memory denoted by tempSRC
(tempDST) : memory denoted by tempDST
SI (DI) : source (destination) increment flag (resides in CONTROL area)
SU (DU) : source (destination) update flag (resides in CONTROL area)

Fig. 3-13 Bidirectional transfer flow chart

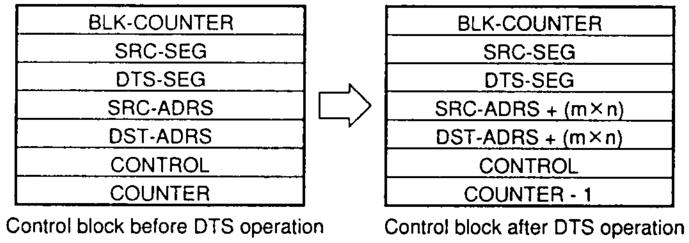
The content of the DTS control block after one DTS cycle depends on the operation mode, as shown in **Figs. 3-14 to 3-16.**



n:0 or 1 or 2※
n=0:
No increment
n=1:
Increment
(byte transfer)
n=2:
Increment
(word transfer)

※ The value when update is enabled by bits SU and SI.

Fig. 3-14 Content of DTS control block in the single transfer mode

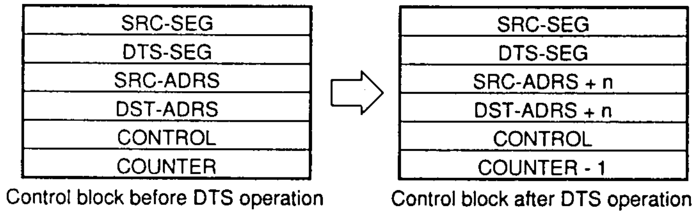


m:counts set in BLK-COUNTER area

n:0 or 1 or 2※
n=0:
No increment
n=1:
Increment
(byte transfer)
n=2:
Increment
(word transfer)

※ The value when update is enabled by bits SU and SI.

Fig. 3-15 Content of DTS control block in the block transfer mode



n:0 or 2 or 4※
n=0:
No increment
n=2:
Increment
(byte transfer)
n=4:
Increment
(word transfer)

※ The value when update is enabled by bits SU and SI.

Fig. 3-16 Content of DTS control block in the bidirectional transfer mode

3.6.3 Example of DTS operation

Figure 3-17 shows data storage into memory when DTS operates in the bidirectional mode while the A/D converter is successively operating. (A/D conversion mode) The operating conditions are as follows.

- DTS is triggered at the end of A/D conversion (register INTREQ0 bit 0 = 1)
- A set of DTS cycles is 10 (COUNTER area ← 0AH)
- Use a move instruction e.g. MOV to store the first control data to the control register ADCC to start the A/D converter. The 11th control data must be dummy (not start the A/D conversion).
- Bits SI, DI and DU in the CONTROL area are set at "1". That is, the source address is incremented during DTS cycle but not updated. The destination address is incremented and updated.

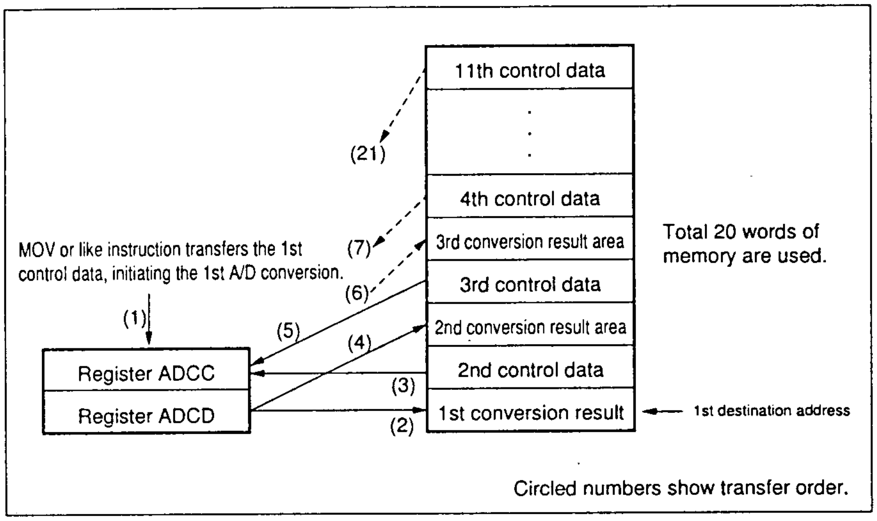

11th control data
(21)
4th control data
MOV or like instruction transfers the 1st control data, initiating the 1st A/D conversion.
(7)
3rd conversion result area
Total 20 words of memory are used.
(6)
3rd control data
(1)
(5)
(4)
2nd conversion result area
Register ADCC
2nd control data
(3)
Register ADCD
1st conversion result
(2)
1st destination address
Circled numbers show transfer order.

Fig. 3-17 Memory mapping during DTS operation

Before starting A/D conversion, the user program should store data into 2nd to 11th control data storage memories. Because the DTS operates in bidirectional mode, DTS ends its operation after sending the 11th control data to the control register ADCC. The 11th control data must be a dummy value not allow the D/A converter to continue.

The A/D converter starts upon storing of the 1st control data into the register ADCC. At the end of A/D conversion, DTS starts operation. During DTS cycle, the conversion result stored in the data register ADCD is transferred to the conversion result area shown in **Fig. 3-17.** The register ADCC is filled with the control data (next A/D conversion start data). Total 10 A/D conversions: upon starting of A/D conversion and 9 conversions during one set of DTS.

At the end of 10th DTS cycle, the interrupt request occurs to indicate the end of A/D conversion (end of a set of DTS). This interrupt request may not be handled first because the microcomputer handles the interrupts in the order of priority level.

3.6.4 Considerations on DTS operation

- DTS source and destination counts cannot exceed the segment.

During DTS operation, DST-ADRS and SRC-ADRS areas can be incremented but the segment data cannot be incremented. When setting these areas keep them in the segement even after increment.

- DTS forces the program to stop

The program stops during DTS cycles, disabling handling of new interrupt. However, bit of the interrupt request register (INTREQ0, INTREQ1) corresponding to the new interrupt is set. Peripheral functional blocks remain active during DTS cycles.

- Process after DTS operation is in the order of interrupt priority

The SM6010 handles the interrupts in the order of priority level. That is, DTS end interrupt may not be handled first.

- Set even address for the source and destination for word transfer

To allow the DTS to transfer word, set the source and destination to even address. Odd address will not ensure operation of the microcomputer. Odd address has no problem with byte transfer.

- Counts set in COUNTER area (not to 00H)

Set the number of DTS operations in COUNTER area. For example, set 01H into COUNTER area when DTS is to operate once. If 00H is set, a borrow will be generated from COUNTER area, which will not ensure operation of microcomputer.

3.7 Standby function

Standby is a feature to force the CPU to temporarily pause the program run to save power.

Standby mode includes halt mode and stop mode, one of which can be selected at a time.

Executing HALT or STOP instruction while in the operation mode enters the halt or stop mode selected. Any interrupt event can be used to return from the standby mode *. For whole operation of the SM6010, refer to **3.3 Operations.**

Table 3-6 shows entering/exiting standby mode, and status of functional blocks during standby mode.

- *Interrupt events related to the functional blocks which do not operate while in the standby mode cannot be used for the purpose of returning from the standby mode. (See* ***Table 3-6.****)*

Table 3-6 System status in the standby mode

<table>
<tr><th colspan="2">Standby mode</th><th>Halt</th><th>Stop</th></tr>
<tr><td colspan="2">To enter</td><td>Execute Halt instruction</td><td>Execute Stop instruction</td></tr>
<tr><td colspan="2">To return</td><td>Use external reset, or interrupt request ※1</td><td>Use external reset, or interrupt request ※1</td></tr>
<tr><td colspan="2">Main clock</td><td>Continues</td><td>Stops</td></tr>
<tr><td rowspan="10">Functional blocks</td><td>CPU (system clock)</td><td colspan="2">Stops</td></tr>
<tr><td>register</td><td colspan="2">Retained ※2</td></tr>
<tr><td>I/O ports</td><td colspan="2">Retained</td></tr>
<tr><td>Capture trigger</td><td>Operates</td><td>Stops</td></tr>
<tr><td>WDT</td><td>Operates</td><td>Stops</td></tr>
<tr><td>PWM</td><td>Operates</td><td>Stops</td></tr>
<tr><td>A/D converter</td><td>Disabled</td><td>Stops</td></tr>
<tr><td>SCI</td><td>Operates</td><td>Stops</td></tr>
<tr><td>RTC</td><td>Operates</td><td>Operates</td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

※1 Interrupt events related to the functional blocks which do not operate while in the standby mode cannot be used for the purpose returning from the standby mode.
Interrupt requests include DTS interrupt which causes the program to continue at the address immediately following the address of halt or stop instruction whichever executed, at the end of 1 DTS cycle after returning from the standby mode.
Only external interrupt events (EXINT0-EXINT7)&RTC interrupt can be used to return from the stop mode.

※2 Data in the general purpose registers, control register are retained except for data in the blocks (e.g. interrupt request registers INTREQ0 and INTREQ1) which can

3.7.1 Entering standby mode

From the user program, enable the desired event which is used to return the system from the standby mode. This is similar to to enable an interrupt *1. Next, execute Halt or Stop instruction. Executing the halt instruction enters the halt mode, and executing the stop instruction enters the stop mode *2.

Executing the halt instruction enters the halt mode. Do not enter the halt mode while the A/D converter is running. Once in the halt mode, the system clock stops and the CPU stops the user program. The main clock continues allowing the blocks operating from divided-by-n main clock to operate.

Executing the stop instruction enters the stop mode. Once in the stop mode the main clock stops. This causes all the blocks operating from the main clock as well as the user program to stop.

*1 *Some functional blocks won't operate or disabled while in halt or stop mode. Interrupt events related to these blocks cannot be used as halt/stop release event. Applicable standby release interrupt events depend on the mode (halt or stop).*

*2 *When in the standby mode release condition, the microcomputer won't enter the standby mode. Instead, it handles the interrupt which establishes the standby mode release condition.*

3.7.2 Returning from standby mode

(1) External reset input

An external reset input starts the hardware reset sequence which first allows the system to return from the halt/stop mode, and the program starts at address 000100H. For the external reset input, refer to **3.4.1 (1) Hardware reset.**

(2) Interrupt

When an enabled interrupt request is issued, the CPU returns from the halt/stop mode: in the halt mode, the interrupt request allows immediate return to the normal operation mode; in the stop mode, it allows returning to the normal operation mode after the warming up period (2^{18} main clocks). For the standby mode release timing, refer to **3.7.3 Standby mode release timing.**

After returning to the normal operation, run the user program in the same way as normal interrupt processing. Execute IRET instruction at the end of interrupt processing routine. The instruction forces the user program to go to the address next to the halt or stop instruction address.

3.7.3 Standby mode release timing

Fig. 3-18 shows halt mode release timing and **Fig. 3-19** stop mode release timing.

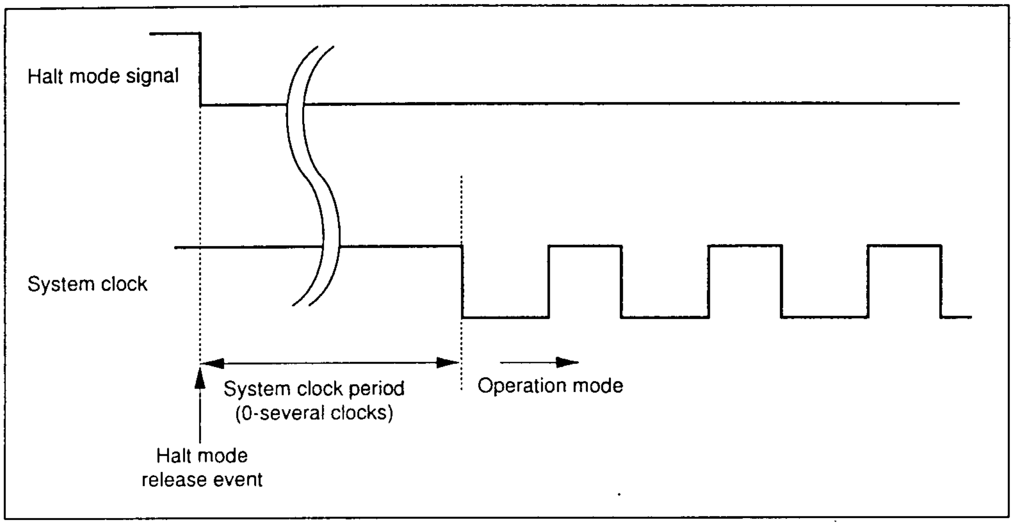


Fig. 3-18 Halt mode release timing chart

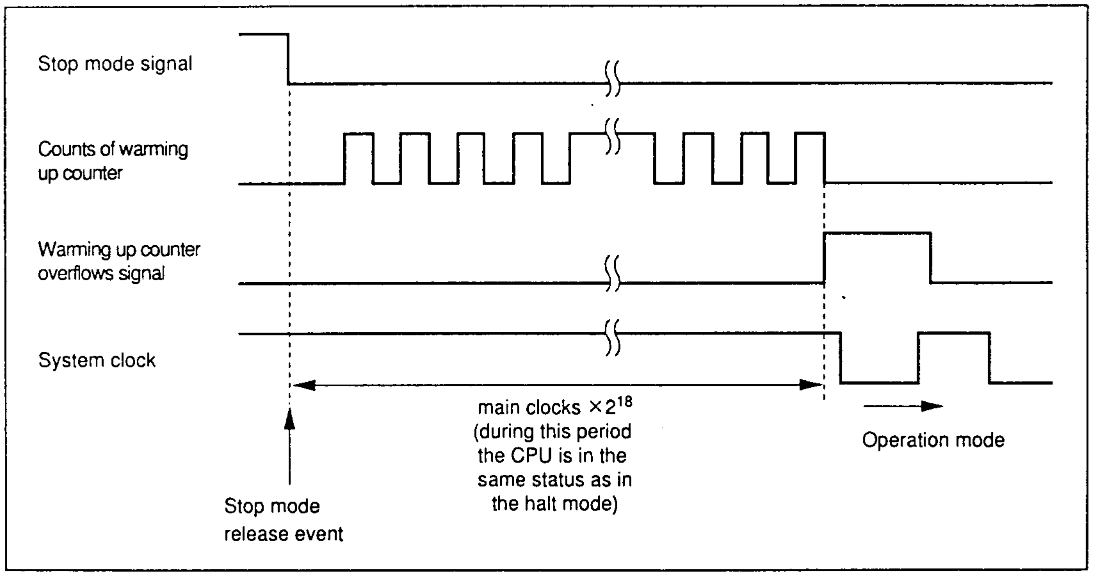


Fig. 3-19 Stop mode release timing chart

3.7.4 Standby function considerations

● Selection between halt mode and stop mode

The halt mode allows the system to return to the normal operation mode upon occuring of the halt mode release event. When keys are to be used frequently, select the halt mode. All interrupt events (except for A/D converter, illegal instruction, software trap) can be used as a releasing trigger.

Returning from the stop mode requires the warming up period after the stop mode releasing event occurs. A functional block which uses the main clock cannot be used as a stop mode release event.

Only external interrupt inputs (EXINT0-EXINT7) can be used as a stop mode release event.

When quick return to the normal operation mode is not required, select the stop mode which more effectively saves power than halt mode.

● Establish low current condition before entering standby mode

Setting of I/O ports and level on the output pins are retained during the standby mode. Before entering standby mode, the user program must put these ports and pins to the status so that they require minimum current.

● Do not enter the halt mode while A/D converter is running

Internal clocks are supplied to the peripheral blocks while in the halt mode. Therefore, the A/D converter can also run in the halt mode but its operation is not guaranteed if it has operated before execution of the halt instruction.

(1) Timing diagram - Address/data bus nonmultiplex, 2 cycle 8-bit bus size

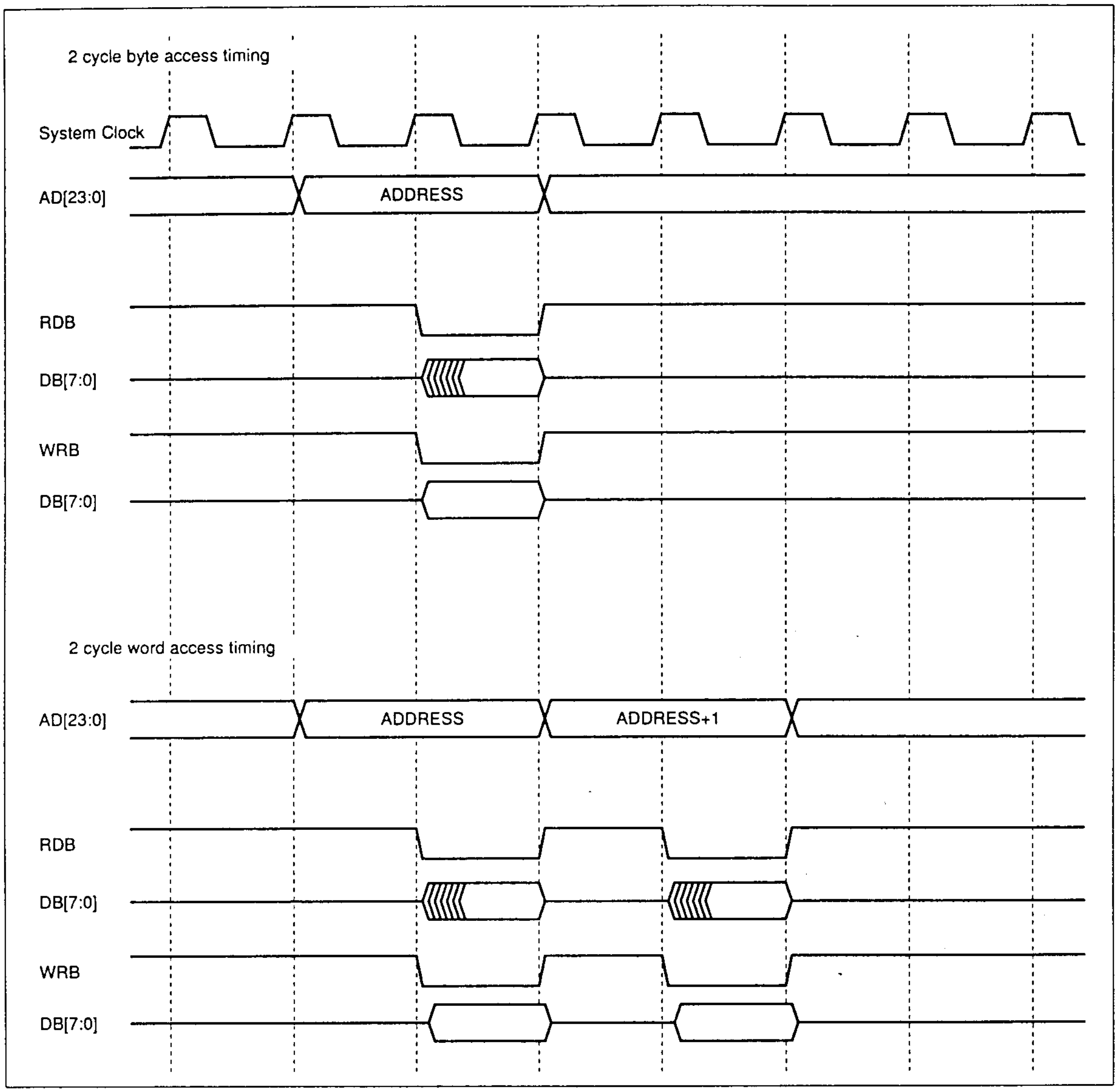


Fig. 3-21 Address/data bus nonmultiplex, 2 cycle 8-bit bus size

(2) Timing diagram - Address/data bus nonmultiplex, 2 cycle 16-bit bus size

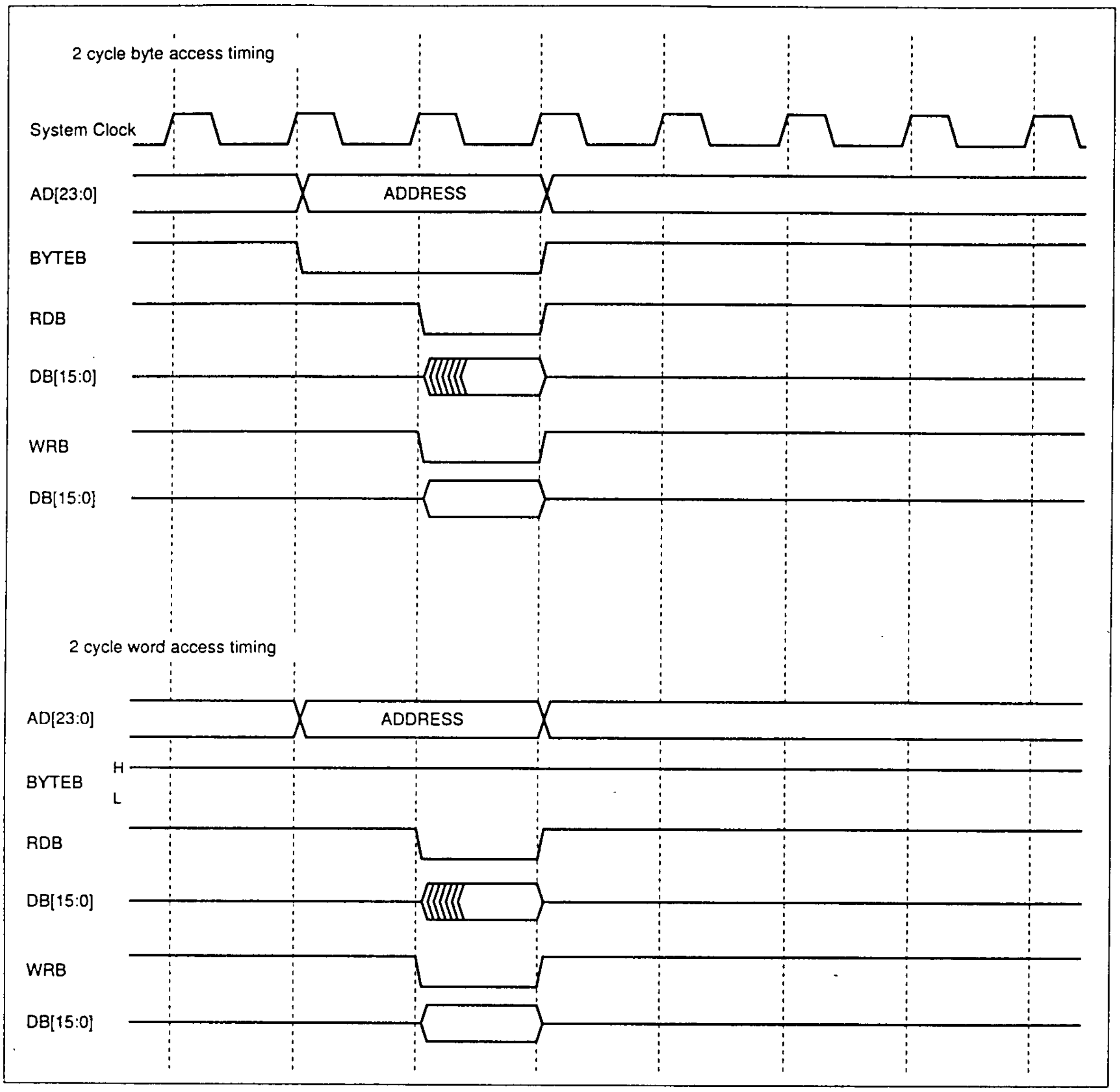


Fig. 3-22 Address/data bus nonmultiplex, 2 cycle 16-bit bus size

(3) Timing diagram - Address/data bus nonmultiplex, 3 cycle 8-bit bus size

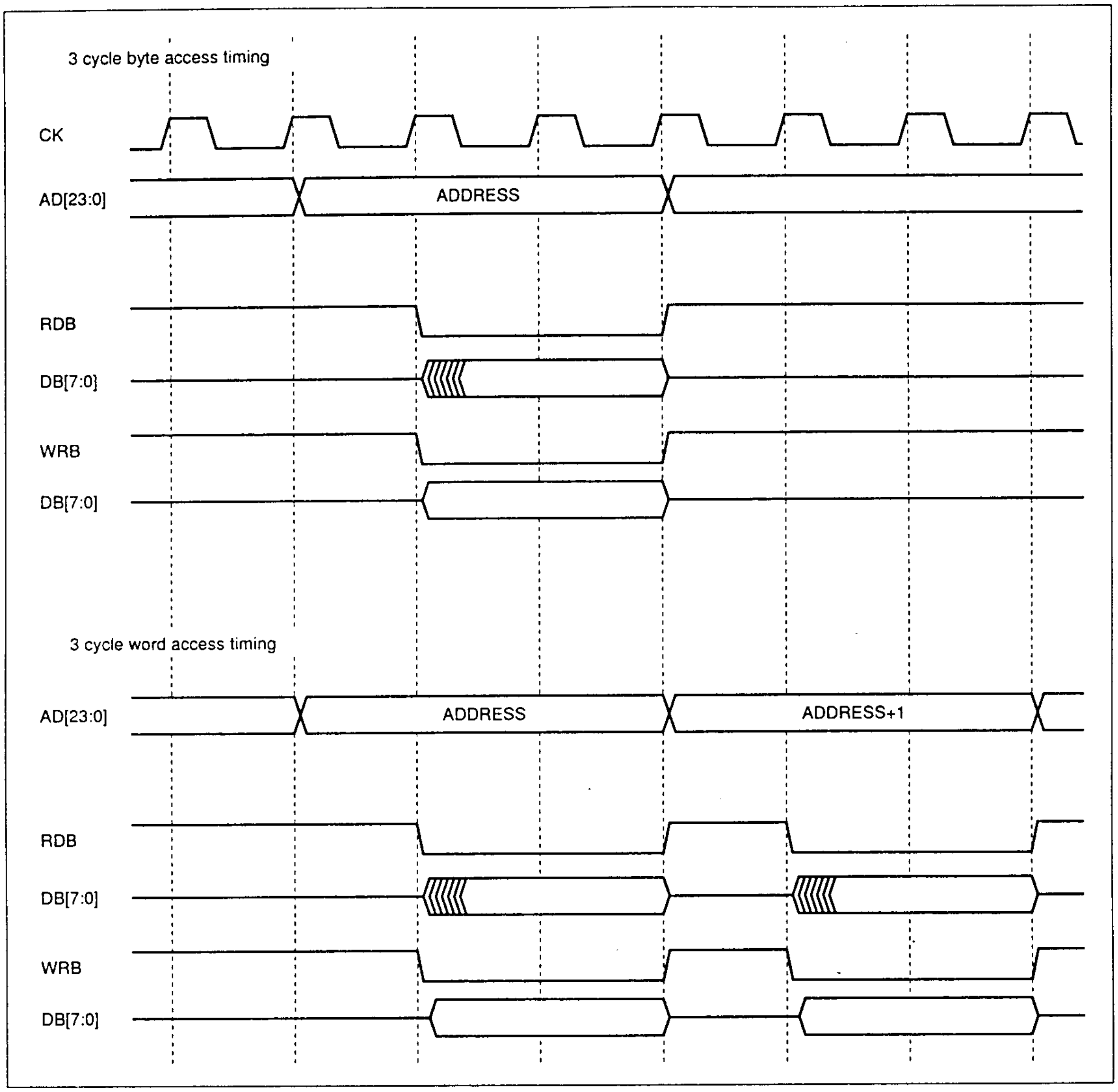


Fig. 3-23 Address/data bus nonmultiplex, 3 cycle 8-bit bus size

(4) Timing diagram - Address/data bus nonmultiplex, 3 cycle 16-bit bus size

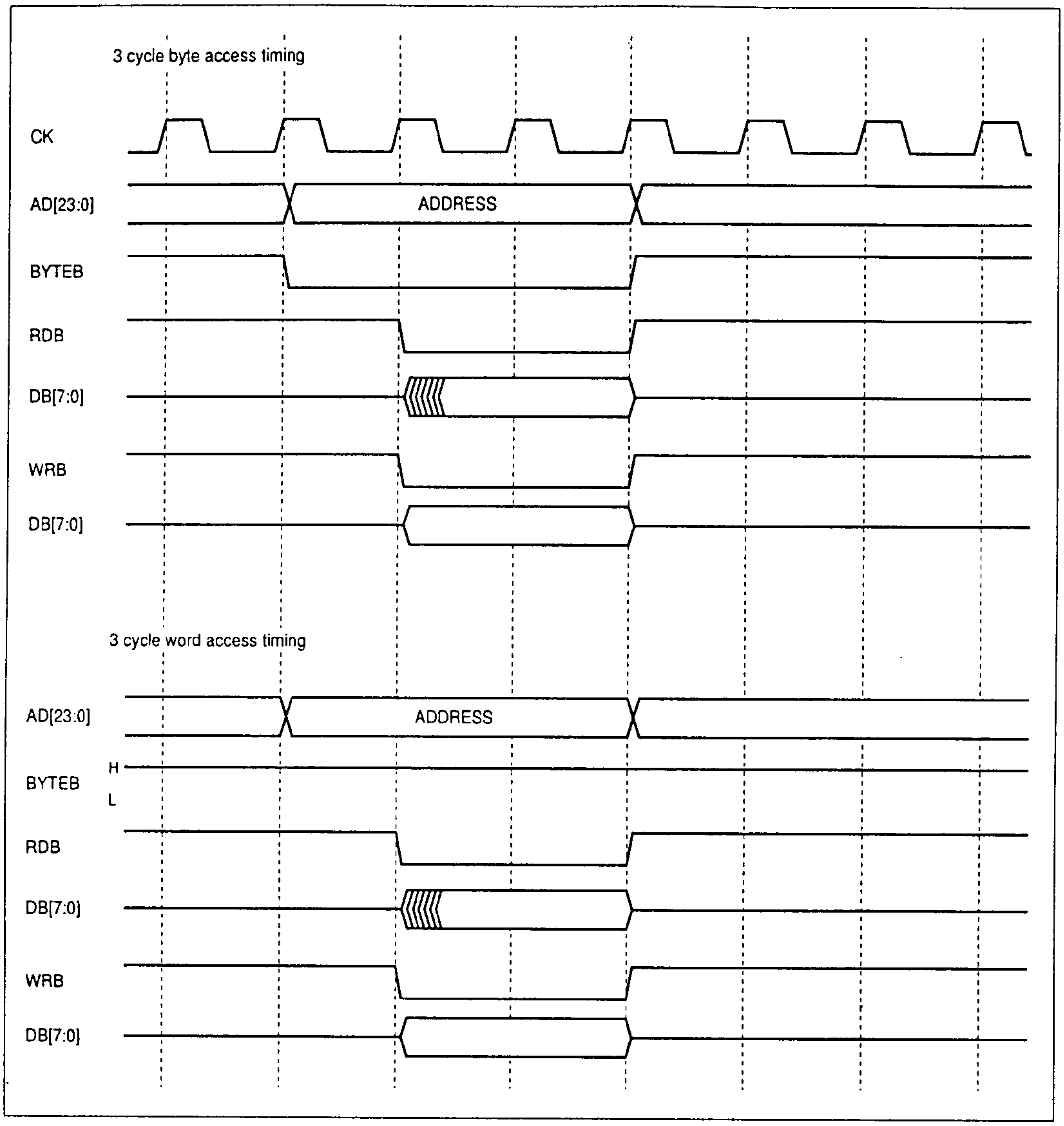


Fig. 3-24 Address/data bus nonmultiplex, 3 cycle 16-bit bus size

Chapter 4 I/O port

The SM6010 has 5 8-bit I/O ports. These ports, P0-P4, also serve as functional port. This chapter describes the registers related to P0-P4 ports.

■ Direction of each port line can be individually set

The direction of P0-P4 (except P2)ports can be individually set from direction registers P0D-P4D. A port when used as a functional pin, its direction must be set.

■ Ports serve as function pins

Ports serve as functional pin according to the setting of the related mode register.

> Expandable pin (for accessing external memory), SCI (UART/SIO), PWM output, external interrupt input, analog input.

■ Port structure, input

The input is protected with a diode and without pull resistor. Some rectangular wave input pins have schmitt trigger circuit which also used for nomal input.

■ Port structure, output

Default setting is CMOS output but SCI output pin can be changed to N-ch open drain by the user program. The protection diode remains connected.

For port structure and list of functional pins, and terminaltion of unused pins, refer to **1.3 Pin description.**

In this manual, ports not used as functional pins are referred to as:

- Standard I/O port
- Standard I/O pin (or port)
- Standard input (output) pin (or port)

4.1 Port register

(2) P0-P4: port 0-4 data registers

Register	Symbol	Bit length	Address	SFR
Port 0 data register	P0	Byet	00FF00H	80H
Port 1 data register	P1	Byet	00FF04H	82H
Port 3 data register	P3	Byet	00FF14H	8AH
Port 4 data register	P4	Byet	00FF0EH	87H

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0
Bit symbol								

Data registrs P0-P4 are used to read data from ports P0-P4 and set output data into ports P0-P4. Read/write operation of the data register depends on direction of ports.

- When pin is set to input

  Read operation reads the level on the pin into memory.

  Write operation updates the bits of the register without affecting the function of the input pin.

- When pin is set to output

  Read operation transfers the content of the data register into memory.

  Write operation updates the bits of the data register which represents value to be output.

The pins of each port are set by direction registers (P0D-P4D).

(3) P2: port 2 data register

Register	Symbol	Bit length	Address	SFR
Port 2 data register	P2	Byte	00FF1CH	8EH

Bit	7	6	5	4	3	2	1	0
R/W	R	R	R	R	R	R	R	R
Initial value	-	-	-	-	-	-	-	-
Bit symbol								

The data register P2 represents the input from port P2. When read, the content of the register is transferred into memory.

(4) POD-P4D: PO-P4 direction registers

Register	Symbol	Bit length	Address	SFR
Port 0 direction register	POD	Byte	00FF02H	81H
Port 1 direction register	P1D	Byte	00FF06H	83H
Port 3 direction register	P3D	Byte	00FF16H	8BH
Port 4 direction register	P4D	Byte	00FF10H	88H

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0
Bit symbol								

Each of these 4 direction register sets the direction of the port. Bits 0-7 of the direction register correspond to bits 0-7 of the port.

- When the bit of a direction register is 0, the port is set to input.
- When the bit of a direction register is 1, the port is set to output.

To set ports P0-P4(except P2) to functional pins, also set the direction of the pins: input pin to input and output pin to output.

As a rule, the direction of the pins serving as input should be set to input and that of output pin to output. For direction settings, refer to individual port description.

P0M: port 0 mode register

Register	Symbol	Bit Length	Address	SFR
Port 0 mode register	P0M	Byte	00FF08H	84H

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Bit Symbol	P0M7	P0M6	P0M5	P0M4	P0M3	P0M2	P0M1	P0M0

This mode register sets the configuration of P0 port. P0D need to be programmed correctly for each mode.

Bit7-0: P0[7:0] pin mode

P0M[7:0]	P0[7:0] pins mode
0	Standard I/O port0[7:0]
1	Address port AD[23:16]

P1M: port 1 mode register

Register	Symbol	Bit Length	Address	SFR
Port 1 mode register	P1M	Byte	00FF18H	8CH

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	NA	0	0	0	0	0
Bit Symbol	P1M7	P1M6	P1M5		SCK0		NO0	SO0

This mode register sets the configuration of P1 port. P1D need to be programmed correctly for each mode.

Bit 7: P17/AD[24] pin output format select(P1M7)

P1M7	P17 pin mode
0	Standard I/O port
1	Address port AD24

Bit 6: P16/LCDCTL pin output format select(P1M6)

P1M6	P16 pin mode
0	Standard I/O port
1	LCDCTL

Bit 5: P15/PWMOUT pin output format select(P1M5)

P1M5	P15 pin mode
0	Standard I/O port
1	PWMOUT

Bit 3-2: P12/SCK0 pin mode set(SCK0)

SCK0		P12/SCK0 pin mode
0	0	Standard I/O port12 (input:schmit)
0	1	SCI0 synchronous clock input(scmit)
1	0	SCI0 clock output(UART: bit rate x 16; SIO:present bit rate)
1	1	SCI0 clock(bit rate)

Bit 1: P1o/TXD0 pin output format select(NO0)

NO0	P1o/TXD0 pin output mode
0	CMOS output
1	N-ch open drain output

Bit 0: P1o/TXD0 pin mode set(SO0)

SO0	P1o/TXD0 pin mode
0	Standard output port1o
1	TXD0

P4M0: port 4 mode register 0

Register	Symbol	Bit Length	Address	SFR
Port 4 mode register 0	P4M0	Byte	00FF12H	89H

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Bit Symbol	EXINT3		EXINT2		EXINT1		EXINT0	

Port4 can serve as external interrupt inputs(EXINT7 - EXINT0)

Bit7-Bit0: P4[3:0] pins interrupt set(EXINT3 - EXINT0)

EXINT		P4$_{3\text{-}0}$ external interrupt trigger
0	0	Standard Port
0	1	Interrupt request occurs on rising edge (Interrupt request occurs on high level in stop mode)
1	0	Interrupt request occurs on falling edge (Interrupt request occurs on high level in stop mode)
1	1	Interrupt request occurs on rising and falling edges

P4M1: port 4 mode register 1

Register	Symbol	Bit Length	Address	SFR
Port 4 mode register 1	P4M1	Byte	00FF1AH	8DH

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Bit Symbol	EXINT7		EXINT6		EXINT5		EXINT4	

Port4 can serve as external interrupt inputs(EXINT7 - EXINT0)

Bit7-Bit0: P4[7:4] pins interrupt set(EXINT7 - EXINT3)

EXINT		P47-4 external interrupt trigger
0	0	Standard Port
0	1	Interrupt request occurs on rising edge (Interrupt request occurs on high level in stop mode)
1	0	Interrupt request occurs on falling edge (Interrupt request occurs on high level in stop mode)
1	1	Interrupt request occurs on rising and falling edges

Chapter 5 WDT

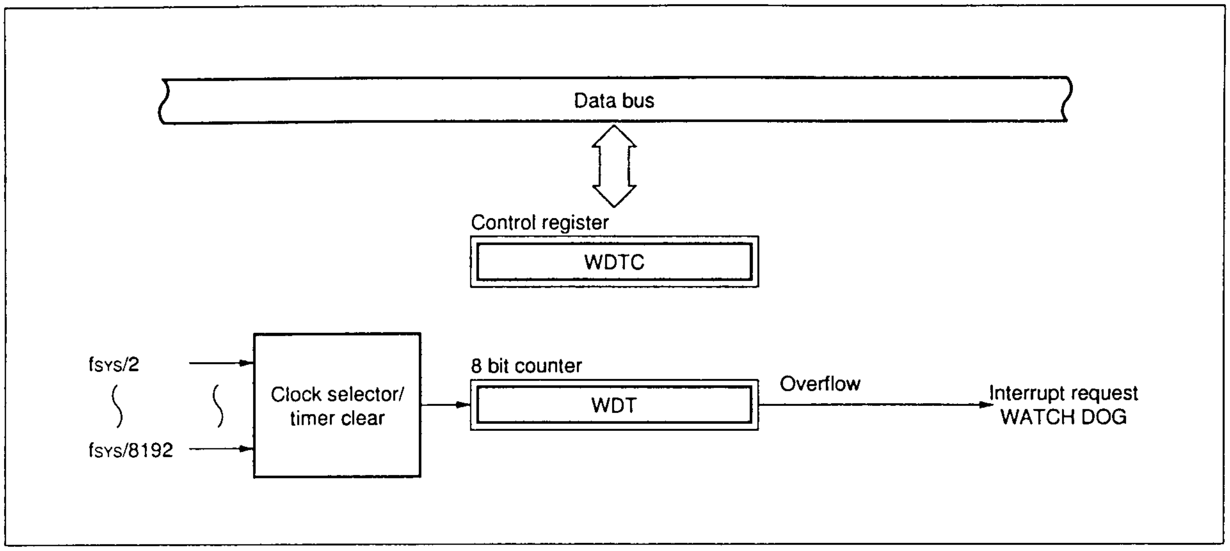


Fig. 5-1 Block diagram of WDT

Table 5-1 Timer functions

Timer No.	Up counter		Capture/compare register	Count clock		Clock output		Capture input	Interrupt event	Others
	Free running※1	Comparing※2		Internal	External	PWM	Compare			
WDT (watch dog timer, 8 bit)	○	–	–	$f_{SYS}/2$, $f_{SYS}/32$ $f_{SYS}/128$, $f_{SYS}/256$ $f_{SYS}/512$, $f_{SYS}/2048$ $f_{SYS}/4096$, $f_{SYS}/8192$	–	–	–	–	· Counter overflow × 1 (nonmaskable interrupt occurs.)	· Counter stop function · Counter clear bit

※1 : Counter clears itself upon overflow. ※2 : Counter is cleared upon compare match occurs. ※3 : When contents of counter and compare register match. ※4 : Count on rising edge.

5.1 WDT register

(1) WDTC: WDT control register

This register sets the watch dog timer.

Register	Symbol	Bit length	Address	SFR
WDT control register	WDTC	Byte	00FF62H	B1H

Bit	7	6	5	4	3	2	1	0
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	-	-	-	0	0	0	0	0
Bit symbol	-	-	-	CCL	CEN	SEL		

■ Bit 4: count clear (CCL)

Writing 1 into this bit place clears the counter to 0000H. This bit is reset to 0 as the counter is cleared.

■ Bit 3: count operation control (CEN)

Bit CEN	Operation
0	Stop watch dog timer count
1	Start watch dog timer count

This bit starts and stops the counting operation of the watch dog timer. The watch dog timer can be started by a program.

Writing 0F7H into address 00FF63H (not the address of this register), direct addressing DA mode or a byte transfer instruction stops the counting. Only these three operation can stop the watch dog timer. For further information, refer to **5.2 (12) operation of watch dog timer** and **5.4 Timer operating considerations.**

■ Bits 2-0: WDT count clock select (SEL)

Bit SEL			Operation	
			Count clock	Overflow time (fsys = 10 MHz))
0	0	0	fSYS/2	51.2 µs
0	0	1	fSYS/32	819.2 µs
0	1	0	fSYS/128	3.3 ms
0	1	1	fSYS/256	6.6 ms
1	0	0	fSYS/512	13.1 ms
1	0	1	fSYS/2048	52.4ms
1	1	0	fSYS/4096	104.9ms
1	1	1	fSYS/8192	209.8ms

(2) Watch dog timer (WDT counter)

This is the counter clock of the watch dog timer and 8 bit free running configuration. This counter is not mapped on address space and therefore cannot be read and written directly.

Writing 1 into bit place CCL (bit 4: WDTC) clears the counter.

Chapter 6 A/D converter

6

SM6010 has a 10-bit A/D converter which has 8 multiplexer analog inputs.

The A/D converter operates in either A/D conversion mode or level comparison mode.

The A/D converter starts when a value is set in the control register or when triggered by a timer output (auto start). By using DTS, continuously repeated conversion of multiple channels is made possible.

Pins P2o (AINo)-P27 (AIN7) are used as analog input to the A/D converter.

Operation clock (A/D clock) and sampling clock are variable so that the A/D converter can operate under various conditions.

The figure below shows the blocks of the A/D converter.

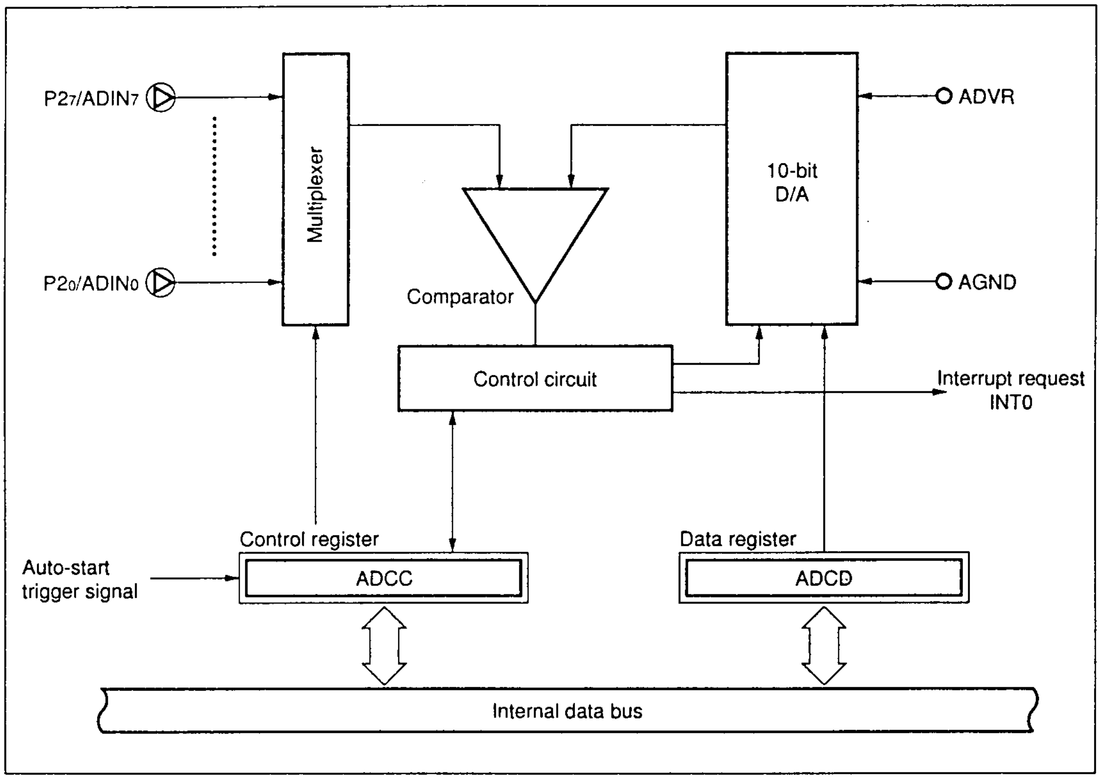


Fig.7-1 Block diagram of A/D converter

Below shows examples of conversion (comparison) times:

- A/D conversion mode: approx. 16-23 µs (main clock at 20 MHz)
- Level comparison mode: approx. 7-14 µs (main clock at 20 MHz)

6.1 Description of registers

(1) ADCD: A/D data register

The data register ADCD stores 10-bit A/D data whose field can be read and written. The bits other than data bits are always set at 0.

Register	Symbol	Bit length	Address	SFR
A/D data register	ADCD	Word	00FF80H	C0H

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit symbol	0	0	0	0	0	0										

In the A/D conversion mode, the register stores digital value (A/D data) equivalent of the analog input voltage. In the level comparison mode, the register stores the reference digital value which is tó be compared with the analog input voltage. The reference value must be stored after entering the comparision mode because ADCD cannot be written in the A/D conversion mode.

(2) ADCC: A/D control register

This register controls operations of the A/D converter. These contols include A/D power supply control, operation mode selection, selection of input pins, storage of comparison results in the level comparison mode and start/stop of A/D conversion. The register also controls A/D converter interrupt and sets auto start and operating conditions.

Register	Symbol	Bit length	Address	SFR
A/D control register	ADCC	Word	00FF82H	C1H

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	-	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit symbol	-	SMP		-	-	CKS	BE	IE	ADP	CMF	ADS	ADM	SELIN			

■ Bits 14-13: select sampling clock (SMP)

Bit SMP		Operation
0	0	A/D clock × 1
0	1	A/D clock × 2
1	0	A/D clock × 4
1	1	A/D clock × 8

The register sets the sampling time of the A/D converter. The setting value should be selected according to the operating conditions of the microcomputer. (Refer to **6.2 Operation in A/D conversion mode.**) One cycle of operation clock selected by bit CKS ($f_{SYS}/10$ or $f_{SYS}/20$) is called 1 A/D clock.

- Bit 12-11: Always set at "0" with SM6010

- Bit 10: A/D clock select (CKS)

Bit CKS	A/D converter operation clock (A/D clock)
0	fsys/20 (2 µs: 20 MHz main clock)
1	fsys/10 (1 µs: 20 MHz main clock)

This bit determines the speed of the A/D converter: standard set is 1.

- Bit 9: boost circuit enable (BE)

Bit BE	Operation
0	Set to $V_{DD} > 3$ V operation
1	Set to $V_{DD} \leq 3$ V operation

- Bit 8: interrupt enable (IE)

Bit IE	Operation
0	After completion of A/D operation, disable interrupt request
1	After completion of A/D operation, enable interrupt request

To issue the interrupt at the end of A/D converter operation, set this bit to 1. To verify the end of A/D converter operation by monitoring bit ADS (bit 5: ADCC) instead of using the interrupt, set this bit to 0.

- Bit 7: control A/D power supply (ADP)

Bit ADP	Operation
0	A/D supply off state (no A/D operation)
1	A/D supply on state (A/D operation enable)

Turning on of the A/D power supply feeds the voltage on ADVR pin to the A/D converter block. Bit ADP is set or cleared by the operations of the microcomputer as well as the by set/clear operation with the program.

- The bit is automatically set when the auto start successfully starts.
- The bit is automatically cleared at the end of A/D conversion.

- Bit 6: storage of result of level comparison (CMP)

Bit CMP	Result
0	Input voltage < reference voltage (ADCD register settings)
1	Input voltage > reference voltage (ADCD register settings)

The value stored in bit CMP is unknown when the input voltage is equal to the reference voltage.

■ Bit 5: start/clear (ADS)

Bit ADS	Operation
0	Stop A/D converter. Remains 0 while the converter is idling.
1	Start A/D conversion and clear at the end of conversion

Bit ADS is set or cleared by the operations of the microcomputer as well as by set/clear operation with the program.

- The bit is automatically set when the auto start successfully starts.
- The bit is automatically cleared at the end of A/D conversion.

To know the end of operation of the A/D converter without using the interrupt, monitor this bit.

■ Bit 4: operation mode select (ADM)

Bit ADM	Operation
0	A/D conversion mode
1	Level comparison mode

To store the A/D data into the data register ADCD for comparison, set ADM for selecting the level comparison mode.

■ Bits 3-0: analog input pin select (SELIN)

Bit SELIN				Operation
0	0	0	0	Set P2₀/AIN0 pin to analog input
0	0	0	1	Set $P2_1/AIN_1$ pin to analog input
0	0	1	0	Set $P2_2/AIN_2$ pin to analog input
0	0	1	1	Set $P2_3/AIN_3$ pin to analog input
0	1	0	0	Set $P2_4/AIN_4$ pin to analog input
0	1	0	1	Set $P2_5/AIN_5$ pin to analog input
0	1	1	0	Set $P2_6/AIN_6$ pin to analog input
0	1	1	1	Set $P2_7/AIN_7$ pin to analog input
Other bit settings				Disable setting

Set these bits between 0000 and 0111. No other values are invalid.

6.2 Operations in A/D conversion mode

The A/D conversion mode converts the analog voltage from one of analog input pins into the digital equivalent value.

The A/D converter operates in the successive comparison mode. That is, it compares the analog input value with the binary weighted reference, voltage charged in the capacitor array (charge redistribution type).

At the end of the A/D conversion, the A/D converter stores the resultant 10 bit digital data into the data register ADCD.

Below details the operation in the A/D conversion mode. For manipulation of the A/D converter, refer to **6.4 Operating procedure in the A/D conversion mode.**

● Conversion time

- Conversion time = A/D operating clock × 16-23

  (16-23μs at 10 MHz)

The time depends on the following parameter settings:

- Conversion time = A + B + C

  A: sampling time (A/D clock × 1-8)

  B: successive comparison time (A/D clock × 13)

  C: delay time (A/D clock × 2 or so)

The A/D operating clock is the reference clock to which the A/D converter ticks. This clock is $f_{SYS}/10$ (divided-by-10 system clock) or $f_{SYS}/20$ (divided-by-20 system clock) depending on the setting of bit CKS (bit 10: ADCC).

The sampling time (A) can be selected among (1, 2, 4 or 8) times the A/D clock, by the setting of bit SMP (bits 14-13: ADCC).

Successive comparison time (B) is almost equal to 13 A/D clocks in A/D conversion mode.

Delay time (C) includes variations at conversion start timing and time delay to the interrupt request generation. This duraion amounts to approx. 2 A/D operating clocks.

● Relationship between input voltages and conversion results

The result value of the A/D conversion (A/D data) is stored in the lower 10 bit places of data register ADCD. Once the A/D value is obtained, the value of the input analog voltage can be calculated using the following equation. In the equation, the A/D data must be expressed in decimal and the range of A/D data is 0 - 1,023 in decimal.

- Input analog voltage = A/D data (10 bit)÷1,024 × ADVR (V)

● Supplying reference voltage

Supply the A/D converter reference voltage across pins ADVR and AGND as follows:

- ADVR: the value between the allowable minimum V_{DD} and the V_{DD} level used on the system
- AGND: 0 V, that is, the same potential on pin GND

● Setting analog input pins

The 8 pins $P2_0/AIN_0$-$P2_7/AIN_7$ are designed for use as analog input pin. P2 port is originally designed as input only without connecting to any pull up resistor, it can inherently serve as A/D input and requires no specific setting.

Since the A/D converter accommodates only one analog channel at a time, select one analog input pin among 8 pins ($P2_0/AIN_0$-$P2_7/AIN_7$) by setting bit SELIN (bits 4-0: ADCC).

● Controlling interrupt

By setting bit IE (bit 9: ADCC), the interrupt request at the end of A/D conversion can be enabled or disabled.

● Selecting A/D clock

The clock that ticks the A/D converter (A/D clock) are adjustable according to the selected main clock to keep the A/D converter operating under the best condition.

Select $f_{SYS}/10$ or $f_{SYS}/20$ (divided-by-10 or 20 system clock) by setting bit CKS (bit 10: ADCC).

With the SM6010 series select $f_{SYS}/10$. The A/D converter operates at lower speed when $f_{SYS}/20$ selected but has the same conversion accuracy when compared with the higher speed operation on $f_{SYS}/10$.

● Using boost circuit

When the power supply voltage is lower than 3 V, turn on the boost circuit to obtain precise A/D conversions. Set bit BE (bit 9: ADCC) to turn on/off the circuit. When using the power supply of 3 V or higher set bit BE to 0 to turn off the circuit. If the booster is enabled when the power supply voltage is 3 v or higher, internal A/D reference voltage becomes higher than the allowable value, the operation of the microcomputer cannot be guaranteed. Bit BE is set to 1 but the boost circuit won't start until the A/D power supply is turned on.

● Setting sampling time

The on resistance of analog input multiplexer becomes larger as the ADVR voltage decreases. The time required to charge the internal capacitor (sampling time) becomes longer as the input impedance of the analog input pin increases. With the A/D converter the sampling time is adjsutable so that it can operate under various conditions. Set bit SMP (bits 14-13: ADCC) for appropriate sampling time which depends on operation condition of the system. Before determining the best possible sampling time, confirm the system operation.

● At the end of A/D conversion

When the A/D converter completes the conversion cycle, bits ADP and ADS are cleared regardless of the operation mode. ADP is the A/D power supply bit and ADS is the A/D start bit of the control register.

When bit IE (bit8: ADCC) has been set 1, the interrupt is generated at the end of A/D conversion.

6.3 Operation in the level comparison mode

In the level comparison mode, the converter compares the analog voltage from one of analog input pins with the internal reference voltage and stores the result in bit CMP (bit 6: ADCC). The reference voltage is set by the digital value stored in the data register ADCD.

The following description details the level comparison operation. For operating procedure, see section **6.5 Operation in the level comparison mode example.**

● Comparison time

The following time is required for the system to complete one comparison cycle:

- Comparison time = A/D clock × 7-14

  (7-14μs at 20 MHz main clock)

The time of the comparison depends on the following parameter settings:

- Comparison time = A + B + C

  A: sampling time (A/D clock × 1-8)

  B: successive comparison time (A/D clock × 4)

  C: delay time (A/D clock x 2 or so)

The A/D clock is the reference clock to which the A/D converter ticks. This clock is $f_{SYS}/10$ (divided-by-10 system clock) or $f_{SYS}/20$ (divided-by-20 system clock) depending on the setting of bit CKS (bit 10: ADCC).

The sampling time (A) can be selected among (1, 2, 4 or 8) times the A/D clock, by the setting of bit SMP (bits 14-13: ADCC).

Successive comparison time (B) is almost equal to 4 A/D clocks in level comparison mode.

Delay time (C) includes variations at comparison start timing and time delay to the interrupt request generation. This duraion amounts to approx. 2 A/D clocks.

● Data of internal reference voltage for comparison

Before starting the level comparison, store the reference value (A/D data) into the lower 10 bit places of the data register ADCD. The relationship between the A/D data and the resulting internal reference voltage is expressed by the equation shown below.

A/D data must be expressed in decimal and the range of A/D data is 0 - 1,023 in decimal.

- Internal reference voltage = A/D data (10 bit)÷1,024 × ADVR (V)

● Result of comparison

The result of level comparison is stored in the bit place CMP (bit 6: ADCC). The bit is set as follows:

- Bit CMP = 0

  Input voltage (level on the analog pin) < comparison voltage (A/D data converted into the analog voltage)

- Bit CMP = 1

  Input voltage (level on the analog pin) > comparison voltage (A/D data converted to the analog voltage)

Bit CMP is unknown when the input voltage is equal to the comparison voltage.

Since the following are identical as in the case of A/D conversion mode, refer to the description in section **6.2 Operations in A/D conversion mode.**

- ● Supplying reference voltage
- ● Setting analog input pin
- ● Controlling interrupt
- ● Operation of DTS
- ● Selecting operating clock
- ● Using boost circuit
- ● Setting sampling time
- ● Operation at the end of A/D conversion cycle

6.4 Operating procedure in the A/D conversion mode

This section describes the procedure to be taken in the A/D conversion mode.

The value set in the control register is expressed as follows:

1: set to 1

0: set to 0

*: set to a value depending on the operating mode

-: bit irrelevant to control or can be left unset

The following description assumes that the A/D converter is to be operated under the following conditions.

- Operation mode: A/D conversion
- ADVR level: 2.5 V (use boost circuit)
- Pin used: AIN$_5$ (P2$_5$)
- Auto start: not used
- Interrupt: interrupt request enabled
- Main clock: 20 MHz
- A/D clock: f$_{SYS}$/10
- Sampling time: 8 A/D clocks

(1) Interrupt related settings

Perform interrupt related settings. When no interrupt function has not been used in the previous A/D conversion, the interrupt request bit may be left at 1. Clear the bit to 0.

(2) Settings and start of A/D conversion

```
ADCC  ←  -11- 0111 1-10 0101B
```

All conditions are set and A/D conversion starts.

(3) End of A/D conversion

An interrupt request is generated at the end of A/D conversion cycle.

(4) Storage of A/D result

The 10 bit data obtained in the A/D conversion is stored in the bit places 9 to 0 of the data register ADCD.

The result of the A/D conversion is stored into the data register ADCD upon completion of the A/D conversion.

This data is retained until the next conversion ends.

6.5 Operation in the level comparison mode example

The procedure in the level comparison mode is as described below.

The value in the control register is expressed as follows:

1: set to 1

0: set to 0

*: set to a value depending on the operating mode

-: bit irrelevant to control or can be left unset

The following description assumes that the A/D converter is to be operated under the following conditions.

- Operation mode: level comparison
- ADVR level: 5 V (boost circuit not used)
- Comparison value: 11 1101 1101B (10 bit data)
- Pin used: AIN_0 ($P2_0$)
- Auto start: not used
- Interrupt: interrupt request enabled
- Main clock: 30 MHz
- A/D clock: $f_{SYS}/10$
- Sampling time: 2 A/D clocks

(1) Interrupt related settings

Perform required settings related to interrupt. When no interrupt function has not been used in the previous A/D conversion, the interrupt request bit of the A/D converter may be left at 1. Clear the bit to 0.

(2) Setting operation mode

ADCC ← -01- 0011 0-01 0000B

The A/D converter is set to the level comparison mode. This is necessary to store the A/D data which is used to generate the reference voltage for comparison. In the above setting, bits ADP (bit 7: ADCC) and ADS (bit 5: ADCC) are set to 0. This sets the operating conditions of the A/D converter.

(3) Storage of A/D data for comparison

ADCD ← 0000 0011 1101 1101B

Lower 10 bits of the A/D data for reference voltage are stored in the data register ADCD.

(4) Settings and start of level comparison

ADCC ← -01- 0001 1-11 0000B

Afeter setting, level comparison process starts.

(5) End of level comparison

An iterrupt request is generated at the end of the comparison cycle.

(6) Storage of level comparison resultant data

The result is sotred into the bit place CMP (bit 6: ADCC).

6.7 A/D converter considerations

- Apply power supply to the analog block through ADVR pin.

ADVR pin feeds the A/D converter block in addition to serve as A/D reference voltage. The power capacity to this pin must meet these power demands.

- Operating conditions

A/D clock, sampling time and A/D converter operating conditions must be set to meet the accuracy of the A/D requried by the system. Take into account other sytem operation conditions.

- Voltage on the ADVR pin must be equal to or below V_{DD}

Application of a voltage higher than V_{DD} level to ADVR pin affects guaranteed microcomputer operation.

- Considerations of boost circuit

When the voltage on ADVR pin is 3 V or higher, keep the boost circuit off.Otherwise, the microcomputer judges that the level on ADVR pin exceeds V_{DD} and will not operate as designed.

- Turn off A/D power in lower power consumption operation

In a standby mode, turn off the A/D power since it supples current to idling circuits. To conserve power, turn on the A/D power only when the A/D convert should work.

Bit ADP (bit 8: ADCC) is set to off (0 level) upon completion of A/D cycle. Always keep this bit at 0 if the A/D converter is not used. Otherwise, the A/D power source is left on.

- Operating considerations of A/D converter

Never write into the A/D converter control register unless the purpose is to stop the A/D converter. Writing into the register affects A/D operation.

Chapter 7 SCI (UART/SIO selectable serial interface)

7

This chapter describes UART/SIO selectable interface.

When distiguishment between UART and SIO is necessary, these terms are indenpendently used in the text.

The SCI is divided into two blocks: transmitter and receiver.

Outline of SCI is as follows:

■ Transfer can be made through either UART or SIO: program selectable.

■ UART

- Full duplex communications are possible because UART is divided into transmitter and receiver sections.
- The receiver has double buffers which enable successive receiving operation.
- Data length: 7, 8 or 9 bits (no parity in 9 bit format)
- Stop bit: 1 or 2
- Parity: Even/odd/no parity
- Interrupt: End of transmission, end of receive/receive error (common)
- Error detection: Parity, overrun, frame

■ SIO

- Data length: 8 bits
- Error detection: Overrun

■ Bit rate setting of synchronous clock

The internal synchronous clock generator can set the bit rate in a range of 150 and 38,400 bps. The synchronous clock can be fed to external devices. Contrarily, an external clock can be used in place of the internal clock.

■ Pins used

- SCI0: $P1_0$/TxD0, $P1_1$/RxD0, $P1_2$/SCK0

Fig. 9-1 is a block diagram of SCI.

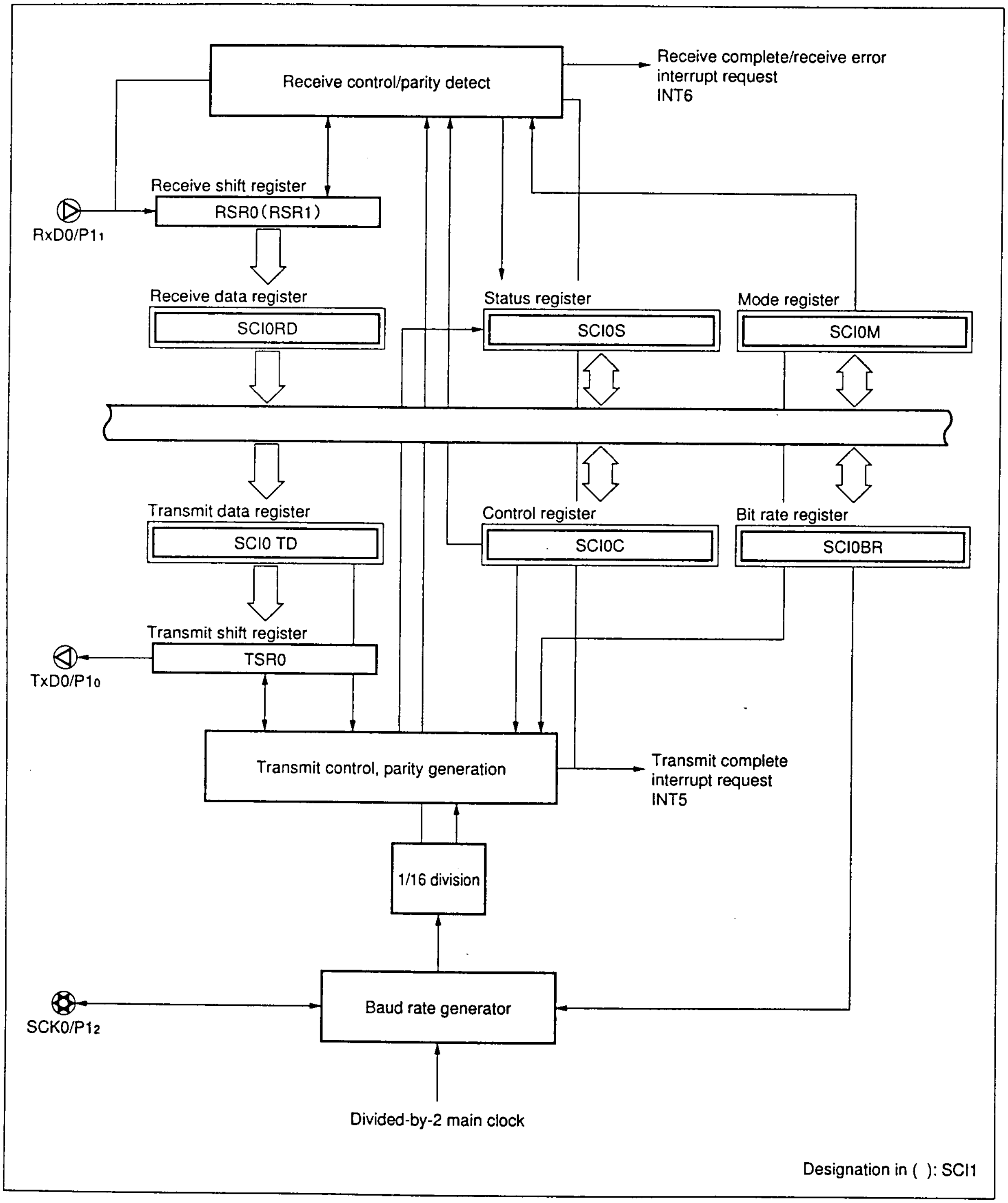


Fig. 9-1 Block diagram of SCI

7.1 SCI registers

(1) SCI0TD: SCI0 transmit data register

The register which stores the data of the SCI to be transmitted.

Register	Symbol	Bit length	Address	SFR
SCI0 transmit data register	SCI0TD	Lower byte	00FF88H	C4H

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1
Bit symbol								

Writing into this register while transmission enable starts the transmitter.

The transmitter section of SCI contains two buffer types, data register SCI0TD and transmit shift register TSR0 . During transmission the data written in the data register SCI0TD is transferred to the transmit shift register TSR0 from which the data is sent out.

The timing of writing into the data register SCI0TD can be monitored by checking the status of the bit in the status register SCI0S with the user program or by enabling the interrupt request at the end of transmission.

Since the shift register TSR0 is not mapped on the address space, the register is not accessible.

(2) SCI0RD : SCI0 receive data register

The register which stores the received data of the SCI.

Register	Symbol	Bit length	Address	SFR
SCI0 receive data register	SCI0RD	Lower byte	00FF8AH	C5H

Bit	7	6	5	4	3	2	1	0
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0
Bit symbol								

The receiver section of SCI contains two buffer types, data register SCI0RD and shift register RSR0. During receive operation, received data is first stored in the shift register RSR0 and then transferred to the data register SCI0RD automatically at the end of the successful receiving cycle. This is not the case when receive overrun error occurs.

Certain data received in the 9-bit transfer mode may not be transferred. Refer to **7.1 (6) SCI0C: SCI0 control register**, bit WU.

The timing of data receive and receive error can be monitored by checking the status of the bit in the status register SCI0S with the user program or by enabling the interrupt request at the end of receive (or receive error).

Since the shift register RSR0 is not mapped on the address space, the register is not accessible.

(3) SCI0BR : SCI0 bit rate register

This register sets the bit rate of the SCI.

Register	Symbol	Bit length	Address	SFR
SCI0 bit rate register	SCI0BR	Word	00FF8CH	C6H

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bit symbol	0	0														

Below shows the relationship between the settings in the register SCI0BR and the resulting bit rate.

- UART mode

$$\text{Bit rate (bit/s)} = \frac{X_{TAL}}{64 \times (N+1)} \times 10^{6}$$

X_{TAL}: main clock frequency (MHz)
N: set value ($0 \leqq N \leqq 16{,}383$)

- SIO mode

$$\text{Bit rate (bit/s)} = \frac{X_{TAL}}{4 \times (N+1)} \times 10^{6}$$

X_{TAL}: main clock frequency (MHz)
N: set value ($0 \leqq N \leqq 16{,}383$)

Using an external synchronous clock causes the microcomputer to ignore settings of the register SCI0BR . Instead, the microcomputer uses the clock from SCK0 pin (or divided-by-16 clock) as the synchronous clock.

(4) SCI0S : SCI0 status register

This register represents the SCI transfer status.

Register	Symbol	Bit length	Address	SFR
SCI0 status register	SCI0S	Upper byte	00FF8BH	C5H

Bit	7	6	5	4	3	2	1	0
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	1	1	0	0	0	0
Bit symbol	RB8	RXBSY	TXEMP	TXRDY	OR	FE	PE	RDRF

■ Bit 7: received bit 8

This bit stores the 9th bit of the data to be received through the UART in 9 bit receive mode.

This bit place is always 0 in other modes or the receiving is made through SIO.

The bit place used to store 9th bit (bit TB8) of the data to be transmitted is located in the mode register SCI0M.

■ Bit 6: receiver busy (RxBSY)

When the signal from RxD0 pin reaches the receiver, it judges that the data is being received and sets this bit to 1. When RxD0 is no longer supplying input data, this bit is reset to 0.

■ Bit 5: transmitter empty (TxEMP)

This bit is turned to 1 when the data register SCI0TD and the shift data register TSR0 have no data to be transmitted and as the stop bit of the last data is transmitted.

This bit is initialized to 1 after returning from the hardware reset status.

■ Bit 4: transmitter ready (TxRDY)

This bit is turned to 1 when the data to be transmitted has been transferred from the data register SCI0TD to the shift register TSR0 and the former is ready to accept the next data.

This bit is initialized to 1 after returning from the hardware reset status.

■ Bit 3: overrun error (OR)

This bit is turned to 1 when overrun error occurs. This means that the next data has been received while the program cannot read out the received data stored in the data register SCI0RD and store the data into memory. Reading the register SCI0S clears the bit OR to 0.

■ Bit 2: frame error (FE)

This bit is turned to 1 when frame error occurs. The frame error occurs if the duration of the stop bit at the end of receiving is shorter than the preset value. If both frame error and overrun error occur at the same time, this bit is not set to 1.

Reading the register SCI0S clears the bit FE to 0.

In SIO mode, the content of this bit is invalid.

■ Bit 1: parity error (PE)

This bit is turned to 1 when parity error occurs. This means that the parity bit is wrong when parity check is enabled. If both parity error and overrun error occur at the same time, this bit is not set to 1.

Reading the register SCI0S clears the bit PE to 0.

In SIO mode, the content of this bit is invalid.

■ Bit 0: receive data register full (RDRF)

This bit is turned to 1 when the data received and stored in the shift data register RSR0 is transferred to the data register SCI0RD automatically.

Reading the register SCI0RD clears the bit RDRF.

CAUTION

When checking the bits (bit testing) in the status register SCI0S , load the contents of the register to memory and then check the loaded contents. Bits OR, FE, PE and RDRF are check and clear, that is, reading these bits clear the bits. Remember that, reading one of these bits also clears other check and clear bits: these bits could not be tested.

(5) SCI0M : SCI0 mode register

This bit sets the transfer format of SCI, and also selects between UART and SIO.

Register	Symbol	Bit length	Address	SFR
SCI0 mode register	SCI0M	Upper byte	00FF89H	C4H

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0
Bit symbol	SCM	TB8	MSB	EOP	FL		PEN	SBL

■ Bit 7: SCI mode set (SCM)

Sets the operation mode (UART/SIO) of the SCI.

Bit SCM	Mode
0	UART
1	SIO

■ Bit 6: tramsmit bit 8 (TB8)

This bit stores the 9th bit of the data to be transmitted through the UART in 9 bit transfer mode. This bit is made invalid in other modes or the transfer is made through SIO and have no effect on transmission. The content in this bit place will not change even if transmit is disabled.

The bit place used to store the 9th bit (bit RB8) of the data received is located in the status register SCI0S .

■ Bit 5: transfer direction change (MSB)

Sets the flow direction of transmit/receive data. This bit is valid in both UART and SIO mode.

Bit MSB	Operation
0	Transmit/receive shift registers TSR0 and RSR0 in the order of LSB to MSB.
1	Transmit/receive shift registers TSR0 and RSR0 in the order of MSB to LSB.

■ Bit 4: even/odd parity (EOP)

Selects a parity option when parity is enabled. This bit is invalid when transfer is made through SIO or in other modes than 9 bit mode through UART.

Parity is enabled/disabled by the setting of bit PEN [(bit 1: SCI0M)].

Bit EOP	Even/odd parity select
0	Select even parity
1	Select odd parity

■ Bits 3-2: frame length (FL)

Selects the data length of transmit/receive data in UART mode. This setting is ignored in SIO mode.

Bit FL		Transfer data length
0	0	8 bits
0	1	7 bits
1	0	9 bits
1	1	9 bits

■ Bit 1: parity enable (PEN)

Enables or disables parity in UART mode. When this bit is set for parity enable, the parity bit is added during transmission and parity check is performed during receive operation.

This bit is invalid when transfer is made through SIO or in other modes than 9 bit transfer mode through UART.

Bit PEN	Operation mode in transmit/receive	
0	Transmit	Do not add parity bit
	Receive	Do not check parity bit
1	Transmit	Add parity bit
	Receive	Check parity bit

■ Bit 0: stop bit length (SBL)

Select the stop bit length in UART mode. This bit is ignored in SIO mode.

Bit SBL	Stop bit length
0	1 bit
1	2 bits

(6) SCI0C : SCI0 control register

This register controls the transmit/receive of SCI and sets interrupt.

Register	Symbol	Bit length	Address	SFR
SCI0 control register	SCI0C	Byte	00FF8EH	C7H

Bit	7	6	5	4	3	2	1	0
R/W	R/W	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	0	-	-	0	0	0	0	0
Bit symbol	WU	-	-	REIE	TIE	RIE	TE	RE

■ Bit 7: wake-up (WU)

In UART and 9-bit mode, this bit causes to accept or ignore the received data depending on the content of the 9th bit (bit RB8) of the received data.

Bit WU	Operation
0	Receive data regardless of the content of bit RB8.
1	Receive data only when bit RB8 is 1 and ignore when bit RB8 is 0.

When the received data is ignored, the microcomputer discards the data stored in the shift register RSR0 (RSR1) without transferring it to the data register SCI0RD. Interrupt request will not be generated.

■ Bit 4: receive error interrupt enable (REIE)

Enables or disables interrupt request upon occurrence of a receive error. When set to 1, this bit generates an interrupt request whenever one of error bits PE, FE and OR is set. SCI receive interrupt and receive error interrupt are assigned in the same vector address.

■ Bit 3: transmit interrupt enable (TIE)

Enables or disables interrupt request when bit TxEMP = 1 (transmit ready). When set at 1, this bit enables the interrupt request once the transmit ready condition is established.

For details of bit TxRDY, refer to the status register SCI0S.

■ Bit 2: receive interrupt enable (RIE)

Enables or disables interrupt request when bit RDRF = 1 (end of receive). When set at 1, this bit enables the interrupt request at the end of receive.

For details of bit RDRF, refer to the status register SCI0S.

■ Bit 1: transmitter enable (TE)

When set at 1, this bit enables transmission. Writing into the data register SCI0TD with transmitter section initialized starts transmission .

■ Bit 0: receiver enable (RE)

When set at 1, this bit enables receive. Receive process starts as the signal comes from the RxD0 pin with the receiver initialized.

Normally the level on RxD0 pin must be high. When this pin is low for one half the period of synchronous clock, the signal on the pin is recognized as the start bit and the microcomputer starts receive procedure.

7.2 Operation of UART

This section describes the operation of SCI in the UART mode: each opeation for each channel. In tis description, transmission procedure and receive procedure are performed separetely, but both procedures can be made at the same time for full duplex communications.

(1) Transmission/receive procedures

● Transfer format

Transfer format depends on SCI mode setting. Below shows the transfer formats.

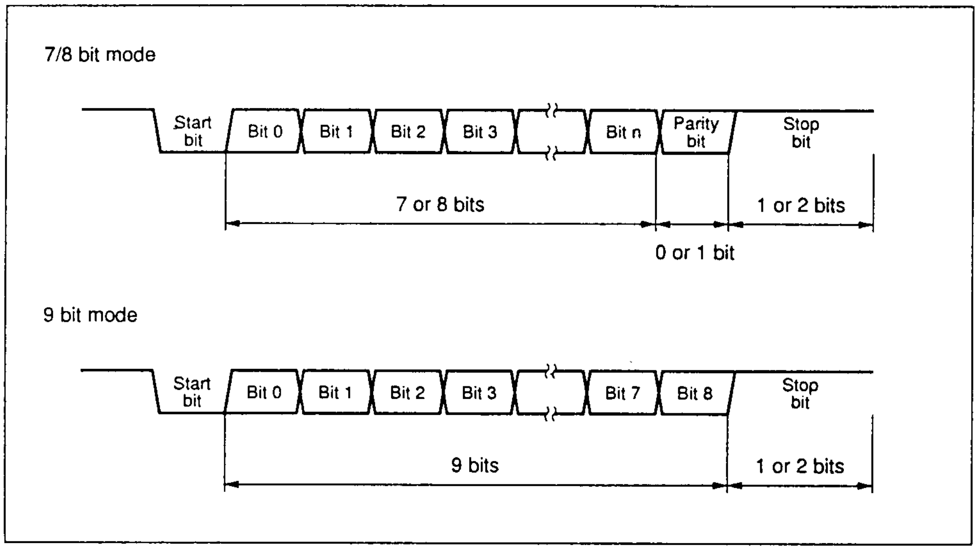


Fig. 9-2 UART transfer format

Parity is not added regardless of parity setting in 9 bit transfer mode.

● Setting bit rate

When using the internal synchronous clock generator, set the bit rate from the bit rate register SCI0BR. Relationship between the register value and the bit rate is as shown below.

$$\text{Bit rate (bit/s)} = \frac{X_{TAL}}{64 \times (N+1)} \times 10^6$$

XTAL: main clock frequency (MHz)
N: set value (0≦N≦16,383)

Synchronous clock generated internally can be sent to the outside world through SCK0 pin.

In the UART mode, the speed of the output clock is at the bit rate or 16 times the bit rate, whichever selected.

When using the external synchronous clock, set its frequency to the value 16 times the desired bit rate. The microcomputer divides the frequency by 16. Set related pins as described below.

● Setting SCI pin

The following pins are used in UART mode.

- SCI0: $P1_0$/TxD0, $P1_1$/RxD0, $P1_2$/SCK0

Note that SCK0 pin is not used for UART transfer. It can be used to transfer the SM6010 internal synchronous clock to an external device, and vice versa.

Flow direction through each pin is set by the direction register P1D.

- TxD0 : output
- RxD0 : input
- SCK0 : according to synchoronous clock input/output

Set the mode of the pins by referring the mode register P1M.

- TxD0 : bit SO0 (bit 0: P1M) ← 1

The above setting sets TxD0 for transmit pin.

- RxD0 : no setting

RxD0 serves as SCI pin when set to input. The input circuit is configured as schmitt trigger type in all modes.

- SCK0 : bit SCK0 (bits 3-2: P1M) ←**
  bit NO0 (bit 1: P1M)←*

Set bits according to the table below.

Bit SCK0		$P1_2$ pin outpout mode select
0	0	Standard I/O pin (Input direction: Schmitt)
0	1	SCI clock input (Schmitt)
1	0	SCI synchronous clock output (UART: bit rate × 16 clocks; SIO: preset bit rate)
1	1	SCI synchronous clock output

Bit NO0	$P1_2$ output format select
0	CMOS output
1	N-ch open drain output

The mode of these 6 pins can be individually set. The pins not used for SCI can be used as normal I/O pins.

● Initializing SCI (a)

Initialization of SCI is to put it in a transmission ready state so that it immediately starts transmission upon writing data into the transmit data register SCI0TD ; or to put it in a receive ready state so that it immediately starts receiving operation upon entering of the start bit from RxD0 pin.

When using SCI first time or when changing transfer format, mode is changed. It must be performed while transmission/receive is disabled.

Set the control register SCI0C as follows.

- SCI0

  Bits TE, RE [bits 1, 0: SCI0C) ← 00

The above procedure stops SCI and initializes shift registers TSR0 and RSR0 to 0FFH.

● Initializing SCI (b)

Check the status of the SCI: using user program, verify that the status register SCI0S is set as follows:

- Register SCI0S = 30H

The 30H shows that the SCI is in stop; no untransmitted data exists; and no data to be stored in memory exists.

Status of related pins are as follows:

- TxD0 : at high level
- RxD0 : schmitt trigger input: no pull up resistor
- SCK0 : high level output (may be N-ch open drain) or schmidt trigger input depending on the synchronous clock I/O sestting

● Initializing SCI (c)

Transfer format should be set. The format is set through setting of the mode register SCI0M . For further information on mode register SCI0M , refer to section **7.1 SCI registers.**

The following setting is necessary when using UART.

- Operation mode of SCI

  To use SCI in UART mode, set bit SCM [(bit 7: SCI0M)] to 0.

- Transfer order

Set the transfer order of the bits in the transmit shift register TSR0 and receive shift register RSR0. To do so, use bit MSB (bit 5: SCI0M).

Bit MSB = 0: LSB of the shift register is first transferred in both transmit and receive mode.

Bit MSB = 1: MSB of the shift register is first transferred in both transmit and receive mode.

- Data length

Data length can be 7, 8 or 9. Selection is through bit setting: bit FL [(bits 3-2: SCI0M),].

Bit FL		Transfer data length
0	0	8 bits
0	1	7 bits
1	0	9 bits
1	1	9 bits

- Parity

Select parity enable or disable and when enable even or odd.

Selection is through bit PEN (bit 1: SCI0M) and bit EOP (bit 4: SCI0M).

Bit PEN	Operation mode in transmit/receive	
0	Transmit	Do not add parity bit
	Receive	Do not check parity bit
1	Transmit	Add parity bit
	Receive	Check parity bit

Bit EOP	Even/odd parity select
0	Select even parity
1	Select odd parity

(2) Transmit operation

The following describes operation of UART in transmit mode assuming that the SCI is initialized according to the procedure described in para. (1) above.

● Setting in 9-bit mode

Before using UART in 9 bit mode, set the 9th bit of the transmit data. This data is to be stored in bit TB8 [(bit 6: SCI0M)]. The stored data is retained until updated by the user program.

● Setting interrupt

Determine whether the interrupt is to be generated at the end of a transmit, through bit TIE [(bit 3: SCI0C)].

Bit TIE	Setting
0	Do not generate interrupt request at the end of transmission.
1	Generate interrupt request INT5 at the end of transmission.

Interrupt request is generated after all the data including the stop bit has been transmitted. Bit TxEMP are set at the same time.

● Setting transmit enable

Set the transmitter to transmit enable state.

- Bit TE [(bit 1: SCI0C)] ←1

These steps initialize the transmitter.

● Starting transmission

Once the above mentioned settings are completed, write the data to be transmitted into the transmit data register SCI0TD . The data is then loaded to the shift data register TSR0 and the transmission starts.The register SCI0TD becomes empty and ready to accept the next transmit data.

While the transmit enable is retained, continuous transmission through UART is possible as long as data is stored in register SCI0TD .

● Transmit status and storage of the next data

To continue transmission, the user program should monitor the interrupt request and the content of the status register to determine when to store the next transmit data into the register SCI0TD. If transmit interrupt request is not used, use the following bit content to determine the storage timing.

- Bit TxRDY, TxEMP = 0,0; the next transmit data cannot be stored.

  →Transmission is taking place and the next transmit data is still stored in register SCI0TD .
- Bit TxRDY, TxEMP = 1, 0: the next transmit data can be stored.

  →Transmission is taking place and no transmit data is stored in the register SCI0TD .
- Bit TxRDY, TxEMP = 1, 1: the next transmit data can be stored.

  →No transmit data are stored in the registers TSR0 and SCI0TD .

● Haulting transmission

When bit TE = 0 (bit 1: SCI0C), UART stops transmitting after it transmits all the data from the shift register. To stop the data stored in the data register SCI0TD upon completion of transmission, the user program should monitor bit TxEMP [(bit 5: SCI0S)] and set bit TE to 0 when TxEMP is reset to 1.

(3) Receive operation

The following describes operation of UART in receive mode assuming that the SCI is initialized according to the procedure described in para. (1) above.

● Setting in 9-bit mode

Before using UART in 9 bit mode, set the wake up. This setting is to be made throgh bit WU [(bit 7: SCI0C)].

Bit WU	Operation
0	Receive data regardless of the content of bit RB8.
1	Receive data only when bit RB8 is 1 and ignore when bit RB8 is 0.

● Setting interrupt

Determine whether the interrupt is to be generated at the end of a receive, through bit RIE [(bit 2: SCI0C)].

Bit RIE	Setting
0	Do not generate interrupt request at the end of receive.
1	Generate interrupt request INT6 at the end of receive.

Interrupt request is generated when bit RDRF [(bit 0: SCI0S)] is set. Bit RDRF is set when the data in the shift register RSR0 is automatically transferred to SCI0RD upon completion of a receive.

Additional interrupt request may be issued upon occurrence of a receive error. To generate the interrupt request upon error, set bit REIE [(bit 4: SCI0C)].

Bit REIE	Setting
0	Do not generate interrupt request upon receive error.
1	Generate interrupt request INT6 upon receive error.

Both receive end interrupt request and receive error interrupt request are assigned in the same vector address.

● Setting receive enable

Set the receiver to receive enable status.

- Bit RE [(bit 0: SCI0C)] ← 1

These steps initialize the receiver. The receiver starts receiving operation when the start bit is input to RxD0 (described later).

● Starting receive timing

While any communication is not taking place, pin RxD0 must be at high level (high level input or pulled up externally). When the pin RxD0 turns low and stays at low level for 1/2 the synchronous clock period, the receiver recognizes this status as start bit input and starts receiving operation.

● Storage of receive data

When the current receive operation ends, the data stored in the shift register RSR0 is automatically transferred to the data register SCI0RD , allowing the next data receive.

When this automatic transfer is taking place, bit RDRF [(bit 0: SCI0S)] is set.

If the interrupt is enabled, the receive end interrupt request is generated upon setting of bit RDRF.

To store the receive data into memory or the like, read the register SCI0RD . Bit RDRF remains 1 until it is read. No other operations cannot clear this bit to 0.

If the register SCI0RD is not read before the next receive operation, overrun error will result (described below).

● Receive error

SCI detects the following receive errors.

- Overrun error: bit OR [(bit 3: SCI0S)].

  This error occurs when the next data has been received before the user program read out the data previously received and stored in the data register SCI0RD . Once the overrun error occurs, no automatic data transfer cannot start after end of current receive.

- Frame error: bit FE [(bit 2: SCI0S)]

  Frame error occurs if the duration of stop bit (high level duration) is shorter than the set value. Even if this error occurs, automatic data transfer starts.

- Parity error: bit PE [(bit 1: SCI0S)]

  This error occurs when the parity check is enabled (bit PEN = 1) and the parity bit is wrong. Even if this error occurs, automatic data transfer starts.

If the transfer format is ignored, the receiver places corresponding error bit: in specific case both PE and FE bits. The overrun error has a priority over other errors. If this error is detected, no other error bits are not set even if corresponding error is detected.

7.3 Operation in SIO mode

This section describes the operation when using SCI in the SIO mode. In this mode the SCI basically transmits and receives at the same time. If it is only to transmit or receive, leave the unused pins as normal I/O pin (port setting).

- Transfer timing

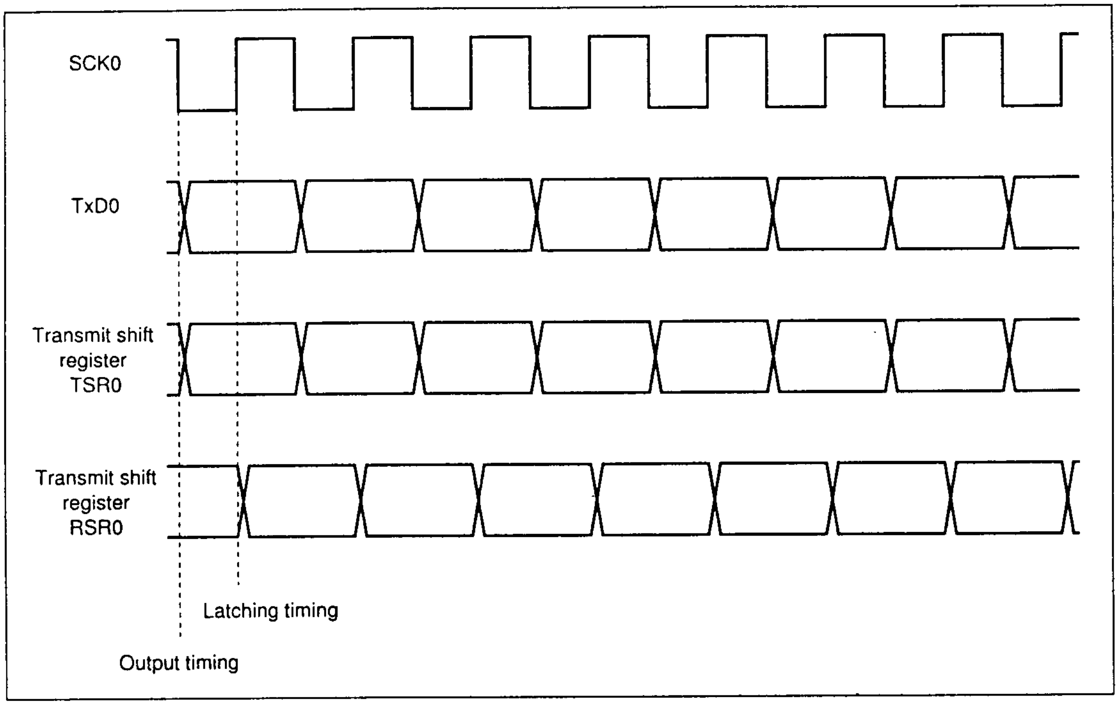


Fig. 9-3 SIO transfer timing

- Setting bit rate

When using the internal synchronous clock generator, set the bit rate by bit rate register SCI0BR . Below shows the relationship between the register value and bit rate.

$$\text{Bit rate (bit/s)} = \frac{X_{TAL}}{4 \times (N+1)} \times 10^6$$

X_{TAL}: main clock frequency (MHz)
N: set value ($0 \leqq N \leqq 16{,}383$)

Synchronous clock generated in the microcomputer can be sent outside of the computer through SCK0 pin. Unlike UART mode, the SIO mode cannot use multiplied-by-16 synchronous clock.

Using an external synchronous clock disables the setting in the register SCI0BR. Unlike UART mode, the SIO mode directly uses the clock frequency coming through SCK0 .

● Setting SIO pin

When operating in the SIO mode, use the following pins.

- SCI0: $P1_0$/TxD0, $P1_1$/RxD0, $P1_2$/SCK0

Set the direction of the pins by using the direction register P1D.

- TxD0 : output
- RxD0 : input
- SCK0 : according to synchronous clock input/output

Next, set the mode of the pins by referring to the description on mode register P1M.

- TxD0 : bit SO0 (bit 0: P1M) ← 1

  Now TxD0 pin is set to transmit pin.

- RxD0 : input (no setting required as functional pin)

  RxD0 pin, when set as input, serves as SIO pin which is now configured as schmitt trigger circuit.

- SCK0 : bit SCK0 (bits 3-2: P1M) ←**

  bit NO0 (bit 1: P1M) ←*

Referring to the table below, set each bit.

Bit SCK0		$P1_2$ pin outpout mode select
0	0	Standard I/O pin (Input direction: Schmitt)
0	1	SCI clock input (Schmitt)
1	0	SCI synchronous clock output (UART: bit rate × 16 clocks; SIO: preset bit rate)
1	1	SCI synchronous clock output

Bit NO0	$P1_2$ output format select
0	CMOS output
1	N-ch open drain output

The mode of these 3 pins can be set individually. The pins not included in SIO can be used as normal I/O pins.

● Initializing SIO (a)

Initialization of SIO is to put it in a transmission/receive ready state so that it immediately starts transmission upon writing data into the transmit data register SCI0TD .

When using SIO first time or when changing transfer format, mode is changed. It must be performed while transmit/receive is disabled.

Set the control register SCI0C as follows.

- SCI0

  Bits TE, RE [bits 1, 0: SCI0C) ← 00

The above procedure stops SIO block and initializes shift registers TSR0 and RSR0 to 0FFH.

● Initializing SIO (b)

Check the status of the SIO operation: using the user program, verify that the status register SCI0S is set as follows:

- Register SCI0S = 30H

The 30H shows that the SCI is in stop; no untransmitted data exists; and no data to be stored in memory exists.

Status of related pins are as follows:

- TxD0 : at high level output
- RxD0 : schmitt trigger input: no pull up resistor
- SCK0 : high level output (may be N-ch open drain) or schmitt trigger input depending on the synchronous clock I/O setting

● Initializing SIO (c)

Transfer format should be set. The format is set through setting of the mode register SCI0M . For further information on this registr, refer to section **7.1 SCI registers.**

- Operation mode of SCI

  To use SCI in SIO mode, set bit SCM [(bit 7: SCI0M)] to 1.

- Transfer order

  Set the transfer order of the bits in the transmit shift register TSR0 and receive shift register RSR0. To do so, use bit MSB (bit 5: SCI0M).

  Bit MSB = 0: LSB of the shift register is first transferred in both transmit and receive mode.

  Bit MSB = 1: MSB of the shift register is first transferred in both transmit and receive mode.

Setting interrupt

Determine whether the interrupt is to be generated at the end of a transfer, through bit TIE [(bit 3: SCI0C)].

Bit TIE	Setting
0	Do not generate interrupt request at the end of transmission.
1	Generate interrupt request INT5 at the end of transmission.

In the SIO mode, bit RIE can also used to enable the interrupt request at the end of transmission. However, only one interrupt request enable is used (transmit: bit TIE, receive: RIE). Should both TIE and RIE bits are set, an interrupt request is generated at both end of transmission and receive.

Setting transfer enable

Set SCI block to transfer enabled status.

- Bit TE [(bit 1: SCI0C) ← 1
- Bit RE [(bit 0: SCI0C)] ← 1

This initializes SIO.

Start and stop of transfer

Writing new data into the transmit data register SCI0TD causes the data to be loaded into the shift data register TSR0 and transfer (transmit/receive) operation starts. The register SCI0TD becomes empty and waits the writing of the data to be transmitted next. When the previous data has been transmitted and the received data has been read out and stored into memory, the next data is written into the register. When transfer completes, data in the receive shift register RSR0 is automatically sent to the data register SCI0RD .

When interrupt request is enabled, end of a transfer is indicated by this request.

If the interrupt request is disabled, end of transfer is verified by monitoring the following bit.

- Bit TxEMP [(bit 5: SCI0S)] is set, and
- Bit RDRF [(bit 0: SCI0S)] is set.

Transfer error (overrun)

If a receive operation ends before the previously received data stored in the receive data register SCI0RD is read out, overrun error will occur and bit OR [(bit 3: SCI0S)] is set. Once this error occurs, no automatic transfer takes place at the end of receive operation.

Chapter 8 Instruction set

8

The SM6000CPU has simple instructions suitable for high speed operations. These instructions are classified into the following 7 functional groups.

(1) Data transfer

(2) Arithmetic operation

(3) Logical operation

(4) Shift and rotate

(5) Bit manipulation

(6) Branch

(7) CPU control

For details of instructions and addressing mode, refer to **2.6 Instruction set and addressing mode.** For the type of instructions which can handle data, refer to **2.5 Data type**. For further information on features of SM6000CPU, refer to **Chapter 2 SM6000CPU.**

8.1 Instruction description

This section gives description on assembly language instructions in the alphabetical order of instruction mnemonics. Each description contains the following information.

(1) Mnemonic

(2) Script: assembler script format

(3) Instruction format: instruction code (binary code), number of bytes/cycles to execute the instruction and addressing mode applicable to the source and destination.

(4) Operation: processes done according to the instruction.

(5) Flag: state of the flags affected by the instruction. These flags are typically C, Z, N and V, and in particular cased I flag. These flags are bits stored in register PSW.

The following symbols are used throughout this section.

- PC: program counter
- SP: stack pointer
- PSW: processor status (register PSW)
- dst: destination
- src: source
- cc: condition code (Refer to **Appendix B-4 Condition code table.**)
- BF, I, V: bit B, bit I, bit V of register PSW
- SFR.b: optional bit of one of the registers mapped to SFR area. (b = 0-15)
- RA, RAH, RAL: Relative address. RA: 8 bit address; RAH: upper 4 bits of 12 bit address; RAL: lower 4 bits of 12 bit address
- IM4: immediate data of nibble (4 bits)
- IM8: immediate data of byte (8 bits)
- IM: immediate data of word (16 bits)
- Rb: lower byte of general purpose register
- seg.DA: 24 bit effective address denoted by the content of the segment register and the direct address.
- Reg's: source or destination is more than one general purpose register
- TEMP: hidden temporary register (used to show progress of instruction execution)

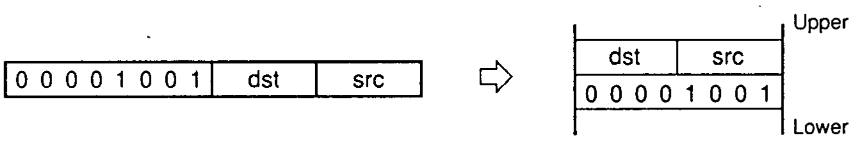
The symbol "←" represents replacement. For example, the operation dst ← dst + src is to add source data to destination data and store the result in the destination.

The symbols "src" and "dst" following the title "Instruction" in the description of an instruction set may collectively represent addressing symbols.

For logical operation, the following symbols are used.

- %: MOD
- &: logical AND
- |: logical OR
- ^: exclusive OR
- !: NOT

An instruction code is written in the order of the byte stored in lower address of the memory space. The example below shows ADC R,R instruction.



When number of cycles are shown as 6/8, the former indicates no branch is made and the latter indicates brach is made according to the condition specified upon execution of the instruction.

The table below shows the SM6010 addressing mode and its symbol.

Addressing mode		Symbol
Register direct	: R addressing	Rn
Register pair direct	: RR addressing	RRn
Register indirect	: (R) addressing	(Rn)
Register indirect with displacement	: disp (R) addressing	disp (Rn)
Post increment register indirect	: (R) + addressing	(Rn) +
Pre decrement register indirect	: - (R) addressing	- (Rn)
Absolute address	: DA addressing	DA
Immediate	: IM addressing	IM
Program counter relative	: RA addressing	RA
Special function register	: SFR addressing	SFR

Rn : R0-R15
RRn : RR0-RR14

In the description, Rn is written as R and RRn as RR.

ADC

(Add With Carry)

Instruction ADC dst, src

Format

	dst	src	byte	cycle
[0 0 0 0 1 0 0 1] [dst] [src]	R	R	2	2
[0 0 0 1 1 0 0 1] [dst] [IM4]	R	IM4	2	2
[0 0 1 0 1 0 0 1] [dst] [src]	SFR	IM	4	8
[0 0 1 1 1 0 0 1] [dst] [src]	SFR	DA	4	10
[0 1 0 0 1 0 0 1] [src] [dst]	DA	SFR	4	10
[0 1 0 1 1 0 0 1] [1 1 1 1 1 1 1 1] [src] [dst]	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.

The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst + src + C

Add the source word and carry to the destination word and store the result into the destination.

Flags

C: Set when the operation results in a carry at MSB. Otherwise, cleared.

Z: Set when the operation results in "0". Otherwise, cleared.

N: Set when the operation results in "1" at MSB. Otherwise, cleared.

V: Set when the operation results in an overflow. Otherwise, cleared.

ADCB

(Add Bytes With Carry)

Instruction ADCB dst, src

Format

				dst	src	byte	cycle
0 0 0 0 0 0 0 1	dst	src		Rb	Rb	2	2
0 0 0 1 0 0 0 1	dst	IM4		Rb	IM4	2	2
0 0 1 0 0 0 0 1	dst	src	1 1 1 1 1 1 1 1	SFR	IMb	4	8
0 0 1 1 0 0 0 1	dst	src		SFR	DA	4	10
0 1 0 0 0 0 0 1	src	dst		DA	SFR	4	10
0 1 0 1 0 0 0 1	1 1 1 1 1 1 1 1	src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.
With this instruction the unit of data of the operand is byte. If a word length register (including general purpose register) is written as an operand, only the lower byte (even address) of the operand is to be executed.The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst + src + C

Add the source byte and carry to the destination byte and store the result into the destination.

Flags

C: Set when the operation results in a carry at MSB. Otherwise, cleared.

Z: Set when the operation results in "0". Otherwise, cleared.

N: Set when the operation results in "1" at MSB. Otherwise, cleared.

V: Set when the operation results in an overflow. Otherwise, cleared.

ADD

(Add Words)

Instruction ADD dst, src

Format

				dst	src	byte	cycle
00001000	dst	src		R	R	2	2
00011000	dst	IM4		R	IM4	2	2
00101000	dst	src		SFR	IM	4	8
00111000	dst	src		SFR	DA	4	10
01001000	src	dst		DA	SFR	4	10
01011000	11111111	src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.

The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst + src

Add the source word to the destination word and store the result into the destination.

Flags C: Set when the operation results in a carry at MSB. Otherwise, cleared.

Z: Set when the operation results in "0". Otherwise, cleared.

N: Set when the operation results in "1" at MSB. Otherwise, cleared.

V: Set when the operation results in an overflow. Otherwise, cleared.

ADDB

(ADD Bytes)

Instruction ADDB dst, src

Format

	dst	src	byte	cycle
0 0 0 0 0 0 0 0 | dst | src	Rb	Rb	2	2
0 0 0 1 0 0 0 0 | dst | IM4	Rb	IM4	2	2
0 0 1 0 0 0 0 0 | dst | src | 1 1 1 1 1 1 1 1	SFR	IMb	4	8
0 0 1 1 0 0 0 0 | dst | src	SFR	DA	4	10
0 1 0 0 0 0 0 0 | src | dst	DA	SFR	4	10
0 1 0 1 0 0 0 0 | 1 1 1 1 1 1 1 1 | src | dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.
With this instruction the unit of data of the operand is byte. If a word length register (including general purpose register) is written as an operand, only the lower byte (even address) of the operand is to be executed.The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst + src

Add the source byte to the destination byte and store the result into the destination.

Flags

C: Set when the operation results in a carry at MSB. Otherwise, cleared.

Z: Set when the operation results in "0". Otherwise, cleared.

N: Set when the operation results in "1" at MSB. Otherwise, cleared.

V: Set when the operation results in an overflow. Otherwise, cleared.

AND

(Logical And)

Instruction AND dst, src

Format

					dst	src	byte	cycle
0 0 0 0 1 1 0 0	dst	src			R	R	2	2
0 0 0 1 1 1 0 0	dst	IM4			R	IM4	2	2
0 0 1 0 1 1 0 0	dst		src		SFR	IM	4	8
0 0 1 1 1 1 0 0	dst		src		SFR	DA	4	10
0 1 0 0 1 1 0 0	src		dst		DA	SFR	4	10
0 1 0 1 1 1 0 0	1 1 1 1 1 1 1 1		src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.
The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst & src

Perform logical AND between the destination word and the source word bit by bit and store the result into the destination.

Flags

C: Left unchanged.

Z: Set when the operation results in "0". Otherwise, cleared.

N: Set when the operation results in "1" at MSB. Otherwise, cleared.

V: Always cleared.

ANDB

(Logical And Bytes)

Instruction ANDB dst, src

Format

	dst	src	byte	cycle
[0 0 0 0 0 1 0 0][dst][src]	Rb	Rb	2	2
[0 0 0 1 0 1 0 0][dst][IM4]	Rb	IM4	2	2
[0 0 1 0 0 1 0 0][dst] [src][1 1 1 1 1 1 1 1]	SFR	IMb	4	8
[0 0 1 1 0 1 0 0][dst] [src]	SFR	DA	4	10
[0 1 0 0 0 1 0 0][src] [dst]	DA	SFR	4	10
[0 1 0 1 0 1 0 0][1 1 1 1 1 1 1 1] [src][dst]	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.
With this instruction the unit of data of the operand is byte. If a word length register (including general purpose register) is written as an operand, only the lower byte (even address) of the operand is to be executed.The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst & src

Perform logical AND between the destination byte and the source byte bit by bit and store the result into the destination.

Flags C: Left unchanged.

Z: Set when the operation results in "0". Otherwise, cleared.

N: Set when the operation results in "1" at MSB. Otherwise, cleared.

V: Always cleared.

BAND

(Bit And)

Instruction BAND BF, src

Format

					dst	src	byte	cycle
0 1 1 1 0 1 1 1	0 0 0 1	src bit	src	1 1 1 0 1 1 1 1	BF	SFR.b	4	10

Operation BF ← BF & SFR.b

Perform logical AND between B flag (BF) of register PSW and a bit of the source word in SFR space and store the result into B flag.

Flags C: Left unchanged

Z: Left unchanged

N: Left unchanged

V: Left unchanged

B: Stores the operation result.

BBC

(Branch on Bit Clear)

Instruction BBC src, RA

Format

					dst	src	byte	cycle
1 1 1 0 1 0 0 0	bit	RAH	SFR	RAL		SFR.b	4	6/8
1 1 1 0 1 1 0 0	bit	R	IM8	RA		IM8(R).b	4	7/9

Branch on cycles: yes/no

Operation

```
if (src.b = = 0) then
  PC ← PC + dst
else
  PC ← PC + 2
endif
```

The (src.b) represents the bit denoted by the source operand.

If the bit denoted by the source operand is 0, add the relative value denoted by the RA to the current PC and move the control to the address specified by the operation result. The BBC instruction cannot operate beyond segment boundary. The range of relative address of BBC instruction is -4096 to + 4094. This is because the instruction code is always at a word boundary, and the addressing is calculated to obtain the signed 13 bit displacement with the LSB of the relative address set at 0. If register indirect with displacement is the addressing mode, relative address range is between -256 and +254.

Flags All flags left unchanged.

BBCS

(Branch on Bit Clear With Set)

Instruction BBCS src, RA

Format

					dst	src	byte	cycle
1 1 1 0 1 0 1 0	bit	RAH	SFR	RAL		SFR.b	4	6/9
1 1 1 0 1 1 1 0	bit	R	IM8	RA		IM8(R).b	4	7/10

Branch on cycles: yes/no

Operation

```
if (src.b = = 0) then
  PC ← PC + dst
  src.b ← 1
else
  PC ← PC + 2
endif
```

The (src.b) represents the bit denoted by the source operand.

If the bit denoted by the source operand is 0, add the relative value denoted by the RA to the current PC and move the control to the address specified by the operation result and set the bit denoted by the source operand to 1. The BBC instruction cannot operate beyond segment boundary. The range of relative address of BBCS instruction is -4096 to + 4094. This is because the instruction code is always at a word boundary, and the addressing is calculated to obtain a signed 13 bit displacement with the LSB of the relative address set at 0. If register indirect with displacement is the addressing mode, relative address range is between -256 and +254.

Flags All flags left unchanged.

When the source operand is C, Z, N or V flag, it is set to 1 if the level of the specified bit before execution of BBCS instruction is 0.

BBS

(Branch on Bit Set)

Instruction BBS src, RA

Format

					dst	src	byte	cycle
1 1 1 0 1 0 0 1	bit	RAH	SFR	RAL		SFR.b	4	6/8
1 1 1 0 1 1 0 1	bit	R	IM8	RA		IM8(R).b	4	7/9

Branch on cycles: yes/no

Operation

```
if (src.b == 1) then
    PC ← PC + dst
else
    PC ← PC + 2
endif
```

The (src.b) represents the bit denoted by the source operand.

If the bit denoted by the source operand is 1, add the relative value denoted by the RA to the current PC and move the control to the address specified by the operation result. The BBS instruction cannot operate beyond segment boundary. The range of relative address of BBS instruction is -4096 to + 4094. This is because the instruction code is always at a word boundary, and the addressing is calculated to obtain a signed 13 bit displacement with the LSB of the relative address set at 0. If register indirect with displacement is the addressing mode, relative address range is between -256 and +254.

Flags All flags left unchanged.

BBSC

(Branch on Bit Clear With Set)

Instruction BBSC dst, RA

Format

					dst	src	byte	cycle
1 1 1 0 1 0 1 1	bit	RAH	SFR	RAL		SFR.b	4	6/9
1 1 1 0 1 1 1 1	bit	R	IM8	RA		IM8(R).b	4	7/10

Branch on cycles: yes/no

Operation

```
if (src.b == 1) then
    PC ← PC + dst
    src.b ← 0
else
    PC ← PC + 2
endif
```

The (src.b) represents the bit denoted by the source operand.

If src.b is 1, add the relative value denoted by the RA to the current PC and move the control to the address specified by the operation result, and clear src.b to 0. The BBSC instruction cannot operate beyond segment boundary. The range of relative address of BBSC instruction is -4096 to + 4094. This is because the instruction code is always at a word boundary, and the addressing is calculated to obtain a signed 13 bit displacement with the LSB of the relative address set at 0. If register indirect with displacement is the addressing mode, relative address range is between -256 and +254.

Flags All flags left unchanged.

When the source operand is C, Z, N or V flag, it is set to 1 if the level of the specified bit before execution of BBCS instruction is 0.

BCLR

(Bit Clear)

Instruction BCLR dst

Format

				dst	src	byte	cycle
1 0 0 0	bit	dst		SFR.b		2	8
1 1 1 0 0 1 1 0	bit	R	IM	IM(R).b		4	9

Operation dst.b ← 0 n = 0 - 15

Clear a bit of the destination word. This instruction cannot be applied to a bit of byte operand.

The (src.b) represents the bit denoted by the destination operand.

Flags All flags left unchanged.

Cleared to 0 if C, Z, N or V flag is specified to the source operand.

BMOV

(Bit Move)

Instruction BMOV dst, src

Format

					dst	src	byte	cycle
0 1 1 1 0 1 1 1	dts bit	src bit	src	dst	SFR.b	SFR.b	4	10

Operation SFR.b (dst) ← SFR.b (src)

Transfer a bit data of the source word in SFR space to a bit place of the destination word in the SFR space.

Flags All flags left unchanged.

Destination bit value is set if C, Z, N or V flag is specified as the source operand.

BOR

(Bit Or)

Instruction BOR BF, src

Format

						dst	src	byte	cycle
0 1 1 1 0 1 1 1	0 0 1 0	src bit	src	1 1 1 0 1 1 1 1		BF	SFR.b	4	10

Operation BF ← BF | SFR.b

Perform logical OR between B flag (BF) of PSW register and a bit of the source word in SFR space and store the result into B flag.

Flags

C: Left unchanged.

Z: Left unchanged.

N: Left unchanged.

V: Left unchanged.

B: Stores the operation result.

BR

(Branch)

Instruction BR cc, RA

Format

1 0 1 1	cc	RA

dst	src	byte	cycle
		2	3/5

Branch on cycles: yes/no

Operation

```
if (cc == true) then
    PC ← PC + dst
else
    PC ← PC + 2
endif
```

If the specified condition (cc) matches, add the value denoted by the destination operand to the current PC and move the control to the address specified by the addition operation result. The BR instruction cannot operate beyond segment boundary. The range of relative address of BR instruction is -256 to + 254. This is because the instruction code is always at a word boundary, and the addressing is calculated to obtain signed 9 bit with the LSB of the relative address set at 0.

Flags All flags left unchanged.

BSET

(Bit Set)

Instruction BSET dst

Format

				dst	src	byte	cycle
1 0 0 1	bit	dst		SFR.b		2	8
1 1 1 0 0 1 1 1	bit	R	IM	IM(R).b		4	9

Operation dst.b ← 1 n = 0 - 15

Set a bit of the destination word. Not applicable to a bit of byte operand.

The (dst.b) represents the bit denoted by the destination operand.

Flags All flags left unchanged. Set to 1 if C, Z, N or V flag is specified to the source operand.

BTST

(Bit Test)

Instruction BTST dst, src

Format

0 1 1 1 0 1 1 0	dst	src

dst	src	byte	cycle
SFR	IM	4	6

Operation dst & src

Perform logical AND between the destination word and the source word bit by bit and store the result into flag. The value is not returned to any place.

Flags C: Left unchanged.

Z: Set if operation results in "0". Otherwise, cleared.

N: Set if operation results in "1" at MSB. Otherwise, cleared.

V: Always cleared.

BXOR

(Bit Exclusive Or)

Instruction BXOR BF, src

Format

					dst	src	byte	cycle
0 1 1 1 0 1 1 1	0 0 1 1	src bit	src	1 1 1 0 1 1 1 1	BF	SFR.b	4	10

Operation BF ← BF ^ src.bit

Perform exclusive OR between B flag (BF) of PSW register and a bit of the source word in SFR space and store the result into B flag.

Flags

C: Left unchanged.

Z: Left unchanged.

N: Left unchanged.

V: Left unchanged.

B: Set to the operation result.

CALL

(Call Subroutine)

Instruction CALL cc, dst

Format

				dst	src	byte	cycle
1 1 1 0 0 0 0 1	cc	dst		(R)		2	3/7
1 1 1 0 0 0 0 0	cc	1 1 1 1	dst	DA		4	4/7

Branch on cycles: yes/no

Operation

```
if (cc == true) then
    SP ← SP - 2
    (SP) ← PC
    PC ← dst
else
    PC ← PC+2
endif
```

When the specified condition is met, move the control to the address denoted by the destination operand. At the same time, the return address is stored in the memory location denoted by the stack pointer. CALL instruction cannot operate beyond the segment boundary.

Flags All flags left unchanged.

CALR

(Call Relative Subroutine)

Instruction CALR dst

Format

1 0 1 0	dst H	dst L

dst	src	byte	cycle
RA		2	7

Operation SP ← SP – 2

(SP) ← PC

PC ← PC + dst

Add the value denoted by the destination operand to the current PC and move the control to the address specified by the operation result. At the same time, the return address is stored in the memory location denoted by the stack pointer. CALR instruction cannot operate beyond the segment boundary. The range of relative address of CALR instruction is -4096 to + 4094. This is because the instruction code is always at a word boundary, and the addressing is calculated to obtain a signed 13 bit displacement with the LSB of the relative address set at 0.

Flags All flags left unchanged.

CLR

(Clear Word)

Instruction CLR dst

Format

0 1 1 1 0 1 0 1	dst

dst	src	byte	cycle
SFR		2	6

Operation dst ← 0

Clear the destination word data.

Flags All flags left unchanged.

Always cleared if PSW register is specified as the source operand.

CMP

(Compare Words)

Instruction CMP dst, src

Format

					dst	src	byte	cycle
0 0 0 0 1 1 1 1	dst	src			R	R	2	2
0 0 0 1 1 1 1 1	dst	IM4			R	IM4	2	2
0 0 1 0 1 1 1 1	dst		src		SFR	IM	4	6
0 0 1 1 1 1 1 1	dst		src		SFR	DA	4	8
0 1 0 0 1 1 1 1	src		dst		DA	SFR	4	8
0 1 0 1 1 1 1 1	1 1 1 1 1 1 1 1		src	dst	SFR	SFR	4	8

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.

The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst – src

Subtract the source word from the destination word and set each flag without store the result.

Flags C: Set if a borrow occurs at MSB as the result of the operation. Otherwise, cleared.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in "1" at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

CMPB

(Compare Bytes)

Instruction CMPB dst, src

Format

				dst	src	byte	cycle
0 0 0 0 0 1 1 1	dst	src		Rb	Rb	2	2
0 0 0 1 0 1 1 1	dst	IM4		Rb	IM4	2	2
0 0 1 0 0 1 1 1	dst	src	1 1 1 1 1 1 1 1	SFR	IMb	4	6
0 0 1 1 0 1 1 1	dst	src		SFR	DA	4	8
0 1 0 0 0 1 1 1	src	dst		DA	SFR	4	8
0 1 0 1 0 1 1 1	1 1 1 1 1 1 1 1	src	dst	SFR	SFR	4	8

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.

With this instruction the unit of data of the operand is byte. If a word length register (including general purpose register) is written as an operand, only the lower byte (even address) of the operand is to be executed.The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst – src

Subtract the source bytes from the destination bytes and set each flag without store the result.

Flags

C: Set if a borrow occurs at MSB as the result of the operation. Otherwise, cleared.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

COM

(Complement Words)

Instruction COM dst

Format

0 1 1 1 0 0 0 1	dst

dst	src	byte	cycle
SFR		2	8

Operation dst ← !dst

Replace the word data denoted by the operand with 1's complement.

Flags

C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared

DADD

(Decimal Add Bytes)

Instruction DADD dst, src

Format

0 1 1 1 1 1 1 0	dst	src

dst	src	byte	cycle
Rb	Rb	2	3

Operation dst ← dst(BCD) + src(BCD)

Add the source byte to the destination byte, both bytes should be recognized as 2 digit BCD number. Store the operation result onto the destination.

Flags

C: Set if the operation results in a carry at MSB. Otherwise, cleared.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

DBNZ

(Decriment and Branch on Non-Zero)

Instruction DBNZ R, dst

Format

1 1 1 1 1 1 0 0	R	1 1 1 1	dst

dst	src	byte	cycle
DA		4	6/6

Branch on cycles: yes/no

Operation

```
R ← R – 1
if (R ! = 0) then
    PC ← dst
else
    PC ← PC + 2
endif
```

Subtract 1 from the content of the register specified by R. If the operation result is not 0, move the control to the address specified by the destination. DBNZ instruction cannot operate beyond the segment boundary.

Flags All flags left unchanged.

DI

(Disable Interrupts)

Instruction DI

Format

1 1 1 1 0 1 1 1	1 1 1 1 1 1 1 1

dst	src	byte	cycle
		2	2

Operation I ← 0

Disable maskable interrupts.

Flags

C: Left unchanged.

Z: Left unchanged.

N: Left unchanged.

V: Left unchanged.

I: Cleared.

DIVLS

(Divide Long Signed)

Instruction DIVLS dst, src

Format

0 1 1 1 1 1 0 0	dst	src

dst	src	byte	cycle
RR	RR	2	39

Operation dst ← dst ÷ src(L)

src(H) ← dst % src(L)

(32 bits ÷ 16 bits → 32 bits .... 16 bits)

Divide the destination long word by the source lower word, both treated as signed data. The operation results in 32 bit quotient and 16 bit remainder.

Only register pair can be assigned as addressing mode to both destination and source. The quotient of the result of division of lower destination register pair with source lower word is stored in the register pair denoted by the destination and the remainder in the upper word of the register pair denoted by the source.

For example, the dividend (dst) and divisor (src) are stored in registers PR0 and R2. Executing

```
    DIVLS  RR0, RR2
```

causes the quotient to be stored in PR0 and remainder in R3. The upper word of the source and destination are updated. The value of the lower word (divisor R2) remains unchanged. Register pair can be used in addressing mode and only by multiplication or division instruction, and cannot be used for other purposes (e.g. transfer).

If 0 division, overflow flag V is set but operation result value is not defined, i.e. stored value is unknown.

Flags C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

DIVLU

(Divide Long Unsigned)

Instruction DIVLU   dst, src

Format

0 1 1 1 1 1 0 1	dst	src

dst	src	byte	cycle
RR	RR	2	39

Operation dst ← dst ÷ src(L)

src(H) ← dst % src(L)

(32 bits ÷ 16 bits → 32 bits ... 16 bits)

Divide the destination long word by the source lower word. Both words should be treated as unsigned data. The operation results in 32 bit quotient and 16 bit remainder.

Only register pair can be assigned as addressing mode to both destination and source. The quotient as the result of division of destination register pair by the source lower word is stored in the register pair denoted by the destination and the remainder in the upper word of the register pair denoted by the source.

For example, the dividend (dst) and divisor (src) are stored in registers PR0 and R2. Executing

```
    DIVLU  RR0, RR2
```

causes the quotient to be stored in PR0 and remainder in R3. The upper word of the source and destination are updated. The value of the lower word (divisor R2) remains unchanged. Register pair can be used in addressing mode and only by multiplication or division instruction, and cannot be used for other purposes (e.g. transfer). If 0 division, overflow flag V is set but operation result value is not defined, i.e. stored value is unknown.

Flags C: Left unchanged

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

DIVS

(Divide Signed)

Instruction DIVS dst, src

Format

0 1 1 1 1 0 1 0	dst	src

dst	src	byte	cycle
RR	R	2	18

Operation dst(L) ← dst(L) ÷ src

dst(H) ← dst(L) % src

(16 bits ÷ 16 bits → 16 bits ... 16 bits)

Divide the destination word by the source word. Both words should be treated as signed data. The operation result is stored in destination. The operation results in 16 bit quotient and 16 bit remainder.

Only register pair can be assigned as addressing mode. The quotient as the result of division of lower word of destination register pair by the source is stored in the lower word of the register pair denoted by the destination and the remainder in the upper word of the register pair.

For example, the dividend (DST) and divisor (SRC) are stored in registers R0 and R3. Executing

```
DIVS    RR0, R3
```

causes the quotient to be stored in R0 and remainder in R1. The upper word of the register pair denoted by the destination is updated by the remainder data. Register pair can be used in addressing mode and only by multiplication or division instruction, and cannot be used for other purposes (e.g. transfer). If 0 division or maximum negative value (-32768)÷(-1) is executed, overflow flag V is set but operation result value is not defined, i.e. stored value is unknown.

Flags C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

DIVU

(Divide Unsigned)

Instruction DIVU dst, src

Format

0 1 1 1 1 0 1 1	dst	src

dst	src	byte	cycle
RR	R	2	18

Operation st(L) ← dst(L) ÷ src

dst(H) ← dst(L) % src

(16 bits ÷ 16 bits → 16 bits ... 16 bits)

Divide the destination word by the source word. Both words should be treated as unsigned data. The operation is stored in destination. The operation results in 16 bit quotient and 16 bit remainder. Only register pair can be assigned as addressing mode. The quotient as the result of division of lower word of destination register pair by the source is stored in the lower word of the register pair denoted by the destination and the remainder in the upper word of the register pair.

For example, the dividend (dst) and divisor (src) are stored in registers R0 and R3. Executing

```
    DIVS  RR0, R3
```

causes the quotient to be stored in R0 and remainder in R1. The upper word of the register pair denoted by the destination is updated by the remainder data. Register pair can be used in addressing mode and only by multiplication or division instruction, and cannot be used for other purposes (e.g. transfer). If 0 division is executed, overflow flag V is set but operation result value is not defined, i.e. stored value is unknown.

Flags C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

DSUB

(Decimal Subtract Bytes)

Instruction DSUB dst, src

Format

0 1 1 1 1 1 1 1	dst	src

dst	src	byte	cycle
Rb	Rb	2	3

Operation dst ← dst(BCD) – src(BCD)

Subtract the destination bytes from the source bytes. Both bytes should be treated as two digit BCD number. Store the result into the destination.

Flags

C: Set if the operation results in a borrow at MSB. Otherwise, cleared.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

EI

(Enable Interrupts)

Instruction EI

Format

		dst	src	byte	cycle
1 1 1 1 0 1 1 0	1 1 1 1 1 1 1 1			2	2

Operation I ← 1

Enable maskable interrupts. After recovery from hardware reset, all maskable interrupts are disabled. To use interrupts, this instruction must be executed. In addition, enable or disable of each maskable interrupt must be set. See related functional description.

Flags

C: Left unchanged.

Z: Left unchanged.

N: Left unchanged.

V: Left unchanged.

I: Set

HALT

(CPU HALT)

Instruction HALT

Format

		dst	src	byte	cycle
1 1 1 1 0 0 0 1	1 1 1 1 1 1 1 1			2	-

Operation CPU HALT

Stop supplying the system clock to the CPU to stop the program counter PC.

HALT instruction does not stop the main clock. Clocks (divided main clock) necessary for operation of peripheral blocks and sampling clocks for I/Os will also continue.

The CPU exits the halt mode when it receives an external reset signal or an interrupt request.

Flags All flags left unchanged.

IRET

(Return From Interrupt)

Instruction IRET cc

Format

1 1 1 1 0 1 0 1	cc	1 1 1 1

dst	src	byte	cycle
		2	6/10

Operation

```
if (cc == true) then
    PSW ← (SP)
    SP ← SP + 2
    SEG ← (SP)
    SP ← SP + 2
    PC ← (SP)
    SP ← SP + 2
endif
```

Return from the interrupt routine when the specified condition (cc) is met. Do not use this instruction for returning after executing CALL, CALR or SCALL instruction.

Flags

C: Returns to the previous state.

Z: Returns to the previous state.

N: Returns to the previous state.

V: Returns to the previous state.

I: Returns to the previous state.

JMP

(Jump)

Instruction JMP cc, dst

Format

				dst	src	byte	cycle
1 1 1 0 0 1 0 0	cc	dst		(R)		2	3/5
1 1 1 0 0 0 1 1	cc	1 1 1 1	dst	DA		4	5/5

Branch on cycles: yes/no

Operation

```
if (cc== ture) then
    PC ← dst
else
    PC ← PC + 2
endif
```

Move the control to the address specified by the destination operand when the specified condition is met. Jump instruction cannot operate beyond the segment boundary.

Flags All flags left unchanged.

LDM

(Load Multiple)

Instruction LDM dst, src

Format

				dst	src	byte	cycle
1 1 1 1 1 1 0 1	src	1 1 1 1	dst	Reg's	(R)	4	10-55

The cycle number depends on the number of registers to load.

Operation

```
TEMP ← R(src)
CNT ← 0
While (CNT ! == 16)
{
    if (dst.CNT == 1) then
        R(CNT) ← (TEMP)
        TEMP ← TEMP + 2
    endif
    CNT ← CNT + 1
}
```

Load the content at address specified by the source register to the registers denoted by the destination. In the same way as POP instruction, data is loaded onto the registers and the pointer is incremented. The registers to be loaded are specified as shown below.

```
LDMR0-R5:R7:R9:R14, (R10)
```

R0, R1, R2, R3, R4, R5, R7, R9, R14 are loaded in that order.

The register value set into the source is not updated after execution of LDM instruction. That is, the value of the source register is not changed even if the instruction is executed.

Bits 0-15 in the destination operand (Reg's) denoted by the instruction correspond to R0-R15 registers. When Reg's is converted into an instruction code, the bits corresponding to the registers pointed to by the operand are set. For example, when R0-R5:R7:R9:R14 are described in the destination operand, Reg's is decoded into the instruction 0100 0010 1011 1111B.

Flags All flags left unchanged.

LINK

(Link Stack Frame)

Instruction LINK src

Format

1 1 1 1 0 0 1 0	src

dst	src	byte	cycle
	IM8	2	7

Operation SP ← SP – 2

(SP) ← FP

FP ← SP

SP ← SP – src

Push the frame pointer FP onto the stack, store the content of stack pointer SP, and subtract the value specified by the source from SP to secure the variable area on the stack which can be accessed by FP. The stack frame can be returned back to the previous status by using UNLINK instruction. Up to 255 byte frame can be configured.

Flags All flags left unchanged.

MOV

(Move Words)

Instruction MOV dst, src

Format

					dst	src	byte	cycle
1 1 0 0 1 0 0 0	dst	src			R	R	2	2
1 1 0 0 1 0 0 1	dst	src			R	IM4	2	2
1 1 0 0 1 0 1 0	dst	src			R	(R)	2	6
1 1 0 0 1 0 1 1	dst	src			R	(R)+	2	6
1 1 0 0 1 1 0 0	dst	src			(R)	R	2	6
1 1 0 0 1 1 0 1	dst	src			−(R)	R	2	7
1 1 0 0 1 1 1 0	dst	src	IM		R	IM(R)	4	7
1 1 0 0 1 1 1 1	dst	src	IM		IM(R)	R	4	7
1 1 0 1 1 0 0 0	dst		IM		SFR	IM	4	6
1 1 0 1 1 0 0 1	dst		src		SFR	DA	4	8
1 1 0 1 1 0 1 0	src		dst		DA	SFR	4	8
1 1 0 1 1 0 1 1	1 1 1 1 1 1 1 1		src	dst	SFR	SFR	4	8

Operation dst ← src

Store the source word in the destination.

Flags All flags left unchanged.

MOVB

(Move Bytes)

Instruction `MOVB    dst, src`

Format

					dst	src	byte	cycle
1 1 0 0 0 0 0 0	dst	src			Rb	Rb	2	2
1 1 0 0 0 0 0 1	dst	src			Rb	IM4	2	2
1 1 0 0 0 0 1 0	dst	src			Rb	(R)	2	6
1 1 0 0 0 0 1 1	dst	src			Rb	(R)+	2	6
1 1 0 0 0 1 0 0	dst	src			(R)	Rb	2	6
1 1 0 0 0 1 0 1	dst	src			−(R)	Rb	2	7
1 1 0 0 0 1 1 0	dst	src	IM		Rb	IM(R)	4	7
1 1 0 0 0 1 1 1	dst	src	IM		IM(R)	Rb	4	7
1 1 0 1 0 0 0 0	dst		IMb	1 1 1 1 1 1 1 1	SFR	IMb	4	6
1 1 0 1 0 0 0 1	dst		src		SFR	DA	4	8
1 1 0 1 0 0 1 0	src		dst		DA	SFR	4	8
1 1 0 1 0 0 1 1	1 1 1 1 1 1 1 1		src	dst	SFR	SFR	4	8

Operation dst ← src

Store the source byte in the destination.

Flags All flags left unchanged.

MOVSE

(Move Byte with Sign Extend)

Instruction MOVSE dst, src

Format

			dst	src	byte	cycle
1 1 0 1 0 1 1 0	dst	src	R	Rb	2	2
1 1 0 1 0 1 1 1	dst	src	SFR	DA	4	8

Operation dst ← Extend Sign (src)

Extend the sign of the source byte data and store it into the destination as word data.

Flags C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Left unchanged

MOVZE

(Move Byte with Zero Extend)

Instruction MOVZE dst, src

Format

			dst	src	byte	cycle
1 1 0 1 1 1 1 0	dst	src	R	Rb	2	2
1 1 0 1 1 1 1 1	dst	src	SFR	DA	4	8

Operation dst ← Extend Zero (src)

Zero extend the source byte data and store it into the destination as word data.

Flags

C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Always cleared.

V: Left unchanged.

MULS

(Multiply Signed)

Instruction MULS dst, src

Format

0 1 1 1 1 0 0 0	dst	src

dst	src	byte	cycle
RR	R	2	10

Operation dst ← dst × src

(16 bits × 16 bits → 32 bits)

Multiply the destination and source words as signed data and store the result (32 bits) into the destination. Only register pair can be assigned as addressing mode for destination. Multiply the lower words of destination register pair and source register and the result is stored in the register pair specified by the destination. For example, executing the following operation with multiplicand (dst) and multiplier (src) stored in R0 and R2,

```
MULS   RR0, R2
```

the upper word from the operation is stored into R1 and the lower word into R0. The upper word of the register pair denoted by the destination is updated with the upper word data obtained from the operation. Register pair can be used in addressing mode and only by multiplication or division instruction, and cannot be used for other purposes (e.g. transfer).

Flags C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

MULU

(Multiply Unsigned)

Instruction MULU dst, src

Format

0 1 1 1 1 0 0 1	dst	src

dst	src	byte	cycle
RR	R	2	10

Operation dst ← dst × src

(16 bits × 16 bits → 32 bits)

Multiply the destination and source words as unsigned data and store the result (32 bits) into the destination. Only register pair can be assigned as addressing mode for destination. Multiply the lower words of destination register pair and source register and the result is stored in the register specified by the destination. For example, executing the following operation with multiplicand (dst) and multiplier (src) stored in R0 and R2,

```
MULU    RR0, R2
```

the upper word from the operation is stored into R1 and the lower word into R0. The upper word of the register pair denoted by the destination is updated with the upper word data obtained from the operation.Register pair can be used in addressing mode and only by multiplication or division instruction, and cannot be used for other purposes (e.g. transfer).

Flags

C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

NEG

(Negate Words)

Instruction NEG dst

Format

0 1 1 1 0 0 0 0	dst

dst	src	byte	cycle
SFR		2	8

Operation dst ← –dst

Replace the word data specified by the operand with 2's complement. However, when the operation results in the maximum negative number (8000H), this value is stored. This is because there is no positive maximum number against the negative maximum number in 2's complement expression.

Flags

C: Set if the operation results in a carry at MSB. Otherwise, cleared.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

NOP

(No Operation)

Instruction NOP

Format

Format	dst	src	byte	cycle
1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1			2	2

Operation No Operation

Execute nothing. Usually used for waiting a program.

Flags All flags left unchanged.

OR

(Logical Or Words)

Instruction OR dst, src

Format

				dst	src	byte	cycle
0 0 0 0 1 1 0 1	dst	src		R	R	2	2
0 0 0 1 1 1 0 1	dst	IM4		R	IM4	2	2
0 0 1 0 1 1 0 1	dst	src		SFR	IM	4	8
0 0 1 1 1 1 0 1	dst	src		SFR	DA	4	10
0 1 0 0 1 1 0 1	src	dst		DA	SFR	4	10
0 1 0 1 1 1 0 1	1 1 1 1 1 1 1 1	src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.
The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst |src

Perform logical OR between the destination word and source word bit by bit and store the result into the destination.

Flags

C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

ORB

(Logical Or Byte)

Instruction ORB dst, srcc

Format

				dst	src	byte	cycle
0 0 0 0 0 1 0 1	dst	src		Rb	Rb	2	2
0 0 0 1 0 1 0 1	dst	IM4		Rb	IM4	2	2
0 0 1 0 0 1 0 1	dst	src	1 1 1 1 1 1 1 1	SFR	IMb	4	8
0 0 1 1 0 1 0 1	dst	src		SFR	DA	4	10
0 1 0 0 0 1 0 1	src	dst		DA	SFR	4	10
0 1 0 1 0 1 0 1	1 1 1 1 1 1 1 1	src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.
With this instruction the unit of data of the operand is byte. If a word length register (including general purpose register) is written as an operand, only the lower byte (even address) of the operand is to be executed.
The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst |src

Perform logical OR between the destination byte and the source byte bit by bit and store the result into the destination.

Flags

C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

POP

(Pop Word From Stack)

Instruction POP dst

Format

0 1 1 1 0 1 0 0	dst

dst	src	byte	cycle
SFR		2	8

Operation dst ← (SP)

SP ← SP + 2

Return the word data of the stack to the destination. Byte operand cannot be applied.

Flags All flags left unchanged.

If the register PSW is the destination operand, the content of PSW is changed according to the word data of the stack to be returned.

PUSH

(Push Word to Stack)

Instruction PUSH src

Format

0 1 1 1 0 0 1 1	src

dst	src	byte	cycle
	SFR	2	8

Operation SP ← SP − 2

(SP) ← src

Save the source word data onto the stack. Byte operand cannot be applied.

Flags All flags left unchanged.

RET

(Return From Subroutine)

Instruction RET cc

Format

1 1 1 1 1 0 0 0	cc	1 1 1 1

dst	src	byte	cycle
		2	6/6

Branch on cycles: yes/no

Operation

```
if (cc == true) then
    PC ← (SP)
    SP ← SP + 2
endif
```

When the specified condition (cc) is met, read the return address stored on the stack and move the control to that address. This instruction is for CALL and CALR instructions only and should not be used for returning after execution of SCALL instruction.

Flags All flags left unchanged.

ROL

(Rotate Left)

Instruction ROL dst, src

Format

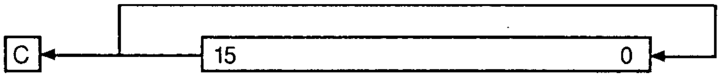
			dst	src	byte	cycle
0 1 1 0 0 1 0 0	dst	src	R	R	2	5+
0 1 1 0 1 1 0 0	dst	IM4	R	IM4	2	5+

When R is specified as the source, only lower 4 bits are effective.

The execution cycle increases by 1 as the number of shifts is increased by 1.

The "cycle" in this format represents no shift.

Operation Left rotate the destination word the number of times denoted by the source. The MSB is inserted into the carry and LSB places. The maximum number of rotations is 15.



Flags

C: Set if the operation results in a carry at MSB. Otherwise, cleared.

(No change at 0 time rotation)

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

ROR

(Rotate Right)

Instruction ROR dst, src

Format

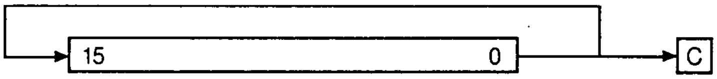
			dst	src	byte	cycle
0 1 1 0 0 0 1 1	dst	src	R	R	2	5+
0 1 1 0 1 0 1 1	dst	IM4	R	IM4	2	5+

When R is specified as the source, only lower 4 bits are effective.

The execution cycle increases by 1 as the number of shifts is increased by 1.

The "cycle" in this format represents no shift.

Operation Right rotate the destination word the number of times denoted by the source. The LSB is inserted into the carry and MSB places. The maximum number of rotations is 15.



Flags C: Set if the operation results in a carry at LSB. Otherwise, cleared.

(No change at 0 time rotation)

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

RST

(Restart CPU)

Instruction RST

Format

1 1 1 1 1 0 1 1	1 1 1 1 1 1 1 1

dst	src	byte	cycle
		2	-

Operation PC ← 000100H

Generate software reset signal and initialize the CPU and I/O's.

After software resetting, the program continues at address 000100H.

Flags No flags are undefined.

SBC

(Subtract Words With Carry)

Instruction SBC dst, src

Format

					dst	src	byte	cycle
0 0 0 0 1 0 1 1	dst	src			R	R	2	2
0 0 0 1 1 0 1 1	dst	IM4			R	IM4	2	2
0 0 1 0 1 0 1 1	dst		src		SFR	IM	4	8
0 0 1 1 1 0 1 1	dst		src		SFR	DA	4	10
0 1 0 0 1 0 1 1	src		dst		DA	SFR	4	10
0 1 0 1 1 0 1 1	1 1 1 1 1 1 1 1		src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.
The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst – src – C

Subtract the source word and carry from the destination word and store the result into the destination.

Flags

C: Set if the operation results in a borrow at MSB. Otherwise, cleared.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

SBCB

(Subtract Bytes With Carry)

Instruction SBCB dst, src

Format

					dst	src	byte	cycle
0 0 0 0 0 0 1 1	dst	src			Rb	Rb	2	2
0 0 0 1 0 0 1 1	dst	IM4			Rb	IM4	2	2
0 0 1 0 0 0 1 1	dst		src	1 1 1 1 1 1 1 1	SFR	IMb	4	8
0 0 1 1 0 0 1 1	dst		src		SFR	DA	4	10
0 1 0 0 0 0 1 1	src		dst		DA	SFR	4	10
0 1 0 1 0 0 1 1	1 1 1 1 1 1 1 1		src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.
With this instruction the unit of data of the operand is byte. If a word length register (including general purpose register) is written as an operand, only the lower byte (even address) of the operand is to be executed.The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst – src – C

Subtract the source bytes and carry from the destination bytes and store the result into the destination.

Flags

C: Set if the operation results in a borrow at MSB. Otherwise, cleared.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

SCALL

(Segment Call Subroutine)

Instruction SCALL dst

Format

			dst	src	byte	cycle
1 1 1 0 0 0 1 0	seg	dst	seg:DA		4	9

Operation SP ← SP − 2

(SP) ← PC

SP ← SP − 2

(SP) ← SEG

SEG ← seg

PC ← dst

Move the control to the segment address denoted by the destination operand (24 bit effective address). The return segment and address are stored in the memory denoted by the stack pointer. After using SCALL instruction to move to a different subroutine, use SRET instruction for returning. Do not use RET instruction - the program cannot recover the return segment, causing an error.

Flags All flags left unchanged.

SJMP

(Segment Jump)

Instruction SJMP dst

Format

			dst	src	byte	cycle
1 1 1 0 0 1 0 1	seg	dst	seg:DA		4	5

Operation SEG ← seg

PC ← dst

Move the control to the segment address denoted by the destination operand (24 bit effective address).

Flags All flags left unchanged.

SLL

(Shift Left Logical)

Instruction SLL dst, src

Format

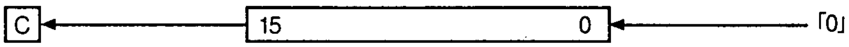
	dst	src	byte	cycle
0 1 1 0 0 0 0 0 | dst | src	R	R	2	5+
0 1 1 0 1 0 0 0 | dst | IM4	R	IM4	2	5+

When R is specified as the source, only lower 4 bits are effective.

The execution cycle increases by 1 as the number of shifts is increased by 1.

The "cycle" in this format represents no shift.

Operation Left shift the destination word the number of times specified by the source. The LSB position is filled with 0 and MSB is sent to the carry. Up to 15 shifts can be made by a single SLL instruction.



Flags

C: Set if the operation results in a carry at MSB. Otherwise, cleared.

(No change at 0 time shift)

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

SRA

(Shift Right Arithmetic)

Instruction SRA dst, src

Format

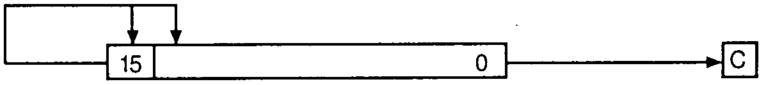
			dst	src	byte	cycle
0 1 1 0 0 0 1 0	dst	src	R	R	2	5+
0 1 1 0 1 0 1 0	dst	IM4	R	IM4	2	5+

When R is specified as the source, only lower 4 bits are effective.

The execution cycle increases by 1 as the number of shifts is increased by 1.

The "cycle" in this format represents no shift.

Operation Right shift the destination word the number of times specified by the source. The value at MSB position and the sign remain unchanged. Up to 15 shifts can be made by a single SRA instruction.



Flags

C: Set if the operation results in a carry at LSB. Otherwise, cleared.

(No change at 0 time)

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

SRET

(Return From Segment Subroutine)

Instruction SRET cc

Format

1 1 1 1 1 0 0 1	cc	1 1 1 1

dst	src	byte	cycle
		2	6/8

Branch on cycles: yes/no

Operation

```
if (cc == ture) then
        SEG ← (SP)
        SP ← SP + 2
        PC ← (SP)
        SP ← SP+2
endif
```

If the specified condition is met, read the return address stored on the stack and move the control to that address. This instruction is for SCALL only, and should not be used for returning after execution of CALL or CALR instruction or returning after interrupt processing.

Flags All flags left unchanged.

SRL

(Shift Right Logical)

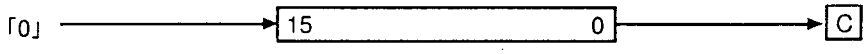
Instruction SRL dst, src

Format

			dst	src	byte	cycle
0 1 1 0 0 0 0 1	dst	src	R	R	2	5+
0 1 1 0 1 0 0 1	dst	IM4	R	IM4	2	5+

When R is specified as the source, only lower 4 bits are effective.
The execution cycle increases by 1 as the number of shifts is increased by 1.
The "cycle" in this format represents no shift.

Operation Right shift the destination word the number of times specified by the source. The MSB position is filled with 0 and LSB is sent to the carry. Up to 15 shifts can be made by a single SRL instruction.



Flags C: Set if the operation results in a carry at LSB. Otherwise, cleared.

(No change at 0 time shift)

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

STM

(Store Multiple)

Instruction STM dst, src

Format

1 1 1 1 1 1 1 0	dst	1 1 1 1	src

dst	src	byte	cycle
(R)	Reg's	4	8-53

Operation

```
TEMP ← R(dst)
CNT ← src
while (CNT != 16)
{
    if (dst.CNT != 1) then
        TEMP ← TEMP+2
        (TEMP) ← R (src)
    endif
    CNT ← CNT-1
}
```

Store content of registers denoted by the source into the addresses specified by the destination. In the same way as PUSH instruction, first decrement the contents of the register and then write the data. When the data of the specified registers are stored, the destination register is updated. In this way, the contents of registers before executing STM instruction can be stored. Specify the registers to store as following example.

```
STM    (R10), R0-R5:R7:R9:R14
```

The registers are stored in the location starting at the address specified by R10 and then R0, R1, R2, R4, R5, R7, R9 and R14. The content of the register specified as the destination is not updated at the end of STM instruction. That is, the value of the destination register remains unchanged even if STM is executed.

Bits 0-15 in the destination operand (Reg's) denoted by the instruction correspond to R0-R15 registers. When Reg's is converted into an instruction code, the bits corresponding to the registers pointed to by the operand are set. For example, when R0-R2:R7:R9:R14:R15 are described in the destination operand, Reg's is decoded into the instruction 1100 0010 1000 0111B.

Flags All flags left unchanged.

STOP

(Stop Oscillation)

Instruction STOP

Format

	dst	src	byte	cycle
1 1 1 1 0 0 0 0 | 1 1 1 1 1 1 1 1			2	-

Operation STOP oscillation

Turn off main clock oscillation. The CPU and peripheral blocks using the main clock and I/O's using the sampling clock stop. An interrupt or hardware reset returns the CPU from the stop condition.

Flags All flags left unchanged.

SUB

(Subtract Words)

Instruction SUB dst, src

Format

					dst	src	byte	cycle
0 0 0 0 1 0 1 0	dst	src			R	R	2	2
0 0 0 1 1 0 1 0	dst	IM4			R	IM4	2	2
0 0 1 0 1 0 1 0	dst		src		SFR	IM	4	8
0 0 1 1 1 0 1 0	dst		src		SFR	DA	4	10
0 1 0 0 1 0 1 0	src		dst		DA	SFR	4	10
0 1 0 1 1 0 1 0	1 1 1 1 1 1 1 1		src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.

The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst – src

Subtract the source word from the destination word and store the result into the destination.

Flags

C: Set if the operation results in a borrow at MSB. Otherwise, cleared.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

SUBB

(Subtract Bytes)

Instruction SUBB dst, src

Format

				dst	src	byte	cycle
0 0 0 0 0 0 1 0	dst	src		Rb	Rb	2	2
0 0 0 1 0 0 1 0	dst	IM4		Rb	IM4	2	2
0 0 1 0 0 0 1 0	dst	src	1 1 1 1 1 1 1 1	SFR	IMb	4	8
0 0 1 1 0 0 1 0	dst	src		SFR	DA	4	10
0 1 0 0 0 0 1 0	src	dst		DA	SFR	4	10
0 1 0 1 0 0 1 0	1 1 1 1 1 1 1 1	src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.

With this instruction the unit of data of the operand is byte. If a word length register (including general register) is written as an operand, only the lower byte (even address) of the operand is to be executed.The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst – src

Subtract the source byte from the destination byte and store the result into the destination.

Flags

C: Set if the operation results in a borrow at MSB. Otherwise, cleared.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Set if the operation results in an overflow. Otherwise, cleared.

SWAP

(Swap Bytes)

Instruction SWAP dst

Format

0 1 1 1 0 0 1 0	dst

dst	src	byte	cycle
SFR		2	8

Operation TEMP ← dst (H)

dst (H) ← dst (L)

dst (L) ← TEMP

Exchange the content of the upper byte of the destination word with that of lower byte.

Flags

C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

TRAP

(Software Trap)

Instruction Trap

Format

	dst	src	byte	cycle
1 1 1 1 0 1 0 1 | 1 1 1 1 1 1 1 1			2	19

Operation SP ← SP – 2

(SP) ← PC

SP ← SP – 2

(SP) ← SEG

SP ← SP–2

(SP) ← PSW

Generate a software interrupt.

Flags

C: Left unchanged.

Z: Left unchanged.

N: Left unchanged.

V: Left unchanged.

I: Cleared.

TRAPV

(Trap Overflow)

Instruction TRAPV

Format

		dst	src	byte	cycle
1 1 1 1 0 1 0 0	1 1 1 1 1 1 1 1			2	3/19

Branch on cycles: yes/no

Operation

```
if (V == 1) then
        PSW.V ← 0
        SP ← SP–2
        (SP) ← PC
        SP ← SP–2
        (SP) ← SEG
        SP ← SP–2
        (SP) ← PSW
else
        PC ← PC + 2
endif
```

If V flag (overflow flag) of PSW register is set, software interrupt will be generated.

Flags

C: Left unchanged.

Z: Left unchanged.

N: Left unchanged.

V: Cleared only if previously set.

I: Cleared

UNLINK

(Unlink Stack Frame)

Instruction `UNLINK`

Format

1 1 1 1 0 0 1 1	1 1 1 1 1 1 1 1

dst	src	byte	cycle
		2	6

Operation SP ← FP

FP ← (SP)

SP ← SP + 2

Store the content of frame pointer FP onto the stack pointer SP to restore the content of stack to FP.

UNLINK instruction delete the stack frame produced by LINK instruction.

Flags All flags left unchanged.

XOR

(Logical Exclusive Or Words)

Instruction XOR dst, src

Format

					dst	src	byte	cycle
0 0 0 0 1 1 1 0	dst	src			R	R	2	2
0 0 0 1 1 1 1 0	dst	IM4			R	IM4	2	2
0 0 1 0 1 1 1 0	dst		src		SFR	IM	4	8
0 0 1 1 1 1 1 0	dst		src		SFR	DA	4	10
0 1 0 0 1 1 1 0	src		dst		DA	SFR	4	10
0 1 0 1 1 1 1 0	1 1 1 1 1 1 1 1		src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.

The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst ^ src

Perform exclusive OR between the destination word and the source word, bit by bit, and store the result into the destination.

Flags

C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

XORB

(Logical Exclusive Or Bytes)

Instruction XORB dst, src

Format

					dst	src	byte	cycle
0 0 0 0 0 1 1 0	dst	src			Rb	Rb	2	2
0 0 0 1 0 1 1 0	dst	IM4			Rb	IM4	2	2
0 0 1 0 0 1 1 0	dst		src	1 1 1 1 1 1 1 1	SFR	IMb	4	8
0 0 1 1 0 1 1 0	dst		src		SFR	DA	4	10
0 1 0 0 0 1 1 0	src		dst		DA	SFR	4	10
0 1 0 1 0 1 1 0	1 1 1 1 1 1 1 1		src	dst	SFR	SFR	4	10

General purpose registers R0-R15, when used as operand, act as R addressing or SFR addressing depending on the type of associated instruction.
With this instruction the unit of data of the operand is byte. If a word length register (including general register) is written as an operand, only the lower byte (even address) of the operand is to be executed.The value of IM and DA is placed in the order of lower byte and upper byte.

Operation dst ← dst ^ src

Perform exclusive OR between the destination byte and the source byte, bit by bit, and store the result into the destination.

Flags

C: Left unchanged.

Z: Set if the operation results in 0. Otherwise, cleared.

N: Set if the operation results in 1 at MSB. Otherwise, cleared.

V: Always cleared.

11.2 Instruction usage considerations

- Byte access instructions

Byte access instructions such as ADCB, ANDB and MOVB access the data in units of byte. Some of these instructions can use SFR addressing which is normally used for word data. With SFR addressing, only the lower byte (even address) of the specified word data can be accessed. The upper byte (odd address) of the data will not be affected by these instructions.

- Number of instructions

In addition to 62 basic instructions referred to in **Chapter 1 General**, there are 10 instructions including byte access and NOP which are also described in Chapter 11.

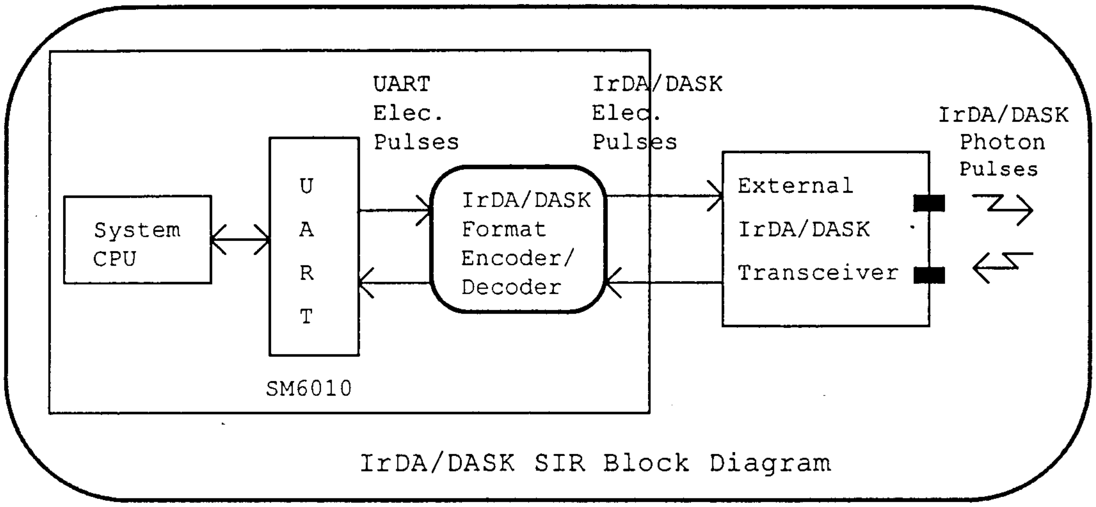
Chapter 9

Other Functions

9

Serial InfraRed Interface

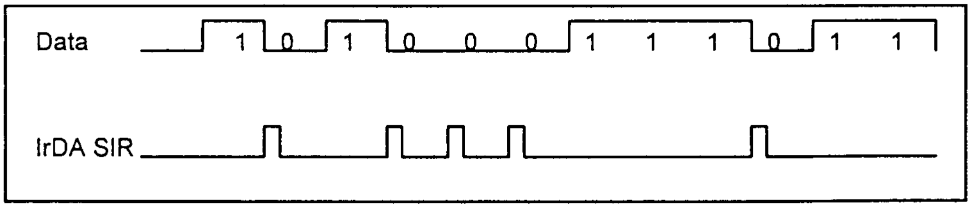
SM6010 has a Serial InfraRed (SIR) interface, that is IrDA & SHARP-DASK compatible.The interface connects to UART and performs Format Encoding/Decoding between UART format and IrDA/DASK SIR format. The SIR interfaces directly to any external IrDA /DASK Transceiver module.



IrDA/DASK SIR Block Diagram

SIR Definition

IrDA SIR uses InfraRed photon pulses to send and receive information serially at Baud rates from 2.4Kbps to 115.2Kbps in a similar fashion to a standard UART. A logical "0" is indicated by an InfraRed pulse and a logical "1" is indicated by no InfraRed pulse.



SIR Applications

The SIR interface is an important feature that allows wireless communications and data exchange between handheld devices, office equipment, and office network. It is an ideal solution for compact, low-power, cost sensitive applications.

Features

- IrDA SIR (version 1.0) compatible
- SHARP DASK SIR compatible
- Adds IR port to UART
- 2.4Kbps to 115.2Kbps IrDA data rate
- 2.4Kbps to 57.6Kbps DASK data rate
- Sleep Mode to save power
- Three Modes of operations:
    - PassThru: 3 Serial ports
    - IrDA SIR: 2 Serial ports + 1 InfraRed port
    - DASK SIR: 2 Serial ports + 1 InfraRed port
- UART format <------> SIR format

IrDA SIR Specification

The following Specification is a subset of the IrDA Serial InfraRed (SIR) Physical Layer Link Specification, Version 1.0, April 27 1994.

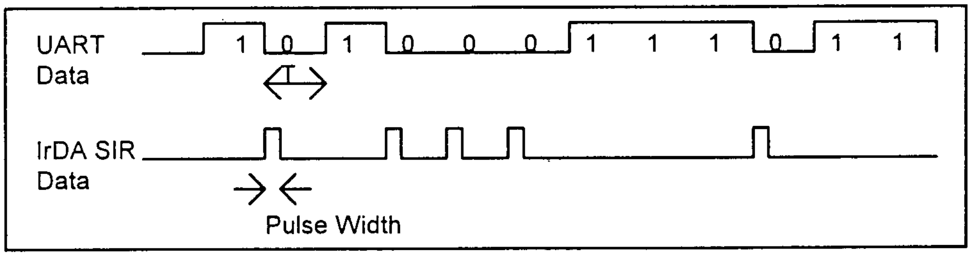
Bit Rate & Pulse Width:

Bit Rate: Bit Rate is the number of transferred bits per second, e.g. 9600 bps. The bit rate has a tolerance of 0.87% as shown in the table below. A 2400 Bit Rate will have a valid range of [2379.12 : 2420.88].

Pulse Width: Time duration of the pulse measured between 50% of the rising edge and 50% of the falling edge. The pulse width has a tolerance of 2.5% of the period.

The minimum pulse width is (3/16)*T-2.5%*T (see note below)
The maximum pulse width is: (3/16)*T+ max [2.5%*T,1.08μs]

Note: The minimum pulse width for all rates is the same, and it is that of the 115.2Kbps rate as shown in the table below..



Bit Rate Kbits/sec	Period (T) (1/Bit Rate)μs	Bit Rate Tolerance % of Bit Rate	Pulse Width Minimum, μs (1)*	Pulse Width (μs) Nominal ((3/16)*T) (2)*	Pulse Width Maximum, μs (3)*
2.4	416.67	+/- 0.87	1.41	78.13	88.55
9.6	104.17	+/- 0.87	1.41	19.53	22.13
19.2	52.08	+/- 0.87	1.41	9.77	11.07
38.4	26.04	+/- 0.87	1.41	4.88	5.96
57.6	17.36	+/- 0.87	1.41	3.26	4.34
115.2	8.68	+/- 0.87	1.41	1.63	2.71

(1) Minimum Pulse Width = Minimum Pulse Width for 115.2Kbps=(3/16)*T-2.5%*T
=1.63-0.22

(2) Nominal Pulse Width = (3/16)*T μs

(3) Maximum Pulse Width = Nominal + max [2.5%*T,1.08 μs]

A pulse is detected if its width meets the above specification for a given bit rate transfer:
Minimum <= Pulse Width <= Maximum

A transmitted pulse should have its width meet the above specification for a given bit rate transfer: Minimum <= Pulse Width <= Maximum

Bit Error Ratio (BER)*

BER should be no greater than 10^{-9}, that means 1 bit error every 10^{9} transfers.

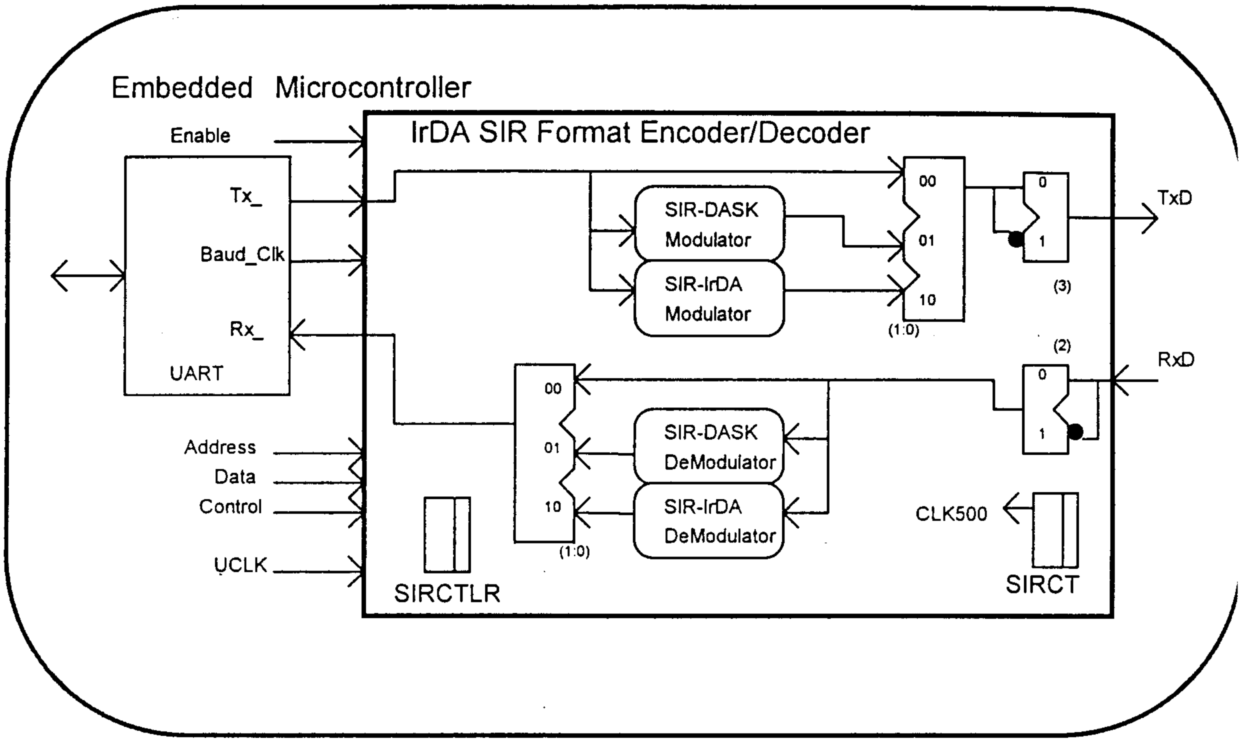
Rise Time (T_r)*

The time interval for the pulse to rise from 10% to 90% of its 100% value. It should be no greater than 0.6 µs.

Fall Time (T_f)*

The time interval for the pulse to fall from 90% to 10% of its 100% value. It should be no greater than 0.6 µs.

Functional Block Diagram



In DASK Mode Bit, SCK0 must be "00" in P1M.

Register Map

The address, SFR, initial value, access and number of bits for each register is as follows:

SIR Register	Address	SFR	Reset Value	Access	Number of Bits
SIRCTL	00FF40H	A0H	0x00	**W**	8
SIRTM	00FF42H	A1H	0x00	**W**	8

Register Description

- SIR Control Register (SIRCTL)

7	6	5	4	3	2	1	0
//////				TxP	RxP	MODE	

The SIR has an 8-bit control register to control its functionality as follows:

Bit Position	Name	Description
7:4	Reserved	Reserved
3	TxP	Tx Polarity: 0 ---> TxD output will not be inverted before it is sent out 1 ---> TxD output will be inverted before it is sent out
2	RxP	Rx Polarity: 0 ---> RxD input will not be inverted before processing it 1 ---> RxD input will be inverted before processing it
1:0	Mode	Operation Mode 00 ---> UART Mode. SIR Modulation/Demodulation are bypassed. 01 ---> DASK-SIR Mode. DASK Modulation/Demodulation are performed. 10 ---> IrDA-SIR Mode. IrDA Modulation/Demodulation are performed. 11 ---> Illegal

- SIR Timer Register (SIRTM)

7	6	5	4	3	2	1	0
ENB	TV(TIMER VALUE)						

The SIR has an 7-bit timer register to generate 500Khz clock(CLK500). The external UART clock will be divided by TV to genarate CLK500. CLK500 is enabled by setting ENB high.

SIR-IrDA Clock

The SIR-IrDA input clock is the same as the Baud Rate generator output clock for UART. It runs 16x Baud Rate. This clock will be used to sample data 16 times per period. For example, at 9.6Kbps, the period is 104.17μs and the SIR clock will be 16x9.6Kbps.

SIR-DASK Clock

The SIR-DASK requires a 500KHz IR carrier wave with 50% duty cycle. SIRTM should be programmed to generate the 500KHz clock. The output of the internal Counter, CLK500, is connected internally to the SIR-DASK Modulator module.

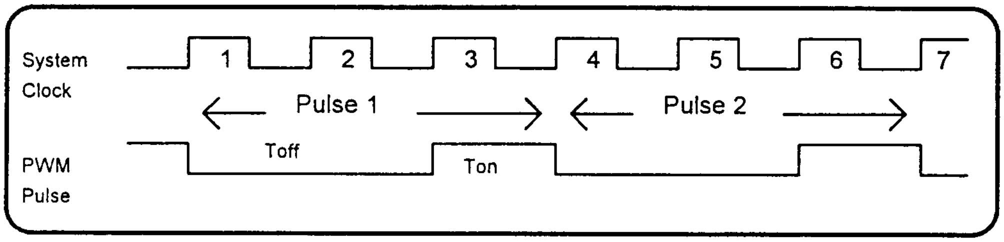
The SIR-DASK also requires a 14.318MHz for its Demodulator module. This clock comes in from an external source that should be connected to SCK input pin. SCK0 must be "00" in P1M. The input pin is internally connected to the SIR-DASK Demodulator module.

Pulse Width Modulator

SM6010 supports 1 programmable Pulse Width Modulator channels (PWM).

PWM Definition

Pulse Width Modulation is simply a method of communicating information to a device. It can be viewed as an analog signal provided in digital form. The following figure shows an example of a PWM output signal derived from the system clock.



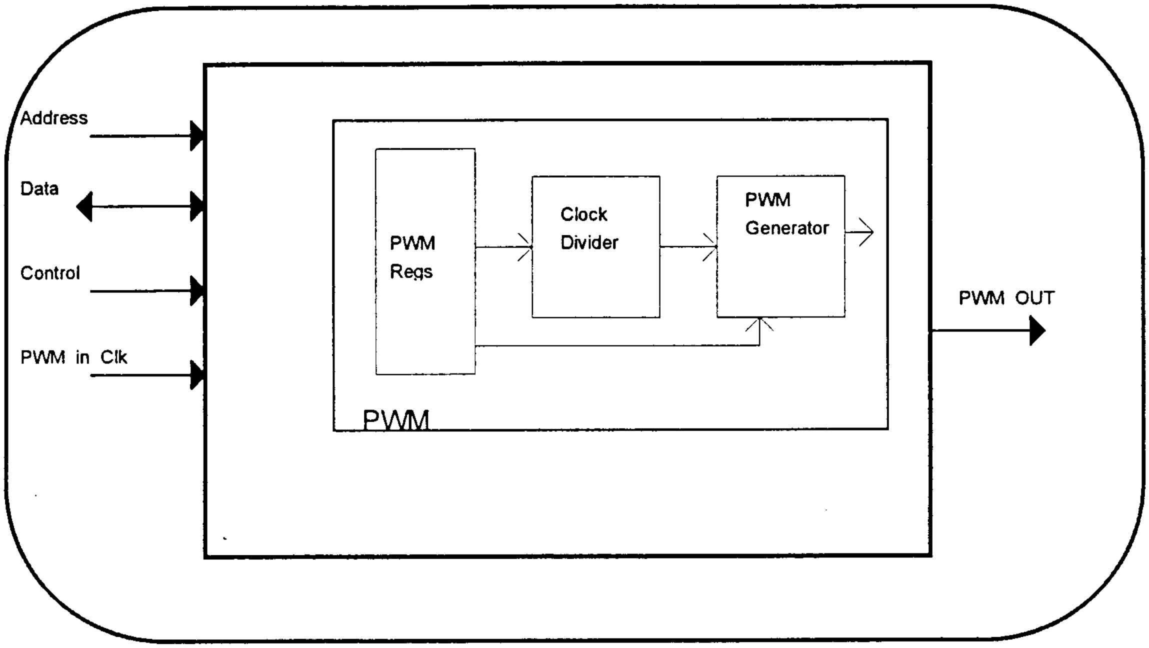
The PWM period and the duty cycle are programmable. The duty cycle is expressed as the duration of T_{on} over the sum of T_{on} and T_{off}. A signal has a constant duty cycle if T_{on} and T_{off} are uniform. If T_{on} is equal to T_{off}, the signal has a 50% duty cycle.

$$\text{Duty Cycle} = T_{on} / (T_{on} + T_{off})$$

Features

- 8-bit Resolution
- Programmable Pulse Width (Duty Cycle), Interval (Frequency) and Polarity
    - Static Programming: PWM is Stopped
    - Dynamic Programming: PWM is Running
        - Double Buffering allows dynamic programming
        - Updates to Duty Cycle, Frequency, and Polarity are done at end of a PWM cycle
- Start/Stop PWM
    - Stop PWM output at end of a PWM cycle
- Sleep Mode to save power

Block Diagram



Register Map

The address, SFR, initial value, access and number of bits for each register is as follows:

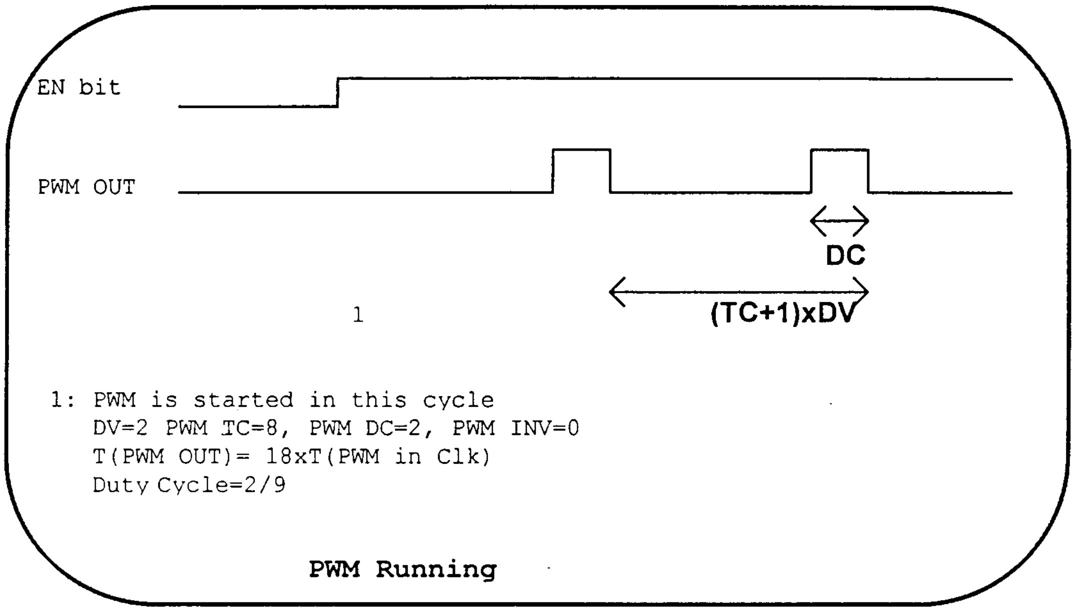
PWM Register	Address	SFR	Reset Value	Access	Number of Bits
PWM_TC	00FF66H	B3H	0x00	Read/Write	8
PWM_DC	00FF68H	B4H	0x00	Read/Write	8
PWM_CTL	00FF64H	B2H	0x00	Read/Write	8

PWM_TC and PWM_DC have ILLEGAL values coming out of reset

General Operation

PWM will be enabled and started via writing a "1" to EN bit in register PWM_CTL. PWM output is valid 1 cycles after EN is set to "1". A PWM can be stopped by writing a "0" to EN. It will stop at the end of the current PWM cycle.

Example: PWM programmed in Normal Mode



```
1: PWM is started in this cycle
   DV=2 PWM TC=8, PWM DC=2, PWM INV=0
   T(PWM OUT)= 18xT(PWM in Clk)
   Duty Cycle=2/9
```

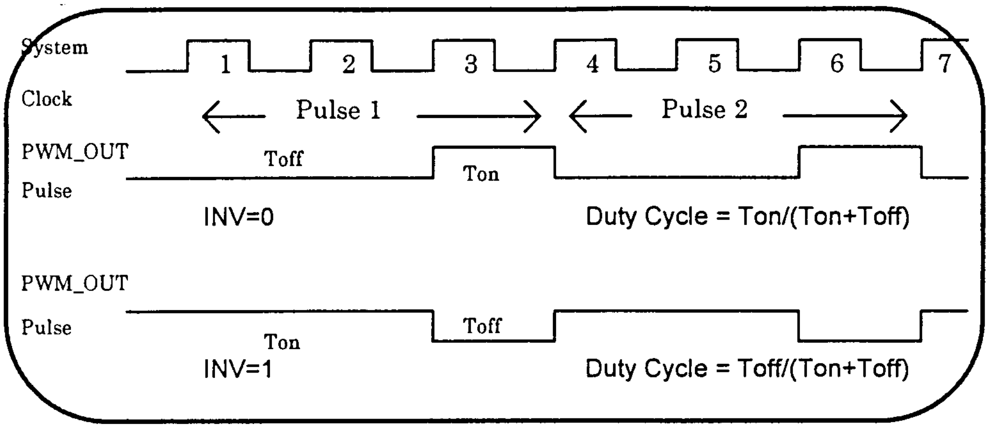
Register Description

- PWM_CTL

7	6	5	0
ENB	INV	DV	

Bit Position	Name	Description
7	ENB	Enable PWM. 0: Disable/Stop PWM. A running PWM is actually stopped when it reaches the end of its cycle. PWMx output will be: 0 ---> If PWM_INV[0] = 0 1 ---> if PWM_INV[0] = 1 1: PWM is Enabled. If PWM is in Normal Mode, it will start running on the next cycle.
6	INV	Invert PWM Output: 0: Output is not inverted. PWM_OUT will output Toff first then Ton. PWM_DC controls Ton 1: Output is inverted. PWM_OUT will output Ton first then Toff. PWM_DC controls Toff
5:0	DV	Divide Value This is used to divide the PWM input clock, PWM_in_Clk, by any Even value in the range [2:62], bits 0 is always treated as 0: **DV** **PWM_CLK** 00000X ----> Illegal 00001X ----> PWM_in_Clk/2 00010X ----> PWM_in_Clk/4 00011X ----> PWM_in_Clk/6 00100X ----> PWM_in_Clk/8 : 00000X ----> PWM_in_Clk/62 The new divided clock, PWM_CLK, will drive the logic to produce a PWM output.

DV is used in conjunction with PWM_TC to adjust the out frequency of PWM. **DV** is double buffered to allow it to be programmed statically (PWM is stopped) or dynamically (PWM is running). Programmed dynamically, **DV** is updated at the end of a PWM Cycle thus causing no output glitches or errors.


System
Clock
1
2
3
4
5
6
7
Pulse 1
Pulse 2
PWM_OUT
Pulse
Toff
Ton
INV=0
Duty Cycle = Ton/(Ton+Toff)
PWM_OUT
Pulse
Ton
Toff
INV=1
Duty Cycle = Toff/(Ton+Toff)

PWM_TC

PWM_TC is an 8-bit Terminal Count. It is used in conjunction with **DV** to adjust the output frequency of PWM. PWM_TC gives PWM an 8-bit resolution. PWM_TC is double buffered to allow it to be programmed statically (PWM is stopped) or dynamically (PWM is running). Programmed dynamically, PWM_TC is updated at the end of a PWM Cycle thus causing no output glitches or errors.

PWM_DC

PWM_DC is an 8-bit Duty Cycle. It is used in conjunction with PWM_TC to adjust the output duty cycle of PWM. PWM_DC gives PWM an 8-bit resolution. PWM_DC is double buffered to allow it to be programmed statically (PWM is stopped) or dynamically (PWM is running). Programmed dynamically, PWM_DC is updated at the end of a PWM Cycle thus causing no output glitches or errors.

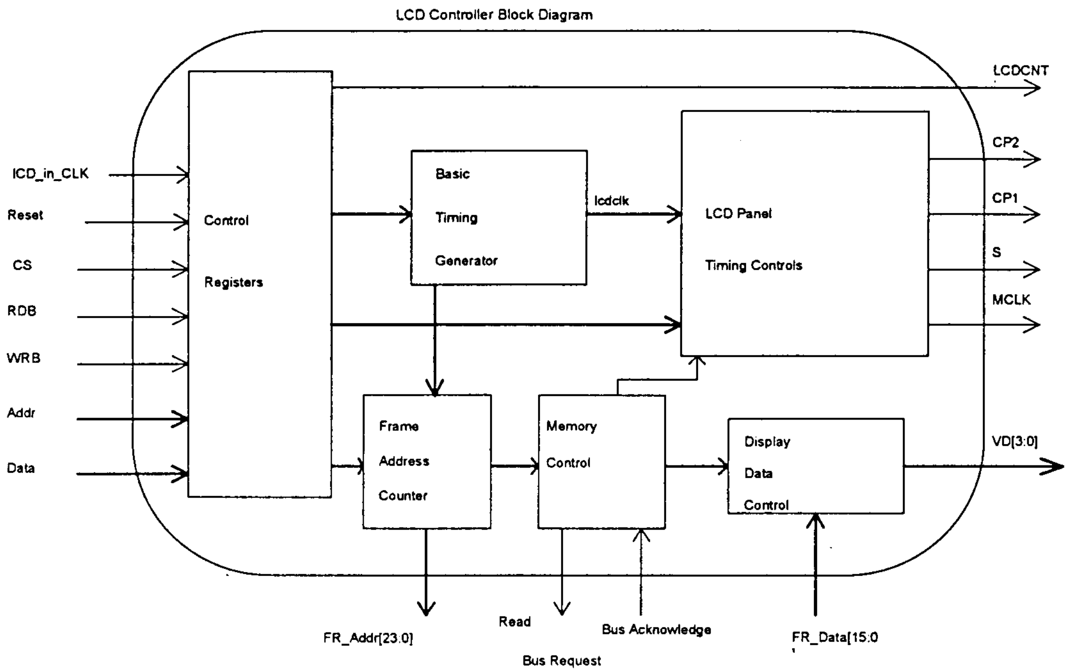
LCD Controller

The LCD Controller provides control and pixel data for directly driving LCD displays. The video frame buffer resides in the system memory to reduce external memory cost. The LCD Controller supports three primary modes, Binary mode (on,off), and 4-level Gray mode (on,off, or two gray shades) and 16-level Gray mode.

Features

- Frame Buffer Resides in Main Memory
- LCD Display Modes:
  - Binary Mode: Pixels are ON or OFF (1 bit/pixel)
  - 4-level Gray Mode: Pixels are ON, OFF, Gray Shade 1 or Gray Shade 2 (2 bits/pixel)
  - 16-level Gray Mode: Gray Shade (4 bits/pixel)
- LCD Panel:
  - Single Scan (4-bit and 2-bit and 1-bit data transfer)
- Maximum Resolution:
  - Horizontal: 1024 pixels in Binary Mode, 512 pixels in 4-level Gray, 256 pixels in 16-level Gray Mode
  - Vertical: 256 lines

Block Diagram



LCD External Interface

Name	Type	Description
VD[3:0]	Output	Video Data
CP2	Output	Shift/Pixel Clock
CP1	Output	Line Pulse/HSYNC
S	Output	Frame Pulse/VSYNC
MCLK	Output	AC Modulation Signal
LCDCNTL	Output	LCD control signal

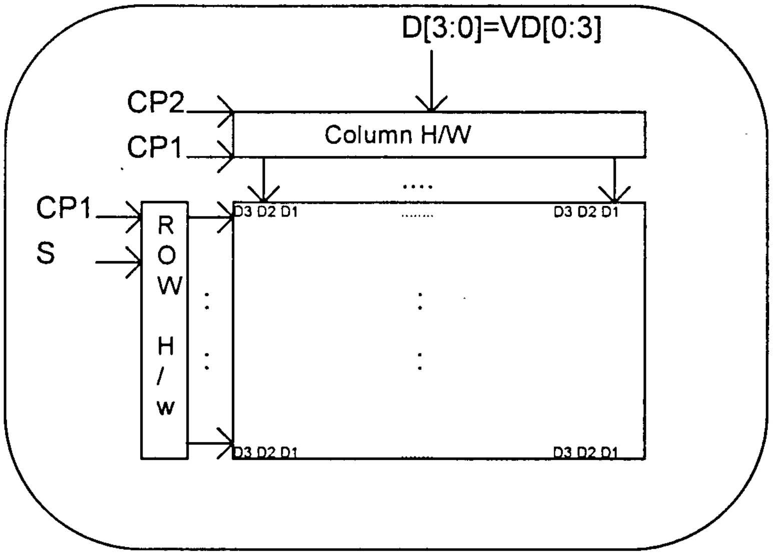
LCD Signals Description

- VD[3:0]

These are the LCD display data. These data outputs relate to a typical LCD panel Data input as follows:

LCD Controller	**Transfer Size**		
	4-bit transfer	2-bit transfer	1-bit transfer
VD[0]	D3	D3	D3
VD[1]	D2	D2	-------
VD[3:2]	D0:D1	--------	--------

D3 is the left most bit displayed on the panel



- CP2

CP2 is the Shift/Transfer clock for the display data VD[3:0]. The contents of VD[3:0], are shifted /transferred into the Column hardware on the falling edge of CP2.

- CP1

CP1 is the Line Pulse/Horizontal Sync signal. This signal will be activated when all the display data associated with a line are transferred to the LCD Panel column hardware. The falling edge of CP1 will signal the LCD Panel to display the transferred line.

- S

This is the Frame Pulse/Vertical Sync signal. It signals to the LCD panel the start of a new frame (a new screen update). It is active High and the LCD panel samples it on the falling edge of CP1.

- MCLK

This signal provides AC Modulation for the LCD drivers to ensure no DC build up which will cause chemical breakdown of the LCD fluid and ruin the panel. MCLK is a square wave signal that will allow the LCD panel to be driven for equal periods of time at positive and negative polarity. The frequency of the MCLK is programmable via LCD_MCLKW register.

Today's panels generally will toggle this signal in the range of 13 to 30 line times (CP1). Most SHARP LCD panels will generate this signal internally based on CP1 and do not need an external MCLK.

- LCDCNTL

This signal reflects the value of the LCDC bit in LCD_MODE register (bit 1) as follows:

LCD_MODE(1)=0 -----> LCDCNTL=0
LCD_MODE(1)=1 -----> LCDCNTL=1

This is a general purpose control signal. Some of the possible usage of this signal would be:

(1) Power ON/OFF sequence of LCD panels
(2) Turn on/off the Backlight of LCD panels to save power

Register Map

The address, SFR, access and reset value are as follows:

Register Name	Address	SFR	Access	Reset Value
LCD_MODE[7:0]	00FFA0H	D0H	R/W	0x00
LCD_BC[6:0]	00FFA2H	D1H	R/W	0x00
LCD_CP1W[6:0]	00FFA4H	D2H	R/W	0x00
LCD_DUTY[7:0]	00FFA6H	D3H	R/W	0x00
LCD_SADRL	00FFA8H	D4H	R/W	0x00
LCD_SADRM	00FFAAH	D5H	R/W	0x00
LCD_SADRH	00FFACH	D6H	R/W	0x00
LCD_VDLT[6:0]	00FFAEH	D7H	R/W	0x00
LCD_GRAY1[7:0]	00FFB0H	D8H	R/W	0x00
LCD_GRAY2[7:0]	00FFB2H	D9H	R/W	0x00
LCD_CLKDIV[7:0]	00FFB4H	DAH	R/W	0x00
LCD_MCLKW [7:0]	00FFB6H	DBH	R/W	0x00
LCD_DMA [1:0]	00FFB8H	DCH	R/W	0x00

Register Description

- MODE Register (LCD_MODE)

7	6	5	4	3	2	1	0
DISP	REV	GS	GS	XFER	XFER	LCDC	LCDA

LCD Register LCD_MODE controls the operational modes of the LCD controller. The individual bits are defined as follows:

Bit 7: DISP

DISP turns the display ON or OFF, according to the following table:

DISP	Mode	VD[3:0]
0	Display OFF	0 (background value)
1	Display ON	1 (foreground value)

The display is turned off by forcing the display data (VD[3:0]) to the background value, VD[3:0]=0. In Reverse display, REV=1, the background value becomes "1".

Bit 6: REV

REV selects between Normal and Reverse display modes, according to the following table:

REV	Mode	Background Value	Foreground Value
0	Normal Display	0	1
1	Reverse Display	1	0

In Normal display mode, "0" is the background data and "1" is the foreground data. This is reversed by setting the REV bit.

Bit [5:4] - GS(GRAY SCALE)

The Gray bits selects among binary (two shades) and 2 types of gray (4 shades or 16 shades) display modes, according to the following table:

GS[5:4]	Mode
00	Binary Display Mode: One bit of data in the frame buffer is used to drive each pixel on the display. There are two shades (ON/OFF).
01	4-level Gray Display Mode: Two bits of data in the frame buffer are used to drive each pixel on the display. There are 4 shades of gray (ON, OFF, GRAY1, GRAY2). GRAY1 and GRAY2 are programable. Refer to Gray Shade Registers.
10	16-level Gray Display Mode: Four bits of data in the frame buffer are used to drive each pixel on the display. There are 16 shades of gray with every 16 frames.
11	4-level Gray Display Mode: Two bits of data in the frame buffer are used to drive each pixel on the display. There are 4 shades of gray (ON, OFF, GRAY1, GRAY2). GRAY1 and GRAY2 are fixed but suitable to get less flicker.

Bit [3:2] - XSIZE

The XSIZE selects one of 1-bit or 2-bit or 4-bit data transfers (per CP2 clock) to the LCD display, according to the following table:

XSIZE	Data Transfer per CP2 Clock
00	1-bit
01	2-bit
10	4-bit
11	Not Used

Bit 1 - LCDC

The LCDC bit controls the state of the LCDCNTL output signal. The LCDCNTL signal is a general purpose control signal as defined earlier. The function of the LCDC bit is described in the following table:

LCDC	Mode
0	LCDCNTL = 0
1	LCDCNTL = 1

Bit 0 - LCDA

The LCDA bit controls the activation of the LCD controller. Upon Reset, this bit is set to "0" disabling the LCD controller internal state machines and thus the controller's output signals (CP1, CP2, S, VD, MCLK, LCDCNTL). LCD_MODE register should be the last LCD register to be programmed with LCDA bit set to "1". This will activate the LCD controller and thus the controller's output signals (CP1, CP2, S, VD, MCLK, LCDCNTL). Once the LCD controller is running, it can be reprogrammed with LCDA=1.

LCDA	LCD Controller
0	Not Active
1	Active

The user should pay a close look at the power on/off sequence of the LCD panel to avoid ruining the LCD panel. Upon reset, and if the LCD panel already have Vdd turned on, the LCD controller should be programmed and activated as soon as possible.

Line Display Byte Count Register (LCD_BC)

6	5	4	3	2	1	0
<------------------------- LCD_BC ------------------------------->						

LCD_BC register controls the number of display bytes per line in the frame buffer. In binary display mode (LCD_MODE(3)=0), a byte in the frame buffer reflects 8 LCD pixels (1-bit/pixel). In the gray display mode, a byte in the frame buffer reflects 4 LCD pixels (2-bits/pixel) in 4-level and 2 LCD pixels (4 bits/pixel). So for a given LCD panel width W pixels, LCD_BC is programmed as W/8 in binary display mode or 2*W/8 in gray display mode.

Example - A 320 pixel wide display:

Binary Display Mode:
LCD_BC = (320 pixels/line)*(1 bit/pixel)*(1 Byte)/(8 bits) = 40 bytes/ line

Gray Display Mode(4-level):
LCD_BC = (320 pixels/line)*(2 bits/pixel)* 1 Byte)/(8 bits) = 80 bytes/ line

The following table shows the relationship between LCD_BC and the number of display pixels. Note that the binary value of the register is the number of bytes less one.

Relation between LCD_BC and the Number of Display Pixels

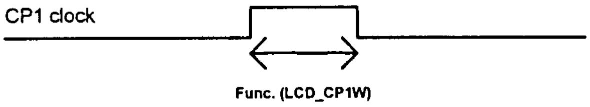
LCD_BC[6:0] (Binary)							No. of Bytes (Decimal)	Display Pixels		
6	5	4	3	2	1	0		Binary Mode	Gray Mode (4-level)	Gray Mode (16-level)
0	0	0	0	0	0	0	1*	NA	NA	NA
0	0	0	0	0	0	1	2	NA	NA	NA
0	0	0	0	0	1	0	3*	NA	NA	NA
:	:	:	:	:	:	:	:	:	:	:
0	0	0	0	1	1	0	7	NA	NA	NA
0	0	0	0	1	1	1	8	64	32	16
:	:	:	:	:	:	:	:	:	:	:
1	0	0	1	1	1	1	80	640	320	160
:	:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:	:
1	1	1	1	1	0	1	126	1008	504	252
1	1	1	1	1	1	0	127*	NA	NA	NA
1	1	1	1	1	1	1	128	1024	512	256

* LCD Panels that are not a multiple of 16 in width (8, 24, ..., 328, ... 1016).
NA: Not Available

Line Pulse Width Register (LCD_CP1W)

6	5	4	3	2	1	0
<------------------------- LCD_CP1W ------------------------->						

LCD_CP1W controls the width of the line pulse high time, CP1, as a function of CP2. This allows the frame frequency to be adjusted to drive various LCD panels. Refer to the Basic Timing Section for more detail.



LCD_CP1W (Binary)							LCD_CP1W (Decimal)
6	**5**	**4**	**3**	**2**	**1**	**0**	
0	0	0	0	0	0	0	illegal
0	0	0	0	0	0	1	illegal
0	0	0	0	0	1	0	illegal
0	0	0	0	0	1	1	6
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
1	1	1	1	1	0	1	250
1	1	1	1	1	1	0	252
1	1	1	1	1	1	1	254

Duty Cycle Register (LCD_DUTY)

7	6	5	4	3	2	1	0
<------------------- LCD_DUTY[7:0] ------------------------------->							

LCD_DUTY registers define the total number of CP1 pulses per frame which is also called the duty cycle. LCD panels typically have the duty cycle equal to the total number of vertical display lines. The LCD controller will request data from external memory for every line programmed in the LCD_DUTY register.

Relation between (LCD_DUTY) and # of CP1 Pulses per Frame

LCD_DUTY[7:0] (Binary)								**# of CP1 Pulses per Frame (Decimal)**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	2
0	0	0	0	0	0	1	0	3
:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:
0	1	1	0	0	0	1	0	100
:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:
1	1	0	0	0	1	1	1	200
:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:
1	1	1	0	1	1	1	1	240
:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:
1	1	1	1	1	1	0	1	254
1	1	1	1	1	1	1	0	255
1	1	1	1	1	1	1	1	256

Frame Buffer Start Address Registers (LCD_SADR1[23:0])

7	6	5	4	3	2	1	0
<--------------------------------------- LCD_SADRL-->							
<--------------------------------------- LCD_SADRM -----\------------------------------>							
<--------------------------------------- LCD_SADRL------\\------------------------------>							

LCD_SADR1 registers define the 24-bit start address of the frame buffer. The setting can range from 0x000000 to 0xFFFFFE. The starting address is half-word aligned, so the LSB bit must always be "0".

The frame buffer should always reside in external memory. The frame buffer should be large enough to hold the image and the extra lines in the case where the duty cycle is larger than the number of vertical lines.

Virtual Display Delta Register (LCD_VDLT)

<table>
<tr><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7"><-------------------------- LCD_VDLT --------------------------></td></tr>
</table>

LCD_VDLT register is the difference between the address of the last halfword displayed on the previous LCD line (A_{old}) and the address of the first halfword to be displayed on the new LCD line (A_{new}).

$$LCD\_VDLT = A_{new} - A_{old}$$

Total number of bytes/frame buffer line = LCD_BC+LCD_VDLT-2

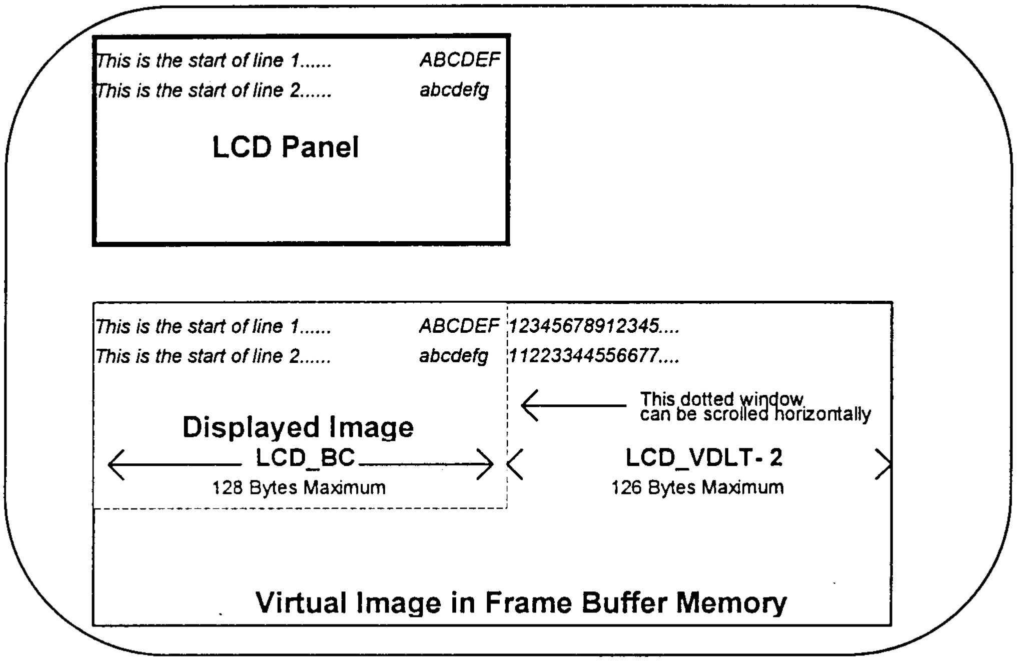
The address of the first halfword to be displayed on each new LCD line is computed by adding LCD_VLDT value to the address of the last halfword displayed on the previous LCD line. By changing the starting address of the frame buffer, a scrolling function can be implemented.

A virtual screen is when an image stored in memory is larger than the LCD panel size. LCD_VDLT allows the virtual image to be displayed. A portion of the image is actually displayed and the rest of the image can be displayed by scrolling horizontally. The width of the virtual display panel is 254 bytes (LCD_BC+LCD_VDLT-2). So the controller can support 2032 pixels in binary mode and 1016 pixels in 4-level gray mode.

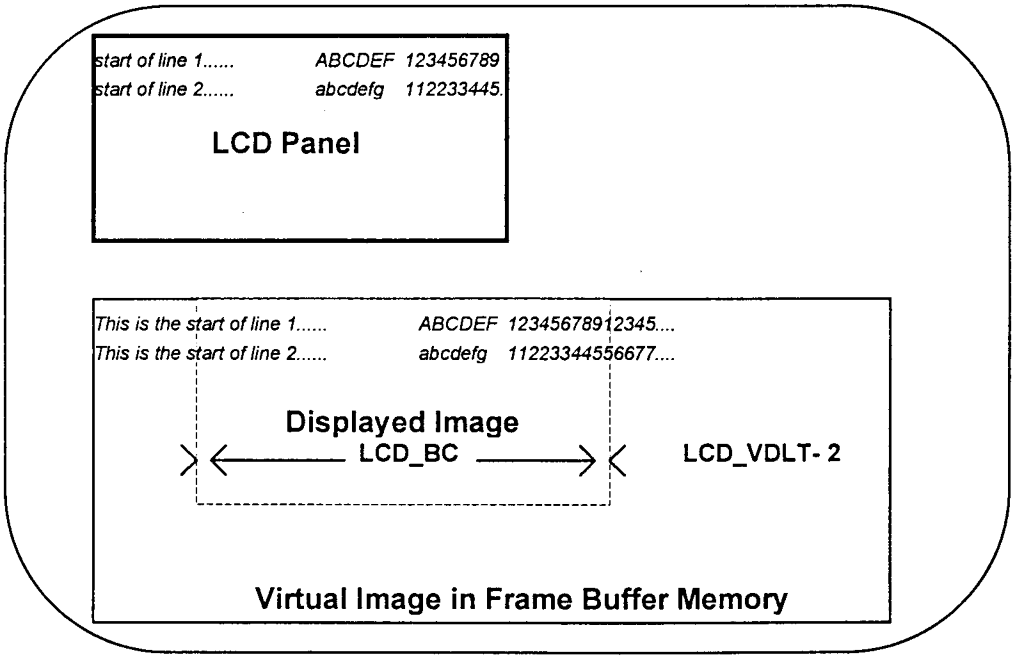
Relationship Between LCD_VDLT and Address Difference

LCD_VDLT							**Address**
6	**5**	**4**	**3**	**2**	**1**	**0**	**Difference**
0	0	0	0	0	0	0	2
0	0	0	0	0	1	0	4
0	0	0	0	1	0	0	6
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
1	1	1	1	0	1	0	124
1	1	1	1	1	0	0	126
1	1	1	1	1	1	0	128

Example 1 : Virtual Display Example (Logical View)


This is the start of line 1...... ABCDEF
This is the start of line 2...... abcdefg
LCD Panel
This is the start of line 1...... ABCDEF 12345678912345....
This is the start of line 2...... abcdefg 11223344556677....
This dotted window can be scrolled horizontally
Displayed Image
LCD_BC
128 Bytes Maximum
LCD_VDLT- 2
126 Bytes Maximum
Virtual Image in Frame Buffer Memory

After scrolling to the left,


start of line 1...... ABCDEF 123456789
start of line 2...... abcdefg 112233445.
LCD Panel
This is the start of line 1...... ABCDEF 12345678912345....
This is the start of line 2...... abcdefg 11223344556677....
Displayed Image
LCD_BC
LCD_VDLT- 2
Virtual Image in Frame Buffer Memory

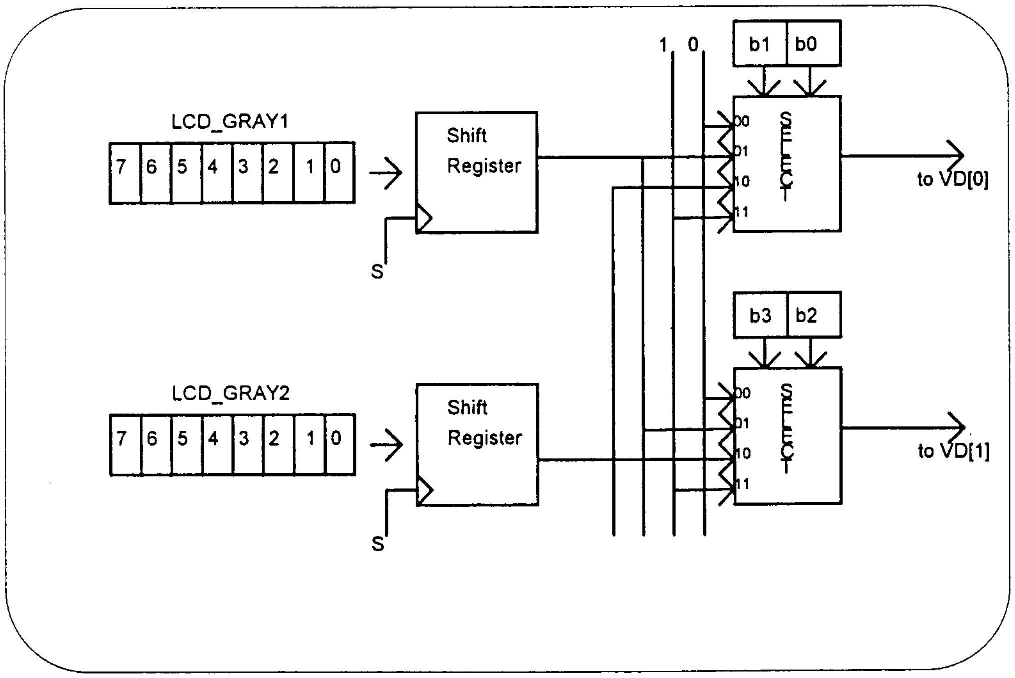
Gray Shade Registers (LCD_GRAY1,LCD_GRAY2)

7	6	5	4	3	2	1	0
<---------------------------------- LCD_GRAY1[7:0] ---------------------------------->							
<---------------------------------- LCD_GRAY2[7:0] ---------------------------------->							

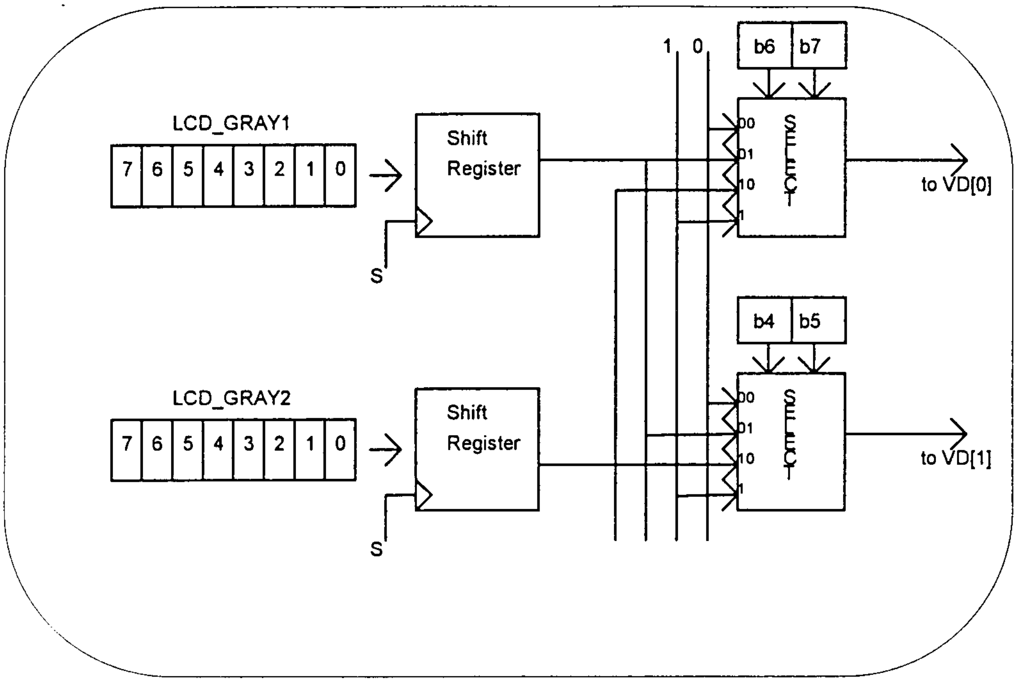
LCD_GRAY1 & LCD_GRAY2 registers control the gray shades on the LCD panel when the LCD controller is running in gray display mode(GS[5:4] is "01" in the Mode Register).

x	y	LCD Display
0	0	Display OFF
0	1	Display Gray Shade based on LCD_GRAY1
1	0	Display Gray Shade based on LCD_GRAY2
1	1	Display ON

From the above table, (x,y)=(0,1) will select LCD_GRAY1 and (x,y)=(1,0) will select LCD_GRAY2. As shown below, LCD_GRAY1 and LCD_GRAY2 are both loaded into a shift register that is clocked by the frame clock, S. The outputs of the shift registers are fed into selectors that has 4 inputs (0,1,LCD_GRAY1,LCD_GRAY2). The outputs of the selectors are connected to the output display data bits, VD[3:0]. Each pair of bits from the frame buffer will select one of four inputs to connect to VD[3:0]. Every frame clock, a new bit of LCD_GRAY1 and LCD_GRAY2 is shifted out and provided as an input to the selector.



Format #0: 4-Level Gray Shade



Format #1: 4-Level Gray Shade

In Format #1, if it is desirable to use (x,y)=(b7,b6) instead of (x,y)=(b6,b7) for gray mode, this can be easily accomplished by only swapping the content of LCD_GRAY1 and LCD_GRAY2 registers in Fromat #1.

Clock Frequency Divider (LCD_CLKDIV)

7	6	5	4	3	2	1	0
<------------------------------------- LCD_CLKDIV[7:0] ---------------------------------->							

LCD_CLKDIV register internally divides the input clock to the LCD controller, LCD_in_CLK, to produce the reference clock, lcdclk. The reference clock, lcdclk, is used to generate all the timing signals for the LCD panel (S,CP1,CP2,MCLK).

$$T_{lcdclk} = T_{LCD\_in\_CLK} * LCD\_CLKDIV$$

Relationship Between LCD_CLKDIV and Divisor Value

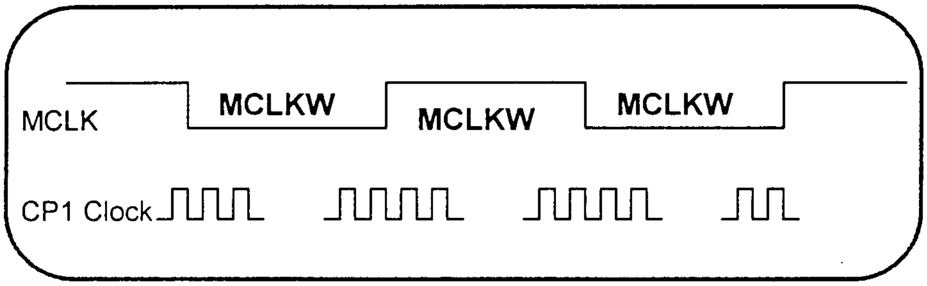
LCD_CLKDIV								Divisor
7	**6**	**5**	**4**	**3**	**2**	**1**	**0***	
0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	1	0	4
0	0	0	0	0	1	0	0	6
.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	
1	1	1	1	1	0	1	0	252
1	1	1	1	1	1	0	0	254
1	1	1	1	1	1	1	0	256

* LCD_CLKDIV(0) must always be 0

MCLK Width (LCD_MCLKW)

7	6	5	4	3	2	1	0
<———— LCD_MCLKW[7:0] ————>							

LCD_MCLKW register is used to program the MCLK width relative to CP1 clock. The MCLK converts the LCD driver voltage to an AC voltage by inverting it periodically. This will keep the LCD panel from being driven at a DC level which will cause chemical breakdown of the LCD fluid and ruin the panel (DC Buildup). Since the frequency of inversion varies from one LCD panel to another, this register allows the user to program the MCLK width.



MCLK & CP1 Relationship

Relation between LCD_MCLKW and CP1 clocks

LCD_MCLKW[7:0]								Width of MCLK in
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**CP1 Clocks**
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	2
0	0	0	0	0	0	1	0	3
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	0	1	254
1	1	1	1	1	1	1	0	255
1	1	1	1	1	1	1	1	256

DMA WAIT (LCD_DMA)

7	6	5	4	3	2	1	0
						WAIT	WAIT

LCD_DMA register is used to control wait state while getting display data from the frame buffer in the system memory. Two-clock cycle is standard. One or two more cycles can be added to access slower memory. Bus width needs to be controlled by BUS8 external pin.

wait[1]	wait[0]	Memory Access Cycle
0	0	Two-Cycle
0	1	Three-Cycle
1	0	Four-Cycle
1	1	Not Available

Bandwidth needs to be considered to get your desirable system performance.
The following example shows an estimation of bandwidth overhead to the data bus.

Screen size: 320 x 240 pixels
Bits per pixel: 2 bits
Screen refresh rate: 80 Hz
System clock: 15 MHz
Bus size: 16-bit
DMA access cycle per 16-bit data: 3 cycles(Independent from LCD_DMA register) + 2 cycles(Based on LCD_DMA register)

The period, Tl, that LCDC must update one line of the screen is,

Tl = 1/80Hz x 1/240lines
= 52.1 us

At the same time, the display data must be transferred. The duration, Tdma, that the DMA access cycle occupies the bus is,

Tdma = (320pixels x 2-bit per pixel x (3+2)cycles) / (15MHz x 16-bit bus)
= 13.3 us

Therefore, the percentage of host bus time taken up by the DMA is Pdma,

Pdma = 13.3 us / 52.1 us
= 25.53%

Real-Time Clock

The real-time clock(RTC) provides a current time of seconds, minutes and hours. The RTC operates on 32KHz subclock.

Register Map

The address, SFR, byte/word, access and reset value for each register are as follows:

Register Name	Address	SFR	Byte/ Word	Access	Reset Value
RTTS	00FF50H	A8H	Byte	R/W	00H
RTTHM	00FF52H	A9H	Word	R/W	0000H
RTTD	00FF54H	AAH	Word	R/W	0000H
RTA	00FF56H	ABH	Word	R/W	0000H
RTCTL	00FF58H	ACH	Byte	R/W	00H

Register Description

Second Register (RTTS)

7	6	5	4	3	2	1	0
-	-	SEC[5:0]					

RTTS Register sets seconds from 0 upto 59.

Hour-Minute Register (RTTHM)

15	14	13	12	11	10	9	8
-	-		HO[4:0]				

7	6	5	4	3	2	1	0
-	-		MN[4:0]				

RTTHM Register sets hours from 0 upto 23 and minutes from 0 upto 59.

Day Register (RTTD)

15	14	13	12	11	10	9	8
-	-					DY[9:8]	

7	6	5	4	3	2	1	0
DY[7:0]							

RTTD Register sets days from 0 upto 1023.

Alarm Register (RTA)

15	14	13	12	11	10	9	8
-	-		AH[4:0]				

7	6	5	4	3	2	1	0
-	-	-	AM[4:0]				

RTA Register sets time tp be alarmed. AH sets hours and AM sets minutes

Control Register (RTCTL)

7	6	5	4	3	2	1	0
EALM	EDAY	EHOU	EMIN	ESEC		RUN	ADJ

Bit 7 -EALM

The EALM bit controls the enable of the Alarm. The function of the RUN bit is described in the following table:

EALM	Mode
0	DISABLE
1	ENABLE

Bit 6 -EDAY

The EDAY bit controls the enable of the interrupt per day. The function of the EDAY bit is described in the following table:

EDAY	Mode
0	DISABLE
1	ENABLE

Bit 5 -EHOU

The EHOU bit controls the enable of the interrupt per hour. The function of the EHOU bit is described in the following table:

EHOU	Mode
0	DISABLE
1	ENABLE

Bit 4 -EMIN

The EMIN bit controls the enable of the interrupt per minute. The function of the EMIN bit is described in the following table:

EMIN	Mode
0	DISABLE
1	ENABLE

Bit 3 -ESEC

The ESEC bit controls the enable of the interrupt per second. The function of the ESEC bit is described in the following table:

ESEC	Mode
0	DISABLE
1	.ENABLE

Bit 1 -RUN

The RUN bit controls the enable of the RTC. The function of the RUN bit is described in the following table:

RUN	Mode
0	DISABLE RTC
1	ENABLE RTC

Bit 0 -ADJ

The ADJ bit controls the initilization of the internal divider into all zeros. The function of the ADJ bit is described in the following table:

ADJ	Mode
0	NORMAL RUN
1	INITIALIZE INTERNAL DIVIDER

EXTENSION DATA MEMORY

The extension data memory provides a way to expand Data Memory Space.
By setting this register one more 16M bytes can be accessed.
This register includes Sub Clock Voltage Select also.

Register Map

The address, SFR, byte/word, access and reset value for each register are as follows:

Register Name	Address	SFR	Byte/ Word	Access	Reset Value
EXTDM	00FF44H	A2H	Byte	R/W	00H

Register Description

Extension Register (EXTDM)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	SUBC	EXT

EXTDM Register sets A24 output. Before using this function P17 needs to be set as A24.

Bit 1 -SUBC Sub Clock Voltage

After reset Sub Clock buffer is not operable with RTCVcc. Therefore it's connected with Vdd first.
If RTC will be operating with RTCVcc(lower than VDD), SUBC needs to be set as '1' after reset.

SUBC	Mode
0	Vdd
1	RTCVcc

Bit 0 -EXT

EXT	Mode
0	A24=0
1	A24=1

Power Management

ThePower Management unit (PM) distributes the clock to peripherals. Peripheral can have their clocks stopped, to further reduce power.

Block Diagram

Register Map

The address, SFR, byte/word, access and reset value for each register are as follows:

Register Name	Address	SFR	Byte/ Word	Access	Reset Value
PH_CKSEL	00FF6AH	B5H	Byte	R/W	00H

Register Description

MODE Register (PH_CLKSEL)

7	6	5	4	3	2	1	0
-	-	-	-	-	PWM	LCDC	SIR

PM Register PH_CLKSEL controls the imput clocks to peripherals like SIR, LCDC, PWM. The individual bits are defined as follows:

Bit 2 - LCDC

LCDC	LCDC Clock
0	Not Active
1	Active

Bit 1 -PWM

PWM	PWM Clock
0	Not Active
1	Active

Bit 0 - SIR

SIR	SIR Clock
0	Not Active
1	Active

Manual title	SM6010 user's manual
Ref. No.	SCX031704E
Issued	Aug. 1997 Preliminary
Issued by	SHARP Corporation, Tenri IC Group
Edition by	SHARP Corporation, Tenri IC Group, Logic IC Engineering Center