



**MiDAS2.1 Family:
Flash / ISP / IAP
8-bit Turbo Microcontrollers**

Preliminary

**Rev. 1.0
March 2007**

**Copyright CORERIVER Semiconductor Co., Ltd. 2007
All Rights Reserved**

- u CORERIVER Semiconductor reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time.*
- u To discontinue any product or service, CORERIVER should inform customers of that before 3 months through its homepage.*
- u Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.*
- u The CORERIVER Semiconductor products listed in this document are intended for usage in general electronics applications. These CORERIVER Semiconductor products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury.*

Preliminary

Table of Contents

1	PRODUCT OVERVIEWS	9
1.1	PRODUCT LINE-UP OF MIDAS2.1 FAMILY	9
2	FEATURES	11
3	BLOCK DIAGRAM	13
4	PIN CONFIGURATIONS	15
5	PIN DESCRIPTION	17
6	FUNCTIONAL DESCRIPTION	19
6.1	CPU DESCRIPTION.....	19
6.1.1	<i>Memory Organization</i>	19
6.1.2	<i>Special Function Registers (SFRs) Map and Description</i>	21
6.1.3	<i>On-Chip External Data Memory Access</i>	24
6.1.4	<i>Instruction Set Summary</i>	25
6.1.5	<i>CPU Timing</i>	28
6.2	PERIPHERAL DESCRIPTION	31
6.2.1	<i>I/O Ports</i>	31
6.2.2	<i>LVD (Low Voltage Detector)</i>	43
6.2.3	<i>WDT (Watchdog Timer)</i>	47
6.2.4	<i>Timer/Counter 0/1</i>	48
6.2.5	<i>Simplified UART (Universal Asynchronous Rx/Tx)</i>	53
6.2.6	<i>PWM (Pulse Width Modulator)</i>	56
6.2.7	<i>A/D Converter (Analog to Digital Converter)</i>	61
6.2.8	<i>I²C Serial I/O</i>	66
6.2.9	<i>Interrupt</i>	76
6.2.10	<i>Reset Circuit</i>	82
6.2.11	<i>Clock Circuits</i>	83
6.2.12	<i>Power Management</i>	85
6.2.13	<i>Programming</i>	87
7	ABSOLUTE MAXIMUM RATINGS	91
8	DC CHARACTERISTICS	93
8.1	GENERAL DC CHARACTERISTICS	93

9	AC CHARACTERISTICS	95
10	ADC SPECIFICATIONS	97
11	PACKAGE DIMENSION	99
12	PRODUCT NUMBERING SYSTEM	101
13	APPENDIX A: INSTRUCTION SET	103
14	APPENDIX B: SFR DESCRIPTION	173
14.1	PORT 0 REGISTER (P0).....	173
14.2	STACK POINTER REGISTER (SP).....	173
14.3	DATA POINTER LOW REGISTER (DPL).....	173
14.4	DATA POINTER HIGH REGISTER (DPH).....	174
14.5	POWER CONTROL REGISTER (PCON).....	174
14.6	TIMER 0/1 CONTROL REGISTER (TCON)	175
14.7	TIMER MODE CONTROL REGISTER (TMOD).....	176
14.8	TIMER 0 LOW BYTE REGISTER (TL0).....	176
14.9	TIMER 1 LOW BYTE REGISTER (TL1).....	177
14.10	TIMER 0 HIGH BYTE REGISTER (TH0).....	177
14.11	TIMER 1 HIGH BYTE REGISTER (TH1)	177
14.12	PORT 1 REGISTER (P1)	177
14.13	EXTERNAL INTERRUPT FLAG REGISTER (EXIF)	178
14.14	PERIPHERAL CLOCK CONTROL REGISTER (CLKOFF).....	178
14.15	RING CONTROL CONFIGURATION REGISTER (RINGCON).....	179
14.16	LVD CONTROL REGISTER (LVDCON)	179
14.17	LVD STATUS REGISTER (LVDST)	180
14.18	SERIAL PORT CONTROL REGISTER (SCON)	180
14.19	SERIAL DATA BUFFER REGISTER (SBUF).....	181
14.20	I ² C SLAVE CONTROL REGISTER (I2C_SCON)	181
14.21	I ² C SLAVE DEVICE ADDRESS REGISTER (I2C_SDEV)	181
14.22	I ² C SLAVE MEMORY ADDRESS REGISTER (I2C_SADR).....	182
14.23	I ² C SLAVE DATA REGISTER (I2C_SDAT)	182
14.24	I ² C MASTER DATA REGISTER (I2C_MDAT).....	182
14.25	PORT 2 REGISTER (P2)	182
14.26	I ² C MASTER CONTROL REGISTER (I2C_MCON).....	183
14.27	I ² C MASTER DEVICE ADDRESS REGISTER (I2C_MDEV)	183

14.28	I ² C MASTER MEMORY ADDRESS REGISTER (I2C_MADR).....	183
14.29	I ² C MASTER MULTI-BYTE NUMBER REGISTER (I2C_MNUM)	184
14.30	I ² C MASTER CLOCK SCALE FACTOR HIGH BYTE REGISTER (I2C_MSCH).....	184
14.31	I ² C MASTER CLOCK SCALE FACTOR LOW BYTE REGISTER (I2C_MSCL).....	184
14.32	INTERRUPT ENABLE REGISTER (IE).....	184
14.33	PORT 3 REGISTER (P3)	185
14.34	INTERRUPT PRIORITY REGISTER (IP).....	186
14.35	INTERNAL RING OSCILLATOR CONTROL REGISTER (OSCICN).....	186
14.36	POWER MANAGEMENT REGISTER (PMR)	187
14.37	STATUS REGISTER (STATUS)	187
14.38	PROGRAM STATUS WORD REGISTER (PSW)	187
14.39	PORT 0 TYPE CONTROL REGISTER (P0TYPE).....	188
14.40	PORT 1 TYPE CONTROL REGISTER (P1TYPE).....	188
14.41	PORT 2 TYPE CONTROL REGISTER (P2TYPE).....	189
14.42	PORT 3 TYPE CONTROL REGISTER (P3TYPE).....	189
14.43	WATCHDOG CONTROL REGISTER (WDCON).....	189
14.44	ADC HIGH CHANNEL SELECTION LOW REGISTER (ADCHL).....	191
14.45	ADC HIGH CHANNEL SELECTION HIGH REGISTER (ADCHH).....	191
14.46	ADC HIGH CHANNEL SELECTION REGISTER (ADCHSEL).....	191
14.47	PWM CONTROL REGISTER (PWMCON).....	192
14.48	PWM0 DUTY DATA REGISTER (PWMD)	193
14.49	ACCUMULATOR (ACC/A)	193
14.50	ADC CHANNEL SELECTION HIGH REGISTER (ADCSELH).....	193
14.51	ADC CHANNEL SELECTION LOW & MUX SELECTION REGISTER (ADCSEL)	194
14.52	ALTERNATE FUNCTION SELECTION REGISTER (ALTSEL)	194
14.53	PORT 0 PULL-UP CONTROL REGISTER (P0SEL).....	195
14.54	PORT 1 PULL-UP CONTROL REGISTER (P1SEL).....	195
14.55	PORT 2 PULL-UP CONTROL REGISTER (P2SEL).....	195
14.56	PORT 3 PULL-UP CONTROL REGISTER (P3SEL).....	196
14.57	EXTERNAL INTERRUPT ENABLE REGISTER (EIE)	196
14.58	ADC RESULT VALUE HIGH REGISTER (ADCR)	196
14.59	ADC CONTROL REGISTER & ADC RESULT LOW REGISTER (ADCON)	197
14.60	B REGISTER (B).....	197
14.61	PORT 0 INPUT/OUTPUT CONTROL REGISTER (P0DIR)	197
14.62	PORT 1 INPUT/OUTPUT CONTROL REGISTER (P1DIR)	198

14.63	PORT 2 INPUT/OUTPUT CONTROL REGISTER (P2DIR)	198
14.64	PORT 3 INPUT/OUTPUT CONTROL REGISTER (P3DIR)	198
14.65	EXTENDED INTERRUPT PRIORITY REGISTER (EIP)	198
14.66	EEPROM ACCESS ENABLE (EEAEN)	199

List of Figures

Figure 3-1	Block Diagram	13
Figure 4-1	Pin Configuration	15
Figure 6-1	Memory Organization	19
Figure 6-2	Timing Comparison of the MiDAS2.1 family and Intel 80C52	29
Figure 6-3	Write Timing of MOVX instruction	30
Figure 6-4	Read Timing of MOVX instruction	30
Figure 6-5	Configuration of PORT 0	34
Figure 6-6	Configuration of P1.0 and P1.1	37
Figure 6-7	Configuration of the other P1 pins	37
Figure 6-8	Configuration of PORT2	39
Figure 6-9	Configuration of PORT3	42
Figure 6-10	Power-On Reset/Power-fail Reset and Power-fail interrupt	46
Figure 6-11	LVD Block Diagram	46
Figure 6-12	Block Diagram for Watchdog Timer	47
Figure 6-13	Timer/Counter 0 operations in Mode 0/1/2/3	51
Figure 6-14	Timer/Counter 1 operation in Mode 2	51
Figure 6-15	UART Mode 1	54
Figure 6-16	UART Mode 1 Timing	55
Figure 6-17	Functional block diagram	57
Figure 6-18	PWM timing diagram	59
Figure 6-19	A/D Converter Block Diagram	65
Figure 6-20	A/D Converter Timing Diagram	66
Figure 6-21	Data transfer including the memory address from a master transmitter to a slave receiver ..	69
Figure 6-22	Data transfer including the memory address from a slave transmitter to a master receiver ..	70
Figure 6-23	I ² C master operation	72
Figure 6-24	I ² C slave operation	74
Figure 6-24	Interrupt Vector Generation Flow	79
Figure 6-25	Hierarchy of Interrupt Priority	80
Figure 6-26	Three Reset Resources	82

Figure 6-27 Clock Generators	85
Figure 6-28 Clock Circuit.....	85
Figure 6-29 Power Management Circuit.....	86
Figure 11-1 LQFP 32-pin Package Dimension.....	99
Figure 11-2 SOIC 28-pin Package Dimension	99
Figure 12-1 Product Numbering System.....	101

List of Tables

Table 1-1 MiDAS2.1 Family - GC89C520AE Series (ISP Flash MCU).....	9
Table 5-1 Pin Description	17
Table 6-1 SFR map (* = bit addressable SFR).....	22
Table 6-2 Summary of SFRs (*: undefined value).....	22
Table 6-3 Summary of The Instruction Set.....	25
Table 6-4 Alternate Functions.....	41
Table 6-5 Read-Modify-Write Instructions.....	42
Table 6-6 Time-out values for the Watchdog timer.....	48
Table 6-7 Operation Summary of the simplified UART	55
Table 6-8 Baudrate Examples of the simplified UART.....	56
Table 6-9 PWM clock rate by PWMCON[6:4].....	58
Table 6-10 Example of conversion time vs. clock frequency	65
Table 6-11 Priority Structure of Interrupts	79
Table 6-12 Protection Modes.....	88
Table 6-13 ISP command.....	89
Table 7-1 Absolute Maximum Ratings	91
Table 8-1 General DC Characteristics	93
Table 9-1 AC Characteristics.....	95
Table 10-1 ADC Specifications.....	97
Table 13-1 Note on instruction set and addressing modes.....	103

1 Product Overviews

The MiDAS2.1 family of CORERIVER is a group of fast 80C52 compatible microcontrollers without wasted system and memory clock cycles. Its processor core was redesigned for reducing the execution time of one machine cycle to four system clock cycles. As a result, almost all of its 8052 instructions are executed about 3 times faster than that of traditional 80C52.

The MiDAS2.1 family offers two timer/counters, one serial port, maximum 30 programmable I/O ports, 28-channel 10-bit ADC (Analog to Digital Converter), one PWM (Pulse Width Modulator) output, one Watchdog timer, POR (Power-On Reset), I²C master/slave, and one LVD (Low Voltage Detector) as peripherals. In addition, it contains an internal ring oscillator, which can generate the 12 MHz system clock signal instead of a crystal oscillator.

The MiDAS2.1 family provides power reduction modes and EMI noise reduction scheme.

To help a user develop a target system, on-chip hardware debugging engine and easy-to-use training kits are also provided.

1.1 Product Line-up of MiDAS2.1 Family

Table 1-1 MiDAS2.1 Family - GC89C520AE Series (ISP Flash MCU)

Product	Mask ROM (byte)	Flash (byte)	EEPROM (byte)	RAM (byte)	Volt (V)	Freq. (MHz)	T/C [16bits]	ADC (bit x ch)	WDT	Serial I/O	PWM (bit x ch)	Package
GC89C520A0	-	7K	1K	512	2.4 ~ 5.5	20 (12)	2	10X8	1	1	8X12	32-LQFP 32-SOIC 28-SPDIP 32-SOIC
GC81C520A0	7K		1K									
GC89C511A0	-	3K	1K									
GC81C511A0	3K		1K									

2 Features

- 8-bit Turbo 80C52 Architecture
- 4 cycles/1 machine cycle
- Instruction level compatible with Intel 80C52
- 7 Kbyte Flash + 1Kbyte User EEPROM
- 512 byte on-chip RAM 256 byte normal internal RAM
 - ü 256 bytes on-chip external RAM accessed by MOVX like an external memory
- Supply voltage: 2.4V ~ 5.5V
- Operating Frequency
 - ü Max. 20MHz @ 4.5V ~ 5.5V
 - ü Max. 12MHz @ 2.7V ~ 3.3V
- Operating temperature: -40 °C ~ 125 °C
- Max. 30 Programmable I/O pins
 - ü Internal pull-up selectable bi-directional ports
 - ü Open drain or push-pull output
 - ü CMOS logic level
 - ü All ports are initialized by the power-on-reset
- EMI reduction mode: Inhibit ALE
- Low Voltage Detector (LVD)
- 16-bit Programmable Watchdog Timer (WDT)
- Two 16-bit Timer/Counters
- Simplified UART
- 1-channels of 8-bit high speed PWM outputs
- 28-channel of 10-bit Analog to Digital Converter (ADC)
 - ü Max. 100K samples per second (@ $F_{ADC} = 8 \text{ MHz}$)
 - ü Programmable prescaler for ADC clock generation
- 13 interrupt sources including 4 external ones
 - ü Timer 0/1, UART, PWM, ADC, WDT, LVD, 2 I²C and four external
 - ü Two level interrupt priority
- Reset scheme
 - ü On-chip Power-On-Reset (POR)
 - ü External reset
 - ü Low Voltage Detector (LVD) reset
 - ü Watchdog timer reset
- Power consumption

- - Active current: Max. 8mA @ 3V, 12MHz
 - Idle current : Max. 3mA @ 3V, 12MHz
 - Stop current : Max. 1uA
- Wake up from stop mode
 - External reset
 - External interrupt 0/1 (Level triggered)
 - Watchdog timer reset or interrupt
- ESD protection up to 2,000V
- Latch-up protection up to ± 200 mA
- Package : 32-LQFP (TQFP) / 28-SOIC,

3 Block Diagram

Figure 3-1 shows the block diagrams of MiDAS2.1 family. The MiDAS2.1 family fetches instructions from the internal program memory (Flash) and executes them. Data are read from or written to data memory (RAM) or special function registers (SFRs) of peripherals. The arithmetic and logic unit (ALU) inside its CPU processes data. Also several registers are used for processing or accessing data.

The MiDAS2.1 internal registers except program counter (PC) are configured as part of the on-chip RAM: therefore each register has an address. This is reasonable for the MiDAS 2.1, since it has so many registers. The bottom 32 locations of internal memory contain the register banks. The MiDAS2.1 instruction set supports 8 registers, R0 through R7, and by default (after a system reset) these registers are at addresses 00H-07H. As well as R0 to R7, there are special function registers (SFRs) at addresses from 80H to FFH. Most SFRs are accessed using direct addressing. However, the accumulator (ACC) is mostly accessed implicitly by the instruction set. The program counter (PC) has no address. So it is accessed only implicitly.

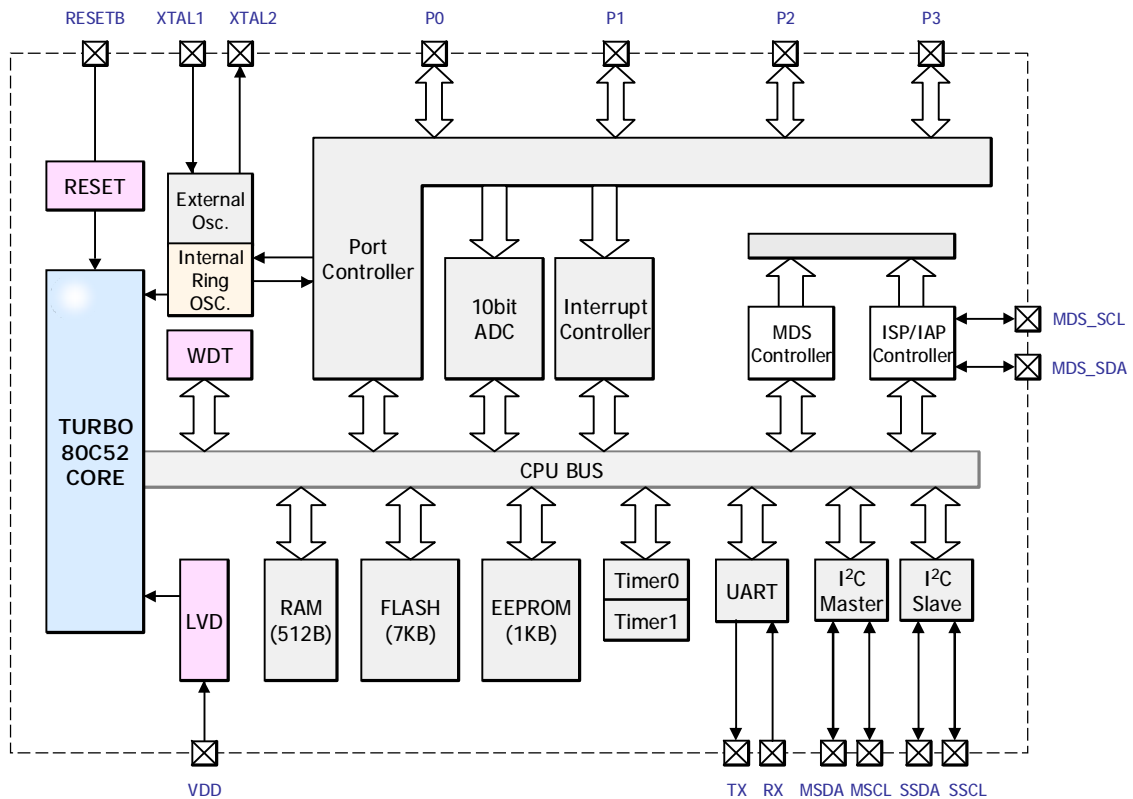


Figure 3-1 Block Diagram

4 Pin Configurations

The MiDAS2.1 family supports several packages, e.g. 32-pin LQFP(TQFP) and 28-pin SOIC. The pin configurations are shown in Figure 4-1.

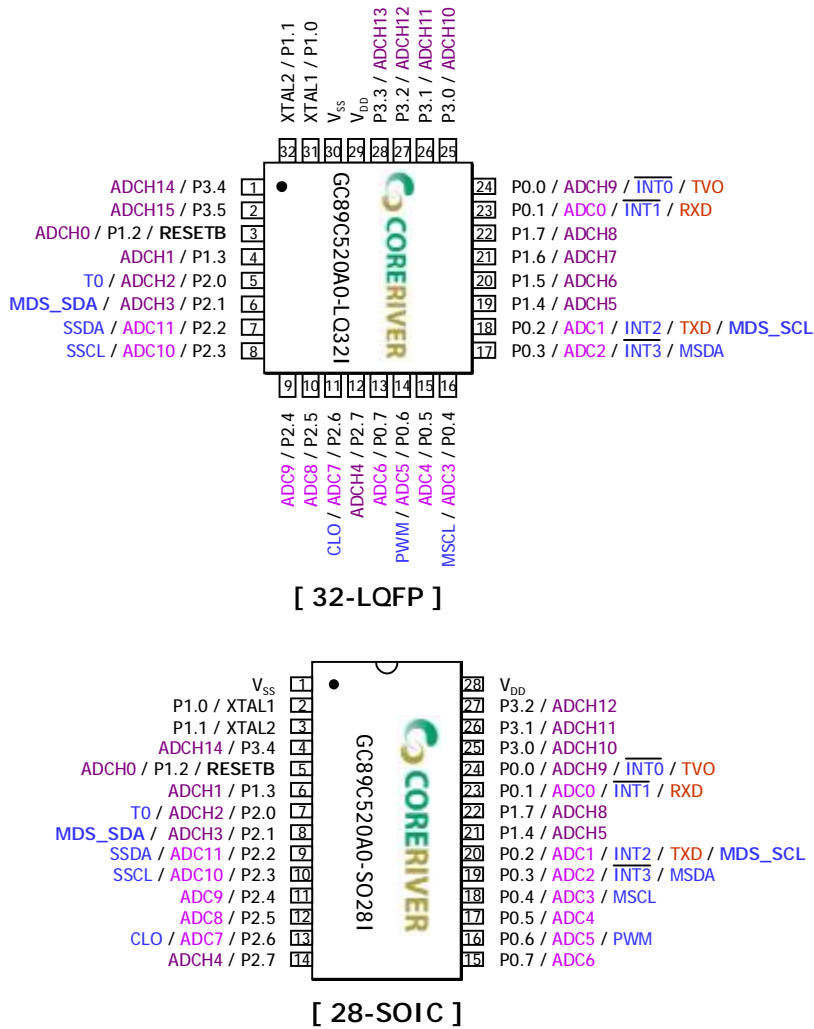


Figure 4-1 Pin Configuration

5 Pin Description

Table 5-1 Pin Description

Symbol	Direction	Description	Share Pins
VDD	Input	Supply Voltage	-
VSS	Input	Ground	-
XTAL1/P1.0	Input/ Output	Input to the inverting oscillator amplifier Programmable I/O pin - Schmitt Trigger Input - Switchable Pull-up - Push-pull Output or Open-drain Output	XTAL1/P1.0 (Crystal Input)
XTAL2/P1.1	Input/ Output	Output from the inverting oscillator amplifier, Programmable I/O pin - Schmitt Trigger Input - Switchable Pull-up - Push-pull Output or Open-drain Output	XTAL2/P1.1 (Crystal Output)
RESETB/P1.2	Input/ Output	External Reset Input Signal (Default) Programmable I/O pin - Schmitt Trigger Input - Switchable Pull-up - Push-pull Output or Open-drain Output	RESETB/P1.2/ADCH0
P1[7:3]	Input/ Output	Programmable I/O pin - Schmitt Trigger Input - Switchable Pull-up - Push-pull Output or Open-drain Output	P1.3/ADCH1, P1.4/ADCH5 P1.5/ADCH6, P1.6/ADCH7 P1.7/ADCH8
P0[7:0]	Input/ Output	Programmable I/O pin - Schmitt Trigger Input - Switchable Pull-up - Push-pull Output - Open-drain Output	P0.0 / ADCH9 / INT0 / TV0 P0.1 / ADC0 / INT1 / RXD P0.2 / ADC1 / INT2 / TXD / MDS_SCK P0.3 / ADC2 / INT3 / MSDA P0.4 / ADC3 / MSCL P0.5 / ADC4 P0.6 / ADC5 / PWM P0.7 / ADC6
P2[7:0]	Input/ Output	Programmable I/O pin - Schmitt Trigger Input - Switchable Pull-up - Push-pull Output - Open-drain Output	P2.0 / ADCH2 / T0 P2.1 / ADCH3 / MDS_SDA P2.2 / ADC11 / SSDA P2.3 / ADC10 / SSCL P2.4 / ADC9, P2.5 / ADC8 P2.6 / ADC7 / CLO P2.7 / ADCH4
P3[5:0]		Programmable I/O pin - Schmitt Trigger Input - Switchable Pull-up - Push-pull Output - Open-drain Output	P3.0 / ADCH10, P3.1 / ADCH11 P3.2 / ADCH12 P3.3 / ADCH13 P3.4 / ADCH14 P3.5 / ADCH15

6 Functional Description

6.1 CPU Description

6.1.1 Memory Organization

Most microprocessors implement a shared memory space for data and programs. This is reasonable, since programs are usually stored on a disk and loaded into RAM for execution; thus both the data and programs reside in the system RAM. Microcontrollers, on the other hand, are rarely used as the CPU in “computer systems.” Instead, they are employed as the central component in control-orient designs. There is limited memory, and there is no disk drive.

For the reason, MiDAS2.1 family has separate address spaces for program and data memory. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by an 8-bit CPU. 7K bytes of program memory can be only read, not written to.

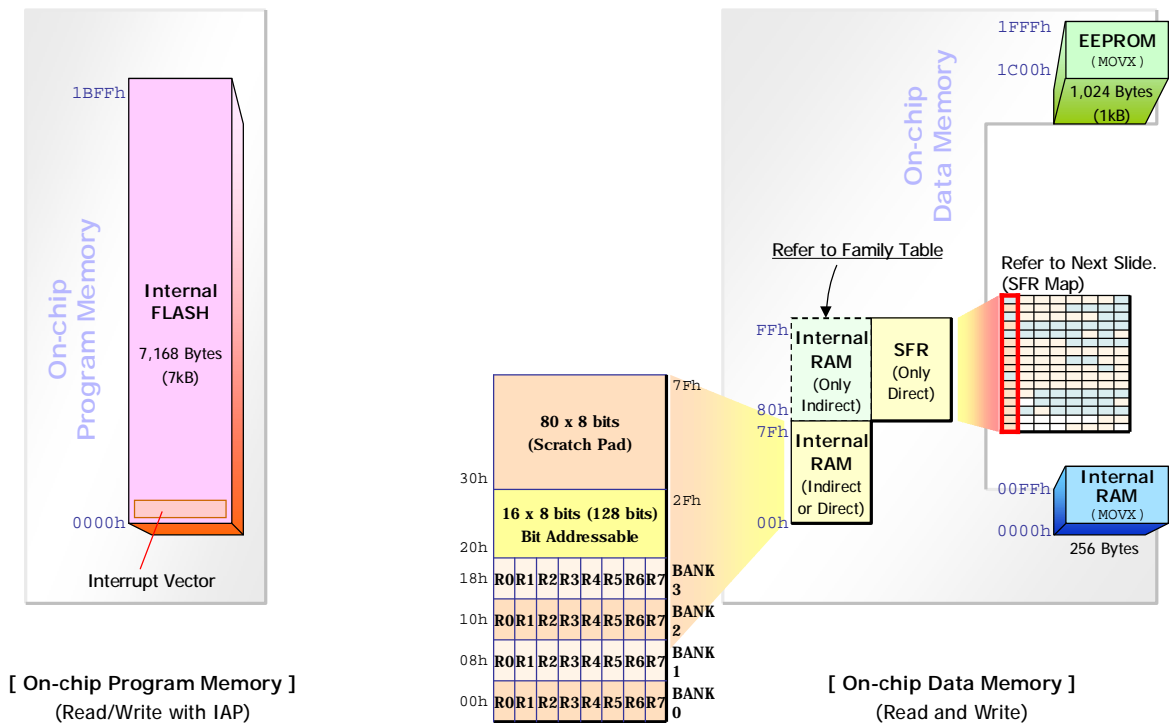


Figure 6-1 Memory Organization

6.1.1.1 Program Memory

The MiDAS2.1 family uses only the on-chip flash program memory. An application program can be transmitted to its flash program memory by the In-System Programming (ISP) method. The MiDAS2.1 family contains the 7 Kbytes of the program memory. It uses no off-chip program memory.

The left diagram of Figure 6-1 shows the map of the Program Memory. After reset, the CPU executes from location 0000H.

As shown in Figure 6-1, each interrupt has the fixed vector address in Program memory. When an interrupt is accepted, the vector address value is loaded into the program counter and the interrupt service routine starts. The vector address of External Interrupt 0, for example, is 0003H. If External Interrupt 0 is accepted, its service routine will begin at location 0003H. If the interrupt is not used, its service location is available as general purpose Program Memory.

The interrupt vector addresses are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1 and etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer than 8-byte service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are used.

6.1.1.2 Data Memory

The MiDAS2.1 family contains two kinds of on-chip data memory. One is the normal internal data memory. The other is an on-chip external data memory accessible only with the MOVX instruction. From now on, this on-chip external data memory will be called 'AUXRAM.'

The address space of the normal internal data memory is divided into three basic, physical separate and distinct blocks as shown in the right diagram of Figure 6-1:

- I The lower 128byte of internal data RAM located to the address range 00h – 7Fh.
- I The upper 128byte of internal data RAM located to the address range 80h – FFh.
- I The 128byte area for special function register (SFR) located to the address range 80h – FFh.

While the upper internal RAM area and the SFR area share the same address locations (80h – FFh), they are accessed through different addressing modes. The upper internal RAM can only be accessed through indirect addressing mode (indirect addressing with general purpose registers R0 or R1), and the special function registers (SFRs) are accessible only by direct addressing mode (address is part of the instruction).

The MiDAS2.1 family contains 256 bytes of AUXRAM. AUXRAM has the address space from 00H to FFH. The MiDAS2.1 family doesn't use any off-chip data memory.

6.1.1.2.1 Register Banks

The lowest 32 bytes of the internal data RAM are grouped into 4 banks of 8 general purpose registers (GPRs). Instructions call out these registers as R0 through R7. RS1 and RS0 bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than direct addressing instructions.

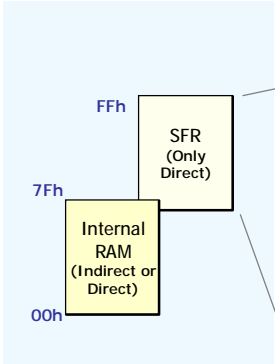
6.1.1.2.2 Bit-addressable Memory Space

The next 16 bytes, locations 20H through 2FH, above the register banks form a block of bit-addressable memory space. There are 128 bits in this area. The 128 bits can be directly addressed by single-bit instructions. The bit addresses in this area are 00H through 7FH.

Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 000B. The bit addresses in this area are 80H through FFH.

6.1.2 Special Function Registers (SFRs) Map and Description

The MiDAS2.1 family uses Special Function Registers (SFRs) to control and monitor peripherals. The SFRs are located at 80h-FFh and are accessed by direct addressing only. Some of the SFRs are bit addressable. This is very useful in cases where one wishes to modify a particular bit without changing the others. The SFRs that are bit addressable are those whose addresses end in 0h or 8h. The MiDAS2.1 family contains all the SFRs present in the standard 80C52. However, some SFRs have been added. In some cases unused bits in the original 80C52 have been given new functions. The list of SFRs is as follows. Refer to Appendix B for more details about SFRs.

Table 6-1 SFR map (* = bit addressable SFR)


Bit addressable

Legend:
 : Newly added SFR at MiDAS2.1 Family
 : Reserved for future use.

F8h	EIP							EEAEN	FFh
F0h	B				PODIR	P1DIR	P2DIR	P3DIR	F7h
E8h	EIE						ADCR	ADCON	EFh
E0h	ACC	ADCSELH	ADCSEL	ALTSEL	POSEL	P1SEL	P2SEL	P3SEL	E7h
D8h	WDCON	ADCHL	ADCHH	ADCHSEL	PWMCON		PWMD		DFh
D0h	PSW				P0TYPE	P1TYPE	P2TYPE	P3TYPE	D7h
C8h									CFh
C0h					PMR	STATUS			C7h
B8h	IP						OSCICN		BFh
B0h	P3								B7h
A8h	IE								AFh
A0h	P2		I2C_MCO N	I2C_MDEV	I2C_MAD R	I2C_MNUM	I2C_MSC L	I2C_MSC H	A7h
98h	SCON	SBUF	I2C_SCO N	I2C_SDEV	I2C_SADR	I2C_DAT	I2C_MDA T		9Fh
90h	P1	EXIF			CLKOFF	RINGCON	LVDCON	LVDST	97h
88h	TCON	TMOD	TL0	TL1	TH0	TH1			8Fh
80h	P0	SP	DPL	DPH				PCON	87h

Table 6-2 Summary of SFRs (*: undefined value)

Name	Symbol	Address	Reset Value	Bit Addressable
Accumulator	ACC		00000000	
B Register	B		00000000	
Program Status Word	PSW		00000000	
Stack Pointer	SP		00000111	
Data Pointer Low byte	DPL		00000000	
Data Pointer High byte	DPH		00000000	
Port 0	P0		11111111	
Port 1	P1		11111111	
Port 2	P2		11111111	
Port 3	P3		**111111	
Port 0 Pull-up Control	POSEL		00000000	
Port 1 Pull-up Control	P1SEL		00000011	

Port 2 Pull-up Control	P2SEL		00000000	
Port 3 Pull-up Control	P3SEL		**000000	
Port 0 Output Type	P0TYPE		00000000	
Port 1 Output Type	P1TYPE		00000000	
Port 2 Output Type	P2TYPE		00000000	
Port 3 Output Type	P3TYPE		**000000	
Port 0 Input/Output Direction	P0DIR		11111111	
Port 1 Input/Output Direction	P1DIR		11111111	
Port 2 Input/Output Direction	P2DIR		11111111	
Port 3 Input/Output Direction	P3DIR		**111111	
Alternate Function Control	ALTSEL		000000**	
PWM Control	PWMCON		0000*000	
PWM Duty Data	PWMD		00000000	
Interrupt Priority	IP		10*00000	
Interrupt Enable Control	IE		00*00000	
T/C 0/1 Control	TCON		00000000	
T/C 0/1 Mode Control	TMOD		***00000	
T/C 0 High byte	TH0		00000000	
T/C 0 Low byte	TL0		00000000	
T/C 1 High byte	TH1		00000000	
T/C 1 Low byte	TL1		00000000	
Serial Port Control of UART	SCON		***0**00	
Serial Data Buffer of UART	SBUF		00000000	
Power Control	PCON		0**10000	
ADC Control & ADC Result Low	ADCON		0010*000	
ADC Result High	ADCR		00000000	
ADC Channel Selection Low and MUX Selection	ADCSEL		11111111	
ADC Channel Selection High	ADCSELH		11111111	
ADC High Channel Selection High	ADCHH		11111111	
ADC High Channel Selection Low	ADCHL		11111111	
ADC High Channel Selection	ADCHSEL		0***0000	
Watchdog Timer Control	WDCON		11010000	
EEPROM Access Enable	EEAEN		*****0	
Power Management	PMR		***0***	

External Interrupt and LVD Control	EXIF		**001001	
Extended Interrupt Priority	EIP		**00**00	
Extended Interrupt Enable	EIE		**00**00	
Crystal Status	STATUS		***0***	
Internal RING Oscillator Control	OSCICN		****1100	
Clock Control for Minimizing Power Consumption	CLKOFF		**00*000	
Internal Ring Oscillator Tuning	RINGCON		01100000	
Power Fail Interrupt Source Selection	LVDCON		*****000	
Current Power Supply Status	LVDST		00000000	
I ² C Master Control	I2C_MCON		***00000	
I ² C Master Device Address	I2C_MDEV		00000000	
I ² C Master Memory Address	I2C_MADR		00000000	
I ² C Master Multi-byte Number	I2C_MNUM		00000000	
I ² C Master Clock Scale Factor Low Byte	I2C_MSCL		00000000	
I ² C Master Clock Scale Factor High Byte	I2C_MSCH		00000000	
I ² C Master Data	I2C_MDAT		00000000	
I ² C Slave Control	I2C_SCON		*000*000	
I ² C Slave Device Address	I2C_SDEV		00000000	
I ² C Slave Memory Address	I2C_SADR		00000000	
I ² C Slave Data	I2C_SDAT		00000000	

CAUTION: Don't modify * bits. Modifying these bits can cause the malfunctions. Especially the following bits of SFRs should not be updated.

- n EXIF.3(XT) : This bit must always remain high which is the default value.

6.1.3 On-Chip External Data Memory Access

There are two types of MOVX instructions: the MOVX @Ri and MOVX @DPTR. In the MOVX @Ri, the address of the on-chip external data memory comes from two sources. The lower 8-bits of the address are stored in the Ri register of the selected register bank. The upper 8-bits of the address come from the P2 SFR. In the MOVX @DPTR type, the full 16-bit address is supplied by the Data Pointers.

6.1.4 Instruction Set Summary

The MiDAS2.1 family executes all the instructions of the standard 80C52. The instructions of the MiDAS2.1 family produce the same result with those of the standard 80C52.

The internal architecture and timing of the MiDAS2.1 family are different from those of the standard 80C52. This means that the machine cycle of an individual instruction (the number of clock cycles per instruction) of the MiDAS2.1 family is different from that of the standard 80C52.

Refer to Appendix A for more information about instruction set.

Table 6-3 Summary of The Instruction Set

Type	Instruction	Description
Arithmetic	ADD	Addition
	ADDC	Addition with Carry
	SUBB	Subtraction with Borrow
	INC	Increment
	DEC	Decrement
	MUL	Multiply
	DIV	Divide
	DA	Decimal Adjust
Logical	ANL	Logical AND
	ORL	Logical OR
	XRL	Logical Exclusive OR
	CLR	Clear
	CPL	Complement
	RL	Rotate Left
	RLC	Rotate Left with Carry
	RR	Rotate Right
	RRC	Rotate Right with Carry
	SWAP	Swap nibbles
Data Transfer	MOV	Move data
	MOVC	Move Code
	MOVB	Move data to external RAM
	PUSH	Push
	POP	Pop
	XCH	Exchange

	XCHD	Exchange low-digit	
Boolean	CLR	Clear bit	
	SETB	Set bit	
	CPL	Complement bit	
	ANL	AND bit	
	ORL	OR bit	
	MOV	Move bit	
	JC	Jump if Carry is set	
	JNC	Jump if Carry is not set	
	JB	Jump if Bit is set	
	JNB	Jump if Bit is not set	
	JBC	Jump if Bit is set & Clear	
	Branch	ACALL	Absolute Call
		LCALL	Long Call
RET		Return from subroutine	
RETI		Return from interrupt	
AJMP		Absolute Jump	
LJMP		Long Jump	
SJMP		Short Jump	
JMP		Jump with DPTR	
JZ		Jump if ACC is zero	
JNZ		Jump if ACC is not zero	
CJNE		Compare and Jump if not equal	
DJNZ		Decrement and Jump if not zero	
NOP		No Operation	

6.1.4.1 Addressing Modes

The operands used in the instructions can be addressed in different modes. The MiDAS2.1 family uses six different addressing modes to this end:

- I Immediate
- I Direct
- I Indirect
- I Index
- I Register
- I Register-specific

6.1.4.1.1 Immediate Addressing Mode

Immediate addressing mode means that an operand is the constant value included in the instruction.

Examples:

MOV R3, #0FFh	R3 is loaded with constant value 0FFh
ORL PSW, #00000101b	Logical OR operation with PSW and 0000 0101b
MOV DPTR, #1243h	Load data pointer with constant value 1234h

6.1.4.1.2 Direct Addressing Mode

In direct addressing, the operand is specified by an 8-bit address field in the instruction.

Note: The lower 128 bytes of the normal internal RAM can be addressed in this mode. SFRs are accessible only with this addressing mode.

Examples:

MOV A, TCON	TCON is the 8-bit address of the SFR TCON.
MOV R4, variable	The variable is the 8-bit address of a memory location in the lower part of the internal RAM.

6.1.4.1.3 Indirect Addressing Mode

In indirect addressing, the instruction specifies a register that contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected register bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit “data pointer” register, DPTR.

Note: The lower and upper part of the internal RAM can be addressed by using 8-bit addresses. AUXRAM can be addressed by 16-bit addresses.

Examples:

MOV A, @R0	RAM is addressed by contents of R0 (8-bit address)
MOVX A, @DPTR	External data memory is addressed by contents of the selected data pointer DPTR (16-bit address)

6.1.4.1.4 Index Addressing Mode

Only program memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in program memory. A 16-bit base register (either DPTR or the PC) points to the base of the table, and the accumulator is set up with the table entry number. The address of the table entry in program memory is formed by adding the accumulator data to the base pointer data. Another type of indexed addressing is used in the “case jump” instruction. In this case, the destination address of a jump instruction is calculated as the sum of the base pointer data and the accumulator data.

Examples:

MOVC A, @A + DPTR	The address is the sum of DPTR and accumulator.
MOVC A, @A + PC	The address is the sum of PC and accumulator
JMP @A + DPTR	Jump to sum of accumulator and DPTR

6.1.4.1.5 Register Addressing Mode

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits (RS0 and RS1) in the PSW.

6.1.4.1.6 Register-Specific Addressing Mode

Some instructions are specific to a certain register. For example, some instructions always operate on the accumulator, or data pointer, etc., so no address byte is needed to point to it.

6.1.5 CPU Timing

The CPU timing for the MiDAS2.1 family is important, especially for those who wish to use software instructions to generate timing delays. Also, it provides the insight into the timing differences between the MiDAS2.1 family and the standard 80C52 as shown in Figure 6-2. In the MiDAS2.1 family, each machine cycle is four clock periods long. Each clock period is designated a state. Thus each machine cycle is made up of four states, S1, S2, S3 and S4 in that order. In order to reduce the instruction execution time, both the clock edges are used for internal timing. Hence it is important that the duty cycle of the clock should be as close to 50% as possible. Since the MiDAS2.1 family fetches one byte per machine cycle, in most of the instructions, the number of machine cycles needed to execute the instruction is equal to the number of bytes in the instruction.

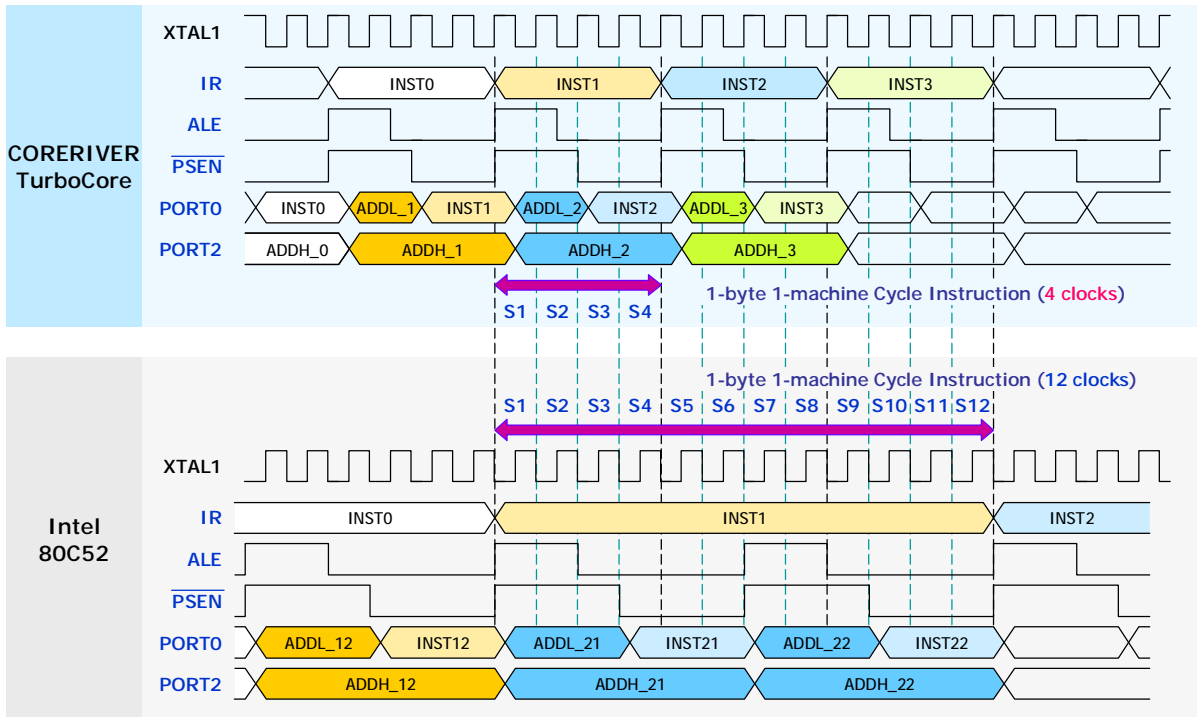


Figure 6-2 Timing Comparison of the MiDAS2.1 family and Intel 80C52

6.1.5.1 MOVX Instruction

The MiDAS2.1 family uses the MOVX instruction to access on-chip external data memory. While the results of the MOVX instruction are the same with those of the standard 80C52, the operation and the timing of the strobe signals have been modified in order to give the user much greater flexibility as shown in Figure 6-3 and Figure 6-4.

There are two type of MOVX instructions: the MOVX @Ri and MOVX @DPTR. In the MOVX @Ri, the address of the external data RAM comes from two sources. The lower 8-bits of the address are stored in the Ri register of the selected register bank. The upper 8-bits of the address come from the P2 register. In the MOVX @DPTR type, the full 16-bit address is supplied by the Data Pointers.

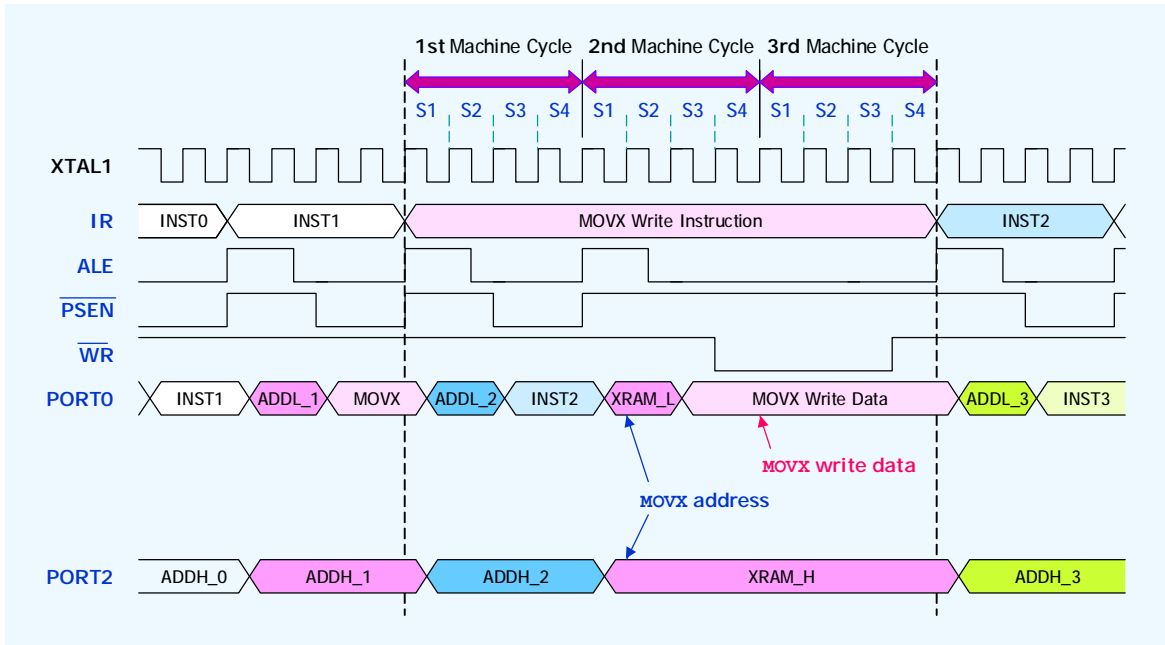


Figure 6-3 Write Timing of MOVX instruction

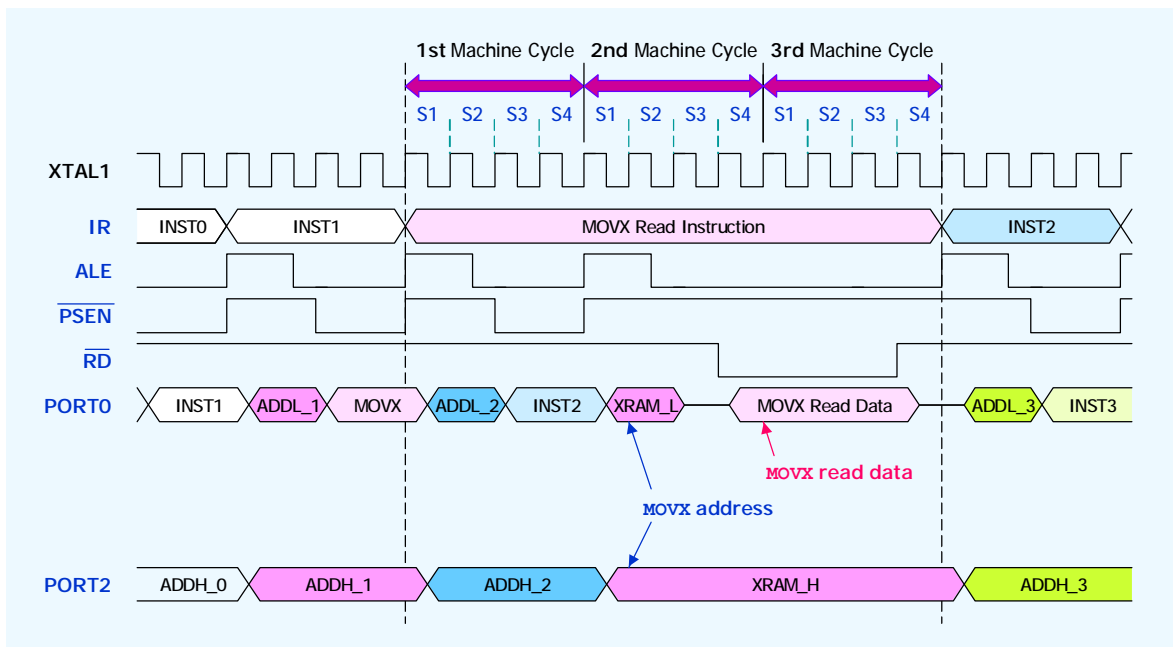


Figure 6-4 Read Timing of MOVX instruction

6.2 Peripheral Description

6.2.1 I/O Ports

Port 0, Port 1, Port 2, and Port 3 are bi-directional I/O ports. So the I/O ports have an I/O direction register (PxDIR). They consist of a latch (P0 through P3 registers), an I/O direction register, an I/O type register, an output driver, an input buffer, and a pull-up device.

6.2.1.1 PORT 0

Port 0 is an 8-bit bi-directional I/O port with pull-ups. Its pull-ups are switched on/off by changing the value of the P0SEL register. Clearing a bit of P0SEL to 0 will switch on the corresponding bit pull-up of Port 0. Otherwise, the pull up will be switched off. The bit value of the P0DIR register determines the I/O direction of Port 0. If you want to use a bit of Port 0 as input, set the corresponding bit of P0DIR to 1. The P0TYPE register determines the type of the output. If a bit of the P0TYPE register is set to 1, the corresponding pin operates in the open-drain mode. Otherwise, it operates in the push-pull mode.

I P0TYPE (D4h): Port 0 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	P0TYPE.7	P0TYPE.6	P0TYPE.5	P0TYPE.4	P0TYPE.3	P0TYPE.2	P0TYPE.1	P0TYPE.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Push-pull Output (Default) / 1 = Open-drain Output

I P0DIR (F4h): Port 0 Input/Output Control Register

Bit No.	7	6	5	4	3	2	1	0
	P0DIR.7	P0DIR.6	P0DIR.5	P0DIR.4	P0DIR.3	P0DIR.2	P0DIR.1	P0DIR.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Output / 1 = Input (Default)

I POSEL (E4h): Port 0 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	POSEL.7	POSEL.6	POSEL.5	POSEL.4	POSEL.3	POSEL.2	POSEL.1	POSEL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Pull-up resistor ON (Default) / 1 = Pull-up resistor OFF

I P0 (80h): Port 0 Register

Bit No.	7	6	5	4	3	2	1	0
	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

In order to configure a pin of Port 0 as an A/D converter channel input, the corresponding A/D converter channel has to be enabled (ADCxB or ADCHxB = 0) and the corresponding pull-up switched off (POSEL.x = 1). Refer to the A/D converter section for more information. Setting the PWM00 (ALTSEL.4) bit to 1 configure P0.0 as the PWM output. To use P0.0 as the TVO output, the TVO (ALTSEL.3) has to be set to 1. Setting the POSEL (PWMCON.7) bit to 1 configures P0.6 as the PWM output. To configure P0.3 and P0.4 as MSDA and MSCL, the I2C_MCON.0 bit must be set to 1. If pull-ups is not connected externally, internal pull-ups should be switched on by setting the bit POSEL.3 and the bit POSEL.4 to 1. Setting the ALTSEL.2 bit to 1 configures as the TXD. The other alternate functions can only be activated if the corresponding bit is configured as input.

I ADCSEL (E2h): ADC Channel Selection Low & MUX Selection

Bit No.	7	6	5	4	3	2	1	0
	ADC3B	ADC2B	ADC1B	ADC0B	CH3	CH2	CH1	CH0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADCXB = 0 : ADCX Input Enable & Digital Input Disable

I ADCSELH (E1h): ADC Channel Selection High Register

Bit No.	7	6	5	4	3	2	1	0
	ADC11B	ADC10B	ADC9B	ADC8B	ADC7B	ADC6B	ADC5B	ADC4B
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADCXB = 0 : ADCX Input Enable & Digital Input Disable

I ADCHH (DAh): ADC Channel Selection High Register

Bit No.	7	6	5	4	3	2	1	0
	ADCH15B	ADCH14B	ADCH13B	ADCH12B	ADCH11B	ADCH10B	ADCH9B	ADCH8B
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADCHXB = 0 : ADCHX Input Enable & Digital Input Disable

Port	Alternate Functions
P0.0	(ADCH9) A/D converter high channel 9 input or (/INT0) External Interrupt 0 or (PWM) Pulse Width modulation output or (TVO) Timer 0 Overflow output
P0.1	(ADC0) A/D converter channel 0 input or (/INT1) External Interrupt 1 input or (RXD) Serial Receive UART
P0.2	(ADC1) A/D converter channel 1 input or (INT2) External Interrupt 2 or (TXD) Serial Transmit UART
P0.3	(ADC2) A/D converter channel 2 input or (/INT3) External Interrupt 3 or (MSDA) I ² C master data
P0.4	(ADC3) A/D converter channel 3 or (MSCL) I ² C master clock
P0.5	(ADC4) A/D converter channel 4 input
P0.6	(ADC5) A/D converter channel 5 input or (PWM) Pulse Width modulation output
P0.7	(ADC6) A/D converter channel 6 input

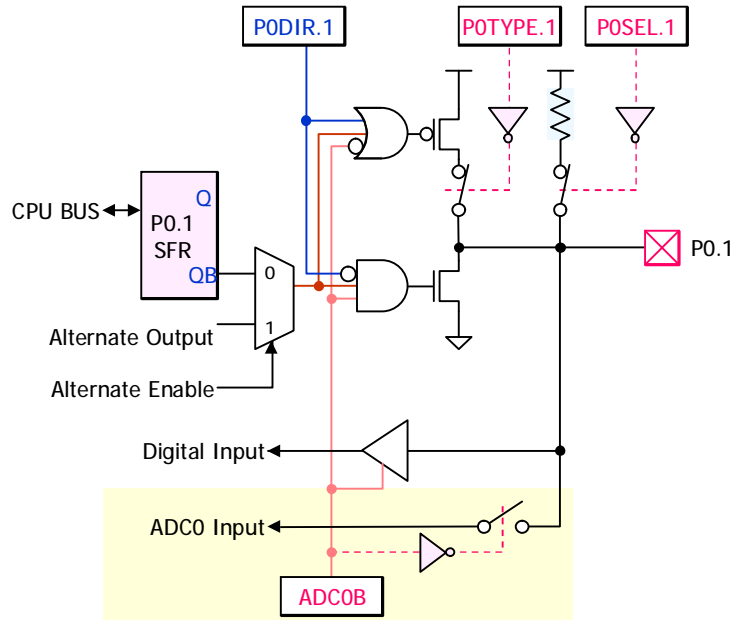


Figure 6-5 Configuration of PORT 0

Note: Read-Modify-Write instructions do not read port pin but SFR register.

ü ANL / ORL / XRL / JBC / CPL / INC / DEC / DJNZ / MOV PX.Y, C / CLR PX.Y / SETB PX.Y

6.2.1.2 PORT 1

Port 1 is an 8-bit bi-directional I/O port with pull-ups. Its pull-ups are switched on/off by changing the value of the P1SEL register. Clearing a bit of P1SEL to 0 will switch on the corresponding bit pull-up of Port 1. Otherwise, the pull up will be switched off. The bit value of the P1DIR register determines the I/O direction of Port 1. If you want to use a bit of Port 1 as input, set the corresponding bit of P0DIR to 1. The P1TYPE register determines the type of the output. If a bit of the P1TYPE register is set to 1, the corresponding pin operates in the open-drain mode. Otherwise, it operates in the push-pull mode.

I P1TYPE (D5h): Port 1 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	P1TYPE.7	P1TYPE.6	P1TYPE.5	P1TYPE.4	P1TYPE.3	P1TYPE.2	P1TYPE.1	P1TYPE.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Push-pull Output (Default) / 1 = Open-drain Output

I P1DIR (F5h): Port 1 Input/Output Control Register

Bit No.	7	6	5	4	3	2	1	0
	P1DIR.7	P1DIR.6	P1DIR.5	P1DIR.4	P1DIR.3	P1DIR.2	P1DIR.1	P1DIR.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Output / 1 = Input (Default)

I P1SEL (E5h): Port 1 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	P1SEL.7	P1SEL.6	P1SEL.5	P1SEL.4	P1SEL.3	P1SEL.2	P1SEL.1	P1SEL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(1)	R/W(1)

0 = Pull-up resistor ON (Default) / 1 = Pull-up resistor OFF

I P1 (90h): Port 1 Register

Bit No.	7	6	5	4	3	2	1	0
	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

I ALTSEL (E3h): Port 1 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	IOXEN	IORSTEN	CLO	PWM00	TV0	TX	-	-
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)		

IOXEN = 1 : XTAL1 and XTAL2 are configured as I/O Ports

I PMR (C4h): Power Management Control Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	XTOFF	-	-	-
					R/W(0)			

XTOFF: Internal amplifier disable for external crystal oscillator.
1 = External crystal will be killed.

0 = External crystal will run (Default).
 Don't set XTOFF bit to 1 when XT/RG = 1.

I STATUS (C5h): Crystal Status Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	XTUP	-	-	-	-
	R(0)							

XTUP: Crystal oscillator warm-up status. (External crystal oscillator)
 It represents the crystal clock is stable (1) or not (0).
 Cleared by H/W when Power-on reset and all kinds of reset.
 Cleared by H/W when XTOFF bit is set.
 Cleared by during Power-down wake-up when
 XT/RG (EXIF.3) = 1.
 Set by H/W after XTAL stabilization time.

I EXIF (91h) : External Interrupt Flag Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	IE3	IE2	XT/RG	RGMD	RGSL	BGS
		-	R/W(0)	R/W(0)	R/W(0)	R(1)	R/W(0)	R/W(1)

XT/RG: System clock selection.
 0 = Internal Ring oscillator is selected as system clock.
 1 = External clock is selected as system clock.

To use P1.0 and P1.1 for crystal input and output, one should set IOXEN (ALTSEL.7) to 1. In order to configure a pin of Port 1 as A/D converter channel input, the corresponding A/D converter channel has to be enabled (ADCxB or ADCHxB = 0) and the corresponding pull-up switched off (P1SEL.x = 1). Setting IORSTEN (ALTSEL.6) to 1 configures P1.2 as a general I/O pin. Otherwise, P1.2 is configured as the external reset input.

Port	Alternate Functions
P1.0	(XTAL1) Crystal Input
P1.1	(XTAL2) Crystal Output
P1.2	(RESETB) External Reset or (ADCH0) ADC input high channel 0
P1.3	(ADCH1) ADC input high channel 1 input
P1.4	(ADCH5) ADC input high channel 5 input

P1.5	(ADCH6) ADC input high channel 6 input
P1.6	(ADCH7) ADC input high channel 7 input
P1.7	(ADCH8) ADC input high channel 8 input

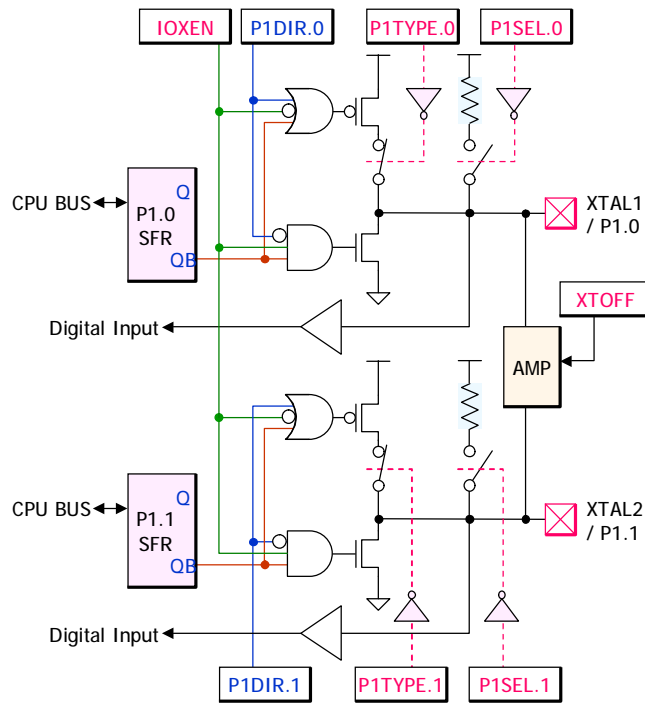


Figure 6-6 Configuration of P1.0 and P1.1

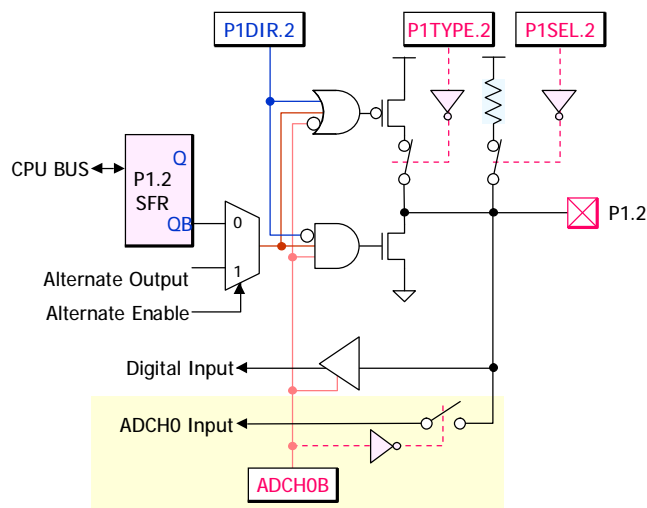


Figure 6-7 Configuration of the other P1 pins

Note: Read-Modify-Write instructions do not read port pin but port register.

ü ANL / ORL / XRL / JBC / CPL / INC / DEC / DJNZ / MOV PX.Y, C / CLR PX.Y / SETB PX.Y

6.2.1.3 PORT 2

Port 2 is an 8-bit bi-directional I/O port with pull-ups. Its pull-ups are switched on/off by changing the value of the P2SEL register. Clearing a bit of P2SEL to 0 will switch on the corresponding bit pull-up of Port 2. Otherwise, the pull up is switched off. The bit value of the P2DIR register determines the I/O direction of Port 2. If you want to use a bit of Port 2 as input, set the corresponding bit of P2DIR to 1. The P2TYPE register determines the type of the output. If a bit of the P2TYPE register is set to 1, the corresponding pin operates in the open-drain mode. Otherwise, it operates in the push-pull mode.

I P2TYPE (D6h): Port 2 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	P2TYPE.7	P2TYPE.6	P2TYPE.5	P2TYPE.4	P2TYPE.3	P2TYPE.2	P2TYPE.1	P2TYPE.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Push-pull Output (Default) / 1 = Open-drain Output

I P2DIR (F6h): Port 2 Input/Output Control Register

Bit No.	7	6	5	4	3	2	1	0
	P2DIR.7	P2DIR.6	P2DIR.5	P2DIR.4	P2DIR.3	P2DIR.2	P2DIR.1	P2DIR.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Output / 1 = Input (Default)

I P2SEL (E6h): Port 2 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	P2SEL.7	P2SEL.6	P2SEL.5	P2SEL.4	P2SEL.3	P2SEL.2	P2SEL.1	P2SEL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Pull-up resistor ON (Default) / 1 = Pull-up resistor OFF

I P2 (A0h): Port 2 Register

Bit No.	7	6	5	4	3	2	1	0
	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

I ADCHL (D9h): ADC Channel Selection Low Register

Bit No.	7	6	5	4	3	2	1	0
	ADCH7B	ADCH6B	ADCH5B	ADCH4B	ADCH3B	ADCH2B	ADCH1B	ADCH0B
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADCHXB = 0 : ADCHX Input Enable & Digital Input Disable

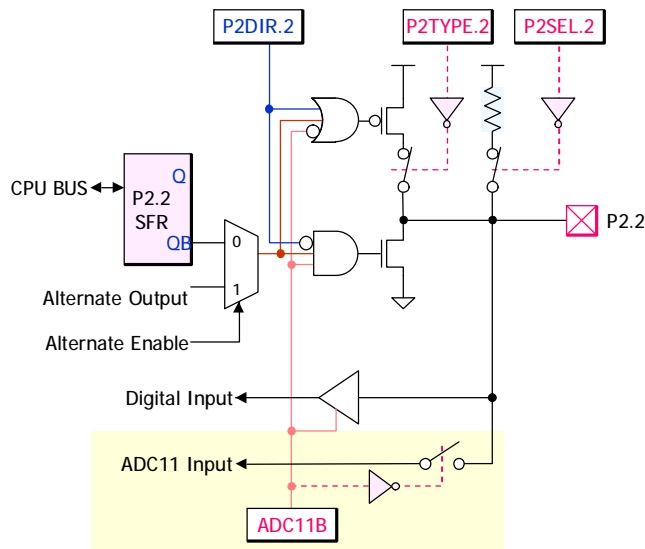


Figure 6-8 Configuration of PORT2

All Port 2 pins also serve the alternate functions specified below. In order to configure a pin of Port 2 as an A/D converter channel input, the corresponding A/D converter channel has to be enabled (ADCxB or ADCHxB = 0) and the corresponding pull-up switched off (P2SEL.x = 1). Refer to the A/D converter section for more information. To use P2.6 as system clock output, the CLO (ALTSEL.5) bit has to be set to 1. To configure P2.3 and P2.4 as SSDA and SSCL, the I2C_SCON.0 bit must be set to 1. If pull-ups is

not connected externally, internal pull-ups should be switched on by setting the bit P2SEL.3 and the bit P2SEL.4 to 1. The other alternate functions can only be activated if the corresponding bit is configured as input.

Port	Alternate Functions
P2.0	(ADCH2) ADC input high channel 2 input or (T0) Timer 0 input
P2.1	(ADCH3) ADC input high channel 3 input
P2.2	(ADC11) ADC input channel 11 input or (SSDA) I ² C slave data
P2.3	(ADC10) ADC input channel 10 input or (SSCL) I ² C slave clock
P2.4	(ADC9) ADC input channel 9 input
P2.5	(ADC8) ADC input channel 8 input
P2.6	(ADC7) ADC input channel 7 input or (CLO) System Clock Output
P2.7	(ADCH4) ADC input high channel 4 input

Note: Read-Modify-Write instructions do not read port pin but SFR register.

ü ANL / ORL / XRL / JBC / CPL / INC / DEC / DJNZ / MOV PX.Y, C / CLR PX.Y / SETB PX.Y

6.2.1.4 PORT 3

Port 3 is a 6-bit bi-directional I/O port with pull-ups. Its pull-ups are switched on/off by changing the value of the P3SEL register. Clearing a bit of P3SEL to 0 will switch on the corresponding bit pull-up of Port 3. Otherwise, the pull up is switched off. The bit value of the P3DIR register determines the I/O direction of Port 3. If you want to use a bit of Port 3 as input, set the corresponding bit of P3DIR to 1. The P3TYPE register determines the type of the output. If a bit of the P3TYPE register is set to 1, the corresponding pin operates in the open-drain mode. Otherwise, it operates in the push-pull mode.

I P3TYPE (D7h): Port 3 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	P3TYPE.5	P3TYPE.4	P3TYPE.3	P3TYPE.2	P3TYPE.1	P3TYPE.0
			R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Push-pull Output (Default) / 1 = Open-drain Output

I P3DIR (F7h): Port 3 Input/Output Control Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	P3DIR.5	P3DIR.4	P3DIR.3	P3DIR.2	P3DIR.1	P3DIR.0
			R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Output / 1 = Input (Default)

I P3SEL (E7h): Port 3 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	P3SEL.5	P3SEL.4	P3SEL.3	P3SEL.2	P3SEL.1	P3SEL.0
			R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Pull-up resistor ON (Default) / 1 = Pull-up resistor OFF

I P3 (B0h): Port 3 Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
			R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

All Port 3 pins also serve the alternate functions specified below. In order to configure a pin of Port 3 as an A/D converter channel input, the corresponding A/D converter channel has to be enabled (ADCxB or ADCHxB = 0) and the corresponding pull-up switched off (P3SEL.x = 1). Refer to the A/D converter section for more information.

Table 6-4 Alternate Functions

Port	Alternate Functions
P3.0	(ADCH10) ADC input high channel 10 input
P3.1	(ADCH11) ADC input high channel 11 input
P3.2	(ADCH12) ADC input high channel 12 input
P3.3	(ADCH13) ADC input high channel 13 input
P3.4	(ADCH14) ADC input high channel 14 input
P3.5	(ADCH15) ADC input high channel 15 input

More than one port pins are assigned to alternate functions without changing the functions of the other pins.

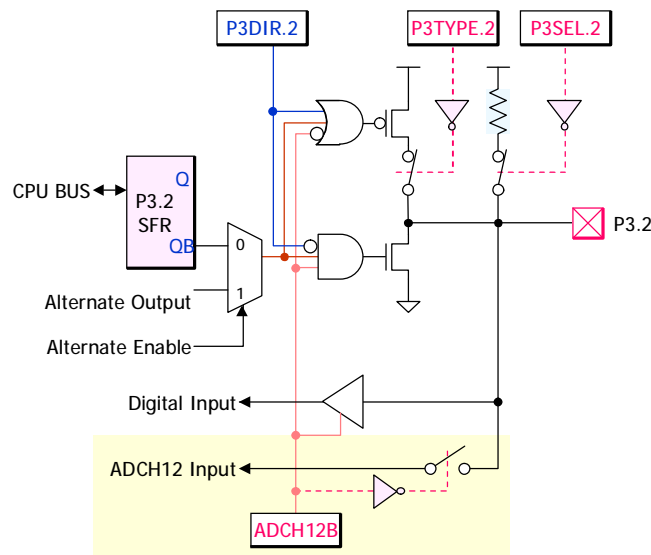


Figure 6-9 Configuration of PORT3

Note: Read-Modify-Write instructions do not read the port pin but the port latch.

ü ANL / ORL / XRL / JBC / CPL / INC / DEC / DJNZ / MOV PX.Y, C / CLR PX.Y / SETB PX.Y

6.2.1.5 Read-Modify-Write Instructions

Some instructions that read a port read the latch and others read the pin. The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called “read-modify-write” instructions. The instructions listed below are read-modify-write instructions.

Table 6-5 Read-Modify-Write Instructions

Instruction	Description
ANL	Logical AND
ORL	Logical OR
XRL	Logical Exclusive OR (XOR)
JBC	Branch if bit is set then clear bit
CPL	Complement bit

INC	Increment
DEC	Decrement
DJNZ	Decrement and branch if not zero
MOV PX.Y, C	Move the carry bit to bit Y of Port X
CLR PX.Y	Clear bit Y of Port X
SETB PX.Y	Set bit Y of Port X

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, then write the new byte back to the latch. The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of transistor. When logic 1 is written to the bit, the transistor is turned on. If the CPU reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as logic 0. Reading the latch rather than the pin will return the correct value of logic 1.

6.2.2 LVD (Low Voltage Detector)

6.2.2.1 Power-On Reset

When power is turned on, MiDAS2.1 Family restarts automatically by generating a power-on reset pulse internally. When using the internal power-on reset generation, power voltage slope must be in the range from 0.0V/us to 0.5V/us. That is, power voltage should be increase monotonically until it reaches to the normal range. Once the supply voltage, V_{CC} , rises above the threshold ($V_{RST} = 2.0V$), the device will execute the program code located at 0000h after about 8ms (3V operation).

The application software can find out using the Power-On Reset flag (POR in WDCON[6] or PCON[4]) that a Power-On Reset has occurred. It should clear the POR bit after reading it since the execution always starts from 0000h after all kinds of resets.

6.2.2.2 Power-Fail Reset

If the supply voltage falls below V_{RST} (Power-Fail), the device will wait recovery of the supply voltage. This condition will be maintained until it rises again above V_{RST} . After that, the device will restart the power-on reset and set the POR flag. Thus the recovery of V_{CC} above V_{RST} produces the same result with power-up.

6.2.2.3 Power-Fail Interrupt

The MiDAS2.1 device has 8 levels to detect the fall of V_{CC} : 4.0V, 3.6V, 3.2V, 3.0V, 2.8V, 2.6V, 2.4V, and

2.2V, The detection result is written in the LVDST register. The value of the LVDCON[2:0] selects the power-fail interrupt level among the 8 levels. The fall of V_{CC} sets the LVDST bit selected by LVDCON[2:0] to 1. Then, the power-fail interrupt occurs if enabled. Among all interrupts, the Power-fail Interrupt has the highest priority. Nobody can change the priority level. But it can be disabled if not needed. The EPFI bit (WDCON[5]) enables or disables the Power-fail Interrupt. This interrupt is independent of the global interrupt enable (EA). The Power-fail interrupt is a level-sensitive interrupt and the value of the selected LVDST will be maintained 'high' as long as V_{CC} remains below the selected level.

The registers listed below are for power voltage state:

- I LVDST: Low voltage detection result.
- I LVDCON[2:0]: Selection of low voltage level.
- I EXIF[0] (BGS) : If BGS = 0, LVD block stops when a device operates in power-down mode. Default value is 1.
- I WDCON[6] (POR) : Power ON Reset flag. After Power-on reset, this flag will be set.
- I WDCON[5] (EPFI) : Power-fail interrupt enable.
- I WDCON[4] (PFI) : Power-fail interrupt flag. When the selected LVDST bit is set to 1, this flag will be set.
- I PCON[4] (POF): Power-Off flag. After Power-on reset, this flag will be set.

The power-on reset are generated when V_{CC} rises above or falls below the reference voltages as shown in Figure 6-10.

I LVDST (97h) : LVD Status Register

Bit No.	7	6	5	4	3	2	1	0
	LVD7	LVD6	LVD5	LVD4	LVD3	LVD2	LVD1	LVD0
	R(0)	R(0)	R(0)	R(0)	R(0)	R(0)	R(0)	R(0)

LVD7 = 1 when $V_{DD} \leq 4.0V$.

LVD6 = 1 when $V_{DD} \leq 3.6V$.

LVD5 = 1 when $V_{DD} \leq 3.2V$.

LVD4 = 1 when $V_{DD} \leq 3.0V$.

LVD3 = 1 when $V_{DD} \leq 2.8V$.

LVD2 = 1 when $V_{DD} \leq 2.6V$.

LVD1 = 1 when $V_{DD} \leq 2.4V$.

LVD0 = 1 when $V_{DD} \leq 2.2V$.

I LVDCON (96h) : LVD Control Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	-	LVDCON.2	LVDCON.1	LVDCON.0
						R/W(0)	R/W(0)	R/W(0)

LVDCON[2:0] : Select the LVD Interrupt Level. (MUX)
 LVDCON[2:0] = 000b, LVD Interrupt Level = 4.0V (Default).
 LVDCON[2:0] = 001b, LVD Interrupt Level = 3.6V.
 LVDCON[2:0] = 010b, LVD Interrupt Level = 3.2V.
 LVDCON[2:0] = 011b, LVD Interrupt Level = 3.0V.
 LVDCON[2:0] = 100b, LVD Interrupt Level = 2.8V.
 LVDCON[2:0] = 101b, LVD Interrupt Level = 2.6V.
 LVDCON[2:0] = 110b, LVD Interrupt Level = 2.4V.
 LVDCON[2:0] = 111b, LVD Interrupt Level = 2.2V.

I WDCON (D8h) : Watchdog Timer Control Register

Bit No.	7	6	5	4	3	2	1	0
	WD1	WD0	EPFI	PFI	WDIF	WTRF	EWT	RWT
	R/W(1)	R/W(1)	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

EPFI: Power-fail interrupt enable.
 PFI : Power-fail interrupt flag.

I EXIF (91h) : External Interrupt Flag Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	IE3	IE2	XT/RG	RGMD	RGSL	BGS
			R/W(0)	R/W(0)	R/W(0)	R(1)	R/W(0)	R/W(1)

BGS: Band-gap select (Default = 1).
 When BGS = 0,
 Band-gap block (LVD) will do not run in power-down mode, but function during normal mode.
 When BGS = 1,
 Band-gap block (LVD) will run in power-down mode.

I PCON (87h) : Power Control Register

Bit No.	7	6	5	4	3	2	1	0
	SMOD1	-	-	POF	GF1	GF0	PD	IDL
	R/W(0)			R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

POF: Power off flag.
 When power-on, this flag bit will be set by H/W.
 PD : Power-down (Stop) mode enable.

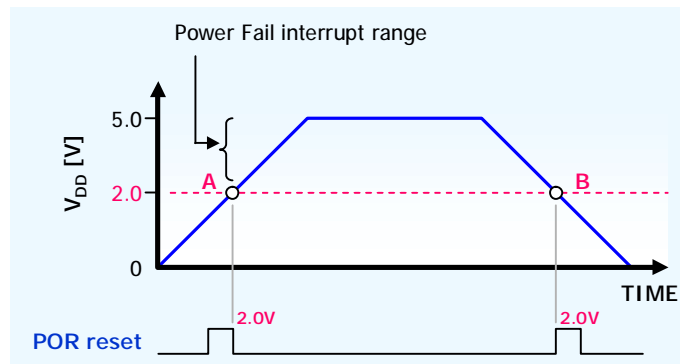


Figure 6-10 Power-On Reset/Power-fail Reset and Power-fail interrupt

The LVD block is described in Figure 6-11. It always remains activated except when $BGS = 0$ and a MiDAS2.1 processor is in power-down mode.

If a system needs to detect the power-fail in power-down mode ($PCON[1]=1$), you should set the BGS bit ($EXIF[0]$) to logic 1. Then, the band-gap reference and the power-fail detection circuits will remain activated in power-down mode. As a result, the power consumption of power-down mode increases. If the LVD block is deactivated, typically $1 \mu A$ of current will flow through the MiDAS2.1 processor. The current will increase to about $200 \mu A$ by activation of the LVD block.

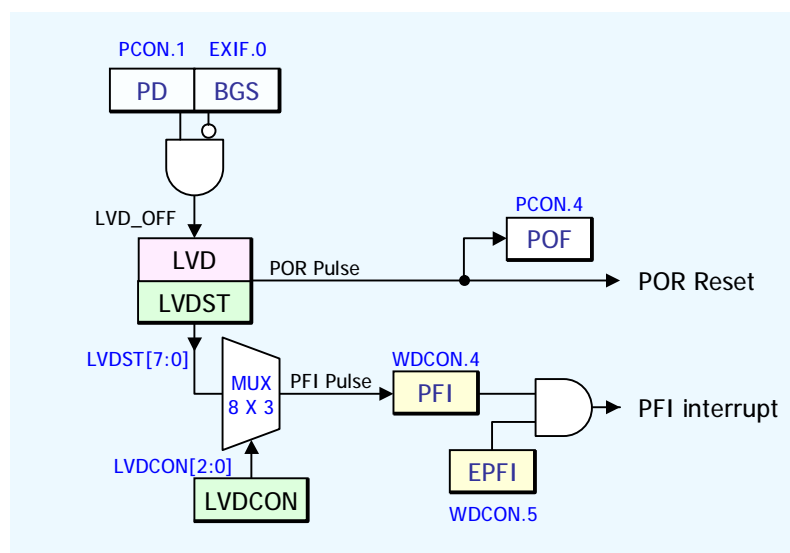


Figure 6-11 LVD Block Diagram

6.2.3 WDT (Watchdog Timer)

The Watchdog timer is a free-running timer with programmable time-out intervals. It is available for a system monitor, a time-base generator or an event timer. It allows an automatic recovery from software upset due to some cause including external noise. It is a kind of counter with the programmable bit length. It counts the system clock pulses. A software can always clear the value of the watchdog timer. When the count rolls over from all 1s to all 0s, the time-out occurs. At that time, the watchdog interrupt flag is set to 1. The watchdog interrupt will occur if EWDT (watchdog interrupt enable) and EA (global enable) are set. If the watchdog reset is enabled by setting the bit EWT to logic 1, the watchdog reset will occur after 256 clock cycles since the watchdog timer interrupt flag is set, However, the watchdog timer can be cleared by setting the bit RWT to logic 1 during the delay of 256 clock cycles. Then the watchdog reset will not occur. The watchdog interrupt and reset may be used independently of each other. So you can use them separately or together. Figure 6-12 shows the block diagram for Watchdog timer.

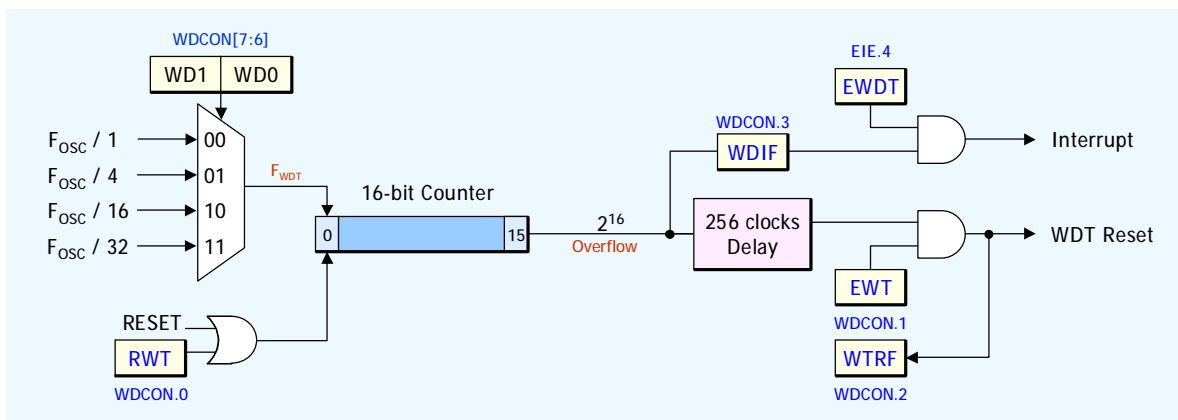


Figure 6-12 Block Diagram for Watchdog Timer

I WDCON (D8h) : Watchdog Timer Control Register

Bit No.	7	6	5	4	3	2	1	0
	WD1	WD0	EPFI	PFI	WDIF	WTRF	EWT	RWT
	R/W(1)	R/W(1)	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

- WD[1:0]: WDT Clock Divide(1/4/16/32)
- WDIF: Watchdog Timer Interrupt Flag
- WTRF: Watchdog Timer Reset Flag. Only cleared by S/W.
- EWT: Watchdog Timer Reset Enable
- RWT: Restart Watchdog Timer

First the Watchdog timer should be cleared by setting the bit RWT (WDCON.0) to 1. Then the timer

restarts from 0. After that, the hardware clears the RWT bit automatically. The time-out interval is selected by the values of the two bits WD1 and WD0 (WDCON.7 and WDCON.6). When the timer reaches to the selected time-out, the Watchdog interrupt flag WDIF (WDCON.3) is set. If the watchdog reset is enabled by setting the bit EWT to 1, the watchdog reset will start after 256 clock cycles since the watchdog timer interrupt flag is set. However, the watchdog timer can be cleared by setting the bit RWT to 1 during the delay of 256 clock cycles. Then the watchdog reset will not occur. After thirty clock cycles from that, the Watchdog timer reset flag WTRF (WDCON.2) will be set. This flag indicates to the software that the Watchdog time-out was the cause of the reset.

If the reset and the interrupt are disabled, the watchdog timer can be used as a simple timer. Every time the selected time-out occurs, the WDIF flag is set. Software can check the WDIF flag to detect a time-out and set the bit RWT to restart the timer. The Watchdog timer can also be used as a timer with a very long time-out interval. The interrupt is enabled in this case. Every time the time-out occurs, the interrupt service routine will be called if the global interrupt enable bit EA is set.

The main purpose of the watchdog timer is detection of software upset. This is important for real-time control applications. In this case, some power glitches or electro-magnetic interferences may cause software upset.

The watchdog time-out interval varies with the clock frequency. The watchdog timer reset will start after 256 clock cycles from the end of the time-out interval.

Table 6-6 Time-out values for the Watchdog timer

WD1	WDO	Interrupt time-out (@4MHz)		Reset time-out (@4MHz)	
0	0	1×2^{16} clocks	16.38ms	$1 \times 2^{16} + 256$ clocks	16.45ms
0	1	4×2^{16} clocks	65.54ms	$4 \times 2^{16} + 256$ clocks	60.60ms
1	0	16×2^{16} clocks	262.14ms	$16 \times 2^{16} + 256$ clocks	261.21ms
1	1	32×2^{16} clocks	524.29ms	$32 \times 2^{16} + 256$ clocks	524.35ms

The Watchdog timer will be disabled by a power-on/fail reset. The Watchdog timer reset does not disable the watchdog timer, but will restart it. The default Watchdog time-out interval is 32×2^{16} clocks, which is the longest time-out period.

6.2.4 Timer/Counter 0/1

The MiDAS2.1 family has two 16-bit programmable timer/counters. Each of these Timer/Counters has two 8-bit registers that form the 16-bit counting register. For Timer/Counter 0, they are the upper 8-bit

register TH0 and the lower 8-bit register TL0. Similarly, Timer/Counter 1 has two 8-bit registers, TH1 and TL1.

6.2.4.1 The Operation of Timer/Counter 0/1

Timer 0 can be configured to operate either as a timer or an event counter. When operating as a timer, the counting registers counts clock pulses. The counting frequency of the timer can be 1/12 of the system clock frequency. In the “Counter” function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 pin for Timer 0 and T1 pin for Timer 1. The T0 and T1 external inputs are sampled during S3 state of every 12-clock system. If the sampled value is high in one machine cycle and low in the next, a valid high-to-low transition on the pin is recognized and the counting register is incremented. Since it takes 24 clock cycles to recognize a negative transition on the pin, the maximum counting rate is 1/24 of the master clock frequency. In either the “Timer” or “Counter” mode, the counting register will be updated at S2 state. Therefore, in the “Counter” mode, the recognized negative transition on T0 or T1 pin causes the counting register value to increase in the next machine cycle after the negative edge was detected.

The “Timer” or “Counter” mode is selected by the value of the C/T bit in the TMOD register; bit 2 of TMOD selects the mode for Timer/Counter 0. In addition, Timer/Counter 0 has four operating modes. The modes are selected by the value of the M0 and M1 bits in the TMOD register.

Timer 1 operates only as a timer and only in the operating mode 2.

6.2.4.1.1 Mode 0

In Mode 0, only the timer/counter 0 operates as an 8-bit counter with a divide-by-32 prescaler. The timer/counter 1 doesn't support this mode 0. This 13-bit counter consists of 8 bits of TH0 and the lower 5 bits of TL0. The upper 3 bits of TL0 are indeterminate and should be ignored.

The value in the TL0 register increases at the negative edge of the clock. Every time the value of the fifth bit in TL0 changes from 1 to 0, the count in the TH0 register increases. When the count in TH0 rolls over from FFh to 00h, the overflow flag TF0 in the TCON register is set. The counting continues only if TR0=1 and either GATE=0 or /INT0=1. When C/T is cleared to 0, it will count clock cycles, and if C/T is set to 1, then it will count 1 to 0 transitions on T0 (P3.4) for Timer 0.

When the 13-bit counter rolls over from 1FFFh to 000H, the timer 0 overflow flag TF0 is set to 1. If enabled, a timer interrupt will occur.

6.2.4.1.2 Mode 1

Mode 1 is similar with Mode 0, except that the timer register operates with all 16 bits. This means that all the bits of TH0 and TL0 are used. Roll-over occurs when the timer changes from FFFFh to 0000h. The timer 0 overflow flag TF0 is set to 1 and an timer interrupt will occur if enabled. The function of GATE bit is the same as Mode 0.

6.2.4.1.3 Mode 2

In Mode 2, the Timer/Counters operate as 8-bit counters with automatic reload. TLx operates as an 8-bit counter, while THx holds the reload value. When the TLx register overflows from FFH to 00H, the TFx bit in TCON is set and TLx is reloaded with the contents of THx, and the counting process continues from reloaded value. The reload operation leaves the contents of the THx register unchanged. Counting of timer 0 is enabled when TR0 = 1 and either GATE = 0 or /INT0 = 1. As in the other two modes 0 and 1, Mode 2 allows counting of a timer clock (clock/12) or pulses on pin T0. Timer 1 doesn't operate as an event timer. Counting of timer 1 is enabled when TR1 = 1.

6.2.4.1.4 Mode 3

Timer/Counter 0 in Mode 3, establishes TL0 and TH0 as two separate 8-bit counters. TL0 uses the Timer/Counter 0 control bits: C/T, GATE, TR0, /INT0 and TF0. It is determined by C/T (TMOD.2) whether the TL0 can count clock cycles (clock/12) or 1-to-0 transitions on pin T0. TH0 is locked in a timer function (clock/12) and takes over the use of TR1 and TF1 from Timer/Counter 1. Mode 3 is used when an extra 8-bit timer is needed. With Timer 0 in Mode 3, Timer 1 can still be used in Modes 2. However, it no longer has control over its overflow flag TF1 and the enable bit TR1.

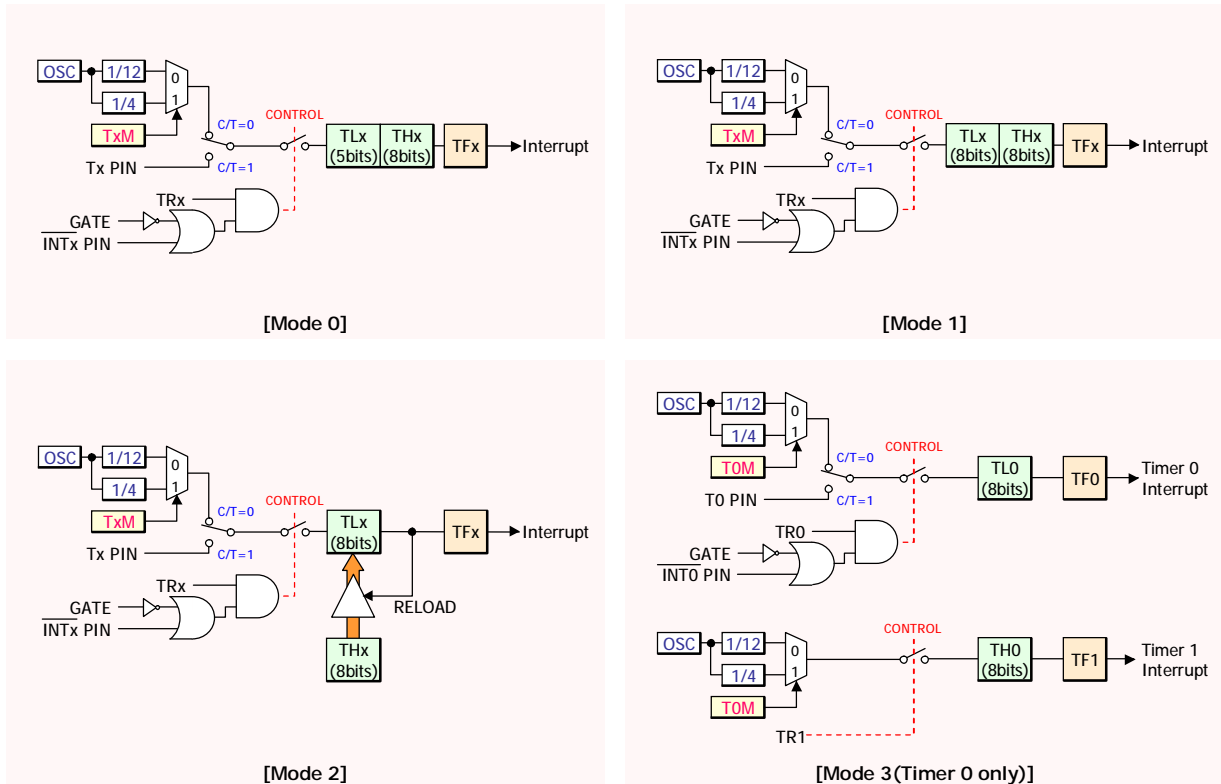


Figure 6-13 Timer/Counter 0 operations in Mode 0/1/2/3

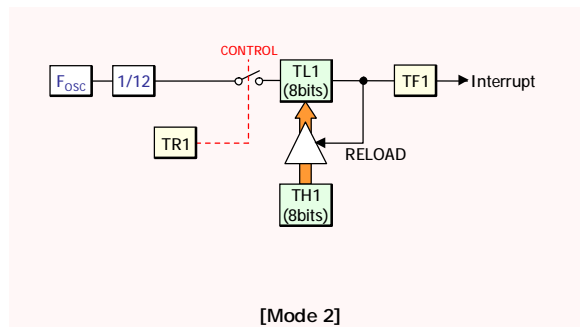


Figure 6-14 Timer/Counter 1 operation in Mode 2

I TCON (88h) : Timer/Counter Control Register

Bit No.	7	6	5	4	3	2	1	0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

TF1: Timer 1 Overflow Flag.
TR1: Timer 1 Run Enable.

TF0: Timer 0 Overflow Flag.
TR0: Timer 0 Run Enable.

I TMOD (89h) : Timer/Counter 0 Mode Control Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	GATE	C/T	M1	M0
					R/W(0)	R/W(0)	R/W(0)	R/W(0)

GATE: Timer 0 Gate control. When TRx (in TCON) is set and GATE=1, Timer x will run only while INTx pin is high (hardware control). When GATE=0, Timer x will run only while TRx=1 (software control).

C/T[2]: Timer 0 Counter/Timer Select.
0 = Timer by FOSC/12. (Default)
1 = Counter by T0 pin.

M1, M0: Timer 0 Mode Select.
[0,0] : Mode 0. 13-bit T/C.
[0,1] : Mode 1. 16-bit T/C.
[1,0] : Mode 2. 8-bit T/C with automatic reload
[1,1] : Mode 3. Two 8-bit T/C

I TL0 (8Ah) : Timer/Counter 0 Low Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TL0.7	TL0.6	TL0.5	TL0.4	TL0.3	TL0.2	TL0.1	TL0.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I TH0 (8Ch) : Timer/Counter 0 High Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TH0.7	TH0.6	TH0.5	TH0.4	TH0.3	TH0.2	TH0.1	TH0.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I TL1 (8Bh) : Timer/Counter 1 Low Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TL1.7	TL1.6	TL1.5	TL1.4	TL1.3	TL1.2	TL1.1	TL1.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I TH1 (8Dh) : Timer/Counter 1 High Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TH1.7	TH1.6	TH1.5	TH1.4	TH1.3	TH1.2	TH1.1	TH1.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

6.2.5 Simplified UART (Universal Asynchronous Rx/Tx)

The MiDAS2.1 family has a serial full duplex port. 'Full duplex' means the port can transmit and receive simultaneously. The simplified UART port supports only asynchronous communication. The transmit buffer and the receive buffer are both addressed as SBUF SFR for UART. However, writing to SBUF for UART loads the transmit buffer, and reading SBUF for UART accesses a physically separate receive buffer. The simplified UART port can operate only in mode 1.

6.2.5.1 Mode 1

In Mode 1, the simplified UART communicates asynchronously in the full duplex. Ten bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receiving, the stop bit goes into RB8 in the SCON register. The baud rate can be programmed to be 1/16 or 1/32 of the Timer 1 overflow. It varies widely according to the automatic reload value of Timer 1.

Transmission starts from any instruction that uses SBUF as a destination register. The serial data is sent to TXD pin at S1 state after the first roll-over of the divide-by-16 counter. The next bit is placed on TXD pin at S1 state after the next roll-over of the divide-by-16 counter. Thus, the transmission is synchronized to the divide-by-16 counter, not directly to the "write to SBUF" signal. After transmission of all 8-bit data, the stop bit is transmitted. The TI flag is set in the S1 state after the stop bit has been sent to TxD pin. This will occur at the 10th roll-over of the divide-by-16 counter after a write to SBUF.

Reception is enabled only if REN is "1". It is initiated by detecting 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate established. When a transition is detected, the divide-by-16 counter is immediately reset. This aligns its roll-overs with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 8th, 9th, and 10th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of three samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register,

and reception of the rest of the frame will proceed.

After shifting in 8-bit data, there is one more shift to do, after which the SBUF and RB8 are loaded and RI is set. However certain conditions must be met before the loading and setting of RI can be done.

1. RI must be 0 and
2. Either SM2 = 0, or the received stop bit = 1

If these conditions are met and the stop bit goes to RB8, the 8-bit data go into SBUF and RI is set. Otherwise the received frame may be lost. After the middle of the stop bit, the receiver goes back to looking for a 1-to-0 transition in RXD.

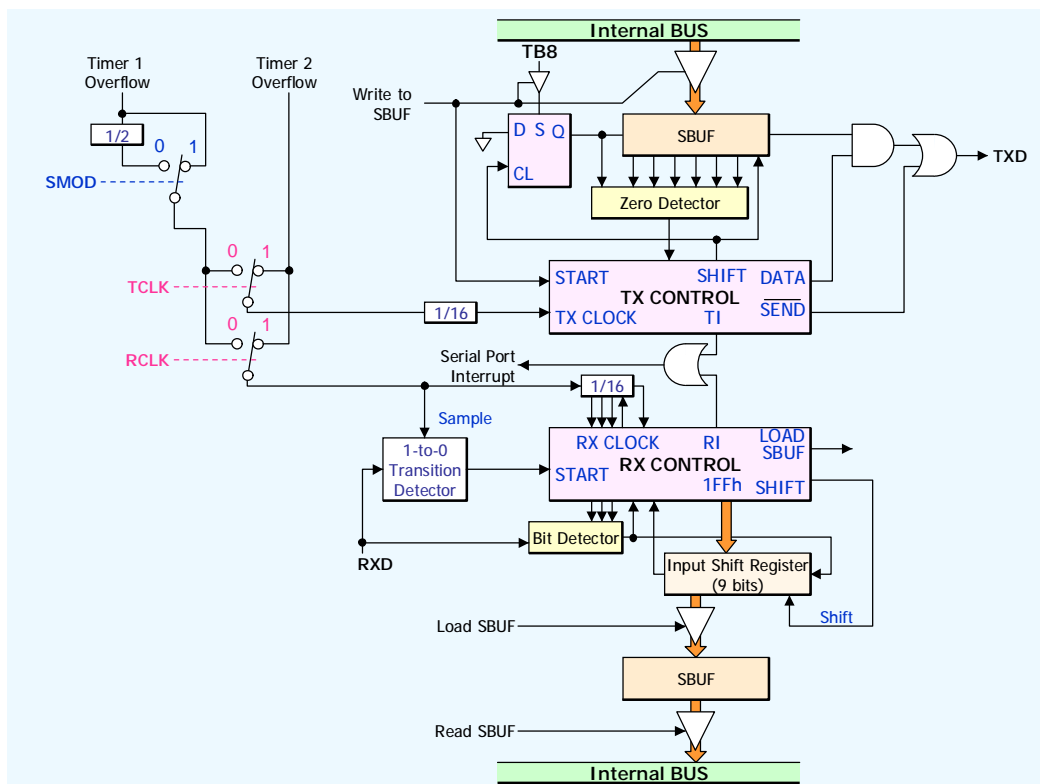


Figure 6-15 UART Mode 1

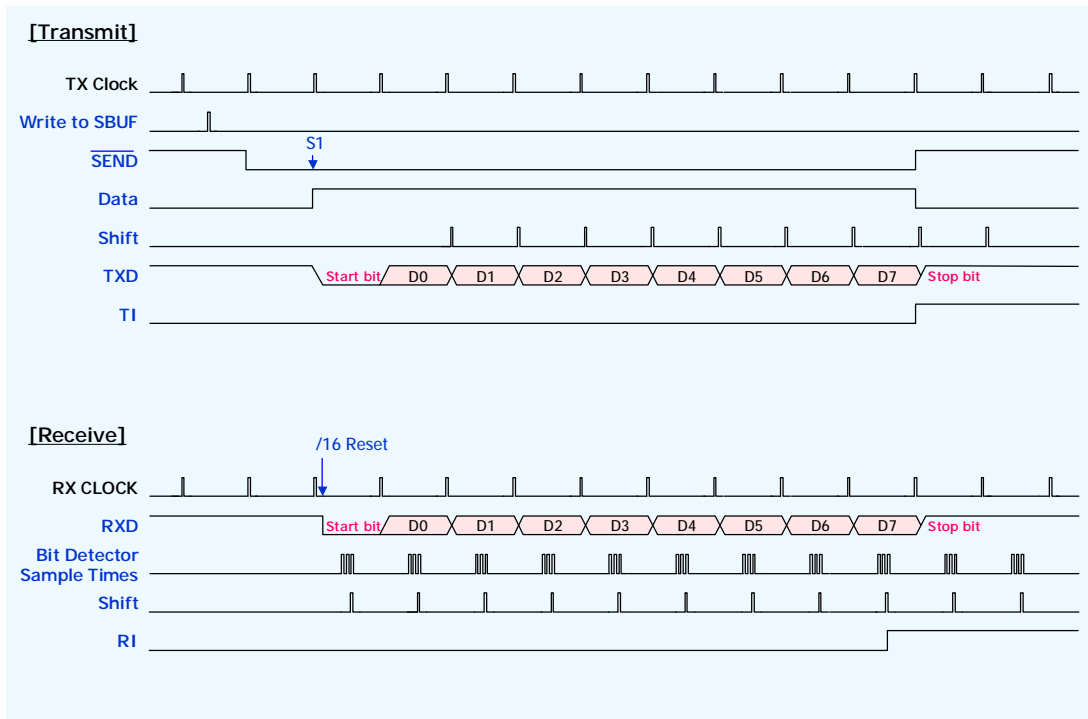


Figure 6-16 UART Mode 1 Timing

Table 6-7 Operation Summary of the simplified UART

Mode	Data Size	Data Format	Baud Rate
1	10 bit	Start bit, 8 data bit, stop bit	1/32 * Timer 1 overflow (SMOD1=0) 1/16 * Timer 1 overflow (SMOD1=1)

6.2.5.2 Baud Rates

In the MiDAS2.1 family, the baud rates in Mode 1 can be determined by the overflow rate of Timer 1.

Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 are determined by the Timer 1 overflow rate and the value of SMOD1 as follows:

$$\text{Modes 1 Baud Rate} = 2^{\text{SMOD1}} / 32 * (\text{Timer 1 Overflow Rate})$$

The Timer 1 interrupt should be disabled in this application. In the most typical applications, it is configured for “timer” operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case, the baud rate is given by the formula

$$\text{Modes 1 Baud Rate} = 2^{\text{SMOD1}}/32 * (\text{Oscillator Frequency}) * 1/12 * 1/[256-(\text{TH1})]$$

Table 6-8 Baudrate Examples of the simplified UART

Baudrate	UART Mode	Fosc [MHz]	SMOD1	Timer 1		
				C/T	Mode	Reload Value (TH1)
62.5 KHz	Mode 1	12	1	0	Mode 2 8-bit Auto-reload	FFh
19.2 KHz		11.0592	1	0		FDh
9.6 KHz		11.0592	0	0		FDh
4.8 KHz		11.0592	0	0		FAh
2.4 KHz		11.0592	0	0		F4h
1.2 KHz		11.0592	0	0		E8h
137.5 KHz		11.0592	0	0		1Dh
110 Hz		6	0	0		72h

6.2.6 PWM (Pulse Width Modulator)

The MiDAS2.1 family has a one-channel 8-bit PWM circuit. The two ports can be selected for the PWM output. The operation of the PWM circuit is controlled by a control register; PWMCON.

The PWM counter is an 8-bit increment counter. The counter increases until overflow occurs. To select P0.6 for the PWM waveform output, set PWMCON[7] to “1”. If ALTSEL[4] is set to 1, P0.0 is selected for the PWM waveform output. If PWMCON[0] is set to 1, the PWM counter starts and the PWM circuit operates. When the counter stops, it retains its current count value; when re-started, it resumes counting from the retained count value. Setting PWMCON[1] to 1 clears the value of the PWM counter to 0. As the counter rolls over from all 1s to all 0s, the PWM interrupt flag (PWMF), PWMCON[2] is set to 1.

You can select the clock frequency for the PWM counter by set PWMCON[6:4]. The clock frequency that you can select is $F_{\text{OSC}}/128$, $F_{\text{OSC}}/64$, $F_{\text{OSC}}/32$, $F_{\text{OSC}}/16$, $F_{\text{OSC}}/8$, $F_{\text{OSC}}/4$, $F_{\text{OSC}}/2$, or $F_{\text{OSC}}/1$.

6.2.6.1 Block Diagram

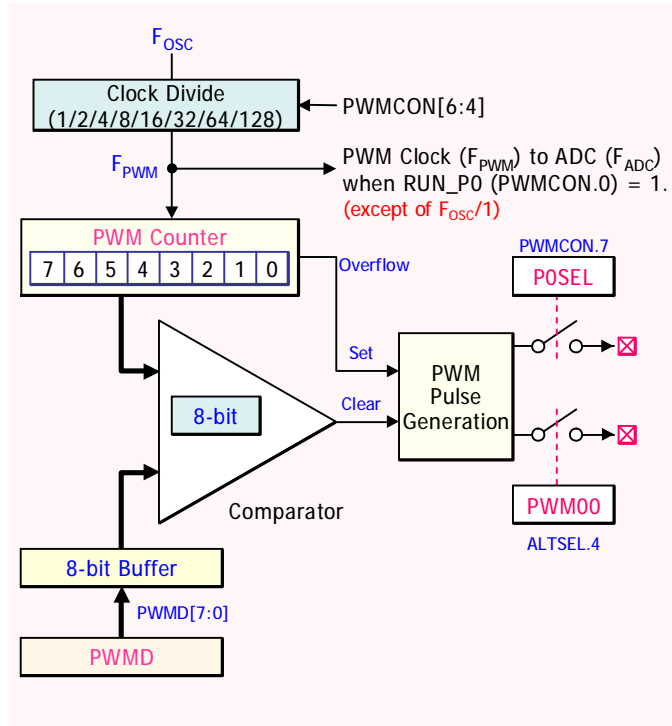


Figure 6-17 Functional block diagram

The PWM counter operates using all its 8 bits and determines the PWM duty rate by comparing its counting value with the value of PWMD.

6.2.6.2 PWM components

The 8-bit PWM circuit has the following components:

- I An 8-bit comparator
- I An 8-bit increment counter
- I An 8-bit reference data register (PWMD)
- I Two PWM output pins (PWM waveform output to P0.6 by PWMCON[7] or P0.0 by ALTSEL[4])

The PWM counter operates as an 8-bit counter. The counter increases using all 8 bits in the counter.

6.2.6.3 PWM clock frequency

The timing characteristics of PWM output are based on the prescaled F_{OSC} clock frequency. The PWM counter clock value is determined by the setting of PWMCON[6:4] (Prescaled Clock Selection for PWM). Table 6-9 shows the clock rate by PWMCON[6:4].

Note: PWM clock (F_{PWM}) to ADC should not be set to $F_{OSC}/1$.

Table 6-9 PWM clock rate by PWMCON[6:4]

PWMCON[6:4]			PWM Clock Rate (F_{PWM})
PS2_P0	PS1_P0	PS0_P0	
0	0	0	$F_{OSC} / 1$
0	0	1	$F_{OSC} / 2$
0	1	0	$F_{OSC} / 4$
0	1	1	$F_{OSC} / 8$
1	0	0	$F_{OSC} / 16$
1	0	1	$F_{OSC} / 32$
1	1	0	$F_{OSC} / 64$
1	1	1	$F_{OSC} / 128$

6.2.6.4 PWM Function Description

The PWM output signal drops to low level when the 8bits counter value matches the reference data register (PWMD) value. If the values in the PWMD register are not zero, an overflow of the 8bits counter causes the PWM output to rise from low level to high level. In this way, the reference data register value determines the module's duty cycle.

6.2.6.5 PWM Timing Diagram

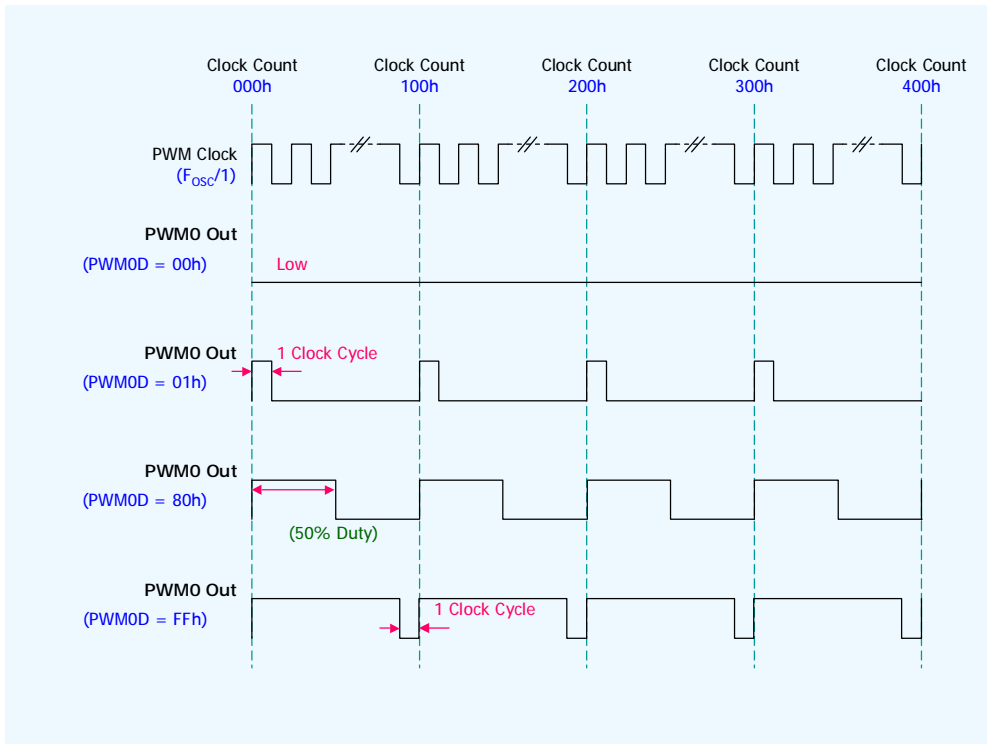


Figure 6-18 PWM timing diagram

6.2.6.6 PWMCON (DCh) : PWM Control Register

Bit No.	7	6	5	4	3	2	1	0
	POSEL	PS2_P0	PS1_P0	PS0_P0	-	PWMF	CLR_P0	RUN_P0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)		R/W(0)	R/W(0)	R/W(0)

Symbol	Description																
POSEL	PWM Waveform Output Enable to P0[6]. 0 = Disable the PWM waveform output to P[6]. (Default) 1 = Enable the PWM waveform output to P[6].																
PS2_P0 PS1_P0 PS0_P0	Prescaled Clock Selection. Note that PWM clock (F_{PWM}) to ADC should not be set to $F_{OSC}/1$.																
	<table border="1"> <thead> <tr> <th>PS2_P0</th> <th>PS1_P0</th> <th>PS0_P0</th> <th>PWM Clock Rate (F_{PWM})</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>$F_{OSC} / 1$</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>$F_{OSC} / 2$</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>$F_{OSC} / 4$</td> </tr> </tbody> </table>	PS2_P0	PS1_P0	PS0_P0	PWM Clock Rate (F_{PWM})	0	0	0	$F_{OSC} / 1$	0	0	1	$F_{OSC} / 2$	0	1	0	$F_{OSC} / 4$
PS2_P0	PS1_P0	PS0_P0	PWM Clock Rate (F_{PWM})														
0	0	0	$F_{OSC} / 1$														
0	0	1	$F_{OSC} / 2$														
0	1	0	$F_{OSC} / 4$														

	0	1	1	$F_{OSC} / 8$
	1	0	0	$F_{OSC} / 16$
	1	0	1	$F_{OSC} / 32$
	1	1	0	$F_{OSC} / 64$
	1	1	1	$F_{OSC} / 128$
-	Reserved.			
PWMF	PWM Interrupt Flag. This flag bit is cleared by software.			
CLR_PO	Counter Reset Enable. This flag bit is cleared by hardware.			
RUN_PO	Counter Start Enable.			

6.2.6.7 PWMD (DEh) : PWM Control Register

Bit No.	7	6	5	4	3	2	1	0
	PWMD.7	PWMD.6	PWMD.5	PWMD.4	PWMD.3	PWMD.2	PWMD.1	PWMD.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

PWMD duty data register, located in DEh, determines the duty cycle of the output waveform. To adjust the PWM output duty cycle, change the values of the 8-bit duty data register (PWMD).

6.2.6.8 ALTSEL (E3h) : Alternate Function Selection Register

Bit No.	7	6	5	4	3	2	1	0
	IOXEN	IORSTEN	CLO	PWM00	TVO	TX	-	-
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Description
PWM00	PWM Waveform Output Enable to P0[0]. 0 = Disable the PWM waveform output to P[0]. (Default) 1 = Enable the PWM waveform output to P[0].
-	Reserved.

6.2.7 A/D Converter (Analog to Digital Converter)

The 10-bit A/D converter converts analog input voltages into 10-bit digital values with successive approximation logic. The analog input voltages can come in through twenty eight input channels. The analog input voltage must be in the range between the V_{DD} and V_{SS} .

The A/D converter consists of the following components:

- I An analog comparator with successive approximation register
- I A D/A converter
- I An A/D converter control register (ADCON)
- I An A/D converter input channel enable high register (ADCSELH)
- I An A/D converter input channel low and mux selection register (ADCSEL)
- I Twenty eight multiplexed analog data input pins (ADC0–ADC11, ADCH0–ADCH15)
- I An A/D converter high channel high enable register (ADCHH)
- I An A/D converter high channel low enable register (ADCHL)
- I An A/D converter high channel selection register (ADCHSEL)
- I An 10-bit A/D conversion result data buffer register (ADCR[7:0] and ADCON[1:0])

An analog-to-digital conversion is started as follows:

- I Select A/D converter clock from PWM clock and oscillation clock / 2. To select PWM clock, assign 1 to the ADIV bit in ADCON. Otherwise, assign 0 to the bit. If PWM clock frequency is the same with that of oscillator clock, don't select PWM clock.
- I Select either the A/D input high channel, ADCH[15:0], or the A/D input low channel, ADC[11:0], by assigning 1 or 0 to ADCHSEL[7].
- I Assign the channel selection data to ADCSEL[3:0] or ADCHSEL[3:0] in order to select one analog input channel among twenty eight analog pins (ADC_n, n = 0 ~ 11 and ADCH_m, m = 0 ~ 15).
- I Set the corresponding bit of ADCSELH, ADCSEL, ADCHH, or ADCHL to 1 in order to enable the selected analog input channel.
- I Set AD_EN bit and AD_REQ bit in ADCON register.

At first, A/D converter logic sets the successive approximation register to 000h. This register will be updated automatically for every conversion step. The successive approximation block performs 10-bit conversions for one input channel at a time. Either a low analog mux or a high analog mux is selected by the value of the CH_SEL bit in ADCHSEL. To select the low analog mux, assign 0 to the bit. Otherwise, assign 1 to it. Clearing a bit of ADCSEL or ADCSELH to 0 enables the corresponding low analog channel.

The input of the analog channel is selected by assign the corresponding value to ADCSEL[3:0]. Clearing a bit of ADCHL or ADCHH to 0 enables the corresponding high analog channel. The input of the analog channel is selected by assign the corresponding value to ADCHSEL[3:0].

To start the A/D conversion, set AD_EN bit (ADCON[7]) and AD_REQ bit (ADCON[6]) to 1. When a conversion is completed, the end-of-conversion (AD_END and ADCF) bits are automatically set to 1. Also AD_REQ bit is automatically cleared to 0 by hardware. AD_END bit is ADCON[5]. ADCF (ADCON[4]) is used for interrupt flag. Conversion result is sent to the ADCR[7:0] and ADCON[1:0] register. Then the A/D converter enters an idle state.

I ADCSEL (E2h): ADC Channel Selection Low & MUX Selection

Bit No.	7	6	5	4	3	2	1	0
	ADC3B	ADC2B	ADC1B	ADC0B	CH3	CH2	CH1	CH0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADCXB = 0 : ADCX Input Enable & Digital Input Disable
 CH[3:0]: ADC MUX Selection.

0000b = ADC0 Selection (0h)
 0001b = ADC1 Selection (1h)
 0010b = ADC2 Selection (2h)
 0011b = ADC3 Selection (3h)
 0100b = ADC4 Selection (4h)
 0101b = ADC5 Selection (5h)
 0110b = ADC6 Selection (6h)
 0111b = ADC7 Selection (7h)
 1000b = ADC8 Selection (8h)
 1001b = ADC9 Selection (9h)
 1010b = ADC10 Selection (Ah)
 1011b = ADC11 Selection (Bh)
 1100b = No ADC input select (Ch)
 1101b = No ADC input select (Dh)
 1110b = No ADC input select (Eh)
 1111b = No ADC input select (Fh, Default)

I ADCSELH (E1h): ADC Channel Selection High Register

Bit No.	7	6	5	4	3	2	1	0
	ADC11B	ADC10B	ADC9B	ADC8B	ADC7B	ADC6B	ADC5B	ADC4B
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADCXB = 0 : ADCX Input Enable & Digital Input Disable

I ADCHH (DAh): ADC Channel Selection High Register

Bit No.	7	6	5	4	3	2	1	0
	ADCH15B	ADCH14B	ADCH13B	ADCH12B	ADCH11B	ADCH10B	ADCH9B	ADCH8B
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADCHXB = 0 : ADCHX Input Enable & Digital Input Disable

I ADCHL (D9h): ADC Channel Selection Low Register

Bit No.	7	6	5	4	3	2	1	0
	ADCH7B	ADCH6B	ADCH5B	ADCH4B	ADCH3B	ADCH2B	ADCH1B	ADCH0B
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADCHXB = 0 : ADCHX Input Enable & Digital Input Disable

I ADCHSEL (DBh) : ADC High Channel Selection Register

Bit No.	7	6	5	4	3	2	1	0
	CH_SEL	-	-	-	CHH3	CHH2	CHH1	CHH0
	R/W(0)				R/W(0)	R/W(0)	R/W(0)	R/W(0)

CH_SEL: ADC MUX Selector with CHH[3:0] & CH[3:0].

0 = CH[3:0] ◇ ADC[11:0] Enable / ADCH[15:0] Disable (Default)

1 = CHH[3:0] ◇ ADC[11:0] Disable/ ADCH[15:0] Enable

CHH[3:0]: ADC MUX Selection for High Channel

- | | |
|------------------------------|-------------------------------|
| 0000b = ADCH0 Selection (0h) | 1000b = ADCH8 Selection (8h) |
| 0001b = ADCH1 Selection (1h) | 1001b = ADCH9 Selection (9h) |
| 0010b = ADCH2 Selection (2h) | 1010b = ADCH10 Selection (Ah) |
| 0011b = ADCH3 Selection (3h) | 1011b = ADCH11 Selection (Bh) |
| 0100b = ADCH4 Selection (4h) | 1100b = ADCH12 Selection (Ch) |
| 0101b = ADCH5 Selection (5h) | 1101b = ADCH13 Selection (Dh) |
| 0110b = ADCH6 Selection (6h) | 1110b = ADCH14 Selection (Eh) |
| 0111b = ADCH7 Selection (7h) | 1111b = ADCH15 Selection (Fh) |

I ADCON (EFh) : ADC Control & ADC Result Low Register: SAR[1:0]

Bit No.	7	6	5	4	3	2	1	0
	AD_EN	AD_REQ	AD_END	ADCF	-	ADIV	SAR1	SAR0
	R/W(0)	R/W(0)	R(1)	R/W(0)		R/W(0)	R/W(0)	R/W(0)

AD_EN: AD Conversion Enable.

AD_REQ: AD Conversion Request.

Cleared by H/W when AD_END goes to 1 from 0.

AD_END: Current ADC Status.

0 = ADC is running now.
 User must check the ADCF instead of AD_END.

ADCF: ADC Interrupt Flag.
 Must be cleared by S/W.

ADIV: ADC Input Clock (FADC) Select.
 0 = System Clock (FOSC) / 2. (Default)
 1 = PWM Input Clock (FPWM)
 PWM Clock for ADC should not be set to FOSC/1.

SAR[1:0]: Low Bits of ADC Result Value. (Total 10 bits)

I APCR (EEh): ADC Result High Register

Bit No.	7	6	5	4	3	2	1	0
	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Remember to read the conversion result before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

NOTE: Because the A/D converter does not contain sample-and-hold circuitry, it is important that any fluctuations in the analog voltage of the analog input pins should be kept in the absolute minimum range during a conversion procedure. Any change in the input voltage, perhaps due to circuit noise, will invalidate the result.

The A/D converter input pins are alternately used for digital input in P0 ~ P3. For the A/D conversion the input pins must be configured as analog input.

6.2.7.1 INTERNAL REFERENCE VOLTAGE LEVELS

In the A/D converter block diagram (Figure 6-19), the analog input voltage is compared to the reference voltage. The analog input voltage must be within the range from V_{SS} to V_{DD} .

The resistor tree generates different voltage reference internally during the analog-to-digital conversion process for each conversion step. The reference voltage level for the first bit conversion is always $1/2 V_{DD}$.

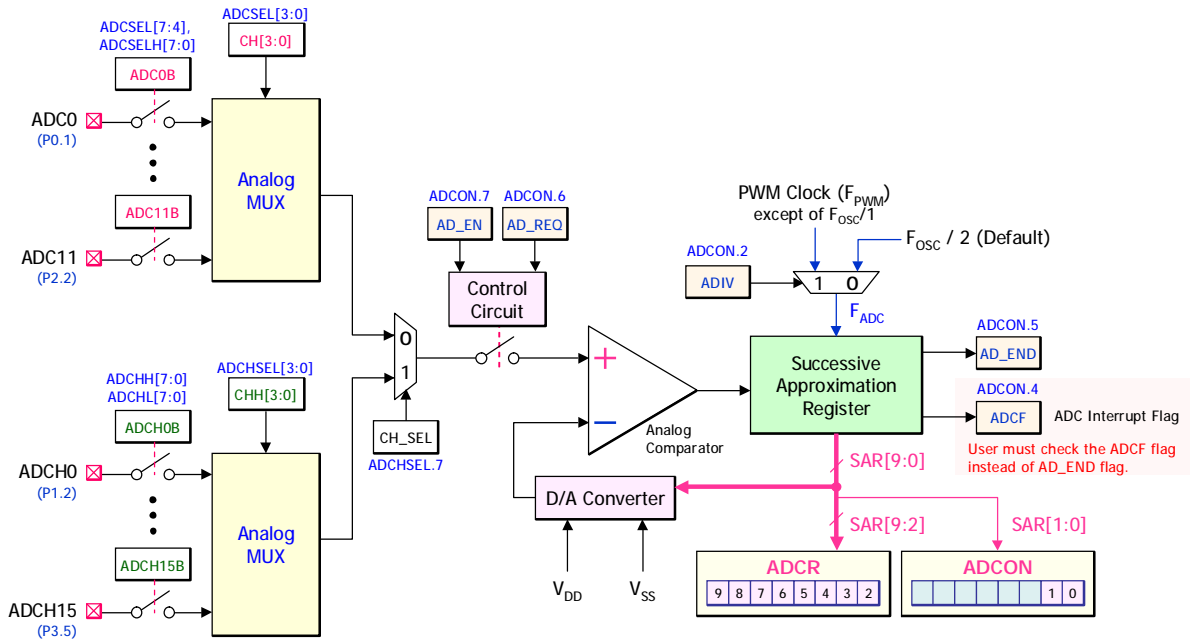


Figure 6-19 A/D Converter Block Diagram

6.2.7.2 CONVERSION TIME

The A/D conversion process requires 96 A/D conversion clock cycles.

- I Setup time: 8 cycles
- I Conversion time: 10bit x 8 cycles = 80 cycles
- I Hold time: 8 cycles

Each conversion time for each system clock frequency is presented in Table 6-10.

Table 6-10 Example of conversion time vs. clock frequency

System Clock (F_{OSC})	Divide (ADIV=0)	F_{ADC}	$T_{ADC}(1/F_{ADC})$	1 sample conversion time
20 MHz @5V	$F_{OSC}/2$	10 MHz	100 ns	9.6 us
10 MHz @5V	$F_{OSC}/2$	5 MHz	200 ns	19.2 us
10 MHz @3V	$F_{OSC}/2$	5 MHz	200 ns	19.2 us

5 MHz @3V	$F_{osc}/2$	2.5 MHz	400 ns	38.4 us
-----------	-------------	---------	--------	---------

6.2.7.3 INTERNAL A/D CONVERSION PROCEDURE

The A/D conversion sequences are as follows:

1. Analog input voltage must be in the voltage range between V_{SS} and V_{DD} .
2. Select either the low analog mux or the high analog mux by assigning the corresponding value to ADIV.
3. If $ADIV = 0$, enable a low analog input channel by setting the corresponding bit of $ADCSEL[7:4]$ and $ADCSELH[7:0]$ to 1. Select the enabled low analog input channel by assigning $ADCSEL[3:0]$.
4. If $ADIV=0$, enable a high analog input channel by setting the corresponding bit of $ADCHH[7:0]$ and $ADCHL[7:0]$ to 1. Select the enabled high analog input channel by assigning $ADCHSEL[3:0]$.
5. Enable A/D conversion by setting AD_EN to 1.
6. Start A/D conversion by setting AD_REQ to 1.
7. When the conversion is completed, AD_END and $ADCF$ will be set to 1.
8. Check AD_END or $ADCF$ flag to see if the conversion is finished or not.
9. The converted digital value is loaded to $ADCR[7:0]$ and $ADCON[1:0]$. Then, the A/D converter enters an idle state.
10. The A/D conversion result can now be read from the $ADCR$ and $ADCON$ register.

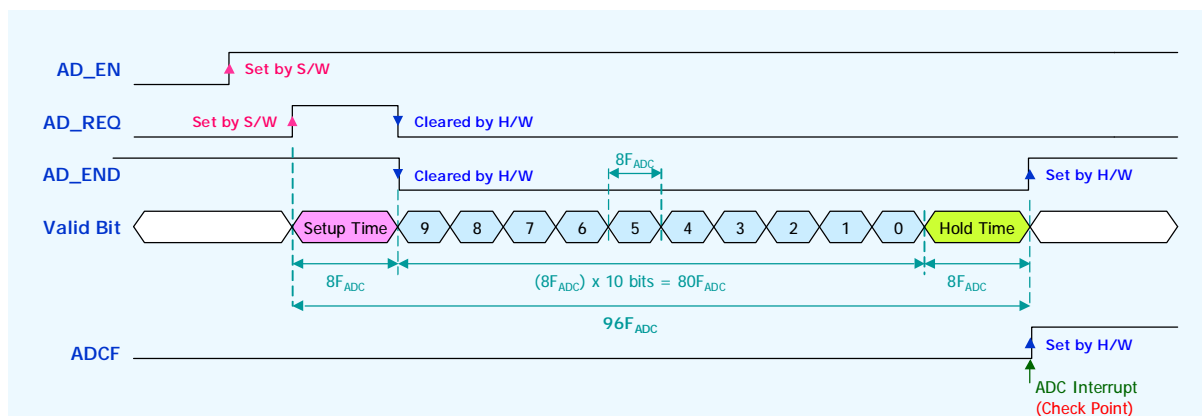


Figure 6-20 A/D Converter Timing Diagram

6.2.8 I²C Serial I/O

The I²C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the bus are:

- I Bidirectional data transfer between masters and slaves
- I Multimaster bus (no central master)
- I Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- I Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- I The I²C bus may be used for test and diagnostic purposes

In order to enable the I²C slave bus, the output latches of P2.2 and P2.3 must be set to logic 1. Their output type must be open-drain and their pull-ups switched on. Similarly, In order to enable the I²C master bus, the output latches of P0.3 and P0.4 must be set to logic 1. Their output type must be open-drain and their pull-ups switched on.

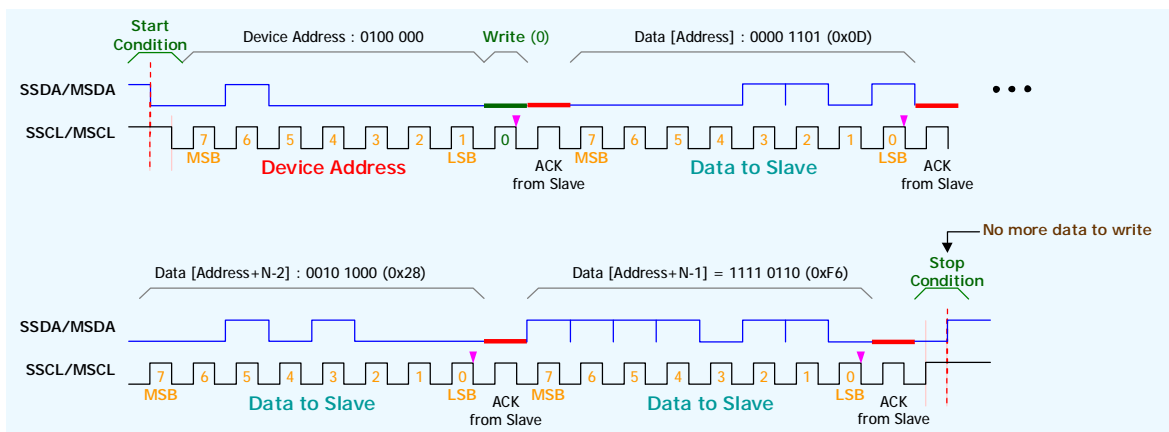
The MiDAS2.1 on-chip I²C logic provides a serial interface that meets the I²C bus specification and supports all transfer modes (other than the low-speed mode) from and to the I²C bus. The I²C logic handles bytes transfer autonomously. It also keeps track of serial transfers, and the control registers (I2C_SCON, I2C_MCON) reflect the status of the I²C logic and bus.

The CPU interfaces to the I²C logic via the following eleven function registers: I2C_SCON (I²C Slave Control Register), I2C_SDEV (I²C Slave Device Address Register), I2C_SADR (I²C Slave Memory Address Register), I2C_SDAT (I²C Slave Data Register), I2C_MDAT (I²C Master Data Register), I2C_MCON (I²C Master Control Register), I2C_MDEV (I²C Master Device Address Register), I2C_MADR (I²C Master Memory Address Register), I2C_MNUM (I²C Master Multi-byte Number Register), I2C_MSCL (I²C Master Clock Scale Factor Low Byte Register), and I2C_MSCH (I²C Master Clock Scale Factor High Byte Register). The I²C logic interfaces to the external I²C bus via two port 0 pins or two port 2 pins: P0.3/MSDA (master serial data line) and P0.4/MSCL (master serial clock line) or P2.2/SSDA (slave serial data line) and P2.3/SSCL (slave serial clock line).

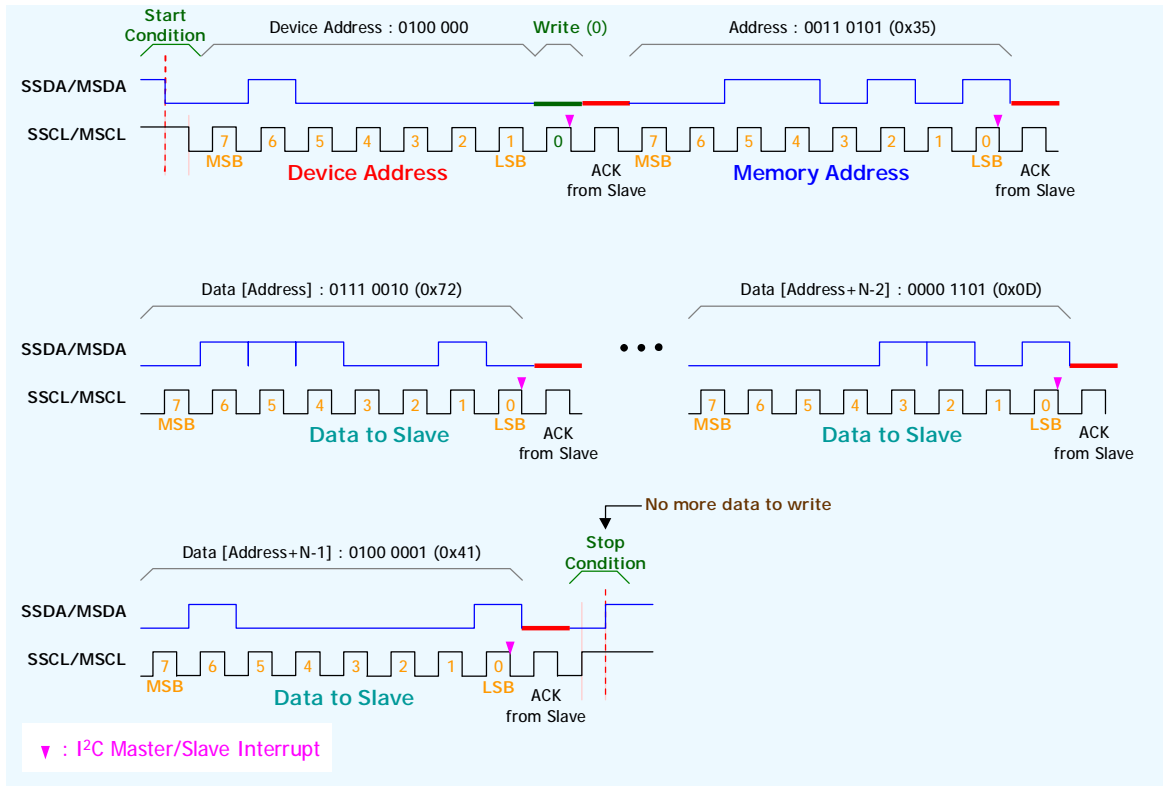
Figure 6-22 and Figure 6-21 show how a data transfer is accomplished on the bus. The first byte transmitted by the master is the slave device address. The byte consists of the 7 bit address and the final R/W bit. The RW bit signals the slave the data transfer direction. A master core treats the slave address transfer as any other write action. Store the slave device address in the I2C_MDAT register and clear the

OP bit. The master will then transfer the slave address on the bus. If you want to send the memory address, set I2C_MCON.1(MODE) and I2C_SCON.1(MODE). Then the memory address will be transmitted to the slave. If I2C_MCON.1(MODE) and I2C_SCON.1(MODE) are cleared, the memory address won't be transmitted. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I²C bus:

1. Figure 6-21: Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. The data direction bit is logic 0. If I2C_MCON.1(MODE) and I2C_SCON.1(MODE) is set, the memory address will be transmitted to the slave. Next follow a number of data bytes. The slave returns an acknowledge bit after each received byte.
2. Figure 6-22: Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. If you don't want to transmit the memory address, the data direction bit must be logic 1. Next follow the data bytes transmitted by the slave to the master. In order to transmit memory address, the data direction bit must be logic 0. Then the master transmits the memory address and generates the Restart condition. After that, the master transmits the slave address again. This time, the data direction bit must be logic 1. The slave returns an acknowledge bit after each received byte. Next follow the data bytes transmit by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a "not acknowledge" is returned.

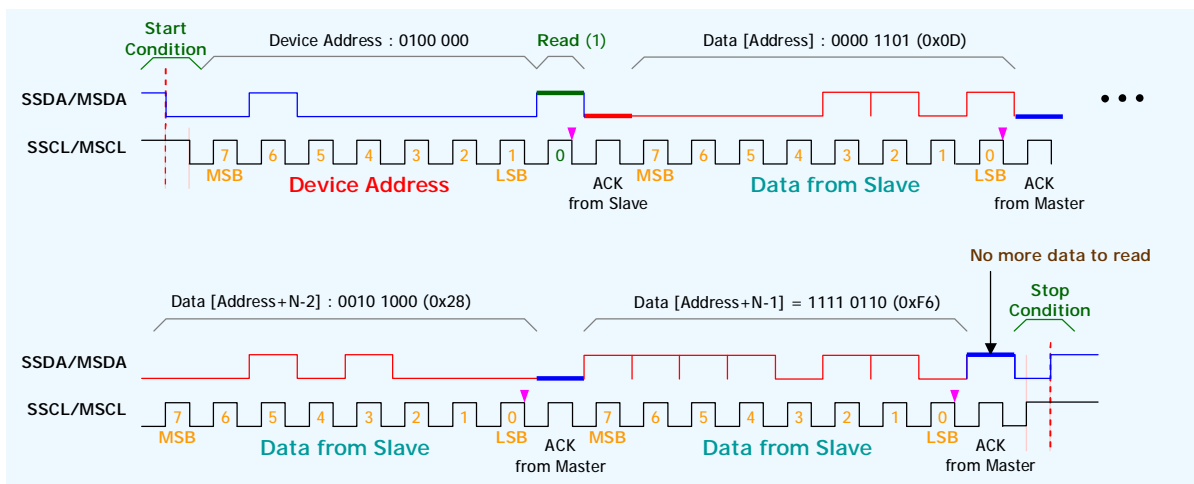


(a) No memory address

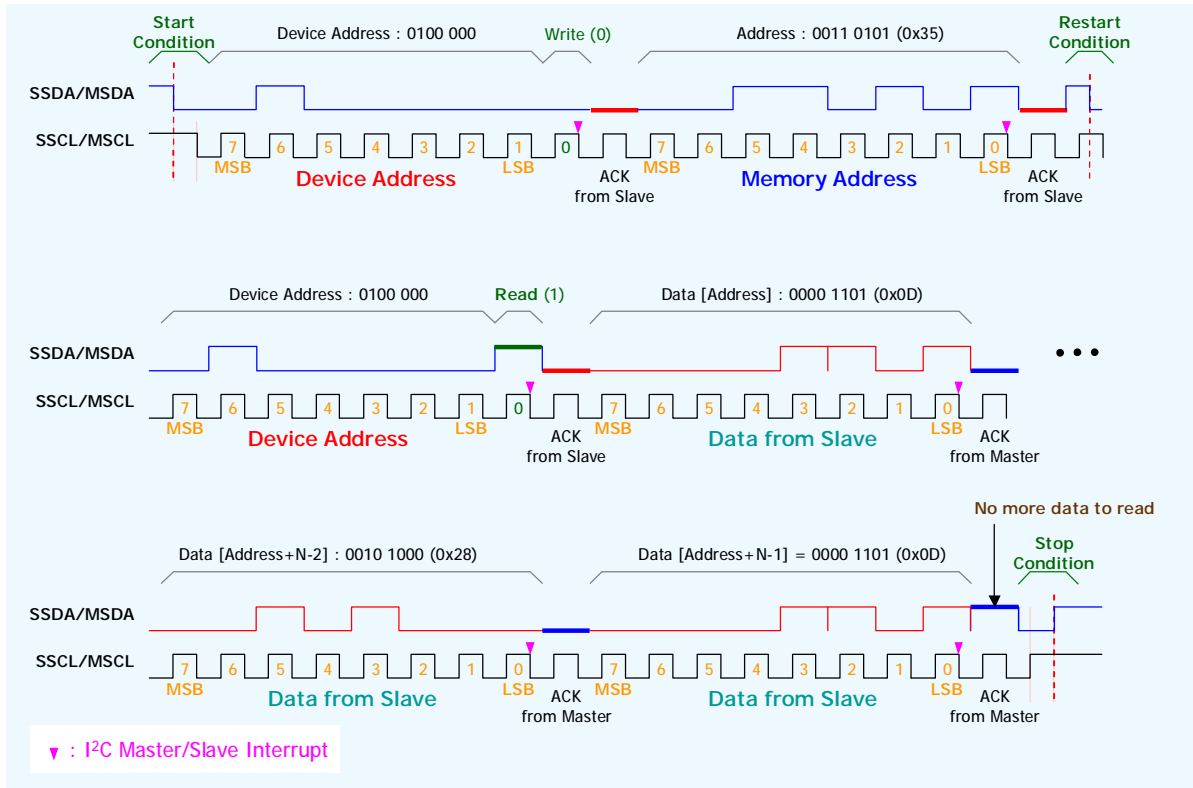


(b) Including the memory address

Figure 6-21 Data transfer including the memory address from a master transmitter to a slave receiver



(a) No memory address



(b) Including the memory address

Figure 6-22 Data transfer including the memory address from a slave transmitter to a master receiver

The master device generates all of the serial clock pulses and the Start, Restart, and Stop conditions. A transfer is ended with a Stop condition or with a Restart condition. Since a Restart condition is also the beginning of the next serial transfer, the I²C bus will not be released.

6.2.8.1 Mode of Operation

The on-chip I²C logic may operate in the following four modes:

1. Master Transmitter Mode:

Serial data output through P0.3/MSDA while P0.4/MSCL outputs the serial clock. The first transmitted byte contains the slave address of the receiving device (7 bits) and the data direction bit. In this case the data direction bit will be log 0, and we say that a “W” is transmitted. Thus the first transmitted byte is SLA+W. Serial data is transmitted 8 bits at a time. After each byte is transmitted,

an acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

2. Master Receiver Mode:

After a Start condition or a Restart condition, the first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this case the data direction bit will be logic 1, and we say that a "R" is transmitted. Thus the first transmitted byte is SLA+R. Serial data is received via P0.3/MSDA, while P0.4/MSCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an acknowledge bit is transmitted. Start, Restart and Stop conditions are output to indicate the beginning and the end of a serial transfer.

3. Slave Receiver Mode:

Serial data and the serial clock are received through P2.2/SSDA and P2.3/SSCL. After each byte is received, an acknowledge bit is transmitted. Start and Stop conditions are recognized as the beginning and the end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.

4. Slave Transmitter Mode:

After a Start condition or a Restart condition, the first byte is received and handled as a slave. However, in this mode, the direction bit will indicate that transfer direction is reversed. Serial data is transmitted via P2.2/SSDA while the serial clock is input through P2.3/SSCL. Start, Restart, and Stop conditions are recognized as the beginning and the end of a serial transfer.

The I²C may operate as a master and as a slave. In the slave mode, the I²C hardware looks for its own address and the general call address. If one of these addresses is detected, an interrupt is requested. When the microcontroller wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the I²C switches to the slave mode immediately and can detect its own slave addresses in the same serial transfer.

6.2.8.2 I²C operation

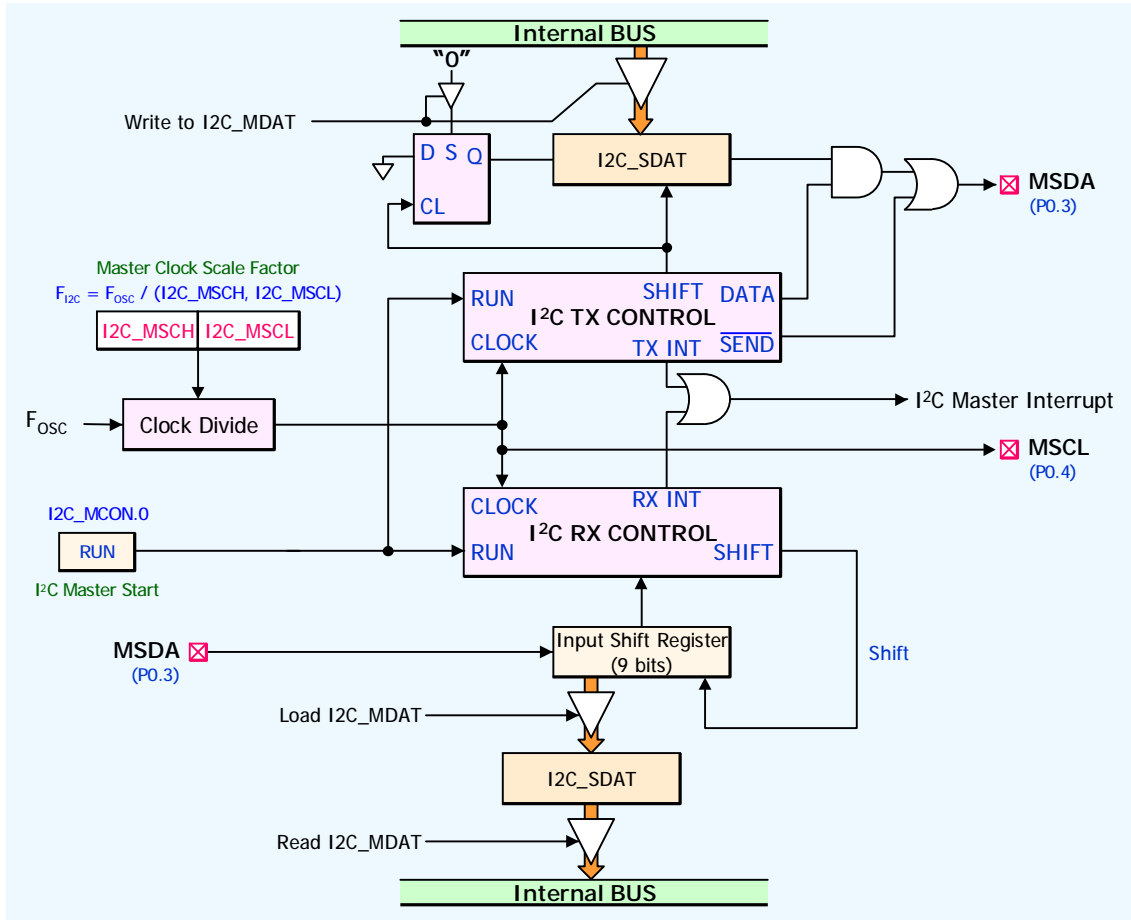


Figure 6-23 I²C master operation

Figure 6-23 shows how the I²C master operates. The maximum communication speed of the I²C master is 300 KHz (12 MHz, 3.3 V_{DD}). From now, its individual blocks are described.

I2C_MADR (A4h): I²C Master Memory Address Register

This 8-bit special function register may be loaded with the 8-bit memory address. When the I²C Master operates as a transmitter or receiver, the data is written to or read from the address location.

Bit No.	7	6	5	4	3	2	1	0
	MADR.7	MADR.6	MADR.5	MADR.4	MADR.3	MADR.2	MADR.1	MADR.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I2C_MDAT (9Eh): I²C Master Data Register

This 8-bit special function register contains a byte of serial data to be transmitted or a byte which has just been received. I2C_MDAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of I2C_MDAT. While data is being shifted out, data on the bus is simultaneously being shifted in; I2C_MDAT always contains the last byte present on the bus.

Bit No.	7	6	5	4	3	2	1	0
	MDAT.7	MDAT.6	MDAT.5	MDAT.4	MDAT.3	MDAT.2	MDAT.1	MDAT.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I2C_MSCH (A7h): I²C Master Clock Scale Factor High Byte Register

I2C_MSCL (A6h): I²C Master Clock Scale Factor Low Byte Register

These special function registers are used for I²C serial clock generation. The I²C serial clock frequencies are: $F_{OSC} / \{(I2C_MSCH, I2C_MSCL+1)*4\}$. The I²C serial clock pulses are generated only in the master mode.

I2C_MDEV (A3h): I²C Master Device Address Register

This 8-bit special function register may be loaded with the 7-bit master address (7 most significant bits) to which the I²C master will respond when programmed as a transmitter or receiver. The LSB is not used.

Bit No.	7	6	5	4	3	2	1	0
	MDEV.7	MDEV.6	MDEV.5	MDEV.4	MDEV.3	MDEV.2	MDEV.1	MDEV.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

MDEV[7:1]: I2C Master Device Address. MDEV[0]: Not Used. Don't Care.

I2C_MCON (A2h): I²C Master Control Register

This 5-bit special function register is used by the microcontroller to control the I²C Master functions: I²C Master interrupt flag, Transmitter/Receiver, Bypass, Including Memory Address/No Memory Address, Start/Stop.

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	I2C_MIF	OP	BYPASS	MODE	RUN
				R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I2C_MIF: I2C Master Interrupt Flag.

It is set each time a byte is received or transmitted.

OP: Cleared by SW.
 I2C Read/Write Operation.
 0 = Write Operation (Default).
 1 = Read Operation.
 BYPASS: Bypass Mode in I2C Master and Slave.
 MODE: I2C Master Mode.
 0 = Mode 0. Including Memory Address (Default).
 1 = Mode 1. No Memory Address.
 RUN: I2C Master Start.
 Cleared by H/W.

I2C_MNUM (A5h): I²C Master Multi-byte Number Register

This 8-bit special function register is load with the number of bytes to communicate. It is (I2CMNUM + 1).

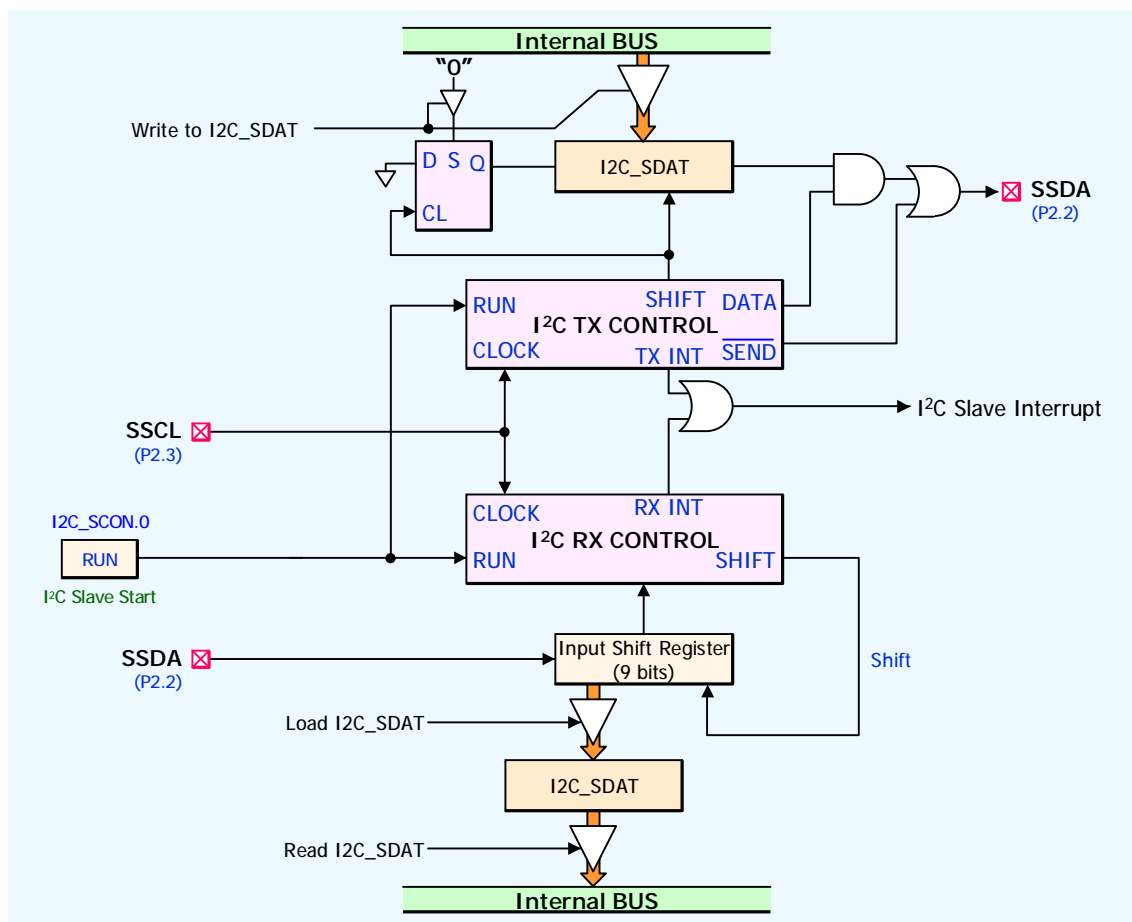


Figure 6-24 I²C slave operation

Figure 6-24 shows how the I²C slave operates. The I²C slave communication speed is from 74 to 317 KHz. (12 MHz, 3.3 V_{DD}). From now, its individual blocks are described.

I2C_SADR (9Ch): I²C Slave Memory Address Register

This 8-bit special function register may be loaded with the 8-bit memory address. When the I²C slave operates as a transmitter or receiver, the data is written to or read from the address location.

Bit No.	7	6	5	4	3	2	1	0
	SADR.7	SADR.6	SADR.5	SADR.4	SADR.3	SADR.2	SADR.1	SADR.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I2C_SDAT (9Dh): I²C Slave Data Register

This 8-bit special function register contains a byte of serial data to be transmitted or a byte which has just been received. I2C_SDAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of I2C_SDAT. While data is being shifted out, data on the bus is simultaneously being shifted in; I2C_SDAT always contains the last byte present on the bus.

Bit No.	7	6	5	4	3	2	1	0
	SDAT.7	SDAT.6	SDAT.5	SDAT.4	SDAT.3	SDAT.2	SDAT.1	SDAT.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I2C_SDEV (9Bh): I²C Slave Device Address Register

This 8-bit special function register may be loaded with the 7-bit slave address (7 most significant bits) to which the I²C slave will respond when programmed as a transmitter or receiver. The LSB is not used.

Bit No.	7	6	5	4	3	2	1	0
	SDEV.7	SDEV.6	SDEV.5	SDEV.4	SDEV.3	SDEV.2	SDEV.1	SDEV.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

SDEV[7:1]: I2C Slave Device Address. SDEV[0]: Not Used. Don't Care.

I2C_SCON (9Ah): I²C Slave Control Register

This 6-bit special function register is used by the microcontroller to control the I²C Master functions: Transmitter Operation, Receiver Operation, Current Status, I²C Slave interrupt flag, Including Memory Address/No Memory Address, Start/Stop.

Bit No.	7	6	5	4	3	2	1	0
	-	WR	RD	BUSY	-	I2C_SIF	MODE	RUN
		R(0)	R(0)	R(0)		R/W(0)	R/W(0)	R/W(0)

WR: I2C Write Operation Status. Cleared by H/W.

RD: I2C Read Operation Status. Cleared by H/W.

I2C_SIF: I2C Slave Interrupt Flag.

It is set each time a byte is received or transmitted.
Cleared by S/W.

MODE: I2C Slave Mode.

0 = Mode 0. Including Memory Address (Default).

1 = Mode 1. No Memory Address.

RUN: I2C Slave Start.

Cleared by H/W.

6.2.9 Interrupt

The MIDAS2.1 family has 13 interrupt sources with two priority levels. Each interrupt source has individual priority bits, a flag, an interrupt vector and an enable bit. All interrupts can be globally enabled or disabled.

I IE (A8h) : Interrupt Enable Register

Bit No.	7	6	5	4	3	2	1	0
	EA	EADC	-	ES	ET1	EX1	ET0	EX0
	R/W(0)	R/W(0)		R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

EA: Global interrupt enable.

EADC: ADC interrupt enable.

ES: Serial port interrupt enable.

ET1: Timer 1 interrupt enable.

EX1: External interrupt 1 enable.

ET0: Timer0 interrupt enable.

EX0: External interrupt 0 enable.

I IP (B8h) : Interrupt Priority Register

Bit No.	7	6	5	4	3	2	1	0
		PADC	-	PS	PT1	PX1	PT0	PX0
		R/W(0)		R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

PADC: ADC interrupt priority.

PS: Serial port interrupt priority.

PT1: Timer 1 interrupt priority.
 PX1: External interrupt 1 priority.
 PT0: Timer 0 interrupt priority.
 PX0: External interrupt 0 priority.

I EXIF (91h) : External Interrupt Flag Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	IE3	IE2	XT/RG	RGMD	RGSL	BGS
			R/W(0)	R/W(0)	R/W(0)	R(1)	R/W(0)	R/W(1)

IE3: External interrupt 3 flag. Cleared by SW.
 IE2: External interrupt 2 flag. Cleared by SW.

I EIE (E8h) : Extended Interrupt Enable Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	EPWM	EWDT	EI2C_S	EI2C_M	EX3	EX2
			R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

EPWM: PWM interrupt enable.
 EWDT: Watchdog interrupt enable.
 EI2C_S: I2C slave interrupt enable.
 EI2C_M: I2C master interrupt enable.
 EX3: External 3 interrupt enable.
 EX2: External 2 interrupt enable.

I EIP (F8h) : Extended Interrupt Priority Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	EPWM	EWDT	EI2C_S	EI2C_M	EX3	EX2
			R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

PPWM: PWM interrupt priority bit.
 PWDT: Watchdog timer interrupt priority bit.
 PI2C_S: I2C slave interrupt priority bit.
 PI2C_M: I2C master interrupt priority bit.
 PX3: External interrupt 3 priority bit.
 PX2: External interrupt 2 priority bit.

I TCON (88h) : Timer/Counter Control Register

Bit No.	7	6	5	4	3	2	1	0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

IE1: External Interrupt 1 Flag.
IT1: External Interrupt 1 Type Select Flag.
Edge Detect (IT1=1). Level Detect (IT1=0).
IE0: External Interrupt 0 Flag.
IT0: External Interrupt 0 Type Select Flag.
Edge Detect (IT0=1). Level Detect (IT0=0).

6.2.9.1 Interrupt Sources

The two external interrupts /INT0 and /INT1 can be either edge or level triggered, depending on bits IT0 and IT1 in the TCON register. The bits IE0 and IE1 in the TCON register are the interrupt request bits. The other two external interrupts of INT2, and /INT3 are only edge triggered. INT2 is positive edge triggered but /INT3 negative edge triggered. The negative interrupt input, /INT3, is sampled in every machine cycle. If the sampling result is high in one cycle and low in the next, a high to low transition is detected and the interrupt request flag IEx in TCON or EXIF is set. Similarly, a low-to-high transition on the positive interrupt inputs is detected and the interrupt request flag is set. Since the external interrupt inputs are sampled every machine cycle, they have to be held high or low for at least one complete machine cycle. The IEx in TCON is cleared automatically when the service routine is called. But, IEx in EXIF must be cleared by software. For a level triggered interrupt, the interrupt input has to be kept low till the interrupt is serviced. The request bits, IE0 and IE1, are not cleared by the hardware for the level triggered interrupt when the interrupt is serviced. If the interrupt input continues to be held low even after the service routine is completed, the processor may acknowledge another interrupt request from the same source. Note that the external interrupts, INT2 and /INT3, are only edge triggered. The individual interrupt flag corresponding to external interrupt 2 and 3 must be cleared manually by software.

The TF0 and TF1 flags generate the Timer 0 and 1 interrupts. These flags are set by the overflow in the Timer 0 and 1. The TF0 and TF1 flags are cleared automatically by the hardware when the timer interrupt is serviced.

The Watchdog timer can be used as a system monitor or a simple timer. In either case, when the time-out is reached, the Watchdog timer interrupt flag WDIF (WDCON.3) is set. If the enable bit EIE.4 enables the interrupt, it will occur.

The UART can generate interrupts on reception or transmission. There are two interrupt sources from the UART, which are obtained by the RI and TI bits in the SCON. This bit is not cleared automatically by the hardware, and software have to clear these bits.

ADCF flag and AD_END flag are set to 1 when A/D conversion is finished. Then the A/D conversion interrupt will be generated if it is enabled. The ADCF flag must be cleared by software in A/D conversion interrupt routine.

PFI flag is set when Vdd drops below the LVD interrupt level. The flag generates LVD interrupt if EPFI is

set to 1. This interrupt is not disabled by EA that is a global interrupt enable bit.

Each individual interrupt can be enabled or disabled by setting or clearing a bit in the IE register. IE also has a global enable/disable bit EA that can be cleared to disable all interrupts.

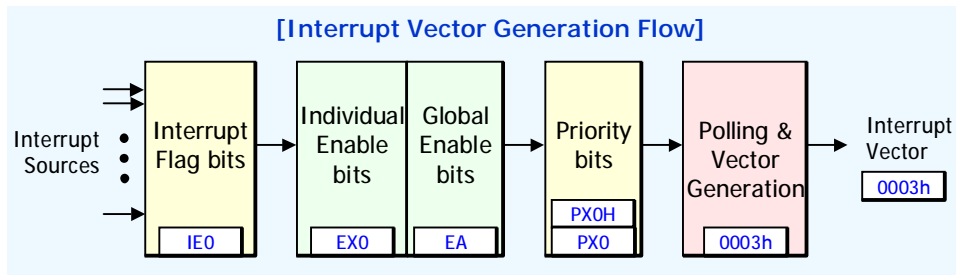


Figure 6-25 Interrupt Vector Generation Flow

6.2.9.2 Priority Level Structure

There are four or two priority levels for the interrupts. Naturally, a higher priority interrupt cannot be interrupted by a lower priority interrupt. However, there exists a pre-defined hierarchy amongst the interrupts themselves. This hierarchy comes into play when the interrupt controller has to resolve simultaneous requests having the same priority level. This hierarchy is defined as shown below; the interrupts are numbered starting from the highest priority to the lowest.

Table 6-11 Priority Structure of Interrupts

Hierarchy	Sources	Vector Address	Priority Level
1 (highest)	LVD	0033h	
2	/INT0	0003h	2 levels
3	TF0	000Bh	2 levels
4	/INT1	0013h	2 levels
5	TF1	001Bh	2 levels
6	RI + TI	0023h	2 levels
8	ADC	003Bh	2 levels
9	INT2	0043h	2 levels
10	/INT3	004Bh	2 levels
11	I ² C master	0053h	2 levels
12	I ² C slave	005Bh	2 levels
13	WDT	0063h	2 levels

14	PWM	006Bh	2 levels
----	-----	-------	----------

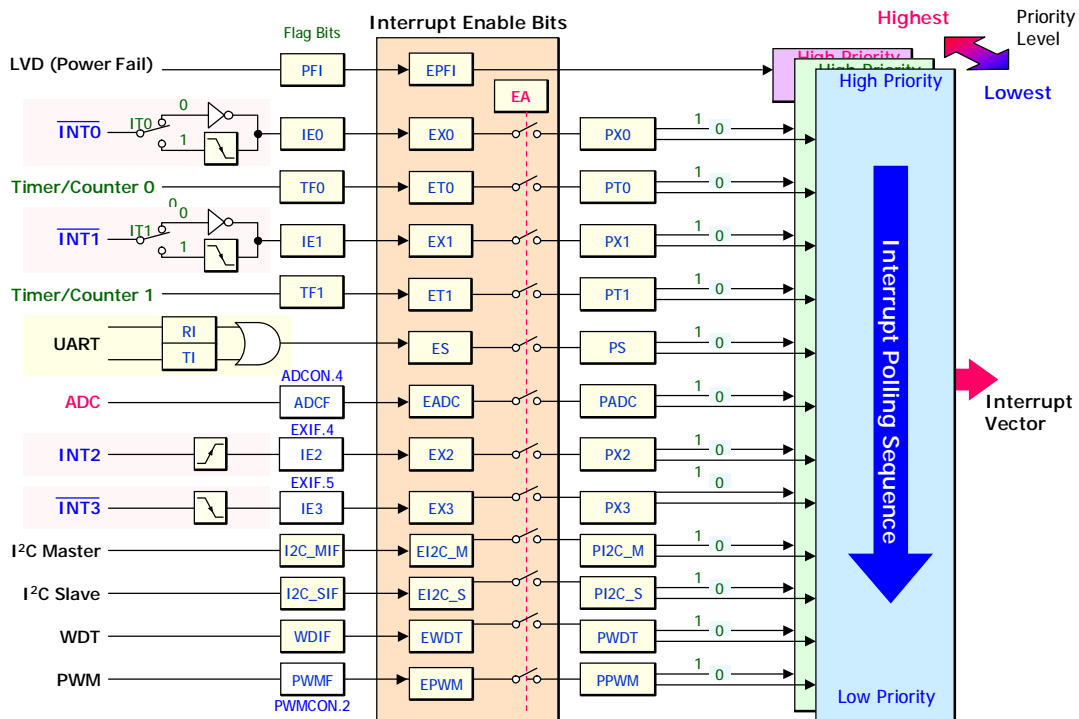


Figure 6-26 Hierarchy of Interrupt Priority

The interrupt flags are sampled at every machine cycle. In the same machine cycle, the sampled interrupts are polled and their priority is resolved. If certain conditions are met, the hardware will execute an internally generated LCALL instruction that will vector the process to the appropriate interrupt vector address. The conditions for generating the LCALL are

1. Neither an equal priority nor a higher priority interrupt is currently being serviced.
2. The current polling cycle is the last machine cycle of the currently executed instruction.
3. The instruction in progress is neither RETI nor any write to IP, IE, EIP, EIE, or EXIF.

If any of these conditions are not satisfied, the LCALL will not be generated. The polling cycle is repeated every machine cycle, with the interrupts sampled in the same machine cycle. If an interrupt flag is active but all of the above conditions are not satisfied, the interrupt is not serviced. Note that if its flag is not still active until all above conditions are satisfied, the interrupt will not be serviced. This means that active interrupt flags are not remembered; every polling cycle is new.

The processor responds to a valid interrupt by executing an LCALL instruction to call the appropriate

service routine. This may or may not clear the flag that caused the interrupt. In case of timer interrupts, the TF0 or TF1 flags are cleared by hardware whenever the processor vectors to the appropriate timer service routine. In case of external interrupts, /INT0 and /INT1, the flags are cleared only if they are edge triggered. In case of UART interrupts, the flags are not cleared by hardware. In the case of Timer 0 interrupt, the flags are not cleared by hardware. Watchdog timer interrupt flag WDIF have to be cleared by software. The hardware LCALL behaves exactly like the software LCALL instruction. This instruction saves the Program Counter value onto the stack, but does not save the PSW. The interrupt vector address is loaded into the Program Counter by the instruction LCALL.

Execution proceeds from the vector address to an RETI instruction. RETI instruction informs the processor that this interrupt routine ends. Then the instruction pops the top two bytes from the stack and reloads the value of the two bytes into the Program Counter. The interrupted program continues to be executed from where it left off. Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking that the service routine was still in progress.

6.2.9.3 Interrupt Response Time

The response time for each interrupt source depends on several factors, such as the feature of the interrupt and the executed instruction. The external interrupt inputs, /INT0 to /INT3, are sampled at S3 state of every machine cycle. After that, their interrupt flags IEx will be set. The Timer 0 and 1 overflow flags are set at S3 state of the machine cycle in which overflow has occurred. These flag values are polled in the next machine cycle. If a request is active and all three conditions are satisfied, the LCALL instruction generated by hardware is executed. This LCALL takes four machine cycles to be completed. Thus there is a minimum of five complete machine cycles between activation of an external interrupt request and the beginning of execution of the service routine's first instruction.

A response time will be extended if any of the three conditions are not satisfied. If a higher or equal priority level is already in progress, the additional wait time obviously depends on the nature of the currently executed service routine. If the polling cycle is not in the last machine cycle of the instruction in progress, then an additional delay is introduced. The maximum response time (if no other interrupt is in service) occurs if the MiDAS2.1 family is performing any write to IE, IP, EIE, EIP, or EXIF and then executes a 4-machine cycle instruction. From the time an interrupt source is activated, the longest reaction time is 11 machine cycles. This includes 1 machine cycle to detect the interrupt, 2 machine cycles to complete the IE, IP, EIE, EIP, or EXIF access, 4 machine cycles to complete the instruction and 4 machine cycles to complete the hardware LCALL to the interrupt vector location.

Thus in a single-interrupt system, the interrupt response time will always be more than 5 machine cycles and not more than 11 machine cycles. The maximum latency of 11 machine cycles is 44 clock cycles.

Note that in the standard 80C52, the maximum latency is 8 machine cycles that equals 96 clock cycles. This is more than 50% reduction in terms of clock periods.

6.2.10 Reset Circuit

Several hardware methods can cause MiDAS2.1 family to enter into reset state. Most register bits will be initialized to their reset values, but there are a few flag bits whose state depends on the reset method. These flags are used to find the cause of reset. Three methods of putting MiDAS2.1 family into reset state as shown in Figure 6-33: Power on/fail reset, External reset, and Watchdog reset.

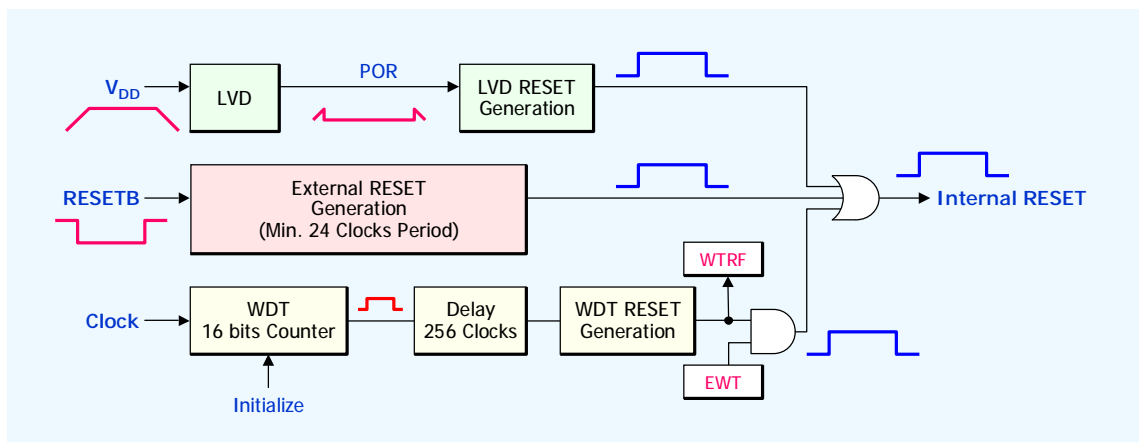


Figure 6-27 Three Reset Resources

6.2.10.1 Power On/Fail Reset

The MiDAS2.1 family has a band-gap voltage reference to recognize that V_{CC} is out of tolerance. When power is turned on, its internal circuit detects the rise of V_{CC} above V_{RST} , the reset threshold (2.0V) voltage. Once V_{CC} is above this voltage, its oscillator starts oscillation. Then its internal reset circuit waits for 50 ms until the supply voltage are stabilized. Next, MiDAS2.1 will exit the reset state. No external components are needed to generate a power-on-reset signal. During power-down or during a severe power glitch, if V_{CC} falls below V_{RST} , it will also set Power-On-Reset flag to high. The reset state is maintained as long as the power voltage remains below the threshold. This will occur automatically, needing no action from the user or from the software.

6.2.10.2 External Reset

The reset input is the RST pin. The external input signal is asynchronous to the internal clock. The RST pin is sampled during state S4 of every machine cycle. The RST pin must be held for at least 6 machine

cycles (24 clocks) to accomplish an external reset. The reset circuitry synchronously generates the internal reset signal. Thus the reset procedure operates synchronously, while the clock is running.

Once the device is in reset state, it will remain so as long as RST is 1. Even after RST is deactivated, the device will continue to be in reset state for up to three machine cycles, and then begin program execution from 0000h. There is no flag associated with the external reset condition. However, since the other two reset sources have flags, the external reset can be considered as the default reset if those two flags are cleared.

6.2.10.3 Watchdog Timer Reset

The Watchdog timer is a free running timer with programmable time-out intervals. The user can clear the watchdog timer at any time, causing it to restart the count. When the time-out interval is reached, an interrupt flag is set. If the Watchdog timer reset is enabled and the Watchdog timer is not cleared, then 256 clocks from the flag being set, the Watchdog timer will generate a reset. This reset condition is maintained by hardware for thirty clock cycles. Once the reset is removed the device will begin execution from 0000h.

6.2.11 Clock Circuits

The MiDAS2.1 family supports three clock sources as shown in Figure 6-28: 1) crystal oscillator, and 2) external oscillator module, 3) internal ring oscillator.

The MiDAS2.1 family has the on-chip oscillator circuitry, which is a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator. Software can turn off its crystal or external oscillator by writing a 1 to the XTOFF bit in PMR. To drive the processor with an external clock source, apply the external clock signal to XTAL1 and leave XTAL2 float, as shown in Figure 6-29. After the crystal oscillator is stabilized, XTUP in the STATUS register will be set to 1.

I PCON (87h) : Power Control Register

Bit No.	7	6	5	4	3	2	1	0
	SMOD1	-	-	POF	GF1	GF0	PD	IDL
	R/W(0)		-	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

PD : Power-down (Stop) mode enable.
IDL: IDLE Mode Enable.

I EXIF (91h) : External Interrupt Flag Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	IE3	IE2	XT/RG	RGMD	RGSL	BGS
			R/W(0)	R/W(0)	R/W(0)	R(1)	R/W(0)	R/W(1)

XT/RG: System clock selection.

0 = Internal Ring oscillator is selected as system clock.

1 = External clock is selected as system clock.

RGMD: Ring mode. Now system clock is Ring or XTAL.

Generally RGMD is the invert of XT/RG.

RGSL: Ring select bit when power-down wake-up.

1 = When wake-up from power-down mode in XTAL clock, use Ring oscillator as system clock during 65,536 XTAL clocks.

BGS: Band-gap select. (Default = 1)

0 = Band-gap block (LVD) will do not run in power-down mode, but function during normal mode.

It will support the significant power savings in power-down mode.

1 = Band-gap block (LVD) will run in power-down mode.

I PMR (C4h): Power Management Control Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	XTOFF	-	-	-
					R/W(0)	-	-	-

XTOFF: Internal amplifier disable for external crystal oscillator.

1 = External crystal will be killed.

0 = External crystal will run (Default).

Don't set XTOFF bit to 1 when XT/RG = 1.

I STATUS (C5h): Crystal Status Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	-
								R(0)

XTUP: Crystal oscillator warm-up status. (External crystal oscillator)

It represents the crystal clock is stable (1) or not (0).

Cleared by H/W when Power-on reset and all kinds of reset.

Cleared by H/W when XTOFF bit is set.

Cleared by during Power-down wake-up when XT/RG (EXIF.3) = 1.

Set by H/W after XTAL stabilization time.

The MiDAS2.1 family also contains the internal ring oscillator. Clearing the RINGON bit to 0 disables the ring oscillator. It generates 12 MHz clock when the operating voltage is 3.3V. The generated clock is prescaled by the bits DIV2, DIV1, and DIV0 in the OSCIN register. A user can select the crystal oscillator or the internal ring oscillator as the system clock with the bit XT/RG in EXIF.

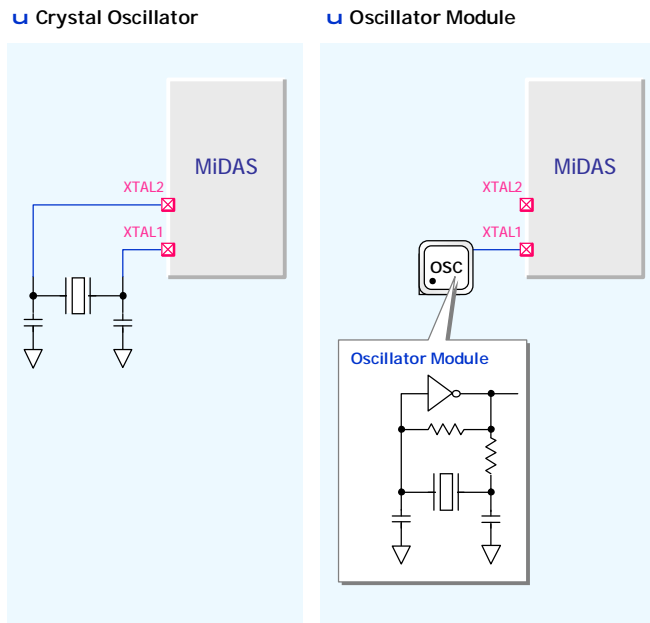


Figure 6-28 Clock Generators

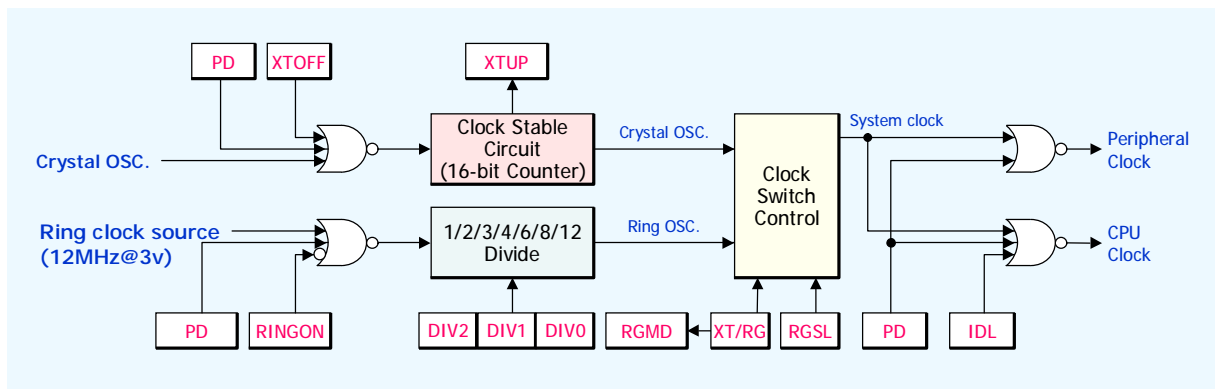


Figure 6-29 Clock Circuit

6.2.12 Power Management

The MiDAS2.1 family has two power saving features (POWER DOWN mode and IDLE mode) that help

one to reduce the power consumption of the device.

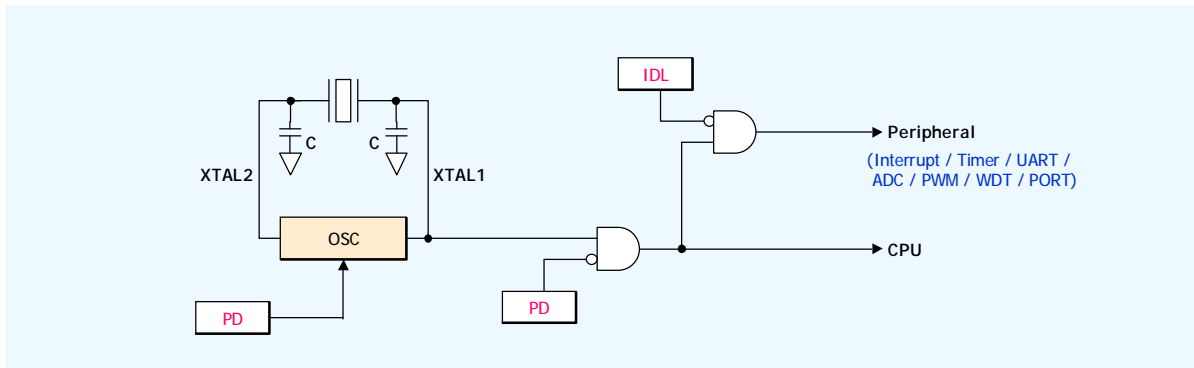


Figure 6-30 Power Management Circuit

6.2.12.1 IDLE Mode

Setting the IDL bit (PCON.0) to 1 puts the MiDAS2.1 family into idle mode. In the idle mode, the internal clock signal is gated off to the CPU, but not to the interrupt and peripherals such as Timers, Watchdog timer, and a serial port block. The CPU status is preserved in its entirety; the Program counter, the Stack Pointer, the Program Status Word, the Accumulator and the other registers maintain their data during the idle mode. The port pins hold the logical states they had at the time the IDLE state had started.

There are two ways to terminate the idle mode. Activation of any enabled interrupt will cause the IDL bit to be cleared by hardware, terminating the idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into idle mode.

The other way of terminating the idle mode is with all kinds of resets. The device can be put into the reset by applying a high on the external RST pin, a Power-On-Reset condition or a Watchdog timer time-out. The external reset pin has to hold high logic level for at least six machine cycles i.e. 24 clock periods to complete the reset. By the reset, the program counter is cleared to 0000h and all the SFRs are set to the default value. Since the clock has been running, without delay execution starts immediately. In the IDLE mode, the Watchdog timer continues to run, and if enabled, a time-out will cause a watchdog timer interrupt that will wake up the device. After the MiDAS2.1 family exits from the idle mode by a reset, the execution starts from address 0000h.

6.2.12.2 Power Down Mode

An instruction that sets the PD bit (PCON.1) to 1 causes MiDAS2.1 to enter into the Power Down mode. In this mode, all the clock circuits are stopped. With the clock frozen, all functions are stopped, but the on-chip data memory and SFRs are held. However, if EA(IE.7) = 1 and EWDT(EIE.4)=1, or if EWT(WDCON.1)=1, either a crystal oscillator or the internal ring oscillator operates in the power down mode. As a result, only the watchdog timer works and a little more power is consumed.

The MiDAS2.1 family can exit the Power Down mode with a hardware reset or the two external interrupts (/INT0 and /INT1). To wake it from the Power Down mode by an external interrupt, the interrupt has to be enabled (EA = 1, IE0 or IE1 = 1) and configured as level-sensitive (IT0 or IT1 = 0).

If EA(IE.7) = 1 and EWDT(EIE.4)=1, it can exit the Power Down mode with a watchdog reset. If EWT(WDCON.1)=1, it can be awakened by a watchdog interrupt. Reset redefines all the SFRs but does not change the on-chip data memory. An interrupt allows both the SFRs and the on-chip data memory to retain their value.

If the MiDAS2.1 family uses the crystal oscillator or the external oscillator for the system clock, the crystal stabilization time is needed. So, the external interrupt pin should be held to "0" for at least 65,536 clocks. After 65,536 clocks, system clock will be running normally and interrupt logic will start the external interrupt. Then the device will execute the interrupt service routine for the corresponding external interrupt. If the internal ring oscillator is used as the system clock, it will execute the external interrupt service routine immediately. An interrupt will be serviced, and following RETI the next to be executed will be the one following the instruction that put the device into the power down mode.

An external reset can be used to exit the Power Down state. The high on RST pin terminates the Power Down mode, and restarts the system clock. At that time, the crystal oscillator or the external oscillator needed clock stabilization time. The program execution will restart from 0000H.

6.2.13 Programming

The many MiDAS2.1 devices is shipped with the on-chip Flash memory array ready to be programmed.

6.2.13.1 Program Memory Lock Bits

A MiDAS2.1 device has six lock bits that can be left unprogrammed (1) or can be programmed (0) to obtain the protection features. When all the lock bits are left unprogrammed, no protection works.

Program code (Flash memory) and user data (EEPROM) can be protected by these lock bits. According to programmed lock bits, the six protection modes are specified in following table.

Table 6-12 Protection Modes

Level	Lock bits						Feature					
	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Flash		EEPROM		MDS	IAP
							Program	Read	Program	Read		
0	1	1	1	1	1	1	Enable	Enable	Enable	Enable	Enable	Enable
1	0	1	1	1	1	1	Disable	Enable	Enable	Enable	Enable	Enable
2	0	0	1	1	1	1	Disable	Disable	Enable	Enable	Enable	Enable
3	0	0	0	1	1	1	Disable	Disable	Disable	Enable	Enable	Enable
4	0	0	0	0	1	1	Disable	Disable	Disable	Disable	Enable	Enable
5	0	0	0	0	0	1	Disable	Disable	Disable	Disable	Disable	Enable
6	0	0	0	0	0	0	Disable	Disable	Disable	Disable	Disable	Disable

0: Programmed, 1: Unprogrammed, [Note] Security level 1 ~ 5 should be set after flash and code verification

6.2.13.2 In-System Programming (ISP)

The program code array (Flash 0000h ~ 1BFFh) and the user data array (EEPROM 1C00h ~ 1FFFh) can be programmed using the serial ISP interface through the EJTAG port in a target system. The EJTAG Port consists of the power supply pins and the serial ISP interface pins: the power supply pins (V_{DD} , V_{SS}) and the serial ISP interface pins (MDS_SCK, MDS_SDA, RESETB).

Command	Function
Blank	<ul style="list-style-type: none"> I Check the blank status of the device currently connected.
Erase Chip	<ul style="list-style-type: none"> I Performs an erase chip, the device's memory, both code and data. <ul style="list-style-type: none"> n Code: Flash n User data: EEPROM n Information data: Lock bits, RING option, PGM/ERS time (ISP) I The device memory will be blank and in a programmable state.
	<ul style="list-style-type: none"> I Reads in the device's memory I The results from the read are loaded into the ISP software's buffer and displayed on the screen.

Write code/EEPROM	<ul style="list-style-type: none"> I Writes all memory locations in the ISP software’s buffer out to the device’s memory.
Verify Chip	<ul style="list-style-type: none"> I Compares the ISP software buffer with the device’s internal memory.
	<ul style="list-style-type: none"> I If the buffers are found to be exact replicas of the device’s memory, a success result is returned.
	<ul style="list-style-type: none"> I If there are any differences, a failure result is returned along with the total number of mismatched bytes.

6.2.13.3 In-Application Programming (IAP)

The MiDAS2.1 family can store operation status/data in the EEPROM (user data memory) or update program code in the Flash (code memory). The code memory (7KB) can be programmed or erased during the execution of the program code. So can the user data memory (1KB). This is called “In-Application Programming (IAP).”

IAP can program code/data memory by the byte but erase them by the sector (256 byte). So IAP moves one sector data to the on-chip external data memory in order to erase only one byte. Then it updates the moved data and erases the target sector. Then, the updated data are moved to the erased sector again. IAP needs about 50 us to program one byte and tens of milliseconds to erase one sector.

The IAP routine begins at 3F00H. The IAP routine shows the execution result through the return value in the accumulator. 83h or 86h means ‘success’; FCh ‘program fail’; FDh ‘address fail’; FEh ‘lock fail’; FFh ‘command fail’. Before executing the IAP routine, the EAEN flag in the EEAEN register must to be set to 1. As a result of executing the IAP routine, the value of the PSW register can be changed. Any interrupt service routine will not be executed timely since the CPU is suspended for tens of milliseconds during executing the IAP routine.

Table 6-13 ISP command

I EEAEN(FFH): IAP Routine access enable register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	EAEN
								R/W(0)

The IAP sequences are as follows:

1. At first, back the special function registers up
2. IAP parameters must be set:
3. Set the EAEN flag to 1.
4. Call the IAP routine.
5. Clear the EAEN flag to 0.
6. Check the return value of the IAP routine.
7. Restore the special function registers.

7 Absolute Maximum Ratings

Table 7-1 Absolute Maximum Ratings

Items	Conditions	Ranges
Voltage on any pin relative to ground	-	-0.5V to ($V_{DD} + 0.5V$)
Voltage on V_{DD} relative to ground	-	-0.5V to 6.5V
Output Voltage	-	-0.5V to ($V_{DD} + 0.5V$)
Output Current High	One I/O pin active	-25mA
	All I/O pins active	-100mA
Output Current Low	One I/O pin active	+30mA
	All I/O pin active	+150mA
Operating Temperature		-40°C to +125°C
Storage Temperature		-65°C to +150°C
Soldering Temperature		160°C for 10 seconds

8 DC Characteristics

8.1 General DC Characteristics

Table 8-1 General DC Characteristics

($T_A = -40^\circ\text{C} \sim +125^\circ\text{C}$, $V_{DD} = 2.4\text{V} \sim 5.5\text{V}$ unless otherwise specified)

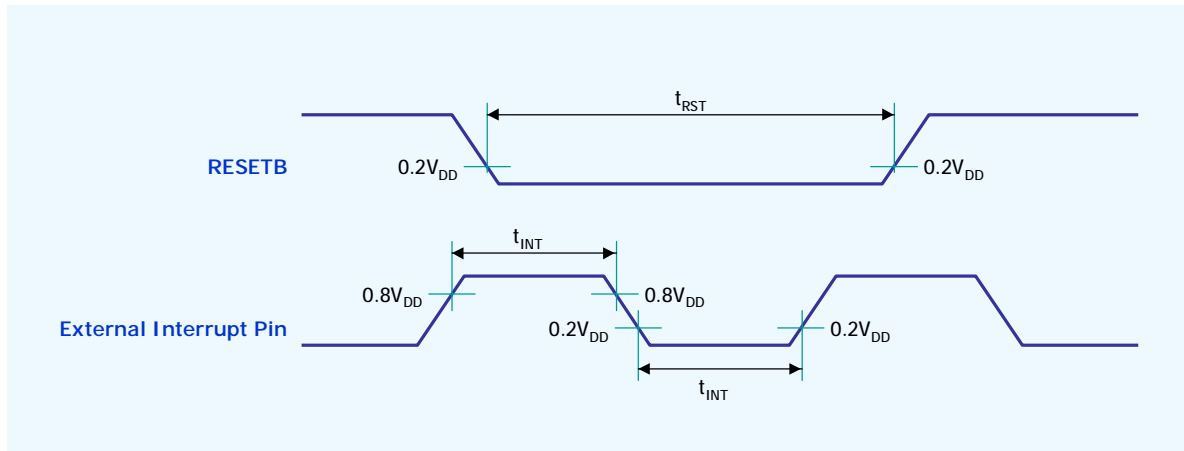
Parameter	Symbol	Pin	Conditions	Value			Unit
				Min.	Typ	Max.	
Input low voltage	V_{IL1}	RESETB,P0,P1,P2,P3	$V_{DD} = 2.4\text{V} \sim 5.5\text{V}$	-0.5	-	$0.2V_{DD} - 0.1$	V
	V_{IL2}	XTAL1,XTAL2		-0.5	-	$0.3V_{DD}$	
Input high voltage	V_{IH1}	RESETB,P0,P1,P2,P3	$V_{DD} = 2.4\text{V} \sim 5.5\text{V}$	$2.0V_{DD} + 1.0$	-	$V_{DD} + 0.5$	V
	V_{IH2}	XTAL1,XTAL2		$0.7V_{DD}$	-	$V_{DD} + 0.5$	
Output low voltage	V_{OL}	All pins	$I_{OL} = 20\text{mA} @ V_{DD} = 5\text{V}$	-	-	$0.3V_{DD}$	V
Output high voltage	V_{OH}	All pins	$I_{OH} = -15\text{mA} @ V_{DD} = 5\text{V}$ ($I_{OH} = -2.5\text{mA} @ V_{DD} = 2.6\text{V}$)	$0.7V_{DD}$	-	-	V
	V_{OHP}	Pull-up	$I_{OH} = -140\mu\text{A} @ V_{DD} = 5\text{V}$ ($I_{OH} = -20\mu\text{A} @ V_{DD} = 2.6\text{V}$)	$0.7V_{DD}$	-	-	
Input leakage current	I_{LI}	All pins except XTAL1,XTAL2	$V_{IN} = V_{IH}$ or V_{IL}	-	-	± 1	μA
Pin capacitance	C_{IO}	All	$V_{DD} = 5\text{V}$	-	10	-	pF

9 AC Characteristics

Table 9-1 AC Characteristics

($T_A = -20^\circ\text{C} \sim +85^\circ\text{C}$ unless otherwise specified)

Parameter	Symbol	Pin	Conditions	Value			Unit
				Min.	Typ.	Max.	
Operating Frequency	F_{OSC}	XTAL1, XTAL2	$V_{\text{DD}} = 5\text{V} \pm 10\%$	1	-	20	MHz
			$V_{\text{DD}} = 3\text{V} \pm 10\%$	1	-	12	
RESETB Input Width	t_{RST}	RESETB	$V_{\text{DD}} = 5\text{V} \pm 10\%$	24	-	-	F_{OSC}
			$V_{\text{DD}} = 3\text{V} \pm 10\%$	24	-	-	
External Interrupt Input Width	t_{INT}	External Interrupt	$V_{\text{DD}} = 5\text{V} \pm 10\%$	4	-	-	F_{OSC}
			$V_{\text{DD}} = 3\text{V} \pm 10\%$	4	-	-	

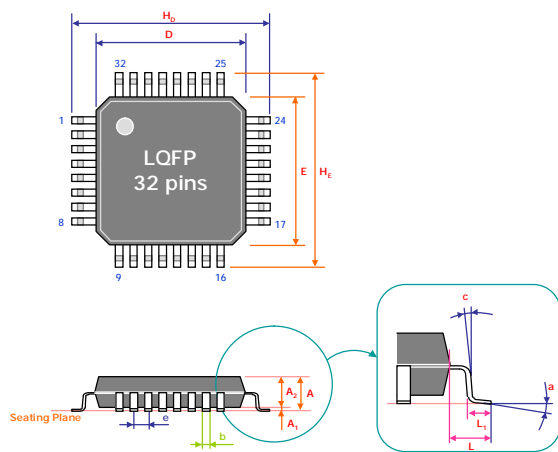


10 ADC Specifications

Table 10-1 ADC Specifications

Parameter	Symbol	Conditions	Value			Unit	
			Min.	Typ.	Max.		
Supply voltage	V_{DDADC}	-	2.7	-	5.5	V	
Input voltage	V_{INADC}	-	V_{SS}	-	V_{DD}	V	
Resolution	RES_{ADC}	-	-	10	-	bit	
Operating frequency	F_{ADC}	$V_{DD}=4.5\sim 5.5V$ $V_{DD}=2.7\sim 3.3V$		-	10 5	MHz	
Conversion time	t_{ADC}	-	-	$96/F_{ADC}$	-	s	
Overall accuracy	OA_{ADC}	$V_{DD}=5V, F_{ADC}=10MHz$ $V_{DD}=3V, F_{ADC}=5MHz$	-	± 2	± 4	LSB	
Integral nonlinearity	INL_{ADC}	$V_{DD}=5V, F_{ADC}=10MHz$ $V_{DD}=3V, F_{ADC}=5MHz$	-	± 2	± 4	LSB	
Differential nonlinearity	DNL_{ADC}	$V_{DD}=5V, F_{ADC}=10MHz$ $V_{DD}=3V, F_{ADC}=5MHz$	-	± 0.5	± 1	LSB	
Zero input error	ZIE_{ADC}	$V_{DD}=5V, F_{ADC}=10MHz$ $V_{DD}=3V, F_{ADC}=5MHz$	-	± 2	± 4	LSB	
Full scale error	FSE_{ADC}	$V_{DD}=5V, F_{ADC}=10MHz$ $V_{DD}=3V, F_{ADC}=5MHz$	-	± 2	± 4	LSB	
Analog input capacitance	C_{INADC}	-	-	10	15	pF	
ADC current	Active	I_{ADC}	$V_{DD}=5V, F_{ADC}=10MHz$	-	1	2	mA
			$V_{DD}=3V, F_{ADC}=5MHz$	-	0.3	0.6	
	Power down	$V_{DD}=5V$	-	-	-	100	nA

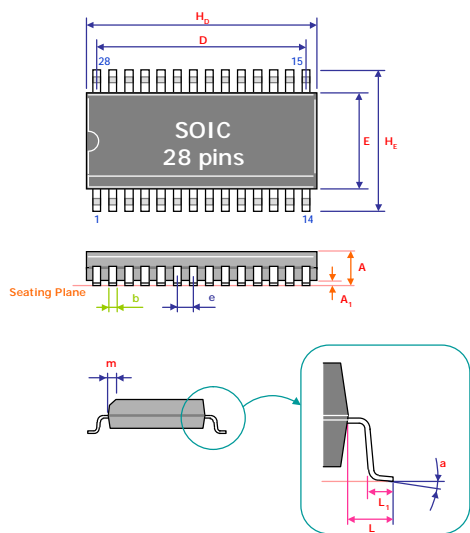
11 Package Dimension



Symbol	Dimension in Inches			Dimension in mm		
	Min.	Nom.	Max.	Min.	Nom.	Max.
A	-	-	0.063	-	-	1.60
A ₁	0.002	-	0.006	0.05	-	0.15
A ₂	0.053	-	0.057	1.35	-	1.45
b	0.012	0.015	0.018	0.30	0.38	0.45
D	0.272	0.276	0.280	6.90	7.00	7.10
E	0.272	0.276	0.280	6.90	7.00	7.10
e	0.0315 BSC			0.80 BSC		
H _D	0.344	0.354	0.364	8.75	9.00	9.25
H _F	0.344	0.354	0.364	8.75	9.00	9.25
L	-	0.039	-	-	1.00	-
L ₁	0.018	-	0.029	0.45	-	0.75
a	0°	-	7°	0°	-	7°
c	0°	-	-	0°	-	-

- Notes:
1. Dimension D * E do not include interlead flash.
 2. Dimension b, does not include dambar protrusion/intrusion.
 3. Controlling dimension: Inches.
 4. General appearance spec. should be based on final visual inspection spec.

Figure 11-1 LQFP 32-pin Package Dimension



Symbol	Dimension in Inches			Dimension in mm		
	Min.	Nom.	Max.	Min.	Nom.	Max.
A	0.093	0.099	0.104	2.35	2.45	2.65
A ₁	0.004	0.008	0.012	0.10	0.20	0.30
b	0.014	0.016	0.019	0.35	0.42	0.49
D	-	0.65	-	-	16.51	-
E	0.291	0.295	0.299	7.40	7.50	7.60
H _D	0.697	0.705	0.713	17.70	17.90	18.10
H _F	0.404	0.411	0.419	10.26	10.45	10.65
L	0.057	0.058	0.060	1.43	1.48	1.53
L ₁	0.034	0.038	0.042	0.86	0.96	1.07
a	0°	-	8°	0°	-	8°
e	0.050 BSC			1.27 BSC		
m	0.020	0.025	0.030	0.50	0.62	0.75

- Notes:
1. Dimension D Max. & S include mold flash or tie bar Burns.
 2. Dimension E, does not include interlead flash.
 3. Dimension D & E, include mold mismatch and are determined at the mold parting line.
 4. Dimension b, does not include dambar protrusion/intrusion.
 5. General appearance spec. should be based on final visual inspection spec.

Figure 11-2 SOIC 28-pin Package Dimension

12 Product Numbering System

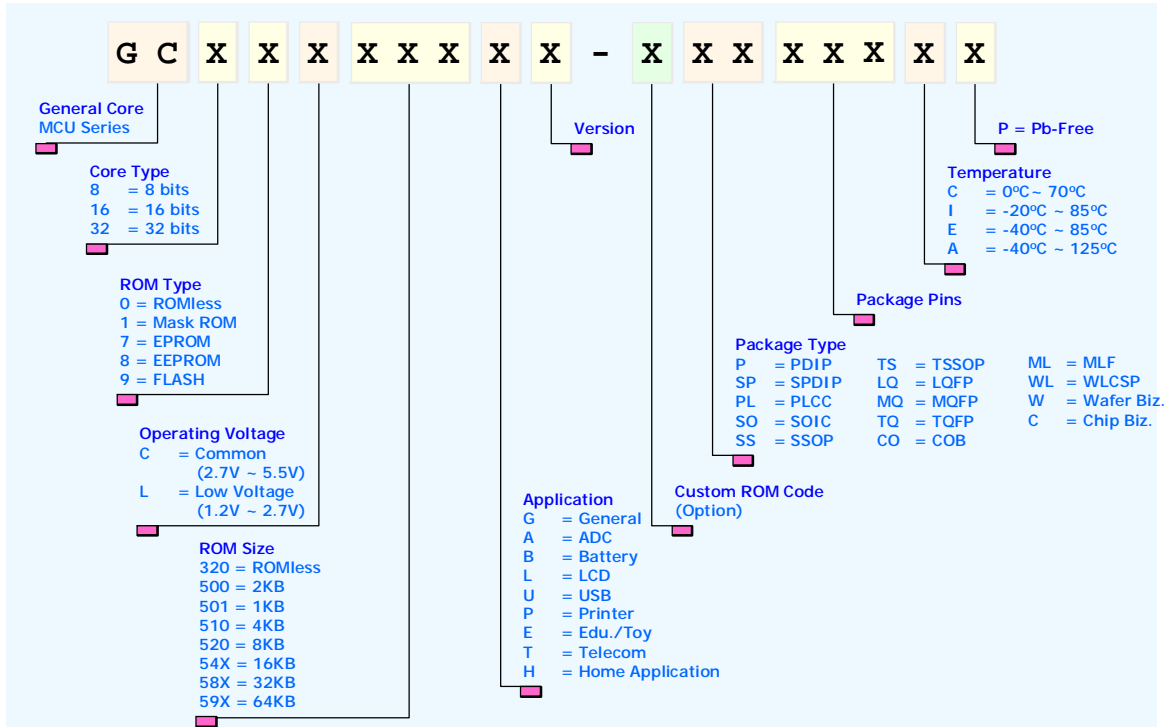


Figure 12-1 Product Numbering System

13 Appendix A: Instruction Set

Table 13-1 Note on instruction set and addressing modes

Notation	Description
Rn	Register R0-R7 of the currently selected Register Bank.
direct	8-bit internal data location's address. This could be an IRAM location (0-127) or a SFR (128-255).
@Ri	8-bit IRAM location (0-255) addressed indirectly through register R0 or R1.
#data	8-bit constant included in instruction
#data16	16-bit constant included in instruction
addr16	16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64Kbyte Program Memory address space.
Addr11	11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2Kbyte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.
bit	Direct addressed bit in IRAM or SFR.

ACALL **addr11**

Function: Absolute Call

Description: ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The called subroutine must therefore start within the same 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example: Initially SP equals 07h. The label “SUBRTN” is at program memory location 0345h. After executing the instruction,

ACALL SUBRTN

at location 0123h, SP will contain 09h, internal RAM locations 08h and 09h will contain 25h and 01h, respectively, and the PC will contain 0345h.

Bytes: 2

Cycles: 3

Encoding:

a10 a9 a8 1	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0
-------------	---------	-------------	-------------

Operation: ACALL
 $(PC) \leftarrow (PC) + 2$
 $(SP) \leftarrow (SP) + 1$
 $((SP)) \leftarrow (PC_{7-0})$
 $(SP) \leftarrow (SP) + 1$
 $((SP)) \leftarrow (PC_{15-8})$
 $(PC_{10-0}) \leftarrow \text{page address}$

ADD A, <src-byte>**Function:** Add

Description: ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3h (11000011b) and register 0 holds 0AAh (10101010b). The instruction,

```
ADD A, R0
```

will leave 6Dh (01101101b) in the Accumulator with the AC flag cleared and both the carry flag and OV set to 1.

ADD A, Rn**Bytes:** 1**Cycles:** 1

Encoding:

0	0	1	0	1	r	r	r
---	---	---	---	---	---	---	---

Operation: ADD
 $(A) \leftarrow (A) + (Rn)$

ADD A, direct**Bytes:** 2**Cycles:** 2

Encoding:

0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

direct address

Operation: ADD
 $(A) \leftarrow (A) + (\text{direct})$

ADD A, @Ri**Bytes:** 1**Cycles:** 1**Encoding:**

0	0	1	0	0	1	1	i
---	---	---	---	---	---	---	---

Operation: ADD
 $(A) \leftarrow (A) + ((Ri))$ **ADD A, #data****Bytes:** 2**Cycles:** 2**Encoding:**

0	0	1	0	0	1	0	0	immediate data
---	---	---	---	---	---	---	---	----------------

Operation: ADD
 $(A) \leftarrow (A) + \#data$

Preliminary

ADDC A, <src-byte>

Function: Add with Carry

Description: ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3h (11000011b) and register 0 holds 0AAh (10101010b) with the carry flag set. The instruction,

ADDC A, R0

will leave 6Eh (01101110b) in the Accumulator with AC cleared and both the carry flag and OV set to 1.

ADDC A, Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	1	r	r	r
---	---	---	---	---	---	---	---

Operation: ADDC
 $(A) \leftarrow (A) + (C) + (Rn)$

ADDC A, direct

Bytes: 2

Cycles: 2

Encoding:

0	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

direct address

Operation: ADDC
 $(A) \leftarrow (A) + (C) + (Rn)$

ADDC A, @Ri**Bytes:** 1**Cycles:** 1**Encoding:**

0	0	1	1	0	1	1	i
---	---	---	---	---	---	---	---

Operation: ADDC
 $(A) \leftarrow (A) + (C) + ((Ri))$ **ADDC A, #data****Bytes:** 2**Cycles:** 2**Encoding:**

0	0	1	1	0	1	0	0	immediate data
---	---	---	---	---	---	---	---	----------------

Operation: ADDC
 $(A) \leftarrow (A) + (C) + \#data$

Preliminary

AJMP **addr11**

Function: Absolute Jump

Description: AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (after incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2K block of program memory as the first byte of the instruction following AJMP.

Example: The label "JMPADR" is at program memory location 0123h. The instruction, AJMP JMPADR is at location 0345h and will load the PC with 0123h.

Bytes: 2

Cycles: 3

Encoding:

a10 a9 a8 0	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0
-------------	---------	-------------	-------------

Operation: AJMP
 $(PC) \leftarrow (PC) + 2$
 $(PC_{10-0}) \leftarrow \text{page address}$

ANL <dest-byte>, <src-byte>

Function: Logical-AND for byte variables

Description: ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register, indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch not the input pins.

Example: If the Accumulator holds 0C3h (11000011b) and register 0 holds 55h (0101010b) then the instruction,

ANL A, R0

will leave 41h (01000001b) in the Accumulator.

When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The instruction,

ANL P1, # 01110011b

will clear bits 7, 3, and 2 of output port 1.

ANL A, Rn

Bytes: 1

Cycles: 1

Encoding:

0	1	0	1	1	r	r	r
---	---	---	---	---	---	---	---

Operation: ANL
 $(A) \leftarrow (A) \wedge (Rn)$

ANL A, direct

Bytes: 2

Cycles: 2

Encoding:

0	1	0	1	0	1	0	1
---	---	---	---	---	---	---	---

direct address

Operation: ANL
 $(A) \leftarrow (A) \wedge (\text{direct})$

ANL A, @Ri**Bytes:** 1**Cycles:** 1**Encoding:**

0 1 0 1	0 1 1 i
---------	---------

Operation: ANL
 $(A) \leftarrow (A) \wedge ((Ri))$ **ANL A, #data****Bytes:** 2**Cycles:** 2**Encoding:**

0 1 0 1	0 1 0 0
---------	---------

immediate data

Operation: ANL
 $(A) \leftarrow (A) \wedge \#data$ **ANL direct, A****Bytes:** 2**Cycles:** 2**Encoding:**

0 1 0 1	0 0 1 0
---------	---------

direct address

Operation: ANL
 $(direct) \leftarrow (direct) \wedge (A)$ **ANL direct, #data****Bytes:** 3**Cycles:** 3**Encoding:**

0 1 0 1	0 0 1 1
---------	---------

direct address

immediate data

Operation: ANL
 $(direct) \leftarrow (direct) \wedge \#data$

ANL C, <src-bit>

Function: Logical-AND for bit variables

Description: If the boolean value of the source bit is a logical 0 then clear the carry flag; otherwise leave the carry flag in its current state. A slash (“/”) preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Only direct addressing is allowed for the source operand.

Example: Set the carry flag if, and only if, P1.0 = 1, ACC. 7 = 1, and OV = 0:

```

MOV    C, P1.0      ; LOAD CARRY WITH INPUT PIN STATE
ANL    C, ACC.7     ; AND CARRY WITH ACCUM. BIT 7
ANL    C, /OV       ; AND WITH INVERSE OF OVERFLOW FLAG
    
```

ANL C, bit

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 1 0	bit address
---------	---------	-------------

Operation: ANL
 $(C) \leftarrow (C) \wedge (\text{bit})$

ANL C, /bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 1	0 0 0 0	bit address
---------	---------	-------------

Operation: ANL
 $(C) \leftarrow (C) \wedge \sim(\text{bit})$

CJNE <dest-byte>, <src-byte>, rel

Function: Compare and Jump if Not Equal.

Description: CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction.
The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example: The Accumulator contains 34h. Register 7 contains 56h. The first instruction in the sequence,

```

                CJNE  R7, #60h, NOT_EQ
;              .....
NOT_EQ:        JC    REQ_LOW
;              .....
                ; R7 = 60h.
                ; IF R7 < 60h.
                ; R7 > 60h.
    
```

sets the carry flag and branches to the instruction at label NOT_EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60h.

If the data being presented to Port 1 is also 34h, then the instruction,

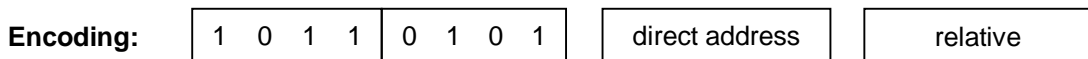
```
WAIT:          CJNE  A, P1, WAIT
```

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34h.)

CJNE A, direct, rel

Bytes: 3

Cycles: 4



Operation:
(PC) ← (PC) + 3
IF (A) <> (direct)
THEN (PC) ← (PC) + relative offset
IF (A) < (direct)
THEN (C) ← 1
ELSE (C) ← 0

CJNE A, #data, rel
Bytes: 3

Cycles: 4

Encoding:

1	0	1	1	0	1	0	0
---	---	---	---	---	---	---	---

immediate data

relative

Operation:

```

(PC) ← (PC) + 3
IF (A) <> data
THEN (PC) ← (PC) + relative offset
IF (A) < data
THEN (C) ← 1
ELSE (C) ← 0
    
```

CJNE Rn, #data, rel
Bytes: 3

Cycles: 4

Encoding:

1	0	1	1	1	r	r	r
---	---	---	---	---	---	---	---

immediate data

relative

Operation:

```

(PC) ← (PC) + 3
IF (Rn) <> data
THEN (PC) ← (PC) + relative offset
IF (Rn) < data
THEN (C) ← 1
ELSE (C) ← 0
    
```

CJNE @Ri, #data, rel
Bytes: 3

Cycles: 4

Encoding:

1	0	1	1	0	1	1	i
---	---	---	---	---	---	---	---

immediate data

relative

Operation:

```

(PC) ← (PC) + 3
IF ((Ri)) <> data
THEN (PC) ← (PC) + relative offset
IF ((Ri)) < data
THEN (C) ← 1
ELSE (C) ← 0
    
```

CLR A

Function: Clear Accumulator

Description: The Accumulator is cleared (all bits set on zero). No flags are affected.

Example: The Accumulator contains 5Ch (01011100b). The instruction, CLR A will leave the Accumulator set to 00h (00000000b).

Bytes: 1

Cycles: 1

Encoding:

1	1	1	0	0	1	0	0
---	---	---	---	---	---	---	---

Operation: CLR
 $A \leftarrow 0$

CLR bit

Function: Clear bit

Description: The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Example: Port 1 has previously been written with 5Dh (01011101b). The instruction, CLR P1.2 will leave the port set to 59h (01011001b).

CLR C

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	0	0	1	1
---	---	---	---	---	---	---	---

Operation: CLR
(C) ← 0

CLR bit

Bytes: 2

Cycles: 2

Encoding:

1	1	0	0	0	0	1	0
---	---	---	---	---	---	---	---

bit address

Operation: CLR
(bit) ← 0

Preliminary

CPL A

Function: Complement Accumulator

Description: Each bit of the Accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to a zero and vice-versa. No flags are affected.

Example: The Accumulator contains 5Ch (01011100b). The instruction, CPL A will leave the Accumulator set to 0A3h(10100011b).

Bytes: 1

Cycles: 1

Encoding:

1	1	1	1	0	1	0	0
---	---	---	---	---	---	---	---

Operation: CPL
(A) ← ~(A)

CPL bit

Function: Complement bit

Description: The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CPL can operate on the carry or any directly addressable bit.

Note: When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, not the input pin.

Example: Port 1 has previously been written with 5Bh (01011101b). The instruction sequence,

CPL P1.1

CPL P1.2

will leave the port set to 5Bh (01011011b).

CPL C

Bytes: 1

Cycles: 1

Encoding:

1	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Operation: CPL
(C) ← ~(C)

CPL bit

Bytes: 2

Cycles: 2

Encoding:

1	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

bit address

Operation: CPL
(bit) ← ~(bit)

DA A

Function: Decimal-adjust Accumulator for Addition

Description: DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx-1111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carryout of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00h, 06h, 60h, or 66h to the Accumulator, depending on initial Accumulator and PSW conditions.

Note: DA A cannot simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DA A apply to decimal subtraction.

Example: The Accumulator holds the value 56h (01010110b) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67h (01100111b) representing the packed BCD digits of the decimal number 67. The carry flag is set. The instruction sequence.

```
ADDC  A, R3
DA    A
```

will first perform a standard two's-complement binary addition, resulting in the value 0BEh (10111110b) in the Accumulator. The carry and auxiliary carry flags will be cleared.

The Decimal Adjust instruction will then alter the Accumulator to the value 24h (00100100b), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag will be set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01h or 99h. If the Accumulator initially holds 30h (representing the digits of 30 decimal), then the instruction sequence,

```
ADD  A, #99h
DA   A
```

will leave the carry set and 29h in the Accumulator, since $30 + 99 = 129$. The low-order byte of the sum can be interpreted to mean $30 - 1 = 29$.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	0	0
---	---	---	---	---	---	---	---

Operation: DA
- contents of Accumulator are BCD
IF $\{[(A_{3-0}) > 9] \vee [(AC) = 1]\}$
THEN $(A_{3-0}) \leftarrow (A_{3-0}) + 6$
AND
IF $\{[(A_{7-4}) > 9] \vee [(C) = 1]\}$
THEN $(A_{7-4}) \leftarrow (A_{7-4}) + 6$

Preliminary

DEC byte

Function: Decrement

Description: The variable indicated is decremented by 1. An original value of 00h will underflow to 0FFh.
No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: Register 0 contains 7Fh (01111111b). Internal RAM locations 7Eh and 7Fh contain 00h and 40h, respectively. The instruction sequence

DEC @R0

DEC R0

DEC @R0

will leave register 0 set to 7Eh and internal RAM locations 7Eh and 7Fh set to 0FFh and 3Fh.

DEC A

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

Operation: DEC
 $(A) \leftarrow (A) - 1$

DEC Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	1	r	r	r
---	---	---	---	---	---	---	---

Operation: DEC
 $(Rn) \leftarrow (Rn) - 1$

DEC direct**Bytes:** 2**Cycles:** 2**Encoding:**

0	0	0	1	0	1	0	1
---	---	---	---	---	---	---	---

direct address

Operation: DEC
 $(\text{direct}) \leftarrow (\text{direct}) - 1$ **DEC @Ri****Bytes:** 1**Cycles:** 1**Encoding:**

0	0	0	1	0	1	1	i
---	---	---	---	---	---	---	---

Operation: DEC
 $((Ri)) \leftarrow ((Ri)) - 1$

Preliminary

DEC DPTR

Function: Decrement Data Pointer

Description: Decrement the 16-bit data pointer by 1. A 16-bit decrement (modulo 2^{16}) is performed; an underflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will decrement the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be decremented.

Example: Registers DPH and DPL contain 12h and 01h, respectively. The instruction sequence,

```
DEC DPTR
DEC DPTR
DEC DPTR
```

will change DPH and DPL to 11h and 0FEh.

Bytes: 1

Cycles: 1

Encoding:

1	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

Operation: DEC
 $(DPTR) \leftarrow (DPTR) - 1$

DIV AB

Function: Divide

Description: DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags will be cleared.

Exception: if B had originally contained 00h, the values returned in the Accumulator and B register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

Example: The Accumulator contains 251 (0FBh or 11111011b) and B contains 18 (12h or 00010010b). The instruction,

DIV AB

will leave 13 in the Accumulator (0Dh or 00001101b) and the value 17 (11h or 00010001b) in B, since $251 = (13 \times 18) + 17$. Carry and OV will both be cleared.

Bytes: 1

Cycles: 3

Encoding:

1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

Operation: DIV
 $(A)_{15-8}, (B)_{7-0} \leftarrow (A) / (B)$

DJNZ <byte>, <rel-addr>

Function: Decrement and Jump if Not Zero**Description:** DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00h will underflow to 0FFh. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: Internal RAM locations 40h, 50h and 60h contain the values 01h, 70h, and 15h, respectively. The instruction sequence,

```
DJNZ 40H,LABEL_1
DJNZ 50H,LABEL_2
DJNZ 60H,LABEL_3
```

will cause a jump to the instruction at label LABEL_2 with the values 00h, 6Fh, and 15h in the three RAM locations. The first jump was not taken because the result was zero.

This instruction provides a simple way of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
TOGGLE:    MOV    R2, #8
           CPL    P1.7
           DJNZ   R2, TOGGLE
```

will toggle P1.7 eight times, causing four output pulses to appear at bit 7 of output Port 1. Each pulse will last three machine cycles; two for DJNZ and one to alter the pin.

DJNZ Rn, rel
Bytes: 2

Cycles: 3

Encoding:

1	1	0	1
---	---	---	---

1	r	r	r
---	---	---	---

relative

Operation: DJNZ
 $(PC) \leftarrow (PC) + 2$
 $(Rn) \leftarrow (Rn) - 1$
 IF $(Rn) > 0$ or $(Rn) < 0$
 THEN $(PC) \leftarrow (PC) + rel$

DJNZ direct, rel
Bytes: 3

Cycles: 4

Encoding:

0	1	0	1
---	---	---	---

0	0	1	1
---	---	---	---

direct address

relative

Operation: DJNZ
 $(PC) \leftarrow (PC) + 2$
 $(direct) \leftarrow (direct) - 1$
 IF $(direct) > 0$ or $(direct) < 0$
 THEN $(PC) \leftarrow (PC) + rel$

Preliminary

INC <byte>

Function: Increment

Description: INC increments the indicated variable by 1. An original value of 0FFh will overflow to 00h. No flags are affected. Three addressing modes are allowed: register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: Register 0 contains 7Eh (01111110b). Internal RAM locations 7Eh and 7Fh contain 0FFh and 40H, respectively. The instruction sequence,

```
INC @R0
INC R0
INC @R0
```

will leave register 0 set to 7Fh and internal RAM locations 7Eh and 7Fh holding 00h and 41h, respectively.

INC A

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---

Operation: INC
 $(A) \leftarrow (A) + 1$

INC Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	1	r	r	r
---	---	---	---	---	---	---	---

Operation: INC
 $(Rn) \leftarrow (Rn) + 1$

INC direct**Bytes:** 2**Cycles:** 2**Encoding:**

0	0	0	0	0	1	0	1	direct address
---	---	---	---	---	---	---	---	----------------

Operation: INC
 $(\text{direct}) \leftarrow (\text{direct}) + 1$ **INC @Ri****Bytes:** 1**Cycles:** 1**Encoding:**

0	0	0	0	0	1	1	i
---	---	---	---	---	---	---	---

Operation: INC
 $((Ri)) \leftarrow ((Ri)) + 1$

Preliminary

INC DPTR

Function: Increment Data Pointer

Description: Increment the 16-bit data pointer by 1. A 16-bit increment (modulo 2^{16}) is performed; an overflow of the low-order byte of the data pointer (DPL) from 0FFh to 00h will increment the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

Example: Registers DPH and DPL contain 12h and 0FEh, respectively. The instruction sequence,

```
INC DPTR
INC DPTR
INC DPTR
```

will change DPH and DPL to 13h and 01h.

Bytes: 1

Cycles: 1

Encoding:

1	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Operation: INC
 $(DPTR) \leftarrow (DPTR) + 1$

JB bit, rel

Function: Jump if Bit set

Description: If the indicated bit is a one, jump to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. The bit tested is not modified. No flags are affected.

Example: The data present at input port 1 is 11001010b. The Accumulator holds 56h (01010110b). The instruction sequence,

```
JB P1.2, LABEL1
JB ACC.2, LABEL2
```

will cause program execution to branch to the instruction at label LABEL2.

Bytes: 3

Cycles: 4

Encoding:

0 0 1 0	0 0 0 0
---------	---------

bit address

relative

Operation: JB
 $(PC) \leftarrow (PC) + 3$
 IF (bit) = 1
 THEN $(PC) \leftarrow (PC) + rel$

Preliminary

JBC bit, rel

Function: Jump if Bit is set and Clear bit

Description: If the indicated bit is one, branch to the address indicated; otherwise proceed with the next instruction. The bit will not be cleared if it is already a zero. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.

Note: When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, not the input pin.

Example: The Accumulator holds 56h (01010110b). The instruction sequence,

```
JBC ACC.3, LABEL1
JBC ACC.2, LABEL2
```

will cause program execution to continue at the instruction identified by the label LABEL2, with the Accumulator modified to 52h (01010010b).

Bytes: 3

Cycles: 4

Encoding:

0 0 0 1	0 0 0 0
---------	---------

bit address

relative

Operation: JBC
 $(PC) \leftarrow (PC) + 3$
 IF (bit) = 1
 THEN (bit) \leftarrow 0
 $(PC) \leftarrow (PC) + rel$

JC rel

Function: Jump if Carry is set

Description: If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example: The carry flag is cleared. The instruction sequence,

```
JC LABEL1
CPL C
JC LABEL2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 3

Encoding:

0 1 0 0	0 0 0 0
---------	---------

relative

Operation: JC
 $(PC) \leftarrow (PC) + 2$
 IF $(C) = 1$
 THEN $(PC) \leftarrow (PC) + rel$

Preliminary

JMP @A + DPTR

Function: Jump indirect

Description: Add the eight-bit unsigned contents of the Accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

Example: An even number from 0 to 6 is in the Accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```

MOV    DPTR, #JMP_TBL
JMP    @A+DPTR
JMP_TBL: AJMP LABEL0
        AJMP LABEL1
        AJMP LABEL2
        AJMP LABEL3

```

If the Accumulator equals 04h when starting this sequence, execution will jump to label LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

Bytes: 1

Cycles: 2

Encoding:

0	1	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Operation: JMP
 $(PC) \leftarrow (A) + (DPTR)$

JNB bit, rel

Function: Jump if Bit Not set

Description: If the indicated bit is a zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. The bit tested is not modified. No flags are affected.

Example: The data present at input port 1 is 11001010b. The Accumulator holds 56h (01010110b). The instruction sequence,

```
JNB P1.3, LABEL1
JNB ACC.3, LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

Bytes: 3

Cycles: 4

Encoding:

0 0 1 1	0 0 0 0	bit address	relative
---------	---------	-------------	----------

Operation: JNB
 $(PC) \leftarrow (PC) + 3$
 IF (bit) = 0
 THEN $(PC) \leftarrow (PC) + rel$

JNC rel

Function: Jump if Carry not set

Description: If the carry flag is a zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example: The carry flag is set. The instruction sequence,

```
JNC LABEL1
CPL C
JNC LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 3

Encoding:

0 1 0 1	0 0 0 0	relative
---------	---------	----------

Operation:

```
JNC
(PC) ← (PC) + 2
IF (C) = 0
  THEN (PC) ← (PC) + rel
```

JNZ rel

Function: Jump if Accumulator Not Zero

Description: If any bit of the Accumulator is a one, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 00h. The instruction sequence,

```
JNZ LABEL1
INC A
JNZ LAEEL2
```

will set the Accumulator to 01h and continue at label LABEL2.

Bytes: 2

Cycles: 3

Encoding:

0 1 1 1	0 0 0 0
---------	---------

relative

Operation: JNZ
 $(PC) \leftarrow (PC) + 2$
 IF $(A) \neq 0$
 THEN $(PC) \leftarrow (PC) + rel$

Preliminary

JZ rel

Function: Jump if Accumulator Zero

Description: If all bits of the Accumulator are zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally contains 01h. The instruction sequence,

```
JZ LABEL1
DEC A
JZ LABEL2
```

will change the Accumulator to 00h and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 3

Encoding:

0	1	1	0	0	0	0	0	relative
---	---	---	---	---	---	---	---	----------

Operation:

```
JNZ
(PC) ← (PC) + 2
IF (A) = 0
  THEN (PC) ← (PC) + rel
```

LCALL addr16

Function: Long call

Description: LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64Kbyte program memory address space. No flags are affected.

Example: Initially the Stack Pointer equals 07h. The label "SUBRTN" is assigned to program memory location 1234h. After executing the instruction,

LCALL SUBRTN

at location 0123h, the Stack Pointer will contain 09h, internal RAM locations 08h and 09h will contain 26h and 01h, and the PC will contain 1234h.

Bytes: 3

Cycles: 4

Encoding:

0	0	0	1
---	---	---	---

0	0	1	0
---	---	---	---

addr15 ~ addr8

addr7 ~ addr0

Operation: LCALL
 $(PC) \leftarrow (PC) + 3$
 $(SP) \leftarrow (SP) + 1$
 $((SP)) \leftarrow (PC_{7-0})$
 $(SP) \leftarrow (SP) + 1$
 $((SP)) \leftarrow (PC_{15-8})$
 $(PC) \leftarrow (PC) + 3$
 $(PC) \leftarrow \text{addr}_{15-0}$

LJMP **addr16**

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64Kbyte program memory address space. No flags are affected.

Example: The label "JMPADR" is assigned to the instruction at program memory location 1234h. The instruction,

LJMP JMPADR

at location 0123h will load the program counter with 1234h.

Bytes: 3

Cycles: 4

Encoding:

0 0 0 0	0 0 1 0	addr15 ~ addr8	addr7 ~ addr0
---------	---------	----------------	---------------

Operation: LJMP
(PC) ← addr₁₅₋₀

MOV <dest-byte>, <src-byte>

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

Example: Internal RAM location 30h holds 40h. The value of RAM location 40h is 10h. The data present at input port 1 is 11001010b (0CAh).

```
MOV R0, #30h ;R0 <= 30h
MOV A, @R0 ;A <= 40h
MOV R1, A ;R1 <= 40h
MOV B, @R1 ;B <= 10h
MOV @R1, P1 ;RAM (40h)<= 0CAh
MOV P2, P1 ;P2 #0CAh
```

leaves the value 30h in register 0, 40h in both the Accumulator and register 1, 10h in register B, and 0CAh (11001010b) both in RAM location 40h and output on port 2.

MOV A, Rn

Bytes: 1

Cycles: 1

Encoding:

1	1	1	0	1	r	r	r
---	---	---	---	---	---	---	---

Operation:

MOV
(A) ← (Rn)

MOV A, direct

Bytes: 2

Cycles: 2

Encoding:

1	1	1	0	0	1	0	1	direct address
---	---	---	---	---	---	---	---	----------------

Operation:

MOV
(A) ← (direct)

MOV A, ACC is not a valid instruction

MOV A, @Ri**Bytes:** 1**Cycles:** 1**Encoding:**

1 1 1 0	0 1 1 i
---------	---------

Operation: MOV
(A) ← ((Ri))**MOV A, #data****Bytes:** 2**Cycles:** 2**Encoding:**

0 1 1 1	0 1 0 0
---------	---------

immediate data

Operation: MOV
(B) ← #data**MOV Rn, A****Bytes:** 1**Cycles:** 1**Encoding:**

1 1 1 1	1 r r r
---------	---------

Operation: MOV
(Rn) ← (A)**MOV Rn, direct****Bytes:** 2**Cycles:** 2**Encoding:**

1 0 1 0	1 r r r
---------	---------

direct address

Operation: MOV
(Rn) ← (direct)**MOV Rn, #data****Bytes:** 2**Cycles:** 2

Encoding:

0	1	1	1
---	---	---	---

1	r	r	r
---	---	---	---

immediate data

Operation: MOV
(Rn) ← #data

MOV direct, A

Bytes: 2

Cycles: 2

Encoding:

1	1	1	1
---	---	---	---

0	1	0	1
---	---	---	---

direct address

Operation: MOV
(direct) ← (A)

MOV direct, Rn

Bytes: 2

Cycles: 2

Encoding:

1	0	0	0
---	---	---	---

1	r	r	r
---	---	---	---

direct address

Operation: MOV
(direct) ← (Rn)

MOV direct, direct

Bytes: 3

Cycles: 3

Encoding:

1	0	0	0
---	---	---	---

0	1	0	1
---	---	---	---

dir. addr. (src)

dir. addr. (dest)

Operation: MOV
(direct) ← (direct)

MOV direct, @Ri

Bytes: 2

Cycles: 2

Encoding:

1	0	0	0
---	---	---	---

0	1	1	i
---	---	---	---

direct address

Operation: MOV
(direct) ← ((Ri))

Preliminary

MOV direct, #data**Bytes:** 3**Cycles:** 3

Encoding:	0 1 1 1	0 1 0 1	direct address	immediate data
------------------	---------	---------	----------------	----------------

Operation: MOV
 (direct) ← #data

MOV @Ri, A**Bytes:** 1**Cycles:** 1

Encoding:	1 1 1 1	0 1 1 i
------------------	---------	---------

Operation: MOV
 ((Ri)) ← (A)

MOV @Ri, direct**Bytes:** 2**Cycles:** 2

Encoding:	1 0 1 0	0 1 1 i	direct address
------------------	---------	---------	----------------

Operation: MOV
 ((Ri)) ← (direct)

MOV @Ri, #data**Bytes:** 2**Cycles:** 2

Encoding:	0 1 1 1	0 1 1 i	immediate data
------------------	---------	---------	----------------

Operation: MOV
 ((Ri)) ← #data

MOV <dest-bit>, <src-bit>

Function: Move bit data

Description: The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: The carry flag is originally set. The data present at input Port 3 is 11000101b. The data previously written to output Port 1 is 35h (00110101b).

```
MOV P1.3, C
MOV C, P3.3
MOV P1.2, C
```

will leave the carry cleared and change Port 1 to 39h (00111001b).

MOV C, bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 0 1 0	bit address
---------	---------	-------------

Operation: MOV
(C) ← (bit)

MOV bit, C

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 0 1 0	bit address
---------	---------	-------------

Operation: MOV
(bit) ← (C)

MOV DPTR, #data16

Function: Load Data Pointer with a 16-bit constant

Description: The Data Pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.

This is the only instruction, which moves 16 bits of data at once.

Example: The instruction,

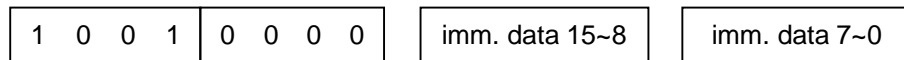
```
MOV DPTR, # 1234h
```

will load the value 1234h into the Data Pointer: DPH will hold 12h and DPL will hold 34h.

Bytes: 3

Cycles: 3

Encoding:



Operation:

```
MOV
(DPTR) ← #data15-0
{DPH, DPL} ← {#data15-8, #data7-0}
```

MOVC A, @A + <base-reg>

Function: Move Code byte

Description: The MOVC instructions load the Accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit Accumulator contents and the contents of a sixteen-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

Example: A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive.

```
REL-PC:      INC    A
              MOVC  A, @A+PC
              RET
              DB    66h
              DB    77h
              DB    88h
              DB    99h
```

If the subroutine is called with the Accumulator equal to 01h, it will return with 77h in the Accumulator. The INC A before the MOVC instruction is needed to “get around” the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the Accumulator instead.

MOVC A, @A + DPTR

Bytes: 1

Cycles: 2

Encoding:

1	0	0	1	0	0	1	1
---	---	---	---	---	---	---	---

Operation: MOV
 $(A) \leftarrow ((A) + (DPTR))$

MOVC A, @A + PC**Bytes:** 1**Cycles:** 2**Encoding:**

1	0	0	1	0	0	1	1
---	---	---	---	---	---	---	---

Operation: MOVC
 $PC \leftarrow PC + 1$
 $(A) \leftarrow ((A) + (PC))$

Note: It is recommended that all interrupts are disabled before using MOVC instruction.

MOVX <dest-byte>, <src-byte>

Function: Move External

Description: The MOVX instructions transfer data between the Accumulator and a byte of external data memory, hence the “X” appended to MOV. There are two types of instructions, differing in whether they provide an eight-bit or sixteen-bit indirect address to the external data RAM.

In the first type the contents of R0 or R1 in the current register bank provide an eight-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, any output port pins can be used to output higher-order address bits. These pins would be controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, the Data Pointer generates a sixteen-bit address. P2 outputs the high-order eight address bits (the contents of DPH) while P0 multiplexes the low-order eight bits (DPL) with data. The P2 Special Function Register retains its previous contents while the P2 output buffers are emitting the contents of DPH. This form is faster and more efficient when accessing very large data arrays (up to 64Kbytes), since no additional instructions are needed to set up the output ports.

It is possible in some situations to mix the two MOVX types. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2 followed by a MOVX instruction using R0 or R1.

Example: An external 256byte RAM using multiplexed address/data lines (e.g., an Intel 8155 RAM / I/O / TIMER) is connected to the 8052 Port 0. Port 3 provides control lines for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12h and 34h. Location 34h of the external RAM holds the value 56h. The instruction sequence,

```
MOVX A, @R1
MOVX @R0, A
```

copies the value 56h into both the Accumulator and external RAM location 12h.

MOVX A, @Ri

Bytes: 1

Cycles: 3

Encoding:

1	1	1	0	0	0	1	i
---	---	---	---	---	---	---	---

Operation: MOVX
(A) ← ((Ri))

MOVX A, @DPTR**Bytes:** 1**Cycles:** 3**Encoding:**

1	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---

Operation: MOVX
(A) ← ((DPTR))**MOVX @Ri, A****Bytes:** 1**Cycles:** 3**Encoding:**

1	1	1	1	0	0	1	i
---	---	---	---	---	---	---	---

Operation: MOVX
((Ri)) ← (A)**MOVC @DPTR, A****Bytes:** 1**Cycles:** 3**Encoding:**

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

Operation: MOVX
(DPTR) ← (A)

MUL AB

Function: Multiply**Description:** MUL AB multiplies the unsigned eight-bit integers in the Accumulator and register B. The low-order byte of the sixteen-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (0FFh) the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.**Example:** Originally the Accumulator holds the value 80 (50h). Register B holds the value 160 (0A0h). The instruction,

MUL AB

will give the product 12,800 (3200h), so B is changed to 32h (00110010b) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes: 1**Cycles:** 3**Encoding:**

1	0	1	0	0	1	0	0
---	---	---	---	---	---	---	---

Operation: MUL
 $(A)_{7-0}, (B)_{15-8} \leftarrow (A) \times (B)$

Preliminary

NOP

Function: No Operation

Description: Execution continues at the following instruction. Other than the PC, no registers or flags are affected.

Example: It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming no interrupts are enabled) with the instruction sequence,

```
CLR    P2.7
NOP
NOP
NOP
NOP
SETB   P2.7
```

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

Operation: NOP
 $(PC) \leftarrow (PC) + 1$

ORL <dest-byte> <src-byte>

Function: Logical-OR for byte variables

Description: ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: If the Accumulator holds 0C3h (11000011b) and R0 holds 55h (01010101b) then the instruction,

```
ORL A, R0
```

will leave the Accumulator holding the value 0D7h (11010111b).

When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction,

```
ORL P1, # 00110010b
```

will set bits 5, 4, and 1 of output Port 1.

ORL A, Rn

Bytes: 1

Cycles: 1

Encoding:

0	1	0	0	1	r	r	r
---	---	---	---	---	---	---	---

Operation: ORL
 $(A) \leftarrow (A) \vee (Rn)$

ORL A, direct

Bytes: 2

Cycles: 2

Encoding:

0	1	0	0	0	1	1	i
---	---	---	---	---	---	---	---

direct address

Operation: ORL
 $(A) \leftarrow (A) \vee (\text{direct})$

ORL A, @Ri

Bytes: 1

Cycles: 1

Encoding:

0 1 0 0	0 1 1 i
---------	---------

Operation: ORL
 $(A) \leftarrow (A) \vee ((Ri))$

ORL A, #data

Bytes: 2

Cycles: 2

Encoding:

0 1 0 0	0 1 0 0
---------	---------

immediate data

Operation: ORL
 $(A) \leftarrow (A) \vee \#data$

ORL direct, A

Bytes: 2

Cycles: 2

Encoding:

0 1 0 0	0 0 1 0
---------	---------

direct address

Operation: ORL
 $(\text{direct}) \leftarrow (\text{direct}) \vee (A)$

ORL direct, #data

Bytes: 3

Cycles: 3

Encoding:

0 1 0 0	0 0 1 1
---------	---------

direct address

immediate data

Operation: ORL
 $(\text{direct}) \leftarrow (\text{direct}) \vee \#data$

ORL C, <src-bit>

Function: Logical-OR for bit variables

Description: Set the carry flag if the Boolean value is a logical 1; leave the carry in its current state otherwise. A slash (“/”) preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Example: Set the carry flag if and only if P1.0 = 1, ACC. 7 = 1, or OV = 0:

```
MOV    C, P1.0      ;LOAD CARRY WITH INPUT PIN P1.0
ORL    C, ACC.7     ;OR CARRY WITH THE ACC.BIT 7
ORL    C, /OV       ;OR CARRY WITH THE INVERSE OF OV.
```

ORL C, bit

Bytes: 2

Cycles: 2

Encoding:

0	1	1	1	0	0	1	0
---	---	---	---	---	---	---	---

bit address

Operation: ORL
 $(C) \leftarrow (C) \vee (\text{bit})$

ORL C, /bit

Bytes: 2

Cycles: 2

Encoding:

1	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---

bit address

Operation: ORL
 $(C) \leftarrow (C) \vee \sim(\text{bit})$

Preliminary

POP direct

Function: Pop from stack.

Description: The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by 1. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

Example: The Stack Pointer originally contains the value 32h, and internal RAM locations 30h through 32h contain the values 20h, 23h, and 01h, respectively. The instruction sequence,

```
POP DPH
POP DPL
```

will leave the Stack Pointer equal to the value 30H and the Data Pointer set to 0123h. At this point the instruction,

```
POP SP
```

will leave the Stack Pointer set to 20h. Note that in this special case the Stack Pointer was decremented to 2Fh before being loaded with the value popped (20h).

Bytes: 2

Cycles: 2

Encoding:

1	1	0	1	0	0	0	0	direct address
---	---	---	---	---	---	---	---	----------------

Operation:

```
POP
(direct) ← ((SP))
(SP) ← (SP) - 1
```

PUSH direct

Function: Push onto stack

Description: The Stack Pointer is incremented by 1. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine the Stack Pointer contains 09h. The Data Pointer holds the value 0123h. The instruction sequence,

```
PUSH DPL
PUSH DPH
```

will leave the Stack Pointer set to 0Bh and store 23h and 01h in internal RAM locations 0Ah and 0Bh, respectively.

Bytes: 2

Cycles: 2

Encoding:

1	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---

direct address

Operation:
 PUSH
 $(SP) \leftarrow (SP) + 1$
 $((SP)) \leftarrow (\text{direct})$

Preliminary

RET

Function: Return from subroutine

Description: RET pops the high-and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by 2. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.

Example: The Stack Pointer originally contains the value 0Bh. Internal RAM locations 0Ah and 0Bh contain the values 23h and 01h, respectively. The instruction,

RET

will leave the Stack Pointer equal to the value 09h. Program execution will continue at location 0123h.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	0	0	0	1	0
---	---	---	---	---	---	---	---

Operation: RET
 $(PC_{15-8}) \leftarrow ((SP))$
 $(SP) \leftarrow (SP) - 1$
 $(PC_{7-0}) \leftarrow ((SP))$
 $(SP) \leftarrow (SP) - 1$

RETI

Function: Return from interrupt

Description: RETI pops the high-and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected; the PSW is not automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower-or same-level interrupt had been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.

Example: The Stack Pointer originally contains the value 0Bh. An interrupt was detected during the instruction ending at location 0122h. Internal RAM locations 0Ah and 0Bh contain the values 23h and 01h, respectively. The instruction,

RETI

will leave the Stack Pointer equal to 09h and return program execution to location 0123h.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	1	0	0	1	0
---	---	---	---	---	---	---	---

Operation: RETI
 $(PC_{15-8}) \leftarrow ((SP))$
 $(SP) \leftarrow (SP) - 1$
 $(PC_{7-0}) \leftarrow ((SP))$
 $(SP) \leftarrow (SP) - 1$

Preliminary

RL A

Function: Rotate Accumulator Left**Description:** The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.**Example:** The Accumulator holds the value 0C5h (11000101b). The instruction,

RL A

leaves the Accumulator holding the value 8Bh (10001011b) with the carry unaffected.

Bytes: 1**Cycles:** 1**Encoding:**

0	0	1	0	0	0	1	1
---	---	---	---	---	---	---	---

Operation: RL
 $(A_n + 1) \leftarrow (A_n) \quad n = 0 \sim 6$
 $(A0) \leftarrow (A7)$

RLC A

Function: Rotate Accumulator Left through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

Example: The Accumulator holds the value 0C5h (11000101b), and the carry is zero. The instruction,

RLC A

leaves the Accumulator holding the value 8Bh (10001010b) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Operation: RLC
 $(A_n + 1) \leftarrow (A_n) \quad n = 0 \sim 6$
 $(A0) \leftarrow (C)$
 $(C) \leftarrow (A7)$

Preliminary

RR A

Function: Rotate Accumulator Right

Description: The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example: The Accumulator holds the value 0C5h (11000101b). The instruction,

RR A

leaves the Accumulator holding the value 0E2h (11100010b) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

Operation: RR
 $(A_n) \leftarrow (A_n + 1) \quad n = 0 \sim 6$
 $(A7) \leftarrow (A0)$

RRC A

Function: Rotate Accumulator Right through Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original value of the carry flag moves into the bit 7 position. No other flags are affected.

Example: The Accumulator holds the value 0C5h (11000101b), the carry is zero. The instruction,

RRC A

leaves the Accumulator holding the value 62h (01100010b) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	0	1	1
---	---	---	---	---	---	---	---

Operation: RRC
 $(A_n) \leftarrow (A_n + 1) \quad n = 0 \sim 6$
 $(A7) \leftarrow (C)$
 $(C) \leftarrow (A0)$

Preliminary

SETB <bit>

Function: Set Bit

Description: SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

Example: The carry flag is cleared. Output Port 1 has been written with the value 34h (00110100b). The instructions,

```
SETB C
SETB PI.0
```

will leave the carry flag set to 1 and change the data output on Port 1 to 35h (00110101b).

SETB C

Bytes: 1

Cycles: 1

Encoding:

1 1 0 1	0 0 1 1
---------	---------

Operation: SETB
(C) ← 1

SETB bit

Bytes: 2

Cycles: 2

Encoding:

1 1 0 1	0 0 1 0
---------	---------

bit address

Operation: SETB
(bit) ← 1

SJMP rel

Function: Short Jump

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Example: The label “RELADR” is assigned to an instruction at program memory location 0123h. The instruction,

SJMP RELADR

will assemble into location 0100h. After the instruction is executed, the PC will contain the value 0123h.

(Note: Under the above conditions the instruction following SJMP will be at 102h. Therefore, the displacement byte of the instruction will be the relative offset $(0123h - 0102h) = 21h$. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

Bytes: 2

Cycles: 3

Encoding:



Operation:

SJMP
 $(PC) \leftarrow (PC) + 2$
 $(PC) \leftarrow (PC) + rel$

SUBB A, <src-byte>

Function: Subtract with borrow

Description: SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set before executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

Example: When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

The Accumulator holds 0C9h (11001001b), register 2 holds 54h (01010100b), and the carry flag is set. The instruction,

SUBB A, R2

will leave the value 74h (01110100b) in the accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9h minus 54h is 75h. The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR C instruction.

SUBB A, Rn

Bytes: 1

Cycles: 1

Encoding:

1	0	0	1	1	r	r	r
---	---	---	---	---	---	---	---

Operation: SUBB
 $(A) \leftarrow (A) - (C) - (Rn)$

SUBB A, direct

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 1 0 1
---------	---------

direct address

Operation: SUBB
 $(A) \leftarrow (A) - (C) - (\text{direct})$

SUBB A, @Ri

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1	0 1 1 i
---------	---------

Operation: SUBB
 $(A) \leftarrow (A) - (C) - ((Ri))$

SUBB A, #data

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 1 0 0
---------	---------

immediate data

Operation: SUBB
 $(A) \leftarrow (A) - (C) - \#data$

SWAP A

Function: Swap nibbles within the Accumulator

Description: SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a four-bit rotate instruction. No flags are affected.

Example: The Accumulator holds the value 0C5h (11000101b). The instruction,
SWAP A
leaves the Accumulator holding the value 5Ch (01011100b).

Bytes:

Cycles: 1

Encoding: 1

Operation:

1	1	0	0	0	1	0	0
---	---	---	---	---	---	---	---

XCH
(A₃₋₀) ↔ (A₇₋₄)

XCH A, <byte>

Function: Exchange Accumulator with byte variable

Description: XCH leads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

Example: R0 contains the address 20h. The Accumulator holds the value 3Fh (00111111b). Internal RAM location 20h holds the value 75h (01110101b). The instruction,

XCH A, @R0

will leave RAM location 20h holding the values 3Fh (00111111b) and 75h (01110101b) in the accumulator.

XCH A, Rn

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	1	r	r	r
---	---	---	---	---	---	---	---

Operation: XCH
(A) ↔ (Rn)

Preliminary

XCH A, direct

Bytes: 2

Cycles: 2

Encoding:

1	1	0	0	0	1	0	1
---	---	---	---	---	---	---	---

direct address

Operation: XCH
(A) ↔ (direct)

XCH A, @Ri

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	0	1	1	i
---	---	---	---	---	---	---	---

Operation: XCH
(A) ↔ ((Ri))

XCHD A, @Ri

Function: Exchange Digit

Description: XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected

Example: R0 contains the address 20h. The Accumulator holds the value 36h (00110110b). Internal RAM location 20h holds the value 75h (01110101b). The instruction,

XCHD A, @R0

will leave RAM location 20h holding the value 76h (01110110b) and 35h (00110101b) in the Accumulator.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	1	i
---	---	---	---	---	---	---	---

Operation: XCH
 $(A_{3-0}) \leftrightarrow ((Rn_{3-0}))$

XRL <dest-byte>, <src-byte>

Function: Logical Exclusive-OR for byte variables

Description: XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

(Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.)

Example: If the Accumulator holds 0C3h (11000011b) and register 0 holds 0Aah (10101010b) then the instruction,

XRL A, R0

will leave the Accumulator holding the value 69h (01101001b).

When the destination is a directly addressed byte this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The instruction,

XRL PI, #00110001b

will complement bits 5, 4, and 0 of output Port 1.

XRL A, Rn

Bytes: 1

Cycles: 1

Encoding:

0	1	1	0	1	r	r	r
---	---	---	---	---	---	---	---

Operation: XRL
 $(A) \leftarrow (A) \otimes (Rn)$

XRL A, direct

Bytes: 2

Cycles: 2

Encoding:

0	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---

direct address

Operation: XRL
 $(A) \leftarrow (A) \otimes (\text{direct})$

XRL A, @Ri

Bytes: 1

Cycles: 1

Encoding:

0	1	1	0	0	1	1	i
---	---	---	---	---	---	---	---

Operation: XRL
 $(A) \leftarrow (A) \otimes ((Ri))$

XRL A, #data

Bytes: 2

Cycles: 2

Encoding:

0	1	1	0	0	1	0	0
---	---	---	---	---	---	---	---

immediate data

Operation: XRL
 $(A) \leftarrow (A) \otimes \#data$

XRL direct, A

Bytes: 2

Cycles: 2

Encoding:

0	1	1	0	0	0	1	0
---	---	---	---	---	---	---	---

direct address

Operation: XRL
 $(\text{direct}) \leftarrow (\text{direct}) \otimes (A)$

XRL direct, #data

Bytes: 3

Cycles: 3

Encoding:

0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---

direct address

immediate data

Operation: XRL
 $(\text{direct}) \leftarrow (\text{direct}) \otimes \#data$

14 Appendix B: SFR description

14.1 Port 0 Register (P0)

Bit No.	7	6	5	4	3	2	1	0
80h	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

This port functions as a multiplexed address/data bus during external memory access, and as a general purpose I/O port on devices which incorporate internal program memory. During external memory cycles, this port will contain the LSB of the address when ALE is high, and data when ALE is low. When used as a general purpose I/O, this port is open-drain and can select pull-up resistors optionally. Pull-up resistors are not required when used as a memory interface.

14.2 Stack Pointer Register (SP)

Bit No.	7	6	5	4	3	2	1	0
81h	SP.7	SP.6	SP.5	SP.4	SP.3	SP.2	SP.1	SP.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(1)	R/W(1)	R/W(1)

The stack pointer identifies the location where the stack will begin. The stack pointer is incremented before every PUSH operation. This register defaults to 07H after reset.

14.3 Data Pointer Low Register (DPL)

Bit No.	7	6	5	4	3	2	1	0
82h	DPL.7	DPL.6	DPL.5	DPL.4	DPL.3	DPL.2	DPL.1	DPL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register is the low byte of the 16-bit data pointer. DPL and DPH are used to point to non-scratchpad data RAM.

14.4 Data Pointer High Register (DPH)

Bit No.	7	6	5	4	3	2	1	0
83h	DPH.7	DPH.6	DPH.5	DPH.4	DPH.3	DPH.2	DPH.1	DPH.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register is the high byte of the 16-bit data pointer. DPL and DPH are used to point to non-scratchpad data RAM.

14.5 Power Control Register (PCON)

Bit No.	7	6	5	4	3	2	1	0
87h	SMOD1	-	-	POF	GF1	GF0	PD	IDL
	R/W(0)			R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
SMOD1	This bit doubles the serial port baud rate in mode 1, 2, and 3 when set to 1.
-	Not implemented, reserved for future use. Note: User software should not write 1s to reserved bits. These bits may be used in future products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.
POF	Power Off flag. When power-on, this bit will be set by H/W.
GF1	General purpose flag bit
GF0	General purpose flag bit
PD	Power down (Stop) mode. Setting this bit causes the MiDAS2.1 family to go into the POWER DOWN mode. In this mode all the clocks are stopped and program execution is frozen.
IDL	Idle mode. Setting this bit causes the MiDAS2.1 family to go into the IDLE mode. In this mode the clocks to the CPU are stopped, so program execution is frozen. But the clock to the peripherals and interrupt blocks is not stopped, and these blocks continue operating.

14.6 Timer 0/1 Control Register (TCON)

Bit No.	7	6	5	4	3	2	1	0
88h	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
TF1	<p>Timer 1 Overflow Flag. This bit indicates when Timer 1 overflows its maximum count as defined by the current mode. This bit can be cleared by software and is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine.</p> <p>0: No Timer 1 overflow has been detected. 1: Timer 1 has overflowed its maximum count</p>
TR1	<p>Timer 1 Run Control. This bit enables/disables the operation of Timer 1. Halting this timer will preserve the current count in TH1 and TL1.</p> <p>0: Timer 1 is halted. 1: Timer 1 is enabled.</p>
TF0	<p>Timer 0 Overflow Flag. This bit indicates when Timer 0 overflows its maximum count as defined by the current mode. This bit can be cleared by software and is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine.</p> <p>0: No Timer 0 overflow has been detected. 1: Timer 0 has overflowed its maximum count</p>
TR0	<p>Timer 0 Run Control. This bit enables/disables the operation of Timer 0. Halting this timer will preserve the current count in TH0 and TL0.</p> <p>0: Timer 0 is halted. 1: Timer 0 is enabled.</p>
IE1	<p>Interrupt 1 Edge Detect. This bit is set when an edge/level of the type defined by IT1 is detected. If IT1=1, this bit will remain set until cleared in software or the start of the External Interrupt 1 service routine. If IT1=0, this bit will inversely reflect the state of the /INT1 pin.</p>
IT1	<p>Interrupt 1 Type Select. This bit selects whether the /INT1 pin will detect edge or level triggered interrupts.</p> <p>0: /INT1 is level triggered. 1: /INT1 is edge triggered.</p>
IE0	<p>Interrupt 0 Edge Detect. This bit is set when an edge/level of the type defined by IT0 is detected. If IT0=1, this bit will remain set until cleared in software or the start of the</p>

	External Interrupt 0 service routine. If ITO=0, this bit will inversely reflect the state of the /INT0 pin.
IT0	Interrupt 0 Type Select. This bit selects whether the /INT0 pin will detect edge or level triggered interrupts. 0: /INT0 is level triggered. 1: /INT0 is edge triggered.

14.7 Timer Mode Control Register (TMOD)

Bit No.	7	6	5	4	3	2	1	0
89h	-	-	-	-	GATE	C/T	M1	M0
					R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
GATE	Timer 0 Gate Control. This bit enables/disables the ability of Timer 0 to increment. 0: Timer 0 will clock when TR0=1, regardless of the states of /INT0. 1: Timer 0 will clock only when TR0=1 and /INT0=1.
C/T	Timer 0 Counter/Timer Select. 0: Timer 0 is incremented by internal clocks. 1: Timer 0 is incremented by pulses on T0 when TR0 (TCON.4) is 1.
M1, M0	Timer 0 Mode Select. These bits select the operating mode of Timer 0. When Timer 0 is in mode 3, TL0 is started/stopped by TR0 and TH0 is started/stopped by TR1. Run control for Timer 1 is then provided via the Timer 1 mode selection.
	M1 M0 Mode
	0 0 Mode 0: 8 bits with 5-bit prescale
	0 1 Mode 1: 16 bits
	1 0 Mode 2: 8 bits with auto-reload
1 1 Mode 3: Timer 1 is halted, but holds its count.	

14.8 Timer 0 Low Byte Register (TL0)

Bit No.	7	6	5	4	3	2	1	0
8Ah	TL0.7	TL0.6	TL0.5	TL0.4	TL0.3	TL0.2	TL0.1	TL0.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register contains the least significant byte of Timer 0.

14.9 Timer 1 Low Byte Register (TL1)

Bit No.	7	6	5	4	3	2	1	0
8Bh	TL1.7	TL1.6	TL1.5	TL1.4	TL1.3	TL1.2	TL1.1	TL1.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register contains the least significant byte of Timer 1.

14.10 Timer 0 High Byte Register (TH0)

Bit No.	7	6	5	4	3	2	1	0
8Ch	TH0.7	TH0.6	TH0.5	TH0.4	TH0.3	TH0.2	TH0.1	TH0.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register contains the most significant byte of Timer 0.

14.11 Timer 1 High Byte Register (TH1)

Bit No.	7	6	5	4	3	2	1	0
8Dh	TH1.7	TH1.6	TH1.5	TH1.4	TH1.3	TH1.2	TH1.1	TH1.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register contains the most significant byte of Timer 1.

14.12 Port 1 Register (P1)

Bit No.	7	6	5	4	3	2	1	0
90h	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

This register functions as a general purpose I/O port.

14.13 External Interrupt Flag Register (EXIF)

Bit No.	7	6	5	4	3	2	1	0
91h	-	-	IE3	IE2	XT/RG	RGMD	RGSL	BGS
			R/W(0)	R/W(0)	R/W(0)	R(1)	R/W(0)	R/W(1)

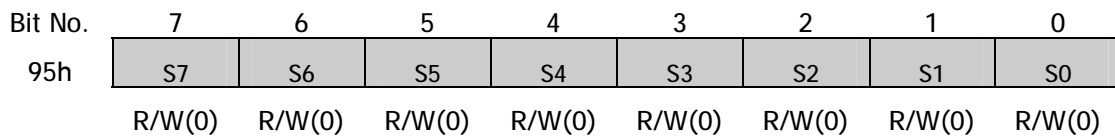
Symbol	Function
IE3	External Interrupt 3 Flag. This bit will be set when a falling edge is detected on /INT3. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.
IE2	External Interrupt 2 Flag. This bit will be set when a rising edge is detected on INT2. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.
XT/RG	Crystal Source Select. 0: The internal Ring oscillator is selected as the clock source. 1: The crystal oscillator is selected.
RGMD	Ring mode. Generally, RGMD is the inversion of XT/RG
RGSL	Oscillator select bit when exiting from the power-down. 1: When exiting from the power-down mode, the system use the internal ring oscillator as system clock during 65,536 XTAL oscillator clock cycles.
BGS	Band-Gap Select. This bit enables/disables the band-gap reference during Stop mode. Disabling the band-gap reference provides significant power savings in Stop mode, but sacrifices the ability to perform a power fail interrupt or power-fail reset while stopped. The state of this bit will be undefined on devices which do not incorporate a band-gap reference. 0: The band-gap reference is disabled in Stop mode but will function during normal operation. 1: The band-gap reference will operate in Stop mode.

14.14 Peripheral Clock Control Register (CLKOFF)

Bit No.	7	6	5	4	3	2	1	0
94h	-	-	OFF_T01	OFF_UART	-	OFF_I2C	OFF_PWM	OFF_ADC
			R/W(0)	R/W(0)		R/W(0)	R/W(0)	R/W(0)

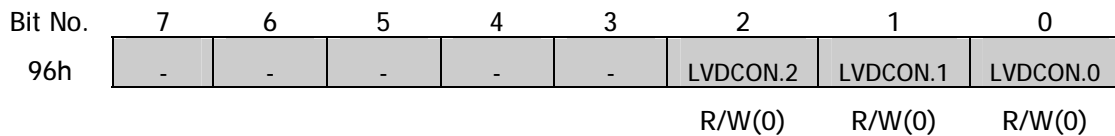
Symbol	Function
OFF_ADC	ADC block OFF, 1 = ADC block will stop.
OFF_PWM	PWM block OFF, 1 = PWM block will stop.
OFF_I2C	I2C block OFF, 1 = I2C Master & Slave block will stop.
OFF_UART	UART block OFF, 1 = UART block will stop.
OFF_T01	Timer0/1 block OFF, 1 = Timer0 and Timer1 will stop.

14.15 Ring Control Configuration Register (RINGCON)



Symbol	Function
RINGCON[7:0]	Internal Ring oscillator can be tuned.

14.16 LVD Control Register (LVDCON)



Symbol	Function
LVDCON[2:0]	Select the LVD Interrupt Level. (MUX) LVDCON[2:0] = 000b, LVD Interrupt Level = 4.0V (Default). LVDCON[2:0] = 001b, LVD Interrupt Level = 3.6V. LVDCON[2:0] = 010b, LVD Interrupt Level = 3.2V. LVDCON[2:0] = 011b, LVD Interrupt Level = 3.0V. LVDCON[2:0] = 100b, LVD Interrupt Level = 2.8V. LVDCON[2:0] = 101b, LVD Interrupt Level = 2.6V. LVDCON[2:0] = 110b, LVD Interrupt Level = 2.4V. LVDCON[2:0] = 111b, LVD Interrupt Level = 2.2V.

14.17 LVD Status Register (LVDST)

Bit No.	7	6	5	4	3	2	1	0
97h	LVD7	LVD6	LVD5	LVD4	LVD3	LVD2	LVD1	LVD1
	R(0)	R(0)	R(0)	R(0)	R(0)	R(0)	R(0)	R(0)

Symbol	Function
LVD7	1 when VDD \leq 4.0V.
LVD6	1 when VDD \leq 3.6V.
LVD5	1 when VDD \leq 3.2V.
LVD4	1 when VDD \leq 3.0V.
LVD3	1 when VDD \leq 2.8V.
LVD2	1 when VDD \leq 2.6V.
LVD1	1 when VDD \leq 2.4V.
LVD0	1 when VDD \leq 2.2V.

14.18 Serial Port Control Register (SCON)

Bit No.	7	6	5	4	3	2	1	0
98h	-	-	-	REN	-	-	TI	RI
				R/W(0)			R/W(0)	R/W(0)

Symbol	Function
REN	Receive Enable. This bit enables/disables the serial port receiver shift register. 0: serial port reception disabled. 1: serial port receiver enabled (modes 1, 2 and 3). Initiate synchronous reception (mode 0).
TI	Transmitter Interrupt Flag. This bit indicates that data in the serial port buffer has been completely shifted out. In serial port mode 0, TI is set at the end of the 8 th data bit. In all other modes, this bit is set at the end of the last data bit. This bit must be manually cleared by software.
RI	Receiver Interrupt Flag. This bit indicates that a byte of data has been received in the serial port buffer. In serial port mode 0, RI is set at the end of the 8 th bit. In serial port mode 1, RI is set after the last sample of the incoming stop bit subject to the state of SM2. In modes 2 and 3, RI is set after the last sample of RB8. This bit must be manually

	cleared by software.
--	----------------------

14.19 Serial Data Buffer Register (SBUF)

Bit No.	7	6	5	4	3	2	1	0
99h	SBUF.7	SBUF.6	SBUF.5	SBUF.4	SBUF.3	SBUF.2	SBUF.1	SBUF.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Data for serial port is read from or written to this location. The serial transmit and receive buffers are separate registers, but both are addressed at the same location.

14.20 I²C Slave Control Register (I2C_SCON)

Bit No.	7	6	5	4	3	2	1	0
9Ah	-	WR	RD	BUSY	-	I2C_SIF	MODE	RUN
		R(0)	R(0)	R(0)		R/W(0)	R/W(0)	R/W(0)

Symbol	Function
WR	I2C Write Operation Status. Cleared by H/W.
RD	I2C Read Operation Status. Cleared by H/W.
BUSY	Current I2C slave status. Cleared by H/W. 1 = I2C slave is running now.
I2C_SIF	I2C Slave Interrupt Flag. It is set each time a byte is received or transmitted. Cleared by S/W.
MODE	I2C Slave Mode. 0 = Mode 0. Including Memory Address (Default). 1 = Mode 1. No Memory Address.
RUN	I2C Slave Start. Cleared by H/W.

14.21 I²C Slave Device Address Register (I2C_SDEV)

Bit No.	7	6	5	4	3	2	1	0
9Bh	SDEV.7	SDEV.6	SDEV.5	SDEV.4	SDEV.3	SDEV.2	SDEV.1	SDEV.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
SDEV[7;1]	I2C Slave Device Address.

SDEV[0]	Not Used. Don't Care.
---------	-----------------------

14.22 I²C Slave Memory Address Register (I2C_SADR)

Bit No.	7	6	5	4	3	2	1	0
9Ch	SADR.7	SADR.6	SADR.5	SADR.4	SADR.3	SADR.2	SADR.1	SADR.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
SADR[7;0]	I2C Slave memory Address.

14.23 I²C Slave Data Register (I2C_SDAT)

Bit No.	7	6	5	4	3	2	1	0
9Dh	SDAT.7	SDAT.6	SDAT.5	SDAT.4	SDAT.3	SDAT.2	SDAT.1	SDAT.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
SDAT[7;0]	I2C Slave Data.

14.24 I²C Master Data Register (I2C_MDAT)

Bit No.	7	6	5	4	3	2	1	0
9Eh	MDAT.7	MDAT.6	MDAT.5	MDAT.4	MDAT.3	MDAT.2	MDAT.1	MDAT.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
MDAT[7;0]	I2C Master Data.

14.25 Port 2 Register (P2)

Bit No.	7	6	5	4	3	2	1	0
A0h	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

This port functions as an address bus during external memory access, and as a general

purpose I/O port on devices which incorporate internal program memory. During external memory cycles, this port will contain the MSB of the address.

14.26 I²C Master Control Register (I2C_MCON)

Bit No.	7	6	5	4	3	2	1	0
A2h	-	-	-	I2C_MIF	OP	BYPASS	MODE	RUN
				R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
I2C_MIF	I2C Slave Interrupt Flag. It is set each time a byte is received or transmitted. Cleared by S/W.
OP	I2C Read/Write Operation. 0 = Write Operation (Default). 1 = Read Operation.
BYPASS	Bypass Mode in I2C Master and Slave
MODE	I2C Master Mode. 0 = Mode 0. Including Memory Address (Default). 1 = Mode 1. No Memory Address.
RUN	I2C Master Start. Cleared by H/W.

14.27 I²C Master Device Address Register (I2C_MDEV)

Bit No.	7	6	5	4	3	2	1	0
A3h	MDEV.7	MDEV.6	MDEV.5	MDEV.4	MDEV.3	MDEV.2	MDEV.1	MDEV.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
MDEV[7;1]	I2C Master Device Address.
MDEV[0]	Not Used. Don't Care.

14.28 I²C Master Memory Address Register (I2C_MADR)

Bit No.	7	6	5	4	3	2	1	0
A4h	MADR.7	MADR.6	MADR.5	MADR.4	MADR.3	MADR.2	MADR.1	MADR.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
SDAT[7:0]	I2C Master Address.

14.29 I²C Master Multi-byte Number Register (I2C_MNUM)

Bit No.	7	6	5	4	3	2	1	0
A4h	MNUM.7	MNUM.6	MNUM.5	MNUM.4	MNUM.3	MNUM.2	MNUM.1	MNUM.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This 8-bit special function register is load with the number of bytes to communicate. It is (I2CMNUM + 1).

14.30 I²C Master Clock Scale Factor High Byte Register (I2C_MSCH)

Bit No.	7	6	5	4	3	2	1	0
A4h	MSCH.7	MSCH.6	MSCH.5	MSCH.4	MSCH.3	MSCH.2	MSCH.1	MSCH.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I²C Master Clock Scale Factor High Byte Register (A7h) and I²C Master Clock Scale Factor Low Byte Register (A6h) are used for I²C serial clock generation.

14.31 I²C Master Clock Scale Factor Low Byte Register (I2C_MSCL)

Bit No.	7	6	5	4	3	2	1	0
A4h	MSCL.7	MSCL.6	MNUM.5	MSCL.4	MSCL.3	MSCL.2	MSCL.1	MSCL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I²C Master Clock Scale Factor High Byte Register (A7h) and I²C Master Clock Scale Factor Low Byte Register (A6h) are used for I²C serial clock generation. The I²C serial clock frequencies are: $F_{OSC} / \{(I2C_MSCH, I2C_MSCL+1)*4\}$. The I²C serial clock pulses are generated only in the master mode.

14.32 Interrupt Enable Register (IE)

Bit No.	7	6	5	4	3	2	1	0
A8h	EA	EADC	ET2	ES	ET1	EX1	ET0	EX0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
EA	Global Interrupt Enable. This bit controls the global masking of all interrupts except Power-Fail interrupt, which is enabled by the EPFI bit (WDCON.5). 0: Disable all interrupt sources. This bit overrides individual interrupt mask settings. 1: Enable all individual interrupt masks. Individual interrupts will occur if enabled.
EADC	Enable ADC Interrupt. This bit controls the masking of the ADC interrupt. 0: Disable all ADC interrupts. 1: Enable interrupt requests generated by the ADCF flag (ADCON.4)
ET2	Enable Timer 2 Interrupt. This bit controls the masking of the Timer 2 interrupt. 0: Disable all Timer 2 interrupts. 1: Enable interrupt requests generated by the TF2 flag (T2CON.7)
ES0	Enable Serial port Interrupt. This bit controls the masking of the serial port interrupt. 0: Disable all serial port interrupts. 1: Enable interrupt requests generated by the RI (SCON.0) or TI (SCON.1) flags
ET1	Enable Timer 1 Interrupt. This bit controls the masking of the Timer 1 interrupt. 0: Disable all Timer 1 interrupts. 1: Enable interrupt requests generated by the TF1 flag (TCON.7).
EX1	Enable External Interrupt 1. This bit controls the masking of external interrupt 1. 0: Disable external interrupt 1. 1: Enable interrupt requests generated by the /INT1 pin.
ET0	Enable Timer 0 Interrupt. This bit controls the masking of the Timer 0 interrupt. 0: Disable all Timer 0 interrupts. 1: Enable interrupt requests generated by the TF0 flag (TCON.5).
EX0	Enable External Interrupt 0. This bit controls the masking of external interrupt 0. 0: Disable external interrupt 0. 1: Enable interrupt requests generated by the /INT0 pin.

14.33 Port 3 Register (P3)

Bit No.	7	6	5	4	3	2	1	0
B0h			P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
			R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

This register functions as a general purpose I/O port.

14.34 Interrupt Priority Register (IP)

Bit No.	7	6	5	4	3	2	1	0
B8h	-	PADC	-	PS	PT1	PX1	PT0	PX0
	R(1)	R/W(0)		R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
-	Reserved.
PADC	ADC Interrupt Priority. This bit controls the priority of the ADC interrupt.
PS	Serial port Interrupt Priority. This bit controls the priority of the serial port interrupt.
PT1	Timer 1 Interrupt Priority. This bit controls the priority of the Timer 1 interrupt.
PX1	External Interrupt 1 Priority. This bit controls the priority of the external interrupt 1.
PT0	Timer 0 Interrupt Priority. This bit controls the priority of the Timer 0 interrupt.
PX0	External Interrupt 0 Priority. This bit controls the priority of the external interrupt 0.

14.35 Internal Ring Oscillator Control Register (OSCICN)

Bit No.	7	6	5	4	3	2	1	0
BEh	-	-	-	-	DIV2	RINGON	DIV1	DIV0
					R/W(1)	R/W(1)	R/W(0)	R/W(0)

Symbol	Function
RINGON	1 = Internal ring oscillator is running. 0 = Internal ring oscillator is killed. Don't clear RINGON bit when XTRG = 0.
DIV2, DIV1, DIV0	Ring oscillator divider. if Ring OSC. is 12MHz(@3V), [0,0,0] = 12MHz/1 (12.0 MHz) [0,0,1] = 12MHz/2 (6.0 MHz) [0,1,0] = 12MHz/4 (3.0 MHz) [0,1,1] = 12MHz/8 (1.5 MHz) [1,0,0] = 12MHz/3 (4.0 MHz) (Default) [1,0,1] = 12MHz/6 (2.0 MHz) [1,1,0] = 12MHz/12 (1.0 MHz)

	[1,1,1] = Reserved.
--	---------------------

14.36 Power Management Register (PMR)

Bit No.	7	6	5	4	3	2	1	0
C4h	-	-	-	-	-	XTOFF	-	-
	-	-	-	-	-	R/W(0)	-	-

Symbol	Function
-	Reserved.
XTOFF	Internal amplifier disable for external crystal oscillator. 1 = External crystal will be killed. 0 = External crystal will run (Default). Don't set XTOFF bit when XT/RG = 1.

14.37 Status Register (STATUS)

Bit No.	7	6	5	4	3	2	1	0
C5h	-	-	-	XTUP	-	-	-	-
	-	-	-	R(0)	-	-	-	-

Symbol	Function
-	Reserved.
XTUP	Crystal oscillator warm-up status. It represents the crystal clock is stable (1) or not (0). Cleared by H/W when Power-on reset and all kinds of reset. Cleared by H/W when XTOFF bit is set. Cleared by during Power-down wake-up when XT/RG = 1. Set by H/W after XTAL stabilization time.

14.38 Program Status Word Register (PSW)

Bit No.	7	6	5	4	3	2	1	0
D0h	CY	AC	F0	RS1	RS0	OV	F1	P
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R(0)

Symbol	Function																				
CY	Carry Flag. This bit is set when if the last arithmetic operation resulted in a carry (during addition) or a borrow (during subtraction). Otherwise it is cleared to 0 by all arithmetic operations.																				
AC	Auxiliary Carry Flag. This bit is set to 1 if the last arithmetic operation resulted in a carry into (during addition), or a borrow (during subtraction) from the high order nibble. Otherwise it is cleared to 0 by all arithmetic operations.																				
F0	User Flag 0. This is a general purpose flag for software control.																				
RS1, RS0	Register Bank Select. These bits select which register bank is addressed during register accesses.																				
	<table border="1"> <thead> <tr> <th>RS1</th> <th>RS0</th> <th>Register Bank</th> <th>Address</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>00H ~ 07H</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>08H ~ 0FH</td> </tr> <tr> <td>1</td> <td>0</td> <td>2</td> <td>10H ~ 17H</td> </tr> <tr> <td>1</td> <td>1</td> <td>3</td> <td>18H ~ 1FH</td> </tr> </tbody> </table>	RS1	RS0	Register Bank	Address	0	0	0	00H ~ 07H	0	1	1	08H ~ 0FH	1	0	2	10H ~ 17H	1	1	3	18H ~ 1FH
	RS1	RS0	Register Bank	Address																	
	0	0	0	00H ~ 07H																	
	0	1	1	08H ~ 0FH																	
1	0	2	10H ~ 17H																		
1	1	3	18H ~ 1FH																		
OV	Overflow Flag. This bit is set to 1 if the last arithmetic operation resulted in a carry (addition), borrow (subtraction), or overflow (multiply or divide). Otherwise it is cleared to 0 by all arithmetic operations.																				
F1	User Flag 1. This is a general purpose flag for software control.																				
P	Parity Flag. This bit is set to 1 if the modulo-2 sum of the eight bits of the accumulator is 1 (odd parity); and cleared to 0 on even parity.																				

14.39 Port 0 Type Control Register (P0TYPE)

Bit No.	7	6	5	4	3	2	1	0
D4h	P0TYPE.7	P0TYPE.6	P0TYPE.5	P0TYPE.4	P0TYPE.3	P0TYPE.2	P0TYPE.1	P0TYPE.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register determines the type of Port 0.
 0 = Push-Pull Output (Default) / 1 = Open-drain Output

14.40 Port 1 Type Control Register (P1TYPE)

Bit No.	7	6	5	4	3	2	1	0
D5h	P1TYPE.7	P1TYPE.6	P1TYPE.5	P1TYPE.4	P1TYPE.3	P1TYPE.2	P1TYPE.1	P1TYPE.0

R/W(0) R/W(0) R/W(0) R/W(0) R/W(0) R/W(0) R/W(0) R/W(0)

This register determines the type of Port 1.
0 = Push-Pull Output (Default) / 1 = Open-drain Output

14.41 Port 2 Type Control Register (P2TYPE)

Bit No.	7	6	5	4	3	2	1	0
D6h	P2TYPE.7	P2TYPE.6	P2TYPE.5	P2TYPE.4	P2TYPE.3	P2TYPE.2	P2TYPE.1	P2TYPE.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register determines the type of Port 2.
0 = Push-Pull Output (Default) / 1 = Open-drain Output

14.42 Port 3 Type Control Register (P3TYPE)

Bit No.	7	6	5	4	3	2	1	0
D7h	-	-	P3TYPE.5	P3TYPE.4	P3TYPE.3	P3TYPE.2	P3TYPE.1	P3TYPE.0
			R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register determines the type of Port 3.
0 = Push-Pull Output (Default) / 1 = Open-drain Output

14.43 Watchdog Control Register (WDCON)

Bit No.	7	6	5	4	3	2	1	0
D8h	WD1	WD0	EPFI	PFI	WDIF	WTRF	EWT	RWT
	R/W(1)	R/W(1)	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
WD1, WD0	Watchdog timer mode select. [0,0]: 1 x 2 ¹⁶ clocks (interrupt) + 256 clocks (reset) [0,1]: 4 x 2 ¹⁶ clocks (interrupt) + 256 clocks (reset) [1,0]: 16 x 2 ¹⁶ clocks (interrupt) + 256 clocks (reset) [1,1]: 32 x 2 ¹⁶ clocks (interrupt) + 256 clocks (reset)

EPFI	<p>Enable Power fail Interrupt. This bit enables/disables the ability of the internal band-gap reference to generate a power-fail interrupt when V_{CC} falls below approximately 4.0V. While in Stop mode, both this bit and the band-gap select bit BGS (EXIF.0) must be set to enable the power-fail interrupt.</p> <p>0: Power-fail interrupt disabled.</p> <p>1: Power-fail interrupt enabled during normal operation. Power-fail interrupt enabled in Stop mode if BGS is set.</p>			
PFI	<p>Power Fail Interrupt Flag. When set, this bit indicates that a power-fail interrupt has occurred. This bit must be cleared in software before exiting the interrupt service routine, or another interrupt will be generated. Setting this bit in software will generate a power-fail interrupt, if enabled.</p>			
WDIF	<p>Watchdog Interrupt Flag. This bit, in conjunction with the Watchdog timer interrupt enable bit EWDI (EIE.4) and Enable Watchdog Timer Reset bit EWT (WDCON.1) indicates if a watchdog timer event has occurred and what action will be taken. This bit must be cleared in software before exiting the interrupt service routine, or another interrupt will be generated. Setting this bit in software will generate a watchdog interrupt if enabled.</p>			
	EWT	EWDI	WDIF	Result
	X	X	0	No watchdog event has occurred.
	0	0	1	Watchdog time-out has expired. No interrupt has been generated.
	0	1	1	Watchdog interrupt has occurred.
	1	0	1	Watchdog time-out has expired. No interrupt has been generated. Watchdog timer reset will occur in 512 cycles if RWT is not strobed.
1	1	1	Watchdog interrupt has occurred. Watchdog timer reset will occur in 512 cycles if RWT is not set.	
WTRF	<p>Watchdog Timer Reset Flag. When set, this bit indicates that a watchdog timer reset has occurred. It is typically interrogated to determine if a reset was caused by watchdog timer reset. It is cleared by a power-on-reset, but otherwise must be cleared by software before the next reset of any kind or software may erroneously determine that a watchdog timer reset has occurred. Setting this bit in software will not generate a watchdog timer reset. If the EWT bit is cleared, the watchdog timer will have no effect on this bit.</p>			

EWT	<p>Enable Watchdog Timer Reset. This bit enables/disables the ability of the watchdog timer to reset the device. This bit has no effect on the ability of the watchdog timer to generate a watchdog interrupt. The time-out period of the watchdog timer is controlled by the Watchdog Timer Mode Select bits (CKCON.7-6). Clearing these bits will disable the ability of the watchdog timer to generate a reset, but have no affect on the timer itself, or its ability to generate a watchdog timer interrupt.</p> <p>0: A time-out of the watchdog timer will not cause the device to reset. 1: A time-out of the watchdog timer will cause the device to reset.</p>
RWT	<p>Reset Watchdog Timer. This bit serves as the strobe for the Watchdog function. During the time-out period, software must set the RWT bit if the Watchdog is enabled. Failing to set the RWT will cause a reset when the time-out has elapsed. There is no need to set the RWT bit to a 0 because it is self-clearing.</p>

14.44 ADC High Channel Selection Low Register (ADCHL)

Bit No.	7	6	5	4	3	2	1	0
D9h	ADCH7B	ADCH6B	ADCH5B	ADCH4B	ADCH3B	ADCH2B	ADCH1B	ADCH0B
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

ADCHXB = 0: ADCHX input enable (Digital input disable)

14.45 ADC High Channel Selection High Register (ADCHH)

Bit No.	7	6	5	4	3	2	1	0
DAh	ADCH15B	ADCH14B	ADCH13B	ADCH12B	ADCH11B	ADCH10B	ADCH9B	ADCH8B
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

ADCHXB = 0: ADCHX input enable (Digital input disable)

14.46 ADC High Channel Selection Register (ADCHSEL)

Bit No.	7	6	5	4	3	2	1	0
DBh	CH_SEL	-	-	-	CHH3	CHH2	CHH1	CHH0
	R/W(0)				R/W(1)	R/W(1)	R/W(1)	R/W(1)

Symbol	Function
--------	----------

CH_SEL	ADC MUX Selector with CHH[3:0] & CH[3:0]. 0 = CH[3:0] ◇ ADC[11:0] Enable / ADCH[15:0] Disable. 1 = CHH[3:0] ◇ ADC[11:0] Disable / ADCH[15:0] Enable.
CHH[3:0]	ADC MUX Selection for High Channel (ADCH[15:0]) 0000b : ADCH0 selection (0h) 1000b : ADCH8 selection (8h) 0001b : ADCH1 selection (1h) 1001b : ADCH9 selection (9h) 0010b : ADCH2 selection (2h) 1010b : ADCH10 selection (Ah) 0011b : ADCH3 selection (3h) 1011b : ADCH11 selection (Bh) 0100b : ADCH4 selection (4h) 1100b : ADCH12 selection (Ch) 0101b : ADCH5 selection (5h) 1101b : ADCH13 selection (Dh) 0110b : ADCH6 selection (6h) 1110b : ADCH14 selection (Eh) 0111b : ADCH7 selection (7h) 1111b : ADCH15 selection (Fh)

14.47 PWM Control Register (PWMCON)

Bit No.	7	6	5	4	3	2	1	0
DCh	POSEL	PS2_P0	PS1_P0	PS0_P0	-	PWMF	CLR_PO	RUN_PO
	R/W(0)	R/W(0)	R/W(0)	R/W(0)		R/W(0)	R/W(0)	R/W(0)

Symbol	Function																																				
POSEL	PWM0 Waveform Output. This bit enables/disables the PWM0 waveform output to Port 0.6.																																				
PS2_P0 PS1_P0 PS0_P0	Prescaled Clock Selection. * PWM Clock (FPWM) to ADC should not be set to $F_{osc}/1$.																																				
	<table border="1"> <thead> <tr> <th>PS2_P0</th> <th>PS1_P0</th> <th>PS0_P0</th> <th>Result</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>$F_{osc}/1$ divide</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>$F_{osc}/2$ divide</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>$F_{osc}/4$ divide</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>$F_{osc}/8$ divide</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>$F_{osc}/16$ divide</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>$F_{osc}/32$ divide</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>$F_{osc}/64$ divide</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>$F_{osc}/128$ divide</td> </tr> </tbody> </table>	PS2_P0	PS1_P0	PS0_P0	Result	0	0	0	$F_{osc}/1$ divide	0	0	1	$F_{osc}/2$ divide	0	1	0	$F_{osc}/4$ divide	0	1	1	$F_{osc}/8$ divide	1	0	0	$F_{osc}/16$ divide	1	0	1	$F_{osc}/32$ divide	1	1	0	$F_{osc}/64$ divide	1	1	1	$F_{osc}/128$ divide
	PS2_P0	PS1_P0	PS0_P0	Result																																	
	0	0	0	$F_{osc}/1$ divide																																	
	0	0	1	$F_{osc}/2$ divide																																	
	0	1	0	$F_{osc}/4$ divide																																	
	0	1	1	$F_{osc}/8$ divide																																	
	1	0	0	$F_{osc}/16$ divide																																	
	1	0	1	$F_{osc}/32$ divide																																	
1	1	0	$F_{osc}/64$ divide																																		
1	1	1	$F_{osc}/128$ divide																																		
PWMF	PWM interrupt flag. Cleared by S/W.																																				
CLR_PO	Counter Reset Enable. This bit is cleared by hardware.																																				

RUN_P0	Counter Start Enable. PWM clock (FPWM) output enable.
--------	---

14.48 PWM0 Duty Data Register (PWMD)

Bit No.	7	6	5	4	3	2	1	0
DEh	PWM0D.7	PWM0D.6	PWM0D.5	PWM0D.4	PWM0D.3	PWM0D.2	PWM0D.1	PWM0D.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

PWM0D duty data register, located in DEh, determines the duty of the output value generated by each 8-bit PWM circuit. In 8-bit counter mode, to program the required PWM output, you load the appropriate initialization values into the 8-bit data register (PWM0D), and in (6+2)-bit counter mode, you load the appropriate initialization values into the 6-bit reference data register (PWM0D[5:0]) and the 2-bit extension data register (PWM0D[7:6]).

14.49 Accumulator (ACC/A)

Bit No.	7	6	5	4	3	2	1	0
E0h	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register serves as the accumulator for arithmetic operations.

14.50 ADC Channel Selection High Register (ADCSELH)

Bit No.	7	6	5	4	3	2	1	0
E1h	ADC11B	ADC10B	ADC9B	ADC8B	ADC7B	ADC6B	ADC5B	ADC4B
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

Symbol	Function
ADC11B	0 = ADC11 input enable & digital input disable at P2.2.
ADC10B	0 = ADC10 input enable & digital input disable at P2.3.
ADC9B	0 = ADC9 input enable & digital input disable at P2.4.
ADC8B	0 = ADC8 input enable & digital input disable at P2.5.
ADC7B	0 = ADC7 input enable & digital input disable at P2.6.
ADC6B	0 = ADC6 input enable & digital input disable at P0.7.

ADC5B	0 = ADC5 input enable & digital input disable at P0.6.
ADC4B	0 = ADC4 input enable & digital input disable at P0.5.

14.51 ADC Channel Selection Low & MUX Selection Register (ADCSEL)

Bit No.	7	6	5	4	3	2	1	0
E2h	ADC3B	ADC2B	ADC1B	ADC0B	CH3	CH2	CH1	CH0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

Symbol	Function
ADC3B	0 = ADC3 input enable & digital input disable at P0.4.
ADC2B	0 = ADC2 input enable & digital input disable at P0.3.
ADC1B	0 = ADC1 input enable & digital input disable at P0.2.
ADC0B	0 = ADC0 input enable & digital input disable at P0.1.
CH[3:0]	ADC MUX Selection. [0,0,0,0] = ADC0 selection (0h) [0,0,0,1] = ADC1 selection (1h) [0,0,1,0] = ADC2 selection (2h) [0,0,1,1] = ADC3 selection (3h) [0,1,0,0] = ADC4 selection (4h) [0,1,0,1] = ADC5 selection (5h) [0,1,1,0] = ADC6 selection (6h) [0,1,1,1] = ADC7 selection (7h) [1,0,0,0] = ADC8 selection (8h) [1,0,0,1] = ADC9 selection (9h) [1,0,1,0] = ADC10 selection (Ah) [1,0,1,1] = ADC11 selection (Bh) [1,1,0,0] = No ADC input select (Ch) [1,1,0,1] = No ADC input select (Dh) [1,1,1,0] = No ADC input select (Eh) [1,1,1,1] = No ADC input select (Fh, Default)

14.52 Alternate Function Selection Register (ALTSEL)

Bit No.	7	6	5	4	3	2	1	0
E3h	IOXEN	IORSTEN	CLO	PWM00	TV0	TX	-	-
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)		

Symbol	Function
IOXEN	1 = XTAL and XTAL2 is configured as I/O. Must be XTOFF (PMR.3) = 1 (Oscillator Amp. Off)
IORSTEN	1 = RESETB is configured as I/O.
CLO	1 = System clock output to P2.6.

PWM00	1 = PWM waveform output enable to P0.0.
TVO	1 = Timer 0 overflow clock to P0.0.
TX	1 = UART TX data output to P0.2. User must set TX bit to use UART.

14.53 Port 0 Pull-up Control Register (P0SEL)

Bit No.	7	6	5	4	3	2	1	0
E4h	P0SEL.7	P0SEL.6	P0SEL.5	P0SEL.4	P0SEL.3	P0SEL.2	P0SEL.1	P0SEL.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

This register enables/disables the internal pull-up resistors in Port 0.

0: The internal pull-up resistors in Port 0 are ON.

1: The internal pull-up resistors in Port 0 are OFF.

14.54 Port 1 Pull-up Control Register (P1SEL)

Bit No.	7	6	5	4	3	2	1	0
E5h	P1SEL.7	P1SEL.6	P1SEL.5	P1SEL.4	P1SEL.3	P1SEL.2	P1SEL.1	P1SEL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register enables/disables the internal pull-up resistors in Port 1.

0: The internal pull-up resistors in Port 1 are ON.

1: The internal pull-up resistors in Port 1 are OFF.

14.55 Port 2 Pull-up Control Register (P2SEL)

Bit No.	7	6	5	4	3	2	1	0
E6h	P2SEL.7	P2SEL.6	P2SEL.5	P2SEL.4	P2SEL.3	P2SEL.2	P2SEL.1	P2SEL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register enables/disables the internal pull-up resistors in Port 2.

0: The internal pull-up resistors in Port 2 are ON.

1: The internal pull-up resistors in Port 2 are OFF.

14.56 Port 3 Pull-up Control Register (P3SEL)

Bit No.	7	6	5	4	3	2	1	0
E7h	-	-	P3SEL.5	P3SEL.4	P3SEL.3	P3SEL.2	P3SEL.1	P3SEL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register enables/disables the internal pull-up resistors in Port 3.

0: The internal pull-up resistors in Port 3 are ON.

1: The internal pull-up resistors in Port 3 are OFF.

14.57 External Interrupt Enable Register (EIE)

Bit No.	7	6	5	4	3	2	1	0
E8h	-	-	EPWM	EWDT	EI2C_S	EI2C_M	EX3	EX2
	-	-	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
-	Reserved.
EPWM	PWM Interrupt Enable.
EWDT	Watchdog timer interrupt Enable. This bit enables/disables the watchdog timer interrupt. 0: Disable the watchdog timer interrupt. 1: Enable the interrupt requests generated by the watchdog timer.
EI2C_S	I2C slave interrupt enable.
EI2C_M	I2C master interrupt enable.
EX3	External Interrupt 3 Enable. This bit enables/disables external interrupt 3. 0: Disable external interrupt 3. 1: Enable the interrupt requests generated by the /INT3 pin.
EX2	External Interrupt 2 Enable. This bit enables/disables external interrupt 2. 0: Disable external interrupt 2. 1: Enable the interrupt requests generated by the INT2 pin.

14.58 ADC Result Value High Register (ADCR)

Bit No.	7	6	5	4	3	2	1	0
EEh	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

[9:2] of ADC conversion result will be stored into ADCR. LSB of conversion result will be stored into ADCON[0].

14.59 ADC Control Register & ADC Result Low Register (ADCON)

Bit No.	7	6	5	4	3	2	1	0
EFh	AD_EN	AD_REQ	AD_END	ADCF	-	ADIV	SAR1	SAR0
	R/W(0)	R/W(0)	R(1)	R/W(0)		R/W(0)	R/W(0)	R/W(0)

Symbol	Function
AD_EN	ADC Ready Enable.
AD_REQ	ADC Start Enable. Cleared by H/W when AD_END goes to 1 from 0.
AD_END	Cleared by H/W when ADC conversion start. Set by H/W when ADC conversion has been finished. 0: Data conversion is now running. 1: ADC is idle state.
ADCF	ADC Interrupt Flag. This bit must be cleared by S/W.
-	Reserved.
ADIV	ADC input clock select. 0 = System clock (FOSC) / 2. (Default) 1 = PWM input clock (FPWM)
SAR1, SAR0	LSBs of ADC result value

14.60 B Register (B)

Bit No.	7	6	5	4	3	2	1	0
F0h	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

This register serves as a second accumulator for certain arithmetic operations.

14.61 Port 0 Input/Output Control Register (P0DIR)

Bit No.	7	6	5	4	3	2	1	0
F4h	P0DIR.7	P0DIR.6	P0DIR.5	P0DIR.4	P0DIR.3	P0DIR.2	P0DIR.1	P0DIR.0

R/W(1) R/W(1) R/W(1) R/W(1) R/W(1) R/W(1) R/W(1) R/W(1)

This register determines the I/O direction of Port 0.

P0DIR.X = 1 Input (Default) / 0 Output

14.62 Port 1 Input/Output Control Register (P1DIR)

Bit No.	7	6	5	4	3	2	1	0
F5h	P1DIR.7	P1DIR.6	P1DIR.5	P1DIR.4	P1DIR.3	P1DIR.2	P1DIR.1	P1DIR.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

This register determines the I/O direction of Port 1.

P1DIR.X = 1 Input (Default) / 0 Output

14.63 Port 2 Input/Output Control Register (P2DIR)

Bit No.	7	6	5	4	3	2	1	0
F6h	P2DIR.7	P2DIR.6	P2DIR.5	P2DIR.4	P2DIR.3	P2DIR.2	P2DIR.1	P2DIR.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

This register determines the I/O direction of Port 2.

P2DIR.X = 1 Input (Default) / 0 Output

14.64 Port 3 Input/Output Control Register (P3DIR)

Bit No.	7	6	5	4	3	2	1	0
F7h	-	-	P3DIR.5	P3DIR.4	P3DIR.3	P3DIR.2	P3DIR.1	P3DIR.0
			R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

This register determines the I/O direction of Port 3.

P3DIR.X = 1 Input (Default) / 0 Output

14.65 Extended Interrupt Priority Register (EIP)

Bit No.	7	6	5	4	3	2	1	0
F8h	-	-	PPWM	PWDT	PI2C_S	PI2C_M	PX3	PX2
	-	-	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Symbol	Function
-	Reserved.
PPWM	PWM interrupt priority bit.
PWDT	Watchdog timer Interrupt Priority. This bit controls the priority of the watchdog timer interrupt. 0: The Watchdog timer interrupt is a low priority interrupt. 1: The Watchdog timer interrupt is a high priority interrupt.
PI2C_S	I2C slave interrupt priority bit.
PI2C_M	I2C master interrupt priority bit.
PX3	External Interrupt 3 Priority. This bit controls the priority of the external interrupt 3. 0: The external interrupt 3 is a low priority interrupt. 1: The external interrupt 3 is a high priority interrupt.
PX2	External Interrupt 2 Priority. This bit controls the priority of the external interrupt 2. 0: The external interrupt 2 is a low priority interrupt. 1: The external interrupt 2 is a high priority interrupt.

14.66 EEPROM Access Enable (EEAEN)

Bit No.	7	6	5	4	3	2	1	0
FFh	-	-	-	-	-	-	-	EAEN
								R/W(0)

EAEN = 1, EEPROM access enable.