MEGAWIN
MAKE YOU WIN

8051-Based MCU

MG82FE/L564
Data Sheet

Version: A1.01

2015/09 version A1.01

Features

- Enhanced 80C51 Central Processing Unit
- 64K bytes of MG82Fx564 on-chip flash memory with ISP/IAP capability
  - ISP memory zone as 1.5KB(default)
  - Flexible IAP size. 2.5KB(default)
  - Code protection for flash memory access

Part No.	AP Flash ROM size	IAP size
MG82Fx564	60KB (Max.)	2.5KB (Min.)

- 256 bytes scratch-pad RAM and 1024 bytes expanded RAM (XRAM)
- Dual data pointer.
- Variable length MOVX for slow SRAM or peripherals.
- Three 16-bit timer/counter, Timer 0, Timer 1 and Timer 2.
  - T0CKO on P34, T1CKO on P35 and T2CKO on P10
  - X12 mode enabled for T0/T1/T2.
- Programmable 16-bit counter/timer Array (PCA) with 6 compare/capture modules
  - Capture mode
  - 16-bit software timer mode
  - High speed output mode
  - 8/10/12/16-bit PWM (Pulse Width Modulator) mode with phase shift function
- Enhanced UART (S0)
  - Framing Error Detection
  - Automatic Address Recognition
  - a speed improvement mechanism (X2/X4 mode)
- Secondary UART (S1)
  - Dedicated Baud Rate Generator
  - S1 shares baud rate generator with S0.
- Interrupt controller
  - 14 sources, four-level-priority interrupt capability
  - Four external interrupt inputs, nINT0, nINT1, nINT2 and nINT3.
  - nINT0/nINT1 trigger type: Low Level or Falling Edge
  - nINT2/nINT3: Low Level, Falling Edge, High Level or Rising Edge
- 10-Bit ADC
  - Programmable throughput up to 200ksps
  - Up to 8 external inputs (Single-ended)
- Master/Slave SPI serial interface
- Keypad Interrupt on Port 2
- Programmable Watchdog Timer
  - one time enabled by CPU or power-on.
  - WDT operating option in MCU power-down.
- Maximum 45 GPIOs in LQFP48 package.
  - P0, P1, P2, P3, P4, P5 can be configured to quasi-bidirectional, push-pull output, open-drain output and input only
  - P6.0 and P6.1 serve quasi-bidirectional mode only and shared with XTAL2 and XTAL1
  - Maximum 41 GPIOs in PQFP44 package.
- Two power control modes: idle mode and power-down mode.
  - All interrupts can wake up IDLE mode.
  - Four external interrupt and keypad interrupt can wake up Power-Down mode.
- Brown-Out Detector: 4.2V for E-series VDD=5V and 2.4V for L-series VDD=3V
  - Option to reset CPU
  - Option to interrupt CPU.

- Operating Voltage:
  - 4.5V~5.5V for MG82Fx564
  - 2.4V~3.6V for MG82Fx564, minimum 2.7V requirement in flash write operation (ISP/IAP)
- Operating Temperature:
  - Industrial (-40℃ to +85℃)*
- Maximum Operation frequency : 24MHz(External crystal)
  - External crystal mode
  - Internal High frequency RC oscillator (22.1184MHz) (Default)
    +/- 1% frequency drift @ 25℃,
    +/- 2% frequency drift @ -20 ~ 50℃,
    +/- 4% frequency drift @ -40 ~ 85℃,.
  - Internal High frequency RC Oscillator output on XTAL2/P6.0.
  - External clock input on XTAL2/P6.0.
  - Internal Low frequency RC Oscillator support.
- Package Types:
  - LQFP48: MG82Fx564AD
  - PQFP44: MG82Fx564AF
  - PDIP40: MG82Fx564AE

*: Tested by sampling.

Content

1. Description

The MG82Fx564 is a single-chip microcontroller based on a high performance 1-T architecture 80C51 CPU that executes instructions in 1~7 clock cycles (about 6~7 times the rate of a standard 8051 device), and has an 8051 compatible instruction set. Therefore at the same performance as the standard 8051, the MG82Fx564 can operate at a much lower speed and thereby greatly reduce the power consumption.

The MG82Fx564 has 64K bytes of embedded Flash memory for code and data. The Flash memory can be programmed in ISP (In-System Programming) mode. And, it also provides the In-Application Programming (IAP) capability. ISP allow the user to download new code without removing the microcontroller from the actual end product; IAP means that the device can write non-volatile data in the Flash memory while the application program is running. There needs no external high voltage for programming due to its built-in charge-pumping circuitry.

The MG82Fx564 retains all features of the standard 80C52 with 256 bytes of scratch-pad RAM, four 8bit I/O ports, two external interrupts, a multi-source 4-level interrupt controller and three timer/counters. In addition, the MG82Fx564 has two extra I/O ports (P4, P5[4:0]), one on-chip XRAM of 1024 bytes, two extra external interrupts with High/low trigger option, 10-btis ADC, a 6-channel PCA, SPI, secondary UART, keypad interrupt, a one-time enabled Watchdog Timer, a Brown-out Detector, an on-chip crystal oscillator(shared with P6.0 and P6.1), a high precision internal oscillator, a more versatile serial channel that facilitates multiprocessor communication (EUART) and a speed improvement mechanism (X2/X4 mode).

The MG82Fx564 has two power-saving modes and an 8-bit system clock pre-scaler to reduce the power consumption. In the Idle mode the CPU is frozen while the peripherals and the interrupt system are still operating. In the Power-Down mode the RAM and SFRs' value are saved and all other functions are inoperative; most importantly, in the Power-down mode the device can be waked up by the external interrupts. And, the user can further reduce the power consumption by using the 8-bit system clock pre-scaler to slow down the operating speed.

Additionally, the MG82Fx564 is equipped with the Megawin proprietary On-Chip Debug (OCD) interface for In-Circuit Emulator (ICE). The OCD interface provides on-chip and in-system non-intrusive debugging without any target resource occupied. Several operations necessary for an ICE are supported such as Reset, Run, Stop, Step, Run to Cursor and Breakpoint Setting. The user has no need to prepare any development board during firmware developing or the socket adapter used in the traditional ICE probe head. All the thing the user needs to do is to prepare a 4-pin connector for the dedicated OCD interface. This powerful feature makes the developing very easy for any user.

2. Order information

Part Number	Package	Operation Voltage
MG82Fx564AE	PDIP-40	x=L: 3V / E: 5V
MG82Fx564AF	PQFP-44	x=L: 3V / E: 5V
MG82Fx564AD	LQFP-48	x=L: 3V / E: 5V

3. Pin Description

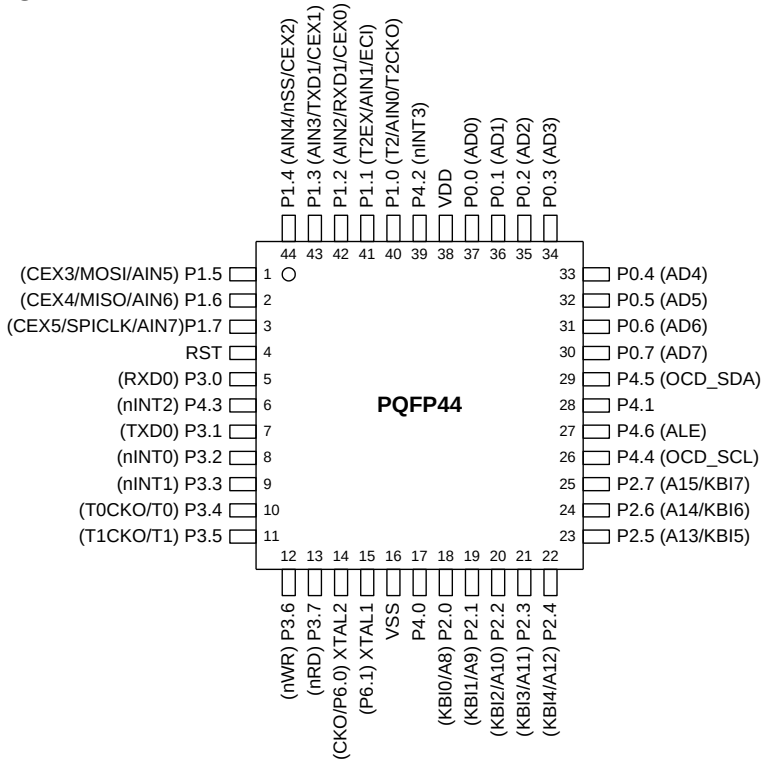
3.1. Pin Definition

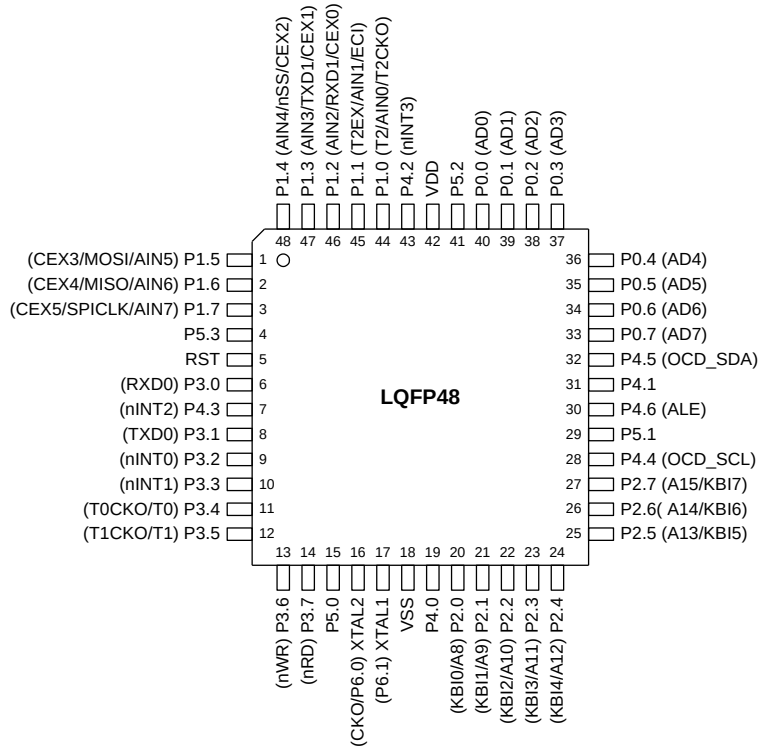
MNEMONIC	PIN NUMBER					I/O TYPE	DESCRIPTION
	40-Pin DIP		44-Pin PQFP	48-Pin LQFP			
P0.0 (AD0)	39		37	40		I/O	* Port 0.0. * AD0: multiplexed A0/D0 during external data memory access.
P0.1 (AD1)	38		36	39		I/O	* Port 0.1. * AD1: multiplexed A1/D1 during external data memory access.
P0.2 (AD2)	37		35	38		I/O	* Port 0.2. * AD2: multiplexed A2/D2 during external data memory access.
P0.3 (AD3)	36		34	37		I/O	* Port 0.3. * AD3: multiplexed A3/D3 during external data memory access.
P0.4 (AD4)	35		33	36		I/O	* Port 0.4. * AD4: multiplexed A4/D4 during external data memory access.
P0.5 (AD5)	34		32	35		I/O	* Port 0.5. * AD5: multiplexed A5/D5 during external data memory access.
P0.6 (AD6)	33		31	34		I/O	* Port 0.6. * AD6: multiplexed A6/D6 during external data memory access.
P0.7 (AD7)	32		30	33		I/O	* Port 0.7. * AD7: multiplexed A7/D7 during external data memory access.
P1.0 (T2) (AIN0) (T2CKO)	1		40	44		I/O	* Port 1.0. * T2: Timer/Counter 2 external input. * AIN0: ADC channel-0 analog input. * T2CKO: programmable clock-out from Timer 2.
P1.1 (T2EX) (AIN1) (ECI)	2		41	45		I/O	* Port 1.1. * T2EX: Timer/Counter 2 Reload/Capture/Direction control. * AIN1: ADC channel-1 analog input. * ECI: PCA external clock input.
P1.2 (AIN2) (RXD1) (CEX0)	3		42	46		I/O	* Port 1.2. * AIN2: ADC channel-2 analog input. * RXD1: UART1 serial input port. * CEX0: PCA module-0 external I/O.
P1.3 (AIN3) (TXD1) (CEX1)	4		43	47		I/O	* Port 1.3. * AIN3: ADC channel-3 analog input. * TXD1: UART1 serial output port. * CEX1: PCA module-1 external I/O.
P1.4 (AIN4) (nSS) (CEX2)	5		44	48		I/O	* Port 1.4. * AIN4: ADC channel-4 analog input. * nSS: SPI Slave select. * CEX2: PCA module-2 external I/O.
P1.5 (AIN5) (MOSI) (CEX3)	6		1	1		I/O	* Port 1.5. * AIN5: ADC channel-5 analog input. * MOSI: SPI master out & slave in. * CEX3: PCA module-3 external I/O.
P1.6 (AIN6) (MISO) (CEX4)	7		2	2		I/O	* Port 1.6. * AIN6: ADC channel-6 analog input. * MISO: SPI master in & slave out. * CEX4: PCA module-4 external I/O.
P1.7 (AIN7) (SPICLK) (CEX5)	8		3	3		I/O	* Port 1.7. * AIN7: ADC channel-7 analog input. * SPICLK: SPI clock, output for master and input for slave. * CEX5: PCA module-5 external I/O.

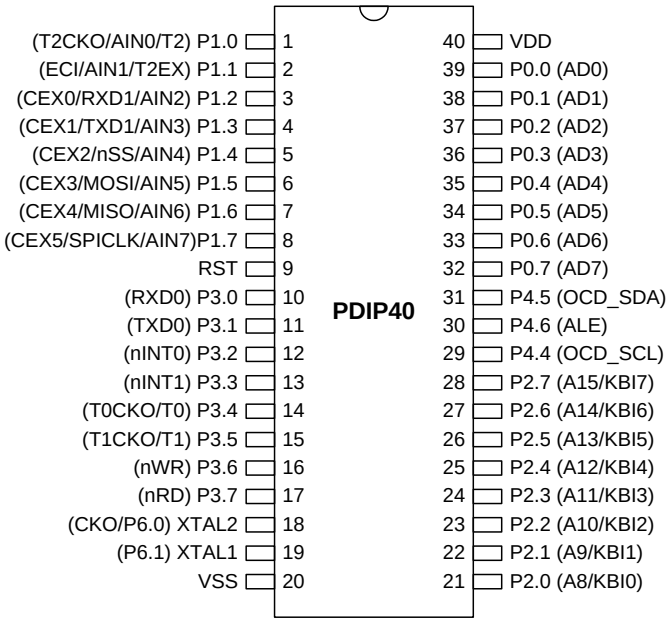
P2.0 (A8) (KBI0)	21		18	20		I/O	* Port 2.0. * A8: A8 output during external data memory access. * KBI0: keypad input 0.
P2.1 (A9) (KBI1)	22		19	21		I/O	* Port 2.1. * A9: A9 output during external data memory access. * KBI1: keypad input 1.
P2.2 (A10) (KBI2)	23		20	22		I/O	* Port 2.2. * A10: A10 output during external data memory access. * KBI2: keypad input 2.
P2.3 (A11) (KBI3)	24		21	23		I/O	* Port 2.3. * A11: A11 output during external data memory access. * KBI3: keypad input 3.
P2.4 (A12) (KBI4)	25		22	24		I/O	* Port 2.4. * A12: A12 output during external data memory access. * KBI4: keypad input 4.
P2.5 (A13) (KBI5)	26		23	25		I/O	* Port 2.5. * A13: A13 output during external data memory access. * KBI5: keypad input 5.
P2.6 (A14) (KBI6)	27		24	26		I/O	* Port 2.6. * A14: A14 output during external data memory access. * KBI6: keypad input 6.
P2.7 (A15) (KBI7)	28		25	27		I/O	* Port 2.7. * A15: A15 output during external data memory access. * KBI7: keypad input 7.
P3.0 (RXD0)	10		5	6		I/O	* Port 3.0. * RXD0: UART0 serial input port.
P3.1 (TXD0)	11		7	8		I/O	* Port 3.1. * TXD0: UART0 serial output port.
P3.2 (nINT0)	12		8	9		I/O	* Port 3.2. * nINT0: external interrupt 0 input.
P3.3 (nINT1)	13		9	10		I/O	* Port 3.3. * nINT1: external interrupt 1 input.
P3.4 (T0) (T0CKO)	14		10	11		I/O	* Port 3.4. * T0: Timer/Counter 0 external input. * T0CKO: programmable clock-out from Timer 0.
P3.5 (T1) (T1CKO)	15		11	12		I/O	* Port 3.5. * T1: Timer/Counter 1 external input. * T1CKO: programmable clock-out from Timer 1.
P3.6 (nWR)	16		12	13		I/O	* Port 3.6. * nWR: external data memory write strobe.
P3.7 (nRD)	17		13	14		I/O	* Port 3 bit-7. * nRD: external data memory read strobe.
P4.0	-		17	19		I/O	* Port 4.0.
P4.1	-		28	31		I/O	* Port 4.1.
P4.2 (nINT3)	-		39	43		I/O	* Port 4.2. * nINT3: external interrupt 3 input.
P4.3 (nINT2)	-		6	7		I/O	* Port 4.3. * nINT2: external interrupt 2 input.
P4.4 (OCD_SCL)	29		26	28		I/O	* Port 4.4. * OCD_SCL: OCD interface, serial clock.
P4.5	31		29	32		I/O	* Port 4.5. * OCD_SDA: OCD interface, serial data.
P4.6 (ALE)	30		27	30		I/O	* Port 4.6. * ALE: Address Latch Enable, output pulse for latching the low byte of the address during an access cycle to external data memory.
P5.0	--		--	15		I/O	* Port 5.0.
P5.1	--		--	29		I/O	* Port 5.1.
P5.2	--		--	41		I/O	* Port 5.2
P5.3	--		--	4		I/O	* Port 5.3
P6.0	18		14	16		I/O	* Port 6.0. It is only accessed in SFR page "**F**".

(CKO) (ECKI) (XTAL2)						O I O	* CKO: Enable internal High frequency RC-Oscillator output. * ECKI: In external clock input mode, this is clock input pin. *XTAL2: Output of on-chip crystal oscillating circuit.
P6.1 (XTAL1)	19		15	17		I/O I	* Port 6.1. It is only accessed in SFR page "**F**". *XTAL1: Input of on-chip crystal oscillating circuit.
RST	9		4	5		I	*RST: External RESET input, high active.
VDD	40		38	42		I	Power supply. Connect to 5V for E-series device and connect to 3.3V for L-series device.
VSS	20		16	18		I	Ground, 0 V reference.

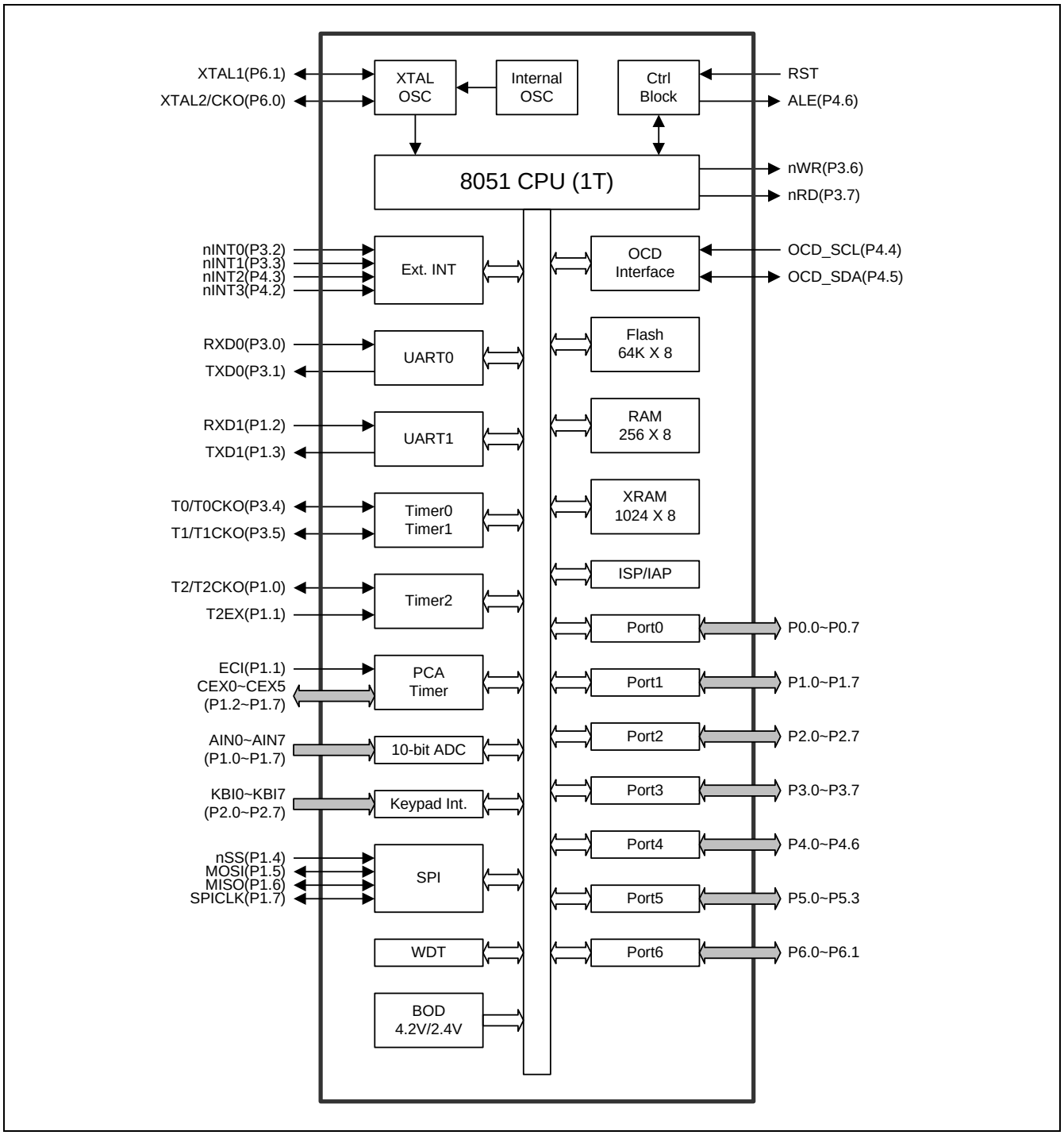
3.2. Package Configuration






(T2CKO/AIN0/T2) P1.0 1
(ECI/AIN1/T2EX) P1.1 2
(CEX0/RXD1/AIN2) P1.2 3
(CEX1/TXD1/AIN3) P1.3 4
(CEX2/nSS/AIN4) P1.4 5
(CEX3/MOSI/AIN5) P1.5 6
(CEX4/MISO/AIN6) P1.6 7
(CEX5/SPICLK/AIN7)P1.7 8
RST 9
(RXD0) P3.0 10
(TXD0) P3.1 11
(nINT0) P3.2 12
(nINT1) P3.3 13
(T0CKO/T0) P3.4 14
(T1CKO/T1) P3.5 15
(nWR) P3.6 16
(nRD) P3.7 17
(CKO/P6.0) XTAL2 18
(P6.1) XTAL1 19
VSS 20
PDIP40
40 VDD
39 P0.0 (AD0)
38 P0.1 (AD1)
37 P0.2 (AD2)
36 P0.3 (AD3)
35 P0.4 (AD4)
34 P0.5 (AD5)
33 P0.6 (AD6)
32 P0.7 (AD7)
31 P4.5 (OCD_SDA)
30 P4.6 (ALE)
29 P4.4 (OCD_SCL)
28 P2.7 (A15/KBI7)
27 P2.6 (A14/KBI6)
26 P2.5 (A13/KBI5)
25 P2.4 (A12/KBI4)
24 P2.3 (A11/KBI3)
23 P2.2 (A10/KBI2)
22 P2.1 (A9/KBI1)
21 P2.0 (A8/KBI0)

4. Block Diagram



5. Special Function Register

5.1. SFR Map

	Page	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8	0 F	P5	CH	CCAP0H	CCAP1H	CCAP2H	CCAP3H	CCAP4H	CCAP5H
F0	0 F	B	--	PCAPWM0	PCAPWM1	PCAPWM2	PCAPWM3	PCAPWM4	PCAPWM5
E8	0 F	P4	CL	CCAP0L	CCAP1L	CCAP2L	CCAP3L	CCAP4L	CCAP5L
E0	0 F	ACC	WDTCR	IFD	IFADRH	IFADRL	IFMT	SCMD	ISPCR
D8	0 F	CCON	CMOD	CCAPM0	CCAPM1	CCAPM2	CCAPM3	CCAPM4	CCAPM5
D0	0 F	PSW	--	--	--	--	KBPATN	KBCON	KBMASK
C8	**0** **F**	**T2CON*** **P6***	T2MOD	RCAP2L	RCAP2H	TL2	TH2	--	--
C0	0 F	XICON	--	--	--	--	ADCON	ADCV	PCON2
B8	0 F	IP0L	SADEN	--	--	--	--	ADCVL	--
B0	0 F	P3	P3M0	P3M1	P4M0	P4M1	P5M0	P5M1	IP0H
A8	0 F	IE	SADDR	--	--	SFRPI*	EIE1	EIP1L	EIP1H
A0	0 F	P2	--	AUXR1	--	--	--	AUXR2	--
98	**0** **1**	**S0CON*** **S1CON***	**S0BUF*** **S1BUF***	**SCFG*** **S1BRT***	--	--	--	--	--
90	0 F	P1	P1M0	P1M1	P0M0	P0M1	P2M0	P2M1	PCON1
88	0 F	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR0	STRETCH
80	0 F	P0	SP	DPL	DPH	SPSTAT	SPCON	SPDAT	PCON0
		0/8	**1/9**	**2/A**	**3/B**	**4/C**	**5/D**	**6/E**	**7/F**

***: User needs to set SFRPI as SFRPI=0x00, or SFRPI=0x01 for SFR page access.**
（MCU will not keep SFRPI value in interrupt. User need to keep SFRPI value in software flow.）

5.2. SFR Bit Assignment

SYMBOL	DESCRIPTION	ADDR	BIT ADDRESS AND SYMBOL								RESET VALUE
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
P0	*Port 0*	*80H*	*P0.7*	*P0.6*	*P0.5*	*P0.4*	*P0.3*	*P0.2*	*P0.1*	*P0.0*	*11111111B*
SP	Stack Pointer	81H									00000111B
DPL	Data Pointer Low	82H									00000000B
DPH	Data Pointer High	83H									00000000B
SPSTAT	SPI Status Register	84H	SPIF	WCOL	--	--	--	--	--	--	00xxxxxxB
SPCON	SPI Control Register	85H	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	00000100B
SPDAT	SPI Data Register	86H									00000000B
PCON0	Power Control 0	87H	SMOD1	SMOD0	GF	POF	GF1	GF0	PD	IDL	00010000B
TCON	*Timer Control*	*88H*	*TF1*	*TR1*	*TF0*	*TR0*	*IE1*	*IT1*	*IE0*	*IT0*	*00000000B*
TMOD	Timer Mode	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00000000B
TL0	Timer Low 0	8AH									00000000B
TL1	Timer Low 1	8BH									00000000B
TH0	Timer High 0	8CH									00000000B
TH1	Timer High 1	8DH									00000000B
AUXR0	Auxiliary Register 0	8EH	P60OC1	P60OC0	P60FD	P34FD	MOVXFD	ADRJ	EXTRAM	--	0000000xB
STRETCH	MOVX Timing Stretch	8FH	EMAI1	--	ALES1	ALES0	RWSH	RWS2	RWS1	RWS0	0x000000B
P1	*Port 1*	*90H*	*P1.7*	*P1.6*	*P1.5*	*P1.4*	*P1.3*	*P1.2*	*P1.1*	*P1.0*	*11111111B*
P1M0	P1 Mode Register 0	91H	P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0	00000000B
P1M1	P1 Mode Register 1	92H	P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0	00000000B
P0M0	P0 Mode Register 0	93H	P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0	00000000B
P0M1	P0 Mode Register 1	94H	P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0	00000000B
P2M0	P2 Mode Register 0	95H	P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0	00000000B
P2M1	P2 Mode Register 1	96H	P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0	00000000B
PCON1	Power Control 1	97H	SWRF	EXRF	BORF	IARF	--	--	--	BOF	0000xxx0B
SCON0	*Serial 0 Control*	*98H*	*SM00 /FE*	*SM10*	*SM20*	*REN0*	*TB80*	*RB80*	*TI0*	*RI0*	*00000000B*
SCON1	*Serial 1 Control*	*98H*	*SM01*	*SM11*	*SM21*	*REN1*	*TB81*	*RB81*	*TI1*	*RI1*	*00000000B*
SBUF0	Serial 0 Buffer	99H									xxxxxxxxB
SBUF1	Serial 1 Buffer	99H									xxxxxxxxB
SCFG		9AH	URTS	SMOD2	URM0X6	S1TR	S1MOD1	S1X12	--	--	000000xxB
S1BRT	S1 Baud-Rate Timer	9AH	S1BRT.7	S1BRT.6	S1BRT.5	S1BRT.4	S1BRT.3	S1BRT.2	S1BRT.1	S1BRT.0	00000000B
P2	*Port 2*	*A0H*	*P2.7*	*P2.6*	*P2.5*	*P2.4*	*P2.3*	*P2.2*	*P2.1*	*P2.0*	*11111111B*
AUXR1	Auxiliary Register 1	A2H	P4KBI	P4PCA	P5SPI	P4S1	--	--	--	DPS	0000xxx0B
AUXR2	Auxiliary Register 2	A6H	T0X12	T1X12	--	--	--	--	T1CKOE	T0CKOE	00xxxx00B
IE	Interrupt Enable	A8H	EA	--	ET2	ES0	ET1	EX1	ET0	EX0	0x000000B
SADDR	Slave Address	A9H									00000000B
SFRPI	SFR Page Index	ACH	--	--	--	--	IDX3	IDX2	IDX1	IDX0	xxxx0000B
EIE1	Extended INT Enable 1	ADH	--	--	EKB	ES1	EBD	EPCA	EADC	ESPI	xx000000B
EIP1L	Ext. INT Priority 1 Low	AEH	--	--	PKBL	PS1L	PBDL	PPCAL	PADCL	PSPIL	xx000000B
EIP1H	Ext. INT Priority 1 High	AFH	--	--	PKBH	PS1H	PBDH	PPCAH	PADCH	PSPIH	xx000000B
P3	*Port 3*	*B0H*	*P3.7*	*P3.6*	*P3.5*	*P3.4*	*P3.3*	*P3.2*	*P3.1*	*P3.0*	*11111111B*
P3M0	P3 Mode Register 0	B1H	P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0	00000000B
P3M1	P3 Mode Register 1	B2H	P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	00000000B
P4M0	P4 Mode Register 0	B3H	--	P4M0.6	P4M0.5	P4M0.4	P4M0.3	P4M0.2	P4M0.1	P4M0.0	x0000000B
P4M1	P4 Mode Register 1	B4H	--	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0	x0000000B
P5M0	P5 Mode Register 0	B5H	--	--	--	--	P5M0.3	P5M0.2	P5M0.1	P5M0.0	xxxx0000B
P5M1	P5 Mode Register 1	B6H	--	--	--	--	P5M1.3	P5M1.2	P5M1.1	P5M1.0	xxxx0000B
IP0H	Interrupt Priority 0 High	B7H	PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	00000000B
IP0L	*Interrupt Priority Low*	*B8H*	***PX3L***	***PX2L***	*PT2L*	*PSL*	*PT1L*	*PX1L*	*PT0L*	*PX0L*	*00000000B*
SADEN	Slave Address Mask	B9H									00000000B
ADCVL	ADC result Low	BEH	--	--	--	--	--	--	ADCV.1	ADCV.0	xx001010B
XICON	External INT Control	C0H	**IT3H**	EX3	IE3	IT3	**IT2H**	EX2	IE2	IT2	00000000B
ADCON	ADC Control	C5H	ADCEN	SPEED1	SPEED0	ADCI	ADCS	CHS2	CHS1	CHS0	00000000B
ADCV	ADC result	C6H	ADCV.9	ADCV.8	ADCV.7	ADCV.6	ADCV.5	ADCV.4	ADCV.3	ADCV.2	00000000B
PCON2	Clock Control 0	C7H	OSCDR	--	--	--	--	SCKS2	SCKS1	SCKS0	xxxxx000B
T2CON	*Timer 2 Control*	*C8H*	*TF2*	*EXF2*	*RCLK*	*TCLK*	*EXEN2*	*TR2*	*C/T2*	*CP/RL*	*00000000B*
P6	*Port 6*	*C8H*	*--*	*--*	*--*	*--*	*--*	*--*	*P6.1*	*P6.0*	*xxxxxx11B*
T2MOD	Timer2 mode	C9H	--	--	--	**T2X12**	--	--	T2OE	DCEN	xxx0xx00B
RCAP2L	Timer2 Capture Low	CAH									00000000B
RCAP2H	Timer2 Capture High	CBH									00000000B
TL2	Timer Low 2	CCH									00000000B
TH2	Timer High 2	CDH									00000000B
PSW	*Program Status Word*	*D0H*	*CY*	*AC*	*F0*	*RS1*	*RS0*	*OV*	*F1*	*P*	*00000000B*
KBPATN	Keypad Pattern	D5H									11111111B
KBCON	Keypad Control	D6H							PATNS	KBIF	xxxxxx00B
KBMASK	Keypad Int. Mask	D7H									00000000B

CCON	*PCA Control Reg.*	*D8H*	*CF*	*CR*	*CCF5*	*CCF4*	*CCF3*	*CCF2*	*CCF1*	*CCF0*	*00000000B*
CMOD	PCA Mode Reg.	D9H	CIDL	FEOV	--	--	--	CPS1	CPS0	ECF	00xxx000B
CCAPM0	PCA Module0 Mode	DAH	--	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x0000000B
CCAPM1	PCA Module1 Mode	DBH	--	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x0000000B
CCAPM2	PCA Module2 Mode	DCH	--	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x0000000B
CCAPM3	PCA Module3 Mode	DDH	--	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x0000000B
CCAPM4	PCA Module4 Mode	DEH	--	ECOM4	CAPP4	CAPN4	MAT4	TOG4	PWM4	ECCF4	x0000000B
CCAPM5	PCA Module5 Mode	DFH	--	ECOM5	CAPP5	CAPN5	MAT5	TOG5	PWM5	ECCF5	x0000000B
ACC	*Accumulator*	*E0H*	*ACC.7*	*ACC.6*	*ACC.5*	*ACC.4*	*ACC.3*	*ACC.2*	*ACC.1*	*ACC.0*	*00000000B*
WDTCR	Watch-dog-timer Control register	E1H	WRF	--	ENW	CLW	WIDL	PS2	PS1	PS0	0x000000B
IFD	ISP Flash data	E2H									11111111B
IFADRH	ISP Flash address High	E3H									00000000B
IFADRL	ISP Flash Address Low	E4H									00000000B
IFMT	ISP Mode Table	E5H	--	--	--	MS4	MS3	MS2	MS1	MS0	xxxx0000B
IAPLB	IAP Low Boundary	Note 1	IAPLB6	IAPLB5	IAPLB4	IAPLB3	IAPLB2	IAPLB1	IAPLB0	--	IFR7
AUXRA	Auxiliary Register A	Note 1	DBOD	BORE	OCDE	ILRCOE	XTALE	IHRCOE	OSCS1	OSCS0	00100100B
AUXRB	Auxiliary Register A	Note 1	--	--	--	IAPO	LPM3	LPM2	--	LPM0	xxx000x0B
SCMD	ISP Serial Command	E6H									xxxxxxxxB
ISPCR	ISP Control Register	E7H	ISPEN	BS	SRST	CFAIL	--	PCKS2	PCKS1	PCKS0	0000x000B
P4	*Port 4*	*E8H*	--	*P4.6*	*P4.5*	*P4.4*	*P4.3*	*P4.2*	*P4.1*	*P4.0*	*x1111111B*
CL	PCA base timer Low	E9H									00000000B
CCAP0L	PCA module0 Capture Low	EAH									00000000B
CCAP1L	PCA module1 capture Low	EBH									00000000B
CCAP2L	PCA module2 capture Low	ECH									00000000B
CCAP3L	PCA module3 capture Low	EDH									00000000B
CCAP4L	PCA module4 capture Low	EEH									00000000B
CCAP5L	PCA module5 capture Low	EFH									00000000B
B	*B Register*	*F0H*	*F7H*	*F6H*	*F5H*	*F4H*	*F3H*	*F2H*	*F1H*	*F0H*	*00000000B*
PCAPWM0	PCA PWM0 Mode	F2H	**P0RS1**	**P0RS0**	**P0PS2**	**P0PS1**	**P0PS0**	**P0INV**	EPC0H	EPC0L	00000000B
PCAPWM1	PCA PWM1 Mode	F3H	**P1RS1**	**P1RS0**	**P1PS2**	**P1PS1**	**P1PS0**	**P1INV**	EPC1H	EPC1L	00000000B
PCAPWM2	PCA PWM2 Mode	F4H	**P2RS1**	**P2RS0**	**P2PS2**	**P2PS1**	**P2PS0**	**P2INV**	EPC2H	EPC2L	00000000B
PCAPWM3	PCA PWM3 Mode	F5H	**P3RS1**	**P3RS0**	**P3PS2**	**P3PS1**	**P3PS0**	**P3INV**	EPC3H	EPC3L	00000000B
PCAPWM4	PCA PWM4 Mode	F6H	**P4RS1**	**P4RS0**	**P4PS2**	**P4PS1**	**P4PS0**	**P4INV**	EPC4H	EPC4L	00000000B
PCAPWM5	PCA PWM5 Mode	F7H	**P5RS1**	**P5RS0**	**P5PS2**	**P5PS1**	**P5PS0**	**P5INV**	EPC5H	EPC5L	00000000B
P5	*Port 5*	*F8H*					*P5.3*	*P5.2*	*P5.1*	*P5.0*	*xxxx1111B*
CH	PCA base timer High	F9H									00000000B
CCAP0H	PCA Module0 capture High	FAH									00000000B
CCAP1H	PCA Module1 capture High	FBH									00000000B
CCAP2H	PCA Module2 capture High	FCH									00000000B
CCAP3H	PCA Module3 capture High	FDH									00000000B
CCAP4H	PCA Module4 capture High	FEH									00000000B
CCAP5H	PCA Module5 capture High	FFH									00000000B

Note1: The registers are addressed by IFMT and SCMD. Please refer the IFMT register description for more detail information.

6. Memory Organization

Like all 80C51 devices, the MG82Fx564 has separate address spaces for program and data memory. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by the 8-bit CPU.

Program memory (ROM) can only be read, not written to. There can be up to 64K bytes of program memory. In the MG82Fx564, all the program memory are on-chip Flash memory, and without the capability of accessing external program memory because of no External Access Enable (/EA) and Program Store Enable (/PSEN) signals designed.

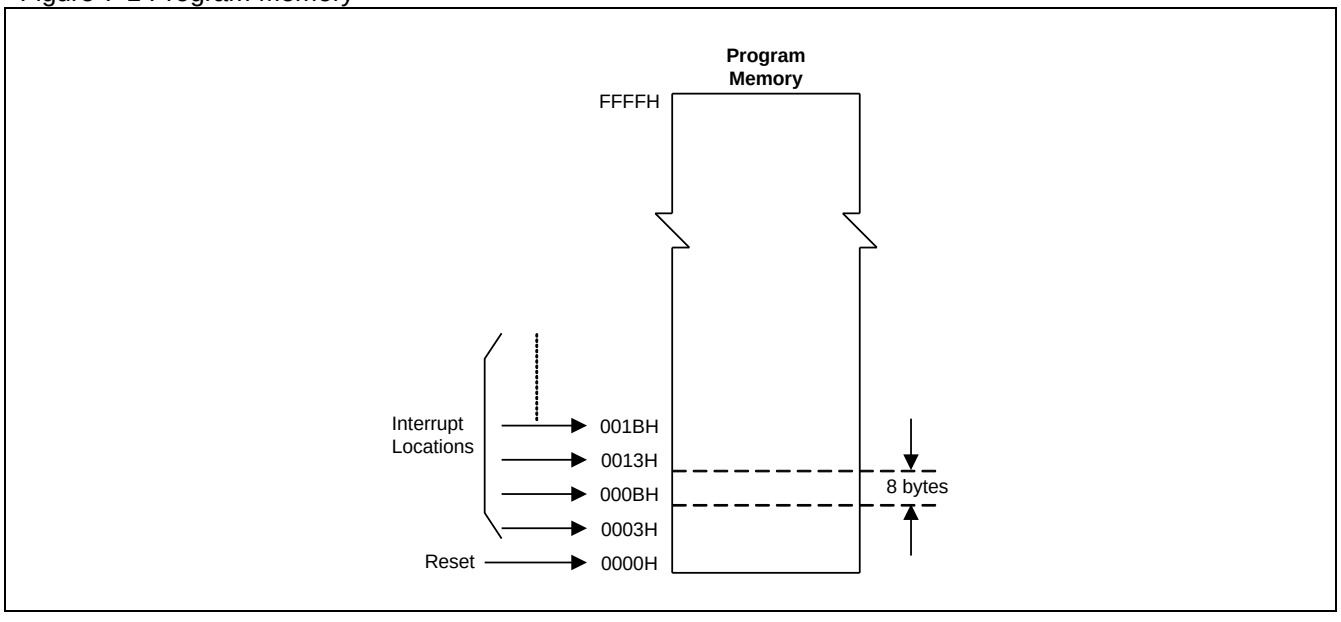
Data memory occupies a separate address space from program memory. In the MG82Fx564, there are 256 bytes of internal scratch-pad RAM and 1024 bytes of on-chip expanded RAM (XRAM).

6.1. On-Chip Program Flash

Program memory is the memory which stores the program codes for the CPU to execute, as shown in Figure 7-1. After reset, the CPU begins execution from location 0000H, where should be the starting of the user's application code. To service the interrupts, the interrupt service locations (called interrupt vectors) should be located in the program memory. Each interrupt is assigned a fixed location in the program memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose program memory.

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

Figure 7-1 Program Memory



6.2. On-Chip Data RAM

Figure 7-2 shows the internal and external data memory spaces available to the MG82Fx564 user. Internal data memory can be divided into three blocks, which are generally referred to as the lower 128 bytes of RAM, the upper 128 bytes of RAM, and the 128 bytes of SFR space. Internal data memory addresses are always 8-bit wide, which implies an address space of only 256 bytes. Direct addresses higher than 7FH access the SFR space; and indirect addresses higher than 7FH access the upper 128 bytes of RAM. Thus the SFR space and the upper 128 bytes of RAM occupy the same block of addresses, 80H through FFH, although they are physically separate entities.

The lower 128 bytes of RAM are present in all 80C51 devices as mapped in Figure 7-3. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing. The next 16 bytes above the register banks form a block of bit-addressable memory space. The 80C51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing while the Upper 128 can only be accessed by indirect addressing.

Figure 7-4 gives a brief look at the Special Function Register (SFR) space. SFRs include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 0H or 8H.

To access the external data memory, the EXTRAM bit should be set to 1. Accesses to external data memory can use either a 16-bit address (using 'MOVX @DPTR') or an 8-bit address (using 'MOVX @Ri'), as described below.

Accessing by an 8-bit address

8-bit addresses are often used in conjunction with one or more other I/O lines to page the RAM. If an 8-bit address is being used, the contents of the Port 2 SFR remain at the Port 2 pins throughout the external memory cycle. This will facilitate paging access. Figure 7-5 shows an example of a hardware configuration for accessing up to 2K bytes of external RAM. In multiplexed mode, Port 0 serves as a multiplexed address/data bus to the RAM, and 3 lines of Port 2 are being used to page the RAM. The CPU generates nRD and nWR (alternate functions of P3.7 and P3.6) to strobe the memory. Of course, the user may use any other I/O lines instead of P2 to page the RAM.

Accessing by a 16-bit address

16-bit addresses are often used to access up to 64k bytes of external data memory. Figure 7-6 shows the hardware configuration for accessing 64K bytes of external RAM. Whenever a 16-bit address is used, in addition to the functioning of P0, nRD and nWR, the high byte of the address comes out on Port 2 and it is held during the read or write cycle.

In multiplexed case, the low byte of the address is time-multiplexed with the data byte on Port 0. ALE (Address Latch Enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before nWR is activated, and remains there until after nWR is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated. During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding.

To access the on-chip expanded RAM (XRAM), the EXTRAM bit should be cleared to 0. Refer to Figure 7-2, the 1024 bytes of XRAM (0000H to 03FFH) are indirectly accessed by move external instruction, MOVX. An access to XRAM will have not any outputting of address, address latch enable and read/write strobe. That means P0, P2, P4.6 (ALE), P3.6 (nWR) and P3.7 (nRD) will keep unchanged during access of on-chip XRAM.

Figure 7-2 Data Memory

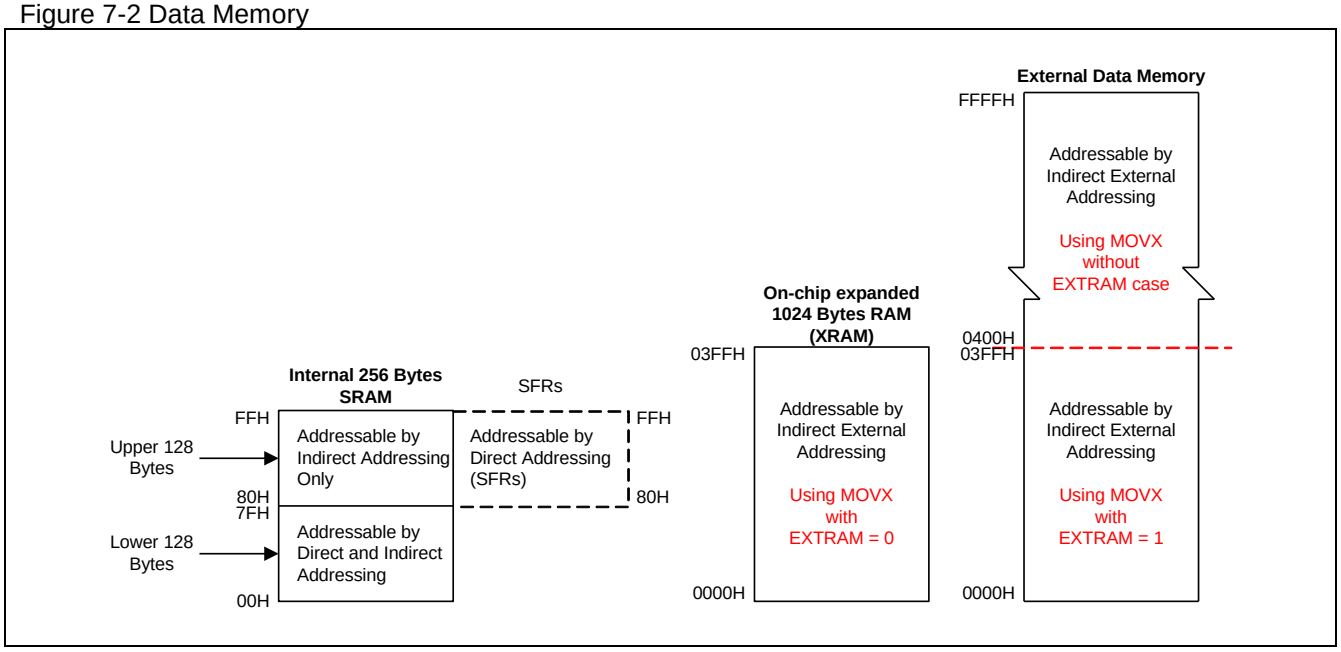


Figure 7-3 Lower 128 Bytes of Internal RAM

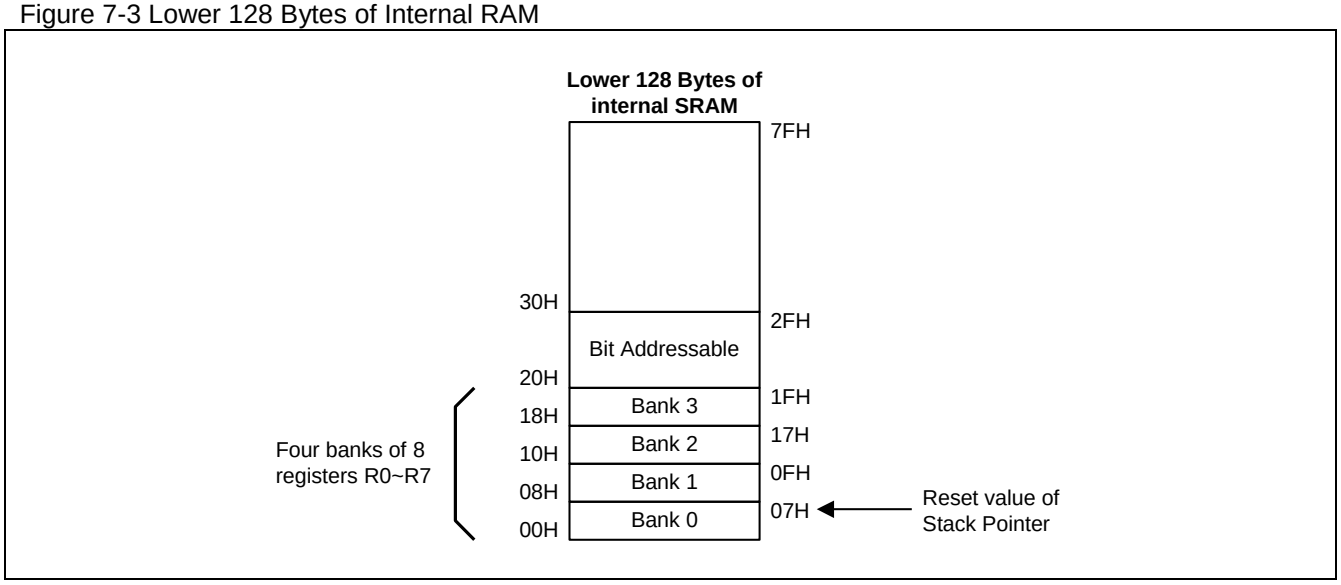


Figure 7-4 SFR Space

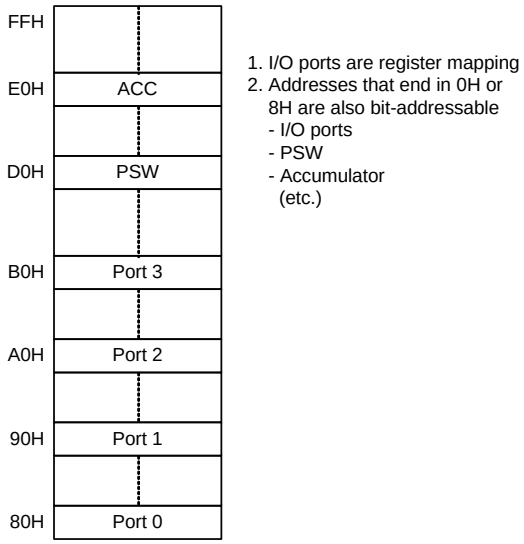
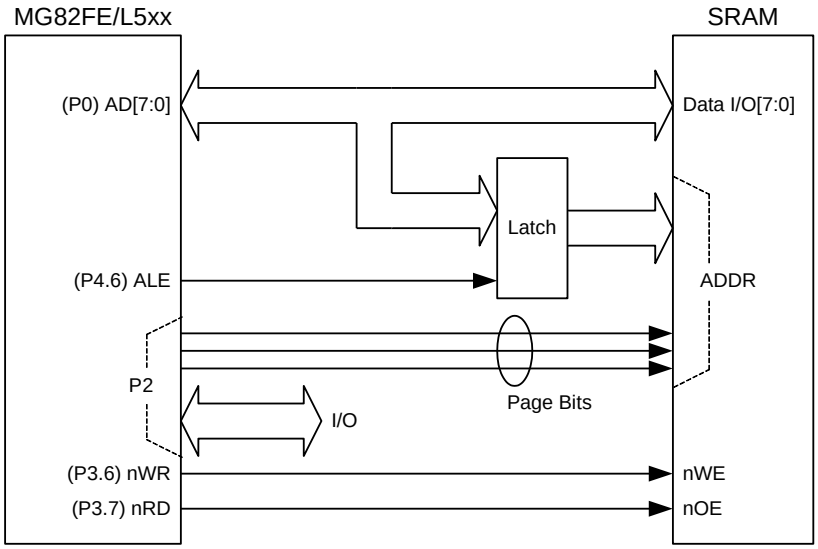


Figure 7-5 External RAM Accessing by an 8-Bit Address (Using 'MOVX @ Ri' and Page Bits)



Note that in this case, the other bits of P2 are available as general I/O pins.

Figure 7-6 External RAM Accessing by a 16-Bit Address (Using 'MOVX @ DPTR')

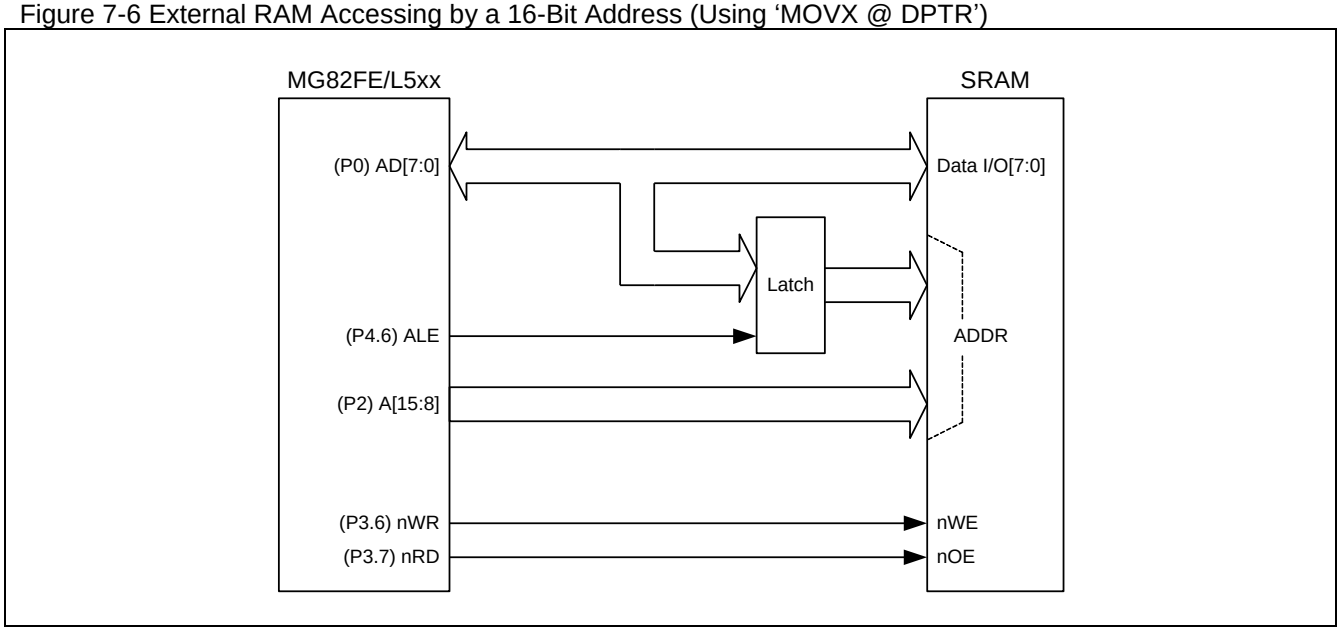
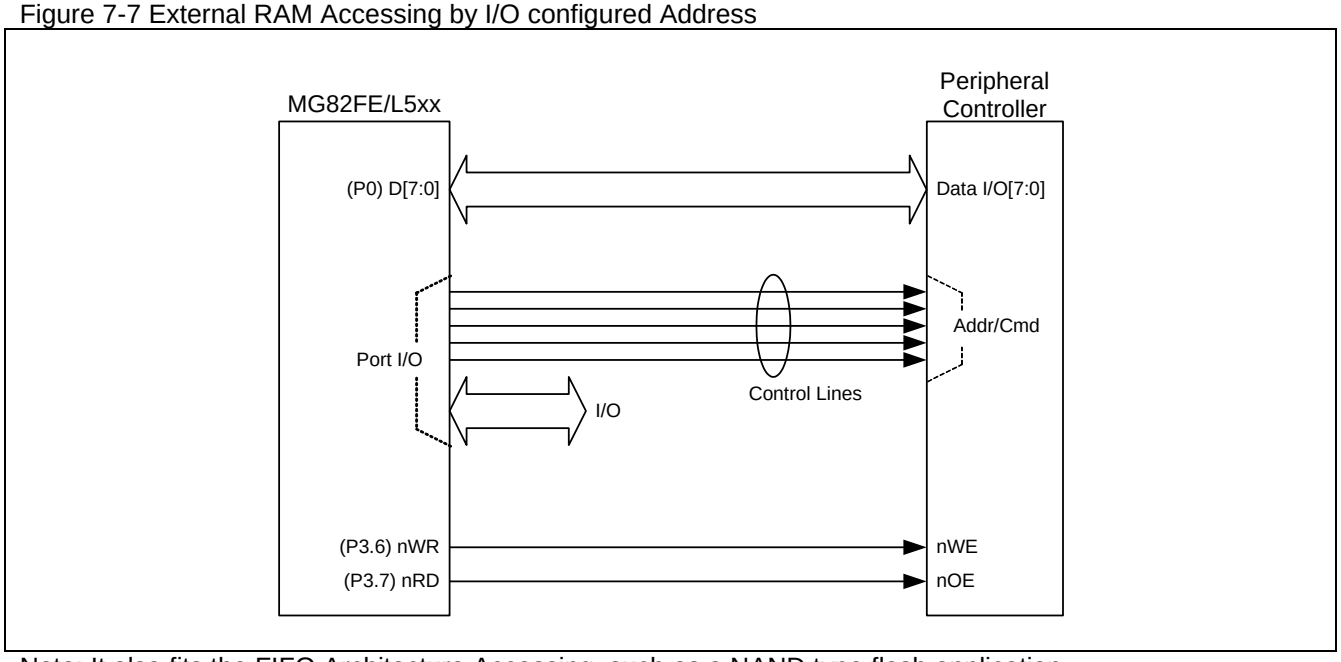


Figure 7-7 External RAM Accessing by I/O configured Address



Note: It also fits the FIFO Architecture Accessing, such as a NAND type flash application.

6.3. On-chip expanded RAM (XRAM)

To access the on-chip expanded RAM (XRAM), refer to Figure 7-2, the 1024 bytes of XRAM (0000H to 03FFH) are indirectly accessed by move external instruction, “MOVX @Ri” and “MOVX @DPTR”. 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 MG82Fx564 hardware can access them correctly.

6.4. External Data Memory access

AUXR0: Auxiliary Register 0

SFR Page = All

SFR Address = 0x8E RESET = 0000-0X0X

7	6	5	4	3	2	1	0
P60OC1	P60OC0	P60FD	P34FD	MOVXFD	ADRJ	EXTRAM	--
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R**

Bit 3: MOVXFD, Fast Driving enabled for MOVX output signals.
0: MOVX output signals with default driving.
1: MOVX output signals with fast driving. If there is an off-chip memory access, MOVX@DPTR or MOVX@Ri, the MOVX output signals require fast driving for stretched ALE/RD/WR pulse frequency more than 12MHz @5V or 6MHz @3.3V.

Bit 1: EXTRAM, External data RAM enable.
0: Enable on-chip expanded data RAM (XRAM 1024 bytes)
1: Disable on-chip expanded data RAM.

Stretch: MOVX Stretch Register

SFR Page = All

SFR Address = 0x8F RESET = 0X00-0000

7	6	5	4	3	2	1	0
EMAI1	--	ALES1	ALES0	RWSH	RWS2	RWS1	RWS0
R/W	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: EMAI1, EMAI1 configures the External data Memory Access Interface mode as following:
0: Multiplexed address/data.
1: No Address phase access

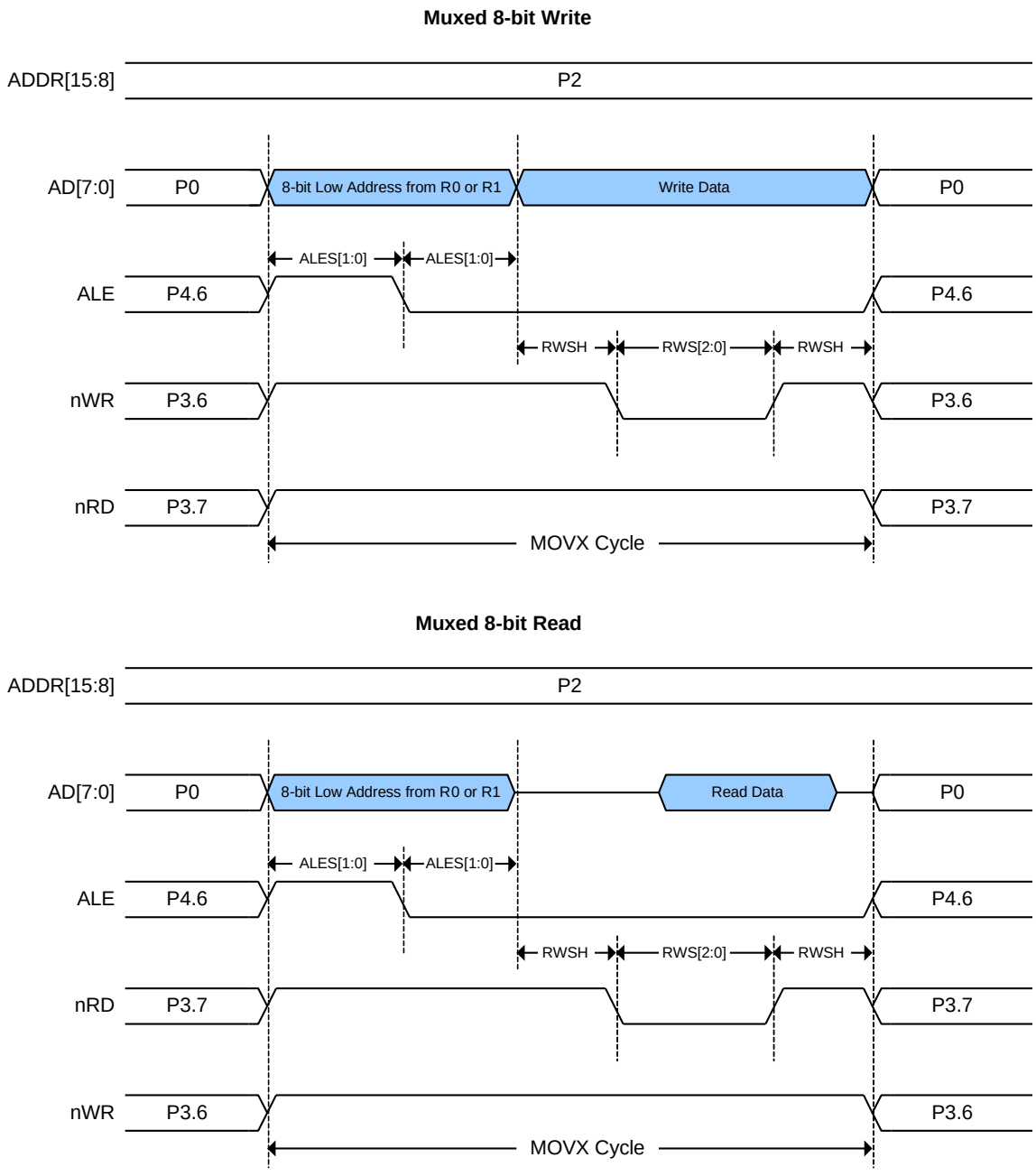
Bit 6: Reserved. Software must write “0” on this bit when STRETCH is written.

Bit 5~4: ALES[1:0], EMAI ALE pulse width select bits. It only has effect when EMAI in Multiplexed mode.
00: ALE high and ALE low pulse width = 1 SYSCLK cycle.
01: ALE high and ALE low pulse width = 2 SYSCLK cycle.
10: ALE high and ALE low pulse width = 3 SYSCLK cycle.
11: ALE high and ALE low pulse width = 4 SYSCLK cycle.

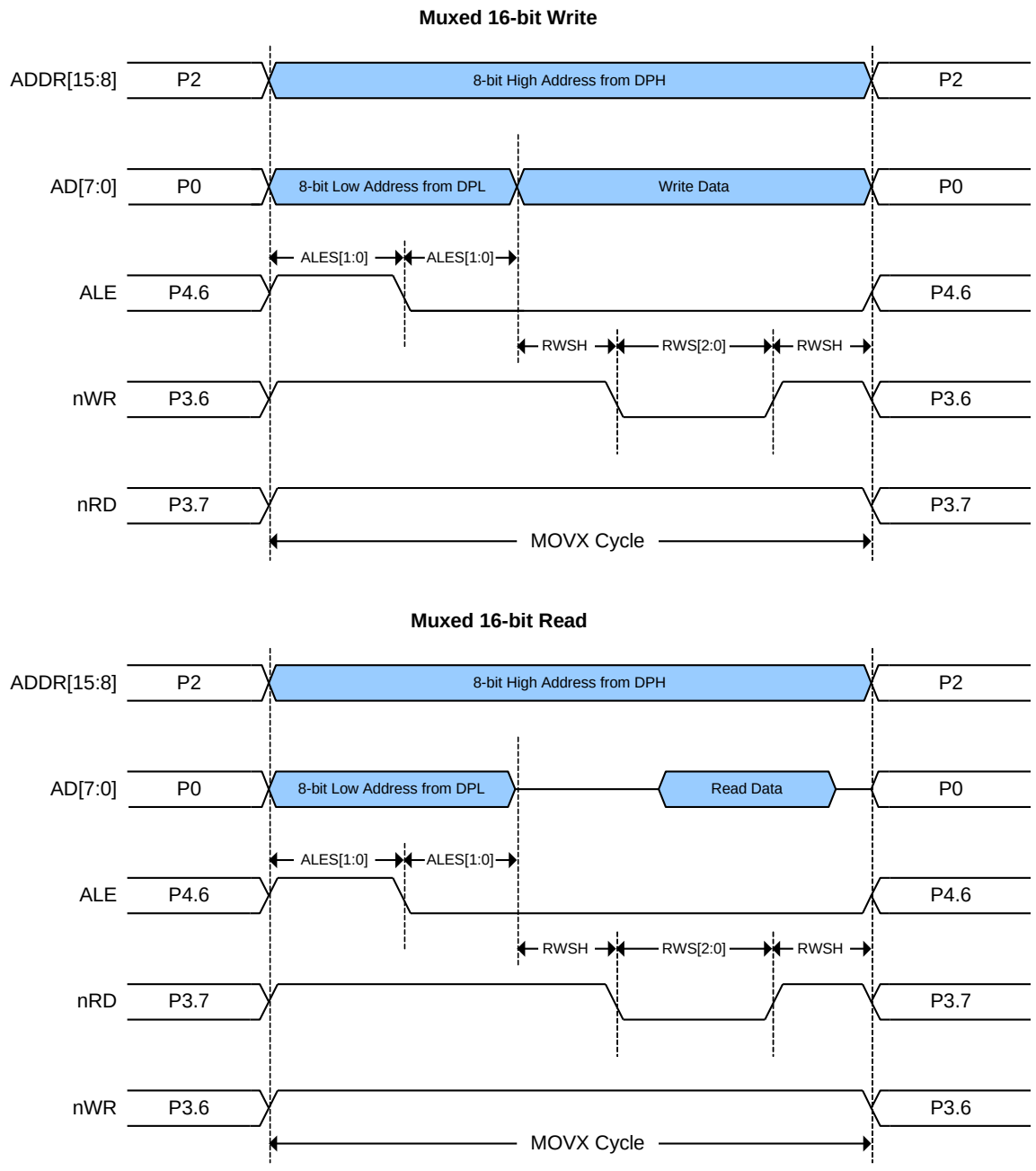
Bit 3: RWSH, EMAI Read/Write pulse Setup/Hold time control.
0: /RD and /WR command Setup/Hold Time = 1 SYSCLK cycle.
1: /RD and /WR command Setup/Hold Time = 2 SYSCLK cycle.

Bit 2~0: RWS[2:0], EMAI Read/Write command pulse width select bits.
000: /RD and /WR pulse width = 1 SYSCLK cycle.
001: /RD and /WR pulse width = 2 SYSCLK cycle.
010: /RD and /WR pulse width = 3 SYSCLK cycle.
011: /RD and /WR pulse width = 4 SYSCLK cycle.
100: /RD and /WR pulse width = 5 SYSCLK cycle.
101: /RD and /WR pulse width = 6 SYSCLK cycle.
110: /RD and /WR pulse width = 7 SYSCLK cycle.
111: /RD and /WR pulse width = 8 SYSCLK cycle.

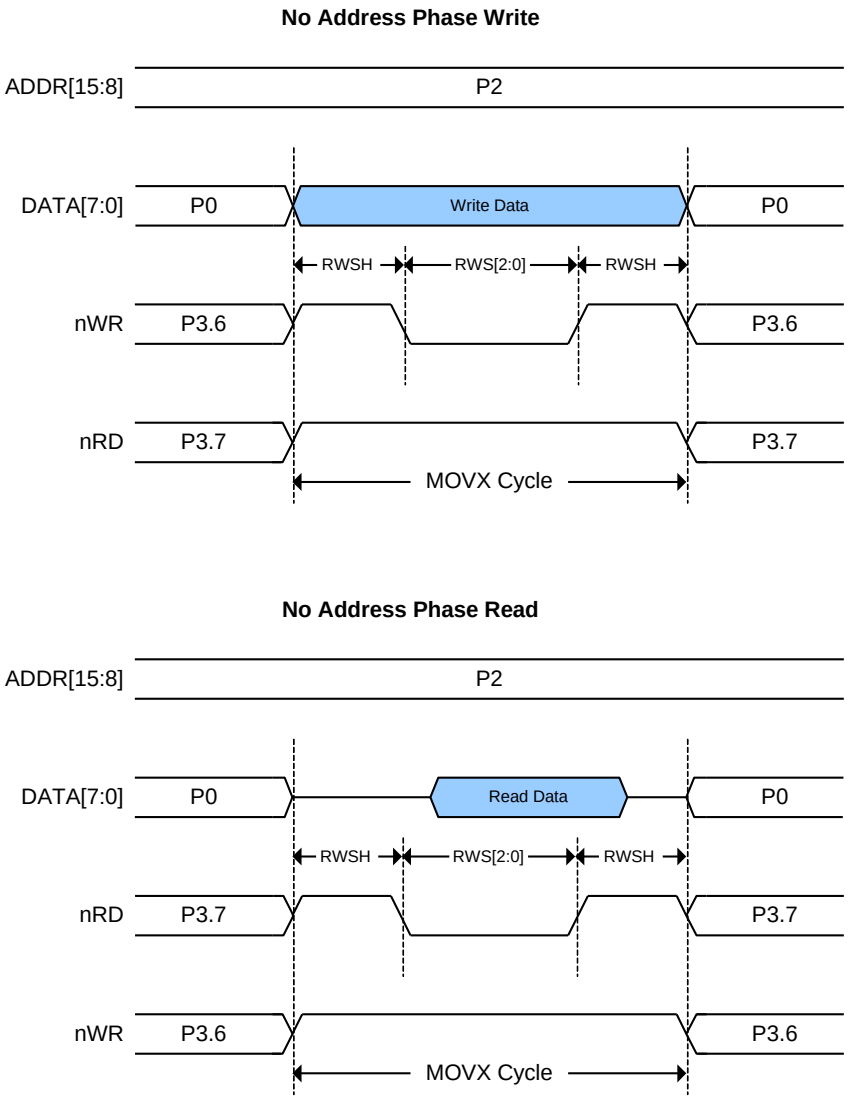
6.4.1. Multiplexed Mode for 8-bit MOVX



6.4.2. Multiplexed Mode for 16-bit MOVX



6.4.3. No Address Phase Mode for MOVX



7. 8051 CPU Description

7.1. CPU Register

PSW: Program Status Word

SFR Page = All

SFR Address = 0xD0 RESET = 0000-0000

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	F1	P
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

CY: Carry bit.
AC: Auxiliary carry bit.
F0: General purpose flag 0.
RS1: Register bank select bit 1.
RS0: Register bank select bit 0.
OV: Overflow flag.
F1: General purpose flag 1.
P: Parity bit.

The program status word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown above, resides in the SFR space. It contains the Carry bit, the Auxiliary Carry(for BCD operation), the two register bank select bits, the Overflow flag, a Parity bit and two user-definable status flags.

The Carry bit, other than serving the function of a Carry bit in arithmetic operations, also serves as the "Accumulator" for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in the on-chip-data-RAM section. A number of instructions refer to these RAM locations as R0 through R7.

The Parity bit reflects the number of 1s in the Accumulator. P=1 if the Accumulator contains an odd number of 1s and otherwise P=0.

SP: Stack Pointer

SFR Page = All

SFR Address = 0x81 RESET = 0000-0111

7	6	5	4	3	2	1	0
SP[7]	SP[6]	SP[5]	SP[4]	SP[3]	SP[2]	SP[1]	SP[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

DPL: Data Pointer Low

SFR Page = All

SFR Address = 0x82 RESET = 0000-0000

7	6	5	4	3	2	1	0
DPL[7]	DPL[6]	DPL[5]	DPL[4]	DPL[3]	DPL[2]	DPL[1]	DPL[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

DPH: Data Pointer High

SFR Page = All

SFR Address = 0x83 RESET = 0000-0000

7	6	5	4	3	2	1	0
DPH[7]	DPH[6]	DPH[5]	DPH[4]	DPH[3]	DPH[2]	DPH[1]	DPH[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

B: B Register

SFR Page = All

SFR Address = 0xF0 RESET = 0000-0000

7	6	5	4	3	2	1	0
B[7]	B[6]	B[5]	B[4]	B[3]	B[2]	B[1]	B[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

7.2. CPU Timing

The MG82Fx564 is a single-chip microcontroller based on a high performance 1-T architecture 80C51 CPU that has an 8051 compatible instruction set, and executes instructions in 1~7 clock cycles (about 6~7 times the rate of a standard 8051 device). It employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. The instruction timing is different than that of the standard 8051.

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the 1T-80C51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles. For more detailed information about the 1T-80C51 instructions, please refer section “Instruction Set” which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

7.3. CPU Addressing Mode

Direct Addressing (DIR)
In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal data RAM and SFRs can be direct addressed.

Indirect Addressing (IND)
In indirect addressing the instruction specified a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

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

Register Instruction (REG)
The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the op-code of the instruction. Instructions that access the registers this way are code efficient because this mode eliminates the need of an extra address byte. When such instruction is executed, one of the eight registers in the selected bank is accessed.

Register-Specific Instruction
Some instructions are specific to a certain register. For example, some instructions always operate on the accumulator or data pointer, etc. No address byte is needed for such instructions. The op-code itself does it.

Immediate Constant (IMM)
The value of a constant can follow the op-code in the program memory.

Index Addressing
Only program memory can be accessed with indexed addressing and it can only be read. This addressing mode is intended for reading look-up tables in program memory. A 16-bit base register (either DPTR or PC) points to the base of the table, and the accumulator is set up with the table entry number. Another type of indexed addressing is used in the conditional jump instruction.
In conditional jump, the destination address is computed as the sum of the base pointer and the accumulator.

7.4. Declaration Identifiers in a C51-Compiler

The declaration identifiers in a C51-compiler for the various MG82Fx564memory spaces are as follows:

data
128 bytes of internal data memory space (00h~7Fh); accessed via direct or indirect addressing, using instructions other than MOVX and MOVC. All or part of the Stack may be in this area.

idata
Indirect data; 256 bytes of internal data memory space (00h~FFh) accessed via indirect addressing using instructions other than MOVX and MOVC. All or part of the Stack may be in this area. This area includes the data area and the 128 bytes immediately above it.

sfr
Special Function Registers; CPU registers and peripheral control/status registers, accessible only via direct addressing.

xdata
External data or on-chip expanded RAM (XRAM); duplicates the classic 80C51 64KB memory space addressed via the “MOVX @DPTR” instruction. The MG82Fx564 has 1024 bytes of on-chip xdata memory.

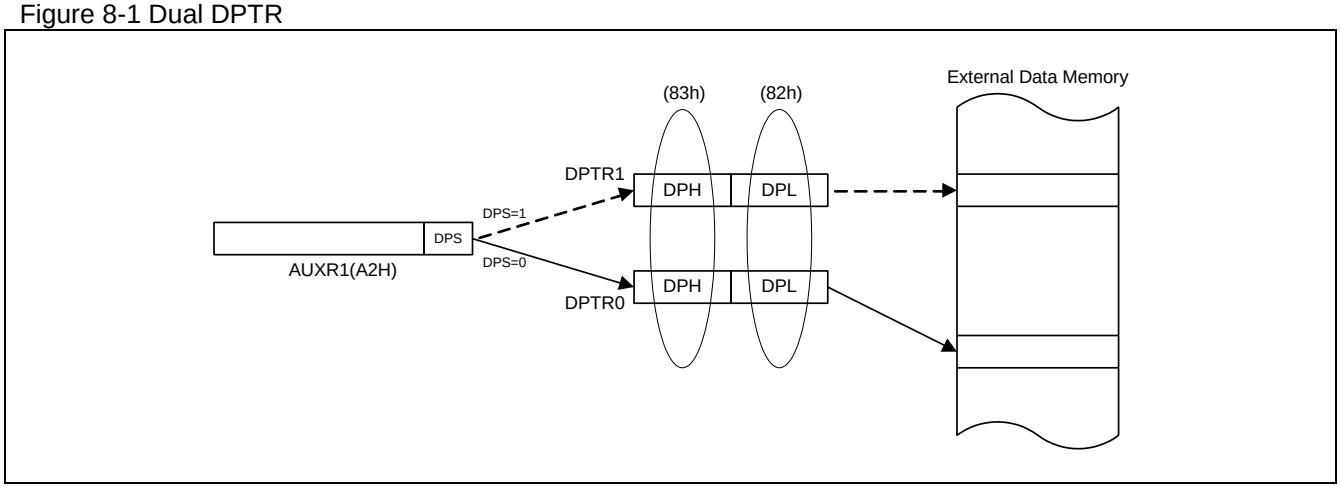
pdata
Paged (256 bytes) external data or on-chip expanded RAM; duplicates the classic 80C51 256 bytes memory space addressed via the “MOVX @Ri” instruction. The MG82Fx564 has 256 bytes of on-chip pdata memory which is shared with on-chip xdata memory.

code
64K bytes of program memory space; accessed as part of program execution and via the “MOVC @A+DTPR” instruction. The MG82Fx564 has 64K bytes of on-chip code memory.

8. Dual Data Pointer Register (DPTR)

The dual DPTR structure as shown in Figure 8-1 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 (AUXR1.0) that allows the program code to switch between them.

Figure 8-1 Dual DPTR



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
```

Note: User should add a NOP when you access internal and external XRAM switching or access XRAM over 1KB address.

AUXR1: Auxiliary Control Register 1

SFR Page = All

SFR Address = 0xA2 RESET = 0000-XXX0

7	6	5	4	3	2	1	0
P4KBI	P4PCA	P5SPI	P4S1	--	--	--	DPS
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**

Bit 0: DPTR select bit, used to switch between DPTR0 and DPTR1.
0: Select DPTR0.
1: Select DPTR1.

DPS	Selected DPTR
0	DPTR0
1	DPTR1

9. Configurable I/O Ports

The MG82Fx564 has following I/O ports: P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.6 and P5.0~P5.3. ALE pin has a swapped function for P4.6. If select internal oscillator as system clock input, XTAL2 and XTAL1 are configured to Port 6.0 and Port 6.1. The exact number of I/O pins available depends upon the package types. See Table 9-1.

Table 9-1 Number of I/O Pins Available

Package Type	I/O Pins	Number of I/O ports
44-pin PQFP	P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.6, XTAL2(P6.0), XTAL1(P6.1)	39 or 41 (INTOSC enabled)
48-pin LQFP	P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.6, P5.0~P5.3,XTAL2(P6.0), XTAL1(P6.1)	43 or 45 (INTOSC enabled)

9.1. IO Structure

Except P6.0 and P6.1, all I/O port pins can be configured to one of four operating modes. These are: quasi-bidirectional (standard 8051 I/O port), push-pull output, input-only (high-impedance input) and open-drain output. P6.0 and P6.1 are only one I/O mode for quasi-bidirectional ports.

Followings describe the configuration of the four types I/O mode.

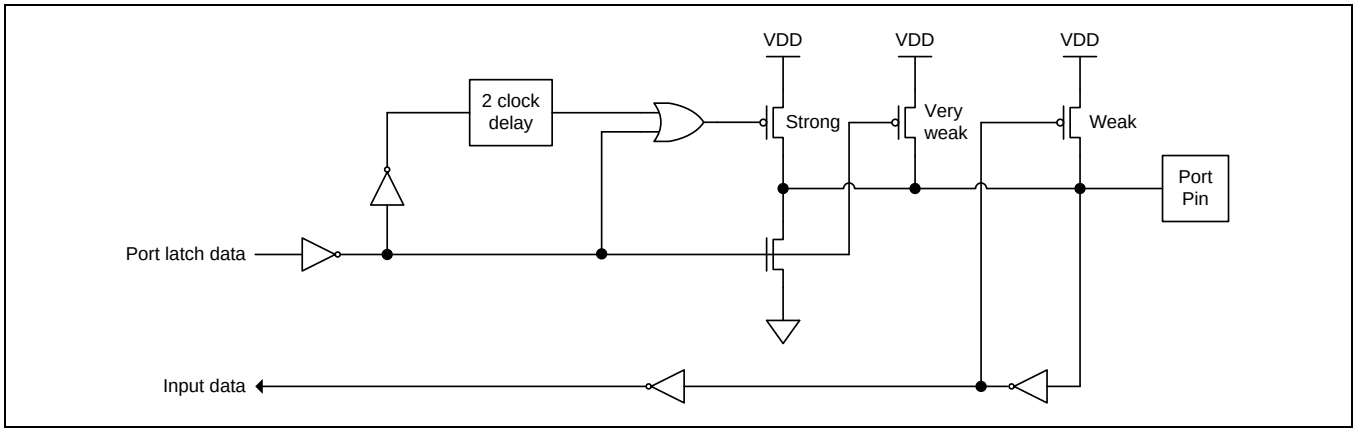
9.1.1. Quasi-Bidirectional IO Structure

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 one CPU clocks, quickly pulling the port pin high.

The quasi-bidirectional port configuration is shown in Figure 9-1.

Figure 9-1 Quasi-Bidirectional I/O

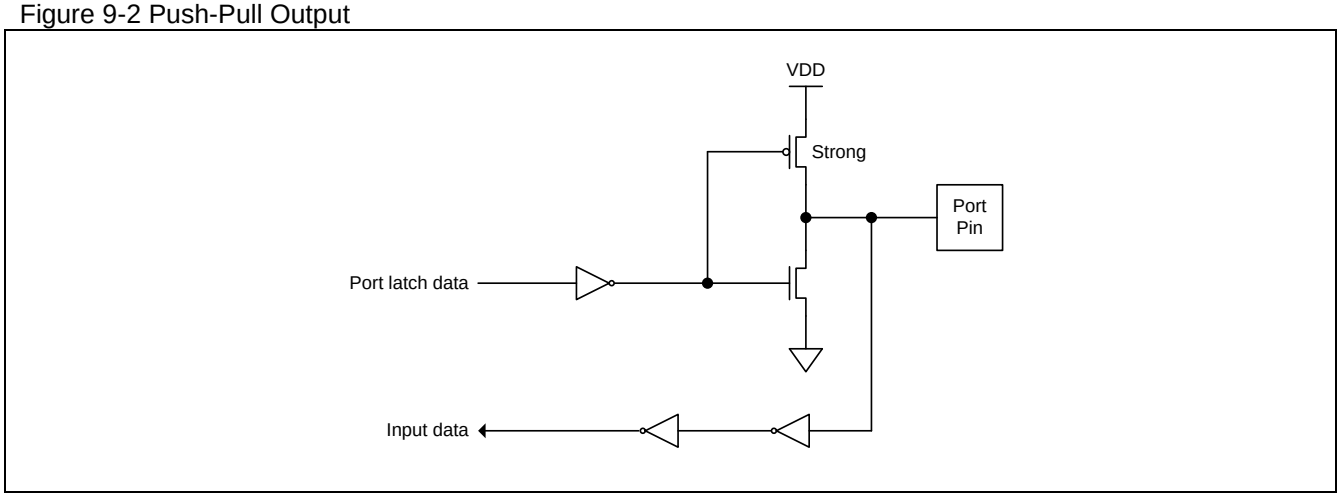


9.1.2. Push-Pull Output Structure

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.

The push-pull port configuration is shown in Figure 9-2.

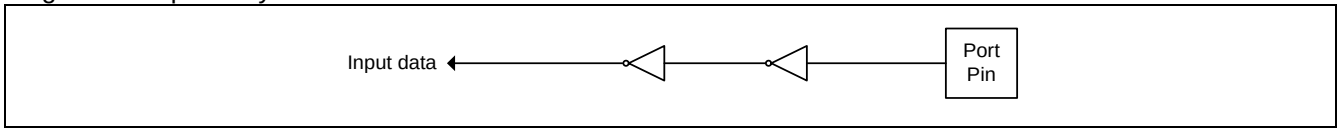
Figure 9-2 Push-Pull Output



9.1.3. Input-Only (High Impedance Input) Structure

The input-only configuration is a input without any pull-up resistors on the pin, as shown in Figure 9-3.

Figure 9-3 Input-Only

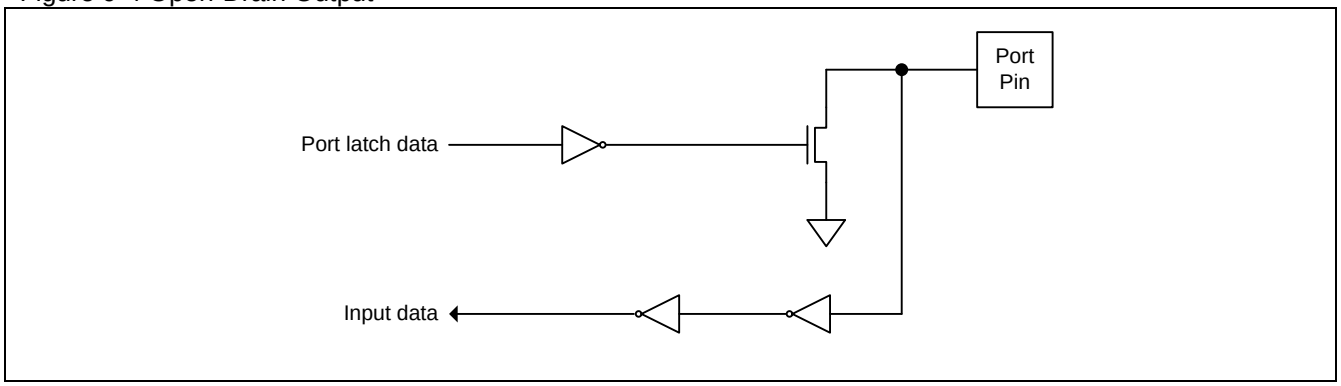


9.1.4. Open-Drain Output Structure

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.

The open-drain port configuration is shown in Figure 9-4.

Figure 9-4 Open-Drain Output



9.2. I/O Port Register

All I/O port pins on the MG82Fx564 may be individually and independently configured by software to one of four types on a bit-by-bit basis, as shown in Table 9-2. Two mode registers for each port select the output type for each port pin.

Table 9-2 Port Configuration Settings

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

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

9.2.1. Port 0 Register

P0: Port 0 Register

SFR Page = All

SFR Address = 0x80 RESET = 1111-1111

7	6	5	4	3	2	1	0
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7~0: P0.7~P0.0 could be set/cleared by CPU.

P0M0: Port 0 Mode Register 0

SFR Address = 0x93

SFR Page = All RESET = 0000-0000

7	6	5	4	3	2	1	0
P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P0M1: Port 0 Mode Register 1

SFR Page = All

SFR Address = 0x94 RESET = 0000-0000

7	6	5	4	3	2	1	0
P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

9.2.2. Port 1 Register

P1: Port 1 Register

SFR Page = All

SFR Address = 0x90 RESET = 1111-1111

7	6	5	4	3	2	1	0
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7~0: P1.7~P1.0 could be only set/cleared by CPU.

P1M0: Port 1 Mode Register 0

SFR Page = All

SFR Address = 0x91 POR+RESET = 0000-0000

7	6	5	4	3	2	1	0
P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P1M1: Port 1 Mode Register 1

SFR Page = All

SFR Address = 0x92 POR+RESET = 0000-0000

7	6	5	4	3	2	1	0
P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

9.2.3. Port 2 Register

P2: Port 2 Register

SFR Page = All

SFR Address = 0xA0 RESET = 1111-1111

7	6	5	4	3	2	1	0
P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7~0: P2.7~P2.0 could be only set/cleared by CPU.

P2M0: Port 2 Mode Register 0

SFR Page = All

SFR Address = 0x95 RESET = 0000-0000

7	6	5	4	3	2	1	0
P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P2M1: Port 2 Mode Register 1

SFR Page = All

SFR Address = 0x96 RESET = 0000-0000

7	6	5	4	3	2	1	0
P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

9.2.4. Port 3 Register

P3: Port 3 Register

SFR Page = All

SFR Address = 0xB0 RESET = 1111-1111

7	6	5	4	3	2	1	0
P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7~0: P3.7~P3.0 could be only set/cleared by CPU.

P3M0: Port 3 Mode Register 0

SFR Address = 0xB1

SFR Page = All RESET = 0000-0000

7	6	5	4	3	2	1	0
P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P3M1: Port 3 Mode Register 1

SFR Page = All

SFR Address = 0xB2 RESET = 0000-0000

7	6	5	4	3	2	1	0
P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

9.2.5. Port 4 Register

P4: Port 4 Register

SFR Page = All

SFR Address = 0xE8 RESET = x111-1111

7	6	5	4	3	2	1	0
--	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 6~0: P4.6~P4.0 could be only set/cleared by CPU.
P4.6 is an alternated function on ALE pin. When CPU executes off-chip memory access, MOVX, this pin is ALE function in MOVX cycle.

P4M0: Port 4 Mode Register 0

SFR Page = All

SFR Address = 0xB3 RESET = x000-0000

7	6	5	4	3	2	1	0
--	P4M0.6	P4M0.5	P4M0.4	P4M0.3	P4M0.2	P4M0.1	P4M0.0
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P4M1: Port 4 Mode Register 1

SFR Page = All

SFR Address = 0xB4 RESET = x000-0000

7	6	5	4	3	2	1	0
--	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0
R	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

9.2.6. Port 5 Register

P5: Port 5 Register

SFR Page = All

SFR Address = 0xF8 RESET = xxxx-1111

7	6	5	4	3	2	1	0
--	--	--	--	P5.3	P5.2	P5.1	P5.0
R	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: P5.3~P5.0 could be only set/cleared by CPU.

P5M0: Port 5 Mode Register 0

SFR Page = All

SFR Address = 0xB5 RESET = xxxx-0000

7	6	5	4	3	2	1	0
--	--	--	--	P5M0.3	P5M0.2	P5M0.1	P5M0.0
R	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**

P5M1: Port 5 Mode Register 1

SFR Page = All

SFR Address = 0xB6 RESET = 0000-0000

7	6	5	4	3	2	1	0
--	--	--	--	P5M1.3	P5M1.2	P5M1.1	P5M1.0
R	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**

9.2.7. Port 6 Register

P6: Port 6 Register

SFR Page = **F** only

SFR Address = **0xC8** RESET = xxxx-xx11

7	6	5	4	3	2	1	0
--	--	--	--	--	--	P6.1	P6.0
R	**R**	**R**	**R**	**R**	**R**	**R/W**	**R/W**

Bit 7~2: Reserved.

Bit 1~0: P6.1~P6.0 could be only set/cleared by CPU. These two I/Os are active when Internal Oscillator is enabled for system clock. Then, XTAL1 and XTAL2 behave P6.1 and P6.0. They only support one I/O mode, quasi-bidirectional mode.

9.3. Alternate Function Redirection

Many I/O pins, in addition to their normal I/O function, also serve the alternate function for internal peripherals. For the peripherals Keypad interrupt, PCA, SPI and UART1, Port 2 and Port 1 serve the alternate function in the default state. However, the user may select Port 4 and Port 5 to serve their alternate function by setting the corresponding control bits P4KB, P4PCA, P5SPI and P4S1 in AUXR1 register. It is especially useful when the packages more than 40 pins are adopted. Note that only one of the four control bits can be set at any time.

AUXR1: Auxiliary Control Register 1

SFR Page = All

SFR Address = 0xA2 RESET = 0000-XXX0

7	6	5	4	3	2	1	0
P4KBI	P4PCA	P5SPI	P4S1	--	--	--	DPS
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**

Bit 7: P4KBI, KBI function on P4/P5.
0: Disable KBI function moved to P4/P5.
1: Set KBI function on P4/P5 as following definition.
'KBI0' function in P2.0 is moved to P4.0.
'KBI1' function in P2.1 is moved to P4.1.
'KBI2' function in P2.2 is moved to P4.2.
'KBI3' function in P2.3 is moved to P4.3.
'KBI5' function in P2.5 is moved to P5.1.
'KBI4' function in P2.4 is moved to P5.0.
'KBI6' function in P2.6 is moved to P5.2.
'KBI7' function in P2.7 is moved to P5.3.

Bit 6: P4PCA, PCA function on P4/P5.
0: Disable PCA function moved to P4/P5.
1: Set PCA function on P4/P5 as following definition.
'ECI' function in P1.1 is moved to P4.2.
'CEX0' function in P1.2 is moved to P4.0.
'CEX1' function in P1.3 is moved to P4.1.
'CEX2' function in P1.4 is moved to P5.0.
'CEX3' function in P1.5 is moved to P5.1
'CEX4' function in P1.6 is moved to P5.2.
'CEX5' function in P1.7 is moved to P5.3.

Bit 5: P5SPI, SPI interface on P5.3~P5.0.
0: Disable SPI function moved to P5.
1: Set SPI function on P5 as following definition.
'nSS' function in P1.4 is moved to P5.0.
'MOSI' function in P1.5 is moved to P5.1.
'MISO' function in P1.6 is moved to P5.2.
'SPICLK' function in P1.7 is moved to P5.3.

Bit 4: P4S1, Serial Port 1 (UART1) on P4.0/P4.1.
0: Disable UART1 function moved to P4.
1: Set UART1 RXD1/TXD1 on P4.0/P4.1 following definition.
'RXD1' function in P1.2 is moved to P4.0.
'TXD1' function in P1.3 is moved to P4.1.

9.4. GPIO Sample Code

(1). Required Function: Set P1.0 to input-only mode

Assembly Code Example:

```
P1Mn0           EQU             01h

   ORL    P1M0, #P1Mn0
   ANL    P1M1, #(0FFh + P1Mn0)      ; Configure P1.0 to input only mode
   SETB   P1.0                       ; Set P1.0 data latch to “1” to enable input mode
```

C Code Example:

```
#define  P1Mn0          0x01

   P1M0 |= P1Mn0;
   P1M1 &= ~P1Mn0;                   // Configure P1.0 to input only mode
   P10 = 1;                          // Set P1.0 data latch to “1” to enable input mode
```

(2). Required Function: Set P1.0 to push-pull output mode

Assembly Code Example:

```
P1Mn0           EQU             01h

   ANL    P1M0, #(0FFh - P1Mn0)
   ORL    P1M1, #P1Mn0            ; Configure P1.0 to push pull mode
   SETB   P1.0
```

C Code Example:

```
#define  P1Mn0          0x01

   P1M0 &= ~P1Mn0;
   P1M1 |= P1Mn0;                    // Configure P1.0 to push pull mode
   P10 = 1;                          // Set P1.0 data latch to “1” to enable push pull mode
```

(3). Required Function: Set P1.0 to open-drain output mode

Assembly Code Example:

```
P1Mn0           EQU             01h

   ORL    P1M0, #P1Mn0
   ORL    P1M1, #P1Mn0              ; Configure P1.0 to open drain mode
   SETB   P1.0                      ; Set P1.0 data latch to “1” to enable open drain mode
```

C Code Example:

```
#define  P1Mn0          0x01

   P1M0 |= P1Mn0;
   P1M1 |= P1Mn0;                    // Configure P1.0 to open drain mode
   P10 = 1;                          // Set P1.0 data latch to “1” to enable open drain mode
```

10. Interrupt

The MG82Fx564 has 14 interrupt sources with a four-level interrupt structure. There are several SFRs associated with the four-level interrupt. They are the IE, IP0L, IP0H, EIE1, EIP1L, EIP1H and XICON. The IP0H (Interrupt Priority 0 High) and EIP1H (Extended Interrupt Priority 1 High) registers make the four-level interrupt structure possible. The four priority level interrupt structure allows great flexibility in handling these interrupt sources.

10.1. Interrupt Structure

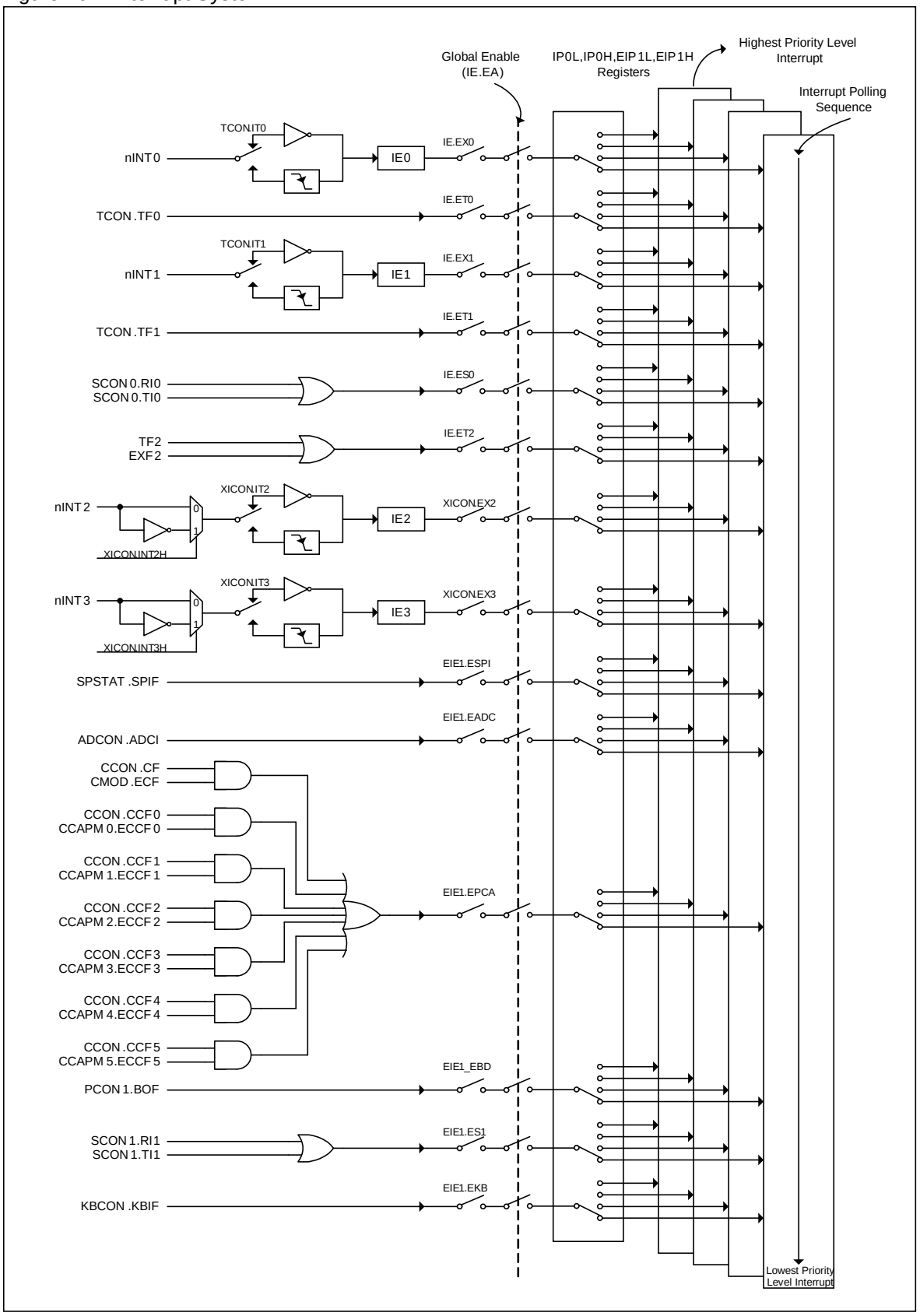
Table 10-1 lists all the interrupt sources. The 'Request Bits' are the interrupt flags that will generate an interrupt if it is enabled by setting the 'Enable Bit'. Of course, the global enable bit EA (in IE0 register) should have been set previously. The 'Request Bits' can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled in software. The 'Priority Bits' determine the priority level for each interrupt. The 'Priority within Level' is the polling sequence used to resolve simultaneous requests of the same priority level. The 'Vector Address' is the entry point of an interrupt service routine in the program memory.

Figure 10-1 shows the interrupt system. Each of these interrupts will be briefly described in the following sections.

Table 10-1. Interrupt Sources

No	Source Name	Enable Bit	Request Bits	Priority Bits	Priority Within Level	Vector Address	C51 Vector No.
#1	External Interrupt, nINT0	EX0	IE0	PX0H, PX0L	(Highest)	0003H	0
#2	Timer 0	ET0	TF0	PT0H, PT0L	…	000Bh	1
#3	External Interrupt, nINT1	EX1	IE1	PX1H, PX1L	...	0013H	2
#4	Timer 1	ET1	TF1	PT1H, PT1L	…	001BH	3
#5	Serial Port 0	ES0	RI0, TI0	PS0H, PS0L	…	0023H	4
#6	Timer 2	ET2	TF2, EXF2	PT2H, PT2L	…	002Bh	5
#7	External Interrupt, nINT2	EX2	IE2	PX2H, PX2L	…	0033H	6
#8	External Interrupt, nINT3	EX3	IE3	PX3H, PX3L	…	003BH	7
#9	SPI	ESPI	SPIF	PSPIH, PSPIL	…	0043H	8
#10	ADC	EADC	ADCI	PADCH, PADCL	…	004Bh	9
#11	PCA	EPCA	CF, CCFn (n=0~5)	PPCAH, PPCAL	…	0053H	10
#12	Brownout Detection	EBD	BOF	PBDH, PBDL	…	005BH	11
#13	Serial Port 1	ES1	RI1, TI1	PS1H, PS1L	…	0063H	12
#14	Keypad Interrupt	EKB	KBIF	PKBH, PKBL	(Lowest)	006BH	13

Figure 10-1 Interrupt System



10.2. Interrupt Register

IE: Interrupt Enable Register

SFR Page = All

SFR Address = 0xE8 RESET = 0X00-0000

7	6	5	4	3	2	1	0
EA	--	ET2	ES0	ET1	EX1	ET0	EX0
R/W	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: EA, All interrupts enable register.
0: Global disables all interrupts.
1: Global enables all interrupts.

Bit 6: Reserved. Software must write “0” on this bit when IE is written.

Bit 5: ET2, Timer 2 interrupt enable register.
0: Disable Timer 2 interrupt.
1: Enable Timer 2 interrupt.

Bit 4: ES, Serial port 0 interrupt enable register.
0: Disable serial port 0 interrupt.
1: Enable serial port 0 interrupt.

Bit 3: ET1, Timer 1 interrupt enable register.
0: Disable Timer 1 interrupt.
1: Enable Timer 1 interrupt.

Bit 2: EX1, External interrupt 1 enable register.
0: Disable external interrupt 1.
1: Enable external interrupt 1.

Bit 1: ET0, Timer 0 interrupt enable register.
0: Disable Timer 0 interrupt.
1: Enable Timer 1 interrupt.

Bit 0: EX0, External interrupt 0 enable register.
0: Disable external interrupt 0.
1: Enable external interrupt 1.

XICON: External Interrupt Control Register

SFR Page = All

SFR Address = 0xC0 RESET = 0000-0000

7	6	5	4	3	2	1	0
INT3H	EX3	IE3	IT3	**INT2H**	EX2	IE2	IT2
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: INT3H, nINT3 High/Rising trigger enable.
0: Maintain nINT3 triggered on low level or falling edge on P4.2.
1: Set nINT3 triggered on high level or rising edge on P4.2.

Bit 6: EX3, external interrupt 3 enable register.
0: Disable external interrupt 3.
1: Enable external interrupt 3.

Bit 5: IE3, External interrupt 3 Edge flag.
0: Cleared by hardware when the interrupt is starting to be serviced. It also could be cleared by CPU.
1: Set by hardware when external interrupt edge detected. It also could be set by CPU.

Bit 4: IT3, Interrupt 3 type control bit.
0: Cleared by CPU to specify low level triggered on nINT3. If INT3H is set, this bit specifies high level triggered on nINT3.
1: Set by CPU to specify falling edge triggered on nINT3. If INT3H is set, this bit specifies rising edge triggered on nINT3.

Bit 3: INT2H, nINT2 High/Rising trigger enable.
0: Maintain nINT2 triggered on low level or falling edge on P4.3.
1: Set nINT2 triggered on high level or rising edge on P4.3.

Bit 2: EX2, external interrupt 2 enable register.
0: Disable external interrupt 2.
1: Enable external interrupt 2.

Bit 1: IE2, External interrupt 2 Edge flag.
0: Cleared by hardware when the interrupt is starting to be serviced. It also could be cleared by CPU.
1: Set by hardware when external interrupt edge detected. It also could be set by CPU.

Bit 0: IT2, Interrupt 2 type control bit.
0: Cleared by CPU to specify low level triggered on nINT2. If INT2H is set, this bit specifies high level triggered on nINT2.
1: Set by CPU to specify falling edge triggered on nINT2. If INT2H is set, this bit specifies rising edge triggered on nINT2.

EIE1: Extended Interrupt Enable 1 Register

SFR Page = All
SFR Address = 0xAD RESET = XX00-0000

7	6	5	4	3	2	1	0
--	--	EKBI	ES1	EBD	EPCA	EADC	ESPI
R	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: Reserved. Software must write ”0”s on these bits when IEI1 is written.

Bit 5: EKBI, Enable Keypad Interrupt.
0: Disable the interrupt when KBCON.KBIF is set in Keypad control module.
1: Enable the interrupt when KBCON.KBIF is set in Keypad control module.

Bit 4: ES1, Enable Serial Port 1 (UART1) interrupt.
0: Disable Serial Port 1 interrupt.
1: Enable Serial Port 1 interrupt.

Bit 3: EBD, Enable Brown-out Detection interrupt.
0: Disable the interrupt when PCON1.BOF is set in brown-out detection module.
1: Enable the interrupt when PCON1.BOF is set in brown-out detection module.

Bit 2: EPCA, Enable PCA interrupt.
0: Disable PCA interrupt.
1: Enable PCA interrupt.

Bit 1: EACI, Enable ADC Interrupt.
0: Disable the interrupt when ADCON.ADCI is set in ADC module.
1: Enable the interrupt when ACCON.ADCI is set in ADC module.

Bit 0: ESPI, Enable SPI Interrupt.
0: Disable the interrupt when SPSTAT.SPIF is set in SPI module.
1: Enable the interrupt when SPSTAT.SPIF is set in SPI module.

IP0L: Interrupt Priority 0 Low Register

SFR Page = All

SFR Address = 0xB8 RESET = 0000-0000

7	6	5	4	3	2	1	0
PX3L	**PX2L**	PT2L	PSL	PT1L	PX1L	PT0L	PX0L
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: PX3L, external interrupt 3 priority-L register.
Bit 6: PX2L, external interrupt 2 priority-L register.
Bit 5: PT2L, Timer 2 interrupt priority-L register.
Bit 4: PSL, Serial port interrupt priority-L register.
Bit 3: PT1L, Timer 1 interrupt priority-L register.
Bit 2: PX1L, external interrupt 1 priority-L register.
Bit 1: PT0L, Timer 0 interrupt priority-L register.
Bit 0: PX0L, external interrupt 0 priority-L register.

IP0H: Interrupt Priority 0 High Register

SFR Page = All

SFR Address = 0xB7 RESET = 0000-0000

7	6	5	4	3	2	1	0
PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: PX3H, external interrupt 3 priority-H register.
Bit 6: PX2H, external interrupt 2 priority-H register.
Bit 5: PT2H, Timer 2 interrupt priority-H register.
Bit 4: PSH, Serial port interrupt priority-H register.
Bit 3: PT1H, Timer 1 interrupt priority-H register.
Bit 2: PX1H, external interrupt 1 priority-H register.
Bit 1: PT0H, Timer 0 interrupt priority-H register.
Bit 0: PX0H, external interrupt 0 priority-H register.

EIP1L: Extended Interrupt Priority 1 Low Register

SFR Page = All

SFR Address = 0xAE RESET = XX00-0000

7	6	5	4	3	2	1	0
--	--	PKBL	PS1L	PBDL	PPCAL	PADCL	PSPIL
R	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: Reserved. Software must write “0”s on these bits when EIP1L is written.
Bit 5: PKBL, keypad interrupt priority-L register.
Bit 4: PS1L, UART1 interrupt priority-L register.
Bit 3: PBDL, brown-out detection interrupt priority-L register.
Bit 2: PPCAL, PCA interrupt priority-L register.
Bit 1: PADCL, ADC interrupt priority-L register.
Bit 0: PSPIL, SPI interrupt 0 priority-L register.

EIP1H: Extended Interrupt Priority 1 High Register

SFR Page = All

SFR Address = 0xAF RESET = XX00-0000

7	6	5	4	3	2	1	0
--	--	PKBH	PS1H	PBDH	PPCAH	PADCH	PSPIH
R	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: Reserved. Software must write “0”s on these bits when EIP1H is written.
Bit 5: PKBH, keypad interrupt priority-H register.
Bit 4: PS1H, UART1 interrupt priority-H register.
Bit 3: PBDH, brown-out detection interrupt priority-H register.
Bit 2: PPCAH, PCA interrupt priority-H register.
Bit 1: PADCH, ADC interrupt priority-H register.
Bit 0: PSPIH, SPI interrupt 0 priority-H register.

IP0L, IP0H, EIP1L and EIP1H are combined to 4-level priority interrupt as the following table.

{IPH.x , IPL.x}	Priority Level
11	1 (highest)
10	2
01	3
00	4

There are 14 interrupt sources available in MG82Fx564. Each interrupt source can be individually enabled or disabled by setting or clearing a bit in the SFRs named IE, EIE1, and XICON. This register also contains a global disable bit(EA), which can be cleared to disable all interrupts at once.

Each interrupt source has two corresponding bits to represent its priority. One is located in SFR named IPxH and the other in IPxL register. Higher-priority interrupt will be not interrupted by lower-priority interrupt request. If two interrupt requests of different priority levels are received simultaneously, the request of higher priority is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determine which request is serviced. The following table shows the internal polling sequence in the same priority level and the interrupt vector address.

Source	Vector address	Priority within level
External interrupt 0	0003H	1 (highest)
Timer 0	000BH	2
External interrupt 1	0013H	3
Timer1	001BH	4
Serial Port 0	0023H	5
Timer2	002BH	6
External interrupt 2	0033H	7
External interrupt 3	003BH	8
SPI	0043H	9
ADC	004BH	10
PCA Counter	0053H	11
Brown-out Detection	005BH	12
Serial Port 1	0063H	13
Keypad Interrupt	006BH	14

The external interrupt nINT0, nINT1, nINT2 and nINT3 can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in register TCON, IT2 and IT3 in register XICON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON, IE2 and IE3 in XICON. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to *only if the interrupt was transition –activated*, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

The Timer0 and Timer1 interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers in most cases. When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The serial port **0** interrupt is generated by the logical OR of **RI0** and **TI0**. Neither of these flags is cleared by hardware when the service routine is vectored to. The service routine should poll **RI0** and **TI0** to determine which one to request service and it will be cleared by software.

The timer2 interrupt is generated by the logical OR of TF2 and EXF2. Just the same as serial port, neither of these flags is cleared by hardware when the service routine is vectored to.

SPI interrupt is generated by

The ADC interrupt is generated by ADCI in ADCON. It will not be cleared by hardware when the service routine is vectored to.

The PCA interrupt is generated by the logical OR of CF, CCF5, CCF4, CCF3, CCF2, CCF1 and CCF0 in CCON. Neither of these flags is cleared by hardware when the service routine is vectored to. The service routine should poll these flags to determine which one to request service and it will be cleared by software.

The BOD interrupt is generated by BOD in PCON1, which is set by on chip Brownout-Detector meets the low voltage event. It will not be cleared by hardware when the service routine is vectored to.

The serial port **1** interrupt is generated by the logical OR of **RI1** and **TI1**. Neither of these flags is cleared by hardware when the service routine is vectored to. The service routine should poll **RI1** and **TI1** to determine which one to request service and it will be cleared by software.

The keypad interrupt is generated by KBCON.KBIF, which is set by Keypad module meets the input pattern. It will not be cleared by hardware when the service routine is vectored to.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. In other words, interrupts can be generated or pending interrupts can be canceled in software.

How hardware see the interrupts

Each interrupt flag is sampled at every system clock cycle. The samples are polled during the next system clock. If one of the flags was in a set condition at first cycle, the second cycle(polling cycle) will find it and the interrupt system will generate an hardware LCALL to the appropriate service routine as long as it is not blocked by any of the following conditions.

Block conditions:

- An interrupt of equal or higher priority level is already in progress.
- The current cycle (polling cycle) is not the final cycle in the execution of the instruction in progress.
- The instruction in progress is RETI or any write to the IE, IP0L, IPH, XICON, EIE1, EIP1L and EIP1H registers.

Any of these three conditions will block the generation of the hardware LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be completed before vectoring into any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least one or more instruction will be executed before any interrupt is vectored to.

The polling cycle is repeated with each system clock cycle, and the values polled are the values that were present at the previous system clock cycle. Note that if an interrupt flag is active but not being responded to for one of the above conditions, if the flag is not still active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not being responded to for one of the above conditions, if the flag is not still active when the blocking condition is removed, the denied interrupt will not be serviced. The interrupt flag was once active but not serviced is not kept in memory. Each polling cycle is new.

10.1. Interrupt Sample Code

(1). Required Function: Set INT0 wake-up MCU in power-down mode

Assembly Code Example:

```
PX0                 EQU         01h
PX0H                EQU         01h
PD                  EQU         02h

   ORG    0000h
   JMP    main

   ORG    00003h
ext_int0_isr:
   to do.....
   RETI

main:

   SETB    P3.2                ;

   ORL     IP,#PX0             ; Select INT0 interrupt priority
   ORL     IPH,#PX0H           ;

   JB      P3.2, $             ; Confirm P3.2 input low?????

   SETB    EX0                 ; Enable INT0 interrupt
   CLR     IE0                 ; Clear INT0 flag
   SETB    EA                  ; Enable global interrupt

   ORL     PCON,#PD            ; Set MCU into Power Down mode

   JMP     $
```

C Code Example:

```
#define  PX0                    0x01
#define  PX0H                   0x01
#define  PD                     0x02

void ext_int0_isr(void) interrupt 0
{
   To do……
}

void main(void)
{
   P32 = 1;

   IP |= PX0;                   // Select INT0 interrupt priority
   IPH |= PX0H;

   while(P32);                  // Confirm P3.2 input low??????

   EX0 = 1;                     // Enable INT0 interrupt
   IE0 = 0;                     // Clear INT0 flag
   EA = 1;                      // Enable global interrupt

   PCON |= PD;                  // Set MCU into Power Down mode

   while(1);
}
```

11. Timers/Counters

MG82Fx564 has three 16-bit Timers/Counters: Timer 0, Timer 1 and Timer 2. All of them can be configured as timers or event counters.

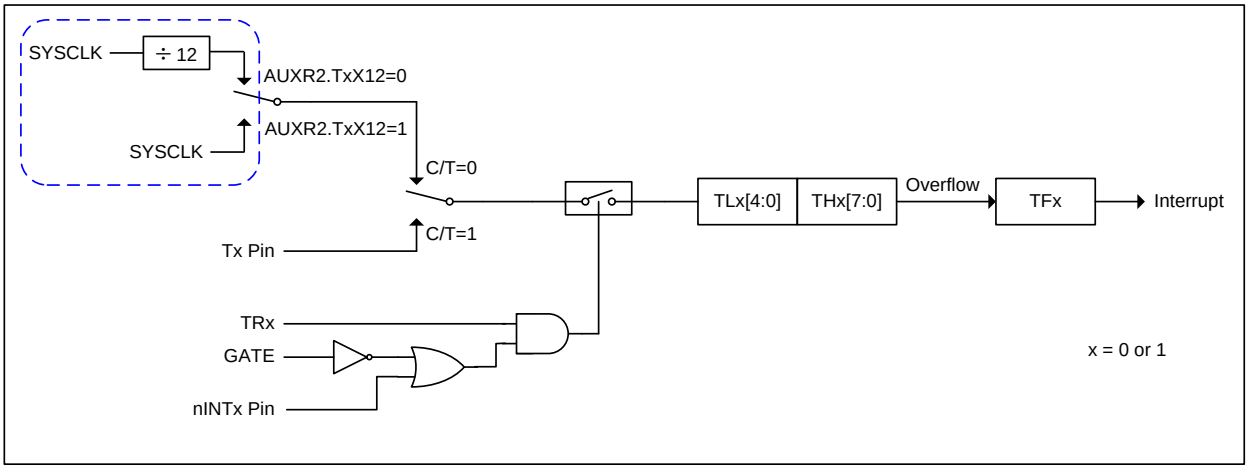
In the “timer” function, the timer rate is prescaled by 12 clock cycle to increment register value. In other words, it is to count the standard C51 machine cycle. AUXR2.T0X12, AUXR2.T1X12 and T2MOD.T2X12 are the function for Timer 0/1/2 to set the timer rate on every clock cycle. It behaves X12 times speed than standard C51 timer function.

In the “counter” function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0, T1 or T2. In this function, the external input is sampled by every timer rate cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register at the end of the cycle following the one in which the transition was detected.

11.1. Timer0 and Timer1

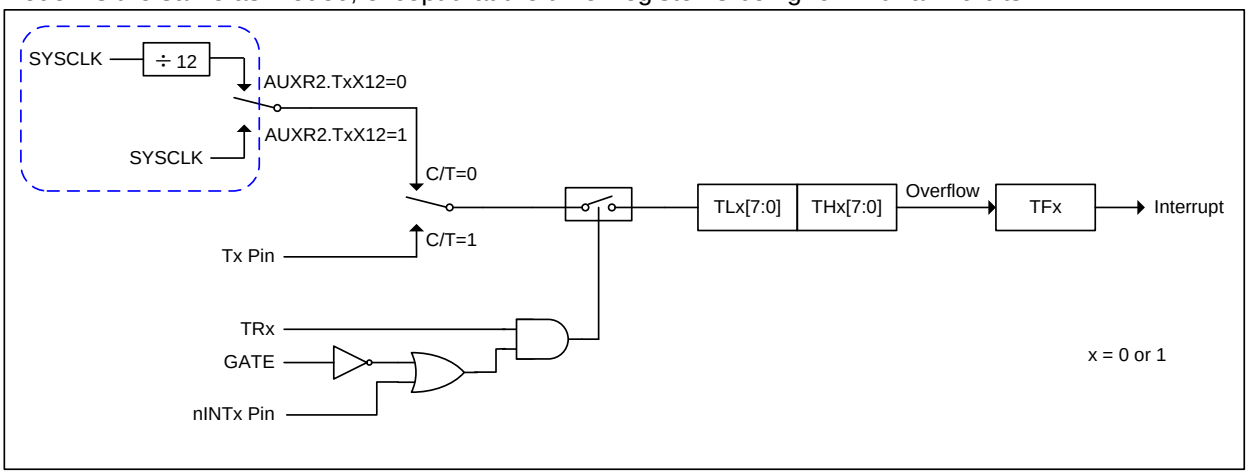
11.1.1. Mode 0 Structure

The timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the timer interrupt flag TFx. The counted input is enabled to the timer when TRx = 1 and either GATE=0 or INTx = 1. Mode 0 operation is the same for Timer0 and Timer1.



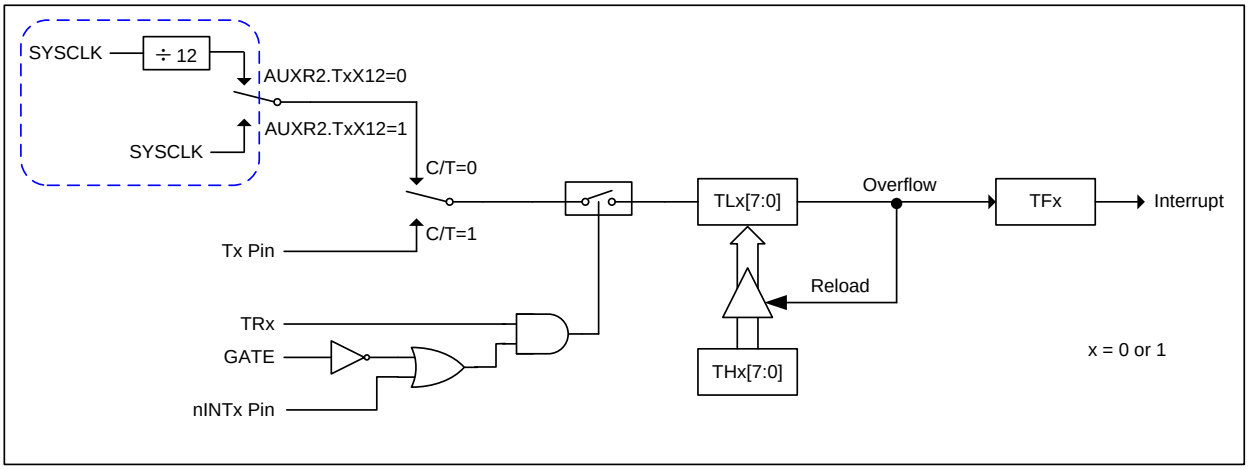
11.1.2. Mode 1 Structure

Mode1 is the same as Mode0, except that the timer register is being run with all 16 bits.



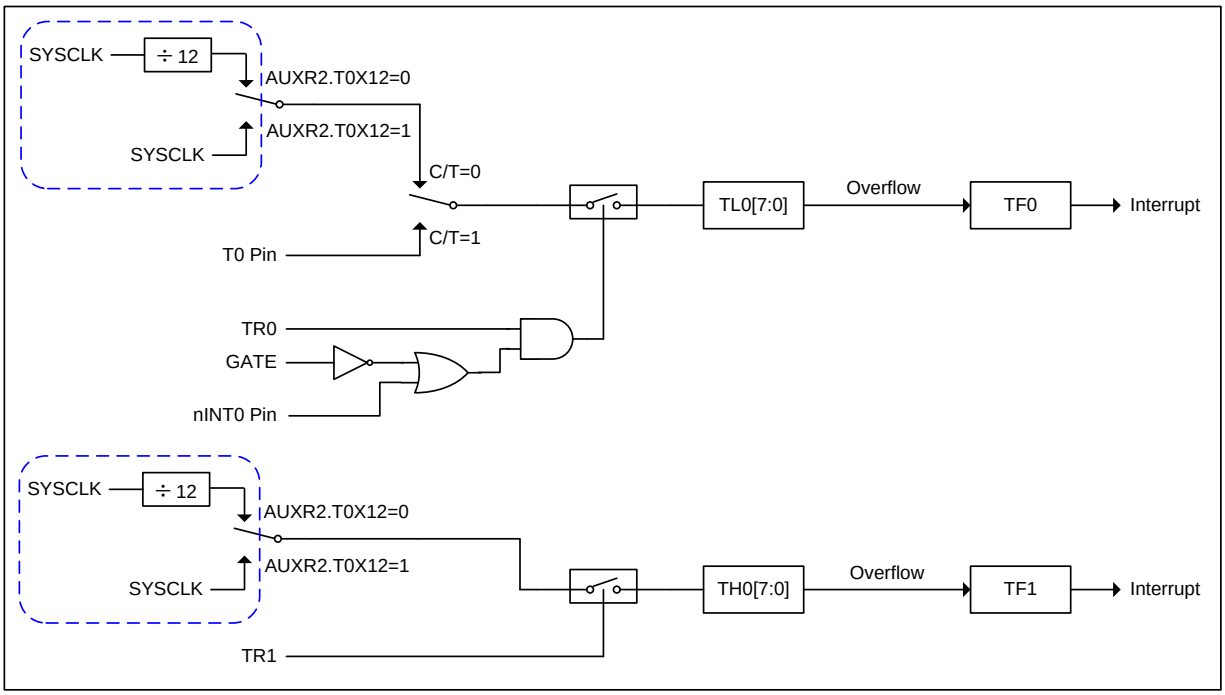
11.1.3. Mode 2 Structure

Mode 2 configures the timer register as an 8-bit counter(TLx) with automatic reload. Overflow from TLx not only set TFx, but also reload TLx with the content of THx, which is determined by software. The reload leaves THx unchanged. Mode 2 operation is the same for Timer0 and Timer1.

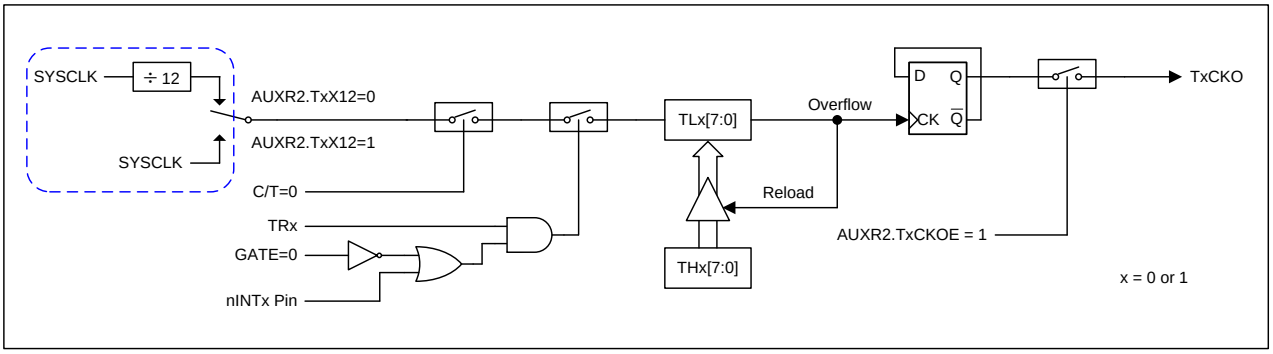


11.1.4. Mode 3 Structure

Timer1 in Mode3 simply holds its count, the effect is the same as setting TR1 = 1. Timer0 in Mode 3 enables TL0 and TH0 as two separate 8-bit counters. TL0 uses the Timer0 control bits such like C/T, GATE, TR0, INT0 and TF0. TH0 is locked into a timer function (can not be external event counter) and take over the use of TR1, TF1 from Timer1. TH0 now controls the Timer1 interrupt.



11.1.5. Timer Clock-Out Structure



$$\text{T0/T1 Clock-out Frequency} = \frac{\text{SYSCLK Frequency}}{n \times (256 - THx)}$$

; n=24, if TxX12=0
; n=2, if TxX12=1
; x = 0 or 1 & C/T = 0

11.1.6. Timer0/1 Register

TMOD: Timer/Counter Mode Control Register

SFR Page = All

SFR Address = 0x89 RESET = 0000-0000

7	6	5	4	3	2	1	0
GATE	C/T	M1	M0	GATE	C/T	M1	M0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**
|←---------- Timer1 ----------→|				|←---------- Timer0 ----------→|		

Bit 7/3: Gate, Gating control for Timer1/0.
0: Disable gating control for Timer1/0.
1: Enable gating control for Timer1/0. When set, Timer1/0 or Counter1/0 is enabled only when /INT1 or /INT0 pin is high and TR1 or TR0 control bit is set.

Bit 6/2: C/T, Timer for Counter function selector.
0: Clear for Timer operation, input from internal system clock.
1: Set for Counter operation, input form T1 input pin.

Bit 5~4/1~0: Operating mode selection.

M1	**M0**	**Operating Mode**
0	0	13-bit timer/counter for Timer0 and Timer1
0	1	16-bit timer/counter for Timer0 and Timer1
1	0	8-bit timer/counter with automatic reload for Timer0 and Timer1
1	1 (Timer0)	TL0 is 8-bit timer/counter, TH0 is locked into 8-bit timer
1	1 (Timer1)	Timer/Counter1 Stopped

TCON: Timer/Counter Control Register

SFR Page = All

SFR Address = 0x88 RESET = 0000-0000

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: TF1, Timer 1 overflow flag.
0: Cleared by hardware when the processor vectors to the interrupt routine, or cleared by software.
1: Set by hardware on Timer/Counter 1 overflow, or set by software.

Bit 6: TR1, Timer 1 Run control bit.
0: Cleared by software to turn Timer/Counter 1 off.
1: Set by software to turn Timer/Counter 1 on.

Bit 5: TF0, Timer 0 overflow flag.
0: Cleared by hardware when the processor vectors to the interrupt routine, or cleared by software.
1: Set by hardware on Timer/Counter 0 overflow, or set by software.

Bit 4: TR0, Timer 0 Run control bit.
0: Cleared by software to turn Timer/Counter 0 off.
1: Set by software to turn Timer/Counter 0 on.

Bit 3: IE1, Interrupt 1 Edge flag.
0: Cleared when interrupt processed on if transition-activated.
1: Set by hardware when external interrupt 1 edge is detected (transmitted or level-activated).

Bit 2: IT1: Interrupt 1 Type control bit.
0: Cleared by software to specify low level triggered external interrupt 1.
1: Set by software to specify falling edge triggered external interrupt 1.

Bit 1: IE0, Interrupt 0 Edge flag.
0: Cleared when interrupt processed on if transition-activated.
1: Set by hardware when external interrupt 0 edge is detected (transmitted or level-activated).

Bit 0: IT0: Interrupt 0 Type control bit.
0: Cleared by software to specify low level triggered external interrupt 0.
1: Set by software to specify falling edge triggered external interrupt 0.

TL0: Timer Low 0 Register

SFR Page = All
SFR Address = 0x8A RESET = 0000-0000

7	6	5	4	3	2	1	0
TL0[7]	TL0[6]	TL0[5]	TL0[4]	TL0[3]	TL0[2]	TL0[1]	TL0[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

TH0: Timer High 0 Register

SFR Page = All
SFR Address = 0x8C RESET = 0000-0000

7	6	5	4	3	2	1	0
TH0[7]	TH0[6]	TH0[5]	TH0[4]	TH0[3]	TH0[2]	TH0[1]	TH0[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

TL1: Timer Low 1 Register

SFR Page = All
SFR Address = 0x8B RESET = 0000-0000

7	6	5	4	3	2	1	0
TL1[7]	TL1[6]	TL1[5]	TL1[4]	TL1[3]	TL1[2]	TL1[1]	TL1[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

TH1: Timer High 1 Register

SFR Page = All
SFR Address = 0x8D RESET = 0000-0000

7	6	5	4	3	2	1	0
TH1[7]	TH1[6]	TH1[5]	TH1[4]	TH1[3]	TH1[2]	TH1[1]	TH1[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

AUXR2: Auxiliary Register 2

SFR Page = All
SFR Address = 0xA6 RESET = 00XX-XX00

7	6	5	4	3	2	1	0
T0X12	T1X12	--	--	--	--	T1CKOE	T0CKOE
R/W	**R/W**	**R**	**R**	**R**	**R**	**R/W**	**R/W**

Bit 7: T0X12, Timer 1 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source.

Bit 6: T1X12, Timer 1 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source.

Bit 1: T1CKOE, Timer 1 Clock Output Enable.
0: Disable Timer 1 clock output.
1: Enable Timer 1 clock output on P3.5.

Bit 0: T0CKOE, Timer 0 Clock Output Enable.
0: Disable Timer 0 clock output.
1: Enable Timer 0 clock output on P3.4.

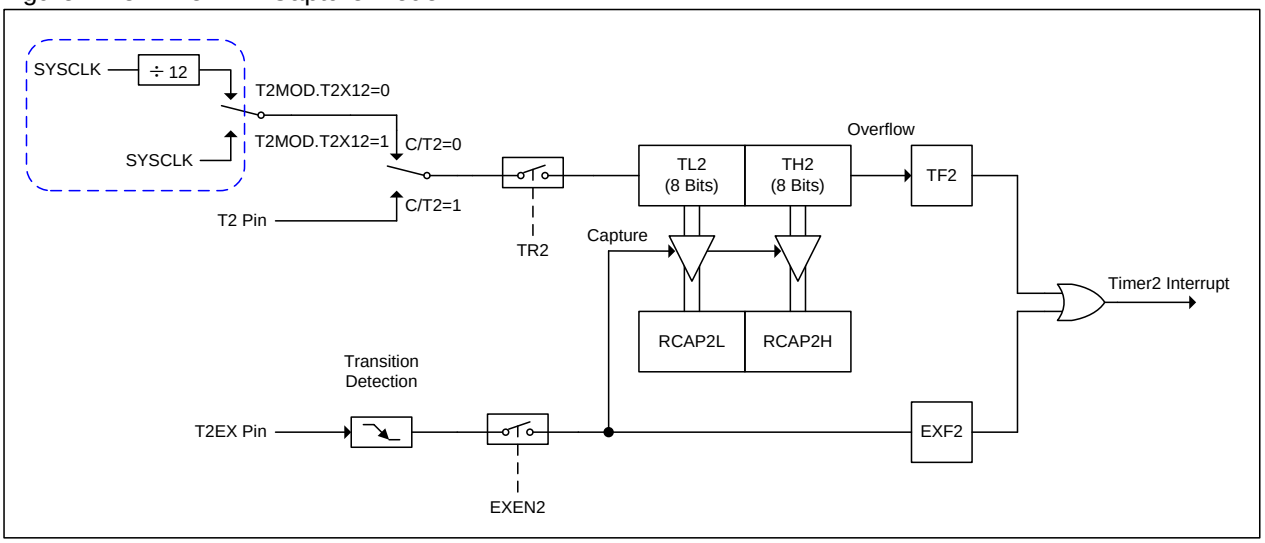
11.2. Timer2

Timer 2 is a 16-bit Timer/Counter which can operate either as a timer or an event counter, as selected by C/T2 in T2CON register. Timer 2 has four operating modes: Capture, Auto-Reload (up or down counting), Baud Rate Generator and Programmable Clock-Out, which are selected by bits in the T2CON and T2MOD registers.

11.2.1. Capture Mode (CP) Structure

In the capture mode there are two options selected by bit EXEN2 in T2CON. If EXEN2=0, Timer 2 is a 16-bit timer or counter which, upon overflow, sets bit TF2 (Timer 2 overflow flag). This bit can then be used to generate an interrupt (by enabling the Timer 2 interrupt bit in the IE register). If EXEN2=1, Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TH2 and TL2, to be captured into registers RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and the EXF2 bit (like TF2) can generate an interrupt which vectors to the same location as Timer 2 overflow interrupt. The capture mode is illustrated in Figure 11-5.

Figure 11-5 Timer 2 in Capture Mode



11.2.2. Auto-Reload Mode (AR) Structure

Figure 11-6 shows DCEN=0, which enables Timer 2 to count up automatically. In this mode there are two options selected by bit EXEN2 in T2CON register. If EXEN2=0, then Timer 2 counts up to 0FFFFH and sets the TF2 (Overflow Flag) bit upon overflow. This causes the Timer 2 registers to be reloaded with the 16-bit value in RCAP2L and RCAP2H. The values in RCAP2L and RCAP2H are preset by firmware. If EXEN2=1, then a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at input T2EX. This transition also sets the EXF2 bit. The Timer 2 interrupt, if enabled, can be generated when either TF2 or EXF2 are 1.

Figure 11-6 Timer 2 in Auto-Reload Mode (DCEN=0)

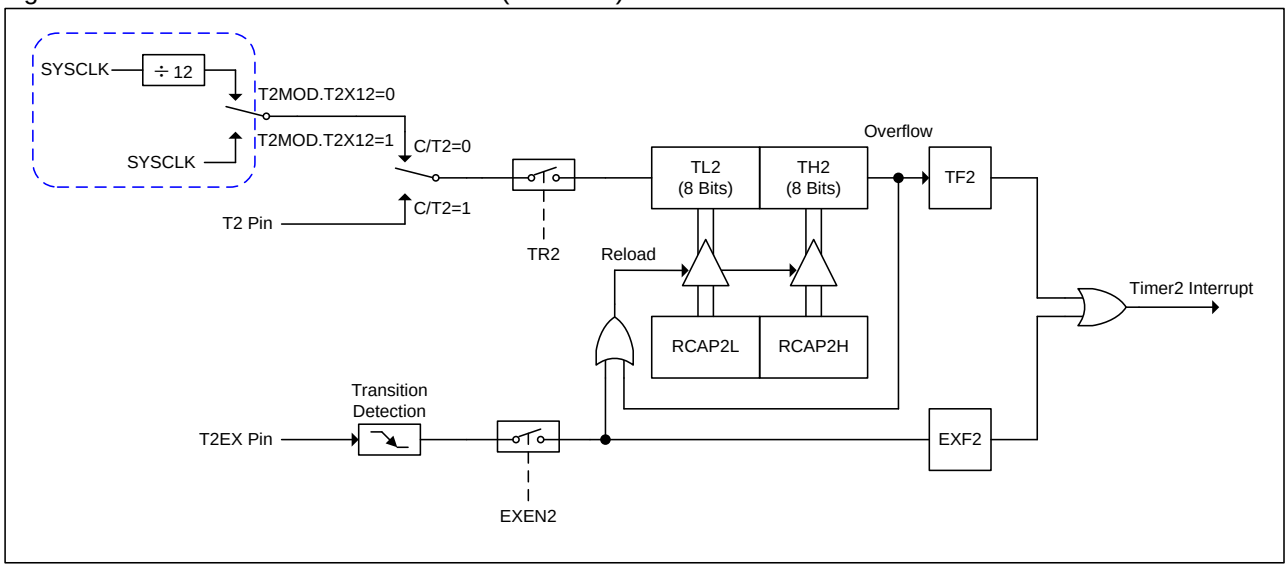
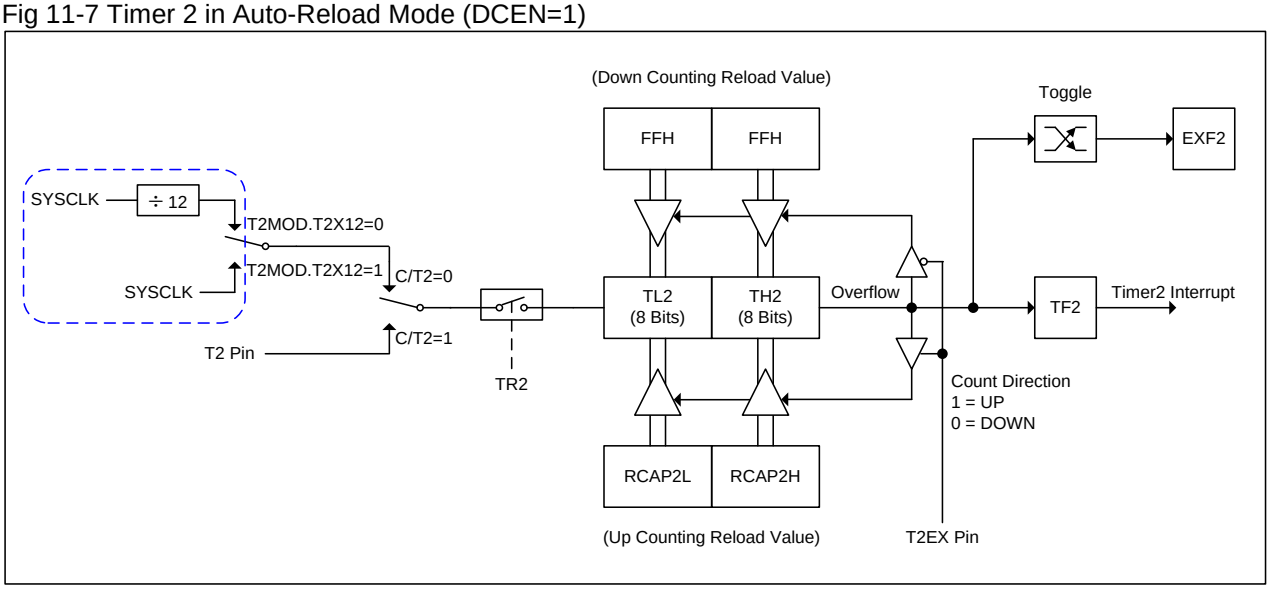


Fig 11-7 shows DCEN=1, which enables Timer 2 to count up or down. This mode allows pin T2EX to control the counting direction. When a logic 1 is applied at pin T2EX, Timer 2 will count up. Timer 2 will overflow at 0FFFFH and set the TF2 flag, which can then generate an interrupt if the interrupt is enabled. This overflow also causes the 16-bit value in RCAP2L and RCAP2H to be reloaded into the timer registers TL2 and TH2. A logic 0 applied to pin T2EX causes Timer 2 to count down. The timer will underflow when TL2 and TH2 become equal to the value stored in RCAP2L and RCAP2H. This underflow sets the TF2 flag and causes 0FFFFH to be reloaded into the timer registers TL2 and TH2.

The external flag EXF2 toggles when Timer 2 underflows or overflows. This EXF2 bit can be used as a 17th bit of resolution if needed. The EXF2 flag does not generate an interrupt in this mode.

Fig 11-7 Timer 2 in Auto-Reload Mode (DCEN=1)



11.2.3. Baud-Rate Generator Mode (BRG) Structure

Bits TCLK and/or RCLK in T2CON register allow the serial port transmit and receive baud rates to be derived from either Timer 1 or Timer 2. When TCLK=0, Timer 1 is used as the serial port transmit baud rate generator. When TCLK= 1, Timer 2 is used as the serial port transmit baud rate generator. RCLK has the same effect for the serial port receive baud rate. With these two bits, the serial port can have different receive and transmit baud rates – one generated by Timer 1, the other by Timer 2.

Fig 11-8 shows the Timer 2 in baud rate generation mode to generate RX Clock and TX Clock into UART engine (See Fig 12 6). The baud rate generation mode is like the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by firmware.

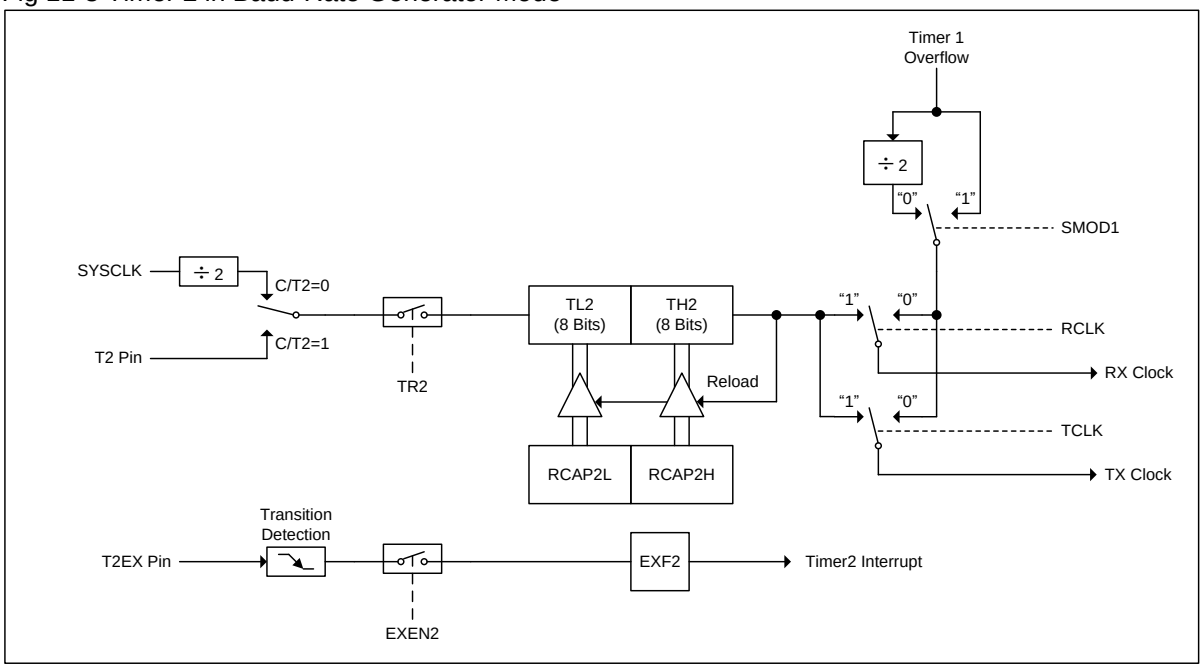
The Timer 2 as a baud rate generator mode is valid only if RCLK and/or TCLK=1 in T2CON register. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Thus, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Also if the EXEN2 (T2 external enable bit) is set, a 1-to-0 transition in T2EX (Timer/counter 2 trigger input) will set EXF2 (T2 external flag) but will not cause a reload from (RCAP2H, RCAP2L) to (TH2,TL2). Therefore when Timer 2 is in use as a baud rate generator, T2EX can be used as an additional external interrupt, if needed.

When Timer 2 is in the baud rate generator mode, one should not try to read or write TH2 and TL2. As a baud rate generator, Timer 2 is incremented at 1/2 the system clock or asynchronously from pin T2; under these conditions, a read or write of TH2 or TL2 may not be accurate. The RCAP2 registers may be read, but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

Note:

Refer to 12.7.3 Baud Rate in Mode 1 & 3 to get baud rate setting value when using Timer 2 as the baud rate generator.

Fig 11-8 Timer 2 in Baud-Rate Generator Mode



11.2.4. Programmable Clock Output from Timer 2 Structure

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, SYSCLK/2, increments the 16-bit timer (TH2, TL2). 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. The following formula gives the clock-out frequency:

$$\text{T2 Clock-out Frequency} = \frac{\text{SYSCLK Frequency}}{4 \times (65536 - (\text{RCAP2H, RCAP2L}))}$$

Note:
(1) Timer 2 overflow flag, TF2, will always not be set in this mode.
(2) For SYSCLK=12MHz, Timer 2 has a programmable output frequency range from 45.7Hz to 3MHz.

How to Program Timer 2 in 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 and RCAP2L registers.
- Enter the same reload value as the initial value in the TH2 and 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, that the baud-rate and the clock-out frequency depend on the same overflow rate of Timer 2.

11.2.5. Timer2 Register

T2MOD: Timer/Counter 2 Mode Control Register

SFR Page = All
SFR Address = 0xC9 RESET= XXX0-XX00

7	6	5	4	3	2	1	0
--	--	--	**T2X12**	--	--	T2OE	DCEN
R	**R**	**R**	**R/W**	**R**	**R**	**R/W**	**R/W**

Bit 7~5: Reserved. Software must write “0” on these bits when T2MOD is written.

Bit 4: T2X12, Timer 2 clock source selector.
0: Select SYSCLK/12 as Timer 2 clock source while T2CON.C/T2 = 0 in Capture Mode and Auto-Reload Mode.
1: Select SYSCLK as Timer 2 clock source while T2CON.C/T2 = 0 in Capture Mode and Auto-Reload

Bit 3~2: Reserved. Software must write “0” on these bits when T2MOD is written.

Bit 1: T2OE, Timer 2 clock-out enable bit.
0: Disable Timer 2 clock output.
1: Enable Timer 2 clock output.

Bit 0: DCEN, Timer 2 down-counting enable bit.
0: Timer 2 always keeps up-counting.
1: Enable Timer 2 down-counting ability.

T2CON: Timer/Counter 2 Mode Control Register

SFR Page = **0 Only**
SFR Address = 0xC8 RESET = 0000-0000

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

Bit 7: TF2, Timer 2 overflow flag.
0: TF2 must be cleared by software.
1: TF2 is set by a Timer 2 overflow happens. TF2 will not be set when either RCLK=1 or TCLK=1.

Bit 6: EXF2, Timer 2 external flag.
0: EXF2 must be cleared by software.
1: 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 does not cause an interrupt in up/down mode (DCEN = 1).

Bit 5: RCLK, Receive clock flag.
0: Causes Timer 1 overflow to be used for the receive clock.
1: Causes the serial port to use Timer 2 overflow pulses for its receive clock in modes 1 and 3.

Bit 4: TCLK, Transmit clock flag.
0: Causes Timer 1 overflows to be used for the transmit clock.
1: Causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3.

Bit 3: EXEN2, Timer 2 external enable flag.
0: Cause Timer 2 to ignore events at T2EX pin.
1: 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.

Bit 2: TR2, Timer 2 Run control bit.
0: Stop the Timer 2.
1: Start the Timer 2.

Bit 1: C/T2, Timer or counter selector.
0: Select Timer 2 as internal timer function.
1: Select Timer 2 as external event counter (falling edge triggered).

Bit 0: CP/-RL2, Capture/Reload flag.
0: Auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX pin when EXEN2=1.
1: Captures will occur on negative transitions at T2EX pin if 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.

When the DCEN is cleared, which makes the function of Timer 2 as the same as the standard 8052 (always counts up). When DCEN is set, 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.

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

TL2: Timer Low 2 Register

SFR Page = All
SFR Address = 0xCC RESET = 0000-0000

7	6	5	4	3	2	1	0
TL2.7	TL2.6	TL2.5	TL2.4	TL2.3	TL2.2	TL2.1	TL2.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

TH2: Timer High 2 Register

SFR Page = All

SFR Address = 0xCD RESET = 0000-0000

7	6	5	4	3	2	1	0
TH2[7]	TH2[6]	TH2[5]	TH2[4]	TH2[3]	TH2[2]	TH2[1]	TH2[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

RCAP2L: Timer 2 Capture Low Register

SFR Page = All

SFR Address = 0xCA RESET = 0000-0000

7	6	5	4	3	2	1	0
RCAP2L[7]	RCAP2L[6]	RCAP2L[5]	RCAP2L[4]	RCAP2L[3]	RCAP2L[2]	RCAP2L[1]	RCAP2L[1]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

RCAP2H: Timer 2 Capture High Register

SFR Page = All

SFR Address = 0xCB RESET = 0000-0000

7	6	5	4	3	2	1	0
RCAP2H[7]	RCAP2H[6]	RCAP2H[5]	RCAP2H[4]	RCAP2H[3]	RCAP2H[2]	RCAP2H[1]	RCAP2H[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

11.3. Timer0/1 Sample Code

(1). Required Function: IDLE mode with T0 wake-up frequency 10KHz, SYSCLK = 12MHz Crystal

Assembly Code Example:

```
T0M0            EQU             01h
T0M1            EQU             02h
PT0             EQU             02h
PT0H            EQU             02h
IDL             EQU             01h

   ORG   0000h
   JMP   main

   ORG   0000Bh
time0_isr:
   to do…
   RETI

main:                          ; (unsigned short value)

   MOV    TH0,#(256-100)           ; Set Timer 0 overflow rate = SYSCLK x 100
   MOV    TL0,#(256-100)           ;
   ANL    TMOD,#0F0h               ; Set Timer 0 to Mode 2
   ORL    TMOD,#T0M1               ;
   CLR    TF0                   ; Clear Timer 0 Flag

   ORL    IP,#PT0                  ; Select Timer 0 interrupt priority
   ORL    IPH,#PT0H                ;

   SETB   ET0                   ; Enable Timer 0 interrupt
   SETB   EA                    ; Enable global interrupt

   SETB   TR0                   ; Start Timer 0 running

   ORL    PCON,#IDL                ; Set MCU into IDLE mode
   JMP    $
```

C Code Example:

```
#define  T0M0                   0x01
#define  T0M1                   0x02
#define  PT0                    0x02
#define  PT0H                   0x02
#define  IDL                    0x01

void time0_isr(void) interrupt 1
{
   To do…
}

void main(void)
{
   TH0 = TL0 = (256-100);           // Set Timer 0 overflow rate = SYSCLK x 100
   TMOD &= 0xF0;                 // Set Timer 0 to Mode 2
   TMOD |= T0M1;
   TF0 = 0;                      // Clear Timer 0 Flag

   IP |= PT0;                     // Select Timer 0 interrupt priority
   IPH |= PT0H;

   ET0 = 1;                      // Enable Timer 0 interrupt
   EA = 1;                       // Enable global interrupt

   TR0 = 1;                      // Start Timer 0 running

   PCON=IDL;                      // Set MCU into IDLE mode
   while(1);
}
```

(2). Required Function: Select Timer 0 clock source from SYSCLK (enable T0X12)

Assembly Code Example:

```
T0M0            EQU             01h
T0M1            EQU             02h
PT0             EQU             02h
PT0H            EQU             02h
T0X12           EQU             80h

   ORG   0000h
   JMP   main

   ORG   0000Bh
time0_isr:
   to do…
   RETI

main:
   ORL    AUXR, #T0X12            ; Select SYSCLK/1 for Timer 0 clock input
   CLR    TF0                     ; Clear Timer 0 Flag

   ORL    IP,#PT0                 ; Select Timer 0 interrupt priority
   ORL    IPH,#PT0H               ;

   SETB   ET0                     ; Enable Timer 0 interrupt
   SETB   EA                      ; Enable global interrupt

   MOV    TH0, #(256 - 240)       ;interrupt interval 20us
   MOV    TL0, #(256 - 240)       ;

   ANL    TMOD,#0F0h              ; Set Timer 0 to Mode 2
   ORL    TMOD,#T0M1              ;


   SETB   TR0                     ; Start Timer 0 running
   JMP    $
```

C Code Example:

```
#define  T0M0                    0x01
#define  T0M1                    0x02
#define  PT0                     0x02
#define  PT0H                    0x02
#define  T0X12                   0x80

   AUXR |= T0X12
   TF0 = 0;

   IP |= PT0;                    // Select Timer 0 interrupt priority
   IPH |= PT0H;

   ET0 = 1;                      // Enable Timer 0 interrupt
   EA = 1;                       // Enable global interrupt

   TH0 = TL0 = (256 - 240);

   TMOD &= 0xF0;                 // Set Timer 0 to Mode 2
   TMOD |= T0M1;

   TR0 = 1;                      // Start Timer 0 running
```

12. Serial Port 0 (UART0)

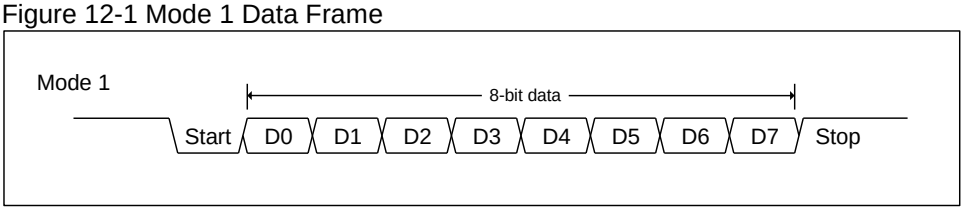
The serial port 0 of MG82Fx564 supports full-duplex transmission, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the register. However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost. The serial port receive and transmit registers are both accessed at special function register SBUF0. Writing to SBUF0 loads the transmit register, and reading from SBUF0 accesses a physically separate receive register.

The serial port can operate in 4 modes: Mode 0 provides *synchronous* communication while Modes 1, 2, and 3 provide *asynchronous* communication. The asynchronous communication operates as a full-duplex Universal Asynchronous Receiver and Transmitter (UART), which can transmit and receive simultaneously and at different baud rates.

Mode 0: 8 data bits (LSB first) are transmitted or received through RXD0(P3.0). TXD0(P3.1) always outputs the shift clock. The baud rate can be selected to 1/12 or 1/2 the system clock frequency by URM0X6 setting in SCFG register.

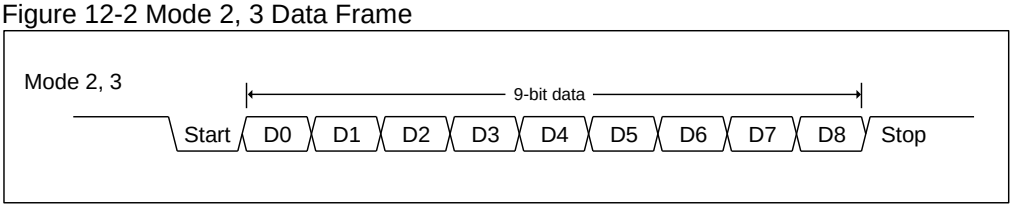
Mode 1: 10 bits are transmitted through TXD0 or received through RXD0. The frame data includes a start bit (0), 8 data bits (LSB first), and a stop bit (1), as shown in Figure 132-1 . On receive, the stop bit would be loaded into RB80 in SCON0 register. The baud rate is variable.

Figure 12-1 Mode 1 Data Frame



Mode 2: 11 bits are transmitted through TXD0 or received through RXD0. The frame data includes a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1), as shown in Figure 13-2. On Transmit, the 9th data bit comes from TB80 in SCON0 register can be assigned the value of 0 or 1. On receive, the 9th data bit would be loaded into RB80 in SCON0 register, while the stop bit is ignored. The baud rate can be configured to 1/32 or 1/64 the system clock frequency.

Figure 12-2 Mode 2, 3 Data Frame



Mode 3: Mode 3 is the same as Mode 2 except the baud rate is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF0 as a destination register. In Mode 0, reception is initiated by the condition RI0=0 and REN0=1. In the other modes, reception is initiated by the incoming start bit with 1-to-0 transition if REN0=1.

In addition to the standard operation, the UART0 can perform framing error detection by looking for missing stop bits, and automatic address recognition.

12.1. Serial Port 0 Mode 0

Serial data enters and exits through RXD0. TXD0 outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The shift clock source can be selected to 1/12 or 1/2 the system clock frequency by URM0X6 setting in SCFG register. Figure 13-3 shows a simplified functional diagram of the serial port 0 in Mode 0.

Transmission is initiated by any instruction that uses SBUF0 as a destination register. The “write to SBUF0” signal triggers the UART0 engine to start the transmission. The data in the SBUF0 would be shifted into the RXD0(P3.0) pin by each raising edge shift clock on the TXD0(P3.1) pin. After eight raising edge of shift clocks passing, TI would be asserted by hardware to indicate the end of transmission. Figure 13-4 shows the transmission waveform in Mode 0.

Reception is initiated by the condition REN0=1 and RI0=0. At the next instruction cycle, the Serial Port 0 Controller writes the bits 11111110 to the receive shift register, and in the next clock phase activates Receive.

Receive enables Shift Clock which directly comes from RX Clock to the alternate output function of P3.1 pin. When Receive is active, the contents on the RXD0(P3.0) pin would be sampled and shifted into shift register by falling edge of shift clock. After eight falling edge of shift clock, RI0 would be asserted by hardware to indicate the end of reception. Figure 13-5 shows the reception waveform in Mode 0.

Figure 12-3 Serial Port 0 Mode 0

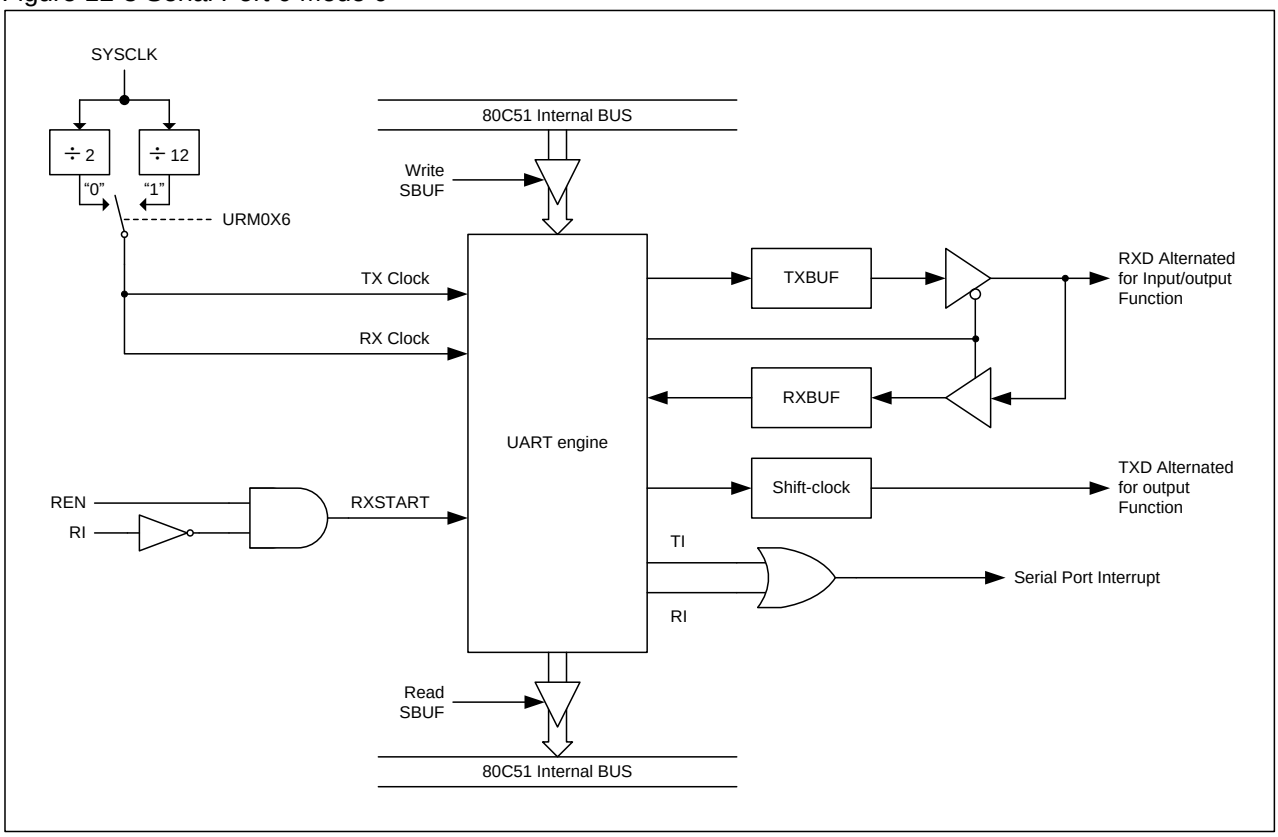


Figure 12-4 Mode 0 Transmission Waveform

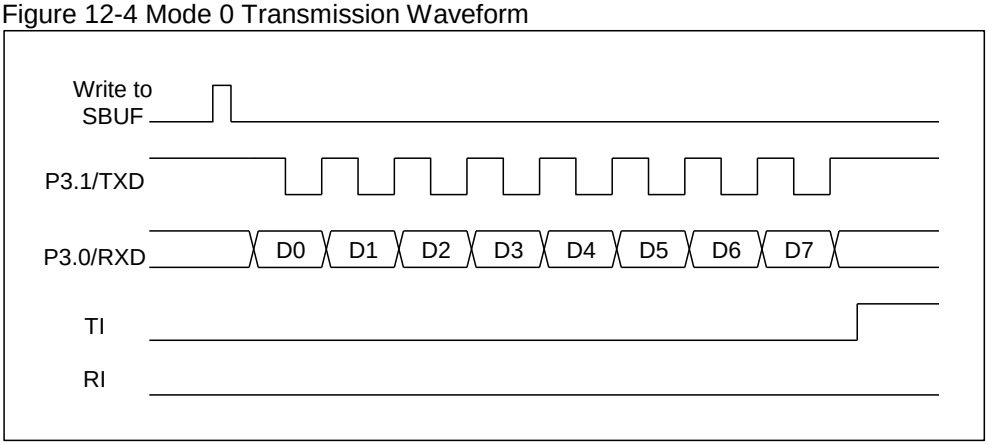
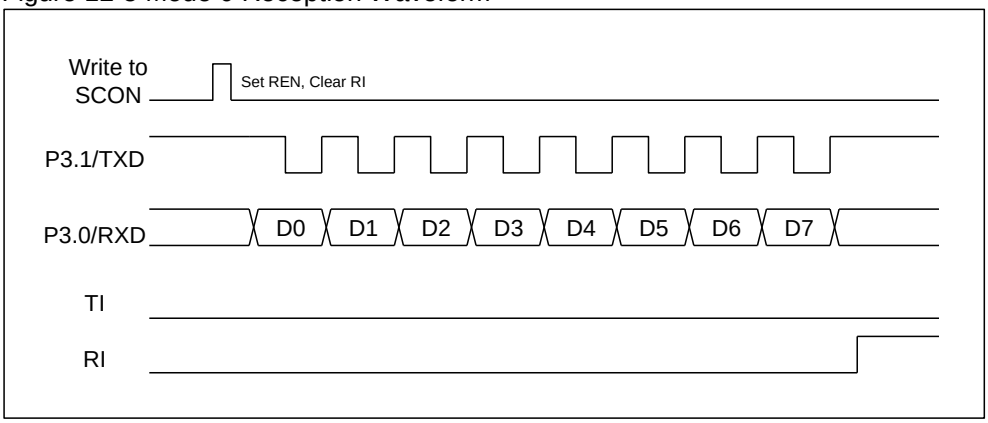


Figure 12-5 Mode 0 Reception Waveform



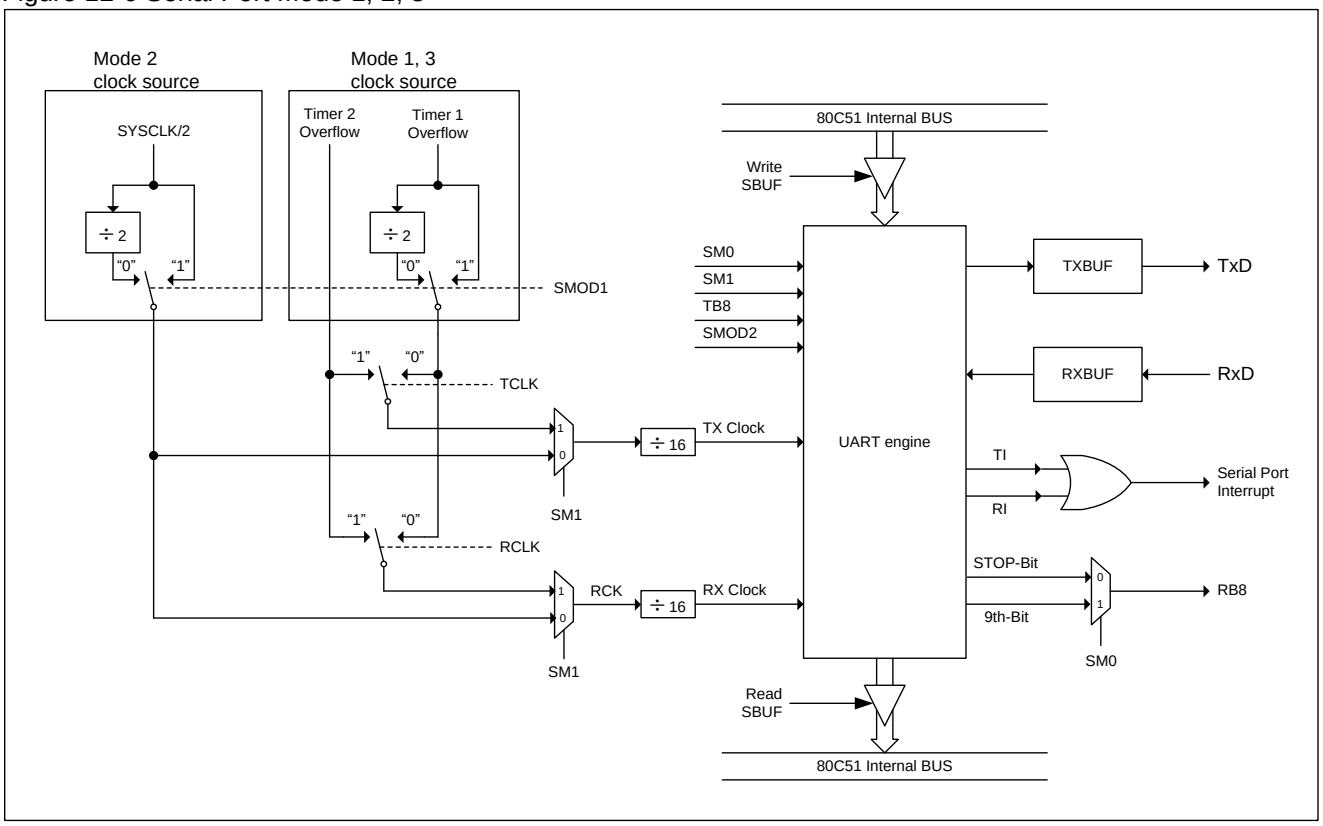
12.2. Serial Port 0 Mode 1

10 bits are transmitted through TXD0, or received through RXD0: a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB80 in SCON0. The baud rate is determined by the Timer 1 or Timer 2 overflow rate. Figure 132-1 shows the data frame in Mode 1 and Figure 13-6 shows a simplified functional diagram of the serial port in Mode 1.

Transmission is initiated by any instruction that uses SBUF0 as a destination register. The “write to SBUF0” signal requests the UART0 engine to start the transmission. After receiving a transmission request, the UART0 engine would start the transmission at the raising edge of TX Clock. The data in the SBUF0 would be serial output on the TXD0 pin with the data frame as shown in Figure 132-1 and data width depend on TX Clock. After the end of 8th data transmission, TI0 would be asserted by hardware to indicate the end of data transmission.

Reception is initiated when Serial Port 0 Controller detected 1-to-0 transition at RXD0 sampled by RCK. The data on the RXD0 pin would be sampled by Bit Detector in Serial Port 0 Controller. After the end of STOP-bit reception, RI0 would be asserted by hardware to indicate the end of data reception and load STOP-bit into RB80 in SCON0 register.

Figure 12-6 Serial Port Mode 1, 2, 3



12.3. Serial Port 0 Mode 2 and Mode 3

11 bits are transmitted through TXD0, or received through RXD0: a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB80) can be assigned the value of 0 or 1. On receive, the 9th data bit goes into RB80 in SCON0. The baud rate is programmable to select one of 1/16, 1/32 or 1/64 the system clock frequency in Mode 2. Mode 3 may have a variable baud rate generated from Timer 1 or Timer 2.

Figure 13-2 shows the data frame in Mode 2 and Mode 3. Figure 13-6 shows a functional diagram of the serial port in Mode 2 and Mode 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

The “write to SBUF0” signal requests the Serial Port 0 Controller to load TB80 into the 9th bit position of the transmit shit register and starts the transmission. After receiving a transmission request, the UART0 engine would start the transmission at the raising edge of TX Clock. The data in the SBUF0 would be serial output on the TXD0 pin with the data frame as shown in Figure 13-2 and data width depend on TX Clock. After the end of 9th data transmission, TI0 would be asserted by hardware to indicate the end of data transmission.

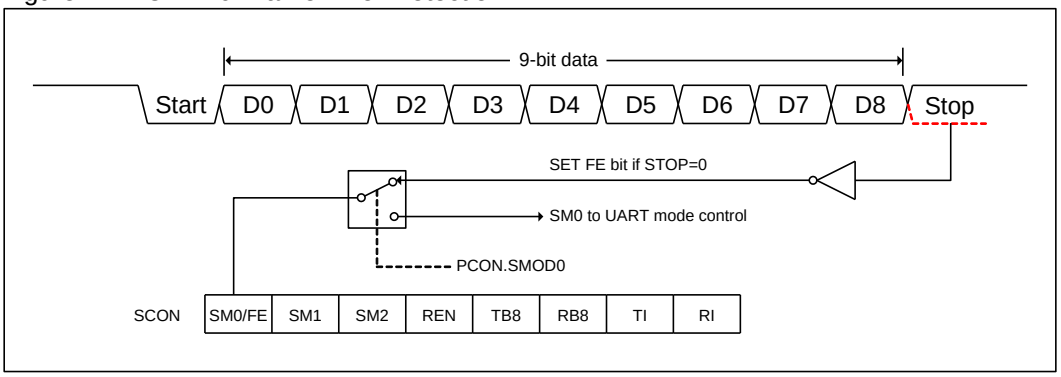
Reception is initiated when the UART0 engine detected 1-to-0 transition at RXD0 sampled by RCK. The data on the RXD0 pin would be sampled by Bit Detector in UART0 engine. After the end of 9th data bit reception, RI0 would be asserted by hardware to indicate the end of data reception and load the 9th data bit into RB80 in SCON0 register.

In all four modes, transmission is initiated by any instruction that use SBUF0 as a destination register. Reception is initiated in mode 0 by the condition RI0 = 0 and REN0 = 1. Reception is initiated in the other modes by the incoming start bit with 1-to-0 transition if REN0=1.

12.4. Frame Error Detection

When used for framing error detection, the UART0 looks for missing stop bits in the communication. A missing stop bit will set the FE bit in the SCON0 register. The FE bit shares the SCON0.7 bit with SM00 and the function of SCON0.7 is determined by SMOD0 bit (PCON.6). If SMOD0 is set then SCON0.7 functions as FE. SCON0.7 functions as SM00 when SMOD0 is cleared. When SCON0.7 functions as FE, it can only be cleared by firmware. Refer to Figure 13-7.

Figure 12-7 UART0 Frame Error Detection



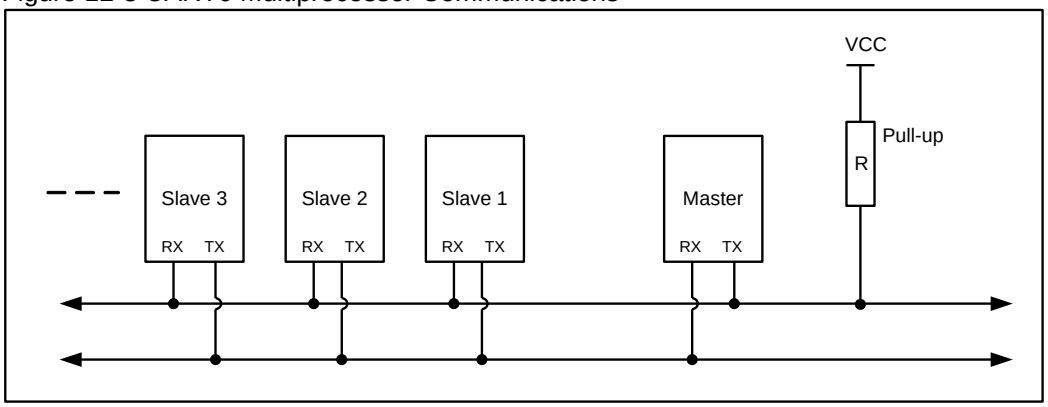
12.5. Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications as shown in Figure 13-8. In these two modes, 9 data bits are received. The 9th bit goes into RB80. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB80=1. This feature is enabled by setting bit SM20 (in SCON0 register). A way to use this feature in multiprocessor systems is as follows:

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM20=1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and check if it is being addressed. The addressed slave will clear its SM20 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM20 set and go on about their business, ignoring the coming data bytes.

SM20 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM20=1, the receive interrupt will not be activated unless a valid stop bit is received.

Figure 12-8 UART0 Multiprocessor Communications



12.6. Automatic Address Recognition

Automatic Address Recognition is a feature which allows the UART0 to recognize certain addresses in the serial bit stream by using hardware to make the comparisons. This feature saves a great deal of firmware overhead by eliminating the need for the firmware to examine every serial address which passes by the serial port. This feature is enabled by setting the SM20 bit in SCON0.

In the 9 bit UART modes, mode 2 and mode 3, the Receive Interrupt flag (RI0) 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 Figure 13-9. The 8 bit mode is called Mode 1. In this mode the RI flag will be set if SM20 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 SM20 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	Slave 1
SADDR = 1100 0000	SADDR = 1100 0000
SADEN = 1111 1101	SADEN = 1111 1110
Given = 1100 00X0	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:

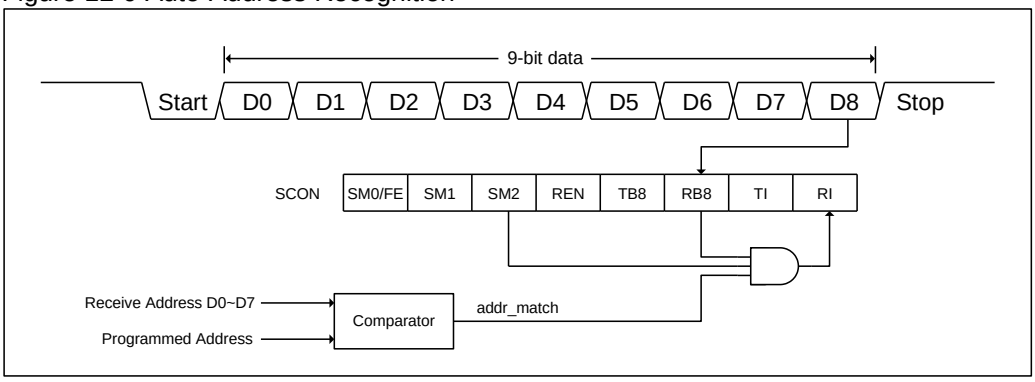
Slave 0	Slave 1	Slave 2
SADDR = 1100 0000	SADDR = 1110 0000	SADDR = 1110 0000
SADEN = 1111 1001	SADEN = 1111 1010	SADEN = 1111 1100
Given = 1100 0XX0	Given = 1110 0X0X	Given = 1110 00XX

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 treated 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 0xA9) and SADEN (SFR address 0xB9) are loaded 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.

Figure 12-9 Auto-Address Recognition



Note: (1) After address matching(addr_match=1), Clear SM20 to receive data bytes
(2) After all data bytes have been received, Set SM20 to wait for next address.

12.7. Baud Rate Setting

Bits AUXR2.T1X12, URM0X6 and SMOD2 in SCFG register provide a new option for the baud rate setting, as listed below.

12.7.1. Baud Rate in Mode 0

$$\text{Mode 0 Baud Rate} = \frac{F_{SYSCLK}}{n} \quad \begin{array}{l} ; \text{n=12, if URM0X6=0} \\ ; \text{n=2, if URM0X6=1} \end{array}$$

Note:

If URM0X6=0, the baud rate formula is as same as standard 8051.

12.7.2. Baud Rate in Mode 2

$$\text{Mode 2 Baud Rate} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{64} \times F_{SYSCLK}$$

Note:

If SMOD2=0, the baud rate formula is as same as standard 8051. If SMOD2=1, there is an enhanced function for baud rate setting. Following table defines the Baud Rate setting with SMOD2 factor in Mode 2 baud rate generator.

SMOD2	SMOD1	Baud Rate	Note
0	0	Default Baud Rate	Standard function
0	1	Double Baud Rate	Standard function
1	0	Double Baud Rate X2	Enhanced function
1	1	Double Baud Rate X4	Enhanced function

12.7.3. Baud Rate in Mode 1 & 3

Using Timer 1 as the Baud Rate Generator

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{32} \times \frac{F_{SYSCLK}}{12 \times (256 - TH1)} \; ; \text{T1X12=0}$$

$$\text{or} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{32} \times \frac{F_{SYSCLK}}{1 \times (256 - TH1)} \; ; \text{T1X12=1}$$

Note:

If SMOD2=0, T1X12=0, the baud rate formula is as same as standard 8051. If SMOD2=1, there is an enhanced function for baud rate setting. Following table defines the Baud Rate setting with SMOD2 factor in Timer 1 baud rate generator.

SMOD2	SMOD1	Baud Rate	Note
0	0	Default Baud Rate	Standard function
0	1	Double Baud Rate	Standard function
1	0	Double Baud Rate X2	Enhanced function
1	1	Double Baud Rate X4	Enhanced function

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.

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{(SMOD2 + 1) \times SMOD1} \times F_{SYSCLK}}{32 \times (65536 - (RCAP2H, RCAP2L))}$$

Note:

If SMOD2=0, the baud rate formula is as same as standard 8051. If SMOD2=1, there is an enhanced function for baud rate setting. Following table defines the Baud Rate setting with SMOD2 factor in Timer 2 baud rate generator.

SMOD2	SMOD1	Baud Rate	Note
0	X	Default Baud Rate	Standard function
1	0	Double Baud Rate	Enhanced function
1	1	Double Baud Rate X2	Enhanced function

12.8. Serial Port 0 Register

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, AUXR2 and SCFG, are related to the baud rate setting:

SCON0: Serial port 0 Control Register

SFR Page = **0 only**

SFR Address = 0x98 RESET = 0000-0000

7	6	5	4	3	2	1	0
SM00/FE	SM10	SM20	REN0	TB80	RB80	TI0	RI0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: FE, Framing Error bit. The SMOD0 bit must be set to enable access to the FE bit.
0: The FE bit is not cleared by valid frames but should be cleared by software.
1: This bit is set by the receiver when an invalid stop bit is detected.

Bit 7: Serial port 0 mode bit 0, (SMOD0 must = 0 to access bit SM00)
Bit 6: Serial port 0 mode bit 1.

SM00	SM10	Mode	Description	Baud Rate
0	0	0	shift register	SYSCLK/12 or /2
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	SYSCLK/64, /32, /16 or /8
1	1	3	9-bit UART	variable

Bit 5: Serial port 0 mode bit 2.
0: Disable SM20 function.
1: Enable the automatic address recognition feature in Modes 2 and 3. If SM20=1, RI0 will not be set unless the received 9th data bit is 1, indicating an address, and the received byte is a Given or Broadcast address. In mode1, if SM20=1 then RI0 will not be set unless a valid stop Bit was received, and the received byte is a Given or Broadcast address. In Mode 0, SM20 should be 0.

Bit 4: REN0, Enable serial reception.
0: Clear by software to disable reception.
1: Set by software to enable reception.

Bit 3: TB80, The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.

Bit 2: RB80, In Modes 2 and 3, the 9th data bit that was received. In Mode 1, if SM20 = 0, RB80 is the stop bit that was received. In Mode 0, RB80 is not used.

Bit 1: TI0. Transmit interrupt flag.
0: Must be cleared by software.
1: Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission.

Bit 0: RI0. Receive interrupt flag.
0: Must be cleared by software.
1: Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM20).

SBUF0: Serial port 0 Buffer Register

SFR Page = **0 only**

SFR Address = 0x99 RESET = XXXX-XXXX

7	6	5	4	3	2	1	0
SBUF0[7]	SBUF0[6]	SBUF0[5]	SBUF0[4]	SBUF0[3]	SBUF0[2]	SBUF0[1]	SBUF0[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: It is used as the buffer register in transmission and reception.

SADDR: Slave Address Register

SFR Page = All

SFR Address = 0xA9 RESET = 0000-0000

7	6	5	4	3	2	1	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

SADEN: Slave Address Mask Register

SFR Page = All

SFR Address = 0xB9 RESET = 0000-0000

7	6	5	4	3	2	1	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

SADDR register is combined with SADEN register to form Given/Broadcast Address for automatic address recognition. In fact, SADEN functions as the “mask” register for SADDR register. The following is the example for it.

SADDR = 1100 0000
SADEN = 1111 1101
Given = 1100 00x0 ⟶ The Given slave address will be checked except bit 1 is treated as “don’t care”

The Broadcast Address for each slave is created by taking the logical OR of SADDR and SADEN. Zero in this result is considered as “don’t care”. Upon reset, SADDR and SADEN are loaded with all 0s. This produces a Given Address of all “don’t care” and a Broadcast Address of all “don’t care”. This disables the automatic address detection feature.

PCON0: Power Control Register 0

SFR Page = All

SFR Address = 0x87 RESET = 00X1-0000

7	6	5	4	3	2	1	0
SMOD1	SMOD0	--	POF	GF1	GF0	PD	IDL
R/W	**R/W**	**R**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: SMOD1, double Baud rate control bit.
0: Disable double Baud rate of the UART.
1: Enable double Baud rate of the UART in mode 1, 2, or 3.

Bit 6: SMOD0, Frame Error select.
0: SCON.7 is SM0 function.
1: SCON.7 is FE function. Note that FE will be set after a frame error regardless of the state of SMOD0.

SCFG: Serial Port Configuration Register

SFR Page = **0 only**

SFR Address = 0x9A RESET = 0000-00XX

7	6	5	4	3	2	1	0
URTS	SMOD2	URM0X6	S1TR	S1MOD	S1TX12	--	--
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R**	**R**

Bit 7: URTS, UART0 Timer Selection.
0: Timer 1 or Timer 2 can be used as the Baud Rate Generator in Mode 1 and Mode 3.
1: Timer 1 overflow signal is replaced by the UART1 Baud Rate Timer overflow signal when Timer 1 is selected as the Baud Rate Generator in Mode1 or Mode 3 of the UART0. (Refer to Section 12-2.)

Bit 6: SMOD2, extra double baud rate selector.
0: Disable extra double baud rate for UART0.
1: Enable extra double baud rate for UART0.

Bit 5: URM0X6, Serial Port mode 0 baud rate selector.
0: Clear to select SYSCLK/12 as the baud rate for UART Mode 0.
1: Set to select SYSCLK/2 as the baud rate for UART Mode 0.

Bit 1~0: Reserved. Software must write “0”s on these bits when SCFG is written.

AUXR2: Auxiliary Register 2

SFR Page = All
SFR Address = 0x87 POR+RESET = 00XX-XX00

7	6	5	4	3	2	1	0
T0X12	T1X12	--	--	--	--	T1CKOE	T0CKOE
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R/W**	**R/W**

Bit 6: T1X12, Timer 1 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source. If set, the UART0 baud rate by Timer 1 in Mode 1 and Mode 3 is 12 times than standard 8051 function.

13. Serial Port 1 (UART1)

The MG82Fx564 is equipped with a secondary UART (hereafter, called UART1), which also has four operation modes the same as the first UART except the following differences:

(1) The UART1 has no enhanced functions: Framing Error Detection and Auto Address Recognition.
(2) The UART1 use the dedicated Baud Rate Timer as its Baud Rate Generator.
(3) The UART1 uses port pin P1.3 (TXD1) and P1.2 (RXD1) for transmit and receive, respectively.

These two UARTs can be operated simultaneously in identical or different modes and communication speeds.

13.1. Serial Port 1 Baud Rates

13.1.1. Baud Rate in Mode 0

$$\text{S1 Mode 0 Baud Rate} = \frac{F_{SYSCLK}}{12}$$

Note:
If URM0X6=0, the baud rate formula is as same as standard 8051.

13.1.2. Baud Rate in Mode 2

$$\text{S1 Mode 2 Baud Rate} = \frac{2^{S1MOD1}}{64} \times F_{SYSCLK}$$

13.1.3. Baud Rate in Mode 2

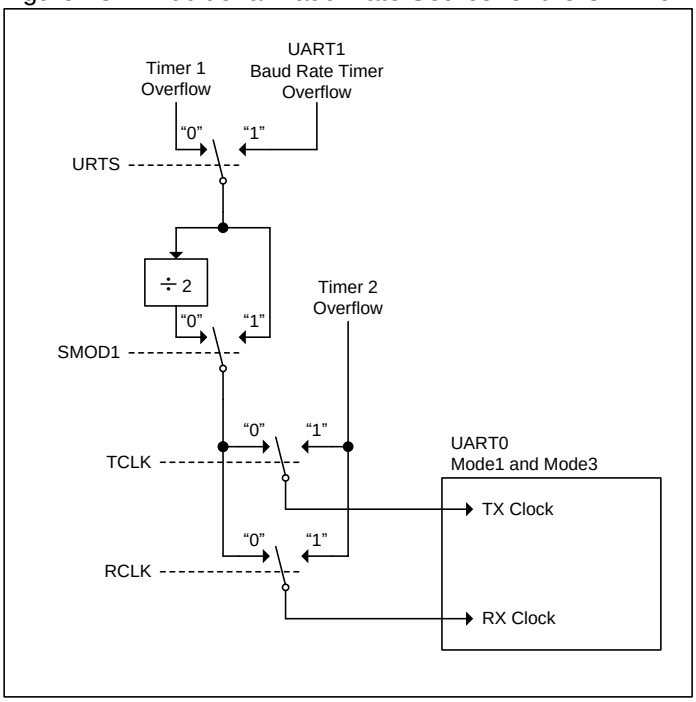
$$\text{S1 Mode 1, 3 Baud Rate} = \frac{2^{S1MOD1}}{32} \times \frac{F_{SYSCLK}}{12 \times (256 - S1BRT)} \quad ; S1X12=0$$

$$\text{or} = \frac{2^{S1MOD1}}{32} \times \frac{F_{SYSCLK}}{1 \times (256 - S1BRT)} \quad ; S1X12=1$$

13.2. UART1 Baud Rate Timer used for UART0

In the Mode 1 and Mode 3 operation of the UART0, the user can select Timer 1 as the Baud Rate Generator by clearing bits TCLK and RCLK in T2CON register. At this time, if URTS bit (in SCFG register) is set, then Timer 1 overflow signal will be replaced by the overflow signal of the UART1 Baud Rate Timer. In other words, the user can adopt UART1 Baud Rate Timer as the Baud Rate Generator for Mode 1 or Mode 3 of the UART0 as long as RCLK=0, TCLK=0 and URTS=1. In this condition, Timer 1 is free for other application. Of course, if UART1 (Mode 1 or Mode 3) is also operated at this time, these two UARTs will have the same baud rates.

Figure 13-1. Additional Baud Rate Source for the UART0



13.3. Serial Port 1 Register

The following special function registers are related to the operation of the UART1:

SCON1: Serial port 1 Control Register

SFR Page = **1 only**

SFR Address = 0x98 POR+RESET = 0000-0000

7	6	5	4	3	2	1	0
SM01	SM11	SM21	REN1	TB81	RB81	TI1	RI1
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: SM01, Serial port 1 mode bit 0.
Bit 6: SM11, Serial port 1 mode bit 1.

SM01	SM11	Mode	Description	Baud Rate
0	0	0	shift register	SYSCLK/12
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	SYSCLK/64, /32
1	1	3	9-bit UART	variable

Bit 5: Serial port 0 mode bit 2.
0: Disable SM21 function.
1: Enable the automatic address recognition feature in Modes 2 and 3. If SM21=1, RI1 will not be set unless the received 9th data bit is 1, indicating an address, and the received byte is a Given or Broadcast address. In mode1, if SM21=1 then RI1 will not be set unless a valid stop Bit was received, and the received byte is a Given or Broadcast address. In Mode 0, SM21 should be 0.

Bit 4: REN1, Enable serial reception.
0: Clear by software to disable reception.
1: Set by software to enable reception.

Bit 3: TB81, The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.

Bit 2: RB81, In Modes 2 and 3, the 9th data bit that was received. In Mode 1, if SM21 = 0, RB81 is the stop bit that was received. In Mode 0, RB81 is not used.

Bit 1: TI1. Transmit interrupt flag.
0: Must be cleared by software.
1: Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission.

Bit 0: RI1. Receive interrupt flag.
0: Must be cleared by software.
1: Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM21).

SBUF1: Serial port 1 Buffer Register

SFR Page = **1 only**
SFR Address = 0x99 POR+RESET = XXXX-XXXX

7	6	5	4	3	2	1	0
SBUF1[7]	SBUF1[6]	SBUF1[5]	SBUF1[4]	SBUF1[3]	SBUF1[2]	SBUF1[1]	SBUF1[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: It is used as the buffer register in transmission and reception.

S1BRT: Serial port 1 Baud Rate Timer Reload Register

SFR Page = **1 only**
SFR Address = 0x9A POR+RESET = 0000-0000

7	6	5	4	3	2	1	0
S1BRT[7]	S1BRT[6]	S1BRT[5]	S1BRT[4]	S1BRT[3]	S1BRT[2]	S1BRT[1]	S1BRT[0]
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~0: It is used as the reload value register for baud rate timer generator that works in a similar manner as Timer 1.

SCFG: Serial Port Configuration Register

SFR Page = **0 only**
SFR Address = 0x9A POR+RESET = 0000-00XX

7	6	5	4	3	2	1	0
URTS	SMOD2	URM0X6	S1TR	S1MOD	S1TX12	--	--
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R**	**R**

Bit 7: UART0 Timer Selection.
0: Timer 1 or Timer 2 can be used as the Baud Rate Generator in Mode 1 and Mode 3.
1: Timer 1 overflow signal is replaced by the UART1 Baud Rate Timer overflow signal when Timer 1 is selected as the Baud Rate Generator in Mode1 or Mode 3 of the UART0. (Refer to Section 12-2.)

Bit 4: S1TR, UART1 Baud Rate Timer control bit.
0: Clear to turn off the S1BRT.
1: Set to turn on S1BRT.

Bit 3: S1SMOD, UART1 double baud rate enable bit.
0: Disable the double baud rate function for UART1.
1: Enable the double baud rate function for UART1.

Bit 2: S1TX12, UART1 Baud Rate Timer clock source select
0: Clear to select SYSCLK/12 as the clock source for S1BRT.
1: Set to select SYSCLK as the clock source for S1BRT.

Bit 1~0: Reserved. Software must write “0”s on these bits when SCFG is written.

13.4. Serial Port Sample Code

(1). Required Function: IDLE mode with RI wake-up capability

Assembly Code Example:

```
PS                  EQU             10h
PSH                 EQU             10h


   ORG    00023h
uart_ri_idle_isr:
   JB     RI,RI_ISR            ;
   JB     TI,TI_ISR            ;
   RETI                        ;

RI_ISR:
;  Process
   CLR    RI                   ;
   RETI                        ;

TI_ISR:
;  Process
   CLR    TI                   ;
   RETI                        ;

main:
   CLR    TI                   ;
   CLR    RI                   ;
   SETB   SM1                  ;
   SETB   REN                  ; 8bit Mode2, Receive Enable

   CALL   UART_Baud_Rate_Setting      ;

   MOV    IP,#PSL              ; Select UART interrupt priority
   MOV    IPH,#PSH             ;

   SETB   ES                   ; Enable S0 interrupt
   SETB   EA                   ; Enable global interrupt

   ORL    PCON,#IDL;           ; Set MCU into IDLE mode
```

C Code Example:

```
#define  PS                         0x10
#define  PSH                        0x10

void uart_ri_idle_isr(void) interrupt 4
{  if(RI)
   {
      RI=0;
      // to do ...
   }

   if(TI)
   {
      TI=0;
      // to do ...
   }
}

void main(void)
{
   TI = RI = 0;
   SM1 = REN = 1;                  // 8bit Mode2, Receive Enable

   UART_Baud_Rate_Setting()        //

   IP = PSL;                       // Select S0 interrupt priority
   IPH = PSH;                  //
```

```
    ES = 1;                         // Enable S0 interrupt
    EA = 1;                         // Enable global interrupt

    PCON |= IDL;                    // Set MCU into IDLE mode
}
```

14. Programmable Counter Array (PCA)

The MG82Fx564 is equipped with a Programmable Counter Array (PCA), which provides more timing capabilities with less CPU intervention than the standard timer/counters. Its advantages include reduced software overhead and improved accuracy.

14.1. PCA Overview

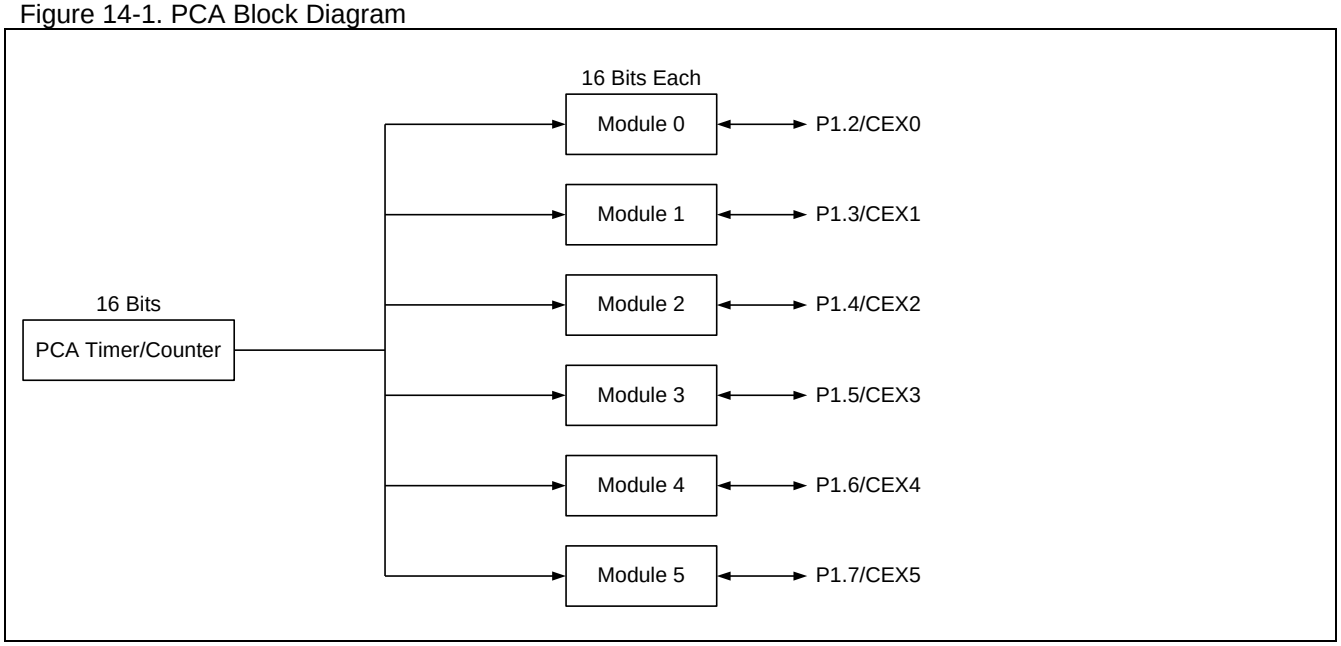
The PCA consists of a dedicated timer/counter which serves as the time base for an array of six compare/capture modules. Figure 14-1 shows a block diagram of the PCA. Notice that the PCA timer and modules are all 16-bits. If an external event is associated with a module, that function is shared with the corresponding Port 1 pin. If the module is not using the port pin, the pin can still be used for standard I/O.

Each of the six modules can be programmed in any one of the following modes:

- Rising and/or Falling Edge Capture
- Software Timer
- High Speed Output
- Pulse Width Modulator (PWM) Output

All of these modes will be discussed later in detail. However, let's first look at how to set up the PCA timer and modules.

Figure 14-1. PCA Block Diagram



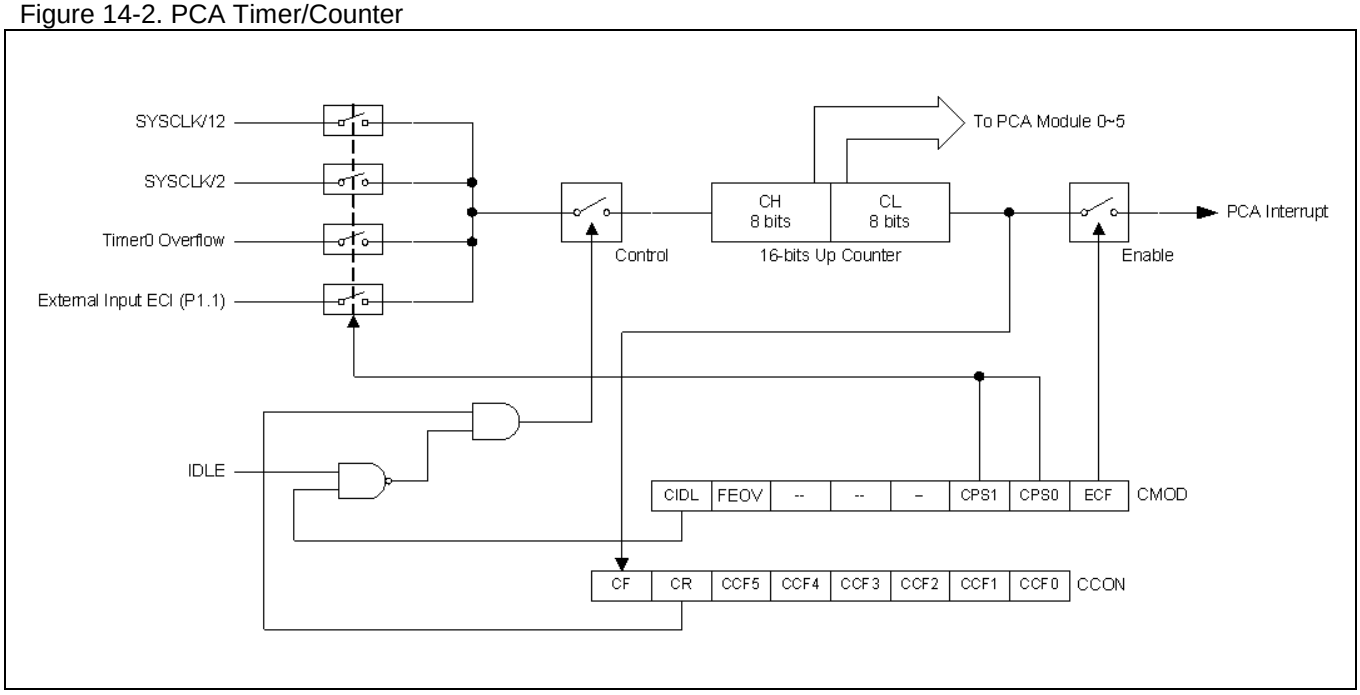
14.2. PCA Timer/Counter

The timer/counter for the PCA is a free-running 16-bit timer consisting of registers CH and CL (the high and low bytes of the count values), as shown in Figure 14-2. It is the common time base for all modules and its clock input can be selected from the following source:

- 1/12 the system clock frequency,
- 1/2 the system clock frequency,
- the Timer 0 overflow, which allows for a range of slower clock inputs to the timer.
- external clock input, 1-to-0 transitions, on ECI pin (P1.1).

Special Function Register CMOD contains the Count Pulse Select bits (CPS1 and CPS0) to specify the PCA timer input. This register also contains the ECF bit which enables an interrupt when the counter overflows. In addition, the user has the option of turning off the PCA timer during Idle Mode by setting the Counter Idle bit (CIDL). This can further reduce power consumption during Idle mode.

Figure 14-2. PCA Timer/Counter



CMOD: PCA Counter Mode Register

SFR Page = All

SFR Address = 0xD9 RESET = 0xxx-x000

7	6	5	4	3	2	1	0
CIDL	FEOV	--	--	--	CPS1	CPS0	ECF
R/W	**R/W**	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**

Bit 7: CIDL, PCA counter Idle control.
0: Lets the PCA counter continue functioning during Idle mode.
1: Lets the PCA counter be gated off during Idle mode.

Bit 6: FEOV, Maximum Counter {CL} value on FE.
FEOV=0 Maximum CL counter value on FF.
FEOV=1 Maximum CL counter value on FE.

Bit 5~3: Reserved. Software must write “0”s on these bits when CMOD is written.

Bit 2~1: CPS1-CPS0, PCA counter clock source select bits.

CPS1	CPS0	PCA Clock Source
0	0	Internal clock, (system clock)/12
0	1	Internal clock, (system clock)/2
1	0	Timer 0 overflow
1	1	External clock at the ECI pin

Bit 0: ECF, Enable PCA counter overflow interrupt.
0: Disables an interrupt when CF bit (in CCON register) is set.
1: Enables an interrupt when CF bit (in CCON register) is set.

The CCON register shown below contains the run control bit for the PCA and the flags for the PCA timer and each module. To run the PCA the CR bit (CCON.6) must be set by software. The PCA is shut off by clearing this bit. The CF bit (CCON.7) is set when the PCA counter overflows and an interrupt will be generated if the ECF bit in the CMOD register is set. The CF bit can only be cleared by software. CCF0 to CCF5 are the interrupt flags for module 0 to module 5, respectively, and they are set by hardware when either a match or a capture occurs. These flags also can only be cleared by software. The PCA interrupt system is shown Figure 14-3.

CCON: PCA Counter Control Register

SFR Page = All
SFR Address = 0xD8 RESET = 0000-0000

7	6	5	4	3	2	1	0
CF	CR	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: CF, PCA Counter Overflow flag.
0: Only be cleared by software.
1: Set by hardware when the counter rolls over. CF flag can generate an interrupt if bit ECF in CMOD is set. CF may be set by either hardware or software.

Bit 6: CR, PCA Counter Run control bit.
0: Must be cleared by software to turn the PCA counter off.
1: Set by software to turn the PCA counter on.

Bit 5: CCF5, PCA Module 5 interrupt flag.
0: Must be cleared by software.
1: Set by hardware when a match or capture occurs.

Bit 4: CCF4, PCA Module 4 interrupt flag.
0: Must be cleared by software.
1: Set by hardware when a match or capture occurs.

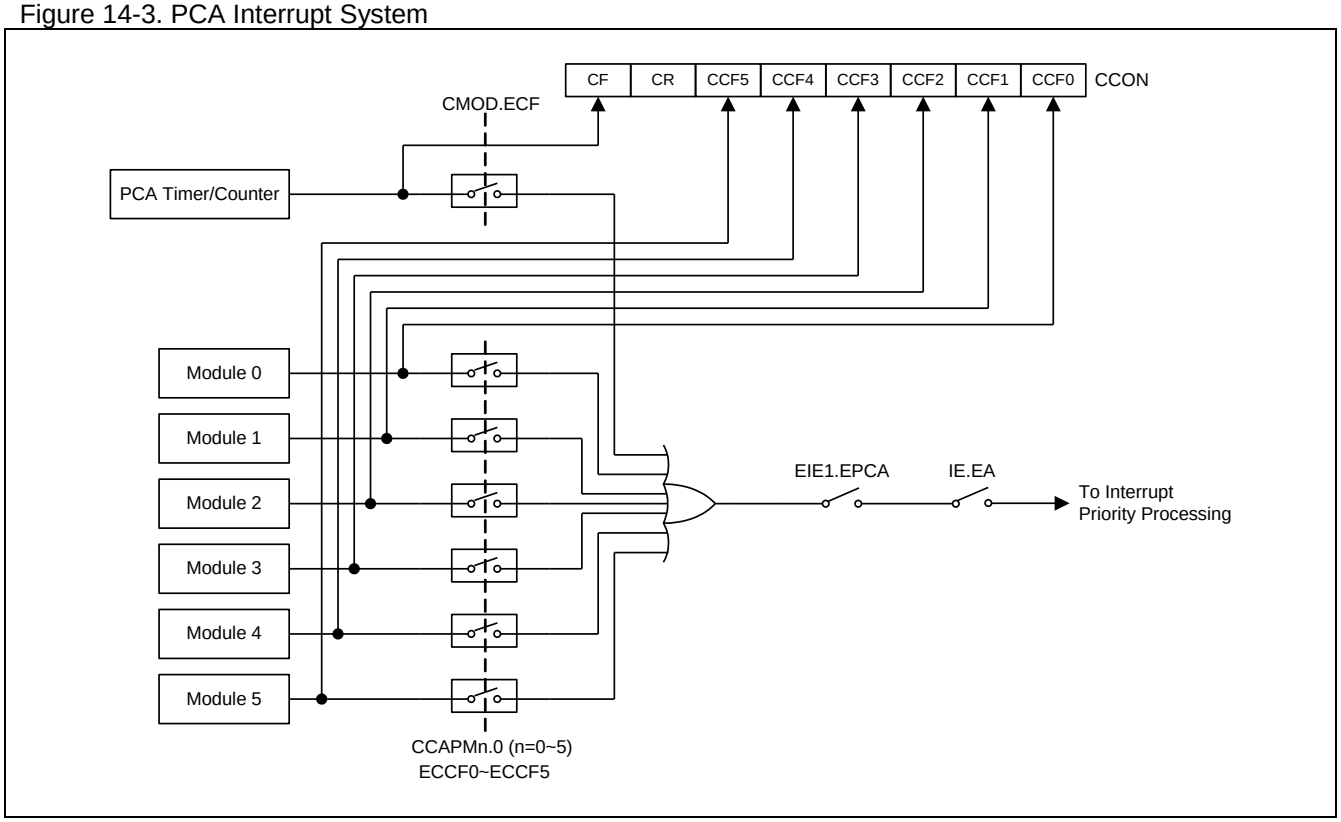
Bit 3: CCF3, PCA Module 3 interrupt flag.
0: Must be cleared by software.
1: Set by hardware when a match or capture occurs.

Bit 2: CCF2, PCA Module 2 interrupt flag.
0: Must be cleared by software.
1: Set by hardware when a match or capture occurs.

Bit 1: CCF1, PCA Module 1 interrupt flag.
0: Must be cleared by software.
1: Set by hardware when a match or capture occurs.

Bit 0: CCF0, PCA Module 0 interrupt flag.
0: Must be cleared by software.
1: Set by hardware when a match or capture occurs.

Figure 14-3. PCA Interrupt System



14.3. Compare/Capture Modules

Each of the six compare/capture modules has a mode register called CCAPMn (n e 0,1,2,3,or 4) to select which function it will perform. Note the ECCFn bit which enables an interrupt to occur when a module's interrupt flag is set.

CCAPMn: PCA Module Compare/Capture Register, n=0~5

SFR Page = All

SFR Address = 0xDA~0xDF RESET = x000-0000

7	6	5	4	3	2	1	0
--	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn
R	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: Reserved. Software must write “0” on this bit when the CCAPMn is written.

Bit 6: ECOMn, Enable Comparator
0: Disable the digital comparator function.
1: Enables the digital comparator function.

Bit 5: CAPPn, Capture Positive enabled.
0: Disable the PCA capture function on CEXn positive edge detected.
1: Enable the PCA capture function on CEXn positive edge detected.

Bit 4: CAPNn, Capture Negative enabled.
0: Disable the PCA capture function on CEXn positive edge detected.
1: Enable the PCA capture function on CEXn negative edge detected.

Bit 3: MATn, Match control.
0: Disable the digital comparator match event to set CCFn.
1: A match of the PCA counter with this module’s compare/capture register causes the CCFn bit in CCON to be set.

Bit 2: TOGn, Toggle control.
0: Disable the digital comparator match event to toggle CEXn.
1: A match of the PCA counter with this module’s compare/capture register causes the CEXn pin to toggle.

Bit 1: PWMn, PWM control.
0: Disable the PWM mode in PCA module.
1: Enable the PWM function and cause CEXn pin to be used as a pulse width modulated output.

Bit 0: ECCFn, Enable CCFn interrupt.
0: Disable compare/capture flag CCFn in the CCON register to generate an interrupt.
1: Enable compare/capture flag CCFn in the CCON register to generate an interrupt.

Note: The bits CAPNn (CCAPMn.4) and CAPPn (CCAPMn.5) determine the edge on which a capture input will be active. If both bits are set, both edges will be enabled and a capture will occur for either transition.

Each module also has a pair of 8-bit compare/capture registers (CCAPnH, CCAPnL) associated with it. These registers are used to store the time when a capture event occurred or when a compare event should occur.
When a module is used in the PWM mode, in addition to the above two registers, an extended register PCAPWMn is used to improve the range of the duty cycle of the output. The improved range of the duty cycle starts from 0%, up to 100%, with a step of 1/256.

PCAPWMn: PWM Mode Auxiliary Register, n=0~5

SFR Page = All

SFR Address = 0xF2~0xF7 RESET = 0000-0000

7	6	5	4	3	2	1	0
PnRS1	**PnRS0**	**PnPS2**	**PnPS1**	**PnPS0**	**PnINV**	ECAPnH	ECAPnL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

ECAPnH: Extended 9th bit (MSB bit), associated with CCAPnH to become a 9-bit register used in PWM mode.
ECAPnL: Extended 9th bit (MSB bit), associated with CCAPnL to become a 9-bit register used in PWM mode.

14.4. Operation Modes of the PCA

Table 14-1 shows the CCAPMn register settings for the various PCA functions.

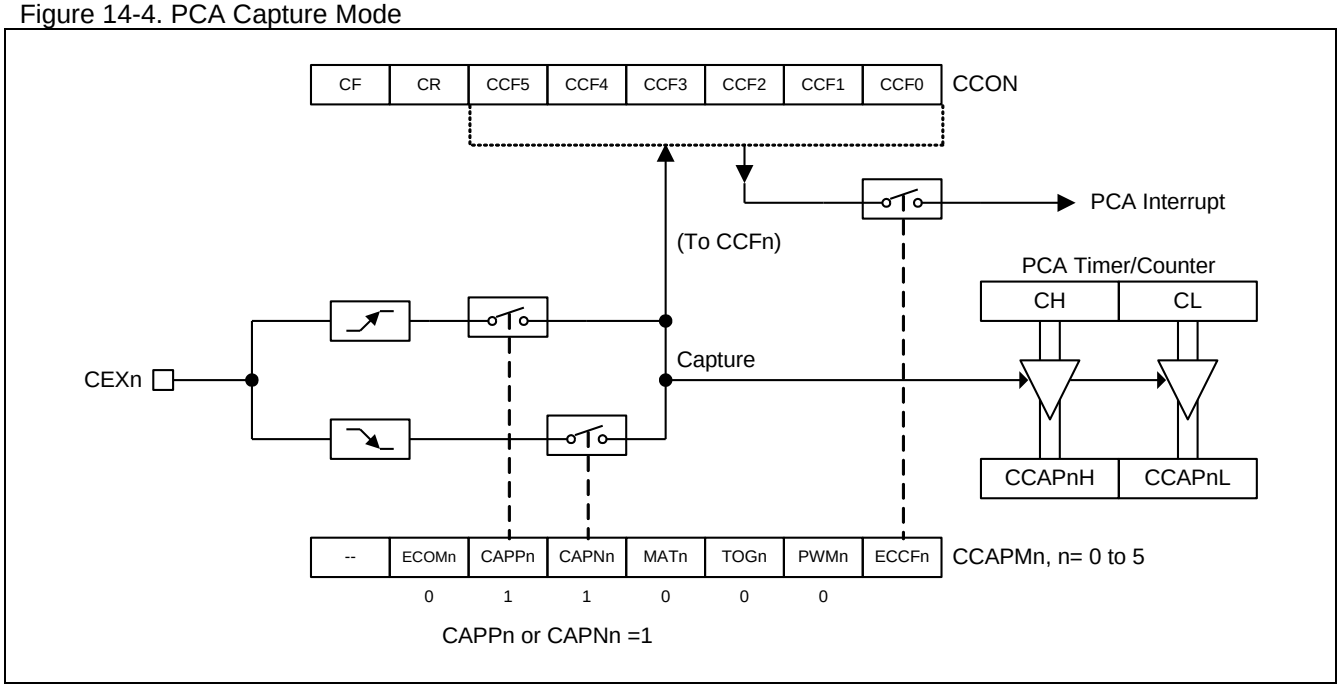
Table 14-1. PCA Module Modes

ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	Module Function
0	0	0	0	0	0	0	No operation
X	1	0	0	0	0	X	16-bit capture by a positive-edge trigger on CEXn
X	0	1	0	0	0	X	16-bit capture by a negative-edge trigger on CEXn
X	1	1	0	0	0	X	16-bit capture by a transition on CEXn
1	0	0	1	0	0	X	16-bit Software Timer
1	0	0	1	1	0	X	16-bit High Speed Output
1	0	0	0	0	1	0	8-bit Pulse Width Modulator (PWM)

14.4.1. Capture Mode

To use one of the PCA modules in the capture mode, either one or both of the bits CAPN and CAPP for that module must be set. The external CEX input for the module is sampled for a transition. When a valid transition occurs the PCA hardware loads the value of the PCA counter registers (CH and CL) into the module's capture registers (CCAPnL and CCAPnH). If the CCFn and the ECCFn bits for the module are both set, an interrupt will be generated.

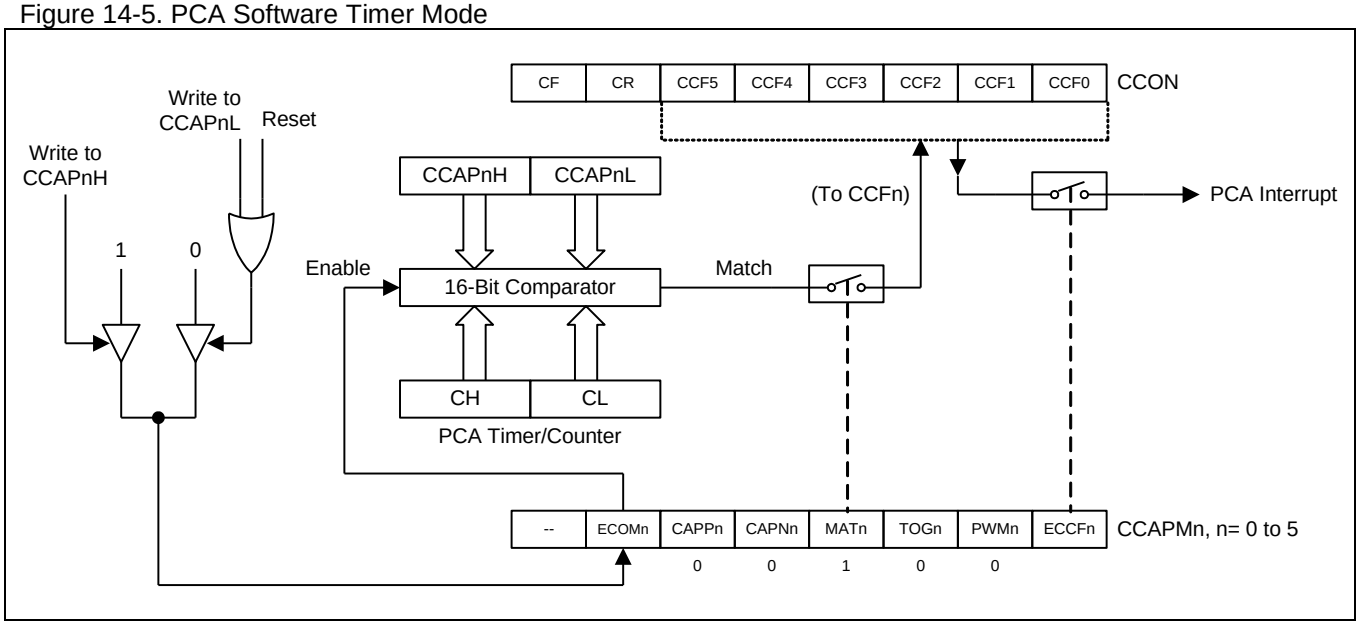
Figure 14-4. PCA Capture Mode



14.4.2. 16-bit Software Timer Mode

The PCA modules can be used as software timers by setting both the ECOM and MAT bits in the module's CCAPMn register. The PCA timer will be compared to the module's capture registers, and when a match occurs an interrupt will occur if the CCFn and the ECCFn bits for the module are both set.

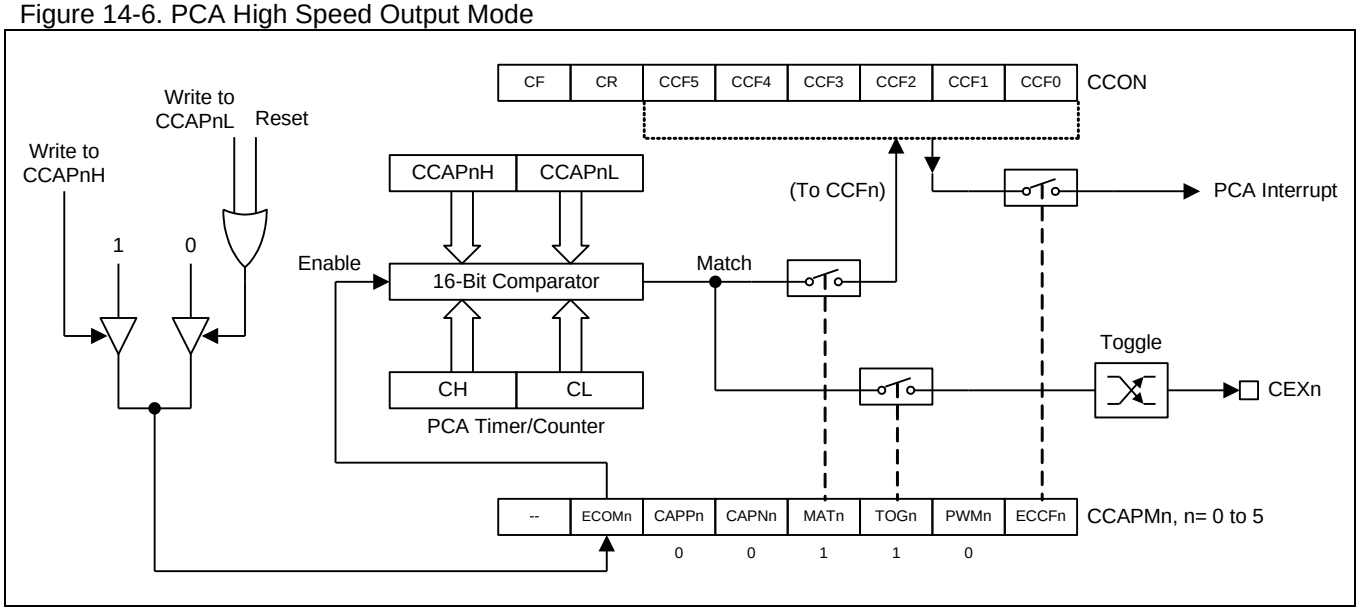
Figure 14-5. PCA Software Timer Mode



14.4.3. High Speed Output Mode

In this mode the CEX output associated with the PCA module will toggle each time a match occurs between the PCA counter and the module's capture registers. To activate this mode, the TOG, MAT and ECOM bits in the module's CCAPMn register must be set.

Figure 14-6. PCA High Speed Output Mode



14.4.4. PWM Mode

All of the PCA modules can be used as PWM outputs. The frequency of the output depends on the clock source for the PCA timer. All of the modules will have the same frequency of output because they all share the PCA timer.

The duty cycle of each module is determined by the module's capture register CCAPnL and the extended 9th bit, ECAPnL. When the 9-bit value of { 0, [CL] } is *less than* the 9-bit value of { ECAPnL, [CCAPnL] } the output will be low, and if *equal to or greater than* the output will be high.

When CL overflows from 0xFF to 0x00, { ECAPnL, [CCAPnL] } is reloaded with the value of { ECAPnH, [CCAPnH] }. This allows updating the PWM without glitches. The PWMn and ECOMn bits in the module's CCAPMn register must be set to enable the PWM mode.

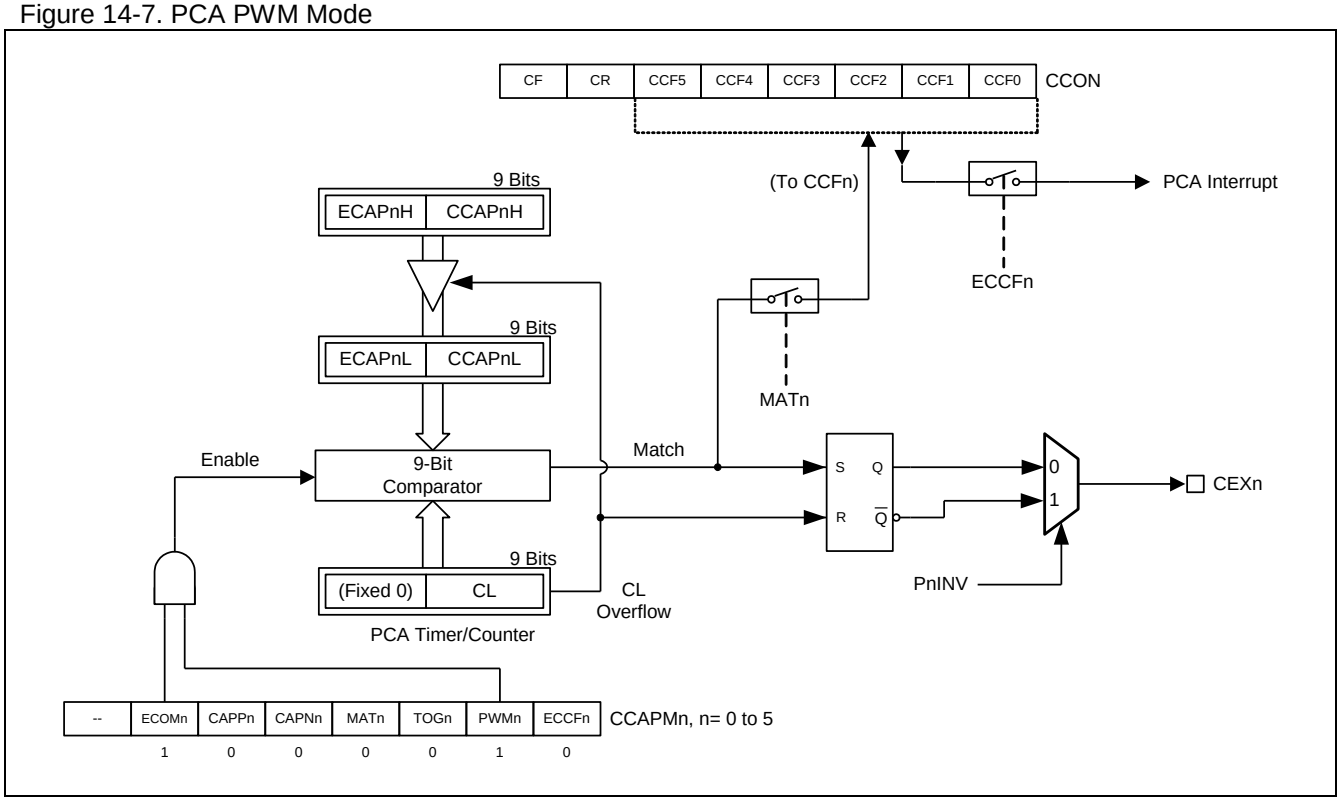
Using the 9-bit comparison, the duty cycle of the output can be improved to really start from 0%, and up to 100%. The formula for the duty cycle is:

Duty Cycle = 1 – { ECAPnH, [CCAPnH] } / 256.

Where, *[CCAPnH]* is the 8-bit value of the CCAPnH register, and *ECAPnH* (bit-1 in the PCAPWMn register) is 1-bit value. So, *{ ECAPnH, [CCAPnH] }* forms a 9-bit value for the 9-bit comparator.

For examples,
a. If ECAPnH=0 & CCAPnH=0x00 (i.e., 0x000), the duty cycle is 100%.
b. If ECAPnH=0 & CCAPnH=0x40 (i.e., 0x040) the duty cycle is 75%.
c. If ECAPnH=0 & CCAPnH=0xC0 (i.e., 0x0C0), the duty cycle is 25%.
d. If ECAPnH=1 & CCAPnH=0x00 (i.e., 0x100), the duty cycle is 0%.

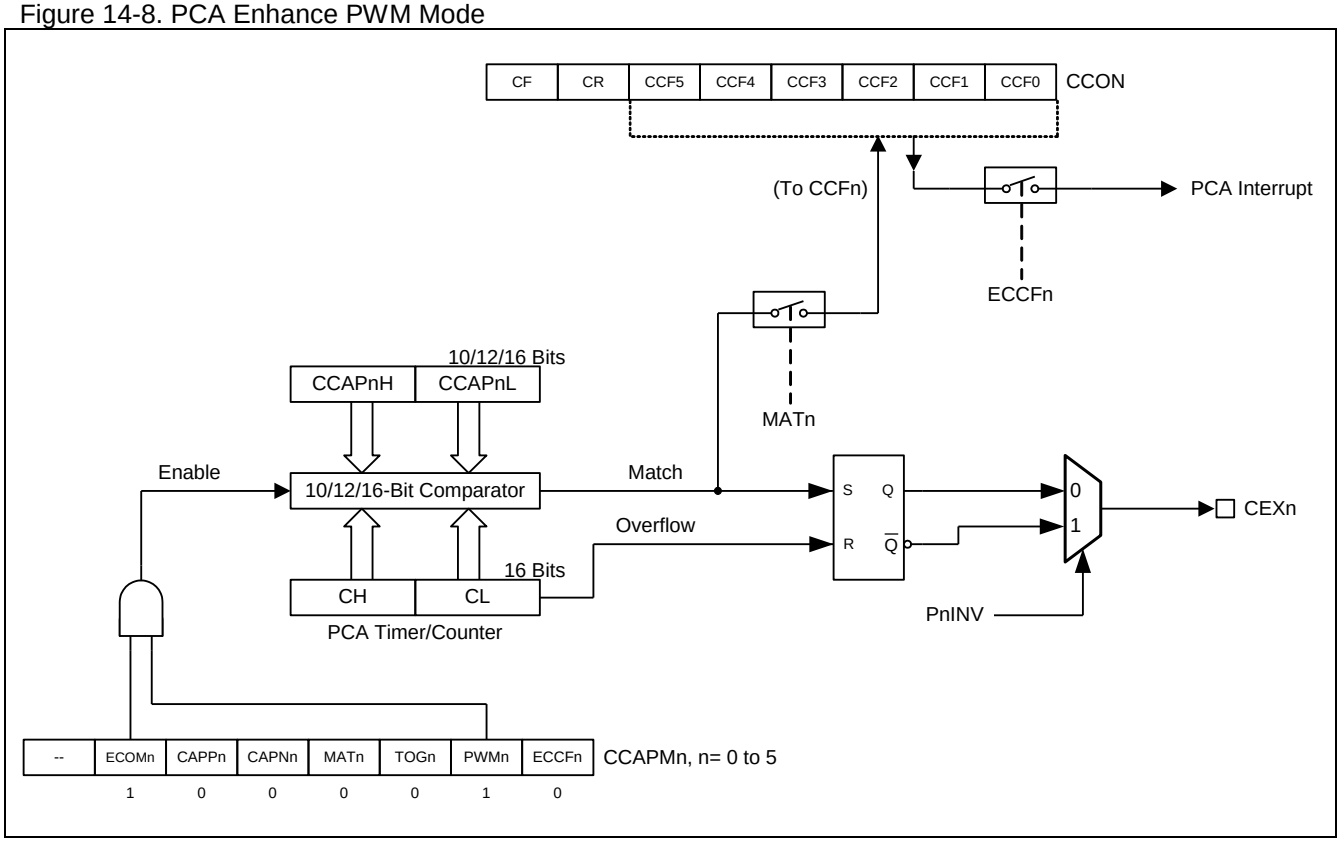
Figure 14-7. PCA PWM Mode



14.4.5. Enhance PWM Mode

The MG82Fx564 provides the variable PWM mode to enhance the control capability on PWM application. There are additional 10/12/16 bits PWM can be assigned in each channel and each PWM channel with different resolution can operate concurrently.

Figure 14-8. PCA Enhance PWM Mode



PCAPWMn: PWM Mode Auxiliary Register, n=0~5

SFR Page = All

SFR Address = 0xF2~0xF7 POR+RESET = 0000-0000

7	6	5	4	3	2	1	0
PnRS1	**PnRS0**	**PnPS2**	**PnPS1**	**PnPS0**	**PnINV**	ECAPnH	ECAPnL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~6: PnRS1~0, PWMn Resolution Setting 1~0.
00: 8 bit PWMn, the overflow is active when {CH,CL} is 0xXXFF → 0xXX00.
01: 10 bit PWMn, the overflow is active when {CH,CL} is 0xXx3FF → 0xXx[00]00.
10: 12 bit PWMn, the overflow is active when {CH,CL} is 0xXxFFF → 0xXx000.
11: 16 bit PWMn, the overflow is active when {CH,CL} is 0xFFFF → 0x0000.

Bit 5~3: PnPS2~0, PWMn Start Phase Setting 2~0.
000: The enabled PWM channel starts at 0 degree and ends at digital comparator matched.
001: The enabled PWM channel starts at 90 degree and ends at digital comparator matched.
010: The enabled PWM channel starts at 180 degree and ends at digital comparator matched.
011: The enabled PWM channel starts at 270 degree and ends at digital comparator matched.
100: The enabled PWM channel starts at 120 degree and ends at digital comparator matched.
101: The enabled PWM channel starts at 240 degree and ends at digital comparator matched.
110: The enabled PWM channel starts at 60 degree and ends at digital comparator matched.
111: The enabled PWM channel starts at 300 degree and ends at digital comparator matched.

Bit 2: PnINV, Invert PWM output on CEXn.
0: Non-inverted PWM output.
1: Inverted PWM output.

Bit 1: ECAPnH: Extended MSB bit, associated with CCAPnH to become a 9th-bit register used in 8-bit PWM mode. As well as for 10/12/16 bit PWM, it will become a 11th/13th/17th bit register.

Bit 0: ECAPnL: Extended MSB bit, associated with CCAPnL to become a 9th-bit register used in 8-bit PWM mode. As well as for 10/12/16 bit PWM, it will become a 11th/13th/17th bit register.

CMOD: PCA Counter Mode Register

SFR Page = All

SFR Address = 0xD9 POR+RESET = 0xxx-x000

7	6	5	4	3	2	1	0
CIDL	**FEOV**	--	--	--	CPS1	CPS0	ECF
R/W	**R/W**	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**

FEOV: Maximum Counter {CL} value on FE.

FEOV=0 Maximum CL counter value on FF.

FEOV=1 Maximum CL counter value on FE.

14.5. PCA Sample Code

(1). Required Function: Set PWM2/PWM3 output with 25% & 75% duty cycle

Assembly Code Example:

```
PWM2        EQU     02h
ECOM2       EQU     40h
PWM3        EQU     02h
ECOM3       EQU     40h


PWM2_PWM3:
        MOV     CCON,#00H                   ; stop CR
        MOV     CMOD,#02H                   ; PCA clock source = system clock / 2

        MOV     CCAPM2, #(ECOM2 + PWM2)     ; enable PCA module 2 (PWM mode)
        MOV     CCAP2H,#0C0H                ; 25%

        MOV     CCAPM3, #(ECOM3 + PWM3)     ; enable PCA module 3 (PWM mode)
        MOV     CCAP3H,#40H                 ; 75%
        ;
        SETB    CR                          ; start PCA
```

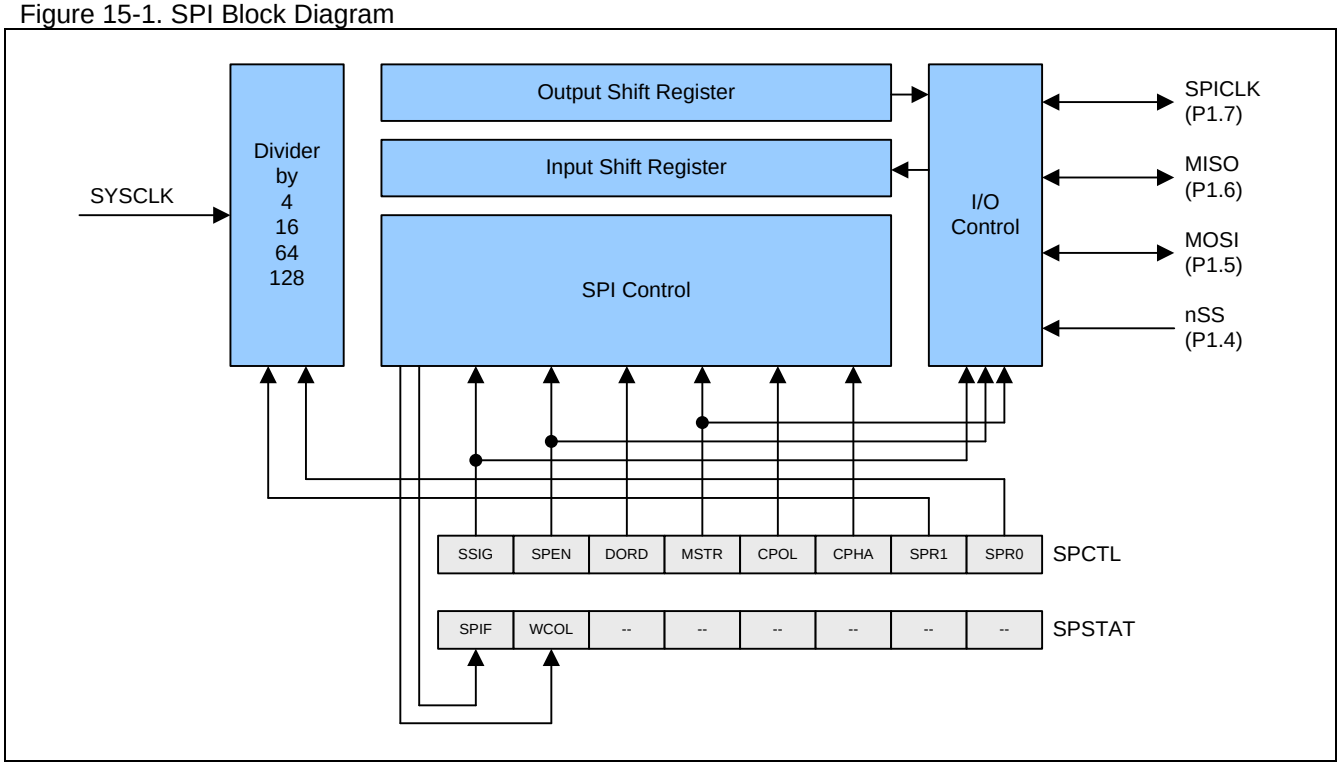
C Code Example:

```
#define  PWM2               EQU     0x02
#define  ECOM2              EQU     0x40
#define  PWM3               EQU     0x02
#define  ECOM3              EQU     0x40

void main(void)
{
        // set PCA
        CCON = 0x00;                        // disable PCA & clear CCF0, CCF1, CCF2, CCF3, CF flag
        CMOD = 0x02;                        // PCA clock source = system clock / 2

        CCAPM2 |= (ECOM2 | PWM2);           // module 2 (Non-inverted)
        CCAP2H = 0xC0;                      // 25%

        CCAPM3 |= (ECOM3 | PWM3);           // module 3
        CCAP3H = 0x40;                      // 75 %
        //---------------------------------------------
        CR = 1;                             // start PCA's PWM output

        while (1);
}
```

15. Serial Peripheral Interface (SPI)

The MG82Fx564 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 3 Mbps can be supported in either Master or Slave mode under a 12MHz system clock. It has a Transfer Completion Flag (SPIF) and Write Collision Flag (WCOL) in the SPI status register (SPSTAT).

Figure 15-1. SPI Block Diagram



The SPI interface has four pins: MISO (P1.6), MOSI (P1.5), SPICLK (P1.7) and /SS (P1.4):

• 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. The /SS is ignored if any of the following conditions are true:

- If the SPI system is disabled, i.e. SPEN (SPCTL.6) = 0 (reset value).
- If the SPI is configured as a master, i.e., MSTR (SPCTL.4) = 1, and P1.4 (/SS) is configured as an output.
- If the /SS pin is ignored, i.e. SSIG (SPCTL.7) bit = 1, this pin is configured for port functions.

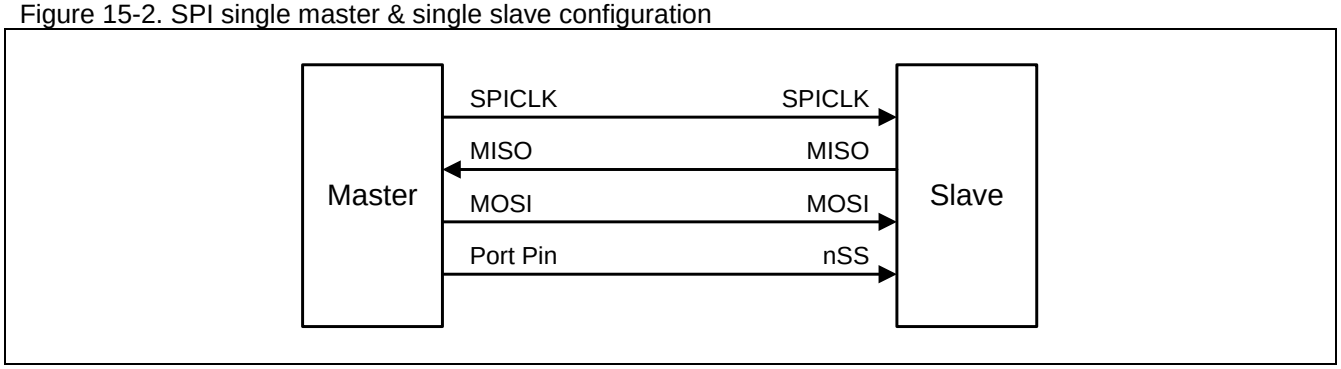
Note that even if the SPI is configured as a master (MSTR=1), it can still be converted to a slave by driving the /SS pin low (if SSIG=0). Should this happen, the SPIF bit (SPSTAT.7) will be set. (See Section 15.2.3: Mode change on /SS-pin)

15.1. Typical SPI Configurations

15.1.1. Single Master & Single Slave

For the master: any port pin, including P1.4 (/SS), can be used to drive the /SS pin of the slave.
For the slave: SSIG is '0', and /SS pin is used to determine whether it is selected.

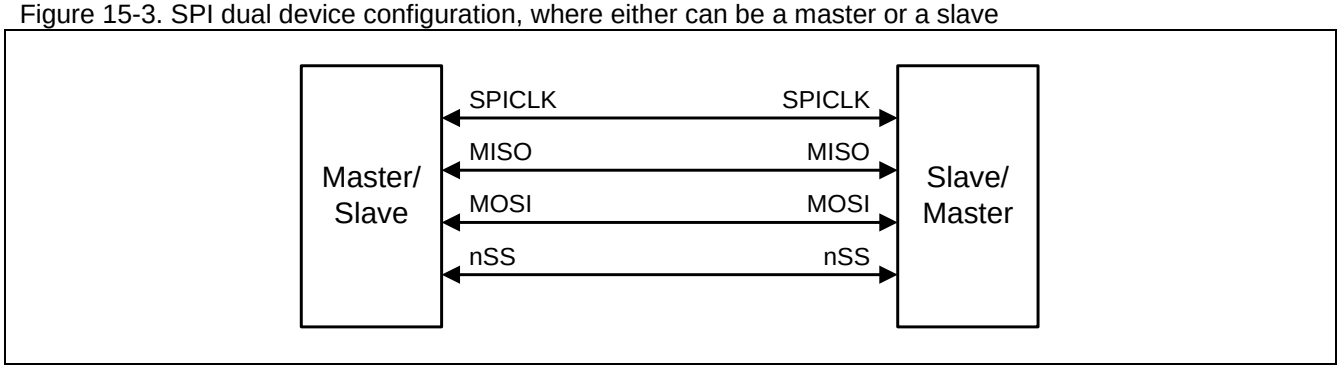
Figure 15-2. SPI single master & single slave configuration



15.1.2. 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 P1.4 (/SS) configured in quasi-bidirectional mode. When any device initiates a transfer, it can configure P1.4 as an output and drive it low to force a “mode change to slave” in the other device. (See Section 15.2.3: Mode change on /SS-pin)

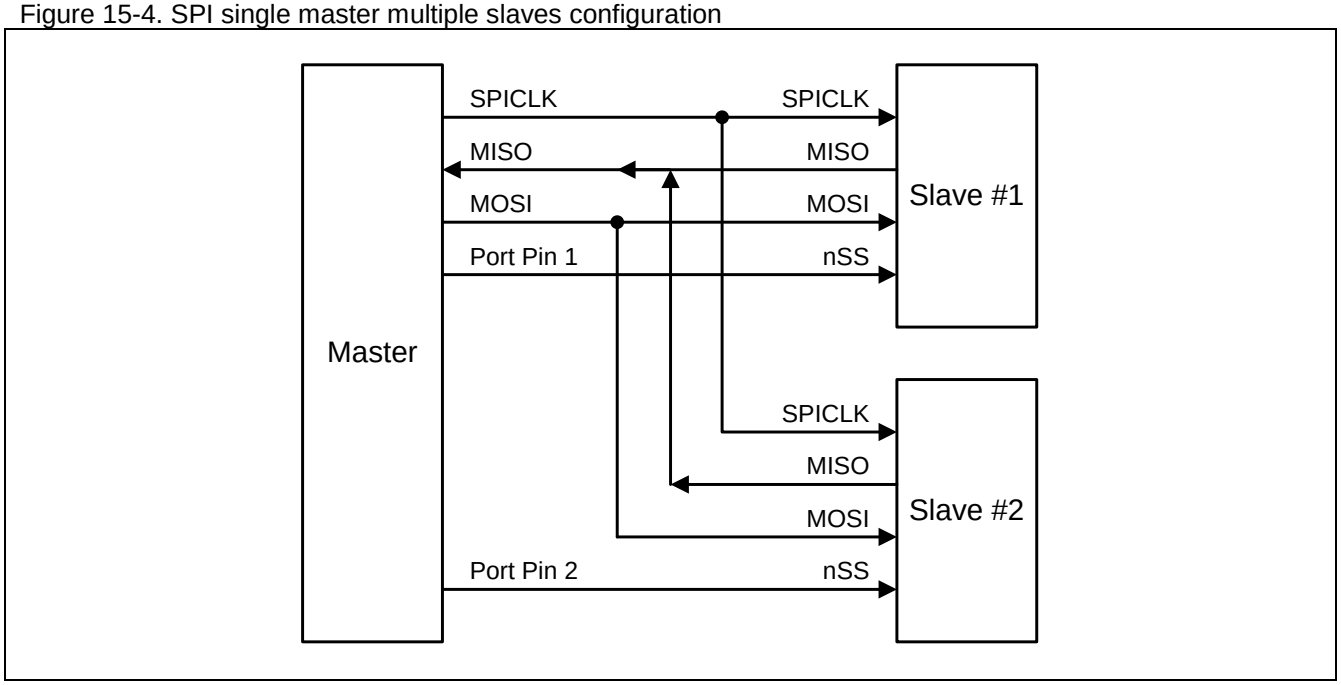
Figure 15-3. SPI dual device configuration, where either can be a master or a slave



15.1.3. Single Master & Multiple Slaves

For the master: any port pin, including P1.4 (/SS), can be used to drive the /SS pins of the slaves.
For all the slaves: SSIG is ‘0’, and /SS pin are used to determine whether it is selected.

Figure 15-4. SPI single master multiple slaves configuration



15.2. Configuring the SPI

Table 15-1 shows configuration for the master/slave modes as well as usages and directions for the modes.

Table 15-1. SPI Master and Slave Selection

SPEN (SPCTL.6)	SSIG (SPCTL.7)	nSS -pin	MSTR (SPCTL.4)	Mode	MISO -pin	MOSI -pin	SPICLK -pin	Remarks
0	X	X	X	SPI disabled	input	input	input	P1.4~P1.7 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".

15.2.1. Additional Considerations for a Slave

When CPHA is 0, SSIG must be 0 and nSS pin must be negated and reasserted between each successive serial byte transfer. Note the SPDAT register cannot be written while nSS 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 nSS 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.

15.2.2. 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 nSS 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.

15.2.3. Mode Change on nSS-pin

If SPEN=1, SSIG=0, MSTR=1 and nSS 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.

15.2.4. Write Collision

The SPI is single buffered in the transmit direction and double buffered in the receive direction. New data for transmission can not be written to the shift register until the previous transaction is complete. The WCOL (SPSTAT.6) bit is set to indicate data collision when the data register is written during transmission. In this case, the data currently being transmitted will continue to be transmitted, but the new data, i.e., the one causing the collision, will be lost.

While write collision is detected for both a master or a slave, it is uncommon for a master because the master has full control of the transfer in progress. The slave, however, has no control over when the master will initiate a transfer and therefore collision can occur.

For receiving data, received data is transferred into a parallel read data buffer so that the shift register is free to accept a second character. However, the received character must be read from the Data Register (SPDAT) before the next character has been completely shifted in. Otherwise, the previous data is lost.

WCOL can be cleared in software by writing ‘1’ to the bit.

15.2.5. SPI Clock Rate Select

The SPI clock rate selection (in master mode) uses the SPR1 and SPR0 bits in the SPCTL register, as shown in Table 15-2.

Table 15-2. SPI Serial Clock Rates

SPR1	SPR0	SPI Clock Rate @ SYSCLK=12MHz	SYSCLK divided by
0	0	3 MHz	4
0	1	750 KHz	16
1	0	187.5 KHz	64
1	1	93.75 KHz	128

Where, SYSCLK is the system clock.

15.3. 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 Clock Phase Bit, CPHA.

Figure 15-5. SPI Slave Transfer Format with CPHA=0

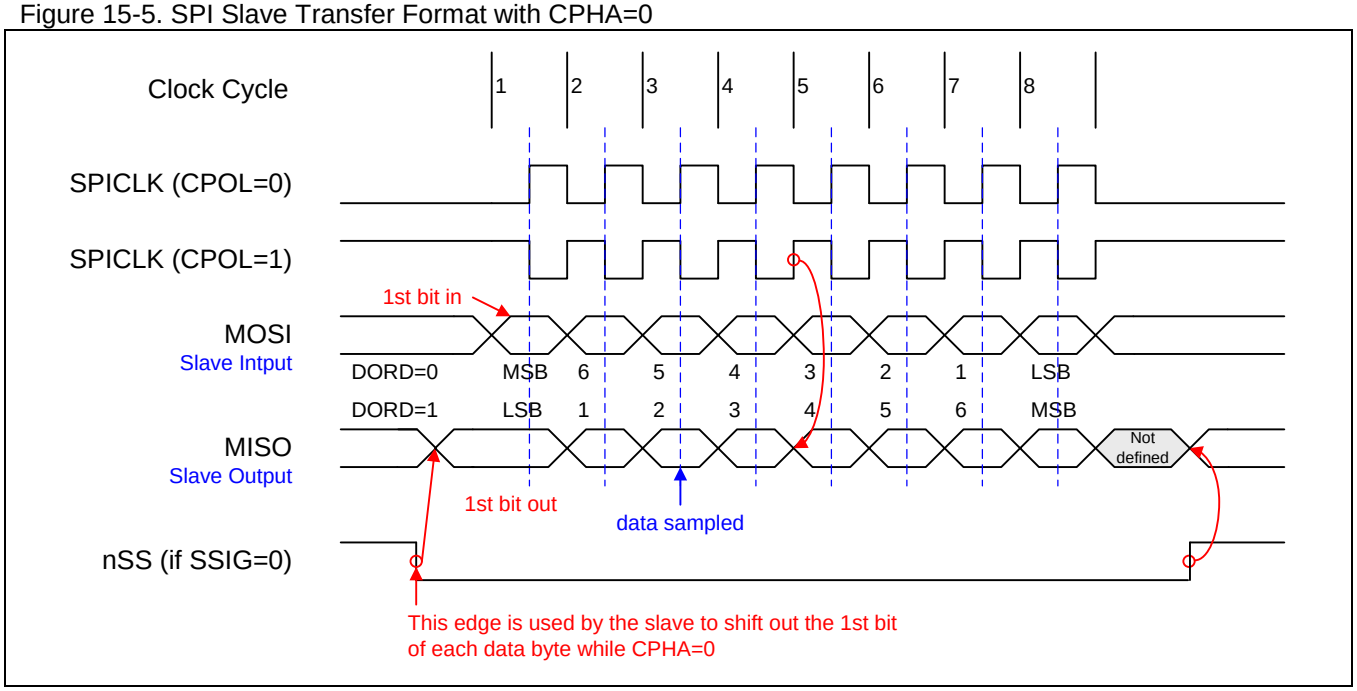


Figure 15-6. Slave Transfer Format with CPHA=1

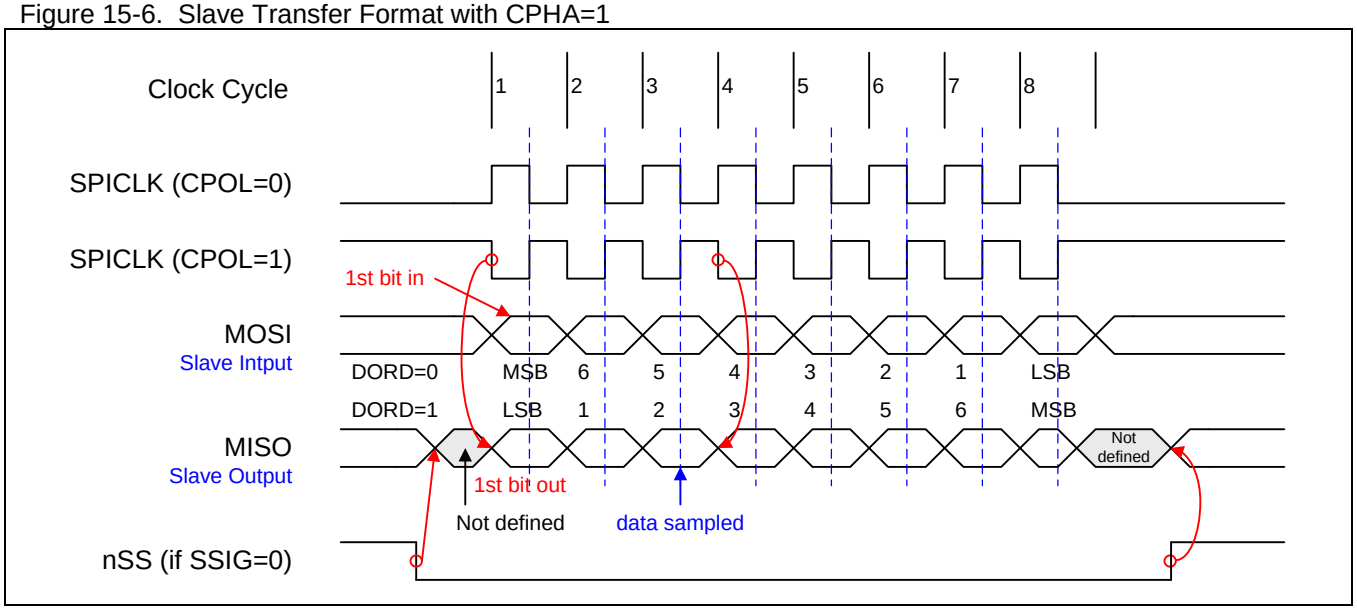


Figure 15-7. SPI Master Transfer Format with CPHA=0

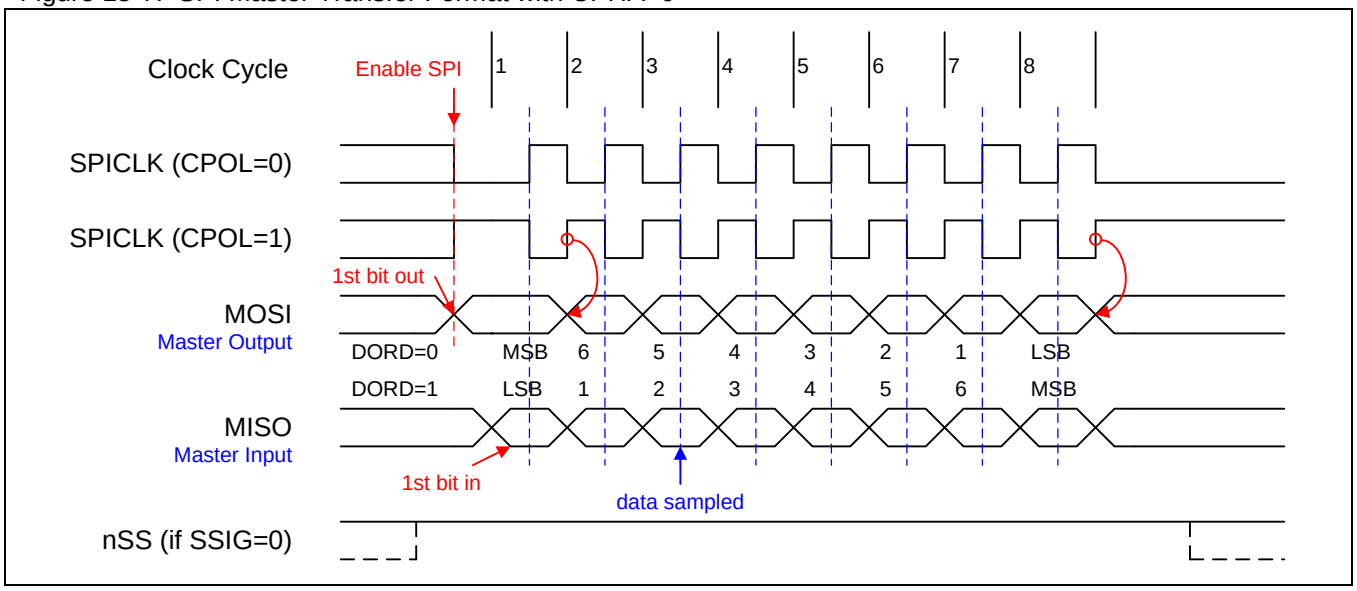
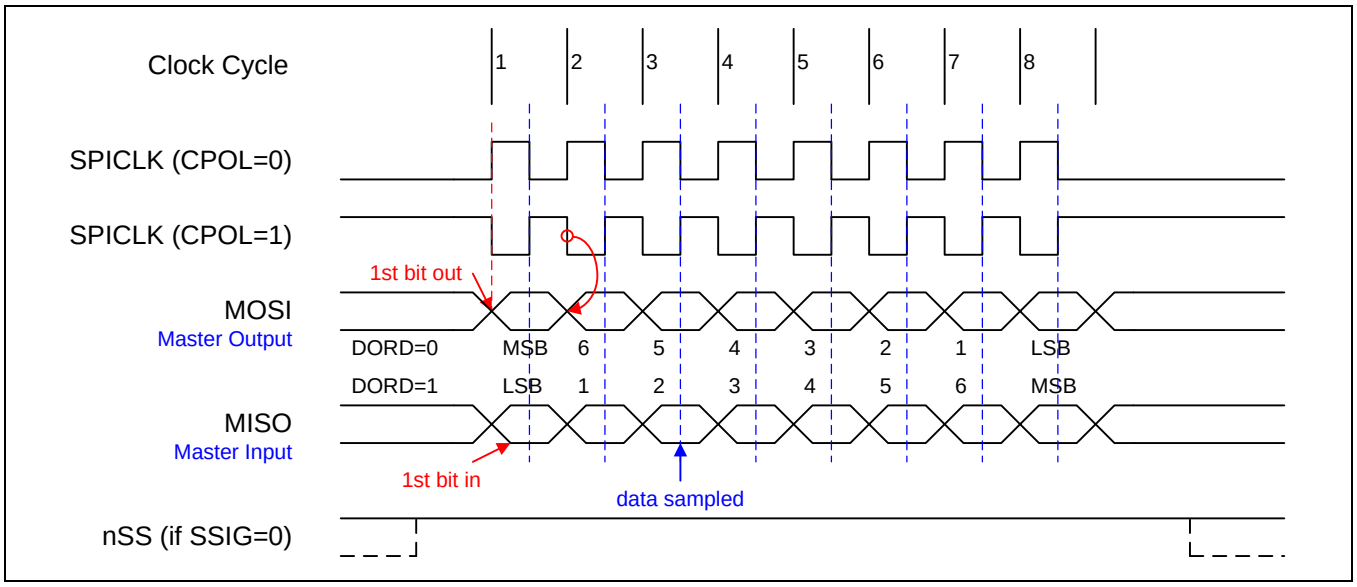


Figure 15-8. SPI Master Transfer Format with CPHA=1



15.4. SPI Register

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

SPCON: SPI Control Register

SFR Page = All

SFR Address = 0x85 RESET= 0000-0100

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

Bit 7: SSIG, nSS is ignored.
0: The nSS pin decides whether the device is a master or slave.
1: MSTR decides whether the device is a master or slave.

Bit 6: SPEN, SPI enable.
0: The SPI interface is disabled and all SPI pins will be general-purpose I/O ports.
1: The SPI is enabled.

Bit 5: DORD, SPI data order.
0: The MSB of the data byte is transmitted first.
1: The LSB of the data byte is transmitted first.

Bit 4: MSTR, Master/Slave mode select
0: Selects slave SPI mode.
1: Selects master SPI mode.

Bit 3: CPOL, SPI clock polarity select
0: SPICLK is low when Idle. The leading edge of SPICLK is the rising edge and the trailing edge is the falling edge.
1: SPICLK is high when Idle. The leading edge of SPICLK is the falling edge and the trailing edge is the rising edge.

Bit 2: CPHA, SPI clock phase select
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.
1: Data is driven on the leading edge of SPICLK, and is sampled on the trailing edge.
(Note: If SSIG=1, CPHA must not be 1, otherwise the operation is not defined.)

Bit 1~0: SPR1-SPR0, SPI clock rate select (in master mode)
00: SYSCLK/4
01: SYSCLK/16
10: SYSCLK/64
11: SYSCLK/128 (Where, SYSCLK is the system clock.)

SPSTAT: SPI Status Register

SFR Page = All

SFR Address = 0x84 RESET= 00XX-XXXX

7	6	5	4	3	2	1	0
SPIF	WCOL	--	--	--	--	--	--
R/W	**R/W**	**R**	**R**	**R**	**R**	**R**	**R**

Bit 7: SPIF, SPI transfer completion flag
0: **The SPIF is cleared in software by writing '1' to this bit.**
1: When a serial transfer finishes, the SPIF bit is set and an interrupt is generated if SPI interrupt is enabled. If nSS pin is driven low when SPI is in master mode with SSIG=0, SPIF will also be set to signal the "mode change".

Bit 6: WCOL, SPI write collision flag.
0: The WCOL flag is cleared in software by writing '1' to this bit.

1: The WCOL bit is set if the SPI data register, SPDAT, is written during a data transfer (see Section 15.2.4: Write Collision).

Bit 5~0: Reserved.

SPDAT: SPI Data Register

SFR Page = All

SFR Address = 0x86 RESET= 0000-0000

7	6	5	4	3	2	1	0
(MSB)							(LSB)
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

SPDAT has two physical buffers for writing to and reading from during transmit and receive, respectively.

15.5. SPI Sample Code

(1). Required Function: SPI Master Read and Write, sample data at rising edge and clock leading edge is rising.

Assembly Code Example:

```
CPHA            EQU             04h
CPOL            EQU             08h
MSTR            EQU             10h
SPEN            EQU             40h
SSIG            EQU             80h
SPIF            EQU             80h

Initial_SPI:                                        ;initial SPI
   ORL    SPICTL, #(SSIG + SPEN + MSTR)             ;enable SPI and Master mode
   RET

SPI_Write:
   MOV    SPIDAT, R7                                ;write arg R7
 wait_write:
   MOV    A, SPISTAT
   JNB    ACC.7, wait_write                         ;wait transfer finishes
   ANL    SPISTAT, #(0FFh - SPIF)                   ;clear SPI interrupt flag
   RET

SPI_Read:
   MOV    SPIDAT, #0FFh                             ;trigger SPI read
 wait_read:
   MOV    A, SPISTAT
   JNB    ACC.7, wait_read                          ;wait read finishes
   ANL    SPISTAT, #(0FFh - SPIF)                   ;clear SPI interrupt flag
   MOV    A, SPIDAT                                 ;move read data to accumulator
   RET
```

C Code Example:

```
#define  CPHA                0x04
#define  CPOL                0x08
#define  MSTR                0x10
#define  SPEN                0x40
#define  SSIG                0x80
#define  SPIF                0x80

void Initial_SPI(void)
{
   SPICTL |= (SSIG | SPEN | MSTR);                  // enable SPI and Master mode
}

void SPI_Write(unsigned char arg)
{
   SPIDAT = arg;                                    //write arg
   while(!(SPISTAT & SPIF));                        //wait transfer finishes
   SPISTAT &= ~SPIF;                                //clear SPI interrupt flag
}

unsigned char SPI_Read(void)
{
   SPIDAT = 0xFF;                                   //trigger SPI read
   while(!SPISTAT & SPIF);                          //wait transfer finishes
   SPISTAT &= ~SPIF;                                //clear SPI interrupt flag
   return SPIDAT;
}
```

(2). Required Function: SPI Master Read and Write, sample data at rising edge and clock leading edge is falling.

Assembly Code Example:

```
CPHA        EQU         04h
CPOL        EQU         08h
MSTR        EQU         10h
SPEN        EQU         40h
SSIG        EQU         80h
SPIF        EQU         80h

Initial_SPI:                                        ;initial SPI
   ORL   SPICTL, #(SSIG + SPEN + MSTR + CPOL)       ;enable SPI and Master mode
   RET

SPI_Write:
   MOV   SPIDAT, R7                                 ;write arg R7
 wait_write:
   MOV   A, SPISTAT
   JNB   ACC.7, wait_write                          ;wait transfer finishes
   ANL   SPISTAT, #(0FFh - SPIF)                    ;clear SPI interrupt flag
   RET

SPI_Read:
   MOV   SPIDAT, #0FFh                              ;trigger SPI read
 wait_read:
   MOV   A, SPISTAT
   JNB   ACC.7, wait_read                           ;wait read finishes
   ANL   SPISTAT, #(0FFh - SPIF)                    ;clear SPI interrupt flag
   MOV   A, SPIDAT                                  ;move read data to accumulator
   RET
```

C Code Example:

```
#define  CPHA            0x04
#define  CPOL            0x08
#define  MSTR            0x10
#define  SPEN            0x40
#define  SSIG            0x80
#define  SPIF            0x80

void Initial_SPI(void)
{
   SPICTL |= (SSIG | SPEN | MSTR | CPOL);           // enable SPI and Master mode
}

void SPI_Write(unsigned char arg)
{
   SPIDAT = arg;                                    //write arg
   while(!(SPISTAT & SPIF));                        //wait transfer finishes
   SPISTAT &= ~SPIF;                                //clear SPI interrupt flag
}

unsigned char SPI_Read(void)
{
   SPIDAT = 0xFF;                                   //trigger SPI read
   while(!SPISTAT & SPIF);                          //wait transfer finishes
   SPISTAT &= ~SPIF;                                //clear SPI interrupt flag
   return SPIDAT;
}
```

(3). Required Function: SPI Master Read and Write, sample data at falling edge and clock leading edge is rising.

Assembly Code Example:

```
CPHA            EQU             04h
CPOL            EQU             08h
MSTR            EQU             10h
SPEN            EQU             40h
SSIG            EQU             80h
SPIF            EQU             80h

Initial_SPI:                                        ;initial SPI
   ORL    SPICTL, #(SSIG + SPEN + MSTR + CPHA)      ;enable SPI and Master mode
   RET

SPI_Write:
   MOV    SPIDAT, R7                                ;write arg R7
 wait_write:
   MOV    A, SPISTAT
   JNB    ACC.7, wait_write                         ;wait transfer finishes
   ANL    SPISTAT, #(0FFh - SPIF)                   ;clear SPI interrupt flag
   RET

SPI_Read:
   MOV    SPIDAT, #0FFh                             ;trigger SPI read
 wait_read:
   MOV    A, SPISTAT
   JNB    ACC.7, wait_read                          ;wait read finishes
   ANL    SPISTAT, #(0FFh - SPIF)                   ;clear SPI interrupt flag
   MOV    A, SPIDAT                                 ;move read data to accumulator
   RET
```

C Code Example:

```
#define  CPHA            0x04
#define  CPOL            0x08
#define  MSTR            0x10
#define  SPEN            0x40
#define  SSIG            0x80
#define  SPIF            0x80

void Initial_SPI(void)
{
   SPICTL |= (SSIG | SPEN | MSTR | CPHA);           // enable SPI and Master mode
}

void SPI_Write(unsigned char arg)
{
   SPIDAT = arg;                                    //write arg
   while(!(SPISTAT & SPIF));                        //wait transfer finishes
   SPISTAT &= ~SPIF;                                //clear SPI interrupt flag
}

unsigned char SPI_Read(void)
{
   SPIDAT = 0xFF;                                   //trigger SPI read
   while(!SPISTAT & SPIF);                          //wait transfer finishes
   SPISTAT &= ~SPIF;                                //clear SPI interrupt flag
   return SPIDAT;
}
```

(4). Required Function: SPI Master Read and Write, sample data at falling edge and clock leading edge is falling.

Assembly Code Example:

```
CPHA        EQU         04h
CPOL        EQU         08h
MSTR        EQU         10h
SPEN        EQU         40h
SSIG        EQU         80h
SPIF        EQU         80h

Initial_SPI:                                          ;initial SPI
   ORL   SPICTL, #(SSIG + SPEN + MSTR + CPOL + CPHA)  ;enable SPI and Master mode
   RET

SPI_Write:
   MOV   SPIDAT, R7                                   ;write arg R7
 wait_write:
   MOV   A, SPISTAT
   JNB   ACC.7, wait_write                            ;wait transfer finishes
   ANL   SPISTAT, #(0FFh - SPIF)                      ;clear SPI interrupt flag
   RET

SPI_Read:
   MOV   SPIDAT, #0FFh                                ;trigger SPI read
 wait_read:
   MOV   A, SPISTAT
   JNB   ACC.7, wait_read                             ;wait read finishes
   ANL   SPISTAT, #(0FFh - SPIF)                      ;clear SPI interrupt flag
   MOV   A, SPIDAT                                    ;move read data to accumulator
   RET
```

C Code Example:

```
#define  CPHA                0x04
#define  CPOL                0x08
#define  MSTR                0x10
#define  SPEN                0x40
#define  SSIG                0x80
#define  SPIF                0x80

void Initial_SPI(void)
{
   SPICTL |= (SSIG | SPEN | MSTR | CPOL | CPHA);   // enable SPI and Master mode
}

void SPI_Write(unsigned char arg)
{
   SPIDAT = arg;                                    //write arg
   while(!(SPISTAT & SPIF));                        //wait transfer finishes
   SPISTAT &= ~SPIF;                                //clear SPI interrupt flag
}

unsigned char SPI_Read(void)
{
   SPIDAT = 0xFF;                                   //trigger SPI read
   while(!SPISTAT & SPIF);                          //wait transfer finishes
   SPISTAT &= ~SPIF;                                //clear SPI interrupt flag
   return SPIDAT;
}
```

16. Keypad Interrupt (KBI)

The Keypad Interrupt function is intended primarily to allow a single interrupt to be generated when Port 2 is equal to or not equal to a certain pattern. This function can be used for bus address recognition or keypad recognition.

There are three SFRs used for this function. The Keypad Interrupt Mask Register (KBMASK) is used to define which input pins connected to Port 2 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 2. 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 EIE1 register and EA=1. The PATN_SEL bit in the Keypad Interrupt Control Register (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 2 which is enabled by KBMASK register will cause the hardware to set the interrupt flag KBIF and generate an interrupt if it has been enabled. The interrupt may wake up the CPU from Idle mode or Power-Down mode. This feature is particularly useful in handheld, battery powered systems that need to carefully manage power consumption but also need to be convenient to use.

16.1. Keypad Register

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

KBPATN: Keypad Pattern Register

SFR Page = All

SFR Address = 0xD5 RESET= 1111-1111

7	6	5	4	3	2	1	0
KBPATN.7	KBPATN.6	KBPATN.5	KBPATN.4	KBPATN.3	KBPATN.2	KBPATN.1	KBPATN.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

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

KBCON: Keypad Control Register

SFR Page = All

SFR Address = 0xD6 RESET= XXXX-XX00

7	6	5	4	3	2	1	0
--	--	--	--	--	--	PATN_SEL	KBIF
R	**R**	**R**	**R**	**R**	**R**	**R/W**	**R/W**

Bit 7~2: Reserved.

Bit 1: PATN_SEL, Pattern Matching Polarity selection.
0: The keypad input has to be not equal to user-defined keypad pattern in KBPATN to generate the interrupt.
1: The keypad input has to be equal to the user-defined keypad pattern in KBPATN to generate the interrupt.

Bit 0: KBIF, Keypad Interrupt Flag.
0: Must be cleared by software by writing “0”.
1: Set when Port 2 matches user defined conditions specified in KBPATN, KBMASK, and PATN_SEL.

KBMASK: Keypad Interrupt Mask Register

SFR Page = All

SFR Address = 0xD7 RESET= 0000-0000

7	6	5	4	3	2	1	0
KBMASK.7	KBMASK.6	KBMASK.5	KBMASK.4	KBMASK.3	KBMASK.2	KBMASK.1	KBMASK.0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

KBMASK.7: When set, enables P2.7 as a cause of a Keypad Interrupt (KBI7).
KBMASK.6: When set, enables P2.6 as a cause of a Keypad Interrupt (KBI6).
KBMASK.5: When set, enables P2.5 as a cause of a Keypad Interrupt (KBI5).
KBMASK.4: When set, enables P2.4 as a cause of a Keypad Interrupt (KBI4).
KBMASK.3: When set, enables P2.3 as a cause of a Keypad Interrupt (KBI3).
KBMASK.2: When set, enables P2.2 as a cause of a Keypad Interrupt (KBI2).
KBMASK.1: When set, enables P2.1 as a cause of a Keypad Interrupt (KBI1).
KBMASK.0: When set, enables P2.0 as a cause of a Keypad Interrupt (KBI0).

16.2. Keypad Interrupt Sample Code

(1). Required Function: Implement a KBI function on P2.3~P2.0

Assembly Code Example:

```
   Jmp     main;
   ORG     0006Bh
KBI_ISR:                          ;
   PUSH    ACC                    ;

   To do   KBI key check……

   ANL     KBCON,#(0FFh - KBIF)          ; Clear KBI flag (write “0”)
   POP     ACC                    ;
   RETI                           ;

main:
   ANL     P2M1,#0F0h             ; Set P2.3~P2.0 to Input mode
   ORL     P2M0,#00Fh             ; Set P2.3~P2.0 to Input mode

   MOV     KBMASK,#00Fh           ; Enable P2.3~P2.0 KBI function

   ANL     KBCON,# (0FFh - KBIF)  ; Clear KBI flag (write “0”)

   ORL     AUXIE,#EKBI            ; Enable KBI interrupt
   SETB    EA                     ; Enable global interrupt

    Jmp     $;
```

C Code Example:

```
void KBI_ISR(void) interrupt 13
{
   // Do KBI key check
   KBCON &= ~KBIF;                // Clear KBI Flag
   }
}

void main(void)
{
   P2M1 &= 0xF0;                  // Set P2.3~P2.0 to Input mode
   P2M0 |= 0x0F;                  // Set P2.3~P2.0 to Input mode


   KBMASK = 0x0F;                 // Enable P2.3~P2.0 KBI function

   KBCON &= ~KBIF;                // Clear KBI flag (write “0”)

   AUXIE |= EKBI;                 // Enable KBI interrupt
   EA = 1;                        // Enable global interrupt

  While(1);
}
```

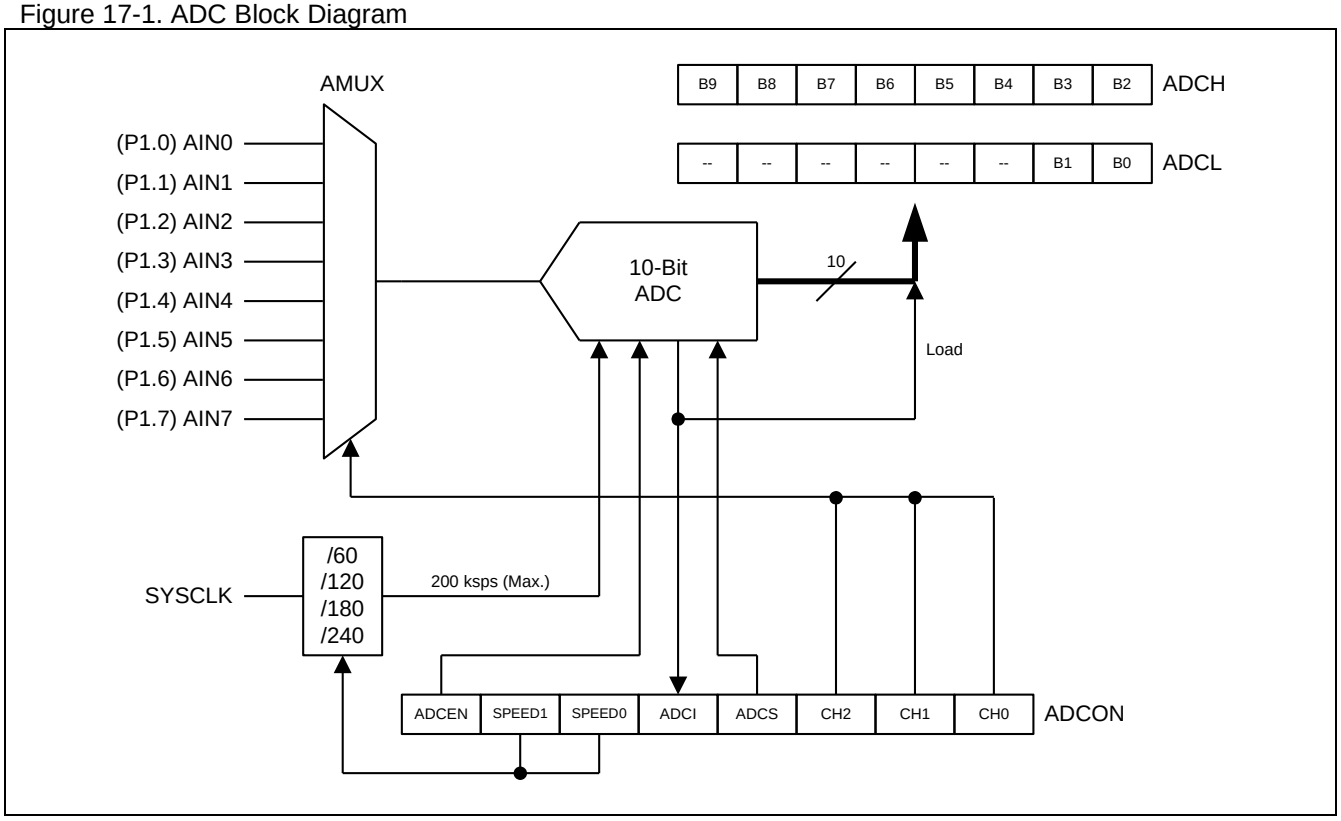
17. 10-Bit ADC

The ADC subsystem for the MG82Fx564 consists of an analog multiplexer (AMUX), and a 200 ksps, 10-bit successive-approximation-register ADC. The AMUX can be configured via the Special Function Registers shown in Figure 17-1. ADC operates in Single-ended modes, and may be configured to measure any of the pins on Port 1. The ADC subsystem is enabled only when the ADEN bit in the ADC Control register (ADCON) is set to logic 1. The ADC subsystem is in low power shutdown when this bit is logic 0.

Note: Suggest user doing ADC conversion under SYSCLK don't over 20MHz in High temperature (Over 60℃) environment.

17.1. ADC Structure

Figure 17-1. ADC Block Diagram



17.2. ADC Operation

ADC has a maximum conversion speed of 200 ksps. The ADC conversion clock is a divided version of the system clock, determined by the SPEED1, SPEED0 bits in the ADCON register.

Software writes a "1" to ADCS to start the ADC operation. After the conversion is complete (ADCI is high), the conversion result can be found in the ADC Result Registers (ADCH, ADCL). For the single ended ADC, the conversion result is:

$$\text{ADC Result} = \frac{V_{IN} \text{ ⓪1024}}{\text{VDD Voltage}}$$

17.2.1. ADC Input Channels

The analog multiplexer (AMUX) selects the inputs to the ADC, allowing any of the pins on Port 1 to be measured in single-ended mode. The ADC input channels are configured and selected by CHS.2~0 in the ADCON register as described in Figure 17-1. The selected pin is measured with respect to GND.

17.2.2. Starting a Conversion

Prior to using the ADC function, the user should:

1) Turn on the ADC hardware by setting the ADCEN bit,
2) Configure the conversion speed by bits SPEED1 and SPEED0,
3) Select the analog input channel by bits CHS2, CHS1 and CHS0,
4) Configure the selected input (shared with P1) to the Input-Only mode by P1M0 and P1M1 registers, and
5) Configure ADC result arrangement using ADRJ bit.

Now, user can set the ADCS bit to start the A-to-D conversion. The conversion time is controlled by bits SPEED1 and SPEED0. Once the conversion is completed, the hardware will automatically clear the ADCS bit, set the interrupt flag ADCI and load the 10 bits of conversion result into ADCH and ADCL (according to ADRJ bit) simultaneously.

As described above, the interrupt flag ADCI, when set by hardware, shows a completed conversion. Thus two ways may be used to check if the conversion is completed: (1) Always polling the interrupt flag ADCI by software; (2) Enable the ADC interrupt by setting bits EADC (in EXIE1 register) and EA (in IE register), and then the CPU will jump into its Interrupt Service Routine when the conversion is completed. Regardless of (1) or (2), the ADCI flag should be cleared by software before next conversion.

17.2.3. Sample Code for ADC

```
start:
        ;...

        MOV    ADCON,#0E2h     ;ADCEN=1, turn on ADC hardware
                               ;(SPEED1,SPEED0)=(1,1), Conv. Time= 60 clock cycles
                               ;select AIN0 (P1.2) as analog input

        ORL    P1M0,#00000100B  ;P1M0,bit2=1 ;configure P1.2 as Input-Only Mode
        ANL    P1M1,#11111011B   ;P1M1,bit2=0 ;

        ANL    AUXR0,#11111011B  ;ADRJ=0: ADCH contains B9~B2; ADCL contains B1,B0

        ;now, suppose the analog input is ready on AIN2 (P1.2)

        ORL    ADCON,#00001000B ;ADCS=1 ➔Start A-to-D conversion

wait_loop:
        MOV    ACC,ADCON
        JNB    ACC.4,wait_loop  ;wait until ADCI=1 ➔conversion completed

        ;now, the 10-bit ADC result is in the ADCH and ADCL.
        ;...
        ;...
```

17.2.4. ADC Conversion Time

The user can select the appropriate conversion speed according to the frequency of the analog input signal. For example, if SYSCLK =12MHz and a conversion speed of 60 clock cycles is selected, then the frequency of the analog input should be no more than 200KHz to maintain the conversion accuracy. (Conversion time = 1/12MHz x 60 = 5us, so the conversion speed = 1/5us = 200KHz.)

17.2.5. I/O Pins Used with ADC Function

The analog input pins used for the A/D converters also have its I/O port ‘s digital input and output function. In order to give the proper analog performance, a pin that is being used with the ADC should have its digital output as disabled. It is done by putting the port pin into the input-only mode as described in the Port Configurations section.

17.2.6. Idle and Power-Down Mode

In Idle mode and Power-Down mode, the ADC does not function. If the A/D is turned on, it will consume a little power. So, power consumption can be reduced by turning off the ADC hardware (ADCEN=0) before entering Idle mode and Power-Down mode.

17.3. ADC Register

ADCON: ADC Control Register

SFR Page = All

SFR Address = 0xC5 RESET = 0000-0000

7	6	5	4	3	2	1	0
ADCEN	SPEED1	SPEED0	ADCI	ADCS	CHS2	CHS1	CHS0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: ADCEN, ADC Enable.
0: Clear to turn off the ADC block..
1: Set to turn on the ADC block.

Bit 6~5: SPEED1 and SPEED0, ADC conversion speed control.

SPEED[1:0]	Conversion Clock Selection
0 0	SYSCLK/60
0 1	SYSCLK/120
1 0	SYSCLK/180
1 1	SYSCLK/240

Bit 4: ADCI, ADC Interrupt Flag.
0: The flag must be cleared by software.
1: This flag is set when an A/D conversion is completed. An interrupt is invoked if it is enabled.

Bit 3: ADCS. ADC Start of conversion.
0: ADCS cannot be cleared by software.
1: Setting this bit by software starts an A/D conversion. On completion of the conversion, the ADC hardware will clear ADCS and set the ADCI. A new conversion may not be started while either ADCS or ADCI is high.

Bit 2~0: CHS2 ~ CHS1, Input Channel Selection for ADC analog multiplexer.

CHS[2:0]	Selected ADC Channel
0 0 0	AIN0 (P1.0)
0 0 1	AIN1 (P1.1)
0 1 0	AIN2 (P1.2)
0 1 1	AIN3 (P1.3)
1 0 0	AIN4 (P1.4)
1 0 1	AIN5 (P1.5)
1 1 0	AIN6 (P1.6)
1 1 1	AIN7 (P1.7)

AUXR0: Auxiliary Register 0

SFR Page = All

SFR Address = 0x8E RESET = 0000-000X

7	6	5	4	3	2	1	0
P60OC1	P60OC0	P60FD	P34FD	MOVXFD	ADRJ	EXTRAM	--
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R**

Bit 2: ADRJ, ADC result Right-Justified selection.
0: The most significant 8 bits of conversion result are saved in ADCH[7:0], while the least significant 2 bits in ADCL[1:0].
1: The most significant 2 bits of conversion result are saved in ADCH[1:0], while the least significant 8 bits in ADCL[7:0].

If ADRJ = 0

ADCH: ADC Result High Byte Register

SFR Page = All

SFR Address = 0xC6 RESET = xxxx-xxxx

7	6	5	4	3	2	1	0
(B9)	(B8)	(B7)	(B6)	(B5)	(B4)	(B3)	(B2)
R	**R**	**R**	**R**	**R**	**R**	**R**	**R**

ADCL: ADC Result Low Byte Register

SFR Page = All

SFR Address = 0xBE RESET = xxxx-xxxx

7	6	5	4	3	2	1	0
--	--	--	--	--	--	(B1)	(B0)
R	**R**	**R**	**R**	**R**	**R**	**R**	**R**

If ADRJ = 1

7	6	5	4	3	2	1	0
--	--	--	--	--	--	(B8)	(B9)
R	**R**	**R**	**R**	**R**	**R**	**R**	**R**

7	6	5	4	3	2	1	0
(B7)	(B6)	(B5)	(B4)	(B3)	(B2)	(B1)	(B0)
R	**R**	**R**	**R**	**R**	**R**	**R**	**R**

17.4. ADC Sample Code

(1). Required Function: ADC sample code for SYSCLK=24MHz, transfer analog input on P1.0/P1.1/P1.2 with SPEED[1:0]=SYSCLK/270 for 88.9KHz conversion rate.

Assembly Code Example:

```
CHS0            EQU             01h
CHS1            EQU             02h
ADCS            EQU             08h
ADCI            EQU             10h
SPEED0          EQU             20h
SPEED1          EQU             40h
ADCON           EQU             80h

INITIAL_ADC_PIN:
        ORL     P1M0, #00000111B                ; P1.0, P1.1, P1.2 = input only
        ANL     P1M1,#11111000B

        MOV     ADCTL,#ADCON                    ; Enable ADC block
        ; delay 5us
        ; call ....

Get_P10:
        MOV     ADCTL, #(ADCON + SPEED1 + SPEED0)
                                                ; Enable ADC block & start conversion
                                                ; Speed at 88.9k @ 24MHz, select P1.0 for ADC input pin
        CALL    delay_5us
        ORL     ADCTL, #ADCS

        MOV     A, ADCTL                        ; check ready?
        JNB     ACC.4,$-3
        ANL     ADCTL#(0FFh - ADCI - ADCS)      ; clear ADCI & ADCS
        MOV     AIN0_data_V,ADCV                ; reserve P1.0 ADC data
        MOV     AIN0_data_VL, ADCVL
        ; to do ...

Get_P11:
        MOV     ADCTL,#(ADCON + SPEED1 + SPEED0 + CHS0)  ; select P1.1
        CALL    delay_5us
        ORL     ADCTL, #ADCS

        MOV     A, ADCTL                        ; check ready?
        JNB     ACC.4,$-3
        ANL     ADCTL,# (0FFh - ADCI - ADCS)    ; clear ADCI & ADCS
        MOV     AIN1_data_V,ADCV
        MOV     AIN1_data_VL, ADCVL
    ; to do ...

Get_P12:
        MOV     ADCTL,#(ADCON + SPEED1 + SPEED0 + CHS1)  ; select P1.2
        CALL    delay_5us
        ORL     ADCTL, #ADCS

        MOV     ACC,ADCTL                       ; check ready?
        JNB     ACC.4,$-3
        ANL     ADCTL,# (0FFh - ADCI - ADCS)    ; clear ADCI & ADCS
        MOV     AIN2_data_V,ADCV
        MOV     AIN2_data_VL, ADCVL


    ; to do ...

        RET
```

C Code Example:

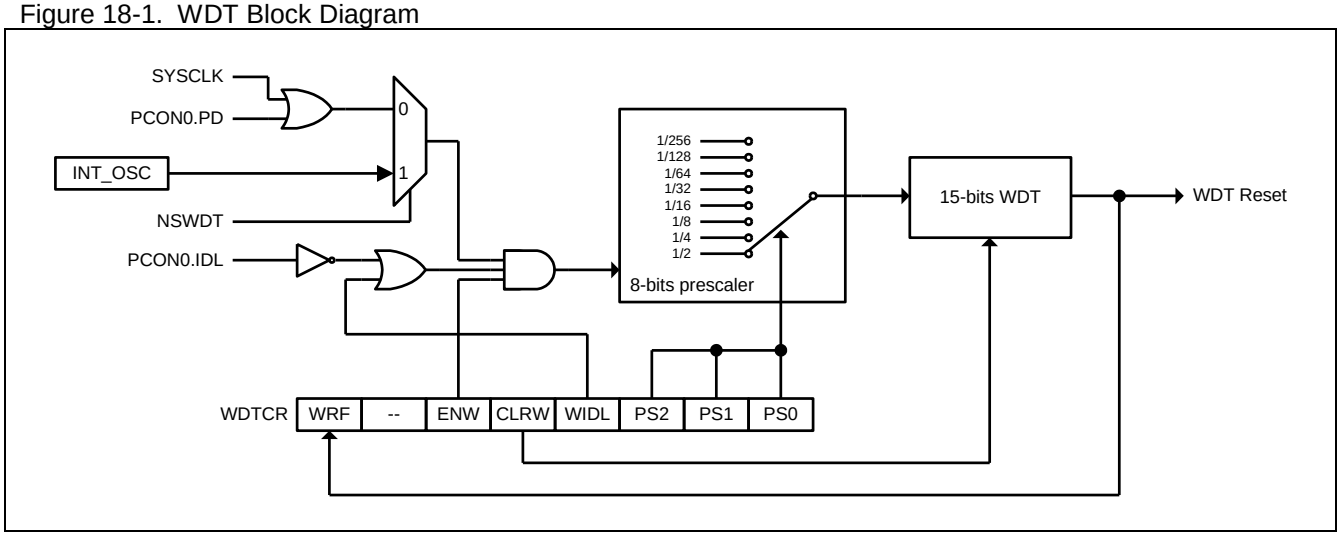
```
#define  CHS0                0x01
#define  CHS1                0x02
#define  ADCS                0x08
#define  ADCI                0x10
#define  SPEED0              0x20
#define  SPEED1              0x40
#define  ADCON               0x80


void main(void)
{
        unsigned char AIN0_data_V, AIN0_data_VL, AIN1_data_V, AIN1_data_VL, AIN2_data_V, AIN2_data_VL;

        P1M0 |= 0x07;                                   // P1.0, P1.1, P1.2 = input only
        P1M1 &= ~0x07;

        ADCTL = ADCON;                                  // Enable ADC block
        // delay 5us
        // ...


        // select P1.0
        ADCTL = (ADCON | SPEED1 | SPEED0);
                                                        // Enable ADC block & start conversion
                                                    // Speed at 88.9k @ 24MHz, select P1.0 for ADC input pin
    Delay_5us();
    ADCTL |= ADCS;


        while ((ADCTL & ADCI) == 0x00);             //wait for complete
        ADCTL &= ~(ADCI | ADCS);
        AIN0_data_V = ADCV;
        AIN0_data_VL = ADCVL;

        // to do ...

        // select P1.1
        ADCTL = (ADCON | SPEED1 | SPEED0 | CHS0);  // select P1.1
    Delay_5us();
    ADCTL |= ADCS;


        while ((ADCTL & ADCI) == 0x00);             //wait for complete
        ADCTL &= ~(ADCI | ADCS);
        AIN1_data_V = ADCV;
        AIN1_data_VL = ADCVL;

        // to do ...

        // select P1.2
        ADCTL = (ADCON | SPEED1 | SPEED0 | CHS1);       // select P1.2
    Delay_5us();
    ADCTL |= ADCS;


        while ((ADCTL & ADCI) == 0x00);             //wait for complete
        ADCTL &= ~(ADCI | ADCS);
        AIN2_data_V = ADCV;
        AIN2_data_VL = ADCVL;

        // to do ...

        while (1);

}
```

18. Watch Dog Timer (WDT)

Watchdog Timer (WDT) is intended as a recovery method in situations (such as power noise/glitches and electrostatic discharge) where the CPU may be subjected to software upset. When software upset happens, the WDT will protect the system from incorrect code execution by causing a system reset. The WDT consists of a 15-bit free-running counter, an 8-bit prescaler and a control register (WDTCR). Figure 18-1 shows the WDT block diagram.

18.1. WDT Structure

Figure 18-1. WDT Block Diagram



18.2. WDT Register

WDTCR: Watch-Dog-Timer Control Register

SFR Page = All

SFR Address = 0xE1 POR = 0X00-0000

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

Bit 7: WRF, WDT reset flag.
0: This bit should be cleared by software.
1: When WDT overflows, this bit is set by hardware to indicate a WDT reset happened.

Bit 6: Reserved. Software must write “0” on this bit when WDTCR is written.

Bit 5: ENW. Enable WDT.
0: ENW can not be cleared by software.
1: Enable WDT while it is set.

Bit 4: CLRW. Clear WDT counter.
0: Hardware will automatically clear this bit.
1: Clear WDT to recount while it is set.

Bit 3: WIDL. WDT idle control.
0: WDT stops counting while the MCU is in idle mode.
1: WDT keeps counting while the MCU is in idle mode.

Bit 2~0: PS2 ~ PS0, select prescaler output for WDT time base input.

PS[2:0]	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

18.3. WDT Hardware Option

WDSFWP, disable Software write on WDTCR.
Enable: The software SFR - WDTCR will be write-protected except the bit – CLRW.
Disable: The software SFR – WDTCR is free for writing of software.

NSWDT, Non-Stopped WDT
Enable: Keep WDT running in power down mode and select internal high frequency RC Oscillator for the clock source of WDT.
Disable: Disable WDT running in power down mode.

HWENW, Hardware loaded for “ENW” of WDTCR.
Enable: Clearing it will enable WDT and load the content of HWWIDL, HWPS2, HWPS1 and HWPS0 to WDTCR SFR when power-up.
Disable: WDT is not enabled automatically during power-up.

HWWIDL, HWPS2, HWPS1, HWPS0
When HWENW is enabled, the content on these four fused bits will be loaded to WDTCR SFR during power-up.

18.4. WDT Sample Code

(1) Required function: Enable WDT and select WDT prescalar to 1/32

Assembly Code Example:

```
PS0             EQU     01h
PS1             EQU     02h
PS2             EQU     04h
WIDL            EQU     08h
CLRW            EQU     10h
ENW             EQU     20h
WRF             EQU     80h

   ANL    WDTCR,#(0FFh - WRF)          ; Clear WRF flag (write “0”)
   MOV    WDTCR,#(ENW + CLRW + PS2)    ; Enable WDT counter and set WDT prescaler to 1/32
```

C Code Example:

```
#define  PS0            0x01
#define  PS1            0x02
#define  PS2            0x04
#define  WIDL           0x08
#define  CLRW           0x10
#define  ENW            0x20
#define  WRF            0x80

   WDTCR &= ~WRF;                    // Clear WRF flag (write “0”)
   WDTCR = (ENW | CLRW | PS2);       // Enable WDT counter and set WDT prescaler to 1/32
                                     // PS[2:0]  | WDT prescaler selection
                                     //   0    | 1/2
                                     //   1    | 1/4
                                     //   2    | 1/8
                                     //   3    | 1/16
                                     //   4    | 1/32
                                     //   5    | 1/64
                                     //   6    | 1/128
                                     //   7    | 1/256
```

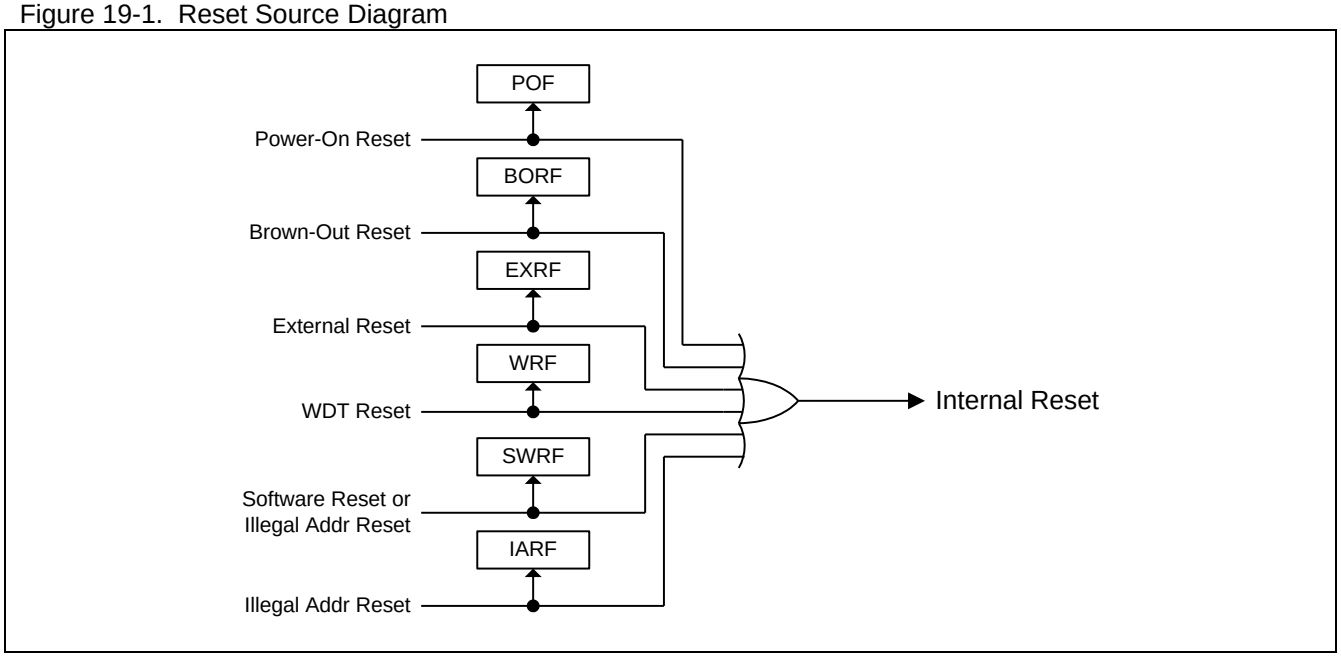
19. Reset

During reset, all I/O Registers are set to their initial values, the port pins are weakly pulled to VDD, and the program starts execution from the Reset Vector, 0000H, or ISP start address by Hardware Option setting. The MG82Fx564 has six sources of reset: power-on reset, WDT reset, software reset, external reset, brown-out reset and illegal address reset.

19.1. Reset Source

There are six reset sources in MG82Fx564 to generate an internal reset to initial CPU and registers. Figure 19-1 shows the reset source diagram.

Figure 19-1. Reset Source Diagram



19.2. Power-On Reset

Power-on reset (POR) is used to internally reset the CPU during power-up. The CPU will keep in reset state and will not start to work until the VDD power rises above the voltage of Power-On Reset. And, the reset state is activated again whenever the VDD power falls below the POR voltage. During a power cycle, VDD must fall below the POR voltage before power is reapplied in order to ensure a power-on reset

PCON0: Power Control Register 0

SFR Page = All

SFR Address = 0x87 POR = 0001-0000, RESET = 000X-0000

7	6	5	4	3	2	1	0
SMOD1	SMOD0	GF	**POF**	GF1	GF0	PD	IDL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4: POF, Power-On Flag.
0: The flag must be cleared by software to recognize next reset type.
1: Set by hardware when VDD rises from 0 to its nominal voltage. POF can also be set by software.

The Power-on Flag, POF, is set to “1” by hardware during power up or when VDD power drops below the POR voltage. It can be clear by firmware and is not affected by any warm reset such as external reset, Brown-Out reset, software reset (ISPCR.5) and WDT reset. It helps users to check if the running of the CPU begins from power up or not. Note that the POF must be cleared by firmware.

19.3. WDT Reset

When WDT overflows, it will cause a system warm reset and set a flag, WRF, to indicate a WDT reset happened.

WDTCR: Watch-Dog-Timer Control Register

SFR Page = All

SFR Address = 0xE1 POR = 0x00-0000

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

Bit 7: WRF, WDT reset flag.
0: This bit should be cleared by software.
1: When WDT overflows, this bit is set by hardware to indicate a WDT reset happened.

19.4. Software Reset

Software can trigger a system warm reset by writing a “1” on SWRST of ISPCR. After the software reset completed, hardware sets a flag, SWRF in PCON1, to indicate a software reset happened.

ISPCR: ISP Control Register

SFR Page = All

SFR Address = 0xE7 RESET = 0000-xxxx

7	6	5	4	3	2	1	0
ISPEN	SWBS	**SWRST**	CFAIL	-	--	--	--
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R**	**R**

Bit 5: SWRST, software reset trigger control.
0: No operation
1: Generate software system reset. It will be cleared by hardware automatically.

PCON1: Power Control Register 1

SFR Page = All

SFR Address = 0x97 POR = 0000-xxx0

7	6	5	4	3	2	1	0
SWRF	EXRF	BORF	IARF	--	--	--	BOD
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**

Bit 7: SWRF, Software Reset Flag.
0: This bit must be cleared by software.
1: This bit is set if a Software Reset occurs.

19.5. External Reset

A reset is accomplished by holding the RESET pin HIGH for at least 24 oscillator periods while the oscillator is running. To ensure a reliable power-up reset, the hardware reset from RST pin is necessary. After the external reset completely, hardware sets a flag, EXRF in PCON1, to indicate a external reset happened.

PCON1: Power Control Register 1

SFR Page = All

SFR Address = 0x97 POR = 0000-XXX0

7	6	5	4	3	2	1	0
SWRF	**EXRF**	BORF	IARF	--	--	--	BOD
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**

Bit 6: EXRF, External Reset Flag.
0: This bit must be cleared by software.
1: This bit is set if an External Reset occurs.

19.6. Brown-Out Reset

In MG82Fx564, if VDD power drops below 4.2V in E series (2.4V in L series), it sets a flag, BOD in PCON1. If BORE of AUXRA is enabled, BOD event will triggers a RESET to CPU and set a flag, BORF, to indicate a Brown-Out Reset happened.

PCON1: Power Control Register 1

SFR Page = All

SFR Address = 0x97 POR = 0000--xxx0

7	6	5	4	3	2	1	0
SWRF	EXRF	**BORF**	IARF	--	--	--	**BOD**
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**

Bit 5: BORF, Brown-Out Reset Flag.
0: This bit must be cleared by software.
1: Set for the event flag of brown-out reset.

Bit 0: BOD, Brown-Out Detection flag.
0: This bit must be cleared by software.
1: This bit is set if the operating voltage matches the detection level of Brown-Out Detector.

AUXRA: Auxiliary Register A

SFR Address = IFMT POR = 0010--0100

7	6	5	4	3	2	1	0
DBOD	**BORE**	OCDE	ILRCOE	XTALE	IHRCOE	OSCS1	OSCS0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 6: BORE, Brown-Out Reset Enable.
0: Disable Reset action if BOD occurs.
1: Enable a Reset action if BOD occurs.

19.7. Illegal Address Reset

In MG82Fx564, if software program runs to the illegal address such as over program ROM limitation, it triggers a warm reset to CPU and set a flag, IARF in PCON1, to indicate a illegal address reset happened.

PCON1: Power Control Register 1

SFR Page = All

SFR Address = 0x97 POR+ = 0000-xxx0

7	6	5	4	3	2	1	0
SWRF	EXRF	BORF	**IARF**	--	--	--	BOD
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**

Bit 4: IARF, Illegal Address Reset Flag.
0: This bit must be cleared by software.
1: This bit is set if a PC illegal address Reset occurs.

19.8. Reset Sample Code

(1) Required function: Trigger a software reset

Assembly Code Example:

```
SWRST           EQU                     20h

   ORL     ISPCR,#SWRST            ; Trigger Software Reset
```

C Code Example:

```
#define   SWRST                 0x20

   ISPCR |= SWRST;              // Trigger Software Reset
```

20. Power Management

The MG82Fx564 supports one power monitor module, Brown-Out Detector, and two power-reducing modes: Idle mode and Power-down mode. These modes are accessed through the PCON0 and PCON1 registers to handle the chip power event.

20.1. Power Saving Mode

20.1.1. Idle Mode

Setting the IDL bit in PCON enters idle mode. Idle mode halts the internal CPU clock. The CPU state is preserved in its entirety, including the RAM, stack pointer, program counter, program status word, and accumulator. The Port pins hold the logical states they had at the time that Idle was activated. Idle mode leaves the peripherals running in order to allow them to wake up the CPU when an interrupt is generated. Timer 0, Timer 1, Timer 2, SPI, KBI, ADC, UART0 and the UART1 will continue to function during Idle mode. PCA Timer and WDT are conditional enabled during Idle mode to wake up CPU. Any enabled interrupt source or reset may terminate Idle mode. When exiting Idle mode with an interrupt, the interrupt will immediately be serviced, and following RETI, the next instruction to be executed will be the one following the instruction that put the device into Idle. There is another Idle existing mechanism by enabled wakeup GPIOs that don't builds interrupt capability.

The channel inputs of ADC should be set to "output 0" or "quasi-bidirectional" when ADC is disabled in idle mode or power-down mode.

20.1.2. Power-down Mode

Setting the PD bit in PCON0 enters Power-down mode. Power-down mode stops the oscillator and powers down the internal macros in order to minimize power consumption. Only the power-on circuitry will continue to draw power during Power-down. During Power-down the power supply voltage may be reduced to the RAM keep-alive voltage. The RAM contents will be retained; however, the SFR contents are not guaranteed once VDD has been reduced out of chip operating voltage range. Power-down may be exited by external reset, power-on reset, enabled external interrupts, enabled KBI or enabled Non-Stop WDT.

The user should not attempt to enter (or re-enter) the power-down mode for a minimum of 4 µs until after one of the following conditions has occurred: Start of code execution (after any type of reset), or Exit from power-down mode.

20.1.3. Interrupt Recovery from Power-down

Four external interrupts may be configured to terminate Power-down mode. External interrupts nINT0 (P3.2), nINT1 (P3.3), nINT2 (P4.3) and nINT3 (P4.2) may be used to exit Power-down. To wake up by external interrupt nINT0, nINT1, nINT2 or nINT3, the interrupt must be enabled and configured for level-sensitive operation.

When terminating Power-down by an interrupt, the wake up period is internally timed. At the recognized level on the interrupt pin, Power-down is exited, the oscillator is restarted, and an internal timer begins counting. The internal clock will not be allowed to propagate and the CPU will not resume execution until after the timer has reached internal counter full. After the timeout period, the interrupt service routine will begin. To prevent the interrupt from re-triggering, the ISR should disable the interrupt before returning. The interrupt pin should be held low (or high for high level or rising edge selected) until the device has timed out and begun executing.

20.1.4. Reset Recovery from Power-down

Wakeup from Power-down through an external reset is similar to the interrupt. At the rising edge of RST, Power-down is exited, the oscillator is restarted, and an internal timer begins counting. The internal clock will not be allowed to propagate to the CPU until after the timer has reached internal counter full. The RST pin must be held high for longer than the timeout period to ensure that the device is reset properly. The device will begin executing once RST is brought low.

It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

20.1.5. KBI wakeup Recovery from Power-down

The Keypad Interrupt of MG82Fx564, P2.7 ~ P2.0 have wakeup CPU capability that are enabled by the control registers in KBI module.

Wakeup from Power-down through an enabled wakeup KBI is same to the interrupt. At the matched condition of enabled KBI pattern and enabled KBI interrupt (EIE1.5, EKB), Power-down is exited, the oscillator is restarted, and an internal timer begins counting. The internal clock will not be allowed to propagate to the CPU until after the timer has reached internal counter full. After the timeout period, CPU will meet a KBI interrupt and execute the interrupt service routine.

20.2. Power Monitor Module

MG82Fx564 has an On-Chip Brown-Out Detection (BOD) for monitoring the Vcc level during operation by comparing it to a fixed trigger level. The trigger level is 4.2V in E-series and 2.4V in L-series. When VDD decreases to a level below the trigger, the BOD of PCON1 is set and requests an interrupt if EBD of EIE1 is enabled. Software can service the interrupt to respond to the BOD event. Until VDD exits the BOD level, hardware will block any write operation to on-chip flash memory.

When BORE of AUXRA is enabled, BOD has the capability to reset MCU. When VDD decreases below BOD level in this case, the Brown-Out Reset is activated. When VDD increases above the trigger level, MCU re-starts the code execution from program counter: 0000H. And hardware sets a Brown-Out Reset Flag, BORF of PAOCN1 to indicate the brown-out reset just finished.

20.3. Power Control Register

PCON0: Power Control Register 0

SFR Page = All

SFR Address = 0x87 POR = 0001-0000, RESET = 000x-0000

7	6	5	4	3	2	1	0
SMOD1	SMOD0	--	POF	GF1	GF0	PD	IDL
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 1: PD, Power-Down control bit.
0: This bit could be cleared by CPU or any exited power-down event.
1: Setting this bit activates power down operation.

Bit 0: IDL, Idle mode control bit.
0: This bit could be cleared by CPU or any exited Idle mode event.
1: Setting this bit activates idle mode operation.

PCON1: Power Control Register 1

SFR Page = All

SFR Address = 0x97 POR = 0000-xxx0

7	6	5	4	3	2	1	0
SWRF	EXRF	BORF	IARF	--	--	--	BOD
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**

Bit 7: SWRF, Software Reset Flag.
0: This bit must be cleared by software.
1: This bit is set if a Software Reset occurs.

Bit 6: EXRF, External Reset Flag.
0: This bit must be cleared by software.
1: This bit is set if an External Reset occurs.

Bit 5: BORF, Brown-Out Reset Flag.
0: This bit must be cleared by software.
1: This bit is set if a Brown-Out Reset occurs.

Bit 4: IARF, Illegal Address Reset Flag.
0: This bit must be cleared by software.
1: This bit is set if a PC illegal address Reset occurs.

Bit 3~1: Reserved.

Bit 0: BOD, Brown-Out Detection flag.
0: This bit must be cleared by software.
1: This bit is set if the operating voltage matches the detection level of Brown-Out Detector.

20.4. Power Control Sample Code

(1) Required function: Select Slow mode with OSCin/128 (default is OSCin/1)

Assembly Code Example:

```
CKS0                EQU         01h
CKS1                EQU         02h
CKS2                EQU         04h

   ORL     PCON2,#( CKS2 + CKS1 + CKS0)   ; Set CKS[2:0] = “111” to select OSCin/128
```

C Code Example:

```
#define  CKS0                           0x01
#define  CKS1                           0x02
#define  CKS2                           0x04

   PCON2 |= (CKS2 | CKS1 | CKS0);         // System clock divider /128
                                 // CKS[2:0], system clock divider
                                 //   0    | OSCin/1
                                 //   1    | OSCin/2
                                 //   2    | OSCin/4
                                 //   3    | OSCin/8
                                 //   4    | OSCin/16
                                 //   5    | OSCin/32
                                 //   6    | OSCin/64
                                 //   7    | OSCin/128
```

21. System Clock

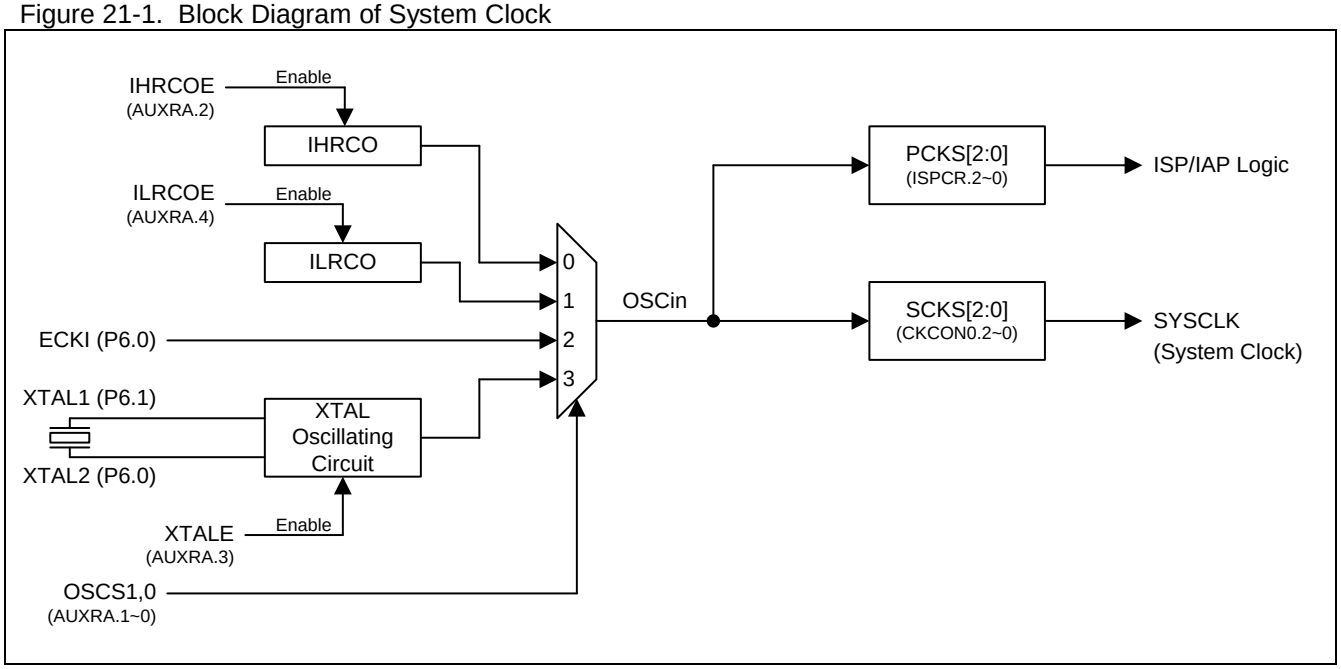
21.1. Clock Structure

There are four clock sources in MG82Fx564 for the system clock: internal high frequency RC oscillator (IHRCO), internal low frequency RC oscillator (ILRCO), external clock input and external crystal oscillator. The system clock, SYSCLK, is obtained from one of these four clock sources through the clock divider, as shown in Figure 21-1. The user can program the divider control bits SCKD2~SCKD0 (in PCON2 register) to get the desired system clock.

The IHRCO (22.12MHz) is enabled and set as the default system clock source after power-on. Software can enable other oscillating circuits and switches them on the fly by programming AUXRA. Such as user selected the external crystal as system clock, software must enable external crystal oscillating circuit first and wait it stable. Then program the OSCS[1:0] to switch the clock source to external crystal. Before software switches the clock source selection, user must be careful to confirm the selected clock source is ready and stable. Otherwise it will cause system fault or CPU hung. After the clock selection, software may disable the un-used oscillating circuit to reduce the power consumption.

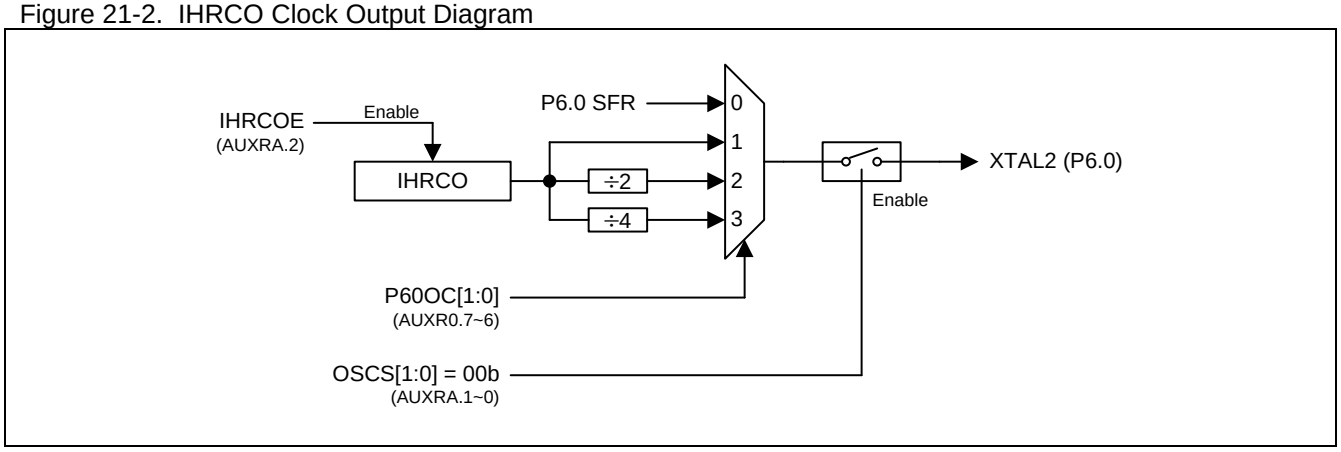
The IHRCO in MG82Fx564 supports internal RC-OSC clock frequencies for user application. The 22.12MHz frequency in IHRCO for system clock is the default setting in Megawin shipped samples..

Figure 21-1. Block Diagram of System Clock



In IHRCO mode, XTAL2 and XTAL1 are the GPIO function on P6.0 and P6.1. By the way, P6.0 can be programmed to output the IHRCO clock output with divided 1, 2 or 4 by software selection on P60OC[1:0] of AUXR1 SFR. Figure 21-2 shows the IHRCO output diagram.

Figure 21-2. IHRCO Clock Output Diagram



21.2. Clock Control Register

***PCON2*: Clock Control Register 2**

SFR Page = All

SFR Address = 0xC7 RESET = xxxx-x000

7	6	5	4	3	2	1	0
OSCDR	-	-	-	-	SCKS2	SCKS1	SCKS0
R/W	**R**	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**

Bit 7: OSCDR, OSC Driving control Register. Default value is load from OSCDN (in hardware option). And it could be read/written by CPU.

0: The driving of crystal oscillator is enough for oscillation up to 24MHz.

1: The driving of crystal oscillator is reduced. It will helpful in EMI reduction. Regarding application not needing high frequency clock, it is recommended to do so.

Bit 6~3: Reserved.

Bit 2~0: SCKS2 ~ SCKS0, programmable System Clock Selection.

SCKS[2:0]	System Clock (Fosc)
0 0 0	OSCin (Default)
0 0 1	OSCin /2
0 1 0	OSCin /4
0 1 1	OSCin /8
1 0 0	OSCin /16
1 0 1	OSCin /32
1 1 0	OSCin /64
1 1 1	COSin /128

AUXR0: Auxiliary Register 0

SFR Page = All

SFR Address = 0x8E RESET = 0000-000x

7	6	5	4	3	2	1	0
P60OC1	P60OC0	P60FD	P34FD	MOVXFD	ADRJ	EXTRAM	--
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R**

Bit 7~6: P60 output configured control bit 1 and 0. The two bits only act when IHRCO is selected for system clock source. In this condition, XTAL2 and XTAL1 are the alternated function for P60 and P61. P60 provides the following selections for GPIO or clock source generator. When P60OC[1:0] index to non-P60 function, XTAL2 will drive the internal high frequency RC oscillator output to provide the clock source for other devices.

P60OC[1:0]	XTAL2 function
00	P60 (Default)
01	IHRCO
10	IHRCO/2

11	IHRCO/2

Bit 5: P60FD, P6.0 Fast Driving.
0: P6.0 output with default driving.
1: P6.0 output with fast driving enabled. If P6.0 is configured to clock output, enable this bit when P6.0 output frequency is more than 12MHz at 5V application or more than 6MHz at 3V application.

AUXRA: Auxiliary Register A

SFR Address = IFMT RESET = 0010--0100

7	6	5	4	3	2	1	0
DBOD	BORE	OCDE	ILRCOE	XTALE	IHRCOE	OSCS1	OSCS0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 4: LIRCOE, Internal Low frequency RC Oscillator Enable.
0: Disable internal low frequency RC oscillator.
1: Enable internal low frequency RC oscillator. It is about 125 KHz. It needs **50 us** to have stable output after ILRCOE is enabled.

Bit 3: XTALE, external Crystal(XTAL) Enable.
0: Disable XTAL oscillating circuit. In this case, XTAL2 and XTAL1 behave as Port 6.0 and Port 6.1.
1: Enable XTAL oscillating circuit. If this bit is set by CPU software, it needs **5 ms** to have stable output after XTALE is enabled.

Bit 3: IHRCOE, Internal High frequency RC Oscillator Enable.
0: Disable internal high frequency RC oscillator.
1: Enable internal low frequency RC oscillator. If this bit is set by CPU software, it needs **50 us** to have stable output after IHRCOE is enabled.

Bit 1~0: OSC input selection.

OSCS[1:0]	OSCin Source	P6.0 Function	P6.1 Function
00	IHRCO (Default)	P6.0 or IHRCO output	P6.1
01	ILRCO	P6.0	P6.1
10	External Clock Input	Clock Input	P6.1
11	Ext. Crystal Oscillating	XTAL2	XTAL1

21.3. Sample code for switching internal RC-OSC Clock to External XTAL

```
;*******************************************************************************
; Demo code for selecting Megawin MG82Fx564 system clock from internal oscillator to external oscillator.
; If user wants to use External Xtal as system clock source, user should Inserted below demo code in program start
;*******************************************************************************
$INCLUDE (REG_MG82Fx564.INC) ;for MG82Fx564 SFR definition
ISP_StandBy     EQU     00h
AUXRA_Wr        EQU     06h
AUXRA_Rd        EQU     07h
;===============================================================================
        CSEG    AT      0000h
        JMP     start
;===============================================================================
code_main       SEGMENT         CODE
        USING   0
start:
        MOV     SP,#stack_space-1

        CLR     TR0
        ORL     TMOD, #01h
        MOV     TH0, #0DCh                      ;use timer0 to delay 5ms
        MOV     TL0, #00h
        CLR     TF0

        ORL     ISPCR, #80h                     ;enable ISP/IAP
        MOV     IFMT, #AUXRA_Rd                 ;set read AUXRA command
        MOV     SCMD, #46h
        MOV     SCMD, #0B9h
        MOV     A, IFD
        ORL     A, #08h                         ;enable XTALE
        MOV     IFD, A
        MOV     IFMT, #AUXRA_Wr                 ;set write AUXRA command
        MOV     SCMD, #46h
        MOV     SCMD, #0B9h

        SETB    TR0                                     ;timer0 run
        JNB     TF0, $                          ;delay 5ms here, for external XTAL stable
        CLR     TF0
        CLR     TR0                                 ;timer0 stop

        MOV     IFMT, #AUXRA_Rd                 ;set read AUXRA command
        MOV     SCMD, #46h
        MOV     SCMD, #0B9h
        MOV     A, IFD
        ORL     A, #03h                         ;select crystal as OSCin
        MOV     IFD, A
        MOV     IFMT, #AUXRA_Wr                 ;set write AUXRA command
        MOV     SCMD, #46h
        MOV     SCMD, #0B9h

        MOV     IFMT, #AUXRA_Rd                 ;set read AUXRA command
        MOV     SCMD, #46h
        MOV     SCMD, #0B9h
        MOV     A, IFD
        ANL     A, #0FBh                                ;disable IHRCO
        MOV     IFD, A
        MOV     IFMT, #AUXRA_Wr                 ;set write AUXRA command
        MOV     SCMD, #46h
        MOV     SCMD, #0B9h
;===============================================================================
;    User code start from here ……..
```

22. In System Programming (ISP)

The ISP in MG82Fx564 makes it possible to update the user's application program (in AP-memory) and non-volatile application data (in IAP-memory) without removing the MCU chip from the actual end product. This useful capability makes a wide range of field-update applications possible. (Note ISP needs the loader program pre-programmed in the ISP-memory.) In general, the user needn't know how ISP operates because Megawin has provided the standard ISP tool and embedded ISP code in Megawin shipped samples.

22.1. ISP (IAP) Control Register

The following special function registers are related to the ISP operation. All these registers can be accessed by software in the user's application program.

IFD: ISP/IAP Flash Data Register

SFR Page = All

SFR Address = 0xE2 RESET = 1111-1111

7	6	5	4	3	2	1	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

IFD is the data port register for ISP/IAP operation. The data in IFD will be written into the desired address in operating ISP/IAP write and it is the data window of readout in operating ISP/IAP read.

If IMFT is indexed on IAPLB, AUXRA or AUXRB access, read/write IFD through SCMD flow will access the register content of IAPLB, AUXRA or AUXRB.

IFADRH: ISP/IAP Address for High-byte addressing

SFR Page = All

SFR Address = 0xE3 RESET = 0000-0000

7	6	5	4	3	2	1	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

IFADRH is the high-byte address port for all ISP/IAP modes.

IFADRL: ISP/IAP Address for Low-byte addressing

SFR Page = All

SFR Address = 0xE4 RESET = 0000-0000

7	6	5	4	3	2	1	0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

IFADRL is the low byte address port for all ISP/IAP modes. In page erase operation, it is ignored.

IFMT: ISP/IAP Flash Mode Table

SFR Page = All

SFR Address = 0xE5 RESET = xxx0-0000

7	6	5	4	3	2	1	0
--	--	--	MS[4]	MS[3]	MS[2]	MS[1]	MS[0]
R	**R**	**R**	**R**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~4: Reserved

Bit 4~0: ISP/IAP operating mode selection. IFMT is used to select the flash mode for performing numerous ISP/IAP function or used to access protected SFRs.

Bit[4:0]	Mode
0 0 0 0 0	Standby
0 0 0 0 1	AP-memory read
0 0 0 1 0	AP-memory program
0 0 0 1 1	AP-memory page erase
0 0 1 0 0	IAPLB write (protected SFR)
0 0 1 0 1	IAPLB read (protected SFR)
0 0 1 1 0	AUXRA write (protected SFR)
0 0 1 1 1	AUXRA read (protected SFR)
1 0 0 0 0	AUXRB write (protected SFR)
1 0 0 0 1	AUXRB read (protected SFR)
Others	**Reserved for test mode. Must not program them.**

SCMD: Sequential Command Register

SFR Page = All

SFR Address = 0xE6 RESET = xxxx-xxxx

7	6	5	4	3	2	1	0
SCMD							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SCMD is the command port for triggering ISP/IAP activity and protected SFRs access. If SCMD is filled with sequential 0x46h, 0xB9h and if ISPCR.7 = 1, ISP/IAP activity or protected SFRS access will be triggered.

ISPCR: ISP Control Register

SFR Page = All

SFR Address = 0xE7 RESET = 0000-x000

7	6	5	4	3	2	1	0
ISPEN	SWBS	SWRST	CFAIL	--	PCKS2	PCKS1	PCKS0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W

Bit 7: ISPEN, ISP/IAP operation enable.
0: Global disable all ISP/IAP program/erase/read function.
1: Enable ISP/IAP program/erase/read function.

Bit 6: SWBS, software boot selection control.
0: Boot from main-memory after reset.
1: Boot from ISP memory after reset.

Bit 5: SWRST, software reset trigger control.
0: No operation
1: Generate software system reset. It will be cleared by hardware automatically.

Bit 4: CFAIL, Command Fail indication for ISP/IAP operation.
0: The last ISP/IAP command has finished successfully.
1: The last ISP/IAP command fails. It could be caused since the access of flash memory was inhibited.

Bit 3: Reserved. Software must write “0” on this bit when ISPCR is written.

Bit 2~0: PCKS2~0, ISP/IAP programming clock source selection.

PCKS[2:0]	OSCin Frequency (MHz)
0 0 0	> 24MHz
0 0 1	20 ~ 24
0 1 0	12 ~ 20
0 1 1	6~ 12
1 0 0	3 ~ 6
1 0 1	2 ~ 3
1 1 0	1 ~ 2
1 1 1	< 1

IAPLB: IAP Low Boundary

SFR Address=indirect RESET = 1111-111x

7	6	5	4	3	2	1	0
IAPLB							--
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

Bit 7~0: The IAPLB determines the IAP-memory lower boundary. Since a Flash page has 512 bytes, the IAPLB must be an even number.

To read IAPLB, MCU need to define the IMFT for mode selection on IAPLB Read and set ISPCR.ISPEN. And then write 0x46h & 0xB9h sequentially into SCMD. The IAPLB content is available in IFD. If write IAPLB, MCU will put new IAPLB setting value in IFD firstly. And then select IMFT, enable ISPCR.ISPEN and then set SCMD. The IAPLB content has already finished the updated sequence.

There are 1.5K bytes for IAP space higher than AP boundary address in MG82Fx564. If the IAP size is not enough for user application and AP region has redundant flash memory more than one page, software can modify IAPLB to move some AP region flash for IAP application memory.
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 in MG82FE/L564, if IAPLB=0xE0 and ISP start address is F000H, then the IAP-memory range is located at E000H ~ EFFFH.

Additional attention point, the IAP low boundary address must not be higher than ISP start address or other non-device defined space. Otherwise, it may cause to corrupt data content in flash memory.

23. In Application Programming (IAP)

MG82FE/L564 available program memory size (AP-memory) is restricted to 64K. The flash memory between IAPLB and ISP start address could be defined as data flash memory and can be accessed by the ISP operation in field application. The size of IAP flash memory is variable. It is defined by IAPLB.

23.1. ISP/IAP Sample Code

(1). Required Function: General function call for ISP/IAP flash read

Assembly Code Example:

```
IxP_Flash_Read  EQU                     01h
ISPEN           EQU                     80h

_ixp_read:
ixp_read:

   MOV    ISPCR,#ISPEN              ; Enable Function
   MOV    IFMT,# IxP_Flash_Read         ; ixp_read=0x01

   MOV    IFADRH,??                 ; fill [IFADRH,IFADRL] with byte address
   MOV    IFADRL,??

   MOV    SCMD,#046h                ;
   MOV    SCMD,#0B9h                ;

     MOV    A,IFD                   ; now, the read data exists in IFD

   MOV    IFMT,#000h                ; Flash_Standby=0x00
   ANL    ISPCR,#(0FFh – ISPEN)         ; Disable Function

   RET
```

C Code Example:

```
#define  Flash_Standby                  0x00
#define  IxP_Flash_Read                 0x01
#define  ISPEN                          0x80


unsigned char ixp_read (void)
{
   unsigned char arg;
     ISPCR = ISPEN;                  // Enable Function
   IFMT = IxP_Flash_Read;              // IxP_Read=0x01

   IFADRH = ??
   IFADRL = ??

   SCMD = 0x46;                    //
   SCMD = 0xB9;                    //

   arg = IFD;

   IFMT = Flash_Standby;               // Flash_Standby=0x00
   ISPCR &= ~ISPEN;

   return arg;
}
```

(2). Required Function: General function call for ISP/IAP flash Erase

Assembly Code Example:

```
IxP_Flash_ Erase EQU                    03h
ISPEN            EQU                    80h

_ixp_erase:
ixp_erase:

   MOV    ISPCR,#ISPEN              ; Enable Function
   MOV    IFMT,# IxP_Flash_Erase        ; ixp_erase=0x03

   MOV    IFADRH,??                 ; fill [IFADRH,IFADRL] with byte address
   MOV    IFADRL,??

   MOV    SCMD,#046h                ;
   MOV    SCMD,#0B9h                ;

   MOV    IFMT,#000h                ; Flash_Standby=0x00
   ANL    ISPCR,#(0FFh – ISPEN)         ; Disable Function

   RET
```

C Code Example:

```
#define  Flash_Standby                  0x00
#define  IxP_Flash_Erase                0x03
#define  ISPEN                          0x80


void ixp_erase (unsigned char Addr_H, unsigned char Addr_L)
{
   ISPCR = ISPEN;                  // Enable Function
   IFMT = IxP_Flash_Erase;             // IxP_Erase=0x03

   IFADRH = Addr_H;
   IFADRL = Addr_L;

   SCMD = 0x46;                    //
   SCMD = 0xB9;                    //

   IFMT = Flash_Standby;               // Flash_Standby=0x00
   ISPCR &= ~ISPEN;
}
```

(3). Required Function: General function call for ISP/IAP flash program

Assembly Code Example:

```
IxP_Flash_Program       EQU                 02h
ISPEN                   EQU                 80h

_ixp_program:
ixp_program:

   MOV    ISPCR,#ISPEN              ; Enable Function
   MOV    IFMT,# IxP_Flash_Program      ; ixp_program=0x03

   MOV    IFADRH,??                 ; fill [IFADRH,IFADRL] with byte address
   MOV    IFADRL,??
     MOV    IFD, A                  ; now, the program data exists in Accumulator

   MOV    SCMD,#046h                ;
   MOV    SCMD,#0B9h                ;

   MOV    IFMT,#000h                ; Flash_Standby=0x00
   ANL    ISPCR,#(0FFh – ISPEN)         ; Disable Function

   RET
```

C Code Example:

```
#define  Flash_Standby                  0x00
#define  IxP_Flash_Program              0x02
#define  ISPEN                          0x80

void ixp_program(unsigned char Addr_H, unsigned char Addr_L, unsigned char dta)
{
   ISPCR = ISPEN;                  // Enable Function
   IFMT = IxP_Flash_Program;           // IxP_Program=0x02

   IFADRH = Addr_H;
   IFADRL = Addr_L;
   IFD = dta;

   SCMD = 0x46;                    //
   SCMD = 0xB9;                    //

   IFMT = Flash_Standby;               // Flash_Standby=0x00
   ISPCR &= ~ISPEN;


}
```

24. Auxiliary SFRs

AUXR0: Auxiliary Register 0

SFR Page = All

SFR Address = 0x8E RESET = 0000-000x

7	6	5	4	3	2	1	0
P60OC1	P60OC0	P60FD	P34FD	MOVXFD	ADRJ	EXTRAM	--
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R**

Bit 7~6: P60 output configured control bit 1 and 0. The two bits only act when IHRCO is selected for system clock source. In this condition, XTAL2 and XTAL1 are the alternated function for P60 and P61. P60 provides the following selections for GPIO or clock source generator. When P60OC[1:0] index to non-P60 function, XTAL2 will drive the internal high frequency RC oscillator output to provide the clock source for other devices.

P60OC[1:0]	XTAL2 function
00	P60 (Default)
01	INTOSC
10	INTOSC/2
11	INTOSC/4

Bit 5: P60FD, P6.0 Fast Driving.
0: P6.0 output with default driving.
1: P6.0 output with fast driving enabled. If P6.0 is configured to clock output, enable this bit when P6.0 output frequency is more than 12MHz at 5V application or more than 6MHz at 3V application.

Bit 4: P34FD, P3.4 Fast Driving.
0: P3.4 output with default driving.
1: P3.4 output with fast driving enabled. If P3.4 is configured to T0CKO, enable this bit when P3.4 output frequency is more than 12MHz at 5V application or more than 6MHz at 3V application.

Bit 3: MOVXFD, Fast Driving enabled for MOVX output signals.
0: MOVX output signals with default driving.
1: MOVX output signals with fast driving. If there is an off-chip memory access, MOVX@DPTR or MOVX@Ri, the MOVX output signals require fast driving for stretched ALE/RD/WR pulse frequency more than 12MHz @5V or 6MHz @3V.

Bit 2: ADRJ, ADC result Right-Justified selection.
0: The most significant 8 bits of conversion result are saved in ADCH[7:0], while the least significant 2 bits in ADCL[1:0].
1: The most significant 2 bits of conversion result are saved in ADCH[1:0], while the least significant 8 bits in ADCL[7:0].

Bit 1: EXTRAM, External data RAM enable.
0: Enable on-chip expanded data RAM (XRAM 1024 bytes)
1: Disable on-chip expanded data RAM.

Bit 0: Reserved. Software must write “0” on this bit when AUXR0 is written.

AUXR1: Auxiliary Control Register 1

SFR Page = All

SFR Address = 0xA2 RESET = 0000-xxx0

7	6	5	4	3	2	1	0
P4KBI	P4PCA	P5SPI	P4S1	--	--	--	DPS
R/W	**R/W**	**R/W**	**R/W**	**R**	**R**	**R**	**R/W**

Bit 7: P4KBI, KBI function on P4/P5.
0: Disable KBI function moved to P4/P5.
1: Set KBI function on P4/P5 as following definition.
‘KBI0’ function in P2.0 is moved to P4.0.
‘KBI1’ function in P2.1 is moved to P4.1.

‘KBI2’ function in P2.2 is moved to P4.2.
‘KBI3’ function in P2.3 is moved to P4.3.
‘KBI5’ function in P2.5 is moved to P5.1.
‘KBI4’ function in P2.4 is moved to P5.0.
‘KBI6’ function in P2.6 is moved to P5.2.
‘KBI7’ function in P2.7 is moved to P5.3.

Bit 6: P4PCA, PCA function on P4/P5.
0: Disable PCA function moved to P4/P5.
1: Set PCA function on P4/P5 as following definition.
‘ECI’ function in P1.1 is moved to P4.2.
‘CEX0’ function in P1.2 is moved to P4.0.
‘CEX1’ function in P1.3 is moved to P4.1.
‘CEX2’ function in P1.4 is moved to P5.0.
‘CEX3’ function in P1.5 is moved to P5.1
‘CEX4’ function in P1.6 is moved to P5.2.
‘CEX5’ function in P1.7 is moved to P5.3.

Bit 5: P5SPI, SPI interface on P5.3~P5.0.
0: Disable SPI function moved to P5.
1: Set SPI function on P5 as following definition.
‘/SS’ function in P1.4 is moved to P5.0.
‘MOSI’ function in P1.5 is moved to P5.1.
‘MISO’ function in P1.6 is moved to P5.2.
‘SPICLK’ function in P1.7 is moved to P5.3.

Bit 4: P4S1, Serial Port 1 (UART1) on P4.0/P4.1.
0: Disable UART1 function moved to P4.
1: Set UART1 RXD1/TXD1 on P4.0/P4.1 following definition.
‘RXD1’ function in P1.2 is moved to P4.0.
‘TXD1’ function in P1.3 is moved to P4.1.

Bit 3~1: Reserved. Software must write “0”s on these bits when AUXR1 is written.

Bit 0: DPS, dual DPTR Selector.
0: Select DPTR0.
1: Select DPTR1.

AUXR2: Auxiliary Register 2

SFR Page = All
SFR Address = 0xA6 RESET = 00xx-xx00

7	6	5	4	3	2	1	0
T0X12	T1X12	--	--	--	--	T1CKOE	T0CKOE
R/W	**R/W**	**R**	**R**	**R**	**R**	**R/W**	**R/W**

Bit 7: T0X12, Timer 1 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source.

Bit 6: T1X12, Timer 1 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source.

Bit 5~2: Reserved.

Bit 1: T1CKOE, Timer 1 Clock Output Enable.
0: Disable Timer 1 clock output.
1: Enable Timer 1 clock output on P3.5.

Bit 0: T0CKOE, Timer 0 Clock Output Enable.
0: Disable Timer 0 clock output.
1: Enable Timer 0 clock output on P3.4.

SFRPI: SFR Page Index Register

SFR Page = All

SFR Address = 0xAC RESET = xxxx-0000

7	6	5	4	3	2	1	0
--	--	--	--	PIDX3	PIDX2	PIDX1	PIDX0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7~4: Reserved. Software must write “0”s on these bits when SFRPI is written.

Bit 3~0: SFR Page Index. The available pages are only page “0”, “1” and “F”.
There are four registers only in Page 0, T2CON(C8H), SCON0(98H), SBUF0(99H) and SCFG(9AH).
Three registers in Page 1, SCON0(98H), SBUF0(99H) and SCFG(9AH).
One register in Page F, P6(C8H).
Other registers are accessed by all pages.

PIDX[3:0]	Selected Page
0000	Page 0
0001	Page 1
0010	Page 2
0011	Page 3
……	……
……	……
……	……
1111	Page F

AUXRA: Auxiliary Register A

SFR Address = IFMT POR = 0010-0100

7	6	5	4	3	2	1	0
DBOD	BORE	OCDE	ILRCOE	XTALE	IHRCOE	OSCS1	OSCS0
R/W	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**	**R/W**

Bit 7: DBOD, Disable BOD.
0: Enable BOD in default state.
1: Disable BOD.
If software re-enables BOD in program flow, must wait more than **50us** for BOD circuit start-up time. In this period, software must disable BOD interrupt and BORE to filter the pseudo BOD event.

Bit 6: BORE, Brown-Out Reset Enable
0: Disable Reset action if BOD occurs.
1: Enable a Reset action if BOD occurs.

Bit 5: OCDE, OCD enable. The initial value is loaded from OR and reset by POR.
0: Disable OCD interface on P4.4 and P4.5
1: Enable OCD interface on P4.4 and P4.5.

Bit 4: LIRCOE, Internal Low frequency RC Oscillator Enable.
0: Disable internal low frequency RC oscillator.
1: Enable internal low frequency RC oscillator. It is about 125 KHz. It needs **50us** to have stable output after ILRCOE is enabled.

Bit 3: XTALE, external Crystal(XTAL) Enable. The default value is set by hardware option on clock source selection.
0: Disable XTAL oscillating circuit. In this case, XTAL2 and XTAL1 behave as Port 6.0 and Port 6.1.
1: Enable XTAL oscillating circuit. If this bit is set by CPU software, it needs **5ms** to have stable output after XTALE is enabled.

Bit 2: IHRCOE, Internal High frequency RC Oscillator Enable. The default value is set by hardware option on clock source selection.
0: Disable internal high frequency RC oscillator.

1: Enable internal low frequency RC oscillator. If this bit is set by CPU software, it needs **50us** to have stable output after IHRCOE is enabled.

Bit 1~0: OSC input selection.

OSCS[1:0]	OSCin Source	P6.0 Function	P6.1 Function
00	IHRCO (Default)	P6.0 or IHRCO output	P6.1
01	ILRCO	P6.0	P6.1
10	External Clock Input	Clock Input	P6.1
11	Ext. Crystal Oscillating	XTAL2	XTAL1

AUXRB: Auxiliary Register B

SFR Address = IFMT RESET = xxx0-00x0

7	6	5	4	3	2	1	0
--	--	--	**IAPO**	**LPM3**	**LPM2**	--	**LPM0**
R	**R**	**R**	**R/W**	**R/W**	**R/W**	**R**	**R/W**

Bit 7~5: Reserved. Software must write “0” on these bits when AUXRB is written.

Bit 4: IAPO, IAP function Only.
0: Maintain IAP region to service IAP function and code execution when the flash region lower than AP boundary defined by IAPLB.
1: Disable the code execution in IAP region and the region only service IAP function.

Bit 3, 2, 0: LPM3, 2, 0. Control bits for Low power mode operation.
If the frequency of OSCin and SYSCLK is slower than 6MHz, software can write “1”s on this bits to reduce operating current. Otherwise, software must write “0”s on this bit to maintain the high speed performance. Other values writing on these bits are not permitted.

Bit 1: Reserved. Software must write “0” on this bit when AUXRB is written.

25. Hardware Option

The MCU's Hardware Option defines the device behavior which cannot be programmed or controlled by software. The hardware options can only be programmed by a Universal Programmer, the "Megawin 8051 Writer" or the "Megawin 8051 ICP Programmer". After whole-chip erased, all the hardware options are left in "disabled" state and there is no ISP-memory and IAP-memory configured. The MG82FE/L564 has the following Hardware Options:

LOCK:
[enabled]: Code dumped on a universal Writer or Programmer is locked to 0xFF for security.
[disabled]: Not locked.

ISP-memory Space:
The ISP-memory space is specified by its starting address. And, its higher boundary is limited by the Flash end address, i.e., 0xFFFF. The following table list the ISP space option in this chip.

ISP-memory Size	ISP Start Address
4K bytes	0xF000
3.5K bytes	0xF200
3K bytes	0xF400
2.5K bytes	0xF600
2K bytes	0xF800
1.5K bytes	0xFA00
1K bytes	0xFC00
No ISP Space	--

HWBS:
[enabled]: When powered up, MCU will boot from ISP-memory if ISP-memory is configured.
[disabled]: MCU always boots from AP-memory.

HWBS2:
[enabled]: Not only power-up but also any reset will cause MCU to boot from ISP-memory if ISP-memory is configured.
[disabled]: Where MCU boots from is determined by HWBS.

BODRE:
[enabled]: BOD will trigger a RESET event to CPU on AP program start address.(4.2V for E-series and 2.4V for L-series)
[disabled]: BOD can not trigger a RESET to CPU.
This bit value is mirrored to AUXRA.6, BORE, which can be controlled by software program.

OSCDN:
[enabled]: The gain of crystal oscillator is reduced. It will helpful in EMI reduction. Regarding application not needing high frequency clock, it is recommended to do so.
[disabled]: The gain of crystal oscillator is enough for oscillation up to 25MHz.

ENRCO:
[enabled]: Enable internal high frequency RC oscillator. If this hardware option is enabled, the IHRCO will be the source into system clock and power down the crystal oscillating circuit. And XTAL2, XTAL1 will be switched to alternated function of P6.0 and P6.1.
[disabled] Disable IHRCO and set the external crystal oscillator as system clock source.

WDSFWP:
[enabled]: The software SFR - WDTCR will be write-protected except the bit – CLRW.
[disabled]: The software SFR – WDTCR is free for writing of software.

HWENW: Hardware loaded for "ENW" of WDTCR.

[enabled]: Clearing it will enable WDT and load the content of HWWIDL, HWPS2, HWPS1 and HWPS0 to WDTCR SFR when power-up.
[disabled]: WDT is not enabled automatically during power-up.

HWWIDL: HWPS2, HWPS1, HWPS0
When HWENW is enabled, the content on these four fused bits will be loaded to WDTCR SFR during power-up.

NSWDT: Non-Stopped WDT
[enabled]: Keep WDT running in power down mode and select internal high frequency RC Oscillator for the clock source of WDT.
[disabled]: Disable WDT running in power down mode.

26. Absolute Maximum Rating

For MG82FE564

Parameter	Rating	Unit
Ambient temperature under bias	-40 ~ +85	°C
Storage temperature	-65 ~ + 150	°C
Voltage on any Port I/O Pin or RESET with respect to Ground	-0.5 ~ VDD + 0.5	V
Voltage on VDD with respect to Ground	-0.5 ~ +6.0	V
Maximum total current through VDD 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.

For MG82FL564

Parameter	Rating	Unit
Ambient temperature under bias	-40 ~ +85	°C
Storage temperature	-65 ~ + 150	°C
Voltage on any Port I/O Pin or RESET with respect to Ground	-0.3 ~ VDD + 0.3	V
Voltage on VDD with respect to Ground	-0.3 ~ +4.2	V
Maximum total current through VDD 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. DC Characteristics

VSS = 0V, TA = 25 ℃, VDD = 5.0V and execute NOP for each machine cycle, unless otherwise specified

Symbol	Parameter	Test Condition	Limits			Unit
			min	typ	max	
V_{IH1}	Input High voltage (All I/O Ports)		2.0			V
V_{IH2}	Input High voltage (RESET)		3.5			V
V_{IL1}	Input Low voltage (All I/O Ports)				0.8	V
V_{IL2}	Input Low voltage (RESET)				1.6	V
I_{IH}	Input High Leakage current (All I/O Ports)	V_{PIN} = VDD		0	10	uA
I_{IL1}	Logic 0 input current (All quasi-I/O Ports)	V_{PIN} = 0.4V		20	50	uA
I_{IL2}	Logic 0 input current (All Input only or open-drain Ports)	V_{PIN} = 0.4V		0	10	uA
I_{H2L}	Logic 1 to 0 input transition current (All quasi-I/O Ports)	V_{PIN} =1.8V		250	500	uA
I_{OH1}	Output High current (All quasi-I/O Ports)	V_{PIN} =2.4V	150	220		uA
I_{OH2}	Output High current (All push-pull output ports)	V_{PIN} =2.4V	12			mA
I_{OL1}	Output Low current (All I/O Ports)	V_{PIN} =0.4V	12			mA
I_{OP}	Operating current	F_{OSC} = 24MHz		22	30	mA
I_{IDLE}	Idle mode current	F_{OSC} = 20MHz		12	20	mA
I_{PD}	Power down current			1	10	uA
R_{RST}	Internal reset pull-down resistance			100		Kohm

VSS = 0V, TA = 25 ℃, VDD = 3.3V and execute NOP for each machine cycle, unless otherwise specified

Symbol	Parameter	Test Condition	Limits			Unit
			min	typ	max	
V_{IH1}	Input High voltage (All I/O Ports)		2.0			V
V_{IH2}	Input High voltage (RESET)		2.8			V
V_{IL1}	Input Low voltage (All I/O Ports)				0.8	V
V_{IL2}	Input Low voltage (RESET)				1.5	V
I_{IH}	Input High Leakage current (All I/O Ports)	V_{PIN} = VDD		0	10	uA
I_{IL1}	Logic 0 input current (All quasi-I/O Ports)	V_{PIN} = 0.4V		7	30	uA
I_{IL2}	Logic 0 input current (All Input only or open-drain Ports)	V_{PIN} = 0.4V		0	10	uA
I_{H2L}	Logic 1 to 0 input transition current (All quasi-I/O Ports)	V_{PIN} =1.8V		100	250	uA
I_{OH1}	Output High current (All quasi-I/O Ports)	V_{PIN} =2.4V	40	70		uA
I_{OH2}	Output High current (All push-pull output ports)	V_{PIN} =2.4V	4			mA
I_{OL1}	Output Low current (All I/O Ports)	V_{PIN} =0.4V	8			mA
I_{OP}	Operating current	F_{OSC} = 20MHz		20	25	mA
I_{IDLE}	Idle mode current	F_{OSC} = 20MHz		9	15	mA
I_{PD}	Power down current			1	5	uA
R_{RST}	Internal reset pull-down resistance			200		Kohm

27.2. AC Characteristics

28. Instruction Set

MNEMONIC	DESCRIPTION	BYTE	EXECUTION Cycles
DATA TRASFER			
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	Move on-chip auxiliary RAM(8-bit address) to Acc	1	3
MOVX A,@DPTR	Move on-chip auxiliary RAM(16-bit address) to Acc	1	3
MOVX @Ri,A	Move Acc to on-chip auxiliary RAM(8-bit address)	1	3
MOVX @DPTR,A	Move Acc to on-chip auxiliary RAM(16-bit address)	1	3
MOVX A,@Ri	Move external RAM(8-bit address) to Acc	1	3 ~ 20*Note1
MOVX A,@DPTR	Move external RAM(16-bit address) to Acc	1	3 ~ 20*Note1
MOVX @Ri,A	Move Acc to external RAM(8-bit address)	1	3 ~ 20*Note1
MOVX @DPTR,A	Move Acc to external RAM(16-bit address)	1	3 ~ 20*Note1
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 Acc	1	4
ARITHEMATIC 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
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
INC DPTR	Increment DPTR	1	1
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	5
DA A	Decimal Adjust Acc	1	4
LOGIC OPERATION			
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
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
BOOLEAN VARIABLE MANIPULATION			
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
PROAGRAM BRACHING			
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
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 equal	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

Note 1: The cycle time for access of external auxiliary RAM is:

EMAI[1:0] = 00: 5 + 2 x ALE_Stretch + RW_Stretch + 2 x RWSH; (5~20)

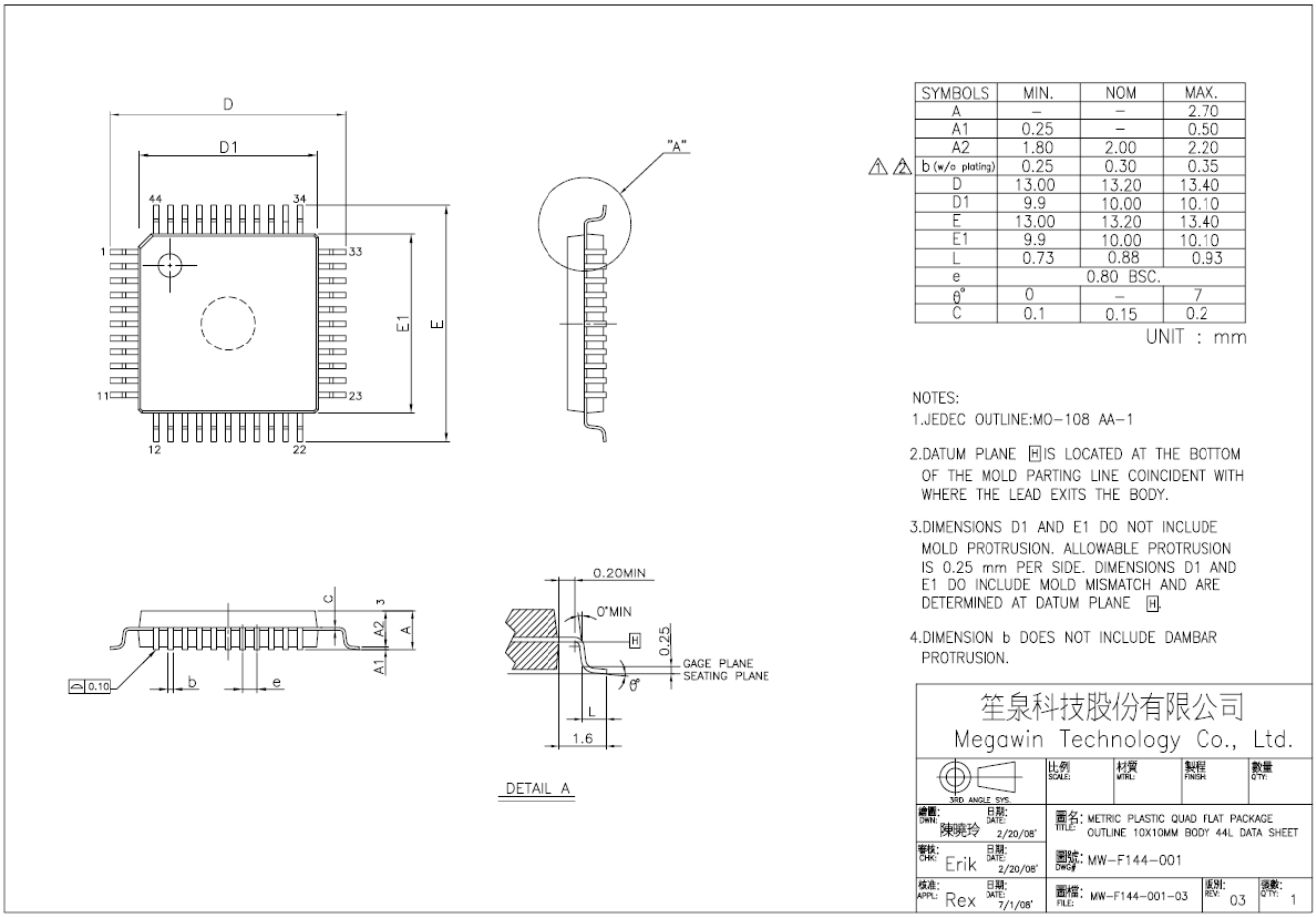
EMAI[1:0] = 01: 3 + RW_Stretch + 2 x RWSH; (3~12)

EMAI[1:0] = 10: 3 + RW_Stretch + 2 x RWSH; (3~12)

EMAI[1:0] = 11: Not Define.

29. Package Dimension

PQFP-44



SYMBOLS	MIN.	NOM	MAX.
A	–	–	2.70
A1	0.25	–	0.50
A2	1.80	2.00	2.20
b (w/o plating)	0.25	0.30	0.35
D	13.00	13.20	13.40
D1	9.9	10.00	10.10
E	13.00	13.20	13.40
E1	9.9	10.00	10.10
L	0.73	0.88	0.93
e		0.80 BSC.	
θ°	0	–	7
C	0.1	0.15	0.2

UNIT : mm

NOTES:

1.JEDEC OUTLINE:MO-108 AA-1

2.DATUM PLANE H IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.

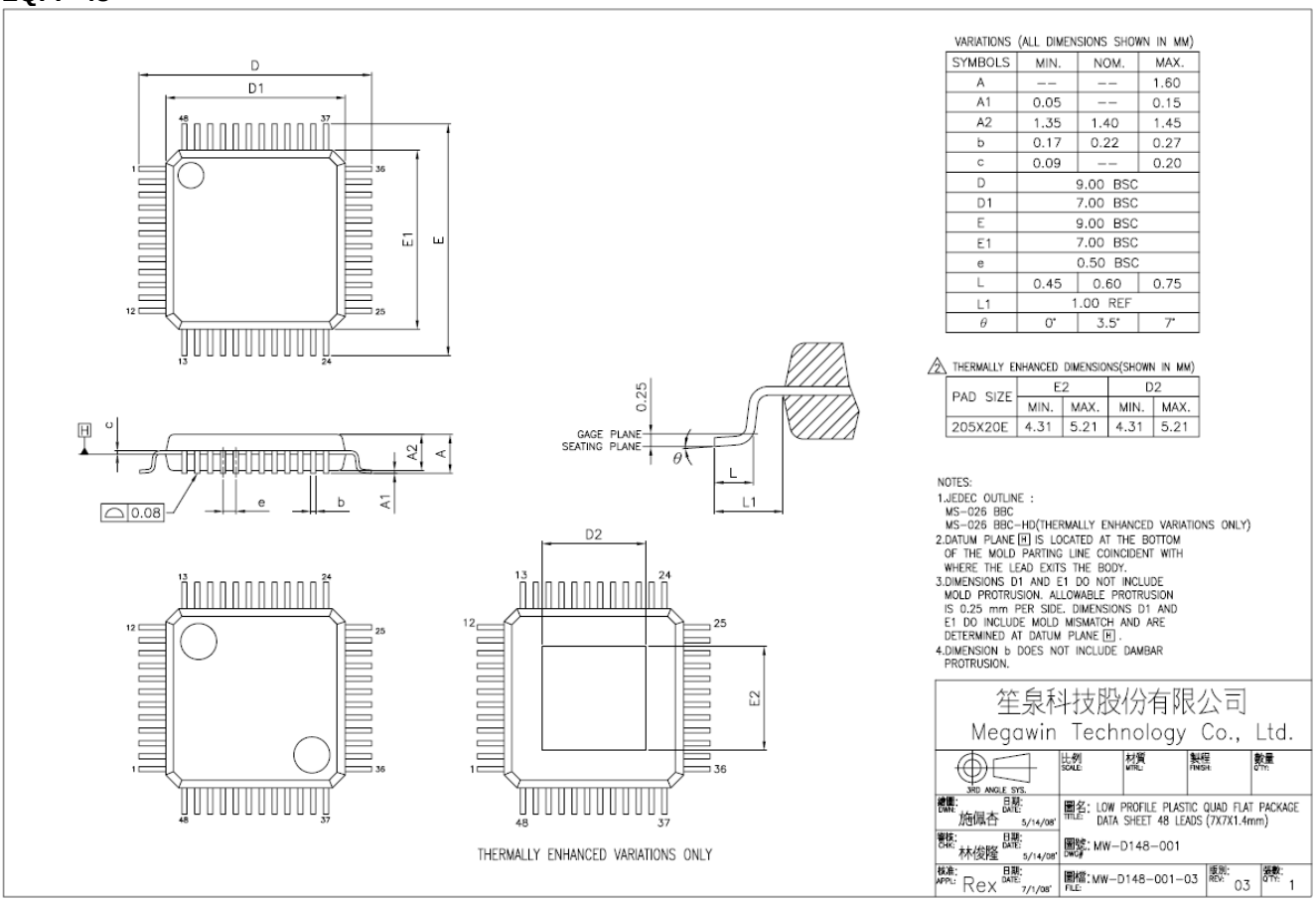
3.DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.

4.DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

笙泉科技股份有限公司
Megawin Technology Co., Ltd.

3RD ANGLE SYS.	比例 SCALE:	材質 MTRL:	製程 FINISH:	數量 QTY:
繪圖: DWN: 陳曉玲 日期: DATE: 2/20/08'	圖名: TITLE: METRIC PLASTIC QUAD FLAT PACKAGE OUTLINE 10X10MM BODY 44L DATA SHEET			
審核: CHK: Erik 日期: DATE: 2/20/08'	圖號: DWG#: MW-F144-001			
核准: APPL: Rex 日期: DATE: 7/1/08'	檔名: FILE: MW-F144-001-03	版別: REV: 03	張數: QTY: 1	

LQFP-48



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.17	0.22	0.27
c	0.09	--	0.20
D	9.00 BSC		
D1	7.00 BSC		
E	9.00 BSC		
E1	7.00 BSC		
e	0.50 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
θ	0°	3.5°	7°

2 THERMALLY ENHANCED DIMENSIONS(SHOWN IN MM)

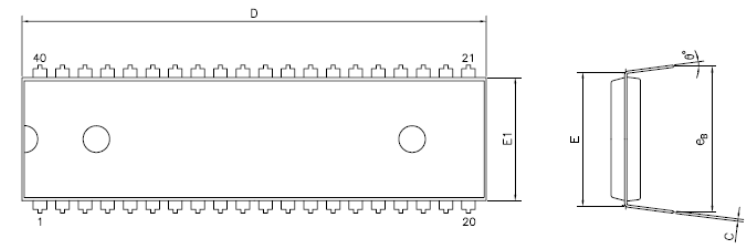
PAD SIZE	E2		D2	
	MIN.	MAX.	MIN.	MAX.
205X20E	4.31	5.21	4.31	5.21

NOTES:
1.JEDEC OUTLINE :
MS-026 BBC
MS-026 BBC-HD(THERMALLY ENHANCED VARIATIONS ONLY)
2.DATUM PLANE H IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.
3.DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.
4.DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

笙泉科技股份有限公司
Megawin Technology Co., Ltd.

3RD ANGLE SYS.	比例 SCALE:	材質 MTRL:	製程 FINISH:	數量 QTY:
繪圖: DWN: 施佩杏 日期: DATE: 5/14/08'	圖名: TITLE: LOW PROFILE PLASTIC QUAD FLAT PACKAGE DATA SHEET 48 LEADS (7X7X1.4mm)			
審核: CHK: 林俊隆 日期: DATE: 5/14/08'	圖號: DWG# MW-D148-001			
核准: APPL: Rex 日期: DATE: 7/1/08'	圖檔: FILE: MW-D148-001-03	版別: REV: 03	張數: QTY: 1	

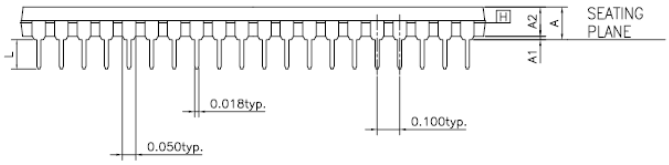
PDIP-40



SYMBOLS	MIN.	NOR.	MAX.
A	–	–	0.190
A1	0.015	–	–
A2	0.150	0.155	0.160
C	0.008	–	0.015
D	2.055	2.060	2.070
E		0.600 BSC	
E1	0.540	0.545	0.550
L	0.120	0.130	0.140
e_B	0.630	0.650	0.670
θ	0	7	15

UNIT : INCH

NOTE:
1.JEDEC OUTLINE : MS-011 AC



笙泉科技股份有限公司
Megawin Technology Co., Ltd.

3RD ANGLE SYS.	比例 SCALE:	材質 MTRL:	製程 FINISH:	數量 Q'TY:
繪圖: DWN: 施佩杏 日期: DATE: 9/30/05'	圖名: TITLE: DUAL INLINE PLASTIC DATA SHEET P-DIP 40 LEADS (600mil)			
審核: CHK: 蔡大猷 日期: DATE: 10/1/05'	圖號: DWG#: MW-E140-001			
核准: APPL: Rex 日期: DATE: 7/1/07'	圖檔: FILE: MW-E140-001-03	版別: DATE: 03	張數: DATE: 1	

30. Revision History

Rev	Descriptions	Date
V1.0	1. Initial release 2. Added application note (sample code) for switching default internal RC-OSC to External XTAL 3. ADC conversion speed note. Suggest user doing ADC conversion under SYSCLK don't over 20MHz in High temperature (Over 60℃) environment.	2010/08/19
V1.1	1. Remove PDIP-40 package	2010/11/03
V1.2	2. Remove PLCC-44 package	2011/05/16
V1.3	1. Add PDIP-40PLCC-44 package 2. PCON2 edit error (**CKCON2**) 3. ISPCR edit error (**SFR address**)	2011/06/20
V1.4	1. COMD.6 edit error (**FEOV bit**) 2. PCAPWMn edit error (**R, R/W**)	2011/06/27
V1.5	Remove PLCC-44 package	2011/12/06
V1.51	Edit content	2012/07/03
A1.0	New form & added sample code	2014/02/25
A1.01	Modify system clock Diagram	2015/09/25

Disclaimers

Herein, Megawin stands for "***Megawin Technology Co., Ltd.***"

Life Support — This product is not designed for use in medical, life-saving or life-sustaining applications, or systems where malfunction of this product can reasonably be expected to result in personal injury. Customers using or selling this product for use in such applications do so at their own risk and agree to fully indemnify Megawin for any damages resulting from such improper use or sale.

Right to Make Changes — Megawin reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in mass production, relevant changes will be communicated via an Engineering Change Notification (ECN).