SIEMENS

C517A

8-Bit CMOS Microcontroller

User's Manual 01.99

C517A User's Manual Revision History :		01.99
Previous Releases :		08.97 (Original Version)
Page (previous version)	Page (new version)	Subjects (changes since last revision)
All sections	All sections	V_{CC} is changed to V_{DD}.
1-2	1-2	"with wake-up capability through INT0 pin" is removed.
1-2	1-2	P-LCC-84 package is added under the main feature list.
1-4	1-4	P-LCC-84 package is added.
1-5	1-5	Figure 1-4; added.
1-5 to 1-10	1-6 to 1-13	Table 1-1; modified, column "P-LCC-84" is added.
1-8	1-9	Description for pin EA is added with "For C517A-4R ..."
6-24	6-24	Table 6-3; modified, column "Pin No. (P-LCC-84)" is added.
9-1	9-1	The whole page is added to contain brief explanation of the power saving modes.
9-2	9-1	Section 9.1 and 9.2 are added to give more explanation of pin $\overline{PE}$/SWD usage.
9-3	9-1	Last paragrapah; the sentence "Changing the logic level ..." in the note under the description of PCON is deleted.
9-5	9-3	1st paragraph; the sentence "If the idle mode ..." is added.
9-7	9-5	3rd paragraph; the paragraph is changed to contain an extra way to leave software power down mode.
9-7	9-5	4th paragraph; the sentece "If the software power down ..." is added.
9-9	9-7	Paragraph "A low signal at the P3.2/INT0 ...." is removed.
Chapter 10	-	The whole chapter is moved to the C517A Data Sheet.

Edition 01.99

This edition was realized using the software system FrameMaker®.

**Published by Siemens AG,
Bereich Halbleiter, Marketing-
Kommunikation, Balanstraße 73,
81541 München**

Contents **Page**

Contents **Page**

Contents **Page**

Contents **Page**

1 Introduction

The C517A is a high-end member of the Siemens C500 family of 8-bit microcontrollers. It is functionally fully compatible with the SAB-80C517A/83C537A-5 microcontrollers.

The C517A basically operates with internal and/or external program memory. The C517A-L is identical to the C517A-4R, except that it lacks the on-chip program memory. Therefore, in this documentation the term C517A refers to all versions within this specification unless otherwise noted.

Figure 1-1 shows the different functional units of the C517A and **figure 1-2** shows the simplified logic symbol of the C517A.

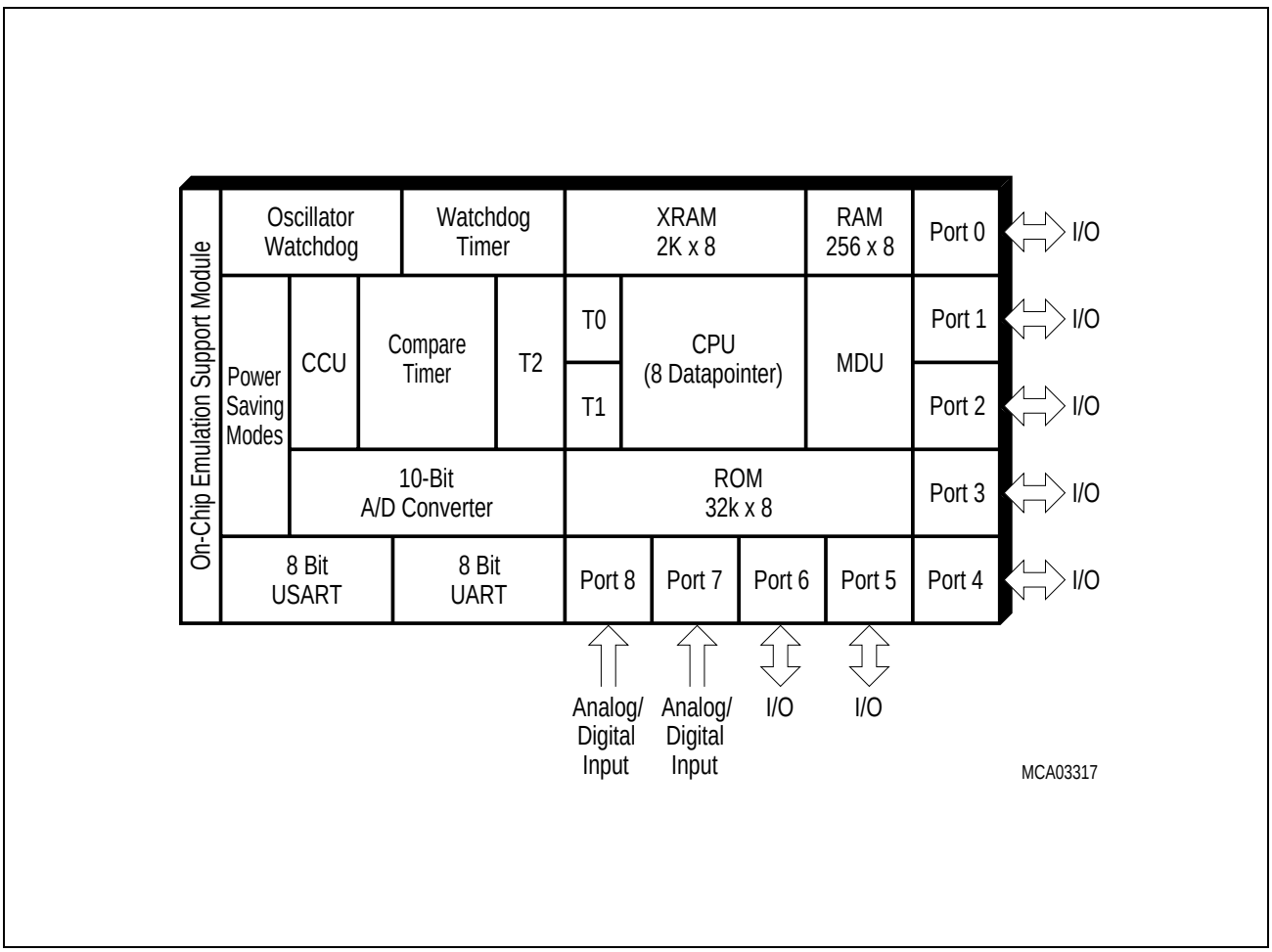


Figure 1-1
C517A Functional Units

Listed below is a summary of the main features of the C517A:

- Full upward compatibility with SAB 80C517A/83C517A-5
- Up to 24 MHz external operating frequency
  - 500 ns instruction cycle at 24 MHz operation
- Superset of the 8051 architecture with 8 datapointers
- 32K byte on-chip ROM (with optional ROM protection)
  - alternatively up to 64K byte external program memory
- Up to 64K byte external data memory
- 256 byte on-chip RAM
- 2K byte on-chip RAM (XRAM)
- Seven 8-bit parallel I/O ports
- Two input ports for analog/digital input
- Two full duplex serial interfaces (USART)
  - 4 operating modes, fixed or variabie baud rates
  - programmable baud rate generators
- Four 16-bit timer/counters
  - Timer 0 / 1 (C501 compatible)
  - Timer 2 for 16-bit reload, compare, or capture functions
  - Compare timer for compare/capture functions
- Powerful 16-bit compare/capture unt (CCU) with up to 21 high-speed or PWM output channels and 5 capture inputs
- 10-bit A/D converter
  - 12 multiplexed analog inputs
  - Built-in self calibration
- Extended watchdog facilities
  - 15-bit programmable watchdog timer
  - Oscillator watchdog
- Power saving modes
  - Slow down mode
  - Idle mode (can be combined with slow down mode)
  - Software power down mode
  - Hardware power down mode
- 17 interrupt sources (7 external, 10 internal) selectable at 4 priority levels
- On-chip emulation support logic (Enhanced Hooks Technology ^{TM})
- P-MQFP-100 and P-LCC-84 packages
- Temperature Ranges : SAB-C517A T_A = 0 to 70 °C
  SAF-C517A T_A = -40 to 85 °C
  SAH-C517A T_A = -40 to 110 °C

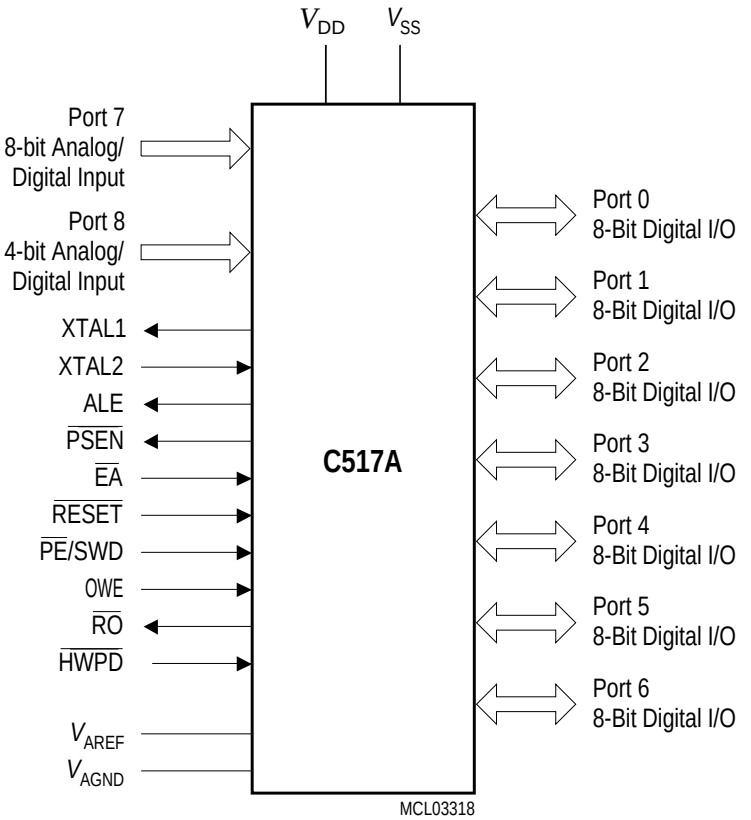


Figure 1-2
Logic Symbol

1.1 Pin Configuration

This section describes the pin configration of the C517A in the P-MQFP-100 and P-LCC-84 packages.

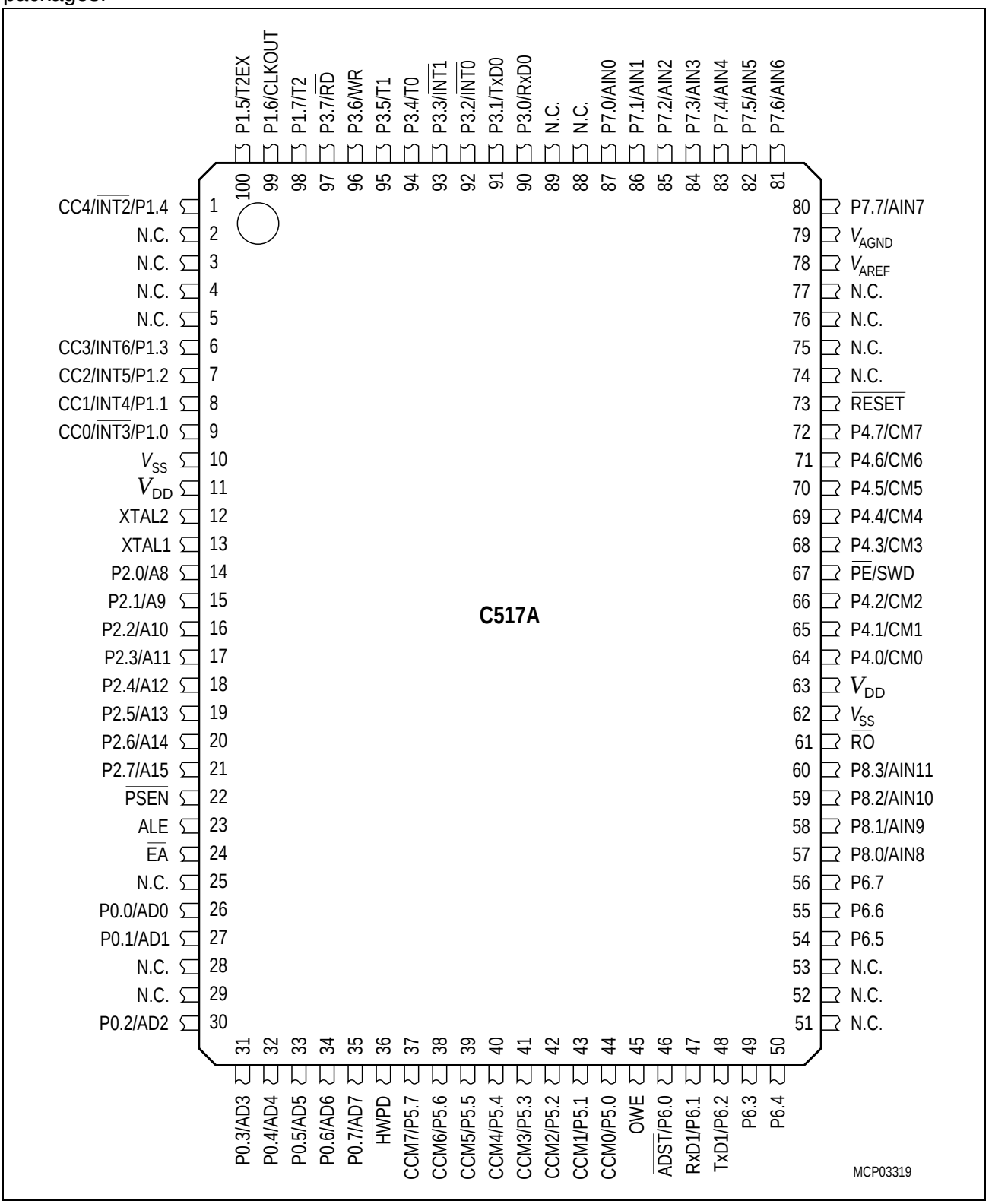


Figure 1-3
Pin Configuration P-MQFP-100 Package (Top View)

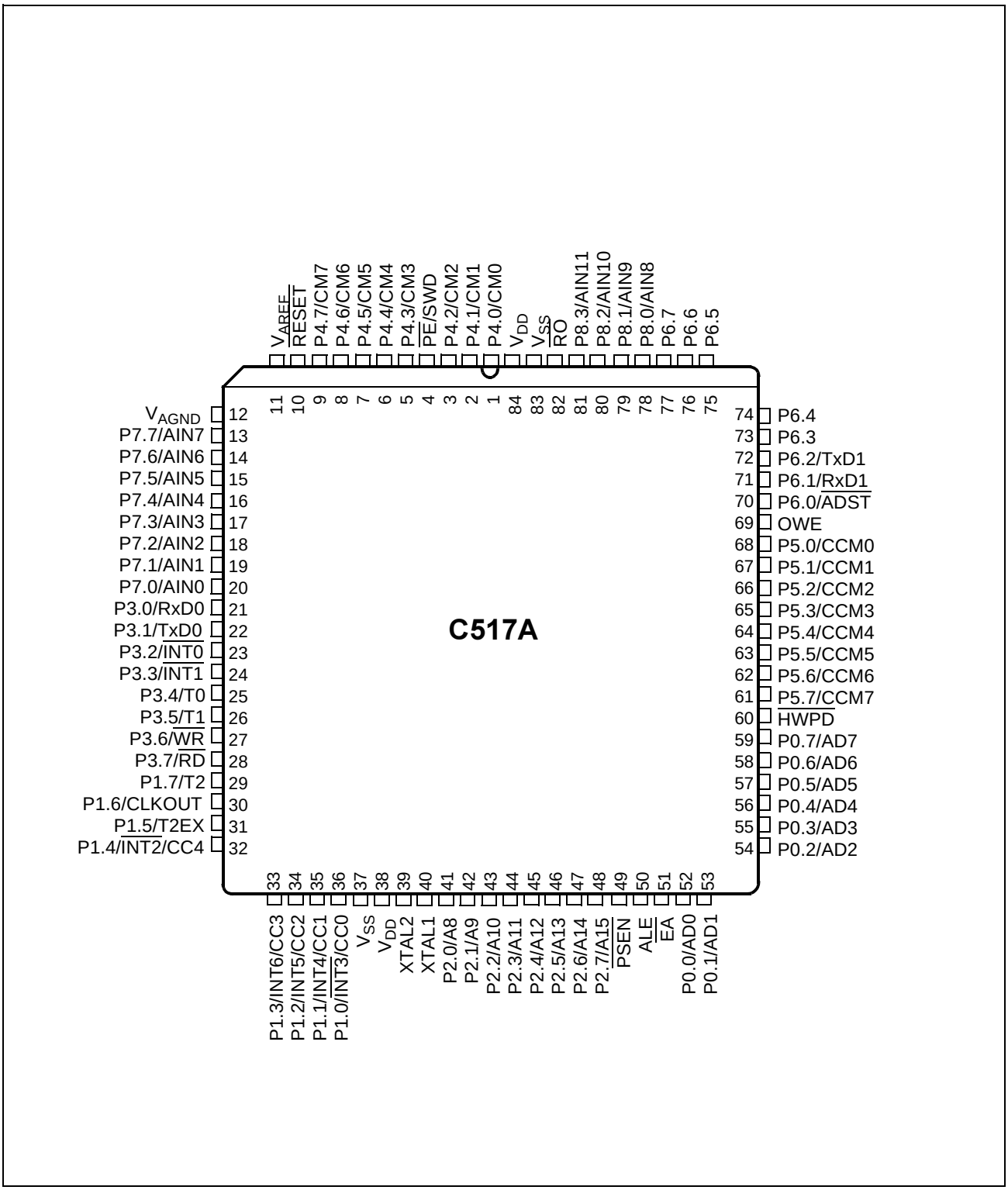


Figure 1-4
Pin Configuration P-LCC-84 Package (Top View)

1.2 Pin Definitions and Functions

This section describes all external signals of the C517A with its function.

Table 1-1
Pin Definitions and Functions

Symbol	Pin Number		I/O*)	Function
	P-MQFP-100	P-LCC-84		
P1.0 - P1.7	9 - 6, 1, 100 - 98	36 - 29	I/O	**Port 1** is an 8-bit quasi-bidirectional I/O port with internal pullup resistors. Port 1 pins that have 1's written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. As inputs, port 1 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. The port is used for the low-order address byte during program verification. Port 1 also contains the interrupt, timer, clock, capture and compare pins that are used by various options. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate (except when used for the compare functions). The secondary functions are assigned to the port 1 pins as follows :
	9	36		P1.0 / $\overline{\text{INT3}}$ / CC0 Interrupt 3 input / compare 0 output / capture 0 input
	8	35		P1.1 / INT4 / CC1 Interrupt 4 input / compare 1 output / capture 1 input
	7	34		P1.2 / INT5 / CC2 Interrupt 5 input / compare 2 output / capture 2 input
	6	33		P1.3 / INT6 / CC3 Interrupt 6 input / compare 3 output / capture 3 input
	1	32		P1.4 / $\overline{\text{INT2}}$ / CC4 Interrupt 2 input / compare 4 output / capture 4 input
	100	31		P1.5 / T2EX Timer 2 external reload / trigger input
	99	30		P1.6 / CLKOUT System clock output
	98	29		P1.7 / T2 Counter 2 input

*) I = Input,
O = Output

Table 1-1
Pin Definitions and Functions (cont'd)

Symbol	Pin Number		I/O*)	Function
	P-MQFP-100	P-LCC-84		
V_{SS}	10, 62	37, 83	–	**Ground** (0V) during normal, idle, and power down operation.
V_{DD}	11, 63	38, 84	–	**Supply voltage** during normal, idle, and power down mode.
XTAL2	12	39	–	**XTAL2** is the input to the inverting oscillator amplifier and input to the internal clock generator circuits. To drive the device from an external clock source, XTAL2 should be driven, while XTAL1 is left unconnected. Minimum and maximum high and low times as well as rise/ fall times specified in the AC characteristics must be observed.
XTAL1	13	40	–	**XTAL1** is the output of the inverting oscillator amplifier. This pin is used for the oscillator operation with crystal or ceramic resonator.
P2.0 - P2.7	14 - 21	41 - 48	I/O	**Port 2** is an 8-bit quasi-bidirectional I/O port with internal pullup resistors. Port 2 pins that have 1's written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. As inputs, port 2 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullup resistors when issuing 1's. During accesses to external data memory that use 8-bit addresses (MOVX @Ri), port 2 issues the contents of the P2 special function register.

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions (cont'd)

Symbol	Pin Number		I/O*)	Function
	P-MQFP-100	P-LCC-84		
$\overline{\text{PSEN}}$	22	49	O	The **Program Store Enable** output is a control signal that enables the external program memory to the bus during external fetch operations. It is activated every six oscillator periods except during external data memory accesses. The signal remains high during internal program execution.
ALE	23	50	O	The **Address Latch enable** output is used for latching the address into external memory during normal operation. It is activated every six oscillator periods except during an external data memory access.
$\overline{\text{EA}}$	24	51	I	**External Access Enable** When held high, the C517A executes instructions from the internal ROM as long as the PC is less than 8000_H. When held low, the C517A fetches all instructions from external program memory. For the C517A-L this pin must be tied low. For the C517A-4R, if the device is protected (**see section 4.6**) then this pin is only latched during reset.
P0.0 - P0.7	26, 27, 30 - 35	52 - 59	I/O	**Port 0** is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1's written to them float, and in that state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pullup resistors when issuing 1's. Port 0 also outputs the code bytes during program verification in the C517A-4R. External pullup resistors are required during program verification.

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions (cont'd)

Symbol	Pin Number		I/O*)	Function
	P-MQFP-100	P-LCC-84		
$\overline{\text{HWPD}}$	36	60	I	**$\overline{\text{Hardware Power Down}}$** A low level on this pin for the duration of one machine cycle while the oscillator is running resets the C517A. A low level for a longer period will force the part into hardware power down mode with the pins floating. There is no internal pullup resistor connected to this pin.
P5.0 - P5.7	44 - 37	68 - 61	I/O	**Port 5** is a quasi-bidirectional I/O port with internal pull-up resistors. Port 5 pins that have 1 s written to them are pulled high by the internal pull-up resistors, and in that state can be used as inputs. As inputs, port 5 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pull-up resistors. This port also serves the alternate function "Concurrent Compare" and "Set/Reset Compare". The secondary functions are assigned to the port 5 pins as follows: CCM0 to CCM7 P5.0 to P5.7 : concurrent compare or Set/Reset lines
OWE	45	69	I	**Oscillator Watchdog Enable** A high level on this pin enables the oscillator watchdog. When left unconnected this pin is pulled high by a weak internal pull-up resisitor. The logic level at OWE should not be changed during normal operation. When held at low level the oscillator watchdog function is turned off. During hardware power down the pullup resistor is switched off.

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions (cont'd)

Symbol	Pin Number		I/O*)	Function
	P-MQFP-100	P-LCC-84		
P6.0 - P6.7	46 - 50, 54 - 56	70 - 77	I/O	**Port 6** is a quasi-bidirectional I/O port with internal pull-up resistors. Port 6 pins that have 1 s written to them are pulled high by the internal pull-up resistors, and in that state can be used as inputs. As inputs, port 6 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pull-up resistors. Port 6 also contains the external A/D converter start control pin and the transmit and receive pins for the serial interface 1. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the pins of port 6, as follows : P6.0 $\overline{\text{ADST}}$ external A/D converter start pin P6.1 RxD1 receiver data input of serial interface 1 P6.2 TxD1 transmitter data input of serial interface 1
	46	70		
	47	71		
	48	72		
P8.0 - P8.3	57 - 60	78 - 81	I	**Port 8** is a 4-bit unidirectional input port. Port pins can be used for digital input, if voltage levels meet the specified input high/low voltages, and for the higher 4-bit of the multiplexed analog inputs of the A/D converter, simultaneously. P8.0 - P8.3 AIN8 - AIN11 analog input 8 - 11
$\overline{\text{RO}}$	61	82	O	**$\overline{\text{Reset Output}}$** This pin outputs the internally synchronized reset request signal. This signal may be generated by an external hardware reset, a watchdog timer reset or an oscillator watchdog reset. The $\overline{\text{RO}}$ output signal is active low.

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions (cont'd)

Symbol	Pin Number		I/O*)	Function
	P-MQFP-100	P-LCC-84		
P4.0 - P4.7	64 - 66, 68 - 72	1 - 3, 5 - 9	I/O	**Port 4** is an 8-bit quasi-bidirectional I/O port with internal pull-up resistors. Port 4 pins that have 1's written to them are pulled high by the internal pull-up resistors, and in that state can be used as inputs. As inputs, port 4 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pull-up resistors. Port 4 also serves as alternate compare functions. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the pins of port 4 as follows : P4.0 - P4.7 CM0 - CM7 Compare channel 0 - 7
$\overline{PE}$/SWD	67	4	I	**$\overline{\text{Power saving mode enable}}$ / Start watchdog timer** A low level at this pin allows the software to enter the power saving modes (idle mode, slow down mode, and power down mode). In case the low level is also seen during reset, the watchdog timer function is off on default. Usage of the software controlled power saving modes is blocked, when this pin is held at high level. A high level during reset performs an automatic start of the watchdog timer immediately after reset. When left unconnected this pin is pulled high by a weak internal pull-up resistor. During hardware power down the pullup resisitor is switched off.

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions (cont'd)

Symbol	Pin Number		I/O*)	Function
	P-MQFP-100	P-LCC-84		
P3.0 - P3.7	90 - 97	21 - 28	I/O	**Port 3** is an 8-bit quasi-bidirectional I/O port with internal pullup resistors. Port 3 pins that have 1's written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. As inputs, port 3 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 3 also contains the interrupt, timer, serial port and external memory strobe pins that are used by various options. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the pins of port 3, as follows:
	90	21		P3.0 / RxD0 — Receiver data input (asynch.) or data input/output (synch.)of serial interface 0
	91	22		P3.1 / TxD0 — Transmitter data output (asynch.) or clock output (synch.) of serial interface 0
	92	23		P3.2 / $\overline{\text{INT0}}$ — External interrupt 0 input / timer 0 gate control input
	93	24		P3.3 / $\overline{\text{INT1}}$ — External interrupt 1 input / timer 1 gate control input
	94	25		P3.4 / T0 — Timer 0 counter input
	95	26		P3.5 / T1 — Timer 1 counter input
	96	27		P3.6 / $\overline{\text{WR}}$ — $\overline{\text{WR}}$ control output; latches the data byte from port 0 into the external data memory
	97	28		P3.7 / $\overline{\text{RD}}$ — $\overline{\text{RD}}$ control output; enables the external data memory

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions (cont'd)

Symbol	Pin Number		I/O*)	Function
	P-MQFP-100	P-LCC-84		
$\overline{\text{RESET}}$	73	10	I	**$\overline{\text{RESET}}$** A low level on this pin for the duration of two machine cycles while the oscillator is running resets the C517A. A small internal pullup resistor permits power-on reset using only a capacitor connected to V_{SS}.
V_{AREF}	78	11	–	**Reference voltage** for the A/D converter
V_{AGND}	79	12	–	**Reference ground** for the A/D converter
P7.0 - P7.7	87 - 80	20-13	I	**Port 7** is an 8-bit unidirectional input port. Port pins can be used for digital input, if voltage levels meet the specified input high/low voltages, and for the lower 8-bit of the multiplexed analog inputs of the A/D converter, simultaneously. P7.0 - P7.7 AIN0 - AIN7 analog input 0 - 7
N.C.	2 - 5, 25, 28, 29, 51 - 53, 74 - 77 88, 89	–	–	**Not connected** These pins of the P-MQFP-100 package must not be connected.

*) I = Input
O = Output

2 Fundamental Structure

The C517A is fully compatible to the architecture of the standard 8051/C501 microcontroller family. While maintaining all architectural and operational characteristics of the C501, the C517A incorporates a CPU with 8 datapointers, a genuine 10-bit A/D converter, a capture/compare unit, two USART serial interfaces, a XRAM data memory as well as some enhancements in the Fail Save Mechanism Unit.

Figure 2-1 shows a block diagram of the C517A.

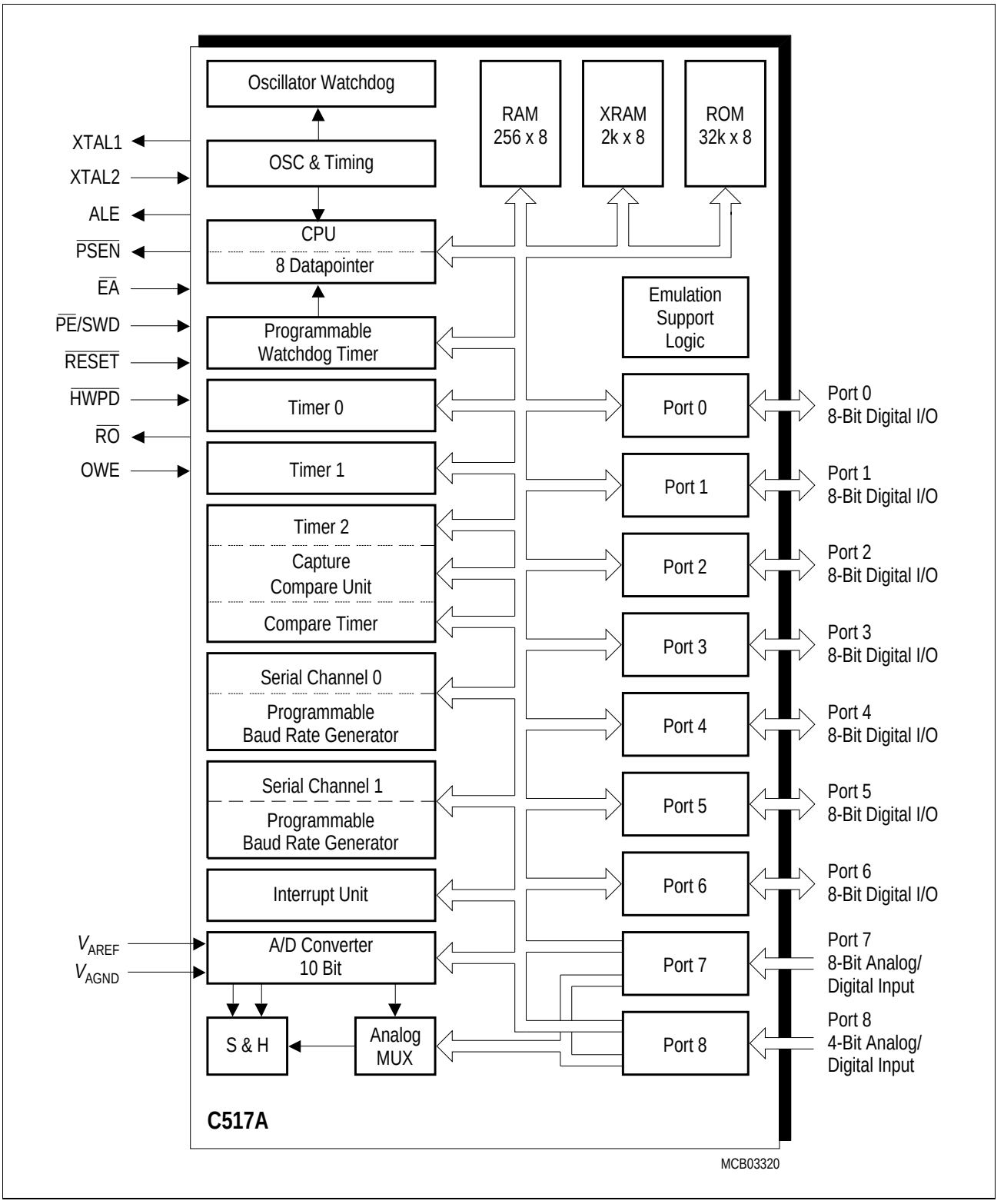


Figure 2-1
Block Diagram of the C517A

2.1 CPU

The CPU is designed to operate on bits and bytes. The instructions, which consist of up to 3 bytes, are performed in one, two or four machine cycles. One machine cycle requires six oscillator cycles (this number of oscillator cycles differs from other members of the C500 microcontroller family). The instruction set has extensive facilities for data transfer, logic and arithmetic instructions. The Boolean processor has its own full-featured and bit-based instructions within the instruction set. The C517A uses five addressing modes: direct access, immediate, register, register indirect access, and for accessing the external data or program memory portions a base register plus index-register indirect addressing. Efficient use of program memory results from an instruction set consisting of 44% one-byte, 41% two-byte, and 15% three-byte instructions. With a 24 MHz clock, 58% of the instructions execute in 500 ns.

The CPU (Central Processing Unit) of the C517A consists of the instruction decoder, the arithmetic section and the program control section. Each program instruction is decoded by the instruction decoder. This unit generates the internal signals controlling the functions of the individual units within the CPU. They have an effect on the source and destination of data transfers and control the ALU processing.

The arithmetic section of the processor performs extensive data manipulation and is comprised of the arithmetic/logic unit (ALU), an A register, B register and PSW register.

The ALU accepts 8-bit data words from one or two sources and generates an 8-bit result under the control of the instruction decoder. The ALU performs the arithmetic operations add, substract, multiply, divide, increment, decrement, BDC-decimal-add-adjust and compare, and the logic operations AND, OR, Exclusive OR, complement and rotate (right, left or swap nibble (left four)). Also included is a Boolean processor performing the bit operations as set, clear, complement, jump-if-not-set, jump-if-set-and-clear and move to/from carry. Between any addressable bit (or its complement) and the carry flag, it can perform the bit operations of logical AND or logical OR with the result returned to the carry flag.

The program control section controls the sequence in which the instructions stored in program memory are executed. The 16-bit program counter (PC) holds the address of the next instruction to be executed. The conditional branch logic enables internal and external events to the processor to cause a change in the program execution sequence.

Accumulator

ACC is the symbol for the accumulator register. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as A.

Program Status Word

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU.

Special Function Register PSW (Address $D0_H$) **Reset Value : 00_H**

Bit No.	MSB							LSB	
	$D7_H$	$D6_H$	$D5_H$	$D4_H$	$D3_H$	$D2_H$	$D1_H$	$D0_H$	
$D0_H$	CY	AC	F0	RS1	RS0	OV	F1	P	PSW

Bit	Function
CY	Carry Flag Used by arithmetic instruction.
AC	Auxiliary Carry Flag Used by instructions which execute BCD operations.
F0	General Purpose Flag
RS1 RS0	Register Bank select control bits These bits are used to select one of the four register banks.
OV	Overflow Flag Used by arithmetic instruction.
F1	General Purpose Flag
P	Parity Flag Set/cleared by hardware after each instruction to indicate an odd/even number of "one" bits in the accumulator, i.e. even parity.

RS1	RS0	Function
0	0	Bank 0 selected, data address 00_H-07_H
0	1	Bank 1 selected, data address 08_H-$0F_H$
1	0	Bank 2 selected, data address 10_H-17_H
1	1	Bank 3 selected, data address 18_H-$1F_H$

B Register

The B register is used during multiply and divide and serves as both source and destination. For other instructions it can be treated as another scratch pad register.

Stack Pointer

The stack pointer (SP) register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions and decremented after data is popped during a POP and RET (RETI) execution, i.e. it always points to the last valid stack byte. While the stack may reside anywhere in the on-chip RAM, the stack pointer is initialized to 07_H after a reset. This causes the stack to begin a location = 08_H above register bank zero. The SP can be read or written under software control.

2.2 CPU Timing

A machine cycle of the C517A consists of 6 states (12 oscillator periods). Each state is devided into a phase 1 half and a phase 2 half. Thus, a machine cycle consists of 12 oscillator periods, numbererd S1P1 (state 1, phase 1) through S6P2 (state 6, phase 2). Each state lasts one oscillator period. Typically, arithmetic and logic operations take place during phase 1 and internal register-to-register transfers take place during phase 2.

The diagrams in **figure 2-2** show the fetch/execute timing related to the internal states and phases. Since these internal clock signals are not user-accessible, the XTAL1 oscillator signals and the ALE (address latch enable) signal are shown for external reference. ALE is normally activated twice during each machine cycle: once during S1P2 and S2P1, and again during S4P2 and S5P1.

Executing of a one-cycle instruction begins at S1P2, when the op-code is latched into the instruction register. If it is a two-byte instruction, the second reading takes place during S4 of the same machine cycle. If it is a one-byte instruction, there is still a fetch at S4, but the byte read (which would be the next op-code) is ignored (discarded fetch), and the program counter is not incremented. In any case, execution is completed at the end of S6P2.

Figures 2-2 (a) and **(b)** show the timing of a 1-byte, 1-cycle instruction and for a 2-byte, 1-cycle instruction.

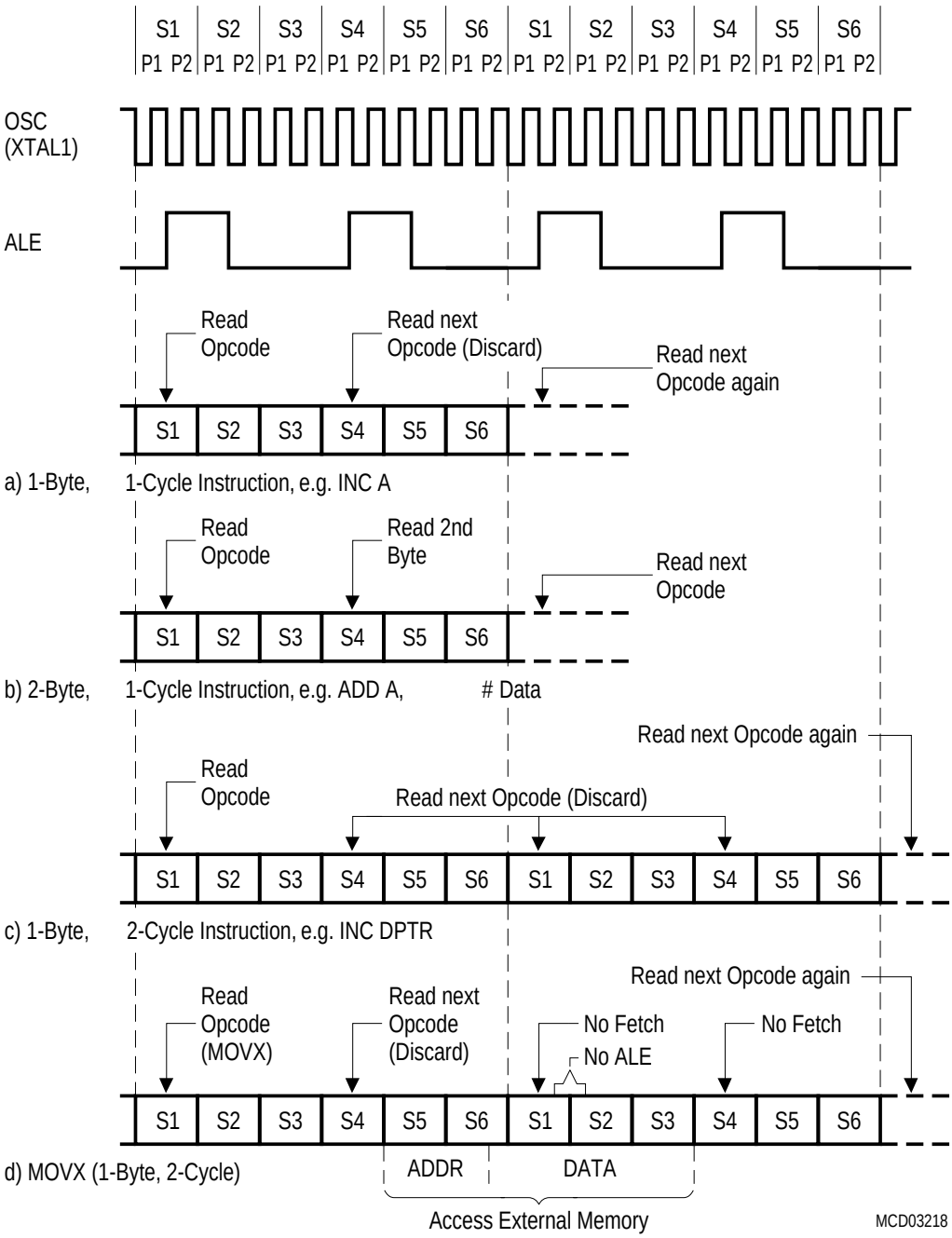


Figure 2-2
Fetch Execute Sequence

3 Memory Organization

The C517A CPU manipulates operands in the following four address spaces:

- up to 64 Kbyte of internal/external program memory
- up to 64 Kbyte of external data memory
- 256 bytes of internal data memory
- 2K bytes of internal XRAM data memory
- a 128 byte special function register area

Figure 3-1 illustrates the memory address spaces of the C517A.

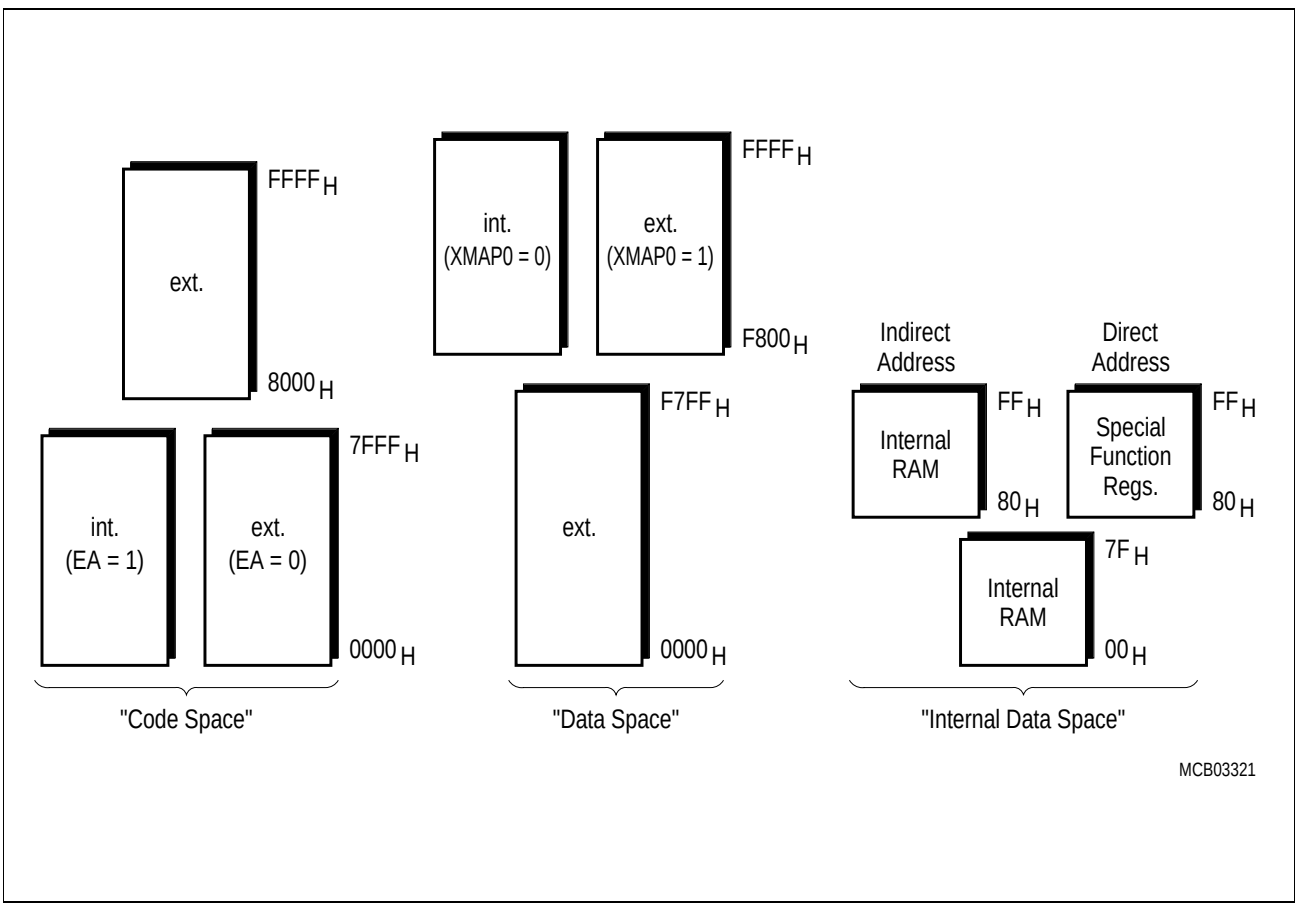


Figure 3-1
C517A Memory Map

3.1 Program Memory, "Code Space"

The C517A-4R has 32 Kbytes of read-only program memory which can be externally expanded up to 64 Kbytes. If the $\overline{EA}$ pin is held high, the C517A-4R executes program code out of the internal ROM unless the program counter address exceeds $7FFF_H$. Address locations 8000_H through $FFFF_H$ are then fetched from the external program memory. If the $\overline{EA}$ pin is held low, the C517A fetches all instructions from the external 64K byte program memory.

3.2 Data Memory, "Data Space"

The data memory address space consists of an internal and an external memory space. The internal data memory is divided into three physically separate and distinct blocks : the lower 128 bytes of RAM, the upper 128 bytes of RAM, and the 128 byte special function register (SFR) area. While the upper 128 bytes of data memory and the SFR area share the same address locations, they are accessed through different addressing modes. The lower 128 bytes of data memory can be accessed through direct or register indirect addressing; the upper 128 bytes of RAM can be accessed through register indirect addressing; the special function registers are accessible through direct addressing. Four 8-register banks, each bank consisting of eight 8-bit multi-purpose registers, occupy locations 0 through $1F_H$ in the lower RAM area. The next 16 bytes, locations 20_H through $2F_H$, contain 128 directly addressable bit locations. The stack can be located anywhere in the internal data memory address space, and the stack depth can be expanded up to 256 bytes.

The external data memory can be expanded up to 64 Kbyte and can be accessed by instructions that use a 16-bit or an 8-bit address. The internal XRAM is located in the external address memory area at addresses $F800_H$ to $FFFF_H$. Using MOVX instruction with addresses pointing to this address area, alternatively internal XRAM or external data RAM are accessed.

3.3 General Purpose Registers

The lower 32 locations of the internal RAM are assigned to four banks with eight general purpose registers (GPRs) each. Only one of these banks may be enabled at a time. Two bits in the program status word, RS0 (PSW.3) and RS1 (PSW.4), select the active register bank (see description of the PSW in chapter 2). This allows fast context switching, which is useful when entering subroutines or interrupt service routines.

The 8 general purpose registers of the selected register bank may be accessed by register addressing. With register addressing the instruction op code indicates which register is to be used. For indirect addressing R0 and R1 are used as pointer or index register to address internal or external memory (e.g. MOV @R0).

Reset initializes the stack pointer to location 07_H and increments it once to start from location 08_H which is also the first register (R0) of register bank 1. Thus, if one is going to use more than one register bank, the SP should be initialized to a different location of the RAM which is not used for data storage.

3.4 XRAM Operation

The XRAM in the C517A is a memory area that is logically located at the upper end of the external memory space, but is integrated on the chip. Because the XRAM is used in the same way as external data memory the same instruction types (MOVX) must be used for accessing the XRAM.

3.4.1 XRAM Access Control

Two bits in SFR SYSCON, XMAP0 and XMAP1, control the accesses to the XRAM. XMAP0 is a general access enable/disable control bit and XMAP1 controls the external signal generation during XRAM accesses.

Special Function Register SYSCON (Address $B1_H$) **Reset Value : $XXXXXX01_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$B1_H$	–	–	–	–	–	–	XMAP1	XMAP0	SYSCON

The functions of the shaded bits are not described in this section.

Bit	Function
–	Reserved bits for future use.
XMAP1	XRAM visible access control Control bit for $\overline{RD}$/$\overline{WR}$ signals during XRAMaccesses. If addresses are outside the XRAM address range or if XRAM is disabled, this bit has no effect. XMAP1 = 0 : The signals $\overline{RD}$ and $\overline{WR}$ are not activated during accesses to the XRAM XMAP1 = 1 : Ports 0, 2 and the signals $\overline{RD}$ and $\overline{WR}$ are activated during accesses to XRAM. In this mode, address and data information during XRAM accesses are visible externally.
XMAP0	Global XRAM access enable/disable control XMAP0 = 0 : The access to XRAM is enabled. XMAP0 = 1 : The access to XRAM is disabled (default after reset!). All MOVX accesses are performed via the external bus. Further, this bit is hardware protected.

When bit XMAP1 in SFR SYSCON is set, during all accesses to XRAM $\overline{RD}$ and $\overline{WR}$ become active and port 0 and 2 drive the actual address/data information which is read/written from/to XRAM. This feature allows to check externally the internal data transfers to the XRAM. When port 0 and 2 are used for I/O purposes, the XMAP1 bit should not be set. Otherwise the I/O function of the port 0 and port 2 lines is interrupted.

After a reset operation, bit XMAP0 is reset. This means that the accesses to the XRAM are generally disabled. In this case, all accesses using MOVX instructions within the address range of $F800_H$ to $FFFF_H$ generate external data memory bus cycles. When XMAP0 is set, the access to the XRAM is enabled and all accesses using MOVX instructions with an address in the range of $F800_H$ to $FFFF_H$ will access internally the XRAM.

Bit XMAP0 is hardware protected. If it is reset once (XRAM access enabled) it cannot be set by software. Only a reset operation will set the XMAP0 bit again. This hardware protection mechanism is done by an unsymmetric latch at the XMAP0 bit. A unintentional disabling of XRAM could be dangerous since indeterminate values could be read from teh external bus. To avoid this the XMAP0 bit is forced to '1' only by a reset operation. Additionally, during reset an internal capacitor is loaded. So the reset state is a disabled XRAM. Because of the load time of the capacitor, XMAP0 bit once written to '0' (that is, discharging the capacitor) cannot be set to '1' again by software. On the other hand, any distortion (software hang up, noise,...) is not able to load this capacitor, too. That is, the stable status is XRAM enabled.

The clear instruction for the XMAP0 bit should be integrated in the program initialization routine before the XRAM is used. In extremely noisy systems the user may have redundant clear instructions.

3.4.2 Accesses to XRAM using the DPTR (16-bit Addressing Mode)

The XRAM can be accessed by two read/write instructions, which use the 16-bit DPTR for indirect addressing. These instructions are :

- MOVX A, @DPTR (Read)
- MOVX @DPTR, A (Write)

For accessing the XRAM, the effective address stored in DPTR must be in the range of $F800_H$ to $FFFF_H$.

3.4.3 Accesses to XRAM using the Registers R0/R1

The 8051 architecture provides also instructions for accesses to external data memory range which use only an 8-bit address (indirect addressing with registers R0 or R1). The instructions are:

MOVX A, @ Ri (Read)
MOVX @Ri, A (Write)

In application systems, either a real 8-bit bus (with 8-bit address) is used or Port 2 serves as page register which selects pages of 256-Byte. However, the distinction, whether Port 2 is used as general purpose I/0 or as "page address" is made by the external system design. From the device's point of view it cannot be decided whether the Port 2 data is used externally as address or as I/0 data.

Hence, a special page register is implemented into the C517A to provide the possibility of accessing the XRAM also with the MOVX @Ri instructions, i.e. XPAGE serves the same function for the XRAM as Port 2 for external data memory.

Special Function Register XPAGE (Address 91_H) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
91_H	.7	.6	.5	.4	.3	.2	.1	.0	XPAGE

Bit	Function
XPAGE.7-0	XRAM high address XPAGE.7-0 is the address part A15-A8 when 8-bit MOVX instructions are used to access the internal XRAM.

Figures 3-2 to **3-4** show the dependencies of XPAGE- and Port 2 - addressing in order to explain the differences in accessing XRAM, ext. RAM or what is to do when Port 2 is used as an I/O-port.

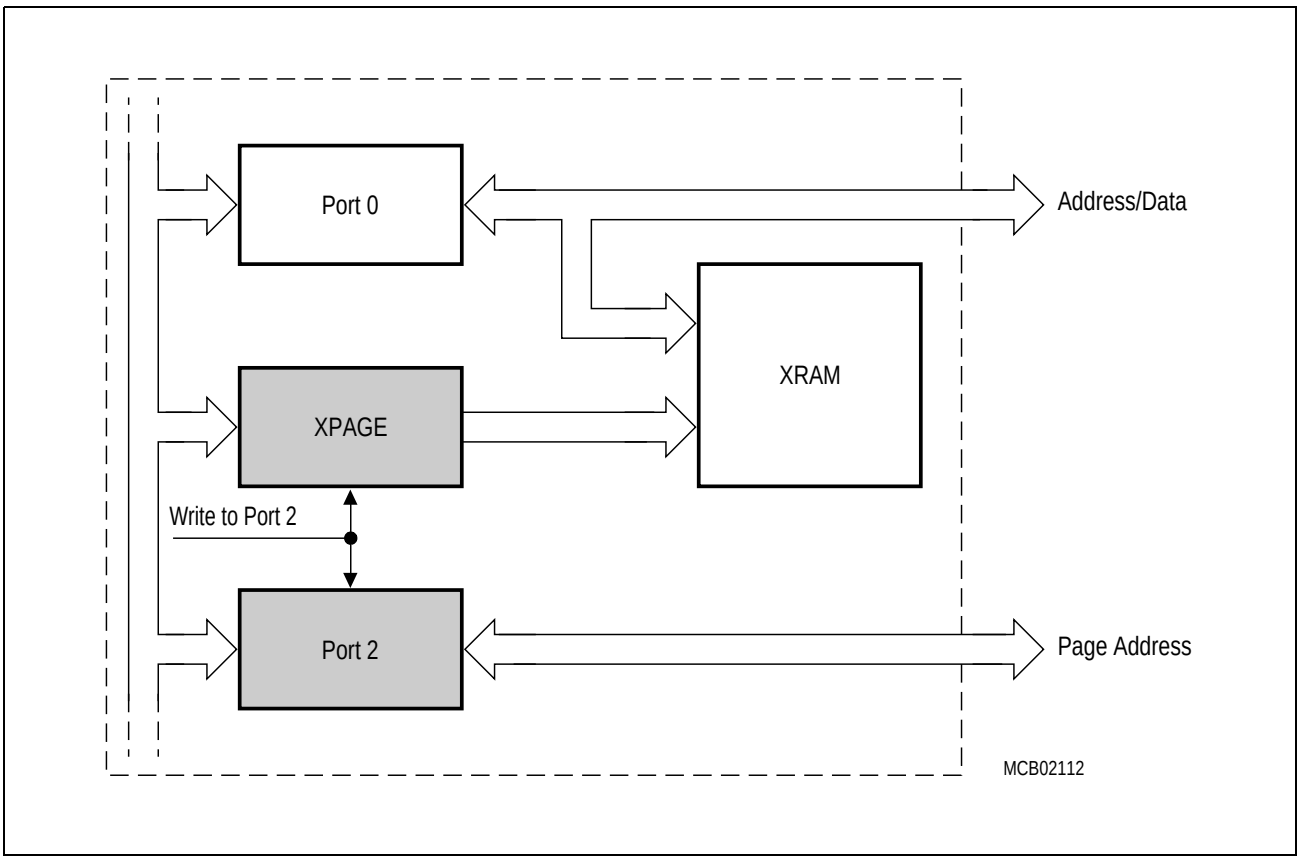


Figure 3-2
Write Page Address to Port 2

“MOV P2,pageaddress“ will write the page address to Port 2 and the XPAGE-Register.

When external RAM is to be accessed in the XRAM address range ($F800_H$ - $FFFF_H$), XRAM has to be disabled. When additional external RAM is to be addressed in an address range < $F800_H$, XRAM may remain enabled and there is no need to overwrite XPAGE by a second move.

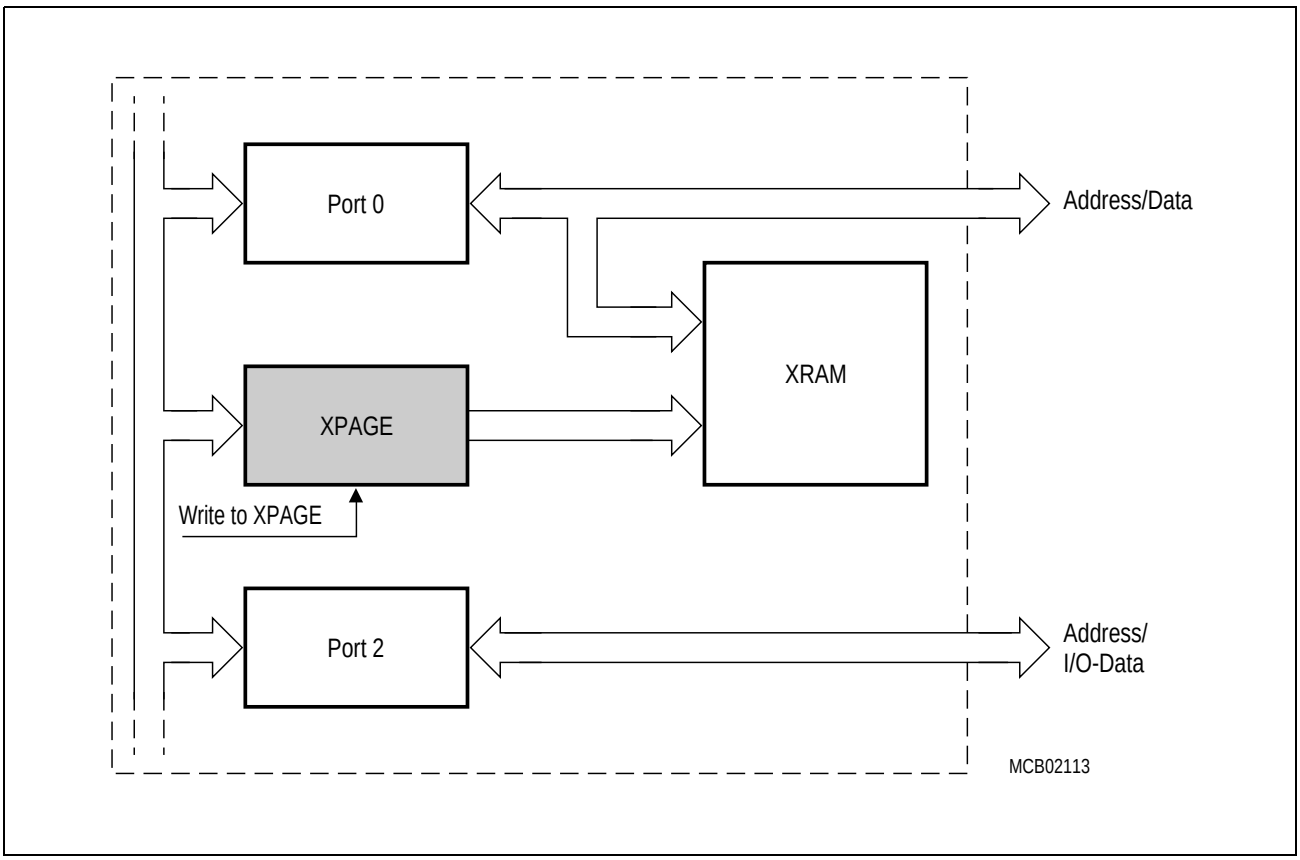


Figure 3-3
Write Page Address to XPAGE

“MOV XPAGE,pageaddress“ will write the page address only to the XPAGE register. Port 2 is available for addresses or I/O data.

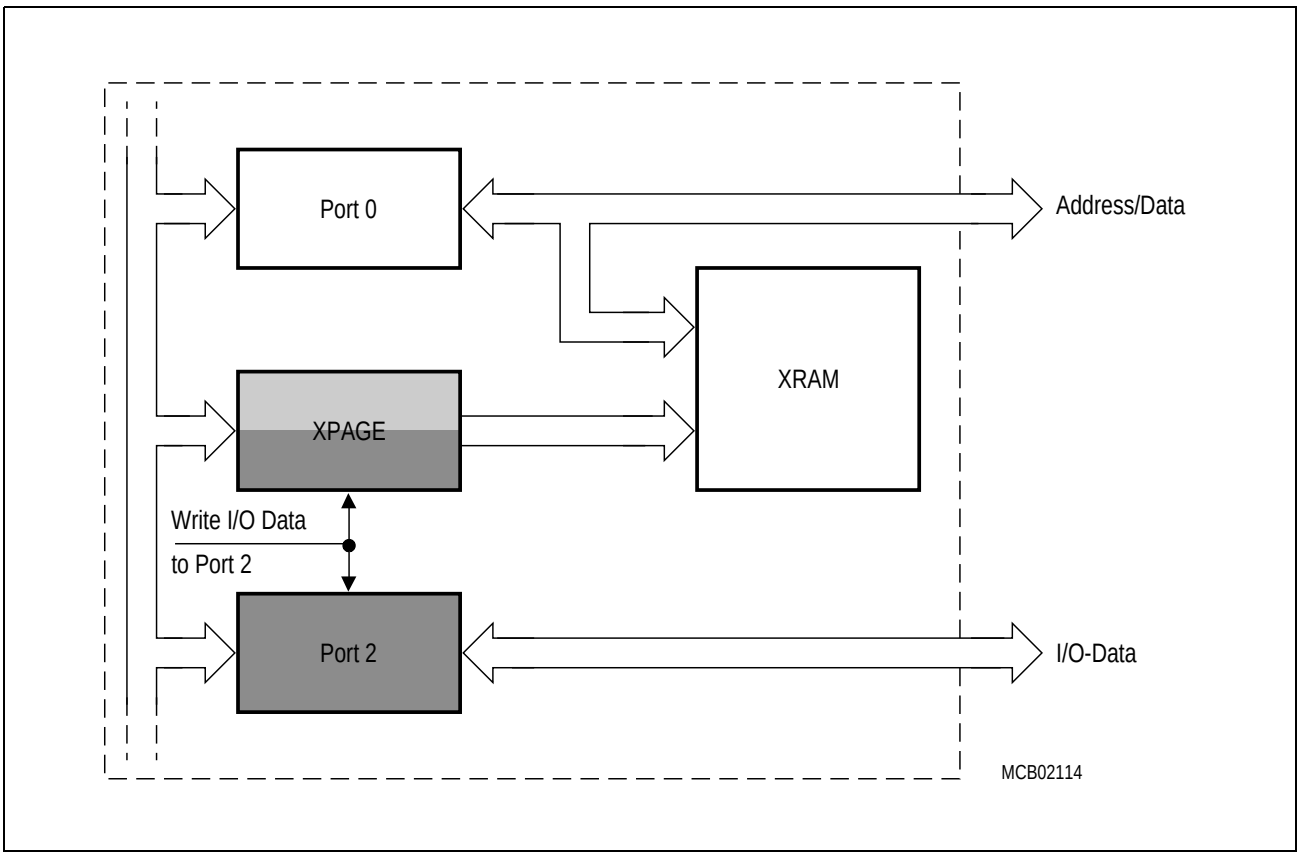


Figure 3-4
Usage of Port 2 as I/O Port

At a write to port 2, the XRAM address in the XPAGE register will be overwritten because of the concurrent write to port 2 and XPAGE register. So, whenever XRAM is used and the XRAM address differs from the byte written to port 2 latch it is absolutely necessary to rewrite XPAGE with the page address.

Example :

I/O data at port 2 shall be AA_H. A byte shall be fetched from XRAM at address $F830_H$.

```
MOV     R0, #30H          ;
MOV     P2, #0AAH         ; P2 shows AAH
MOV     XPAGE, #0F8H      ; P2 still shows AAH but XRAM is addressed
MOVX    A, @R0            ; the contents of XRAM at F830H is moved to accumulator
```

The register XPAGE provides the upper address byte for accesses to XRAM with MOVX @Ri instructions. If the address formed by XPAGE and Ri points outside the XRAM address range, an external access is performed. For the C517A the content of XPAGE must be $F8_H$ - FF_H in order to use the XRAM.

The software has to distinguish two cases, if the MOVX @Ri instructions with paging shall be used :

a) Access to XRAM : The upper address byte must be written to XPAGE or P2; both writes select the XRAM address range.

b) Access to external memory : The upper address byte must be written to P2; XPAGE will be loaded with the same address in order to deselect the XRAM.

3.4.4 Reset Operation of the XRAM

The contents of the XRAM are not affected by a reset. After power-up the contents are undefined, while they remain unchanged during and after a reset as long as the power supply is not turned off. If a reset occurs during a write operation to XRAM, the content of a XRAM memory location depends on the cycle in which the active reset signal is detected (MOVX is a 2-cycle instruction):

Reset during 1st cycle : The new value will not be written to XRAM. The old value is not affected.
Reset during 2nd cycle : The old value in XRAM is overwritten by the new value.

3.4.5 Behaviour of Port0 and Port2

The behaviour of Port 0 and P2 during a MOVX access depends on the control bits in register SYSCON and on the state of pin $\overline{EA}$. The **table 3-7** lists the various operating conditions. It shows the following characteristics :

a) Use of P0 and P2 pins during the MOVX access.
   Bus: The pins work as external address/data bus. If (internal) XRAM is accessed, the data written to the XRAM can be seen on the bus in debug mode.
   I/0: The pins work as Input/Output lines under control of their latch.
b) Activation of the $\overline{RD}$ and $\overline{WR}$ pin during the access.
c) Use of internal or external XDATA memory.

The shaded areas describe the standard operation as each 80C51 device without on-chip XRAM behaves.

		$\overline{EA}$ = 0			$\overline{EA}$ = 1		
		XMAP1, XMAP0			**XMAP1, XMAP0**		
		00	**10**	**X1**	**00**	**10**	**X1**
MOVX @DPTR	DPTR < XRAM address range	a)P0/P2→Bus b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0/P2→Bus b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0/P2→Bus b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0/P2→Bus b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0/P2→Bus b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0/P2→Bus b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used
	DPTR ≥ XRAM address range	a)P0/P2→Bus ($\overline{RD}$/$\overline{WR}$-Data) b)$\overline{RD}$/$\overline{WR}$ inactive c)XRAM is used	a)P0/P2→Bus ($\overline{RD}$/$\overline{WR}$-Data) b)$\overline{RD}$/$\overline{WR}$ active c)XRAM is used	a)P0/P2→Bus b)$\overline{RD}$/$\overline{WR}$ active c) ext.memory is used	a)P0/P2→I/O b)$\overline{RD}$/$\overline{WR}$ inactive c)XRAM is used	a)P0/P2→Bus ($\overline{RD}$/$\overline{WR}$-Data) b)$\overline{RD}$/$\overline{WR}$ active c)XRAM is used	a)P0/P2→Bus b)$\overline{RD}$/$\overline{WR}$ active c) ext.memory is used
MOVX @ Ri	XPAGE < XRAM addr.page range	a)P0→Bus P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0→Bus P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0→Bus P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0→Bus P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0→Bus P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0→Bus P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used
	XPAGE ≥ XRAM addr.page range	a)P0→Bus ($\overline{RD}$/$\overline{WR}$-Data) P2→I/O b)$\overline{RD}$/$\overline{WR}$ inactive c)XRAM is used	a)P0→Bus ($\overline{RD}$/$\overline{WR}$-Data) P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)XRAM is used	a)P0→Bus P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used	a)P0/P2→I/O b)$\overline{RD}$/$\overline{WR}$ inactive c)XRAM is used	a)P0→Bus ($\overline{RD}$/$\overline{WR}$-Data) P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)XRAM is used	a)P0→Bus P2→I/O b)$\overline{RD}$/$\overline{WR}$ active c)ext.memory is used

[shaded box] modes compatible to 8051/C501 family

Table 3-7
Behaviour of P0/P2 and $\overline{RD}$/$\overline{WR}$ During MOVX Accesses

3.5 Special Function Registers

The registers, except the program counter and the four general purpose register banks, reside in the special function register area. All SFRs with addresses where address bits 0-2 are 0 (e.g. 80_H, 88_H, 90_H, 98_H, ..., $F8_H$, FF_H) are bitaddressable.

The 93 special function registers (SFRs) in the SFR area include pointers and registers that provide an interface between the CPU and the other on-chip peripherals. The SFRs of the C517A are listed in **table 3-1** and **table 3-2**. In **table 3-1** they are organized in groups which refer to the functional blocks of the C517A. **Table 3-2** illustrates the contents of the SFRs in numeric order of their addresses..

Table 3-1
Special Function Registers - Functional Blocks

Block	Symbol	Name	Address	Contents after Reset
CPU	ACC	Accumulator	**$E0_H$** 1)	00_H
	B	B-Register	**$F0_H$** 1)	00_H
	DPH	Data Pointer, High Byte	83_H	00_H
	DPL	Data Pointer, Low Byte	82_H	00_H
	DPSEL	Data Pointer Select Register	92_H	$XXXX\ X000_B$ 3)
	PSW	Program Status Word Register	**$D0_H$** 1)	00_H
	SP	Stack Pointer	81_H	07_H
A/D-Converter	ADCON0 2)	A/D Converter Control Register 0	**$D8_H$** 1)	00_H
	ADCON1	A/D Converter Control Register 1	DC_H	$0XXX\ 0000_B$ 3)
	ADDATH	A/D Converter Data Register, High Byte	$D9_H$	00_H
	ADDATL	A/D Converter Data Register, Low Byte	DA_H	$00XX\ XXXX_B$ 3
Interrupt System	IEN0 2)	Interrupt Enable Register 0	**$A8_H$** 1)	00_H
	IEN1 2)	Interrupt Enable Register 1	**$B8_H$** 1)	00_H
	IEN2	Interrupt Enable Register 2	$9A_H$	$XX00\ 00X0_B$ 3)
	IP0 2)	Interrupt Priority Register 0	$A9_H$	00_H
	IP1	Interrupt Priority Register 1	$B9_H$	$XX00\ 0000_B$ 3)
	IRCON0 2)	Interrupt Request Control Register 0	**$C0_H$** 1)	00_H
	IRCON1	Interrupt Request Control Register 1	$D1_H$	00_H
	TCON 2)	Timer 0/1 Control Register	**88_H** 1)	00_H
	T2CON 2)	Timer 2 Control Register	**$C8_H$** 1)	00_H
	S0CON 2)	Serial Channel 0 Control Register	**98_H** 1)	00_H
	S1CON 2)	Serial Channel ! Control Register	$9B_H$	$0X00\ 0000_B$ 3)
	CTCON 2)	Compare Timer Control Register	$E1_H$	$0X00\ 0000_B$ 3)
MUL/DIV Unit	ARCON	Arithmetic Control Register	EF_H	$0XXXXXXX_B$ 3)
	MD0	Multiplication/Division Register 0	$E9_H$	XX_H 3)
	MD1	Multiplication/Division Register 1	EA_H	XX_H 3)
	MD2	Multiplication/Division Register 2	EB_H	XX_H 3)
	MD3	Multiplication/Division Register 3	EC_H	XX_H 3)
	MD4	Multiplication/Division Register 4	ED_H	XX_H 3)
	MD5	Multiplication/Division Register 5	EE_H	XX_H 3)
Timer 0 / Timer 1	TCON 2)	Timer 0/1 Control Register	**88_H** 1)	00_H
	TH0	Timer 0, High Byte	$8C_H$	00_H
	TH1	Timer 1, High Byte	$8D_H$	00_H
	TL0	Timer 0, Low Byte	$8A_H$	00_H
	TL1	Timer 1, Low Byte	$8B_H$	00_H
	TMOD	Timer Mode Register	89_H	00_H

1) Bit-addressable special function registers
2) This special function register is listed repeatedly since some bits of it also belong to other functional blocks.
3) “X“ means that the value is undefined and the location is reserved

Table 3-1
Special Function Registers - Functional Blocks (cont'd)

Block	Symbol	Name	Address	Contents after Reset
Compare/ Capture Unit (CCU) Timer 2	CCEN	Compare/Capture Enable Register	$C1_H$	00_H
	CC4EN	Compare/Capture 4 Enable Register	$C9_H$	00_H
	CCH1	Compare/Capture Register 1, High Byte	$C3_H$	00_H
	CCH2	Compare/Capture Register 2, High Byte	$C5_H$	00_H
	CCH3	Compare/Capture Register 3, High Byte	$C7_H$	00_H
	CCH4	Compare/Capture Register 4, High Byte	CF_H	00_H
	CCL1	Compare/Capture Register 1, Low Byte	$C2_H$	00_H
	CCL2	Compare/Capture Register 2, Low Byte	$C4_H$	00_H
	CCL3	Compare/Capture Register 3, Low Byte	$C6_H$	00_H
	CCL4	Compare/Capture Register 4, Low Byte	CE_H	00_H
	CMEN	Compare Enable Register	$F6_H$	00_H
	CMH0	Compare Register 0, High Byte	$D3_H$	00_H
	CMH1	Compare Register 1, High Byte	$D5_H$	00_H
	CMH2	Compare Register 2, High Byte	$D7_H$	00_H
	CMH3	Compare Register 3, High Byte	$E3_H$	00_H
	CMH4	Compare Register 4, High Byte	$E5_H$	00_H
	CMH5	Compare Register 5, High Byte	$E7_H$	00_H
	CMH6	Compare Register 6, High Byte	$F3_H$	00_H
	CMH7	Compare Register 7, High Byte	$F5_H$	00_H
	CML0	Compare Register 0, Low Byte	$D2_H$	00_H
	CML1	Compare Register 1, Low Byte	$D4_H$	00_H
	CML2	Compare Register 2, Low Byte	$D6_H$	00_H
	CML3	Compare Register 3, Low Byte	$E2_H$	00_H
	CML4	Compare Register 4, Low Byte	$E4_H$	00_H
	CML5	Compare Register 5, Low Byte	$E6_H$	00_H
	CML6	Compare Register 6, Low Byte	$F2_H$	00_H
	CML7	Compare Register 7, Low Byte	$F4_H$	00_H
	CMSEL	Compare Input Select	$F7_H$	00_H
	CRCH	Comp./Rel./Capt. Register High Byte	CB_H	00_H
	CRCL	Comp./Rel./Capt. Register Low Byte	CA_H	00_H
	COMSETL	Compare Set Register Low Byte	$A1_H$	00_H
	COMSETH	Compare Set Register, High Byte	$A2_H$	00_H
	COMCLRL	Compare Clear Register, Low Byte	$A3_H$	00_H
	COMCLRH	Compare Clear Register, High Byte	$A4_H$	00_H
	SETMSK	Compare Set Mask Register	$A5_H$	00_H
	CLRMSK	Compare Clear Mask Register	$A6_H$	00_H
	CTCON [2]	Compare Timer Control Register	$E1_H$	$0X00\ 0000_B$ [3]
	CTRELH	Compare Timer Rel. Register, High Byte	DF_H	00_H
	CTRELL	Compare Timer Rel. Register, Low Byte	DE_H	00_H
	TH2	Timer 2, High Byte	CD_H	00_H
	TL2	Timer 2, Low Byte	CC_H	00_H
	T2CON [2]	Timer 2 Control Register	**$C8_H$** [1]	00_H
	IRCON0 [2]	Interrupt Request Control Register 0	**$C0_H$** [1]	00_H

1) Bit-addressable special function registers
2) This special function register is listed repeatedly since some bits of it also belong to other functional blocks.
3) "X" means that the value is undefined and the location is reserved

Table 3-1
Special Function Registers - Functional Blocks (cont'd)

Block	Symbol	Name	Address	Contents after Reset
Ports	P0	Port 0	**80_H** 1)	FF_H
	P1	Port 1	**90_H** 1)	FF_H
	P2	Port 2	**$A0_H$** 1)	FF_H
	P3	Port 3	**$B0_H$** 1)	FF_H
	P4	Port 4	**$E8_H$** 1)	FF_H
	P5	Port 5	**$F8_H$** 1)	FF_H
	P6	Port 6	FA_H	FF_H
	P7	Port 7, Analog/Digital Input	DB_H	–
	P8	Port 8, Analog/Digital Input, 4-bit	DD_H	–
XRAM	XPAGE	Page Address Register for Extended On-Chip RAM	91_H	00_H
	SYSCON 2)	System/XRAM Control Register	$B1_H$	XXXX $XX01_B$ 3)
Serial Channels	ADCON0 2)	A/D Converter Control Register	**$D8_H$** 1)	**00_H**
	PCON 2)	Power Control Register	87_H	00_H
	S0BUF	Serial Channel 0 Buffer Register	99_H	XX_H 3)
	S0CON 2)	Serial Channel 0 Control Register	**98_H** 1)	**00_H**
	S0RELL	Serial Channel 0 Reload Reg., Low Byte	AA_H	$D9_H$
	S0RELH	Serial Channel 0 Reload Reg., High Byte	BA_H	XXXX $XX11_B$ 3)
	S1BUF	Serial Channel 1 Buffer Register	$9C_H$	XX_H 3)
	S1CON 2)	Serial Channel 1 Control Register	$9B_H$	0X00 0000_B 3)
	S1RELL	Serial Channel 1 Reload Reg., Low Byte	$9D_H$	00_H
	S1RELH	Serial Channel 1 Reload Reg., High Byte	BB_H	XXXX $XX11_B$ 3)
Watchdog	IEN0 2)	Interrupt Enable Register 0	**$A8_H$** 1)	00_H
	IEN1 2)	Interrupt Enable Register 1	**$B8_H$** 1)	00_H
	IP0 2)	Interrupt Priority Register 0	$A9_H$	00_H
	WDTREL	Watchdog Timer Reload Register	86_H	00_H
Pow. Sav. Modes	PCON 2)	Power Control Register	87_H	00_H

1) Bit-addressable special function registers
2) This special function register is listed repeatedly since some bits of it also belong to other functional blocks.
3) "X" means that the value is undefined and the location is reserved.

Table 3-2
Contents of the SFRs, SFRs in numeric order of their addresses

Addr	Register	Content after Reset[1)]	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
80_H [2)]	P0	FF_H	.7	.6	.5	.4	.3	.2	.1	.0
81_H	SP	07_H	.7	.6	.5	.4	.3	.2	.1	.0
82_H	DPL	00_H	.7	.6	.5	.4	.3	.2	.1	.0
83_H	DPH	00_H	.7	.6	.5	.4	.3	.2	.1	.0
83_H	WDTREL	00_H	WDT-PSEL	.6	.5	.4	.3	.2	.1	.0
87_H	PCON	00_H	SMOD	PDS	IDLS	SD	GF1	GF0	PDE	IDLE
88_H [2)]	TCON	00_H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
89_H	TMOD	00_H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
$8A_H$	TL0	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$8B_H$	TL1	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$8C_H$	TH0	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$8D_H$	TH1	00_H	.7	.6	.5	.4	.3	.2	.1	.0
90_H [2)]	P1	FF_H	T2	CLK-OUT	T2EX	$\overline{INT2}$	INT6	INT5	INT4	$\overline{INT3}$
91_H	XPAGE	00_H	.7	.6	.5	.4	.3	.2	.1	.0
92_H	DPSEL	$XXXX-X000_B$	–	–	–	–	–	.2	.1	.0
98_H [2)]	S0CON	00_H	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0
99_H	S0BUF	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$9A_H$	IEN2	$XX00-00X0_B$	–	–	ECR	ECS	ECT	ECMP	–	ES1
$9B_H$	S1CON	$0X00-0000_B$	SM	–	SM21	REN1	TB81	RB81	TI1	RI1
$9C_H$	S1BUF	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$9D_H$	S1RELL	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$A0_H$ [2)]	P2	FF_H	.7	.6	.5	.4	.3	.2	.1	.0
$A1_H$	COMSETL	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$A2_H$	COMSETH	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$A3_H$	COMCLRL	00_H	.7	.6	.5	.4	.3	.2	.1	.0

1) X means that the value is undefined and the location is reserved
2) Shaded registers are bit-addressable special function registers

Table 3-2
Contents of the SFRs, SFRs in numeric order of their addresses (cont'd)

Addr	Register	Content after Reset[1)]	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$A4_H$	COMCLRH	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$A5_H$	SETMSK	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$A6_H$	CLRMSK	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$A8_H$[2)]	IEN0	00_H	EAL	WDT	ET2	ES0	ET1	EX1	ET0	EX0
$A9_H$	IP0	00_H	OWDS	WDTS	.5	.4	.3	.2	.1	.0
AA_H	S0RELL	$D9_H$	.7	.6	.5	.4	.3	.2	.1	.0
$B0_H$[2)]	P3	FF_H	RD	WR	T1	T0	INT1	INT0	TxD0	RxD0
$B1_H$	SYSCON	XXXX-$XX01_B$	–	–	–	–	–	–	XMAP1	XMAP0
$B8_H$[2)]	IEN1	00_H	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC
$B9_H$	IP1	XX00-0000_B	–	–	.5	.4	.3	.2	.1	.0
BA_H	S0RELH	XXXX-$XX11_B$	–	–	–	–	–	–	.1	.0
BB_H	S1RELH	XXXX-$XX11_B$	–	–	–	–	–	–	.1	.0
$C0_H$[2)]	IRCON0	00_H	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC
$C1_H$	CCEN	00_H	COCAH3	COCAL3	COCAH2	COCAL2	COCAH1	COCAL1	COCAH0	COCAL0
$C2_H$	CCL1	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$C3_H$	CCH1	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$C4_H$	CCL2	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$C5_H$	CCH2	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$C6_H$	CCL3	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$C7_H$	CCH3	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$C8_H$[2)]	T2CON	00_H	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0
$C9_H$	CC4EN	00_H	COCOEN1	COCON2	COCON1	COCON0	COCOEN0	COCAH4	COCAL4	COMO

1) X means that the value is undefined and the location is reserved
2) Shaded registers are bit-addressable special function registers

Table 3-2
Contents of the SFRs, SFRs in numeric order of their addresses (cont'd)

Addr	Register	Content after Reset[1)]	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CA_H	CRCL	00_H	.7	.6	.5	.4	.3	.2	.1	.0
CB_H	CRCH	00_H	.7	.6	.5	.4	.3	.2	.1	.0
CC_H	TL2	00_H	.7	.6	.5	.4	.3	.2	.1	.0
CD_H	TH2	00_H	.7	.6	.5	.4	.3	.2	.1	.0
CE_H	CCL4	00_H	.7	.6	.5	.4	.3	.2	.1	.0
CF_H	CCH4	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$D0_H$ [2)]	PSW	00_H	CY	AC	F0	RS1	RS0	OV	F1	P
$D1_H$	IRCON1	00_H	ICMP7	ICMP6	ICMP5	ICMP4	ICMP3	ICMP2	ICMP1	ICMP0
$D2_H$	CML0	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$D3_H$	CMH0	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$D4_H$	CML1	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$D5_H$	CMH1	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$D6_H$	CML2	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$D7_H$	CMH2	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$D8_H$ [2)]	ADCON0	00_H	BD	CLK	ADEX	BSY	ADM	MX2	MX1	MX0
$D9_H$	ADDATH	00_H	.9	.8	.7	.6	.5	.4	.3	.2
DA_H	ADDATL	$00XX\text{-}XXXX_B$	.1	.0	–	–	–	–	–	–
DB_H	P7	–	.7	.6	.5	.4	.3	.2	.1	.0
DC_H	ADCON1	$0XXX\text{-}0000_B$	ADCL	–	–	–	MX3	MX2	MX1	MX0
DD_H	P8	–	–	–	–	–	.3	.2	.1	.0
DE_H	CTRELL	00_H	.7	.6	.5	.4	.3	.2	.1	.0
DF_H	CTRELH	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E0_H$[2)]	ACC	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E1_H$	CTCON	$0X00.0000_B$	T2PS1	–	ICR	ICS	CTF	CLK2	CLK1	CLK0
$E2_H$	CML3	00_H	.7	.6	.5	.4	.3	.2	.1	.0

1) X means that the value is undefined and the location is reserved
2) Shaded registers are bit-addressable special function registers

Table 3-2
Contents of the SFRs, SFRs in numeric order of their addresses (cont'd)

Addr	Register	Content after Reset[1)]	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$E3_H$	CMH3	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E4_H$	CML4	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E5_H$	CMH4	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E6_H$	CML5	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E7_H$	CMH5	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E8_H$[2)]	P4	FF_H	CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0
$E9_H$	MD0	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
EA_H	MD1	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
EB_H	MD2	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
EC_H	MD3	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
ED_H	MD4	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
EE_H	MD5	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
EF_H	ARCON	$0XXX.XXXX_B$	MDEF	MDOV	SLR	SC.4	SC.3	SC.2	SC.1	SC.0
$F0_H$[2)]	B	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$F2_H$	CML6	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$F3_H$	CMH6	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$F4_H$	CML7	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$F5_H$	CMH7	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$F6_H$	CMEN	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$F7_H$	CMSEL	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$F8_H$[2)]	P5	FF_H	CCM7	CCM6	CCM5	CCM4	CCM3	CCM2	CCM1	CCM0
FA_H	P6	FF_H	.7	.6	.5	.4	.3	TxD1	RxD1	ADST

1) X means that the value is undefined and the location is reserved
2) Shaded registers are bit-addressable special function registers

4 External Bus Interface

The C517A allows for external memory expansion. The functionality and implementation of the external bus interface is identical to the common interface for the 8051 architecture.

4.1 Accessing External Memory

It is possible to distinguish between accesses to external program memory and external data memory or other peripheral components respectively. This distinction is made by hardware: accesses to external program memory use the signal $\overline{\text{PSEN}}$ (program store enable) as a read strobe. Accesses to external data memory use $\overline{\text{RD}}$ and $\overline{\text{WR}}$ to strobe the memory (alternate functions of P3.7 and P3.6). Port 0 and port 2 (with exceptions) are used to provide data and address signals. In this section only the port 0 and port 2 functions relevant to external memory accesses are described.

Fetches from external program memory always use a 16-bit address. Accesses to external data memory can use either a 16-bit address (MOVX @DPTR) or an 8-bit address (MOVX @Ri).

4.1.1 Role of P0 and P2 as Data/Address Bus

When used for accessing external memory, port 0 provides the data byte time-multiplexed with the low byte of the address. In this state, port 0 is disconnected from its own port latch, and the address/ data signal drives both FETs in the port 0 output buffers. Thus, in this application, the port 0 pins are not open-drain outputs and do not require external pullup resistors.

During any access to external memory, the CPU writes FF_H to the port 0 latch (the special function register), thus obliterating whatever information the port 0 SFR may have been holding.

Whenever a 16-bit address is used, the high byte of the address comes out on port 2, where it is held for the duration of the read or write cycle. During this time, the port 2 lines are disconnected from the port 2 latch (the special function register).

Thus the port 2 latch does not have to contain 1s, and the contents of the port 2 SFR are not modified.

If an 8-bit address is used (MOVX @Ri), the contents of the port 2 SFR remain at the port 2 pins throughout the external memory cycle. This will facilitate paging. It should be noted that, if a port 2 pin outputs an address bit that is a 1, strong pullups will be used for the entire read/write cycle and not only for two oscillator periods.

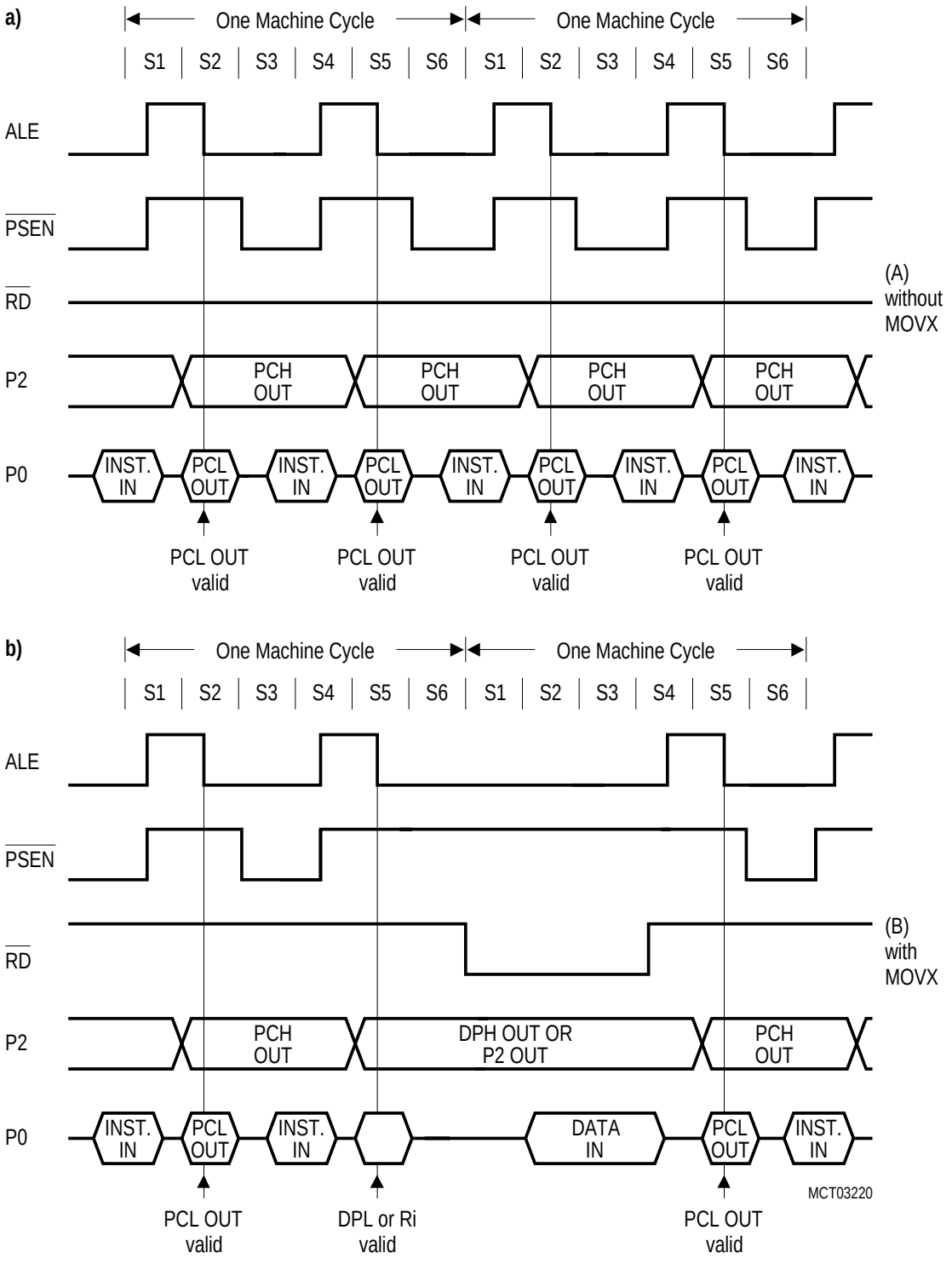


Figure 4-1
External Program Memory Execution

4.1.2 Timing

The timing of the external bus interface, in particular the relationship between the control signals ALE, $\overline{PSEN}$, $\overline{RD}$, $\overline{WR}$ and information on port 0 and port 2, is illustated in **figure 4-1 a)** and **b)**.

Data memory: in a write cycle, the data byte to be written appears on port 0 just before $\overline{WR}$ is activated and remains there until after $\overline{WR}$ is deactivated. In a read cycle, the incoming byte is accepted at port 0 before the read strobe is deactivated.

Program memory: Signal $\overline{PSEN}$ functions as a read strobe.

4.1.3 External Program Memory Access

The external program memory is accessed under two conditions:

- - whenever signal $\overline{EA}$ is active (low); or
- - whenever the program counter (PC) content is greater than $7FFF_H$

When the CPU is executing out of external program memory, all 8 bits of port 2 are dedicated to an output function and must not be used for general-purpose I/O. The content of the port 2 SFR however is not affected. During external program memory fetches port 2 lines output the high byte of the PC, and during accesses to external data memory they output either DPH or the port 2 SFR (depending on whether the external data memory access is a MOVX @DPTR or a MOVX @Ri).

4.2 $\overline{PSEN}$, Program Store Enable

The read strobe for external program memory fetches is $\overline{PSEN}$. It is not activated for internal program memory fetches. When the CPU is accessing external program memory, $\overline{PSEN}$ is activated twice every instruction cycle (except during a MOVX instruction) no matter whether or not the byte fetched is actually needed for the current instruction. When $\overline{PSEN}$ is activated its timing is not the same as for $\overline{RD}$. A complete $\overline{RD}$ cycle, including activation and deactivation of ALE and $\overline{RD}$, takes 6 oscillator periods. A complete $\overline{PSEN}$ cycle, including activation and deactivation of ALE and $\overline{PSEN}$, takes 3 oscillator periods. The execution sequence for these two types of read cycles is shown in **figure 4-1 a)** and **b)**.

4.3 Overlapping External Data and Program Memory Spaces

In some applications it is desirable to execute a program from the same physical memory that is used for storing data. In the C517A the external program and data memory spaces can be combined by the logical-AND of $\overline{PSEN}$ and $\overline{RD}$. A positive result from this AND operation produces a low active read strobe that can be used for the combined physical memory. Since the $\overline{PSEN}$ cycle is faster than the $\overline{RD}$ cycle, the external memory needs to be fast enough to adapt to the $\overline{PSEN}$ cycle.

4.4 Enhanced Hooks Emulation Concept

The Enhanced Hooks Emulation Concept of the C500 microcontroller family is a new, innovative way to control the execution of C500 MCUs and to gain extensive information on the internal operation of the controllers. Emulation of on-chip ROM based programs is possible, too.
Each C500 production chip has built-in logic for the supprt of the Enhanced Hooks Emulation Concept. Therefore, no costly bond-out chips are necessary for emulation. This also ensures that emulation and production chips are identical.

The Enhanced Hooks Technology™ [1], which requires embedded logic in the C500 allows the C500 together with an EH-IC to function similar to a bond-out chip. This simplifies the design and reduces costs of an ICE-system. ICE-systems using an EH-IC and a compatible C500 are able to emulate all operating modes of the different versions of the C500 microcontrollers. This includes emulation of ROM, ROM with code rollover and ROMless modes of operation. It is also able to operate in single step mode and to read the SFRs after a break.

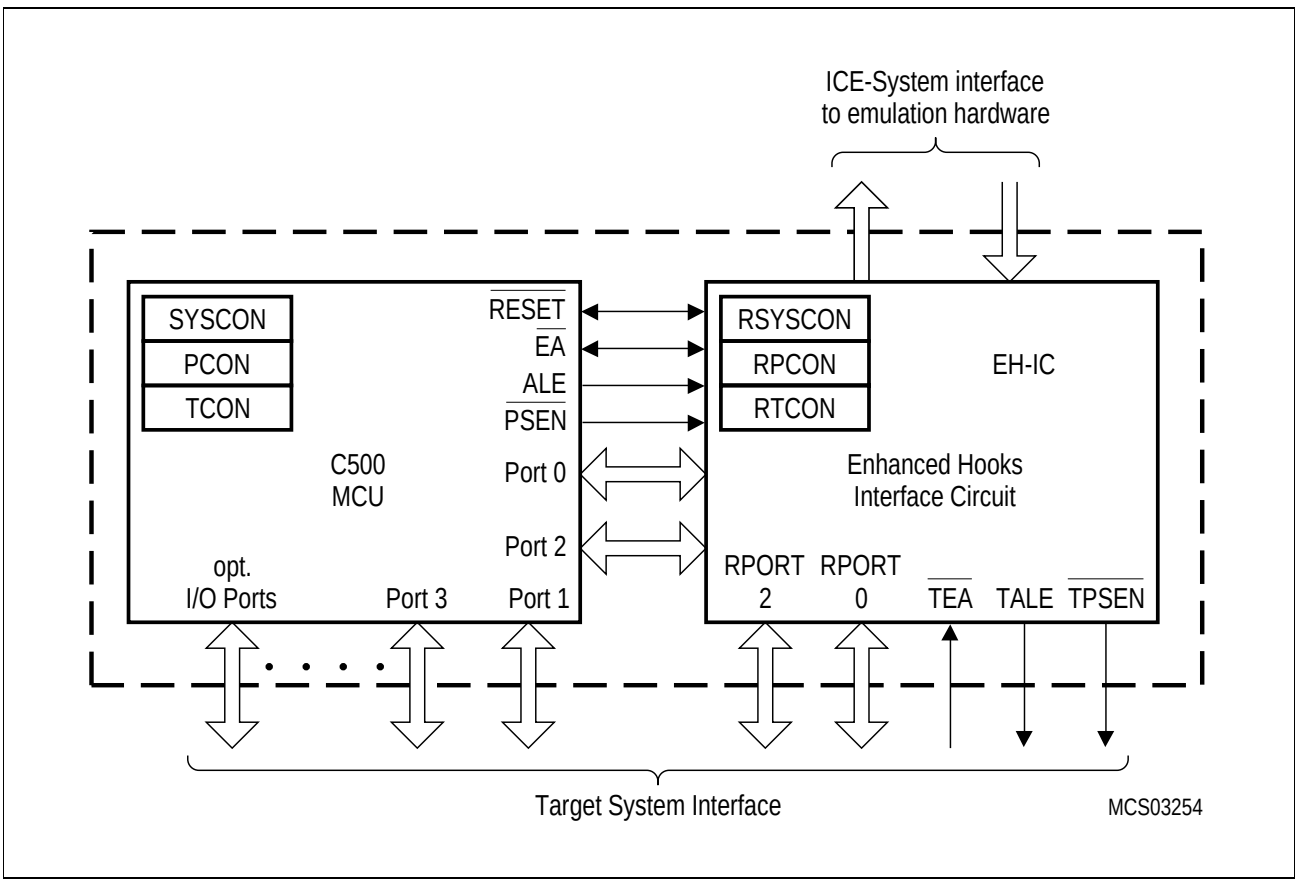


Figure 4-2
Basic C500 MCU Enhanced Hooks Concept Configuration

Port 0, port 2 and some of the control lines of the C500 based MCU are used by Enhanced Hooks Emulation Concept to control the operation of the device during emulation and to transfer informations about the programm execution and data transfer between the external emulation hardware (ICE-system) and the C500 MCU.

1 “Enhanced Hooks Technology“ is a trademark and patent of Metalink Corporation licensed to Siemens.

4.5 Eight Datapointers for Faster External Bus Access

4.5.1 The Importance of Additional Datapointers

The standard 8051 architecture provides just one 16-bit pointer for indirect addressing of external devices (memories, peripherals, latches, etc.). Except for a 16-bit "move immediate" to this datapointer and an increment instruction, any other pointer handling is to be handled bytewise. For complex applications with peripherals located in the external data memory space (e.g. CAN controller) or extended data storage capacity this turned out to be a "bottle neck" for the 8051's communication to the external world. Especially programming in high-level languages (PLM51, C51, PASCAL51) requires extended RAM capacity and at the same time a fast access to this additional RAM because of the reduced code efficiency of these languages.

4.5.2 How the eight Datapointers of the C517A are realized

Simply adding more datapointers is not suitable because of the need to keep up 100% compatibility to the 8051/C501 instruction set. This instruction set, however, allows the handling of only one single 16-bit datapointer (DPTR, consisting of the two 8-bit SFRs DPH and DPL).

To meet both of the above requirements (speed up external accesses, 100% compatibility to 8051 architecture) the C517A contains a set of eight 16-bit registers from which the actual datapointer can be selected.

This means that the user's program may keep up to eight 16-bit addresses resident in these registers, but only one register at a time is selected to be the datapointer. Thus the datapointer in turn is accessed (or selected) via indirect addressing. This indirect addressing is done through a special function register called DPSEL (data pointer select register). All instructions of the C517A which handle the datapointer therefore affect only one of the eight pointers which is addressed by DPSEL at that very moment.

Figure 4-3 illustrates the addressing mechanism: a 3-bit field in register DPSEL points to the currently used DPTRx. Any standard 8051 instruction (e.g. MOVX @DPTR, A - transfer a byte from accumulator to an external location addressed by DPTR) now uses this activated DPTRx.

Special Function Register DPSEL (Address 92_H) **Reset Value : $XXXXX000_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
92_H	–	–	–	–	–	.2	.1	.0	DPSEL

Bit	Function
–	Reserved bits for future use.
DPSEL.2-0	Data pointer select bits DPSEL.2-0 defines the number of the actual active data pointer.DPTR0-7.

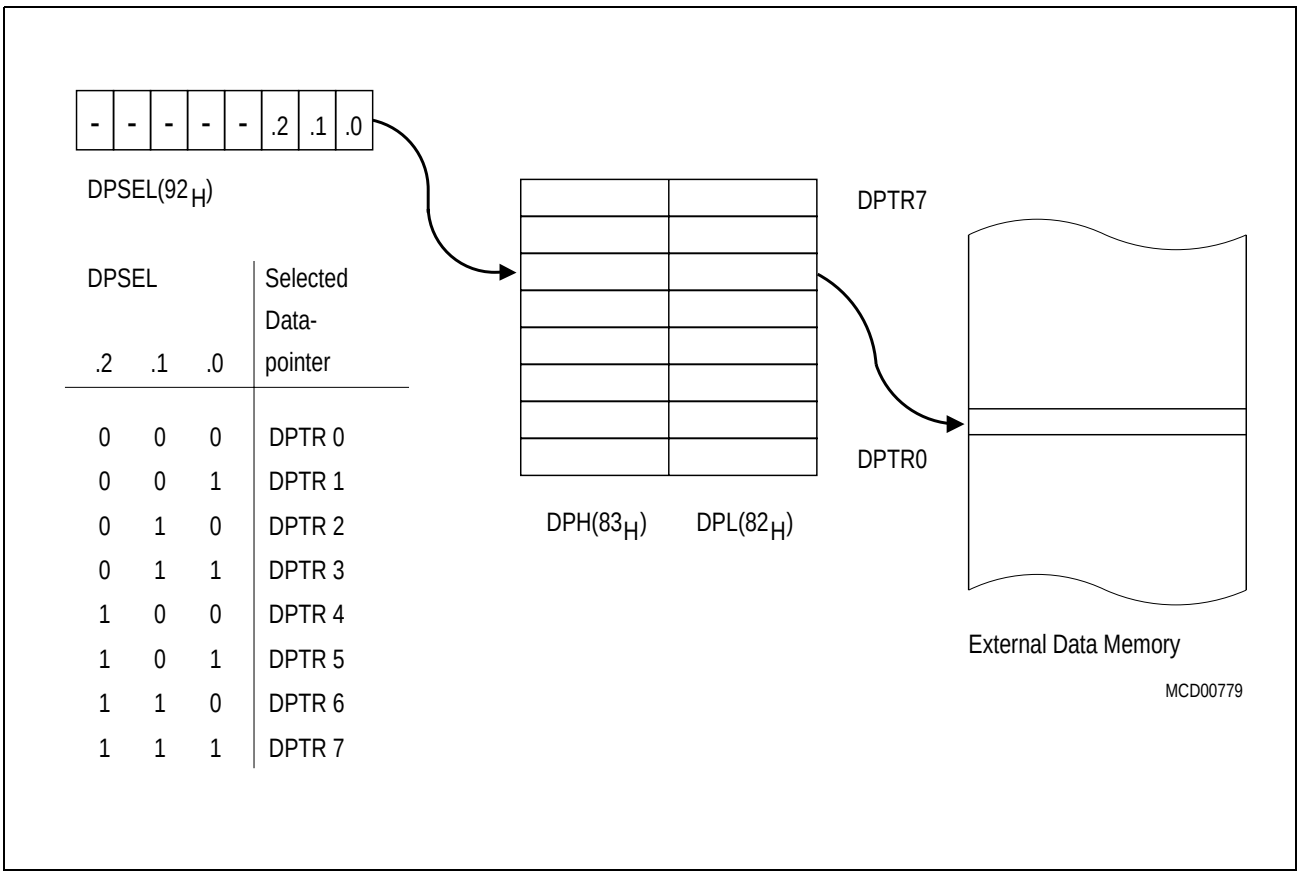


Figure 4-3
Accessing of External Data Memory via Multiple Datapointers

4.5.3 Advantages of Multiple Datapointers

Using the above addressing mechanism for external data memory results in less code and faster execution of external accesses. Whenever the contents of the datapointer must be altered between two or more 16-bit addresses, one single instruction, which selects a new datapointer, does this job. If the program uses just one datapointer, then it has to save the old value (with two 8-bit instructions) and load the new address, byte by byte. This not only takes more time, it also requires additional space in the internal RAM.

4.5.4 Application Example and Performance Analysis

The following example shall demonstrate the involvement of multiple data pointers in a table transfer from the code memory to external data memory.

Start address of ROM source table: $1FFF_H$
Start address of table in external RAM: $2FA0_H$

Example 1 : Using only One Datapointer (Code for a C501)

Initialization Routine

```
MOV     LOW(SRC_PTR), #0FFH     ;Initialize shadow_variables with source_pointer
MOV     HIGH(SRC_PTR), #1FH
MOV     LOW(DES_PTR), #0A0H     ;Initialize shadow_variables with destination_pointer
MOV     HIGH(DES_PTR), #2FH
```

Table Look-up Routine under Real Time Conditions

```
                                ;                                       Number of cycles
PUSH    DPL                     ;Save old datapointer                   2
PUSH    DPH                     ;                                       2
MOV     DPL, LOW(SRC_PTR)       ;Load Source Pointer                    2
MOV     DPH, HIGH(SRC_PTR)      ;                                       2
;INC    DPTR                    Increment and check for end of table (execution time
;CJNE   …                       not relevant for this consideration)    –
MOVC    A,@DPTR                 ;Fetch source data byte from ROM table  2
MOV     LOW(SRC_PTR), DPL       ;Save source_pointer and                2
MOV     HIGH(SRC_PTR), DPH      ;load destination_pointer               2
MOV     DPL, LOW(DES_PTR)       ;                                       2
MOV     DPH, HIGH(DES_PTR)      ;                                       2
INC     DPTR                    ;Increment destination_pointer
                                ;(ex. time not relevant)                –
MOVX    @DPTR, A                ;Transfer byte to destination address   2
MOV     LOW(DES_PTR), DPL       ;Save destination_pointer               2
MOV     HIGH(DES_PTR),DPH       ;                                       2
POP     DPH                     ;Restore old datapointer                2
POP     DPL                     ;                                       2

;                               Total execution time (machine cycles) :  28
```

Example 2 : Using Two Datapointers (Code for an C517A)

Initialization Routine

```
MOV     DPSEL, #06H             ;Initialize DPTR6 with source pointer
MOV     DPTR, #1FFFH
MOV     DPSEL, #07H             ;Initialize DPTR7 with destination pointer
MOV     DPTR, #2FA0H
```

Table Look-up Routine under Real Time Conditions

```
                                ;                                   Number of cycles
PUSH    DPSEL                   ;Save old source pointer            2
MOV     DPSEL, #06H             ;Load source pointer                2
;INC    DPTR                    Increment and check for end of table (execution time
;CJNE   …                       not relevant for this consideration)  –
MOVC    A,@DPTR                 ;Fetch source data byte from ROM table  2
MOV     DPSEL, #07H             ;Save source_pointer and
                                ;load destination_pointer           2
MOVX    @DPTR, A                ;Transfer byte to destination address  2
POP     DPSEL                   ;Save destination pointer and
                                ;restore old datapointer            2

;                               Total execution time (machine cycles) :  12
```

The above example shows that utilization of the C517A's multiple datapointers can make external bus accesses two times as fast as with a standard 8051 or 8051 derivative. Here, four data variables in the internal RAM and two additional stack bytes were spared, too. This means for some applications where all eight datapointers are employed that an C517A program has up to 24 byte (16 variables and 8 stack bytes) of the internal RAM free for other use.

4.6 ROM Protection for the C517A

The C517A-4R allows to protect the contents of the internal ROM against unauthorized read out. The type of ROM protection (protected or unprotected) is fixed with the ROM mask. Therefore, the customer of a C517A-4R version has to define whether ROM protection has to be selected or not.

The C517A-4R devices, which operate from internal ROM, are always checked for correct ROM contents during production test. Therefore, unprotected as well as protected ROMs must provide a procedure to verify the ROM contents. In ROM verification mode 1, which is used to verify unprotected ROMs, a ROM address is applied externally to the C517A-4R and the ROM data byte is output at port 0. ROM verification mode 2, which is used to verify ROM protected devices, operates different : ROM addresses are generated internally and the expected data bytes must be applied externally to the device (by the manufacturer or by the customer) and are compared internally with the data bytes from the ROM. After 16 byte verify operations the state of the P3.5 pin shows whether the last 16 bytes have been verified correctly.

This mechanism provides a very high security of ROM protection. Only the owner of the ROM code and the manufacturer who know the contents of the ROM can read out and verify it with less effort.

The behaviour of the move code instruction, when the code is executed from the external ROM, is in such a way that accessing a code byte from a protected on-chip ROM address is not possible. In this case the byte accessed will be invalid.

4.6.1 Unprotected ROM Mode

If the ROM is unprotected, the ROM verification mode 1 as shown in **figure 4-4** is used to read out the contents of the ROM. The AC timing characteristics of the ROM verification mode is shown in the AC specifications in the C517A Data Sheet.

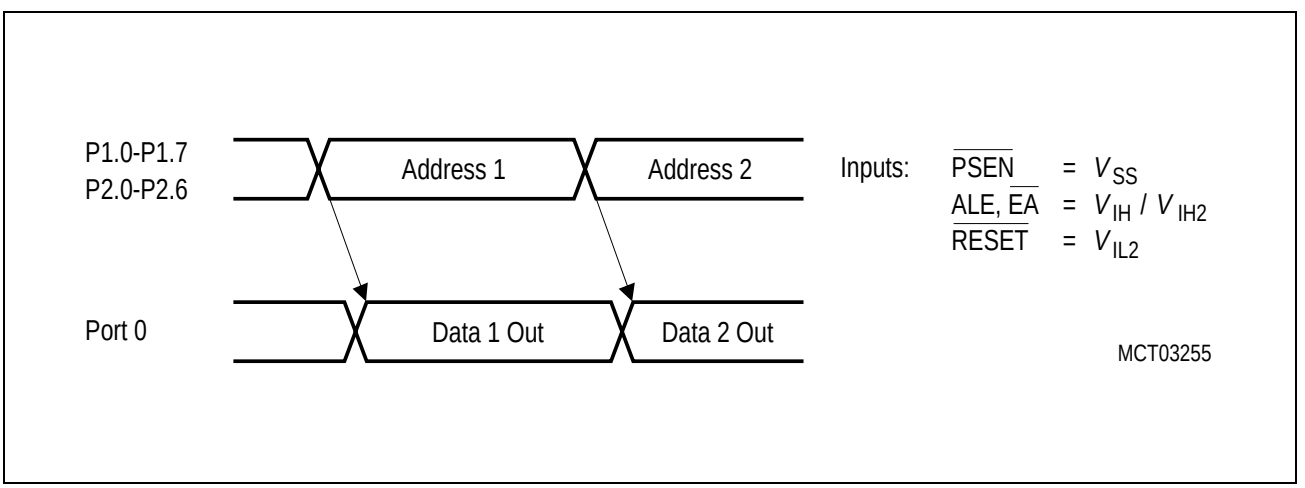


Figure 4-4
ROM Verification Mode 1

ROM verification mode 1 is selected if the inputs $\overline{\text{PSEN}}$, ALE, $\overline{\text{EA}}$, and $\overline{\text{RESET}}$ are put to the specified logic level. Then the 14-bit address of the internal ROM byte to be read is applied to the port 1 and port 2 lines. After a delay time, port 0 outputs the content of the addressed ROM cell. In ROM verification mode 1, the C517A must be provided with a system clock at the XTAL pins and pullup resistors on the port 0 lines.

4.6.2 Protected ROM Mode

If the ROM is protected, the ROM verification mode 2 as shown in **figure 4-5** is used to verify the contents of the ROM. The detailed timing characteristics of the ROM verification mode is shown in the AC specifications in the C517A Data Sheet.

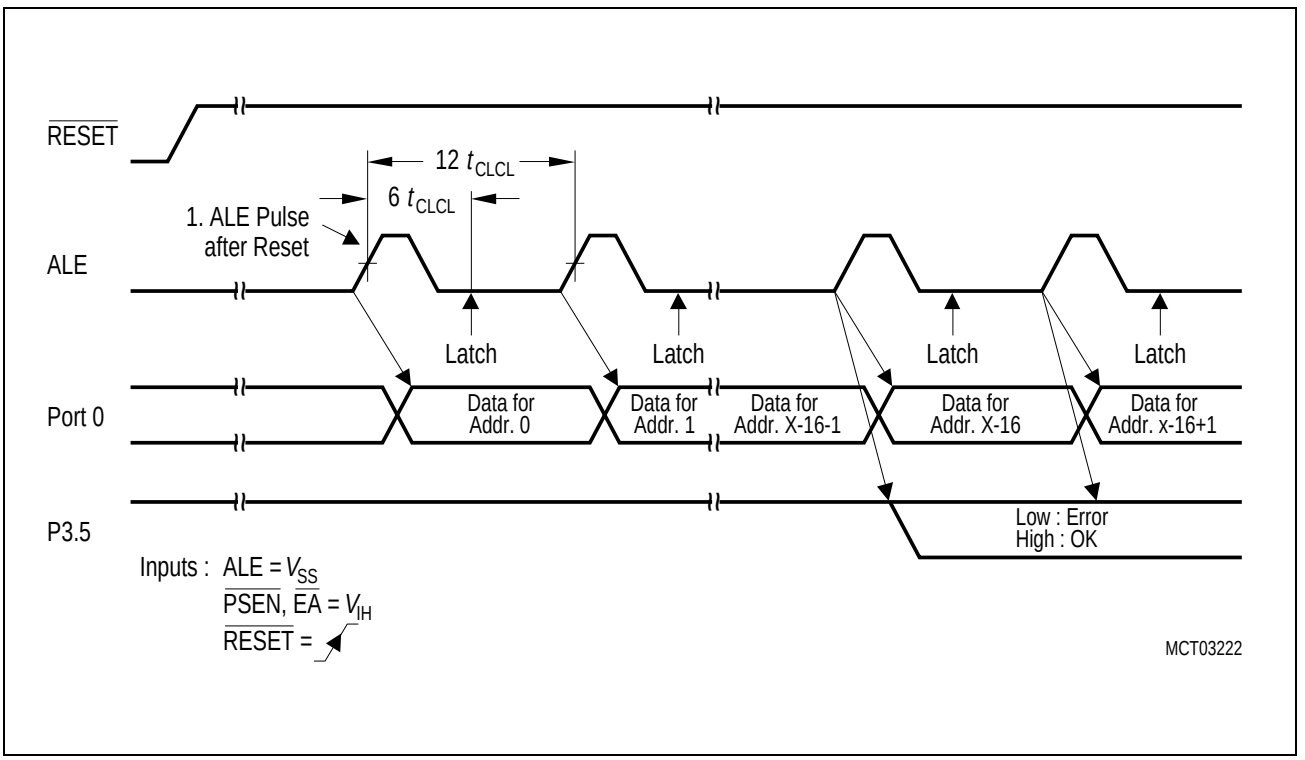


Figure 4-5
ROM Verification Mode 2

ROM verification mode 2 is selected if the inputs $\overline{PSEN}$, $\overline{EA}$, and ALE are put to the specified logic levels. With $\overline{RESET}$ going inactive, the ROM verification mode 2 sequence is started. The C517A outputs an ALE signal with a period of 12 t_{CLCL} and expects data bytes at port 0. The data bytes at port 0 are assigned to the ROM addresses in the following way :

1. Data Byte = content of internal ROM address 0000_H
2. Data Byte = content of internal ROM address 0001_H
3. Data Byte = content of internal ROM address 0002_H
:
16. Data Byte = content of internal ROM address 000FH
:

The C517A-4R does not output any address information during the ROM verification mode 2. The first data byte to be verified is always the byte which is assigned to the internal ROM address 0000_H and must be put onto the data bus with the falling edge of $\overline{RESET}$. With each following ALE pulse the ROM address pointer is internally incremented and the expected data byte for the next ROM address must be delivered externally.
Between two ALE pulses the data at port 0 is latched (at 3 CLP after ALE rising edge) and compared internally with the ROM content of the actual address. If an verify error is detected, the error

condition is stored internally. After each 16th data byte the cumulated verify result (pass or fail) of the last 16 verify operations is output at P3.5. This means that P3.5 stays at static level (low for fail and high for pass) during the 16 bytes are checked. In ROM verification mode 2, the C517A must be provided with a system clock at the XTAL pins.

Figure 4-6 shows an application example of an external circuitry which allows to verify a protected ROM inside the C517A-4R in ROM verification mode 2. With $\overline{\text{RESET}}$ going inactive, the C517A-4R starts the ROM verify sequence. Its ALE is clocking a 16-bit address counter. This counter generates the addresses for an external EPROM which is programmed with the contents of the internal (protected) ROM. The verify detect logic typically displays the pass/fail information of the verify operation. P3.5 can be latched with the falling edge of ALE.

When the last byte of the internal ROM has been handled, the C517A-4R starts generating a $\overline{\text{PSEN}}$ signal. This signal or the CY signal of the address counter indicate to the verify detect logic the end of the internal ROM verification.

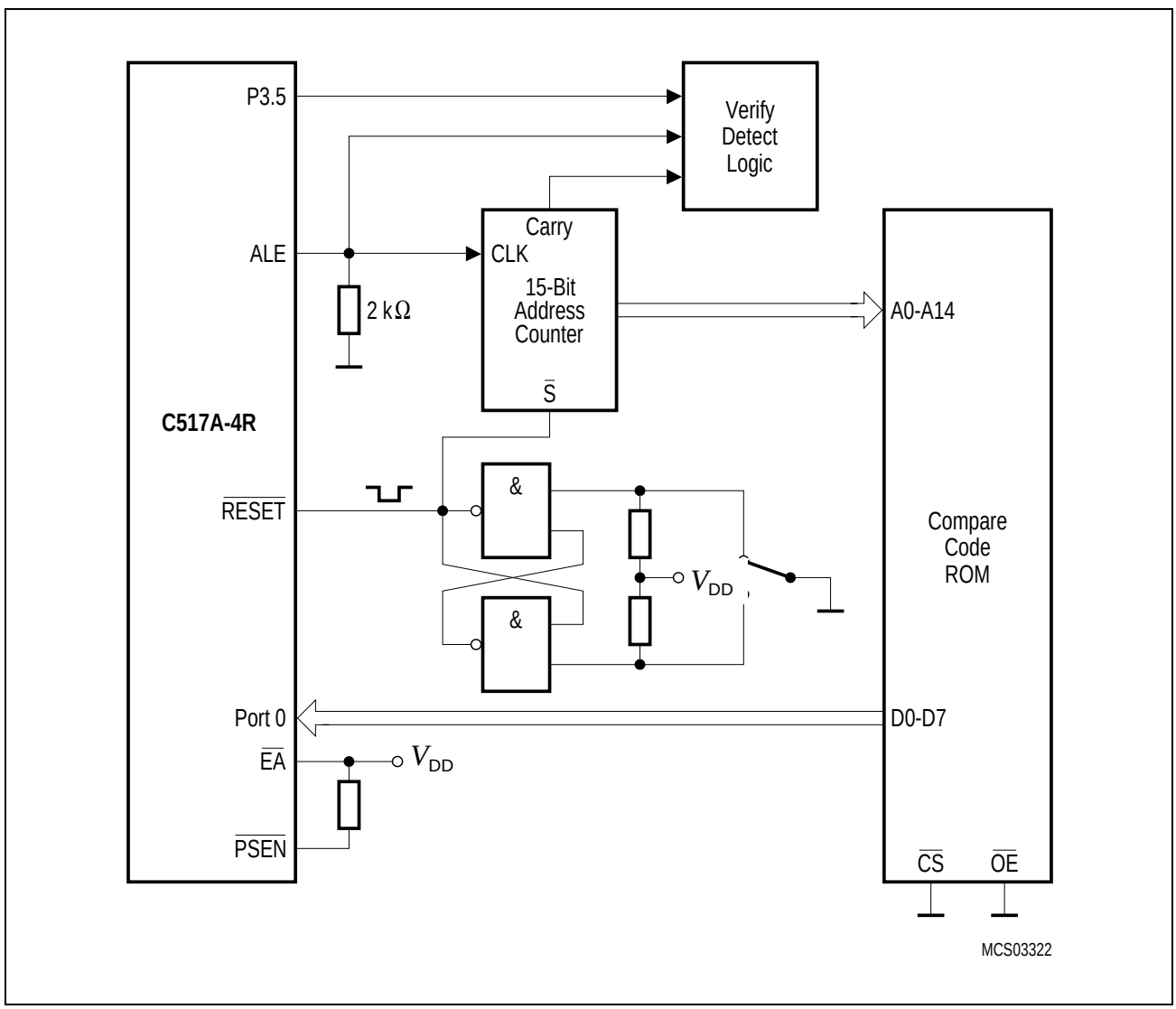


Figure 4-6
ROM Verification Mode 2 - External Circuitry Example

5 Reset and System Clock Operation

5.1 Hardware Reset Operation

The hardware reset function incorporated in the C517A allows for an easy automatic start-up at a minimum of additional hardware and forces the controller to a predefined default state. The hardware reset function can also be used during normal operation in order to restart the device. This is particularly done when the power down mode is to be terminated.

Additional to the hardware reset, which is applied externally to the C517A, there are two internal reset sources, the watchdog timer and the oscillator watchdog. This chapter deals only with the external hardware reset.

The reset input is an active low input. An internal Schmitt trigger is used at the input for noise rejection. Since the reset is synchronized internally, the $\overline{\text{RESET}}$ pin must be held low for at least two machine cycles (24 oscillator periods) while the oscillator is running. With the oscillator running the internal reset is executed during the second machine cycle and is repeated every cycle until $\overline{\text{RESET}}$ goes high again.

During reset, pins ALE and $\overline{\text{PSEN}}$ are configured as inputs and should not be stimulated or driven externally. (An external stimulation at these lines during reset activates several test modes which are reserved for test purposes. This in turn may cause unpredictable output operations at several port pins).

At the $\overline{\text{RESET}}$ pin, a pullup resistor is internally connected to V_{DD} to allow a power-up reset with an external capacitor only. An automatic power-up reset can be obtained when V_{DD} is applied by connecting the reset pin to V_{SS} via a capacitor. After V_{DD} has been turned on, the capacitor must hold the voltage level at the reset pin for a specific time to effect a complete reset.

The time required for a reset operation is the oscillator start-up time plus 2 machine cycles, which, under normal conditions, must be at least 10 - 20 ms for a crystal oscillator. This requirement is typically met using a capacitor of 4.7 to 10 µF. The same considerations apply if the reset signal is generated externally (**figure 5-1 b**). In each case it must be assured that the oscillator has started up properly and that at least two machine cycles have passed before the reset signal goes inactive.

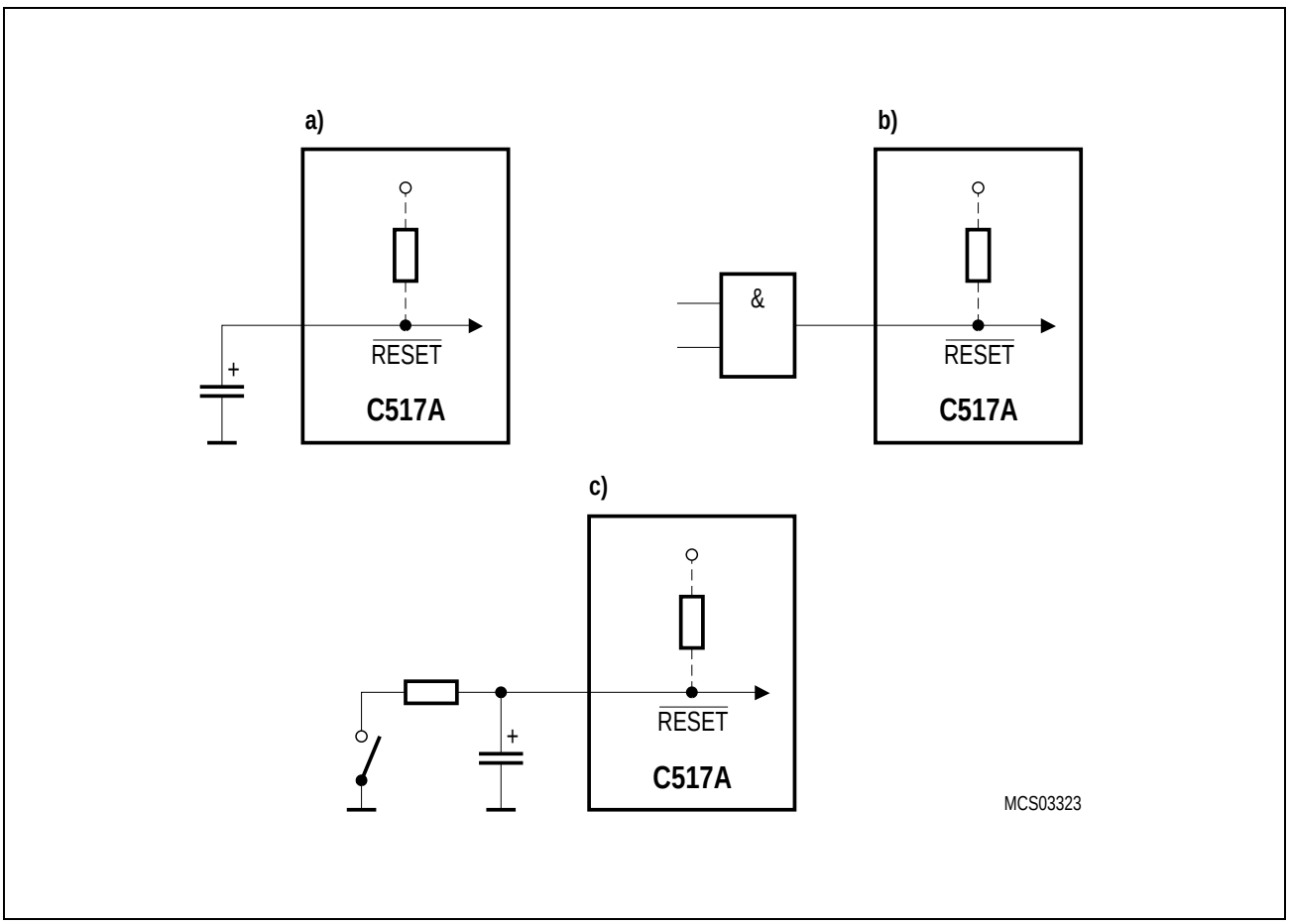


Figure 5-1
Reset Circuitries

A correct reset leaves the processor in a defined state. The program execution starts at location 0000_H. After reset is internally accomplished the port latches of ports 0 to 6 are set to FF_H. This leaves port 0 floating, since it is an open drain port when not used as data/address bus. All other I/O port lines (ports 1,3 to 6) output a one (1). Port 2 lines output a zero (or one) after reset, if $\overline{EA}$ is held low (or high). Port 7 and 8 are input-only ports. They have no internal latch and therefore the contents of the special function registers P7 and P8 depend on the levels applied to port 7 or 8.

The content of the internal RAM of the C517A is not affected by a reset. After power-up the content is undefined, while it remains unchanged during a reset if the power supply is not turned off.

5.2 Fast Internal Reset after Power-On

The C517A uses the oscillator watchdog unit for a fast internal reset procedure after power-on. **Figure 5-1** shows the power-on sequence under control of the oscillator watchdog.

Normally the devices of the 8051 family do not enter their default reset states before the on-chip oscillator starts. The reason is that the external reset signal must be internally synchronized and processed in order to bring the device into the correct reset state. Especially if a crystal is used the start up time of the oscillator is relatively long (typ. 10 ms). During this time period the pins have an undefined state which could have severe effects especially to actuators connected to port pins.

In the C517A the oscillator watchdog unit avoids this situation. In this case, after power-on the oscillator watchdog's RC oscillator starts working within a very short start-up time (typ. less than 2 microseconds). In the following the watchdog circuitry detects a failure condition for the on-chip oscillator because this has not yet started (a failure is always recognized if the watchdog's RC oscillator runs faster than the on-chip oscillator). As long as this condition is detected the watchdog uses the RC oscillator output as clock source for the chip rather than the on-chip oscillator's output. This allows correct resetting of the part and brings also all ports to the defined state (see **figure 5-2**).

Under worst case conditions (fast V_{DD} rise time - e.g. 1µs, measured from V_{DD} = 4.25 V up to stable port condition), the delay between power-on and the correct port reset state is :

- Typ.: 18 µs
- Max.: 34 µs

The RC oscillator will already run at a V_{DD} below 4.25V (lower specification limit). Therefore, at slower V_{DD} rise times the delay time will be less than the two values given above.

After the on-chip oscillator has finally started, the oscillator watchdog detects the correct function; then the watchdog still holds the reset active for a time period of max. 768 cycles of the RC oscillator clock in order to allow the oscillation of the on-chip oscillator to stabilize (**figure 5-2, II**). Subsequently the clock is supplied by the on-chip oscillator and the oscillator watchdog's reset request is released (**figure 5-2, III**). However, an externally applied reset still remains active (**figure 5-2, IV**) and the device does not start program execution (**figure 5-2, V**) before the external reset is also released.

Although the oscillator watchdog provides a fast internal reset it is additionally necessary to apply the external reset signal when powering up. The reasons are as follows:

- Termination of software power down mode
- Reset of the status flag OWDS that is set by the oscillator watchdog during the power up sequence.

Using a crystal or ceramic resonator for clock generation, the external reset signal must be held active at least until the on-chip oscillator has started and the internal watchdog reset phase is completed (after phase III in **figure 5-2**). When an external clock generator is used, phase II is very short. Therefore, an external reset time of typically 1 ms is sufficent in most applications.

Generally, for reset time generation at power-on an external capacitor can be applied to the $\overline{\text{RESET}}$ pin.

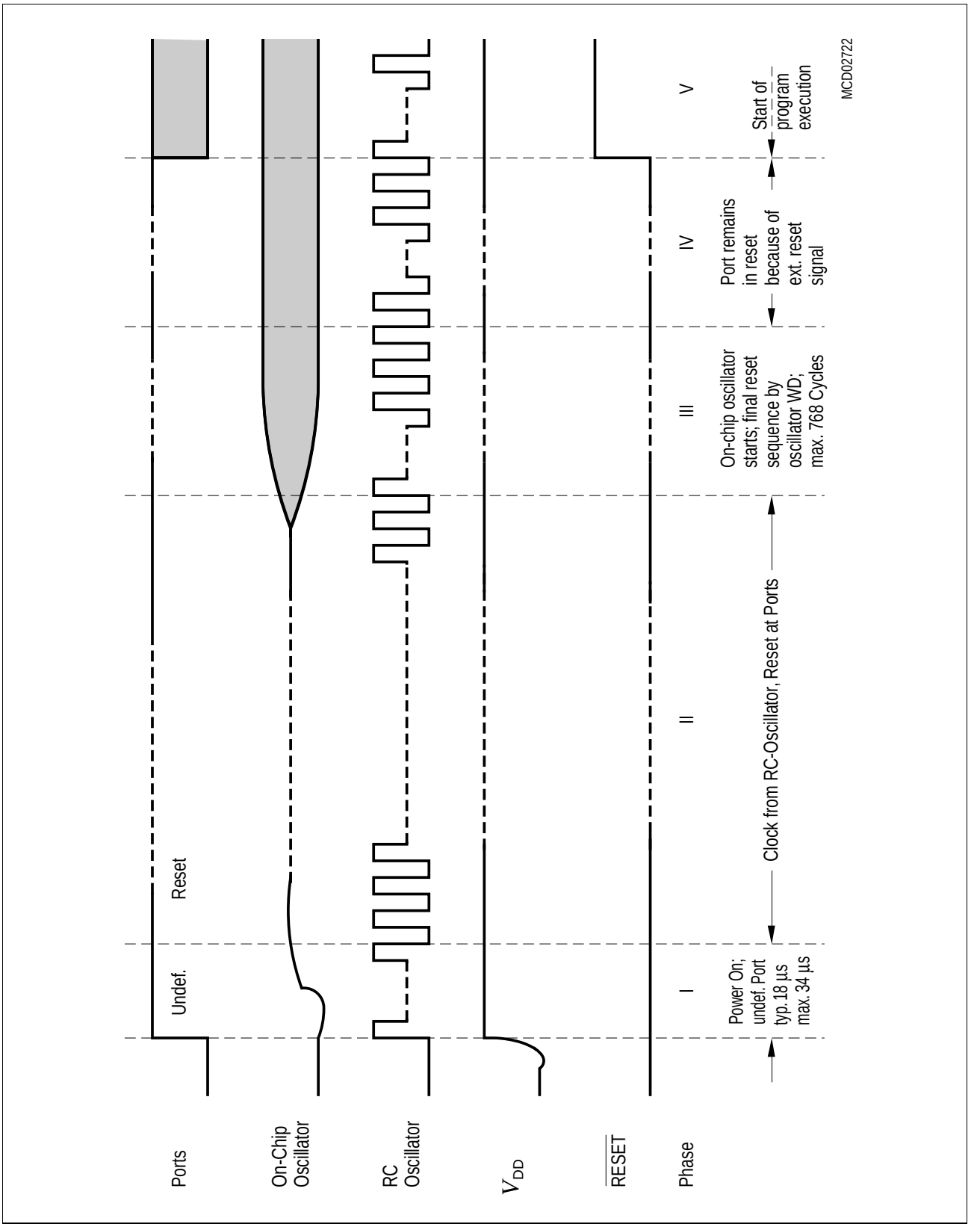


Figure 5-2
Power-On Reset of the C517A

5.3 Hardware Reset Timing

This section describes the timing of the hardware reset signal.

The input pin $\overline{\text{RESET}}$ is sampled once during each machine cycle. This happens in state 5 phase 2. Thus, the external reset signal is synchronized to the internal CPU timing. When the reset is found active (low level) the internal reset procedure is started. It needs two complete machine cycles to put the complete device to its correct reset state, i.e. all special function registers contain their default values, the port latches contain 1's etc. Note that this reset procedure is also performed if there is no clock available at the device. (This is done by the oscillator watchdog, which provides an auxiliary clock for performing a perfect reset without clock at the XTAL1 and XTAL2 pins). The $\overline{\text{RESET}}$ signal must be active for at least two machine cycles; after this time the C517A remains in its reset state as long as the signal is active. When the signal goes inactive this transition is recognized in the following state 5 phase 2 of the machine cycle. Then the processor starts its address output (when configured for external ROM) in the following state 5 phase 1. One phase later (state 5 phase 2) the first falling edge at pin ALE occurs.

Figure 5-3 shows this timing for a configuration with $\overline{\text{EA}}$ = 0 (external program memory). Thus, between the release of the $\overline{\text{RESET}}$ signal and the first falling edge at ALE there is a time period of at least one machine cycle but less than two machine cycles.

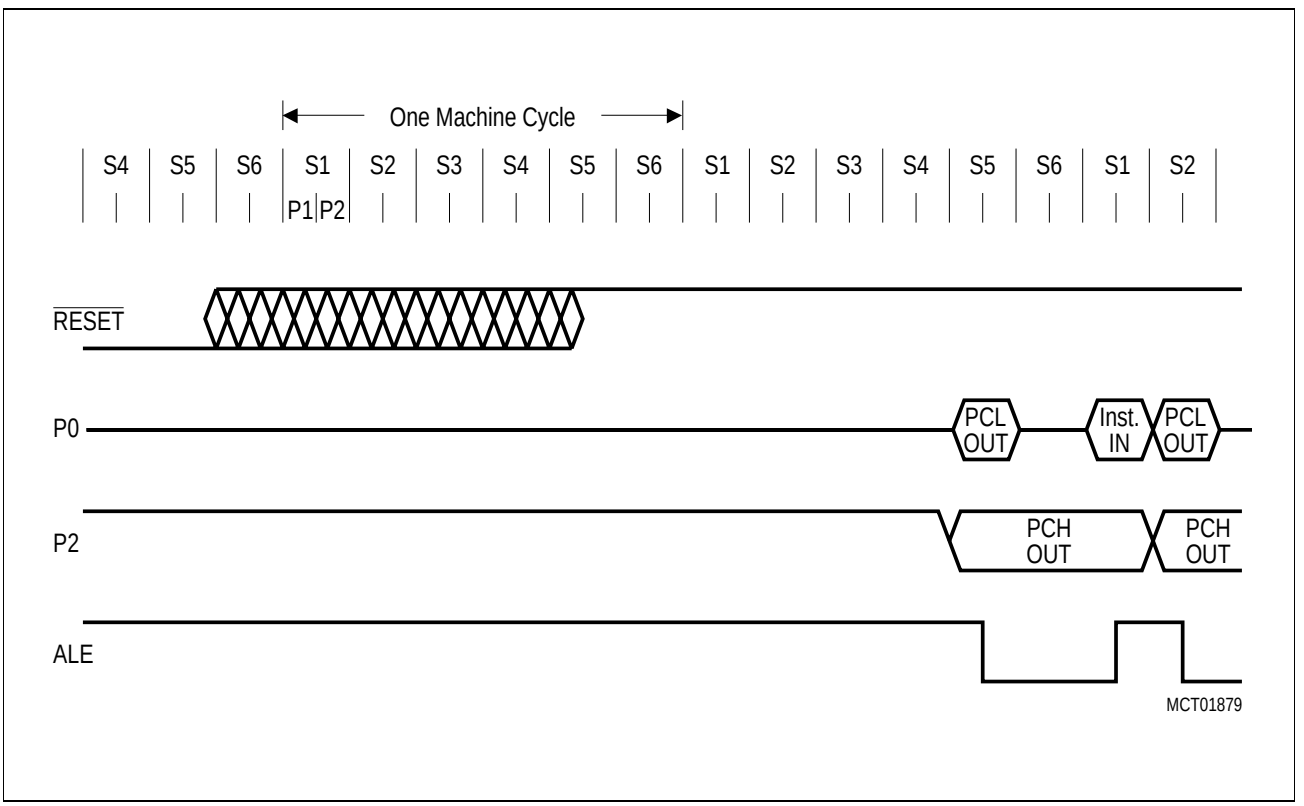


Figure 5-3
CPU Timing after Reset

5.4 Oscillator and Clock Circuit

XTAL1 and XTAL2 are the output and input of a single-stage on-chip inverter which can be configured with off-chip components as a Pierce oscillator. The oscillator, in any case, drives the internal clock generator. The clock generator provides the internal clock signals to the chip. These signals define the internal phases, states and machine cycles.

Figure 5-4 shows the recommended oscillator circuit.

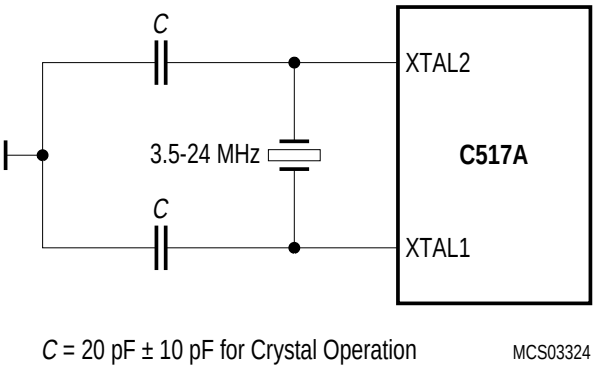


Figure 5-4
Recommended Oscillator Circuit

In this application the on-chip oscillator is used as a crystal-controlled, positive-reactance oscillator (a more detailed schematic is given in **figure 5-5**). It is operated in its fundamental response mode as an inductive reactor in parallel resonance with a capacitor external to the chip. The crystal specifications and capacitances are non-critical. In this circuit 20 pF can be used as single capacitance at any frequency together with a good quality crystal. A ceramic resonator can be used in place of the crystal in cost-critical applications. If a ceramic resonator is used, the two capacitors normally have different values depending on the oscillator frequency. We recommend consulting the manufacturer of the ceramic resonator for value specifications of these capacitors.

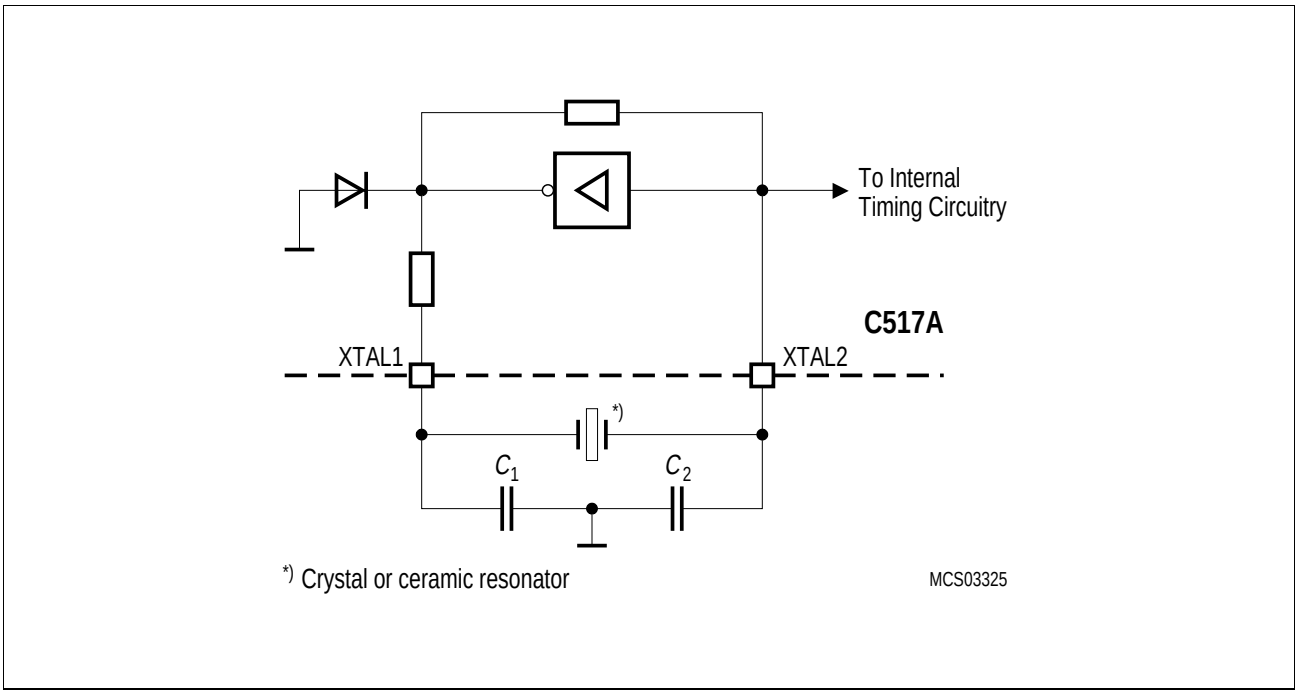


Figure 5-5
On-Chip Oscillator Circuiry

To drive the C517A with an external clock source, the external clock signal has to be applied to XTAL2, as shown in **figure 5-6**. XTAL1 has to be left unconnected. A pullup resistor is suggested (to increase the noise margin), but is optional if V_{OH} of the driving gate corresponds to the V_{IH2} specification of XTAL2.

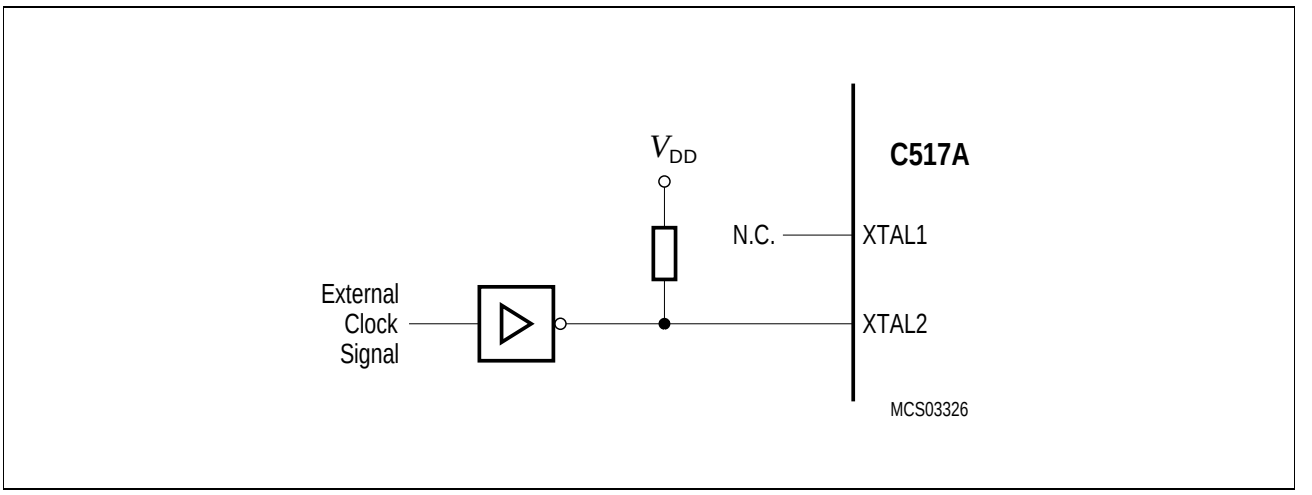


Figure 5-6
External Clock Source

5.5 System Clock Output

For peripheral devices requiring a system clock, the C517A provides a clock output signal derived from the oscillator frequency as an alternate output function on pin P1.6/CLKOUT. If bit CLK is set (bit 6 of special function register ADCON0), a clock signal with 1/12 of the oscillator frequency is gated to pin P1.6/CLKOUT. To use this function the port pin must be programmed to a one (1), which is also the default after reset.

Special Function Register ADCON0 (Address $D8_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	DF_H	DE_H	DD_H	DC_H	DB_H	DA_H	$D9_H$	$D8_H$	
$D8_H$	BD	CLK	ADEX	BSY	ADM	MX2	MX1	MX0	ADCON0

The shaded bits are not used for clock output control.

Bit	Function
CLK	Clock output enable bit When set, pin P1.6/CLKOUT outputs the system clock which is 1/12 of the oscillator frequency.

The system clock is high during S3P1 and S3P2 of every machine cycle and low during all other states. Thus, the duty cycle of the clock signal is 1:6. Associated with a MOVX instruction the system clock coincides with the last state (S3) in which a $\overline{RD}$ or $\overline{WR}$ signal is active. A timing diagram of the system clock output is shown in **figure 5-7**.

Note : During slow-down operation the frequency of the CLKOUT signal is divided by 8.

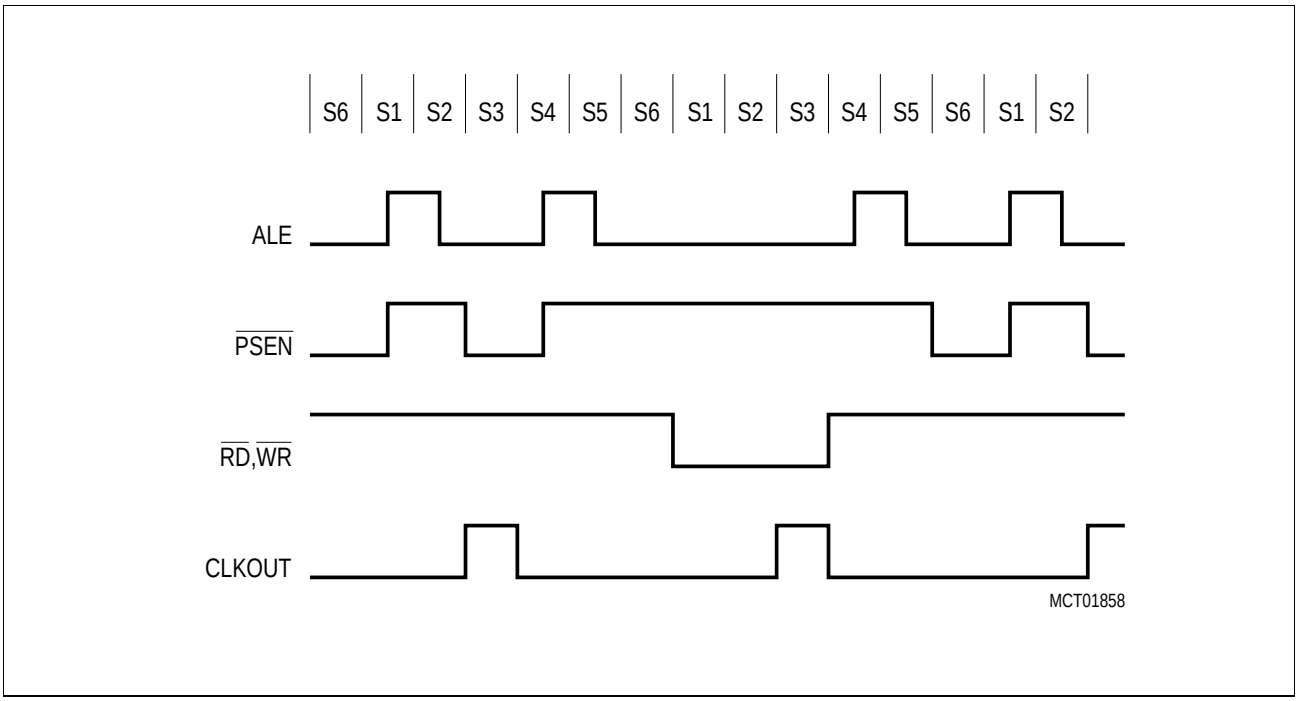


Figure 5-7
Timing Diagram - System Clock Output

6 On-Chip Peripheral Components

This chapter gives detailed information about all on-chip peripherals of the C517A except for the integrated interrupt controller, which is described separately in chapter 7.

6.1 Parallel I/O

The C517A has seven 8-bit digital I/O ports and one 8-bit and one 4-bit input port for analog/digital input. Port 0 is an open-drain bidirectional I/O port, while ports 1 to 6 are quasi-bidirectional I/O ports with internal pullup resistors. That means, when configured as inputs, ports 1 to 6 will be pulled high and will source current when externally pulled low. Port 0 will float when configured as input.

The output drivers of port 0 and 2 and the input buffers of port 0 are also used for accessing external memory. In this application, port 0 outputs the low byte of the external memory address, time multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. Otherwise, the port 2 pins continue emitting the P2 SFR contents. In this function, port 0 is not an open-drain port, but uses a strong internal pullup FET .

6.1.1 Port Structures

The C517A generally allows digital I/O on 56 lines grouped into 7 bidirectional C501 compatible 8-bit ports and one 8-bit and one 4-bit analog/digital input port. Each port bit (except port 7, 8) consists of a latch, an output driver and an input buffer. Read and write accesses to the I/O ports P0 to P6 are performed via their corresponding special function registers. Depending on the specific ports, multiple functions are assigned to the port pins. These alternate functions of the port pins are listed in **table 6-1**.

When port 7 or 8 is used as analog input, an analog channel is switched to the A/D converter through a 4-bit multiplexer, which is controlled by three bits in SFR ADCON1. Port 6 lines may also be used as digital inputs. In this case they are addressed as an input port via SFR P7 or P8. Since ports 7 and 8 have no internal latch, the contents of SFR P7 or P8 only depends on the levels applied to the input lines. It makes no sense to output a value to these input-only port by writing to the SFR P7 or P8. This will have no effect.

Table 6-1
Alternate Functions of Port 1, 3, 4, 5, and 6

Port	Alternate Functions	Description
P1.0	$\overline{\text{INT3}}$ / CC0	External Interrupt 3 input / Capture/compare 0 input/output
P1.1	INT4 / CC1	External Interrupt 4 input / Capture/compare 1 input/output
P1.2	INT5 / CC2	External Interrupt 5 input / Capture/compare 2 input/output
P1.3	INT6 / CC3	External Interrupt 6 input / Capture/compare 3 input/output
P1.4	$\overline{\text{INT2}}$ / CC4	External Interrupt 2 input / Capture/compare 4 input/output
P1.5	T2EX	Timer 2 external reload/trigger input
P1.6	CLKOUT	System clock output
P1.7	T2	Timer 2 external count input
P3.0	RxD0	Serial port 0 receiver data input (asynchronous) or data input/output (synchronous)
P3.1	TxD0	Serial port 0 transmitter data output (asynchronous) or data clock output (synchronous)
P3.2	$\overline{\text{INT0}}$	External interrupt 0 input, timer 0 gate control
P3.3	$\overline{\text{INT1}}$	External interrupt 1 input, timer 1 gate control
P3.4	T0	Timer 0 external count input
P3.5	T1	Timer 1 external count input
P3.6	$\overline{\text{WR}}$	External data memory write strobe
P3.7	$\overline{\text{RD}}$	External data memory read strobe
P4.0	CM0	Compare output for the CM0 register
P4.1	CM1	Compare output for the CM1 register
P4.2	CM2	Compare output for the CM2 register
P4.3	CM3	Compare output for the CM3 register
P4.4	CM4	Compare output for the CM4 register
P4.5	CM5	Compare output for the CM5 register
P4.6	CM6	Compare output for the CM6 register
P4.7	CM7	Compare output for the CM7 register
P5.0	CCM0	Concurrent compare 0 output
P5.1	CCM1	Concurrent compare 1 output
P5.2	CCM2	Concurrent compare 2 output
P5.3	CCM3	Concurrent compare 3 output
P5.4	CCM4	Concurrent compare 4 output
P5.5	CCM5	Concurrent compare 5 output
P5.6	CCM6	Concurrent compare 6 output
P5.7	CCM7	Concurrent compare 7 output
P6.0	$\overline{\text{ADST}}$	External A/D converter start
P6.1	RxD1	Serial port 1 receiver data input
P6.2	TxD1	Serial port 1 transmitter data output

6.1.2 Standard I/O Port Circuitry

Figure 6-1 shows a functional diagram of a typical bit latch and I/O buffer, which is the core of each of the seven I/O-ports. The bit latch (one bit in the port's SFR) is represented as a type-D flip-flop, which will clock in a value from the internal bus in response to a "write-to-latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read-latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read-pin" signal from the CPU. Some instructions that read from a port (i.e. from the corresponding port SFR P0 to P6) activate the "read-latch" signal, while others activate the "read-pin" signal.

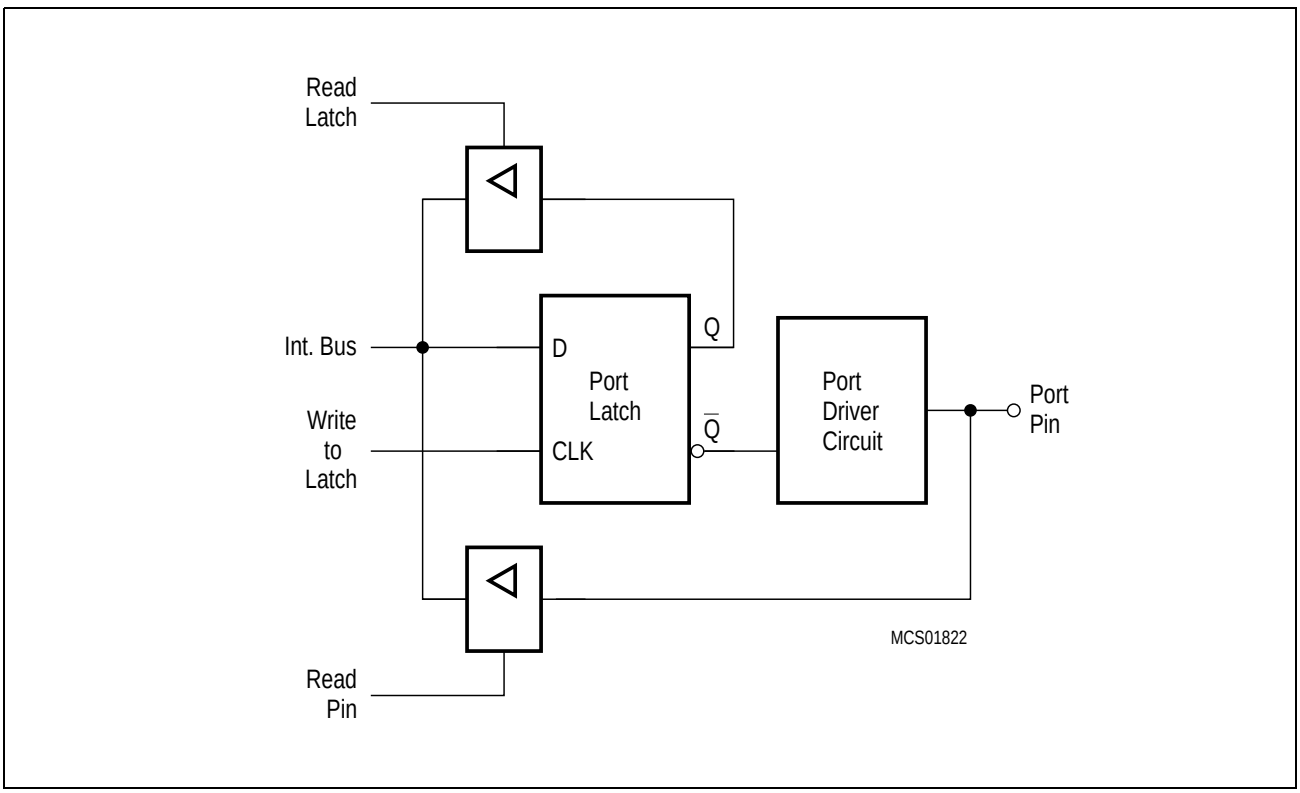


Figure 6-1
Basic Structure of a Port Circuitry

The output drivers of port 1 to 6 have internal pullup FET's (see **figure 6-2**). Each I/O line can be used independently as an input or output. To be used as an input, the port bit stored in the bit latch must contain a one (1) (that means for **figure 6-2**: $\overline{Q}$=0), which turns off the output driver FET n1. Then, for ports 1 to 6 the pin is pulled high by the internal pullups, but can be pulled low by an external source. When externally pulled low the port pins source current (I_{IL} or I_{TL}). For this reason these ports are called "quasi-bidirectional".

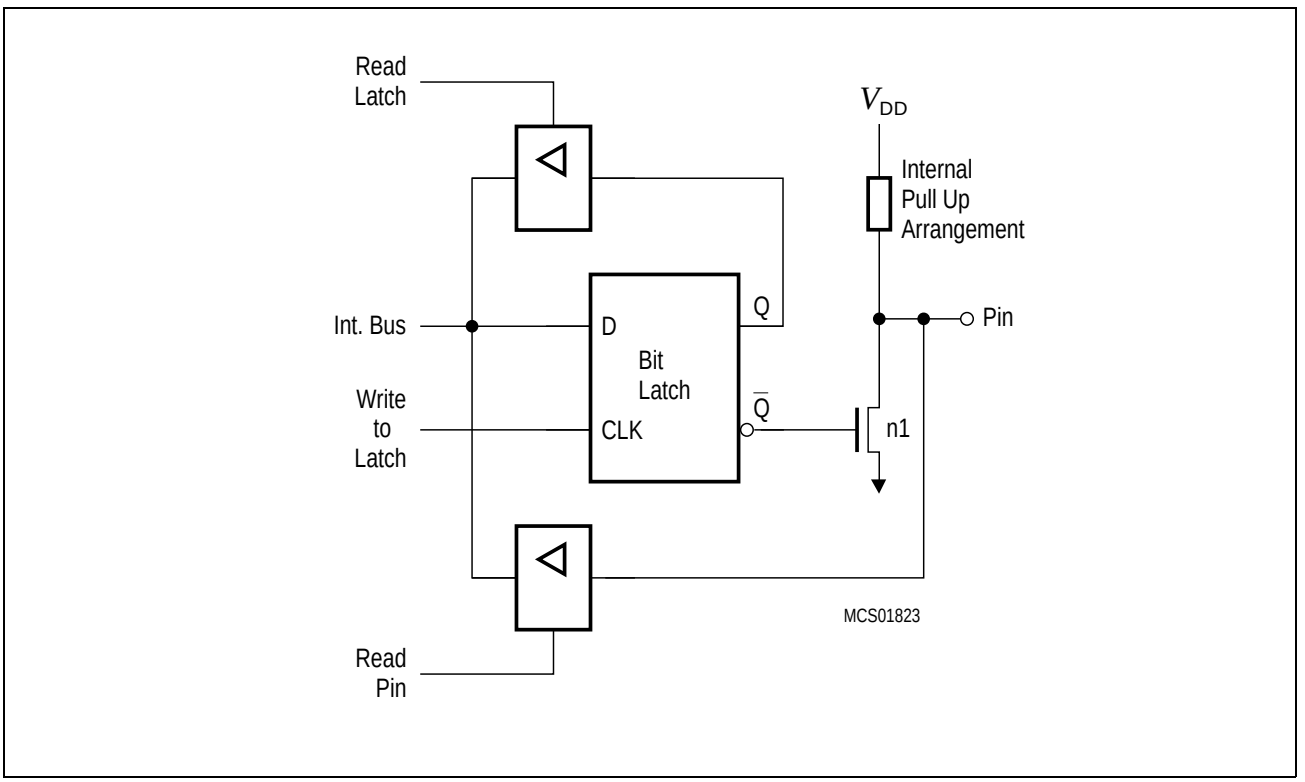


Figure 6-2
Basic Output Driver Circuit of Ports 1 to 6

6.1.2.1 Port 0 Circuitry

Port 0, in contrast to ports 1 to 4, is considered as "true" bidirectional, because the port 0 pins float when configured as inputs. Thus, this port differs in not having internal pullups. The pullup FET in the P0 output driver (see **figure 6-3**) is used only when the port is emitting 1's during the external memory accesses. Otherwise, the pullup is always off. Consequently, P0 lines that are used as output port lines are open drain lines. Writing a "1" to the port latch leaves both output FETs off and the pin floats. In that condition it can be used as high-impedance input. If port 0 is configured as general I/O port and has to emit logic high-level (1), external pullups are required.

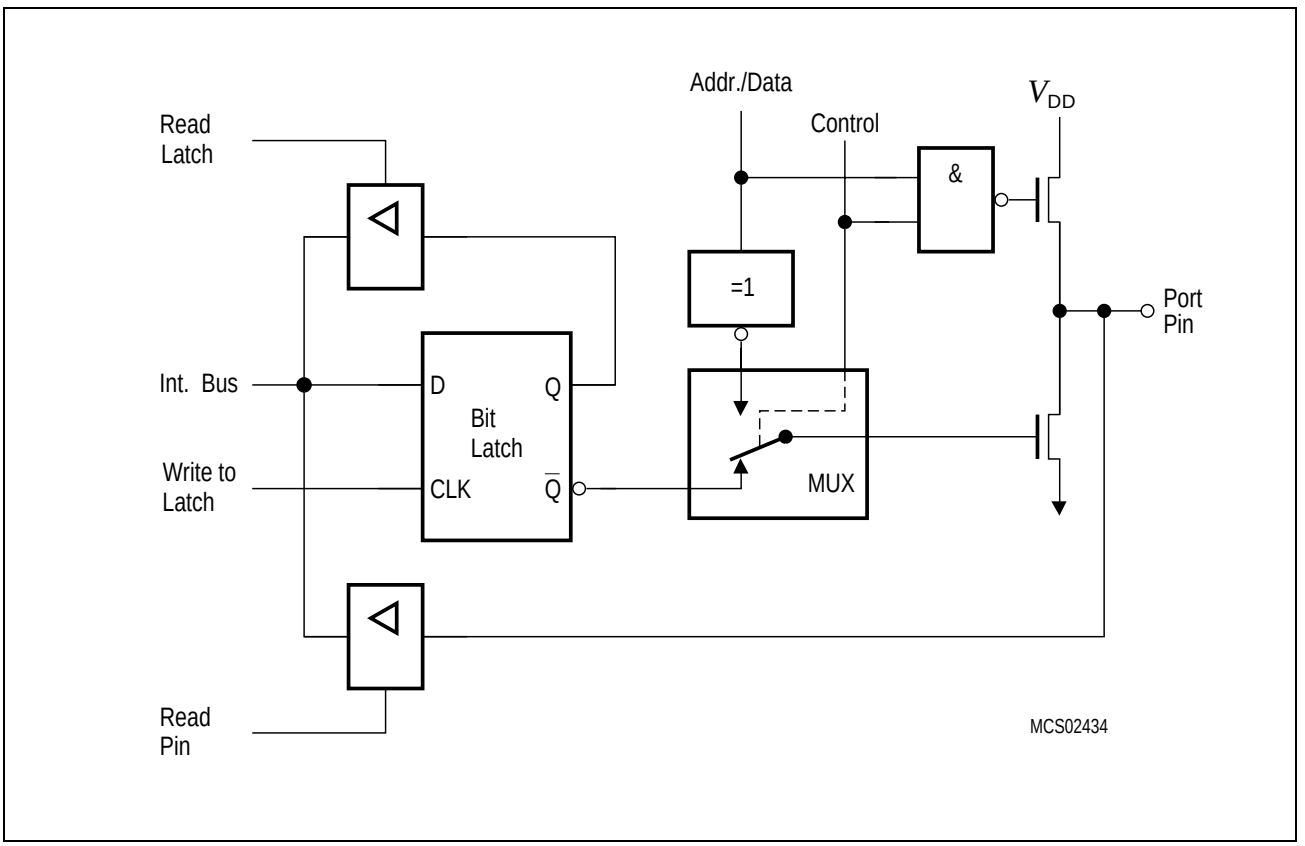


Figure 6-3
Port 0 Circuitry

6.1.2.2 Port 1, Port 3 to Port 6 Circuitry

The pins of ports 1, 3, 4, 5, and 6 are multifunctional. They are port pins and also serve to implement special features as listed in **table 6-1**.

Figure 6-4 shows a functional diagram of a port latch with alternate function. To pass the alternate function to the output pin and vice versa, however, the gate between the latch and driver circuit must be open. Thus, to use the alternate input or output functions, the corresponding bit latch in the port SFR has to contain a one (1); otherwise the pulldown FET is on and the port pin is stuck at 0. After reset all port latches contain ones (1).

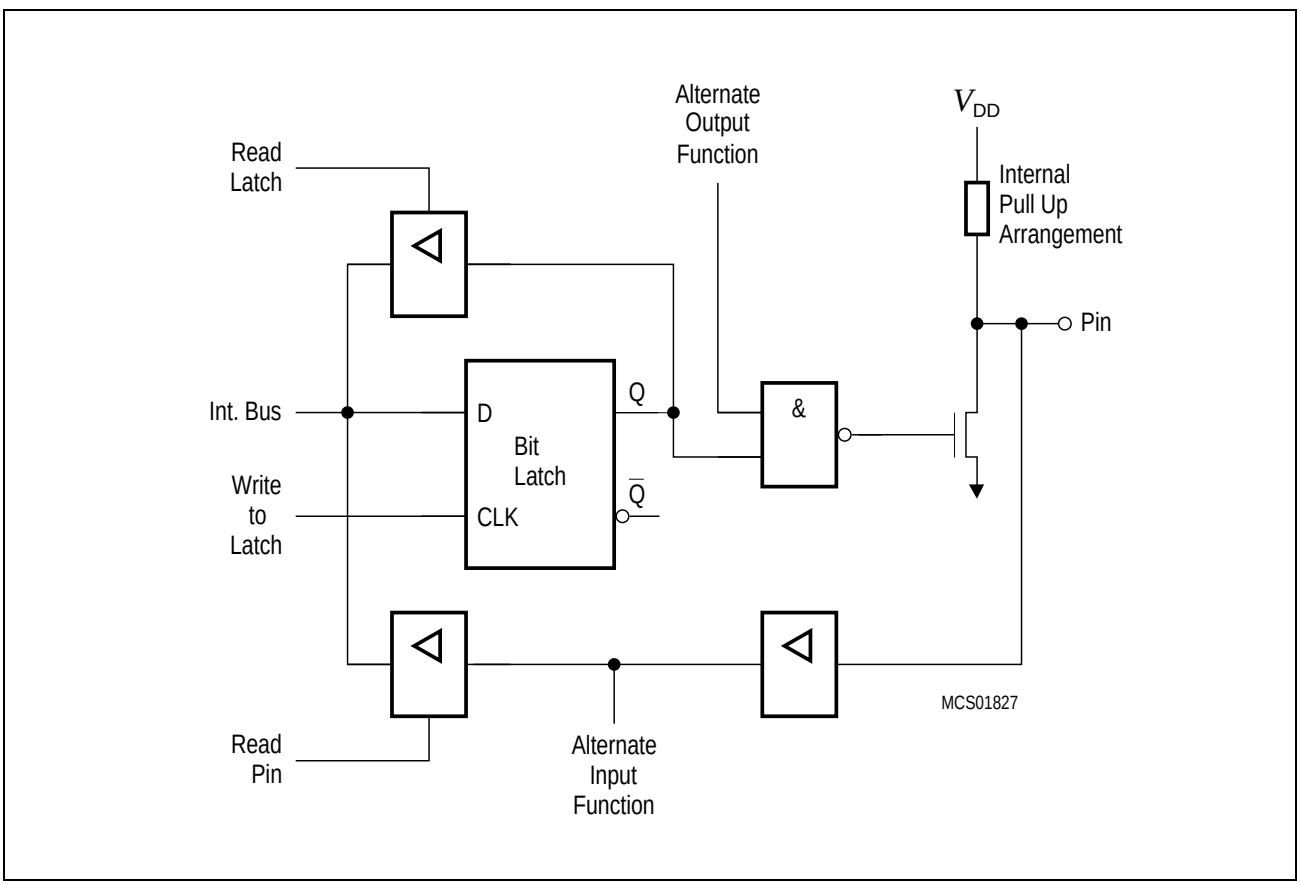


Figure 6-4
Ports 1, 3, 4, 5, and 6

6.1.2.3 Port 2 Circuitry

As shown in **figure 6-3** and below in **figure 6-5**, the output drivers of ports 0 and 2 can be switched to an internal address or address/data bus for use in external memory accesses. In this application they cannot be used as general purpose I/O, even if not all address lines are used externally. The switching is done by an internal control signal dependent on the input level at the $\overline{EA}$ pin and/or the contents of the program counter. If the ports are configured as an address/data bus, the port latches are disconnected from the driver circuit. During this time, the P0/P2 SFR remains unchanged. Being an address/data bus, port 0 uses a pullup FET as shown in **figure 6-3**. When a 16-bit address is used, port 2 uses the additional strong pullups p1 (**figure 6-6**) to emit 1's for the entire external memory cycle instead of the weak ones (p2 and p3) used during normal port activity.

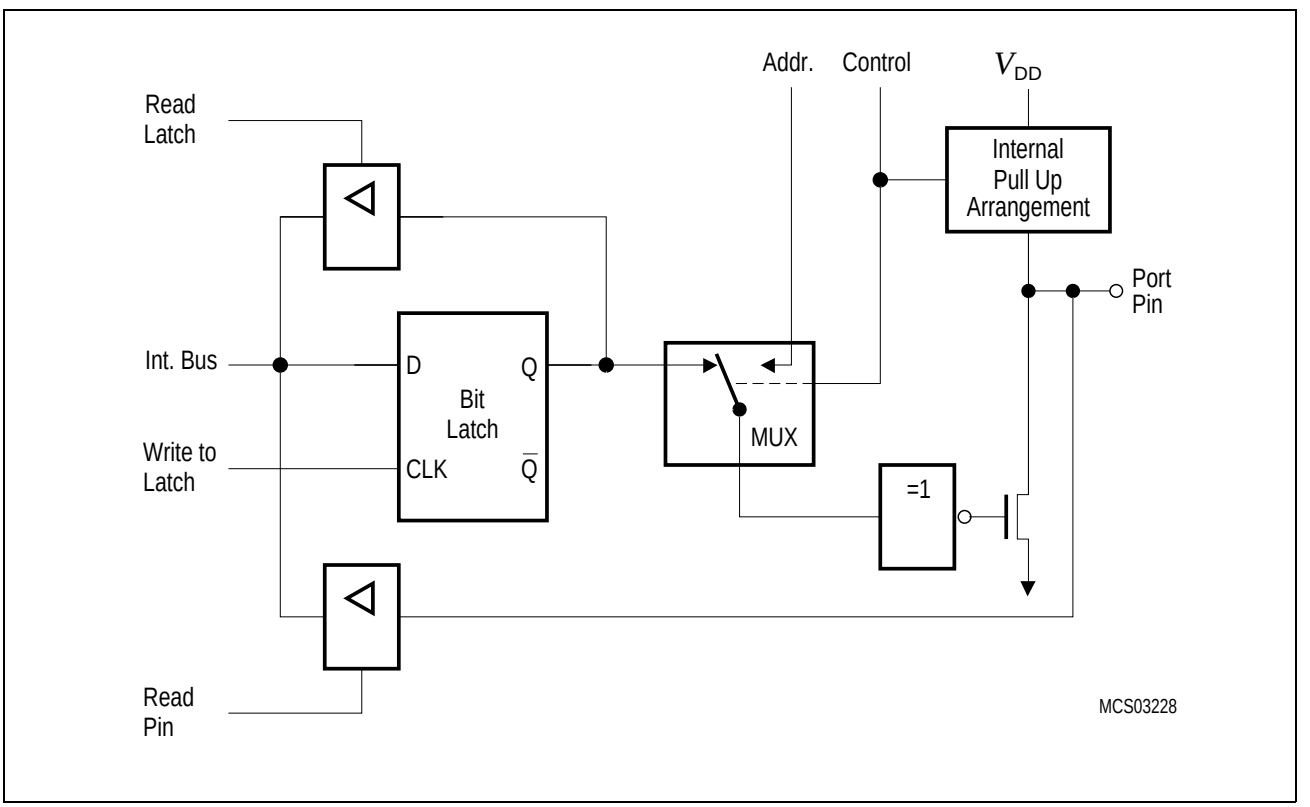


Figure 6-5
Port 2 Circuitry

If no external bus cycles are generated using data or code memory accesses, port 0 can be used for I/O functions.

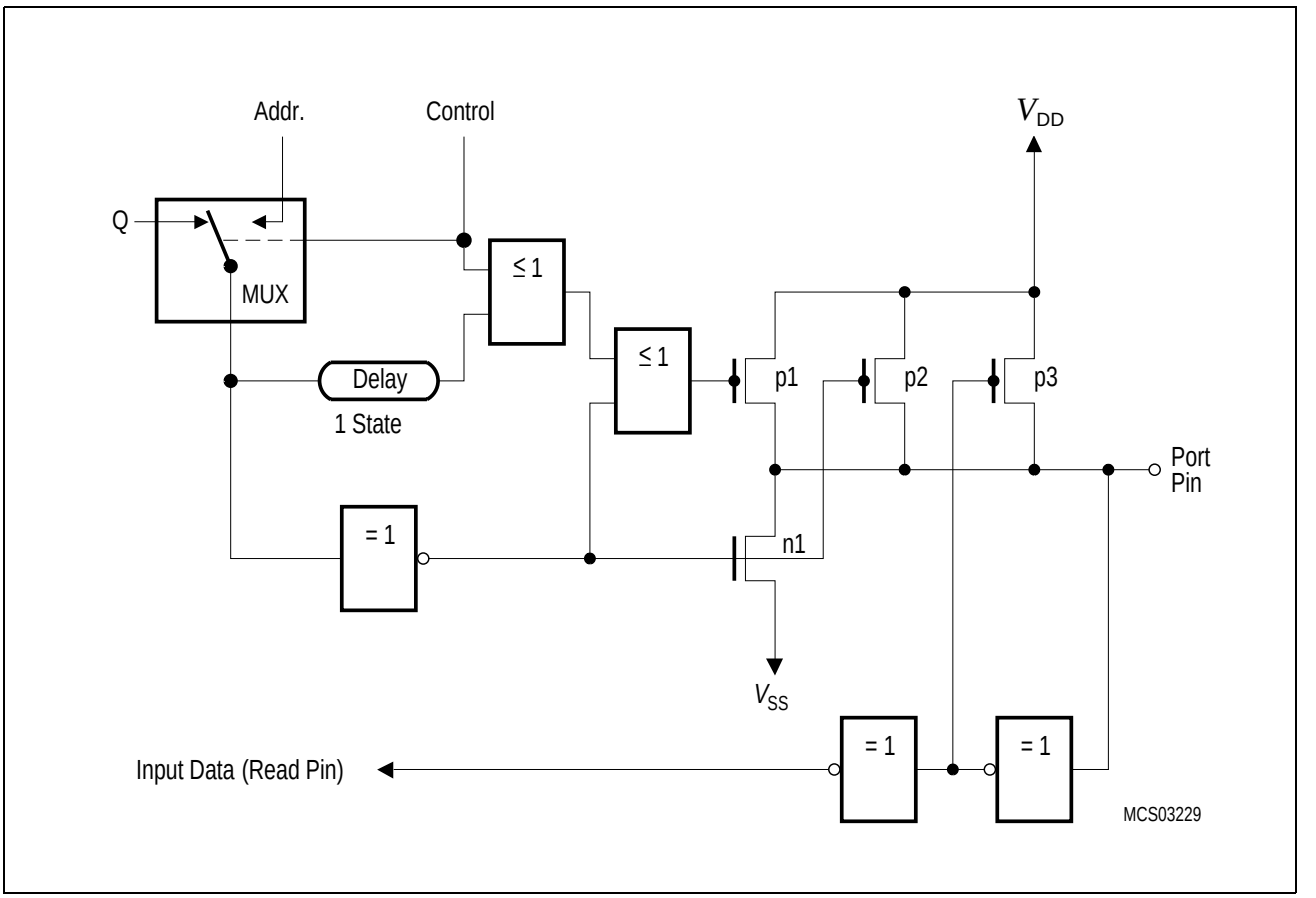


Figure 6-6
Port 2 Pull-up Arrangement

Port 2 in I/O function works similar to the standard port driver circuitry (next section) whereas in address output function it works similar to Port 0 circuitry.

6.1.2.4 Detailed Output Driver Circuitry

In fact, the pullups mentioned before and included in **figure 6-2, 6-4** and **6-5** are pullup arrangements.

Figure 6-7 shows the detailed output driver (pullup arrangement) circuit of the the port 1 and 3 to 6 port lines. The basic circuitry of these ports is shown in **figure 6-4**. The pullup arrangement of these port lines has one n-channel pulldown FET and three pullup FETs:

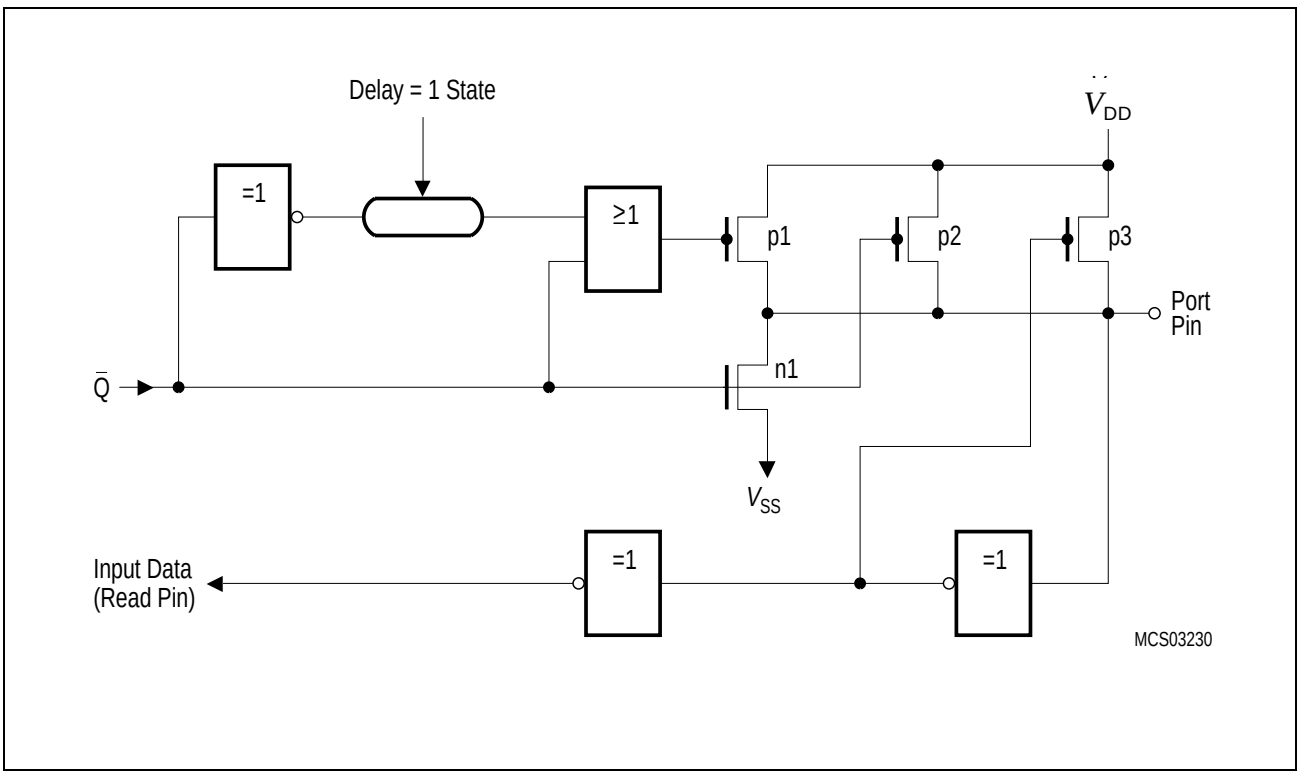


Figure 6-7
Driver Circuit of Ports 1, 3 to 6

- The **pulldown FET n1** is of n-channel type. It is a very strong driver transistor which is capable of sinking high currents (I_{OL}); it is only activated if a "0" is programmed to the port pin. A short circuit to V_{DD} must be avoided if the transistor is turned on, since the high current might destroy the FET. This also means that no “0“ must be programmed into the latch of a pin that is used as input.
- The **pullup FET p1** is of p-channel type. It is activated for two oscillator periods (S1P1 and S1P2) if a 0-to-1 transition is programmed to the port pin, i.e. a "1" is programmed to the port latch which contained a "0". The extra pullup can drive a similar current as the pulldown FET n1. This provides a fast transition of the logic levels at the pin.
- The **pullup FET p2** is of p-channel type. It is always activated when a "1" is in the port latch, thus providing the logic high output level. This pullup FET sources a much lower current than p1; therefore the pin may also be tied to ground, e.g. when used as input with logic low input level.
- The **pullup FET p3** is of p-channel type. It is only activated if the voltage at the port pin is higher than approximately 1.0 to 1.5 V. This provides an additional pullup current if a logic high level shall be output at the pin (and the voltage is not forced lower than approximately 1.0 to

1.5 V). However, this transistor is turned off if the pin is driven to a logic low level, e.g when used as input. In this configuration only the weak pullup FET p2 is active, which sources the current I_{IL} . If, in addition, the pullup FET p3 is activated, a higher current can be sourced (I_{TL}). Thus, an additional power consumption can be avoided if port pins are used as inputs with a low level applied. However, the driving capability is stronger if a logic high level is output.

The described activating and deactivating of the four different transistors translates into four states the pins can be:

- input low state (IL), p2 active only
- input high state (IH) = steady output high state (SOH) p2 and p3 active
- forced output high state (FOH), p1, p2 and p3 active
- output low state (OL), n1 active

If a pin is used as input and a low level is applied, it will be in IL state, if a high level is applied, it will switch to IH state.

If the latch is loaded with "0", the pin will be in OL state.

If the latch holds a "0" and is loaded with "1", the pin will enter FOH state for two cycles and then switch to SOH state. If the latch holds a "1" and is reloaded with a "1" no state change will occur.

At the beginning of power-on reset the pins will be in IL state (latch is set to "1", voltage level on pin is below of the trip point of p3). Depending on the voltage level and load applied to the pin, it will remain in this state or will switch to IH (=SOH) state.
If it is used as output, the weak pull-up p2 will pull the voltage level at the pin above p3's trip point after some time and p3 will turn on and provide a strong "1". Note, however, that if the load exceeds the drive capability of p2 (I_{IL}), the pin might remain in the IL state and provide a week "1" until the first 0-to-1 transition on the latch occurs. Until this the output level might stay below the trip point of the external circuitry.

The same is true if a pin is used as bidirectional line and the external circuitry is switched from output to input when the pin is held at "0" and the load then exceeds the p2 drive capabilities.

If the load exceeds I_{IL} the pin can be forced to "1" by writing a "0" followed by a "1" to the port pin.

6.1.3 Port Timing

When executing an instruction that changes the value of a port latch, the new value arrives at the latch during S6P2 of the final cycle of the instruction. However, port latches are only sampled by their output buffers during phase 1 of any clock period (during phase 2 the output buffer holds the value it noticed during the previous phase 1). Consequently, the new value in the port latch will not appear at the output pin until the next phase 1, which will be at S1P1 of the next machine cycle.

When an instruction reads a value from a port pin (e.g. MOV A, P1) the port pin is actually sampled in state 5 phase 1 or phase 2 depending on port and alternate functions. **Figure 6-8** illustrates this port timing. It must be noted that this mechanism of sampling once per machine cycle is also used if a port pin is to detect an "edge", e.g. when used as counter input. In this case an "edge" is detected when the sampled value differs from the value that was sampled the cycle before. Therefore, there must be met certain reqirements on the pulse length of signals in order to avoid signal "edges" not being detected. The minimum time period of high and low level is one machine cycle, which guarantees that this logic level is noticed by the port at least once.

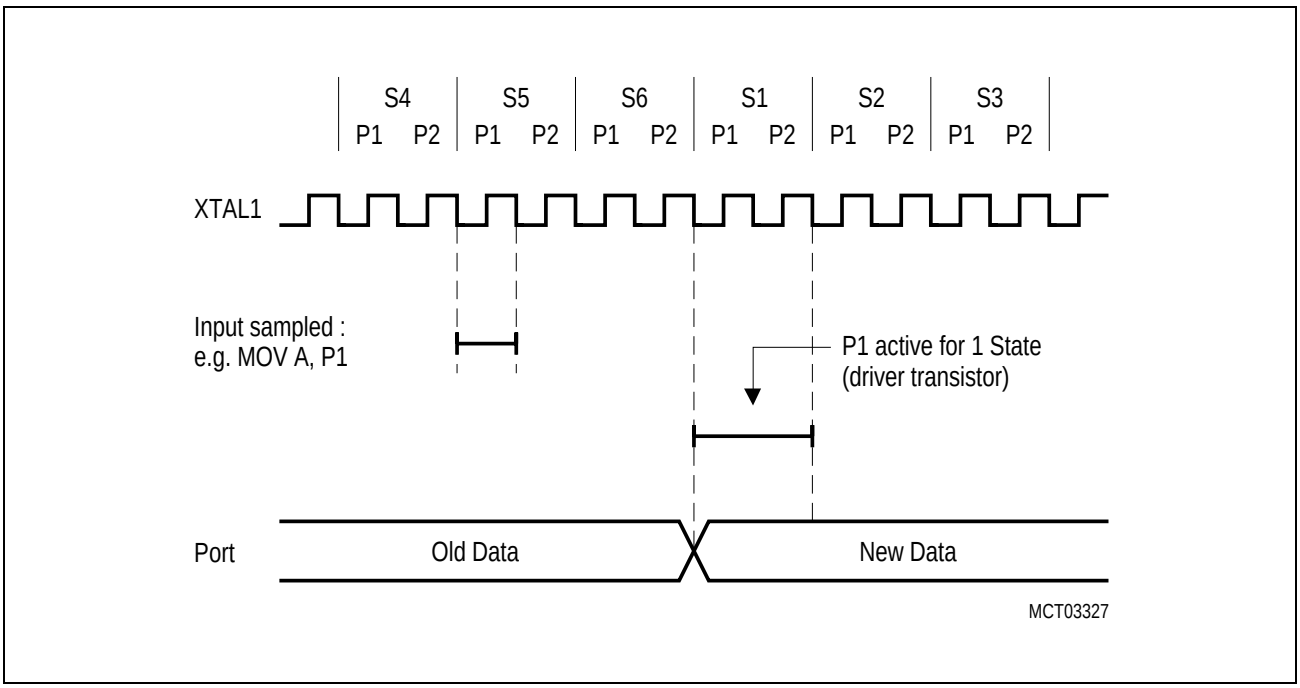


Figure 6-8
Port Timing

6.1.4 Port Loading and Interfacing

The output buffers of ports 1 to 5 can drive TTL inputs directly. The maximum port load which still guarantees correct logic output levels can be be looked up in the DC characteristics in the Data Sheet of the C517A. The corresponding parameters are V_{OL} and V_{OH}.

The same applies to port 0 output buffers. They do, however, require external pullups to drive floating inputs, except when being used as the address/data bus.

When used as inputs it must be noted that the ports 1 to 5 are not floating but have internal pullup transistors. The driving devices must be capable of sinking a sufficient current if a logic low level shall be applied to the port pin (the parameters I_{TL} and I_{IL} in the DC characteristics specify these currents). Port 0 as well as port 1 programmed to analog input function, however, have floating inputs when used for digital input.

6.1.5 Read-Modify-Write Feature of Ports 0 to 6

Some port-reading instructions read the latch and others read the pin. The instructions reading the latch rather than the pin read a value, possibly change it, and then rewrite it to the latch. These are called "read-modify-write"- instructions, which are listed in **table 6-2**. If the destination is a port or a port pin, these instructions read the latch rather than the pin. Note that all other instructions which can be used to read a port, exclusively read the port pin. In any case, reading from latch or pin, resp., is performed by reading the SFR P0, P2 and P3; for example, "MOV A, P3" reads the value from port 3 pins, while "ANL P3, #0AAH" reads from the latch, modifies the value and writes it back to the latch.

It is not obvious that the last three instructions in **table 6-2** are read-modify-write instructions, but they are. The reason is that they read the port byte, all 8 bits, modify the addressed bit, then write the complete byte back to the latch.

Table 6-2
"Read-Modify-Write"-Instructions

Instruction	Function
ANL	Logic AND; e.g. ANL P1, A
ORL	Logic OR; e.g. ORL P2, A
XRL	Logic exclusive OR; e.g. XRL P3, A
JBC	Jump if bit is set and clear bit; e.g. JBC P1.1, LABEL
CPL	Complement bit; e.g. CPL P3.0
INC	Increment byte; e.g. INC P4
DEC	Decrement byte; e.g. DEC P5
DJNZ	Decrement and jump if not zero; e.g. DJNZ P3, LABEL
MOV Px.y,C	Move carry bit to bit y of port x
CLR Px.y	Clear bit y of port x
SETB Px.y	Set bit y of port x

The reason why read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of a transistor. When a "1" is written to the bit, the transistor is turned on. If the CPU then reads the same port bit at the pin rather than the latch, it will read the base voltage of the transitor (approx. 0.7 V, i.e. a logic low level!) and interpret it as "0". For example, when modifying a port bit by a SETB or CLR instruction, another bit in this port with the above mentioned configuration might be changed if the value read from the pin were written back to th latch. However, reading the latch rater than the pin will return the correct value of "1".

6.2 Timers/Counters

The C517A contains three general purpose 16-bit timers/counters, timer 0, 1, and 2, and the compare timer which are useful in many applications for timing and counting.

In "timer" function, the timer register is incremented every machine cycle. Thus one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the counter rate is 1/12 of the oscillator frequency.

In "counter" function, the register is incremented in response to a 1-to-0 transition (falling edge) at its corresponding external input pin, T0 or T1 (alternate functions of P3.4 and P3.5, resp.). In this function the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes two machine cycles (24 oscillator periods) 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 must be held for at least one full machine cycle.

6.2.1 Timer/Counter 0 and 1

Timer / counter 0 and 1 of the C517A are fully compatible with timer / counter 0 and 1 of the C501 and can be used in the same four operating modes:

Mode 0: 8-bit timer/counter with a divide-by-32 prescaler

Mode 1: 16-bit timer/counter

Mode 2: 8-bit timer/counter with 8-bit auto-reload

Mode 3: Timer/counter 0 is configured as one 8-bit timer/counter and one 8-bit timer; Timer/counter 1 in this mode holds its count. The effect is the same as setting TR1 = 0.

External inputs $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ can be programmed to function as a gate for timer/counters 0 and 1 to facilitate pulse width measurements.

Each timer consists of two 8-bit registers (TH0 and TL0 for timer/counter 0, TH1 and TL1 for timer/counter 1) which may be combined to one timer configuration depending on the mode that is established. The functions of the timers are controlled by two special function registers TCON and TMOD.

In the following descriptions the symbols TH0 and TL0 are used to specify the high-byte and the low-byte of timer 0 (TH1 and TL1 for timer 1, respectively). The operating modes are described and shown for timer 0. If not explicity noted, this applies also to timer 1.

6.2.1.1 Timer/Counter 0 and 1 Registers

Totally six special function registers control the timer/counter 0 and 1 operation :

- TL0/TH0 and TL1/TH1 - counter registers, low and high part
- TCON and TMOD - control and mode select registers

Special Function Register TL0 (Address $8A_H$) **Reset Value : 00_H**
Special Function Register TH0 (Address $8C_H$) **Reset Value : 00_H**
Special Function Register TL1 (Address $8B_H$) **Reset Value : 00_H**
Special Function Register TH1 (Address $8D_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$8A_H$	.7	.6	.5	.4	.3	.2	.1	.0	TL0
$8C_H$	.7	.6	.5	.4	.3	.2	.1	.0	TH0
$8B_H$	.7	.6	.5	.4	.3	.2	.1	.0	TL1
$8D_H$	.7	.6	.5	.4	.3	.2	.1	.0	TH1

Bit	Function	
TLx.7-0 x=0-1	Timer/counter 0/1 low register	
	Operating Mode	**Description**
	0	"TLx" holds the 5-bit prescaler value.
	1	"TLx" holds the lower 8-bit part of the 16-bit timer/counter value.
	2	"TLx" holds the 8-bit timer/counter value.
	3	TL0 holds the 8-bit timer/counter value; TL1 is not used.
THx.7-0 x=0-1	Timer/counter 0/1 high register	
	Operating Mode	**Description**
	0	"THx" holds the 8-bit timer/counter value.
	1	"THx" holds the higher 8-bit part of the 16-bit timer/counter value
	2	"THx" holds the 8-bit reload value.
	3	TH0 holds the 8-bit timer value; TH1 is not used.

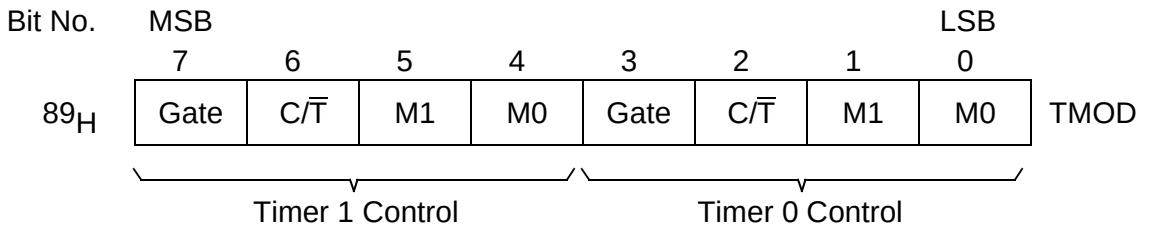
Special Function Register TCON (Address 88_H) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
	$8F_H$	$8E_H$	$8D_H$	$8C_H$	$8B_H$	$8A_H$	89_H	88_H	
88_H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	TCON

The shaded bits are not used in controlling timer/counter 0 and 1.

Bit	Function
TR0	Timer 0 run control bit Set/cleared by software to turn timer/counter 0 ON/OFF.
TF0	Timer 0 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.
TF1	Timer 1 overflow flag Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.

Special Function Register TMOD (Address 89_H) **Reset Value : 00_H**



Bit	Function
GATE	Gating control When set, timer/counter "x" is enabled only while "INT x" pin is high and "TRx" control bit is set. When cleared timer "x" is enabled whenever "TRx" control bit is set.
C/$\overline{T}$	Counter or timer select bit Set for counter operation (input from "Tx" input pin). Cleared for timer operation (input from internal system clock).
M1 M0	Mode select bits

M1	M0	Function
0	0	8-bit timer/counter: "THx" operates as 8-bit timer/counter "TLx" serves 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. Timer 1 : Timer/counter 1 stops

6.2.1.2 Mode 0

Putting either timer/counter 0,1 into mode 0 configures it as an 8-bit timer/counter with a divide-by-32 prescaler. **Figure 6-9** 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 1's to all 0's, it sets the timer overflow flag TF0. The overflow flag TF0 then can be used to request an interrupt. The counted input is enabled to the timer when TR0 = 1 and either Gate = 0 or $\overline{INT0}$ = 1 (setting Gate = 1 allows the timer to be controlled by external input $\overline{INT0}$, to facilitate pulse width measurements). TR0 is a control bit in the special function register TCON; Gate is in TMOD.

The 13-bit register consists of all 8 bits of TH0 and the lower 5 bits of TL0. The upper 3 bits of TL0 are indeterminate and should be ignored. Setting the run flag (TR0) does not clear the registers.

Mode 0 operation is the same for timer 0 as for timer 1. Substitute TR0, TF0, TH0, TL0 and INT0 for the corresponding timer 1 signals in **figure 6-9**. There are two different gate bits, one for timer 1 (TMOD.7) and one for timer 0 (TMOD.3).

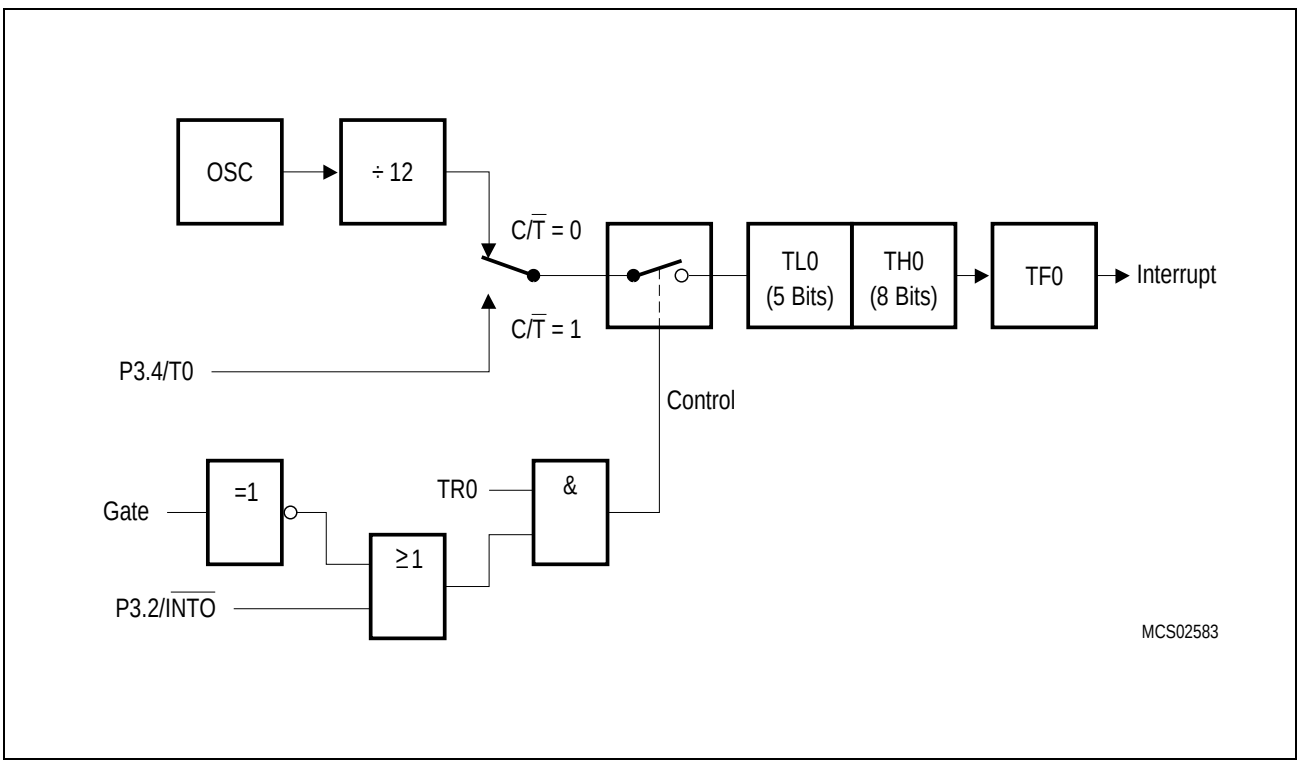


Figure 6-9
Timer/Counter 0, Mode 0: 13-Bit Timer/Counter

6.2.1.3 Mode 1

Mode 1 is the same as mode 0, except that the timer register is running with all 16 bits. Mode 1 is shown in **figure 6-10**.

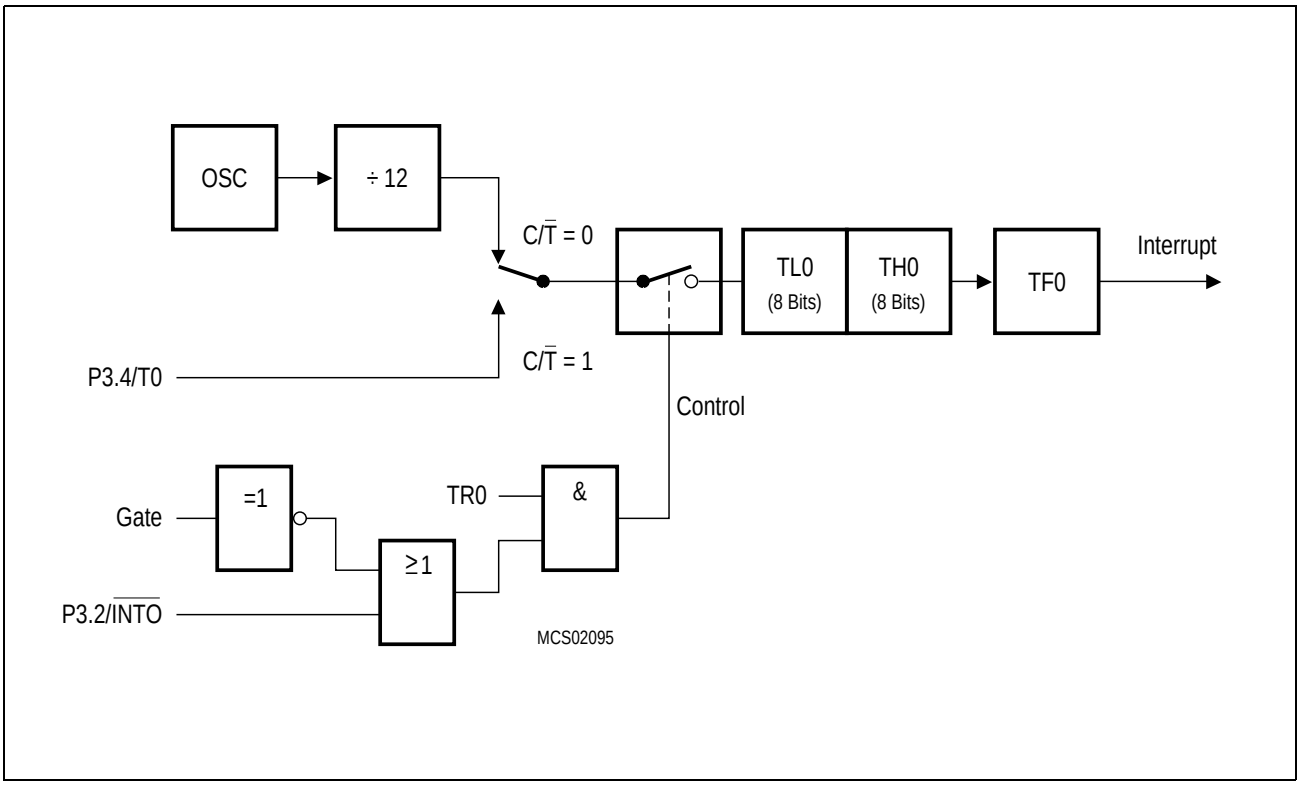


Figure 6-10
Timer/Counter 0, Mode 1: 16-Bit Timer/Counter

6.2.1.4 Mode 2

Mode 2 configures the timer register as an 8-bit counter (TL0) with automatic reload, as shown in **figure 6-11**. Overflow from TL0 not only sets TF0, but also reloads TL0 with the contents of TH0, which is preset by software. The reload leaves TH0 unchanged.

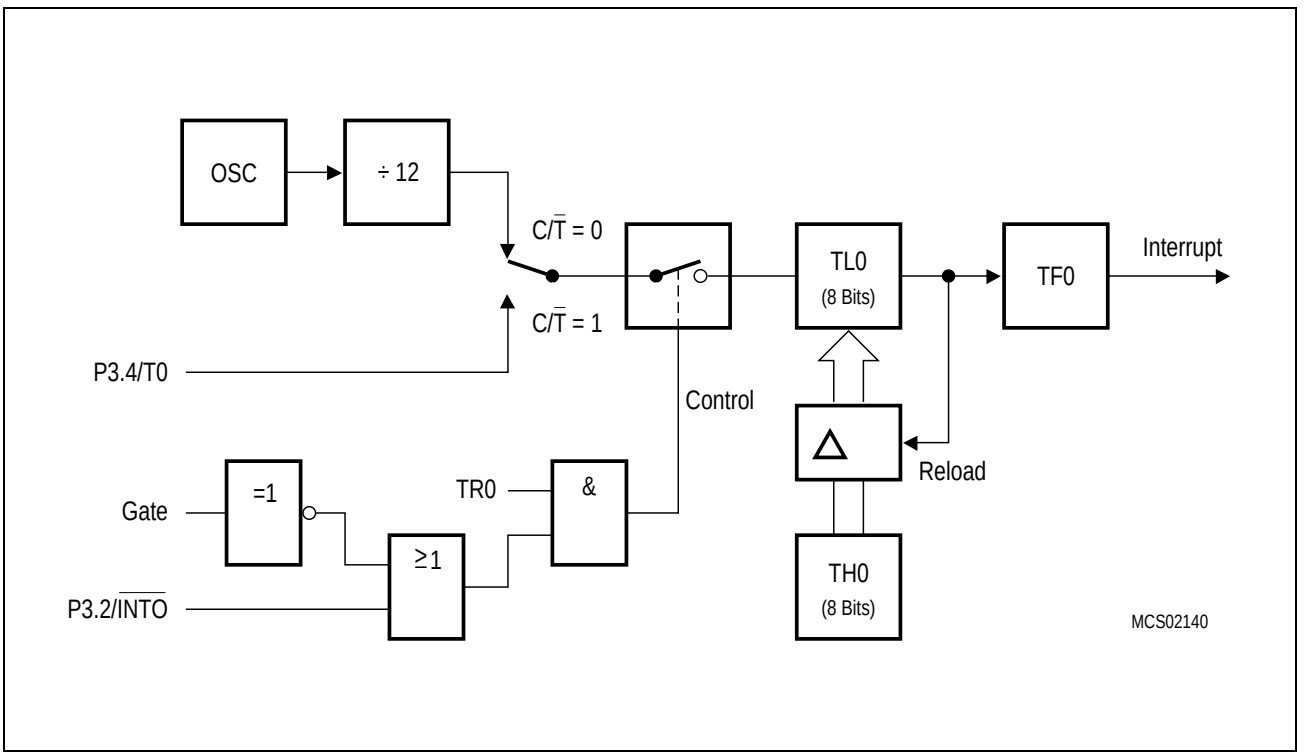


Figure 6-11
Timer/Counter 0,1, Mode 2: 8-Bit Timer/Counter with Auto-Reload

6.2.1.5 Mode 3

Mode 3 has different effects on timer 0 and timer 1. 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 seperate counters. The logic for mode 3 on timer 0 is shown in **figure 6-12**. TL0 uses the timer 0 control bits: C/$\overline{T}$, Gate, TR0, $\overline{INT0}$ 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 channel as a baud rate generator, or in fact, in any application not requiring an interrupt from timer 1 itself.

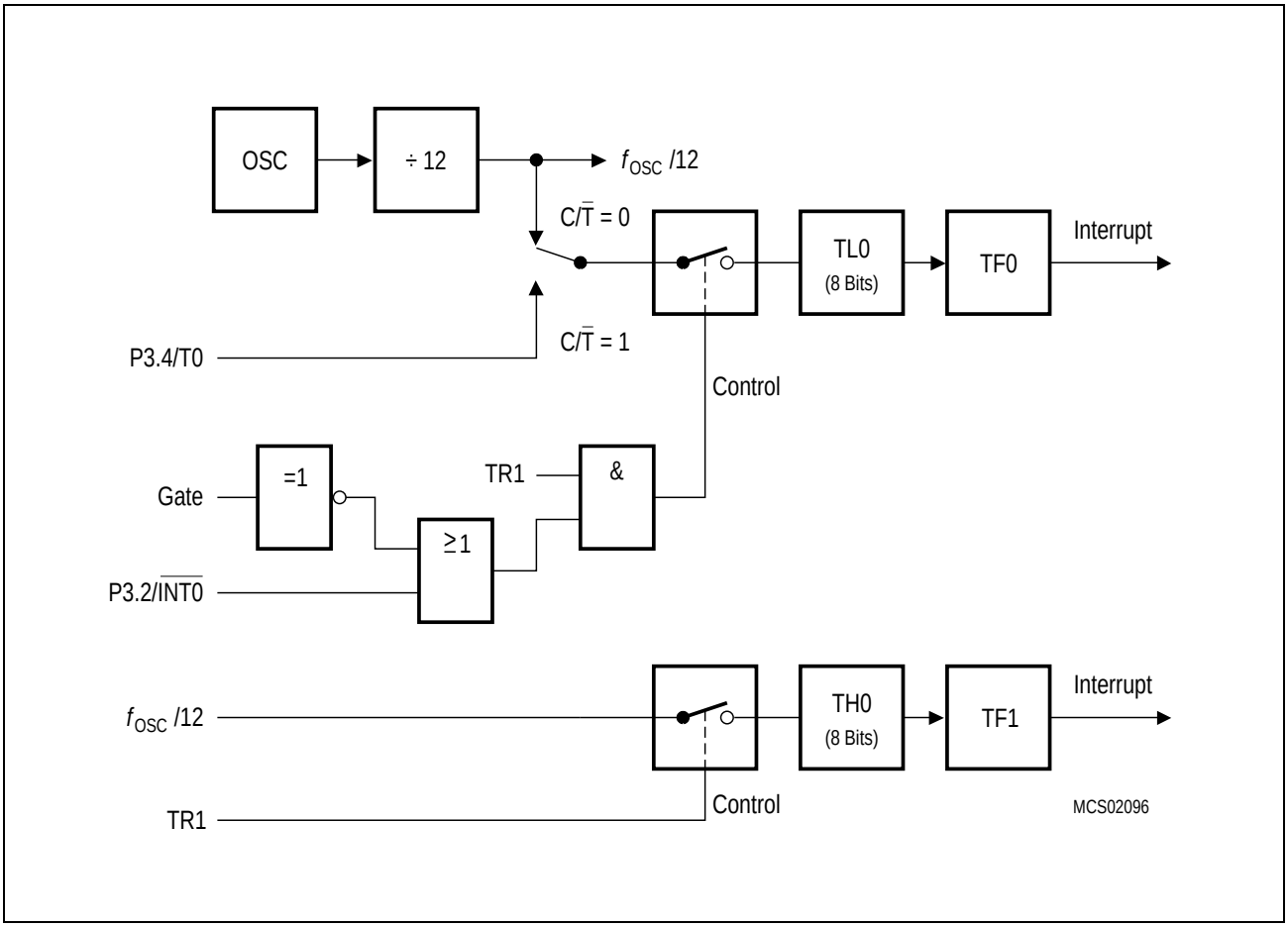


Figure 6-12
Timer/Counter 0, Mode 3: Two 8-Bit Timers/Counters

6.3 The Compare/Capture Unit (CCU)

The compare/capture unit is one of the C517A's most powerful peripheral units for use in all kinds of digital signal generation and event capturing like pulse generation, pulse width modulation, pulse width measuring etc.

The CCU consists of two 16-bit timer/counters with automatic reload feature and an array of 13 compare or compare/capture registers. A set of six control registers is used for flexible adapting of the CCU to a wide variety of user's applications.

The CCU is the ideal peripheral unit for various automotive control applications (ignition/injection control, anti-lock brakes, etc.) as well as for industrial applications (DC, three-phase AC, and stepper motor control, frequency generation, digital-to-analog conversion, process control, etc.)

The detailed description in the following sections refers to the CCU's functional blocks as listed below:

- Timer 2 with $f_{OSC}/12$ input clock, 2-bit prescaler, 16-bit reload, counter/gated timer mode and overflow interrupt request.
- Compare timer with $f_{OSC}/2$ input clock, 3-bit prescaler, 16-bit reload and overflow interrupt request.
- Compare/(reload/)capture register array consisting of four different kinds of registers:
  one 16-bit compare/reload/capture register,
  three 16-bit compare/capture registers,
  one 16-bit compare/capture register with additional "concurrent compare" feature,
  eight 16-bit compare registers with timer-overflow controlled loading.

In summary, the register array may control up to 21 output lines and can request up to 7 independent interrupts.

In the following text all double-byte compare, compare/capture or compare/reload/capture registers are called CMx (x = 0 … 7), CCx (x = 0 … 4) or CRC register, respectively.

The block diagram in **figure 6-13** shows the general configuration of the CCU. All CC1 to CC4 registers and the CRC register are exclusively assigned to timer 2. Each of the eight compare registers CM0 through CM7 can either be assigned to timer 2 or to the faster compare timer, e.g. to provide up to 8 PWM output channels. The assignment of the CMx registers - which can be done individually for every single register - is combined with an automatic selection of one of the two possible compare modes.

Port 5, port 4, and five lines of port 1 have alternate functions dedicated to the CCU. These functions are listed in **table 6-3**. Normally each register controls one dedicated output line at the ports. Register CC4 is an exception as it can manipulate up to nine output lines (one at port 1.4 and the other eight at port 5) concurrently. This function is referenced as "concurrent compare".

Note that for an alternate input function the port latch has to be programmed with a '1'. For bit latches of port pins that are used as compare outputs, the value to be written to the bit latches depends on the compare mode established.

A list of all special function registers concerned with the CCU is given in **table 6-4**.

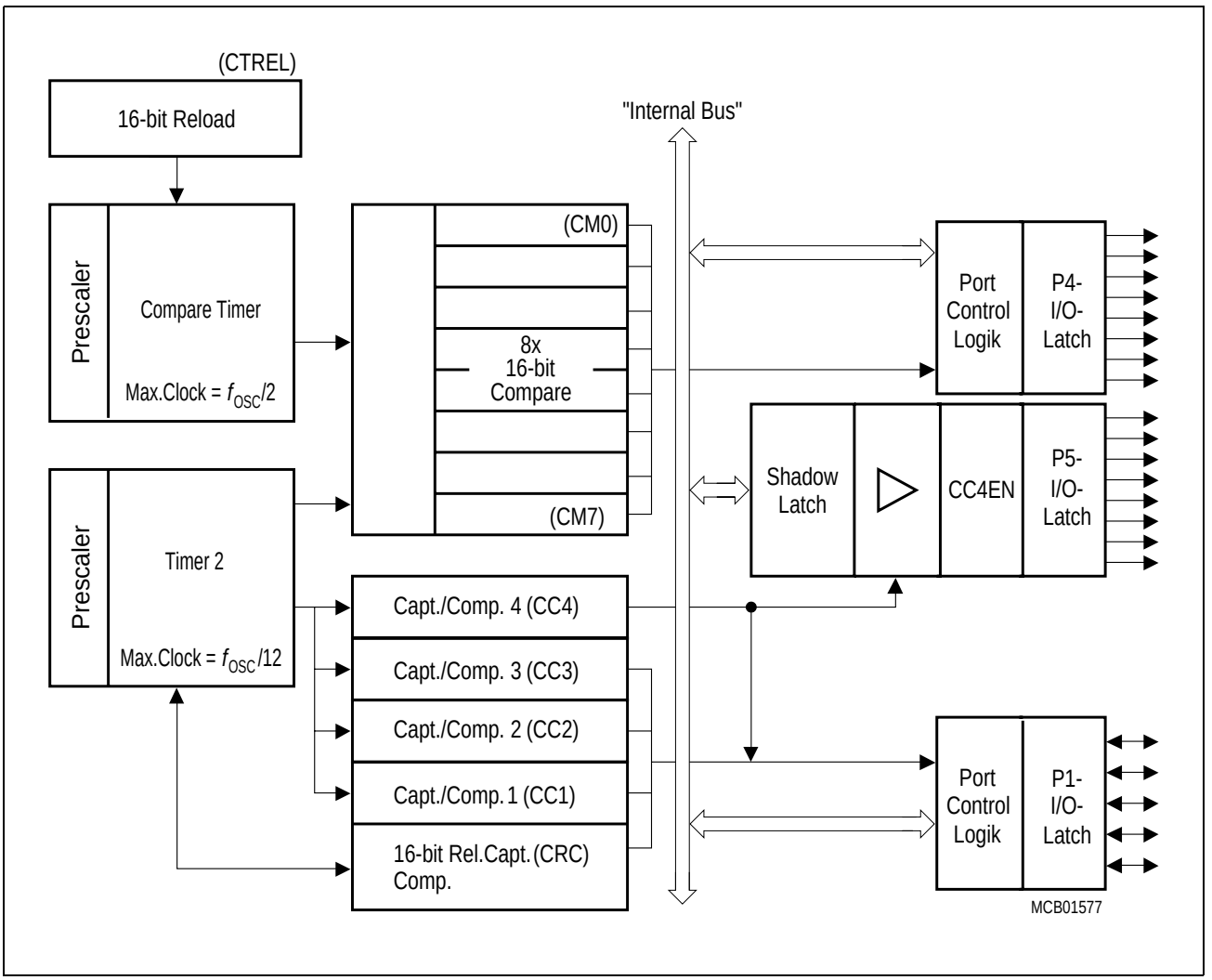


Figure 6-13
Block Diagram of the CCU

Table 6-3
Alternate Port Functions of the CCU

Pin Symbol	Pin No. (P-MQFP-100)	Pin No. (P-LCC-84)	Alternate Function
P1.0/$\overline{\text{INT3}}$/CC0	9	36	Compare output/capture input for CRC register
P1.1/INT4/CC1	8	35	Compare output/capture input for CC1 register
P1.2/INT5/CC2	7	34	Compare output/capture input for CC2 register
P1.3/INT6/CC3	6	33	Compare output/capture input for CC3 register
P1.4/$\overline{\text{INT2}}$/CC4	1	32	Compare output/capture input for CC4 register
P1.5/T2EX	100	31	Timer 2 external reload trigger input
P1.7/T2	98	29	Timer 2 external count/gate input
P4.0/CM0	64	1	Compare output for the CM0 register
P4.1/CM1	65	2	Compare output for the CM1 register
P4.2/CM2	66	3	Compare output for the CM2 register
P4.3/CM3	68	5	Compare output for the CM3 register
P4.4/CM4	69	6	Compare output for the CM4 register
P4.5/CM5	70	7	Compare output for the CM5 register
P4.6/CM6	71	8	Compare output for the CM6 register
P4.7/CM7	72	9	Compare output for the CM7 register
P5.0/CCM0	44	68	Concurrent compare 0 output
P5.1/CCM1	43	67	Concurrent compare 1 output
P5.2/CCM2	42	66	Concurrent compare 2 output
P5.3/CCM3	41	65	Concurrent compare 3 output
P5.4/CCM4	40	64	Concurrent compare 4 output
P5.5/CCM5	39	63	Concurrent compare 5 output
P5.6/CCM6	38	62	Concurrent compare 6 output
P5.7/CCM7	37	61	Concurrent compare 7 output

Table 6-4
Special Function Register of the CCU

Symbol	Description	Address
CCEN	Compare/Capture Enable Register	$C1_H$
CC4EN	Compare/Capture 4 Enable Register	$C9_H$
CCH1	Compare/Capture Register 1, High Byte	$C3_H$
CCH2	Compare/Capture Register 2, High Byte	$C5_H$
CCH3	Compare/Capture Register 3, High Byte	$C7_H$
CCH4	Compare/Capture Register 4, High Byte	CF_H
CCL1	Compare/Capture Register 1, Low Byte	$C2_H$
CCL2	Compare/Capture Register 2, Low Byte	$C4_H$
CCL3	Compare/Capture Register 3, Low Byte	$C6_H$
CCL4	Compare/Capture Register 4, Low Byte	CE_H
CMEN	Compare Enable Register	$F6_H$
CMH0	Compare Register 0, High Byte	$D3_H$
CMH1	Compare Register 1, High Byte	$D5_H$
CMH2	Compare Register 2, High Byte	$D7_H$
CMH3	Compare Register 3, High Byte	$E3_H$
CMH4	Compare Register 4, High Byte	$E5_H$
CMH5	Compare Register 5, High Byte	$E7_H$
CMH6	Compare Register 6, High Byte	$F3_H$
CMH7	Compare Register 7, High Byte	$F5_H$
CML0	Compare Register 0, Low Byte	$D2_H$
CML1	Compare Register 1, Low Byte	$D4_H$
CML2	Compare Register 2, Low Byte	$D6_H$
CML3	Compare Register 3, Low Byte	$E2_H$
CML4	Compare Register 4, Low Byte	$E4_H$
CML5	Compare Register 5, Low Byte	$E6_H$
CML6	Compare Register 6, Low Byte	$F2_H$
CML7	Compare Register 7, Low Byte	$F4_H$
CMSEL	Compare Input Select	$F7_H$
CRCH	Comp./Rel./Capt. Reg. High Byte	CB_H
CRCL	Comp./Rel./Capt. Reg. Low Byte	CA_H
COMSETL	Compare Set Register, Low Byte	$A1_H$
COMSETH	Compare Set Register, High Byte	$A2_H$
COMCLRL	Compare Clear Register, Low Byte	$A3_H$
COMCLRH	Compare Clear Register, High Byte	$A4_H$
SETMSK	Compare Set Mask Register	$A5_H$
CLRMSK	Compare Clear Mask Register	$A6_H$
CTCON	Compare Timer Control Register	$E1_H$
CTRELH	Compare Timer Rel. Reg., High Byte	DF_H
CTRELL	Compare Timer Rel. Reg., Low Byte	DE_H
TH2	Timer 2, High Byte	CD_H
TL2	Timer 2, Low Byte	CC_H
T2CON	Timer 2 Control Register	$C8_H$
IRCON0	Interrupt Control 0 Register	$C0_H$

6.3.1 Timer 2 Operation

Timer 2 is one of the two 16-bit timer units of the capture/compare unit. It can operate as timer, event counter, or gated timer. Prior to the description of the timer 2 operating modes and functions, the timer 2 related special function registers are described.

6.3.1.1 Timer 2 Registers

Timer 2 is controlled by bits of the 5 special function register T2CON, CTCON, IEN0, IEN1, and IRCON0. The related meaning of the timer 2 control bits and flags is shown below.

Special Function Register T2CON (Address $C8_H$) **Reset Value : 00_H**
Special Function Register CTCON (Address $E1_H$) **Reset Value : $0X000000_B$**
Special Function Register IEN0 (Address $A8_H$) **Reset Value : 00_H**
Special Function Register IEN1 (Address $B8_H$) **Reset Value : 00_H**
Special Function Register IRCON0 (Address $C0_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	CF_H	CE_H	CD_H	CC_H	CB_H	CA_H	$C9_H$	$C8_H$	
$C8_H$	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T1I0	T2CON
	7	6	5	4	3	2	1	0	
$E1_H$	T2PS1	–	ICR	ICS	CTF	CLK2	CLK1	CLK0	CTCON
	AF_H	AE_H	AD_H	AC_H	AB_H	AA_H	$A9_H$	$A8_H$	
$A8_H$	EAL	WDT	ET2	ES0	ET1	EX1	ET0	EX0	IEN0
	BF_H	BE_H	BD_H	BC_H	BB_H	BA_H	$B9_H$	$B8_H$	
$B8_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	IEN1
	$C7_H$	$C6_H$	$C5_H$	$C4_H$	$C3_H$	$C2_H$	$C1_H$	$C0_H$	
$C0_H$	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC	IRCON0

The shaded bits are not used for timer/counter 2.

<table>
<tr><th>Bit</th><th>Symbol</th></tr>
<tr><td>T2PS
T2PS1</td><td>Timer 2 prescaler select bits
Based on f_{OSC}/12 these bits define the prescaler divider ratio of the timer 2 input clock according the following table.
<table>
<tr><th>T2PS1</th><th>T2PS</th><th>Timer 2 Input Clock</th></tr>
<tr><td>0</td><td>0</td><td>$f_{OSC} \div 12$</td></tr>
<tr><td>0</td><td>1</td><td>$f_{OSC} \div 24$</td></tr>
<tr><td>1</td><td>0</td><td>$f_{OSC} \div 48$</td></tr>
<tr><td>1</td><td>1</td><td>$f_{OSC} \div 96$</td></tr>
</table></td></tr>
<tr><td>T2R1
T2R0</td><td>Timer 2 reload mode selection
<table>
<tr><th>T2R1</th><th>T2R0</th><th>Reload Mode</th></tr>
<tr><td>0</td><td>X</td><td>Reload disabled</td></tr>
<tr><td>1</td><td>0</td><td>Mode 0 : auto-reload upon timer 2 overflow (TF2)</td></tr>
<tr><td>1</td><td>1</td><td>Mode 1: reload upon falling edge at pin P1.5 / T2EX</td></tr>
</table></td></tr>
<tr><td>T2I1
T2I0</td><td>Timer 2 input selection
<table>
<tr><th>T2I1</th><th>T2I0</th><th>Input Mode</th></tr>
<tr><td>0</td><td>0</td><td>No input selected : timer 2 stops</td></tr>
<tr><td>0</td><td>1</td><td>Timer function : input frequency see table above</td></tr>
<tr><td>1</td><td>0</td><td>Counter function : external input controlled by pin P1.7 / T2</td></tr>
<tr><td>1</td><td>1</td><td>Gated timer function : input controlled by pin T2/P1.7</td></tr>
</table></td></tr>
<tr><td>ET2</td><td>Timer 2 Interrupt Enable.
If ET2 = 0, the timer 2 interrupt is disabled.</td></tr>
<tr><td>EXEN2</td><td>Timer 2 external reload interrupt enable
If EXEN2 = 0, the timer 2 external reload interrupt is disabled.
The external reload function is not affected by EXEN2.</td></tr>
<tr><td>EXF2</td><td>Timer 2 external reload flag
Set when a reload is caused by a negative transition on pin T2EX while EXEN2 = 1. If ET2 in IEN0 is set (timer 2 interrupt enabled), EXF2 = 1 will cause an interrupt. Can be used as an additional external interrupt when the reload function is not used. EXF2 must be cleared by software.</td></tr>
<tr><td>TF2</td><td>Timer 2 overflow flag
Set by a timer 2 overflow and must be cleared by software. If the timer 2 interrupt is enabled, TF2 = 1 will cause an interrupt.</td></tr>
</table>

Special Function Registers TL2/TH2 (Addresses CC_H/CD_H) Reset Value : $00_H/00_H$
Special Function Registers CRCL/CRCH (Addresses CA_H/CB_H) Reset Value : $00_H/00_H$

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
CC_H	.7	.6	.5	.4	.3	.2	.1	.0	TL2
CD_H	.7	.6	.5	.4	.3	.2	.1	.0	TH2
CA_H	.7	.6	.5	.4	.3	.2	.1	.0	CRCL
CB_H	.7	.6	.5	.4	.3	.2	.1	.0	CRCH

Bit	Function
TL2.7 - 0	Timer 2 low byte TL2 contains the 8-bit low byte of the 16-bit timer 2 count value.
TH2.7 - 0	Timer 2 high byte TH2 contains the 8-bit high byte of the 16-bit timer 2 count value.
CRCL.7 - 0	Compare/Reload/Capture register low byte CRCL is the 8-bit low byte of the 16-bit reload register of timer 2. It is also used for compare/capture functions.
CRCH.7 - 0	Compare/Reload/Capture register high byte CRCH is the 8-bit high byte of the 16-bit reload register of timer 2. It is also used for compare/capture functions.

Special Function Registers COMSETL (Address $A1_H$) Reset Value : 00_H
Special Function Registers COMSETH (Address $A2_H$) Reset Value : 00_H
Special Function Registers COMCLRL (Address $A3_H$) Reset Value : 00_H
Special Function Registers COMCLRH (Address $A4_H$) Reset Value : 00_H
Special Function Registers SETMSK (Address $A5_H$) Reset Value : 00_H
Special Function Registers CLRMSK (Address $A6_H$) Reset Value : 00_H

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$A1_H$	.7	.6	.5	.4	.3	.2	.1	.0	COMSETL
$A2_H$	.7	.6	.5	.4	.3	.2	.1	.0	COMSETH
$A3_H$	.7	.6	.5	.4	.3	.2	.1	.0	COMCLRL
$A4_H$	.7	.6	.5	.4	.3	.2	.1	.0	COMCLRH
$A5_H$	.7	.6	.5	.4	.3	.2	.1	.0	SETMSK
$A6_H$	.7	.6	.5	.4	.3	.2	.1	.0	CLRMSK

Bit	Function
COMSETL.7 - 0	Concurrent compare match set register low byte COMSETL contains the low byte of the 16-bit compare value for setting port 5 pins in concurrent compare mode.
COMSETH.7 - 0	Concurrent compare match set register high byte COMSETL contains the high byte of the 16-bit compare value for setting port 5 pins in concurrent compare mode.
COMCLRL.7 - 0	Concurrent compare match clear register low byte COMSETL contains the low byte of the 16-bit compare value for resetting port 5 pins in concurrent compare mode.
COMCLRH.7 - 0	Concurrent compare match clear register high byte COMSETL contains the high byte of the 16-bit compare value for resetting port 5 pins in concurrent compare mode.
SETMSK.7 - 0	Concurrent compare output set mask register If a bit in SETMSK is set, the corresponding port 5 pin is set in concurrent compare mode if a match of timer 2 and the COMSET registers occurs.
CLRMSK.7 - 0	Concurrent compare output clear mask register If a bit in CLRMSK is set, the corresponding port 5 pin is reset in concurrent compare mode if a match of timer 2 and the COMCLR registers occurs.

6.3.1.2 Timer 2 Operating Modes

Figure 6-14 shows a functional block diagram of the timer 2 unit.

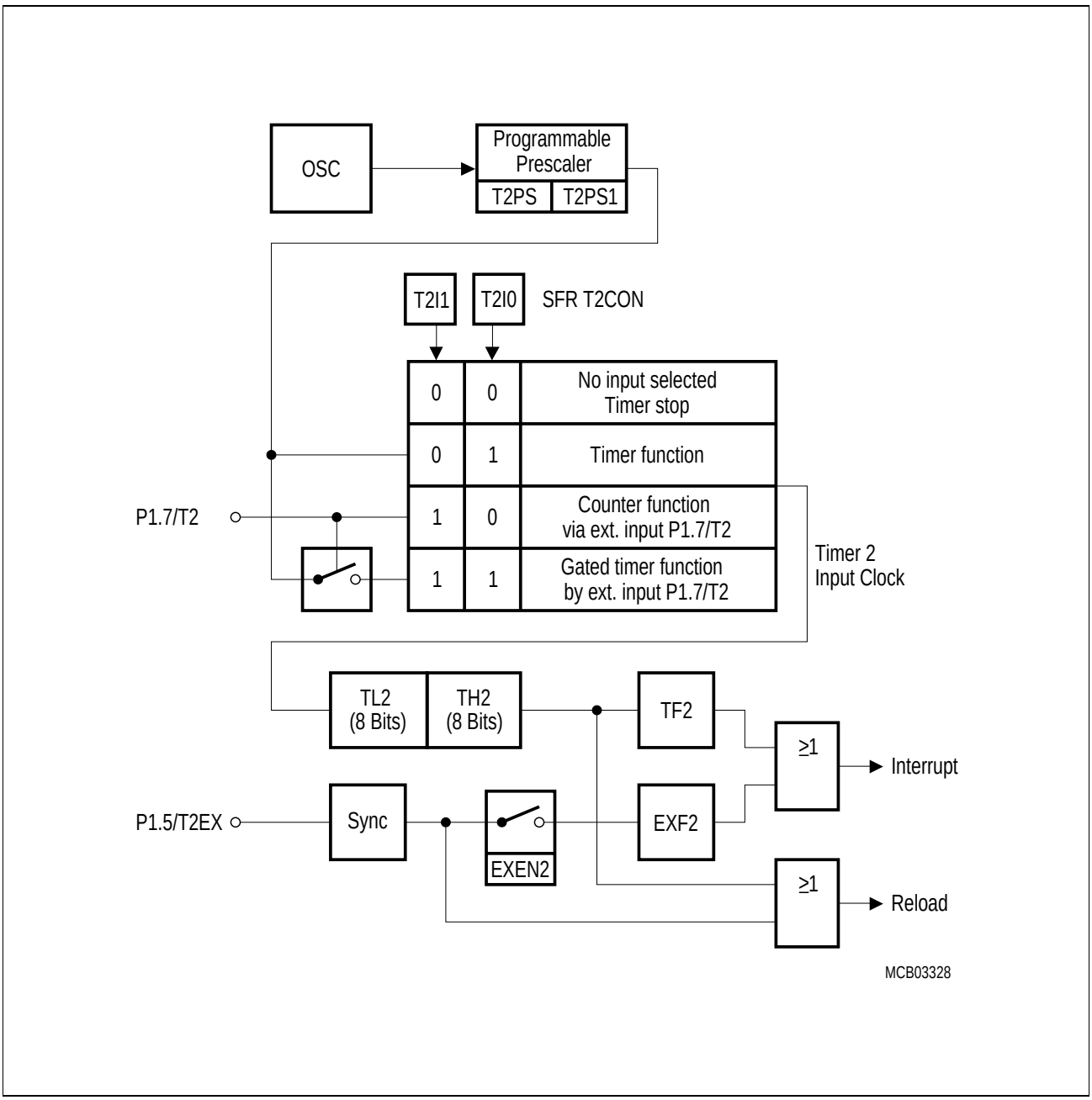


Figure 6-14
Block Diagram of Timer 2

Timer mode

In timer function, the count rate is derived from the oscillator frequency. A prescaler offers the possibility of selecting a count rate of 1/12 to 1/96 of the oscillator frequency. Thus, the 16-bit timer register (consisting of TH2 and TL2) is incremented at maximum in every machine cycle. The prescaler is selected by the bits T2PS1and T2PS.

6.3.1.2.1 Gated Timer Mode

In gated timer function, the external input pin P1.7/T2 operates as a gate to the input of timer 2. If T2 is high, the internal clock input is gated to the timer. T2 = 0 stops the counting procedure. This will facilitate pulse width measurements. The external gate signal is sampled once every machine cycle.

6.3.1.2.2 Event Counter Mode

In the event counter function, the timer 2 is incremented in response to a 1-to-0 transition at its corresponding external input pin P1.7/T2. In this function, the external input is sampled every machine cycle. When the sampled inputs show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the timer register in the cycle following the one in which the transition was detected. Since it takes two machine cycles (24 oscillator periods) 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 must be held stable for at least one full machine cycle.

Note: The prescaler must be turned off for proper counter operation of timer 2 (T2PS1=T2PS=0).

In either case, no matter whether timer 2 is configured as timer, event counter, or gated timer, a rolling-over of the count from all 1's to all 0's sets the timer overflow flag TF2 (bit 6 in SFR IRCON0, interrupt request control) which can generate an interrupt.

If TF2 is used to generate a timer overflow interrupt, the request flag must be cleared by the interrupt service routine as it could be necessary to check whether it was the TF2 flag or the external reload request flag EXF2 which requested the interrupt (for EXF2 see below). Both request flags cause the program to branch to the same vector address.

6.3.1.2.3 Reload of Timer 2

The reload mode for timer 2 (see **figure 6-15**) is selected by bits T2R0 and T2R1 in SFR T2CON. Two reload modes are selectable:

In mode 0, when timer 2 rolls over from all 1's to all 0's, it not only sets TF2 but also causes the timer 2 registers to be loaded with the 16-bit value in the CRC register, which is preset by software. The reload will happen in the same machine cycle in which TF2 is set, thus overwriting the count value 0000_H.

In mode 1, a 16-bit reload from the CRC register is caused by a negative transition at the corresponding input pin P1.5/T2EX. In addition, this transition will set flag EXF2, if bit EXEN2 in SFR IEN1 is set.

If the timer 2 interrupt is enabled, setting EXF2 will generate an interrupt. The external input pin T2EX is sampled in every machine cycle. When the sampling shows a high in one cycle and a low in the next cycle, a transition will be recognized. The reload of timer 2 registers will then take place in the cycle following the one in which the transition was detected.

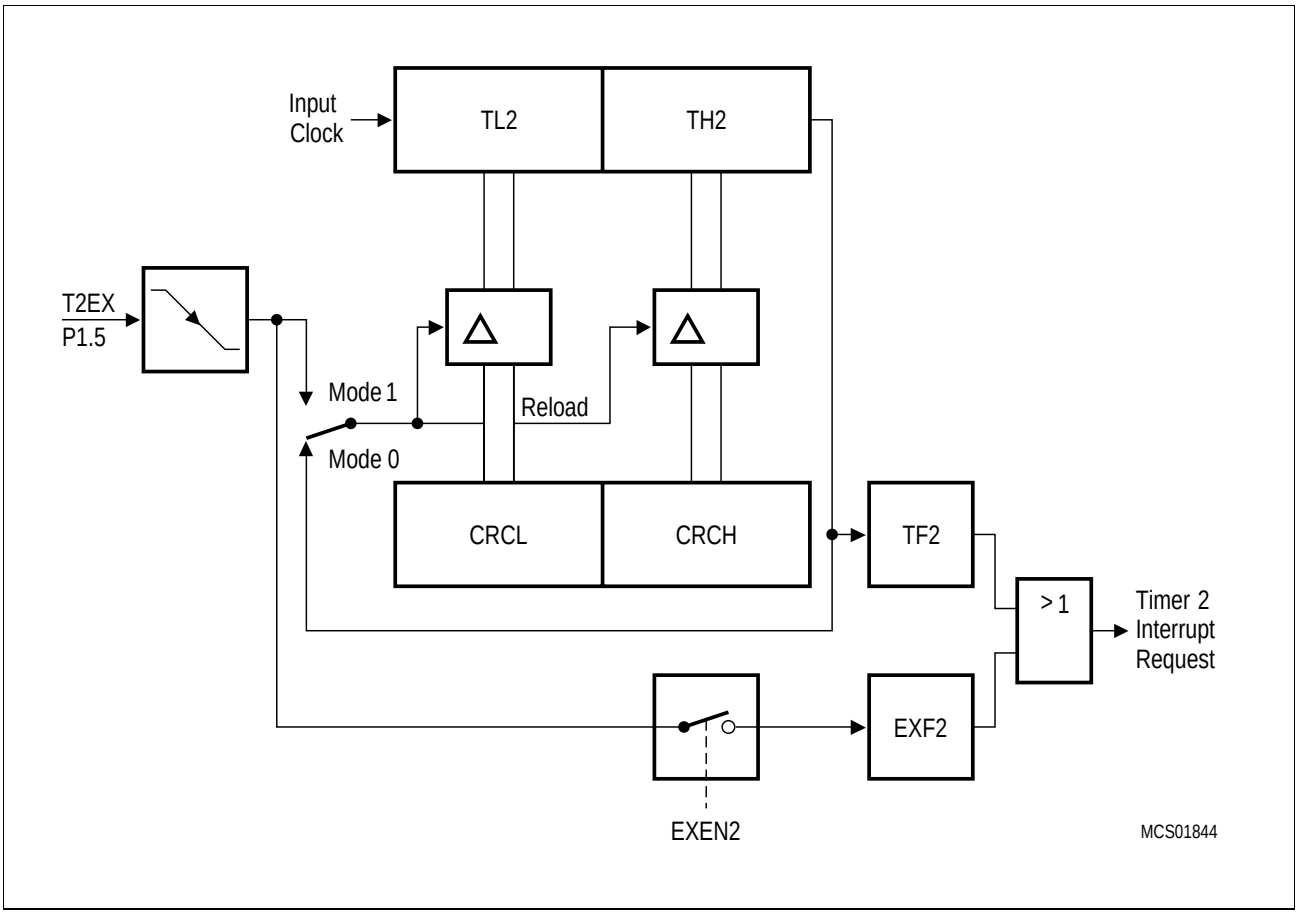


Figure 6-15
Timer 2 in Reload Mode

6.3.2 Operation of the Compare Timer

The compare timer operates as a fast 16-bit time base for the compare registers CM0 to CM7. The compare timer combined with the CMx registers and can be used for high-speed output puposes or as a fast 16-bit pulse-width modulation unit.

Prior to the description of the compare timer operating modes and functions, the compare timer related special function registers are described.

6.3.2.1 Compare Timer Registers

Each of the two compare timers has a 8-bit control control register and a 16-bit reload register. These 6 special function registers are described in this section.

Special Function Register CTCON (Address $E1_H$) **Reset Value : $0X000000_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$E1_H$	T2PS1	–	ICR	ICS	CTF	CLK2	CLK1	CLK0	CTCON

The shaded bits are not used for compare timer control.

Bit	Function
ICR	Interrupt request flag for compare register COMCLR ICR is set when a compare match occured. ICR is cleared ba hardware when the processor vectors to interrupt routine.
ICS	Interrupt request flag for compare register COMSET ICS is set when a compare match occured. ICS is cleared by hardware when the processor vectors to interrupt routine.
CTF	Compare timer overflow flag CTF is set when the compare timer 1 count rolls over from all ones to the reload value. When CTF is set, a compare timer interrupt can be generated (if enabled). CTF is cleared by hardware (RETI instruction) when the compare timer value is no more equal to the reload value.

Bit	Function
CLK2 CLK1 CLK0	Compare timer input clock selection. These bits define the prescaler divider ratio of the compare timer input clock according to the following table

CLK2	CLK1	CLK0	Compare Timer Input Clock
0	0	0	$f_{OSC} \div 2$
0	0	1	$f_{OSC} \div 4$
0	1	0	$f_{OSC} \div 8$
0	1	1	$f_{OSC} \div 16$
1	0	0	$f_{OSC} \div 32$
1	0	1	$f_{OSC} \div 64$
1	1	0	$f_{OSC} \div 128$
1	1	1	$f_{OSC} \div 256$

Special Function Register CTRELL (Address DE_H) **Reset Value : 00_H**
Special Function Register CTRELH (Address DF_H) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
DE_H	.7	.6	.5	.4	.3	.2	.1	.0	CTRELL
DF_H	.7	.6	.5	.4	.3	.2	.1	.0	CTRELH

Bit	Function
CTRELL.7 - 0	Compare timer reload value low part The CTRELL register holds the lower 8 bits of the 16-bit reload value for the compare timer.
CTRELH.7 - 0	Compare timer reload value high part The CTRELH register holds the upper 8 bits of the 16-bit reload value for the compare timer.

6.3.2.2 Operating Modes of the Compare Timers

The compare timer receives its input clock from a programmable prescaler which provides eight input frequencies, ranging from $f_{OSC}/2$ up to $f_{OSC}/256$. This configuration allows a very high flexibility concerning timer period length and input clock frequency. The prescaler ratio is selected by four bits in the special function registers CTCON. **Figure 6-16** shows the block diagram of the compare timer.

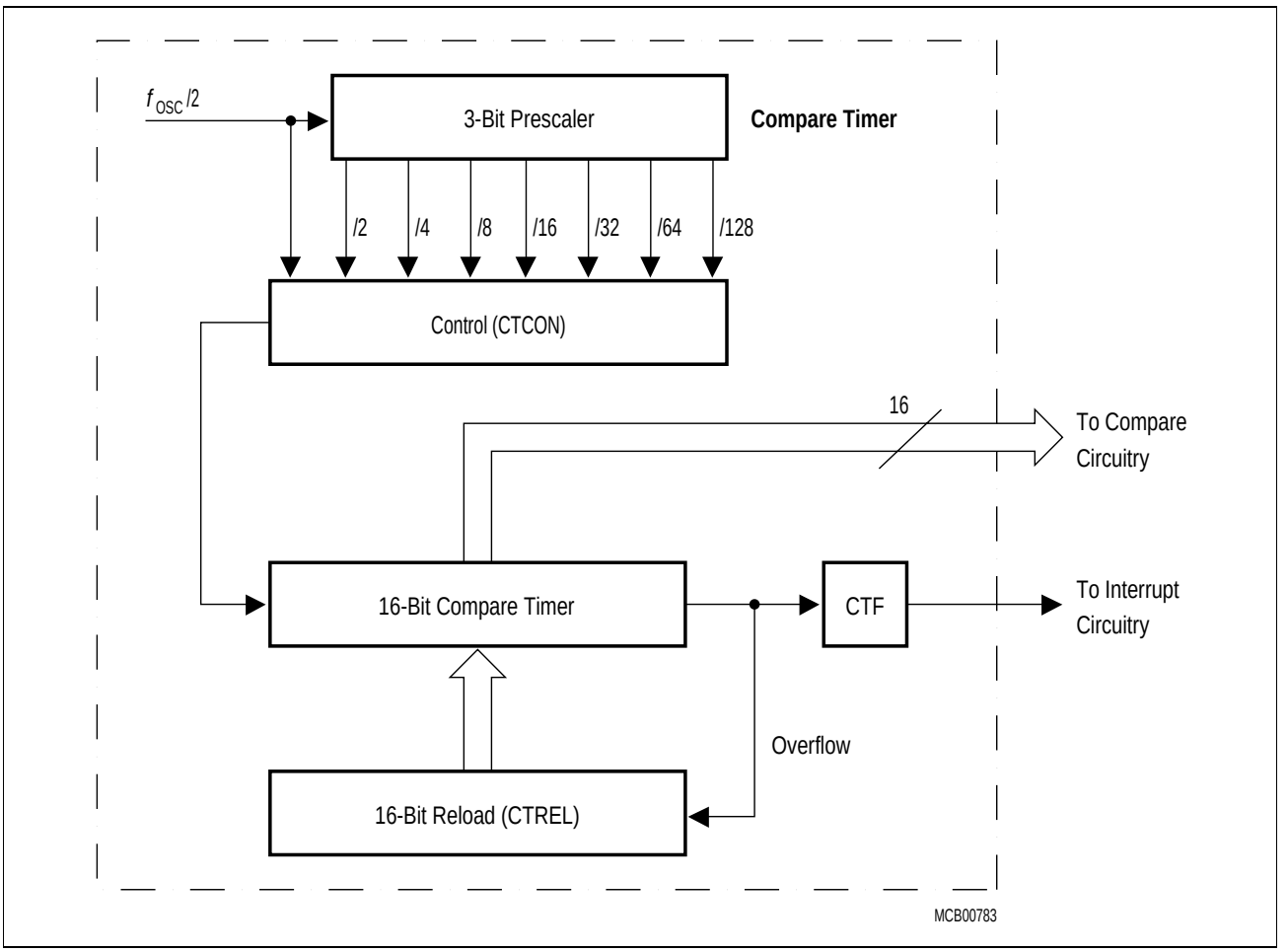


Figure 6-16
Compare Timer Block Diagram

The compare timer is, once started, a free-running 16-bit timer, which upon overflow is automatically reloaded by the content of the 16-bit reload register. This reload register is CTRELL (compare timer reload register, low byte) and CTRELH (compare timer reload register, high byte). An initial writing to the reload register CTRELL starts the corresponding compare timer. If a compare timer is already running, a write to CTRELL again triggers an instantly reload of the timer, in other words loads the timer in the cycle following the write instruction with the new count stored in the reload registers CTRELH/CTRELL.

When the reload register is to be loaded with a 16-bit value, the high byte of the reload register (CTRELH) must be written first to ensure a determined start or restart position. Writing to the low byte (CTRELL) then triggers the actual reload procedure mentioned above. The 16-bit reload value

can be overwritten at any time. Setting reload value to $FFFF_H$ will make the compare timer always equal to $FFFF_H$.

The compare timer has - as any other timer in the C517A - its own interrupt request flag CTF. This flag is located in register CTCON. CTF is set when the timer count rolls over from all ones to the reload value. CTF is reset by hardware when the compare timer value is no more equal to the reload value.

The compare timer overflow interrupt eases e.g. software control of pulse width modulated output signals. A periodic interrupt service routine caused by an overflow of the compare timer can be used to load new values in the assigned compare registers and thus change the corresponding PWM output accordingly. More details about interrupt control are discussed in chapter 7.

6.3.3 Compare Functions of the CCU

The compare function of a timer/register combination can be described as follows. The 16-bit value stored in a compare or compare/capture register is compared with the contents of the timer register. If the count value in the timer register matches the stored value, an appropriate output signal is generated at a corresponding port pin.

The contents of a compare register can be regarded as ’time stamp’ at which a dedicated output reacts in a predefined way (either with a positive or negative transition). Variation of this ’time stamp’ somehow changes the wave of a rectangular output signal at a port pin. This may - as a variation of the duty cycle of a periodic signal - be used for pulse width modulation as well as for a continually controlled generation of any kind of square wave forms. In the case of the C517A, two compare modes are implemented to cover a wide range of possible applications.

In the C517A - thanks to the high number of 13 compare registers and two associated timers - several timer/compare register combinations are selectable. In some of these configurations one of the two compare modes may be freely selected. Others, however, automatically establish a compare mode. In the following the two possible modes are generally discussed. This description will be referred to in later sections where the compare registers are described.

6.3.3.1 Compare Mode 0

In mode 0, upon matching the timer and compare register contents, the output signal changes from low to high. It goes back to a low level on timer overflow. As long as compare mode 0 is enabled, the appropriate output pin is controlled by the timer circuit only, and not by the user. Writing to the port will have no effect. **Figure 6-17** shows a functional diagram of a port circuit when used in compare mode 0. The port latch is directly controlled by the timer overflow and compare match signals. The input line from the internal bus and the write-to-latch line of the port latch are disconnected when compare mode 0 is enabled.

Compare mode 0 is ideal for generating pulse width modulated output signals, which in turn can be used for digital-to-analog conversion via a filter network or by the controlled device itself (e.g. the inductance of a DC or AC motor). Compare mode 0 may also be used for providing output clocks with initially defined period and duty cycle. This is the mode which needs the least CPU time. Once set up, the output goes on oscillating without any CPU intervention. **Figure 6-18** illustrates the function of compare mode 0.

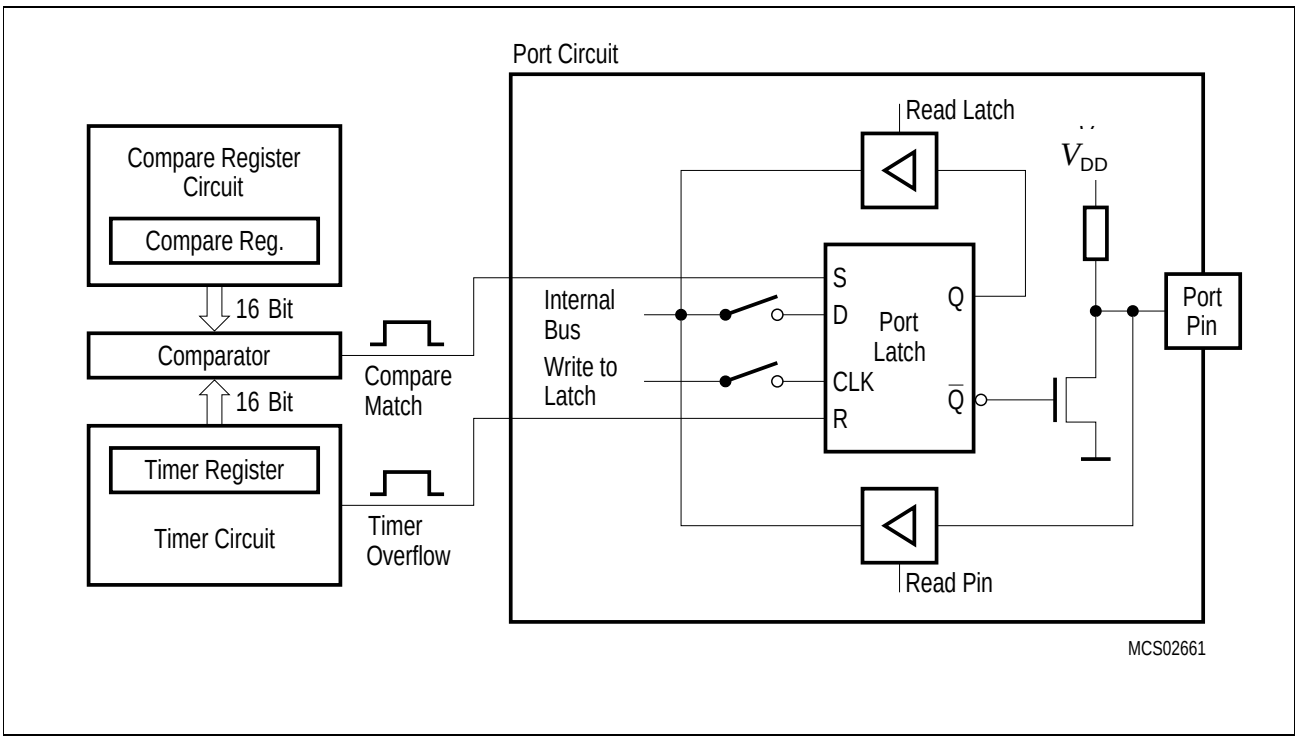


Figure 6-17
Port Latch in Compare Mode 0

Figure 6-18 shows a typical output signal waveform which is generated when compare mode 0 is selected.

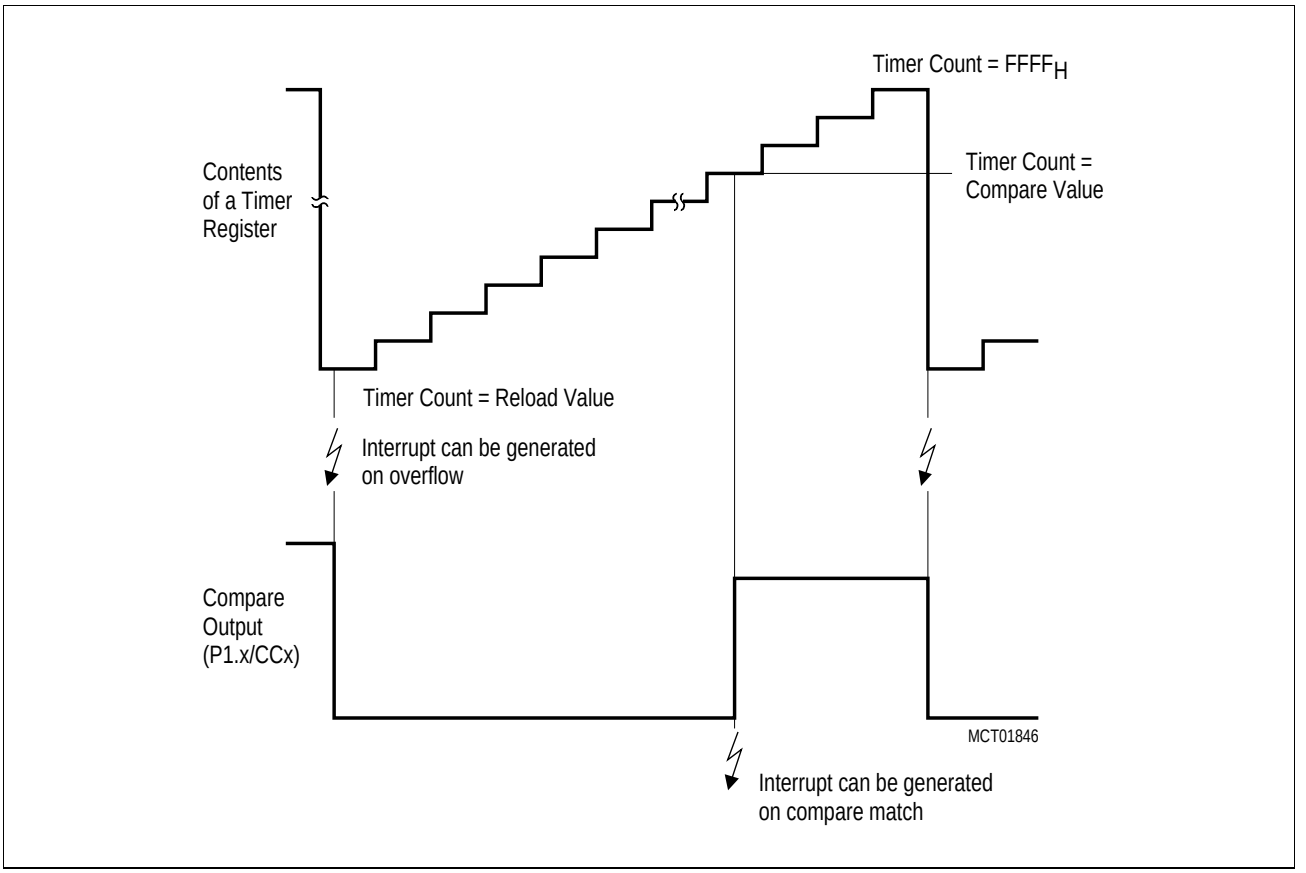


Figure 6-18
Output Waveform of Compare Mode 0

Modulation Range of a PWM Signal and Differences between the Two Timer/Compare Register Configurations in the CCU

There are two timer/compare register configurations in the CCU which can operate in compare mode 0 (either timer 2 with a CCx (CRC and CC1 to CC4) register or the compare timer with a CMx register). They basically operate in the same way, but show some differences concerning their modulation range when used for PWM.

Generally it can be said that for every PWM generation with n-bit wide compare registers there are 2^n different settings for the duty cycle. Starting with a constant low level (0% duty cycle) as the first setting, the maximum possible duty cycle then would be

$$(1 - 1/2^n) \times 100\ \%$$

This means that a variation of the duty cycle from 0% to real 100% can never be reached if the compare register and timer register have the same length. There is always a spike which is as long as the timer clock period.

In the C517A there are two different modulation ranges for the above mentioned two timer/compare register combinations. The difference is the location of the above spike within the timer period: at the end of a timer period or at the beginning plus the end of a timer period. Please refer to the description of the CCU relevant timer/register combinations in section 6.3.4 for details.

6.3.3.2 Compare Mode 1

In compare mode 1, the software adaptively determines the transition of the output signal. This mode can only be selected for compare registers assigned to timer 2. It is commonly used when output signals are not related to a constant signal period (as in a standard PWM generation) but must be controlled very precisely with high resolution and without jitter. In compare mode 1, both transitions of a signal can be controlled. Compare outputs in this mode can be regarded as high speed outputs which are independent of the CPU activity.

If compare mode 1 is enabled and the software writes to the appropriate output latch at the port, the new value will not appear at the output pin until the next compare match occurs. Thus, it can be choosen whether the output signal has to make a new transition (1-to-0 or 0-to-1, depending on the actual pin-level) or should keep its old value at the time when the timer value matches the stored compare value.

Figure 6-19 shows a functional diagram of a port cicuit configuration in compare mode 1. In this mode the port circuit consists of two separate latches. One latch (which acts as a "shadow latch") can be written under software control, but its value will only be transferred to the port latch (and thus to the port pin) when a compare match occurs.

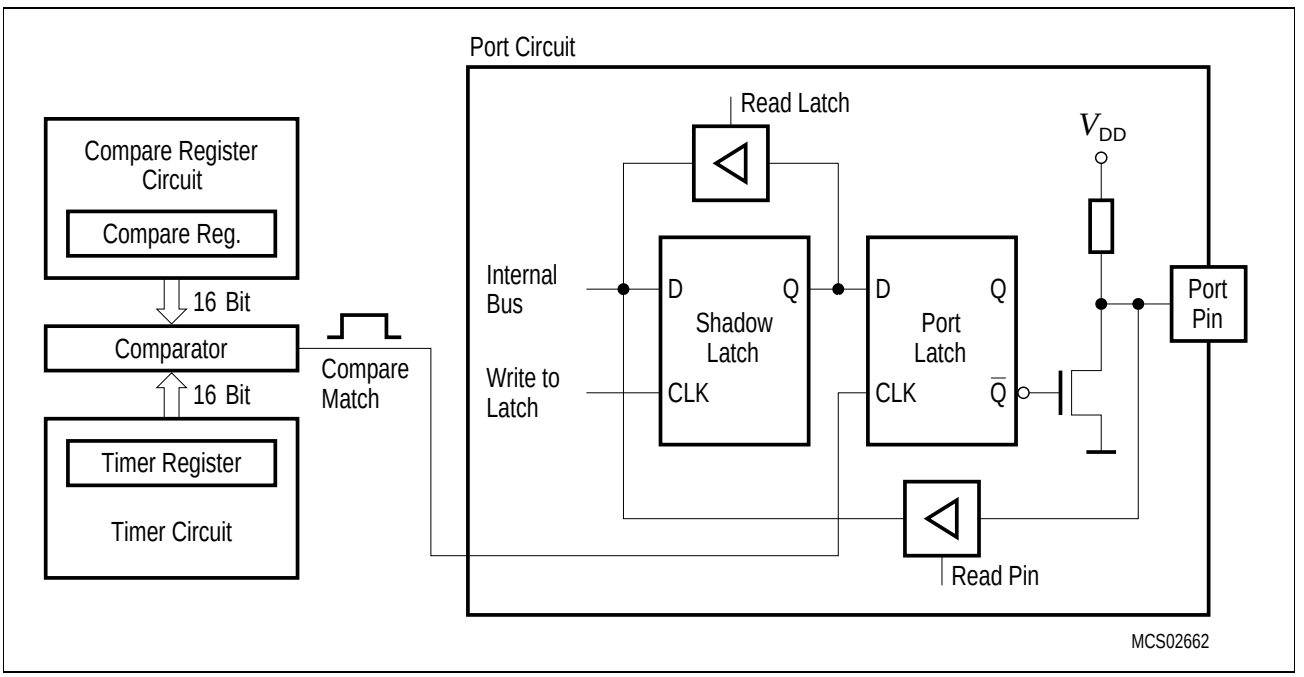


Figure 6-19
Compare Function of Compare Mode 1

Note that the double latch structure is transparent as long as the internal compare signal is active. While the compare signal is active, a write operation to the port will then change both latches. This may become important when timer 2 is driven with a slow input clock. In this case the compare signal could be active for many machine cycles in which the CPU could unintentionally change the contents of the port latch.

A read-modify-write instruction will read the user-controlled "shadow latch" and write the modified value back to this "shadow-latch". A standard read instruction will - as usual - read the pin of the corresponding compare output.

6.3.3.3 Compare Mode 2

In the compare mode 2 in the CCU can be used for the concurrent compare outputs at port 5. In this compare mode 2 the port 5 pins are no longer general purpose I/O pins or under control of compare/ capture register CC4, but under control of the compare registers COMSET and COMCLR. These both 16-bit registers are always associated with timer 2 (same as CRC, CC1 to CC4).

In compare mode 2 the concurrent compare output pins on port 5 are used as shown in **figure 6-20**.

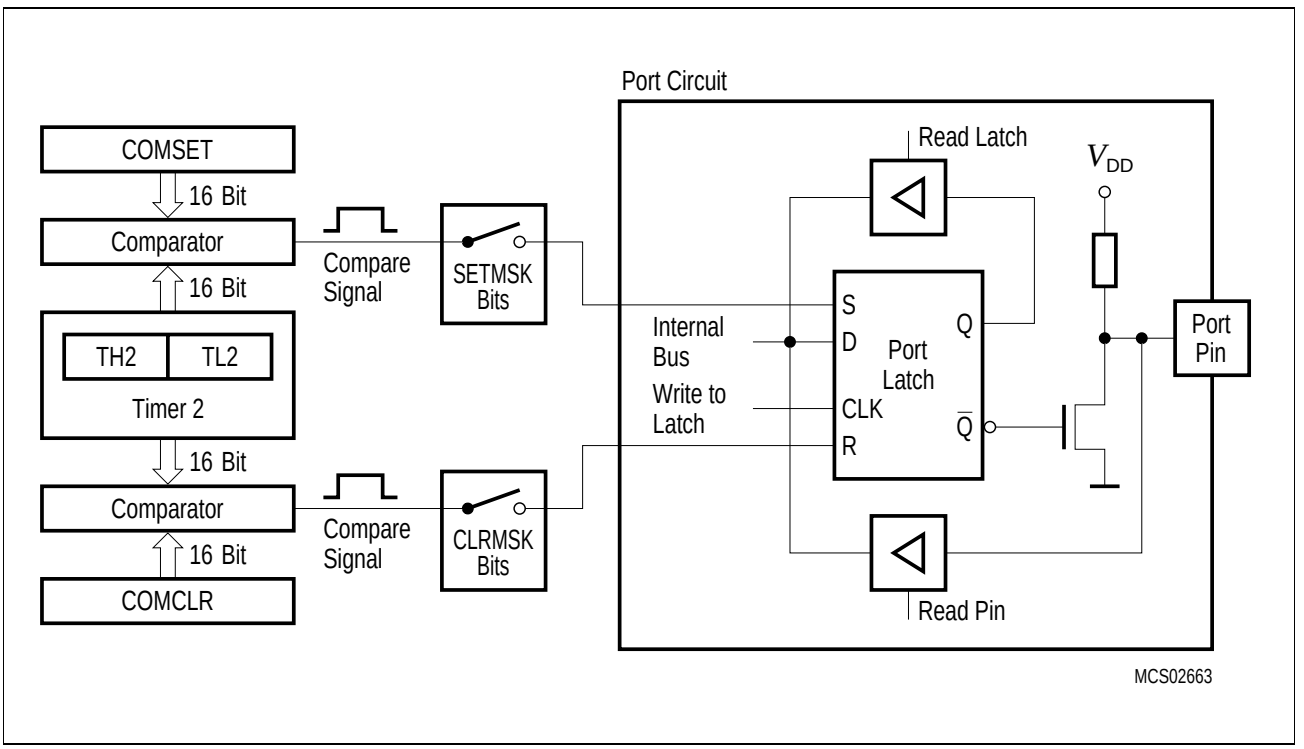


Figure 6-20
Compare Function of Compare Mode 2

- When a compare match occurs with register COMSET, a high level appears at the pins of port 5 when the corresponding bits in the mask register SETMSK are set.
- When a compare match occurs with register COMCLR, a low level appears at the pins of port 5 when the corresponding bits in the mask register CLRMSK are set.
- Additionally, the port 5 pins which are used for compare mode 2 can also be directly written using write instructions to SFR P5. Further, the pins can also be read under program control.

If compare mode 2 shall be selected, register CC4 must operate in compare mode 1 (with the corresponding output pin P1.4); Therefore, compare mode 2 is selected by enabling the compare function for register CC4 (COCAH4=1; COCAL4=0 in SFR CC4EN) and by programming bits COCOEN0 and COCOEN1 in SFR CC4EN. Like in concurrent compare mode associated with CC4, the number of port pins at P5 which serve the compare output function can be selected by bits COCON0-COCON2 (in SFR CC4EN). If a set and reset request occurs at the same time (identical values in COMSET and COMCLR), the set operation takes precedence. It is also possible to use only the interrupts which are generated by matches in COMSET and COMCLR without affecting port 5 ("software compare"). For this "interrupt-only" mode it is not necessary that the compare function at CC4 is selected.

6.3.4 Timer- and Compare-Register Configurations of the CCU

The compare function and the reaction of the corresponding outputs depend on the timer/compare register combination. **Table 6-5** shows the possible configurations of the CCU and the corresponding compare modes which can be selected. The following sections describe the function of these configurations.

Table 6-5
CCU Configurations

Assigned Timer	Compare Register	Compare Output at	Possible Modes
Timer 2	CRCH/CRCL CCH1/CCL1 CCH2/CCL2 CCH3/CCL3 CCH4/CCL4	P1.0/$\overline{\text{INT3}}$/CC0 P1.1/INT4/CC1 P1.2/INT5/CC2 P1.3/INT6/CC3 P1.4/$\overline{\text{INT2}}$/CC4	Compare mode 0, 1 + Reload Compare mode 0, 1 / capture Compare mode 0, 1 / capture Compare mode 0, 1 / capture Compare mode 0, 1 / capture (see 6.3.4.1 and 6.3.4.2)
	CCH4/CCL4	P1.4/$\overline{\text{INT2}}$/CC4 P5.0/CCM0 to P5.7/CCM7	Compare mode 1 “Concurrent compare“ (see 6.3.4.3)
	CMH0/CML0 to CMH7/CML7	P4.0/CM0 to P4.7/CM7	Compare mode 0 (see 6.3.4.4 and 6.3.4.4.2)
	COMSET COMCLR	P5.0/CCM0 to P5.7/CCM7	Compare mode 2 (see 6.3.4.5)
Compare Timer	CMH0/CML0 to CMH7/CML7	P4.0/CM0 to P4.7/CM7	Compare mode 1 (see 6.3.4.4 and 6.3.4.4.1)

6.3.4.1 Timer 2 - Compare Function with Registers CRC, CC1 to CC4

The compare function of registers CRC and CC1 to CC3 is completely compatible with the corresponding function of the C515/C515A. Registers CRC, CC1 to CC3 are permanently connected to timer 2. All four registers are multifunctional as they additionally provide a capture or a reload capability (CRC register only).

A general selection of the compare/capture function is done in register CCEN. For compare function they can be used in compare mode 0 or 1, respectively. The compare mode is selected by setting or clearing bit T2CM in special function register T2CON. Always two bits in register CCEN select the CRC and CC1 to CC3 register functionality which are :

– Disable compare/capture mode (normal I/O at the pin)
– Capture enabled on rising edge at a pin
– Compare enabled (pin becomes a compare output)
– Capture enabled on a write operation into the low part register of CRC or CC1 to CC3

Special Function Register T2CON (Address $C8_H$) **Reset Value : 00_H**
Special Function Register CCEN (Address $C1_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	CF_H	CE_H	CD_H	CC_H	CB_H	CA_H	$C9_H$	$C8_H$	
$C8_H$	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T1I0	T2CON

	7	6	5	4	3	2	1	0	
$C1_H$	COCAH3	COCAL3	COCAH2	COCAL2	COCAH1	COCAL1	COCAH0	COCAL0	CCEN

The shaded bits are not used for compare/capture control.

Bit	Function
T2CM	Compare mode control for CCR and CC1 to CC3 registers When T2CM is cleared, compare mode 0 is selected. When T2CM is set, compare mode 1 is selected.
COCAH3, COCAL3	Compare/capture mode for the CC register 3

COCAH3	COCAL3	Mode
0	0	Compare/capture disabled
0	1	Capture on rising edge at pin P1.3/INT6/CC3
1	0	Compare enabled
1	1	Capture on write operation into register CCL3

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>COCAH2,
COCAL2</td><td>Compare/capture mode for the CC register 2
<table>
<tr><th>COCAH2</th><th>COCAL2</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>Compare/capture disabled</td></tr>
<tr><td>0</td><td>1</td><td>Capture on rising edge at pin P1.2/INT5/CC2</td></tr>
<tr><td>1</td><td>0</td><td>Compare enabled</td></tr>
<tr><td>1</td><td>1</td><td>Capture on write operation into register CCL2</td></tr>
</table></td></tr>
<tr><td>COCAH1,
COCAL1</td><td>Compare/capture mode for the CC register 2
<table>
<tr><th>COCAH1</th><th>COCAL1</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>Compare/capture disabled</td></tr>
<tr><td>0</td><td>1</td><td>Capture on rising edge at pin P1.1/INT4/CC1</td></tr>
<tr><td>1</td><td>0</td><td>Compare enabled</td></tr>
<tr><td>1</td><td>1</td><td>Capture on write operation into register CCL1</td></tr>
</table></td></tr>
<tr><td>COCAH0,
COCAL0</td><td>Compare/capture mode for the CC register 2
<table>
<tr><th>COCAH0</th><th>COCAL0</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>Compare/capture disabled</td></tr>
<tr><td>0</td><td>1</td><td>Capture on rising edge at pin P1.0/$\overline{\text{INT3}}$/CC0</td></tr>
<tr><td>1</td><td>0</td><td>Compare enabled</td></tr>
<tr><td>1</td><td>1</td><td>Capture on write operation into register CRCL</td></tr>
</table></td></tr>
</table>

Figure 6-21 and **6-22** show the general timer/compare register/port latch configuration for registers CRC and CC1 to CC4 in compare mode 0 and compare mode 1. It also shows the interrupt capabilities. The compare interrupts of registers CRC and CC4 can be programmed to be either negative or positive transition activated. Compare interrupts for the CC1 to CC3 registers are always positive transition activated.

The compare function of CC4 is described in section 6.3.4.3.

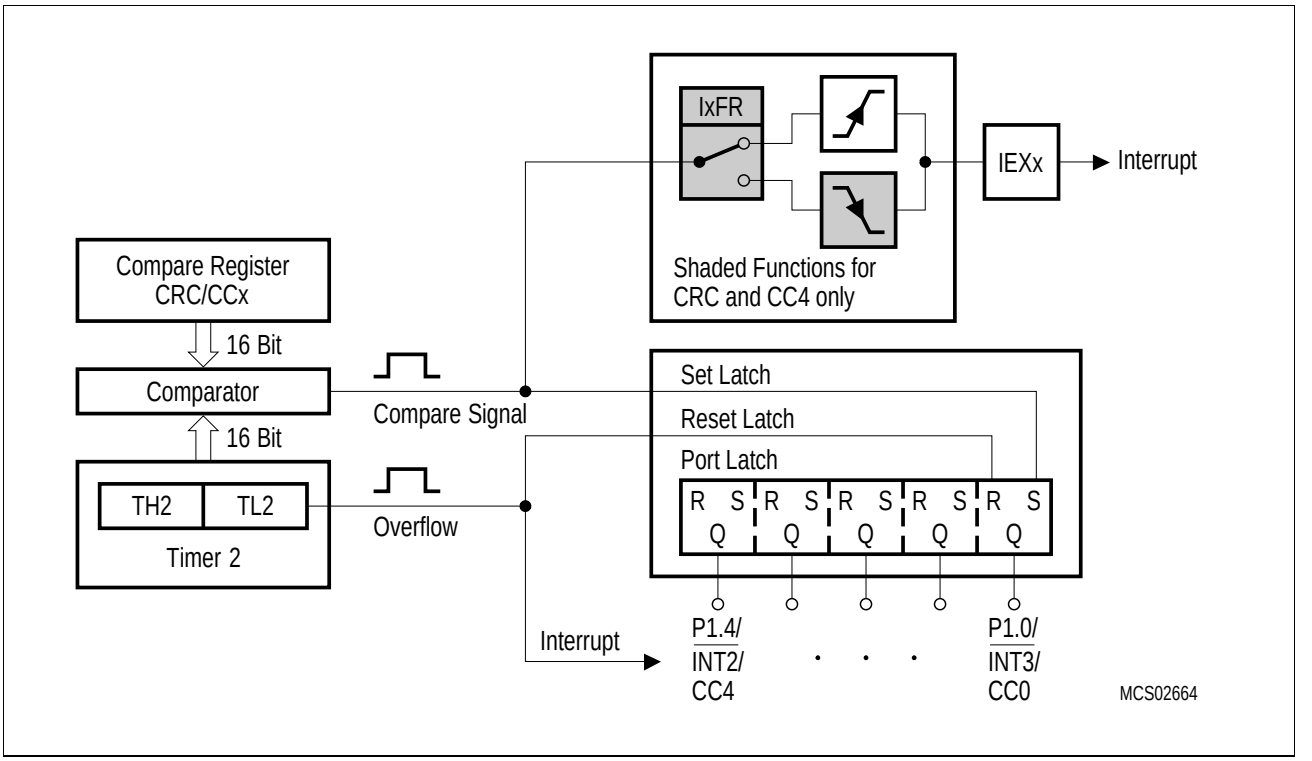


Figure 6-21
Timer 2 with Registers CRC and CC1 to CC4 in Compare Mode 0

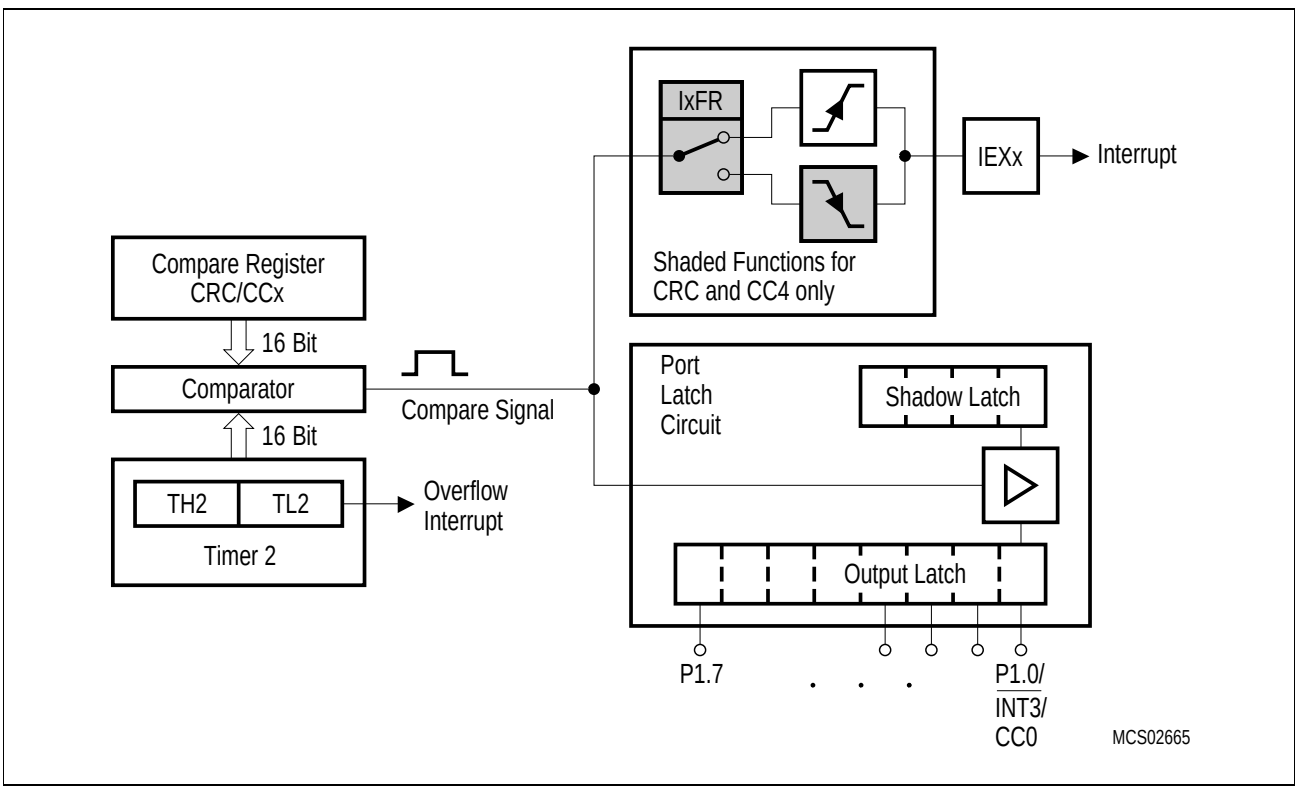


Figure 6-22
Timer 2 with Registers CRC and CC1 to CC3 in Compare Mode 1

6.3.4.2 Timer 2 - Capture Function with Registers CRC, CC1 to CC4

Each of the four compare/capture registers CC1 to CC4 and the CRC register can be used to latch the current 16-bit value of the timer 2 registers TL2 and TH2. Two different modes are provided for the capture function. In capture mode 0, an external event latches the timer 2 contents to a dedicated capture register. In capture mode 1, a capture event will occur when the low order byte of the dedicated 16-bit capture register is written to. This capture mode is provided to allow the software to read the timer 2 contents "on-the-fly".

In capture mode 0, the external event causing a capture is

- for CC1 to CC3 registers : A positive transition at pins CC1 to CC3 of port 1
- for the CRC and CC4 register : A positive or negative transition at the corresponding pins, depending on the status of the bits I3FR and I2FR in SFR T2CON. If the edge flags are cleared, a capture occurs in response to a negative transition; if the edge flags are set a capture occurs in response to a positive transition at pins P1.0/ $\overline{\text{INT3}}$/ CC0 and P1.4/ $\overline{\text{INT2}}$/ CC4.

In both cases the appropriate port 1 pin is used as input and the port latch must be programmed to contain a one (1). The external input is sampled in every machine cycle. When the sampled input shows a low (high) level in one cycle and a high (low) in the next cycle, a transition is recognized. The timer 2 content is latched to the appropriate capture register in the cycle following the one in which the transition was identified.

In capture mode 0 a transition at the external capture inputs of registers CC0 to CC4 will also set the corresponding external interrupt request flags IEX2 to IEX6. If the interrupts are enabled, an external capture signal will cause the CPU to vector to the appropriate interrupt service routine.

In capture mode 1 a capture occurs in response to a write instruction to the low order byte of a capture register. The write-to-register signal (e.g. write-to-CRCL) is used to initiate a capture. The value written to the dedicated capture register is irrelevant for this function. The timer 2 contents will be latched into the appropriate capture register in the cycle following the write instruction. In this mode no interrupt request will be generated.

Figures 6-23 and **6-24** show functional diagrams of the capture function of timer 2. **Figure 6-23** illustrates the operation of the CRC or CC4 register, while **figure 6-24** shows the operation of the compare/capture registers CC1 to CC3.

The two capture modes are selected individually for each capture register by bits in SFR CCEN (compare/capture enable register) and CC4EN (compare/capture 4 enable register). That means, in contrast to the compare modes, it is possible to simultaneously select capture mode 0 for one capture register and capture mode 1 for another register.

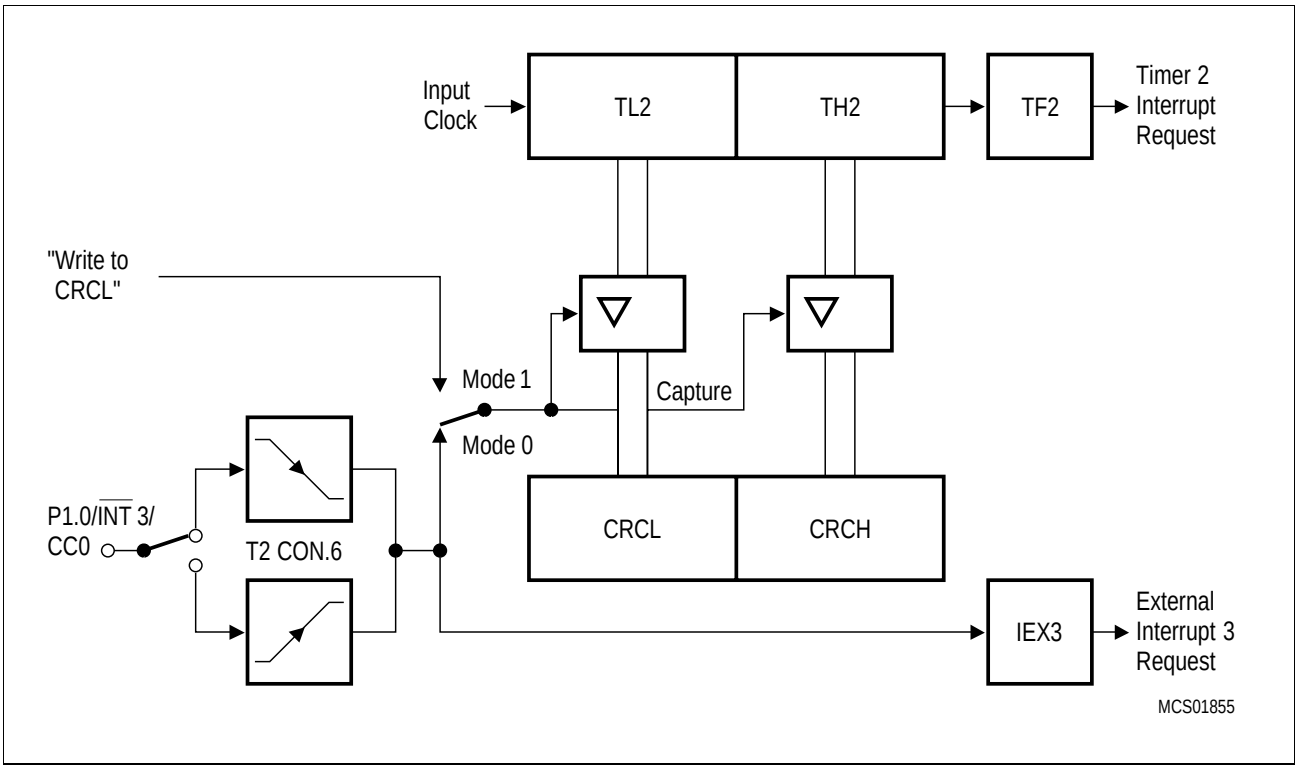


Figure 6-23
Capture with Registers CRC, CC4

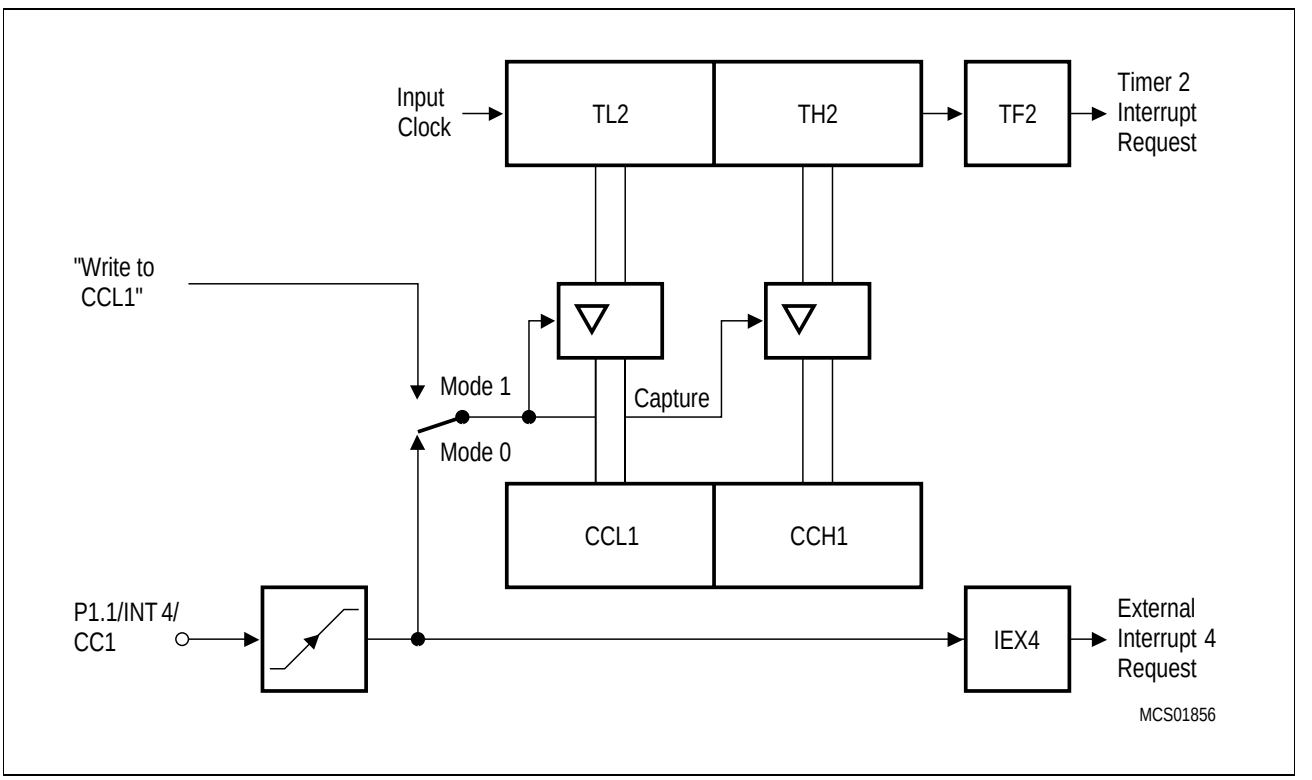


Figure 6-24
Capture with Registers CC1 to CC3

6.3.4.3 Compare Function of Register CC4; "Concurrent Compare"

Compare register CC4 is permanently assigned to timer 2. It has its own compare/capture enable register CC4EN. Register CC4 can be set to operate as any of the other CC registers (see also figures **6-25** and **6-26**). Its output pin is P1.4/$\overline{\text{INT2}}$/CC4 and it has a dedicated compare mode select bit COMO located in register CC4EN.

In addition to the standard operation in compare mode 0 or 1, there is another feature called 'concurrent compare' which is just an application of compare mode 1 to more than one output pin. Concurrent compare means that the comparison of CC4 and timer 2 can manipulate up to nine port pins concurrently. A standard compare register in compare mode 1 normally transfers a preprogrammed signal level, which is stored in the shadow latch to a single output line. Register CC4, however, is able to put a 9-bit pattern to nine output lines. The nine output lines consist of one line at port 1 (P1.4), which is the standard output for register CC4, and additional eight lines at port 5 (see **figure 6-25**).

Concurrent compare is an ideal and effective option where more than one synchronous output signal is to be generated. Applications including this requirement could among others be a complex multiple-phase stepper motor control as well as the control of ignition coils of a car engine. All these applications have in common that predefined bit-patterns must be put to an output port at a precisely predefined moment. This moment refers to a special count of timer 2, which was loaded to compare register CC4.

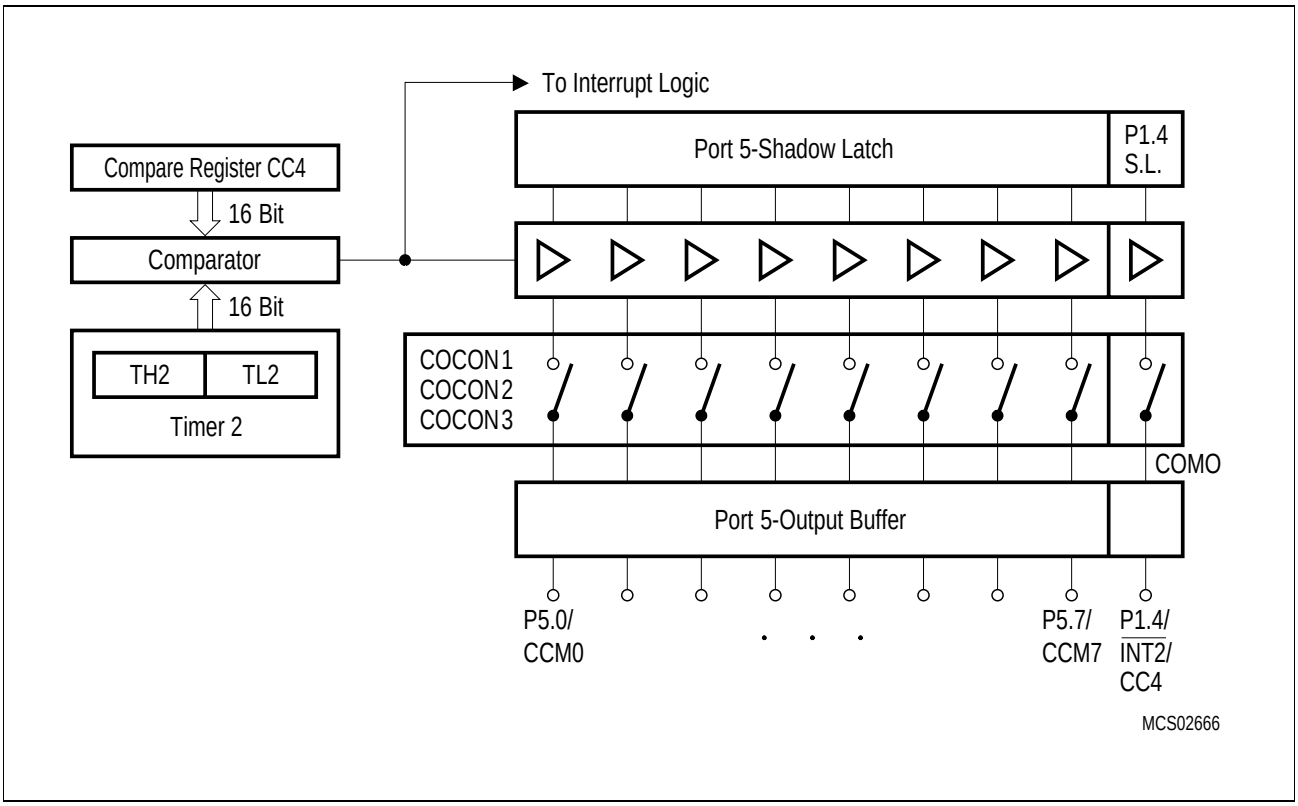


Figure 6-25
"Concurrent Compare" Function of Register CC4

Figure 6-26 gives an example of how to generate eight different rectangular wave forms at port 5 using a pattern table and a time schedule for these patterns. The patterns are moved into port 5 before the corresponding timer count is reached. The (future) timer count at which the pattern shall appear at the port must be loaded to register CC4. Thus the user can mask each port bit differently depending on whether he wants the output to be changed or not.

Concurrent compare is enabled by setting bit COCOEN in special function register CC4EN. A ’1’ in this bit automatically sets compare mode 1 for register CC4, too. A 3-bit field in special function register CC4EN determines the additional number of output pins at port 5. Port P1.4/$\overline{\text{INT2}}$/CC4 is used as a standard output pin in any compare mode for register CC4.

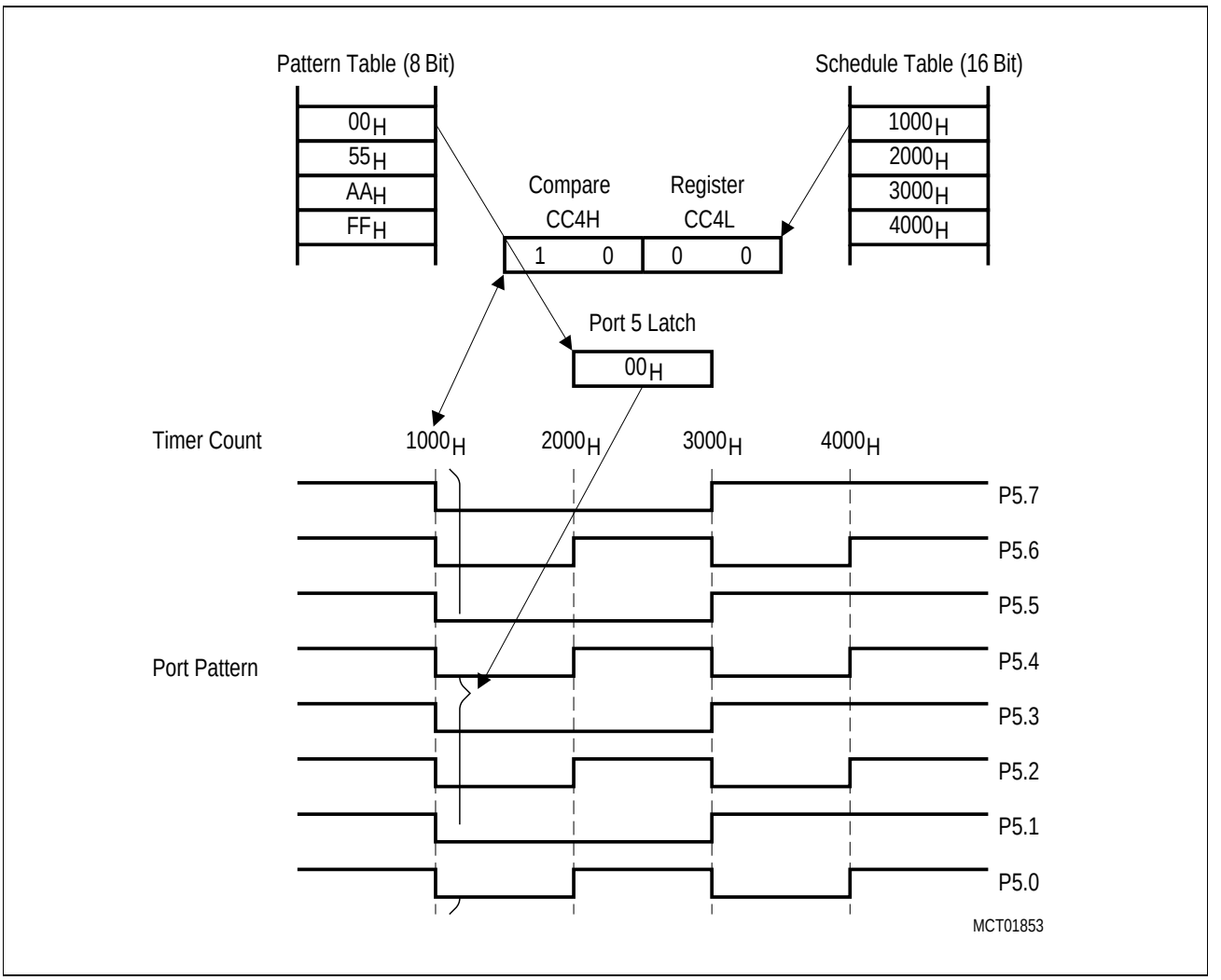


Figure 6-26
Example for a "Concurrent Compare" Waveform at Port 5

Special Function Register CC4EN (Address $C9_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$C9_H$	COCO EN1	COCO N2	COCO N1	COCO N0	COCO EN0	COCAH4	COCAL4	COMO	CC4EN

Bit	Function
COCOEN1 COCOEN0	Selection of compare modes 1 and 2 at port 5 For details on mode selection with COCOEN1/COCOEN0 see **table 6-6**.
COCON2 COCON1 COCON0	Port 5 compare outputs selection These bits select the number of compare outputs at port 5 according the following table.

COCON2	COCON1	COCON0	Function
0	0	0	One additional output of CC4 at P5.0
0	0	1	Additional outputs of CC4 at P5.0 to P5.1
0	1	0	Additional outputs of CC4 at P5.0 to P5.2
0	1	1	Additional outputs of CC4 at P5.0 to P5.3
1	0	0	Additional outputs of CC4 at P5.0 to P5.4
1	0	1	Additional outputs of CC4 at P5.0 to P5.5
1	1	0	Additional outputs of CC4 at P5.0 to P5.6
1	1	1	Additional outputs of CC4 at P5.0 to P5.7

COCAH4 COCAL4	Compare/capture mode selection for the CC register 4 For details on mode selection with COCAH4/COCAL4 see **table 6-6**.
COMO	CC4 compare mode select bit When set, compare mode 1 is selected for CC4. COMO = 0 selects compare mode 0 for CC4. Setting of bit COCOEN0 automatically sets COMO..

Table 6-6
Configurations for Concurrent Compare Mode and Compare Mode 2 at Port 5

COCAH4	COCAL4	COCOEN1	COCOEN0	Function of CC4	Function of Compare Modes at P5
0	0	0	0	Compare / Capture disabled	Disabled
		1			Compare mode 2 selected, but only interrupt generation (ICR, ICS); no output signals.at P5
		1	1		Compare Mode 2 selected at P5
0	1	0	0	Capture on falling/ rising edge at pin P1.4/$\overline{\text{INT2}}$/CC4	Disabled
		1			Compare mode 2 selected, but only interrupt generation (ICR, ICS); no output signals at P5
1	0	0	0	Compare enable at CC4; mode 0/1 is selected by COMO	Disabled
		1			Compare mode 2 selected, but only interrupt generation (ICR, ICS); no output signals at P5
		0	1	Compare mode 1 enabled at CC4; COMO is automatically set	Concurrent compare (mode 1) selected at P5
		1			Compare mode 2 selected at P5
1	1	0	0	Capture on write operation into register CCL4	Disabled
		1			Compare mode 2 selected, but only interrupt generation (ICR, ICS); no output signals at P5

Note : All other combinations of the 4 mode select bits are reserved and must not be used.

6.3.4.4 Compare Function of Registers CM0 to CM7

The CCU of the C517A contains another set of eight compare registers and an additional timer, the compare timer, and some control SFRs. These compare registers and the compare timer are mainly dedicated to PWM applications.

The compare registers CM0 to CM7, however, are not permanently assigned to the compare timer, each register may individually be configured to work either with timer 2 or the compare timer. This flexible assignment of the CMx registers allows an independent use of two time bases whereby different application requirements can be met. Any CMx register connected to the compare timer automatically works in compare mode 0 e.g. to provide fast PWM with low CPU intervention. CMx registers which are assigned to timer 2, operate in comare mode 1. This allows the CPU to control the compare output transitions directly.

The assignment of the eight registers CM0 to CM7 to either timer 2 or to the compare timer is done by a multiplexer which is controlled by the bits in the SFR CMSEL. The compare function itself can individually be enabled in the SFR CMEN. These two registers are not bit-addressable. This means, that the value of single bits can only be changed by AND-ing or OR-ing the register with a certain mask.

Special Function Register CMSEL (Address $F7_H$) **Reset Value : 00_H**
Special Function Register CMEN (Address $F6_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$F7_H$	.7	.6	.5	.4	.3	.2	.1	.0	CMSEL
	7	6	5	4	3	2	1	0	
$F6_H$	.7	.6	.5	.4	.3	.2	.1	.0	CMEN

Bit	Function
CMSEL.7 - 0	Select bits for CMx registers (x = 0-7) When set, the CMLx/CMHx registers are assigned to the compare timer and compare mode 0 is enabled. The compare registers are assigned to timer 2 if CMSELx = 0. In this case compare mode 1 is selected.
CMEN.7 - 0	Enable bits for compare registers CMx (x = 0 - 7) When set, the compare function is enabled and led to the output lines of port 4.

6.3.4.4.1 CMx Registers Assigned to the Compare Timer

Every CMx register assigned to the compare timer as a time base operates in compare mode 0 and uses a port 4 pin as an alternate output function.

The "Timer Overflow Controlled" (TOC) Loading

There is one great difference between a CMx register and the other previously described compare registers: compare outputs controlled by CMx registers have no dedicated interrupt function. They use a "timer overflow controlled loading" (further on called "TOC loading") to reach the same performance as an interrupt controlled compare. To show what this "TOC loading" is for, it will be explained more detailed in the following:

The main advantage of the compare function in general is that the controller's outputs are precisely timed by hardware, no matter which task is running on the CPU. This in turn means that the CPU normally does not know about the timer count. So, if the CPU writes to a compare register only in relation to the program flow, then it could easily be that a compare register is overwritten before the timer had the chance to reach the previously loaded compare value. Hence, there must be something to "synchronize" the loading of the compare registers to the running timer circuitry. This could either be an interrupt caused by the timer circuitry (as described before) or a special hardware circuitry.

Thus "TOC-Ioading" means that there is dedicated hardware in the CCU which synchronizes the loading of the compare registers CMx in such a way that there is no loss of compare events. lt also relieves the CPU of interrupt load.

A CMx compare register in compare mode 0 consists of two latches. When the CPU tries to access a CMx register it only addresses a register latch and not the actual compare latch which is connected to the comparator circuit. The contents of the register latch may be changed by the CPU at any time because this change would never affect the compare event for the current timer period. The compare latch (the "actual" latch) holds the compare value for the present timer period. Thus the CPU only changes the compare event for the next timer period since the loading of the latch is performed by the timer overflow signal of the compare timer.

This means for an application which uses several PWM outputs that the CPU does not have to serve every single compare line by an individual interrupt. lt only has to watch the timer overflow of the compare timer and may then set up the compare events of all compares for the next timer period. This job may take the whole current timer period since the TOC loading prevents unintentional overwriting of the actual (and prepared) value in the compare latch.

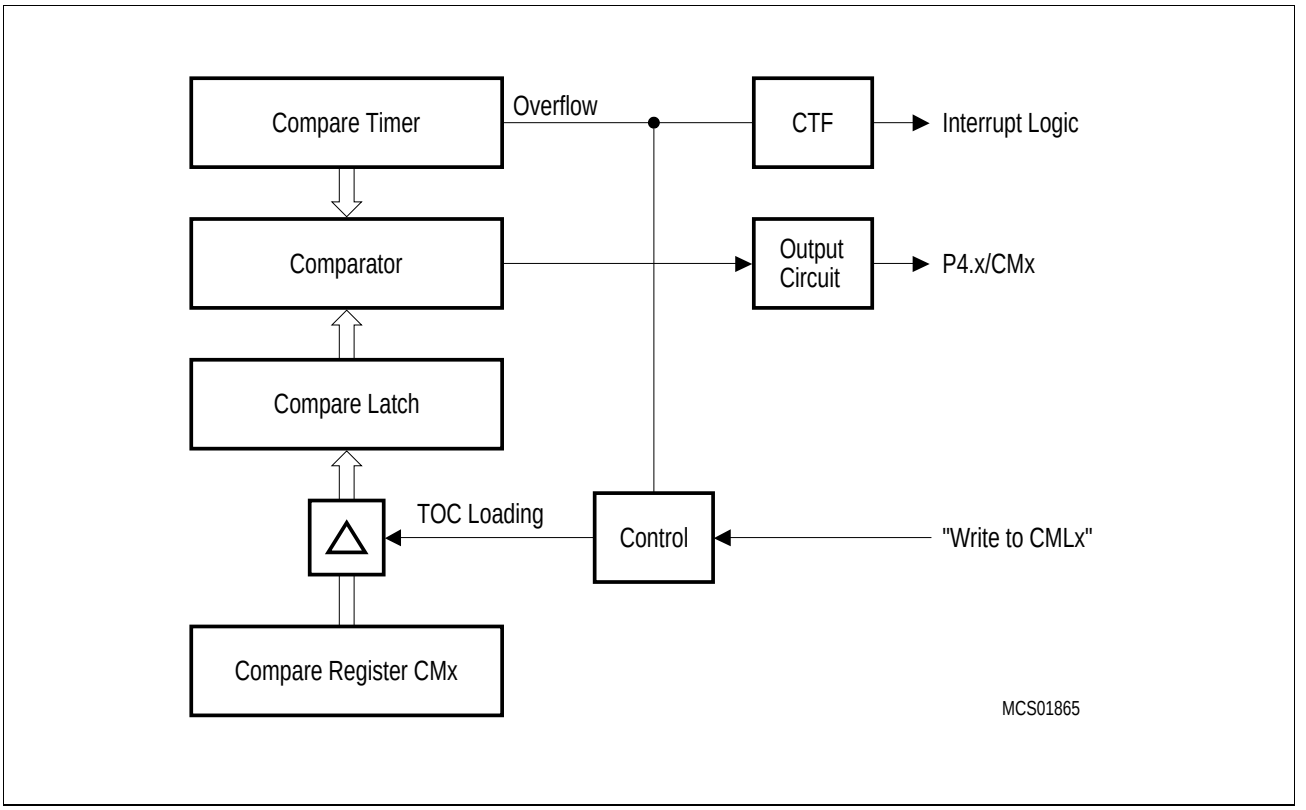


Figure 6-27
Compare Function of a CMx Register Assigned to the Compare Timer

Figure 6-27 shows a more detailed block diagram of a CMx register connected to the compare timer. It illustrates that the CPU can only access the special function register CMx; the actual compare latch is, however, loaded at timer overflow. The timer overflow signal also sets an interrupt request flag (CTF in register CTCON) which may be used to inform the CPU by an interrupt that a new timer cycle has started and that the compare values for the next cycle may be programmed from now on.

The activation of the TOC loading depends on a few conditions described in the following. A TOC loading is performed only if the CMLx register has been changed by the CPU. A write instruction to the low byte of the CMx register is used to enable the loading. The 8-bit architecture of the C517A requires such a defined enable mechanism because 16-bit values are to be transferred in two portions (= two instructions).

Imagine the following situation: one instruction (e.g. loading the low byte of the compare register) is executed just before timer overflow and the other instruction (loading the high byte) after the overflow. If there were no "rule", the TOC loading would just load the new low byte into the compare latch. The high byte - written after timer overflow - would have to wait till the next timer overflow.

The mentioned condition for TOC loading prevents such undesired behavior. If the user writes the high byte first then no TOC loading will happen before the low byte has been written - even if there is a timer overflow in between. If the user just intends to change the low byte of the compare latch then the high byte may be left unaffected.

Summary of the TOC loading capability :

- The CMx registers are - when assigned to the compare timer - protected from direct loading by the CPU. A register latch couple provides a defined load time at timer overflow.
- Thus, the CPU has a full timer period to load a new compare value: there is no danger of overwriting compare values which are still needed in the current timer period.
- When writing a 16-bit compare value, the high byte should be written first since the write-to-low-byte instruction enables a 16-bit wide TOC loading at next timer overflow.
- If there was no write access to a CMx low byte then no TOC loading will take place.
- Because of the TOC loading, all compare values written to CMx registers are only activated in the next timer period.

Initializing the Compare Register/Compare Latch Circuit

Normally when the compare function is desired the initialization program would just write to the compare register (called 'register latch'). The compare latch itself cannot be accessed directly by a move instruction, it is exclusively loaded by the timer overflow signal.

In some very special cases, however, an initial loading of the compare latch could be desirable. If the following sequence in **table 6-7** is observed during initialization then latches, the register and the compare latch, can be loaded before the compare mode is enabled.

Table 6-7
Compare Register/Latch Initializing Sequence

Step	Action	Comment
1	Select compare mode 1 (CMSEL.x = 0).	This is also the default value after reset.
2	Move the compare value for the first timer period to the compare register CMx (high byte first).	In compare mode 1 latch is loaded directly after a write-to-CMLx. Thus the value slips directly into the compare latch.
3	Switch on compare mode 0 (CMSEL.x = 1)	Now select the right compare mode.
4	Move the compere value for the second timer period to the compare register.	The register latch is loaded; this value is used after the first timer overflow.
5	Enable the compare function (CMEN.x = 1)	–
6	Set up the prescaler for the compare timer.	–
7	Set specific compare output to low level (CLR P4.x)	The compare output is switched to low level.
8	Start the compare timer with a desired value (write-to-CTREL)	Compare function is initialized; the output will oscillate.

6.3.4.4.2 CMx Registers Assigned to the Timer 2

Any CMx register assigned to timer 2 as a time base operates in compare mode 1. In this case CMx registers behave like any other compare register connected to timer 2 (e.g. the CRC or CCx registers).

Since there are no dedicated interrupts for the CMx compare outputs, again a buffered compare register structure is used to determine an exact 16-bit wide loading of the compare value: the compare value is transferred to the actual compare latches at a write-to-CMLx instruction (low byte of CMx). Thus, the CMx register is to be written in a fixed order, too: high byte first, low byte second. If the high byte may remain unchanged it is sufficient to load only the low byte. See **figure 6-28**, block diagram of a CMx register connected to timer 2.

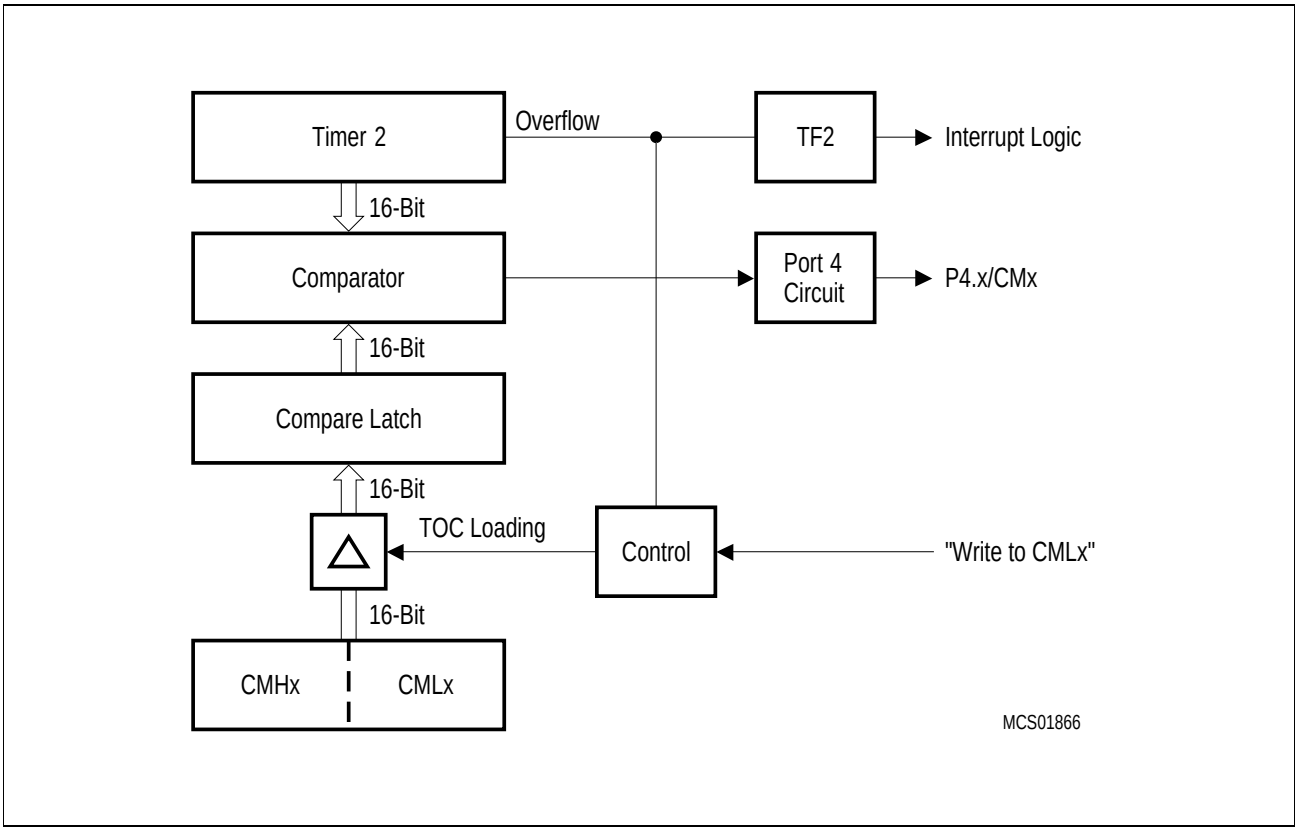


Figure 6-28
CMx-Register Assigned to Timer 2

6.3.4.5 Timer 2 Operating in Compare Mode 2

The compare mode 2 of the CCU can be used for the concurrent compare output function at port 5. In compare mode 2, the port 5 pins are no longer general purpose I/O pins or under control of the compare/capture register CC4, but under control of the compare registers COMSET and COMCLR. The details of compare mode 2 are already described in section 6.3.3.3. **Figure 6-29** shows the complete compare mode 2 configuration of the CCU and the port 5 pins.

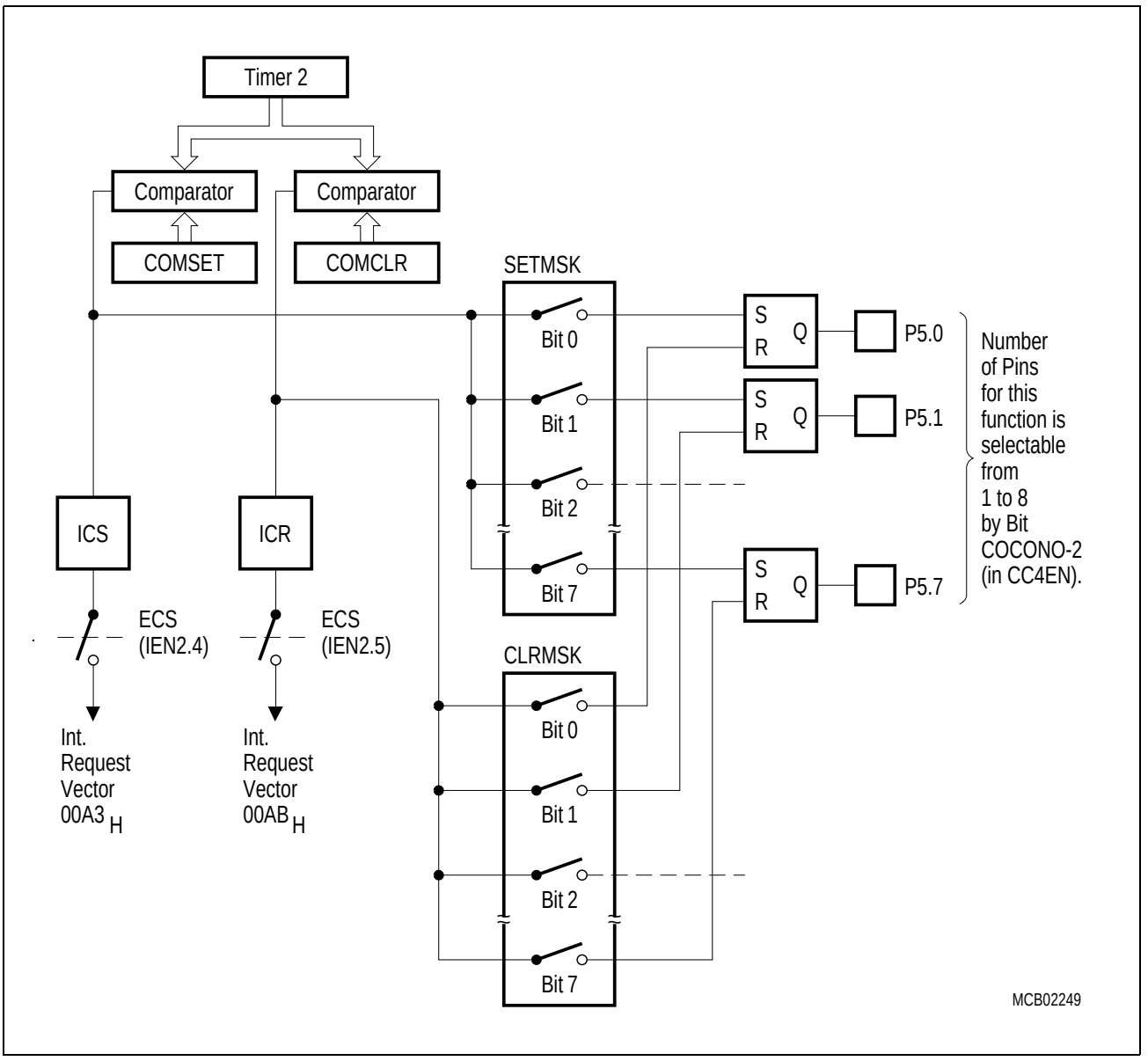


Figure 6-29
Compare Mode 2 (Port 5 only)

The compare registers COMSET and COMCLR have their dedicated interrupt vectors. The corresponding request flags are ICS for register COMSET and ICR for register COMCLR. The flags are set by a match in registers COMSET and COMCLR, when enabled. As long as the match condition is valid the request flags can't be reset (neither by hardware nor software). The request flags are located in SFR CTCON.

6.3.5 Modulation Range in Compare Mode 0

In compare mode 0, a 100% variation of the duty cycle of a PWM signal cannot be reached. A time portion of $1/(2^n)$ of an n-bit timer period is always left over. This "spike" may either appear when the compare register is set to the reload value (limiting the lower end of the modulation range) or it may occur at the end of a timer period.

In a timer 2 / CCx register configuration in compare mode 0 this spike is divided into two halves: one at the beginning when the contents of the compare register is equal to the reload value of the timer; the other half when the compare register is equal to the maximum value of the timer register (here: $FFFF_H$). Please refer to **figure 6-30** where the maximum and minimum duty cycle of a compare output signal is illustrated. Timer 2 is incremented with the processor cycle ($f_{OSC}/12$), thus at 12 MHz operating frequency, these spikes are both approx. 500 ns long.

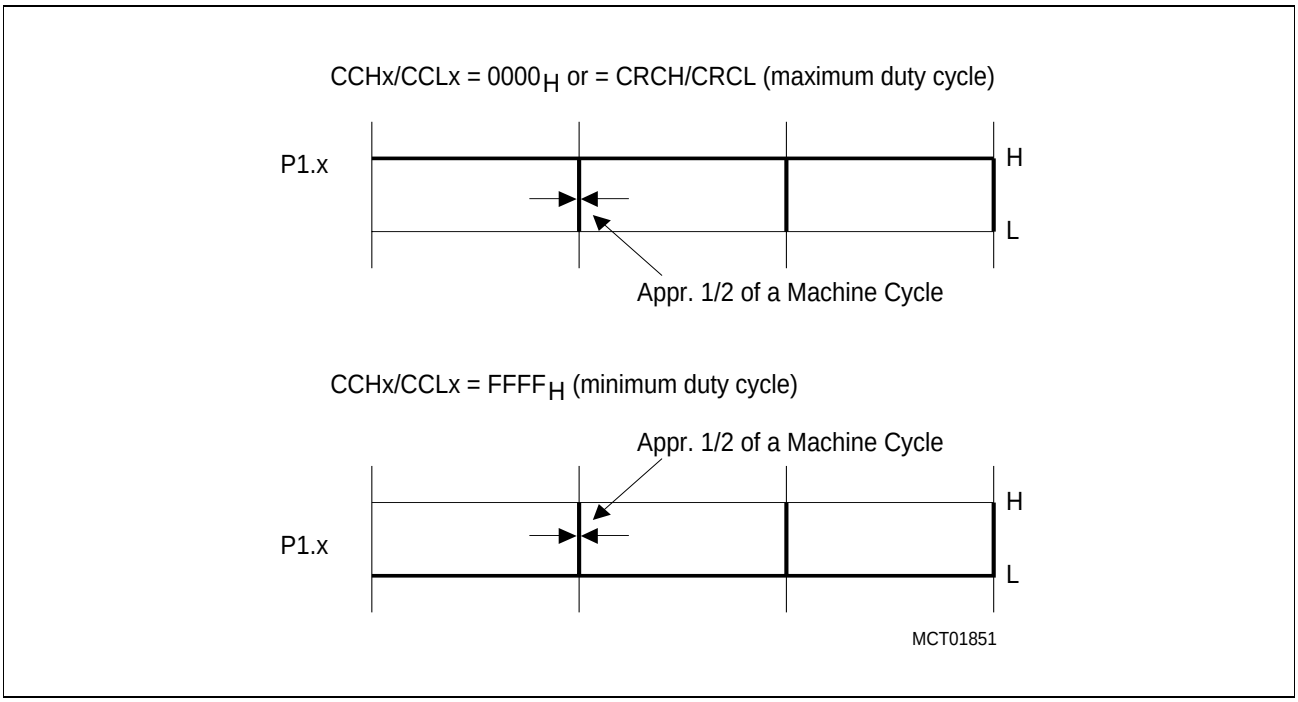


Figure 6-30
Modulation Range of a PMW Signal Generated with a Timer 2 / CCx Register Combination in Compare Mode 0

The following example shows how to calculate the modulation range for a PWM signal. For the calculation with reasonable numbers, a reduction of the resolution to 8-bit is used. Otherwise (for the maximum resolution of 16-bit) the modulation range would be so severely limited that it would be negligible.

Example:

Timer 2 in auto-reload mode; contents of reload register CRC = $FF00_H$

$$\text{Restriction of modulation. range} = \frac{1}{256 \times 2} \times 100\% = 0.195\%$$

This leads to a variation of the duty cycle from 0.195% to 99.805% for a timer 2 / CCx register configuration when 8 of 16 bits are used.

In a compare timer/CMx register configuration, the compare output is set to a constant high level if the contents of the compare registers are equal to the reload register (CTREL). The compare output shows a high level for one timer clock period when a CMx register is set to $FFFF_H$. Thus, the duty cycle can be varied from 0.xx% to 100% depending on the resolution selected. In **figure 6-31** the maximum and minimum duty cycle of a compare output signal is illustrated. One clock period of the compare timer is equal to one machine state (= 2 oscillator periods) if the prescaler is off. Thus, at 12 MHz system clock the spike is approx. 166.6 ns long.

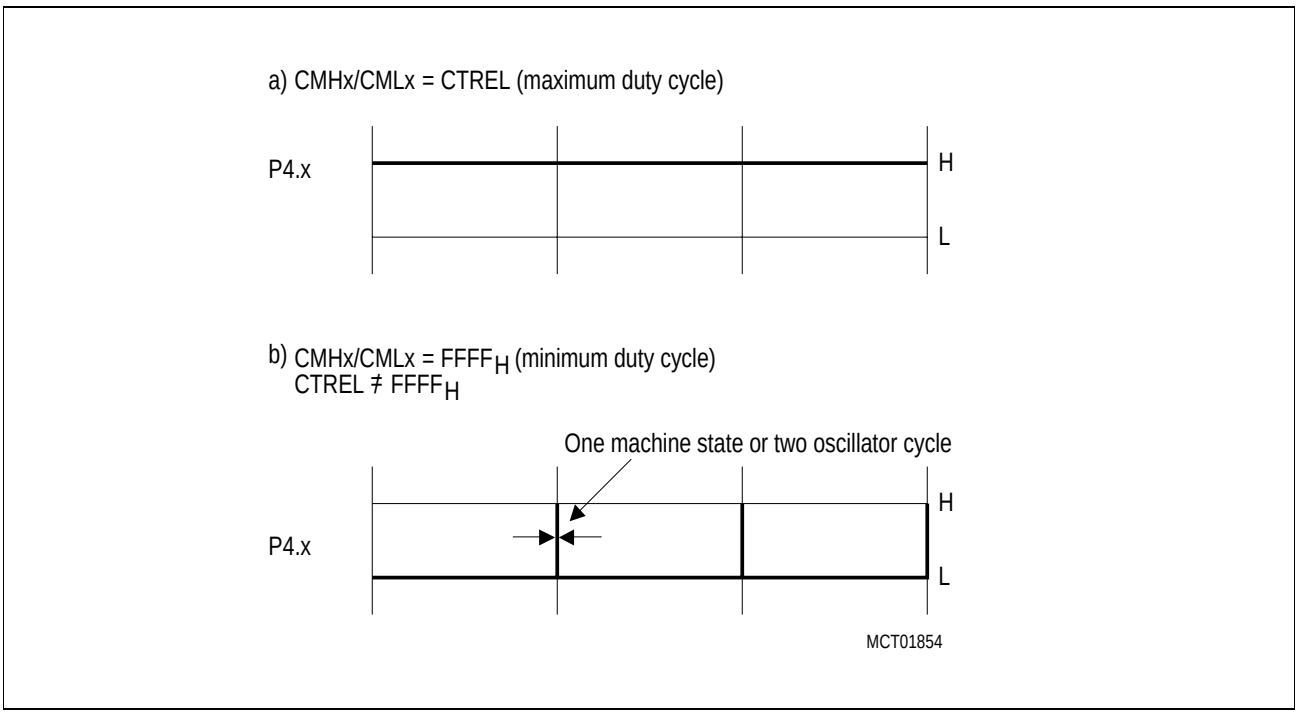


Figure 6-31
Modulation Range of a PWM Signal Generated with a Compare Timer/CMx Register Combination

6.3.6 Using Interrupts in Combination with the Compare Function

The compare service of registers CRC, CC1, CC2, CC3 and CC4 is assigned to alternate output functions at port pins P1.0 to P1.4. Another option of these pins is that they can be used as external interrupt inputs. However, when using the port lines as compare outputs then the input line from the port pin to the interrupt system is disconnected (but the pin's level can still be read under software control). Thus, a change of the pin's level will not cause a setting of the corresponding interrupt flag. In this case, the interrupt input is directly connected to the (internal) compare signal thus providing a compare interrupt.

The compare interrupt can be used very effectively to change the contents of the compare registers or to determine the level of the port outputs for the next "compare match". The principle is, that the internal compare signal (generated at a match between timer count and register contents) not only manipulates the compare output but also sets the corresponding interrupt request flag. Thus, the current task of the CPU is interrupted - of course provided the priority of the compare interrupt is higher than the present task priority - and the corresponding interrupt service routine is called. This service routine then sets up all the necessary parameters for the next compare event.

6.3.6.1 Advantages in Using Compare Interrupts

Firstly, there is no danger of unintentional overwriting a compare register before a match has been reached. This could happen when the CPU writes to the compare register without knowing about the actual timer 2 count.

Secondly, and this is the most interesting advantage of the compare feature, the output pin is exclusively controlled by hardware therefore completely independent from any service delay which in real time applications could be disastrous. The compare interrupt in turn is not sensitive to such delays since it loads the parameters for the next event. This in turn is supposed to happen after a sufficient space of time.

Please note two special cases where a program using compare interrupts could show a "surprising" behavior:

The first configuration has already been mentioned in the description of compare mode 1. The fact that the compare interrupts are transition activated becomes important when driving timer 2 with a slow external clock. In this case it should be carefully considered that the compare signal is active as long as the timer 2 count is equal to the contents of the corresponding compare register, and that the compare signal has a rising and a falling edge. Furthermore, the "shadow latches" used in compare mode 1 are transparent while the compare signal is active.

Thus, with a slow input clock for timer 2, the comparator signal is active for a long time (= high number of machine cycles) and therefore a fast interrupt controlled reload of the compare register could not only change the "shadow latch" - as probably intended - but also the output buffer.

When using the CRC or CC4 register, you can select whether an interrupt should be generated when the compare signal goes active or inactive, depending on the status of bits I3FR or I2FR in T2CON, respectively.

Initializing the interrupt to be negative transition triggered is advisive in the above case. Then the compare signal is already inactive and any write access to the port latch just changes the contents of the "shadow-latch".

Please note that for CC registers 1 to 3 an interrupt is always requested when the compare signal goes active.

The second configuration which should be noted is when compare functions are combined with negative transition activated interrupts. If the port latch of port P1.0 or P.1.4 contains a 1, the interrupt request flags IEX3 or IEX2 will immediately be set after enabling the compare mode for the CRC or CC4 register. The reason is that first the external interrupt input is controlled by the pin's level. When the compare option is enabled the interrupt logic input is switched to the internal compare signal, which carries a low level when no true comparison is detected. So the interrupt logic sees a 1-to-0 edge and sets the interrupt request flag.

An unintentional generation of an interrupt during compare initialization can be prevented if the request flag is cleared by software after the compare is activated and before the external interrupt is enabled.

6.3.6.2 Interrupt Enable Bits of the Compare/Capture Unit

This section summarizes all CCU related interrupt enable control bits. The interrupt enable bits for the compar timer and the compare match and capture interrupt capture are located in the SFR IEN2:

Special Function Register IEN2 (Address $9A_H$) **Reset Value : $XX0000X0_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$9A_H$	–	–	ECR	ECS	ECT	ECMP	–	ES1	IEN2

The shaded bits are not used for CCU interrupt control.

Bit	Function
ECR	COMCLR register compare match interrupt enable If ECR = 0, the COMCLR compare match interrupt is disabled.
ECS	COMSET register compare match interrupt enable If ECS = 0, the COMSET compare match interrupt is disabled.
ECT	Enable compare timer interrupt If ECT = 0, the compare timer overflow interrupt is disabled.
ECMP	CM0-7 register compare match interrupt If ECMP = 0, the CM0-7 compare match interrupt is disabled.

6.3.6.3 Interrupt Flags of the Compare/Capture Unit

This section handles the CCU related compare match interrupt flags. The timer 2 and compare timer overflow interrupt flags (TF2 and CTF) are described in detail in section 6.3.1.1 and 6.3.2.1.

The compare timer match interrupt occurs on a compare match of the CM0 to CM7 registers with the compare timer when compare mode 1 is selected for the corresponding channel. There are 8 compare match interrupt flags available in SFR IRCON1 which are or-ed together for a single interrupt request. Thus, a compare match interrupt service routine has to check which compare match has requested the compare match interrupt. The ICMPx flags must be cleared by software.

Only if timer 2 is assigned to the CMx registers (compare mode 0), an ICMPx request flag is set by every match in the compare channel. When the compare timer is assigned to the CMx registers (compare mode 1), an ICMPx request flag will not be set by a compare match event.

Special Function Register IRCON1 (Address $D1_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$D1_H$	ICMP7	ICMP6	ICMP5	ICMP4	ICMP3	ICMP2	ICMP1	ICMP0	IRCON1

Bit	Function
ICMP7 - 0	Compare timer match with register CM7 - CM0 interrupt flags ICMPx is set by hardware when a compare match of the compare timer with the compare register CMx occurs but only if the compare function for CMx has been enabled. ICMPx must be cleared by software (CMSEL.x = 0 and CMEN.x = 1).

6.4 Arithmetic Unit

This on-chip arithmetic unit of the C517A provides fast 32-bit division, 16-bit multiplication as well as shift and normalize features. All operations are unsigned integer operations.

The arithmetic unit (further on also called MDU for "Multiplication/Division Unit") has been integrated to support the C500 core of the C517A in real-time control applications. It can increase the execution speed of math-intensive software routines by factor 5 to 10.

The MDU is handled by seven registers, which are memory mapped as special function registers like any other registers for peripheral control. Therefore, the arithmetic unit allows operations concurrently to and independent of the CPU's activity. **Table 6-8** describes the four general operations the MDU is able to perform:

Table 6-8
MDU Operation Characteristics

Operation	Result	Remainder	Execution Time
32bit/16bit	32bit	16bit	6 t_{CY} 1)
16bit/16bit	16bit	16bit	4 t_{CY} 1)
16bit x 16bit	32bit	–	4 t_{CY} 1)
32-bit normalize	–	–	6 t_{CY} 2)
32-bit shift L/R	–	–	6 t_{CY} 2)

1) 1 t_{CY} = 12 t_{CLCL}= 1 machine cycle = 500 ns at 24 MHz oscillator frequency
2) The maximal shift speed is 6 shifts per machine cycle

6.4.1 MDU Register

The seven SFRs of the MDU consist of registers MD0 to MD5, which contain the operands and the result (or the remainder, resp.) and one control register called ARCON.

Thus MD0 to MD5 are used twofold:

- for the operands before a calculation has been started and
- for storage of the result or remainder after a calculation.

This means that any calculation of the MDU overwrites its operands. If a program needs the original operands for further use, they should be stored in general purpose registers in the internal RAM. **Table 6-8** list the MDU registers with its addresses :

Table 6-9
MDU Registers

SFR	Address	Name
ARCON	EF_H	MDU Control Register
MD0	$E9_H$	MDU Data Register 0
MD1	EA_H	MDU Data Register 1
MD2	EB_H	MDU Data Register 2
MD3	EC_H	MDU Data Register 3
MD4	ED_H	MDU Data Register 4
MD5	EE_H	MDU Data Register 5

The arithmetic control register ARCON contains control flags and the shift counter of the MDU. It triggers a shift or a normalize operation in register MD0 to MD3 when being written to.

Special Function Register ARCON (Address EF_H) **Reset Value : $0XXXXXXX_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
EF_H	MDEF	MDOV	SLR	SC.4	SC.3	SC.2	SC.1	SC.0	ARCON

Bit	Function
MDEF	Error flag Indicates an improperly performed operation. MDEF is set by hardware when an operation is retriggered by a write access to MDx before the first operation has been completed. MDEF is automatically cleared after being read.
MDOV	Overflow flag Exclusively controlled by hardware. MDOV is set by following events: – division by zero – multiplication with a result greater than $FFFF_H$.
SLR	Shift direction bit When set, shift right is performed. SLR = 0 selects shift left operation.
SC.4 - SC.0	Shift counter bits When preset with 00000_B, normalizing is selected. After operation SC.0 to SC.4 contain the number of normalizing shifts performed. When set with a value ≠ 0, shift operation is started. The number of shifts performed is determined by the count written to SC.0 to SC.4.

6.4.2 Operation of the MDU

The MDU can be regarded as a special coprocessor for multiplication, division and shift. Its operations can be divided into three phases (see **figure 6-32**):

1) Loading the MDx registers
2) Executing the calculation
3) Reading the result from the MDx registers

During phase two, the MDU works on its own parallelly to the CPU. Execution times of the above table refer to this phase. Because of the fast operation and the determined execution time for C517A's instructions, there is no need for a busy flag. The CPU may execute a determined number of instructions before the result is fetched. The result and the remainder of an operation may also be stored in the MDx registers for later use.

Phase one and phase three require CPU activity. In these phases the CPU has to transfer the operands and fetch the results.

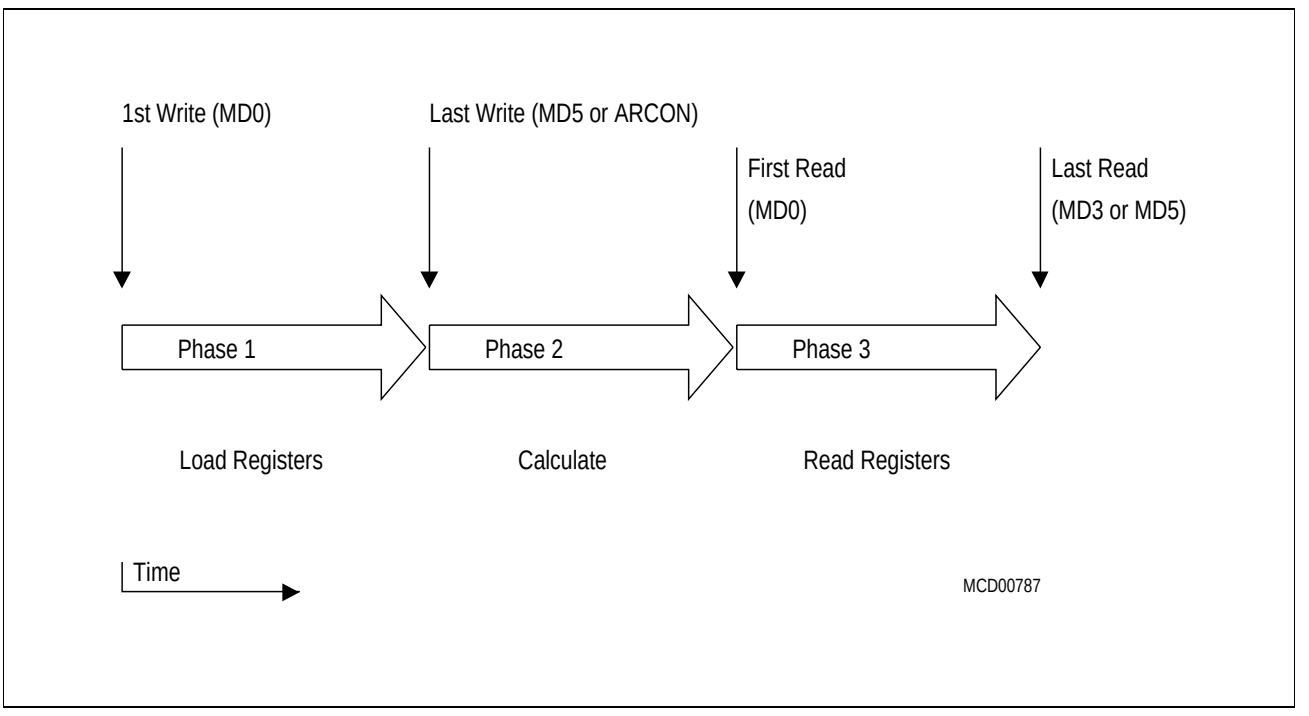


Figure 6-32
Operating Phases of the MDU

The MDU has no dedicated instruction register (only for shift and normalize operations, register ARCON is used in such a way). The type of calculation the MDU has to perform is selected following the order in which the MDx registers are written to (see **table 6-10**). This mechanism also reduces execution time spent for controlling the MDU. Hence, a special write sequence selects an operation.

The MDU monitors the whole write and read-out sequence to ensure that the CPU has fetched the result correctly and was not interrupted by another calculation task.

Thus, a complete operation lasts from writing the first byte of the operand in phase 1 until reading the last byte of the result in phase 3.

6.4.3 Multiplication/Division

The general mechanism to start an MDU activity has been described above. The following description of the write and read sequences adds to the information given in the table below where the write and read operations necessary for a multiplication or division are listed.

Table 6-10
Programming the MDU for Multiplication and Division

Operation	32Bit/16Bit		16Bit/16Bit		16Bit x 16Bit	
First Write	MD0	D'endL	MD0	D'endL	MD0	M'andL
	MD1	D'end	MD1	D'endH	MD4	M'orL
	MD2	D'end				
	MD3	D'endH	MD4	D'orL	MD1	M'andH
	MD4	D'orL				
Last Write	MD5	D'orH	MD5	D'orH	MD5	M'orH
First Read	MD0	QuoL	MD0	QuoL	MD0	PrL
	MD1	Quo	MD1	QuoH	MD1	
	MD2	Quo				
	MD3	QuoH	MD4	RemL	MD2	
	MD4	RemL				
Last Read	MD5	RemH	MD5	RemH	MD3	PrH

Abrevations :

D'end : Dividend, 1st operand of division
D'or : Divisor, 2nd operand of division
M'and : Multiplicand, 1st operand of multiplication
M'or : Multiplicator, 2nd operand of multiplication
Pr : Product, result of multiplication
Rem : Remainder
Quo : Quotient, result of division
...L : means, that this byte is the least significant of the 16-bit or 32-bit operand
...H : means, that this byte is the most significant of the 16-bit or 32-bit operand

Write Sequence

The first and the last write operation in phase one are fixed for every calculation of the MDU. All write operations inbetween determine the type of MDU calculation.

- A write-to-MD0 is the first transfer to be done in any case. This write resets the MDU and triggers the error flag mechanism (see below).
- The next two or three write operations select the calculation type (32bit/16bit, 16bit/16bit, 16bit x 16bit)
  The last write-to-MD5 finally starts the selected MUL/DIV operation

Read Sequence

- Any read-out of the MDx registers should begin with MD0
- The last read from MD5 (division) or MD3 (multiplication) determines the end of a whole calculation and releases the error flag mechanism.

There is no restriction on the time within which a calculation must be completed. The CPU is allowed to continue the program simultaneously to phase 2 and to fetch the result bytes at any time.

If the user's program takes care that interrupting a calculation is not possible, monitoring of the calculation process is probably not needed. In this case, only the write sequence must be observed.

Any new write access to MD0 starts a new calculation, no matter whether the read-out of the former result has been completed or not.

6.4.4 Normalize and Shift

Register ARCON controls an up to 32-bit wide normalize and shift operation in registers MD0 to MD3. It also contains the overflow flag and the error flag which are described in the next two sections.

Write Sequence

- A write-to-MD0 is also the first transfer to be done for normalize and shift. This write resets the MDU and triggers the error flag mechanism (see below).
- To start a shift or normalize operation the last write must access register ARCON.

Read Sequence

- The order in which the first three registers MD0 to MD2 are read is not critical
- The last read from MD3 determines the end of a whole shift or normalize procedure and releases the error flag mechanism.

Note : Any write access to ARCON triggers a shift or normalize operation and therefore changes the contents of registers MD0 to MD3 !

Normalizing

Normalizing is done on an integer variable stored in MD0 (least significant byte) to MD3 (most significant byte). This feature is mainly meant to support applications where floating point arithmetic is used. "To normalize" means, that all reading zeroes of an integer variable in registers MD0 to MD3 are removed by shift left operations. The whole operation is completed when the MSB (most significant bit) contains a ’1’.

To select a normalize operation, the five bit field ARCON.0 to ARCON.4 must be cleared. That means, a write-to-ARCON instruction with the value XXX0 0000_B starts the operation.

After normalizing, bits ARCON.0 to ARCON.4 contain the number of shift left operations which were done. This number may further on be used as an exponent. The maximum number of shifts in a normalize operation is 31 (= 2^5 – 1). The operation takes six machine cycles at most, that means 3 microseconds at 24 MHz.

Shifting

In the same way - by a write-to-ARCON instruction - a shift left/right operation can be started. In this case register bit SLR (ARCON.5) has to contain the shift direction, and ARCON.0 to ARCON.4 the shift count (which must not be 0, otherwise a normalize operation would be executed). During shift, zeroes come into the left or right end of the registers MD0 or MD3, respectively.

The first machine cycle of a shift left/right operation executes four shifts, while all following cycles perform 6 shifts. Hence, a 31-bit shift takes 3 microseconds at 24 MHz.

Completion of both operations, normalize and shift, can also be controlled by the error flag mechanism. The error flag is set if one of the relevant registers (MD0 through MD3) is accessed before the previously commenced operation has been completed.

For proper operation of the error flag mechanism, it is necessary to take care that the right write or read sequence to or from registers MD0 to MD3 (see **table 6-11**) is maintained.

Table 6-11
Programming a Shift or Normalize Operation

Operation	Normalize, Shift Left, Shift Right	
First write	MD0	least significant byte
	MD1	.
	MD2	.
	MD3	most significant byte
Last write	ARCON	start of conversion
First read	MD0	least significant byte
	MD1	.
	MD2	.
Last read	MD3	most significant byte

6.4.5 The Overflow Flag

An overflow flag is provided for some exceptions during MDU calculations. There are three cases where flag MDOV ARCON.6 is set by hardware:

- Division by zero
- Multiplication with a result greater then 0000 $FFFF_H$
  (= auxiliary carry of the lower 16bit)
- Start of normalizing if the most significant bit of MD3 is set (MD3.7 = 1).

Any operation of the MDU which does not match the above conditions clears the overflow flag. Note that the overflow flag is exclusively controlled by hardware. It cannot be written to.

6.4.6 The Error Flag

The error flag, bit MDEF in register ARCON is provided to indicate whether one of the arithmetic operations of the MDU (multiplication, division, normalize, shift left/right) has been restarted or interrupted by a new operation.

This can possibly happen e.g. when an interrupt service routine interrupts the writing or reading sequence of the arithmetic operation in the main program and starts a new operation. Then the contents of the corresponding registers are indeterminate (they would normally show the result of the last operation executed).

In this case the error flag can be used to indicate whether the values in the registers MD0 to MD5 are the expected ones or whether the operation must be repeated. For a multiplication/division, the error flag mechanism is automatically enabled with the first write instruction to MD0 (phase 1). According to the above described programming sequences, this is the first action for every type of calculation. The mechanism is disabled with the final read instruction from MD3 or MD5 (phase 3). Every instruction which rewrites MD0 (and therefore tries to start a new calculation) in phases 1 through 3 of the same process sets the error flag.

The same applies for any shift operation (normalize, shift left/right). The error flag is set if the user's program reads one of the relevant registers (MD0 to MD3) or if it writes to MD0 again before the shift operation has been completed.

Please note that the error flag mechanism is just an option to monitor the MDU operation. lf the user's program is designed such that an MDU operation cannot be interrupted by other calculations, then there is no need to pay attention to the error flag. In this case it is also possible to change the order in which the MDx registers are read, or even to skip some register read instructions. Concerning the shift or normalize instructions, it is possible to read the result before the complete execution time of six machine cycles has passed (e.g. when a small number of shifts has been programmed). All of the above "illegal" actions would set the error flag, but on the other hand do not affect a correct MDU operation. The user has just to make sure that everything goes right.

The error flag (MDEF) is located in ARCON and can be read only. lt is automatically cleared after being read.

6.5 Serial Interfaces

The C517A has two serial interfaces which are functionally nearly identical concerning the asynchronous modes of operation. The two channels are full-duplex, meaning they can transmit and receive simultaneously. They are also receive buffered, meaning they can commence reception of a second byte before a previously received byte has been read from the receive register (however, if the first byte still has not been read by the time reception of the second byte is complete, the last received byte will be lost). The serial channel 0 is completely compatible with the serial channel of the C501. Serial channel 1 has the same functionality in its asynchronous modes, but the synchronous mode is missing.

6.5.1 Serial Interface 0

6.5.1.1 Operating Modes of Serial Interface 0

The serial interface 0 can operate in four modes (one synchronous mode, three asynchronous modes). The baud rate clock for this interface is derived from the oscillator frequency (mode 0, 2) or generated either by timer 1 or by a dedicated baud rate generator (mode 1, 3). A more detailed description of how to set the baud rate will follow in section 6.5.1.4.

Mode 0: Shift register (synchronous) mode:

Serial data enters and exits through RxD0. TxD0 outputs the shift clock. 8 data bits are transmitted/ received (LSB first). The baud rate is fixed at 1/6 of the oscillator frequency. (See section 6.5.3.1 for more detailed information)

Mode 1: 8-bit UART, variable baud rate:

10 bits are transmitted (through TXD0) or received (through RXD0): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On reception, the stop bit goes into RB80 in special function register S0CON. The baud rate is variable. (See section 6.5.3.2 for more detailed information)

Mode 2: 9-bit UART, fixed baud rate:

11 bits are transmitted (through TXD0) or received (through RXD0): a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). On transmission, the 9th data bit (TB80 in S0CON) can be assigned to the value of 0 or 1. For example, the parity bit (P in the PSW) could be moved into TB80 or a second stop bit by setting TB80 to 1. On reception the 9th data bit goes into RB80 in special function register S0CON, while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 of the oscillator frequency. (See section 6.5.3.3 for more detailed information)

Mode 3: 9-bit UART, variable baud rate:

11 bits are transmitted (through TXD0) or received (through RXD0): a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). On transmission, the 9th data bit (TB80 in S0CON) can be assigned to the value of 0 or 1. For example, the parity bit (P in the PSW) could be moved into TB80 or a second stop bit by setting TB80 to 1. On reception, the 9th data bit goes into RB80 in special function register S0CON, while the stop bit is ignored. In fact, mode 3 is the same as mode 2 in all respects except the baud rate. The baud rate in mode 3 is variable. (See section 6.5.3.4 for more detailed information)

In all four modes, transmission is initiated by any instruction that uses S0BUF as a destination register. Reception is initiated in mode 0 by the condition RI0 = 0 and REN0 = 1. Reception is initiated in the other modes by the incoming start bit if REN0 = 1. The serial interfaces also provide interrupt requests when a transmission or a reception of a frame has completed. The corresponding interrupt request flags for serial interface 0 are TI0 or RI0, resp. See chapter 7 of this user manual for more details about the interrupt structure. The interrupt request flags TI0 and RI0 can also be used for polling the serial interface 0 if the serial interrupt is not to be used (i.e. serial interrupt 0 not enabled).

6.5.1.2 Multiprocessor Communication Feature

Modes 2 and 3 of the serial interface 0 have a special provision for multi-processor communication. In these modes, 9 data bits are received. The 9th bit goes into RB80. Then a stop bit follows. The port can be programmed such that when the stop bit is received, the serial port 0 interrupt will be activated (i.e. the request flag RI0 is set) only if RB80 = 1. This feature is enabled by setting bit SM20 in S0CON. A way to use this feature in multiprocessor communications is as follows.

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

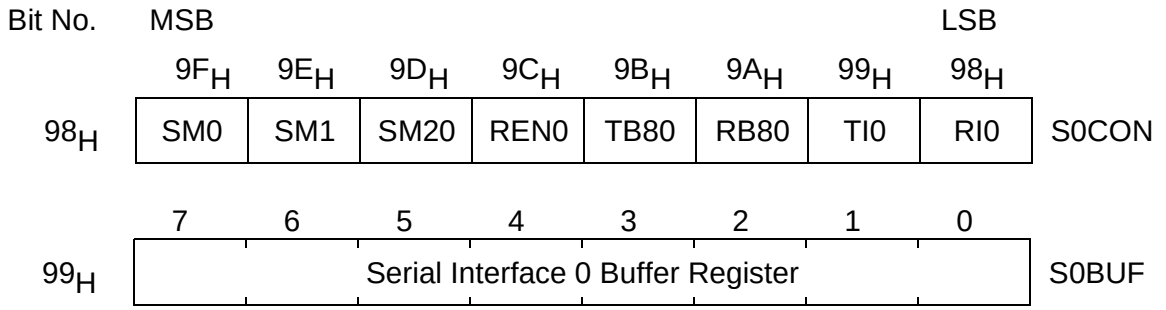
SM20 has no effect in mode 0. In mode 1 SM20 can be used to check the validity of the stop bit. If SM20 = 1 in mode 1, the receive interrupt will not be activated unless a valid stop bit is received.

6.5.1.3 Serial Port Registers

The serial port control and status register is the special function register S0CON. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB80 and RB80), and the serial port interrupt bits (TI0 and RI0).

S0BUF is the receive and transmit buffer of serial interface 0. Writing to S0BUF loads the transmit register and initiates transmission. Reading out S0BUF accesses a physically separate receive register.

Special Function Register S0CON (Address 98_H) **Reset Value : 00_H**
Special Function Register S0BUF (Address 99_H) **Reset Value : XX_H**



Bit	Function
SM0 SM1	Serial port 0 mode selection bits

SM0	SM1	Selected operating mode
0	0	Serial mode 0 : Shift register, fixed baud rate (f_{osc}/12)
0	1	Serial mode 1 : 8-bit UART, variable baud rate
1	0	Serial mode 2 : 9-bit UART, fixed baud rate (f_{osc}/32 or f_{osc}/64)
1	1	Serial mode 3 : 9-bit UART, variable baud rate

SM20	Enable serial port 0 multiprocessor communication in modes 2 and 3 In mode 2 or 3, if SM20 is set to 1 then RI0 will not be activated if the received 9th data bit (RB80) is 0. In mode 1, if SM20 = 1 then RI0 will not be activated if a valid stop bit was not received. In mode 0, SM20 should be 0.
REN0	Serial port 0 receiver enable Enables serial reception. Set by software to enable serial reception. Cleared by software to disable serial reception.
TB80	Serial port 0 transmitter bit 9 TB80 Is the 9th data bit that will be transmitted in modes 2 and 3. Set or cleared by software as desired.
RB80	Serial port 0 receiver bit 9 In modes 2 and 3, RB80 is the 9th data bit that was received. In mode 1, if SM2 = 0, RB80 is the stop bit that was received. In mode 0, RB80 is not used.
TI0	Serial port 0 transmitter interrupt flag TI0 is 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. TI0 must be cleared by software.
RI0	Serial port 0 receiver interrupt flag RI0 is 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 (exception see SM20). RI0 must be cleared by software.

6.5.1.4 Baud Rates of Serial Channel 0

There are several possibilities to generate the baud rate clock for the serial interface 0 depending on the mode in which it is operated.

For clarification some terms regarding the difference between "baud rate clock" and "baud rate" should be mentioned. The serial interface requires a clock rate which is 16 times the baud rate for internal synchronization. Therefore, the baud rate generators have to provide a "baud rate clock" to the serial interface which - there divided by 16 - results in the actual "baud rate". However, all formulas given in the following section already include the factor and calculate the final baud rate. Further, the abrevation f_{OSC} refers to the oscillator frequency (crystal or external clock operation).

The baud rate of the serial channel 0 is controlled by several bits which are located in the special function registers as shown below.

Special Function Register ADCON0 (Address $D8_H$) **Reset Value : 00_H**
Special Function Register PCON (Address 87_H) **Reset Value : 00_H**

Bit No.	MSB							LSB	
	DF_H	DE_H	DD_H	DC_H	DB_H	DA_H	$D9_H$	$D8_H$	
$D8_H$	BD	CLK	ADEX	BSY	ADM	MX2	MX1	MX0	ADCON0
	7	6	5	4	3	2	1	0	
87_H	SMOD	PDS	IDLS	SD	GF1	GF0	PDE	IDLE	PCON

The shaded bits are not used in controlling serial interface 0.

Bit	Function
BD	Baud rate generator enable When set, the baud rate of serial interface 0 is derived from a dedicated programmable baud rate generator. When cleared (default after reset), baud rate is derived from the timer 1 overflow rate.
SMOD	Double baud rate When set, the baud rate of serial interface 0 in modes 1, 2, 3 is doubled. After reset this bit is cleared.

Figure 6-33 shows the configuration for the baud rate generation of serial channel 0.

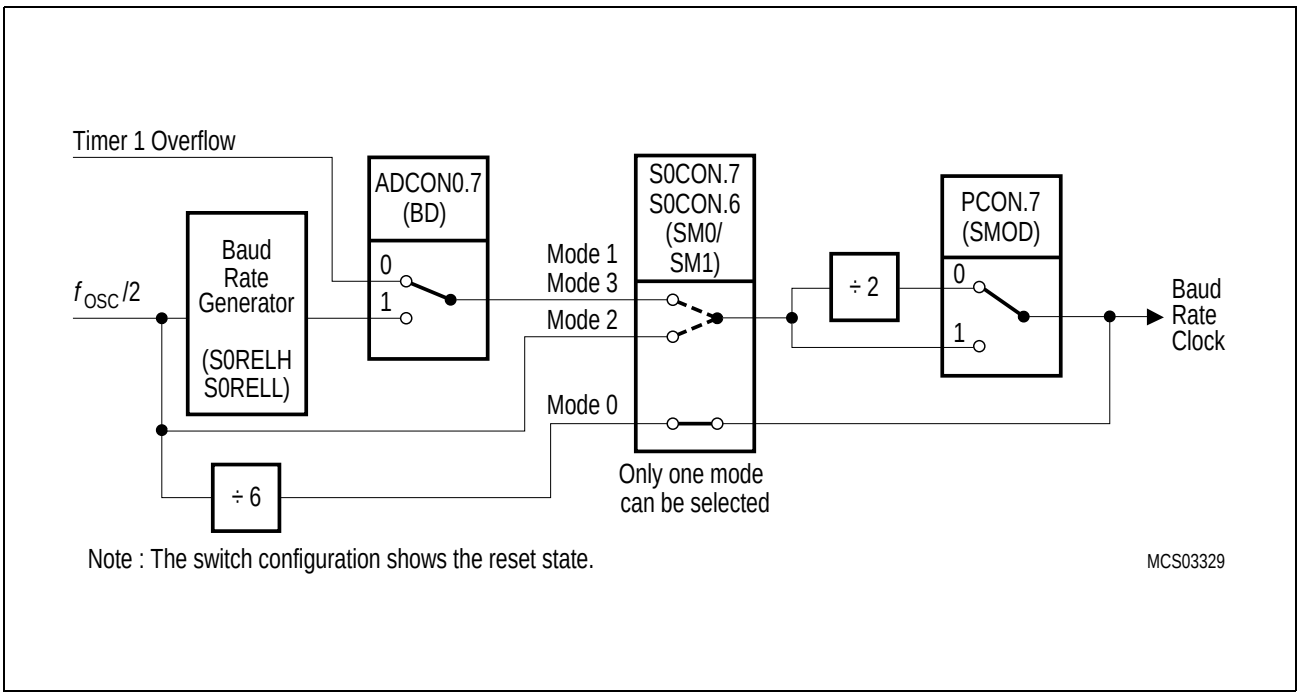


Figure 6-33
Baud Rate Generation for Serial Channel 0

Depending on the programmed operating mode different paths are selected for the baud rate clock generation. **Table 6-12** shows the dependencies of the serial port 0 baud rate clock generation from the 3 control bits and from the mode which is selected in the special function register S0CON.

Table 6-12
Serial Interface 0 - Baud Rate Dependencies

Serial Interface 0 Operating Modes	Active Control Bits		Baud Rate Clock Generation
	BD	SMOD	
Mode 0 (Shift Register)	–	–	Fixed baud rate clock fosc/12
Mode 1 (8-bit UART) Mode 3 (9-bit UART)	X	X	BD=0 : Timer 1 overflow is used for baud rate generation; SMOD controls a divide-by-2 option. BD=1 : Baud rate generator is used for baud rate generation; SMOD controls a divide-by-2 option .
Mode 2 (9-bit UART)	–	X	Fixed baud rate clock fosc/32 (SMOD=1) or fosc/64 (SMOD=0)

6.5.1.4.1 Baud Rate in Mode 0

The baud rate in mode 0 is fixed to :

$$\text{Mode 0 baud rate} = \frac{\text{oscillator frequency}}{12}$$

6.5.1.4.2 Baud Rate in Mode 2

The baud rate in mode 2 depends on the value of bit SMOD in special function register PCON. If SMOD = 0 (which is the value after reset), the baud rate is 1/64 of the oscillator frequency. If SMOD = 1, the baud rate is 1/32 of the oscillator frequency.

$$\text{Mode 2 baud rate} = \frac{2^{\text{SMOD}}}{64} \times \text{oscillator frequency}$$

6.5.1.4.3 Baud Rate in Mode 1 and 3

In these modes the baud rate is variable and can be generated alternatively by the programmable baud rate generator or by timer 1.

Using the Programmable Baud Rate Generator:

In modes 1 and 3, the C517A can use an internal baud rate generator for serial interface 0. To enable this feature, bit BD (bit 7 of special function register ADCON0) must be set. Bit SMOD (PCON.7) controls a divide-by-2 circuit which affects the input and output clock signal of the baud rate generator. After reset the divide-by-2 circuit is active and the resulting overflow output clock will be divided by 2. The input clock of the baud rate generator is $f_{OSC}/2$.

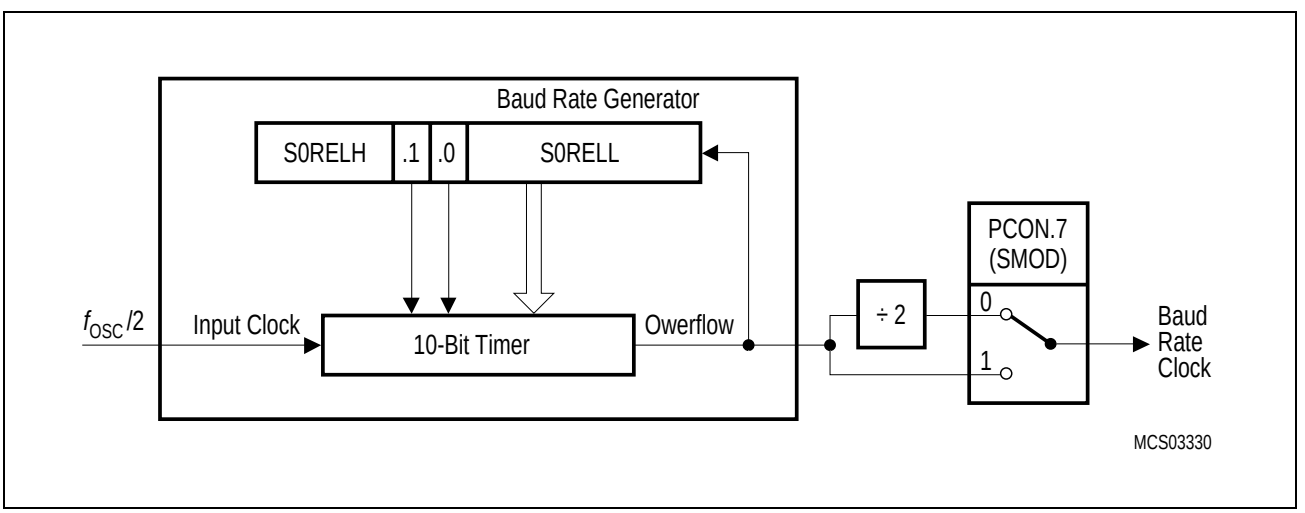


Figure 6-34
Serial Interface 0 Input Clock using the Baud Rate Generator

The baud rate generator consists of a free running upward counting 10-bit timer. On overflow of this timer (next count step after counter value $3FF_H$) there is an automatic 10-bit reload from the registers S0RELL and S0RELH. The lower 8 bits of the timer are reloaded from S0RELL, while the upper two bits are reloaded from bit 0 and 1 of register S0RELH. The baud rate timer is reloaded by writing to S0RELL.

Special Function Register S0RELH (Address BA_H) **Reset Value : $XXXXXX11_B$**
Special Function Register S0RELL (Address AA_H) **Reset Value : $D9_H$**

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
BA_H	–	–	–	–	–	–	MSB	.0	S0RELH
	7	6	5	4	3	2	1	0	
AA_H	.7	.6	.5	.4	.3	.2	.1	LSB	S0RELL

Bit	Function
S0RELH.0-1	Baudrate generator for serial interface 0 reload high value Upper two bits of the baudrate timer reload value.
S0RELL.0-7	Baudrate generator for serial interface 0 reload low value Lower 8 bits of the baudrate timer reload value.

After reset S0RELH and S0RELL have a reload value of $3D9_H$. With this reload value the baud rate generator has an overflow rate of input clock/39. With this reset value and a 12-MHz oscillator frequency, the commonly used baud rates 4800 baud (SMOD = 0) and 9600 baud (SMOD = 1) are available (with 0.16 % deviation).

With the baud rate generator as clock source for the serial port 0 in mode 1 and 3, the baud rate of can be determined as follows:

$$\text{Mode 1, 3 baud rate} = \frac{2^{SMOD} \times \text{oscillator frequency}}{64 \times (\text{baud rate generator overflow rate})}$$

$$\text{Baud rate generator overflow rate} = 2^{10} - \text{S0REL}$$
$$\text{with S0REL} = \text{S0RELH.1} - 0, \text{S0RELL.7} - 0$$

Using Timer 1 for Baud Rate Generation

In mode 1 and 3 of serial interfacel 0 timer 1 can be used for generating baud rates. Then the baud rate is 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 is usually disabled in this application. Timer 1 itself can be configured for either "timer" or "counter" operation, and in any of its operating modes. In most typical applications, it is configured for "timer" operation in the auto-reload mode (high nibble of TMOD = 0010_B). In this case the baud rate is given by the formula:

$$\text{Mode 1, 3 baud rate} = \frac{2^{SMOD} \times \text{oscillator frequency}}{32 \times 12 \times (256 - (TH1))}$$

Very low baud rates can be achieved with timer 1 if leaving the timer 1 interrupt enabled, configuring the timer to run as 16-bit timer (high nibble of TMOD = 0001_B), and using the timer 1 interrupt for a 16-bit software reload.

Table 6-13 lists various commonly used baud rates and shows how these baud rates can be obtained from timer 1 or from the baud rate generator.

Table 6-13
Commonly used Baud Rates

Baud Rate		f_{osc} (MHz)	SMOD	BD	Timer 1	
					Mode	Reload Value
Mode 1, 3 :	62.5 Kbaud	12.0	1	0	2	FF_H
	125 Kbaud	24.0	1	0	2	FF_H
	19.5 Kbaud	11.059	1	0	2	FD_H
	9.6 Kbaud	11.059	0	0	2	FD_H
	4.8 Kbaud	11.059	0	0	2	FA_H
	2.4 Kbaud	11.059	0	0	2	$F4_H$
	1.2 Kbaud	11.059	0	0	2	$E8_H$
	110 Baud	6.0	0	0	2	72_H
	110 Baud	12.0	0	0	1	$FEEB_H$
					Baud Rate Generator Reload value	
	375 Kbaud	12.0	1	1	$3FF_H$	
	562.5 Kbaud	18.0	1	1	$3FF_H$	
	750 Kbaud	24.0	1	1	$3FF_H$	
	9.6 Kbaud	12.0	1	1	$3D9_H$	
	9.6 Kbaud	18.0	1	1	$3C5_H$	
	9.6 Kbaud	24.0	1	1	$3B2_H$	
Mode 0 :	1 Mbaud	12.0	-	-	-	-
	1.5 Mbaud	18.0	-	-	-	-
	2 Mbaud	24.0	-	-	-	-
Mode 2 :	187.5 Kbaud	12.0	0	-	-	-
	375 Kbaud	12.0	1	-	-	-
	281 Kbaud	18.0	0	-	-	-
	562.5 Kbaud	18.0	1	-	-	-
	375 Kbaud	24.0	0	-	-	-
	750 Kbaud	24.0	1	-	-	-

6.5.2 Serial Interface 1

6.5.2.1 Operating Modes of Serial Interface 1

The serial interface 1 is an asynchronous unit only and is able to operate in two modes, as an 8-bit or 9-bit UART. These modes, however, correspond to the above mentioned modes 1, 2 and 3 of serial interface 0. The multiprocessor communication feature is identical with this feature in serial interface 0. The serial interface 1 has its own interrupt request flags RI1 and TI1 which have a dedicated interrupt vector location. The baud rate clock for this interface is generated by a dedicated baud rate generator.

Mode A: 9-bit UART, variable baud rate:

11 bits are transmitted (through TXD1) or received (through RXD1): a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). On transmission, the 9th data bit (TB81 in S1CON) can be assigned to the value of 0 or 1. For example, the parity bit (P in the PSW) could be moved into TB81 or a second stop bit by setting TB81 to 1. On reception the 9th data bit goes into RB81 in special function register S1CON, while the stop bit is ignored. In fact, mode A of serial interface 1 is identical with mode 2 or 3 of serial interface 0 in all respects except the baud rate generation.

Mode B: 8-bit UART, variable baud rate:

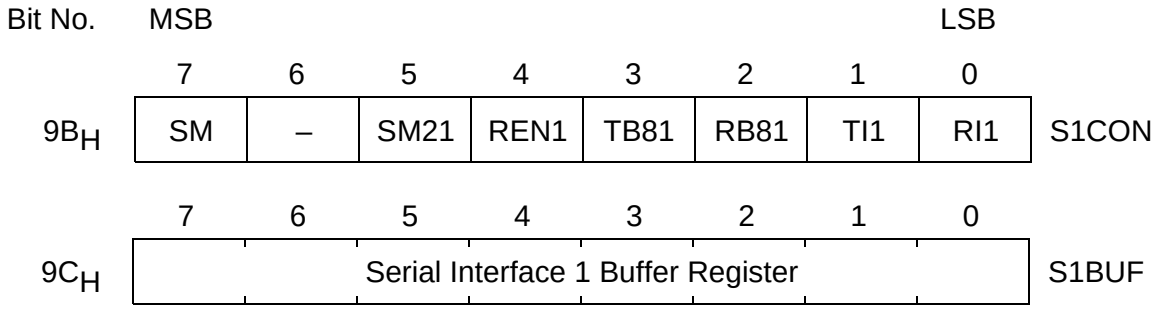
10 bits are transmitted (through TXD1) or received (through RXD1): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On reception, the stop bit goes into RB81 in special function register S1CON. In fact, mode B of serial interface 1 is identical with mode 1 of serial interface 0 in all respects except for the baud rate generation.

In both modes, transmission is initiated by any instruction that uses S1BUF as a destination register. Reception is initiated by the incoming start bit if REN1 = 1. The serial interfaces also provide interrupt requests when a transmission or a reception of a frame has completed. The corresponding interrupt request flags for serial interface 1 are TI1 or RI1, respectively. The interrupt request flags TI1 and RI1 can also be used for polling the serial interface 1 if the serial interrupt shall not be used (i.e. serial interrupt 1 not enabled).

The control and status bits of the serial channel 1 in special function register S1CON and the transmit/receive data register S1BUF are shown on the next page. Writing to S1BUF loads the transmit register and initiates transmission. Reading out S1BUF accesses a physically separate receive register. Note that these special function registers are not bit-addressable. Due to this fact bit instructions cannot be used for manipulating these registers. This is important especially for S1CON where a polling and resetting of the RI1 or TI1 request flag cannot be performed by JNB and CLR instructions but must be done by a sequence of byte instructions, e.g.:

```
LOOP:   MOV     A,S1CON
        JNB     ACC.0,LOOP      ;Testing of RI1
        ANL     S1CON,#0FEH     ;Resetting of RI1
```

Special Function Register S1CON (Address $9B_H$) **Reset Value : $0X000000_B$**
Special Function Register S1BUF (Address $9C_H$) **Reset Value : XX_H**



Bit	Function
SM	Serial port 1 mode select bit SM = 0 : Serial mode A; 9-bit UART SM = 1 : Serial mode B; 8-bit UART
SM21	Enable serial port 1 multiprocessor communication in mode A If SM21 is set to 1 in mode A, RI1 will not be activated if the received 9th data bit (RB81) is 0. In mode B, if SM21 = 1, RI1 will not be activated if a valid stop bit was not received.
REN1	Enable receiver of serial port 1 Set by software to enable serial reception. Cleared by software to disable reception.
TB81	Serial port 1 transmitter bit 9 TB81 is the 9th data bit that will be transmitted in mode A. Set or cleared by software as desired.
RB81	Serial port 1 receiver bit 9 RB81 is the 9th data bit that was received in mode A. In mode B, if SM21 = 0, RB81 is the stop bit that was received.
TI1	Serial port 1 transmitter interrupt flag TI1 is set by hardware at the beginning of the stop bit in any serial transmission. TI1 must be cleared by software.
RI1	Serial port 1 receiver interrupt flag RI1 is set by hardware at the halfway through the stop bit time in any serial reception. RI1 must be cleared by software.

6.5.2.2 Multiprocessor Communication Feature

Mode A of the serial interface 1 has a special provision for multiprocessor communication. In this mode, 9 data bits are received. The 9th bit goes into RB81. Then a stop bit follows. The port can be programmed such that when the stop bit is received, the serial port 1 interrupt will be activated (i.e. the request flag RI1 is set) only if RB81 = 1. This feature is enabled by setting bit SM21 in S1CON. A way to use this feature in multiprocessor communications is as follows.

If the master processor wants to transmit a block of data to one of the 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 SM21 = 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 see if it is being addressed. The addressed slave will clear its SM21 bit and prepare to receive the data bytes that will be coming. After having received a complete message, the slave is setting SM21 again. The slaves that were not addressed leave their SM21 set and go on about their business, ignoring the incoming data bytes.

In mode B, SM21 can be used to check the validity of the stop bit. If SM21 = 1 in mode B, the receive interrupt will not be activated unless a valid stop bit is received.

6.5.2.3 Baud Rates of Serial Channel 1

As already mentioned serial interface 1 uses its own dedicated baud rate generator for baud rate generation in both operating modes (see **figure 6-35**). This baud rate generator consists of a free running 10-bit timer with $f_{OSC}/2$ input frequency. On overflow of this timer (next count step after counter value $3FF_H$) there is an automatic 10-bit reload from the registers S1RELL and S1RELH. The lower 8 bits of the timer are reloaded from S1RELL, while the upper two bits are reloaded from bit 0 and 1 of register S1RELH. The baud rate timer is reloaded by writing to S1RELL.

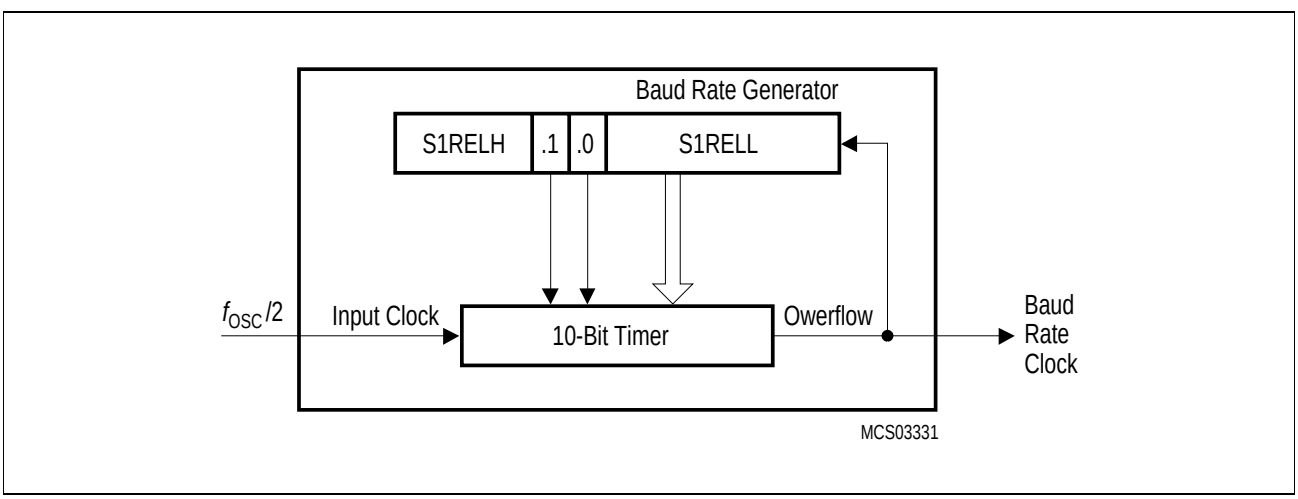


Figure 6-35
Baud Rate Generator for Serial Interface 1

Special Function Register S1RELH (Address BB_H) **Reset Value : $XXXXXX11_B$**
Special Function Register S1RELL (Address $9D_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
BB_H	–	–	–	–	–	–	MSB	.0	S1RELH
	7	6	5	4	3	2	1	0	
$9D_H$	.7	.6	.5	.4	.3	.2	.1	LSB	S1RELL

Bit	Function
S1RELH.0-1	Baudrate generator for serial interface 1 reload high value Upper two bits of the baudrate timer reload value.
S1RELL.0-7	Baudrate generator for serial interface 1 reload low value Lower 8 bits of the baudrate timer reload value.

The baud rate in operating modes A and B can be determined by following formula:

$$\text{Mode A, B baud rate} = \frac{\text{oscillator frequency}}{32 \times (\text{baud rate generator overflow rate})}$$

Baud rate generator overflow rate = 2^{10} – S1REL
with S1REL = S1RELH.1 – 0, S1RELL.7 – 0

6.5.3 Detailed Description of the Operating Modes

The following sections give a more detailed description of the several operating modes of the two serial interfaces.

6.5.3.1 Mode 0, Synchronous Mode (Serial Interface 0)

Serial data enters and exits through RXD0. TXD0 outputs the shift clock. 8 data bits are transmitted/ received (LSB first). The baud rate is fixed at 1/12 of the oscillator frequency.

Figure 6-36 shows a simplified functional diagram of the serial port in mode 0. The associated timing is illustrated in **figure 6-37**.

Transmission is initiated by any instruction that uses S0BUF as a destination register. The "Write-to-S0BUF" signal at S6P2 also loads a 1 into the 9th bit position of the transmit shift register and tells the TX control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "Write-to-S0BUF" and activation of SEND.

SEND enables the output of the shift register to the alternate output function line P3.0, and also enables SHIFT CLOCK to the alternate output function line P3.1. SHIFT CLOCK is low during S3, S4, and S5 of every machine cycle, and high during S6, S1, and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the transmit shift register is shifted one position to the right.

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 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 deactivates SEND and sets TI0. Both of these actions occur at S1P1 in the 10th machine cycle after "Write-to-S0BUF".

Reception is initiated by the condition REN0 = 1 and RI0 = 0. At S6P2 in the next machine cycle, the RX control unit writes the bits 1111 1110 to the receive shift register, and in the next clock phase activates RECEIVE.

RECEIVE enables SHIFT CLOCK to the alternate output function line of P3.1. SHIFT CLOCK makes transitions at S3P1 and S6P1 in every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted one position to the left. The value that comes in from the right is the value that was sampled at the P3.0 pin at S5P2 in the same machine cycle.

As data bits come in from the right, 1 s 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 S0BUF. At S1P1 in the 10th machine cycle after the write to S0CON that cleared RI0, RECEIVE is cleared and RI0 is set.

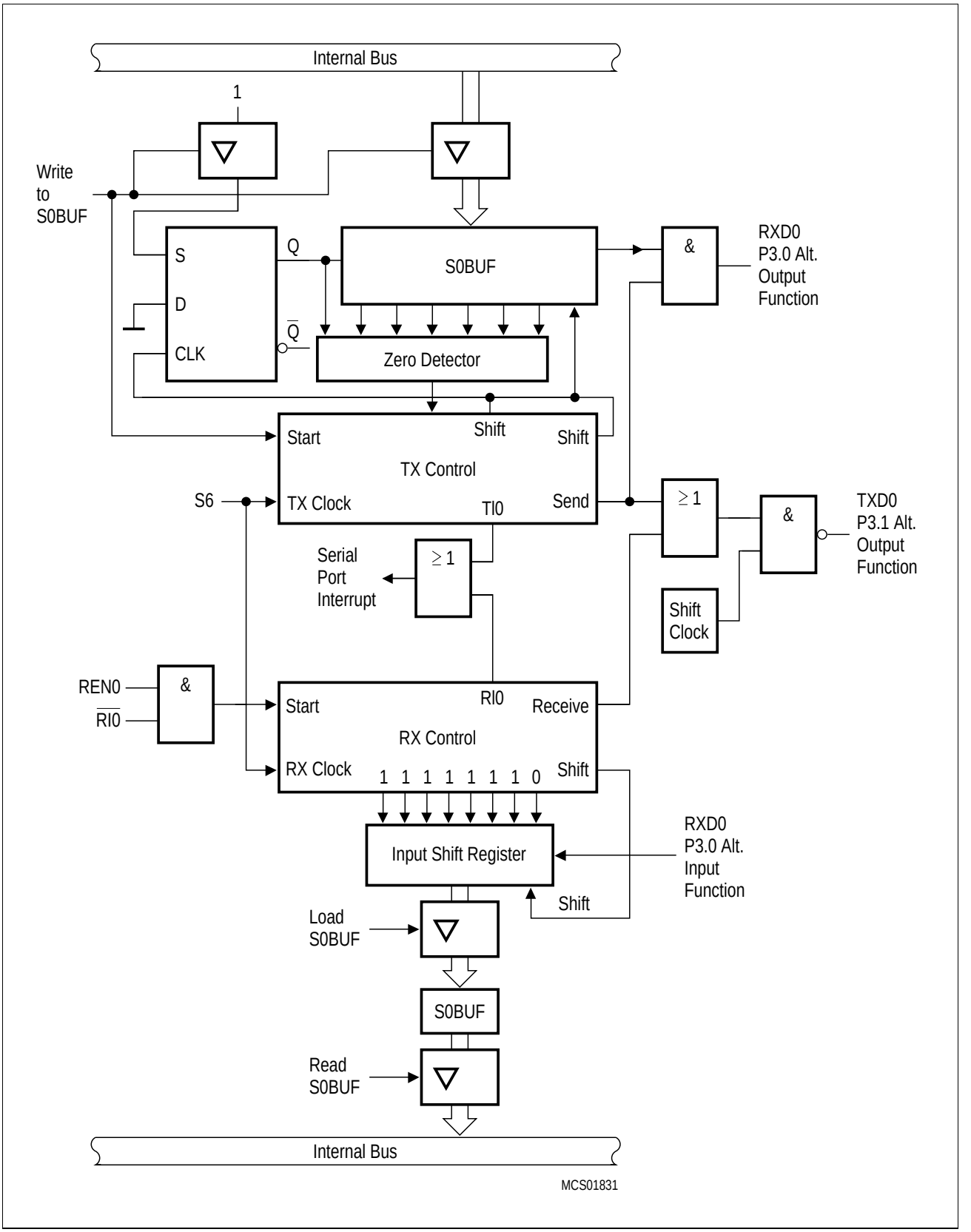


Figure 6-36
Functional Diagram - Serial Interface 0, Mode 0

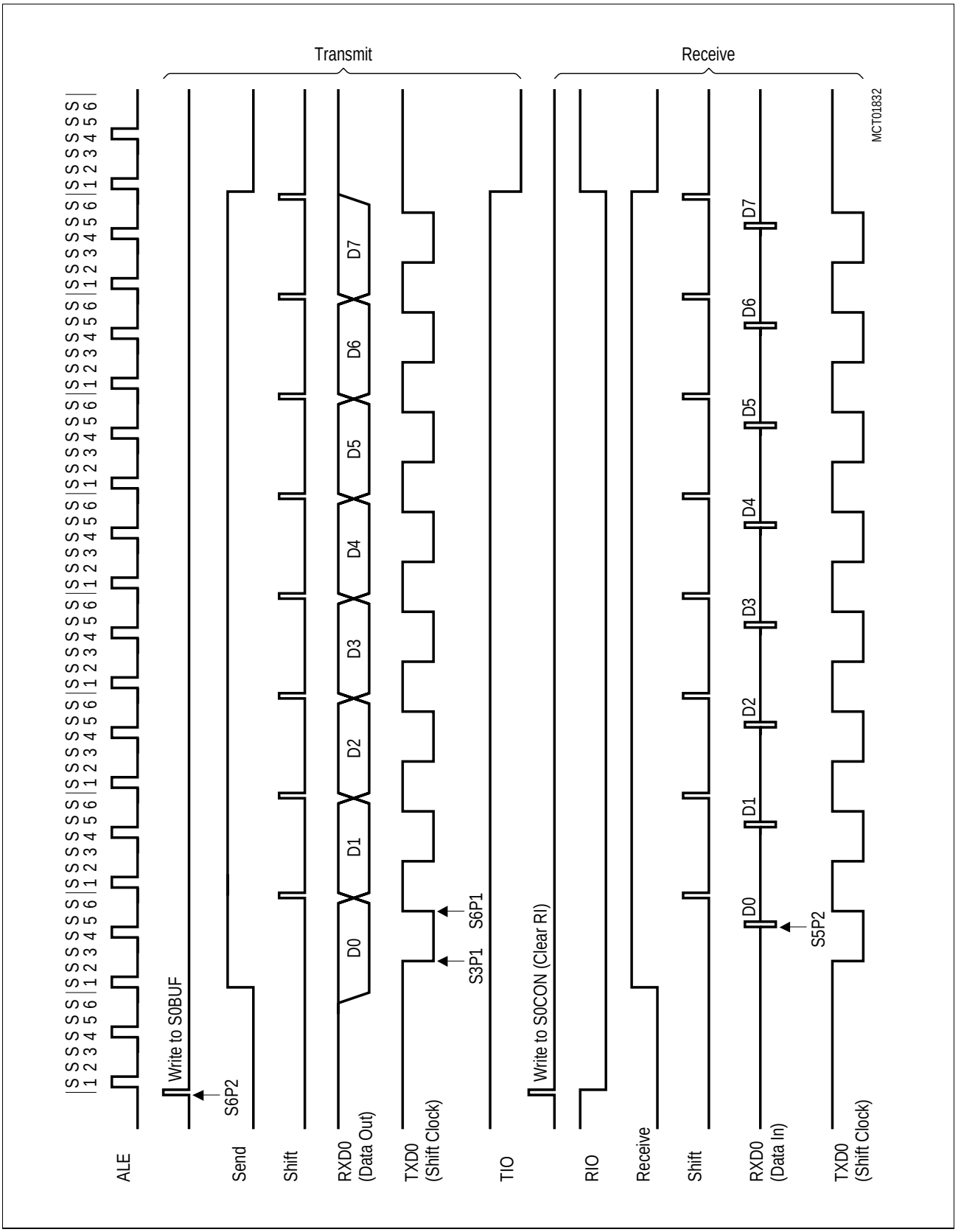


Figure 6-37
Timing Diagram - Serial Interface 0, Mode 0

6.5.3.2 Mode 1/Mode B, 8-Bit UART (Serial Interfaces 0 and 1)

Ten bits are transmitted (through TXD0 or TXD1), or received (through RXD0 or RXD1): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On reception through RXD0, the stop bit goes into RB80 (S0CON), on reception through RXD1, RB81 (S1C0N) stores the stop bit.

The baud rate for serial interface 0 is determined by the timer 1 overflow rate or by the internal baud rate generator of serial interface 0. Serial interface 1 receives the baud rate clock from its own baud rate generator.

Figure 6-38 shows a simplified functional diagram of the both serial channels in mode 1 or mode B, respectively. The associated timing is illustrated in **figure 6-39**.

Transmission is initiated by any instruction that uses S0BUF/S1BUF as a destination register. The "write-to-S0BUF/S1BUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX control block that a transmission is requested. Transmission actually commences at S1P1 of the machine cycle following the next roll-over in the divide-by-16 counter (thus, the bit times are synchronized to the divide-by-16 counter, not to the "write-to-S0BUF/S1BUF" signal).

The transmission begins with activation of $\overline{\text{SEND}}$, which puts the start bit to TXD0/TXD1. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD0/ TXD1. 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 to do one last shift and then deactivate $\overline{\text{SEND}}$ and set TI0/TI1. This occurs at the 10th divide-by-16 rollover after "write-to-S0BUF/S1BUF".

Reception is initiated by a detected 1-to-0 transition at RXD0/RXD1. For this purpose RXD0/RXD1 is sampled at a rate of 16 times whatever baud rate has been established. When a reception is detected, the divide-by-16 counter is immediately reset, and $1FF_H$ is written into the input shift register, and reception of the rest of the frame will proceed.

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 RXD0/RXD1. 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 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 from the right, 1’s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in mode 1/B is a 9-bit register), it flags the RX control block to do one last shift. The signal to load S0BUF/S1BUF and RB80/RB81, and to set RI0/RI1 will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1) RI0/RI1 = 0, and
2) either SM20/SM21 = 0 or the received stop bit = 1

If one of these two conditions is not met the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB80/RB81, the 8 data bits go into S0BUF/S1BUF, and RI0/RI1 is activated. At this time, no matter whether the above conditions are met or not, the unit goes back to looking for a 1-to-0 transition in RxD0/RxD1.

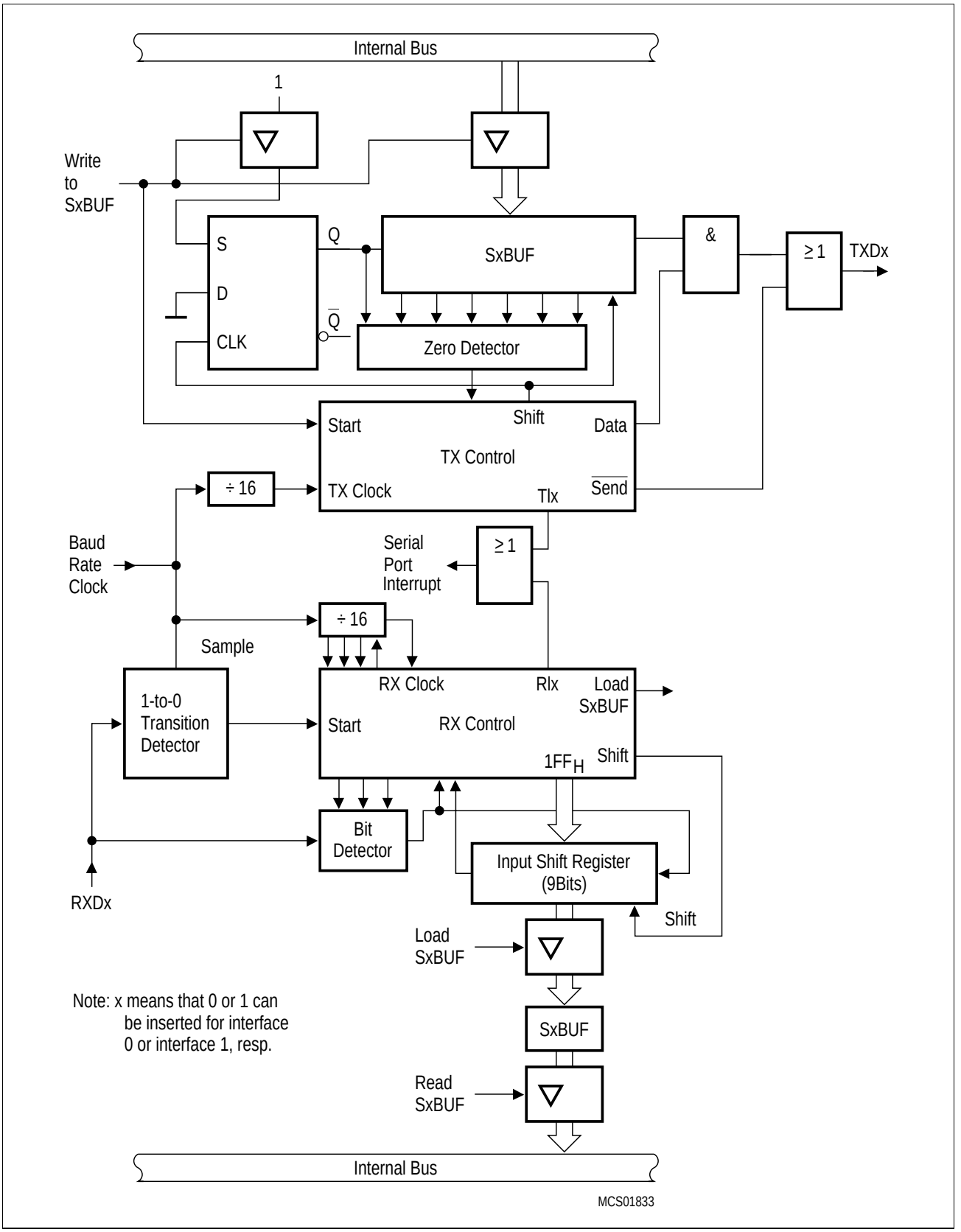


Figure 6-38
Functional Diagram - Serial Interfaces 0 and 1, Mode 1 / Mode B

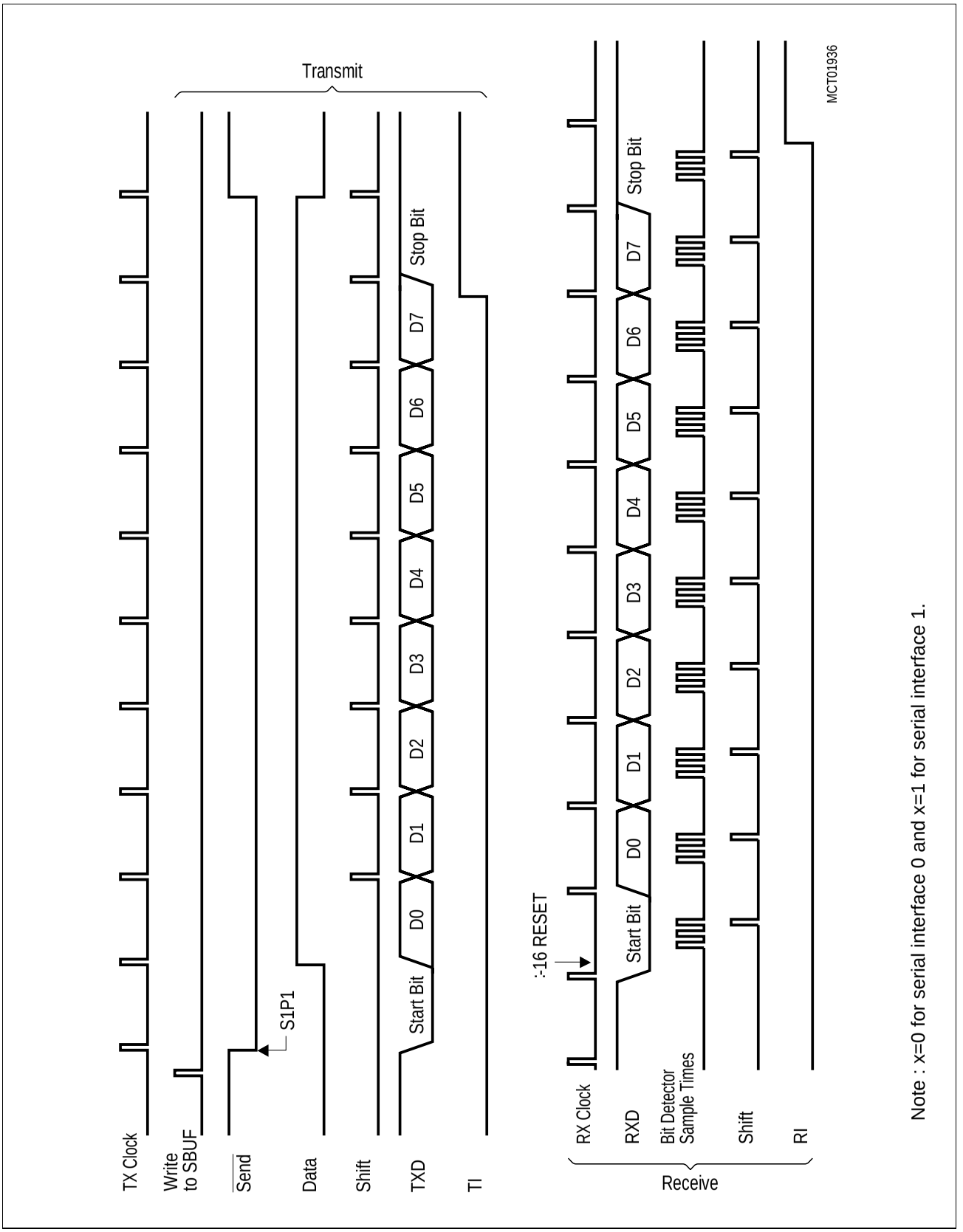


Figure 6-39
Timing Diagram - Serial Interfaces 0 and 1, Mode 1 / Mode B

6.5.3.3 Mode 2, 9-Bit UART (Serial Interface 0)

Mode 2 is functionally identical to mode 3 (see below). The only exception is, that in mode 2 the baud rate can be programmed to two fixed quantities: either 1/32 or 1/64 of the oscillator frequency. Note that serial interface 0 cannot achieve this baud rate in mode 3. Its baud rate clock is generated by timer 1, which is incremented by a rate of $f_{OSC}/12$. The dedicated baud rate generator of serial interface 1 however is clocked by a $f_{OSC}/2$ signal and so its maximum baud rate is $f_{OSC}/32$.

6.5.3.4 Mode 3 / Mode A, 9-Bit UART (Serial Interfaces 0 and 1)

Eleven bits are transmitted (through TXD0/TXD1), or received (through RXD0/RXD1): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmission, the 9th data bit (TB80/TB81) can be assigned the value of 0 or 1. On reception the 9th data bit goes into RB80/ RB81 in S0CON/S1CON. Mode 3 may have a variable baud rate generated from either timer 1 or 2 depending on the state of TCLK and RCLK in SFR T2CON.

Figure 6-40 shows a simplified functional diagram of the both serial channels in mode 2 an 3 or mode A, respectively. The associated timing is illustrated in **figure 6-41**. 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 S0BUF/S1BUF as a destination register. The "write to S0BUF/S1BUF" signal also loads TB80/TB81 into the 9th bit position of the transmit shift register and flags the TX control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter (thus the bit times are synchronized to the divide-by-16 counter, and not to the "write-to-S0BUF/S1BUF" signal).

The transmission begins with the activation of $\overline{\text{SEND}}$, which puts the start bit to TXD0/TXD1. One bit time later, DATA is activated which enables the output bit of transmit shift register to TXD0/TXD1. 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 shift out to the right, zeros are clocked in from the left. When TB80/TB81 is at the output position of the shift register, then the stop bit is just left of the TB80/TB81, 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 TI0/TI1. This occurs at the 11th divide-by-16 rollover after "write-to-S0BUF/S1BUF".

Reception is initiated by a detected 1-to-0 transition at RXD0/RXD1. For this purpose RXD0/RXD1 is sampled of 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 $1FF_H$ is written to the input shift register.

At the 7th, 8th and 9th counter state of each bit time, the bit detector samples the value of RxD0/ RxD1. 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 from the right, 1's shift out to the left. When the start bit arrives at the leftmost position in the shift register (which is a 9-bit register), it flags the RX control block to do one last shift, load S0BUF/S1BUF and RB80/ RB81, and set RI0/RI1. The signal to load S0BUF/S1BUF and RB80/RB81, and to set RI0/RI1, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1) RI0/RI1 = 0, and
2) either SM20/SM21 = 0 or the received 9th data bit = 1

If either one of these two conditions is not met, the received frame is irretrievably lost, and RI0/RI1 is not set. If both conditions are met, the received 9th data bit goes into RB80/RB81, the first 8 data bits go into S0BUF/S1BUF. One bit time later, no matter whether the above conditions are met or not, the unit goes back to look for a 1-to-0 transition at the RXD0/RXD1 input.

Note that the value of the received stop bit is irrelevant to S0BUF/S1BUF, RB80/RB81, or RI0/RI1.

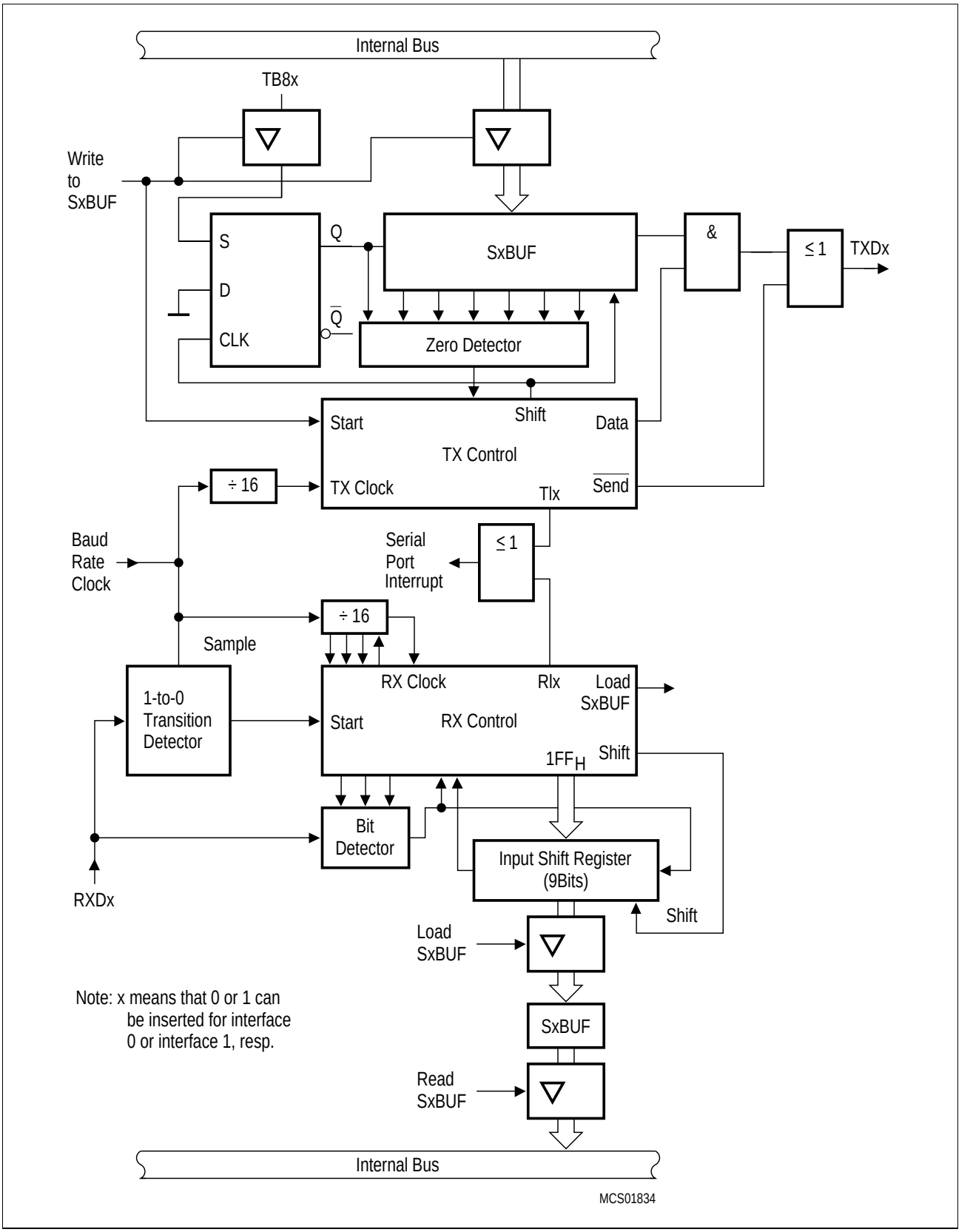


Figure 6-40
Functional Diagram - Serial Interfaces 0 and 1, Modes 2 and 3 / Mode A

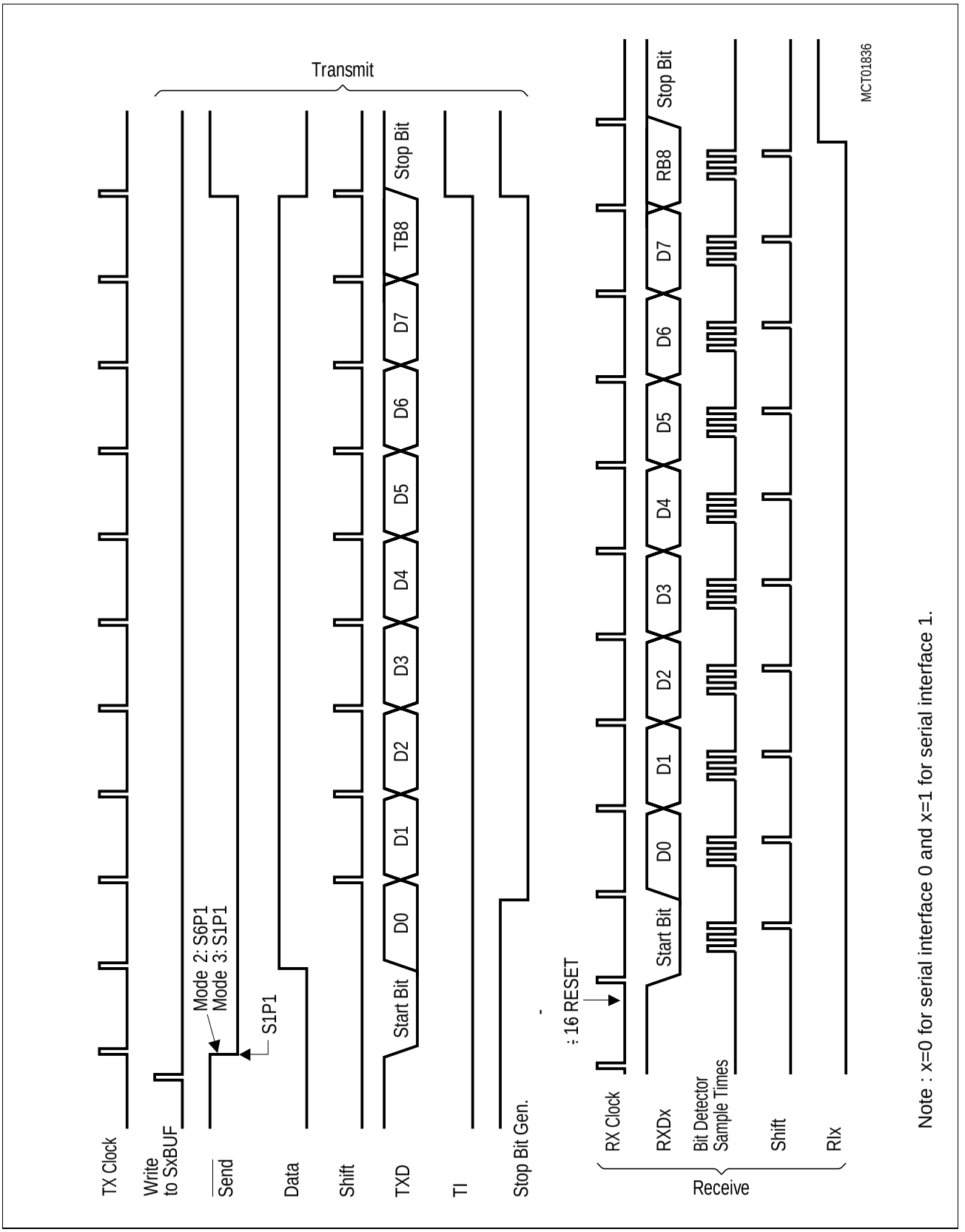


Figure 6-41
Timing Diagram - Serial Interfaces 0 and 1, Modes 2 and 3 / Mode A

6.6 10-bit A/D Converter

The C517A includes a high performance / high speed 10-bit A/D-Converter (ADC) with 12 analog input channels. It operates with a successive approximation technique and uses self calibration mechanisms for reduction and compensation of offset and linearity errors. The A/D converter provides the following features:

- 12 multiplexed input channels (port 7, 8), which can also be used as digital inputs
- 10-bit resolution
- Single or continuous conversion mode
- Internal or external start-of-conversion trigger capability
- Interrupt request generation after each conversion
- Using successive approximation conversion technique via a capacitor array
- Built-in hidden calibration of offset and linearity errors

The externally applied reference voltage range has to be held on a fixed value within the specifications.The main functional blocks of the A/D converter are shown in **figure 6-42**.

6.6.1 A/D Converter Operation

An internal start of a single A/D conversion is triggered by a write-to-ADDATL instruction. The start procedure itself is independent of the value which is written to ADDATL. When single conversion mode is selected (bit ADM=0) only one A/D conversion is performed. In continuous mode (bit ADM=1), after completion of an A/D conversion a new A/D conversion is triggered automatically until bit ADM is reset.

An externally controlled conversion can be achieved by setting the bit ADEX. In this mode one single A/D conversion is triggered by a 1-to-0 transition at pin P6.0/$\overline{\text{ADST}}$ (when ADM is 0). P6.0/$\overline{\text{ADST}}$ is sampled during S5P2 of every machine cycle. When the samples show a logic high in one cycle and a logic low in the next cycle the transition is detected and the A/D conversion is started. When ADM and ADEX is set, a continuous conversion is started when pin P6.0/$\overline{\text{ADST}}$ sees a low level. Only if no A/D conversion (single or continuous) has occurred after the last reset operation, a 1-to-0 transition is required at pin P6.0/$\overline{\text{ADST}}$ for starting the continuous conversion mode externally. The continuous A/D conversion is stopped when the pin P6.0/$\overline{\text{ADST}}$ goes back to high level. The last running A/D conversion during P6.0/$\overline{\text{ADST}}$ low level will be completed.

The busy flag BSY (ADCON0.4) is automatically set when an A/D conversion is in progress. After completion of the conversion it is reset by hardware. This flag can be read only, a write has no effect. The interrupt request flag IADC (IRCON0.0) is set when an A/D conversion is completed.

The bits MX0 to MX3 in special function register ADCON0 and ADCON1 are used for selection of the analog input channel. The bits MX0 to MX2 are represented in both registers ADCON0 and ADCON1; however, these bits are present only once. Therefore, there are two methods of selecting an analog input channel: If a new channel is selected in ADCON1 the change is automatically done in the corresponding bits MX0 to MX2 in ADCON0 and vice versa.

Port 7 and 8 are dual purpose input ports. lf the input voltage meets the specified logic levels, it can also be used as digital inputs regardless of whether the pin levels are sampled by the A/D converter at the same time.

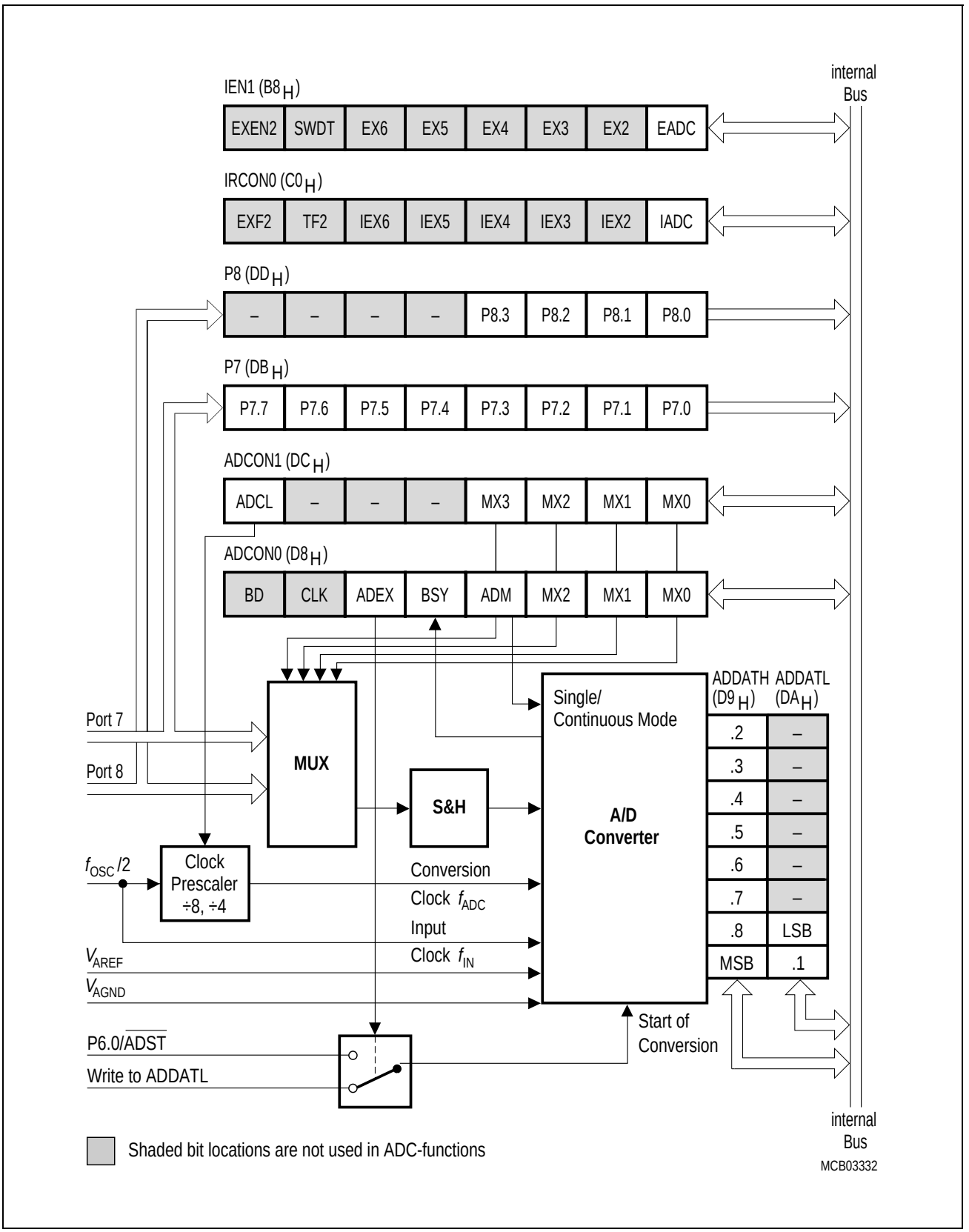


Figure 6-42
Block Diagram A/D Converter

6.6.2 A/D Converter Registers

This section describes the bits/functions of all registers which are used by the A/D converter.

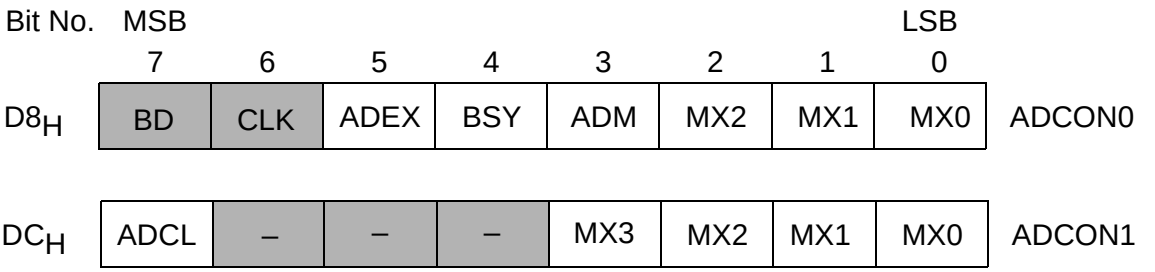
Special Function Registers ADDATH (Address $D9_H$) **Reset Value : 00_H**
Special Function Registers ADDATL (Address DA_H) **Reset Value : $00XXXXXX_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$D9_H$	MSB .9	.8	.7	.6	.5	.4	.3	.2	ADDATH
DA_H	.1	LSB .0	–	–	–	–	–	–	ADDATL

The registers ADDATH and ADDATL hold the 10-bit conversion result in left justified data format. The most significant bit of the 10-bit conversion result is bit 7 of ADDATH. The least significant bit of the 10-bit conversion result is bit 6 of ADDATL. To get a 10-bit conversion result, both ADDAT register must be read. If an 8-bit conversion result is required, only the reading of ADDATH is necessary. The data remain in ADDATH/ADDATL until it is overwritten by the next converted data. ADDAT can be read or written under software control. lf the A/D converter of the C517A is not used, register ADDATH can be used as an additional general purpose register.

Each A/D conversion is started by writing to SFR ADDATL with dummy data. If continuous conversion is selected, ADDATL must be written only once to start continuous conversion.

Special Function Registers ADCON0 (Address $D8_H$) **Reset Value : 00_H**
Special Function Registers ADCON1 (Address DC_H) **Reset Value : $0XXX0000_B$**



The shaded bits are not used for A/D converter control.

Bit	Function
–	Reserved bits for future use
ADEX	Internal / external start of converrsion When set, the external start of an A/D conversion by a falling edge at pin P6.0 / $\overline{ADST}$ is enabled.
BSY	Busy flag This flag indicates whether a conversion is in progress (BSY = 1). The flag is cleared by hardware when the conversion is finished.
ADM	A/D conversion mode When set, a continuous A/D conversion is selected. If cleared, the converter stops after one A/D conversion.
MX3 - MX0	A/D converter input channel select bits Bits MX3-0 can be written or read either in ADCON0 or ADCON1. The channel selection done by writing to ADCON 1(0) overwrites the selection in ADCON 0(1) when ADCON 1(0) is written after ADCON 0(1). The analog inputs are selected according the following table :

MX3	MX2	MX1	MX0	Selected Analog Input
0	0	0	0	P7.0 / AN0
0	0	0	1	P7.1 / AN1
0	0	1	0	P7.2 / AN2
0	0	1	1	P7.3 / AN3
0	1	0	0	P7.2 / AN4
0	1	0	1	P7.3 / AN5
0	1	1	0	P7.4 / AN6
0	1	1	1	P7.5 / AN7
1	0	0	0	P8.0 / AN8
1	0	0	1	P8.1 / AN9
1	0	1	0	P8.2 / AN10
1	0	1	1	P8.3 / AN11

Bit	Function
ADCL	A/D converter clock prescaler selection ADCL selects the prescaler ratio for the A/D conversion clock f_{ADC}. Depending on the clock rate f_{OSC} of the C517A, f_{ADC} must be adjusted in a way that the resulting f_{ADC} clock is less or equal 2 MHz. The prescaler ratio is selected according the following table :

ADCL	Prescaler Ratio
0	divide by 4 (default after reset)
1	divide by 8

Note : Generally, before entering the power-down mode, an A/D conversion in progress must be stopped. If a single A/D conversion is running, it must be terminated by polling the BSY bit or waiting for the A/D conversion interrupt. In continuous conversion mode, bit ADM must be cleared and the last A/D conversion must be terminated before entering the power-down mode.

A single A/D conversion is started by writing to SFR ADDATL with dummy data. A continuous conversion is started under the following conditions :

- By setting bit ADM during a running single A/D conversion
- By setting bit ADM when at least one A/D conversion has occured after the last reset operation.
- By writing ADDATL with dummy data after bit ADM has been set before (if no A/D conversion has occured after the last reset operation).

When bit ADM is reset by software in continuous conversion mode, the just running A/D conversion is stopped after its end.

The A/D converter interrupt is controlled by bits which are located in the SFRs IEN1 and IRCON0.

Special Function Register IEN1 (Address $B8_H$) **Reset Value : 00_H**
Special Function Register IRCON0 (Address $C0_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	BF_H	BE_H	BD_H	BC_H	BB_H	BA_H	$B9_H$	$B8_H$	
$B8_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	IEN1

	$C7_H$	$C6_H$	$C5_H$	$C4_H$	$C3_H$	$C2_H$	$C1_H$	$C0_H$	
$C0_H$	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC	IRCON0

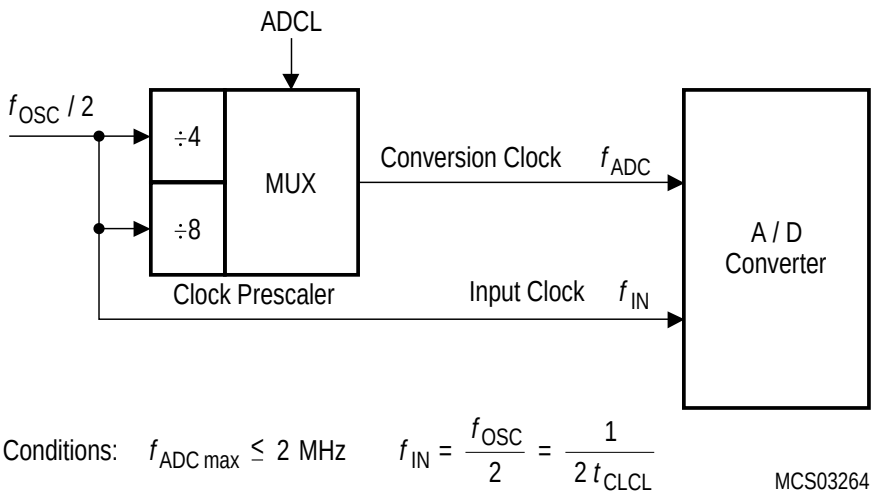
The shaded bits are not used for A/D converter control.

Bit	Function
EADC	Enable A/D converter interrupt If EADC = 0, the A/D converter interrupt is disabled.
IADC	A/D converter interrupt request flag Set by hardware at the end of an A/D conversion. Must be cleared by software.

6.6.3 A/D Converter Clock Selection

The ADC uses two clock signals for operation : the conversion clock f_{ADC} (=1/t_{ADC}) and the input clock f_{IN} (=1/t_{IN}). Both clock signals are derived from the C517A system clock f_{OSC} which is applied at the XTAL pins. The input clock f_{IN} is always $f_{OSC}/2$ while the conversion clock must be adapted to the input clock f_{OSC}. The conversion clock is limited to a maximum frequency of 2 MHz. Therefore, the ADC clock prescaler must be programmed to a value which assures that the conversion clock does not exceed 2 MHz. The prescaler is selected by the bit ADCL in SFR ADCON1.

The table in **figure 6-43** shows the prescaler ratio which must be selected for typical system clock rates. Up to 16 MHz system clock the prescaler ratio 4 is selected. Using a system clock greater than 16 MHz (max. 24 MHz) the prescaler ratio of at least 8 must be selected. The prescaler ratio 8 is recommended when the input impedance of the analog source is to high to reach the maximum accuracy.



MCU System Clock Rate (f_{OSC})	f_{IN} [MHz]	Prescaler Ratio	f_{ADC} [MHz]	ADCL1
3.5 MHz	1.75	÷ 4	.438	0
12 MHz	6	÷ 4	1.5	0
16 MHz	8	÷ 4	2	0
18 MHz	9	÷ 8	1.125	1
24 MHz	12	÷ 8	1.5	1

Figure 6-43
A/D Converter Clock Selection

The duration of an A/D conversion is a multiple of the period of the f_{IN} clock signal. The calculation of the A/D conversion time is shown in the next section.

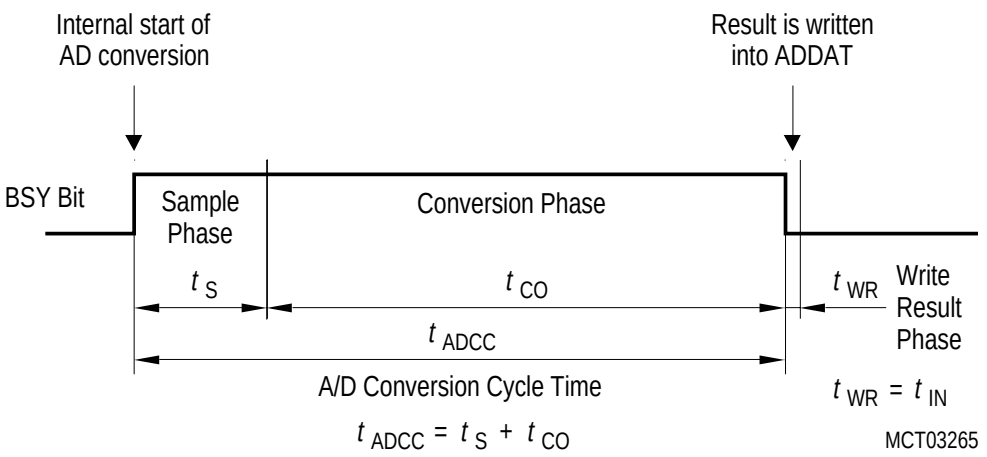
6.6.4 A/D Conversion Timing

An A/D conversion is internally started by writing into the SFR ADDATL with dummy data. A write to SFR ADDATL will start a new conversion even if a conversion is currently in progress. The conversion begins with the next machine cycle, and the BSY flag in SFR ADCON0 will be set.

The A/D conversion procedure is divided into three parts :

- Sample phase (t_S), used for sampling the analog input voltage.
- Conversion phase (t_{CO}), used for the A/D conversion (includes calibration)
- Write result phase (t_{WR}), used for writing the conversion result into the ADDAT registers.

The total A/D conversion time is defined by t_{ADCC} which is the sum of the two phase times t_S and t_{CO}. The duration of the three phases of an A/D conversion is specified by its specific timing parameter as shown in **figure 6-44**.



Selected Divider Ratio	t_S	t_{CO}	t_{ADCC}
÷ 4	8 x t_{IN}	40 x t_{IN}	48 x t_{IN}
÷ 8	16 x t_{IN}	80 x t_{IN}	96 x t_{IN}

Figure 6-44
A/D Conversion Timing

Sample Time t_S :
During this time the internal capacitor array is connected to the selected analog input channel and is loaded with the analog voltage to be converted. The analog voltage is internally fed to a voltage comparator. With beginning of the sample phase the BSY bit in SFR ADCON0 is set.

Conversion Time t_{CO} :
During the conversion time the analog voltage is converted into a 10-bit digital value using the successive approximation technique with a binary weighted capacitor network. During an A/D

conversion also a calibration takes place. During this calibration alternating offset and linearity calibration cycles are executed (see also section 6.6.5). At the end of the conversion time the BSY bit is reset and the IADC bit in SFR IRCON0 is set indicating an A/D converter interrupt condition.

Write Result Time t_{WR} :
At the result phase the conversion result is written into the ADDAT registers.

Figure 6-45 shows how an A/D conversion is embedded into the microcontroller cycle scheme using the relation 12 x t_{IN} = 1 instruction cycle. It also shows the behaviour of the busy flag (BSY) and the interrupt flag (IADC) during an A/D conversion.

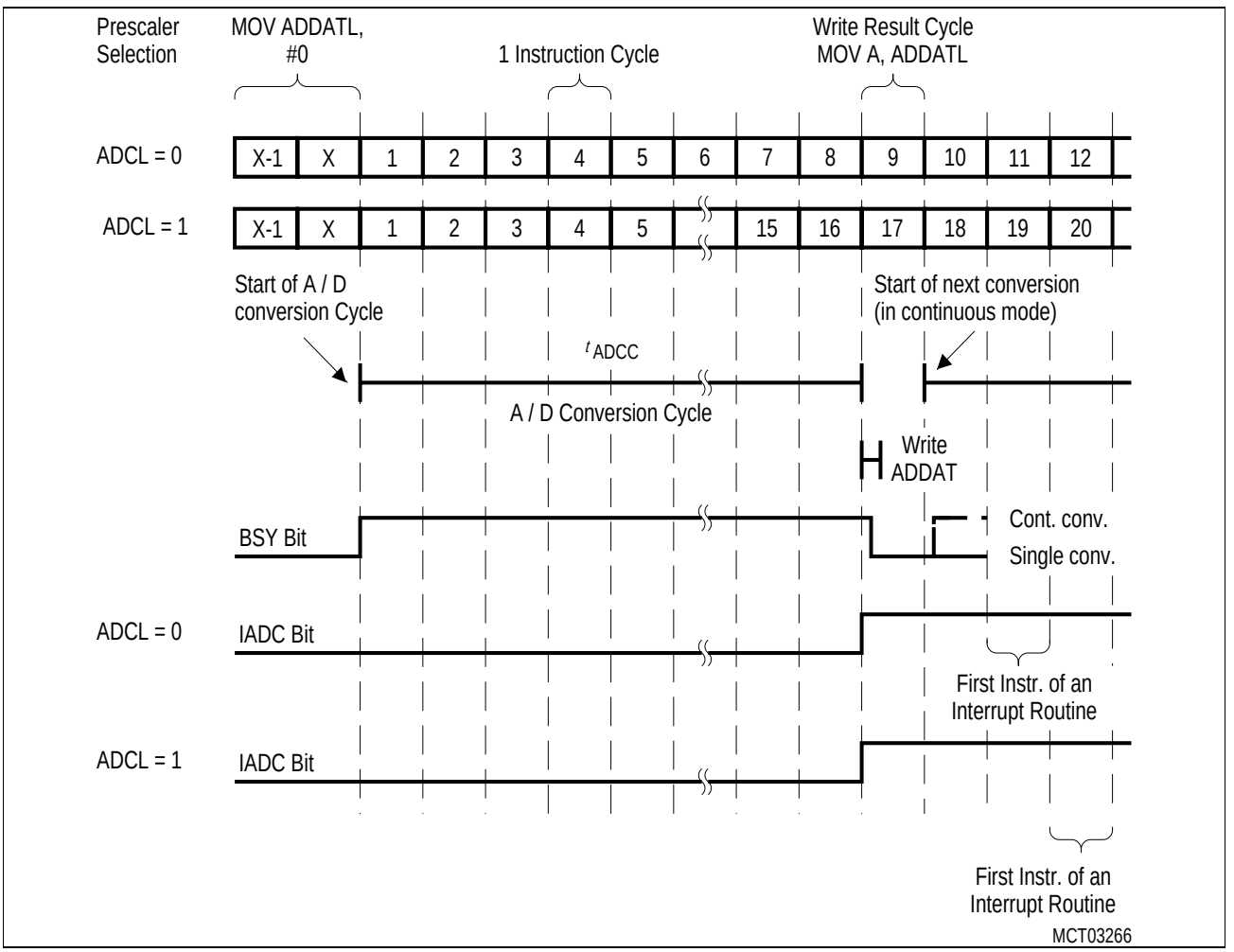


Figure 6-45
A/D Conversion Timing in Relation to Processor Cycles

Depending on the selected prescaler ratio (see **figure 6-43**), two different relationships between machine cycles and A/D conversion are possible. The A/D conversion is always started with the beginning of a processor cycle when it has been started by writing SFR ADDATL with dummy data or after an high-to-low transition has been detcted at P6.0 / $\overline{\text{ADST}}$. The ADDATL write operation may take one or two machine cycles. In **figure 6-45**, the instruction MOV ADDATL,#00 starts the A/D conversion (machine cycle X-1 and X). The total A/D conversion is finished with the end of the 8th or 16th machine cycle after the A/D conversion start. In the next machine cycle the conversion result is written into the ADDAT registers and can be read in the same cycle by an instruction (e.g.

MOV A,ADDATL). If continuous conversion is selected (bit ADM set), the next conversion is started with the beginning of the machine cycle which follows the writre result cycle..

The BSY bit is set at the beginning of the first A/D conversion machine cycle and reset at the beginning of the write result cycle. If continuous conversion is selected, BSY is again set with the beginning of the machine cycle which follows the write result cycle.

The interrupt flag IADC is set at the end of the A/D conversion. If the A/D converter interrupt is enabled and the A/D converter interrupt is priorized to be serviced immediately, the first instruction of the interrupt service routine will be executed in the third machine cycle which follows the write result cycle. IADC must be reset by software.

Depending on the application, typically there are three methods to handle the A/D conversion in the C517A .

- Software delay
  The machine cycles of the A/D conversion are counted and the program executes a software delay (e.g. NOPs) before reading the A/D conversion result in the write result cycle. This is the fastest method to get the result of an A/D conversion.
- Polling BSY bit
  The BSY bit is polled and the program waits until BSY=0. Attention : a polling JB instruction which is two machine cycles long, possibly may not recognize the BSY=0 condition during the write result cycle in the continuous conversion mode.
- A/D conversion interrupt
  After the start of an A/D conversion the A/D converter interrupt is enabled. The result of the A/D conversion is read in the interrupt service routine. If other C517A interrupts are enabled, the interrupt latency must be regarded. Therefore, this software method is the slowest method to get the result of an A/D conversion.

Depending on the oscillator frequency of the C517A and the selected divider ratio of the A/D converter prescaler the total time of an A/D conversion is calculated according **figure 6-44** and **table 6-14. Figure 6-46** on the next page shows the minimum A/D conversion time in relation to the oscillator frequency f_{OSC}. The minimum conversion time is 6 µs which can be achieved at f_{OSC} of 16 or 32 MHz.

Table 6-14
A/D Conversion Time for Dedicated System Clock Rates

f_{OSC} [MHz]	Prescaler Ratio	f_{ADC} [MHz]	Sample Time t_S [µs]	Total Conversion Time t_{ADCC} [µs]
3.5	4	.438	4.57	27.43
12	4	1.5	1.33	8
16	4	2	1	6
18	8	1.125	1.78	10.67
24	8	1.5	1.33	8

Note : The prescaler ratios in **table 6-14** are mimimum values. At system clock rates (f_{OSC}) up to 16 MHz the divider ratio 4 and 8 can be used. At system clock rates greater than 16 MHz only the divider ratio 8 can be used. Using higher divider ratios than required increases the total conversion time but can be useful in applications which have voltage sources with higher input resistances for the analog inputs (increased sample phase).

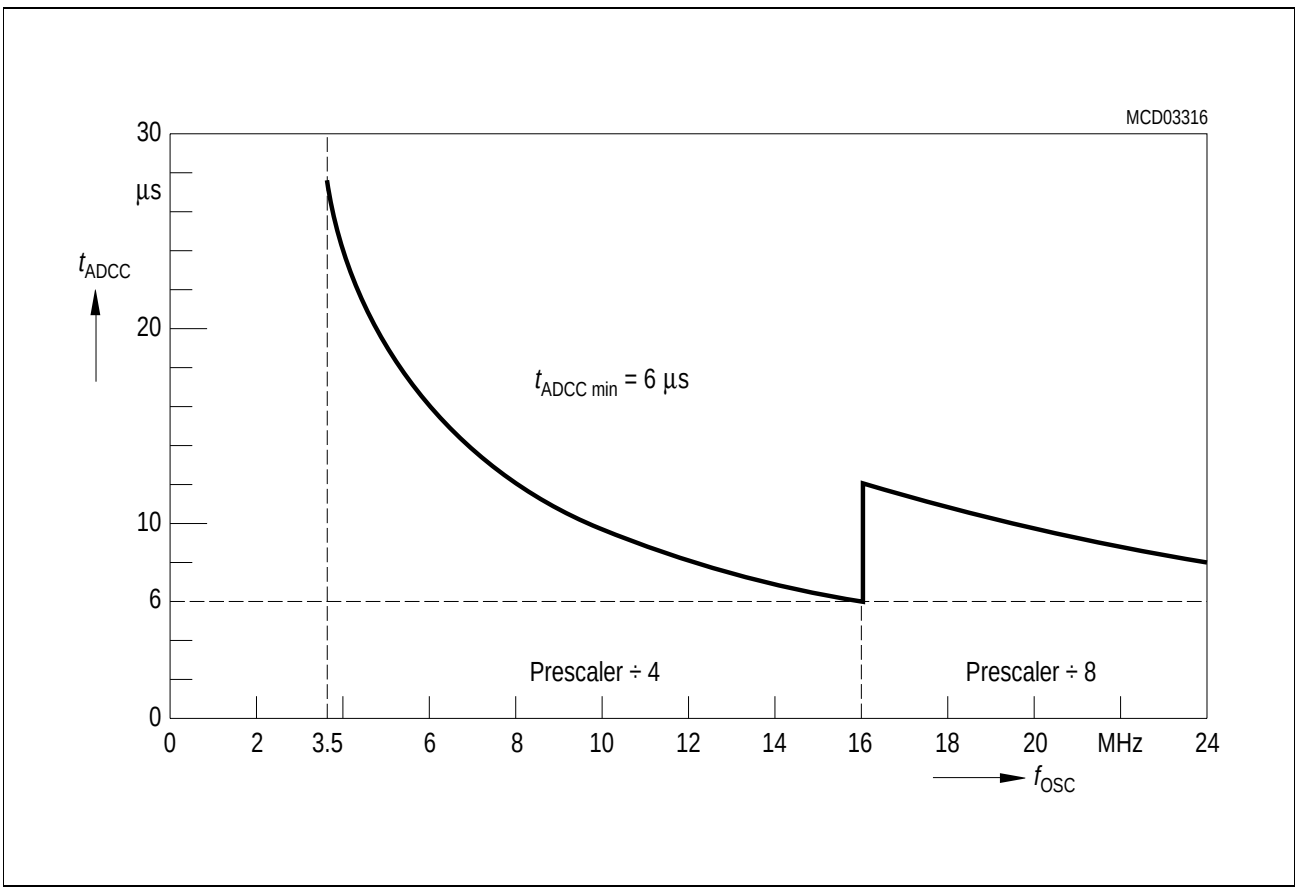


Figure 6-46
Minimum A/D Conversion Time in Relation to System Clock

6.6.5 A/D Converter Calibration

The C517A A/D converter includes hidden internal calibration mechanisms which assure a save functionality of the A/D converter according to the DC characteristics. The A/D converter calibration is implemented in a way that a user program which executes A/D conversions is not affected by its operation. Further, the user program has no control on the calibration mechanism. The calibration itself executes two basic functions :

- Offset calibration : compensation of the offset error of the internal comparator
- Linearity calibration : correction of the binary weighted capacitor network

The A/D converter calibration operates in two phases : calibration after a reset operation and calibration at each A/D conversion. The calibration phases are controlled by a state machine in the A/D converter. This state machine once executes a reset calibration phase after each reset operation of the C517A and stores the result values of the reset calibration phase after its end in an internal RAM. Further, these values are updated after each A/D conversion.

After a reset operation the A/D calibration is automatically started. This reset calibration phase which takes 3328 f_{ADC} clocks, alternating offset and linearity calibration is executed. Therefore, at 12 MHz oscillator frequency and with the default prescaler value of 4, a reset calibration time of approx. 2.2 ms is reached. The reset calibration phase is defined as follows (t_{OSC} = 1 / f_{OSC}) :

- Prescaler 4 selected : Reset calibration phase = 3328 x f_{ADC} = 13312 x t_{IN} = 26624 x t_{OSC}
- Prescaler 8 selected : Reset calibration phase = 3328 x f_{ADC} = 26624 x t_{IN} = 53248 x t_{OSC}

For achieving a proper reset calibration, the f_{ADC} prescaler value must satisfy the condition $f_{ADCmax} \leq$ 2 MHz. For oscillator frequencies above 16 MHz this condition is not met with the default prescaler value (÷4) after reset. Therefore, the prescaler of the A/D converter must be adjusted by software immediately after reset by setting bit ADCL in SFR ADCON1. When setting bit ADCL directly after reset as required for oscillator clocks greater or equal 16 MHz, the clock prescaler ratio ÷8 is selected and therefore the absolute value for the reset calibration phase will be extended by factor 2.

After a reset operation of the C517A, this means when a reset calibration phase is started, the total unadjusted error TUE of the A/D converter is ± 6 LSB. After the reset calibration phase the A/D converter is calibrated according to its DC characteristics (TUE = ± 2 LSB). Nevertheless, during the reset calibration phase single or continuous A/D can be executed. In this case it must be regarded that the reset calibration is interrupted and continued after the end of the A/D conversion. Therefore, interrupting the reset calibration phase by A/D conversions extends the total reset calibration time. If the specified total unadjusted error (TUE) has to be valid for an A/D conversion, it is recommended to start the first A/D conversions after reset when the reset calibration phase is finished.

After the reset calibration, a second calibration mechanism is initiated. This calibration is coupled to each A/D conversion**.** With this second calibration mechanism alternatively offset and linearity calibration values, stored in the calibration RAM, are always checked when an A/D conversion is executed and corrected if required.

7 Interrupt System

The C517A provides 17 interrupt sources with four priority levels. Ten interrupts can be generated by the on-chip peripherals (timer 0, timer 1, timer 2, compare timer, compare match/set/clear, A/D converter, and serial interface 0 and 1) and seven interrupts may be triggered externally (P3.2/$\overline{\text{INT0}}$, P3.3/$\overline{\text{INT1}}$, P1.4/$\overline{\text{INT2}}$, P1.0/$\overline{\text{INT3}}$, P1.1/INT4, P1.2/INT5, P1.3/INT6).

This chapter shows the interrupt structure, the interrupt vectors and the interrupt related special function registers. **Figure 7-1** to **7-3** give a general overview of the interrupt sources and illustrate the request and the control flags which are described in the next sections.

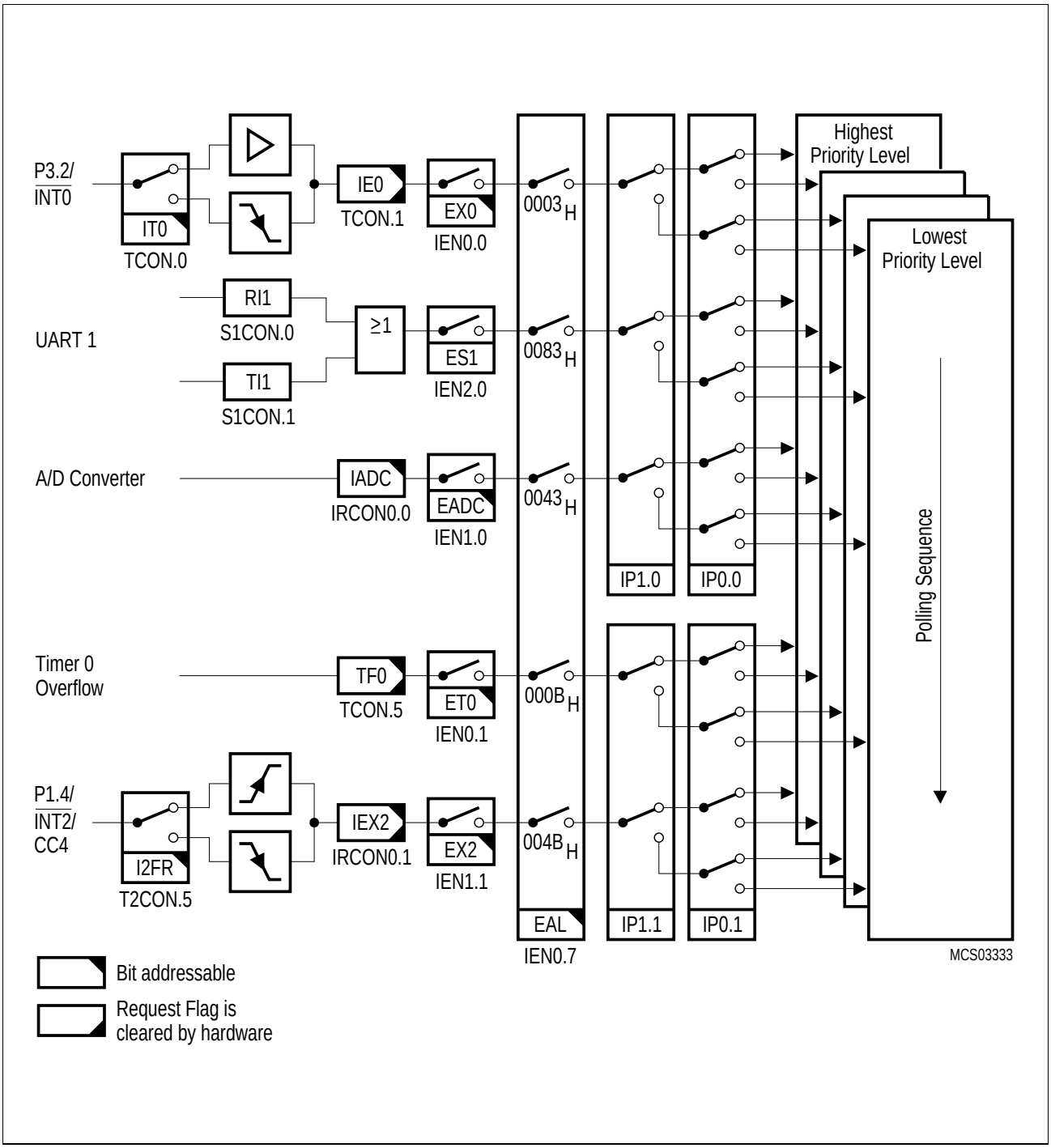


Figure 7-1
Interrupt Structure, Overview Part 1

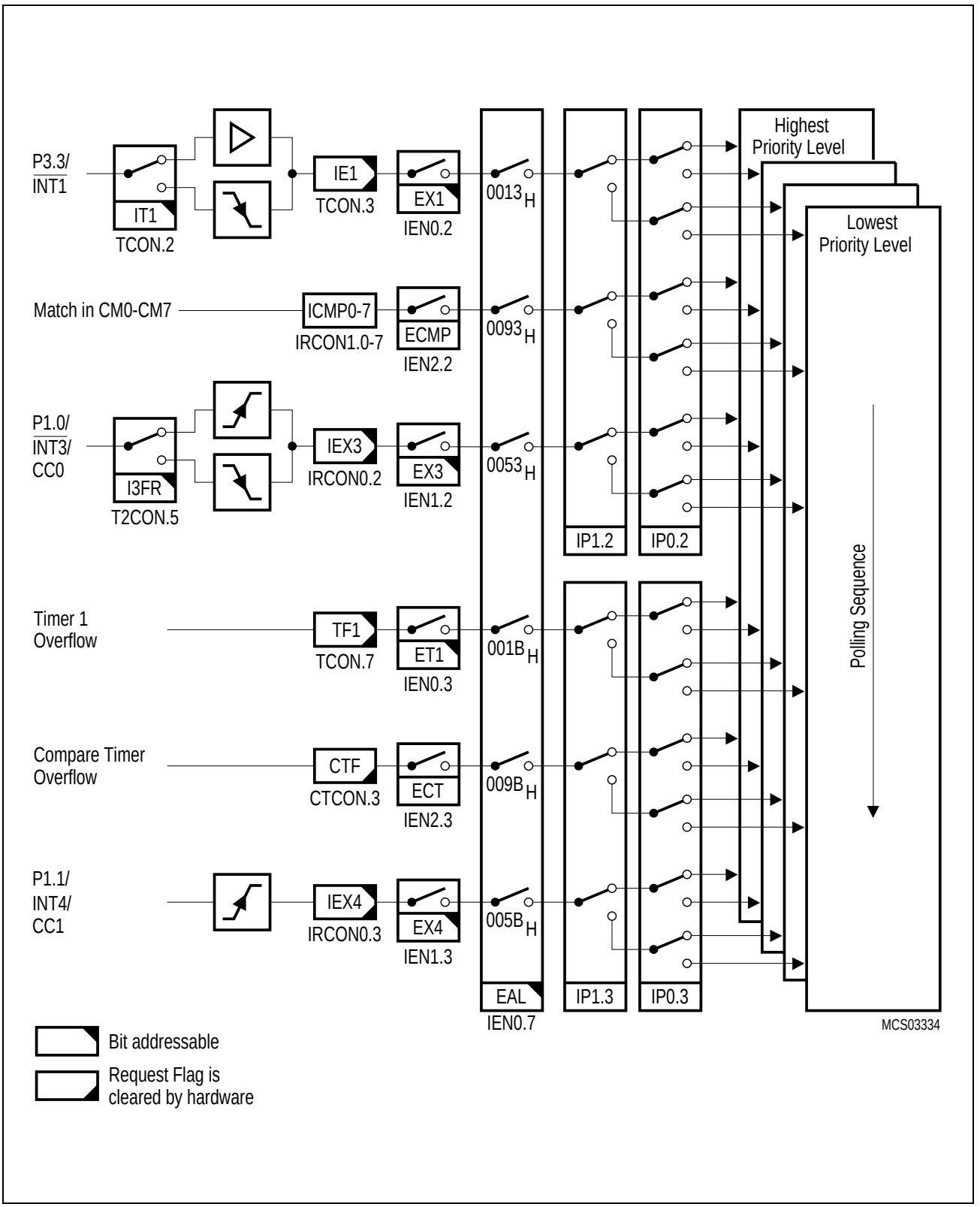


Figure 7-2 :
Interrupt Structure, Overview Part 2

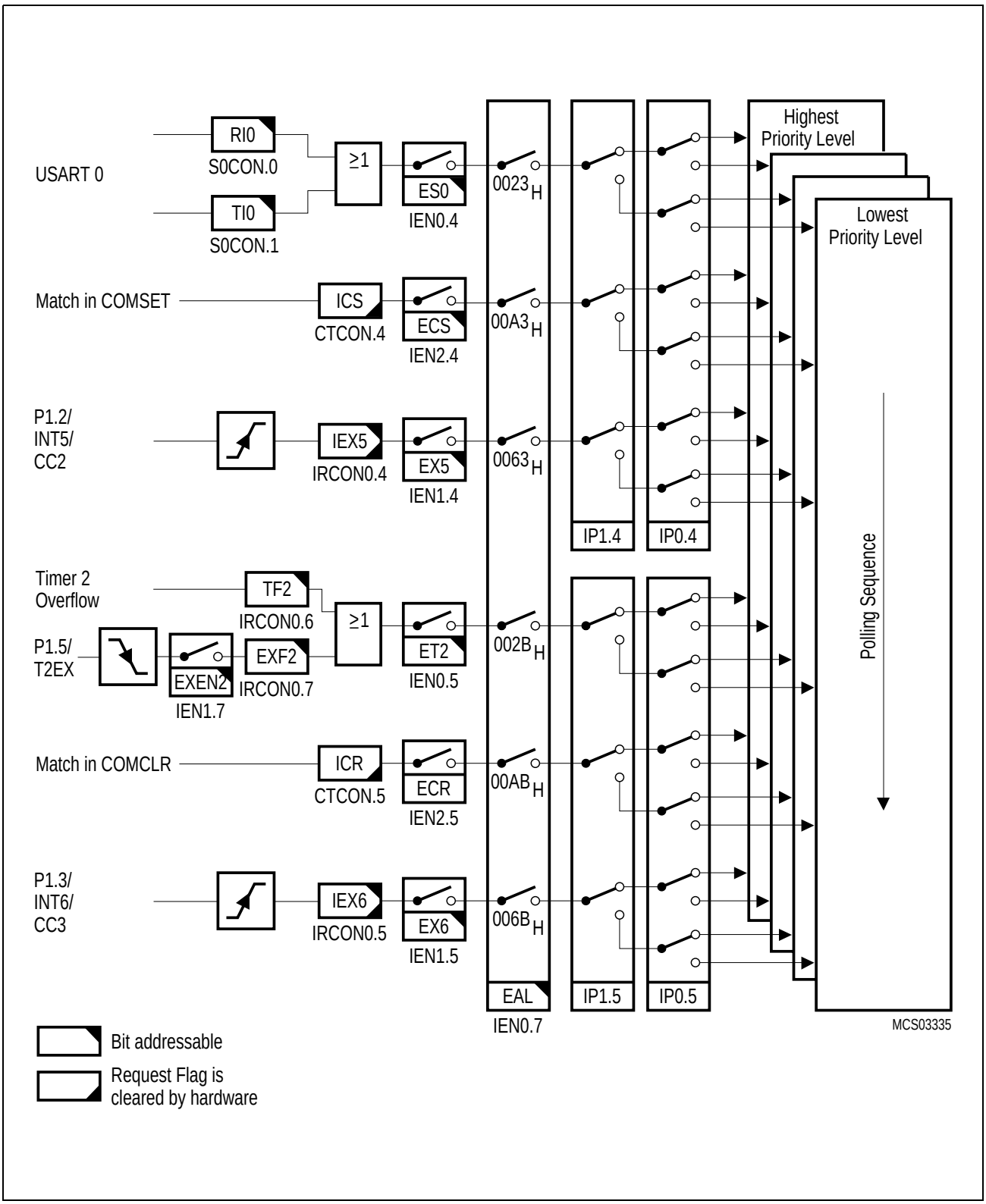


Figure 7-3 :
Interrupt Structure, Overview Part 3

7.1 Interrupt Registers

7.1.1 Interrupt Enable Registers

Each interrupt vector can be individually enabled or disabled by setting or clearing the corresponding bit in the interrupt enable registers IEN0, IEN1, and IEN2. Register IEN0 also contains the global disable bit (EAL), which can be cleared to disable all interrupts at once. Some interrupts sources have further enable bits (e.g. EXEN2). Such interrupt enable bits are controlled by specific bits in the SFRs of the corresponding peripheral units. This section describes the locations and meanings of the interrupt enable bits in detail.

After reset the enable bits of the interrupt enable registers IEN0 to IEN2 are set to 0. That means that the corresponding interrupts are disabled.

The SFR IEN0 includes the enable bits for the external interrupts 0 and 1, the timer 0,1, and 2 interrupts, the serial interface 0 interrupt, and the general interrupt enable control bit EAL.

Special Function Register IEN0 (Address $A8_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	AF_H	AE_H	AD_H	AC_H	AB_H	AA_H	$A9_H$	$A8_H$	
$A8_H$	EAL	WDT	ET2	ES0	ET1	EX1	ET0	EX0	IEN0

The shaded bit is not used for interrupt control

Bit	Function
EAL	Enable/disable all interrupts. If EA=0, no interrupt will be acknowledged. If EA=1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
ET2	Timer 2 interrupt enable. If ET2 = 0, the timer 2 interrupt is disabled.
ES0	Serial channel 0 interrupt enable If ES0 = 0, the serial channel interrupt 0 is disabled.
ET1	Timer 1 overflow interrupt enable. If ET1 = 0, the timer 1 interrupt is disabled.
EX1	External interrupt 1 enable. If EX1 = 0, the external interrupt 1 is disabled.
ET0	Timer 0 overflow interrupt enable. If ET0 = 0, the timer 0 interrupt is disabled.
EX0	External interrupt 0 enable. If EX0 = 0, the external interrupt 0 is disabled.

The SFR IEN1 includes the enable bits for the external interrupts 2 to 6, for the AD converter interrupt, and for the timer 2 external reload interrupt.

Special Function Register IEN1 (Address $B8_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	BF_H	BE_H	BD_H	BC_H	BB_H	BA_H	$B9_H$	$B8_H$	
$B8_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	IEN1

The shaded bit is not used for interrupt control

Bit	Function
EXEN2	Timer 2 external reload interrupt enable If EXEN2 = 0, the timer 2 external reload interrupt is disabled. The external reload function is not affected by EXEN2.
EX6	External interrupt 6 / capture/compare interrupt 3 enable If EX6 = 0, external interrupt 6 is disabled.
EX5	External interrupt 5 / capture/compare interrupt 2 enable If EX5 = 0, external interrupt 5 is disabled.
EX4	External interrupt 4 / capture/compare interrupt 1 enable If EX4 = 0, external interrupt 4 is disabled.
EX3	External interrupt 3 / capture/compare interrupt 0 enable If EX3 = 0, external interrupt 3 is disabled.
EX2	External interrupt 2 / capture/compare interrupt 4 enable If EX2 = 0, external interrupt 2 is disabled.
EADC	Timer 2 external reload interrupt enable If EADC = 0, the A/D converter interrupt is disabled

The SFR IEN2 includes the enable bits for the compare match with compare register interrupts, the compare timer overflow interrupt, and the serial interface 1 interrupt.

Special Function Register IEN2 (Address $9A_H$) **Reset Value : $XX0000X0_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$9A_H$	–	–	ECR	ECS	ECT	ECMP	–	ES1	IEN2

Bit	Function
–	Reserved bits for future use.
ECR	COMCLR register compare match interrupt enable If ECR = 0, the COMCLR compare match interrupt is disabled.
ECS	COMSET register compare match interrupt enable If ECS = 0, the COMSET compare match interrupt is disabled.
ECT	Enable compare timer interrupt enable If ECT = 0, the compare timer overflow interrupt is disabled.
ECMP	CM0-7 register compare match interrupt enable If ECMP = 0, the CM0-7 compare match interrupt is disabled.
ES1	Serial Interface 1 interrupt enable if ES1 = 0, the serial interrupt 1 is disabled.

7.1.2 Interrupt Request / Control Flags

The request flags for the different interrupt sources are located in several special function registers. This section describes the locations and meanings of these interrupt request flags in detail.

The external interrupts 0 and 1 (P3.2/$\overline{INT0}$ and P3.3/$\overline{INT1}$) can each be either level-activated or negative transition-activated, depending on bits IT0 and IT1 in SFR TCON. The flags that actually generate these interrupts are bits IE0 and lE1 in SFR TCON. When an external interrupt is generated, the flag that generated this interrupt is cleared by the hardware when the service routine is vectored to, but only if the interrupt was transition-activated. If the interrupt was level-activated, then the requesting external source directly controls the request flag, rather than the on-chip hardware.

The timer 0 and timer 1 interrupts are generated by TF0 and TF1 in register TCON, which are set by a rollover in their respective timer/counter registers (exception is timer 0 in mode 3). When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

Special Function Register TCON (Address 88_H) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	$8F_H$	$8E_H$	$8D_H$	$8C_H$	$8B_H$	$8A_H$	89_H	88_H	
88_H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	TCON

The shaded bits are not used for interrupt purposes.

Bit	Function
TF1	Timer 1 overflow flag Set by hardware on timer/counter 1 overflow. Cleared by hardware when processor vectors to interrupt routine.
TF0	Timer 0 overflow flag Set by hardware on timer/counter 0 overflow. Cleared by hardware when processor vectors to interrupt routine.
IE1	External interrupt 1 request flag Set by hardware. Cleared by hardware when processor vectors to interrupt routine (if IT1 = 1) or by hardware (if IT1 = 0).
IT1	External interrupt 1 level/edge trigger control flag If IT1 = 0, level triggered external interrupt 1 is selected. If IT1 = 1, negative edge triggered external interrupt 1 is selected.
IE0	External interrupt 0 request flag Set by hardware. Cleared by hardware when processor vectors to interrupt routine (if IT0 = 1) or by hardware (if IT0 = 0).
IT0	External interrupt 0 level/edge trigger control flag If IT0 = 0, level triggered external interrupt 0 is selected. If IT0 = 1, negative edge triggered external interrupt 0 is selected.

The interrupt of the serial interface 0 is generated by the request flags RI0 and TI0 in SFR S0CON. The two request flags of the serial interface are logically OR-ed together. Neither of these flags is cleared by hardware when the service routine is vectored too. In fact, the service routine of each interface will normally have to determine whether it was the receive interrupt flag or the transmission interrupt flag that generated the interrupt, and the bit will have to be cleared by software.

The interrupt of the serial interface 1 is generated by the request flags RI1 and TI1 in SFR S1CON. The two request flags of the serial interface are logically OR-ed together. Neither of these flags is cleared by hardware when the service routine is vectored too. In fact, the service routine of each interface will normally have to determine whether it was the receive interrupt flag or the transmission interrupt flag that generated the interrupt, and the bit will have to be cleared by software.

Special Function Register S0CON (Address 98_H) **Reset Value : 00_H**
Special Function Register S1CON (Address $9B_H$) **Reset Value : $0X000000_B$**

	MSB							LSB	
Bit No.	$9F_H$	$9E_H$	$9D_H$	$9C_H$	$9B_H$	$9A_H$	99_H	98_H	
98_H	SM0	SM10	SM20	REN0	TB80	RB80	TI0	RI0	S0CON

	7	6	5	4	3	2	1	0	
$9B_H$	SM	–	SM21	REN1	TB81	RB81	TI1	RI1	S1CON

The shaded bits are not used for interrupt purposes.

Bit	Function
TI0	Serial interface 0 transmitter interrupt flag Set by hardware at the end of a serial data transmission. Must be cleared by software.
RI0	Serial interface 0 receiver interrupt flag Set by hardware if a serial data byte has been received. Must be cleared by software.
TI1	Serial interface 1 transmitter interrupt flag Set by hardware at the end of a serial data transmission. Must be cleared by software.
RI1	Serial interface 1 receiver interrupt flag Set by hardware if a serial data byte has been received. Must be cleared by software.

The external interrupt 2 ($\overline{INT2}$/CC4) can be either positive or negative transition-activated depending on bit I2FR in register T2CON. The flag that actually generates this interrupt is bit IEX2 in register IRCON. In addition, this flag will be set if a compare event occurs at the corresponding output pin P1.4/$\overline{INT2}$/CC4, regardless of the compare mode established and the transition at the respective pin. If an interrupt 2 is generated, flag IEX2 is cleared by hardware when the service routine is vectored to.

Like the external interrupt 2, **the external interrupt 3** can be either positive or negative transition-activated, depending on bit I3FR in register T2CON. The flag that actually generates this interrupt is bit IEX3 in register IRCON0. In addition, this flag will be set if a compare event occurs at pin P1.0/$\overline{INT3}$/CC0, regardless of the compare mode established and the transition at the respective pin. The flag IEX3 is cleared by hardware when the service routine is vectored to.

The external interrupts 4 (INT4), 5 (INT5), and 6 (INT6) are positive transition-activated. The flags that actually generate these interrupts are bits IEX4, IEX5, and IEX6 in register IRCON0. In addition, these flags will be set if a compare event occurs at the corresponding output pin P1.1/INT4/CC1, P1.2/INT5/CC2, and P1.3/INT6/CC3, regardless of the compare mode established and the transition at the respective pin. When an interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The timer 2 interrupt is generated by the logical OR of bit TF2 in SFR T2CON and bit EXF2 in SFR IRCON0. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and the bit will have to be cleared by software.

The A/D converter interrupt is generated by IADC in register IRCON0. It is set some cycles before the result is available. That is, if an interrupt is generated, in any case the converted result in ADDATH/ADDADL is valid on the first instruction of the interrupt service routine (with respect to the minimal interrupt response time). If continuous conversions are established, IADC is set once during each conversion. If an A/D converter interrupt is generated, flag IADC must be cleared by software.

Special Function Register T2CON (Address $C8_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	CF_H	CE_H	CD_H	CC_H	CB_H	CA_H	$C9_H$	$C8_H$	
$C8_H$	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T1I0	T2CON

The shaded bits are not used for interrupt purposes.

Bit	Function
I3FR	External interrupt 3 rising/falling edge control flag If I3FR = 0, the external interrupt 3 is activated by a negative transition at $\overline{INT3}$. If I3FR = 1, the external interrupt 3 is activated by a positive transition at $\overline{INT3}$.
I2FR	External interrupt 2 rising/falling edge control flag If I3FR = 0, the external interrupt 3 is activated by a negative transition at $\overline{INT2}$. If I3FR = 1, the external interrupt 3 is activated by a positive transition at $\overline{INT2}$.

Special Function Register IRCON0 (Address $C0_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	$C7_H$	$C6_H$	$C5_H$	$C4_H$	$C3_H$	$C2_H$	$C1_H$	$C0_H$	
$C0_H$	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC	IRCON0

Bit	Function
EXF2	Timer 2 external reload flag Set when a reload is caused by a negative transition on pin T2EX while EXEN2 = 1. If ET2 in IEN0 is set (timer 2 interrupt enabled), EXF2 = 1 will cause an interrupt. Can be used as an additional external interrupt when the reload function is not used. EXF2 must be cleared by software.
TF2	Timer 2 overflow flag Set by a timer 2 overflow and must be cleared by software. If the timer 2 interrupt is enabled, TF2 = 1 will cause an interrupt.
IEX6	External interrupt 6 edge flag Set by hardware when external interrupt edge was detected or when a compare event occurred at pin 1.3/INT6/CC3. Cleared by hardware when processor vectors to interrupt routine.
IEX5	External interrupt 5 edge flag Set by hardware when external interrupt edge was detected or when a compare event occurred at pin 1.2/INT5/CC2. Cleared by hardware when processor vectors to interrupt routine.
IEX4	External interrupt 4 edge flag Set by hardware when external interrupt edge was detected or when a compare event occurred at pin 1.1/INT4/CC1. Cleared by hardware when processor vectors to interrupt routine.
IEX3	External interrupt 3 edge flag Set by hardware when external interrupt edge was detected or when a compare event occurred at pin 1.0/$\overline{\text{INT3}}$/CC0. Cleared by hardware when processor vectors to interrupt routine.
IEX2	External interrupt 2 edge flag Set by hardware when external interrupt edge was detected or when a compare event occurred at pin 1.4/$\overline{\text{INT2}}$/CC4. Cleared by hardware when processor vectors to interrupt routine.
IADC	A/D converter interrupt request flag Set by hardware at the end of a conversion. Must be cleared by software.

The compare timer match interrupt occurs on a compare match of the CM0 to CM7 registers with the compare timer when compare mode 1 is selected for the corresponding channel. There are 8 compare match interrupt flags available in SFR IRCON1 which are or-ed together for a single interrupt request. Thus, a compare match interrupt service routine has to check which compare match has requested the compare match interrupt. The ICMPx flags must be cleared by software.

Only if timer 2 is assigned to the CMx registers (compare mode 0), an ICMPx request flag is set by every match in the compare channel. When the compare timer is assigned to the CMx registers (compare mode 1), an ICMPx request flag will not be set by a compare match event.

Special Function Register IRCON1 (Address $D1_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$D1_H$	ICMP7	ICMP6	ICMP5	ICMP4	ICMP3	ICMP2	ICMP1	ICMP0	IRCON1

Bit	Function
ICMP7 - 0	Compare timer match with register CM7 - CM0 interrupt flags ICMPx is set by hardware when a compare match of the compare timer with the compare register CMx occurs but only if the compare function for CMx has been enabled. ICMPx must be cleared by software (CMSEL.x = 0 and CMEN.x = 1).

The compare timer interrupt is generated by bit CTF in register CTCON, which is set by a rollover in the compare timer. If a compare timer interrupt is generated, flag CTF can be cleared by software.

The timer 2 compare match set and compare match clear interrupt is generated by bits ICS and ICR in register CTCON. These flags are set by a match in registers COMSET and COMCLR, when enabled. As long as the match condition is valid the request flags can't be reset (neither by hardware nor software).

Special Function Register CTCON (Address. $E1_H$) **Reset Value : $0X000000_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$E1_H$	T2PS1	–	ICR	ICS	CTF	CLK2	CLK1	CLK0	CTCON

The shaded bits are not used for interrupt purposes.

Bit	Function
ICR	Interrupt request flag for compare register COMCLR ICR is set when a compare match occured. ICR is cleared ba hardware when the processor vectors to interrupt routine.
ICS	Interrupt request flag for compare register COMSET ICS is set when a compare match occured. ICS is cleared by hardware when the processor vectors to interrupt routine.
CTF	Compare timer overflow flag CTF is set when the compare timer 1 count rolls over from all ones to the reload value. When CTF is set, a compare timer interrupt can be generated (if enabled). CTF is cleared by hardware when the compare timer value is no more equal to the reload value.

All of these interrupt request bits that generate interrupts can be set or cleared by software, with the same result as if they had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled by software. The only exceptions are the request flags IE0 and lE1. If the external interrupts 0 and 1 are programmed to be level-activated, IE0 and lE1 are controlled by the external source via pin $\overline{INT0}$ and $\overline{INT1}$, respectively. Thus, writing a one to these bits will not set the request flag IE0 and/or lE1. In this mode, interrupts 0 and 1 can only be generated by software and by writing a 0 to the corresponding pins $\overline{INT0}$ (P3.2) and $\overline{INT1}$ (P3.3), provided that this will not affect any peripheral circuit connected to the pins.

7.1.3 Interrupt Priority Registers

The 17 interrupt sources of the C517A are combined to six interrupt groups. Each of the six interrupt groups can be programmed to one of four priority levels by setting or clearing a bit in the IP0 and IP1 priority registers. Further deails abaout the interrupt priority structure are given in chapter 7.2.

Special Function Register IP0 (Address $A9_H$) **Reset Value : 00_H**
Special Function Register IP1 (Address $B9_H$) **Reset Value : $XX000000_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$A9_H$	OWDS	WDTS	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0	IP0
	7	6	5	4	3	2	1	0	
$B9_H$	–	–	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0	IP1

The shaded bits are not used for interrupt purposes.

Bit	Function
IP1.x IP0.x	Interrupt Priority level bits (x=0-5)

IP1.x	IP0.x	Function
0	0	Interrupt group x is set to priority level 0 (lowest)
0	1	Interrupt group x is set to priority level 1
1	0	Interrupt group x is set to priority level 2
1	1	Interrupt group x is set to priority level 3 (highest)

7.2 Interrupt Priority Level Structure

The 17 interrupt sources of the C517A are combined in six groups. **Table 7-1** lists the structure of these interrupt groups.

Table 7-1
Interrupt Source Structure

Interrupt Group	Associated Interrupts High Priority → Low Priority			Priority
1	External interrupt 0	Serial port 1 interrupt	A/D converter interrupt	High
2	Timer 0 overflow	–	External interrupt 2	↓
3	External interrupt 1	CM0-7 match interrupt	External interrupt 3	
4	Timer 1 overflow	Compare timer interrupt	External interrupt 4	
5	Serial port 0 interrupt	Match in COMSET	External interrupt 5	
6	Timer 2 interrupt	Match in COMCLR	External interrupt 6	Low

Each group of interrupt sources can be programmed individually to one of four priority levels by setting or clearing one bit in the special function register IP0 and one in IP1. A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another interrupt of the same or a lower priority. An interrupt of the highest priority level cannot be interrupted by another interrupt source.

If two or more requests of different priority leveis are received simultaneously, the request of the highest priority is serviced first. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is to be serviced first. Thus, within each priority level there is a second priority structure determined by the polling sequence, as follows.

- Within one interrupt group the “left“ interrupt is serviced first
- The interrupt groups are serviced from top to bottom of the table.

7.3 How Interrupts are Handled

The interrupt flags are sampled at S5P2 in each machine cycle. The sampled flags are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the preceeding cycle, the polling cycle will find it and the interrupt system will generate a 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 is already in progress.
2. The current (polling) cycle is not in the final cycle of the instruction in progress.
3. The instruction in progress is RETI or any write access to registers IEN0, IEN1, IEN2 or IP0/IP1.

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 is completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any write access to the IEN or IP registers, then at least one more instruction will be executed before any interrupt is vectored to; this delay guarantees that changes of the interrupt status can be observed by the CPU.

The polling cycle is repeated with each machine cycle and the values polled are the values that were present at S5P2 of the previous machine cycle. Note that if any interrupt flag is active but not being responded to for one of the conditions already mentioned, or if the flag is no longer 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 interrogates only the pending interrupt requests.

The polling cycle/LCALL sequence is illustrated in **figure 7-4**.

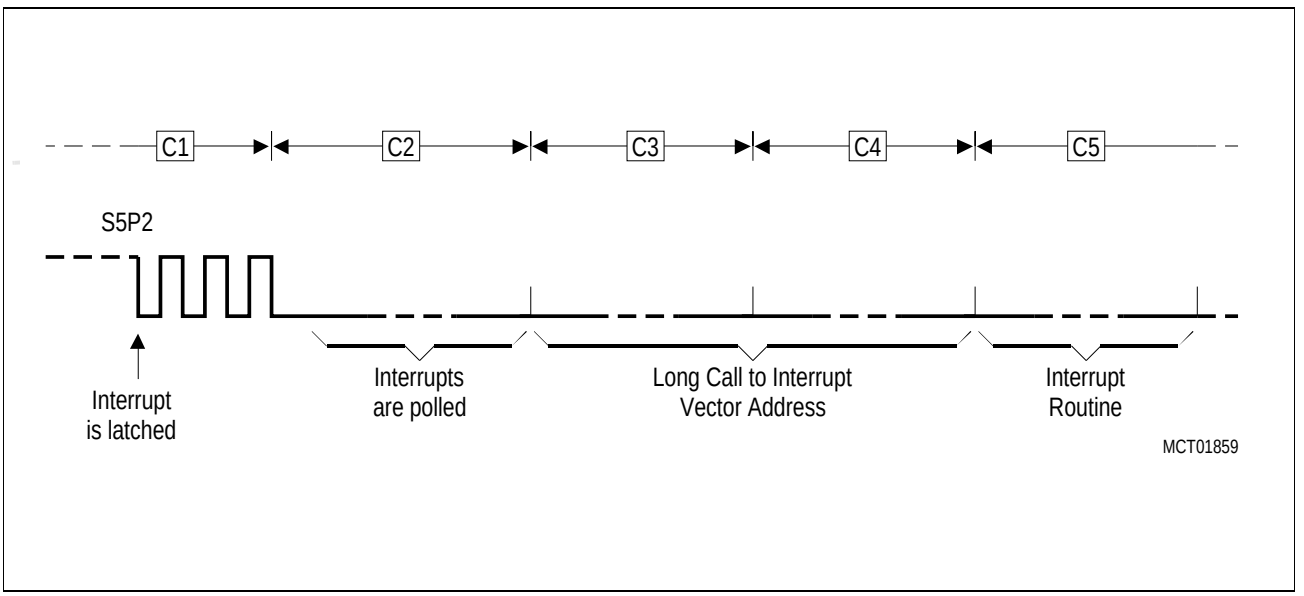


Figure 7-4
Interrupt Response Timing Diagram

Note that if an interrupt of a higher priority level goes active prior to S5P2 in the machine cycle labeled C3 in **figure 7-4** then, in accordance with the above rules, it will be vectored to during C5 and C6 without any instruction for the lower priority routine to be executed.

Thus, 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, while in other cases it does not. Then this has to be done by the user's software. The hardware clears the external interrupt flags IE0 and IE1 only if they were 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 program counter with an address that depends on the source of the interrupt being vectored to, as shown in **table 7-2**.

Table 7-2
Interrupt Source and Vectors

Interrupt Source	Interrupt Vector Address	Interrupt Request Flags
External Interrupt 0	0003_H	IE0
Timer 0 Overflow	$000B_H$	TF0
External Interrupt 1	0013_H	IE1
Timer 1 Overflow	$001B_H$	TF1
Serial Channel 0	0023_H	RI0 / TI0
Timer 2 Overflow / Ext. Reload	$002B_H$	TF2 / EXF2
A/D Converter	0043_H	IADC
External Interrupt 2	$004B_H$	IEX2
External Interrupt 3	0053_H	IEX3
External Interrupt 4	$005B_H$	IEX4
External Interrupt 5	0063_H	IEX5
External Interrupt 6	$006B_H$	IEX6
Serial Channel 1	0083_H	RI1 / TI1
Compare Match Interupt of Compare Registers CM0-CM7 assigned to Timer 2	0093_H	ICMP0 - ICMP7
Compare Timer Overflow	$009B_H$	CTF
Compare Match Interupt of Compare Register COMSET	$00A3_H$	ICS
Compare Match Interupt of Compare Register COMCLR	$00AB_H$	ICR

Execution proceeds from that location until the RETI instruction is encountered. The RETI instruction informs the processor that the interrupt routine is no longer in progress, then pops the two top bytes from the stack and reloads the program counter. Execution of the interrupted program continues from the point where it was stopped. Note that the RETI instruction is very important because it informs the processor that the program left the current interrupt priority level. A simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking an interrupt was still in progress. In this case no interrupt of the same or lower priority level would be acknowledged.

7.4 External Interrupts

The external interrupts 0 and 1 can be programmed to be level-activated or negative-transition activated by setting or clearing bit IT0 or IT1, respectively, in register TCON. If ITx = 0 (x = 0 or 1), external interrupt x is triggered by a detected low level at the $\overline{\text{INTx}}$ pin. If ITx = 1, external interrupt x is negative edge-triggered. In this mode, if successive samples of the $\overline{\text{INTx}}$ pin show a high in one cycle and a low in the next cycle, interrupt request flag IEx in TCON is set. Flag bit IEx then requests the interrupt.

If the external interrupt 0 or 1 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.

The external interrupts 2 and 3 can be programmed to be negative or positive transition-activated by setting or clearing bit I2FR or I3FR in register T2CON. If IxFR = 0 (x = 2 or 3), external interrupt x is negative transition-activated. If IxFR = 1, external interrupt is triggered by a positive transition.

The external interrupts 4, 5, and 6 are activated by a positive transition. The external timer 2 reload trigger interrupt request flag EXF2 will be activated by a negative transition at pin P1.5/T2EX but only if bit EXEN2 is set.

Since the external interrupt pins ($\overline{\text{INT2}}$ to INT6) are sampled once in each machine cycle, an input high or low should be held for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin low (high for $\overline{\text{INT2}}$ and $\overline{\text{INT3}}$, if it is programmed to be negative transition-active) for at least one cycle, and then hold it high (low) for at least one cycle to ensure that the transition is recognized so that the corresponding interrupt request flag will be set (see **figure 7-5**). The external interrupt request flags will automatically be cleared by the CPU when the service routine is called.

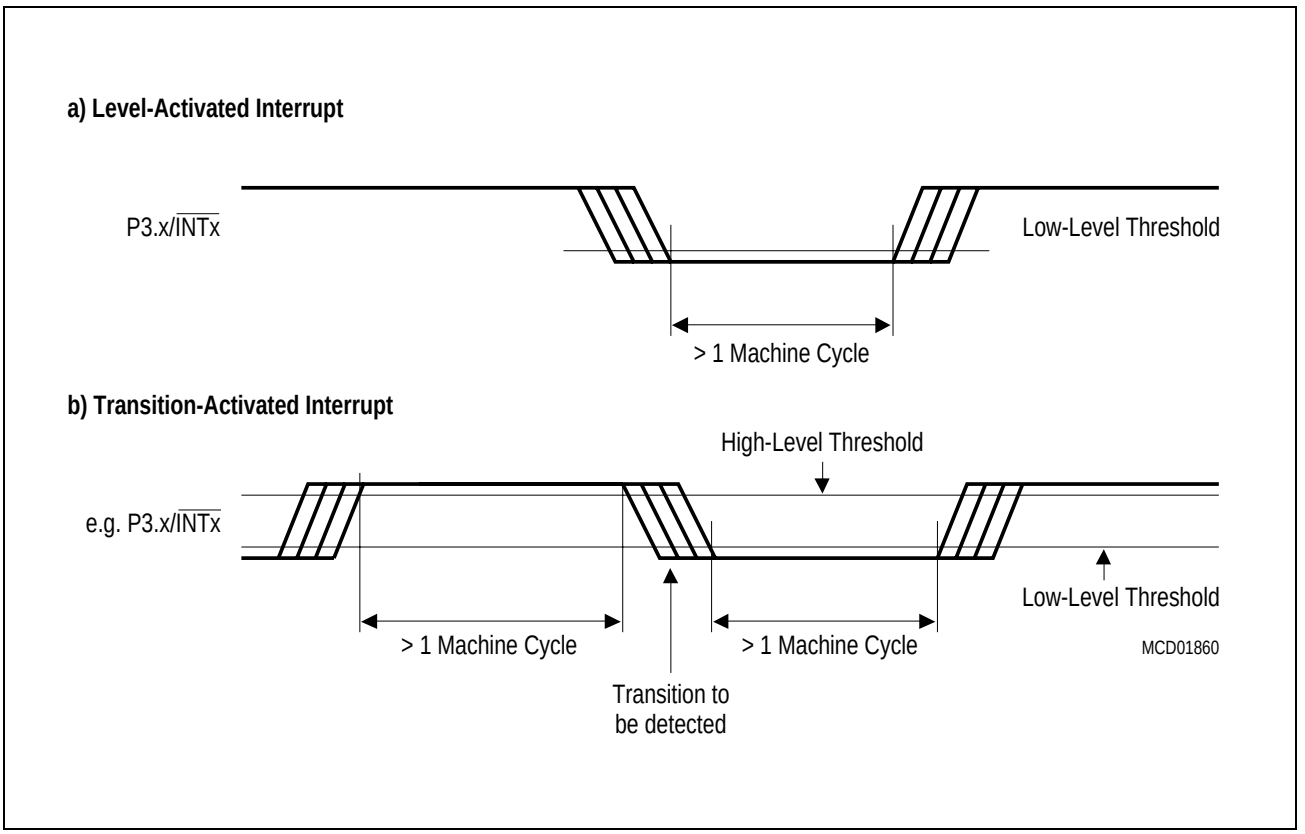


Figure 7-5
External Interrupt Detection

7.5 Interrupt Response Time

If an external interrupt is recognized, its corresponding request flag is set at S5P2 in every machine cycle. The value is not polled by the circuitry until the next machine cycle. If the request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be next instruction to be executed. The call itself takes two cycles. Thus a minimum of three complete machine cycles will elapse between activation and external interrupt request and the beginning of execution of the first instruction of the service routine.

A longer response time would be obtained if the request was blocked by one of the three previously listed conditions. If an interrupt of equal or higer priority is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles since the longest instructions (MUL and DIV) are only 4 cycles long; and, if the instruction in progress is RETI or a write access to registers IE or IP the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction, if the instruction is MUL or DIV).

Thus a single interrupt system, the response time is always more than 3 cycles and less than 9 cycles.

8 Fail Safe Mechanisms

The C517A offers enhanced fail safe mechanisms, which allow an automatic recovery from software upset or hardware failure :

- a programmable watchdog timer (WDT), with variable time-out period from 512 µs up to approx. 1.1 s at 12 MHz.
- an oscillator watchdog (OWD) which monitors the on-chip oscillator and forces the microcontroller into reset state in case the on-chip oscillator fails; it also provides the clock for a fast internal reset after power-on.

8.1 Programmable Watchdog Timer

To protect the system against software upset, the user's program has to clear this watchdog within a previously programmed time period. If the software fails to do this periodical refresh of the watchdog timer, an internal hardware reset will be initiated. The software can be designed so that the watchdog times out if the program does not work properly. It also times out if a software error is based on hardware-related problems.

The watchdog timer in the C517A is a 15-bit timer, which is incremented by a count rate of $f_{OSC}/24$ up to $f_{OSC}/384$. The system clock of the C517A is divided by two prescalers, a divide-by-two and a divide-by-16 prescaler. For programming of the watchdog timer overflow rate, the upper 7 bit of the watchdog timer can be written. **Figure 8-1** shows the block diagram of the watchdog timer unit.

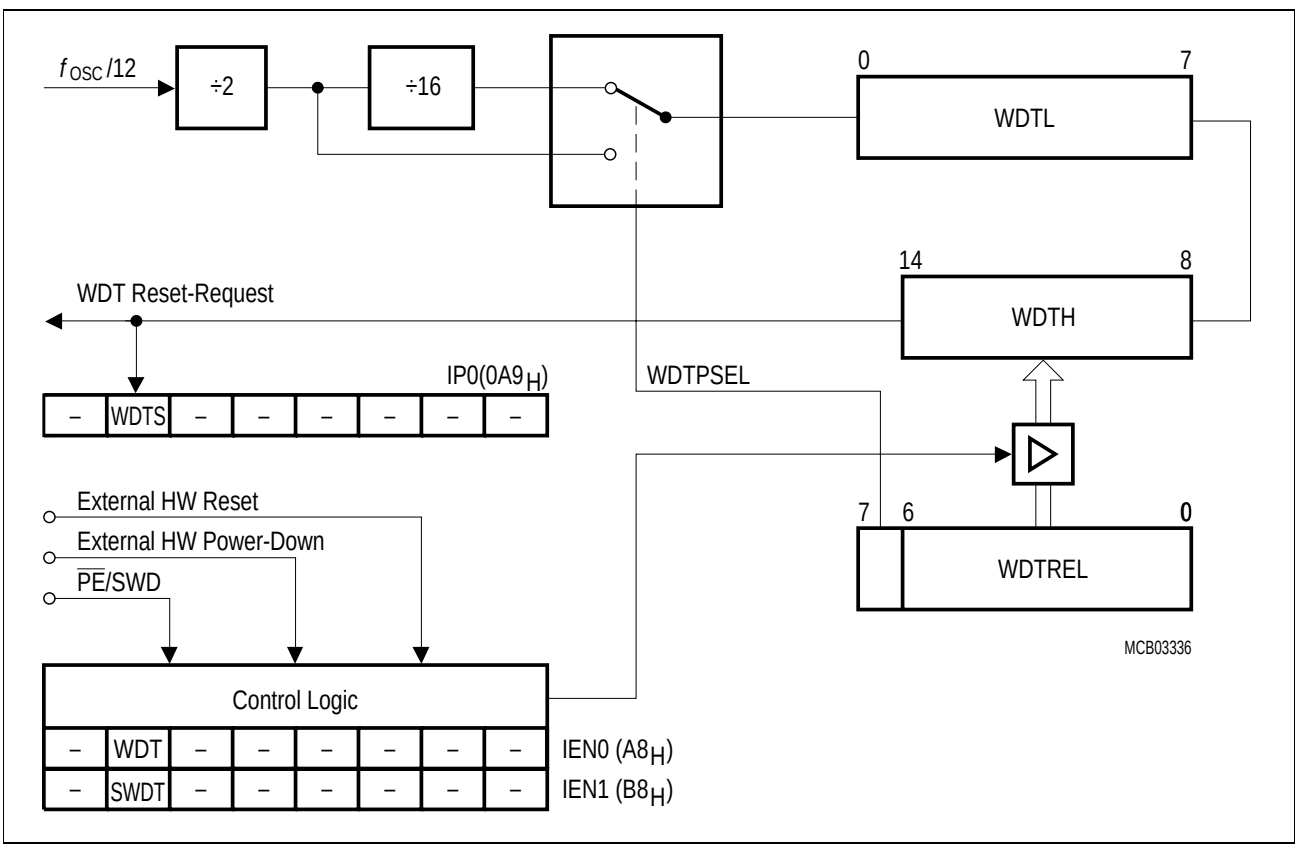
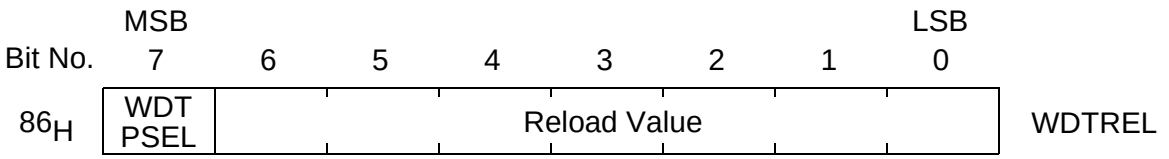


Figure 8-1
Block Diagram of the Programmable Watchdog Timer

8.1.1 Input Clock Selection

The input clock rate of the watchdog timer is derived from the system clock of the C517A. There is a prescaler available, which is software selectable and defines the input clock rate. This prescaler is controlled by bit WDTPSEL in the SFR WDTREL. **Tabel 8-1** shows resulting timeout periods at f_{OSC} = 12 and 24 MHz.

Special Function Register WDTREL (Address 86_H) **Reset Value : 00_H**



Bit	Function
WDTPSEL	Watchdog timer prescaler select bit. When set, the watchdog timer is clocked through an additional divide-by-16 prescaler.
WDTREL.6 - 0	Seven bit reload value for the high-byte of the watchdog timer. This value is loaded to WDTH when a refresh is triggered by a consecutive setting of bits WDT and SWDT.

Table 8-1
Watchdog Timer Time-Out Periods (WDTPSEL = 0)

WDTREL	Time-Out Period		Comments
	f_{osc} = 12 MHz	f_{osc} = 24 MHz	
00_H	65.535 ms	32.768 ms	This is the default value
80_H	1.1 s	0.55 s	Maximum time period
$7F_H$	512 µs	256 µs	Minimum time period

8.1.2 Watchdog Timer Control / Status Flags

The watchdog timer is controlled by two control flags (located in SFR IEN0 and IEN1) and one status flags (located in SFR IP0).

Special Function Register IEN0 (Address $A8_H$) **Reset Value : 00_H**
Special Function Register IEN1 (Address $B8_H$) **Reset Value : 00_H**
Special Function Register IP0 (Address $A9_H$) **Reset Value : 00_H**

	MSB							LSB	
	AF_H	AE_H	AD_H	AC_H	AB_H	AA_H	$A9_H$	$A8_H$	
$A8_H$	EAL	WDT	ET2	ES	ET1	EX1	ET0	EX0	IEN0
	BF_H	BE_H	BD_H	BC_H	BB_H	BA_H	$B9_H$	$B8_H$	
$B8_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	IEN1
Bit No.	7	6	5	4	3	2	1	0	
$A9_H$	OWDS	WDTS	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0	IP0

The shaded bits are not used for fail save control.

Bit	Function
WDT	Watchdog timer refresh flag. Set to initiate a refresh of the watchdog timer. Must be set directly before SWDT is set to prevent an unintentional refresh of the watchdog timer.
SWDT	Watchdog timer start flag. Set to activate the Watchdog Timer. When directly set after setting WDT, a watchdog timer refresh is performed.
WDTS	Watchdog timer status flag. Set by hardware when a watchdog Timer reset occured. Can be cleared and set by software.

8.1.3 Starting the Watchdog Timer

Immediately after start (see next section for the start procedure), the watchdog timer is initialized to the reload value programmed to WDTREL.0 - WDTREL.6. After an external HW or $\overline{\text{HWPD}}$ reset, an oscillator power on reset, or a watchdog timer reset, register WDTREL is cleared to 00_H. WDTREL can be loaded by software at any time.

There are two ways to start the watchdog timer depending on the level applied to pin $\overline{\text{PE}}$/SWD. This pin serves two functions, because it is also used for blocking the power saving modes. (see also chapter 9).

8.1.3.1 The First Possibility of Starting the Watchdog Timer

The automatic start of the watchdog timer directly while an external HW reset is a hardware start initialized by strapping pin $\overline{\text{PE}}$/SWD to V_{DD}. In this case the power saving modes (power down mode, idle mode and slow down mode) are also disabled and cannot be started by software. If pin $\overline{\text{PE}}$/SWD is left unconnected, a weak pull-up transistor ensures the automatic start of the watchdog timer.

The self-start of the watchdog timer by a pin option has been implemented to provide high system security in electrically very noisy environments.

Note : The automatic start of the watchdog timer is only performed if $\overline{\text{PE}}$/SWD (power-save enable/ start watchdog timer) is held at high level while $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ is active. A positive transition at these pins during normal program execution will not start the watchdog timer. Furthermore, when using the hardware start, the watchdog timer starts running with its default time-out period. The value in the reload register WDTREL, however, can be overwritten at any time to set any time-out period desired.

8.1.3.2 The Second Possibility of Starting the Watchdog Timer

The watchdog timer can also be started by software. Setting of bit SWDT in SFR IEN1 starts the watchdog timer. Using the software start, the timeout period can be programmed before the watchdog timer starts running.

Note that once the watchdog timer has been started it can only be stopped if one of the following conditions are met :

- active external hardware reset through pin $\overline{\text{RESET}}$ with a low level at pin $\overline{\text{PE}}$/SWD
- active hardware power down signal $\overline{\text{HWPD}}$, independently of the level at $\overline{\text{PE}}$/SWD
- entering idle mode or power down mode by software

See chapter 9 for entering the power saving modes by software.

8.1.4 Refreshing the Watchdog Timer

At the same time the watchdog timer is started, the 7-bit register WDTH is preset by the contents of WDTREL.0 to WDTREL.6. Once started the watchdog cannot be stopped by software but can only be refreshed to the reload value by first setting bit WDT (IEN0.6) and by the next instruction setting SWDT (IEN1.6). Bit WDT will automatically be cleared during the second machine cycle after having been set. For this reason, setting SWDT bit has to be a one cycle instruction (e.g. SETB SWDT). This double-instruction refresh of the watchdog timer is implemented to minimize the chance of an unintentional reset of the watchdog.

The reload register WDTREL can be written to at any time, as already mentioned. Therefore, a periodical refresh of WDTREL can be added to the above mentioned starting procedure of the watchdog timer. Thus a wrong reload value caused by a possible distortion during the write operation to the WDTREL can be corrected by software.

8.1.5 Watchdog Reset and Watchdog Status Flag

If the software fails to clear the watchdog in time, an internally generated watchdog reset is entered at the counter state $7FFC_H$. The duration of the reset signal then depends on the prescaler selection (either 8 cycles or 128 cycles). This internal reset differs from an external one only in so far as the watchdog timer is not disabled and bit WDTS (watchdog timer status, bit 6 in SFR IP0) is set. **Figure 8-2** shows a block diagram of all reset requests in the C517A and the function of the watchdog status flags. The WDTS flag is a flip-flop, which is set by a watchdog timer reset and cleared by an external HW reset. Bit WDTS allows the software to examine from which source the reset was activated. The watchdog timer status flag can also be cleared by software.

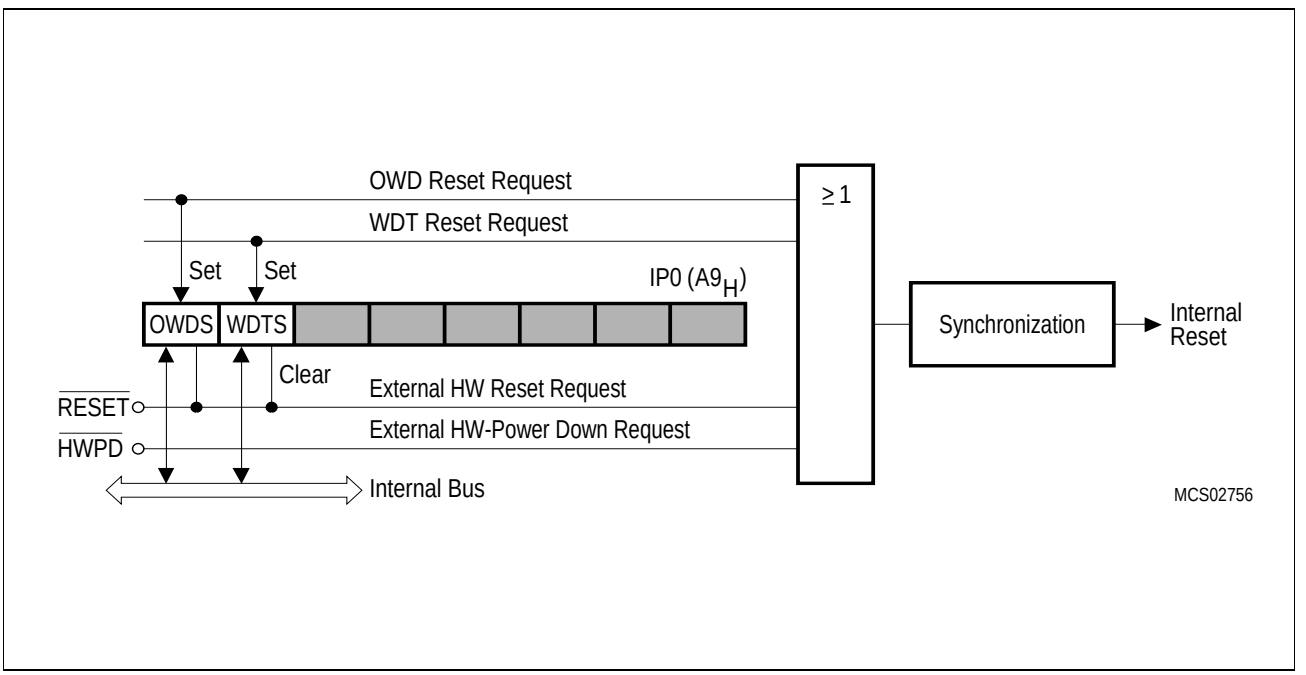


Figure 8-2
Watchdog Timer Status Flags and Reset Requests

8.2 Oscillator Watchdog Unit

The oscillator watchdog unit serves for four functions:

- **Monitoring of the on-chip oscillator's function**
  The watchdog supervises the on-chip oscillator's frequency; if it is lower than the frequency of the auxiliary RC oscillator in the watchdog unit, the internal clock is supplied by the RC oscillator and the device is brought into reset; if the failure condition disappears (i.e. the on-chip oscillator has a higher frequency than the RC oscillator), the part executes a final reset phase of typ. 1 ms in order to allow the oscillator to stabilize; then the oscillator watchdog reset is released and the part starts program execution again.
- **Fast internal reset after power-on**
  The oscillator watchdog unit provides a clock supply for the reset before the on-chip oscillator has started. The oscillator watchdog unit also works identically to the monitoring function.
- **Restart from the hardware power down mode.**
  If the hardware power down mode is terminated the oscillator watchdog has to control the correct start-up of the on-chip oscillator and to restart the program. The oscillator watchdog function is only part of the complete hardware power down sequence; however, the watchdog works identically to the monitoring function.

Note: The oscillator watchdog unit is always enabled.

8.2.1 Description of the Oscillator Watchdog Unit

Figure 8-3 shows the block diagram of the oscillator watchdog unit. It consists of an internal RC oscillator which provides the reference frequency for the comparison with the frequency of the on-chip oscillator. It also shows the modifications which have been made for integration of the wake-up from power down mode capability.

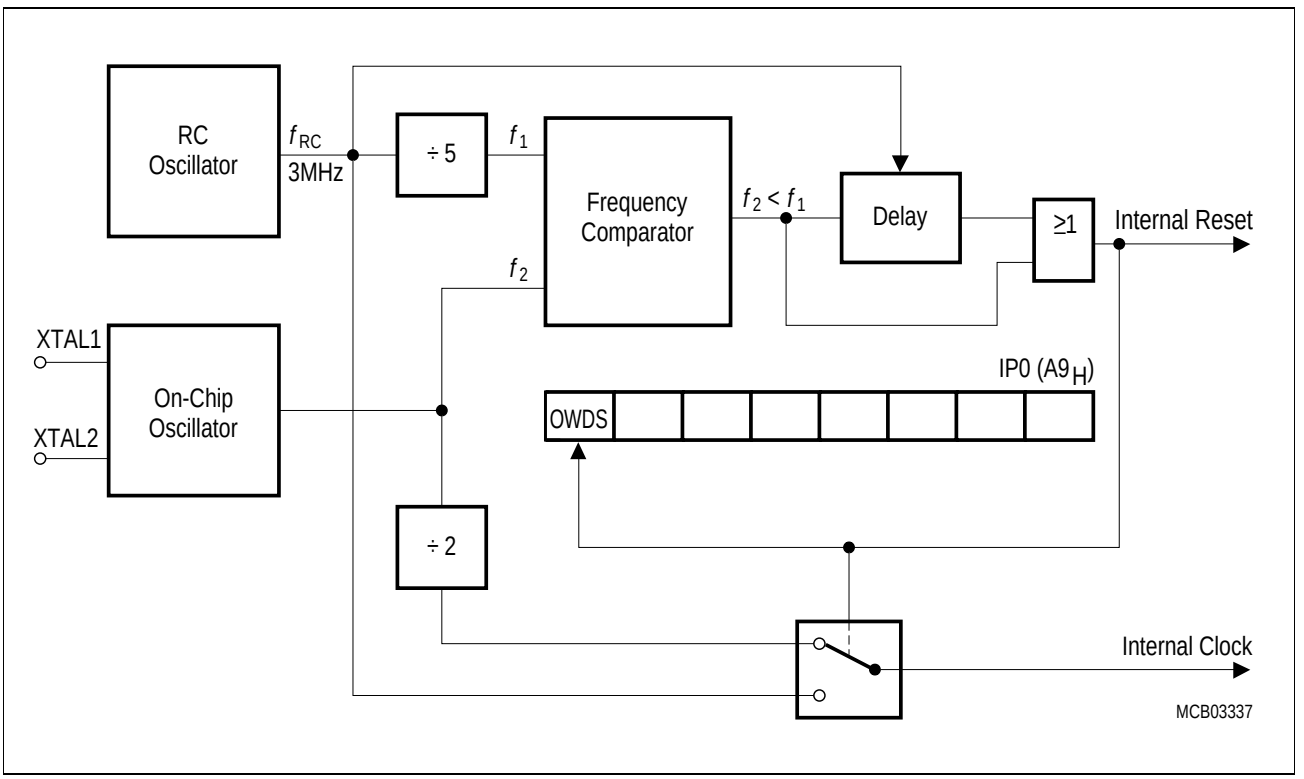


Figure 8-3
Functional Block Diagram of the Oscillator Watchdog

The frequency coming from the RC oscillator is divided by 5 and compared to the on-chip oscillator's frequency. If the frequency coming from the on-chip oscillator is found lower than the frequency derived from the RC oscillator the watchdog detects a failure condition (the oscillation at the on-chip oscillator could stop because of crystal damage etc.). In this case it switches the input of the internal clock system to the output of the RC oscillator. This means that the part is being clocked even if the on-chip oscillator has stopped or has not yet started. At the same time the watchdog activates the internal reset in order to bring the part in its defined reset state. The reset is performed because clock is available from the RC oscillator. This internal watchdog reset has the same effects as an externally applied reset signal with the following exceptions: The Watchdog Timer Status flag WDTS is not reset (the Watchdog 99Timer however is stopped); and bit OWDS is set. This allows the software to examine error conditions detected by the Watchdog Timer even if meanwhile an oscillator failure occured.

The oscillator watchdog is able to detect a recovery of the on-chip oscillator after a failure. If the frequency derived from the on-chip oscillator is again higher than the reference the watchdog starts a final reset sequence which takes typ. 1 ms. Within that time the clock is still supplied by the RC oscillator and the part is held in reset. This allows a reliable stabilization of the on chip oscillator. After that, the watchdog toggles the clock supply back to the on-chip oscillator and releases thereset

request. If no reset is applied in this moment the part will start program execution. If an external reset is active, however, the device will keep the reset state until also the external reset request disappears.

Furthermore, the status flag OWDS is set if the oscillator watchdog was active. The status flag can be evaluated by software to detect that a reset was caused by the oscillator watchdog. The flag OWDS can be set or cleared by software. An external reset request, however, also resets OWDS (and WDTS).

Special Function Register IP0 (Address $A9_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$A9_H$	OWDS	WDTS	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0	IP0

The shaded bits are not used for fail save control.

Bit	Function
OWDS	Oscillator Watchdog Timer Status Flag. Set by hardware when an oscillator watchdog reset occured. Can be set and cleared by software.

8.2.2 Fast Internal Reset after Power-On

The C517A can use the oscillator watchdog unit for a fast internal reset procedure after power-on.

Normally the members of the 8051 family (e. g. SAB 80C52) enter their default reset state not before the on-chip oscillator starts. The reason is that the external reset signal must be internally synchronized and processed in order to bring the device into the correct reset state. Especially if a crystal is used the start up time of the oscillator is relatively long (typ. 1 ms). During this time period the pins have an undefined state which could have severe effects e.g. to actuators connected to port pins.

In the C517A the oscillator watchdog unit avoids this situation. After power-on the oscillator watchdog's RC oscillator starts working within a very short start-up time (typ. less than 2 microseconds). In the following the watchdog circuitry detects a failure condition for the on-chip oscillator because this has not yet started (a failure is always recognized if the watchdog's RC oscillator runs faster than the on-chip oscillator). As long as this condition is valid the watchdog uses the RC oscillator output as clock source for the chip. This allows correct resetting of the part and brings all ports to the defined state (see also chapter 5 of this manual). The delay time between power-on and correct reset state is max 34 µs (more details see chapter 5.2).

9 Power Saving Modes

The C517A provides three modes in which power consumption can be significantly reduced.

- **Idle mode**
  The CPU is gated off from the oscillator. All peripherals are still provided with the clock and are able to work.
- **Power down mode**
  The operation of the C517A is completely stopped and the oscillator is turned off. This mode is used to save the contents of the internal RAM with a very low standby current. Power down mode can be entered by software or by hardware.
- **Slow-down mode**
  The controller keeps up the full operating functionality, but its normal clock frequency is internally divided by eight. This slows down all parts of the controller, the CPU and all peripherals, to 1/8 th of their normal operating frequency. Slowing down the frequency greatly reduces power consumption.

All of these modes - a detailed description of each is given in the following sections - are entered by software. Special function register PCON (power control register) is used to select one of these modes.

9.1 Hardware Enable for the Use of the Power Saving Modes

To provide power saving modes together with effective protection against unintentional entering of these modes, the C517A has an extra pin disabling the use of the power saving modes. As this pin will most likely be used only in critical applications it is combined with an automatic start of the watchdog timer (see the description in **chapter 8** “Fail Save Mechanisms”). This pin is called $\overline{\text{PE}}$/SWD (power saving enable/start watchdog timer) and its function is as follows:

$\overline{\text{PE}}$/SWD = 1 (logic high level)

- Use of the power saving modes is not possible. The instruction sequences used for entering these modes will not affect the normal operation of the device.
- If and only if $\overline{\text{PE}}$/SWD is held at high level during reset, the watchdog timer is started immediately after reset is released.

$\overline{\text{PE}}$/SWD = 0 (logic low level)

- All power saving modes can be activated as described in the following sections
- The watchdog timer has to be started by software if system protection is desired.

When left unconnected, the pin $\overline{\text{PE}}$/SWD is pulled to high level by a weak internal pullup. This is done to provide system protection by default.

The logic level applied to pin $\overline{\text{PE}}$/SWD can be changed during program execution in order to allow or block the use of the power saving modes without any effect on the on-chip watchdog circuitry; (the watchdog timer is started only if $\overline{\text{PE}}$/SWD is on high level at the moment when reset is released; a change at $\overline{\text{PE}}$/SWD during program execution has no effect on the watchdog timer; this only enables or disables the use of the power saving modes. A change of the pin’s level is detected in state 3, phase 1. A Schmitt trigger is used at the input to reduce susceptibility to noise.

In addition to the hardware enable/disable of the power saving modes, a double-instruction sequence which is described in the corresponding sections is necessary to enter power down and idle mode. The combination of all these safety precautions provide a maximum of system protection.

9.2 Application Example for Switching Pin $\overline{\text{PE}}$/SWD

For most applications in noisy environments, components external to the chip are used to give warning of a power failure or a turn off of the power supply. These circuits could be used to control the $\overline{\text{PE}}$/SWD pin. The possible steps to go into power down mode could then be as follows:

- A power-fail signal forces the controller to go into a high priority interrupt routine. This interrupt routine saves the actual program status. At the same time pin $\overline{\text{PE}}$/SWD is pulled low by the power-fail signal.
- Finally the controller enters power down mode by executing the relevant double-instruction sequence.

9.3 Power Saving Mode Control Registers

The functions of the power saving modes are controlled by bits which are located in the special function registers PCON which is located at SFR address 87_H.

The bits PDE, PDS and IDLE, IDLS located in SFR PCON select the power down mode or the idle mode, respectively. If the power down mode and the idle mode are set at the same time, power down takes precedence.

Special Function Register PCON (Address 87_H) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
87_H	SMOD	PDS	IDLS	SD	GF1	GF0	PDE	IDLE	PCON

The function of the shaded bit is not described in this section.

Symbol	Function
PDS	Power down start bit The instruction that sets the PDS flag bit is the last instruction before entering the power down mode
IDLS	Idle start bit The instruction that sets the IDLS flag bit is the last instruction before entering the idle mode.
SD	Slow down mode bit When set, the slow down mode is enabled
GF1	General purpose flag
GF0	General purpose flag
PDE	Power down enable bit When set, starting of the power down is enabled
IDLE	Idle mode enable bit When set, starting of the idle mode is enabled

Note : The PDS bit, which controls the software power down mode is forced to logic low whenever the external $\overline{PE}$/SWD pin is held at logic high level.

9.4 Idle Mode

In the idle mode the oscillator of the C517A continues to run, but the CPU is gated off from the clock signal. However, the interrupt system, the serial port, the A/D converter, and all timers with the exception of the watchdog timer are further provided with the clock. The CPU status is preserved in its entirety: the stack pointer, program counter, program status word, accumulator, and all other registers maintain their data during idle mode.

The reduction of power consumption, which can be achieved by this feature depends on the number of peripherals running. If all timers are stopped and the A/D converter, and the serial interfaces are not running, the maximum power reduction can be achieved. This state is also the test condition for the idle mode I_{DD}.

Thus, the user has to take care which peripheral should continue to run and which has to be stopped during idle mode. Also the state of all port pins – either the pins controlled by their latches or controlled by their secondary functions – depends on the status of the controller when entering idle mode.

Normally, the port pins hold the logical state they had at the time when the idle mode was activated. If some pins are programmed to serve as alternate functions they still continue to output during idle mode if the assigned function is on. This especially applies to the system clock output signal at pin P1.6/CLKOUT and to the serial interfaces in case it cannot finish reception or transmission during normal operation. The control signals ALE and $\overline{\text{PSEN}}$ are hold at logic high levels.

As in normal operation mode, the ports can be used as inputs during idle mode. Thus a capture or reload operation can be triggered, the timers can be used to count external events, and external interrupts will be detected.

The idle mode is a useful feature which makes it possible to "freeze" the processor's status - either for a predefined time, or until an external event reverts the controller to normal operation, as discussed below. The watchdog timer is the only peripheral which is automatically stopped during idle mode.

If the idle mode is to be used the pin $\overline{PE}$/SWD must be held low. The idle mode is entered by two consecutive instructions. The first instruction sets the flag bit IDLE (PCON.0) and must not set bit IDLS (PCON.5), the following instruction sets the start bit IDLS (PCON.5) and must not set bit IDLE (PCON.0). The hardware ensures that a concurrent setting of both bits, IDLE and IDLS, does not initiate the idle mode. Bits IDLE and IDLS will automatically be cleared after being set. If one of these register bits is read the value that appears is 0. This double instruction is implemented to minimize the chance of an unintentional entering of the idle mode which would leave the watchdog timer's task of system protection without effect.

Note:

PCON is not a bit-addressable register, so the above mentioned sequence for entering the idle mode is obtained by byte-handling instructions, as shown in the following example:

```
ORL     PCON,#00000001B     ;Set bit IDLE, bit IDLS must not be set
ORL     PCON,#00100000B     ;Set bit IDLS, bit IDLE must not be set
```

The instruction that sets bit IDLS is the last instruction executed before going into idle mode.

There are two ways to terminate the idle mode:

- The idle mode can be terminated by activating any enabled interrupt. This interrupt will be serviced and normally the instruction to be executed following the RETI instruction will be the one following the instruction that sets the bit IDLS.
- The other way to terminate the idle mode, is a hardware reset. Since the oscillator is still running, the hardware reset must be held active only for two machine cycles for a complete reset.

9.5 Slow Down Mode Operation

In some applications, where power consumption and dissipation is critical, the controller might run for a certain time at reduced speed (e.g. if the controller is waiting for an input signal). Since in CMOS devices there is an almost linear dependence of the operating frequency and the power supply current, a reduction of the operating frequency results in reduced power consumption.

In the slow down mode all signal frequencies that are derived from the oscillator clock are divided by 8. This also includes the clock output signal at pin P1.6/CLKOUT. Further, if the slow down mode is used pin $\overline{\text{PE}}$/SWD must be held low.

The slow down mode is activated by setting the bit SD in SFR PCON. If the slow down mode is enabled, the clock signals for the CPU and the peripheral units are reduced to 1/8 of the nominal system clock rate. The controller actually enters the slow down mode after a short synchronization period (max. two machine cycles). The slow down mode is disabled by clearing bit SD.

The slow down mode can be combined with the idle mode by performing the following double instruction sequence:

```
ORL   PCON,#00000001B     ; preparing idle mode: set bit IDLE (IDLS not set)
ORL   PCON,#00110000B     ; entering idle mode combined with the slow down mode:
                          ; (IDLS and SD set)
```

There are two ways to terminate the combined Idle and Slow Down Mode :

- The idle mode can be terminated by activation of any enabled interrupt. The CPU operation is resumed, the interrupt will be serviced and the next instruction to be executed after the RETI instruction will be the one following the instruction that sets the bits IDLS and SD. Nevertheless the slow down mode keeps enabled and if required has to be disabled by clearing the bit SD in the corresponding interrupt service routine or after the instruction that sets the bits IDLS and SD.
- The other possibility of terminating the combined idle and slow down mode is a hardware reset. Since the oscillator is still running, the hardware reset has to be held active for only two machine cycles for a complete reset.

9.6 Software Power Down Mode

In the software power down mode, the RC osciillator and the on-chip oscillator which operates with the XTAL pins is stopped. Therefore, all functions of the microcontroller are stopped and only the contents of the on-chip RAM, XRAM and the SFR's are maintained. The port pins, which are controlled by their port latches, output the values that are held by their SFR's. The port pins which serve the alternate output functions show the values they had at the end of the last cycle of the instruction which initiated the software power down mode. ALE and $\overline{\text{PSEN}}$ hold at logic low level (see **table 9-1**).

In the software power down mode of operation, V_{DD} can be reduced to minimize power consumption. It must be ensured, however, that is V_{DD} not reduced before the software power down mode is invoked, and that V_{DD} is restored to its normal operating level before the software power down mode is terminated.

The software power down mode can be terminated in three ways :

- An active reset signal. Using reset to leave software power down mode puts the microcontroller with its SFRs into the reset state.
- A rising edge at $\overline{\text{PE}}$/SWD. If this pin is rising during the software power down mode, the microcontroller will go into the reset state.

Leaving software power down mode should not be done before V_{DD} is restored to its nominal operating level.

9.6.1 Invoking Software Power Down Mode

If the software power down mode is to be used, the pin $\overline{\text{PE}}$/SWD must be held low. The software power down mode is entered by two consecutive instructions. The first instruction has to set the flag bit PDE (PCON.1) and must not set bit PDS (PCON.6), the following instruction has to set the start bit PDS (PCON.6) and must not set bit PDE (PCON.1). The hardware ensures that a concurrent setting of both bits, PDE and PDS, does not initiate the software power down mode. Bits PDE and PDS will automatically be cleared after having been set and the value shown by reading one of these bits is always 0. This double instruction is implemented to minimize the chance of unintentionally entering the software power down mode which could possibly "freeze" the chip's activity in an undesired status.

PCON is not a bit-addressable register, so the above mentioned sequence for entering the software power down mode is obtained by byte-handling instructions, as shown in the following example:

```
ORL     PCON,#00000010B     ;set bit PDE, bit PDS must not be set
ORL     PCON,#01000000B     ;set bit PDS, bit PDE must not be set, enter power down
```

The instruction that sets bit PDS is the last instruction executed before going into software power down mode.

9.6.2 Exit from Software Power Down Mode

If software power down mode is exit via a hardware reset, the microcontroller with its SFRs is put into the hardware reset state and the content of RAM and XRAM are not changed. The reset signal that terminates the software power down mode also restarts the RC oscillator and the on-chip oscillatror. The reset operation should not be activated before V_{DD} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize (similar to power-on reset).

9.7 State of Pins in Software Initiated Power Saving Modes

In the idle mode and in the software power down mode the port pins of the C517A have a well defined status which is listed in the following **table 9-1**. This state of some pins also depends on the location of the code memory (internal or external).

Table 9-1
Status of External Pins During Idle and Software Power Down Mode

Outputs	Last Instruction Executed from Internal Code Memory		Last Instruction Executed from External Code Memory	
	Idle	**Power Down**	**Idle**	**Power Down**
ALE	High	Low	High	Low
$\overline{\text{PSEN}}$	High	Low	High	Low
PORT 0	Data	Data	Float	Float
PORT 2	Data	Data	Address	Data
PORT 1, 3, 4, 5, 6	Data / alternate outputs	Data / last output	Data / alternate outputs	Data / last output
P7.0	Data	Data	Data	Data

9.8 Hardware Power Down Mode

The power down mode of the C517A can also be initiated by an external signal at the pin $\overline{\text{HWPD}}$. Because this power down mode is activated by an external hardware signal it mode is referred to as hardware power down mode in opposite to the program controlled software power down mode.

Pin $\overline{\text{PE}}$/SWD has no control function for the hardware power down mode; it enables and disables only the use of all software controlled power saving modes (idle mode, software power down mode).

The function of the hardware power down mode is as follows:

– The pin $\overline{\text{HWPD}}$ controls this mode. If it is on logic high level (inactive) the part is running in the normal operating modes. If pin $\overline{\text{HWPD}}$ gets active (low level) the part enters the hardware power down mode; as mentioned above this is independent of the state of pin $\overline{\text{PE}}$/SWD.

$\overline{\text{HWPD}}$ is sampled once per machine cycle. If it is found active, the device starts a complete internal reset sequence. This takes two machine cycles; all pins have their default reset states during this time. This reset has exactly the same effects as a hardware reset; i.e.especially the watchdog timer is stopped and its status flag WDTS is cleared. In this phase the power consumption is not yet reduced. After completion of the internal reset both oscillators of the chip are disabled, the on-chip oscillator as well as the oscillator watchdog's RC oscillator. At the same time the port pins and several control lines enter a floating state as shown in **table 9-2**. In this state the power consumption is reduced to the power down current I_{PD}. Also the supply voltage can be reduced.

Table 9-2 also lists the voltages which may be applied at the pins during hardware power down mode without affecting the low power consumption.

Table 9-2
Status of all Pins During Hardware Power Down Mode

Pins	Status	Voltage Range at Pin During HW-Power Down
P0, P1, P2, P3, P4, P5, P6	Floating outputs/ Disabled input function	$V_{SS} \le V_{IN} \le V_{DD}$
$\overline{\text{EA}}$	Active input	$V_{IN} = V_{DD}$ or $V_{IN} = V_{SS}$
$\overline{\text{PE}}$/SWD	Active input, Pull-up resistor Disabled during HW power down	$V_{IN} = V_{DD}$ or $V_{IN} = V_{SS}$
XTAL 1	Active output	pin may not be driven
XTAL 2	Disabled input function	$V_{SS} \le V_{IN} \le V_{DD}$
$\overline{\text{PSEN}}$, ALE	Floating outputs/ Disabled input function (for test modes only)	$V_{SS} \le V_{IN} \le V_{DD}$
$\overline{\text{RESET}}$	Active input; must be at high level if $\overline{\text{HWPD}}$ is used	$V_{IN} = V_{DD}$
V_{AREF}	ADC reference supply input	$V_{SS} \le V_{IN} \le V_{DD}$

The hardware power down mode is maintained while pin $\overline{\text{HWPD}}$ is held active. If $\overline{\text{HWPD}}$ goes to high level (inactive state) an automatic start up procedure is performed:

- First the pins leave their floating condition and enter their default reset state as they had immediately before going to float state.
- Both oscillators are enabled. While the on-chip oscillator (with pins XTAL1 and XTAL2) usually needs a longer time for start-up, if not externally driven (with crystal approx. 1 ms), the oscillator watchdog's RC oscillator has a very short start-up time (typ. less than 2 µs).
- Because the oscillator watchdog is active it detects a failure condition if the on-chip oscillator hasn't yet started. Hence, the watchdog keeps the part in reset and supplies the internal clock from the RC oscillator.
- Finally, when the on-chip oscillator has started, the oscillator watchdog releases the part from reset after it performed a final internal reset sequence and switches the clock supply to the on-chip oscillator. This is exactly the same procedure as when the oscillator watchdog detects first a failure and then a recovering of the oscillator during normal operation. Therefore, also the oscillator watchdog status flag is set after restart from hardware power down mode.
  When automatic start of the watchdog was enabled ($\overline{\text{PE}}$/SWD connected to V_{DD}), the watchdog timer will start, too (with its default reload value for time-out period).

The SWD-Function of the $\overline{\text{PE}}$/SWD Pin is sampled only by a hardware reset.Therefore at least one power-on reset has to be performed.

9.9 Hardware Power Down Reset Timing

The following figures show the timing diagrams for entering (**figure 9-1**) and leaving (**figure 9-2**) the hardware power down mode. If there is only a short signal at pin $\overline{\text{HWPD}}$ (i.e. $\overline{\text{HWPD}}$ is sampled active only once), then a complete internal reset is executed. Afterwards, the normal program execution starts again (**figure 9-3**).

Note: Delay time caused by internal logic is not included.

The $\overline{\text{RESET}}$ pin overrides the hardware power down function, i.e. if reset gets active during hardware power down it is terminated and the device performs the normal reset function. Thus, pin $\overline{\text{RESET}}$ has to be inactive during hardware power down mode.

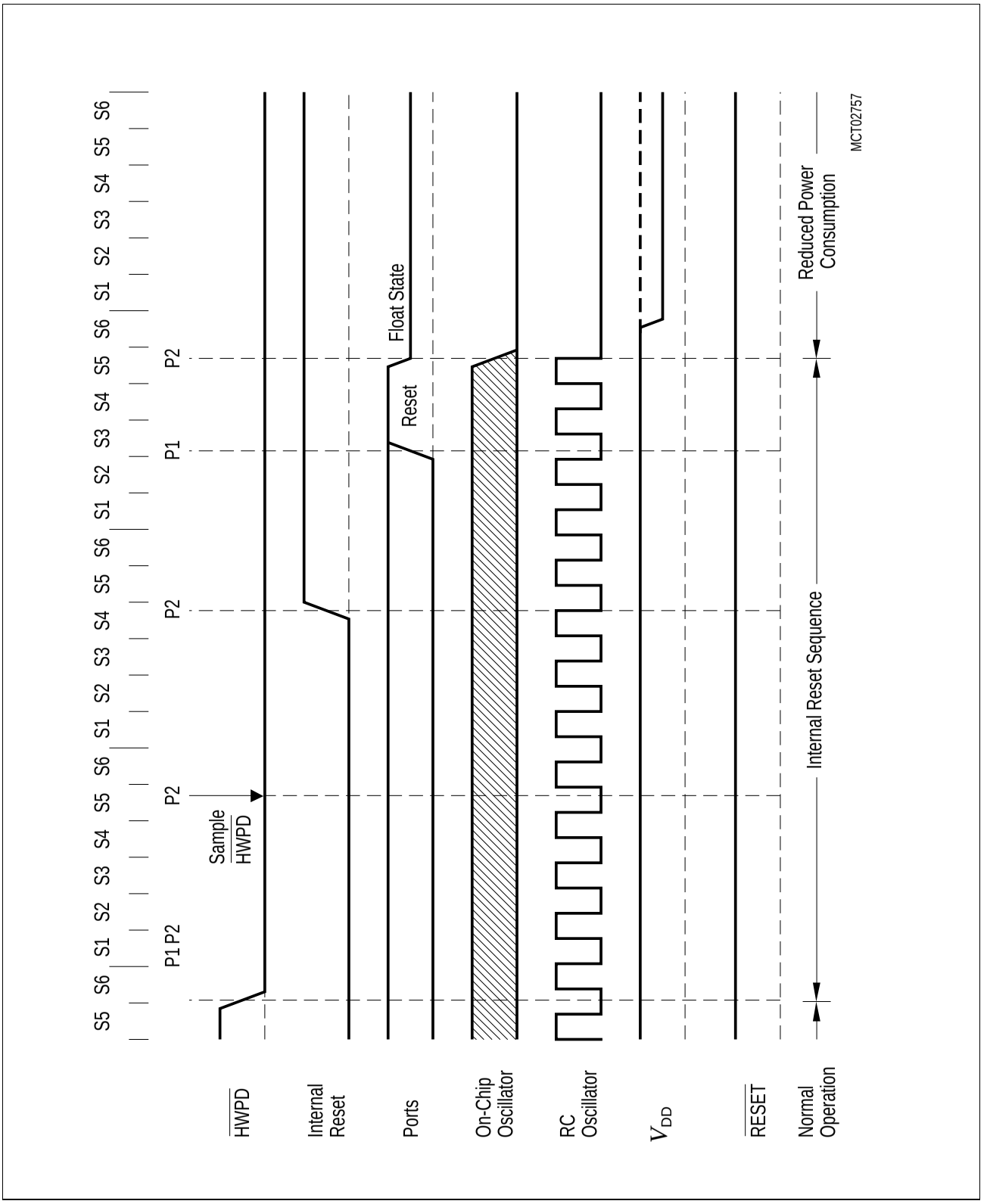


Figure 9-1
Timing Diagram of Entering Hardware Power Down Mode

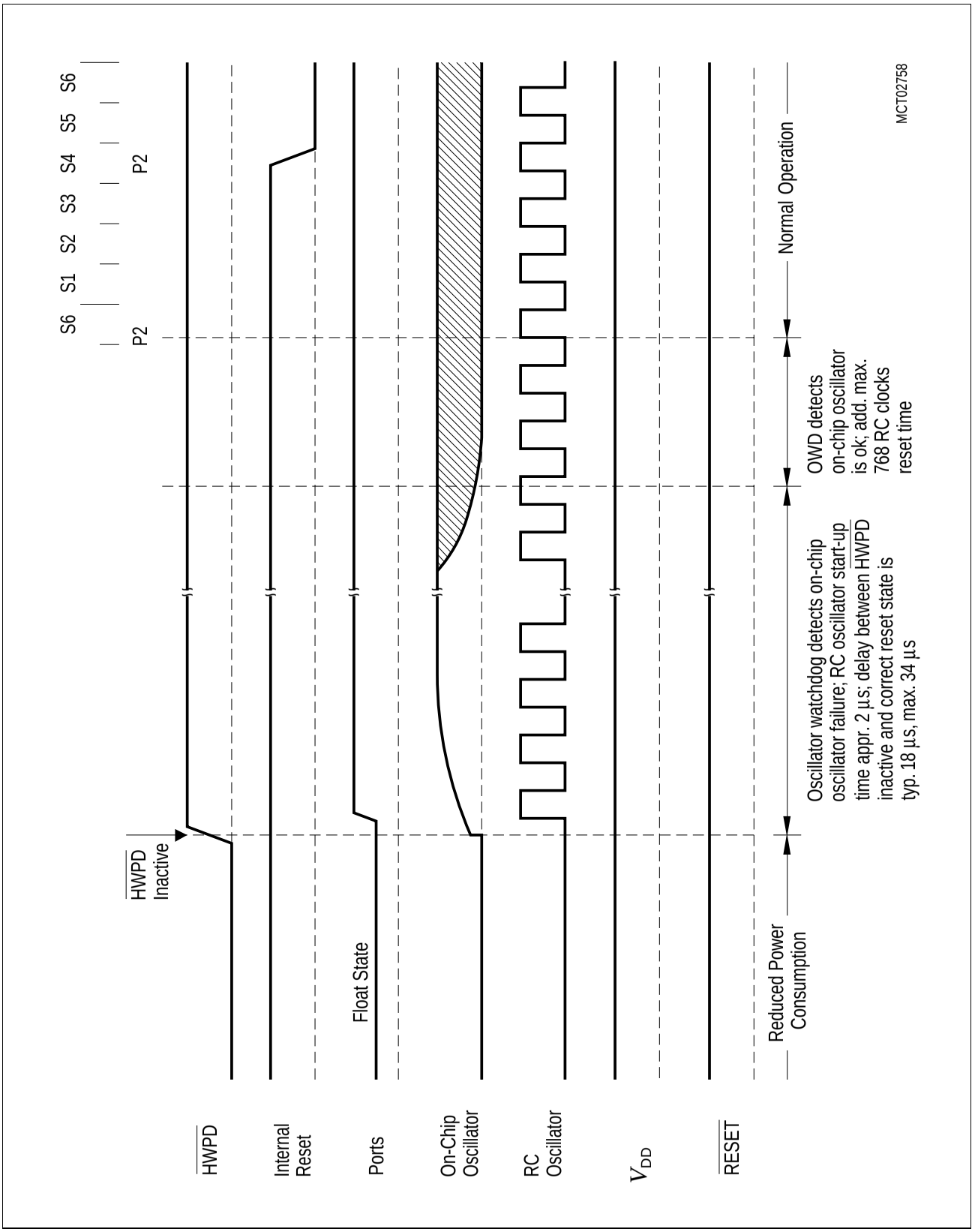


Figure 9-2
Timing Diagram of Leaving Hardware Power Down Mode

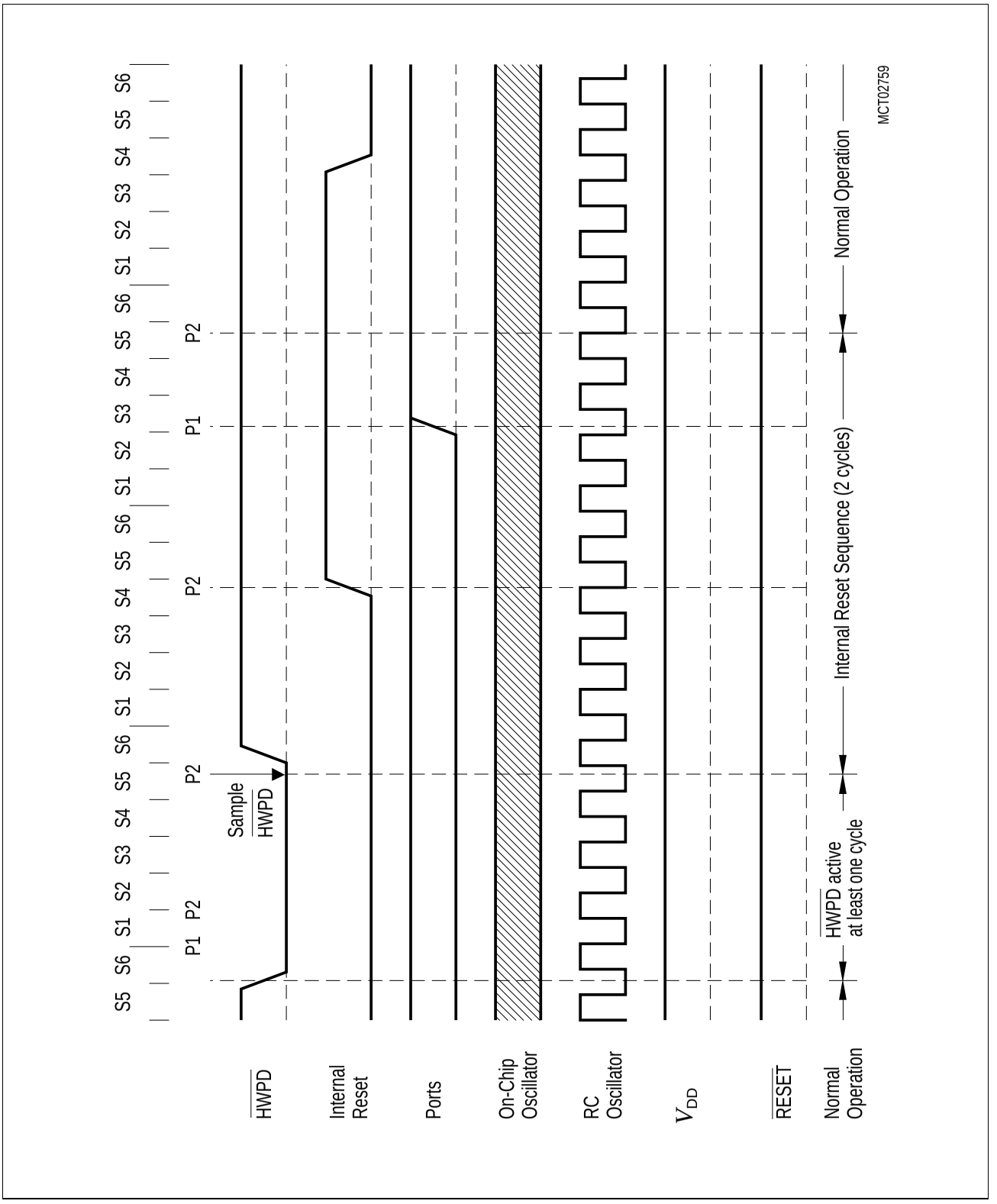


Figure 9-3
Timing Diagram of Hardware Power Down Mode, HWPD-Pin is active for only one cycle

10 Index

Note: Bold page numbers refer to the main definition part of SFRs or SFR bits.

D

E

F