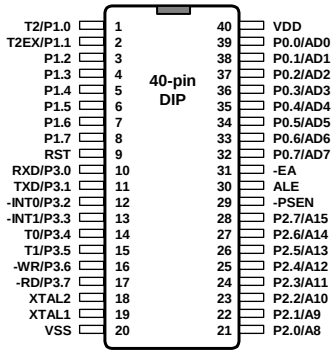
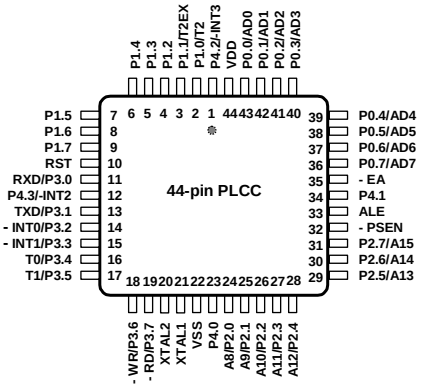
Contents

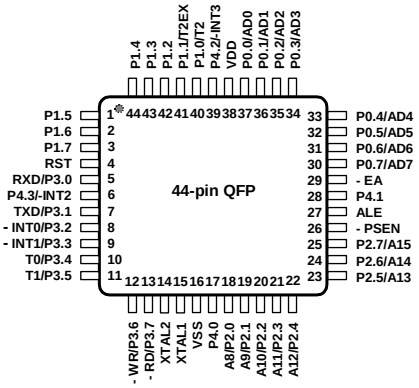
1. Pin Assignment



MPC89L51/52/53/54/58/515AE
MPC89E51/52/53/54/58/515AE



MPC89L51/52/53/54/58/515AP
MPC89E51/52/53/54/58/515AP



MPC89L51/52/53/54/58/515AF
MPC89E51/52/53/54/58/515AF

2. Special Function Registers

The Traditional 8052 SFRs

SYMBOL	DESCRIPTION	ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATE PORT FUNCTION 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	**96H** P1.6	**95H** P1.5	**94H** P1.4	**93H** P1.3	**92H** P1.2	**91H** P1.1 T2EX	**90H** P1.0 T2	FFH
P2*	Port 2	**A0H**	**A7H** P2.7	**A6H** P2.6	**A5H** P2.5	**A4H** P2.4	**A3H** P2.3	**A2H** P2.2	**A1H** P2.1	**A0H** P2.0	FFH
P3*	Port 3	**B0H**	**B7H** P3.7 RD	**B6H** P3.6 WR	**B5H** P3.5 T1	**B4H** P3.4 T0	**B3H** P3.3 /INT1	**B2H** P3.2 /INT0	**B1H** P3.1 TXD	**B0H** P3.0 RXD	FFH
IP*	Interrupt Priority	**B8H**	**BFH** -	**BEH** -	**BDH** PT2	**BCH** PS	**BBH** PT1	**BAH** PX1	**B9H** PT0	**B8H** PX0	x0000000B
IE*	Interrupt Enable	**A8H**	**AFH** EA	**AEH** -	**ADH** ET2	**ACH** ES	**ABH** ET1	**AAH** EX1	**A9H** ET0	**A8H** EX0	00H
TMOD	Timer Mode	**89H**	GATE	C/-T	M1	M0	GATE	C/-T	M1	M0	00H
TCON*	Timer Control	**88H**	**8FH** TF1	**8EH** TR1	**8DH** TF0	**8CH** TR0	**8BH** IE1	**8AH** IT1	**89H** IE0	**88H** IT0	00H
T2CON*	Timer 2 Control	**C8H**	**CFH** TF2	**CEH** EXF2	**CDH** RCLK	**CCH** TCLK	**CBH** EXEN2	**CAH** TR2	**C9H** C/-T2	**C8H** CP/-RL2	00H
TH0	Timer 1 High	**8CH**									00H
TL0	Timer 0 Low	**8AH**									00H
TH1	Timer 1 High	**8DH**									00H
TL1	Timer 0 Low	**8BH**									00H
TH2	Timer 2 High	**CDH**									00H
TL2	Timer 2 Low	**CCH**									00H
RCAP2H	Timer 2 Capture High	**CBH**									00H
RCAP2L	Timer 2 Capture Low	**CAH**									00H
SCON*+	Serial Port Control	**98H**	**9FH** SM0/FE	**9EH** SM1	**9DH** SM2	**9CH** REN	**9BH** TB8	**9AH** RB8	**99H** TI	**98H** RI	00H
SBUF	Serial Data Buffer	**99H**									xxxxxxxxB
PCON+	Power Control	**87H**	SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL	00xx0000B

Notes:

*: bit addressable

+: modified from the 8052 SFRs

-: reserved bit

The New-added SFRs for MPC89L51/52/53/54/58/515 & MPC89E51/52/53/54/58/515

SYMBOL	DESCRIPTION	ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATE PORT FUNCTION MSB							LSB	RESET VALUE
AUXR#	Auxiliary	**8EH**	-	-	-	-	-	-	EXTRAM	AO	xxxxxx00B
AUXR1#	Auxiliary 1	**A2H**	-	-	-	-	GF2	-	-	DPS	xxxxxxx0B
IPH#	Interrupt Priority High	**B7H**	-	PADCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	x0000000B
P4*#	Port 4	**E8H**	**EFH** -	**EEH** -	**EDH** -	**ECH** -	**EBH** P4.3	**EAH** P4.2	**E9H** P4.1	**E8H** P4.0	xxxx1111B
SADEN#	Slave Address Mask	**B9H**									00H
SADDR#	Slave Address	**A9H**									00H
T2MOD#	Timer 2 Mode Control	**C9H**	-	-	-	-	-	-	T2OE	DCEN	xxxxxx00B
XICON*#	External Interrupt Control	**C0H**	**C7H** PX3	**C6H** EX3	**C5H** IE3	**C4H** IT3	**C3H** PX2	**C2H** EX2	**C1H** IE2	**C0H** IT2	xxxxxxxxB
WDTCR#	Watch-dog Timer	**E1H**	-	-	ENW	CLRW	WIDL	PS2	PS1	PS0	xx000000B
ISPCR#	ISP Control Register	**E7H**	ISPEN	SWBS	SWRST	-	-	ICK2	ICK1	ICK0	00H
IFMT#	ISP Mode Table	**E5H**									xxxx0000B
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**									xxxxxxxxB

Notes:

*: bit addressable

#: new-added

-: reserved bit

3. Extended General Purpose I/O Port, P4

There is an extended ***bit-addressable*** GPIO port, P4 (P4.0~P4.3), for the 44-pin PLCC & QFP packages. Also, P4.2 and P4.3 have the alternate interrupt function, /INT3 and /INT2, respectively.

4. On-chip eXpanded RAM (XRAM)

In addition to the traditional 256 bytes of internal RAM, the MPC89L(E)51/52/53 has ***256 bytes*** of expanded RAM (XRAM), while the MPC89L(E)54/58/515 has ***1024 bytes***. Clear EXTRAM bit (in AUXR, bit 1) to enable XRAM accessing. User can use the instructions "MOVX @Ri" or "MOVX @DPTR" to access XRAM. While executing these instructions, the I/O status of P0, P2, P3.6 (/WR) and P3.7 (/RD) are not changed, and they still function as GPIO (General Purpose IO).

Set EXTRAM bit will disable XRAM accessing (but leave XRAM unchanged), and both "MOVX @Ri" and "MOVX @DPTR" will access the external data memory as the traditional 8051 does.

AUXR (Auxiliary register)

7	6	5	4	3	2	1	0
-	**-**	**-**	**-**	**-**	**-**	**EXTRAM**	**AO**

DEXTRAM: Set to enable external data memory accessing.
AO: Set to turn off ALE output.

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 MPC89L(E)51/../515 H/W can access them correctly. See the following descriptions, which is obtained from "***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 @Rn.

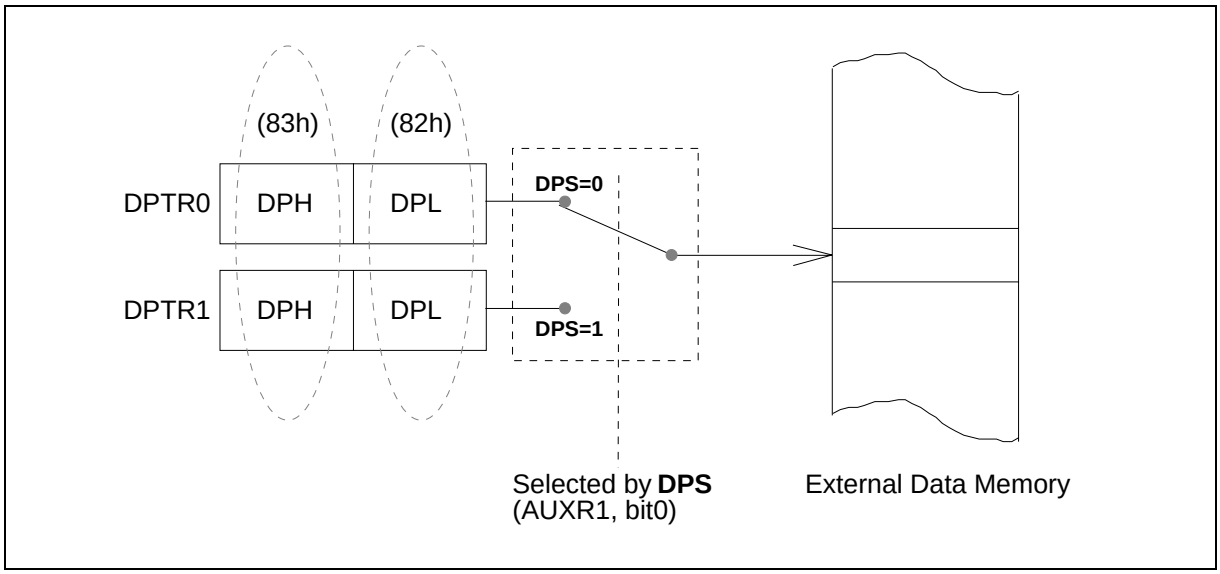
As with the **signed** and **unsigned** attributes, you may include memory type specifiers in the variable declaration.

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;
```

5. Second DPTR

The dual DPTR structure (see the following Figure) is a way by which the chip can specify the address of an external data memory location. There are two 16-bit DPTR registers that address the external memory, and a single bit called DPS (in AUXR1, bit 0) that allows the program code to switch between them.



DPTR Instructions

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

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

AUXR1 (Auxiliary 1 register)

7	6	5	4	3	2	1	0
-	**-**	**-**	**-**	**GF2**	**-**	**-**	**DPS**

GF2: General purpose flag.
DPS: DPTR selector, Clear/Set to select DPTR0/DPTR1, respectively.

6. Up/Down Counting of Timer 2

In the 16-bit auto-reload mode, Timer 2 can be configured as either a timer or a counter by C/T2 (T2CON.1), then be programmed to count up or down. The counting direction, up or down, is determined by bit DCEN (in T2MOD, bit 0) and pin T2EX (P1.1).

The reset value of DCEN bit is 0, which makes Timer 2 function as the standard 8052 (always count up). If **DCEN=1**, Timer 2 can count up or count down according to the logic level of the T2EX pin.

When a logic-1 is applied at T2EX pin, Timer 2 will count up.

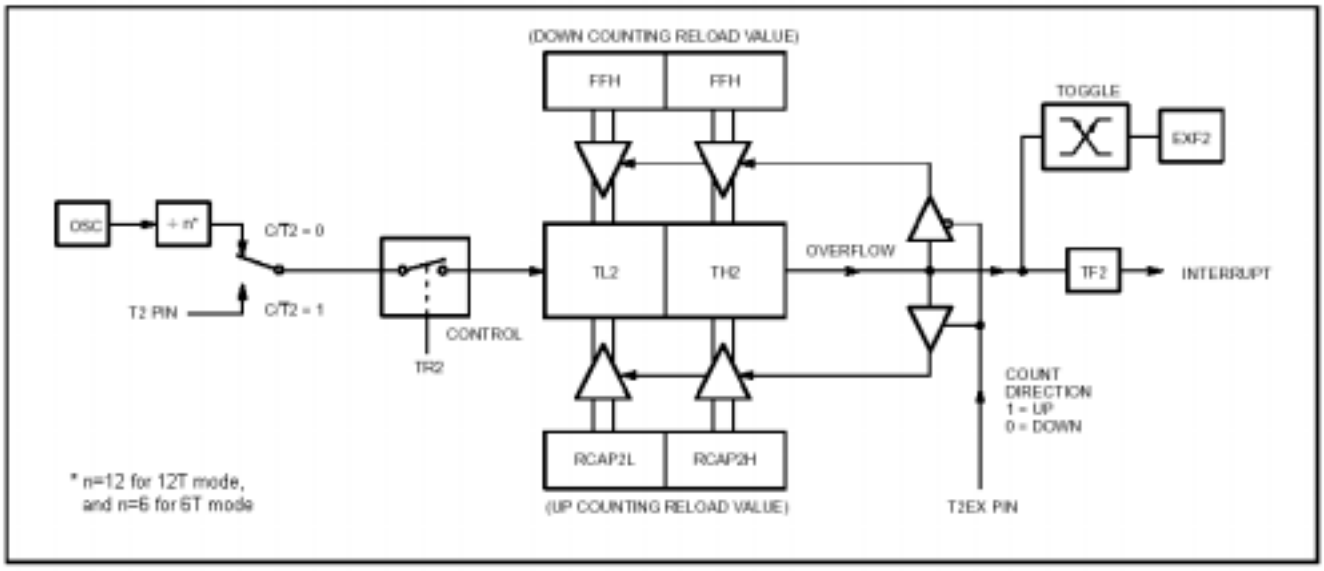
It overflows at the transition of [TH2,TL2] from #FFFFH to #0000H. At this time, TF2 flag (T2CON.7) is set by hardware and CPU will enter the interrupt service routine if the interrupt is previously enabled. Simultaneously, this overflow causes the 16-bit value of [RCAP2H,RCAP2H] to be reloaded into [TH2,TL2].

When a logic-0 is applied at T2EX pin, Timer 2 will count down.

It underflows when [TH2,TL2] becomes equal to [RCAP2H,RCAP2L]. At this time, TF2 flag is set by hardware and CPU will enter the interrupt service routine if the interrupt is previously enabled. Simultaneously, this underflow causes the value 0xFFFF to be reloaded into the [TH2,TL2].

The external flag EXF2 (T2CON.6) toggles when Timer 2 overflows or underflows. It can be used as the 17th bit of resolution if needed. The EXF2 flag does not cause an interrupt while DCEN=1.

Timer 2 in Auto-Reload Mode (with DCEN=1)



T2MOD (Timer 2 Mode Control register)

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

T2OE: Timer 2 Output Enable bit.
DCEN: Down Count Enable bit.

T2CON (Timer 2 Control Register)

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

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

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

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

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

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

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

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

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

7. Programmable Clock-Out

A 50% duty cycle clock can be programmed to come out on P1.0. Besides being a regular I/O pin, P1.0 has two alternate functions as shown below.

P1.0 Alternate Functions

Function	C/T2 (T2CON.1)	T2OE (in T2MOD, bit 1)
Timer 2 external clock input (T2)	1	0
50% duty cycle clock output	0	1

To configure the Timer/Counter2 as a clock generator, bit C/T2 (T2CON.1) must be cleared and bit T2OE (in T2MOD, bit 1) must be set. Of course, bit TR2 (T2CON.2) also must be set to start the timer.

The Clock-Out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L) as shown in this equation:

$$\frac{\text{Operating Frequency}}{n \times (65536 - [RCAP2H, RCAP2L])}$$

where,
n=4 for 12T mode
n=2 for 6T mode

In the Clock-Out mode, Timer 2 roll-overs will not generate an interrupt. This is similar to when it is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note that the baud-rate, however, and the clock-out frequency will be the same.

T2MOD (Timer 2 Mode Control Register)

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

T2OE: Timer 2 Output Enable bit.
DCEN: Down Count Enable bit.

8. Enhanced UART

8.1 Frame Error Detection

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

SCON (Serial Port Control Register)

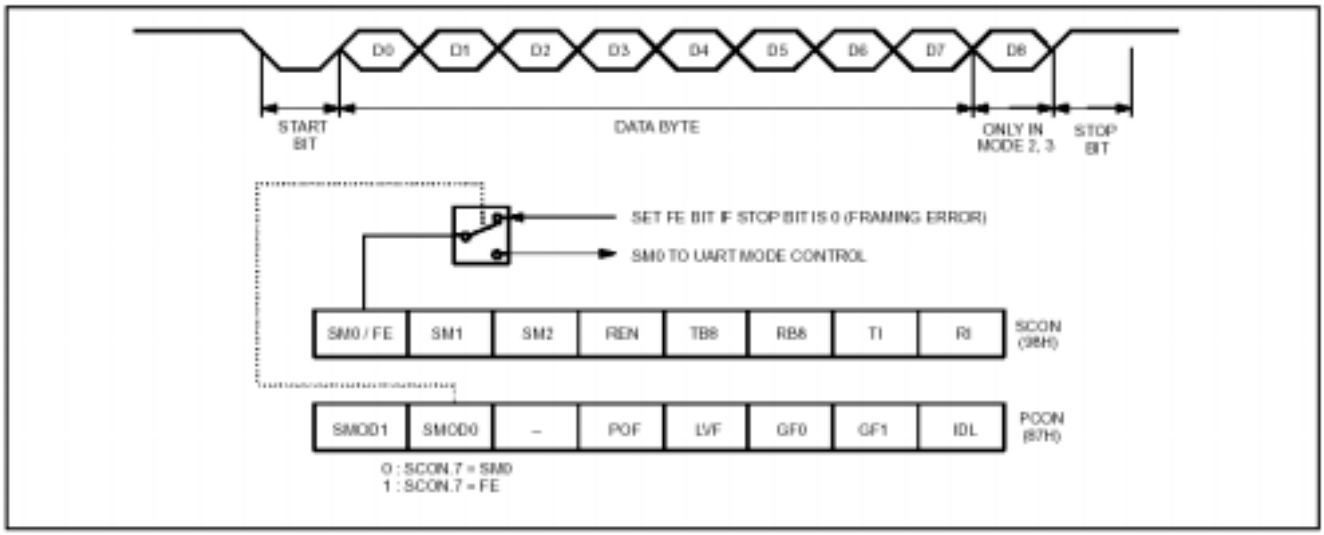
7	6	5	4	3	2	1	0
SM0/FE	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**

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

PCON (Power Control Register)

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

SMOD0: 0 to let SCON.7=SM0, and 1 to let SCON.7=FE.
POF: Power-ON Flag.



8.2 Automatic Address Recognition

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

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

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

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

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

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

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

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

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

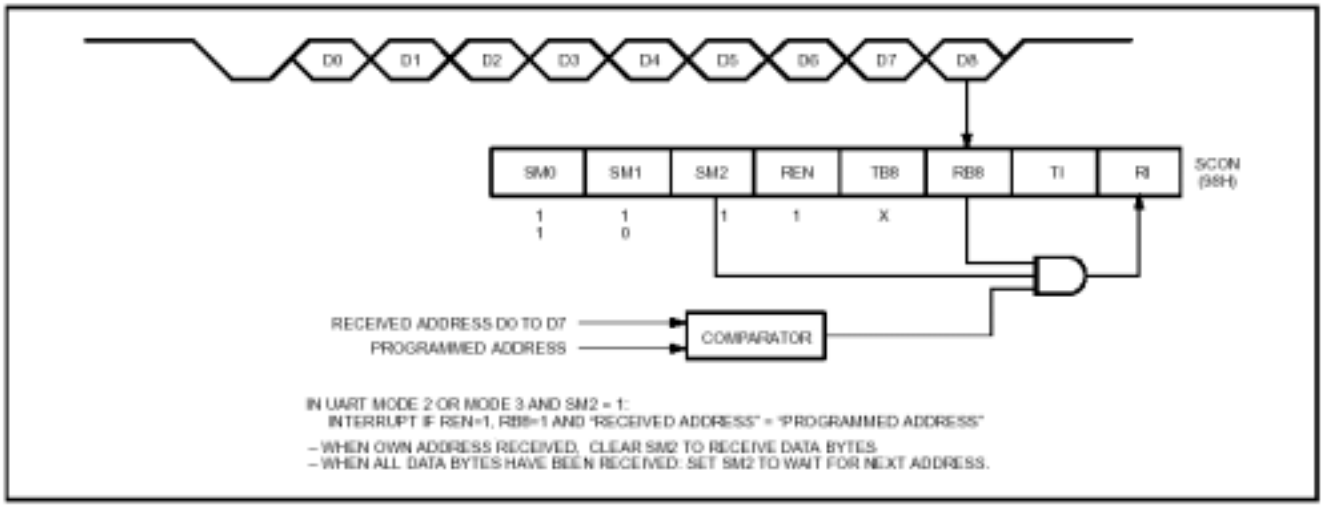
In a more complex system the following could be used to select slaves 1 and 2 while excluding slave 0:

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

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

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

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


D0
D1
D2
D3
D4
D5
D6
D7
D8
SM0
SM1
SM2
REN
TB8
RB8
TI
RI
SCON
(98H)
1
1
1
0
1
1
X
RECEIVED ADDRESS D0 TO D7
PROGRAMMED ADDRESS
COMPARATOR
IN UART MODE 2 OR MODE 3 AND SM2 = 1:
INTERRUPT IF REN=1, RB8=1 AND "RECEIVED ADDRESS" = "PROGRAMMED ADDRESS"
– WHEN OWN ADDRESS RECEIVED, CLEAR SM2 TO RECEIVE DATA BYTES.
– WHEN ALL DATA BYTES HAVE BEEN RECEIVED: SET SM2 TO WAIT FOR NEXT ADDRESS.

9. Eight Interrupt Sources and Four-Priority-Level Nested Structure

The MPC89L(E)51/52/53/54/58/515 have eight interrupt sources as shown in the following table.

Interrupt Source	Polling Priority	Interrupt Vector Address
IE0	(highest priority)	0003H
TF0	.	000BH
IE1	.	0013H
TF1	.	001BH
RI+TI	.	0023H
TF2+EXF2	.	002BH
IE2	.	0033H
IE3	(lowest priority)	003BH

IE (Interrupt Enable Register)

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

XICON (External Interrupt Control Register)

7	6	5	4	3	2	1	0
PX3	**EX3**	**IE3**	**IT3**	**PX2**	**EX2**	**IE2**	**IT2**

PX3: defines the priority level of /INT3 interrupt. If PX3=1, the /INT3 interrupt has the higher priority level.
EX3: enables or disables the /INT3 interrupt. If EX3=1, the /INT3 interrupt is enabled.
IE3: /INT3 interrupt edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT3: /INT3 type control bit. Set/cleared by software to specified falling-edge/low-level triggered.
PX2: defines the priority level of /INT2 interrupt. If PX2=1, the /INT2 interrupt has the higher priority level.
EX2: enables or disables the /INT2 interrupt. If EX2=1, the /INT2 interrupt is enabled.
IE2: /INT2 interrupt edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT2: /INT2 type control bit. Set/cleared by software to specified falling-edge/low-level triggered.

IP (Interrupt Priority Register)

7	6	5	4	3	2	1	0
-	**-**	**PT2**	**PS**	**PT1**	**PX1**	**PT0**	**PX0**

IPH (Interrupt Priority High Register)

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

The IPH register combined with IP register makes the four-level interrupt structure possible. The priority level of each interrupt is determined according to the following table.

IPH	IP	Priority Level
1	1	Level 3 (highest)
1	0	Level 2
0	1	Level 1
0	0	Level 0 (lowest)

For example, if (PT1H,PT1)=(1,0), then priority level of Timer 1 interrupt is 2, which is higher than level 1 with (PT1H,PT1)=(0,1) and level 0 with (PT1H,PT1)=(0,0).

10. Wake-up from Power-down by External Interrupt

Entering power-down mode, which is invoked by software, can save even more power. In power-down mode, the oscillator is stopped and the instruction that invokes power-down is the last instruction executed. And, the SFRs and on-chip RAM/XRAM retain their values.

Either hardware reset or ***external interru*****pt** *(/INT0, /INT1, /INT2 or /INT3) can be used to exit from power-down mode.* Hardware reset initializes all the SFRs but does not change the on-chip RAM/XRAM, while external interrupt allows both the SFRs and the on-chip RAM/XRAM to retain their values. To properly exit from power-down mode, *the external interrupt which is selected for waking up CPU must be enabled before power-down*. After power-down, trigger the selected external interrupt to wake up the CPU. Once the interrupt is serviced, the next instruction (after RETI) to be executed will be the one following the instruction that invoked the power-down mode. *Note, this instruction must be a "NOP".*

Sample code for *Wake-up-from-power-down* *(/INT0 is used in this code)*

```
;*******************************************************************************************
; 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
         SETB   EX0        ;enable INT0 interrupt

         ORL    PCON,#02h  ;invoke power-down

         ;... Now, CPU is in power-down mode
         ;... Wait for an external falling-edge on INT0-pin
         ;...

         NOP               ;! Note: here must be a NOP

wake_up:
         ;If INT0(P3.2) is triggered by a falling-edge,
         ; the MCU will wake up, and enter "IE0_isr",
         ;then return here to run continuously !
         ;...
         ;...
;
```

11. 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, a 8-bit prescaler and the WDT control register (WDTCR). The WDT is disabled at reset. To enable the WDT, user must set ENW bit in the WDTCR. When the WDT is enabled, it will increment every machine cycle (12 clocks in 12T mode, and 6 clocks in 6T mode), and ***there is no way to disable the WDT except through reset (either hardware reset, software reset or WDT reset)***.

When WDT is enabled, the user needs to service it by writing 1 to the CLRW bit to avoid WDT overflow. When the 15-bit counter overflows, the chip reset signal will reset the MCU.

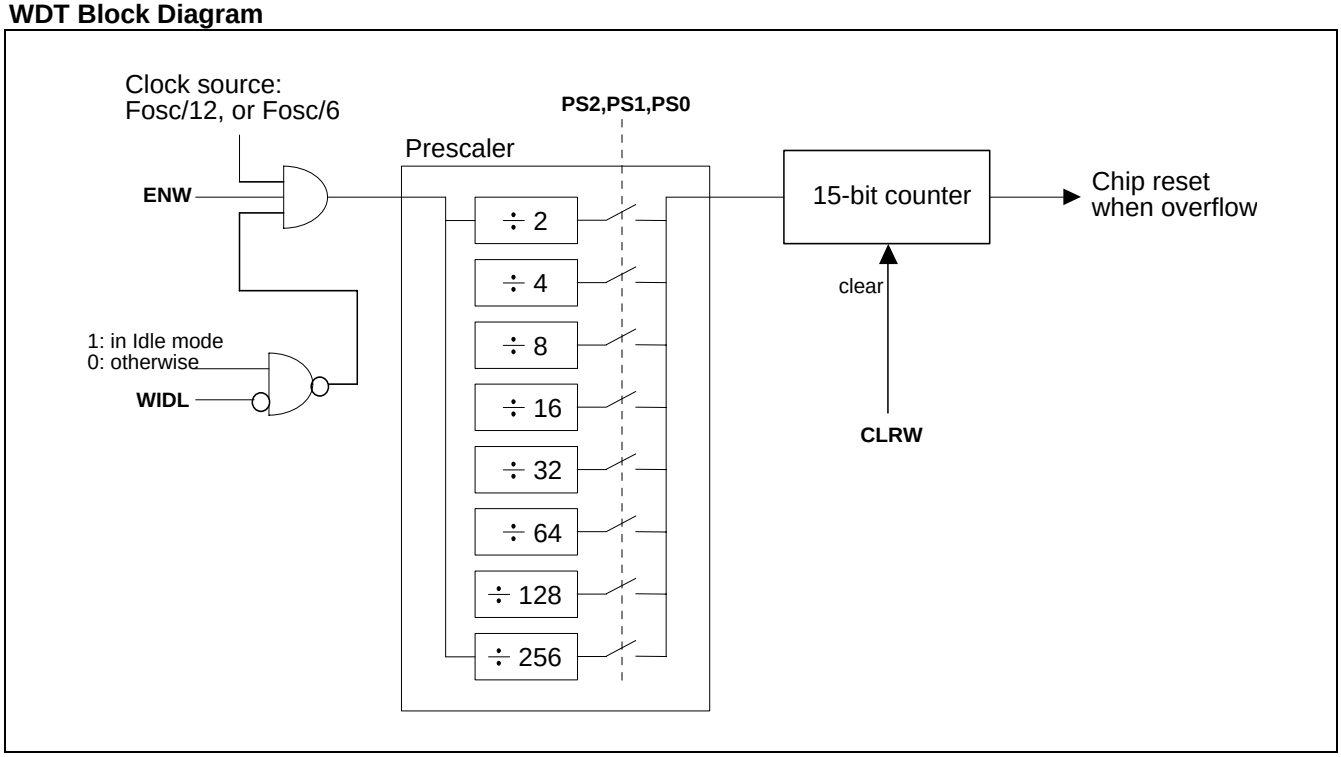
WDTCR (Watch-Dog-Timer Control Register)

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

Note: This is a Write-only register.

ENW: Set to enable WDT. *This bit can only be cleared by any kind of reset. Software can not clear it.*
CLRW: Write 1 to this bit to clear WDT. After WDT cleared, it is automatically cleared by H/W.
WIDL: Set this bit (!! Clear this bit for MPC89L516(556)X2) to let WDT keep counting while the MCU is in the Idle mode.
PS2~**PS1**: Used to determine the prescaler value.

WDT Block Diagram



WDT overflow period

The WDT overflow period is determined by the following equation:

(N x Prescaler x 2^{15}) / Fosc, where: N=12 for 12T mode, N=6 for 6T mode.

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

Example of WDT Overflow Period, @Fosc=12MHz

PS2	PS1	PS0	Prescaler value	12T mode	6T mode
0	0	0	2	65.536 ms	32.768 ms
0	0	1	4	131.072 ms	65.536 ms
0	1	0	8	262.144 ms	131.072 ms
0	1	1	16	524.288 ms	262.144 ms
1	0	0	32	1.048 s	524.288 ms
1	0	1	64	2.097 s	1.048 s
1	1	0	128	4.194 s	2.097 s
1	1	1	256	8.389 s	4.194 s

Sample code for Watchdog Timer

Condition: MCU runs at 12MHz and at 6T mode
Target: WDT Overflow Period = 524.288ms

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

          MOV    WDTCR_buf,#00h  ;clear buffer for WDTCR

          ANL    WDTCR_buf,#0F8h ;(PS2,PS1,PS0)=(1,0,0), prescaler=32
          ORL    WDTCR_buf,#04h  ;@12MHz/6T, WDT_Overflow_Period=524.288ms
          MOV    WDTCR,WDTCR_buf ;

          ORL    WDTCR_buf,#20h  ;enable WDT
          MOV    WDTCR,WDTCR_buf ;

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 ;
```

12. Low EMI Mode (inhibit ALE)

Set the AO bit (in AUXR, bit 0) will turn off the ALE output unless the CPU needs to access external data memory or run out of external program memory. This may reduce EMI (Electro Magnetic Interference) in some degree.

AUXR (Auxiliary Register)

7	6	5	4	3	2	1	0
-	**-**	**-**	**-**	**-**	**-**	**EXTRAM**	**AO**

EXTRAM: Set to enable external data memory accessing.
AO: Set to turn off ALE output.

13. 12T Mode and 6T Mode

The standard 8051 MCU can only run at 12T mode (each machine-cycle has only 12 clocks). While the MPC89L(E)51/52/53/54/58/515 can also run at 6T mode (each machine-cycle has only 6 clocks).
Its performance is twice that of 12T mode at the same Fosc.

To enable 6T mode, program 0 to EN6T bit in **OR1** by the WRITER.
Note: After whole-chip erasing (including Option Registers), this bit becomes 1, so the CPU will run at 12T mode.

Timing Calculation

While the MCU runs at ***6T-mode*** and at ***Fosc***, the user may regard it as ***12T-mode*** and at ***Fosc x 2***. So, all MCU's timing (including internal and external) calculation can be based on ***Fosc x 2***.

For example,

Suppose that the MCU runs at 6T-mode and Fosc=18MHz, we can regard it as a standard 8051 running at 36MHz. So, we will get

(1) One machine cycle has 1/36MHz X12 = 0.33us

(2) ALE output frequency = 36MHz/6 = 6MHz

(3) External program/data accessing timing, I/O timing and internal Timer/counter timing are calculated on the base of 36MHz.

14. In-System-Programming (ISP)

This MCU support the In-System-Programming (ISP) function. ISP allows the MCU to alter its program memory, in the actual end product, under software control. This function opens up a range of applications that need the ability to field update the application firmware.

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 0x46h then 0xB9h to trigger ISP operation)

ISPCR (ISP Control Register)

7	6	5	4	3	2	1	0
ISPEN	**SWBS**	**SWRST**	**-**	**-**	**ICK2**	**ICK1**	**ICK0**

ISPEN: Set to 1 to enable ISP function.

SWBS: Software boot select. Set/Cleared to select ISP-memory/AP-memory to boot from after reset.

SWRST: Write 1 to trigger S/W reset.

ICK2-0: Set programming time of ISP. See the following table for their setting in several Fosc ranges.

ISP Programming Time Setting

Fosc	ICK2	ICK1	ICK0
20~48 MHz	0	0	0
10~20 MHz	0	0	1
5~10 MHz	0	1	0
0~5 MHz	0	1	1

IFMT: ISP mode select register

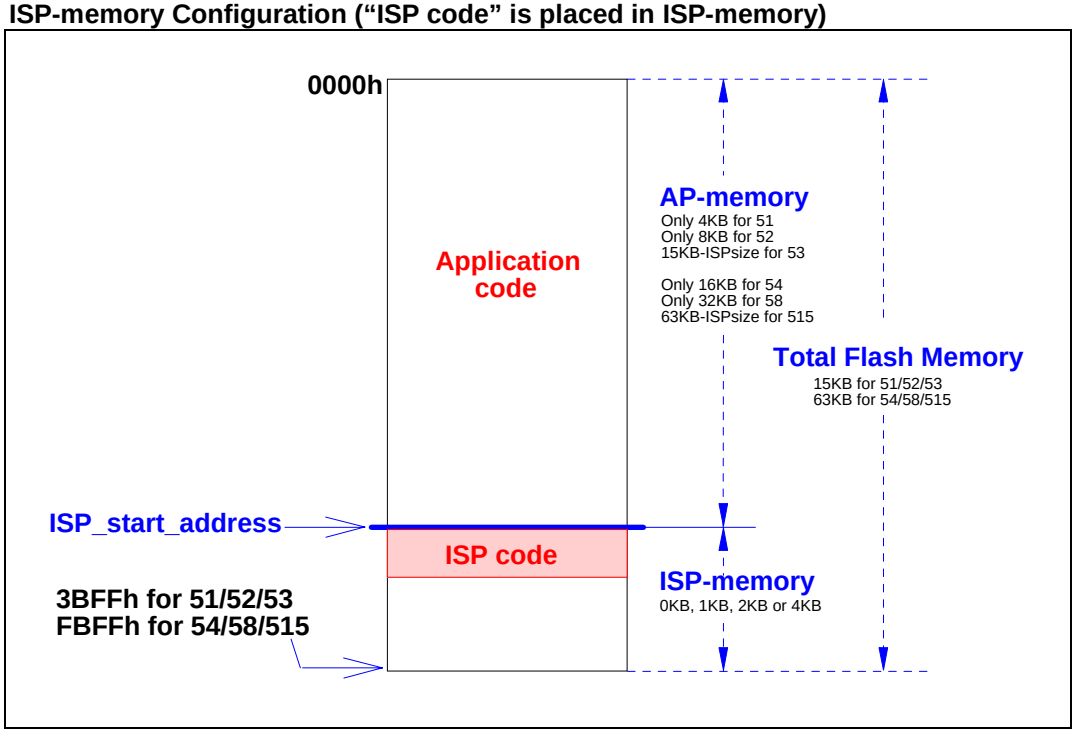
ISP Mode Select Table

IFMT value	ISP Mode
0	Standby
1	Read
2	Byte Program
3	Page Erase
5	OR1 Read (with IFADRL=1)
6	OR1 Program (with IFADRL=1)
7	OR1 Erase (with IFADRL=1)

14-1 How to boot rom ISP-memory to run “ISP code”

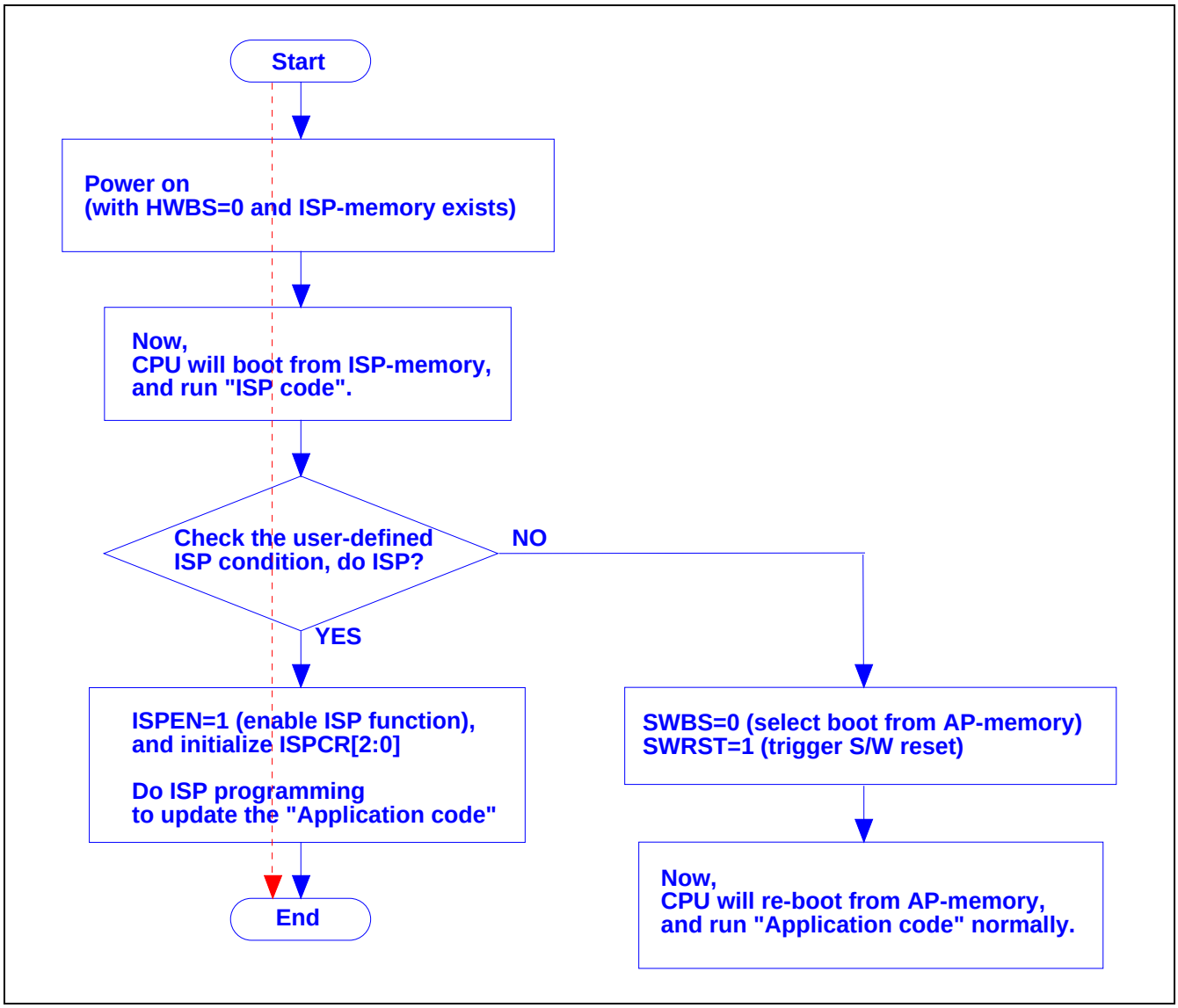
As shown in the following figure, the ISP-memory has been configured. To run the “ISP code”, the CPU should boot from the ISP-memory. Two methods can be used to boot from the ISP-memory.

ISP-memory Configuration (“ISP code” is placed in ISP-memory)

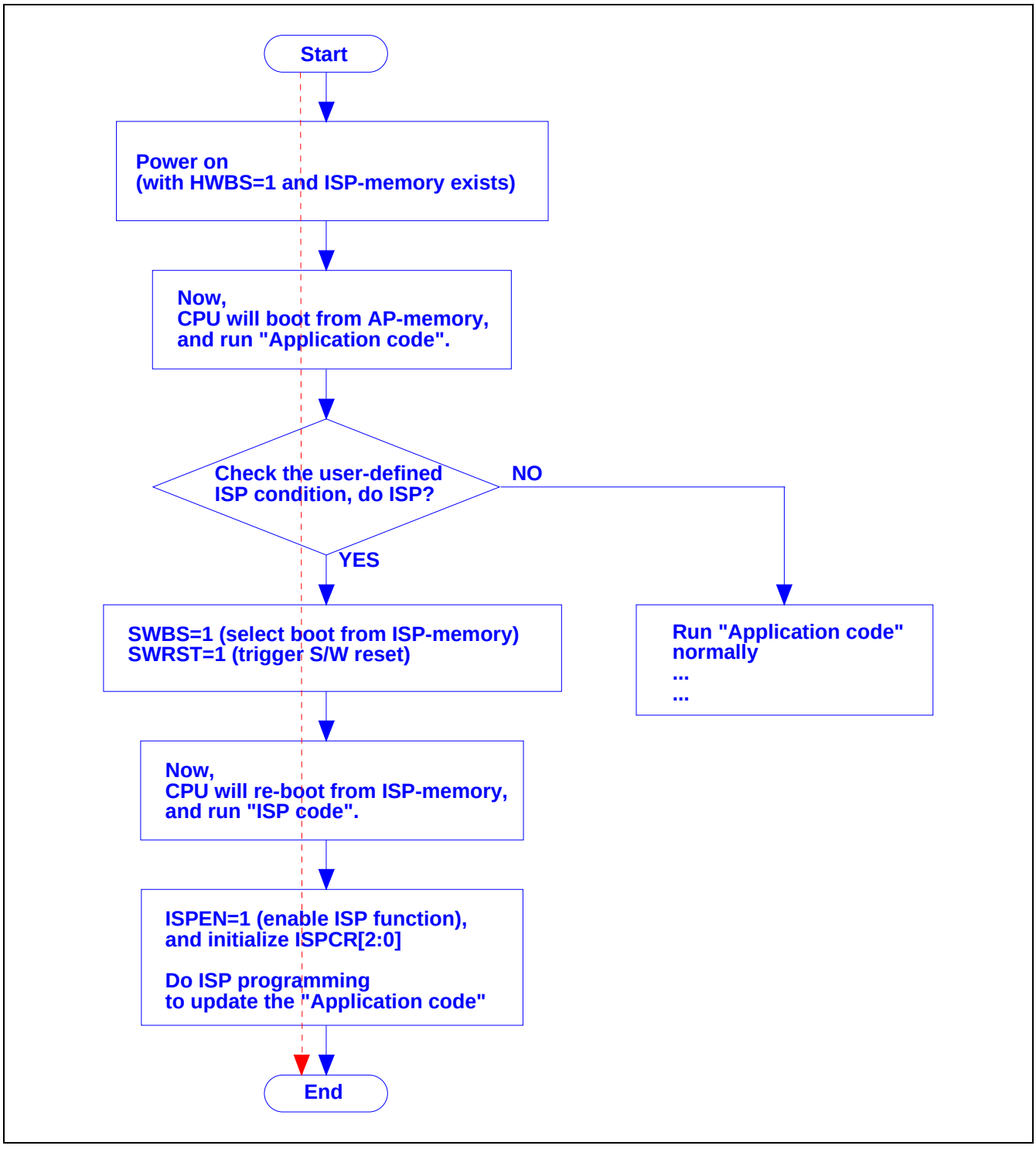


Notes:
Use a general Programmer (ex., Hi-Lo All-11) to configure the ISP-memory and program the “ISP code” into the ISP-memory.

Method 1: Directly Boot from ISP-memory (while HWBS=0)



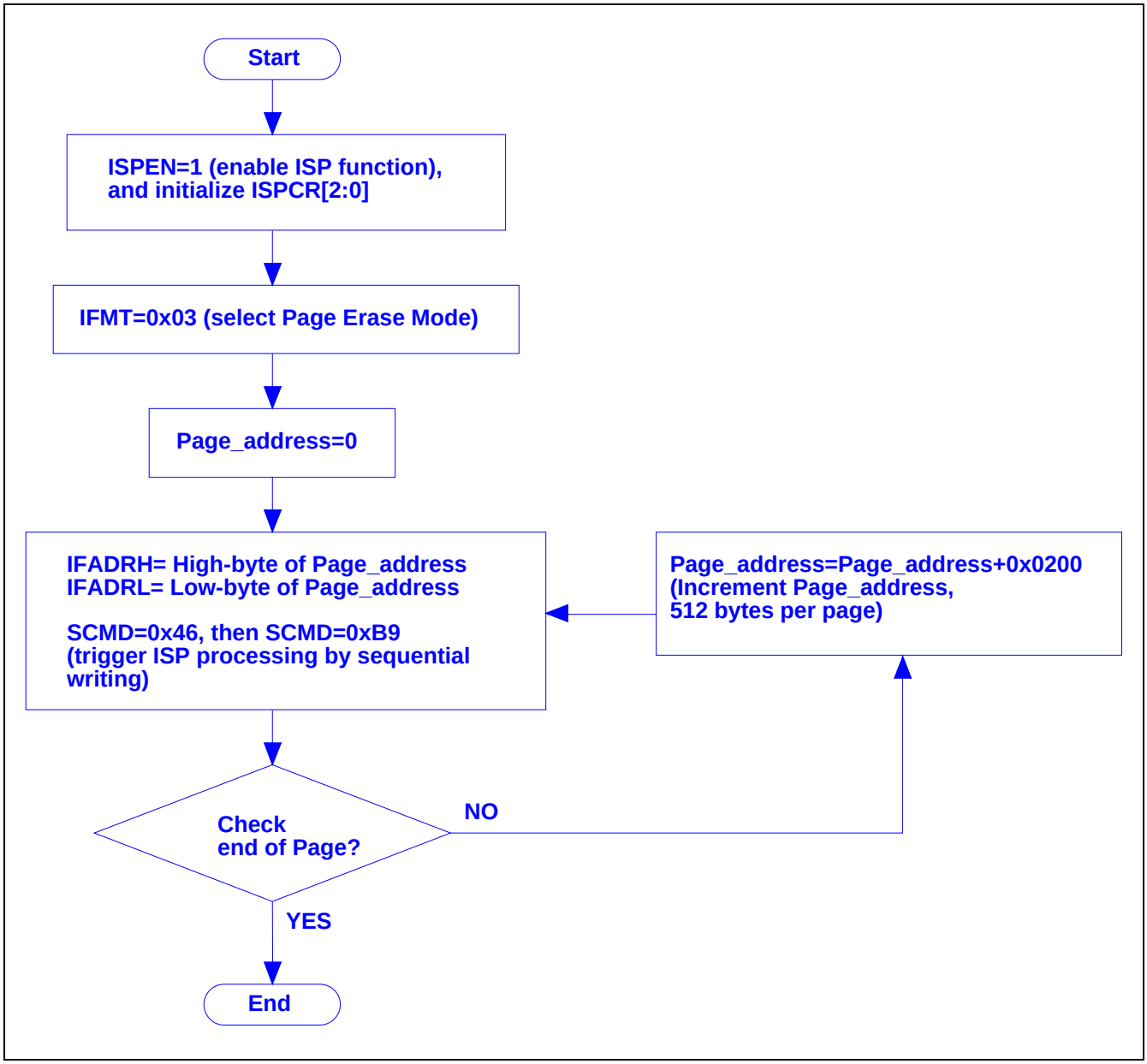
Method 2: Boot from ISP-memory through AP-memory (while HWBS=1)



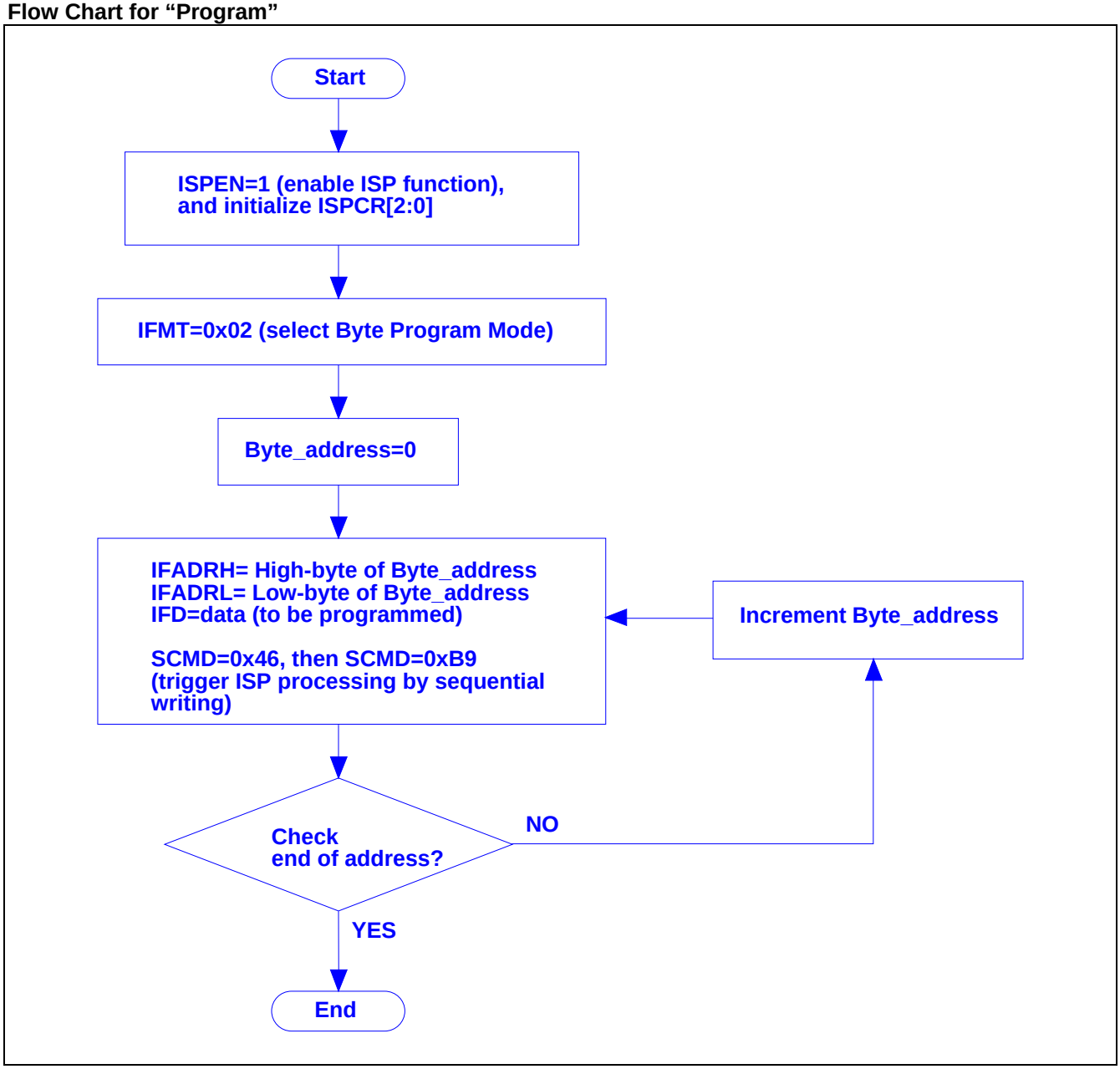
14-2 Operation Flow of ISP

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

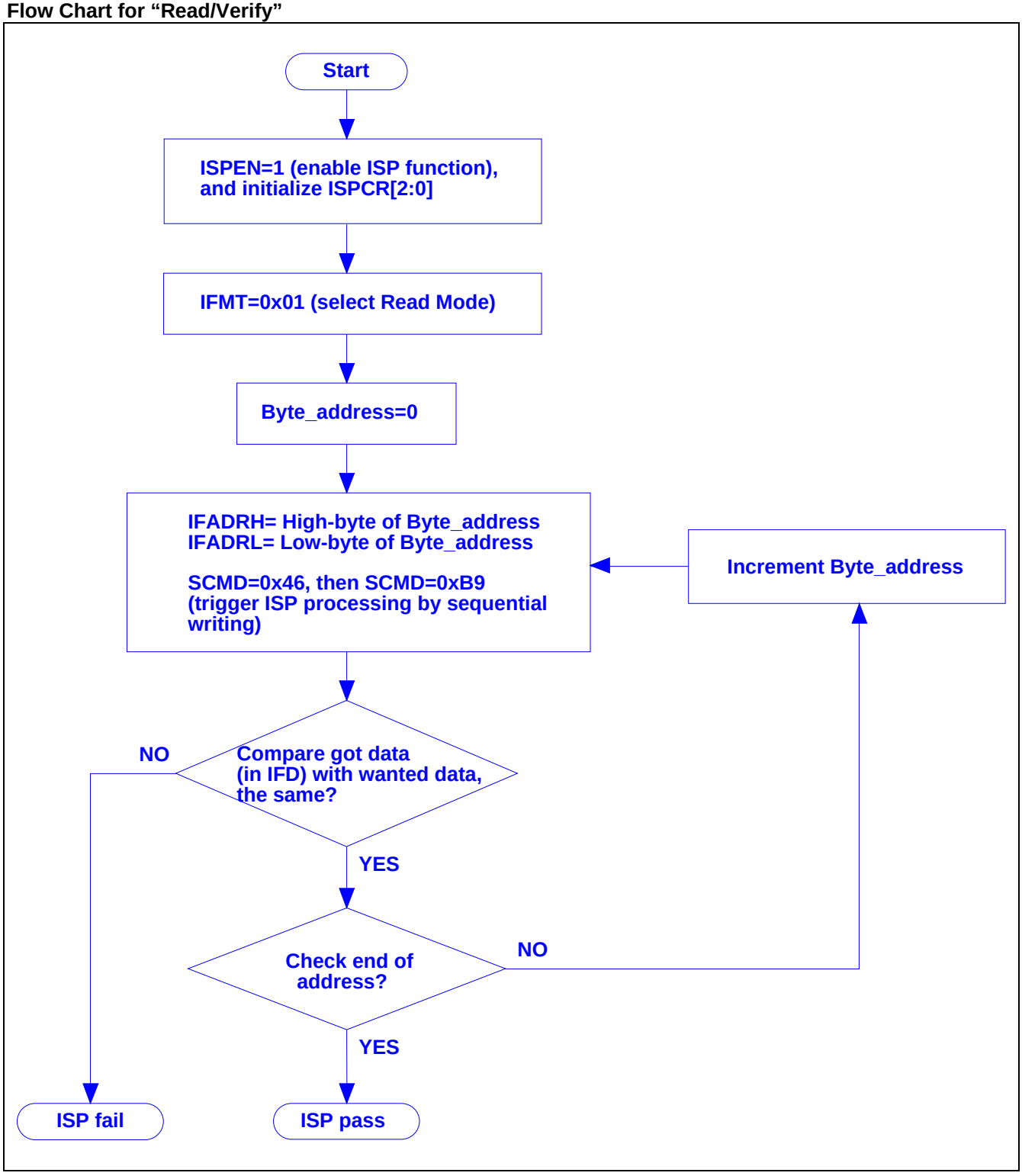
Flow Chart for “Erase”



Flow Chart for “Program”



Flow Chart for "Read/Verify"



14-3 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,#10000001b  ;ISPCR.7=1, enable ISP
                                  ;ISPCR[2:0]=001, for Fosc=20MHz

;=========================================================================
; 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. Read Mode
;=========================================================================

         MOV    IFMT,#01h  ;select 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
;
```

15. Non-volatile Data using IAP-memory

Refer to ***Section 18. Flash Memory Configuratiion***, there exists an *IAP-memory* (In-Application-Programming memory) in the MPC89L(E)51/52/54/58, excluding 53 and 515, for non-volatile data storage. Prior to using the IAP-memory, the ISP-memory must be configured as 1K, 2K or 4K Bytes. It is because the IAP-memory needs an upper boundary (equal to the ISP's lower boundary minus 1) to prevent from user's careless excess erasing/writing to the ISP-memory region, even the user doesn't use the ISP feature. The range and size of the IAP-memory depend on the ISP-memory size, as listed below.

Part No.	**IAP-memory range and size**		
	ISP_size=4KB	ISP_size =2KB	ISP_size =1KB
MPC89L(E)51	**0x1000~0x2BFF (7KB)**	**0x1000~0x33FF (9KB)**	**0x1000~0x37FF (10KB)**
MPC89L(E)52	**0x2000~0x2BFF (3KB)**	**0x2000~0x33FF (5KB)**	**0x2000~0x37FF (6KB)**
MPC89L(E)54	**0x4000~0xEBFF (43KB)**	**0x4000~0xF3FF (45KB)**	**0x4000~0xF7FF (46KB)**
MPC89L(E)58	**0x8000~0xEBFF (27KB)**	**0x8000~0xF3FF (29KB)**	**0x8000~0xF7FF (30KB)**

Any Flash page (with 512 bytes) in the IAP-memory can be erased, programmed and read by using the ISP operation, as described in Section 14-2. This useful feature can be applied to the application where data must be kept after power off. Thus, there is no need to use an external EEPROM for saving non-volatile data.

To update the non-volatile data, user should take the following steps:

Step1) Save the whole non-volatile data page (512 bytes) to a buffer (with size of 512 bytes).
Step2) Update some specific bytes in the buffer.
Step3) Erase this non-volatile data page (using ISP's Page Erase mode).
Step4) Program the updated data out of the buffer to this page (using ISP's Byte Program mode).

For MPC89L(E)51/52, there is not enough internal RAM for this buffer except using external RAM. We recommend users to use the on-chip 256 bytes of XRAM as the buffer. Of course, there should be no user data saved there. So, one 512-byte Flash page can save only 256 bytes of non-volatile data due to the limitation of buffer size (256 bytes). If more non-volatile data are needed, use another Flash page to save one more 256 bytes.

For MPC89L(E)54/58, there is enough internal RAM for this buffer while the on-chip 1024 bytes of XRAM are used (only 512 bytes are needed). Of course, there should be no user data saved there. So, one 512-byte Flash page can save 512 bytes of non-volatile data. If more non-volatile data are needed, use another Flash page to save one more 512 bytes.

As we have known, the IAP-memory is also Flash, the VDD power must be greater than 2.4V for L-series and 4.5V for E-series for operation safety.

16. Power-ON Flag

The Power-On Flag (POF) is set by hardware when the VDD level rises from 0. The POF bit can be set or cleared by software and thus allows user to determine if the start of CPU is ***cool*** (from power-on) or ***warm*** (from hardware reset or Watchdog Timer reset).

PCON (Power Control Register)

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

SMOD0: Cleared for SCON.7=SM0, and set for SCON.7=FE.
POF: Power-ON Flag.

17. Option Registers

Note: The programming value of the Option Registers won't become effective until next power-on.

OR0 (Option Register 0)

7	6	5	4	3	2	1	0
(reserved)	(reserved)	**ISPAS1**	**ISPAS0**	(reserved)	**MOVCL**	**SB**	**LOCK**

Note: The reserved bit must be kept 1.

(**ISPAS1,ISPAS0**): used for ISP-memory configuration, see the following table.

ISP-memory Setting Table

(**ISPAS1, ISPAS0**)	ISP-memory size	MPC89L51/52/53 MPC89E51/52/53		MPC89L54/58/515 MPC89E54/58/515	
		ISP-memory start ~ end	AP-memory start ~ end	ISP-memory start ~ end	AP-memory start ~ end
(0, 0)	**4KB**	**0x2C00~0x3BFF**	51: 0x0000~0x0FFF 52: 0x0000~0x1FFF 53: 0x0000~**0x2BFF**	**0xEC00~0xFBFF**	54: 0x0000~0x3FFF 58: 0x0000~0x7FFF 515: 0x0000~**0xEBFF**
(0, 1)	**2KB**	**0x3400~0x3BFF**	51: 0x0000~0x0FFF 52: 0x0000~0x1FFF 53: 0x0000~**0x33FF**	**0xF400~0xFBFF**	54: 0x0000~0x3FFF 58: 0x0000~0x7FFF 515: 0x0000~**0xF3FF**
(1, 0)	**1KB**	**0x3800~0x3BFF**	51: 0x0000~0x0FFF 52: 0x0000~0x1FFF 53: 0x0000~**0x37FF**	**0xF800~0xFBFF**	54: 0x0000~0x3FFF 58: 0x0000~0x7FFF 515: 0x0000~**0xF7FF**
(1, 1)	**0**	**None**	51: 0x0000~0x0FFF 52: 0x0000~0x1FFF 53: 0x0000~**0x3BFF**	**None**	54: 0x0000~0x3FFF 58: 0x0000~0x7FFF 515: 0x0000~**0xFBFF**

MOVCL:
0: MOVC instruction executed from external program memory is disabled for security.
1: MOVC is always available.

SB:
0: Code dumped on a Writer is scrambled for security.
1: Code dumped on a Writer is not scrambled.

LOCK:
0: Code dumped on a Writer is locked to 0xFF for security.
1: Code dumped on a Writer is not locked.

OR1 (Option Register 1)

7	6	5	4	3	2	1	0
FZWDTCR	(reserved)	(reserved)	**OSCDN**	(reserved)	(reserved)	**HWBS**	**EN6T**

Note: The reserved bit must be kept 1.

FZWDTCR:
0: The WDTCR register will be initialized to its reset value only by power-on reset.
1: The WDTCR register will be initialized to its reset value by all reset (including power-on, H/W, S/W and WDT reset).

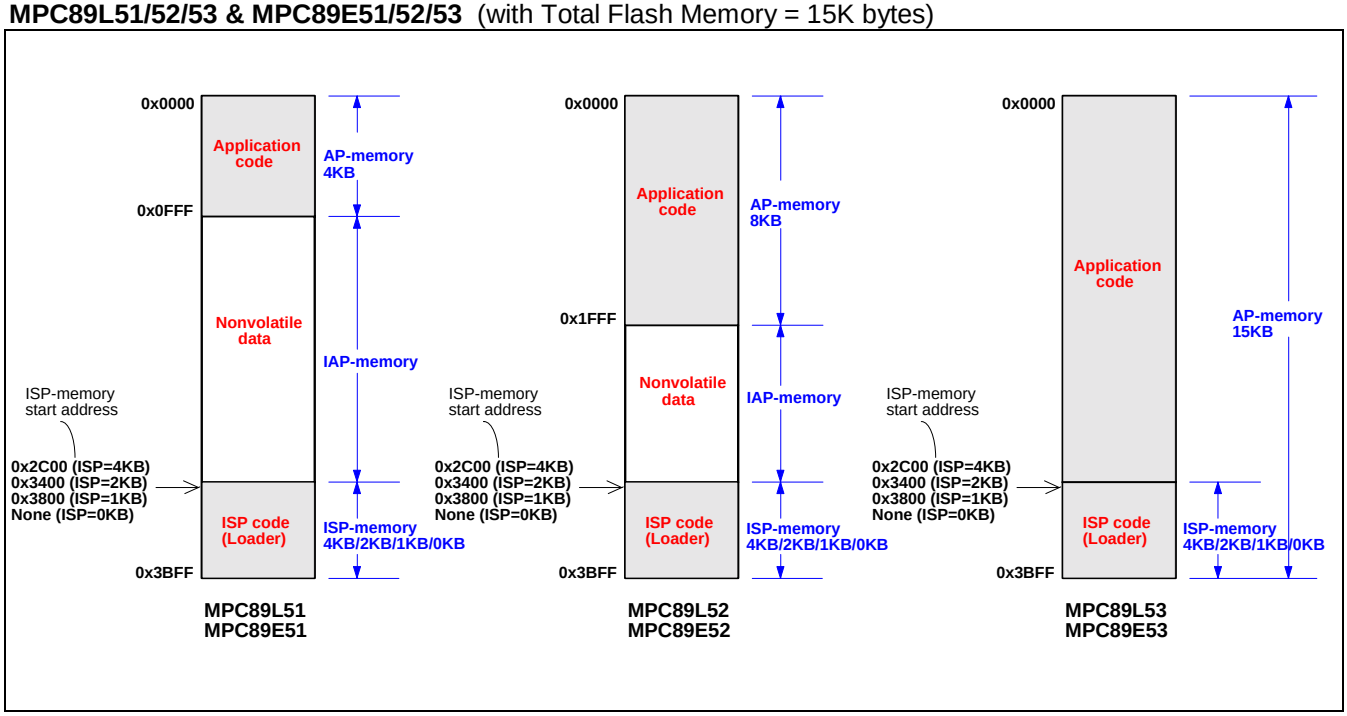
OSCDN:
0: Used under 25MHz for EMI reduction. (The gain of oscillating amplifier is reduced.)
1: The gain of crystal oscillator is enough for higher Fosc oscillating.

HWBS:
0: When power-on, H/W will set SWBS and CPU will boot from ISP-memory if ISP-memory exists.
1: (No action)

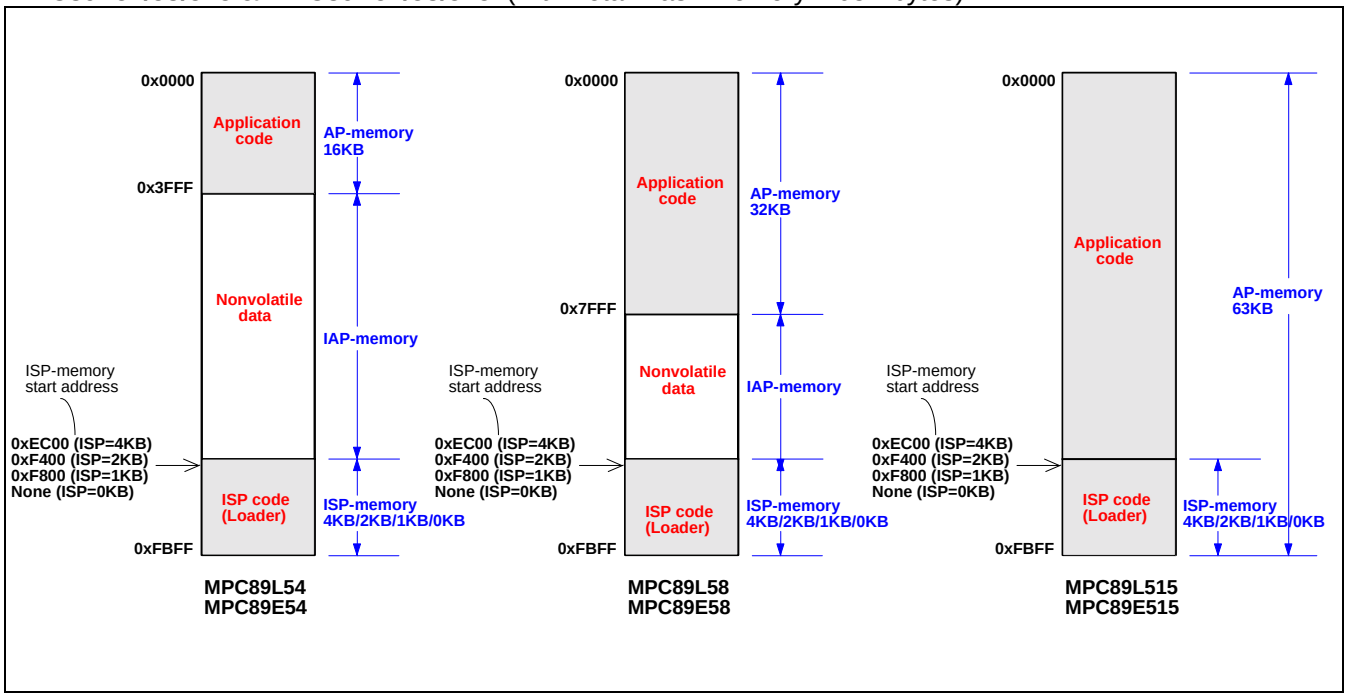
EN6T:
0: MCU runs at 6T mode (each machine-cycle has 6 clocks).
1: MCU runs at 12T mode (each machine-cycle has 12 clocks).

18. Flash Memory Configuration

MPC89L51/52/53 & MPC89E51/52/53 (with Total Flash Memory = 15K bytes)



MPC89L54/58/515 & MPC89E54/58/515 (with Total Flash Memory = 63K bytes)



Note !!!

For MPC89L(E)53/515, there is no IAP-memory, and the ISP-memory overlaps with the AP-memory.

19. On-chip Oscillator and Reset Circuitry

19-1 XTAL Oscillating Requirement

As shown in the Oscillating Circuit, to achieve successful and exact oscillating, the values for C1, C2 and R1 are recommended as below, where OSCDN is in OR1 (Option Register 1)

While OSCDN=1

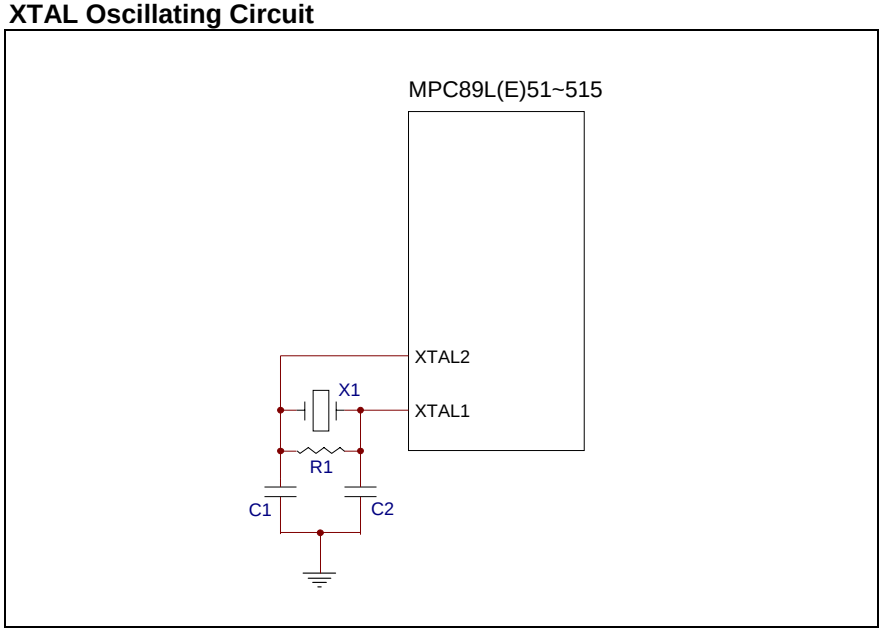
X1	2~19MHz	20~25MHz	26~30MHz	31~35MHz	36~39MHz	40~43MHz	44~48MHz
C1, C2	47pF	33pF	10pF	10pF	10pF	10pF	5pF
R1	-	-	6.8KΩ	5.1KΩ	4.7KΩ	3.3KΩ	3.3KΩ

While OSCDN=0

X1	2~19MHz	20~25MHz	26~30MHz	31~35MHz	36~39MHz	40~43MHz	44~48MHz
C1, C2	47pF	33pF	5pF	-	-	-	-
R1	-	-	6.8KΩ	5.1KΩ	4.7KΩ	3.3KΩ	3.3KΩ

Note:
(1) '-' means no need.
(2) C1 and C2 include the parasitic capacitance in the PCB.

XTAL Oscillating Circuit



19-2 ALE Output Frequency

At 12T-mode, for a successful and exact XTAL oscillating, the ALE output frequency will be Fosc/6 (as that of the standard 8051). While at 6T-mode (EN6T=0 in OR1), it will be Fosc*2/6.

For example,

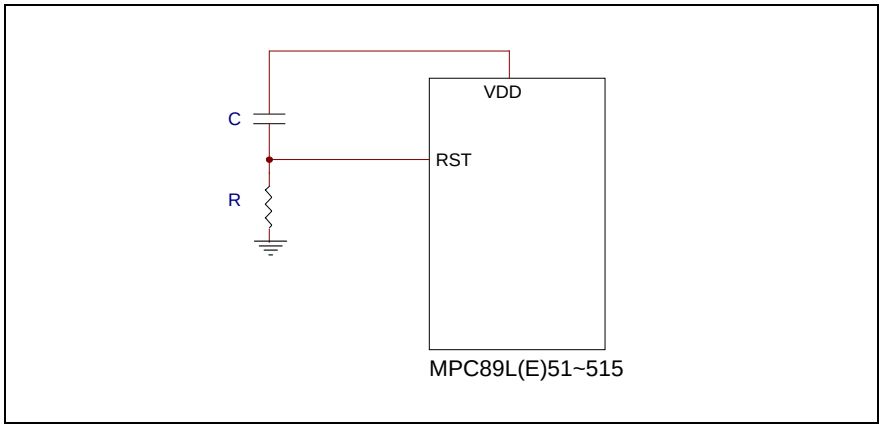
If Fosc=**24**MHz, the ALE output frequency will be **4**MHz for 12T-mode, and **8**MHz for 6T-mode.

19-3 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 suitable for all of the operation frequency range.

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

Reset Circuit



!!! Note:

If your system has parasitic charges (which can not be fully discharged) after powered off, the combination of (R,C)=(1uF,130KΩ) is the best choice for this system to get well reset on next power up.

20. Power Consumption

Fosc	L-series @VDD=3.3V, 25°C						E-series @VDD=5.0V, 25°C					
	12T mode			6T mode			12T mode			6T mode		
	I_{nop} (mA)	I_{idle} (mA)	I_{pwdn} (uA)	I_{nop} (mA)	I_{idle} (mA)	I_{pwdn} (uA)	I_{nop} (mA)	I_{idle} (mA)	I_{pwdn} (uA)	I_{nop} (mA)	I_{idle} (mA)	I_{pwdn} (uA)
4MHz	5.0	1.7	< 1.0	5.4	1.8	< 1.0	8.0	3.7	< 1.0	8.4	3.8	< 1.0
6MHz	5.4	1.8		5.9	2.0		8.9	5.0		9.5	5.0	
8MHz	5.8	2.2		6.5	2.4		10.1	6.2		10.8	6.3	
10MHz	5.7	2.0		6.6	2.2		9.3	5.2		10.3	5.4	
12MHz	6.0	2.0		7.0	2.3		9.5	5.3		10.7	5.6	
16MHz	6.6	2.4		7.9	2.7		10.4	5.9		12.0	6.3	
20MHz	7.0	2.6		8.7	3.0		11.4	6.6		13.3	7.1	
24MHz	7.4	2.7		9.4	3.1		11.6	6.5		13.9	7.1	
27MHz	8.0	3.2		10.2	3.6		14.7	9.4		17.0	9.9	
30MHz	8.8	3.8		11.3	4.3		16.8	11.2		19.3	11.7	
32MHz	9.0	3.9		11.7	4.4		17.1	11.5		19.9	12.1	
36MHz	10.4	5.1		13.4	5.7		20.3	14.7		23.9	15.3	
40MHz	10.6	5.1		14.0	5.8		19.6	13.8		24.2	14.8	
42MHz	10.4	4.7		13.9	5.4		20.9	14.9		25.6	15.9	
44MHz	11.0	5.2		14.7	6.0		21.0	14.9		25.7	15.9	
48MHz	11.4	5.3		15.1	6.1		20.7	14.3		-	-	

Notes:

(1) **I_{nop}** : The I_{DD} current when the MCU executes the following instructions:

```
Loop:
        NOP
        JMP  Loop
```

(2) **I_{idle}** : The I_{DD} current when the MCU is in its idle mode.

(3) **I_{pwdn}** : The I_{DD} current when the MCU is in its power-down mode.

21. How to Reduce EMI

To reduce EMI (Electro Magnetic Interference) generated by the MCU, the following suggestions are useful:

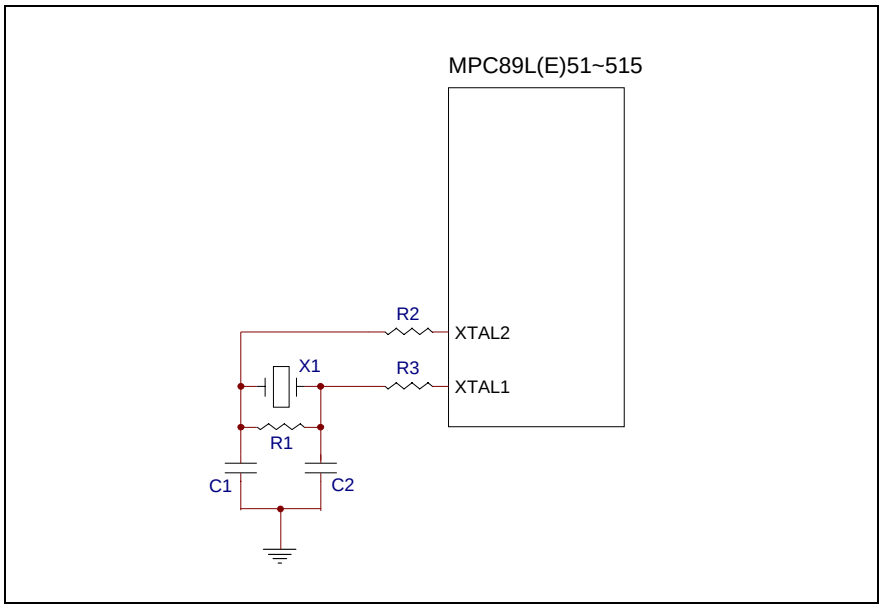
(1) Turn off ALE output as described in Section 12.

(2) Program OSCDN bit (in OR1) to 0 by a Programmer/Writer as described in Section 17.

(3) Use lower Fosc, e.g., lower than 24MHz. (The lower Fosc is, the lower the EMI will be.)

If the performance of 40MHz@12T-mode is necessary, it is strongly recommended to use 20MHz@6T-mode for lower EMI.

(4) Add series resistors R2 and R3 at XTAL1 and XTAL2, respectively, as shown below.



R2 & R3 resistance (for both L-series and E-series)

X1		8MHz	12MHz	16MHz	20MHz	24MHz
C1, C2		47pF	47pF	47pF	47pF	47pF
R1		-	-	-	-	-
R2, R3	OSCDN=1	33~680Ω	33~560Ω	33~430Ω	33~360Ω	33~300Ω
	OSCDN=0	33~560Ω	33~470Ω	33~300Ω	33~220Ω	33~150Ω

22. UART Baudrate Setting

To use **Timer 1** as the baudrate generator, follow the steps:

1) T2CON=xx00xxxx, where x means “don’t care” bit.

2) Timer 1 operates at its Mode 2 (8-bit auto-reload), i.e., TMOD=0010xxxx, where x means “don’t care” bit.

3) Find TH1 by:

For 12T-mode, baudrate = 2^{SMOD} **x** Fosc / (32 **x** 12 **x** (256 – TH1)).
So, TH1 = 256 – 2^{SMOD} **x** Fosc / (baudrate **x** 32**x** 12).

For 6T-mode, baudrate = 2^{SMOD} **x** Fosc **x** 2 / (32 **x** 12 **x** (256 – TH1)).
So, TH1 = 256 – 2^{SMOD} **x** Fosc **x** 2 / (baudrate **x** 32 **x** 12).

Where SMOD is bit-7 of the PCON register.

4) Start Timer 1, set bit TR1 in the TCON register.

To use **Timer 2** as the baudrate generator, follow the steps:

1) Timer 2 operates at its baudrate generator mode, i.e., T2CON=xx11xxxx, where x means “don’t care” bit.

2) Find [RCAP2H,RCAP2L] by:

For 12T-mode, baudrate = Fosc / (2 **x** 16 **x** (65536 – [RCAP2H, RCAP2L])).
So, [RCAP2H, RCAP2L] = 65536 – Fosc / (baudrate **x** 2 **x** 16).

For 6T-mode, baudrate = Fosc **x** 2 / (2 **x** 16 **x** (65536 – [RCAP2H, RCAP2L])).
So, [RCAP2H, RCAP2L] = 65536 – Fosc **x** 2 / (baudrate **x** 2 **x** 16).

3) Start Timer 2, set bit TR2 in the T2CON register

The familiar XTAL frequencies used for UART application are ***11.0592MHz***, ***18.432MHz***, ***22.1184MHz***, ***36.864MHz*** and ***44.2368MHz***. The following tables show the TH1 and [RCAP2H, RCAP2L] values for a variety of the standard baudrate.

Fosc=11.0592MHz

Baudrate	Timer1 (TH1)				Timer2 ([RCAP2H, RCAP2L])	
	12T-mode		6T-mode		12T-mode	6T-mode
	SMOD=0	SMOD=1	SMOD=0	SMOD=1		
300	160	64	64	-	64384	63232
600	208	160	160	64	64960	64384
1200	232	208	208	160	65248	64960
1800	240	224	224	192	65344	65152
2400	244	232	232	208	65392	65248
4800	250	244	244	232	65464	65392
7200	252	248	248	240	65488	65440
9600	253	250	250	244	65500	65464
14400	254	252	252	248	65512	65488
19200	-	253	253	250	65518	65500
38400	-	-	-	253	65527	65518
57600	-	255	255	254	65530	65524
115200	-	-	-	255	65533	65530

Fosc=18.432MHz

Baudrate	Timer1 (TH1)				Timer2 ([RCAP2H, RCAP2L])	
	12T-mode		6T-mode		12T-mode	6T-mode
	SMOD=0	SMOD=1	SMOD=0	SMOD=1		
300	96	-	-	-	63616	61696
600	176	96	96	-	64576	63616
1200	216	176	176	96	65056	64576
1800	-	-	-	-	65216	64896
2400	236	216	216	176	65296	65056
4800	246	236	236	216	65416	65296
7200	-	-	-	-	65456	65376
9600	251	246	246	236	65476	65416
14400	-	-	-	-	65496	65456
19200	-	251	251	246	65506	65476
38400	-	-	-	251	65521	65506
57600	-	-	-	-	65526	65516
115200	-	-	-	-	65531	65526

Fosc=22.1184MHz

Baudrate	Timer1 (TH1)				Timer2 ([RCAP2H, RCAP2L])	
	12T-mode		6T-mode		12T-mode	6T-mode
	SMOD=0	SMOD=1	SMOD=0	SMOD=1		
300	64	-	-	-	63232	60928
600	160	64	64	-	64384	63232
1200	208	160	160	64	64960	64384
1800	224	192	192	128	65152	64768
2400	232	208	208	160	65248	64960
4800	244	232	232	208	65392	65248
7200	248	240	240	224	65440	65344
9600	250	244	244	232	65464	65392
14400	252	248	248	240	65488	65440
19200	253	250	250	244	65500	65464
38400	-	253	253	250	65518	65500
57600	255	254	254	252	65524	65512
115200	-	255	255	254	65530	65524

Fosc=36.864MHz

Baudrate	Timer1 (TH1)				Timer2 ([RCAP2H, RCAP2L])	
	12T-mode		6T-mode		12T-mode	6T-mode
	SMOD=0	SMOD=1	SMOD=0	SMOD=1		
300	-	-	-	-	61696	57856
600	96	-	-	-	63616	61696
1200	176	96	96	-	64576	63616
1800	-	-	-	-	64896	64256
2400	216	176	176	96	65056	64576
4800	236	216	216	176	65296	65056
7200	-	-	-	-	65376	65216
9600	246	236	236	216	65416	65296
14400	-	-	-	-	65456	65376
19200	251	246	246	236	65476	65416
38400	-	251	251	246	65506	65476
57600	-	-	-	-	65516	65496
115200	-	-	-	-	65526	65516

Fosc=44.2368MHz

Baudrate	Timer1 (TH1)				Timer2 ([RCAP2H, RCAP2L])	
	12T-mode		6T-mode		12T-mode	6T-mode
	SMOD=0	SMOD=1	SMOD=0	SMOD=1		
300	-	-	-	-	60928	56320
600	64	-	-	-	63232	60928
1200	160	64	64	-	64384	63232
1800	192	128	128	0	64768	64000
2400	208	160	160	64	64960	64384
4800	232	208	208	160	65248	64960
7200	240	224	224	192	65344	65152
9600	244	232	232	208	65392	65248
14400	248	240	240	224	65440	65344
19200	250	244	244	232	65464	65392
38400	253	250	250	244	65500	65464
57600	254	252	252	248	65512	65488
115200	255	254	254	252	65524	65512

23. Notes on Using External Interrupt

The way to manage external interrupts may exist a difference between the so-called 8051 compatible MCUs.

To safely use external interrupts, please refer to the following example, which includes some useful suggestion.

An Example:

```
         ORG    0000h
         JMP    main

         ORG    0003h   ;/INT0 interrupt vector
         JMP    INT0_isr

;-------------------------------------------------------------
main:
         CLR    IT0     ;IT0=0, select /INT0 as low-level-triggered

         SETB   EA      ;enable /INT0 interrupt
         SETB   EX0     ;
;
loop:
         LCALL  check_alarm
         JMP    loop

;
check_alarm:
;(Suppose: while “check_alarm” is being executed, it must not be interrupted by /INT0.)

         PUSH   IE      ;save IE for the following “CLR EX0”

         CLR    EX0     ;!!! Note: /INT0 interrupt may happen at this time

         NOP            ;!!! insert 3 NOPs here for the case:
         NOP            ;    if /INT0 really happens and CPU can run its i.s.r here
         NOP            ;

         MOV    P2,#81h ;(!!! these instructions must not be interrupted by /INT0)
         MOV    P0,#00h ;
         SET    P1.5    ;
         CLR    P1.5    ;
         MOV    P0,#FFh ;
         CLR    P3.7    ;
         MOV    A,P0    ;
         SETB   P3.7    ;

         POP    IE      ;restore IE

         RET
;
;-------------------------------------------------------------
INT0_isr:
         ;…
         ;…
         SETB   EX0     ;!!! delete this instruction if have,
                        ;    to prevent from /INT0 being enabled again while main program’s
                        ;    "CLR EX0" is being executed
         ;…
         ;…
         RETI
;
```