SIEMENS

Microcomputer Components

C509-L
8-Bit CMOS Microcontroller

User’s Manual   07.96

SIEMENS

Microcomputer Components

C509-L
8-Bit CMOS Microcontroller

User’s Manual   07.96

C509-L **Revision History: Current Version: 07.96**		
Previous Version:		
Page (in previous Version)	Page (in new Version)	Subjects (major changes since last revision)

Edition 07.96

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 (see address list).

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

Table of Contents Page

1 Introduction

The C509-L is a high-end microcontroller in the Siemens SAB-C500 8-bit microcontroller family. It is based on the well-known industry standard 8051 architecture; a great number of enhancements and new peripheral features extend its capabilities to meet the extensive requirements of new applications. Further, the C509-L is a superset of the Siemens SAB 80C517/80C517A 8-bit microcontroller thus offering an easy upgrade path for SAB 80C517/80C517A users.

The high performance of the C509-L microcontroller is achieved by the C500-Core with a maximum operating frequency of 16 MHz internal (and external) CPU clock. While maintaining all the features of the SAB 80C517A, the C509-L is expanded by one I/O port, in its compare/capture capabilities, by A/D converter functions, by additional 1 KByte of on-chip RAM (now 3 KByte XRAM) and by an additional user-selectable CMOS port structure. The C509-L is mounted in a P-MQFP-100-2 package.

Figure 1-1 shows the different functional units of the C509-L and **figure 1-2** shows the simplified logic symbol of the C509-L.

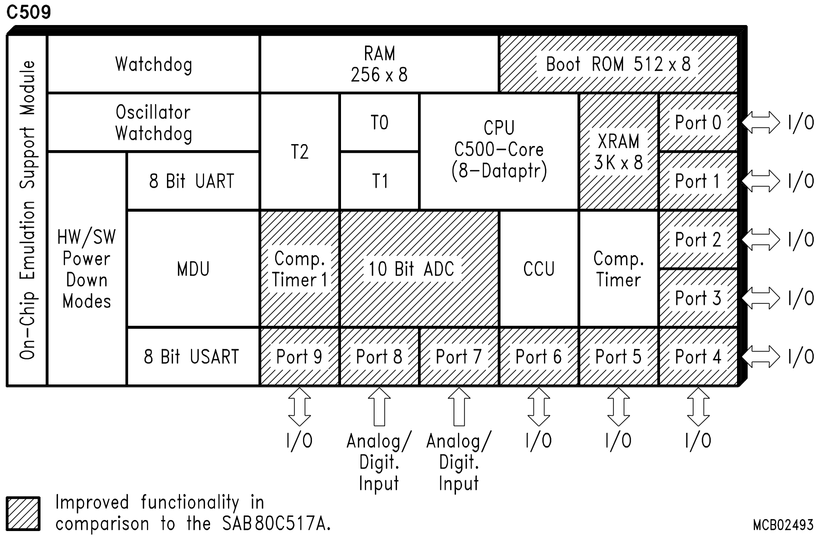


Figure 1-1
C509-L Functional Units

Listed below is a summary of the main features of the C509-L:

- Full upward compatibility with SAB 80C517/80C517A
- 256 byte on-chip RAM
- 3K byte of on-chip XRAM
- 256 directly addressable bits
- 375 ns instruction cycle at 16-MHz oscillator frequency
- 64 of 111 instructions are executed in one instruction cycle
- External program and data memory expandable up to 64 Kbyte each
- On-chip emulation support logic (Enhanced Hooks Technology ^{TM})
- 10-bit A/D converter
  - 15 multiplexed inputs
  - Built-in self calibration
- Two 16-bit timers/counters (8051 compatible)
- Three 16-bit timers/counters (can be used in combination with the compare/capture unit)
- Powerful compare/capture unit (CCU) with up to 29 high-speed or PWM output channels or 13 capture inputs
- Arithmetic unit for division, multiplication, shift and normalize operations
- Eight datapointers instead of one for indirect addressing of program and external data memory
- Extended watchdog facilities
  - 15-bit programmable watchdog timer
  - Oscillator watchdog
- Ten I/O ports
  - Eight bidirectional 8-bit I/O ports with selectable port structure
    quasi-bidirectional port structure (8051 compatible)
    bidirectional port structure with CMOS voltage levels
  - One 8-bit and one 7-bit input port for analog and digital input signals
- Two full-duplex serial interfaces with own baud rate generators
- Four priority level interrupt systems, 19 interrupt vectors
- Three power saving modes
  - Slow-down mode
  - Idle mode
  - Power-down mode
- Siemens high-performance ACMOS technology
- M-QFP-100 rectangular quad flat package
- Temperature Ranges : SAB-C509-L T_A = 0 to 70 °C
  SAF-C509-L T_A = -40 to 85 °C

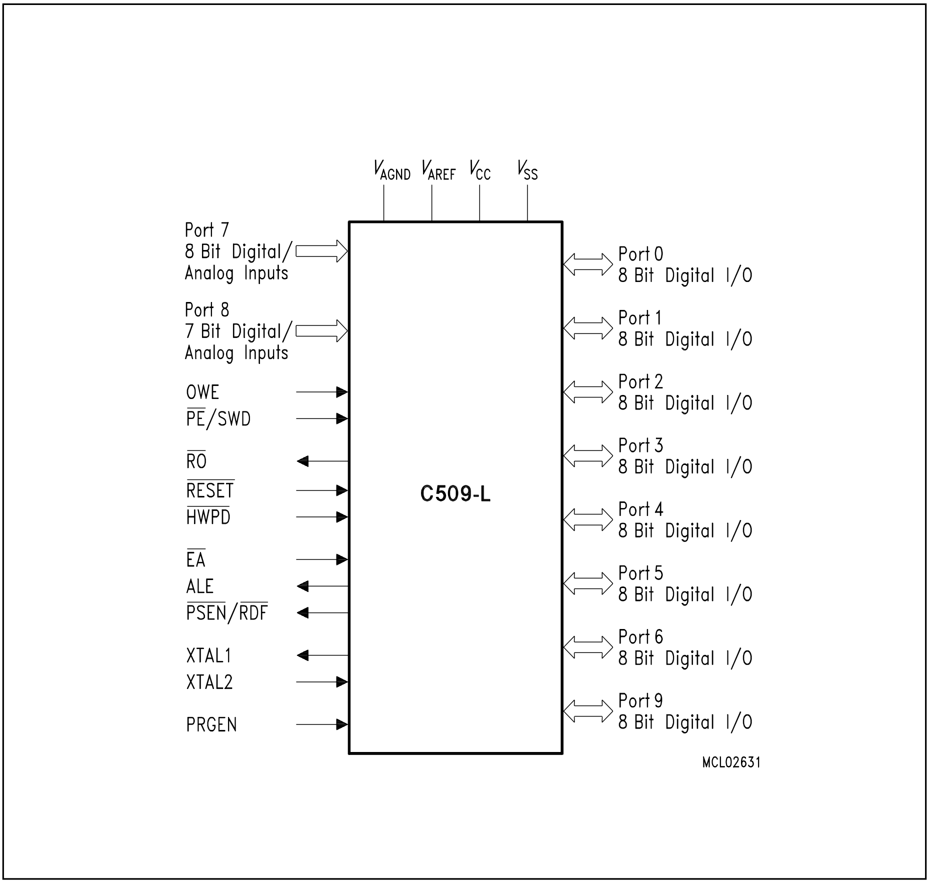


Figure 1-2
C509-L Logic Symbol

1.1 Pin Configuration

This section describes the pin configuration of the C509-L in the P-MQFP-100-2 package.

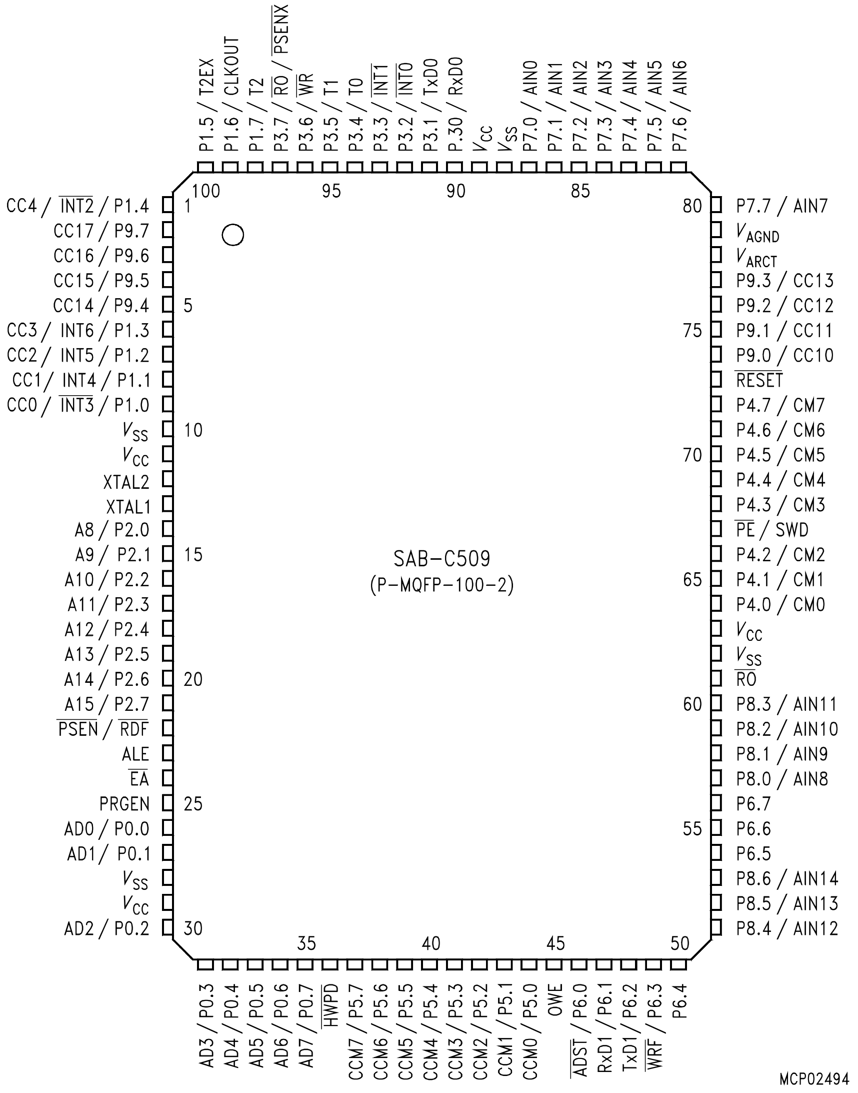


Figure 1-3
C509-L Pin Configuration (P-MQFP-100-2, top view)

1.2 Pin Definitions and Functions

This section describes all external signals of the C509-L with its function.

Table 1-1
Pin Definitions and Functions

Symbol	Pin Number	I/O*)	Function
P1.0 - P1.7	9-6, 1, 100-98	I/O	**Port 1** is an 8-bit bidirectional I/O port with internal pullup resistors. Port 1 pins that have 1's written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. As inputs, port 1 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 1 can also be switched into a bidirectional mode, in which CMOS levels are provided. In this bidirectional mode, each port 1 pin can be programmed individually as input or output. Port 1 also contains the interrupt, timer, clock, capture and compare pins that are used by various options. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate (except when used for the compare functions). The secondary functions are assigned to the pins of port 1 as follows :
	9		P1.0 $\overline{\text{INT3}}$ CC0 $\overline{\text{Interrupt 3 input}}$ / compare 0 output / capture 0 input
	8		P1.1 INT4 CC1 Interrupt 4 input / compare 1 output / capture 1 input
	7		P1.2 INT5 CC2 Interrupt 5 input / compare 2 output / capture 2 input
	6		P1.3 INT6 CC3 Interrupt 6 input / compare 3 output / capture 3 input
	1		P1.4 $\overline{\text{INT2}}$ CC4 $\overline{\text{Interrupt 2}}$ input / compare 4 output / capture 4 input
	100		P1.5 T2EX Timer 2 external reload trigger input
	99		P1.6 CLKOUT System clock output
	98		P1.7 T2 Counter 2 input

*) I = Input
O = Output

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

Symbol	Pin Number	I/O*)	Function
P9.0 - P9.7	74-77, 5-2	I/O	**Port 9** is an 8-bit bidirectional I/O port with internal pullup resistors. Port 9 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 9 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 9 can also be switched into a bidirectional mode, in which CMOS levels are provided. In this bidirectional mode, each port 1 pin can be programmed individually as input or output. Port 9 also serves alternate compare functions. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the pins of port 9 as follows : P9.0-P9.7 CC10-CC17 Compare/capture channel 0-7 output/input
XTAL2	12	–	**XTAL2** is the input to the inverting oscillator amplifier and input to the internal clock generator circuits. When supplying the C509-L with an external clock source, XTAL2 should be driven, while XTAL1 is left unconnected. A duty cycle of 0.4 to 0.6 of the clock signal is required. Minimum and maximum high and low times as well as rise/fall times specified in the AC characteristics must be observed.
XTAL1	13	–	**XTAL1** Output of the inverting oscillator amplifier. This pin is used for the oscillator operation with crystal or ceramic resonartor

*) I = Input
O = Output

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

Symbol	Pin Number	I/O*)	Function
P2.0 – P2.7	14-21	I/O	**Port 2** is a 8-bit I/O port. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullup resistors when issuing 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. P2.0 - P2.7 A8 - A15 Address lines 8 - 15
$\overline{\text{PSEN}}$ / $\overline{\text{RDF}}$	22	O	**$\overline{\text{Program Store Enable}}$ / $\overline{\text{Read FLASH}}$** The $\overline{\text{PSEN}}$ output is a control signal that enables the external program memory to the bus during external code fetch operations. It is activated every third oscillator period. $\overline{\text{PSEN}}$ is not activated during external data memory accesses caused by MOVX instructions. $\overline{\text{PSEN}}$ is not activated when instructions are executed from the internal Boot ROM or from the XRAM. In external programming mode $\overline{\text{RDF}}$ becomes active when executing external data memory read (MOVX) instructions.
ALE	23	O	**Address Latch Enable** This output is used for latching the low byte of the address into external memory during normal operation. It is activated every third oscillator period except during an external data memory access caused by MOVX instructions.
$\overline{\text{EA}}$	24	I	**$\overline{\text{External Access Enable}}$** The status of this pin is latched at the end of a reset. When held at low level, the C509-L fetches all instructions from the external program memory. For the C509-L this pin must be tied low.
PRGEN	25	I	**External Flash-EPROM Program Enable** A low level at this pin disables the programming of an external Flash-EPROM. To enable the programming of an external Flash-EPROM, the pin PRGEN must be held at high level and bit PRGEN1 in SFR SYSCON1 has to be set. There is no internal pullup resistor connected to this pin.

*) I = Input
O = Output

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

Symbol	Pin Number	I/O*)	Function
P0.0 – P0.7	26, 27, 30-35	I/O	**Port 0** is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1s 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 operating mode it uses strong internal pullup resistors when issuing 1 s. P0.0 - P0.7 AD0-AD7 Address/data lines 0 - 7
$\overline{\text{HWPD}}$	36	I	**$\overline{\text{Hardware Power Down}}$** A low level on this pin for the duration of one machine cycle while the oscillator is running resets the C509-L. A low level for a longer period will force the part to power down mode with the pins floating. There is no internal pullup resistor connected to this pin.
P5.0 - P5.7	44-37	I/O	**Port 5** is an 8-bit bidirectional I/O port with internal pullup resistors. Port 5 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 5 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 5 can also be switched into a bidirectional mode, in which CMOS levels are provided. In this bidirectional mode, each port 5 pin can be programmed individually as input or output. Port 5 also serves as alternate function for “Concurrent Compare” and "Set/Reset compare” functions. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the pins of port 5 as follows : P5.0 - P5.7 CCM0-CCM7 Concurrent Compare or Set/Reset lines 0 - 7

*) I = Input
O = Output

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

Symbol	Pin Number	I/O*)	Function
OWE	45	I	**Oscillator Watchdog Enable** A high level on this pin enables the oscillator watchdog. When left unconnected, this pin is pulled high by a weak internal pullup resistor. The logic level at OWE should not be changed during normal operation. When held at low level the oscillator watchdog function is turned off. During hardware power down the pullup resistor is switched off.
P6.0 - P6.7	46-50, 54-56	I/O	**Port 6** is an 8-bit bidirectional I/O port with internal pullup resistors. Port 6 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 6 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 6 can also be switched into a bidirectional mode, in which CMOS levels are provided. In this bidirectional mode, each port 6 pin can be programmed individually as input or output. Port 6 also contains the external A/D converter control pin, the receive and transmission lines for the serial port 1, and the write-FLASH control signal. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the pins of port 6 as follows :
	46		P6.0 $\overline{\text{ADST}}$ External A/D converter start pin
	47		P6.1 R×D1 Receiver data input of serial interface 1
	48		P6.2 T×D1 Transmitter data output of serial interface 1
	49		P6.3 $\overline{\text{WRF}}$ The $\overline{\text{WRF}}$ (write Flash) signal is active when the programming mode is selected. In this mode $\overline{\text{WRF}}$ becomes active when executing external data memory write (MOVX) instructions.

*) I = Input
O = Output

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

Symbol	Pin Number	I/O*)	Function
P8.0 - P8.6	57-60, 51-53	I	**Port 8** is a 7-bit unidirectional input port. Port pins can be used for digital input if voltage levels meet the specified input high/low voltages, and for the higher 7-bit of the multiplexed analog inputs of the A/D converter simultaneously. P8.0 - P8.6 AIN8 - AIN14 Analog input 8 - 14
$\overline{\text{RO}}$	61	O	**$\overline{\text{Reset Output}}$** This pin outputs the internally synchronized reset request signal. This signal may be generated by an external hardware reset, a watchdog timer reset or an oscillator watchdog reset. The $\overline{\text{RO}}$ output is active low.
P4.0 – P4.7	64-66, 68-72	I/O	**Port 4** is an 8-bit bidirectional I/O port with internal pull-up resistors. Port 4 pins that have 1's written to them are pulled high by the internal pull-up resistors, and in that state can be used as inputs. As inputs, port 4 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pull-up resistors. Port 4 also erves as alternate compare functions. The output latch corresponding to a secondary functionmust be programmed to a one (1) for that function to operate. The secondary functions are assigned to the pins of port 4 as follows : P4.0 - P4.7 CM0 - CM7 Compare channel 0 - 7
$\overline{\text{PE}}$ / SWD	67	I	**$\overline{\text{Power Saving Modes Enable}}$ / Start Watchdog Timer** A low level on this pin allows the software to enter the power down mode, idle and slow down mode. If the low level is also seen during reset, the watchdog timer function is off on default. Usage of the software controlled power saving modes is blocked, when this pin is held on high level. A high level during reset performs an automatic start of the watchdog timer immediately after reset. When left unconnected this pin is pulled high by a weak internal pullup resistor. During hardware power down the pullup resistor is switched off.

*) I = Input
O = Output

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

Symbol	Pin Number	I/O*)	Function
$\overline{\text{RESET}}$	73	I	**$\overline{\text{RESET}}$** A low level on this pin for the duration of one machine cycle while the oscillator is running resets the C509-L. A small internal pullup resistor permits power-on reset using only a capacitor connected to V_{SS}.
V_{AREF}	78	–	**Reference voltage** for the A/D converter
V_{AGND}	79	–	**Reference ground** for the A/D converter
P7.0 - P7.7	87-80	I	**Port 7** Port 7 is an 8-bit unidirectional input port. Port pins can be used for digital input if voltage levels meet the specified input high/low voltages, and for the lower 8-bit of the multiplexed analog inputs of the A/D converter simultaneously. P7.0 - P7.7 AIN0 - AIN7 Analog input 0 - 7

*) I = Input
O = Output

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

Symbol	Pin Number	I/O*)	Function
P3.0 – P3.7	90-97	I/O	**Port 3** is an 8-bit bidirectional I/O port with internal pullup resistors. Port 3 pins that have 1's written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. As inputs, port 3 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 3 also contains two external interrupt inputs, the timer 0/1 inputs, the serial port 0 receive/transmit line and the external memory strobe pins. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the port pins of port 3 as follows
	90		P3.0 R×D0 Receiver data input (asynchronous) or data input/output (synchronous) of serial interface 0
	91		P3.1 T×D0 Transmitter data output (asynchronous) or clock output (synchronous) of the serial interface 0
	92		P3.2 $\overline{\text{INT0}}$ $\overline{\text{Interrupt 0 input}}$ / timer 0 gate control
	93		P3.3 $\overline{\text{INT1}}$ $\overline{\text{Interrupt 1 input}}$ / timer 1 gate control
	94		P3.4 T0 Counter 0 input
	95		P3.5 T1 Counter 1 input
	96		P3.6 $\overline{\text{WR}}$ The write control signal latches the data byte from port 0 into the external data memory
	97		P3.7 $\overline{\text{RD}}$ / The read control signal enables the external data memory to port 0 $\overline{\text{PSENX}}$ $\overline{\text{PSENX}}$ (external program store enable) enables the external code memory when the external / internal XRAM mode or external / internal programming mode is selected.
V_{SS}	10, 28, 62, 88	–	**Circuit ground potential**
V_{CC}	11, 29, 63, 89	–	**Supply terminal** for all operating modes

*) I = Input
O = Output

2 Fundamental Structure

The C509-L is the high-end 8051-compatible microcontroller of the C500 microcontroller family with a significantly increased performance of CPU and peripheral units. It includes the complete SAB 80C517A functionality, providing 100% upward compatibility. This means that all existing 80517A programs or user's program libraries can be used further on without restriction and may be easily extended to the C509-L.

Some of the various on-chip peripherals of the C509-L support the 8-bit core in case of stringent real-time requirements. The 32-bit/16-bit arithmetic unit, the improved 4-level interrupt structure and eight 16-bit datapointers are meant to give such a CPU support. But strict compatibility to the 8051 architecture is a principle of the C509-L design.

Furthermore, the C509-L contains eight 8-bit I/O ports and fifteen general input lines. The second serial channel is compatible to the 8051-UART and provided with an independent and freely programmable baud rate generator. An 10-bit resolution A/D-converter with built-in self calibration has been integrated to allow analog signal processing. The C509-L further includes a powerful compare/capture unit with two 16-bit compare timers for all kinds of digital signal processing. The controller has been completed with well considered provisions for “fail-safe” reaction in critical applications and offers all CMOS features like low power consumption as well as an idle, power-down and slow-down mode.

Figure 2-1 shows the block diagram of the C509-L.

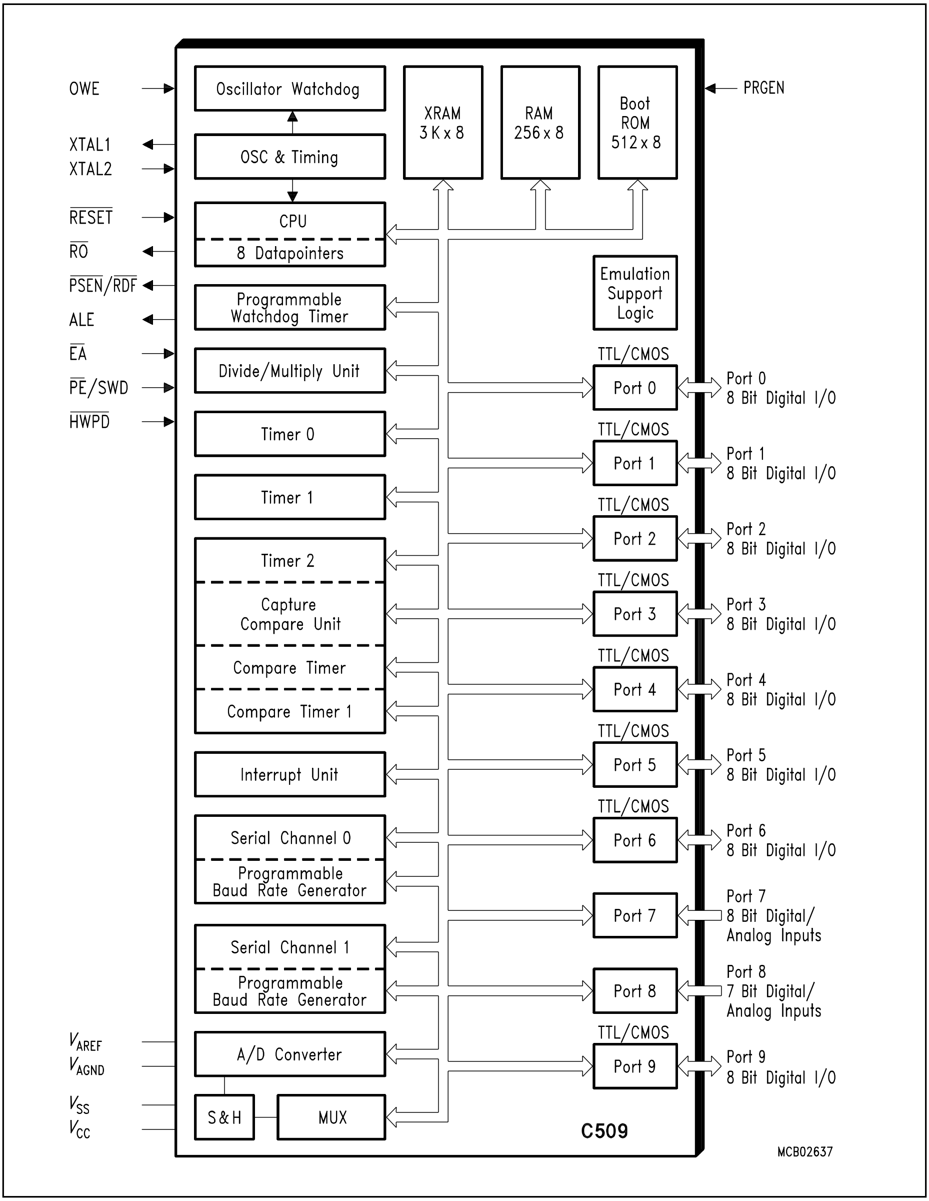


Figure 2-1
Block Diagram of the C509-L

2.1 CPU

The CPU is designed to operate on bits and bytes. The instructions, which consist of up to 3 bytes, are performed in one, two or four machine cycles. One machine cycle requires six oscillator cycles (this number of oscillator cycles differs from other members of the C500 microcontroller family). The instruction set has extensive facilities for data transfer, logic and arithmetic instructions. The Boolean processor has its own full-featured and bit-based instructions within the instruction set. The C509-L uses five addressing modes: direct access, immediate, register, register indirect access, and for accessing the external data or program memory portions a base register plus index-register indirect addressing.Efficient use of program memory results from an instruction set consisting of 44 % one-byte, 41 % two-byte, and 15 % three-byte instructions. With a 16 MHz clock, 58 % of the instructions execute in 375 ns.

The CPU (Central Processing Unit) of the C509-L 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, completement, jump-if-not-set, jump-if-set-and-clear and move to/from carry. Between any addressable bit (or its complement) and the carry flag, it can perform the bit operations of logical AND or logical OR with the result returned to the carry flag.

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

Accumulator

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

Program Status Word

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

Special Function Register PSW (Address D0$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	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

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>CY</td><td>Carry Flag
Used by arithmetic instructions.</td></tr>
<tr><td>AC</td><td>Auxiliary Carry Flag
Used by instructions which execute BCD operations)</td></tr>
<tr><td>F0</td><td>General Purpose Flag</td></tr>
<tr><td>RS1
RS0</td><td>Register Bank select control bits
These bits are used to select one of the four register banks.
<table>
<tr><th>RS1</th><th>RS0</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>Bank 0 selected, data address 00$_H$-07$_H$</td></tr>
<tr><td>0</td><td>1</td><td>Bank 1 selected, data address 08$_H$-0F$_H$</td></tr>
<tr><td>1</td><td>0</td><td>Bank 2 selected, data address 10$_H$-17$_H$</td></tr>
<tr><td>1</td><td>1</td><td>Bank 3 selected, data address 18$_H$-1F$_H$</td></tr>
</table></td></tr>
<tr><td>OV</td><td>Overflow Flag</td></tr>
<tr><td>F1</td><td>General Purpose Flag
Used by arithmetic instructions.</td></tr>
<tr><td>P</td><td>Parity Flag
Set/cleared by hardware after each instruction cycle to indicate an odd/even number of “one” bits in the accumulator, i.e. even parity.</td></tr>
</table>

B Register

The B register is used by 8-bit multiply and divide instructions and serves as both source and destination. For other instructions it can be treated as another scratch pad register.

Stack Pointer

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

2.2 CPU Timing

A machine cycle consists of 6 states (6 oscillator periods). Each state is divided into a phase 1 half, during OSC is high, and a phase 2 half, during which OSC is low. Thus, a machine cycle consists of 6 oscillator periods, numbered S1P1 (state 1, phase 1) through S6P2 (state 6, phase 2). Each state lasts for one oscillator period. Typically, arithmetic and logical operations take place during phase 1 and internal register-to-register transfers take place during phase 2.

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

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

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

Most C509-L 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.

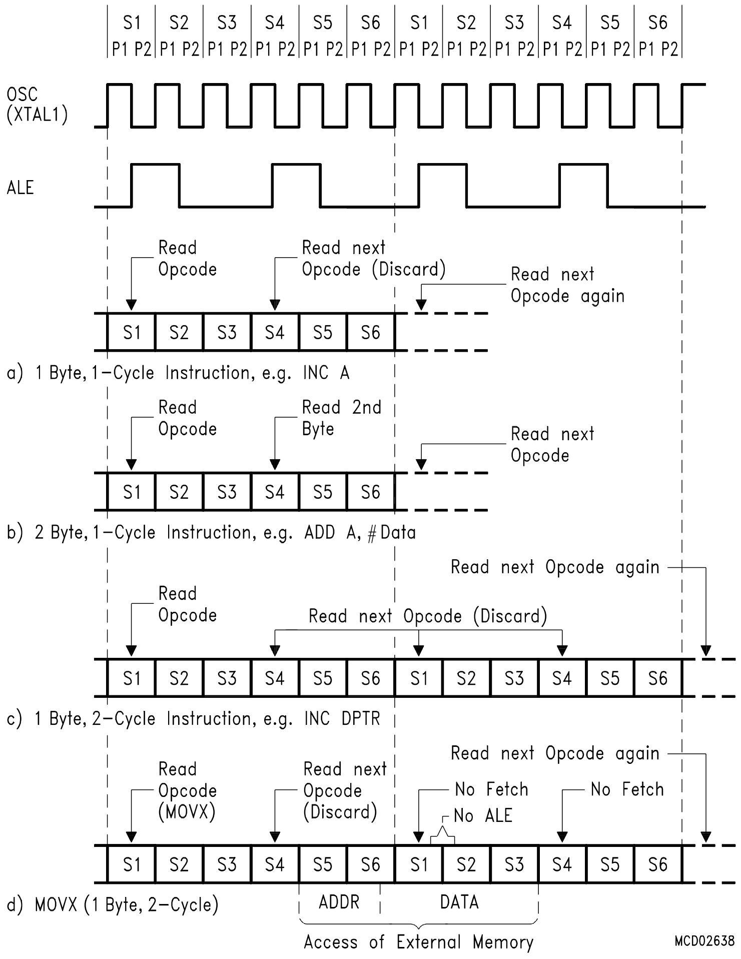

S1 S2 S3 S4 S5 S6 S1 S2 S3 S4 S5 S6
P1 P2
OSC (XTAL1)
ALE
Read Opcode
Read next Opcode (Discard)
Read next Opcode again
a) 1 Byte, 1−Cycle Instruction, e.g. INC A
Read Opcode
Read 2nd Byte
Read next Opcode
b) 2 Byte, 1−Cycle Instruction, e.g. ADD A, #Data
Read next Opcode again
Read Opcode
Read next Opcode (Discard)
c) 1 Byte, 2−Cycle Instruction, e.g. INC DPTR
Read next Opcode again
Read Opcode (MOVX)
Read next Opcode (Discard)
No Fetch
No ALE
No Fetch
d) MOVX (1 Byte, 2−Cycle)
ADDR
DATA
Access of External Memory
MCD02638

Figure 2-2
Fetch and Execute Sequences

3 Memory Organization

The C509-L CPU manipulates data and operands in the following five address spaces:

- up to 64 Kbyte of external program memory
- up to 64 Kbyte of external data memory
- 512 byte of internal Boot ROM (program memory)
- 256 bytes of internal data memory
- 3 Kbyte of external XRAM data memory
- a 128 byte special function register area

Figure 3-1 illustrates the memory address spaces of the C509-L.

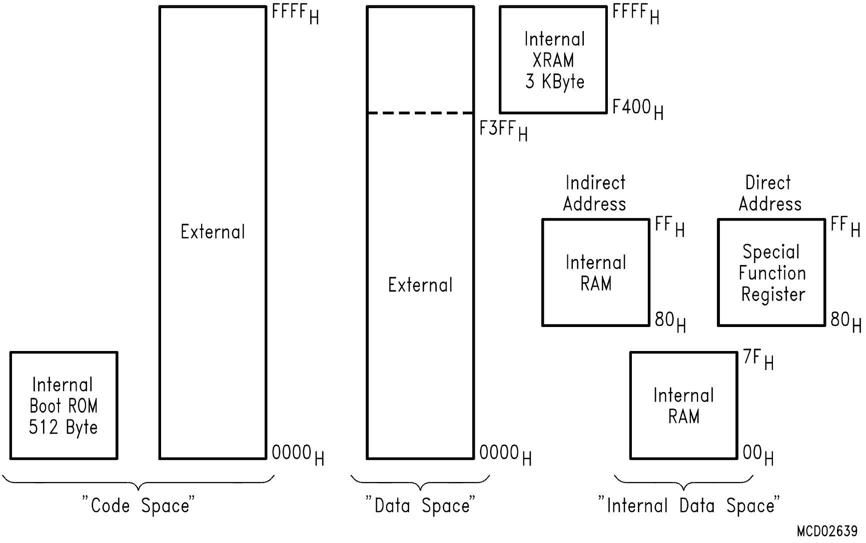


Figure 3-1
C509-L Memory Map

The internal XRAM data memory overlaps with the external data memory in the address range from F400$_H$ to FFFF$_H$. In this address range, either external or internal data RAM can be used. If the internal XRAM has been enabled, it only can be disabled by an active $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ signal.

The internal Boot ROM also overlaps with the external code memory in the address range from 0000$_H$ to 01FF$_H$. Depending on the selected operating mode (chipmode), either external code memory or the internal Boot ROM is accessed in this address range.

3.1 Program Memory, “Code Space”

Besides the internal Boot ROM, the C509-L has no internal program memory (ROM). In normal mode the program memory of the C509-L is located externally and can be expanded up to 64 Kbyte. In the normal mode the C509-L fetches all instructions from the external program memory. Therefore, the pin $\overline{\text{EA}}$ of the C509-L must be always tied to low level.

The Boot ROM includes a bootstrap loader program for the bootstrap mode of the C509-L. The software routines of the bootstrap loader program allow the easy and quick programming or loading of the internal XRAM ($F400_H$ to $FFFF_H$) via the serial interface while the MCU is in-circuit. This allows to transfer custom routines to the XRAM, which will program an external 64 KByte FLASH memory. The routines of the bootstrap loader program may be executed or even can be blocked to prevent unauthorized persons from reading out or writing to the external FLASH memory. Therefore, the bootstrap loader checks an external FLASH memory for existing custom software and executes it. The bootstrap loader program is described in detail in **chapter 10**.

3.2 Data Memory, “Data Space”

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

The external data memory can be expanded up to 64 Kbytes and can be accessed by instructions that use a 16-bit or an 8-bit address. The internal XRAM is also located in the external data memory area and must be accessed by external data memory instructions (MOVX). The XRAM can also serve as code memory in the XRAM mode and in the FLASH programming mode. In these modes program code which has been prior loaded via the bootstrap loader program, is executed in the XRAM.

3.3 General Purpose Registers

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

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

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

3.4 Program and Data Memory Organization

The C509-L can operate in four different operating modes (chipmodes) with different program and data memory organizations:

- Normal Mode
- XRAM Mode
- Bootstrap Mode
- Programming Mode

Table 3-1 describes the program and data memory areas which are available in the different chipmodes of the C509-L. It also shows the control bits of SFR SYSCON1, which are used for the software selection of the chipmodes.

Table 3-1
Overview of Program and Data Memory Organization

Operating Mode (Chipmode)	Program Memory		Data Memory		SYSCON1 Bits	
	Ext.	**Int.**	**Ext.**	**Int.**	**PRGEN 1**	**SWAP**
Normal Mode	0000_H - $FFFF_H$	–	0000_H - $F3FF_H$	$F400_H$ - $FFFF_H$ (XRAM)	0	0
XRAM Mode	0200_H - $F3FF_H$	0000_H - $01FF_H$ = Boot ROM; $F400_H$ - $FFFF_H$ = (XRAM)	0000_H - $FFFF_H$ (read only)	–	0	1
Bootstrap Mode	0200_H - $F3FF_H$	0000_H - $01FF_H$ = Boot ROM	0000_H - $F3FF_H$	$F400_H$ - $FFFF_H$ (XRAM)	1	0
Programming Mode	0200_H - $FFFF_H$	0000_H - $01FF_H$ = Boot ROM; $F400_H$ - $FFFF_H$ = XRAM	0000_H - $FFFF_H$ (read and write)	–	1	1

3.4.1 Interface of External FLASH/ROM/EPROM and External SRAM Memory

The external FLASH/ROM/EPROM memory and the external SRAM memory can be used in the chipmodes either as code memory or as data memory. The basic memory configuration is shown in **figure 3-2**.

The following alternate function pins control the read/write accesses for code/data memories:

$\overline{\text{PSEN}}$ / $\overline{\text{RDF}}$	Read control signal for the FLASH/ROM/EPROM memory
P6.3 / $\overline{\text{WRF}}$	Write control signal for the FLASH/ROM/EPROM memory
P3.6 / $\overline{\text{WR}}$	Write control signal for the SRAM memory
P3.7 / $\overline{\text{RD}}$ / $\overline{\text{PSENX}}$	Read control signal for the SRAM memory

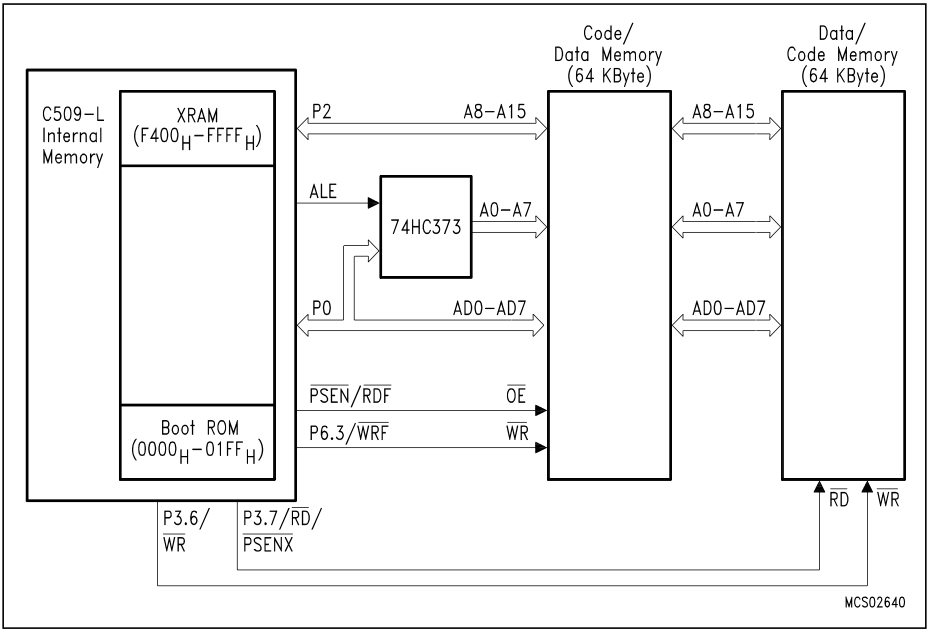


Figure 3-2
Interface of External FLASH/ROM/EPROM and External SRAM Memory

3.4.2 Normal Mode Configuration

The Normal Mode is the standard 8051 compatible operating mode of the C509-L. In this mode 64K byte external code memory and 61K byte external SRAM as well as 3K byte internal data memory (XRAM) are provided. If the XRAM is disabled (default after reset), totally 64K byte external data memory are available. The Boot ROM is disabled. The external program memory is controlled by the $\overline{\text{PSEN}}/\overline{\text{RDF}}$ signal. Read and write accesses to the external data memory are controlled by the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ pins of port 3.

The Normal Mode is entered by keeping the pin PRGEN at a logic low during the rising edge of the external $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ signal. The locations of the code- and data-memory in Normal Mode are shown in **figure 3-4**.

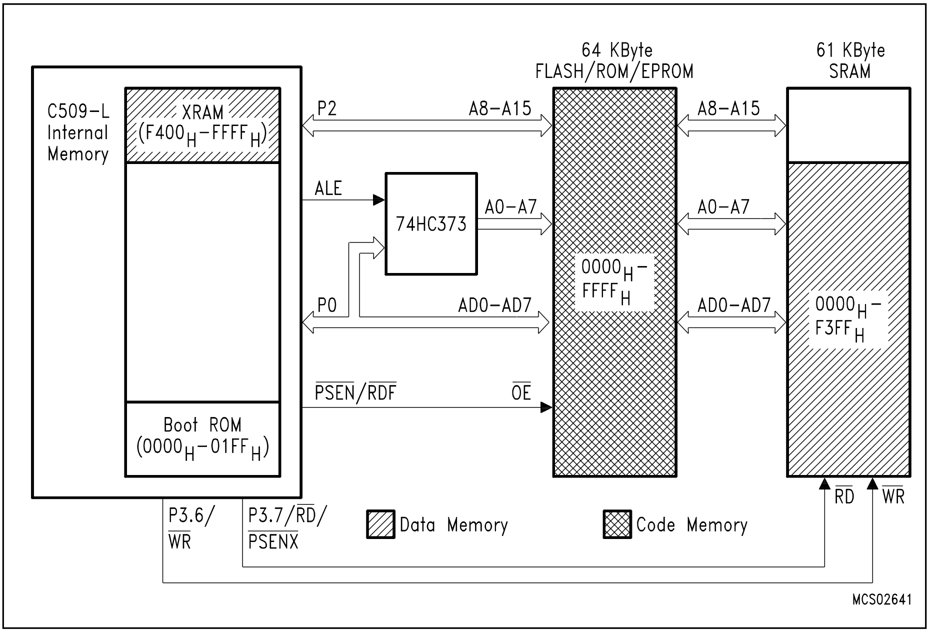


Figure 3-3
Locations of Code- and Data Memory in Normal Mode

3.4.3 XRAM Mode Configuration

The XRAM Mode is implemented in the C509-L for executing e.g. up to 3K byte diagnostic software which has been loaded into the XRAM in the Bootstrap Mode via the serial interface. In this operating mode the Boot ROM, the XRAM, and the external data memory are mapped into the code memory area, while the external ROM/EPROM is mapped into the external data memory area. External program memory fetches from the SRAM are controlled by the P3.7/$\overline{RD}$/$\overline{PSENX}$ pin. External data memory read accesses from the ROM/EPROM are controlled by the $\overline{PSEN}$/$\overline{RDF}$ pin. In XRAM mode, the external data memory can only be read but not written.

The XRAM mode is entered by setting the SWAP bit, while the PRGEN pin (PRGEN1 bit) is kept low. The locations of the code- and data-memory in the XRAM mode are shown in **figure 3-4**.

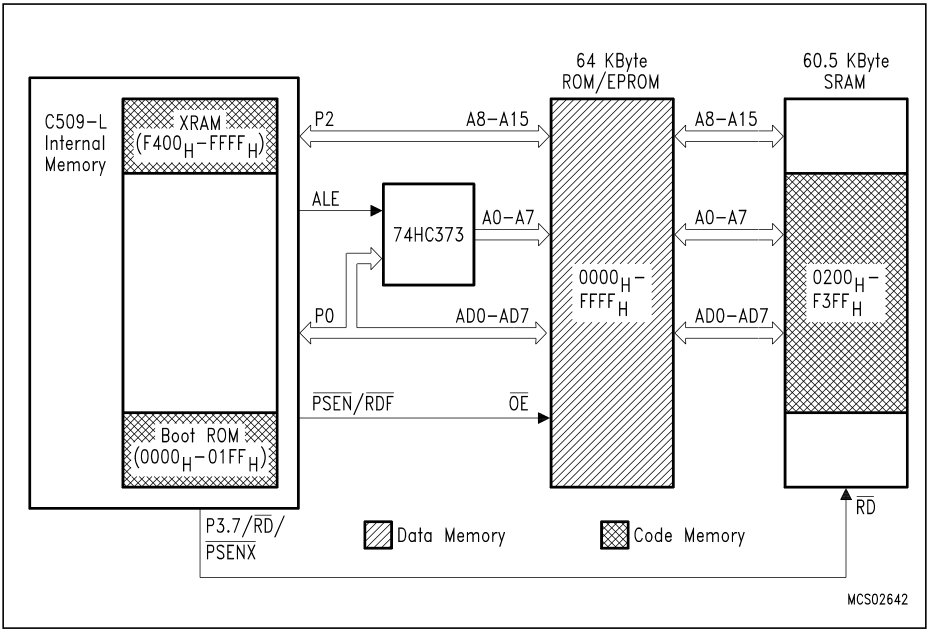


Figure 3-4
Locations of Code- and Data Memory in XRAM Mode

Notes: In the XRAM mode, programming of the external FLASH EPROM is not possible because the PRGEN pin (PRGEN1 bit) is at logic low level (HW-protection).
The internal XRAM is selected automatically in the code memory, if the SWAP bit is set. When leaving the XRAM Mode**,** the XRAM is disabled (only if the XMAP0 bit was not cleared by software before).

3.4.4 Bootstrap Mode Configuration

In the Bootstrap Mode the Boot ROM and the external FLASH/ROM/EPROM are mapped into the code memory area. 61K byte external SRAM as well as 3K byte internal data memory (XRAM) are provided in the external data memory area. The external program memory is controlled by the $\overline{\text{PSEN}}/\overline{\text{RDF}}$ signal. Read and write accesses to the external data memory are controlled by the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ pins of port 3.

The Bootstrap Mode is entered by keeping the pin PRGEN at a logic high level during the rising edge of the external $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ signal (→ PRGEN1=1). The locations of the code- and data memory in the external boot-strap mode are shown in **figure 3-5**.

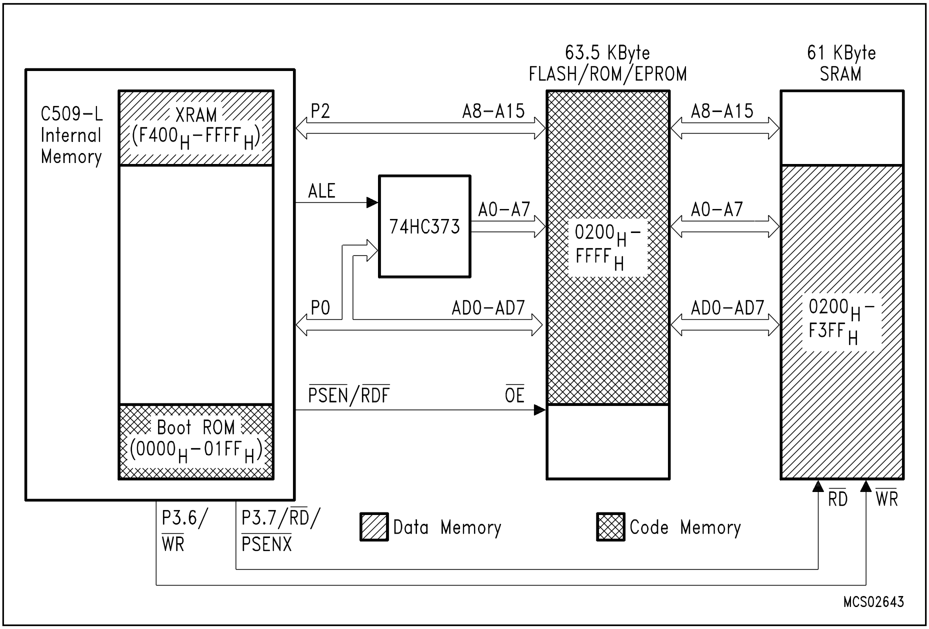


Figure 3-5
Locations of Code- and Data Memory in Bootstrap Mode

In Bootstrap Mode the internal XRAM is selected automatically as data memory. When leaving the Bootstrap Mode**,** the XRAM is disabled (only if the XMAP0 bit was not cleared by software before).

3.4.5 Programming Mode Configuration

The External Programming Mode is implemented for the in-circuit programming of external 5V-only FLASH EPROMs. Similar as in the XRAM mode, the Boot ROM, the XRAM, and the external data memory (SRAM) are mapped into the code memory area, while the external FLASH memory is mapped into the external data memory area. Additionally to the XRAM mode, the FLASH memory can also be written through external data memory accesses (MOVX instructions). External program memory fetches from the SRAM are controlled by the P3.7/$\overline{\text{RD}}$/$\overline{\text{PSENX}}$ pin. External data memory read/write accesses from/to the ROM/EPROM are controlled by the $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$ and P6.3/$\overline{\text{WRF}}$ pin.

The Programming Mode is entered by setting the bits PRGEN1 and SWAP in SFR SYSCON1. The locations of the code- and data memory in the Programming Mode are shown in **figure 3-6**.

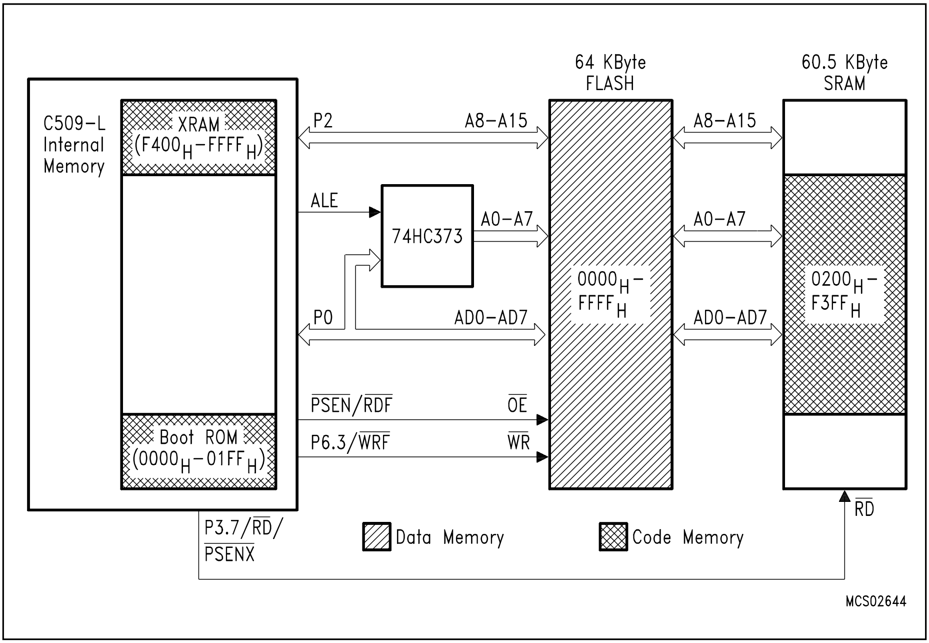


Figure 3-6
Locations of Code- and Data Memory in Programming Mode

Prior to the usage of the Programming Mode, the XRAM has to be loaded by the FLASH specific programming software algorithms (see also **chapter 10** “The Bootstrap Loader”). This XRAM load operation can be done using the Bootstrap Mode.

Notes: The internal XRAM is enabled automatically in the code memory area if the SWAP bit is set. When leaving the Programming Mode, the XRAM is disabled (only if the XMAP0 bit was not cleared by software before).

Leaving the Programming Mode can be accomplished by:

– Clearing the bits PRGEN1 and SWAP prior to executing the special software unlock sequence followed by a LJMP to “startaddress” in the FLASH memory returns the C509-L into normal mode (program execution will start at “startaddress”).
– Activating the $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ signal with the PRGEN pin at logic low level will clear the bits PRGEN1 and SWAP.

Notes: Switching the PRGEN pin to logic low level during programming of an external FLASH will reset the PRGEN1 bit. As a result the XRAM mode is entered and this inhibits any write access to the external FLASH EPROM.

Clearing the SWAP bit during Programming Mode and executing the special software unlock sequence will force the C509-L into the Bootstrap Mode.

3.4.6 Special Function Register SYSCON1

There are five control bits located in SFR SYSCON1 ($B2_H$) which control the code and data memory organization of the C509-L. Two of these bits (PRGEN1 and SWAP) cannot be programmed as normal bits but with a special software unlock sequence. The special software unlock sequence was implemented to prevent unintentional changing of these bits and consists of consecutive followed instructions which have to set two dedicated enable bits.

Special Function Register SYSCON1 (Address $B2_H$) **Reset Value : $00XXXEE0_B$ [1]**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$B2_H$	ESWC	SWC	–	EA1	EA0	PRGEN1	PRGEN0	SWAP	SYSCON1

[1] "E" means that the value of the bit is defined through the external logic level at pin PRGEN at the rising edge of the external $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ signals.

Bit	Function
ESWC	Enable Switch Chipmode When selecting the chipmode with the bits SWAP and PRGEN1, the ESWC bit has to be set simultaneously as the first instruction in the special software unlock sequence. The bit ESWC will be cleared by hardware after 2 instruction cycles.
SWC	Switch Chip Mode The SWC bit has to be set as the second instruction in a special software unlock sequence directly after having set bit ESWC. The new chipmode becomes active after the second instruction cycle which follows the special software unlock sequence. These two instruction cycles are used for initializing of the program counter (LJMP instruction) SWC is a write only bit. Reading SWC will always return a '0'.
–, EA1, EA0	Reserved bits; at read operations these bits are undefined; at write operations to SYSCON1 these bits can be written with "0" or "1".
PRGEN1	Program Enable Bit 1 The PRGEN1 bit enables/disables the write accesses to an external FLASH EPROM. The PRGEN1 bit contains the logic value of the external PRGEN pin which is latched at the rising edge of the external $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ signal. When the logic low level appears at the PRGEN pin, the PRGEN1 bit will be cleared in the next instruction cycle without the need of a special software unlock sequence. PRGEN1 = 0: Write access (programming) of external FLASH EPROM is disabled PRGEN1 = 1: Write access (programming) of external FLASH EPROM is enabled Any changing the PRGEN1 bit without using a special software unlock sequence with the ESWC and SWC bits will have no effect and the former selected status will be kept.

Bit	Function
PRGEN0	Program Enable Bit 0 The PRGEN0 bit is a read-only bit and represents the logic level of the external PRGEN pin. PRGEN0 = 1: The PRGEN1 bit can be set or cleared under software control. PRGEN0 = 0: The PRGEN1 bit is held at logic low level and cannot be set under software control. Notes: Clearing the PRGEN0 bit by changing the logic level at the PRGEN pin in the Normal Mode disables write accesses to the external FLASH EPROM. Enable/disable write accesses to external FLASH EPROM can be done by changing the PRGEN1 bit with a special software unlock sequence.
SWAP	Swap Code- and Data Memory SWAP = 0: The data memory and the code memory remain in their predefined locations. SWAP = 1: The following address areas and memory locations are assigned to code memory: 0000_H - $01FF_H$ → Boot ROM 0200_H - $F3FF_H$ → External data memory is swapped to external code memory $F400_H$ - $FFFF_H$ → The XRAM is enabled and automatically mapped into code memory space. (independent of bit XMAP0 in SFR SYSCON) The following address areas and memory locations ar assigned to data memory: 0000_H - $FFFF_H$ → External FLASH /ROM/EPROM The former code memory is assigned as data memory and is now addressable by using MOVX instructions.

3.4.7 Operating Mode (Chipmode) Selection

The chipmode selection can be done by hardware after an active $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ signal. Further, the logic state of pin PRGEN is used for hardware selection. The chipmodes can also be selected by software. Software selection is achieved by programming specific bits of SFR SYSCON1.

The following **table 3-1** shows the hardware and software selection capabilities of the chipmodes.

Table 3-2
Hardware and Software Selection of Chipmodes

Operating Mode (Chipmode)	Hardware Selection	Software Selection
Normal Mode	PRGEN pin = low ① $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$: ↑ edge	PRGEN pin = low; ④ setting bits PRGEN1,SWAP = 0,0; executing unlock sequence
XRAM Mode	from Programming Mode: ② setting PRGEN pin = low	PRGEN pin = low; ⑤ setting bits PRGEN1,SWAP = 0,1; executing unlock sequence
Bootstrap Mode	PRGEN pin = high ③ $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$: ↑ edge	PRGEN pin = high; ⑥ setting bits PRGEN1,SWAP =1,0; executing unlock sequence
Programming Mode	not possible	PRGEN pin = high; ⑦ setting bits PRGEN1,SWAP = 1,1; executing unlock sequence

Figure 3-7 below shows the information of **table 3-1** as a state diagram (reference number in the circle).

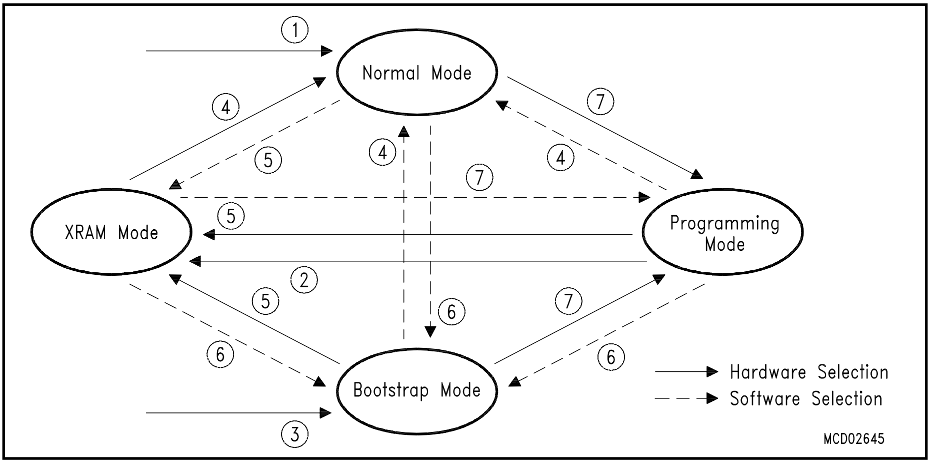


Figure 3-7
State Diagram of Chipmode Selection

3.4.7.1 Special Software Unlock Sequence

The bits ESWC and SWC in SFR SYSCON1 are implemented in a way to prevent unintentional changing of the bits SWAP or PRGEN1. Any changing the bits SWAP or PRGEN1 without using the ESWC and SWC bits in a special software unlock sequence will have no effect and the above bits will get back their old values two instruction cycles after being changed.

The following programming steps must be executed at the ESWC/SWC unlock sequence:

1.) **First instruction:**
Changing the value of the bits SWAP or PRGEN1 with one or more consecutive instructions simultaneously with setting of bit ESWC:

```
ANL  SYSCON1, #11111X1YB    ; clearing of bits PRGEN1 (X=0) and/or SWAP (Y=0)
ORL  SYSCON1, #10000X0YB    ; setting of ESWC bit with setting of PRGEN1 or SWAP
                            ; e.g. clearing of the SWAP bit:
                                                   ANL  SYSCON1,#11111110B
                                                   ORL  SYSCON1,#10000000B
```

<u>or:</u>

```
ORL  SYSCON1, #10000X0YB    ; setting of the bits PRGEN1 (X=1) and/or SWAP (Y=1) and
                            ; setting the ESWC bit simultaneously
                            ; e.g. setting of the SWAP bit:
                                                   ORL  SYSCON1,#10000001B
```

2.) **Second instruction:**
Setting of bit SWC immediately after 1.) with

```
ORL  SYSCON1, #40H          ;
```

The new chipmode becomes active two instruction cycles <u>after</u> the instruction which sets the bit SWC (see 2.). These two instruction cycle delay should normally be used for initialization of the program counter to the 16 bit start-address of the new code memory resource, e.g. with:

```
LJMP  0XXXXH          ; XXXX = 16-bit hex address in new code memory
```

If the code memory resource is not changed by the new chipmode there is no need of a new initialization of the program counter. However the new Chipmode becomes active two instruction cycles after 2.).

The special software unlock instruction sequence cannot be interrupted by an interrupt request. Any write or read operation to SFR SYSCON1 will block the interrupt generation for the first cycle of the directly following instruction.Therefore, the response time of an interrupt request may be additionally delayed from minimal five instruction cycles up to eight instruction cycles: four or six instruction cycles for setting ESWC and SWC (depends on the used instructions) and one or two instruction cycles depending on the instruction used in 3.). When using a one cycle instruction in 3.) an enabled interrupt may be performed and the interrupt vector address may reside in a “new” code memory resource due to the new selected Chipmode.

The bits SWAP and PRGEN1 are built up by three shadow latches each. Unintentional changing of one of this three latches (e.g. by RFI) will have no effect and the former value will be restored in the next instruction cycle.

3.4.7.2 Control Signals of the Chipmodes

As shown in **chapter 3.4.2** to **3.4.5**, each chipmode uses specific read/write control signals, **Table 3-3** gives a detailed survey about each chipmode and its control signals.

Table 3-3
Usage of Control Signals in the different Chipmodes

Operating Mode (Chipmode)	**Code Memory Control**		**Data Memory Control**			
	$\overline{\text{PSEN}}$ / $\overline{\text{RDF}}$	**P3.7 / $\overline{\text{RD}}$ / $\overline{\text{PSENX}}$**	**P3.7 / $\overline{\text{RD}}$ / $\overline{\text{PSENX}}$**	**P3.6 / $\overline{\text{WR}}$**	**$\overline{\text{PSEN}}$ / $\overline{\text{RDF}}$**	**P6.3 / $\overline{\text{WRF}}$**
Normal Mode	0000$_H$ - FFFF$_H$	–	XMAP0=1 : 0000$_H$ - FFFF$_H$ XMAP0=0 : 0000$_H$ - F3FF$_H$	XMAP0=1 : 0000$_H$ - FFFF$_H$ XMAP0=0 : 0000$_H$ - F3FF$_H$	–	–
XRAM Mode	–	0200$_H$ - F3FF$_H$	–	–	0000$_H$ - FFFF$_H$	–
Bootstrap Mode	0200$_H$ - FFFF$_H$	–	0000$_H$ - F3FF$_H$	0000$_H$ - F3FF$_H$	–	–
Programming Mode	–	0200$_H$ - F3FF$_H$	–	–	0000$_H$ - FFFF$_H$	0000$_H$ - FFFF$_H$

3.4.7.3 Switching of the SWAP-Bit

Setting or clearing the SWAP-bit in SYSCON1 will change code memory and data memory assignment. To assure a well defined behavior of the controller, it is necessary to perform a specific code sequence by the user. Any write access to SYSCON1 which changes the SWAP bit has to be followed by a 2-cycle instruction. This 2-cycle instruction will be the last instruction, which is performed by the CPU in the former code memory. For that reason, this 2-cycle instruction has to be a LJMP-instruction which specifies the new address for the program counter. The next instruction, which will be performed by the CPU is at the specified destination address of the JMP instruction in the former data memory (which now has become to code memory).

Figure 3-8 gives an overview about the behavior of the external pin $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$ and $\overline{\text{RD}}$/$\overline{\text{PSENX}}$ when chipmodes with different SWAP bits are switched. These signals change their functions depending on the value of the SWAP bit.

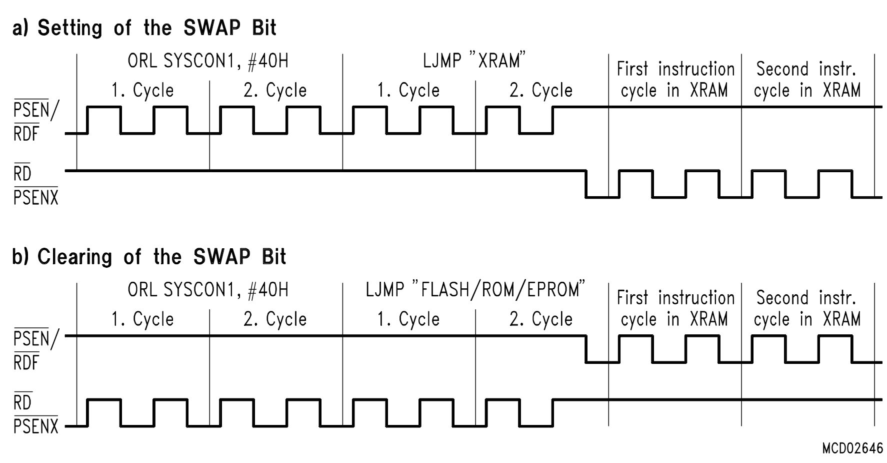


Figure 3-8
Switching of the SWAP Bit

Figure 3-8 a): When the SWAP bit is set, the “program store enable” function of the $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$ pin, which is connected to $\overline{\text{OE}}$ (output enable) of the FLASH/ROM/EPROM memory, is switched to the $\overline{\text{RD}}$/$\overline{\text{PSENX}}$ pin. The “read enable” function of the $\overline{\text{RD}}$/$\overline{\text{PSENX}}$ pin, which is connected to the $\overline{\text{RD}}$ (“read“) input of an external SRAM memory, is switched to the $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$ signal. $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$ becomes now active during MOVX instructions.

When the SWAP bit is cleared, the “read enable” function (MOVX instructions) of the $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$ pin, which is connected to $\overline{\text{OE}}$ (output enable) of the FLASH/ROM/EPROM memory, is switched to the $\overline{\text{RD}}$/$\overline{\text{PSENX}}$ pin. The “program store enable” function of the $\overline{\text{RD}}$/$\overline{\text{PSENX}}$ pin, which is connected to the $\overline{\text{RD}}$ (“read“) input of the external SRAM memory, is switched to the $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$ pin. $\overline{\text{RD}}$/$\overline{\text{PSENX}}$ becomes now active during MOVX instructions.

3.4.8 Watchdog Timer - Behaviour in the Different Operating Modes

This section describes the chipmode specific behavior of the watchdog timer unit. Further details about the watchdog timer are given in **chapter 8**.

Normal Mode:
The watchdog timer function in Normal Mode is not restricted.

XRAM Mode:
To avoid deadlocks during program execution in the XRAM, the once enabled watchdog timer keeps on running but the refresh is inhibited. A refresh sequence of a previously enabled watchdog timer is not permitted. An unintentional refresh sequence in the XRAM mode does not reload the watchdog timer and an internally generated watchdog reset will be executed at the watchdog counter state $7FFC_H$. The watchdog timer reset generation is enabled.

For refreshing the watchdog timer, the user has to leave the XRAM Mode and must enter the Normal Mode by clearing the SWAP bit followed by a LJMP or LCALL to “startaddress of watchdog timer refresh” in the external ROM/EPROM. In the normal mode the watchdog timer refresh can be executed successfully. If required, XRAM Mode can again be entered after the watchdog timer refresh operation has been finished.

Bootstrap Mode and Programming Mode
In the Bootstrap Mode and in the Programming Mode the watchdog timer increment is inhibited. In this way, programming the external FLASH EPROM in Programming Mode cannot be aborted by the watchdog timer.

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 two mapped special function register areas. Several special function registers of the C509-L (CC10-17, CT1REL, CC1EN, CAFR) are 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.

For accessing the port direction registers, which define the input or output function of the bidirectional port structure, bit PDIR in SFR IP1 is used. This port direction register access operates similar to the mapped SFR accesses, but a double instruction sequence must be executed. The first instruction has to set bit PDIR. Thereafter, the second instruction can read or write the direction register. Further details about port direction selection see **chapter 6.1.1.1**.

The most right column of **table 3-5** indicates if an SFR is accessed with a mapped procedure controlled by either RMAP or PDIR.

Special Function Register SYSCON (Address B1$_H$) **Reset Value : 1010XX01$_B$**
Special Function Register IP1 (Address B9$_H$) **Reset Value : 0X000000$_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
B1$_H$	CLKP	PMOD	1	RMAP	–	–	XMAP1	XMAP0	SYSCON
B9$_H$	PDIR	–	.5	.4	.3	.2	.1	.0	IP1

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

Bit	Function
RMAP	Special function register map bit RMAP = 0: The access to the non-mapped (standard) special function register area is enabled (reset value). RMAP = 1: The access to the mapped special function register area is enabled.
PDIR	Direction register enable PDIR = 0: Port register access is enabled (reset value). PDIR = 1: Direction register is enabled. PDIR will automatically be cleared after the second machine cycle (S2P2) after having been set.

As long as bit RMAP is set, mapped special function registers 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 by software, respectively each.

There are also 128 directly addressable bits available within each SFR area (standard and mapped SFR area). All SFRs with addresses where address bits 0-2 are 0 (e.g. 80_H, 88_H, 90_H, 98_H, …, $F8_H$, FF_H) are bitaddressable.

The 103 special function register (SFR) include pointers and registers that provide an interface between the CPU and the other on-chip peripherals. The SFRs of the C509-L are listed in **table 3-4** and **table 3-5**. In **table 3-4** they are organized in groups which refer to the functional blocks of the C509-L. **Table 3-5** illustrates the contents of the SFRs in numeric order of their addresses.

Table 3-4
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	**D0$_H$** [1]	**00$_H$**
	SP	Stack Pointer	81$_H$	07$_H$
	SYSCON1	System Control Register 1	B2$_H$	00XXXEE0$_B$ [3][6]
SFR Mapping	SYSCON [2]	System Control Register	B1$_H$	1010XX01$_B$ [3]
Interrupt System	IEN0	Interrupt Enable Register 0	**A8$_H$** [1]	**00$_H$**
	CTCON [2]	Compare Timer Control Register	E1$_H$	01000000$_B$ [3]
	CT1CON [2]	Compare Timer 1 Control Register	BC$_H$	X1XX0000$_B$ [3]
	IEN1 [2]	Interrupt Enable Register 1	**B8$_H$** [1]	**00$_H$**
	IEN2 [2]	Interrupt Enable Register 2	9A$_H$	XX0000X0$_B$ [3]
	IEN3	Interrupt Enable Register 3	BE$_H$	XXXX00XX$_B$ [3]
	IP0 [2]	Interrupt Priority Register 0	A9$_H$	00$_H$
	IP1 [2]	Interrupt Priority Register 1	B9$_H$	0X000000$_B$ [3]
	IRCON0	Interrupt Request Control Register 0	**C0$_H$** [1]	**00$_H$**
	IRCON1	Interrupt Request Control Register 1	D1$_H$	00$_H$
	IRCON2 [4]	Interrupt Request Control Register 2	BF$_H$	00$_H$
	EICC1 [4]	Interrupt Request Enable Register for CT1	BF$_H$	FF$_H$
	TCON [2]	Timer Control Register	**88$_H$** [1]	**00$_H$**
	T2CON [2]	Timer 2 Control Register	**C8$_H$** [1]	**00$_H$**
XRAM	XPAGE	Page Address Register for XRAM	91$_H$	00$_H$
	SYSCON [2]	System Control Register	B1$_H$	1010XX01$_B$ [3]
A/D Converter	ADCON0	A/D Converter Control Register 0	**D8$_H$** [1]	**00$_H$**
	ADCON1	A/D Converter Control Register 1	DC$_H$	01000000$_B$ [3]
	ADDATH	A/D Converter Data Register, High Byte	D9$_H$	00$_H$
	ADDATL	A/D Converter Data Register, Low Byte	DA$_H$	00$_H$

[1] Bit-addressable special function registers
[2] This special function register is listed repeatedly since some bits of it also belong to other functional blocks.
[3] X means that the value is indeterminate or the location is reserved
[4] Register is mapped by bit PDIR.
[5] Register is mapped by bit RMAP.
[6] “E” means that the value of the bit is defined by the logic level at pin PRGEN at the rising edge of the $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ signals.

Table 3-4
Special Function Registers - Functional Blocks (cont'd)

Block	Symbol	Name	Address	Contents after Reset
Compare / Capture Unit (CCU) Timer 2	CCEN	Compare/Capture Enable Register	C1$_H$	00$_H$
	CC4EN	Compare/Capture 4 Enable Register	C9$_H$	00$_H$
	CCH1	Compare/Capture Register 1, High Byte	C3$_H$	00$_H$
	CCH2	Compare/Capture Register 2, High Byte	C5$_H$	00$_H$
	CCH3	Compare/Capture Register 3, High Byte	C7$_H$	00$_H$
	CCH4	Compare/Capture Register 4, High Byte	CF$_H$	00$_H$
	CCL1	Compare/Capture Register 1, Low Byte	C2$_H$	00$_H$
	CCL2	Compare/Capture Register 2, Low Byte	C4$_H$	00$_H$
	CCL3	Compare/Capture Register 3, Low Byte	C6$_H$	00$_H$
	CCL4	Compare/Capture Register 4, Low Byte	CE$_H$	00$_H$
	CMEN [5]	Compare Enable Register	F6$_H$	00$_H$
	CMH0 [5]	Compare Register 0, High Byte	D3$_H$	00$_H$
	CMH1 [5]	Compare Register 1, High Byte	D5$_H$	00$_H$
	CMH2 [5]	Compare Register 2, High Byte	D7$_H$	00$_H$
	CMH3 [5]	Compare Register 3, High Byte	E3$_H$	00$_H$
	CMH4 [5]	Compare Register 4, High Byte	E5$_H$	00$_H$
	CMH5 [5]	Compare Register 5, High Byte	E7$_H$	00$_H$
	CMH6 [5]	Compare Register 6, High Byte	F3$_H$	00$_H$
	CMH7 [5]	Compare Register 7, High Byte	F5$_H$	00$_H$
	CML0 [5]	Compare Register 0, Low Byte	D2$_H$	00$_H$
	CML1 [5]	Compare Register 1, Low Byte	D4$_H$	00$_H$
	CML2 [5]	Compare Register 2, Low Byte	D6$_H$	00$_H$
	CML3 [5]	Compare Register 3, Low Byte	E2$_H$	00$_H$
	CML4 [5]	Compare Register 4, Low Byte	E4$_H$	00$_H$
	CML5 [5]	Compare Register 5, Low Byte	E6$_H$	00$_H$
	CML6 [5]	Compare Register 6, Low Byte	F2$_H$	00$_H$
	CML7 [5]	Compare Register 7, Low Byte	F4$_H$	00$_H$
	CC1EN [5]	Compare/Capture Enable Register	F6$_H$	00$_H$
	CC1H0 [5]	Compare/Capture 1 Register 0, High Byte	D3$_H$	00$_H$
	CC1H1 [5]	Compare/Capture 1 Register 1, High Byte	D5$_H$	00$_H$
	CC1H2 [5]	Compare/Capture 1 Register 2, High Byte	D7$_H$	00$_H$
	CC1H3 [5]	Compare/Capture 1 Register 3, High Byte	E3$_H$	00$_H$
	CC1H4 [5]	Compare/Capture 1 Register 4, High Byte	E5$_H$	00$_H$
	CC1H5 [5]	Compare/Capture 1 Register 5, High Byte	E7$_H$	00$_H$
	CC1H6 [5]	Compare/Capture 1 Register 6, High Byte	F3$_H$	00$_H$
	CC1H7 [5]	Compare/Capture 1 Register 7, High Byte	F5$_H$	00$_H$
	CC1L0 [5]	Compare/Capture 1 Register 0, Low Byte	D2$_H$	00$_H$
	CC1L1 [5]	Compare/Capture 1 Register 1, Low Byte	D4$_H$	00$_H$
	CC1L2 [5]	Compare/Capture 1 Register 2, Low Byte	D6$_H$	00$_H$
	CC1L3 [5]	Compare/Capture 1 Register 3, Low Byte	E2$_H$	00$_H$
	CC1L4 [5]	Compare/Capture 1 Register 4, Low Byte	E4$_H$	00$_H$
	CC1L5 [5]	Compare/Capture 1 Register 5, Low Byte	E6$_H$	00$_H$
	CC1L6 [5]	Compare/Capture 1 Register 6, Low Byte	F2$_H$	00$_H$
	CC1L7 [5]	Compare/Capture 1 Register 7, Low Byte	F4$_H$	00$_H$
	CMSEL[5]	Compare Input Select	F7$_H$	00$_H$

[5] Register is mapped by bit RMAP.

Table 3-4
Special Function Registers - Functional Blocks (cont'd)

Block	Symbol	Name	Address	Contents after Reset
Compare / Capture Unit (CCU) Timer 2 cont'd	CAFR [5]	Capture 1, Falling/Rising Edge Register	F7$_H$	00$_H$
	CRCH	Comp./Rel./Capt. Reg. High Byte	CB$_H$	00$_H$
	CRCL	Comp./Rel./Capt. Reg. Low Byte	CA$_H$	00$_H$
	COMSETL	Compare Set Register, Low Byte	A1$_H$	00$_H$
	COMSETH	Compare Set Register, High Byte	A2$_H$	00$_H$
	COMCLRL	Compare Clear Register, Low Byte	A3$_H$	00$_H$
	COMCLRH	Compare Clear Register, High Byte	A4$_H$	00$_H$
	SETMSK	Compare Set Mask Register	A5$_H$	00$_H$
	CLRMSK	Compare Clear Mask Register	A6$_H$	00$_H$
	CTCON [2]	Compare Timer Control Register	E1$_H$	01000000$_B$ [3]
	CTRELH [5]	Compare Timer Rel. Reg., High Byte	DF$_H$	00$_H$
	CTRELL [5]	Compare Timer Rel. Reg., Low Byte	DE$_H$	00$_H$
	CT1RELH [5]	Compare Timer 1 Rel. Reg., High Byte	DF$_H$	00$_H$
	CT1RELL [5]	Compare Timer 1 Rel. Reg., Low Byte	DE$_H$	00$_H$
	TH2	Timer 2, High Byte	CD$_H$	00$_H$
	TL2	Timer 2, Low Byte	CC$_H$	00$_H$
	T2CON [2]	Timer 2 Control Register	**C8$_H$** [1]	**00$_H$**
	CT1CON [2]	Compare Timer 1 Control Register	BC$_H$	X1XX0000$_B$ [3]
	PRSC [2]	Prescaler Control Register	B4$_H$	11010101$_B$ [3]
Serial Channels	ADCON0 [2]	A/D Converter Control Register	**D8$_H$** [1]	**00$_H$**
	PCON [2]	Power Control Register	87$_H$	00$_H$
	S0BUF	Serial Channel 0 Buffer Register	99$_H$	XX$_H$ [3]
	S0CON	Serial Channel 0 Control Register	**98$_H$** [1]	**00$_H$**
	S0RELL	Serial Channel 0 Reload Reg., Low Byte	AA$_H$	D9$_H$
	S0RELH	Serial Channel 0 Reload Reg., High Byte	BA$_H$	XXXXXX11$_B$ [3]
	S1BUF	Serial Channel 1 Buffer Register	9C$_H$	XX$_H$ [3]
	S1CON	Serial Channel 1 Control Register	9B$_H$	01000000$_B$ [3]
	S1RELL	Serial Channel 1 Reload Reg., Low Byte	9D$_H$	00$_H$
	S1RELH	Serial Channel 1 Reload Reg., High Byte	BB$_H$	XXXXXX11$_B$ [3]
Watchdog	IEN0 [2]	Interrupt Enable Register 0	**A8$_H$** [1]	**00$_H$**
	IEN1 [2]	Interrupt Enable Register 1	**B8$_H$** [1]	**00$_H$**
	IP0 [2]	Interrupt Priority Register 0	A9$_H$	00$_H$
	IP1 [2]	Interrupt Priority Register 1	B9$_H$	0X000000$_B$ [3]
	WDTREL	Watchdog Timer Reload Register	86$_H$	00$_H$
	WDTL [6]	Watchdog Timer Register, Low Byte	84$_H$	00$_H$
	WDTH [6]	Watchdog Timer Register, High Byte	85$_H$	00$_H$

[1] Bit-addressable special function registers
[2] This special function register is listed repeatedly since some bits of it also belong to other functional blocks.
[3] X means that the value is indeterminate or the location is reserved
[4] Register is mapped by bit PDIR.
[5] Register is mapped by bit RMAP.
[6] Registers are only readable and cannot be written.

Table 3-4
Special Function Registers - Functional Blocks (cont'd)

Block	Symbol	Name	Address	Contents after Reset
MUL/DIV Unit	ARCON	Arithmetic Control Register	EF$_H$	0XXXXXXX$_B$ [3]
	MD0	Multiplication/Division Register 0	E9$_H$	XX$_H$ [3]
	MD1	Multiplication/Division Register 1	EA$_H$	XX$_H$ [3]
	MD2	Multiplication/Division Register 2	EB$_H$	XX$_H$ [3]
	MD3	Multiplication/Division Register 3	EC$_H$	XX$_H$ [3]
	MD4	Multiplication/Division Register 4	ED$_H$	XX$_H$ [3]
	MD5	Multiplication/Division Register 5	EE$_H$	XX$_H$ [3]
Timer 0 / Timer 1	TCON	Timer 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$_H$
	PRSC [2]	Prescaler Control Register	B4$_H$	11010101$_B$ [3]
Ports	P0 [4]	Port 0	**80$_H$** [1]	**FF$_H$**
	DIR0 [4]	Direction Register Port 0	**80$_H$** [1]	**FF$_H$**
	P1 [4]	Port 1	**90$_H$** [1]	**FF$_H$**
	DIR1 [4]	Direction Register Port 1	**90$_H$** [1]	**FF$_H$**
	P2 [4]	Port 2	**A0$_H$** [1]	**FF$_H$**
	DIR2 [4]	Direction Register Port 2	**A0$_H$** [1]	**FF$_H$**
	P3 [4]	Port 3	**B0$_H$** [1]	**FF$_H$**
	DIR3 [4]	Direction Register Port 3	**B0$_H$** [1]	**FF$_H$**
	P4 [4]	Port 4	**E8$_H$** [1]	**FF$_H$**
	DIR4 [4]	Direction Register Port 4	**E8$_H$** [1]	**FF$_H$**
	P5 [4]	Port 5	**F8$_H$** [1]	**FF$_H$**
	DIR5 [4]	Direction Register Port 5	**F8$_H$** [1]	**FF$_H$**
	P6 [4]	Port 6	FA$_H$	FF$_H$
	DIR6 [4]	Direction Register Port 6	FA$_H$	FF$_H$
	P7	Port 7, Analog/Digital Input	DB$_H$	--
	P8	Port 8, Analog/Digital Input	DD$_H$	--
	P9 [4]	Port 9	F9$_H$	FF$_H$
	DIR9 [4]	Direction Register Port 9	F9$_H$	FF$_H$
Power Saving Modes	PCON	Power Control Register	87$_H$	00$_H$

[1] Bit-addressable special function registers
[2] This special function register is listed repeatedly since some bits of it also belong to other functional blocks.
[3] X means that the value is indeterminate and the location is reserved
[4] Register is mapped by bit PDIR.
[5] Register is mapped by bit RMAP.

Table 3-5
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	Mapped by [2]
80_H	P0	FF_H	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=0
80_H	DIR0	FF_H	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=1
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	–
84_H	WDTL	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
85_H	WDTH	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
86_H	WDTREL	00_H	WPSEL	.6	.5	.4	.3	.2	.1	.0	–
87_H	PCON	00_H	SMOD	PDS	IDLS	SD	GF1	GF0	PDE	IDLE	–
88_H	TCON	00_H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	–
89_H	TMOD	00_H	GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0	–
$8A_H$	TL0	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
$8B_H$	TL1	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
$8C_H$	TH0	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
$8D_H$	TH1	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
90_H	P1	FF_H	T2	CLK-OUT	T2EX	$\overline{INT2}$	INT6	INT5	INT4	$\overline{INT3}$	PDIR=0
90_H	DIR1	FF_H	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=1
91_H	XPAGE	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
92_H	DPSEL	XXXX. $X000_B$	–	–	–	–	–	.2	.1	.0	–
98_H	S0CON	00_H	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0	–
99_H	S0BUF	XX_H	.7	.6	.5	.4	.3	.2	.1	.0	–
$9A_H$	IEN2	XX00. $00X0_B$	–	–	ECR	ECS	ECT	ECMP	–	ES1	–
$9B_H$	S1CON	0100. 0000_B	SM	S1P	SM21	REN1	TB81	RB81	TI1	RI1	–
$9C_H$	S1BUF	XX_H	.7	.6	.5	.4	.3	.2	.1	.0	–
$9D_H$	S1RELL	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
$A0_H$	P2	FF_H	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=0
$A0_H$	DIR2	FF_H	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=1
$A1_H$	COMSETL	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–

[1] X means that the value is indeterminate or the location is reserved.
[2] SFRs with a comment in this column are mapped registers.
Shaded registers are bit-addressable special function registers.

Table 3-5
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	Mapped by [2]
A2$_H$	COMSETH	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
A3$_H$	COMCLRL	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
A4$_H$	COMCLRH	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
A5$_H$	SETMSK	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
A6$_H$	CLRMSK	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
A8$_H$	IEN0	00$_H$	EAL	WDT	ET2	ES0	ET1	EX1	ET0	EX0	–
A9$_H$	IP0	00$_H$	OWDS	WDTS	.5	.4	.3	.2	.1	.0	–
AA$_H$	S0RELL	D9$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
B0$_H$	P3	FF$_H$	$\overline{\text{RD}}$	WR	T1	T0	$\overline{\text{INT1}}$	$\overline{\text{INT0}}$	TxD0	RxD0	PDIR=0
B0$_H$	DIR3	FF$_H$	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=1
B1$_H$	SYSCON	1010.XX01$_B$	CLKP	PMOD	1	RMAP	–	–	XMAP1	XMAP0	–
B2$_H$	SYSCON1 [3]	00XX.XEE0$_B$	ESWC	SWC	–	EA1	EA0	PRGEN1	PRGEN0	SWAP	–
B4$_H$	PRSC	1101.0101$_B$	WDTP	S0P	T2P1	T2P0	T1P1	T1P0	T0P1	T0P0	–
B8$_H$	IEN1	00$_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	–
B9$_H$	IP1	0X00.0000$_B$	PDIR	–	.5	.4	.3	.2	.1	.0	–
BA$_H$	S0RELH	XXXX.XX11$_B$	–	–	–	–	–	–	.1	.0	–
BB$_H$	S1RELH	XXXX.XX11$_B$	–	–	–	–	–	–	.1	.0	–
BC$_H$	CT1CON	X1XX.0000$_B$	–	CT1P	–	–	CT1F	CLK12	CLK11	CLK10	–
BE$_H$	IEN3	XXXX.00XX$_B$	–	–	–	–	ECT1	ECC1	–	–	–
BF$_H$	IRCON2	00$_H$	ICC17	ICC16	ICC15	ICC14	ICC13	ICC12	ICC11	ICC10	PDIR=0
BF$_H$	EICC1	FF$_H$	EICC17	EICC16	EICC15	EICC14	EICC13	EICC12	EICC11	EICC10	PDIR=1
C0$_H$	IRCON0	00$_H$	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC	–
C1$_H$	CCEN	00$_H$	COCAH3	COCAL3	COCAH2	COCAL2	COCAH1	COCAL1	COCAH0	COCAL0	–

[1] X means that the value is indeterminate or the location is reserved.
[2] SFRs with a comment in this column are mapped registers.
[3] "E" means that the value of the bit is defined by the logic level at pin PRGEN at the rising edge of the $\overline{\text{RESET}}$ or $\overline{\text{HWPD}}$ signals.
Shaded registers are bit-addressable special function registers.

Table 3-5
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	Mapped by [2]
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$_H$	T2CON	00$_H$	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0	–
C9$_H$	CC4EN	00$_H$	COCOEN1	COCON2	COCON1	COCON0	COCOEN0	COCAH4	COCAL4	COM0	–
CA$_H$	CRCL	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
CB$_H$	CRCH	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
CC$_H$	TL2	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
CD$_H$	TH2	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
CE$_H$	CCL4	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
CF$_H$	CCH4	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	–
D0$_H$	PSW	00$_H$	CY	AC	F0	RS1	RS0	OV	F1	P	–
D1$_H$	IRCON1	00$_H$	ICMP7	ICMP6	ICMP5	ICMP4	ICMP3	ICMP2	ICMP1	ICMP0	–
D2$_H$	CML0	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
D2$_H$	CC1L0	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
D3$_H$	CMH0	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
D3$_H$	CC1H0	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
D4$_H$	CML1	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
D4$_H$	CC1L1	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
D5$_H$	CMH1	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
D5$_H$	CC1H1	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
D6$_H$	CML2	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
D6$_H$	CC1L2	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
D7$_H$	CMH2	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
D7$_H$	CC1H2	00$_H$	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
D8$_H$	ADCON0	00$_H$	BD	CLK	ADEX	BSY	ADM	MX2	MX1	MX0	–
D9$_H$	ADDATH	00$_H$	.7 (MSB)	.6	.5	.4	.3	.2	.1	.0	–

[1] X means that the value is indeterminate or the location is reserved.

[2] SFRs with a comment in this column are mapped registers.
Shaded registers are bit-addressable special function registers.

Table 3-5
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	Mapped by [2]
DA_H	ADDATL	00_H	.7	.6 (LSB)	–	–	–	–	–	–	–
DB_H	P7	–	.7	.6	.5	.4	.3	.2	.1	.0	–
DC_H	ADCON1	0100.0000_B	ADCL1	ADCL0	ADST1	ADST0	MX3	MX2	MX1	MX0	–
DD_H	P8	–	–	.6	.5	.4	.3	.2	.1	.0	–
DE_H	CTRELL	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
DE_H	CT1RELL	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
DF_H	CTRELH	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
DF_H	CT1RELH	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$E0_H$	ACC	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
$E1_H$	CTCON	0100.0000_B	T2PS1	CTP	ICR	ICS	CTF	CLK2	CLK1	CLK0	–
$E2_H$	CML3	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$E2_H$	CC1L3	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$E3_H$	CMH3	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$E3_H$	CC1H3	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$E4_H$	CML4	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$E4_H$	CC1L4	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$E5_H$	CMH4	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$E5_H$	CC1H4	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$E6_H$	CML5	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$E6_H$	CC1L5	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$E7_H$	CMH5	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$E7_H$	CC1H5	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$E8_H$	P4	FF_H	CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0	PDIR=0
$E8_H$	DIR4	FF_H	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=1
$E9_H$	MD0	XX_H	.7	.6	.5	.4	.3	.2	.1	.0	–
EA_H	MD1	XX_H	.7	.6	.5	.4	.3	.2	.1	.0	–
EB_H	MD2	XX_H	.7	.6	.5	.4	.3	.2	.1	.0	–
EC_H	MD3	XX_H	.7	.6	.5	.4	.3	.2	.1	.0	–
ED_H	MD4	XX_H	.7	.6	.5	.4	.3	.2	.1	.0	–

[1] X means that the value is indeterminate or the location is reserved.
[2] SFRs with a comment in this column are mapped registers.
Shaded registers are bit-addressable special function registers.

Table 3-5
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	Mapped by [2]
EE_H	MD5	XX_H	.7	.6	.5	.4	.3	.2	.1	.0	–
EF_H	ARCON	$0XXX.XXXX_B$	MDEF	MDOV	SLR	SC.4	SC.3	SC.2	SC.1	SC.0	–
$F0_H$	B	00_H	.7	.6	.5	.4	.3	.2	.1	.0	–
$F2_H$	CML6	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$F2_H$	CC1L6	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$F3_H$	CMH6	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$F3_H$	CC1H6	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$F4_H$	CML7	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$F4_H$	CC1L7	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$F5_H$	CMH7	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$F5_H$	CC1H7	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$F6_H$	CMEN	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$F6_H$	CC1EN	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$F7_H$	CMSEL	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=0
$F7_H$	CAFR	00_H	.7	.6	.5	.4	.3	.2	.1	.0	RMAP=1
$F8_H$	P5	FF_H	CCM7	CCM6	CCM5	CCM4	CCM3	CCM2	CCM1	CCM0	PDIR=0
$F8_H$	DIR5	FF_H	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=1
$F9_H$	P9	FF_H	CC17	CC16	CC15	CC14	CC13	CC12	CC11	CC10	PDIR=0
$F9_H$	DIR9	FF_H	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=1
FA_H	P6	FF_H	.7	.6.	.5	.4	.3	TxD1	RxD1	$\overline{\text{ADST}}$	PDIR=0
FA_H	DIR6	FF_H	.7	.6	.5	.4	.3	.2	.1	.0	PDIR=1

[1] X means that the value is indeterminate or the location is reserved.
[2] SFRs with a comment in this column are mapped registers.
Shaded registers are bit-addressable special function registers.

4 External Bus Interface

The C509-L allows for external memory expansion. To accomplish this, the external bus interface common to most 8051-based microcontrollers is employed.

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 PSEN (program store enable) as a read strobe. Accesses to external data memory use RD and 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.

4.1.2 Timing

The timing of the external bus interface, in particular the relationship between the control signals ALE, PSEN, RD, WR and information on port 0 and port 2, is illustrated 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 WR is activated and remains there until after WR is deactivated. In a read cycle, the incoming byte is accepted at port 0 before the read strobe is deactivated.

Program memory: Signal PSEN functions as a read strobe.

4.1.3 External Program Memory Access

The external program memory is accessed whenever signal EA is active (low): Due to the 64K internal ROM, no mixed internal/external program memory execution is possible.

When the CPU is executing out of external program memory, all 8 bits of port 2 are dedicated to an output function and may not be used for general-purpose I/O. The contents 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).

When the C509-L executes instructions from external program memory, port 2 is at all times dedicated to output the high-order address byte. This means that port 0 and port 2 of the C509-L can never be used as general-purpose I/O.

4.1.4 PSEN, Program Store Enable

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

4.1.5 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 C509-L the external program and data memory spaces can be combined by AND-ing PSEN and RD. A positive logic AND of these two signals produces an active low read strobe that can be used for the combined physical memory. Since the PSEN cycle is faster than the RD cycle, the external memory needs to be fast enough to adapt to the PSEN cycle.

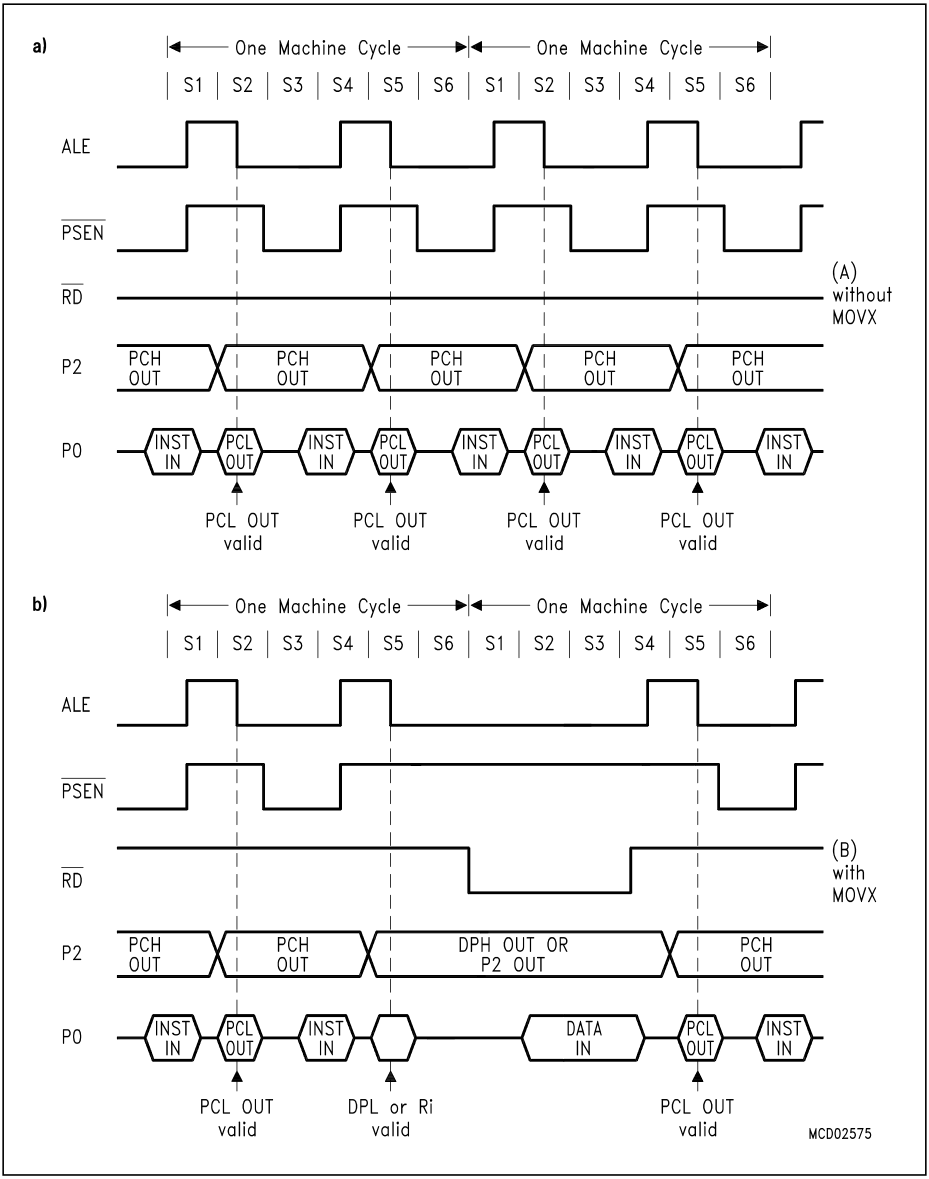


Figure 4-1
External Program Memory Execution

4.1.6 ALE, Address Latch Enable

The main function of ALE is to provide a properly timed signal to latch the low byte of an address from P0 into an external latch during fetches from external memory. The address byte is valid at the negative transition of ALE. For that purpose, ALE is activated twice every machine cycle. This activation takes place even if the cycle involves no external fetch. The only time no ALE pulse comes out is during an access to external data memory when RD/WR signals are active. The first ALE of the second cycle of a MOVX instruction is missing (see **figure 4-1 b**). Consequently, in any C509-L system that does not use data memory, ALE is activated at a constant rate of 1/3 of the oscillator frequency and can be used for external clocking or timing purposes.

As a reserved function for future versions, the C509-L allows to switch off the ALE output signal by a bit in SFR SYSCON. If $\overline{\text{EA}}$=0 (this is always the case for the C509-L), the ALE generation is always enabled and resetting of bit 5 of SFR SYSCON has no effect.

Special Function Register SYSCON (Address B1$_H$) **Reset Value : 1010XX01$_B$**

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
B1$_H$	CLKP	PMOD	1	RMAP	–	–	XMAP1	XMAP0	SYSCON

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

Bit	Function
Bit 5	Reserved bit for future use: Enable/disable ALE output. Bit 5 is at “1” after reset and can be written with “0” or “1”.

4.2 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 (not true for the C509-l, because it lacks internal program memory).
Each 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^{TM}, 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.

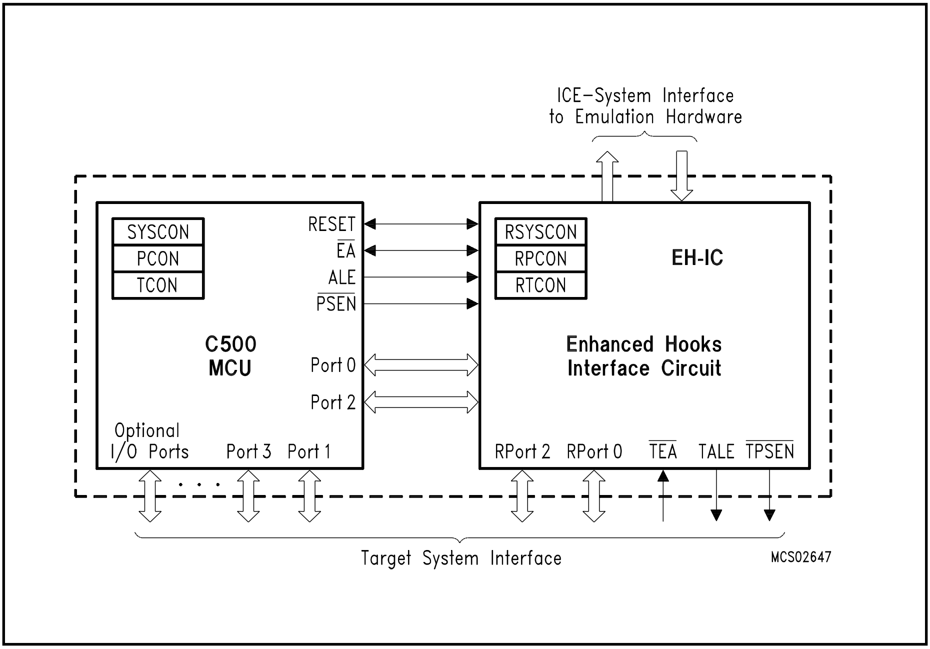


Figure 4-2
Basic C500 MCU Enhanced Hooks Concept Configuration

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.

4.3 Eight Datapointers for Faster External Bus Access

4.3.1 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.

4.3.2 How the Eight Datapointers of the C509-L 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 C509-L contains a set of eight 16-bit registers from which the actual datapointer can be selected.

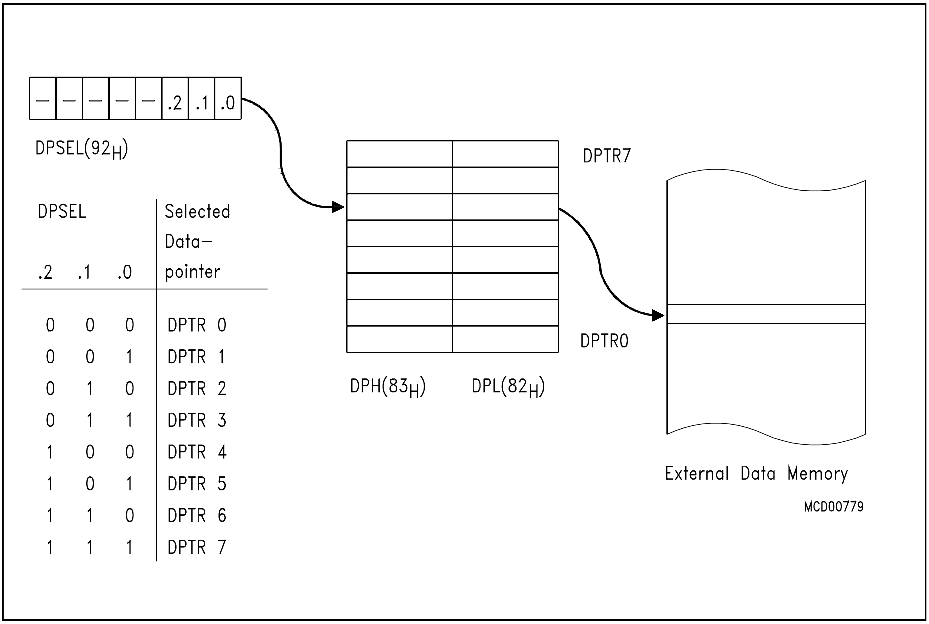
This means that the user’s program may keep up to eight 16-bit addresses resident in these registers, but only one register at a time is selected to be the datapointer. Thus the datapointer in turn is accessed (or selected) via indirect addressing. This indirect addressing is done through a special function register called DPSEL (data pointer select register). All instructions of the C509-L 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							LSB	
	7	6	5	4	3	2	1	0	
92_H	–	–	–	–	–	.2	.1	.0	DPSEL

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



DPSEL .2	.1	.0	Selected Data-pointer
0	0	0	DPTR 0
0	0	1	DPTR 1
0	1	0	DPTR 2
0	1	1	DPTR 3
1	0	0	DPTR 4
1	0	1	DPTR 5
1	1	0	DPTR 6
1	1	1	DPTR 7

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

4.3.3 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.

4.3.4 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 an 8051)

Initialization Routine

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

Table Look-up Routine under Real Time Conditions

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

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

Example 2: Using Two Datapointers (Code for an C509)

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 C509’s multiple datapointers can make external bus accesses two times as fast as with a standard 8051 or 8051 derivative. Here, four data variables in the internal RAM and two additional stack bytes were spared, too. This means for some applications where all eight datapointers are employed that an C509 program has up to 24 byte (16 variables and 8 stack bytes) of the internal RAM free for other use.

4.4 XRAM Operation

The XRAM in the C509-L is a memory area that is logically located at the upper end of the external memory space, but is integrated on the chip. Because the XRAM is used in the same way as external data memory the same instruction types (MOVX) must be used for accessing the XRAM.

4.4.1 XRAM Access Control

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

Special Function Register SYSCON (Address B1$_H$) **Reset Value : 1010XX01$_B$**

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
B1$_H$	CLKP	PMOD	1	RMAP	–	–	XMAP1	XMAP0	SYSCON

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

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

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

After a reset operation, bit XMAP0 is reset. This means that the accesses to the XRAM is generally disabled. In this case, all accesses using MOVX instructions with an address in the range of $F400_H$ to $FFFF_H$ generate external data memory bus cycles. When XMAP0 is set, the access to the XRAM is enabled and all accesses using MOVX instructions with an address in the range of $F400_H$ to $FFFF_H$ will access internally the XRAM.

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

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

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

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

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

For accessing the XRAM, the effective address stored in DPTR must be in the range of $F400_H$ to $FFFF_H$. For accessing the CAN controller, the effective address stored in DPTR must be in the range of $F700_H$ to $F7FF_H$.

4.4.3 Accesses to XRAM using the Registers R0/R1

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

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

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

Hence, a special page register is implemented into the C509-L to provide the possibility of accessing the XRAM also with the MOVX @Ri instructions, i.e. XPAGE serves the same function for the XRAM as Port 2 for external data memory.

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

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

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

Figures 4-4 and **4-6** 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.

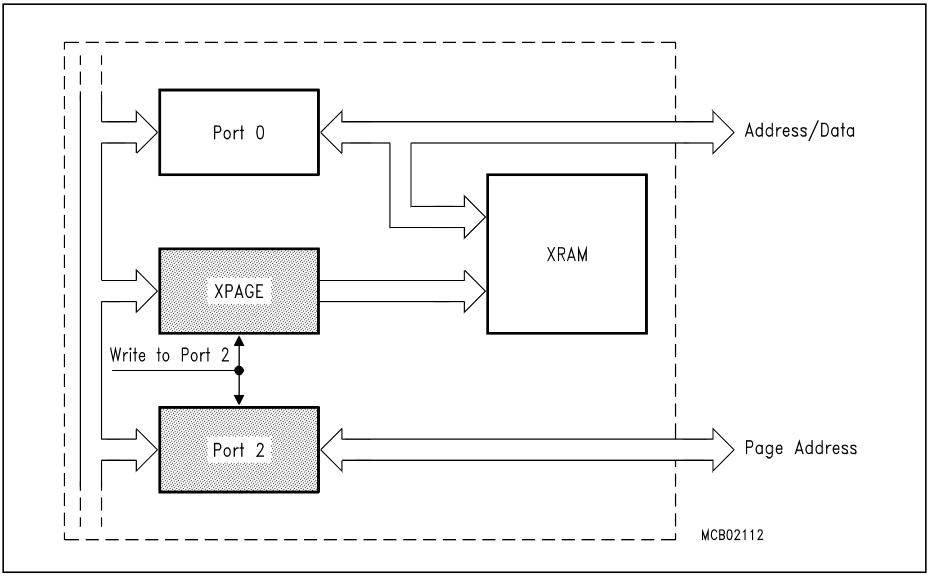


Figure 4-4
Write Page Address to Port 2

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

When external RAM is to be accessed in the XRAM address range (F400$_H$ - FFFF$_H$), XRAM has to be disabled. When additional external RAM is to be addressed in an address range ≤ XRAM (F400$_H$) XRAM may remain being enabled and there is no need to overwrite XPAGE by a second move.

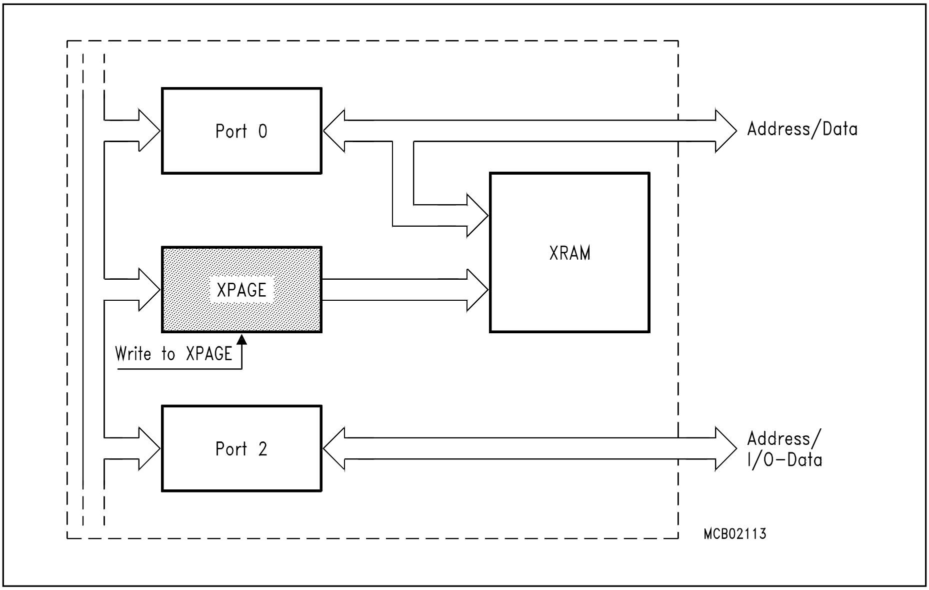


Figure 4-5
Write Page Address to XPAGE

The page address is only written to the XPAGE register. Port 2 is available for addresses or I/O data. See **figure 4-6** to see what happens when Port 2 is used as I/O-Port.

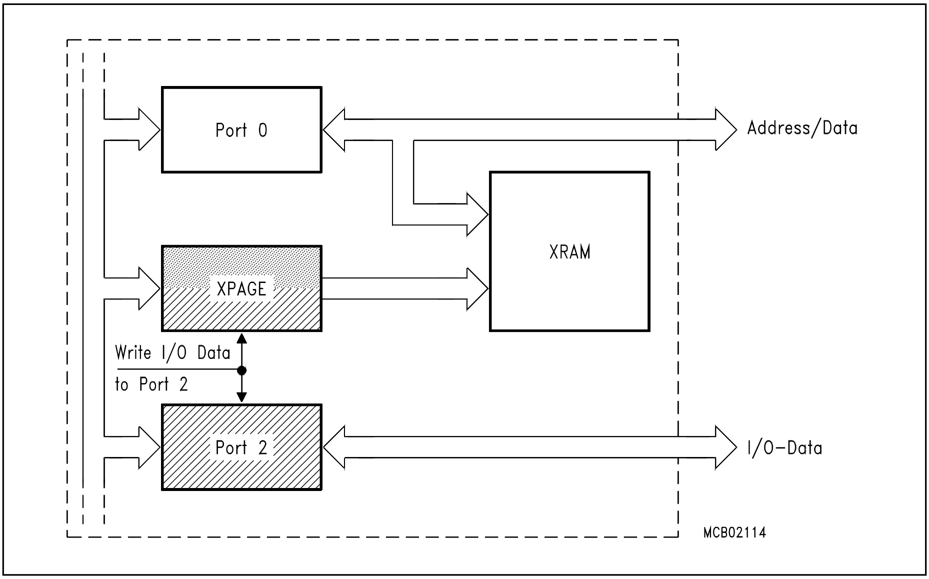


Figure 4-6
Usage of Port 2 as I/O Port

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

Example:

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

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

The register XPAGE provides the upper address byte for accesses to XRAM with MOVX @Ri instructions. If the address formed by XPAGE and Ri points outside the XRAM address range, an external access is performed. For the C509-L the contents of XPAGE must be greater or equal $F3_H$ in order to use the XRAM/.

Thus, the register XPAGE is used for addressing of the XRAM; additionally its contents are used for generating the internal XRAM select. If the contents of XPAGE is less than the XRAM address range then an external bus access is performed where the upper address byte is provided by P2 and not by XPAGE!

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

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

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

4.4.4 Reset Operation of the XRAM

The content of the XRAM is not affected by a reset. After power-up the content is undefined, while it remains 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.

4.4.5 Behaviour of Port0 and Port2

The behaviour of Port 0 and P2 during a MOVX access depends on the control bits in register SYSCON and on the state of pin EA. The **table 4-1** 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 is accessed, the data written to the XRAM can be seen on the bus in debug mode.
I/0: The pins work as Input/Output lines under control of their latch.
b) Activation of the RD and WR pin during the access.
c) Use of internal or external XDATA memory.

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

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

▨ modes compatible to 8051/C501 family

Table 4-1
Behaviour of P0/P2 and $\overline{RD}/\overline{WR}$ During MOVX Accesses

5 Reset and System Clock Operation

5.1 Reset Function and Circuitries

The hardware reset function incorporated in the C509-L 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.

Additionally to the hardware reset, which is applied externally to the C509-L, there are two internal reset sources, the watchdog timer and the oscillator watchdog. They are described in detail in section 8 "Fail-Save Mechanisms". The actual chapter only deals with the external hardware reset.

The reset input is an active low input at pin $\overline{\text{RESET}}$. An internal Schmitt trigger is used at the input for noise rejection. Since the reset is synchronized internally, the $\overline{\text{RESET}}$ pin must be held low for at least two machine cycles (12 oscillator periods) while the oscillator is running. With the oscillator running the internal reset is executed during the second machine cycle in which $\overline{\text{RESET}}$ is low and is repeated every cycle until $\overline{\text{RESET}}$ goes high again.

During reset, pins ALE and $\overline{\text{PSEN}}$ are configured as inputs and should not be stimulated 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).

A pullup resistor is internally connected to V_{CC} to allow a power-up reset with an external capacitor only. An automatic reset can be obtained when V_{CC} is applied by connecting the reset pin to V_{SS} via a capacitor as shown in **figure 5-1 a)** and **c)**. After V_{CC} has been turned on, the capacitor must hold the voltage level at the reset pin for a specified time below the upper threshold of the Schmitt trigger 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.

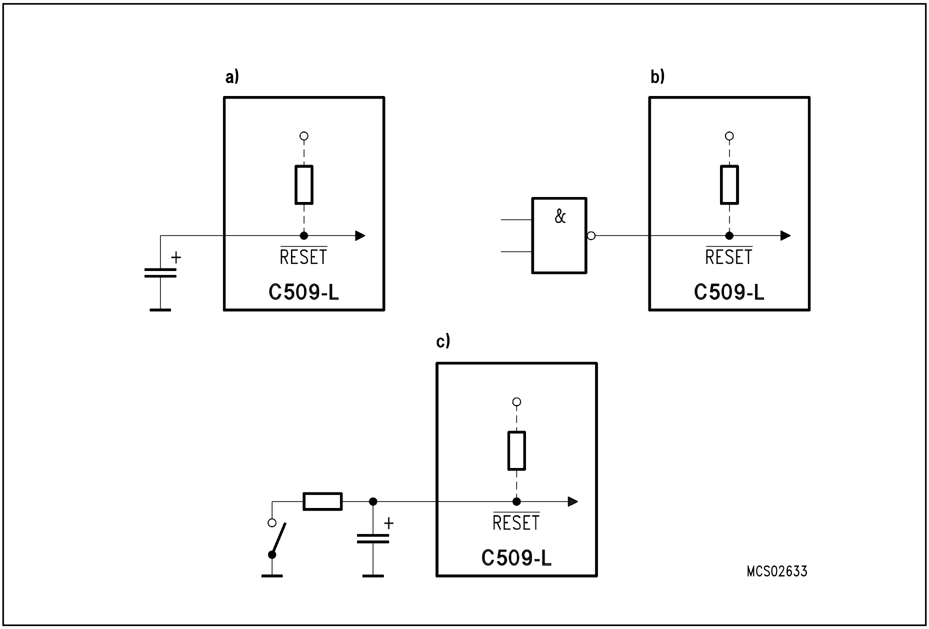


Figure 5-1
Reset Circuitries

A correct reset leaves the processor in a defined state. The program execution starts at location 0000_H. Depending on the state of the PRGEN pin, either external ROM/EPROM is accessed (Normal Mode) or the C509-L starts with the bootstrap loader program located in the Boot ROM (Bootstrap Mode). The default values of the special function registers (SFR) to which they are forced during reset are listed in **table 3-4** and **3-5** of chapter 3.

After reset is internally accomplished the port latches of ports 0 to 6 are set to FF_H. This leaves port 0 floating, since it is an open drain port when not used as data/address bus. All other I/O port lines (ports 1 to 6 and 9) output a one (1). Ports 7 and 8, which are input-only ports, have no internal latch and therefore the contents of the special function registers P7 and P8 depend on the levels applied to ports 7 and 8.

The content of the internal RAM of the C509-L is not affected by a reset. After power-up the content is undefined, while it remains unchanged during a reset it the power supply is not turned off.

5.2 Hardware Reset Timing

This section describes the timing of the hardware reset signal.

The input pin $\overline{\text{RESET}}$ is sampled once during each machine cycle. This happens in state 5 phase 2. Thus, the external reset signal is synchronized to the internal CPU timing. When the reset is found active (low level at pin $\overline{\text{RESET}}$) 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 not performed if there is no clock available at the device (This can be avoided using the oscillator watchdog, which provides an auxiliary clock for performing a correct reset without clock at the XTAL1 and XTAL2 pins). The $\overline{\text{RESET}}$ signal must be active for at least two machine cycles; after this time the C509-L 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 5-2 shows this timing for a configuration with $\overline{\text{EA}}$ = 0 (external program memory). Thus, between the release of the $\overline{\text{RESET}}$ signal and the first falling edge at ALE there is a time period of at least one machine cycle but less than two machine cycles.

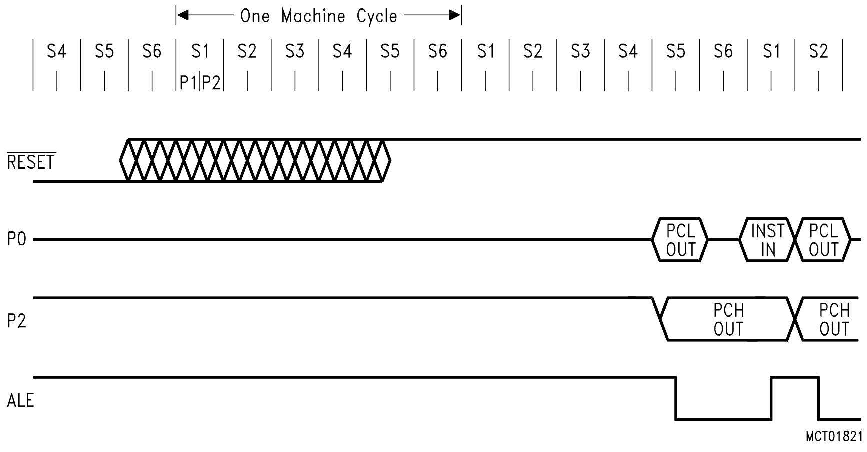


Figure 5-2
CPU Timing after Reset

5.3 Fast Internal Reset after Power-On

The C509-L uses the oscillator watchdog unit (see also chapter 8) for a fast internal reset procedure after power-on. **Figure 5-3** shows the power-on sequence under control of the oscillator watchdog.

Normally the devices of the 8051 family enter their default reset state not before the on-chip oscillator starts. The reason is that the external reset signal must be internally synchronized and processed in order to bring the device into the correct reset state. Especially if a crystal is used the start up time of the oscillator is relatively long (max. 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 C509-L 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 5-3**).

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 finally has 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 5-3, II**). Subsequently, the clock is supplied by the on-chip oscillator and the oscillator watchdog's reset request is released (**figure 5-3, III**). However, an externally applied reset still remains active (**figure 5-3, IV**) and the device does not start program execution (**figure 5-3, 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 Hardware Power-Down Mode (a $\overline{\text{HWPD}}$ signal is overriden by reset)
- Termination of the 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 for clock generation, the external reset signal must be hold active at least until the on-chip oscillator has started (max.10 ms) and the internal watchdog reset phase is completed (after phase III in **figure 5-3**). When an external clock generator is used, phase II is very short. Therefore, an external reset time of typically 1 ms is sufficient in most applications.

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

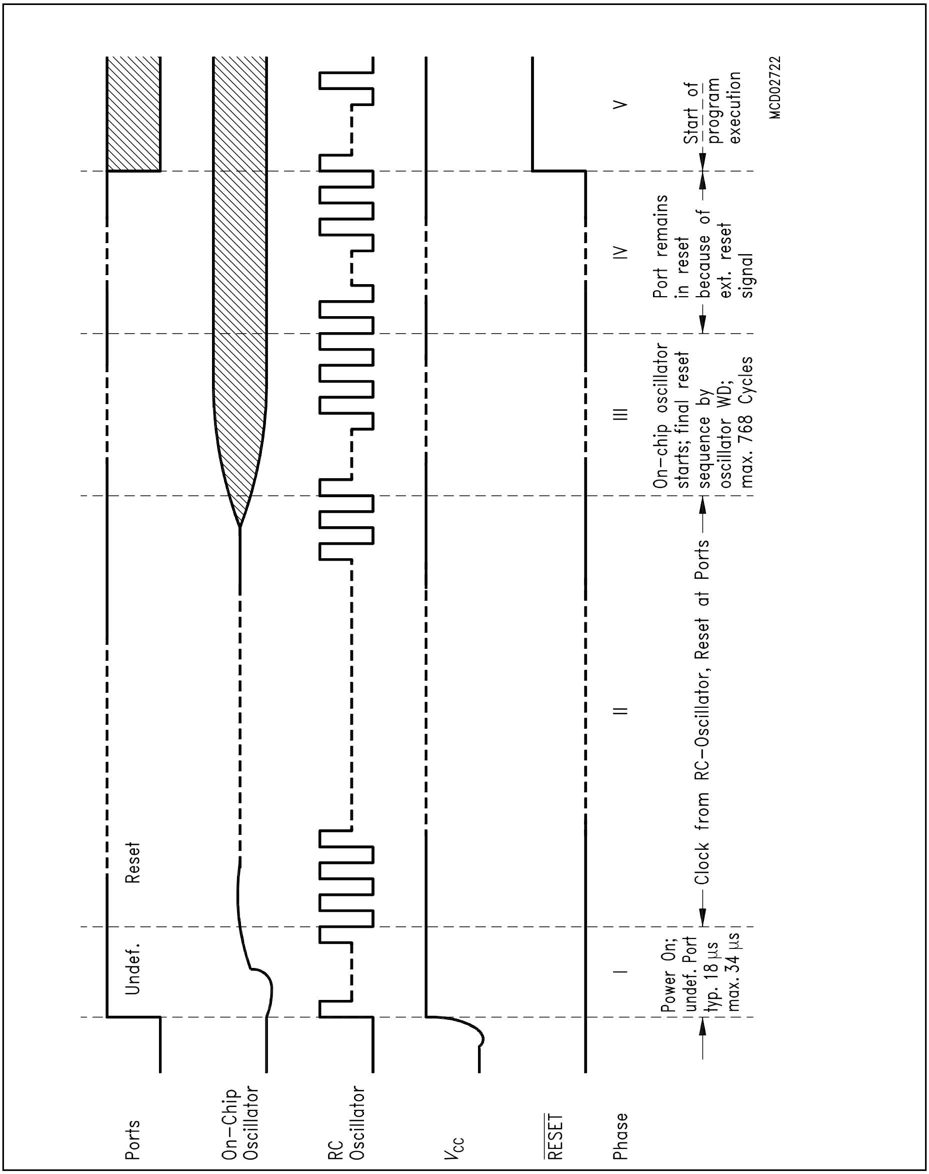

Ports
On-Chip Oscillator
RC Oscillator
V_{CC}
$\overline{\text{RESET}}$
Phase
Undef.
Reset
I
II
III
IV
V
Power On; undef. Port typ. 18 µs max. 34 µs
Clock from RC-Oscillator, Reset at Ports
On-chip oscillator starts; final reset sequence by oscillator WD; max. 768 Cycles
Port remains in reset because of ext. reset signal
Start of program execution
MCD02722

Figure 5-3
Power-On Reset of the C509-L

5.4 Reset Output Pin ($\overline{\text{RO}}$)

As mentioned before the C509-L internally synchronizes an external reset signal at pin $\overline{\text{RESET}}$ in order to perform a reset procedure. Additionally, the C509-L provides several "fail-save" mechanisms, e.g. watchdog timer and oscillator watchdog, which can internally generate a reset, too. Thus, it is often important to inform also the peripherals external to the chip that a reset is being performed and that the controller will soon start its program again.

For that purpose, the C509-L has a pin dedicated to output the internal reset request. This reset output ($\overline{\text{RO}}$) shows the internal (and already synchronized) reset signal requested by any of the three possible sources in the C509-L: external hardware reset, watchdog timer reset, or oscillator watchdog reset. The duration of the active low signal of the reset output depends on the source which requests it. In the case of the external hardware reset it is the synchronized external reset signal at pin $\overline{\text{RESET}}$. In the case of a watchdog timer reset or oscillator watchdog reset the $\overline{\text{RO}}$ signal takes at least two machine cycles, which is the minimal duration for a reset request allowed. For details - how the reset requests are OR-ed together and how long they last - see also chapter 8 "Fail-Save Mechanisms" .

5.5 Oscillator and Clock Circuit

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

Figure 5-4 shows the recommended oscillator circuit.

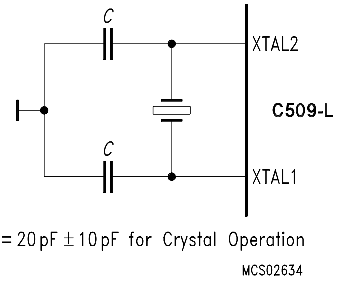


Figure 5-4
Recommended Oscillator Circuit

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. It a ceramic resonator is used, C_1 and C_2 are normally selected to be of somewhat higher values, typically 47 pF. We recommend consulting the manufacturer of the ceramic resonator for value specifications of these capacitors.

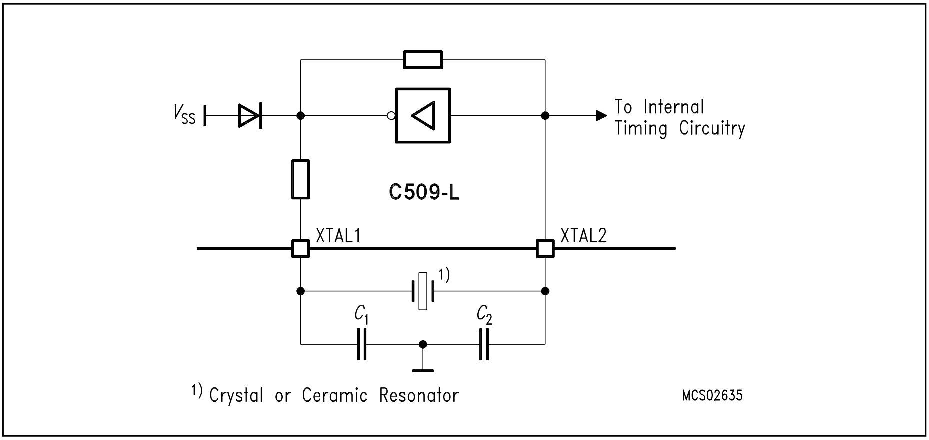


Figure 5-5
On-Chip Oscillator Circuitry

To drive the C509-L with an external clock source, the external clock signal has to be applied to XTAL2, as shown in **figure 5-6**. XTAL1 has to be left unconnected. A pullup resistor is suggested (to increase the noise margin), but is optional if V_{OH} of the driving gate corresponds to the VI_{H2} specification of XTAL2.

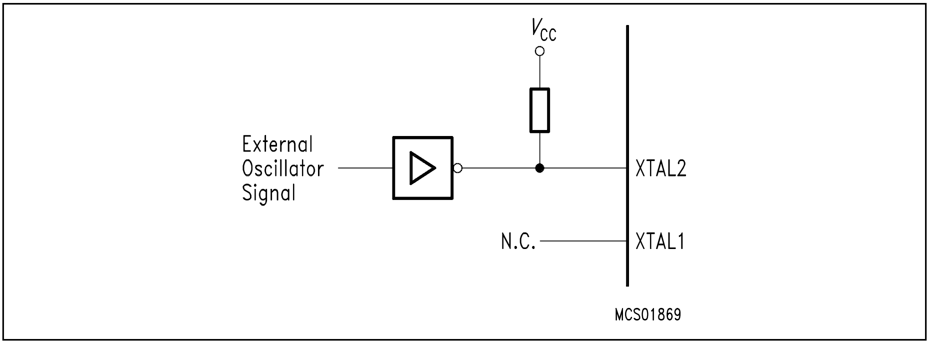


Figure 5-6
External Clock Source

5.6 System Clock Output

For peripheral devices requiring a system clock, the C509-L 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 or 1/12 of the oscillator frequency (depending on bit CLKP in SFR SYSCON) is gated to pin P1.6/CLKOUT. To use this function the port pin must be programmed to a one (1), which is also the default after reset.

Special Function Register ADCON0 (Address $D8_H$) **Reset Value : 00_H**
Special Function Register SYSCON (Address $B1_H$) **Reset Value : $1010XX01_B$**

	MSB							LSB	
Bit No.	DF_H	DE_H	DD_H	DC_H	DB_H	DA_H	$D9_H$	$D8_H$	
$D8_H$	BD	CLK	ADEX	BSY	ADM	MX2	MX1	MX0	ADCON0
	7	6	5	4	3	2	1	0	
$B1_H$	CLKP	PMOD	EALE	RMAP	–	–	XMAP1	XMAP0	SYSCON

These bits are not used in controlling the clock output function.

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

CLKP	Prescaler control for the clock output signal CLKOUT
	CLKP = 0 : CLKOUT frequency is $f_{OSC}/6$
	CLKP = 1 : CLKOUT frequency is $f_{OSC}/12$ (default after reset)

A timing diagram of the system clock output is shown in **figure 5-7**. This timing assumes that CLK=1 and CLKP=0.

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

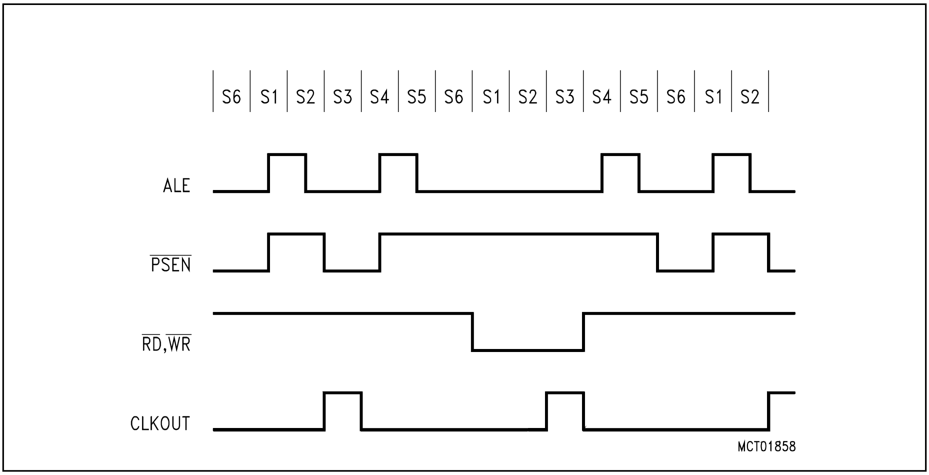


Figure 5-7
Timing Diagram - System Clock Output

6 On-Chip Peripheral Components

This chapter gives detailed information about all on-chip peripherals of the C509-L except for the integrated interrupt controller, which is described separately in **chapter 7**.

6.1 Parallel I/O

6.1.1 Port Structures

Digital I/O Ports

The C509-L allows for digital I/O on 64 lines grouped into 8 bidirectional 8-bit ports. Each port bit consists of a latch, an output driver and an input buffer. Read and write accesses to the I/O ports P0 through P6 and P9 are performed via their corresponding special function registers P0 to P6 and P9. The port structure of the C509-L is designed to operate either as a quasi-bidirectional port structure, compatible to the standard 8051-Family, or as a genuine bidirectional port structure. This port operating mode can be selected by software (setting or clearing the bit PMOD in the SFR SYSCON).

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.

Analog Input Ports

Ports 7 and 8 are available as input ports only and provide for two functions. When used as digital inputs, the corresponding SFR’s P7 and P8 contain the digital value applied to port 7 and port 8 lines. When used for analog inputs the desired analog channel is selected by a three-bit field in SFR ADCON0 or a four-bit field in SFR ADCON1. Of course, it makes no sense to output a value to these input-only ports by writing to the SFR’s P7 or P8; this will have no effect.

If a digital value is to be read, the voltage levels are to be held within the input voltage specifications (V_{IL}/V_{IH}). Since P7 and P8 are not bit-addressable registers, all input lines of P7 or P8 are read at the same time by byte instructions.

Nevertheless, it is possible to use ports 7 and 8 simultaneously for analog and digital input. However, care must be taken that all bits of P7 or P8 that have an undetermined value caused by their analog function are masked.

In order to guarantee a high-quality A/D conversion, digital input lines of port 7 and port 8 should not toggle while a neighboring port pin is executing an A/D conversion. This could produce crosstalk to the analog signal.

Figure 6-1 shows a functional diagram of a typical bit latch and I/O buffer, which is the core of each of the 8 digital 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 self is placed on the internal bus in response to a "read-pin" signal from the CPU. Some instructions that read from a port activate the "read-port-latch" signal, while others activate the "read-port-pin" signal.

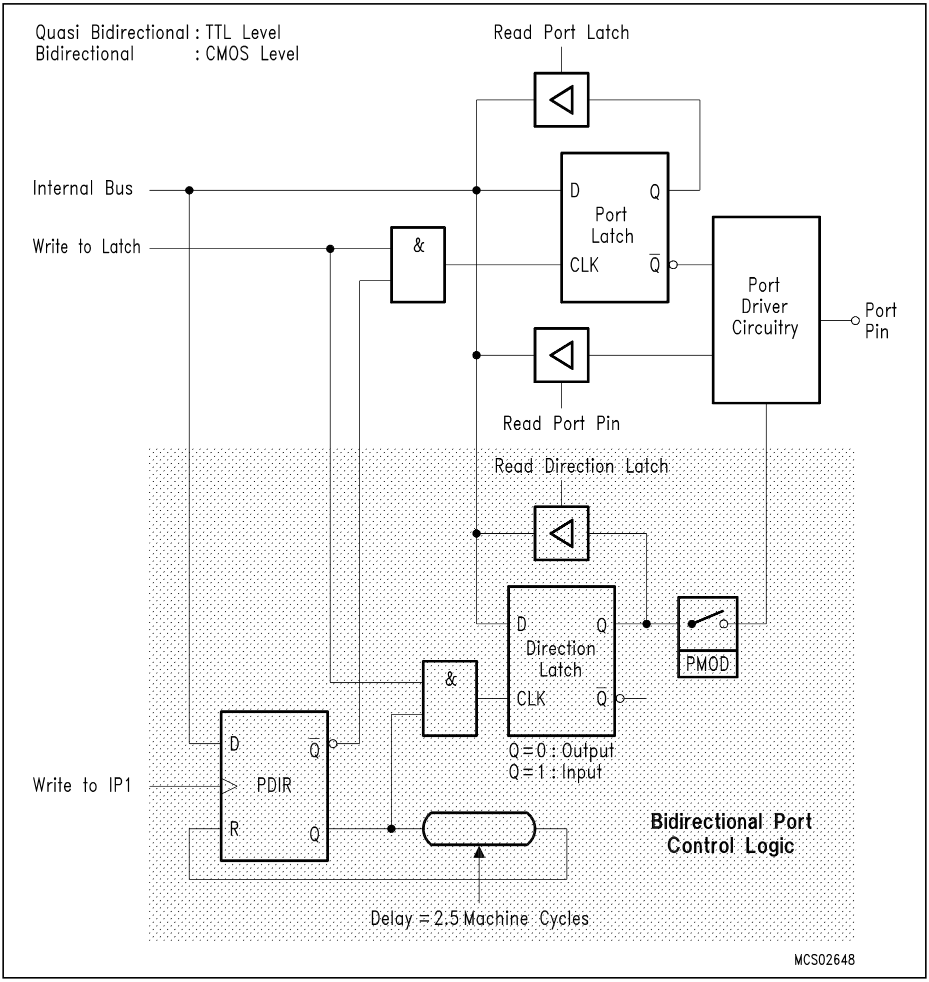
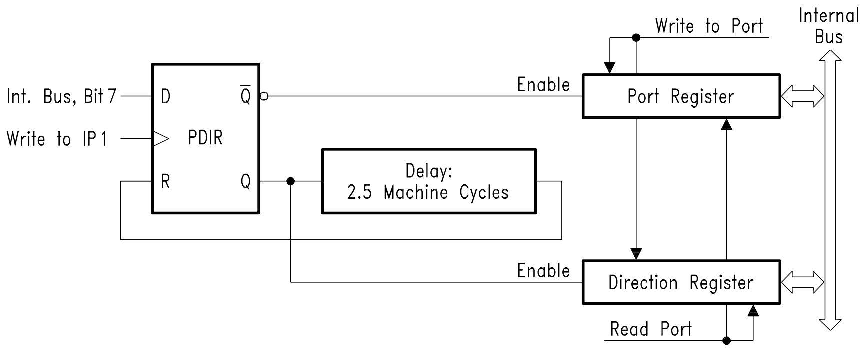


Figure 6-1
Basic Structure of a Port Circuitry

The shaded area in **figure 6-1** shows the control logic in the C509-L, which has been added to the functionality of the standard 8051 digital I/O port structure. This control logic is used to provide an additional bidirectional port structure with CMOS voltage levels.

6.1.1.1 Port Structure Selection

After a reset operation of the C509-L, the quasi-bidirectional 8051-compatible port structure is selected. For selection of the bidirectional port structure (CMOS) the bit PMOD of SFR SYSCON must be set. Because each port pin can be programmed as an input or an output, additionally, after the selection of the bidirectional mode the direction register of the ports must be written (except the analog/digital input ports 7,8). This direction registers are mapped to the port registers. This means, the port register address is equal to its direction register address. **Figure 6-2** illustrates the port- and direction register configuration.



Instruction sequence for the programming of the direction registers:

```
ORL   IP1, #80H      ; Set bit PDIR
MOV   DIRx, #0YYH    ; Write port x direction register with value YYH
```

Figure 6-2
Port Register, Direction Register

For the access the direction registers a double instruction sequence must be executed. The first instruction has to set bit PDIR in SFR IP1. Thereafter, a second instruction can read or write the direction registers. PDIR will automatically be cleared after the second machine cycle (S2P2) after having been set. For this time, the access to the direction register is enabled and the register can be read or written. Further, the double instruction sequence as shown in **figure 6-2**, cannot be interrupted by an interrupt,
When the bidirectional port structure is activated (PMOD=1) after a reset, the ports are defined as inputs (direction registers default values after reset are set to FF_H).
With PMOD = 0 (quasi-bidirectional port structure selected), any access to the direction registers has no effect on the port driver circuitries.

Special Function Register SYSCON (Address B1$_H$) **Reset Value : 1010XX01$_B$**
Special Function Register IP1 (Address B9$_H$) **Reset Value : 0X000000$_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
B1$_H$	CLKP	PMOD	1	RMAP	–	–	XMAP1	XMAP0	SYSCON
	7	6	5	4	3	2	1	0	
B9$_H$	PDIR	–	.5	.4	.3	.2	.1	.0	IP1

The shaded bits are not used for port selection.

Bit	Function
PMOD	Port mode selection PMOD = 0: Quasi-bidirectional port structure is selected (reset value) PMOD = 1: Bidirectional port structure is selected.
PDIR	Direction register enable PDIR = 0: Port register access is enabled (reset value) PDIR = 1: Direction register is enabled. PDIR will automatically be cleared after the second machine cycle (S2P2) after having been set.

Direction Registers DIR0-DIR6, DIR9 (Mapped Address = Port Address) **Reset Value : FF$_H$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
Address. = Port-Addr.	.7	.6	.5	.4	.3	.2	.1	.0	Direction Register

Bit	Function
Bit 7...0	Port driver circuitry, input/output selection Bit = 0: Port line is in output mode Bit = 1: Port line is in input mode (reset value). This register can only be read and written by software when bit PDIR (IP1) was set one instruction before.

6.1.1.2 Quasi-Bidirectional Port Structure

6.1.1.2.1 Basic Port Circuitry of Port 1 to 6 and 9

The basic quasi-bidirectional port structure as shown in the upper part of the schematics of **figure 6-1** provides a port driver circuit which is build up by an internal pullup FET as shown in **figure 6-3**. 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 6-3**: Q=0), which turns off the output driver FET n1. Then, for ports 1 to 6 and 9, 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 sometimes called “quasi-bidirectional”.

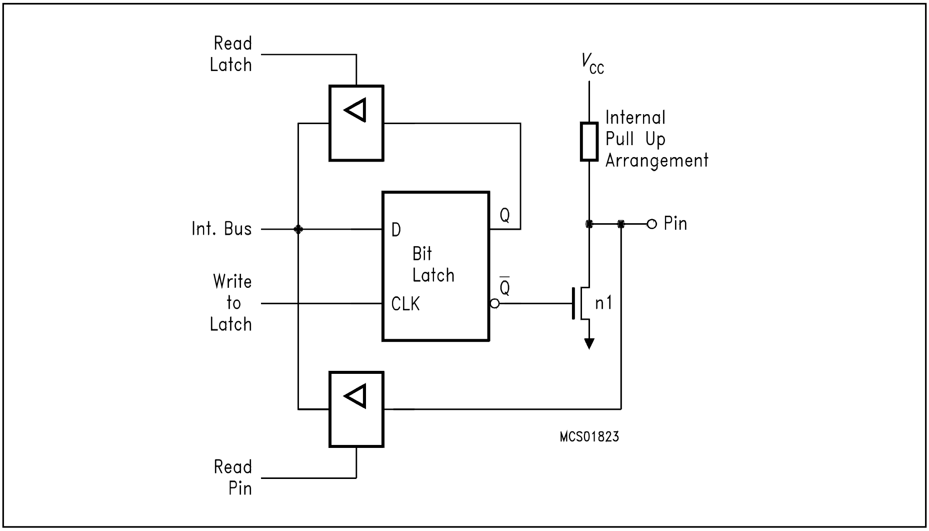


Figure 6-3
Basic Output Driver Circuit of Ports 1 to 6 and 9

In fact, the pullups mentioned before and included in **figure 6-3** are pullup arrangements as shown in **figure 6-4**. One n-channel pulldown FET and three pullup FETs are used:

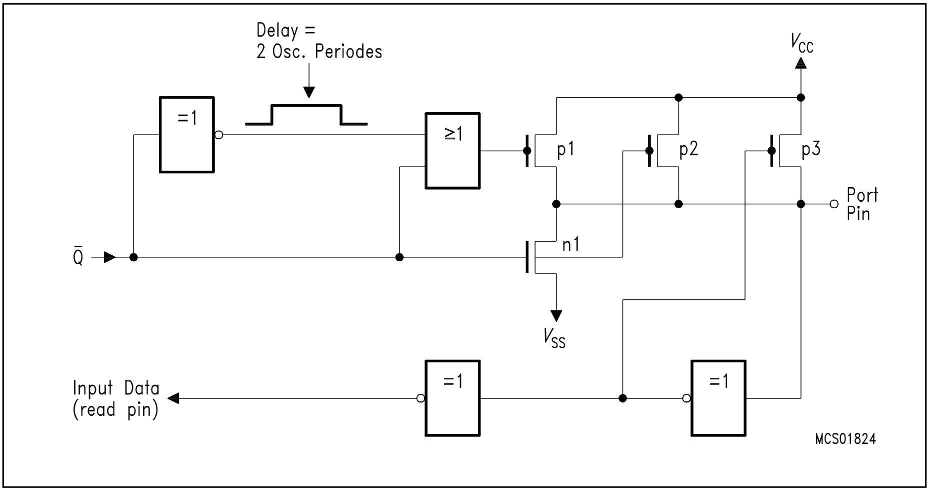


Figure 6-4
Output Driver Circuit of Ports 1 to 6 and 9

- 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 results in four states which 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.

6.1.1.2.2 Port 0 Circuitry

Port 0, in contrast to ports 1 to 6 and 9, 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 6-5**) 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.

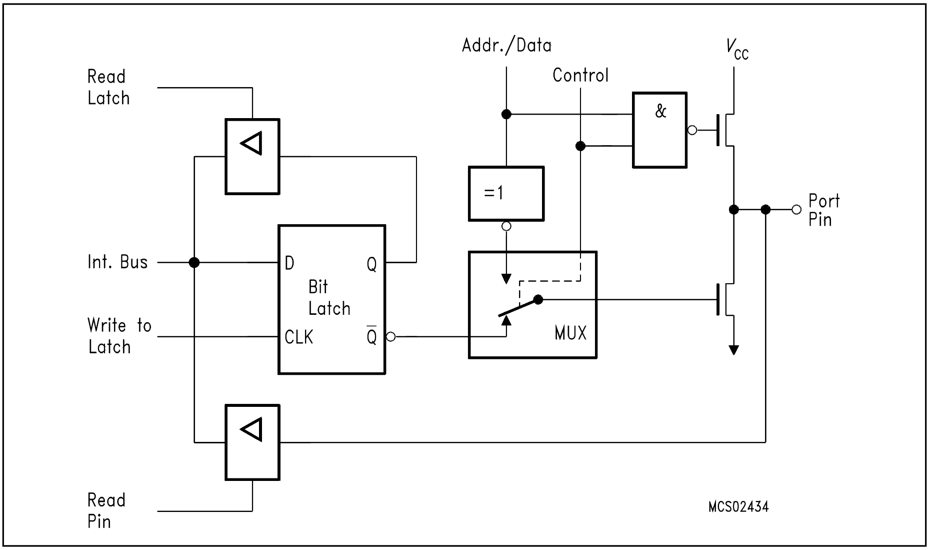


Figure 6-5
Port 0 Circuitry

6.1.1.2.3 Port 0 and Port 2 used as Address/Data Bus

As shown in **figure 6-5** and below in **figure 6-6**, 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 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 P2 SFR remains unchanged while the P0 SFR has 1's written to it. Being an address/data bus, port 0 uses a pullup FET as shown in **figure 6-5**. When a 16-bit address is used, port 2 uses the additional strong pullups p1 to emit 1's for the entire external memory cycle instead of the weak ones (p2 and p3) used during normal port activity.

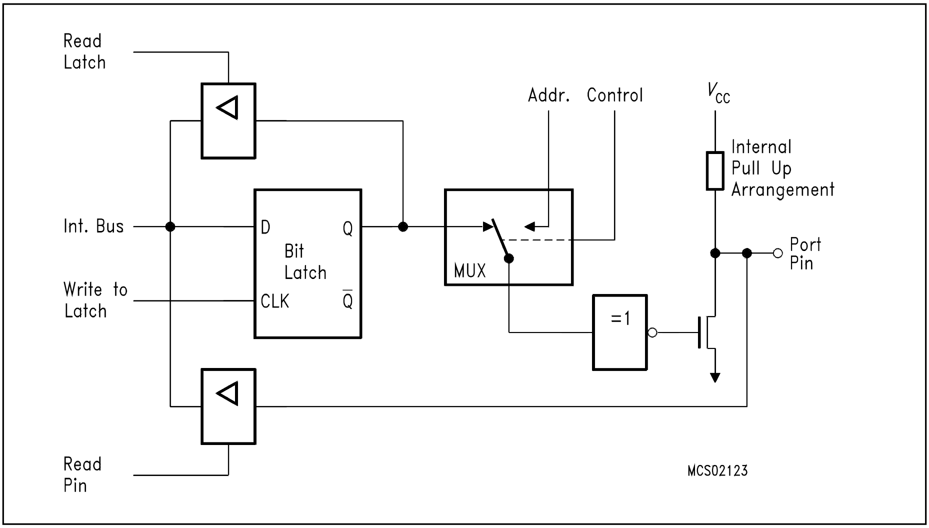


Figure 6-6
Port 2 Circuitry

6.1.1.3 Bidirectional (CMOS) Port Structure

6.1.1.3.1 Input Mode

Figure 6-7 shows the bidirectional port structure in the input mode.

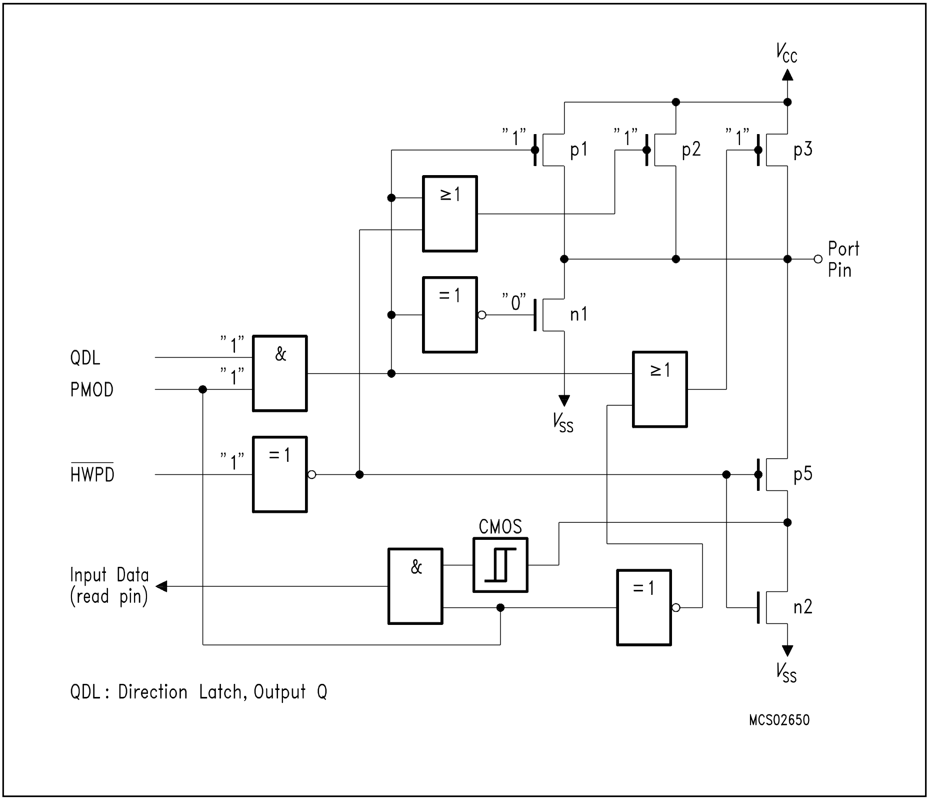


Figure 6-7
Bidirectional Port Structure - Input Mode

The input mode for a port pin is selected by programming the corresponding direction bit to '1' (QDL='1'). The FETs p1, p2, p3 and n1 are switched off. Through a Schmitt-Trigger, designed to detect CMOS levels, the input signal is lead to the internal bus where it can be read by the microcontroller.

6.1.1.3.2 Output Mode

The output mode for a port pin is selected by programming the corresponding direction bit to '0' (QDL='0'). The contents of the port latch determines whether a '1' ($\overline{\text{QPL}}$='0') or a '0' ($\overline{\text{QPL}}$='1') is driven. **Figure 6-8** shows the port structure in the output mode driving a '1' while **figure 6-9** illustrates the port structure in the output mode driving a '0'.

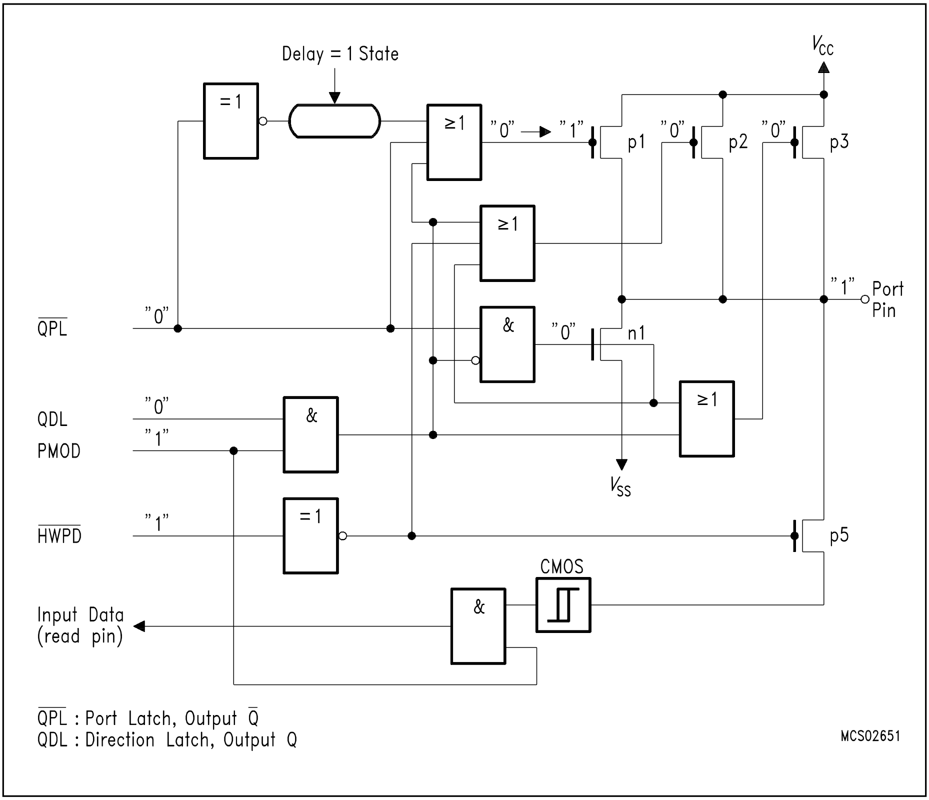


Figure 6-8
Bidirectional Port Structure - Output Mode - “1”

The FET n1 is switched off. FET p1 is activated for one state (dashed lines) 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 FETs p2, p3 are both active and are driving the '1' at the port pin.

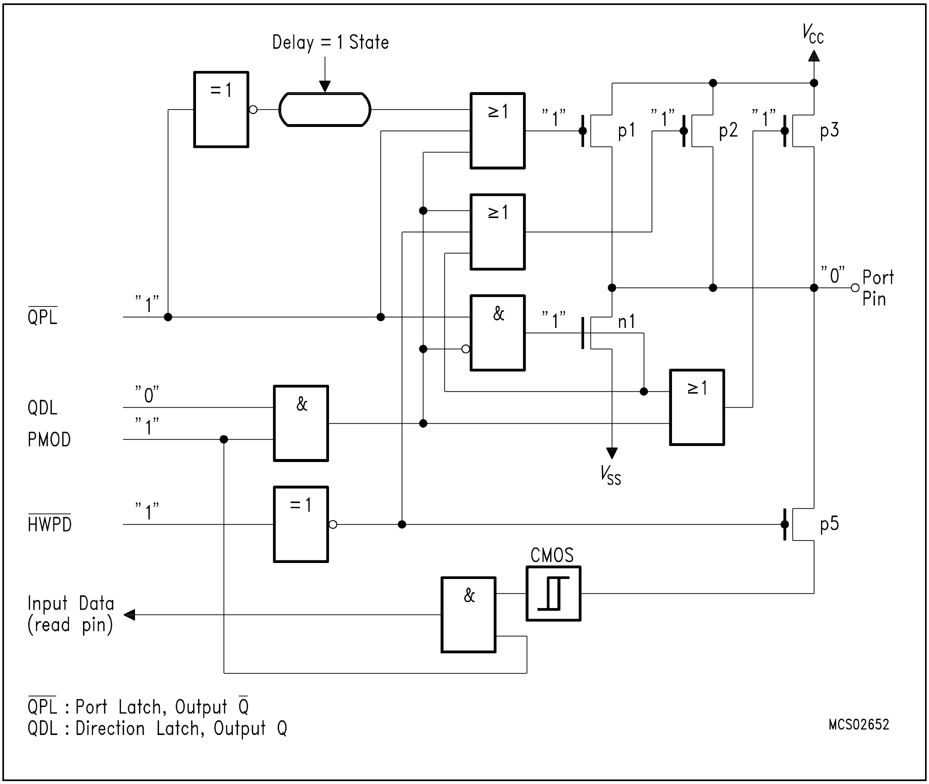


Figure 6-9
Bidirectional Port Structure - Output Mode - “0”

The FET n1 is switched on and is driving a '0' at the port pin. FETs p1, p2, p3 are switched off.

6.1.1.3.3 Hardware Power Down Mode

Figure 6-10 shows the port structure when the $\overline{\text{HWPD}}$-pin becomes active ($\overline{\text{HWPD}}$='0'). First of all the SFRs are written with their reset values. Therefore, the bit PMOD is cleared (PMOD=0), quasi-bidirectional port structure is enabled after leaving the hardware power down mode) and the port latch and direction latch contain a '1' ($\overline{\text{QPL}}$ = '0', QDL='1'). Then the hardware power down mode with ports in tri-state status is entered.

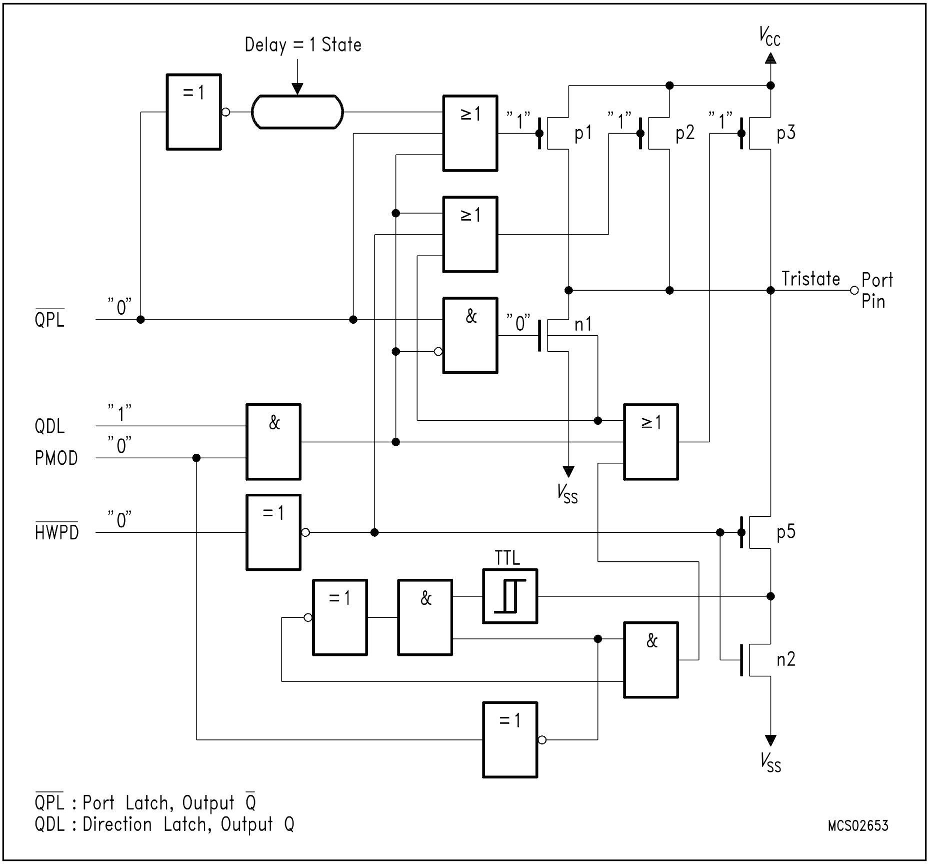


Figure 6-10
Bidirectional Port Structure - Hardware Power Down Mode

Due to $\overline{\text{HWPD}}$='0' the FET n2 becomes active and the FETs p2 and p5 are switched off. The FETs p1, p3 and n1 are switched off caused by the status of the port latch, direction latch and of PMOD (reset values).

6.1.2 Alternate Functions

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

Figure 6-11 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 pull-down FET is on and the port pin is stuck at 0. (This does not apply to ports 1.0 to 1.4 and ports 5.0 to 5.7 when operating in compare output mode). After reset all port latches contain ones (1).

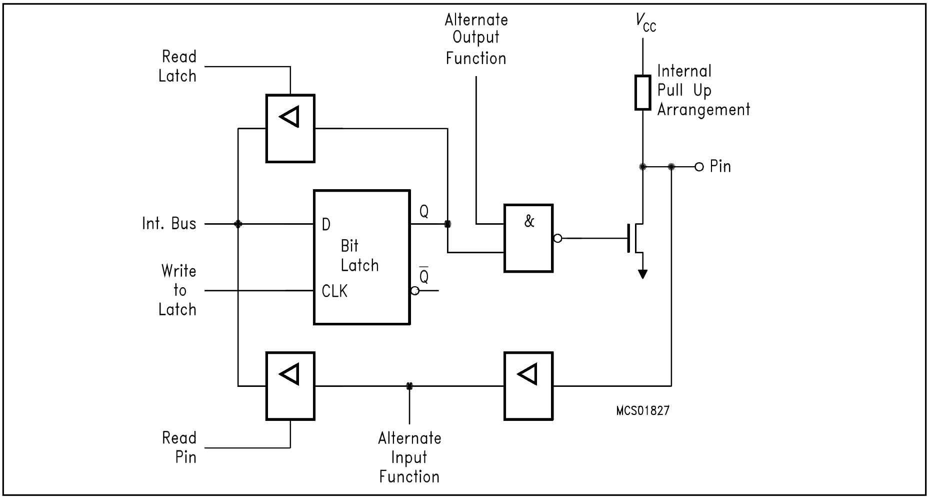


Figure 6-11
Circuitry of Ports 1, 3, 4, 5 and 6.0 through 6.2

Ports 6.3 through 6.7 have no alternate functions as described above. Therefore, the port circuitry can do without the switching capability between alternate function and normal I/O operation. This more simple circuitry is shown as basic port structure in **figure 6-3.**

Table 6-1 :
Alternate Functions of Port Pins

Port	Pin	Alternate Function
P1.0	INT3/CC0	External interrupt 3/capture 0/compare 0
P1.1	INT4/CC1	External interrupt 4/capture 1/compare 1
P1.2	INT5/CC2	External interrupt 5/capture 2/compare 2
P1.3	INT6/CC3	External interrupt 6/capture 3/compare 3
P1.4	INT2/CC4	External interrupt 2/capture 4/compare 4
P1.5	T2EX	Timer 2 ext. reload trigger input
P1.6	CLKOUT	System clock output
P1.7	T2	Timer 2 external count input

Table 6-1 :
Alternate Functions of Port Pins (cont'd)

Port	Pin	Alternate Function
P3.0	RXD0	Serial input channel 0
P3.1	TXD0	Serial output channel 0
P3.2	INT0	External interrupt 0
P3.3	INT1	External interrupt 1
P3.4	T0	Timer 0 external count input
P3.5	T1	Timer 1 external count input
P3.6	WR	External data memory write strobe
P3.7	RD	External data memory read strobe
P4.0	CM0	Compare output for the CM0 register
P4.1	CM1	Compare output for the CM1 register
P4.2	CM2	Compare output for the CM2 register
P4.3	CM3	Compare output for the CM3 register
P4.4	CM4	Compare output for the CM4 register
P4.5	CM5	Compare output for the CM5 register
P4.6	CM6	Compare output for the CM6 register
P4.7	CM7	Compare output for the CM7 register
P5.0	CCM0	Concurrent compare 0 output
P5.1	CCM1	Concurrent compare 1 output
P5.2	CCM2	Concurrent compare 2 output
P5.3	CCM3	Concurrent compare 3 output
P5.4	CCM4	Concurrent compare 4 output
P5.5	CCM5	Concurrent compare 5 output
P5.6	CCM6	Concurrent compare 6 output
P5.7	CCM7	Concurrent compare 7 output
P6.0	ADST	External A/D converter start
P6.1	RXD1	Serial input channel 1
P6.2	TXD1	Serial output channel 1
P6.3	$\overline{\text{WRF}}$	Write external FLASH
P9.0	CC10	Compare output/capture input for CC10 register
P9.1	CC11	Compare output/capture input for CC11 register
P9.2	CC12	Compare output/capture input for CC12 register
P9.3	CC13	Compare output/capture input for CC13 register
P9.4	CC14	Compare output/capture input for CC14 register
P9.5	CC15	Compare output/capture input for CC15 register
P9.6	CC16	Compare output/capture input for CC16 register
P9.7	CC17	Compare output/capture input for CC17 register

6.1.3 Port Handling

6.1.3.1 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 6-12** 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 requirements 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.

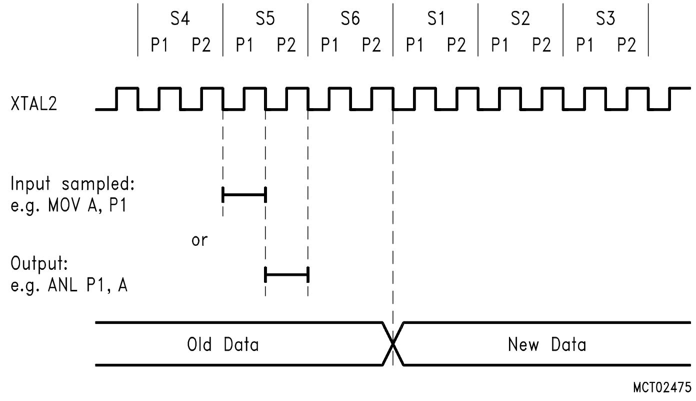


Figure 6-12
Port Timing

6.1.3.2 Port Loading and Interfacing

The output buffers of ports 1 to 6 and 9 can drive TTL inputs directly. The maximum port load which still guarantees correct logic output levels can be looked up in the C509-L DC characteristics in chapter 11 or in the data sheet. 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 6 and 9 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 C509-L DC characteristics specify these currents). Port 0 has floating inputs when used for digital input.

6.1.3.3 Read-Modify-Write Feature of Ports 1 to 6 and 9

Some port-reading instructions read the latch and others read the pin. The instructions reading the latch rather than the pin read a value, possibly change it, and then rewrite it to the latch. These are called “read-modify-write”- instructions, which are listed in **table 6-2**. If the destination is a port or a port pin, these instructions read the latch rather than the pin. Note that all other instructions which can be used to read a port, exclusively read the port pin. In any case, reading from latch or pin, resp., is performed by reading the SFRs P0, P1, P2 and P3; for example, “MOV A, P3” reads the value from port 3 pins, while “ANL P3, #0AAH” reads from the latch, modifies the value and writes it back to the latch.
It is not obvious that the last three instructions in **table 6-2** are read-modify-write instructions, but they are. The reason is that they read the port byte, all 8 bits, modify the addressed bit, then write the complete byte back to the latch.

Table 6-2 : “Read-Modify-Write”- Instructions

Instruction	Function
ANL	Logic AND; e.g. ANL P1, A
ORL	Logic OR; e.g. ORL P2, A
XRL	Logic exclusive OR; e.g. XRL P3, A
JBC	Jump if bit is set and clear bit; e.g. JBC P1.1, LABEL
CPL	Complement bit; e.g. CPL P3.0
INC	Increment byte; e.g. INC P1
DEC	Decrement byte; e.g. DEC P1
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 transistor (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 the latch. However, reading the latch rater than the pin will return the correct value of “1”.

6.2 Timer/Counter 0 and 1

The C509-L has a number of general purpose 16-bit timer/counters: timer 0, timer 1, timer 2 and the compare timer (timer 2 and the compare timer are discussed separately in **section 6.3** “Compare/Capture Unit”). Timer/counter 0 and 1 are fully compatible with timer/counters 0 and 1 of the SAB 8051 and can be used in the same operating modes.

Timer/counter 0 and 1 which are discussed in this section can be configured to operate either as timers or event counters:

- In “timer” function, the register is incremented at a maximum every machine cycle. Thus one can think of it as counting machine cycles. Since a machine cycle consists of 6 oscillator periods, the count rate is 1/6 of the oscillator frequency.
- In “counter” function, the register is incremented in response to a 1-to-0 transition (falling edge) at its corresponding external input pin, T0 or T1 (alternate functions of P3.4 and P3.5, resp.). In this function the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes two machine cycles (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.

In addition to the “timer” and “counter” selection, timer/counters 0 and 1 have four operating modes from which to select.

6.2.1 Time/Counter 0 / 1 Registers

The Timer 0/1 unit of the C509-L is controlled by totally 8 special function registers: TCON, TMOD, PRSC, IEN0, TL0, TH0, TL1, and TH1.

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. The interrupt enable control bits are located in the SFR IEN0.

The upper four bits of the special function register TCON are the run and overflow flags for timer 0 and 1.

Special Function Register TCON (Address. 88$_H$) **Reset Value : 00$_H$**
Special Function Register IEN0 (Address. A8$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	8F$_H$	8E$_H$	8D$_H$	8C$_H$	8B$_H$	8A$_H$	89$_H$	88$_H$	
88$_H$	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	TCON
	AF$_H$	AE$_H$	AD$_H$	AC$_H$	AB$_H$	AA$_H$	A9$_H$	A8$_H$	
A8$_H$	EAL	WDT	ET2	ES0	ET1	EX1	ET0	EX0	IEN0

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

Bit	Function
TF1	Timer 1 overflow flag Set by hardware on timer/counter 1 overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	Timer 1 run control bit. Set/cleared by software to turn timer/counter 1 on/off.
TF0	Timer 0 overflow flag Set by hardware on timer/counter 0 overflow. Cleared by hardware when processor vectors to interrupt routine.
TR0	Timer 0 run control bit. Set/cleared by software to turn timer/counter 0 on/off.
ET1	Timer 1 overflow interrupt enable. If ET1 = 0, the timer 1 interrupt is disabled.
ET0	Timer 0 overflow interrupt enable. If ET0 = 0, the timer 0 interrupt is disabled.

Special function register TMOD is used for mode select, gating, and counter/timer select purposes. The lower 4 bits control timer 0 and the upper 4 bits control timer 1.

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

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
89$_H$	GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0	TMOD
	Timer 1 Control				Timer 0 Control				

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>GATE</td><td>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.</td></tr>
<tr><td>C/T</td><td>Counter or timer select bit
Set for counter operation (input from “Tx” input pin).
Cleared for timer operation (input from internal system clock).</td></tr>
<tr><td>M1
M0</td><td>Operating mode select bits
<table>
<tr><th>M1</th><th>M0</th><th>Operating mode</th></tr>
<tr><td>0</td><td>0</td><td>8-bit timer/counter</td></tr>
<tr><td>0</td><td>1</td><td>16-bit timer/counter</td></tr>
<tr><td>1</td><td>0</td><td>8-bit auto-reload timer/counter</td></tr>
<tr><td>1</td><td>1</td><td>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</td></tr>
</table></td></tr>
</table>

The 4 lower bits of special function register PRSC control the timer 0/1 input clock prescaler operation.

Special Function Register PRSC (Address B4$_H$) **Reset Value : 11010101$_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
B4$_H$	WDTP	S0P	T2P1	T2P0	T1P1	T1P0	T0P1	T0P0	PRSC
					Timer 1 Control		Timer 0 Control		

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

Bit	Function
T1P1 T1P0	Prescaler select bits for timer 1
	T1P1 | T1P0 | Divider Ratio: 0 | 0 | 1 : 1 0 | 1 | 1 : 2 (reset value) 1 | 0 | 1 : 4 1 | 1 | 1 : 8
T0P1 T0P0	Prescaler select bits for timer 0
	T0P1 | T0P0 | Divider Ratio: 0 | 0 | 1 : 1 0 | 1 | 1 : 2 (reset value) 1 | 0 | 1 : 4 1 | 1 | 1 : 8

T1P1	T1P0	Divider Ratio
0	0	1 : 1
0	1	1 : 2 (reset value)
1	0	1 : 4
1	1	1 : 8

T0P1	T0P0	Divider Ratio
0	0	1 : 1
0	1	1 : 2 (reset value)
1	0	1 : 4
1	1	1 : 8

The 4 registers TL0, TH0, TL1, and TH1 are the count/reload registers of the timer 0 and 1:

Special Function Registers TL0/TH0 (Addresses $8A_H$/$8C_H$) Reset Value : 00_H
Special Function Registers TL1/TH1 (Addresses $8B_H$/$8D_H$) Reset Value : 00_H

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	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
TL0.7 - 0	Timer 0 low byte The TL0 is the 8-bit low byte count register of timer 0.
TH0.7 - 0	Timer 0 high byte The TH0 is the 8-bit high byte count/reload register of timer 0.
TL1.7 - 0	Timer 1 low byte The TL1 is the 8-bit low byte count register of timer 1.
TH1.7 - 0	Timer 1 high byte The TH1 is the 8-bit high byte count/reload register of timer 1.

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

6.2.2 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 6-13** 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 (see section 8 for details about the interrupt structure). The counted input is enabled to the timer when TR0 = 1 and either GATE = 0 or INT0 = 1 (setting GATE = 1 allows the timer to be controlled by external input 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 T0P1, T0P0, TR0, TF0, TH0, TL0, T0 and INT0 with the corresponding timer 1 signals in **figure 6-13**.

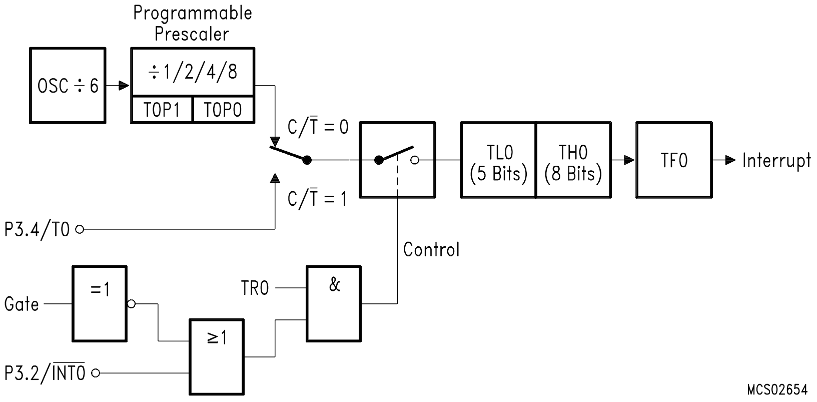


Figure 6-13
Timer/Counter 0/1, Mode 0: 13 Bit Timer/Counter

6.2.3 Mode 1

Mode 1 is the same as mode 0, except that the timer register runs with all 16 bits. Mode 1 is shown in **figure 6-14**.

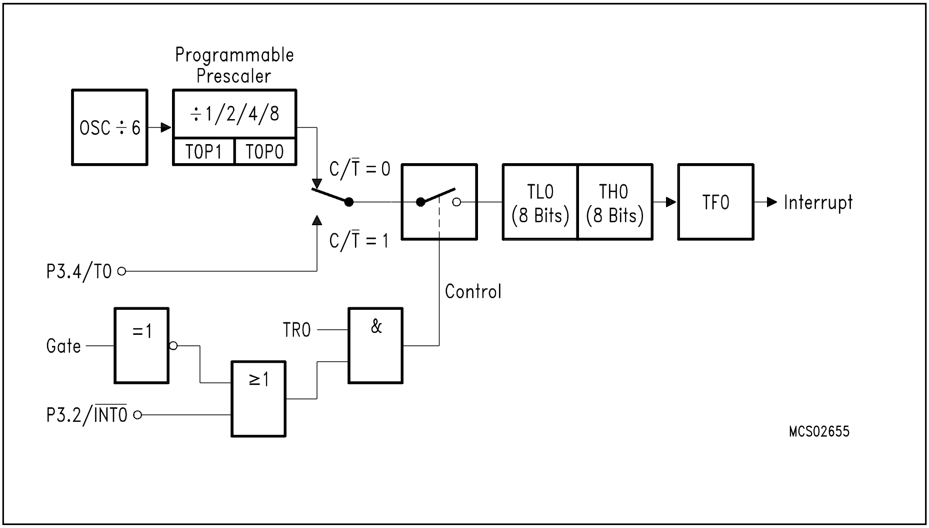


Figure 6-14
Timer/Counter 0/1, Mode 1: 16-Bit Timer/Counter

6.2.4 Mode 2

Mode 2 configures the timer register as an 8-bit counter (TL0) with automatic reload, as shown in **figure 6-15**. 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.

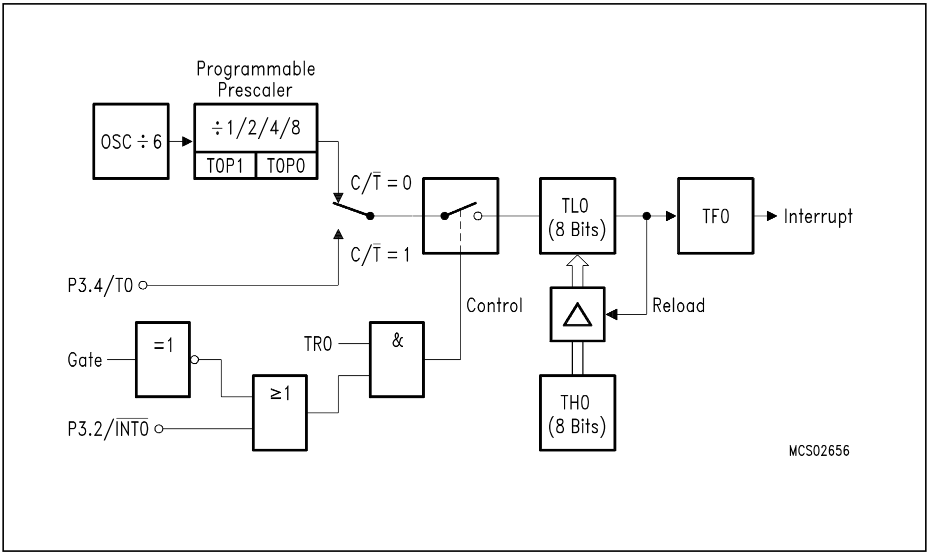


Figure 6-15
Timer/Counter 0/1, Mode 2: 8-Bit Timer/Counter with Auto-Reload

6.2.5 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 separate counters. The logic for mode 3 on timer 0 is shown in **figure 6-16**. TL0 uses the timer 0 control bits: C/T, GATE, TR0, INT0, and TF0. TH0 is locked into a timer function 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.

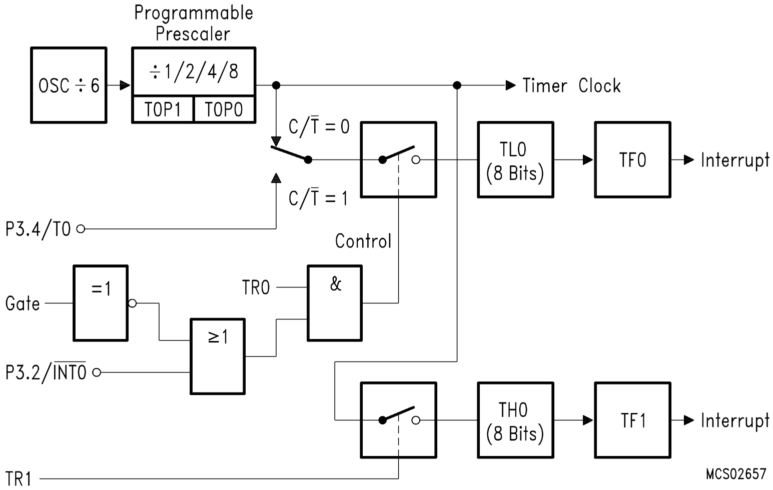


Figure 6-16
Timer/Counter 0, Mode 3: Two 8-Bit Timer/Counter

6.3 The Compare/Capture Unit (CCU)

The compare/capture unit is one of the C509-L's most powerful peripheral units for use in all kinds of digital signal generation and event capturing like pulse generation, pulse width modulation, pulse width measuring etc.

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

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

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

- Timer 2 with f_{OSC}/6 input clock, 4-bit prescaler, 16-bit reload, counter/gated timer mode and overflow interrupt request.
- Compare timers with f_{OSC} input clock, 4-bit prescaler, 16-bit reload and overflow interrupt request.
- Compare/(reload/)capture register array consisting of four different kinds of registers:
  one 16-bit compare/reload/capture register,
  three 16-bit compare/capture registers,
  one 16-bit compare/capture register with additional "concurrent compare" feature,
  eight 16-bit compare registers with timer-overflow controlled loading.

In summary, the register array may control up to 29 output lines and can request up to 11 independent interrupts.

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

The block diagram in **figure 6-17** shows the general configuration of the CCU. All CC1 to CC4 registers and the CRC register are exclusively assigned to timer 2. Each of the eight compare registers CM0 through CM7 can either be assigned to timer 2 or to the faster compare timer, e.g. to provide up to 8 PWM output channels. The assignment of the CMx registers - which can be done individually for every single register - is combined with an automatic selection of one of the two possible compare modes. The compare/capture registers CC10 to CC17 and the reload register CT1REL are assigned to compare timer 1 and are mapped to the corresponding registers of the compare timer.

Port 4, port 5, port 9 and five lines of port 1 have alternate functions dedicated to the CCU. These functions are listed in **table 6-3**. Normally each register controls one dedicated output line at the ports. Register CC4 is an exception as it can manipulate up to nine output lines (one at port 1.4 and the other eight at port 5) concurrently. This function is referenced as "concurrent compare".

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

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

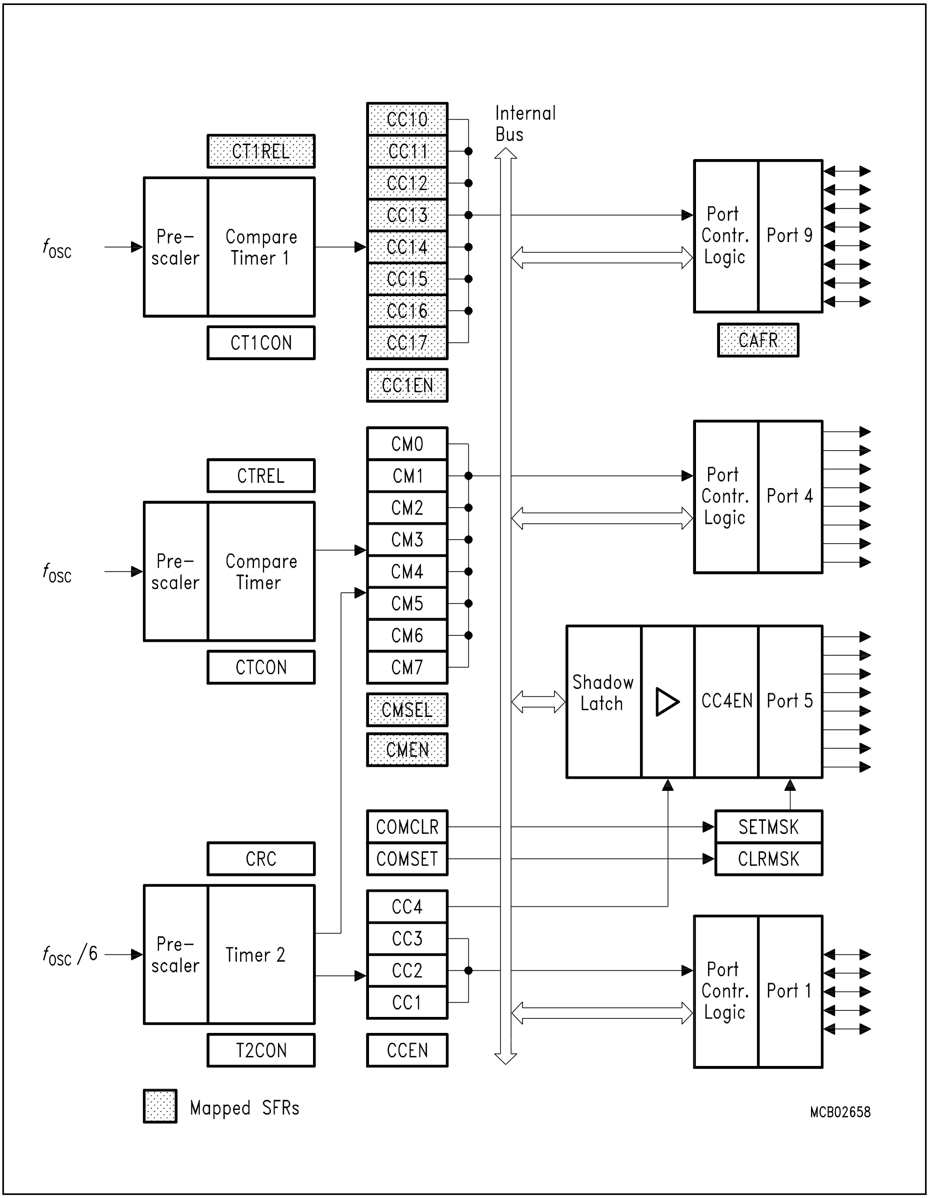


Figure 6-17
Block Diagram of the CCU

Table 6-3
Alternate Port Functions of the CCU

Pin Symbol	Pin No.	Alternate Function
P1.0/$\overline{\text{INT3}}$/CC0	9	Compare output/capture input for CRC register
P1.1/INT4/CC1	8	Compare output/capture input for CC1 register
P1.2/INT5/CC2	7	Compare output/capture input for CC2 register
P1.3/INT6/CC3	6	Compare output/capture input for CC3 register
P1.4/$\overline{\text{INT2}}$/CC4	1	Compare output/capture input for CC4 register
P1.5/T2EX	100	Timer 2 external reload trigger input
P1.7/T2	98	Timer 2 external count input
P4.0/CM0	64	Compare output for the CM0 register
P4.1/CM1	65	Compare output for the CM1 register
P4.2/CM2	66	Compare output for the CM2 register
P4.3/CM3	68	Compare output for the CM3 register
P4.4/CM4	69	Compare output for the CM4 register
P4.5/CM5	70	Compare output for the CM5 register
P4.6/CM6	71	Compare output for the CM6 register
P4.7/CM7	72	Compare output for the CM7 register
P5.0/CCM0	44	Concurrent compare 0 output
P5.1/CCM1	43	Concurrent compare 1 output
P5.2/CCM2	42	Concurrent compare 2 output
P5.3/CCM3	41	Concurrent compare 3 output
P5.4/CCM4	40	Concurrent compare 4 output
P5.5/CCM5	39	Concurrent compare 5 output
P5.6/CCM6	38	Concurrent compare 6 output
P5.7/CCM7	37	Concurrent compare 7 output
P9.0/CC10	74	Compare output/capture input for CC10 register
P9.1/CC11	75	Compare output/capture input for CC11 register
P9.2/CC12	76	Compare output/capture input for CC12 register
P9.3/CC13	77	Compare output/capture input for CC13 register
P9.4/CC14	5	Compare output/capture input for CC14 register
P9.5/CC15	4	Compare output/capture input for CC15 register
P9.6/CC16	3	Compare output/capture input for CC16 register
P9.7/CC17	2	Compare output/capture input for CC17 register

Table 6-4
Special Function Register of the CCU

Symbol	Description	Address with	
		RMAP=0	RMAP=1
CCEN	Compare/Capture Enable Register	$C1_H$	–
CC4EN	Compare/Capture 4 Enable Register	$C9_H$	–
CCH1	Compare/Capture Register 1, High Byte	$C3_H$	–
CCH2	Compare/Capture Register 2, High Byte	$C5_H$	–
CCH3	Compare/Capture Register 3, High Byte	$C7_H$	–
CCH4	Compare/Capture Register 4, High Byte	CF_H	–
CCL1	Compare/Capture Register 1, Low Byte	$C2_H$	–
CCL2	Compare/Capture Register 2, Low Byte	$C4_H$	–
CCL3	Compare/Capture Register 3, Low Byte	$C6_H$	–
CCL4	Compare/Capture Register 4, Low Byte	CE_H	–
CMEN	Compare Enable Register	$F6_H$	–
CMH0	Compare Register 0, High Byte	$D3_H$	–
CMH1	Compare Register 1, High Byte	$D5_H$	–
CMH2	Compare Register 2, High Byte	$D7_H$	–
CMH3	Compare Register 3, High Byte	$E3_H$	–
CMH4	Compare Register 4, High Byte	$E5_H$	–
CMH5	Compare Register 5, High Byte	$E7_H$	–
CMH6	Compare Register 6, High Byte	$F3_H$	–
CMH7	Compare Register 7, High Byte	$F5_H$	–
CML0	Compare Register 0, Low Byte	$D2_H$	–
CML1	Compare Register 1, Low Byte	$D4_H$	–
CML2	Compare Register 2, Low Byte	$D6_H$	–
CML3	Compare Register 3, Low Byte	$E2_H$	–
CML4	Compare Register 4, Low Byte	$E4_H$	–
CML5	Compare Register 5, Low Byte	$E6_H$	–
CML6	Compare Register 6, Low Byte	$F2_H$	–
CML7	Compare Register 7, Low Byte	$F4_H$	–
CC1EN	Compare/Capture Enable Register	–	$F6_H$
CC1H0	Compare/Capture 1 Register 0, High Byte	–	$D3_H$
CC1H1	Compare/Capture 1 Register 1, High Byte	–	$D5_H$
CC1H2	Compare/Capture 1 Register 2, High Byte	–	$D7_H$
CC1H3	Compare/Capture 1 Register 3, High Byte	–	$E3_H$
CC1H4	Compare/Capture 1 Register 4, High Byte	–	$E5_H$
CC1H5	Compare/Capture 1 Register 5, High Byte	–	$E7_H$
CC1H6	Compare/Capture 1 Register 6, High Byte	–	$F3_H$
CC1H7	Compare/Capture 1 Register 7, High Byte	–	$F5_H$
CC1L0	Compare/Capture 1 Register 0, Low Byte	–	$D2_H$
CC1L1	Compare/Capture 1 Register 1, Low Byte	–	$D4_H$
CC1L2	Compare/Capture 1 Register 2, Low Byte	–	$D6_H$
CC1L3	Compare/Capture 1 Register 3, Low Byte	–	$E2_H$
CC1L4	Compare/Capture 1 Register 4, Low Byte	–	$E4_H$
CC1L5	Compare/Capture 1 Register 5, Low Byte	–	$E6_H$
CC1L6	Compare/Capture 1 Register 6, Low Byte	–	$F2_H$
CC1L7	Compare/Capture 1 Register 7, Low Byte	–	$F4_H$

Table 6-4
Special Function Register of the CCU

Symbol	Description	Address with	
		RMAP=0	RMAP=1
CMSEL	Compare Input Select	$F7_H$	–
CAFR	Capture 1, Falling/Rising Edge Register	–	$F7_H$
CRCH	Comp./Rel./Capt. Reg. High Byte	CB_H	–
CRCL	Comp./Rel./Capt. Reg. Low Byte	CA_H	–
COMSETL	Compare Set Register, Low Byte	$A1_H$	–
COMSETH	Compare Set Register, High Byte	$A2_H$	–
COMCLRL	Compare Clear Register, Low Byte	$A3_H$	–
COMCLRH	Compare Clear Register, High Byte	$A4_H$	–
SETMSK	Compare Set Mask Register	$A5_H$	–
CLRMSK	Compare Clear Mask Register	$A6_H$	–
CTCON	Compare Timer Control Register	$E1_H$	–
CTRELH	Compare Timer Rel. Reg., High Byte	DF_H	–
CTRELL	Compare Timer Rel. Reg., Low Byte	DE_H	–
CT1RELH	Compare Timer 1 Rel. Reg., High Byte	–	DF_H
CT1RELL	Compare Timer 1 Rel. Reg., Low Byte	–	DE_H
TH2	Timer 2, High Byte	CD_H	–
TL2	Timer 2, Low Byte	CC_H	–
T2CON	Timer 2 Control Register	$C8_H$	–
CT1CON	Compare Timer 1 Control Register	BC_H	–
PRSC	Prescaler Control Register	$B4_H$	–

The compare/capture registers and reload registers related to compare timer 1 are mapped to the registers of the compare timer. This means, that they have the same register address. These compare timer 1 related registers are selected when bit RMAP, which located in the SFR SYSCON, is set during a register access. If RMAP=0, the compare timer related register are accessed. The rightmost 2 columns of **table 6-4** show the SFR address of each CCU register with the state of bit RMAP when they are accessed (RMAP description see also chapter 3.5).

6.3.1 Timer 2 Operation

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

6.3.1.1 Timer 2 Registers

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

Special Function Register T2CON	**(Address C8$_H$)**	**Reset Value : 00$_H$**
Special Function Register PRSC	**(Address B4$_H$)**	**Reset Value : 11010101$_B$**
Special Function Register CTCON	**(Address E1$_H$)**	**Reset Value : 01000000$_B$**
Special Function Register IEN0	**(Address A8$_H$)**	**Reset Value : 00$_H$**
Special Function Register IEN1	**(Address B8$_H$)**	**Reset Value : 00$_H$**
Special Function Register IRCON0	**(Address C0$_H$)**	**Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	CF$_H$	CE$_H$	CD$_H$	CC$_H$	CB$_H$	CA$_H$	C9$_H$	C8$_H$	
C8$_H$	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T1I0	T2CON

	7	6	5	4	3	2	1	0	
B4$_H$	WDTP	S0P	T2P1	T2P0	T1P1	T1P0	T0P1	T0P0	PRSC

	7	6	5	4	3	2	1	0	
E1$_H$	T2PS1	CTP	ICR	ICS	CTF	CLK2	CLK1	CLK0	CTCON

	AF$_H$	AE$_H$	AD$_H$	AC$_H$	AB$_H$	AA$_H$	A9$_H$	A8$_H$	
A8$_H$	EAL	WDT	ET2	ES0	ET1	EX1	ET0	EX0	IEN0

	BF$_H$	BE$_H$	BD$_H$	BC$_H$	BB$_H$	BA$_H$	B9$_H$	B8$_H$	
B8$_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	IEN1

	C7$_H$	C6$_H$	C5$_H$	C4$_H$	C3$_H$	C2$_H$	C1$_H$	C0$_H$	
C0$_H$	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC	IRCON0

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

The bits T2PS, T2PS1, T2P1, and T2P0 are prescaler select bits. T2R1 and T2R0 select the reload mode. T2I1 and T2I0 select the clock input source of timer 2. Bits ET2, EXEN2, EXF2, and TF2 are interrupt control bits/flags. Detailed bit definition is given in the table below.

<table>
<tr><th>Bit</th><th>Symbol</th></tr>
<tr><td>T2PS
T2P1
T2P0
T2PS1</td><td>Timer 2 prescaler select bits
Based on $f_{OSC}/6$ these bits define the prescaler divider ratio of the timer 2 input clock according the following table.
<table>
<tr><th>T2P1</th><th>T2P0</th><th>T2PS1</th><th>Timer 2 Input Clock T2PS=0</th><th>Timer 2 Input Clock T2PS=1</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$f_{OSC} \div 6$</td><td>$f_{OSC} \div 12$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{OSC} \div 24$</td><td>$f_{OSC} \div 48$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{OSC} \div 12$</td><td>$f_{OSC} \div 24$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{OSC} \div 48$</td><td>$f_{OSC} \div 96$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{OSC} \div 24$</td><td>$f_{OSC} \div 48$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$f_{OSC} \div 96$</td><td>$f_{OSC} \div 192$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$f_{OSC} \div 48$</td><td>$f_{OSC} \div 96$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$f_{OSC} \div 192$</td><td>$f_{OSC} \div 384$</td></tr>
</table></td></tr>
<tr><td>T2R1
T2R0</td><td>Timer 2 reload mode selection
<table>
<tr><th>T2R1</th><th>T2R0</th><th>Reload Mode</th></tr>
<tr><td>0</td><td>X</td><td>Reload disabled</td></tr>
<tr><td>1</td><td>0</td><td>Mode 0: auto-reload upon timer 2 overflow (TF2)</td></tr>
<tr><td>1</td><td>1</td><td>Mode 1: reload upon falling edge at pin P1.5 / T2EX</td></tr>
</table></td></tr>
<tr><td>T2I1
T2I0</td><td>Timer 2 input selection
<table>
<tr><th>T2I1</th><th>T2I0</th><th>Input Mode</th></tr>
<tr><td>0</td><td>0</td><td>No input selected : timer 2 stops</td></tr>
<tr><td>0</td><td>1</td><td>Timer function : input frequency see table above</td></tr>
<tr><td>1</td><td>0</td><td>Counter function : external input controlled by pin P1.7 / T2</td></tr>
<tr><td>1</td><td>1</td><td>Gated timer functiol : input controlled by pin T2/P1.7</td></tr>
</table></td></tr>
<tr><td>ET2</td><td>Timer 2 Interrupt Enable.
If ET2 = 0, the timer 2 interrupt is disabled.</td></tr>
</table>

Bit	Symbol
EXEN2	Timer 2 external reload interrupt enable If EXEN2 = 0, the timer 2 external reload interrupt is disabled. The external reload function is not affected by EXEN2.
EXF2	Timer 2 external reload flag Set when a reload is caused by a negative transition on pin T2EX while EXEN2 = 1. If ET2 in IEN0 is set (timer 2 interrupt enabled), EXF2 = 1 will cause an interrupt. Can be used as an additional external interrupt when the reload function is not used. EXF2 must be cleared by software.
TF2	Timer 2 overflow flag Set by a timer 2 overflow and must be cleared by software. If the timer 2 interrupt is enabled, TF2 = 1 will cause an interrupt.

Special Function Registers TL2/TH2 (Addresses CC_H/CD_H) Reset Value : 00_H
Special Function Registers CRCL/CRCH (Addresses CA_H/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	.0	TL2
CD_H	.7	.6	.5	.4	.3	.2	.1	.0	TH2
CA_H	.7	.6	.5	.4	.3	.2	.1	.0	CRCL
CB_H	.7	.6	.5	.4	.3	.2	.1	.0	CRCH

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

6.3.1.2 Timer 2 Operating Modes

Figure 6-18 shows a functional block diagram of the timer 2 unit.

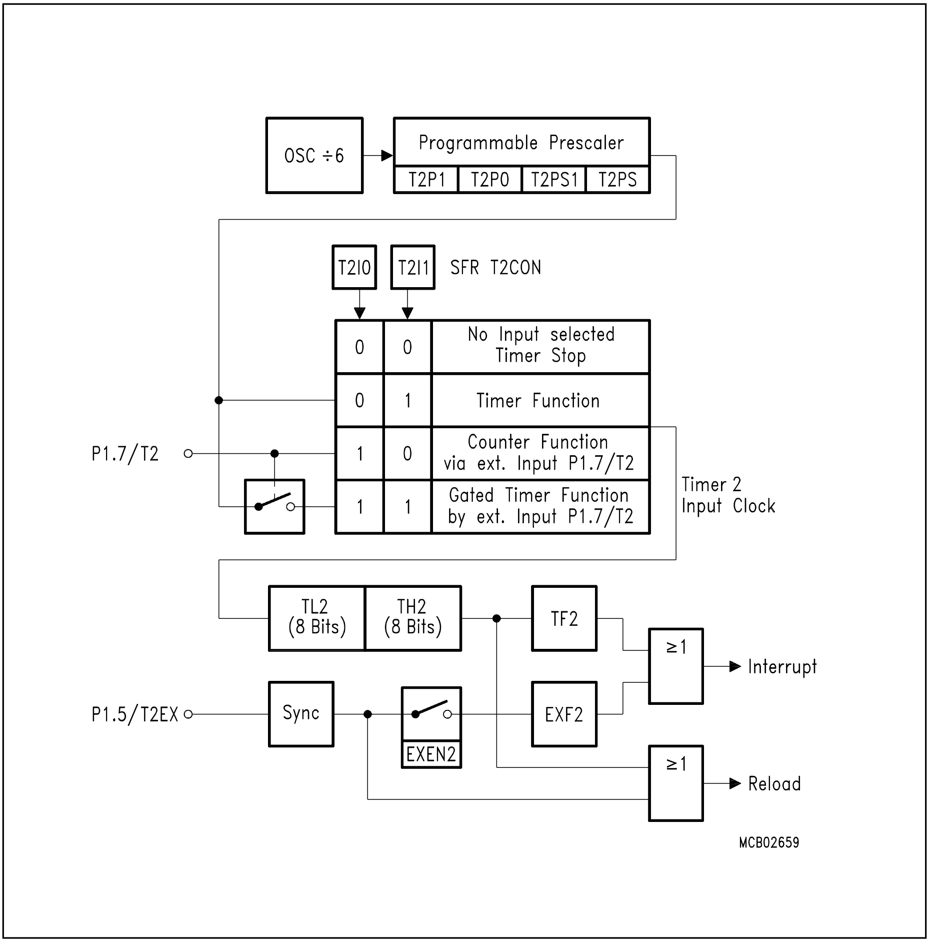


Figure 6-18
Block Diagram of Timer 2

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 to 1/384 of the oscillator frequency. Thus, the 16-bit timer register (consisting of TH2 and TL2) is incremented at maximum in every machine cycle. The prescaler is selected by the bits T2P1, T2P0, T2PS1, and T2PS (definition see previous page).

6.3.1.2.1 Gated Timer Mode

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

6.3.1.2.2 Event Counter Mode

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

Note:
The prescaler must be turned off for proper counter operation of timer 2 (T2P1=T2P0= T2PS1=T2PS=0).

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

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

6.3.1.2.3 Reload of Timer 2

The reload mode for timer 2 (see **figure 6-19**) is selected by bits T2R0 and T2R1 in SFR T2CON. Two reload modes are selectable:

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

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

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

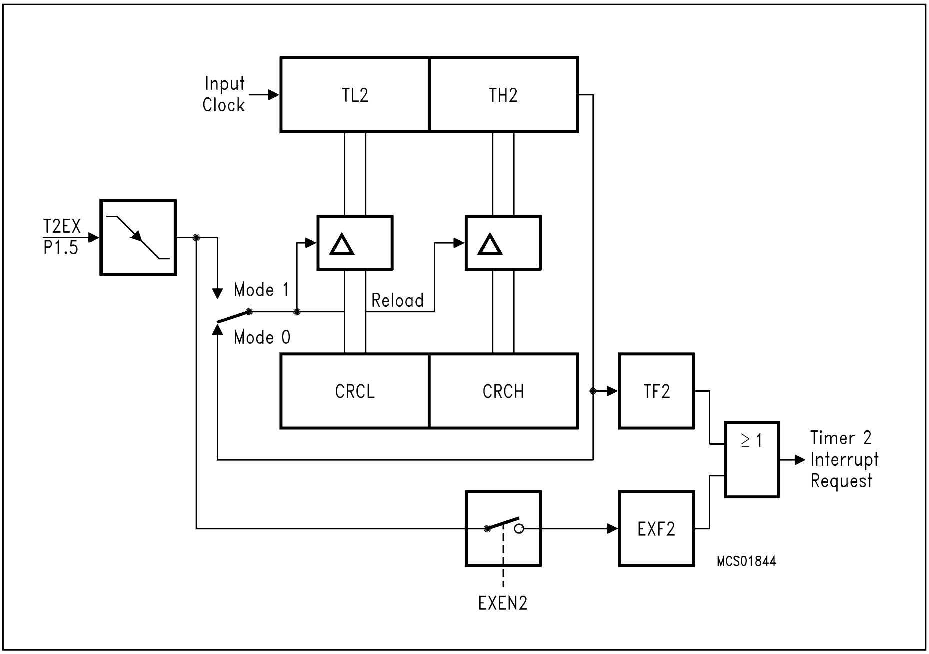


Figure 6-19
Timer 2 in Reload Mode

6.3.2 Operation of the Compare Timers

These two timers - the fourth and fifth timer in the C509-L - are implemented to function as a fast 16-bit time base for the compare registers CM0 to CM7 and CC10 to CC17. The compare timers are combined with the CMx/CC1x registers and can be used for high-speed output purposes or as a fast 16-bit pulse-width modulation unit.

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

6.3.2.1 Compare Timer and Compare Timer 1 Registers

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

Special Function Register CTCON (Address $E1_H$) Reset Value : 01000000_B
Special Function Register CT1CON (Address BC_H) Reset Value : $X1XX0000_B$

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$E1_H$	T2PS1	CTP	ICR	ICS	CTF	CLK2	CLK1	CLK0	CTCON
	7	6	5	4	3	2	1	0	
BC_H	–	CT1P	–	–	CT1F	CLK12	CLK11	CLK10	CT1CON

The shaded bits are not used for compare timer control.

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

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>CTP
CLK2
CLK1
CLK0</td><td>Compare timer input clock selection.
Based on f_{OSC} these bits define the prescaler divider ratio of the compare timer input clock according to the following table
<table>
<tr><th>CLK2</th><th>CLK1</th><th>CLK0</th><th>Compare Timer Input Clock CTP = 0</th><th>Compare Timer Input Clock CTP = 1 (default after reset)</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{OSC}</td><td>$f_{\text{OSC}} \div 2$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{\text{OSC}} \div 2$</td><td>$f_{\text{OSC}} \div 4$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{\text{OSC}} \div 4$</td><td>$f_{\text{OSC}} \div 8$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{\text{OSC}} \div 8$</td><td>$f_{\text{OSC}} \div 16$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{\text{OSC}} \div 16$</td><td>$f_{\text{OSC}} \div 32$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$f_{\text{OSC}} \div 32$</td><td>$f_{\text{OSC}} \div 64$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$f_{\text{OSC}} \div 64$</td><td>$f_{\text{OSC}} \div 128$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$f_{\text{OSC}} \div 128$</td><td>$f_{\text{OSC}} \div 256$</td></tr>
</table></td></tr>
<tr><td>CT1P
CLK12
CLK11
CLK10</td><td>Compare timer 1 input clock selection.
Based on f_{OSC} these bits define the prescaler divider ratio of the compare timer 1 input clock according to the following table
<table>
<tr><th>CLK12</th><th>CLK11</th><th>CLK10</th><th>Compare Timer 1 Input Clock CT1P = 0</th><th>Compare Timer 1 Input Clock CT1P = 1</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{OSC}</td><td>$f_{\text{OSC}} \div 2$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{\text{OSC}} \div 2$</td><td>$f_{\text{OSC}} \div 4$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{\text{OSC}} \div 4$</td><td>$f_{\text{OSC}} \div 8$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{\text{OSC}} \div 8$</td><td>$f_{\text{OSC}} \div 16$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{\text{OSC}} \div 16$</td><td>$f_{\text{OSC}} \div 32$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$f_{\text{OSC}} \div 32$</td><td>$f_{\text{OSC}} \div 64$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$f_{\text{OSC}} \div 64$</td><td>$f_{\text{OSC}} \div 128$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$f_{\text{OSC}} \div 128$</td><td>$f_{\text{OSC}} \div 256$</td></tr>
</table></td></tr>
<tr><td>CT1F</td><td>Compare timer 1 overflow flag
CT1F is set when the compare timer 1 count rolls over from all ones to the reload value. When CT1F is set, a compare timer 1 interrupt can be generated (if enabled). CT1F is cleared by hardware when the compare timer 1 value is no more equal its reload value.</td></tr>
</table>

The reload register of the two compare timers are located at the same SFR addresses.

Special Function Register CTRELL (Address DE$_H$) Reset Value : 00$_H$
Special Function Register CTRELH (Address DF$_H$) Reset Value : 00$_H$
Special Function Register CT1RELL (Mapped Address DE$_H$) Reset Value : 00$_H$
Special Function Register CT1RELH (Mapped Address DF$_H$) Reset Value : 00$_H$

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
DE$_H$	.7	.6	.5	.4	.3	.2	.1	.0	CTRELL
DF$_H$	.7	.6	.5	.4	.3	.2	.1	.0	CTRELH
DE$_H$ RMAP=1	.7	.6	.5	.4	.3	.2	.1	.0	CT1RELL
DF$_H$ RMAP=1	.7	.6	.5	.4	.3	.2	.1	.0	CT1RELH

Bit	Function
CTRELL.7 - 0	Compare timer reload value low part The CTRELL register holds the lower 8 bits of the 16-bit reload value for the compare timer.
CTRELH.7 - 0	Compare timer reload value high part The CTRELH register holds the upper 8 bits of the 16-bit reload value for the compare timer.
CT1RELL.7 - 0	Compare timer 1 reload value low part The CT1RELL register holds the lower 8 bits of the 16-bit reload value for the compare timer 1.
CT1RELH.7 - 0	Compare timer 1 reload value high part The CT1RELH register holds the upper 8 bits of the 16-bit reload value for the compare timer 1.

6.3.2.2 Operating Modes of the Compare Timers

The compare timers receive its input clock from a programmable prescaler which provides nine input frequencies, ranging from f_{OSC} up to $f_{OSC}/256$. This configuration allows a very high flexibility concerning timer period length and input clock frequency. The prescaler ratios are selected each by four bits in the special function registers CTCON and CT1CON. **Figure 6-20** shows the block diagram of the compare timer and compare timer 1.

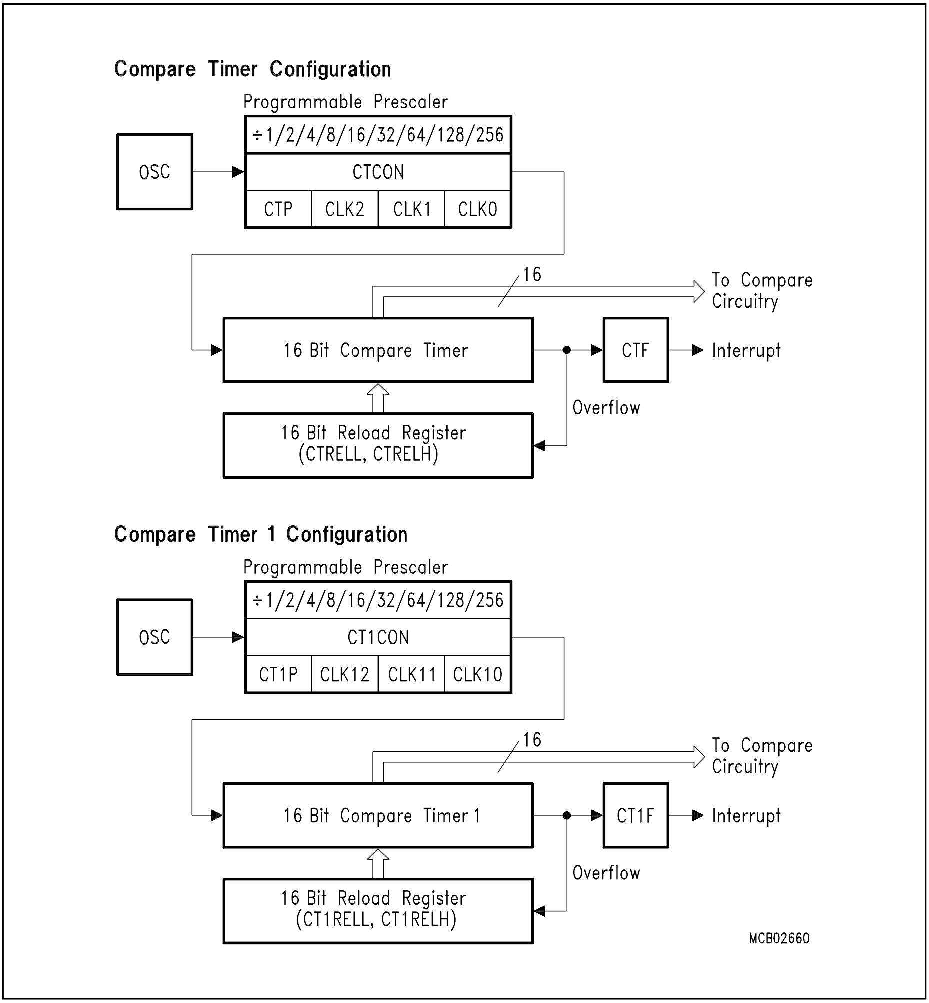


Figure 6-20
Compare Timer and Compare Timer 1 Block Diagram

The compare timers are, once started, free-running 16-bit timers, which on overflow are automatically reloaded by the contents of the 16-bit reload registers. These reload registers are CTRELL (compare timer reload register, low byte) and CTRELH (compare timer reload register, high byte) and the corresponding registers of the compare timer 1 (CTR1RELH, CT1RELH). An initial writing to the reload register CTRELL/CT1RELL starts the corresponding compare timer. If a compare timer is already running, a write to CTRELL/CT1RELL again triggers an instantly reload of the timer, in other words loads the timer in the cycle following the write instruction with the new count stored in the reload registers CTREL or CT1REL.

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

The compare timer have - as any other timer in the C509-L - their own interrupt request flags CTF and CT1F. These flags are located in the registers CTCON and CT1CON. CTF/CT1F are set when the timer count rolls over from all ones to the reload value. They are reset by hardware when the compare timer value is no more equal to the reload value.

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

6.3.3 Compare Functions of the CCU

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

The contents of a compare register can be regarded as ’time stamp’ at which a dedicated output reacts in a predefined way (either with a positive or negative transition). Variation of this ’time stamp’ somehow changes the wave of a rectangular output signal at a port pin. This may - as a variation of the duty cycle of a periodic signal - be used for pulse width modulation as well as for a continually controlled generation of any kind of square wave forms. In the case of the C509-L, two compare modes are implemented to cover a wide range of possible applications.

In the C509-L - thanks to the high number of 21 compare registers and two associated timers - several timer/compare register combinations are selectable. In some of these configurations one of the two compare modes may be freely selected, others, however, automatically establish a compare mode. In the following the two possible modes are generally discussed. This description will be referred to in later sections where the compare registers are described.

As already mentioned, there are only a few compare registers with their corresponding port circuitry which are able to serve both compare modes. In most cases the mode is automatically set depending on the timer which is used as time base or depending on the port which outputs the compare signal.

6.3.3.1 Compare Mode 0

In mode 0, upon matching the timer and compare register contents, the output signal changes from low to high. It goes back to a low level on timer overflow. As long as compare mode 0 is enabled, the appropriate output pin is controlled by the timer circuit only, and not by the user. Writing to the port will have no effect. **Figure 6-21** shows a functional diagram of a port circuit when used in compare mode 0. The port latch is directly controlled by the timer overflow and compare match signals. The input line from the internal bus and the write-to-latch line of the port latch are disconnected when compare mode 0 is enabled.

Compare mode 0 is ideal for generating pulse width modulated output signals, which in turn can be used for digital-to-analog conversion via a filter network or by the controlled device itself (e.g. the inductance of a DC or AC motor). Compare mode 0 may also be used for providing output clocks with initially defined period and duty cycle. This is the mode which needs the least CPU time. Once set up, the output goes on oscillating without any CPU intervention. **Figure 6-22** illustrates the function of compare mode 0.

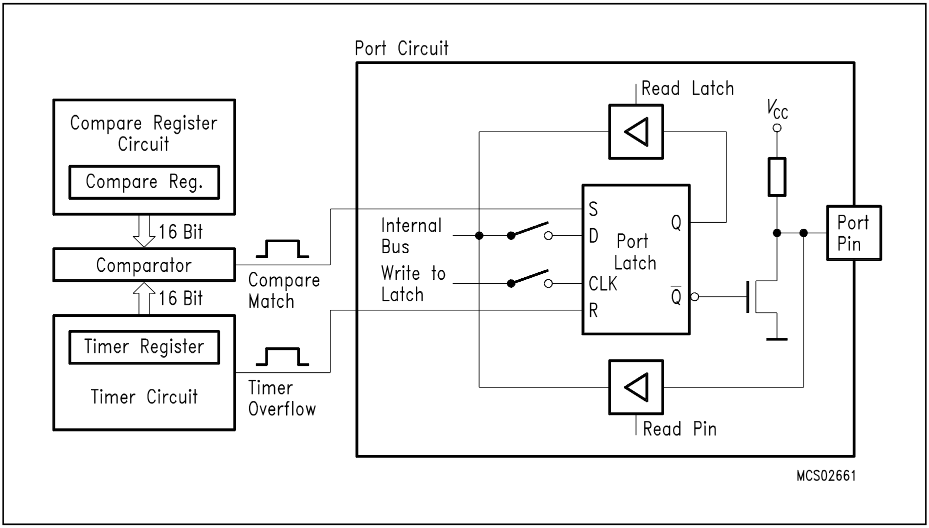


Figure 6-21
Port Latch in Compare Mode 0

Figure 6-22 shows a typical output signal waveform which is generated when compare mode 0 is selected.

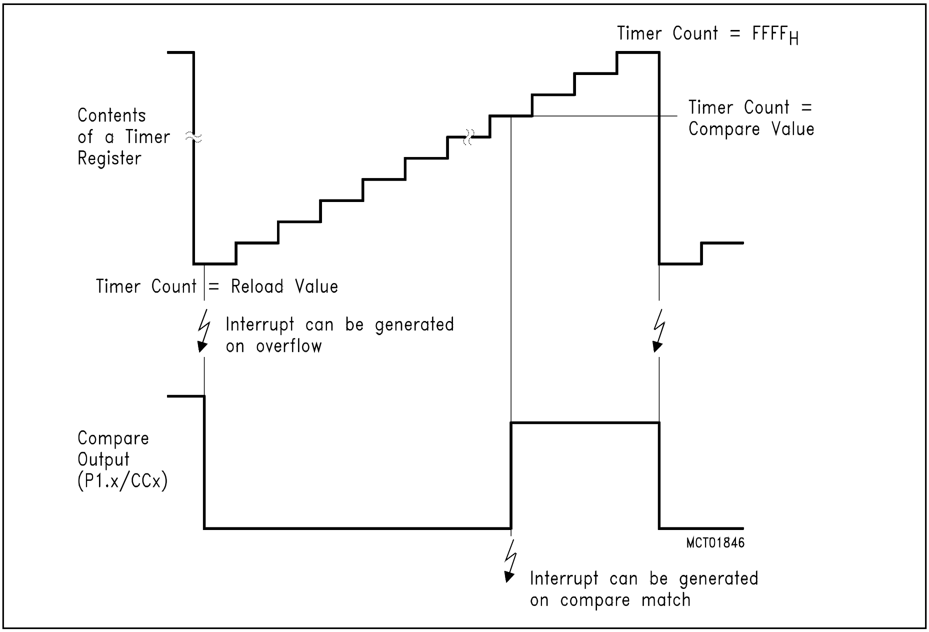

Timer Count = FFFF$_H$
Contents of a Timer Register
Timer Count = Compare Value
Timer Count = Reload Value
Interrupt can be generated on overflow
Compare Output (P1.x/CCx)
MCT01846
Interrupt can be generated on compare match

Figure 6-22
Output Waveform of Compare Mode 0

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

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

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

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

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

In the C509-L there are two different modulation ranges for the above mentioned two timer/compare register combinations. The difference is the location of the above spike within the timer period: at the end of a timer period or at the beginning plus the end of a timer period. Please refer to the description of the CCU relevant timer/register combination in section 6.3.4 for details.

6.3.3.2 Compare Mode 1

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

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

Figure 6-23 shows a functional diagram of a port circuit configuration in compare mode 1. In this mode the port circuit consists of two separate latches. One latch (which acts as a "shadow latch") can be written under software control, but its value will only be transferred to the port latch (and thus to the port pin) when a compare match occurs.

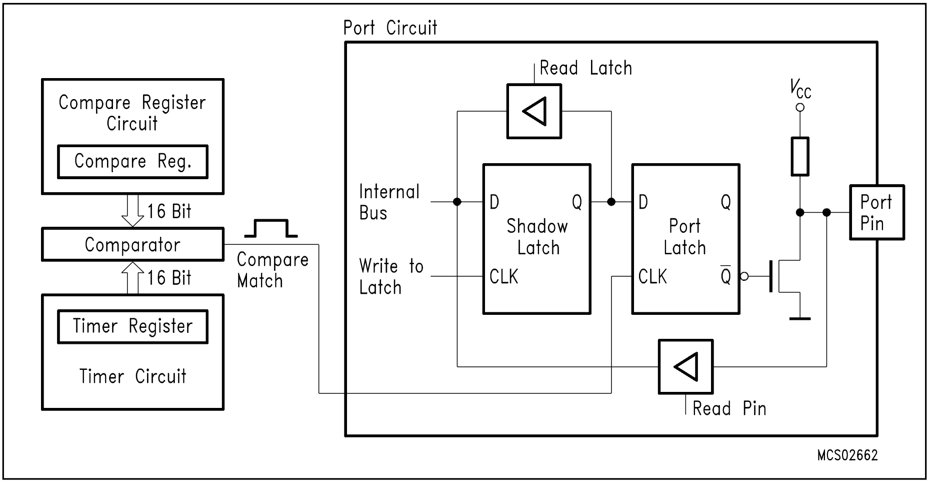


Figure 6-23
Compare Function of Compare Mode 1

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

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

6.3.3.3 Compare Mode 2

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

In compare mode 2 the concurrent compare output pins at port 5 are used as shown in **figure 6-24**.:

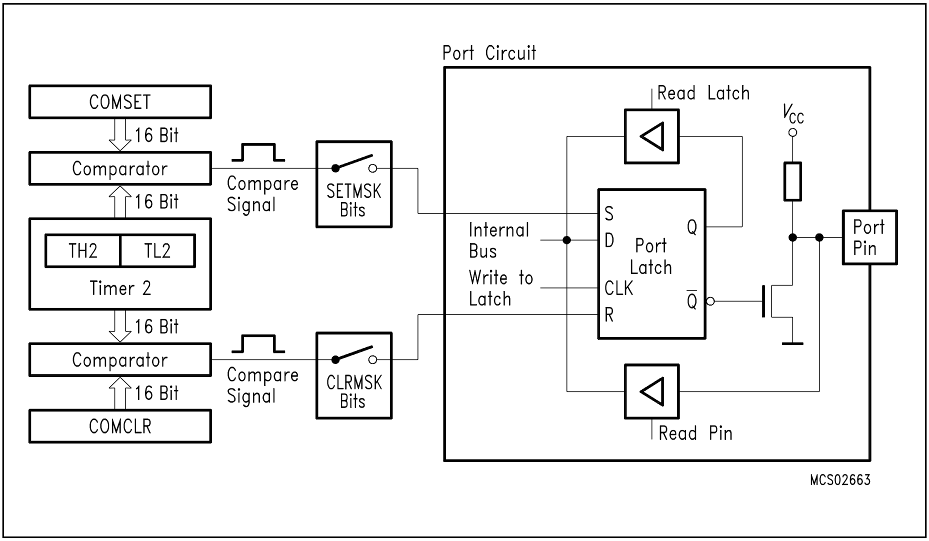


Figure 6-24
Compare Function of Compare Mode 2

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

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

6.3.4 Timer- and Compare-Register Configurations of the CCU

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

Table 6-5
CCU Configurations

Assigned Timer	Compare Register	Compare Output at	Possible Modes
Timer 2	CRCH/CRCL	P1.0/INT3/CC0	Compare mode 0, 1 + Reload
	CCH1/CCL1	P1.1/INT4/CC1	Compare mode 0, 1 / capture
	CCH2/CCL2	P1.2/INT5/CC2	Compare mode 0, 1 / capture
	CCH3/CCL3	P1.3/INT6/CC3	Compare mode 0, 1 / capture
	CCH4/CCL4	P1.4/INT2/CC4	Compare mode 0, 1 / capture (see 6.3.4.1 and 6.3.4.2)
	CCH4/CCL4	P1.4/INT2/CC4 P5.0/CCM0 to P5.7/CCM7	Compare mode 1 “Concurrent compare“ (see 6.3.4.3)
	CMH0/CML0 to CMH7/CML7	P4.0/CM0 to P4.7/CM7	Compare mode 0 (see 6.3.4.4 and 6.3.4.4.2)
	COMSET COMCLR	P5.0/CCM0 to P5.7/CCM7	Compare mode 2 (see 6.3.4.5)
Compare Timer	CMH0/CML0 to CMH7/CML7	P4.0/CM0 to P4.7/CM7	Compare mode 1 (see 6.3.4.4 and 6.3.4.4.1)
Compare Timer 1	CC1H0/CC1L0 to CC1H7/CC1L7	P5.0/CCM0 to P5.7/CCM7	Compare mode 0 / capture (see 6.3.4.6)

6.3.4.1 Timer 2 - Compare Function with Registers CRC, CC1 to CC3

The compare function of registers CRC and CC1 to CC3 is completely compatible with the corresponding function of the SAB 80C515. Registers CRC, CC1 to CC3 are permanently connected to timer 2. All four registers are multifunctional as they additionally provide a capture or a reload capability (CRC register only).

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

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

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

	MSB							LSB	
Bit No.	CF_H	CE_H	CD_H	CC_H	CB_H	CA_H	$C9_H$	$C8_H$	
$C8_H$	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T1I0	T2CON
	7	6	5	4	3	2	1	0	
$C1_H$	COCAH3	COCAL3	COCAH2	COCAL2	COCAH1	COCAL1	COCAH0	COCAL0	CCEN

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

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>T2CM</td><td>Compare mode control for CCR and CC1 to CC3 registers
When T2CM is cleared, compare mode 0 is selected.
When T2CM is set, compare mode 1 is selected.</td></tr>
<tr><td>COCAH3,
COCAL3</td><td>Compare/capture mode for the CC register 3
<table>
<tr><th>COCAH3</th><th>COCAL3</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>Compare/capture disabled</td></tr>
<tr><td>0</td><td>1</td><td>Capture on rising edge at pin P1.3/INT6/CC3</td></tr>
<tr><td>1</td><td>0</td><td>Compare enabled</td></tr>
<tr><td>1</td><td>1</td><td>Capture on write operation into register CCL3</td></tr>
</table></td></tr>
</table>

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

Figure 6-25 and **6-26** show the general timer/compare register/port latch configuration for registers CRC and CC1 to CC4 in compare mode 0 and compare mode 1. It also shows the interrupt capabilities. The compare interrupts of registers CRC and CC4 can be programmed to be either negative or positive transition activated. Compare interrupts for the CC1 to CC3 registers are always positive transition activated.

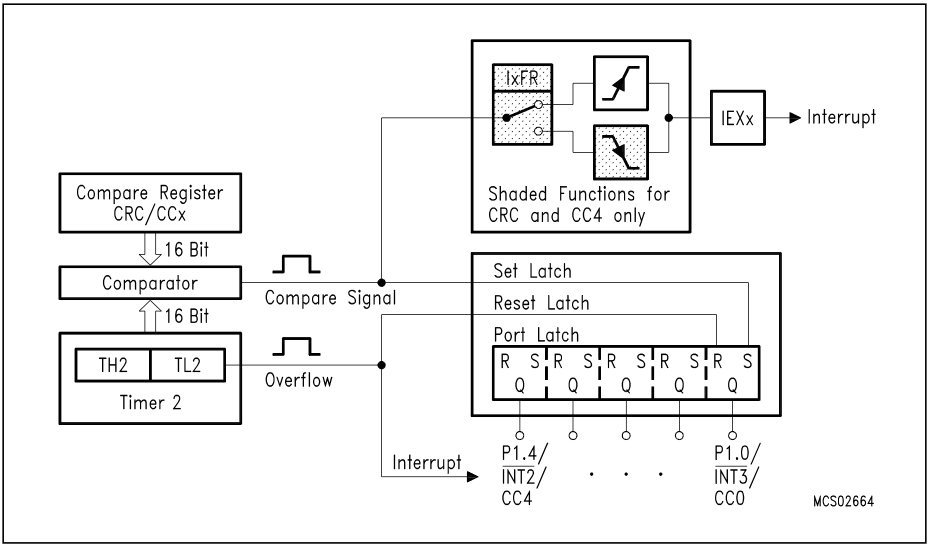


Figure 6-25
Timer 2 with Registers CRC and CC1 to CC4 in Compare Mode 0

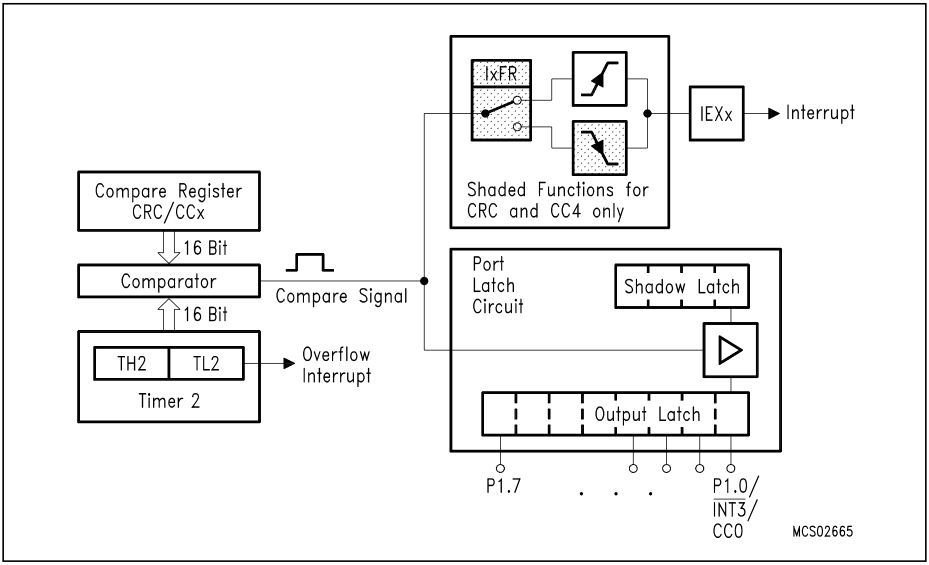


Figure 6-26
Timer 2 with Registers CRC and CC1 to CC4 in Compare Mode 1

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

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

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

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

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

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

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

Figures 6-27 and **6-28** show functional diagrams of the capture function of timer 2. **Figure 6-27** illustrates the operation of the CRC or CC4 register, while **figure 7-28** shows the operation of the compare/capture registers 1 to 3.

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

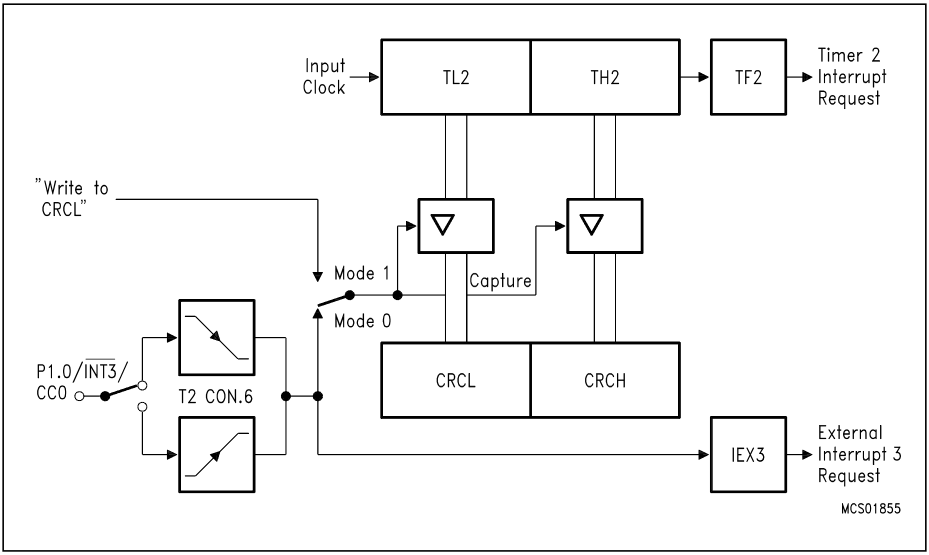


Figure 6-27
Capture with Registers CRC, CC4

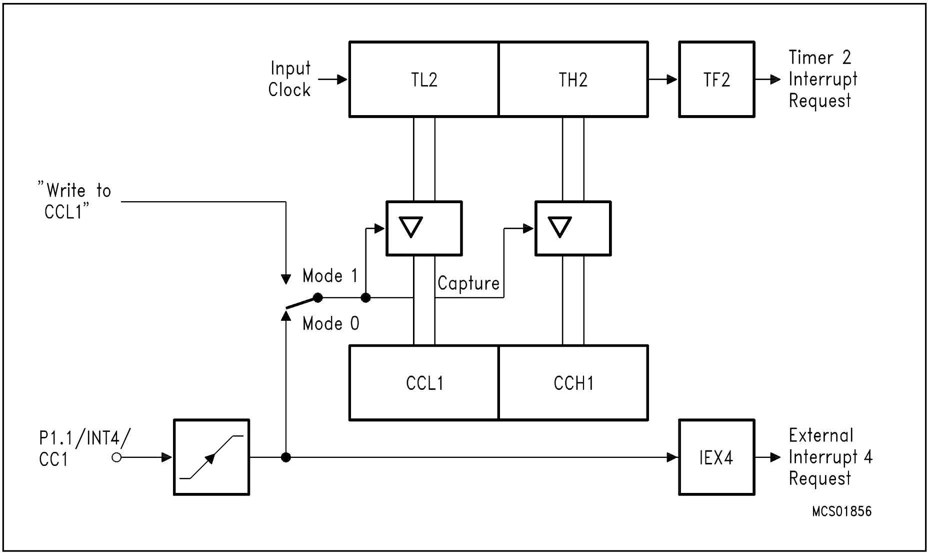


Figure 6-28
Capture with Registers CC1 to CC3

6.3.4.3 Compare Function of Register CC4; “Concurrent Compare”

Compare register CC4 is permanently assigned to timer 2. It has its own compare/capture enable register CC4EN. Register CC4 can be set to operate as any of the other CC registers (see also figures **7-29** and **7-30**). Its output pin is P1.4/CC4/INT2 and it has a dedicated compare mode select bit COMO located in register CC4EN.

In addition to the standard operation in compare mode 0 or 1, there is another feature called ‘concurrent compare’ which is just an application of compare mode 1 to more than one output pin. Concurrent compare means that the comparison of CC4 and timer 2 can manipulate up to nine port pins concurrently. A standard compare register in compare mode 1 normally transfers a preprogrammed signal level, which is stored in the shadow latch to a single output line. Register CC4, however, is able to put a 9-bit pattern to nine output lines. The nine output lines consist of one line at port 1 (P1.4), which is the standard output for register CC4, and additional eight lines at port 5 (see **figure 6-29**).

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

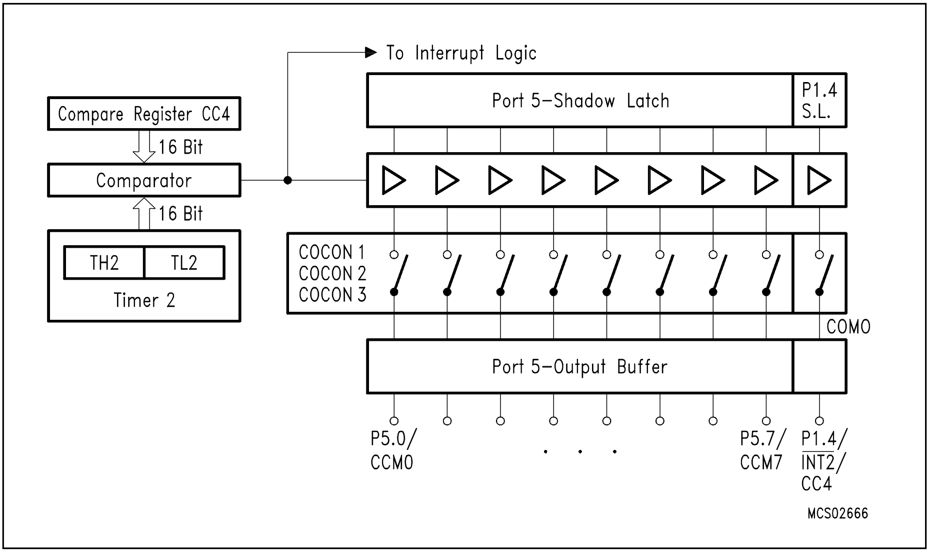


Figure 6-29
“Concurrent Compare” Function of Register CC4

Figure 6-30 gives an example of how to generate eight different rectangular wave forms at port 5 using a pattern table and a time schedule for these patterns. The patterns are moved into port 5 before the corresponding timer count is reached. The (future) timer count at which the pattern shall appear at the port must be loaded to register CC4. Thus the user can mask each port bit differently depending on whether he wants the output to be changed or not.

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

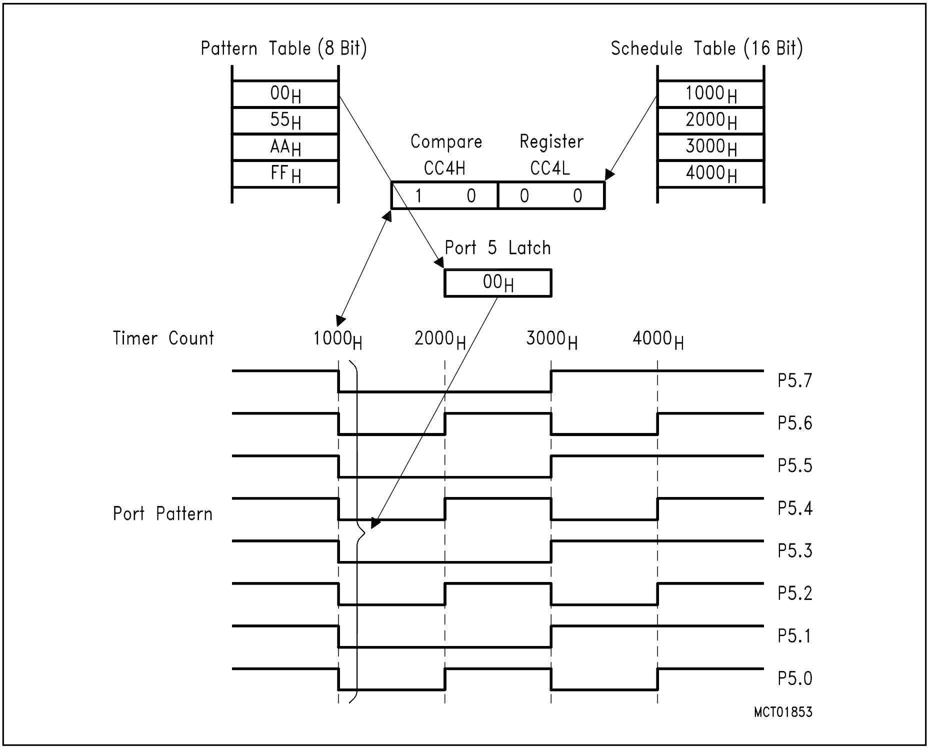


Figure 6-30
Example for a “Concurrent Compare” Waveform at Port 5

Special Function Register CC4EN (Address C9$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
C9$_H$	COCO EN1	COCO N2	COCO N1	COCO N0	COCO EN0	COCAH4	COCAL4	COMO	CC4EN

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>COCOEN1
COCOEN0</td><td>Selection of compare modes 1 and 2 at port 5
For details on mode selection with COCOEN1/COCOEN0 see <b>table 6-6</b>.</td></tr>
<tr><td>COCON2
COCON1
COCON0</td><td>Port 5 compare outputs selection
These bits select the number of compare outputs at port 5 according the following table.
<table>
<tr><th>COCON2</th><th>COCON1</th><th>COCON0</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>One additional output of CC4 at P5.0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Additional outputs of CC4 at P5.0 to P5.1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Additional outputs of CC4 at P5.0 to P5.2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Additional outputs of CC4 at P5.0 to P5.3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Additional outputs of CC4 at P5.0 to P5.4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Additional outputs of CC4 at P5.0 to P5.5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Additional outputs of CC4 at P5.0 to P5.6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Additional outputs of CC4 at P5.0 to P5.7</td></tr>
</table></td></tr>
<tr><td>COCAH4
COCAL4</td><td>Compare/capture mode selection for the CC register 4
For details on mode selection with COCAH4/COCAL4 see <b>table 6-6</b>.</td></tr>
<tr><td>COMO</td><td>CC4 compare mode select bit
When set, compare mode 1 is selected for CC4. COMO = 0 selects compare mode 0 for CC4. Setting of bit COCOEN0 automatically sets COMO.</td></tr>
</table>

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

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

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

6.3.4.4 Compare Function of Registers CM0 to CM7

The CCU of the C509-L contains another set of eight compare registers and an additional timer, the compare timer, and some control SFRs. These compare registers and the compare timer are mainly dedicated to PWM applications.

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

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

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

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

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

6.3.4.4.1 CMx Registers Assigned to the Compare Timer

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

The "Timer Overflow Controlled" (TOC) Loading

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

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

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

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

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

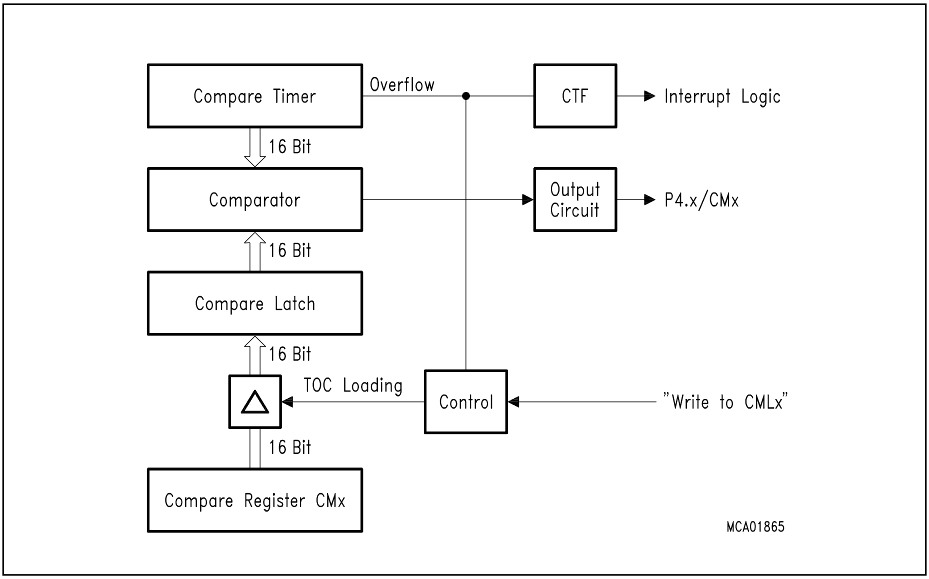


Figure 6-31
Compare Function of a CMx Register Assigned to the Compare Timer

Figure 6-31 shows a more detailed block diagram of a CMx register connected to the compare timer. It illustrates that the CPU can only access the special function register CMx; the actual compare latch is, however, loaded at timer overflow. The timer overflow signal also sets an interrupt request flag (CTF in register CTCON) which may be used to inform the CPU by an interrupt that a new timer cycle has started and that the compare values for the next cycle may be programmed from now on.

The activation of the TOC loading depends on a few conditions described in the following. A TOC loading is performed only if the CMLx register has been changed by the CPU. A write instruction to the low byte of the CMx register is used to enable the loading. The 8-bit architecture of the C509-L requires such a defined enable mechanism because 16-bit values are to be transferred in two portions (= two instructions).

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

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

Summary of the TOC loading capability :

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

Initializing the Compare Register/Compare Latch Circuit

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

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

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

6.3.4.4.2 CMx Registers Assigned to the Timer 2

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

Since there are no dedicated interrupts for the CMx compare outputs, again a buffered compare register structure is used to determine an exact 16-bit wide loading of the compare value: the compare value is transferred to the actual compare latches at a write-to-CMLx instruction (low byte of CMx). Thus, the CMx register is to be written in a fixed order, too: high byte first, low byte second. If the high byte may remain unchanged it is sufficient to load only the low byte. See **figure 6-32**, block diagram of a CMx register connected to timer 2.

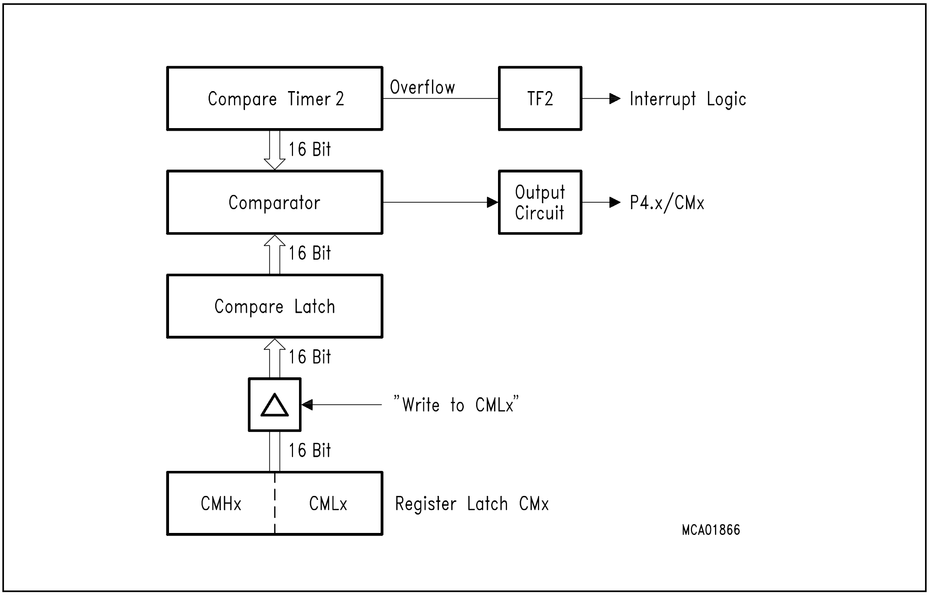


Figure 6-32
CMx-Register Assigned to Timer 2

6.3.4.5 Timer 2 Operating in Compare Mode 2

The compare mode 2 of the CCU can be used for the concurrent compare output function at port 5. In compare mode 2, the port 5 pins are no longer general purpose I/O pins or under control of the compare/capture register CC4, but under control of the compare registers COMSET and COMCLR. The details of compare mode 2 are already described in section 6.3.3.3. **Figure 6-33** shows the complete compare mode 2 configuration of the CCU and the port 5 pins.

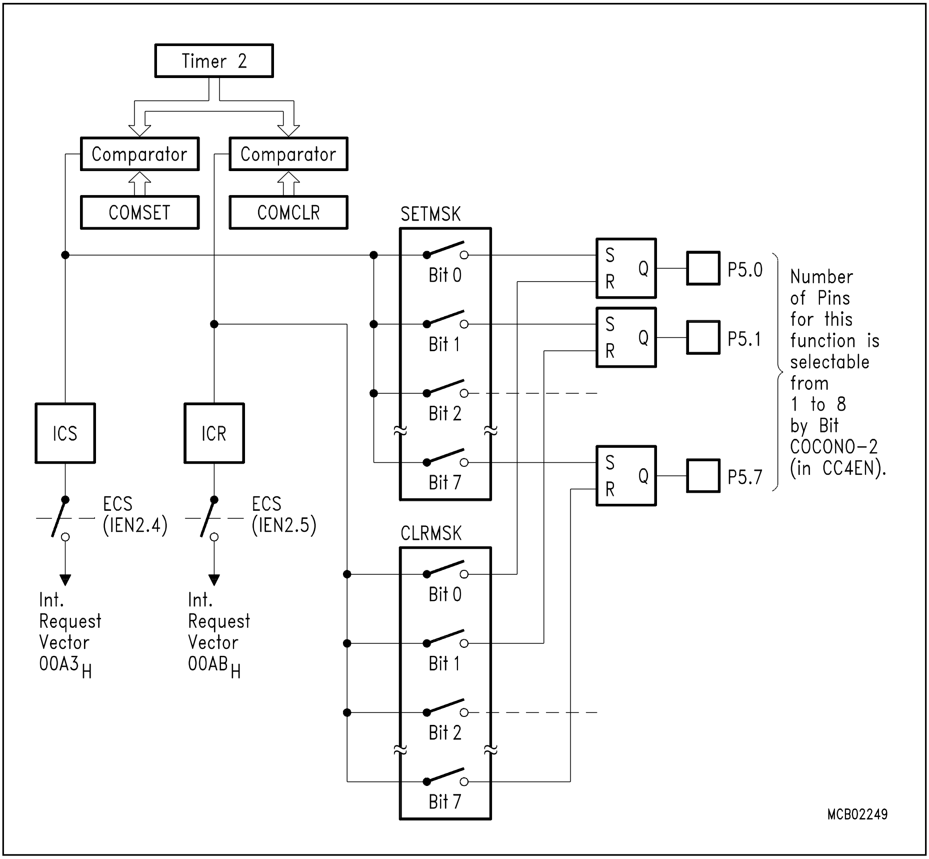


Figure 6-33
Compare Mode 2 (Port 5 only)

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

6.3.4.6 Compare / Capture Operation with Compare Timer 1

The C509-L Compare/Capture Unit (CCU) has a third timer - the compare timer 1 (CT1). This timer is assigned to port 9. The port 9 alternate functions are dedicated for the CT1 controlled compare/capture capability.

The compare/capture registers, the reload registers, and the control registers (except CT1CON) related to compare timer 1 are mapped to the registers of the compare timer. This means, that the CT1 registers have the same register address as the compare timer registers. The compare timer 1 related registers are selected when bit RMAP, which located in the SFR SYSCON, is set during a register access. If RMAP=0, the compare timer related register are accessed.

As an extract from **table 6-4**, **table 6-4** lists all CT1 registers which are mapped to the compare timer registers. Further details about the RMAP bit are given in chapter 3.5.

Table 6-7
Compare Timer / Compare Timer 1 Related Mapped Special Function Registers of the CCU

Address	Symbol	Description	Access by RMAP =
$D2_H$	CML0 CC1L0	Compare Register 0, Low Byte Compare/Capture 1 Register 0, Low Byte	0 1
$D3_H$	CMH0 CC1H0	Compare Register 0, High Byte Compare/Capture 1 Register 0, High Byte	0 1
$D4_H$	CML1 CC1L1	Compare Register 1, Low Byte Compare/Capture 1 Register 1, Low Byte	0 1
$D5_H$	CMH1 CC1H1	Compare Register 1, High Byte Compare/Capture 1 Register 1, High Byte	0 1
$D6_H$	CML2 CC1L2	Compare Register 2, Low Byte Compare/Capture 1 Register 2, Low Byte	0 1
$D7_H$	CMH2 CC1H2	Compare Register 2, High Byte Compare/Capture 1 Register 2, High Byte	0 1
$E2_H$	CML3 CC1L3	Compare Register 3, Low Byte Compare/Capture 1 Register 3, Low Byte	0 1
$E3_H$	CMH3 CC1H3	Compare Register 3, High Byte Compare/Capture 1 Register 3, High Byte	0 1
$E4_H$	CML4 CC1L4	Compare Register 4, Low Byte Compare/Capture 1 Register 4, Low Byte	0 1
$E5_H$	CMH4 CC1H4	Compare Register 4, High Byte Compare/Capture 1 Register 4, High Byte	0 1
$E6_H$	CML5 CC1L5	Compare Register 5, Low Byte Compare/Capture 1 Register 5, Low Byte	0 1
$E7_H$	CMH5 CC1H5	Compare Register 5, High Byte Compare/Capture 1 Register 5, High Byte	0 1
$F2_H$	CML6 CC1L6	Compare Register 6, Low Byte Compare/Capture 1 Register 6, Low Byte	0 1

Table 6-7
Compare Timer / Compare Timer 1 Related Mapped Special Function Registers of the CCU

Address	Symbol	Description	Access by RMAP =
$F3_H$	CMH6 CC1H6	Compare Register 6, High Byte Compare/Capture 1 Register 6, High Byte	0 1
$F4_H$	CML7 CC1L7	Compare Register 7, Low Byte Compare/Capture 1 Register 7, Low Byte	0 1
$F5_H$	CMH7 CC1H7	Compare Register 7, High Byte Compare/Capture 1 Register 7, High Byte	0 1
$F6_H$	CMEN CC1EN	Compare Enable Register Compare/Capture Enable Register	0 1
$F7_H$	CMSEL CAFR	Compare Input Select Capture 1, Falling/Rising Edge Register	0 1
DE_H	CTRELL CT1RELL	Compare Timer Reload Register, Low Byte Compare Timer 1 Reload Register, Low Byte	0 1
DF_H	CTRELH CT1RELH	Compare Timer Reload Register, High Byte Compare Timer 1 Reload Register, High Byte	0 1

The SFR CT1CON contains the control bits for the input clock selection and the compare timer 1 overflow flag CT1F. CT1CON and the reload registers (CT1RELL, CT1RELH) are described in detail in section 6.3.2.
The SFR CC1EN contains the enable bits for the compare/capture registers CC10 to CC17. When set, compare/capture function is enabled and connected to the port 9 pins. AFR CC1EN is a mapped register.and can only be accessed when bit RMAP in SFR SYSCON is set.

Special Function Register CC1EN (Mapped Address $F6_H$) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$F6_H$	.7	.6	.5	.4	.3	.2	.1	.0	CC1EN

Bit	Function
CC1EN.7 - 0	Compare timer 1 capture/compare register enable bits If CC1EN.x is set, the compare/capture function for P9.x is enabled and the port line is connected with the corresponding compare/capture register CC(10+x). If CC1EN.x is cleared, the port pin P9.x operates as standard digital I/O.

6.3.4.6.1 Compare Function of Registers CC10 to CC17

The compare function of the registers CC10 to CC17 is equivalent to the compare function of the CM1 to CM7 registers (when CM0-7 are controlled by the compare timer). The compare timer 1 configuration operates in compare mode 0. The description of the compare mode operation of the CMx registers which is given in detail in section 6.3.4.4.1, is also valid for the compare mode operation of the CC1x registers. Only the compare output port, the enable control, and the compare timer flag are different.

Figure 6-34 below shows the CC1x registers when they are used for compare function.

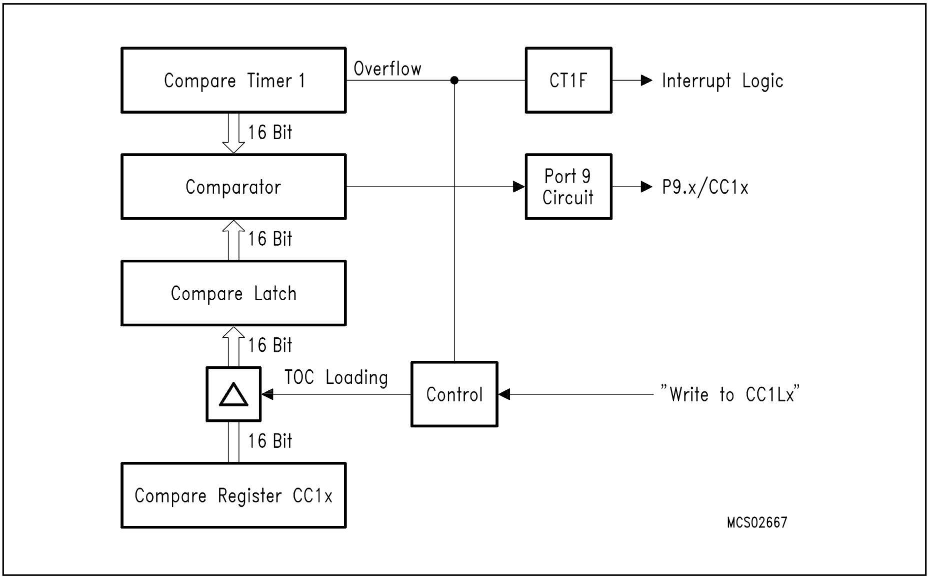


Figure 6-34
Compare Function of a CC1x Register Assigned to the Compare Timer 1

The compare function of the compare time 1 configuration is selected by programming the direction register of port 9 (DIR9) with the direction bit = 0. The direction register DIR9 is written to by a double instruction sequence (first instruction sets PDIR in SFR IP1, second instruction writes to direction register). The direction register has the same address as port 9.

6.3.4.6.2 Capture Function of Registers CC10 to CC17

The capture function is selected by programming the direction register related to port 9 (DIR9). With the direction bit of DIR9 = 1 the capture function at the corresponding port 9 pin is selected. Each of the eight compare/capture registers CC10 to CC17 can be used to latch the current 16-bit value of the compare timer 1 registers by a positive or a negative transition at the corresponding port 9 pin.

The bits located in the SFR CAFR are used to define selectively, whether a positive or a negative transition at a port 9 pin forces a capture event. CAFR is a mapped register.and can only be accessed when bit RMAP in SFR SYSCON is set.

Special Function Register CAFR (Mapped Address F7$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
F7$_H$	.7	.6	.5	.4	.3	.2	.1	.0	CAFR

Bit	Function
CAFR.7-0	Capture on falling / rising edge selection bits CAFR.x = 0 : Capture into CC(10+x) on falling edge at pin P9.x is selected (reset value) CAFR.x = 1 : Capture into CC(10+x) on rising edge at pin P9.x is selected.)

The level at the corresponding port 9 pin is sampled every state within a machine cycle. For example, with fosc = 16 MHz a state lasts for 62.5 ns. When the compare timer 1 input clock is selected to fosc, the resulting resolution is 125 ns. The rise/fall time of the external capture signal must be observed.

At a capture event with the CC10-CC17 register the corresponding interrupt request flag which is located in SFR IRCON2 is set

6.3.5 Modulation Range in Compare Mode 0

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

In a timer 2 / CCx register configuration in compare mode 0 this spike is divided into two halves: one at the beginning when the contents of the compare register is equal to the reload value of the timer; the other half when the compare register is equal to the maximum value of the timer register (here: FFFF_H). Please refer to **figure 6-35** where the maximum and minimum duty cycle of a compare output signal is illustrated. Timer 2 is incremented with the processor cycle ($f_{OSC}/6$), thus at 12-MHz operating frequency, these spikes are both approx. 250 ns long.

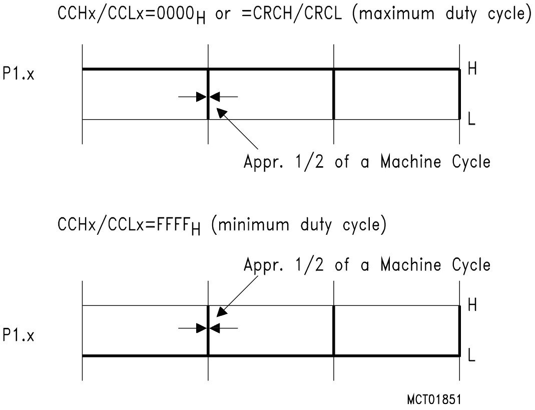

CCHx/CCLx=0000H or =CRCH/CRCL (maximum duty cycle)
P1.x
H
L
Appr. 1/2 of a Machine Cycle
CCHx/CCLx=FFFFH (minimum duty cycle)
Appr. 1/2 of a Machine Cycle
P1.x
H
L
MCT01851

Figure 6-35
Modulation Range of a PMW Signal Generated with a Timer 2 / CCx Register Combination in Compare Mode 0

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

Example:

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

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

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

In a compare timer/CMx register configuration, the compare output is set to a constant high level if the contents of the compare registers are equal to the reload register (CTREL). The compare output shows a high level for one timer clock period when a CMx register is set to $\mathrm{FFFF_H}$. Thus, the duty cycle can be varied from 0.xx% to 100% depending on the resolution selected. In **figure 6-36** the maximum and minimum duty cycle of a compare output signal is illustrated. One clock period of the compare timer is equal to one machine state (= 1 oscillator periods) if the prescaler is off. Thus, at 12 MHz system clock the spike is approx. 83.3 ns long.

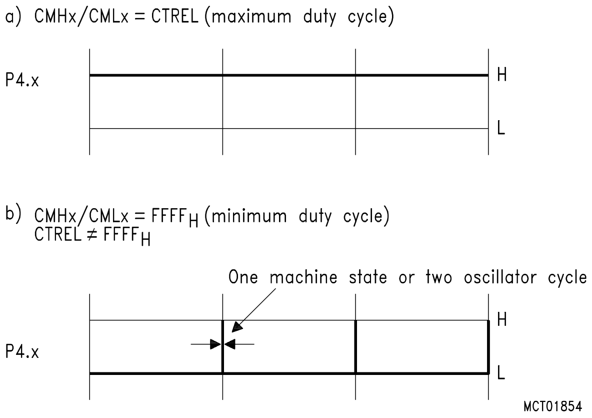

a) CMHx/CMLx = CTREL (maximum duty cycle)
P4.x
H
L
b) CMHx/CMLx = $\mathrm{FFFF_H}$ (minimum duty cycle)
CTREL ≠ $\mathrm{FFFF_H}$
One machine state or two oscillator cycle
P4.x
H
L
MCT01854

Figure 6-36
Modulation Range of a PWM Signal Generated with a Compare Timer/CMx Register Combination

6.3.6 Using Interrupts in Combination with the Compare Function

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

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

6.3.6.1 Some advantages in using compare interrupts

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

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

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

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

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

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

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

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

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

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

6.3.6.2 Interrupt Enable Bits of the Compare/Capture Unit

This section summarizes all CCU related interrupt enable control bits. The interrupt enable bits for the two compare timers and the compare match and capture interrupt capture are located in the three SFRs IEN2, IEN3, and EICC1. Note : EICC1 is a mapped SFR:

Special Function Register IEN2 (Address 9A$_H$) Reset Value : XX0000X0$_B$
Special Function Register IEN3 (Address BE$_H$) Reset Value : XXXX00XX$_B$
Special Function Register EICC1 (Mapped Address BF$_H$) Reset Value : FF$_H$

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
9A$_H$	–	–	ECR	ECS	ECT	ECMP	–	ES1	IEN2
	7	6	5	4	3	2	1	0	
BE$_H$	–	–	–	–	ECT1	ECC1	–	–	IEN3
	7	6	5	4	3	2	1	0	
BF$_H$	EICC17	EICC16	EICC15	EICC14	EICC13	EICC12	EICC11	EICC10	EICC1

The shaded bits are not used for CCU interrupt control.

Bit	Function
ECR	COMCLR register compare match interrupt enable If ECR = 0, the COMCLR compare match interrupt is disabled.
ECS	COMSET register compare match interrupt enable If ECS = 0, the COMSET compare match interrupt is disabled.
ECT	Enable compare timer interrupt If ECT = 0, the compare timer overflow interrupt is disabled.
ECMP	CM0-7 register compare match interrupt If ECMP = 0, the CM0-7 compare match interrupt is disabled.
ECT1	Compare timer 1 overflow interrupt enable If ECT1 = 0, the interrupt at compare timer 1 overflow is disabled.
ECC1	Compare timer 1, general capture/compare interrupt enable This bit enables the interrupt on an capture/compare event in the capture/compare registers CC10 - CC17. Additionally, the SFR EICC1 must be programmed to enable the interrupt request. With ECC1=0, the general capture/compare timer 1 interrupt is disabled.
EICC17 - EICC10	Compare timer 1, specific capture/compare interrupt enable This bit enables the interrupt on an capture/compare event in the capture/compare registers CC10 - CC17. When EICC1x=0, the interrupt request flag ICC1x has no effect on the interrupt unit. EICC17 refers to CC17 etc.

6.3.6.3 Interrupt Flags of the Compare/Capture Unit

This section summarizes the CCU related compare match interrupt flags. The two compare timer overflow interrupt flags (CT and CT1) are described in detail in section 6.3.2.1.

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

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

The compare timer 1 match interrupt occurs on a compare match of the CC10 to CC17 registers with the compare timer 1. There are 8 compare match interrupt flags available in SFR IRCON2 which are or-ed together for a single interrupt request. Thus, a compare match interrupt service routine has to check which compare match has requested the compare match interrupt. The ICC1x flags must be cleared by software.

Special Function Register IRCON1 (Address D1$_H$) **Reset Value : 00$_H$**
Special Function Register IRCON2 (Address BF$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
D1$_H$	ICMP7	ICMP6	ICMP5	ICMP4	ICMP3	ICMP2	ICMP1	ICMP0	IRCON1
	7	6	5	4	3	2	1	0	
BF$_H$	ICC17	ICC16	ICC15	ICC14	ICC13	ICC12	ICC11	ICC10	IRCON2

Bit	Function
ICMP7 - 0	Compare timer match with register CM7 - CM0 interrupt flags ICMPx is set by hardware when a compare match of the compare timer with the compare register CMx occurs but only if the compare function for CMx has been enabled. ICMPx must be cleared by software (CMSEL.x = 0 and CMEN.x = 1).
ICC17 - 10	Compare timer 1 match with register CC17 - CC10 interrupt flags ICC1x is set by hardware when a compare match of the compare timer 1 with the compare register CC1x occurs but only if the compare function for CC1x has been enabled. ICC1x must be cleared by software.

6.4 Arithmetic Unit

This on-chip arithmetic unit of the C509-L provides fast 32-bit division, 16-bit multiplication as well as shift and normalize features. All operations are unsigned integer operations.

The arithmetic unit (further on also called MDU for “Multiplication/Division Unit”) has been integrated to support the C500 core of the C509-L in real-time control applications. It can increase the execution speed of math-intensive software routines by factor 5 to 10.

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

Table 6-8
MDU Operation Characteristics

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

[1] 1 t_{CY} = 6 • CLP = 1 machine cycle = 375 ns at 16-MHz oscillator frequency
[2] The maximal shift speed is 6 shifts per machine cycle

6.4.1 MDU Register

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

Thus MD0 to MD5 are used twofold:

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

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

Table 6-9
MDU Registers

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

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

Special Function Register ARCON (Address EF$_H$) **Reset Value : 0XXXXXXX$_B$**

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

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

6.4.2 Operation of the MDU

The MDU can be regarded as a special coprocessor for multiplication, division and shift. Its operations can be divided into three phases (see **figure 6-37**):

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

During phase two, the MDU works on its own parallelly to the CPU. Execution times of the above table refer to this phase. Because of the fast operation and the determined execution time for C509-L's instructions, there is no need for a busy flag. The CPU may execute a determined number of instructions before the result is fetched. The result and the remainder of an operation may also be stored in the MDx registers for later use.

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

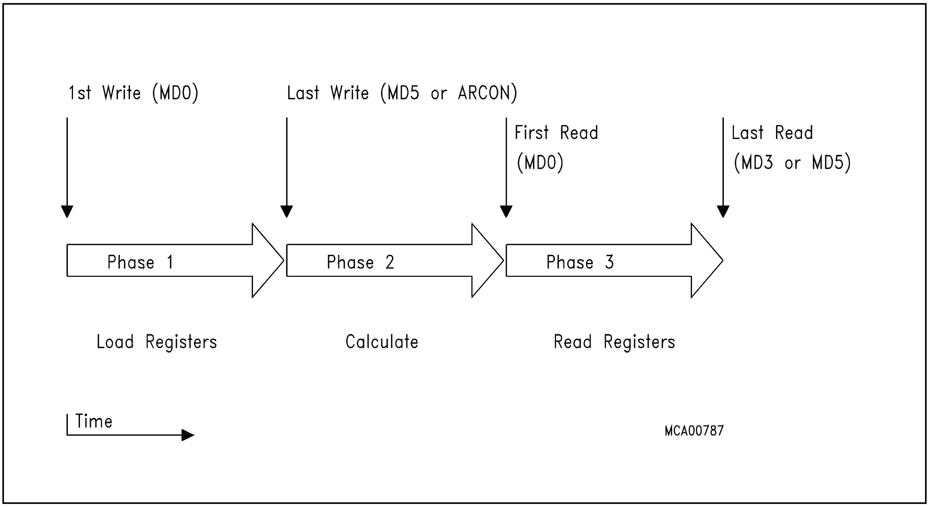


Figure 6-37
Operating Phases of the MDU

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

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

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

6.4.3 Multiplication/Division

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

Table 6-10
Programming the MDU for Multiplication and Division

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

Write Sequence

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

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

Read Sequence

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

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

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

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

6.4.4 Normalize and Shift

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

Write Sequence

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

Read Sequence

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

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

Normalizing

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

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

After normalizing, bits ARCON.0 to ARCON.4 contain the number of shift left operations which were done. This number may further on be used as an exponent. The maximum number of shifts in a normalize operation is 31 (= 2^5 – 1). The operation takes six machine cycles at most, that means 1.5 microseconds at 12 MHz.

Shifting

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

The first machine cycle of a shift left/right operation executes four shifts, while all following cycles perform 6 shifts. Hence, a 31-bit shift takes 1.5 microseconds at 12 MHz.

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

For proper operation of the error flag mechanism, it is necessary to take care that the right write or read sequence to or from registers MD0 to MD3 (see **table 6-12**) is maintained.

Table 6-11
Programming a Shift or Normalize Operation

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

6.4.5 The Overflow Flag

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

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

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

6.4.6 The Error Flag

An error flag, bit MDEF in register ARCON (**figure 7-56**), is provided to indicate whether one of the arithmetic operations of the MDU (multiplication, division, normalize, shift left/right) has been restarted or interrupted by a new operation.

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

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

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

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

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

6.5 Serial Interfaces

The C509-L has two serial interfaces which are functionally nearly identical concerning the asynchronous modes of operation. The two channels are full-duplex, meaning they can transmit and receive simultaneously. They are also receive buffered, meaning they can commence reception of a second byte before a previously received byte has been read from the receive register (however, if the first byte still has not been read by the time reception of the second byte is complete, the last received byte will be lost). The serial channel 0 is completely compatible with the serial channel of the C501. Serial channel 1 has the same functionality in its asynchronous modes, but the synchronous mode is missing.

6.5.1 Serial Interface 0

6.5.1.1 Operating Modes of Serial Interface 0

The serial interface 0 can operate in four modes (one synchronous mode, three asynchronous modes). The baud rate clock for this interface is derived from the oscillator frequency (mode 0, 2) or generated either by timer 1 or by a dedicated baud rate generator (mode 1, 3). A more detailed description of how to set the baud rate will follow in **section 6.5.1.3**.

Mode 0: Shift register (synchronous) mode:

Serial data enters and exits through RXD0. TXD0 outputs the shift clock. 8 data bits are transmitted/received (LSB first). The baud rate is fixed at 1/6 of the oscillator frequency.

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

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

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

11 bits are transmitted (through TXD0) or received (through RXD0): a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). On transmission, the 9th data bit (TB80 in S0CON) can be assigned to the value of 0 or 1. For example, the parity bit (P in the PSW) could be moved into TB80 or a second stop bit by setting TB80 to 1. On reception the 9th data bit goes into RB80 in special function register S0CON, while the stop bit is ignored. The baud rate is programmable to either 1/16 or 1/32 of the oscillator frequency.

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

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

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

The control and status bits of the serial interface 0 are located in special function register S0CON. S0BUF is the receive and transmit buffer of serial interface 0. Writing to S0BUF loads the transmit register and initiates transmission. Reading out S0BUF accesses a physically separate receive register.

6.5.1.2 Multiprocessor Communication Feature

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

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

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

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

Bit No.	MSB 9F$_H$	9E$_H$	9D$_H$	9C$_H$	9B$_H$	9A$_H$	99$_H$	LSB 98$_H$	
98$_H$	SM0	SM1	SM20	REN0	TB80	RB80	TI0	RI0	S0CON

	7	6	5	4	3	2	1	0	
99$_H$	Serial Interface 0 Buffer Register								S0BUF

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>SM0
SM1</td><td>Serial port 0 mode selection bits
<table>
<tr><th>SM0</th><th>SM1</th><th>Selected operating mode</th></tr>
<tr><td>0</td><td>0</td><td>Serial mode 0 : Shift register, fixed baud rate (f_{OSC}/6)</td></tr>
<tr><td>0</td><td>1</td><td>Serial mode 1 : 8-bit UART, variable baud rate</td></tr>
<tr><td>1</td><td>0</td><td>Serial mode 2 : 9-bit UART, fixed baud rate (f_{OSC}/16 or f_{OSC}/32)</td></tr>
<tr><td>1</td><td>1</td><td>Serial mode 3 : 9-bit UART, variable baud rate</td></tr>
</table></td></tr>
<tr><td>SM20</td><td>Enable serial port 0 multiprocessor communication in modes 2 and 3
In mode 2 or 3, if SM2 is set to 1 then RI0 will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI0 will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0.</td></tr>
<tr><td>REN0</td><td>Enable receiver of serial port 0
Enables serial reception. Set by software to enable serial reception. Cleared by software to disable serial reception.</td></tr>
<tr><td>TB80</td><td>Serial port 0 transmitter bit 9
TB80 Is the 9th data bit that will be transmitted in modes 2 and 3. Set or cleared by software as desired.</td></tr>
<tr><td>RB80</td><td>Serial port 0 receiver bit 9
In modes 2 and 3, RB80 is the 9th data bit that was received. In mode 1, if SM2 = 0, RB80 is the stop bit that was received. In mode 0, RB80 is not used.</td></tr>
<tr><td>TI0</td><td>Serial port 0 transmitter interrupt flag
TI0 is set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. TI0 must be cleared by software.</td></tr>
<tr><td>RI0</td><td>Serial port 0 receiver interrupt flag
RI0 is set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes, in any serial reception (exception see SM20). RI0 must be cleared by software.</td></tr>
</table>

6.5.1.3 Baud Rates of Serial Channel 0

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

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

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

Special Function Register ADCON0 (Address $D8_H$) **Reset Value : 00_H**
Special Function Register PCON (Address 87_H) **Reset Value : 00_H**
Special Function Register PRSC (Address $B4_H$) **Reset Value : 11010101_B**

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

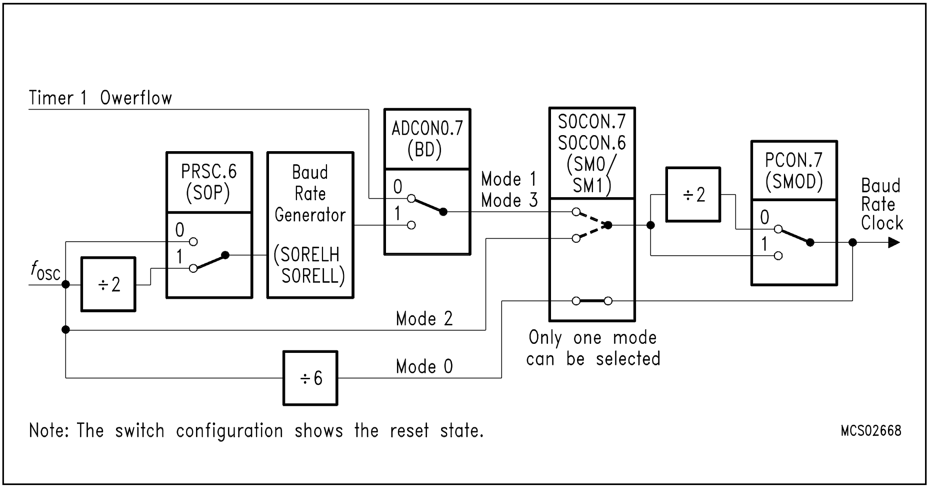
	7	6	5	4	3	2	1	0	
87_H	SMOD	PDS	IDLS	SD	GF1	GF0	PDE	IDLE	PCON

	7	6	5	4	3	2	1	0	
$B4_H$	WDTP	S0P	T2P1	T2P0	T1P1	T1P2	T0P1	T0P0	PRSC

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

Bit	Function
BD	Baud rate generator enable When set, the baud rate of serial interface 0 is derived from a dedicated baud rate generator. When cleared (default after reset), baud rate is derived from the timer 1 overflow rate.
SMOD	Double baud rate When set, the baud rate of serial interface 0 in modes 1, 2, 3 is doubled. After reset this bit is cleared.
S0P	Serial interface 0 prescaler When set (default after reset), an additional prescaler by 2 is active at the dedicated baud rate generator.

Figure 6-38 shows the configuration for the baud rate generation of serial channel 0.



Note: The switch configuration shows the reset state.

Figure 6-38
Baud Rate Generation for Serial Channel 0

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

Table 6-12
Serial Interface 0 - Baud Rate Dependencies

Serial Interface 0 Operating Modes	Active Control Bits			Baud Rate Clock Generation
	BD	S0P	SMOD	
Mode 0 (Shift Register)	–	–	–	Fixed baud rate clock fosc/6
Mode 1 (8-bit UART) Mode 3 (9-bit UART)	X	X	X	BD=0: Timer 1 overflow is used for baud rate generation; SMOD controls a divide-by-2 option. BD=1: Baud rate generator is used for rate generation; S0P and SMOD control two divide-by-2 options.
Mode 2 (9-bit UART)	–	–	X	Fixed baud rate clock fosc/16 (SMOD=1) or fosc/32 (SMOD=0)

Note: If the internal baud generator is selected in mode 1,3, S0P has no effect on the prescaler function if the reload value in S0RELL/S0RELH is $3FF_H$ (S0P divider ratio of ÷1 implicitly selected).

Baud Rate in Mode 0

The baud rate in mode 0 is fixed to:

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

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.

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

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.

Using the baud rate generator:

In modes 1 and 3, the C509-L can use an internal baud rate generator for serial interface 0. To enable this feature, bit BD (bit 7 of special function register ADCON0) must be set. Bit S0P (PRSC.6) and SMOD (PCON.7) control divide-by-2 circuits which affect the input and output clock signal of the baud rate generator. After reset both divide-by-2 circuits are active. So, the input clock of the baud rate generator is $f_{OSC}/2$ and the resulting overflow output clock will be divided by 2.

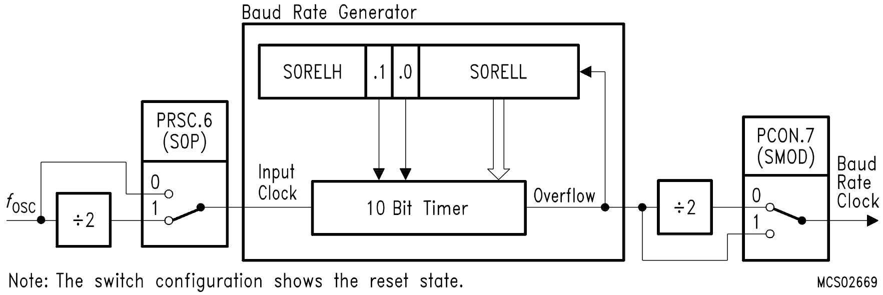


Figure 6-39
Serial Interface 0 Input Clock using the Baud Rate Generator

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

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

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

Bit	Function
S0RELH.0-1	Reload value. Upper two bits of the timer reload value.
S0RELL.0-7	Reload value. Lower 8 bit of timer reload value.

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

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

$$\text{Mode 1, 3 baud rate} = \frac{2^{S0P} \text{ x } 2^{SMOD} \text{ x oscillator frequency}}{64 \text{ x (baud rate generator overflow rate)}}$$

Baud rate generator overflow rate = 2^{10} – S0REL
with S0REL = S0RELH.1 – 0, S0RELL.7 – 0

Using timer 1 to generate baud rates:

In mode 1 and 3 of serial interface 0 timer 1 can be used for generating baud rates. Then the baud rate is determined by the timer 1 overflow rate and the value of SMOD as follows:

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

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:

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

PRSC is the prescaler ratio which is selected via the bits T1P1 and T1P0 in the special function register PRSC.

Timer 1 Prescaler Bits		PRSC Value in the forrmula above
T1P1	**T1P0**	
0	0	1
0	1	2
1	0	4
1	1	8

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.

6.5.2 Serial Interface 1

6.5.2.1 Operating Modes of Serial Interface 1

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

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

11 bits are transmitted (through TXD1) or received (through RXD1): a start bit (0), 8 data bits (LSB first), a programmable 9th bit, and a stop bit (1). On transmission, the 9th data bit (TB81 in S1CON) can be assigned to the value of 0 or 1. For example, the parity bit (P in the PSW) could be moved into TB81 or a second stop bit by setting TB81 to 1. On reception the 9th data bit goes into RB81 in special function register S0CON, while the stop bit is ignored. In fact, mode A of serial interface 1 is identical with mode 2 or 3 of serial interface 0 in all respects except the baud rate generation.

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

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

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

The control and status bits of the serial channel 1 in special function register S1CON and the transmit/receive data register S1BUF are shown on the next page. Writing to S1BUF loads the transmit register and initiates transmission. Reading out S1BUF accesses a physically separate receive register.

Note that these special function registers are not bit-addressable. Due to this fact bit instructions cannot be used for manipulating these registers. This is important especially for S1CON where a polling and resetting of the RI1 or TI1 request flag cannot be performed by JNB and CLR instructions but must be done by a sequence of byte instructions, e.g.:

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

Special Function Register S1CON (Address $9B_H$) **Reset Value : 01000000_B**
Special Function Register S1BUF (Address $9C_H$) **Reset Value : XX_H**

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
$9B_H$	SM	S1P	SM21	REN1	TB81	RB81	TI1	RI1	S1CON
	7	6	5	4	3	2	1	0	
$9C_H$	Serial Interface 1 Buffer Register								S1BUF

Bit	Function

SM	Serial port 1 mode select bit
	SM = 0: Serial mode A; 9-bit UART
	SM = 1: Serial mode B; 8-bit UART

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

6.5.2.2 Multiprocessor Communication Feature

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

If the master processor wants to transmit a block of data to one of the several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM21 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM21 bit and prepare to receive the data bytes that will be coming. After having received a complete message, the slave is setting SM21 again. The slaves that were not addressed leave their SM21 set and go on about their business, ignoring the incoming data bytes.

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

6.5.2.3 Baud Rates of Serial Channel 1

As already mentioned serial interface 1 uses its own dedicated baud rate generator for baud rate generation in both operating modes (see **figure 6-40**). This baud rate generator consists of a free running 10-bit timer with $f_{OSC}/2$ (S1P=1) or f_{OSC} (S1P=0) as input frequency. On overflow of this timer (next count step after counter value $3FF_H$) there is an automatic 10-bit reload from the registers S1RELL and S1RELH. The lower 8 bits of the timer are reloaded from S1RELL, while the upper two bits are reloaded from bit 0 and 1 of register S1RELH. The baud rate timer is reloaded by writing to S1RELL.

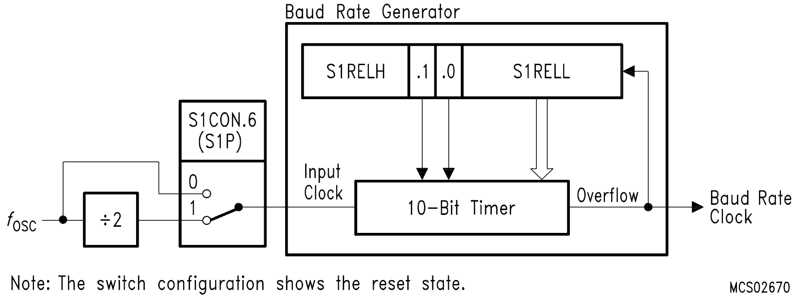


Figure 6-40
Baud Rate Generator for Serial Interface 1

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

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

Bit	Function
S1RELH.0-1	Reload value. Upper two bits of the timer reload value.
S1RELL.0-7	Reload value. Lower 8 bit of timer reload value.

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

$$\text{Mode A, B baud rate} = \frac{2^{\text{S0P}} \text{ x } \text{ oscillator frequency}}{\text{32 x (baud rate generator overflow rate)}}$$

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

At 12-MHz oscillator frequency a baud rate range from about 366 baud up to 750 kbaud is covered.

6.5.3 Detailed Description of the Operating Modes

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

6.5.3.1 Mode 0, Synchronous Mode (Serial Interface 0)

Serial data enters and exits through RXD0. TXD0 outputs the shift clock. 8 data bits are transmitted/received (LSB first). The baud rate is fixed at 1/6 of the oscillator frequency.

Figure 6-41a shows a simplified functional diagram of the serial port in mode 0. The associated timing is illustrated in **figure 2-42b**.

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

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

As data bits shift out to the right, zeros come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just left of the MSB, and all positions to the left of that contain zeros. This condition flags the TX control block to do one last shift and then deactivates SEND and sets TI0. Both of these actions occur at S1P1 in the 10th machine cycle after “Write-to-S0BUF”.

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

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

As data bits come in from the right, 1 s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX control block to do one last shift and load S0BUF. At S1P1 in the 10th machine cycle after the write to S0CON that cleared RI0, RECEIVE is cleared and RI0 is set.

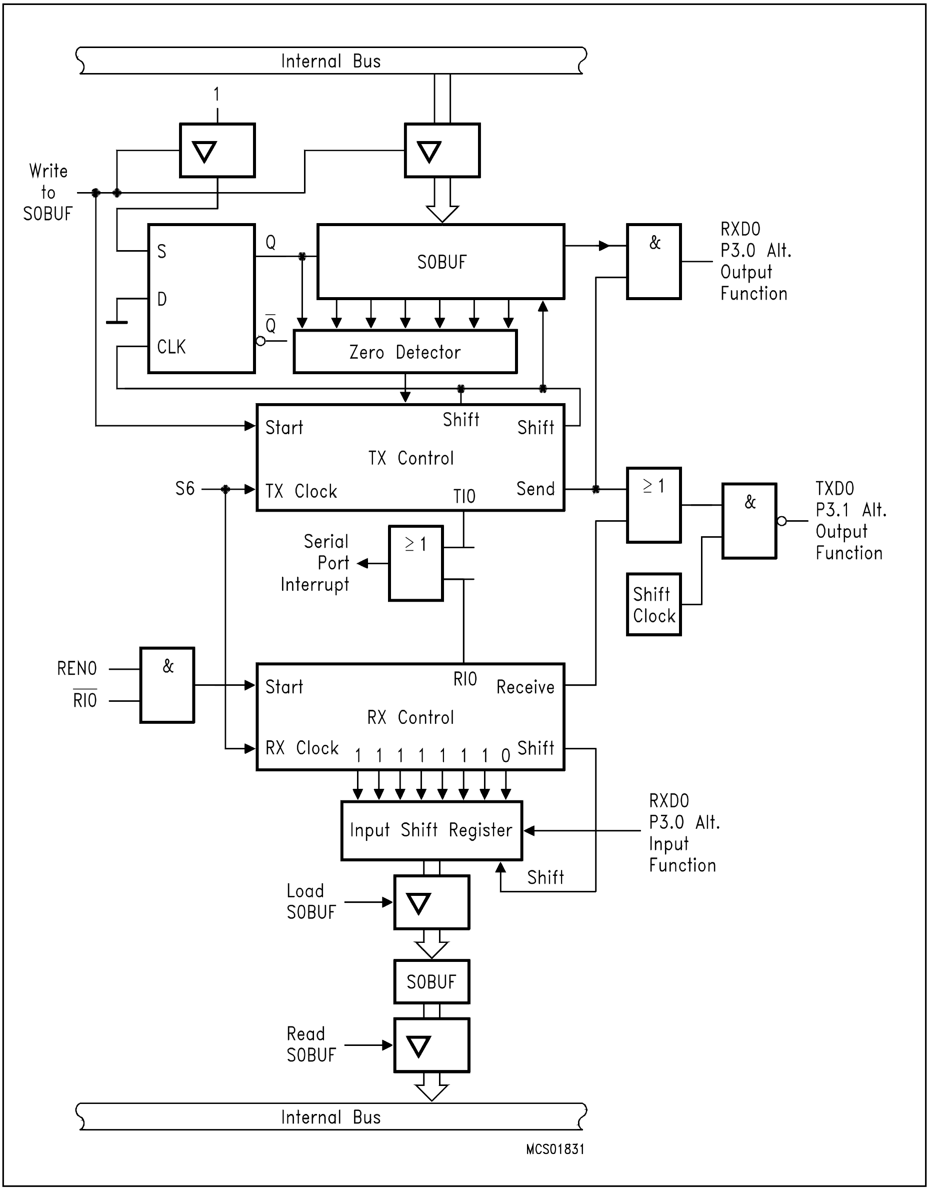


Figure 6-41
Functional Diagram - Serial Interface 0, Mode 0

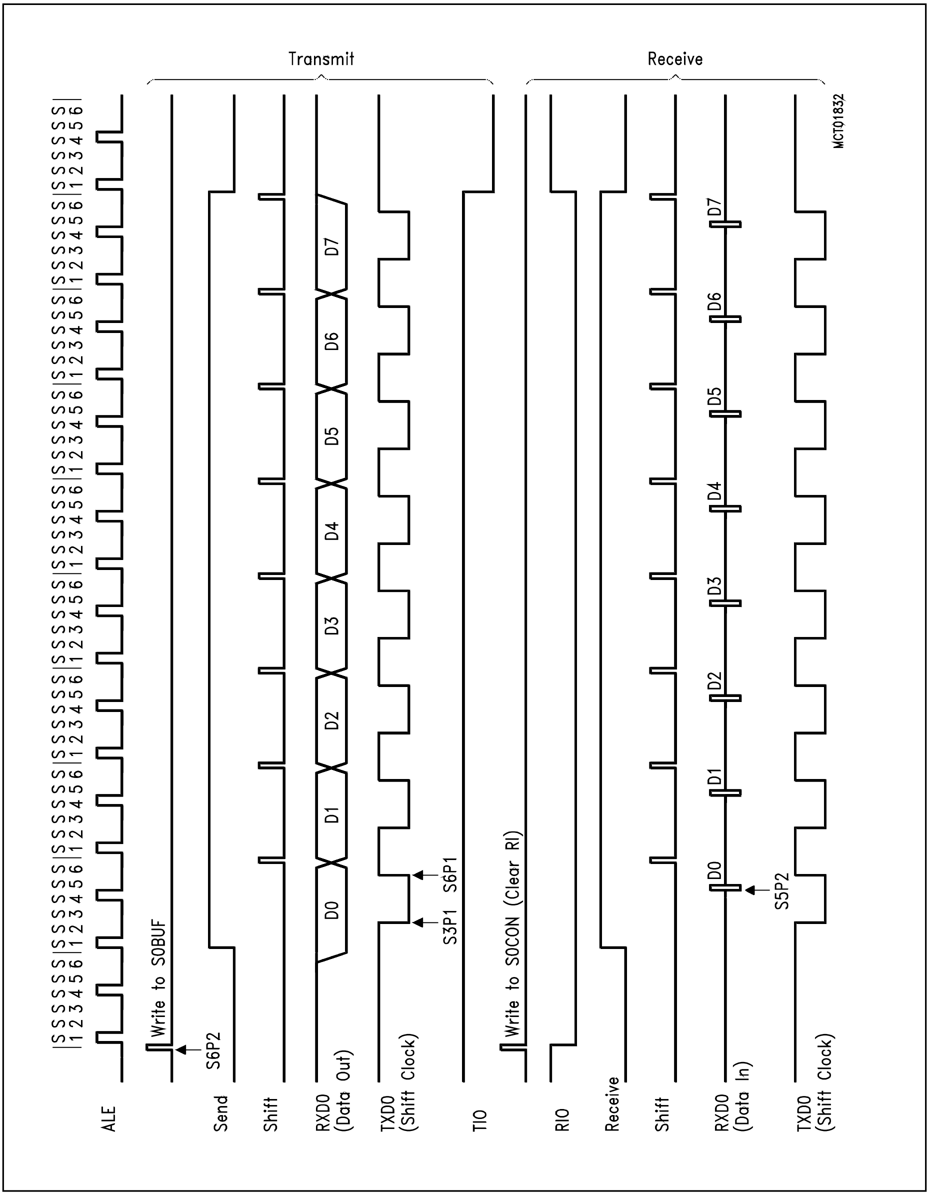


Figure 6-42
Timing Diagram - Serial Interface 0, Mode 0

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

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

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

Figure 6-43a shows a simplified functional diagram of the both serial channels in mode 1 or mode B, respectively. The associated timing is illustrated in **figure 2-44b**.

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

The transmission begins with activation of $\overline{\text{SEND}}$, which puts the start bit to TXD0/TXD1. One bit time later, DATA is activated, which enables the output bit of the transmit shift register to TXD0/TXD1. The first shift pulse occurs one bit time after that.

As data bits shift out to the right, zeros are clocked in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position is just left of the MSB, and all positions to the left of that contain zero. This condition flags the TX control to do one last shift and then deactivate $\overline{\text{SEND}}$ and set TI0/TI1. This occurs at the 10th divide-by-16 rollover after "write-to-S0BUF/S1BUF".

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

The 16 states of the counter divide each bit time into 16 counter states. At the 7th, 8th and 9th counter state of each bit time, the bit detector samples the value of RXD0/RXD1. The value accepted is the value that was seen in at least 2 of the 3 samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

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

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

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

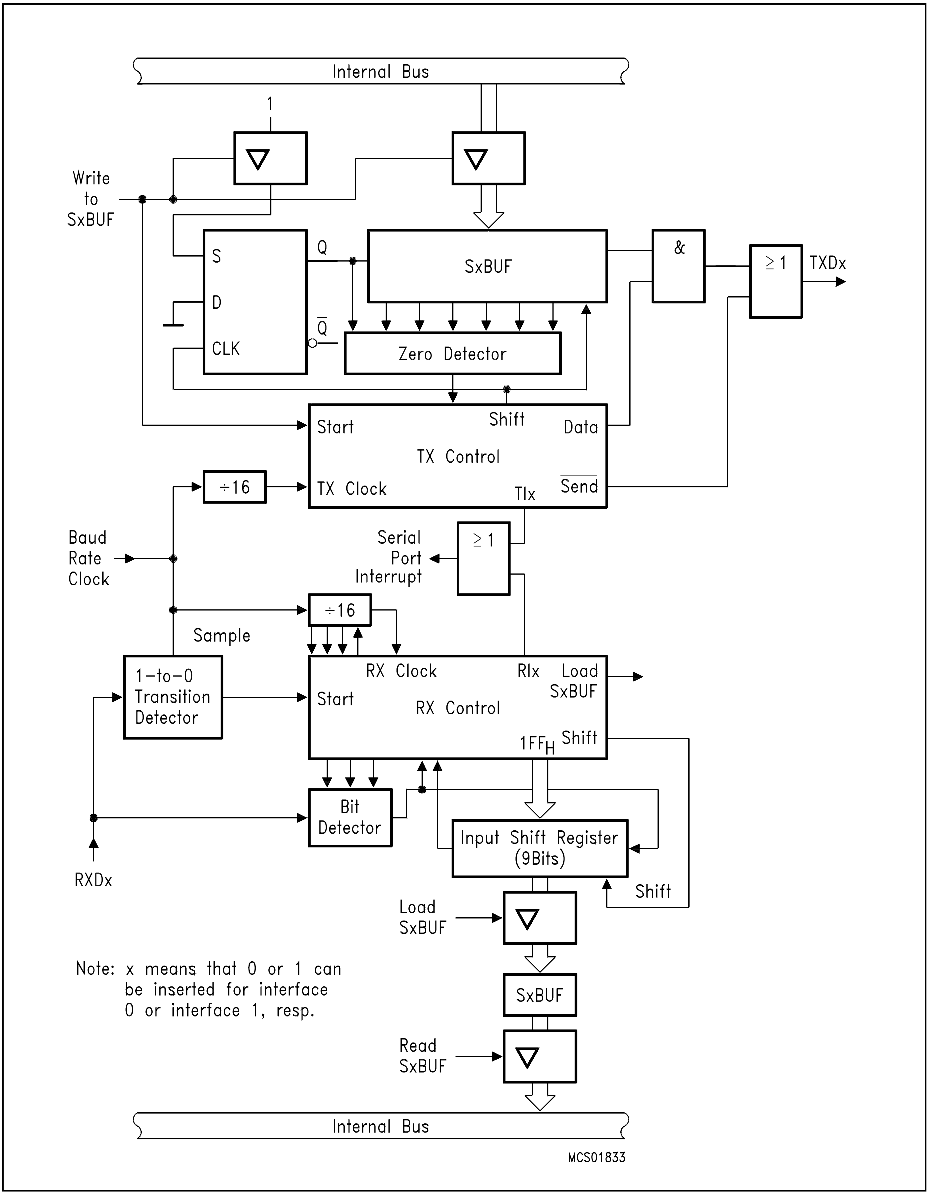
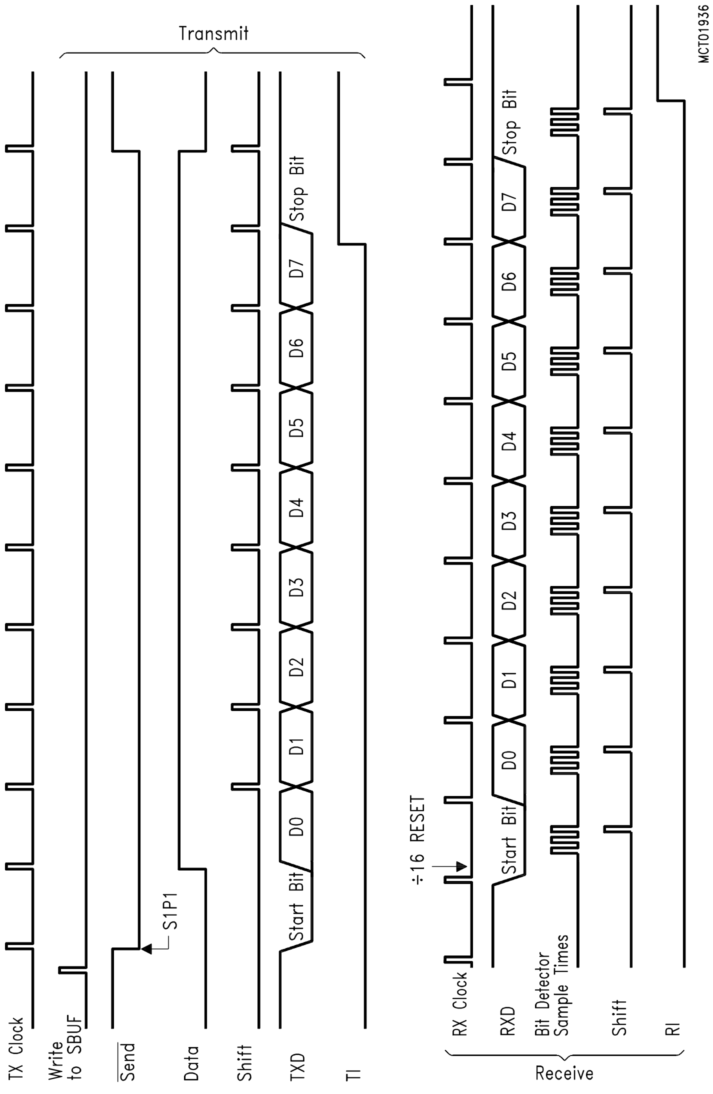


Figure 6-43
Functional Diagram - Serial Interfaces 0 and 1, Mode 1 / Mode B



Note : x=0 for serial interface 0 and x=1 for serial interface 1.

Figure 6-44
Timing Diagram - Serial Interfaces 0 and 1, Mode 1 / Mode B

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

Mode 2 is functionally identical to mode 3 (see below). The only exception is, that in mode 2 the baud rate can be programmed to two fixed quantities: either 1/16 or 1/32 of the oscillator frequency. Note that serial interface 0 cannot achieve this baud rate in mode 3. Its baud rate clock is generated by timer 1, which is incremented by a rate of f_{OSC}/6. The dedicated baud rate generator of serial interface 1 however is clocked by a f_{OSC} signal and so its maximum baud rate is f_{OSC}/16.

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

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

Figure 6-45a shows a simplified functional diagram of the both serial channels in mode 2 an 3 or mode A, respectively. The associated timing is illustrated in **figure 6-46b**. The receive portion is exactly the same as in mode 1. The transmit portion differs from mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses S0BUF/S1BUF as a destination register. The “write to S0BUF/S1BUF” signal also loads TB80/TB81 into the 9th bit position of the transmit shift register and flags the TX control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter (thus the bit times are synchronized to the divide-by-16 counter, and not to the “write-to-S0BUF/S1BUF” signal).

The transmission begins with the activation of SEND, which puts the start bit to TXD0/TXD1. One bit time later, DATA is activated which enables the output bit of transmit shift register to TXD0/TXD1. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeros are clocked in. Thus, as data shift out to the right, zeros are clocked in from the left. When TB80/TB81 is at the output position of the shift register, then the stop bit is just left of the TB80/TB81, and all positions to the left of that contain zeros.

This condition flags the TX control unit to do one last shift and then deactivate SEND and set TI0/TI1. This occurs at the 11th divide-by-16 rollover after “write-to-S0BUF/S1BUF”.

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

At the 7th, 8th and 9th counter state of each bit time, the bit detector samples the value of RxD0/RxD1. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

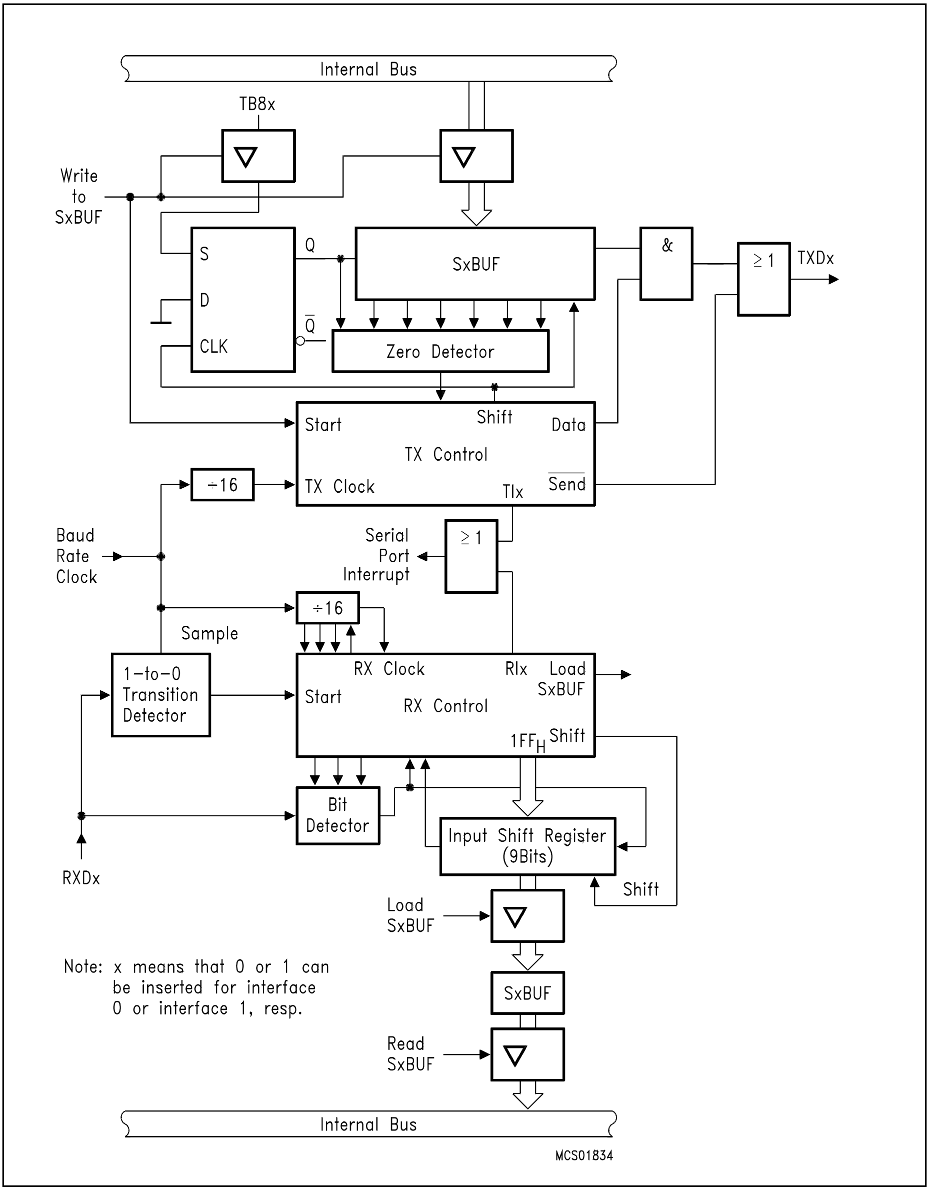
As data bits come from the right, 1’s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which is a 9-bit register), it flags the RX control block to do one last shift, load S0BUF/S1BUF and RB80/ RB81, and set RI0/RI1. The signal to load S0BUF/S1BUF and

RB80/RB81, and to set RI0/RI1, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

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

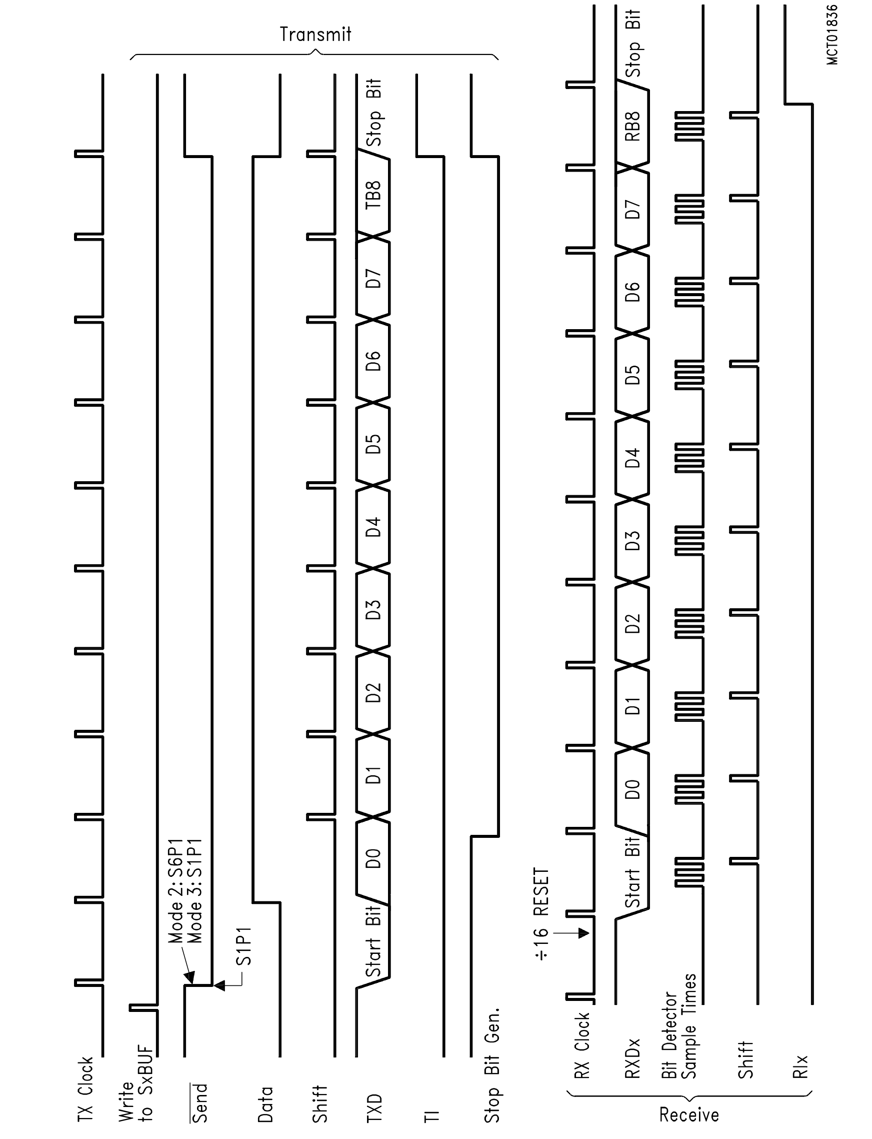
If either one of these two conditions is not met, the received frame is irretrievably lost, and RI0/Rl1 is not set. If both conditions are met, the received 9th data bit goes into RB80/RB81, the first 8 data bits go into S0BUF/S1BUF. One bit time later, no matter whether the above conditions are met or not, the unit goes back to look for a 1-to-0 transition at the RXD0/RXD1 input.

Note that the value of the received stop bit is irrelevant to S0BUF/S1BUF, RB80/RB81, or RI0/Rl1.



Note: x means that 0 or 1 can be inserted for interface 0 or interface 1, resp.

Figure 6-45
Functional Diagram - Serial Interfaces 0 and 1, Modes 2 and 3 / Mode A



Note : x=0 for serial interface 0 and x=1 for serial interface 1.

Figure 6-46
Timing Diagram - Serial Interfaces 0 and 1, Modes 2 and 3 / Mode A

6.6 A/D Converter

The C509-L includes a high performance / high speed 10-bit A/D-Converter (ADC) with 15 analog input channels. This A/D converter operates with a successive approximation technique and uses self calibration mechanisms for reduction and compensation of offset and linearity errors. The A/D converter provides the following features:

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

For the conversion, the method of successive approximation via capacitor array is used. The externally applied reference voltage range has to be held on a fixed value within the specifications.

Figure 6-47 shows a block diagram of the A/D converter. There are four user-accessible special function registers, ADCON1/ADCON0 (A/D converter control registers) and ADDATH/ADDATL (A/D converter data registers). The analog input channels (port 7 and port 8) can also be used for digital input. For interrupt processing of the A/D converter, two bits are located in the special function registers IEN2 and IRCON0.

6.6.1 A/D Converter Operation

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

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

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

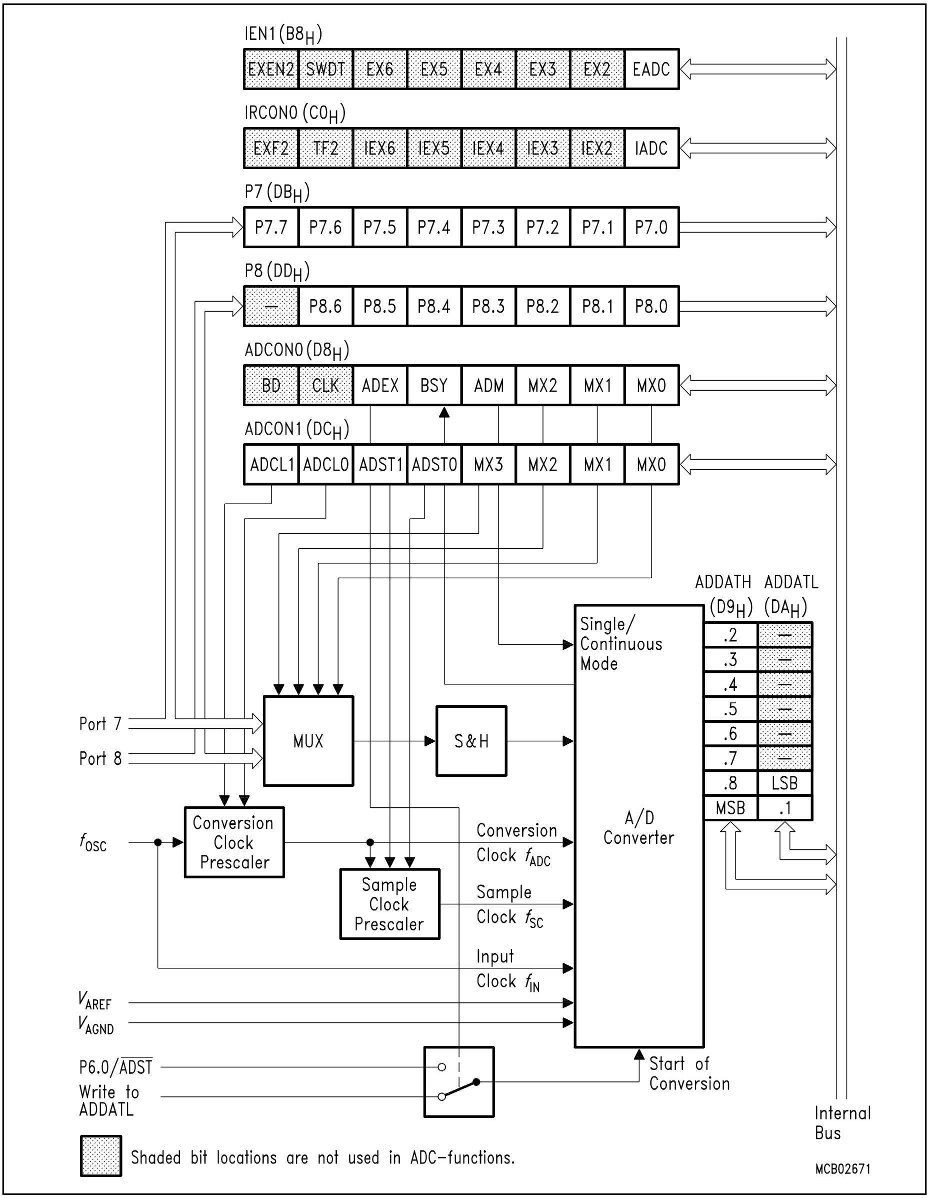


Figure 6-47
Block Diagram of the A/D Converter

The bits MX0 to MX2 in special function register ADCON0 and the bits MX0 to MX3 in ADCON1 are used for selection of the analog input channel. The bits MX0 to MX2 are represented in both registers ADCON0 and ADCON1; however, these bits are present only once. If all 15 multiplexed input channels are required register ADCON1 has to be used for channel selection. It contains the four-bit field MX0-3 to select one of the 15 input channels, the eight inputs at port 7 and the seven inputs at port 8. Thus, there are two methods of selecting a channel of port 7 and it does not matter which is used: 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. If bit MX3 is set, the additional analog inputs at port 8 are used. MX0-2 then determine which channel of port 8 is being selected.

Ports P7 and P8 are dual purpose input ports. If the input voltage meets the specified logic levels, they can be also used as digital inputs regardless of whether the pin levels are sampled by the A/D converter at the same time.

The special function register ADDATH and ADDATL hold the converted digital 10-bit data result. 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 C509-L is not used, register ADDATH can be used as an additional general purpose register.

6.6.2 A/D Converter Registers

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

Special Function Registers ADDATH (Address D9$_H$) **Reset Value : 00$_H$**
Special Function Registers ADDATL (Address DA$_H$) **Reset Value : 00$_H$**

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
D9$_H$	MSB .9	.8	.7	.6	.5	.4	.3	.2	ADDATH
DA$_H$	.1	LSB .0	–	–	–	–	–	–	ADDATL

The registers ADDATH and ADDATL contain the 10-bit conversion result in left justified data format. The most significant bit of the 10-bit conversion result is bit 7 of ADDATH. The least significant bit of the 10-bit conversion result is bit 6 of ADDATL. To get a 10-bit conversion result, both ADDAT register must be read. If an 8-bit conversion result is required, only the reading of ADDATH is necessary. The data 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 C509-L is not used, register ADDATH can be used as an additional general purpose register.

Special Function Registers ADCON0 (Address D8$_H$) **Reset Value : 00$_H$**
Special Function Registers ADCON1 (Address DC$_H$) **Reset Value : 01000000$_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
D8$_H$	BD	CLK	ADEX	BSY	ADM	MX2	MX1	MX0	ADCON0
DC$_H$	ADCL1	ADCL0	ADST1	ADST0	MX3	MX2	MX1	MX0	ADCON1

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

Bit	Function
ADEX	Internal / external start of A/D conversion When set, the external start of an A/D conversion at pin P6.0/$\overline{\text{ADST}}$ is enabled.
BSY	Busy flag This flag indicates whether a conversion is in progress (BSY = 1). The flag is cleared by hardware when the conversion is finished.
ADM	A/D conversion mode When set, a continuous A/D conversion is selected. If cleared during a running A/D conversion, the conversion is stopped at its end.
MX3 - MX0	A/D converter input channel select bits Bits MX3-0 can be written or read either in ADCON0 or ADCON1. When writing to ADCON1, the ADCON0 bits MX2-0 are overwritten. Writing ADCON0, only MX2-0 are affected. The analog inputs are selected according the following table:

MX3	MX2	MX1	MX0	Selected Analog Input
0	0	0	0	P7.0 / AN0
0	0	0	1	P7.1 / AN1
0	0	1	0	P7.2 / AN2
0	0	1	1	P7.3 / AN3
0	1	0	0	P7.2 / AN4
0	1	0	1	P7.3 / AN5
0	1	1	0	P7.4 / AN6
0	1	1	1	P7.5 / AN7
1	0	0	0	P8.0 / AN8
1	0	0	1	P8.1 / AN9
1	0	1	0	P8.2 / AN10
1	0	1	1	P8.3 / AN11
1	1	0	0	P8.4 / AN12
1	1	0	1	P8.5 / AN13
1	1	1	0	P8.6 / AN14

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>ADCL1
ADCL0</td><td>A/D conversion 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 C509-L. f_{ADC} must be selected by the two prescaler bits ADCL1 and ADCL0 in a way that the resulting frequency of the conversion clock f_{ADC} is less or equal 2 MHz.
The prescaler ratio is selected according the following table:
<table>
<tr><th>ADCL1</th><th>ADCL0</th><th>f_{ADC} 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>ADST1
ADST0</td><td>A/D sample clock prescaler selection
ADST1 and ADST0 select the prescaler ratio for the sample clock f_{SC}.
The prescaler ratio is selected according the following table:
<table>
<tr><th>ADST1</th><th>ADST0</th><th>f_{SC} Prescaler Ratio</th></tr>
<tr><td>0</td><td>0</td><td>divide by 2 (default after reset)</td></tr>
<tr><td>0</td><td>1</td><td>divide by 8</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>
</table>

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

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

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

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

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

Special Function Register IEN1 (Address B8$_H$) Reset Value : 00$_H$
Special Function Register IRCON0 (Address C0$_H$) Reset Value : 00$_H$

	MSB							LSB	
Bit No.	BF$_H$	BE$_H$	BD$_H$	BC$_H$	BB$_H$	BA$_H$	B9$_H$	B8$_H$	
B8$_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	IEN1
	C7$_H$	C6$_H$	C5$_H$	C4$_H$	C3$_H$	C2$_H$	C1$_H$	C0$_H$	
C0$_H$	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC	IRCON0

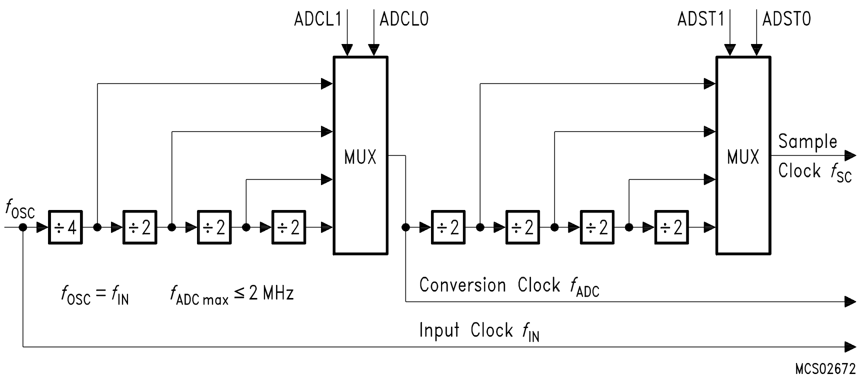
The shaded bits are not used for A/D converter control.

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

6.6.3 A/D Converter Clock Selection

The ADC uses basically three clock signals for operation: the input clock f_{IN} (=1/t_{IN}), the conversion clock f_{ADC} (=1/t_{ADC}) and the sample clock f_{SC} (=1/t_{SC}). All clock signals are derived from the C509-L system clock f_{OSC} which is applied at the XTAL pins. The input clock f_{IN} is equal to f_{OSC} while the conversion clock and the sample clock must be adapted. The conversion clock is limited to a maximum frequency of 2 MHz. Therefore, the conversion clock prescaler must be programmed to a value which assures that the conversion clock does not exceed 2 MHz. The prescaler ratio of the conversion clock is selected by the bits ADCL1 and ADCL0 of SFR ADCON1. The sample clock f_{SC} can be adapted to the requirements of the impedance of A/D converter input signal sources. The prescaler ratio of the sample clock is selected by the bits ADST1 and ADST0 of SFR ADCON1.

Figure 6-48 shows the configuration of the two A/D converter prescalers. The table in **figure 6-48** defines the divider ratio for the conversion and sample clock of each combination of the prescaler bits.



Conversion Clock f_{ADC}			Sample Clock f_{SC}			
ADCL1	ADCL0	f_{ADC}	ADST1 ADST0 0 0	ADST1 ADST0 0 1	ADST1 ADST0 1 0	ADST1 ADST0 1 1
0	0	f_{IN} / 4	f_{IN} / 8	f_{IN} / 16	f_{IN} / 32	f_{IN} / 64
0	1	f_{IN} / 8	f_{IN} / 16	f_{IN} / 32	f_{IN} / 64	f_{IN} / 128
1	0	f_{IN} / 16	f_{IN} / 32	f_{IN} / 64	f_{IN} / 128	f_{IN} / 256
1	1	f_{IN} / 32	f_{IN} / 64	f_{IN} / 128	f_{IN} / 256	f_{IN} / 512

Figure 6-48
A/D Converter Clock Selection

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

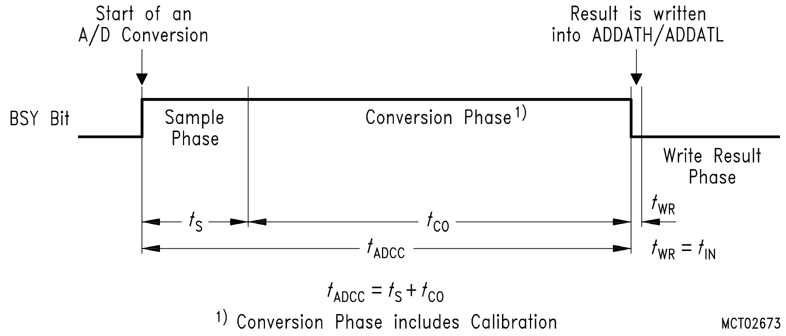
6.6.4 A/D Conversion Timing

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

Basically, 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 real A/D conversion.(includes calibration)
– Write result phase (t_{WR}), used for writing the conversion result into the ADDAT registers.

The total A/D conversion time is defined by t_{ADCC} which is the sum of the two phase times t_S and t_{CO}. The duration of the three phases of an A/D conversion is specified by its specific timing parameter as shown in **figure 6-49**.



Conversion Clock Prescaler		Sample Clock Prescaler		Sample Time	Conversion Time	ConversionTime	
						t_{ADCC}	CPU Cycles
ADCL1	ADCL0	ADST1	ADST0	t_S	t_{CO}		n Tim. Ref
0	0	0	0	8 x t_{IN}	40 x t_{IN}	48 x t_{IN}	8 a)
		0	1	16 x t_{IN}		56 x t_{IN}	9 b)
		1	0	32 x t_{IN}		72 x t_{IN}	12 a)
		1	1	64 x t_{IN}		104 x t_{IN}	17 b)
0	1	0	0	16 x t_{IN}	80 x t_{IN}	96 x t_{IN}	16 a)
		0	1	32 x t_{IN}		112 x t_{IN}	18 c)
		1	0	64 x t_{IN}		144 x t_{IN}	24 a)
		1	1	128 x t_{IN}		208 x t_{IN}	34 c)
1	0	0	0	32 x t_{IN}	160 x t_{IN}	192 x t_{IN}	32 a)
		0	1	64 x t_{IN}		224 x t_{IN}	37 b)
		1	0	128 x t_{IN}		288 x t_{IN}	48 a)
		1	1	256 x t_{IN}		416 x t_{IN}	69 b)
1	1	0	0	64 x t_{IN}	320 x t_{IN}	384 x t_{IN}	64 a)
		0	1	128 x t_{IN}		448 x t_{IN}	74 c)
		1	0	256 x t_{IN}		576 x t_{IN}	96 a)
		1	1	512 x t_{IN}		832 x t_{IN}	138 c)

Figure 6-49
A/D Conversion Timing

Sample Time t_S :
During this time the internal capacitor array is connected to the selected analog input channel and is loaded with the analog voltage to be converted. The external analog source needs to be strong enough to source the current to load the analog input capacitance during the load time. This causes some restrictions for the impedance of the analog source. The analog voltage is internally fed to a voltage comparator. With beginning of the sample phase the BSY bit in SFR ADCON0 is set.

Conversion Time t_{CO} :
During the conversion time the analog voltage is converted into a 10-bit digital value using the successive approximation technique with a binary weighted capacitor network. During a conversion alternating offset and linearity calibration cycles are executed (see also section 6.6.6). At the end of the conversion phase the BSY bit is reset and the IADC bit in SFR ADCON0 is set indicating a A/D converter interrupt condition.

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

Figure 6-50 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.

Depending on the selected prescaler ratios (see **figure 6-48**), 3 different relationships between machine cycles and A/D conversion are possible. These 3 relationships are referenced in the rightmost column of the table in **figure 6-49** as a), b), and c).

The single A/D conversion is started when SFR ADDATL is written with dummy data. This write operation may take one or two machine cycles. In **figure 6-50**, the instruction MOV ADDATL,#0 starts the A/D conversion (machine cycle X-1 and X). The total A/D conversion (sample, conversion, and calibration phase) is finished after the end of the n-th machine cycle. In the next machine cycle n+1 the conversion result is written into the ADDAT registers and can be read in the same cycle by an instruction (e.g. MOV A,ADDATL). If continuous conversion is selected (bit ADM set), the next conversion is started with the beginning of the machine cycle which follows the writre result cycle.

The BSY bit is set at the beginning of the first A/D conversion machine cycle and reset at the beginning of the write result cycle. If continuous conversion is selected, BSY is again set with the beginning of the machine cycle which follows the write result cycle (n+1).

The interrupt flag IADC is set at the end of the A/D conversion. If the A/D converter interrupt is enabled and the A/D converter interrupt is priorized to be serviced immediately, the first instruction of the interrupt service routine will be executed in machine cycle n+4 or n+5. IADC must be reset by software.

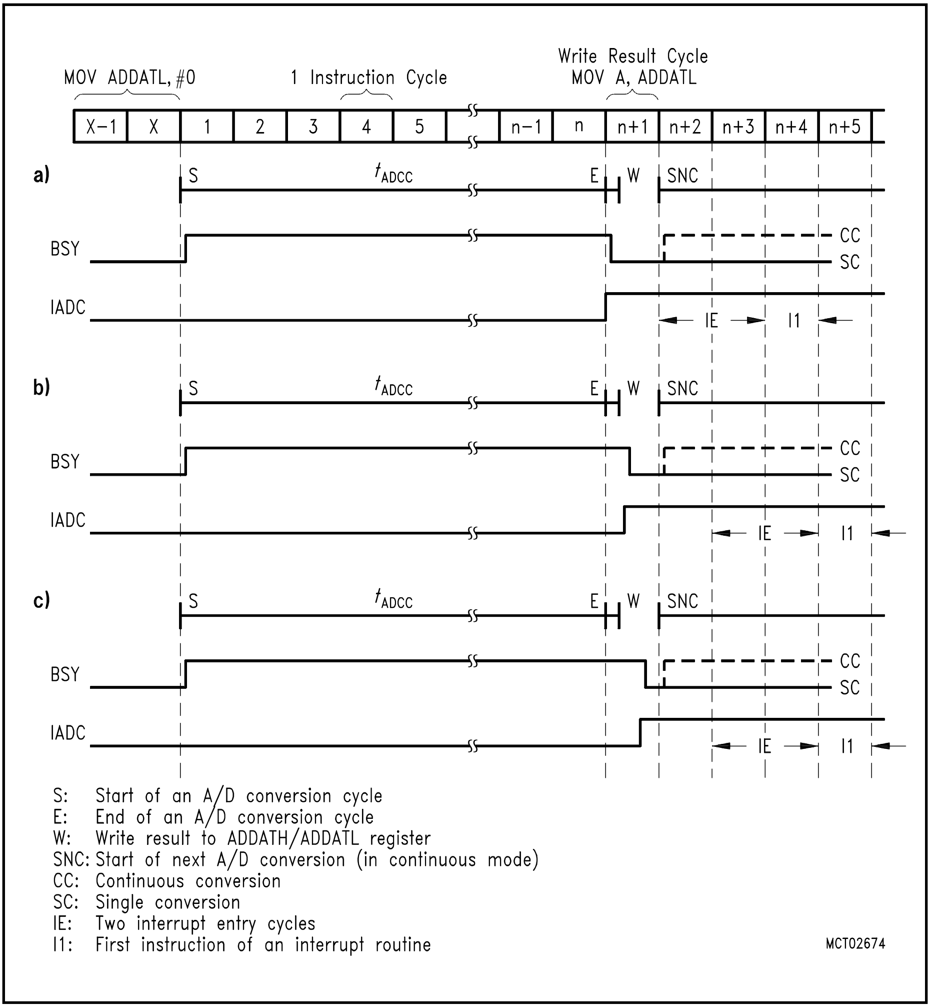

Write Result Cycle
MOV A, ADDATL
MOV ADDATL, #0
1 Instruction Cycle
X−1
X
1
2
3
4
5
n−1
n
n+1
n+2
n+3
n+4
n+5
a)
b)
c)
S
t_{ADCC}
E
W
SNC
BSY
IADC
CC
SC
IE
I1
S: Start of an A/D conversion cycle
E: End of an A/D conversion cycle
W: Write result to ADDATH/ADDATL register
SNC: Start of next A/D conversion (in continuous mode)
CC: Continuous conversion
SC: Single conversion
IE: Two interrupt entry cycles
I1: First instruction of an interrupt routine
MCT02674

Figure 6-50
A/D Conversion Timing in Relation to Processor Cycles

Depending on the application, typically there are three software methods to handle the A/D conversion in the C509-L.

- Software delay
  The machine cycles of the A/D conversion are counted and the program executes a software delay (e.g. NOPs) before reading the A/D conversion result in the write result cycle. This is the fastest method to get the result of an A/D conversion.
- Polling BSY bit
  The BSY bit is polled and the program waits until BSY=0. Attention : a polling JB instruction which is two machine cycles long, possibly may not recognize the BSY=0 condition during the write result cycle in the continuous conversion mode.
- A/D conversion interrupt
  After the start of an A/D conversion the A/D converter interrupt is enabled. The result of the A/D conversion is read in the interrupt service routine. If other SAB-C509 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 C509-L and the selected divider ratios of the A/D converter prescalers the total time of an A/D conversion is calculated according to **table 6-13.** The minimum conversion time is 6 µs.

Table 6-13
A/D Conversion Times

Conversion Clock Prescaler		Sample Clock Prescaler		Processor Clock Rate							
				3.5 MHz		8 MHz		12 MHz		16 MHz	
ADCL1	ADCL0	ADST1	ADST0	f_{ADC} [MHz]	t_{ADCC} [µs]	f_{ADC} [MHz]	t_{ADCC} [µs]	f_{ADC} [MHz]	t_{ADCC} [µs]	f_{ADC} [MHz]	t_{ADCC} [µs]
0	0	0	0	0.875	13.7	2	6	3	4	4	3
		0	1		16		7		4.7		3.5
		1	0		20.6		9		6		4.5
		1	1		29.7		13		8.7		6.5
0	1	0	0	0.438	27.4	1	12	1.5	8	2	6
		0	1		32		14		9.3		7
		1	0		41.1		18		12		9
		1	1		59.4		26		17.3		13
1	0	0	0	0.219	54.9	0.5	24	0.75	16	1	12
		0	1		64		28		18.7		14
		1	0		82.3		36		24		18
		1	1		118.9		52		34.7		26
1	1	0	0	0.109	109.7	0.25	48	0.375	32	0.5	24
		0	1		128		56		37.3		28
		1	0		164.6		72		48		36
		1	1		237.7		104		69.3		52

Note : The shaded prescaler combination cannot be used. Reason : f_{ADC} > 2 MHz.

6.6.5 Adjustment of the Sample Time

As already discussed, the maximum input clock f_{ADC} for the A/D converter is 2 MHz. This must be taken into account when programming the A/D converter input clock prescalers. Additionally, the C509-L allows to adapt the sample phase of an A/D conversion to the impedance of an analog input source. This chapter gives some hints how an optimal adaption of the sample phase is achieved.

At the end of an A/D conversion (single conversion mode) the internal capacitor array is internally precharged to a voltage level between V_{AGND} and V_{AREF} (approx. 2-3 V). When an A/D conversion of an analog voltage U_0 is started, the voltage level at the analog input pin drops (or raises) shortly to the precharge voltage level U_P and then raises (or drops) back to U_0 with a typical RC load curve (see **figure 6-51**).

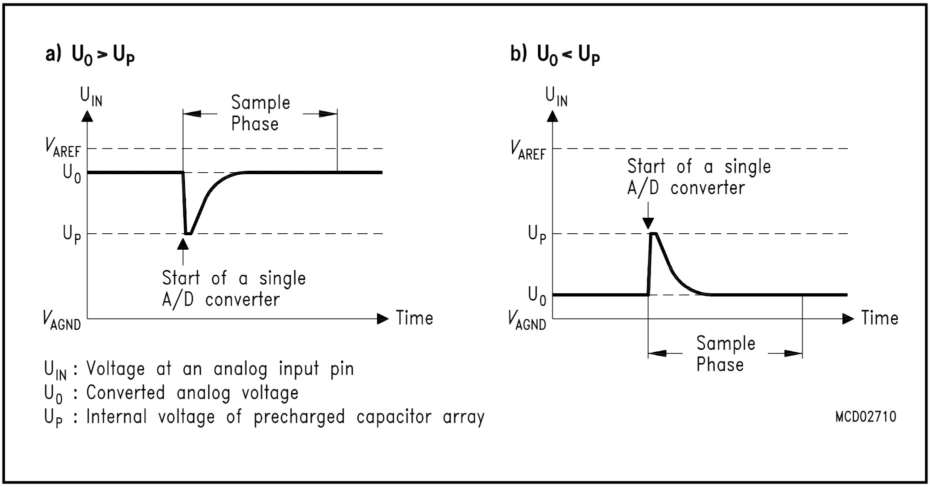


Figure 6-51
Typical Wavefoms at the Analog Inputs During the Sample Phase

The characteristics of the RC load curves as shown in **figure 6-51 a)** and **b)** is determined by the input capacitance C_{IN} of the analog input pin, by the impedance R of the driving analog source, and by the voltage difference between U_0 and U_P.

The external analog source needs to be strong enough (low impedance) to source the current for loading the analog input capacitance. Depending on the accuracy or error requirements of an A/D conversion, the limits for the input impedance of the analog source can be calculated using the following formula :

$$R_{MAX} = - t_S / (C_{IN} \bullet \ln (\text{error})) \qquad \text{with} \quad \text{error} = 1 - \frac{u(t)}{U_0}$$

t_S : sample time
C_{IN} : analog input capacitance (typically 50 pF)
error : allowed deviation of U_{IN} from U_0 at the end of the sample time, given in %

Table 6-14 shows the resulting maximum values for R_{MAX} at f_{OSC} = 16 MHz and a maximum error of 0.1% (1 or 2 LSB error allowed, additional to the specified total unadjusted error).

Table 6-14
Limit Values for the impedance of the Analog Source at f_{OSC} = 16 MHz (C_{IN} = 50pF)

Conversion Clock Prescaler		Sample Clock Prescaler		Sample Time (16 MHz) t_S [µs]	Impedance R_{MAX} [kΩ] at 16MHz	
ADCL1	ADCL0	ADST1	ADST0		Error	
					1 LSB	2 LSB
0	0	0	0	0.5	–	–
		0	1	1	–	–
		1	0	2	–	–
		1	1	4	–	–
0	1	0	0	1	2.9	3.2
		0	1	2	5.8	6.4
		1	0	4	11.6	12.9
		1	1	8	23.1	25.7
1	0	0	0	2	5.8	6.4
		0	1	4	11.6	12.9
		1	0	8	23.1	25.7
		1	1	16	46.3	51.5
1	1	0	0	4	11.6	12.9
		0	1	8	23.1	25.7
		1	0	16	46.3	51.5
		1	1	32	92.6	103

Note : The shaded prescaler combination can only be used up to f_{OSC} = 8 MHz.
Reason : f_{ADC} > 2 MHz for f_{OSC} > 8 MHz.

The values in **table 6-14** are just calculated values and are no assured limit characteristics. System environment and application dependencies must be taken into account, too.

6.6.6 A/D Converter Calibration

The C509-L A/D converter includes hidden internal calibration mechanisms which assure a save functionality of the A/D converter according to the DC characteristics. The A/D converter calibration is implemented in a way that a user program which executes A/D conversions is not affected by its operation. Further, the user program has no control on the calibration mechanism. The calibration itself executes two basic functions :

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

The A/D converter calibration operates in two phases : calibration after a reset operation and calibration at each A/D conversion. The calibration phases are controlled by a state machine in the A/D converter. This state machine executes the calibration phases and stores the calibration results dynamically in a small calibration RAM

After a reset operation the A/D calibration is automatically started. This reset calibration phase which takes 3328 f_{ADC} clocks, alternating offset and linearity calibration is executed. Therefore, at 12 MHz oscillator frequency and with the default after reset prescaler value of 8, a reset calibration time of approx. 2.2 ms is reached. For achieving a proper reset calibration, the f_{ADC} prescaler value must satisfy the condition $f_{ADC\ max} \leq 2$ MHz.

After the reset calibration phase the A/D converter is calibrated according to its DC characteristics. Nevertheless, during the reset calibration phase single or continuous A/D can be executed. In this case it must be regarded that the reset calibration is interrupted and continued after the end of the A/D conversion. Therefore, interrupting the reset calibration phase by A/D conversions extends the total reset calibration time. If the specified total unadjusted error (TUE) has to be valid for an A/D conversion, it is recommended to start the first A/D conversions after reset when the reset calibration phase is finished. Depending on the oscillator frequency used, the reset calibration phase can be possibly shortened by setting ADCL1 and ADCL0 (prescaler value) to its final value immediately after reset.

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

7 Interrupt System

The C509-L provides 19 interrupt sources with four priority levels. 12 interrupts can be generated by the on-chip peripherals and 7 interrupts may be triggered externally.

The interrupt structure of the C509-L has been mainly adapted from the SAB 80C517A microcontroller. Thus, each interrupt source has its dedicated interrupt vector and can be enabled/ disabled individually. There are also four priority levels available.

In the C509-L the 19 interrupt sources are combined to six groups of three or four interrupt sources. Each interrupt group can be programmed to one of the four interrupt priority levels.

7.1 Structure of the Interrupt System

Figure 7-1 to **7-4** give a general overview of the interrupt sources and illustrate the request and control flags described in the next sections.

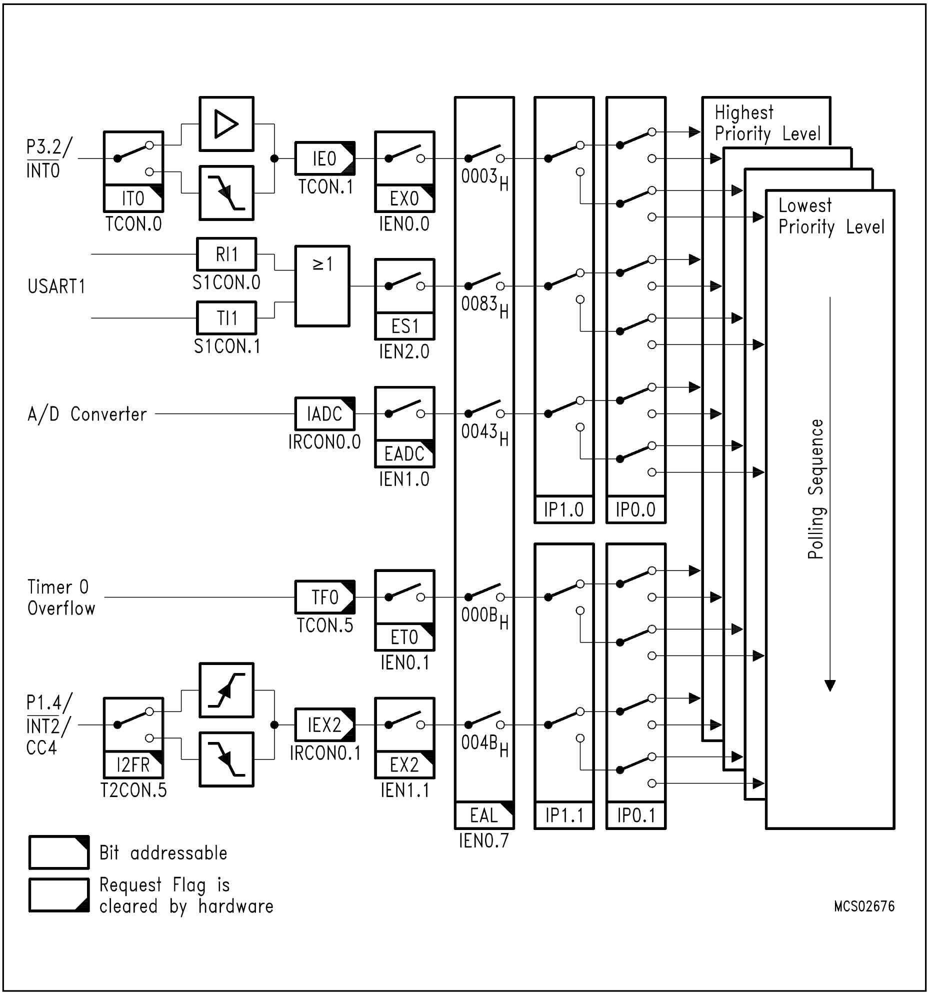


Figure 7-1
Interrupt Structure, Overview Part 1

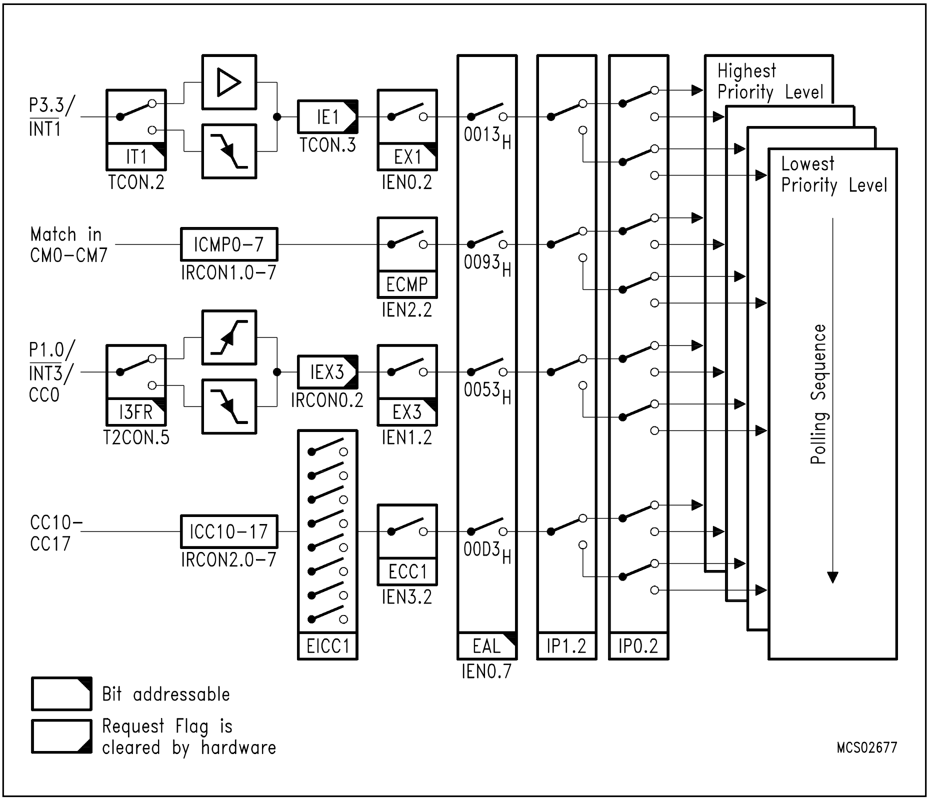


Figure 7-2 :
Interrupt Structure, Overview Part 2

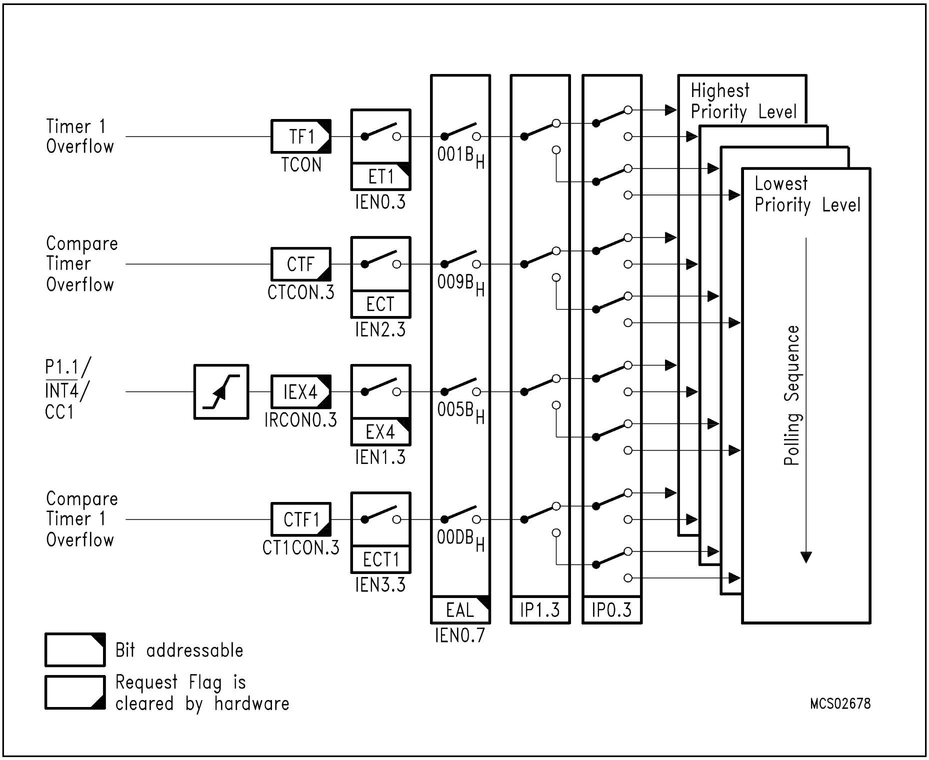


Figure 7-3 :
Interrupt Structure, Overview Part 3

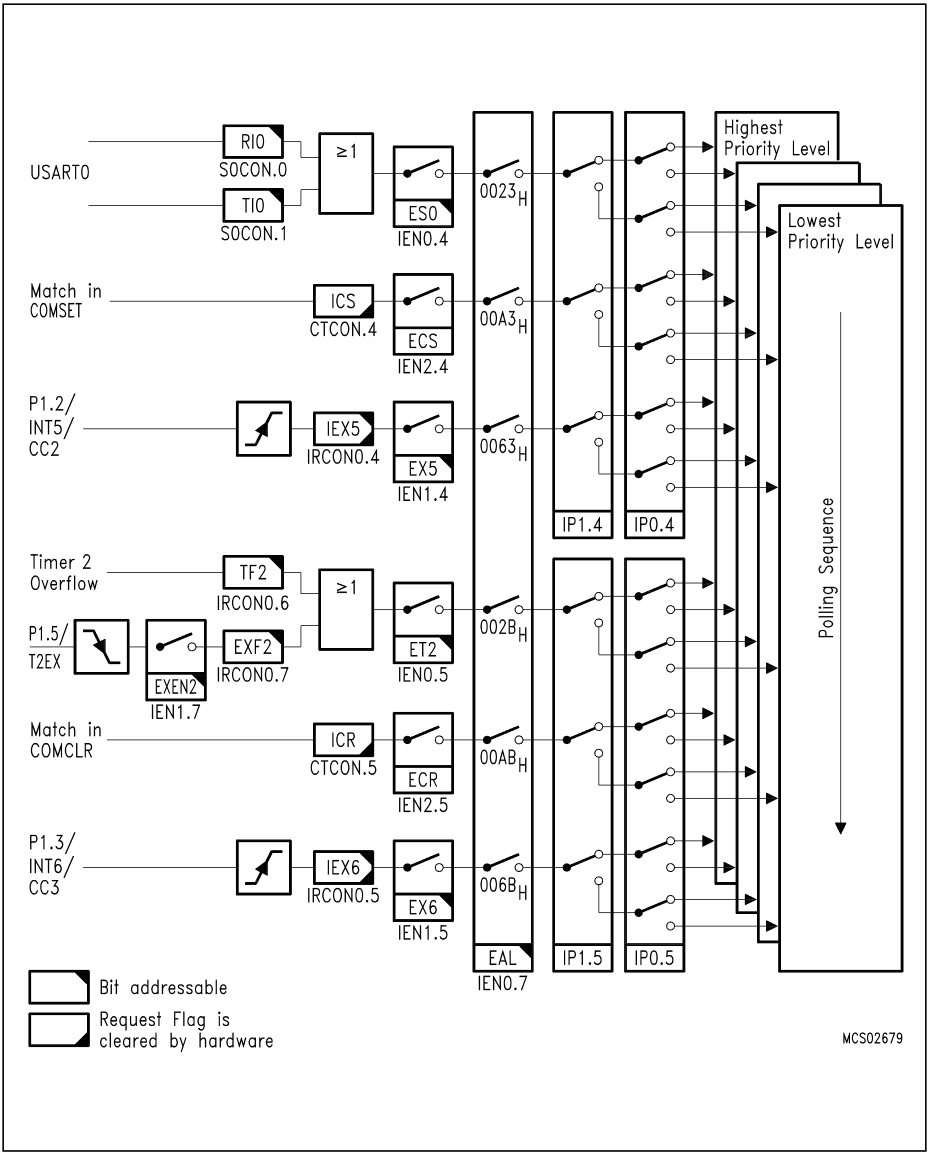


Figure 7-4 :
Interrupt Structure, Overview Part 4

7.2 Interrupt Registers

7.2.1 Interrupt Enable Registers

Each interrupt vector can be individually enabled or disabled by setting or clearing the corresponding bit in the interrupt enable registers IEN0, IEN1, IEN2. and IEN3. Register IEN0 also contains the global disable bit (EAL), which can be cleared to disable all interrupts at once. Some interrupts sources have further enable bits (e.g. EXEN2). Such interrupt enable bits are controlled by specific bits in the SFRs of the corresponding peripheral units (described in **chapter 6**). This section describes the locations and meanings of these interrupt enable bits in detail.

After reset the enable bits of the interrupt enable registers IEN0 to IEN3 are set to 0. That means that the corresponding interrupts are disabled.

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

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

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

The shaded bit is not used for interrupt control

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

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

Special Function Register IEN1 (Address B8$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	BF$_H$	BE$_H$	BD$_H$	BC$_H$	BB$_H$	BA$_H$	B9$_H$	B8$_H$	
B8$_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	IEN1

The shaded bit is not used for interrupt control

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

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

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

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

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

The SFR IEN3 includes the enable bits for the compare timer 1 overflow interrupt and the general capture/compare interrupt of the compare timer 1.

compare match with compare register interrupts, the compare, and the serial interface 1 interrupt.

Special Function Register IEN3 (Address BE$_H$) **Reset Value : XXXX00XX$_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
BE$_H$	–	–	–	–	ECT1	ECC1	–	–	IEN3

Bit	Function
–	Reserved bits for future use.
ECT1	Compare timer 1 overflow interrupt enable If ECT1 = 0, the interrupt at compare timer 1 overflow is disabled.
ECC1	Compare timer 1, general capture/compare interrupt enable This bit enables the interrupt on an capture/compare event in the capture/compare registers CC10 - CC17. Additionally, the SFR EICC1 must be programmed to enable the interrupt request. With ECC1=0, the general capture/compare timer 1 interrupt is disabled.

The SFR EICC1 includes the enable bits for the specific capture/compare interrupt of the compare timer 1.

Special Function Register EICC1 (Mapped Address BF$_H$) **Reset Value : FF$_H$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
BF$_H$	EICC17	EICC16	EICC15	EICC14	EICC13	EICC12	EICC11	EICC10	EICC1

Bit	Function
EICC17 - EICC10	Compare timer 1, specific capture/compare interrupt enable This bit enables the interrupt on an capture/compare event in the capture/compare registers CC10 - CC17. When EICC1x=0, the interrupt request flag ICC1x has no effect on the interrupt unit. EICC17 refers to CC17 etc.

EICC1 is mapped to the SFR IRCON2. It is selected by a double instruction sequence with PDIR set.

7.2.2 Interrupt Priority Registers

The 19 interrupt sources of the C509-L are combined to six interrupt groups (see **table 7-1**). Each of the six interrupt groups can be programmed to one of four priority levels by setting or clearing a bit in the IP0 and IP1 priority registers. Further details about the interrupt priority structure are given in **chapter 7.3**.

Special Function Register IP0 (Address A9$_H$) **Reset Value : 00$_H$**
Special Function Register IP1 (Address B9$_H$) **Reset Value : 0X000000$_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
A9$_H$	OWDS	WDTS	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0	IP0
	7	6	5	4	3	2	1	0	
B9$_H$	PDIR	–	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0	IP1

The shaded bits are not used for interrupt purposes.

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>IP1.x
IP1.x</td><td>Interrupt Priority level bits (x=0-5)
<table>
<tr><th>IP1.x</th><th>IP0.x</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>Interrupt group x is set to priority level 0 (lowest)</td></tr>
<tr><td>0</td><td>1</td><td>Interrupt group x is set to priority level 1</td></tr>
<tr><td>1</td><td>0</td><td>Interrupt group x is set to priority level 2</td></tr>
<tr><td>1</td><td>1</td><td>Interrupt group x is set to priority level 3 (highest)</td></tr>
</table>
</td></tr>
</table>

7.2.3 Interrupt Request Flags

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

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

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

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

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

The shaded bits are not used for interrupt purposes.

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

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

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

Special Function Register S0CON (Address 98_H) **Reset Value : 00_H**
Special Function Register S1CON (Address $9B_H$) **Reset Value : 01000000_B**

	MSB							LSB	
Bit No.	$9F_H$	$9E_H$	$9D_H$	$9C_H$	$9B_H$	$9A_H$	99_H	98_H	
98_H	SM0	SM10	SM20	REN0	TB80	RB80	TI0	RI0	S0CON
	7	6	5	4	3	2	1	0	
$9B_H$	SM	S1P	SM21	REN1	TB81	RB81	TI1	RI1	S1CON

The shaded bits are not used for interrupt purposes.

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

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

Like the external interrupt 2, **the external interrupt 3** can be either positive or negative transition-activated, depending on bit I3FR in register T2CON. The flag that actually generates this interrupt is bit IEX3 in register IRCON. In addition, this flag will be set if a compare event occurs at pin P1.0/$\overline{\text{INT3}}$/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 too.

The external interrupts 4 (INT4), 5 (INT5), 6 (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 P1.1/INT4/CC1, P1.2/INT5/CC2, and P1.3/INT6/CC3, regardless of the compare mode established and the transition at the respective pin. When an interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored too.

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

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

Special Function Register T2CON (Address C8$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	CF$_H$	CE$_H$	CD$_H$	CC$_H$	CB$_H$	CA$_H$	C9$_H$	C8$_H$	
C8$_H$	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T1I0	T2CON

The shaded bits are not used for interrupt purposes.

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

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

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

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

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

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

Special Function Register IRCON1 (Address D1$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
D1$_H$	ICMP7	ICMP6	ICMP5	ICMP4	ICMP3	ICMP2	ICMP1	ICMP0	IRCON1

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

The compare timer 1 match interrupt occurs on a compare match of the CC10 to CC17 registers with the compare timer 1 or (if selected) at a capture event at the pins CC10 to CC17. There are 8 compare match/capture interrupt flags available in SFR IRCON2 which are or-ed together for a single interrupt request. Thus, a compare match/capture interrupt service routine has to check which compare match/capture event has occurred. The ICC1x flags must be cleared by software.

Special Function Register IRCON2 (Address BF$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
BF$_H$	ICC17	ICC16	ICC15	ICC14	ICC13	ICC12	ICC11	ICC10	IRCON2

Bit	Function
ICC17 - 10	Compare timer 1 match with register CC17 - CC10 interrupt flags ICC1x is set by hardware when a compare match of the compare timer 1 with the compare register CC1x occurs but only if the compare function for CC1x has been enabled. ICC1x must be cleared by software.

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

The compare timer 1 interrupt is generated by bit CT1F in register CT1CON, which is set by a rollover in the compare timer. If a compare timer interrupt is generated, flag CT1F can be cleared by software.

Special Function Register CTCON (Address. E1$_H$) **Reset Value : 01000000$_B$**
Special Function Register CT1CON (Address BC$_H$) **Reset Value : X1XX0000$_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
E1$_H$	T2PS1	CTP	ICR	ICS	CTF	CLK2	CLK1	CLK0	CTCON

	7	6	5	4	3	2	1	0	
BC$_H$	–	CT1P	–	–	CT1F	CLK12	CLK11	CLK10	CT1CON

The shaded bits are not used for interrupt purposes.

Bit	Function
CTF	Compare timer overflow flag CTF is set when the compare timer 1 count rolls over from all ones to the reload value. When CTF is set, a compare timer interrupt can be generated (if enabled). CTF is cleared by hardware when the compare timer value is no more equal to the reload value.
CT1F	Compare timer 1 overflow flag CT1F is set when the compare timer 1 count rolls over from all ones to the reload value. When CT1F is set, a compare timer 1 interrupt can be generated (if enabled). CT1F is cleared by hardware when the compare timer 1 value is no more equal its reload value.

7.3 Interrupt Priority Level Structure

The 19 interrupt sources of the C509-L are combined in six groups. **Table 7-1** lists the structure of these interrupt groups.

Table 7-1
Interrupt Source Structure

Interrupt Groups	Associated Interrupts			
1	External interrupt 0	Serial channel 1 interrupt	A/D converter interrupt	--
2	Timer 0 overflow	--	External interrupt 2	--
3	External interrupt 1	Match in CM0-CM7	External interrupt 3	Match in CC10-CC17 or capture event at port 9
4	Timer 1 overflow	Compare timer overflow	External interrupt 4	Compare timer 1 overflow
5	Serial channel 0 interrupt	Match in COMSET	External interrupt 5	--
6	Timer 2 overflow or external reload interrupt	Match in COMCLR	External interrupt 6	--

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

If two or more requests of different priority 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 (**table 7-2**).

Table 7-2
Interrupts - Priority-within-Level

Interrupt Group	Priority Bits of Interrupt Group	Interrupt Source Priority				Interrupt Group Priority
		High Priority	→		Low Priority	
1	IP1.0 / IP0.0	IE0	RI1 + TI1	IADC	–	High
2	IP1.1 / IP0.1	TF0	–	IEX2	–	↓
3	IP1.2 / IP0.0	IE0	ICMP0-7	IEX3	ICC10-17	
4	IP1.3 / IP0.3	TF1	CTF	IEX4	CTF1	
5	IP1.4 / IP0.4	RI0 + TI0	ICS	IEX5	–	
6	IP1.5 / IP0.5	TF2 + EXF2	ICR	IEX6	–	Low

Within one pair or triplet the leftmost interrupt is serviced first, then the second and third, when available. The interrupt groups are serviced from top to bottom of the table. A low-priority interrupt can itself be interrupted by a higher-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 which is illustrated in **table 7-2**.

The "priority-within-level" structure is only used to resolve simultaneous requests of the same priority level.

7.4 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.
1) The current (polling) cycle is not in the final cycle of the instruction in progress.
1) The instruction in progress is RETI or any write access to registers IEN0, IEN1, IEN2, IEN3, EICC1, IP1 or IP0.

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 IE or IP, then at least one more instruction will be executed before any interrupt is vectored too; 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 7-5**.

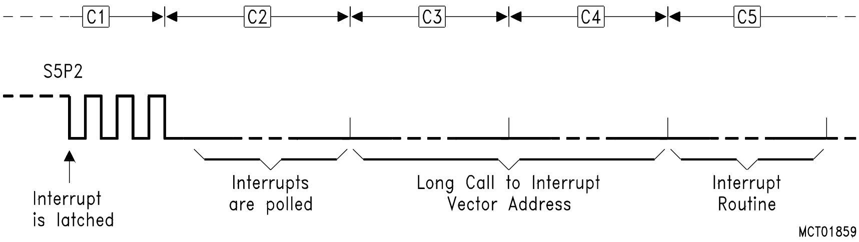


Figure 7-5
Interrupt Response Timing Diagram

Note that if an interrupt of a higher priority level goes active prior to S5P2 in the machine cycle labeled C3 in **figure 7-5** 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 too, as shown in **table 7-3**.

Table 7-3
Interrupt Source and Vectors

Interrupt Source	Interrupt Vector Address	Interrupt Request Flags
External Interrupt 0	0003_H	IE0
Timer 0 Overflow	$000B_H$	TF0
External Interrupt 1	0013_H	IE1
Timer 1 Overflow	$001B_H$	TF1
Serial Channel 0	0023_H	RI0 / TI0
Timer 2 Overflow / Ext. Reload	$002B_H$	TF2 / EXF2
A/D Converter	0043_H	IADC
External Interrupt 2	$004B_H$	IEX2
External Interrupt 3	0053_H	IEX3
External Interrupt 4	$005B_H$	IEX4
External Interrupt 5	0063_H	IEX5
External Interrupt 6	$006B_H$	IEX6
Serial Channel 1	0083_H	RI1 / TI1
Compare Match Interrupt of Compare Registers CM0-CM7 assigned to Timer 2	0093_H	ICMP0 - ICMP7
Compare Timer Overflow	$009B_H$	CTF
Compare Match Interrupt of Compare Register COMSET	$00A3_H$	ICS
Compare Match Interrupt of Compare Register COMCLR	$00AB_H$	ICR
Compare / Capture Event interrupt	$00D3_H$	ICC10 - ICC17
Compare Timer 1 Overflow	$00DB_H$	CT1F

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.

7.5 External Interrupts

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

If the external interrupt 0 or 1 is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

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

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

Since the external interrupt pins ($\overline{\text{INT2}}$ to INT6) are sampled once in each machine cycle, an input high or low should be held for at least 12 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin low (high for $\overline{\text{INT2}}$ and $\overline{\text{INT3}}$, if it is programmed to be negative transition-active) for at least one cycle, and then hold it high (low) for at least one cycle to ensure that the transition is recognized so that the corresponding interrupt request flag will be set (see **figure 7-6**). The external interrupt request flags will automatically be cleared by the CPU when the service routine is called.

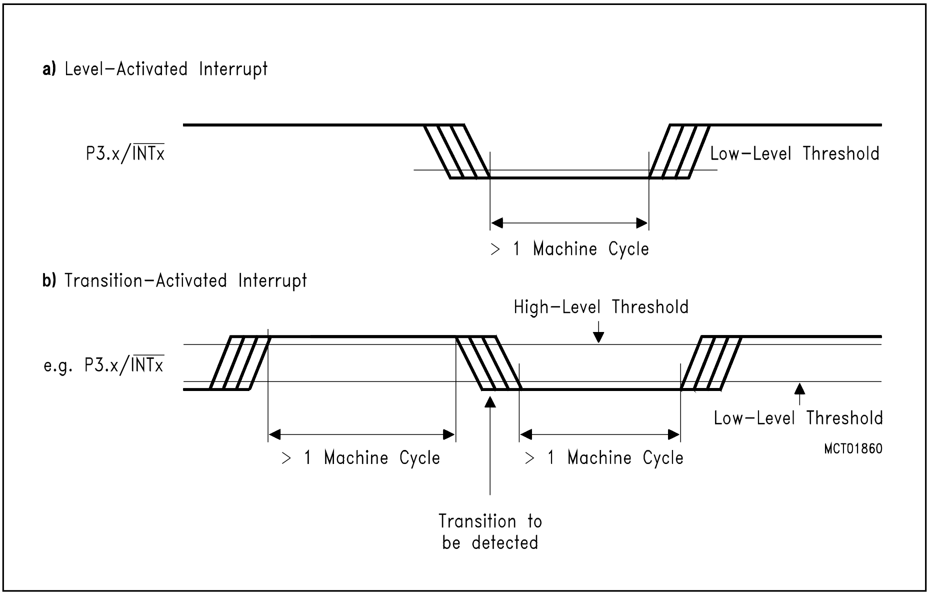


Figure 7-6
External Interrupt Detection

7.6 Interrupt Response Time

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

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

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

8 Fail Save Mechanisms

The C509-L offers two on-chip peripherals which monitor the program flow and ensure an automatic “fail-safe” reaction for cases where the controller’s hardware fails or the software hangs up:

– A programmable watchdog timer (WDT) with variable time-out period from 189 microseconds up to approx. 0.79 seconds at 16 MHz.

– An oscillator watchdog (OWD) which monitors the on-chip oscillator and forces the microcontroller into the reset state if the on-chip oscillator fails.

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 C509-L is a 15-bit timer, which is incremented by a count rate of $f_{OSC}/12$ up to $f_{OSC}/384$. For programming of the watchdog timer overflow rate, the upper 7 bit of the watchdog timer can be written. **Figure 8-1** shows the block diagram of the watchdog timer unit.

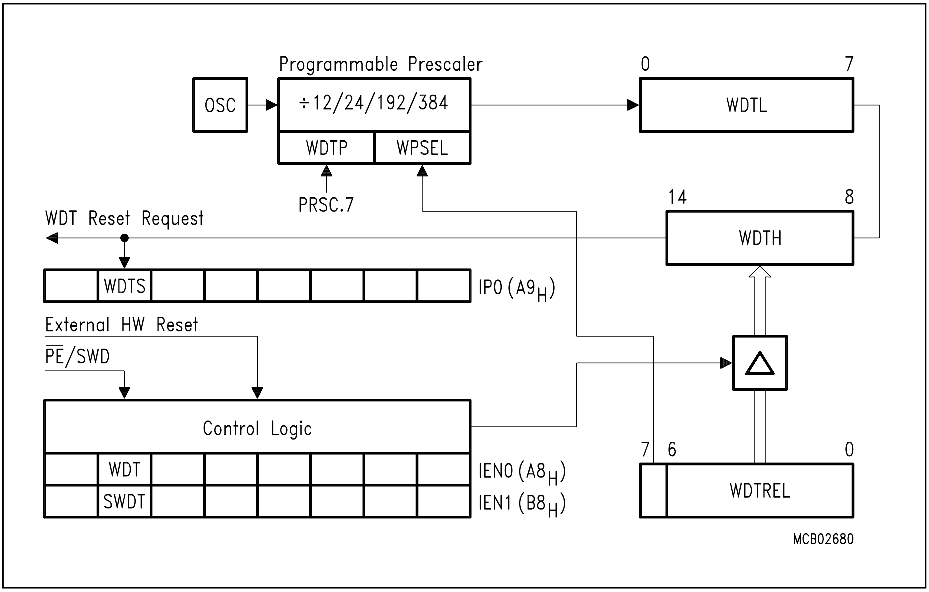


Figure 8-1
Block Diagram of the Programmable Watchdog Timer

8.1.1 Input Clock Selection

The input clock rate of the watchdog timer is derived from the system clock of the C509-L. There are two prescalers which define the input clock rate. These prescalers are controlled by two bits in the SFRs PRSC and WDTREL. **Table 8-1** shows the resulting timeout periods at f_{OSC} = 16 MHz.

Special Function Register PRSC (Address $B4_H$) Reset Value : 11010101_B
Special Function Register WDTREL (Address 86_H) Reset Value : 00_H

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
$B4_H$	WDTP	S0P	T2P1	T2P0	T1P1	T1P2	T0P1	T0P0	PRSC

	7	6	5	4	3	2	1	0	
86_H	WPSEL	Watchdog timer reload value							WDTREL

The shaded bits are not used for the watchdog timer.

<table>
<tr><th>Bit</th><th>Function</th></tr>
<tr><td>WDTP
WPSEL</td><td>Prescaler select bits for the watchdog input clock
The two control bits WDTP and WPSEL define the input clock frequency f_{IN} of the watchdog timer.
<table>
<tr><th>WDTP</th><th>WPSEL</th><th>Input clock</th></tr>
<tr><td>0</td><td>0</td><td>$f_{IN} = f_{OSC}/12$</td></tr>
<tr><td>0</td><td>1</td><td>$f_{IN} = f_{OSC}/192$</td></tr>
<tr><td>1</td><td>0</td><td>$f_{IN} = f_{OSC}/24$ (reset value)</td></tr>
<tr><td>1</td><td>1</td><td>$f_{IN} = f_{OSC}/384$</td></tr>
</table></td></tr>
<tr><td>WDTREL.6-0</td><td>Watchdog timer reload value
Seven bit reload value for the high-byte of the watchdog timer. This value is loaded to the WDT when a refresh is triggered by a consecutive setting of bits WDT and SWDT.</td></tr>
</table>

Table 8-1
Watchdog Timer Timeout Periods at f_{osc} = 16 MHz

WDTREL	Time-Out Periods				Comments
	$f_{osc}/12$	$f_{osc}/24$	$f_{osc}/192$	$f_{osc}/384$	
00_H	24.6 ms	49.1 ms	393 ms	786 ms	Maximum time period (default after reset) (128 WDTL overflows)
$7E_H$	381 µs	762 µs	6.10 ms	12.2 ms	Two WDTL overflows
$7F_H$	189 µs	378 µs	3.02 ms	6.05 ms	Min. time period (One WDTL overflow)

8.1.2 Watchdog Timer Control Flags

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

Special Function Register IEN0 (Address A8$_H$) **Reset Value : 00$_H$**
Special Function Register IEN1 (Address B8$_H$) **Reset Value : 00$_H$**
Special Function Register IP0 (Address A9$_H$) **Reset Value : 00$_H$**

	MSB							LSB	
Bit No.	AF$_H$	AE$_H$	AD$_H$	AC$_H$	AB$_H$	AA$_H$	A9$_H$	A8$_H$	
A8$_H$	EAL	WDT	ET2	ES0	ET1	EX1	ET0	EX0	IEN0
	BF$_H$	BE$_H$	BD$_H$	BC$_H$	BB$_H$	BA2$_H$	B91$_H$	B8$_H$	
B8$_H$	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	IEN1
	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 in controlling the watchdog timer.

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 occurred. Can be cleared or set by software

8.1.3 Starting the Watchdog Timer

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

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

8.1.3.1 The First Possibility of Starting the Watchdog Timer

The automatic start of the watchdog timer directly after an external HW reset is a hardware start initialized by strapping pin PE/SWD to V_{CC}. In this case the power-saving modes (power-down mode, idle mode and slow-down mode) are also disabled and cannot be started by software.

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

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

8.1.3.2 The Second Possibility of Starting the Watchdog Timer

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

Note that once the watchdog timer has been started it cannot be stopped by anything but an external hardware reset through pin $\overline{\text{RESET}}$ with a low level applied to pin PE/SWD.

8.1.4 Refreshing the Watchdog Timer

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

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

8.1.5 Watchdog Reset and Watchdog Status Flag

If the software fails to clear the watchdog in time, an internally generated watchdog reset is entered at the counter state 7FFC$_H$. The duration of the reset signal then depends on the prescaler selection (either 4, 8, 64, 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 special function register IP0) is set. **Figure 8-2** shows a block diagram of all reset requests in the C509-L and the function of the watchdog status flags. The WDTS flag is a flip-flop, which is set by a watchdog timer reset and cleared by an external HW reset. Bit WDTS allows the software to examine from which source the reset was activated. The watchdog timer status flag can also be cleared by software.

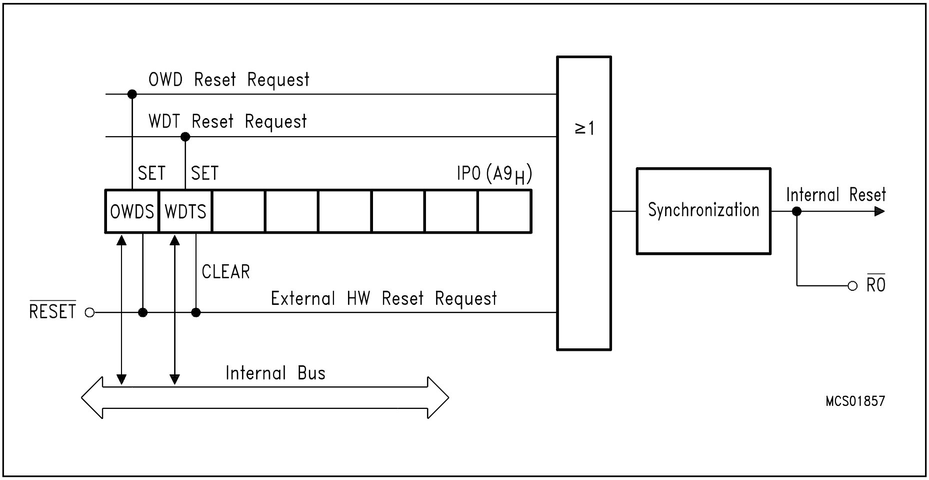


Figure 8-2
Watchdog Timer Status Flags and Reset Requests

8.2 Oscillator Watchdog

The C509-L has a oscillator watchdog unit that is fully compatible with the SAB 80C517A's oscillator watchdog.

The oscillator watchdog unit serves 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 executes a final reset phase of appr. 0.5 ms in order to allow the oscillator to stabilize; then the oscillator watchdog reset is released and the part starts program execution again.
- Restart from the hardware power down mode.
  If the hardware power down mode is terminated the oscillator watchdog has to control the correct start-up of the on-chip oscillator and to restart the program. The oscillator watchdog function is only part of the complete hardware power down sequence; however, the watchdog works identically to the monitoring function.
- Fast internal reset after power-on.
  In this function the oscillator watchdog unit provides a clock supply for the reset before the on-chip oscillator has started. In this case the oscillator watchdog unit also works identically to the monitoring function.

If the oscillator watchdog unit shall be used it must be enabled (this is done by applying high level to the control pin OWE).

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. The RC oscillator can be enabled/disabled by the control pin OWE. If it is disabled the complete unit has no function.

The SFR IP0 contains a status flag of the oscillator watchdog as shown below.

Special Function Register IP0 (Address A9$_H$) **Reset Value : 00$_H$**

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 in controlling the oscillator watchdog.

Bit	Function
OWDS	Oscillator watchdog timer status flag Set by hardware when an oscillator watchdog reset occurred. Can be cleared or set by software.

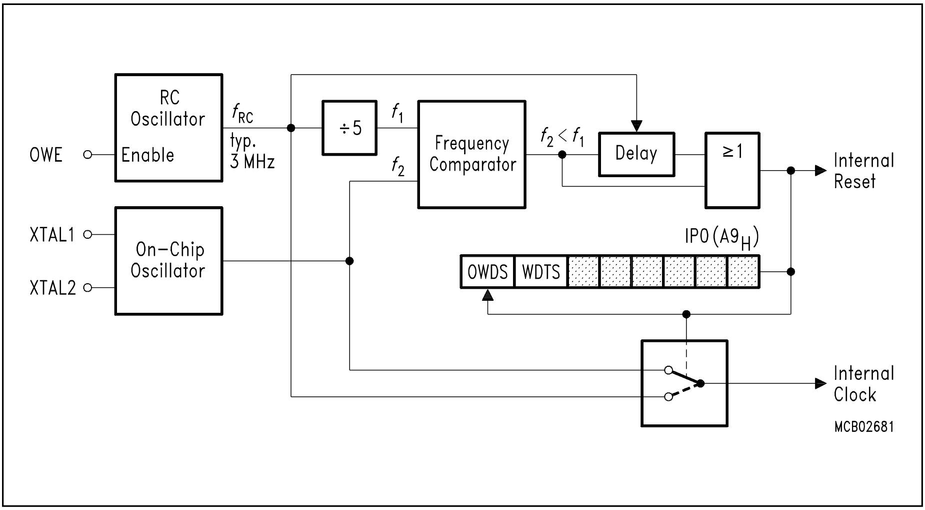


Figure 8-3
Functional Block Diagram of the Oscillator Watchdog

The frequency coming from the RC oscillator is divided by 5 and compared to the on-chip oscillator's frequency. If the frequency coming from the on-chip oscillator is found lower than the frequency derived from the RC oscillator the watchdog detects a failure condition (the oscillation at the on-chip oscillator could stop because of crystal damage etc.). In this case it switches the input of the internal clock system to the output of the RC oscillator. This means that the part is being clocked even if the on-chip oscillator has stopped or has not yet started. At the same time the watchdog activates the internal reset in order to bring the part in its defined reset state. The reset is performed because clock is available from the RC oscillator. This internal watchdog reset has the same effects as an externally applied reset signal with the following exception: The watchdog timer status flag WDTS (IP0.6) is not reset (the Watchdog Timer however is stopped) and bit OWDS is set. This allows the software to examine error conditions detected by the watchdog timer even if meanwhile an oscillator failure occurred.

The oscillator watchdog is able to detect a recovery of the on-chip oscillator after a failure. If the frequency derived from the on-chip oscillator is again higher than the reference the watchdog starts a final reset sequence which takes typ. 1 ms. Within that time the clock is still supplied by the RC oscillator and the part is held in reset. This allows a reliable stabilization of the on chip oscillator. After that, the watchdog toggles the clock supply back to the on-chip oscillator and releases the reset request. If no external reset is applied in this moment the part will start program execution. If an external reset is active, however, the device will keep the reset state until also the external reset request disappears.

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

9 Power Saving Modes

The C509-L provides three modes in which power consumption can be significantly reduced.

– Idle mode
  The CPU is gated off from the oscillator. All peripherals are still provided with the clock and are able to work.
– Power down mode
  The operation of the C509-L is completely stopped and the oscillator is turned off. This mode is used to save the contents of the internal RAM with a very low standby current. Power down mode can be entered by software or by hardware.
– Slow-down mode
  The controller keeps up the full operating functionality, but its normal clock frequency is internally divided by eight. This slows down all parts of the controller, the CPU and all peripherals, to 1/8 th of their normal operating frequency. Slowing down the frequency greatly reduces power consumption.

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

These power saving modes, especially the power down mode, replace the hardware power down supply for the internal RAM via a dedicated pin. During the power saving modes, the power supply for the C509-L are all V_{CC} pins. There is no further dedicated pin for power down supply.

For the C509-L several provisions have been made to quality it for both electrically noisy environments and applications requiring high system security. In such applications unintentional entering of the power saving modes must be absolutely avoided. A power saving mode would reduce the controller’s performance (in the case of slow-down mode) or even stop any operation (in the case of power down mode). This situation might be fatal for the system, which is controlled by the microcontroller. Such critical applications often use the watchdog timer to prevent the system from program upsets. Then, an accidental entering of the power saving modes would even stop the watchdog timer and would circumvent the watchdog timer’s task of system protection.

9.1 Hardware Enable for the Use of the Power Saving Modes

To provide power saving modes together with effective protection against unintentional entering of these modes, the C509-L has an extra pin disabling the use of the power saving modes. As this pin will most likely be used only in critical applications it is combined with an automatic start of the watchdog timer (see the description in **section 7.8** “Fail Save Mechanisms”). This pin is called PE/SWD (power saving enable/start watchdog timer) and its function is as follows:

PE/SWD = 1 (logic high level)

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

PE/SWD = 0 (logic low level)

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

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

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

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

Application Example for Switching Pin PE/SWD

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

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

9.2 Power Saving Mode Control Register PCON

The SFR PCON is used for control of the power saving modes. It also contains two general purpose flags, which can be used e.g. by software for power saving mode management.

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

Bit No.	MSB							LSB	
	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 power saving mode control.

Bit	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 IDSL flag bit is the last instruction before entering the idle mode.
SD	Slow down 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 the power down mode is enabled.
IDLE	Idle mode enable bit When set, starting the idle mode is enabled.

9.3 Idle Mode

In idle mode the oscillator of the C509-L continues to run, but the CPU is gated off from the clock signal. However, the interrupt system, the serial channels, the A/D converter, the oscillator watchdog, the division/multiplication unit and all timers, except for 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 division/multiplication unit are not running, maximum power reduction can be achieved. This state is also the test condition for the idle I_{CC} in the DC characteristics.

Thus, the user has to take into account that the right peripheral continues to run or is stopped, respectively, during idle. 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.

Normally the port pins hold the logical state they had at the time idle was activated. If some pins are programmed to serve their alternate functions they still continue to output during idle if the assigned function is on. This applies for the compare outputs as well as for the system clock output signal and the serial interface in case the latter could not finish reception or transmission during normal operation. The control signals ALE and PSEN are held at logic high levels (see **table 9-1**).

During idle, as in normal operating mode, the ports can be used as inputs. Thus, a capture or reload operation as well as an A/D conversion can be triggered, the timers can be used to count external events and external interrupts can be detected.

Table 9-1
Status of External Pins During Idle and Software Power Down Mode

Pins	Idle Mode	Software Power Down Mode
ALE	High	Low
$\overline{\text{PSEN}}$	High	Low
Port 0	Float	Float
Port 1	Data / alternate outputs	Data / last output
Port 2	Address	Data
Port 3	Data / alternate outputs	Data / last output
Port 4	Data / alternate outputs	Data / last output
Port 5	Data / alternate outputs	Data / last output
Port 6	Data / alternate outputs	Data / last output
Port 9	Data / alternate outputs	Data / last output

The watchdog timer is the only peripheral which is automatically stopped during idle mode. The idle mode makes it possible to “freeze” the processor’s status for a certain time or until an external event causes the controller to go back into normal operating mode. Since the watchdog timer is stopped during idle mode, this useful feature of the C509-L is provided even if the watchdog function is used simultaneously.

If the idle mode is to be used the pin PE/SWD must be held low. Entering the idle mode is to be done by two consecutive instructions immediately following each other. The first instruction has to set the flag bit IDLE (PCON.0) and must not set bit IDLS (PCON.5), the following instruction has to set 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 will not initiate the idle mode. Bits IDLE and IDLS will automatically be cleared after having been set. If one of these register bits is read the value shown is zero (0). This double-instruction sequence is implemented to minimize the chance of unintentionally entering the idle mode.

Note that PCON is not a bit-addressable register, so the above mentioned sequence for entering the idle mode is to be done by byte handling instructions.

The following instruction sequence may serve as an example:

```
– ORLPCON,#00000001B;Set bit IDLE, bit IDLS must not be set
  ORLPCON,#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.

Termination of the Idle 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 set the bit IDLS.
- The other possibility of terminating the idle mode is a hardware reset. Since the oscillator is still running, the hardware reset is held active for only two machine cycles for a complete reset.

9.4 Software Power Down Mode

In the software power down mode, the on-chip oscillator is stopped. Therefore, all functions are stopped, only the contents of the on-chip RAM and the SFR's are held. The port pins 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; when enabled, the clockout signal (P1.6/CLKOUT) will stop at low level. ALE and PSEN are held at logic low level (see **table 9-1**).

lf the software power down mode is to be used, the pin PE/SWD must be held low. Entering the software power down mode is done by two consecutive instructions immediately following each other. 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 and must not set bit PDE. The hardware ensures that a concurrent setting of both bits, PDE and PDS, will not initiate the power down mode. Bit PDE and PDS will automatically be cleared after having been set and the value shown when reading one of these bits is always zero (0). This double-instruction sequence is implemented to minimize the chance of unintentional entering the software power down mode, which could possibly “freeze” the chip's activity in an undesired status.

Note that PCON is not a bit-addressable register, so the above mentioned sequence for entering the software power down mode is composed of byte handling instructions.

The following instruction sequence may serve as an 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
```

The instruction that sets bit PDS is the last instruction executed before going into power down mode. lf idle mode and software power down mode are invoked simultaneously, the software power down mode takes precedence.

The only exit from software power down mode is a hardware reset. Reset will redefine all SFR's, but will not change the contents of the internal RAM.

In the software power down mode, V_{CC} can be reduced to minimize power consumption. Care must be taken, however, to ensure that V_{CC} is not reduced before the software power down mode is invoked, and that V_{CC} is restored to its normal operating level before software the power down mode is terminated. The reset signal that terminates the software power down mode also frees the oscillator. The reset 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).

9.5 Slow Down Mode

In some applications, where power consumption and dissipation is critical, the controller might run for a certain time at reduced speed (e.g. if the controller is waiting for an input signal). Since in CMOS devices there is an almost linear interdependence 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 eight. This also includes the clockout signal at pin P1.6/CLKOUT.

lf the slow down mode is to be used the pin PE/SWD must be held low.

The slow down mode is entered by setting bit SD (PCON.4). The controller actually enters the slow down mode after a short synchronization period (max. two machine cycles). The slow down mode can be used together with idle and power down mode.

The slow down mode is disabled by clearing bit SD.

9.6 Hardware Power Down Mode

The C509-L has also a hardware power down mode. This mode can be initiated by an external signal at the pin HWPD. This mode is referenced as hardware power down mode in opposite to the program controlled software power down mode.

For a correct function of the hardware power down mode the oscillator watchdog unit including its internal RC oscillator is needed. Therefore this unit must be enabled by pin OWE (OWE = high), if the hardware power down mode shall be used. However, the control pin PE/SWD has no control function for the hardware power down mode; it enables and disables only the use of all software controlled power saving modes (slow down mode, idle mode, software power down mode).

The function of the hardware power down mode is as follows:

– The pin HWPD controls this mode. If it is on logic high level (inactive) the part is running in the normal operating modes. If pin HWPD gets active (low level) the part enters the hardware power down mode; as mentioned above this is independent of the state of pin PE/SWD.

HWPD is sampled once per machine cycle. If it is found active, the device starts a complete internal reset sequence. This takes two machine cycles; all pins have their default reset states during this time. This reset has exactly the same effects as a hardware reset; i.e.especially the watchdog timer is stopped and its status flag WDTS is cleared. In this phase the power consumption is not yet reduced. After completion of the internal reset both oscillators of the chip are disabled, the on-chip oscillator as well as the oscillator watchdog's RC oscillator. At the same time the port pins and several control lines enter a floating state as shown in **table 9-2**. In this state the power consumption is reduced to the power down current IPD. Also the supply voltage can be reduced.
Table 9-2 also lists the voltages which may be applied at the pins during hardware power down mode without affecting the low power consumption.

Table 9-2
Status of all Pins During Hardware Power Down Mode

Pins	Status	Voltage Range at Pin During HW-Power Down
P0, P1, P2, P3, P4, P5, P6, P7, P8, P9	Floating outputs / disabled input function	$V_{SS} \leq V_{IN} \leq V_{CC}$
$\overline{\text{EA}}$	active input	$V_{IN} = V_{CC}$ or $V_{IN} = V_{SS}$
PE/SWD	active input, pull-up resistor disabled during HW power down	$V_{IN} = V_{CC}$ or $V_{IN} = V_{SS}$
XTAL1	active output	pin may not be driven
XTAL2	disabled input function	$V_{SS} \leq V_{IN} \leq V_{CC}$
PSEN/RDF, ALE	Floating outputs / disabled input function (for test modes only)	$V_{SS} \leq V_{IN} \leq V_{CC}$
VAREF, VAGND	active supply pins	$V_{AGnd} \leq V_{IN} \leq V_{CC}$
OWE	active input; must be at high level for start-up after HW power down; pull up resistor disabled during HW-power down	$V_{IN} = V_{CC}$ or ($V_{IN} = V_{SS}$)
$\overline{\text{RESET}}$	active input; must be on high level if HW power down is used	$V_{IN} = V_{CC}$
$\overline{\text{R0}}$	Floating output	$V_{SS} \leq V_{IN} \leq V_{CC}$

The power down state is maintained while pin HWPD is held active. If $\overline{\text{HWPD}}$ goes to high level (inactive state) an automatic start up procedure is performed:

- First the pins leave their floating condition and enter their default reset state as they had immediately before going to float state.
- Both oscillators are enabled (only if OWE = high). While the on-chip oscillator (with pins XTAL1 and XTAL2) usually needs a longer time for start-up, if not externally driven (with crystal approx. 1 ms), the oscillator watchdog's RC oscillator has a very short start-up time (typ. less than 2 microseconds).
- Because the oscillator watchdog is active it detects a failure condition if the on-chip oscillator hasn't yet started. Hence, the watchdog keeps the part in reset and supplies the internal clock from the RC oscillator.
- Finally, when the on-chip oscillator has started, the oscillator watchdog releases the part from reset after it performed a final internal reset sequence and switches the clock supply to the on-chip oscillator. This is exactly the same procedure as when the oscillator watchdog detects first a failure and then a recovering of the oscillator during normal operation. Therefore, also the oscillator watchdog status flag is set after restart from hardware power down mode.
  When automatic start of the watchdog was enabled (PE/SWD connected to V_{CC}), the Watchdog Timer will start, too (with its default reload value for time-out period).

The SWD-Function of the PE/SWD Pin is sampled only by a hardware reset. Therefore at least one Power On Reset has to be performed.

9.7 Hardware Power Down Mode Reset Timing

The following figures show timing diagrams for entering (**figure 8-1**) and leaving (**figure 8-2**) the hardware power down mode. If there is only a short signal at pin HWPD (i.e. HWPD is sampled active only once), then a complete internal reset is executed. Afterwards the normal program execution starts again (**figure 8-3**).

Note: Delay time caused by internal logic is not included.

The RESET pin overrides the hardware power down function, i.e. if RESET becomes active during hardware power down it is terminated and the device performs the normal reset function. Thus, pin RESET has to be inactive during hardware power down mode.

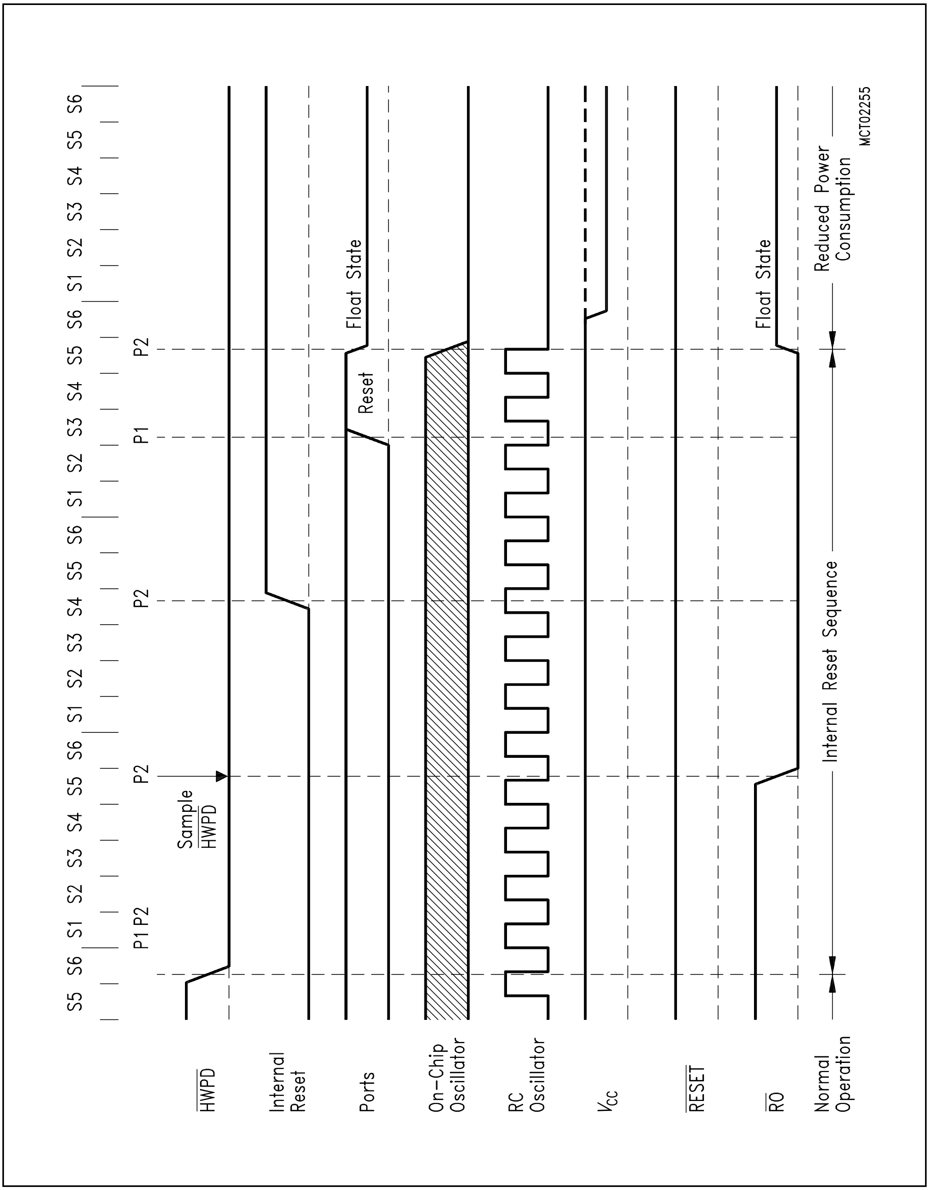


Figure 9-1
Timing Diagram of Entering Hardware Power Down Mode

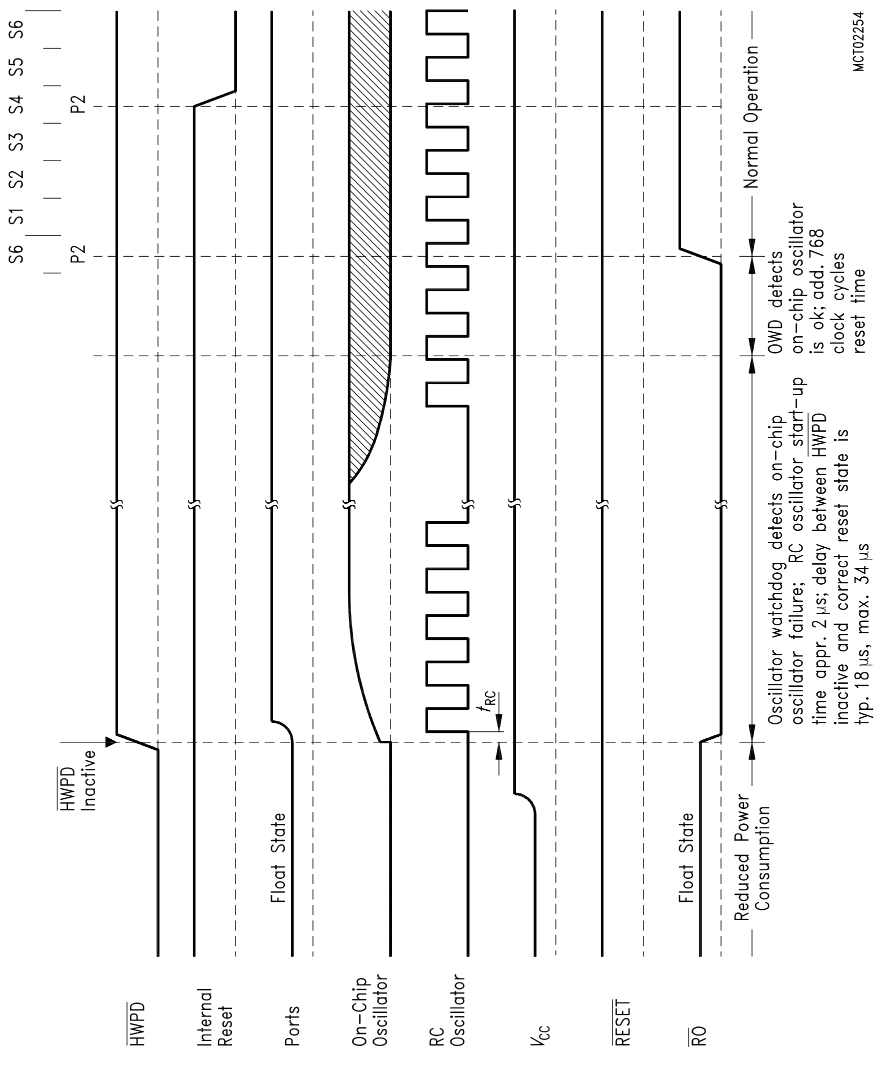


Figure 9-2
Timing Diagram of Leaving Hardware Power Down Mode

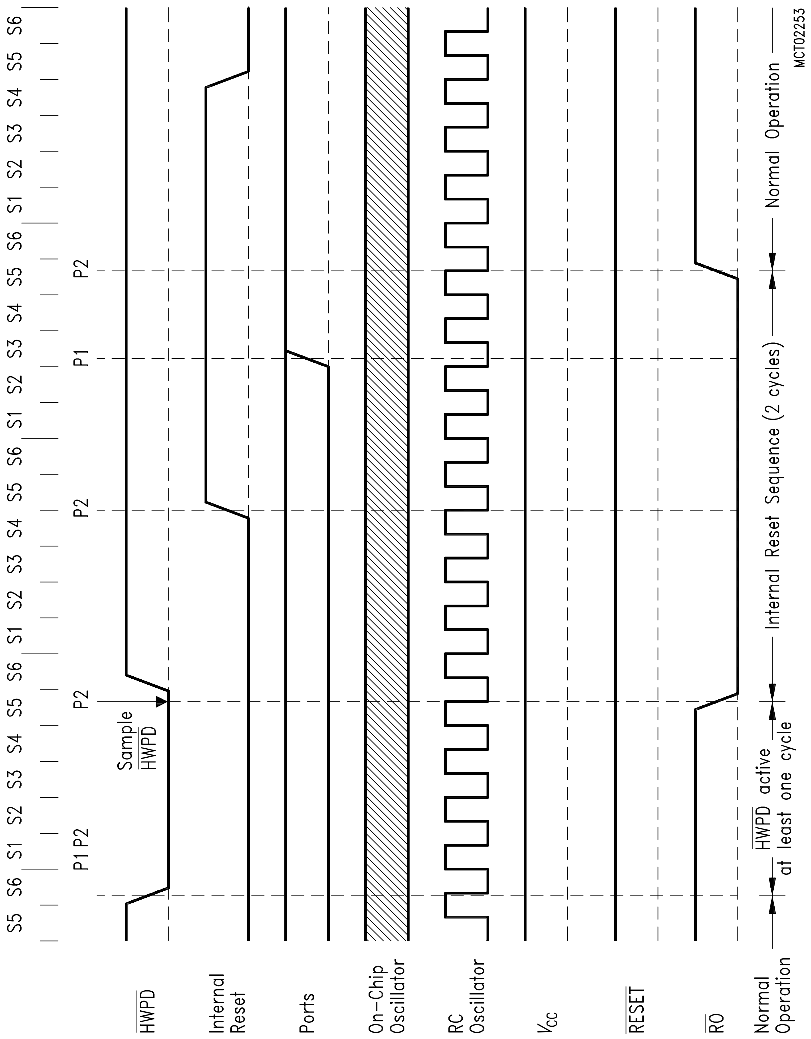

S5 S6 S1 S2 S3 S4 S5 S6 S1 S2 S3 S4 S5 S6 S1 S2 S3 S4 S5 S6 S1 S2 S3 S4 S5 S6
P1 P2
P2
P2
P1
P2
Sample HWPD
HWPD
Internal Reset
Ports
On-Chip Oscillator
RC Oscillator
V_{CC}
RESET
RO
Normal Operation
HWPD active at least one cycle
Internal Reset Sequence (2 cycles)
Normal Operation
MCT02253

Figure 9-3
Timing Diagram of Hardware Power Down Mode, HWPD-Pin is active for only one Cycle

10 The Bootstrap Loader

The C509-L includes a bootstrap mode, which is activated by setting the PRGEN pin at logic high level at the rising edge of the $\overline{\text{RESET}}$ or the $\overline{\text{HWPD}}$ signal (bit PRGEN1=1). In this mode software routines of the bootstrap loader, located at the addresses 0000_H to $01FF_H$ in the boot ROM will be executed. Its purpose is to allow the easy and quick programming of the internal XRAM ($F400_H$ to $FFFF_H$) via serial interface while the MCU is in-circuit. This allows to transfer custom routines to the XRAM, which will program an external 64 KByte FLASH memory. The serial routines of the bootstrap loader may be replaced by own custom software or even can be blocked to prevent unauthorized persons from reading out or writing to the external FLASH memory. Therefore the bootstrap loader checks an external FLASH memory for existing custom software and executes it.

The bootstrap loader consists of three functional parts which represent the three phases as described below.

10.1 General Functions of the Bootstrap Loader

Phase I : Check for existing custom software in the external FLASH memory and execute it.

Phase II : Establish a serial connection and automatically synchronize to the transfer speed (baud rate) of the serial communication partner (host).

Phase III : Perform the serial communication to the host. The host controls the bootstrap loader by sending header informations, which select one of four operating modes.
These modes are:

Mode 0: Transfer a custom program from the host to the XRAM ($F400_H$ - $FFFF_H$). This mode returns to the beginning of phase III.

Mode 1: Execute a custom program in the XRAM at any start address from $F400_H$ to $FFFF_H$.

Mode 2: Check the contents of any area of the external FLASH memory by calculating a checksum. This mode returns to the beginning of phase III.

Mode 3: Execute a custom program in the FLASH memory at any start address beyond 0200_H (at addresses 0000_H to $01FF_H$ the boot-ROM is active).

The three phases and their connections are illustrated in **figure 10-1**.

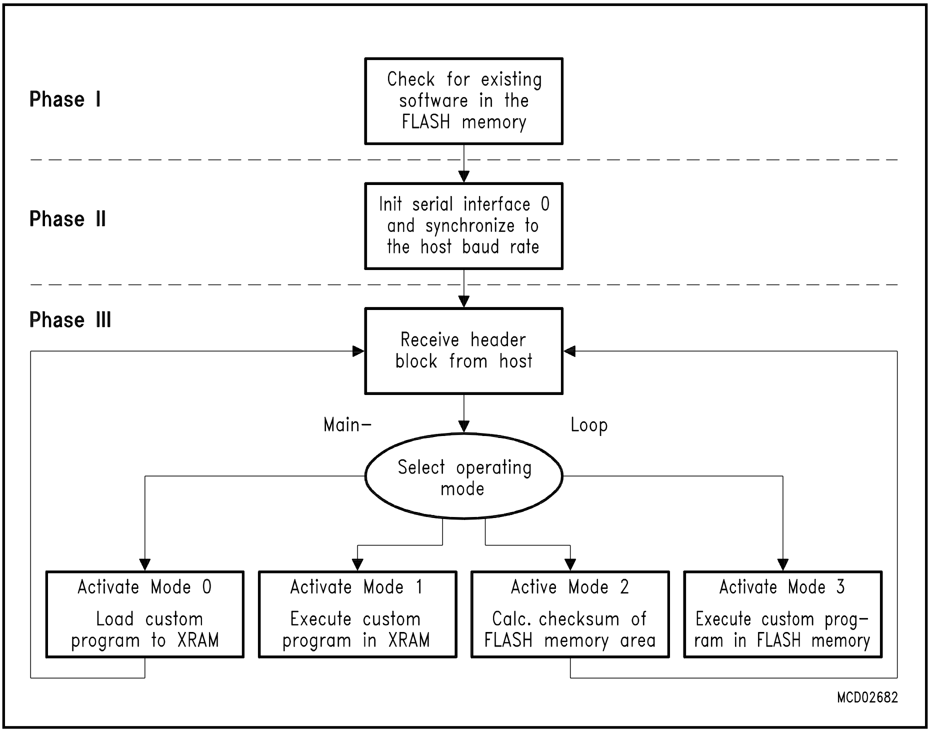

Phase I
Check for existing software in the FLASH memory
Phase II
Init serial interface 0 and synchronize to the host baud rate
Phase III
Receive header block from host
Main–
Loop
Select operating mode
Activate Mode 0
Load custom program to XRAM
Activate Mode 1
Execute custom program in XRAM
Active Mode 2
Calc. checksum of FLASH memory area
Activate Mode 3
Execute custom prog– ram in FLASH memory
MCD02682

Figure 10-1
The Three Phases of the Bootstrap Loader

The serial communication, which is activated in phase II is performed with the integrated serial interface 0 of the C509-L. Using a full- or half-duplex serial cable (RS232) the MCU must be connected to the serial port of the host computer as shown in **figure 10-**.

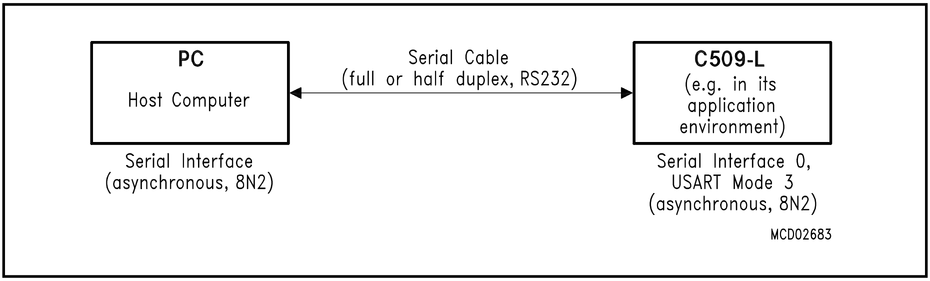


Figure 10-2
Bootstrap Loader Interface to the PC

The serial transfer operates in asynchronous mode with the serial parameters 8N2 (eight data bits, no parity and two stop bits). The baud rate however can be varied by the host in a wide range, because the bootstrap loader does an automatic synchronization in phase II.

The bootstrap loader itself does not use any of the implemented interrupts of the MCU for its work. But for further custom program execution in the bootstrap mode two interrupts are supported. These are the interrupts for the timer 0 (TIMER0) and for the serial interface 0 (SINT0). As the interrupt vectors are located in the address area of the bootstrap loader (0000$_H$ - 01FF$_H$), the appropriate interrupt vector addresses are routed by the bootstrap loader via a LJMP instruction to a reserved XRAM area at F400$_H$ - F41F$_H$ as shown below:

TIMER0:	usually at 000B$_H$ is routed to F400$_H$ in XRAM
SINT0:	usually at 0023$_H$ is routed to F410$_H$ in XRAM

At these addresses (F400$_H$ - F40F$_H$ and F410$_H$ - F41F$_H$) the user specified interrupt routines can be loaded to handle the interrupts in a user defined way. To avoid unexpected software behavior when these interrupts are used, the reserved memory area should be used only by interrupt handling routines. Notice: In this case the XRAM for custom programs is reduced to F420$_H$ - FFFF$_H$. If there is no need of these interrupts, the reserved memory area can be programmed with custom software.

It is recommended not to activate other interrupts than TIMER0 and SINT0, because this could lead to uncontrolled software execution in the interrupt vector area (0000$_H$ - 0100$_H$)!

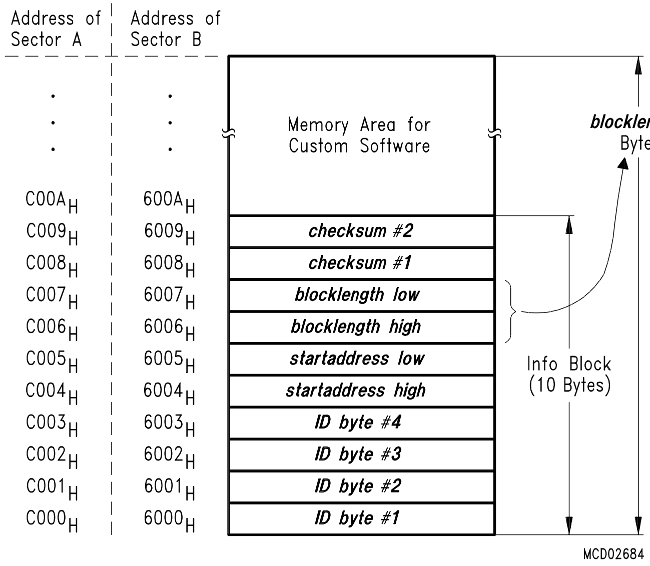
10.2 Phase I: Check for Existing Custom Software in the External FLASH Memory

The first action of the bootstrap trap loader is to check two blocks in the external FLASH memory for the existence of custom software. If the check is successful, the custom software is started directly. If no software is found, phase II is entered to establish a serial communication with the connected host. This feature can be used to protect the external FLASH memory contents against unauthorized in-system reading and writing in the bootstrap mode. The security check can be used if needed, but can also be skipped as well, as described in the next section.

10.2.1 Custom Software Check by the Info Block

For the above mentioned custom software two memory sectors (sector A: at C000$_H$, sector B: at 6000$_H$) of the external FLASH memory are reserved. For activation of the FLASH memory check, an info block is required, which has to be written at the beginning of the corresponding external FLASH memory sector. The structure of the info block is shown in **figure 10-3**.

The sector addresses C000$_H$ and 6000$_H$ are valid for the C509-L devices with stepping code CA and later.



Info Block Byte	Description
ID byte #1 - #4	Four ID bytes which mark the external FLASH memory sector as programmed.
startaddress (high, low)	The start address, where the custom program starts.
blocklength (high, low)	The length of the memory block, which is filled with the custom program code.
checksum #1, #2	Two checksum bytes for the external FLASH memory block
memory area for custom software	The area for the custom program starting at C00A$_H$ in sector A and 600A$_H$ in sector B

Figure 10-3
The Structure of the FLASH Memory Info Block

The info block starts with a fixed sequence of four identification bytes (***ID bytes***). They mark the corresponding memory sector as custom programmed. If the ID bytes are not present at the beginning of the info block, the bootstrap loader assumes, that the corresponding sector is not custom programmed.

The four identification bytes must be absolutely definite to prevent the bootstrap loader from recognizing normal program code as identification bytes. Therefore the four bytes represent a not senseful instruction sequence in 8051-code, which should never occur in normal programs. The definition of the identification bytes is shown in **table 10-1**.

Table 10-1
ID Bytes of the Info Block

ID byte	Value	8051-code
#1	23_H	RL A
#2	03_H	RR A
#3	33_H	RLC A
#4	13_H	RRC A

To check, if the custom software in the relevant sector is functional, the bootstrap loader calculates a checksum consisting of two independent checksum bytes over the appropriate memory area. This area begins at the address ***startaddress*** and has a length of ***blocklength***. The two generated checksum bytes are compared with ***checksum #1*** and ***#2*** of the info block. If the checksums are equal, the bootstrap loader starts the custom routine in this sector at the address ***startaddress***.

The bootstrap loader starts checking sector A. If either the identification bytes or the checksums of sector A are not correct, the checking procedure is done with sector B of the external FLASH memory. The check of two sectors (A and B) is necessary for a maximum of security, e.g. if the custom software in one sector is not functional. This can happen e.g., if the programming of the corresponding external FLASH memory sector suddenly is interrupted by the cause of a power failure.

When the check fails in both sectors, the bootstrap loader leaves phase I and enters phase II to establish the serial communication to a connected host via serial interface 0.

If the customer does not want to use this external FLASH sector check, he can override it by programming other values than the four ID bytes in the corresponding FLASH memory addresses in sector A and B. If this is done, the remaining bytes of the two external FLASH memory sectors may be used for normal program execution as well.

The flowchart in **figure 10-4** shows the detailed actions of the bootstrap loader in phase I.

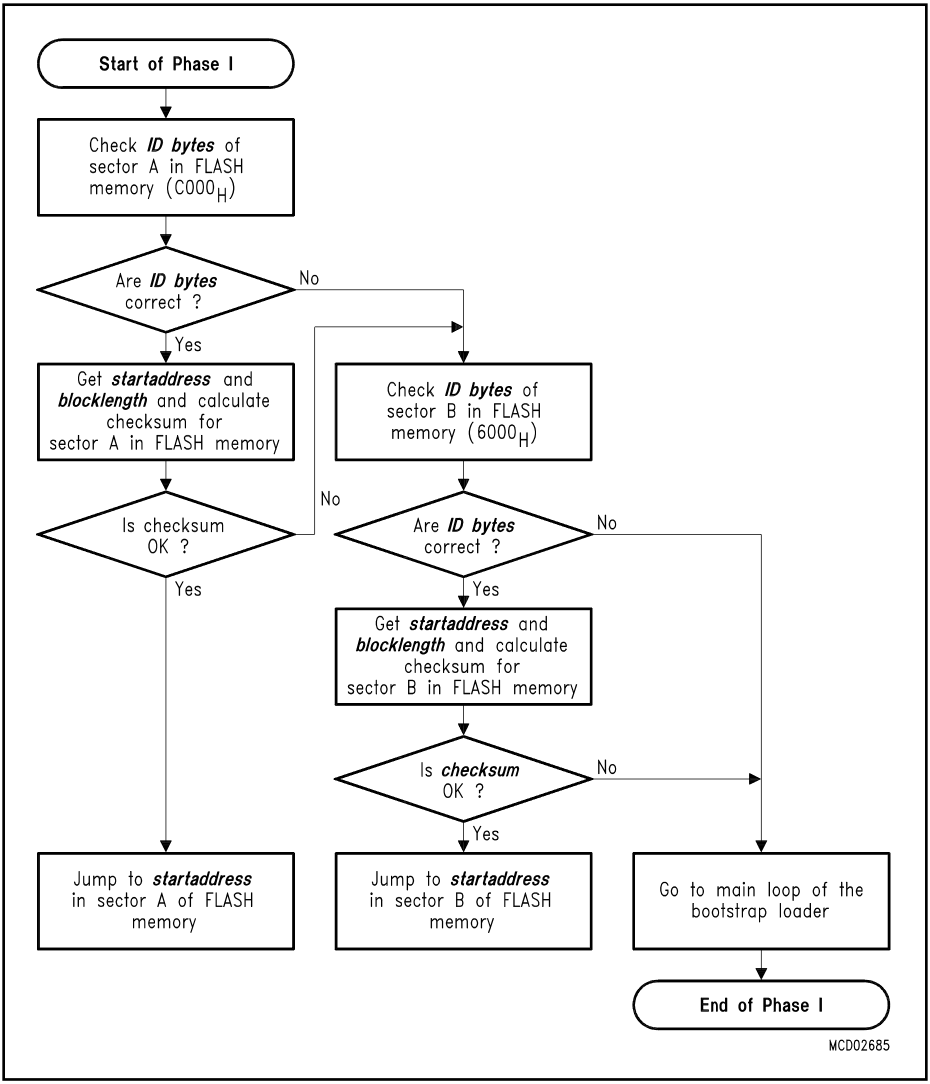

Start of Phase I
Check ID bytes of sector A in FLASH memory (C000H)
Are ID bytes correct ?
No
Yes
Get startaddress and blocklength and calculate checksum for sector A in FLASH memory
Is checksum OK ?
No
Yes
Check ID bytes of sector B in FLASH memory (6000H)
Are ID bytes correct ?
No
Yes
Get startaddress and blocklength and calculate checksum for sector B in FLASH memory
Is checksum OK ?
No
Yes
Jump to startaddress in sector A of FLASH memory
Jump to startaddress in sector B of FLASH memory
Go to main loop of the bootstrap loader
End of Phase I
MCD02685

Figure 10-4
Flowchart of Phase I Actions

10.2.2 Checksum Calculation

The external FLASH memory check in phase I uses a checksum algorithm, which is based on a continuous addition and 8-bit-left-rotation of all relevant data bytes. The data in the corresponding FLASH memory area is split into two parts, on which an extra checksum is built. This is done to reach a maximum of data security. ***Checksum #1*** is build by bytes at odd addresses and ***checksum #2*** is calculated with bytes at even addresses. The checksum calculation itself is described in the flowchart in **figure 10-5**.

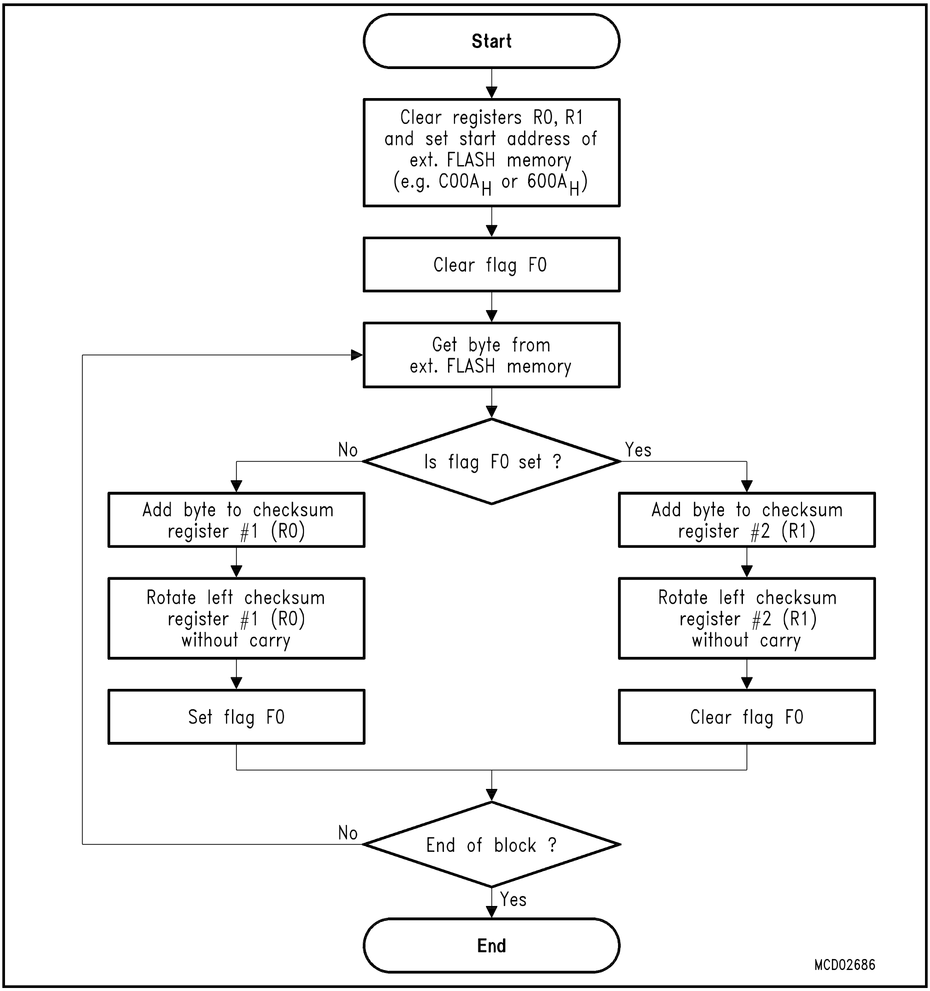

Start
Clear registers R0, R1 and set start address of ext. FLASH memory (e.g. C00A$_H$ or 600A$_H$)
Clear flag F0
Get byte from ext. FLASH memory
Is flag F0 set ?
No
Yes
Add byte to checksum register #1 (R0)
Add byte to checksum register #2 (R1)
Rotate left checksum register #1 (R0) without carry
Rotate left checksum register #2 (R1) without carry
Set flag F0
Clear flag F0
End of block ?
No
Yes
End
MCD02686

Figure 10-5
Flowchart of the Checksum Calculation for the external FLASH Memory

10.3 Phase II: Automatic Serial Synchronization with the Host

When the bootstrap loader leaves phase I and enters phase II**,** the synchronization procedure between MCU and host will be started. The synchronization must be handled by the host system as shown in **figure 10-6**.

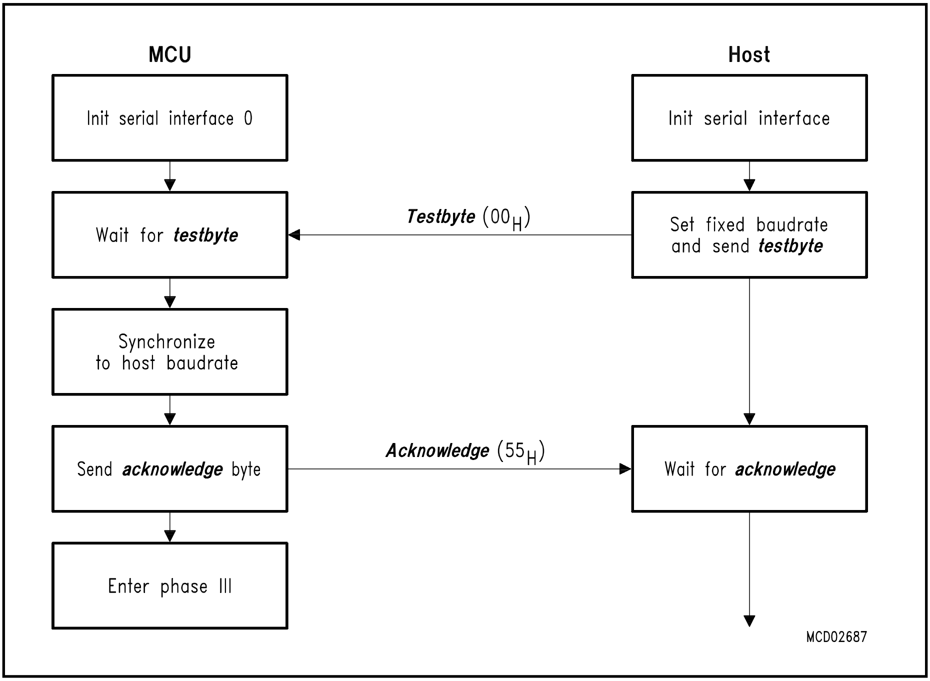


Figure 10-6
The Synchronization between MCU and Host

After receiving the ***testbyte*** from the host, the bootstrap loader calculates the actual baud rate and activates the baud rate generator of the serial interface 0. When the synchronization is accomplished, the MCU sends an ***acknowledge*** byte (55_H) back to the host. The baud rate calculation works correctly only in a specific range of baud rates. If the synchronization fails, the baud rates between MCU and host are different, and the acknowledge code from the MCU can't be received properly by the host. In this case, the host software may give a message to the customer, e.g. that he has to repeat the synchronization procedure.

Attention: the bootstrap loader doesn't recognize, if the synchronization was correct. It always enters phase III after sending the acknowledge byte. Therefore, if synchronization fails, a reset of the MCU has to be invoked, to restart the bootstrap loader for a new synchronization attempt.

10.3.1 Automatic Synchronization Procedure

In phase II the bootstrap loader starts the serial communication with the host via the serial interface 0 of the C509-L. The interface of the MCU is set to mode 3, which means asynchronous transmission with the data format 8N2 (eight data bits, no parity, two stop bits). The host has to use the same serial parameters.

For the baud rate synchronization of the MCU to the fixed baud rate of the host, the bootstrap loader waits for the ***testbyte*** (00_H), which has to be sent by the host. By polling the receive port of the serial interface 0 (P3.0 / RxD) the bootstrap loader measures the receiving time of the test byte by using timer 0 as shown in the **figure 10-7**.

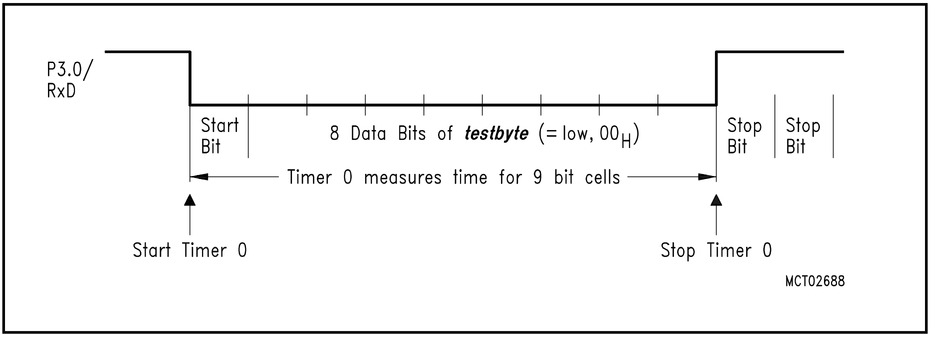

P3.0/
RxD
Start
Bit
8 Data Bits of testbyte (= low, 00H)
Stop
Bit
Stop
Bit
Timer 0 measures time for 9 bit cells
Start Timer 0
Stop Timer 0
MCT02688

Figure 10-7
Measuring the receive time of a zero byte by using Timer 0

The resulting timer value is used to calculate the reload value for the 10-bit baud rate generator of the serial interface 0 (S0REL). This calculation needs two formulas: the correlation between the baud rate (Bd) and the reload value (S0REL) depending on the oscillator frequency of the MCU (fosc)

$$\mathrm{Bd} = \frac{\mathrm{fosc}}{32 \times (1024 - \mathrm{S0REL})} \qquad (1)$$

and the relation between the baud rate (Bd) and the value of timer 0 (T0) depending on the oscillator frequency (fosc) and the number of received bits (Nb).

$$\mathrm{Bd} = \frac{\mathrm{fosc} \times \mathrm{Nb}}{\mathrm{T0} \times 12} \qquad (2)$$

Equations (1) and (2) and resolving the result to S0REL leads to the formula

$$\mathrm{S0REL} = 1024 - \frac{\mathrm{T0} \times 12}{32 \times \mathrm{Nb}}$$

which is independent from the oscillator frequency of the MCU (fosc). The value of Nb is nine, because one start bit plus eight data bits are measured. The resulting formula then is

$$\mathrm{S0REL} = 1024 - \frac{\mathrm{T0} \times 12}{32 \times 9}$$

This equation contains the constant factor

$$\frac{12}{32 \times 9} = 0.0417$$

So the formula can be written as

$$\mathrm{S0REL} = 1024 - 0.0147 \times \mathrm{T0}$$

To avoid complicated float point arithmetic the factor 0.0417 is scaled by multiplying it with 4096 (result is 171) and then performing an integer multiplication with T0. In the next step the product is rescaled by a integer division with 4096, which can be simply achieved by a twelve bit right-shift operation. The final formula for calculating the reload value S0REL including scaling and rescaling is therefore:

$$\mathrm{S0REL} = 1024 - \frac{171 \times \mathrm{T0}}{4096} \qquad (3)$$

Additionally, the result of the division is rounded by a simple bit comparison of the last right shifted bit. After setting S0REL to the calculated value and activating the baud rate generator of the serial interface 0, the bootstrap loader sends an ***acknowledge*** byte (55_H) to the host. If this byte is received correctly, it will be assured, that both serial interfaces are working with the same baud rate.

The flowchart in **figure 10-8** shows the calculation of the reload value S0REL for the baud rate generator of the serial interface 0.

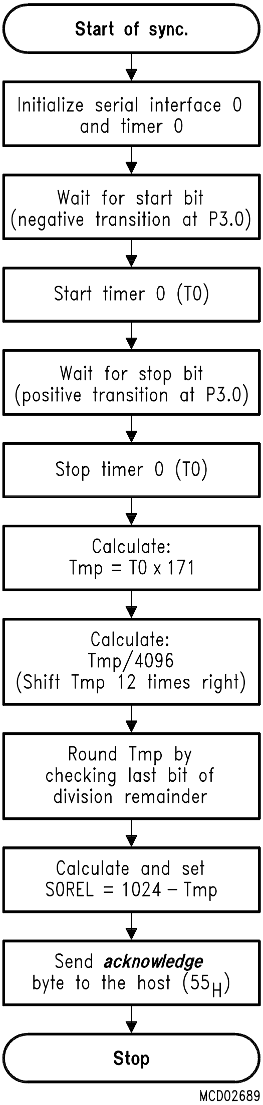

Start of sync.
Initialize serial interface 0 and timer 0
Wait for start bit (negative transition at P3.0)
Start timer 0 (T0)
Wait for stop bit (positive transition at P3.0)
Stop timer 0 (T0)
Calculate: Tmp = T0 x 171
Calculate: Tmp/4096 (Shift Tmp 12 times right)
Round Tmp by checking last bit of division remainder
Calculate and set S0REL = 1024 – Tmp
Send acknowledge byte to the host (55H)
Stop
MCD02689

Figure 10-8
Calculation Scheme of the S0REL Value

10.3.2 Baud Rates for Correct Synchronization

The automatic baud rate synchronization will work correctly only in a specific range of baud rates, which depends on the oscillator frequency fosc of the C509-L and the resolution of the timer 0 (T0).

The minimum baud rate (Bd_{min}) results in the possible underflow of SFR S0REL when the value of T0 gets greater than 24556. In this case the value of SFR S0REL**,** corresponding formula (3)**,** gets below zero, which leads to a underflow of the S0REL register and therefore to a incorrect baud rate of the baud rate generator. The formula for calculating this underflow margin is derived from formula (1) and is reduced to

$$Bd_{min} = \frac{fosc}{32768}$$

The theoretical maximum baud rate (Bd_{high}) can be attained if S0REL is set to its maximum value of 1023**.** In this case formula (1) reduces to

$$Bd_{high} = \frac{fosc}{32}$$

The real maximum baud rate (Bd_{max}) is smaller, because of the decreasing resolution of S0REL and T0 at higher baud rates, This causes an increasing deviation between the host baud rate and the MCU baud rate. To perform a correct transfer between the MCU and the host without transmission errors, the deviation Fb of the host baud rate to the MCU baud rate may not exceed 2.5%.

$$Fb = 0.025 \le \frac{Bd_{host} - Bd_{MCU}}{Bd_{host}}$$

Between Bd_{min} and Bd_{max} every host baud rate can be synchronized successfully by the bootstrap loader. Above Bd_{max} only discreet baud rates with a deviation of less than 2.5 % may be used. **Table 10-2** shows the guaranteed range of baud rates Bd_{min} to Bd_{max} and typical functional host baud rates above Bd_{max} for some MCU clock rates fosc.

Table 10-2
Typical Baudrate Selections

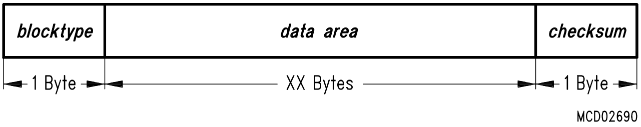
MCU clock rate fosc	Minimum baud rate Bd_{min}	Maximum baud rate Bd_{max} (F_b 2,5%)	Functional higher baud rates
8 MHz	250 baud	6580 baud	9600, 19200 baud
12 MHz	370 baud	9380 baud	9600, 19200 baud
16 MHz	490 baud	12830 baud	19200, 38400 baud

10.4 Phase III: Serial Communication with the Host

After the successful synchronization with the host the bootstrap loader enters phase III, in which it communicates with the host to select the desired operating modes. This communication between host and bootstrap loader in phase III is based on different types of transfer blocks. Prior to the description of the phase III operations, the transfer blocks are described in the next section.

10.4.1 Block Transfer Protocol

The communication between the host and the bootstrap loader is done by a simple transfer protocol, which is based on a specified block structure. The communication is nearly unidirectional, that means, that the host is sending several transfer blocks and the bootstrap loader is just confirming them by sending back single acknowledge or error bytes. The MCU itself does not send any transfer blocks. **Figure 10-9** shows the general format of the transfer block.



Format Item	Description
blocktype	This byte determines how data in the ***data area*** field has to be interpreted. The implemented blocktypes are: 00_H Type: “HEADER” 01_H Type: “DATA” 02_H Type: “END OF TRANSMISSION” (EOT)"
data area	This area contains a number of bytes which represent the data of the transfer block. The number of bytes in the data area may range between 00_H and $7C_H$.
checksum	For safety purposes a checksum is build over ***blocktype*** and ***data area*** field and sent after the data area.

Figure 10-9
Basic Structure of a Bootstrap Loader Transfer Block

A transfer block is built by the host depending on the data (header or program data) it contains. For safety purposes the host calculates a simple checksum of the whole block (***blocktype*** and ***data area***) to attach it at the end of the block. The checksum must be generated by EXOR-ing all bytes of the transfer block with themselves. Every time the bootstrap loader receives a transfer block, it recalculates the checksum of the received bytes (***blocktype*** and ***data area***) and compares it with the attached checksum. If the comparison fails, the bootstrap loader is rejecting the transfer block by sending back a checksum error byte (FE_H) to the host. Another possible error is a wrong block type. In this case the bootstrap loader sends back a block error byte (FF_H) to the host. In both error cases the bootstrap loader awaits the actual transfer block from the host again. If a block is received correctly, an acknowledge byte (55_H) is sent to the host.

Table 10-3
Confirmation Bytes of the Bootstrap-Loader

Receive status	Transmitted code to host
Acknowledge	55_H
Block Error	FF_H
Checksum Error	FE_H

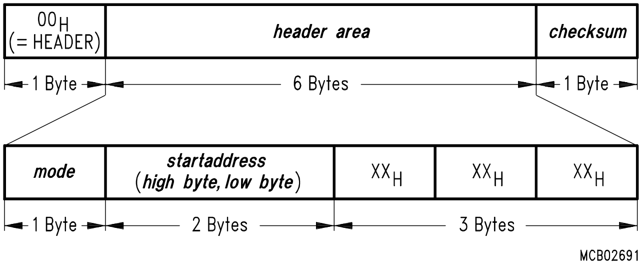
Three types of transfer blocks depending on the value of ***blocktype*** are implemented in the transfer protocol. **Table 10-4** gives an overview of these block types. The detailed structures of the blocks are described in the following sections.

Table 10-4
Types of Transfer Blocks

Block Name	***blocktype***	Description
Header Block	00_H (HEADER)	This block always has a length of 8 bytes (including the attached checksum) and contains special information in the ***data area***, which selects the operating mode of the bootstrap loader in phase III.
Data Block	01_H (DATA)	This block is used in operating mode 0 to transfer a portion of normal data in the ***data area*** (e.g. program code) from the host to the XRAM of the MCU. The length of this block depends on the information given in the header block before.
EOT Block	02_H (EOT)	This block is used to indicate the end of a data transmission in operating mode 0. It contains the last bytes of the transferred data. The length of this block depends on the information given in a header block before.

10.4.1.1 Header Block Definition

The header block from the host, which contains the mode number and additional data to start the selected operating mode, is a normal transfer block with the block type HEADER (00_H) and a fixed length of eight bytes (including the attached checksum of the transfer block). **Figure 10-10** shows the general structure of this header block is shown below.



Format Item	Description
mode	This field contains the number of the operating mode: 00_H Load custom program into the XRAM memory. 01_H Jump to the XRAM memory at the specified address ***startaddress***. 02_H Calculate the checksum of a specified part of the external FLASH memory which is defined by startaddress and datalength. 03_H Jump to the FLASH memory at the specified address startaddress.
startaddress	This 16-bit address defines the start address either in XRAM or in ext. FLASH memory. The first byte to be transmitted is the high order byte of the 16-bit address.
XX_H	These bytes have a different meaning depending on the mode byte of a header block. The detailed definition of these bytes is given in **figures 10-11**, **10-18**, **10-21**, and **10-24**.

Figure 10-10
Structure of the HEADER Block

10.4.1.2 Data and EOT Block Definition

Data and EOT transfer blocks are used by the bootstrap loader to transfer data from the host to the XRAM of the MCU. A data block contains a bulk of data, which has to be copied to the XRAM beginning at an address which is defined by ***startaddress*** of a preceding header block. The number of ***data bytes*** is also specified by the ***blocklength*** field of a proceeding header block.

The EOT block contains the last bytes of a data transmission to the XRAM. The number of the remaining relevant bytes in the EOT block is given in ***no. of bytes***.

Figure 10-11 shows the structure of Data and EOT block.

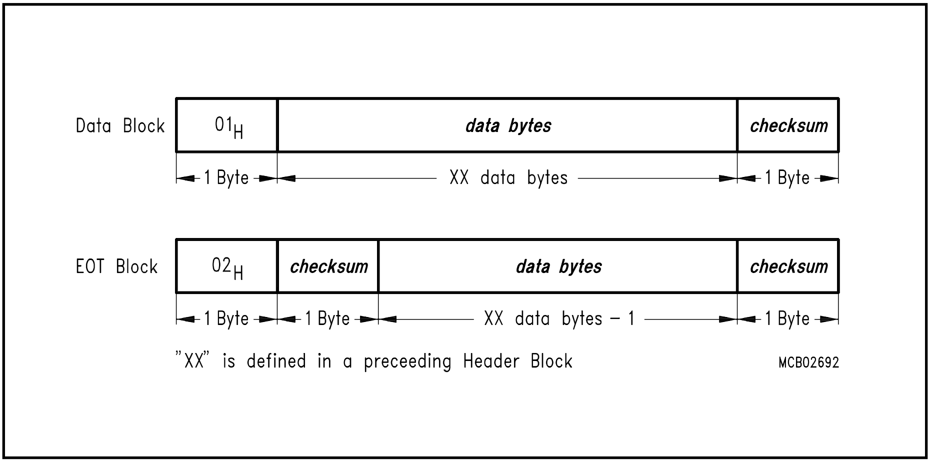


Figure 10-11
Data Block and EOT Block Structure

10.4.2 Operating Mode Selection

When the bootstrap loader enters phase III, it first waits for an eight byte long header block from the host, which will be confirmed with an acknowledge byte (55_H). The header block contains the information for the selection of the operating mode. Depending on this data the bootstrap loader selects and activates the desired operating mode. This select procedure shows the block diagram in **figure 10-12**.

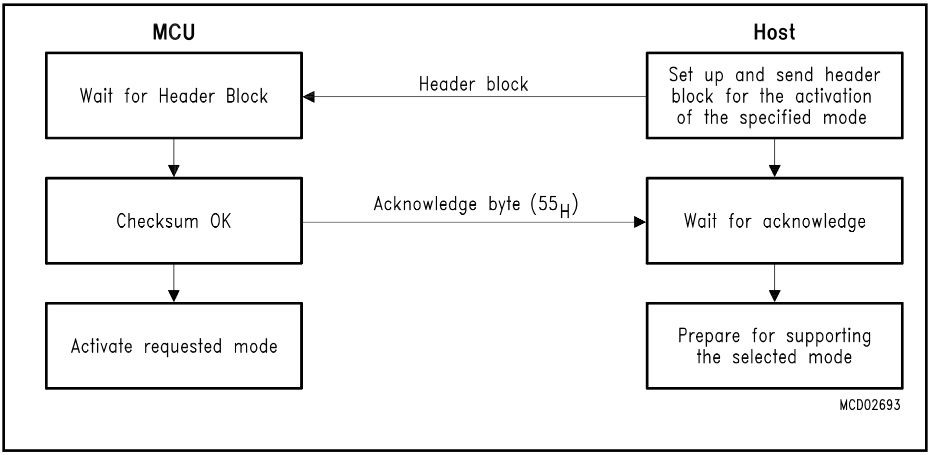

MCU
Host
Wait for Header Block
Header block
Set up and send header block for the activation of the specified mode
Checksum OK
Acknowledge byte (55_H)
Wait for acknowledge
Activate requested mode
Prepare for supporting the selected mode
MCD02693

Figure 10-12
Operating Mode Selection Procedure

If the MCU receives an incorrect header block, e.g. because of a bad serial transmission, the bootstrap loader sends, instead of an acknowledge byte, a checksum- or block error byte (see **table 10-3**) to the host and awaits the header block again. In this case the host may react by resending the header block or by releasing a message to the customer.

The flowchart in **figure 10-13** shows the structure of the phase III bootstrap loader part in detail.

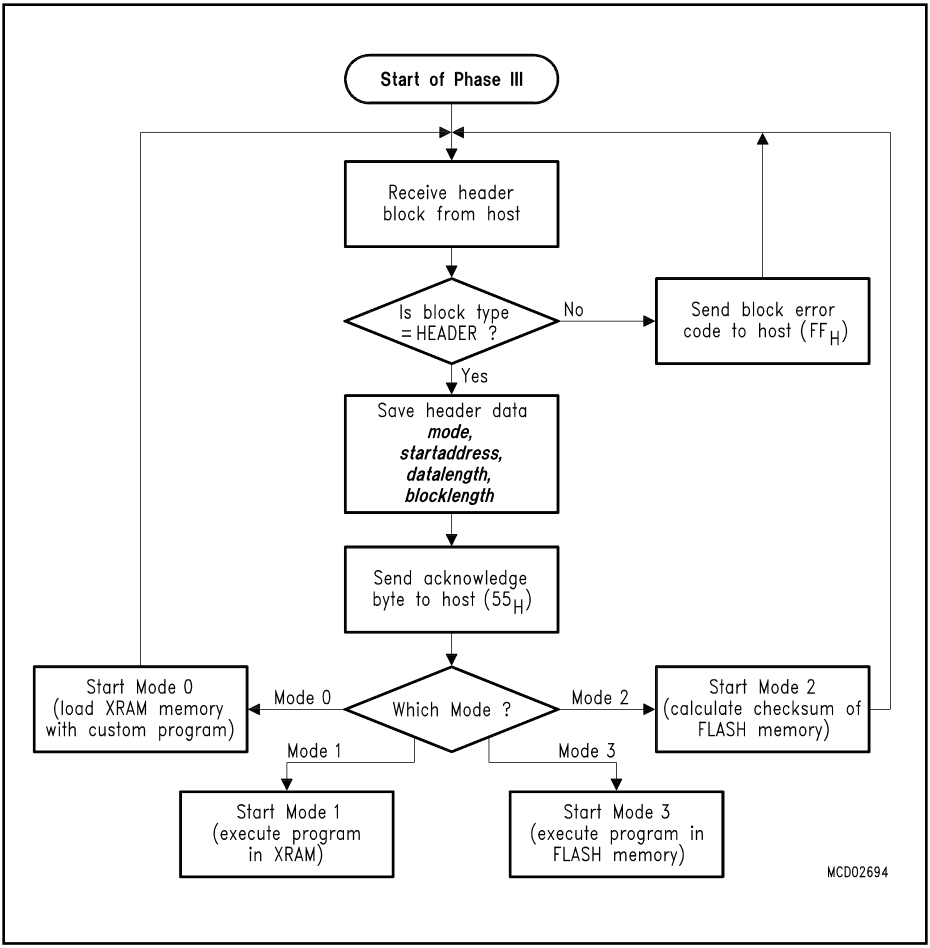

Start of Phase III
Receive header block from host
Is block type = HEADER ?
No
Send block error code to host (FF$_H$)
Yes
Save header data *mode, startaddress, datalength, blocklength*
Send acknowledge byte to host (55$_H$)
Which Mode ?
Mode 0
Start Mode 0 (load XRAM memory with custom program)
Mode 2
Start Mode 2 (calculate checksum of FLASH memory)
Mode 1
Start Mode 1 (execute program in XRAM)
Mode 3
Start Mode 3 (execute program in FLASH memory)
MCD02694

Figure 10-13
Flowchart of Phase III

10.4.2.1 Selection of Operating Mode 0

Operating mode 0 is used to transfer a program from the host to the XRAM of the MCU via serial interface. The header block, which has to be prepared and sent by the host for the activation of operating mode 0 must have the structure as shown in **figure 10-11**.

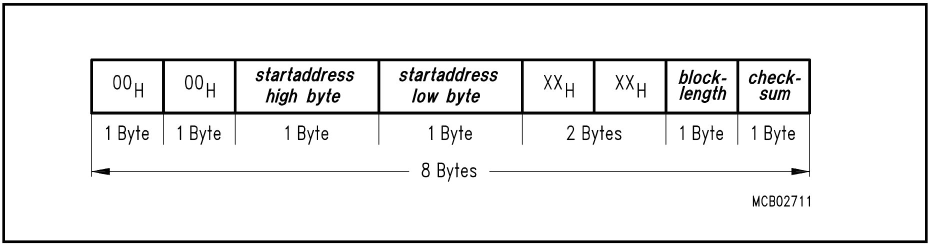


Figure 10-14
Header Block for Operating Mode 0

The operating mode 0 header block transfers the 16-bit XRAM ***startaddress*** for the following data blocks, the number of data bytes in the following data or EOT blocks ***(blocklength***), and a ***checksum*** byte.

After confirming the received header block, the bootstrap loader enters mode 0, in which the desired data is transmitted from the host to the XRAM of the MCU using the two transfer blocks of the types DATA and EOT, which are shown in **figure 10-11**.

The complete communication for mode 0 (including entering mode 0) between the host and the MCU after the synchronization shows the block diagram in **figure 10-15**.

If an error occurs while transmitting data blocks, the host software has to react on error codes, sent by the bootstrap loader when rejecting a block. This case is illustrated in the block diagram of **figure 10-16**.

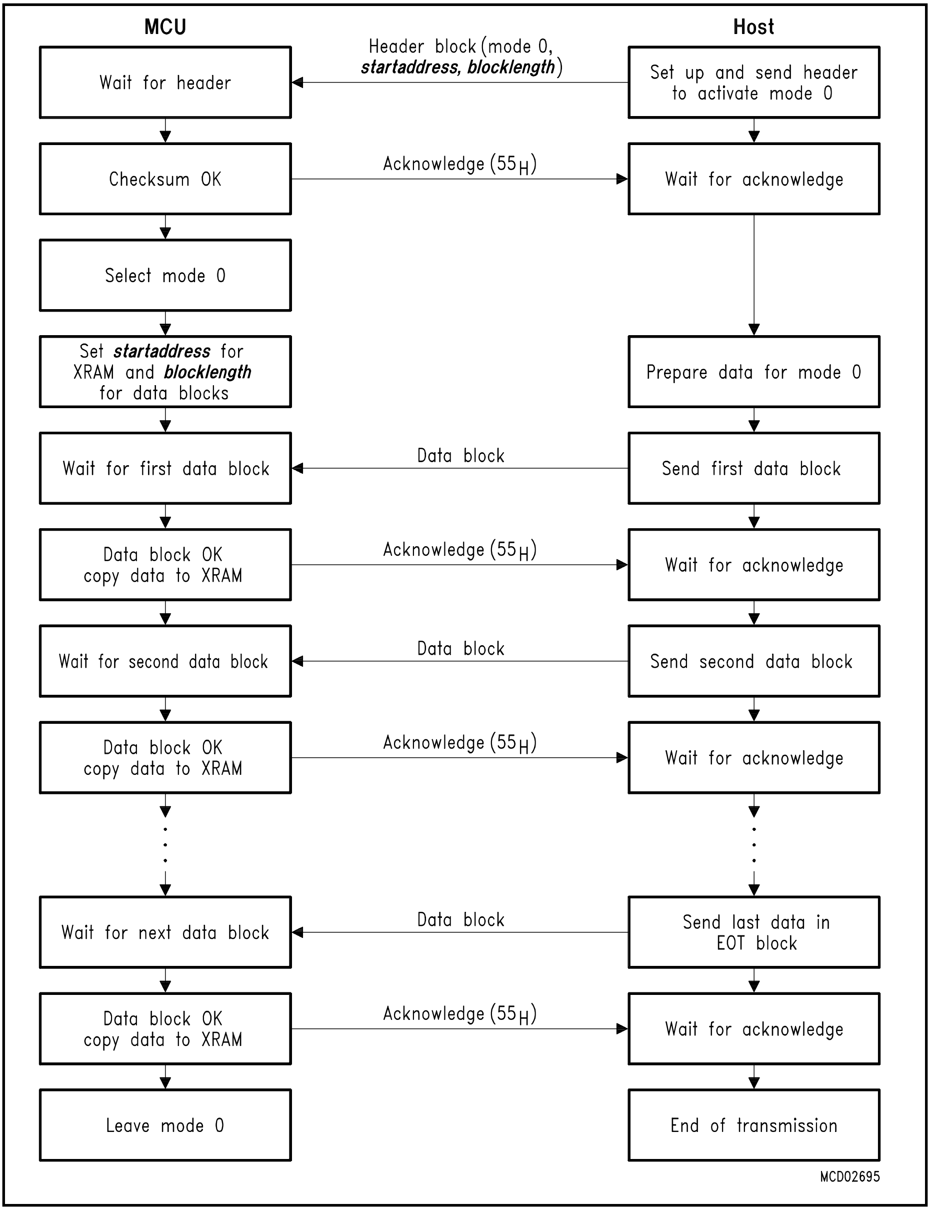


Figure 10-15
Operating Mode 0 Communication Structure

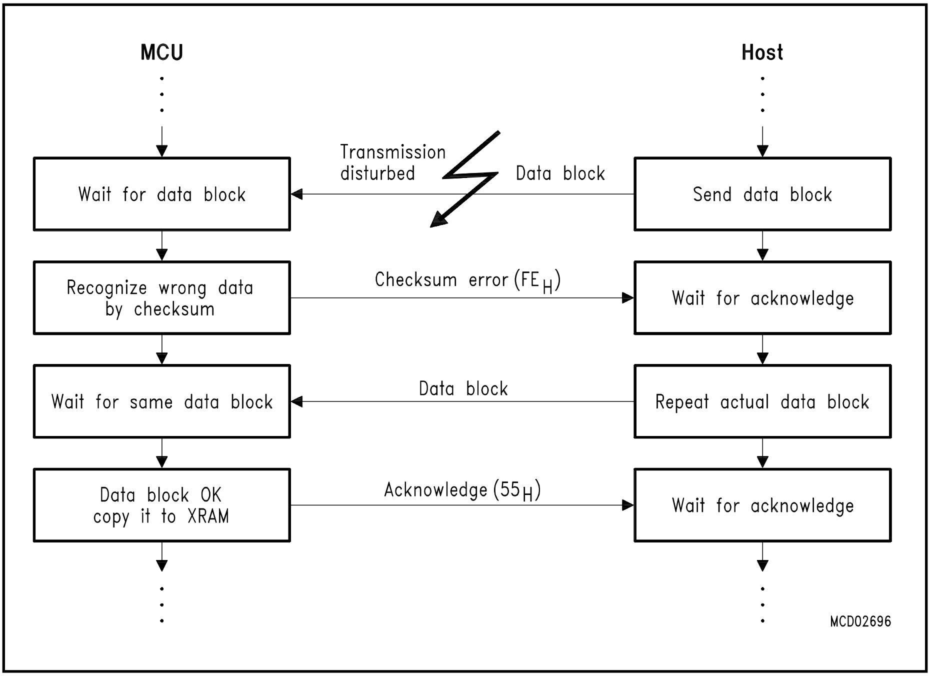


Figure 10-16
Handling Transmission Errors in Operating Mode 0

If the host sends a header block or a block that is not implemented in the protocol (a block type number higher than 02_H), the bootstrap loader reacts in a similar way as described in the figure above, with the exception, that now a block error code (FE_H) is sent to the host. It is up to the host software to handle this error properly.

The bootstrap loader flowchart of the complete transfer protocol of mode 0 is shown in **figure 10-17**.

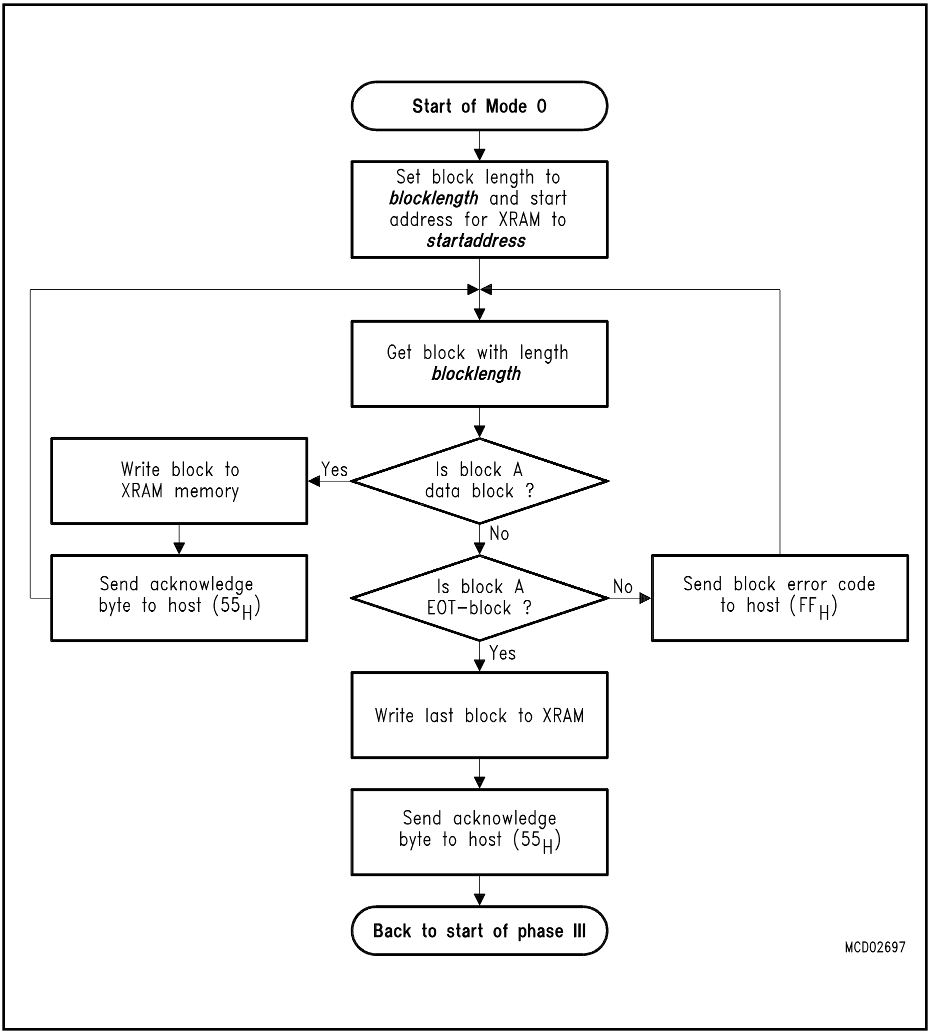

Start of Mode 0
Set block length to *blocklength* and start address for XRAM to *startaddress*
Get block with length *blocklength*
Is block A data block ?
Yes
Write block to XRAM memory
Send acknowledge byte to host (55_H)
No
Is block A EOT–block ?
No
Send block error code to host (FF_H)
Yes
Write last block to XRAM
Send acknowledge byte to host (55_H)
Back to start of phase III
MCD02697

Figure 10-17
Bootstrap Loader Flowchart of Operating Mode 0

10.4.2.2 Selection of Operating Mode 1

Mode 1 is used to execute a custom program in the XRAM of the MCU at a given start address. The header block, which has to be prepared and sent by the host for the activation of operating mode 1 has the structure as shown in **figure 10-18**.

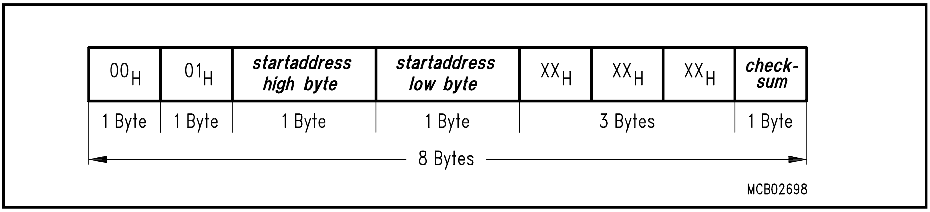


Figure 10-18
Header Block for Operating Mode 1

The operating mode 1 header block transfers the 16-bit XRAM ***startaddress*** for program execution in the XRAM, three dummy bytes, and a ***checksum*** byte.

After sending the appropriate header block for mode 1, no further serial communication is necessary. The block diagram for the communication between the host and the MCU is shown in **figure 10-19**:

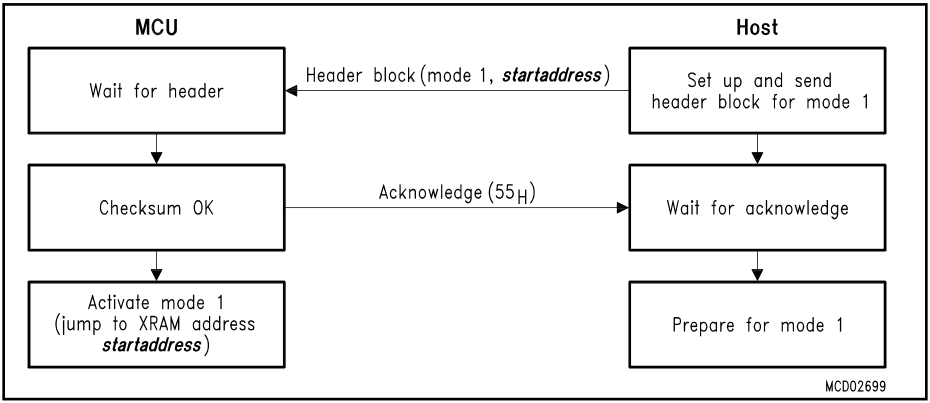

MCU
Host
Wait for header
Header block (mode 1, startaddress)
Set up and send header block for mode 1
Checksum OK
Acknowledge (55_H)
Wait for acknowledge
Activate mode 1 (jump to XRAM address startaddress)
Prepare for mode 1
MCD02699

Figure 10-19
Operating Mode 1 Communication Structure

Mode 1 swaps the XRAM to code memory by setting the corresponding swap-bit in SFR SYSCON1 and starts the program execution in the XRAM at any start address given in the header block at ***startaddress***.

Note: If the supported interrupts TIMER 0 and SINT 0 are used as described in section 10.1, the XRAM area from $F400_H$ to $F41F_H$ is reserved for interrupt handling routines. Therefore ***startaddress*** has to be greater than or equal to $F420_H$.

The corresponding bootstrap loader flowchart of mode 1 is shown in **figure 10-20**.

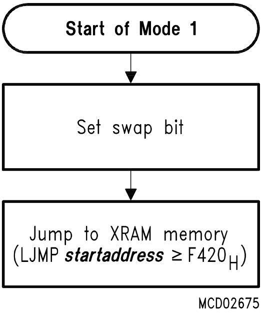
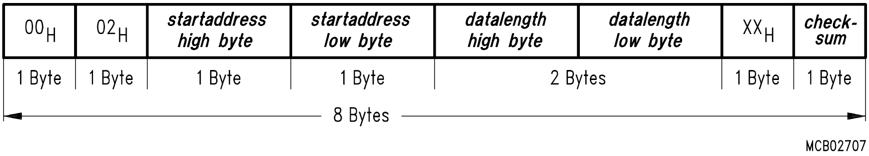


Figure 10-20
Bootstrap Loader Flowchart of Operating Mode 1

10.4.2.3 Selection of Operating Mode 2

Mode 2 is used to calculate a checksum of an external FLASH memory sector beginning at a ***startaddress*** with the specified length ***datalength***. The header block, which has to be prepared and sent by the host for the activation of operating mode 2 is shown in **figure 10-21**.



00$_H$	02$_H$	*startaddress high byte*	*startaddress low byte*	*datalength high byte*	*datalength low byte*	XX$_H$	*check-sum*
1 Byte	1 Byte	1 Byte	1 Byte	2 Bytes		1 Byte	1 Byte

Figure 10-21
Header Block for Operating Mode 2

The operating mode 2 header block transfers the 16-bit ***startaddress*** of the external FLASH memory block, which has to be checked, the number of bytes (***datalength***), which have to be checked beginning at ***startaddress***, a dummy byte, and a ***checksum*** byte.

Mode 2 calculates a checksum of any area of the external FLASH memory starting at ***startaddress*** with a length of ***datalength*** (see also section 10.2.2). The endaddress of the corresponding memory block is ***startaddress + datalength***. The two calculated checksum bytes are compared with two fixed checksum values which must be placed at the end of the checked external FLASH memory block, that is at addresses ***startaddress* + *datalength* + 1** and ***startaddress* + *datalength* + 2**. If these checksum values are equal to the calculated checksums, an acknowledge byte (55$_H$) is sent to the host. Otherwise the checksum error code (FE$_H$) is transmitted to the host.

The block diagram in **figure 10-22** shows the mode 2 checksum calculation in operating mode 2.

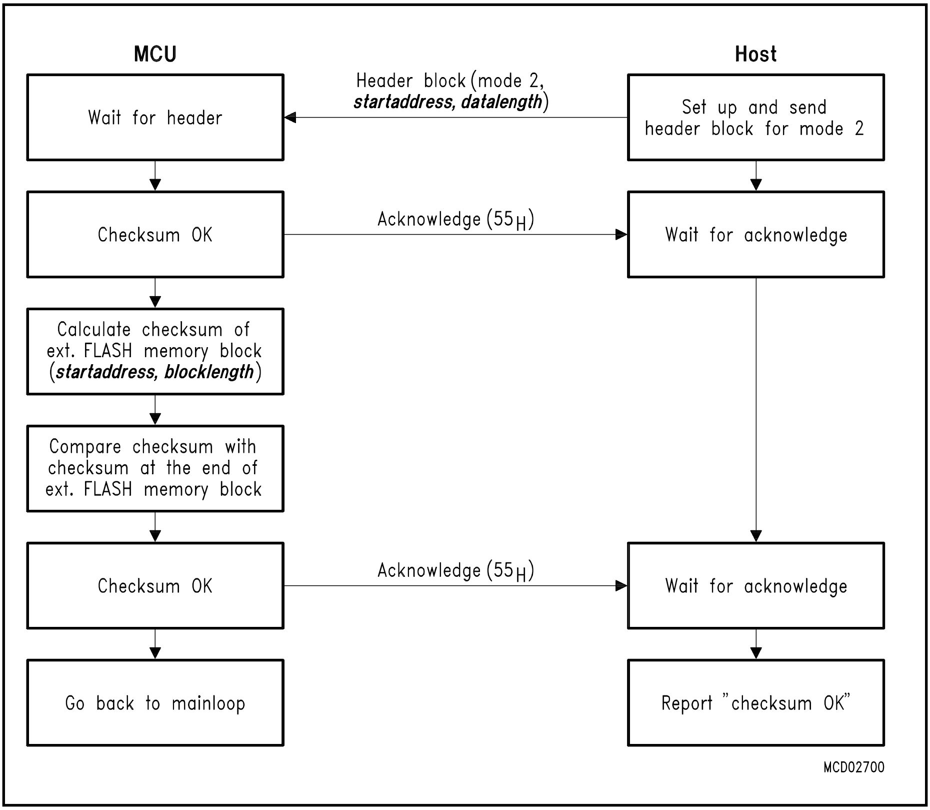

MCU
Host
Header block (mode 2, *startaddress, datalength*)
Wait for header
Set up and send header block for mode 2
Checksum OK
Acknowledge (55H)
Wait for acknowledge
Calculate checksum of ext. FLASH memory block (*startaddress, blocklength*)
Compare checksum with checksum at the end of ext. FLASH memory block
Checksum OK
Acknowledge (55H)
Wait for acknowledge
Go back to mainloop
Report "checksum OK"
MCD02700

Figure 10-22
Operating Mode 2 Communication Structure

Figure 10-23 shows the bootstrap loader flowchart of operating mode 2.

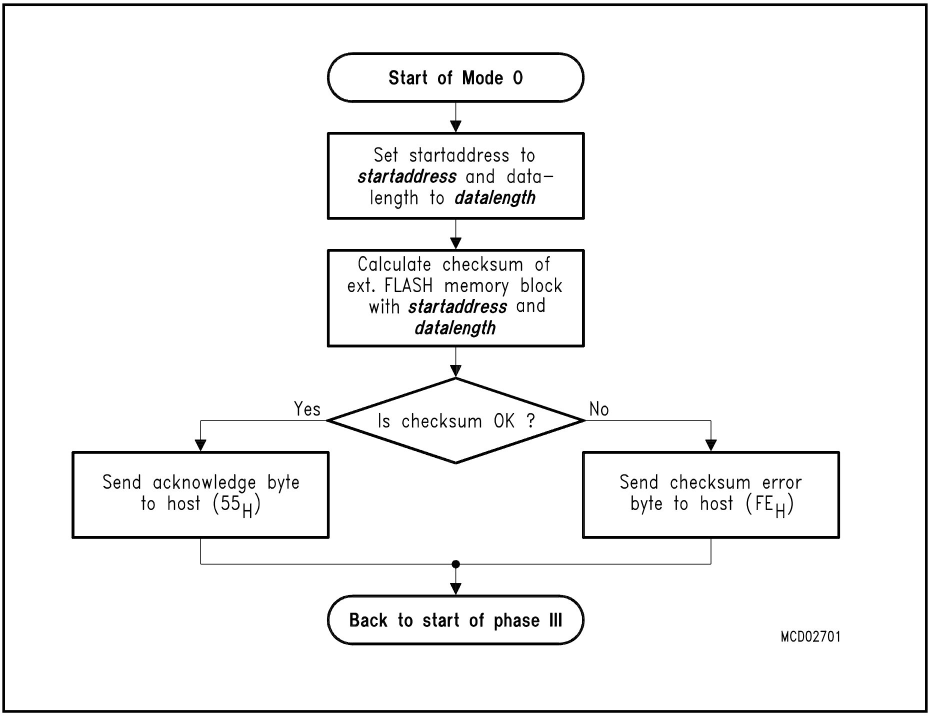

Start of Mode 0
Set startaddress to *startaddress* and data-length to *datalength*
Calculate checksum of ext. FLASH memory block with *startaddress* and *datalength*
Is checksum OK ?
Yes
No
Send acknowledge byte to host (55H)
Send checksum error byte to host (FEH)
Back to start of phase III
MCD02701

Figure 10-23
Bootstrap Loader Flowchart of Operating Mode 2

10.4.2.4 Selection of Operating Mode 3

Mode 3 is used to execute a custom software in the external FLASH memory at any start address. The header block, which has to be prepared and sent by the host for the activation of operating mode 3 has the structure as shown in **figure 10-24**.

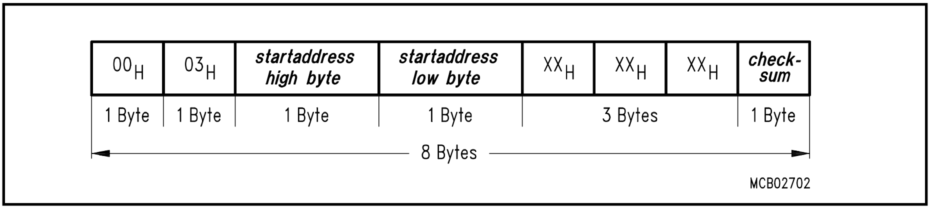


Figure 10-24
Header Block for Operating Mode 3

Mode 3 performs a direct program execution at a given ***startaddress***, which is sent in the header block. After sending the appropriate header block no further serial communication is necessary for this mode. The block diagram for the communication between the host and the MCU in mode 3 is shown in **figure 10-25**.

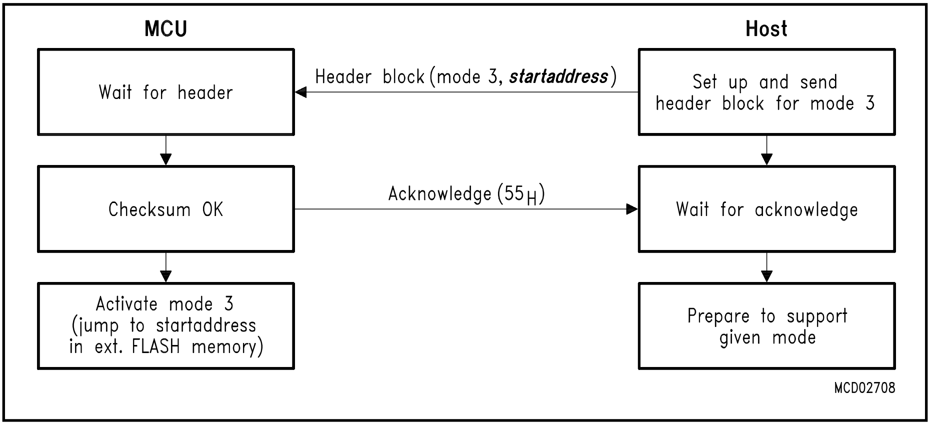

MCU
Host
Wait for header
Header block (mode 3, *startaddress*)
Set up and send header block for mode 3
Checksum OK
Acknowledge (55_H)
Wait for acknowledge
Activate mode 3 (jump to startaddress in ext. FLASH memory)
Prepare to support given mode
MCD02708

Figure 10-25
Operating Mode 3 Communication Structure

Note: The start address ***startaddress*** has to be greater than 200_H, because in the bootstrap mode the bootstrap loader overlaps the code address area of the external FLASH memory from 0000_H to $01FF_H$.

Figure 10-26 shows the bootstrap loader flowchart of operating mode 3 as flowchart:

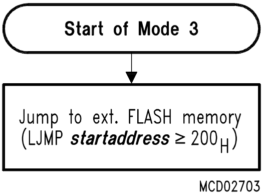


Figure 10-26
Bootstrap Loader Flowchart of Operating Mode 3

10.5 Description of the Bootstrap Loader Subroutines

This section describes the software of the bootstrap loader with its most important subroutines and start addresses, which can be used by the customer for own purposes. This technical reference is valid for C509-L parts with a stepping code “CA” or later.

Table 10-5 shows the subroutines of the bootstrap loader which can be used by the customer, when executing custom programs.

Table 10-5
Bootstrap Loader Subroutines - Survey

Address	Function	Registers	Description
$000E_H$	SendByte	In : **A** - Byte to send Out : None	Send a byte to the serial interface 0
0016_H	SendAckn	In : None Out : None	Send an acknowledge code (55_H) to the serial interface 0
$001B_H$	SendBlockErr	In : None Out : None	Send a block error code (FF_H) to the serial interface 0
0074_H	CalcBaudRate	In : **TH0**/**TL0** - measured value for test byte in **T0** Out : **R1**/**R2** - value for S0REL	Calculate the value for the 10 bit baud rate generator reload register S0REL of the serial interface 0 depending on the value of **T0** (**TH0**/**TL0**) for receiving the test byte (00_H)
$00A6_H$	CheckBaud	In : None Out : None	The complete baud rate synchronization: 1. Prepare the timer 0 for measurement 2. Wait for the test byte from host 3. Measure the time between the start bit and the stop bit 4. Calculate the baudrate from the value of timer 0 (T0) 5. Initialize the serial interface 0 by setting the baud rate and the serial parameters (8N2) 6. Send an acknowledge code (55_H) to the host

Table 10-5
Bootstrap Loader Subroutines - Survey (cont'd)

Address	Function	Registers	Description
00C7$_H$	GetBlock	In : **R7** - block length Out : None	Get an amount (in **R7**) of data bytes from the host and save it to a temporary buffer starting at address 70$_H$ and ending at address FF$_H$. The last received byte contains the checksum of the data block. Examine the sent checksum and send back a checksum error code, if it is incorrect.
00E9$_H$	CalcChks	In : **R0** - start address of the temporary buffer with length **R1** Out : **A** - calculated checksum	Calculate the checksum of the data block starting at address in **R0** with a length of **R1**. The checksum is saved in **A**.
00F0$_H$	CalcChksFLASH	In : **DPTR** - start address of the area in the FLASH memory **R4**/**R5** - length of the FLASH memory area Out : **R0** - calculated checksum #1 **R1** - calculated checksum #2	Calculate the special checksum of an area in the FLASH memory starting at address in **DPTR** with a length of **R4**/**R5** (high, low). The checksums #1 and #2 are saved in **R0** and **R1**.
0119$_H$	CheckFLASH	In : **R2**/**R3** - start address of corresponding FLASH memory block Out : **C** - carry flag	Check if the FLASH memory block starting at address **R2**/**R3** contains a functional custom program. 1. Check the ID bytes of the FLASH memory block 2. Calculate the checksum of the given memory block 3. Set the carry flag if no custom program exists
0161$_H$	Block2XRAM	In : **R0** - start address of the temporary buffer **R1** - length of the buffer **DPTR** - actual address in the XRAM Out : **DPTR** - actual address in the XRAM	Copy the contents of the temporary buffer starting at address in **R0** with a length of **R1** to the actual address (**DPTR**) in the XRAM. The data pointer **DPTR** is incremented automatically.

Table 10-5
Bootstrap Loader Subroutines - Survey (cont'd)

Address	Function	Registers	Description
0168$_H$	CheckHeader	In : **R0** - start address of the temporary buffer Out : **R1** - operating mode **R2**/**R3, DPTR** - header data **startaddress** **R4**/**R5** - header data **datalength** **R7** - header data **blocklength** **C** - Carry flag	Analyze the received header and save the header data into the corresponding registers. If the checked block is not of type **HEADER,** a block error code (FF$_H$) is sent to the host and the carry flag **C** is set.
018A$_H$	SendCheckErr	In : None Out : None	Send a checksum error code to the serial interface 0
018F$_H$	Mode0	In : **DPTR** - start address of XRAM to copy a custom program **R7** - length of the data blocks received via serial interface 0 Out : **DPTR** - actual address in the XRAM	Activate operating mode 0. The header data must be received or the registers **DPTR** and **R7** must be set manually, before this routine can work correctly.
01B1$_H$	Mode1	In : **DPTR** - address to start a custom program in the XRAM Out : None	Activate operating mode 1, that is to start custom program in the XRAM at address in **DPTR**. Attention: This routine does not return, when finished.
01BB$_H$	Mode2	In : **DPTR** - start address of the area in the FLASH memory **R4**/**R5** - length of the FLASH memory area Out : None	Activate operating mode 2, that is to calculate a special checksum of a FLASH memory area given by the start address in **DPTR** and the area length in **R4**/**R5**.
01D6$_H$	Mode3	In : **DPTR** - address to start a custom program in the FLASH memory Out : None	Activate operating mode 3, that is to start a custom program in the FLASH memory at address in **DPTR**. Attention: This routine does not return, when finished.

The bootstrap loader is sequentially working through the three phases. The addresses given in **table 10-6** allow the customer to step into the bootstrap loader in several phases.

Table 10-6
Further Bootstrap Loader Reference Addresses

Address	Function	Description
0000_H	Main	Start the complete bootstrap loader
0026_H	LookFLASH1	Phase I: Search for a functional custom program in the ext. FLASH memory sectors A and B
0035_H	LookFLASH2	Phase I: Search for a functional custom program only in the ext. FLASH memory sector B
0044_H	PrepSerial	Phase II: Initialize the serial interface 0 and synchronize it to the host baud rate
$004A_H$	WaitHeader	Phase III: Wait for the header to select the operating mode
$004F_H$	Header	Phase III: Check the header block information and save it in specific registers
0054_H	Jump2Mode	Phase III: Select and activate the operating mode given in register R1

11 Device Specifications

11.1 Absolute Maximum Ratings

Ambient temperature under bias (T_A)	– 40 to 110 °C
Storage temperature (T_{stg})	– 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	I 100 mA I
Power dissipation	1 W

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.

11.2 DC Characteristics

V_{CC} = 5 V + 10%, – 15%; V_{SS} = 0 V T_A = 0 to 70 °C   for the SAB-C509
T_A = – 40 to 85 °C   for the SAF-C509

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
Input low voltage (except $\overline{\text{EA}}$, $\overline{\text{RESET}}$, $\overline{\text{HWPD}}$)	V_{IL}	– 0.5	0.2 V_{CC} – 0.1	V	–
Input low voltage ($\overline{\text{EA}}$)	V_{IL1}	– 0.5	0.2 V_{CC} – 0.3	V	–
Input low voltage ($\overline{\text{HWPD}}$, $\overline{\text{RESET}}$)	V_{IL2}	– 0.5	0.2 V_{CC} + 0.1	V	–
Input low voltage (CMOS) (ports 0 - 9)	V_{ILC}	– 0.5	0.3 V_{CC}	V	–
Input high voltage (except $\overline{\text{RESET}}$, XTAL2 and $\overline{\text{HWPD}}$	V_{IH}	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V	–
Input high voltage to XTAL2	V_{IH1}	0.7 V_{CC}	V_{CC} + 0.5	V	–
Input high voltage to $\overline{\text{RESET}}$ and $\overline{\text{HWPD}}$	V_{IH2}	0.6 V_{CC}	V_{CC} + 0.5	V	–
Input high voltage (CMOS) (ports 0 - 9)	V_{IHC}	0.7 V_{CC}	V_{CC} + 0.5	V	–

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
CMOS input hysteresis (ports 1, 3 to 9)	V_{IHYS}	0.1	–	V	–
Output low voltage (ports 1, 2, 3, 4, 5, 6, 9)	V_{OL}	–	0.45	V	I_{OL} = 1.6 mA [1]
Output low voltage (ports ALE, $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$, $\overline{\text{RO}}$)	V_{OL1}	–	0.45	V	I_{OL} = 3.2mA [1]
Output high voltage (ports 1, 2, 3, 4, 5, 6, 9)	V_{OH}	2.4 0.9 V_{CC}	– –	V V	I_{OH} = –80 µA I_{OH} = –10 µA
Output high voltage (port 0 in external bus mode, ALE, $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$, $\overline{\text{RO}}$)	V_{OH1}	2.4 0.9 V_{CC}	– –	V V	I_{OH} = –800 µA [2] I_{OH} = –80 µA [2]
Output high voltage (CMOS) (ports 1, 2, 3, 4, 5, 6, 9)	V_{OHC}	0.9 V_{CC}	–	V	I_{OH} = –800 µA
Logic input low current (ports 1, 2, 3, 4, 5, 6, 9)	I_{IL}	– 10	– 70	µA	V_{IN} = 0.45 V
Logical 1-to-0 transition current (ports 1, 2, 3, 4, 5, 6, 9)	I_{TL}	– 65	– 650	µA	V_{IN} = 2 V
Input leakage current [7] (port 0, 7, 8, $\overline{\text{HWPD}}$)	I_{LI}	–	± 100	nA	$0.45 < V_{\text{IN}} < V_{\text{CC}}$
(port 0 in CMOS)			± 150	nA	$0.45 < V_{\text{IN}} < V_{\text{CC}}$ $T_{\text{A}} > 100\ ^{\text{o}}\text{C}$
Input leakage current ($\overline{\text{EA}}$, PRGEN) (ports 1, 2, 3, 4, 5, 6, 9 in CMOS)	I_{LIC}	–	± 1	µA	$0.45 < V_{\text{IN}} < V_{\text{CC}}$
Input low current to $\overline{\text{RESET}}$ for reset	I_{LI2}	– 10	–100	µA	V_{IN} = 0.45 V
Input low current (XTAL2)	I_{LI3}	–	– 15	µA	V_{IN} = 0.45 V
Input low current ($\overline{\text{PE}}$/SWD, OWE)	I_{LI4}	–	– 20	µA	V_{IN} = 0.45 V
Pin capacitance	C_{IO}	–	10	pF	f_{C} = 1 MHz $T_{\text{A}} = 25\ ^{\text{o}}\text{C}$
Overload current	I_{OV}	–	± 5	mA	[10] [11]

Parameter	Symbol	Limit Values		Unit	Test Condition
		typ. 12)	max.		
Power supply current:					
C509-L, Active mode, 12 MHz 8)	I_{CC}	9)	TBD	mA	V_{CC} = 5 V, 4)
C509-L, Active mode, 16 MHz 8)	I_{CC}	9)	TBD	mA	V_{CC} = 5 V, 4)
C509-L, Idle mode, 12 MHz 8)	I_{CC}	–	TBD	mA	V_{CC} = 5 V, 5)
C509-L, Idle mode, 16 MHz 8)	I_{CC}	9)	TBD	mA	V_{CC} = 5 V, 5)
C509-L, Slow down mode, 12 MHz	I_{CC}	–	TBD	mA	V_{CC} = 5 V, 6)
C509-L, Slow down mode, 16 MHz	I_{CC}	–	TBD	mA	V_{CC} = 5 V, 6)
C509-L, Power Down Mode	I_{PD}	5	50	µA	V_{CC} = 2...5.5

Notes :

1) Capacitive loading on ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL} of ALE and port 1, 3, 4, 5, 6, and 9. 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 PSEN/$\overline{\text{RDF}}$ 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:
EA = $\overline{\text{RESET}}$ = V_{CC}; Port0 = Port7 = Port8 = V_{CC}; XTAL1 = N.C.; XTAL2 = V_{SS}; $\overline{\text{PE}}$/SWD = OWE = V_{SS}; $\overline{\text{HWDP}}$ = V_{CC}; V_{AREF} = V_{CC}; V_{AGND} = V_{SS}; all other pins are disconnected.
Hardware power down mode current (I_{PD}) is measured with OWE =V_{CC} or V_{SS}.

4) I_{CC} (active mode) is measured with:
XTAL2 driven with t_R/t_F = 5 ns , V_{IL} = V_{SS} + 0.5 V, V_{IH} = V_{CC} – 0.5 V; XTAL1 = N.C.; EA = $\overline{\text{PE}}$/SWD= V_{CC} ; Port0 = Port7 = Port8 = V_{CC} ; $\overline{\text{HWPD}}$ = V_{CC} ; $\overline{\text{RESET}}$ = V_{SS}; all other pins are disconnected. I_{CC} would be slightly higher if a crystal oscillator is used.

5) I_{CC} (idle mode) is measured with all output pins disconnected and with all peripherals disabled;
XTAL2 driven with t_R/t_F = 5 ns, V_{IL} = V_{SS} + 0.5 V, V_{IH} = V_{CC} – 0.5 V; XTAL1 = N.C.; $\overline{\text{RESET}}$ = V_{CC}; $\overline{\text{HWPD}}$ = V_{CC} ; Port0 = Port7 = Port8 = V_{CC} ; $\overline{\text{EA}}$ = $\overline{\text{PE}}$/SWD = V_{SS}; all other pins are disconnected;

6) I_{CC} (slow down mode) is measured with all output pins disconnected and with all peripherals disabled;
XTAL2 driven with t_R/t_F = 5 ns, V_{IL} = V_{SS} + 0.5 V, V_{IH} = V_{CC} – 0.5 V; XTAL1 = N.C.; $\overline{\text{RESET}}$ = V_{CC}; $\overline{\text{HWPD}}$ = V_{CC} ; Port7 = Port8 = V_{CC} ; $\overline{\text{EA}}$ = $\overline{\text{PE}}$/SWD = V_{SS}; all other pins are disconnected;

7) Input leakage current for port 0 is measured with $\overline{\text{RESET}}$ = V_{CC}.

8) $I_{CC\,max}$ at other frequencies is given by:
active mode:TBD
idle mode:TBD
where f_{osc} is the oscillator frequency in MHz. I_{CC} values are given in mA and measured at V_{CC} = 5 V.

9) Typical power supply current ($I_{CC\,typ}$) with test conditiones as defined in note 4 and 5 is given by:
active mode, 12 MHz : 45 mA
active mode, 16 MHz : 72 mA
idle mode, 16 MHz : 29 mA

10) Overload conditions occur if the standard operating conditions are exeeded, 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.

11) Not 100% tested, guaranteed by design characterization.

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

11.3 A/D Converter Characteristics

T_A = 0 to 70 °C for the SAB-C509
T_A = – 40 to 85 °C for the SAF-C509
V_{CC} = 5 V + 10%, – 15%; V_{SS} = 0 V
4 V ≤ V_{AREF} ≤ V_{CC}+0.1 V ; V_{SS}-0.1 V ≤ V_{AGND} ≤ V_{SS}+0.2 V

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
Analog input voltage	V_{AIN}	V_{AGND}	V_{AREF}	V	[1]
Sample time	t_S	8 t_{IN}	512 t_{IN}		[2] see table below
Conversion time	t_{ADCC}	48 t_{IN}	832 t_{IN}		[3] see table below
Total unadjusted error	TUE	–	± 2	LSB	[4]
Internal resistance of reference voltage source	R_{AREF}	–	t_{ADC} / 250 - 0.25	kΩ	t_{ADC} in [ns] [5] [6]
Internal resistance of analog source	R_{ASRC}	–	t_S / 500 - 0.25	kΩ	t_S in [ns] [3] [6]
ADC input capacitance	C_{AIN}	–	50	pF	[6]

Notes see next page.

Clock calculation table

Conversion Clock Selection			Sample Clock Selection			Sample Time t_S	Conversion Time t_{ADCC}
ADCL1	ADCL0	Prescaler CCP	ADST1	ADST0	Prescaler SCP		
0	0	4	0 0 1 1	0 1 0 1	2	8 x t_{IN} 16 x t_{IN} 32 x t_{IN} 64 x t_{IN}	48 x t_{IN} 56 x t_{IN} 72 x t_{IN} 104 x t_{IN}
0	1	8	0 0 1 1	0 1 0 1	4	16 x t_{IN} 32 x t_{IN} 64 x t_{IN} 128 x t_{IN}	96 x t_{IN} 112 x t_{IN} 144 x t_{IN} 208 x t_{IN}
1	0	16	0 0 1 1	0 1 0 1	8	32 x t_{IN} 64 x t_{IN} 128 x t_{IN} 256 x t_{IN}	192 x t_{IN} 224 x t_{IN} 288 x t_{IN} 416 x t_{IN}
1	1	32	0 0 1 1	0 1 0 1	16	64 x t_{IN} 128 x t_{IN} 256 x t_{IN} 512 x t_{IN}	384 x t_{IN} 448 x t_{IN} 576 x t_{IN} 832 x t_{IN}

Further timing conditions : t_{ADC} min = 500 ns = CCP x CLP
t_{IN} = 1 / f_{OSC} = CLP
t_{SC} = t_{ADC} x SCP

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 $X000_H$ or $X3FF_H$, respectively.

2) During the sample time the input capacitance C_{AIN} can 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 and the time for the calibration. Values for the conversion clock f_{ADC} depend on programming and can be taken from the table below.

4) T_{UE} is tested at V_{AREF} = 5.0 V, V_{AGND} = 0 V, V_{CC} = 4.9 V. It is guaranteed by design characterization for all other voltages within the defined voltage range.
If an overload condition occurs on maximum 2 not selected 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.

11.4 AC Characteristics

V_{CC} = 5 V + 10%, – 15%; V_{SS} = 0 V T_A = 0 to 70 °C for the SAB-C509
T_A = – 40 to 85 °C for the SAF-C509

(C_L for port 0, ALE and $\overline{\text{PSEN}}$ 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 = 3.5 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
Address to valid instruction in	t_{LLIV}	–	75	–	2 CLP-50	ns
ALE to $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$	t_{LLPL}	10	–	TCL$_{Lmin}$-15	–	ns
$\overline{\text{PSEN}}$/$\overline{\text{RDF}}$ pulse width	t_{PLPH}	73	–	CLP+ TCL$_{Hmin}$-15	–	ns
$\overline{\text{PSEN}}$/$\overline{\text{RDF}}$ to valid instruction in	t_{PLIV}	–	38	–	CLP+ TCL$_{Hmin}$-50	ns
Input instruction hold after $\overline{\text{PSEN}}$/ $\overline{\text{RDF}}$	t_{PXIX}	0	–	0	–	ns
Input instruction float after $\overline{\text{PSEN}}$/ $\overline{\text{RDF}}$	t_{PXIZ} *)	–	15	–	TCL$_{Lmin}$-10	ns
Address valid after $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$	t_{PXAV} *)	20	–	TCL$_{Lmin}$-5	–	ns
Address to valid instruction in	t_{AVIV}	–	95	–	2 CLP+ TCL$_{Hmin}$-55	ns
Address float to $\overline{\text{PSEN}}$/$\overline{\text{RDF}}$	t_{AZPL}	- 5		- 5	–	ns

*) Interfacing the C509-L to devices with float times up to 20 ns is permissible. This limited bus contention will not cause any damage to port 0 drivers.

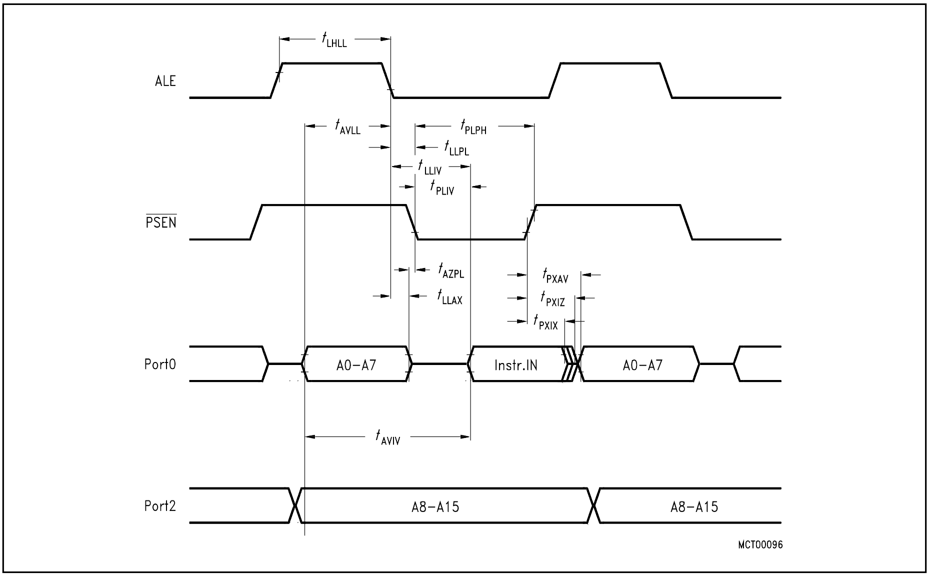
External Data Memory Characteristics

Parameter	Symbol	Limit Values				Unit
		16-MHz clock Duty Cycle 0.4 to 0.6		Variable Clock 1/CLP= 3.5 MHz to 16 MHz		
		min.	max.	min.	max.	
$\overline{\text{RD}}$ pulse width	t_{RLRH}	158	–	3 CLP-30	–	ns
$\overline{\text{WR}}$ pulse width	t_{WLWH}	158	–	3 CLP-30	–	ns
Address hold after ALE	t_{LLAX2}	48	–	CLP -15	–	ns
$\overline{\text{RD}}$ to valid data in	t_{RLDV}	–	100	–	2 CLP+ TCL_{Hmin}-50	ns
Data hold after $\overline{\text{RD}}$	t_{RHDX}	0		0	–	ns
Data float after $\overline{\text{RD}}$	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 $\overline{\text{WR}}$ or $\overline{\text{RD}}$	t_{LLWL}	73	103	CLP+ TCL_{Lmin}-15	CLP+ TCL_{Lmin}+15	ns
Address valid to $\overline{\text{WR}}$	t_{AVWL}	95	–	2 CLP-30	–	ns
$\overline{\text{WR}}$ or $\overline{\text{RD}}$ high to ALE high	t_{WHLH}	10	40	TCL_{Hmin}-15	TCL_{Hmin}+15	ns
Data valid to $\overline{\text{WR}}$ transition	t_{QVWX}	5	–	TCL_{Lmin}-20	–	ns
Data setup before $\overline{\text{WR}}$	t_{QVWH}	163	–	3 CLP+ TCL_{Lmin}-50	–	ns
Data hold after $\overline{\text{WR}}$	t_{WHQX}	5	–	TCL_{Hmin}-20	–	ns
Address float after $\overline{\text{RD}}$	t_{RLAZ}	–	0	–	0	ns

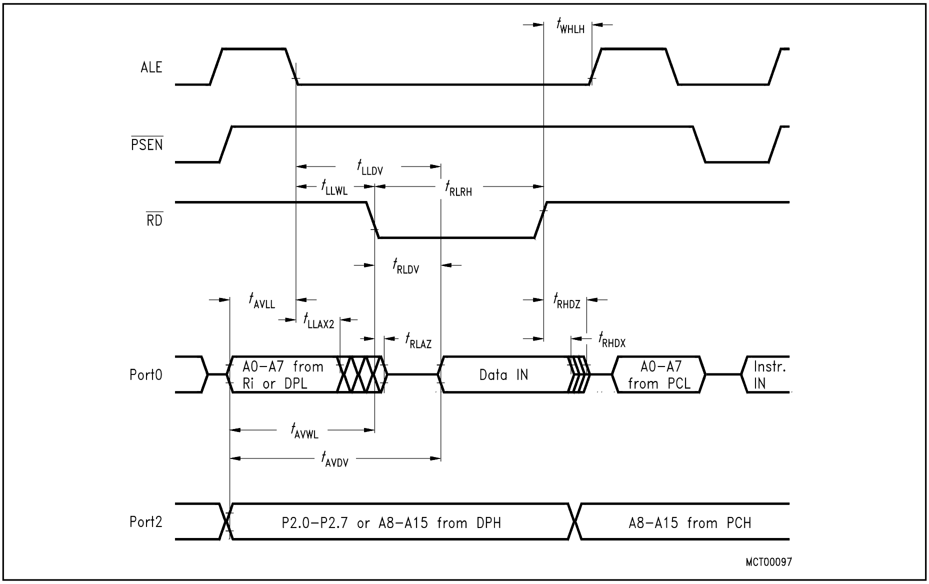
External Clock Drive XTAL2

Parameter	Symbol	CPU Clock = 16 MHz Duty cycle 0.4 to 0.6		Variable CPU Clock 1/CLP = 3.5 to 16 MHz		Unit
		min.	max.	min.	max.	
Oscillator period	CLP	62.5	62.5	62.5	285	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

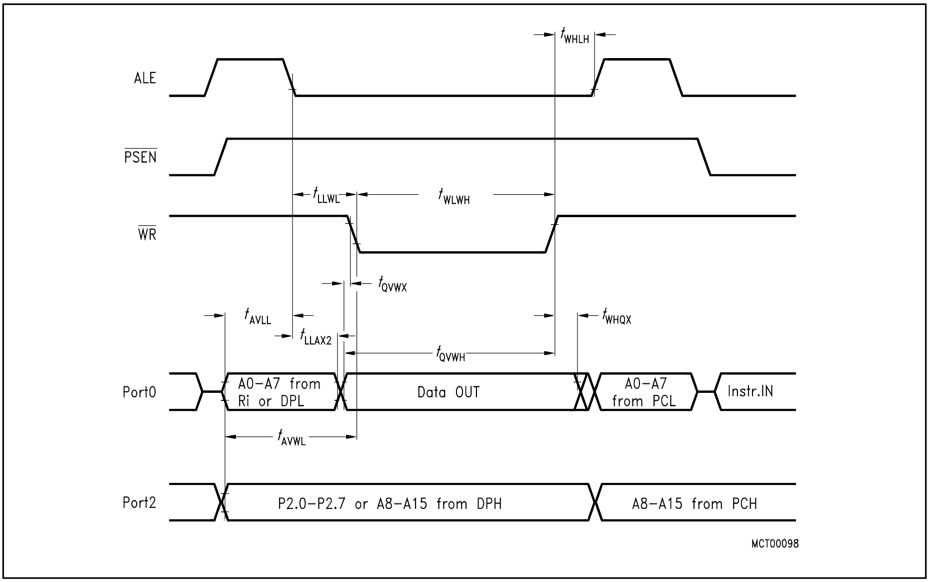
Note: The 16 MHz values in the tables are given as an example for a typical duty cycle variation of the oscillator clock from 0.4 to 0.6.



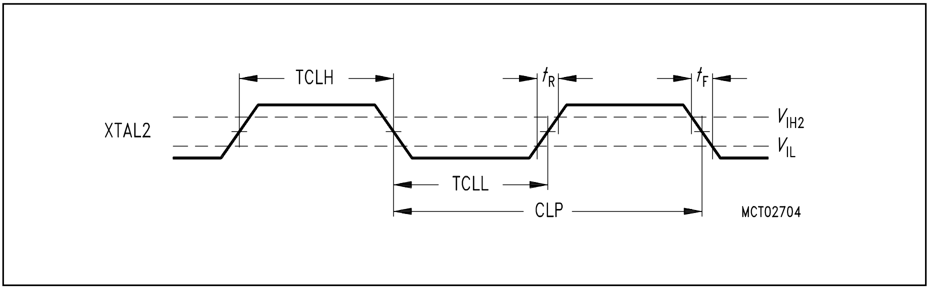
Program Memory Read Cycle



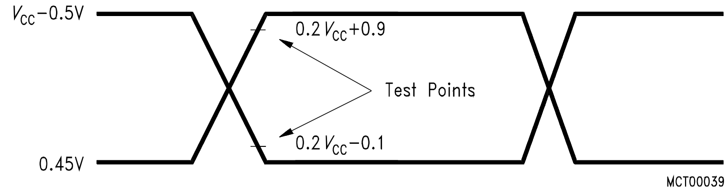
Data Memory Read Cycle



Data Memory Write Cycle

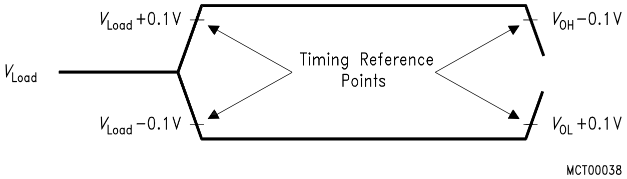


External Clock Drive Drive XTAL2



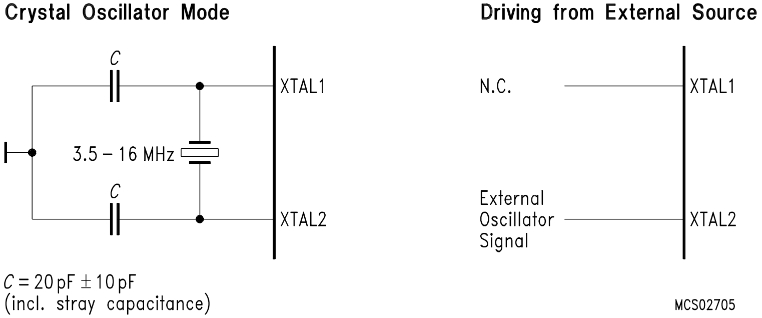
AC Inputs during testing are driven at V_{CC} - 0.5 V for a logic ’1’ and 0.45 V for a logic ’0’. Timing measurements are made at V_{IHmin} for a logic ’1’ and V_{ILmax} for a logic ’0’.

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} ≥ ± 20 mA

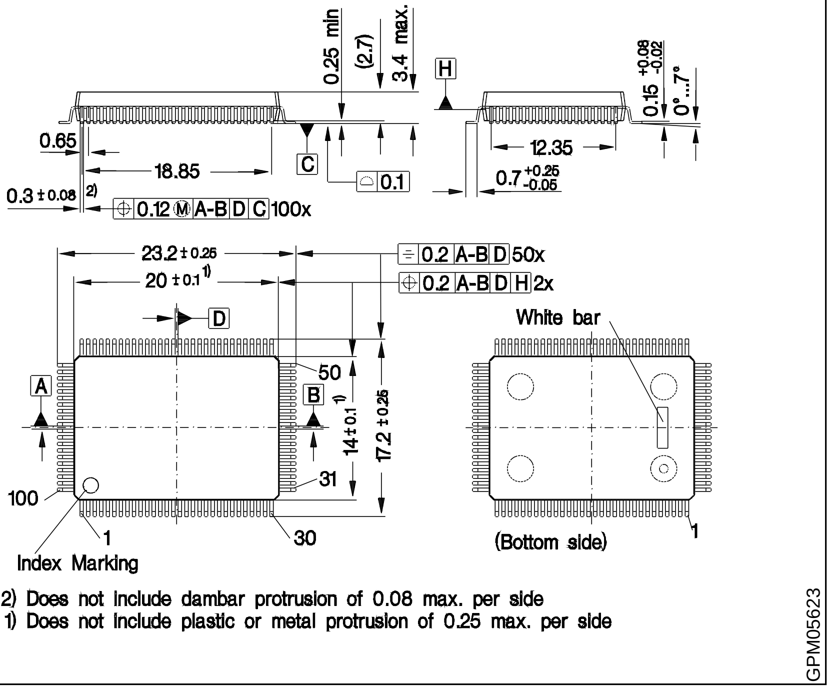
AC Testing: Float Waveforms



Recommended Oscillator Circuits for Crystal Oscillators up to 16 MHz

Package Outlines

P-MQFP-100-2
(Plastic Metric Quad Flat Package)



2) Does not include dambar protrusion of 0.08 max. per side
1) Does not include plastic or metal protrusion of 0.25 max. per side

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

SMD = Surface Mounted Device

Dimensions in mm

12 Index