SIEMENS

C505
C505C

8-Bit CMOS Microcontroller

User's Manual 08.97

C505 User's Manual Revision History :		1997-08-01
Previous Releases :		Original Version
Page (previous version)	Page (new version)	Subjects (changes since last revision)

Edition 1997-08-01

This edition was realized using the software system FrameMaker®.

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

Attention please!

As far as patents or other rights of third parties are concerned, liability is only assumed for components, not for applications, processes and circuits implemented within components or assemblies.

The information describes the type of component and shall not be considered as assured characteristics.

Terms of delivery and rights to change design reserved.

For questions on technology, delivery and prices please contact the Semiconductor Group Offices in Germany or the Siemens Companies and Representatives worldwide.

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Siemens Office, Semiconductor Group.

Siemens AG is an approved CECC manufacturer.

Packing

Please use the recycling operators known to you. We can also help you – get in touch with your nearest sales office. By agreement we will take packing material back, if it is sorted. You must bear the costs of transport.

For packing material that is returned to us unsorted or which we are not obliged to accept, we shall have to invoice you for any costs incurred.

Components used in life-support devices or systems must be expressly authorized for such purpose!

Critical components[1] of the Semiconductor Group of Siemens AG, may only be used in life-support devices or systems[2] with the express written approval of the Semiconductor Group of Siemens AG.

1 A critical component is a component used in a life-support device or system whose failure can reasonably be expected to cause the failure of that life-support device or system, or to affect its safety or effectiveness of that device or system.

2 Life support devices or systems are intended (a) to be implanted in the human body, or (b) to support and/or maintain and sustain human life. If they fail, it is reasonable to assume that the health of the user may be endangered.

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

1 Introduction

The C505 microcontroller is a member of the Siemens C500 family of 8-bit microcontrollers. The C505 is fully compatible to the standard 8051 microcontroller. Additionally the C505 provides extended power save provisions, on-chip RAM, 16K of on-chip program memory, and RFI related improvements. The C505 does not have an internal clock prescaler and with a maximum external clock rate of 20 MHz it achieves a 300 ns instruction cycle time.
The C505-2R operates with internal and/or external program memory. The C505-L is identical to the C505-2R, except that it lacks the on-chip porgram memory. The C505C-2R and C505C-L, are identical to the C505-2R and the C505-L respectively, except that they have, in addition, the full CAN interface. Therefore, the term C505 refers to all the above four versions within this documentation unless otherwise noted.
Figure 1-1 shows the different functional units of the C505 and **Figure 1-2** shows the simplified logic symbol of the C505.

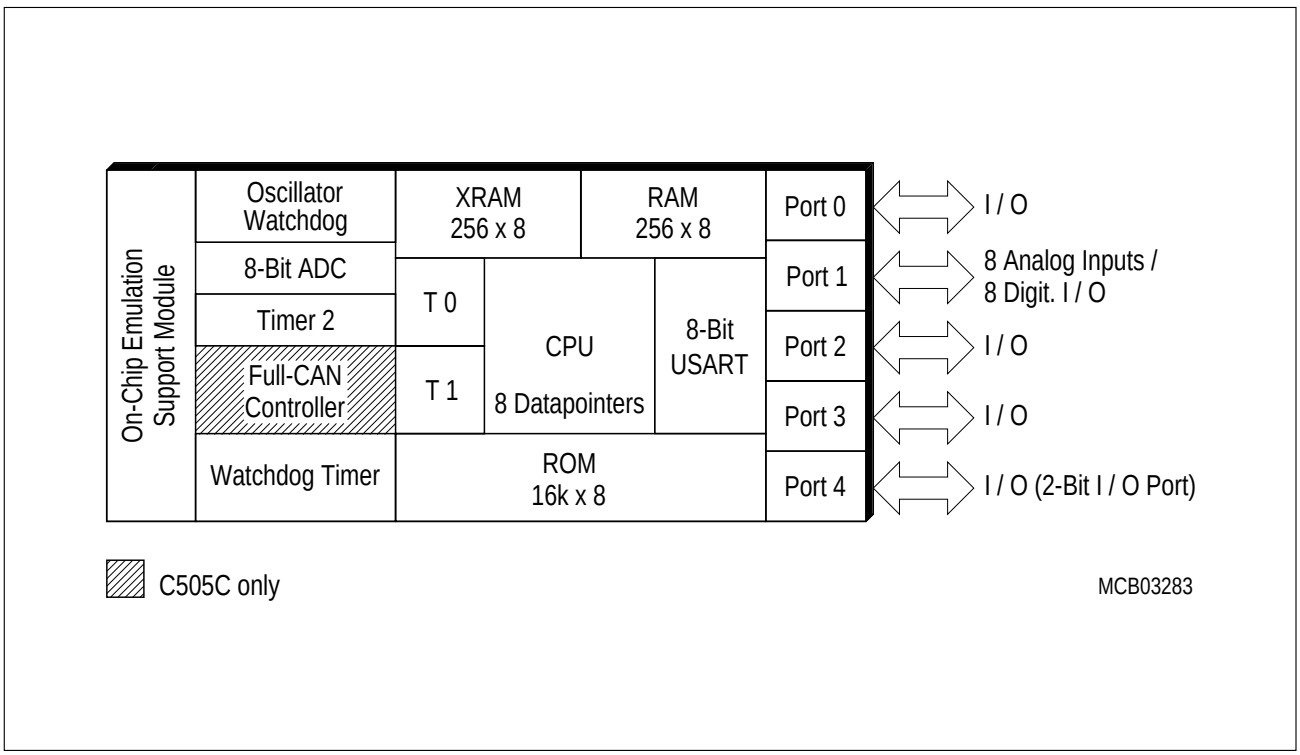


Figure 1-1
C505 Functional Units

Listed below is a summary of the main features of the C505 family:

- Fully compatible to standard 8051 microcontroller
- Superset of the 8051 architecture with 8 datapointers
- Up to 20 MHz operating frequency
  - 375 ns instruction cycle time @ 16 MHz
  - 300 ns instruction cycle time @ 20 MHz (50 % duty cycle)
- 16K byte on-chip ROM (C505-2R and C505C-2R only)
  - Optional ROM protection available
- 256 byte on-chip RAM
- 256 byte on-chip XRAM
- Five ports: 32 + 2 digital I/O lines(Port 1 with mixed analog/digital I/O capability)
- Three 16-bit timers/counters
  - Timer 0 / 1 (C501 compatible)
  - Timer 2 with 4 channels for 16-bit capture/compare operation
- Full CAN Module (C505C only)
  - 256 register/data bytes located in external data memory area
  - 1 MBaud CAN baudrate when operating frequency is equal to or above 8 MHz
  - internal CAN clock prescaler when input frequency is over 10 MHz
- Full duplex serial interface with programmable baudrate generator (USART)
- 8-bit A/D Converter
- Twelve interrupt sources with four priority levels
- On-chip emulation support logic (Enhanced Hooks)
- Programmable 15-bit Watchdog Timer
- Oscillator Watchdog
- Fast Power On Reset
- Power Saving Modes
  - Slow-down mode
  - Idle mode (can be combined with slow-down mode)
  - Software power-down mode with wake up capability through $\overline{\text{INT0}}$ or P4.1 pin
- P-MQFP-44 package
- Pin configuration is compatible to C501, C504, C511/C513-family
- Temperature ranges:

SAB-C505	T_A = 0 to 70 °C	
SAF-C505	T_A = – 40 to 85 °C	
SAH-C505	T_A = – 40 to 110 °C	(max. operating frequency: TBD)
SAK-C505	T_A = – 40 to 125 °C	(max. operating frequency: 8 MHz)

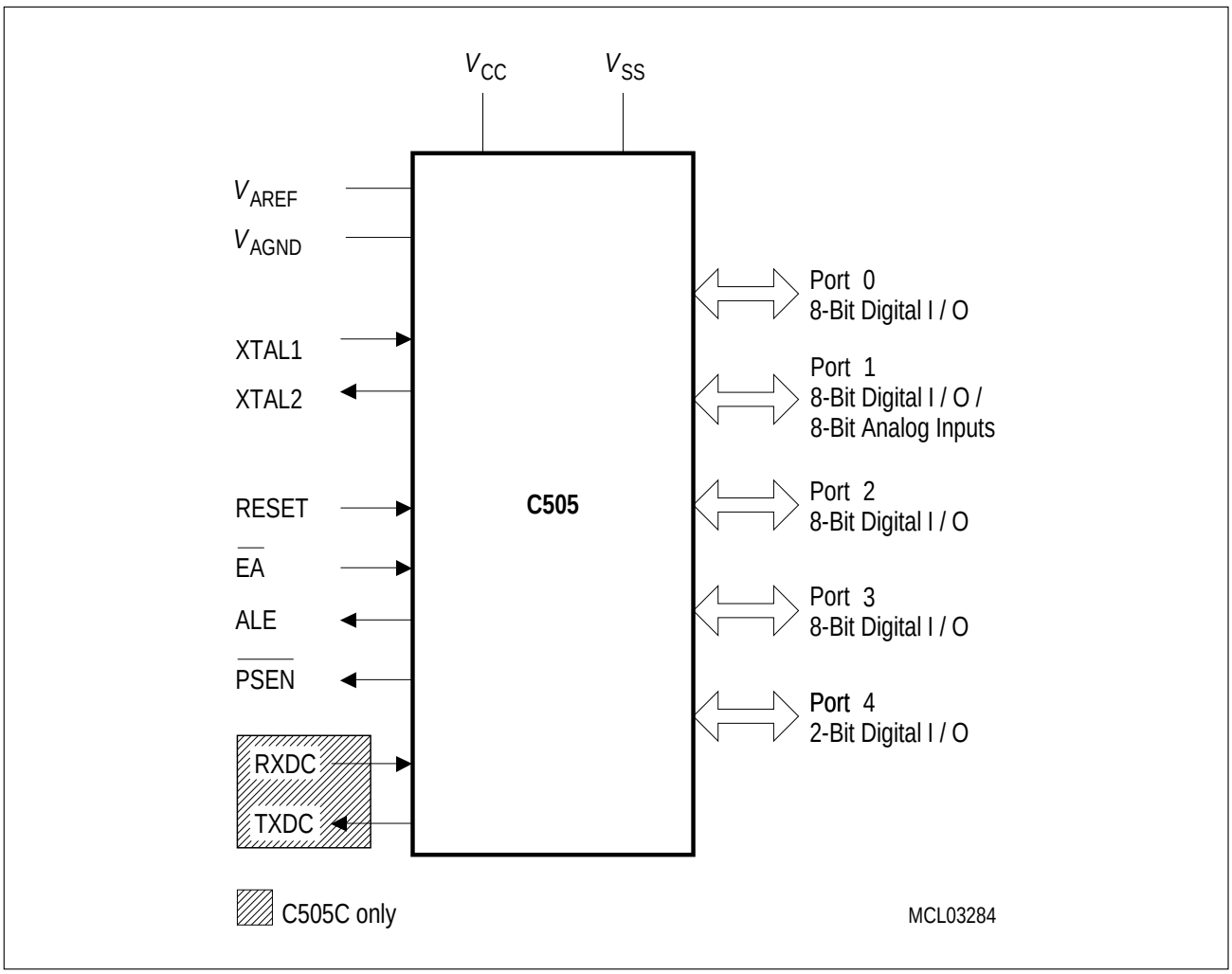


Figure 1-2
Logic Symbol

1.1 Pin Configuration

This section shows the pin configuration of the C505 in the P-MQFP-44 package.

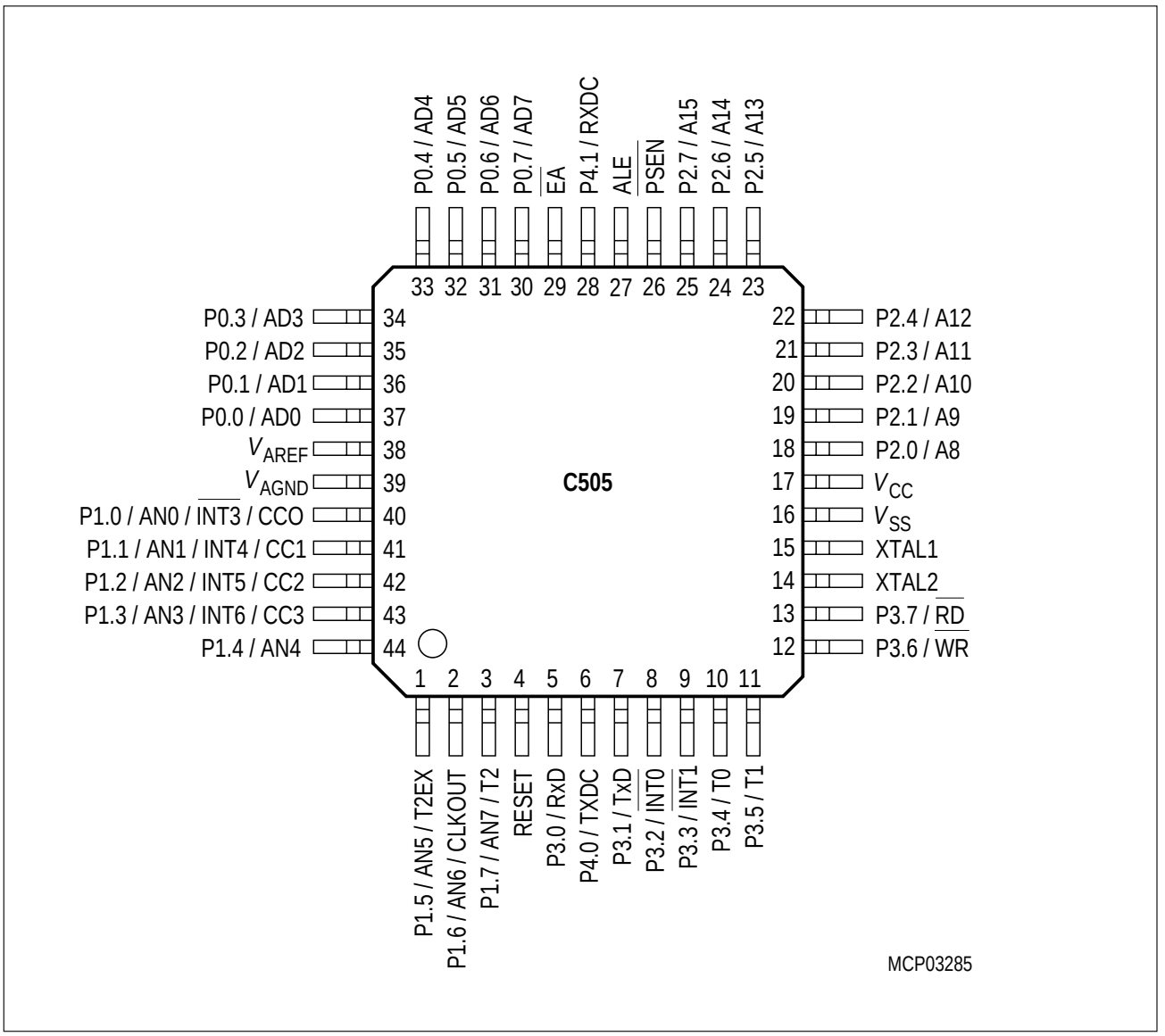


Figure 1-3
Pin Configuration (top view)

1.2 Pin Definitions and Functions

This section describes all external signals of the C505 with its function.

Table 1-1
Pin Definitions and Functions

Symbol	Pin Number	I/O*)	Function
P1.0-P1.7	40-44,1-3	I/O	**Port 1** is an 8-bit quasi-bidirectional port with internal pull-up arrangement. Port 1 pins can be used for digital input/output or as analog inputs of the A/D converter. Port 1 pins that have 1's written to them are pulled high by internal pull-up transistors 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 transistors. Port 1 pins are assigned to be used as analog inputs via the register P1ANA. As secondary digital functions, port 1 contains the interrupt, timer, clock, capture and compare pins. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate (except for compare functions). The secondary functions are assigned to the pins of port 1 as follows:
	40		P1.0 / AN0 / $\overline{\text{INT3}}$ / CC0 — Analog input channel 0 / Interrupt 3 input / capture/compare channel 0 I/O
	41		P1.1 / AN1 / INT4 / CC1 — Analog input channel 1/ Interrupt 4 input / capture/compare channel 1 I/O
	42		P1.2 / AN2 / INT5 / CC2 — Analog input channel 2 / Interrupt 5 input / capture/compare channel 2 I/O
	43		P1.3 / AN3 / INT6 / CC3 — Analog input channel 3 / Interrupt 6 input / capture/compare channel 4 I/O
	44		P1.4 / AN4 — Analog input channel 4
	1		P1.5 / AN5 / T2EX — Analog input channel 5 / Timer 2 external reload / trigger input
	2		P1.6 / AN6 / CLKOUT — Analog input channel 6 / System clock output
	3		P1.7 / AN7 / T2 — Analog input channel 7 / Counter 2 input
			Port 1 is used for the low-order address byte during program verification of the C505-2R.

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions

Symbol	Pin Number	I/O*)	Function
RESET	4	I	**RESET** A high level on this pin for two machine cycles while the oscillator is running resets the device. An internal diffused resistor to V_{SS} permits power-on reset using only an external capacitor to V_{CC}.
P3.0-P3.7	5, 7-13	I/O	**Port 3** is an 8-bit quasi-bidirectional port with internal pull-up arrangement. Port 3 pins that have 1's written to them are pulled high by the internal pull-up transistors 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 transistors. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate (except for TxD and $\overline{WR}$). The secondary functions are assigned to the pins of port 3 as follows:
	5		P3.0 / RxD — Receiver data input (asynch.) or data input/output (synch.) of serial interface
	7		P3.1 / TxD — Transmitter data output (asynch.) or clock output (synch.) of serial interface
	8		P3.2 / $\overline{INT0}$ — External interrupt 0 input / timer 0 gate control input
	9		P3.3 / $\overline{INT1}$ — External interrupt 1 input / timer 1 gate control input
	10		P3.4 / T0 — Timer 0 counter input
	11		P3.5 / T1 — Timer 1 counter input
	12		P3.6 / $\overline{WR}$ — $\overline{WR}$ control output; latches the data byte from port 0 into the external data memory
	13		P3.7 / $\overline{RD}$ — $\overline{RD}$ control output; enables the external data memory

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions

Symbol	Pin Number	I/O*)	Function
P4.0 P4.1	6 28	I/O I/O	**Port 4** is a 2-bit quasi-bidirectional port with internal pull-up arrangement. Port 4 pins that have 1's written to them are pulled high by the internal pull-up transistors 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 pullup transistors. The output latch corresponding to the secondary function RXDC must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the two pins of port 4 as follows (C505C only): P4.0 / TXDC Transmitter output of CAN controller P4.1 / RXDC Receiver input of CAN controller
XTAL2	14	O	**XTAL2** Output of the inverting oscillator amplifier.
XTAL1	15	I	**XTAL1** Input to the inverting oscillator amplifier and input to the internal clock generator circuits. To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected. To operate above a frequency of 16 MHz, a duty cycle of 50 % should be maintained. Minimum and maximum high and low times as well as rise/ fall times specified in the AC characteristics must be observed.

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions

Symbol	Pin Number	I/O*)	Function
P2.0-P2.7	18-25	I/O	**Port 2** is a 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 transistors when issuing 1s. During accesses to external data memory that use 8-bit addresses (MOVX @Ri), port 2 issues the contents of the P2 special function register and uses only the internal pullup resistors.
$\overline{\text{PSEN}}$	26	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 three oscillator periods except during external data memory accesses. Remains high during internal program execution. This pin should not be driven during reset operation.
ALE	27	O	The **Address Latch Enable** output is used for latching the low-byte of the address into external memory during normal operation. It is activated every three oscillator periodes except during an external data memory access. When instructions are executed from internal ROM ($\overline{\text{EA}}$=1) the ALE generation can be disabled by bit EALE in SFR SYSCON. This pin should not be driven during reset operation.

*) I = Input
O = Output

Table 1-1
Pin Definitions and Functions

Symbol	Pin Number	I/O*)	Function
$\overline{EA}$	29	I	**$\overline{\text{External Access}}$ Enable** When held at high level, instructions are fetched from the internal ROM when the PC is less than 4000_H. When held at low level, the C505 fetches all instructions from external program memory. This pin should not be driven during reset operation. For the C505-L and the C505C-L this pin must be tied low.
P0.0-P0.7	37-30	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-impendance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program or data memory. In this application it uses strong internal pullup transistors when issuing 1's. Port 0 also outputs the code bytes during program verification in the C505C-2R. External pullup resistors are required during program verification.
V_{AREF}	38	–	**Reference voltage** for the A/D converter.
V_{AGND}	39	–	**Reference ground** for the A/D converter.
V_{SS}	16	–	**Ground** (0V)
V_{CC}	17	–	**Power Supply** (+ 5 V)

*) I = Input
O = Output

2 Fundamental Structure

The C505 is fully compatible to the architecture of the standard 8051/C501 microcontroller family. While maintaining all architectural and operational characteristics of the C501, the C505 incorporates a CPU with 8 datapointers, an 8-bit A/D converter, a 4-channel capture/compare unit, a Full-CAN controller unit (C505C only), an XRAM data memory as well as some enhancements in the Fail Save Mechanism Unit. **Figure 2-1** shows a block diagram of the C505.

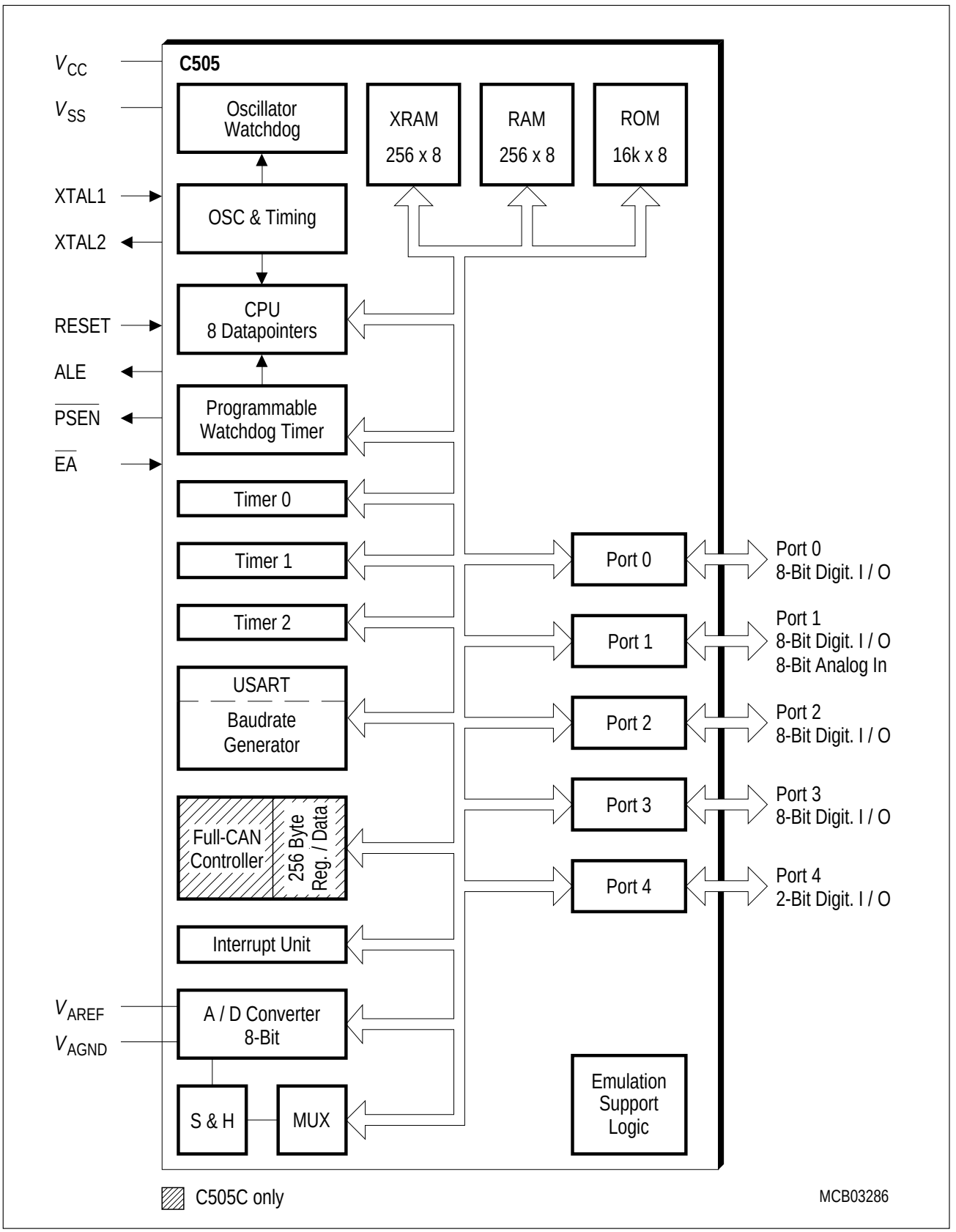


Figure 2-1
Block Diagram of the C505

2.1 CPU

The C505 is efficient both as a controller and as an arithmetic processor. It has extensive facilities for binary and BCD arithmetic and excels in its bit-handling capabilities. 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 16 MHz external clock, 58% of the instructions execute in 375 ns (20 MHz : 300 ns).

The CPU (Central Processing Unit) of the C505 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-set, 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.

Additionally to the CPU functionality of the C501/8051 standard microcontroller, the C505 contains 8 datapointers. 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.

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 instructions.
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 instructions.
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

The C505 has no clock prescaler. Therefore, a machine cycle of the C505 consists of 6 states (6 oscillator periods). Each state is devided into a phase 1 half and a phase 2 half. Thus, a machine cycle consists of 6 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.

Execution 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.

Most C505 instructions are executed in one cycle. MUL (multiply) and DIV (divide) are the only instructions that take more than two cycles to complete; they take four cycles. Normally two code bytes are fetched from the program memory during every machine cycle. The only exception to this is when a MOVX instruction is executed. MOVX is a one-byte, 2-cycle instruction that accesses external data memory. During a MOVX, the two fetches in the second cycle are skipped while the external data memory is being addressed and strobed. **Figure 2-2 (c)** and **(d)** show the timing for a normal 1-byte, 2-cycle instruction and for a MOVX instruction.

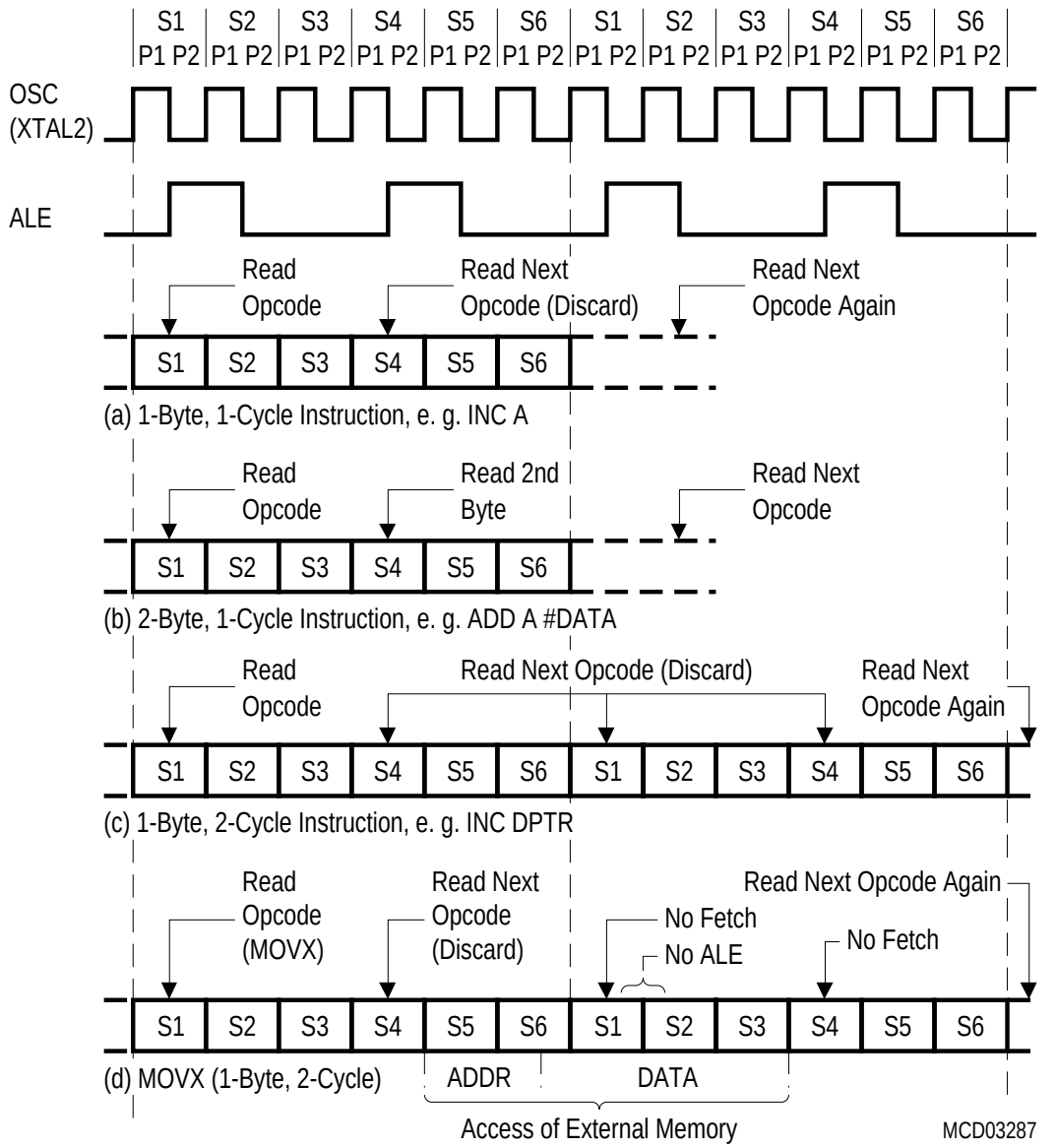


Figure 2-2
Fetch Execute Sequence

3 Memory Organization

The C505 CPU manipulates operands in the following four address spaces:

- up to 64 Kbytes of program memory (16K on-chip program memory for C505-2R)
- up to 64 Kbytes of external data memory
- 256 bytes of internal data memory
- 256 bytes of internal XRAM data memory
- 256 bytes CAN controller registers / data memory (C505C only)
- a 128 byte special function register area

Figure 3-1 illustrates the memory address spaces of the C505.

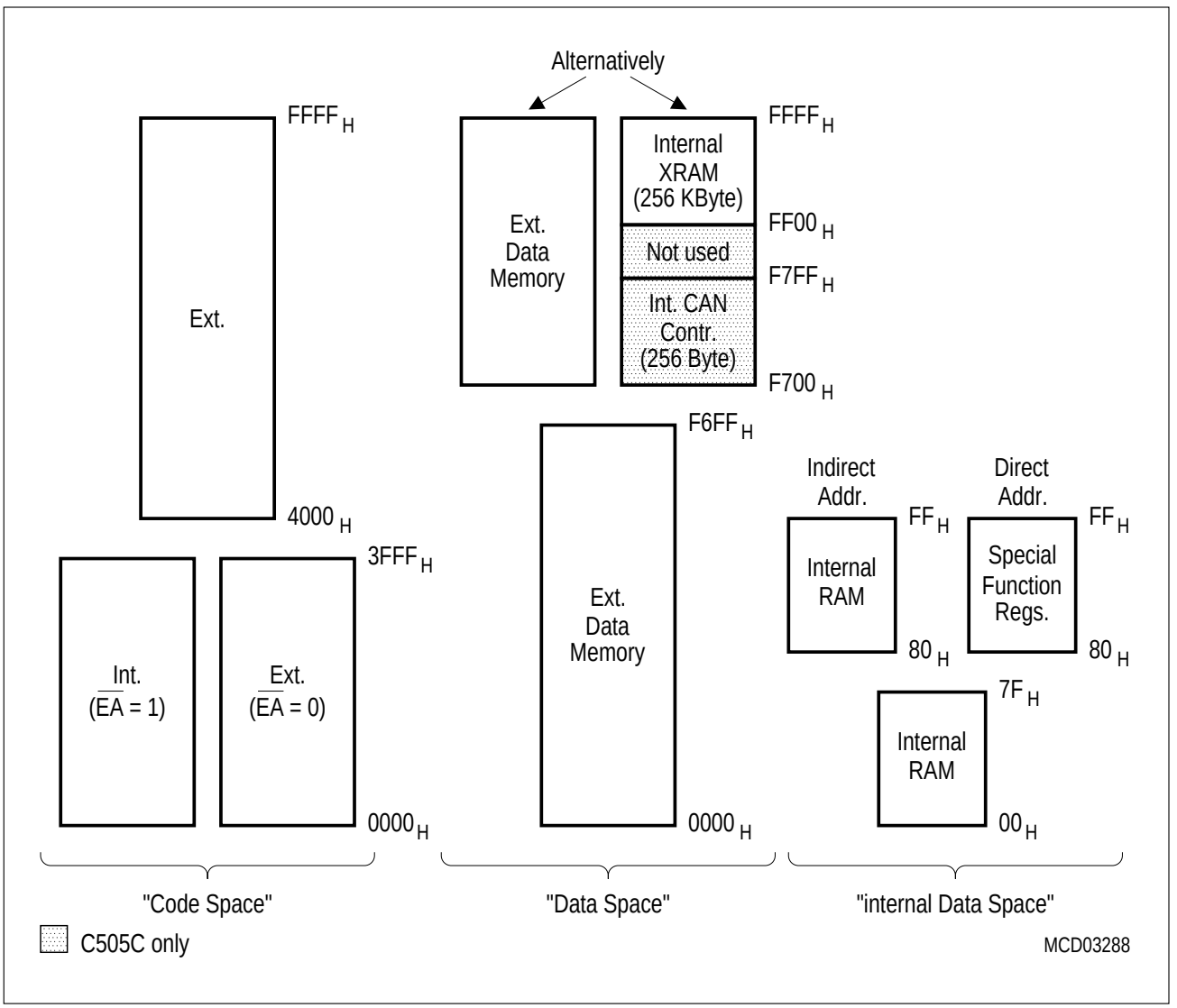


Figure 3-1
C505 Memory Map

3.1 Program Memory, "Code Space"

The C505-2R has 16 Kbytes of read-only program memory which can be externally expanded up to 64 Kbytes. If the $\overline{EA}$ pin is held high, the C505-2R executes program code out of the internal ROM unless the program counter address exceeds $3FFF_H$. Address locations 4000_H through $FFFF_H$ are then fetched from the external program memory. If the $\overline{EA}$ pin is held low, the C505 fetches all instructions from the external 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 general-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 RAM area, 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 CAN controller (C505C only) and the internal XRAM are located in the external memory address area at addresses $F700_H$ to $F7FF_H$ and $FF00_H$ to $FFFF_H$ respectively. The CAN controller registers and internal XRAM can therefore be accessed using MOVX instructions with addresses pointing to the respective address areas.

3.3 General Purpose Registers

The lower 32 locations of the internal RAM are assigned to four banks of 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 C505 is a memory area that is logically located at the upper end of the external data 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/CAN Controller Access Control

Two bits in SFR SYSCON, XMAP0 and XMAP1, control the accesses to XRAM and the CAN controller. XMAP0 is a general access enable/disable control bit and XMAP1 controls the external signal generation during XRAM/CAN controller accesses. CAN controller accesses are applicable only in the case of the C505C versions.

Special Function Register SYSCON (Address $B1_H$) **Reset Value : $XX100X01_B$**

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

The functions of the shaded bits are not described here.

Bit	Function
XMAP1	XRAM/CAN controller visible access control Control bit for $\overline{RD}$/$\overline{WR}$ signals during XRAM/CAN Controller accesses. If addresses are outside the XRAM/CAN controller 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/CAN Controller XMAP1 = 1 : Ports 0, 2 and the signals $\overline{RD}$ and $\overline{WR}$ are activated during accesses to XRAM/CAN Controller. In this mode, address and data information during XRAM/CAN Controller accesses are visible externally.
XMAP0	Global XRAM/CAN controller access enable/disable control XMAP0 = 0 : The access to XRAM and CAN controller is enabled. XMAP0 = 1 : The access to XRAM and CAN controller is disabled (default after reset!). All MOVX accesses are performed via the external bus. Further, this bit is hardware protected.
–	Reserved bits for future use. Read by CPU returns undefined values.

When bit XMAP1 in SFR SYSCON is set, during all accesses to XRAM and CAN Controller $\overline{RD}$ and $\overline{WR}$ become active and port 0 and 2 drive the actual address/data information which is read/written from/to XRAM or CAN controller. This feature allows to check the internal data transfers to XRAM and CAN controller. 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 set. This means that the accesses to XRAM and CAN controller are generally disabled. In this case, all accesses using MOVX instructions within the address range of $F700_H$ to $FFFF_H$ generate external data memory bus cycles. When XMAP0 is cleared, the access to XRAM and CAN controller is enabled and all accesses using MOVX instructions with an address in the range of $F700_H$ to $F7FF_H$ will access the CAN controller and $FF00_H$ to $FFFF_H$ will access the internal XRAM. Internal accesses (XMAP0=0) in the address range gap from $F800_H$ to $FEFF_H$ for the C505C (as shown in **figure 3-1**) will have undefined data. In the case of the pure C505 microcontroller (without CAN controller), internal accesses in the address range of $F700_H$ to $FEFF_H$ will have undefined data

Bit XMAP0 is hardware protected. If it is cleared once (XRAM and CAN controller 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 asymmetric latch at XMAP0 bit. An unintentional disabling of XRAM and CAN controller could be dangerous since indeterminate values could be read from the external bus. To avoid this the XMAP0 bit is forced to '1' only by a reset operation. Additionally, during reset an internal capacitor is charged. So the reset state is a disabled XRAM and CAN controller. Because of the charge 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 charge this capacitor, too. That is, the stable status is XRAM and CAN controller enabled.

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

Note: The CAN controller peripheral exists in the **C505C** only.

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

The XRAM and CAN controller 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 $FF00_H$ to $FFFF_H$. For accessing the CAN controller, the effective address stored in DPTR must be in the range of $F700_H$ to $F7FF_H$.

3.4.3 Accesses to XRAM using the Registers R0/R1 (8-bit Addressing Mode)

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)

Aspecial page register is implemented in the C505 to provide the possibility of accessing the XRAM or CAN controller also with the MOVX @Ri instructions, i.e. XPAGE serves the same function for the XRAM and CAN controller 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/CAN controller high address XPAGE.7-0 is the address part A15-A8 when 8-bit MOVX instructions are used to access internal XRAM or CAN controller.

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

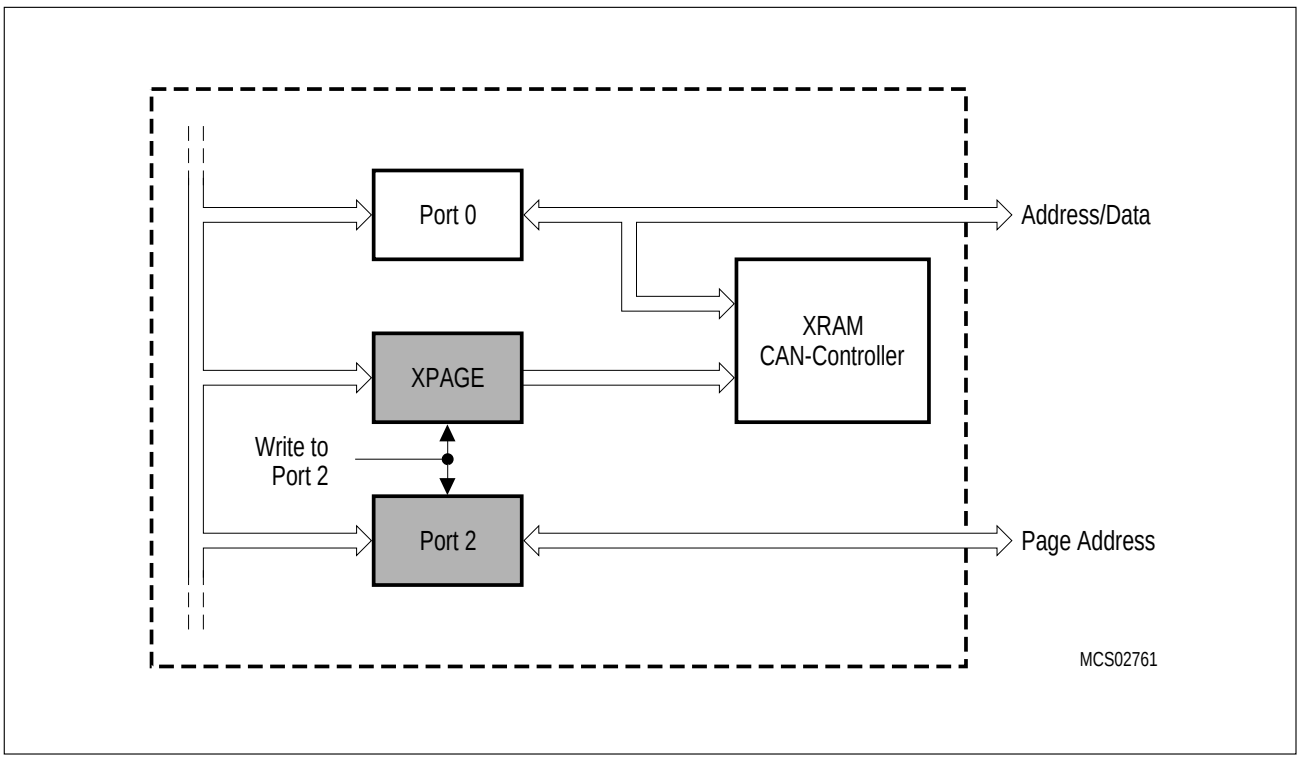


Figure 3-2
Write Page Address to Port 2

“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/CAN controller address range, the XRAM/CAN controller has to be disabled. When additional external RAM is to be addressed in an address range < $F700_H$, the XRAM/CAN controller may remain enabled and there is no need to overwrite XPAGE by a second move.

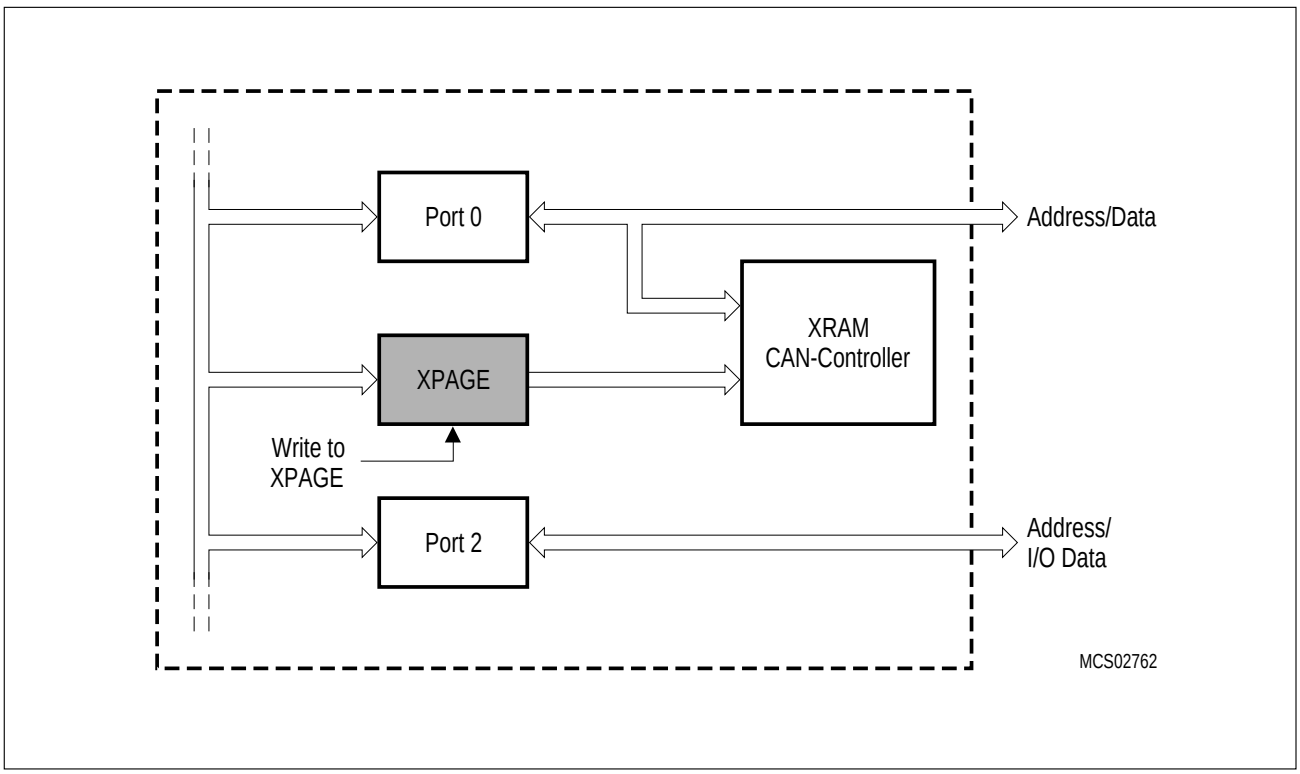


Figure 3-3
Write Page Address to XPAGE

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

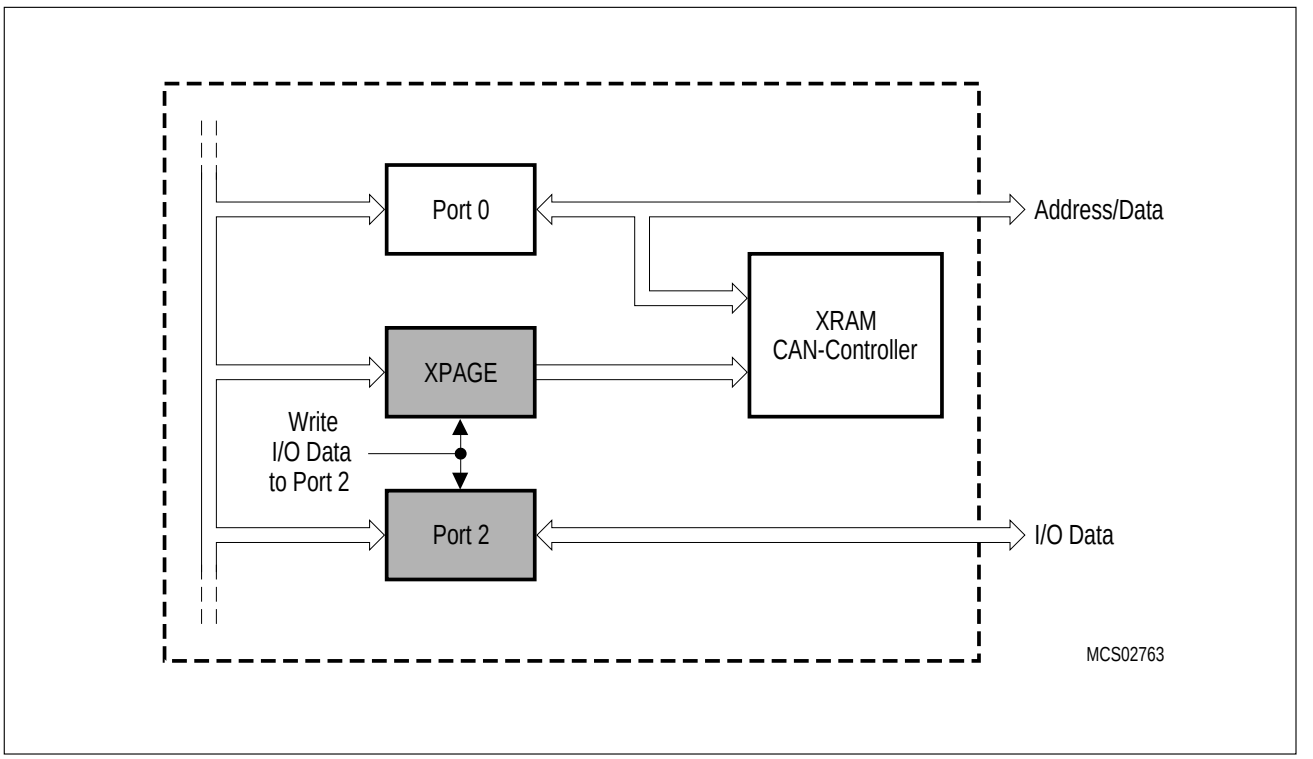


Figure 3-4
Use of Port 2 as I/O Port

At a write to port 2, the XRAM/CAN controller address in 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 $FF30_H$.

```
MOV     R0, #30H        ;
MOV     P2, #0AAH       ; P2 shows AAH and XPAGE contains AAH
MOV     XPAGE, #0FFH    ; P2 still shows AAH but XRAM is addressed
MOVX    A, @R0          ; the contents of XRAM at FF30H 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/CAN Controller address range, an external access is performed. For the C505 the content of XPAGE must be $F7_H$ FF_H in order to use the XRAM/CAN Controller.

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

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

b) Access to external memory : The upper address byte must be written to P2; XPAGE will be automatically 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 Port 0 and Port 2

The behaviour of port 0 and port 2 during a MOVX access depends on the control bits in register SYSCON and on the state of pin $\overline{EA}$. The **table 3-1** 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/CAN controller is accessed, the data written to the XRAM/CAN controller 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 (XRAM/CAN controller) or external XDATA memory.

The shaded areas describe the standard operation as each C5xx device without on-chip XRAM/ CAN controller behaves.

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

modes compatible to 8051/C501 family

Table 3-1 - 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. The special function register area consists of two portions : the standard special function register area and the mapped special function register area. One special function register of the C505 (PCON1) is located in the mapped special function register area. For accessing the mapped special function register area, bit RMAP in special function register SYSCON must be set. All other special function registers are located in the standard special function register area which is accessed when RMAP is cleared (“0“).

The registers and data locations of the CAN controller (CAN-SFRs) are located in the external data memory area at addresses $F700_H$ to $F7FF_H$. Details about the access of these registers is described in section 3.4.1 of this chapter.

Special Function Register SYSCON (Address $B1_H$) **Reset Value : $XX100X01_B$**

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

The functions of the shaded bits are not described here.

Bit	Function
RMAP	Special function register map bit RMAP = 0 : The access to the non-mapped (standard) special function register area is enabled. RMAP = 1 : The access to the mapped special function register area is enabled.
–	Reserved bits for future use. Read by CPU returns undefined values.

As long as bit RMAP is set, mapped special function register area can be accessed. This bit is not cleared by hardware automatically. Thus, when non-mapped/mapped registers are to be accessed, the bit RMAP must be cleared/set respectively by software.

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 52 special function registers (SFRs) in the standard and mapped SFR area include pointers and registers that provide an interface between the CPU and the other on-chip peripherals. The SFRs of the C505 are listed in **table 3-2** and **table 3-3**. In **table 3-2** they are organized in groups which refer to the functional blocks of the C505. The CAN-SFRs (applicable for the C505C only) are also included in **table 3-2**. **Table 3-3** illustrates the contents of the SFRs in numeric order of their addresses. **Table 3-4** list the CAN-SFRs in numeric order of their addresses. .

Table 3-2
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	$XXXXX000_B$ 3)
	PSW	Program Status Word Register	**$D0_H$** 1)	00_H
	SP	Stack Pointer	81_H	07_H
	SYSCON 2)	System Control Register	$B1_H$	$XX100X01_B$ 3)
	VR0 4)	Version Register 0	FC_H	$C5_H$
	VR1 4)	Version Register 1	FD_H	05_H
	VR2 4)	Version Register 2	FE_H	5)
A/D-Converter	ADCON0 2)	A/D Converter Control Register 0	**$D8_H$** 1)	$00X00000_B$ 3)
	ADCON1	A/D Converter Control Register 1	DC_H	$01XXX000_B$ 3)
	ADDAT	A/D Converter Data Register	$D9_H$	00_H
	ADST	A/D Converter Start Register	DA_H	XX_H 3)
	P1ANA 2)	Port 1 Analog Input Selection Register	90_H 4)	FF_H
Interrupt System	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
	IP1	Interrupt Priority Register 1	$B9_H$	$XX000000_B$ 3)
	TCON 2)	Timer Control Register	**88_H** 1)	00_H
	T2CON 2)	Timer 2 Control Register	**$C8_H$** 1)	$00X00000_B$
	SCON 2)	Serial Channel Control Register	**98_H** 1)	00_H
	IRCON	Interrupt Request Control Register	**$C0_H$** 1)	00_H
XRAM	XPAGE	Page Address Register for Extended on-chip XRAM and CAN Controller	91_H	00_H
	SYSCON 2)	System Control Register	$B1_H$	$XX100X01_B$ 3)
Ports	P0	Port 0	**80_H** 1)	FF_H
	P1	Port 1	**90_H** 1)	FF_H
	P1ANA 2)	Port 1 Analog Input Selection Register	**90_H** 1) 4)	FF_H
	P2	Port 2	**$A0_H$** 1)	FF_H
	P3	Port 3	**$B0_H$** 1)	FF_H
	P4	Port 4	**E8H** 1)	$XXXXXX11_B$

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
4) This SFR is a mapped SFR. For accessing this SFR, bit RMAP in SFR SYSCON must be set.
5) The content of this SFR varies with the actual step of the C505 (eg. 01_H for the first step)

Table 3-2
Special Function Registers - Functional Blocks (cont'd)

Block	Symbol	Name	Address	Contents after Reset
Serial Channel	ADCON0 2)	A/D Converter Control Register 0	**$D8_H$** 1	$00X00000_B$ 3)
	PCON 2)	Power Control Register	87_H	00_H
	SBUF	Serial Channel Buffer Register	99_H	XX_H 3)
	SCON	Serial Channel Control Register	**98_H** 1)	00_H
	SRELL	Serial Channel Reload Register, low byte	AA_H	$D9_H$
	SRELH	Serial Channel Reload Register, high byte	BA_H	$XXXXXX11_B$ 3)
Timer 0/ Timer 1	TCON	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
Compare/ Capture Unit / Timer 2	CCEN	Comp./Capture Enable Reg.	$C1_H$	00_H 3)
	CCH1	Comp./Capture Reg. 1, High Byte	$C3_H$	00_H
	CCH2	Comp./Capture Reg. 2, High Byte	$C5_H$	00_H
	CCH3	Comp./Capture Reg. 3, High Byte	$C7_H$	00_H
	CCL1	Comp./Capture Reg. 1, Low Byte	$C2_H$	00_H
	CCL2	Comp./Capture Reg. 2, Low Byte	$C4_H$	00_H
	CCL3	Comp./Capture Reg. 3, Low Byte	$C6_H$	00_H
	CRCH	Reload Register High Byte	CB_H	00_H
	CRCL	Reload Register Low Byte	CA_H	00_H
	TH2	Timer 2, High Byte	CD_H	00_H
	TL2	Timer 2, Low Byte	CC_H	00_H
	T2CON	Timer 2 Control Register	**$C8_H$** 1)	$00X00000_B$ 3)
	IEN0 2)	Interrupt Enable Register 0	**$A8_H$** 1)	00_H
	IEN1 2)	Interrupt Enable Register 1	**$B8_H$** 1)	00_H
Watchdog	WDTREL	Watchdog Timer Reload Register	86_H	00_H
	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
Power Save Modes	PCON 2)	Power Control Register	87_H	00_H
	PCON1 4)	Power Control Register 1	**88_H** 1)	$0XX0XXXX_B$ 3)

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
4) SFR is located in the mapped SFR area. For accessing this SFR, bit RMAP in SFR SYSCON must be set.

Table 3-2
Special Function Registers - Functional Blocks (cont'd)

Block	Symbol	Name	Address	Contents after Reset
CAN Controller (C505C)	CR	Control Register	$F700_H$	01_H
	SR	Status Register	$F701_H$	XX_H 3)
	IR	Interrupt Register	$F702_H$	XX_H 3)
	BTR0	Bit Timing Register Low	$F704_H$	UU_H 3)
	BTR1	Bit Timing Register High	$F705_H$	$0UUUUUUU_B$ 3)
	GMS0	Global Mask Short Register Low	$F706_H$	UU_H 3)
	GMS1	Global Mask Short Register High	$F707_H$	$UUU11111_B$ 3)
	UGML0	Upper Global Mask Long Register Low	$F708_H$	UU_H 3)
	UGML1	Upper Global Mask Long Register High	$F709_H$	UU_H 3)
	LGML0	Lower Global Mask Long Register Low	$F70A_H$	UU_H 3)
	LGML1	Lower Global Mask Long Register High	$F70B_H$	$UUUUU000_B$ 3)
	UMLM0	Upper Mask of Last Message Register Low	$F70C_H$	UU_H 3)
	UMLM1	Upper Mask of Last Message Register High	$F70D_H$	UU_H 3)
	LMLM0	Lower Mask of Last Message Register Low	$F70E_H$	UU_H 3)
	LMLM1	Lower Mask of Last Message Register High	$F70F_H$	$UUUUU000_B$ 3)
		Message Object Registers :		
	MCR0	Message Control Register Low	$F7n0_H$ 5)	UU_H 3)
	MCR1	Message Control Register High	$F7n1_H$ 5)	UU_H 3)
	UAR0	Upper Arbitration Register Low	$F7n2_H$ 5)	UU_H 3)
	UAR1	Upper Arbitration Register High	$F7n3_H$ 5)	UU_H 3)
	LAR0	Lower Arbitration Register Low	$F7n4_H$ 5)	UU_H 3)
	LAR1	Lower Arbitration Register High	$F7n5_H$ 5)	$UUUUU000_B$ 3)
	MCFG	Message Configuration Register	$F7n6_H$ 5)	$UUUUUU00_B$ 3)
	DB0	Message Data Byte 0	$F7n7_H$ 5)	XX_H 3)
	DB1	Message Data Byte 1	$F7n8_H$ 5)	XX_H 3)
	DB2	Message Data Byte 2	$F7n9_H$ 5)	XX_H 3)
	DB3	Message Data Byte 3	$F7nA_H$ 5)	XX_H 3)
	DB4	Message Data Byte 4	$F7nB_H$ 5)	XX_H 3)
	DB5	Message Data Byte 5	$F7nC_H$ 5)	XX_H 3)
	DB6	Message Data Byte 6	$F7nD_H$ 5)	XX_H 3)
	DB7	Message Data Byte 7	$F7nE_H$ 5)	XX_H 3)

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. "U" means that the value is unchanged by a reset operation. "U" values are undefined (as "X") after a power-on reset operation
4) SFR is located in the mapped SFR area. For accessing this SFR, bit RMAP in SFR SYSCON must be set.
5) The notation "n" (n= 1 to F) in the message object address definition defines the number of the related message object.

Table 3-3
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
86_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
88_H [3)]	PCON1	$0XX0\text{-}XXXX_B$	EWPD	–	–	WS	–	–	–	–
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	.4	.3	INT5	INT4	.0
90_H [3)]	P1ANA	FF_H	EAN7	EAN6	EAN5	EAN4	EAN3	EAN2	EAN1	EAN0
91_H	XPAGE	00_H	.7	.6	.5	.4	.3	.2	.1	.0
92_H	DPSEL	$XXXX\text{-}X000_B$	–	–	–	–	–	.2	.1	.0
98_H [2)]	SCON	00_H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
99_H	SBUF	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$A0_H$ [2)]	P2	FF_H	.7	.6	.5	.4	.3	.2	.1	.0
$A8_H$ [2)]	IEN0	00_H	EA	WDT	ET2	ES	ET1	EX1	ET0	EX0
$A9_H$	IP0	00_H	OWDS	WDTS	.5	.4	.3	.2	.1	.0
AA_H	SRELL	$D9_H$	.7	.6	.5	.4	.3	.2	.1	.0

1) X means that the value is undefined and the location is reserved
2) Bit-addressable special function registers
3) SFR is located in the mapped SFR area. For accessing this SFR, bit RMAP in SFR SYSCON must be set.

Table 3-3
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
$B0_H$[2)]	P3	FF_H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TxD	RxD
$B1_H$	SYSCON	$XX10\text{-}0X01_B$	–	–	EALE	RMAP	CMOD	–	XMAP1	XMAP0
$B8_H$[2)]	IEN1	00_H	EXEN2	SWDT	EX6	EX5	EX4	EX3	ECAN	EADC
$B9_H$	IP1	$XX00\text{-}0000_B$	–	–	.5	.4	.3	.2	.1	.0
BA_H	SRELH	$XXXX\text{-}XX11_B$	–	–	–	–	–	–	.1	.0
$C0_H$[2)]	IRCON	00_H	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	SWI	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	$00X0\text{-}0000_B$	T2PS	I3FR	–	T2R1	T2R0	T2CM	T2I1	T2I0
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
$D0_H$[2)]	PSW	00_H	CY	AC	F0	RS1	RS0	OV	F1	P
$D8_H$[2)]	ADCON0	$00X0\text{-}0000_B$	BD	CLK	–	BSY	ADM	MX2	MX1	MX0
$D9_H$	ADDAT	00_H	.7	.6	.5	.4	.3	.2	.1	.0
DA_H	ADST	$XXXX\text{-}XXXX_B$	–	–	–	–	–	–	–	–

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

Table 3-3
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
DC_H	ADCON1	$01XX\text{-}X000_B$	ADCL1	ADCL0	–	–	–	MX2	MX1	MX0
$E0_H$[2)]	ACC	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E8_H$[2)]	P4	$XXXX\text{-}XX11_B$	–	–	–	–	–	–	RXDC	TXDC
$F0_H$[2)]	B	00_H	.7	.6	.5	.4	.3	.2	.1	.0
FC_H[3)4)]	VR0	$C5_H$	1	1	0	0	0	1	0	1
FD_H[3)4)]	VR1	05_H	0	0	0	0	0	1	0	1
FE_H[3)4)]	VR2	5)	.7	.6	.5	.4	.3	.2	.1	.0

1) X means that the value is undefined and the location is reserved.
2) Bit-addressable special function registers.
3) SFR is located in the mapped SFR area. For accessing this SFR, bit RMAP in SFR SYSCON must be set.
4) These are read-only registers.
5) The content of this SFR varies with the actual of the step C505 (eg. 01_H for the first step).

Table 3-4
Contents of the CAN Registers in numeric order of their addresses (C505C only)

Addr. n=1-F_H 1)	Register	Content after Reset 2)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$F700_H$	CR	01_H	TEST	CCE	0	0	EIE	SIE	IE	INIT
$F701_H$	SR	XX_H	BOFF	EWRN	–	RXOK	TXOK	LEC2	LEC1	LEC0
$F702_H$	IR	XX_H	INTID							
$F704_H$	BTR0	UU_H	SJW		BRP					
$F705_H$	BTR1	$0UUU.UUUU_B$	0	TSEG2			TSEG1			
$F706_H$	GMS0	UU_H	ID28-21							
$F707_H$	GMS1	$UUU1.1111_B$	ID20-18			1	1	1	1	1
$F708_H$	UGML0	UU_H	ID28-21							
$F709_H$	UGML1	UU_H	ID20-13							
$F70A_H$	LGML0	UU_H	ID12-5							
$F70B_H$	LGML1	$UUUU.U000_B$	ID4-0					0	0	0
$F70C_H$	UMLM0	UU_H	ID28-21							
$F70D_H$	UMLM1	UU_H	ID20-18			ID17-13				
$F70E_H$	LMLM0	UU_H	ID12-5							
$F70F_H$	LMLM1	$UUUU.U000_B$	ID4-0					0	0	0
$F7n0_H$	MCR0	UU_H	MSGVAL		TXIE		RXIE		INTPND	
$F7n1_H$	MCR1	UU_H	RMTPND		TXRQ		MSGLST CPUUPD		NEWDAT	
$F7n2_H$	UAR0	UU_H	ID28-21							
$F7n3_H$	UAR1	UU_H	ID20-18			ID17-13				
$F7n4_H$	LAR0	UU_H	ID12-5							
$F7n5_H$	LAR1	$UUUU.U000_B$	ID4-0					0	0	0
$F7n6_H$	MCFG	$UUUU.UU00_B$	DLC				DIR	XTD	0	0

1) The notation “n” (n= 1 to F) in the address definition defines the number of the related message object.
2) “X” means that the value is undefined and the location is reserved. “U” means that the value is unchanged by a reset operation. “U” values are undefined (as “X”) after a power-on reset operation.

Table 3-4
Contents of the CAN Registers in numeric order of their addresses (cont'd) (C505C only)

Addr. n=1-F_H 1)	Register	Content after Reset 2)	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
$F7n7_H$	DB0n	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$F7n8_H$	DB1n	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$F7n9_H$	DB2n	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$F7nA_H$	DB3n	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$F7nB_H$	DB4n	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$F7nC_H$	DB5n	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$F7nD_H$	DB6n	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
$F7nE_H$	DB7n	XX_H	.7	.6	.5	.4	.3	.2	.1	.0

1) The notation "n" (n= 1 to F) in the address definition defines the number of the related message object.
2) "X" means that the value is undefined and the location is reserved. "U" means that the value is unchanged by a reset operation. "U" values are undefined (as "X") after a power-on reset operation.

4 External Bus Interface

The C505 allows for external memory expansion. The functionality and implementation of the external bus interface is identical to the common interface for the 8051 architecture with one exception : if the C505 is used in systems with no external memory the generation of the ALE signal can be suppressed. Resetting bit EALE in SFR SYSCON register, the ALE signal will be gated off. This feature reduces RFI emissions of the system.

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.

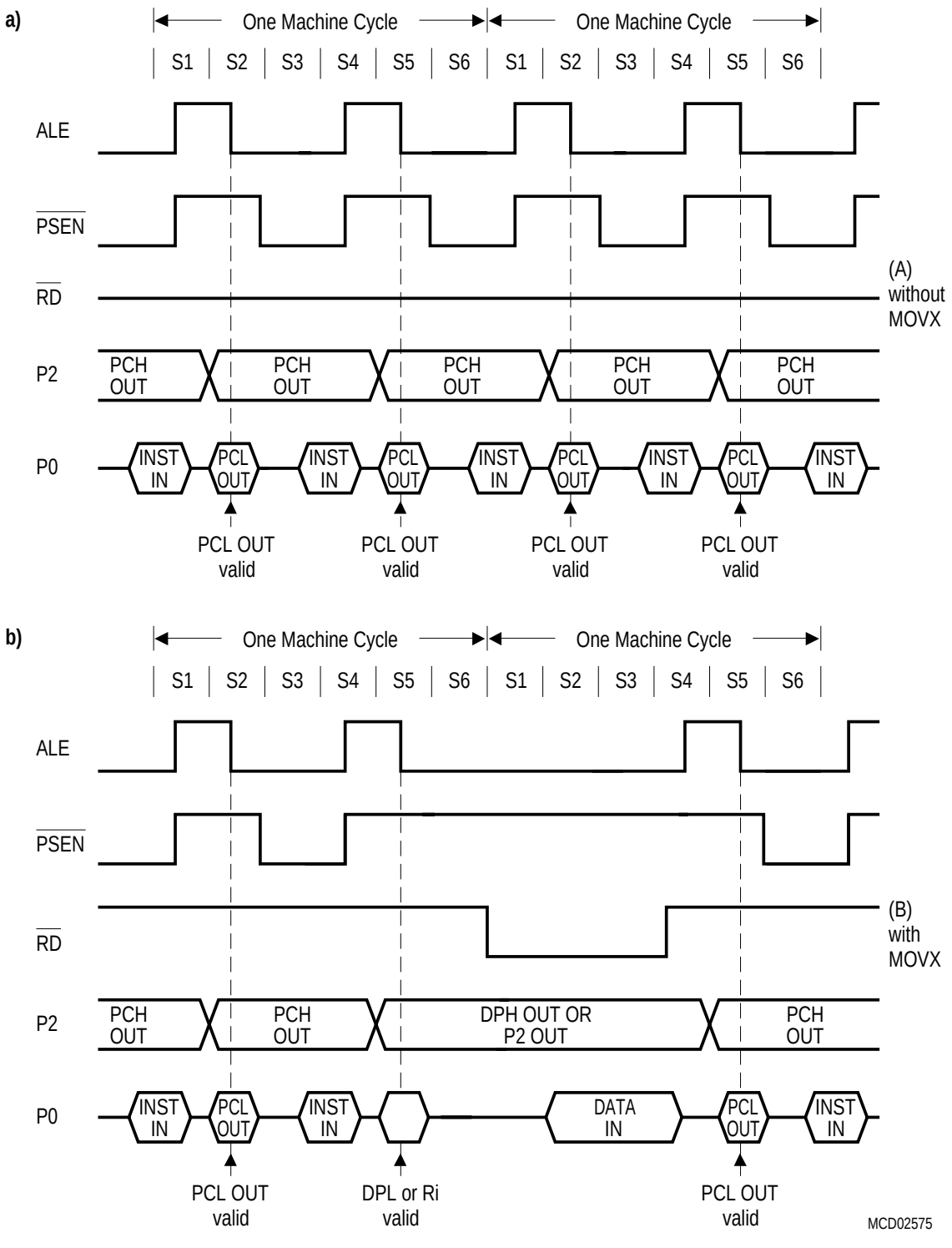


Figure 4-1
External Program Memory Execution

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 $3FFF_H$

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 C505 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 ALE, Address Latch Enable

The C505 allows to switch off the ALE output signal. If the internal ROM is used ($\overline{EA}$=1 and PC $\leq$ $3FFF_H$) and ALE is switched off by EALE=0, then, ALE will only go active during external data memory accesses (MOVX instructions). If $\overline{EA}$=0, the ALE generation is always enabled and the bit EALE has no effect.

After a hardware reset the ALE generation is enabled.

Special Function Register SYSCON (Address $B1_H$) **Reset Value : $XX100X01_B$**

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

The shaded bits are not described in this section.

Bit	Function
EALE	Enable ALE output EALE = 0 : ALE generation is disabled; disables ALE signal generation during internal code memory accesses ($\overline{EA}$=1). With $\overline{EA}$=1, ALE is automatically generated at MOVX instructions. EALE = 1 : ALE generation is enabled If $\overline{EA}$=0, the ALE generation is always enabled and the bit EALE has no effect on the ALE generation.
–	Reserved bits for future use. Read by CPU returns undefined values.

4.5 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 support of the Enhanced Hooks Emulation Concept. Therefore, no costly bond-out chips are necessary for emulation. This also ensure 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.

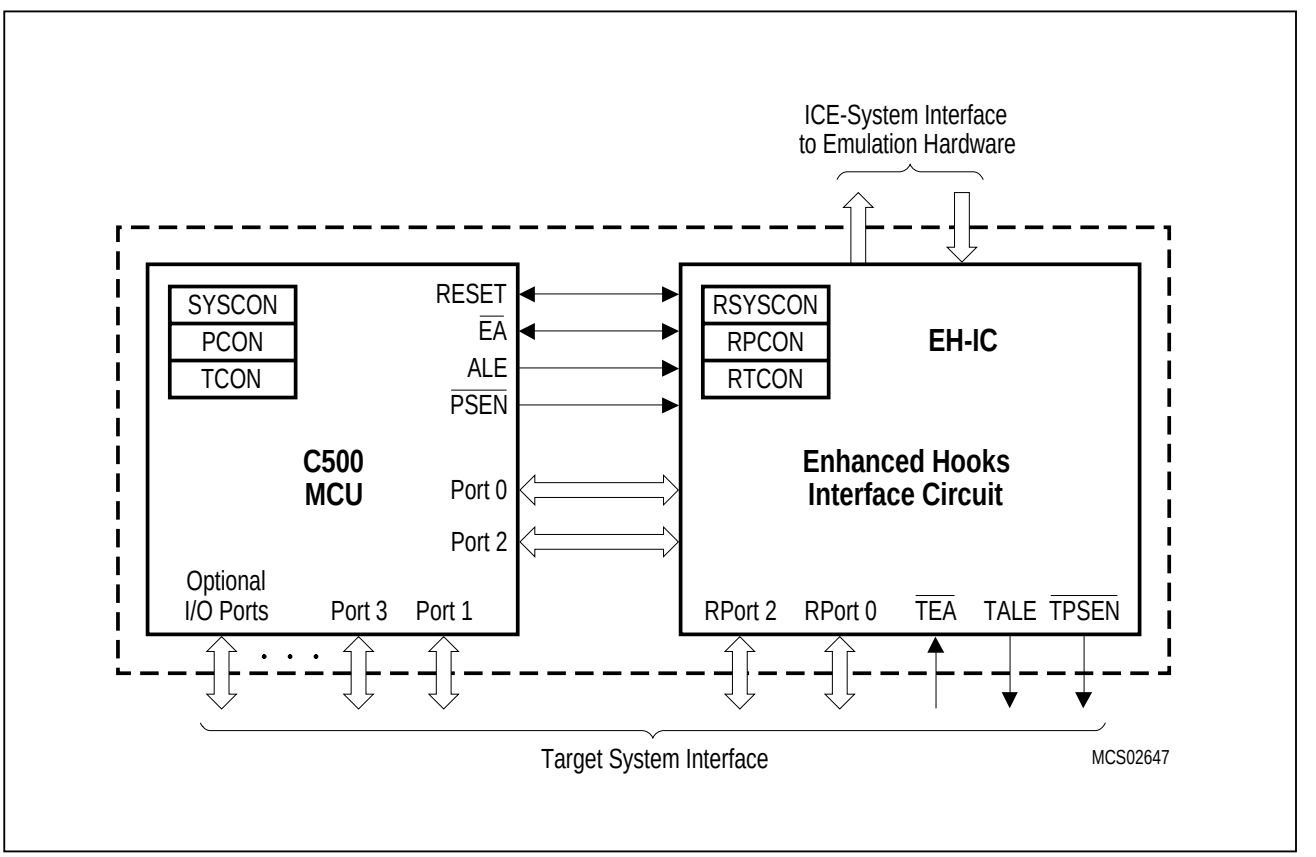


Figure 4-2
Basic C500 MCU Enhanced Hooks Concept Configuration

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 program 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 licenced to Siemens.

4.6 Eight Datapointers for Faster External Bus Access

4.6.1 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.

4.6.2 How the eight Datapointers of the C505 are realized

Simply adding more datapointers is not suitable because of the need to keep up 100% compatibility to the 8051 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 C505 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 C505 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
DPSEL.2-0	Data pointer select bits DPSEL.2-0 defines the number of the actual active data pointer.DPTR0-7.

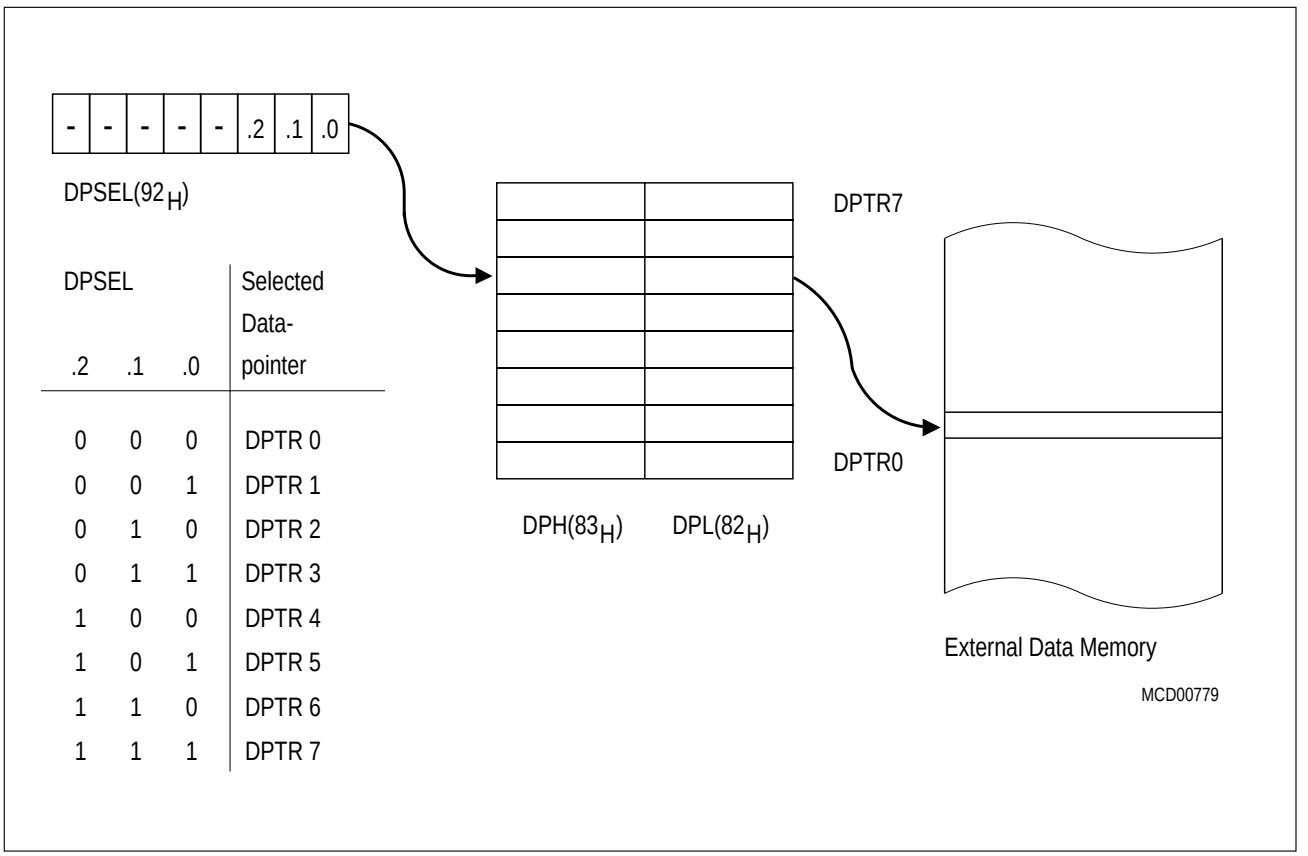


Figure 4-3
Accessing of External Data Memory via Multiple Datapointers

4.6.3 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.

4.6.4 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 a C505)

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 C505'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 a C505 program has up to 24 byte (16 variables and 8 stack bytes) of the internal RAM free for other use.

4.7 ROM Protection for the C505

The C505-2R 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 C505-2R version has to define whether ROM protection has to be selected or not.

The C505-2R 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 C505-2R 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.

4.7.1 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 (chapter 10).

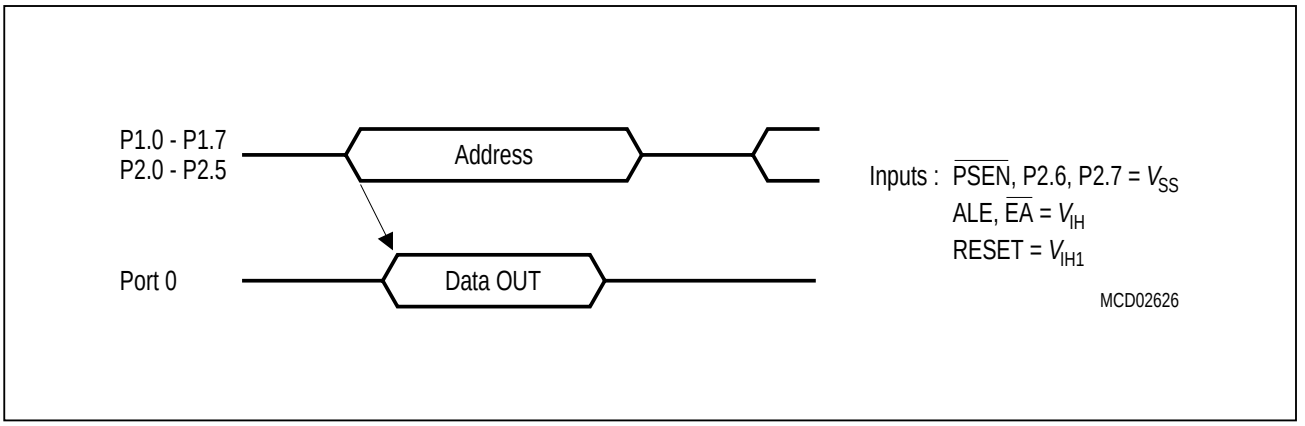


Figure 4-4
ROM Verification Mode 1

ROM verification mode 1 is selected if the inputs $\overline{\text{PSEN}}$, ALE, $\overline{\text{EA}}$, and 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 C505 must be provided with a system clock at the XTAL pins and pullup resistors on the port 0 lines.

4.7.2 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 (chapter 10).

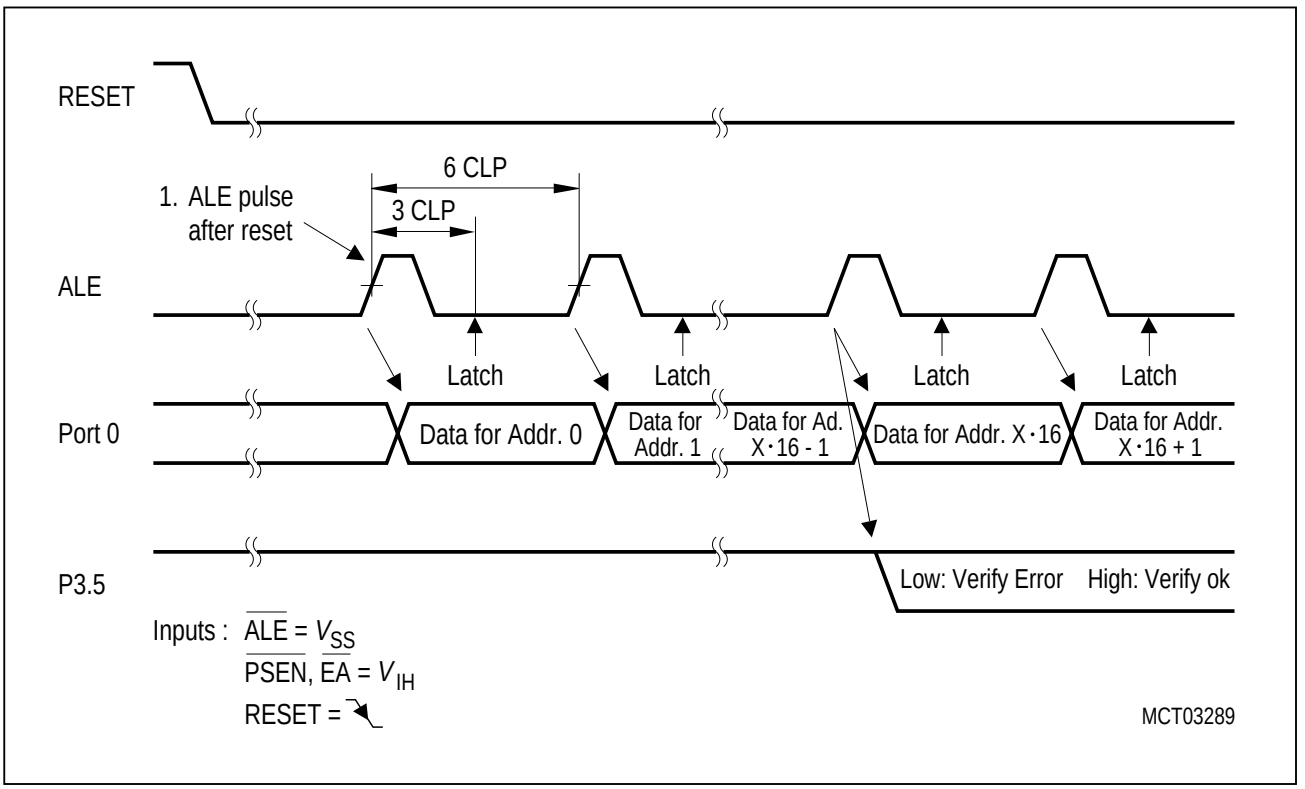


Figure 4-5
ROM Verification Mode 2

ROM verification mode 2 is selected if the inputs $\overline{PSEN}$, $\overline{EA}$, and ALE are put to the specified logic levels. With RESET going inactive, the ROM verification mode 2 sequence is started. The C505 outputs an ALE signal with a period of 3 CLP 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 C505-2R 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 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 time when the following 16 bytes are checked. In ROM verification mode 2, the C505 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 C505-2R in ROM verification mode 2. With RESET going inactive, the C505-2R starts the ROM verify sequence. Its ALE is clocking a 14-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 C505-2R 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.

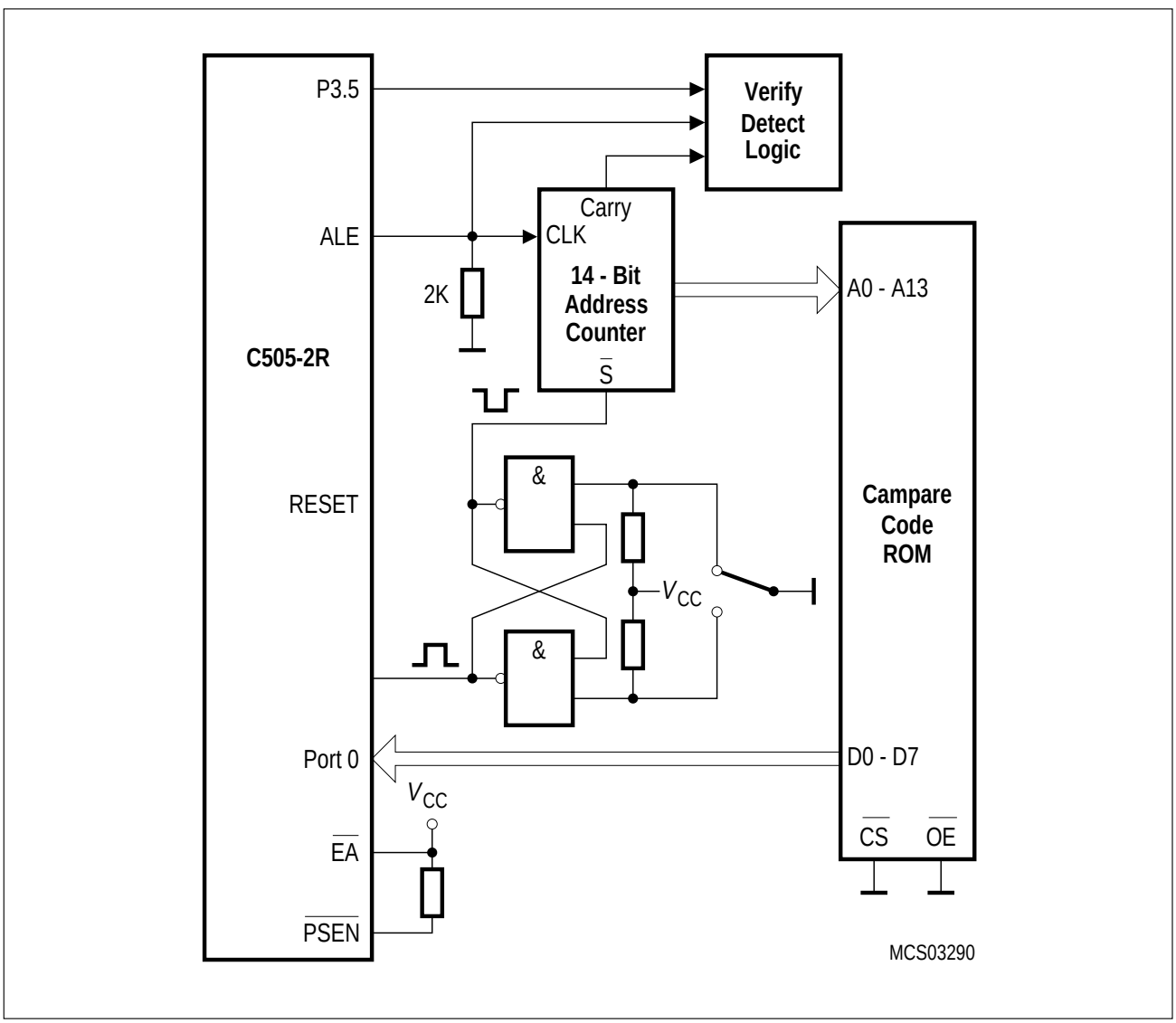


Figure 4-6
ROM Verification Mode 2 - External Circuitry Example

4.8 Version Registers

Version registers are typically used for adapting the programming firmware to specific device characteristics such as ROM / OTP size etc.

Three version registers are implemented in the C505. They can be read during normal program execution mode as mapped SFRs when the bit RMAP in SFR SYSCON is set.

The first step of the C505 will contain the following information in the version registers. Version register 2 will be incremented with each new step of the C505.

Contents of Version registers

Name	Address	C505-2R
Version Register 0	FC_H	$C5_H$
Version Register 1	FD_H	05_H
Version Register 2	FE_H	01_H

5 System Reset

5.1 Hardware Reset Operation

The hardware reset function incorporated in the C505 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 C505, 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 high input. An internal Schmitt trigger is used at the input for noise rejection. Since the reset is synchronized internally, the RESET pin must be held high for at least two machine cycles (12 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 RESET goes low again.

During reset, pins ALE and $\overline{\text{PSEN}}$ are configured as inputs and should not be stimulated 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 reset pin, a pulldown resistor is internally connected to V_{SS} to allow a power-up reset with an external capacitor only. An automatic power-up reset can be obtained, when V_{CC} is applied, by connecting the reset pin to V_{CC} via a capacitor. After V_{CC} 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.

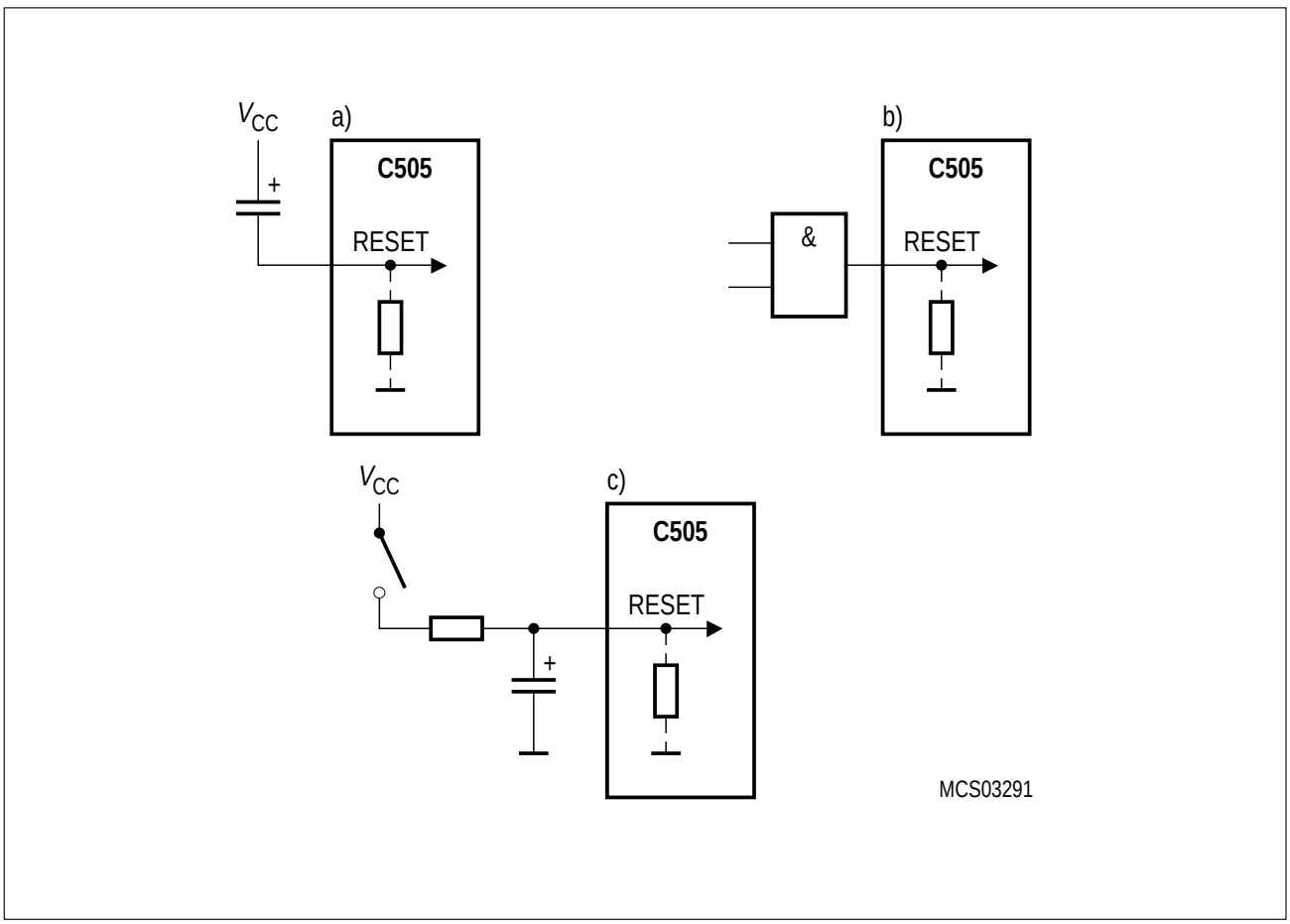


Figure 5-1
Reset Circuitries

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 4 default in 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 and 4) output a one (1). Port 2 lines output a zero (or one) after reset, if the $\overline{EA}$ is held low (or high). The internal SFRs are set to their initial states as defined in **table 3-2**.

The contents of the internal RAM and XRAM of the C505 are not affected by a reset. After power-up the contents are 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 C505 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 C505 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_{CC} rise time - e.g. 1µs, measured from V_{CC} = 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_{CC} below 4.25V (lower specification limit). Therefore, at slower V_{CC} 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 RESET pin.

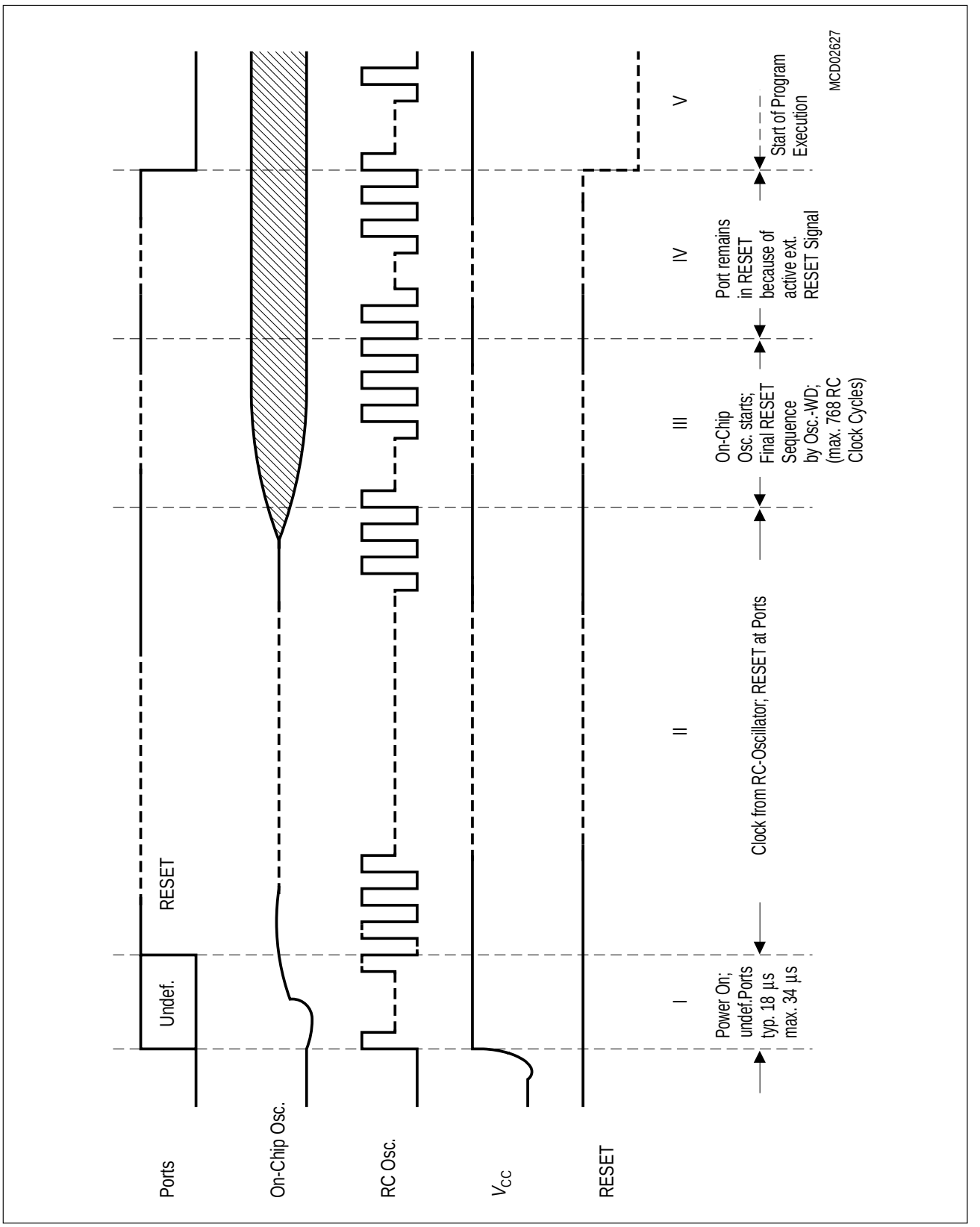


Figure 5-2
Power-On Reset of the C505

5.3 Hardware Reset Timing

This section describes the timing of the hardware reset signal.

The input pin 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 (high 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 RESET signal must be active for at least one machine cycle; after this time the C505 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 $\overline{EA}$ = 0 (external program memory). Thus, between the release of the 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.

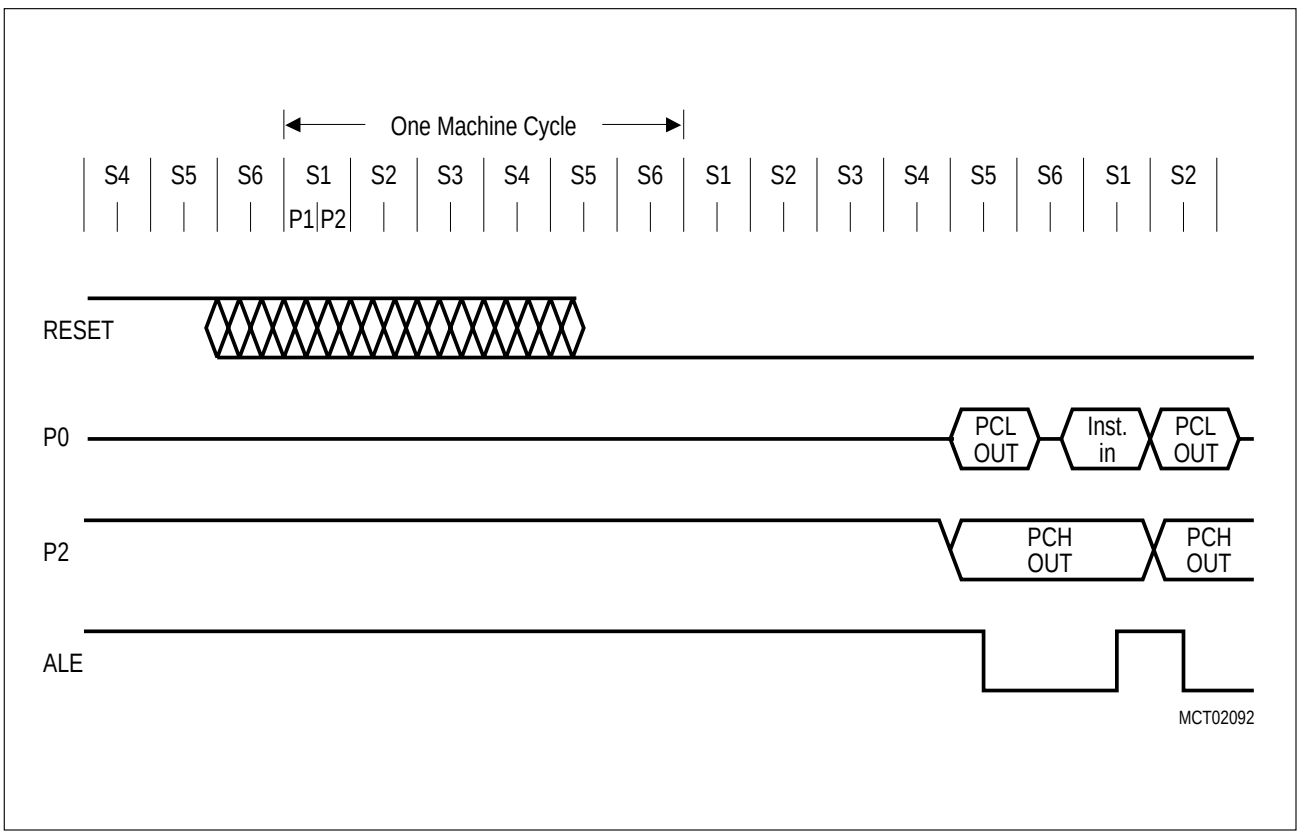


Figure 5-3
CPU Timing after Reset

5.4 Oscillator and Clock Circuit

XTAL1 and XTAL2 are the input and output 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.

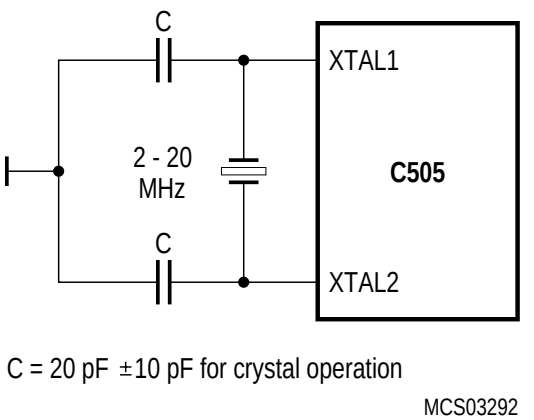


Figure 5-4
Recommended Oscillator Circuit

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.

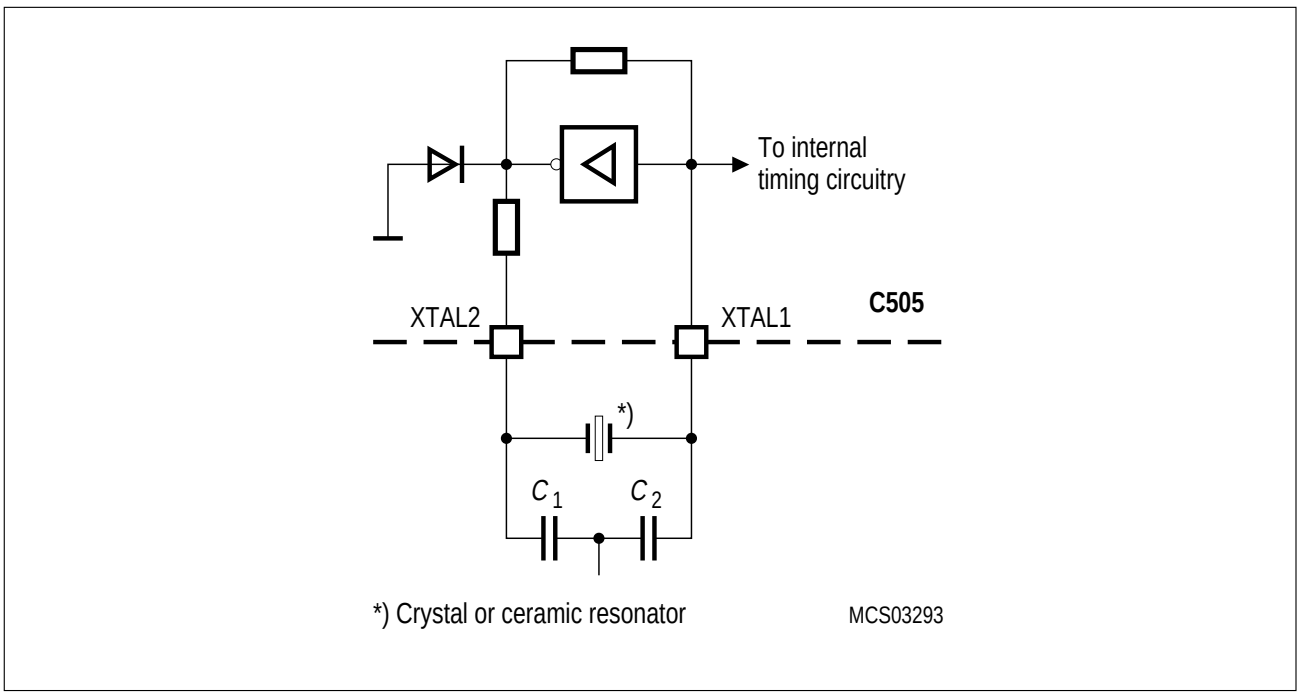


Figure 5-5
On-Chip Oscillator Circuiry

To drive the C505 with an external clock source, the external clock signal has to be applied to XTAL1, as shown in **figure 5-6**. XTAL2 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 XTAL1.

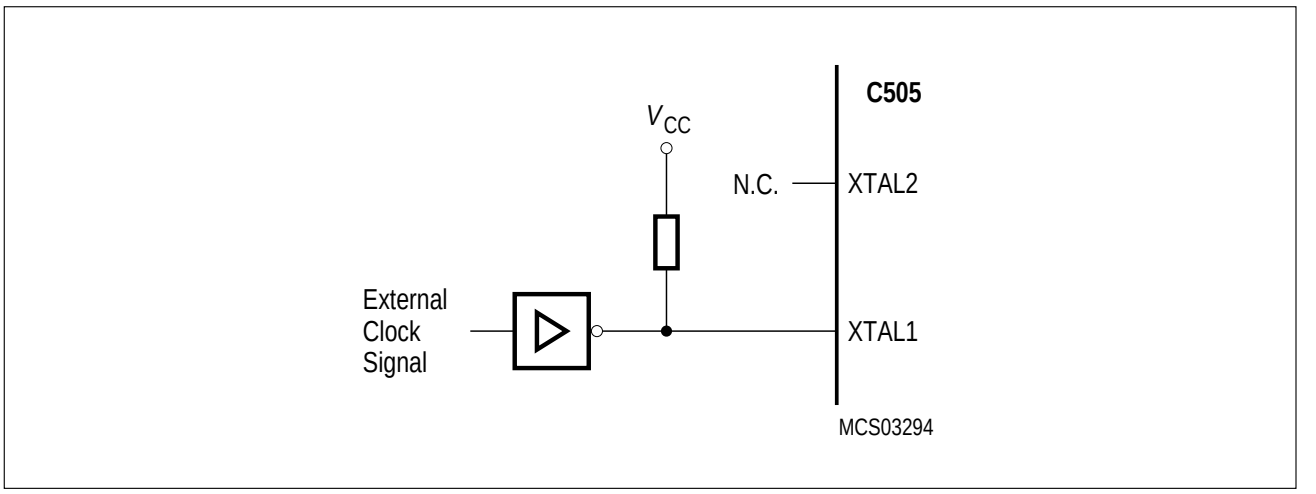


Figure 5-6
External Clock Source

5.5 System Clock Output

For peripheral devices requiring a system clock, the C505 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 1/6 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 : $00X000000_B$**

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

The shaded bits are not used for clock output control.

Bit	Function
CLK	Clockout enable bit When set, pin P1.6/CLKOUT outputs the system clock which is 1/6 of the oscillator frequency.
–	Reserved bits for future use. Read by CPU returns undefined values.

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 32.

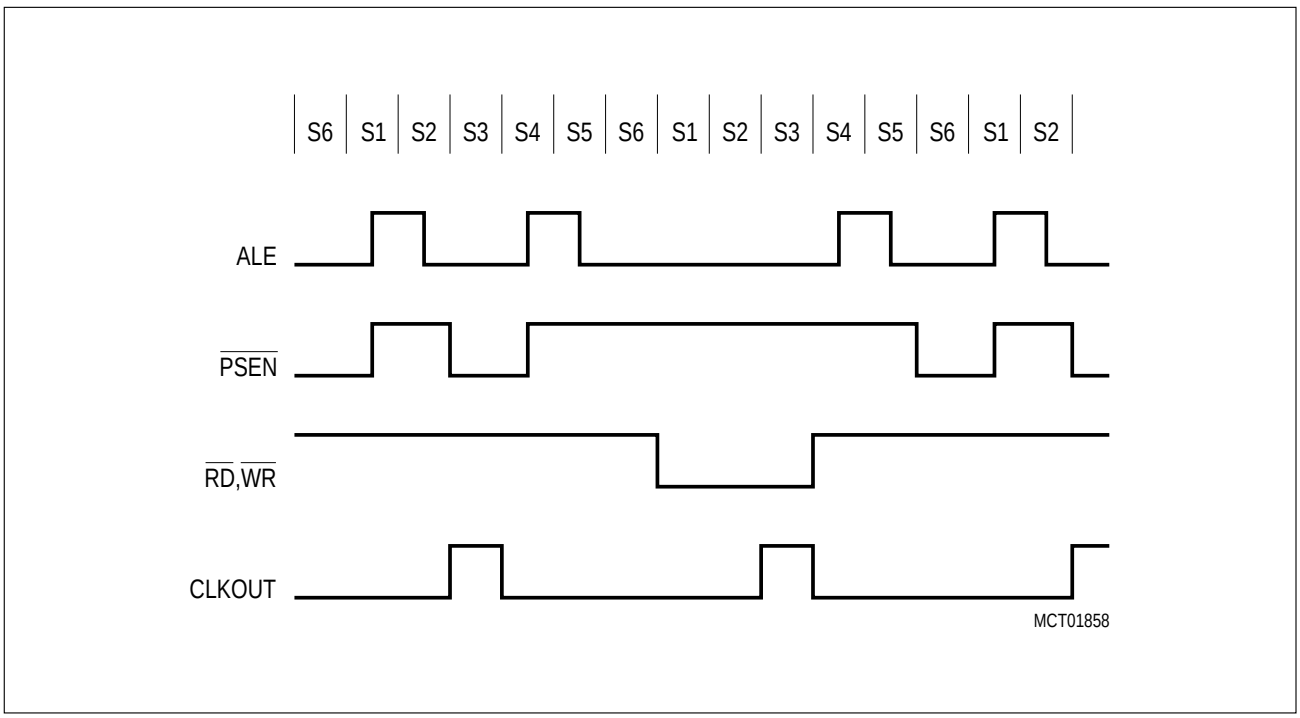


Figure 5-7
Timing Diagram - System Clock Output

6 On-Chip Peripheral Components

This chapter gives detailed information about all on-chip peripherals of the C505 except for the integrated interrupt controller, which is described separately in chapter 7.

6.1 Parallel I/O

The C505 has four 8-bit I/O ports and one 2-bit I/O port. Port 0 is an open-drain bidirectional I/O port, while ports 1 to 4 are quasi-bidirectional I/O ports with internal pullup resistors. That means, when configured as inputs, ports 1 to 4 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 .

Port 4 is 2-bit I/O port with CAN controller specific alternate functions. This port has no available bits at bit positions 2-7.

6.1.1 Port Structures

The C505 generally allows digital I/O on 34 lines grouped into 4 bidirectional 8-bit ports and one 2-bit port. Each port bit consists of a latch, an output driver and an input buffer. Read and write accesses to the I/O ports P0-P4 are performed via their corresponding special function registers. Depending on the specific ports, multiple functions are assigned to the port pins. Therefore, the parallel I/O ports of the C505 can be grouped into three different types which are listed in **table 6-1**.

Table 6-1
C505 Port Structure Types

Type	Description
A	Standard digital I/O ports which can also be used for external address/data bus.
B	Standard multifunctional digital I/O port lines
C	Mixed digital/analog I/O port lines with programmable analog input function

Type A and B port pins are standard C501 compatible I/O port lines, which can be used for digital I/O. The type A ports (port 0 and port 2) are also designed for accessing external data or program memory. Type B port lines are located at port 3 and port 4 to provide alternate functions for the serial interface and CAN controller I/O lines respectively, or are used as control outputs during external data memory accesses.

The C505 provides eight analog input lines which are realized as mixed digital/analog inputs (type C). The 8 analog inputs, AN0-AN7, are located at the port 1 pins P1.0 to P1.7. After reset, all analog inputs are disabled and the related pins of port 1 are configured as digital inputs. The analog function of the specific port 1 pins are enabled by bits in the SFRs P1ANA. Writing a 0 to a bit position of P1ANA assigns the corresponding pin to operate as analog input.

Note : P1ANA is a mapped SFR and can be only accessed if bit RMAP in SFR SYSCON is set.

As already mentioned, port 1, 3 and 4 are provided for multiple alternate functions. These functions are listed in **table 6-2**:

Table 6-2
Alternate Functions of Port 1, 3 and 4

Port	Second / third Function	Port Type	Function
P1.0	AN0 / $\overline{\text{INT3}}$ / CC0	C	Analog input channel 0 / External Interrupt 3 input / Capture/compare 0 input/output
P1.1	AN1 / INT4 / CC1	C	Analog input channel 1 / External Interrupt 4 input / Capture/compare 1 input/output
P1.2	AN2 / INT5 / CC2	C	Analog input channel 2 / External Interrupt 5 input / Capture/compare 2 input/output
P1.3	AN3 / INT6 / CC3	C	Analog input channel 3 / External Interrupt 6 input / Capture/compare 3 input/output
P1.4	AN4	C	Analog input channel 4
P1.5	AN5 / T2EX	C	Analog input channel 5 / Timer 2 external reload/trigger input
P1.6	AN6 / CLKOUT	C	Analog input channel 6 / System clock output
P1.7	AN7 / T2	C	Analog input channel 7 / Timer 2 external count input
P3.0	RxD	B	Serial port's receiver data input (asynchronous) or data input/ output (synchronous)
P3.1	TxD	B	Serial port's transmitter data output (asynchronous) or data clock output (synchronous)
P3.2	$\overline{\text{INT0}}$	B	External interrupt 0 input, timer 0 gate control
P3.3	$\overline{\text{INT1}}$	B	External interrupt 1 input, timer 1 gate control
P3.4	T0	B	Timer 0 external counter input
P3.5	T1	B	Timer 1 external counter input
P3.6	$\overline{\text{WR}}$	B	External data memory write strobe
P3.7	$\overline{\text{RD}}$	B	External data memory read strobe
P4.0	TXDC	B	CAN controller transmit output (C505C only)
P4.1	RXDC	B	CAN controller receive input (C505C only)

Prior to the description of the port type, specific port configurations the general port structure is described in the next section.

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 five 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 P4) activate the "read-latch" signal, while others activate the "read-pin" signal.

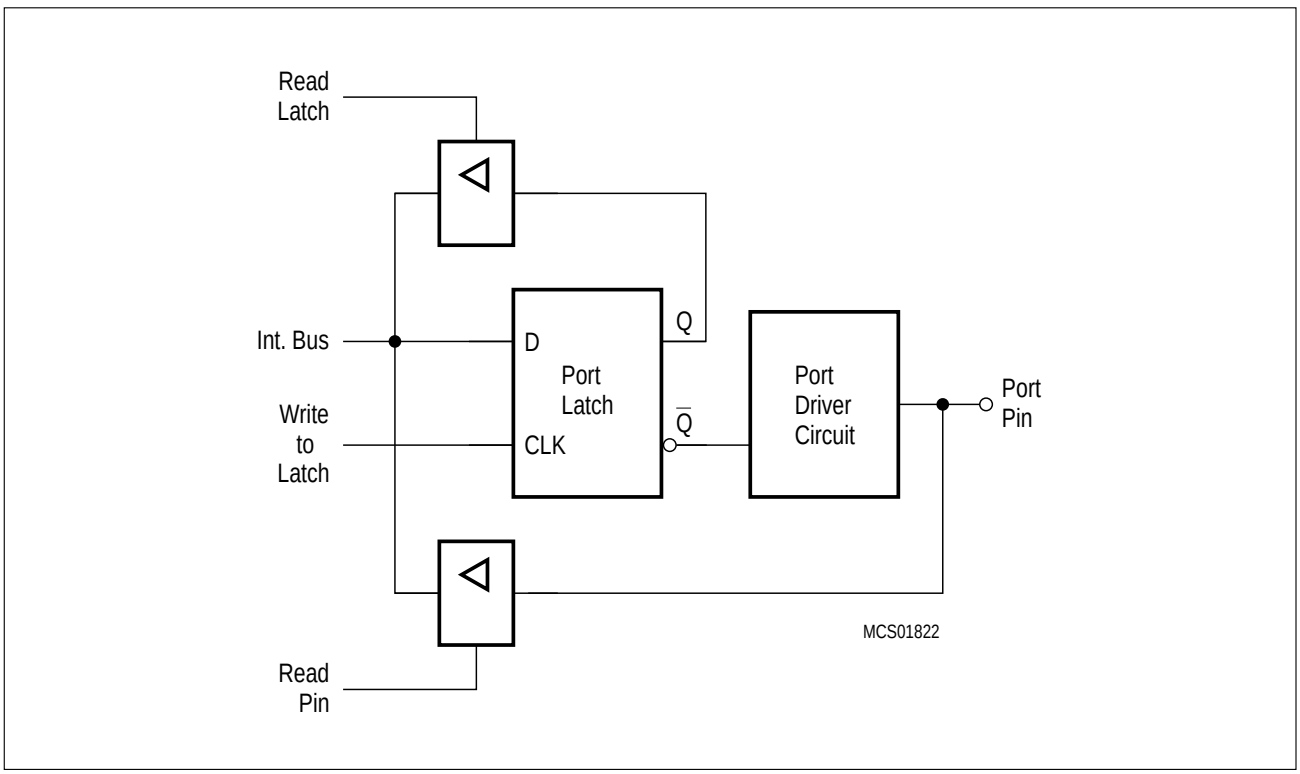


Figure 6-1
Basic Structure of a Port Circuitry

The output drivers of Port 1 to 4 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 4 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".

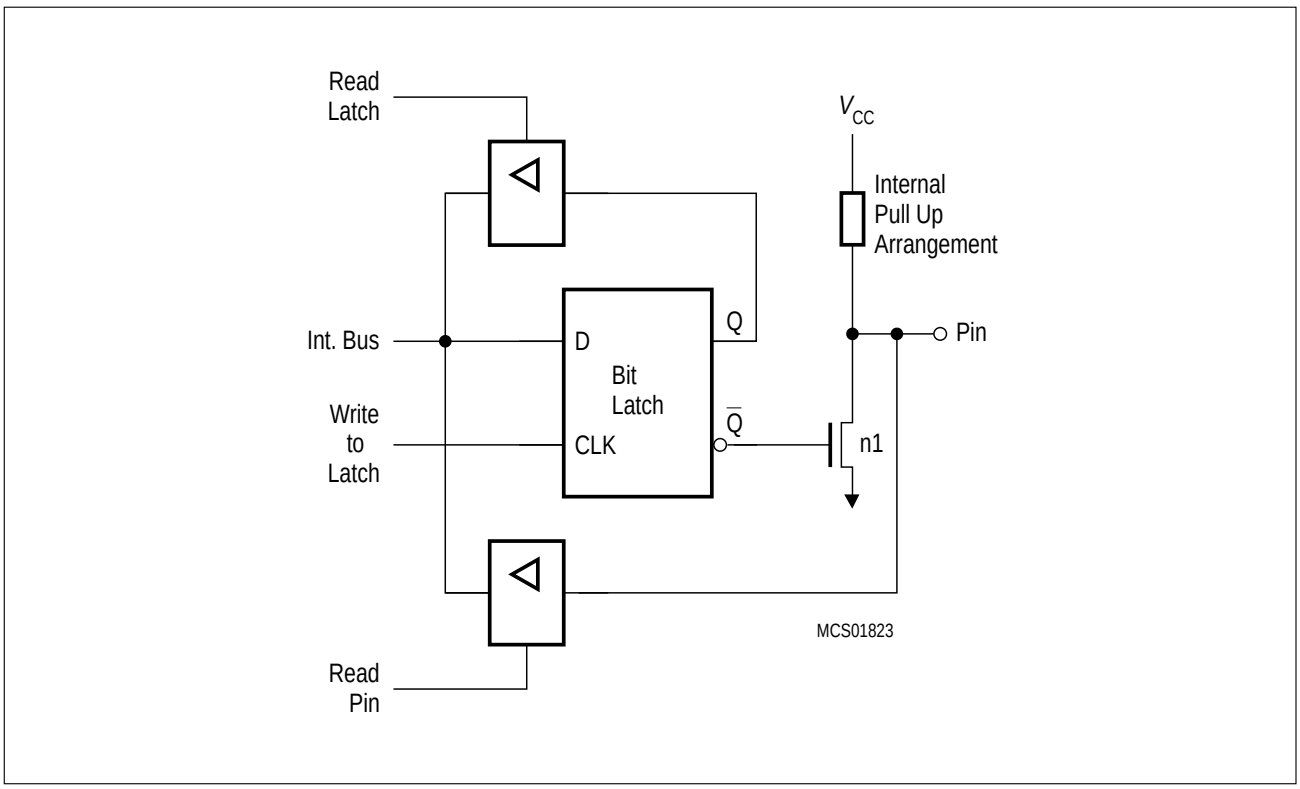


Figure 6-2
Basic Output Driver Circuit of Ports 1 to 4

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.

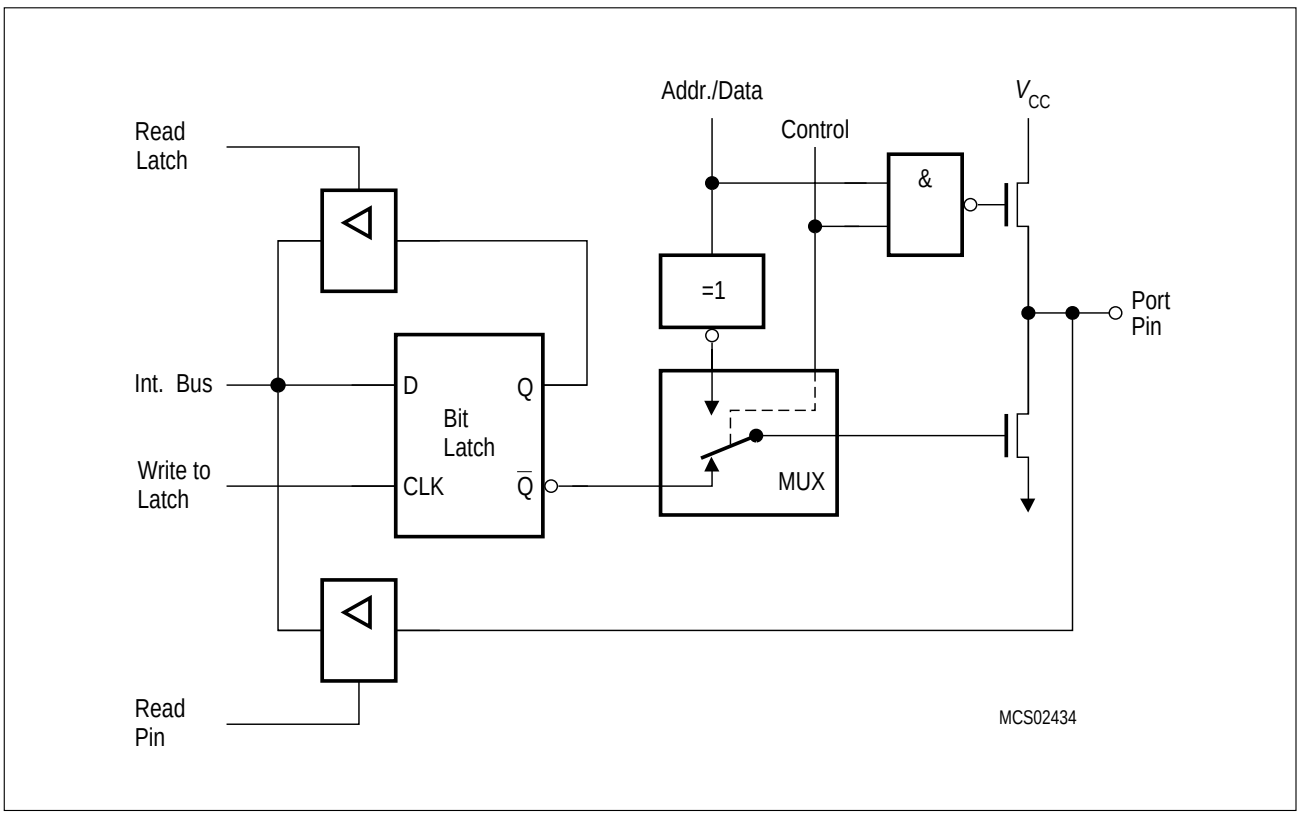


Figure 6-3
Port 0 Circuitry

6.1.2.2 Port 1, Port 3 and Port 4 Circuitry

The pins of ports 1, 3 and 4 are multifunctional. They are port pins and also serve to implement special features as listed in **table 6-2**.

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).

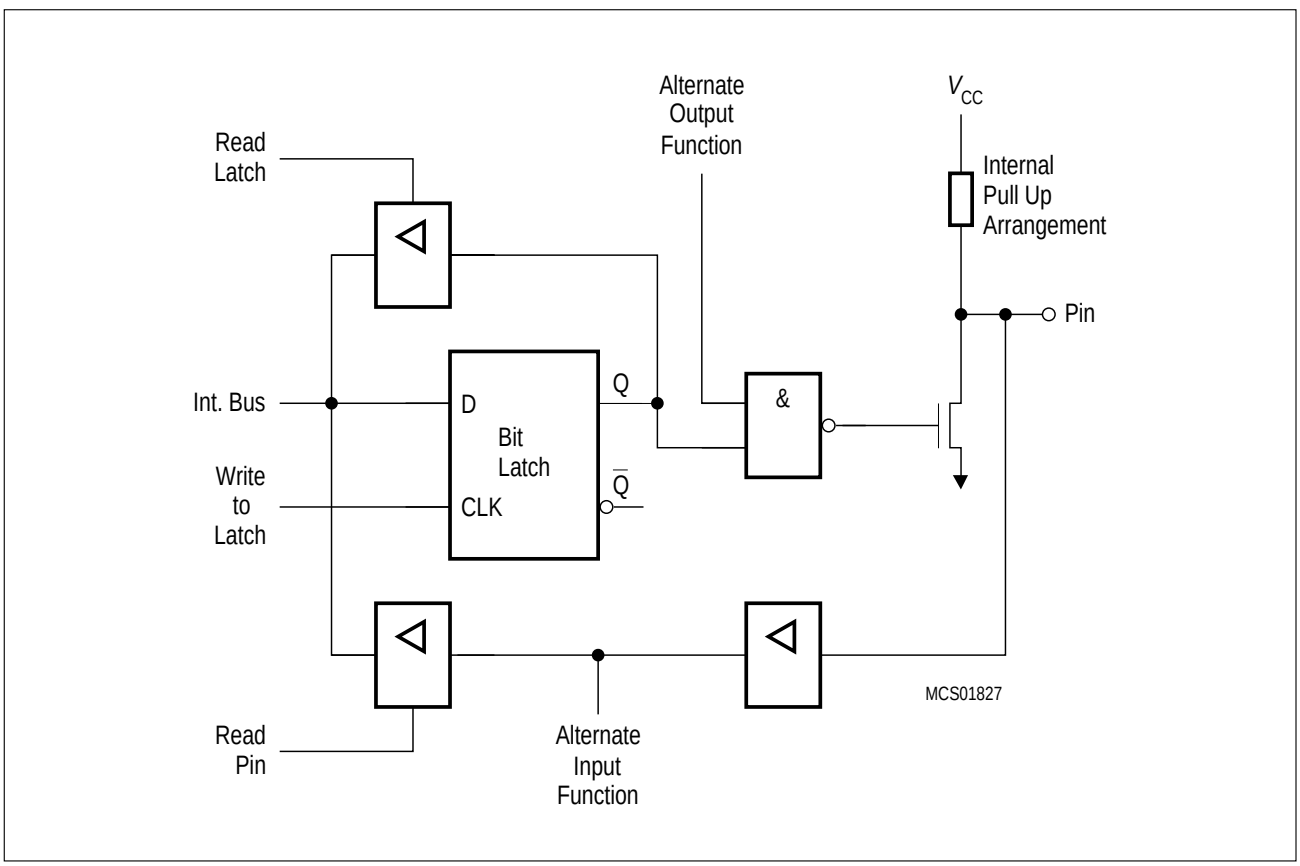


Figure 6-4
Ports 1, 3 and 4

The alternate functions of Port 4 pins are available for the C505C only.

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 6-5a**) to emit 1's for the entire external memory cycle instead of the weak ones (p2 and p3) used during normal port activity.

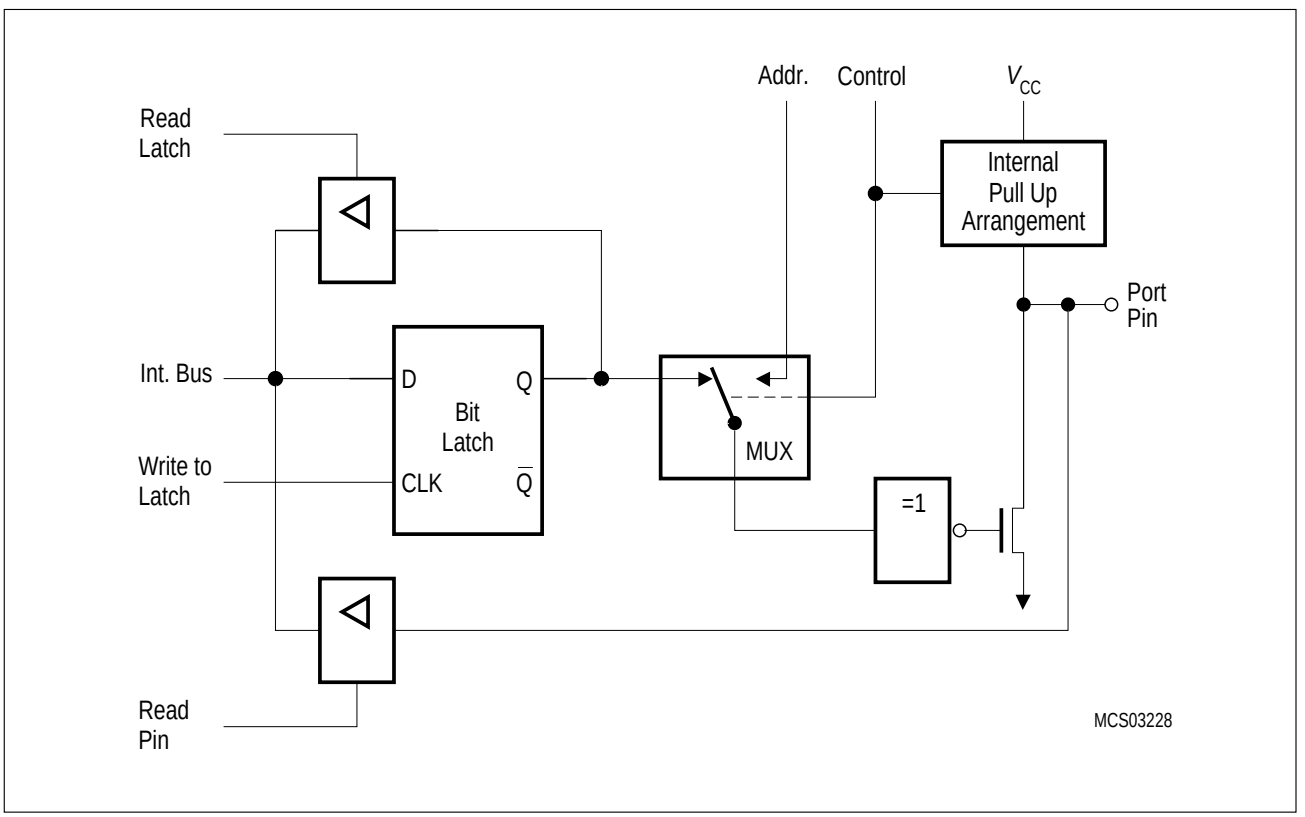


Figure 6-5
Port 2 Circuitry

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

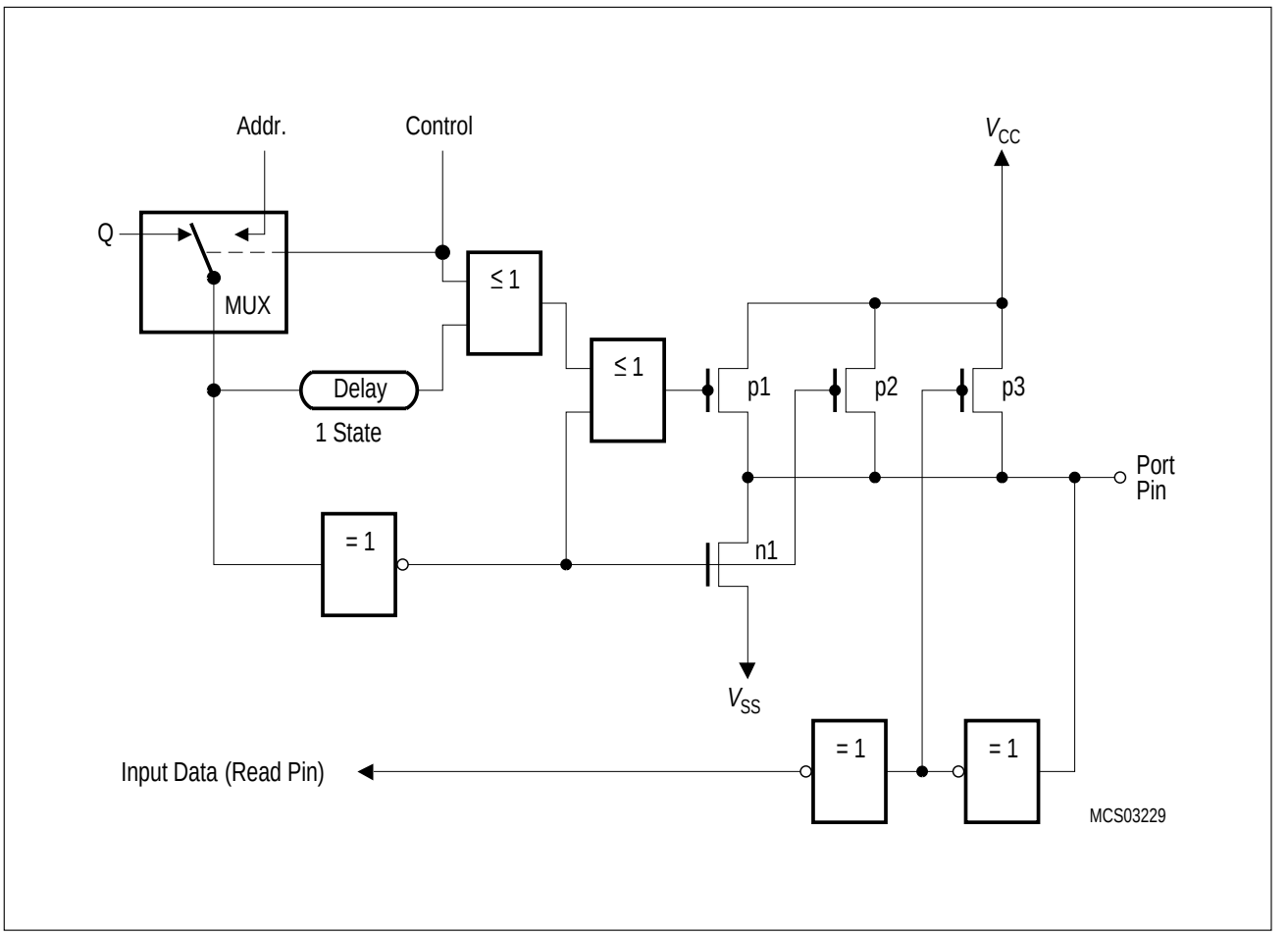


Figure 6-5a
Port 2 Pull-up Arrangement

Port 2 in I/O function works similar to the Type B port driver circuitry (section 6.1.3.1) whereas in address output function it works similar to Port 0 circuitry.

6.1.3 Detailed Output Driver Circuitry

In fact, the pullups mentioned before and included in **figure 6-2, 6-4** and **6-5** are pullup arrangements. The differences of the port types available in the C505 are described in the next sections.

6.1.3.1 Type B Port Driver Circuitry

Figure 6-6 shows the output driver circuit of the type B multifunctional digital I/O port lines. The basic circuitry of these ports is shown in **figure 6-4**. The pullup arrangement of type B port lines has one n-channel pulldown FET and three pullup FETs:

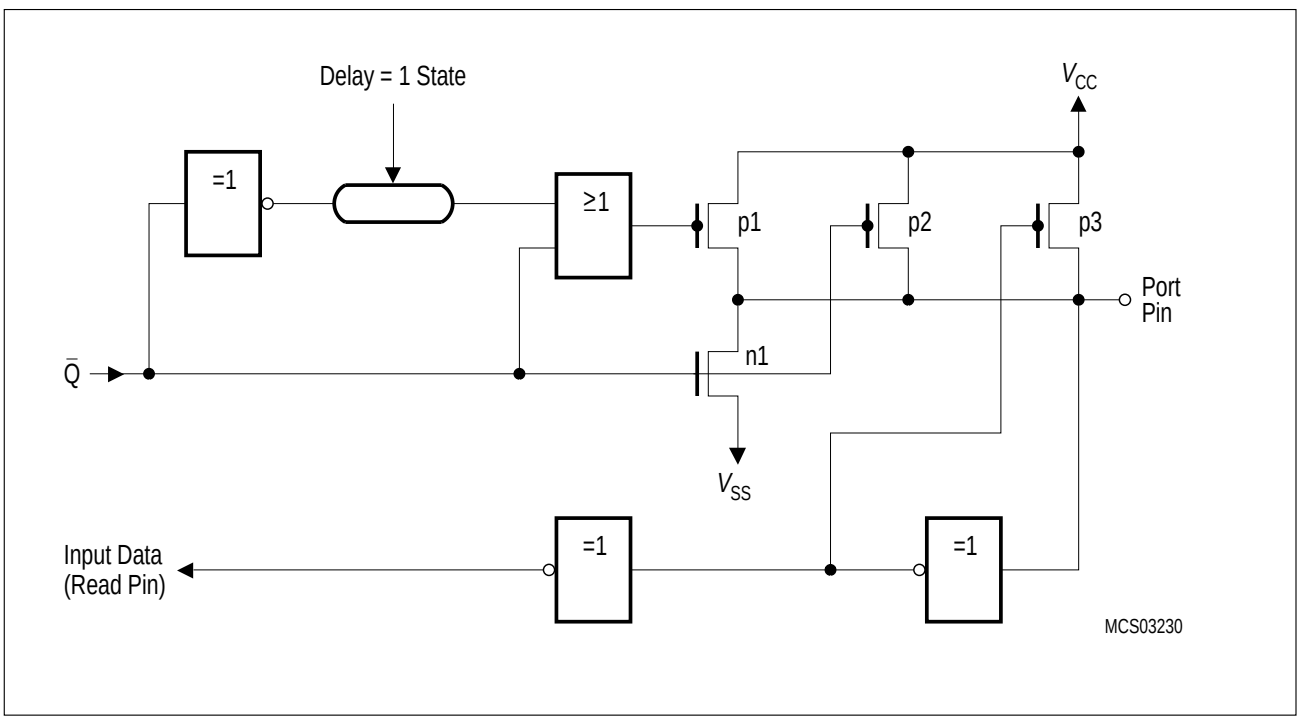


Figure 6-6
Driver Circuit of Type B Port Pins

- 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_{CC} 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.

6.1.3.2 Type C Port Driver Circuitry

Figure 6-7 shows the port driver circuit of the type C mixed digital/analog I/O port 1 lines of the C505. The analog function is selected by the bits in the SFR P1ANA. When analog function is selected, all output driver transistors (p1, p2, p3 and n1) are switched off.

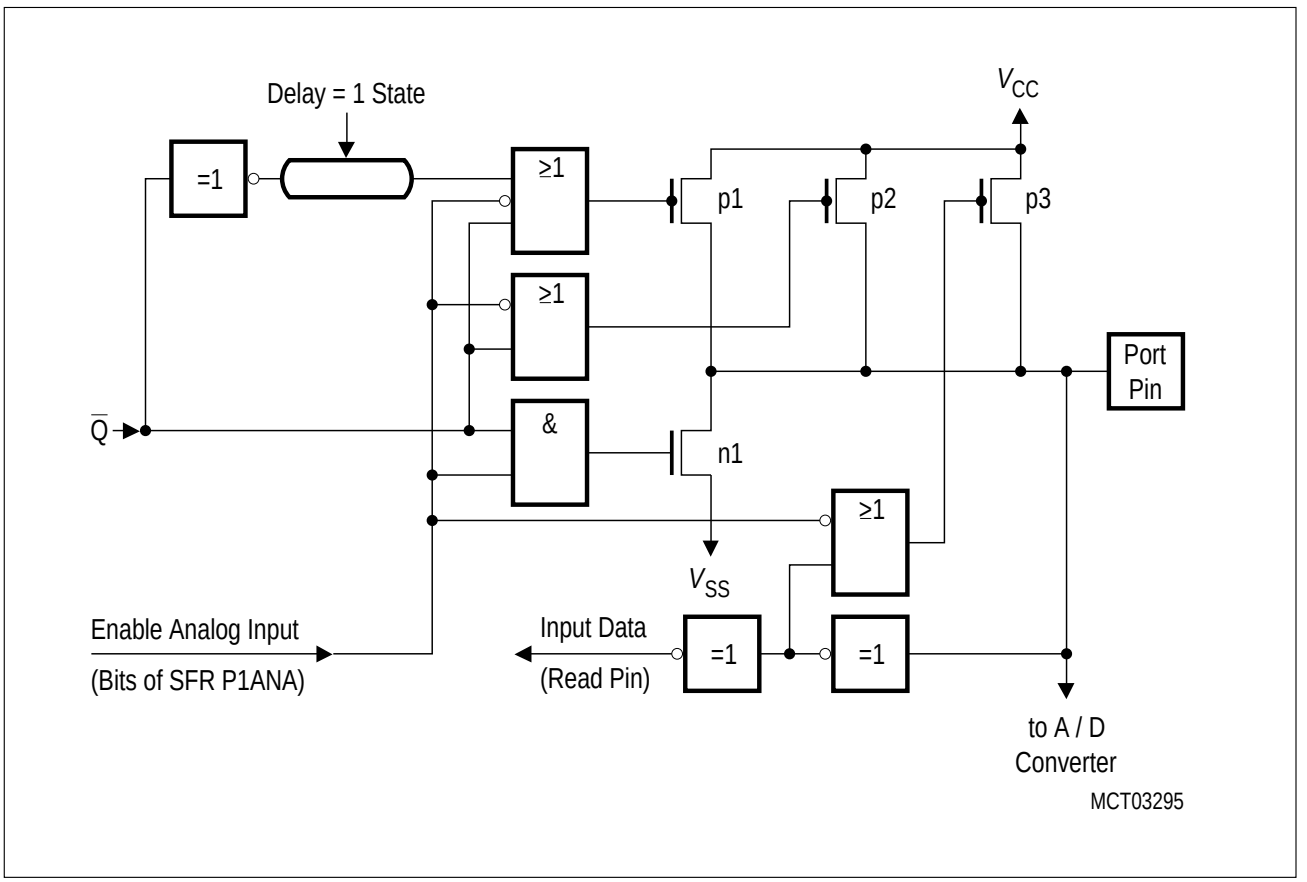


Figure 6-7
Driver Circuit of Type C Port Pins

6.1.4 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.

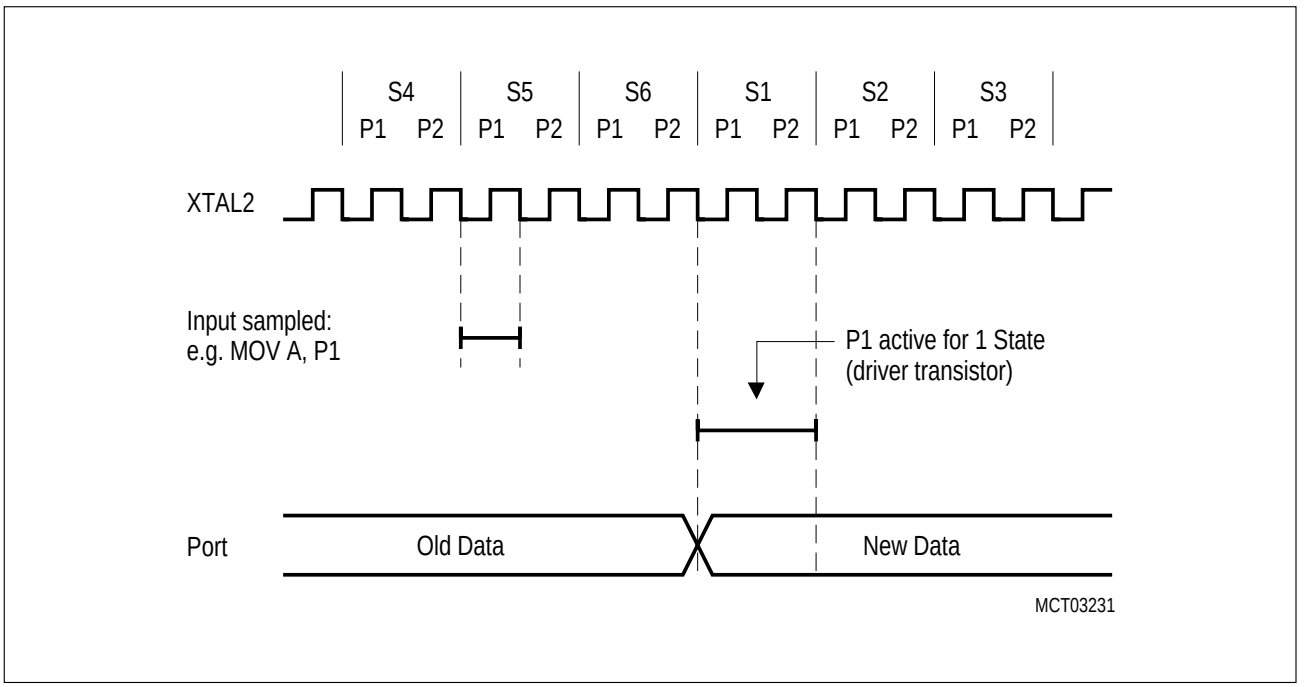


Figure 6-8
Port Timing

6.1.5 Port Loading and Interfacing

The output buffers of ports 1 to 4 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 C505 or in chapter 10 of this User's Manual. 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 4 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.

6.1.6 Read-Modify-Write Feature of Ports 0 to 4

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 6-3**. 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 6-3** 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 6-3
"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 rather than the pin will return the correct value of "1".

6.2 Timers/Counters

The C505 contains three 16-bit timers/counters, timer 0, 1, and 2, 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 6 oscillator periods, the counter rate is 1/6 of the oscillator frequency.

In "counter" function, the timer register is incremented in response to a 1-to-0 transition (falling edge) at its corresponding external input pin, T0, T1, or T2 (alternate functions of P3.4, P3.5 and P1.7 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 (12 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is 1/12 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 C505 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.

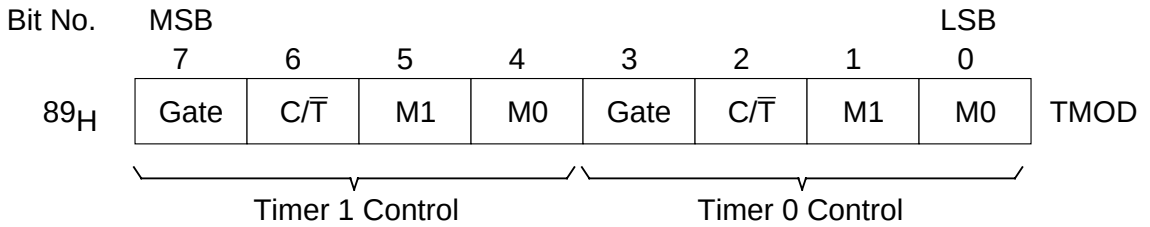
Special Function Register TCON (Address 88_H) **Reset Value : 00_H**

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 for 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).

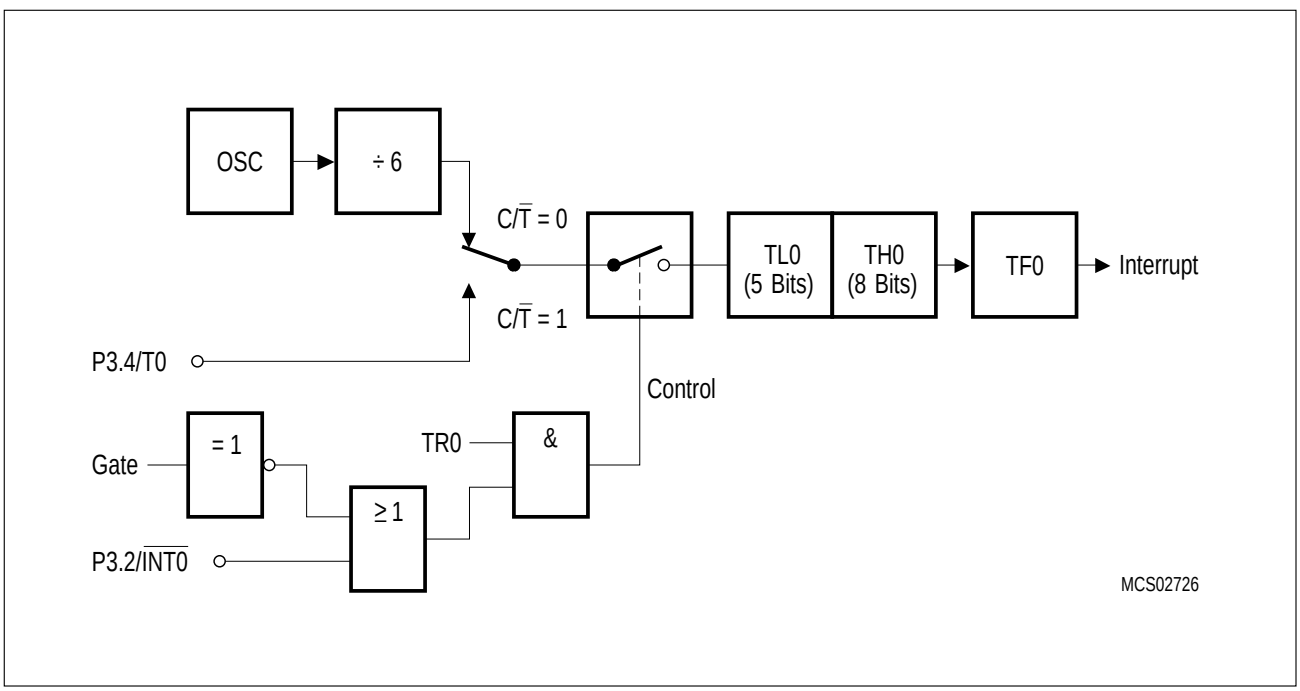


Figure 6-9
Timer/Counter 0, Mode 0: 13-Bit Timer/Counter

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**.

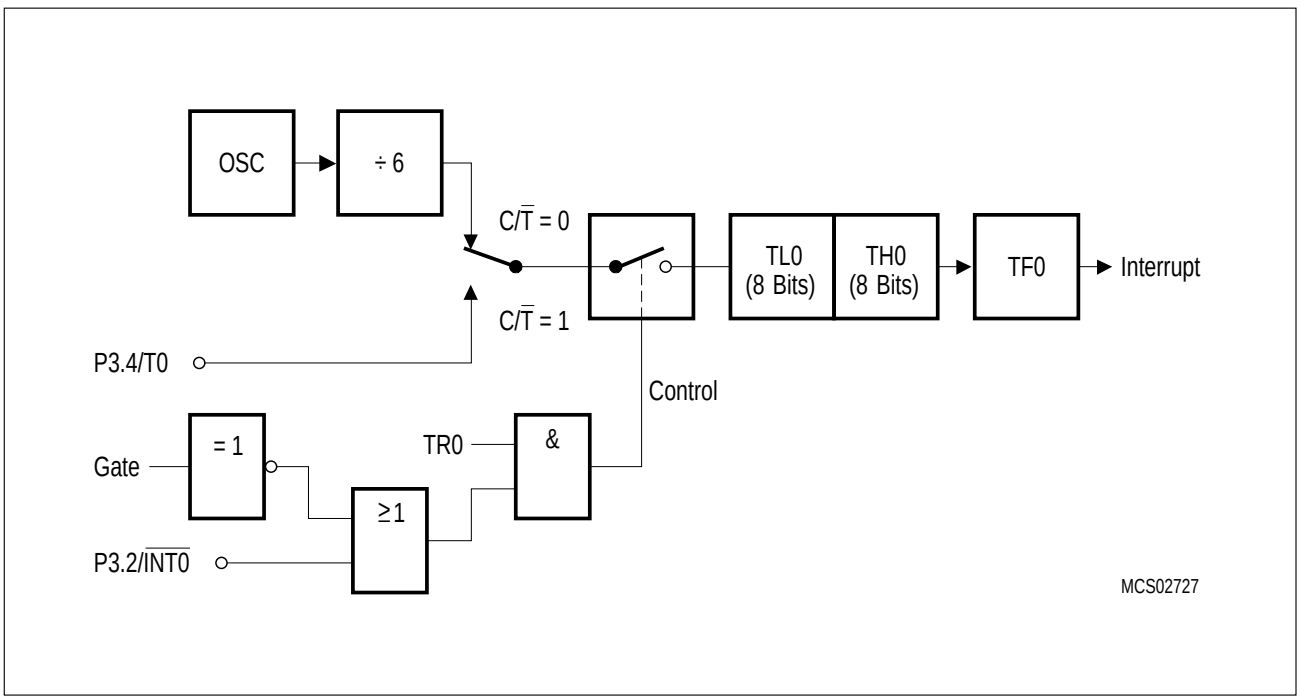


Figure 6-10
Timer/Counter 0, Mode 1: 16-Bit Timer/Counter

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.

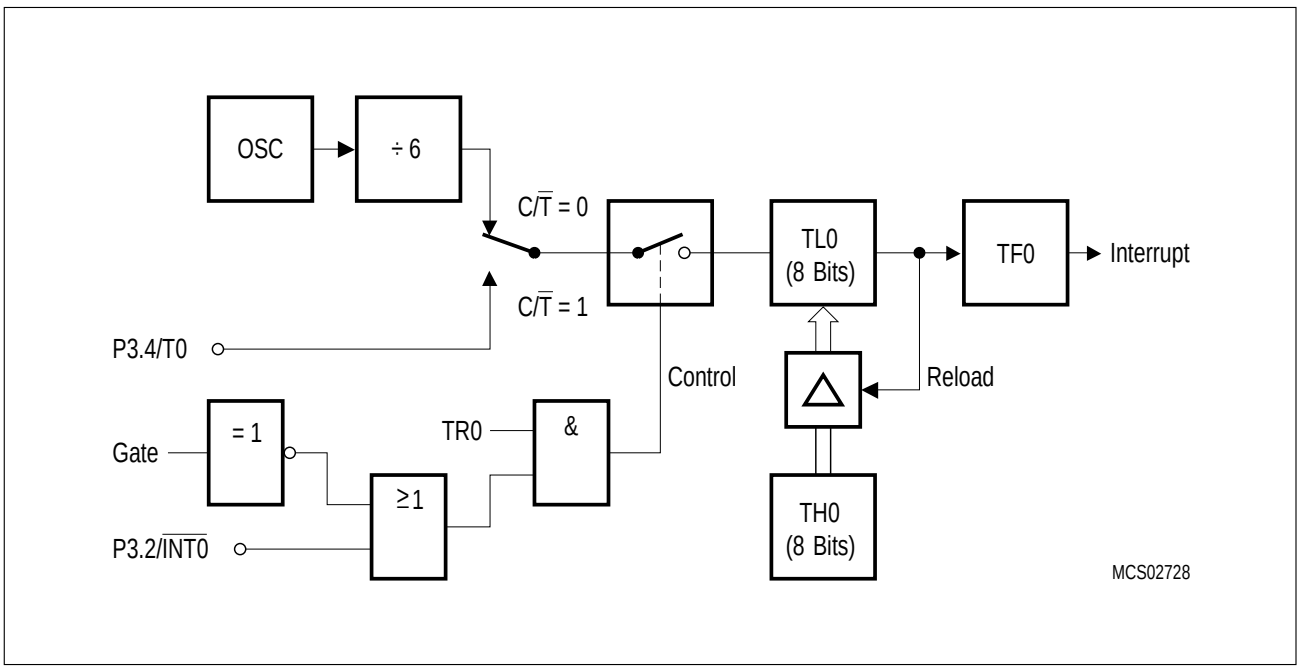


Figure 6-11
Timer/Counter 0,1, Mode 2: 8-Bit Timer/Counter with Auto-Reload

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.

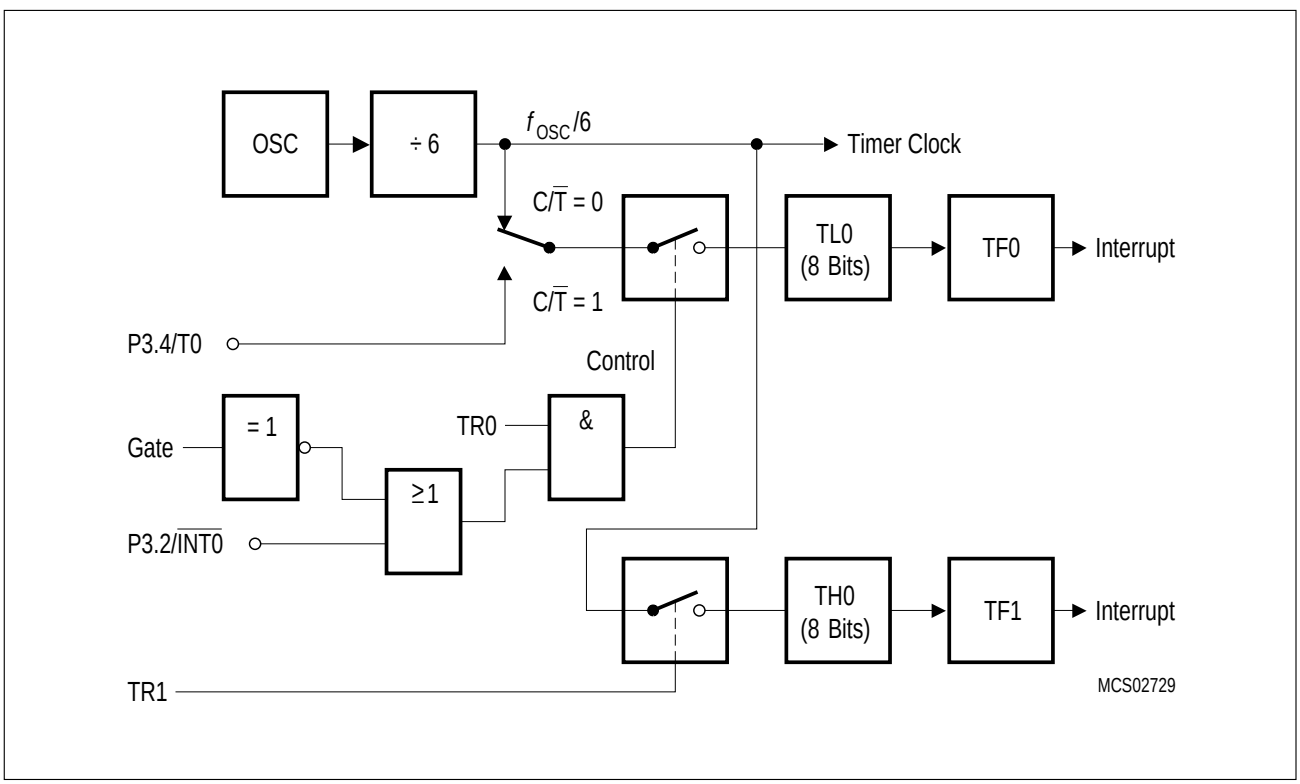


Figure 6-12
Timer/Counter 0, Mode 3: Two 8-Bit Timers/Counters

6.2.2 Timer/Counter 2 with Additional Compare/Capture/Reload

The timer 2 with additional compare/capture/reload features is one of the most powerful peripheral units of the C505. It can be used for all kinds of digital signal generation and event capturing like pulse generation, pulse width modulation, pulse width measuring etc.

Timer 2 is designed to support various automotive control applications as well as industrial applications (frequency generation, digital-to-analog conversion, process control ...). Please note that the functionality of this timer is not equivalent to timer 2 of the C501.

The C505 timer 2 in combination with the compare/capture/reload registers allows the following operating modes:

- Compare : up to 4 PWM output signals with 65535 steps at maximum, and 300 ns resolution
- Capture : up to 4 high speed capture inputs with 300 ns resolution
- Reload : modulation of timer 2 cycle time

The block diagram in **figure 6-13** shows the general configuration of timer 2 with the additional compare/capture/reload registers. The I/O pins which can be used for timer 2 control are located as multifunctional port functions at port 1 (see **table 6-4**).

Table 6-4
Alternate Port Functions of Timer 2

Pin Symbol	Function
P1.0 / AN0 / $\overline{\text{INT3}}$ / CC0	Compare output / capture input for CRC register
P1.1 / AN1 / INT4 / CC1	Compare output / capture input for CC register 1
P1.2 / AN2 / INT5 / CC2	Compare output / capture input for CC register 2
P1.3 / AN3 / INT6 / CC3	Compare output / capture input for CC register 3
P1.5 / AN5 / T2EX	External reload trigger input
P1.7 / AN7 / T2	External count or gate input to timer 2

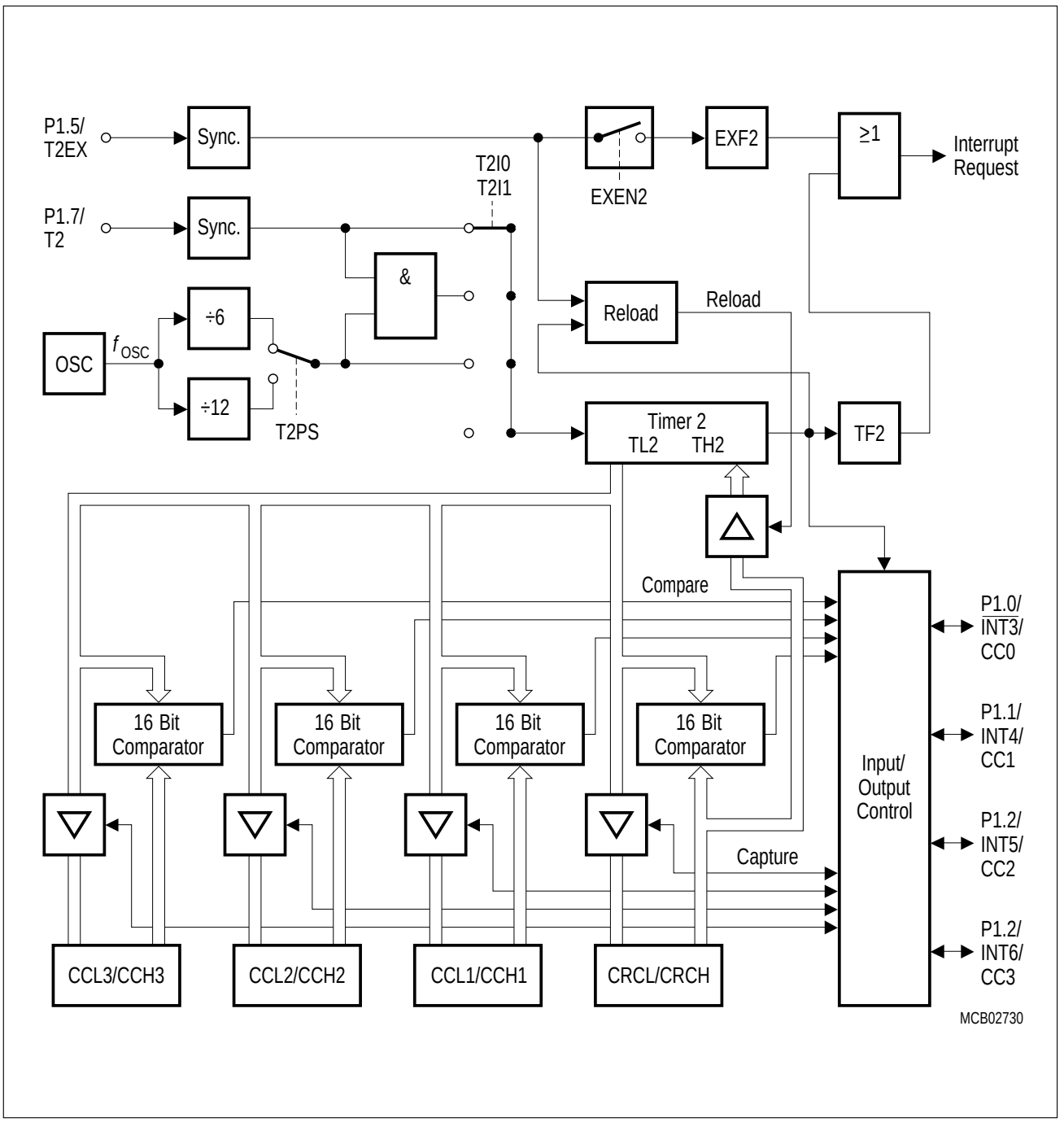


Figure 6-13
Timer 2 Block Diagram

6.2.2.1 Timer 2 Registers

This chapter describes all timer 2 related special function registers of timer 2. The interrupt related SFRs are also included in this section. **Table 6-5** summarizes all timer 2 SFRs.

Table 6-5
Special Function Registers of the Timer 2 Unit

Symbol	Description	Address
T2CON	Timer 2 control register	$C8_H$
TL2	Timer 2, low byte	CC_H
TH2	Timer 2, high byte	CD_H
CCEN	Compare / capture enable register	$C1_H$
CRCL	Compare / reload / capture register, low byte	CA_H
CRCH	Compare / reload / capture register, high byte	CB_H
CCL1	Compare / capture register 1, low byte	$C2_H$
CCH1	Compare / capture register 1, high byte	$C3_H$
CCL2	Compare / capture register 2, low byte	$C4_H$
CCH2	Compare / capture register 2, high byte	$C5_H$
CCL3	Compare / capture register 3, low byte	$C6_H$
CCH3	Compare / capture register 3, high byte	$C7_H$
IEN0	Interrupt enable register 0	$A8_H$
IEN1	Interrupt enable register 1	$B8_H$
IRCON	Interrupt control register	$C0_H$

The T2CON timer 2 control register is a bit-addressable register which controls the timer 2 function and the compare mode of registers CRC, CC1 to CC3.

Special Function Register T2CON (Address $C8_H$) **Reset Value : $00X00000_B$**

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
	CF_H	CE_H	CD_H	CC_H	CB_H	CA_H	$C9_H$	$C8_H$	
$C8_H$	T2PS	I3FR	–	T2R1	T2R0	T2CM	T2I1	T2I0	T2CON

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

Bit	Function
T2PS	Prescaler select bit When set, timer 2 is clocked in the “timer“ or “gated timer“ function with 1/12 of the oscillator frequency. When cleared, timer 2 is clocked with 1/6 of the oscillator frequency. T2PS must be 0 for the counter operation of timer 2.
I3FR	External interrupt 3 falling / rising edge flag Used for capture function in combination with register CRC. If set, a capture to register CRC (if enabled) will occur on a positive transition at pin P1.0/AN0/$\overline{INT3}$/ CC0
T2R1 T2R0	Timer 2 reload mode selection

T2R1	T2R0	Function
0	X	Reload disabled
1	0	Mode 0 : auto-reload upon timer 2 overflow (TF2)
1	1	Mode 1 : reload on falling edge at pin P1.5 / AN5 / T2EX

T2CM	Compare mode bit for registers CRC, CC1 through CC3 When set, compare mode 1 is selected. T2CM = 0 selects compare mode 0.
T2I1 T2I0	Timer 2 input selection

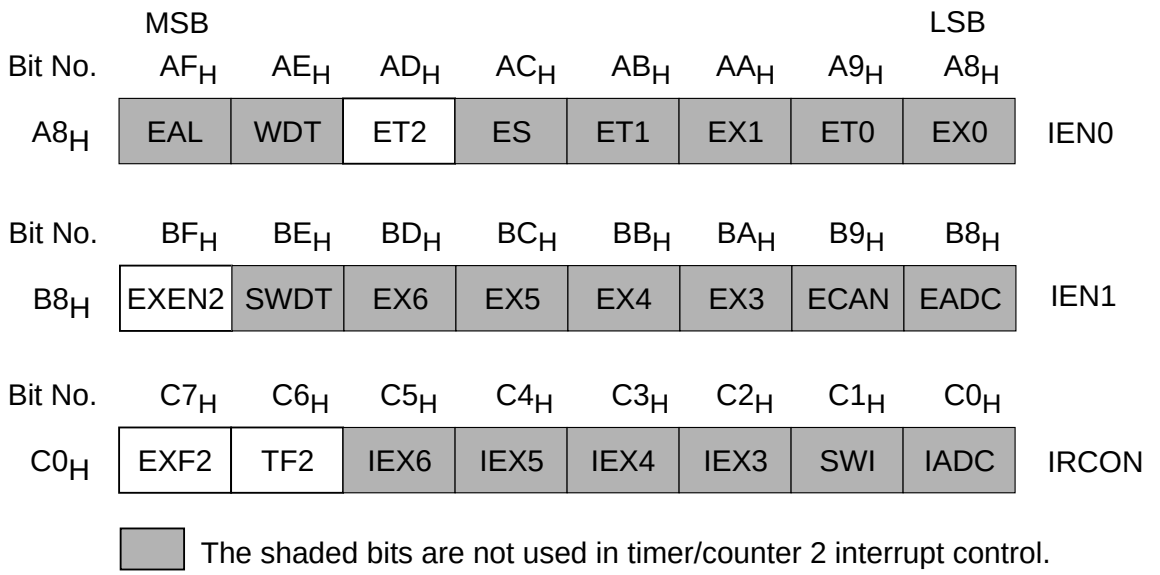
T2I1	T2I0	Function
0	0	No input selected, timer 2 stops
0	1	Timer function : input frequency = $f_{osc}/6$ (T2PS = 0) or $f_{osc}/12$ (T2PS = 1)
1	0	Counter function : external input signal at pin P1.7 / AN7 / T2
1	1	Gated timer function : input controlled by pin P1.7 / AN7 / T2

Special Function Register TL2 (Address CC_H) **Reset Value : 00_H**
Special Function Register TH2 (Address CD_H) **Reset Value : 00_H**
Special Function Register CRCL (Address CA_H) **Reset Value : 00_H**
Special Function Register CRCH (Address CB_H) **Reset Value : 00_H**

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

Bit	Function
TL2.7-0	Timer 2 value low byte The TL2 register holds the 8-bit low part of the 16-bit timer 2 count value.
TH2.7-0	Timer 2 value high byte The TH2 register holds the 8-bit high part of the 16-bit timer 2 count value.
CRCL.7-0	Reload 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	Reload 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 Register IEN0 (Address $A8_H$) **Reset Value : 00_H**
Special Function Register IEN1 (Address $B8_H$) **Reset Value : 00_H**
Special Function Register IRCON (Address $C0_H$) **Reset Value : 00_H**



Bit	Function
ET2	Timer 2 overflow / external reload interrupt enable. If ET2 = 0, the timer 2 interrupt is disabled. If ET2 = 1, the timer 2 interrupt is enabled.
EXEN2	Timer 2 external reload interrupt enable If EXEN2 = 0, the timer 2 external reload interrupt is disabled. If EXEN2 = 1, the timer 2 external reload interrupt is enabled. The external reload function is not affected by EXEN2.
EXF2	Timer 2 external reload flag EXF2 is set when a reload is caused by a falling edge on pin T2EX while EXEN2 = 1. If ET2 in IEN0 is set (timer 2 interrupt enabled), EXF2 = 1 will cause an interrupt. EXF2 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.

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

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
$C1_H$	COCAH3	COCAL3	COCAH2	COCAL2	COCAH1	COCAL1	COCAH0	COCAL0	CCEN

Bit	Function
COCAH3 COCAL3	Compare/capture mode for CC register 3

COCAH3	COCAL3	Function
0	0	Compare/capture disabled
0	1	Capture on rising edge at pin P1.3 / AN3 / INT6 / CC3
1	0	Compare enabled
1	1	Capture on write operation into register CCL3

Bit	Function
COCAH2 COCAL2	Compare/capture mode for CC register 2

COCAH2	COCAL2	Function
0	0	Compare/capture disabled
0	1	Capture on rising edge at pin P1.2 / AN2 / INT5 / CC2
1	0	Compare enabled
1	1	Capture on write operation into register CCL2

Bit	Function
COCAH1 COCAL1	Compare/capture mode for CC register 1

COCAH1	COCAL1	Function
0	0	Compare/capture disabled
0	1	Capture on rising edge at pin P1.1 / AN1 / INT4 / CC1
1	0	Compare enabled
1	1	Capture on write operation into register CCL1

Bit	Function
COCAH0 COCAL0	Compare/capture mode for CRC register

COCAH0	COCAL0	Function
0	0	Compare/capture disabled
0	1	Capture on falling/rising edge at pin P1.0 / AN0 / $\overline{INT3}$ / CC0
1	0	Compare enabled
1	1	Capture on write operation into register CRCL

6.2.2.2 Timer 2 Operation

The timer 2, which is a 16-bit-wide register, can operate as timer, event counter, or gated timer. The detailed operation is described below.

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 1/6 or 1/12 of the oscillator frequency. Thus, the 16-bit timer register (consisting of TH2 and TL2) is either incremented in every machine cycle or in every second machine cycle. The prescaler is selected by bit T2PS in special function register T2CON. lf T2PS is cleared, the input frequency is 1/6 of the oscillator frequency. if T2PS is set, the 2:1 prescaler gates 1/12 of the oscillator frequency to the timer.

Gated Timer Mode

In gated timer function, the external input pin T2 (P1.7) functions as a gate to the input of timer 2. lf T2 is high, the internal clock input is gated to the timer. T2 = 0 stops the counting procedure. This facilitates pulse width measurements. The external gate signal is sampled once every machine cycle.

Event Counter Mode

In the counter function, the timer 2 is incremented in response to a 1-to-0 transition at its corresponding external input pin T2 (P1.7). 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 (12 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is 1/12 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.

Note: The prescaler must be off for proper counter operation of timer 2, i.e. T2PS must be 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 in SFR IRCON, which can generate an interrupt.

lf 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. Both request flags cause the program to branch to the same vector address.

Reload of Timer 2

The reload mode for timer 2 is selected by bits T2R0 and T2R1 in SFR T2CON. **Figure 6-14** shows the configuration of timer 2 in reload mode.

Mode 0 : When timer 2 rolls over from all l'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 registers, which are preset by software. The reload will happen in the same machine cycle in which TF2 is set, thus overwriting the count value 0000_H.

Mode 1 : A 16-bit reload from the CRC register is caused by a negative transition at the corresponding input pin P1.5/AN5/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.

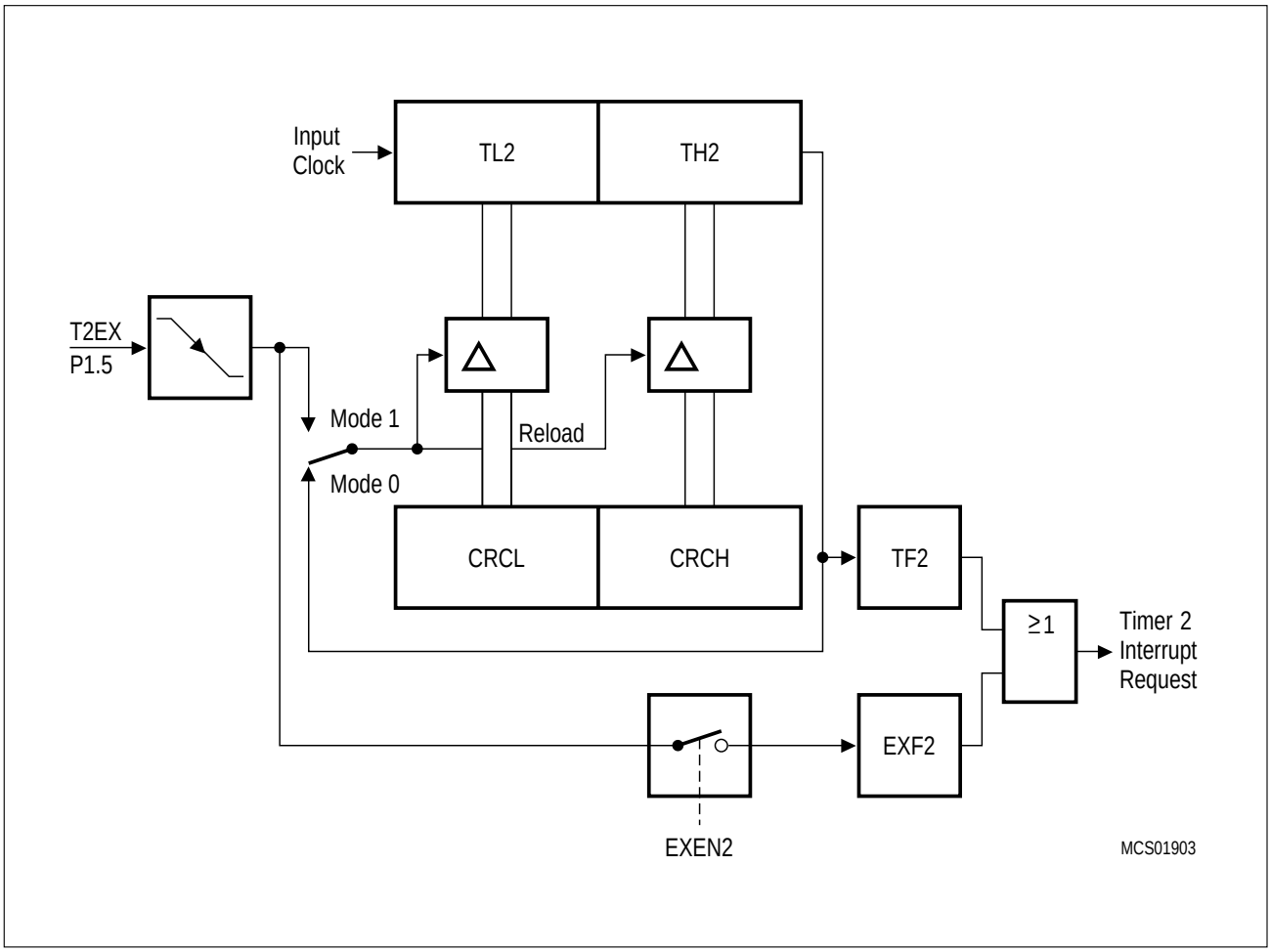


Figure 6-14
Timer 2 in Reload Mode

6.2.2.3 Compare Function of Registers CRC, CC1 to CC3

The compare function of a timer/register combination can be described as follows. The 16-bit value stored in a 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, and an interrupt is requested.

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. Two compare modes are implemented to cover a wide range of possible applications.

The compare modes 0 and 1 are selected by bit T2CM in special function register T2CON. In both compare modes, the new value arrives at the port pin 1 within the same machine cycle in which the internal compare signal is activated.

The four registers CRC, CC1 to CC3 are multifunctional as they additonally provide a capture, compare or reload capability (CRC register only). A general selection of the function is done in register CCEN. Please note that the compare interrupt CC0 can be programmed to be negative or positive transition activated. The internal compare signal (not the output signal at the port pin!) is active as long as the timer 2 contents is equal to the one of the appropriate compare registers, and it has a rising and a falling edge. Thus, when using the CRC register, it can be selected whether an interrupt should be caused when the compare signal goes active or inactive, depending on bit I3FR in T2CON. For the CC registers 1 to 3 an interrupt is always requested when the compare signal goes active (see **figure 6-16**).

6.2.2.3.1 Compare Mode 0

In mode 0, upon matching the timer and compare register contents, the output signal changes from low to high. lt 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 6-15** shows a functional diagram of a port latch in compare mode 0. The port latch is directly controlled by the two signals timer overflow and compare. The input line from the internal bus and the write-to-latch line 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). 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 6-16** and **6-17** illustrate the function of compare mode 0.

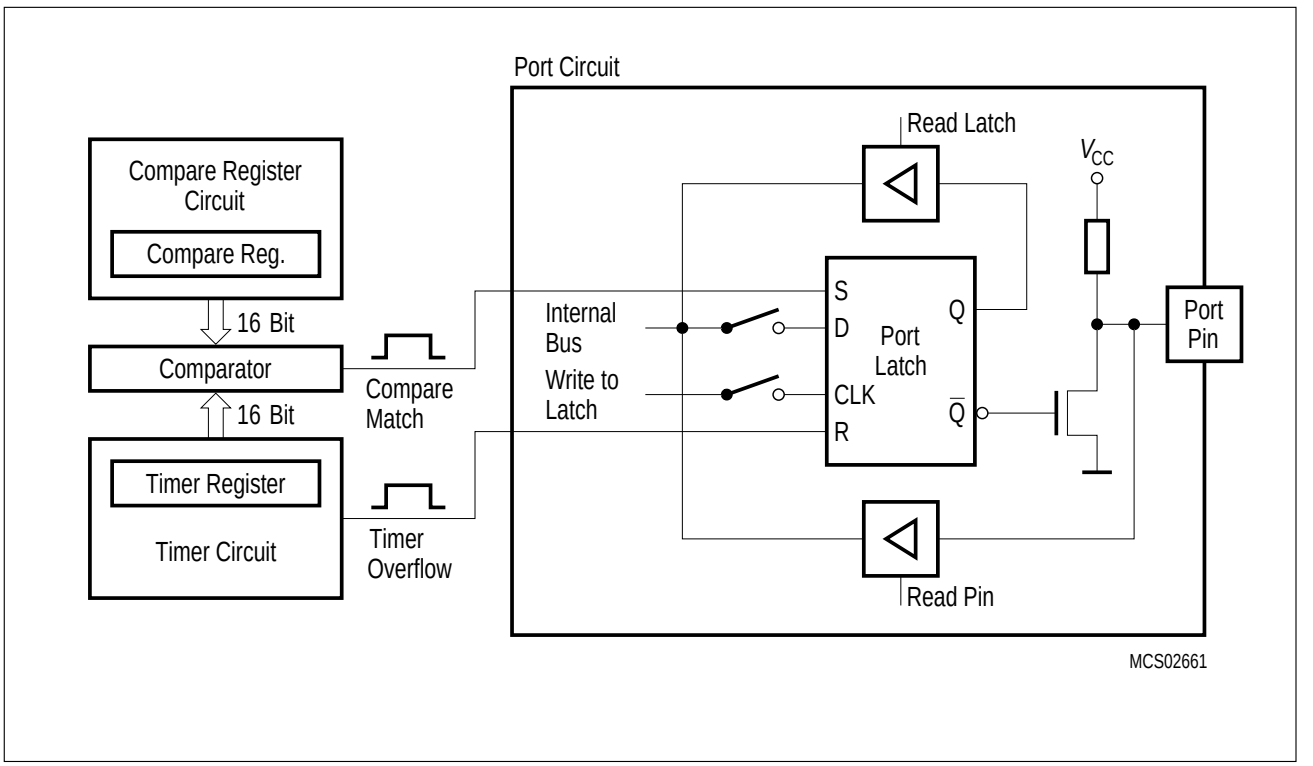


Figure 6-15
Port Latch in Compare Mode 0

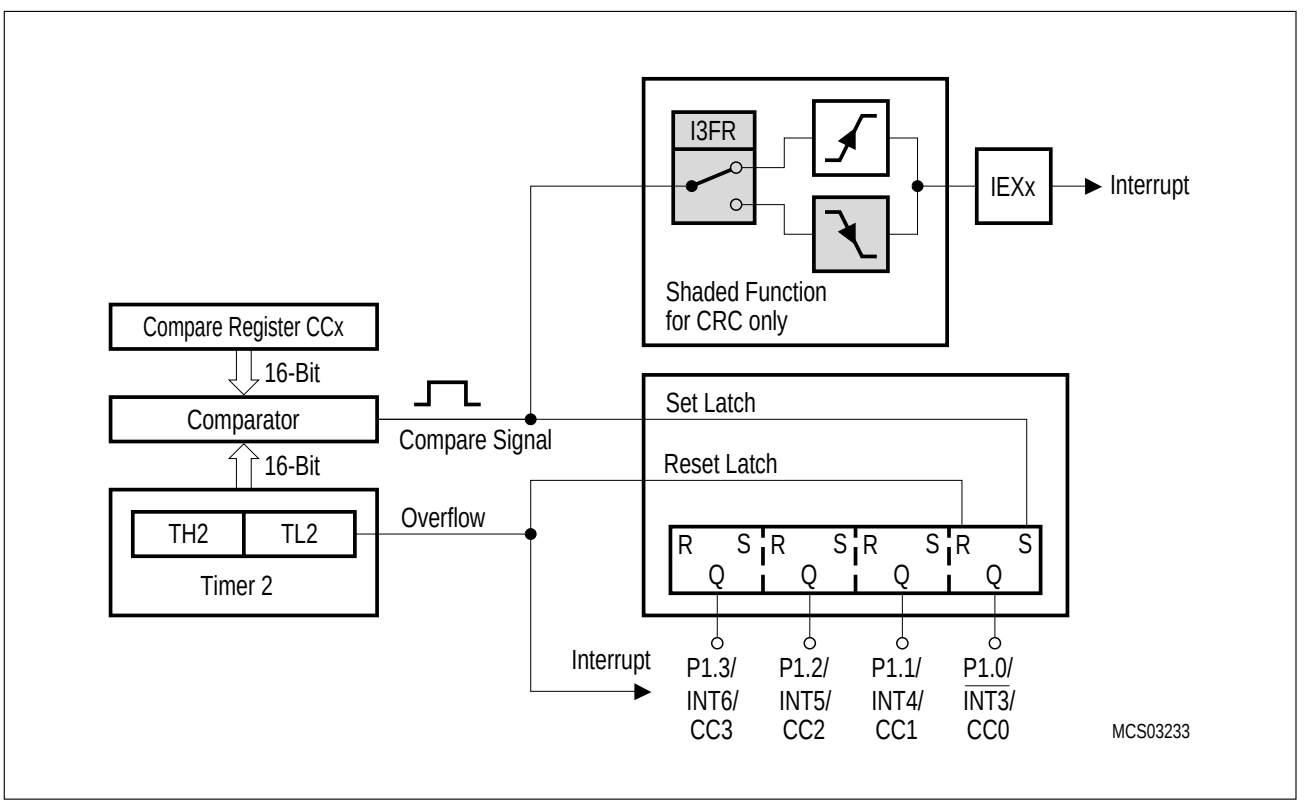


Figure 6-16
Timer 2 with Registers CCx in Compare Mode 0

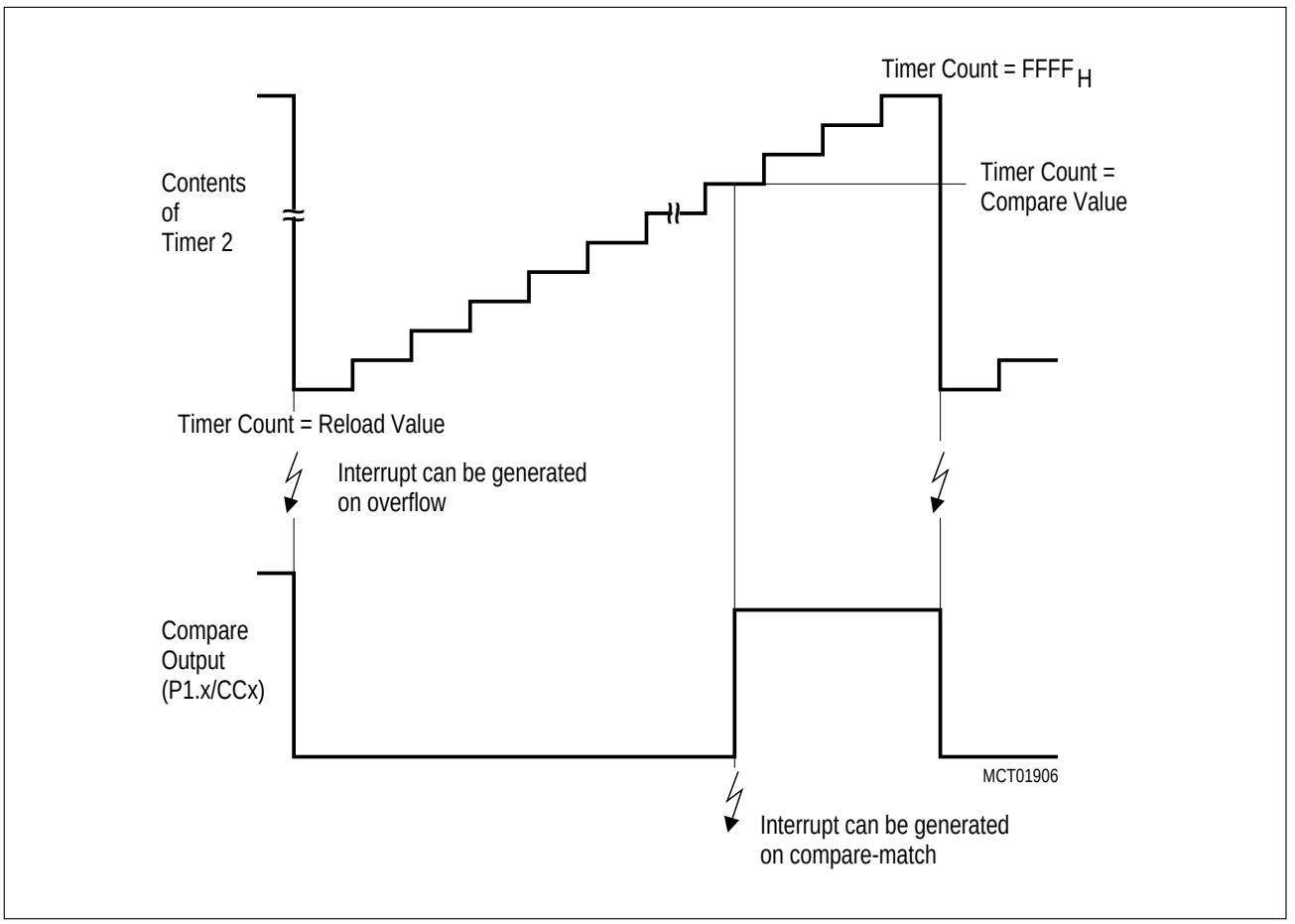


Figure 6-17
Function of Compare Mode 0

6.2.2.3.2 Modulation Range in Compare Mode 0

Generally it can be said that for every PWM generation in compare mode 0 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 would then 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.

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 6-18** where the maximum and minimum duty cycle of a compare output signal are illustrated. Timer 2 is incremented with the machine clock (fosc/6), thus at 20-MHz operational frequency, these spikes are both approx. 150 ns long.

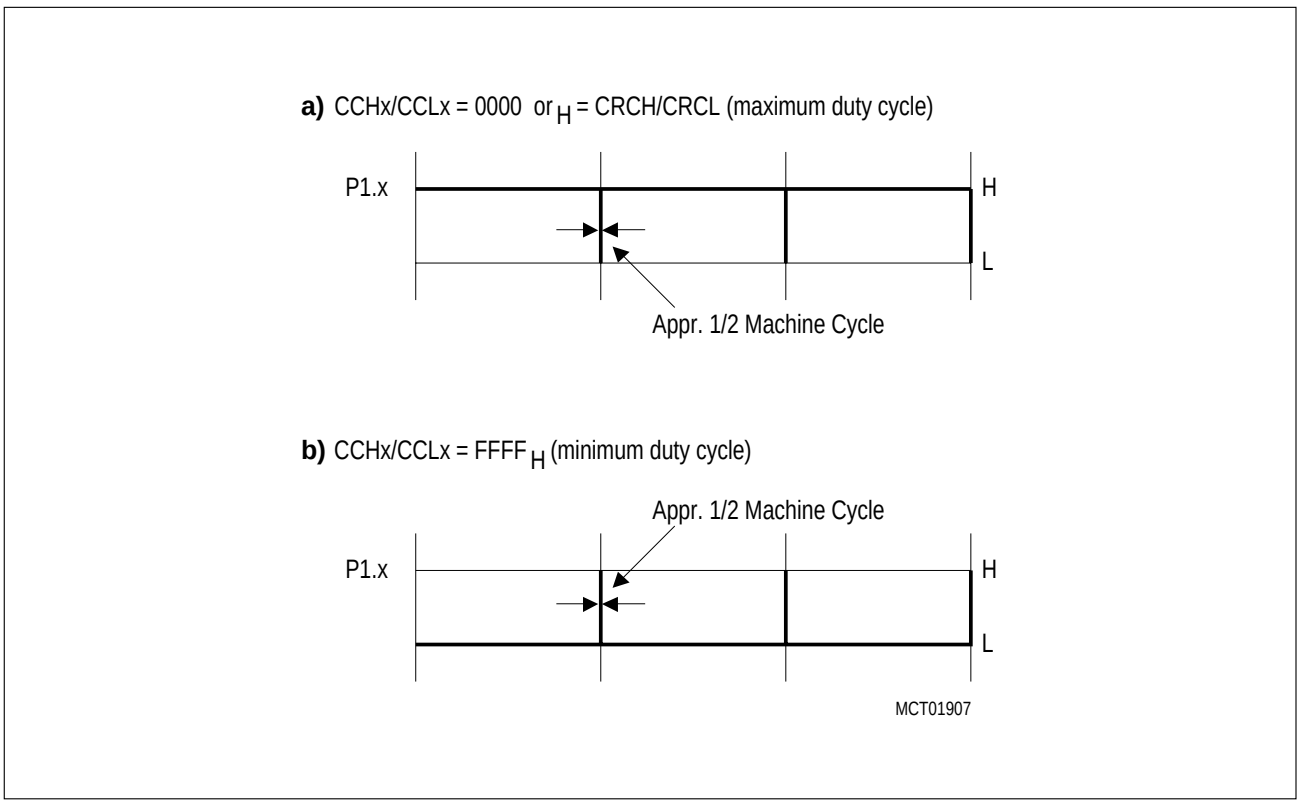


Figure 6-18
Modulation Range of a PWM Signal, generated with a Timer 2/CCx Register Combination in Compare Mode 0*

The following example shows how to calculate the modulation range for a PWM signal. To calculate 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 = $FF00_H$

Restriction of modulation range = 1 / (256 x 2) x 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.

6.2.2.3.3 Compare Mode 1

In compare mode 1, the software adaptively determines the transition of the output signal. lt 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.

lf 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, one can choose whether the output signal is to make a new transition (1-to-0 or 0-to-1, depending on the actual pinlevel) or should keep its old value at the time the timer 2 count matches the stored compare value.

Figure 6-19 and **figure 6-20** show functional diagrams of the timer/compare register/port latch configuration in compare mode 1. In this function, the port latch consists of two separate latches. The upper latch (which acts as a “shadow latch“) can be written under software control, but its value will only be transferred to the output latch (and thus to the port pin) in response to a compare match.

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 driving timer 2 with a slow external 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.

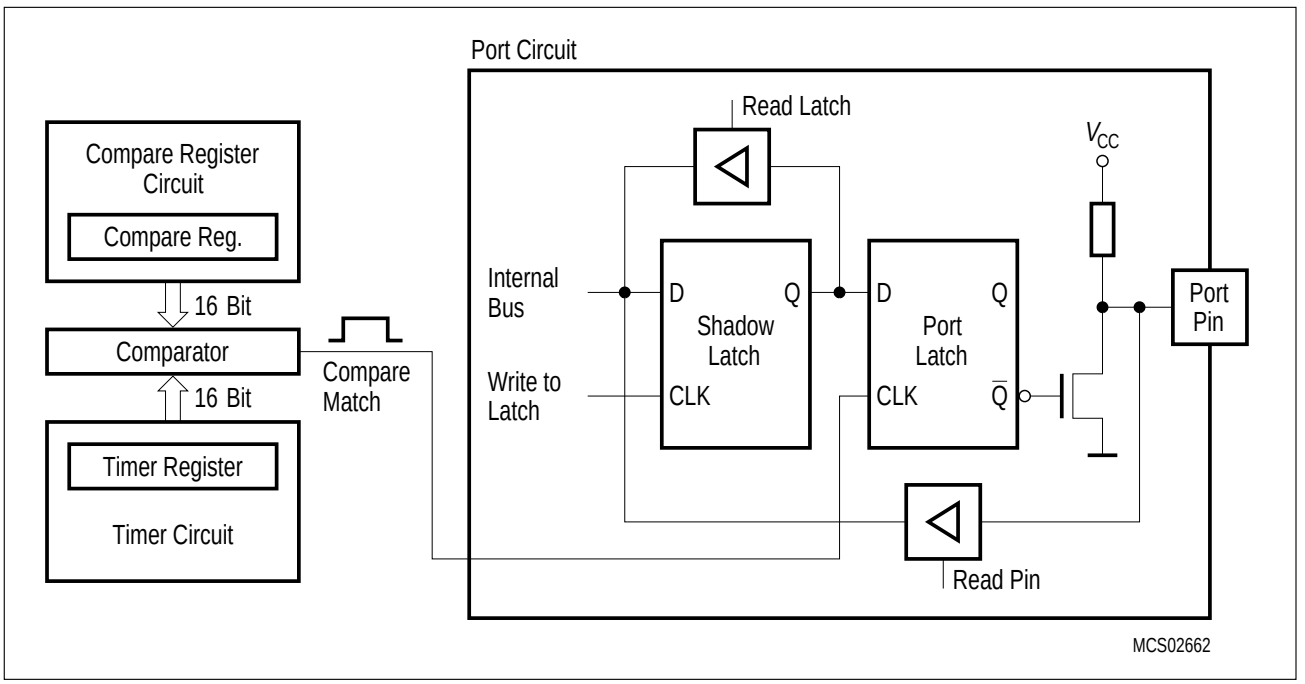


Figure 6-19
Port Latch in Compare Mode 1

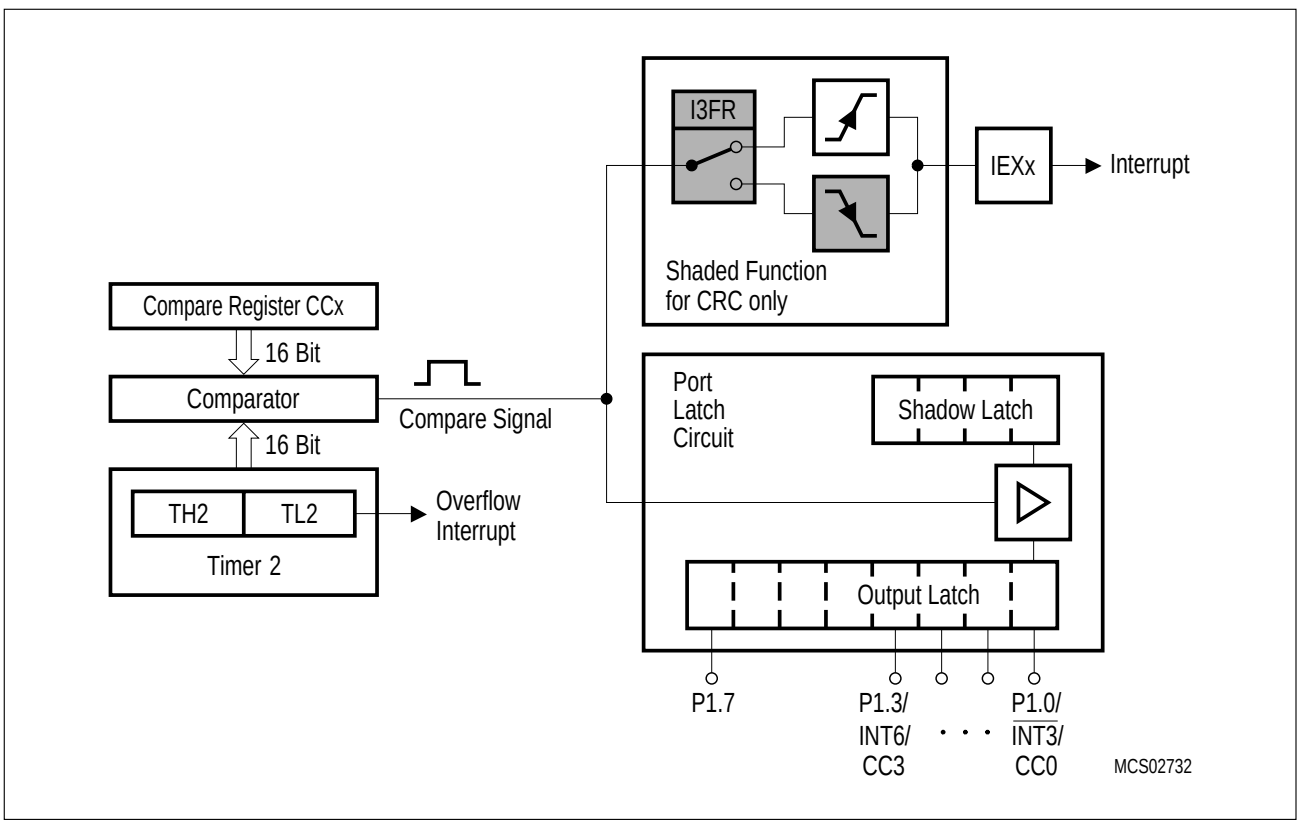


Figure 6-20
Timer 2 with Registers CCx in Compare Mode 1

(CCx stands for CRC, CC1 to CC3, IEXx stands for IEX3 to IEX6)

6.2.2.4 Using Interrupts in Combination with the Compare Function

The compare service of registers CRC, CC1, CC2 and CC3 are assigned to alternate output functions at port pins P1.0 to P1.3. 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.

Advantages when 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 suff icient space of time.

Please note the following special case where a program using compare interrupts could show a "surprising" behavior:

The 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, you can select whether an interrupt should be generated when the compare signal goes active or inactive, depending on the status of bit I3FR in T2CON.

Initializing the interrupt to be negative transition triggered is advisable 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 CC1 to CC3 registers an interrupt is always requested when the compare signal goes active.

The second configuration which should be noted is when compare function is combined with negative transition activated interrupts. lf the port latch of port P1.0 contains a 1, the interrupt request flags IEX3 will immediately be set after enabling the compare mode for the CRC 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 lf the request flag is cleared by software after the compare is activated and before the external interrupt is enabled.

6.2.2.5 Capture Function

Each of the compare/capture registers CC1 to CC3 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 this function. In mode 0, an external event latches the timer 2 contents to a dedicated capture register. In mode 1, a capture will occur upon writing to the low order byte of the dedicated 16-bit capture register. This mode is provided to allow the software to read the timer 2 contents “on-the-fly“.

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

- for CC registers 1 to 3: a positive transition at pins CC1 to CC3 of port 1
- for the CRC register: a positive or negative transition at the corresponding pin, depending on the status of the bit I3FR in SFR T2CON. If the edge flag is cleared, a capture occurs in response to a negative transition; If the edge flag is set a capture occurs in response to a positive transition at pin P1.0 / $\overline{\text{INT3}}$ / CC0.

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 contents is latched to the appropriate capture register in the cycle following the one in which the transition was identified.

In mode 0 a transition at the external capture inputs of registers CC1 to CC3 will also set the corresponding external interrupt request flags IEX3 to IEX6. If the interrupts are enabled, an external capture signal will cause the CPU to vector to the appropriate interrupt service routine.

In 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.

Figure 6-21 illustrates the operation of the CRC register, while **Figure 6-21a** shows the operation of the compare/ capture registers 1 to 3.

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

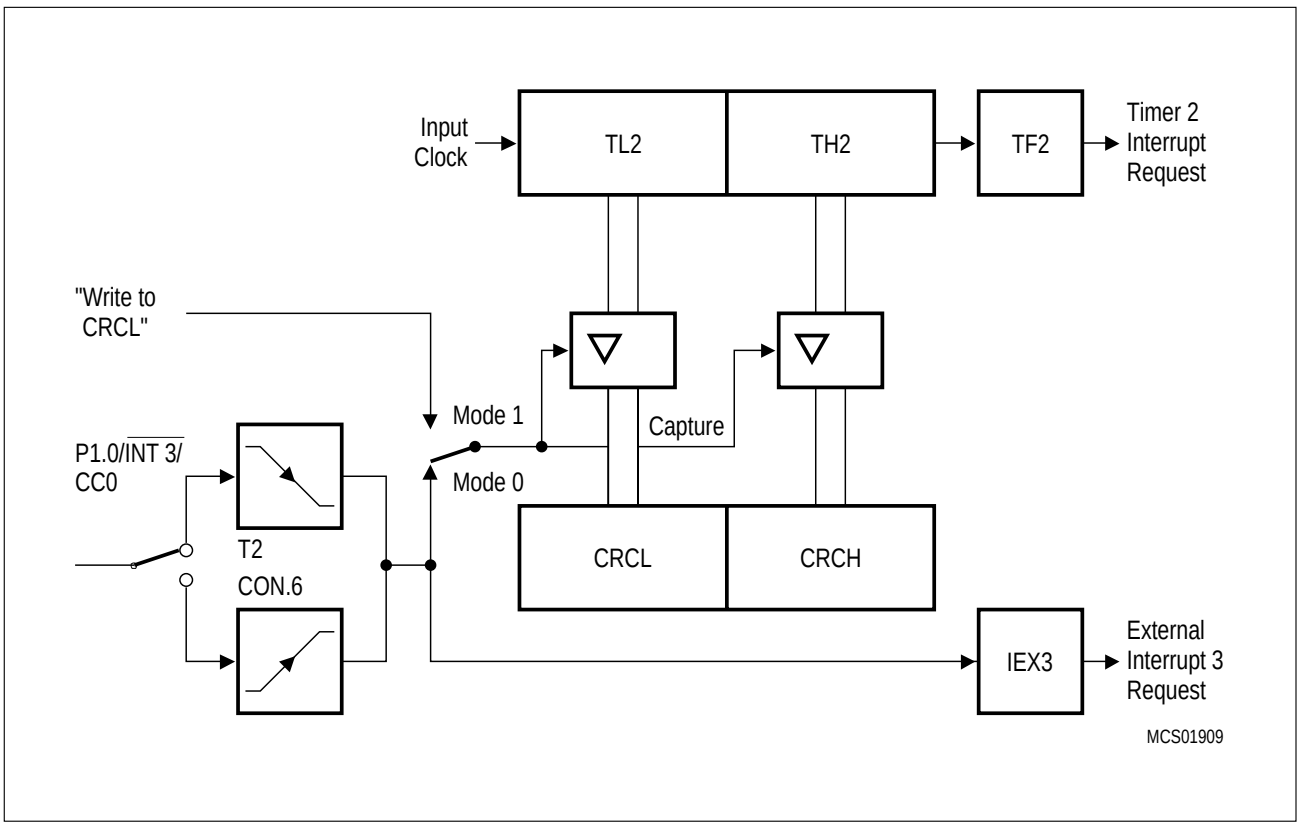


Figure 6-21
Timer 2 - Capture with Register CRC

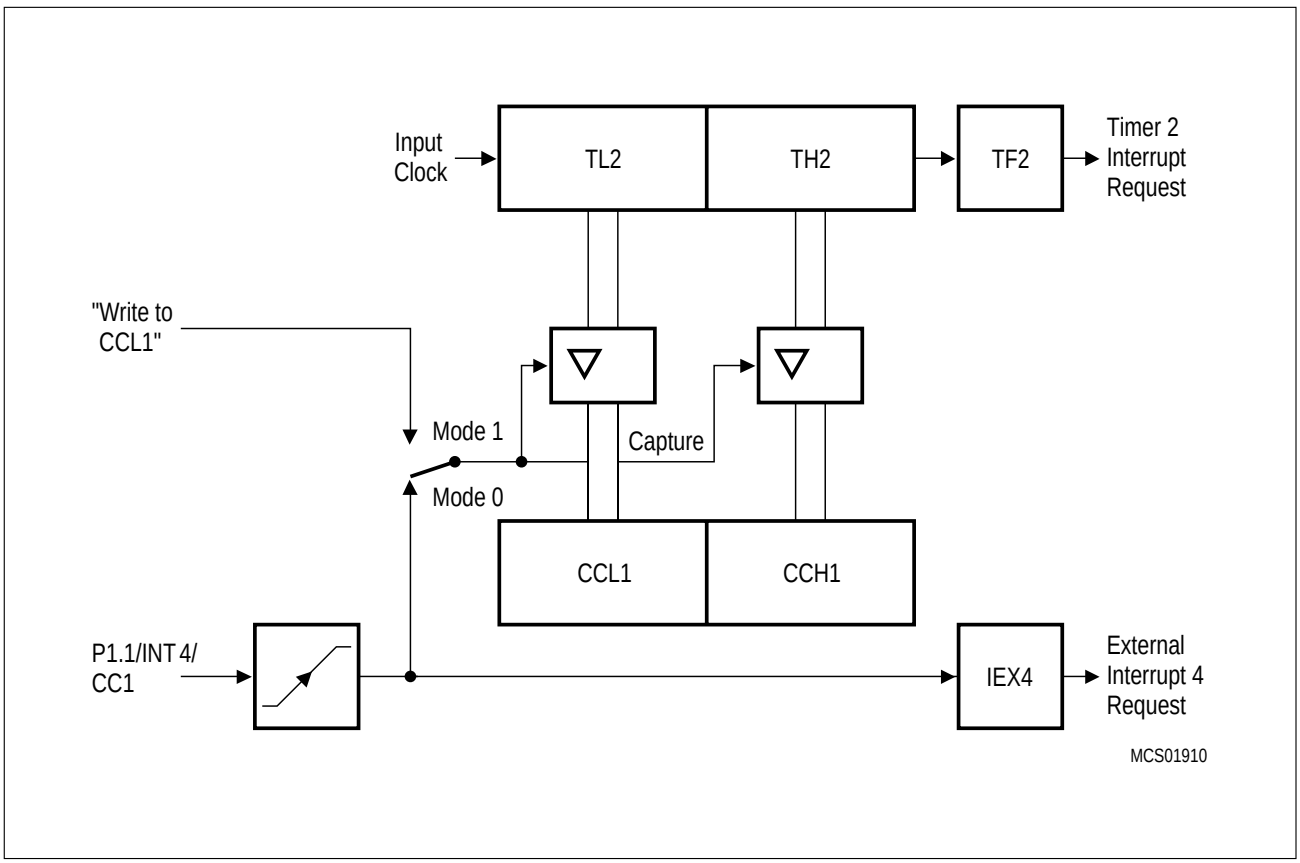


Figure 6-21a
Timer 2 - Capture with Registers CC1 to CC3

6.3 Serial Interface

The serial port of the C505 is full duplex, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the receive register (however, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost). The serial port receive and transmit registers are both accessed at special function register SBUF. Writing to SBUF loads the transmit register, and reading SBUF accesses a physically separate receive register.

The serial port can operate in 4 modes (one synchronous mode, three asynchronous modes). The baud rate clock for the serial port 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).

Mode 0, Shift Register (Synchronous) Mode:

Serial data enters and exits through RxD. TxD 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 6.3.4 for more detailed information)

Mode 1, 8-Bit USART, Variable Baud Rate:

10 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in special function register SCON. The baud rate is variable. (See section 6.3.5 for more detailed information)

Mode 2, 9-Bit USART, Fixed Baud Rate:

11 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). On transmit, the 9th data bit (TB8 in SCON) can be assigned to the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On receive, the 9th data bit goes into RB8 in special function register SCON, while the stop bit is ignored. The baud rate is programmable to either 1/16 or 1/32 of the oscillator frequency. (See section 6.3.6 for more detailed information)

Mode 3, 9-Bit USART, Variable Baud Rate:

11 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). In fact, mode 3 is the same as mode 2 in all respects except the baud rate. The baud rate in mode 3 is variable. (See section 6.3.6 for more detailed information)

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

The serial interface also provides interrupt requests when transmission or reception of a frames have been completed. The corresponding interrupt request flags are TI or RI, resp. See chapter 7 of this user manual for more details about the interrupt structure. The interrupt request flags TI and RI can also be used for polling the serial interface, if the serial interrupt is not to be used (i.e. serial interrupt not enabled).

6.3.1 Multiprocessor Communication

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

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is beeing addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2s set and go on about their business, ignoring the incoming data bytes.

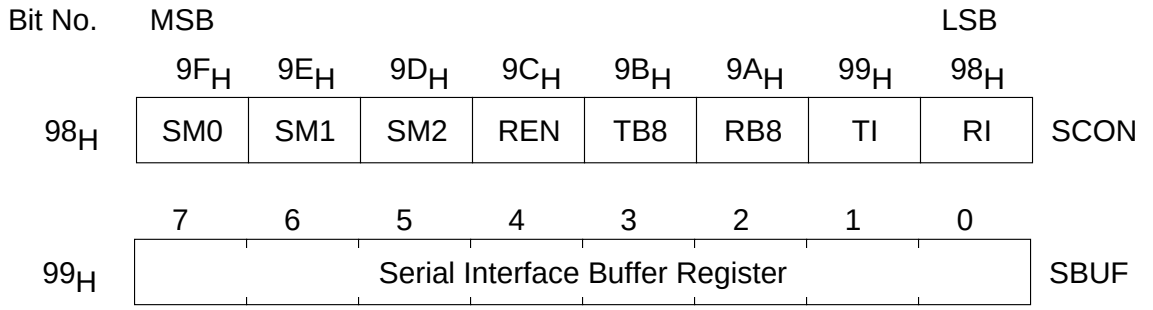
SM2 has no effect in mode 0. SM2 can be used in mode 1 to check the validity of the stop bit. In a mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

6.3.2 Serial Port Registers

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

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

Special Function Register SCON (Address 98_H) **Reset Value : 00_H**
Special Function Register SBUF (Address 99_H) **Reset Value : XX_H**



Bit	Function
SM0 SM1	Serial port 0 operating mode selection bits

SM0	SM1	Selected operating mode
0	0	Serial mode 0 : Shift register, fixed baud rate (f_{OSC}/6)
0	1	Serial mode 1 : 8-bit UART, variable baud rate
1	0	Serial mode 2 : 9-bit UART, fixed baud rate (f_{OSC}/16 or f_{OSC}/32)
1	1	Serial mode 3 : 9-bit UART, variable baud rate

SM2	Enable serial port multiprocessor communication in modes 2 and 3 In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0.
REN	Enable receiver of serial port Enables serial reception. Set by software to enable serial reception. Cleared by software to disable serial reception.
TB8	Serial port transmitter bit 9 TB8 is the 9th data bit that will be transmitted in modes 2 and 3. Set or cleared by software as desired.
RB8	Serial port receiver bit 9 In modes 2 and 3, RB8 is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.
TI	Serial port transmitter interrupt flag TI 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. TI must be cleared by software.
RI	Serial port receiver interrupt flag RI 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 SM2). RI must be cleared by software.

6.3.3 Baud Rate Generation

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

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 external clock frequency (oscillator or external input clock operation).

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

Special Function Register ADCON0 (Address $D8_H$) **Reset Value : $00X00000_B$**
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	–	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 for controlling the baud rate.

Bit	Function
BD	Baud rate generator enable When set, the baud rate of serial interface is derived from the dedicated 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 in modes 1, 2, 3 is doubled. After reset this bit is cleared.
–	Reserved bits for future use. Read by CPU returns undefined values.

Figure 6-22 shows the configuration for the baud rate generation of the serial port.

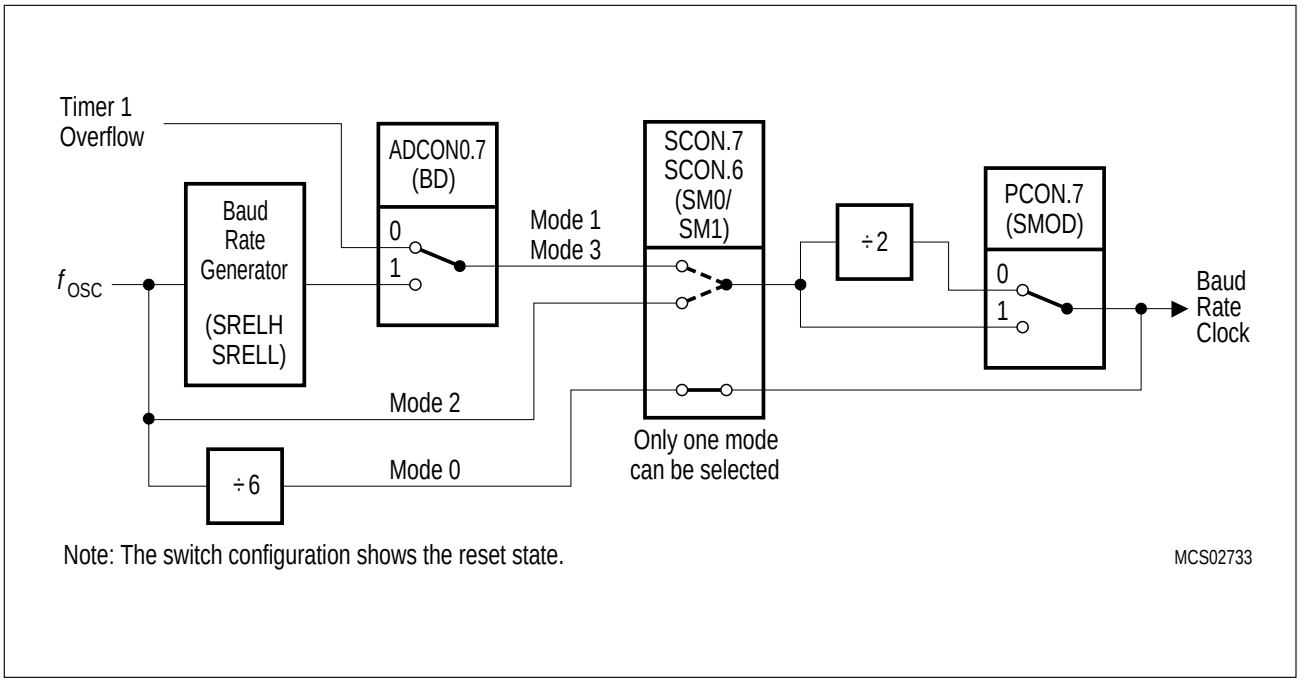


Figure 6-22
Baud Rate Generation for the Serial Port

Depending on the programmed operating mode different paths are selected for the baud rate clock generation. **Figure 6-22** shows the dependencies of the serial port baud rate clock generation on the two control bits and from the mode which is selected in the special function register SCON.

6.3.3.1 Baud Rate in Mode 0

The baud rate in mode 0 is fixed to :

$$\text{Mode 0 baud rate} = \frac{\text{oscillator frequency}}{6}$$

6.3.3.2 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 1/32 of the oscillator frequency. If SMOD = 1, the baud rate is 1/16 of the oscillator frequency.

$$\text{Mode 2 baud rate} = \frac{2^{\text{SMOD}}}{32} \times \text{oscillator frequency}$$

6.3.3.3 Baud Rate in Mode 1 and 3

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

6.3.3.3.1 Using the Internal Baud Rate Generator

In modes 1 and 3, the C505 can use an internal baud rate generator for the serial port. 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 affect 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}.

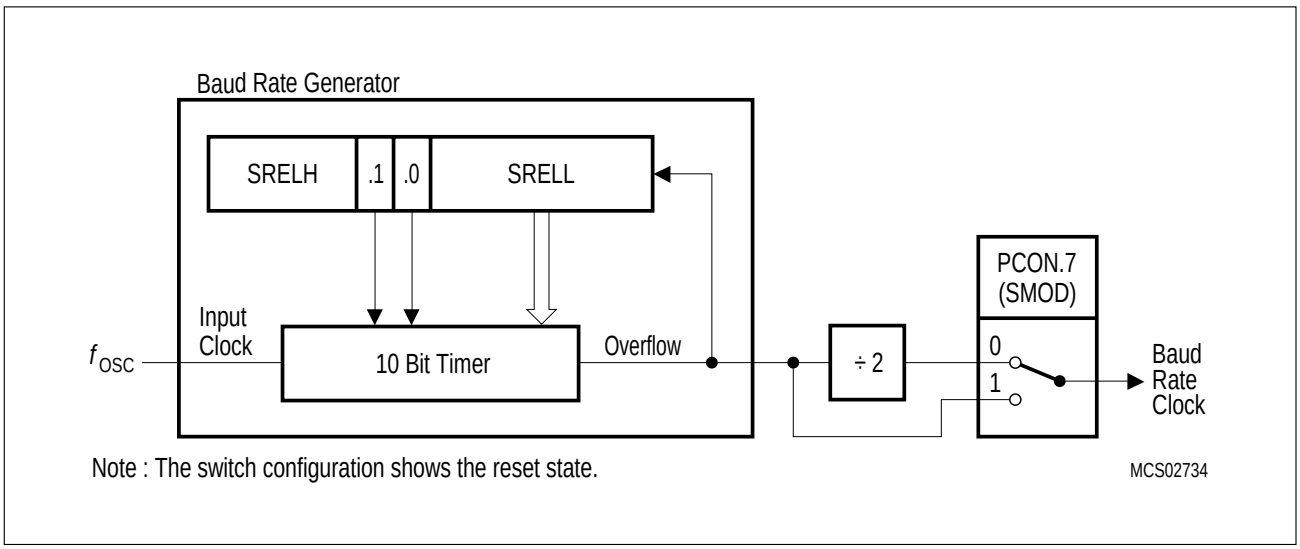
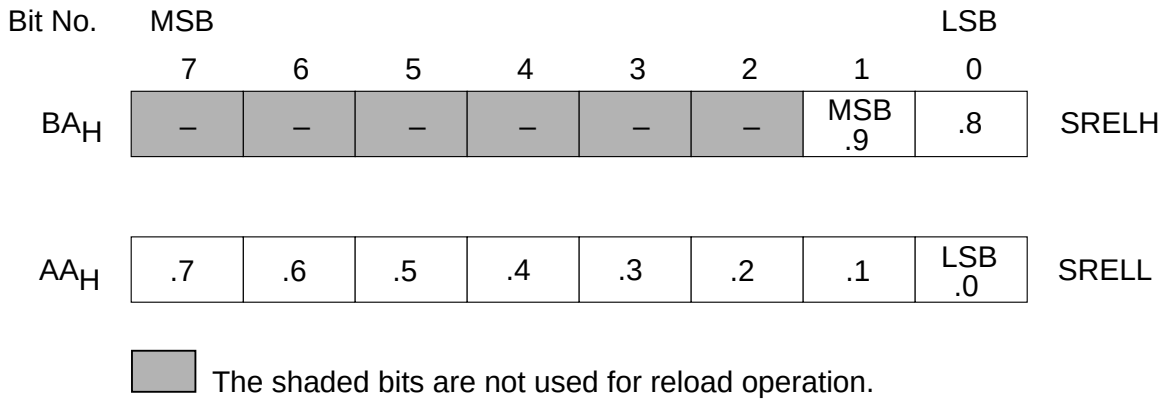


Figure 6-23
Serial Port Input Clock when using the Baud Rate Generator

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 SRELL and SRELH. The lower 8 bits of the timer are reloaded from SRELL, while the upper two bits are reloaded from bit 0 and 1 of register SRELH. The baud rate timer is reloaded by writing to SRELL.

Special Function Register SRELH (Address BA_H) **Reset Value : $XXXXXX11_B$**
Special Function Register SRELL (Address AA_H) **Reset Value : $D9_H$**



Bit	Function
SRELH.0-1	Baudrate generator reload high value Upper two bits of the baudrate timer reload value.
SRELL.0-7	Baudrate generator reload low value Lower 8 bits of the baudrate timer reload value.
–	Reserved bits for future use. Read by CPU returns undefined values.

After reset SRELH and SRELL have a reload value of $3D9_H$. With this reload value the baud rate generator has an overflow rate of input clock / 39. With a 6 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 in mode 1 and 3, the baud rate of can be determined as follows:

$$\text{Mode 1, 3 baud rate} = \frac{2^{SMOD} \times \text{oscillator frequency}}{32 \times (\text{baud rate generator overflow rate})}$$

$$\text{Baud rate generator overflow rate} = 2^{10} - \text{SREL}$$
$$\text{with SREL} = \text{SRELH.1} - 0, \text{SRELL.7} - 0$$

6.3.3.3.2 Using Timer 1 to Generate Baud Rates

In mode 1 and 3 of the serial port also 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:

$$\text{Mode 1, 3 baud rate} = \frac{2^{SMOD} \times \text{oscillator frequency}}{32 \times 6 \times (256 - (TH1))}$$

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.

6.3.4 Details about Mode 0

Serial data enters and exists through RxD. TxD outputs the shift clock. 8 data bits are transmitted/ received: (LSB first). The baud rate is fixed at $f_{OSC}/6$.

Figure 6-24 shows a simplified functional diagram of the serial port in mode 0. The associated timing is illustrated in **figure 6-25**.

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

SEND enables the output of the shift register to the alternate output function line of P3.0, and also enables SHIFT CLOCK to the alternate output function line of 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 are shifted to the right one position.

As data bits shift out to the right, zeroes come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just to the left of the MSB, and all positions to the left of that contain zeroes. This condition flags the TX control block to do one last shift and then deactivate SEND and set TI. Both of these actions occur at S1P1 of the 10th machine cycle after "Write-to-SBUF".

Reception is initiated by the condition REN = 1 and RI = 0. At S6P2 of 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 of every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the receive shift register are shifted to the left one position. The value that comes in from the right is the value that was sampled at the P3.0 pin at S5P2 of the same machine cycle.

As data bit comes in from the right, 1s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX control block to do one last shift and load SBUF. At S1P1 of the 10th machine cycle after the write to SCON that cleared RI, RECEIVE is cleared and RI is set.

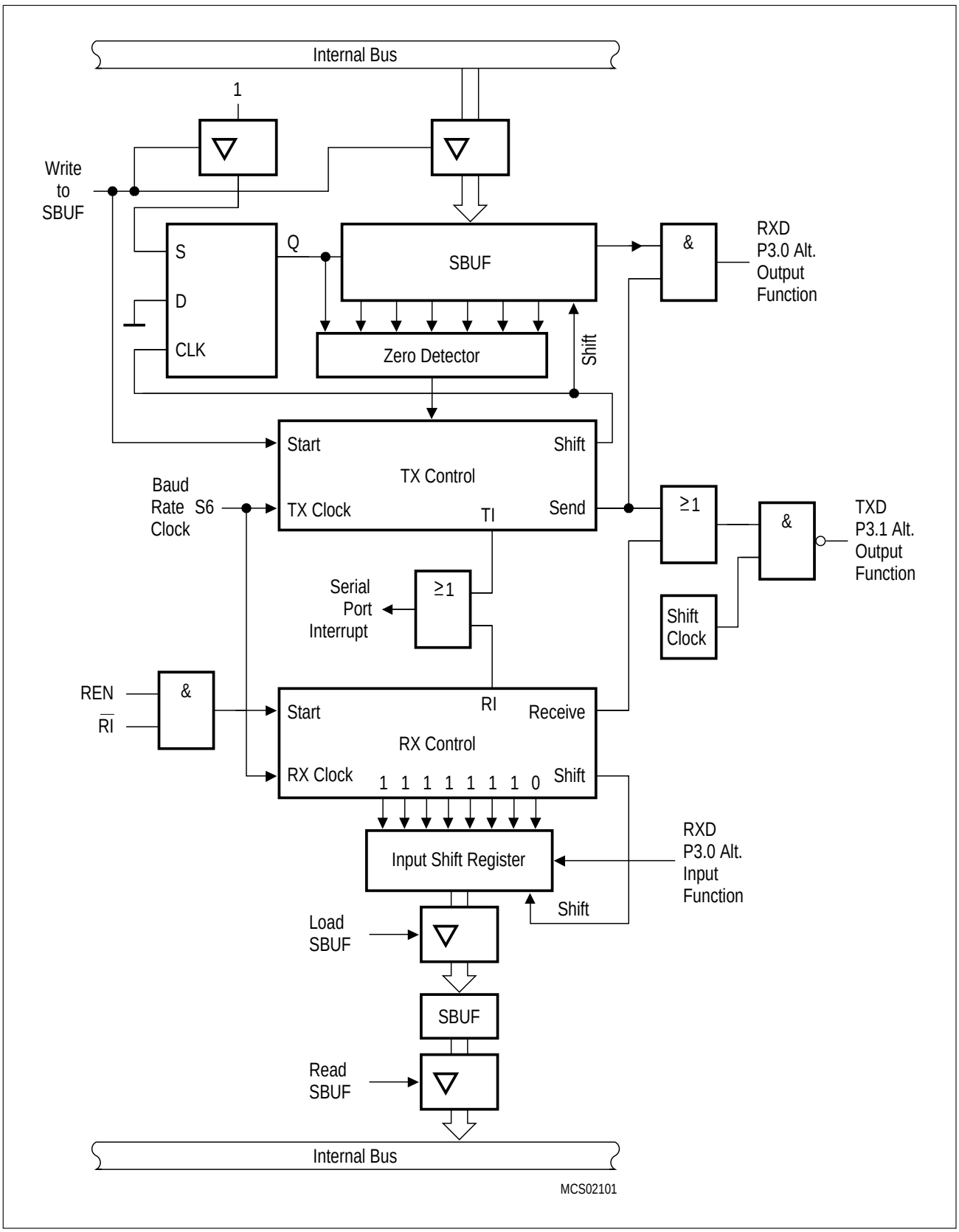


Figure 6-24
Serial Interface, Mode 0, Functional Diagram

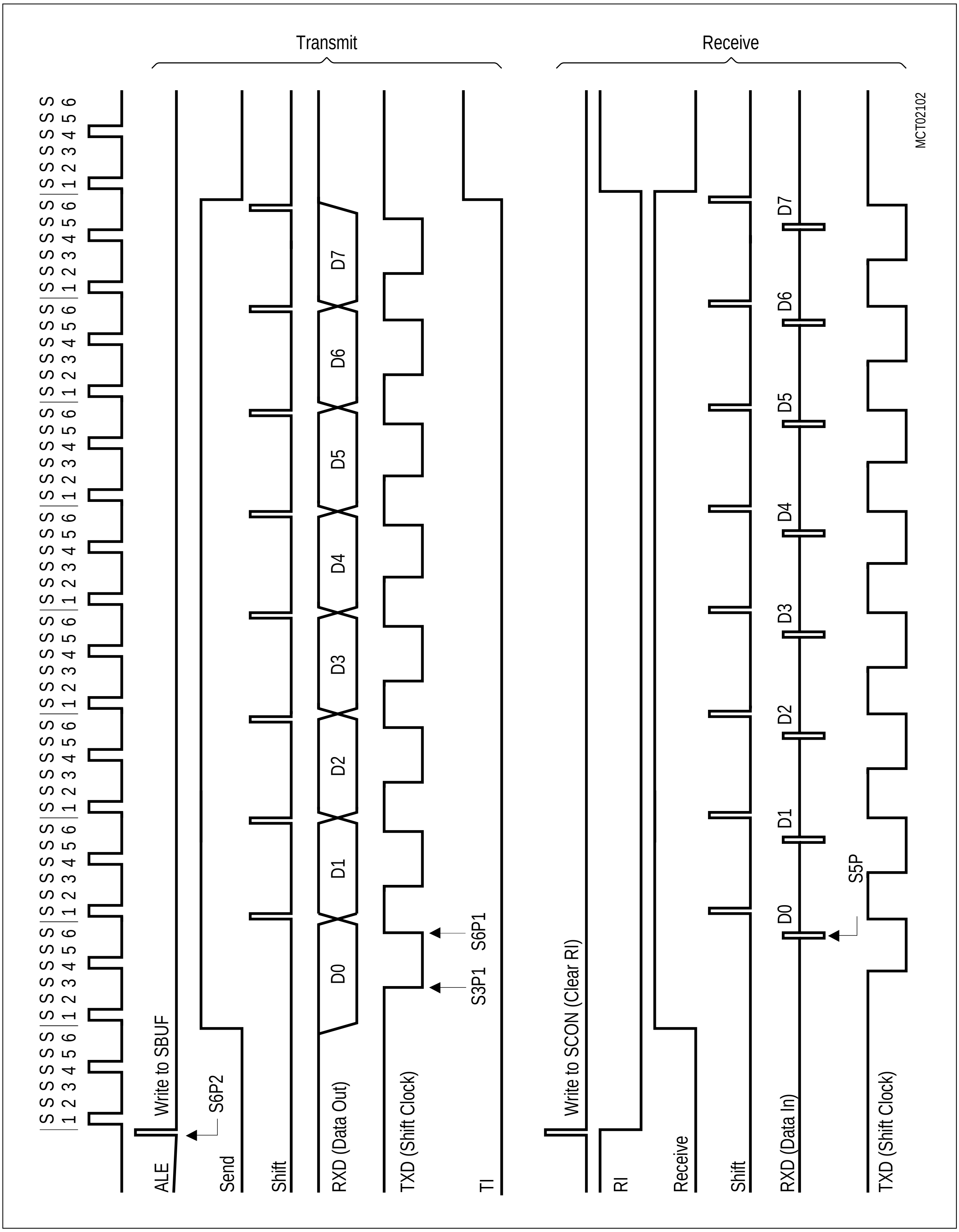


Figure 6-25
Serial Interface, Mode 0, Timing Diagram

6.3.5 Details about Mode 1

Ten bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On reception, the stop bit goes into RB8 in SCON. The baud rate is determined either by the timer 1 overflow rate or by the internal baud rate generator.

Figure 6-26 shows a simplified functional diagram of the serial port in mode 1. The associated timings for transmit/receive are illustrated in **figure 6-27**.

Transmission is initiated by an instruction that uses SBUF as a destination register. The "Write-to-SBUF" signal also loads a 1 into the 9th bit position of the transmit shift register and flags the TX control unit that a transmission is requested. Transmission starts at the next rollover in the divide-by-16 counter. (Thus, the bit times are synchronized to the divide-by-16 counter, not to the "Write-to-SBUF" signal).

The transmission begins with activation of $\overline{\text{SEND}}$, which puts the start bit at TxD. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TxD. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeroes 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 zeroes. This condition flags the TX control unit to do one last shift and then deactivate $\overline{\text{SEND}}$ and set TI. This occurs at the 10th divide-by-16 rollover after "Write-to-SBUF".

Reception is initiated by a detected 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate has been established. When a transition is detected, the divide-by-16 counter is immediately reset, and $1FF_H$ 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 RxD. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for the noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection or false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

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

1) RI = 0, and
2) either SM2 = 0, or the received stop bit = 1

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

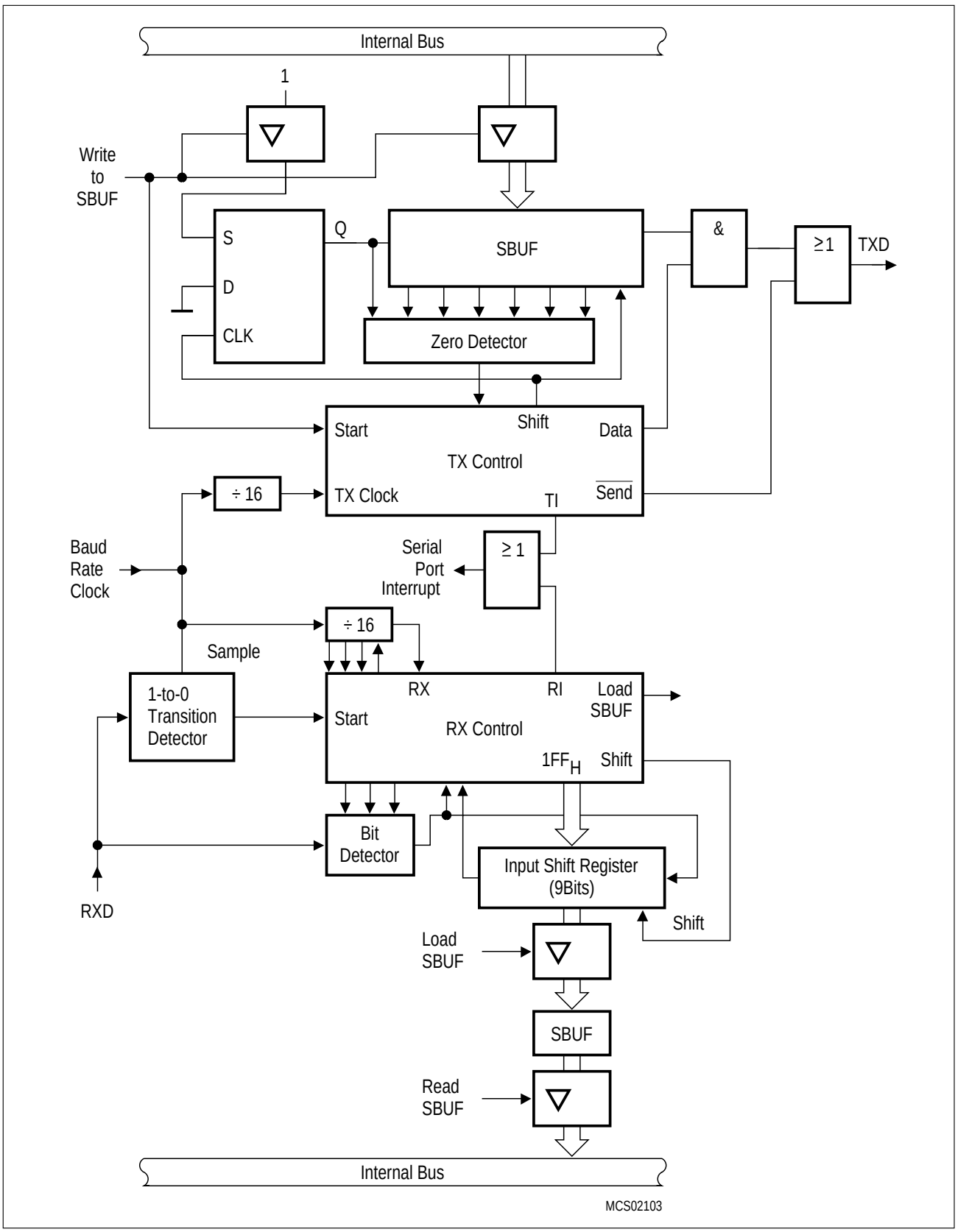


Figure 6-26
Serial Interface, Mode 1, Functional Diagram

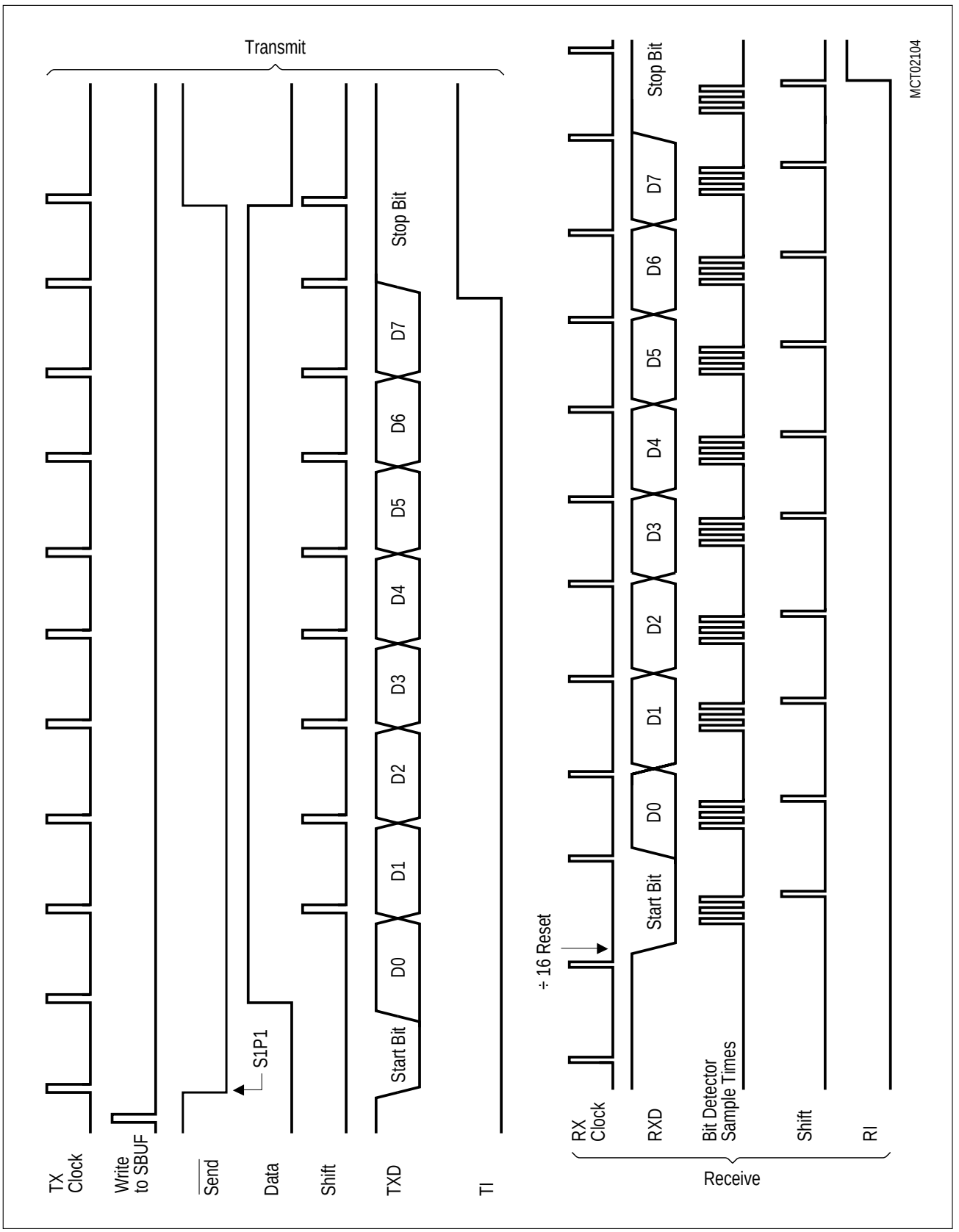


Figure 6-27
Serial Interface, Mode 1, Timing Diagram

6.3.6 Details about Modes 2 and 3

Eleven bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmission, the 9th data bit (TB8) can be assigned the value of 0 or 1. On reception, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to either 1/16 or 1/32 the oscillator frequency in mode 2 (When bit SMOD in SFR PCON (87_H) is set, the baud rate is $f_{OSC}/16$). In mode 3 the baud rate clock is generated by timer 1, which is incremented by a rate of $f_{OSC}/6$ or by the internal baud rate generator.

Figure 6-28 shows a functional diagram of the serial port in modes 2 and 3. The receive portion is exactly the same as in mode 1. The transmit portion differs from mode 1 only in the 9th bit of the transmit shift register. The associated timings for transmit/receive are illustrated in **figure 6-29**.

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

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

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

At the 7th, 8th and 9th counter states of each bit time, the bit detector samples the value of RxD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

As data bit come from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in modes 2 and 3 is a 9-bit register), it flags the RX control block to do one last shift, load SBUF and RB8, and to set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

1) RI = 0, and
2) Either SM2 = 0 or the received 9th data bit = 1

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

Note that the value of the received stop bit is irrelevant to SBUF, RB8 or RI.

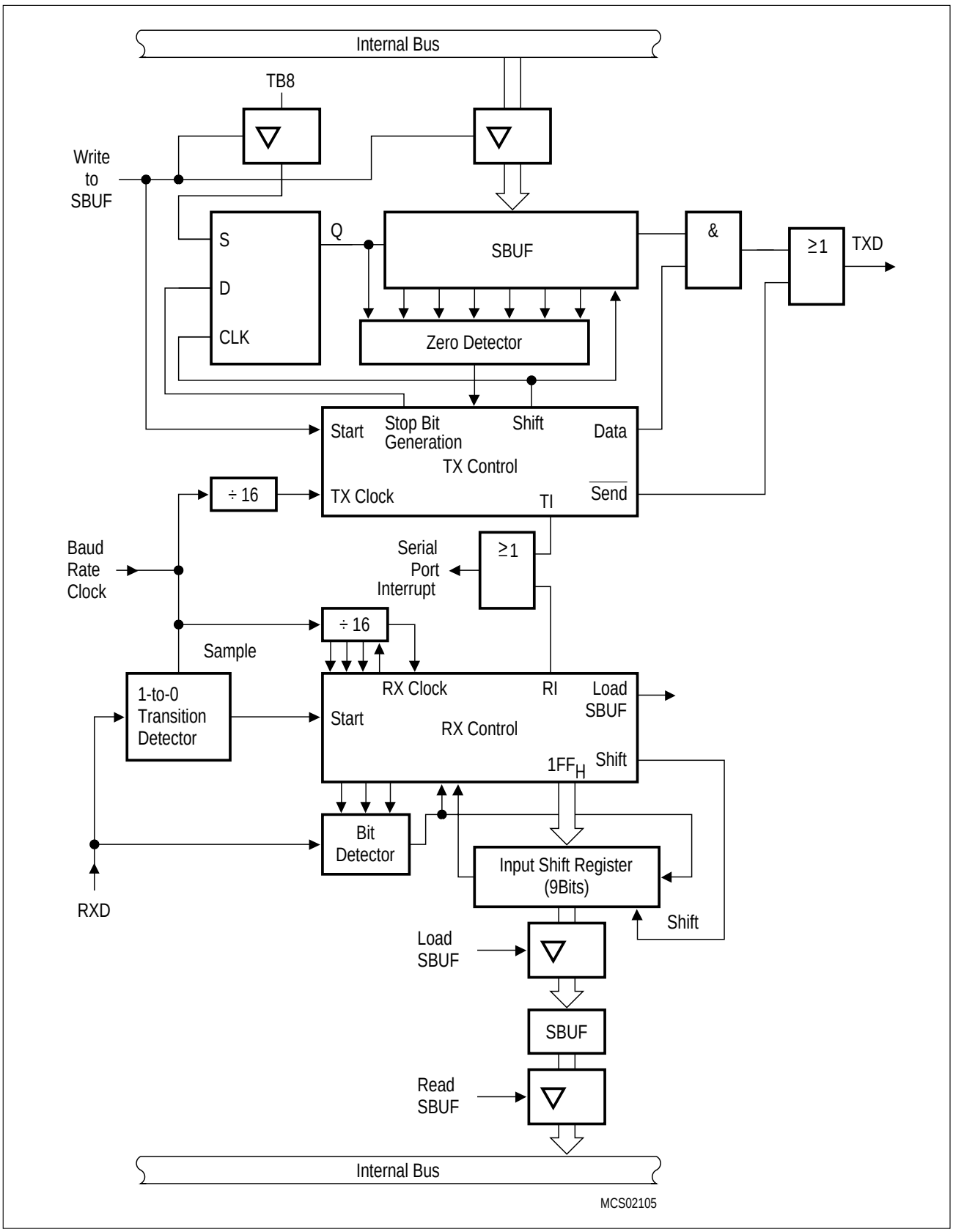


Figure 6-28
Serial Interface, Mode 2 and 3, Functional Diagram

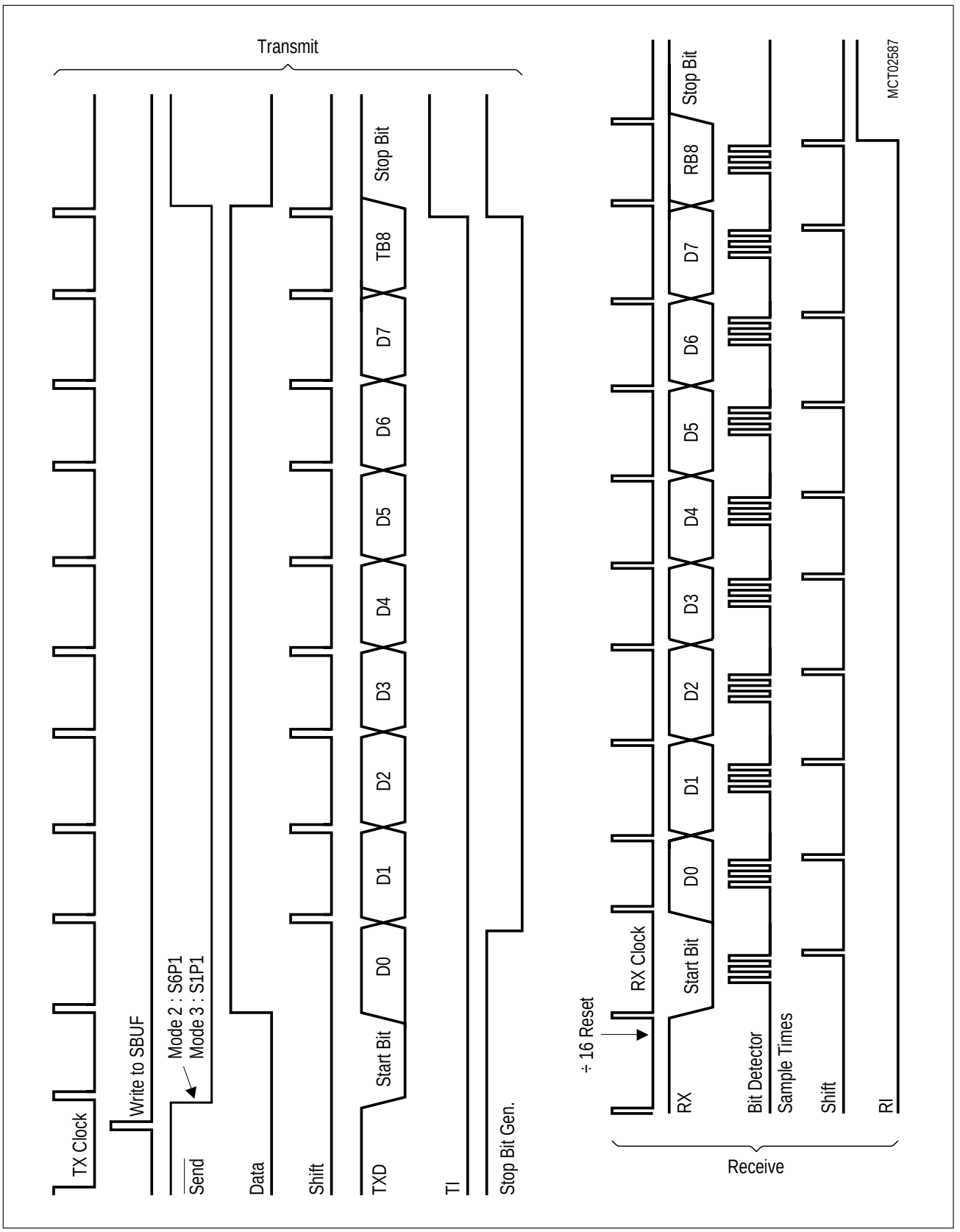


Figure 6-29
Serial Interface, Mode 2 and 3, Timing Diagram

6.4 The On-Chip CAN Controller

The Controller Area Network (CAN) bus with its associated protocol allows communication between a number of stations which are connected to this bus with high efficiency. Efficiency in this context means:

- Transfer speed (data rates of up to 1 Mbit/sec can be achieved)
- Data integrity (the CAN protocol provides several means for error checking)
- Host processor unloading (the controller here handles most of the tasks autonomously)
- Flexible and powerful message passing (the extended CAN protocol is supported)

Note: The CAN interface is a part of the C505C derivatives only.

The CAN interface which is integrated in the C505C is functionally fully compatible with the CAN module which is available in the 8-bit microcontroller C515C and in the 16-bit microcontroller C167CR. The CAN module of the C167CR has been adapted with its internal bus interface, clock generation logic, register access control logic, and interrupt function to the requirements of the 8-bit C500 microcontroller architecture.

Generally, the CAN interface is made of two major blocks :

- The CAN controller
- The internal bus interface

The CAN controller is the functional heart which provides all resources that are required to run the standard CAN protocol (11-bit identifiers) as well as the extended CAN protocol (29-bit identifiers). It provides a sophisticated object layer to relieve the CPU of as much overhead as possible when controlling many different message objects (up to 15). This includes bus arbitration, resending of garbled messages, error handling, interrupt generation, etc. In order to implement the physical layer, external components have to be connected to the C505C.

The internal bus interface connects the on-chip CAN controller to the internal bus of the microcontroller. The registers and data locations of the CAN interface are mapped to a specific 256 byte wide address range of the external data memory area ($F700_H$ to $F7FF_H$) and can be accessed using MOVX instructions.

6.4.1 Basic CAN Controller Functions

The on-chip CAN controller combines several functional blocks (see **figure 6-30**) that work in parallel and contribute to the controller's performance. These units and the functions they provide are described below.

The CAN controller provides storage for up to 15 message objects of maximum 8 data bytes length. Each of these objects has a unique identifier and its own set of control and status bits. Each object can be configured with its direction as either transmit or receive, except the last message which is only a receive buffer with a special mask register.

An object with its direction set as transmit can be configured to be automatically sent whenever a remote frame with a matching identifier (taking into account the respective global mask register) is received over the CAN bus. By requesting the transmission of a message with the direction set as receive, a remote frame can be sent to request that the appropriate object be sent by some other node. Each object has separate transmit and receive interrupts and status bits, allowing the microcontroller full flexibility in detecting when a remote/data frame has been sent or received.

For general purpose two masks for acceptance filtering can be programmed, one for identifiers of 11 bits and one for identifiers of 29 bits. However the microcontroller must configure bit XTD (Normal or Extended Frame Identifier) in the message configuration register for each valid message to determine whether a standard or extended frame will be accepted.

The last message object has its own programmable mask for acceptance filtering, allowing a large number of infrequent objects to be handled by the system.

The object layer architecture of the CAN controller is designed to be as regular and orthogonal as possible. This makes it easy to use and small for implementation.

The message storage is implemented in an intelligent memory which can be addressed by the CAN controller and the microcontroller interface. The content of various bit fields in the object are used to perform the functions of acceptance filtering, transmit search, interrupt search and transfer completion. It can be filled with up to 15 messages of 8 bytes data.

The CAN controller offers significantly improved status information over earlier versions, enabling a much easier diagnosis of the state of the network.

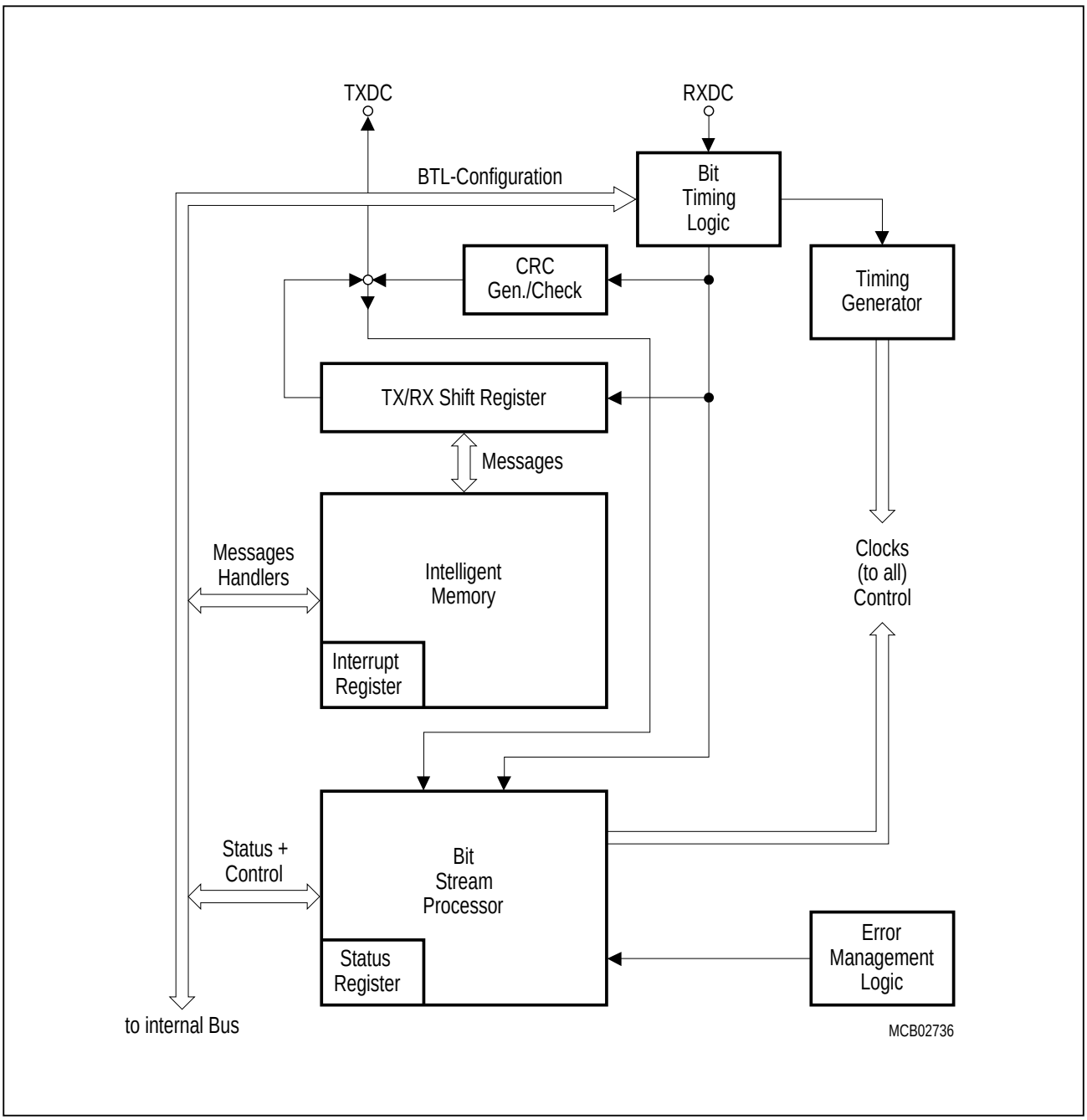


Figure 6-30
CAN Controller Block Diagram

TX/RX Shift Register

The Transmit / Receive Shift Register holds the destuffed bit stream from the bus line to allow the parallel access to the whole data or remote frame for the acceptance match test and the parallel transfer of the frame to and from the Intelligent Memory.

Bit Stream Processor (BSP)

The Bit Stream Processor is a sequencer controlling the sequential data stream between the TX/RX Shift Register, the CRC Register, and the bus line. The BSP also controls the EML and the parallel data stream between the TX/RX Shift Register and the Intelligent Memory such that the processes of reception, arbitration, transmission, and error signalling are performed according to the CAN protocol. Note that the automatic retransmission of messages which have been corrupted by noise or other external error conditions on the bus line is handled by the BSP.

Cyclic Redundancy Check Register (CRC)

This register generates the Cyclic Redundancy Check code to be transmitted after the data bytes and checks the CRC code of incoming messages. This is done by dividing the data stream by the code generator polynomial.

Error Management Logic (EML)

The Error Management Logic is responsible for the fault confinement of the CAN device. Its counters, the Receive Error Counter and the Transmit Error Counter, are incremented and decremented by commands from the Bit Stream Processor. According to the values of the error counters, the CAN controller is set into the states error *active*, error *passive* and busoff.

The CAN controller is *error active*, if both error counters are below the *error passive* limit of 128. It is *error passive*, if at least one of the error counters equals or exceeds 128.

It goes *busoff*, if the Transmit Error Counter equals or exceeds the *busoff* limit of 256. The device remains in this state, until the busoff recovery sequence is finished.

Additionally, there is the bit EWRN in the Status Register, which is set, if at least one of the error counters equals or exceeds the error warning limit of 96. EWRN is reset, if both error counters are less than the error warning limit.

Bit Timing Logic (BTL)

This block monitors the busline input RXDC and handles the busline related bit timing according to the CAN protocol.

The BTL synchronizes on a *recessive* to *dominant* busline transition at *Start of Frame* (hard synchronization) and on any further *recessive* to *dominant* busline transition, if the CAN controller itself does not transmit a *dominant* bit (resynchronization).

The BTL also provides programmable time segments to compensate for the propagation delay time and for phase shifts and to define the position of the Sample Point in the bit time. The programming of the BTL depends on the baudrate and on external physical delay times.

Intelligent Memory

The Intelligent Memory (CAM/RAM array) provides storage for up to 15 message objects of maximum 8 data bytes length. Each of these objects has a unique identifier and its own set of control and status bits. After the initial configuration, the Intelligent Memory can handle the reception and transmission of data without further microcontroller actions.

Organization of Registers and Message Objects

All registers and message objects of the CAN controller are located in the CAN address area of 256 bytes, which is mapped into the external data memory area of the C505C.

6.4.2 CAN Register Description

Notational Conventions

Each CAN register is described with its bit symbols, address, reset value, and a functional description of each bit or bitfield. Also the access type is indicated for each bit or bitfield :

r : for read access
w : for write access
rw : for read and write access

After reset the CAN registers either contain a defined reset value, keep their previous contents (UU_H), or are undefined (XX_H). Locations that are unchanged (UU_H) after reset, of course are undefined (XX_H) after a power-on reset operation. The reset values are defined either in hex (with index $_H$) or binary (with index $_B$) expressions.

The notation “n“ in the address definition of the message object registers defines the number of the related message object (n=1-15).

6.4.2.1 General Registers

The general registers of the CAN controller are located at the external data memory location $F700_H$ to $F70F_H$. The registers of this general register block is shown in **figure 6-31**. The address mapping of the 16 registers/bytes of a message object is shown in **figure 6-32**.

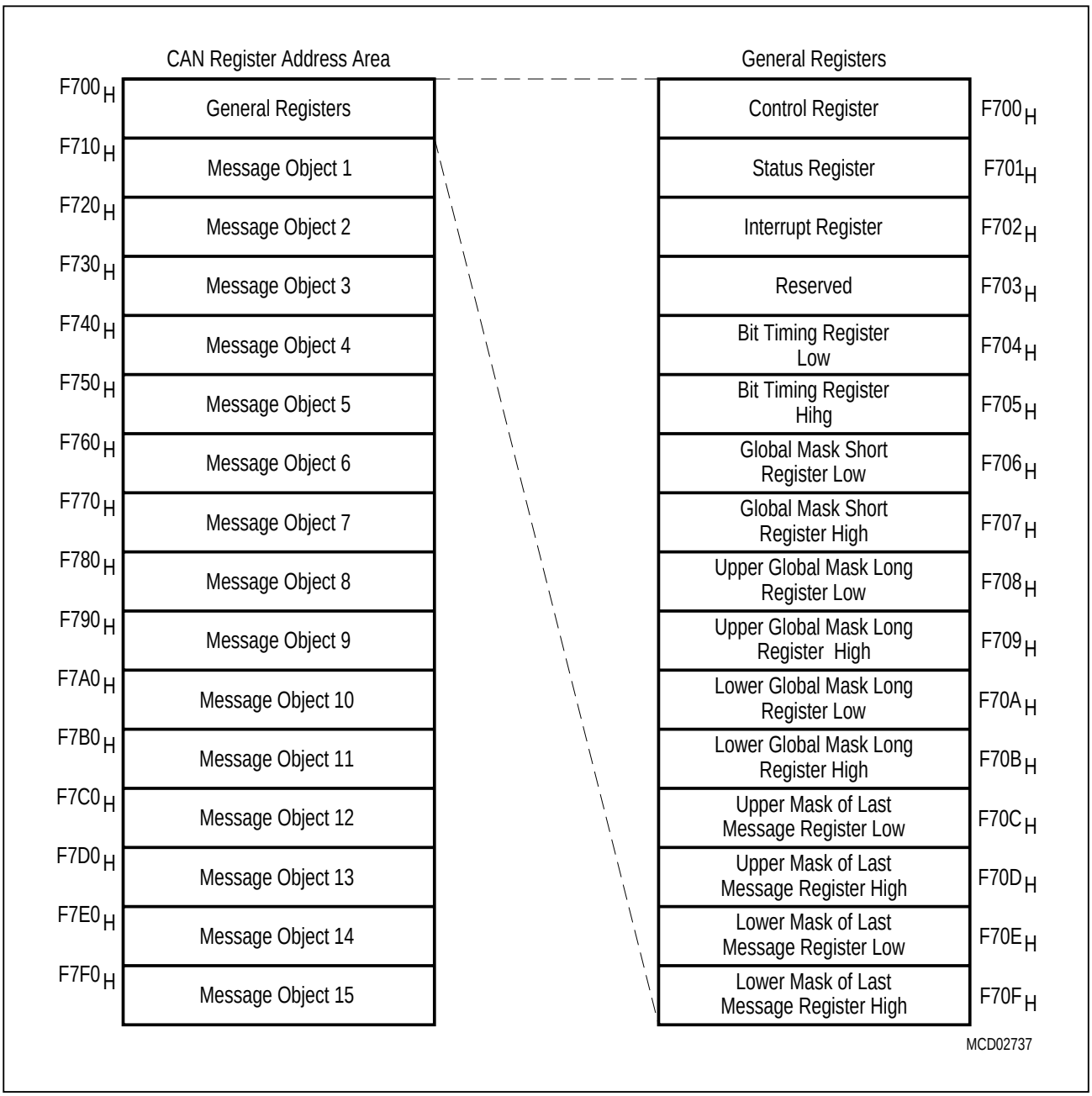


Figure 6-31
CAN Module Address Map

CAN Control Register CR (Address $F700_H$) **Reset Value : 01_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$F700_H$	TEST	CCE	0	0	EIE	SIE	IE	INIT	CR
	rw	rw	r	r	rw	rw	rw	rw	

Bit	Function
TEST	Test mode Make sure that bit 7 is cleared when writing to the control register, as this bit controls a special test mode, that is used for production testing. During normal operation, however, this test mode may lead to undesired behaviour of the device.
CCE	Configuration change enable Allows or inhibits microcontroller access to the bit timing register.
EIE	Error interrupt enable Enables or disables interrupt generation on a change of bit BOFF or EWRN in the status register.
SIE	Status change interrupt enable Enables or disables interrupt generation when a message transfer (reception or transmission) is successfully completed or a CAN bus error is detected (and registered in the status register).
IE	Interrupt enable Enables or disables interrupt generation from the CAN module to the interrupt controller of the C505C. Does not affect status updates. Additionally, bit ECAN in SFR IEN1 and bit EAL in SFR IEN0 must be set when a CAN controller interrupt should be generated.
INIT	Initialization Starts the initialization of the CAN controller, when set.

CAN Status Register SR (Address $F701_H$) **Reset Value : XX_H**

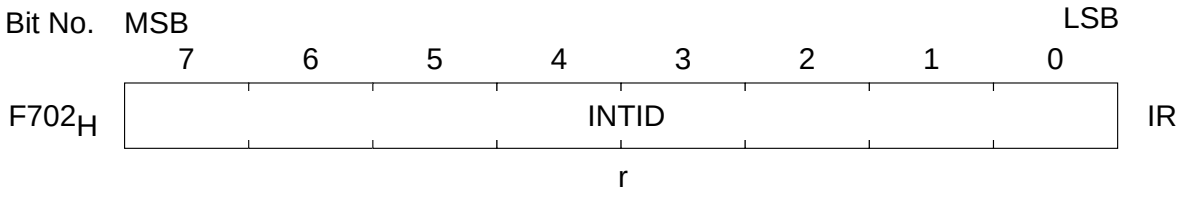
Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$F701_H$	BOFF	EWRN	–	RXOK	TXOK		LEC		SR
	r	r	r	rw	rw		rw		

Bit	Function
BOFF	Busoff status Indicates when the CAN controller is in busoff state (see EML).
EWRN	Error warning status Indicates that at least one of the error counters in the EML has reached the error warning limit of 96.
RXOK	Received message successfully Indicates that a message has been received successfully, since this bit was last reset by the CPU (the CAN controller does not reset this bit!).
TXOK	Transmitted message successfully Indicates that a message has been transmitted successfully (error free and acknowledged by at least one other node), since this bit was last reset by the CPU (the CAN controller does not reset this bit!).

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>LEC</td><td>Last error code
This field holds a code which indicates the type of the last error occurred on the CAN bus. If a message has been transferred (reception or transmission) without error, this field will be cleared. Code “7” is unused and may be written by the microcontroller to check for updates.
<table>
<tr><th colspan="3">LEC2-0</th><th>Error</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>No Error</td><td>–</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Stuff Error</td><td>More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Form Error</td><td>A fixed format part of a received frame has the wrong format</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Ack Error</td><td>The message this CAN controller transmitted was not acknowledged by another node.</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Bit1 Error</td><td>During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (“1”), but the monitored bus value was dominant.</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Bit0 Error</td><td>During the transmission of a message (or acknowledge bit, active error flag, or overload flag), the device wanted to send a dominant level (“0”), but the monitored bus value was recessive. During busoff recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the microcontroller to monitor the proceeding of the busoff recovery sequence (indicating the bus is not stuck at dominant or continously disturbed).</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>CRC Error</td><td>The CRC check sum was incorrect in the message received.</td></tr>
</table></td></tr>
</table>

Note : Reading the SR when an interrupt is pending, resets the pending interrupt request. (please see section 6.4.6 for further details about CAN interrupt handling.

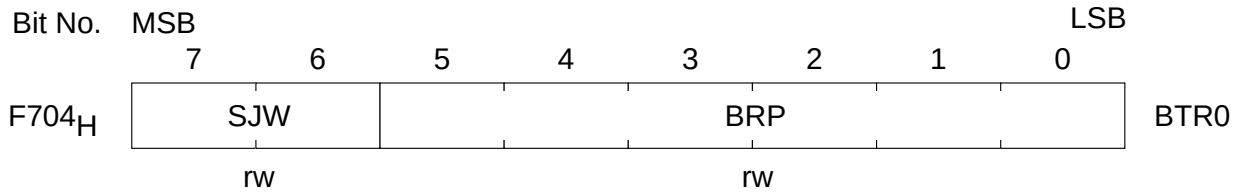
CAN Interrupt Register IR (Address $F702_H$) **Reset Value : XX_H**



Bit	Function
INTID	Interrupt identifier This number indicates the cause of the interrupt. When no interrupt is pending, the value will be “00”.

See also section 6.4.6 with **table 6-7** for further details about the CAN controller interrupt handling.

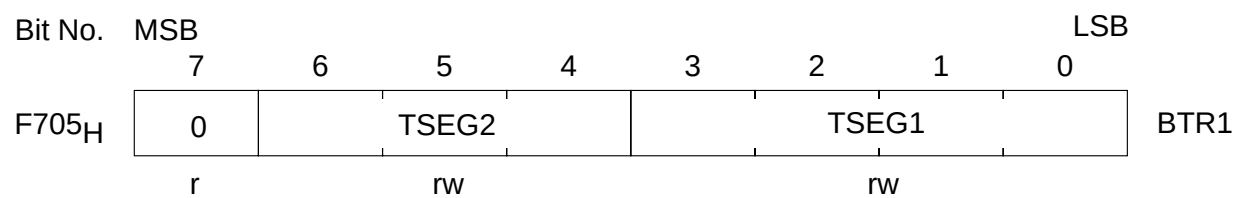
CAN Bit Timing Register Low BTR0 (Address $F704_H$) **Reset Value : UU_H**



Bit	Function
SJW	(Re)Synchronization jump width Adjust the bit time by (SJW+1) time quanta for resynchronization.
BRP	Baud rate prescaler For generating the bit time quanta the oscillator frequency is divided by (BRP+1).

Note : This register can only be written, if the configuration change enable bit (CCE) is set.

CAN Bit Timing Register High BTR1 (Address $F705_H$) **Reset Value : $0UUUUUUU_B$**



Bit	Function
TSEG2	Time segment after sample point There are (TSEG2+1) time quanta after the sample point. Valid values for TSEG2 are “1...7”.
TSEG1	Time segment before sample point There are (TSEG1+1) time quanta before the sample point. Valid values for TSEG1 are “2...15”.

Note : This register can only be written, if the configuration change enable bit (CCE) is set.

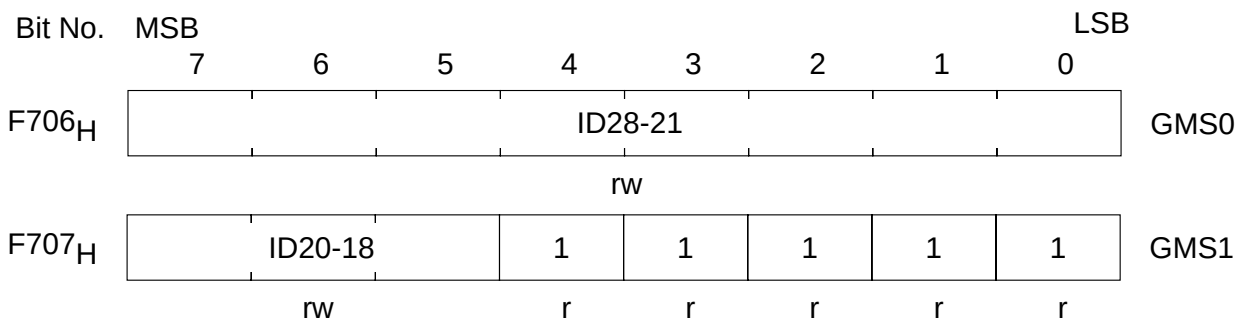
Mask Registers

Messages can use standard or extended identifiers. Incoming frames are masked with their appropriate global masks. Bit IDE of the incoming message determines, if the standard 11-bit mask in global mask short is to be used, or the 29-bit extended mask in global mask long. Bits holding a “0” mean “don’t care”, ie. do not compare the message’s identifier in the respective bit position.

The last message object (15) has an additional individually programmable acceptance mask (mask of last message) for the complete arbitration field. This allows classes of messages to be received in this object by masking some bits of the identifier.

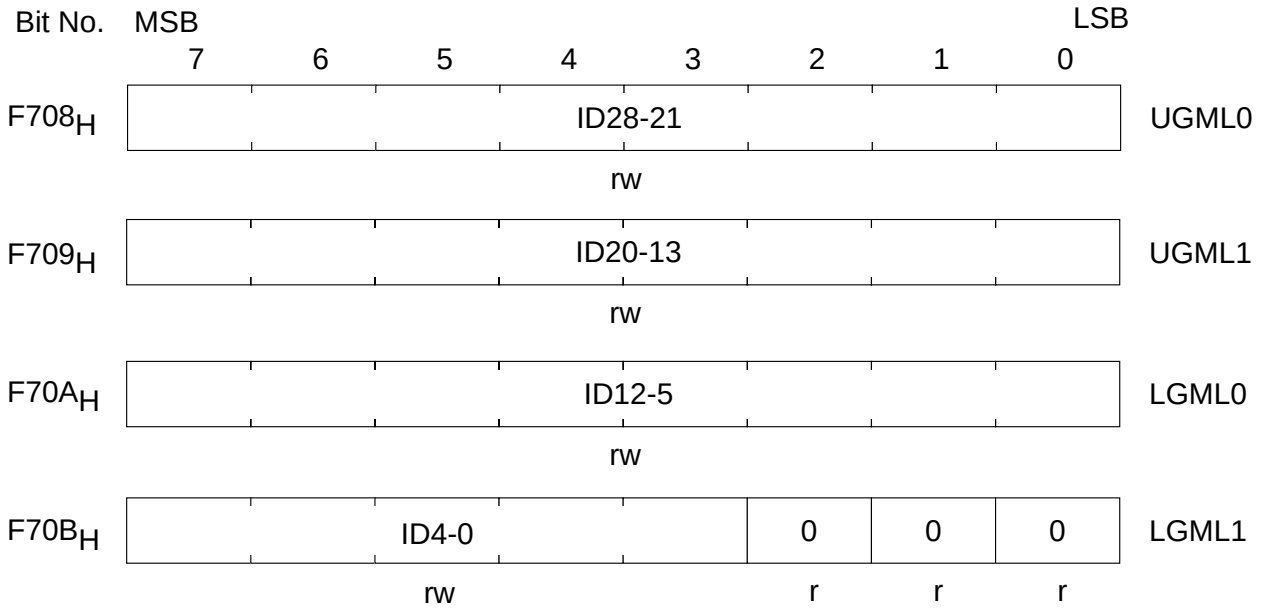
Note : The mask of last message is ANDed with the global mask that corresponds to the incoming message.

CAN Global Mask Short Register Low GMS0 (Address $F706_H$) **Reset Value : UU_H**
CAN Global Mask Short Register High GMS1 (Address $F707_H$) **Reset Value : $UUU11111_B$**



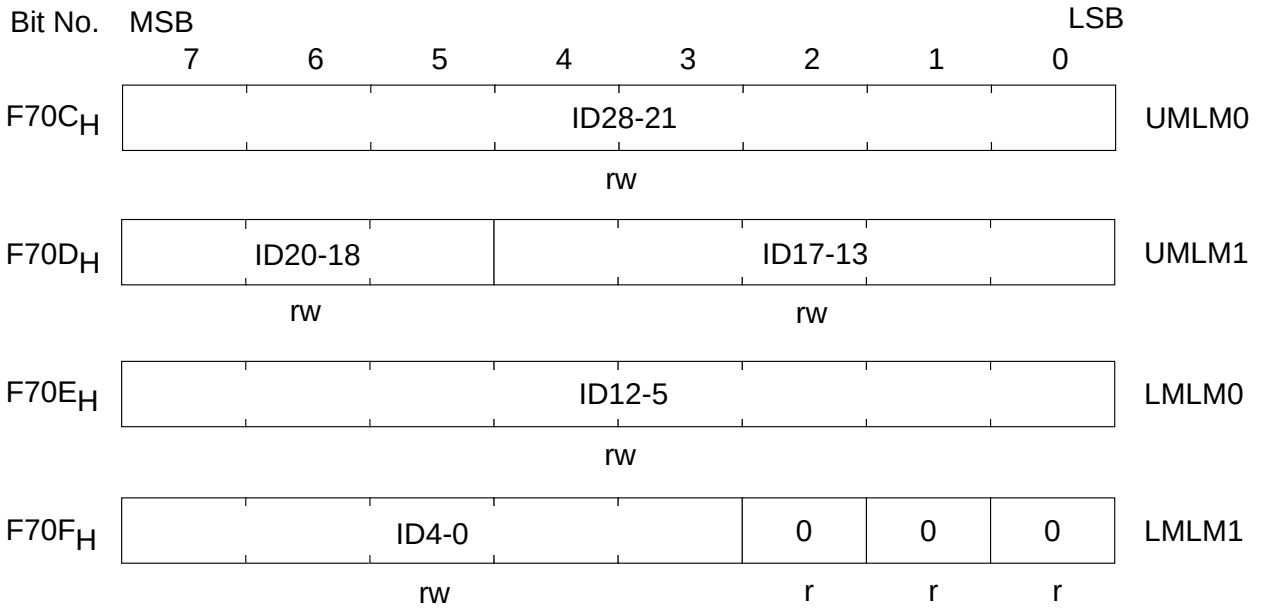
Bit	Function
ID28-18	Identifier (11-bit) Mask to filter incoming messages with standard identifier.

CAN Upper Global Mask Long Register Low UGML0 (Addr. $F708_H$) **Reset Value : UU_H**
CAN Upper Global Mask Long Register High UGML1 (Addr. $F709_H$) **Reset Value : UU_H**
CAN Lower Global Mask Long Register Low LGML0 (Addr. $F70A_H$) **Reset Value : UU_H**
CAN Lower Global Mask Long Register High LGML1 (Addr. $F70B_H$) **Reset Val. $UUUUU000_B$**



Bit	Function
ID28-0	Identifier (29-bit) Mask to filter incoming messages with extended identifier.

CAN Upper Mask of Last Message Register Low UMLM0 (Addr. F70C$_H$) Reset Value : UU$_H$
CAN Upper Mask of Last Message Register High UMLM1 (Addr. F70D$_H$) Reset Value : UU$_H$
CAN Lower Mask of Last Message Register Low LMLM0 (Addr. F70E$_H$) Reset Value : UU$_H$
CAN Lower Mask of Last Message Reg. High LMLM1 (Addr. F70F$_H$) Reset Val.: UUUUU000$_B$



Bit	Function
ID28-0	Identifier (29-bit) Mask to filter the last incoming message (no. 15) with standard or extended identifier (as configured).

6.4.2.2 The Message Object Registers / Data Bytes

The message object is the primary means of communication between microcontroller and CAN controller. Each of the 15 message objects uses 15 consecutive bytes (see **figure 6-32**) and starts at an address that is a multiple of 16 (message object n base address; N = 1 to F_H).

Note : All message objects must be initialized by the C505C, even those which are not going to be used, before clearing the INIT bit.

	Offset
Message Control Reg. Low	+0
Message Control Reg. High	+1
Upper Arbitration Reg. Low	+2
Upper Arbitration Reg. High	+3
Lower Arbitration Reg. Low	+4
Lower Arbitration Reg. High	+5
Message Configuration Reg.	+6
Data Byte 0	+7
Data Byte 1	+8
Data Byte 2	+9
Data Byte 3	+10
Data Byte 4	+11
Data Byte 5	+12
Data Byte 6	+13
Data Byte 7	+14
Reserved	+15

Register Address Calculation:

Message Object n Register Address =
Message Object n Base Address + Offset

(see also **Figure 6-31**)

Message Object n (n=1 to F_H)	Message Object n Base Address
1	$F710_H$
2	$F720_H$
3	$F730_H$
4	$F740_H$
5	$F750_H$
6	$F760_H$
7	$F770_H$
8	$F780_H$
9	$F790_H$
A_H	$F7A0_H$
B_H	$F7B0_H$
C_H	$F7C0_H$
D_H	$F7D0_H$
E_H	$F7E0_H$
F_H	$F7F0_H$

MCD02738

Figure 6-32
Message Object Address Map

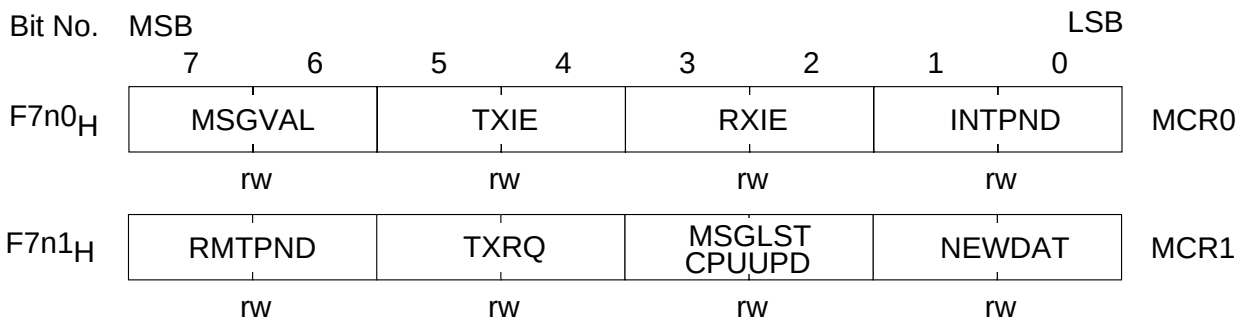
Each element of the message control register is made of two complementary bits. This special mechanism allows to selectively set or reset specific elements (leaving others unchanged) without requiring read-modify-write cycles. None of these elements will be affected by reset.

Table 6-6 below shows how to use and interpret these 2-bit fields.

Table 6-6 : Set/Reset Bits

Value of the 2-bit Field		Function on Write	Meaning on Read
0	**0**	reserved	reserved
0	**1**	Reset element	Element is reset
1	**0**	Set element	Element is set
1	**1**	Leave element unchanged	reserved

CAN Message Control Register Low MCR0 (Address F7n0$_H$) **Reset Value : UU$_H$**
CAN Message Control Register High MCR1 (Address F7n1$_H$) **Reset Value : UU$_H$**



Bit	Function
MSGVAL	Message valid Indicates, if the corresponding message object is valid or not. The CAN controller only operates on valid objects. Message objects can be tagged invalid, while they are changed, or if they are not used at all.
TXIE	Transmit interrupt enable Defines, if bit INTPND is set after successful transmission of a frame. [1)]
RXIE	Receive interrupt enable Defines, if bit INTPND is set after successful reception of a frame.
INTPND	Interrupt pending Indicates, if this message object has generated an interrupt request (see TXIE and RXIE), since this bit was last reset by the microcontroller, or not.

Bit	Function
RMTPND	Remote pending (used for transmit-objects) Indicates that the transmission of this message object has been requested by a remote node, but the data has not yet been transmitted. When RMTPND is set, the CAN controller also sets TXRQ. RMTPND and TXRQ are cleared, when the message object has been successfully transmitted.
TXRQ	Transmit request Indicates that the transmission of this message object is requested by the CPU or via a remote frame and is not yet done. TXRQ can be disabled by CPUUPD. 1) 3)
MSGLST	Message lost (this bit applies to receive-objects only!) Indicates that the CAN controller has stored a new message into this object, while NEWDAT was still set, ie. the previously stored message is lost.
CPUUPD	CPU update (this bit applies to transmit-objects only!) Indicates that the corresponding message object may not be transmitted now. The microcontroller sets this bit in order to inhibit the transmission of a message that is currently updated, or to control the automatic response to remote requests.
NEWDAT	New data Indicates, if new data has been written into the data portion of this message object by microcontroller (transmit-objects) or CAN controller (receive-objects) since this bit was last reset, or not. 2)

1) In message object 15 (last message) these bits are hardwired to “0” (inactive) in order to prevent transmission of message 15.

2) When the CAN controller writes new data into the message object, unused message bytes will be overwritten by non specified values. Usually the microcontroller will clear this bit before working on the data, and verify that the bit is still cleared once it has finished working to ensure that it has worked on a consistent set of data and not part of an old message and part of the new message.
For transmit-objects the microcontroller will set this bit along with clearing bit CPUUPD. This will ensure that, if the message is actually being transmitted during the time the message was being updated by the microcontroller, the CAN controller will not reset bit TXRQ. In this way bit TXRQ is only reset once the actual data has been transferred.

3) When the microcontroller requests the transmission of a receive-object, a remote frame will be sent instead of a data frame to request a remote node to send the corresponding data frame. This bit will be cleared by the CAN controller along with bit RMTPND when the message has been successfully transmitted, if bit NEWDAT has not been set. If there are several valid message objects with pending transmission request, the message with the lowest message number is transmitted first.

Arbitration Registers

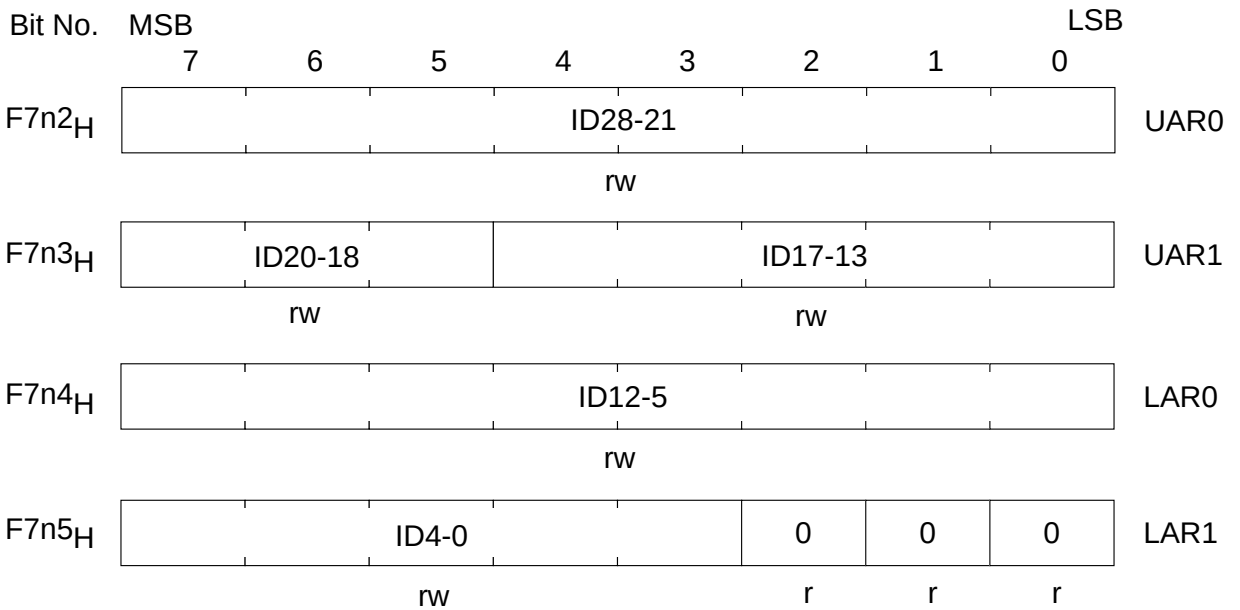
The arbitration registers are used for acceptance filtering of incoming messages and to define the identifier of outgoing messages. A received message is stored into the valid message object with a matching identifier and DIR=”0” (data frame) or DIR=”1” (remote frame). Extended frames can be stored only in message objects with XTD=”1”, standard frames only in message objects with XTD=”0”. For matching, the corresponding global mask has to be considered (in case of message object 15 also the mask of last message). If a received message (data frame or remote frame) matches with more than one valid message object, it is stored into that with the lowest message number.

When the CAN controller stores a data frame, not only the data bytes, but the whole identifier and the data length code are stored into the corresponding message object (standard identifiers have bits ID17...0 filled with “0”). This is implemented to keep the data bytes connected with the identifier, even if arbitration mask registers are used. When the CAN controller stores a remote frame, only the data length code is stored into the corresponding message object. The identifier and the data bytes remain unchanged.

There must not be more than one valid message object with a particular identifier at any time. If some bits are masked by the global mask registers (ie. “don’t care”), then the identifiers of the valid message objects must differ in the remaining bits which are used for acceptance filtering.

If a received data frame is stored into a message object, the identifier of this message object is updated. If some of the identifier bits are set to “don’t care” by the corresponding mask register, these bits may be changed in the message object. If a remote frame is received, the identifier in transmit-object remain unchanged, except for the last message object (which cannot start a transmission). Here, the identifier bits corresponding to the “don’t care” bits of the last message object’s mask may be overwritten by the incoming message.

CAN Upper Arbitration Register Low UAR0 (Address F7n2$_H$) **Reset Value : UU$_H$**
CAN Upper Arbitration Register High UAR1 (Address F7n3$_H$) **Reset Value : UU$_H$**
CAN Lower Arbitration Register Low LAR0 (Address F7n4$_H$) **Reset Value : UU$_H$**
CAN Lower Arbitration Register High LAR1 (Address F7n5$_H$) **Reset Value : UUUUU000$_B$**



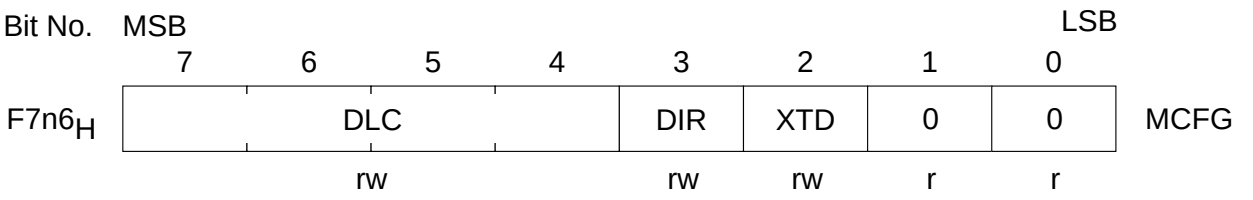
Bit	Function
ID28-0	Identifier (29-bit) Identifier of a standard message (ID28...18) or an extended message (ID28...0). For standard identifiers bits ID17...0 are “don’t care”.

Message Configuration and Data

The following fields hold a description of the message within this object. The data field occupies the following 8 byte positions after the message configuration register.

Note: There is no “don’t care” option for bits XTD and DIR. So incoming frames can only match with corresponding message objects, either standard (XTD=0) or extended (XTD=1). Data frames only match with receive-objects, remote frames only match with transmit-objects.
When the CAN controller stores a data frame, it will write all the eight data bytes into a message object. If the data length code was less than 8, the remaining bytes of the message object will be overwritten by non specified values.

CAN Message Configuration MCFG Register (Address $F7n6_H$) ResetValue:$UUUUUU00_B$



Bit	Function
DLC	Data length code Valid values for the data length are 0...8.
DIR	Message direction DIR=”1”: transmit. On TXRQ, the respective message object is transmitted. On reception of a remote frame with matching identifier, the TXRQ and RMTPND bits of this message object are set. DIR=”0”: receive. On TXRQ, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, that message is stored in this message object.
XTD	Extended identifier Indicates, if this message object will use an extended 29-bit identifier or a standard 11-bit identifier

CAN Data Bytes DB0-DB7 (Addresses $F7n7_H$-$F7nE_H$) **Reset Value : XX_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$F7n7_H$ - $F7nE_H$	.7	.6	.5	.4	.3	.2	.1	.0	DB0-7
	rw	rw	rw	rw	rw	rw	rw	rw	

Message data for message object 15 (last message) will be written into a two-message-alternating buffer to avoid the loss of a message, if a second message has been received, before the microcontroller has read the first one.

6.4.3 Handling of Message Objects

The following diagrams (**figures 6-33** to **6-38**) summarize the actions that have to be taken in order to transmit and receive messages over the CAN bus. The actions taken by the CAN controller are described as well as the actions that have to be taken by the microcontroller (i.e. the servicing program).

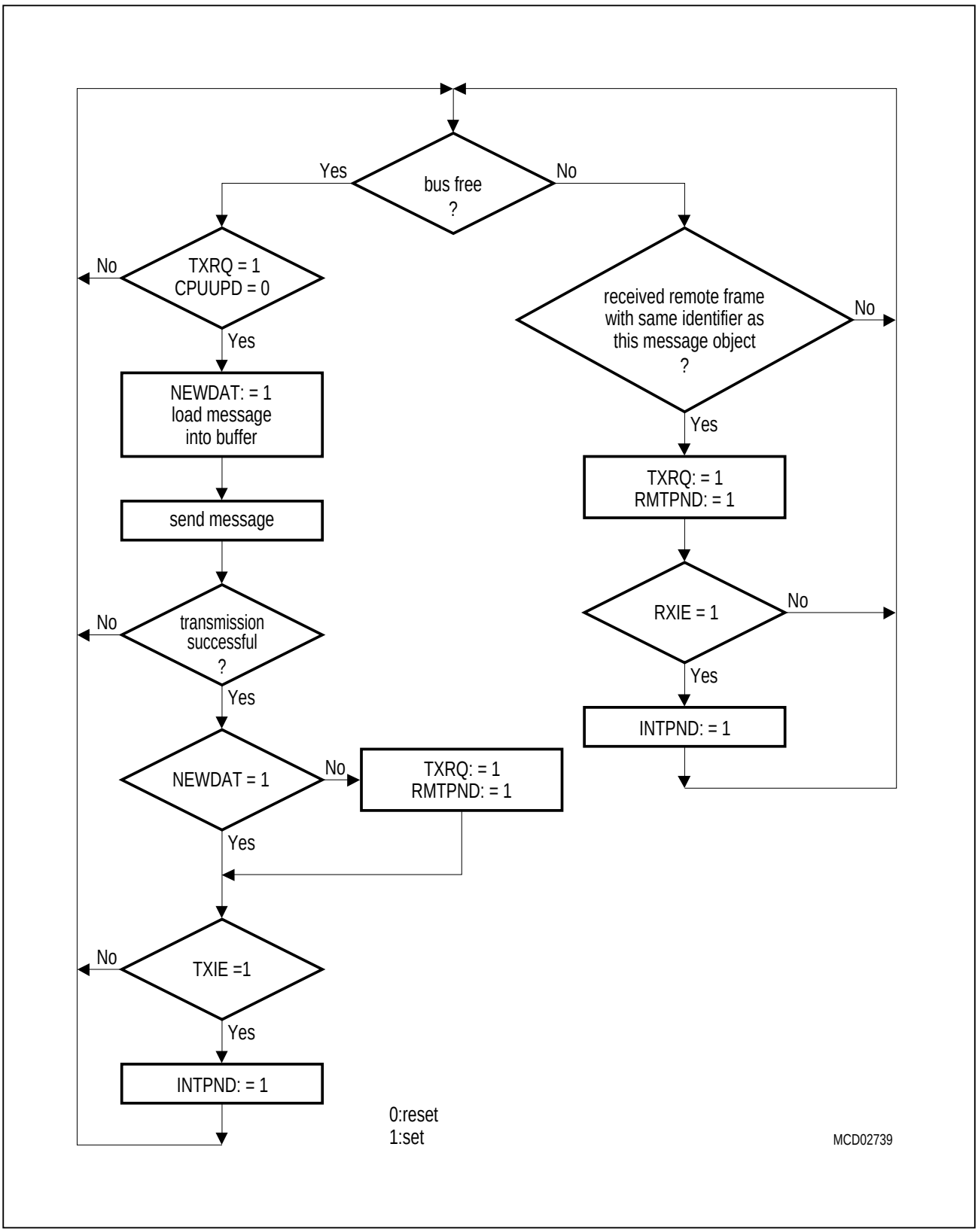


Figure 6-33
CAN Controller Handling of Message Objects with Direction = transmit

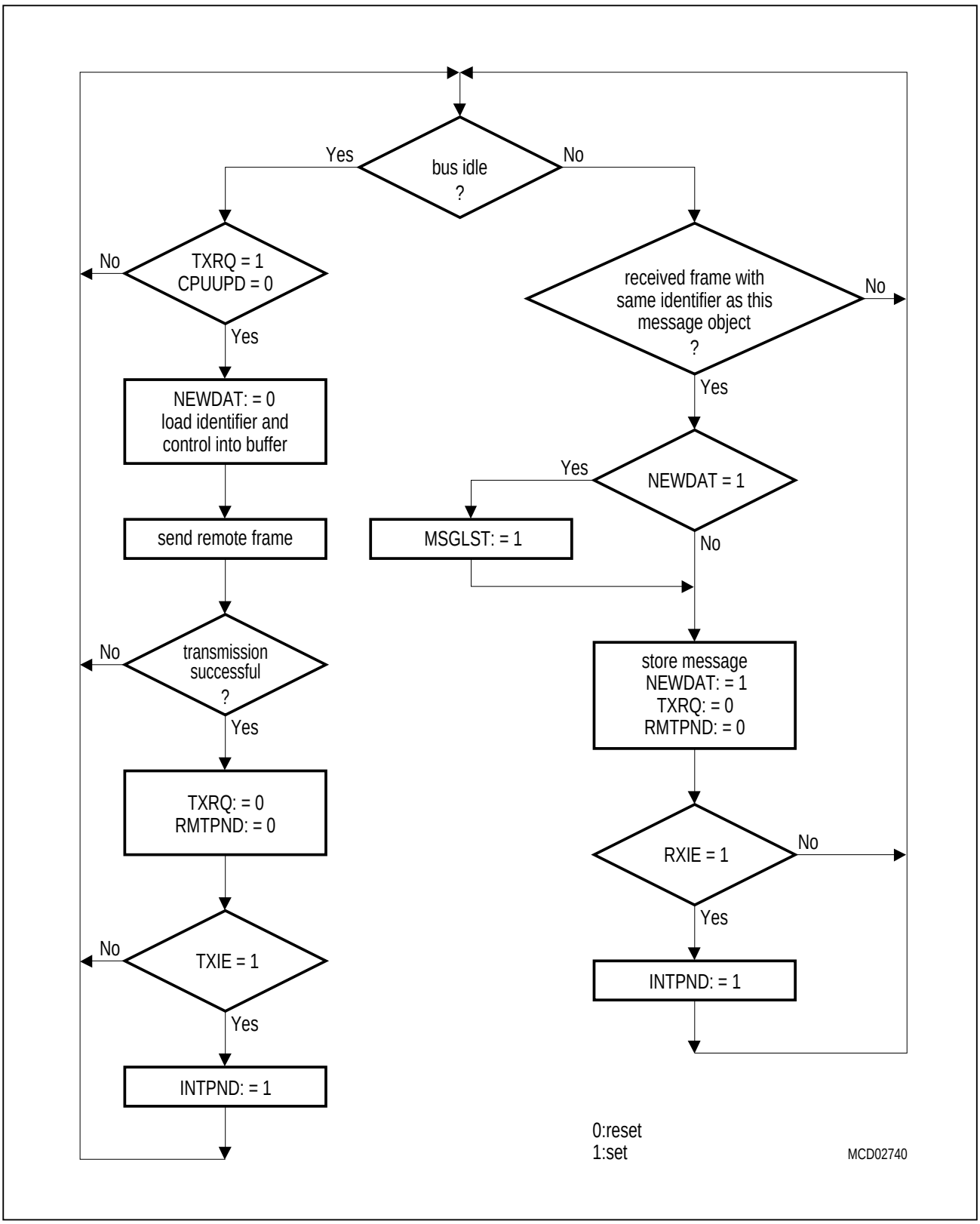


Figure 6-34
CAN Controller Handling of Message Objects with Direction = receive

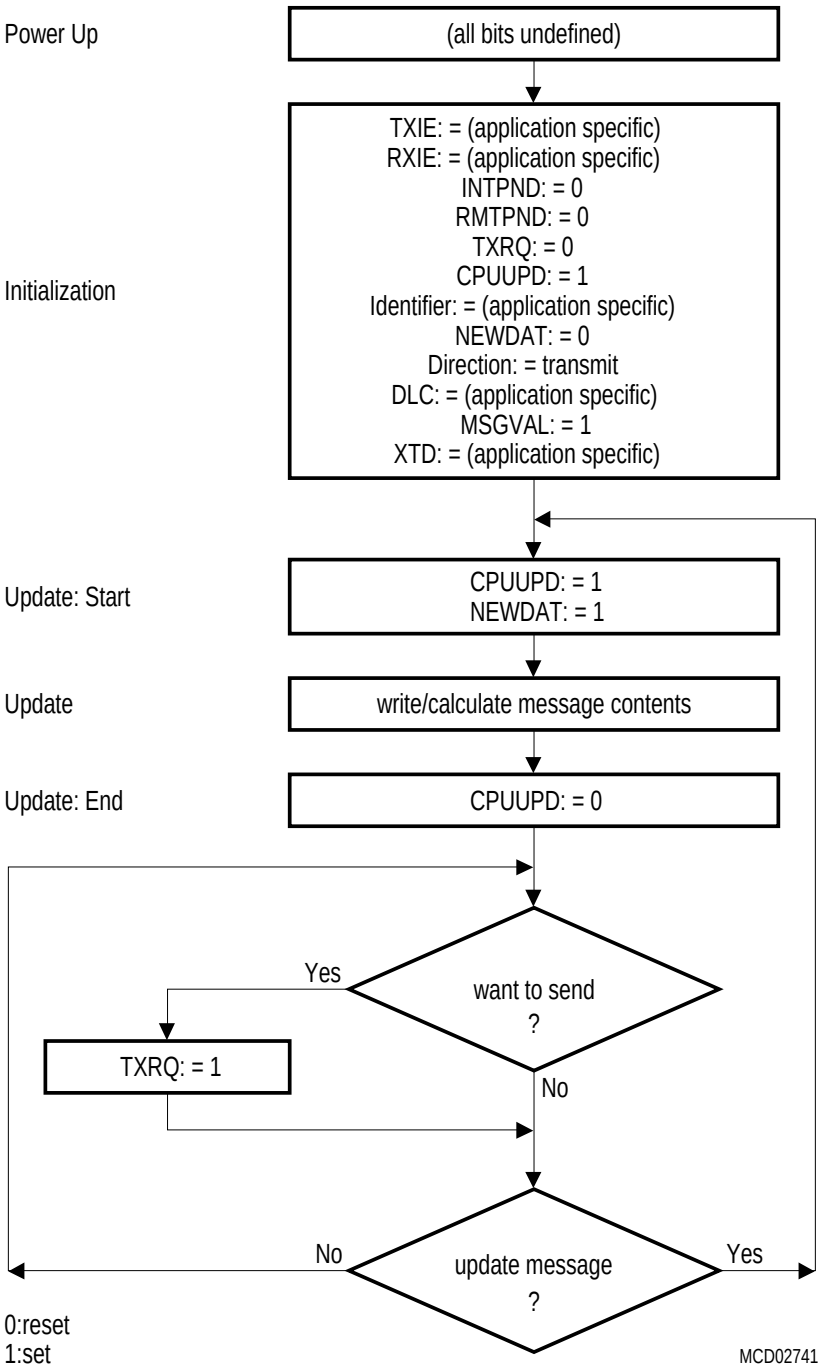


Figure 6-35
Microcontroller Handling of Message Objects with Direction = transmit

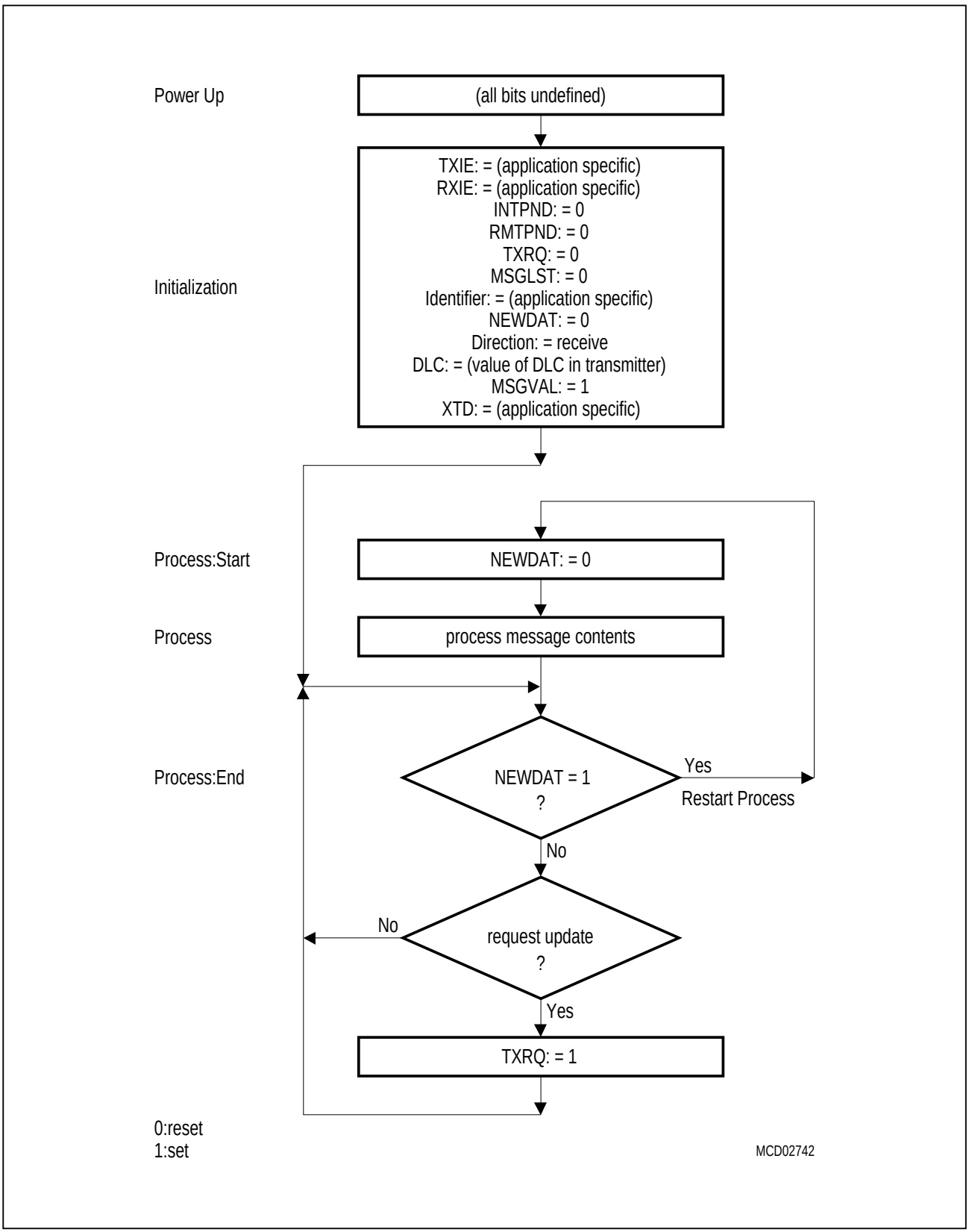


Figure 6-36
Microcontroller Handling of Message Objects with Direction = receive

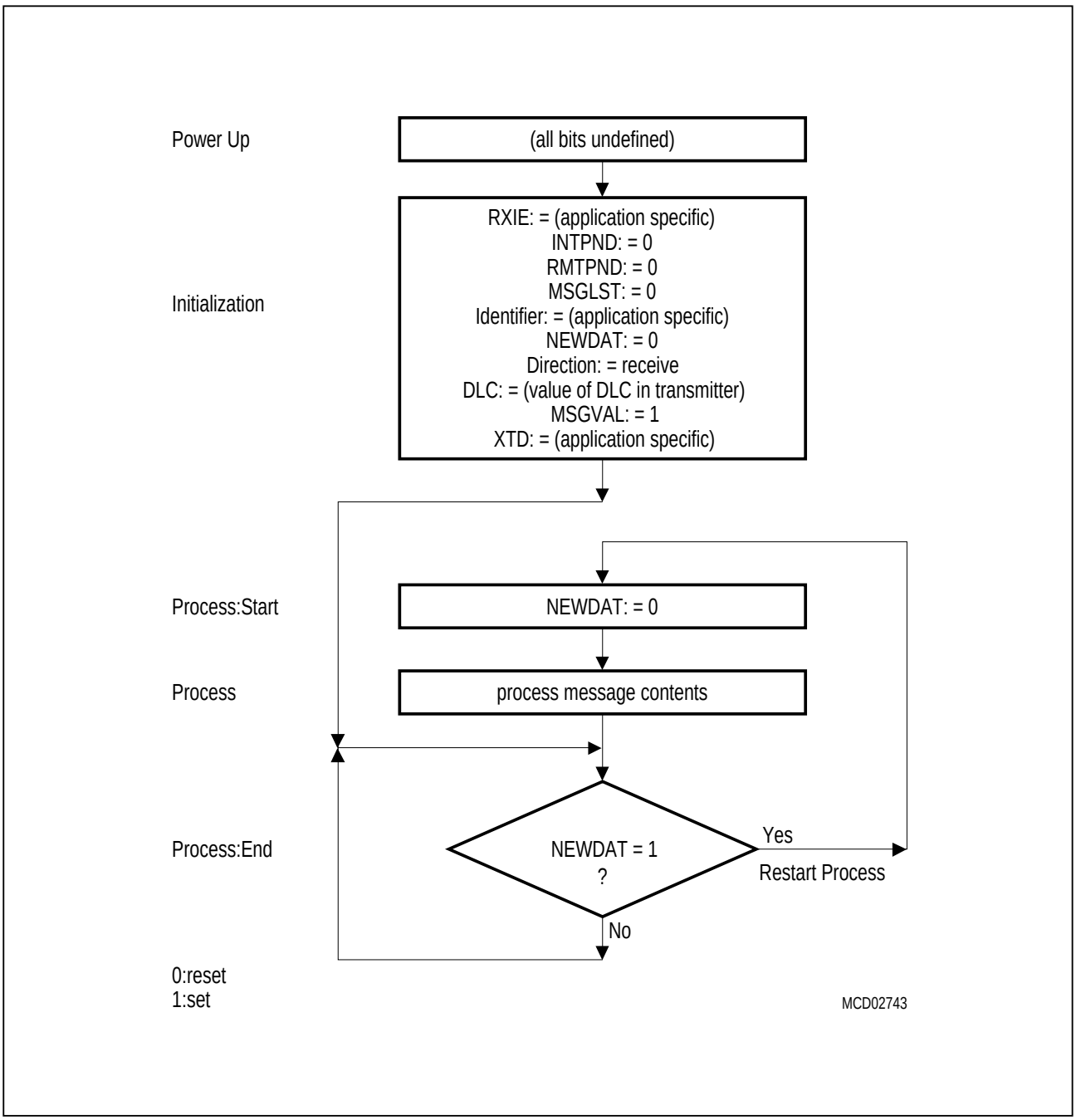


Figure 6-37
Microcontroller Handling of the Last Message Object

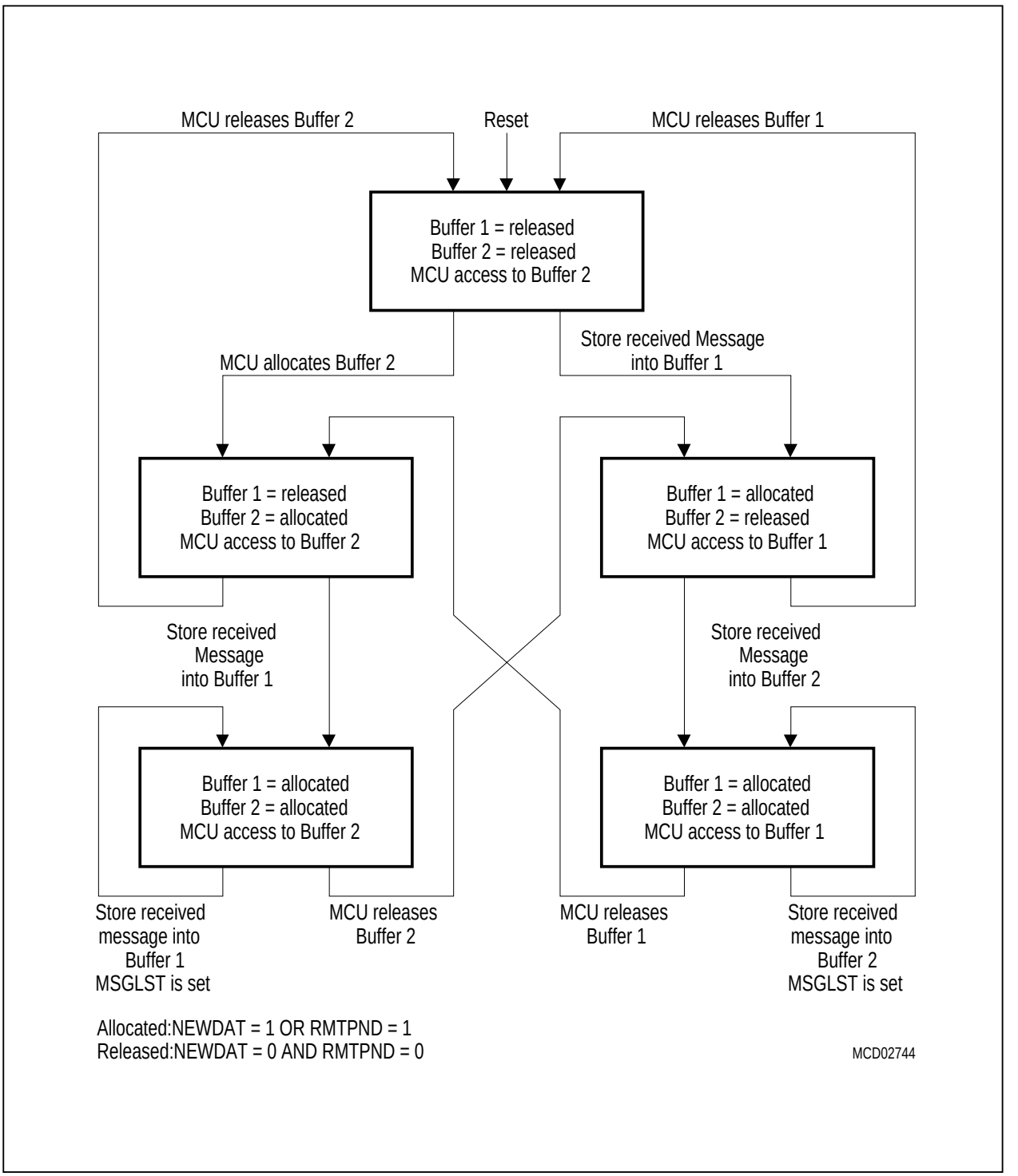


Figure 6-38
Microcontroller Handling of the Last Message Object's Alternating Buffer

6.4.4 Initialization and Reset

The CAN controller is reset by a hardware reset, the oscillator watchdog reset or by a watchdog timer reset of the C505C. A reset operation of the CAN controller performs the following actions :

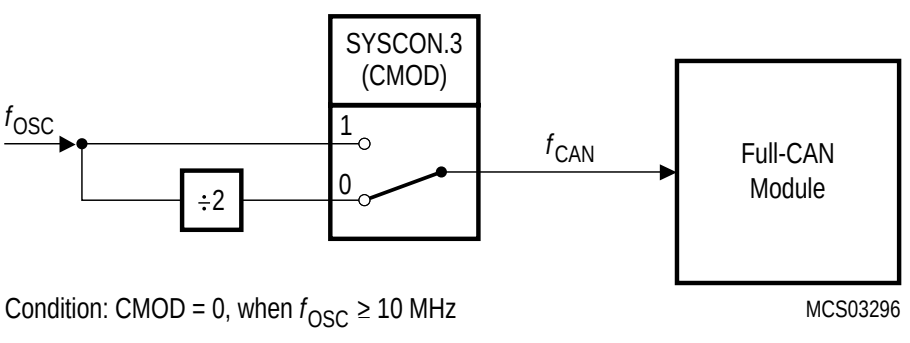
- sets the TXDC output to “1” (recessive)
- clears the error counters
- resets the busoff state
- switches the control register’s low byte to 01_H
- leaves the control register’s high byte and the interrupt register undefined
- does not change the other registers including the message objects (notified as UUUU)
- selects the prescaler for the CAN controller clock using the bit CMOD in SYSCON register.

The first hardware reset after power-on leaves the unchanged registers in an undefined state, of course. The value 01_H in the control register’s low byte prepares for software initialization.

Software Initialization

The very first step of the initialization is the CAN controller input clock selection. A ÷ 2 prescaler is enabled by default after reset (**figure 6-39**). Setting bit CMOD (SYSCON.3) disables the prescaler. The purpose of the prescaler selection is:
i) to ensure that the CAN controller is operable when f_{OSC} is over 10 MHz (bit CMOD =0)
ii) to achieve the maximum CAN baudrate of 1 Mbaud when f_{OSC} is 8 MHz (bit CMOD=1)

.



Frequency (MHz)		CMOD (SYSCON.3)	BRP (BTR0.0-5)	CAN baudrate (Mbaud/sec)
f_{OSC}	f_{CAN}			
8	8	1	000000_B	1
8	4	0	000000_B	0.5
16	8	0	000000_B	1

Note : The switch configuration shows the reset state of bit CMOD.

Figure 6-39
CAN controller input clock selection

Special Function Register SYSCON (Address $B1_H$) **Reset Value : $XX100X01_B$**

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

The functions of the shaded bits are not described here.

Bit	Function
CMOD	Prescaler selection for CAN controller Control bit for CAN controller input clock selection. The time quantum (t_q) of the CAN controller timing is affected by this (and hecne the baudrate). CMOD = 0 : The ÷ 2 prescaler is enabled (reset value). CMOD = 1 : The ÷ 2 prescaler is disabled. This bit must be cleared when f_{osc} is over 10 MHz.

The software Initialization is enabled by setting bit INIT in the control register. This can be done by the microcontroller via software, or automatically by the CAN controller on a hardware reset, or if the EML switches to busoff state.

While INIT is set

- all message transfer from and to the CAN bus is stopped
- the CAN bus output TXDC is “1” (recessive)
- the control bits NEWDAT and RMTPND of the last message object are reset
- the counters of the EML are left unchanged.

Setting bit CCE in addition, allows to change the configuration in the bit timing register.

For initialization of the CAN Controller, the following actions are required:

- configure the bit timing register (CCE required)
- set the Global Mask Registers
- initialize each message object.

If a message object is not needed, it is sufficient to clear its message valid bit (MSGVAL), ie. to define it as not valid. Otherwise, the whole message object has to be initialized.

After the initialization sequence has been completed, the microcontroller clears the INIT bit.

Now the BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (ie. bus idle) before it can take part in bus activities and start message transfers.

The initialization of the message objects is independent of the state of bit INIT and can be done on the fly, the message objects should all be configured to particular identifiers or set to not valid before the BSP starts the message transfer, however.

To change the configuration of a message object during normal operation, the microcontroller first clears bit MSGVAL, which defines it as not valid. When the configuration is completed, MSGVAL is set again.

Note that the busoff recovery sequence cannot be shortened by setting or resetting INIT. If the device goes busoff, it will set INIT of its own accord, stopping all bus activities. Once INIT has been cleared by the microcontroller, the device will then wait for 129 occurrences of Bus Idle before resuming normal operation. At the end of the busoff recovery sequence, the error management counters will be reset.

During the waiting time after the resetting of INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the control register, enabling the microcontroller to check up whether the CAN bus is stuck at dominant or continously disturbed and to monitor the proceeding of the busoff recovery sequence.

6.4.5 Configuration of the Bit Timing

According to the CAN specification, a bit time is subdivided into four segments (see **figure 6-40**). Each segment is a multiple of the time quantum t_q. The synchronization segment (Sync-Seg) is always one t_q long. The propagation time segment and the phase buffer segment1 (combined to Tseg1) defines the time before the sample point, while phase buffer segment2 (Tseg2) defines the time after the sample point. The length of these segments is programmable (except Sync-Seg).

Note : For exact definition of these segments please refer to the CAN specification.

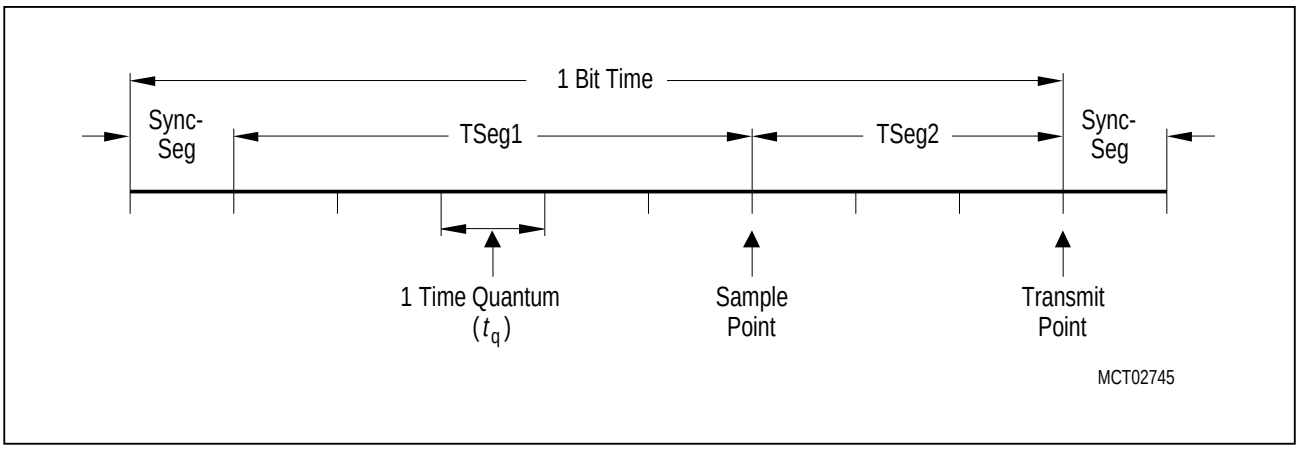


Figure 6-40
Bit Timing Definition

The bit time is determined by the C505C clock period CLP (see AC characteristics), the Baud Rate Prescaler, and the number of time quanta per bit:

bit time	$= t_{Sync\text{-}Seg} + t_{TSeg1} + t_{TSeg2}$	
$t_{Sync\text{-}Seg}$	$= 1 \times t_q$	
t_{TSeg1}	$= (TSEG1 + 1) \times t_q$	$(= \text{min. } 4 \times t_q)$
t_{TSeg2}	$= (TSEG2 + 1) \times t_q$	$(= \text{min. } 3 \times t_q)$
t_q	$= (BRP + 1) \times 2^{(1 - CMOD)} \times CLP$	

TSEG1, TSEG2, and BRP are the programmed numerical values from the respective fields of the Bit Timing Register and the bit CMOD in the SYSCON register (bit 3).

6.4.5.1 Hard Synchronization and Resynchronization

To compensate phase shifts between clock oscillators of different CAN controllers, any CAN controller has to synchronize on any edge from recessive to dominant bus level, if the edge lies between a sample point and the next synchronization segment, and on any other edge, if it itself does not send a dominant level. If the hard synchronization is enabled (at the start of frame), the bit time is restarted at the synchronization segment, otherwise, the resynchronization jump width (SJW) defines the maximum number of time quanta a bit time may be shortened or lengthened by one resynchronization. The current bit time is adjusted by

$t_{SJW} = (SJW + 1) \times t_q$

Note: SJW is the programmed numerical value from the respective field of the bit timing register.

6.4.5.2 Calculation of the Bit Time

The programming of the bit time according to the CAN specification depends on the desired baudrate, the CLP microcontroller system clock rate and on the external physical delay times of the bus driver, of the bus line and of the input comparator. These delay times are summarised in the propagation time segment t_{Prop}, where

t_{Prop} is two times the maximum of the sum of physical bus delay, the input comparator delay, and the output driver delay rounded up to the nearest multiple of t_q.

To fulfill the requirements of the CAN specification, the following conditions must be met:

$t_{TSeg2} \geq 3 \times t_q$ = *Information Processing Time*

$t_{TSeg2} \geq t_{SJW}$

$t_{TSeg1} \geq 4 \times t_q$

$t_{TSeg1} \geq t_{SJW} + t_{Prop}$

Note: In order to achieve correct operation according to the CAN protocol the total bit time should be at least 8 t_q, i.e. $t_{TSeg1} + t_{TSeg2} \geq 7\ t_q$.
So, to operate with a baudrate of 1 MBit/sec, the CLP frequency has to be at least 8 MHz (with bit CMOD = 1).

The maximum tolerance for CLP depends on the phase buffer segment1 (PB1), the phase buffer segment2 (PB2), and the resynchronization jump width (SJW):

$$df \leq \frac{\min(PB1, PB2)}{2 \times (13 \times \text{bit time} - PB2)}$$

AND

$$df \leq \frac{t_{SJW}}{20 \times \text{bit time}}$$

6.4.6 CAN Interrupt Handling

The CAN controller has one interrupt output, which is connected with the interrupt controller unit in the C505C. This interrupt can be enabled/disabled using bit ECAN of SFR IEN1 (further details about interrupt vector, priority, etc. see chapter 7). Additionally, three bits in the CAN control register ($F701_H$) are used to enable specific interrupt sources for interrupt generation.

Since an interrupt request of the CAN-Module can be generated due to different conditions, the appropriate CAN interrupt status register must be read in the service routine to determine the cause of the interrupt request. The interrupt identifier INTID (a number) in the interrupt register indicates the cause of an interrupt. When no interrupt is pending, the identifier will have the value “00”.

If the value in INTID is not “00”, then there is an interrupt pending. If bit IE in the control register is set, also the interrupt line is activated. The interrupt line remains active until either INTID gets “00” (ie. the interrupt requester has been serviced) or until IE is reset (ie. interrupts are disabled).

The interrupt with the lowest number has the highest priority. If a higher priority interrupt (lower number) occurs before the current interrupt is processed, INTID is updated and the new interrupt overrides the last one.

Table 6-7 below lists the valid values for INTID and their corresponding interrupt sources.

Table 6-7 : Interrupt IDs

INTID	Cause of the Interrupt
00	Interrupt idle There is no interrupt request pending.
01	Status change interrupt The CAN controller has updated (not necessarily changed) the status register. This can refer to a change of the error status of the CAN controller (EIE is set and BOFF or EWRN change) or to a CAN transfer incident (SIE must be set), like reception or transmission of a message (RXOK or TXOK is set) or the occurrence of a CAN bus error (LEC is updated). The microcontroller may clear RXOK, TXOK, and LEC, however, writing to the status partition of the control register can never generate or reset an interrupt. To update the INTID value the status partition of the control register must be read.
02	Message 15 interrupt Bit INTPND in the message control register of message object 15 (last message) has been set. The last message object has the highest interrupt priority of all message objects. [1)]
(2+N)	Message N interrupt: Bit INTPND in the message control register of message object ‘N’ has been set (N = 1...14). Note that a message interrupt code is only displayed, if there is no other interrupt request with a higher priority. [1)]

[1)] Bit INTPND of the corresponding message object has to be cleared to give messages with a lower priority the possibility to update INTID or to reset INTID to “00” (idle state).

6.4.7 CAN Controller in Power Saving Modes

Idle mode

In the idle mode of the C505C the CAN controller is fully operable. When a CAN controller interrupt becomes active and the CAN controller interrupt is enabled, the C505C restarts, returns to normal operation mode and starts executing the CAN controller interrupt routine.

Slow Down Mode

When the slow down mode is enabled the CAN controller is clocked with the reduced system clock rate (1/32 of the nominal clock rate). Therefore, also the CAN bit timing in slow down mode is reduced to 1/32 of the bit timing in normal mode. The slow down mode can be also combined with idle mode.

Power Down Mode

If the C505C enters software Power Down Mode, the system clock signal is turned off which will stop the operation of the CAN-Module. Any message transfer is interrupted. In order to ensure that the CAN controller is not stopped while sending a dominant level (“0”) on the CAN bus, the microcontroller should set bit INIT in the Control Register prior to entering Power Down Mode. The microcontroller can check, if a transmission is in progress by reading bits TXRQ and NEWDAT in the message objects and bit TXOK in the Control Register. After returning from Power Down Mode the CAN-Module has to be reconfigured.

6.4.8 Configuration Examples of a Transmission Object

The microcontroller wishes to configure an object for transmission. It wants to allow automatic transmission in response to remote frames but does not wish to receive interrupts for this object.

Initialization: The identifier and direction are set up.

Message Control Register

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
	1	0	0	1	0	1	0	1	MCR0
	MSGVAL		TXIE		RXIE		INTPND		
	0	1	0	1	1	0	0	1	MCR1
	RMTPND		TXRQ		CPUUPD		NEWDAT		

CPUUPD is initially set, as the data bytes for the message have not yet been initialized.

Configuration after remote frame received.

Message Control Register

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
	1	0	0	1	0	1	0	1	MCR0
	MSGVAL		TXIE		RXIE		INTPND		
	1	0	1	0	1	0	0	1	MCR1
	RMTPND		TXRQ		CPUUPD		NEWDAT		

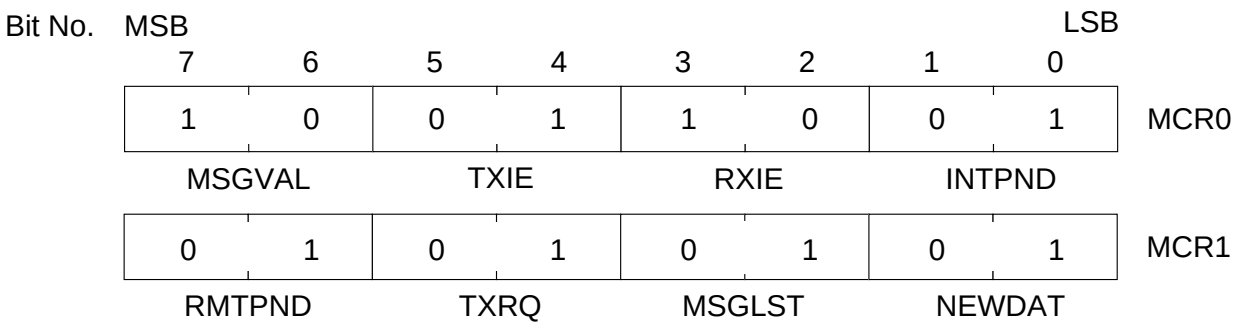
After updating the message the microcontroller should clear CPUUPD and set NEWDAT. If the microcontroller wants to transmit the message it should also set TXRQ, which should otherwise be left alone.

6.4.9 Configuration Examples of a Reception Object

The microcontroller wishes to configure an object for reception. It wishes to receive an interrupt each time new data comes in. From time to time the microcontroller sends a remote request to trigger the sending of this data from a remote node.

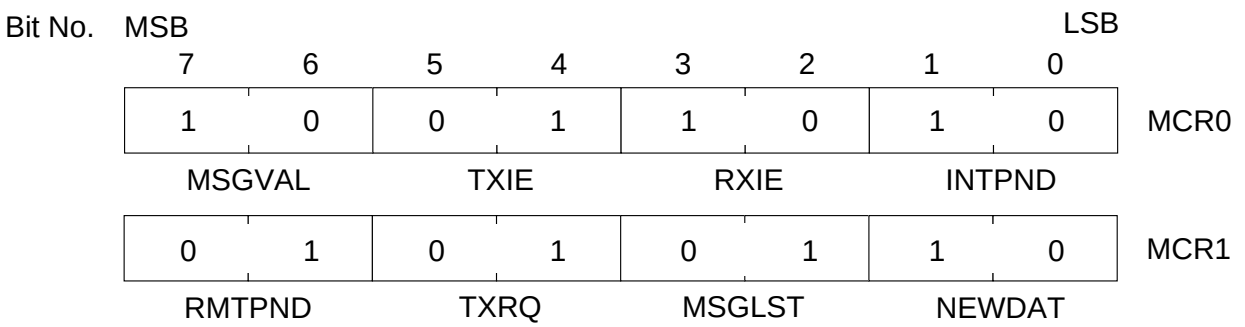
Initialization: The identifier and direction are set up.

Message Control Register



Configuration after reception of data.

Message Control Register



To process the message the microcontroller should clear IntPnd, clear NewDat, process the data, and check that NewDat is still clear. If not, it should repeat the process again. To send a remote frame to request the data, the microcontroller simply needs to set the TXRQ bit. This bit will be cleared by the CAN controller, once the remote frame has been sent or if the data is received before the CAN controller could transmit the remote frame.

6.4.10 The CAN Application Interface

The on-chip CAN controller of the C505C does not incorporate the physical layer that connects to the CAN bus. This must be provided externally. The module's CAN controller is connected to this physical layer (ie. the CAN bus) via two signals:

CAN Signal	Function
P4.1 / RXDC	Receive data from the physical layer of the CAN bus.
P4.0 / TXDC	Transmit data to the physical layer of the CAN bus.

A logic low level ("0") is interpreted as the dominant CAN bus level, a logic high level ("1") is interpreted as the recessive CAN bus level.

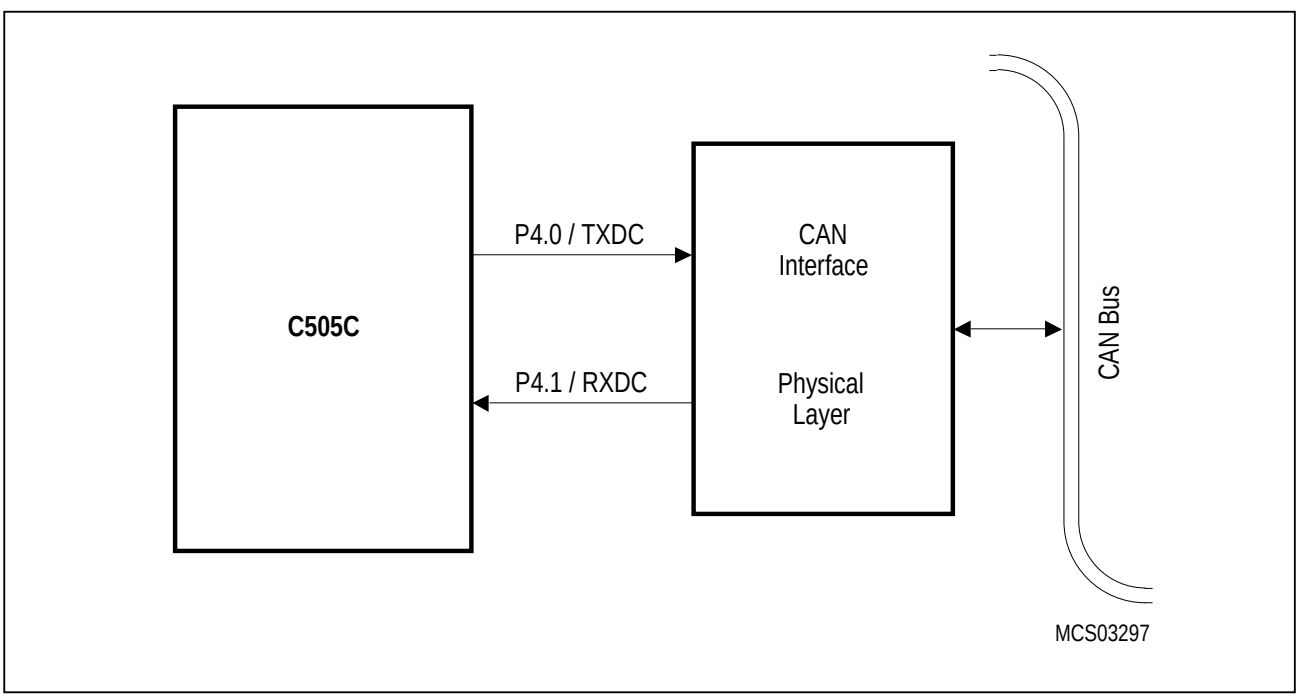


Figure 6-41
Connection to the CAN Bus

6.5 A/D Converter

The C505 includes a high performance / high speed 8-bit A/D converter with 8 analog input channels. It operates with a successive approximation technique and provides the following features:

- 8 multiplexed input channels (port 1), which can also be used as digital outputs/inputs
- 8-bit resolution
- Internal start-of-conversion trigger
- Interrupt request generation after each conversion
- Single or continuous conversion mode

The externally applied reference voltage range has to be held on a fixed value within the DC specifications (chapter 10). The main functional blocks of the A/D converter are shown in **Figure 6-42**.

6.5.1 A/D Converter Operation

An internal start of an A/D conversion is triggered by a write-to-ADST instruction. The start procedure itself is independent of the value which is written to the ADST register. 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 one A/D conversion a new A/D conversion is triggered automatically until the bit ADM (ADCON0.3) is reset.

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 is read-only and a write has no effect. The interrupt request flag IADC (IRCON.0) is set when an A/D conversion is completed.

The bits MX0 to MX2 of special function register ADCON0 are used for selection of the analog input channel. The bits MX0 to MX2 are represented in the register ADCON1 as well but in reality are present only once. Therefore, there are two methods of selecting an anlog 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.

The ADCON1 register is also used for selecting the required prescaler (bits 6 and 7) for achieving the proper clock input to the A/D converter.

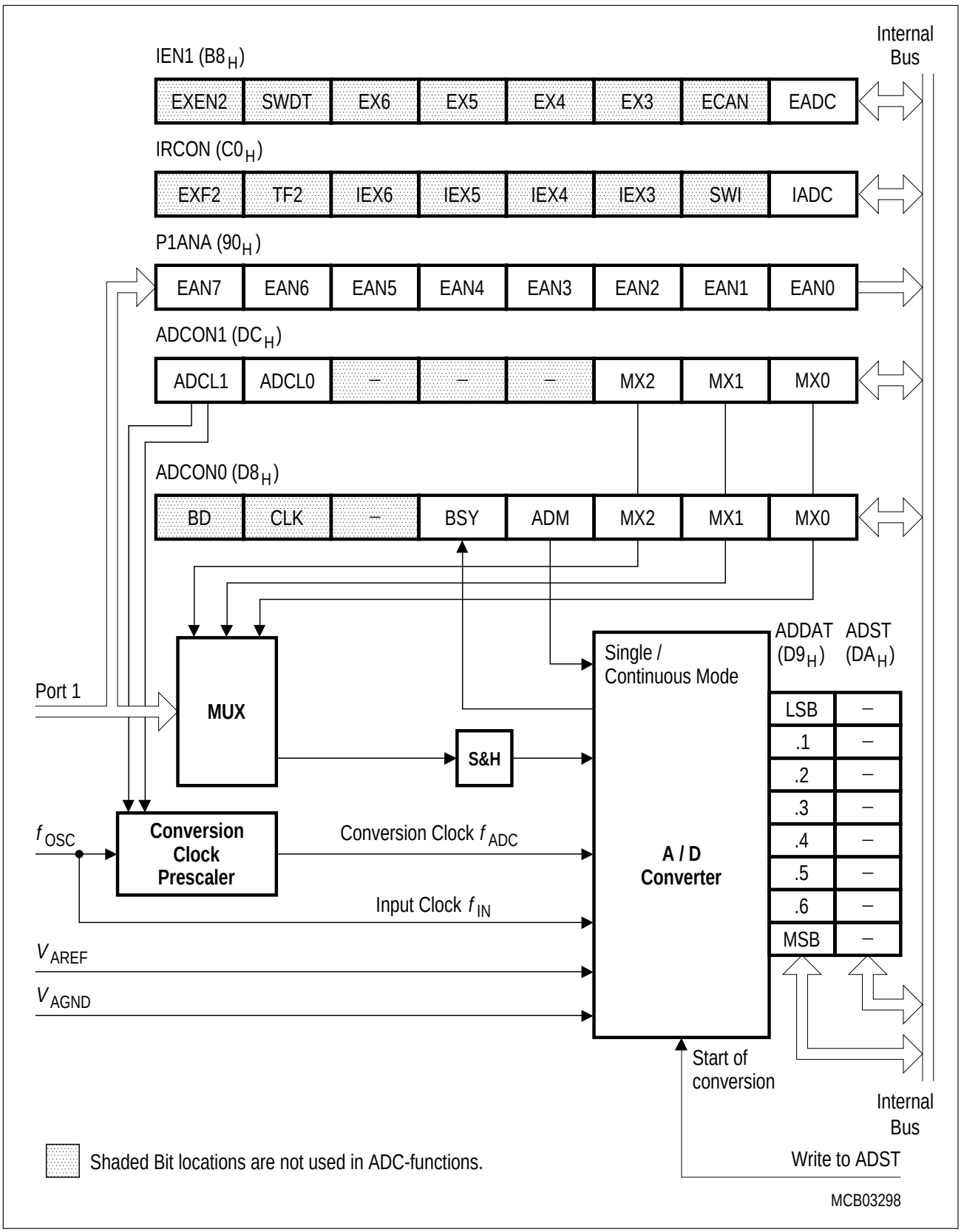


Figure 6-42
Block Diagram of the A/D Converter

6.5.2 A/D Converter Registers

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

Special Function Register ADDAT (Address $D9_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$D9_H$	.7	.6	.5	.4	.3	.2	.1	.0	ADDAT

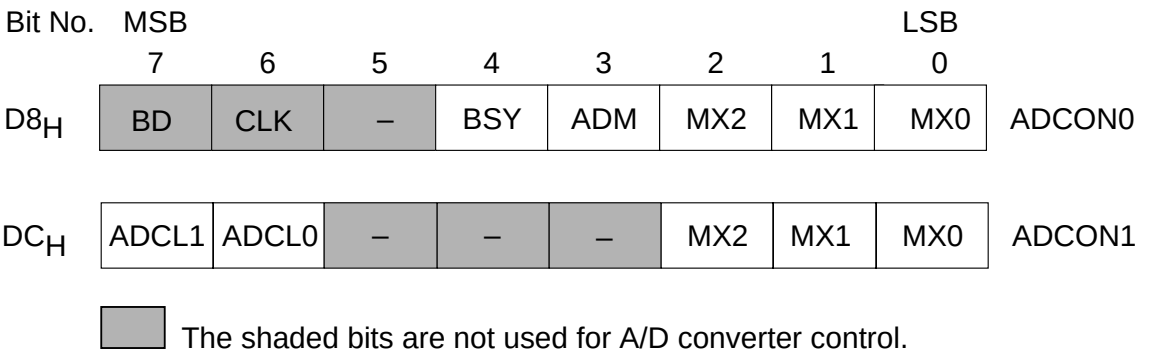
This register holds the result of the 8-bit conversion. The most significant bit of the 8-bit conversion result is bit 7 and the least significant bit is bit 0 of the ADDAT register. The data remains in ADDAT until it is overwritten by the next converted data. ADDAT can be read or written under software control. If the A/D converter of the C505 is not used, register ADDAT can be used as an additional general purpose register.

Special Function Register ADST (Address DA_H) **Reset Value : XX_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
DA_H	–	–	–	–	–	–	–	–	ADST

This is the A/D conversion start register. Any write operation on the ADST register starts a new conversion process irrespective of the actual data written to this register. Read by the CPU returns unefined values.

Special Function Register ADCON0 (Address $D8_H$) **Reset Value : $00X00000_B$**
Special Function Register ADCON1 (Address DC_H) **Reset Value : $01XXX000_B$**



Bit	Function
BSY	Busy flag This flag indicates whether a conversion is in progress (BSY = 1). The flag is set by hardware during an A/D conversion and cleared when the conversion is completed.
ADM	A/D conversion mode When set, a continuous A/D conversion is selected. If cleared during a running A/D conversion, the conversion is stopped after current conversion process is completed.
MX2 - MX0	A/D converter input channel select bits Bits MX2-0 can be written or read either in ADCON0 or ADCON1. The channel selection done by writing to ADCON1(0) overwrites the selection in ADCON 0(1) when ADCON1(0) is written after ADCON0(1). The analog inputs are selected according to the following table :

MX2	MX1	MX0	Selected Analog Input
0	0	0	P1.0 / AN0 / $\overline{INT3}$ / CC0
0	0	1	P1.1 / AN1 / INT4 / CC1
0	1	0	P1.2 / AN2 / INT5 / CC2
0	1	1	P1.3 / AN3 / INT6 / CC3
1	0	0	P1.4 / AN4
1	0	1	P1.5 / AN5 / T2EX
1	1	0	P1.6 / AN6 / CLKOUT
1	1	1	P1.7 / AN7 / T2

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>ADCL1
ADCL0</td><td>A/D converter clock prescaler selection
ADCL1 and ADCL0 select the prescaler ratio for the A/D conversion clock f_{ADC}. Depending on the clock rate f_{OSC} of the C505, f_{ADC} must be adjusted in a way that the resulting conversion clock f_{ADC} is less than or equal to 1.25 MHz (see section 6.5.3).
The prescaler ratio is selected according to the following table:
<table>
<tr><th>ADCL1</th><th>ADCL0</th><th>Prescaler Ratio</th></tr>
<tr><td>0</td><td>0</td><td>divide by 4</td></tr>
<tr><td>0</td><td>1</td><td>divide by 8 (default after reset)</td></tr>
<tr><td>1</td><td>0</td><td>divide by 16</td></tr>
<tr><td>1</td><td>1</td><td>divide by 32</td></tr>
</table></td></tr>
<tr><td>–</td><td>Reserved bits for future use.Read by CPU returns undefined values.</td></tr>
</table>

Note : Generally, before entering the software 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 software power-down mode.

A single A/D conversion is started by writing to SFR ADST 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 ADST 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, any ongoing A/D conversion is stopped only after it is completed.

In any case the A/D conversion is started with the next S1P1.

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

Special Function Register IEN1 (Address $B8_H$) **Reset Value : 00_H**
Special Function Register IRCON (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	ECAN	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	0	IADC	IRCON

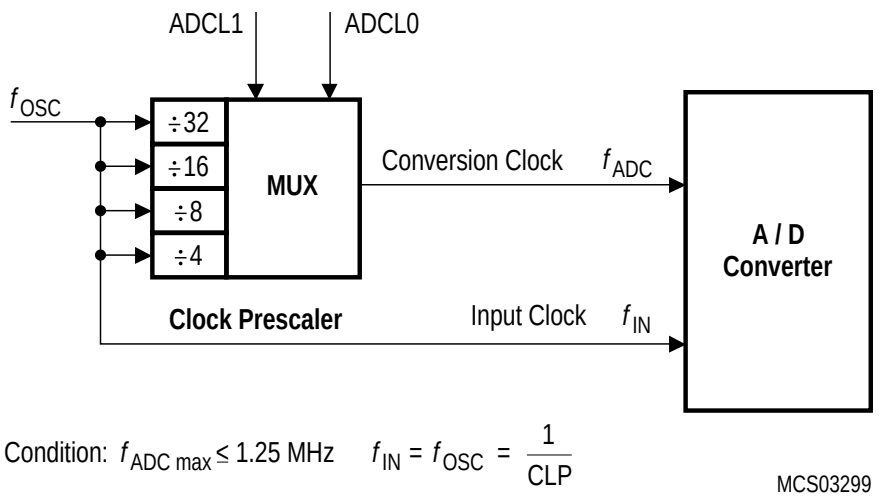
The shaded bits are not used for A/D converter control.

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.

6.5.3 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}). f_{ADC} is derived from the C505 system clock f_{OSC} which is applied at the XTAL pins via the ADC clock prescaler as shown in **figure 6-43**. The input clock is equal to f_{OSC}. The conversion clock f_{ADC} is limited to a maximum frequency of 1.25 MHz. Therefore, the ADC clock prescaler must be programmed to a value which assures that the conversion clock does not exceed 1.25 MHz. The prescaler ratio is selected by the bits ADCL1 and ADCL0 of SFR ADCON1.

The table in **figure 6-43** shows the prescaler ratio which must be selected by ADCL1 and ADCL0 for typical system clock rates. Up to 5 MHz system clock the prescaler ratio 4 is selected. Using a system clock greater than 5 and less than 10 MHz, the prescaler ratio of at least 8 must be selected. A prescaler ratio of 16 must be selected when using a system clock greater than 10 MHz.



MCU System Clock Rate (f_{OSC})	f_{IN} [MHz]	Prescaler Ratio	f_{ADC} [MHz]	ADCL1	ADCL0
2 MHz	2	÷ 4	0.5	0	0
5 MHz	5	÷ 4	1.25	0	0
6 MHz	6	÷ 8	0.75	0	1
10 MHz	10	÷ 8	1.25	0	1
12 MHz	12	÷ 16	0.75	1	0
16 MHz	16	÷ 16	1	1	0
20 MHz	20	÷ 16	1.25	1	0

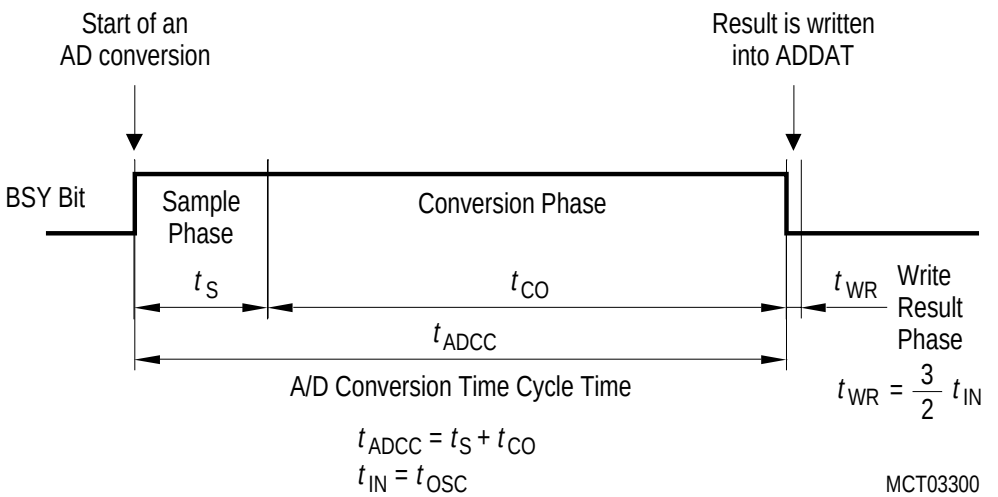
Figure 6-43
A/D Converter Clock Selection

6.5.4 A/D Converter Timing

An A/D conversion is started by writing into special function register ADST. A write-to-ADST will start a new conversion even if a conversion is currently in progress. The conversion begins with the next machine cycle and the busy flag BSY 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.
- Write result phase (t_{WR}), used for writing the conversion result into the ADDAT register.

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**.



Prescalar Ratio (=PS)	t_S = 2 x PS x t_{IN}	t_{CO} = 8 x PS x t_{IN}	Conversion Time		
			t_{ADCC}	CPU Cycles n Time	Ref.
32	64 x t_{IN}	256 x t_{IN}	320 x t_{IN}	53 + 2 x t_{IN}	a)
16	32 x t_{IN}	128 x t_{IN}	160 x t_{IN}	26 + 4 x t_{IN}	b)
8	16 x t_{IN}	64 x t_{IN}	80 x t_{IN}	13 + 2 x t_{IN}	a)
4	8 x t_{IN}	32 x t_{IN}	40 x t_{IN}	6 + 4 x t_{IN}	b)

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 input voltage to be converted. The analog voltage is internally fed to a voltage comparator. With the 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 an 8-bit digital value using the successive approximation technique with a binary weighted capacitor network. At the end of the conversion time the BSY bit is reset and the IADC bit in SFR IRCON 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 6-43 shows how an A/D conversion is embedded into the microcontroller cycle scheme using the relation 6 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.

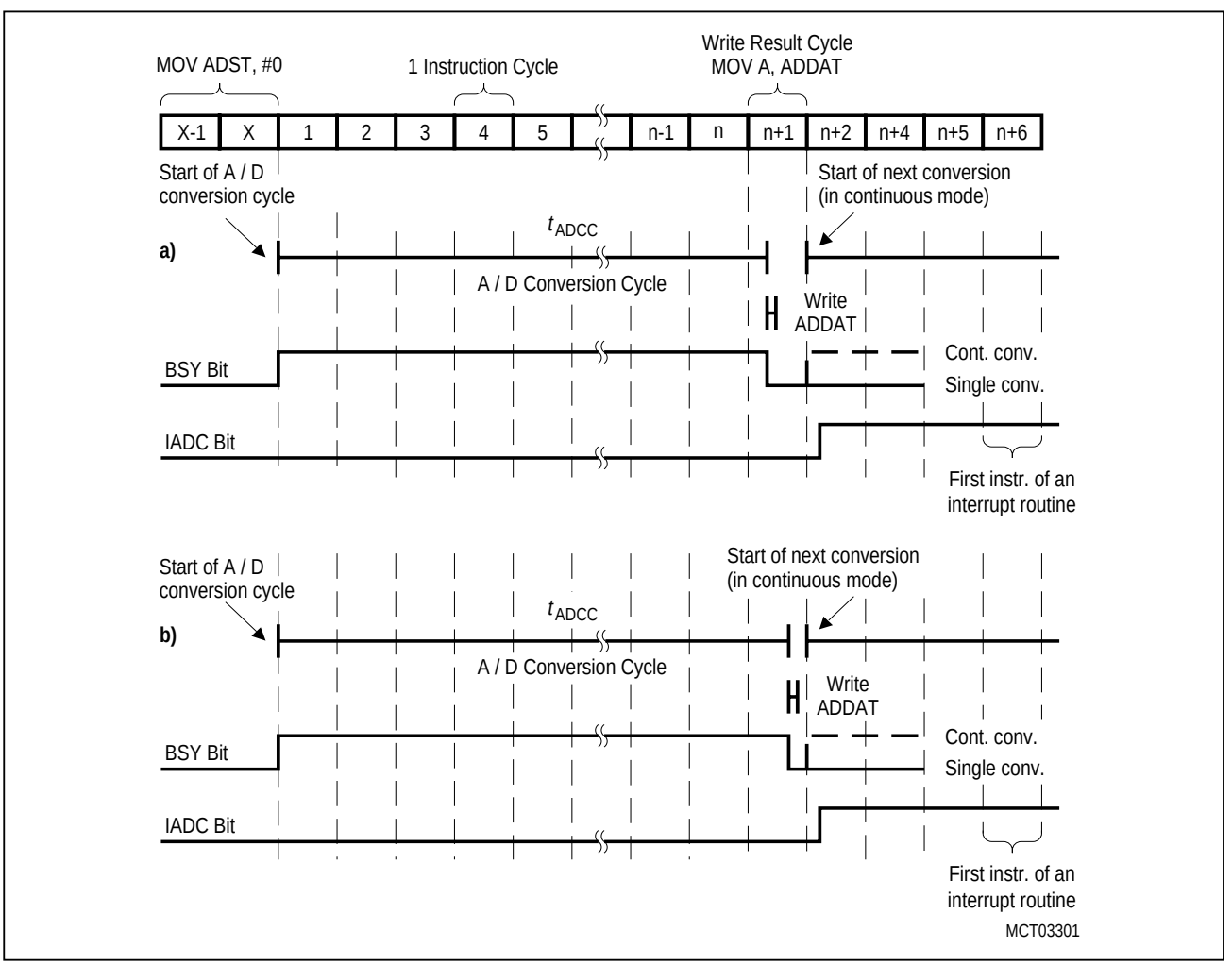


Figure 6-45
A/D Conversion Timing in Relation to Processor Cycles

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 ADST with dummy data. The ADST write operation may take one or two machine cycles. In **figure 6-45**, the instruction MOV ADST,#0 starts the A/D conversion (machine cycles X-1 and X). The total A/D conversion (sample and conversion phase) is finished with the end of the 10th ADC clock cycle after the A/D conversion start. The actual machine number of machine cycles needed follows the table in **Figure 6-44**. In the next machine cycle the conversion result is written into the ADDAT register and can be read in the same cycle by an instruction (e.g. MOV A,ADDAT). If continuous conversion is selected (bit ADM set), the next conversion is started with the beginning of the machine cycle which follows the write 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. This means that in continuous conversion mode BSY is not set for a complete machine cycle. Therefore, in continuous conversion mode it is not recommended to poll the BSY bit using for e.g. the JNB instruction.

The interrupt flag IADC is set at the end of the A/D conversion so that it is polled the first time in S1P2 of the machine cycle following the completion of conversion. If the A/D converter interrupt is enabled and the A/D converter interrupt is prioritized to be serviced immediately, the first instruction of the interrupt service routine will be executed in the fourth machine cycle which follows the write result cycle. The IADC bit must be reset by software.

Depending on the application, typically there are three methods to handle the A/D conversion in the C505.

- Software delay
  The machine cycles during an 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. The end of conversion is indicated by the IADC flag.
- Polling BSY bit,
  The BSY bit is polled and the program waits until BSY=0. Attention : a polling JB 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 C505 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 C505 and the selected divider ratio of the conversion clock prescaler the total time of an A/D conversion is calculated according **figure 6-44** and **table 6-8. Figure 6-46** shows the minimum A/D conversion time in relation to the oscillator frequency f_{OSC}. The minimum conversion time is 8 µs and can be achieved at f_{OSC} of 10 MHz and with a prescalar ratio of 8 (or whenever f_{ADC} = 1.25 MHz).

Table 6-8
A/D Conversion Time for Dedicated System Clock Rates

f_{OSC} [MHz]	Prescaler Ratio PS	f_{ADC} [MHz]	Sample Time t_S [µs]	Total Conversion Time t_{ADCC} [µs]
2 MHz	÷ 4	.5	4	20
5 MHz	÷ 4	1.25	1.6	8
6 MHz	÷ 8	0.75	2.67	13.33
10 MHz	÷ 8	1.25	1.6	8
12 MHz	÷ 16	0.75	2.67	13.33
16 MHz	÷ 16	1	2	10
20 MHz	÷ 16	1.25	1.6	8

Note : The prescaler ratios in **table 6-8** are minimum values.

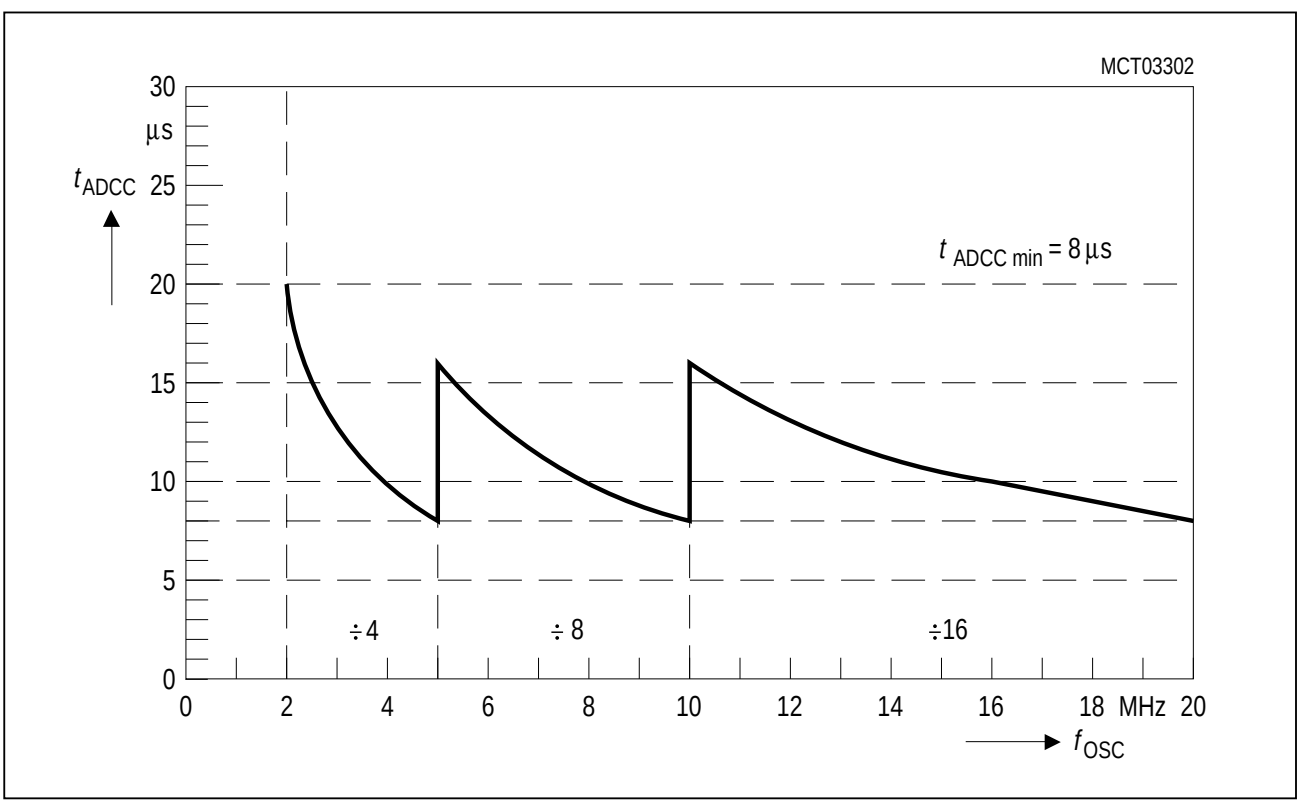


Figure 6-46
Minimum A/D Conversion Time in Relation to System Clock

6.5.5 A/D Converter Analog Input Selection

The analog inputs are located at port 1. The corresponding pins have a port structure, which allows to use them either as digital I/O pins or as analog inputs (see section 6.1.3.2). The analog input function of these digital/analog port lines are selected via the register P1ANA. This register lies in the mapped SFR area and can be accessed when bit RMAP in SFR SYSCON is set when writing to its address (90_H). If a specific bit location of P1ANA is set, the corresponding port line is configured as a digital input. With a 0 in the bit location the port line operates as analog port.

Special Function Registers P1ANA (Address 90_H) **Reset Value : FF_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
90_H	EAN7	EAN6	EAN5	EAN4	EAN3	EAN2	EAN1	EAN0	P1ANA

Bit	Function
EAN7 - EAN0	Enable analog port 1 inputs If EANx (x = 7-0) is cleared, port pin P1.x is enabled for operation as an analog input. If EANx is set, port pin P1.x is enabled for digital I/O function (default after reset).

7 Interrupt System

The C505 provides 12 interrupt vectors with four priority levels. Five interrupt requests can be generated by the on-chip peripherals (timer 0, timer 1, timer 2, serial interface, A/D converter). One interrupt can be generated by the CAN controller (C505C only) or by a software setting and in this case the interrupt vector is the same. Six interrupts may be triggered externally (P3.2/$\overline{\text{INT0}}$, P3.3/$\overline{\text{INT1}}$, P1.0/AN0/$\overline{\text{INT3}}$/CC0, P1.1/AN1/INT4/CC1, P1.2/AN2/INT5/CC2, P1.3/AN3/INT6/CC3). Additionally, the P1.5/AN5/T2EX can trigger an interrupt. The wake-up from power-down mode interrupt has a special functionality which allows to exit from the software power-down mode by a short low pulse at either pin P3.2/$\overline{\text{INT0}}$ or the pin P4.1/RXDC (please refer to chapter 9 for further details).

The four external interrupts ($\overline{\text{INT3}}$, INT4, INT5 and INT6) can also be generated by the timer 2 in capture/compare mode.

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.

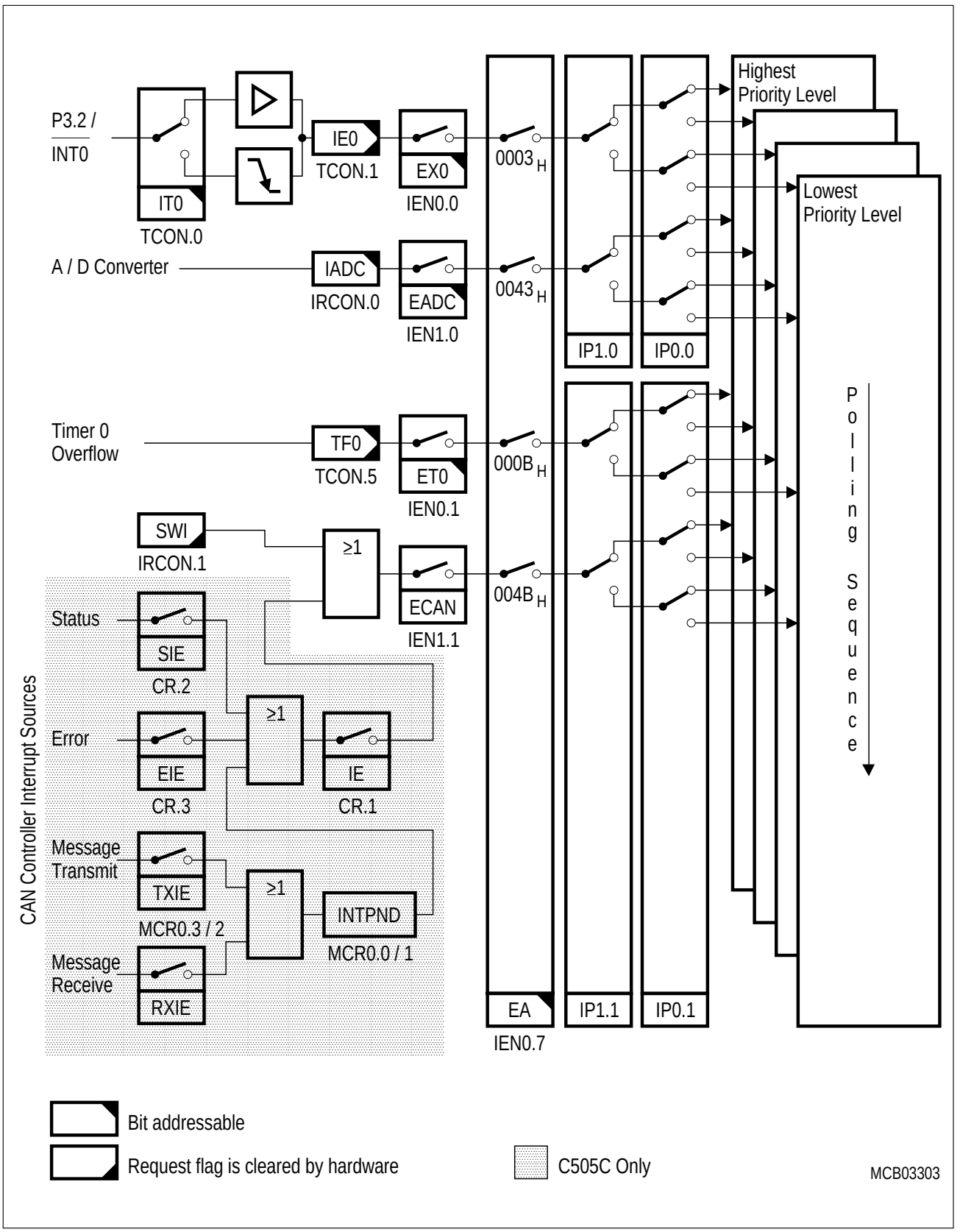


Figure 7-1
Interrupt Structure, Overview Part 1

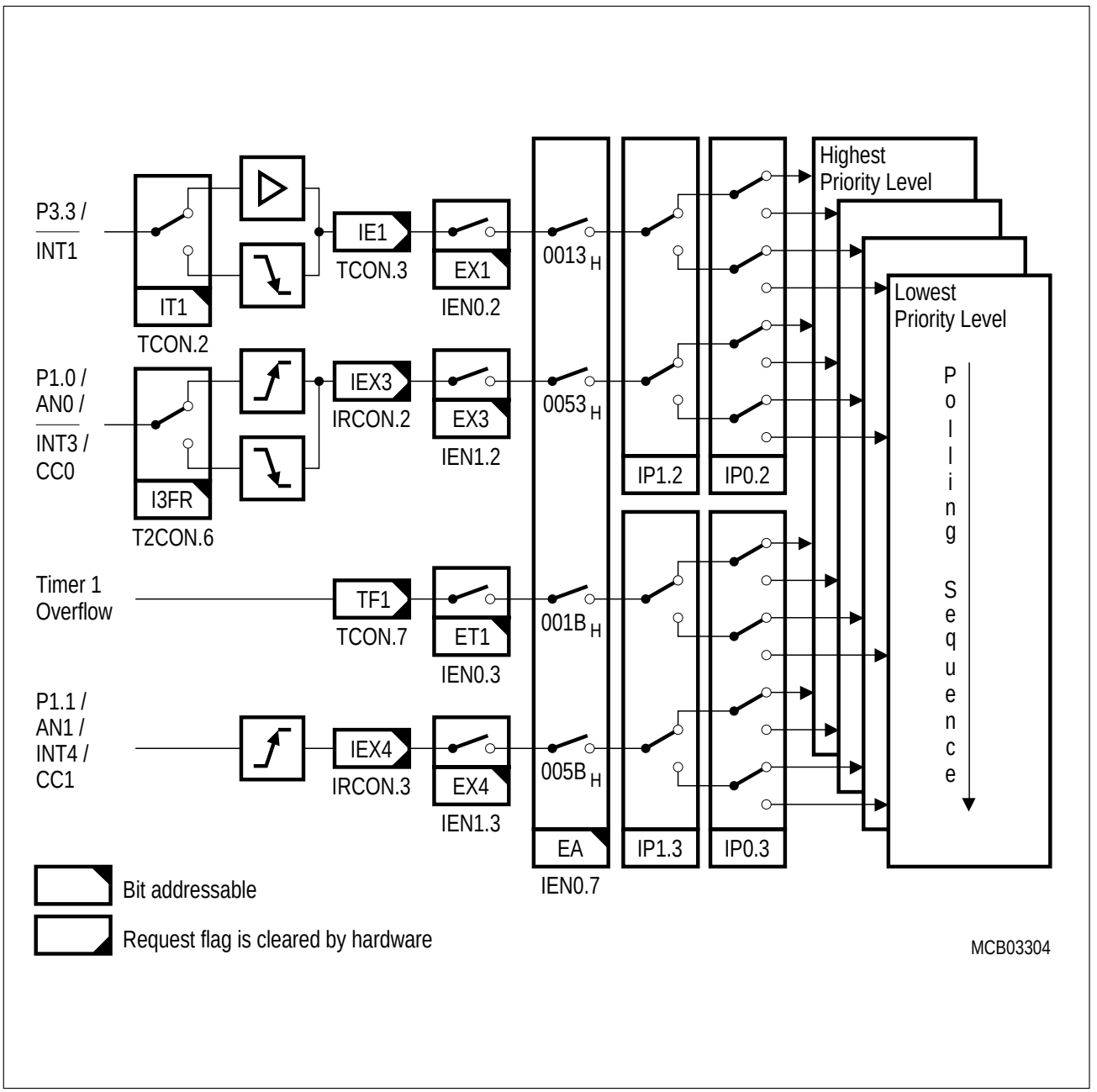


Figure 7-2
Interrupt Structure, Overview Part 2

Note: Each of the 15 CAN controller message objects (C505C only), shown in the shaded area of **Figure 7-1** provides the bits/flags.

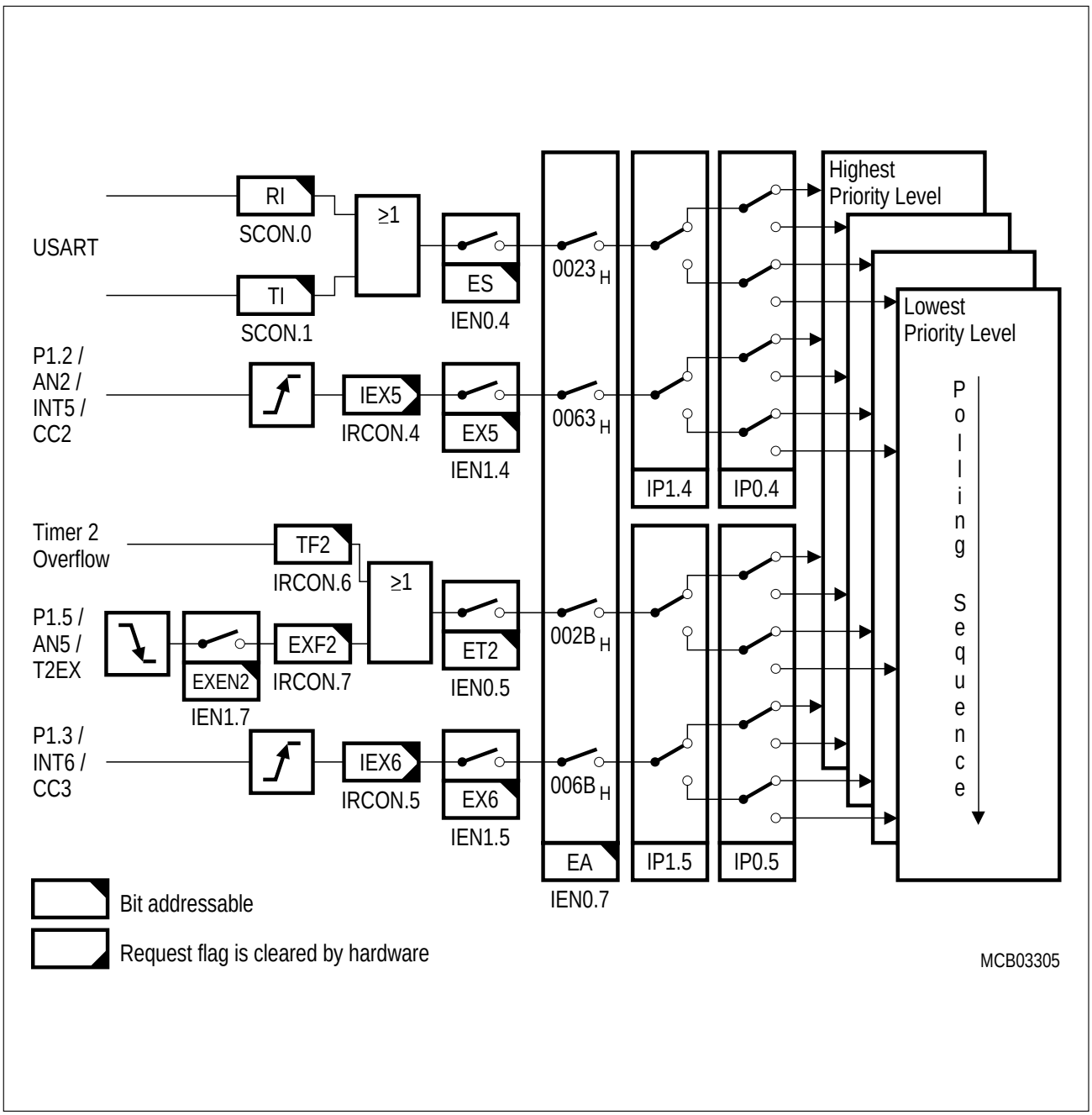


Figure 7-3
Interrupt Structure, Overview Part 3

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 and IEN1. Register IEN0 also contains the global disable bit (EA), which can be cleared to disable all interrupts at once. Generally, after reset all interrupt enable bits are set to 0. That means that the corresponding interrupts are disabled.

The IEN0 register contains the general enable/disable flags of the external interrupts 0 and 1, the timer interrupts, and the USART interrupt.

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	$A9_H$	$A8_H$	
$A8_H$	EA	WDT	ET2	ES	ET1	EX1	ET0	EX0	IEN0

The shaded bits are not used for interrupt control.

Bit	Function
EA	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 overflow / external reload interrupt enable. If ET2 = 0, the timer 2 interrupt is disabled. If ET2 = 1, the timer 2 interrupt is enabled.
ES	Serial channel (USART) interrupt enable If ES = 0, the serial channel interrupt 0 is disabled. If ES = 1, the serial channel interrupt 0 is enabled.
ET1	Timer 1 overflow interrupt enable. If ET1 = 0, the timer 1 interrupt is disabled. If ET1 = 1, the timer 1 interrupt is enabled.
EX1	External interrupt 1 enable. If EX1 = 0, the external interrupt 1 is disabled. If EX1 = 1, the external interrupt 1 is enabled.
ET0	Timer 0 overflow interrupt enable. If ET0 = 0, the timer 0 interrupt is disabled. If ET0 = 1, the timer 0 interrupt is enabled.
EX0	External interrupt 0 enable. If EX0 = 0, the external interrupt 0 is disabled. If EX0 = 1, the external interrupt 0 is disabled.

The IEN1 register contains enable/disable flags of the timer 2 external timer reload interrupt, the external interrupts 2 and 3, the CAN controller interrupt and the A/D converter interrupt.

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

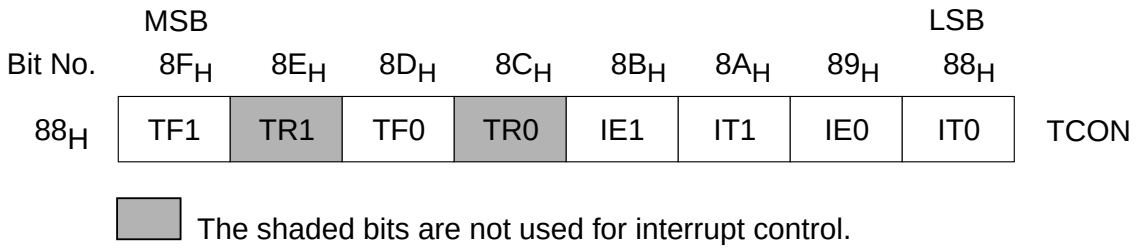
	MSB							LSB	
Bit No.	BF_H	BE_H	BD_H	BC_H	BB_H	BA_H	$B9_H$	$B8_H$	
$B8_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	ECAN	EADC	IEN1

The shaded bits are 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. If EXEN2 = 1, the timer 2 external reload interrupt is enabled. 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. If EX6 = 1, external interrupt 6 is enabled.
EX5	External interrupt 5 / capture/compare interrupt 2 enable If EX5 = 0, external interrupt 5 is disabled. If EX5 = 1, external interrupt 5 is enabled.
EX4	External interrupt 4 / capture/compare interrupt 1 enable If EX4 = 0, external interrupt 4 is disabled. If EX4 = 1, external interrupt 4 is enabled.
EX3	External interrupt 3 / capture/compare interrupt 0 enable If EX3 = 0, external interrupt 3 is disabled. If EX3 = 1, external interrupt 3 is enabled.
ECAN	CAN controller interrupt enable (C505C only) If ECAN = 0, the CAN controller interrupt is disabled. If ECAN = 1, the CAN controller interrupt is enabled. This bit must be set in order to enable the software interrupt at bit SWI (for C505 and C505C)
EADC	A/D converter interrupt enable If EADC = 0, the A/D converter interrupt is disabled. If EADC = 1, the A/D converter interrupt is enabled.

7.1.2 Interrupt Request / Control Flags

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



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 when external interrupt 1 edge is detected. Cleared by hardware when processor vectors to interrupt routine.
IT1	External interrupt 1 level/edge trigger control flag If IT1 = 0, low level triggered external interrupt 1 is selected. If IT1 = 1, falling edge triggered external interrupt 1 is selected.
IE0	External interrupt 0 request flag Set by hardware when external interrupt 0 edge is detected. Cleared by hardware when processor vectors to interrupt routine.
IT0	External interrupt 0 level/edge trigger control flag If IT0 = 0, low level triggered external interrupt 0 is selected. If IT0 = 1, falling edge triggered external interrupt 0 is selected.

The **external interrupts 0 and 1** ($\overline{INT0}$ and $\overline{INT1}$) can each be either level-activated or negative transition-activated, depending on bits IT0 and IT1 in register TCON. The flags that actually generate these interrupts are bits IE0 and lE1 in 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. 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 T2CON (Address $C8_H$) **Reset Value : $00X00000_B$**

	MSB							LSB	
Bit No.	CF_H	CE_H	CD_H	CC_H	CB_H	CA_H	$C9_H$	$C8_H$	
$C8_H$	T2PS	I3FR	–	T2R1	T2R0	T2CM	T2I1	T2I0	T2CON

The shaded bits are not used for interrupt control.

Bit	Function
I3FR	External interrupt 3 rising/falling edge control flag If I3FR = 0, the external interrupt 3 is activated by a falling edge at P1.0/AN0/$\overline{INT3}$/CC0. If I3FR = 1, the external interrupt 3 is activated by a rising edge at P1.0/AN0/$\overline{INT3}$/CC0.
–	This bit has no effect in the C505.

The **external interrupt 3** ($\overline{INT3}$) 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 IRCON. In addition, this flag will be set if a compare event occurs at pin P1.0/AN0/$\overline{INT3}$/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.

Special Function Register IRCON (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	SWI	IADC	IRCON

Bit	Function
EXF2	Timer 2 external reload flag EXF2 is set when a reload is caused by a falling edge on pin T2EX while EXEN2 = 1. If ET2 in IEN0 is set (timer 2 interrupt enabled), EXF2 = 1 will cause an interrupt. EXF2 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 occured at P1.3/AN3/INT6/CC3. Cleared when interrupt is processed.
IEX5	External interrupt 5 edge flag Set by hardware when external interrupt edge was detected or when a compare event occured at P1.2/AN2/INT5/CC2. Cleared when interrupt is processed.
IEX4	External interrupt 4 edge flag Set by hardware when external interrupt edge was detected or when a compare event occured at P1.1/AN1/INT4/CC1. Cleared when interrupt is processed.
IEX3	External interrupt 3 edge flag Set by hardware when external interrupt edge was detected or when a compare event occured at P1.0/AN0/$\overline{\text{INT3}}$/CC0. Cleared when interrupt is processed.
SWI	This bit can be set by software to generate an interrupt. This bit is cleared when the interrupt is processed. The interrupt vector address is $004B_H$.
IADC	A/D converter interrupt request flag Set by hardware at the end of an A/D conversion. Must be cleared by software.

The **timer 2 interrupt** is generated by the logical OR of bit TF2 in register T2CON and bit EXF2 in register IRCON. 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 bit in register IRCON. If an interrupt is generated, in any case the converted result in ADDAT is valid on the first instruction of the interrupt service routine. lf continuous conversion is established, IADC is set once during each conversion. lf an A/D converter interrupt is generated, flag IADC will have to be cleared by software.

The **external interrupts 4 to 6** (INT4, INT5 and INT6) are positive transition-activated. The flags that actually generate these interrupts are bits IEX4, IEX5 and IEX6 in register IRCON. In addition, these flags will be set if a compare event occurs at the corresponding output pin P1.3/AN3/INT6/CC3, P1.2/AN2/INT5/CC2, and P1.1/AN1/INT4/CC1, 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.

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. lf the external interrupts 0 and 1 are programmed to be level-activated, IE0 and lE1 are controlled by the external source via pin $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$, 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 $\overline{\text{INT0}}$ (P3.2) and $\overline{\text{INT1}}$ (P3.3), provided that this will not affect any peripheral circuit connected to the pins.

The bit IRCON.1 can be set by software to vector to location $\mathbf{004B_H}$. In the case of the C505C, care should be taken while manipulating this bit to avoid any erroneous CAN interrupt generation.

Special Function Register SCON (Address. 98_H) **Reset Value : 00_H**

	MSB							LSB
Bit No.	$9F_H$	$9E_H$	$9D_H$	$9C_H$	$9B_H$	$9A_H$	99_H	98_H
98_H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SCON

The shaded bits are not used for interrupt control.

Bit	Function
TI	Serial interface transmitter interrupt flag Set by hardware at the end of a serial data transmission. Must be cleared by software.
RI	Serial interface receiver interrupt flag Set by hardware if a serial data byte has been received. Must be cleared by software.

The **serial port interrupt** is generated by a logical OR of flag RI and TI in SFR SCON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine 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.

7.1.3 Interrupt Priority Registers

The lower six bits of these two registers are used to define the interrupt priority level of the interrupt groups as they are defined in **table 7-1** in the next section.

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
Bit No.	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 control.

Bit	Function
IP1.x IP0.x	Interrupt group priority level bits (x=0-5, see **table 7-1**)
–	Reserved bits for future use. Read by CPU returns undefined values.

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 following table shows the interrupt grouping of the C505 interrupt sources.

Table 7-1
Interrupt Source Structure

Interrupt Group	Associated Interrupts High priority → Low priority		Priority
1	External interrupt 0	A/D converter interrupt	High
2	Timer 0 overflow	CAN controller interrupt (C505C) / Software Interrupt (IRCON.1)	↓
3	External interrupt 1	External interrupt 3	
4	Timer 1 overflow	External interrupt 4	
5	Serial channel interrupt	External interrupt 5	
6	Timer 2 interrupt	External interrupt 6	Low

Each pair 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 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 levels 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 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 registers IEN0/IEN1 or IP0/IP1, 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**.

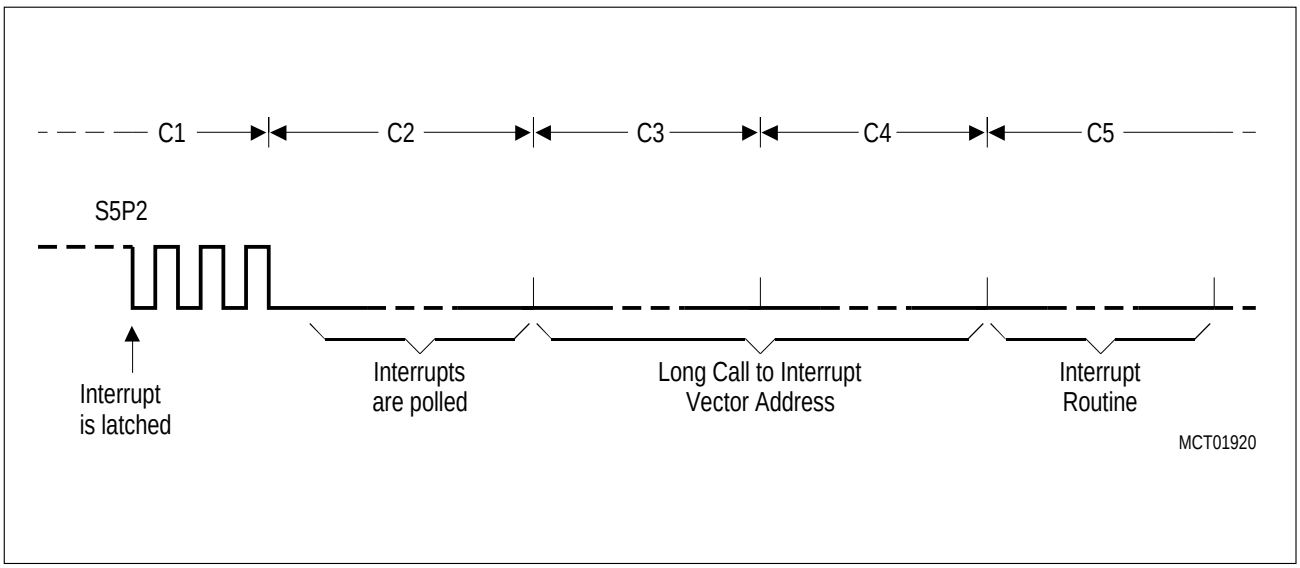


Figure 7-4
Interrupt Response Timing Diagram

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 the following **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	0023_H	RI / TI
Timer 2 Overflow / Ext. Reload	$002B_H$	TF2 / EXF2
A/D Converter	0043_H	IADC
CAN Controller / Software Interrupt	$004B_H$	–
External interrupt 3	0053_H	IEX3
External Interrupt 4	$005B_H$	IEX4
External Interrupt 5	0063_H	IEX5
External interrupt 6	$006B_H$	IEX6
Wake-up from power-down mode	$007B_H$	–

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 ITx (x = 0 or 1), respectively in register TCON. If ITx = 0, external interrupt x is triggered by a detected low level at the $\overline{INTx}$ pin. If ITx = 1, external interrupt x is negative edge-triggered. In this mode, if successive samples of the $\overline{INTx}$ 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=1 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 4, 5 and 6 are activated only 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/AN5/ T2EX but only if bit EXEN2 is set.

Since the external interrupt pins (INT4, INT5 and INT6) are sampled once in each machine cycle, an input high or low should be held for at least 6 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin low for at least one cycle, and then hold it high 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.

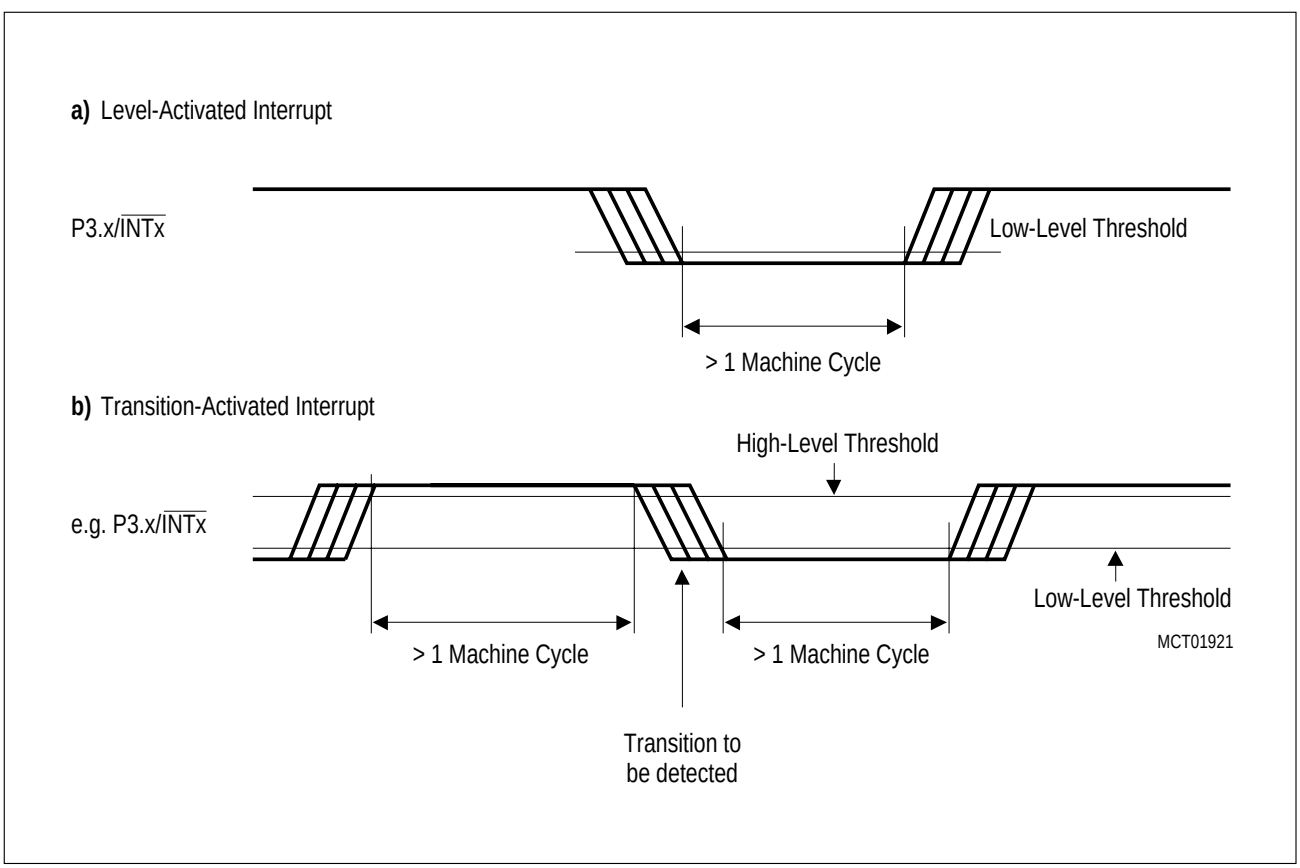


Figure 7-5
External Interrupt Detection

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 the 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 IEN0, IEN1 or IP0, IP1 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 Save Mechanisms

The C505 offers enhanced fail save mechanisms, which allow an automatic recovery from software upset or hardware failure :

- a programmable watchdog timer (WDT), with variable time-out period from 192 µs up to approx. 412.5 ms at 16 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 C505 is a 15-bit timer, which is incremented by a count rate of $f_{OSC}/12$ upto $f_{OSC}/192$. The machine clock of the C505 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 bits of the watchdog timer can be written. **Figure 8-1** shows the block diagram of the watchdog timer unit.

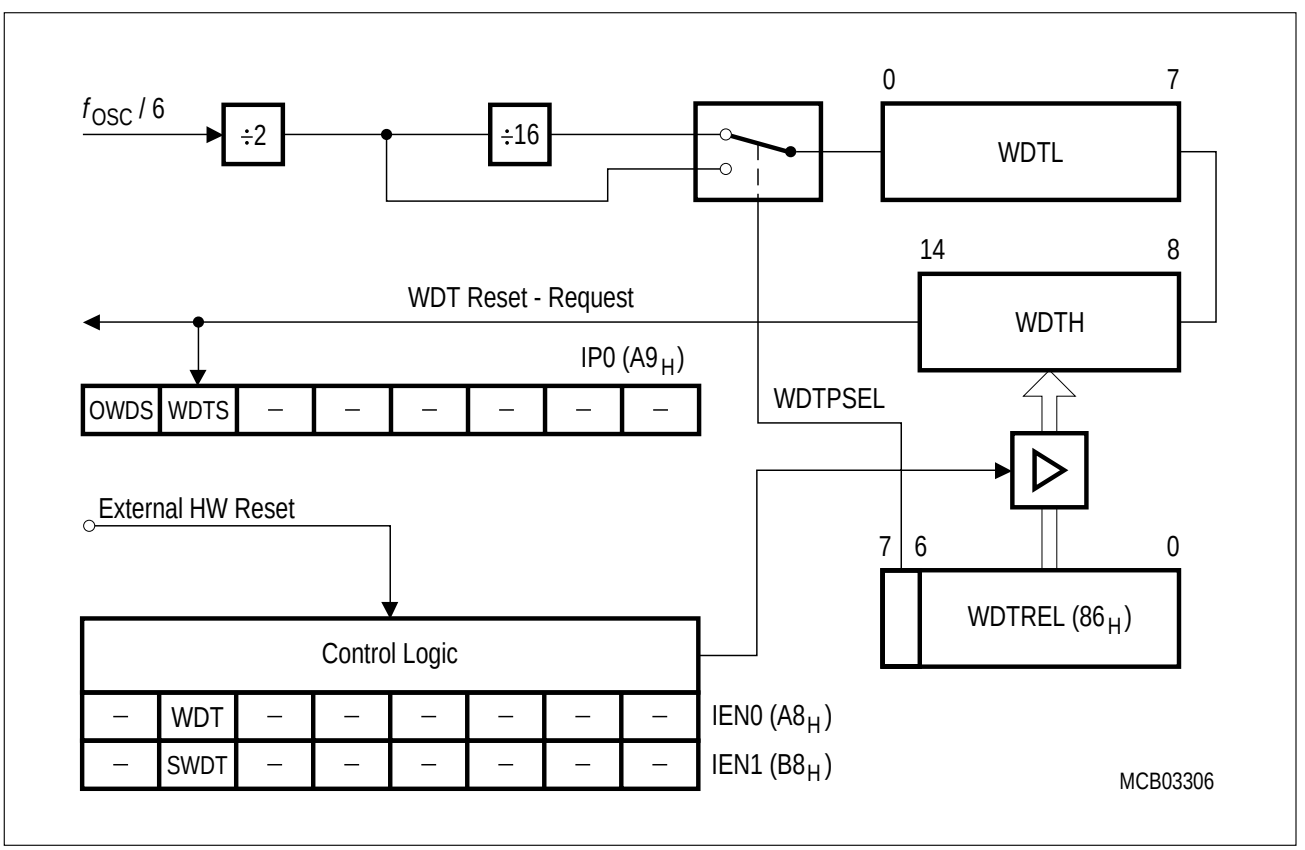
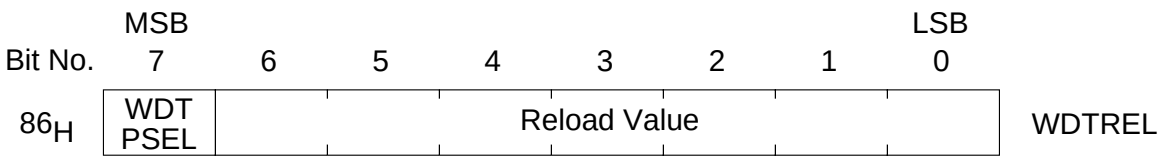


Figure 8-1
Block Diagram of the Programmable Watchdog Timer

8.1.1 Input Clock Selection

The input clock rate of the watchdog timer is derived from the system clock of the C505. 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 16 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

WDTREL	Time-Out Period		Comments
	f_{osc} = 12 MHz	f_{osc} = 16 MHz	
00_H	32.768 ms	24.576 ms	This is the default value
80_H	524.2 ms	393.2 ms	Maximum time period
$7F_H$	256 µs	192 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 flag (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	ECAN	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.

Immediately after start, the Watchdog Timer is initialized to the reload value programmed in WDTREL.0-WDTREL.6. After an external HW reset, an oscillator watchdog power on reset, or a watchdog timer reset, register WDTREL is cleared to 00_H. The lower seven bits of WDTREL can be loaded by software at any time.

8.1.3 Starting the Watchdog Timer

The Watchdog Timer can be started by software (bit SWDT in SFR IEN1), but it cannot be stopped during active mode of the device. If the software fails to clear the watchdog timer an internal reset will be initiated. The reset cause (external reset or reset caused by the watchdog) can be examined by software (status flag WDTS in IP0 is set). A refresh of the watchdog timer is done by setting bits WDT (SFR IEN0) and SWDT consecutively. This double instruction sequence has been implemented to increase system security.

It must be noted, however, that the watchdog timer is halted during the idle mode and power-down mode of the processor (see section "Power Saving Modes"). It is not possible to use the idle mode in combination with the watchdog timer function. Therefore, even the watchdog timer cannot reset the device when one of the power saving modes has been entered accidentally.

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 refresh 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 C505 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 eamine from which source the reset was activated. The watchdog timer status flag can also be cleared by software.

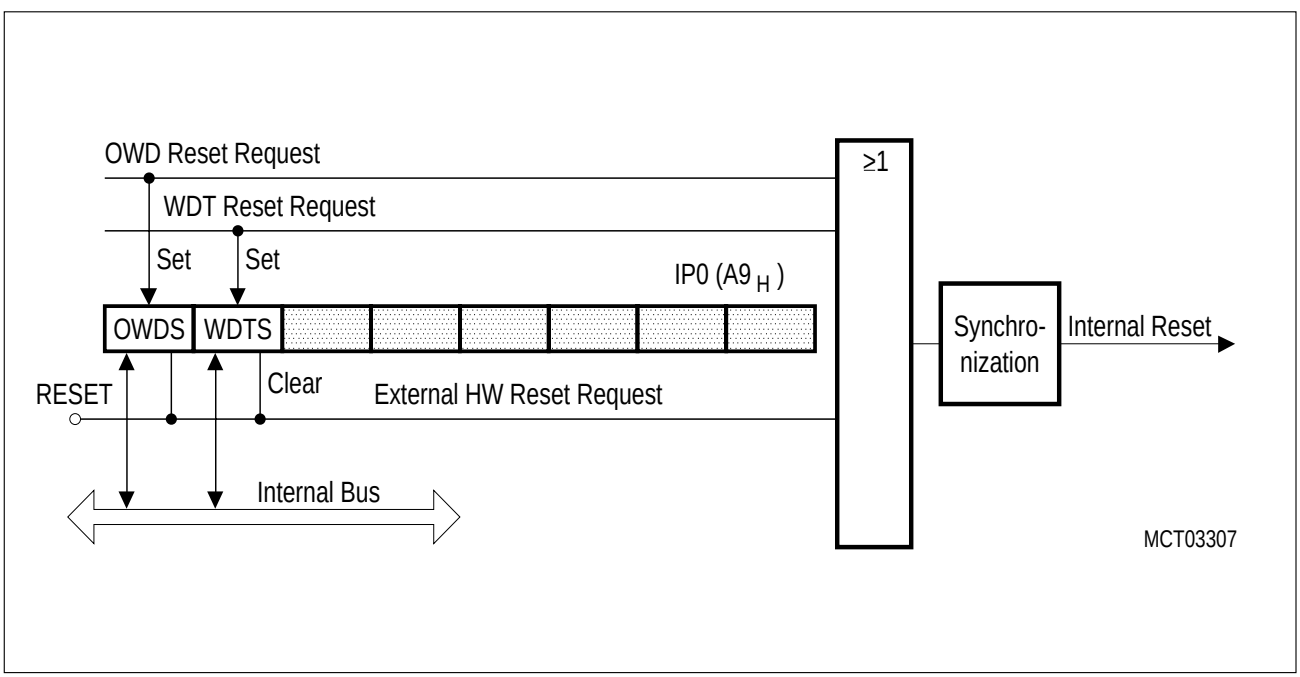


Figure 8-2
Watchdog Timer Status Flags and Reset Requests

8.2 Oscillator Watchdog Unit

The oscillator watchdog unit serves for three 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, in order to allow the oscillator to stabilize, executes a final reset phase of typ. 1 ms; then the oscillator watchdog reset is released and the part starts program execution from address 0000_H 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.
- **Control of external wake-up from software power-down mode**
  When the power-down mode is left by a low level at the P3.2/$\overline{INT0}$ pin or the P4.1/RXDC pin, the oscillator watchdog unit assures that the microcontroller resumes operation (execution of the power-down wake-up interrupt) with the nominal clock rate. In the power-down mode the RC oscillator and the on-chip oscillator are stopped. Both oscillators are started again when power-down mode is released. When the on-chip oscillator has a higher frequency than the RC oscillator, the microcontroller starts program execution by processing a power down interrupt after a final delay of typ. 1 ms in order to allow the on-chip oscillator to stabilize.

Note: The oscillator watchdog unit is always enabled.

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 additional provisions for integration of the wake-up from power down mode.

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 Status Flag. Set by hardware when an oscillator watchdog reset occured. Can be set and cleared by software.

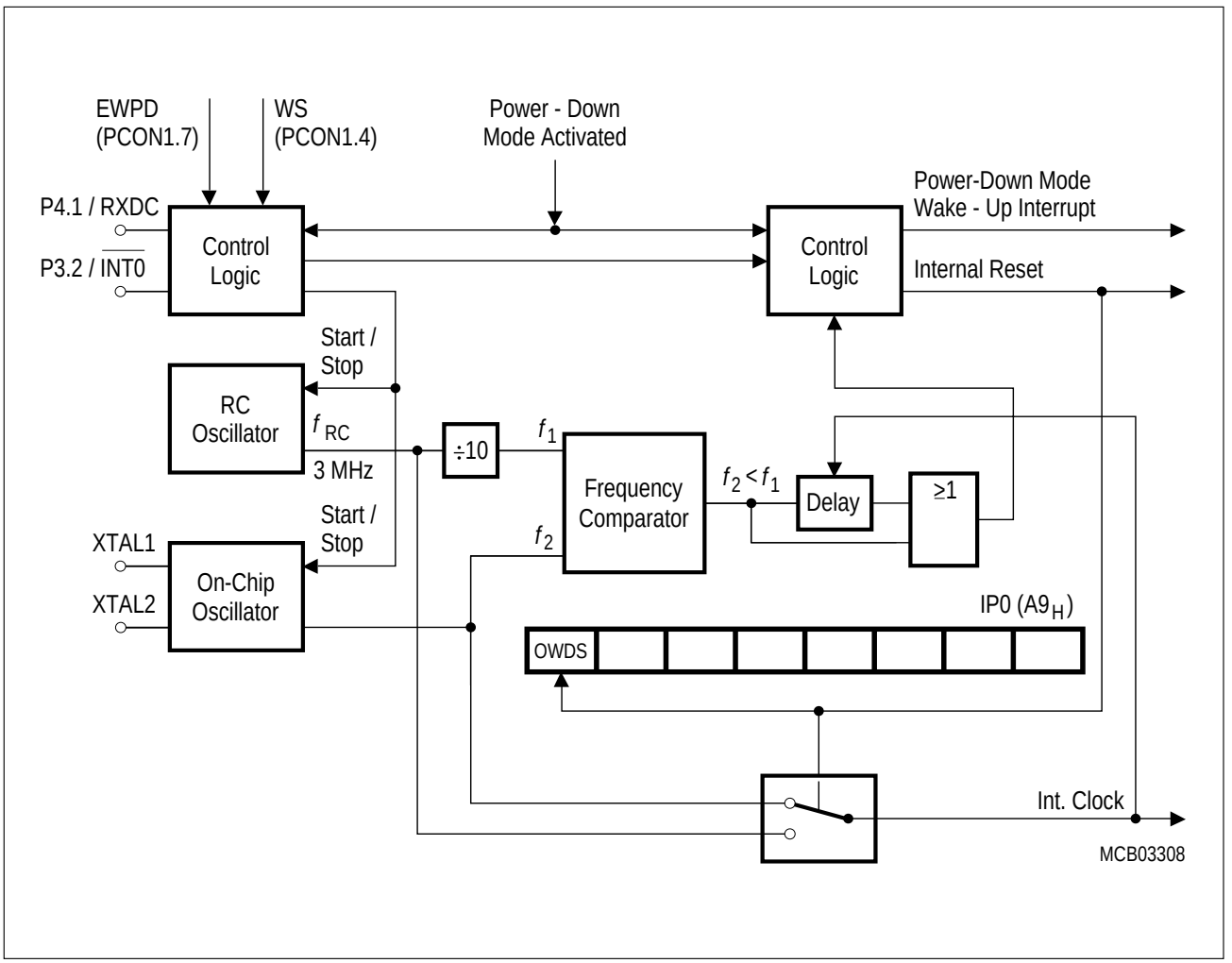


Figure 8-3
Functional Block Diagram of the Oscillator Watchdog

The frequency coming from the RC oscillator is divided by 10 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 Timer is, however, stopped); and bit OWDS is set. This allows the software to examine error conditions detected by the Watchdog unit 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 switches the clock supply back to the on-chip oscillator and releases the oscillator watchdog reset. If no other reset is applied at this time the part will start program execution. If an external reset or a watchdog timer reset is active, however, the device will retain the reset state until the other 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).

If software power-down mode is activated the RC oscillator and the on-chip oscillator are stopped. Both oscillators are again started in power-down mode when a low level is detected at either the P3.2/$\overline{\text{INT0}}$ input pin or the P4.1/RXDC pin and when bit EWPD in SFR PCON1 is set (wake-up from power-down mode enabled). The wake-up source is chosen from one of P3.2/$\overline{\text{INT0}}$ and P4.1/RXDC by bit WS in SFR PCON1. In this case the oscillator watchdog does not execute an internal reset during startup of the on-chip oscillator. After the start-up phase of the on-chip oscillator, the watchdog generates a power-down mode wake-up interrupt. Detailed description of the wake-up from software power-down mode is given in section 9.4.2.

Fast Internal Reset after Power-On

The C505 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) do not enter their default reset state 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 C505 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). Then 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).

9 Power Saving Modes

The C505 provides two basic power saving modes, the idle mode and the power down mode. Additionally, a slow down mode is available. This power saving mode reduces the internal clock rate in normal operating mode and it can be also used for further power reduction in idle mode.

9.1 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 and PCON1. The SFR PCON is located at SFR address 87_H. PCON1 is located in the mapped SFR area (RMAP=1) at SFR address 88_H. Bit RMAP, which controls the access to the mapped SFR area, is located in SFR SYSCON ($B1_H$).

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. Furthermore, register PCON contains two general purpose flags. For example, the flag bits GF0 and GF1 can be used to give an indication if an interrupt occurred during normal operation or during an idle. For this, an instruction that activates idle can also set one or both flag bits. When idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

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

Special Function Register PCON1 (Mapped Address $88_{H)}$ **Reset Value : $0XX0XXXX_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
88_H	EWPD	–	–	WS	–	–	–	–	PCON1

Symbol	Function
EWPD	External wake-up from power down enable bit Setting EWPD before entering power down mode, enables the external wake-up from power down mode capability (more details see section 9.4.2).
WS	Wake-up from power-down source select WS = 0 : wake-up via pin P3.2/$\overline{INT0}$. WS = 1 : wake-up via pin P4.1/RXDC. Pin P3.2/$\overline{INT0}$ is selected as wake-up source after reset.
–	Reserved bits for future use. Read by CPU returns undefined values.

9.2 Idle Mode

In the idle mode the oscillator of the C505 continues to run, but the CPU is gated off from the clock signal. However, the interrupt system, the serial port, the A/D converter, the CAN controller (C505C only), 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_{CC}.

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 serial interface in case it cannot finish reception or transmission during normal operation. The control signals ALE and $\overline{\text{PSEN}}$ are held 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.

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. 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 had set 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.

9.3 Slow Down Mode Operation

In some applications, where power consumption and dissipation are 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 32.

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 1/32 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 terminated 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 had set the bits IDLS and SD. Nevertheless the slow down mode keeps enabled and if required has to be terminated by clearing the bit SD in the corresponding interrupt service routine or at any point in the program where the user no longer requires the slow-down mode power saving.
- 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.

9.4 Software Power Down Mode

In the software power down mode, the RC oscillator 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 power down mode. ALE and $\overline{\text{PSEN}}$ held at logic low level (see **table 9–1**).

In the power down mode of operation, V_{CC} can be reduced to minimize power consumption. It must be ensured, however, that V_{CC} is not reduced before the power down mode is invoked, and that V_{CC} is restored to its normal operating level before the power down mode is terminated.

The software power down mode can be left either by an active reset signal or by a low signal at one of the wake-up source pins. Using reset to leave power down mode puts the microcontroller with its SFRs into the reset state. Using either the P3.2/$\overline{\text{INT0}}$ pin or the P4.1/RXDC pin for power down mode exit starts the RC oscillator and the on-chip oscillator and maintains the state of the SFRs, which have been frozen when power down mode is entered. Leaving power down mode should not be done before V_{CC} is restored to its nominal operating level.

9.4.1 Invoking Software Power Down Mode

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 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 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 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 power down mode. When the double instruction sequence shown above is used, the power down mode can only be left by a reset operation. If the external wake-up from power down capability has also to be used, its function must be enabled using the following instruction sequence prior to executing the double instruction sequence shown above.

```
ORL     SYSCON,#00010000B   ;set RMAP
ORL     PCON1,#80H          ;enable wake-up from power down via P3.2/INT0
ANL     SYSCON,#11101111B   ;reset RMAP (for future SFR accesses)
```

Setting EWPD automatically disables all interrupts still maintaining all actual values of the interrupt enable bits. In the above sequence the value of register PCON1 should be modified for choosing a wake-up via the P4.1/RXDC (bit PCON1.4 should be set).

Note : Before entering the power down mode, an A/D conversion in progress must be stopped.

9.4.2 Exit from Software Power Down Mode

If 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 power down mode also restarts the RC oscillator and the on-chip oscillatror. The reset operation should not be activated before V_{CC} 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).

Figure 9-1 shows the procedure which must is executed when power down mode is left via the P3.2/$\overline{\text{INT0}}$ or the P4.1/RXDC wake-up capability.

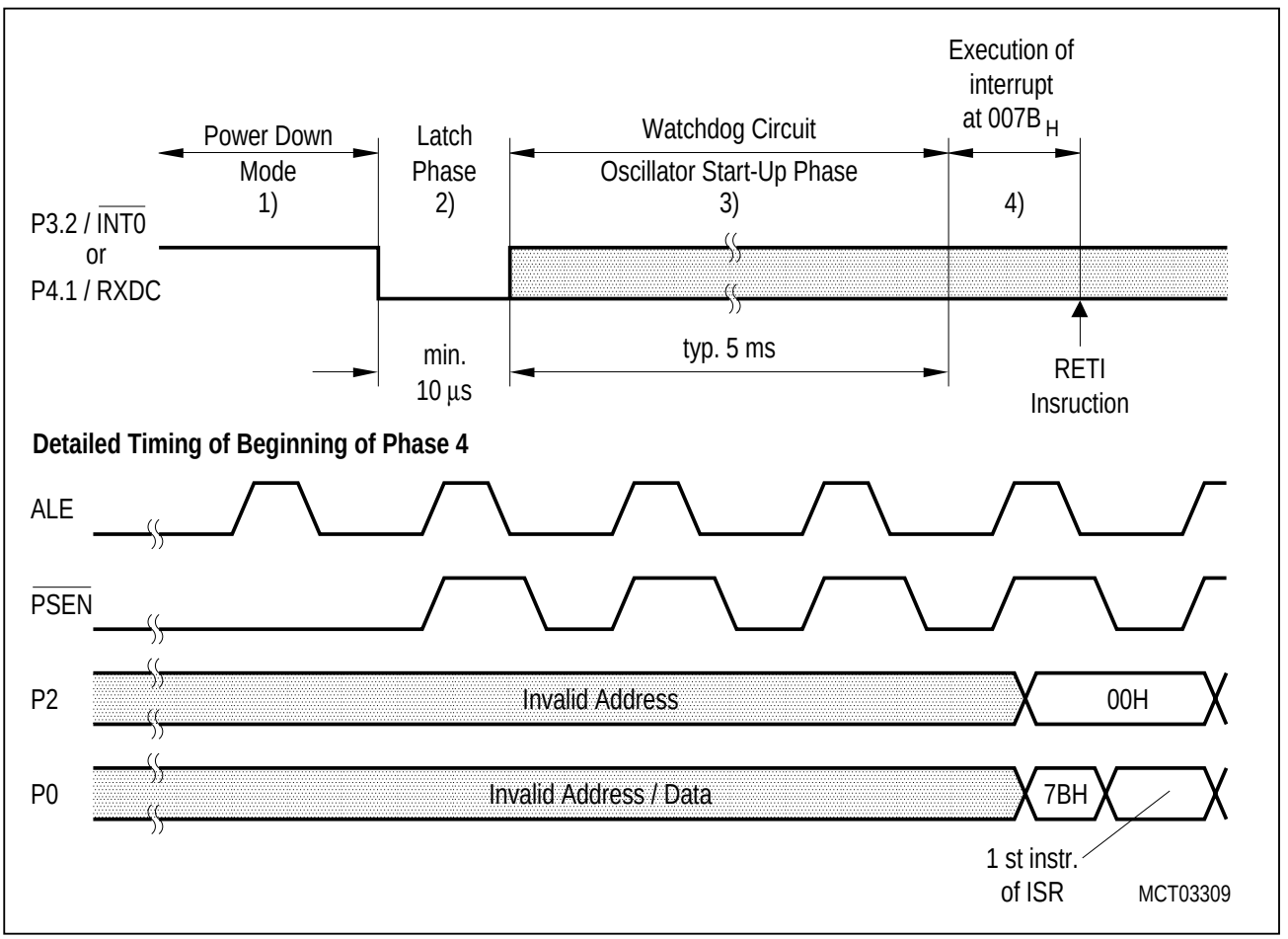


Figure 9-1
Wake-up from Power Down Mode Procedure

When the power down mode wake-up capability has been enabled (bit EWPD in SFR PCON1 set) prior to entering power down mode and bit WS in SFR PCON1 is cleared, the power down mode can be exit via $\overline{\text{INT0}}$ while executing the following procedure :

1. In power down mode pin P3.2/$\overline{\text{INT0}}$ must be held at high level.
2. Power down mode is left when P3.2/$\overline{\text{INT0}}$ goes low for at least 10 µs (latch phase). After this delay the internal RC oscillator and the on-chip oscillator are started, the state of pin P3.2/$\overline{\text{INT0}}$ is internally latched, and P3.2/$\overline{\text{INT0}}$ can be set again to high level if required. Thereafter, the oscillator watchdog unit controls the wake-up procedure in its start-up phase.

3. The oscillator watchdog unit starts operation. When the on-chip oscillator clock is detected for stable nominal frequency, the microcontroller starts again with its operation initiating the power down wake-up interrupt. The interrupt address of the first instruction to be executed after wake-up is $007B_H$. ALE and $\overline{PSEN}$ are in their power-down state up to this time. At the end of phase 3 the CPU processes the interrupt call and during these two machine cycles, ALE and $\overline{PSEN}$ behave as shown in **figure 9-1** (i.e. at the begining of phase 4). Instruction fetches during the interrupt call are, however, discarded.
4. After the RETI instruction of the power down wake-up interrupt routine has been executed, the instruction which follows the initiating power down mode double instruction sequence will be executed. The peripheral units timer 0/1/2 , CAN controller, and WDT are frozen until end of phase 4.

All interrupts of the C505 are disabled from phase 2) until the end of phase 4). Other Interrupts can be first handled after the RETI instruction of the wake-up interrupt routine.

The procedure to exit the software power down mode via the P4.1/RXDC pin is identical to the above procedure except that in this case pin P4.1/RXDC replaces pin P3.2/$\overline{INT0}$, and bit WS in SFR PCON1 should be set prior to entering software power down mode.

9.5 State of Pins in Software Initiated Power Saving Modes

In the idle mode and in the power down mode the port pins of the C505 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
$\overline{PSEN}$	High	Low	High	Low
PORT 0	Data	Data	Float	Float
PORT 2	Data	Data	Address	Data
PORT 1, 3, 4	Data / alternate outputs	Data / last output	Data / alternate outputs	Data / last output

10 Device Specifications

10.1 Absolute Maximum Ratings

Ambient temperature under bias (T_A) ..... – 40 ˚C to + 125 ˚C
Storage temperature (T_{ST}) .....– 65 ˚C to + 150 ˚C
Voltage on V_{CC} pins with respect to ground (V_{SS}) .....– 0.5 V to 6.5 V
Voltage on any pin with respect to ground (V_{SS}).....– 0.5 V to V_{CC} + 0.5 V
Input current on any pin during overload condition.....– 10 mA to + 10 mA
Absolute sum of all input currents during overload condition .....| 100 mA |
Power dissipation.....TBD

Note: *Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage of the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for longer periods may affect device reliability. During overload conditions ($V_{IN} > V_{CC}$ or $V_{IN} < V_{SS}$) the Voltage on V_{CC} pins with respect to ground (V_{SS}) must not exceed the values defined by the absolute maximum ratings.*

10.2 DC Characteristics

V_{CC} = 5 V + 10%, – 15%; V_{SS} = 0 V

T_A = 0 to 70 °C for the SAB-C505
T_A = – 40 to 85 °C for the SAF-C505
T_A = – 40 to 110 °C for the SAH-C505
T_A = – 40 to 125 °C for the SAK-C505

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
Input low voltages					
all except $\overline{EA}$, RESET	V_{IL}	– 0.5	0.2 V_{CC} - 0.1	V	–
$\overline{EA}$ pin	V_{IL1}	– 0.5	0.2 V_{CC} - 0.3	V	–
RESET pin	V_{IL2}	– 0.5	0.2 V_{CC} + 0.1	V	–
Input high voltages					
all except XTAL1, RESET	V_{IH}	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V	–
XTAL1 pin	V_{IH1}	0.7 V_{CC}	V_{CC} + 0.5	V	–
RESET pin	V_{IH2}	0.6 V_{CC}	V_{CC} + 0.5	V	–
Output low voltages					
Ports 1, 2, 3, 4	V_{OL}	–	0.45	V	I_{OL} = 1.6 mA [1]
Port 0, ALE, $\overline{PSEN}$	V_{OL1}	–	0.45	V	I_{OL} = 3.2 mA [1]
Output high voltages					
Ports 1, 2, 3, 4	V_{OH}	2.4	–	V	I_{OH} = – 80 µA
		0.9 V_{CC}	–	V	I_{OH} = – 10 µA
Port 0 in external bus mode,	V_{OH2}	2.4	–	V	I_{OH} = – 800 µA
ALE, $\overline{PSEN}$		0.9 V_{CC}	–	V	I_{OH} = – 80 µA [2]
Logic 0 input current					
Ports 1, 2, 3, 4	I_{IL}	– 10	– 70	µA	V_{IN} = 0.45 V
Logical 0-to-1 transition current Ports 1, 2, 3, 4	I_{TL}	– 65	– 650	µA	V_{IN} = 2 V
Input leakage current					
Port 0, AN0-7 (Port 1), $\overline{EA}$	I_{LI}	–	± 1	µA	0.45 < V_{IN} < V_{CC}
Pin capacitance	C_{IO}	–	10	pF	f_c = 1 MHz, T_A = 25 °C
Overload current	I_{OV}	–	± 5	mA	[8] [9]

Power Supply Current

Parameter		Symbol	Limit Values		Unit	Test Condition
			typ. [10]	max.[11]		
Active Mode	16 MHz 20 MHz	I_{CC} I_{CC}	26 32	TBD TBD	mA	V_{CC} = 5 V [4]
Idle Mode	16 MHz 20 MHz	I_{CC} I_{CC}	14.8 17.8	TBD TBD	mA	V_{CC} = 5 V [5]
Active Mode with slow-down enabled	16 MHz 20 MHz	I_{CC} I_{CC}	TBD TBD	TBD TBD	mA	V_{CC} = 5 V [6]
Idle Mode with slow-down enabled	16 MHz 20 MHz	I_{CC} I_{CC}	TBD TBD	TBD TBD	mA	V_{CC} = 5 V [7]
Power down current		I_{PD}	TBD	TBD	µA	V_{CC} = 2…5.5 V [3]

1) Capacitive loading on ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL} of ALE and port 3. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operation. In the worst case (capacitive loading > 100 pF), the noise pulse on ALE line may exceed 0.8 V. In such cases it may be desirable to qualify ALE with a schmitt-trigger, or use an address latch with a schmitt-trigger strobe input.

2) Capacitive loading on ports 0 and 2 may cause the V_{OH} on ALE and $\overline{PSEN}$ to momentarily fall below the 0.9 V_{CC} specification when the address lines are stabilizing.

3) I_{PD} (power-down mode) is measured under following conditions:
$\overline{EA}$ = Port 0 = V_{SS} ; RESET = V_{CC} ; XTAL2 = N.C.; XTAL1 = V_{SS} ; V_{AGND} = V_{SS} ; V_{AREF} = V_{CC} ; all other pins are disconnected.

4) I_{CC} (active mode) is measured with:
XTAL1 driven with t_R , t_F = 5 ns , V_{IL} = V_{SS} + 0.5 V, V_{IH} = V_{CC} – 0.5 V; XTAL2 = N.C.;
$\overline{EA}$ = Port 0 = V_{CC} ; RESET = V_{SS} ; all other pins are disconnected. I_{CC} would be slightly higher if the crystal oscillator is used (approx. 1 mA).

5) I_{CC} (idle mode) is measured with all output pins disconnected and with all peripherals disabled;
XTAL1 driven with t_R , t_F = 5 ns, V_{IL} = V_{SS} + 0.5 V, V_{IH} = V_{CC} – 0.5 V; XTAL2 = N.C.;
RESET = $\overline{EA}$ = V_{SS} ; Port0 = V_{CC} ; all other pins are disconnected;

6) I_{CC} (active mode with slow-down mode) is measured : TBD

7) I_{CC} (idle mode with slow-down mode) is measured : TBD

8) Overload conditions occur if the standard operating conditions are exceeded, ie. the voltage on any pin exceeds the specified range (i.e. V_{OV} > V_{CC} + 0.5 V or V_{OV} < V_{SS} – 0.5 V). The supply voltage V_{CC} and V_{SS} must remain within the specified limits. The absolute sum of input currents on all port pins may not exceed 50 mA.

9) Not 100% tested, guaranteed by design characterization

10) The typical I_{CC} values are periodically measured at T_A = + 25 °C but not 100% tested.

11) The maximum I_{CC} values are measured under worst case conditions (T_A = 0 °C or – 40 °C and V_{CC} = 5.5 V).

10.3 A/D Converter Characteristics

V_{CC} = 5 V + 10%, – 15%; V_{SS} = 0 V

T_A = 0 to 70 °C for the SAB-C505
T_A = – 40 to 85 °C for the SAF-C505
T_A = – 40 to 110 °C for the SAH-C505
T_A = – 40 to 125 °C for the SAK-C505

4 $V \le V_{AREF} \le V_{CC} + 0.1\,V$; $V_{SS} - 0.1\,V \le V_{AGND} \le V_{SS} + 0.2\,V$

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
Analog input voltage	V_{AIN}	V_{AGND} - 0.2	V_{AREF} + 0.2	V	1)
Sample time	t_S	–	64 x t_{IN} 32 x t_{IN} 16 x t_{IN} 8 x t_{IN}	ns	Prescaler ÷ 32 Prescaler ÷ 16 Prescaler ÷ 8 Prescaler ÷ 4 2)
Conversion cycle time	t_{ADCC}	–	320 x t_{IN} 160 x t_{IN} 80 x t_{IN} 40 x t_{IN}	ns	Prescaler ÷ 32 Prescaler ÷ 16 Prescaler ÷ 8 Prescaler ÷ 4 3)
Total unadjusted error	T_{UE}	–	± 2	LSB	V_{SS}+0.5V ≤ V_{AIN} ≤ V_{CC}-0.5V 4)
Internal resistance of reference voltage source	R_{AREF}	–	t_{ADC} / 500 - 1	kΩ	t_{ADC} in [ns] 5) 6)
Internal resistance of analog source	R_{ASRC}	–	t_S / 500 - 1	kΩ	t_S in [ns] 2) 6)
ADC input capacitance	C_{AIN}	–	50	pF	6)

Notes see next page.

Clock calculation table :

Clock Prescaler Ratio	ADCL1, 0		t_{ADC}	t_S	t_{ADCC}
÷ 32	1	1	32 x t_{IN}	64 x t_{IN}	320 x t_{IN}
÷ 16	1	0	16 x t_{IN}	32 x t_{IN}	160 x t_{IN}
÷ 8	0	1	8 x t_{IN}	16 x t_{IN}	80 x t_{IN}
÷ 4	0	0	4 x t_{IN}	8 x t_{IN}	40 x t_{IN}

Further timing conditions : t_{ADC} min = 800 ns
$t_{IN} = 1 / f_{OSC} = t_{CLP}$

Notes:

1) V_{AIN} may exeed V_{AGND} or V_{AREF} up to the absolute maximum ratings. However, the conversion result in these cases will be 00_H or FF_H, respectively.

2) During the sample time the input capacitance C_{AIN} must be charged/discharged by the external source. The internal resistance of the analog source must allow the capacitance to reach their final voltage level within t_S. After the end of the sample time t_S, changes of the analog input voltage have no effect on the conversion result.

3) This parameter includes the sample time t_S, the time for determining the digital result. Values for the conversion clock t_{ADC} depend on programming and can be taken from the table on the previous page.

4) T_{UE} (max.) is tested at $-40 \le T_A \le 125$ °C ; $V_{CC} \le$ 5.5 V; $V_{AREF} \le V_{CC}$ + 0.1 V and $V_{SS} \le V_{AGND}$. It is guaranteed by design characterization for all other voltages within the defined voltage range.
If an overload condition occurs on maximum 2 unused analog input pins and the absolute sum of input overload currents on all analog input pins does not exceed 10 mA, an additional conversion error of 1/2 LSB is permissible.

5) During the conversion the ADC's capacitance must be repeatedly charged or discharged. The internal resistance of the reference source must allow the capacitance to reach their final voltage level within the indicated time. The maximum internal resistance results from the programmed conversion timing.

6) Not 100% tested, but guaranteed by design characterization.

10.4 AC Characteristics (16 MHz) for C505

V_{CC} = 5 V + 10%, – 15%; V_{SS} = 0 V T_A = 0 to 70 °C for the SAB-C505
T_A = – 40 to 85 °C for the SAF-C505

(C_L for port 0, ALE and $\overline{PSEN}$ outputs = 100 pF; C_L for all other outputs = 80 pF)

Program Memory Characteristics

Parameter	Symbol	Limit Values				Unit
		16-MHz clock Duty Cycle 0.4 to 0.6		Variable Clock 1/CLP = 2 MHz to 16 MHz		
		min.	max.	min.	max.	
ALE pulse width	t_{LHLL}	48	–	CLP - 15	–	ns
Address setup to ALE	t_{AVLL}	10	–	TCL_{Hmin} -15	–	ns
Address hold after ALE	t_{LLAX}	10	–	TCL_{Hmin} -15	–	ns
ALE to valid instruction in	t_{LLIV}	–	75	–	2 CLP - 50	ns
ALE to $\overline{PSEN}$	t_{LLPL}	10	–	TCL_{Lmin} -15	–	ns
$\overline{PSEN}$ pulse width	t_{PLPH}	73	–	CLP+ TCL_{Hmin} -15	–	ns
$\overline{PSEN}$ to valid instruction in	t_{PLIV}	–	38	–	CLP+ TCL_{Hmin}- 50	ns
Input instruction hold after $\overline{PSEN}$	t_{PXIX}	0	–	0	–	ns
Input instruction float after $\overline{PSEN}$	t_{PXIZ} *)	–	15	–	TCL_{Lmin} -10	ns
Address valid after $\overline{PSEN}$	t_{PXAV} *)	20	–	TCL_{Lmin} - 5	–	ns
Address to valid instruction in	t_{AVIV}	–	95	–	2 CLP + TCL_{Hmin} -55	ns
Address float to $\overline{PSEN}$	t_{AZPL}	-5	–	-5	–	ns

*) Interfacing the C505 to devices with float times up to 20 ns is permissible. This limited bus contention will not cause any damage to port 0 drivers.

AC Characteristics (16 MHz) for C505 (cont'd)

External Data Memory Characteristics

Parameter	Symbol	Limit Values				Unit
		16-MHz clock Duty Cycle 0.4 to 0.6		Variable Clock 1/CLP= 2 MHz to 16 MHz		
		min.	max.	min.	max.	
$\overline{RD}$ pulse width	t_{RLRH}	158	–	3 CLP - 30	–	ns
$\overline{WR}$ pulse width	t_{WLWH}	158	–	3 CLP - 30	–	ns
Address hold after ALE	t_{LLAX2}	48	–	CLP- 15	–	ns
$\overline{RD}$ to valid data in	t_{RLDV}	–	100	–	2 CLP+ TCL_{Hmin} - 50	ns
Data hold after $\overline{RD}$	t_{RHDX}	0	–	0	–	ns
Data float after $\overline{RD}$	t_{RHDZ}	–	51	–	CLP - 12	ns
ALE to valid data in	t_{LLDV}	–	200	–	4 CLP - 50	ns
Address to valid data in	t_{AVDV}	–	200	–	4 CLP + TCL_{Hmin} -75	ns
ALE to $\overline{WR}$ or $\overline{RD}$	t_{LLWL}	73	103	CLP + TCL_{Lmin} - 15	CLP+ TCL_{Lmin}+ 15	ns
Address valid to $\overline{WR}$	t_{AVWL}	95	–	2 CLP - 30	–	ns
$\overline{WR}$ or $\overline{RD}$ high to ALE high	t_{WHLH}	10	40	TCL_{Hmin} - 15	TCL_{Hmin} + 15	ns
Data valid to $\overline{WR}$ transition	t_{QVWX}	5	–	TCL_{Lmin} - 20	–	ns
Data setup before $\overline{WR}$	t_{QVWH}	163	–	3 CLP + TCL_{Lmin} - 50	–	ns
Data hold after $\overline{WR}$	t_{WHQX}	5	–	TCL_{Hmin} - 20	–	ns
Address float after $\overline{RD}$	t_{RLAZ}	–	0	–	0	ns

AC Characteristics (16 MHz) for C505 (cont'd)

External Clock Drive Characteristics

Parameter	Symbol	CPU Clock = 16 MHz Duty Cycle 0.4 to 0.6		Variable CPU Clock 1/CLP = 2 to 16 MHz		Unit
		min.	max.	min.	max.	
Oscillator period	CLP	62.5	62.5	62.5	500	ns
High time	TCL_H	25	–	25	CLP - TCL_L	ns
Low time	TCL_L	25	–	25	CLP - TCL_H	ns
Rise time	t_R	–	10	–	10	ns
Fall time	t_F	–	10	–	10	ns
Oscillator duty cycle	DC	0.4	0.6	25 / CLP	1 - 25 / CLP	–
Clock cycle	TCL	25	37.5	CLP * DC_{min}	CLP * DC_{max}	ns

Note: The 16 MHz values in the tables are given as an example for a typical duty cycle variation of the oscillator clock from 0.4 to 0.6.

10.5 AC Characteristics (20 MHz) for C505

V_{CC} = 5 V + 10%, – 15%; V_{SS} = 0 V T_A = 0 to 70 °C for the SAB-C505
T_A = – 40 to 85 °C for the SAF-C505

(C_L for port 0, ALE and $\overline{PSEN}$ outputs = 100 pF; C_L for all other outputs = 80 pF)

Program Memory Characteristics

Parameter	Symbol	Limit Values				Unit
		20-MHz clock Duty Cycle 0.5 to 0.5		Variable Clock 1/CLP = 2 MHz to 20 MHz		
		min.	max.	min.	max.	
ALE pulse width	t_{LHLL}	35	–	CLP - 15	–	ns
Address setup to ALE	t_{AVLL}	10	–	TCL_{Hmin} - 15	–	ns
Address hold after ALE	t_{LLAX}	10	–	TCL_{Hmin} - 15	–	ns
ALE low to valid instr in	t_{LLIV}	–	55	–	2 CLP - 45	ns
ALE to $\overline{PSEN}$	t_{LLPL}	10	–	TCL_{Lmin} - 15	–	ns
$\overline{PSEN}$ pulse width	t_{PLPH}	60	–	CLP + TCL_{Hmin} - 15	–	ns
$\overline{PSEN}$ to valid instr in	t_{PLIV}	–	25	–	CLP + TCL_{Hmin} - 50	ns
Input instruction hold after $\overline{PSEN}$	t_{PXIX}	0	–	0	–	ns
Input instruction float after $\overline{PSEN}$	t_{PXIZ} *)	–	20	–	TCL_{Lmin} - 5	ns
Address valid after $\overline{PSEN}$	t_{PXAV} *)	20	–	TCL_{Lmin} - 5	–	ns
Address to valid instr in	t_{AVIV}	–	65	–	2 CLP + TCL_{Hmin} - 60	ns
Address float to $\overline{PSEN}$	t_{AZPL}	- 5	–	- 5	–	ns

*) Interfacing the C505 to devices with float times up to 25 ns is permissible. This limited bus contention will not cause any damage to port 0 drivers.

AC Characteristics (20 MHz) for C505 (cont'd)

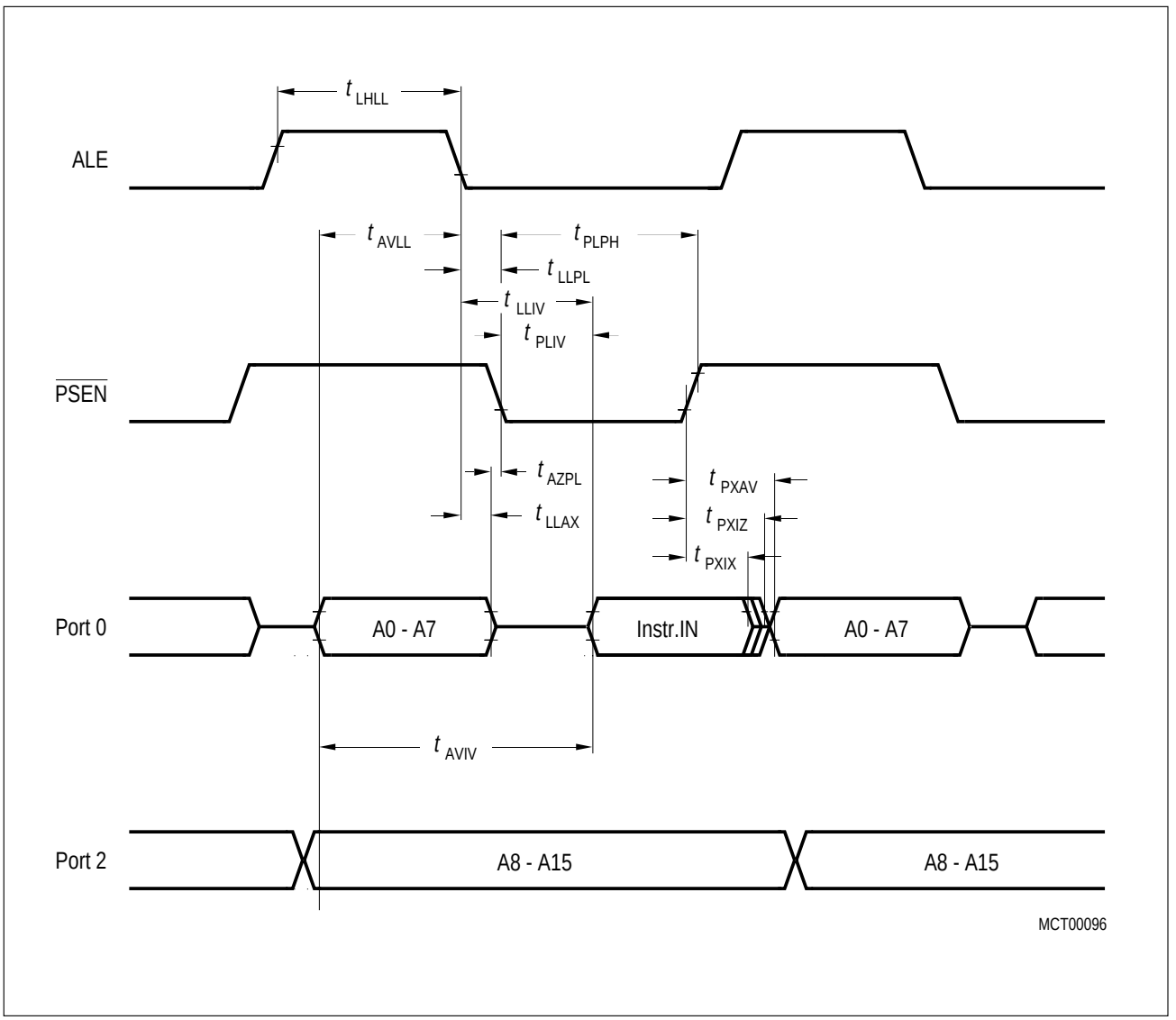
External Data Memory Characteristics

Parameter	Symbol	Limit Values				Unit
		20-MHz clock Duty Cycle 0.5 to 0.5		Variable Clock 1/CLP = 2 MHz to 20 MHz		
		min.	max.	min.	max.	
$\overline{RD}$ pulse width	t_{RLRH}	120	–	3 CLP-30	–	ns
$\overline{WR}$ pulse width	t_{WLWH}	120	–	3 CLP-30	–	ns
Address hold after ALE	t_{LLAX2}	35	–	CLP-15	–	ns
$\overline{RD}$ to valid data in	t_{RLDV}	–	75	–	2 CLP+ TCL_{Hmin}-50	ns
Data hold after $\overline{RD}$	t_{RHDX}	0	–	0	–	ns
Data float after $\overline{RD}$	t_{RHDZ}	–	38	–	CLP-12	ns
ALE to valid data in	t_{LLDV}	–	150	–	4 CLP-50	ns
Address to valid data in	t_{AVDV}	–	150	–	4 CLP + TCL_{Hmin}-75	ns
ALE to $\overline{WR}$ or $\overline{RD}$	t_{LLWL}	60	90	CLP + TCL_{Lmin}-15	CLP + TCL_{Lmin}+15	ns
Address valid to $\overline{WR}$	t_{AVWL}	70	–	2 CLP-30	–	ns
$\overline{WR}$ or $\overline{RD}$ high to ALE high	t_{WHLH}	10	40	TCL_{Hmin}-15	TCL_{Hmin}+15	ns
Data valid to $\overline{WR}$ transition	t_{QVWX}	5	–	TCL_{Lmin}-20	–	ns
Data setup before $\overline{WR}$	t_{QVWH}	125	–	3 CLP+ TCL_{Lmin}-50	–	ns
Data hold after $\overline{WR}$	t_{WHQX}	5	–	TCL_{Hmin}-20	–	ns
Address float after $\overline{RD}$	t_{RLAZ}	–	0	–	0	ns

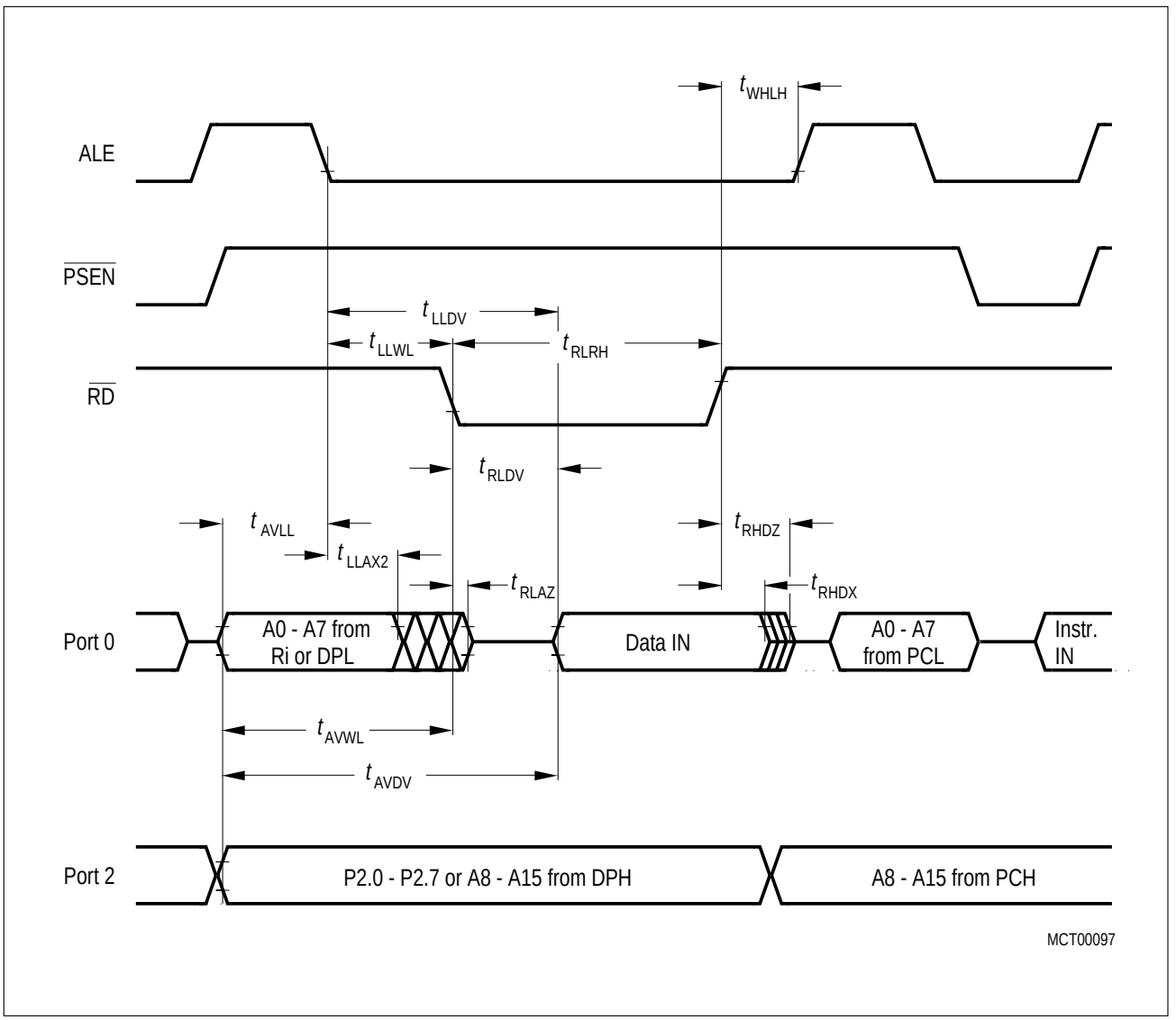
External Clock Drive Characteristics

Parameter	Symbol	Limit Values		Unit
		Variable Clock Freq. = 2 MHz to 20 MHz		
		min.	max.	
Oscillator period	CLP	50	500	ns
High time	TCL_H	25	CLP-TCL_L	ns
Low time	TCL_L	25	CLP-TCL_H	ns
Rise time	t_R	–	10	ns
Fall time	t_F	–	10	ns

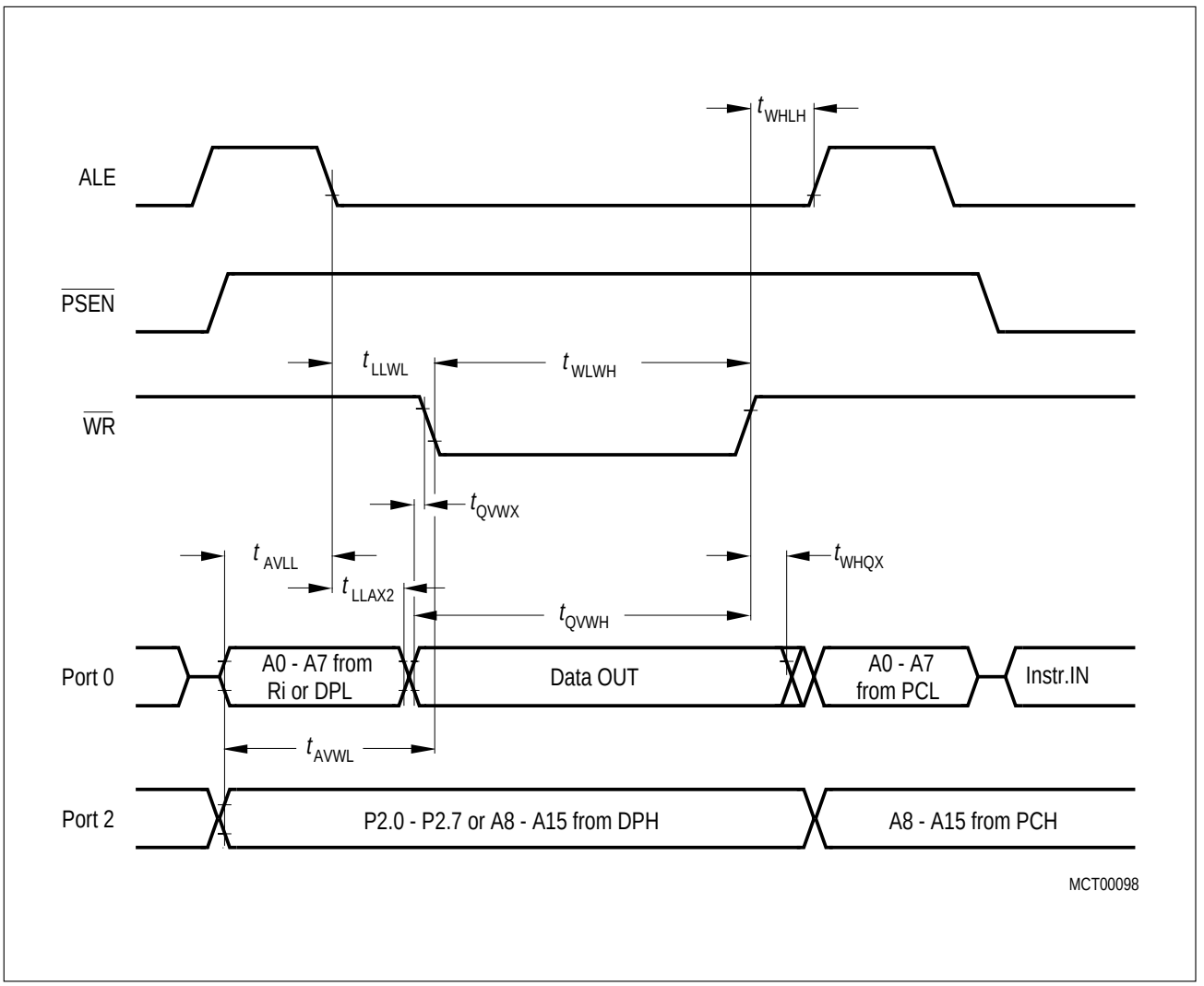
Note: The 20 MHz values in the tables are given as an example for a typical duty cycle of the oscillator clock of 50 %.



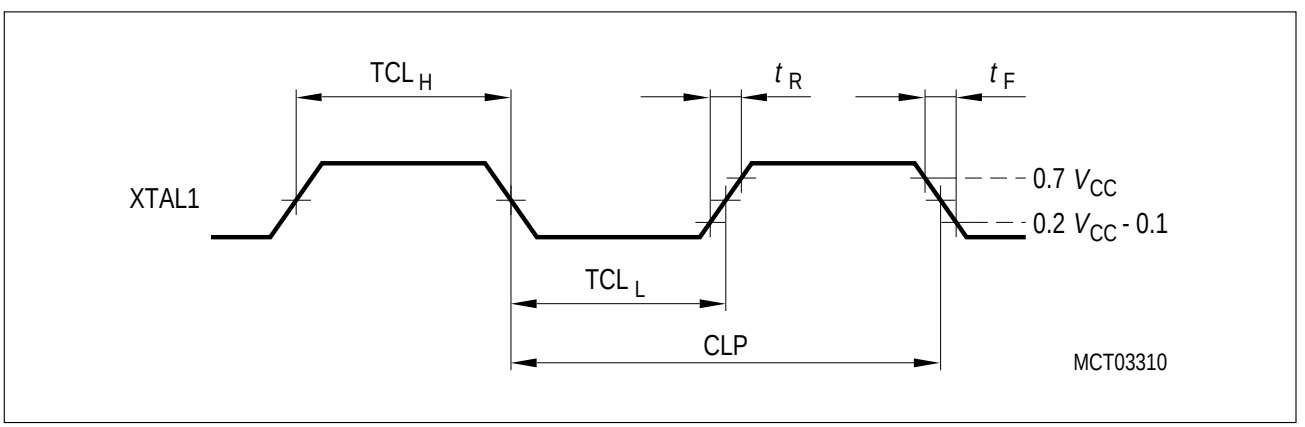
Program Memory Read Cycle



Data Memory Read Cycle



Data Memory Write Cycle

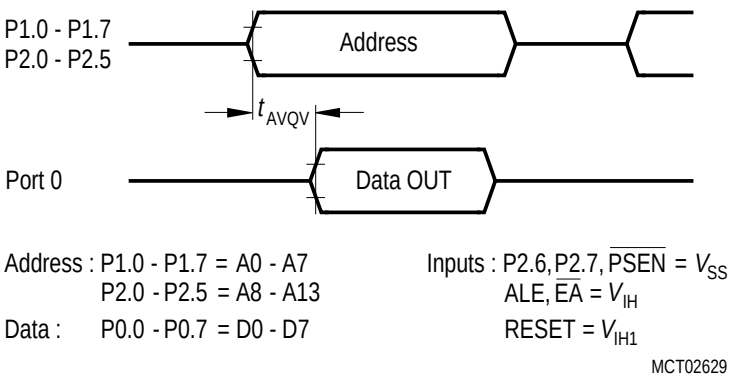


External Clock Drive on XTAL1

10.6 ROM Verification Characteristics for C505-2R

ROM Verification Mode 1

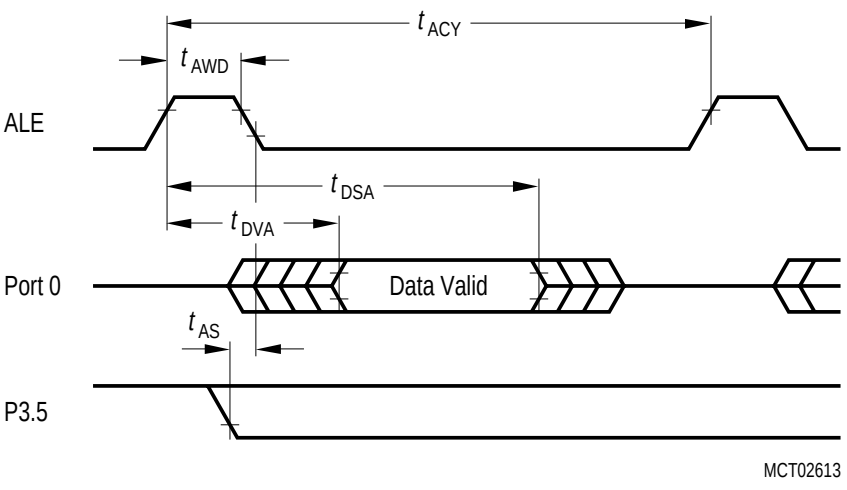
Parameter	Symbol	Limit Values		Unit
		min.	max.	
Address to valid data	t_{AVQV}	–	5 CLP	ns



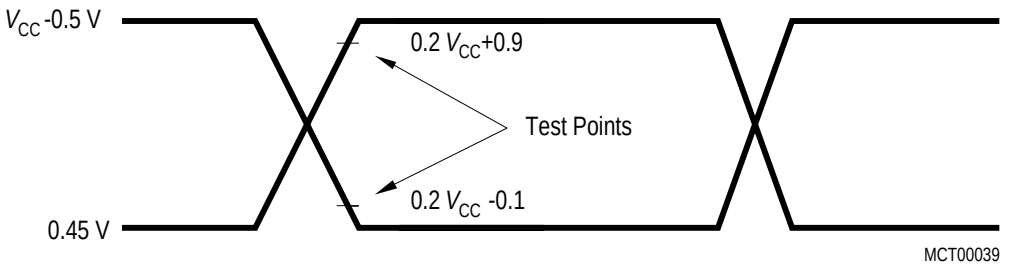
ROM Verification Mode 1

ROM Verification Mode 2

Parameter	Symbol	Limit Values			Unit
		min.	typ	max.	
ALE pulse width	t_{AWD}	–	CLP	–	ns
ALE period	t_{ACY}	–	6 CLP	–	ns
Data valid after ALE	t_{DVA}	–	–	2 CLP	ns
Data stable after ALE	t_{DSA}	4 CLP	–	–	ns
P3.5 setup to ALE low	t_{AS}	–	TCL_H	–	ns
Oscillator frequency	1/ CLP	4	–	6	MHz

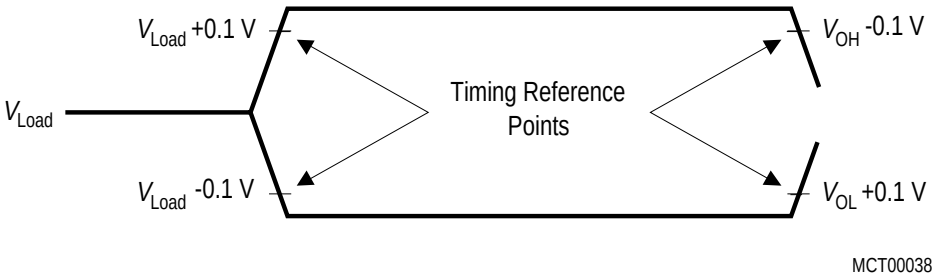


ROM Verification Mode 2



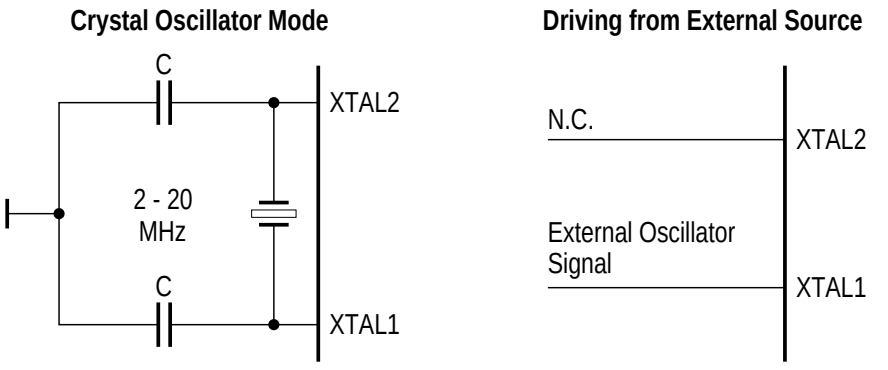
AC Inputs during testing are driven at V_{CC} - 0.5 V for a logic '1' and 0.45 V for a logic '0'. Timing measurements are made at V_{IHmin} for a logic '1' and V_{ILmax} for a logic '0'.

AC Testing: Input, Output Waveforms



For timing purposes a port pin is no longer floating when a 100 mV change from load voltage occurs and begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs. $I_{OL}/I_{OH} \geq \pm 20$ mA

AC Testing : Float Waveforms

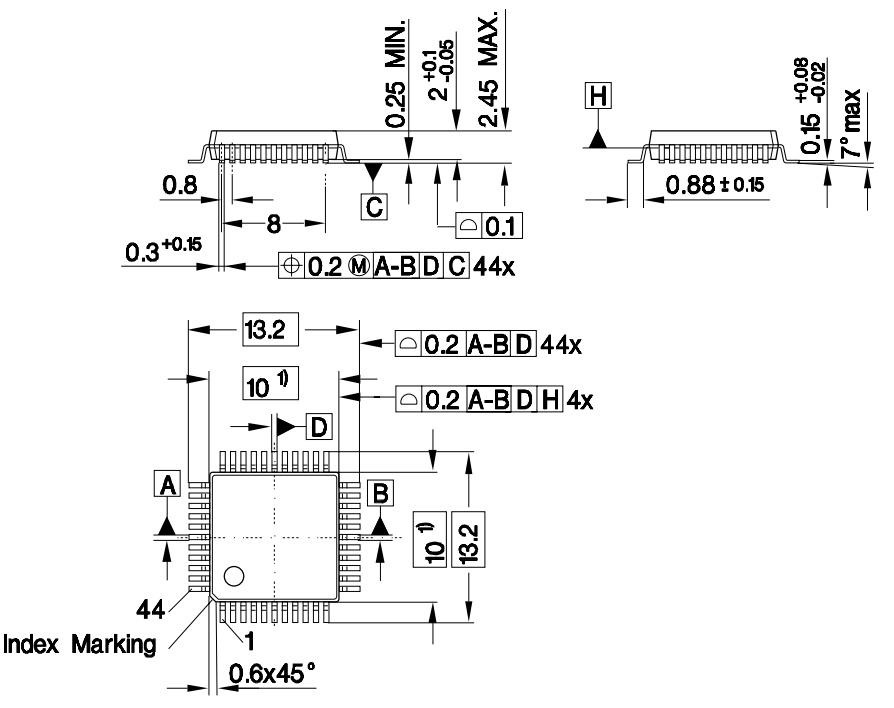


Crystal Mode: C = 20 pF ± 10 pF (incl. stray capacitance)

Recommended Oscillator Circuits for Crystal Oscillator

10.7 Package Information

Plastic Package, P-MQFP-44-2 (SMD)
(Plastic Metric Quad Flat Pack)



1) Does not include plastic or metal protrusion of 0.25 max. per side

GPM05622

Sorts of Packing
Package outlines for tubes, trays etc. are contained in our Data Book “Package Information”.
SMD = Surface Mounted Device

11 Index

Note: Bold page numbers refer to the main definition part of SFRs or SFR bits.