Contents

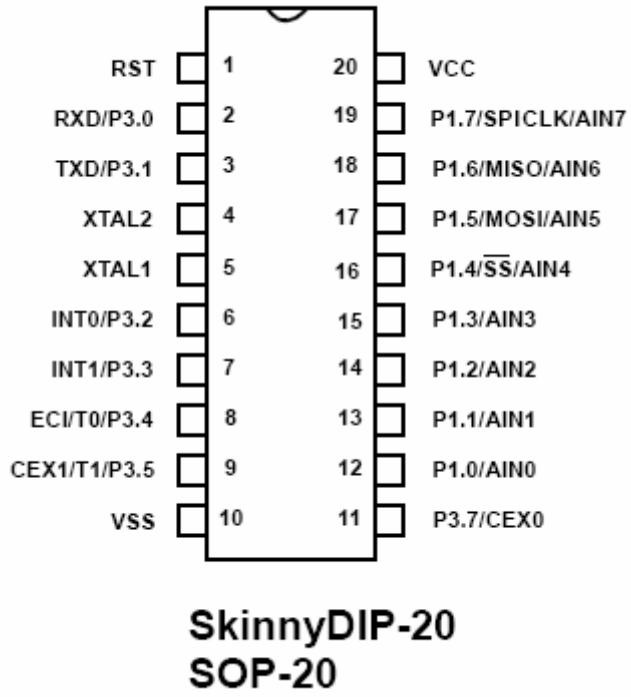
Preface

The User Manual describes only the special & new features which the MPC82L (E) 52 and MPC82L (E) 54 have, while the standard 8051 MCU doesn't have. It provides a great help for users to use these two chips.

1 General

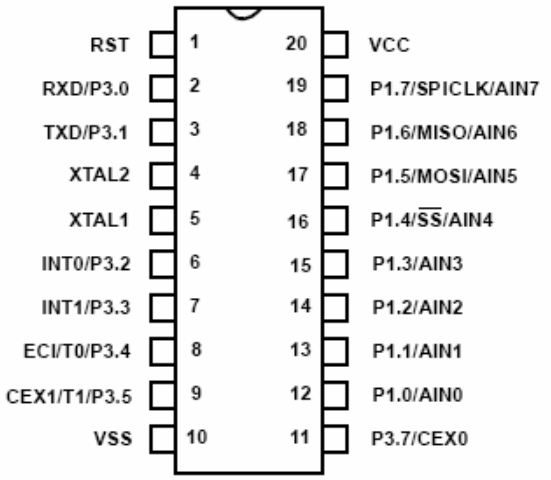
1.1 Pin Assignment

MPC82L (E) 52

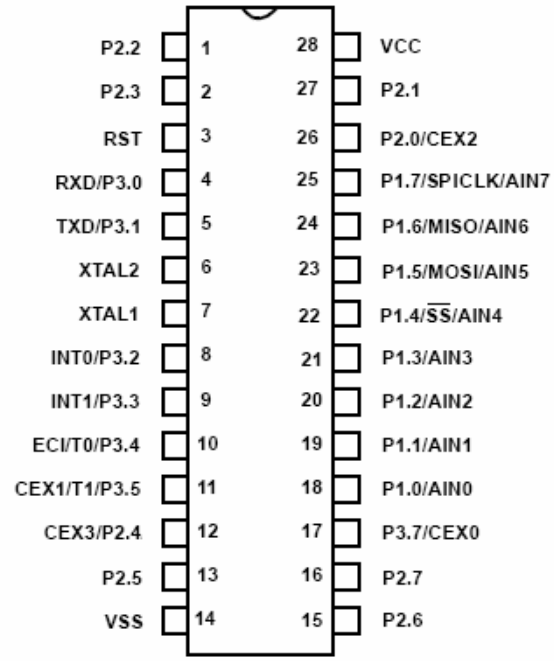


SkinnyDIP-20
SOP-20

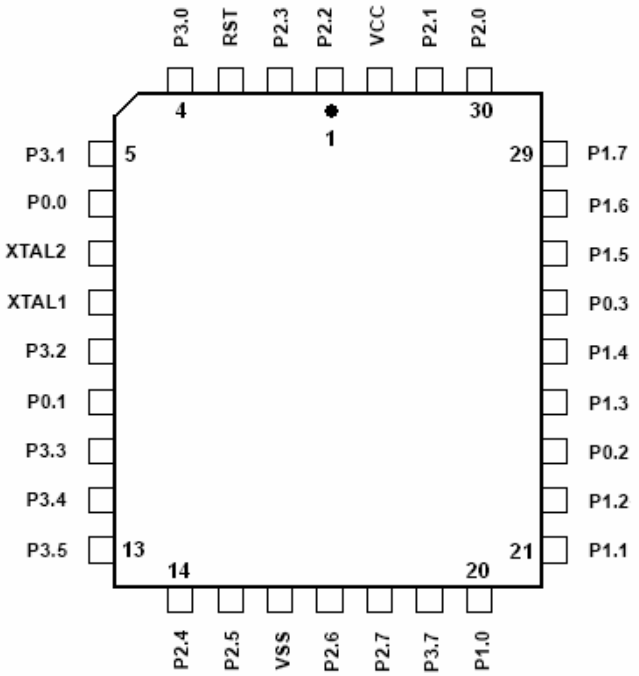
MPC82L (E) 54



SkinnyDIP-20
SOP-20



SkinnyDIP-28
SOP-28
SSOP-28



PLCC-32

1.2 Special Function Registers (SFRs)

MPC82L (E) 52 SFRs

SYMBOL	DESCRIPTION	ADDR	BIT ADDRESS & SYMBOL MSB							LSB	RESET VALUE
ACC*	Accumulator	**E0H**									00H
B*	B Register	**F0H**									00H
PSW*	Program Status Word	**D0H**	**D7H** CY	**D6H** AC	**D5H** F0	**D4H** RS1	**D3H** RS0	**D2H** OV	**D1H** -	**D0H** P	000000x0B
SP	Stack Pointer	**81H**									07H
DPH	Data Pointer High	**83H**									00H
DPL	Data Pointer Low	**82H**									00H
P1*	Port 1	**90H**	**97H** P1.7 AIN7 SPICLK	**96H** P1.6 AIN6 MISO	**95H** P1.5 AIN5 MOSI	**94H** P1.4 AIN4 /SS	**93H** P1.3 AIN3	**92H** P1.2 AIN2	**91H** P1.1 AIN1	**90H** P1.0 AIN0	FFH
P3*	Port 3	**B0H**	**B7H** P3.7 CEX0	**B6H** -	**B5H** P3.5 T1 CEX1	**B4H** P3.4 T0 ECI	**B3H** P3.3 /INT1	**B2H** P3.2 /INT0	**B1H** P3.1 TXD	**B0H** P3.0 RXD	1x111111B
IP*	Interrupt Priority	**B8H**	**BFH** -	**BEH** PPCA_LVD	**BDH** PSPI_ADC	**BCH** PS	**BBH** PT1	**BAH** PX1	**B9H** PT0	**B8H** PX0	x0000000B
IE*	Interrupt Enable	**A8H**	**AFH** EA	**AEH** EPCA_LVD	**ADH** ESPI_ADC	**ACH** ES	**ABH** ET1	**AAH** EX1	**A9H** ET0	**A8H** EX0	00H
TMOD	Timer Mode	**89H**	GATE	C/-T	M1	M0	GATE	C/-T	M1	M0	00H
TCON*	Timer Control	**88H**	**8FH** TF1	**8EH** TR1	**8DH** TF0	**8CH** TR0	**8BH** IE1	**8AH** IT1	**89H** IE0	**88H** IT0	00H
TH0	Timer 0 High	**8CH**									00H
TL0	Timer 0 Low	**8AH**									00H
TH1	Timer 1 High	**8DH**									00H
TL1	Timer 1 Low	**8BH**									00H
SCON*	Serial Port Control	**98H**	**9FH** SM0/FE	**9EH** SM1	**9DH** SM2	**9CH** REN	**9BH** TB8	**9AH** RB8	**99H** TI	**98H** RI	00H
SBUF	Serial Data Buffer	**99H**									xxH
PCON	Power Control	**87H**	SMOD	SMOD0	LVF	POF	GF1	GF0	PD	IDL	00xx0000B

The following SFRs are new added.

SYMBOL	DESCRIPTION	ADDR	BIT ADDRESS & SYMBOL MSB							LSB	RESET VALUE
AUXR	Auxiliary Register	**8EH**	T0X12	T1X12	URM0X6	EADCI	ESPI	ENLVFI	-	-	000000xxB
PCON2	Power Control 2	**C7H**	-	-	-	-	-	CKS2	CKS1	CKS0	xxxxx000B
IPH	Interrupt Priority High	**B7H**	-	PPCAH_ LVD	PSPIH_ ADC	PSH	PT1H	PX1H	PT0H	PX0H	x0000000B
SADEN	Slave Address Mask	**B9H**									00H
SADDR	Slave Address	**A9H**									00H
ADCTL	ADC Control Register	**C5H**	ADCON	SPEED1	SPEED0	ADCI	ADCS	CHS2	CHS1	CHS0	00H
ADC	ADC Result	**C6H**	B7	B6	B5	B4	B3	B2	B1	B0	xxH
WDTCR	Watch-dog Timer	**E1H**	WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0	0x000000B
P1M0	Port1 Mode Register 0	**91H**	P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0	00H
P1M1	Port1 Mode Register 1	**92H**	P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0	00H
P3M0	Port3 Mode Register 0	**B1H**	P3M0.7	-	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0	00H
P3M1	Port3 Mode Register 1	**B2H**	P3M1.7	-	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	00H
CCON*	PCA Counter Control Register	**D8H**	**DFH** CF	**DEH** CR	**DDH** -	**DCH** -	**DBH** -	**DAH** -	**D9H** CCF1	**D8H** CCF0	00xxxx00B
CMOD	PCA Counter Mode Register	**D9H**	CIDL	-	-	-	-	CPS2	CPS1	ECF	0xxxx000B
CH	PCA Counter, HB	**F9H**									00H
CL	PCA Counter, LB	**E9H**									00H
CCAPM0	PCA Module0 Com/Cap Register	**DAH**	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x0000000B
CCAPM1	PCA Module1 Com/Cap Register	**DBH**	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x0000000B
CCAP0H	PCA Module0 Capture Register, HB	**FAH**									00H
CCAP0L	PCA Module0 Capture Register, LB	**EAH**									00H
CCAP1H	PCA Module1 Capture Register, HB	**FBH**									00H
CCAP1L	PCA Module1 Capture Register, LB	**EBH**									00H
PCAPWM0	PWM Mode, auxiliary 0	**F2H**	-	-	-	-	-	-	ECAP0H	ECAP0L	xxxxxx00B
PCAPWM1	PWM Mode, auxiliary 1	**F3H**	-	-	-	-	-	-	ECAP1H	ECAP1L	xxxxxx00B
SPSTAT	SPI Status Register	**84H**	SPIF	WCOL	-	-	-	-	-	-	00xxxxxxB
SPCTL	SPI Control Register	**85H**	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	04H
SPDAT	SPI Data Register	**86H**									00H
ISPCR	ISP Control Register	**E7H**	ISPEN	SWBS	SWRST	CFAIL	-	ICK2	ICK1	ICK0	0000x000B
IFMT	ISP Flash Mode Table	**E5H**	-	-	-	-	-	-	MS1	MS0	xxxxx000B
IFADRH	ISP Flash Address High	**E3H**									00H
IFADRL	ISP Flash Address Low	**E4H**									00H
IFD	ISP Flash Data	**E2H**									00H
SCMD	ISP Sequential Command	**E6H**									xxH

Notes:

*: bit addressable

-: reserved bit

MPC82L(E)54 SFRs

SYMBOL	DESCRIPTION	ADDR	BIT ADDRESS & SYMBOL MSB							LSB	RESET VALUE
ACC*	Accumulator	**E0H**									00H
B*	B Register	**F0H**									00H
PSW*	Program Status Word	**D0H**	**D7H** CY	**D6H** AC	**D5H** F0	**D4H** RS1	**D3H** RS0	**D2H** OV	**D1H** -	**D0H** P	000000x0B
SP	Stack Pointer	**81H**									07H
DPH	Data Pointer High	**83H**									00H
DPL	Data Pointer Low	**82H**									00H
P0*	Port 0	**80H**	**87H** P0.7	**86H** P0.6	**85H** P0.5	**84H** P0.4	**83H** P0.3	**82H** P0.2	**81H** P0.1	**80H** P0.0	FFH
P1*	Port 1	**90H**	**97H** P1.7 AIN7 SPICLK	**96H** P1.6 AIN6 MISO	**95H** P1.5 AIN5 MOSI	**94H** P1.4 AIN4 /SS	**93H** P1.3 AIN3	**92H** P1.2 AIN2	**91H** P1.1 AIN1	**90H** P1.0 AIN0	FFH
P2*	Port 2	**A0H**	**A7H** P2.7	**A6H** P2.6	**A5H** P2.5	**A4H** P2.4 CEX3	**A3H** P2.3	**A2H** P2.2	**A1H** P2.1	**A0H** P2.0 CEX2	FFH
P3*	Port 3	**B0H**	**B7H** P3.7 CEX0	**B6H** -	**B5H** P3.5 T1 CEX1	**B4H** P3.4 T0 ECI	**B3H** P3.3 /INT1	**B2H** P3.2 /INT0	**B1H** P3.1 TXD	**B0H** P3.0 RXD	1x111111B
IP*	Interrupt Priority	**B8H**	**BFH** -	**BEH** PPCA_LVD	**BDH** PSPI_ADC	**BCH** PS	**BBH** PT1	**BAH** PX1	**B9H** PT0	**B8H** PX0	x0000000B
IE*	Interrupt Enable	**A8H**	**AFH** EA	**AEH** EPCA_LVD	**ADH** ESPI_ADC	**ACH** ES	**ABH** ET1	**AAH** EX1	**A9H** ET0	**A8H** EX0	00H
TMOD	Timer Mode	**89H**	GATE	C/-T	M1	M0	GATE	C/-T	M1	M0	00H
TCON*	Timer Control	**88H**	**8FH** TF1	**8EH** TR1	**8DH** TF0	**8CH** TR0	**8BH** IE1	**8AH** IT1	**89H** IE0	**88H** IT0	00H
TH0	Timer 1 High	**8CH**									00H
TL0	Timer 0 Low	**8AH**									00H
TH1	Timer 1 High	**8DH**									00H
TL1	Timer 0 Low	**8BH**									00H
SCON*	Serial Port Control	**98H**	**9FH** SM0/FE	**9EH** SM1	**9DH** SM2	**9CH** REN	**9BH** TB8	**9AH** RB8	**99H** TI	**98H** RI	00H
SBUF	Serial Data Buffer	**99H**									xxH
PCON	Power Control	**87H**	SMOD	SMOD0	LVF	POF	GF1	GF0	PD	IDL	00xx0000B

The following SFRs are new added.

SYMBOL	DESCRIPTION	ADDR	BIT ADDRESS & SYMBOL MSB							LSB	RESET VALUE
AUXR	Auxiliary Register	**8EH**	T0X12	T1X12	URM0X6	EADCI	ESPI	ENLVFI	-	-	000000xxB
PCON2	Power Control 2	**C7H**	-	-	-	-	-	CKS2	CKS1	CKS0	xxxxx000B
IPH	Interrupt Priority High	**B7H**	-	PPCAH_ LVD	PSPIH_ ADC	PSH	PT1H	PX1H	PT0H	PX0H	x0000000B
SADEN	Slave Address Mask	**B9H**									00H
SADDR	Slave Address	**A9H**									00H
ADCTL	ADC Control Register	**C5H**	ADCON	SPEED1	SPEED0	ADCI	ADCS	CHS2	CHS1	CHS0	00H
ADCH	ADC Result, Higher 8 bits	**C6H**	B9	B8	B7	B6	B5	B4	B3	B2	xxH
ADCL	ADC Result, Lower 2 bits	**BEH**	-	-	-	-	-	-	B1	B0	xxH
WDTCR	Watch-dog Timer	**E1H**	WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0	0x000000B
P0M0	Port0 Mode Register 0	**93H**	-	-	-	-	P0M0.3	P0M0.2	P0M0.1	P0M0.0	00H
P0M1	Port0 Mode Register 1	**94H**	-	-	-	-	P0M1.3	P0M1.2	P0M1.1	P0M1.0	00H
P1M0	Port1 Mode Register 0	**91H**	P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0	00H
P1M1	Port1 Mode Register 1	**92H**	P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0	00H
P2M0	Port2 Mode Register 0	**95H**	P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0	00H
P2M1	Port2 Mode Register 1	**96H**	P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0	00H
P3M0	Port3 Mode Register 0	**B1H**	P3M0.7	-	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0	00H
P3M1	Port3 Mode Register 1	**B2H**	P3M1.7	-	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	00H
CCON*	PCA Counter Control Register	**D8H**	**DFH** CF	**DEH** CR	**DDH** -	**DCH** -	**DBH** CCF3	**DAH** CCF2	**D9H** CCF1	**D8H** CCF0	00xxxx00B
CMOD	PCA Counter Mode Register	**D9H**	CIDL	-	-	-	-	CPS2	CPS1	ECF	0xxxx000B
CH	PCA Counter, HB	**F9H**									00H
CL	PCA Counter, LB	**E9H**									00H
CCAPM0	PCA Module0 Com/Cap Register	**DAH**	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x0000000B
CCAPM1	PCA Module1 Com/Cap Register	**DBH**	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x0000000B
CCAPM2	PCA Module2 Com/Cap Register	**DCH**	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x0000000B
CCAPM3	PCA Module3 Com/Cap Register	**DDH**	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x0000000B
CCAP0H	PCA Module0 Capture Register, HB	**FAH**									00H
CCAP0L	PCA Module0 Capture Register, LB	**EAH**									00H
CCAP1H	PCA Module1 Capture Register, HB	**FBH**									00H
CCAP1L	PCA Module1 Capture Register, LB	**EBH**									00H
CCAP2H	PCA Module2 Capture Register, HB	**FCH**									00H
CCAP2L	PCA Module2 Capture Register, LB	**ECH**									00H
CCAP3H	PCA Module3 Capture Register, HB	**FDH**									00H
CCAP3L	PCA Module3 Capture Register, LB	**EDH**									00H

(Continued)

PCAPWM0	PWM Mode, auxiliary 0	**F2H**	-	-	-	-	-	-	ECAP0H	ECAP0L	xxxxxx00B
PCAPWM1	PWM Mode, auxiliary 1	**F3H**	-	-	-	-	-	-	ECAP1H	ECAP1L	xxxxxx00B
PCAPWM2	PWM Mode, auxiliary 2	**F4H**	-	-	-	-	-	-	ECAP2H	ECAP2L	xxxxxx00B
PCAPWM3	PWM Mode, auxiliary 3	**F5H**	-	-	-	-	-	-	ECAP3H	ECAP3L	xxxxxx00B
SPSTAT	SPI Status Register	**84H**	SPIF	WCOL	-	-	-	-	-	-	00xxxxxxB
SPCTL	SPI Control Register	**85H**	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	04H
SPDAT	SPI Data Register	**86H**									00H
ISPCR	ISP Control Register	**E7H**	ISPEN	SWBS	SWRST	CFAIL	-	ICK2	ICK1	ICK0	0000x000B
IFMT	ISP Flash Mode Table	**E5H**	-	-	-	-	-	-	MS1	MS0	xxxxx000B
IFADRH	ISP Flash Address High	**E3H**									00H
IFADRL	ISP Flash Address Low	**E4H**									00H
IFD	ISP Flash Data	**E2H**									00H
SCMD	ISP Sequential Command	**E6H**									xxH

Notes:

*: bit addressable

-: reserved bit

1.3 Flash Memory Configuration

MPC82L(E)52

(With total Flash Memory = 8K bytes)

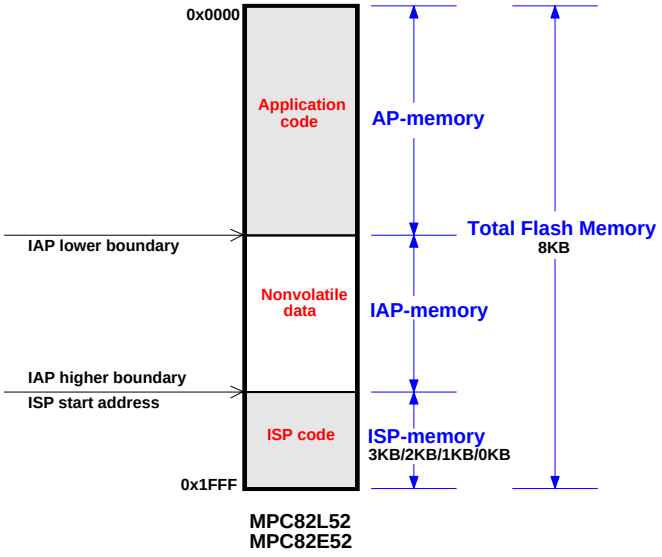
Note:

(1) ISP start address has four options:

0x1400 if ISP=3KB, or
0x1800 if ISP=2KB, or
0x1C00 if ISP=1KB, or
0x2000 if no ISP.

(2) IAP lower boundary = [IAPLB]x256.

(3) IAP higher boundary = ISP start address - 1.



MPC82L(E)54

(With total Flash Memory = 15.5K bytes)

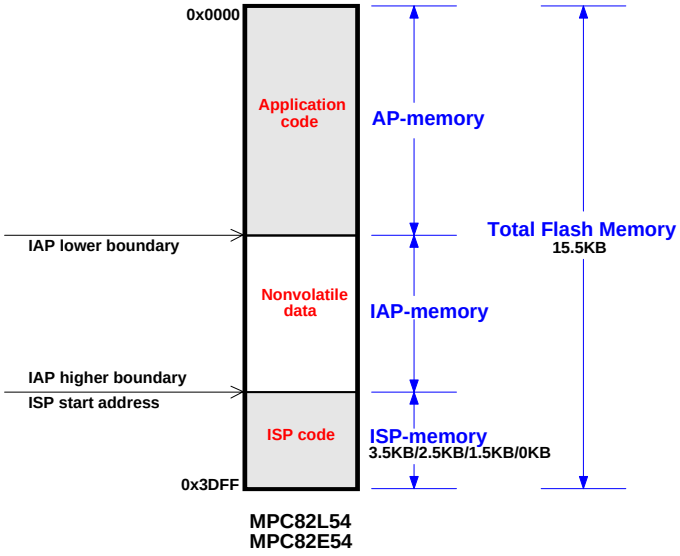
Note:

(1) ISP start address has four options:

0x3000 if ISP=3.5KB, or
0x3400 if ISP=2.5KB, or
0x3800 if ISP=1.5KB, or
0x3E00 if no ISP.

(2) IAP lower boundary = [IAPLB]x256.

(3) IAP higher boundary = ISP start address - 1.



2 I/O Ports

The MPC82L(E)52 has two I/O ports: P1 and P3; while the MPC82L(E)54 has four I/O ports: P0, P1, P2 and P3. The exact number of I/O pins available depends on their package type (see the following Table).

Table: I/O Pins Available v.s. Package Type

	SkinnyDIP-20 **SOP-20** (15 I/O pins)	**SkinnyDIP-28** **SOP-28** **SSOP-28** (23 I/O pins)	**PLCC-32** (27 I/O pins)
MPC82L(E)52	P1.0~P1.7 (8) P3.0~P3.5, P3.7 (7)	-	-
MPC82L(E)54	P1.0~P1.7 (8) P3.0~P3.5, P3.7 (7)	P1.0~P1.7 (8) P2.0~P2.7 (8) P3.0~P3.5, P3.7 (7)	P0.0~P0.3 (4) P1.0~P1.7 (8) P2.0~P2.7 (8) P3.0~P3.5, P3.7 (7)

2.1 Port Configurations

All these port pins can be individually and independently configured to one of four modes: *quasi-bidirectional (standard 8051 I/O port)*, *push-pull output*, *open-drain output* or *input-only (high-impedance)*. *All port pins are in the quasi-bidirectional mode after power-up or reset*. Each port pin has a Schmitt-triggered input to improve input noise rejection. Each port has two configuration registers to configure the I/O type for each port pin (see the following Table).

Table : Port Configuration Settings

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

Where x=0, 1, 2, or 3, and y=0~7. Registers PxM0 and PxM1 are described as follows.

P0M0 (Port0 Mode Register 0)

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

P0M1 (Port0 Mode Register 1)

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

P1M0 (Port1 Mode Register 0)

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

P1M1 (Port1 Mode Register 1)

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

P2M0 (Port2 Mode Register 0)

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

P2M1 (Port2 Mode Register 1)

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

P3M0 (Port3 Mode Register 0)

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

P3M1 (Port3 Mode Register 1)

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

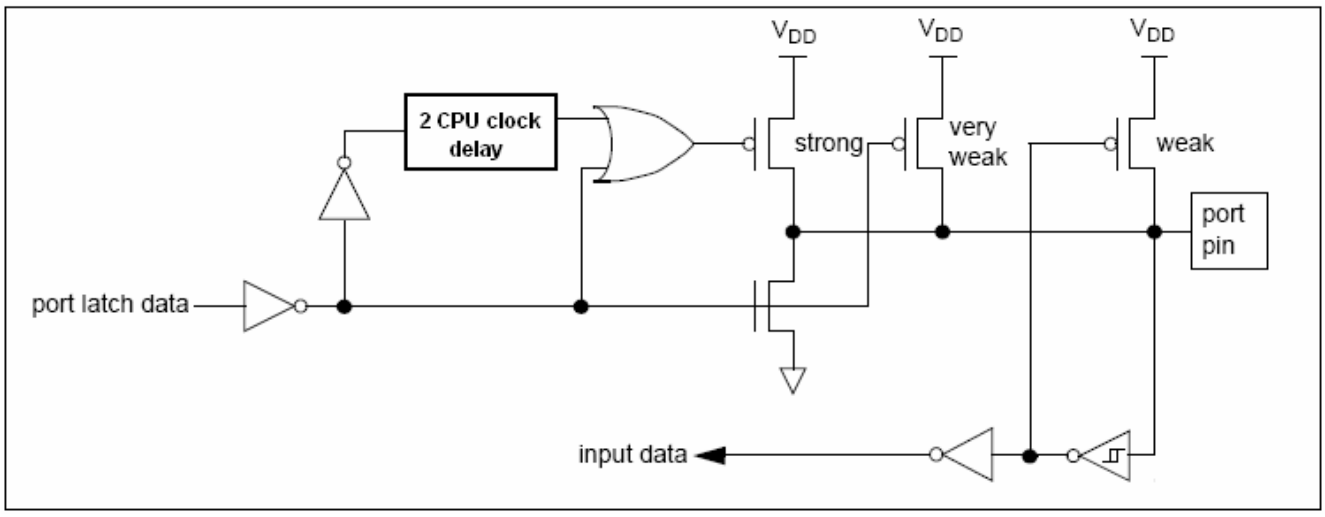
2.1.1 Quasi-bidirectional

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

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

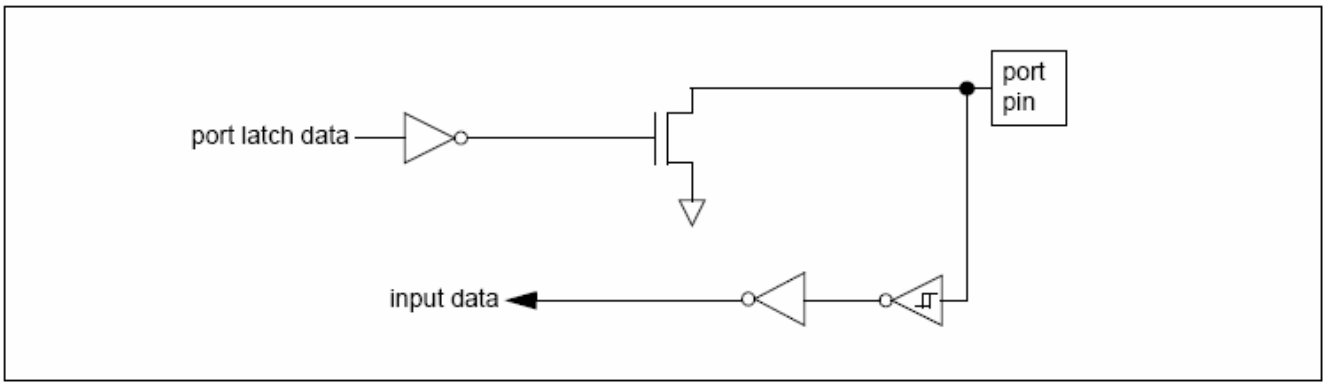
A second pull-up, called the “weak” pull-up, is turned on when the port register for the pin contains a logic “1” and the pin itself is also at a logic “1” level. This pull-up provides the primary source current for a quasi-bidirectional pin that is outputting a 1. If this pin is pulled low by the external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

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



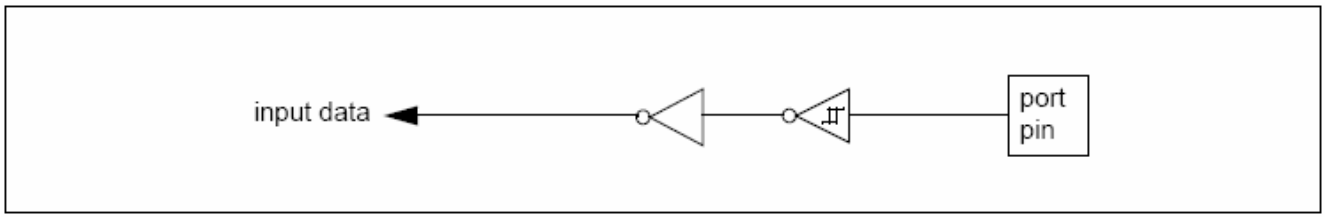
2.1.2 Open-Drain Output

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



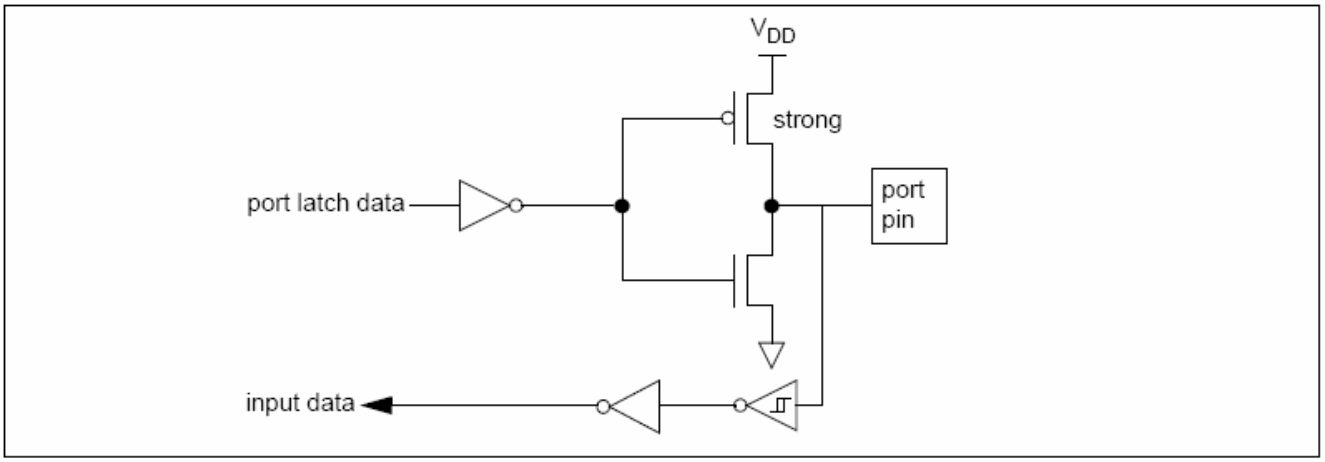
2.1.3 Input-Only (Hi-Z)

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



2.1.4 Push-Pull Output

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



2.2 Maximum Ratings for Port Outputs

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

3 On-chip expanded RAM (XRAM)

In addition to the standard 256 bytes of internal RAM, the MPC82L(E)54 has on-chip 256 bytes of expanded RAM (XRAM). User can use "***MOVX @Ri***" or "***MOVX @DPTR***" to access them. Since these "MOVX" instructions are modified for XRAM accessing, the I/O status of P0, P2 and P3.7 (/RD) are not affected while these instructions are executed.

Using the XRAM in Software

For KEIL-C51 compiler, to assign the variables to be located at XRAM, the "**pdata**" or "**xdata**" definition should be used. After being compiled, the variables declared by "**pdata**" and "**xdata**" will become the memories accessed by "MOVX @Ri" and "MOVX @DPTR", respectively. Thus the MPC82L(E)54 hardware can access them correctly. The user can get the following descriptions from the "*Keil Software — Cx51 Compiler User's Guide*".

Explicitly Declared Memory Types

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

The following table summarizes the available memory type specifiers.

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

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

Example:

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

4 Timer 0 and Timer 1

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

AUXR (Auxiliary Register)

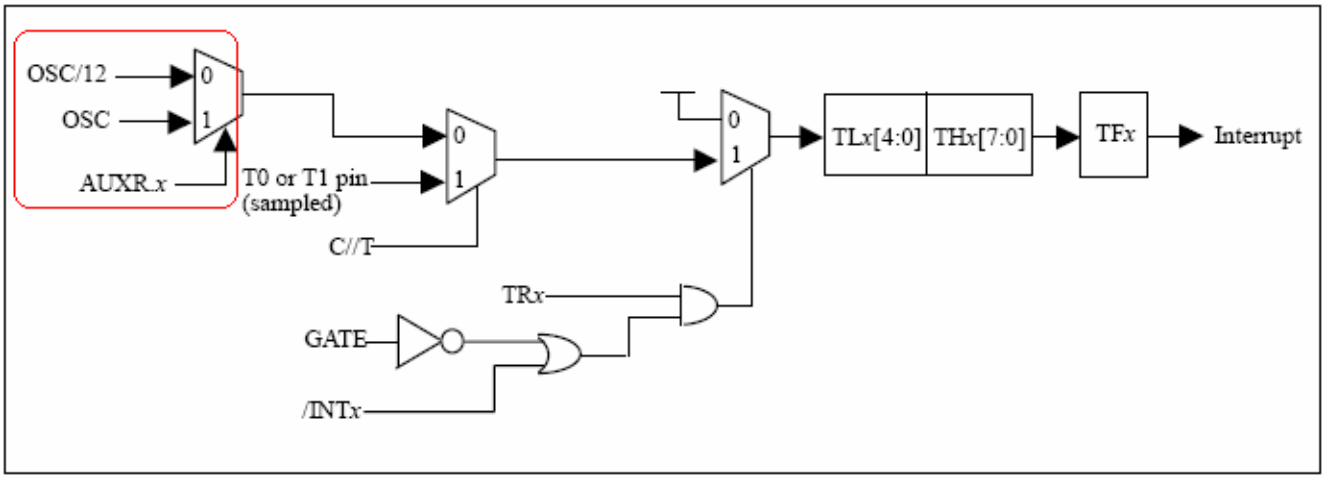
7	6	5	4	3	2	1	0
T0X12	**T1X12**	**URM0X6**	**EADCI**	**ESPI**	**ENLVFI**	**-**	**-**

T0X12: Timer 0 clock source select. Set to select Fosc as the clock source, and clear to select Fosc/12.
T1X12: Timer 1 clock source select. Set to select Fosc as the clock source, and clear to select Fosc/12.

Note:
Both these two bits are 0 after power-up or reset, and thus select Fosc/12 as the clock source by default, like the standard 8051 does.

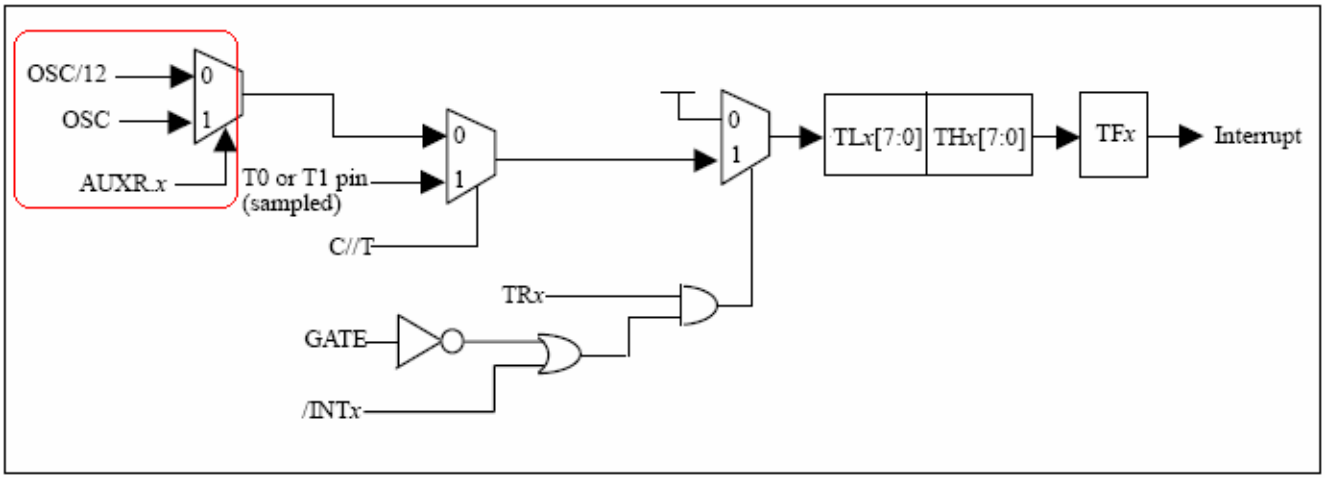
The following figures show the selection of alternate clock source for Timer 0 and Timer1.

4.1 Mode 0



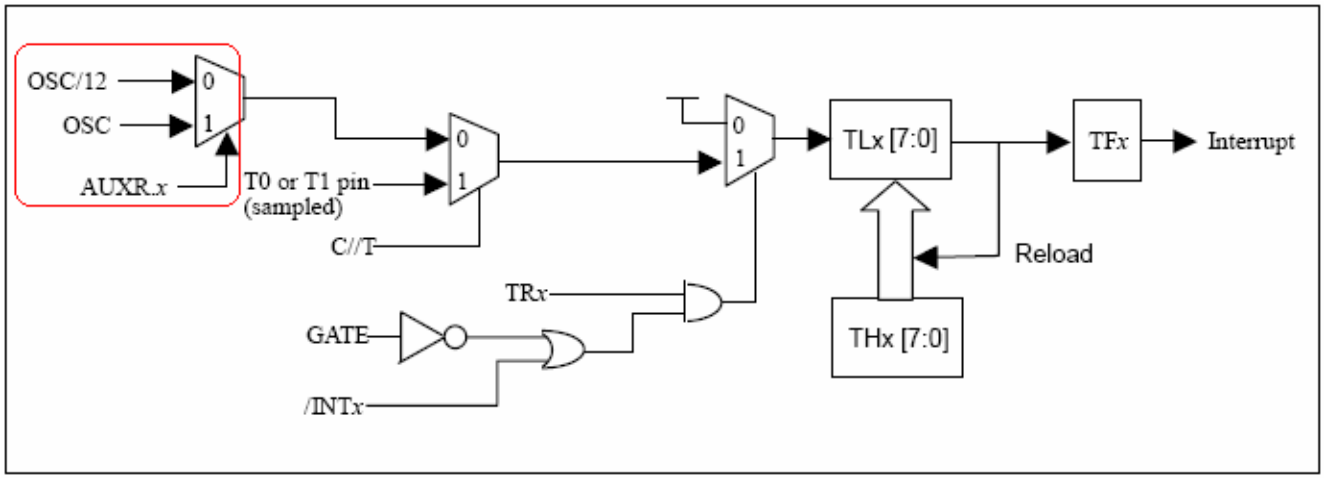
Where OSC means Fosc, the system clock.

4.2 Mode 1



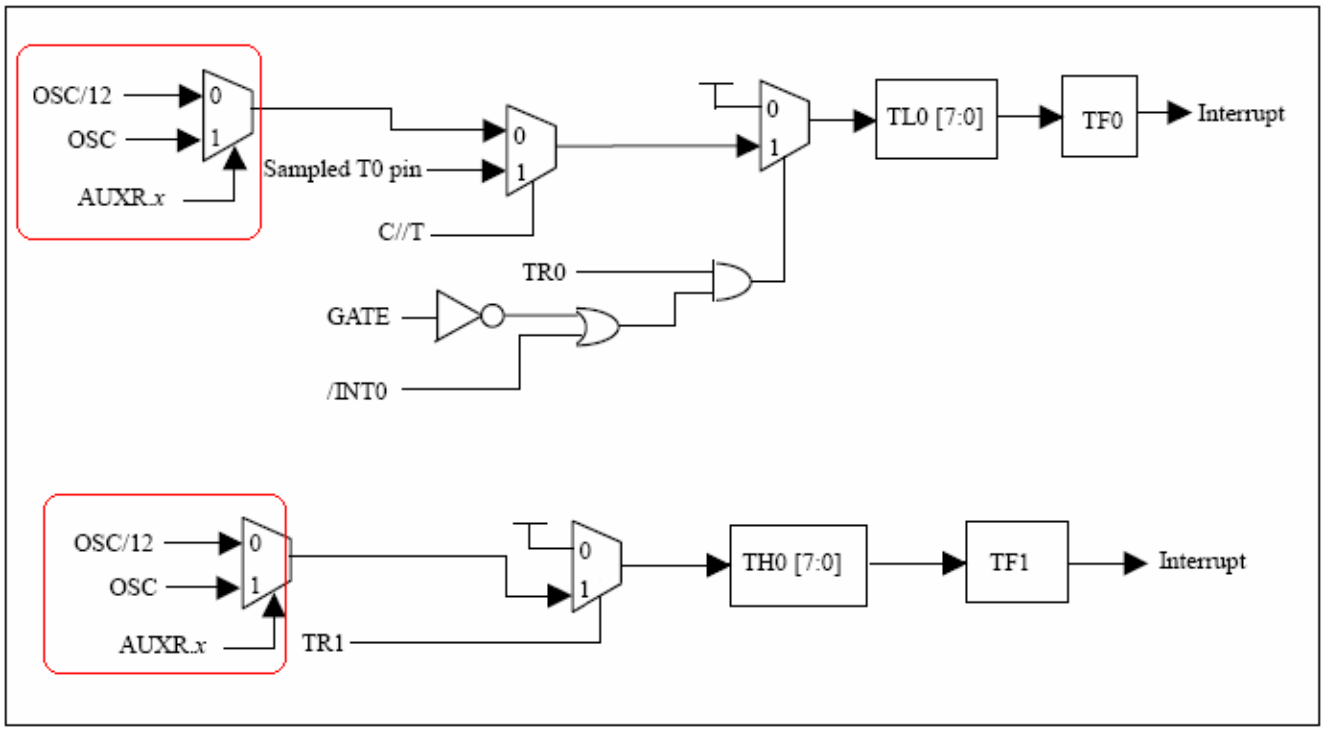
Where OSC means Fosc, the system clock.

4.3 Mode 2



Where OSC means Fosc, the system clock.

4.4 Mode 3



Where OSC means Fosc, the system clock.

5 Serial Port

5.1 Baudrate Setting

All the four operation modes of the serial port are the same as those of the standard 8051 except the baudrate setting. The bit-6 and bit-5 in AUXR provide a new option for the baudrate setting, as listed below.

Table: Baudrate Setting

Serial Port Mode	Compatible to standard 8051	New option
Mode 0	Baudrate = Fosc/12 **if AUXR.5 (URM0X6) is 0**	Baudrate = Fosc/2 **if AUXR.5 (URM0X6) is 1**
Mode 1 & 3	Baudrate = (2^{SMOD}/32) x Fosc/[12x(256-TH1)] **if AUXR.6 (T1X12) is 0**	Baudrate = (2^{SMOD}/32) x Fosc/(256-TH1) **if AUXR.6 (T1X12) is 1**
Mode 2	Baudrate = (2^{SMOD}/64) x Fosc	-

Where Fosc is the system clock.

AUXR (Auxiliary Register)

7	6	5	4	3	2	1	0
T0X12	**T1X12**	**URM0X6**	**EADCI**	**ESPI**	**ENLVFI**	**-**	**-**

T1X12: Timer 1 clock source select. Set to select Fosc as the clock source, and clear to select Fosc/12.
URM0X6: Serial Port mode 0 baudrate select.

Note:
Since these two bits are 0 after any reset, it is compatible to the standard 8051 by default.

5.2 Enhanced Feature: Frame Error Detection

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

SCON (Serial Port Control Register)

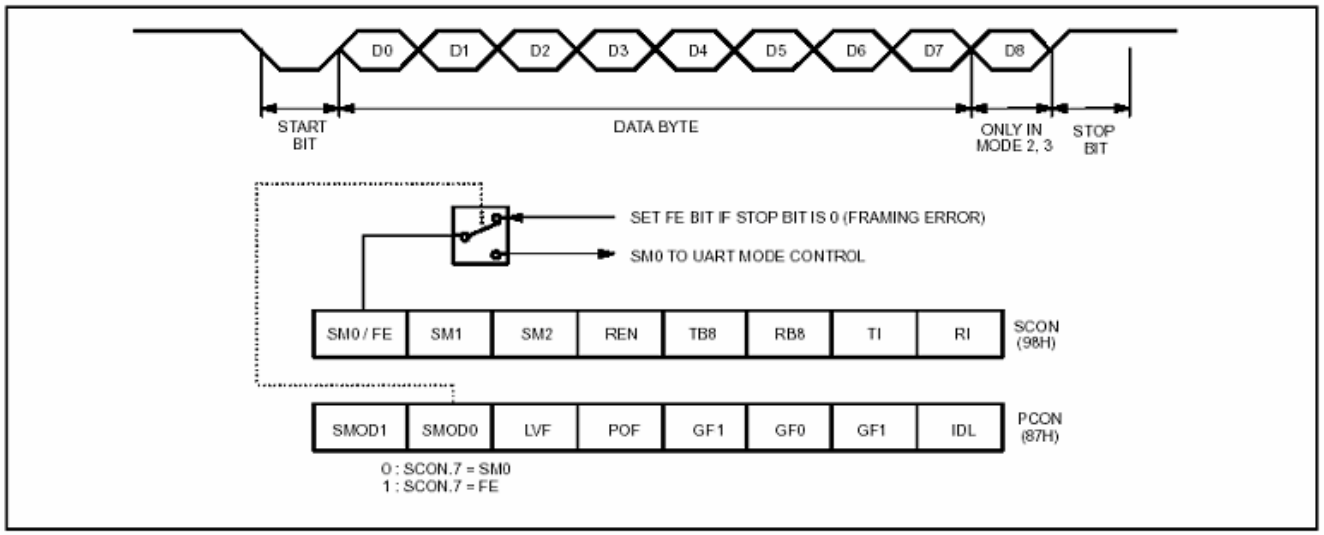
7	6	5	4	3	2	1	0
SM0/FE	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**

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

PCON (Power Control Register)

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

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



5.3 Enhanced Feature: Automatic Address Recognition

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

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

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

Using the Automatic Address Recognition feature allows a master to selectively communicate with one or more slaves by invoking the Given slave address or addresses. All of the slaves may be contacted by using the Broadcast address. Two special Function Registers are used to define the slave's address, SADDR, and the address mask, SADEN.

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

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

```
Slave 0
         SADDR = 1100 0000
         SADEN = 1111 1101
         Given = 1100 00X0

Slave 1
         SADDR = 1100 0000
         SADEN = 1111 1110
         Given = 1100 000X
```

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

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

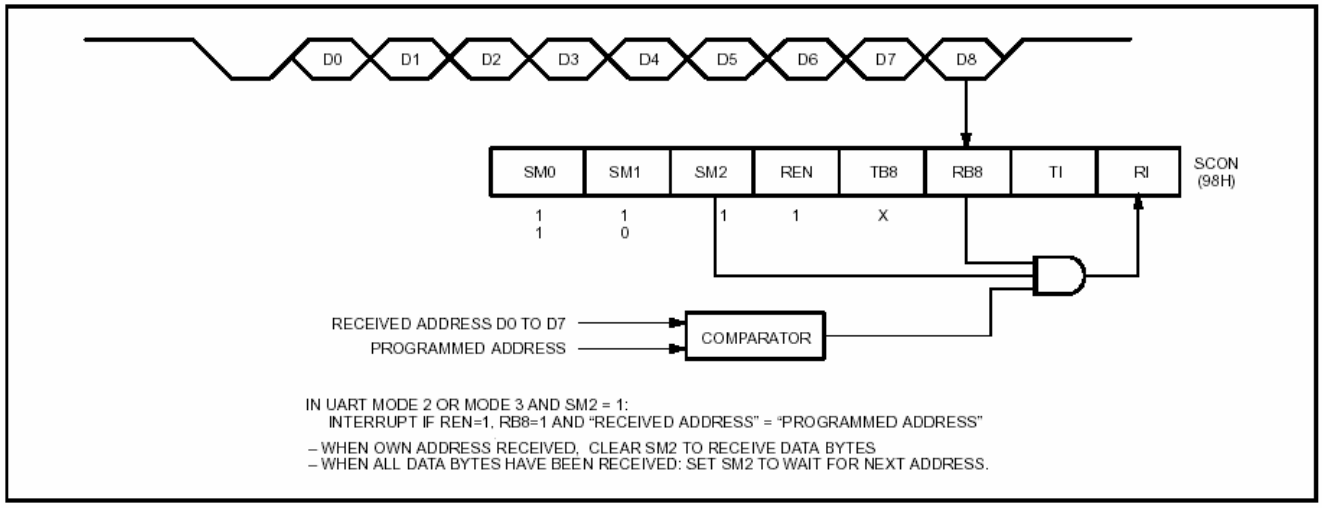
```
Slave 0
         SADDR = 1100 0000
         SADEN = 1111 1001
         Given = 1100 0XX0
Slave 1
         SADDR = 1110 0000
         SADEN = 1111 1010
         Given = 1110 0X0X
```

```
Slave 2
        SADDR = 1110 0000
        SADEN = 1111 1100
        Given = 1110 00XX
```

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

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

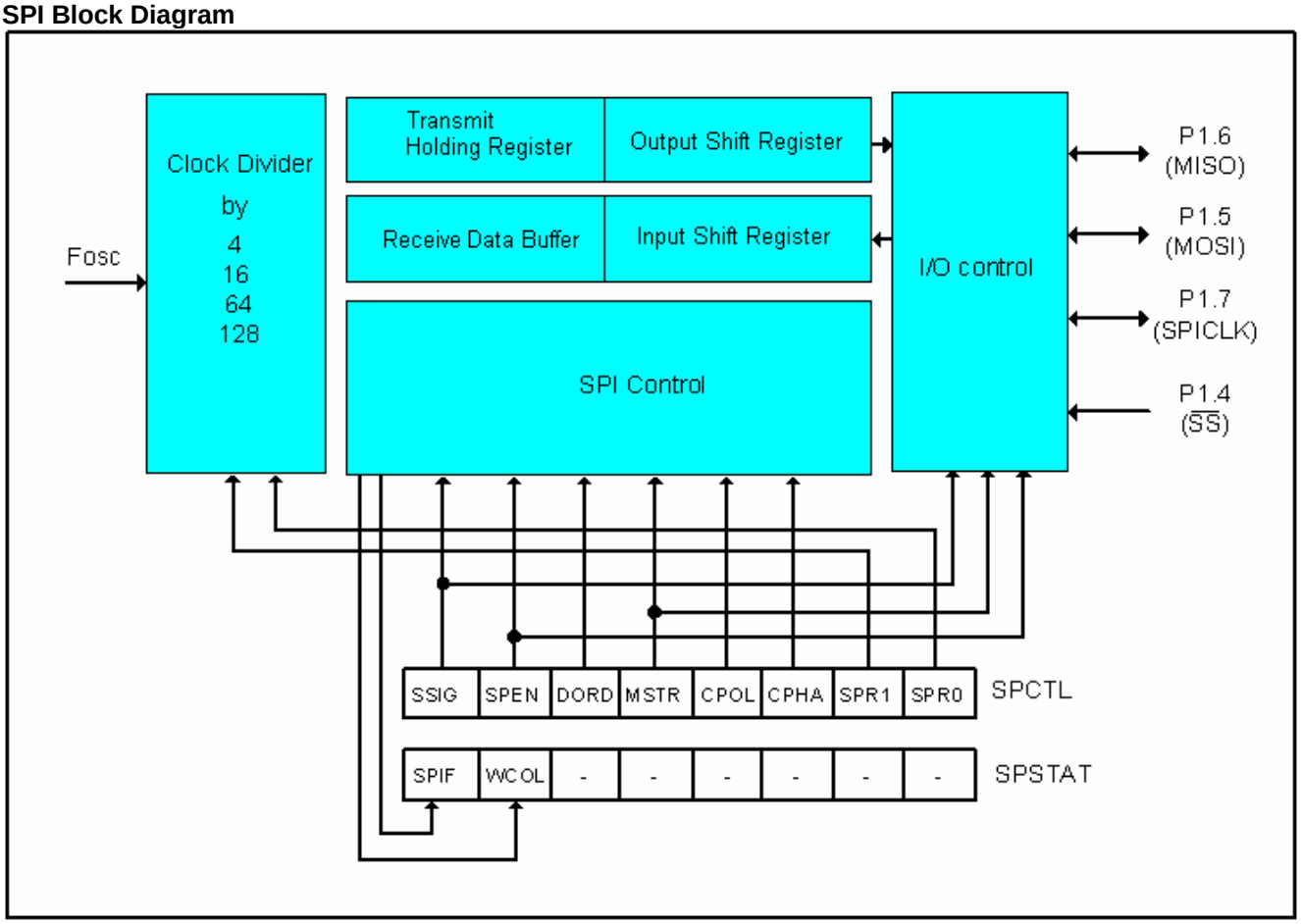
Upon reset SADDR (SFR address 0A9H) and SADEN (SFR address 0B9H) are leaded with 0s. This produces a given address of all "don't cares" as well as a Broadcast address of all "don't cares". This effectively disables the Automatic Addressing mode and allows the micro-controller to use standard 80C51 type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



6 Serial Peripheral Interface (SPI)

The MPC82L(E)52/54 provide another 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 the 12MHz system clock. It has a Transfer Completion Flag (SPIF) and Write Collision Flag (WCOL) in the SPI status register (SPSTAT register).

SPI Block Diagram



The SPI interface has four pins: SPICLK, MOSI, MISO and /SS:

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

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

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

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

SPCTL (SPI Control Register)

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

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

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

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

MSTR: Master/Slave mode select

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

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

SPR1-SPR0: SPI clock rate select (when in master mode)
00 : Fosc/4
01 : Fosc/16
10 : Fosc/64
11 : Fosc/128

Where, Fosc is the system clock.

SPSTAT (SPI Status Register)

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

SPIF: SPI transfer completion flag
When a serial transfer finishes, the SPIF bit is set and an interrupt is generated if the ESPI (AUXR.3) bit, the ESPI_ADC (IE.5) bit and the EA (IE.7) bit are set. If /SS pin is driven low when SPI is in master mode with SSIG=0, SPIF will also be set to signal the “mode change”. The SPIF is cleared in software by writing ‘1’ to this bit.

WCOL: SPI write collision flag. The WCOL bit is set if the SPI data register, SPDAT, is written during a data transfer. The WCOL flag is cleared in software by writing ‘1’ to this bit.

SPDAT (SPI Data Register)

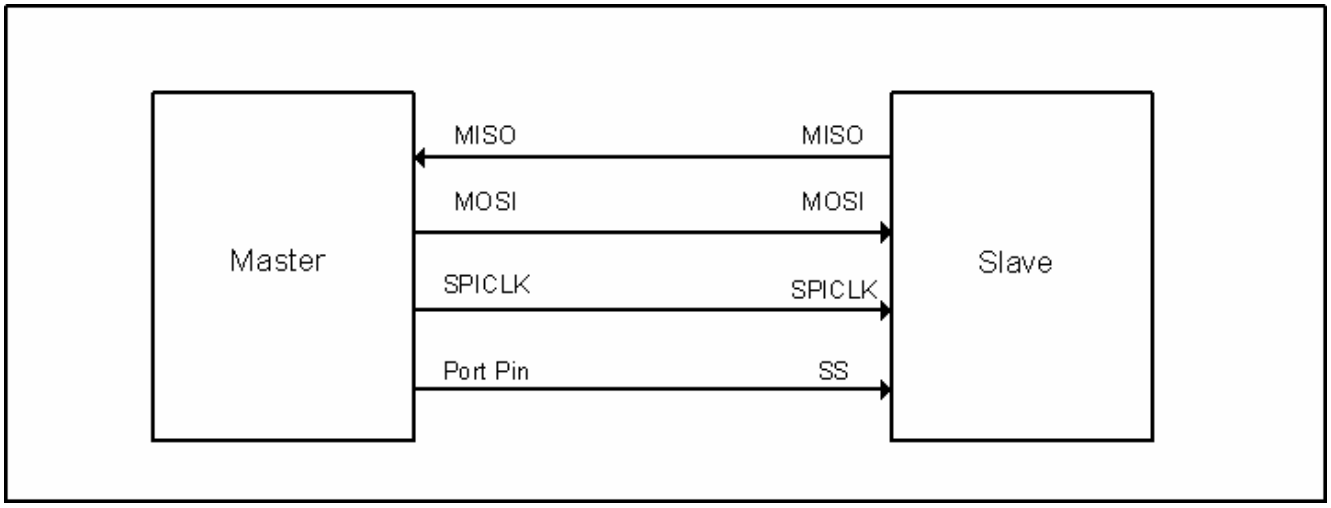
7	6	5	4	3	2	1	0
(MSB)							(LSB)

SPI data buffer for transmit and receive.

6.1 Typical SPI Configurations

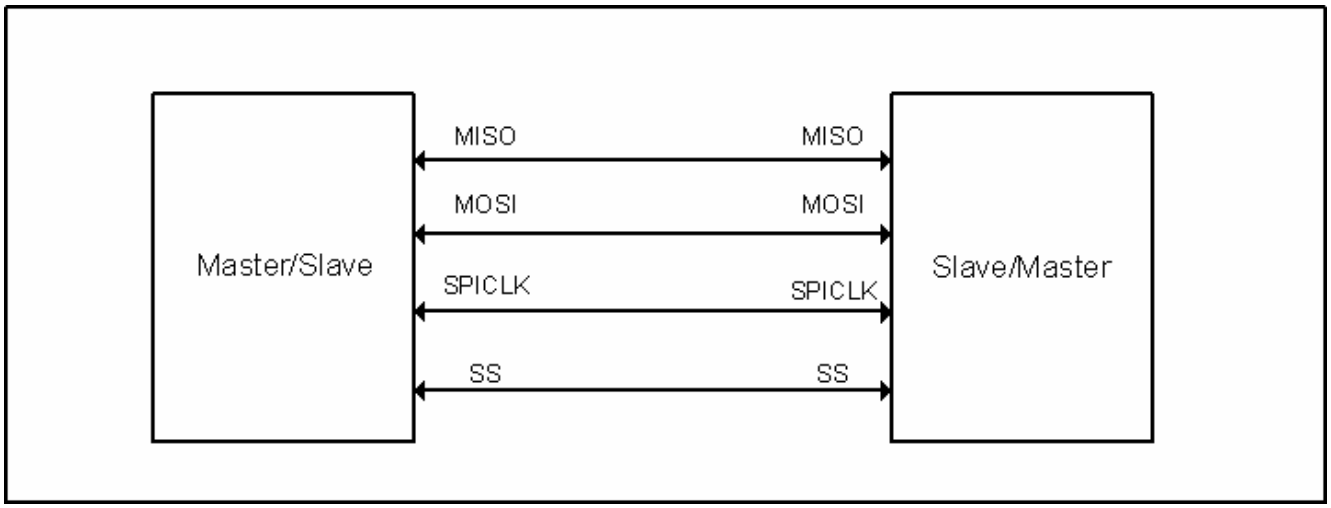
6.1.1 Single Master & Single Slave

For the master: can use any port pin, including P1.4 (/SS), to drive the /SS pin of the slave.
For the slave: SSIG is '0', and /SS pin is used to select the slave.



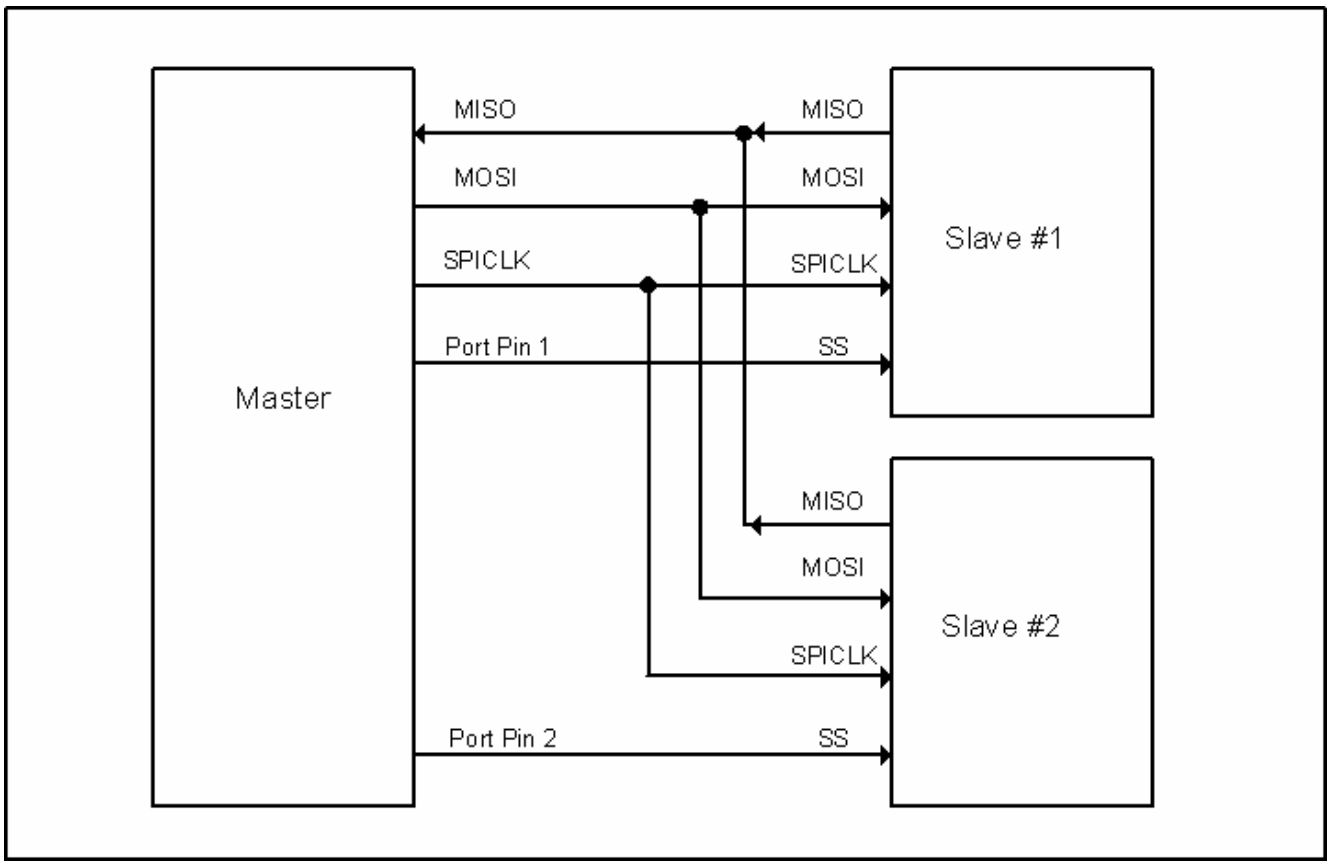
6.1.2 Dual Device, where either can be a Master or a Slave

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.



6.1.3 Single Master & Multiple Slaves

For the master: can use any port pin, including P1.4 (/SS) to drive the /SS pins of the slaves.
For all the slaves: SSIG is '0', and are selected by their corresponding /SS pins.



6.2 Configuring the SPI

Table: SPI Master and Slave Selection

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

6.2.1 Additional Considerations for a Slave

When CPHA is 0, SSIG must be 0 and /SS pin must be negated and reasserted between each successive serial byte transfer. If the SPDAT register is written while /SS pin is active (low), a write collision error results and WCOL is set. The operation is undefined if CPHA is 0 and SSIG is 1.

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

6.2.2 Additional Considerations for a Master

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

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

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

6.2.3 Mode Change on /SS-pin

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

6.2.4 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.

6.2.5 SPI Clock Rate Select

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

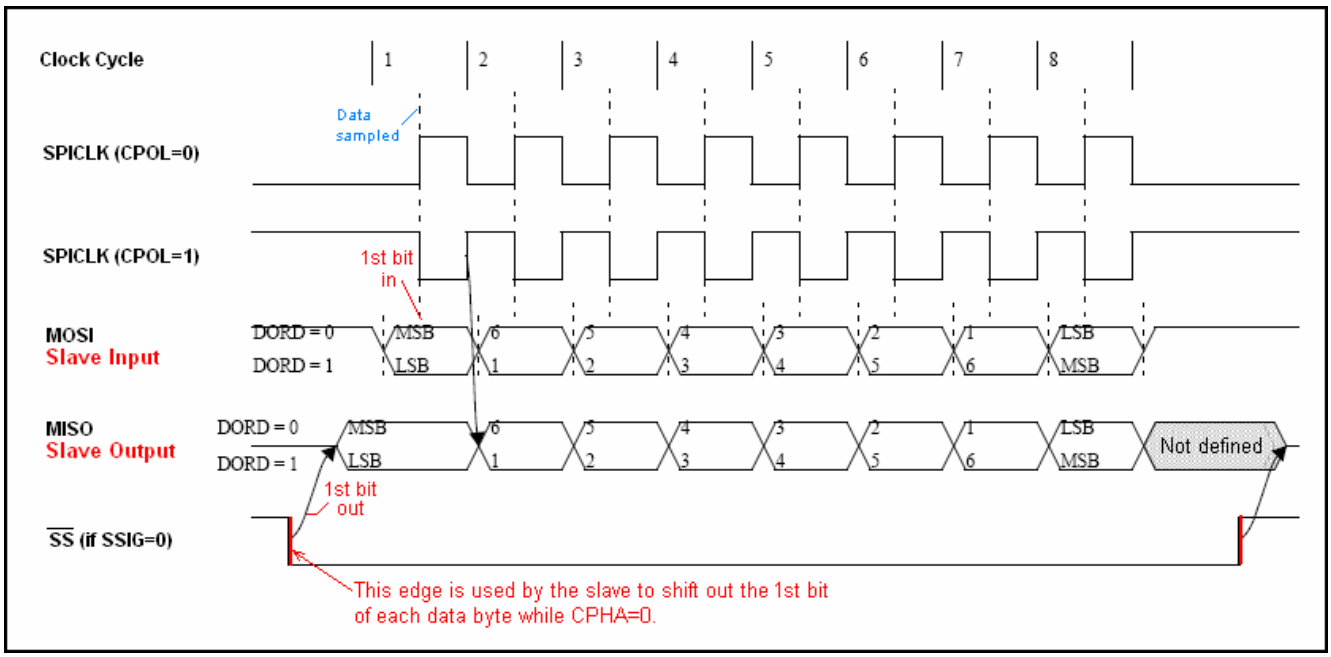
SPR1, SPR0	SPI Clock Rate
0, 0	Fosc/4
0, 1	Fosc/16
1, 0	Fosc/64
1, 1	Fosc/128

Where, Fosc is the system clock.

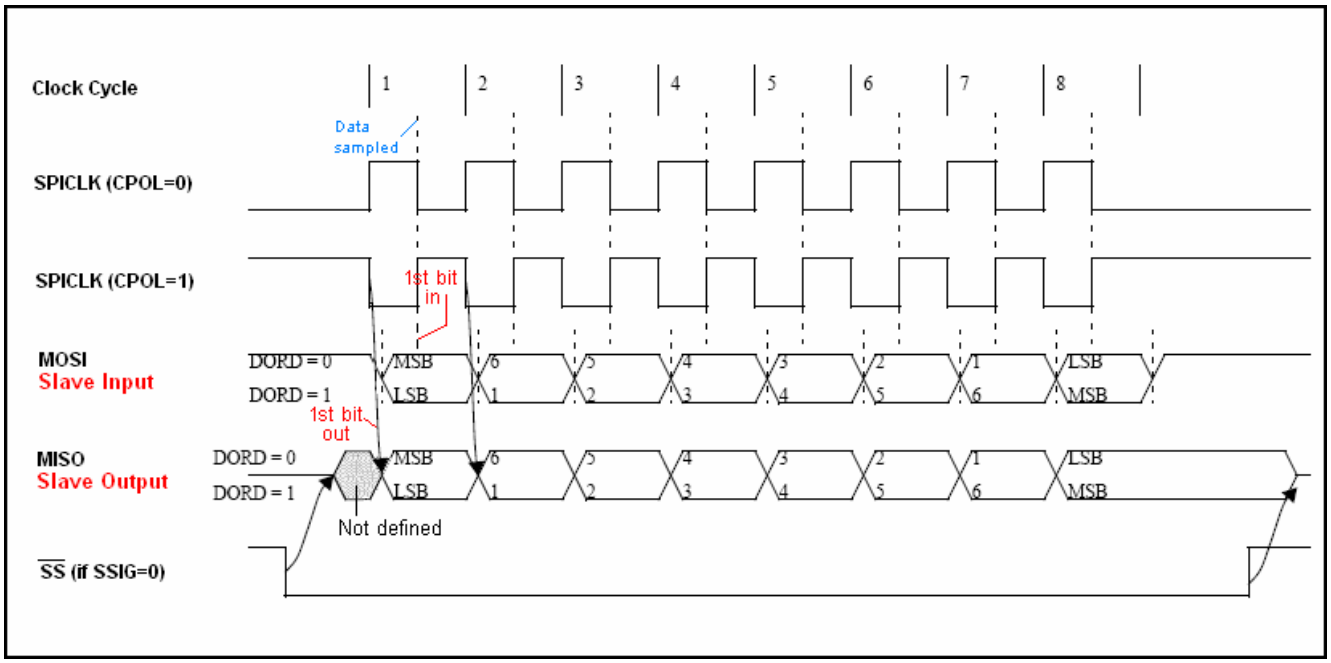
6.3 Data Mode

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

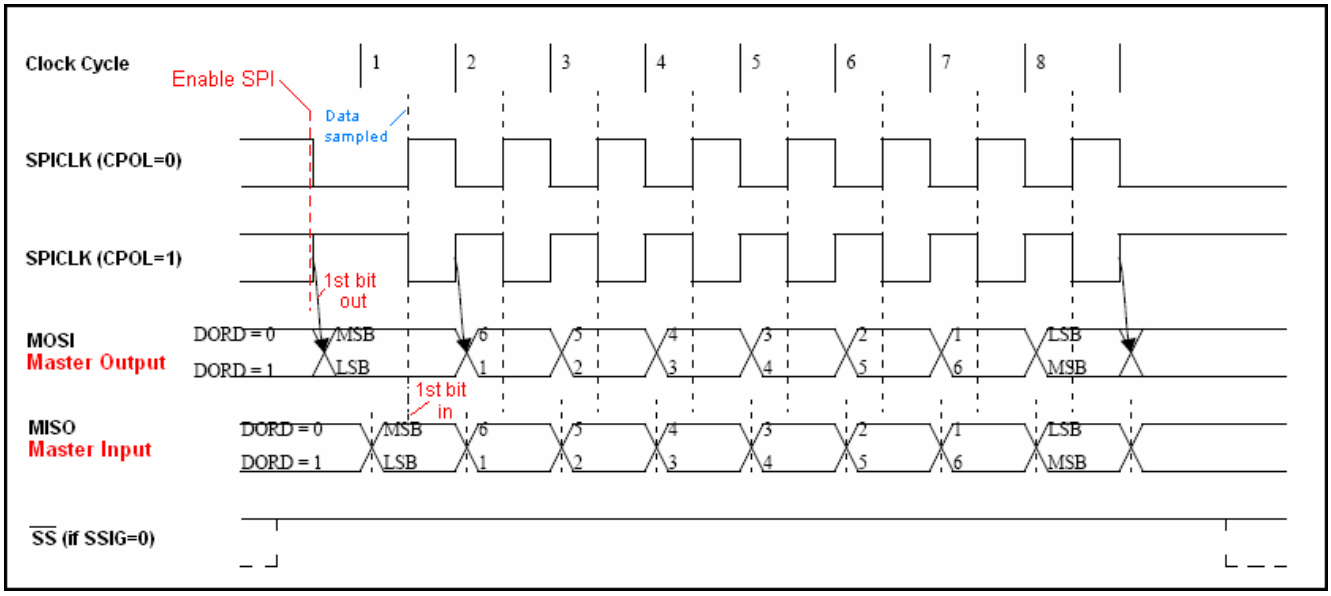
6.3.1 SPI Slave Transfer Format with CPHA=0



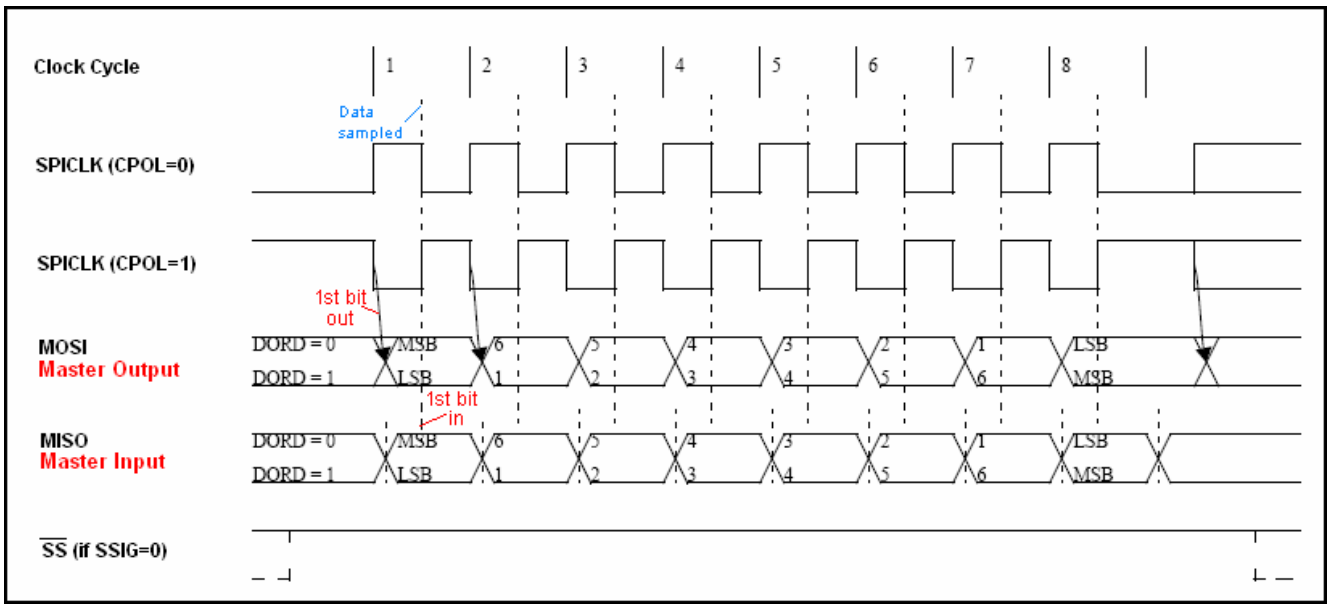
6.3.2 SPI Slave Transfer Format with CPHA=1



6.3.3 SPI Master Transfer Format with CPHA=0



6.3.4 SPI Master Transfer Format with CPHA=1

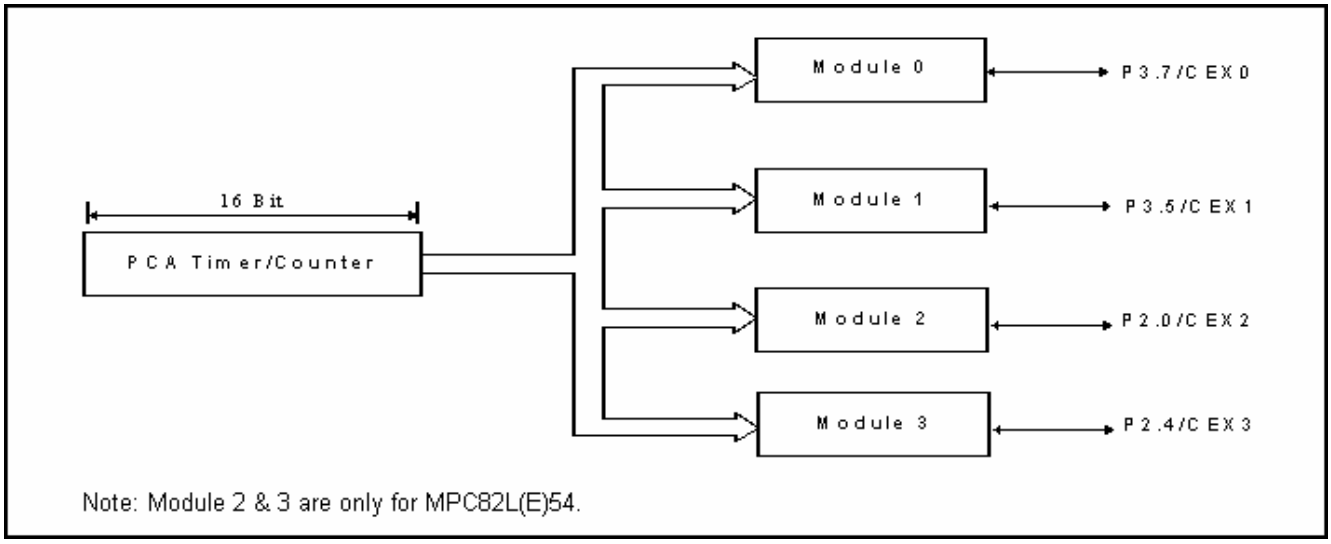


7 Programmable Counter Array (PCA)

7.1 Introduction to the PCA

The Programmable Counter Array (PCA) is a special 16-bit Timer that has four 16-bit capture/compare modules associated with it. Each of the modules can be programmed to operate in one of four modes: *rising and/or falling edge capture*, *software timer*, *high-speed output*, or *PWM (pulse width modulation) output*. Each module has a pin associated with it: module 0 is connected to P3.7, module 1 to P3.5, module 2 to P2.0 and module 3 to P2.4. The basic PCA configuration is shown in the figure as follows.

PCA Configuration



The PCA timer is a common time base for all modules and can be programmed to run at: 1/12 the oscillator frequency, 1/2 the oscillator frequency, the Timer 0 overflow, or the input on ECI pin (P3.4). The clock source for the timer is determined by the CPS1 and CPS0 bits in the CMOD register as follows.

CMOD (PCA Counter Mode Register)

7	6	5	4	3	2	1	0
CIDL	**-**	**-**	**-**	**-**	**CPS1**	**CPS0**	**ECF**

CIDL: PCA counter idle control.
CIDL=0 lets the PCA counter continue functioning during idle mode.
CIDL=1 lets the PCA counter be gated off during idle mode.

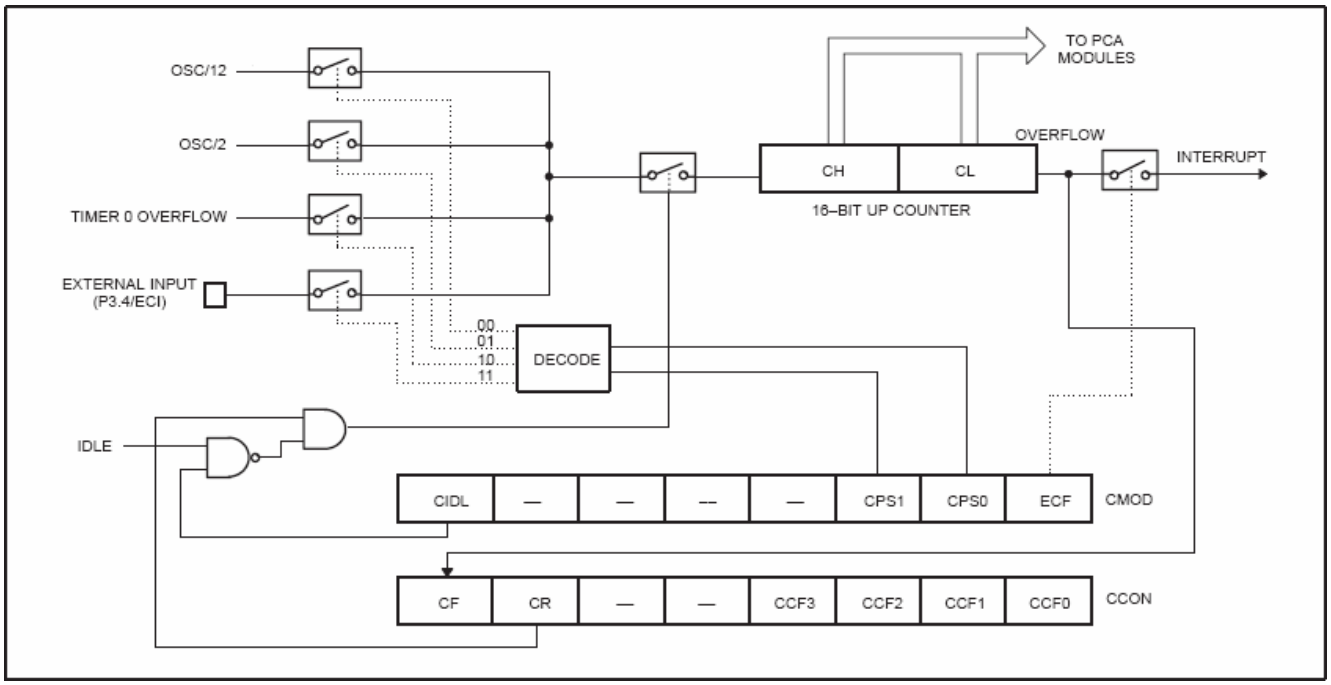
CPS1-CPS0: PCA counter clock source select bits.

0	0	Internal clock, Fosc/12 (*Where, Fosc is the system clock.)*
0	1	Internal clock, Fosc/2
1	0	Timer 0 overflow
1	1	External clock at the ECI pin (P3.4)

ECF: Enable PCA counter overflow interrupt.
ECF=1 enables an interrupt when CF bit (in CCON register) is set.

In the CMOD SFR are two additional bits associated with the PCA. They are CIDL which allows the PCA to stop during idle mode, and ECF which when set causes an interrupt if the PCA overflow flag CF (in the CCON SFR) is set either when the counter overflows or by software. The following figure shows these functions.

PCA Timer/Counter



Where OSC means Fosc, the system clock.

The CCON register 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 ~ CCF3 are the flags for module 0 ~ module 3, respectively, and they are set by hardware when either a match or a capture occurs. These flags also can only be cleared by software.

CCON (PCA Counter Control Register)

7	6	5	4	3	2	1	0
CF	**CR**	**-**	**-**	**CCF3**	**CCF2**	**CCF1**	**CCF0**

CF: PCA Counter Overflow flag. 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 but can only be cleared by software.

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

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

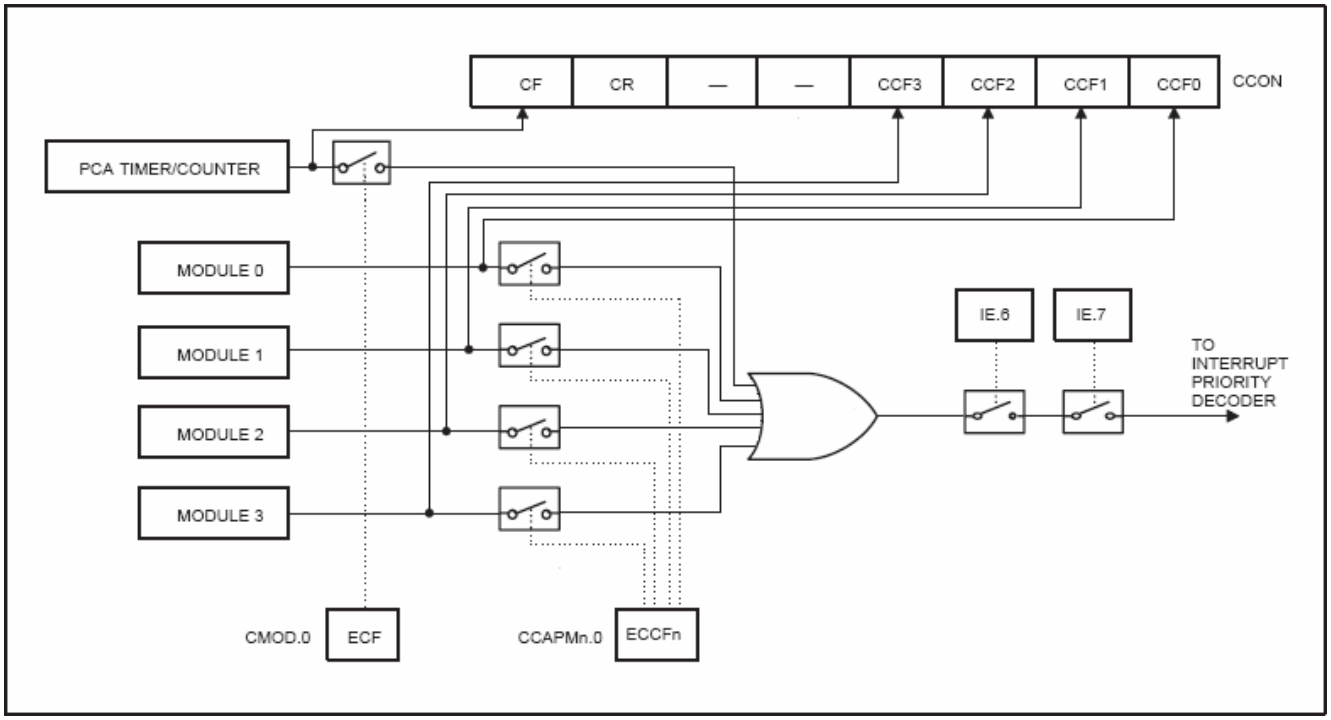
CCF2: PCA Module 2 interrupt flag.
Set by hardware when a match or capture occurs. Must be cleared by software.

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

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

The PCA interrupt system is shown below.

PCA Interrupt System



Each module in the PCA has a special function register associated with it. These registers are CCAPMn, where n = 0,1, 2 and 3 for module 0 to module 3, respectively. See the register description as follows.

CCAPM0/CCAPM1/CCAPM2/CCAPM3 (PCA Module Compare/Capture Registers)

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

ECOMn: Enable Comparator. ECOMn=1 enables the comparator function.

CAPPn: Capture Positive. CAPPn=1 enables positive edge capture.

CAPNn: Capture Negative. CAPNn=1 enables negative edge capture.

MATn: Match control. When MATn=1, a match of the PCA counter with this module's compare/capture Register causes the CCFn bit in CCON to be set.

TOGn: Toggle control. When TOGn=1, a match of the PCA counter with this module's compare/capture register causes the CEXn pin to toggle.

PWMn: PWM control. PWMn=1 enables the CEXn pin to be used as a pulse width modulated output.

ECCFn: Enable CCFn interrupt. Enables 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.

There are two additional registers associated with each of the PCA modules: CCAPnH and CCAPnL. They are the registers that store the 16-bit count when a capture occurs or a compare 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.

PCAPWM0/PCAPWM1/PCAPWM2/PCAPWM3 (PWM Mode Auxiliary Registers)

7	6	5	4	3	2	1	0
-	**-**	**-**	**-**	**-**	**-**	**ECAPnH**	**ECAPnL**

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.

7.2 Operation Modes of the PCA

The following Table shows the CCAPMn register settings for the various PCA functions.

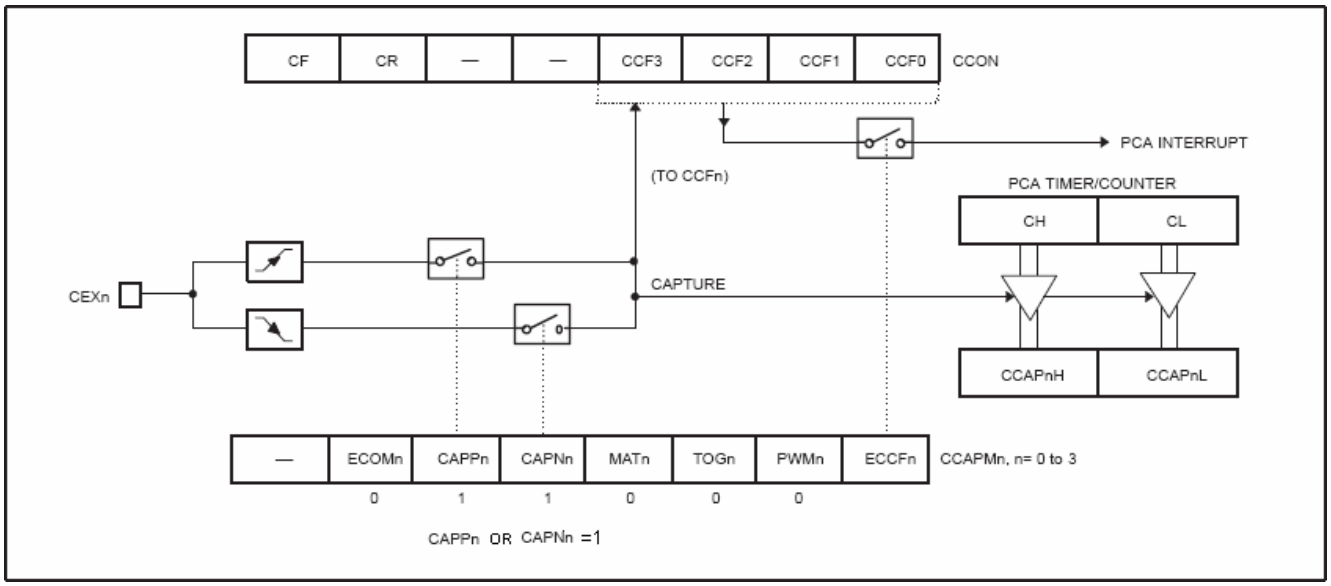
Table: PCA Module Modes

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)

7.2.1 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.

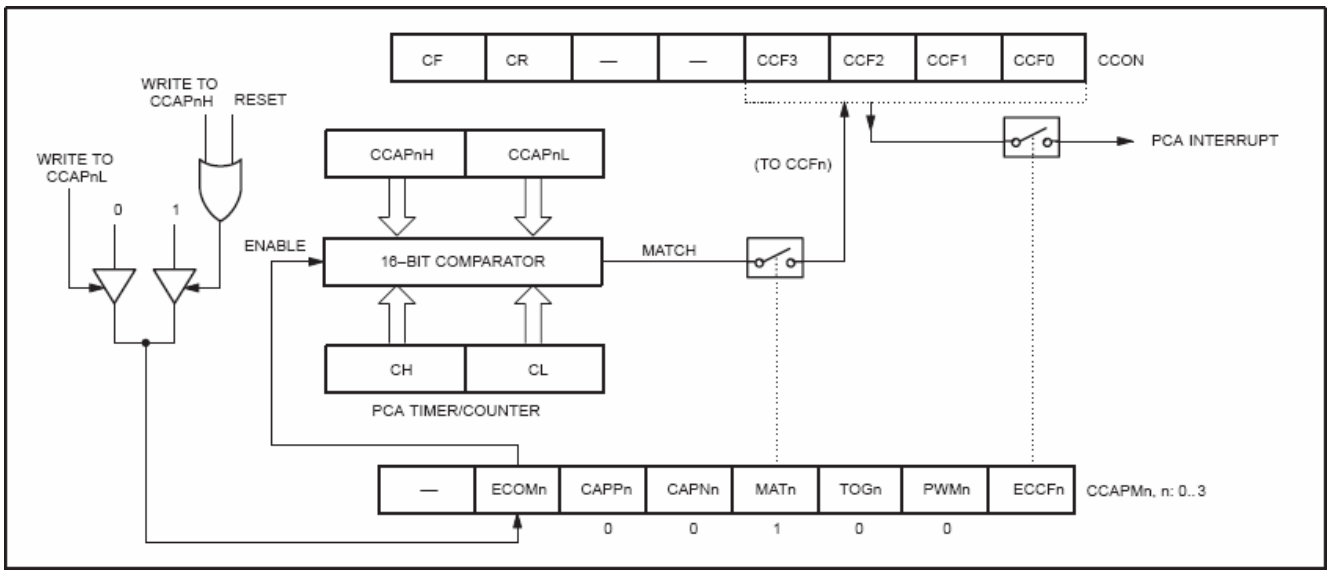
Function Block Diagram



7.2.2 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.

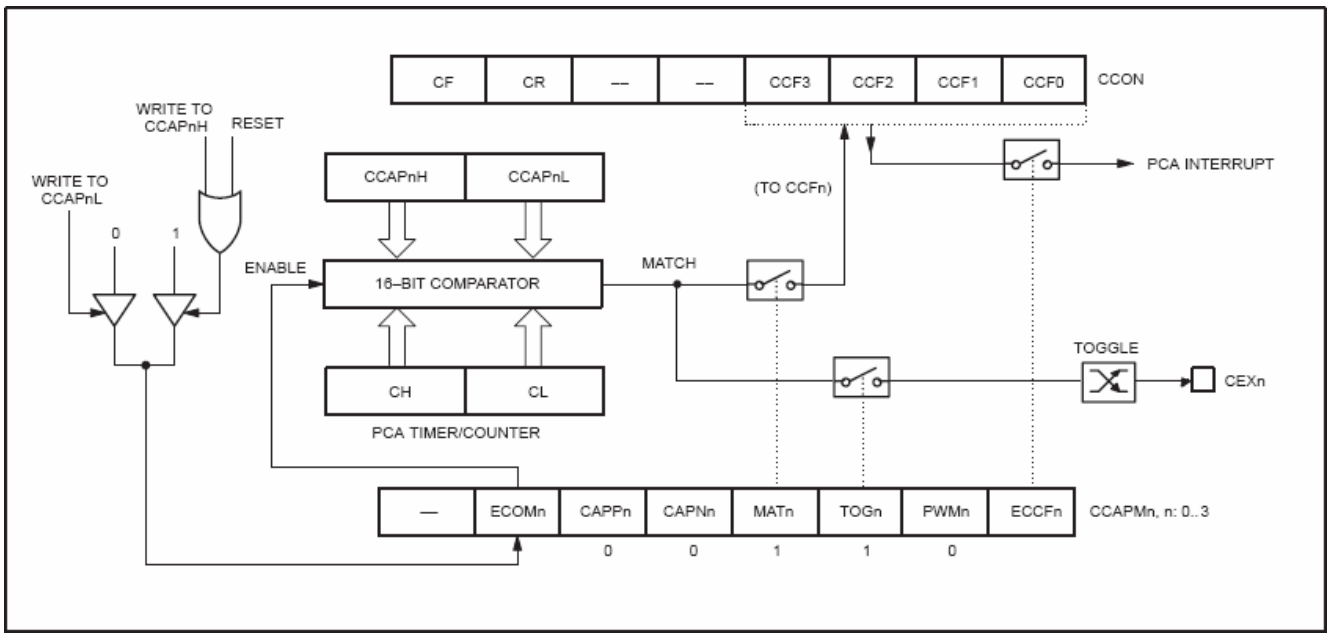
Function Block Diagram



7.2.3 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.

Function Block Diagram



7.2.4 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.

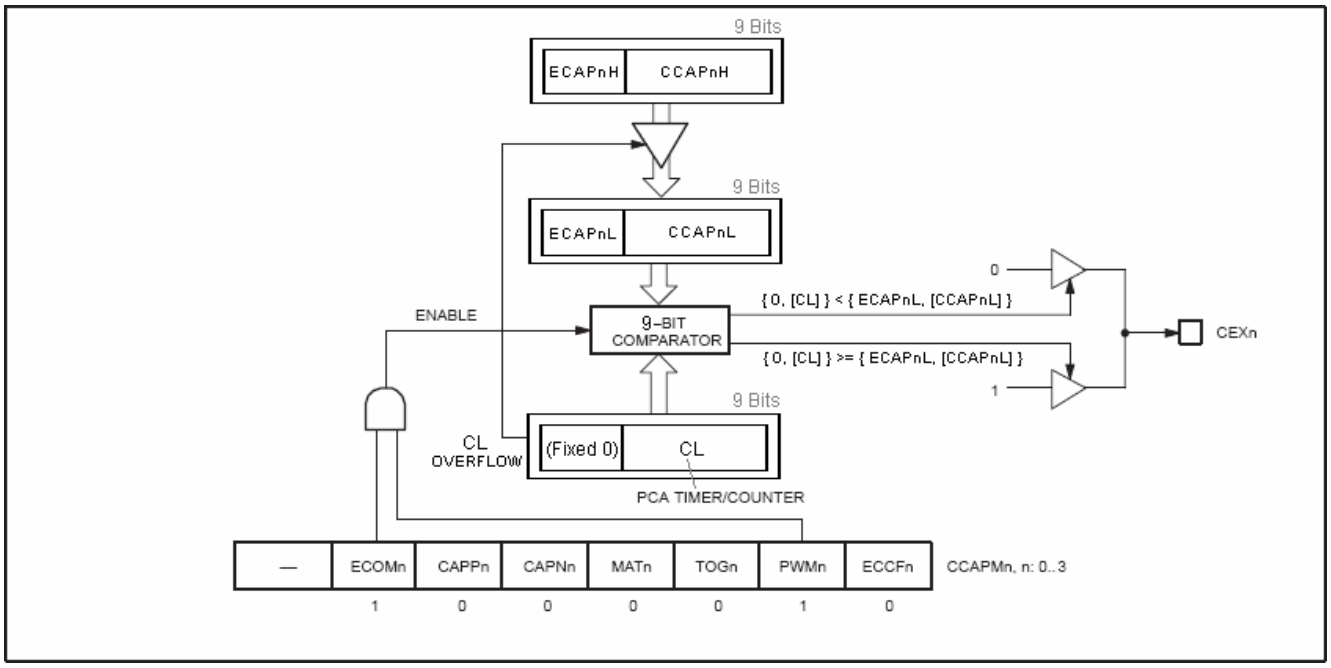
Using the 9-bit comparison, the duty cycle of the output can be improved to really start from 0%, and up to 100%. The formula for the duty cycle is:

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%.

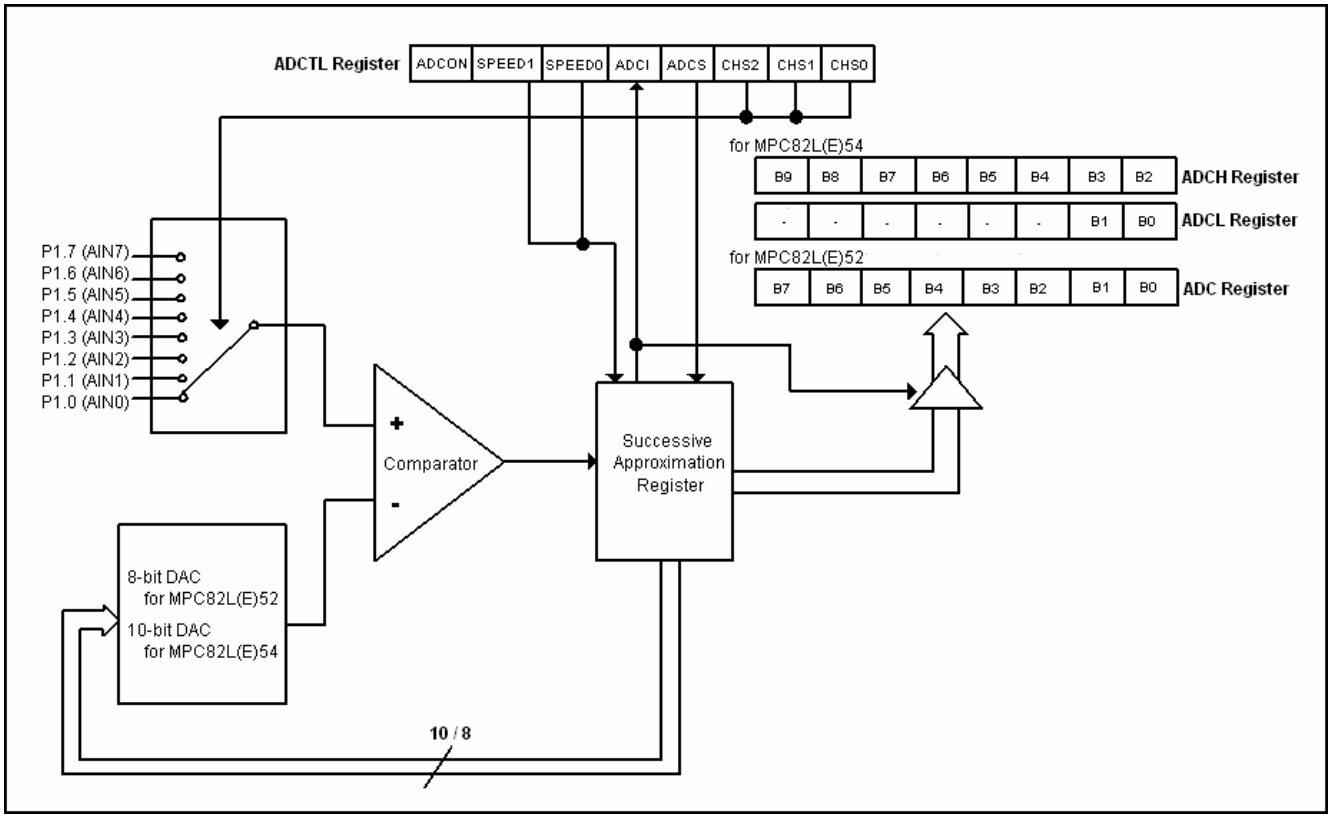
Function Block Diagram



8 Analog-to-Digital Converter (ADC)

MPC82L(E)52 and MPC82L(E)54 have a built-in ADC (implemented by SAR approach) with 8-bit and 10-bit resolution, respectively. The ADC has eight multiplexed analog inputs sharing pins of Port 1, a control register ADCTL and conversion result registers ADC, ADCH & ADCL, as shown in the block diagram.

ADC Block Diagram



ADCTL (AD Control Register)

7	6	5	4	3	2	1	0
ADCON	**SPEED1**	**SPEED0**	**ADCI**	**ADCS**	**CHS2**	**CHS1**	**CHS0**

ADCON: Cleared to turn off the ADC block. Set to turn on the ADC block.

SPEED1~SPEED0: conversion speed selection.
- 00: 840 clock cycles are taken for a conversion.
- 01: 630 clock cycles are taken for a conversion.
- 10: 420 clock cycles are taken for a conversion.
- 11: 210 clock cycles are taken for a conversion.

ADCS : ADC start bit, which is set by software and automatically cleared by hardware.

ADCI : ADC interrupt flag, which is set by hardware and should be cleared by software.

CHS2~CHS0: channel selection.
- 000: select P1.0 as the analog input
- 001: select P1.1 as the analog input
- 010: select P1.2 as the analog input
- 011: select P1.3 as the analog input
- 100: select P1.4 as the analog input
- 101: select P1.5 as the analog input
- 110: select P1.6 as the analog input
- 111: select P1.7 as the analog input

Prior to using the ADC function, users should:

1) enable the ADC block by setting the ADCON bit,
2) select the analog input pin by bits CHS2~CHS0, and
3) configure the selected pin to its *Input-Only* mode by P1M0 and P1M1 registers.

Now, user can start the A-to-D conversion by setting the ADCS bit. The conversion time is controlled by bits SPEED1 and SPEED0. Normally, 210 clock cycles are enough for a conversion under Fosc=12MHz. For higher Fosc, 420 clock cycles may be needed.

Once the conversion is finished, the hardware will:

1) automatically clear the ADCS bit,
2) load the conversion result into the **ADC** register for MPC82L(E)52 and **[ADCH, ADCL]** registers for MPC82L(E)54, and
3) set the interrupt flag ADCI.

User can check if the conversion is finished by polling the ADCI flag. If the ADC interrupt is enabled by setting bits *EADCI (AUXR.4)* and *ESPI_ADC (IE.5)*, the CPU will enter its *Interrupt Service Routine* when the conversion is completed. And, the ADCI flag should be cleared by software. (Refer to Section 9 for interrupt of the ADC.)

Note the ADC should be powered off before entering idle mode to reduce power consumption.

9 Interrupt

The MPC82L(E)52/54 has a 9-source 4-level interrupt structure. The 9 interrupt sources and their interrupt vectors are shown in the following table.

Table : Interrupt Sources

Source	Request Bits	Polling Priority	Vector Address
INT0	IE0	(highest priority)	0003H
Timer 0	TF0	.	000BH
INT1	IE1	.	0013H
Timer 1	TF1	.	001BH
Serial Port	RI+TI	.	0023H
SPI or ADC	SPIF+ADCI	.	002BH
PCA or Low Voltage Detect	CF+CCFn+LVF (n=0~3)	(lowest priority)	0033H

Note:
CCF2 and CCF3 are available only in MPC82L(E)54.

The following SFRs are associated with the interrupts. Among them, the IPH (Interrupt Priority High) register makes the four-level interrupt structure possible.

IE (Interrupt Enable Register)

7	6	5	4	3	2	1	0
EA	**EPCA_LVD**	**ESPI_ADC**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

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

EPCA_LVD: PCA & LVD interrupts enable bit

ESPI_ADC: SPI & ADC interrupts enable bit.

ES: Serial Port interrupt enable bit.

ET1: Timer 1 interrupt enable bit.

EX1: External interrupt 1 enable bit.

ET0: Timer 0 interrupt enable bit.

EX0: External interrupt 0 enable bit.

IP (Interrupt Priority Register)

7	6	5	4	3	2	1	0
-	**PPCA_LVD**	**PSPI_ADC**	**PS**	**PT1**	**PX1**	**PT0**	**PX0**

PPCA_LVD: PCA & LVD interrupts priority bit

PSPI_ADC: SPI & ADC interrupts priority bit.

PS: Serial Port interrupt priority bit.

PT1: Timer 1 interrupt priority bit.

PX1: External interrupt 1 priority bit.

PT0: Timer 0 interrupt priority bit.

PX0: External interrupt 0 priority bit.

IPH (Interrupt Priority High Register)

7	6	5	4	3	2	1	0
-	**PPCAH_LVD**	**PSPIH_ADC**	**PSH**	**PT1H**	**PX1H**	**PT0H**	**PX0H**

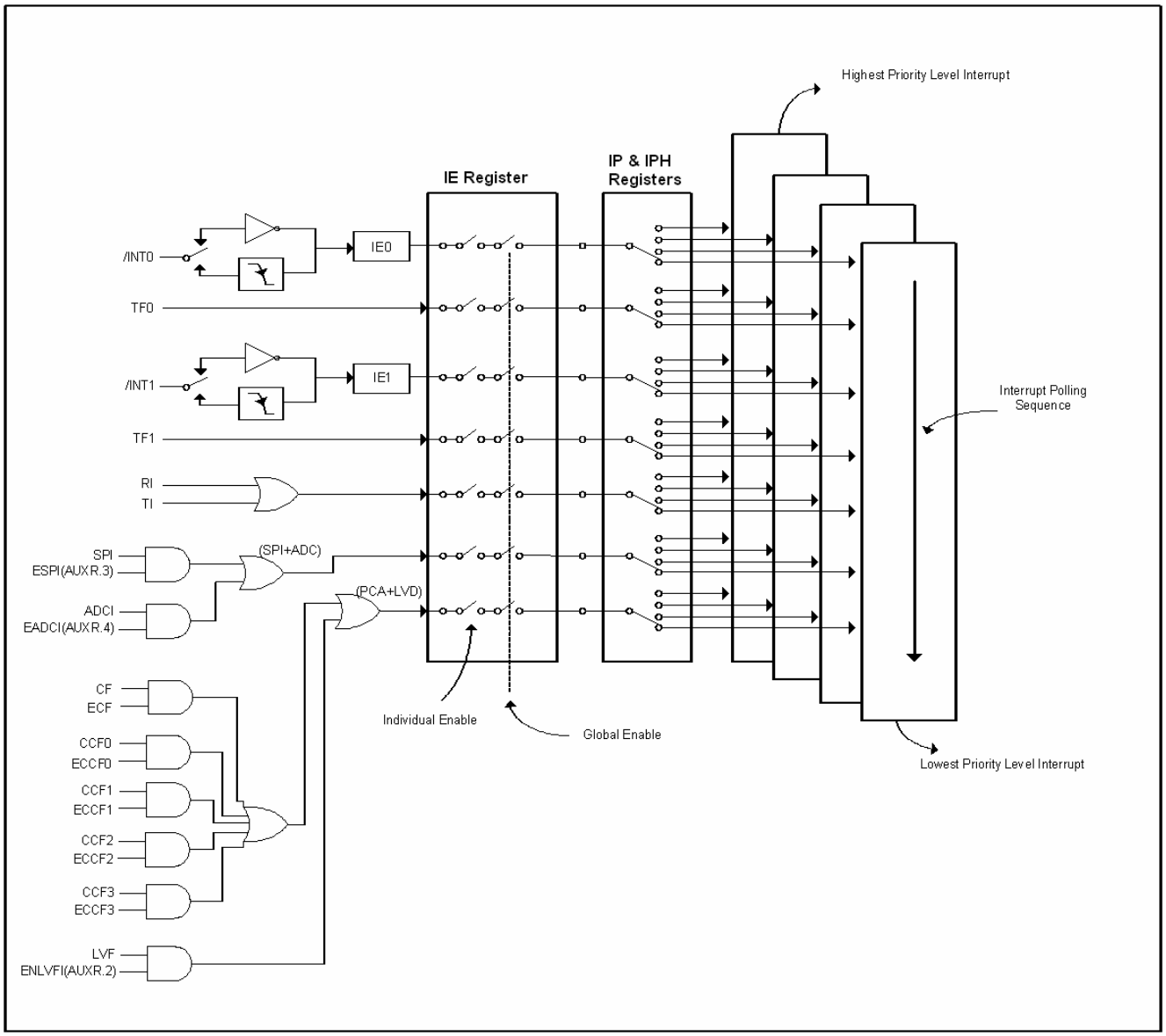
PPCAH_LVD: PCA & LVD interrupts priority bit high.
PSPIH_ADC: SPI & ADC interrupts priority bit high.
PSH: Serial Port interrupt priority bit high.
PT1H: Timer 1 interrupt priority bit high.
PX1H: External interrupt 1 priority bit high.
PT0H: Timer 0 interrupt priority bit high.
PX0H: External interrupt 0 priority bit high.

The IPH register, when combined with the IP register, determines the priority of each interrupt. The priority of each interrupt is determined as shown in the following table:

IPH.x	IP.x	Interrupt Priority Level
1	1	Level 3 (highest priority)
1	0	Level 2
0	1	Level 1
0	0	Level 0 (lowest priority)

For example, if (IPH.3, IP.3)=(1, 0), then Timer 1 has the priority level equal to 2, which is higher than level 1 with (IPH.3, IP.3)=(0, 1) or level 0 with (IPH.3, IP.3)=(0, 0).

9.1 Interrupt Architecture



Note:
CCF2 and CCF3 are available only in MPC82L(E)54.

9.2 Note on Interrupt during ISP/IAP

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

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

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

10 One-time Enabled Watchdog Timer (WDT)

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upset. The WDT consists of a 15-bit free-running counter, an 8-bit prescaler and a control register (WDTCR). To enable the WDT, users must set ENW bit (WDTCR.5). When the WDT is enabled, the counter will increment one by an interval of (***12 x Prescaler / Fosc***). And now the user needs to clear it by writing "1" to the CLRW bit (WDTCR.4) before WDT overflows. When WDT overflows, the MCU will reset itself and re-start.

Why is the WDT called "One-time Enabled"? It is because: *Once the WDT is enabled (ENW=1), there is no way to disable it except power-on reset (ENW=0).* The WDTCR register will keep the previous programmed value unchanged after hardware (RST-pin) reset, software reset and WDT reset. For example, if the WDTCR is 0x2D, it still keeps at 0x2D rather than 0x00 after these resets. Only power-on reset can initialize it to 0x00.

WDTCR (Watch-Dog-Timer Control Register)

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

Note: This is a Write-only register.

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

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

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

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

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

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

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

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

(1) set ENW bit,

(2) load **HWWIDL** into WIDL bit, and

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

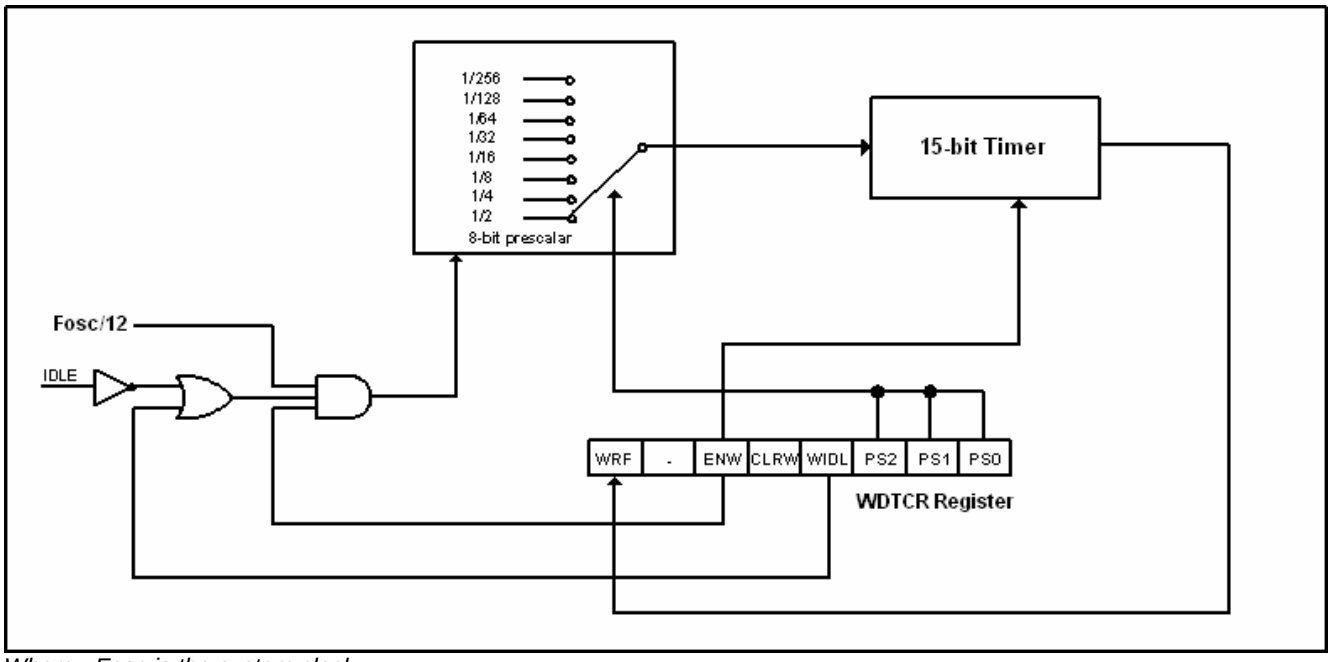
For example:

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

WDTCR (Watch-Dog-Timer Control Register)

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

WDT Block Diagram



Where, Fosc is the system clock.

WDT overflow period

The WDT overflow period is determined by the formula:

2^{15} x (12 x Prescaler / Fosc).

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

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

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

Sample code for WDT

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

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

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

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

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

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

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

11 In System Programming (ISP)

(First, please refer to Section 1.3 for Flash Memory Configuration.)

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

Prior to using the ISP feature, users should configure an ISP-memory using a universal Writer or Programmer.

Several SFRs are related to ISP:

IFADRH: ISP Flash address high register

IFADRL: ISP Flash address low register

IFD: ISP Flash data register

SCMD: ISP sequential command register (filled sequentially with 0x46 then 0xB9 to trigger ISP operation)

ISPCR (ISP Control Register)

7	6	5	4	3	2	1	0
ISPEN	**SWBS**	**SWRST**	**CFAIL**	**-**	**CKS2**	**CKS1**	**CKS0**

ISPEN: Set to enable ISP function.

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

SWRST: Write 1 to trigger software reset.

CFAIL: ISP fail flag. It is set by H/W if something error during ISP processing, and should be cleared by S/W.

CKS2~CKS0: Set the ISP timing according to Fosc (the system clock), as listed below.

Fosc	**CKS2**	**CKS1**	**CKS0**
30 ~ 24 MHz	0	0	0
24 ~ 20 MHz	0	0	1
20 ~ 12 MHz	0	1	0
12 ~ 6 MHz	0	1	1
6 ~ 3MHz	1	0	0
3 ~ 2MHz	1	0	1
2 ~ 1MHz	1	1	0
< 1MHz	1	1	1

IFMT (ISP Mode Register)

7	6	5	4	3	2	1	0
-	**-**	**-**	**-**	**-**	**-**	**MS1**	**MS0**

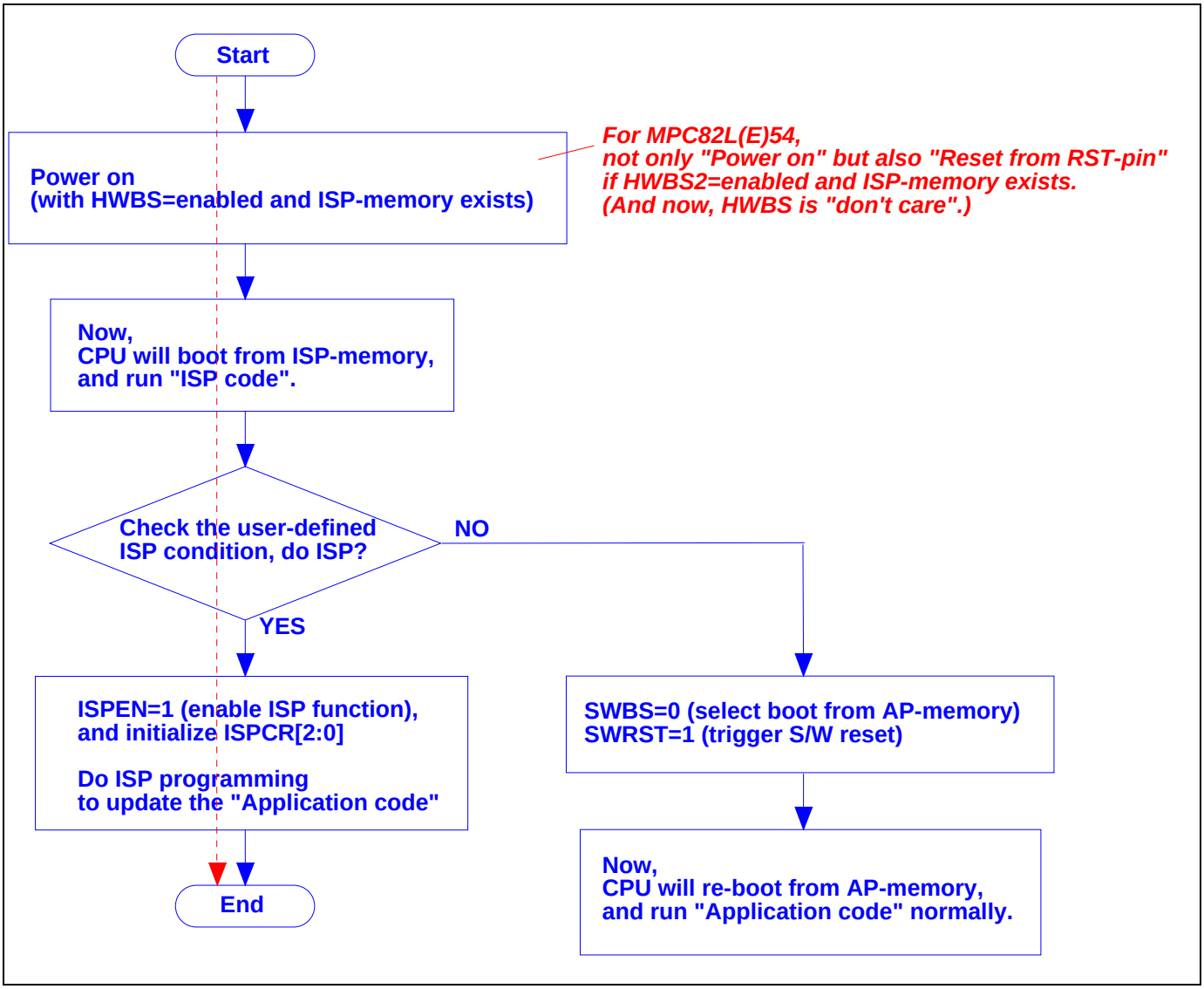
MS1 & MS0: Used to select the ISP mode, as listed below.

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

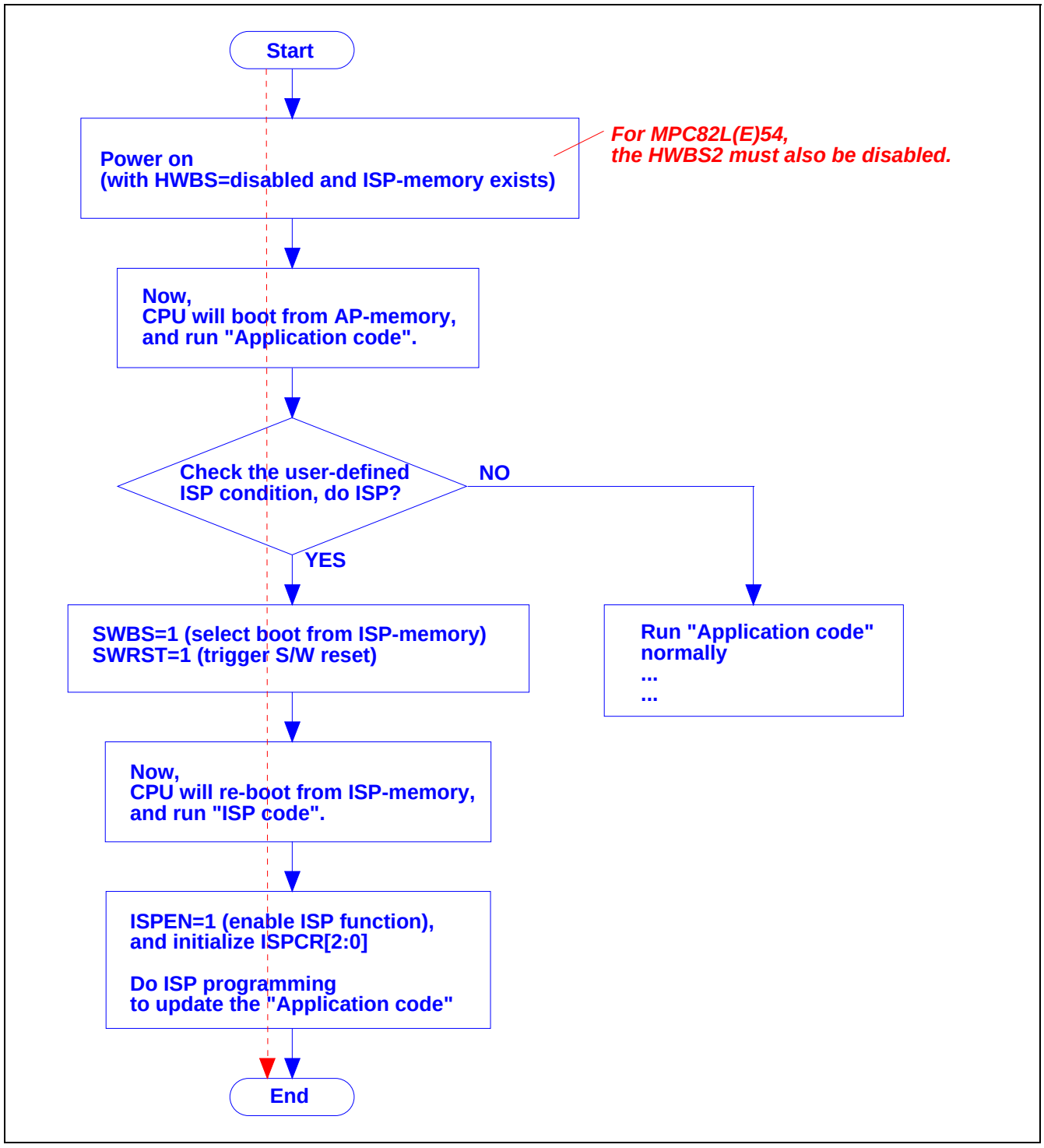
11.1 Boot from ISP-memory to run “ISP code”

To run the “ISP code”, the CPU should boot from the ISP-memory. By means of the following two methods, the CPU can boot from the ISP-memory.

Method 1: Directly Boot from ISP-memory (while HWBS or HWBS2 is enabled)



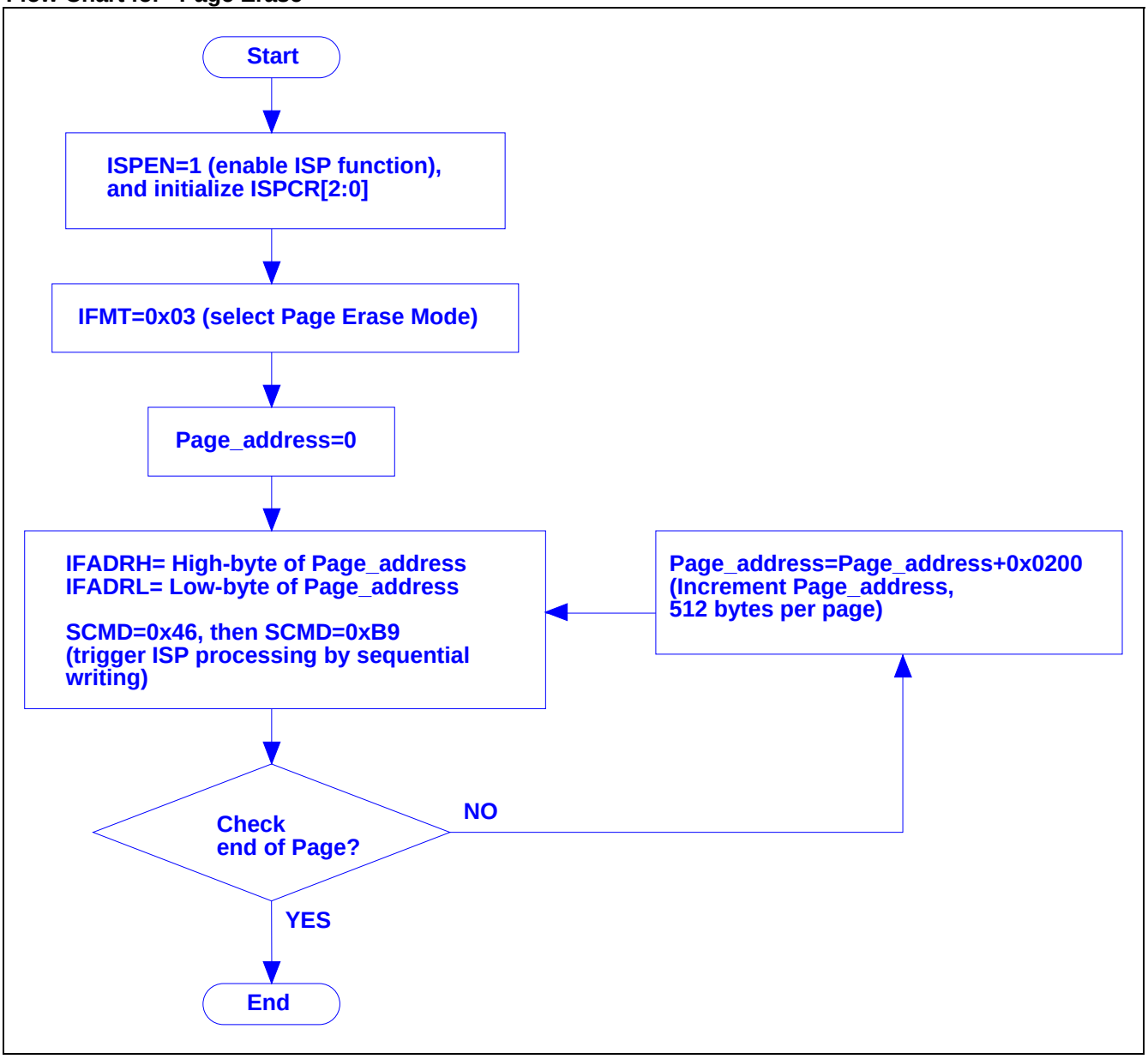
Method 2: Boot from ISP-memory through AP-memory (while HWBS and HWBS2 are disabled)



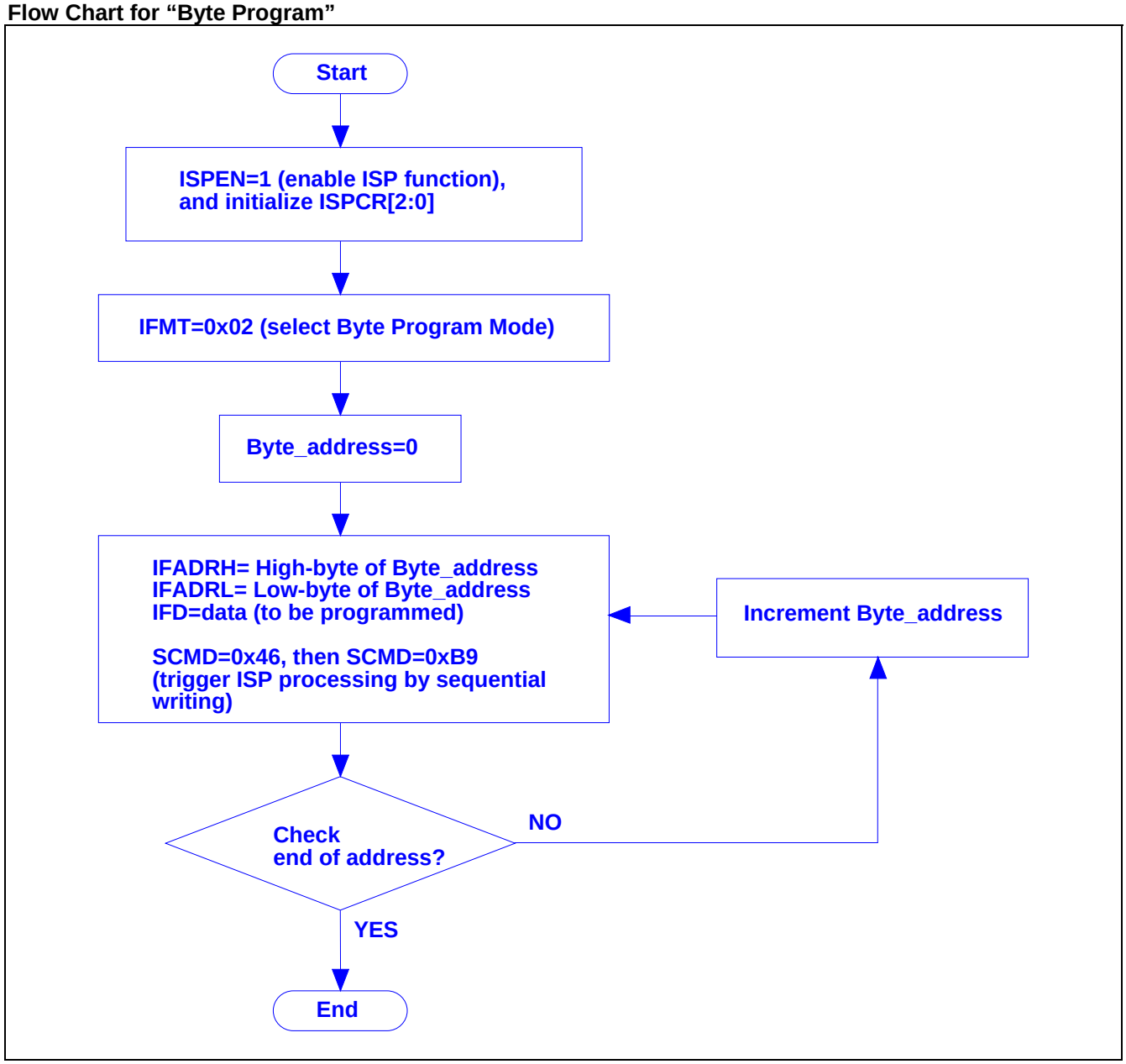
11.2 Operation Flow of ISP

The following figures show the flow chart for the various ISP modes used in the “ISP code” for ISP processing.

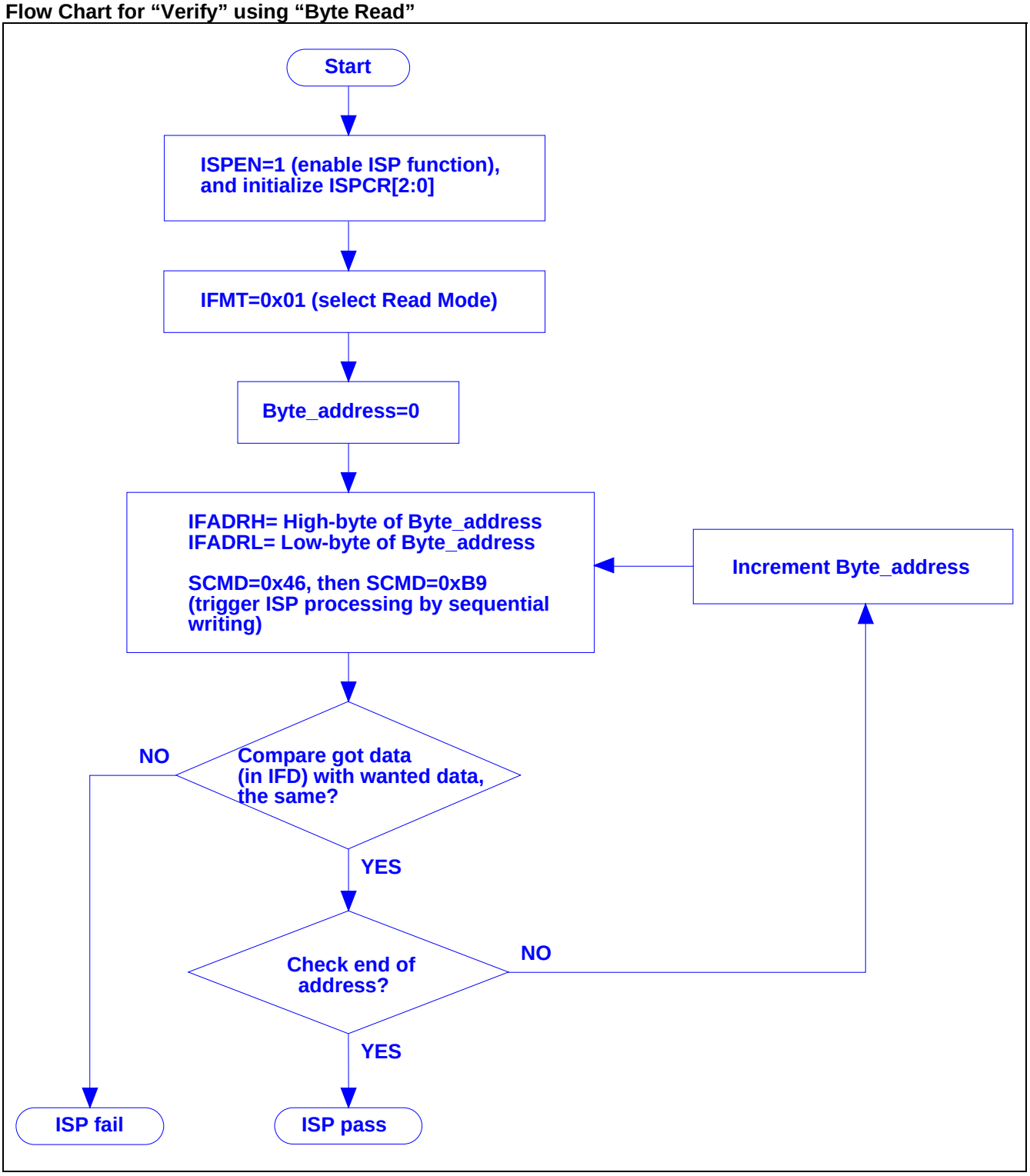
Flow Chart for “Page Erase”



Flow Chart for “Byte Program”



Flow Chart for “Verify” using “Byte Read”



11.3 Demo of the “ISP code”

```
;*************************************************************************************
; ISP modes
;*************************************************************************************

IFD      DATA   0E2h
IFADRH   DATA   0E3h
IFADRL   DATA   0E4h
IFMT     DATA   0E5h
SCMD     DATA   0E6h
ISPCR    DATA   0E7h
;
         MOV    ISPCR,#10000011b  ;ISPCR.7=1, enable ISP
                                  ;ISPCR[2:0]=011, for Fosc=11.0592MHz

;=========================================================================
; 1. Page Erase Mode (512 bytes per page)
;=========================================================================

         MOV    IFMT,#03h  ;select Page Erase Mode

         MOV    IFADRH,??  ;fill page address in IFADRH & IFADRL
         MOV    IFADRL,??  ;

         MOV    SCMD,#46h  ;trigger ISP processing
         MOV    SCMD,#0B9h ;

         ;Now in processing...(CPU will halt here until complete)

;=========================================================================
; 2. Byte Program Mode
;=========================================================================

         MOV    IFMT,#02h  ;select Byte Program Mode

         MOV    IFADRH,??  ;fill byte address in IFADRH & IFADRL
         MOV    IFADRL,??  ;
         MOV    IFD,??     ;fill the data to be programmed in IFD

         MOV    SCMD,#46h  ;trigger ISP processing
         MOV    SCMD,#0B9h ;

         ;Now in processing...(CPU will halt here until complete)

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

         MOV    IFMT,#01h  ;select Byte Read Mode

         MOV    IFADRH,??  ;fill byte address in IFADRH & IFADRL
         MOV    IFADRL,??  ;

         MOV    SCMD,#46h  ;trigger ISP processing
         MOV    SCMD,#0B9h ;

         ;Now in processing...(CPU will halt here until complete)

         MOV    A,IFD      ;data will be in IFD
         CJNE   A,wanted,ISP_error ;compare with the wanted value
         ...
         ...
ISP_error:
         ...
;
```

11.4 Note on In-System-Programming

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

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

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

12 In Application Programming (IAP)

(First, please refer to Section 1.3 for Flash Memory Configuration.)

The MPC82L(E)52/54 is *In Application Programmable* (IAP), which allows some region in the Flash memory to be used as non-volatile data storage while the application program is running. This useful feature can be applied to the application where the data must be kept after power off. Thus, there is no need to use an external serial EEPROM (such as 93C46, 24C01, .., and so on) for saving the non-volatile data.

12.1 IAP-memory Boundary/Range

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

Prior to using the IAP feature, users should configure an IAP-memory by programming a proper value to the option ***IAPLB[7:0]*** using a universal Writer or Programmer. The range of the IAP-memory depends on the contents of IAPLB, as listed below.

IAP lower boundary = IAPLB[7:0]x256, and

IAP higher boundary = ISP start address –1.

Where, the IAPLB must be an even number.

Section 12.3 lists all the possible ranges for IAP-memory versus different ISP size and programmed IAPLB.

12.2 Update the data in the IAP-memory

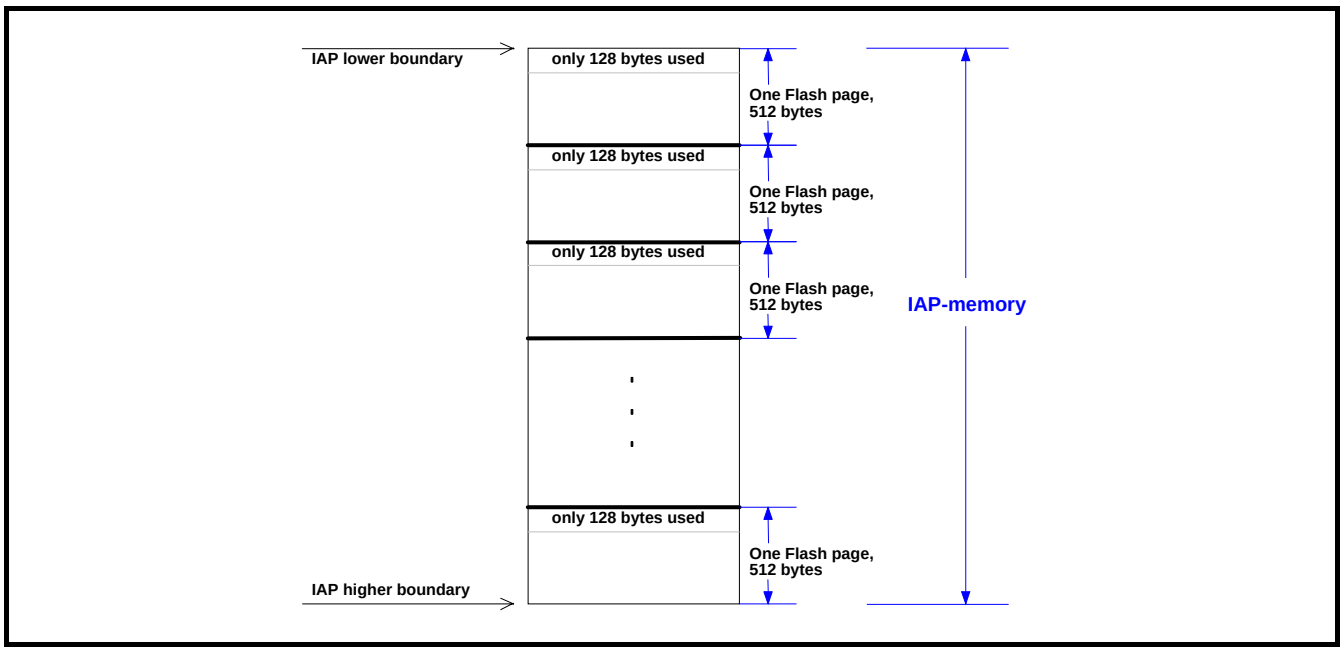
Because the IAP-memory is a part of Flash memory, only *Page Erase* is provided for Flash erasing. *To update "one byte" in the IAP-memory, users can not directly program the new datum into that byte*. The following steps show the correct updating procedure:

Step1) Save the whole page data (512 bytes) to a buffer (also with size of 512 bytes).
Step2) Erase this page (***using Page Erase mode of ISP***).
Step3) Update the wanted byte(s) in the buffer.
Step4) Program the updated data out of the buffer into this page (***using Byte Program mode of ISP***).

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

Users might ask: *Where is the buffer?*. The buffer comes from the internal RAM. The buffer size, however, might not be 512 bytes due to the limitation of RAM size. According to the available size of the internal RAM (which can be dedicated for the buffer), we can know how many bytes in each Flash page can be used as the non-volatile storage. That is, if 128 bytes can be dedicated for the buffer, then only 128 bytes in each page can be used, as the following Figure shows. (Of course, if only 64 bytes dedicated for the buffer, then 128 is replaced with 64.)

In detailed description, if the available buffer size is 128 bytes, and total 260 bytes of non-volatile storage are needed, the user should configure 1.5K bytes (3 Flash pages, because 260=128+128+4 ➔ 3 pages) of IAP-memory for this application.



12.3 IAP-memory vs. Settings of ISP-memory and IAPLB

As we have known, the IAP-memory and ISP-memory are user-configured. Table 12.3a ~ Table 12.3h show a variety of settings.

12.3.1 MPC82L(E)52 IAP-memory

Table 12.3a: No ISP

For MPC82L(E)52

Available **IAPLB**	AP-memory Range (Size)	IAP-memory	
		IAP_low_boundary	Range (Size)
0x02	0x0000 ~ 0x01FF (0.5KB)	0x0200	0x0200 ~ 0x1FFF (7.5KB)
0x04	0x0000 ~ 0x03FF (1.0KB)	0x0400	0x0400 ~ 0x1FFF (7.0KB)
0x06	0x0000 ~ 0x05FF (1.5KB)	0x0600	0x0600 ~ 0x1FFF (6.5KB)
0x08	0x0000 ~ 0x07FF (2.0KB)	0x0800	0x0800 ~ 0x1FFF (6.0KB)
0x0A	0x0000 ~ 0x09FF (2.5KB)	0x0A00	0x0A00 ~ 0x1FFF (5.5KB)
0x0C	0x0000 ~ 0x0BFF (3.0KB)	0x0C00	0x0C00 ~ 0x1FFF (5.0KB)
0x0E	0x0000 ~ 0x0DFF (3.5KB)	0x0E00	0x0E00 ~ 0x1FFF (4.5KB)
0x10	0x0000 ~ 0x0FFF (4.0KB)	0x1000	0x1000 ~ 0x1FFF (4.0KB)
0x12	0x0000 ~ 0x11FF (4.5KB)	0x1200	0x1200 ~ 0x1FFF (3.5KB)
0x14	0x0000 ~ 0x13FF (5.0KB)	0x1400	0x1400 ~ 0x1FFF (3.0KB)
0x16	0x0000 ~ 0x15FF (5.5KB)	0x1600	0x1600 ~ 0x1FFF (2.5KB)
0x18	0x0000 ~ 0x17FF (6.0KB)	0x1800	0x1800 ~ 0x1FFF (2.0KB)
0x1A	0x0000 ~ 0x19FF (6.5KB)	0x1A00	0x1A00 ~ 0x1FFF (1.5KB)
0x1C	0x0000 ~ 0x1BFF (7.0KB)	0x1C00	0x1C00 ~ 0x1FFF (1.0KB)
0x1E	0x0000 ~ 0x1DFF (7.5KB)	0x1E00	0x1E00 ~ 0x1FFF (0.5KB)
>= 0x20	0x0000 ~ 0x1FFF (8.0KB)	(NA)	(NA)

Table 12.3b: ISP=1KB (ISP start address=0x1C00)

For MPC82L(E)52

Available **IAPLB**	AP-memory Range (Size)	IAP-memory	
		IAP_low_boundary	Range (Size)
0x02	0x0000 ~ 0x01FF (0.5KB)	0x0200	0x0200 ~ 0x1BFF (6.5KB)
0x04	0x0000 ~ 0x03FF (1.0KB)	0x0400	0x0400 ~ 0x1BFF (6.0KB)
0x06	0x0000 ~ 0x05FF (1.5KB)	0x0600	0x0600 ~ 0x1BFF (5.5KB)
0x08	0x0000 ~ 0x07FF (2.0KB)	0x0800	0x0800 ~ 0x1BFF (5.0KB)
0x0A	0x0000 ~ 0x09FF (2.5KB)	0x0A00	0x0A00 ~ 0x1BFF (4.5KB)
0x0C	0x0000 ~ 0x0BFF (3.0KB)	0x0C00	0x0C00 ~ 0x1BFF (4.0KB)
0x0E	0x0000 ~ 0x0DFF (3.5KB)	0x0E00	0x0E00 ~ 0x1BFF (3.5KB)
0x10	0x0000 ~ 0x0FFF (4.0KB)	0x1000	0x1000 ~ 0x1BFF (3.0KB)
0x12	0x0000 ~ 0x11FF (4.5KB)	0x1200	0x1200 ~ 0x1BFF (2.5KB)
0x14	0x0000 ~ 0x13FF (5.0KB)	0x1400	0x1400 ~ 0x1BFF (2.0KB)
0x16	0x0000 ~ 0x15FF (5.5KB)	0x1600	0x1600 ~ 0x1BFF (1.5KB)
0x18	0x0000 ~ 0x17FF (6.0KB)	0x1800	0x1800 ~ 0x1BFF (1.0KB)
0x1A	0x0000 ~ 0x19FF (6.5KB)	0x1A00	0x1A00 ~ 0x1BFF (0.5KB)
>= 0x1C	0x0000 ~ 0x1BFF (7.0KB)	(NA)	(NA)

Table 12.3c: ISP=2KB (ISP start address=0x1800)

For MPC82L(E)52

Available **IAPLB**	AP-memory Range (Size)	IAP-memory	
		IAP_low_boundary	Range (Size)
0x02	0x0000 ~ 0x01FF (0.5KB)	0x0200	0x0200 ~ 0x17FF (5.5KB)
0x04	0x0000 ~ 0x03FF (1.0KB)	0x0400	0x0400 ~ 0x17FF (5.0KB)
0x06	0x0000 ~ 0x05FF (1.5KB)	0x0600	0x0600 ~ 0x17FF (4.5KB)
0x08	0x0000 ~ 0x07FF (2.0KB)	0x0800	0x0800 ~ 0x17FF (4.0KB)
0x0A	0x0000 ~ 0x09FF (2.5KB)	0x0A00	0x0A00 ~ 0x17FF (3.5KB)
0x0C	0x0000 ~ 0x0BFF (3.0KB)	0x0C00	0x0C00 ~ 0x17FF (3.0KB)
0x0E	0x0000 ~ 0x0DFF (3.5KB)	0x0E00	0x0E00 ~ 0x17FF (2.5KB)
0x10	0x0000 ~ 0x0FFF (4.0KB)	0x1000	0x1000 ~ 0x17FF (2.0KB)
0x12	0x0000 ~ 0x11FF (4.5KB)	0x1200	0x1200 ~ 0x17FF (1.5KB)
0x14	0x0000 ~ 0x13FF (5.0KB)	0x1400	0x1400 ~ 0x17FF (1.0KB)
0x16	0x0000 ~ 0x15FF (5.5KB)	0x1600	0x1600 ~ 0x17FF (0.5KB)
>= 0x18	0x0000 ~ 0x17FF (6.0KB)	(NA)	(NA)

Table 12.3d: ISP=3KB (ISP start address=0x1400)

For MPC82L(E)52

Available **IAPLB**	AP-memory Range (Size)	IAP-memory	
		IAP_low_boundary	Range (Size)
0x02	0x0000 ~ 0x01FF (0.5KB)	0x0200	0x0200 ~ 0x13FF (4.5KB)
0x04	0x0000 ~ 0x03FF (1.0KB)	0x0400	0x0400 ~ 0x13FF (4.0KB)
0x06	0x0000 ~ 0x05FF (1.5KB)	0x0600	0x0600 ~ 0x13FF (3.5KB)
0x08	0x0000 ~ 0x07FF (2.0KB)	0x0800	0x0800 ~ 0x13FF (3.0KB)
0x0A	0x0000 ~ 0x09FF (2.5KB)	0x0A00	0x0A00 ~ 0x13FF (2.5KB)
0x0C	0x0000 ~ 0x0BFF (3.0KB)	0x0C00	0x0C00 ~ 0x13FF (2.0KB)
0x0E	0x0000 ~ 0x0DFF (3.5KB)	0x0E00	0x0E00 ~ 0x13FF (1.5KB)
0x10	0x0000 ~ 0x0FFF (4.0KB)	0x1000	0x1000 ~ 0x13FF (1.0KB)
0x12	0x0000 ~ 0x11FF (4.5KB)	0x1200	0x1200 ~ 0x13FF (0.5KB)
>= 0x14	0x0000 ~ 0x13FF (5.0KB)	(NA)	(NA)

12.3.2 MPC82L(E)54 IAP-memory

Table 12.3e: No ISP

For MPC82L(E)54

Available **IAPLB**	AP-memory Range (Size)	IAP-memory	
		IAP_low_boundary	Range (Size)
0x02	0x0000 ~ 0x01FF (0.5KB)	0x0200	0x0200 ~ 0x3DFF (15.0KB)
0x04	0x0000 ~ 0x03FF (1.0KB)	0x0400	0x0400 ~ 0x3DFF (14.5KB)
0x06	0x0000 ~ 0x05FF (1.5KB)	0x0600	0x0600 ~ 0x3DFF (14.0KB)
0x08	0x0000 ~ 0x07FF (2.0KB)	0x0800	0x0800 ~ 0x3DFF (13.5KB)
0x0A	0x0000 ~ 0x09FF (2.5KB)	0x0A00	0x0A00 ~ 0x3DFF (13.0KB)
0x0C	0x0000 ~ 0x0BFF (3.0KB)	0x0C00	0x0C00 ~ 0x3DFF (12.5KB)
0x0E	0x0000 ~ 0x0DFF (3.5KB)	0x0E00	0x0E00 ~ 0x3DFF (12.0KB)
0x10	0x0000 ~ 0x0FFF (4.0KB)	0x1000	0x1000 ~ 0X3DFF (11.5KB)
0x12	0x0000 ~ 0x11FF (4.5KB)	0x1200	0x1200 ~ 0x3DFF (11.0KB)
0x14	0x0000 ~ 0x13FF (5.0KB)	0x1400	0x1400 ~ 0x3DFF (10.5KB)
0x16	0x0000 ~ 0x15FF (5.5KB)	0x1600	0x1600 ~ 0x3DFF (10.0KB)
0x18	0x0000 ~ 0x17FF (6.0KB)	0x1800	0x1800 ~ 0x3DFF (9.5KB)
0x1A	0x0000 ~ 0x19FF (6.5KB)	0x1A00	0x1A00 ~ 0x3DFF (9.0KB)
0x1C	0x0000 ~ 0x1BFF (7.0KB)	0x1C00	0x1C00 ~ 0x3DFF (8.5KB)
0x1E	0x0000 ~ 0x1DFF (7.5KB)	0x1E00	0x1E00 ~ 0x3DFF (8.0KB)
0x20	0x0000 ~ 0x1FFF (8.0KB)	0x2000	0x2000 ~ 0X3DFF (7.5KB)
0x22	0x0000 ~ 0x21FF (8.5KB)	0x2200	0x2200 ~ 0x3DFF (7.0KB)
0x24	0x0000 ~ 0x23FF (9.0KB)	0x2400	0x2400 ~ 0x3DFF (6.5KB)
0x26	0x0000 ~ 0x25FF (9.5KB)	0x2600	0x2600 ~ 0x3DFF (6.0KB)
0x28	0x0000 ~ 0x27FF (10.0KB)	0x2800	0x2800 ~ 0x3DFF (5.5KB)
0x2A	0x0000 ~ 0x29FF (10.5KB)	0x2A00	0x2A00 ~ 0x3DFF (5.0KB)
0x2C	0x0000 ~ 0x2BFF (11.0KB)	0x2C00	0x2C00 ~ 0x3DFF (4.5KB)
0x2E	0x0000 ~ 0x2DFF (11.5KB)	0x2E00	0x2E00 ~ 0x3DFF (4.0KB)
0x30	0x0000 ~ 0x2FFF (12.0KB)	0x3000	0x3000 ~ 0X3DFF (3.5KB)
0x32	0x0000 ~ 0x31FF (12.5KB)	0x3200	0x3200 ~ 0x3DFF (3.0KB)
0x34	0x0000 ~ 0x33FF (13.0KB)	0x3400	0x3400 ~ 0x3DFF (2.5KB)
0x36	0x0000 ~ 0x35FF (13.5KB)	0x3600	0x3600 ~ 0x3DFF (2.0KB)
0x38	0x0000 ~ 0x37FF (14.0KB)	0x3800	0x3800 ~ 0x3DFF (1.5KB)
0x3A	0x0000 ~ 0x39FF (14.5KB)	0x3A00	0x3A00 ~ 0x3DFF (1.0KB)
0x3C	0x0000 ~ 0x3BFF (15.0KB)	0x3C00	0x3C00 ~ 0x3DFF (0.5KB)
>= 0x3E	0x0000 ~ 0x3DFF (15.5KB)	(NA)	(NA)

Table 12.3f: ISP=1.5KB (ISP start address=0x3800)

For MPC82L(E)54

Available **IAPLB**	AP-memory Range (Size)	IAP-memory	
		IAP_low_boundary	Range (Size)
0x02	0x0000 ~ 0x01FF (0.5KB)	0x0200	0x0200 ~ 0x37FF (13.5KB)
0x04	0x0000 ~ 0x03FF (1.0KB)	0x0400	0x0400 ~ 0x37FF (13.0KB)
0x06	0x0000 ~ 0x05FF (1.5KB)	0x0600	0x0600 ~ 0x37FF (12.5KB)
0x08	0x0000 ~ 0x07FF (2.0KB)	0x0800	0x0800 ~ 0x37FF (12.0KB)
0x0A	0x0000 ~ 0x09FF (2.5KB)	0x0A00	0x0A00 ~ 0x37FF (11.5KB)
0x0C	0x0000 ~ 0x0BFF (3.0KB)	0x0C00	0x0C00 ~ 0x37FF (11.0KB)
0x0E	0x0000 ~ 0x0DFF (3.5KB)	0x0E00	0x0E00 ~ 0x37FF (10.5KB)
0x10	0x0000 ~ 0x0FFF (4.0KB)	0x1000	0x1000 ~ 0X37FF (10.0KB)
0x12	0x0000 ~ 0x11FF (4.5KB)	0x1200	0x1200 ~ 0x37FF (9.5KB)
0x14	0x0000 ~ 0x13FF (5.0KB)	0x1400	0x1400 ~ 0x37FF (9.0KB)
0x16	0x0000 ~ 0x15FF (5.5KB)	0x1600	0x1600 ~ 0x37FF (8.5KB)
0x18	0x0000 ~ 0x17FF (6.0KB)	0x1800	0x1800 ~ 0x37FF (8.0KB)
0x1A	0x0000 ~ 0x19FF (6.5KB)	0x1A00	0x1A00 ~ 0x37FF (7.5KB)
0x1C	0x0000 ~ 0x1BFF (7.0KB)	0x1C00	0x1C00 ~ 0x37FF (7.0KB)
0x1E	0x0000 ~ 0x1DFF (7.5KB)	0x1E00	0x1E00 ~ 0x37FF (6.5KB)
0x20	0x0000 ~ 0x1FFF (8.0KB)	0x2000	0x2000 ~ 0X37FF (6.0KB)
0x22	0x0000 ~ 0x21FF (8.5KB)	0x2200	0x2200 ~ 0x37FF (5.5KB)
0x24	0x0000 ~ 0x23FF (9.0KB)	0x2400	0x2400 ~ 0x37FF (5.0KB)
0x26	0x0000 ~ 0x25FF (9.5KB)	0x2600	0x2600 ~ 0x37FF (4.5KB)
0x28	0x0000 ~ 0x27FF (10.0KB)	0x2800	0x2800 ~ 0x37FF (4.0KB)
0x2A	0x0000 ~ 0x29FF (10.5KB)	0x2A00	0x2A00 ~ 0x37FF (3.5KB)
0x2C	0x0000 ~ 0x2BFF (11.0KB)	0x2C00	0x2C00 ~ 0x37FF (3.0KB)
0x2E	0x0000 ~ 0x2DFF (11.5KB)	0x2E00	0x2E00 ~ 0x37FF (2.5KB)
0x30	0x0000 ~ 0x2FFF (12.0KB)	0x3000	0x3000 ~ 0X37FF (2.0KB)
0x32	0x0000 ~ 0x31FF (12.5KB)	0x3200	0x3200 ~ 0x37FF (1.5KB)
0x34	0x0000 ~ 0x33FF (13.0KB)	0x3400	0x3400 ~ 0x37FF (1.0KB)
0x36	0x0000 ~ 0x35FF (13.5KB)	0x3600	0x3600 ~ 0x37FF (0.5KB)
>= 0x38	0x0000 ~ 0x37FF (14.0KB)	(NA)	(NA)

Table 12.3g: ISP=2.5KB (ISP start address=0x3400)

For MPC82L(E)54

Available **IAPLB**	AP-memory Range (Size)	IAP-memory	
		IAP_low_boundary	Range (Size)
0x02	0x0000 ~ 0x01FF (0.5KB)	0x0200	0x0200 ~ 0x33FF (12.5kB)
0x04	0x0000 ~ 0x03FF (1.0KB)	0x0400	0x0400 ~ 0x33FF (12.0KB)
0x06	0x0000 ~ 0x05FF (1.5KB)	0x0600	0x0600 ~ 0x33FF (11.5KB)
0x08	0x0000 ~ 0x07FF (2.0KB)	0x0800	0x0800 ~ 0x33FF (11.0KB)
0x0A	0x0000 ~ 0x09FF (2.5KB)	0x0A00	0x0A00 ~ 0x33FF (10.5KB)
0x0C	0x0000 ~ 0x0BFF (3.0KB)	0x0C00	0x0C00 ~ 0x33FF (10.0KB)
0x0E	0x0000 ~ 0x0DFF (3.5KB)	0x0E00	0x0E00 ~ 0x33FF (9.5KB)
0x10	0x0000 ~ 0x0FFF (4.0KB)	0x1000	0x1000 ~ 0X33FF (9.0KB)
0x12	0x0000 ~ 0x11FF (4.5KB)	0x1200	0x1200 ~ 0x33FF (8.5KB)
0x14	0x0000 ~ 0x13FF (5.0KB)	0x1400	0x1400 ~ 0x33FF (8.0KB)
0x16	0x0000 ~ 0x15FF (5.5KB)	0x1600	0x1600 ~ 0x33FF (7.5KB)
0x18	0x0000 ~ 0x17FF (6.0KB)	0x1800	0x1800 ~ 0x33FF (7.0KB)
0x1A	0x0000 ~ 0x19FF (6.5KB)	0x1A00	0x1A00 ~ 0x33FF (6.5KB)
0x1C	0x0000 ~ 0x1BFF (7.0KB)	0x1C00	0x1C00 ~ 0x33FF (6.0KB)
0x1E	0x0000 ~ 0x1DFF (7.5KB)	0x1E00	0x1E00 ~ 0x33FF (5.5KB)
0x20	0x0000 ~ 0x1FFF (8.0KB)	0x2000	0x2000 ~ 0X33FF (5.0KB)
0x22	0x0000 ~ 0x21FF (8.5KB)	0x2200	0x2200 ~ 0x33FF (4.5KB)
0x24	0x0000 ~ 0x23FF (9.0KB)	0x2400	0x2400 ~ 0x33FF (4.0KB)
0x26	0x0000 ~ 0x25FF (9.5KB)	0x2600	0x2600 ~ 0x33FF (3.5KB)
0x28	0x0000 ~ 0x27FF (10.0KB)	0x2800	0x2800 ~ 0x33FF (3.0KB)
0x2A	0x0000 ~ 0x29FF (10.5KB)	0x2A00	0x2A00 ~ 0x33FF (2.5KB)
0x2C	0x0000 ~ 0x2BFF (11.0KB)	0x2C00	0x2C00 ~ 0x33FF (2.0KB)
0x2E	0x0000 ~ 0x2DFF (11.5KB)	0x2E00	0x2E00 ~ 0x33FF (1.5KB)
0x30	0x0000 ~ 0x2FFF (12.0KB)	0x3000	0x3000 ~ 0X33FF (1.0KB)
0x32	0x0000 ~ 0x31FF (12.5KB)	0x3200	0x3200 ~ 0x33FF (0.5KB)
>= 0x34	0x0000 ~ 0x33FF (13.0KB)	(NA)	(NA)

Table 12.3h: ISP=3.5KB (ISP start address=0x3000)

For MPC82L(E)54

Available **IAPLB**	AP-memory Range (Size)	IAP-memory	
		IAP_low_boundary	Range (Size)
0x02	0x0000 ~ 0x01FF (0.5KB)	0x0200	0x0200 ~ 0x2FFF (11.5KB)
0x04	0x0000 ~ 0x03FF (1.0KB)	0x0400	0x0400 ~ 0x2FFF (11.0KB)
0x06	0x0000 ~ 0x05FF (1.5KB)	0x0600	0x0600 ~ 0x2FFF (10.5KB)
0x08	0x0000 ~ 0x07FF (2.0KB)	0x0800	0x0800 ~ 0x2FFF (10.0KB)
0x0A	0x0000 ~ 0x09FF (2.5KB)	0x0A00	0x0A00 ~ 0x2FFF (9.5KB)
0x0C	0x0000 ~ 0x0BFF (3.0KB)	0x0C00	0x0C00 ~ 0x2FFF (9.0KB)
0x0E	0x0000 ~ 0x0DFF (3.5KB)	0x0E00	0x0E00 ~ 0x2FFF (8.5KB)
0x10	0x0000 ~ 0x0FFF (4.0KB)	0x1000	0x1000 ~ 0X2FFF (8.0KB)
0x12	0x0000 ~ 0x11FF (4.5KB)	0x1200	0x1200 ~ 0x2FFF (7.5KB)
0x14	0x0000 ~ 0x13FF (5.0KB)	0x1400	0x1400 ~ 0x2FFF (7.0KB)
0x16	0x0000 ~ 0x15FF (5.5KB)	0x1600	0x1600 ~ 0x2FFF (6.5KB)
0x18	0x0000 ~ 0x17FF (6.0KB)	0x1800	0x1800 ~ 0x2FFF (6.0KB)
0x1A	0x0000 ~ 0x19FF (6.5KB)	0x1A00	0x1A00 ~ 0x2FFF (5.5KB)
0x1C	0x0000 ~ 0x1BFF (7.0KB)	0x1C00	0x1C00 ~ 0x2FFF (5.0KB)
0x1E	0x0000 ~ 0x1DFF (7.5KB)	0x1E00	0x1E00 ~ 0x2FFF (4.5KB)
0x20	0x0000 ~ 0x1FFF (8.0KB)	0x2000	0x2000 ~ 0X2FFF (4.0KB)
0x22	0x0000 ~ 0x21FF (8.5KB)	0x2200	0x2200 ~ 0x2FFF (3.5KB)
0x24	0x0000 ~ 0x23FF (9.0KB)	0x2400	0x2400 ~ 0x2FFF (3.0KB
0x26	0x0000 ~ 0x25FF (9.5KB)	0x2600	0x2600 ~ 0x2FFF (2.5KB)
0x28	0x0000 ~ 0x27FF (10.0KB)	0x2800	0x2800 ~ 0x2FFF (2.0KB)
0x2A	0x0000 ~ 0x29FF (10.5KB)	0x2A00	0x2A00 ~ 0x2FFF (1.5KB)
0x2C	0x0000 ~ 0x2BFF (11.0KB)	0x2C00	0x2C00 ~ 0x2FFF (1.0KB)
0x2E	0x0000 ~ 0x2DFF (11.5KB)	0x2E00	0x2E00 ~ 0x2FFF (0.5KB)
>= 0x30	0x0000 ~ 0x2FFF (12.0KB)	(NA)	(NA)

12.4 Note on In-Application-Programming

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

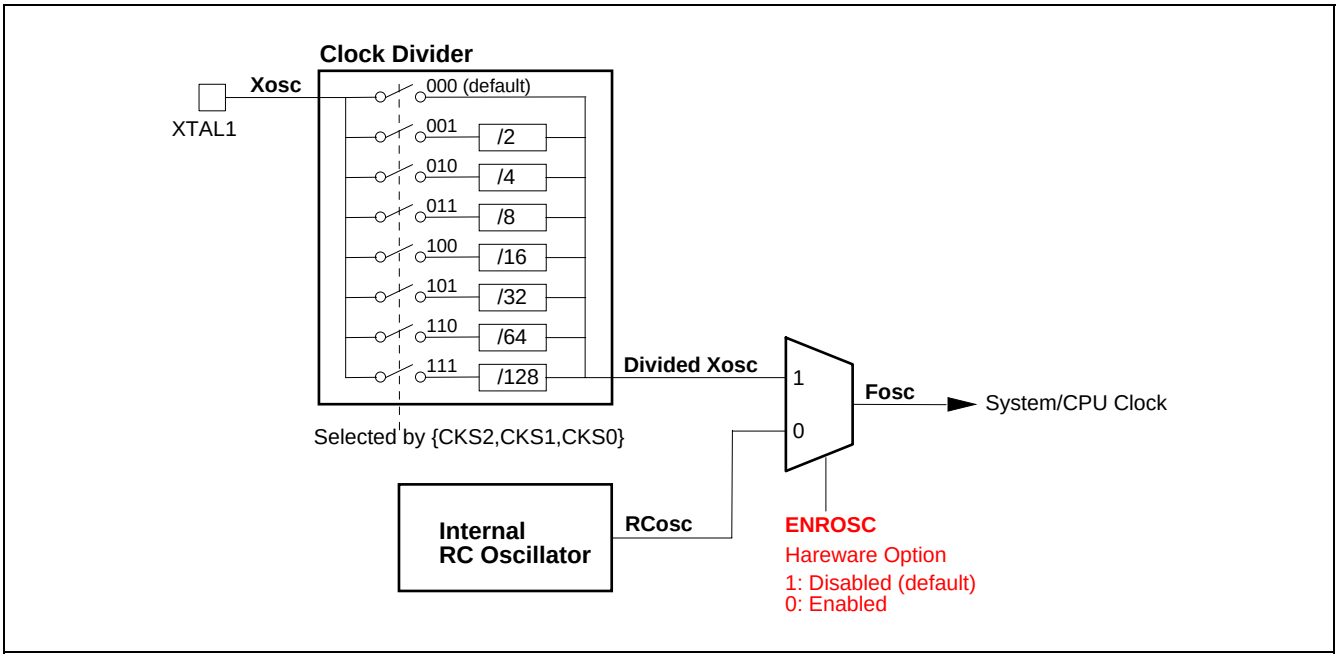
(1) Any interrupt can not be serviced in time during the CPU halts for IAP processing.

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

13 Programmable System Clock

The system clock (or CPU clock) of the device is programmable and source-selectable. The user can program the system clock frequency by PCON2 register, and select the clock source by the hardware option bit ENROSC, which can only be programmed by a universal Programmer/Writer. The following diagram shows the clock structure.

System Clock Structure



PCON2 (Power Control Register 2)

7	6	5	4	3	2	1	0
-	**-**	**-**	**-**	**-**	**CKS2**	**CKS1**	**CKS0**

CKS2, CKS1, CKS0: System clock select bits.

CKS2	CKS1	CKS0	Fosc (System clock)
0	0	0	Xosc (default)
0	0	1	Xosc/2
0	1	0	Xosc/4
0	1	1	Xosc/8
1	0	0	Xosc/16
1	0	1	Xosc/32
1	1	0	Xosc/64
1	1	1	Xosc/128

Where, Xosc is the frequency of the signal on XTAL1-pin.

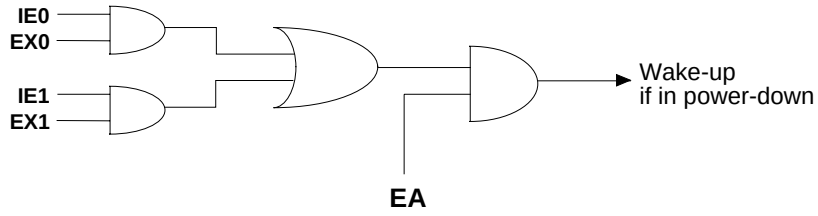
In the default state, the reset value of PCON2 is 0x00, so the system clock frequency is the same as that on the XTAL1-pin. The user can modify PCON2 at any time to get a new system clock, which will be active just after the modifying is completed.

This feature is especially useful for further power saving in idle mode as long as the user programs a non-zero value to the CKS[2:0] bits before entering the idle mode.

14 Wake-up from Power-down Mode

When the CPU is put into power-down mode, the external interrupts (/INT0 and /INT1) can be used to wake up the CPU if any of them is enabled. Once the CPU is wakened up, the interrupt service routine is serviced until the “RETI” instruction is encountered, and, the next instruction to be executed will be the one following the instruction that put the CPU into power-down mode. The following figure shows the power-down wake-up sources.

Power-down Wake-up Sources



Sample Code for Wake-up from Power-down (/INT0 is used in this example)

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

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

          SETB   INT0        ;pull high P3.2

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

          ORL    PCON,#02h   ;put MCU into power-down mode

          ;... Now, CPU is in power-down mode
          ;...

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

          NOP                ;! Note: here must be a NOP

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

15 Power-On Flag and Brown-Out Detection

15.1 Power-On Flag

The CPU will not start to work until the VCC power rises up to the Power-On Reset (POR) threshold voltage: 1.9V and 3.3V for MPC82L52/54 and MPC82E52/54, respectively. It means the POR state is activated whenever the VCC level is below the POR voltage.

The Power-On Flag, *POF bit (PCON.4)*, is set to "1" by the activated POR signal. Two occasions make the POR signal activated:

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

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

PCON (Power Control Register)

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

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

LVF: Low Voltage Flag.
Once brown-out condition is detected (VCC power decreases to than LVD voltage), it is set by hardware (and should be cleared by software).

15.2 Brown-Out Detection

Besides the POR voltage, there is a higher threshold voltage (LVD voltage, Low-Voltage-Detection voltage) for brown-out detection. The LVD voltage is 2.3V and 3.7V for MPC82L52/54 and MPC82E52/54, respectively. When the VCC power decreases to the LVD voltage, the Low-Voltage-Flag, *LVF bit (PCON.5)*, will be set by hardware. (*Note that during power-up, this flag will also be set, and the user should clear it by software for the following brown-out detecting.*) This flag can also generate an interrupt if bits *ENLVFI (AUXR.2)* and *EPCA_LVD (IE.6)* are both set to 1. (Refer to Section 9, Interrupt)

Further, once brown-out happens, the user can:

(1) enable the option ***ENLVR*** to let the CPU enter reset state, and/or

(2) enable the option ***LVFWP*** to inhibit any writing to the Flash memory.

Although the CPU can still work well between the ranges 1.9~2.3V (for MPC82L52/54) and 3.3~3.7V (for MPC82E52/54), they are not safe enough for the ISP/IAP operation (especially writing Flash memory). *So, we strongly recommend the user to use Low Voltage Detection combined with enabling the options ENLVR or LVFWP if the IAP feature is used in his application.*

16 Built-in Oscillator

The MPC82L(E)52/54 have a built-in oscillator with the rough oscillating frequency of 6MHz. It can be used to replace the external crystal oscillator in the application which doesn't need an exact oscillating frequency. To enable the built-in oscillator, users should enable the hardware option ***ENROSC*** by a universal Writer/Programmer.

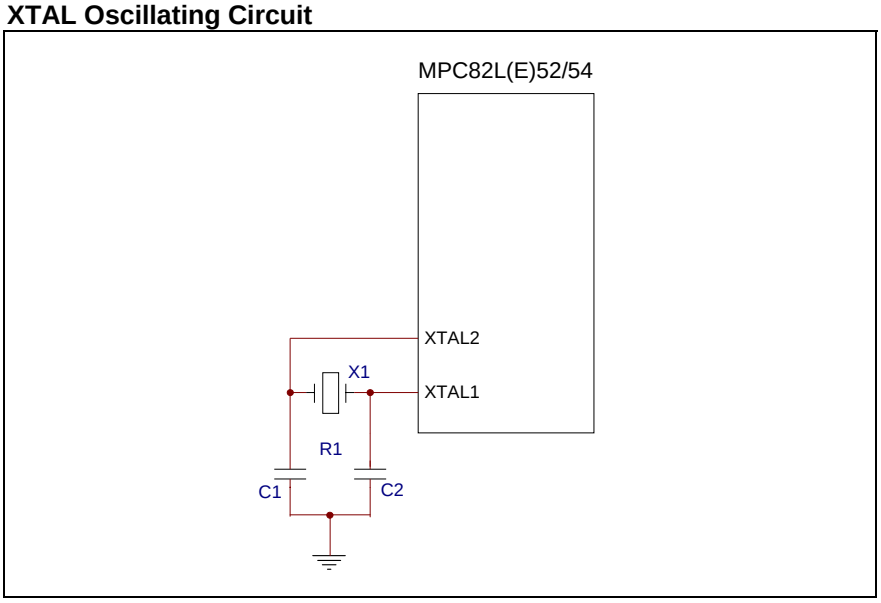
Typically, the oscillating frequency is about 6MHz at room temperature (25°C). And, the variation may be up to ± 30% over the temperature of –40°C to +85°C (+30% at –40°C, and –30% at +85°C). *So, it is only for the application which doesn't ask an exact oscillating frequency.*

17 XTAL Oscillating and Reset Circuitry

17.1 XTAL Oscillating

As shown in the XTAL Oscillating Circuit, to achieve successful and exact oscillating, the C1 and C2 (about 20pF~150pF) are necessary regardless of enabled or disabled OSCDN within the whole operation frequency range.

XTAL Oscillating Circuit

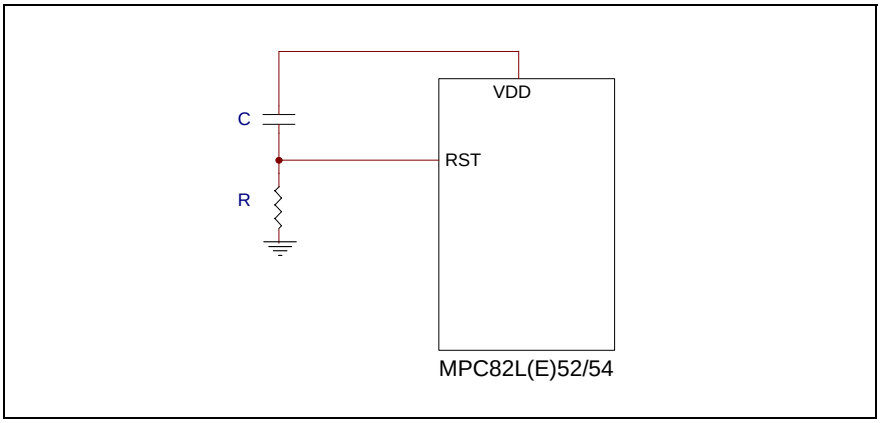


17.2 Reset Circuitry

When power is turned on, the circuit holds the RST pin high for an amount of time that depends on the capacitor value and the rate at which it charges. To ensure a valid reset, the RST pin must be held high long enough to allow the oscillator to start up plus two machine cycles. The following Table shows the best (R,C) combinations which are adequate to all different operating frequencies.

C	1uF	4.7uF	10uF
R	About 130K	About 27K	About 15K

Reset Circuit



18 Hardware Options

The MPC82L(E)52/54 have the following hardware options. All the options should be programmed by using a universal Writer or Programmer.

IAPLB[7:0]:
IAP Lower Boundary, which must be an even number. The boundary is determined by :

IAP lower boundary = [IAPLB] x 256.

LVFWP:
[enabled]: Enable LVFWP (Low-Voltage Flash-Write Protection) while IAP or ISP programming.
[disabled]: Disable LVFWP.

ENLVR:
[enabled]: Enable Low-Voltage Reset (LVR).
[disabled]: Disable LVR.

HWBS:
[enabled]: When power-up, MCU will boot from ISP-memory if ISP-memory is configured.
[disabled]: MCU always boots from AP-memory.

SB:
[enabled]: Code dumped on a universal Writer or Programmer is scrambled for security.
[disabled]: Not scrambled.

LOCK:
[enabled]: Code dumped & Device ID read on a universal Writer or Programmer is locked to 0xFF for security.
[disabled]: Not locked.

OSCDN:
[enabled]: Oscillating gain is reduced down for EMI reduction.
[disabled]: Normal gain.

HWBS2: (*only for MPC82L/E54*)
[enabled]: In addition to power-up, the reset from RST-pin will also force MCU to boot from ISP-memory if ISP-memory is configured.
[disabled]: Where MCU boots from is determined by **HWBS**.

ENROSC:
[enabled]: Enable built-in RC oscillator.
[disabled]: Disable built-in RC oscillator.

HWENW (accompanied with arguments **HWWIDL** and **HWPS[2:0]**):
[enabled]: Automatically enable Watch-dog Timer by hardware when MCU is powered up.

It means that:
In the WDTCR register, H/W will automatically
(1) set *ENW* bit,
(2) load **HWWIDL** into *WIDL* bit, and
(3) load **HWPS[2:0]** into *PS[2:0]* bits.

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

WDTCR (Watch-Dog-Timer Control Register)

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

[disabled]: No action on Watch-dog Timer when MCU powered up.

19 Instruction Set

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

The "*MOVX*" instructions, however, are omitted for MPC82L(E)52 and function-modified for MPC82L(E)54 for XRAM access.

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

Rn	Working register R0-R7 of the currently selected Register Bank.
direct	128 internal RAM locations, any I/O port, control or status register.
@Ri	Indirect internal RAM location addressed by register R0 or R1.
#data	8-bit constant included in instruction.
#data16	16-bit constant included in instruction.
addr16	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64K-byte program memory address space.
addr11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.
rel	Signed 8-bit offset byte. Used by SJMP and all conditional jumps. Range is –128 to +127 bytes relative to first byte of the following instruction.
bit	128 direct bit-addressable bits in internal RAM, any I/O pin, control or status bit.

19.1 Arithmetic Operations

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

19.2 Logic Operations

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

19.3 Data Transfer

Mnemonic	Description	Byte	Execution Clock Cycles
DATA TRANSFER			
MOV A,Rn	Move register to Acc	1	1
MOV A,direct	Move direct byte o Acc	2	2
MOV A,@Ri	Move indirect RAM to Acc	1	2
MOV A,#data	Move immediate data to Acc	2	2
MOV Rn,A	Move Acc to register	1	2
MOV Rn,direct	Move direct byte to register	2	4
MOV Rn,#data	Move immediate data to register	2	2
MOV direct,A	Move Acc to direct byte	2	3
MOV direct,Rn	Move register to direct byte	2	3
MOV direct,direct	Move direct byte to direct byte	3	4
MOV direct,@Ri	Move indirect RAM to direct byte	2	4
MOV direct,#data	Move immediate data to direct byte	3	3
MOV @Ri,A	Move Acc to indirect RAM	1	3
MOV @Ri,direct	Move direct byte to indirect RAM	2	3
MOV @Ri,#data	Move immediate data to indirect RAM	2	3
MOV DPTR,#data16	Load DPTR with a 16-bit constant	3	3
MOVC A,@A+DPTR	Move code byte relative to DPTR to Acc	1	4
MOVC A,@A+PC	Move code byte relative to PC to Acc	1	4
MOVX A,@Ri *Note*	Move external RAM (8-bit address) to Acc	1	3
MOVX A,@DPTR *Note*	Move external RAM (16-bit address) to Acc	1	3
MOVX @Ri,A *Note*	Move Acc to external RAM (8-bit address)	1	4
MOVX @DPTR,A *Note*	Move Acc to external RAM (16-bit address)	1	3
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

Note:
Only used by MPC82L(E)54 to access the on-chip expanded RAM (XRAM).

19.4 Boolean Variable Manipulation

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

19.5 Program and Machine Control

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