1. General Description

MG84FL54B is an enhanced single-chip 8-bit microcontroller manufactured in an advanced Embedded-Flash process. The instruction set is fully compatible with that of the 8051. With the enhanced CPU core, the device needs only 1 to 7 clock cycles to complete an instruction, and thus provides much higher performance than the standard 8051, which needs 12 to 48 clock cycles to complete an instruction. So, at the same performance as the standard 8051, the device can operate at a much lower speed and thereby greatly reduce the power consumption.

The device has on-chip 16KB Flash memory that is parallel programmable (via a universal programmer), In-System Programmable (via USB DFU). ISP allows the device to alter its own program memory without being removed from the actual end product under software control. This opens up a range of applications that need the ability to field update the application firmware. The other important and useful feature, In-Application-Programming (IAP), provides the device with the ability to save non-volatile data in its Flash memory.

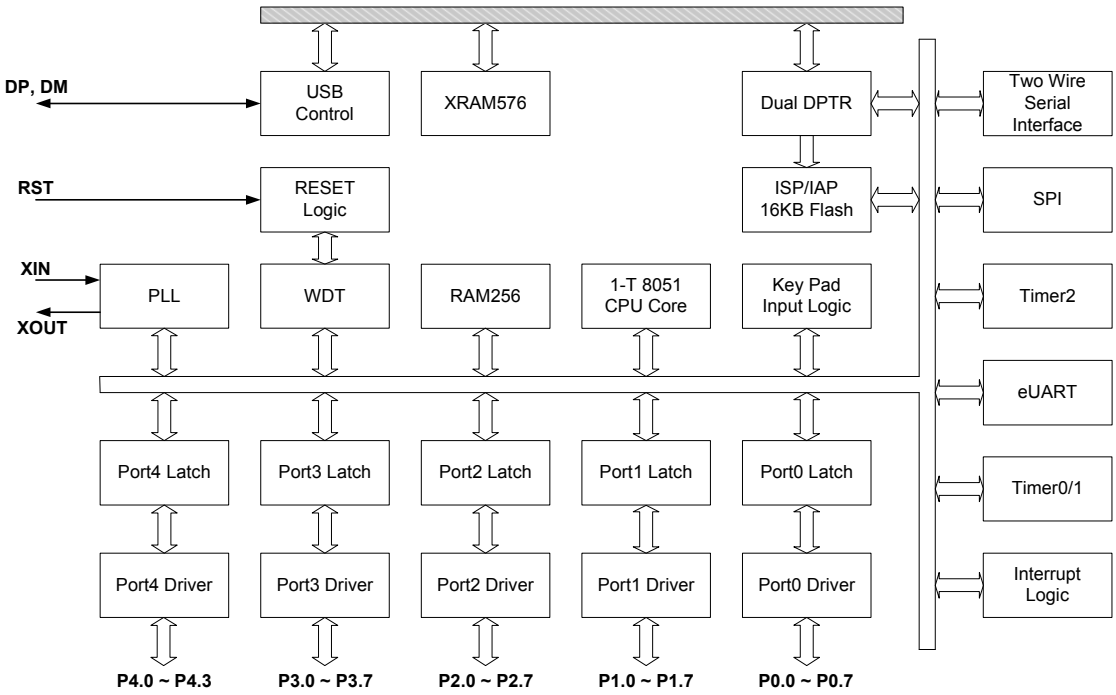
And in addition to the 256 bytes of internal scratch-pad RAM, the device has 576 bytes of on-chip expanded RAM (XRAM) for the applications that require extra memory. The device has also four 8-bit I/O ports and one 4-bit I/O ports, three 16-bit timers/counters, a multi-source/two-priority-level/nested interrupt structure, an enhanced UART input. More important, the added features such as KBI, SPI, TWSI bus and USB1.1 make it a powerful microcontroller and suitable for wide field applications.

2. Features

- 1-T 8051 CPU Core
- 16K bytes of on-chip Flash program memory with ISP/IAP function
- 256 bytes internal scratch-pad RAM and 576 bytes on-chip expanded RAM (XRAM)
- Dual DPTR (Data Pointer register)
- Four and half configurable I/O ports
- Three 16-bits Timers
- Enhanced UART
- Two-priority-level interrupt structure
- Additional external interrupts, INT2 and INT3
- Keypad interrupt (P0)
- Wake-up from power-down mode
- Serial Peripheral Interface (SPI)
- 2-wire Serial Interface (TWSI)
- One-time-enabled Watch-dog Timer (WDT)
- Programmable system clock
- USB specification 2.0 and 1.1 compliant
  - Built in full speed (12Mbps) USB transceiver
  - Intel 8X931 like USB control flow
  - One 256 bytes FIFO for USB endpoint-shared buffer
    - Maximum 64 bytes data for EP0 control-in/out buffer
    - Maximum 64 bytes data for EP1 bulk/interrupt-in buffer
    - Maximum 64 bytes data for EP2 bulk/interrupt/isochronous-in buffer, it could be configured to two 32 bytes dual-buffer-mode in bulk and isochronous operating.
    - Maximum 64 bytes data for EP3 bulk/interrupt/isochronous-out buffer, it could be configured to two 32 bytes dual-buffer-mode in bulk and isochronous operating. Additionally, it also can be configured to an interrupt-in buffer on EP3 function.
  - Supports USB suspend/resume and remote wake-up
  - Software-controlled USB connection/disconnection mechanism
  - Support USB DFU (Device Firmware Update)
- Power saving modes
  - Idle mode
  - Power-down mode
- Operating voltage
  - 2.4 ~ 5.5V on VDD_IO, 2.7V ~ 3.6V on VDD_CORE and VDD_PLL, 3.0V~3.6V on VDDA.
  - Built-in Low-Voltage Reset circuit.
- Operating temperature
  - Industrial (-40°C to +85°C)*
- Maximum operating frequency
  - Up to 24MHz, Industrial range
- Flash Quality criterion:
  - Flash data endurance: 20K erase/write cycles
  - Flash data retention: 100 years under room temperature
- 2-level code protection: SB (code scrambled) & LOCK (code locked)
- Package: LQFP-48

*: Tested by sampling.

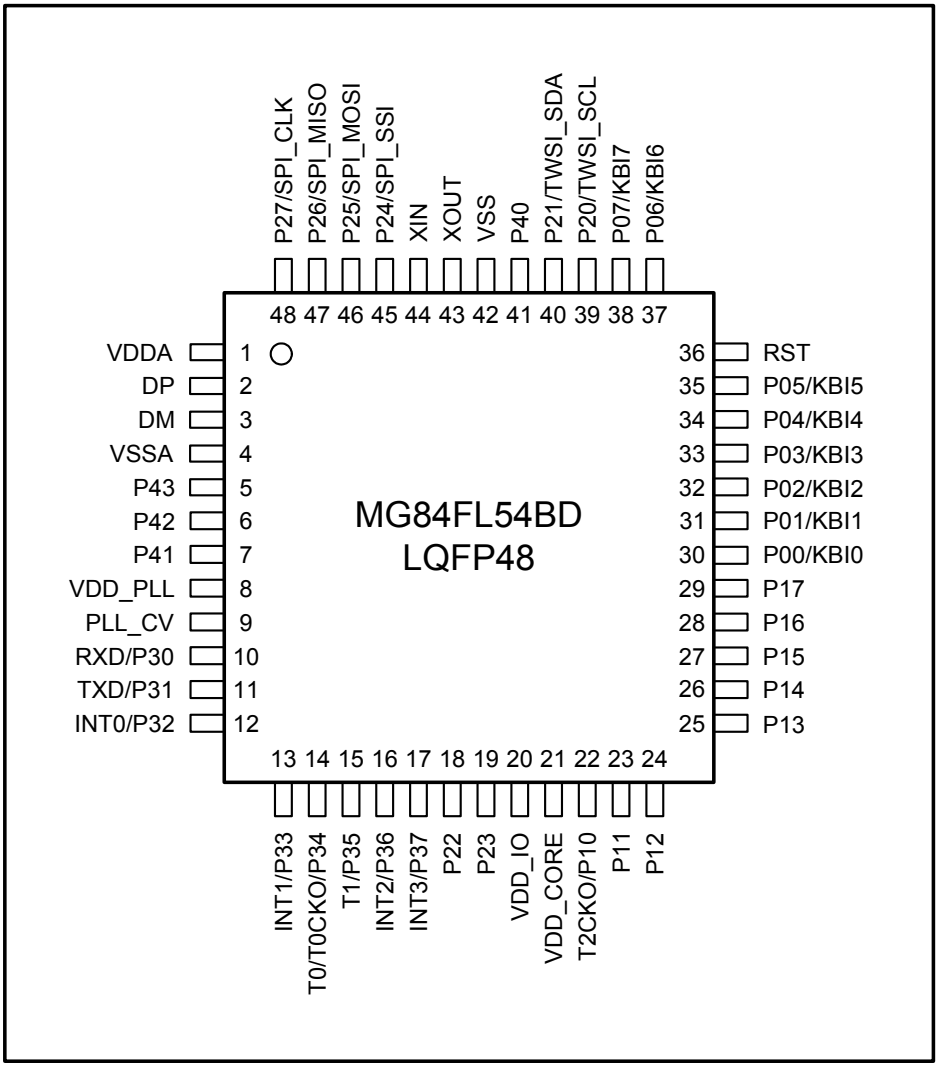
3. Block Diagram



MG84FL54B Block Diagram

4. Pin Configurations

4.1. Pin-out for 48-pin Package



4.2. Pin Description

Pin No.	Name	Type	Description
1	VDDA	P	3.3V Analog Power
2	DP	I/O	USB DP I/O.
3	DM	I/O	USB DM I/O.
4	VSSA	P	Analog Ground.
5	P4.3	I/O	P4.3
6	P4.2	I/O	P4.2
7	P4.1	I/O	P4.1
8	VDD_PLL	P	Power input of PLL.
9	PLL_CV	I/O	Reference for internal PLL.
10	P3.0 /RXD	I/O	P3.0 & Serial port RXD.
11	P3.1 /TXD	I/O	P3.1 & Serial port TXD.
12	P3.2 /INT0	I/O	P3.2 & External interrupt 0.
13	P3.3 /INT1	I/O	P3.3 & External interrupt 1.
14	P3.4 /T0 /T0CKO	I/O	P3.4, Timer 0 external input & Timer 0 clock output.
15	P3.5 /T1	I/O	P3.5 & Timer 1 external input.
16	P3.6 /INT2	I/O	P3.6 & External interrupt 2.
17	P3.7 /INT3	I/O	P3.7 & External interrupt 3.
18	P2.2	I/O	P2.2.
19	P2.3	I/O	P2.3.
20	VDD_IO	P	Digital power for I/O pads.
21	VDD_CORE	P	Digital power for I/O internal core logic.
22	P1.0 /T2CKO	I/O	P1.0 & Timer 2 clock output.
23	P1.1	I/O	P1.1
24	P1.2	I/O	P1.2.
25	P1.3	I/O	P1.3.
26	P1.4	I/O	P1.4.
27	P1.5	I/O	P1.5.
28	P1.6	I/O	P1.6.
29	P1.7	I/O	P1.7.
30	P0.0	I/O	P0.0 & Keypad input 0.
31	P0.1	I/O	P0.1 & Keypad input 1.
32	P0.2	I/O	P0.2 & Keypad input 2.
33	P0.3	I/O	P0.3 & Keypad input 3.

34	P0.4	I/O	P0.4 & Keypad input 4.
35	P0.5	I/O	P0.5 & Keypad input 5.
36	RST	I	System reset input, high active.
37	P0.6	I/O	P0.6 & Keypad input 6.
38	P0.7	I/O	P0.7 & Keypad input 7.
39	P2.0 /TWSI_SCL	I/O	P2.0 & TWSI_SCL.
40	P2.1 /TWSI_SDA	I/O	P2.1 & TWSI_SDA.
41	P4.0	I/O	P4.0
42	VSS	P	Digital ground.
43	XOUT	O	Crystal output pad.
44	XIN	I	Crystal input pad.
45	P2.4 /SPI_SSI	I/O	P2.4 & SPI_ SSI.
46	P2.5 /SPI_MOSI	I/O	P2.5 & SPI_MOSI.
47	P2.6 /SPI_MISO	I/O	P2.6 & SPI_MISO.
48	P2.7 /SPI_CLK	I/O	P2.7 & SPI_CLK.

5. Special Function Registers (SFRs)

5.1. SFR Mapping

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8H	**SICON**								FFH
F0H	**B**								F7H
E8H	**P4**								EFH
E0H	**ACC**	**WDTCR**	**IFD**	**IFADRH**	**IFADRL**		**SCMD**	**ISPCR**	E7H
D8H									DFH
D0H	**PSW**	**SIADR**	**SIDAT**	**SISTA**		**KBPATN**	**KBCON**	**KBMASK**	D7H
C8H	**T2CON**	**T2MOD**	**RCAP2L**	**RCAP2H**	**TL2**	**TH2**			CFH
C0H	**XICON**							**CKCON**	C7H
B8H	**IP**	**SADEN**						**CKCON2**	BFH
B0H	**P3**	**P3M0**	**P3M1**	**P4M0**	**P4M1**				B7H
A8H	**IE**	**SADDR**				**AUXIE**	**AUXIP**		AFH
A0H	**P2**						**AUXR2**	**TSTWD**	A7H
98H	**SCON**	**SBUF**							9FH
90H	**P1**	**P1M0**	**P1M1**	**P0M0**	**P0M1**	**P2M0**	**P2M1**		97H
88H	**TCON**	**TMOD**	**TL0**	**TL1**	**TH0**	**TH1**	**AUXR**		8FH
80H	**P0**	**SP**	**DPL**	**DPH**	**SPSTAT**	**SPCTL**	**SPDAT**	**PCON**	87H
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

5.2. The Standard 8051 SFRs

SYMBOL	DESCRIPTION	ADDR	BIT ADDRESS & SYMBOL								RESET VALUE
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
ACC*	Accumulator	**E0H**									00H
B*	B Register	**F0H**									00H
PSW*	Program Status Word	**D0H**	**D7H** CY	**D6H** AC	**D5H** F0	**D4H** RS1	**D3H** RS0	**D2H** OV	**D1H** -	**D0H** P	00H
SP	Stack Pointer	**81H**									07H
DPH	Data Pointer High	**83H**									00H
DPL	Data Pointer Low	**82H**									00H
P0*	Port 0	**80H**	**87H** P0.7 KBI7	**86H** P0.6 KBI6	**85H** P0.5 KBI5	**84H** P0.4 KBI4	**83H** P0.3 KBI3	**82H** P0.2 KBI2	**81H** P0.1 KBI1	**80H** P0.0 KBI0	FFH
P1*	Port 1	**90H**	**97H** P1.7	**96H** P1.6	**95H** P1.5	**94H** P1.4	**93H** P1.3	**92H** P1.2	**91H** P1.1	**90H** P1.0	FFH
P2*	Port 2	**A0H**	**A7H** P2.7 SPICLK	**A6H** P2.6 MISO	**A5H** P2.5 MOSI	**A4H** P2.4 /SS	**A3H** P2.3	**A2H** P2.2	**A1H** P2.1 SDA	**A0H** P2.0 SCL	FFH
P3*	Port 3	**B0H**	**B7H** P3.7 INT3	**B6H** P3.6 INT2	**B5H** P3.5 T1	**B4H** P3.4 T0	**B3H** P3.3 /INT1	**B2H** P3.2 /INT0	**B1H** P3.1 TXD	**B0H** P3.0 RXD	FFH
IP*	Interrupt Priority	**B8H**	**BFH** PX3	**BEH** PX2	**BDH** PT2	**BCH** PS	**BBH** PT1	**BAH** PX1	**B9H** PT0	**B8H** PX0	00H
IE*	Interrupt Enable	**A8H**	**AFH** EA	**AEH** -	**ADH** ET2	**ACH** ES	**ABH** ET1	**AAH** EX1	**A9H** ET0	**A8H** EX0	00H
TMOD	Timer Mode	**89H**	GATE	C/-T	M1	M0	GATE	C/-T	M1	M0	00H
TCON*	Timer Control	**88H**	**8FH** TF1	**8EH** TR1	**8DH** TF0	**8CH** TR0	**8BH** IE1	**8AH** IT1	**89H** IE0	**88H** IT0	00H
T2CON*	Timer 2 Control	**C8H**	**CFH** TF2	**CEH** EXF2	**CDH** RCLK	**CCH** TCLK	**CBH** EXEN2	**CAH** TR2	**C9H** C/-T2	**C8H** CP/-RL2	00H
TH0	Timer 0, High-byte	**8CH**									00H
TL0	Timer 0, Low-byte	**8AH**									00H
TH1	Timer 1, High-byte	**8DH**									00H
TL1	Timer 1, Low-byte	**8BH**									00H
TH2	Timer 2, High-byte	**CDH**									00H
TL2	Timer 2, Low-byte	**CCH**									00H
RCAP2H	Timer 2 Capture, High	**CBH**									00H
RCAP2L	Timer 2 Capture, Low	**CAH**									00H
SCON*	Serial Port Control	**98H**	**9FH** SM0/FE	**9EH** SM1	**9DH** SM2	**9CH** REN	**9BH** TB8	**9AH** RB8	**99H** TI	**98H** RI	00H
SBUF	Serial Data Buffer	**99H**									xxH
PCON	Power Control	**87H**	SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL	00H

Notes:

*: bit addressable

-: reserved bit

5.3. The Auxiliary SFRs

SYMBOL	DESCRIPTION	ADDR	BIT ADDRESS & SYMBOL								RESET VALUE
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
Interrupt											
XICON*	External Interrupt Control	**C0H**	**C7H** IL3	**C6H** EX3	**C5H** IE3	**C4H** IT3	**C3H** IL2	**C2H** EX2	**C1H** IE2	**C0H** IT2	00H
AUXIE	Auxiliary Interrupt Enable	**ADH**	EUSB	ETWSI	EKB	-	-	-	-	ESPI	00H
AUXIP	Auxiliary Interrupt Priority	**AEH**	PUSB	PTWSI	PKB	-	-	-	-	PSPI	00H
I/O Port											
P4*	Port 4	**E8H**	**EFH** -	**EEH** -	**EDH** -	**ECH** -	**EBH** P4.3	**EAH** P4.2	**E9H** P4.1	**E8H** P4.0	FFH
P0M0	Port 0 Mode Register 0	**93H**	P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0	00H
P0M1	Port 0 Mode Register 1	**94H**	P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0	00H
P1M0	Port 1 Mode Register 0	**91H**	P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0	00H
P1M1	Port 1 Mode Register 1	**92H**	P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0	00H
P2M0	Port 2 Mode Register 0	**95H**	P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0	00H
P2M1	Port 2 Mode Register 1	**96H**	P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0	00H
P3M0	Port 3 Mode Register 0	**B1H**	P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0	00H
P3M1	Port 3 Mode Register 1	**B2H**	P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	00H
P4M0	Port 4 Mode Register 0	**B3H**	-	-	-	-	P4M0.3	P4M0.2	P4M0.1	P4M0.0	00H
P4M1	Port 4 Mode Register 1	**B4H**	-	-	-	-	P4M1.3	P4M1.2	P4M1.1	P4M1.0	00H
Keypad Interrupt											
KBCON	Keypad Control	**D6H**	-	-	-	-	-	-	PTNS	KPI	00H
KBPATN	Keypad Pattern	**D5H**									FFH
KBMASK	Keypad Mask	**D7H**									00H
Serial Port											
SADEN	Slave Address Mask	**B9H**									00H
SADDR	Slave Address	**A9H**									00H
TWSI											
SICON*	TWSI Control Register	**F8H**	CR2	ENSI	STA	STO	SI	AA	CR1	CR0	00H
SIADR	TWSI Address Register	**D1H**	(Own Slave Address)							GC	00H
SIDAT	TWSI Data Register	**D2H**									00H
SISTA	TWSI Status Register	**D3H**									F8H
SPI											
SPCTL	SPI Control Register	**85H**	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	04H
SPSTAT	SPI Status Register	**84H**	SPIF	THRE	-	-	-	-	-	SPR2	00H
SPDAT	SPI Data Register	**86H**									00H
Others											
AUXR	Auxiliary Register	**8EH**	-	-	BRADJ0	-	T2X12	-	-	DPS	00H
AUXR2	Auxiliary Register 2	**A6H**	T0X12	T1X12	URM0x6	-	-	-	-	T0CKOE	00H
T2MOD	Timer 2 Mode Control	**C9H**	T2CPCF	-	DUTY1	DUTY0	FIXV	FIXEN	T2OE	DCEN	00H
CKCON	Clock Control	**C7H**	XCKS4	XCKS3	XCKS2	XCKS1	XCKS0	CKS2	CKS1	CKS0	28H
CKCON2	Clock Control 2	**BFH**	-	-	OSCDR0	-	EN_USB	EN_PLL	PLL_RDY	CK_SEL	00H
WDTCR	Watch-dog Timer	**E1H**	WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0	00H
ISP											
ISPCR	ISP Control Register	**E7H**	ISPEN	SWBS	SWRST	-	-	-	MS1	MS0	00H

IFADRH	ISP Flash Address High	**E3H**		00H
IFADRL	ISP Flash Address Low	**E4H**		00H
IFD	ISP Flash Data	**E2H**		FFH
SCMD	ISP Sequential Command	**E6H**		xxH

Notes:

*: bit addressable

-: reserved bit

6. Flash Memory Configuration

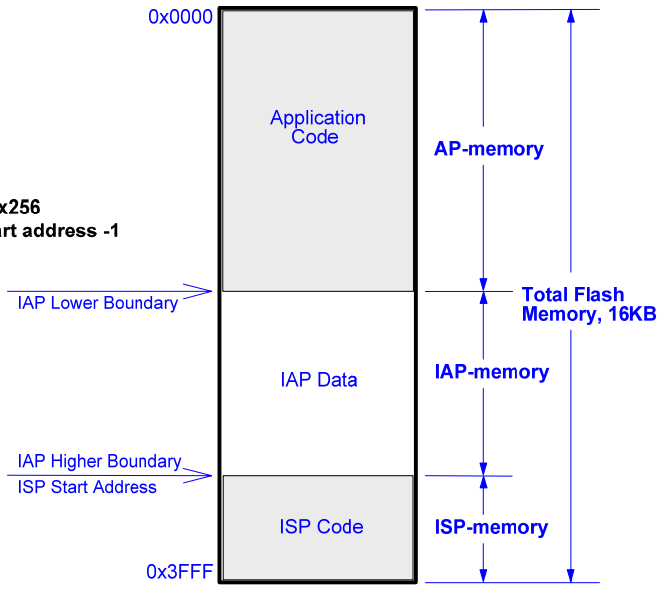
There are total 16K bytes of Flash Memory.

Note:

(1) ISP Start Address:

0x3000 if ISP=4KB
0x3200 if ISP=3.5KB
0x3400 if ISP=3KB
0x3600 if ISP=2.5KB
0x3800 if ISP=2KB
0x3A00 if ISP=1.5KB
0x3C00 if ISP=1KB

(2) IAP Lower Boundary = IAPLBx256
IAP Higher Boundary = ISP start address -1



Note: (**default**)

2KB ISP code present, 1KB IAP space configured, Lock enabled.

7. On-chip expanded RAM (XRAM)

In addition to the 256 bytes of scratch-pad RAM, there are extra 576 bytes of on-chip expanded RAM (XRAM) in USB MCU Mode. They may be accessed by the instructions “MOVX @Ri” and “MOVX @DPTR”.

Using the XRAM in Software

For KEIL-C51 compiler, to assign the variables to be located at XRAM, the “pdata” or “xdata” definition should be used. After being compiled, the variables declared by “pdata” and “xdata” will become the memories accessed by “MOVX @Ri” and “MOVX @DPTR”, respectively. Thus the BA126 hardware can access them correctly. The user can get the following descriptions from the “*Keil Software — Cx51 Compiler User’s Guide*”.

Explicitly Declared Memory Types

You may specify where variables are stored by including a memory type specifier in the variable declaration.

The following table summarizes the available memory type specifiers.

Memory Type	Description
code	Program memory (64 KBytes); accessed by opcode MOVC @A+DPTR.
data	Directly addressable internal data memory; fastest access to variables (128 bytes).
idata	Indirectly addressable internal data memory; accessed across the full internal address space (256 bytes).
bdata	Bit-addressable internal data memory; supports mixed bit and byte access (16 bytes).
xdata	External data memory (64 KBytes); accessed by opcode MOVX @DPTR.
far	Extended RAM and ROM memory spaces (up to 16MB); accessed by user defined routines or specific chip extensions (Philips 80C51MX, Dallas 390).
pdata	Paged (256 bytes) external data memory; accessed by opcode MOVX @R*n*.

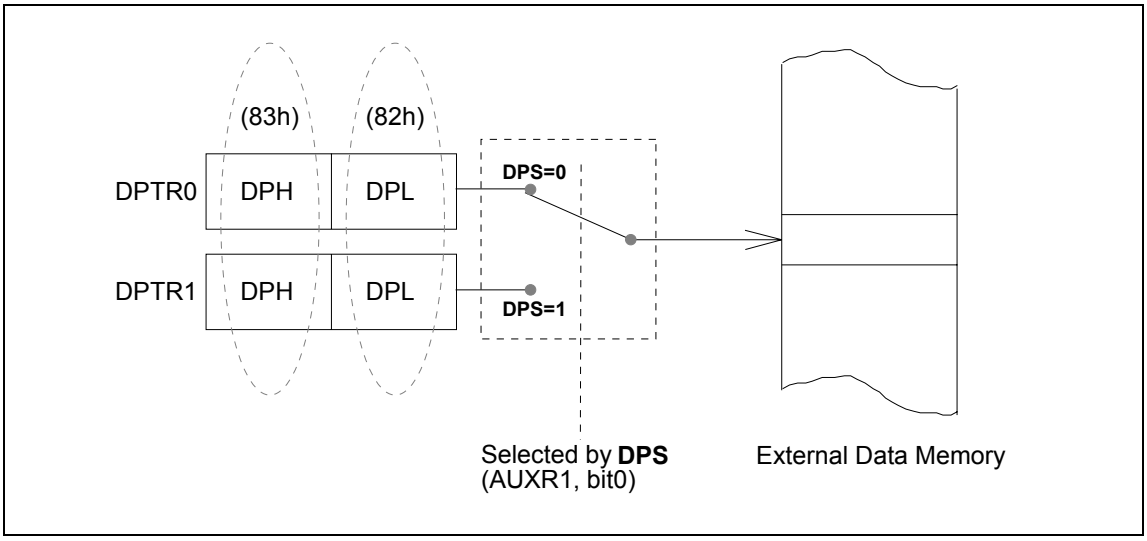
As with the **signed** and **unsigned** attributes, you may include memory type specifiers in the variable declaration.

Example:

```
char data var1;
char code text[] = "ENTER PARAMETER:";
unsigned long xdata array[100];
float idata x,y,z;
unsigned int pdata dimension;
unsigned char xdata vector[10][4][4];
char bdata flags;
```

8. Dual Data Pointer Register (DPTR)

The dual DPTR structure (see the following Figure) is a way by which the chip can specify the address of an external data memory location. There are two 16-bit DPTR registers that address the external memory, and a single bit called DPS (AUXR.0) that allows the program code to switch between them.



DPTR Instructions

The six instructions that refer to DPTR currently selected using the DPS bit are as follows:

```
INC  DPTR; Increments the data pointer by 1
MOV  DPTR,#data16 ; Loads the DPTR with a 16-bit constant
MOV  A,@A+DPTR ;Move code byte relative to DPTR to ACC
MOVX A,@DPTR ;Move external RAM (16-bit address) to ACC
MOVX @DPTR,A ;Move ACC to external RAM (16-bit address)
JMP @A+DPTR ;Jump indirect relative to DPTR
```

AUXR (Address=8EH, Auxiliary Register)

7	6	5	4	3	2	1	0
-	-	BRADJ0	-	T2X12	-	-	DPS

DPS: DPTR select bit, used to switch between DPTR0 and DPTR1.

The DPS bit status should be saved by software when switching between DPTR0 and DPTR1.

DPS		DPTR selected
0		DPTR0
1		DPTR1

9. Configurable I/O Ports

9.1. Port Configurations

The device has five I/O ports, Port 0 ~ Port 4. All the port pins can be individually and independently configured to one of four modes: *quasi-bidirectional (standard 8051 I/O port)*, *push-pull output*, *open-drain output* or *input-only (high-impedance)*. Each port pin is equipped with a Schmitt-triggered input to improve input noise rejection. Each port has two configuration registers, PxM0 and PxM1, to configure the I/O type for each port pin. Where, x=0~4.

Table : Port Configuration Settings

PxM0.y	PxM1.y	Port Mode
0	0	Quasi-bidirectional
0	1	Push-Pull Output
1	0	Input-Only (High Impedance, Hi-Z)
1	1	Open-Drain Output

Where x=0~4 (port number), and y=0~7 (port pin). The registers PxM0 and PxM1 are listed below.

P0M0 (Address=93H, Port 0 Mode Register 0)

7	6	5	4	3	2	1	0
P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0

P0M1 (Address=94H, Port 0 Mode Register 1)

7	6	5	4	3	2	1	0
P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0

P1M0 (Address=91H, Port 1 Mode Register 0)

7	6	5	4	3	2	1	0
P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0

P1M1 (Address=92H, Port 1 Mode Register 1)

7	6	5	4	3	2	1	0
P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0

P2M0 (Address=95H, Port 2 Mode Register 0)

7	6	5	4	3	2	1	0
P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0

P2M1 (Address=96H, Port 2 Mode Register 1)

7	6	5	4	3	2	1	0
P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0

P3M0 (Address=B1H, Port 3 Mode Register 0)

7	6	5	4	3	2	1	0
P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0

P3M1 (Address=B2H, Port 3 Mode Register 1)

7	6	5	4	3	2	1	0
P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0

P4M0 (Address=B3H, Port 4 Mode Register 0)

7	6	5	4	3	2	1	0
				P4M0.3	P4M0.2	P4M0.1	P4M0.0

P4M1 (Address=B4H, Port 4 Mode Register 1)

7	6	5	4	3	2	1	0
				P4M1.3	P4M1.2	P4M1.1	P4M1.0

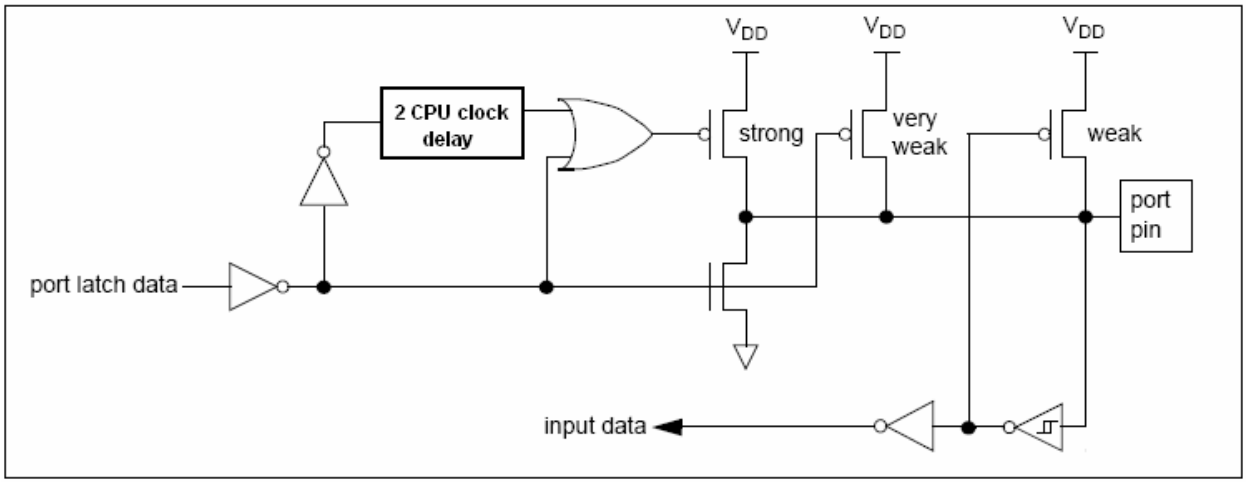
9.1.1. Quasi-bidirectional

Port pins in quasi-bidirectional mode are similar to the standard 8051 port pins. A quasi-bidirectional port can be used as an input and output without the need to reconfigure the port. This is possible because when the port outputs a logic high, it is weakly driven, allowing an external device to pull the pin low. When the pin outputs low, it is driven strongly and able to sink a large current. There are three pull-up transistors in the quasi-bidirectional output that serve different purposes.

One of these pull-ups, called the “very weak” pull-up, is turned on whenever the port register for the pin contains a logic “1”. This very weak pull-up sources a very small current that will pull the pin high if it is left floating.

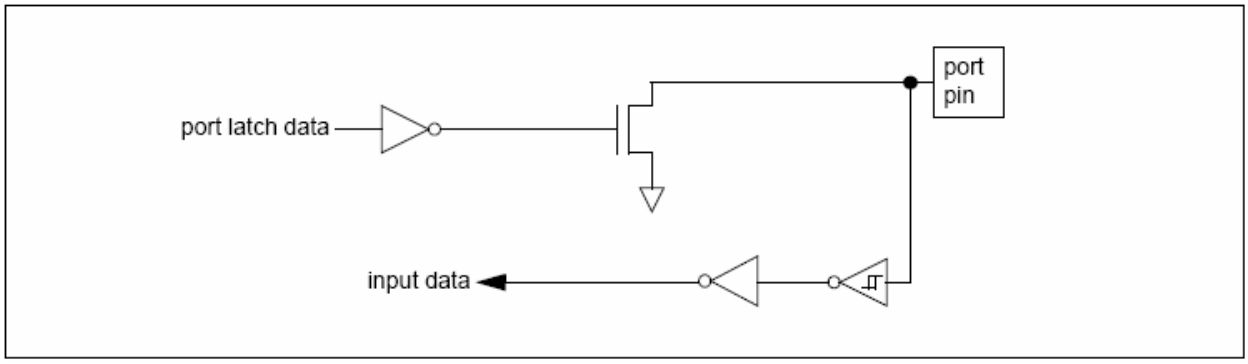
A second pull-up, called the “weak” pull-up, is turned on when the port register for the pin contains a logic “1” and the pin itself is also at a logic “1” level. This pull-up provides the primary source current for a quasi-bidirectional pin that is outputting a 1. If this pin is pulled low by the external device, this weak pull-up turns off, and only the very weak pull-up remains on. In order to pull the pin low under these conditions, the external device has to sink enough current to over-power the weak pull-up and pull the port pin below its input threshold voltage.

The third pull-up is referred to as the “strong” pull-up. This pull-up is used to speed up low-to-high transitions on a quasi-bidirectional port pin when the port register changes from a logic “0” to a logic “1”. When this occurs, the strong pull-up turns on for two CPU clocks, quickly pulling the port pin high.



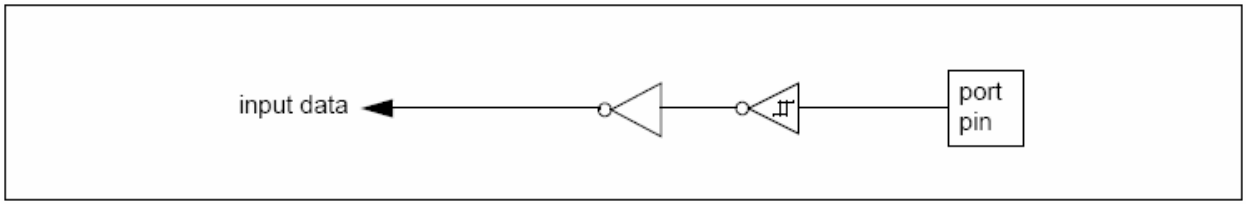
9.1.2. Open-Drain Output

The open-drain output configuration turns off all pull-ups and only drives the pull-down transistor of the port pin when the port register contains a logic “0”. To use this configuration in application, a port pin must have an external pull-up, typically a resistor tied to VDD. The pull-down for this mode is the same as for the quasi-bidirectional mode. In addition, the input path of the port pin in this configuration is also the same as quasi-bidirectional mode.



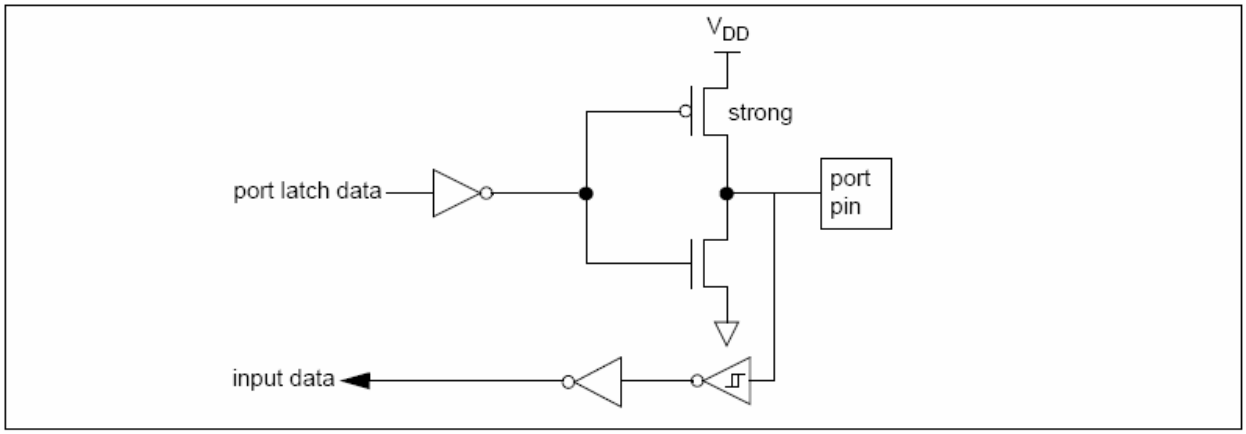
9.1.3. Input-Only (Hi-Z)

The input-only configuration is a Schmitt-triggered input without any pull-up resistors on the pin.



9.1.4. Push-Pull Output

The push-pull output configuration has the same pull-down structure as both the open-drain and the quasi-bidirectional output modes, but provides a continuous strong pull-up when the port register contains a logic “1”. The push-pull mode may be used when more source current is needed from a port output. In addition, the input path of the port pin in this configuration is also the same as quasi-bidirectional mode.



9.2. Maximum Ratings for Port Outputs

While port pins function as outputs (which can source or sink a current), to prevent the device from being permanently damaged, users should take care the total current *not more than 40mA* for sourcing or sinking regardless of a 3.3V device or a 5V device. That means that the device can source total 40mA and sink total 40mA at the same time without causing any damage to itself.

10. Three 16-bit Timers

10.1. Timer 0 and Timer 1

After power-up or reset, the default function and operation of Timer 0 and Timer 1 is fully compatible with the standard 8051 MCU. The only difference is that besides Fosc/12 the user can select an alternate clock source, the Fosc. The bit-7 and bit-6 in AUXR2 provide this selection.

AUXR2 (Address=A6H, Auxiliary Register 2)

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0x6	-	-	-	-	T0CKOE

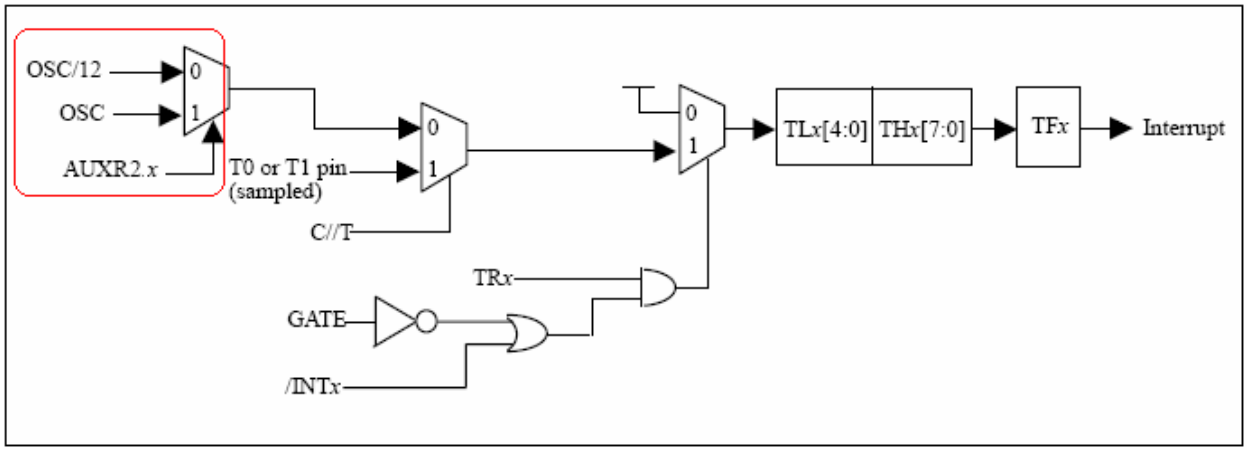
T0X12: Timer 0 clock source select while C/T=0.
Set to select Fosc as the clock source, and clear to select Fosc/12.

T1X12: Timer 1 clock source select while C/T=0.
Set to select Fosc as the clock source, and clear to select Fosc/12.

T0CKOE: Set to enable Timer 0 clock output on P3.4.

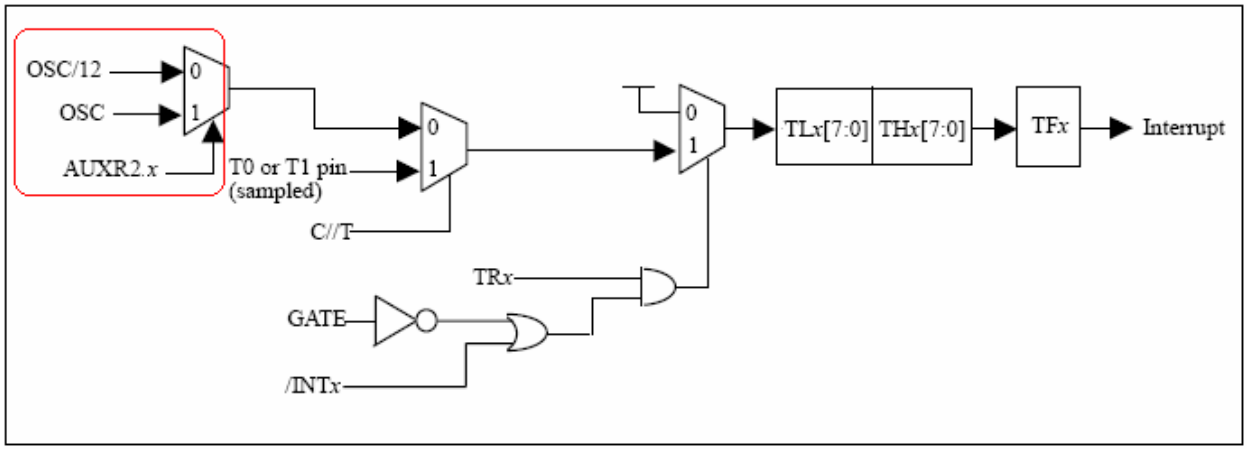
The following figures show the selection of alternate clock source for Timer 0 and Timer1.

10.1.1. Mode 0: 13-bit Counter



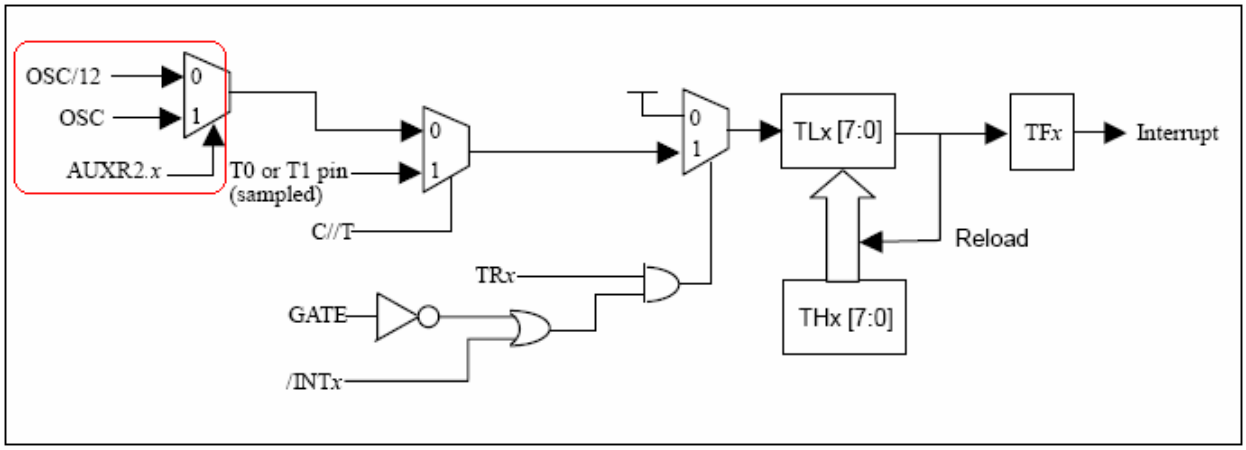
Where OSC means Fosc, the system clock.

10.1.2. Mode 1: 16-bit Counter



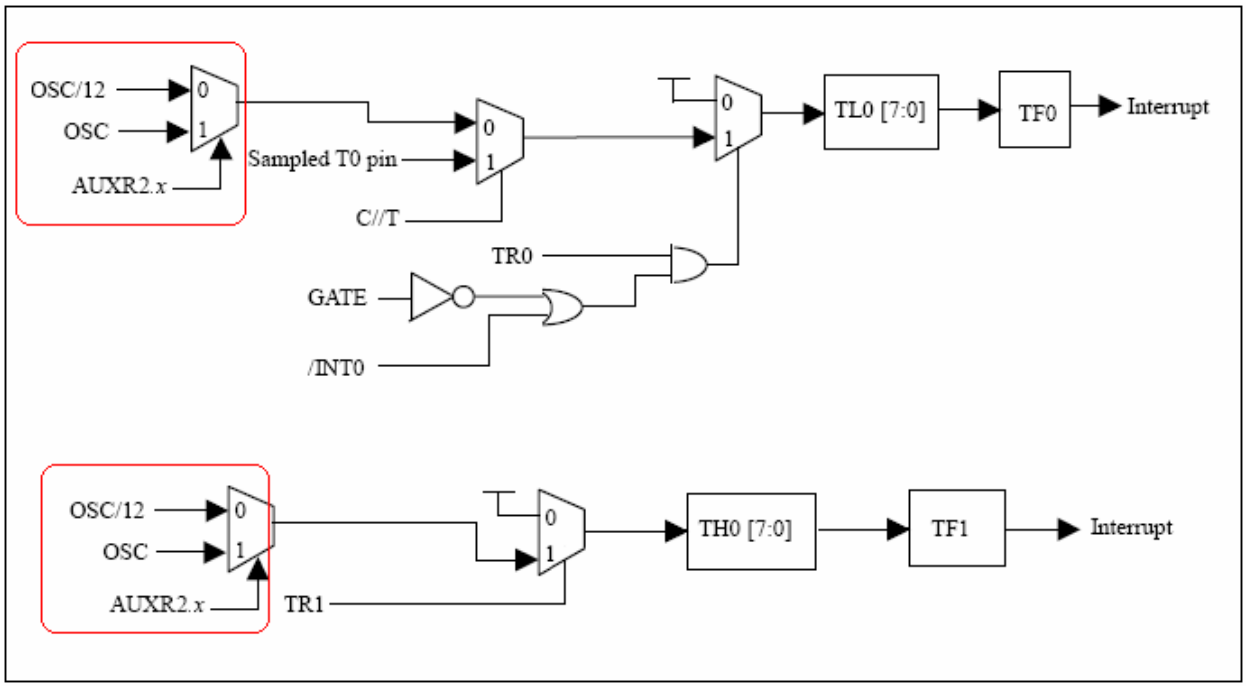
Where OSC means Fosc, the system clock.

10.1.3. Mode 2: 8-bit Auto-reload



Where OSC means Fosc, the system clock.

10.1.4. Mode 3: Timer 0 as Two 8-bit Counter



Where OSC means Fosc, the system clock.

10.1.5. Programmable Clock Output from Timer 0

The user can get a 50% duty-cycle clock output on P3.4 by configuring Timer 0 as 8-bit auto-reload and setting **T0CKOE** to "1". Of course, the bit TR0 (TCON.4) must also be set to start the timer. For a 12 MHz system clock, Timer 0 has a programmable output frequency range of 1953 Hz to 6 MHz.

The clock frequency is equal to (Timer 0 overflow rate / 2), that is

$$\text{Clock freq.} = \frac{\text{Fosc}}{\text{n x (256-TH0)}}$$

where,
n=24 if T0X12=0
n=2 if T0X12=1

(Fosc is the system clock.)

10.2. Timer 2

Three special function registers, AUXR, T2MOD and T2CON, are related to the operation of Timer 2, as listed below.

AUXR (Address=8EH, Auxiliary Register)

7	6	5	4	3	2	1	0
-	-	BRADJ0	-	T2X12	-	-	DPS

T2X12: Timer 2 clock source select while C/T2 (T2CON.1)=0 in *Capture Mode* and *Auto-Reload Mode*.
Set to select Fosc as the clock source, and clear to select Fosc/12.

T2MOD (Address=C9H, Timer 2 Mode Control register)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	T2OE	DCEN

T2OE: Timer 2 clock-out enable bit: 0 to disable, and 1 to enable.
DCEN: Timer 2 down-counting enable bit: 0 to disable, and 1 to enable.

T2CON (Address=C8H, Timer 2 Control Register)

7	6	5	4	3	2	1	0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/-RL2

TF2:
Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK=1 or TCLK=1.

EXF2:
Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX pin and EXEN2=1. When Timer 2 interrupt is enabled, EXF2=1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down mode (DCEN = 1).

RCLK:
Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for it's receive clock in modes 1 and 3. RCLK=0 causes Timer 1 overflow to be used for the receive clock.

TCLK:
Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for it's transmit clock in modes 1 and 3. TCLK=0 causes Timer 1 overflows to be used for the transmit clock.

EXEN2:
Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX pin if Timer 2 is not being used to clock the serial port. EXEN2=0 causes Timer 2 to ignore events at T2EX pin.

TR2:
Start/stop control for Timer 2. A logic 1 starts the timer.

C/T2:
Timer or counter select. When cleared, select internal timer, when set, select external event counter (falling edge triggered).

CP/-RL2:
Capture/Reload flag. When set, captures will occur on negative transitions at T2EX pin if EXEN2=1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX pin when EXEN2=1. When either RCLK=1 or TCLK=1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

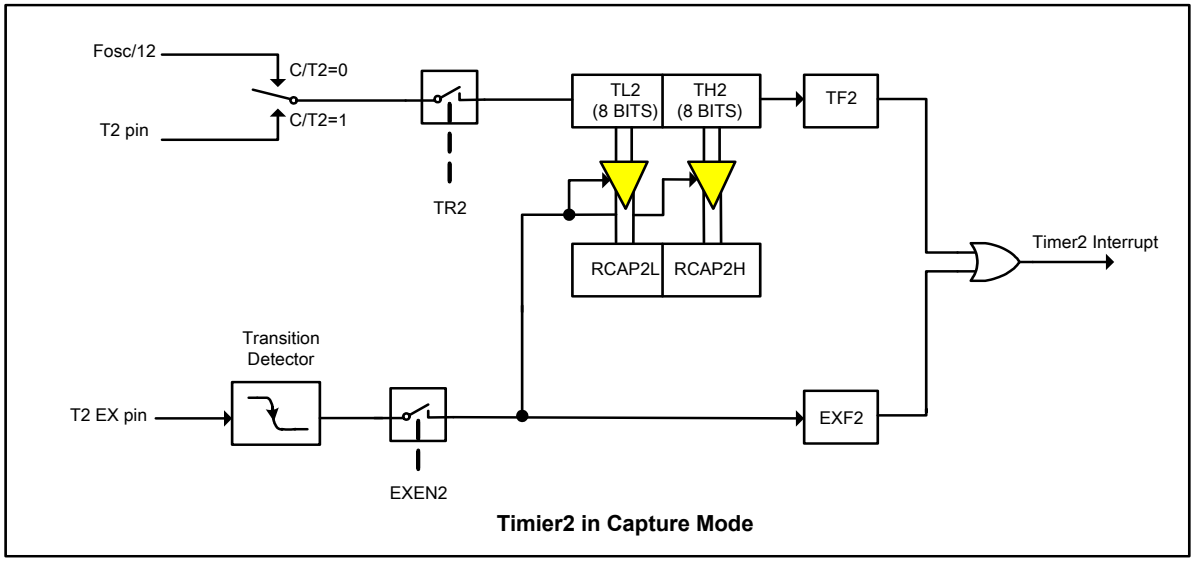
After reset, the ***DCEN=0*** (T2MOD.0), which makes the function of Timer 2 all the same as the standard 8052 (always counts up). While ***DCEN=1***, Timer 2 can count up or count down according to the logic level of the T2EX pin (P1.1). The following Table shows the operation modes of Timer 2.

Table: Timer 2 Mode

RCLK + TCLK	CP/-RL2	TR2	DCEN	T2OE	Mode
x	x	0	x	0	(off)
1	x	1	0	0	Baud-rate generator
0	1	1	0	0	16-bit capture
0	0	1	0	0	16-bit auto-reload (counting-up only)
0	0	1	**1**	0	16-bit auto-reload (counting-up or counting-down)
0	0	1	0	**1**	Clock output

10.2.1. Capture Mode (CP)

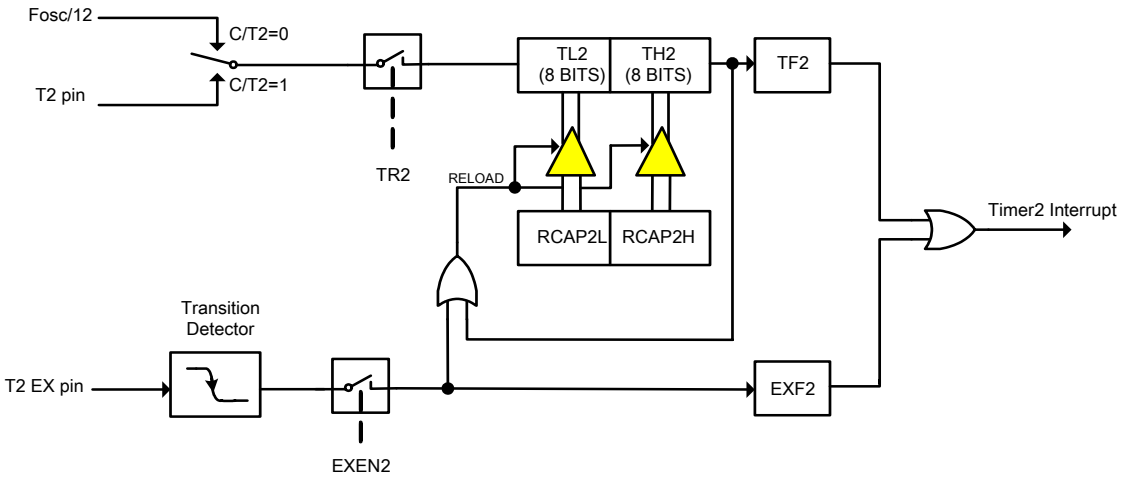
In the Capture mode, Timer2 is incremented by either Fosc/12 or external pin (T2) 1-to-0 transition. TR2 controls the event to timer2 and a 1-to-0 transition on T2EX pin will trigger RCAP2H and RCAP2L registers to capture the Timer2 contents onto them if EXEN2 is set. An overflow in Timer2 set TF2 flag and a 1-to-0 transition in T2EX pin sets EXF2 flag if EXEN2=1. TF2 and EXF2 is ORed to request the interrupt service.



Timier2 in Capture Mode

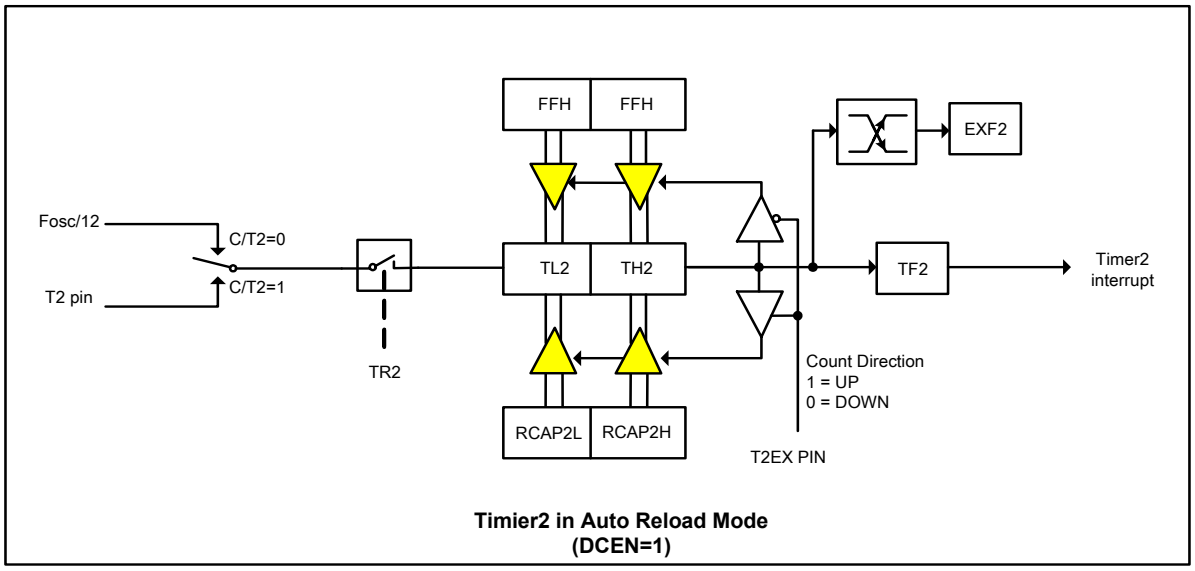
10.2.2. Auto-Reload Mode (AR)

In AR mode, Timer2 can be configured to count up or count down depending on DCEN bit in T2MOD register. When reset is applied(DCEN =0, CP/RL2=0), Timer2 is at auto-reload mode and only counting up is available. An overflow on Timer2 or 1-to-0 transition on T2EX pin will load RCAP2H and RCAP2L contents onto Timer2, also set TF2 and EXF2, respectively.



Timier2 in Auto Reload Mode (DCEN=0)

When DCEN =1 and in AR mode, Timer2 can be configured to count up or down . The counting direction is determined by T2EX pin. If T2EX=1, counting up, otherwise counting down. An overflow on Timer2 will set TF2 and toggle EXF2. EXF2 can not generate interrupt request in auto-reload mode with DCEN=1. If the counting direction is DOWN, Timer2 is loaded with 0xFFFF when the content of Timer2 equals to the values stored on RCAP2H and RCAP2L. But if counting direction is UP, an overflow of timer2 loads RCAP2H,RCAP2L contents onto Timer2.



Timier2 in Auto Reload Mode
(DCEN=1)

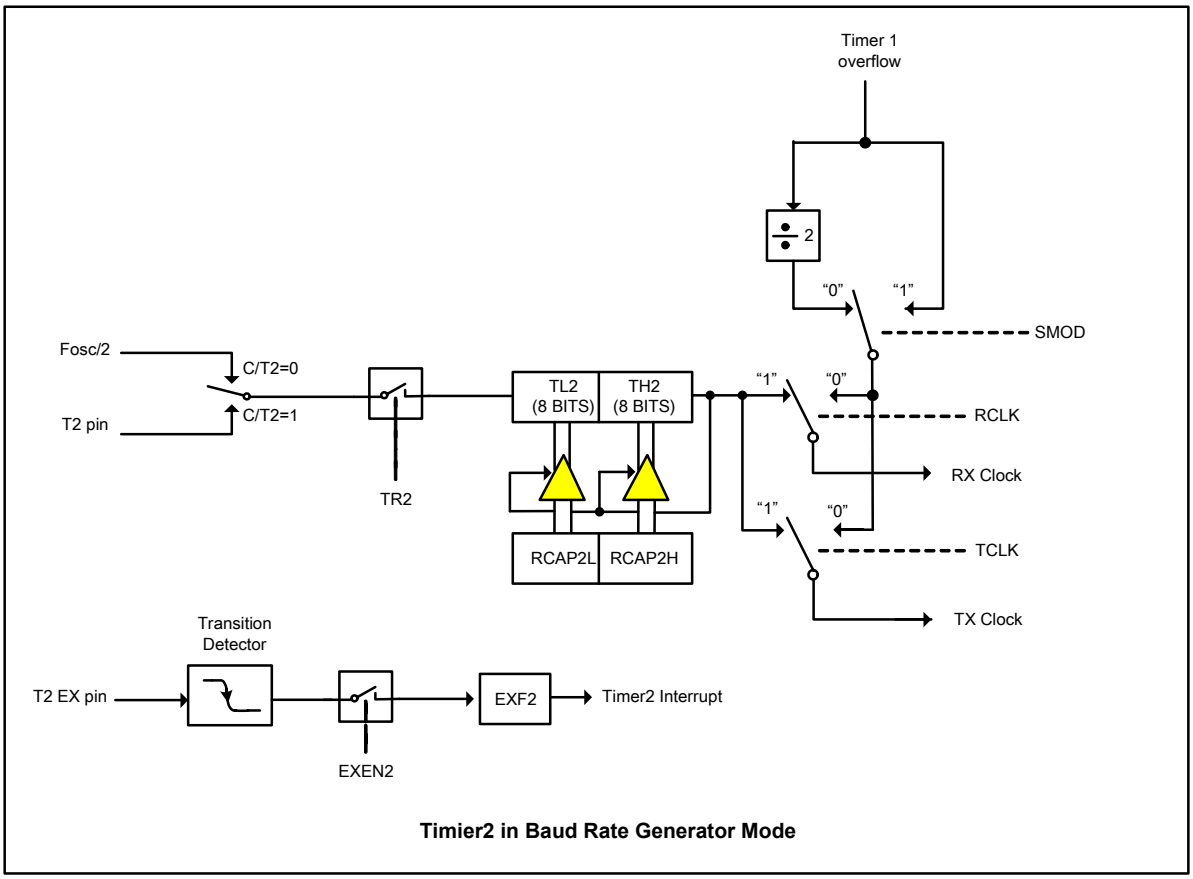
10.2.3. Baud-Rate Generator Mode (BRG)

Timer2 can be configured to generate various baud-rate. TCLK and/or RCLK in T2CON allow the serial port transmit and receive baud rates to be derived from either Timer1 or Timer2. When TCLK=0, Timer1 or S2BRT is used as the serial port transmit baud rate generator. When TCLK=1, Timer2 is used as the serial port transmit baud rate generator. RCLK has the same effect for the serial port baud rate. With these two bits, the serial port can have different receive and transmit baud rates – one generated from Timer1 or S2BRT and the other from Timer2.

In BRG mode, Timers is operated very like auto-reload counting-up mode except that the T2EX pin can not control reload. An overflow on Timer2 will load RCAP2H,RCAP2L content onto Timer2 but TF2 will not be set. A 1-to-0 transition on P2EX pin can set EXF2 to request interrupt service if EXEN2=1.

The baud rate in UART Mode1 and Mode3 are determined by Timer2's overflow rate given below :

Baud Rate = Oscillator Frequency / (2 x (65536 – [RCAP2H,RCAP2L]))



Timier2 in Baud Rate Generator Mode

10.2.4. Programmable Clock Output from Timer 2

Timer 2 has a Clock-Out Mode (while CP/-RL2=0 & T2OE=1). In this mode, Timer 2 operates as a programmable clock generator with 50% duty-cycle. The generated clocks come out on P1.0. The input clock, Fosc/2, increments the 16-bit timer [TH2, TL2], where Fosc is the system clock. The timer repeatedly counts to overflow from a loaded value. Once overflows occur, the contents of [RCAP2H, RCAP2L] are loaded into [TH2, TL2] for the consecutive counting. *Note that the Timer 2 overflow flag, TF2, will always not be set in this mode.* The following formula gives the clock-out frequency:

$$\textbf{Clock - out Frequency} = \frac{\textbf{Fosc}}{\textbf{4 x (65536 - [RCPA2H,RCAP2L])}}$$

(For a 12 MHz system clock, Timer 2 has a programmable output frequency range of 45.7 Hz to 3 MHz.)

How to Program Timer 2 as Its Clock-out Mode

• Set T2OE bit in T2MOD register.

• Clear C/T2 bit in T2CON register.

• Determine the 16-bit reload value from the formula and enter it in the [RCAP2H, RCAP2L] registers.

• Enter the same reload value as the initial value in the [TH2, TL2] registers.

• Set TR2 bit in T2CON register to start the Timer 2.

In the Clock-Out mode, Timer 2 rollovers will not generate an interrupt. This is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud rate generator and a clock generator simultaneously. Note, however, in this configuration, the baud rates and clock frequencies are not independent since both functions use the same reload values in the [RCAP2H, RCAP2L] registers.

11. Enhanced UART

11.1. Frame Error Detection

While the SMOD0 bit (in PCON, bit 6) is set, the hardware will set the FE bit (SCON.7) when an invalid stop bit is detected. The FE bit is not cleared by valid frames but should be cleared by software.

SCON (Address=98H, Serial Port Control Register)

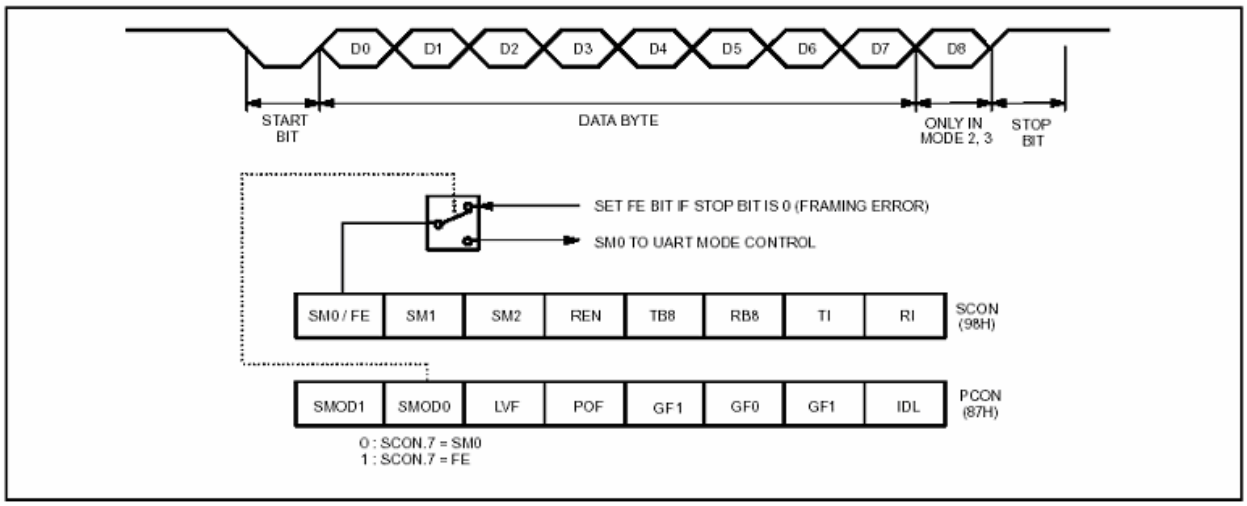
7	6	5	4	3	2	1	0
SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI

SM0/FE:
SM0: Serial Port Mode bit0 (when SMOD0=0).
FE: Frame Error bit (when SMOD0=1).

PCON (Address=87H, Power Control Register)

7	6	5	4	3	2	1	0
SMOD	SMOD0	LVF1	POF	GF1	GF0	PD	IDL

SMOD0: Clear to let SCON.7 function as 'SM0', and set to let SCON.7 function as 'FE'.



11.2. Automatic Address Recognition

Automatic Address Recognition is a feature which allows the UART to recognize certain addresses in the serial bit stream by using hardware to make the comparisons. This feature saves a great deal of software overhead by eliminating the need for the software to examine every serial address which passes by the serial port. This feature is enabled by setting the SM2 bit in SCON. In the 9 bit UART modes, mode 2 and mode 3, the Receive Interrupt flag (RI) will be automatically set when the received byte contains either the "Given" address or the "Broadcast" address. The 9-bit mode requires that the 9th information bit is a 1 to indicate that the received information is an address and not data. Automatic address recognition is shown in the following figure.

The 8 bit mode is called Mode 1. In this mode the RI flag will be set if SM2 is enabled and the information received has a valid stop bit following the 8 address bits and the information is either a Given or Broadcast address.

Mode 0 is the Shift Register mode and SM2 is ignored.

Using the Automatic Address Recognition feature allows a master to selectively communicate with one or more slaves by invoking the Given slave address or addresses. All of the slaves may be contacted by using the

Broadcast address. Two special Function Registers are used to define the slave's address, SADDR, and the address mask, SADEN.

SADEN is used to define which bits in the SADDR are to be used and which bits are "don't care". The SADEN mask can be logically ANDed with the SADDR to create the "Given" address which the master will use for addressing each of the slaves. Use of the Given address allows multiple slaves to be recognized while excluding others.

The following examples will help to show the versatility of this scheme:

Slave 0

SADDR = 1100 0000
SADEN = 1111 1101
Given = 1100 00X0

Slave 1

SADDR = 1100 0000
SADEN = 1111 1110
Given = 1100 000X

In the above example SADDR is the same and the SADEN data is used to differentiate between the two slaves. Slave 0 requires a 0 in bit 0 and it ignores bit 1. Slave 1 requires a 0 in bit 1 and bit 0 is ignored. A unique address for Slave 0 would be 1100 0010 since slave 1 requires a 0 in bit 1. A unique address for slave 1 would be 1100 0001 since a 1 in bit 0 will exclude slave 0. Both slaves can be selected at the same time by an address which has bit 0 = 0 (for slave 0) and bit 1 = 0 (for slave 1). Thus, both could be addressed with 1100 0000.

In a more complex system the following could be used to select slaves 1 and 2 while excluding slave 0:

Slave 0

SADDR = 1100 0000
SADEN = 1111 1001
Given = 1100 0XX0

Slave 1

SADDR = 1110 0000
SADEN = 1111 1010
Given = 1110 0X0X

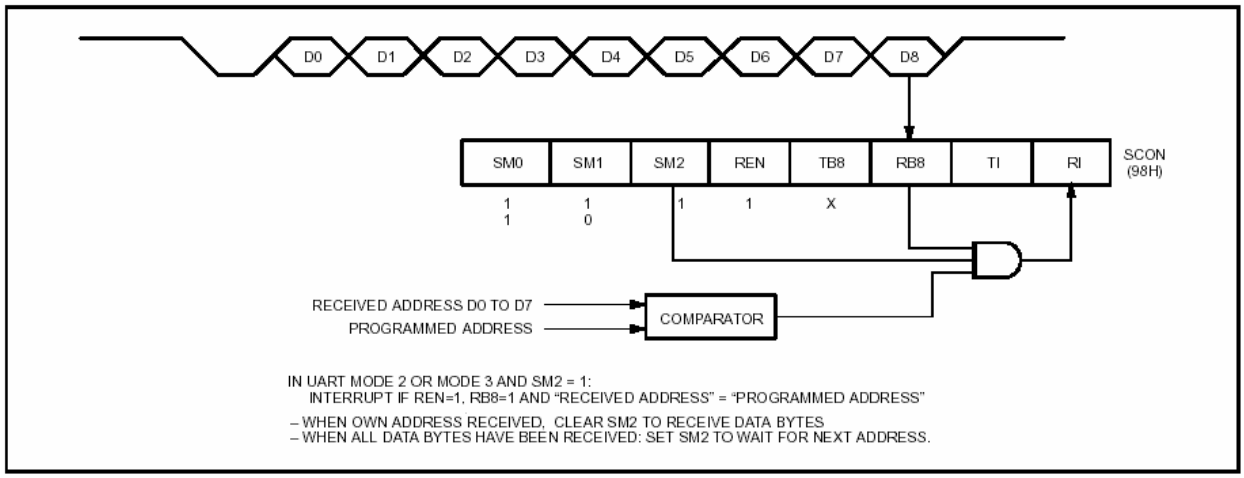
Slave 2

SADDR = 1110 0000
SADEN = 1111 1100
Given = 1110 00XX

In the above example the differentiation among the 3 slaves is in the lower 3 address bits. Slave 0 requires that bit 0 = 0 and it can be uniquely addressed by 1110 0110. Slave 1 requires that bit 1 = 0 and it can be uniquely addressed by 1110 0101. Slave 2 requires that bit 2 = 0 and its unique address is 1110 0011. To select Slaves 0 and 1 and exclude Slave 2 use address 1110 0100, since it is necessary to make bit 2 = 1 to exclude slave 2.

The Broadcast Address for each slave is created by taking the logical OR of SADDR and SADEN. Zeros in this result are trended as don't-cares. In most cases, interpreting the don't-cares as ones, the broadcast address will be FF hexadecimal.

Upon reset SADDR (SFR address 0A9H) and SADEN (SFR address 0B9H) are leaded with 0s. This produces a given address of all "don't cares" as well as a Broadcast address of all "don't cares". This effectively disables the Automatic Addressing mode and allows the micro-controller to use standard 80C51 type UART drivers which do not make use of this feature.



11.3. Baud Rate Setting

All the four operation modes of the serial port are the same as those of the standard 8051 except the baud rate setting. Three registers, PCON, AUXR and AUXR2, are related to the baud rate setting:

PCON (Address=87H, Power Control Register)

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL

SMOD: Double baud rate control bit.

AUXR (Address=8EH, Auxiliary Register)

7	6	5	4	3	2	1	0
-	-	BRADJ	-	T2X12	-	-	DPS

BRADJ: Baud rate adjustment index. Set to upgrade the baud rate of the Serial Port. The selection is as following table:

BRADJ0	Enhanced Baud Rate Setting
0	Default Baud Rate
1	Double Baud Rate

AUXR2 (Address=A6H, Auxiliary Register 2)

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	-	-	-	-	T0CKOE

T1X12: Timer 1 clock source select. Set to select Fosc as the clock source, and clear to select Fosc/12.

URM0X6: Serial Port mode 0 baud rate select. Set to select Fosc/2.

Bits T1X12 and URM0X6 provide a new option for the baud rate setting, as listed below.

Baud Rate in Mode 0

URM0X6 = 0	URM0X6 = 1
$\textbf{B.R.} = \frac{\textbf{Fosc}}{\textbf{12}}$ (The same as standard 8051)	$\textbf{B.R.} = \frac{\textbf{Fosc}}{\textbf{2}}$

Baud Rate in Mode 2

BRADJ = 0	BRADJ = 1
$B.R. = \frac{2^{SMOD}}{64} \times Fosc$ (The same as standard 8051)	$B.R. = \frac{2^{SMOD}}{32} \times Fosc$

Baud Rate in Mode 1 & 3

(1) Using Timer 1 as the Baud Rate Generator

	BRADJ = 0	BRADJ = 1
T1X12 = 0	$B.R. = \frac{2^{SMOD}}{32} \times \frac{Fosc}{12 \times (256-TH1)}$ (The same as standard 8051)	$B.R. = \frac{2^{SMOD}}{16} \times \frac{Fosc}{12 \times (256-TH1)}$
T1X12 = 1	$B.R. = \frac{2^{SMOD}}{32} \times \frac{Fosc}{256-TH1}$	$B.R. = \frac{2^{SMOD}}{16} \times \frac{Fosc}{256-TH1}$ ***Note1***

Note1:
For Fosc=12MHz, if {BRADJ1,BRADJ0}=01, T1X12=1, SMOD=1 and TH1=243, then we can get:
Baud Rate = (2/16) x 12MHz / (256-243) = 115385 bps.
Where, the deviation to the standard 115200 bps is +0.16%, which is acceptable in an UART communication.

(2) Using Timer 2 as the Baud Rate Generator

When Timer 2 is used as the baud rate generator (either TCLK or RCLK in T2CON is '1'), the baud rate is as follows.

BRADJ = 0	BRADJ = 1
$\textbf{B.R.} = \frac{\textbf{Fosc}}{\textbf{32}} \times \frac{\textbf{1}}{\textbf{65536-[RCAP2H,RCAP2L]}}$ (The same as standard 8051)	$\textbf{B.R.} = \frac{\textbf{Fosc}}{\textbf{8}} \times \frac{\textbf{1}}{\textbf{65536-[RCAP2H,RCAP2L]}}$ ***Note2***

Note2:
For Fosc=12MHz, if BRADJ=1 and [RCAP2H,L]=65523, then we can get:
Baud Rate = (12MHz/8) x 1/(65536-65523) = 115385 bps.
Where, the deviation to the standard 115200 bps is +0.16%, which is acceptable in UART communication.

12. Interrupt

The device has a total of 12 interrupt sources. Each of the interrupt sources can be individually enabled or disabled by setting or clearing a bit in the Interrupt Enable registers (IE, AUXIE and XICON). And, each interrupt source can also be individually programmed to one of two priority levels by setting or clearing bit in the Interrupt Priority registers (IP & AUXIP). The two priority level interrupt structure allows great flexibility in controlling the handling of these interrupt sources. The following table lists all the interrupt sources.

Table: Interrupt Sources

Source No.	Interrupt Name	Interrupt Enable Bit	Interrupt Flag Bit	Interrupt Priority Bits	Polling Priority	Vector Address
#1	External Interrupt, INT0	EX0	IE0	PX0	(Highest)	0003H
#2	Timer 0	ET0	TF0	PT0	.	000BH
#3	External Interrupt, INT1	EX1	IE1	PX1	.	0013H
#4	Timer 1	ET1	TF1	PT1	.	001BH
#5	Serial Port	ES	RI+TI	PS	.	0023H
#6	Timer 2	ET2	TF2+ EXF2	PT2	.	002BH
#7	External Interrupt, INT2*	EX2	IE2	PX2	.	0033H
#8	External Interrupt, INT3*	EX3	IE3	PX3	.	003BH
#9	SPI	ESPI	SPIF	PSPI	.	0043H
#10	-	-	-	-	.	004BH
#11	-	-	-	-	.	0053H
#12	-	-	-	-	.	005BH
#13	-	-	-	-	.	0063H
#14	Keypad Interrupt	EKBI	KBIF	PKBI	.	006BH
#15	Two Wire Serial Interface	ETWSI	SI	PTWSI	.	0073H
#16	USB	EUSB	(See Note1)	PUSB	(Lowest)	007BH

Note1: The USB interrupt flags include:

(1) URST, URSM and USUS: contained in USB register UPCON.
(2) UTXD0, URXD0, UTXD1, UTXD2, ASOFIF and SOFIF: contained in USB register UIFLG.
(3) UTXD3, URXD3, URXD4 and UTXD5: contained in USB register UIFLG1

IE (Address=A8H, Interrupt Enable Register)

7	6	5	4	3	2	1	0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA: Global disable bit. If EA = 0, all interrupts are disabled. If EA = 1, each interrupt can be individually enabled or disabled by setting or clearing its enable bit.

ET2: Timer 2 interrupt enable bit.

ES: Serial Port interrupt enable bit.

ET1: Timer 1 interrupt enable bit.

EX1: External interrupt 1 enable bit.

ET0: Timer 0 interrupt enable bit.

EX0: External interrupt 0 enable bit.

AUXIE (Address=ADH, Interrupt Enable Register 2)

7	6	5	4	3	2	1	0
EUSB	ETWSI	EKBI	-	-	-	-	ESPI

EUSB: USB interrupt enable bit.

ETWSI: 2-wire-Serial-Interface interrupt enable bit.

EKBI: Keypad interrupt enable bit.

ESPI: SPI interrupt enable bit.

XICON (Address=C0H, External Interrupt Control Register)

7	6	5	4	3	2	1	0
IL3	EX3	IE3	IT3	IL2	EX2	IE2	IT2

IL3: External interrupt 3 level control bit. 1: rising-edge/high-level activated; 0: falling-edge/low-level activated.

EX3: External interrupt 3 enable bit.

IE3: External interrupt 3 interrupt flag.

IT3: External interrupt 3 type control bit. 1: edge-triggered; 0: level-triggered.

IL2: External interrupt 2 level control bit. 1: rising-edge/high-level activated; 0: falling-edge/low-level activated.

EX2: External interrupt 2 enable bit.

IE2: External interrupt 2 interrupt flag.

IT2: External interrupt 2 type control bit. 1: edge-triggered; 0: level-triggered.

IP (Address=B8H, Interrupt Priority Register)

7	6	5	4	3	2	1	0
PX3	PX2	PT2	PS	PT1	PX1	PT0	PX0

PX3: External interrupt 3 priority bit.

PX2: External interrupt 2 priority bit.

PT2: Timer 2 interrupt priority bit.

PS: Serial Port interrupt priority bit.

PT1: Timer 1 interrupt priority bit.

PX1: External interrupt 1 priority bit.

PT0: Timer 0 interrupt priority bit.

PX0: External interrupt 0 priority bit.

AUXIP (Address=AEH, Auxiliary Interrupt Priority Register)

7	6	5	4	3	2	1	0
PUSB	PTWSI	PKBI	-	-	-	-	PSPI

PUSB: USB interrupt priority bit.

PTWSI: 2-wire-Serial-Interface interrupt priority bit.

PKBI: Keypad interrupt priority bit.

PSPI: SPI interrupt priority bit.

12.1. Two Priority Levels

The bit values in the register IP and AUXIP determine what priority level each interrupt has. The following tables show the bit values and priority levels associated with each combination.

Table: Priority Level Determined by [IP]

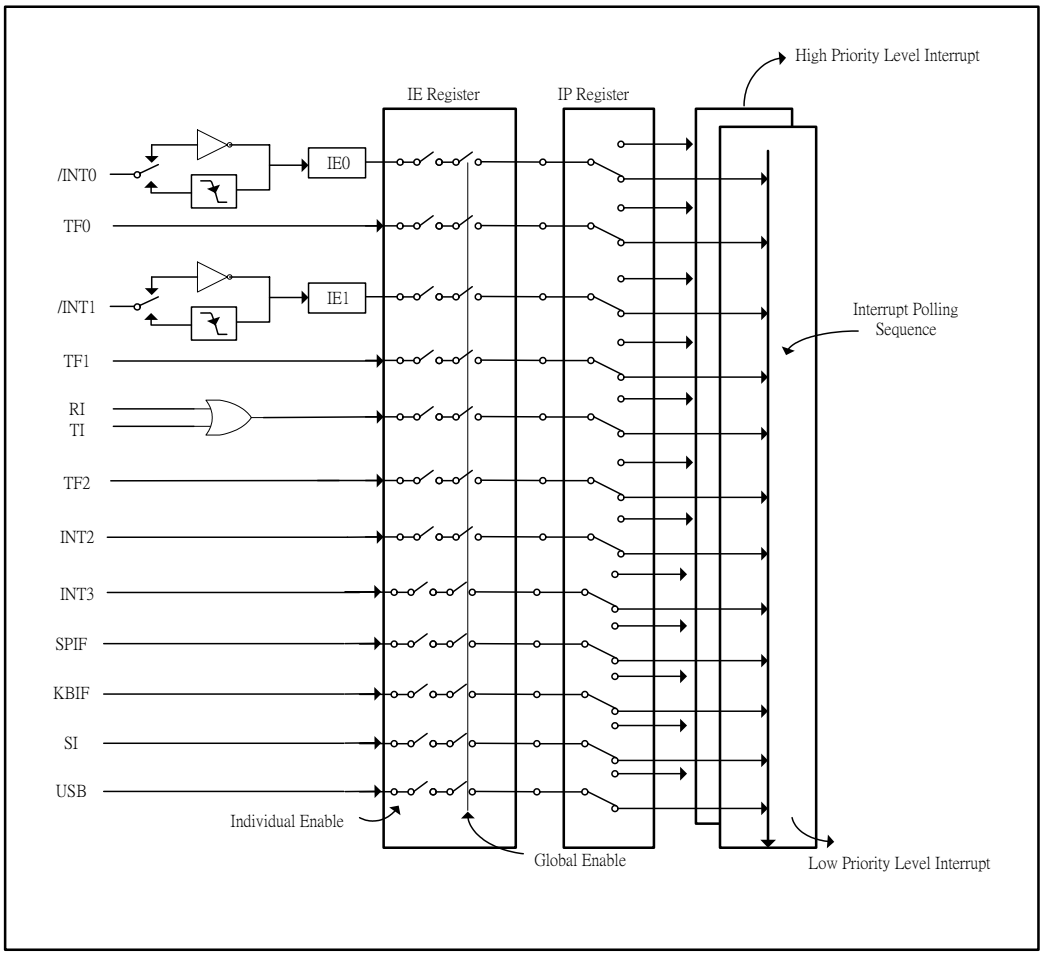
IP.x	Interrupt Priority Level
1	Level 1 (high priority)
0	Level 0 (low priority)

Table: Priority Level Determined by [AUXIP]

AUXIP.x	Interrupt Priority Level
1	Level 1 (high priority)
0	Level 0 (low priority)

For example, if (IP.3)=(1), then Timer 1 has the priority level equal to 1, which is higher than level 0 with (IP.3)=(0).

12.2. Interrupt System



12.3. Note on Interrupt during ISP/IAP

During ISP/IAP, the CPU halts for a while for internal ISP/IAP processing. At this time, the interrupt will queue up for being serviced if the interrupt is enabled previously. Once the ISP/IAP is complete, the CPU continues running and the interrupts in the queue will be serviced immediately if the interrupt flag is still active. Users, however, should be aware of the following:

(1) Any interrupt can not be serviced in time during the CPU halts for ISP/IAP processing.

*(2) The **low-level triggered** external interrupts, /INT0, /INT1, INT2 and INT3, should keep active until the ISP/IAP is complete, or they will be neglected.*

13. Additional External Interrupts (INT2 and INT3)

The device has two additional external interrupt inputs: INT2 (P3.6) and INT3 (P3.7). They are identical to /INT0 (P3.2) or /INT1 (P3.3) except the edge-triggered type (ITx=1) can be programmed to be rising-edge or falling-edge activated, and level-triggered type (ITx=0) can be programmed to be high-level or low-level activated. The following special function registers are related to their operation.

XICON (Address=C0H, External Interrupt Control Register)

7	6	5	4	3	2	1	0
IL3	EX3	IE3	IT3	IL2	EX2	IE2	IT2

IL3: External interrupt 3 level control bit. 1: rising-edge/high-level activated; 0: falling-edge/low-level activated.

EX3: External interrupt 3 enable bit.

IE3: External interrupt 3 interrupt flag.

IT3: External interrupt 3 type control bit. 1: edge-triggered; 0: level-triggered.

IL2: External interrupt 2 level control bit. 1: rising-edge/high-level activated; 0: falling-edge/low-level activated.

EX2: External interrupt 2 enable bit.

IE2: External interrupt 2 interrupt flag.

IT2: External interrupt 2 type control bit. 1: edge-triggered; 0: level-triggered.

IP (Address=B8H, Interrupt Priority Register)

7	6	5	4	3	2	1	0
PX3	PX2	PT2	PS	PT1	PX1	PT0	PX0

PX3: External interrupt 3 priority bit.

PX2: External interrupt 2 priority bit.

PT2: Timer 2 interrupt priority bit.

PS: Serial Port interrupt priority bit.

PT1: Timer 1 interrupt priority bit.

PX1: External interrupt 1 priority bit.

PT0: Timer 0 interrupt priority bit.

PX0: External interrupt 0 priority bit.

14. Keypad Interrupt

The Keypad Interrupt function is intended primarily to allow a single interrupt to be generated when Port 1 is equal to or not equal to a certain pattern. This function can be used for bus address recognition or keypad recognition. The user can configure the port via SFRs for different tasks.

There are three SFRs related to this function. The Keypad Interrupt Mask Register (KBMASK) is used to define which input pins connected to Port 0 are enabled to trigger the interrupt. The Keypad Pattern Register (KBPATN) is used to define a pattern that is compared to the value of Port 0. The Keypad Interrupt Flag (KBIF) in the Keypad Interrupt Control Register (KBCON) is set by hardware when the condition is matched. An interrupt will be generated if it has been enabled by setting the EKBI bit in AUXIE register and EA=1. The PATN_SEL bit (in KBCON) is used to define “equal” or “not-equal” for the comparison.

In order to use the Keypad Interrupt as the “Keyboard” Interrupt, the user needs to set KBPATN=0xFF and PATN_SEL=0 (not equal). Then, any key connected to Port 0 which is enabled by KBMASK register will cause the hardware to set KBIF and generate an interrupt if it has been enabled. The interrupt may wake up the CPU from Idle or Power down modes.

The following special function registers are related to the KBI operation:

KBPATN (Address=D5H, Keypad Pattern Register)

7	6	5	4	3	2	1	0
KBPATN.7	KBPATN.6	KBPATN.5	KBPATN.4	KBPATN.3	KBPATN.2	KBPATN.1	KBPATN.0

KBPATN.7-0: The keypad pattern, reset value is 0xFF.

KBCON (Address=D6H, Keypad Control Register)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	PATN_SEL	KBIF

PATN_SEL: Pattern Matching Polarity selection.
When set, Port 0 has to be ***equal*** to the user-defined Pattern in KBPATN to generate the interrupt.
When clear, Port 0 has to be ***not equal*** to the value of KBPATN register to generate the interrupt.

KBIF: Keypad Interrupt Flag. Set when Port 0 matches user defined conditions specified in KBPATN, KBMASK, and PATN_SEL. Needs to be cleared by software by writing “0”.

KBMASK (Address=D7H, Keypad Interrupt Mask Register)

7	6	5	4	3	2	1	0
KBMASK.7	KBMASK.6	KBMASK.5	KBMASK.4	KBMASK.3	KBMASK.2	KBMASK.1	KBMASK.0

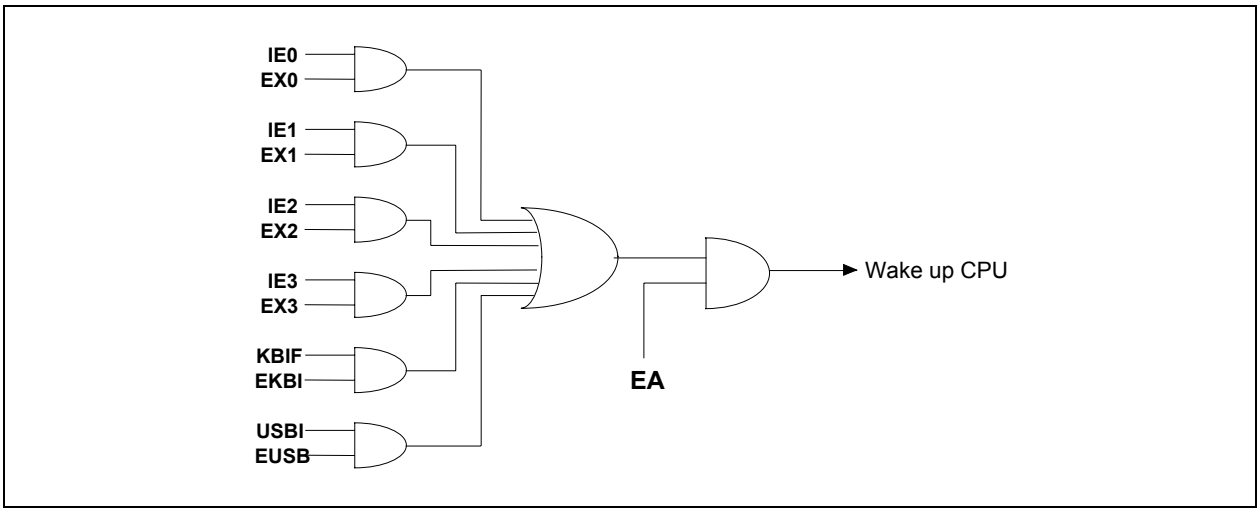
KBMASK.7: When set, enables P0.7 as a cause of a Keypad Interrupt.
KBMASK.6: When set, enables P0.6 as a cause of a Keypad Interrupt.
KBMASK.5: When set, enables P0.5 as a cause of a Keypad Interrupt.
KBMASK.4: When set, enables P0.4 as a cause of a Keypad Interrupt.
KBMASK.3: When set, enables P0.3 as a cause of a Keypad Interrupt.
KBMASK.2: When set, enables P0.2 as a cause of a Keypad Interrupt.
KBMASK.1: When set, enables P0.1 as a cause of a Keypad Interrupt.
KBMASK.0: When set, enables P0.0 as a cause of a Keypad Interrupt.

15. Wake-up from Power-down Mode

When the CPU is put into power-down mode, the external interrupts (/INT0, /INT1, INT2 and INT3), keypad interrupt and USB interrupt will wake up the CPU if any of them is enabled.

15.1. Power-down Wake-up Sources

The following figure shows the power-down wake-up sources.



Once the CPU is wakened up, the interrupt service routine is serviced until the “RETI” instruction is encountered, and, the next instruction to be executed will be the one following the instruction that put the CPU into power-down mode.

15.2. Sample Code for Wake-up from Power-down

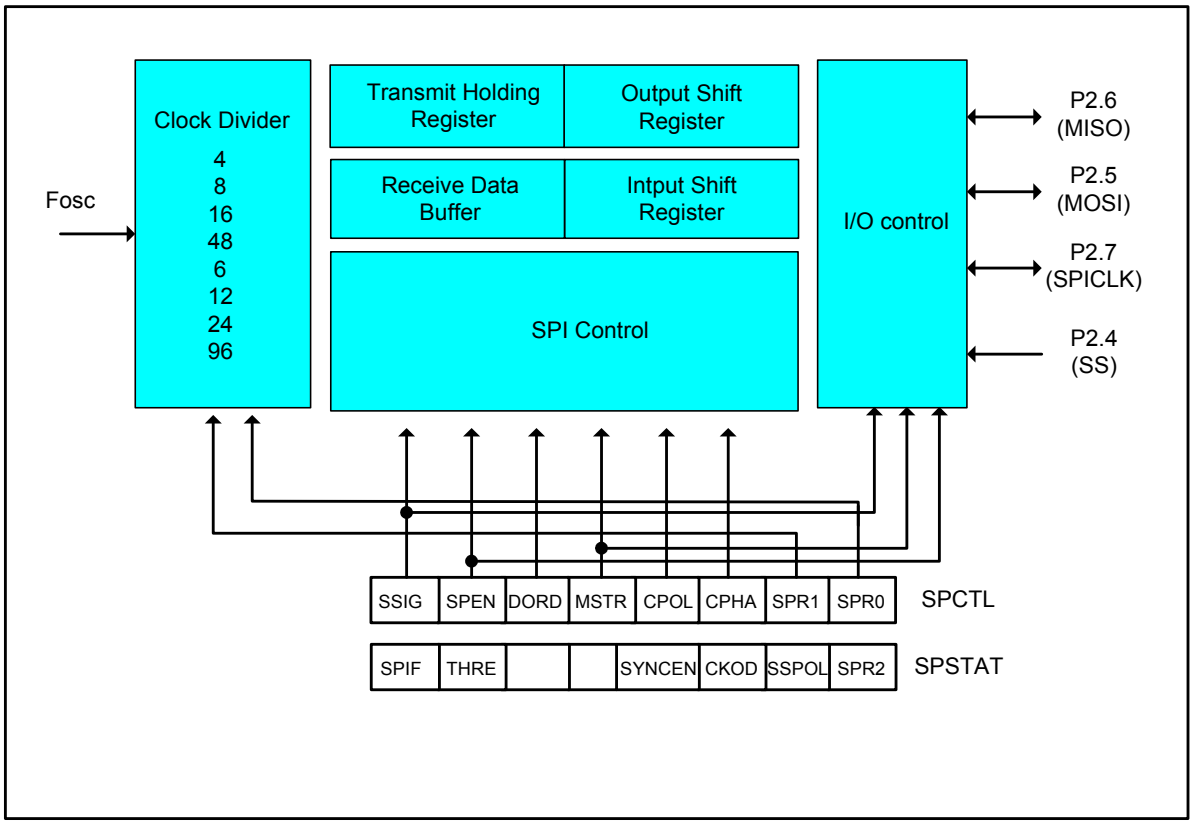
Note: /INT0 is used in this example.

```
;*******************************************************************************************
; Wake-up-from-power-down by /INT0 interrupt
;*******************************************************************************************
INT0      BIT     0B2H        ;P3.2
EA        BIT     0AFH        ;IE.7
EX0       BIT     0A8H        ;IE.0

          CSEG    AT  0000h
          JMP     start
;
          CSEG    AT  0003h   ;/INT0 interrupt vector, address=0003h
          JMP     IE0_isr
IE0_isr:
          CLR     EX0
          ;... do something
          ;...
          RETI
;
start:
          ;...
          ;...

          SETB    INT0        ;pull high P3.2

          CLR     IE0         ;clear /INT0 interrupt flag
          SETB    IT0         ;may select falling-edge/low-level triggered
          SETB    EA          ;enable global interrupt
          SETB    EX0         ;enable /INT0 interrupt

          ORL     PCON,#02h   ;put MCU into power-down mode
          NOP                 ;! Note: here must be a NOP

Resume_operation:
          ;If /INT0 is triggered by a falling-edge, the MCU will wake up, enter "IE0_isr",
          ;and then return here to run continuously !


          ;...
          ;...
;
```

16. Serial Peripheral Interface (SPI)

The device provides a high-speed serial communication interface, the SPI interface. SPI is a full-duplex, high-speed and synchronous communication bus with two operation modes: Master mode and Slave mode. Up to 4 Mbps can be supported in either Master or Slave mode under the 12MHz system clock. A specially designed Transmit Holding Register (THR) improves the transmit performance compared to the conventional SPI.

SPI Block Diagram



SPI block diagram

The SPI interface has four pins: MISO (P2.6), MOSI (P2.5), SPICLK (P2.7) and /SS (P2.4):

• SPICLK, MOSI and MISO are typically tied together between two or more SPI devices. Data flows from master to slave on the MOSI pin (Master Out / Slave In) and flows from slave to master on the MISO pin (Master In / Slave Out). The SPICLK signal is output in the master mode and is input in the slave mode. If the SPI system is disabled, i.e., SPEN (SPCTL.6) = 0, these pins function as normal I/O pins.

• /SS is the optional slave select pin. In a typical configuration, an SPI master asserts one of its port pins to select one SPI device as the current slave. An SPI slave device uses its SS pin to determine whether it is selected. But if SPEN (SPCTL.6) = 0 or SSIG (SPCTL.7) = 1, the /SS pin is ignored.

Note that even if the SPI is configured as a master (MSTR = 1), it can still be converted to a slave by driving the /SS pin low (if SSIG = 0). Should this happen, the SPIF bit (SPSTAT.7) will be set. See Section "Mode change on /SS-pin".

The following special function registers are related to the SPI operation:

SPCTL (Address=85H, SPI Control Register)

7	6	5	4	3	2	1	0
SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

SSIG: /SS is ignored
If SSIG=1, MSTR decides whether the device is a master or slave.
If SSIG=0, the /SS pin decides whether the device is a master or slave.

SPEN: SPI enable
If SPEN=1, the SPI is enabled.
If SPEN=0, the SPI interface is disabled and all SPI pins will be general-purpose I/O ports.

DORD: SPI data order
1: The LSB of the data byte is transmitted first.
0: The MSB of the data byte is transmitted first.

MSTR: Master/Slave mode select

CPOL: SPI clock polarity select
1: SPICLK is high when idle. The leading edge of SPICLK is the falling edge and the trailing edge is the rising edge.
0: SPICLK is low when idle. The leading edge of SPICLK is the rising edge and the trailing edge is the falling edge.

CPHA: SPI clock phase select
1: Data is driven on the leading edge of SPICLK, and is sampled on the trailing edge.
0: Data is driven when /SS pin is low (SSIG=0) and changes on the trailing edge of SPICLK. Data is sampled on the leading edge of SPICLK.
(Note : If SSIG=1, CPHA must not be 1, otherwise the operation is not defined.)

SPR1-SPR0: SPI clock rate select (associated with SPR2, when in master mode)
{SPR2,SPR1,SPR0} = 000: Fosc/3 100: Fosc/16
001: Fosc/6 101: Fosc/24
010: Fosc/8 110: Fosc/48
011: Fosc/12 111: Fosc/96 (*Where, Fosc is the system clock.)*

SPSTAT (Address=84H, SPI Status Register)

7	6	5	4	3	2	1	0
SPIF	THRE	-	-	-	-	-	SPR2

SPIF: SPI transfer completion flag
When a serial transfer finishes, the SPIF bit is set and an interrupt is generated if SPI interrupt is enabled. If /SS pin is driven low when SPI is in master mode with SSIG=0, SPIF will also be set to signal the “mode change”. The SPIF is cleared in software by writing ‘1’ to this bit.

THRE *(read-only)*: Transmit Holding Register (THR) Empty flag.
0: means the THR is “empty”.
This bit is cleared by hardware when the THR is empty. That means the data in THR is loaded (by H/W) into the Output Shift Register to be transmitted, and now the user can write the next data byte to SPDAT for next transmission.
1: means the THR is “not empty”.
This bit is set by hardware just when SPDAT is written by software.

SPR2: SPI clock rate select (associated with SPR1 and SPR0)

SPDAT (Address=86H, SPI Data Register)

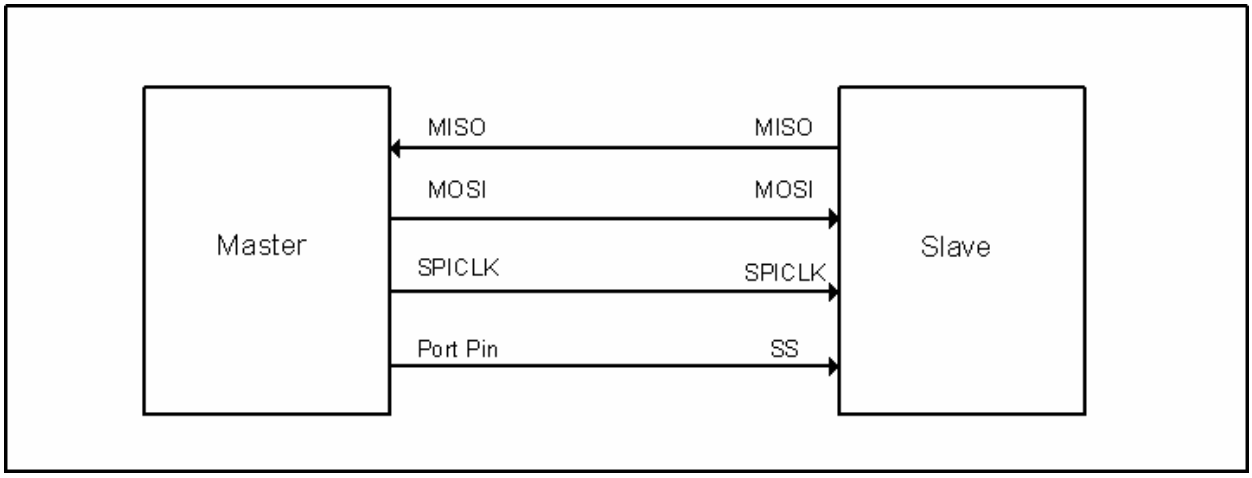
7	6	5	4	3	2	1	0
(MSB)							(LSB)

SPDAT has two physical registers for writing to and reading from:
(1) For writing to: SPDAT is the THR containing data to be loaded into the output shift register for transmit.
(2) For reading from: SPDAT is the input shift register containing the received data.

16.1. Typical SPI Configurations

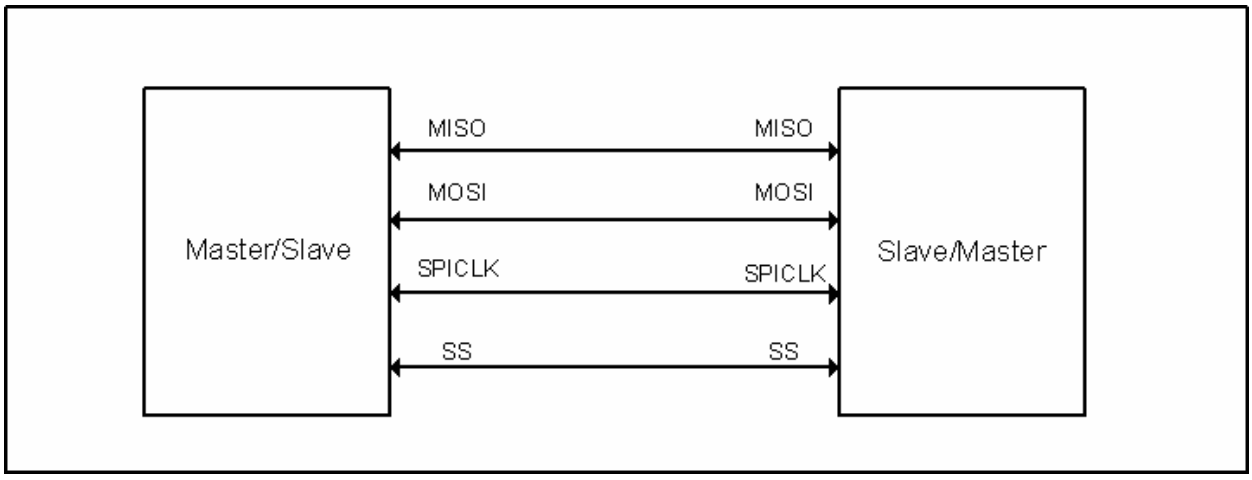
16.1.1. Single Master & Single Slave

For the master: can use any port pin, including P2.4 (/SS), to drive the /SS pin of the slave.
For the slave: SSIG is '0', and /SS pin is used to select the slave.



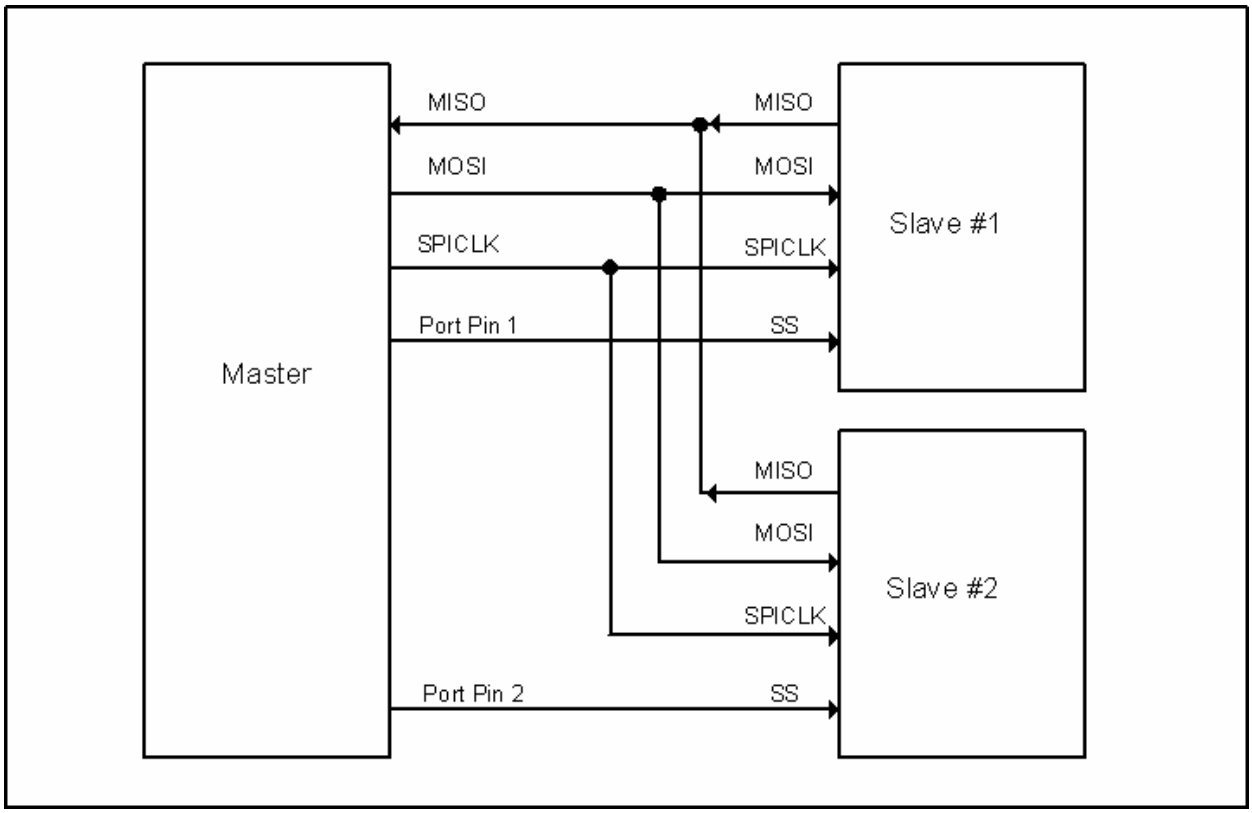
16.1.2. Dual Device, where either can be a Master or a Slave

Two devices are connected to each other and either device can be a master or a slave. When no SPI operation is occurring, both can be configured as masters with MSTR=1, SSIG=0 and P2.4 (/SS) configured in quasi-bidirectional mode. When any device initiates a transfer, it can configure P2.4 as an output and drive it low to force a "mode change to slave" in the other device.



16.1.3. Single Master & Multiple Slaves

For the master: can use any port pin, including P2.4 (/SS) to drive the /SS pins of the slaves.
For all the slaves: SSIG is '0', and are selected by their corresponding /SS pins.



16.2. Configuring the SPI

Table: SPI Master and Slave Selection

SPEN (SPCTL. 6)	SSIG (SPCTL. 7)	/SS -pin	MSTR (SPCTL. 4)	Mode	MISO -pin	MOSI -pin	SPICLK -pin	Remarks
0	X	X	X	SPI disabled	input	input	input	P2.4~P2.7 are used as general port pins.
1	0	0	0	Salve (selected)	output	input	input	Selected as slave.
1	0	1	0	Slave (not selected)	Hi-Z	input	input	Not selected.
1	0	0	1 ➔ 0	Slave (by mode change)	output	input	input	Mode change to slave if /SS pin is driven low, and MSTR will be cleared to '0' by H/W automatically.
1	0	1	1	Master (idle)	input	Hi-Z	Hi-Z	MOSI and SPICLK are at high impedance to avoid bus contention when the Master is idle.
				Master (active)		output	output	MOSI and SPICLK are push-pull when the Master is active.
1	1	X	0	Slave	output	input	input	
1	1	X	1	Master	input	output	output	

"X" means "don't care".

16.2.1. Additional Considerations for a Slave

When CPHA is 0, SSIG must be 0 and /SS pin must be negated and reasserted between each successive serial byte transfer. Note the SPDAT register cannot be written while /SS pin is active (low), and the operation is undefined if CPHA is 0 and SSIG is 1.

When CPHA is 1, SSIG may be 0 or 1. If SSIG=0, the /SS pin may remain active low between successive transfers (can be tied low at all times). This format is sometimes preferred for use in systems having a single fixed master and a single slave configuration.

16.2.2. Additional Considerations for a Master

In SPI, transfers are always initiated by the master. If the SPI is enabled (SPEN=1) and selected as master, writing to the SPI data register (SPDAT) by the master starts the SPI clock generator and data transfer. The data will start to appear on MOSI about one half SPI bit-time to one SPI bit-time after data is written to SPDAT.

Before starting the transfer, the master may select a slave by driving the /SS pin of the corresponding device low. Data written to the SPDAT register of the master is shifted out of MOSI pin of the master to the MOSI pin of the slave. And, at the same time the data in SPDAT register of the selected slave is shifted out on MISO pin to the MISO pin of the master.

After shifting one byte, the SPI clock generator stops, setting the transfer completion flag (SPIF) and an interrupt will be created if the SPI interrupt is enabled. The two shift registers in the master CPU and slave CPU can be considered as one distributed 16-bit circular shift register. When data is shifted from the master to the slave, data is also shifted in the opposite direction simultaneously. This means that during one shift cycle, data in the master and the slave are interchanged.

16.2.3. Mode Change on /SS-pin

If SPEN=1, SSIG=0, MSTR=1 and /SS pin=1, the SPI is enabled in master mode. In this case, another master can drive this pin low to select this device as an SPI slave and start sending data to it. To avoid bus contention, the SPI becomes a slave. As a result of the SPI becoming a slave, the MOSI and SPICLK pins are forced to be an input and MISO becomes an output. The SPIF flag in SPSTAT is set, and if the SPI interrupt is enabled, an SPI interrupt will occur. User software should always check the MSTR bit. If this bit is cleared by a slave select and the user wants to continue to use the SPI as a master, the user must set the MSTR bit again, otherwise it will stay in slave mode.

16.2.4. No Write Collision

The SPI is double buffered in the transmit direction and also double buffered in the receive direction. New data for transmission can not be written to the Transmit Holding Register (THR) until the previous data is loaded to the Output Shift Register to be transmitted. The THRE (SPSTAT.6) bit is set to indicate the user can write a new data byte to the THR for the following proceeding transmission. This architecture makes higher throughput compared to the one with Write Collision indication.

16.2.5. SPI Clock Rate Select

The SPI clock rate selection (in master mode) uses the SPR1 and SPR0 bits in the SPCTL register, as shown below.

Table: Serial Clock Rates

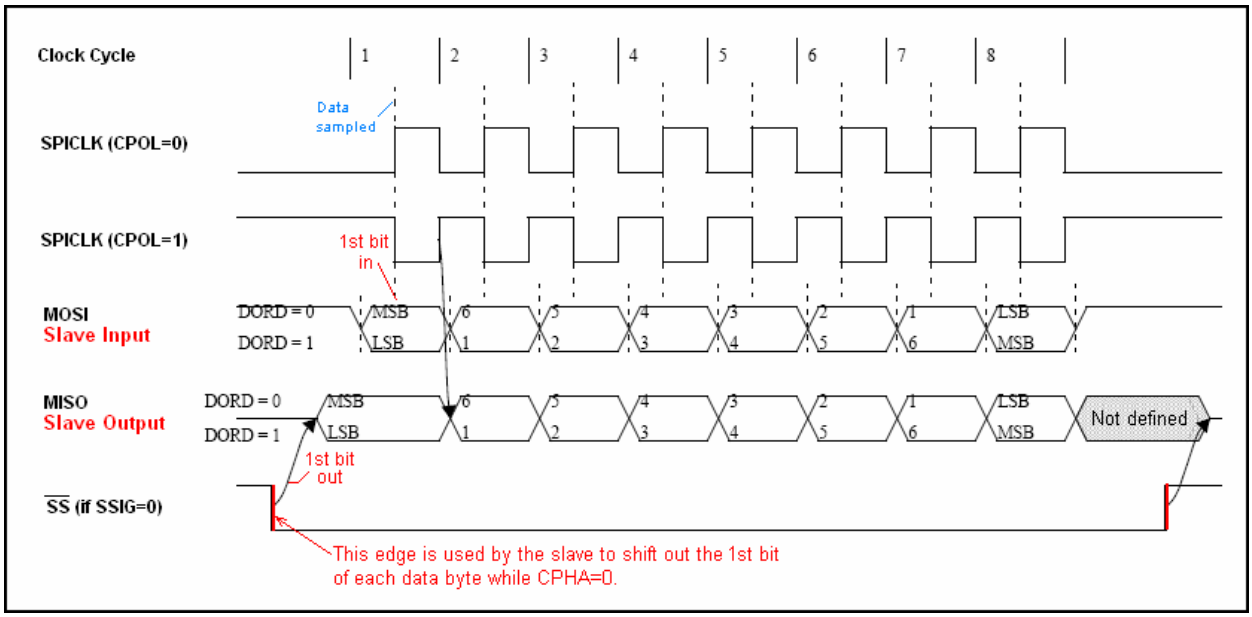
SPR2	SPR1	SPR0	SPI Clock Rate @ Fosc=12MHz	Fosc divided by
0	0	0	3 MHz	4
0	0	1	2 MHz	6
0	1	0	1.5 MHz	8
0	1	1	1 MHz	12
1	0	0	750 KHz	16
1	0	1	500 KHz	24
1	1	0	250 KHz	48
1	1	1	125 KHz	96

Where, Fosc is the system clock.

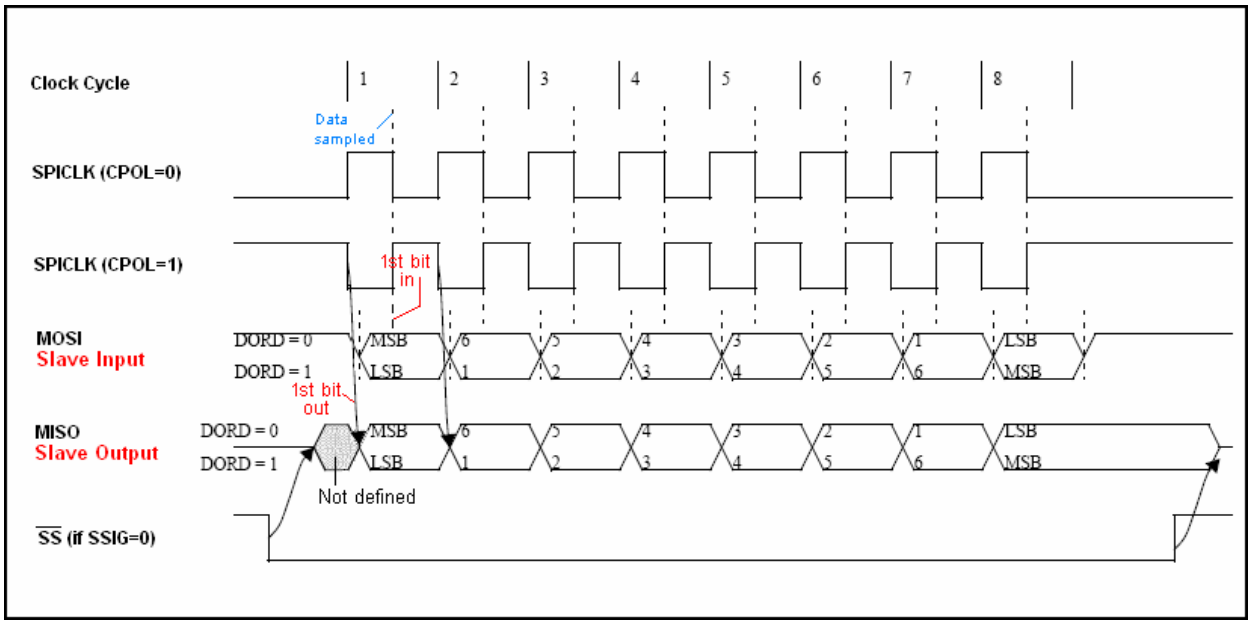
16.3. Data Mode

Clock Phase Bit (CPHA) allows the user to set the edges for sampling and changing data. The Clock Polarity bit, CPOL, allows the user to set the clock polarity. The following figures show the different settings of CPHA.

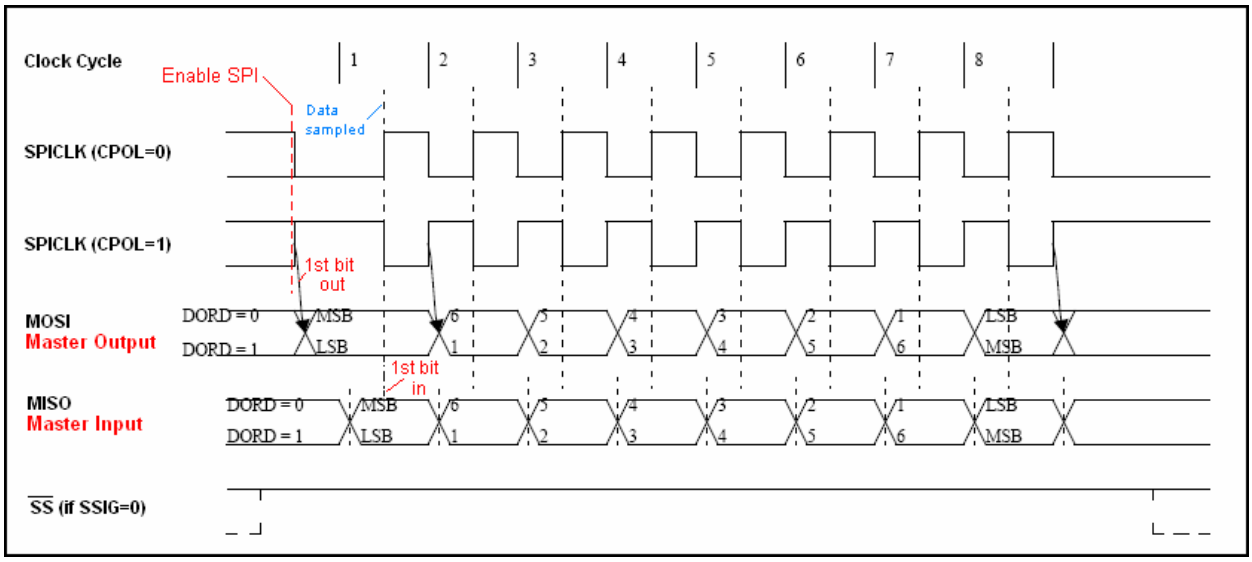
16.3.1. SPI Slave Transfer Format with CPHA=0



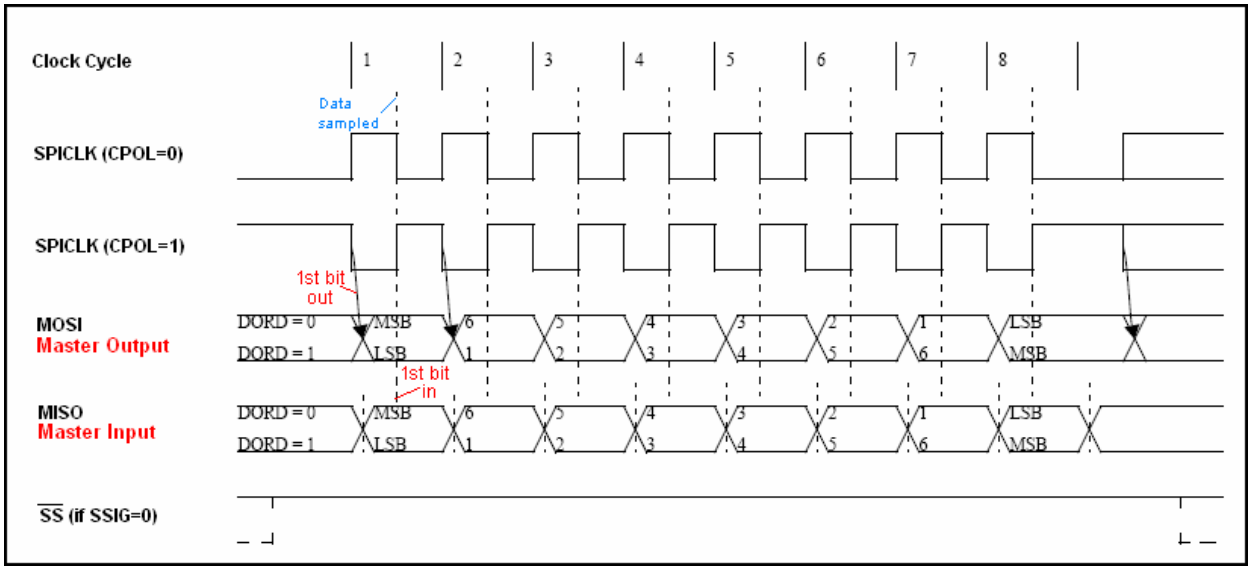
16.3.2. SPI Slave Transfer Format with CPHA=1



16.3.3. SPI Master Transfer Format with CPHA=0



16.3.4. SPI Master Transfer Format with CPHA=1



17. 2-wire Serial Interface (TWSI)

Features

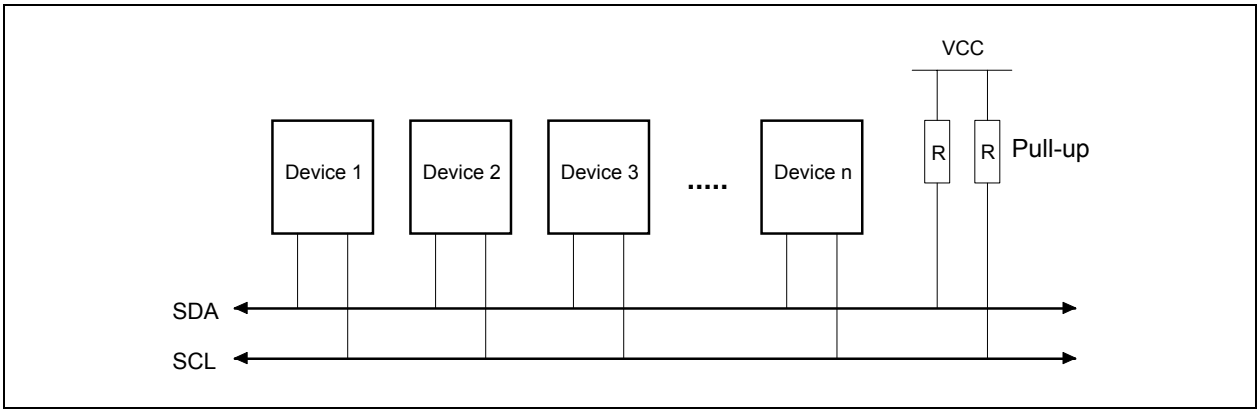
- Simple yet powerful and flexible communication interface, only two bus lines needed.
- Both Master and Slave operation supported, and device can operate as Transmitter or Receiver.
- 7-bit address space allows up to 128 different Slave addresses.
- Multi-master arbitration support.
- Up to 400 kHz data transfer speed.
- Programmable Slave address with General Call support.

Description

The 2-wire Serial Interface (TWSI) is ideally suited for typical microcontroller applications. The TWSI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement this bus is a single pull-up resistor for each of the TWSI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWSI protocol.

The CPU interfaces to the TWSI through the following four special function registers: SIADR (serial interface address register), SIDAT (serial interface data register), SICON (serial interface control register) and SISTA (serial interface status register). And, the TWSI hardware interfaces to the serial bus via two lines: SDA (serial data line, P2.1) and SCL (serial clock line, P2.0).

TWSI Bus Interconnection



17.1. The Special Function Registers for TWSI

The Serial Interface Address Register, SIADR, Address=D1H

The CPU can read from and write to this register directly. SIADR is not affected by the TWSI hardware. The contents of this register are irrelevant when TWSI is in a master mode. In the slave mode, the seven most significant bits must be loaded with the microcontroller's own slave address, and, if the least significant bit (GC) is set, the general call address (00H) is recognized; otherwise it is ignored. The most significant bit corresponds to the first bit received from the TWSI bus after a START condition.

SIADR (Address=D1H, TWSI Address Register)

7	6	5	4	3	2	1	0
(A6)	(A5)	(A4)	(A3)	(A2)	(A1)	(A0)	GC

|<---------- Own Slave Address ----------->|

The Serial Interface Data Register, SIDAT, Address=D2H

This register contains a byte of serial data to be transmitted or a byte which has just been received. The CPU can read from or write to this register directly while it is not in the process of shifting a byte. This occurs when TWSI is in a defined state and the serial interrupt flag (SI) is set. Data in SIDAT remains stable as long as SI is set. While data is being shifted out, data on the bus is simultaneously being shifted in; SIDAT always contains the last data byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in SIDAT.

SIDAT (Address=D2H, TWSI Data Register)

7	6	5	4	3	2	1	0
(D7)	(D6)	(D5)	(D4)	(D3)	(D2)	(D1)	(D0)

<---------- Shift direction ---------->

SIDAT and the ACK flag form a 9-bit shift register which shifts in or shifts out an 8-bit byte, followed by an acknowledge bit. The ACK flag is controlled by the TWSI hardware and cannot be accessed by the CPU. Serial data is shifted through the ACK flag into SIDAT on the rising edges of serial clock pulses on the SCL line. When a byte has been shifted into SIDAT, the serial data is available in SIDAT, and the acknowledge bit is returned by the control logic during the 9th clock pulse. Serial data is shifted out from SIDAT on the falling edges of clock pulses on the SCL line.

When the CPU writes to SIDAT, the bit SD7 is the first bit to be transmitted to the SDA line. After nine serial clock pulses, the eight bits in SIDAT will have been transmitted to the SDA line, and the acknowledge bit will be present in the ACK flag. Note that the eight transmitted bits are shifted back into SIDAT.

The Serial Interface Control Register, SICON, Address=F8H

The CPU can read from and write to this register directly. Two bits are affected by the TWSI hardware: the SI bit is set when a serial interrupt is requested, and the STO bit is cleared when a STOP condition is present on the bus. The STO bit is also cleared when ENS1="0".

SICON (Address=F8H, TWSI Control Register)

7	6	5	4	3	2	1	0
CR2	ENSI	STA	STO	SI	AA	CR1	CR0

ENSI, the TWSI Hardware Enable Bit
When ENSI is "0", the SDA and SCL outputs are in a high impedance state. SDA and SCL input signals are ignored, TWSI is in the not-addressed slave state, and STO bit in SICON is forced to "0". No other bits are affected, and, P4.3 (SDA) and P4.2 (SCL) may be used as open drain I/O pins. When ENSI is "1", TWSI is enabled, and, the P4.3 and P4.2 port latches must be set to logic 1 for the following serial communication.

STA, the START Flag
When the STA bit is set to enter a master mode, the TWSI hardware checks the status of the serial bus and generates a START condition if the bus is free. If the bus is not free, then TWSI waits for a STOP condition and generates a START condition after a delay. If STA is set while TWSI is already in a master mode and one or more bytes are transmitted or received, TWSI transmits a repeated START condition. STA may be set at any time. STA may also be set when TWSI is an addressed slave. When the STA bit is reset, no START condition or repeated START condition will be generated.

STO, the STOP Flag
When the STO bit is set while TWSI is in a master mode, a STOP condition is transmitted to the serial bus. When the STOP condition is detected on the bus, the TWSI hardware clears the STO flag. In a slave mode, the STO flag may be set to recover from an bus error condition. In this case, no STOP condition is transmitted to the bus. However, the TWSI hardware behaves as if a STOP condition has been received and switches to the defined not addressed slave receiver mode. The STO flag is automatically cleared by hardware. If the STA and STO bits are both set, then a STOP condition is transmitted to the bus if TWSI is in a master mode (in a slave mode, TWSI generates an internal STOP condition which is not transmitted), and then transmits a START condition.

SI, the Serial Interrupt Flag
When a new TWSI state is present in the SISTA register, the SI flag is set by hardware. And, if the TWSI interrupt is enabled, an interrupt service routine will be serviced. The only state that does not cause SI to be set is state F8H, which indicates that no relevant state information is available. When SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. A high level on the SCL line is unaffected by the serial interrupt flag. SI must be cleared by software. When the SI flag is reset, no serial interrupt is requested, and there is no stretching on the serial clock on the SCL line.

AA, the Assert Acknowledge Flag
If the AA flag is set to “1”, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when:
1) The own slave address has been received.
2) A data byte has been received while TWSI is in the master/receiver mode.
3) A data byte has been received while TWSI is in the addressed slave/receiver mode.

If the AA flag is reset to “0”, a not acknowledge (high level to SDA) will be returned during the acknowledge clock pulse on SCL when:
1) A data has been received while TWSI is in the master/receiver mode.
2) A data byte has been received while TWSI is in the addressed slave/receiver mode.

CR0, CR1 and CR2, the Clock Rate Bits
These three bits determine the serial clock frequency when TWSI is in a master mode. The clock rate is not important when TWSI is in a slave mode because TWSI will automatically synchronize with any clock frequency, which is from a master, up to 100KHz. The various serial clock rates are shown in the following table.

Table: Serial Clock Rates

CR2	CR1	CR0	Serial Clock Rate @ Fosc=12MHz	Fosc divided by
0	0	0	1.5 MHz	8
0	0	1	1M KHz	12
0	1	0	400 KHz	30
0	1	1	200 KHz	60
1	0	0	100 KHz	120
1	0	1	50 KHz	240
1	1	0	25 KHz	480
1	1	1	12.5 KHz	960

Where, Fosc is the system clock.

The Status Register, SISTA, Address=D3H

SISTA is an 8-bit read-only register. The three least significant bits are always 0. The five most significant bits contain the status code. There are a number of possible status codes. When SISTA contains F8H, no serial interrupt is requested. All other SISTA values correspond to defined TWSI states. When each of these states is entered, a status interrupt is requested (SI=1). A valid status code is present in SISTA when SI is set by hardware.

In addition, state 00H stands for a Bus Error. A Bus Error occurs when a START or STOP condition is present at an illegal position, such as inside an address/data byte or just on an acknowledge bit.

SISTA (Address=D3H, TWSI Status Register)

7	6	5	4	3	2	1	0
(b7)	(b6)	(b5)	(b4)	(b3)	(b2)	(b1)	(b0)

17.2. Operating Modes

There are four operating modes for the TWSI: 1) Master/Transmitter mode, 2) Master/Receiver mode, 3) Slave/Transmitter mode and 4) Slave/Receiver mode. Bits STA, STO and AA in SICON decide the next action which the TWSI hardware will take after SI is cleared by software. When the next action is completed, a new status code in SISTA will be updated and SI will be set by hardware in the same time. Now, the interrupt service routine is entered (if the TWSI interrupt is enabled), and the new status code can be used to determine which appropriate routine the software is to branch to.

17.2.1. Master Transmitter Mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver. Before the master transmitter mode can be entered, SICON must be initialized as follows:

SICON

7	6	5	4	3	2	1	0
CR2	ENSI	STA	STO	SI	AA	CR1	CR0
Bit rate	1	0	0	0	x	Bit rate	

CR0, CR1, and CR2 define the serial bit rate. ENSI must be set to logic 1 to enable TWSI. If the AA bit is reset, TWSI will not acknowledge its own slave address or the general call address in the event of another device becoming master of the bus. In other words, if AA is reset, TWSI cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit using the SETB instruction. The TWSI logic will now test the serial bus and generate a START condition as soon as the bus becomes free. When a

START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (SISTA) will be 08H. This status code must be used to vector to an interrupt service routine that loads SIDAT with the slave address and the data direction bit (SLA+W). The SI bit in SICON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in SISTA are possible. There are 18H, 20H, or 38H for the master mode and also 68H, 78H, or B0H if the slave mode was enabled (AA=1). The appropriate action to be taken for each of these status codes is detailed in the following operating flow chart. After a repeated START condition (state 10H), TWSI may switch to the master receiver mode by loading SIDAT with SLA+R.

17.2.2. Master Receiver Mode

In the master receiver mode, a number of data bytes are received from a slave transmitter. SICON must be initialized as in the master transmitter mode. When the start condition has been transmitted, the interrupt service routine must load SIDAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in SICON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in SISTA are possible. They are 40H, 48H, or 38H for the master mode and also 68H, 78H, or B0H if the slave mode was enabled (AA=1). The appropriate action to be taken for each of these status codes is detailed in the following operating flow chart. After a repeated start condition (state 10H), TWSI may switch to the master transmitter mode by loading SIDAT with SLA+W.

17.2.3. Slave Transmitter Mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver. To initiate the slave transmitter mode, SIADR and SICON must be loaded as follows:

SIADR

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	GC

|<---------- Own Slave Address ---------->|

The upper 7 bits are the address to which TWSI will respond when addressed by a master. If the LSB (GC) is set, TWSI will respond to the general call address (00H); otherwise it ignores the general call address.

SICON

7	6	5	4	3	2	1	0
CR2	ENSI	STA	STO	SI	AA	CR1	CR0
x	1	0	0	0	1	x	x

CR0, CR1, and CR2 do not affect TWSI in the slave mode. ENSI must be set to “1” to enable TWSI. The AA bit must be set to enable TWSI to acknowledge its own slave address or the general call address. STA, STO, and SI must be cleared to “0”.

When SIADR and SICON have been initialized, TWSI waits until it is addressed by its own slave address followed by the data direction bit which must be “1” (R) for TWSI to operate in the slave transmitter mode. After its own slave address and the “R” bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from SISTA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status codes is detailed in the following operating flow chart. The slave transmitter mode may also be entered if arbitration is lost while TWSI is in the master mode (see state B0H).

If the AA bit is reset during a transfer, TWSI will transmit the last byte of the transfer and enter state C0H or C8H. TWSI is switched to the not-addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, TWSI does not respond to its own

slave address or a general call address. However, the serial bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate TWSI from the bus.

17.2.4. Slave Receiver Mode

In the slave receiver mode, a number of data bytes are received from a master transmitter. Data transfer is initialized as in the slave transmitter mode.

When SIADR and SICON have been initialized, TWSI waits until it is addressed by its own slave address followed by the data direction bit which must be “0” (W) for TWSI to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from SISTA. This status code is used to vector to an interrupt service routine, and the appropriate action to be taken for each of these status codes is detailed in the following operating flow chart. The slave receiver mode may also be entered if arbitration is lost while TWSI is in the master mode (see status 68H and 78H).

If the AA bit is reset during a transfer, TWSI will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, TWSI does not respond to its own slave address or a general call address. However, the serial bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate from the bus.

17.3. Miscellaneous States

There are two SISTA codes that do not correspond to a defined TWSI hardware state, as described below.

S1STA = F8H:
This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when TWSI is not involved in a serial transfer.

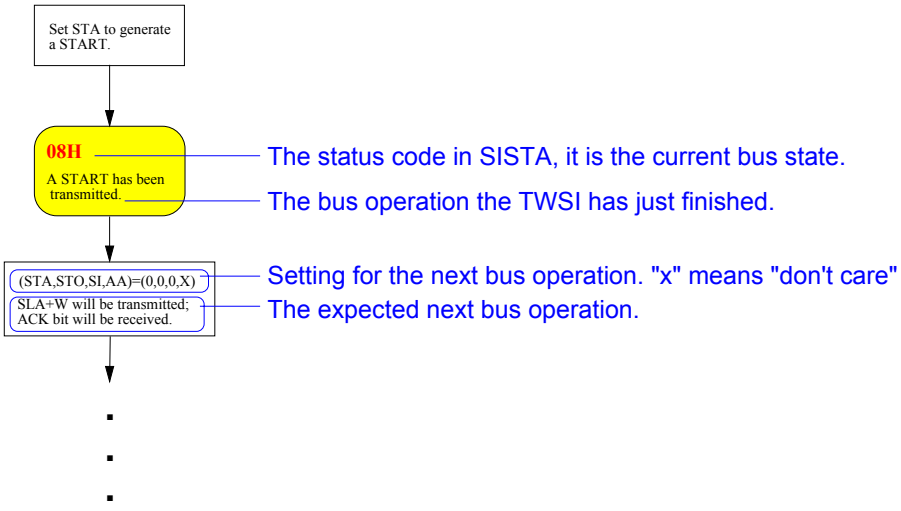
S1STA = 00H:
This status code indicates that a bus error has occurred during an TWSI serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal TWSI signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared by software. This causes TWSI to enter the “not-addressed” slave mode (a defined state) and to clear the STO flag (no other bits in SICON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

17.4. Using the TWSI

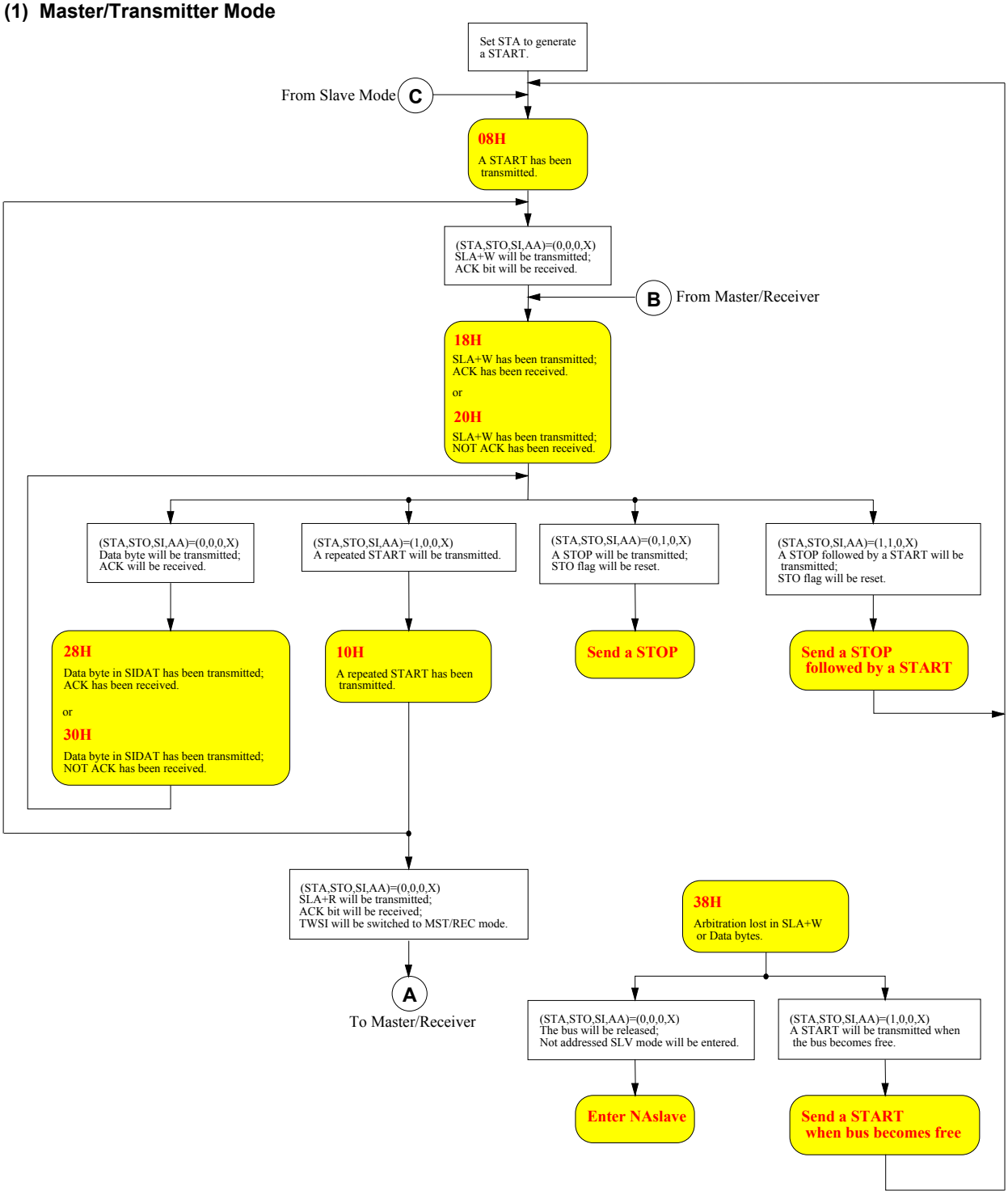
The TWSI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWSI is interrupt-based, the application software is free to carry on other operations during a TWSI byte transfer. Note that the TWSI interrupt enable bit ETWSI bit (AUXIE.6) together with the EA bit allow the application to decide whether or not assertion of the SI Flag should generate an interrupt request. When the SI flag is asserted, the TWSI has finished an operation and awaits application response. In this case, the status register SISTA contains a status code indicating the current state of the TWSI bus. The application software can then decide how the TWSI should behave in the next TWSI bus operation by properly programming the STA, STO and AA bits (in SICON).

The following operating flow charts will instruct the user to use the TWSI using state-by-state operation. First, the user should fill SIADR with its own Slave address (refer to the previous description about SIADR). To act as a master, after initializing the SICON, the first step is to set “STA” bit to generate a START condition to the bus. To act as a slave, after initializing the SICON, the TWSI waits until it is addressed. And then follow the operating flow chart for a number a next actions by properly programming (STA,STO,SI,AA) in the SICON. Since the TWSI hardware will take next action when SI is just cleared, it is recommended to program (STA,STO,SI,AA) by two steps, first STA, STO and AA, then clear SI bit (may use instruction “CLR SI”) for safe operation.

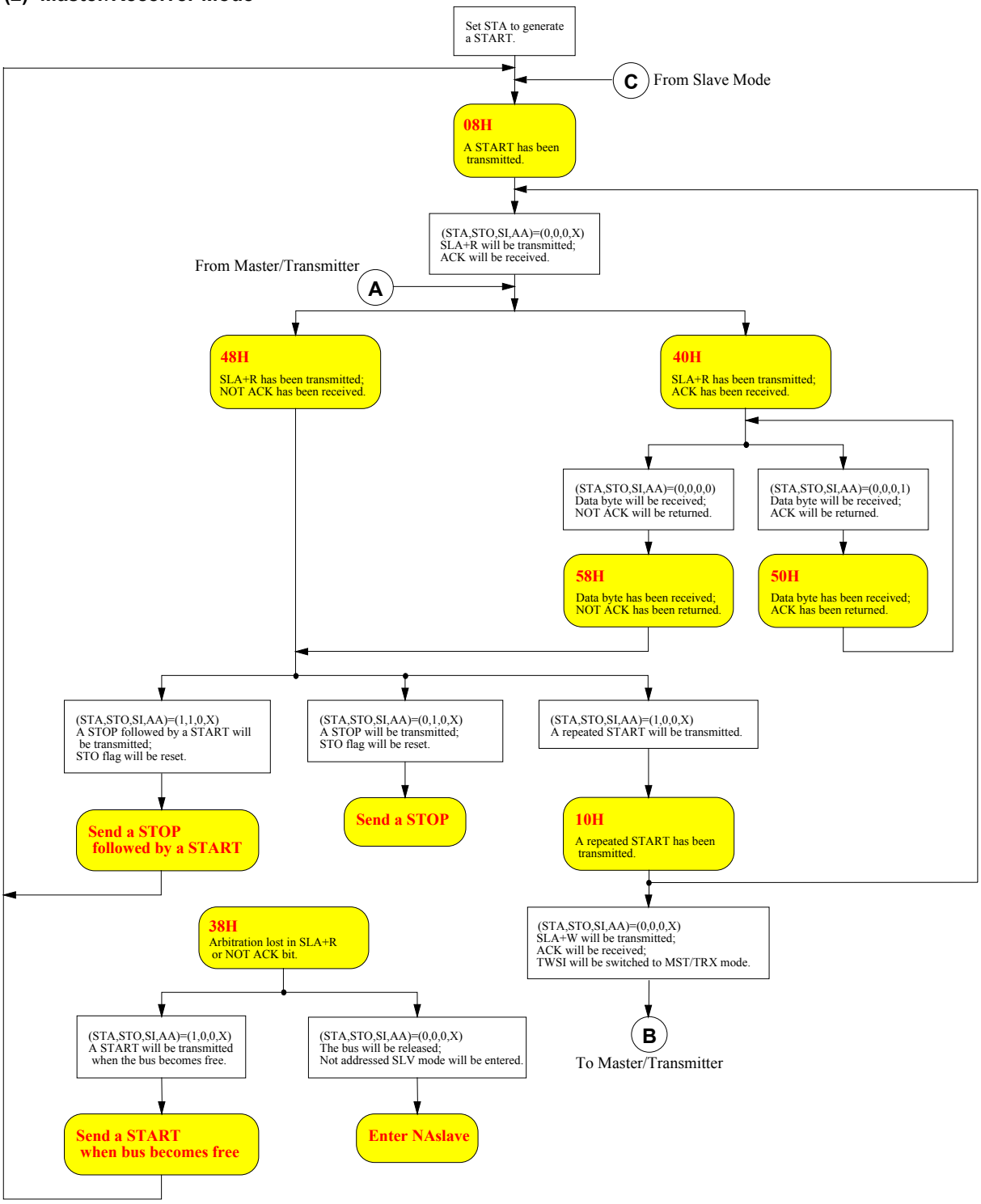
The figure below shows how to read the flow charts.



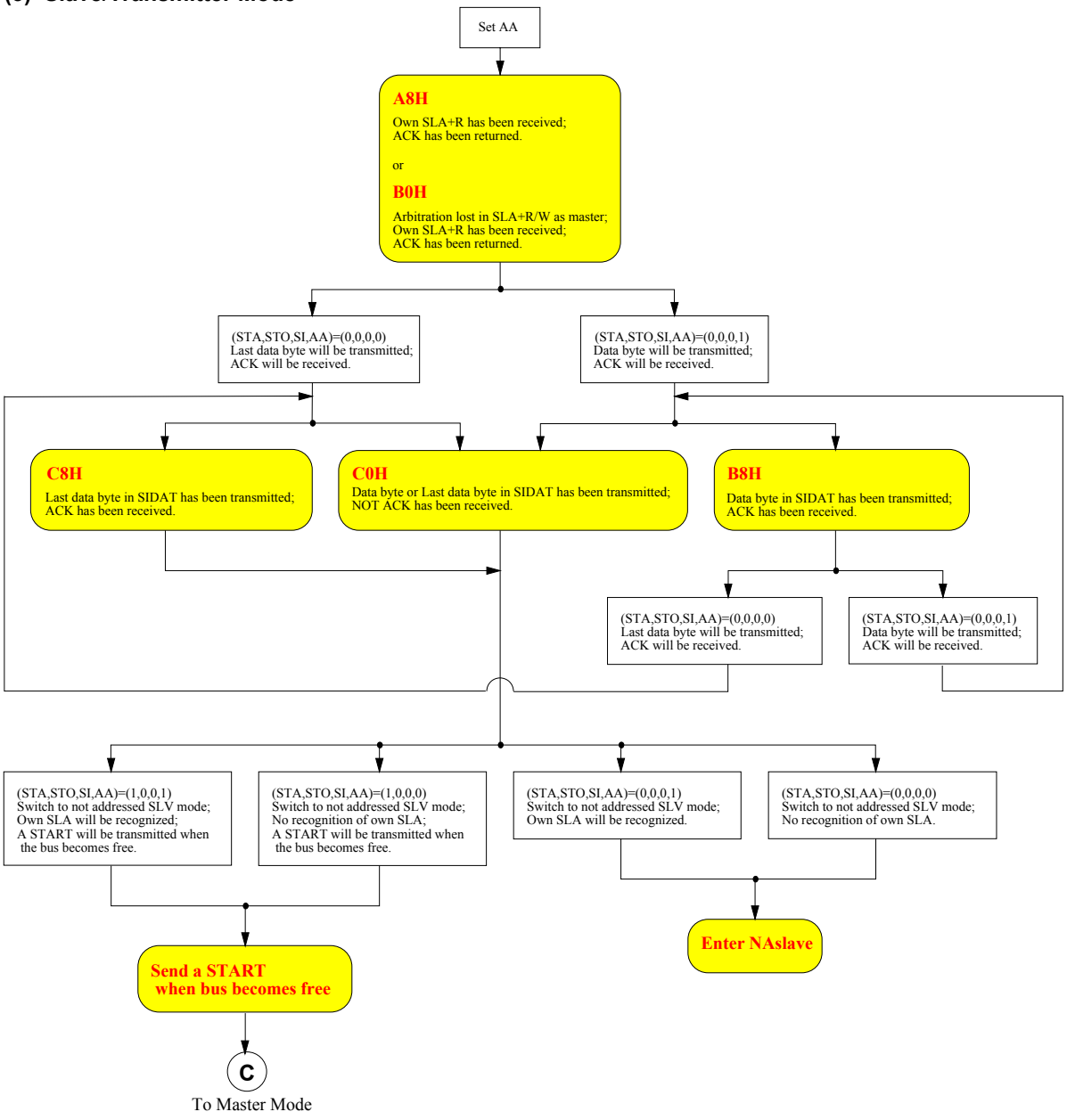
(1) Master/Transmitter Mode



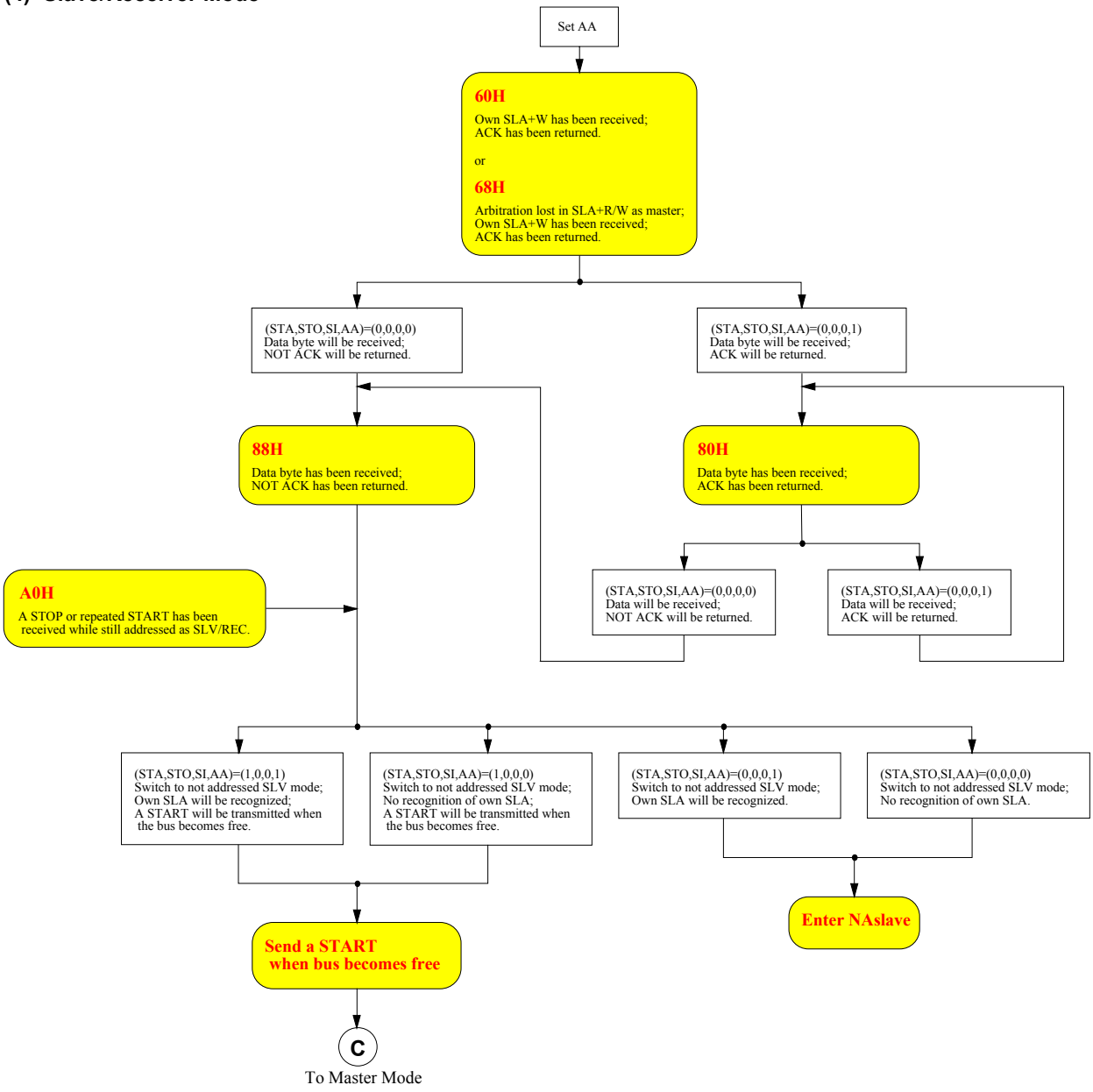
(2) Master/Receiver Mode



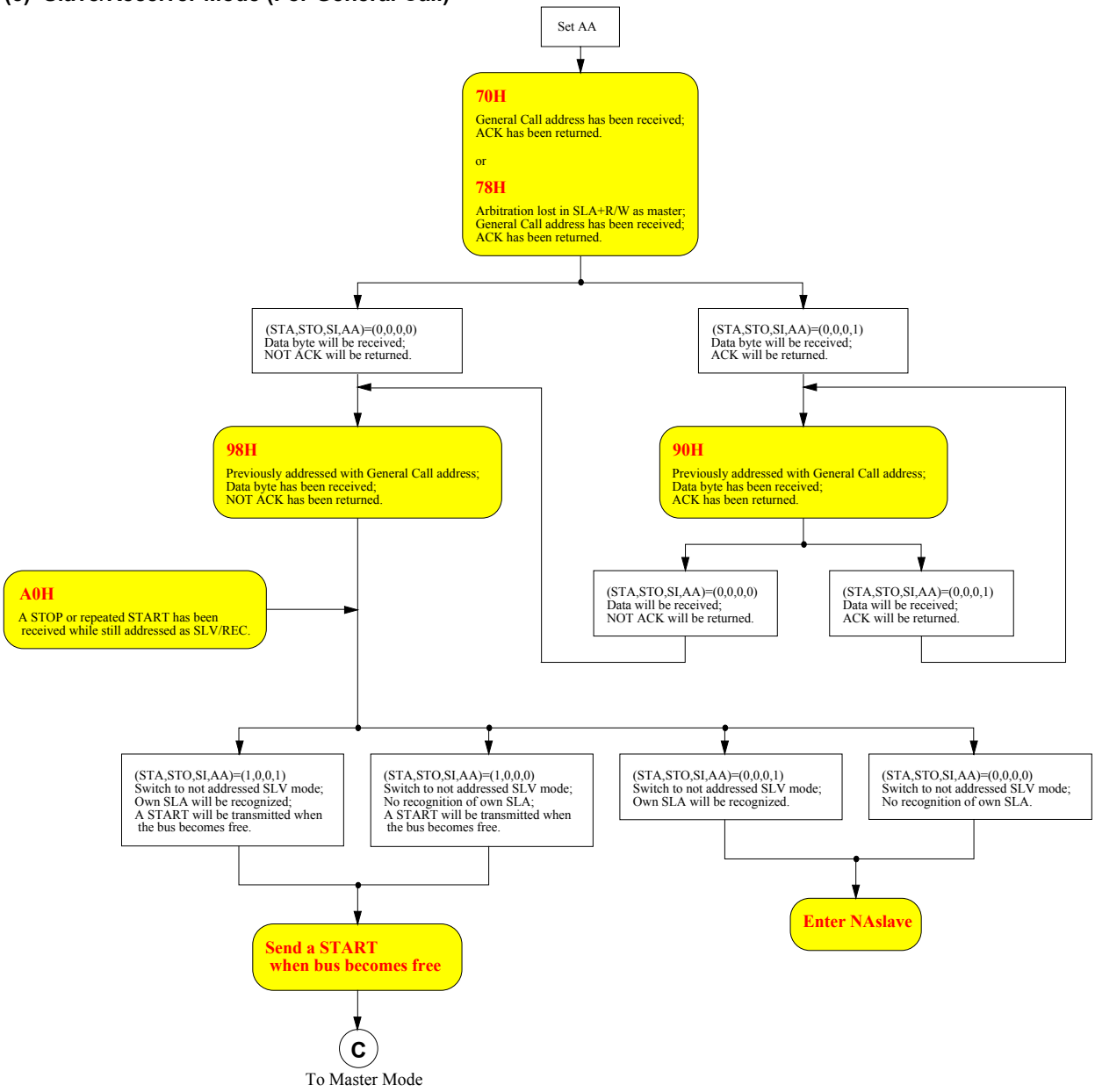
(3) Slave/Transmitter Mode



(4) Slave/Receiver Mode



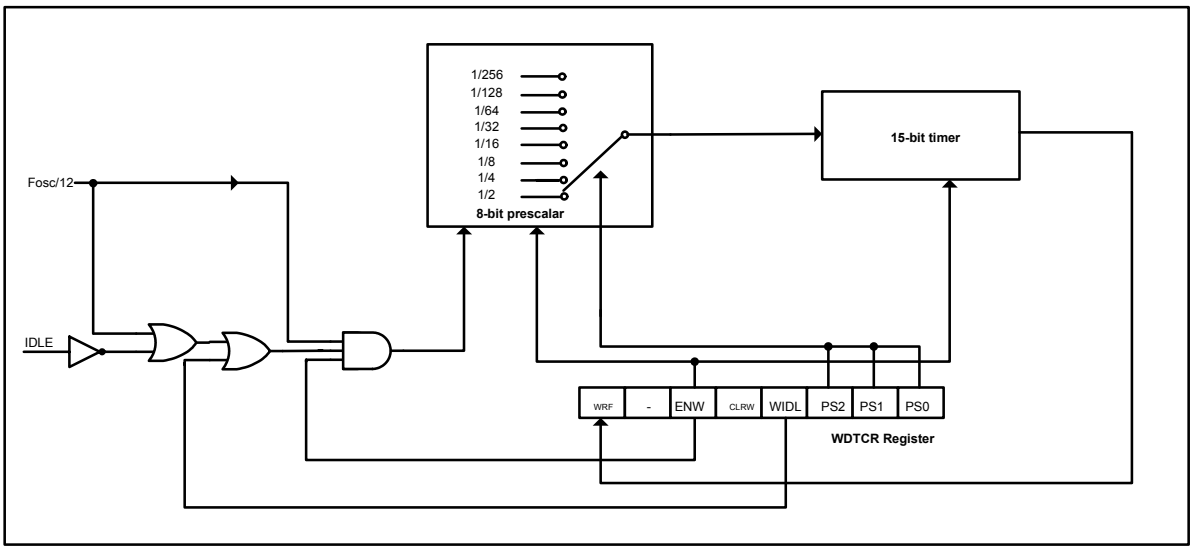
(5) Slave/Receiver Mode (For General Call)



18. One-Time-Enabled Watchdog Timer (WDT)

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upset. The WDT consists of a 15-bits free-running counter, an 8-bit prescaler and a control register (WDTCR). System clock (Fosc) available for the WDT. The block diagram is shown below.

18.1. WDT Block Diagram



To enable the WDT, users must set ENW bit (WDTCR.5). When the WDT is enabled, the counter will increment one by an interval of (***12 x Prescaler / Fosc***). And now the user needs to clear it by writing "1" to the CLRW bit (WDTCR.4) before WDT overflows. When WDT overflows, the MCU will reset itself and re-start.

Why is the WDT called "One-time Enabled"? It is because: *Once the WDT is enabled by setting ENW bit, there is no way to disable it except through power-on reset, which will clear the ENW bit.* The WDTCR register will keep the previous programmed value unchanged after hardware (RST-pin) reset, software reset and WDT reset. For example, if the WDTCR is 0x2D, it still keeps at 0x2D rather than 0x00 after these resets. Only power-on reset can initialize it to 0x00.

WDTCR (Address=E1H, Watch-Dog-Timer Control Register)

7	6	5	4	3	2	1	0
WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0

Note: This is a Write-only register, and it can only be reset to its initial value through power-on reset.

WRF: WDT reset flag. When WDT overflows, this bit is set by H/W. It should be cleared by software.

ENW: Enable WDT. Set to enable WDT. *(Note: Once set, it can only be cleared by power-on reset.)*

CLRW: Clear WDT. "Writing 1" to this bit will clear WDT. *(Note: It has no need to be cleared by "writing 0".)*

WIDL: WDT in Idle mode. Set this bit to let WDT keep counting while the MCU is in the Idle mode.

PS2~PS1: Prescaler select. See the following Table.

PS2	PS1	PS0	Prescaler value
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

18.2. WDT During Idle and Power Down

In the Idle mode, the WIDL bit (WDTCR.3) determines whether WDT counts or not. Set this bit to let WDT keep counting in the Idle mode.

18.3. WDT Automatically Enabled by Hardware

In addition to being initialized by software, the WDTCR register can also be automatically initialized at power-up by the hardware options HWENW, HWWIDL and HWPS[2:0], which should be programmed by a universal Writer or Programmer, as described below.

If HWENW is programmed to “enabled”, then hardware will automatically do the following initialization for the WDTCR register at power-up:

(1) set ENW bit,

(2) load HWWIDL into WIDL bit, and

(3) load HWPS[2:0] into PS[2:0] bits.

For example:

If HWWIDL and HWPS[2:0] are programmed to be 1 and 5, respectively, then **WDTCR** will be initialized to be 0x2D when MCU is powered up, as shown below.

WDTCR (Watch-Dog-Timer Control Register)

7	6	5	4	3	2	1	0
WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0

load HWENW load HWWIDL load HWPS[2:0]

18.4. WDT Overflow Period

The WDT overflow period is determined by the formula:

2^{15} x (12 x Prescaler / Fosc), or ***2^{15} x (12 x Prescaler / RCosc)***.

The following Table shows the WDT overflow period for MCU running at 6MHz and 12MHz. The period is the maximum interval for the user to clear the WDT to prevent from chip reset.

Table: WDT Overflow Period at Fosc = 6MHz & 12MHz

PS2	PS1	PS0	Prescaler value	Fosc=6MHz	Fosc=12MHz
0	0	0	2	131.072 ms	65.536 ms
0	0	1	4	262.144 ms	131.072 ms
0	1	0	8	524.288 ms	262.144 ms
0	1	1	16	1.048 s	524.288 ms
1	0	0	32	2.097 s	1.048 s
1	0	1	64	4.194 s	2.097 s
1	1	0	128	8.389 s	4.194 s
1	1	1	256	16.778 s	8.389 s

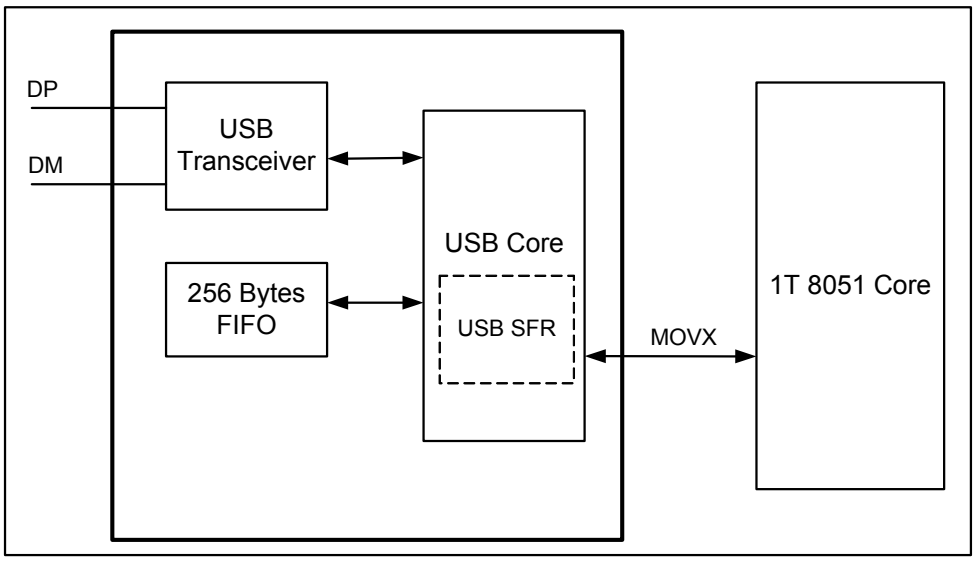
18.5. Sample Code for WDT

Condition: Fosc=6MHz
Target: WDT Overflow Period = 1.048 seconds

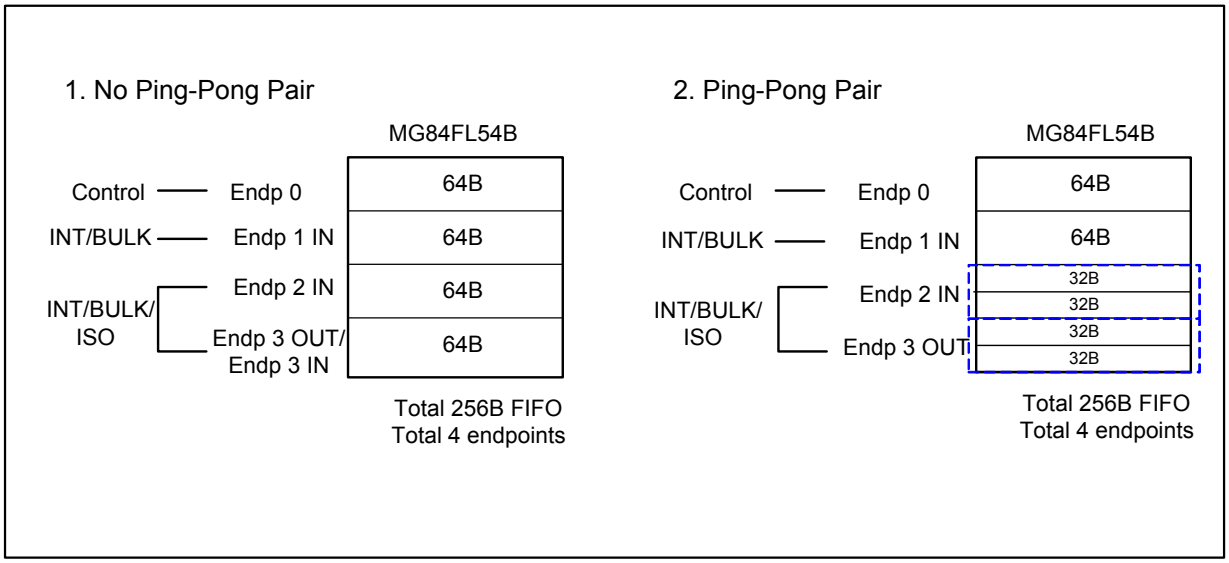
```
WDTCR_buf DATA  30h             ;declare a buffer for WDTCR register
                                ;(because WDTCR is a Write-only register)
start:
         ;...
         ;...

         MOV    WDTCR_buf,#00h  ;clear buffer for WDTCR register

         ANL    WDTCR_buf,#0F8h ;(PS2,PS1,PS0)=(0,1,1), prescaler=16
         ORL    WDTCR_buf,#03h  ;@Fosc=6MHz, WDT_Overflow_Period=1.048s
         MOV    WDTCR,WDTCR_buf ;

         ORL    WDTCR_buf,#20h  ;enable WDT
         MOV    WDTCR,WDTCR_buf ;write to WDTCR register

main_loop:
         ORL    WDTCR_buf,#10h  ;clear WDT
         MOV    WDTCR,WDTCR_buf ;
         ;...
         ;...
         JMP    main_loop

         ANL    WDTCR_buf,#0DFh ;disable WDT
         MOV    WDTCR,WDTCR_buf ;
```

19. Universal Serial Bus (USB)

19.1. USB Block Diagram



19.2. USB FIFO Management



19.3. USB Special Function Registers

To activate the USB operation, the user should enable PLL (by setting bit 'EN_PLL') and enable USB function (by setting bit 'EN_USB'). Clearing bit 'EN_USB' will deactivate the USB operation and let the USB function enter its power-down mode. These two control bits are contained in the CKCON2 register, as follows.

CKCON2 (Address=BFH, Clock Control Register 2)

7	6	5	4	3	2	1	0
-	-	OSCDR0	-	EN_USB	EN_PLL	PLL_RDY	CK_SEL

EN_USB: USB function enable control bit. Set/Clear to enable/disable the USB function.

EN_PLL: PLL enable bit. 1: Enable; 0: Disable

PLL_RDY: It is a read only bit. If this bit is set, indicates the PLL had locked. If cleared, PLL is un-locked.

The special function registers which are dedicated to the USB operation are grouped in the external memory address space and share the addresses 0xFF00 to 0xFFFF with the physical external data memory. That is, the user should use the instruction "***MOVX @DPTR***" to access these USB SFRs.

19.3.1. USB SFR Memory Mapping

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
FFF8H									FFFFH
FFF0H		**EPINDEX**	**TXSTAT**	**TXDAT**	**TXCON**		**TXCNT**		FFF7H
FFE8H									FFEFH
FFE0H		**EPCON**	**RXSTAT**	**RXDAT**	**RXCON**		**RXCNT**		FFE7H
FFD8H	**UADDR**	**IEN**	**UIE**	**UIFLG**	**UIE1**	**UIFLG1**			FFDFH
FFD0H									FFD7H
FFC8H		**UPCON**							FFCFH
FFC0H	**DCON**		**SIOCTL**						FFC7H
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

19.3.2. USB SFR Description

SYMBOL	DESCRIPTION	"MOVX" ADDR	BIT SYMBOL Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	RESET VALUE
DCON	Device Control Register	C0H	-	-	-	-	EP3DIR	-	-	-	xxxx0xxxB
UADDR	USB Address Register	D8H	-	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0	x0000000B
UPCON	USB Power Control Register	C9H	CONEN	-	URWU	-	-	URST	URSM	USUS	0x0xx000B
IEN	Interrupt Enable Register	D9H	-	-	-	-	-	EFSR	EF	-	xxxxx00xB
UIE	USB Interrupt Enable Register	DAH	SOFIE	ASOFIE	-	UTXIE2	-	UTXIE1	URXIE0	UTXIE0	00x0x000B
UIFLG	USB Interrupt Flag Register	DBH	SOFIF	ASOFIF	-	UTXD2	-	UTXD1	URXD0	UTXD0	00x0x000B
UIE1	USB Interrupt Enable Register 1	DCH	-	-	-	-	-	-	URXIE3	UTXIE3	xxxxxx00B
UIFLG1	USB Interrupt Flag Register 1	DDH	-	-	-	-	-	-	URXD3	UTXD3	xxxxxx00B
EPINDEX	Endpoint Index Register	F1H	-	-	-	-	-	-	EPINX1	EPINX0	xxxxxx00B
EPCON	Endpoint Control Register	E1H	RXSTL	TXSTL	RXDBM	TXDBM	RXISO	RXEPEN	TXISO	TXEPEN	00000000B
RXSTAT	Endpoint Receive Status Register	E2H	RXSEQ	RXSETUP	STOVW	EDOVW	RXSOVW	ISOOVW	-	-	000000xxB
RXDAT	FIFO Receive Data Register	E3H	RXD7	RXD6	RXD5	RXD4	RXD3	RXD2	RXD1	RXD0	xxxxxxxxB
RXCON	FIFO Receive Control Register	E4H	RXCLR	-	-	RXFFRC	-	-	-	-	0xx0xxxxB
RXCNT	FIFO Receive Byte Count Register	E6H	-	RXBC6	RXBC5	RXBC4	RXBC3	RXBC2	RXBC1	RXBC0	00000000B
TXSTAT	Endpoint Transmit Status Register	F2H	TXSEQ	-	-	-	TXSOVW	-	-	-	0xxx0xxxB
TXDAT	FIFO Transmit Data Register	F3H	TXD7	TXD6	TXD5	TXD4	TXD3	TXD2	TXD1	TXD0	xxxxxxxxB
TXCON	FIFO Transmit Control Register	F4H	TXCLR	-	-	TXFFRC	-	-	-	-	0xx0xxxxB
TXCNT	FIFO Transmit Byte Count Register	F6H	-	TXBC6	TXBC5	TXBC4	TXBC3	TXBC2	TXBC1	TXBC0	00000000B
SIOCTL	Serial I/O Control Register	C2H	DPI	DMI	-	-	-	-	-	-	xxxxxxxxB

DCON (Device Control Register, Address=C0H, SYS_reset=xxxx-0xxx, Read/Write)

7	6	5	4	3	2	1	0
-	-	-	-	EP3DIR	-	-	-

Bit7~4: Reserved, always write 0.

Bit3: EP3DIR-- USB Endpoint 3 Direction select.
When this bit is set to “1”, EP3 will behave as an IN endpoint.
When this bit is cleared to “0”, EP3 will behave as an OUT endpoint. Default is out endpoint.

Bit2~0: Reserved, always write 0.

UADDR (USB Address Register, Address=D8H, SYS/USB_reset=x000-0000, Read/Write)

7	6	5	4	3	2	1	0
-	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0

Bit7: Reserved.

Bit6~0: UADD[6:0]-- USB Function Address.
This register holds the address for the USB function. During bus enumeration, it is written with a unique value assigned by the host.

UPCON (USB Power Control Register, Address=C9H, SYS/USB_reset=0x0x-x000, Read/Write)

7	6	5	4	3	2	1	0
CONEN	-	URWU	-	-	URST	FRSM	FSUS

Bit7: CONEN-- USB Connect Enable.
Default is cleared to '0' after reset. FW should set '1' to enable connection to upper host/hub.

Bit6: Reserved.

Bit5: URWU-- Function Remote Wake-Up Trigger.
This bit is set by the uC to initiate a remote wake-up on the USB bus when uC is wake-up by external trigger. It will be cleared by hardware when remote-wakeup is completed. Don't set this bit unless the function is suspended

Bit4~3: Reserved.

Bit2: URST-- USB Reset Flag.
Set by hardware when the function detects the USB bus reset. If this bit is set, the chip will generate an USRT interrupt to uC. It would be cleared by firmware when serving the USB reset interrupt. This bit is cleared when firmware writes '1' to it.

Bit1: URSM-- USB Resume Flag.
Set by hardware when the function detects the resume state on the USB bus. If this bit is set, the chip will generate an interrupt to uC. It would be cleared by firmware when serving the function resume interrupt. This bit is cleared when firmware writes '1' to it.

Bit0: USUS-- USB Suspend Flag.
Set by hardware when the function detects the suspend state on the USB bus. If this bit is set, the chip will generate an interrupt to uC. During the function suspend interrupt-service routine, firmware should clear this bit before enter the suspend mode. This bit is cleared when firmware writes '1' to it.

IEN (Interrupt Enable Register, Address=D9H, SYS_reset=xxxx-x00x, Read/Write)

7	6	5	4	3	2	1	0
-	-	-	-	-	EFSR	EF	-

Bit7~3: Reserved.

Bit2: EFSR-- Enable USB Function's Suspend/Resume interrupt.
If this bit is set, enables function's interrupt of UPCON events. Function suspend/resume/remote-wakeup/USB-reset interrupt enable bit. This bit doesn't be reset USB_RESET. Default is cleared.

Bit1: EF-- Enable USB Function's interrupt Flag.
If this bit is set, enables function's interrupt of UIFLG. Transmit/receive done interrupt enable bit for USB function endpoints. This bit doesn't be reset by USB_RESET. Default is cleared.

Bit0: Reserved.

UIE (USB Interrupt Enable Register, Address=DAH, SYS/USB_reset=00x0-x000, Read/Write)

7	6	5	4	3	2	1	0
SOFIE	ASOFIE	-	UTXIE2	-	UTXIE1	URXIE0	UTXIE0

Bit7: SOFIE-- Host SOF received Interrupt Enable.
If this bit is set, enables the Host SOF received interrupt. Default is cleared.

Bit6: ASOFIE-- ART SOF received Interrupt Enable.
If this bit is set, enables the ART SOF received interrupt. Default is cleared.

Bit5: Reserve.

Bit4: UTXIE2-- Function Transmit Interrupt Enable 2.
If this bit is set, enables the transmit done interrupt for USB endpoint 2 (UTXD2). Default is cleared.

Bit3: Reserved.

Bit2: UTXIE1-- Function Transmit Interrupt Enable 1.
If this bit is set, enables the transmit done interrupt for USB endpoint 1 (UTXD1). Default is cleared.

Bit1: URXIE0-- Function Receive Interrupt Enable 0.
If this bit is set, enables the receive done interrupt for USB endpoint 0 (URXD0). Default is cleared.

Bit0: UTXIE0-- Function Transmit Interrupt Enable 0.
If this bit is set, enables the transmit done interrupt for USB endpoint 0 (UTXD0). Default is cleared.

UIFLG (USB Interrupt Flag Register, Address=DBH, SYS/USB_reset=00x0-x000, Read/Write)

7	6	5	4	3	2	1	0
SOFIF	ASOFIF	-	UTXD2	-	UTXD1	URXD0	UTXD0

Bit7: SOFIF-- Host SOF received Interrupt Flag.
This bit is set by hardware when detected a host SOF. UC can read/write-clear on this bit. This bit is cleared when firmware writes '1' to it.

Bit6: ASOFIF-- ART SOF received Interrupt Flag.
This bit is set by hardware when detected an ART SOF. UC can read/write-clear on this bit. This bit is cleared when firmware writes '1' to it.

Bit5: Reserved.

Bit4: UTXD2-- USB Transmit Done Flag for endpoint 2.
This bit is set by hardware when detected a transmit done on endpoint 2. UC can read/write-clear on this bit. This bit is cleared when firmware writes '1' to it.

Bit3: Reserved.

Bit2: UTXD1-- USB Transmit Done Flag for endpoint 1.
This bit is set by hardware when detected a transmit done on endpoint 1. UC can read/write-clear on this bit. This bit is cleared when firmware writes '1' to it.

Bit1: URXD0-- USB Receive Done Flag for endpoint 0.
This bit is set by hardware when detected a receive done on endpoint 0. UC can read/write-clear on this bit. This bit is cleared when firmware writes '1' to it.

Bit0: UTXD0-- USB Transmit Done Flag for endpoint 0.
This bit is set by hardware when detected a transmit done on endpoint 0. UC can read/write-clear on this bit. This bit is cleared when firmware writes '1' to it.

UIE1 (USB Interrupt Enable Register 1, Address=DCH, SYS/USB_reset=xxxx-xx00, Read/Write)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	URXIE3	UTXIE3

Bit7~2: Reserved, always write 0.

Bit1: URXIE3-- USB Function Receive Interrupt Enable 3.
If this bit is set, enables the receive done interrupt for USB endpoint 3 (URXD3). Default is cleared.

Bit0: UTXIE3-- USB Function Transmit Interrupt Enable 3.
If this bit is set, enables the transmit done interrupt for USB endpoint 3 (UTXD3). Default is cleared.

UIFLG1 (USB Interrupt Flag Register 1, Address=DDH, SYS/USB_reset=xxxx-xx00, Read/Write)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	URXD3	UTXD3

Bit7~2: Reserve.

Bit1: URXD3-- USB Receive Done Flag for endpoint 3.
This bit is set by hardware when detected a receive done on endpoint 3. UC can read/write-clear on this bit. This bit is cleared when firmware writes '1' to it. This bit is invalid only if DCON.EP3DIR is set.

Bit0: UTXD3-- USB Transmit Done Flag for endpoint 3.
This bit is set by hardware when detected a transmit done on endpoint 3. UC can read/write-clear on this bit. This bit is cleared when firmware writes '1' to it. This bit is invalid only if DCON.EP3DIR is unset.

EPINDEX (Endpoint Index Register, Address=F1H, SYS/USB_reset=xxxx-x000, Read/Write)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	EPINX1	EPINX0

Bit7~2: Reserved, always write 0.

Bit1~0: EPINX[1:0]-- Endpoint Index Bits [1:0]
2'b00: Function Endpoint 0.
2'b01: Function Endpoint 1.
2'b10: Function Endpoint 2.
2'b11: Function Endpoint 3.

EPCON (Endpoint Control Register, Endpoint-Indexed, Address=E1H, SYS/USB_reset=0000-0000, Read/Write)

7	6	5	4	3	2	1	0
RXSTL	TXSTL	RXDBM	TXDBM	RXISO	RXEPEN	TXISO	TXEPEN

Bit7: RXSTL-- Receive Endpoint Stall.
Set this bit to stall the receive endpoint.

Bit6: TXSTL-- Transmit Endpoint Stall.
Set this bit to stall the transmit endpoint.

Bit5: RXDBM-- Receive Endpoint Double Buffer Mode.
Set this bit to enable the double buffer transfer for OUT transaction. Default is cleared.
This bit is only valid for endpoint 3 receive mode.

Bit4: TXDBM-- Transmit Endpoint Double Buffer Mode.
Set this bit to enable the double buffer transfer for IN transaction. Default is cleared.
This bit is only valid for endpoint 2.

Bit3: RXISO-- Receive Isochronous Type Enable.
Set this bit to configure the endpoint for Isochronous-Out transfer type. When disabled, the endpoint is for Bulk/Interrupt-Out transfer type. The default value is 0.
This bit is only valid for endpoint 3 receive mode.

Bit2: RXEPEN-- Receive Endpoint Enable.
Set this bit to enable the receive endpoint. When disabled, the endpoint does not respond to a valid OUT or SETUP token. This bit in endpoint 0 is enabled after reset.

Bit1: TXISO-- Transmit Isochronous Type Enable.
Set this bit to configure the endpoint for Isochronous-In transfer type. When disabled, the endpoint is for Bulk/Interrupt-In transfer type. The default value is 0.
This bit is only valid for endpoint 2.

Bit0: TXEPEN-- Transmit Endpoint Enable.
Set this bit to enable the transmit endpoint. When disabled, the endpoint does not respond to a valid IN token. This bit in endpoint 0 is enabled after reset.

RXSTAT (Endpoint Receive Status Register, Endpoint-Indexed, Address=E2H, SYS/USB_reset=0000-0xxx, Read/Write)

7	6	5	4	3	2	1	0
RXSEQ	RXSETUP	STOVW	EDOVW	RXSOVW	ISOOVW	-	-

Bit7: RXSEQ-- Receive Endpoint Sequence Bit (read, conditional write).
The bit will be toggled on completion of an ACK handshake in response to an OUT token. This bit can be written by firmware if the RXOVW bit is set when written along with the new RXSEQ value.

Bit6: RXSETUP-- Received Setup Transaction.
This bit is set by hardware when a valid SETUP transaction has been received. Clear this bit upon detection of a SETUP transaction or the firmware is ready to handle the data/status stage of control transfer.

Bit5: STOVW-- Start Overwrite Flag (read-only).
Set by hardware upon receipt of a SETUP token for the control endpoint to indicate that the receive FIFO is being overwritten with new SETUP data. This bit is used only for control endpoints.

Bit4: EDOVW-- End Overwrite Flag.
This flag is set by hardware during the handshake phase of a SETUP transaction. This bit is cleared by firmware to read the FIFO data. This bit is only used for control endpoints.

Bit3: RXSOVW-- Receive Data Sequence Overwrite Bit.
Write '1' to this bit to allow the value of the RXSEQ bit to be overwritten. Writing a '0' to this bit has no effect on RXSEQ. This bit always returns '0' when read.

Bit2: ISOOVW-- Isochronous receive data Overwrite Bit.
This bit is set by hardware as a FIFO access conflict happen when uc read the last data and usb host send the next data in the same time. Firmware can use this bit to make sure whether the data had been overwritten or not. When this bit is set, Firmware should write ‘0’ to clear this bit.

Bit1~0: Reserved.

RXDAT (Receive FIFO Data Register, Endpoint-Indexed, Address=E3H, SYS/USB_reset=xxxx-xxxx, Read-only)

7	6	5	4	3	2	1	0
RXD7	RXD6	RXD5	RXD4	RXD3	RXD2	RXD1	RXD0

Bit7~0: RXD[7:0]-- Receive FIFO Data.

Receive FIFO data specified by EPINDEX is stored and read from this register.

RXCON (Receive FIFO Control Register, Endpoint-Indexed, Address=E4H, SYS/USB_reset=0xxx-0xxx, Write-only)

7	6	5	4	3	2	1	0
RXCLR	-	-	-	RXFFRC	-	-	-

Bit7: RXCLR-- Receive FIFO Clear.
Set this bit to flush the entire receive FIFO. All FIFO statuses are reverted to their reset states. Hardware clears this bit when the flush operation is completed.

Bit6~4: Reserved.

Bit3: RXFFRC-- Receive FIFO Read Complete.
Set this bit to release the receive FIFO when data set read is complete. Hardware clears this bit after the FIFO release operation has been finished.

Bit2~0: Reserved.

RXCNT (Receive FIFO Byte Count Register, Endpoint-Indexed, Address=E6H, SYS/USB_reset=0000-0000, Read-only)

7	6	5	4	3	2	1	0
-	RXBC6	RXBC5	RXBC4	RXBC3	RXBC2	RXBC1	RXBC0

Bit6~0: RXBC[6:0]-- Receive Byte Count.
Store the byte count for the data packet received in the receive FIFO specified by EPINDEX.

TXSTAT (Endpoint Transmit Status Register, Endpoint-Indexed, Address=F2H, SYS/USB_reset=0xxx-0xxx, Read/Write)

7	6	5	4	3	2	1	0
TXSEQ	-	-	-	TXSOVW	-	-	-

Bit7: TXSEQ-- Transmit Endpoint Sequence Bit (read, conditional write).
The bit will be transmitted in the next PID and toggled on a valid ACK handshake of an IN transaction. This bit can be written by firmware if the TXOVW bit is set when written along with the new TXSEQ value.

Bit6~4: Reserved.

Bit3: TXSOVW-- Transmit Data Sequence Overwrite Bit.
Write '1' to this bit to allow the value of the TXSEQ bit to be overwritten. Writing a '0' to this bit has no effect on TXSEQ. This bit always returns '0' when read.

Bit2~0: Reserved.

TXDAT (Transmit FIFO Data Register, Endpoint-Indexed, Address=F3H, SYS/USB_reset=xxxx-xxxx, Write-only)

7	6	5	4	3	2	1	0
TXD7	TXD6	TXD5	TXD4	TXD3	TXD2	TXD1	TXD0

Bit7~0: TXD[7:0]-- Transmit FIFO Data.
Data to be transmitted in the FIFO specified by EPINDEX is written to this register.

TXCON (Transmit FIFO Control Register, Endpoint-Indexed, Address=F4H, SYS/USB_reset=0xxx-0xxx, Write-only)

7	6	5	4	3	2	1	0
TXCLR	-	-	-	TXFFRC	-	-	-

Bit7: TXCLR-- Transmit FIFO Clear.

Set this bit to flush the entire transmit FIFO. All FIFO statuses are reverted to their reset states. Hardware clears this bit when the flush operation is completed.

Bit6~4: Reserved.

Bit3: TXFFRC-- Transmit FIFO Write Complete.
Set this bit to release the transmit FIFO when data set write is complete. Hardware clears this bit after the FIFO release operation has been finished. Firmware should write this bit only after firmware finished writing TXCNT register.

Bit2~0: Reserved.

TXCNT (Transmit FIFO Byte Count Register, Endpoint-Indexed, Address=F6H, SYS/USB_reset=xxxx-xxxx, Write-only)

7	6	5	4	3	2	1	0
-	TXBC6	TXBC5	TXBC4	TXBC3	TXBC2	TXBC1	TXBC0

Bit6~0: TXBC[6:0]-- Transmit Byte Count.
Stored the byte count for the data packet in the transmit FIFO specified by EPINDEX.

SIOCTL (Serial I/O Control Register, Address=C2H, SYS_RESET=xxxx-xxxx, Read-only)

7	6	5	4	3	2	1	0
DPI	DMI	-	-	-	-	-	-

Bit7: DPI-- USB DP port state, read only.
Read the port status on USB DP.

Bit6: DMI-- USB DM port state, read only.
Read the port status on USB DM.

Bit5~0: Reserved.

20. In-System-Programming (ISP)

The Flash program memory supports both parallel programming and serial In-System Programming (ISP). Parallel programming mode offers high-speed programming. ISP allows a device to be reprogrammed in the end product under software control. The capability to field update the application firmware makes a wide range of applications possible.

Prior to using the ISP feature, the user should configure an ISP-memory by a universal Writer or Programmer. Refer to Section “Hardware Option” for the ISP-memory configuration.

The following special function registers are related to ISP:

ISPCR: Address=E7H, ISP Control Register

IFADRH: Address=E3H, ISP Flash Address High Register

IFADRL: Address=E4H, ISP Flash Address Low Register

IFD: Address=E2H, ISP Flash Data Register

SCMD: Address=E6H, ISP Sequential Command Register.

ISPCR (Address=E7H, ISP Control Register)

7	6	5	4	3	2	1	0
ISPEN	SWBS	SWRST	-	-	-	MS1	MS0

Note: The reset value is #00000000B.

ISPEN: Set to enable ISP function.

SWBS:
Software boot select. Set to select booting from ISP-memory, and clear to select booting from AP-memory after software reset.

SWRST: Write ‘1’ to trigger software reset.

MS2~MS0: ISP mode select, as listed below.

MS1	MS0	ISP Mode
0	0	Standby
0	1	Read
1	0	Byte Program
1	1	Page Erase

20.1. Description for ISP Operation

Before doing ISP operation, the user should fill the bits XCKS4~XCKS0 in CKCON register with a proper value. (Refer to Section “System Clock”.)

To do Page Erase (64 Bytes per Page)

Step1: Set [MS1,MS0]=[1,1] in ISPCR register to select Page Erase Mode.
Step2: Fill page address in IFADRH & IFADRL registers.
Step3: Sequentially write 0x46 then 0xB9 to SCMD register to trigger an ISP processing.

To do Byte Program

Step1: Set [MS1,MS0]=[1,0] in ISPCR register to select Byte Program Mode.
Step2: Fill byte address in IFADRH & IFADRL registers.
Step3: Fill data to be programmed in IFD register.
Step4: Sequentially write 0x46 then 0xB9 to SCMD register to trigger an ISP processing.

To do Read

Step1: Set [MS1,MS0]=[0,1] in ISPCR register to select Read Mode.
Step2: Fill byte address in IFADRH & IFADRL registers.
Step3: Sequentially write 0x46 then 0xB9 to SCMD register to trigger an ISP processing.
Step4: Now, the Flash data is in IFD register.

20.2. Demo Program for ISP

```
;*******************************************************************************************
; Demo Program for the ISP
;*******************************************************************************************
IFD       DATA    0E2h
IFADRH    DATA    0E3h
IFADRL    DATA    0E4h
ISPTME    DATA    0E5h
SCMD      DATA    0E6h
ISPCR     DATA    0E7h
;
          MOV     ISPCR,#10000000b  ;ISPCR.7=1, enable ISP

;===============================================================================
; 1. Page Erase Mode (64 bytes per page)
;===============================================================================
          ORL     ISPCR,#03h  ;[MS1,MS0]=[1,1], select Page Erase Mode
          MOV     IFADRH,??   ;fill page address in IFADRH & IFADRL
          MOV     IFADRL,??   ;
          MOV     SCMD,#46h   ;trigger ISP processing
          MOV     SCMD,#0B9h  ;
          ;Now in processing...(CPU will halt here until complete)

;===============================================================================
; 2. Byte Program Mode
;===============================================================================
          ORL     ISPCR,#02h  ;[MS1,MS0]=[1,0], select Byte Program Mode
          ANL     ISPCR,#0FEh ;
          MOV     IFADRH,??   ;fill byte address in IFADRH & IFADRL
          MOV     IFADRL,??   ;
          MOV     IFD,??      ;fill the data to be programmed in IFD
          MOV     SCMD,#46h   ;trigger ISP processing
          MOV     SCMD,#0B9h  ;
          ;Now in processing...(CPU will halt here until complete)

;===============================================================================
; 3. Verify using Read Mode
;===============================================================================

          ANL     ISPCR,#0FDh ;[MS1,MS0]=[0,1], select Byte Read Mode
          ORL     ISPCR,#01h  ;
          MOV     IFADRH,??   ;fill byte address in IFADRH & IFADRL
          MOV     IFADRL,??   ;
          MOV     SCMD,#46h   ;trigger ISP processing
          MOV     SCMD,#0B9h  ;
          ;Now in processing...(CPU will halt here until complete)
          MOV     A,IFD       ;data will be in IFD
          CJNE    A,wanted,ISP_error ;compare with the wanted value
          ...
ISP_error:
          ...
;
```

21. In-Application-Programming (IAP)

The device is *In Application Programmable* (IAP), which allows some region in the Flash memory to be used as non-volatile data storage while the application program is running. This useful feature can be applied to the application where the data must be kept after power off. Thus, there is no need to use an external serial EEPROM (such as 93C46, 24C01, .., and so on) for saving the non-volatile data.

21.1. IAP-memory Boundary/Range

In fact, the operating of IAP is the same as that of ISP except the Flash range to be programmed is different. The programmable Flash range for ISP operating is located within the AP-memory, while the range for IAP operating is located within the configured IAP-memory.

Prior to using the IAP feature, users should configure an IAP-memory through **OR1** (IAPLB) by using a universal Writer or Programmer. The range of the IAP-memory is determined by IAPLB and the ISP starts address as listed below.

IAP lower boundary = IAPLBx256, and

IAP higher boundary = ISP start address –1.

For example, if IAPLB=0x12 and the ISP start address is 0x3800, then the IAP-memory range is located at 0x1200 ~ 0x37FF.

Refer to Section “Hardware Option” for OR1 (IAPLB).

21.2. Update the data in the IAP-memory

Because the IAP-memory is a part of Flash memory, only *Page Erase, no Byte Erase,* is provided for Flash erasing. *To update “one byte” in the IAP-memory, users can not directly program the new datum into that byte*. The following steps show the proper procedure:

Step1) Save the data in the whole page (with 512 bytes) which contains the data to be updated into a buffer.

Step2) Erase this page (***using Page Erase mode of ISP***).

Step3) Update the wanted byte(s) in the buffer.

Step4) Program the updated data out of the buffer into this page (***using Byte Program mode of ISP***).

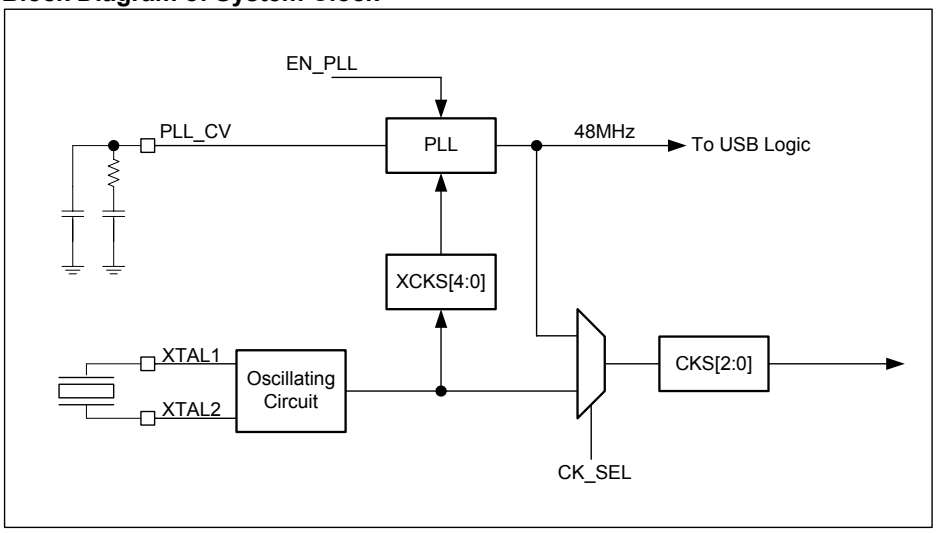
To read the data in the IAP-memory, users can use either the “MOVC A,@A+DPTR” instruction or the ***Read mode of ISP***.

22. System Clock

22.1. Programmable System Clock

The system clock (or CPU clock) of the device is programmable and source-selectable. The user can program the system clock frequency by bits CKS2~CKS0 (CKCON.2 ~ CKCON.0) and select the clock source by bit CK_SEL (CKCON2.0). The block diagram of system clock is shown below.

Block Diagram of System Clock



Note:In the Power down mode, the XTAL oscillating circuit will stop.

CKCON (Address=C7H, Clock Control Register)

7	6	5	4	3	2	1	0
XCKS4	XCKS3	XCKS2	XCKS1	XCKS0	CKS2	CKS1	CKS0

XCKS4~XCKS0 (or XCKS[4:0]): Filled with a proper value according to OSCin, as listed below.

[XCKS4~XCKS0] = OSCin – 1, w*here OSCin=1~32 (MHz).*

For examples,
(1) If OSCin=12MHz, then fill [XCKS4~XCKS0] with 11, i.e., 01011B.
(2) If OSCin=6MHz, then fill [XCKS4~XCKS0] with 5, i.e., 00101B.

CKS2~CKS0: System clock divider select bits, as follows.

CKS2	CKS1	CKS0	Fosc (System Clock)
0	0	0	CLKin → *default*
0	0	1	CLKin /2
0	1	0	CLKin /4
0	1	1	CLKin /8
1	0	0	CLKin /16
1	0	1	CLKin /32
1	1	0	CLKin /64
1	1	1	CLKin /128

CKCON2 (Address=BFH, Clock Control Register 2)

7	6	5	4	3	2	1	0
-	-	OSCDR0	-	EN_USB	EN_PLL	PLL_RDY	CK_SEL

OSCDR0: On-chip XTAL oscillating driving control bits.
0 select the maximum driving, and 1 selects the minimum driving.

EN_PLL: PLL enable bit. 1: Enable; 0: Disable

PLL_RDY: It is a read only flag to indicate the PLL status on ready or not.

CK_SEL: System clock divider input select bit. 1: CLKin=48MHz; 0: CLKin=OSCin.

In the default state, the reset value of CKCON is 0x00 (CK_SEL=0, and CKS2/1/0=000B), so the system clock is the XTAL1-pin signal. The user can modify CKCON at any time to get a new system clock, which will be active just after the modifying is completed.

In the applications which don't care the frequency of system clock, the user can fill CKS2~CKS0 bits with a non-zero value to slow the system clock before entering idle mode to get power saving.

22.2. On-chip XTAL Oscillating Driving Control

To reduce the operating power consumption resulted from the XTAL oscillating circuit, a smart driving control mechanism is designed. The control bit OSCDR0 in CKCON2 is used for the driving control. When powered on, the bit is 0 that select the maximum driving to easily start the XTAL oscillating. And, after MCU successfully runs up, the user can program the bit to some values that can keep XTAL oscillating stable. Refer to the following table for the values.

OSCDR0	XTAL ranges
0	Up to 32MHz(TBD)
1	Down to 1MHz(TBD)

23. Power-On Reset

The CPU will not start to work until the VCC power rises up to the Power-On Reset (POR) voltage. It means the POR state is activated whenever the VCC level is below the POR voltage.

The Power-On Flag, ***POF***, is set to "1" by the activated POR signal. Two occasions make the POR signal activated:

(1) during power up (i.e., cold start), or
(2) when VCC power drops below the POR voltage.

It helps users to check if the running of the CPU begins from *cold reset* (power up) or *warm reset* such as RST-pin reset, software reset (ISPCR.5) or Watchdog Timer reset. The POF bit should be cleared by software.

PCON (Address=87H, Power Control Register)

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL

POF: Power-ON Flag.
POF is set to "1" by hardware during power up (i.e., cold start) or when VCC power drops below the POR voltage. It can be set or reset under software control and is not affected by any warm reset such as RST-pin reset, software reset (ISPCR.5) and WDT reset. Note that it should be cleared by software.

24. Hardware Option

The device has the variety of hardware options, which can only be programmed through a universal Writer/Programmer.

OR0 (Option Register 0)

7	6	5	4	3	2	1	0
ISP_S2	ISP_S1	ISP_S0	-	HWBS	HWBS2	SB	LOCK

HWBS:
0 (enabled): When power-up, MCU will boot from ISP-memory if ISP-memory is configured.
1 (disabled): MCU always boots from AP-memory.

HWBS2:
0 (enabled): In addition to power-up, the reset from RST-pin will also force MCU to boot from ISP-memory if ISP-memory is configured.
1 (disabled): Where MCU boots from is determined by HWBS.

SB:
0 (enabled): Code dumped on a universal Writer or Programmer is scrambled for security.
1 (disabled): Not scrambled.

LOCK:
0 (enabled): Code is locked for security.
1 (disabled): Not locked.

{ISP_S2, ISP_S1, ISP_S0}: See the following Table.

ISP_S2, ISP_S1, ISP_S0	ISP-memory Size	ISP Start Address
0, 0, 0	4K bytes	0x3000
0, 0, 1	3.5K bytes	0x3200
0, 1, 0	3K bytes	0x3400
0, 1, 1	2.5K bytes	0x3600
1, 0, 0	2K bytes	0x3800
1, 0, 1	1.5K bytes	0x3A00
1, 1, 0	1K bytes	0x3C00
1, 1, 1	(No ISP space is configured.)	-

OR1 (Option Register 1)

7	6	5	4	3	2	1	0
IAPLB							

The IAPLB determines the IAP-memory lower boundary. Since a Flash page has ***512*** bytes, the IAPLB must be an even number.

The range of the IAP-memory is determined by IAPLB and the ISP start address as listed below.

IAP lower boundary = IAPLBx256, and

IAP higher boundary = ISP start address –1.

For example, if IAPLB=0x12 and the ISP start address is 0x3800, then the IAP-memory range is located at 0x1200 ~ 0x37FF.

OR2 (Option Register 2)

7	6	5	4	3	2	1	0
-	-	-	PSEN	-	-	-	-

PSEN:
0 (enabled): TBD.
1 (disable): TBD.

OR3 (Option Register 3)

7	6	5	4	3	2	1	0
WDTCR_WP	-	HWENW	-	HWWIDL	HWPS2	HWPS1	HWPS0

WDTCR_WP:
0 (enabled):
If CPU runs in AP-memory, the register WDTCR will be software-write-protected except the bit CLRW.
If CPU runs in ISP-memory, the register WDTCR will be software-write-protected except the bits CLRW, PS2, PS1 and PS0.
1 (disabled): The register WDTCR can be freely written by software.

HWENW: (accompanied with arguments HWWIDL and HWPS[2:0]):
0 (enabled): Automatically enable Watch-dog Timer by hardware when MCU is powered up.
It means that:
In the WDTCR register, H/W will automatically
(1) set ENW bit,
(2) load HWWIDL into WIDL bit, and
(3) load HWPS[2:0] into PS[2:0] bits.

For example:
If HWWIDL and HWPS[2:0] are programmed to be 1 and 5, respectively, then **WDTCR** will be initialized to be 0x2D when MCU is powered up, as shown below.

WDTCR (Watch-Dog-Timer Control Register)

7	6	5	4	3	2	1	0
WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0

load HWENW
load HWWIDL
load HWPS[2:0]

1 (disabled): No action on Watch-dog Timer when MCU powered up.

25. Instruction Set

The Instruction Set is fully compatible with that of the standard 8051 except the execution time, i.e., the number of clock cycles required to execute an instruction. The shortest execution time is just one system clock cycle (1/Fosc) and the longest is 6 system clock cycles (6/Fosc).

Prior to introducing the Instruction Set, users should take care the following notes:

Rn	Working register R0-R7 of the currently selected Register Bank.
direct	128 internal RAM locations, any I/O port, control or status register.
@Ri	Indirect internal RAM location addressed by 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 and LJMP. A branch can be anywhere within the 64K-byte program memory address space.
addr11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.
rel	Signed 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	128 direct bit-addressable bits in internal RAM, any I/O pin, control or status bit.

25.1. Arithmetic Operations

Mnemonic	Description	Byte	Execution Clock Cycles
ARITHMETIC OPERATIONS			
ADD A,Rn	Add register to ACC	1	2
ADD A,direct	Add direct byte to ACC	2	3
ADD A,@Ri	Add indirect RAM to ACC	1	3
ADD A,#data	Add immediate data to ACC	2	2
ADDC A,Rn	Add register to ACC with Carry	1	2
ADDC A,direct	Add direct byte to ACC with Carry	2	3
ADDC A,@Ri	Add indirect RAM to ACC with Carry	1	3
ADDC A,#data	Add immediate data to ACC with Carry	2	2
SUBB A,Rn	Subtract register from ACC with borrow	1	2
SUBB A,direct	Subtract direct byte from ACC with borrow	2	3
SUBB A,@Ri	Subtract indirect RAM from ACC with borrow	1	3
SUBB A,#data	Subtract immediate data from ACC with borrow	2	2
INC A	Increment ACC	1	2
INC Rn	Increment register	1	3
INC direct	Increment direct byte	2	4
INC @Ri	Increment indirect RAM	1	4
INC DPTR	Increment data pointer	1	1
DEC A	Decrement ACC	1	2
DEC Rn	Decrement register	1	3
DEC direct	Decrement direct byte	2	4
DEC @Ri	Decrement indirect RAM	1	4
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	5
DA A	Decimal Adjust ACC	1	4

25.2. Logic Operations

Mnemonic	Description	Byte	Execution Clock Cycles
LOGIC OPERATIONS			
ANL A,Rn	AND register to ACC	1	2
ANL A,direct	AND direct byte to ACC	2	3
ANL A,@Ri	AND indirect RAM to ACC	1	3
ANL A,#data	AND immediate data to ACC	2	2
ANL direct,A	AND ACC to direct byte	2	4
ANL direct,#data	AND immediate data to direct byte	3	4
ORL A,Rn	OR register to ACC	1	2
ORL A,direct	OR direct byte to ACC	2	3
ORL A,@Ri	OR indirect RAM to ACC	1	3
ORL A,#data	OR immediate data to ACC	2	2
ORL direct,A	OR ACC to direct byte	2	4
ORL direct,#data	OR immediate data to direct byte	3	4
XRL A,Rn	Exclusive-OR register to ACC	1	2
XRL A,direct	Exclusive-OR direct byte to ACC	2	3
XRL A,@Ri	Exclusive-OR indirect RAM to ACC	1	3
XRL A,#data	Exclusive-OR immediate data to ACC	2	2
XRL direct,A	Exclusive-OR ACC to direct byte	2	4
XRL direct,#data	Exclusive-OR immediate data to direct byte	3	4
CLR A	Clear ACC	1	1
CPL A	Complement ACC	1	2
RL A	Rotate ACC Left	1	1
RLC A	Rotate ACC Left through the Carry	1	1
RR A	Rotate ACC Right	1	1
RRC A	Rotate ACC Right through the Carry	1	1
SWAP A	Swap nibbles within the ACC	1	1

25.3. Data Transfer

Mnemonic	Description	Byte	Execution Clock Cycles
DATA TRANSFER			
MOV A,Rn	Move register to ACC	1	1
MOV A,direct	Move direct byte o ACC	2	2
MOV A,@Ri	Move indirect RAM to ACC	1	2
MOV A,#data	Move immediate data to ACC	2	2
MOV Rn,A	Move ACC to register	1	2
MOV Rn,direct	Move direct byte to register	2	4
MOV Rn,#data	Move immediate data to register	2	2
MOV direct,A	Move ACC to direct byte	2	3
MOV direct,Rn	Move register to direct byte	2	3
MOV direct,direct	Move direct byte to direct byte	3	4
MOV direct,@Ri	Move indirect RAM to direct byte	2	4
MOV direct,#data	Move immediate data to direct byte	3	3
MOV @Ri,A	Move ACC to indirect RAM	1	3
MOV @Ri,direct	Move direct byte to indirect RAM	2	3
MOV @Ri,#data	Move immediate data to indirect RAM	2	3
MOV DPTR,#data16	Load DPTR with a 16-bit constant	3	3
MOVC A,@A+DPTR	Move code byte relative to DPTR to ACC	1	4
MOVC A,@A+PC	Move code byte relative to PC to ACC	1	4
MOVX A,@Ri *Note1*	Move on-chip XRAM (8-bit address) to ACC	1	3
MOVX A,@DPTR *Note1*	MOVE ON-CHIP XRAM (16-BIT ADDRESS) TO ACC	1	3
MOVX @Ri,A *Note1*	MOVE ACC TO ON-CHIP XRAM (8-BIT ADDRESS)	1	4
MOVX @DPTR,A *Note1*	MOVE ACC TO ON-CHIP XRAM (16-BIT ADDRESS)	1	3
MOVX A,@Ri *Note1*	Move external RAM (8-bit address) to ACC	1	7 *Note2*
MOVX A,@DPTR *Note1*	MOVE EXTERNAL RAM (16-BIT ADDRESS) TO ACC	1	7 *Note2*
MOVX @Ri,A *Note1*	MOVE ACC TO EXTERNAL RAM (8-BIT ADDRESS)	1	7 *Note2*
MOVX @DPTR,A *Note1*	MOVE ACC TO EXTERNAL RAM (16-BIT ADDRESS)	1	7 *Note2*
PUSH direct	PUSH DIRECT BYTE ONTO STACK	2	4
POP direct	POP DIRECT BYTE FROM STACK	2	3
XCH A,Rn	EXCHANGE REGISTER WITH ACC	1	3
XCH A,direct	EXCHANGE DIRECT BYTE WITH ACC	2	4
XCH A,@Ri	EXCHANGE INDIRECT RAM WITH ACC	1	4
XCHD A,@Ri	EXCHANGE LOW-ORDER DIGIT INDIRECT RAM WITH	1	4

Note1:If EXTRAM=1, all “MOVX” instructions are used for the external data memory accessing. And, if EXTRAM=0, all “MOVX” instructions are directed to the on-chip XRAM (if address= 0x0000~0x0FFF) and USB SFRs (if address = 0xFFC0~0xFFFF).

Note2: The cycle time for access of external data memory is:

7 + 2 x (ALE_Stretched_Clocks) + (RW_Stretched_Clocks)

25.4. Boolean Variable Manipulation

Mnemonic	Description	Byte	Execution Clock Cycles
BOOLEAN VARIABLE MANIPULATION			
CLR C	Clear Carry	1	1
CLR bit	Clear direct bit	2	4
SETB C	Set Carry	1	1
SETB bit	Set direct bit	2	4
CPL C	Complement Carry	1	1
CPL bit	Complement direct bit	2	4
ANL C,bit	AND direct bit to Carry	2	3
ANL C,/bit	AND complement of direct bit to Carry	2	3
ORL C,bit	OR direct bit to Carry	2	3
ORL C,/bit	OR complement of direct bit to Carry	2	3
MOV C,bit	Move direct bit to Carry	2	3
MOV bit,C	Move Carry to direct bit	2	4

25.5. Program and Machine Control

Mnemonic	Description	Byte	Execution Clock Cycles
PROAGRAM AND MACHINE CONTROL			
ACALL addr11	Absolute subroutine call	2	6
LCALL addr16	Long subroutine call	3	6
RET	Return from subroutine	1	4
RETI	Return from interrupt subroutine	1	4
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump	2	3
JMP @A+DPTR	Jump indirect relative to DPTR	1	3
JZ rel	Jump if ACC is zero	2	3
JNZ rel	Jump if ACC not zero	2	3
JC rel	Jump if Carry is set	2	3
JNC rel	Jump if Carry not set	2	3
JB bit,rel	Jump if direct bit is set	3	4
JNB bit,rel	Jump if direct bit not set	3	4
JBC bit,rel	Jump if direct bit is set and then clear bit	3	5
CJNE A,direct,rel	Compare direct byte to ACC and jump if not equal	3	5
CJNE A,#data,rel	Compare immediate data to ACC and jump if not equal	3	4
CJNE Rn,#data,rel	Compare immediate data to register and jump if not equal	3	4
CJNE @Ri,#data,rel	Compare immediate data to indirect RAM and jump if not	3	5
DJNZ Rn,rel	Decrement register and jump if not equal	2	4
DJNZ direct,rel	Decrement direct byte and jump if not equal	3	5
NOP	No operation	1	1

26. Absolute Maximum Rating

Parameter	Rating	Unit
Ambient temperature under bias	-55 ~ +125	°C
Storage temperature	-65 ~ + 150	°C
Voltage on any Port I/O Pin or RST with respect to Ground	-0.3 ~ VCC + 0.3	V
Voltage on VCC with respect to Ground	-0.3 ~ +4.2	V
Maximum total current through VCC and Ground	400	mA
Maximum output current sunk by any Port pin	40	mA

*Note: stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the devices at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

27. Electrical Characteristics

27.1. Global DC Electrical Characteristics

VSS = 0V, TA = 25 ℃, VDD_IO= 5.0V, VDD_CORE= VDDA= VDD_PLL= 3.3V, unless otherwise specified

Symbol	Parameter	Test Condition	Limits			Unit
			min	typ	max	
V_{IH1}	Input High voltage for P0,P1,P2 and P3	VDD_IO= 5.0V	2.0			V
V_{IH2}	Input High voltage for RESET pin	VDD_IO= 5.0V	3.5			V
V_{IL}	Input Low voltage	VDD_IO= 5.0V			0.8	V
I_{OL}	Output Low current	V_{PIN} = 0.45V	12	20		mA
I_{OH1}	Output High current(push-pull)	V_{PIN} = 2.4V	12	20		mA
I_{OH2}	Output High current(Quasi-bidirectional)	V_{PIN} = 2.4V		220		uA
I_{IL1}	Logic 0 input current(Quasi-bidirectional)	V_{PIN} = 0.45V		17	50	uA
I_{IL2}	Logic 0 input current(Input-Only)	V_{PIN} = 0.45V		0	10	uA
I_{LK}	Input Leakage current(Open-Drain output)	V_{PIN} = VDD_IO		0	10	uA
I_{H2L}	Logic 1 to 0 transition current	V_{PIN} = 1.8V		230	500	uA
I_{OP}	Operating current	F_{OSC} = 12MHz		12	18	mA
I_{IDLE}	Idle mode current	F_{OSC} = 12MHz		6	9	mA
I_{PD}	Power down current	VDD_IO= 5.0V		0.1	10	uA
R_{RST}	Internal reset pull-down resistance	VDD_IO= 5.0V	100		150	Kohm

VSS = 0V, TA = 25 ℃, VDD_IO= VDD_CORE= VDDA= VDD_PLL= 3.3V, unless otherwise specified

Symbol	Parameter	Test Condition	Limits			Unit
			min	typ	max	
V_{IH1}	Input High voltage for P1 and P3	VDD_IO= 3.3V	2.0			V
V_{IH2}	Input High voltage for RESET pin	VDD_IO= 3.3V	2.8			V
V_{IL}	Input Low voltage	VDD_IO= 3.3V			0.8	V
I_{OL}	Output Low current	V_{PIN} = 0.45V	8	14		mA
I_{OH1}	Output High current(push-pull)	V_{PIN} = 2.4V	4	8		mA
I_{OH2}	Output High current(Quasi-bidirectional)	V_{PIN} = 2.4V		64		uA
I_{IL1}	Logic 0 input current(Quasi-bidirectional)	V_{PIN} = 0.45V		7	50	uA
I_{IL2}	Logic 0 input current(Input-Only)	V_{PIN} = 0.45V		0	10	uA
I_{LK}	Input Leakage current(Open-Drain output)	V_{PIN} = V_{CC}		0	10	uA
I_{H2L}	Logic 1 to 0 transition current(P1,3)	V_{PIN} = 1.4V		100	600	uA
I_{OP}	Operating current	F_{OSC} = 12MHz		9	14.5	mA
I_{IDLE}	Idle mode current	F_{OSC} = 12MHz		3.5	5.3	mA
I_{PD}	Power down current	VDD_IO= 3.3V		0.1	10	uA
R_{RST}	Internal reset pull-down resistance	VDD_IO= 3.3V	160		240	Kohm

27.2. USB Transceiver Electrical Characteristics

VSS = 0V, TA = 25 ℃, VDD_IO= 2.4V~5.5V, VDD_CORE= VDDA= VDD_PLL= 3.3V, unless otherwise specified

Symbol	Parameter	Test Condition	Limits			Unit
			min	typ	max	
Transmitter						
V_{OH}	Output High Voltage		2.8			V
V_{OL}	Output Low Voltage				0.8	V
V_{CRS}	Output Cross Over point		1.3		2.0	V
Z_{DRVH}	Output Impedance on Driving High		28		44	ohm
Z_{DRVL}	Output Impedance on Driving Low		28		44	ohm
R_{PU}	Pull-Up Resistance	On DP	1.425	1.5	1.575	Kohm
T_R	Output Rise Time		4		20	ns
T_F	Output Fall Time		4		20	ns
Receiver						
V_{DI}	Differential Input Sensitivity	| DP – DM |	0.2			V
V_{CM}	Differential Input Common Mode Range		0.8		2.5	V
I_L	Input Leakage current	Pull-up Disabled		<1.0		uA

28. Field Applications

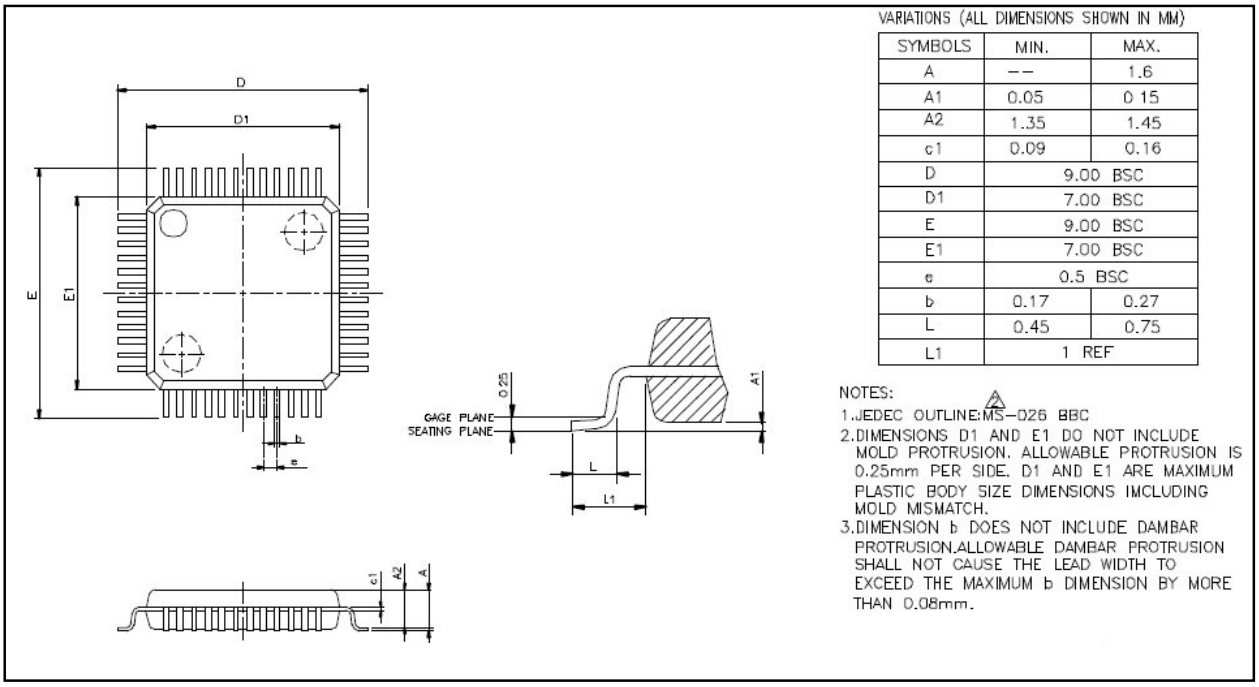
- Home Appliance
- Healthcare
- POS Control
- Wireless Dongle
- Joy Stick
- Wireless Keyboard/Mouse

29. Order Information

Part Number	Temperature Range	Package	Packing	Operation Voltage
MG84FL54BD	-40℃~85℃	LQFP-48	Tray	3.3V

30. Package Dimension

MG84FL54BD (LQFP-48)



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	MAX.
A	--	1.6
A1	0.05	0 15
A2	1.35	1.45
c1	0.09	0.16
D	9.00 BSC	
D1	7.00 BSC	
E	9.00 BSC	
E1	7.00 BSC	
e	0.5 BSC	
b	0.17	0.27
L	0.45	0.75
L1	1 REF	

NOTES:

1. JEDEC OUTLINE: MS-026 BBC
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. D1 AND E1 ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08mm.

31. Revision History

Version	Date	Page	Description
V0.96	2007/11		- Initial public data sheet.
V0.97	2007/12	P95,96	- Add maximum rating and Electrical Characteristics.
V0.98	2008/01	P7 P9,10 P91 P95,96	- Extend flash data retention from 7 to 100 years. - Add T2CKO on P10. - Modify R_{RST} max/min value in Dc Table. - Finalize Electrical Characteristics.