Contents

List of Figures

List of Tables

1 Description

The MPC82G516A is a single-chip microcontroller based on a high performance 1-T architecture 80C51 CPU that executes instructions in 1~7 clock cycles (about 6~7 times the rate of a standard 8051 device), and has an 8051 compatible instruction set. Therefore at the same performance as the standard 8051, the MPC82G516A can operate at a much lower speed and thereby greatly reduce the power consumption.

The MPC82G516A has 64K bytes of embedded Flash memory for code and data. The Flash memory can be programmed either in parallel mode or in serial mode with the In-System Programming (ISP) and In-Circuit Programming (ICP) capability. And, it also provides the In-Application Programming (IAP) capability. ISP and ICP allow the user to download new code without removing the microcontroller from the actual end product; IAP means that the device can write non-volatile data in the Flash memory while the application program is running. There needs no external high voltage for programming due to its built-in charge-pumping circuitry.

In addition to the standard features of an 8051 MCU (such as 256 bytes scratch-pad RAM, four 8-bit I/O ports, three timer/counters, full-duplex serial port and a multi-source 4-level interrupt controller), many system-level functions have been incorporated into the MPC82G516A. The functions are on-chip 1024 bytes expanded RAM (XRAM), an extra I/O port (P4), 10-bit ADC, PCA, SPI, secondary UART, keypad interrupt, one-time enabled Watchdog Timer, and so forth. These additional functions greatly reduce the discrete component, board space and system cost, and also make the MPC82G516A become a powerful microcontroller for a wide range of applications.

The MPC82G516A has two power-saving modes and an 8-bit system clock prescaler to reduce the power consumption. In the Idle mode the CPU is frozen while the peripherals and the interrupt system are still operating. In the Power-Down mode the RAM and SFRs' value are saved and all other functions are inoperative; most importantly, in the Power-down mode the device can be waked up by the external interrupts. And, the user can further reduce the power consumption by using the 8-bit system clock prescaler to slow down the operating speed.

Additionally, the MPC82G516A is equipped with the Megawin proprietary On-Chip Debug (OCD) interface for In-Circuit Emulator (ICE). The OCD interface provides on-chip and in-system non-intrusive debugging without any target resource occupied. Several operations necessary for an ICE are supported such as Reset, Run, Stop, Step, Run to Cursor and Breakpoint Setting. The user has no need to prepare any development board during firmware developing or the socket adapter used in the traditional ICE probe head. All the thing the user needs to do is to prepare a 4-pin connector for the dedicated OCD interface. This powerful feature make the developing very easy for any user.

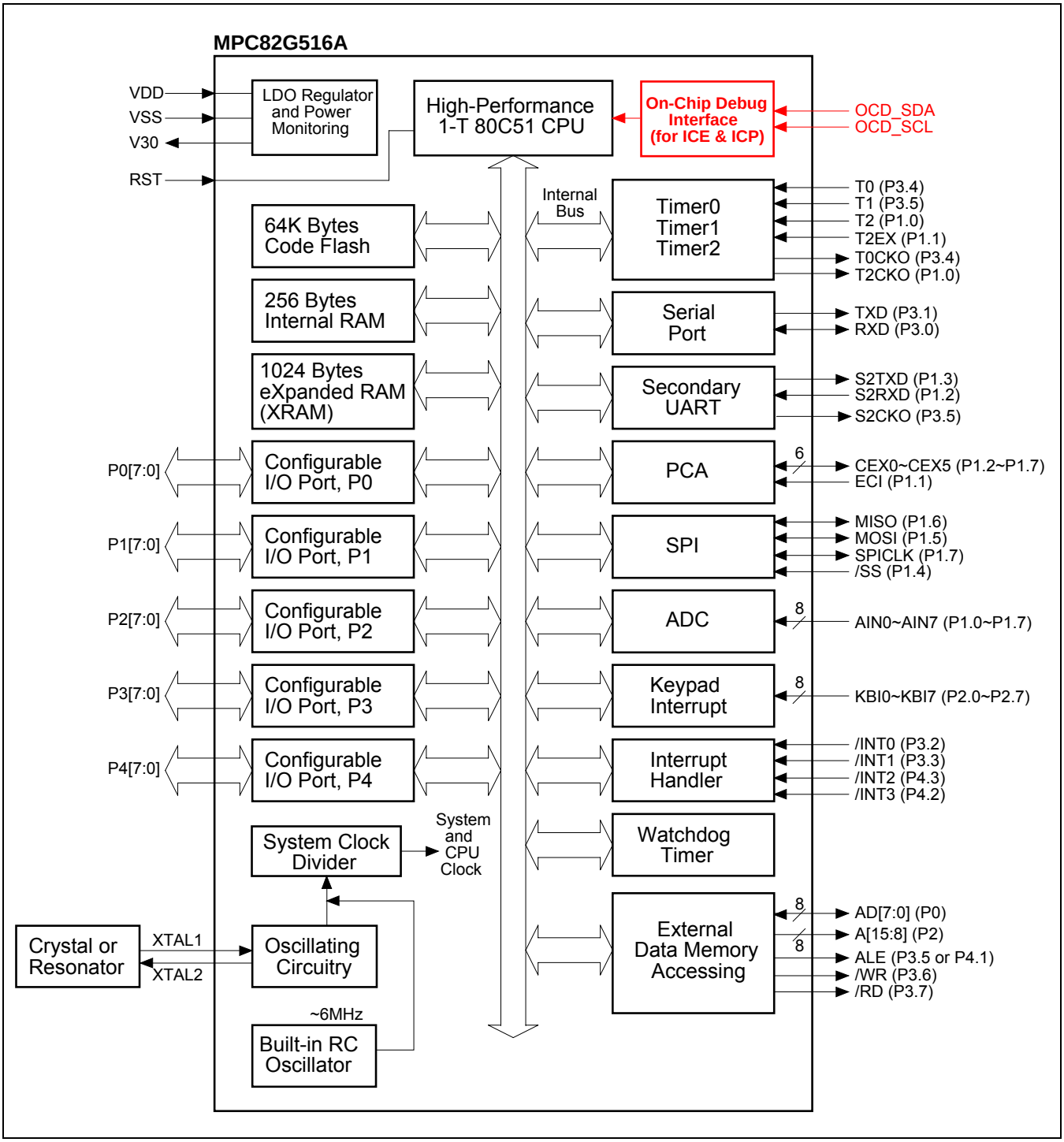
2 Features

- General 8051 Functions
  - 8051 Compatible Instruction Set
  - 256 Bytes Internal Scratch-pad RAM
  - 64K External Data Memory Space
  - Four 8-Bit Bidirectional I/O Ports
  - Three 16-Bit Timer/Counters
  - Full-duplex UART
  - 14 Interrupt Sources with 4 Priority Levels
  - Power Saving Modes: Idle Mode & Power-Down Mode
- High Performance 1-T Architecture 80C51 CPU
- On-chip 64K bytes Flash Program Memory
- On-chip 1024 bytes eXpanded RAM (XRAM)
- Additional Bit-addressable I/O Port, P4
- Configurable I/O Port Type
  - Quasi-bidirectional Output
  - Open-drain Output
  - Input-only
  - Push-pull Output
- Additional External Interrupts /INT2 & /INT3
- Down-counting Capability in Timer2
- Enhanced UART Functions
  - Framing Error Detection
  - Automatic Address Recognition
- Secondary UART with Dedicated Baud Rate Generator
- PCA (Programmable Counter Array) with 6 Modules
  - Capture Mode
  - 16-bit Software Timer Mode
  - High Speed Output Mode
  - PWM (Pulse Width Modulator) Mode
- SPI Interface (Master/Slave Mode)
- 10-Bit ADC with 8 Multiplexed Analog Inputs
- Keypad Interrupt with 8 Inputs
- Wake-up from Power-down Mode by an External Interrupt
- Three Programmable Clock Outputs
- One-time Enabled Watchdog Timer
- Dual DPTR
- Variable Access Timing of 'MOVX' Instruction for Slow External Data Memory
- Configurable System Clock for Reduction of Power Consumption
- Power Monitoring Functions: Brownout Detection & Power-on Flag
- ISP (In-System Programming) & ICP (In-Circuit Programming) to Update Program Memory
- IAP (In-Application Programming) Flash for Applications with Non-volatile Data
- OCD (On-Chip Debug) Interface for ICE
- Flash Endurance: 20,000 Erase/Write cycles
- Operating Frequency: Up to 24MHz
- Power Supply:
  2.4V~3.6V (for 3.3V System), or 2.7V~5.5V (for 5V or Wide-Range System)
- Industrial Temperature Range: -40 to +85 °C
- Packages: PDIP40, PLCC44, PQFP44, LQFP48

3 Block Diagram

Figure 3-1 shows the functional block diagram of the MPC82G516A. It gives the outline of the device. The user can easily find all the device's peripheral functions in the diagram.

Figure 3-1. Block Diagram



4 Pin Configuration

4.1 Pin Assignment

Figure 4-1. Pin Assignment: 40-Pin PDIP

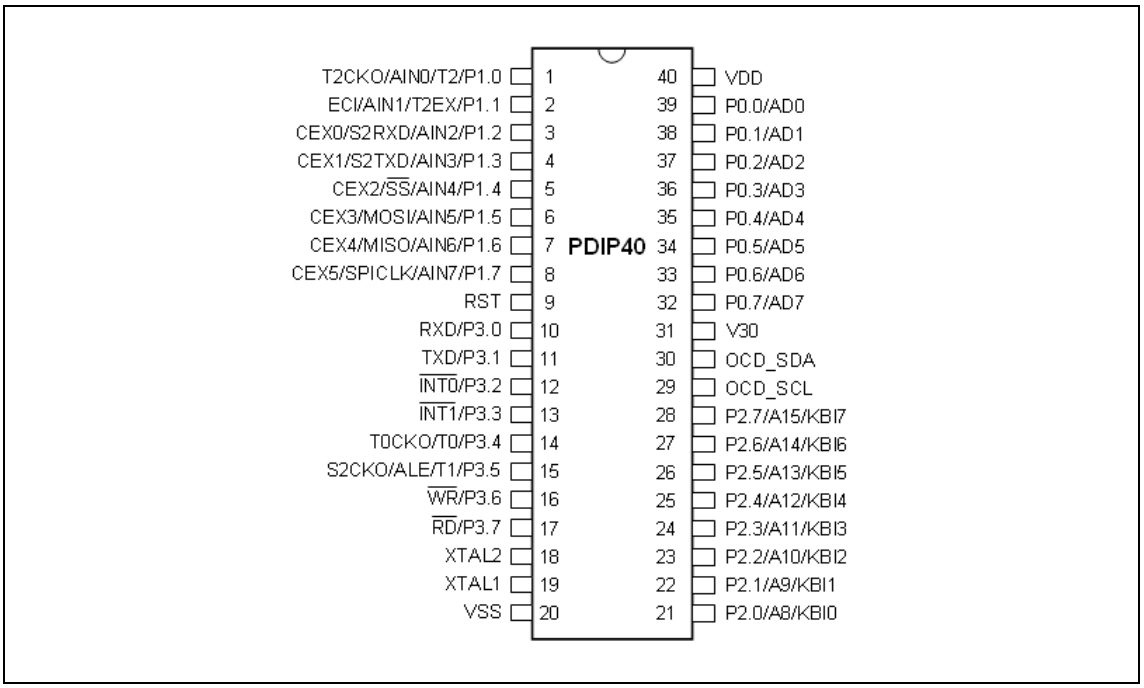


Figure 4-3. Pin Assignment: 44-Pin PLCC

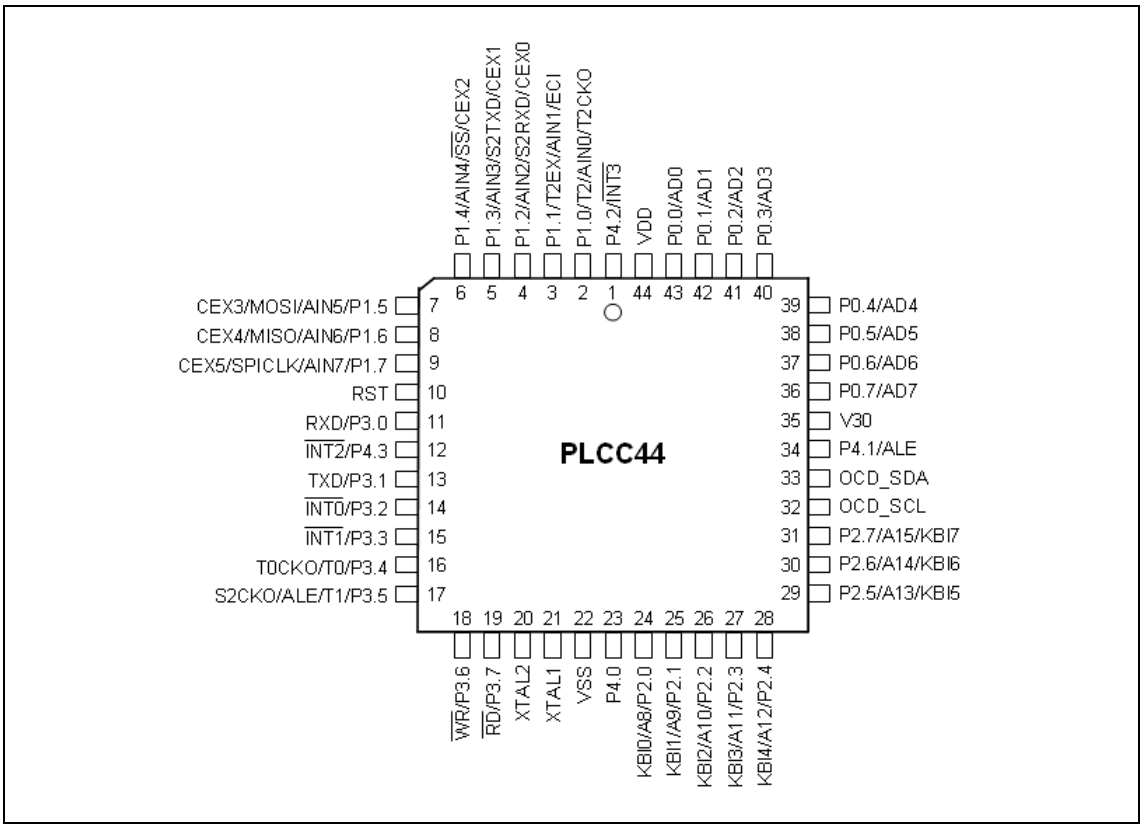


Figure 4-4. Pin Assignment: 44-Pin PQFP

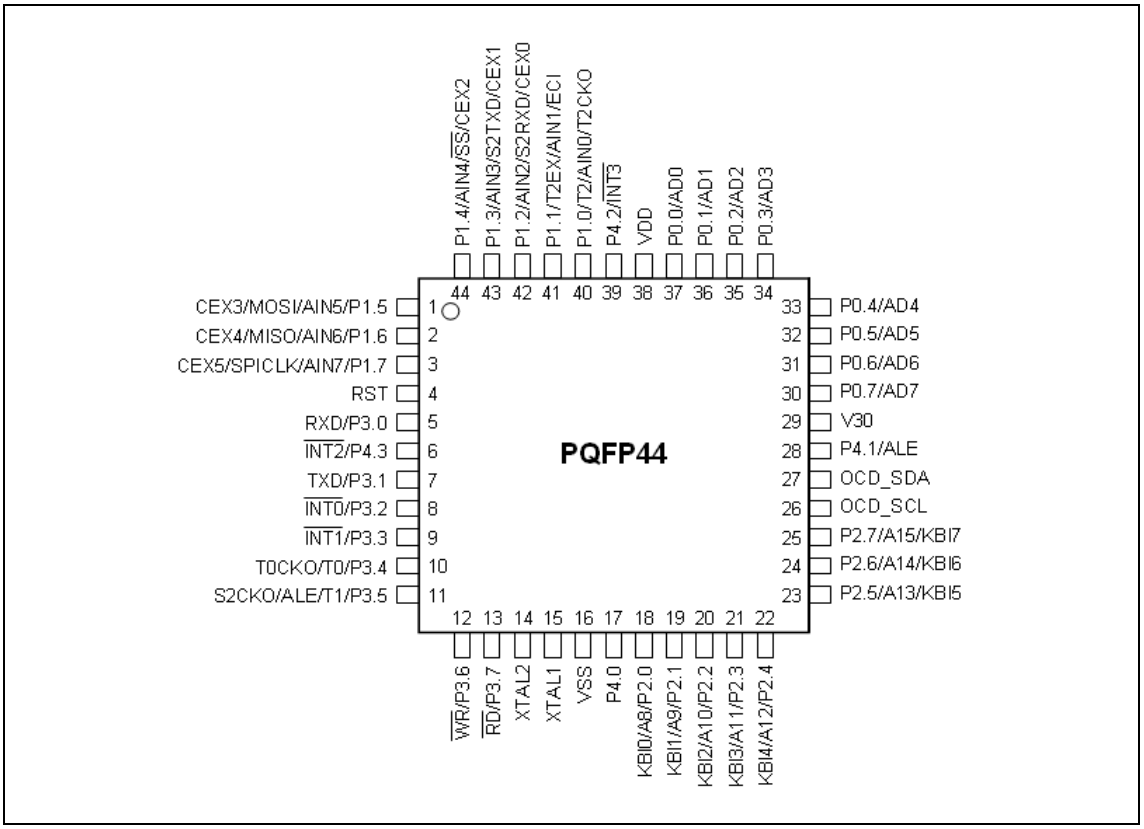
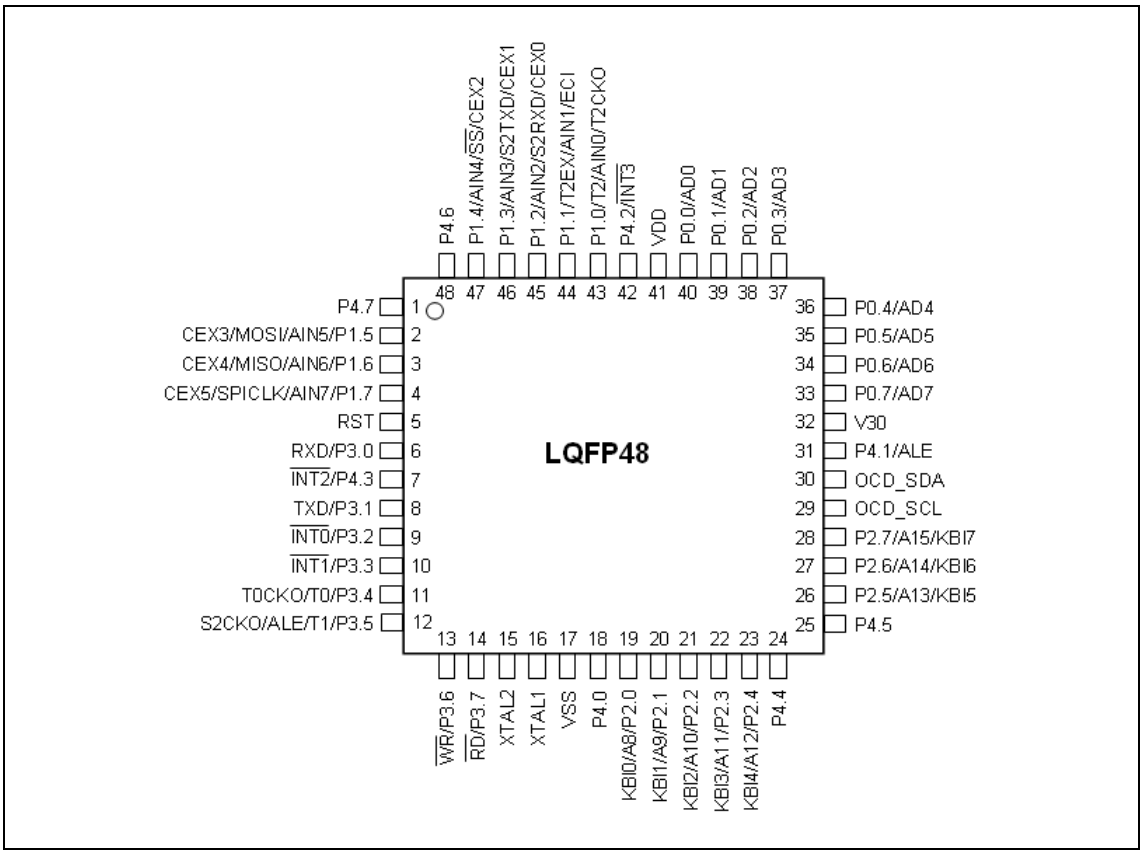


Figure 4-5. Pin Assignment: 48-Pin LQFP



4.2 Pin Description

Table 4-1. Pin Description

MNEMONIC	PIN NUMBER					I/O TYPE	DESCRIPTION
	40-Pin DIP	44-Pin PLCC	44-Pin PQFP	48-Pin LQFP			
P0.0 (Alt. Fun.) AD0	39	43	37	40		I/O I/O	* Port 0 bit-0. * AD0: multiplexed A0/D0 during external data memory access.
P0.1 (Alt. Fun.) AD1	38	42	36	39		I/O I/O	* Port 0 bit-1. * AD1: multiplexed A1/D1 during external data memory access.
P0.2 (Alt. Fun.) AD2	37	41	35	38		I/O I/O	* Port 0 bit-2. * AD2: multiplexed A2/D2 during external data memory access.
P0.3 (Alt. Fun.) AD3	36	40	34	37		I/O I/O	* Port 0 bit-3. * AD3: multiplexed A3/D3 during external data memory access.
P0.4 (Alt. Fun.) AD4	35	39	33	36		I/O I/O	* Port 0 bit-4. * AD4: multiplexed A4/D4 during external data memory access.
P0.5 (Alt. Fun.) AD5	34	38	32	35		I/O I/O	* Port 0 bit-5. * AD5: multiplexed A5/D5 during external data memory access.
P0.6 (Alt. Fun.) AD6	33	37	31	34		I/O I/O	* Port 0 bit-6. * AD6: multiplexed A6/D6 during external data memory access.
P0.7 (Alt. Fun.) AD7	32	36	30	33		I/O I/O	* Port 0 bit-7. * AD7: multiplexed A7/D7 during external data memory access.
P1.0 (Alt. Fun.) T2 (Alt. Fun.) AIN0 (Alt. Fun.) T2CKO	1	2	40	43		I/O I I O	* Port 1 bit-0. * T2: Timer/Counter 2 external input. * AIN0: ADC channel-0 analog input. * T2CKO: programmable clock-out from Timer 2.
P1.1 (Alt. Fun.) T2EX (Alt. Fun.) AIN1 (Alt. Fun.) ECI	2	3	41	44		I/O I I I	* Port 1 bit-1. * T2EX: Timer/Counter 2 Reload/Capture/Direction control. * AIN1: ADC channel-1 analog input. * ECI: PCA external clock input.
P1.2 (Alt. Fun.) AIN2 (Alt. Fun.) S2RXD (Alt. Fun.) CEX0	3	4	42	45		I/O I I I/O	* Port 1 bit-2. * AIN2: ADC channel-2 analog input. * S2RXD: 2nd UART serial input port. * CEX0: PCA module-0 external I/O.
P1.3 (Alt. Fun.) AIN3 (Alt. Fun.) S2TXD (Alt. Fun.) CEX1	4	5	43	46		I/O I O I/O	* Port 1 bit-3. * AIN3: ADC channel-3 analog input. * S2TXD: 2nd UART serial output port. * CEX1: PCA module-1 external I/O.
P1.4 (Alt. Fun.) AIN4 (Alt. Fun.) /SS (Alt. Fun.) CEX2	5	6	44	47		I/O I I I/O	* Port 1 bit-4. * AIN4: ADC channel-4 analog input. * /SS: SPI Slave select. * CEX2: PCA module-2 external I/O.
P1.5 (Alt. Fun.) AIN5 (Alt. Fun.) MOSI (Alt. Fun.) CEX3	6	7	1	2		I/O I I/O I/O	* Port 1 bit-5. * AIN5: ADC channel-5 analog input. * MOSI: SPI master out & slave in. * CEX3: PCA module-3 external I/O.
P1.6 (Alt. Fun.) AIN6 (Alt. Fun.) MISO (Alt. Fun.) CEX4	7	8	2	3		I/O I I/O I/O	* Port 1 bit-6. * AIN6: ADC channel-6 analog input. * MISO: SPI master in & slave out. * CEX4: PCA module-4 external I/O.
P1.7 (Alt. Fun.) AIN7 (Alt. Fun.) SPICLK (Alt. Fun.) CEX5	8	9	3	4		I/O I I/O I/O	* Port 1 bit-7. * AIN7: ADC channel-7 analog input. * SPICLK: SPI clock, output for master and input for slave. * CEX5: PCA module-5 external I/O.

(Continued)

MNEMONIC	PIN NUMBER					I/O TYPE	DESCRIPTION
	40-Pin DIP	44-Pin PLCC	44-Pin PQFP	48-Pin LQFP			
P2.0 (Alt. Fun.) A8 (Alt. Fun.) KBI0	21	24	18	19		I/O O I	* Port 2 bit-0. * A8: A8 output during external data memory access. * KBI0: keypad input 0.
P2.1 (Alt. Fun.) A9 (Alt. Fun.) KBI1	22	25	19	20		I/O O I	* Port 2 bit-1. * A9: A9 output during external data memory access. * KBI1: keypad input 1.
P2.2 (Alt. Fun.) A10 (Alt. Fun.) KBI2	23	26	20	21		I/O O I	* Port 2 bit-2. * A10: A10 output during external data memory access. * KBI2: keypad input 2.
P2.3 (Alt. Fun.) A11 (Alt. Fun.) KBI3	24	27	21	22		I/O O I	* Port 2 bit-3. * A11: A11 output during external data memory access. * KBI3: keypad input 3.
P2.4 (Alt. Fun.) A12 (Alt. Fun.) KBI4	25	28	22	23		I/O O I	* Port 2 bit-4. * A12: A12 output during external data memory access. * KBI4: keypad input 4.
P2.5 (Alt. Fun.) A13 (Alt. Fun.) KBI5	26	29	23	26		I/O O I	* Port 2 bit-5. * A13: A13 output during external data memory access. * KBI5: keypad input 5.
P2.6 (Alt. Fun.) A14 (Alt. Fun.) KBI6	27	30	24	27		I/O O I	* Port 2 bit-6. * A14: A14 output during external data memory access. * KBI6: keypad input 6.
P2.7 (Alt. Fun.) A15 (Alt. Fun.) KBI7	28	31	25	28		I/O O I	* Port 2 bit-7. * A15: A15 output during external data memory access. * KBI7: keypad input 7.
P3.0 (Alt. Fun.) RXD	10	11	5	6		I/O I/O	* Port 3 bit-0. * RXD: serial input port, data i/o in mode 0.
P3.1 (Alt. Fun.) TXD	11	13	7	8		I/O O	* Port 3 bit-1. * TXD: serial output port.
P3.2 (Alt. Fun.) /INT0	12	14	8	9		I/O I	* Port 3 bit-2. * /INT0: external interrupt 0 input.
P3.3 (Alt. Fun.) /INT1	13	15	9	10		I/O I	* Port 3 bit-3. * /INT1: external interrupt 1 input.
P3.4 (Alt. Fun.) T0 (Alt. Fun.) T0CKO	14	16	10	11		I/O I O	* Port 3 bit-4. * T0: Timer/Counter 0 external input. * T0CKO: programmable clock-out from Timer 0.
P3.5 (Alt. Fun.) T1 (Alt. Fun.) ALE (Alt. Fun.) S2CKO	15	17	11	12		I/O I O O	* Port 3 bit-5. * T1: Timer/Counter 1 external input. * ALE: Address Latch Enable, output pulse for latching the low byte of the address during an access to external data memory. * S2CKO: programmable clock-out from Timer S2BRT.
P3.6 (Alt. Fun.) /WR	16	18	12	13		I/O O	* Port 3 bit-6. * /WR: external data memory write strobe.
P3.7 (Alt. Fun.) /RD	17	19	13	14		I/O O	* Port 3 bit-7. * /RD: external data memory read strobe.

(Continued)

MNEMONIC	PIN NUMBER					I/O TYPE	DESCRIPTION
	40-Pin DIP	44-Pin PLCC	44-Pin PQFP	48-Pin LQFP			
P4.0	-	23	17	18		I/O	* Port 4 bit-0.
P4.1 (Alt. Fun.) ALE	-	34	28	31		I/O O	* Port 4 bit-1. * ALE: Address Latch Enable, output pulse for latching the low byte of the address during an access to external data memory.
P4.2 (Alt. Fun.) /INT3	-	1	39	42		I/O I	* Port 4 bit-2. * /INT3: external interrupt 3 input.
P4.3 (Alt. Fun.) /INT2	-	12	6	7		I/O I	* Port 4 bit-3. * /INT2: external interrupt 2 input.
P4.4	-	-	-	24		I/O	* Port 4 bit-4.
P4.5	-	-	-	25		I/O	* Port 4 bit-5.
P4.6	-	-	-	48		I/O	* Port 4 bit-6.
P4.7	-	-	-	1		I/O	* Port 4 bit-7.
OCD_SDA	30	33	27	30		I/O	On-Chip Debug Interface, serial data.
OCD_SCL	29	32	26	29		I	On-Chip Debug Interface, serial clock.
XTAL1	19	21	15	16		I	Crystal1: input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	20	14	15		O	Crystal 2: output from the inverting oscillator amplifier.
RST	9	10	4	5		I	A high on this pin for 24 clock cycles while the oscillator is running, resets the device.
V30	31	35	29	32		O	Output of the internal LDO (Low Drop-Out voltage regulator): should be connected to ground through an external capacitor (4.7μF~100μF) in the application with VDD power higher than 3.6V; and tied to VDD pin in the application with VDD power lower than 3.6V.
VDD	40	44	38	41		I	Power supply, for normal, idle and power-down operation.
VSS	20	22	16	17		I	Ground, 0 V reference.

Note: "(Alt. Fun.)" means the Alternate Function of this pin.

4.3 Alternate Function Redirection

Many I/O pins, in addition to their normal I/O function, also serve the alternate function for internal peripherals. For the peripherals Keypad interrupt, PCA, SPI and UART2, Port 2 and Port 1 serve the alternate function in the default state. However, the user may select Port 4 to serve their alternate function by setting the corresponding control bits P4KB, P4PCA, P4SPI and P4S2 in AUXR1 register. It is especially useful when the package more than 40 pins is adopted. Note that only one of the four control bits can be set at any time.

AUXR1 (Address=A2H, Auxiliary Register1, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P4KB	P4PCA	P4SPI	P4S2	GF2	-	-	DPS

P4KB: When set, the Keypad interface is directed to P4, as shown below.
'KBI7' function in P2.7 is moved to P4.7.
'KBI6' function in P2.6 is moved to P4.6.
'KBI5' function in P2.5 is moved to P4.5.
'KBI4' function in P2.4 is moved to P4.4.
'KBI3' function in P2.3 is moved to P4.3.
'KBI2' function in P2.2 is moved to P4.2.
'KBI1' function in P2.1 is moved to P4.1.
'KBI0' function in P2.0 is moved to P4.0.

P4PCA: When set, the PCA interface is directed to P4, as shown below.
'ECI' function in P1.1 is moved to P4.1.
'CEX0' function in P1.2 is moved to P4.2.
'CEX1' function in P1.3 is moved to P4.3.
'CEX2' function in P1.4 is moved to P4.4.
'CEX3' function in P1.5 is moved to P4.5
'CEX4' function in P1.6 is moved to P4.6.
'CEX5' function in P1.7 is moved to P4.7.

P4SPI: When set, the SPI interface is directed to P4, as shown below.
'/SS' function in P1.4 is moved to P4.4.
'MOSI' function in P1.5 is moved to P4.5.
'MISO' function in P1.6 is moved to P4.6.
'SPICLK' function in P1.7 is moved to P4.7.

P4S2: When set, the UART2 interface is directed to P4, as shown below.
'S2RXD' function in P1.2 is moved to P4.2.
'S2TXD' function in P1.3 is moved to P4.3.

5 Memory Organization

Like all 80C51 devices, the MPC82G516A has separate address spaces for program and data memory. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by the 8-bit CPU.

Program memory (ROM) can only be read, not written to. There can be up to 64k bytes of program memory. In the MPC82G516A, all the program memory are on-chip Flash memory, and without the capability of accessing external program memory because of no External Access Enable (/EA) and Program Store Enable (/PSEN) signals designed.

Data memory occupies a separate address space from program memory. In the MPC82G516A, there are 256 bytes of internal scratch-pad RAM. Up to 64K bytes of memory space for external data memory. The CPU generates the 16-bit addresses through the DPTR register and read and write strobe signals (/RD and /WR, respectively) as needed during external data memory accesses. For many applications which need a little more internal RAM, the MPC82G516A has incorporated 1024 bytes of external RAM to become on-chip expanded RAM (called XRAM).

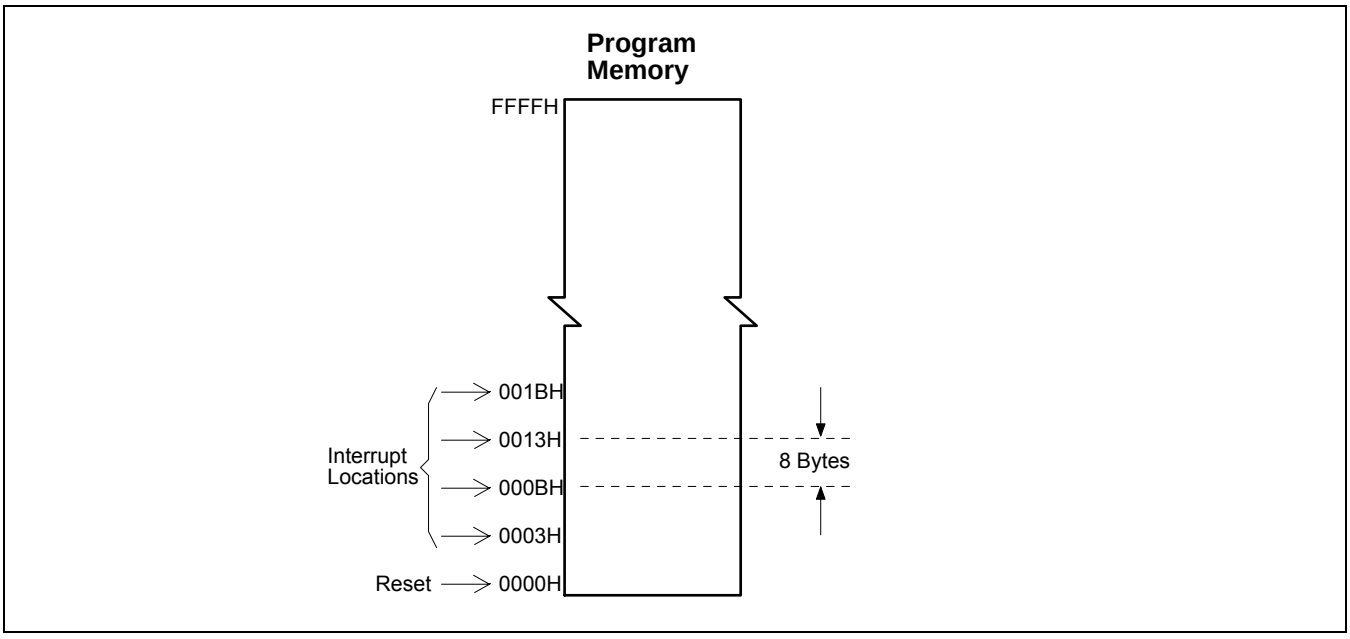
5.1 Program Memory

Program memory is the memory which stores the program codes for the CPU to execute, as shown in Figure 5-1. After reset, the CPU begins execution from location 0000H, where should be the starting of the user's application code. To service the interrupts, the interrupt service locations (called interrupt vectors) should be located in the program memory. Each interrupt is assigned a fixed location in the program memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose program memory.

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

Note that, for the MPC82G516A, there is no capability of execution of external program. All the user's program code are programmed in the on-chip Flash memory. So, the /EA and /PSEN signals are no more needed and thus omitted. The user should pay attention to it.

Figure 5-1. Program Memory



5.2 Data Memory

Figure 5-2 shows the internal and external data memory spaces available to the MPC82G516A user. Internal data memory can be divided into three blocks, which are generally referred to as the lower 128 bytes of RAM, the upper 128 bytes of RAM, and the 128 bytes of SFR space. Internal data memory addresses are always 8-bit wide, which implies an address space of only 256 bytes. Direct addresses higher than 7FH access the SFR space; and indirect addresses higher than 7FH access the upper 128 bytes of RAM. Thus the SFR space and the upper 128 bytes of RAM occupy the same block of addresses, 80H through FFH, although they are physically separate entities.

The lower 128 bytes of RAM are present in all 80C51 devices as mapped in Figure 5-3. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing. The next 16 bytes above the register banks form a block of bit-addressable memory space. The 80C51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

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

Figure 5-4 gives a brief look at the Special Function Register (SFR) space. SFRs include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 0H or 8H.

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

Accessing by an 8-bit address

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

Accessing by a 16-bit address

16-bit addresses are often used to access up to 64k bytes of external data memory. Figure 5-6 shows the hardware configuration for accessing 64K bytes of external RAM. Whenever a 16-bit address is used, in addition to the functioning of P0, /RD and /WR, the high byte of the address comes out on Port 2 and it is held during the read or write cycle.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. ALE (Address Latch Enable) should be used to capture the address byte into an external latch. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before /WR is activated, and remains there until after /WR is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated. During any access to external memory, the CPU writes 0FFH to the Port 0 latch (the Special Function Register), thus obliterating whatever information the Port 0 SFR may have been holding. Note that in the MPC82G516A, there is no dedicated pin for ALE signal. The ALE becomes an alternate function of P3.5 or P4.1, which can be selected by control bits P35ALE and P41ALE in the AUXR register.

To access the on-chip expanded RAM (XRAM), the EXTRAM bit should be cleared to 0. Refer to Figure 5-2, the 1024 bytes of XRAM (0000H to 03FFH) are indirectly accessed by move external instruction, MOVX. An access to XRAM will have not any outputting of address, address latch enable and read/write strobe. That means P0, P2, P3.5/P4.1(ALE), P3.6 (/WR) and P3.7 (/RD) will keep unchanged during access of XRAM.

Figure 5-2. Data Memory

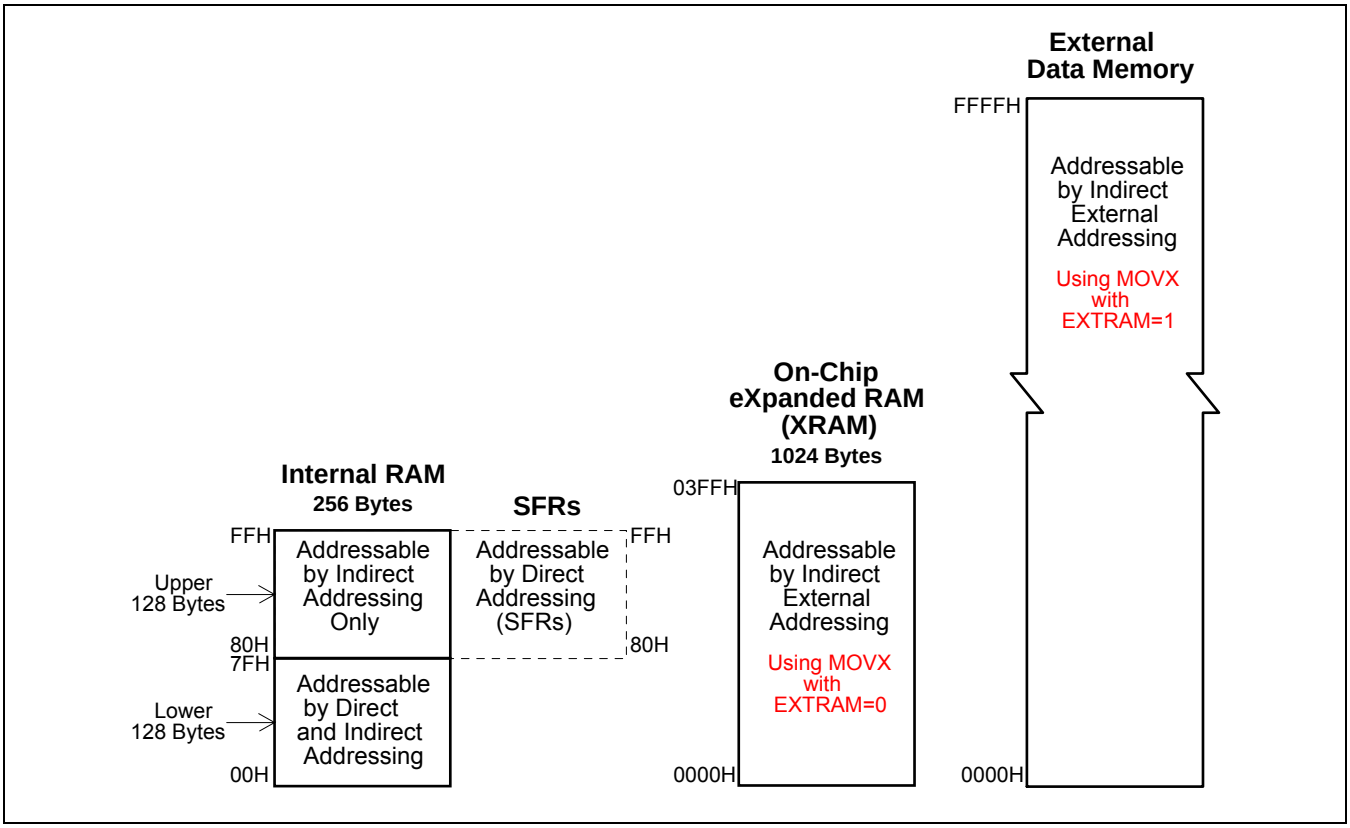


Figure 5-3. Lower 128 Bytes of Internal RAM

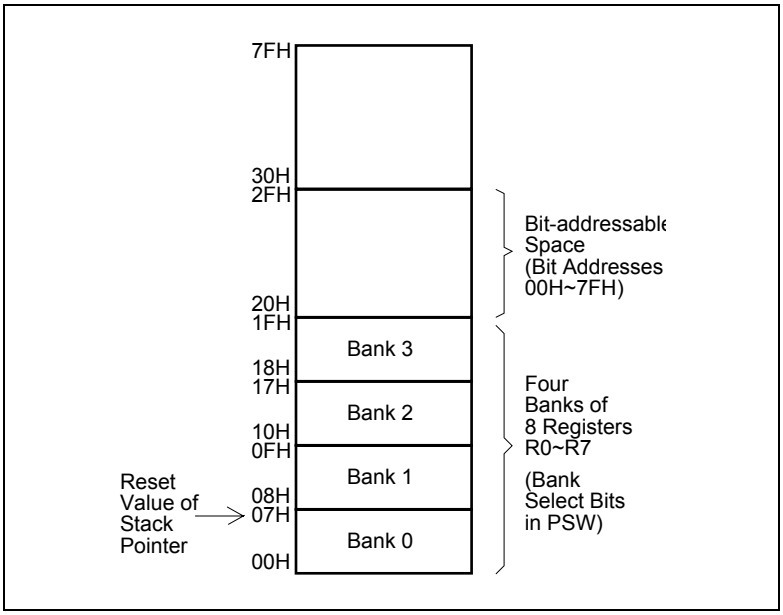


Figure 5-4. SFR Space

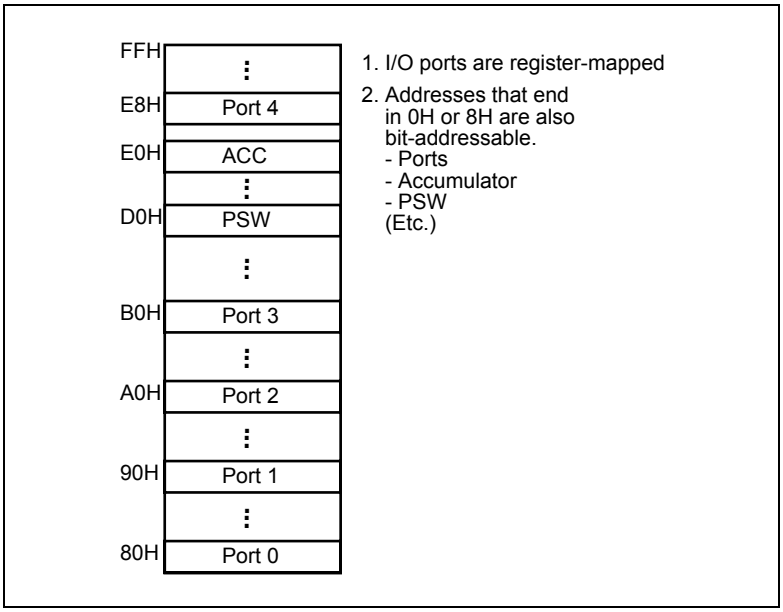
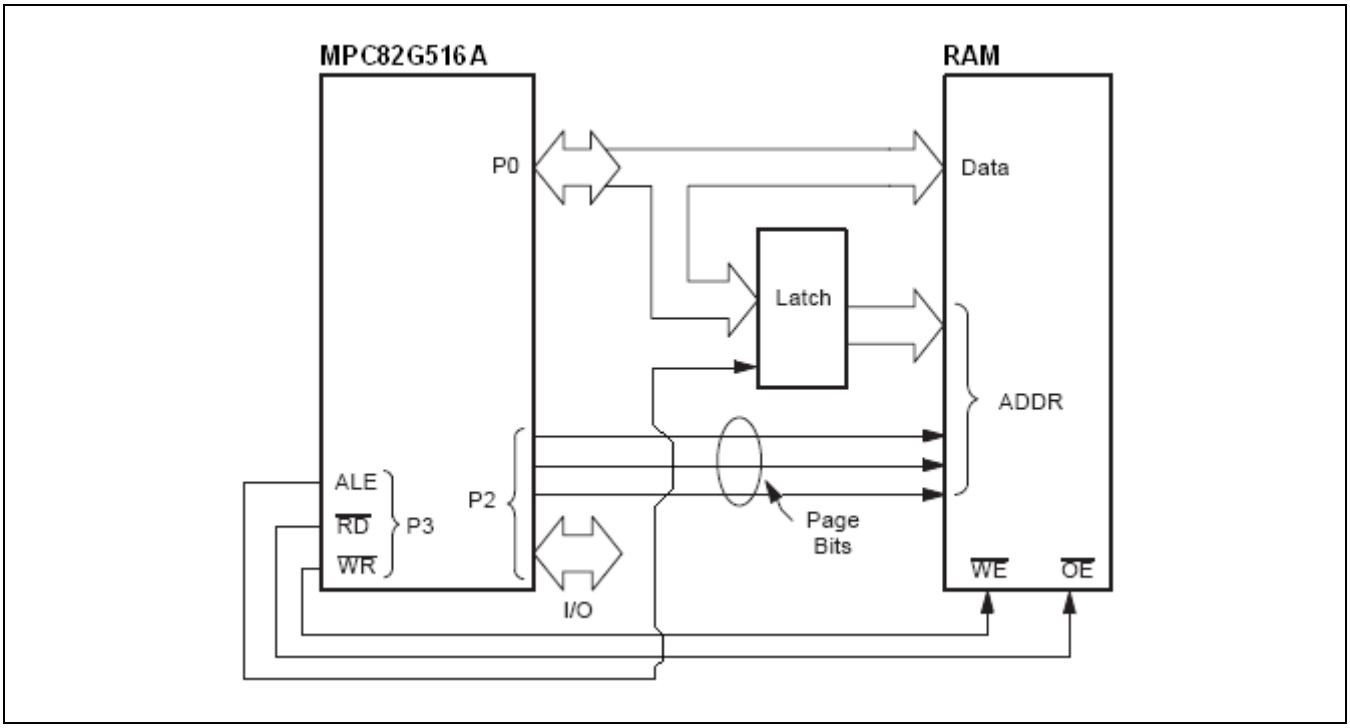
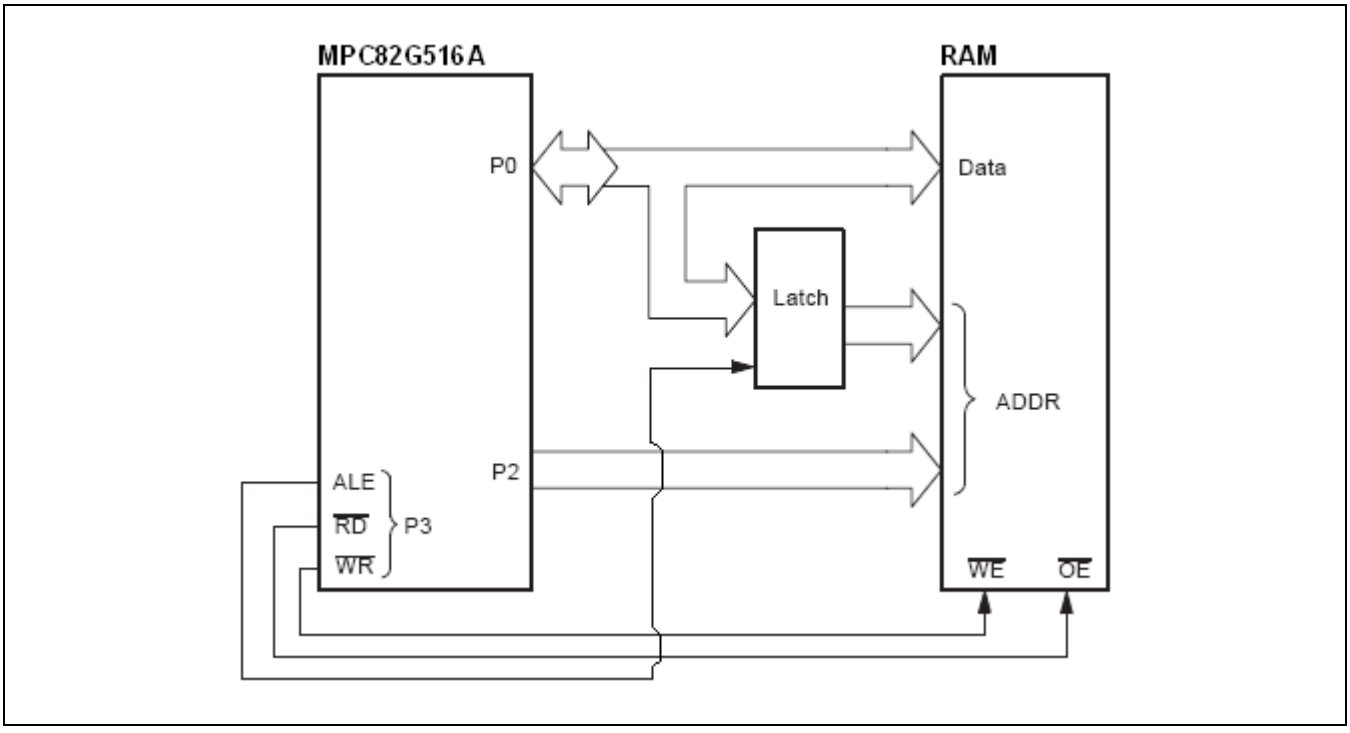


Figure 5-5. External RAM Accessing by an 8-Bit Address (using 'MOVX @ Ri' and Page Bits)



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

Figure 5-6. External RAM Accessing by a 16-Bit Address (using 'MOVX @ DPTR')



5.3 Declaration Identifiers in a C51-Compiler

The declaration identifiers in a C51-compiler for the various MPC82G516A memory spaces are as follows:

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

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

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

xdata
External data or on-chip eXpanded RAM (XRAM); duplicates the classic 80C51 64K bytes memory space addressed via the "MOVX @DPTR" instruction. The MPC82G516A has 1024 bytes of on-chip ***xdata*** memory.

pdata
Paged (256 bytes) external data or on-chip eXpanded RAM; duplicates the classic 80C51 64KB memory space addressed via the "MOVX @Ri" instruction. The MPC82G516A has 1024 bytes of on-chip ***xdata*** memory.

code
64K bytes of program memory space; accessed as part of program execution and via the "MOVC @A+DTPR" instruction. The MPC82G516A has 64K bytes of on-chip ***code*** memory.

6 Special Function Registers (SFRs)

6.1 SFR Memory Map

A map of the internal memory area for the Special Function Register space is called “SFR Memory Map”, as shown in Table 6-1. In the SFR Memory Map, not all of the addresses are occupied. Unoccupied addresses are not implemented or designed for internal testing purpose; read accesses to these addresses will in general return random data, and write accesses may cause unpredictable hardware action. The user software had better not access the unoccupied addresses.

Table 6-1. SFR Memory Map

	8 BYTES								
F8H	**-**	**CH**	**CCAP0H**	**CCAP1H**	**CCAP2H**	**CCAP3H**	**CCAP4H**	**CCAP5H**	FFH
F0H	**B**	**-**	**PCAPWM0**	**PCAPWM1**	**PCAPWM2**	**PCAPWM3**	**PCAPWM4**	**PCAPWM5**	F7H
E8H	**P4**	**CL**	**CCAP0L**	**CCAP1L**	**CCAP2L**	**CCAP3L**	**CCAP4L**	**CCAP5L**	EFH
E0H	**ACC**	**WDTCR**	**IFD**	**IFADRH**	**IFADRL**	**IFMT**	**SCMD**	**ISPCR**	E7H
D8H	**CCON**	**CMOD**	**CCAPM0**	**CCAPM1**	**CCAPM2**	**CCAPM3**	**CCAPM4**	**CCAPM5**	DFH
D0H	**PSW**	**-**	**-**	**-**	**-**	**KBPATN**	**KBCON**	**KBMASK**	D7H
C8H	**T2CON**	**T2MOD**	**RCAP2L**	**RCAP2H**	**TL2**	**TH2**	**-**	**-**	CFH
C0H	**XICON**	**-**	**-**	**-**	**-**	**ADCTL**	**ADCH**	**PCON2**	C7H
B8H	**IP**	**SADEN**	**S2BRT**	**-**	**-**	**-**	**ADCL**	**-**	BFH
B0H	**P3**	**P3M0**	**P3M1**	**P4M0**	**P4M1**			**IPH**	B7H
A8H	**IE**	**SADDR**	**S2CON**	**-**	**-**	**AUXIE**	**AUXIP**	**AUXIPH**	AFH
A0H	**P2**		**AUXR1**	**-**	**-**	**-**	**AUXR2**	**-**	A7H
98H	**SCON**	**SBUF**	**S2BUF**	**-**	**-**	**-**	**-**	**-**	9FH
90H	**P1**	**P1M0**	**P1M1**	**P0M0**	**P0M1**	**P2M0**	**P2M1**	**EVRCR**	97H
88H	**TCON**	**TMOD**	**TL0**	**TL1**	**TH0**	**TH1**	**AUXR**	**STRETCH**	8FH
80H	**P0**	**SP**	**DPL**	**DPH**	**SPSTAT**	**SPCTL**	**SPDAT**	**PCON**	87H

↑ Bit-Addressable SFRs

Note that new added SFRs are marked by the blue bold.

6.2 SFR Introduction

6.2.1 The Standard 80C51 SFRs

The standard 80C51 SFRs are shown in Table 6-2. Among them, the functions of the C51 core registers are outlined below. More information on the use of the other standard SFRs is included in the description of those peripherals.

C51 Core Registers

Accumulator: ACC is the Accumulator register. The mnemonics for Accumulator-Specific instructions, however, refer to the Accumulator simply as A.

B Register: The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad or general purpose register.

Stack Pointer: The Stack Pointer register is 8 bits wide. It denotes the top of the Stack, which is the last used value. The user can place the Stack anywhere in the internal scratchpad RAM by setting the Stack Pointer to the desired location, although the lower bytes are normally used for working registers. On reset, the Stack Pointer is initialized to 07H. This causes the stack to begin at location 08H.

Data Pointer: The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address to assign a memory address for the MOVX instructions. This address can point to a program/data memory location, either on- or off-chip, or a memory-mapped peripheral. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

Program Status Word: The PSW register contains program status information as detailed in the following.

PSW (Address=D0H, Program Status Word, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS2	OV	-	P

CY: Carry flag.
This bit is set when the last arithmetic operation resulted in a carry (addition) or a borrow (subtraction). It is cleared to logic 0 by all other arithmetic operations.

AC: Auxiliary Carry flag. (For BCD Operations)
This bit is set when the last arithmetic operation resulted in a carry into (addition) or a borrow from (subtraction) the high order nibble. It is cleared to logic 0 by all other arithmetic operations.

F0: Flag 0.
This is a bit-addressable, general purpose flag for use under software control.

RS1: Register bank select bit 1.

RS0: Register bank select bit 0.

(RS1, RS0)	Working Register Bank and Address
(0, 0)	Bank 0 (00H~07H)
(0, 1)	Bank 1 (08H~0FH)
(1, 0)	Bank 2 (10H~17H)
(1, 1)	Bank 3 (18H~1FH)

OV: Overflow flag.
This bit is set to 1 under the following circumstances:
• An ADD, ADDC, or SUBB instruction causes a sign-change overflow.
• An MUL instruction results in an overflow (result is greater than 255).
• A DIV instruction causes a divide-by-zero condition.
The OV bit is cleared to 0 by the ADD, ADDC, SUBB, MUL, and DIV instructions in all other cases.

P: Parity flag.
Set/cleared by hardware each instruction cycle to indicate an odd/even number of “one” bits in the Accumulator, i.e., even parity.

(Note: PSW register is bit-addressable. All the released bits can be set and cleared by software in bit-level.)

Table 6-2. The Standard 80C51 SFRs

SYMBOL	DESCRIPTION	ADDR	BIT ADDRESS & SYMBOL								RESET VALUE
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
ACC*	Accumulator	**E0H**	**E7H** ACC.7	**E6H** ACC.6	**E5H** ACC.5	**E4H** ACC.4	**E3H** ACC.3	**E2H** ACC.2	**E1H** ACC.1	**E0H** ACC.0	00H
B*	B Register	**F0H**	**F7H** B.7	**F6H** B.6	**F5H** B.5	**F4H** B.4	**F3H** B.3	**F2H** B.2	**F1H** B.1	**F0H** B.0	00H
PSW*	Program Status Word	**D0H**	**D7H** CY	**D6H** AC	**D5H** F0	**D4H** RS1	**D3H** RS0	**D2H** OV	**D1H** -	**D0H** P	00H
SP	Stack Pointer	**81H**									07H
DPH	Data Pointer High	**83H**									00H
DPL	Data Pointer Low	**82H**									00H
P0*	Port 0	**80H**	**87H** P0.7	**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	**90H** P1.0	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	**B6H** P3.6	**B5H** P3.5	**B4H** P3.4	**B3H** P3.3	**B2H** P3.2	**B1H** P3.1	**B0H** P3.0	FFH
IP*	Interrupt Priority	**B8H**	**BFH** -	**BEH** -	**BDH** PT2	**BCH** PS	**BBH** PT1	**BAH** PX1	**B9H** PT0	**B8H** PX0	00H
IE*	Interrupt Enable	**A8H**	**AFH** EA	**AEH** -	**ADH** ET2	**ACH** ES	**ABH** ET1	**AAH** EX1	**A9H** ET0	**A8H** EX0	00H
TMOD	Timer Mode	**89H**	GATE	C/-T	M1	M0	GATE	C/-T	M1	M0	00H
TCON*	Timer Control	**88H**	**8FH** TF1	**8EH** TR1	**8DH** TF0	**8CH** TR0	**8BH** IE1	**8AH** IT1	**89H** IE0	**88H** IT0	00H
T2CON*	Timer 2 Control	**C8H**	**CFH** TF2	**CEH** EXF2	**CDH** RCLK	**CCH** TCLK	**CBH** EXEN2	**CAH** TR2	**C9H** C/-T2	**C8H** CP/-RL2	00H
TH0	Timer 0, High-byte	**8CH**									00H
TL0	Timer 0, Low-byte	**8AH**									00H
TH1	Timer 1, High-byte	**8DH**									00H
TL1	Timer 1, Low-byte	**8BH**									00H
TH2	Timer 2, High-byte	**CDH**									00H
TL2	Timer 2, Low-byte	**CCH**									00H
RCAP2H	Timer 2 Capture, High	**CBH**									00H
RCAP2L	Timer 2 Capture, Low	**CAH**									00H
SCON*	Serial Port Control	**98H**	**9FH** SM0/FE	**9EH** SM1	**9DH** SM2	**9CH** REN	**9BH** TB8	**9AH** RB8	**99H** TI	**98H** RI	00H
SBUF	Serial Data Buffer	**99H**									xxH
PCON	Power Control	**87H**	SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL	10H#

Notes:

*: Bit addressable

-: Reserved bit

#: Reset value depends on reset source.

6.2.2 The New Added SFRs

The new added SFRs are shown in Table 6-3. More information on the use of these new SFRs is included in the description of those peripherals.

Table 6-3. The New Added SFRs

SYMBOL	DESCRIPTION	ADDR	BIT ADDRESS & SYMBOL								RESET VALUE
			Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	
Interrupt											
XICON*	External Interrupt Control	**C0H**	**C7H** PX3	**C6H** EX3	**C5H** IE3	**C4H** IT3	**C3H** PX2	**C2H** EX2	**C1H** IE2	**C0H** IT2	00H
IPH	Interrupt Priority High	**B7H**	PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	00H
AUXIE	Auxiliary Interrupt Enable	**ADH**	-	-	EKB	ES2	EBD	EPCA	EADC	ESPI	00H
AUXIP	Auxiliary Interrupt Priority	**AEH**	-	-	PKB	PS2	PBD	PPCA	PADC	PSPI	00H
AUXIPH	Auxiliary Interrupt Priority High	**AFH**	-	-	PKBH	PS2H	PBDH	PPCAH	PADCH	PSPIH	00H
I/O Port											
P4*	Port 4	**E8H**	**EFH** P4.7	**EEH** P4.6	**EDH** P4.5	**ECH** P4.4	**EBH** P4.3	**EAH** P4.2	**E9H** P4.1	**E8H** P4.0	FFH
P0M0	Port 0 Mode Register 0	**93H**	P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0	00H
P0M1	Port 0 Mode Register 1	**94H**	P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0	00H
P1M0	Port 1 Mode Register 0	**91H**	P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0	00H
P1M1	Port 1 Mode Register 1	**92H**	P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0	00H
P2M0	Port 2 Mode Register 0	**95H**	P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0	00H
P2M1	Port 2 Mode Register 1	**96H**	P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0	00H
P3M0	Port 3 Mode Register 0	**B1H**	P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0	00H
P3M1	Port 3 Mode Register 1	**B2H**	P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0	00H
P4M0	Port 4 Mode Register 0	**B3H**	P4M0.7	P4M0.6	P4M0.5	P4M0.4	P4M0.3	P4M0.2	P4M0.1	P4M0.0	00H
P4M1	Port 4 Mode Register 1	**B4H**	P4M1.7	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0	00H
Keypad Interrupt											
KBCON	Keypad Control	**D6H**	-	-	-	-	-	-	PATNS	KBIF	00H
KBPATN	Keypad Pattern	**D5H**	KBPATN.7	KBPATN.6	KBPATN.5	KBPATN.4	KBPATN.3	KBPATN.2	KBPATN.1	KBPATN.0	FFH
KBMASK	Keypad Mask	**D7H**	KBMASK.7	KBMASK.6	KBMASK.5	KBMASK.4	KBMASK.3	KBMASK.2	KBMASK.1	KBMASK.0	00H
Serial Port											
SADEN	Slave Address Mask	**B9H**	SADEN.7	SADEN.6	SADEN.5	SADEN.4	SADEN.3	SADEN.2	SADEN.1	SADEN.0	00H
SADDR	Slave Address	**A9H**	SADDR.7	SADDR.6	SADDR.5	SADDR.4	SADDR.3	SADDR.2	SADDR.1	SADDR.0	00H
S2CON	UART2 Control	**AAH**	S2SM0	S2SM1	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	00H
S2BRT	UART2 Baud Rate Timer	**BAH**									00H
S2BUF	UART2 Serial Buffer	**9AH**									xxH
ADC											
ADCTL	ADC Control Register	**C5H**	ADCON	SPEED1	SPEED0	ADCI	ADCS	CHS2	CHS1	CHS0	00H
ADCH	ADC Result , High-byte	**C6H**									xxH
ADCL	ADC Result , Low-byte	**BEH**									xxH
SPI											
SPCTL	SPI Control Register	**85H**	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	04H
SPSTAT	SPI Status Register	**84H**	SPIF	WCOL	-	-	-	-	-	-	00H
SPDAT	SPI Data Register	**86H**									00H

(Continued)

PCA											
CCON*	PCA Counter Control	**D8H**	**DFH** CF	**DEH** CR	**DDH** CCF5	**DCH** CCF4	**DBH** CCF3	**DAH** CCF2	**D9H** CCF1	**D8H** CCF0	00H
CMOD	PCA Counter Mode	**D9H**	CIDL	-	-	-	-	CPS2	CPS1	ECF	00H
CH	PCA Counter, HB	**F9H**									00H
CL	PCA Counter, LB	**E9H**									00H
CCAPM0	PCA Module0 Compare/Capture Reg.	**DAH**	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	00H
CCAPM1	PCA Module1 Compare/Capture Reg.	**DBH**	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	00H
CCAPM2	PCA Module2 Compare/Capture Reg.	**DCH**	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	00H
CCAPM3	PCA Module3 Compare/Capture Reg.	**DDH**	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	00H
CCAPM4	PCA Module4 Compare/Capture Reg.	**DEH**	-	ECOM4	CAPP4	CAPN4	MAT4	TOG4	PWM4	ECCF4	00H
CCAPM5	PCA Module5 Compare/Capture Reg.	**DFH**	-	ECOM5	CAPP5	CAPN5	MAT5	TOG5	PWM5	ECCF5	00H
CCAP0H	PCA Module0 Capture Register, High-Byte	**FAH**									00H
CCAP0L	PCA Module0 Capture Register, Low-Byte	**EAH**									00H
CCAP1H	PCA Module1 Capture Register High-Byte	**FBH**									00H
CCAP1L	PCA Module1 Capture Register, Low-Byte	**EBH**									00H
CCAP2H	PCA Module2 Capture Register, High-Byte	**FCH**									00H
CCAP2L	PCA Module2 Capture Register, Low-Byte	**ECH**									00H
CCAP3H	PCA Module3 Capture Register, High-Byte	**FDH**									00H
CCAP3L	PCA Module3 Capture Register, Low-Byte	**EDH**									00H
CCAP4H	PCA Module4 Capture Register, High-Byte	**FEH**									00H
CCAP4L	PCA Module4 Capture Register, Low-Byte	**EEH**									00H
CCAP5H	PCA Module5 Capture Register, High-Byte	**FFH**									00H
CCAP5L	PCA Module5 Capture Register, Low-Byte	**EFH**									00H
PCAPWM 0	PCA PWM Mode, Auxiliary Register 0	**F2H**	-	-	-	-	-	-	ECAP0H	ECAP0L	00H
PCAPWM 1	PCA PWM Mode, Auxiliary Register 1	**F3H**	-	-	-	-	-	-	ECAP1H	ECAP1L	00H
PCAPWM 2	PCA PWM Mode, Auxiliary Register 2	**F4H**	-	-	-	-	-	-	ECAP2H	ECAP2L	00H
PCAPWM 3	PCA PWM Mode, Auxiliary Register 3	**F5H**	-	-	-	-	-	-	ECAP3H	ECAP3L	00H
PCAPWM 4	PCA PWM Mode, Auxiliary Register 4	**F6H**	-	-	-	-	-	-	ECAP4H	ECAP4L	00H
PCAPWM 5	PCA PWM Mode, Auxiliary Register 5	**F7H**	-	-	-	-	-	-	ECAP5H	ECAP5L	00H
Others											
AUXR	Auxiliary Register	**8EH**	URTS	ADRJ	P41ALE	P35ALE	-	-	EXTRAM	-	00H
AUXR1	Auxiliary Register 1	**A2H**	P4KB	P4PCA	P4SPI	P4S2	GF2	-	-	DPS	00H
AUXR2	Auxiliary Register 2	**A6H**	T0X12	T1X12	URM0X6	S2TR	S2SMOD	S2TX12	S2CKOE	T0CKOE	00H
T2MOD	Timer 2 Mode Control	**C9H**	-	-	-	-	-	-	T2OE	DCEN	00H
STRETCH	External Access Stretch	**8FH**	-	-	ALES1	ALES0	-	RWS2	RWS1	RWS0	23H
PCON2	Power Control 2	**C7H**	-	-	-	-	-	SCKD2	SCKD1	SCKD0	00H
WDTCR	Watch-dog Timer	**E1H**	WRF	-	ENW	CLRW	WIDL	PS2	PS1	PS0	00H#
EVRCR	EVR Control Register	**97H**	EOPFI	ECPFI	OPF	CPF	PMUOFF	-	-	-	30H#

(Continued)

ISP											
ISPCR	ISP Control Register	**E7H**	ISPEN	SWBS	SWRST	-	-	CKS2	CKS1	CKS0	00H
IFMT	ISP Mode Select	**E5H**	-	-	-	-	-	-	MS1	MS0	63H
IFADRH	ISP Flash Address, High-byte	**E3H**									00H
IFADRL	ISP Flash Address, Low-byte	**E4H**									00H
IFD	ISP Flash Data	**E2H**									FFH
SCMD	ISP Sequential Command	**E6H**									xxH

Notes:

*: Bit addressable

-: Reserved bit

#: Reset value depends on reset source.

7 On-Chip eXpanded RAM (XRAM)

To access the on-chip expanded RAM (XRAM), the EXTRAM bit should be cleared to 0. The 1024 bytes of XRAM (with addresses 0000H to 03FFH) are indirectly accessed by move external instruction, “MOVX @DPTR”. An access to XRAM will have not any outputting of address, address latch enable and read/write strobe. That means P0, P2, ALE (P3.5 or P4.1), /WR (P3.6) and /RD (P3.7) will keep unchanged during access of XRAM. However, if the address is more than 0x03FF, the access will be automatically switched to the external data memory.

AUXR (Address=8EH, Auxiliary Register, Reset Value=0000,xx0xB)

7	6	5	4	3	2	1	0
URTS	ADRJ	P41ALE	P35ALE	-	-	EXTRAM	-

EXTRAM:

0: Disable accessing to external data memory while address less than 0x0400;
Accessing of addresses 0x0000~0x03FF are automatically switched to on-chip XRAM.

1: Enable accessing to whole external data memory with addresses 0x0000~0xFFFF;
Accessing of on-chip XRAM is disabled.

7.1 Using the XRAM in Software

For Keil-C51 compiler, to assign the variables to be located at XRAM, the “xdata” declaration should be used. After being compiled, the variables declared by “xdata” will become the memories accessed by “MOVX @DPTR”. The user can get the following descriptions from the “*Keil Software — Cx51 Compiler User’s Guide*”.

Table 7-1. Declaration of XRAM Memory Type

Explicitly Declared Memory Types

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

The following table summarizes the available memory type specifiers.

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

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

Example:

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

8 External Data Memory Accessing

As described in Section 5.2, to access the external data memory, the EXTRAM bit should be set to 1. Accesses to external data memory can use either a 16-bit address (using 'MOVX @DPTR') or an 8-bit address (using 'MOVX @Ri'), as described below.

Accessing by an 8-bit address

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

Accessing by a 16-bit address

16-bit addresses are often used to access up to 64k bytes of external data memory. Figure 5-6 shows the hardware configuration for accessing 64K bytes of external RAM. Whenever a 16-bit address is used, in addition to the functioning of P0, /RD and /WR, the high byte of the address comes out on Port 2 and it is held during the read or write cycle.

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

8.1 ALE-Pin Configuration

For the MPC82G516A, there is no dedicated pin for ALE signal. The ALE becomes an alternate function of P3.5 or P4.1, which can be selected by control bits P35ALE and P41ALE in the AUXR register, as shown below. And, although an 80C51 MCU always outputs the ALE signal even there in no external accessing, the device doesn't output any ALE signal except when accessing the external data memory (EXTRAM=1).

AUXR (Address=8EH, Auxiliary Register, Reset Value=0000, xx0xB)

7	6	5	4	3	2	1	0
URTS	ADRJ	P41ALE	P35ALE	-	-	EXTRAM	-

P41ALE: when set, P4.1 functions as the ALE pin for external MOVX accessing.

P35ALE: when set, P3.5 functions as the ALE pin for external MOVX accessing.

EXTRAM:

0: Disable accessing to external data memory while address less than 0x0400;
Accessing of addresses 0x0000~0x03FF are automatically switched to on-chip XRAM.

1: Enable accessing to whole external data memory with addresses 0x0000~0xFFFF;
Accessing of on-chip XRAM is disabled.

8.2 Access Timing Stretching for Low-speed Memory

To access the low-speed external data memory, the timing-stretch mechanism is designed to control the access timing of the “MOVX” instructions. The bits ALES1 and ALES0, in STRETCH register, control the stretching of the setup time and hold time with respect to ALE negative edge. And, the bits RWS2, RWS1 and RWS0 control the stretching of the read/write pulse width. The user should configure STRETCH register properly to conform to the read/write requirements of the external data memory being used.

STRETCH (Address=8FH, Stretch Register, Reset Value=0010,0011B)

7	6	5	4	3	2	1	0
-	-	ALES1	ALES0	-	RWS2	RWS1	RWS0

{ALES1,ALES0} :
00: No stretch, the P0’s address setup/hold time to the following ALE falling edge is 1 clock cycle.
01: 1 clock stretched, the P0’s address setup/hold time to the following ALE falling edge is 2 clock cycles.
10: 2 clocks stretched, the P0’s address setup/hold time to the following ALE falling edge is 3 clock cycles.
11: 3 clocks stretched, the P0’s address setup/hold time to the following ALE falling edge is 4 clock cycles.

{RWS2,RWS1,RWS0} :
000: No stretch, the MOVX read/write pulse is 1 clock cycle.
001: 1 clock stretched, the MOVX read/write pulse is 2 clock cycles.
010: 2 clocks stretched, the MOVX read/write pulse is 3 clock cycles.
011: 3 clocks stretched, the MOVX read/write pulse is 4 clock cycles.
100: 4 clocks stretched, the MOVX read/write pulse is 5 clock cycles.
101: 5 clocks stretched, the MOVX read/write pulse is 6 clock cycles.
110: 6 clocks stretched, the MOVX read/write pulse is 7 clock cycles.
111: 7 clocks stretched, the MOVX read/write pulse is 8 clock cycles.

See the following timing waveforms for demonstration.

Figure 8-1. “MOVX @DPTR,A” without Stretch

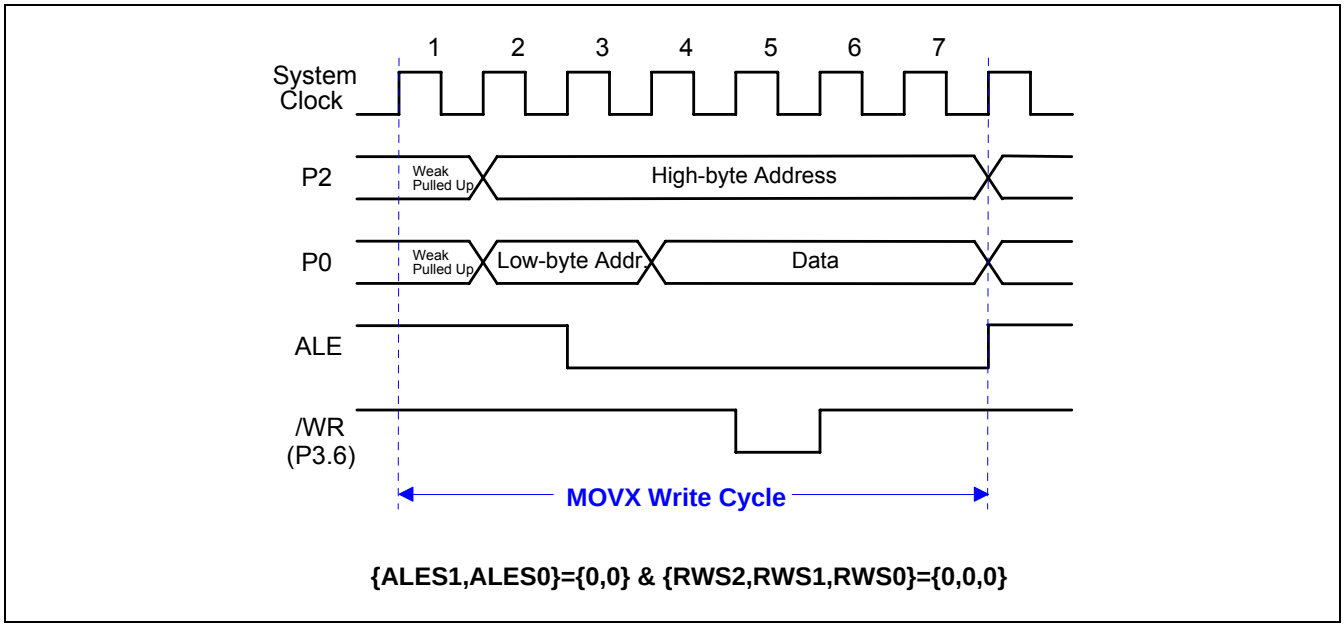


Figure 8-2. “MOVX @DPTR,A” with Stretch

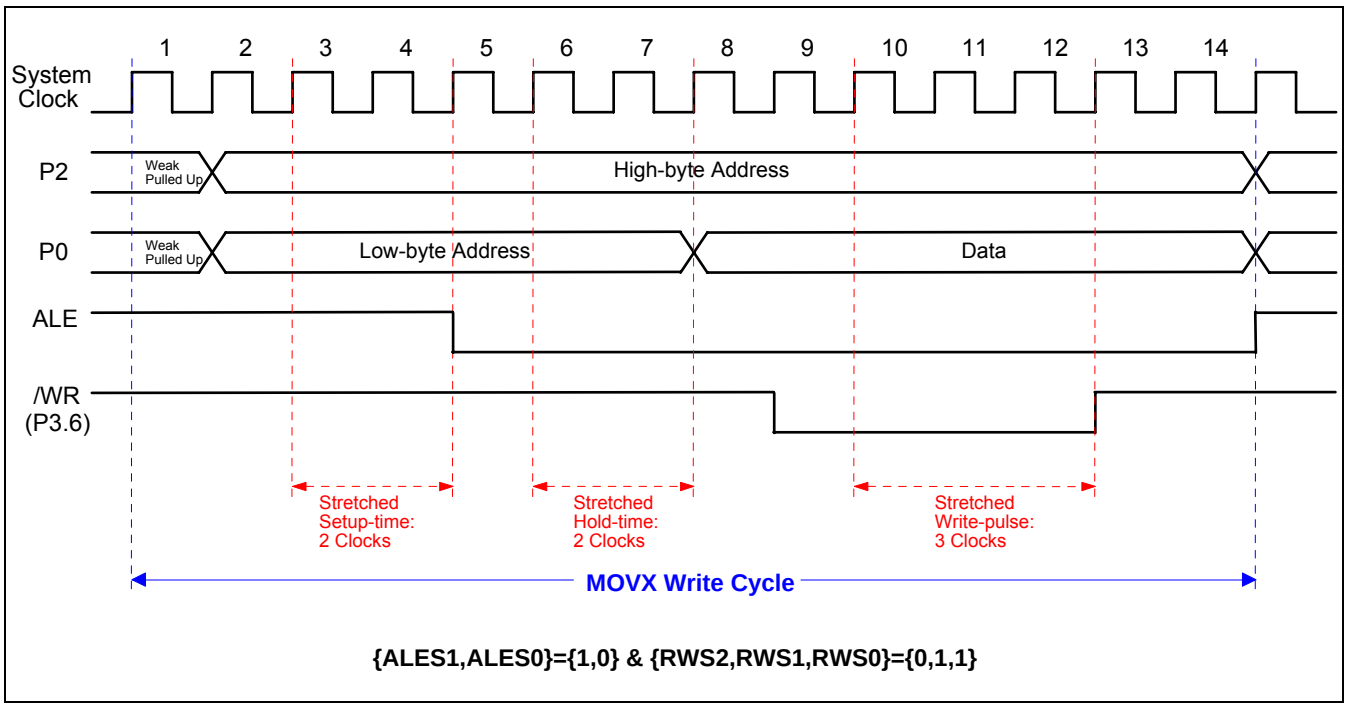


Figure 8-3. “MOVX A,@DPTR” without Stretch

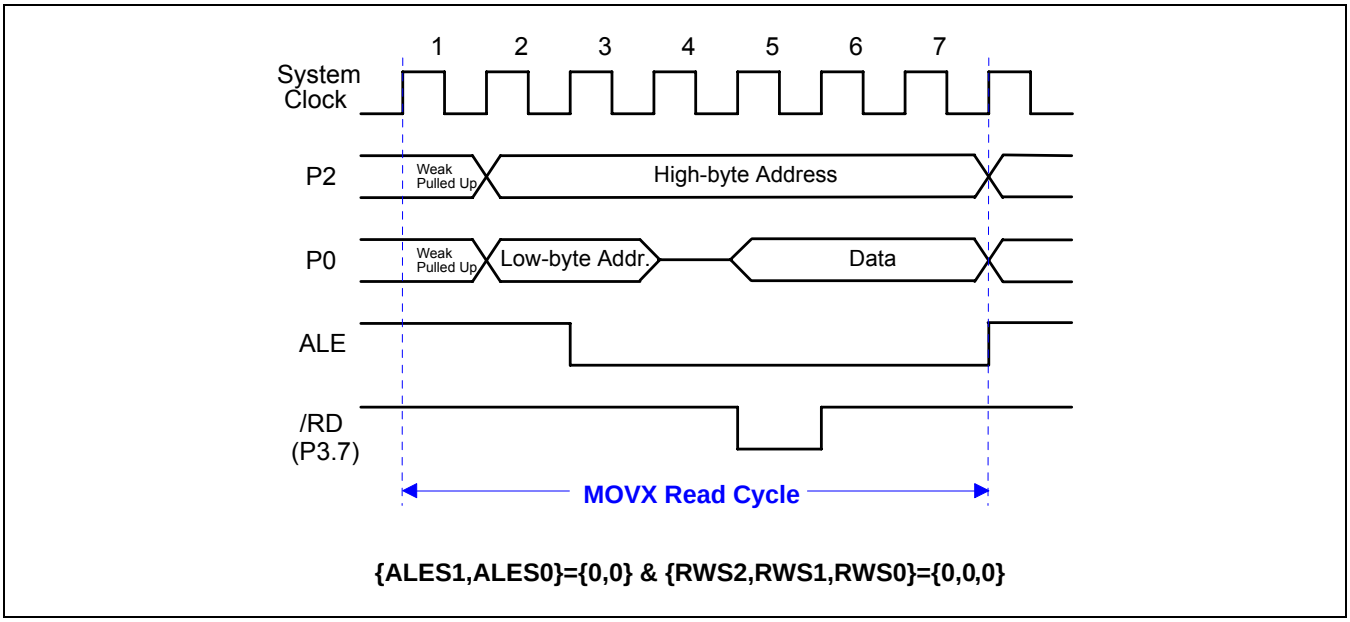
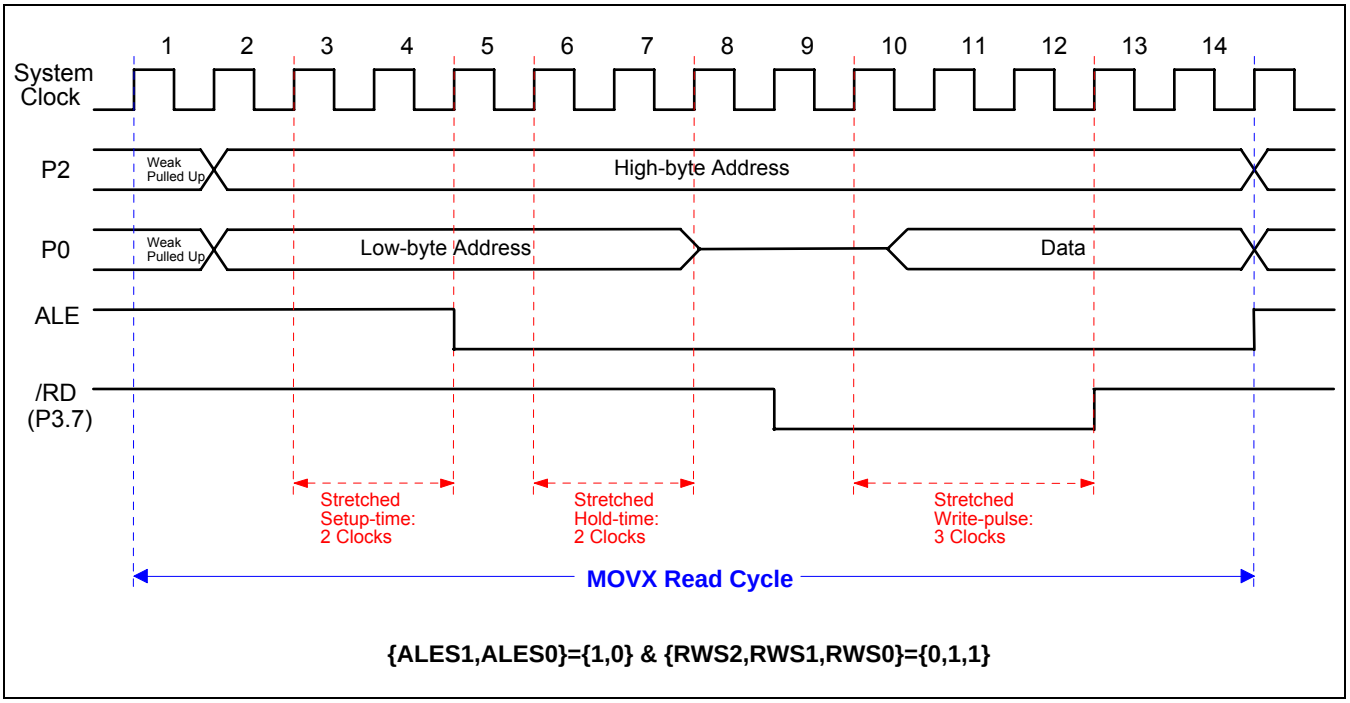


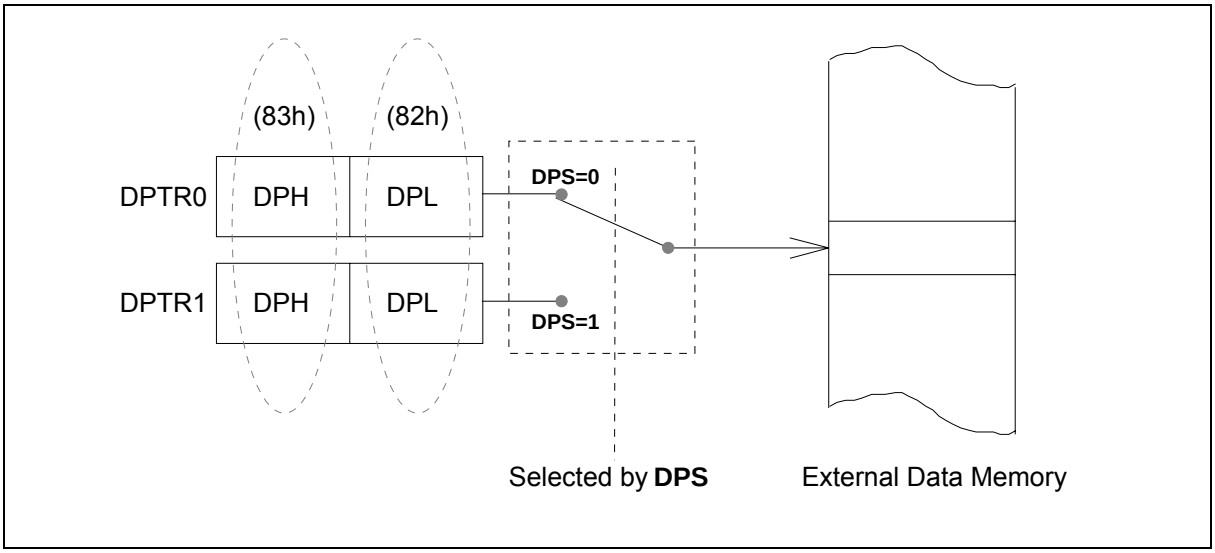
Figure 8-4. “MOVX A,@DPTR” with Stretch



9 Dual Data Pointer Register (DPTR)

The additional data pointer can be used to speed up code execution and reduce code size. The dual DPTR structure is a way by which the chip will specify the address of an external data memory location. There are two 16-bit DPTR registers that address the external memory, and a single control bit called DPS (AUXR1.0) that allows the program code to switch between the external memory.

Figure 9-1. Use of Dual DPTR



DPTR Instructions

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

```
INC  DPTR          ;Increments the data pointer by 1
MOV  DPTR,#data16 ;Loads the DPTR with a 16-bit constant
MOVC 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 (Address=8EH, Auxiliary Register1, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P4KB	P4PCA	P4SPI	P4S2	GF2	-	-	DPS

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

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

DPS	DPTR selected
0	DPTR0
1	DPTR1

10 I/O Port Structure and Operation

The MPC82G516A has five I/O ports: Port 0, Port 1, Port2, Port 3 and Port4. All ports are 8-bit ports. The exact number of I/O pins available depends upon the package types. See Table 10-1.

Table 10-1. Number of I/O Pins Available

Package Type	I/O Pins	Number of I/O Pins
40-pin DIP	P0, P1, P2, P3	32
28-pin SSOP	P0.1~P0.4, P0.6, P1.0, P1.2, P1.3, P1.5, P2.0, P2.1, P2.3, P2.4, P2.5 P3.0~P3.3, P3.5 P4.2	20
44-pin PLCC	P0, P1, P2, P3, P4.0~P4.3	36
44-pin PQFP	P0, P1, P2, P3, P4.0~P4.3	36
48-pin LQFP	P0, P1, P2, P3, P4	40

10.1 Port Configurations

All I/O port pins on the MPC82G516A may be individually and independently configured by software to one of four types on a bit-by-bit basis, as shown in Table 10-2. These are: quasi-bidirectional (standard 8051 I/O port), push-pull output, open-drain output, and input-only (high-impedance input). Two mode registers for each port select the output type for each port pin.

Table 10-2. Port Configuration Settings

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

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

P0M0 (Address=93H, Port 0 Mode Register 0, Reset Value=0000,0000B)

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

P0M1 (Address=94H, Port 0 Mode Register 1, Reset Value=0000,0000B)

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

P1M0 (Address=91H, Port 1 Mode Register 0, Reset Value=0000,0000B)

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

P1M1 (Address=92H, Port 1 Mode Register 1, Reset Value=0000,0000B)

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

P2M0 (Address=95H, Port 2 Mode Register 0, Reset Value=0000,0000B)

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

P2M1 (Address=96H, Port 2 Mode Register 1, Reset Value=0000,0000B)

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

P3M0 (Address=B1H, Port 3 Mode Register 0, Reset Value=0000,0000B)

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

P3M1 (Address=B2H, Port 3 Mode Register 1, Reset Value=0000,0000B)

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

P4M0 (Address=B3H, Port 4 Mode Register 0, Reset Value=0000,0000B)

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

P4M1 (Address=B4H, Port 4 Mode Register 1, Reset Value=0000,0000B)

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

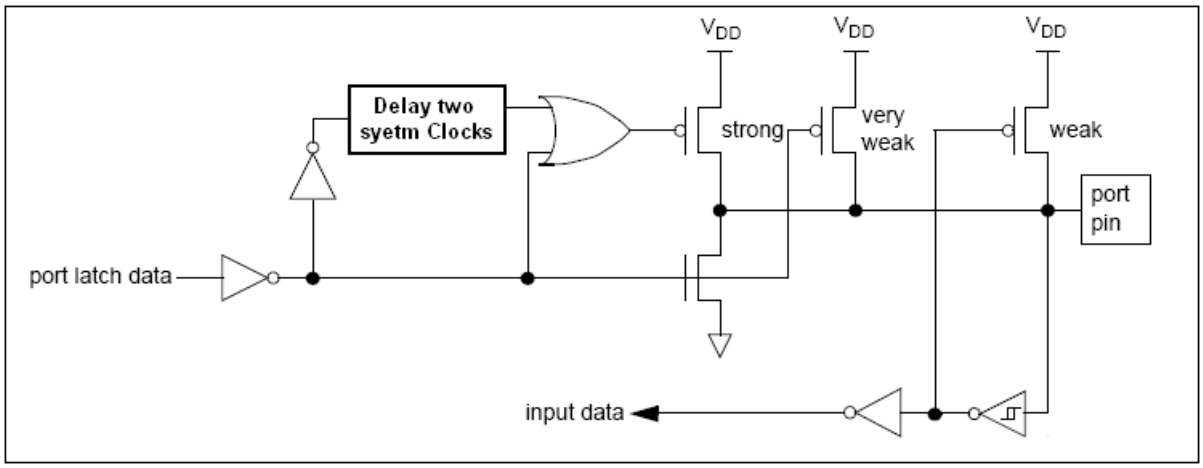
10.1.1 Quasi-Bidirectional I/O

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.

The quasi-bidirectional port configuration is shown in Figure 10-1. A quasi-bidirectional port pin has a Schmitt-triggered input for noise suppression.

Figure 10-1. Quasi-Bidirectional I/O

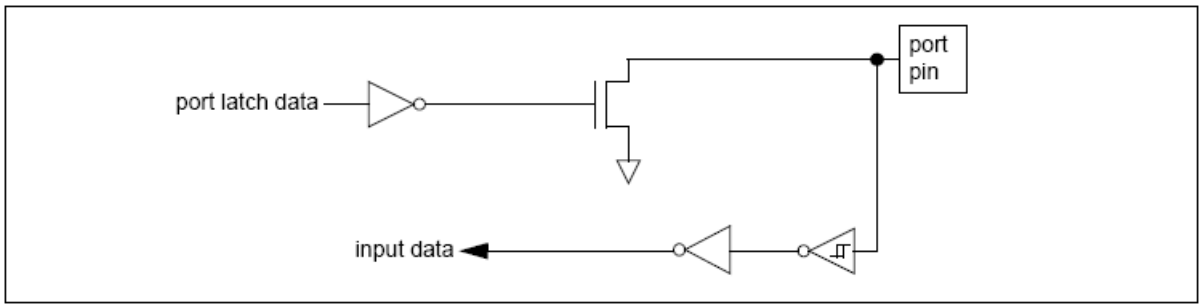


10.1.2 Open-Drain Output

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.

The open-drain port configuration is shown in Figure 10-2. An open drain port pin also has a Schmitt-triggered input for noise suppression.

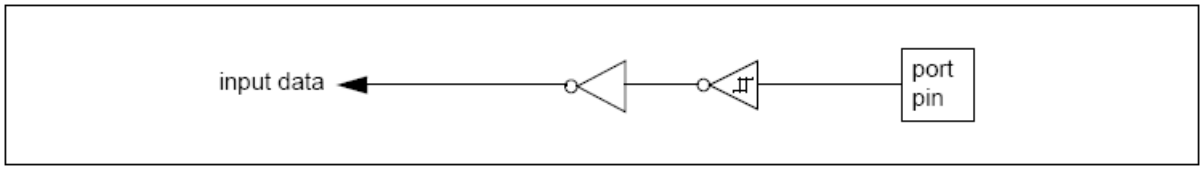
Figure 10-2. Open-Drain Output



10.1.3 Input-Only (High Impedance Input)

The input-only configuration is a Schmitt-triggered input without any pull-up resistors on the pin, as shown in Figure 10-3.

Figure 10-3. Input-Only

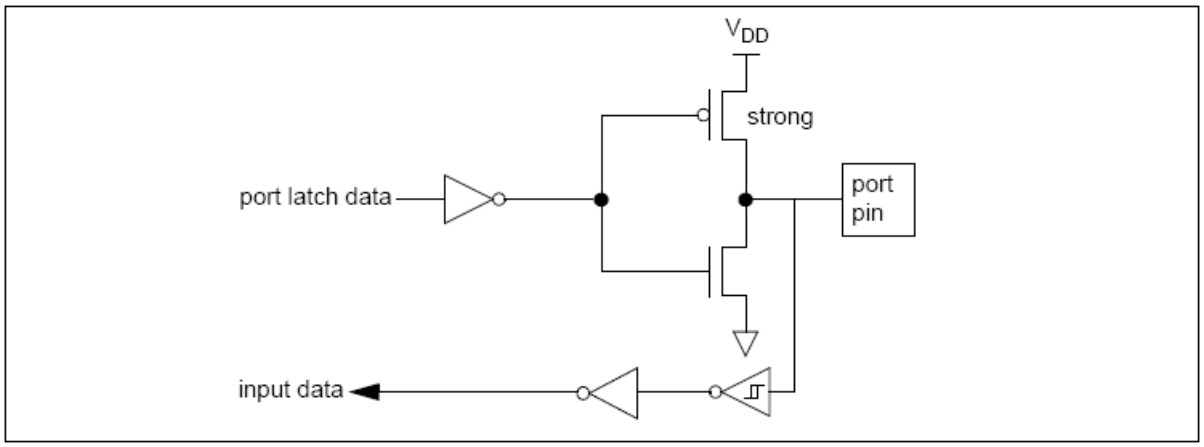


10.1.4 Push-Pull Output

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

The push-pull port configuration is shown in Figure 10-4. A push-pull port pin also has a Schmitt-triggered input for noise suppression.

Figure 10-4. Push-Pull Output



10.2 I/O Pins Used with ADC Function

The Port 1 serves as the alternate function of the ADC analog input. In order to get the best analog performance, the pins that are being used with the ADC should have their digital outputs disabled. This can be achieved by putting the port pins into the Input-Only mode.

10.3 Additional Note for I/O Port

Every output on the MPC82G516A has been designed to sink typical LED drive current. However, there is a maximum total output current for all ports which must not be exceeded. Please refer to Section 29: Absolute Maximum Ratings.

11 Timers/Counters

The MPC82G516A has three 16-bit Timer/Counters: Timer 0, Timer 1 and Timer 2. Each consists of two 8-bit registers, THx and TLx (where, x= 0, 1, or 2). All of them can be configured to operate either as timers or event counters.

In the Timer function, the TLx register is incremented every 12-clock cycle or 1-clock cycle, which is selectable by software. Thus one can think of it as counting clock cycles. When counting every 12 clock cycles, the count rate is 1/12 of the oscillator frequency.

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin- T0, T1, or T2. In this function, the external input is sampled every clock cycle for T0 pin and T1 pin, and 12-clock cycle for T2 pin. When the samples show a high and then a low, the count is incremented. The new count value appears in the register when the transition was detected. For Timer 0 and Timer1, it takes 2 clock cycles to recognize a 1-to-0 transition, the maximum count rate is 1/2 of the oscillator frequency; for Timer 2, it takes 24 clock cycles to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one clock cycle for Timer 0 and Timer 1, and 12-clock cycles for Timer 2.

For Timer 0 and Timer 2, in addition to their standard 8051's timer function, some special new functions are added in. The following sub-sections will describe these timer/counters in detail.

11.1 Timer 0 and Timer 1

The Timer or Counter function is selected by control bits C/-T in the Special Function Register TMOD, as shown below. These two Timer/Counters have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. Mode 0, 1 and 2 are the same for these two Timer/Counters. Mode 3 is different. In addition to TMOD, another Special Function Registers TCON and AUXR2 contains several control bits and status flags related to these two Timers, as also shown below.

TMOD (Address=89H, Timer/Counter Mode Control Register, Reset Value=0000,0000B)

Timer 1				Timer 0			
7	6	5	4	3	2	1	0
GATE	C/-T	M1	M0	GATE	C/-T	M1	M0

GATE: Gating control when set. Timer/Counter 0 or 1 is enabled only while /INT0 or /INT1 pin is high and TR0 or TR1 control pin is set. When cleared, Timer 0 or 1 is enabled whenever TR0 or TR1 control bit is set.

C/-T: Timer or Counter Selector. Clear for Timer operation (input from internal system clock). Set for Counter operation (input from T0 or T1 input pin).

M1	M0	Operating Mode
0	0	8-bit Timer/Counter. THx with TLx as 5-bit prescaler.
0	1	16-bit Timer/Counter. THx and TLx are cascaded; there is no prescaler.
1	0	8-bit auto-reload Timer/Counter. THx holds a value which is to be reloaded into TLx each time it overflows.
1	1	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer/Counter stopped.

TCON (Address=88H, Timer/Counter Control Register, Reset Value=0000,0000B)

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

TF1: Timer 1 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.

TR1: Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter 1 on/off.

TF0: Timer 0 overflow Flag. Set by hardware on Timer/Counter 0 overflow. Cleared by hardware when processor vectors to interrupt routine.

TR0: Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter 0 on/off.

AUXR2 (Address=A6H, Auxiliary Register2, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	S2TR	S2SMOD	S2TX12	S2CKOE	T0CKOE

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

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

T0CKOE: Set/clear to enable/disable Timer 0 clock-out function from P3.4.

The four operating modes are described in the following text.

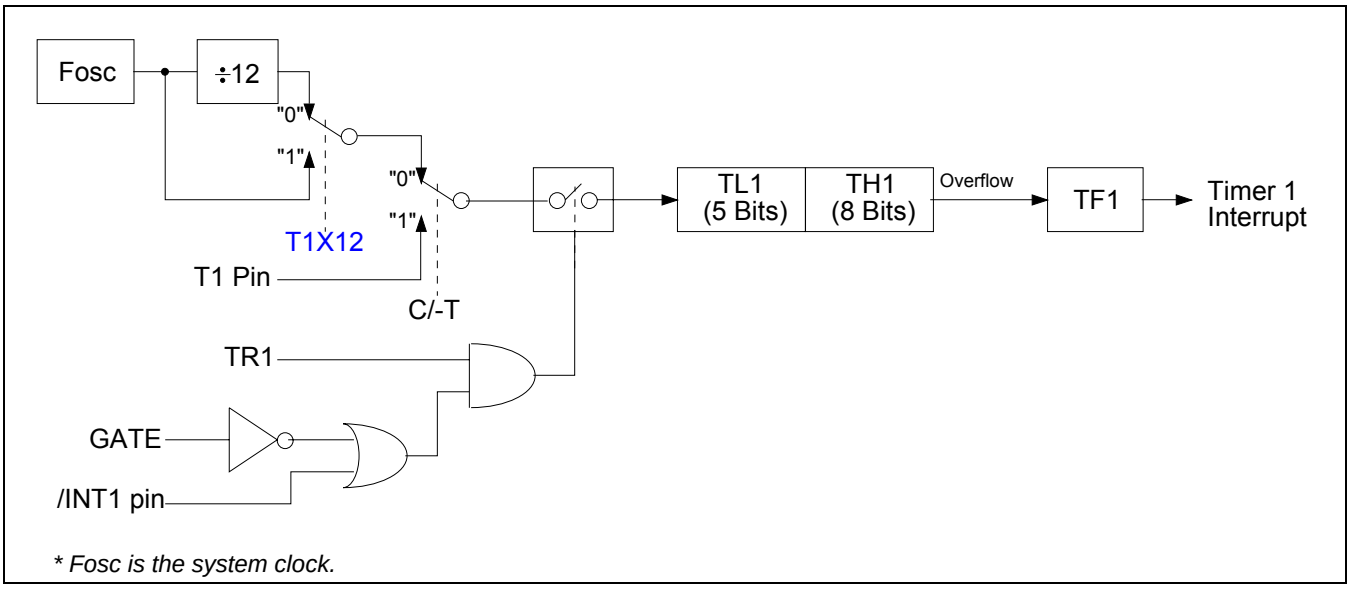
11.1.1 Mode 0: 13-Bit Timer/Counter

Timer 0 and Timer 1 in Mode 0 look like an 8-bit Counter with a divide-by-32 prescaler. And, Mode 0 operation is the same for these two timers. Figure 11-1 shows the Mode 0 operation.

In this mode, the Timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TFx. The counted input is enabled to the Timer when TRx=1 and either GATE=0 or /INTx=1. (Setting GATE=1 allows the Timer to be controlled by external input /INTx, to facilitate pulse width measurements). TRx and TFx are control bits in SFR TCON. The GATE bit is in TMOD. There are two different GATE bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3).

The 13-bit register consists of all 8 bits of THx and the lower 5 bits of TLx. The upper 3 bits of TLx are indeterminate and should be ignored. Setting the run flag (TRx) does not clear these registers. That is to say the user should initialize THx an TLx before start counting.

Figure 11-1. Timer 1 in Mode 0: 13-Bit Timer/Counter

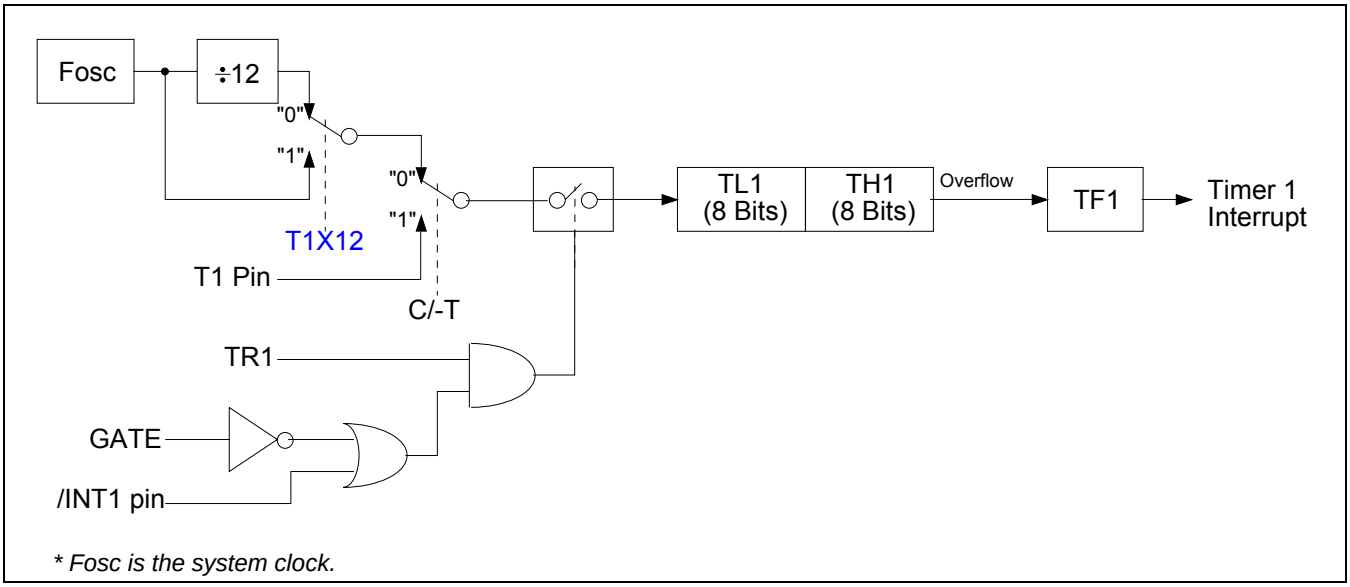


** Fosc is the system clock.*

11.1.2 Mode 1: 16-Bit Timer/Counter

Mode 1 is the same as Mode 0, except that the Timer register uses all 16 bits. Refer to Figure 11-2. In this mode, THx and TLx are cascaded; there is no prescaler.

Figure 11-2. Timer 1 in Mode 1: 16-Bit Timer/Counter

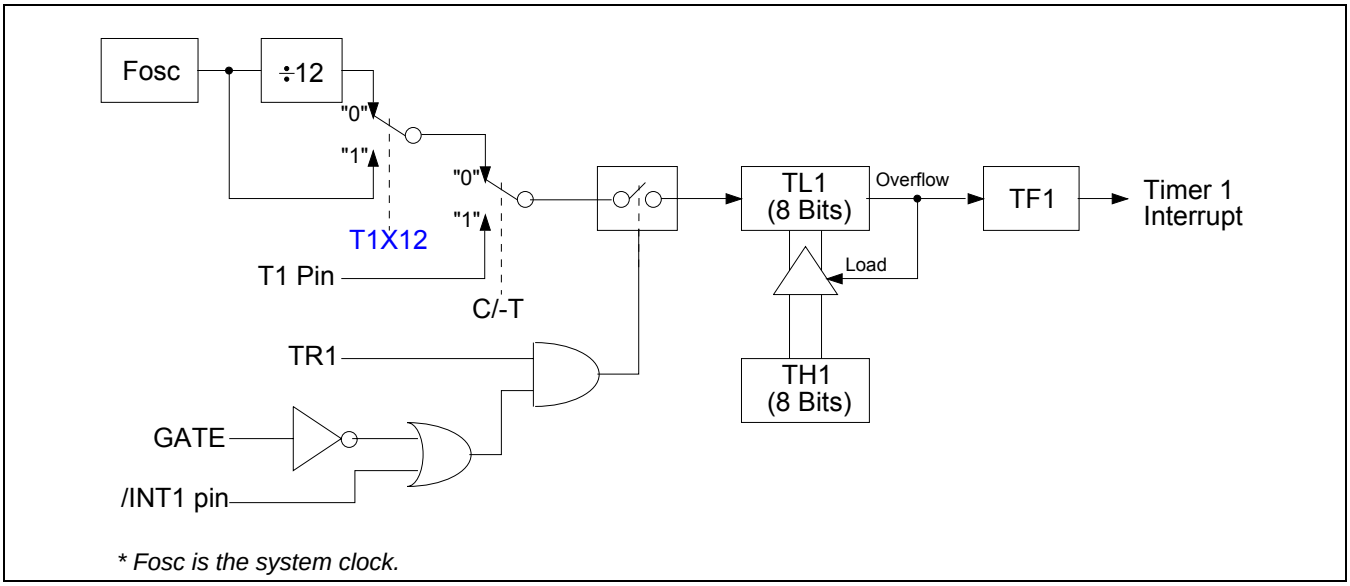


** Fosc is the system clock.*

11.1.3 Mode 2: 8-Bit Auto-Reload

Mode 2 configures the Timer register as an 8-bit Counter (TLx) with automatic reload, as shown in Figure 11-3. Overflow from TLx not only sets TFx, but also reloads TLx with the contents of THx, which is preset by software. The reload leaves THx unchanged.

Figure 11-3. Timer 1 in Mode 2: 8-Bit Auto-Reload



** Fosc is the system clock.*

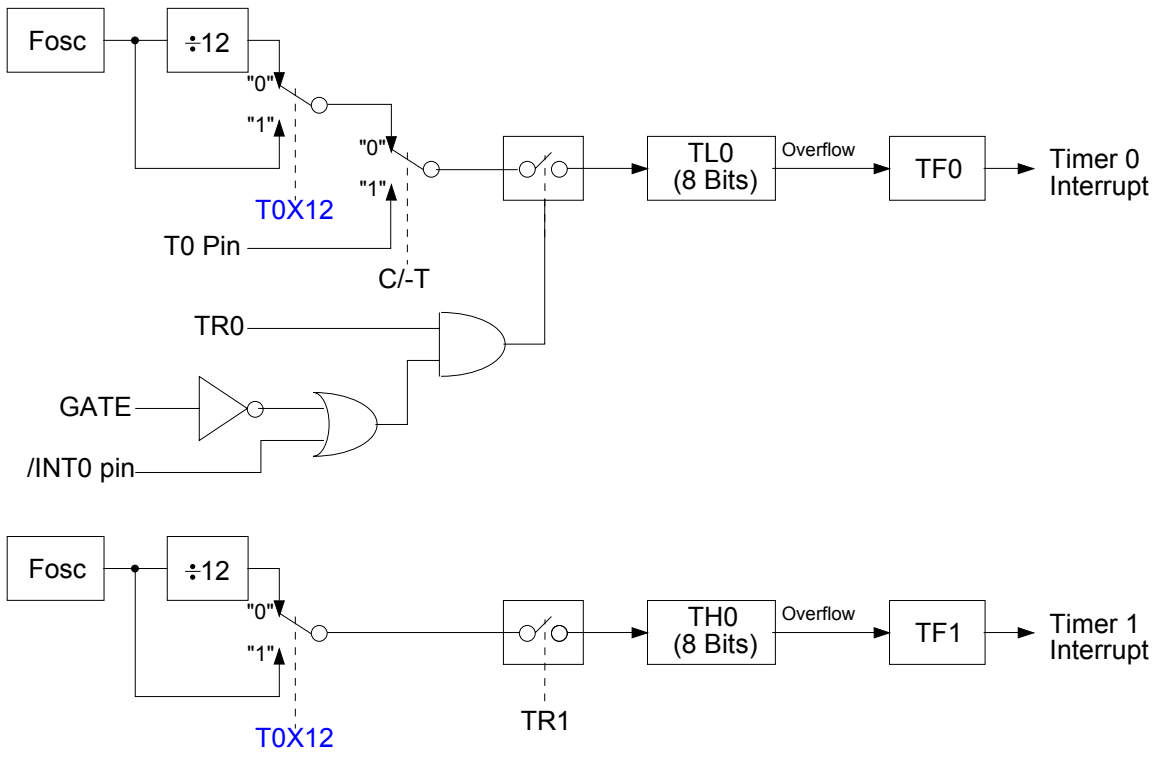
11.1.4 Mode 3: Two 8-Bit Timer/Counters

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting TR1=0.

Timer 0 in Mode 3 establishes TL0 and TH0 as two separate counters. The logic for Mode 3 on Timer 0 is shown in Figure 11-4. TL0 uses the Timer 0 control bits: C/-T, GATE, TR0, /INT0, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from Timer 1. Thus TH0 now controls the Timer 1 interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3, or can still be used by the serial port as a baud rate generator, or in any application not requiring an interrupt.

Figure 11-4. Timer 0 in Mode 3: Two 8-Bit Timer/Counters



** Fosc is the system clock.*

11.1.5 Programmable Clock-Out from Timer 0

Using Timer 0, a 50% duty cycle clock can be programmed to come out from pin T0CKO (P3.4).

The clock-out frequency depends on the system clock frequency (Fosc) and the reload value filled in the TH0 register, as shown in the following formula:

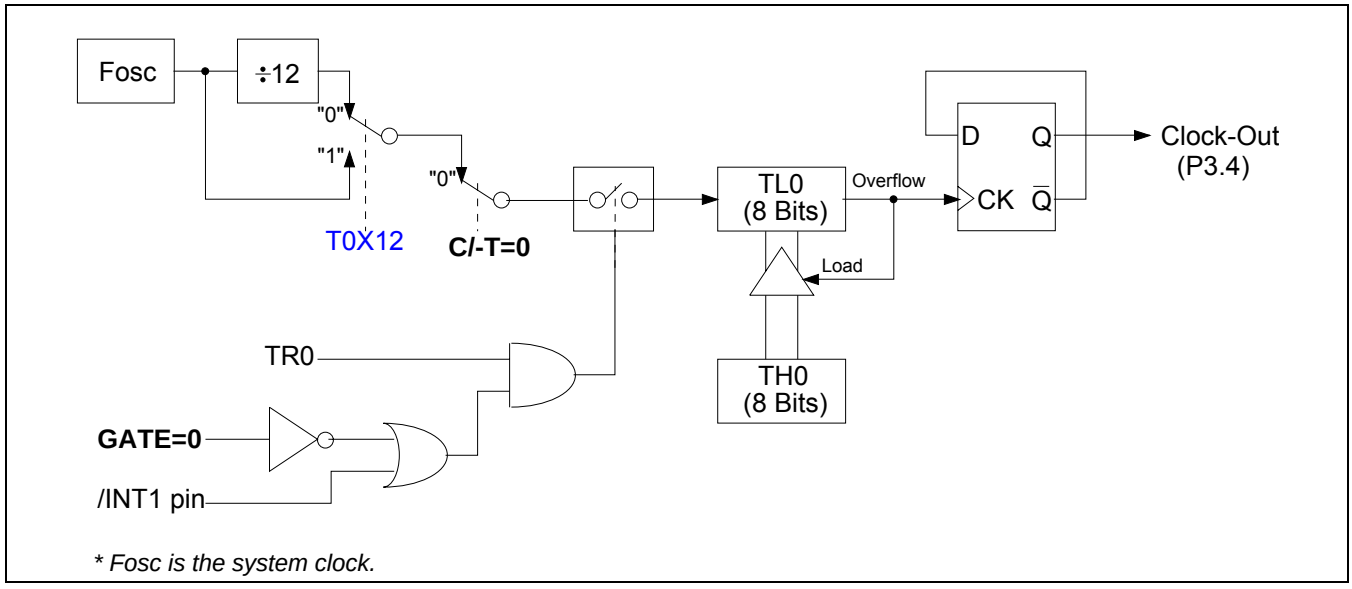
$$\text{Clock-Out Frequency} = \frac{\text{Fosc}}{n \times (256 - \text{TH0})}$$

Where,
n=24 if T0X12=0,
n=2 if T0X12=1.

Timer 0 is programmed for the clock-out mode as follows:

- Set T0CKOE bit in AUXR2 register.
- Clear C/-T bit of Timer 0 in TMOD register.
- Clear GATE bit of Timer 0 in TMOD register.
- Determine the 8-bit reload value from the formula and enter it in TH0 register.
- Enter an 8-bit initial value in TL0 register. It should be the same as the reload value.
- Start the timer by setting the run control bit TR0 in TCON register.

Figure 11-5. Programmable Clock-Out from Timer 0



** Fosc is the system clock.*

11.2 Timer 2

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

T2CON (Address=C8H, Timer/Counter 2 Control Register, Reset Value=0000,0000B)

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 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 counter mode (DCEN=1).

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

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

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

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 control bit. When set, captures will occur on negative transitions at T2EX if EXEN2=1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX 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.

T2MOD (Address=C9H, Timer 2 Mode Control Register, Reset Value=xxxx,xx00B)

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

T2OE: Timer 2 clock-out enable bit, set to enable and clear to disable.

DCEN: Timer 2 down-counting enable bit, set to enable and clear to disable.

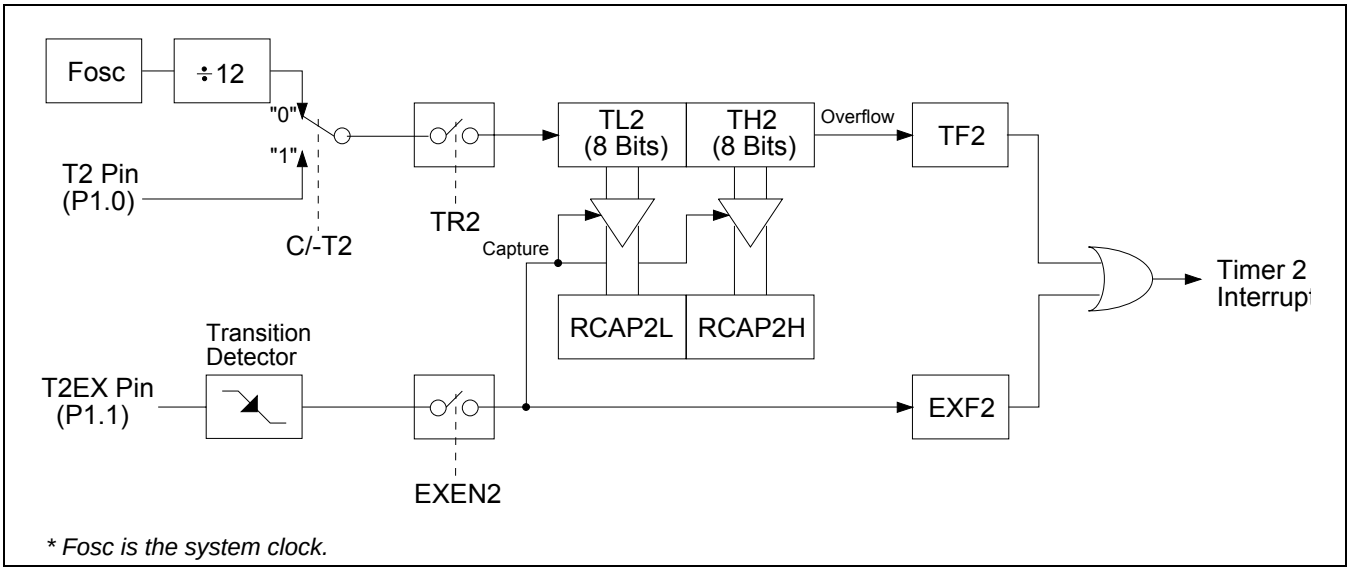
Table 11-1. Timer 2 Operating Modes

RCLK + TCLK	CP/-RL2	TR2	DCEN	T2OE	Mode
x	x	0	x	0	(Timer Off)
1	x	1	0	0	Baud-rate Generator
0	1	1	0	0	16-bit Capture
0	0	1	0	0	16-bit Auto-reload (counting-up only)
0	0	1	**1**	0	16-bit Auto-reload (counting-up or counting-down)
0	0	1	0	**1**	Programmable clock-out

11.2.1 Capture Mode

In the capture mode there are two options selected by bit EXEN2 in T2CON. If EXEN2=0, Timer 2 is a 16-bit timer or counter which, upon overflow, sets bit TF2- the Timer 2 overflow flag. This bit can then be used to generate an interrupt (by enabling the Timer 2 interrupt bit in the IE register). If EXEN2=1, Timer 2 still does the above, but with the added feature that a 1-to-0 transition at external input T2EX causes the current value in the Timer 2 registers, TH2 and TL2, to be captured into registers RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set, and the EXF2 bit (like TF2) can generate an interrupt (which vectors to the same location as Timer 2 overflow interrupt). The capture mode is illustrated in Figure 11-6. (There is no reload value for TL2 and TH2 in this mode. Even when a capture event occurs from T2EX, the counter keeps on counting T2EX pin transitions or Fosc/12 pulses).

Figure 11-6. Timer 2 in Capture Mode



** Fosc is the system clock.*

11.2.2 Auto-Reload Mode (Up or Down Counter)

In the 16-bit auto-reload mode, Timer 2 can be configured as either a timer or counter (C/-T2 in T2CON), then programmed to count up or down. The counting direction is determined by bit DCEN (Down Counter Enable) which is located in the T2MOD register. After reset, DCEN=0 which means Timer 2 will default to counting up. If DCEN is set, Timer 2 can count up or down depending on the value of the T2EX pin.

Figure 11-7 shows DCEN=0, which enables Timer 2 to count up automatically. In this mode there are two options selected by bit EXEN2 in T2CON register. If EXEN2=0, then Timer 2 counts up to 0FFFFH and sets the TF2 (Overflow Flag) bit upon overflow. This causes the Timer 2 registers to be reloaded with the 16-bit value in RCAP2L and RCAP2H. The values in RCAP2L and RCAP2H are preset by software. If EXEN2=1, then a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at input T2EX. This transition also sets the EXF2 bit. The Timer 2 interrupt, if enabled, can be generated when either TF2 or EXF2 are 1.

Figure 11-8 shows DCEN=1, which enables Timer 2 to count up or down. This mode allows pin T2EX to control the counting direction. When a logic 1 is applied at pin T2EX, Timer 2 will count up. Timer 2 will overflow at 0FFFFH and set the TF2 flag, which can then generate an interrupt if the interrupt is enabled. This overflow also causes the 16-bit value in RCAP2L and RCAP2H to be reloaded into the timer registers TL2 and TH2. A logic 0 applied to pin T2EX causes Timer 2 to count down. The timer will underflow when TL2 and TH2 become equal to the value stored in RCAP2L and RCAP2H. This underflow sets the TF2 flag and causes 0FFFFH to be reloaded into the timer registers TL2 and TH2.

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

Figure 11-7. Timer 2 in Auto-Reload Mode (DCEN=0)

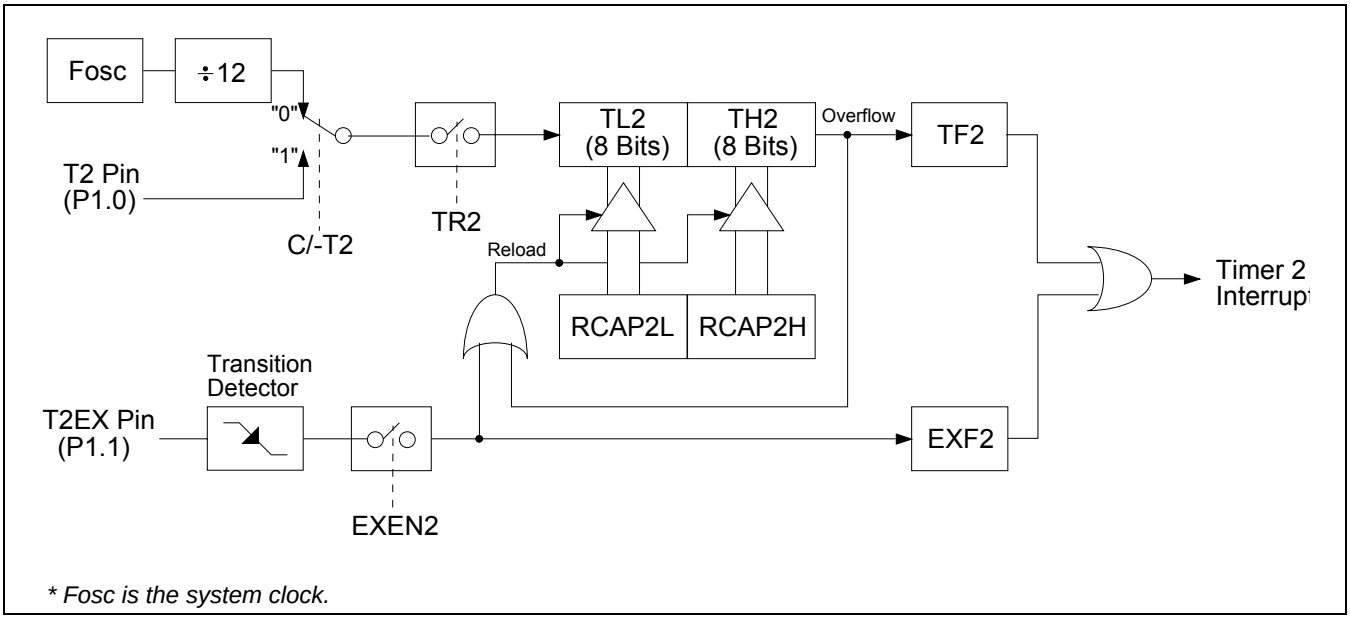
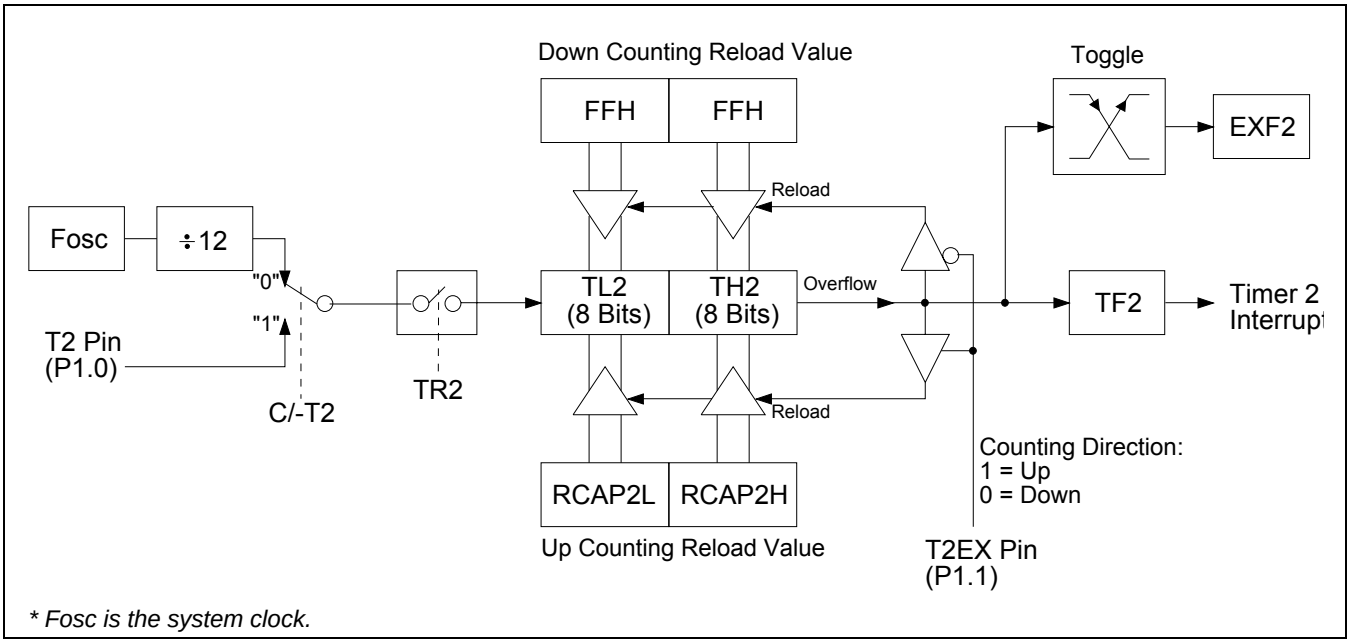


Figure 11-8. Timer 2 in Auto-Reload Mode (DCEN=1)



11.2.3 Baud Rate Generator Mode

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

Figure 11-9 shows the Timer 2 in baud rate generation mode. The baud rate generation mode is like the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in modes 1 and 3 are determined by Timer 2's overflow rate given below:

$$\text{Mode 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The timer can be configured for either “Timer” or “Counter” operation. In many applications, it is configured for “Timer” operation (C/-T2=0). Timer operation is different for Timer 2 when it is being used as a baud rate generator.

Usually, as a timer it would increment at 1/12 the system clock frequency. As a baud rate generator, it increments at 1/2 the system clock. Thus the baud rate formula is as follows:

$$\text{Mode 1 and 3 Baud Rates} = \frac{\text{Fosc}}{2 \times (65536 - [\text{RCAP2H,RCAP2L}])} \times \frac{1}{16}$$

Where:

Fosc is the system clock. And (RCAP2H, RCAP2L)= The content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer, which can be derived by:

$$[\text{RCAP2H,RCAP2L}] = 65536 - \frac{\text{Fosc}}{32 \times \text{Baud Rate}}$$

The Timer 2 as a baud rate generator mode shown in Figure 11-9 is valid only if RCLK and/or TCLK=1 in T2CON register. Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Thus, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Also if the EXEN2 (T2 external enable bit) is set, a 1-to-0 transition in T2EX (Timer/counter 2 trigger input) will set EXF2 (T2 external flag) but will not cause a reload from (RCAP2H, RCAP2L) to (TH2,TL2). Therefore when Timer 2 is in use as a baud rate generator, T2EX can be used as an additional external interrupt, if needed.

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

Figure 11-9. Timer 2 in Baud Rate Generator Mode

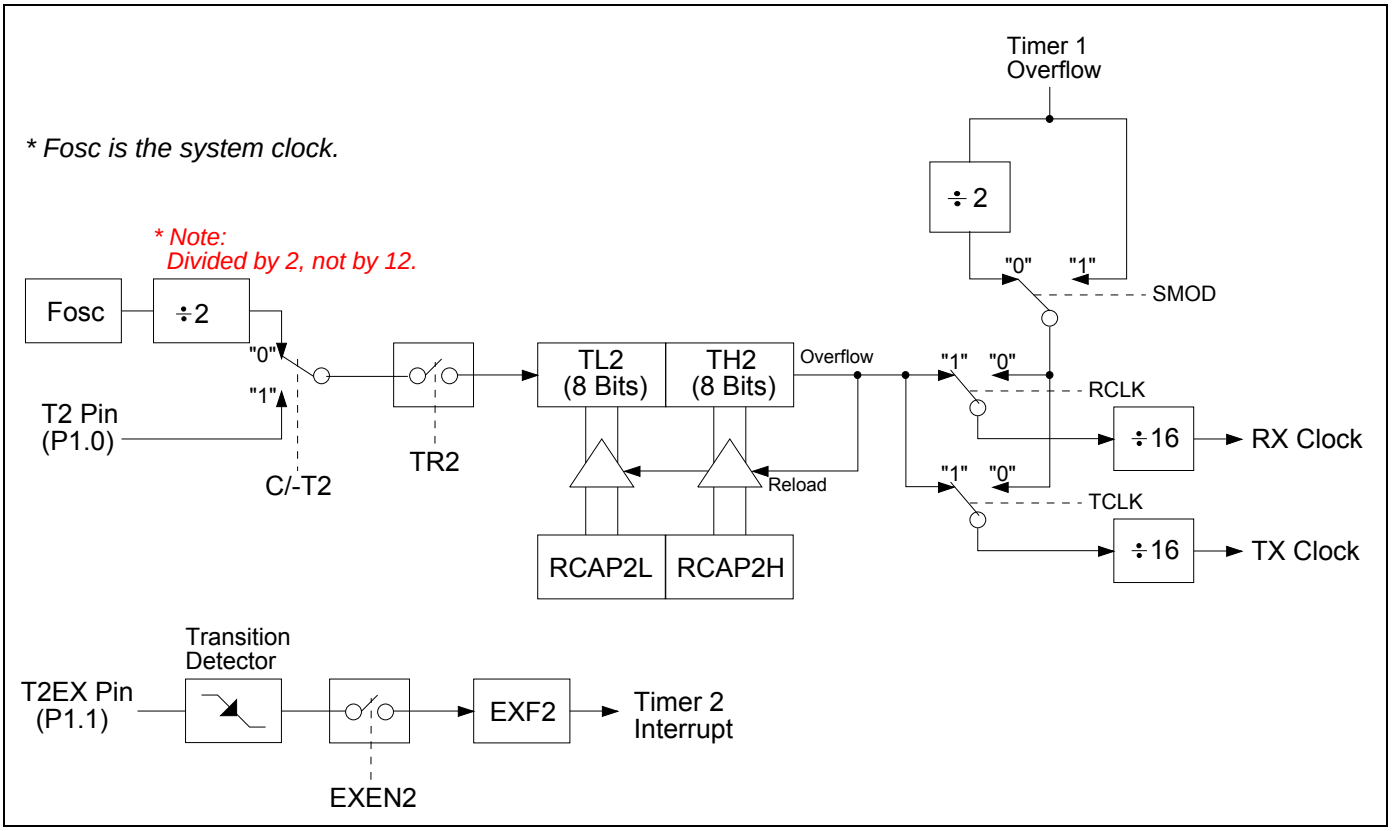


Table 11-2 & Table 11-3 list various commonly used baud rates and how they can be obtained from Timer 2.

Table 11-2. Timer 2 Generated Commonly Used Baud Rates @ Fosc=11.0592MHz

Baud Rate	Timer 2 in Baud Rate Generator Mode		
	[RCAP2H, RCAP2L]	RCAP2H	RCAP2L
300	64384	FBH	80H
600	64960	FDH	C0H
1200	65152	FEH	80H
1800	65248	FEH	E0H
2400	65392	FFH	70H
4800	65440	FFH	A0H
7200	65464	FFH	B8H
9600	65488	FFH	D0H
14400	65500	FFH	DCH
19200	65518	FFH	EEH
38400	65524	FFH	F4H
57600	65530	FFH	FAH
115200	65533	FFH	FDH

Table 11-3. Timer 2 Generated Commonly Used Baud Rates @ Fosc=22.1184MHz

Baud Rate	Timer 2 in Baud Rate Generator Mode		
	[RCAP2H, RCAP2L]	RCAP2H	RCAP2L
300	63232	F7H	00H
600	64384	FBH	80H
1200	64960	FDH	C0H
1800	65152	FEH	80H
2400	65248	FEH	E0H
4800	65392	FFH	70H
7200	65440	FFH	A0H
9600	65464	FFH	B8H
14400	65488	FFH	D0H
19200	65500	FFH	DCH
38400	65518	FFH	EEH
57600	65524	FFH	F4H
115200	65530	FFH	FAH

11.2.4 Programmable Clock-Out from Timer 2

Using Timer 2, a 50% duty cycle clock can be programmed to come out from pin T2CKO (P1.0).

The clock-out frequency depends on the system clock frequency (Fosc) and the reload value filled in the RCAP2H and RCAP2L registers, as shown in the following formula:

$$\text{Clock-Out Frequency} = \frac{\text{Fosc}}{4 \times (65536 - [\text{RCAP2H,RCAP2L}])}$$

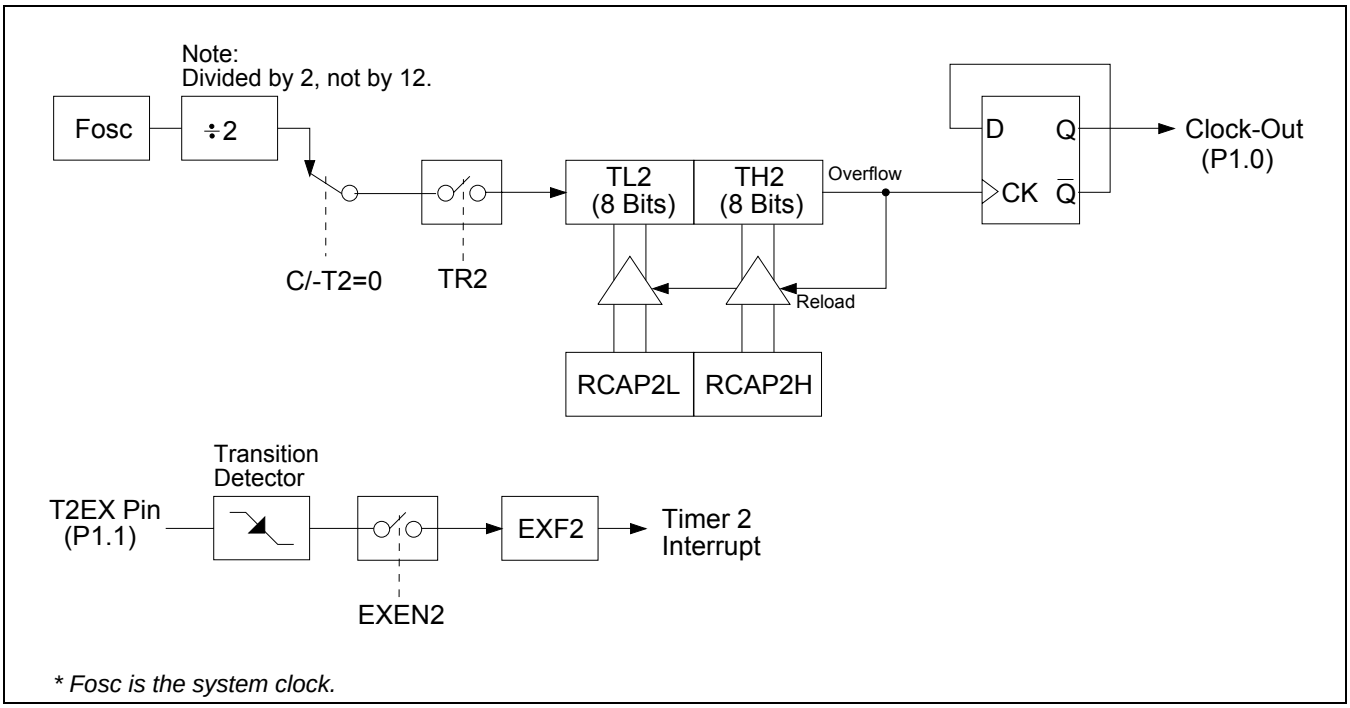
Where [RCAP2H,RCAP2L]= the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 is programmed for the clock-out mode as follows:

- Set T2OE bit in T2MOD register.
- Clear C/-T2 bit in T2CON register.
- Determine the 16-bit reload value from the formula and enter it in RCAP2H and RCAP2L registers.
- Enter a 16-bit initial value in TH2 and TL2 registers. It should be the same as the reload value.
- Start the timer by setting the run control bit TR2 in T2CON register.

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, however, that the baud-rate and the clock-out frequency depend on the same overflow rate of Timer 2.

Figure 11-10. Programmable Clock-Out from Timer 2



** Fosc is the system clock.*

12 Serial Port

12.1 Standard UART Operation

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

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

Mode 0: 8 data bits (LSB first) are transmitted or received through RXD. TXD always outputs the shift clock. The baud rate is fixed at 1/12 the system clock frequency, i.e., Fosc/12.

Mode 1: 10 bits are transmitted through TXD or received through RXD: a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in special function register SCON. The baud rate is variable.

Mode 2: 11 bits are transmitted through TXD or received through RXD: start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On Transmit, the 9th data bit (TB8 in SCON register) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in PSW register) could be moved into TB8. On receive, the 9th data bit goes into RB8 in SCON register, while the stop bit is ignored. The baud rate can be configured to 1/32 or 1/64 the system clock frequency, i.e., Fosc/64 or Fosc/32.

Mode 3: 11 bits are transmitted through TXD or received through RXD: a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). In fact, Mode 3 is the same as Mode 2 in all respects except baud rate. The baud rate in Mode 3 is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. In Mode 0, reception is initiated by the condition RI=0 and REN=1. In the other modes, reception is initiated by the incoming start bit if REN=1.

12.1.1 Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications. In these two modes, 9 data bits are received. The 9th bit goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8=1. This feature is enabled by setting bit SM2 (in SCON register). A way to use this feature in multiprocessor systems is as follows:

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2=1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and check if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2 set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2=1, the receive interrupt will not be activated unless a valid stop bit is received.

12.1.2 Serial Port Related Registers

The serial port control and status register is the special function register SCON. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).

SCON (Address=98H, Serial Port Control Register, Reset Value=0000,0000B)

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

FE: Framing Error bit. This bit is set by the receiver when an invalid stop bit is detected. The FE bit is not cleared by valid frames but should be cleared by software. The SMOD0 bit (in PCON register) must be '1' to enable access to the FE bit.

SM0: Serial Port Mode Bit 0 (SMOD0 must be '0' to access bit SM0).
SM1: Serial Port Mode Bit 1.

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift Register	Fosc/12 or Fosc/2 *Note: dependent on bit URM0X6
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	Fosc/64 or Fosc/32
1	1	3	9-bit UART	Variable

Where, Fosc is the system clock frequency.

SM2: Enables the Automatic Address Recognition feature in Modes 2 or 3. If SM2=1 then RI will not be set unless the received 9th data bit (RB8) is '1', indicating an address, and the received byte is a Given or Broadcast Address. In Mode 1, if SM2=1 then RI will not be activated unless a valid stop bit was received, and the received byte is a Given or Broadcast Address. In Mode 0, SM2 should be '0'.

REN: Enables serial reception. Set by software to enable reception. Cleared by software to disable reception.

TB8: The 9th data bit that will be transmitted in Modes 2 and 3. Set or cleared by software as desired.

RB8: In modes 2 and 3, the 9th data bit that was received. In Mode 1, if SM2=0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.

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

RI: Receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

PCON (Address=87H, Power Control Register, Reset Value=00xx,0000B (or 00x1,0000B after Power-On Reset))

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

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

AUXR2 (Address=A6H, Auxiliary Register 2, Reset Value=00x0,0000B)

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	S2TR	S2SMOD	S2TX12	S2CKOE	T0CKOE

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

URM0X6:
Set to select Fosc/2 as the baud rate for UART Mode 0.
Clear to select Fosc/12 as the baud rate for UART Mode 0.

12.1.3 Baud Rates

The baud rate in Mode 0 can be Fosc/12 or Fosc/2 dependent on the control bit URM0X6 (in AUXR2 register). Where, Fosc is the system clock frequency. The baud rates in Modes 1 and 3 are determined by the Timer 1 or Timer 2 overflow rate. The baud rate in Mode 2 depends on the value of bit SMOD in PCON register. If SMOD=0 (which is the value on reset), the baud rate is Fosc/64; if SMOD=1, the baud rate is Fosc/32, as shown below.

$$\text{Mode 2 Baud Rate} = \frac{2^{\text{SMOD}}}{64} \times \text{Fosc}$$

12.1.4 Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator (T2CON.RCLK=0, T2CON.TCLK=0), the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times (\text{Timer 1 Overflow Rate})$$

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either “timer” or “counter” operation, and in any of its 3 running modes. In the most typical applications, it is configured for “timer” operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case the baud rate is given by the formula:

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{SMOD}}{32} \times \frac{Fosc}{n \times (256-TH1)}$$

Where, n=12 if T1X12=0; n=1 if T1X12=1. *(Note: T1X12 bit is in AUXR2 register.)*

One can achieve very low baud rates with Timer 1 by leaving the Timer 1 interrupt enabled, and configuring the Timer to run as a 16-bit timer (high nibble of TMOD=0001B), and using the Timer 1 interrupt to do a 16-bit software reload.

Table 12-1 & Table 12-2 list various commonly used baud rates and how they can be obtained from Timer 1 in its 8-Bit Auto-Reload Mode.

Table 12-1. Timer 1 Generated Commonly Used Baud Rates @ Fosc=11.0592MHz

Baud Rate	TH1, the Reload Value			
	T1X12=0		T1X12=1	
	SMOD=0	SMOD=1	SMOD=0	SMOD=1
300	160	64	-	-
600	208	160	-	-
1200	232	208	-	-
1800	240	224	64	-
2400	244	232	112	-
4800	250	244	184	112
7200	252	248	208	160
9600	253	250	220	184
14400	254	252	232	208
19200	-	253	238	220
38400	-	-	247	238
57600	-	255	250	244
115200	-	-	253	250

Table 12-2. Timer 1 Generated Commonly Used Baud Rates @ Fosc=22.1184MHz

Baud Rate	TH1, the Reload Value			
	T1X12=0		T1X12=1	
	SMOD=0	SMOD=1	SMOD=0	SMOD=1
300	64	-	-	-
600	160	64	-	-
1200	208	160	-	-
1800	224	192	-	-
2400	232	208	-	-
4800	244	232	112	-
7200	248	240	160	64
9600	250	244	184	112
14400	252	248	208	160
19200	253	250	220	184
38400	-	253	238	220
57600	255	254	244	232
115200	-	255	250	244

12.1.5 More About Mode 0

Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The baud rate is fixed at 1/12 the system clock frequency. Figure 12-1 shows a simplified functional diagram of the serial port in Mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The “write to SBUF” signal also loads a 1 into the 9th position of the transmit shift register and tells the TX Control block to commence a transmission. The internal timing is such that appropriate time will elapse between “write to SBUF” and activation of Send.

Send enables the output of the shift register to the alternate output function line of P3.0 and also enable Shift Clock to the alternate output function line of P3.1. Shift Clock is low for 6 clocks, and high for 6 clocks. At every 12-clock cycle the Send is active, the contents of the transmit shift are shifted to the right one position.

As data bits shift out to the right, zeros come in from the left. When the MSB of the data byte is at the output position of the shift register, then the ‘1’ that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX Control block to do one last shift and then deactivate Send and set TI. Both of these actions occur at the 10th bit duration after “write to SBUF.”

Reception is initiated by the condition REN=1 and RI=0. At the next instruction cycle, the RX Control unit writes the bits 11111110 to the receive shift register, and in the next clock phase activates Receive.

Receive enables Shift Clock to the alternate output function of P3.1 pin. Shift Clock makes transitions every 6 clock cycles. When Receive is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the P3.0 pin.

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX Control block to do one last shift and load SBUF. At the 10th bit duration after the write to SCON that cleared RI, Receive is cleared as RI is set.

12.1.6 More About Mode 1

10 bits are transmitted through TXD, or received through RXD: a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. The baud rate is determined by the Timer 1 or Timer 2 overflow rate. Figure 12-2 shows a simplified functional diagram of the serial port in Mode 1, and associated timings for transmit/receive.

Transmission is initiated by any instruction that uses SBUF as a destination register. The “write to SBUF” signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission actually commences at the instruction cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the “write to SBUF” signal.)

The transmission begins with activation of Send which puts the start bit at TXD. One bit time later, data is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeros are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just to the left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX Control unit to do one last shift and then deactivate Send and set TI. This occurs at the 10th divide-by-16 rollover after “write to SBUF.”

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written into the input shift register. Resetting the divide-by-16 counter aligns its rollovers with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16ths. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive

circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in mode 1 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:
1. RI = 0, and
2. Either SM2 = 0, or the received stop bit = 1.

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF, and RI is activated. At this time, whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition at the RXD input.

12.1.7 More About Modes 2 and 3

11 bits are transmitted through TXD, or received through RXD: a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to either 1/32 or 1/64 the system clock frequency in Mode 2. Mode 3 may have a variable baud rate generated from Timer 1 or Timer 2. Figure 12-3 and Figure 12-4 show a functional diagram of the serial port in Modes 2 and 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The “write to SBUF” signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX Control unit that a transmission is requested. Transmission commences at the instruction cycle following the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the “write to SBUF” signal.)

The transmission begins with activation of Send, which puts the start bit at TXD. One bit time later, data is activated, which enables the output bit of the transmit shift register to TXD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeros are clocked in. Thus, as data bits shift out to the right, zeros are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just to the left of TB8, and all positions to the left of that contain zeros. This condition flags the TX Control unit to do one last shift and then deactivate Send and set TI. This occurs at the 11th divide-by-16 rollover after “write to SUBF.”

Reception is initiated by a detected 1-to-0 transition at RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and 1FFH is written to the input shift register. At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in Modes 2 and 3 is a 9-bit register), it flags the RX Control block to do one last shift, load SBUF and RB8, and set RI.

The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated.
1. RI = 0, and
2. Either SM2 = 0, or the received 9th data bit = 1.

If either of these conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, and the first 8 data bits go into SBUF. One bit time later, whether the above conditions were met or not, the unit goes back to looking for a 1-to-0 transition at the RXD input.

Figure 12-1. Serial Port Mode 0

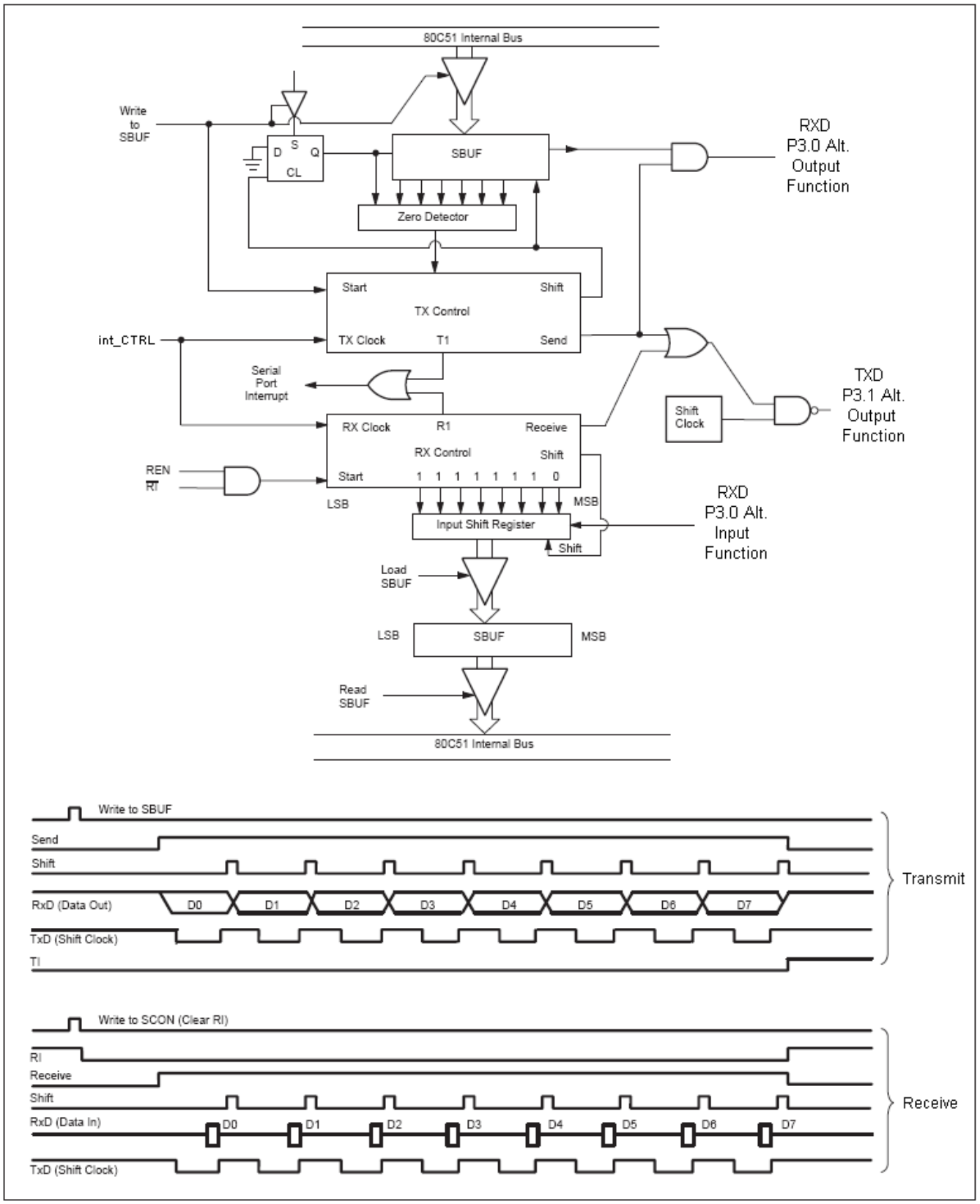


Figure 12-2. Serial Port Mode 1

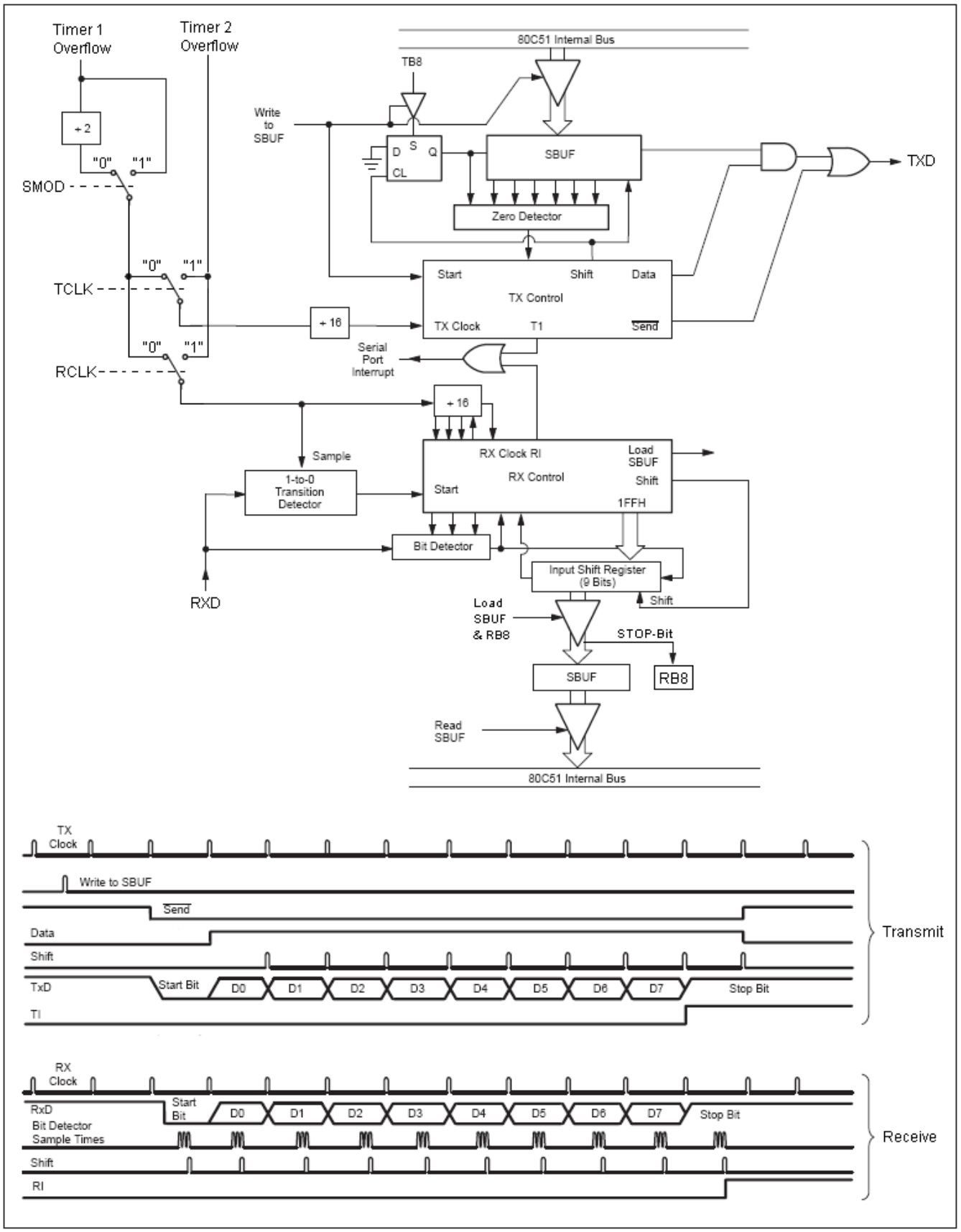


Figure 12-3. Serial Port Mode 2

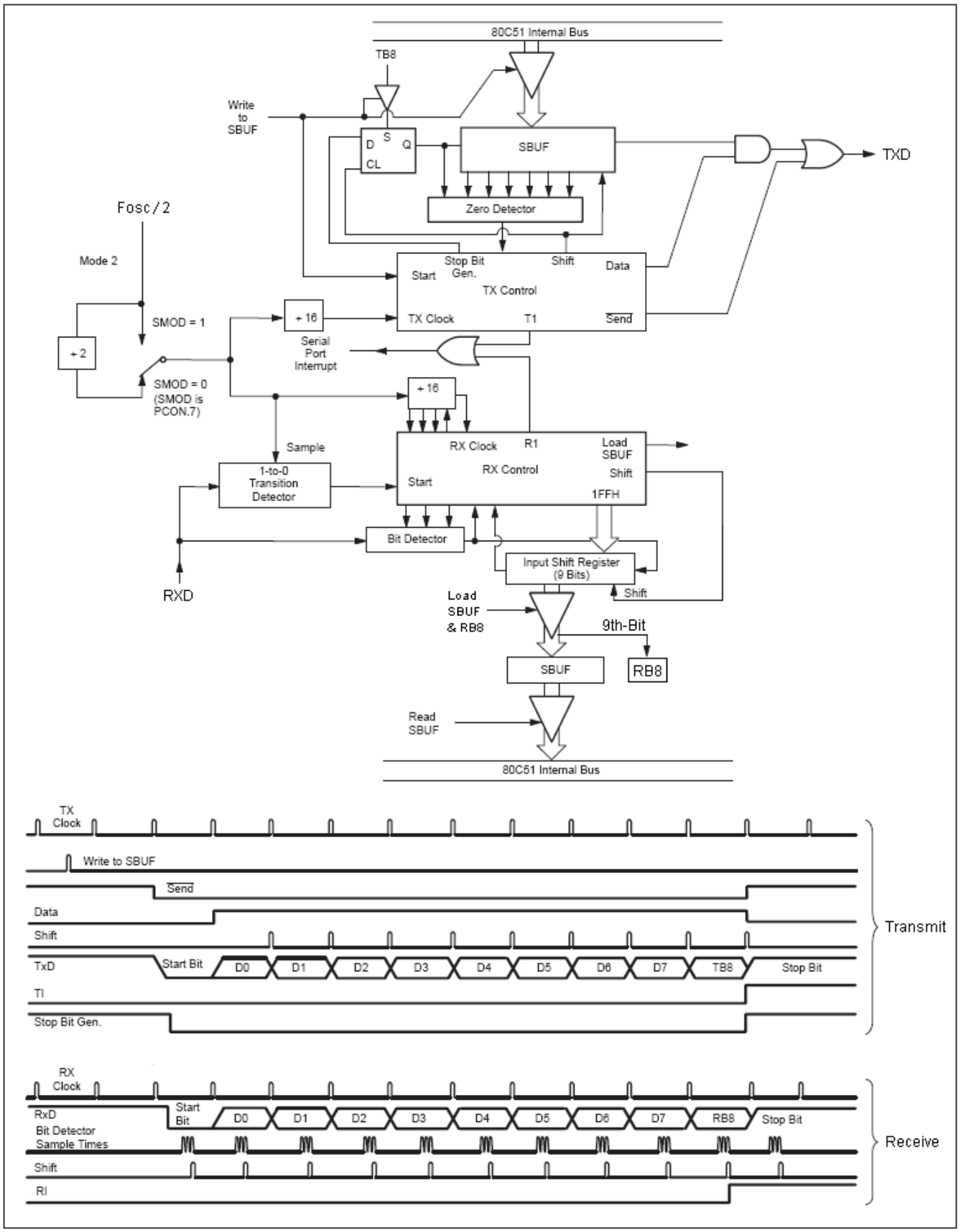
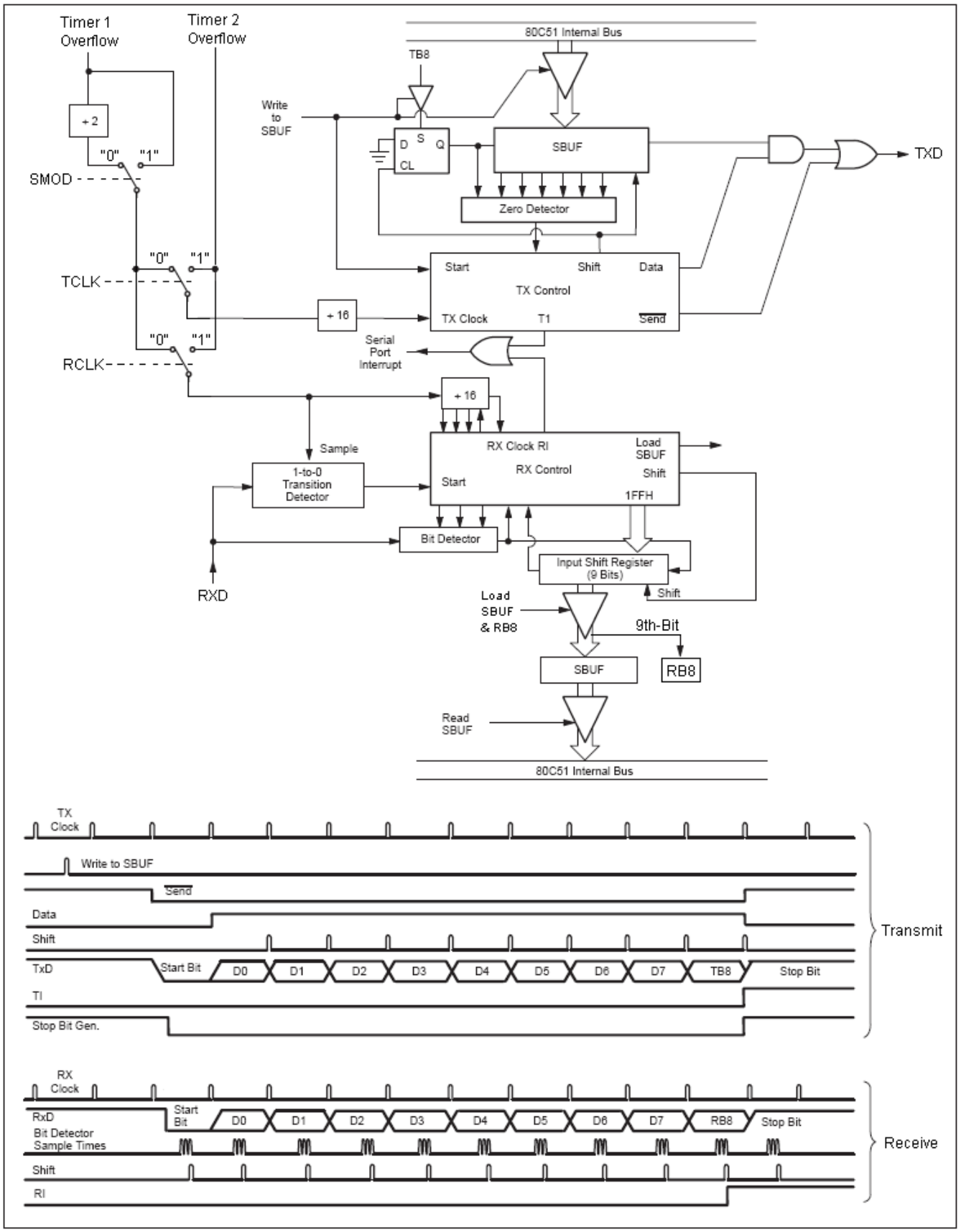


Figure 12-4. Serial Port Mode 3



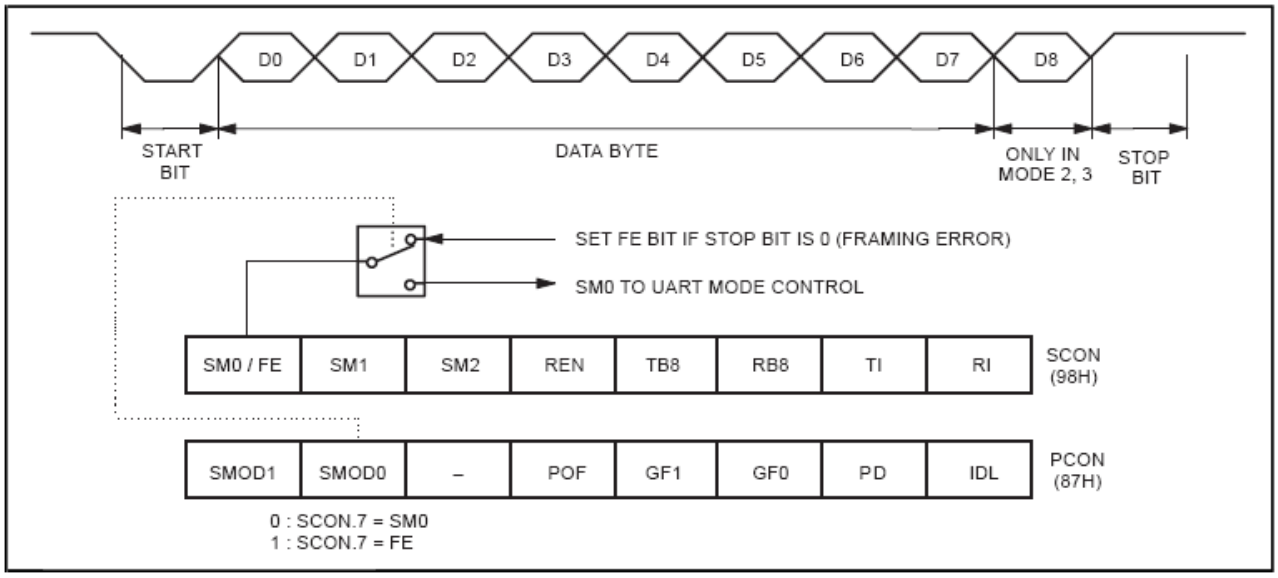
12.2 Enhanced UART Functions

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

12.2.1 Framing Error Detection

When used for framing error detection, the UART looks for missing stop bits in the communication. A missing stop bit will set the FE bit in the SCON register. The FE bit shares the SCON.7 bit with SM0 and the function of SCON.7 is determined by SMOD0 bit (PCON.6). If SMOD0 is set then SCON.7 functions as FE. SCON.7 functions as SM0 when SMOD0 is cleared. When SCON.7 functions as FE, it can only be cleared by software. Refer to Figure 12-5.

Figure 12-5. UART Framing Error Detection



12.2.2 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 Figure 12-6. 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                    Slave 1
    SADDR = 1100 0000          SADDR = 1100 0000
    SADEN = 1111 1101          SADEN = 1111 1110
    Given = 1100 00X0          Given = 1100 000X
```

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

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

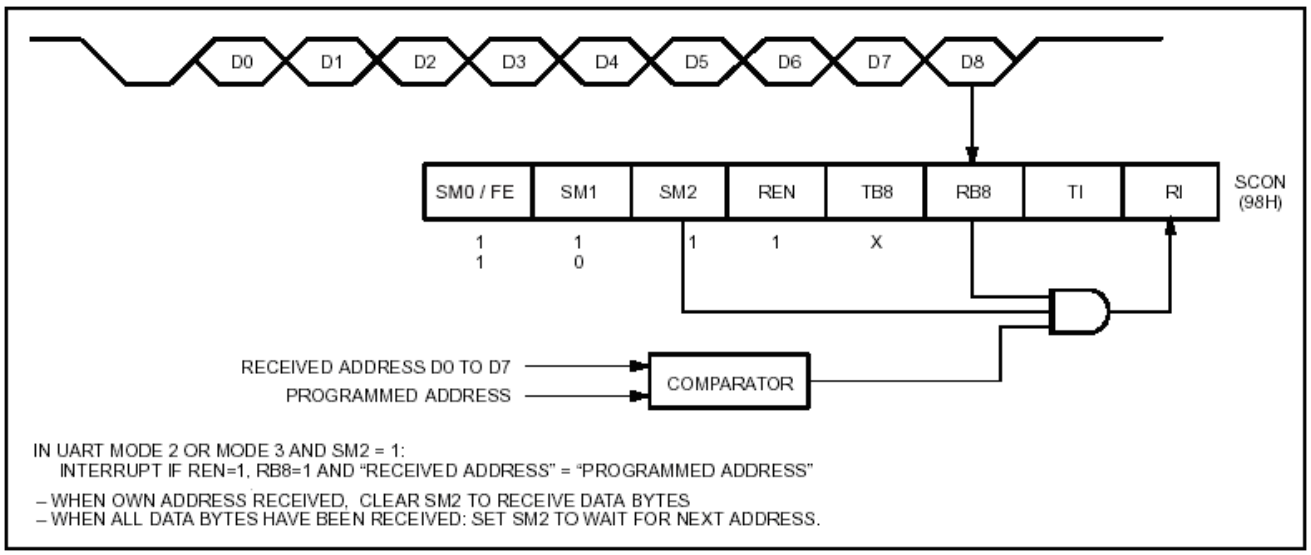
```
Slave 0                    Slave 1                    Slave 2
    SADDR = 1100 0000          SADDR = 1110 0000          SADDR = 1110 0000
    SADEN = 1111 1001          SADEN = 1111 1010          SADEN = 1111 1100
    Given = 1100 0XX0          Given = 1110 0X0X          Given = 1110 00XX
```

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

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

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

Figure 12-6. UART Multiprocessor Communication, Auto Address Recognition



13 Secondary UART (UART2)

The MPC82G516A is equipped with a secondary UART (hereafter, called UART2), which also has four operation modes the same as the first UART except the following differences:

(1) The UART2 has no enhanced functions: Framing Error Detection and Auto Address Recognition.
(2) The UART2 use the dedicated Baud Rate Timer as its Baud Rate Generator.
(3) The UART2 uses port pin P1.3 (S2TXD) and P1.2 (S2RXD) for transmit and receive, respectively.

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

13.1 UART2 Related Registers

The following special function registers are related to the operation of the UART2:

S2CON (Address=AAH, UART2 Control Register, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
S2SM0	S2SM1	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI

S2SM0: UART2 Mode Select Bit 0.
S2SM1: UART2 Mode Select Bit 1.

S2SM0	S2SM1	Mode	Description	Baud Rate
0	0	0	Shift Register	Fosc/12
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	Fosc/64 or Fosc/32
1	1	3	9-bit UART	Variable

Where, Fosc is the system clock frequency.

S2SM2: Enables the multiprocessor communication feature in Modes 2 or 3. If SM2=1 then RI will not be set unless the received 9th data bit (RB8) is '1', indicating an address, and the received byte is a Given or Broadcast Address. In Mode 1, if SM2=1 then RI will not be activated unless a valid stop bit was received, and the received byte is a Given or Broadcast Address. In Mode 0, SM2 should be '0'.

S2REN: Enables serial reception. Set by software to enable reception. Cleared by software to disable reception.

S2TB8: The 9th data bit that will be transmitted in Modes 2 and 3. Set or cleared by software as desired.

S2RB8: In modes 2 and 3, the 9th data bit that was received. In Mode 1, if SM2=0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.

S2TI: Transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.

S2RI: Receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

S2BUF (Address=9AH, UART2 Serial Data Buffer, Reset Value=xxH)

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

S2BRT (Address=BAH, UART2 Baud Rate Timer Reload Register, Reset Value=00H)

7	6	5	4	3	2	1	0
(Baud Rate Timer Reload Value)							

AUXR2 (Address=A6H, Auxiliary Register 2, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
T0X12	T1X12	URM0X6	S2TR	S2SMOD	S2TX12	S2CKOE	T0CKOE

S2TR: UART2 Baud Rate Timer control bit. Set/clear to turn on/off, respectively.
S2SMOD: UART2 double baud rate enable bit. When set, the baud rate is doubled.
S2TX12: UART2 Baud Rate Timer clock source select. Set to select Fosc, and clear to select Fosc/12.
S2CKOE: Set to enable the clock output of UART2 Baud Rate Timer on P3.5.

AUXR (Address=8EH, Auxiliary Register, Reset Value=0000, xx0xB)

7	6	5	4	3	2	1	0
URTS	ADRJ	P41ALE	P35ALE	-	-	EXTRAM	-

URTS:

0: Timer 1 or Timer 2 can be used as the Baud Rate Generator in Mode 1 and Mode 3.

1: Timer 1 overflow signal is replaced by the UART2 Baud Rate Timer overflow signal when Timer 1 is selected as the Baud Rate Generator in Mode1 or Mode 3 of the first UART. (Refer to Section 13-3.)

13.2 UART2 Baud Rates

13.2.1 Mode 0

If URM0X6=0,

$$\text{Mode 0 Baud Rate} = \frac{Fosc}{12}$$

If URM0X6=1,

$$\text{Mode 0 Baud Rate} = \frac{Fosc}{2}$$

13.2.2 Mode 1 and Mode 3

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{S2SMOD}}{32} \times \frac{Fosc}{n \times (256-S2BRT)}$$

Where,

n=12 if S2X12=0,
n=1 if S2X12=1.

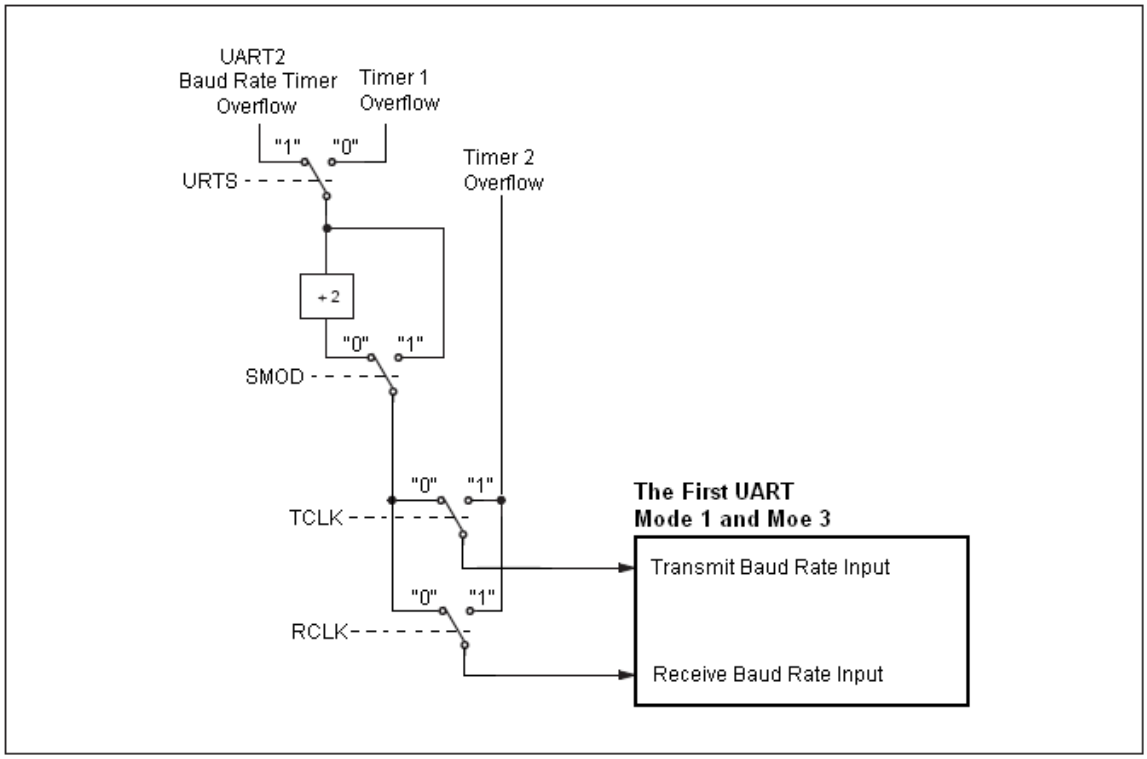
13.2.3 Mode 2

$$\text{Mode 2 Baud Rate} = \frac{2^{S2SMOD}}{64} \times Fosc$$

13.3 UART2 Baud Rate Timer Used by the First UART

In the Mode 1 and Mode 3 operation of the first UART, the user can select Timer 1 as the Baud Rate Generator by clearing bits TCLK and RCLK in T2CON register. At this time, if URTS bit (in AUXR register) is set, then Timer 1 overflow signal will be replaced by the overflow signal of the UART2 Baud Rate Timer. In other words, the user can adopt UART2 Baud Rate Timer as the Baud Rate Generator for Mode 1 or Mode 3 of the first UART as long as RCLK=0, TCLK=0 and URTS=1. In this condition, Timer 1 is free for other application. Of course, if UART2 (Mode 1 or Mode 3) is also operated at this time, these two UARTs will have the same baud rates.

Figure 13-1. New Baud Rate Source for the 1st UART



13.4 Programmable Clock-Out from UART2 Baud Rate Timer

Using the UART2 Baud Rate Timer, a 50% duty cycle clock can be programmed to come out from pin S2CKO (P3.5). The clock-out frequency depends on the system clock frequency (Fosc) and the reload value filled in the S2BRT register, as shown in the following formula:

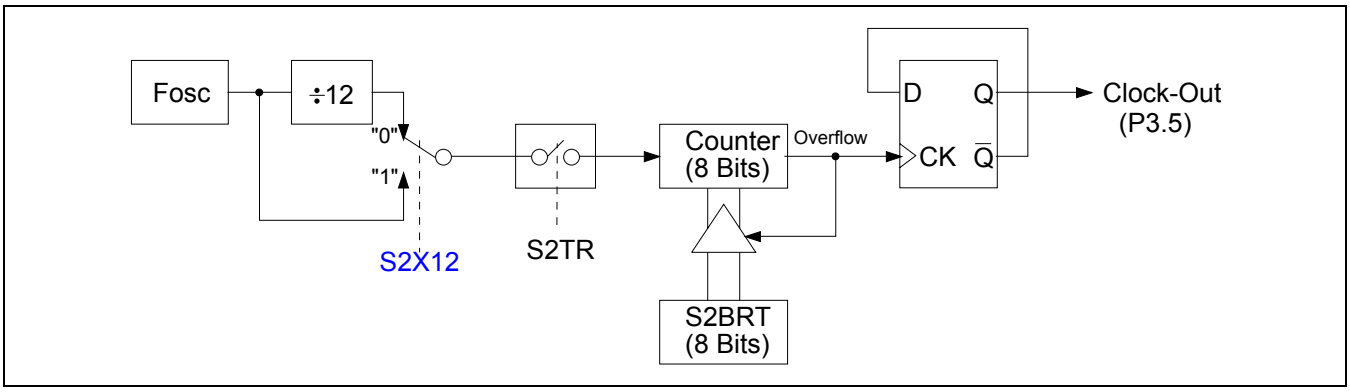
$$\text{Clock-Out Frequency} = \frac{\text{Fosc}}{n \times (256 - \text{S2BRT})}$$

Where,
n=24 if S2X12=0,
n=2 if S2X12=1.

The UART2 Baud Rate Timer is programmed for the clock-out mode as follows:

- Set S2CKOE bit in AUXR2 register.
- Determine the 8-bit reload value from the formula and enter it in S2BRT register.
- Start the timer by setting the run control bit S2TR in AUXR2 register.

Figure 13-2. Programmable Clock-Out from UART2 Baud Rate Timer



14 Programmable Counter Array (PCA)

The MPC82G516A is equipped with a Programmable Counter Array (PCA), which provides more timing capabilities with less CPU intervention than the standard timer/counters. Its advantages include reduced software overhead and improved accuracy.

14.1 PCA Overview

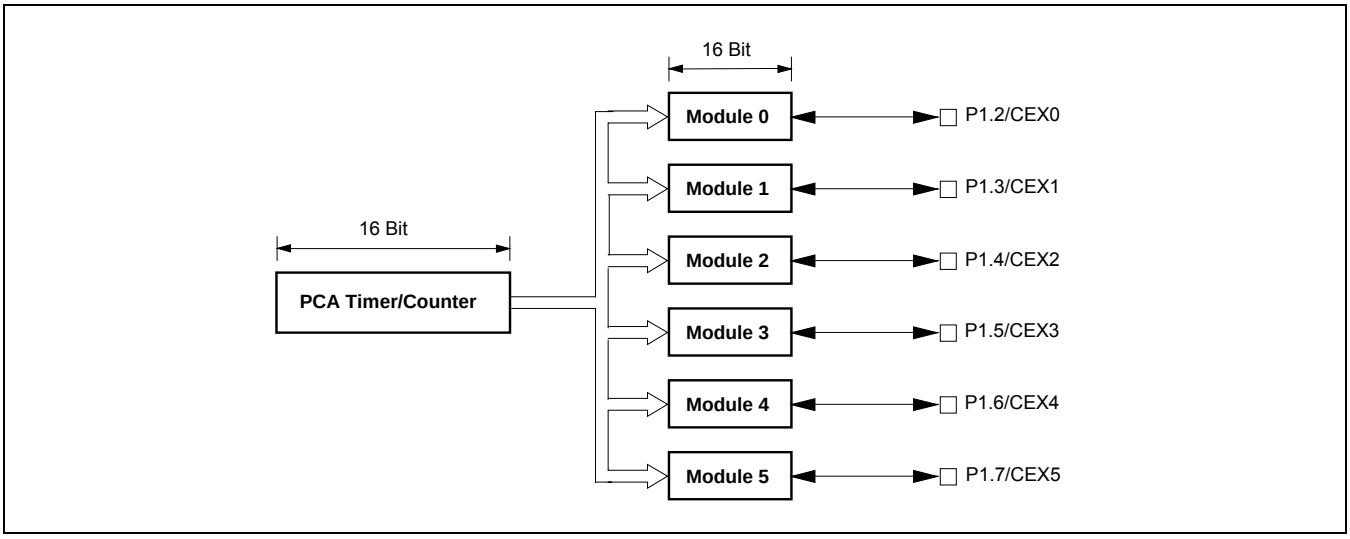
The PCA consists of a dedicated timer/counter which serves as the time base for an array of six compare/ capture modules. Figure 14-1 shows a block diagram of the PCA. Notice that the PCA timer and modules are all 16-bits. If an external event is associated with a module, that function is shared with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

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

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

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

Figure 14-1. PCA Block Diagram



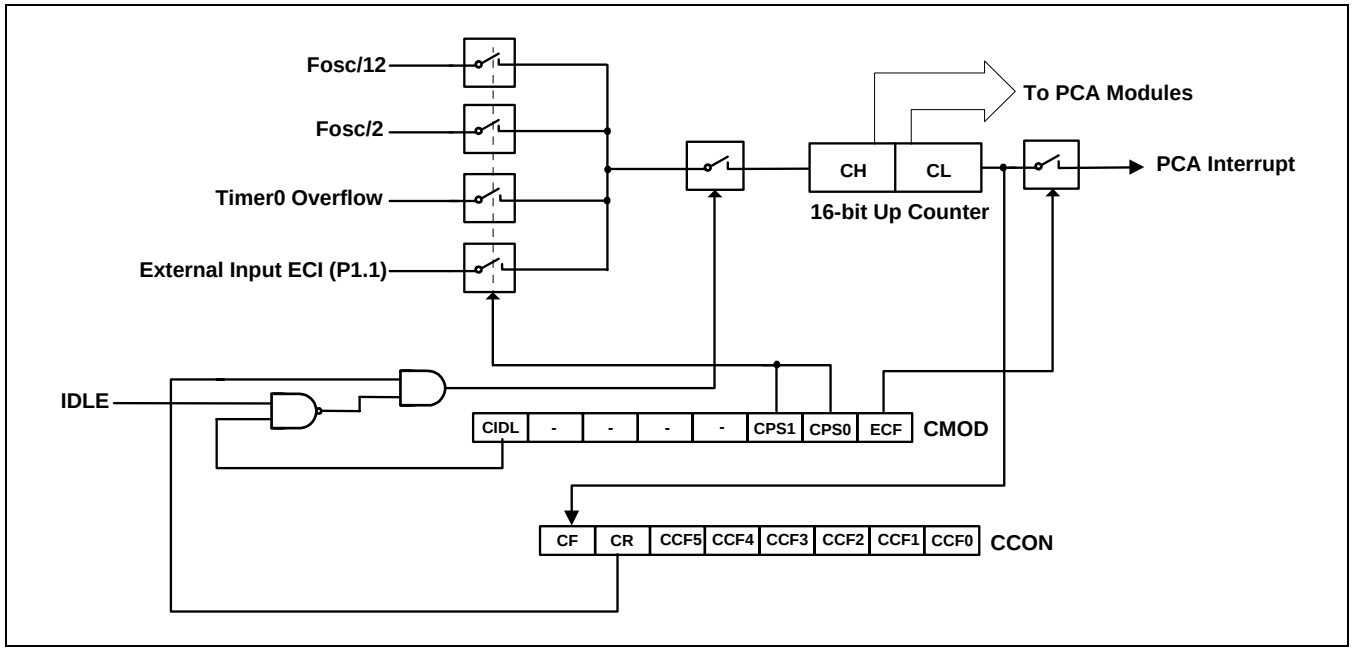
14.2 PCA Timer/Counter

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

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

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

Figure 14-2. PCA Timer/Counter



Where, Fosc is the system clock.

CMOD (Address=D9H, 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 (*Fosc is the system clock.)*
0	1	Internal clock, Fosc/2
1	0	Timer 0 overflow
1	1	External clock at the ECI pin.

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

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

CCON (Address=D8H, PCA Counter Control Register)

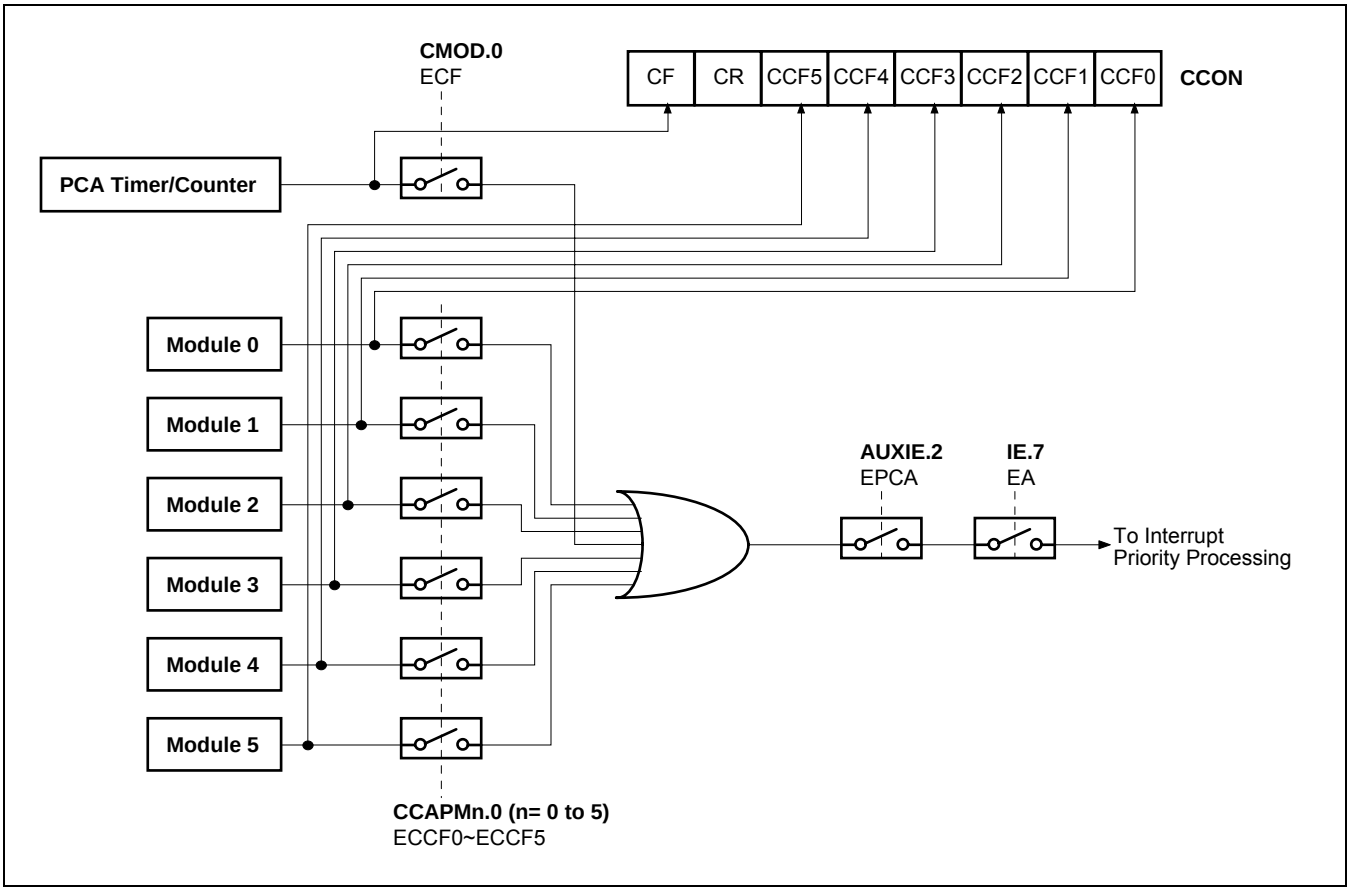
7	6	5	4	3	2	1	0
CF	CR	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0

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.

CCF0~CCF5: PCA Module 0 to Module 5 interrupt flags.
Set by hardware when a match or capture occurs. Must be cleared by software.

Figure 14-3. PCA Interrupt System



14.3 Compare/Capture Modules

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

CCAPMn, n=0~5 (Address=DAH~DFH, 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.

Each module also has a pair of 8-bit compare/capture registers (CCAPnH, CCAPnL) associated with it. These registers are used to store the time when a capture event occurred or when a compare event should occur.
When a module is used in the PWM mode, in addition to the above two registers, an extended register

PCAPWMn is used to improve the range of the duty cycle of the output. The improved range of the duty cycle starts from 0%, up to 100%, with a step of 1/256.

PCAPWMn, n=0~5 (Address=F2H~F7H, 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.

14.4 Operation Modes of the PCA

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

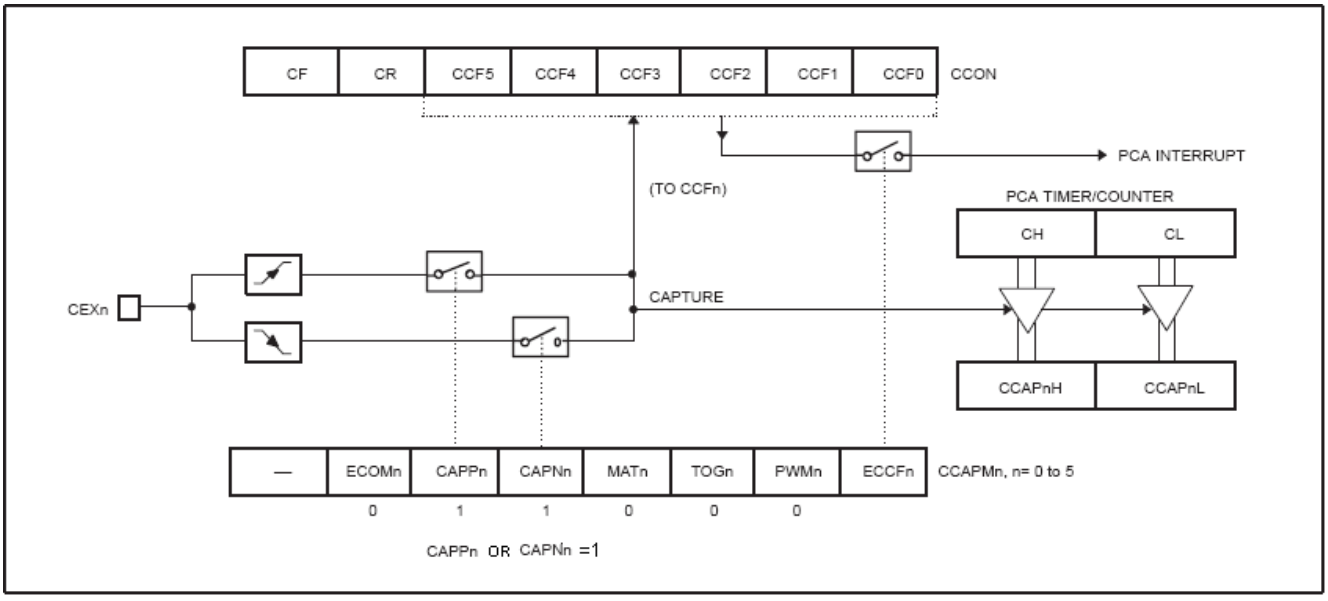
Table 14-1. PCA Module Modes

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

14.4.1 Capture Mode

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

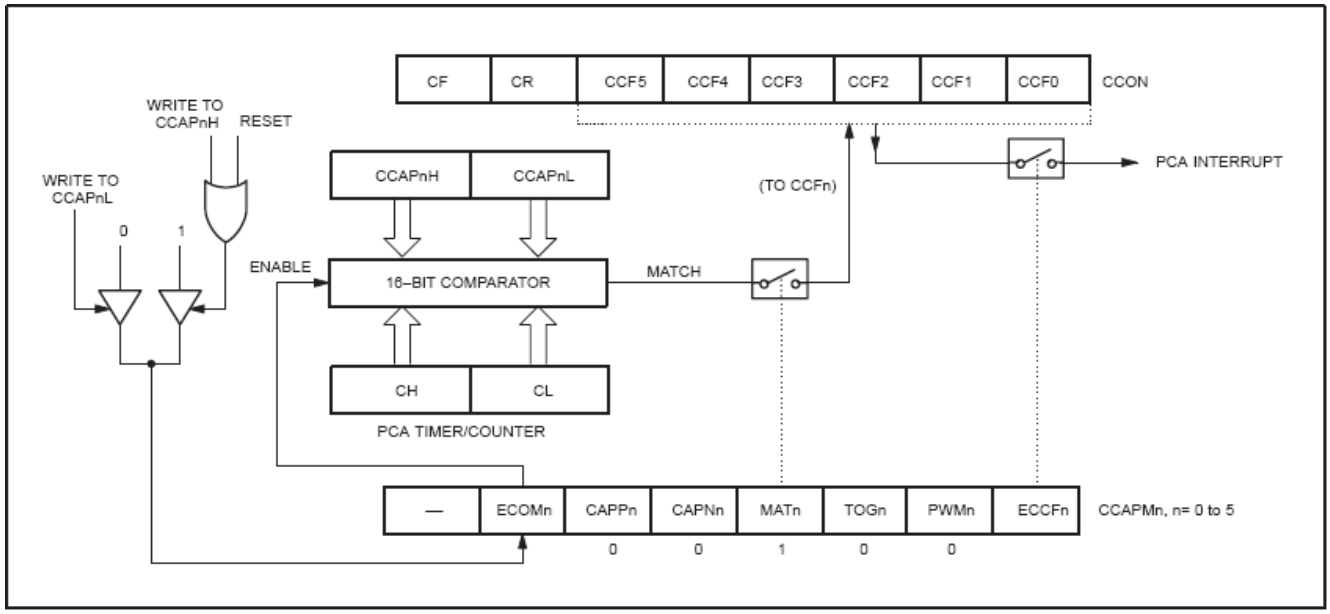
Figure 14-4. PCA Capture Mode



14.4.2 16-bit Software Timer Mode

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

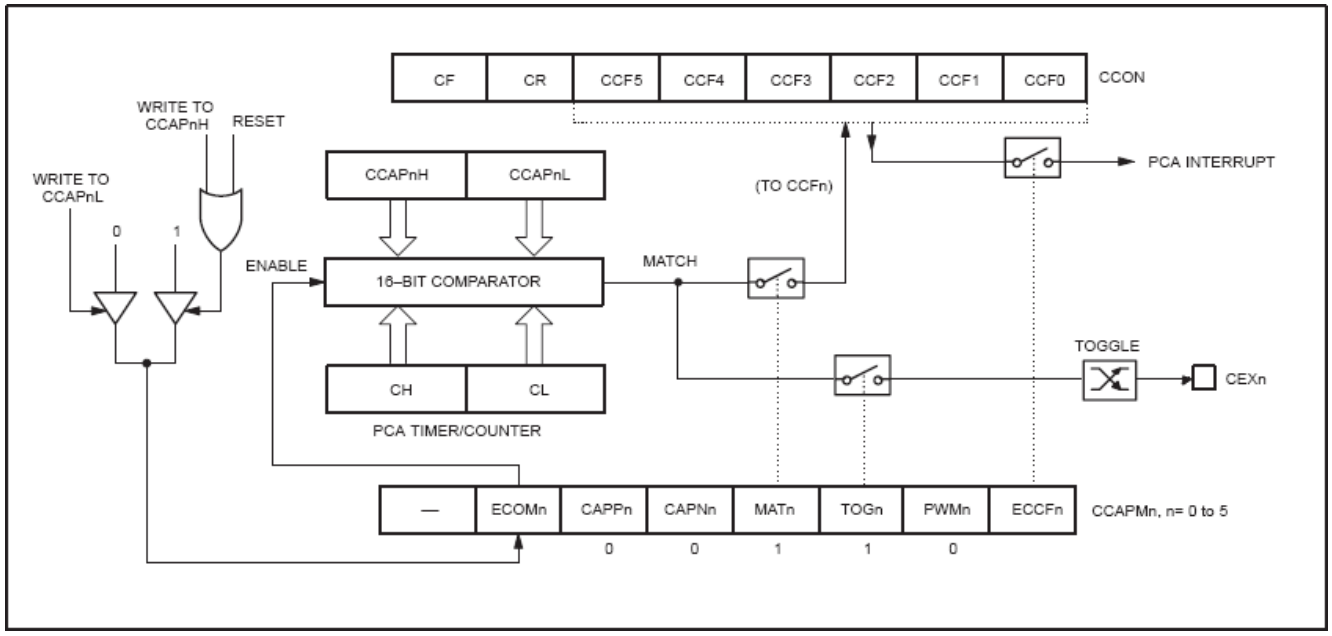
Figure 14-5. PCA Software Timer Mode



14.4.3 High Speed Output Mode

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

Figure 14-6. PCA High Speed Output Mode



14.4.4 PWM Mode

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

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

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

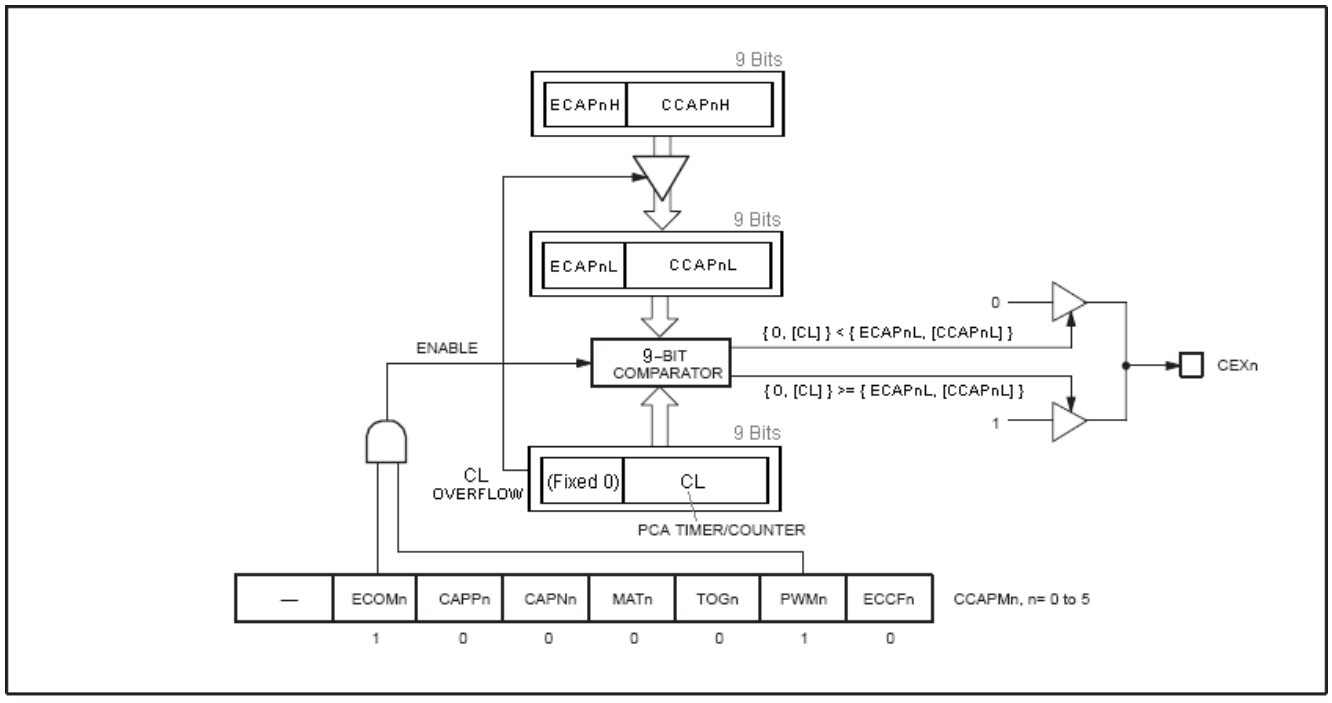
Using the 9-bit comparison, the duty cycle of the output can be improved to really start from 0%, and up to 100%. The formula for the duty cycle is:

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

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

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

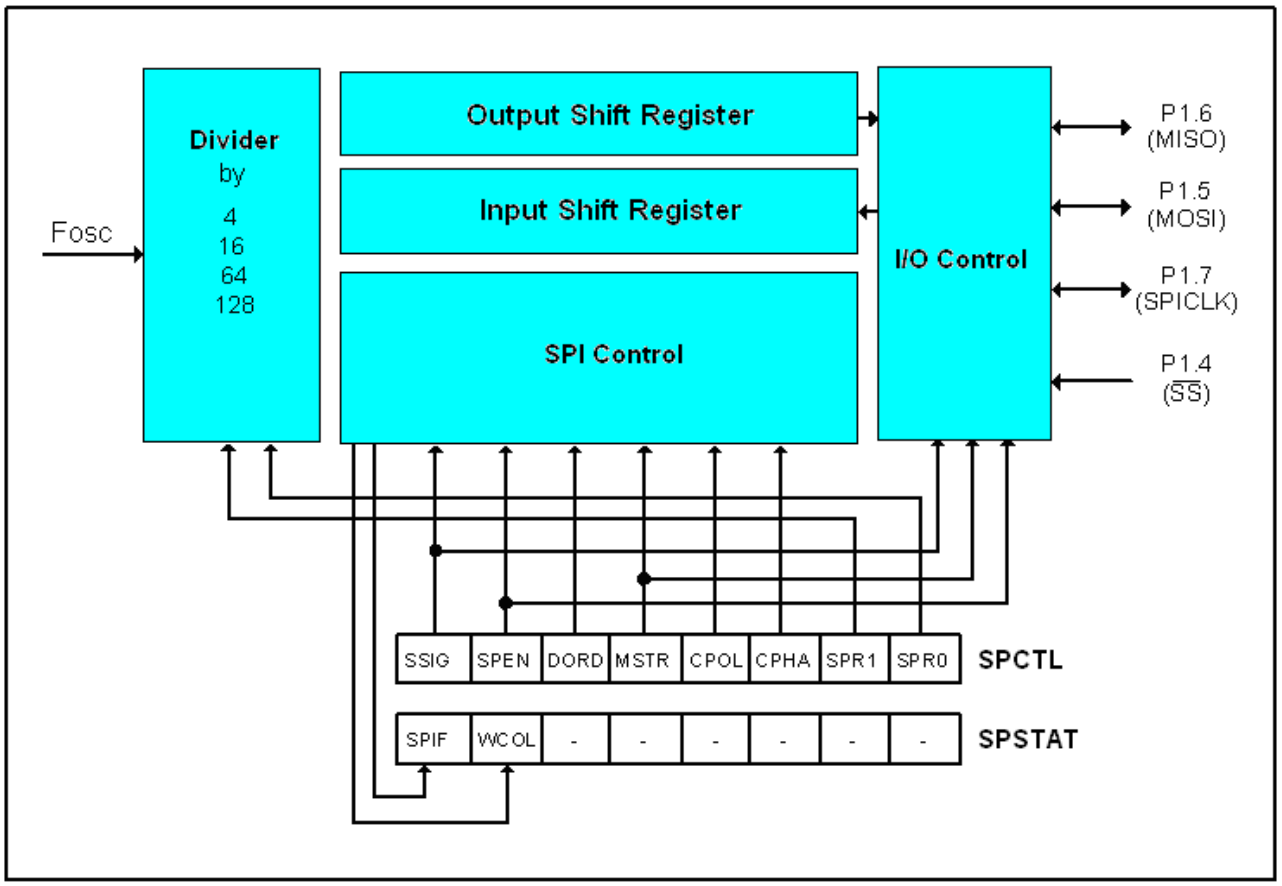
Figure 14-7. PCA PWM Mode



15 Serial Peripheral Interface (SPI)

The MPC82G516A provides a high-speed serial communication interface, the SPI interface. SPI is a full-duplex, high-speed and synchronous communication bus with two operation modes: Master mode and Slave mode. Up to 3 Mbps can be supported in either Master or Slave mode under a 12MHz system clock. It has a Transfer Completion Flag (SPIF) and Write Collision Flag (WCOL) in the SPI status register (SPSTAT).

Figure 15-1. SPI Block Diagram



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

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

• /SS is the optional slave select pin. In a typical configuration, an SPI master asserts one of its port pins to select one SPI device as the current slave. An SPI slave device uses its /SS pin to determine whether it is selected. The /SS is ignored if any of the following conditions are true:

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

Note that even if the SPI is configured as a master (MSTR=1), it can still be converted to a slave by driving the /SS pin low (if SSIG=0). Should this happen, the SPIF bit (SPSTAT.7) will be set. (See Section 15.5: Mode change on /SS-pin)

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

SPCTL (Address=85H, SPI Control Register, Reset Value=0000,0100B)

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 (in master mode)
00 : Fosc/4
01 : Fosc/16
10 : Fosc/64
11 : Fosc/128 (*Where, Fosc is the system clock.)*

SPSTAT (Address=84H, SPI Status Register, Reset Value=00xx,xxxxB)

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 SPI interrupt is enabled. If /SS pin is driven low when SPI is in master mode with SSIG=0, SPIF will also be set to signal the “mode change”. The SPIF is cleared in software by writing ‘1’ to this bit.

WCOL: SPI write collision flag.
The WCOL bit is set if the SPI data register, SPDAT, is written during a data transfer (see Section 15.6: Write Collision). The WCOL flag is cleared in software by writing ‘1’ to this bit.

SPDAT (Address=86H, SPI Data Register, Reset Value=0000,0000B)

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

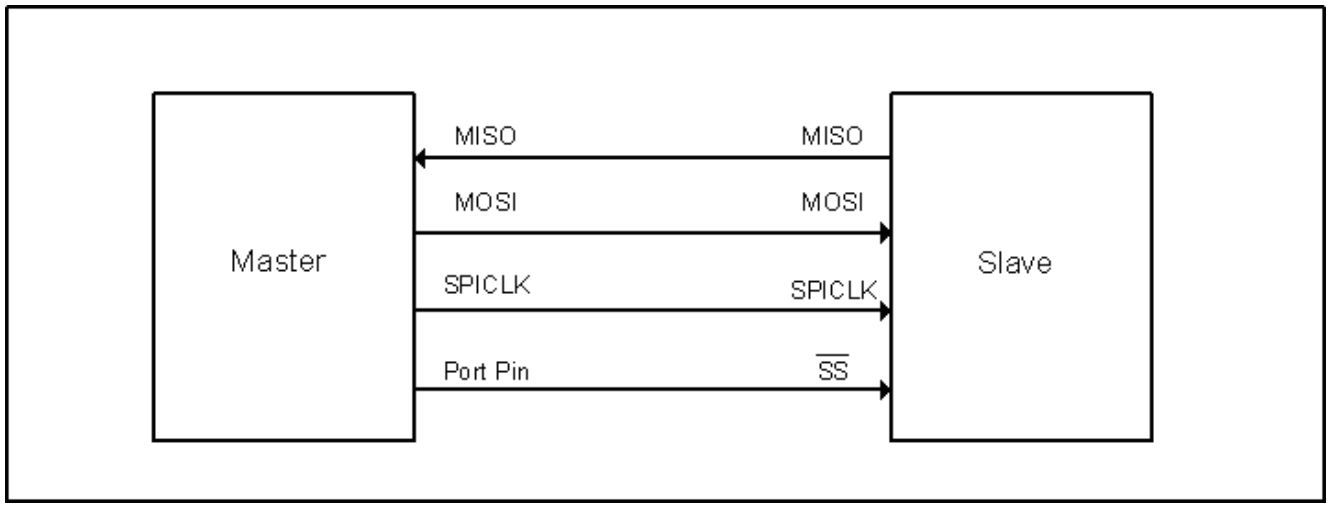
SPDAT has two physical buffers for writing to and reading from during transmit and receive, respectively.

15.1 Typical SPI Configurations

15.1.1 Single Master & Single Slave

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

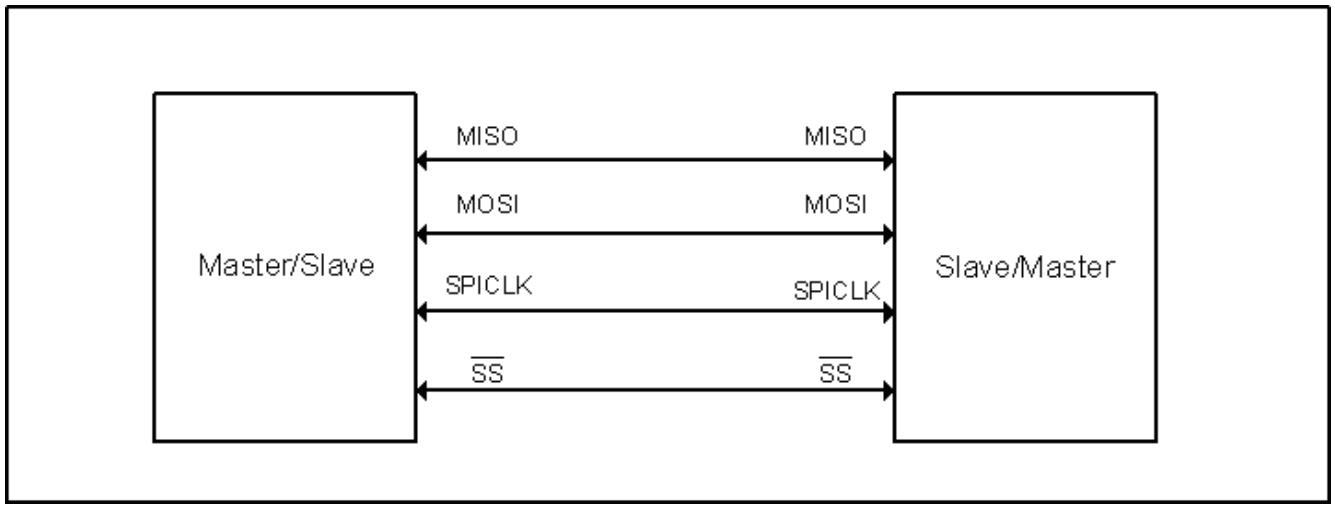
Figure 15-2. SPI single master single slave configuration



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

Two devices are connected to each other and either device can be a master or a slave. When no SPI operation is occurring, both can be configured as masters with MSTR=1, SSIG=0 and P1.4 (/SS) configured in quasi-bidirectional mode. When any device initiates a transfer, it can configure P1.4 as an output and drive it low to force a "mode change to slave" in the other device. (See Section 15.5: Mode change on /SS-pin)

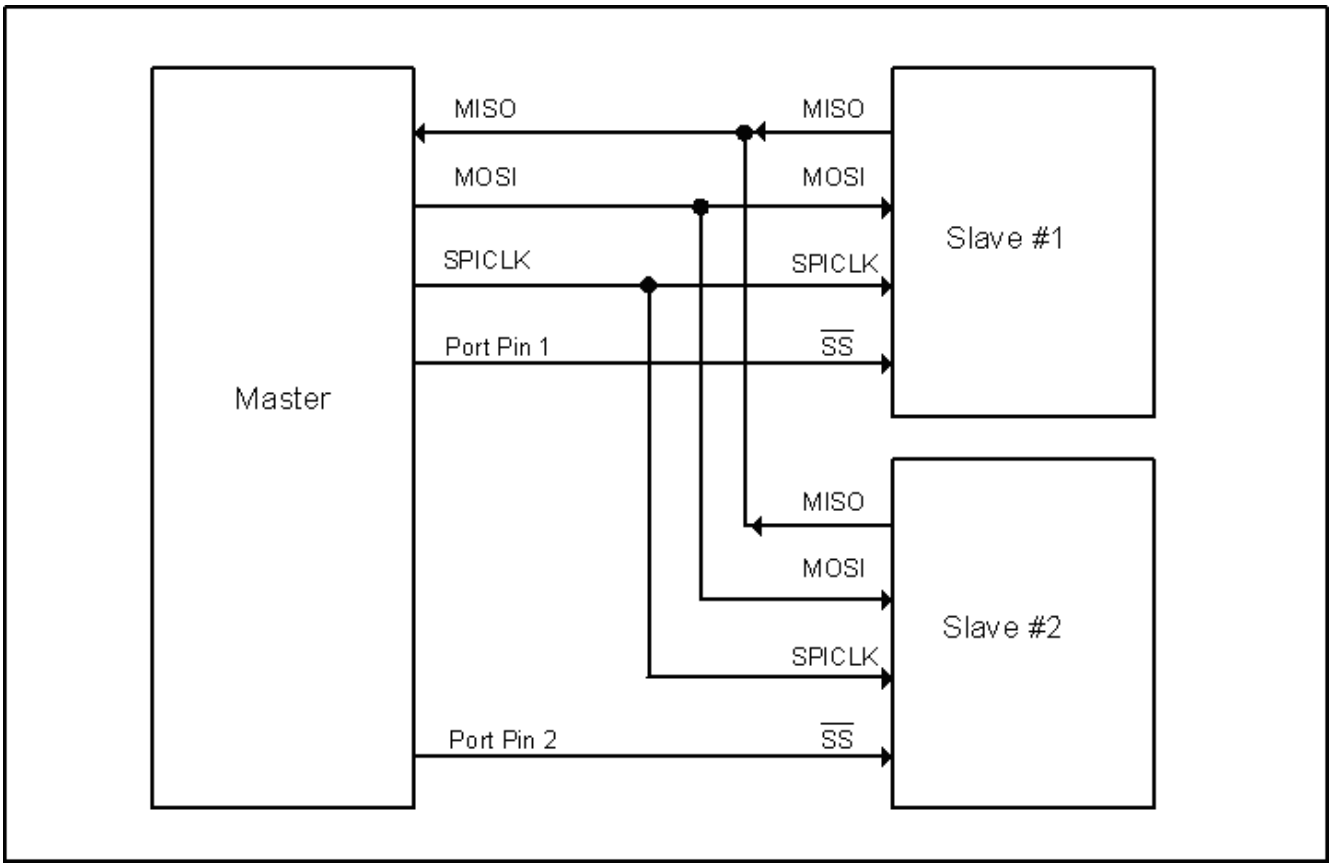
Figure 15-3. SPI dual device configuration, where either can be a master or a slave



15.1.3 Single Master & Multiple Slaves

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

Figure 15-4. SPI single master multiple slaves configuration



15.2 Configuring the SPI

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

Table 15-1. SPI Master and Slave Selection

SPEN (SPCTL.6)	SSIG (SPCTL.7)	/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".

15.3 Additional Considerations for a Slave

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

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

15.4 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.

15.5 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.

15.6 Write Collision

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

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

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

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

15.7 SPI Clock Rate Select

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

Table 15-2. SPI Serial Clock Rates

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

Where, Fosc is the system clock.

15.8 Data Mode

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

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

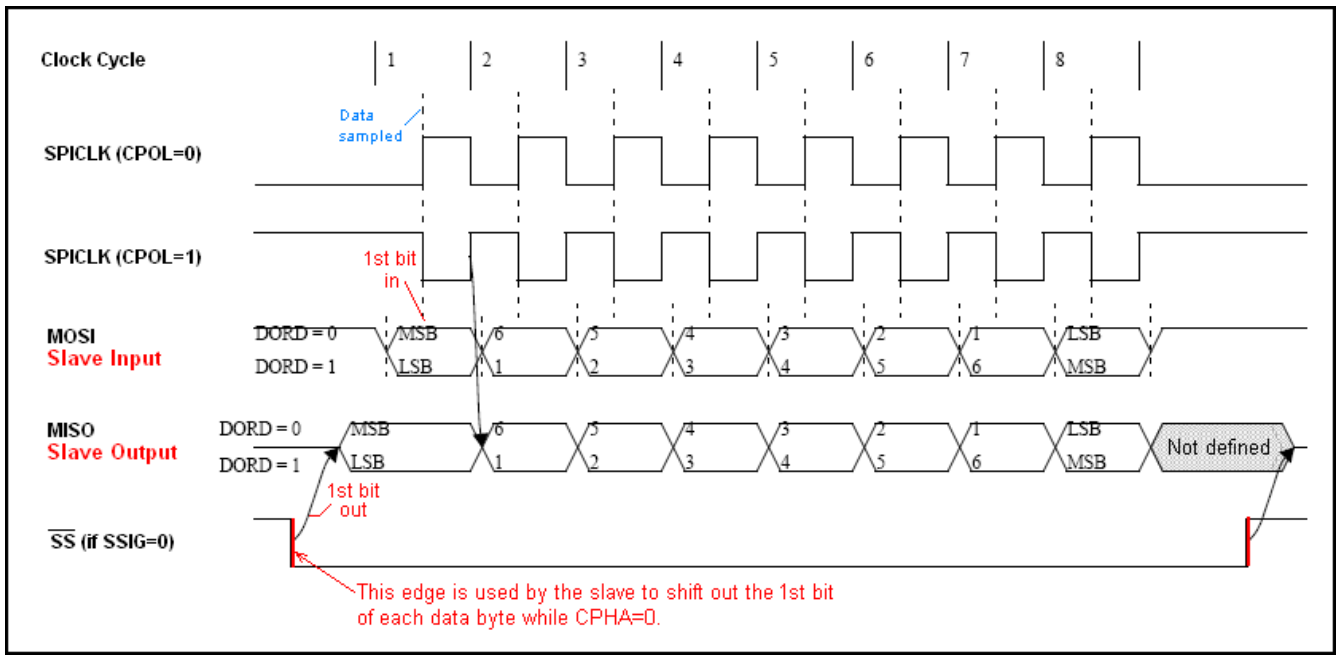


Figure 15-6. Slave Transfer Format with CPHA=1

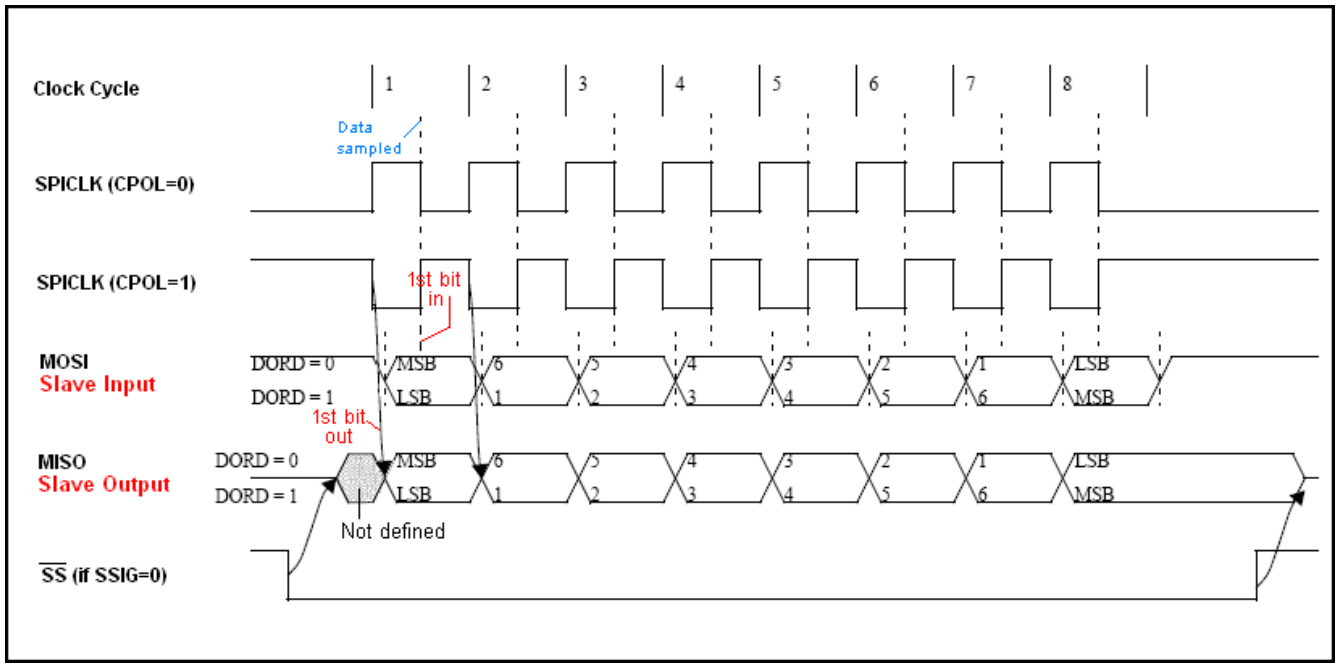


Figure 15-7. SPI Master Transfer Format with CPHA=0

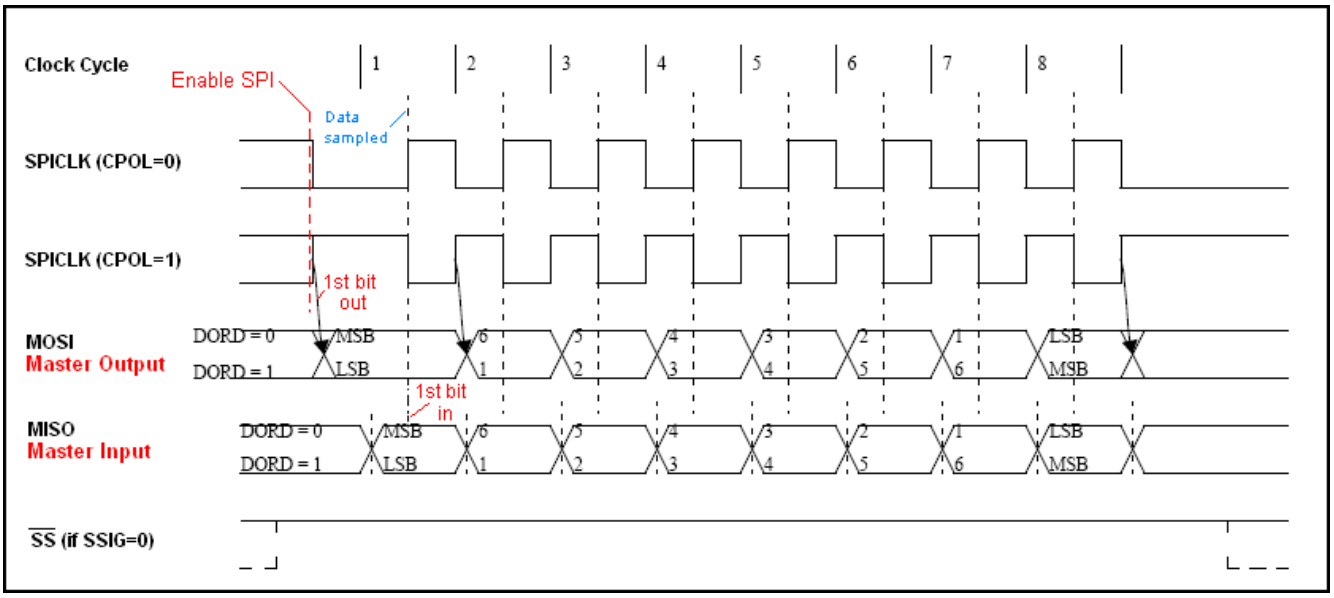
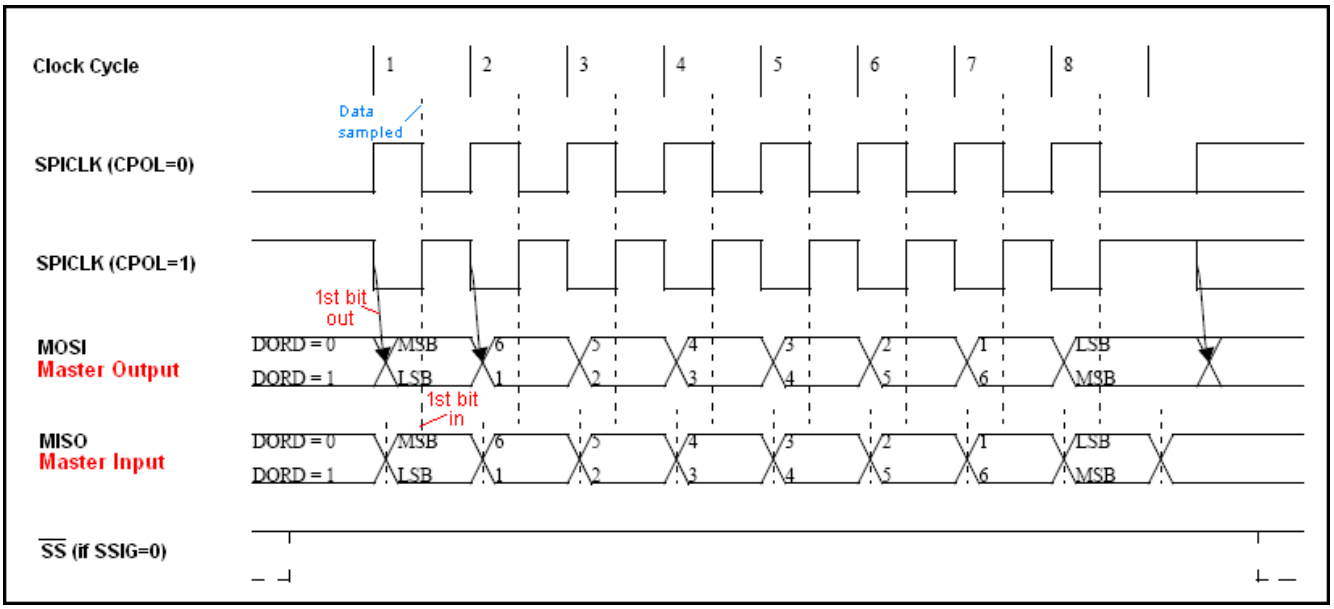


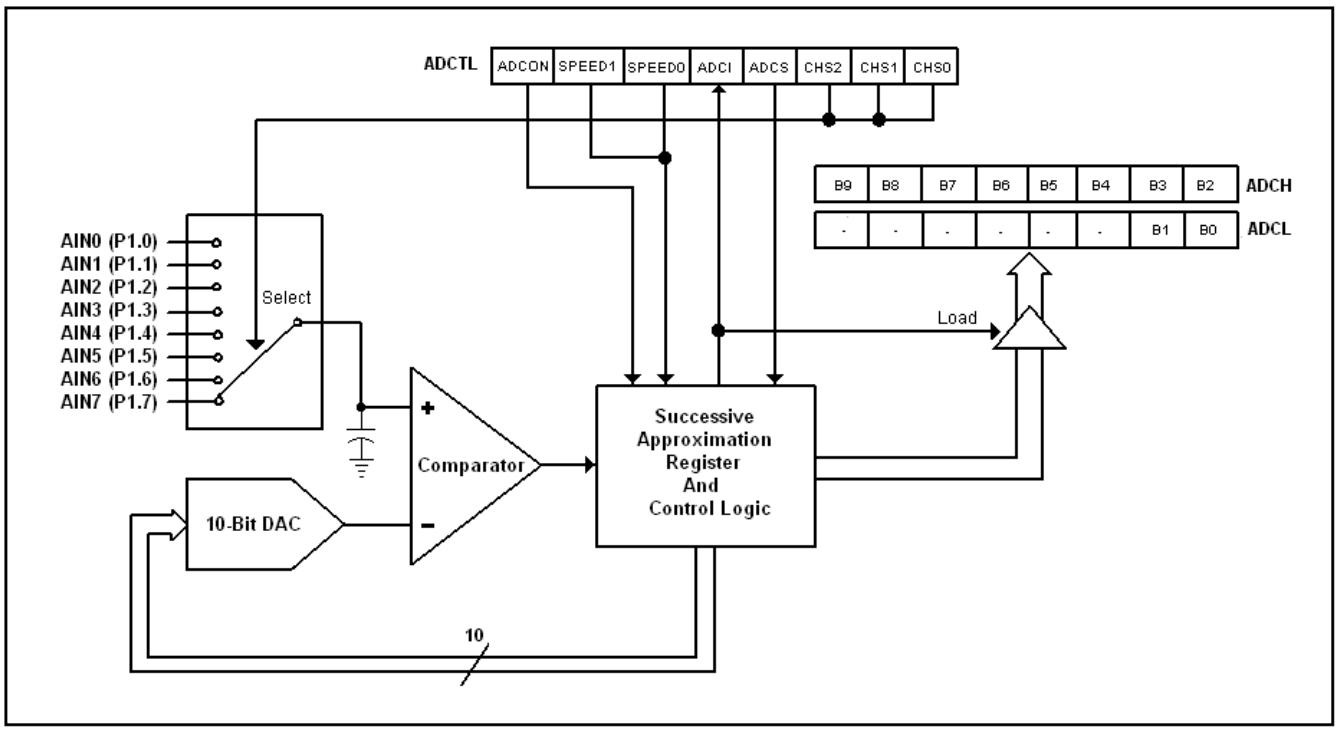
Figure 15-8. SPI Master Transfer Format with CPHA=1



16 A/D Converter

The MPC82G516A has one 10-bit, 8-channel multiplexed inputs analog-to-digital converter, which is implemented with successive approximation register (SAR) approach. Figure 16-1 shows the block diagram of the A/D converter. The eight multiplexed analog inputs share input pins with Port 1. The multiplexed input has a sample and hold circuit to feed the input analog voltage to the comparator input, and the output of the comparator is fed to the SAR for successive approximating operation.

Figure 16-1. ADC Block Diagram



16.1 ADC Control Registers

ADCTL (Address=C5H, ADC Control Register, Reset Value=0000,0000B)

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

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

SPEED1 and SPEED0: A-to-D conversion speed selection bits.
(0,0): 1080 clock cycles are taken for a conversion.
(0,1): 540 clock cycles are taken for a conversion.
(1,0): 360 clock cycles are taken for a conversion.
(1,1): 270 clock cycles are taken for a conversion.

Note 1 clock cycle time is equal to 1/Fosc.

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

ADCI: ADC interrupt flag.
This flag is set when an A/D conversion is completed. An interrupt is invoked if it is enabled. The flag should be cleared by software.

CHS2, CHS1 and CHS0: Multiplexed input channel selection bits.
(0,0,0): select AIN0 (P1.0) as the analog input
(0,0,1): select AIN1 (P1.1) as the analog input
(0,1,0): select AIN2 (P1.2) as the analog input
(0,1,1): select AIN3 (P1.3) as the analog input

(1,0,0): select AIN4 (P1.4) as the analog input
(1,0,1): select AIN5 (P1.5) as the analog input
(1,1,0): select AIN6 (P1.6) as the analog input
(1,1,1): select AIN7 (P1.7) as the analog input

AUXR (Address=8EH, Auxiliary Register, Reset Value=0000,xx0xB)

7	6	5	4	3	2	1	0
URTS	ADRJ	P41ALE	P35ALE	-	-	EXTRAM	-

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

If ADRJ=0

ADCH (Address=C6H, ADC Result High-byte Register, Reset Value=xxH)

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

ADCL (Address=BEH, ADC Result Low-byte Register, Reset Value=xxH)

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

If ADRJ=1

ADCH (Address=C6H, ADC Result High-byte Register, Reset Value=xxH)

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

ADCL (Address=BEH, ADC Result Low-byte Register, Reset Value=xxH)

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

16.2 ADC Operation

For consideration of pin-compatible with the standard 8051 MCU, the ADC hardware cannot have separated input pins for internal positive (Vref+) and negative (Vref-) reference voltages. The Vref+ and Vref- inputs have been internally tied to VDD pin and ground, respectively. So, the full-scale voltage Vref+ – Vref- becomes VDD.

The A/D conversion result can be calculated from the following formula:

$$\text{ADC Result} = 1024 \times \frac{\text{Vin} - \text{Vref-}}{\text{Vref+} - \text{Vref-}} = \frac{\text{AIN}x\ \text{Analog Input Voltage}}{\text{VDD Voltage}}$$

Where, Vin is the analog input voltage and x = 0~7 (any pin of AIN0~AIN7).

The input analog voltage should be between Vref+ and Vref-, i.e., VDD and ground. For input voltages between *Vref-* and *Vref- + 1/2 LSB*, the 10-bit conversion result will be 00,0000,0000B = 000H. For input voltages between *Vref+ – 3/2 LSB* and *Vref+*, the conversion result will be 11,1111,1111B = 3FFH. Where:

$$1\ \text{LSB} = \frac{\text{Vref+} - \text{Vref-}}{1024} = \frac{\text{VDD}}{1024}$$

Prior to using the ADC function, the user should:

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

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

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

16.3 Sample Code for ADC

```
start:
        ;...
        ;...

        MOV    ADCTL,#0E2h      ;ADCON=1, turn on ADC hardware
                                ;(SPEED1,SPEED0)=(1,1), Conv. Time= 270 clock cycles
                                ;select AIN0 (P1.2) as analog input

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

        ANL    AUXR,#10111111B  ;ADRJ=0: ADCH contains B9~B2; ADCL contains B1,B0

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

        ORL    ADCTL,#00001000B ;ADCS=1 ➔Start A-to-D conversion
wait_loop:
        MOV    ACC,ADCTL
        JNB    ACC.4,wait_loop  ;wait until ADCI=1 ➔conversion completed

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

        ;...
        ;...
```

16.4 Notes on ADC

Several notes on ADC are listed below.

16.4.1 A/D Conversion Time

The user can select the appropriate conversion speed according to the frequency of the analog input signal. For example, if Fosc=10MHz and a conversion speed of 270 clock cycles is selected, then the frequency of the analog input should be no more than 37KHz to maintain the conversion accuracy. (Conversion time = 1/10MHz x 270 = 27us, so the conversion speed = 1/27us = 37KHz.)

16.4.2 I/O Pin Used with ADC Function

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

16.4.3 Idle and Power-Down Mode

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

16.4.4 Requirements on VDD Power Supply

As previously described, the Vref+ and Vref- are internally tied to VDD pin and ground, respectively, and the operating voltage on VDD pin may be 2.7V~5.5V (in 5V application system) or 2.4V~3.6V (in 3.3V application system), so the full-scale voltage, Vref+ – Vref- = VDD, is not fixed. However, the conversion formula is kept unchanged. That is, the same Vin will get a different conversion result when the VDD voltage is changed, and therefore the VDD must be kept fixed for an absolute conversion. The user should pay attention to it!

Since the VDD functions as the positive reference Vref+, the user should keep VDD as pure as possible in order to achieve the best ADC performance.

17 Keypad Interrupt

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

There are three SFRs used for this function. The Keypad Interrupt Mask Register (KBMASK) is used to define which input pins connected to Port 2 are enabled to trigger the interrupt. The Keypad Pattern Register (KBPATN) is used to define a pattern that is compared to the value of Port 2. The Keypad Interrupt Flag (KBIF) in the Keypad Interrupt Control Register (KBCON) is set by hardware when the condition is matched. An interrupt will be generated if it has been enabled by setting the EKBI bit in AUXIE register and EA=1. The PATN_SEL bit in the Keypad Interrupt Control Register (KBCON) is used to define "equal" or "not-equal" for the comparison.

In order to use the Keypad Interrupt as the "Keyboard" Interrupt, the user needs to set KBPATN=0xFF and PATN_SEL=0 (not equal), then any key connected to Port 2 which is enabled by KBMASK register will cause the hardware to set the interrupt flag KBIF and generate an interrupt if it has been enabled. The interrupt may wake up the CPU from Idle mode or Power-Down mode. This feature is particularly useful in handheld, battery powered systems that need to carefully manage power consumption but also need to be convenient to use.

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

KBPATN (Address=D5H, Keypad Pattern Register, Reset Value=1111,1111B)

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

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

KBCON (Address=D6H, Keypad Control Register, Reset Value=xxxx,xx00B)

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

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

KBIF:
Keypad Interrupt Flag. Set when Port 2 matches user defined conditions specified in KBPATN, KBMASK, and PATN_SEL. Needs to be cleared by software by writing "0".

KBMASK (Address=D7H, Keypad Interrupt Mask Register, Reset Value=0000,0000B)

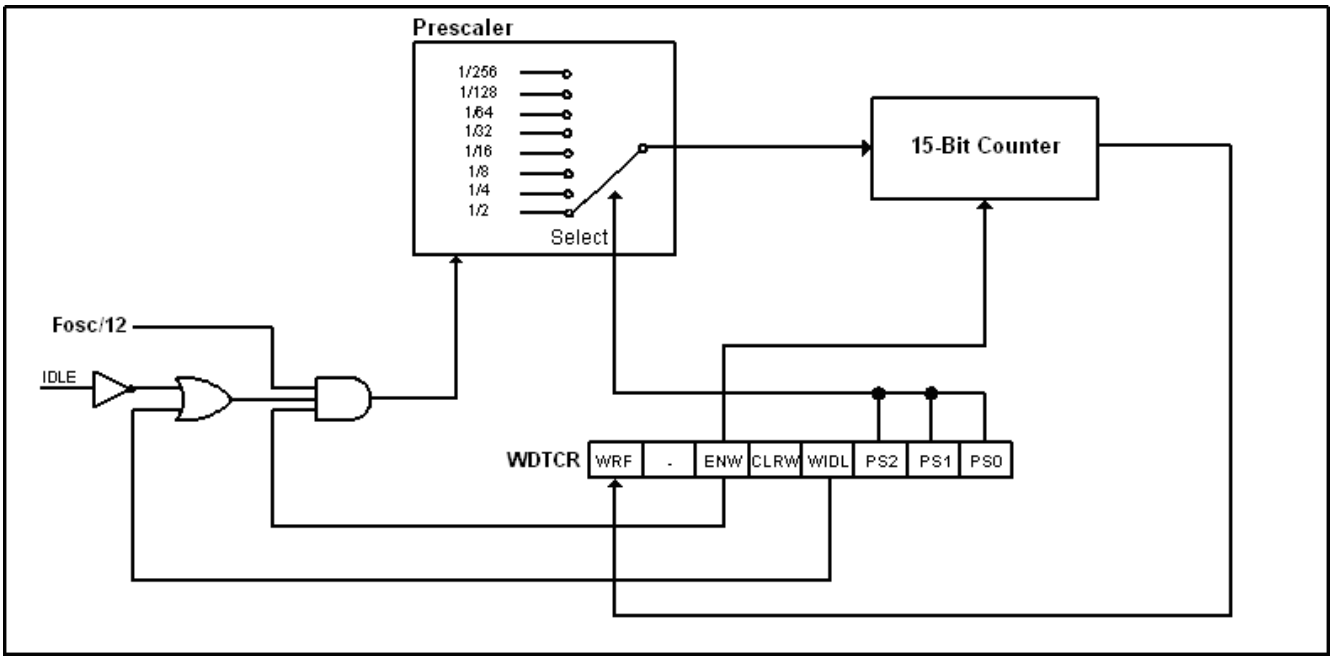
7	6	5	4	3	2	1	0
KBMASK.7	KBMASK.6	KBMASK.5	KBMASK.4	KBMASK.3	KBMASK.2	KBMASK.1	KBMASK.0

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

18 Watchdog Timer

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

Figure 18-1. WDT Block Diagram



18.1 WDT Control Register

WDTCR (Address=E1H, Watch-Dog-Timer Control Register, Power-on Reset Value=0x00,0000B)

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

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

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

CLRW: WDT clear bit. Writing "1" to this bit will clear the 15-bit WDT counter to 0000H. Note this bit 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~PS0: Prescaler select bits.

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

18.2 WDT Operation

The WDT is by default disabled after power-on reset. To enable the WDT, the user must set the ENW bit. When the WDT is enabled, the user needs to service it by setting the CLW bit to clear the WDT counter and avoid an overflow. The 15-bit WDT counter will overflow when it reaches 32767 (7FFFH) and this will reset the device. And, when the WDT is enabled, it will increment every 12 system clock cycles (12/Fosc) while the oscillator is running. This means the user must clear the WDT counter at least every 32767 x12 system cock cycles.

The WDT in this device is one-time enabled. The so-called "one-time enabled" means: *Once the WDT is enabled by setting ENW bit, there is no way to disable it except through power-on reset, which will clear the ENW bit.* And, the WDTCR register will keep the previous programmed value unchanged after any resets (including hardware reset, software reset and WDT reset) except the power-on reset. For example, if the WDTCR is 0x2D, it still keeps at 0x2D rather than 0x00 after resets. Only power-on reset can initialize it to 0x00. In other words, the WDT can only be disabled by a power-on reset. Thus it is called "one-time enabled" WDT.

The WDT overflow period is determined by the formula:

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

Table 18-1 shows the WDT overflow period for MCU running at 6/12/24MHz. The period is the maximum interval for the user to clear the WDT to prevent from chip reset.

Table 18-1. WDT Overflow Period

PS2	PS1	PS0	Prescaler value	Fosc=6MHz	Fosc=12MHz	Fosc=24MHz
0	0	0	2	131.072 ms	65.536 ms	32.768 ms
0	0	1	4	262.144 ms	131.072 ms	65.536 ms
0	1	0	8	524.288 ms	262.144 ms	131.072 ms
0	1	1	16	1.048 s	524.288 ms	262.144 ms
1	0	0	32	2.097 s	1.048 s	524.288 ms
1	0	1	64	4.194 s	2.097 s	1.048 s
1	1	0	128	8.389 s	4.194 s	2.097 s
1	1	1	256	16.778 s	8.389 s	4.194 s

18.3 Sample Code for WDT

Condition: WDT Overflow Period = 1.048 seconds @Fosc=12MHz

```
WDTCR_buf DATA  30h             ;declare a buffer for WDTCR register
                                ;(because WDTCR is a Write-only register)
start:
         ;...
         MOV    WDTCR_buf,#00h  ;initialize the WDTCR buffer

         ORL    WDTCR_buf,#04h  ;PS2=1
         ANL    WDTCR_buf,#0FCh ;PS1=0,PS0=0
         MOV    WDTCR,WDTCR_buf ;write to WDTCR ➔(PS2,PS1,PS0)=(1,0,0), prescaler=32

         ORL    WDTCR_buf,#20h  ;ENW=1
         MOV    WDTCR,WDTCR_buf ;write to WDTCR register ➔enable WDT

main_loop:
         ORL    WDTCR_buf,#10h  ;CLRW=0
         MOV    WDTCR,WDTCR_buf ;write to WDTCR register ➔clear WDT counter
         ;...
         ;...
         JMP    main_loop
```

18.4 WDT during Power-Down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode the user does not need to service the WDT. There are 3 methods of exiting Power-down mode: by a hardware reset, via an external interrupt (/INT0~/INT3) or Keypad interrupt which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally should whenever the device is reset. Exiting Power-down with an external interrupt or Keypad interrupt is significantly different. When the interrupt is serviced just after exiting the power-down, to prevent the WDT from resetting the device, it is suggested that the WDT counter should be cleared during the interrupt service routine.

Of course, to ensure that the WDT does not overflow within a little time after exiting of power-down, it is better to clear the WDT counter just before entering power-down.

In the Idle mode, the oscillator continues to run. The user may either set the WIDL bit to have WDT keep working or clear the WIDL bit to have WDT stop while the MCU is in the Idle mode. For the former case, to prevent the WDT from overflowing while in Idle mode, the user should always set up a timer that will periodically exit Idle, service the WDT, and re-enter Idle mode.

18.5 WDT Initialized by Hardware Option

Besides 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. (Refer to Section 25: MCU's Hardware Option.)

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.

WDTCR (Watch-Dog-Timer Control Register)

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

19 Interrupt System

The MPC82G516A has 14 interrupt sources with a four-level interrupt structure. There are several SFRs associated with the four-level interrupt. They are the IE, IP, IPH, AUXIE, AUXIP, AUXIPH, XICON and TCON. The IPH (Interrupt Priority High) and AUXIPH (Auxiliary Interrupt Priority High) registers make the four-level interrupt structure possible. The four priority level interrupt structure allows great flexibility in handling these interrupt sources.

19.1 Interrupt Sources

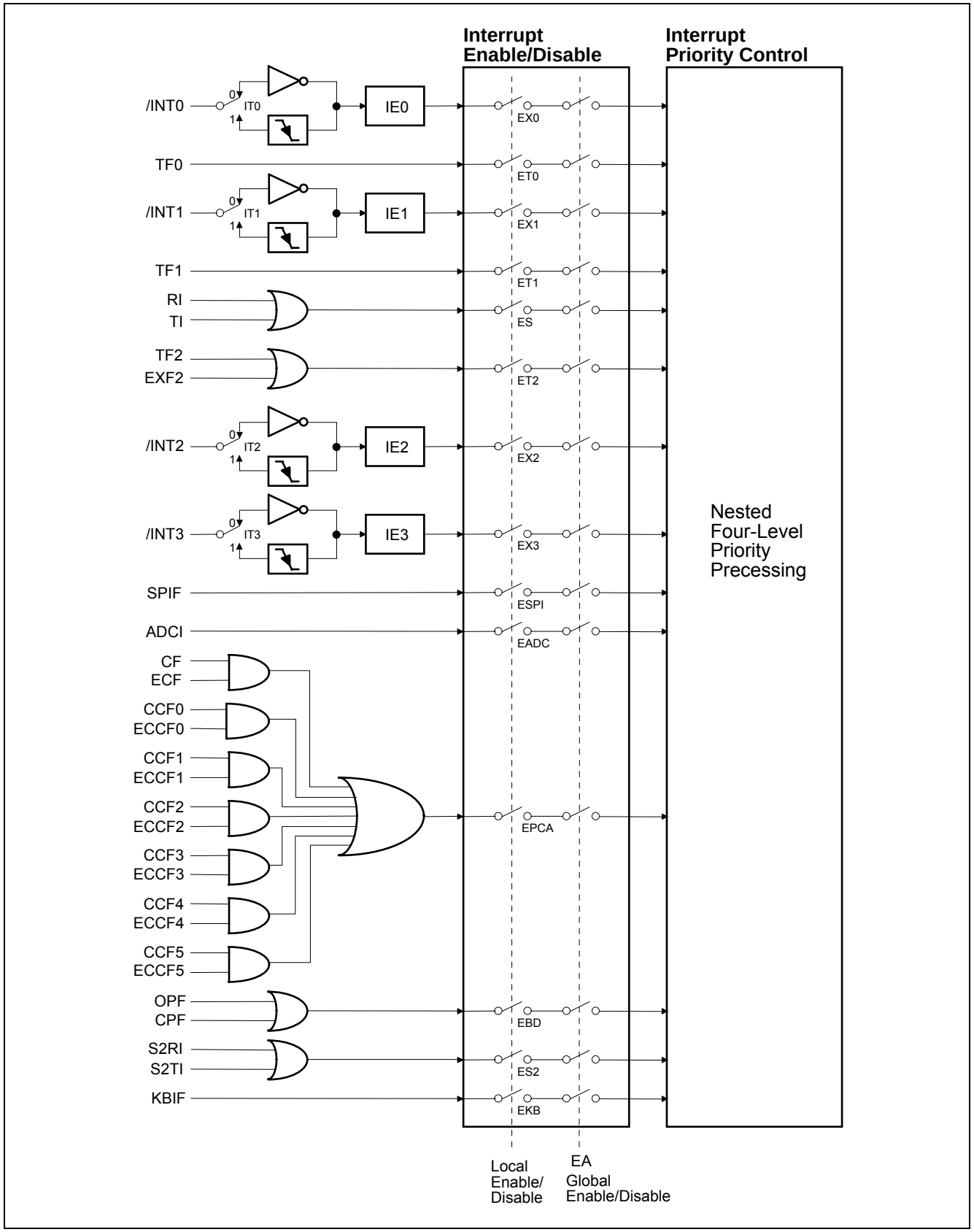
Table 19-1 lists all the interrupt sources. The 'Request Bits' are the interrupt flags that will generate an interrupt if it is enabled by setting the 'Enable Bit'. Of course, the global enable bit EA (in IE register) should have been set previously. The 'Request Bits' can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled in software. The 'Priority Bits' determine the priority level for each interrupt. The 'Priority within Level' is the polling sequence used to resolve simultaneous requests of the same priority level. The 'Vector Address' is the entry point of an interrupt service routine in the program memory.

Figure 19-1 shown the interrupt system. Each of these interrupts will be briefly described in the following sections.

Table 19-1. Interrupt Sources

No	Source Name	Enable Bit	Request Bits	Priority Bits	Priority within Level	Vector Address
#1	External Interrupt, INT0	EX0	IE0	PX0H, PX0	(Highest)	0003H
#2	Timer 0	ET0	TF0	PT0H, PT0	.	000BH
#3	External Interrupt, INT1	EX1	IE1	PX1H, PX1	.	0013H
#4	Timer 1	ET1	TF1	PT1H, PT1	.	001BH
#5	Serial Port	ES	RI, TI	PSH, PS	.	0023H
#6	Timer 2	ET2	TF2, EXF2	PT2H, PT2	.	002BH
#7	External Interrupt, INT2	EX2	IE2	PX2H, PX2	.	0033H
#8	External Interrupt, INT3	EX3	IE3	PX3H, PX3	.	003BH
#9	SPI	ESPI	SPIF	PSPIH, PSPI	.	0043H
#10	ADC	EADC	ADCI	PADCH, PADC	.	004BH
#11	PCA	EPCA	CF, CCFn (n=0~5)	PPCAH, PPCA	.	0053H
#12	Brownout Detection	EBD	OPF, CPF	PBDH, PBD	.	005BH
#13	UART2	ES2	S2RI, S2TI	PS2H, PS2	.	0063H
#14	Keypad Interrupt	EKB	KBIF	PKBH, PKB	(Lowest)	006BH

Figure 19-1. Interrupt System



19.2 SFRs Associated with Interrupts

The Special Function Registers associated with the interrupts are shown below.

IE (Address=A8H, Interrupt Enable Register, Reset Value=0x00,0000B)

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

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

ET2: Timer 2 interrupt enable bit.

ES: Serial Port interrupt enable bit.

ET1: Timer 1 interrupt enable bit.

EX1: External interrupt 1 enable bit.

ET0: Timer 0 interrupt enable bit.

EX0: External interrupt 0 enable bit.

IP (Address=B8H, Interrupt Priority Register, Reset Value=xx00,0000B)

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

PT2: Timer 2 interrupt priority bit.

PS: Serial Port interrupt priority bit.

PT1: Timer 1 interrupt priority bit.

PX1: External interrupt 1 priority bit.

PT0: Timer 0 interrupt priority bit.

PX0: External interrupt 0 priority bit.

IPH (Address=B7H, Interrupt Priority High Register, Reset Value=0000,0000B)

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

PX3H: External interrupt 3 priority bit, high.

PX2H: External interrupt 2 priority bit, high.

PT2H: Timer 2 interrupt 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.

AUXIE (Address=ADH, Auxiliary Interrupt Enable Register, Reset Value=xx00,0000B)

7	6	5	4	3	2	1	0
-	-	EKB	ES2	EBD	EPCA	EADC	ESPI

EKB: Keypad interrupt enable bit.

ES2: UART2 interrupt enable bit.

EBD: Brownout Detection interrupt enable bit.

EPCA: PCA interrupt enable bit.

EADC: ADC interrupt enable bit.

ESPI: SPI interrupt enable bit.

AUXIP (Address=AEH, Auxiliary Interrupt Priority Register, Reset Value=xx00,0000B)

7	6	5	4	3	2	1	0
-	-	PKB	PS2	PBD	PPCA	PADC	PSPI

PKB: Keypad interrupt priority bit.

PS2: UART2 interrupt priority bit.

PBD: Brownout Detection interrupt priority bit.

PPCA: PCA interrupt priority bit.

PADC: ADC interrupt priority bit.

PSPI: SPI interrupt priority bit.

AUXIPH (Address=AFH, Auxiliary Interrupt Priority High Register, Reset Value=xx00,0000B)

7	6	5	4	3	2	1	0
-	-	PKBH	PS2H	PBDH	PPCAH	PADCH	PSPIH

PKBH: Keypad interrupt priority bit, high.

PS2H: UART2 interrupt priority bit, high.

PBDH: Brownout Detection interrupt priority bit, high.

PPCAH: PCA interrupt 1 priority bit, high.

PADCH: ADC interrupt priority bit, high.

PSPIH: SPI interrupt 0 priority bit, high.

XICON (Address=C0H, External Interrupt Control Register, Reset Value=0000,0000B)

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

PX3: External interrupt 3 priority bit.

EX3: External interrupt 3 enable bit.

IE3: External interrupt 3 interrupt flag.

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

PX2: External interrupt 2 priority bit.

EX2: External interrupt 2 enable bit.

IE2: External interrupt 2 interrupt flag.

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

TCON (Address=88H, Timer/Counter Control Register, Reset Value=0000,0000B)

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

IE1: Interrupt 1 flag. Set by hardware when external interrupt 1 edge is detected (transmitted or level-activated). Cleared when interrupt processed only if transition activated.

IT1: Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupt 1.

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

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

19.3 Interrupt Enable

Each of these interrupt sources can be individually enabled or disabled by setting or clearing an interrupt enable bit in the registers IE, AUXIE and XICON. Note that IE also contains a global disable bit, EA. If EA is set to '1', the interrupts are individually enabled or disabled by their corresponding enable bits. If EA is cleared to '0', all interrupts are disabled.

19.4 Interrupt Priority

The priority scheme for servicing the interrupts is the same as that for the 80C51, except there are four interrupt levels rather than two as on the 80C51. The Priority Bits (see Table 19-1) determine the priority level of each interrupt. Table 19-2, as an illustration example using External Interrupt 0, shows the bit values and priority levels associated with each combination.

Table 19-2. Four Priority Level of External Interrupt 0

Priority Bits		Priority Level
PX0H	PX0	
0	0	Level 0 (Lowest)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (Highest)

An interrupt will be serviced as long as an interrupt of equal or higher priority is not already being serviced. If an interrupt of equal or higher level priority is being serviced, it won't be stopped and the new interrupt will wait until it is finished. If a lower priority level interrupt is being serviced, it will be stopped and the new interrupt will be serviced immediately. When the new interrupt is finished, the lower priority level interrupt that was stopped will be completed. In other words, a priority interrupt can itself be interrupted by a higher priority interrupt, but not by another equal or lower priority interrupt.

If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the same priority level are received simultaneously, the "Priority within Level" determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence. Note the 'Priority within Level' is only used to resolve simultaneous requests of the same priority level.

19.5 How Interrupts are Handled

The interrupt flags are sampled every instruction cycle. The samples are polled during the following instruction cycle. If one of the interrupt flags was in a set condition in the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the registers associated the interrupts.

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

The polling cycle is repeated with each instruction cycle, and the values polled are the values that were present in the previous instruction cycle. If the interrupt flag for a level-sensitive external interrupt is active but not being responded to for one of the above conditions and is not still active when the blocking condition is removed, the denied interrupt will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle is new.

The processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate

servicing routine. In some cases it also clears the flag that generated the interrupt, and in other cases it doesn't. It never clears the interrupt flags of Timer2, Serial Port, PCA, Brownout Detection and UART2. This has to be done in the user's software. It clears an external interrupt flag (IE0, IE1, IE2 or IE3) only if it was transition-activated. The hardware-generated LCALL pushes the contents of the Program Counter onto the stack (but it does not save the PSW) and reloads the PC with an Vector Address that depends on the source of the interrupt being vectored to, as shown in Table 19-1.

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stack and reloads the Program Counter. Execution of the interrupted program continues from where it left off.

Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking the interrupt was still in progress.
Note that the starting addresses of consecutive interrupt service routines are only 8 bytes apart. That means if consecutive interrupts are being used (IE0 and TF0, for example, or TF0 and IE1), and if the first interrupt routine is more than 7 bytes long, then that routine will have to execute a jump to some other memory location where the service routine can be completed without overlapping the starting address of the next interrupt routine

19.6 External Interrupts

The external sources includes /INT0, /INT1, /INT2 and /INT3, which can each be either level-activated (low-level) or transition-activated (falling-edge), depending on bits IT0, IT1, IT2 and IT3 in registers TCON and XICON. If ITx = 0, external interrupt x is triggered by a detected low at the INTx pin. If ITx = 1, external interrupt x is negative edge-triggered. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON and IE2 and IE3 in XICON. These flags are cleared by hardware when the service routine is vectored to only if the interrupt was transition-activated. If the interrupt was level-activated, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

Since the external interrupt pins are sampled once each instruction cycle, an input high or low should hold for at least one oscillator period to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one cycle, and then hold it low for at least one cycle to ensure that the transition is seen so that interrupt request flag IEx will be set. IEx will be automatically cleared by the CPU when the service routine is called.

If external interrupt is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated right again.

19.7 Single-Step Operation

The 80C51 interrupt structure allows single-step execution with very little software overhead. As previously noted, an interrupt request will not be responded to while an interrupt of equal or higher priority level is still in progress, nor will it be responded to after RETI until at least one other instruction has been executed. Thus, once an interrupt routine has been entered, it cannot be re-entered until at least one instruction of the interrupted program is executed. One way to use this feature for single-step operation is to program one of the external interrupts (e.g., INT0) to be level-activated. The service routine for the interrupt will terminate with the following code:

```
JNB   P3.2,$   ;Wait Till INT0 Goes High
JB    P3.2,$   ;Wait Till INT0 Goes Low
RETI           ;Go Back and Execute One Instruction
```

Now if the INT0 pin, which is also the P3.2 pin, is held normally low, the CPU will go right into the External Interrupt 0 routine and stay there until INT0 is pulsed (from low to high to low). Then it will execute RETI, go back to the task program, execute one instruction, and immediately re-enter the External Interrupt 0 routine to await the next pulsing of P3.2. One step of the task program is executed each time P3.2 is pulsed.

20 ISP, IAP and ICP

The embedded Flash memory of the MPC82G516A can be programmed using the following methods.

(1) The traditional parallel programming method: generally for a Universal Programmer (not described here).
(2) In-System Programming method (ISP): under control of the loader program.
(3) In-Application Programming method (IAP): under control of the user's application program.
(4) In-Circuit Programming (ICP): under control of the proprietary ICP Programmer (see Section 20.4.1).

Refer to Figure 20-1 for the MPC82G516A Flash Configuration. The Flash of MPC82G516A can be partitioned into AP-memory, IAP-memory and ISP-memory. AP-memory is used to store the user's application program; IAP-memory is used to store the non-volatile application data; and, ISP-memory is used to store the loader program for In-System Programming.

The traditional parallel programming and ICP can program anywhere in the MCU, including the whole Flash and MCU's Hardware Option. The ISP and IAP can only program some specific area of the Flash; the ISP can program both AP-memory and IAP-memory while the IAP can only program the IAP-memory. Table 20-1 shows the comparison between the various programming methods listed above.

Table 20-1. Comparison between the Various Programming Methods

Items	**Parallel Programming**	**ISP**	**IAP**	**ICP**
Erase/Program/Verify	Yes	Yes	Yes	Yes
Programming Area	Whole Flash & MCU's Hardware Option	AP-memory & IAP-memory	IAP-memory	Whole Flash & MCU's Hardware Option
Controlled by Hardware or Software?	Hardware-Controlled	Software-Controlled	Software-Controlled	Hardware-Controlled
Programming Interface	Parallel Interface	Serial Interface Using DTA (P3.1)	None	Serial Interface Using Dedicated SDA & SCL
Preparation for the Programming	None	Loader Program Pre-programmed & HWBS enabled	None	None
Programming Tool	Universal Programmer or "Megawin 8051 Writer"	"Megawin ISP Programmer"	None	"Megawin ICP Programmer"

Why ISP?

ISP makes it possible to update the user's application program (in AP-memory) and non-volatile application data (in IAP-memory) without removing the MCU chip from the actual end product. This useful capability makes a wide range of field-update applications possible. (Note ISP needs the loader program pre-programmed in the ISP-memory.)

Why IAP?

The IAP-memory provides a non-volatile storage for the applications which need to keep its application data not lost after the system is powered off. So, there is no need of an extra serial EEPROM such as the 93C46 or 24C01 devices.

Why ICP?

ICP makes it possible to update anywhere in the MCU (including the whole Flash and MCU's Hardware Option) without removing the MCU chip from the actual end product. Like the ISP, it also makes a wide range of field-update applications possible.

20.1 Embedded Flash

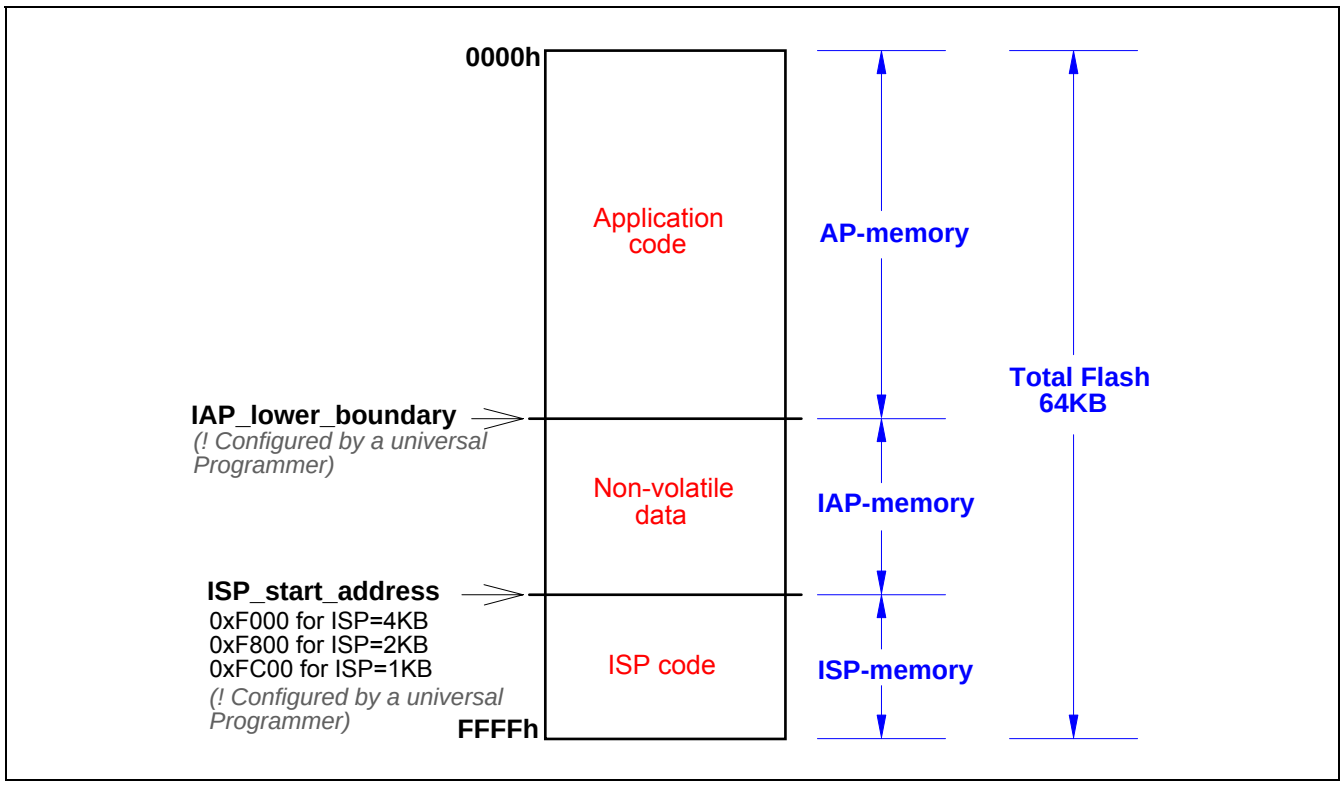
20.1.1 Flash Features

- The Flash can only perform '*page*' erasing, not byte erasing. And, the erased page data will become all 0xFF.
- Only the byte with value of 0xFF can programmed into a non-0xFF byte. Any non-0xFF byte can not be reversely programmed into a 0xFF byte unless using the page erasing.
- Each page has 512 bytes, and the page address is always located at 0x0200*****N, where N (=0,1,2,3,..) means the N^{th} page.
- Endurance: 20,000 Erase/Write Cycles.

20.1.2 Flash Configuration

Figure 20-1 shows the Flash configuration of MPC82G516A. The Flash an be partitioned into AP-memory, IAP-memory and ISP-memory. AP-memory is used to store the user's application program; IAP-memory is used to store the non-volatile application data; and, ISP-memory is used to store the loader program for In-System Programming. The total Flash size is 64K bytes, where the space of IAP-memory and ISP-memory can be configured by a Universal Programmer, the "Megawin 8051 Writer" or the "Megawin 8051 ICP Programmer" (see Section 20.4.1).

Figure 20-1. Flash Configuration



20.2 ISP Operation

In general, the user needn't know how ISP operates because Megawin has provided the standard ISP tool (see Section 20.2.5). For the user who wants to design his own ISP operation, this section includes all the necessary technical information for ISP.

20.2.1 SFRs for ISP

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

ISPCR (Address=E7H, ISP Control Register, Reset Value=000x,x000B)

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

ISPEN: Set to enable ISP function.

SWBS: Software boot select. Set/clear to select booting from ISP-memory/AP-memory for software reset.

SWRST: Write '1' to this bit to trigger a software reset.

CKS2~CKS0: Configure ISP timing according to the oscillator frequency, see Table 20-2.

Table 20-2. ISP Timing Setting

CKS2	CKS1	CKS0	Oscillator Frequency (MHz)
0	0	0	> 24
0	0	1	20 ~ 24
0	1	0	12 ~ 20
0	1	1	6~ 12
1	0	0	3 ~ 6
1	0	1	2 ~ 3
1	1	0	1 ~ 2
1	1	1	< 1

IFMT (Address=E5H, ISP Mode Register, Reset Value=xxxx,x000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	-	-	-	-	MS2	MS1	MS0

MS2, MS1 and MS0: ISP mode select bits, see Table 20-3.

Table 20-3. ISP Mode Select

MS2	MS1	MS0	ISP Mode
0	0	0	Standby
0	0	1	Read
0	1	0	Program
0	1	1	Page Erase

Standby Mode: Keep the ISP hardware in the deactivated state

Page Erase Mode: Erase one page (512 bytes) specified by the page address in [IFADRH,IFADRL]

Program Mode: Program data to Flash specified by the byte address in [IFADRH,IFADRL]

Read Mode: Read data from Flash specified by the byte address in [IFADRH,IFADRL]

IFADRH (Address=E3H, ISP Flash Address High Register, Reset Value=0000,0000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
(High-byte Address, A15~A8)							

IFADRL (Address=E4H, ISP Flash Address Low Register, Reset Value=0000,0000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
(Low-byte Address, A7~A0)							

IFD (Address=E2H, ISP Flash Data Register, Reset Value=0000,0000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
(Data to be programmed or data to be read)							

SCMD (Address=E6H, ISP Sequential Command Register, Reset Value=0000,0000B)

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
(ISP-Triggering Command)							

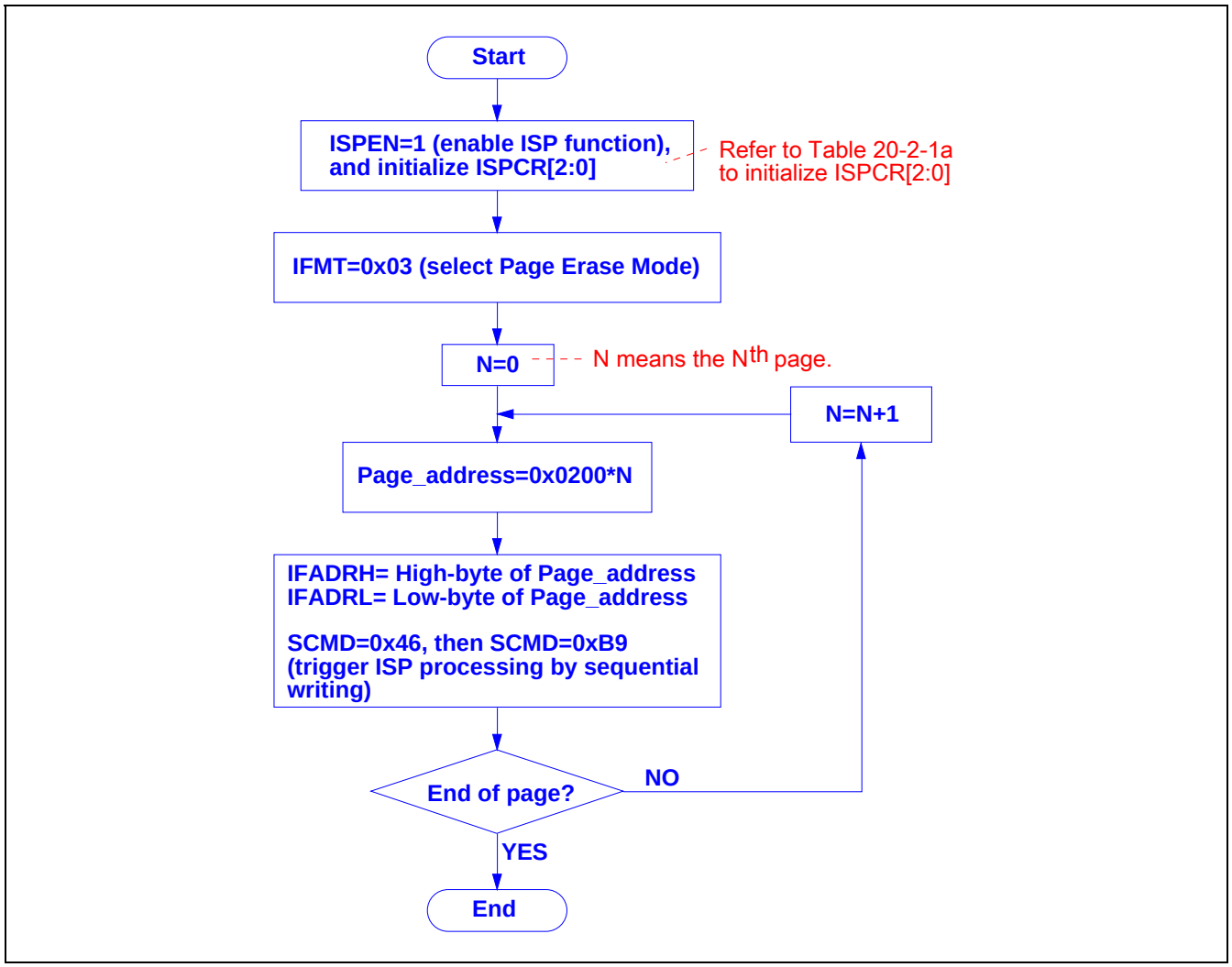
To trigger the ISP processing, write 0x46 then 0xB9 to this register in sequence.

20.2.2 Introduction to the ISP Modes

The ISP modes are used in the *loader program* to program both the AP-memory and IAP-memory. And, they can also be used in *user's application program* to program the IAP-memory. This section shows the flow chart and demo code for the various ISP modes.

20.2.2.1 Flash Page Erase Mode

Figure 20-2. Flow Chart for "Flash Page Erase"

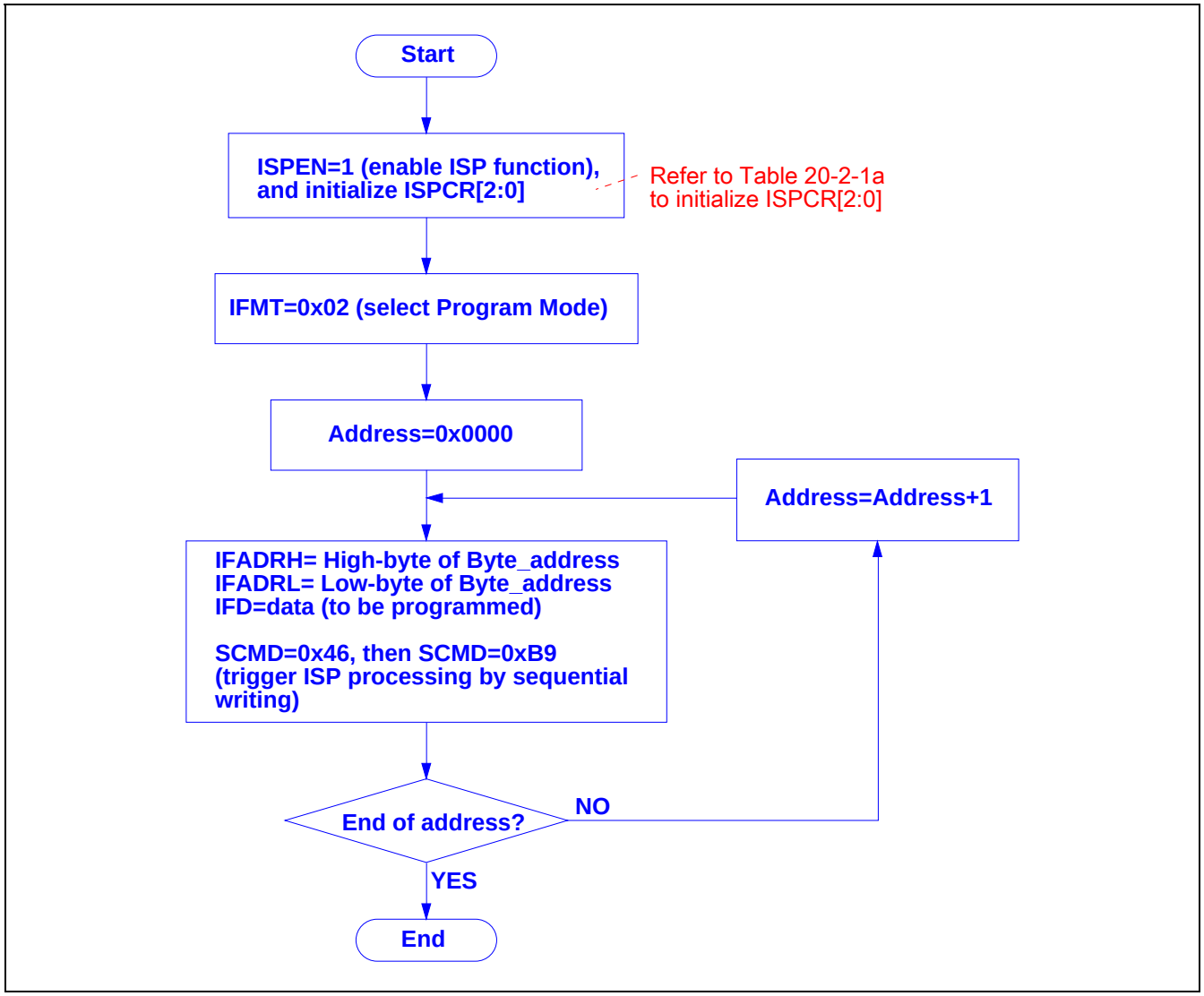


Demo code for triggering the "Page Erase Mode"

```
MOV     ISPCR,#10000011b ;ISPCR.7=1, enable ISP
                         ;ISPCR[2:0]=011, suppose MPC82-series running @11.0592MHz

MOV     IFMT,#03h        ;select Page Erase Mode

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

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

;Now, MCU will halt here until processing completed
```

20.2.2.2 Flash Program Mode

Figure 20-3. Flow Chart for “Flash Program”

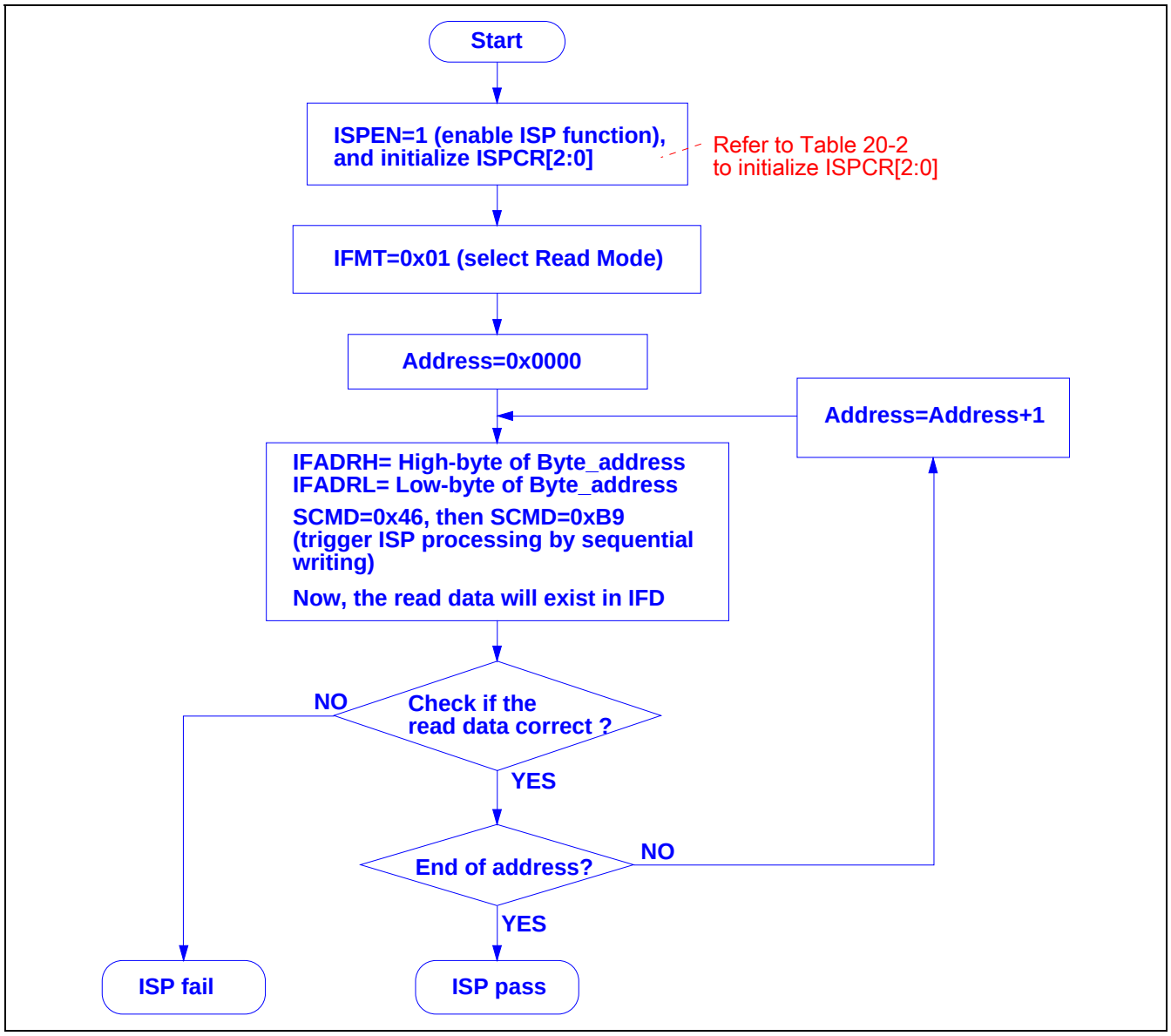


Demo code for triggering the “Program Mode”

```
MOV     ISPCR,#10000011b ;ISPCR.7=1, enable ISP
                         ;ISPCR[2:0]=011, suppose MPC82-series running @11.0592MHz

MOV     IFMT,#02h        ;select Program Mode

MOV     IFADRH,??        ;fill [IFADRH,IFADRL] with byte address
MOV     IFADRL,??        ;
MOV     IFD,??           ;fill IFD with the data to be programmed

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

;Now, MCU will halt here until processing completed
```

20.2.2.3 Flash Read Mode

Figure 20-4. Flow Chart for “Flash Read”



Demo code for triggering the “Read Mode”

```
   MOV    ISPCR,#10000011b ;ISPCR.7=1, enable ISP
                           ;ISPCR[2:0]=011, suppose MPC82-series running @11.0592MHz
   MOV    IFMT,#01h        ;select Read Mode
   MOV    IFADRH,??        ;fill [IFADRH,IFADRL] with byte address
   MOV    IFADRL,??        ;
   MOV    SCMD,#46h        ;trigger ISP processing
   MOV    SCMD,#0B9h       ;
   ;Now, MCU will halt here until processing completed
   MOV    A,IFD            ;now, the read data exists in IFD
   CJNE   A,??,isp_error   ;and, the user can check if the data is correct
   ;...
isp_error:
   JMP    $
```

20.2.3 How to Implement In-System Programming

Before using the ISP function, the user should use a Universal Programmer, the “Megawin 8051 Writer” or the “Megawin 8051 ICP Programmer” (see Section 20.4.1) to do the following configuration:

(1) Properly configure an ISP-memory according to size of the ‘loader program’.

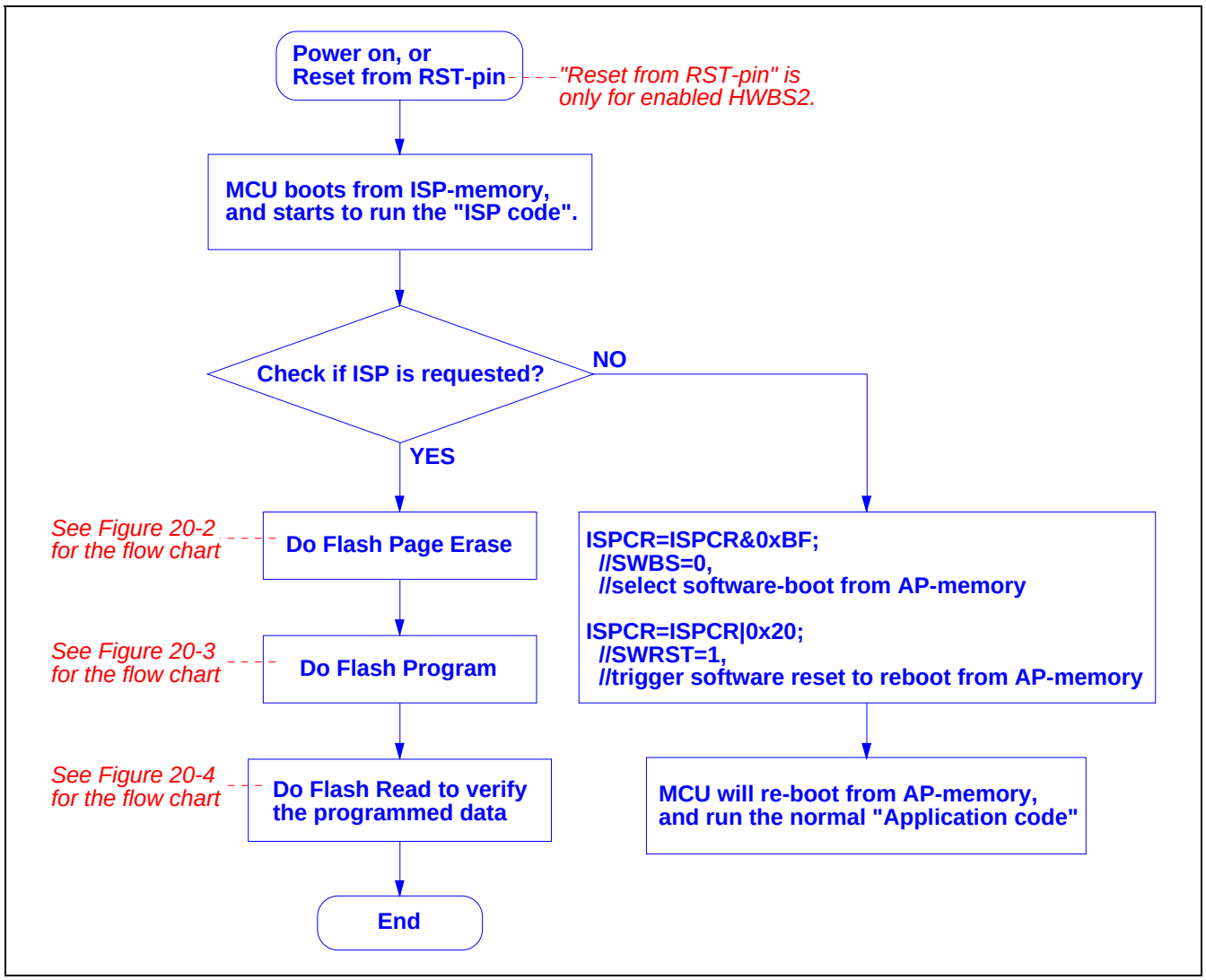
(2) Program the ‘loader program’ (hereafter called ‘**ISP code**’) into this configured ISP-memory.

As we have known, the purpose of the ISP code is to program both AP-memory and IAP-memory. Therefore, ***the MCU must boot from the ISP-memory in order to execute the ISP code***. There are two methods to implement In-System Programming according to how the MCU boots from the ISP-memory.

Method 1: MCU Directly Boots from ISP-memory at Power-up

To make the MCU directly boot from the ISP-memory when it is just powered on, the MCU’s hardware option *HWBS* or *HWBS2* must be enabled. Once *HWBS* or *HWBS2* is enabled, the MCU will always boot from the ISP-memory to execute the ISP code when it is just powered on. The first thing the ISP code should do is to check if there is an ISP request. If there is no ISP requested, the ISP code should trigger a software reset to make the MCU re-boot from the AP-memory to run the user’s application program. See the following flow chart.

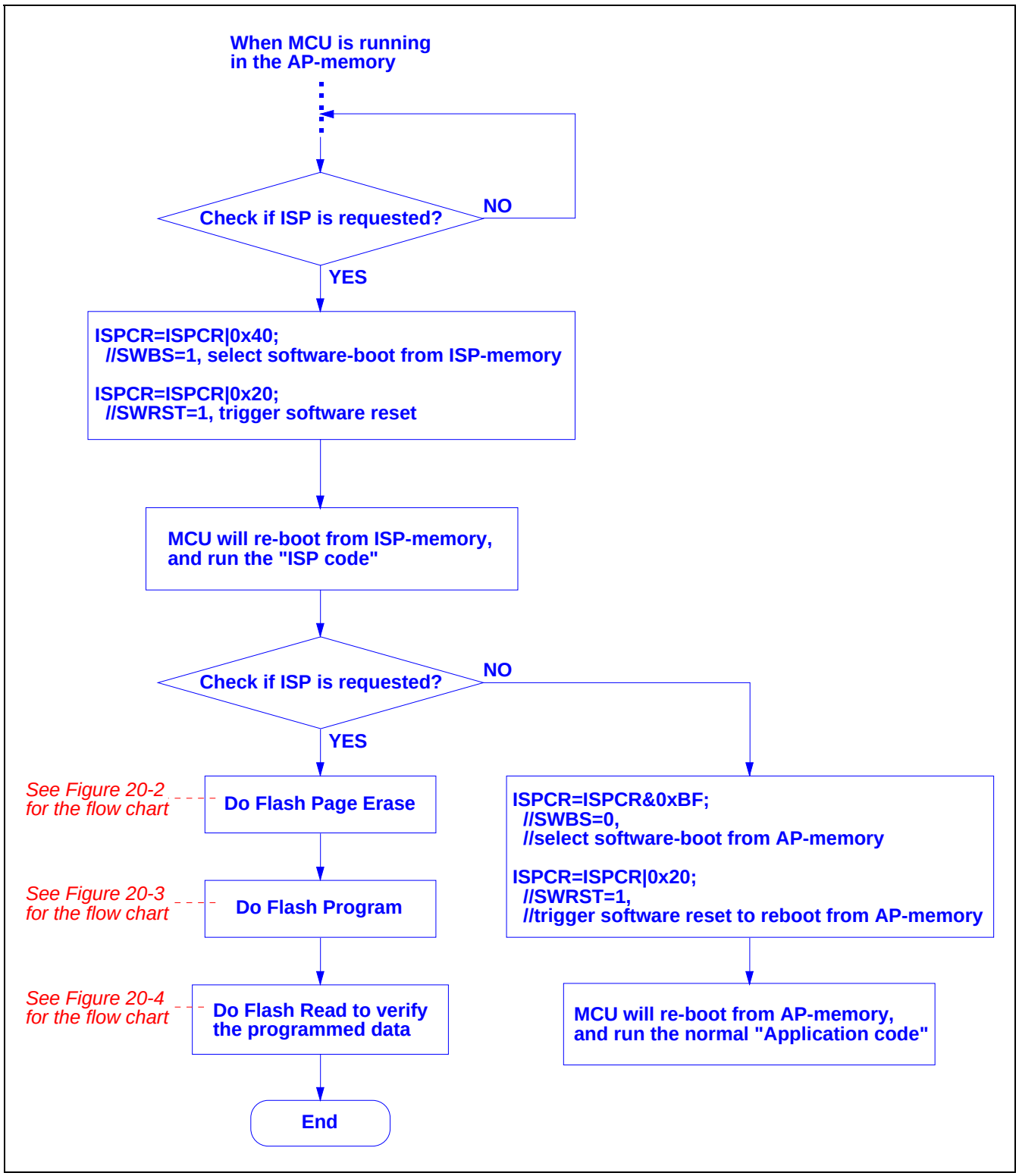
Figure 20-5. Directly boot from ISP-memory (HWBS or HWBS2 is enabled)



Method 2: MCU Re-boots from ISP-memory through AP-memory

The alternative method to make the MCU boot from the ISP-memory is to trigger a software reset while the MCU is running in the AP-memory. In this case, neither HWBS nor HWBS2 is enabled. The only way for the MCU to boot from the ISP-memory is to trigger a software reset when running in the AP-memory. See the following flow chart.

Figure 20-6. Re-boot from ISP-memory through AP-memory



20.2.4 Notes for ISP

Developing of the ISP Code

Although the ISP code is programmed in the ISP-memory that has an *ISP Start Address* in the MCU's Flash (see Figure 20-1), it doesn't mean you need to put this offset (= *ISP Start Address*) in your source code. The code offset is automatically manipulated by the hardware. You just needs to develop it like you develop your application program in the AP-memory.

Interrupts during ISP

After triggering the ISP processing, the MCU will halt for a while for internal ISP processing until the processing is completed. At this time, the interrupt will queue up for being serviced if the interrupt is enabled previously. Once the processing is completed, the MPU continues running and the interrupts in the queue will be serviced immediately if the interrupt flag is still active. The user, however, should be aware of the following:

(1) Any interrupt can not be in-time serviced when the MCU halts for ISP processing.

(2) The low-level triggered external interrupts, /INTx, should keep activated until the ISP is completed, or they will be neglected.

Accessing Destination of ISP

As mentioned previously, the ISP is used to program both the AP-memory and the IAP-memory. Once the accessing destination address is beyond that of the last byte of the IAP-memory, the hardware will automatically neglect the triggering of ISP processing. That is the triggering of ISP is invalid and the hardware does nothing.

Flash Endurance for ISP

The endurance of the embedded Flash is 20,000 erase/write cycles, that is to say, the erase-then-write cycles shouldn't exceed 20,000 times. Thus the user should pay attention to it in the application which needs to frequently update the AP-memory and IAP-memory.

Flash Write Protection during Low Power

To ensure a successful programming using ISP, the power coming from LDO output and supplied to the Flash memory should be higher than 2.4V (see Figure 23-1). The user may enable the hardware option LVFWP for write protection during the LDO output power falls below 2.4V during ISP processing. Refer to Section 25: MCU's Hardware Option.

20.2.5 ISP Tools Provided by Megawin

Although the user may design his own ISP, Megawin provides two types of ISP tool for the user; one is the *ISP Programmer*, which connects the target MCU to the PC through the USB port; and the other through the COM port, which needs no additional hardware except an RS232 transceiver. This section shows a brief description for these tools. The user may contact Megawin for further detailed information.

20.2.5.1 The "Megawin 8051 ISP Programmer"

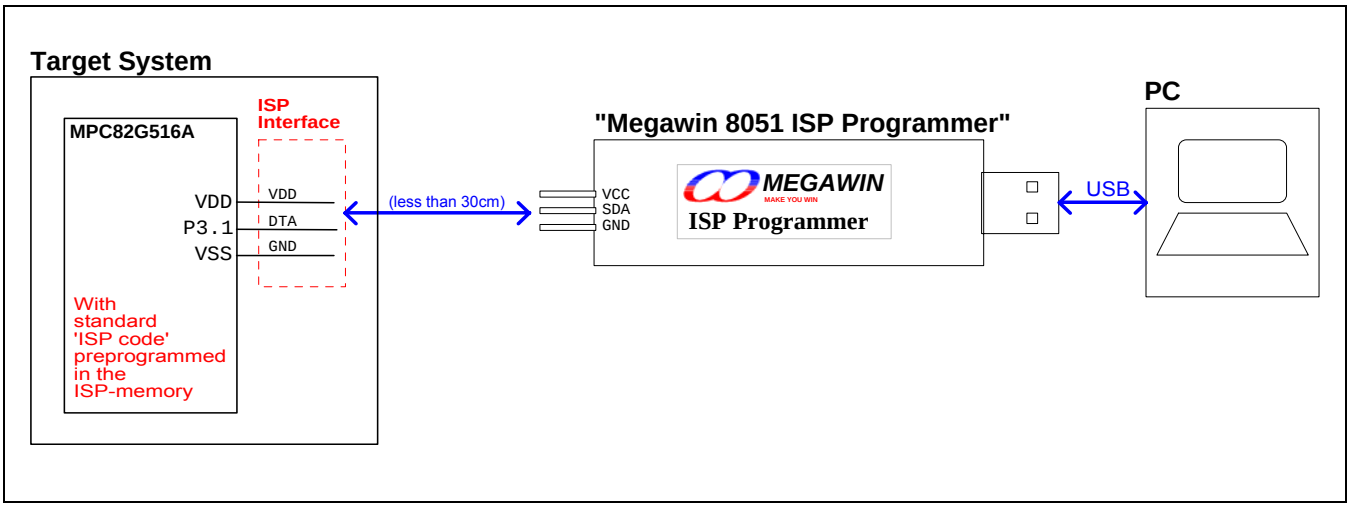
Features

- The standard 'ISP code' is pre-programmed in factory before shipping.
- Only one port pin (P3.1) used for the ISP interface.
- Operation independent of the oscillator frequency.
- Capable of stand-alone working without host's intervention.

The above valuable features make the ISP Programmer very friendly to the user. Particularly, it is capable of stand-alone working after the programming data is downloaded. This is especially useful in the field without a PC. The picture and system diagram of the "Megawin 8051 ISP Programmer" are shown below. Only three pins are used for the ISP interface: the ***DTA*** line transmits the programming data from the ISP Programmer to the target MCU; the ***VCC*** & ***GND*** are the power supply entry of the ISP Programmer. The USB connector can be directly plugged into the PC's USB port to download the programming data from PC to the ISP Programmer.

Figure 20-7. Picture of the "8051 ISP Programmer"

Figure 20-8. System Diagram for the ISP Function



20.2.5.2 ISP through COM Port

Features

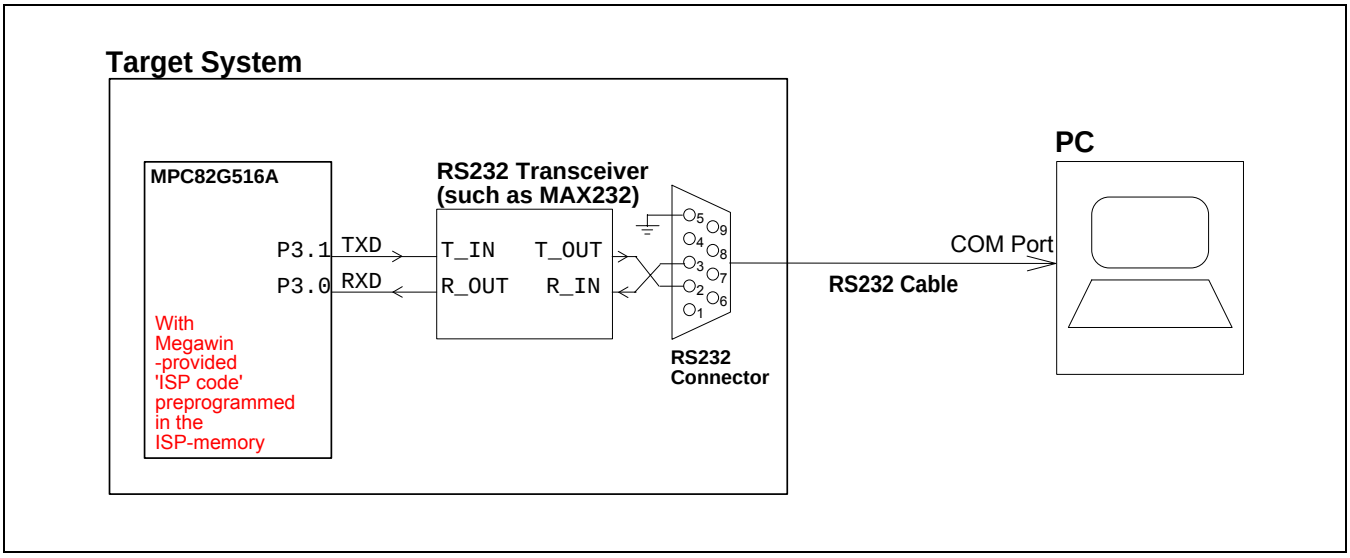
- The 'ISP code' for this mechanism is provided by Megawin.
- Auto baudrate detection & operation independent of the oscillator frequency.
- No additional hardware except an RS232 transceiver.

This ISP mechanism connects the target MCU to the PC through the COM port, which needs no additional hardware except an RS232 transceiver chip (such as the MAX232 chip).

The following system diagram shows the connection between the target MCU and the PC. Where, P3.0 and P3.1 function as the UART's RXD and TXD, respectively; and they are connected to the PC's COM port through the MAX232 chip.

It is not recommended to adopt this mechanism for ISP because the modern computer, especially a notebook computer, is not equipped with a COM port gradually.

Figure 20-9. System Diagram for ISP via COM Port



20.3 IAP Operation

The way to program the IAP-memory is called the 'In-Application Programming' (IAP), which is all the same as the ISP except the following differences:

(1) The IAP can only program the *IAP-memory* while the ISP can program both the *AP-memory* and *IAP-memory*.
(2) The program code to execute IAP is located in the *AP-memory* while the program code to execute ISP is located in the *ISP-memory*.

All the ISP modes (see Section 20.2.2) can also be applied to the IAP operation. Prior to using the IAP function, there must exist an IAP-memory. For the MPC82G516A, the user can configure an IAP-memory by a Universal Programmer, the "Megawin 8051 Writer" or the "Megawin 8051 ICP Programmer" (see Section 20.4.1).

20.3.1 Update the Data in the IAP-memory

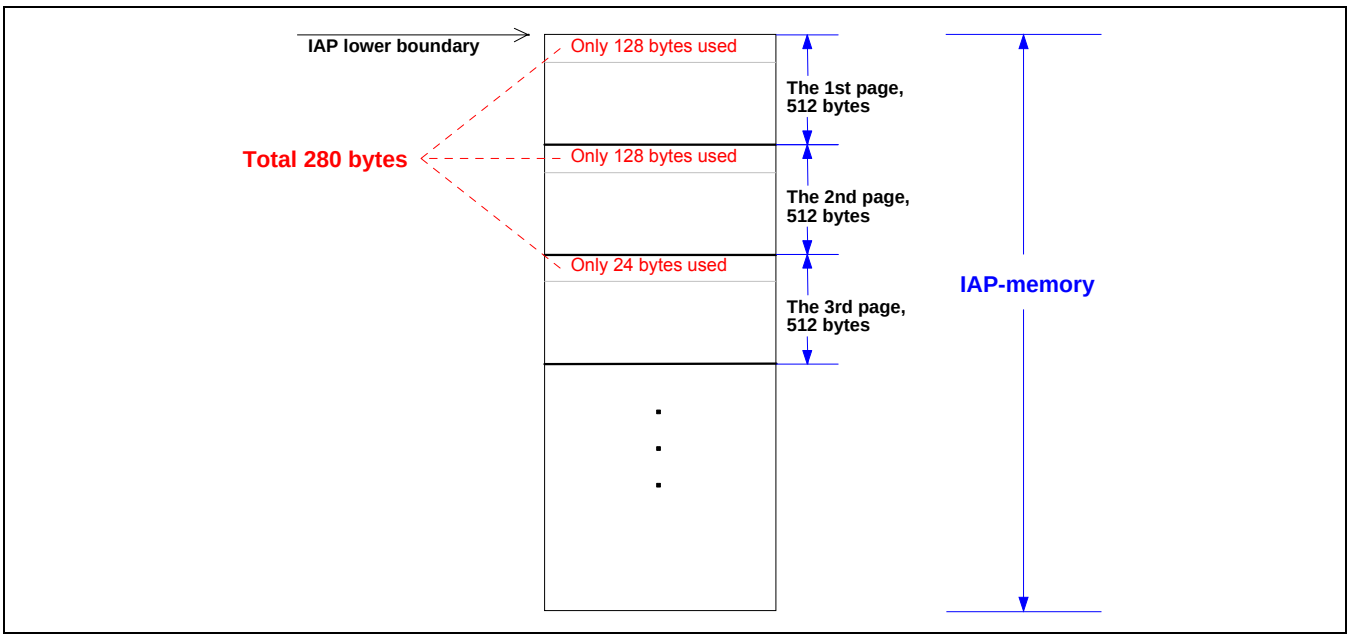
Because the Flash can only perform page erasing, and only the 0xFF byte can be programmed into a non-0xFF byte. So, if some bytes (in the same page) to be changed are not 0xFF, the user should comply with the following steps:

Step1) Read all the data in that page and save them in a page_buffer.
Step2) Erase that page.
Step3) Update the wanted bytes in the page_buffer.
Step4) Re-program that page with all the data out of the updated page_buffer.

Here the user might question: *Where is the **page_buffer**?* The page_buffer may exist in the traditional 256 bytes of scratchpad RAM or the on-chip eXpanded RAM (i.e., XRAM, accessed by the 'MOVX' instruction).

Normally, the page_buffer size should be equal to the Flash page size (i.e., 512 bytes). However, sometimes it is impossible to meet it due to the limited available space of RAM or XRAM. Here is an example that shows how to solve it. Suppose 280 bytes of non-volatile storage are wanted, but only 128 bytes of RAM are available for the page_buffer. In this case, we may use 3 pages to offer the 280 bytes of non-volatile storage: the 1st page offers 128 bytes, the 2nd page offers 128 bytes and the 3rd page offers 24 bytes, as shown in the Figure 20-10. Note that we have no choice but to take this usage!

Figure 20-10. Usage of IAP-memory when the page buffer is less than 512 bytes



20.3.2 Demo Code for IAP

As mentioned above, all the ISP modes can also be applied to the IAP operation. The demo codes for these modes are shown below.

Demo code for triggering the “Page Erase Mode”

```
MOV    ISPCR,#10000011b ;ISPCR.7=1, enable ISP
                        ;ISPCR[2:0]=011, suppose MPC82-series running @11.0592MHz

MOV    IFMT,#03h        ;select Page Erase Mode

MOV    IFADRH,??        ;fill [IFADRH,IFADRL] with page address
MOV    IFADRL,??        ;! the page address must be within the IAP-memory

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

;Now, MCU will halt here until processing completed
```

Demo code for triggering the “Program Mode”

```
MOV    ISPCR,#10000011b ;ISPCR.7=1, enable ISP
                        ;ISPCR[2:0]=011, suppose MPC82-series running @11.0592MHz

MOV    IFMT,#02h        ;select Program Mode

MOV    IFADRH,??        ;fill [IFADRH,IFADRL] with byte address
MOV    IFADRL,??        ;! the byte address must be within the IAP-memory
MOV    IFD,??           ;fill IFD with the data to be programmed

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

;Now, MCU will halt here until processing completed
```

Demo code for triggering the “Read Mode”

```
MOV    ISPCR,#10000011b ;ISPCR.7=1, enable ISP
                        ;ISPCR[2:0]=011, suppose MPC82-series running @11.0592MHz

MOV    IFMT,#01h        ;select Read Mode

MOV    IFADRH,??        ;fill [IFADRH,IFADRL] with byte address
MOV    IFADRL,??        ;! the byte address must be within the IAP-memory

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

;Now, MCU will halt here until processing completed

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

20.3.3 Notes for IAP

Interrupts during IAP

After triggering the ISP processing for In-Application Programming, the MCU will halt for a while for internal ISP processing until the processing is completed. At this time, the interrupt will queue up for being serviced if the interrupt is enabled previously. Once the processing is completed, the MCU 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 in-time serviced during the MCU halts for ISP processing.
(2) The low-level triggered external interrupts, /INTx, should keep activated until the ISP is completed, or they will be neglected.

Accessing Destination of IAP

As mentioned previously, the IAP is used to program only the IAP-memory. Once the accessing destination is not within the IAP-memory, the hardware will automatically neglect the triggering of ISP processing. That is the triggering of ISP is invalid and the hardware does nothing.

An Alternative Method to Read IAP Data

To read the Flash data in the IAP-memory, in addition to using the Flash Read Mode, the alternative method is using the instruction “MOVC A,@A+DPTR”. Where, DPTR and ACC are filled with the wanted address and the offset, respectively. And, the accessing destination must be within the IAP-memory, or the read data will be indeterminate. Note that using ‘MOVC’ instruction is much faster than using the Flash Read Mode.

Flash Endurance for IAP

The endurance of the embedded Flash is 20,000 erase/write cycles, that is to say, the erase-then-write cycles shouldn’t exceed 20,000 times. Thus the user should pay attention to it in the application which needs to frequently update the IAP-memory.

Flash Write Protection during Low Power

To ensure a successful programming using IAP, the power coming from LDO output and supplied to the Flash memory should be higher than 2.4V (see Figure 23-1). The user may enable the hardware option LVFWP for write protection during the LDO output power falls below 2.4V during ISP processing. Refer to Section 25: MCU’s Hardware Option.

20.4 About ICP

The ICP, like the traditional parallel programming method, can be used to program anywhere in the MCU, including the Flash and MCU's Hardware Option. And, owing to its dedicated serial programming interface (via the On-Chip Debug path), the ICP can update the MCU without removing the MCU chip from the actual end product, just like the ISP does.

20.4.1 The "Megawin 8051 ICP Programmer"

Only the proprietary "Megawin 8051 ICP Programmer" can support the In-Circuit Programming of MPC82G516A. This section gives a rough description for it.

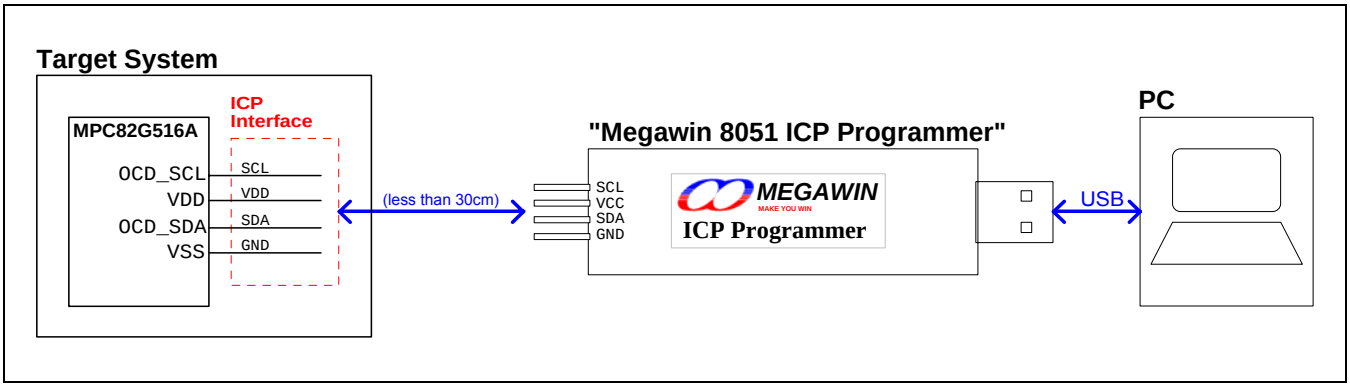
Features

- No need to have a loader program pre-programmed in the target MCU.
- Dedicated serial interface; no port pin is occupied.
- The target MCU needn't be in running state; it just needs to be powered.
- Capable of stand-alone working without host's intervention.

The above valuable features make the ICP Programmer very friendly to the user. Particularly, it is capable of stand-alone working after the programming data is downloaded. This is especially useful in the field without a PC. The picture and system diagram of the "Megawin 8051 ICP Programmer" are shown below. Only four pins are used for the ICP interface: the ***SDA*** line and ***SCL*** line function as serial data and serial clock, respectively, to transmit the programming data from the ICP Programmer to the target MCU; the ***VCC*** & ***GND*** are the power supply entry of the ICP Programmer. The USB connector can be directly plugged into the PC's USB port to download the programming data from PC to the ICP Programmer.

Figure 20-11. Picture of the "8051 ICP Programmer"

Figure 20-12. System Diagram for the ICP Function



21 Power Saving Modes

The MPC82G516A has two power-saving modes and an 8-bit system clock prescaler to reduce the power consumption. In the Idle mode the CPU is frozen while the peripherals and the interrupt system are still operating. In the Power-down mode the RAM and SFRs' value are saved and all other functions are inoperative; most importantly, in the Power-down mode the device can be waked up by the external interrupts. And, the user can further reduce the power consumption by using the 8-bit system clock prescaler to slow down the operating speed.

Registers PCON and PCON2 are related to power saving, as listed below.

PCON (Address=87H, Power Control Register, Reset Value=00xx,0000B (or 00x1,0000B after Power-On Reset))

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

GF1: General-purpose flag bit 1.

GF0: General-purpose flag bit 0.

PD: Power-Down mode bit. Setting this bit activates Power-Down mode.

IDL: Idle mode bit. Setting this bit activate Idle mode.

PCON2 (Address=C7H, Power Control Register 2, Reset Value=00x0,0000B)

7	6	5	4	3	2	1	0
-	-	-	-	-	SCKD2	SCKD1	SCKD0

SCKD2~SCKD0: System clock divider control bits.

SCKD2	SCKD1	SCKD0	Fosc (System Clock)
0	0	0	OSC_Freq
0	0	1	OSC_Freq /2
0	1	0	OSC_Freq /4
0	1	1	OSC_Freq /8
1	0	0	OSC_Freq /16
1	0	1	OSC_Freq /32
1	1	0	OSC_Freq /64
1	1	1	OSC_Freq /128

(Refer to Section 22: System Clock.)

21.1 Idle Mode

An instruction that sets IDL bit (PCON.0) causes that to be the last instruction executed before going into the Idle mode, the internal clock signal is gated off to the CPU but not to the peripherals that need to keep working in the mode, such as Interrupt, Timer, Serial Port functions and so forth. The CPU contents, the on-chip RAM, and all of the Special Function Registers remain intact during Idle. The port pins hold the logical states they had at the time Idle was activated.

There are two ways to terminate the Idle. Activation of the enabled interrupts such as External Interrupt, Timer, Serial Port and Keypad Interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI, the next instruction to be executed will be the one following the instruction that put the device into Idle. The other way of terminating the Idle mode is with a hardware reset from the RST pin. Since the clock oscillator is still running, the hardware reset needs to be held active for 24 clock cycles to complete the reset.

The flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during an Idle. For example, an instruction that activates Idle can also set one or both flag bits. When Idle is terminated by an interrupt, the interrupt service routine can examine the flag bits to differentiate normal operation and Idle.

21.2 Power-Down Mode

To save even more power, the Power-Down mode can be invoked by software .An instruction that sets PD bit (PCON.1) causes that to be the last instruction executed before going into the Power-Down mode. In the Power-Down mode, the on-chip oscillating is stopped. With the clock frozen, all functions are stopped, the contents of the on-chip RAM and all of the Special Function Registers retain their values. The port pins output the values held by their respective SFRs.

Either a hardware reset from the RST pin or the External Interrupt (INT0~INT3) & Keypad Interrupt can be used to exit from Power-Down. Reset initializes all the SFRs but does not change the on-chip RAM. The External Interrupt & Keypad Interrupt allow both the SFRs and the on-chip RAM to retain their values; and, once the interrupt is serviced, the next instruction to be executed after RETI will be the one following the instruction that put the device into Power-Down.

21.2.1 Wake-up from Power-Down Mode

To exit from Power-Down mode by External Interrupts or Keypad Interrupt, it is recommended to insert at least one NOP instruction following the instruction that invokes Power-Down mode. The NOP instruction is used to eliminate the possibility of unexpected code execution when returning from the interrupt service routine.

Note: /INT0 is used in this example.

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

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

          SETB   INT0       ;pull high P3.2

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

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

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



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

21.3 Slow-Down Operation

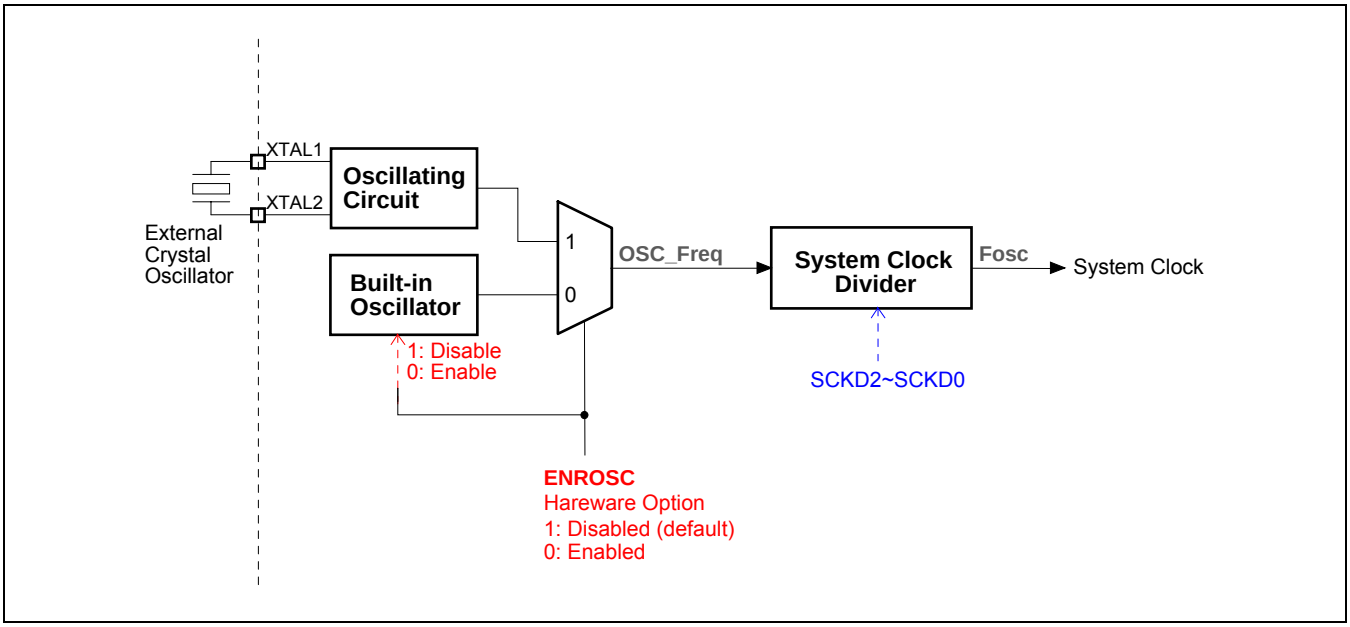
The alternative to save the operating power is to slow the MCU's operating speed by programming SCKD2~SCKD0 bits (in PCON2 register, see Section 22) to a non-0/0/0 value. The user should examine which program segments are suitable for lower operating speed. In principle, the lower operating speed should not affect the system's normal function. Then, restore its normal speed in the other program segments.

22 System Clock

There are two clock sources for the system clock: external crystal oscillator and the built-in oscillator. The system clock, Fosc, is obtained from one of these two clock sources through the clock divider, as shown in Figure 22-1. The user can program the divider control bits SCKD2~SCKD0 (in PCON2 register) to get the desired system clock.

The built-in oscillator is enabled by the hardware option ENROSC. Refer to Section 25: MCU's Hardware Option.

Figure 22-1. Block Diagram of System Clock



PCON2 (Address=C7H, Power Control Register 2, Reset Value=00x0,0000B)

7	6	5	4	3	2	1	0
-	-	-	-	-	SCKD2	SCKD1	SCKD0

SCKD2~SCKD0: System clock divider control bits.

SCKD2	SCKD1	SCKD0	Fosc (System Clock)
0	0	0	OSC_Freq
0	0	1	OSC_Freq /2
0	1	0	OSC_Freq /4
0	1	1	OSC_Freq /8
1	0	0	OSC_Freq /16
1	0	1	OSC_Freq /32
1	1	0	OSC_Freq /64
1	1	1	OSC_Freq /128

22.1 Built-in Oscillator

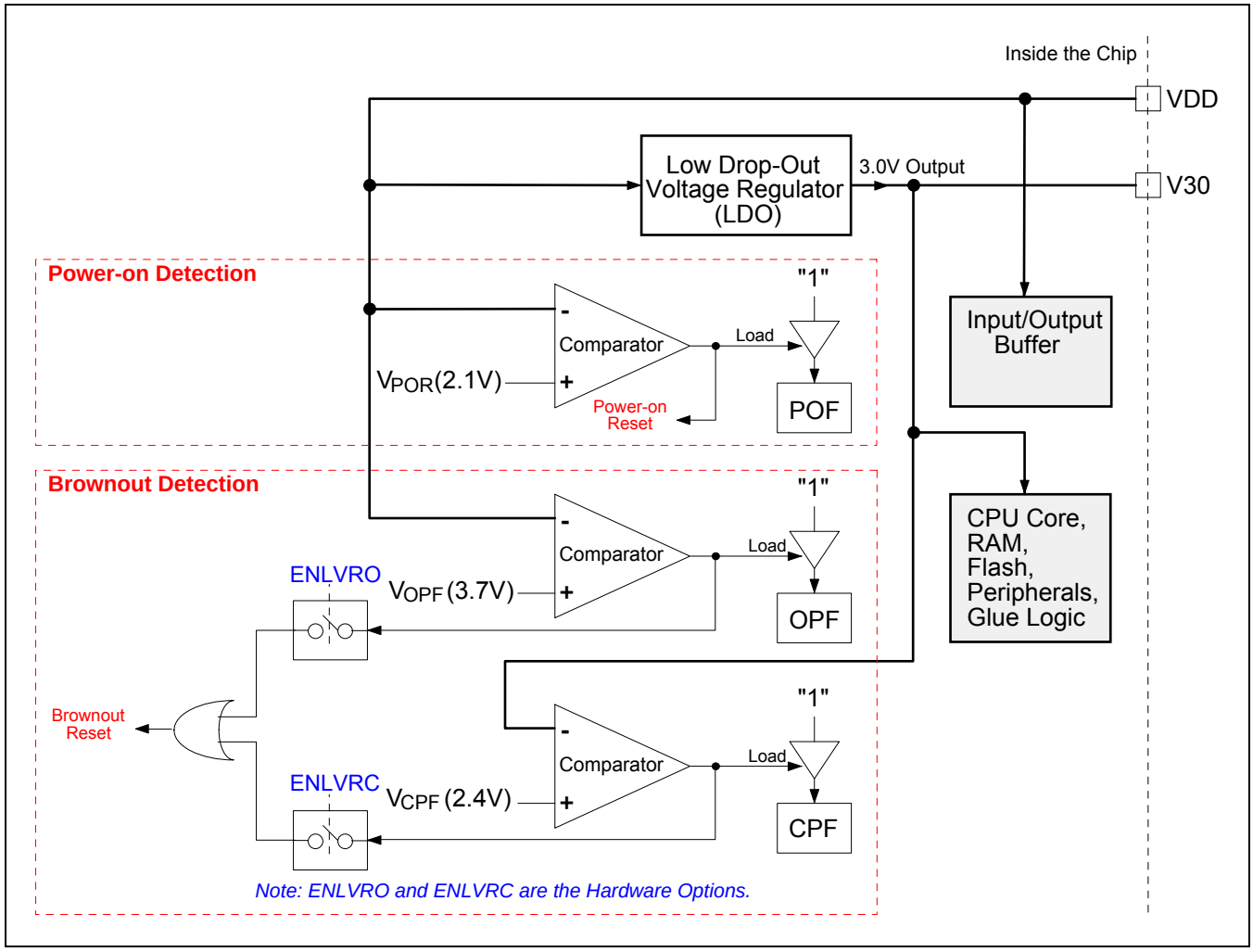
The MPC82G516A has a built-in oscillator with the rough oscillating frequency of 6MHz. It can be used to replace the external crystal oscillator in the application which doesn't need an exact oscillating frequency. To enable the built-in oscillator, the user should enable the hardware option ENROSC by 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 does not require an exact oscillating frequency.*

23 Power Monitoring Function

The MPC82G516A incorporates power monitoring functions designed to prevent incorrect operation during initial power-up and power loss or reduction during operation. This is accomplished with two hardware functions: Power-on Detection and Brownout Detection. Figure 23-1 shows the block diagram of power monitoring function.

Figure 23-1. Power Monitor Block Diagram



23.1 Power-on Detection

The POF flag (PCON.4) is set by hardware to indicate an initial power-up condition. It remains set until cleared by software, and therefore helps the user to check if the start-running of the MCU comes from *cold start* (i.e., power up) or *warm start* (such as hardware reset from RST pin, software reset or WDT reset). In addition to an initial power-up condition, the POF flag will also be set by hardware whenever the VDD power falls below V_{POR}.

PCON (Address=87H, Power Control Register, Reset Value=00xx,0000B (or 00x1,0000B after Power-On Reset))

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

POF: Power-ON Flag. This bit is set during power-on reset. It can also be set by software. However, it can only be cleared by software.

23.2 Brownout Detection

The brownout detection function determines if the power supply voltage falls below a certain level. The default operation for a brownout detection is to cause the power-fail flag to be set, which can then generate an interrupt if the Brownout interrupt (#12 in Table 19-1) is enabled. However, it may alternatively be configured to cause a hardware reset by enabling the related hardware option.

There are two kinds of brownout detection:

(1) VDD power-fail detection: When power supplied to the VDD pin falls below V_{OPF} (3.7V), the OPF flag will be set by hardware to indicate a VDD power-fail condition. The detection is used in a 5V or wide-range system.

(2) LDO power-fail detection: When the LDO output power falls below V_{CPF} (2.4V), the CPF flag will be set by hardware to indicate an LDO power-fail condition. The detection is used in a 3.3V system.

Note: See Section 27.1 for 3.3V, 5V or wide-range system.

Note that during power-up, the power-fail flags OPF and CPF are set by hardware owing to the power ever lower than 3.7V and 2.4V, respectively. The user should clear them by software before a normal brownout detection. The flags OPF and CPF may trigger a brownout interrupt if EA is set and EOPFI or EOPCI is set (Refer to Section 19: Interrupt System).

To trigger an internal reset when brownout occurs, the hardware option ENLVRO or ENLVRC should be enabled. (Refer to Section 25: MCU's Hardware Option).

The EVRCR register contains the flags and control bits for brownout detection.

EVRCR (Address=97H, EVR Control Register, Reset Value=00xx,0000B (or 0011,0000B after Power-On Reset))

7	6	5	4	3	2	1	0
EOPFI	ECPFI	OPF	CPF	PMUOFF	(Reserved)	(Reserved)	(Reserved)

EOPFI: Set/clear to enable/disable the interrupt when OPF=1.

ECPFI: Set/clear to enable/disable the interrupt when CPF=1.

OPF: VDD Power-Fail flag. This bit is set by hardware when the power supplied to the VDD pin falls below 3.7V. It can only be cleared by software.

CPF: LDO Power-Fail flag. This bit is set by hardware when the LDO output power falls below 2.4V. It can only be cleared by software.

PMUOFF: Set to turn off the Power Monitor Unit to save power consumption while power monitoring function is not used.

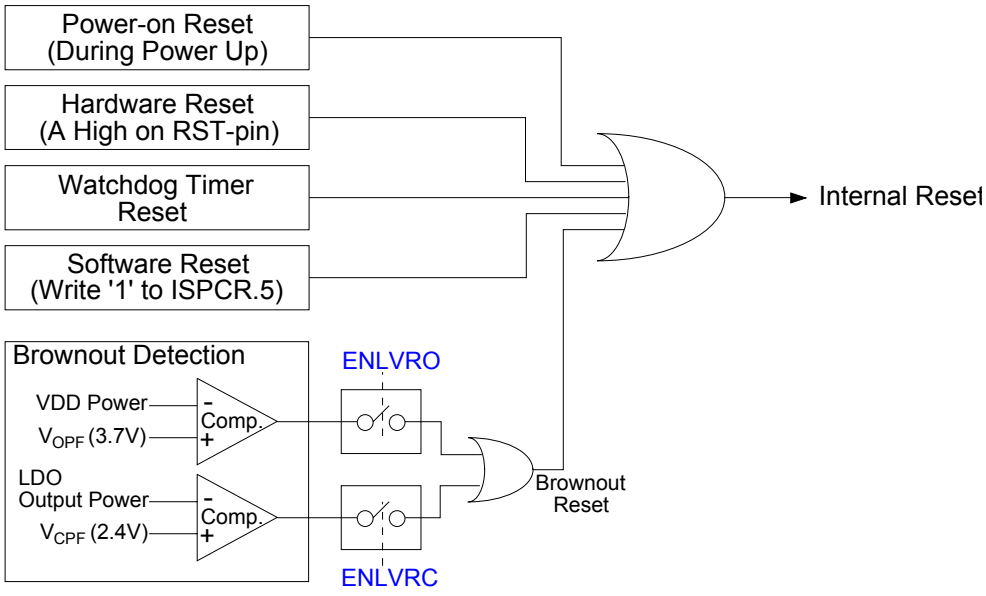
Note all the reserved bits in this register must be kept '0'.

24 Reset Sources

Reset can be triggered from the following reset sources (see Figure 24-1):

- Power-on reset
- Hardware reset from RST-pin
- Watchdog Timer reset
- Software reset
- Brownout reset from Power Monitor

Figure 24-1. Block Diagram of Reset



Note: ENLVRO and ENLVRC are the Hardware Options.

24.1 Power-On Reset

Power-on reset (POR) is used to internally reset the MCU during power-up. The MCU will keep in reset state and will not start to work until the VDD power rises above V_{POR} (the POR threshold voltage). And, the reset state is activated again whenever the VDD power falls below V_{POR}. During a power cycle, VDD must fall below V_{POR} before power is reapplied in order to ensure a power-on reset. See Section 30: DC Characteristics for V_{POR}.

24.2 Hardware Reset from RST-Pin

A reset is accomplished by holding the RST pin HIGH for at least 24 oscillator periods while the oscillator is running. To ensure a reliable power-up reset, the hardware reset from RST pin is necessary.

24.3 Watchdog Timer Reset

When the Watchdog Timer is enabled, it will increment every 12 system clock cycles (12/Fosc) while the oscillator is running. And, the user needs to service it to avoid an overflow, which will generate an internal reset signal.

24.4 Software Reset

Writing '1' to bit SWRST will trigger a software reset, which causes the MCU to re-boot from the AP-memory or the ISP-memory according to the SWBS bit. See the ISPCR register shown below.

ISPCR (Address=E7H, ISP Control Register, Reset Value=000x,x000B)

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

SWBS: Software boot select. Set/clear to select booting from ISP-memory/AP-memory for software reset.

SWRST: Write '1' to this bit to trigger a software reset.

24.5 Brownout Reset from Power Monitor

To trigger an internal reset when brownout occurs, the hardware option ENLVRO or ENLVRC should be enabled. (Refer to Section 25: MCU's Hardware Option).

25 MCU's Hardware Option

The MCU's Hardware Option defines the device behavior which cannot be programmed or controlled by software. The hardware options can only be programmed by a Universal Programmer, the "Megawin 8051 Writer" or the "Megawin 8051 ICP Programmer". After whole-chip erased, all the hardware options are left in "disabled" state and there is no ISP-memory and IAP-memory configured. The MPC82G516A has the following Hardware Options:

ISP-memory Space:
The ISP-memory space is specified by its starting address. And, its higher boundary is limited by the Flash end address, i.e., 0xFFFF. (See Section 20.1.2: Flash Configuration.)

IAP-memory Space:
The IAP-memory space is specified by its lower boundary. And, its higher boundary is limited by the starting address of the ISP-memory space if the ISP-memory is configured; otherwise, its higher boundary is located at address 0xFFFF. (See Section 20.1.2: Flash Configuration.)

LVFWP:
[enabled]: Flash write protection is enabled during IAP/ISP processing when LDO output power falls below V_{CPF} (i.e., 2.4V).
[disabled]: No Flash write protection.

ENLVRC:
[enabled]: Enable brownout reset when LDO output power falls below V_{CPF} (2.4V).
[disabled]: No brownout reset when LDO output power falls below V_{CPF} (2.4V).

HWBS:
[enabled]: When powered 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]: Oscillating gain is normal.

HWBS2:
[enabled]: Not only power-up but also any reset will cause MCU to boot from ISP-memory if ISP-memory is configured.
[disabled]: Where MCU boots from is determined by **HWBS**.

ENLVRO:
[enabled]: Enable brownout reset when VDD power falls below V_{OPF} (3.7V).
[disabled]: No brownout reset when VDD power falls below V_{OPF} (3.7V).

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

WDSFWP:
[enabled]: The special function register WDTCR will be software-write-protected except the bit CLRW.
[disabled]: The special function register WDTCR is free to be written by software.

HWENW (accompanied with arguments **HWWIDL** and **HWPS[2:0]**):

[enabled]: Automatically enable Watch-dog Timer by the hardware when the MCU is powered up.

It means that:

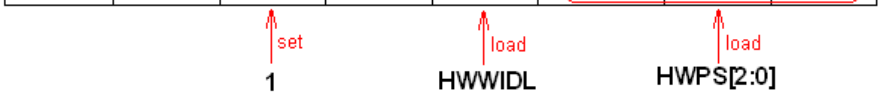
In the WDTCR register, the hardware will automatically:
(1) set *ENW* bit,
(2) load **HWWIDL** into *WIDL* bit, and
(3) load **HWPS[2:0]** into *PS[2:0]* bits.

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 the MCU is powered up.

26 Instruction Set

The 80C51 instruction set is optimized for 8-bit control applications. It provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. The instruction set provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that require Boolean processing.

The MPC82G516A instruction set is fully compatible with those of the 80C51 except the execution time, i.e., the number of clock cycles required to execute an instruction. The shortest execution time is just one clock cycle and the longest is 7 clock cycles.

Addressing Modes

The addressing modes in the 80C51 instruction set are as follows:

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

Indirect Addressing
In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed. The address register for 8-bit addresses can be R0 or R1 of the selected bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit “data pointer” register, DPTR.

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

Register-Specific Instructions
Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumulator as A assemble as accumulator specific opcodes.

Immediate Constants
The value of a constant can follow the opcode in Program Memory. For example, “MOV A, #100” loads the Accumulator with the decimal number 100. The same number could be specified in hex digits as 64H.

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

Prior to introducing the instruction set, the user 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.

26.1 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

26.2 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

26.3 Data Transfer

Mnemonic		Description	Byte	Execution Clock Cycles
DATA TRANSFER				
MOV	**A,Rn**	Move register to ACC	1	1
MOV	**A,direct**	Move direct byte o ACC	2	2
MOV	**A,@Ri**	Move indirect RAM to ACC	1	2
MOV	**A,#data**	Move immediate data to ACC	2	2
MOV	**Rn,A**	Move ACC to register	1	2
MOV	**Rn,direct**	Move direct byte to register	2	4
MOV	**Rn,#data**	Move immediate data to register	2	2
MOV	**direct,A**	Move ACC to direct byte	2	3
MOV	**direct,Rn**	Move register to direct byte	2	3
MOV	**direct,direct**	Move direct byte to direct byte	3	4
MOV	**direct,@Ri**	Move indirect RAM to direct byte	2	4
MOV	**direct,#data**	Move immediate data to direct byte	3	3
MOV	**@Ri,A**	Move ACC to indirect RAM	1	3
MOV	**@Ri,direct**	Move direct byte to indirect RAM	2	3
MOV	**@Ri,#data**	Move immediate data to indirect RAM	2	3
MOV	**DPTR,#data16**	Load DPTR with a 16-bit constant	3	3
MOVC	**A,@A+DPTR**	Move code byte relative to DPTR to ACC	1	4
MOVC	**A,@A+PC**	Move code byte relative to PC to ACC	1	4
MOVX	**A,@Ri** *Note1*	Move on-chip XRAM (8-bit address) to ACC	1	3
MOVX	**A,@DPTR** *Note1*	Move on-chip XRAM (16-bit address) to ACC	1	3
MOVX	**@Ri,A** *Note1*	Move ACC to on-chip XRAM (8-bit address)	1	4
MOVX	**@DPTR,A** *Note1*	Move ACC to on-chip XRAM (16-bit address)	1	3
MOVX	**A,@Ri** *Note2*	Move external data memory (8-bit address) to ACC	1	7 *Note3*
MOVX	**A,@DPTR** *Note2*	Move external data memory (16-bit address) to ACC	1	7 *Note3*
MOVX	**@Ri,A** *Note2*	Move ACC to external data memory (8-bit address)	1	7 *Note3*
MOVX	**@DPTR,A** *Note2*	Move ACC to external data memory (16-bit address)	1	7 *Note3*
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

Note1:
For the control bit EXTRAM=0, all "MOVX" instructions are directed to the on-chip expanded XRAM.

Note2:
For the control bit EXTRAM=1, all "MOVX" instructions are directed to the external data memory.

Note3:
The cycle time for access of external data memory is:

7 + 2 x (ALE_Stretched_Clocks) + (RW_Stretched_Clocks)

26.4 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

26.5 Program and Machine Control

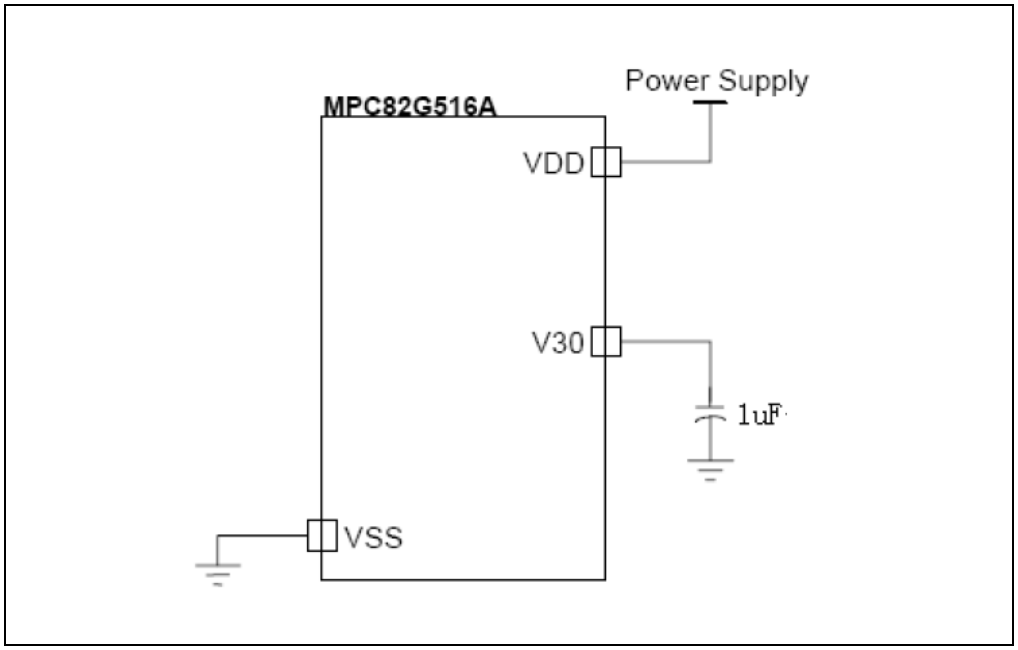
Mnemonic	Description	Byte	Execution Clock Cycles
PROAGRAM AND MACHINE CONTROL			
ACALL addr11	Absolute subroutine call	2	6
LCALL addr16	Long subroutine call	3	6
RET	Return from subroutine	1	4
RETI	Return from interrupt subroutine	1	4
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short 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

27 Application Notes

27.1 MPC82G516A V30 pin application note

The pin V30 has to connect with 1uF capacitance in wide power supply range (**2.7 V~ 5.5 V**). Shown as below diagram.

Figure 27-1. V30 pin connected



MPC82G516A Pin V30

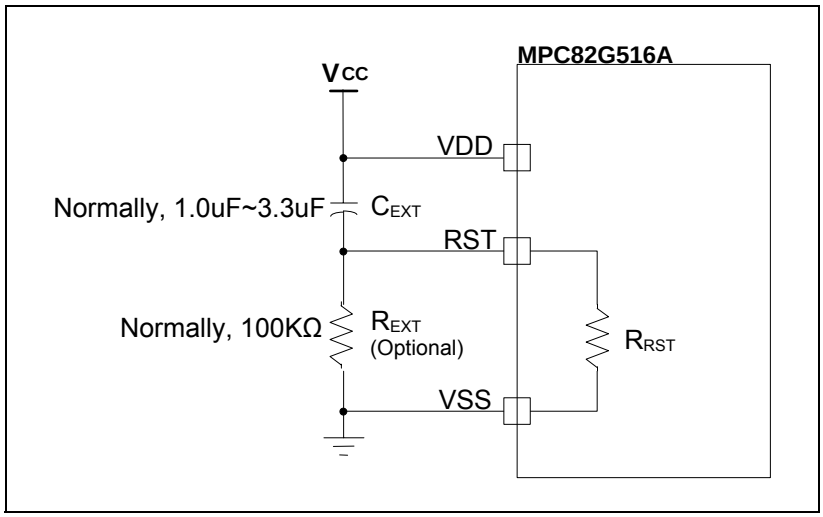
27.2 Reset Circuit

Normally, the power-on reset can be successfully generated during power-up. However, to further ensure the MCU a reliable reset during power-up, the external reset is necessary. Figure 27-4 shows the external reset circuit, which consists of a capacitor C_{EXT} connected to VDD (power supply) and a resistor R_{EXT} connected to VSS (ground).

In general, R_{EXT} is optional because the RST pin has an internal pull-down resistor (R_{RST}). This internal diffused resistor to VSS permits a power-up reset using only an external capacitor C_{EXT} to VDD.

See Section 30: DC Characteristics for R_{RST}.

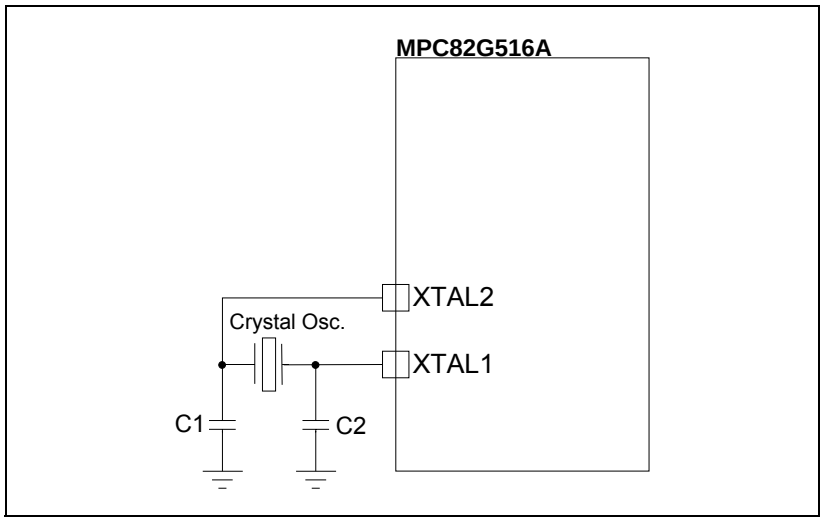
Figure 27-4. Reset Circuit



27.3 XTAL Oscillating Circuit

To achieve successful and exact oscillating (up to 24MHz), the capacitors C1 and C2 are necessary regardless of state of the hardware option OSCDN (enabled or disabled). Normally, C1 and C2 have the same value of about 20pF~150pF.

Figure 27-5. XTAL Oscillating Circuit



28 On-Chip Debug Function

The MPC82G516A is equipped with a proprietary On-Chip Debug (OCD) interface for In-Circuit Emulator (ICE). The OCD interface provides on-chip and in-system non-intrusive debugging without any target resource occupied. Several operations necessary for an ICE are supported, such as Reset, Run, Stop, Step, Run to Cursor and Breakpoint Setting.

Using the OCD technology, Megawin provides the "Megawin 8051 OCD ICE" for the user, as shown in Figure 28-1. The user has no need to prepare any development board during developing, or the socket adapter used in the traditional ICE probe. All the thing the user needs to do is to reserve a 4-pin connector on the system for the dedicated OCD interface: *VCC*, *OCD_SDA*, *OCD_SCL* and *GND*. Figure 28-2 shows the system diagram of the OCD ICE.

In addition, the most powerful feature is that it can directly connect the user's target system to the *Keil 8051 IDE software* for debugging, which directly utilizes the Keil IDE's *dScope*-Debugger function. Of course, all the advantages are based on your using *Keil 8051 IDE software*.

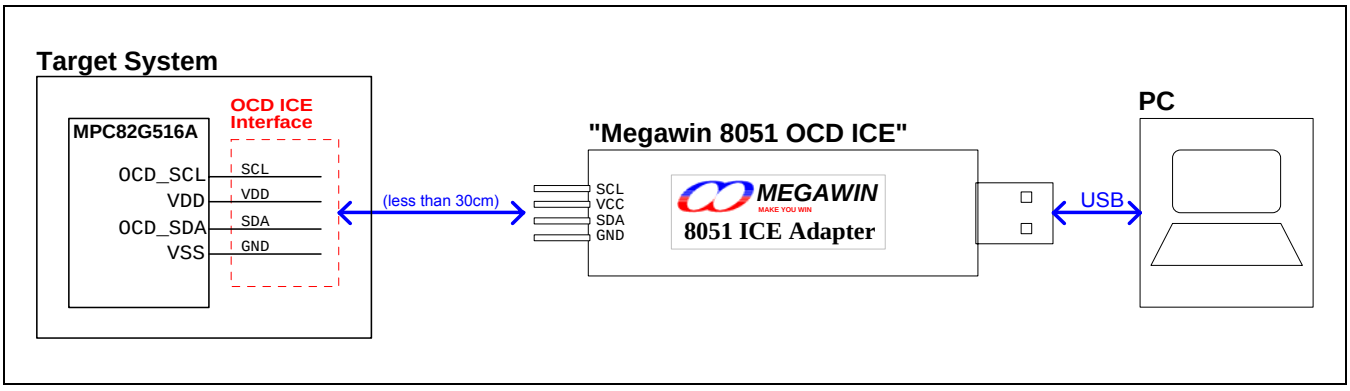
Note:
"Keil" is the trade mark of "Keil Elektronik GmbH and Keil Software, Inc.", and "Keil 8051 IDE software" is the most popular C51 compiler for 8051 embedded system development.

Features

- Megawin proprietary OCD (On-Chip-Debug) technology
- On-chip & in-system real-time debugging
- Two-pin dedicated serial interface for OCD, no target resource occupied
- Directly linked to the debugger function of the *Keil 8051 IDE Software*
- USB connection between target and host (PC)
- Helpful debug actions: *Reset*, *Run*, *Stop*, *Step* and *Run to Cursor*
- Programmable breakpoints, up to 4 breakpoints can be inserted simultaneously
- Several debug-helpful windows: Register/Disassembly/Watch/Memory Windows
- Source-level (*Assembly* or *C-language*) debugging capability

Figure 28-1. Picture of the "8051 ICE Adapter"

Figure 28-2. System Diagram for the ICE Function



Note: For more detailed information about the OCD ICE, please feel free to contact Megawin.

29 Absolute Maximum Ratings

Parameter	Rating	Unit
Operating temperature under bias *Note4	-40 ~ +85	℃
Storage temperature	-55 ~ +125	℃
Voltage on VDD to VSS	-0.5 ~ +6.5	V
Voltage on any other pin to VSS	-0.5 ~ VDD+0.5	V
Maximum I_{OL}/I_{OH} per output *Note5	20	mA
Maximum total I_{OL}/I_{OH} for all outputs *Note5	100	mA
Power dissipation *Note6	1.5	W

NOTES:

1. Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation or extended exposure beyond these ratings is not recommended and may affect device reliability.
2. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying greater than the rated maximum.
3. Parameters are valid over operating temperature range unless otherwise specified.
4. Tested by sampling.
5. Under steady state (non-transient) conditions, I_{OL}/I_{OH} must be externally limited.
6. Based on package heat transfer limitations, not device power consumption.

30 DC Characteristics

[Condition 1] 3.3V System

F_{OSC}=12MHz, T_{amb}=-40℃~+85℃, V_{DD}=2.4V~3.6V, unless otherwise specified

Symbol	Parameter	Test Conditions	Min	Typ*[1]	Max	Unit
I_{IL1}	Logic 0 input current, P0/P1/P2/P3/P4 (*Quasi-bidirectional*)	V_{DD}=3.6V and V_{IN}=0.4V	-	-	-10	μA
I_{IL2}	Logic 0 input current, P0/P1/P2/P3/P4 (*Input-only*)	V_{DD}=3.6V and V_{IN}=0.4V	-	0	-	μA
I_{TL}*[2]	Logic 1-to-0 transition current, P0/P1/P2/P3/P4 (*Quasi-bidirectional*)	V_{DD}=3.6V and V_{IN}=1.5V	-	-	-120	μA
V_{IH1}	Input high voltage, P0/P1/P2/P3/P4 (*Quasi-bidirectional or Input-only*)		0.3V_{DD} +0.5	-	V_{DD}+ 0.5	V
V_{IH2}	Input high voltage, RST		0.25V_{DD} +0.5	-	V_{DD}+ 0.5	V
V_{IH3}	Input high voltage, XTAL1		0.4V_{DD}	-	V_{DD}+ 0.5	V
V_{IL1}	Input low voltage, P0/P1/P2/P3/P4 (*Quasi-bidirectional or Input-only*)		-0.5	-	0.3V_{DD}	V
V_{IL2}	Input low voltage, RST		-0.5	-	0.3V_{DD}	V
V_{IL3}	Input low voltage, XTAL1		-0.5	-	0.35V_{DD}	V
V_{OH1}*[3]	Output high current, P0/P1/P2/P3/P4 (*Quasi-bidirectional*)	V_{DD}=2.4V and I_{OH}=-17μA V_{DD}=3.6V and I_{OH}=-70μA	2.0 2.4	-	-	V
V_{OH2}*[3]	Output high current, P0/P1/P2/P3/P4 (*Push-pull output*)	V_{DD}=2.4V and I_{OH}=-2.1mA V_{DD}=3.6V and I_{OH}=-8.5mA	2.0 2.4	-	-	V
V_{OH3}*[3]	Output high current, XTAL2	V_{DD}=2.4V and I_{OH}=-0.9mA V_{DD}=3.6V and I_{OH}=-3.2mA	2.0 2.4	-	-	V
V_{OL1}*[3]	Output low current, P0/P1/P2/P3/P4 (*Quasi-bidirectional*)	V_{DD}=2.4V and I_{OL}=+7.0mA V_{DD}=3.6V and I_{OL}=+10.2mA	-	-	0.4	V
V_{OL2}*[3]	Output low current, P0/P1/P2/P3/P4 (*Push-pull output*)	V_{DD}=2.4V and I_{OL}=+7.0mA V_{DD}=3.6V and I_{OL}=+10.2mA	-	-	0.4	V
V_{OL3}*[3]	Output low current, P0/P1/P2/P3/P4 (*Open-drain output*)	V_{DD}=2.4V and I_{OL}=+7.0mA V_{DD}=3.6V and I_{OL}=+10.2mA	-	-	0.4	V
V_{OL4}*[3]	Output low current, XTAL2	V_{DD}=2.4V and I_{OL}=+1.4mA V_{DD}=3.6V and I_{OL}=+1.6mA	-	-	0.4	V
R_{RST}	Internal reset pull-down resistor		180	-	320	KΩ
V_{CPF}	Brownout threshold, LDO output		-	2.4	-	V
V_{RAM}*[4]	RAM keep-alive voltage		1.2	-	-	V

(Continued)

Symbol	Parameter	Test Conditions	Min	Typ*[1]	Max	Unit
V_{DD}*[5,*8]	Power supply voltage	F_{osc}=6MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.1 1.8	 -	 3.6	V
		F_{osc}=12MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.1 1.8	 -	 3.6	
		F_{osc}=24MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.3 2.3	 -	 3.6	
V_{POR}	Power-on reset threshold voltage	T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	-	2.4 2.1 1.8	-	V
I_{DD}*[6]	Power supply current, operating*[7]	V_{DD}=2.4V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	 4.0 6.2 11.0	 5.0 7.5 13.5	mA
		V_{DD}=3.6V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	 9.1 13.0 20.8	 11.0 15.5 25.0	
	Power supply current, idle	V_{DD}=2.4V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	 1.7 2.6 4.5	 2.5 3.5 5.5	mA
		V_{DD}=3.6V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	-	 3.1 4.8 8.2	 4.0 6.0 10.0	
	Power supply current, power-down	F_{osc}=24MHz, V_{DD}=2.4V~3.6V	-	1	10	μA

Notes:

*1: Typical values are based on a limited number of samples and are not guaranteed. The values listed are at room temperature unless otherwise specified.

*2: Port pins source a transition current when used in quasi-bidirectional mode and externally driven from logic 1 to logic 0. This current is highest when V_{IN} is approximately 1.5V.

*3: See Section 29: Absolute Maximum Ratings for steady state (non-transient) limits on I_{OH} and I_{OL}.
If I_{OH} exceeds the test condition, V_{OH} will be lower than the listed specification.
If I_{OL} exceeds the test condition, V_{OL} will be higher than the listed specification.

*4: Supply voltage for RAM data retention.

*5: With the Hardware Option ENLVRO & ENLVRC disabled.

*6: With the Hardware Option OSCDN enabled.

*7: Tested while CPU running in a NOP loop, as shown below.

```
Loop:
    NOP
    JMP   Loop
```

*8: The listed minimum supply voltage is obtained under normal work of logic function, excluding Flash erase and write. The range 2.4V~3.6 is the supply voltage for normal work of whole chip function.

[Condition 2] 5V or Wide-Range System

F_{OSC}=12MHz, T_{amb}=-40℃~+85℃, V_{DD}=2.7V~5.5V, unless otherwise specified

Symbol	Parameter	Test Conditions	Min	Typ*[1]	Max	Unit
I_{IL1}	Logic 0 input current, P0/P1/P2/P3/P4 (*Quasi-bidirectional*)	V_{DD}=5.5V and V_{IN}=0.4V	-	-	-30	μA
I_{IL2}	Logic 0 input current, P0/P1/P2/P3/P4 (*Input-only*)	V_{DD}=5.5V and V_{IN}=0.4V	-	0	-	μA
I_{TL}*[2]	Logic 1-to-0 transition current, P0/P1/P2/P3/P4 (*Quasi-bidirectional*)	V_{DD}=5.5V and V_{IN}=2.0V	-	-	-260	μA
V_{IH1}	Input high voltage, P0/P1/P2/P3/P4 (*Quasi-bidirectional or Input-only*)		$0.25V_{DD}$ +0.7	-	V_{DD}+ 0.5	V
V_{IH2}	Input high voltage, RST		$0.25V_{DD}$ +0.5	-	V_{DD}+ 0.5	V
V_{IH3}	Input high voltage, XTAL1		$0.5V_{DD}$ -0.3	-	V_{DD}+ 0.5	V
V_{IL1}	Input low voltage, P0/P1/P2/P3/P4 (*Quasi-bidirectional or Input-only*)		-0.5	-	$0.25V_{DD}$ +0.1	V
V_{IL2}	Input low voltage, RST		-0.5	-	$0.25V_{DD}$ +0.2	V
V_{IL3}	Input low voltage, XTAL1		-0.5	-	$0.35V_{DD}$ -0.1	V
V_{OH1}*[3]	Output high current, P0/P1/P2/P3/P4 (*Quasi-bidirectional*)	V_{DD}=2.7V and I_{OH}=-17μA V_{DD}=5.5V and I_{OH}=-210μA	2.4	-	-	V
V_{OH2}*[3]	Output high current, P0/P1/P2/P3/P4 (*Push-pull output*)	V_{DD}=2.7V and I_{OH}=-2.1mA V_{DD}=5.5V and I_{OH}=-25.0mA	2.4	-	-	V
V_{OH3}*[3]	Output high current, XTAL2	V_{DD}=2.7V and I_{OH}=-0.9mA V_{DD}=5.5V and I_{OH}=-9.4mA	2.4	-	-	V
V_{OL1}*[3]	Output low current, P0/P1/P2/P3/P4 (*Quasi-bidirectional*)	V_{DD}=2.7V and I_{OL}=+11.8mA V_{DD}=5.5V and I_{OL}=+17.6mA	-	-	0.4	V
V_{OL2}*[3]	Output low current, P0/P1/P2/P3/P4 (*Push-pull output*)	V_{DD}=2.7V and I_{OL}=+11.8mA V_{DD}=5.5V and I_{OL}=+17.6mA	-	-	0.4	V
V_{OL3}*[3]	Output low current, P0/P1/P2/P3/P4 (*Open-drain output*)	V_{DD}=2.7V and I_{OL}=+11.8mA V_{DD}=5.5V and I_{OL}=+17.6mA	-	-	0.4	V
V_{OL4}*[3]	Output low current, XTAL2	V_{DD}=2.7V and I_{OL}=+1.4mA V_{DD}=5.5V and I_{OL}=+1.6mA	-	-	0.4	V
R_{RST}	Internal reset pull-down resistor		110	-	280	KΩ
V_{OPF}	Brownout threshold, VDD power		-	3.7	-	V
V_{RAM}*[4]	RAM keep-alive voltage		1.5	-	-	V

(Continued)

Symbol	Parameter	Test Conditions	Min	Typ*[1]	Max	Unit
V_{DD}*[5, *8]	Power supply voltage	F_{osc}=6MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.1 1.8	 -	 5.5	V
		F_{osc}=12MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.2 1.9	 -	 5.5	
		F_{osc}=24MHz, T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 2.4 2.4 2.4	 -	 5.5	
V_{POR}	Power-on reset threshold voltage	T_{amb}= -40℃ T_{amb}= +25℃ T_{amb}= +85℃	 -	2.4 2.1 1.8	 -	V
I_{DD}*[6]	Power supply current, operating*[7]	V_{DD}=2.7V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	 -	 4.7 6.9 11.0	 6.0 8.5 13.5	mA
		V_{DD}=5.5V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	 -	 8.9 11.6 18.1	 11.0 14.0 22.0	
	Power supply current, idle	V_{DD}=2.7V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	 -	 1.9 2.9 5.0	 2.5 3.5 6.0	mA
		V_{DD}=5.5V F_{osc}=6MHz F_{osc}=12MHz F_{osc}=24MHz	 -	 4.1 5.1 8.1	 5.0 6.5 10.0	
	Power supply current, power-down	F_{osc}=24MHz, V_{DD}=2.7V~5.5V	-	1	10	μA

Note:

*1: Typical values are based on a limited number of samples and are not guaranteed. The values listed are at room temperature unless otherwise specified.

*2: Port pins source a transition current when used in quasi-bidirectional mode and externally driven from logic 1 to logic 0. This current is highest when V_{IN} is approximately 2.0V.

*3: See Section 29: Absolute Maximum Ratings for steady state (non-transient) limits on I_{OH} and I_{OL}.
If I_{OH} exceeds the test condition, V_{OH} will be lower than the listed specification.
If I_{OL} exceeds the test condition, V_{OL} will be higher than the listed specification.

*4: Supply voltage for RAM data retention.

*5: With the Hardware Option ENLVRO & ENLVRC disabled.

*6: With the Hardware Option OSCDN enabled.

*7: Tested while CPU running in a NOP loop, as shown below.

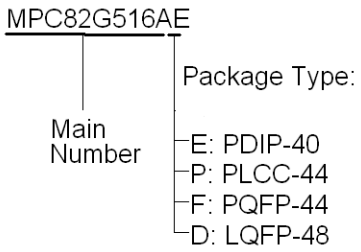
```
Loop:
    NOP
    JMP   Loop
```

*8: The listed minimum supply voltage is obtained under normal work of logic function, excluding Flash erase and write. The range 2.7V~5.5 is the supply voltage for normal work of whole chip function.

31 Ordering Information

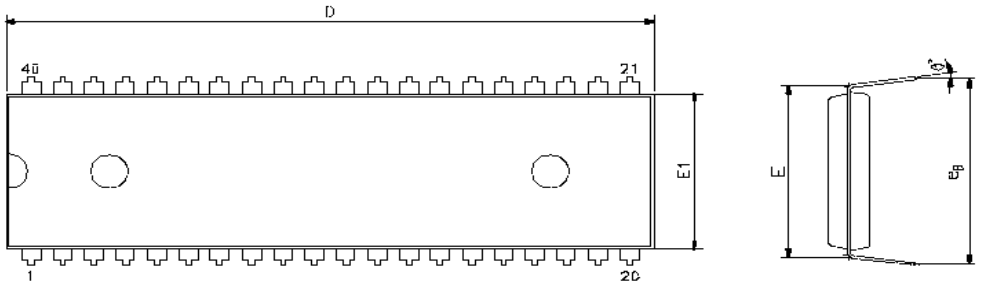
Part Number	Package		Packing
	Name	Description	
MPC82G516AE	PDIP-40	Plastic Dual In-line Package; 40 leads (600 mil)	Tube
MPC82G516AP	PLCC-44	Plastic Leaded Chip Carrier; 44 leads	Tube
MPC82G516AF	PQFP-44	Plastic Quad Flat Package; 44 leads; body 10x10x2.0 mm	Tray
MPC82G516AD	LQFP-48	Plastic Low-profile Quad Flat Package; 48 leads; body 7x7x1.4 mm	Tray

Part Number Derivation



32 Package Outline

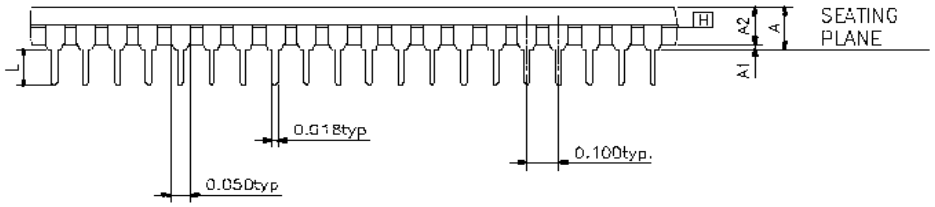
40-Pin PDIP Package



Symbols	MIN		NOR		MAX	
A					0.220	5.59
A1	0.015	0.38				
A2	0.150	3.81	0.155	3.94	0.160	4.06
D	2.055	52.20	2.060	52.32	2.070	52.58
E			0.600	15.24	SBC	
E1	0.540	13.72	0.545	13.84	0.550	13.97
L	0.115	2.92	0.130	3.30	0.150	3.81
E8	0.630	16.00	0.650	16.51	0.670	17.02
θ°	0	0	7	7	15	15

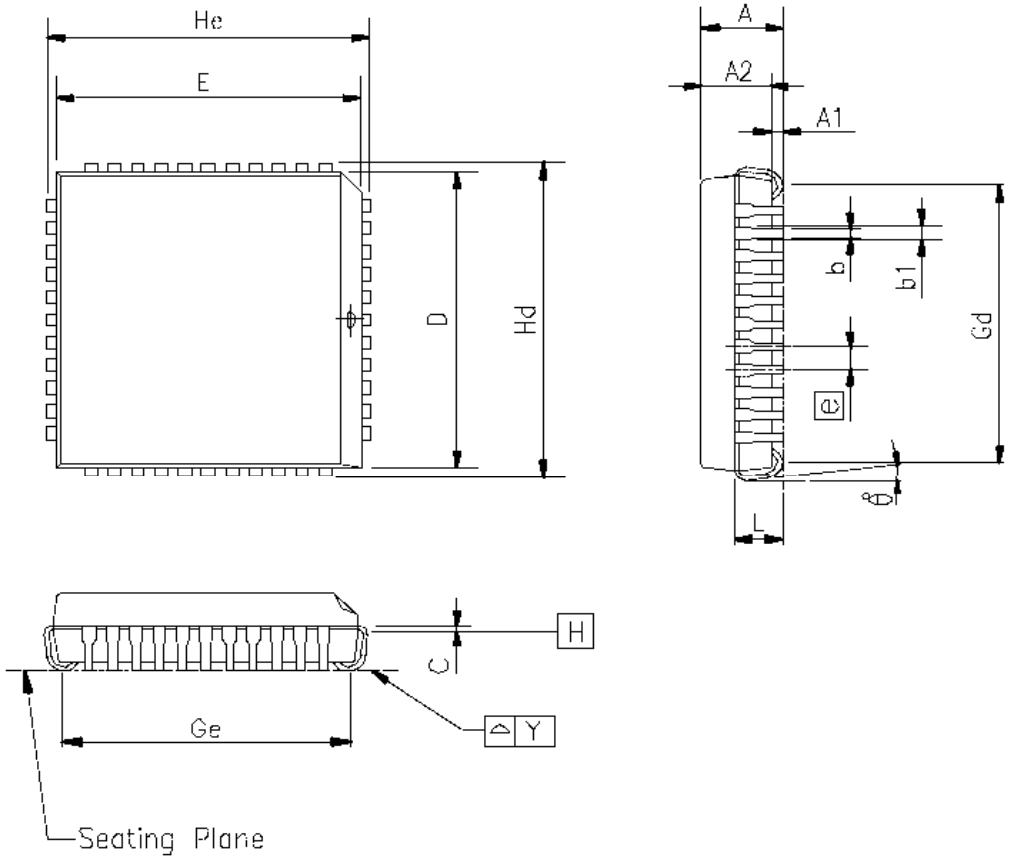
Inch/mm

NOTE:
1.JEDEC OUTLINE : MS-011 AC



華泉科技股份有限公司 Megawin Technology Co.,				
3RD ANGLE SYS.	比例 SCALE:	材質 MTRL:	製程 FINISH:	數量 QTY:
繪圖: DWN: 日期: DATE:	圖名: TITLE: DUAL INLINE PLASTIC DATA SHEET P-DIP 40 LEADS (600mil)			
檢核: CHK: 日期: DATE:	圖號: DWG#: J1-0140P-001			
核准: APPL: 日期: DATE:	圖檔: FILE: J1-0140P-001-01	版別: DATE: D1	張數: DATE: 1	

44-Pin PLCC Package



Symbols	MIN		NOR		MAX	
A					0.185	4.70
A1	0.020	0.51				
A2	0.145	3.68	0.150	3.81	0.155	3.94
b1	0.026	0.66	0.028	0.71	0.032	0.81
b	0.016	0.41	0.018	0.46	0.022	0.56
c	0.007	0.18	0.010	0.25	0.013	0.33
D	0.648	16.46	0.653	16.59	0.658	16.71
E	0.648	16.46	0.653	16.59	0.658	16.71
e			0.050	1.27	BSC	
Gd	0.590	14.99	0.610	15.49	0.630	16.00
Ge	0.590	14.99	0.610	15.49	0.630	16.00
Hd	0.680	17.27	0.690	17.53	0.700	17.78
He	0.680	17.27	0.690	17.53	0.700	17.78
L	0.090	2.29	0.100	2.54	0.110	2.79
Y					0.004	0.10

Inch/mm

* ***NOTE:***

1.JEDEC OUTLINE : MO-047 AC

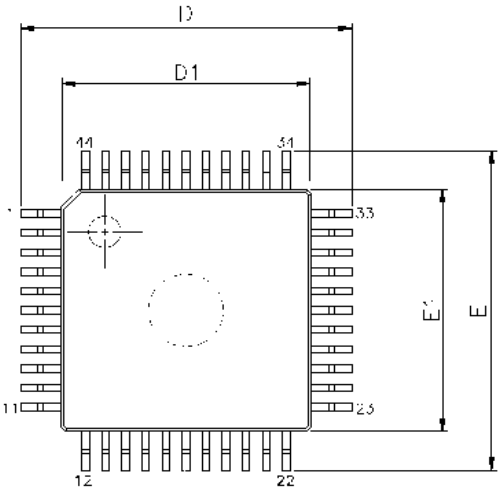
2.DATUM PLANE H IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY

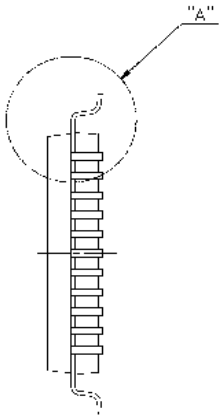
3.DIMENSIONS E AND D DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 10 MIL PER SIDE. DIMENSIONS E AND D DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.

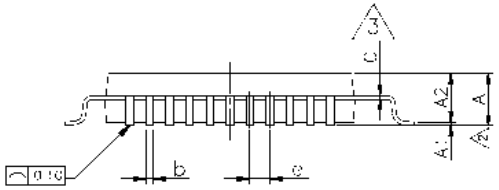
4.DIMENSION b1 DOES NOT INCLUDE DAMBAR PROTRUSION .

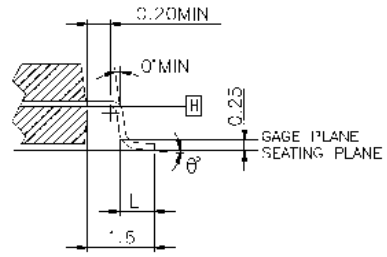
笙泉科技股份有限公司 Megawin Technology Co.,			
3RD ANGLE SYS	SCALE	MTRL	FINISH / QTY
DWN / DATE	TITLE: PLASTIC CHIP CARRIER DATA SHEET 44 LEADS 0.05" LEAD SPACING SQUARE		
CHK / DATE	NUMBER: J1-06044-001		
APPD / DATE	FILE: J-044P-1	REV: 02	QTY: 1

44-Pin PQFP Package









DETAIL A

Symbols	MIN		NOR		MAX	
A					0.106	2.70
A1	0.010	0.25	0.012	0.30	0.014	0.35
A2	0.075	1.90	0.079	2.00	0.087	2.20
b			0.012	0.30	(TYP.)	
D	0.512	13.00	0.520	13.20	0.528	13.4
D1	0.390	9.90	0.394	10.00	0.398	10.10
E	0.512	13.00	0.520	13.20	0.528	13.40
E1	0.390	9.90	0.394	10.00	0.398	10.10
L	0.029	0.73	0.035	0.88	0.037	0.93
e			0.031	0.80	(TYP.)	
θ°	0	0			7	7
C	0.004	0.10	0.006	0.15	0.008	0.2

Inch/mm

NOTES:

1.JEDEC OUTLINE:MO-108 AA-1

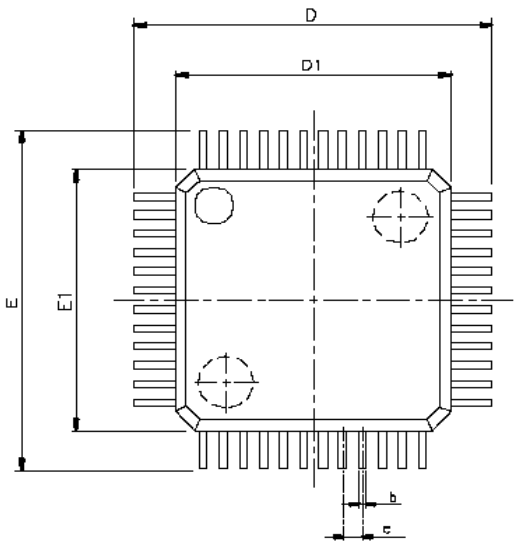
2.DATUM PLANE [H] IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.

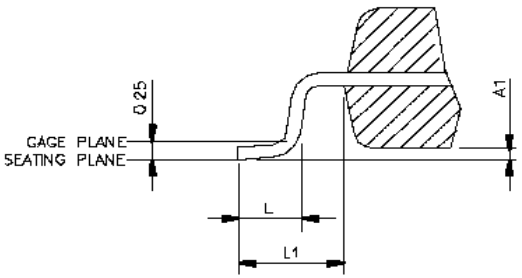
3.DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE [H].

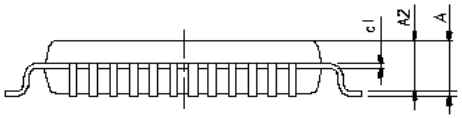
4.DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

笙泉科技股份有限公司 Megawin Technology Co.			
3RD ANGLE	比例 SCALE	材質 MTRL	製程 FINISH	數量 QTY
繪圖: DWN. 日期: DATE:	圖名: TITLE: METRIC PLASTIC QUAD FLAT PACKAGE OUTLINE 10X10MM BODY 44L DATA SHEET		
審核: CHK. 日期: DATE:	圖號: DWG# J1-0344Q-001		
核定: APPL. 日期: DATE:	圖檔: FILE: J-QFP44-1	版別: REV: 03	張數: QTY: 1

48-Pin LQFP Package







SYMBOL	MIN		NOR		MAX
A					1.6
A1	0.002	0.05		0.006	0.15
A2	0.053	1.35		0.057	1.45
c1	0.004	0.09		0.006	0.16
D	9.00 BSS				
D1	7.00 BSC				
E	9.00 BSD				
E1	7.00 BSC				
e	0.5 BSC				
b	0.007	0.17		0.011	0.27
L	0.018	0.45		0.030	0.75
L1					

Inch/mm

NOTES:

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

笙泉科技股份有限公司 Megawin Technology Co.,				
3RD ANGLE SYS.	比例 SCALE	材質 MTRL	製程 FINISH	數量 QTY
繪圖 DWN: 日期: DATE:	圖名: TITLE: THIN QUAD FLAT PACKAGE OUTLINE 48PIN 1.4MM THICK BODY DATA SHEET			
審核: CHK: 日期: DATE:	圖號: DWG#: J1-0848F-001			
核准: APPL. 日期: DATE:	圖檔: FILE: JQFP48-1	版別: REV: 02	張數: QTY: 1	

33 Disclaimers

Herein, Megawin stands for "***Megawin Technology Co., Ltd.***"

Life Support — This product is not designed for use in medical, life-saving or life-sustaining applications, or systems where malfunction of this product can reasonably be expected to result in personal injury. Customers using or selling this product for use in such applications do so at their own risk and agree to fully indemnify Megawin for any damages resulting from such improper use or sale.

Right to Make Changes — Megawin reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in mass production, relevant changes will be communicated via an Engineering Change Notification (ECN).

Revision History

Version	Date	Page	Description
A1	2007/07		- Initial issue.
A2	2008/03		- Modify some specifications.
A3	2008/06		- Rewrite the datasheet to enrich the contents.
A4	2008/12		- Formatting
A5	2010/04	17	- Modify SFR AUXR1 address error - Remove SSOP-28 package - Add V30 pin connected