SIEMENS

C541U

8-Bit CMOS Microcontroller

User's Manual 04.99

http://www.siemens.de/Semiconductor/

C541U User's Manual **Revision History :**		04.99
Previous Releases :		11.97(Original Version)
Page (previous version)	Page (new version)	Subjects (changes since last revision)
All chapters	All chapters	All references to C540U is removed.
All chapters	All chapters	V_{CC} is changed to V_{DD}.
1-2	1-2	USB feature list; Compliant to USB Specification "Rev 1.0".
1-2	1-2	Power supply voltage range changed to 4.25V to 5.5V.
1-2	1-2	Last line; "* P-SDIP-52 package ..." is added.
1-4	1-4	Figure 1-3; pin 2 is changed to ECAP.
1-5	1-5	Figure 1-4 is removed.
1-6 to 1-9	1-5 to 1-8	Table 1-1; column P-SDIP-52 is deleted and any references to P-SDIP-52 is also removed, the definition of pin 2 is changed to ECAP.
1-5	1-5	Table 1-2; functionality of Port 1 and 3 is added with the sentence "The pins with LED drive ... :
3-1	3-1	Correction in text: 8 KByte (only) on-chip OTP program memory.
3-5	3-5	Table 3-1; modified with addition of USB Low Speed registers.
3-7	3-7	Table 3-2; modified with addition of bit DRVI in GEPIR register and USB Low Speed registers.
3-8	3-8	Table 3-3; modified with addition of bit DRVIE in DPWDR register.
6-1	6-1	The first sentence of 6.1 is modified, to remove reference to P-SDIP-52 package.
6-34	6-34	Seventh line "The following sections ..." is added.
6-53	6-53	Last paragraph of 6.4.6.1 is added.
6-56	6-56	Sentence "Sequence of two ..." is added.
6-58	6-58	The definition of GEPIR register is added with bit DRVI.
6-61	6-61	The definition of DPWDR register is added with bit DRVIE and XVREG.
6-75	6-75	Section 6.4.8 Low Speed Mode is added.
6-78	6-88	Figure 6-39 is removed, as well as any references to it.
7-2	7-2	Figure 7-2 is modified to include USB Low Speed interrupts.
7-7	7-7	Definition of DRVIE, SUSPIE and DADDIE are added.
7-11	7-12	End of first paragraph; "(and GEPIR ..." is added.
7-12	7-13	Definition of DRVI, SUSP and DADD flag are added.
10-3	10-3	Figure 10-3 is removed.
10-4 to 10-5	10-3 to 10-4	Table 10-1; column P-SDIP-52 is removed.
Chapter 11	-	The whole chapter is moved to the C541U Data Sheet.

Edition 04.99
Published by Siemens AG,
Bereich Halbleiter, Marketing-
Kommunikation, Balanstraße 73,
81541 München
© Siemens AG 1999.
All Rights Reserved.

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**

1 Introduction 1-1
1.1 Pin Configuration 1-4
1.2 Pin Definitions and Functions 1-5

2 Fundamental Structure 2-1
2.1 CPU 2-2
2.2 CPU Timing 2-4

3 Memory Organization 3-1
3.1 Program Memory, "Code Space" 3-2
3.2 Data Memory, "Data Space" 3-2
3.3 General Purpose Registers 3-2
3.4 Special Function Registers 3-2

4 External Bus Interface 4-1
4.1 Accessing External Memory 4-1
4.1.1 Role of P0 and P2 as Data/Address Bus 4-1
4.1.2 Timing 4-3
4.1.3 External Program Memory Access 4-3
4.2 **PSEN**, Program Store Enable 4-3
4.3 Overlapping External Data and Program Memory Spaces 4-3
4.4 ALE, Address Latch Enable 4-4
4.5 Enhanced Hooks Emulation Concept 4-5

5 Reset and System Clock Operation 5-1
5.1 Hardware Reset Operation 5-1
5.2 Fast Internal Reset after Power-On 5-3
5.3 Hardware Reset Timing 5-5
5.4 Oscillator and Clock Circuit 5-6

6 On-Chip Peripheral Components 6-1
6.1 Parallel I/O 6-1
6.1.1 Port Structures 6-2
6.1.1.1 Basic Port Circuirty of Port 1 to 3 6-3
6.1.1.2 SSC Port Pins of Port 1 6-6
6.1.1.3 Port 0 Circuitry 6-8
6.1.1.4 Port 0 and Port 2 used as Address/Data Bus 6-9
6.1.2 Alternate Functions 6-10
6.1.3 Port Handling 6-12
6.1.3.1 Port Timing 6-12
6.1.3.2 Port Loading and Interfacing 6-13
6.1.3.3 Read-Modify-Write Feature of Ports 1,2 and 3 6-14
6.2 Timers/Counters 6-15
6.2.1 Timer/Counter 0 and 1 6-15
6.2.1.1 Timer/Counter 0 and 1 Registers 6-16
6.2.1.2 Mode 0 6-19
6.2.1.3 Mode 1 6-20
6.2.1.4 Mode 2 6-21

Table of Contents **Page**

6.2.1.5 Mode 3 6-22
6.3 SSC Interface 6-23
6.3.1 General Operation of the SSC 6-24
6.3.2 Enable/Disable Control 6-24
6.3.3 Baudrate Generation (Master Mode only) 6-25
6.3.4 Write Collision Detection 6-25
6.3.5 Master/Slave Mode Selection 6-26
6.3.6 Data/Clock Timing Relationships 6-27
6.3.6.1 Master Mode Operation 6-27
6.3.6.2 Slave Mode Operation 6-28
6.3.7 Register Description 6-29
6.4 USB Module 6-34
6.4.1 Transfer Modes 6-35
6.4.2 USB Memory Buffer Modes 6-36
6.4.2.1 Overview 6-36
6.4.2.2 Single Buffer Mode 6-37
6.4.2.2.1 USB Write Access 6-37
6.4.2.2.2 USB Read Access 6-39
6.4.2.3 Dual Buffer Mode 6-42
6.4.3 USB Memory Buffer Organization 6-49
6.4.4 USB Memory Buffer Address Generation 6-50
6.4.5 Initialization of USB Module 6-51
6.4.6 Control Transfer 6-53
6.4.6.1 Setup Stage 6-53
6.4.6.2 Data Stage 6-53
6.4.6.3 Status Stage 6-53
6.4.7 Register Set 6-54
6.4.7.1 Global Registers 6-55
6.4.7.2 Device Registers 6-59
6.4.7.3 Endpoint Registers 6-66
6.4.8 Low Speed Mode 6-75
6.4.8.1 Initialization 6-75
6.4.8.2 Transfer Modes 6-76
6.4.8.3 Control Transfer 6-76
6.4.8.3.1 Setup Stage 6-76
6.4.8.3.2 Data Stage 6-76
6.4.8.3.3 Status Stage 6-76
6.4.8.4 Interrupt Transfer 6-77
6.4.8.5 Register Set 6-78
6.4.8.6 USB Low Speed Interrupts 6-83
6.4.9 On-Chip USB Transceiver 6-84
6.4.10 Detection of Connected Devices 6-86
6.4.11 Detach / Attach Detection 6-87
6.4.11.1 Self-Powered Mode 6-87
6.4.11.2 Bus-Powered Mode 6-87

Table of Contents **Page**

7 Interrupt System 7-1
7.1 Interrupt Registers 7-4
7.1.1 Interrupt Enable Registers 7-4
7.1.2 Interrupt Request / Control Flags 7-10
7.1.3 Interrupt Prioritiy Registers 7-16
7.2 Interrupt Priority Level Structure 7-17
7.3 How Interrupts are Handled 7-18
7.4 External Interrupts 7-20
7.5 Interrupt Response Time 7-22

8 Fail Safe Mechanisms 8-1
8.1 Programmable Watchdog Timer 8-1
8.1.1 Input Clock Selection 8-2
8.1.2 Watchdog Timer Control / Status Flags 8-3
8.1.3 Starting the Watchdog Timer 8-4
8.1.4 Refreshing the Watchdog Timer 8-4
8.1.5 Watchdog Reset and Watchdog Status Flag 8-4
8.2 Oscillator Watchdog Unit 8-5
8.2.1 Functionality of the Oscillator Watchdog Unit 8-6
8.2.2 Fast Internal Reset after Power-On 8-7

9 Power Saving Modes 9-1
9.1 Idle Mode 9-3
9.1.1 Entering Idle Mode 9-4
9.1.2 Exit from Idle Mode 9-4
9.2 Power Down Mode 9-5
9.2.1 Entering Power Down Mode 9-6
9.2.2 Exit from Power Down Mode 9-7
9.2.2.1 Exit via Pin P3.2/INT0 9-8
9.2.2.2 Exit via UBS Bus 9-8

10 OTP Memory Operation 10-1
10.1 Programming Configuration 10-1
10.2 Pin Configuration 10-2
10.3 Pin Definitions 10-3
10.4 Programming Mode Selection 10-5
10.4.1 Basic Programming Mode Selection 10-5
10.4.2 OTP Memory Access Mode Selection 10-6
10.5 Program / Read OTP Memory Bytes 10-7
10.6 Lock Bits Programming / Read 10-9
10.7 Access of Version Bytes 10-11
10.8 OTP Verify with Protection Level 1 10-12

11 Index 12-1

1 Introduction

The C541U is a member of the Siemens C500 family of 8-bit microcontrollers They are fully compatible to the standard 80C51 architecture.

The C541U especially provides an on-chip USB module compliant to the USB specification, which is capable to operate either in low or full speed mode. The five endpoints can be easily controlled by the CPU via special function registers. Due to the on-chip USB transceiver circuits the C541U can be directly connected to the USB bus.

Figure 1-1 shows the different functional units of the C541U and **figure 1-2** shows the simplified logic symbol of the C541U.

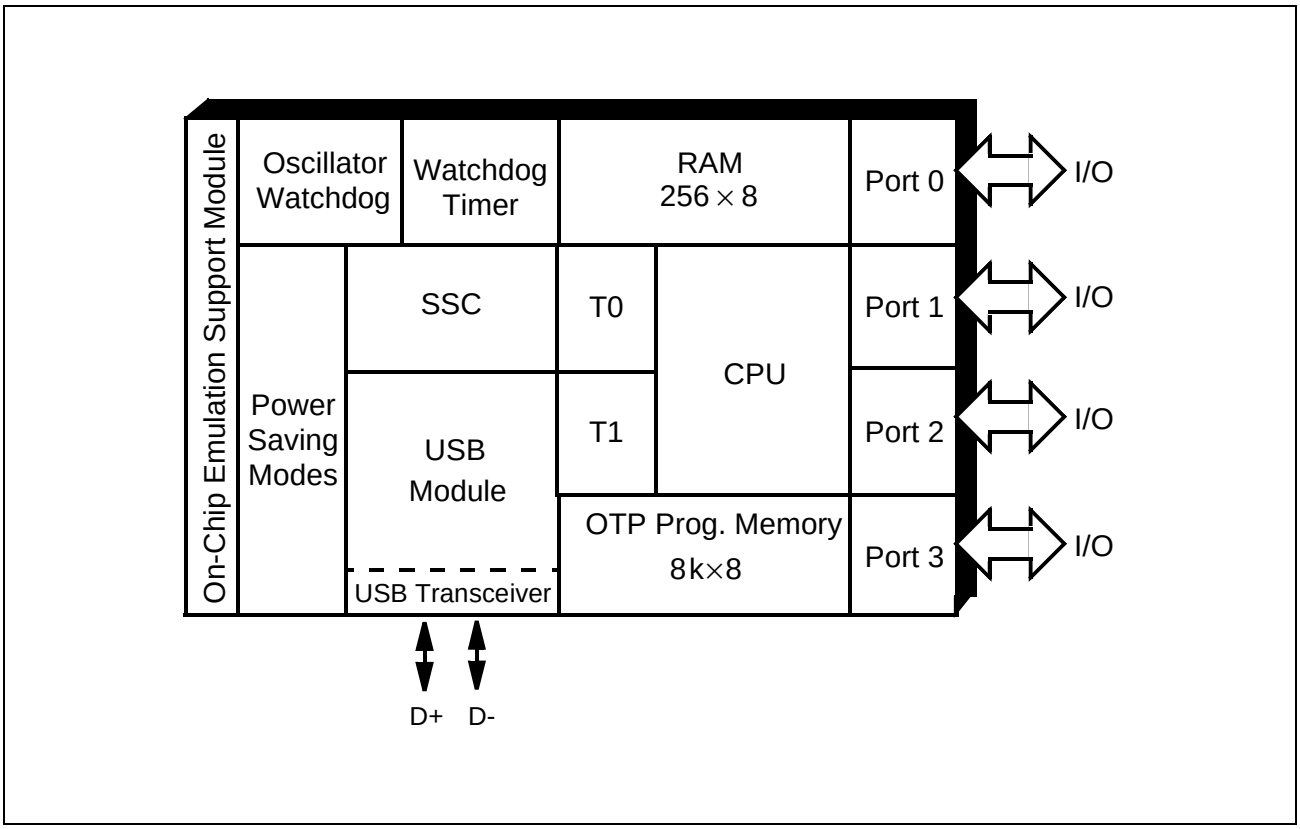


Figure 1-1
C541U Functional Units

Listed below is a summary of the main features of the C541U :

- Enhanced 8-bit C500 CPU
  - Full software/toolset compatible to standard 80C51/80C52 microcontrollers
- 12 MHz external operating frequency
  - 500 ns instruction cycle
- Built-in PLL for USB synchronization
- On-chip OTP program memory
  - 8K byte
  - Alternatively up to 64K byte external program memory
  - Optional memory protection
- Up to 64K byte external data memory
- 256 byte on-chip RAM
- Four parallel I/O ports
  - P-LCC-44 package : three 8-bit ports and one 6-bit port
  - P-SDIP-52* package : four 8-bit ports
  - LED current drive capability for 3 pins (10 mA)
- Two 16-bit timer/counters (C501 compatible)
- On-chip USB module
  - Compliant to USB specification Rev1.0
  - Full speed or low speed operation
  - Five endpoints : one bidirectional control endpoint
four versatile programmable endpoints
  - Registers are located in special function register area
  - On-chip USB transceiver
- SSC synchronous serial interface (SPI compatible)
  - Master and slave capable
  - Programmable clock polarity / clock-edge to data phase relation
  - LSB/MSB first selectable
  - 1.5 MBaud transfer rate at 12 MHz operating frequency
- 7 interrupt sources (2 external, 5 internal with 2 USB interrupts) selectable at 2 priority levels
- Enhanced fail safe mechanisms
  - Programmable watchdog timer
  - Oscillator watchdog
- Power saving modes
  - idle mode
  - software power down mode with wake-up capability through $\overline{\text{INT0}}$ pin or USB
- On-chip emulation support logic (Enhanced Hooks Technology ™)
- P-LCC-44 and P-SDIP-52* packages
- Power supply voltage range : 4.25V to 5.5V
- Temperature Range : SAB-C541U T_A = 0 to 70 °C

* P-SDIP-52 package is available on specific request from customer

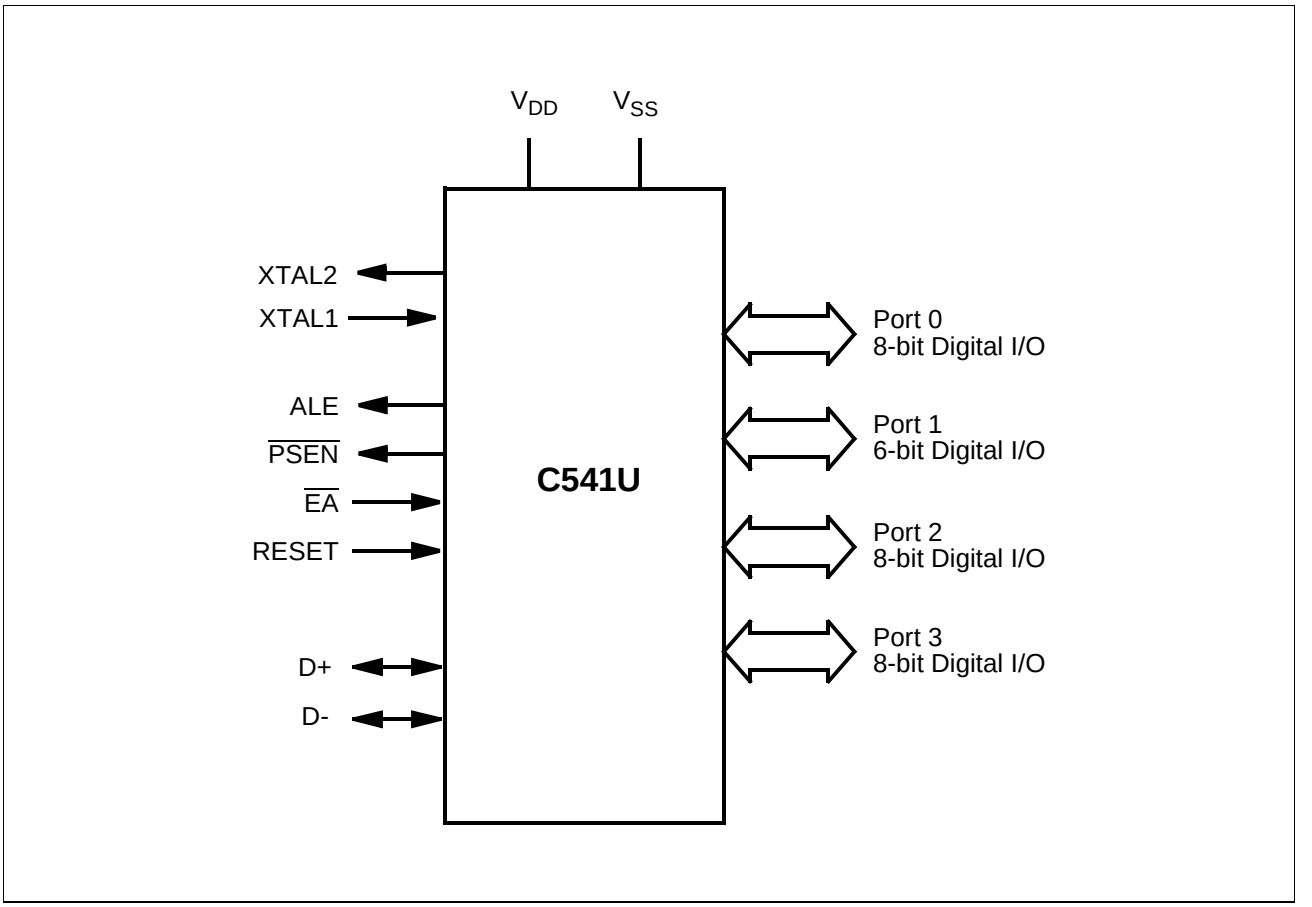


Figure 1-2
Logic Symbol

1.1 Pin Configuration

This section describes the pin configurations of the C541U in the P-LCC-44 package.

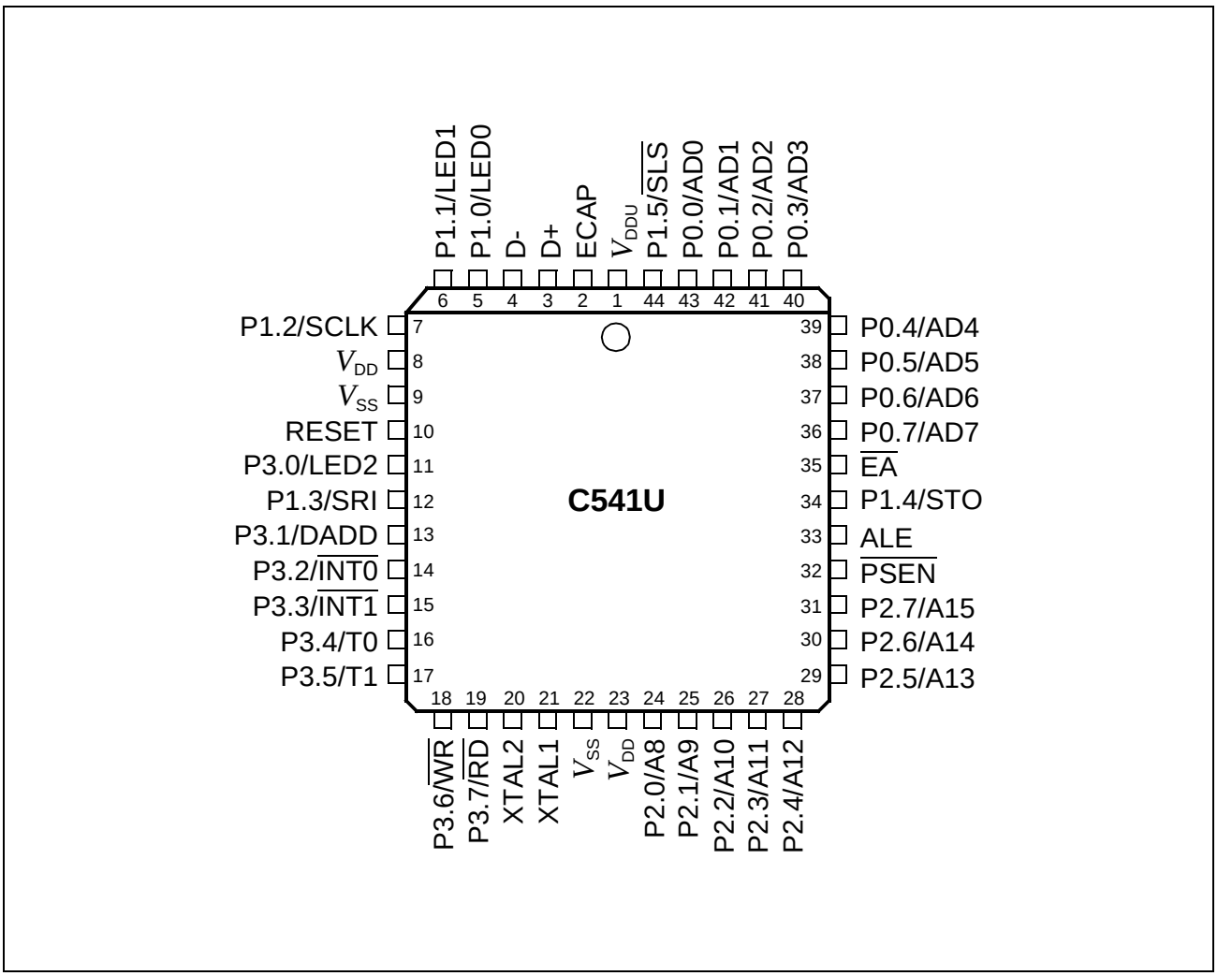


Figure 1-3
Pin Configuration(top view)

1.2 Pin Definitions and Functions

This section describes all external signals of the C541U with its function.

Table 1-1
Pin Definitions and Functions

Symbol	Pin Numbers P-LCC-44	I/O*)	Function
D+	3	I/O	**USB D+ Data Line** The pin D+ can be directly connected to USB cable (transceiver is integrated on-chip).
D-	4	I/O	**USB D- Data Line** The pin D- can be directly connected to USB cable (transceiver is integrated on-chip).
P1.0 - P1.4	5 - 7, 12, 34, 44 5 6 7 12 34 44	I/O	**Port 1** is an 6-bit quasi-bidirectional I/O port with internal pullup resistors. Port 1 pins that have 1's written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. As inputs, port 1 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 1 also contains two outputs with LED drive capability as well as the four pins of the SSC. The pins with LED drive capability are able to sink current up to 10 mA. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate (except when used for the compare functions). The secondary functions are assigned to the port 1 pins as follows : P1.0 / LED0 LED0 output P1.1 / LED1 LED1 output P1.2 / SCLK SSC Master Clock Output / SSC Slave Clock Input P1.3 / SRI SSC Receive Input P1.4 / $\overline{\text{STO}}$ SSC Transmit Output P1.5 / $\overline{\text{SLS}}$ SSC Slave Select Inp.
RESET	10	I	**RESET** A high level on this pin for the duration of two machine cycles while the oscillator is running resets the C541U. A small internal pulldown resistor permits power-on reset using only a capacitor connected to V_{DD}.

*) I = Input
O = Output

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

Symbol	Pin Numbers	I/O*)	Function
	P-LCC-44		
P3.0 - P3.7	11, 13 - 19	I/O	**Port 3** is an 8-bit quasi-bidirectional I/O port with internal pullup resistors. Port 3 pins that have 1's written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. As inputs, port 3 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 3 also contains the interrupt, timer, serial port and external memory strobe pins that are used by various options. The pin with LED drive capability is able to sink current up to 10 mA. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the pins of port 3, as follows: 11 P3.0 / LED2 LED2 output 13 P3.1 / DADD Device attached input 14 P3.2 / $\overline{\text{INT0}}$ External interrupt 0 input / timer 0 gate control input 15 P3.3 / $\overline{\text{INT1}}$ External interrupt 1 input / timer 1 gate control input 16 P3.4 / T0 Timer 0 counter input 17 P3.5 / T1 Timer 1 counter input 18 P3.6 / $\overline{\text{WR}}$ $\overline{\text{WR}}$ control output; latches the data byte from port 0 into the external data memory 19 P3.7 / $\overline{\text{RD}}$ $\overline{\text{RD}}$ control output; enables the external data memory
XTAL2	20	–	**XTAL2** is the output of the inverting oscillator amplifier. This pin is used for the oscillator operation with crystal or ceramic resonator.
XTAL1	21	–	**XTAL1** is the input to the inverting oscillator amplifier and input to the internal clock generator circuits. To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected. Minimum and maximum high and low times as well as rise/fall times specified in the AC characteristics must be observed.

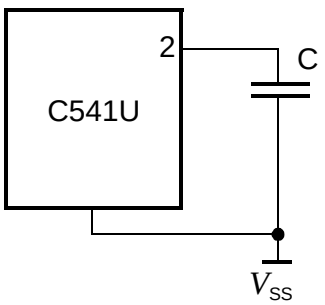
*) I = Input
O = Output

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

Symbol	Pin Numbers	I/O*)	Function
	P-LCC-44		
P2.0 - P2.7	24 - 31	I/O	**Port 2** is an 8-bit quasi-bidirectional I/O port with internal pullup resistors. Port 2 pins that have 1's written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. As inputs, port 2 pins being externally pulled low will source current (I_{IL}, in the DC characteristics) because of the internal pullup resistors. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullup resistors when issuing 1's. During accesses to external data memory that use 8-bit addresses (MOVX @Ri), port 2 issues the contents of the P2 special function register.
$\overline{\text{PSEN}}$	32	O	The **Program Store Enable** output is a control signal that enables the external program memory to the bus during external fetch operations. It is activated every three oscillator periods except during external data memory accesses. The signal remains high during internal program execution.
ALE	33	O	The **Address Latch enable** output is used for latching the address into external memory during normal operation. It is activated every three oscillator periods except during an external data memory access.
$\overline{\text{EA}}$	35	I	**External Access Enable** When held high, the C541U executes instructions from the internal OTP program memory as long as the PC is less than 2000_H for the C541U. When held low, the C541U fetches all instructions from external program memory. For the C541U-L this pin must be tied low.
P0.0 - P0.7	43 - 36	I/O	**Port 0** is an 8-bit open-drain bidirectional I/O port. Port 0 pins that have 1's written to them float, and in that state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory. In this application it uses strong internal pullup resistors when issuing 1's.

*) I = Input
O = Output

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

Symbol	Pin Numbers	I/O*)	Function
	P-LCC-44		
ECAP	2	–	**External Capacitor** This pin is required to be connected to an external capacitor which is connected to V_{SS}. The value of the capacitor is 6 nF. 
V_{DDU}	1	–	**Supply voltage** for the on-chip USB transceiver circuitry.
V_{DD}	8, 23	–	**Supply voltage** for ports and internal logic circuitry during normal, idle, and power down mode.
V_{SS}	9, 22	–	**Ground** (0V) during normal, idle, and power down mode.

*) I = Input
O = Output

2 Fundamental Structure

The C541U is fully compatible to the architecture of the standard 8051/C501 microcontroller family. While maintaining the typical architectural characteristics of the C501, the C541U incorporates a SSC synchronous serial interface, a versatile USB module as well as some enhancements in the Fail Save Mechanism Unit.

Figure 2-1 shows a block diagram of the C541U.

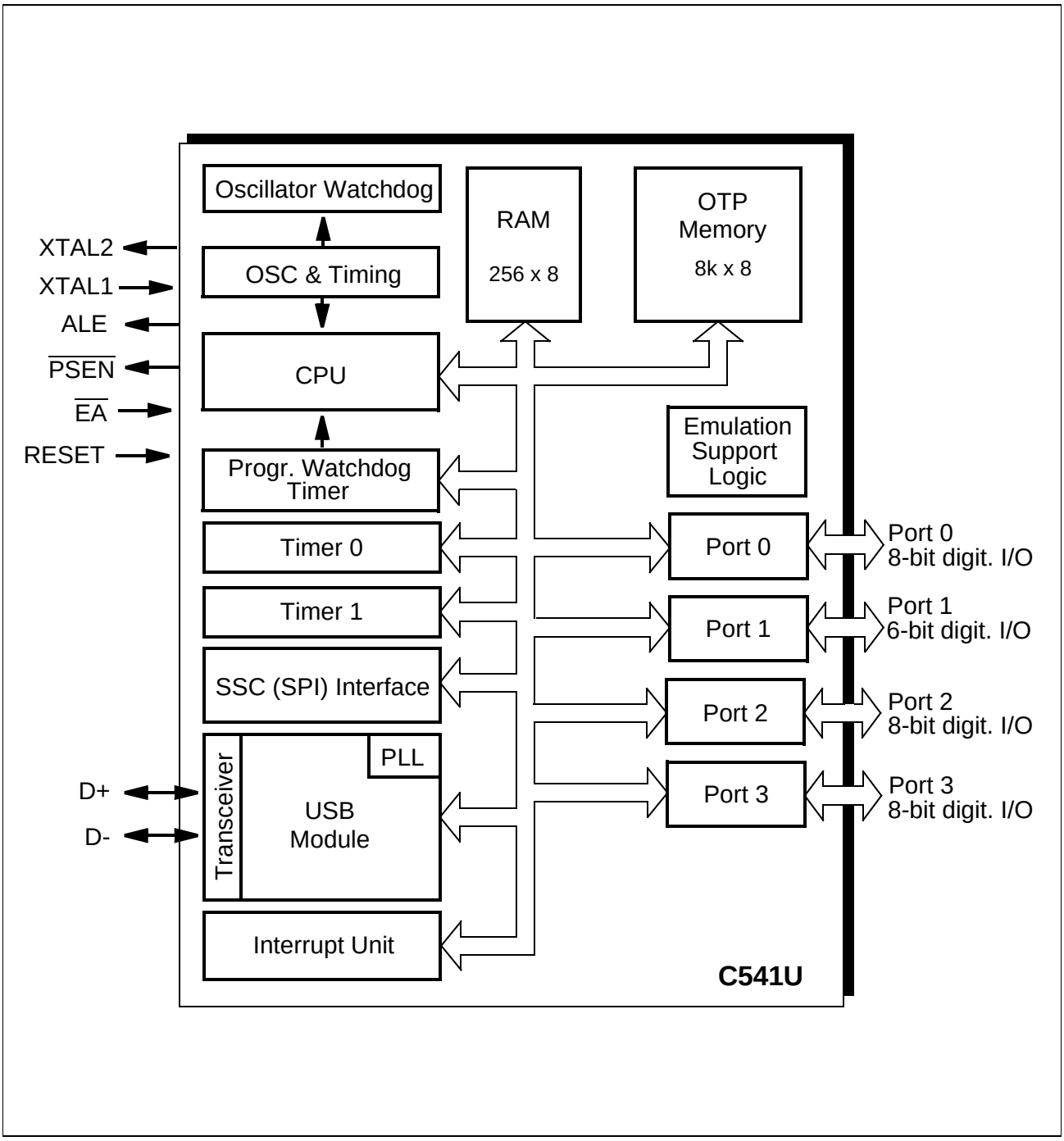


Figure 2-1
Block Diagram of the C541U

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 C541U 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 12 MHz clock, 58% of the instructions execute in 500 ns.

The CPU (Central Processing Unit) of the C541U consists of the instruction decoder, the arithmetic section and the program control section. Each program instruction is decoded by the instruction decoder. This unit generates the internal signals controlling the functions of the individual units within the CPU. They have an effect on the source and destination of data transfers and control the ALU processing.

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

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

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

Accumulator

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

Program Status Word

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

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

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

Bit	Function
CY	**Carry Flag** Used by arithmetic instruction.
AC	**Auxiliary Carry Flag** Used by instructions which execute BCD operations.
F0	**General Purpose Flag**
RS1 RS0	**Register Bank select control bits** These bits are used to select one of the four register banks.

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

OV	**Overflow Flag** Used by arithmetic instruction.
F1	**General Purpose Flag**
P	**Parity Flag** Set/cleared by hardware after each instruction to indicate an odd/even number of "one" bits in the accumulator, i.e. even parity.

B Register

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

Stack Pointer

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

2.2 CPU Timing

The C541U has no clock prescaler. Therefore, a machine cycle of the C541U consists of 6 states (6 oscillator periods). Each state is devided into a phase 1 half and a phase 2 half. Thus, a machine cycle consists of 6 oscillator periods, numbererd S1P1 (state 1, phase 1) through S6P2 (state 6, phase 2). Each state lasts one oscillator period. Typically, arithmetic and logic operations take place during phase 1 and internal register-to-register transfers take place during phase 2.

The diagrams in **figure 2-2** show the fetch/execute timing related to the internal states and phases. Since these internal clock signals are not user-accessible, the XTAL2 oscillator signals and the ALE (address latch enable) signal are shown for external reference. ALE is normally activated twice during each machine cycle: once during S1P2 and S2P1, and again during S4P2 and S5P1.

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

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

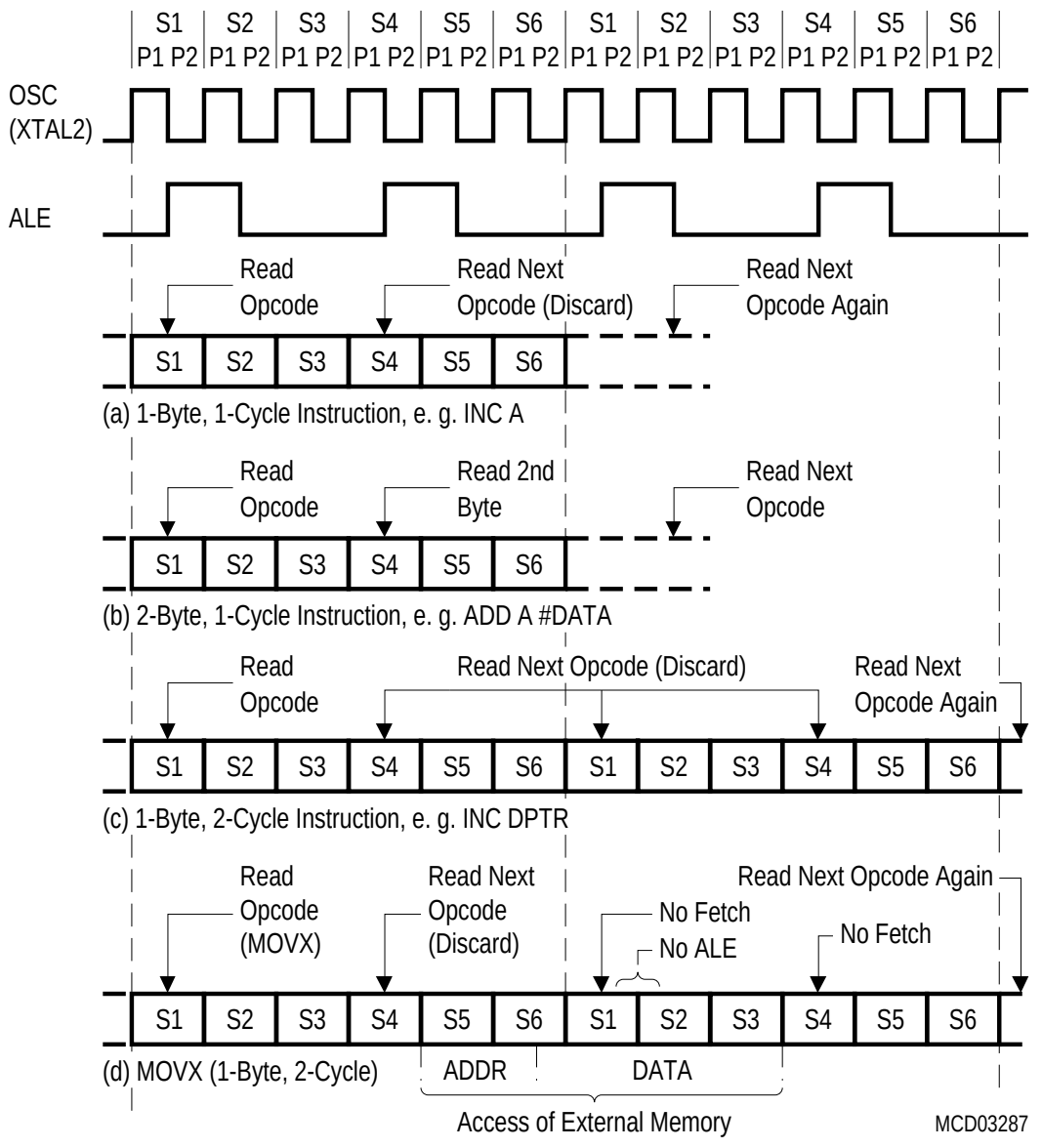


Figure 2-2
Fetch Execute Sequence

3 Memory Organization

The C541U CPU manipulates operands in the following four address spaces:

- 8 KByte on-chip OTP program memory
- Totally up to 64 Kbyte internal/external program memory
- up to 64 Kbyte of external data memory
- 256 bytes of internal data memory
- a 128 byte special function register area

Figure 3-1 illustrates the memory address spaces of the C541U.

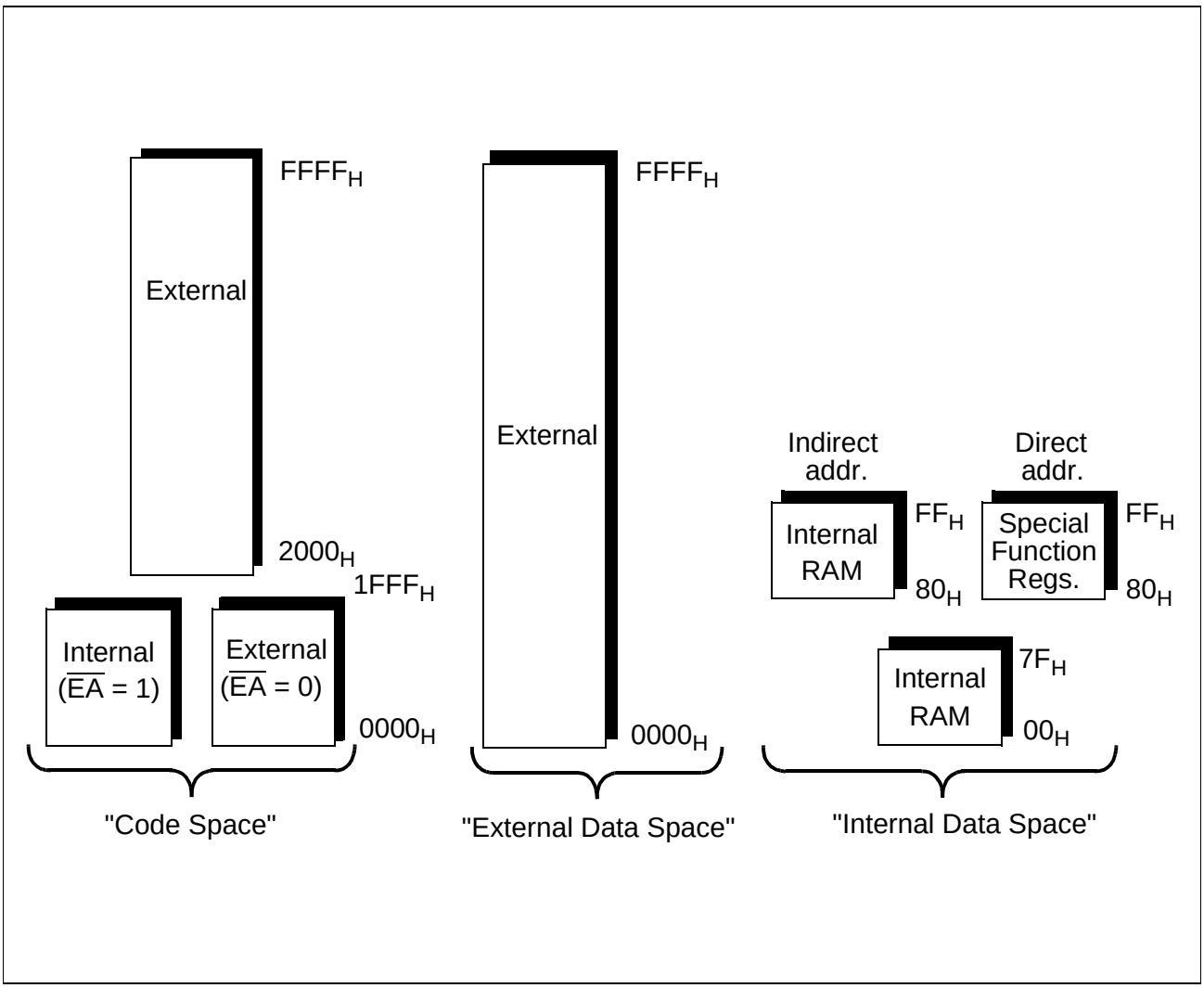


Figure 3-1
C541U Memory Map

3.1 Program Memory, "Code Space"

The C541U has 8 Kbyte of OTP program memory which can be externally expanded up to 64 Kbytes. If the $\overline{EA}$ pin is held high, the C541U executes program code out of the internal OTP program memory unless the program counter address exceeds $1FFF_H$. Address locations 2000_H through $FFFF_H$ are then fetched from the external program memory. If the $\overline{EA}$ pin is held low, the C541U fetches all instructions from an external 64K byte program memory.

3.2 Data Memory, "Data Space"

The data memory address space consists of an internal and an external memory space. The internal data memory is divided into three physically separate and distinct blocks : the lower 128 bytes of RAM, the upper 128 bytes of RAM, and the 128 byte special function register (SFR) area.

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

The external data memory can be expanded up to 64 Kbyte and can be accessed by MOVX instructions that use a 16-bit or an 8-bit address.

Note : The registers of the USB module are accessed through special function registers in the SFR area.

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 Special Function Registers

The registers, except the program counter and the four general purpose register banks, reside in the special function register area. The special function register area consists of two portions: the

standard special function register area and the mapped special function register area. One special function register of the C541U (PCON1) is located in the mapped special function register area. All other SFRs are located in the standard special function register area.

For accessing PCON1 in the mapped special function register area, bit RMAP in special function register SYSCON must be set.

Special Function Register SYSCON (Address $B1_H$) **Reset Value : $XX10XXXX_B$**

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

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. RMAP = 1 : The access to the mapped special function register area (PCON1) is enabled.

As long as bit RMAP is set, a mapped special function register 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.

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

The 75 special function registers (SFRs) in the SFR area include pointers and registers that provide an interface between the CPU and the other on-chip peripherals. The SFRs of the C541U are listed in **table 3-1** and **table 3-2**. In **table 3-1** they are organized in groups which refer to the functional blocks of the C541U. **Table 3-2** illustrates the contents of the SFRs in numeric order of their addresses.

Table 3-1
Special Function Registers - Functional Blocks

Block	Symbol	Name	Address	Contents after Reset
CPU	ACC	Accumulator	**$E0_H$** [1]	00_H
	B	B Register	**$F0_H$** [1]	00_H
	DPH	Data Pointer, High Byte	83_H	00_H
	DPL	Data Pointer, Low Byte	82_H	00_H
	PSW	Program Status Word Register	**$D0_H$** [1]	00_H
	SP	Stack Pointer	81_H	07_H
	VR0	Version Register 0	FC_H	$C5_H$
	VR1	Version Register 1	FD_H	$C1_H$
	VR2	Version Register 2	FE_H	YY_H [3]
	SYSCON	System Control Register	$B1_H$	$XX10XXXX_B$ [2]
Interrupt System	IEN0	Interrupt Enable Register 0	**$A8_H$** [1]	$0XXX0000_B$ [2]
	IEN1	Interrupt Enable Register 1	$A9_H$	$XXXXX000_B$ [2]
	IP0	Interrupt Priority Register 0	**$B8_H$** [1]	$XXXX0000_B$ [2]
	IP1	Interrupt Priority Register 1	$B9_H$ [1]	$XXXXX000_B$ [2]
	ITCON	External Interrupt Trigger Condition Register	$9A_H$	$XXXX1010_B$ [2]
Ports	P0	Port 0	**80_H** [1]	FF_H
	P1	Port 1	**90_H** [1]	FF_H
	P2	Port 2	**$A0_H$** [1]	FF_H
	P3	Port 3	**$B0_H$** [1]	FF_H
Timer 0 / Timer 1	TCON	Timer 0/1 Control Register	**88_H** [1]	00_H
	TH0	Timer 0, High Byte	$8C_H$	00_H
	TH1	Timer 1, High Byte	$8D_H$	00_H
	TL0	Timer 0, Low Byte	$8A_H$	00_H
	TL1	Timer 1, Low Byte	$8B_H$	00_H
	TMOD	Timer Mode Register	89_H	00_H
SSC Interface	SSCCON	SSC Control Register	93_H [1]	07_H
	STB	SSC Transmit Buffer	94_H	XX_H [2]
	SRB	SSC Receive Register	95_H	XX_H [2]
	SCF	SSC Flag Register	AB_H [1]	$XXXXXX00_B$ [2]
	SCIEN	SSC Interrupt Enable Register	AC_H	$XXXXXX00_B$ [2]
	SSCMOD	SSC Mode Test Register	96_H	00_H
Watchdog	WDCON	Watchdog Timer Control Register	**$C0_H$** [1]	$XXXX0000_B$ [2]
	WDTREL	Watchdog Timer Reload Register	86_H	00_H

1) Bit-addressable special function registers
2) “X“ means that the value is undefined and the location is reserved
3) The content of this SFR varies with the actual of the step C541U (eg. 01_H for the first step)
4) This SFR is located in the mapped SFR area. For accessing this SFR, bit RMAP in SFR SYSCON must be set.

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

Block	Symbol	Name	Address	Contents after Reset
Pow. Sav. Modes	PCON	Power Control Register	87_H	$X00X0000_B$ 2)
	PCON1	Power Control Register 1	88_H 4)	$0XX0XXXX_B$ 2)
USB Module	EPSEL	USB Endpoint Select Register	$D2_H$	80_H
	USBVAL	USB Data Register	$D3_H$	00_H
	ADROFF	USB Address Offset Register	$D4_H$	00_H 2)
	GEPIR	USB Global Endpoint Interrupt Request Reg.	$D6_H$	00_H
	DCR	USB Device Control Register	$C1_H$	$000X0000_B$
	DPWDR	USB Device Power Down Register	$C2_H$	00_H
	DIER	USB Device Interrupt Control Register	$C3_H$	00_H
	DIRR	USB Device Interrupt Request Register	$C4_H$	00_H
	FNRL	USB Frame Number Register, Low Byte	$C6_H$	XX_H
	FNRH	USB Frame Number Register, High Byte	$C7_H$	$00000XXX_B$
	EPBCn 1)	USB Endpoint n Buffer Control Register	$C1_H$	00_H
	EPBSn 1)	USB Endpoint n Buffer Status Register	$C2_H$	20_H
	EPIEn 1)	USB Endpoint n Interrupt Enable Register	$C3_H$	00_H
	EPIRn 1)	USB Endpoint n Interrupt Request Register	$C4_H$	10_H 3)
	EPBAn 1)	USB Endpoint n Base Address Register	$C5_H$	00_H
	EPLENn 1)	USB Endpoint n Buffer Length Register	$C6_H$	$0XXXXXXX_B$
	USBPWD4)	USB Power Down Register	$E6_H$	00_H
	USBDCR 4)	USB Control Register	$E7_H$	00_H
	USBDR0 4)	USB Data Register 0	$E8_H$	00_H
	USBDR1 4)	USB Data Register 1	$E9_H$	00_H
	USBDR2 4)	USB Data Register 2	EA_H	00_H
	USBDR3 4)	USB Data Register 3	EB_H	00_H
	USBDR4 4)	USB Data Register 4	EC_H	00_H
	USBDR5 4)	USB Data Register 5	ED_H	00_H
	USBDR6 4)	USB Data Register 6	EE_H	00_H
	USBDR7 4)	USB Data Register 7	EF_H	00_H

1) These registers are multiple registers (n=0-4) with the same SFR address; selection of register "n" is done by SFR EPSEL.
2) The reset value of ADROFF is valid only if USBVAL has not been read or written since the last hardware reset.
3) The reset value of EPIR0 is 11_H.
4) These registers are only used in USB low-speed operation.

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

Addr	Register	Reset Value[1)]	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
80_H [2)]	P0	FF_H	.7	.6	.5	.4	.3	.2	.1	.0
81_H	SP	07_H	.7	.6	.5	.4	.3	.2	.1	.0
82_H	DPL	00_H	.7	.6	.5	.4	.3	.2	.1	.0
83_H	DPH	00_H	.7	.6	.5	.4	.3	.2	.1	.0
86_H	WDTREL	00_H	WDT PSEL	.6	.5	.4	.3	.2	.1	.0
87_H	PCON	$X00X-0000_B$	–	PDS	IDLS	–	GF1	GF0	PDE	IDLE
88_H [2)]	TCON	00_H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
88_H [2) 3)]	PCON1	$0XX0-XXXX_B$	EWPD	–	–	WS	–	–	–	–
89_H	TMOD	00_H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
$8A_H$	TL0	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$8B_H$	TL1	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$8C_H$	TH0	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$8D_H$	TH1	00_H	.7	.6	.5	.4	.3	.2	.1	.0
90_H [2)]	P1	FF_H	.7	.6	$\overline{SLS}$	STO	SRI	SCLK	LED1	LED0
93_H	SSCCON	07_H	SCEN	TEN	MSTR	CPOL	CPHA	BRS2	BRS1	BRS0
94_H	STB	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
95_H	SRB	XX_H	.7	.6	.5	.4	.3	.2	.1	.0
96_H	SSCMOD	00_H	LOOPB	TRIO	0	0	0	0	0	LSBSM
$9A_H$	ITCON	$XXXX-1010_B$	–	–	–	–	I1ETF	I1ETR	I0ETF	I0ETR
$A0_H$ [2)]	P2	FF_H	.7	.6	.5	.4	.3	.2	.1	.0
$A8_H$ [2)]	IEN0	$0XXX-0000_B$	EA	–	–	–	ET1	EX1	ET0	EX0
$A9_H$	IEN1	$XXXX-X000_B$	–	–	–	–	–	EUDI	EUEI	ESSC
AB_H	SCF	$XXXX-XX00_B$	–	–	–	–	–	–	WCOL	TC

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

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

Addr	Register	Reset Value[1)]	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AC_H	SCIEN	XXXX-XX00$_B$	–	–	–	–	–	–	WCEN	TCEN
$B0_H$[2)]	P3	FF_H	RD	WR	T1	T0	INT1	INT0	DADD	LED2
$B1_H$	SYSCON	XX10-XXXX$_B$	–	–	EALE	RMAP	–	–	–	–
$B8_H$[2)]	IP0	XXXX-0000$_B$	–	–	–	–	PT1	PX1	PT0	PX0
$B9_H$	IP1	XXXX-X000$_B$	–	–	–	–	–	PUDI	PUEI	PSSC
$C0_H$ [2)]	WDCON	XXXX-0000$_B$	–	–	–	–	OWDS	WDTS	WDT	SWDT
$C1_H$ to $C7_H$		USB Device and Endpoint Register definition see **table 3-3**								
$D0_H$ [2)]	PSW	00_H	CY	AC	F0	RS1	RS0	OV	F1	P
$D2_H$	EPSEL	80_H	EPS7	0	0	0	0	EPS2	EPS1	EPS0
$D3_H$	USBVAL	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$D4_H$	ADROFF	00_H [6)]	0	0	AO5	AO4	AO3	AO2	AO1	AO0
$D6_H$	GEPIR	00_H	DRVI	0	0	EPI4	EPI3	EPI2	EPI1	EPI0
$E0_H$[2)]	ACC	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E6_H$ [7)]	USBPWD	00_H	0	0	SUSPIE	DADDIE	SUSP	DADD	TPWD	RPWD
$E7_H$ [7)]	USBDCR	00_H	TYPE3	TYPE2	TYPE1	TYPE0	LEN3	LEN2	LEN1	LEN0
$E8_H$ [7)]	USBDR0	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$E9_H$ [7)]	USBDR1	00_H	.7	.6	.5	.4	.3	.2	.1	.0
EA_H [7)]	USBDR2	00_H	.7	.6	.5	.4	.3	.2	.1	.0
EB_H [7)]	USBDR3	00_H	.7	.6	.5	.4	.3	.2	.1	.0
EC_H[7)]	USBDR4	00_H	.7	.6	.5	.4	.3	.2	.1	.0
ED_H[7)]	USBDR5	00_H	.7	.6	.5	.4	.3	.2	.1	.0
EE_H [7)]	USBDR6	00_H	.7	.6	.5	.4	.3	.2	.1	.0

1) X means that the value is undefined and the location is reserved
2) Bit-addressable special function registers
3) SFR is located in the mapped SFR area. For accessing this SFR, bit RMAP in SFR SYSCON must be set.
4) These are read-only registers
5) The content of this SFR varies with the actual step of the C541U (e.g. 01_H for the first step)
6) The reset value of ADROFF is valid only if USBVAL has not been read or written since the last hardware reset
7) These registers are only used in USB low-speed operation.

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

Addr	Register	Reset Value[1)]	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EF_H 7)	USBDR7	00_H	.7	.6	.5	.4	.3	.2	.1	.0
$F0_H$ 2)	B	00_H	.7	.6	.5	.4	.3	.2	.1	.0
FC_H 3) 4)	VR0	$C5_H$	1	1	0	0	0	1	0	1
FD_H 3) 4)	VR1	$C1_H$	1	1	0	0	0	0	0	1
FE_H 3) 4)	VR2	5)	.7	.6	.5	.4	.3	.2	.1	.0

1) X means that the value is undefined and the location is reserved
2) Bit-addressable special function registers
3) SFR is located in the mapped SFR area. For accessing this SFR, bit RMAP in SFR SYSCON must be set.
4) These are read-only registers
5) The content of this SFR varies with the actual step of the C541U (e.g. 01_H for the first step)
6) The reset value of ADROFF is valid only if USBVAL has not been read or written since the last hardware reset.
7) These registers are only used in USB low-speed operation.

Table 3-3
Contents of the USB Device and Endpoint Registers (Addr. $C1_H$ to $C7_H$)

Addr	Register	Reset Value	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EPSEL = $1XXX.XXXX_B$	**Device Registers**									
$C1_H$	DCR	$000X.0000_B$	SPEED	DA	SWR	SUSP	DINIT	RSM	UCLK	PCLK
$C2_H$	DPWDR	00_H	DRVIE	XVREG	0	0	0	0	TPWD	RPWD
$C3_H$	DIER	00_H	SE0IE	DAIE	DDIE	SBIE	SEIE	STIE	SUIE	SOFIE
$C4_H$	DIRR	00_H	SE0I	DAI	DDI	SBI	SEI	STI	SUI	SOFI
$C5_H$	reserved									
$C6_H$	FNRL	XX_H	FNR7	FNR6	FNR5	FNR4	FNR3	FNR2	FNR1	FNR0
$C7_H$	FNRH	$0000.0XXX_B$	0	0	0	0	0	FNR10	FNR9	FNR8
EPSEL = $0XXX.X000_B$	**Endpoint 0 Registers**									
$C1_H$	EPBC0	00_H	STALL0	0	0	GEPIE0	SOFDE0	INCE0	0	DBM0
$C2_H$	EPBS0	20_H	UBF0	CBF0	DIR0	ESP0	SETRD0	SETWR0	CLREP0	DONE0
$C3_H$	EPIE0	00_H	AIE0	NAIE0	RLEIE0	–	DNRIE0	NODIE0	EODIE0	SODIE0
$C4_H$	EPIR0	11_H	ACK0	NACK0	RLE0	–	DNR0	NOD0	EOD0	SOD0
$C5_H$	EPBA0	00_H	PAGE0	0	0	0	A06	A05	A04	A03
$C6_H$	EPLEN0	$0XXX.XXXX_B$	0	L06	L05	L04	L03	L02	L01	L00
$C7_H$	reserved									
EPSEL = $0XXX.X001_B$	**Endpoint 1 Registers**									
$C1_H$	EPBC1	00_H	STALL1	0	0	GEPIE1	SOFDE1	INCE1	0	DBM1
$C2_H$	EPBS1	20_H	UBF1	CBF1	DIR1	ESP1	SETRD1	SETWR1	CLREP1	DONE1
$C3_H$	EPIE1	00_H	AIE1	NAIE1	RLEIE1	–	DNRIE1	NODIE1	EODIE1	SODIE1
$C4_H$	EPIR1	10_H	ACK1	NACK1	RLE1	–	DNR1	NOD1	EOD1	SOD1
$C5_H$	EPBA1	00_H	PAGE1	0	0	0	A16	A15	A14	A13
$C6_H$	EPLEN1	$0XXX.XXXX_B$	0	L16	L15	L14	L13	L12	L11	L10
$C7_H$	reserved									

Table 3-3
Contents of the USB Device and Endpoint Registers (Addr. $C1_H$ to $C7_H$) (cont'd)

Addr	Register	Reset Value	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EPSEL = 0XXX.X010$_B$	**Endpoint 2 Registers**									
$C1_H$	EPBC2	00_H	STALL2	0	0	GEPIE2	SOFDE2	INCE2	0	DBM2
$C2_H$	EPBS2	20_H	UBF2	CBF2	DIR2	ESP2	SETRD2	SETWR2	CLREP2	DONE2
$C3_H$	EPIE2	00_H	AIE2	NAIE2	RLEIE2	–	DNRIE2	NODIE2	EODIE2	SODIE2
$C4_H$	EPIR2	10_H	ACK2	NACK2	RLE2	–	DNR2	NOD2	EOD2	SOD2
$C5_H$	EPBA2	00_H	PAGE2	0	0	0	A62	A52	A42	A32
$C6_H$	EPLEN2	0XXX. $XXXX_B$	0	L62	L52	L42	L32	L22	L12	L02
$C7_H$	reserved									
EPSEL = 0XXX.X011$_B$	**Endpoint 3 Registers**									
$C1_H$	EPBC3	00_H	STALL3	0	0	GEPIE3	SOFDE3	INCE3	0	DBM3
$C2_H$	EPBS3	20_H	UBF3	CBF3	DIR3	ESP3	SETRD3	SETWR3	CLREP3	DONE3
$C3_H$	EPIE3	00_H	AIE3	NAIE3	RLEIE3	–	DNRIE3	NODIE3	EODIE3	SODIE3
$C4_H$	EPIR3	10_H	ACK3	NACK3	RLE3	–	DNR3	NOD3	EOD3	SOD3
$C5_H$	EPBA3	00_H	PAGE3	0	0	0	A63	A52	A43	A33
$C6_H$	EPLEN3	0XXX. $XXXX_B$	0	L63	L53	L43	L33	L23	L13	L03
$C7_H$	reserved									
EPSEL = 0XXX.X100$_B$	**Endpoint 4 Registers**									
$C1_H$	EPBC4	00_H	STALL4	0	0	GEPIE4	SOFDE4	INCE4	0	DBM4
$C2_H$	EPBS4	20_H	UBF4	CBF4	DIR4	ESP4	SETRD4	SETWR4	CLREP4	DONE4
$C3_H$	EPIE4	00_H	AIE4	NAIE4	RLEIE4	–	DNRIE4	NODIE4	EODIE4	SODIE4
$C4_H$	EPIR4	10_H	ACK4	NACK4	RLE4	–	DNR4	NOD4	EOD4	SOD4
$C5_H$	EPBA4	00_H	PAGE4	0	0	0	A64	A54	A44	A34
$C6_H$	EPLEN4	0XXX. $XXXX_B$	0	L64	L54	L44	L34	L24	L14	L04
$C7_H$	reserved									

4 External Bus Interface

The C541U allows for external memory expansion. The functionality and implementation of the external bus interface is identical to the common interface for the 8051 architecture.

4.1 Accessing External Memory

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

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

4.1.1 Role of P0 and P2 as Data/Address Bus

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

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

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

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

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

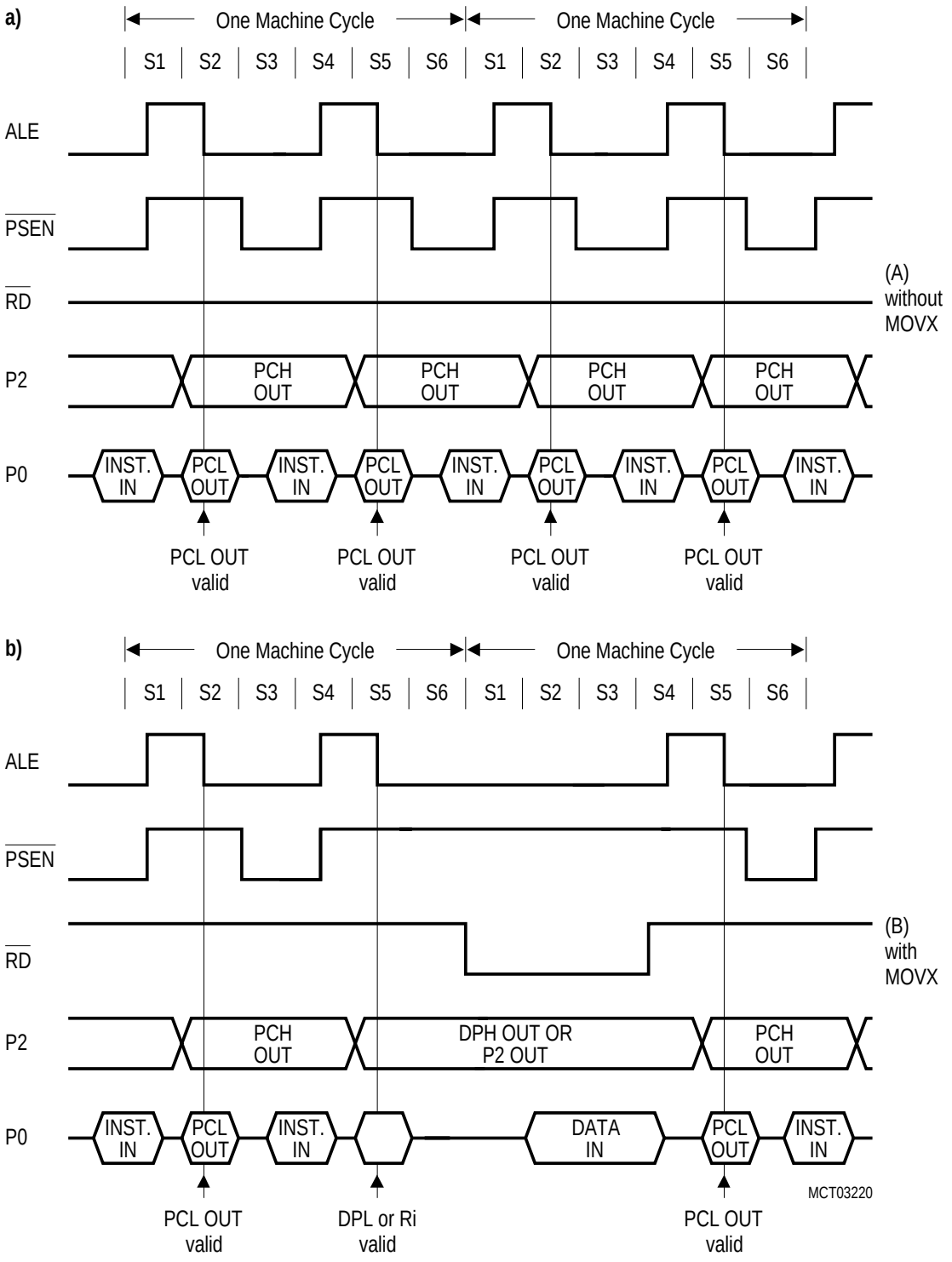


Figure 4-1
External Program Memory Execution

4.1.2 Timing

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

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

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

4.1.3 External Program Memory Access

The external program memory is accessed under two conditions:

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

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

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

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

4.3 Overlapping External Data and Program Memory Spaces

In some applications it is desirable to execute a program from the same physical memory that is used for storing data. In the C541U the external program and data memory spaces can be combined by the logical-AND of $\overline{PSEN}$ and $\overline{RD}$. A positive result from this AND operation produces a low active read strobe that can be used for the combined physical memory. Since the $\overline{PSEN}$ cycle is faster than the $\overline{RD}$ cycle, the external memory needs to be fast enough to adapt to the $\overline{PSEN}$ cycle.

4.4 ALE, Address Latch Enable

The 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 $\overline{RD}$/$\overline{WR}$ signals are active. The first ALE of the second cycle of a MOVX instruction is missing (see **figure 4-1 b**). Consequently, in any system that does not use data memory, ALE is activated at a constant rate of 1/6 of the oscillator frequency and can be used for external clocking or timing purposes.

The C541U allows to switch off the ALE output signal. If the internal OTP program memory is used ($\overline{EA}$=1) and ALE is switched off by EALE=0, ALE will only go active during external data memory accesses (MOVX instructions) and code memory accesses with an address greater than $1FFF_H$ for the C541U (external code memory fetches). If $\overline{EA}$=0, the ALE generation is always enabled and the bit EALE has no effect.

After a hardware reset the ALE generation is enabled.

Special Function Register SYSCON (Address $B1_H$) **Reset Value : $XX10XXXX_B$**

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

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

Bit	Function
–	Not implemented. Reserved for future use.
EALE	**Enable ALE output** EALE = 0 : ALE generation is disabled; disables ALE signal generation during internal code memory accesses ($\overline{EA}$=1). With $\overline{EA}$=1, ALE is automatically generated at MOVX instructions and code memory accesses with an address greater $1FFF_H$. EALE = 1 : ALE generation is enabled If $\overline{EA}$=0, the ALE generation is always enabled and the bit EALE has no effect on the ALE generation.

4.5 Enhanced Hooks Emulation Concept

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

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

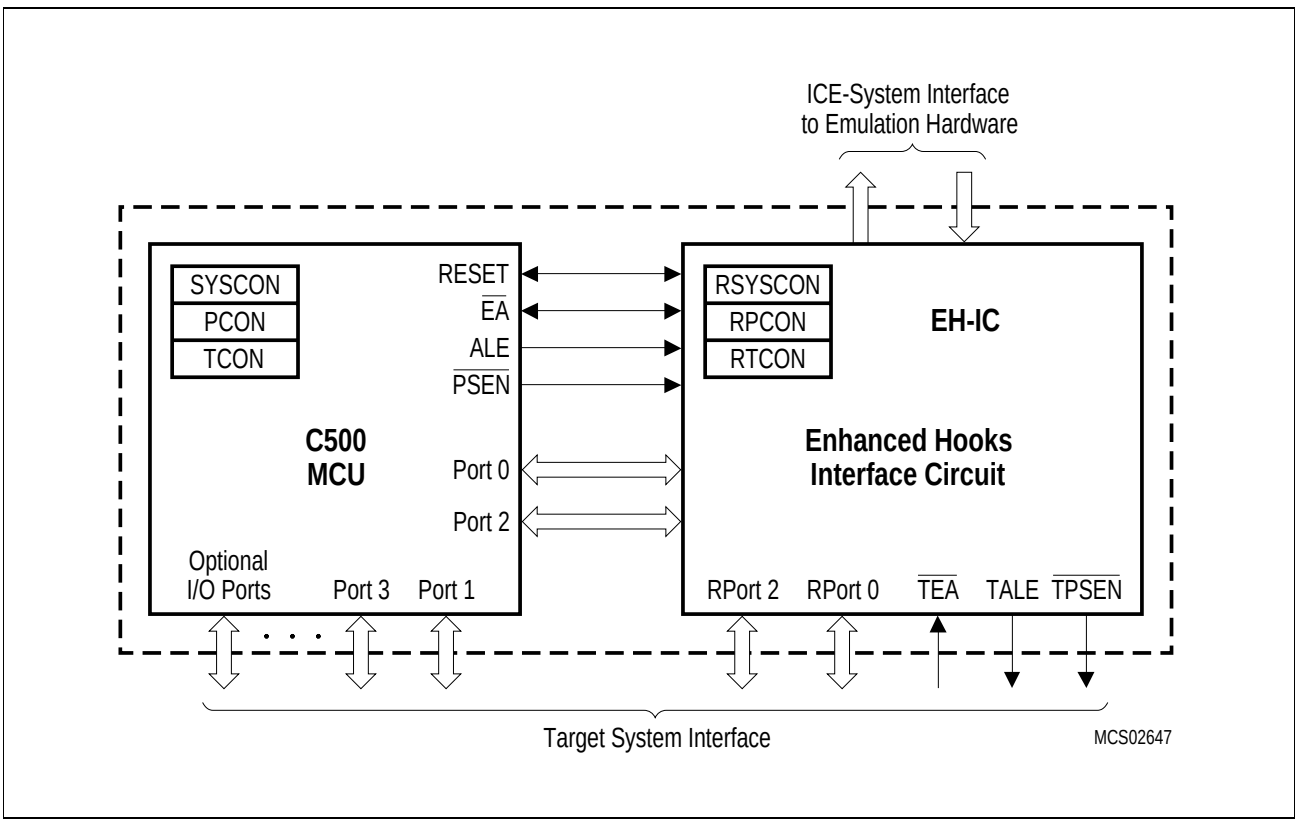


Figure 4-2
Basic C500 MCU Enhanced Hooks Concept Configuration

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

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

5 Reset and System Clock Operation

5.1 Hardware Reset Operation

The hardware reset function incorporated in the C541U allows for an easy automatic start-up at a minimum of additional hardware and forces the controller to a predefined default state. The hardware reset function can also be used during normal operation in order to restart the device. This is particularly done when the power down mode is to be terminated.

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

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

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

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

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

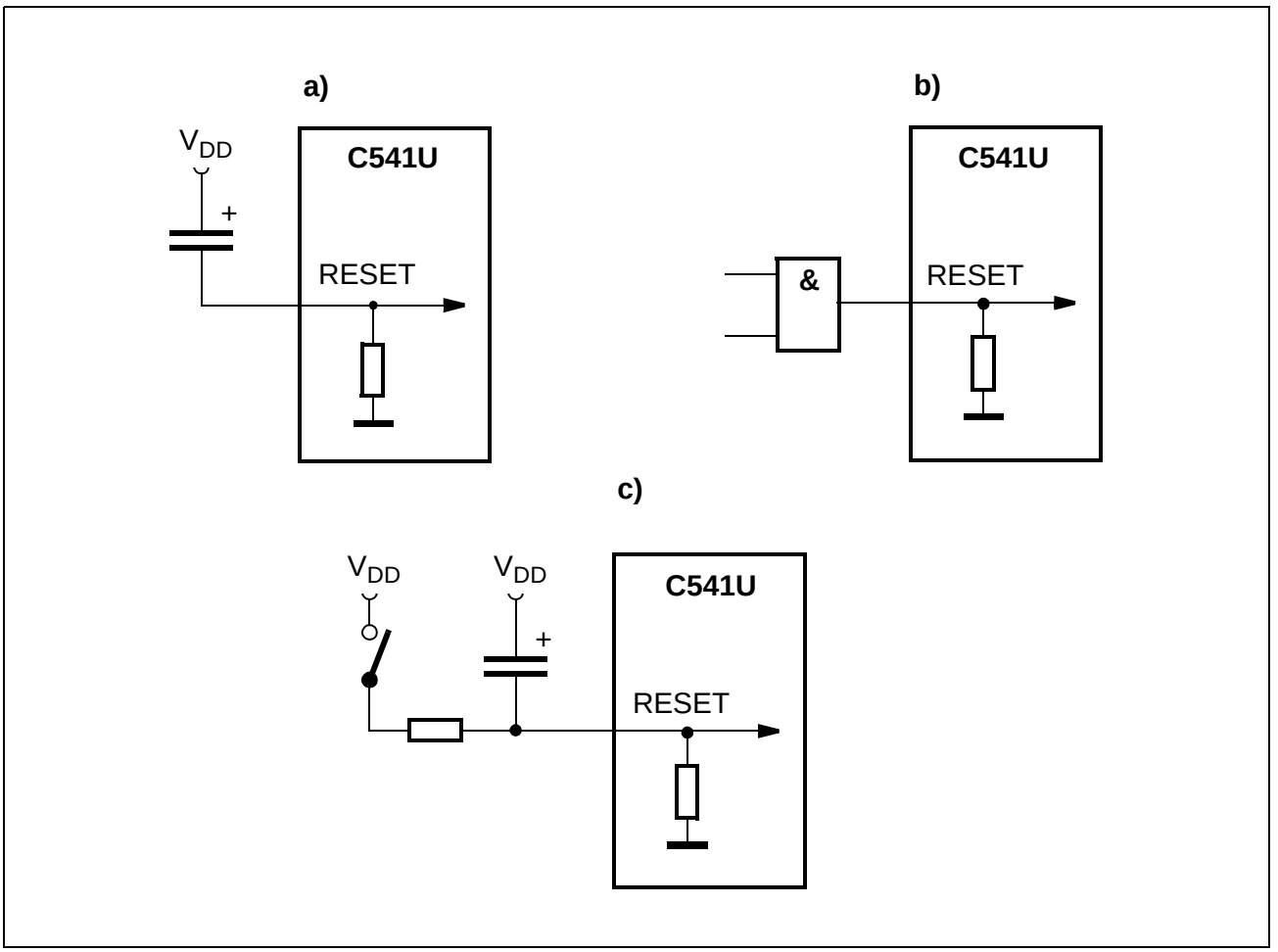


Figure 5-1
Reset Circuitries

A correct reset leaves the processor in a defined state. The program execution starts at location 0000_H. After reset is internally accomplished the port latches of ports 0 to 3 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 and 3) output a one (1). Port 2 lines output a zero (or one) after reset, if $\overline{EA}$ is held low (or high).

The content of the internal RAM of the C541U is not affected by a reset. After power-up the content is undefined, while it remains unchanged during a reset if the power supply is not turned off.

A reset operation of the USB module in the C541U can only be achieved under software control. A hardware reset operation puts only the internal CPU interface of the USB module and its MMU into a well defined reset state.

The software reset, which must be executed after a hardware reset, is initiated by setting bit SWR in SFR DCR by software. Bit SWR is reset automatically by hardware when the software reset operation of the USB module is finished. Further, with the reset of bit SWR, bit DINIT in DCR is set indicating the CPU that it has to initialize the endpoints of USB module.

5.2 Fast Internal Reset after Power-On

The C541U uses the oscillator watchdog unit for a fast internal reset procedure after power-on. **Figure 5-1** shows the power-on sequence under control of the oscillator watchdog.

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

In the C541U the oscillator watchdog unit avoids this situation. In this case, after power-on the oscillator watchdog's RC oscillator starts working within a very short start-up time (typ. less than 2 microseconds). In the following the watchdog circuitry detects a failure condition for the on-chip oscillator because this has not yet started (a failure is always recognized if the watchdog's RC oscillator runs faster than the on-chip oscillator). As long as this condition is detected the watchdog uses the RC oscillator output as clock source for the chip rather than the on-chip oscillator's output. This allows correct resetting of the part and brings also all ports to the defined state (see **figure 5-2**).

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

- Typ.: 18 µs
- Max.: 34 µs

The RC oscillator will already run at a V_{DD} below 4.0V (lower specification limit). Therefore, at slower V_{DD} rise times the delay time will be less than the two values given above.

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

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

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

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

Generally, for reset time generation at power-on an external capacitor can be applied to the RESET pin.

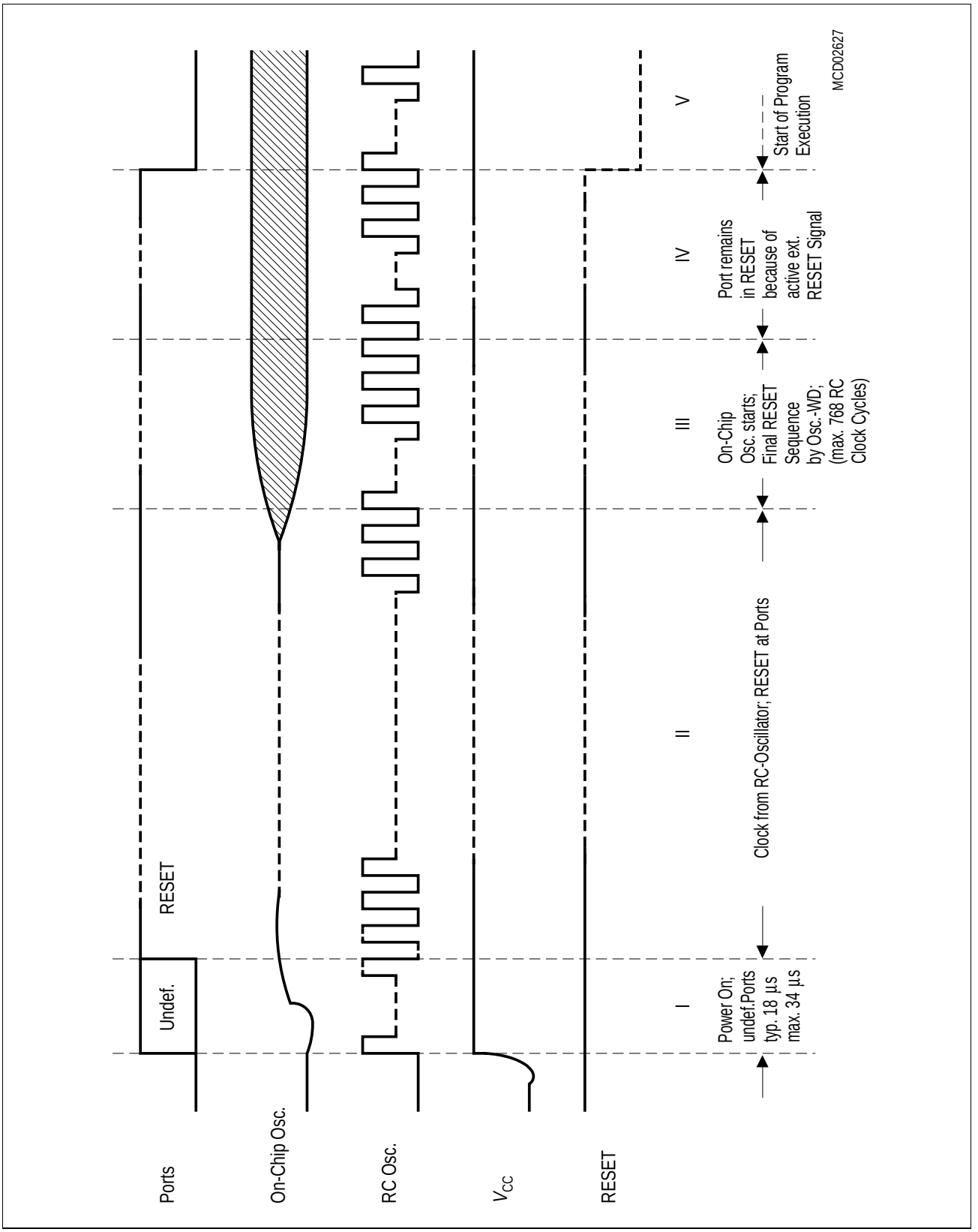


Figure 5-2
Power-On Reset of the C541U

5.3 Hardware Reset Timing

This section describes the timing of the hardware RESET signal.

The input pin RESET is sampled once during each machine cycle. This happens in state 5 phase 2. Thus, the external reset signal is synchronized to the internal CPU timing. When RESET is found active (high level) the internal reset procedure is started. It needs two complete machine cycles to put the complete device to its correct reset state, i.e. all special function registers contain their default values, the port latches contain 1's etc. Note that this reset procedure is also performed if there is no clock available at the device. (This is done by the oscillator watchdog, which provides an auxiliary clock for performing a perfect reset without clock at the XTAL1 and XTAL2 pins). The RESET signal must be active for at least two machine cycles; after this time the C541U 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 program memory) in the following state 5 phase 1. One phase later (state 5 phase 2) the first falling edge at pin ALE occurs.

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

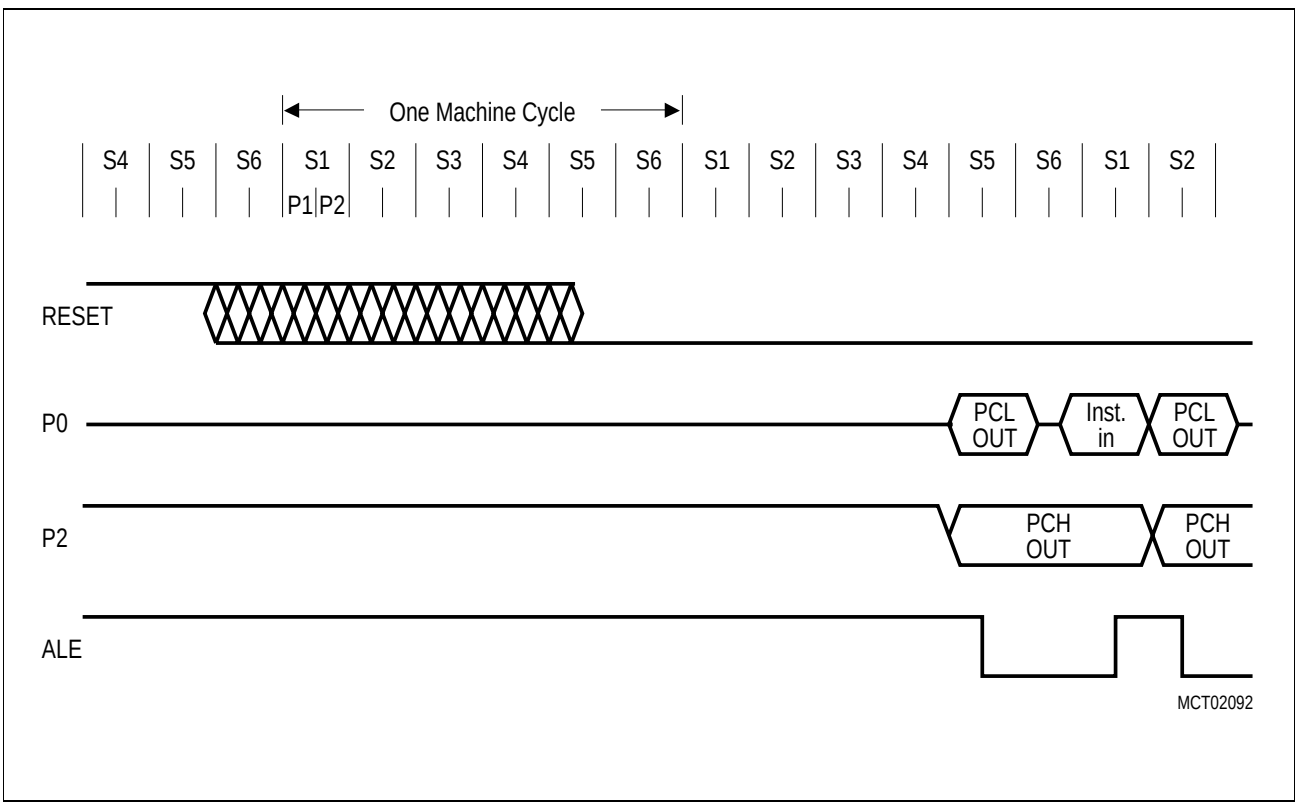


Figure 5-3
CPU Timing after Reset

5.4 Oscillator and Clock Circuit

The oscillator and clock generation circuitry of the C541U is shown in **figure 5-4**. The crystal oscillator generates the system clock for the microcontroller. The USB module can be provided with the following clocks :

– Full speed operation : 48 MHz with a data rate of 12 Mbit/s
– Low speed operation : 6 MHz with a data rate of 1.5 Mbit/s

The low speed clock is generated by dividing the system clock by 2. The full speed clock is generated by a PLL, which multiplies the system clock by a fix factor of 4. This PLL can be enabled or disabled by bit PCLK of SFR DCR. Depending on full or low speed operation of the USB bit SPEED of SFR has to be set or cleared for the selection of the USB clock. Bit UCLK is a general enable bit for the USB clock.

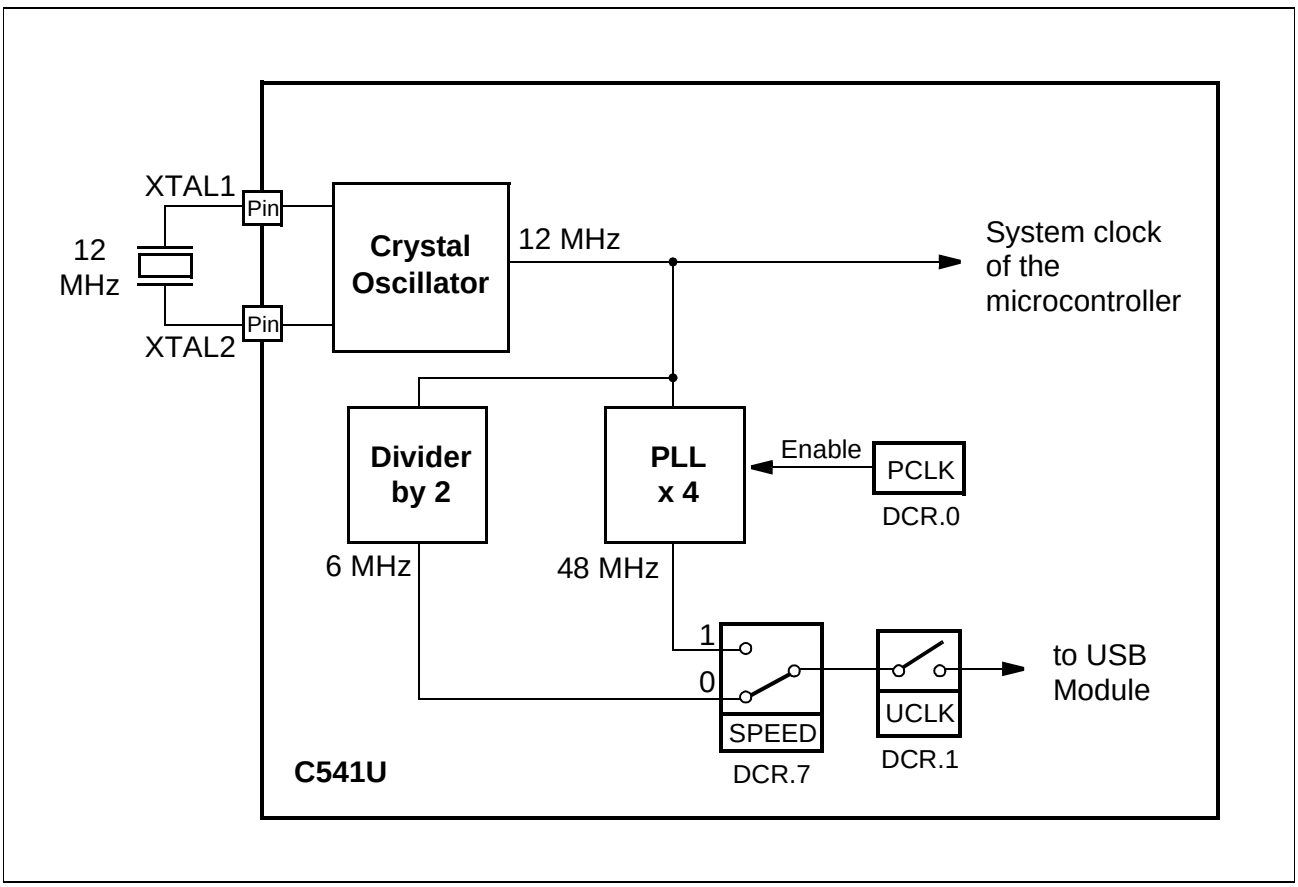


Figure 5-4
Block Diagram of the Clock Generation Circuitry

In low speed mode the PLL is not required. Therefore, the PLL should be always disabled in low speed mode. This also reduces the power consumption and the EMC of the C541U when used in low speed mode.

Note: For correct function of the USB module the C541U must operate with 12 MHz external clock. The microcontroller (except the USB module) is capable to operate down to 2 MHz

After a hardware reset operation bits PCLK, SPEED, and UCLK are set to 0. Depending on the required operating mode of the USB module a well defined procedure must be executed for switching on the clock for the USB module :

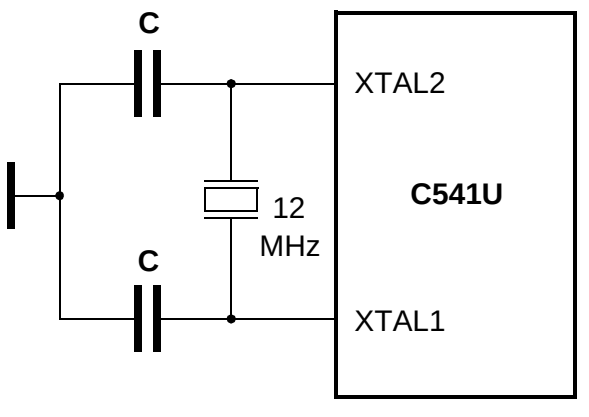
– Full speed mode USB PLL is switched on by setting bit PCLK
waiting 3 ms for PLL being locked
setting bit UCLK

– Low speed mode setting bit UCLK only

The switch-on procedure after hardware reset assures a proper operation of the USB clock system. A software reset operation of the USB module must follow this clock system switch-on procedure. Details of the software reset operation are described in chapter 6.4.5 on page 6-51.

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

Figure 5-5 shows the recommended oscillator circuit.



C = 20pF ± 10pF for crystal operation

Figure 5-5
Recommended Crystal 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 5-6**). It operates in 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.

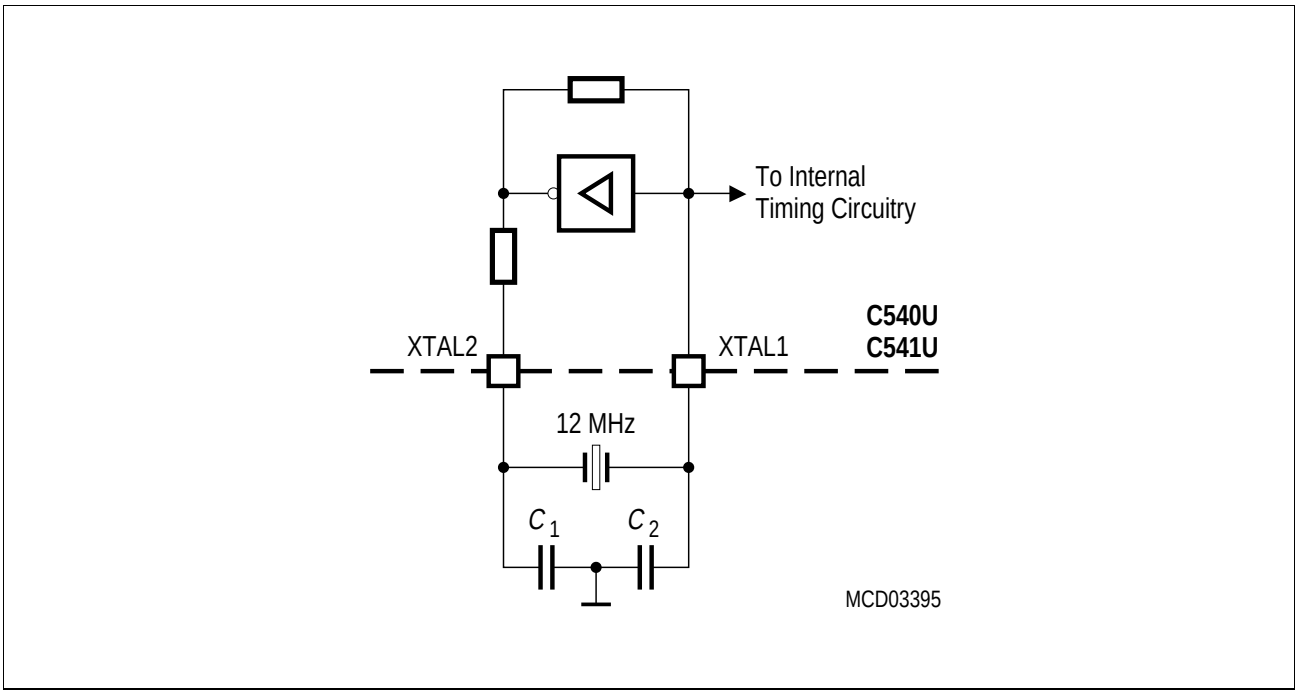


Figure 5-6
On-Chip Oscillator Circuiry

To drive the C541U with an external clock source, the external clock signal has to be applied to XTAL1, as shown in **figure 5-7**. XTAL2 has to be left unconnected. A pullup resistor is suggested (to increase the noise margin), but is optional if V_{OH} of the driving gate corresponds to the V_{IH2} specification of XTAL1.

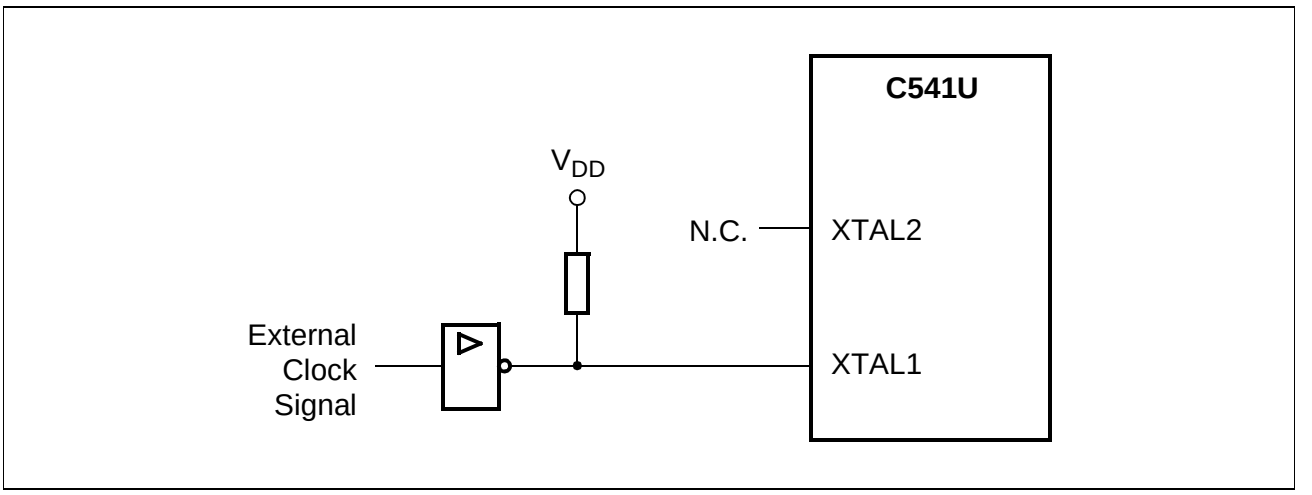


Figure 5-7
External Clock Source

6 On-Chip Peripheral Components

This chapter gives detailed information about all on-chip peripherals of the C541U except for the integrated interrupt controller, which is described separately in chapter 7.

6.1 Parallel I/O

The C541U three 8-bit I/O ports and one 6-bit I/O port (Port 1). Port 0 is an open-drain bidirectional I/O port, while ports 1 to 3 are quasi-bidirectional I/O ports with internal pullup resistors. That means, when configured as inputs, ports 1 to 3 will be pulled high and will source current when externally pulled low. Port 0 will float when configured as input.

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

Two port lines of port 1 (P1.0/LED0, P1.1/LED1) and one port line of port 3 (P3.0/LED2) have the capability of driving external LEDs in the output low state.

6.1.1 Port Structures

The C541U allows for digital I/O on 30 lines grouped into 4 bidirectional 8-/6-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 P3 are performed via their corresponding special function registers P0 to P3.

Figure 6-1 shows a functional diagram of a typical bit latch and I/O buffer, which is the core of each of the 4 I/O-ports. The bit latch (one bit in the port's SFR) is represented as a type-D flip-flop, which will clock in a value from the internal bus in response to a "write-to-latch" signal from the CPU. The Q output of the flip-flop is placed on the internal bus in response to a "read-latch" signal from the CPU. The level of the port pin itself is placed on the internal bus in response to a "read-pin" signal from the CPU. Some instructions that read from a port (i.e. from the corresponding port SFR P0 to P3) activate the "read-latch" signal, while others activate the "read-pin" signal.

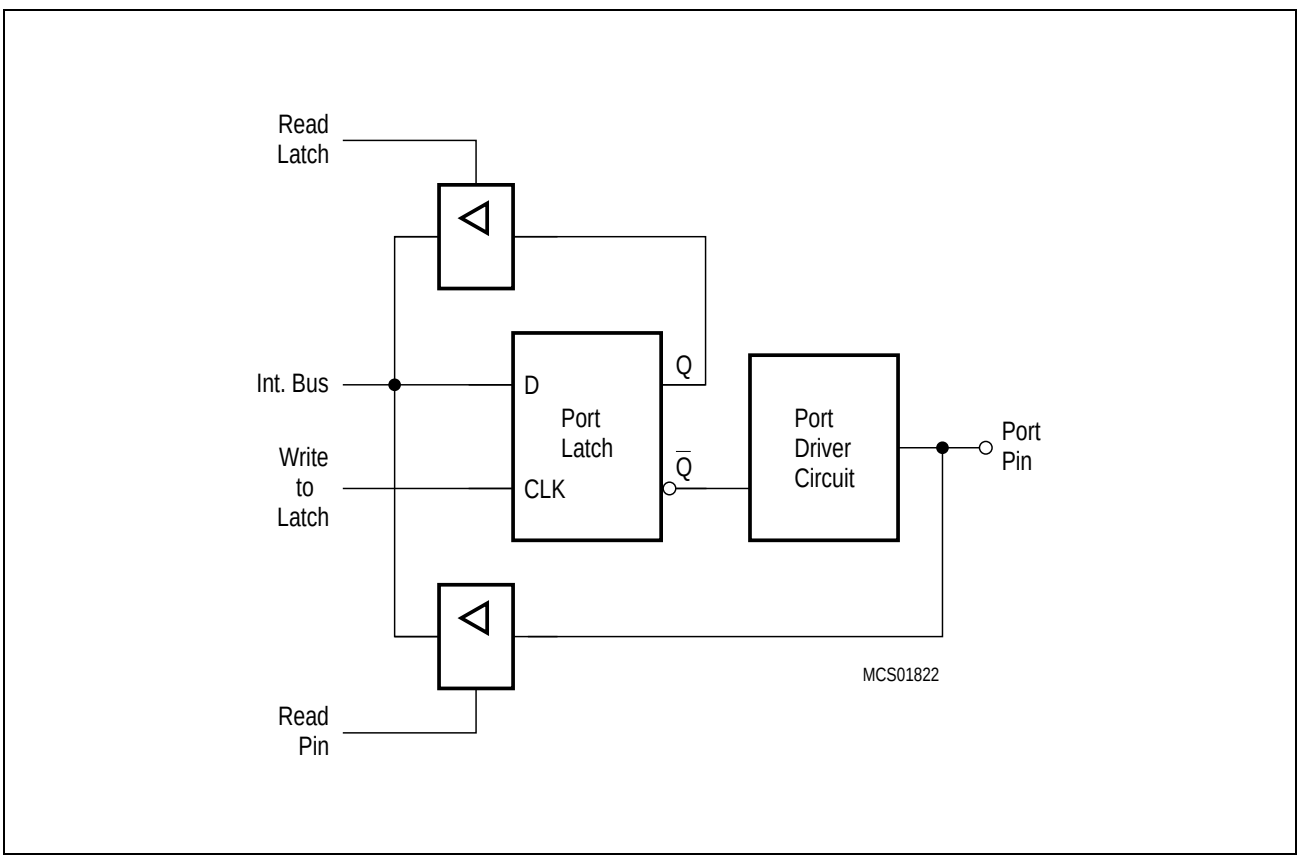


Figure 6-1
Basic Structure of a Port Circuitry

6.1.1.1 Basic Port Circuirty of Port 1 to 3

Port 1, 2 and 3 output drivers have internal pullup FET's (see **figure 6-2**). Each I/O line can be used independently as an input or output. To be used as an input, the port bit stored in the bit latch must contain a one (1) (that means for **figure 6-2**: Q=0), which turns off the output driver FET n1. Then, for ports 1, 2 and 3, 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".

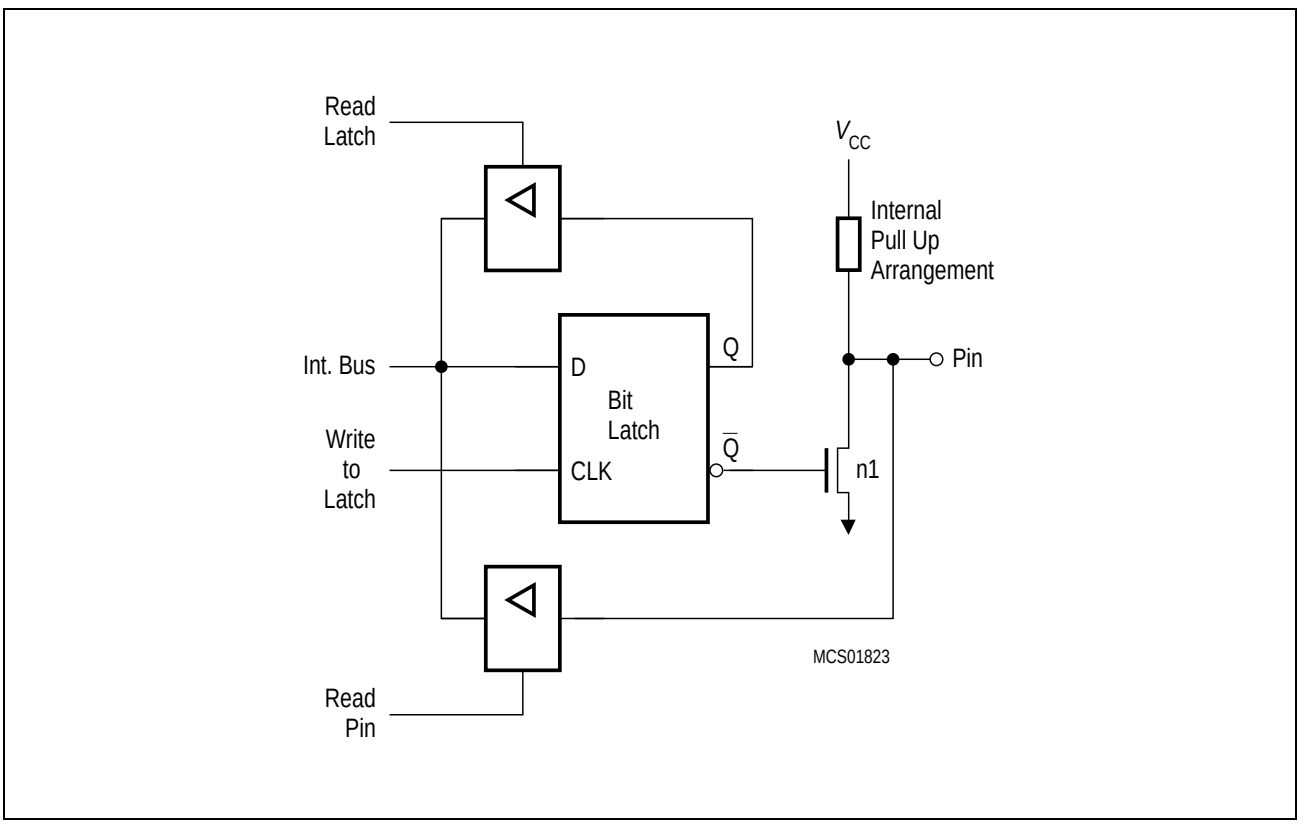


Figure 6-2
Basic Output Driver Circuit of Ports 1, 2, and 3

In fact, the pullups mentioned before and included in **figure 6-2** are pullup arrangements as shown in **figure 6-3**. One n-channel pulldown FET and three pullup FETs are used:

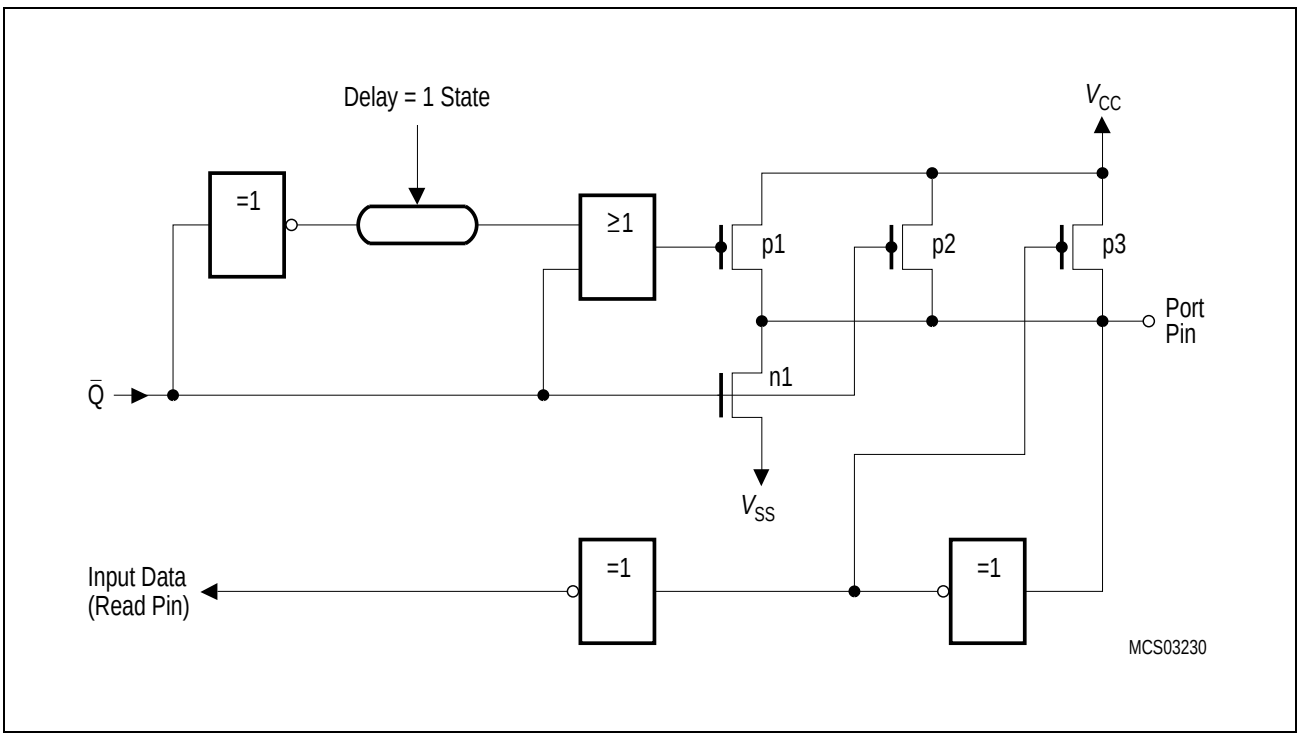


Figure 6-3
Output Driver Circuit of Ports 1 to 5 and 7

- The **pulldown FET n1** is of n-channel type. It is a very strong driver transistor which is capable of sinking high currents (I_{OL}); it is only activated if a "0" is programmed to the port pin. A short circuit to V_{DD} must be avoided if the transistor is turned on, since the high current might destroy the FET. This also means that no ”0“ must be programmed into the latch of a pin that is used as input.
- The **pullup FET p1** is of p-channel type. It is activated for 1 state (S1) 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 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 SSC Port Pins of Port 1

The port pins of the SSC interface are located as alternate functions at four lines of port 4 :

- P1.2/SCLK : when used as SSC clock output, pin becomes a true push-pull output
- P1.3/SRI : when used as SSC receiver input, pin becomes an input without pullups.
- P1.4/STO : when used as SSC transmitter output, pin becomes a true push-pull output with tristate capability
- P1.5/$\overline{\text{SLS}}$: when used as SSC slave select input, pin directly controls the tristate condition of P4.3

The modified port structure is illustrated in **figures 6-4** and **6-5**.

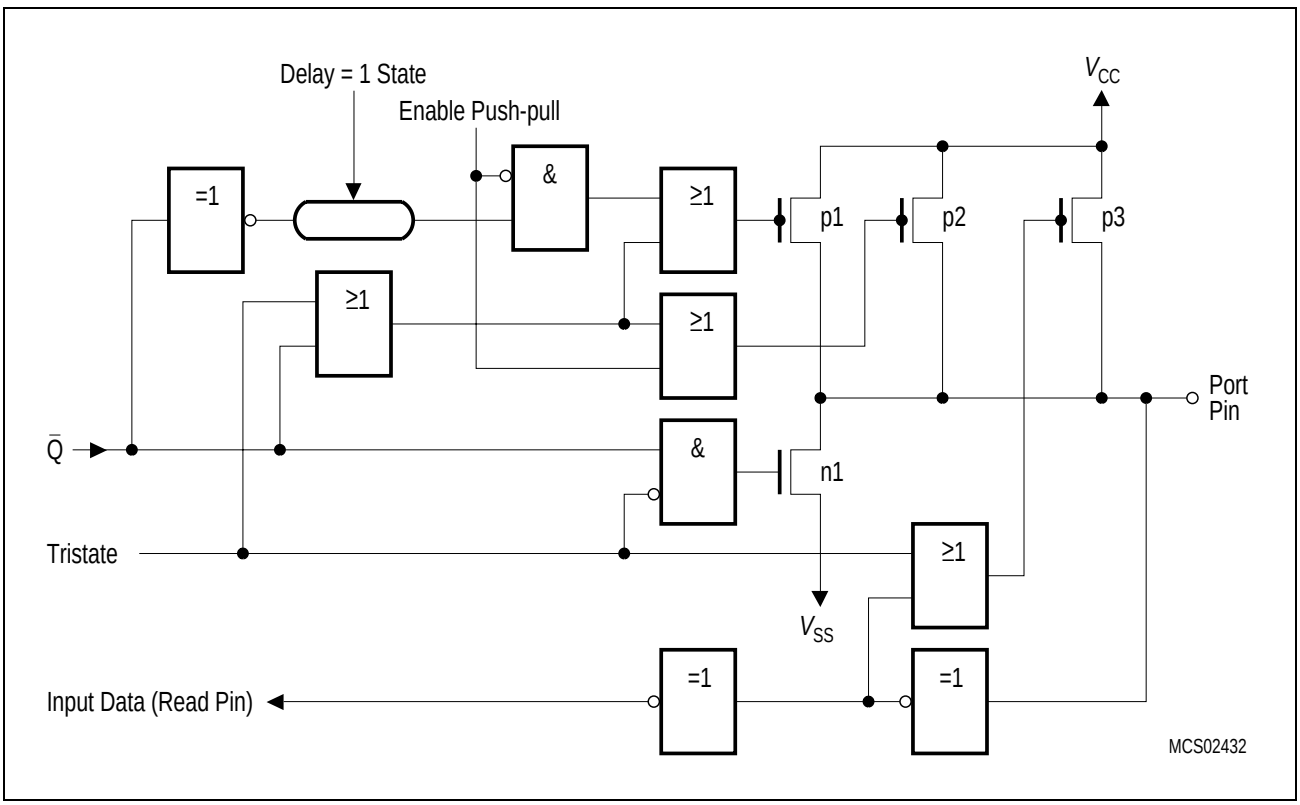


Figure 6-4
Driver Circuit of Port 1 pins P1.2 and P1.4 (when used for SLCK and STO)

Pin Control for P1.2/SCLK

When the SSC is disabled, both Enable Push-pull and Tristate will be inactive, the pin behaves like a standard I/O pin. In master mode and with SSC enabled, Enable Push-pull will be active and Tristate will be inactive. In slave mode and with SSC enabled, Enable Push-pull will be inactive and Tristate will be active.

Pin Control for P1.4/STO

When the SSC is disabled, both Enable Push-pull and Tristate will be inactive. In master mode and SSC enabled, Enable Push-pull will be active and Tristate will be inactive. In slave mode and SSC enabled, Enable Push-pull will be active. If the transmitter is enabled ($\overline{\text{SLS}}$ and TEN active), Tristate will be inactive. If the transmitter is disabled (either $\overline{\text{SLS}}$ or TEN inactive), Tristate will be active.

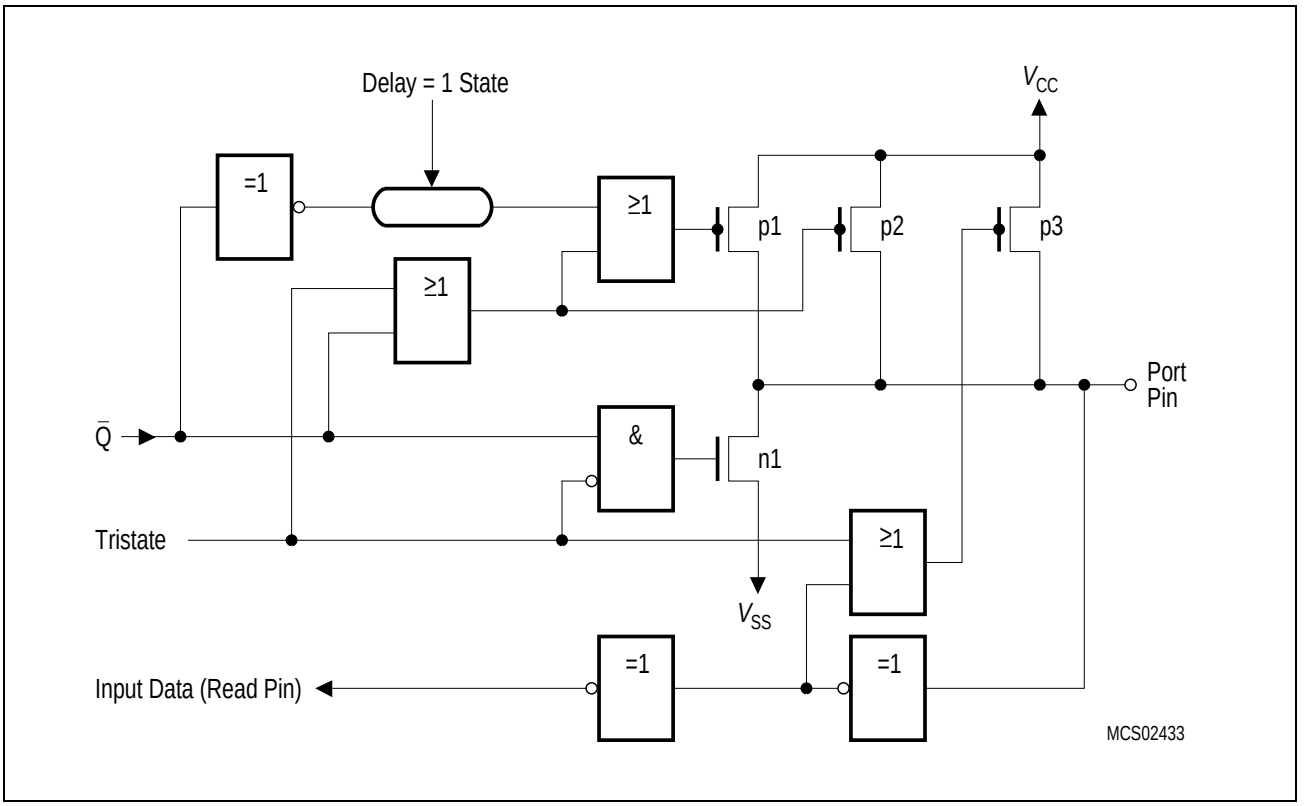


Figure 6-5
Driver Circuit of Port 1 pins P1.3 and P1.5 (when used for SRI and $\overline{SLS}$)

When enabling the SSC, inputs used for the SSC will be switched into a high-impedance mode. For P1.3/SRI, Tristate will be enabled, when the SSC is enabled. For P1.5/$\overline{SLS}$, Tristate will be enabled, when the SSC is enabled and is switched to slave mode. In master mode this pin will remain a regular I/O pin.

6.1.1.3 Port 0 Circuitry

Port 0, in contrast to ports 1, 2 and 3, 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 6-6**) 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.

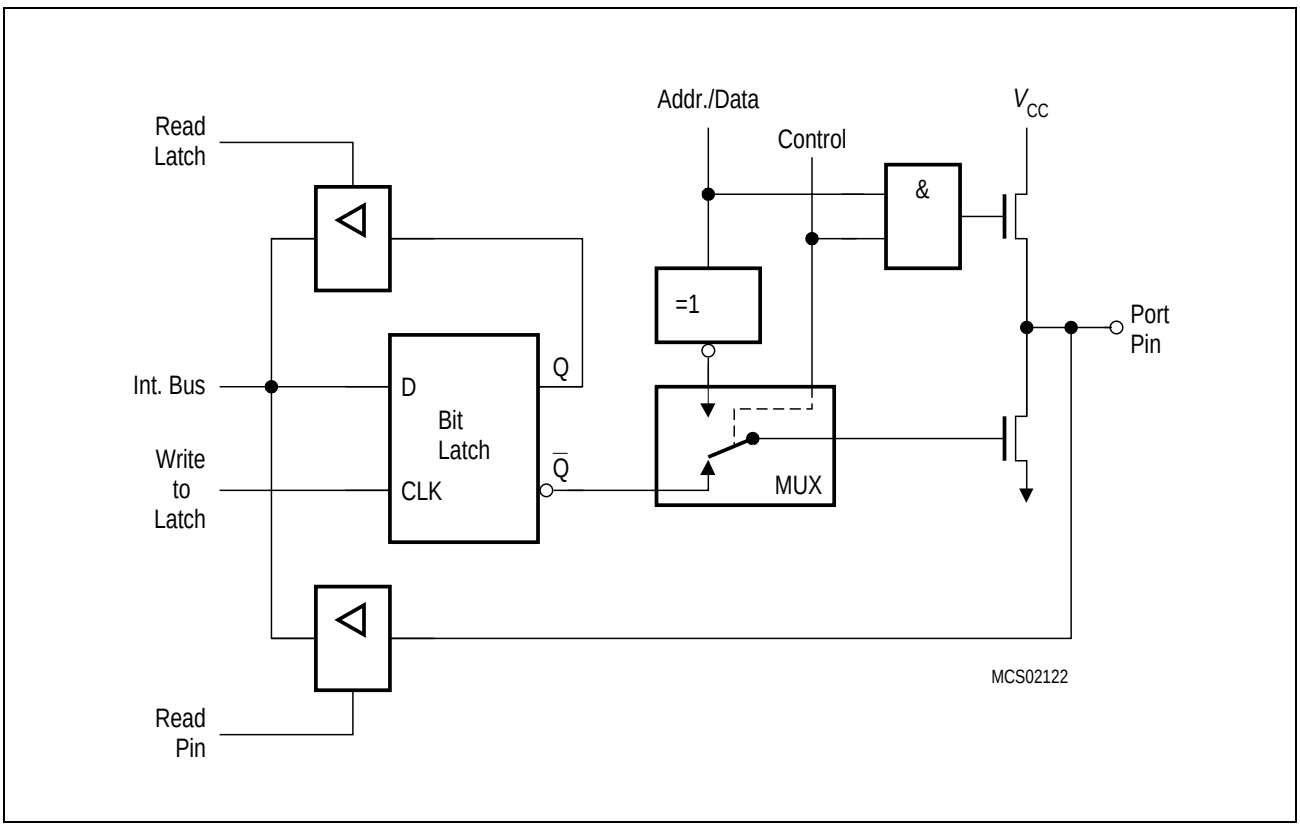


Figure 6-6
Port 0 Circuitry

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

As shown in **figure 6-6** and below in **figure 6-7**, the output drivers of ports 0 and 2 can be switched to an internal address or address/data bus for use in external memory accesses. In this application they cannot be used as general purpose I/O, even if not all address lines are used externally. The switching is done by an internal control signal dependent on the input level at the $\overline{EA}$ pin and/or the contents of the program counter. If the ports are configured as an address/data bus, the port latches are disconnected from the driver circuit. During this time, the 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 6-6**. 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.

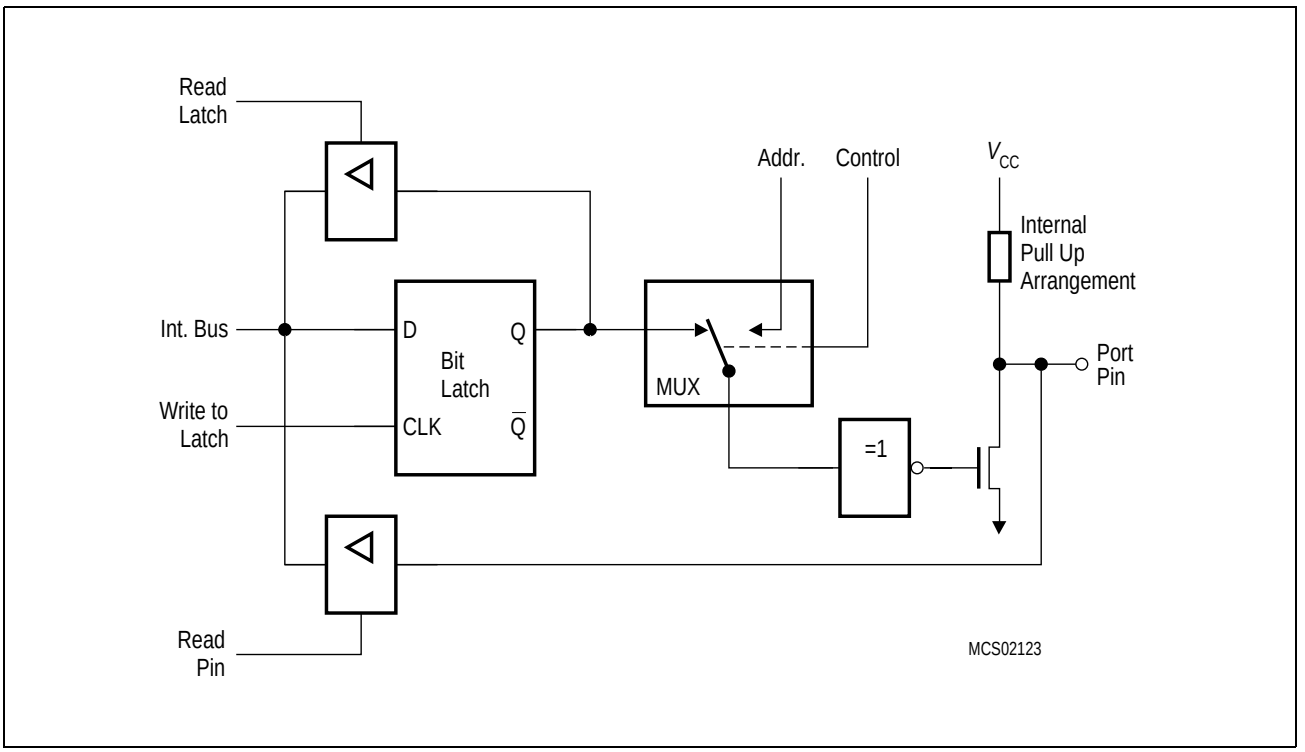


Figure 6-7
Port 2 Circuitry

6.1.2 Alternate Functions

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

Figure 6-8 shows a functional diagram of a port latch with alternate function. To pass the alternate function to the output pin and vice versa, however, the gate between the latch and driver circuit must be open. Thus, to use the alternate input or output functions, the corresponding bit latch in the port SFR has to contain a one (1); otherwise the pulldown FET is on and the port pin is stuck at 0. After reset all port latches contain ones (1).

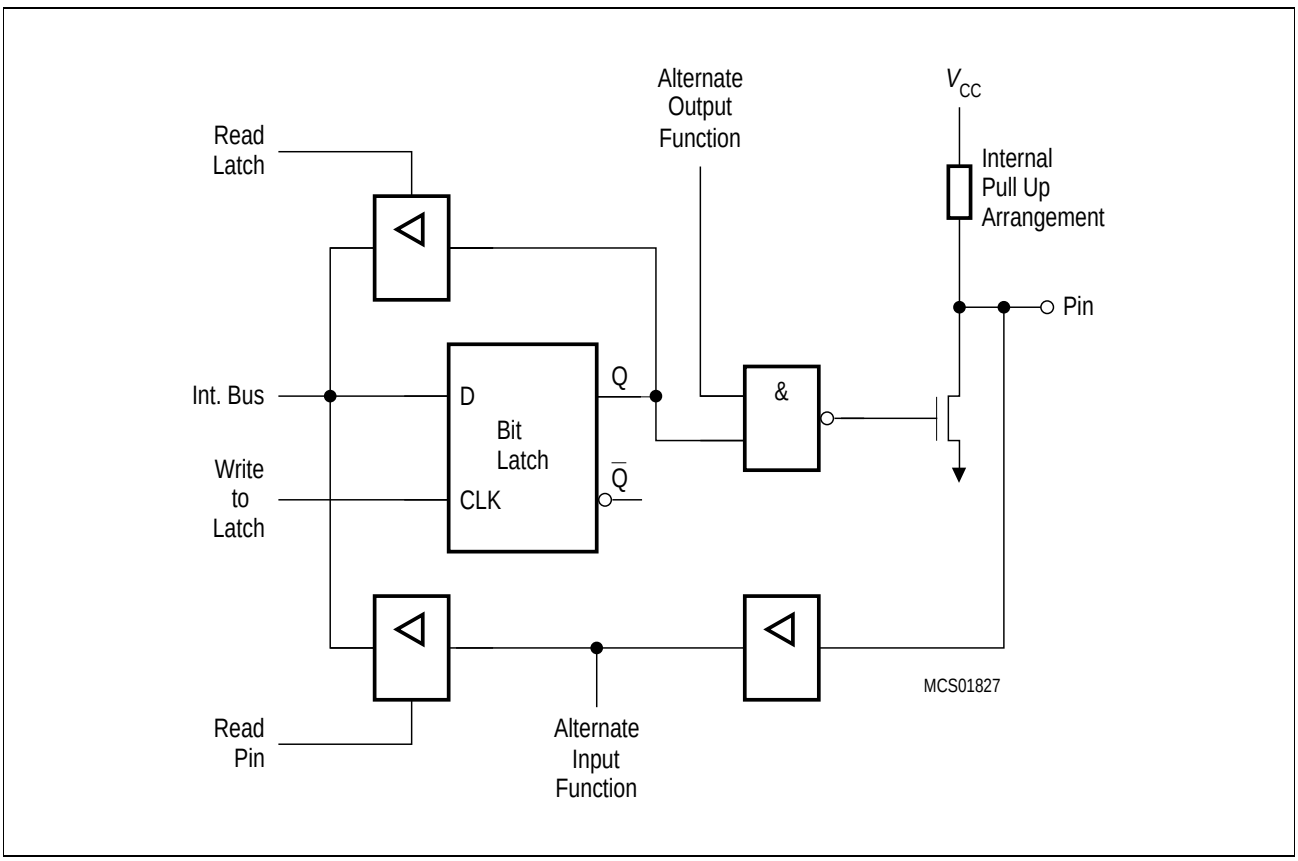


Figure 6-8
Circuitry of Ports 1 and 3

Ports 1 and 3 are provided for several alternate functions, as listed in **table 6-1**:

Table 6-1
Alternate Functions of Port 1 and 3

Port	Pin	Alternate Function
P1.0	T2	Input to counter 2
P1.1	T2EX	Capture-reload trigger of timer 2 / up down count
P1.2	SCLK	SSC master clock output, slave clock input
P1.3	SRI	SSC serial data input
P1.4	STO	SSC serial data output
P1.5	SLS	SSC slave select input
P3.1	DADD	Device attached input of the USB module
P3.2	INT0	External interrupt 0 input, timer 0 gate control
P3.3	INT1	External interrupt 1 input, timer 1 gate control
P3.4	T0	Timer 0 external counter input
P3.5	T1	Timer 1 external counter input
P3.6	WR	External data memory write strobe
P3.7	RD	External data momory read strobe

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 6-9** 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.

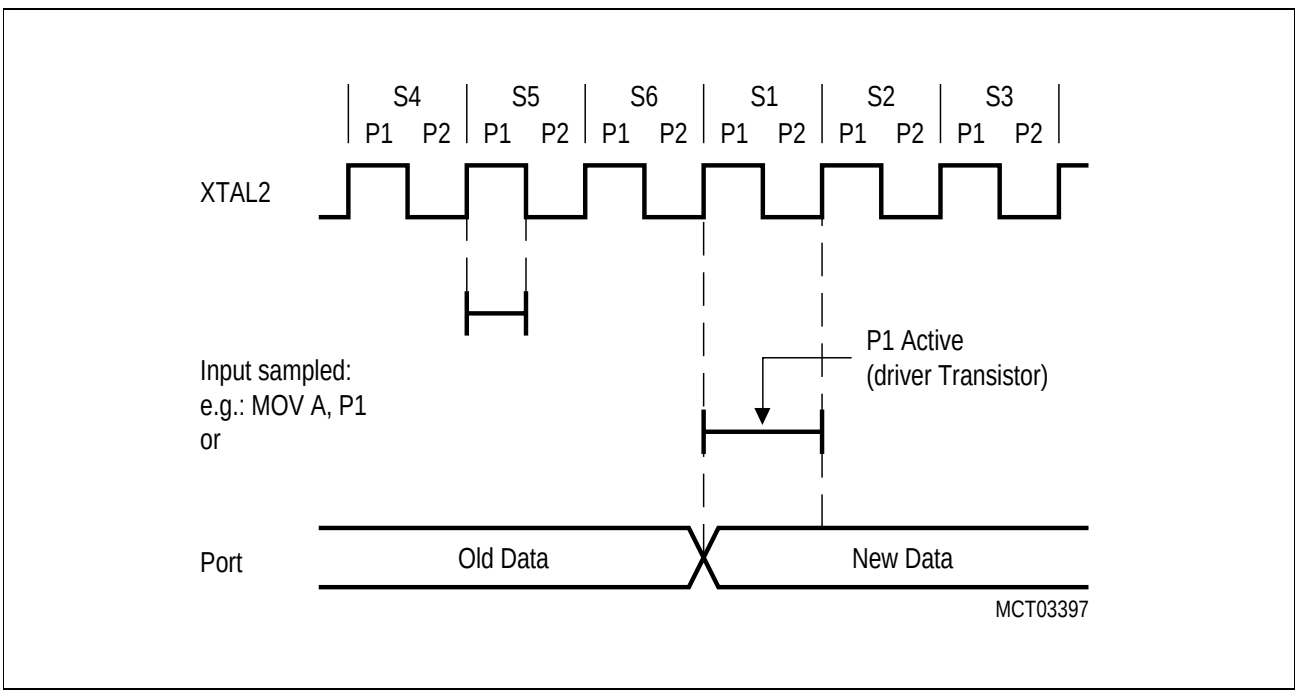


Figure 6-9
Port Timing

6.1.3.2 Port Loading and Interfacing

The output buffers of ports 1, 2 and 3 can drive TTL inputs directly. The maximum port load which still guarantees correct logic output levels can be looked up in the C541U DC characteristics in chapter 10. 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, 2 and 3 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 C541U 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 1,2 and 3

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, respectively, is performed by reading the SFR 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 Timers/Counters

The C541U contains two 16-bit timers/counters, timer 0 and 1, which are useful in many applications for timing and counting.

In "timer" function, the timer register is incremented every machine cycle. Thus one can think of it as counting machine cycles. Since a machine cycle consists of 6 oscillator periods, the counter rate is 1/6 of the oscillator frequency.

In "counter" function, the timer register is incremented in response to a 1-to-0 transition (falling edge) at its corresponding external input pin, T0 or T1 (alternate functions of P3.4 and P3.5). In this function the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since it takes two machine cycles (12 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is 1/12 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it must be held for at least one full machine cycle.

6.2.1 Timer/Counter 0 and 1

Timer / counter 0 and 1 of the C541U are fully compatible with timer / counter 0 and 1 of the 80C51/ C501 and can be used in the same four operating modes:

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

Mode 1: 16-bit timer/counter

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

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

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

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

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

6.2.1.1 Timer/Counter 0 and 1 Registers

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

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

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

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

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

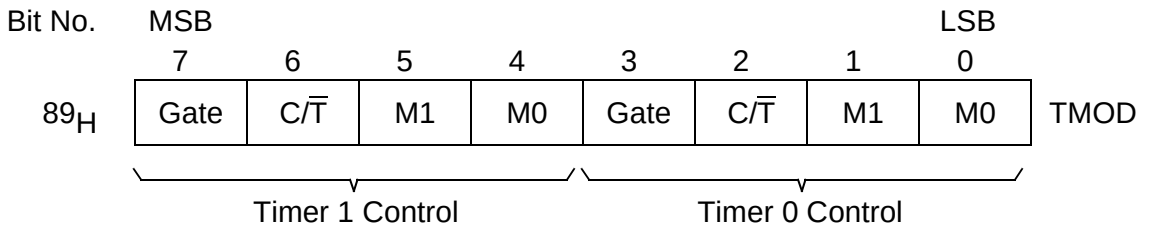
Special Function Register TCON (Address 88_H) **Reset Value : 00_H**

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

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

Bit	Function
TR0	**Timer 0 run control bit** Set/cleared by software to turn timer/counter 0 ON/OFF.
TF0	**Timer 0 overflow flag** Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	**Timer 1 run control bit** Set/cleared by software to turn timer/counter 1 ON/OFF.
TF1	**Timer 1 overflow flag** Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.

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



Bit	Function
GATE	**Timer 1/0 gating control** When set, timer/counter "x" is enabled only while "INT x" pin is high and "TRx" control bit is set. When cleared timer "x" is enabled whenever "TRx" control bit is set.
C/$\overline{T}$	**Timer 1/0 counter or timer select bit** Set for counter operation (input from "Tx" input pin). Cleared for timer operation (input from internal system clock).
M1 M0	**Timer 1/0 mode select bits**

M1	M0	Function
0	0	**8-bit timer/counter:** "THx" operates as 8-bit timer/counter "TLx" serves as 5-bit prescaler
0	1	**16-bit timer/counter.** "THx" and "TLx" are cascaded; there is no prescaler
1	0	**8-bit auto-reload timer/counter.** "THx" holds a value which is to be reloaded into "TLx" each time it overflows
1	1	**Timer 0 :** TL0 is an 8-bit timer/counter controlled by the standard timer 0 control bits. TH0 is an 8-bit timer only controlled by timer 1 control bits. Timer 1 : Timer/counter 1 stops

6.2.1.2 Mode 0

Putting either timer/counter 0,1 into mode 0 configures it as an 8-bit timer/counter with a divide-by-32 prescaler. **Figure 6-10** shows the mode 0 operation.

In this mode, the timer register is configured as a 13-bit register. As the count rolls over from all 1's to all 0's, it sets the timer overflow flag TF0. The overflow flag TF0 then can be used to request an interrupt. The counted input is enabled to the timer when TR0 = 1 and either Gate = 0 or $\overline{INT0}$ = 1 (setting Gate = 1 allows the timer to be controlled by external input $\overline{INT0}$, to facilitate pulse width measurements). TR0 is a control bit in the special function register TCON; Gate is in TMOD.

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

Mode 0 operation is the same for timer 0 as for timer 1. Substitute TR0, TF0, TH0, TL0 and $\overline{INT0}$ for the corresponding timer 1 signals in **figure 6-10**. There are two different gate bits, one for timer 1 (TMOD.7) and one for timer 0 (TMOD.3).

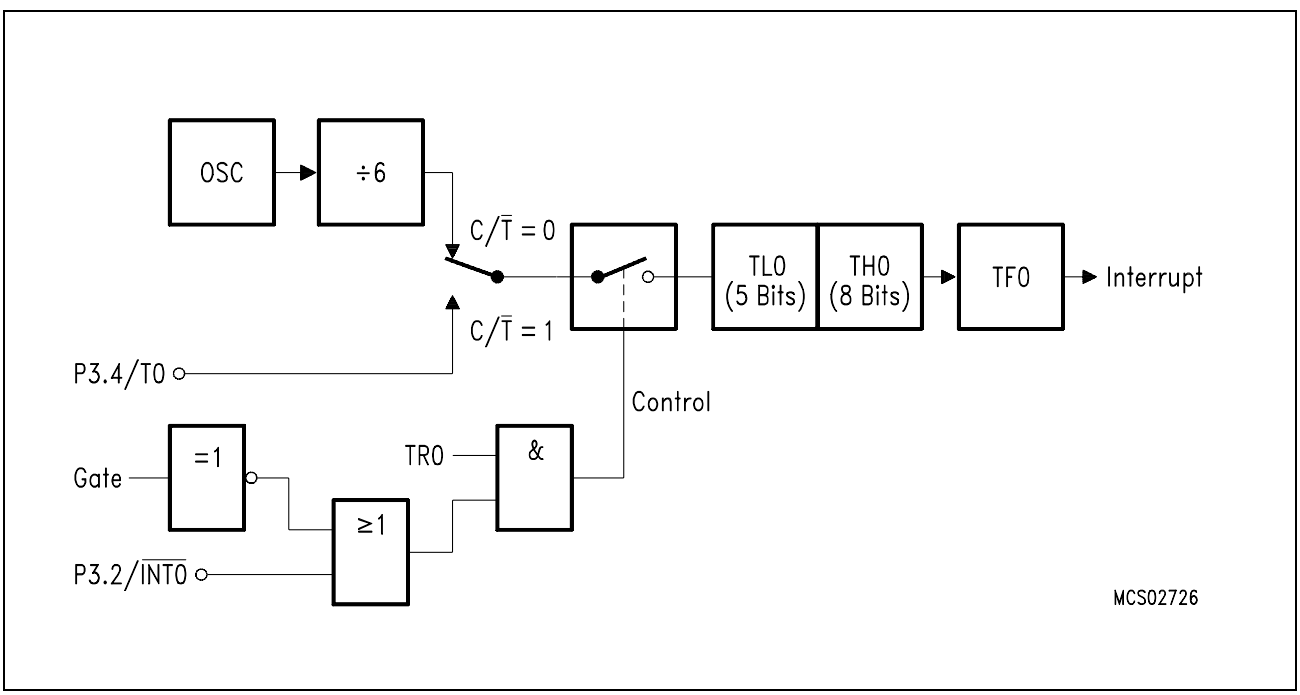


Figure 6-10
Timer/Counter 0, Mode 0: 13-Bit Timer/Counter

6.2.1.3 Mode 1

Mode 1 is the same as mode 0, except that the timer register is running with all 16 bits. Mode 1 is shown in **figure 6-11**.

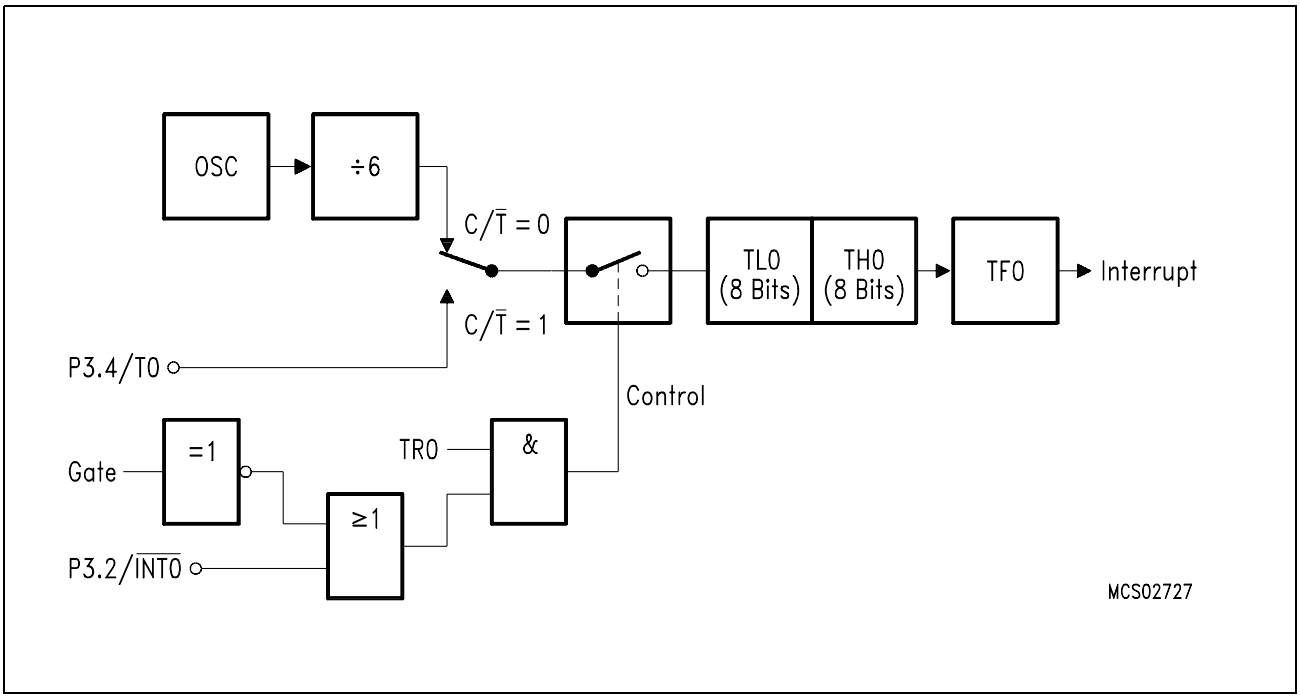


Figure 6-11
Timer/Counter 0, Mode 1: 16-Bit Timer/Counter

6.2.1.4 Mode 2

Mode 2 configures the timer register as an 8-bit counter (TL0) with automatic reload, as shown in **figure 6-12**. 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.

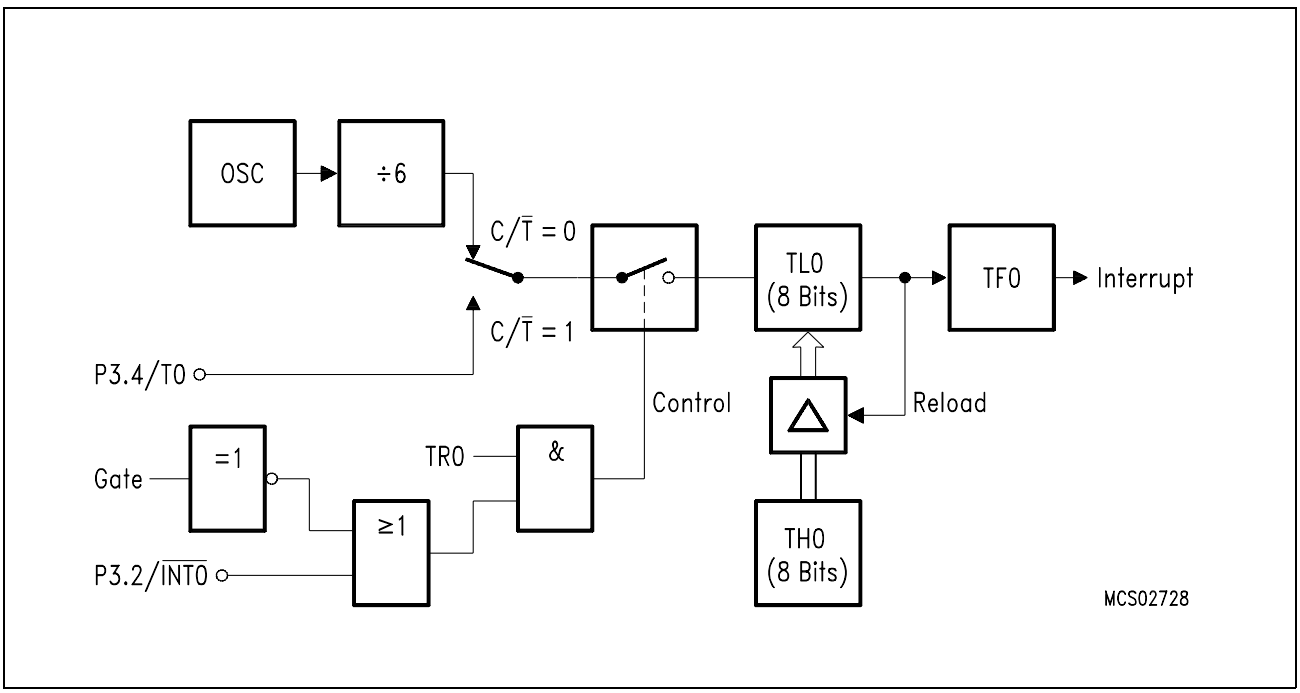


Figure 6-12
Timer/Counter 0,1, Mode 2: 8-Bit Timer/Counter with Auto-Reload

6.2.1.5 Mode 3

Mode 3 has different effects on timer 0 and timer 1. Timer 1 in mode 3 simply holds its count. The effect is the same as setting TR1=0. Timer 0 in mode 3 establishes TL0 and TH0 as two seperate counters. The logic for mode 3 on timer 0 is shown in **figure 6-13**. TL0 uses the timer 0 control bits: C/$\overline{T}$, Gate, TR0, $\overline{INT0}$ and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from timer 1. Thus, TH0 now controls the "timer 1" interrupt.

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

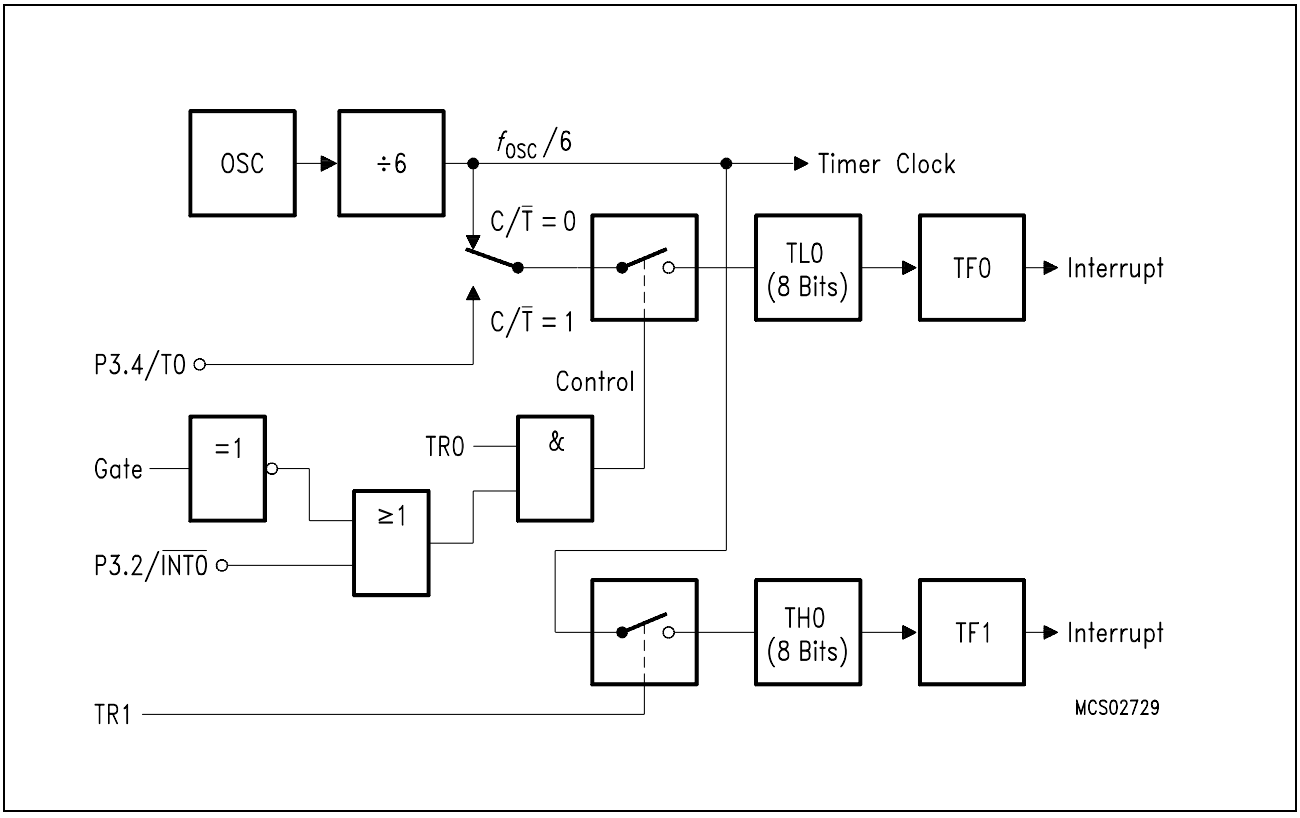


Figure 6-13
Timer/Counter 0, Mode 3: Two 8-Bit Timers/Counters

6.3 SSC Interface

The C541U microcontroller provides a Synchronous Serial Channel unit, the SSC. This interface is compatible to the popular SPI serial bus interface. It can be used for simple I/O expansion via shift registers, for connection of a variety of peripheral components, such as A/D converters, EEPROMs etc., or for allowing several microcontrollers to be interconnected in a master/slave structure. It supports full-duplex or half-duplex operation and can run in a master or a slave mode.

Figure 6-14 shows the block diagram of the SSC. The central element of the SSC is an 8-bit shift register. The input and the output of this shift register are each connected via a control logic to the pin P1.3 / SRI (SSC Receiver In) and P1.4 / STO (SSC Transmitter Out). This shift register can be written to (SFR STB) and can be read through the Receive Buffer Register SRB.

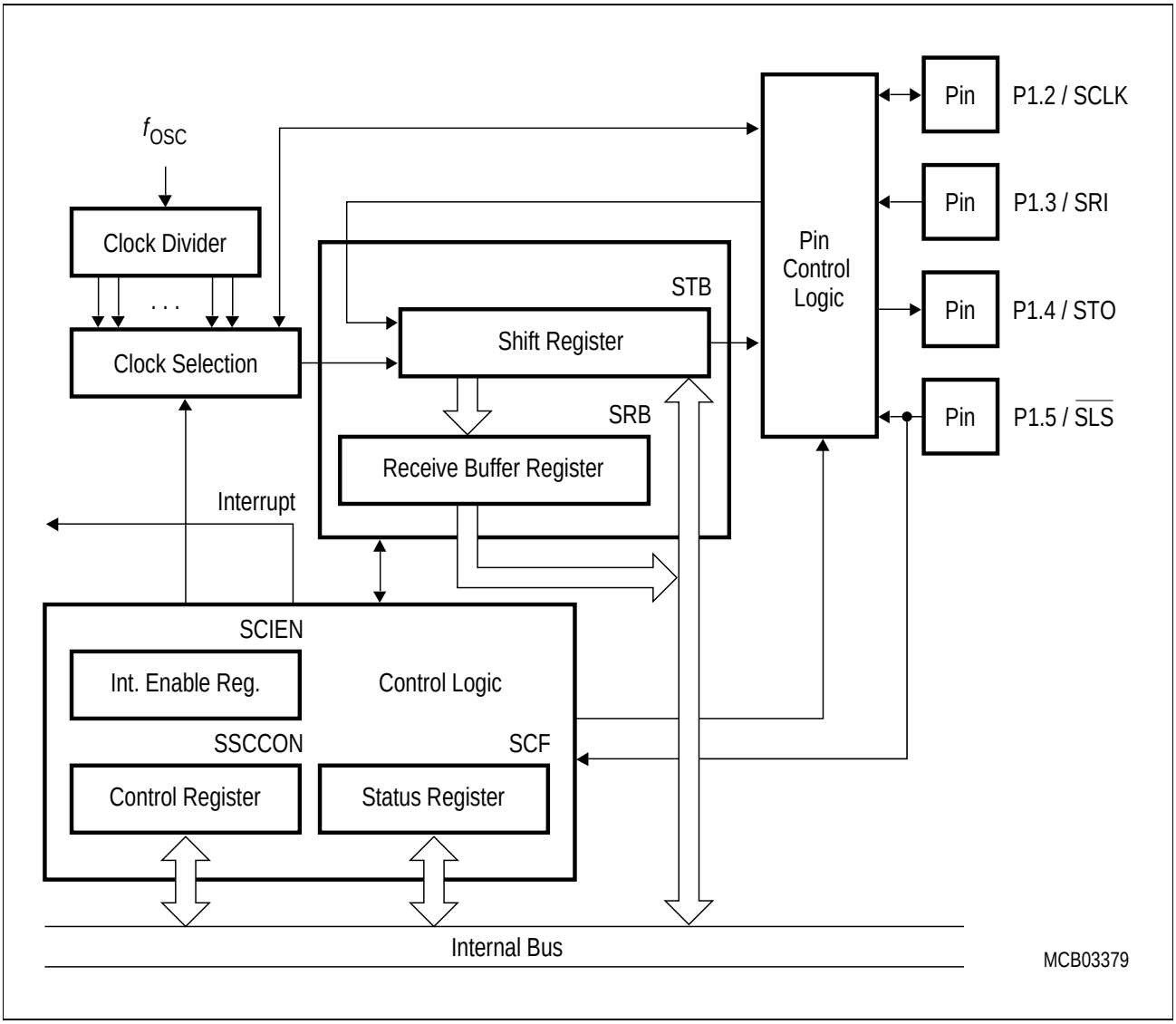


Figure 6-14
SSC Block Diagram

As the SSC is a synchronous serial interface, for each transfer a dedicated clock signal sequence must be provided. The SSC has implemented a clock control circuit, which can generate the clock via a baud rate generator in the master mode, or receive the transfer clock in the slave mode. The clock signal is fully programmable for clock polarity and phase. The pin used for the clock signal is P1.2 / SCLK.

When operating in slave mode, a slave select input $\overline{SLS}$ is provided which enables the SSC interface and also will control the transmitter output. The pin used for this is P1.5 / $\overline{SLS}$. In addition to this there is an additional option for controlling the transmitter output by software.

The SSC control block is responsible for controlling the different modes and operation of the SSC, checking the status, and generating the respective status and interrupt signals.

6.3.1 General Operation of the SSC

After initialization of the SSC, the data to be transmitted has to be written into the shift register STB.

In master mode this will initiate the transfer by resetting the baudrate generator and starting the clock generation. The control bits CPOL and CPHA in the SSCCON register determine the idle polarity of the clock (polarity between transfers) and which clock edges are used for shifting and sampling data (see **figure 6-16**).
While the transmit data in the shift register is shifted out bit per bit starting with the MSB or LSB, the incoming receive data are shifted in, synchronized with the clock signal at pin SCLK. When the eight bits are shifted out (and the same number is of course shifted in), the contents of the shift register is transferred to the receive buffer register SRB, and the transmission complete flag TC is set. If enabled an interrupt request will be generated.
After the last bit has been shifted out and was stable for one bit time, the STO output will be switched to "1" (forced "1"), the idle state of STO. This allows connection of standard asynchronous receivers to the SSC in master mode.

In slave mode the device will wait for the slave select input $\overline{SLS}$ to be activated (=low) and then will shift in the data provided on the receive input according to the clock provided at the SCLK input and the setting of the CPOL ad CPHA bits. After eight bits have been shifted in, the content of the shift register is transferred to the receive buffer register and the transmission complete flag TC is set. If the transmitter is enabled in slave mode (TEN bit set to 1), the SSC will shift out at STO at the same time the data currently contained in the shift register. If the transmitter is disabled, the STO output will remain in the tristate state. This allows more than one slave to share a common select line.

If $\overline{SLS}$ is inactive the SSC will be inactive and the content of the shift register will not be modified.

6.3.2 Enable/Disable Control

Bit SSCEN of the SSCCON register globally enables or disables the synchronous serial interface. Setting SSCEN to “0” stops the baud rate generator and all internal activities of the SSC. Current transfers are aborted. The alternate output functions at pins P1.3 / SRI, P1.4 / STO, P1.5 / $\overline{SLS}$, and P1.2 / SCLK return to their primary I/O port function. These pins can now be used for general purpose I/O.

When the SSC is enabled and in master mode, pins P1.3 / SRI, P1.4 / STO, and P1.2 / SCLK will be switched to the SSC control function. P1.4 / STO and P1.2 / SCLK actively will drive the lines P1.5 / $\overline{SLS}$ will remain a regular I/O pin.

The output latches of port pins dedicated to alternate functions must be programmed to logic 1 (= state after reset).

In slave mode all four control pins will be switched to the alternate function. However, STO will stay in the tristate state until the transmitter is enabled by $\overline{SLS}$ input being low and the TEN control bit is set to 1. This allows for more than one slave to be connected to one select line and the final selection of the slave will be done by a software protocol.

6.3.3 Baudrate Generation (Master Mode only)

The baudrate clock is generated out of the processor clock (fosc). This clock is fed into a resetable divider with seven outputs for different baudrate clocks (fosc/4 to fosc/256). One of these eight clocks is selected by the bits BRS2,1,0 in SSCCON and provided to the shift control logic.

Whenever the shift register is loaded with a new value, the baudrate generation is restarted with the trailing edge of the write signal to the shift register. In the case of CPHA = 0 the baudrate generator will be restarted in a way, that the first SCLK clock transisition will not occur before one half transmit clock cycle time after the register load. This ensures that there is sufficient setup time between MSB or LSB valid on the data output and the first sample clock edge and that the MSB or LSB has the same length than the other bits. (No special care is necessary in case of CPHA=1, because here the first clock edge will be used for shifting).

6.3.4 Write Collision Detection

When an attempt is made to write data to the shift register while a transfer is in progress, the WCOL bit in the status register will be set. The transfer in progress continues uninterrupted, the write will not access the shift register and will not corrupt data.
However, the data written erroneously will be stored in a shadow register and can be read by reading the STB register.

Depending on the operation mode there are different definitions for a transfer being considered to be in progress:

Master Mode :

CPHA=0: from the trailing edge of the write into STB until the last sample clock edge
CPHA=1: from the first SCLK clock edge until the last sample clock edge

Note, that this also means, that writing new data into STB immediately after the transfer complete flag has been set (also initiated with the last sample clock edge) will not generate a write collision. However, this may shorten the length of the last bit (especially at slow baudrates) and prevent STO from switching to the forced "1" between transmissions.

Slave Mode :

CPHA=0: while $\overline{SLS}$ is active
CPHA=1: from the first SCLK clock edge until the last sample clock edge

6.3.5 Master/Slave Mode Selection

The selection whether the SSC operates in master mode or in slave mode has to be made depending on the hardware configuration before the SSC will be enabled.

Normally a specific device will operate either as master or as slave unit. The SSC has no on-chip support for multimaster configurations (switching between master and slave mode operation). Operating the SSC as a master in a multimaster environment requires external circuitry for swapping transmit and receive lines.

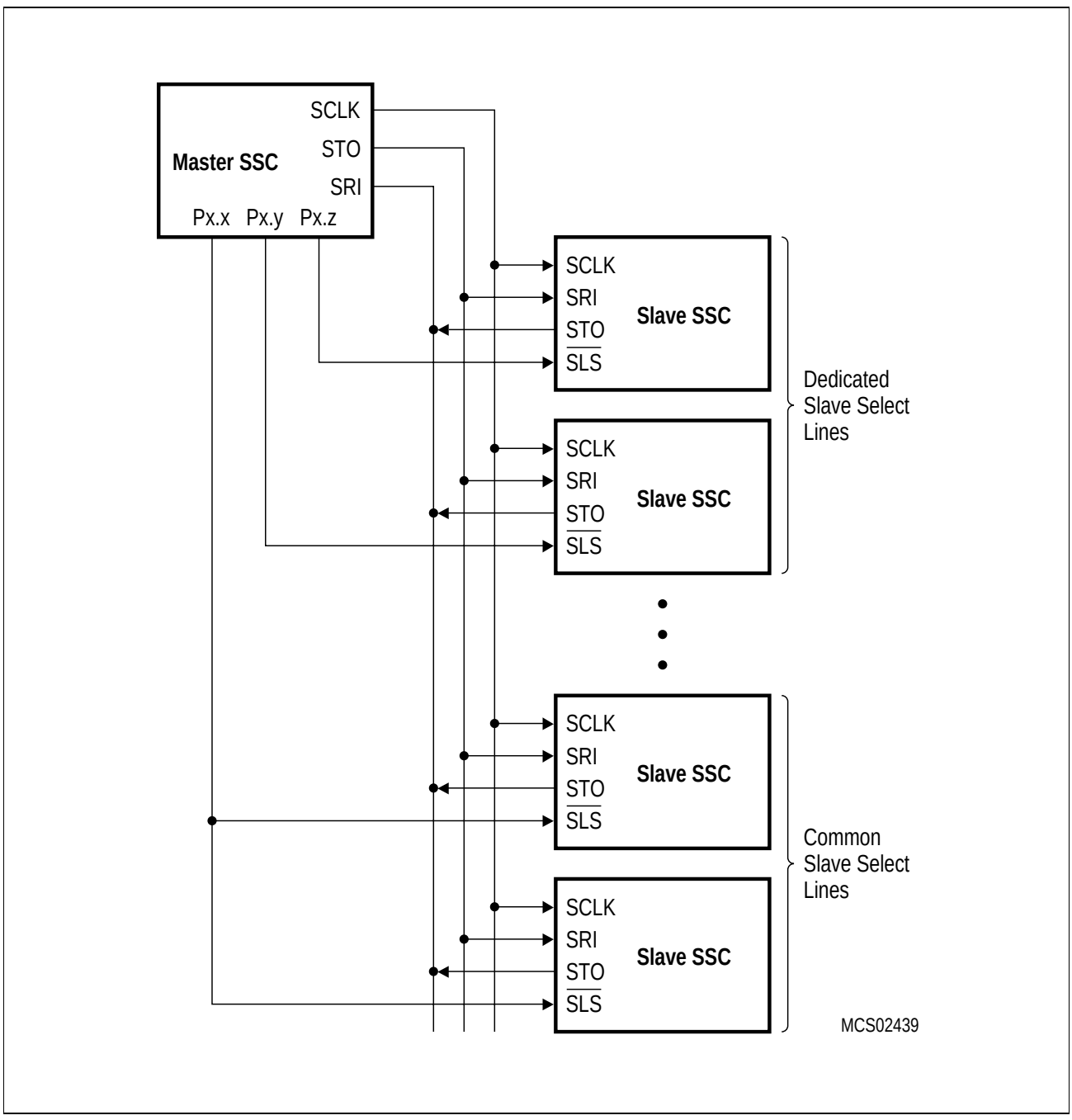


Figure 6-15
Typical SSC System Configuration

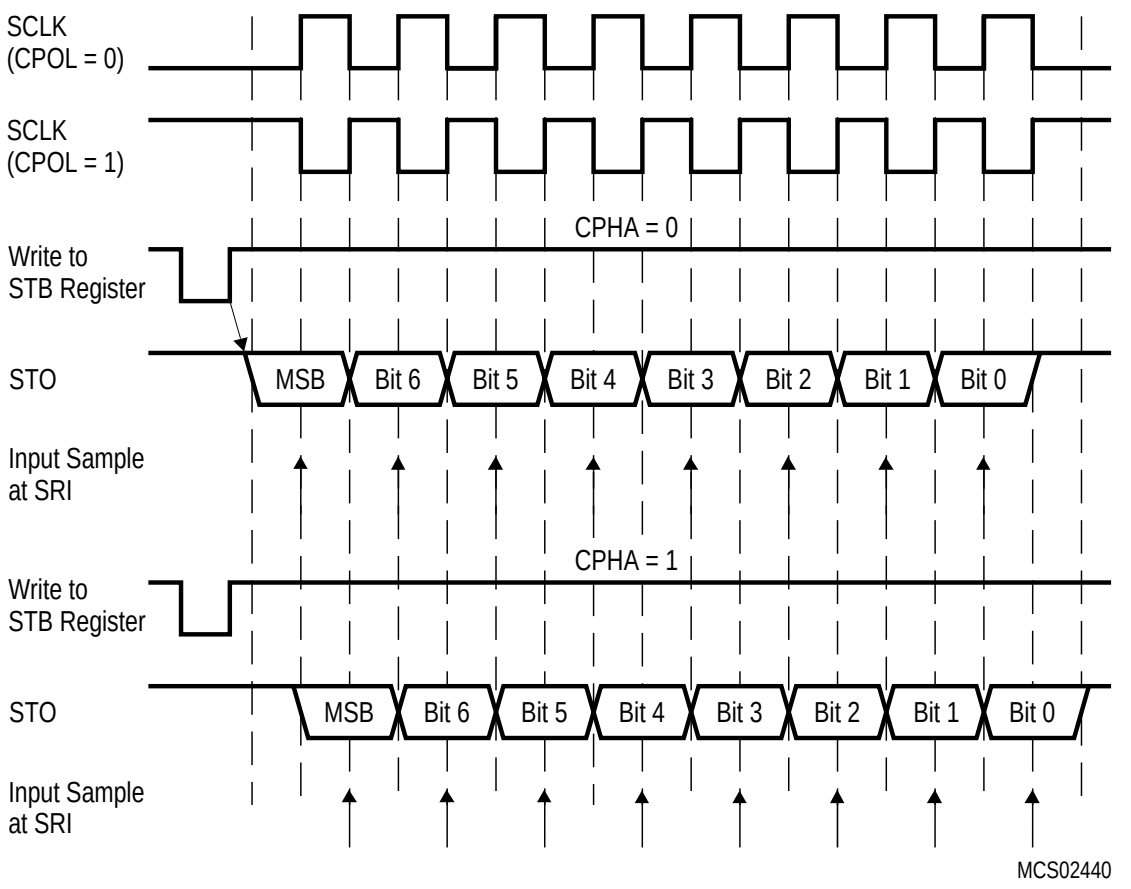
6.3.6 Data/Clock Timing Relationships

The SSC provides four different clocking schemes for clocking the data in and out of the shift register. Controlled by two bits in SSCCON, the clock polarity (idle state of the clock, control register bit CPOL) and the clock/data relationship (phase control, control register bit CPHA), i.e. which clock edges will be used for sample and shift. The following figures show the various possibilities.

6.3.6.1 Master Mode Operation

Figure 6-16 shows the clock/data/control relationship of the SSC in master mode. When CPHA is set to 1, the MSB (or LSB) of the data that was written into the shift register will be provided on the transmitter output after the first clock edge, the receiver input will sample with the next clock edge. The direction (rising or falling) of the respective clock edge is depending on the clock polarity selected. After the last bit has been shifted out, the data output STO will go to the high output level (logic 1) and remain there until the next transmission is started. However, when enabling the SSC after reset, the logic level of STO will be undefined, until the first transmission starts.

When CPHA is 0, the MSB (or LSB) will output immediately after the data was written into the shift register. The first clock edge of SCLK will be used for sampling the input data, the next to shift out the next bit. Between transmissons the data output STO will be "1".



1) MSB shift first mode is assumed (Bit LSBSM in register SCCMOD is 0)

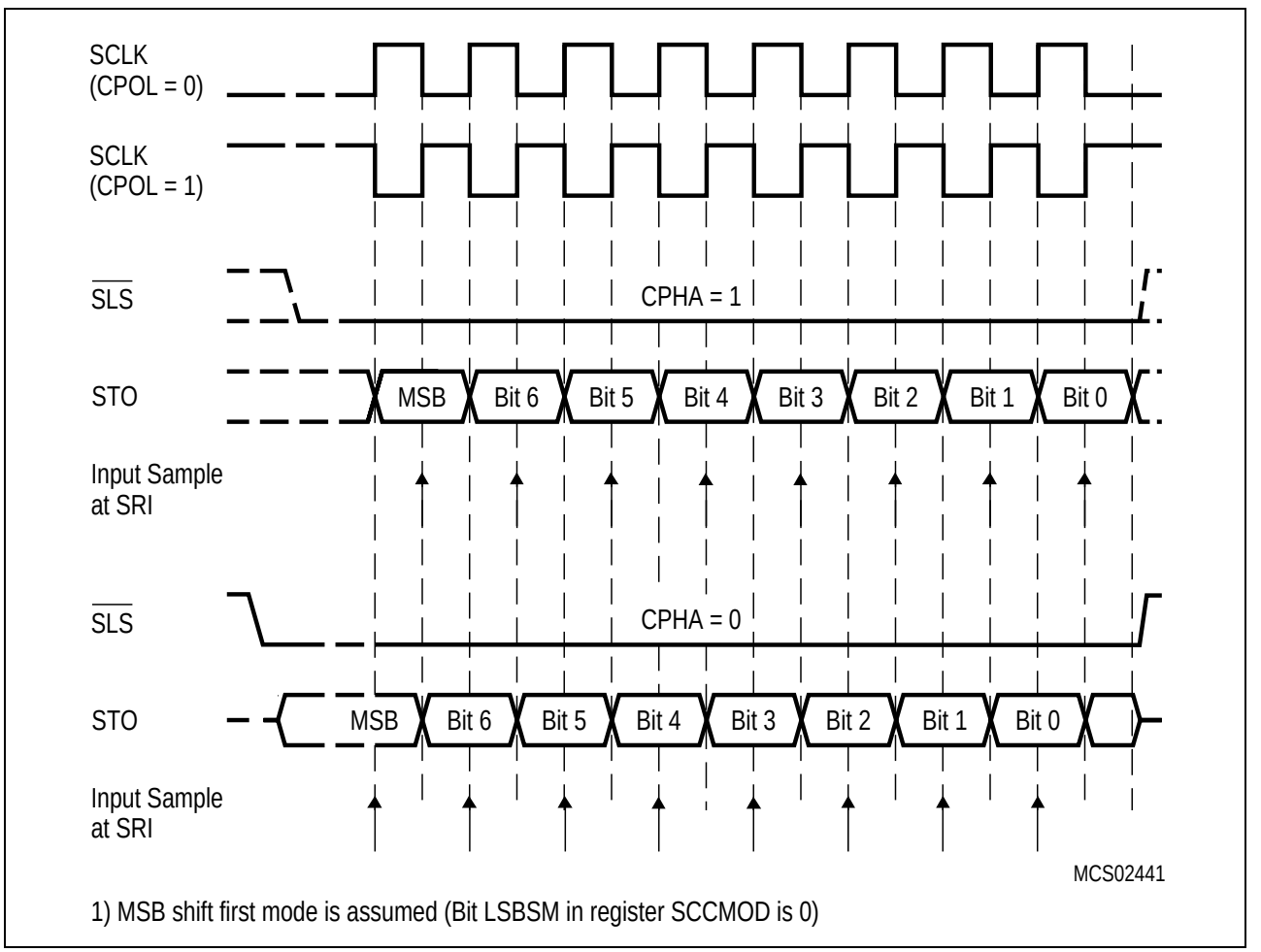
Figure 6-16
Master Mode Operation of SSC

6.3.6.2 Slave Mode Operation

Figure 6-17 shows the clock/data/control relationship of the SSC in slave mode. When $\overline{SLS}$ is active (low) and CPHA is 1, the MSB (or LSB) of the data that was written into the shift register will be provided on the transmitter output after the first clock edge (if the transmitter was enabled by setting the TEN bit to 1), the receiver input will sample the input data with the next clock edge. The direction (rising or falling) of the respective clock edge is depending on the clock polarity selected. In this case (CPHA = 1) the $\overline{SLS}$ input may stay active during the transmission of consecutive bytes.

When CPHA = 0 and the transmitter is enabled, the MSB (or LSB) of the shift register is provided immediately after the $\overline{SLS}$ input is pulled to active state (low). The receiver will sample the input with the first clock edge, and the transmitter will shift out the next bit with the following clock edge. If the transmitter is disabled the output will remain in the high impedance state. In this case (CPHA=0), correct operation requires that the $\overline{SLS}$ input to go inactive between consecutive bytes.

When $\overline{SLS}$ is inactive the internal shift clock is disabled and the content of the shift register will not be modified. This also means that $\overline{SLS}$ must stay active until the transmission is completed.
If during a transmission $\overline{SLS}$ goes inactive before all eight bits are received, the reception process will be aborted and the internal frame counter will be reset. TC will not be set in this case. With the next activation of $\overline{SLS}$ a new reception process will be started.



1) MSB shift first mode is assumed (Bit LSBSM in register SCCMOD is 0)

Figure 6-17
Slave Mode Operation of SSC

6.3.7 Register Description

The SSC interface has six SFRs which are listed in **table 6-3**.

Table 6-3 Special Function Registers of the COMP Unit

Symbol	Description	Address
SSCCON	SSC Control Register	93_H
SCIEN	SSC Interrupt Enable Register	AC_H
SCF	SSC Status Register	AB_H
STB	SSC Transmit Buffer Register	94_H
SRB	SSC Receive Buffer Register	95_H
SSCMOD	SSC Mode Test Register	96_H

The register SSCCON provides the basic control of the SSC functions like general enable/disable, mode selections and transmitter control.

Special Function Register SSCCON (Address 93_H) **Reset Value : 07_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
93_H	SCEN	TEN	MSTR	CPOL	CPHA	BRS2	BRS1	BRS0	SSCCON

Bit	Function
SCEN	**SSC system enable** SCEN=0 : SSC subsystem is disabled, related pins are available as general I/O. SCEN=1 : SSC subsystem is enabled.
TEN	**Slave mode - transmitter enable** TEN =0 : Transmitter output STO will remain in tristate state, regardless of the state of $\overline{SLS}$. TEN = 1 and $\overline{SLS}$ = 0 : Transmitter will drive the STO output. In master mode the transmitter will be enabled all the time, regardless of the setting of TEN.
MSTR	**Master mode selection** MSTR=0 : Slave mode is selected MSTR=1 : Master mode is selected This bit has to be set to the correct value depending on the hardware setup of the system before the SSC will be enabled. It must not be modified afterwards. There is no on-chip support for dynamic switching between master and slave mode operation.

Bit	Function
CPOL	**Clock polarity** This bit controls the polarity of the shift clock and in conjunction with the CPHA bit which clock edges are used for sample and shift. CPOL=0 : SCLK idle state is low. CPOL=1 : SCLK idle state is high.
CPHA	**Clock phase** This bit controls in conjunction with the CPOL bit controls which clock edges are used for sample and shift CPHA=0 : The first clock edge of SCLK is used to sample the data, the second to shift the next bit out at STO. In master mode the transmitter will provide the first data bit on STO immediately after the data was written into the STB register. In slave mode the transmitter (if enabled via TEN) will shift out the first data bit with the falling edge of $\overline{SLS}$. CPHA=1 : The first data bit is shifted out with the first clock edge of SCLK and sampled with the second clock edge
BRS2, BRS1, BRS0	**Baudrate selection bits** These bits select one of the possible divide factors for generating the baudrate out of the micrcontroller clock rate f_{osc} . The baudrate is defined by . $$Baudrate = \frac{fosc}{Dividefactor} = \frac{fosc}{2 \bullet 2^{BRS(2-0)}}$$ for BRS (2-0) ≠ 0

BRS(2-0)	Divide Factor	Example: Baudrate for fosc = 12 MHz
0	reserved	reserved
1	reserved	reserved
2	8	1.5 MBaud
3	16	750 kBaud
4	32	375 kBaud
5	64	187.5 kBaud
6	128	93.75 kBaud
7	256	46.875 kBaud

Note: SSCCON must be programmed only when the SSC is idle. Modifying the contents of SSCCON while a transmission is in progress will corrupt the current transfer and will lead to unpredictable results.

This register enables or disables interrupt request for the status bits. SCIEN must only be written when the SSC interrupts are disabled in the general interrupt enable register IEN2 ($9A_H$) using bit ESSC otherwise unexpected interrupt requests may occur.

Special Function Register SCIEN (Address AC_H) **Reset Value : $XXXXXX00_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
AC_H	–	–	–	–	–	–	WCEN	TCEN	SCIEN

Bit	Function
–	Reserved for future use.
WCEN	**SSC write collision interrupt enable** WCEN =0 : No interrupt request will be generated if the WCOL bit in the status register SCF is set. WCEN=1 : An interrupt is generated if the WCOL bit in the status register SCF is set.
TCEN	**SSC transfer completed interrupt enable** TCEN =0 : No interrupt request will be generated if the TC bit in the status register SCF is set. TCEN=1 : An interrupt is generated if the TC bit in the status register SCF is set.

Note: The SSC interrupt behaviour is in addition affected by bit ESSC in the interrupt enable register IEN2 and by bit 2 in the interrupt priority registers IP0 and IP1.

Special Function Register SCF (Address AB_H) **Reset Value : $XXXXXX00_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
AB_H	–	–	–	–	–	–	WCOL	TC	SCF

Bit	Function
–	Reserved for future use.
WCOL	**SSC write collision detect** If WCOL is set it indicates that an attempt was made to write to the shift register STB while a data transfer was in progress and not fully completed. This bit will be set at the trailing edge of the write signal during the erronous write attempt. This bit can be reset in two different ways : 1. writing a "0" to the bit (bit access, byte access or read-modify-write access); 2. by reading the bit or the status register, followed by a write access to STB. If bit WCEN in the SCIEN register is set, an interrupt request will be generated if WCOL is set.
TC	**SSC transfer completed** If TC is set it indicates that the last transfer has been completed. It is set with the last sample clock edge of a reception process. This bit can be reset in two different ways: 1. writing a "0" to the bit (bit access, byte access or read-modify-write access) after the receive buffer register SRB has been read; 2. by reading the bit or the status register, followed by a read access to SRB. If bit TCEN in the SCIEN register is set, an interrupt request will be generated if TC is set.

The register STB (at SFR address 94_H) holds the data to be transmitted while SRB (at SFR address 95_H) contains the data which was received during the last transfer. A write to the STB places the data directly into the shift register for transmission. Only in master mode this also will initiate the transmission/reception process. When a write collision occurs STB will hold the value written erroneously. This value can be read by reading from STB.

A read from the receive buffer register SRB will transfer the data of the last transfer completed. This register must be read before the next transmission completes or the data will be lost. There is no indication for this overrun condition.

Special Function Register STB (Address 94_H) **Reset Value : XX_H**
Special Function Register SRB (Address 95_H) **Reset Value : XX_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
94_H	.7	.6	.5	.4	.3	.2	.1	.0	STB
95_H	.7	.6	.5	.4	.3	.2	.1	.0	SRB

After reset the contents of the shift register and the receive buffer register are undefined.

The register SSCMOD is used to enable test modes during factory test. It must not be written or modified during normal operation of the C541U.

Special Function Register SSCMOD (Address 96_H) **Reset Value : 00_H**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
96_H	LOOPB	TRIO	0	0	0	0	0	LSBSM	SSCMOD

Bit	Function
LOOPB	**SSC loopback enable** This bit should be used for test purposes only. LOOPB = 0 : The SSC operates as specified. LOOPB = 1 : The STO output is connected internally via an inverter to the SRI input, allowing to check the transfer locally without a second SSC device.
TRIO	**SSC disable tristate mode of SSC inputs** This bit should be used for test purposes only. TRIO = 0 : The SSC operates as specified. TRIO = 1 : The SSC inputs will be connected to the output latch of the corresponding port pin. This allows a test of the SSC in slave mode by simulating a transfer via a program setting the port latches accordingly.
5-1	All bits of this register are set to 0 after reset. When writing SSCMOD, these bits must be written with 0.
LSBSM	**SSC LSB shift mode** If LSBSM is cleared, the SSC will shift out the MSB of the data first LSB last. If LSBSM is set, the SSC will shift out LSB first and MSB last.

6.4 USB Module

The USB module in the C541U handles all transactions between the serial USB bus and the internal (parallel) bus of the microcontroller. The USB module includes several units which are required to support data handling with the USB bus : the on-chip USB bus transceiver, the USB memory with two pages of 128 bytes each, the memory management unit (MMU) for USB and CPU memory access control, the UDC device core for USB protocol handling, the microcontroller interface with the USB specific special function registers and the interrupt control logic. A clock generation unit provides the clock signal for the USB module for full speed and low speed USB operation. The following sections describe the full speed operation, while the low speed operation is described specifically in section 6.4.8. **Figure 6-18** shows the block diagram of the functional units of the USB module with their interfaces.

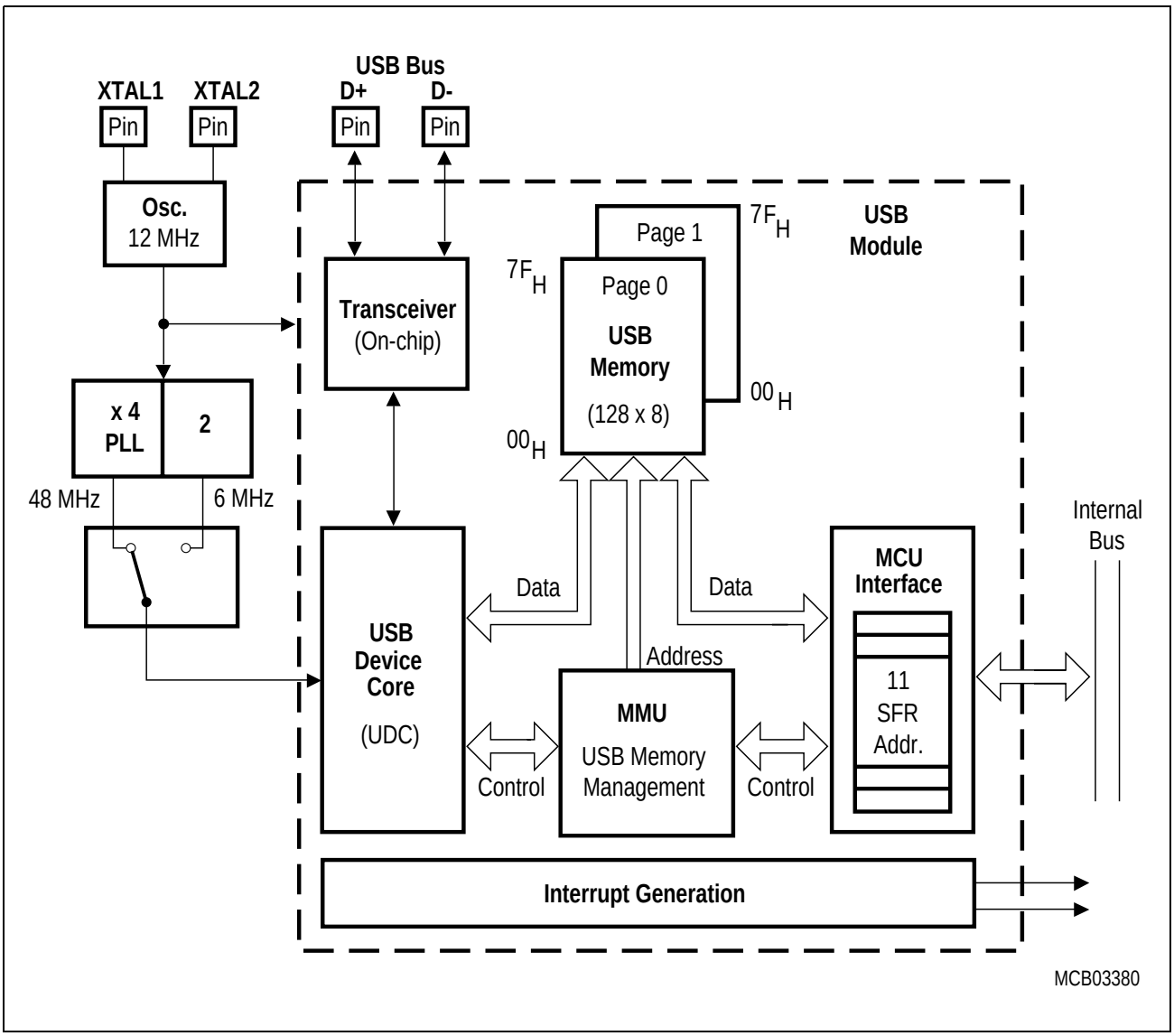


Figure 6-18
USB Module Block Diagram

6.4.1 Transfer Modes

USB data transfers take place between host software and a particular endpoint on a USB device. A given USB device may support multiple data transfer endpoints. The USB host treats communications with any endpoint of a USB device independently from any other endpoint. Such associations between the host software and a USB device endpoint are called pipes. As an example, a given USB device could have an endpoint supporting a pipe for transporting data to the USB device and another endpoint supporting a pipe for transporting data from the USB device.

The USB architecture comprehends four basic types of data transfers.

Table 6-4
USB Transfer Modes

Mode	Function
Control	Control data are used to configure devices, data transmission is lossless. Control pipes are bidirectional, data transfer is possible in both directions via one pipe. Endpoint 0 is always configured as control endpoint with a maximum buffer length of 8 bytes. The control endpoint can be configured to handle data packets of 64 bytes maximum length.
Isochronous (full speed mode only)	Isochronous data are continuous and real-time in creation and consumption, such as voice data. In this case, real-time is defined from frame to frame. Isochronous data transfer has the highest priority, but is not always lossless. Isochronous pipes are always unidirectional, so one endpoint can be associated to an IN pipe or an OUT pipe. The C541U supports up to 64 bytes.
Interrupt	Interrupt data are a small amount of data, which are transferred to the host every n frames, with n being programmable by the host. Data delivery is lossless. Interrupt pipes are always unidirectional IN pipes, the maximum data packet length is limited to 64 bytes.
Bulk (full speed mode only)	Bulk data can be a larger amount of data, which can be split by the host in several data packets witin one frame. Data delivery is lossless. Bulk pipes are always unidirectional, so one endpoint can be associated to an IN pipe or an OUT pipe. The maximum data packet length is limited to 64 bytes.

6.4.2 USB Memory Buffer Modes

6.4.2.1 Overview

Every endpoint of the USB module in the C541U can operate in two modes, dual buffer mode and single buffer mode. Each mode provides random or sequential access to the USB memory. **Figure 6-19** shows the possible buffer modes.

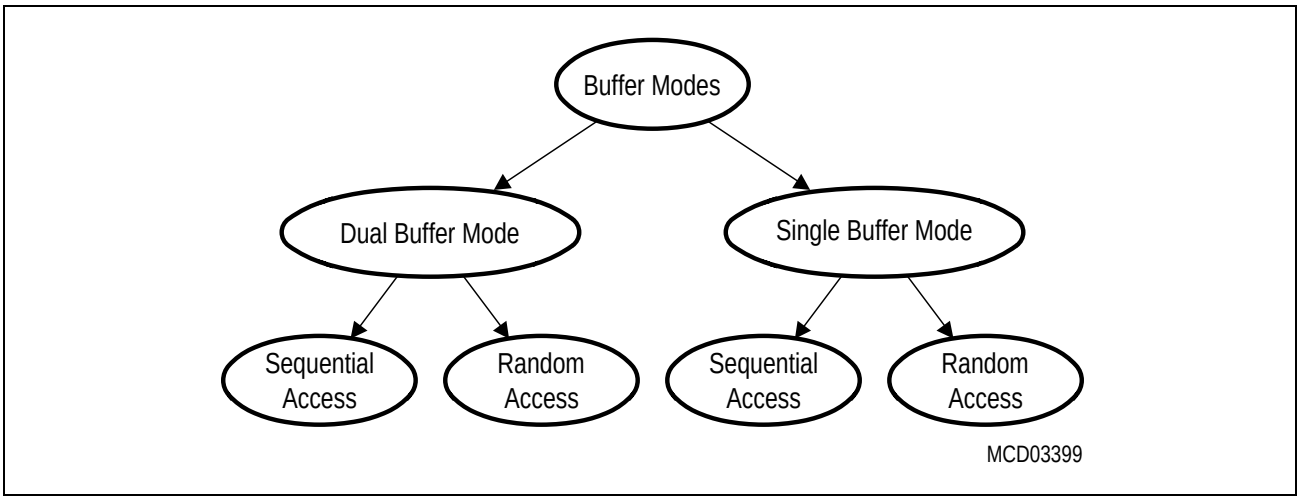


Figure 6-19
Buffer Modes of the C541U USB Module

Single Buffer Mode

In single buffer mode, the USB and the CPU use one common USB memory page. The active buffer page is either page 0 or page 1.

Dual Buffer Mode

In dual buffer mode the USB and the CPU write into different USB memory pages allowing back-to-back data transfers. Switching between the pages is done fully automatically, enabling a high data transfer rate between CPU and USB module.

Random Access

Random access is available in single buffer mode and dual buffer mode. Random access allows to change only a few bytes in a data block of the USB memory buffer. When the CPU has modified the bytes in the data block, setting of bit DONE by software marks the buffer ready for transmission or reception of data over the USB pipe. For modification of a specific byte in the buffer, the CPU must write the address to SFR ADROFF and read/write the data byte from/to register USBVAL.

Sequential access

In sequential access mode the CPU accesses the data register USBVAL continuously without setting the address of the next USB memory buffer location. This is done automatically if bit INCE (increment enable) in the related SFR EPBCRn is set. After a specific number of CPU accesses (as defined in SFR EPLENn), the buffer has been read/written by the CPU and is empty/full. Setting of bit DONE by software, manually or automatically, marks the USB buffer ready.

Note : Only buffers for device to host pipes can be written.

6.4.2.2 Single Buffer Mode

In single buffer mode the USB and the CPU share one common USB memory page. The active buffer page can be either page 0 or page 1. Back-to-back transfers are not possible in this mode. Easy data storage and controlling can be achieved in this mode. E.g. a once created data set for an interrupt endpoint can be stored permanently in USB memory. As a result, an additional memory space for data storage is no longer needed.

6.4.2.2.1 USB Write Access

Figure 6-20 shows the basic flowchart of a USB write access to one USB memory buffer in single buffer mode.

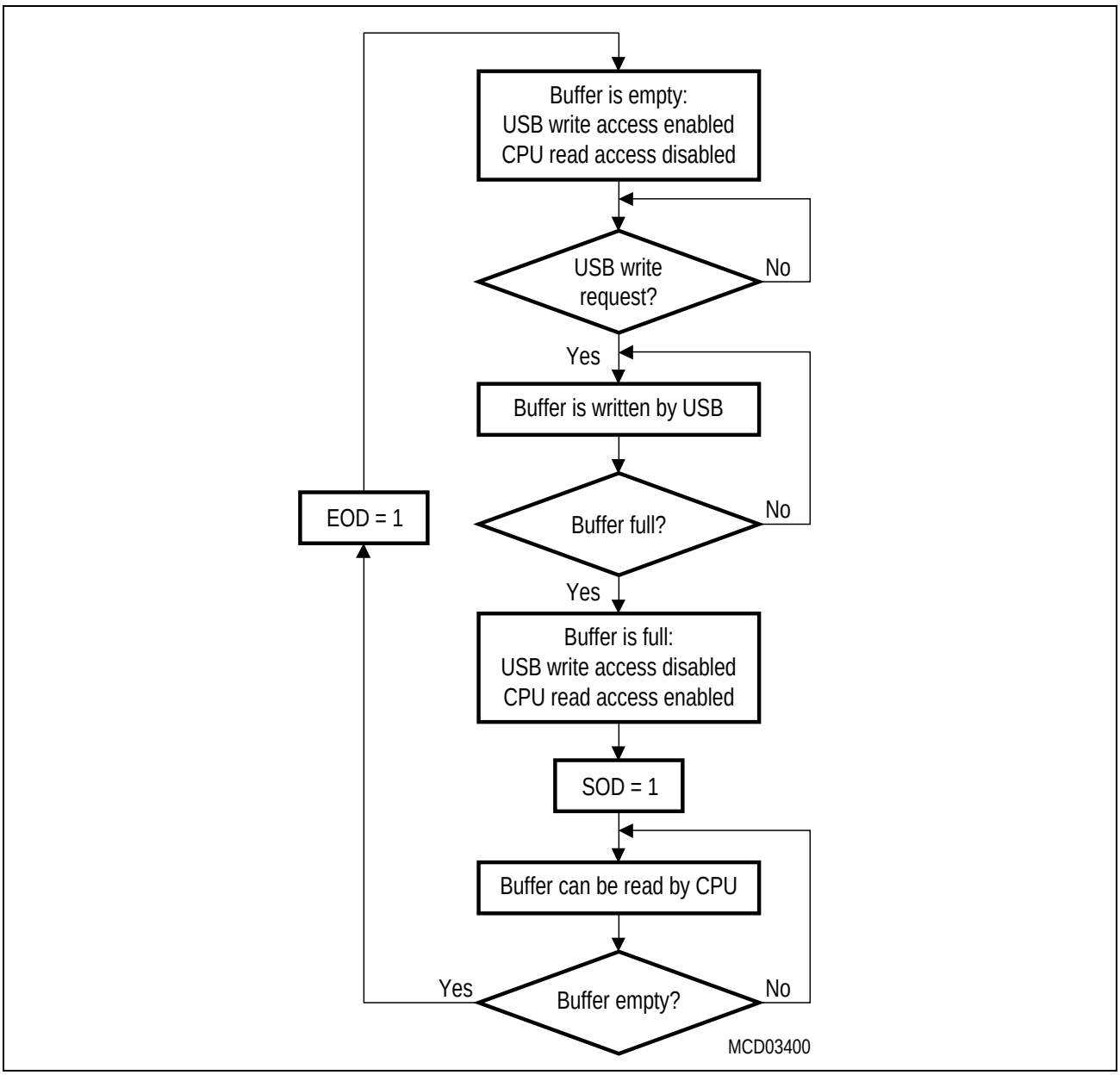


Figure 6-20
USB Write Access in Single Buffer Mode - Buffer Handling

Figure 6-21 shows more details of an USB write access to USB memory in single buffer mode. After SOF(n) (start of frame) occured at ①, the USB starts writing at ② a fixed number of bytes into the USB memory. A byte counter is incremented after every USB memory write operation. When the USB memory write operation (Len(n)) is finished correctly, bit SOD (start of data) is set at ③, indicating a full USB memory buffer. Furthermore, the byte counter value is stored in the corresponding length register, indicating the number of bytes which have been transferred and can be now read by the CPU. Subsequently, the CPU can read data bytes from USB memory, generating an EOD (end of data) at ④ after the last byte has been read. Bit EOD set indicates an empty USB buffer, which now can be written again by the USB.

Figure 6-21 also shows a second USB write access operation with a different number of bytes (Len(n+1)), where the CPU read operation from the USB memory is interrupted twice.

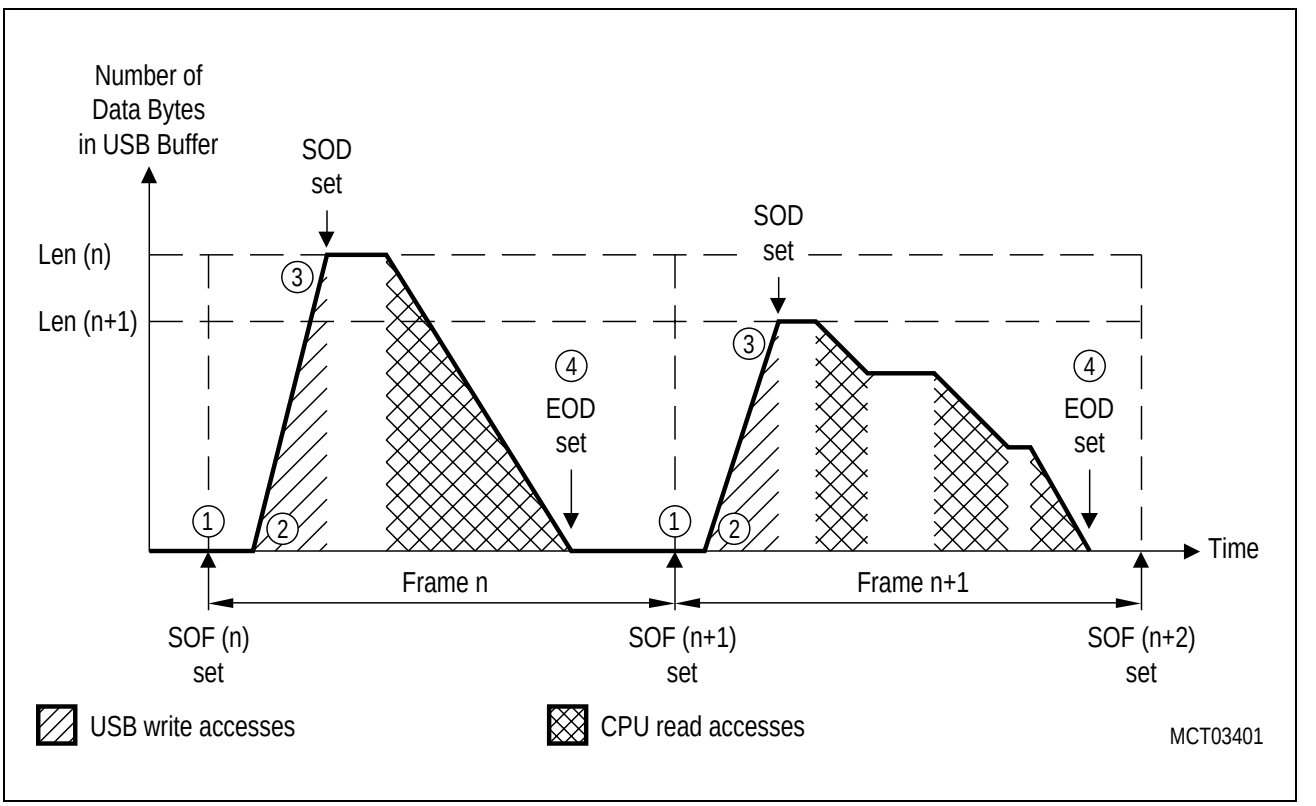


Figure 6-21
Single Buffer Mode : Standard USB Write Access

Note: The CPU accesses shown in the following diagrams assume that bit INCE in the corresponding endpoint control register is set.
A frame is the 1 ms time interval defined by the USB host.
Every frame begins with a SOF token (start-of-frame).

6.4.2.2.2 USB Read Access

Figure 6-22 shows the basic flowchart of a USB read access from one USB memory buffer in single buffer mode.

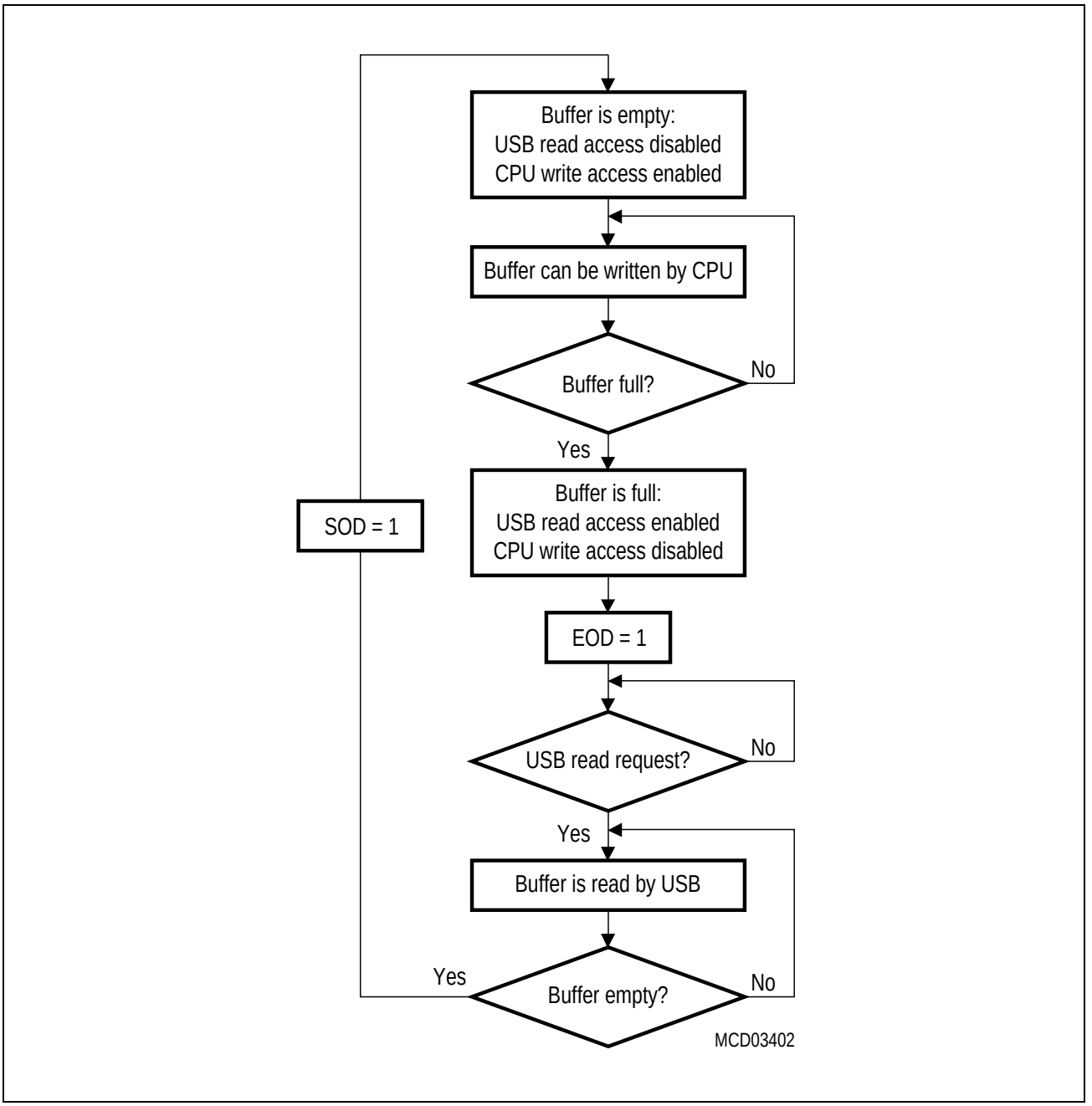


Figure 6-22
USB Read Access in Single Buffer Mode - Buffer Handling

The standard USB read access as shown in **figure 6-23** supports random and sequential CPU access mode of the USB memory. The memory buffer full condition is true when a predefined number of bytes (MaxLen) has been written by the CPU or when bit DONE has been set by software.

After SOF(n) occured at ① with a full USB memory buffer, the USB reads the buffer. Bit SOD is set at the end of the USB buffer read operation at ②, indicating an empty USB memory buffer. Now, the CPU can write again data into the USB memory buffer until a determined number (MaxLen) of bytes are transfered or until bit DONE has been set by software. The MaxLen value must be previously set by software. When the actual USB memory buffer address offset is equal to MaxLen, bit EOD is set at ③ to indicate a full buffer. The USB memory buffer address offset is automatically incremented with every CPU write access to USB memory buffer if bit INCE is set.

During the next frame (after SOF(n+1)) is set at ④) the USB memory buffer can be read by the USB. Bit SOD is set again when the USB memory buffer becomes empty again. If bit DONE is set by the CPU (at ⑤), the buffer is declared by the CPU to be full, even if the address offset does not reach the value of MaxLen.

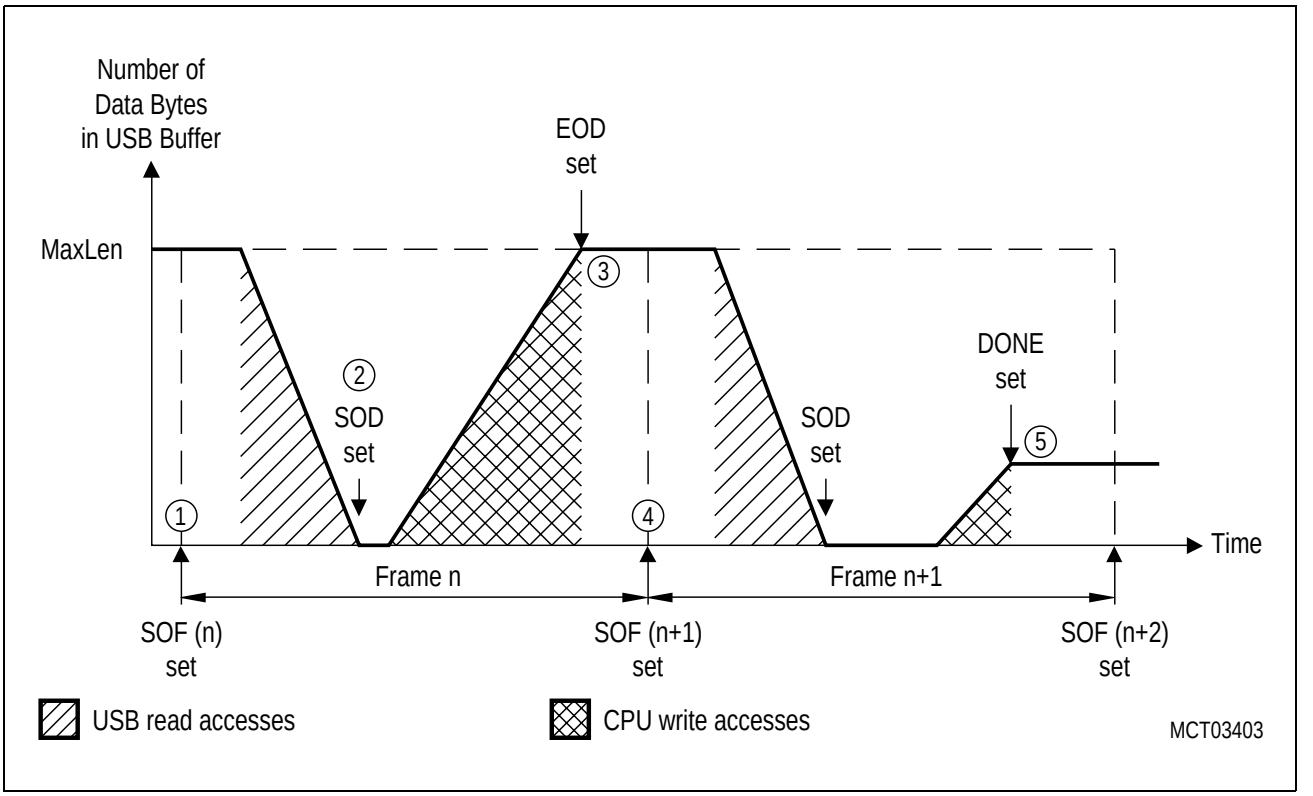


Figure 6-23
Single Buffer Mode : Standard USB Read Access

The start-of-frame-done enable feature (SOFDE=1) is useful for USB memory read accesses when the number of data bytes to be transferred from CPU to USB is not predictable (see **figure 6-24**). The CPU can write data as desired to USB memory until a SOF occures (every 1 ms). The automatic setting of bit SOF causes bit EOD to be set (at ①). This indicates the CPU that no CPU action on this buffer is required until a USB read operation has been finished (bit SOD set at ②). Setting of SOD indicates an empty USB memory to the CPU which can start again writing data into USB memory.

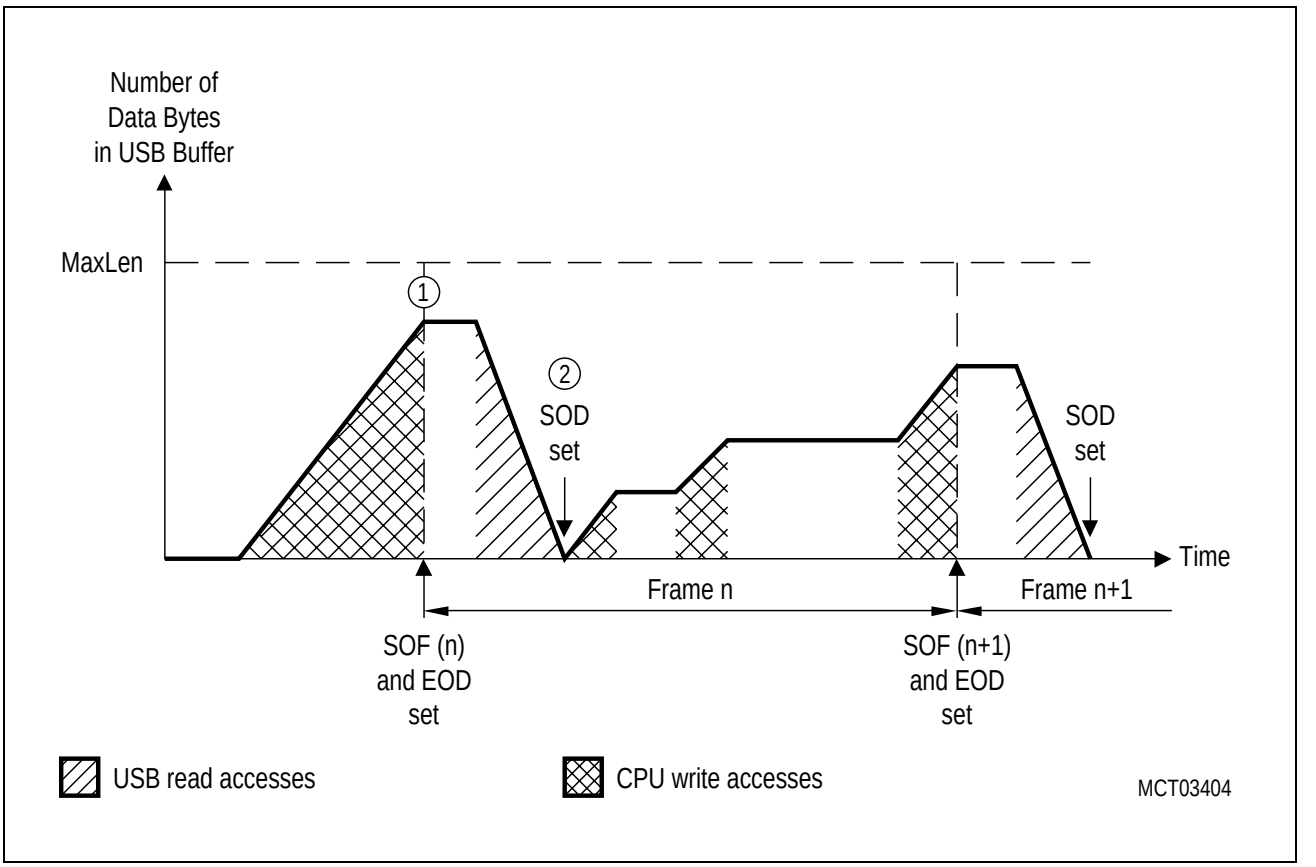


Figure 6-24
Single Buffer Mode : USB Read Access with Start-of-Frame-Done Enabled

6.4.2.3 Dual Buffer Mode

In dual buffer mode, both USB memory pages (page 0 and page 1) are used for data transfers. The logical assignment of the memory pages to CPU or USB is automatically switched. The following two figures show the buffer handling concept in dual buffer mode for the USB read access and USB write access.

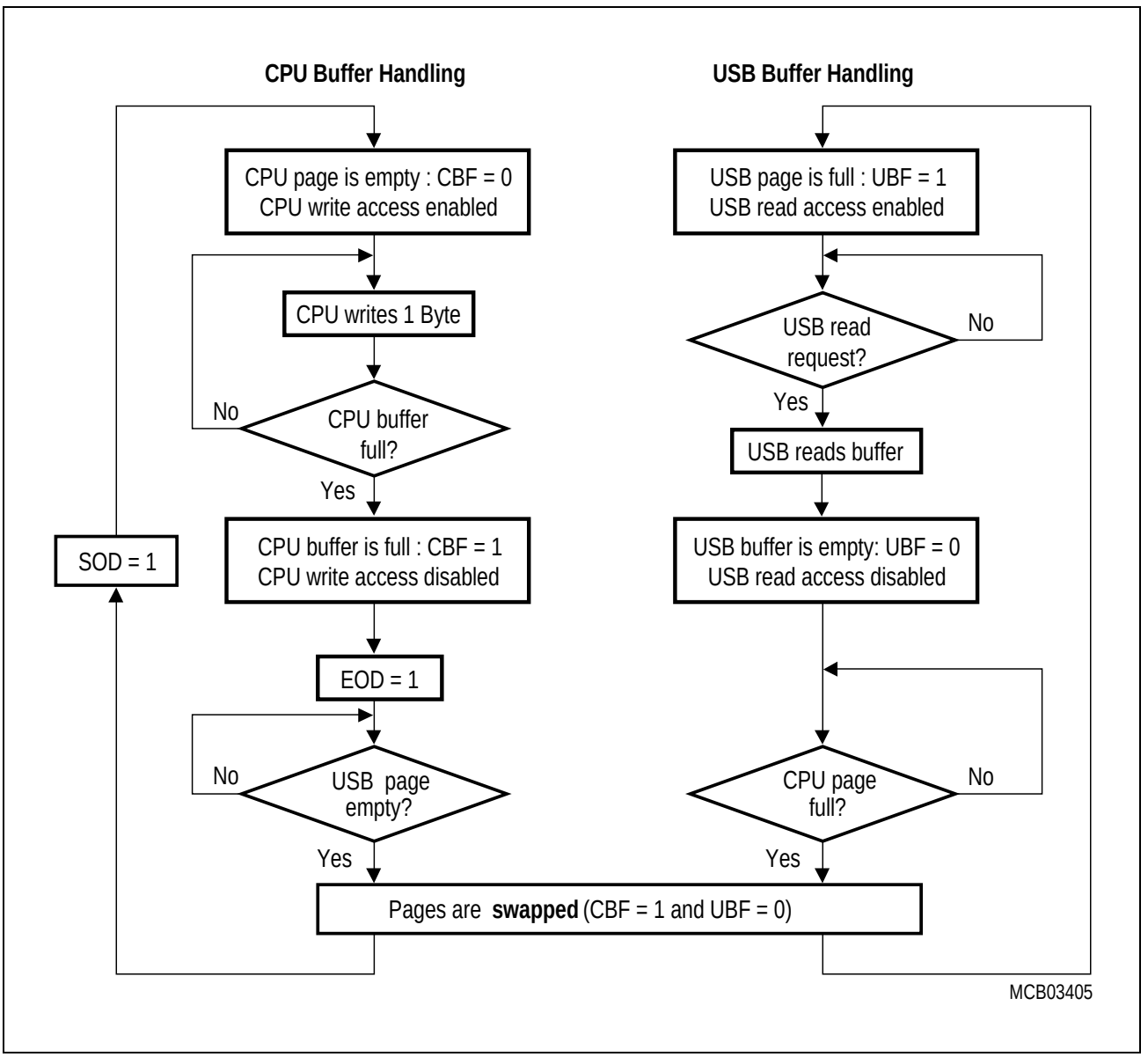


Figure 6-25
USB Read Access in Dual Buffer Mode - Buffer Handling

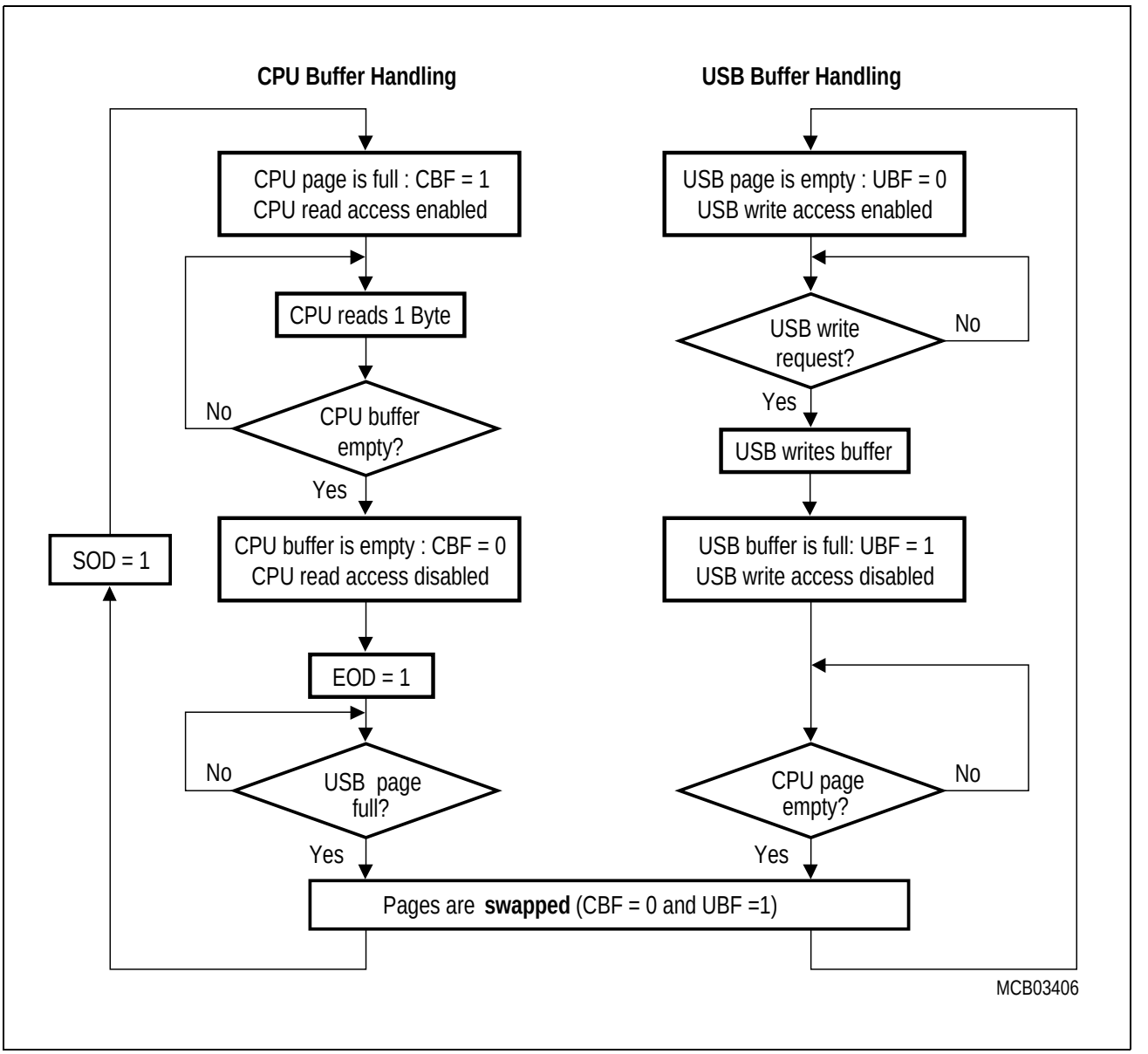


Figure 6-26
USB Write Access in Dual Buffer Mode - Buffer Handling

Figure 6-27 describes an example of a USB read operation in sequential mode with both buffers empty at the beginning of the USB read operation.
The CPU starts writing data with sequential access (INCE=1) to the buffer assigned to the CPU at ①. By definition, the buffer is full when MaxLen is reached at ②. The second buffer assigned to the USB is empty (UBF=0) and as a result both buffers are logically swapped. Now the buffer assigned to USB is full (UBF=1) and an USB read access can take place. After the USB read access, the buffer assigned to the USB is empty again with UBF=0. During the USB read access the CPU is still allowed to write into its assigned buffer. When reaching MaxLen at ③, the CPU buffer is full and both buffers are again logically swapped. The USB further execute its read access.

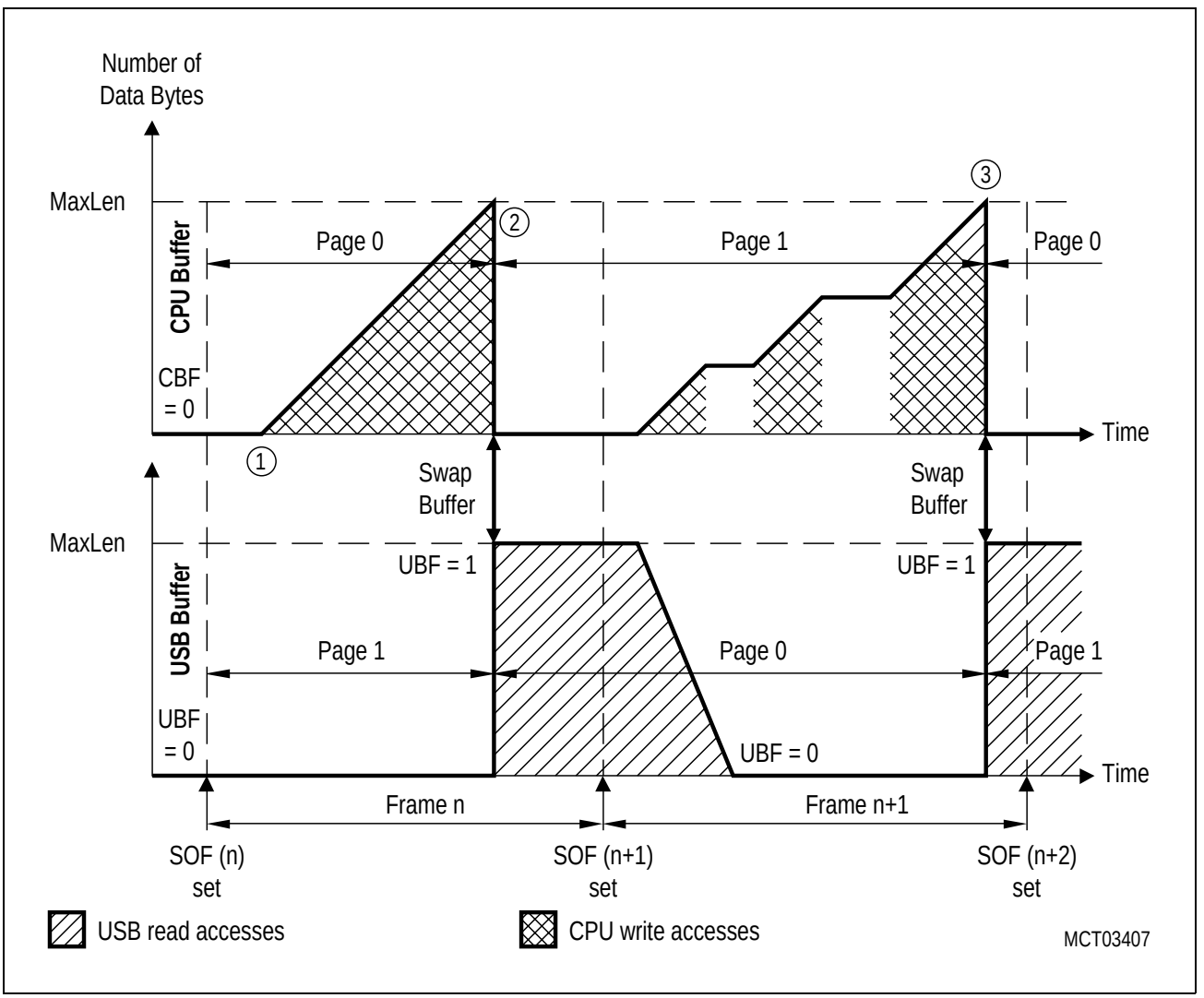


Figure 6-27
Dual Buffer Mode USB Read Access : Buffer Switching when MaxLen is reached

In dual buffer mode, the physical assignment of the USB memory pages (page 0 or page 1) to either CPU buffer or USB buffer is controlled automatically in the USB module and cannot be selected by software.

Another way to initiate buffer switching is setting bit DONE by software. This feature, which is shown in **figure 6-28** for USB read access, can be used to transfer a variable number of bytes. The maximum number of bytes to be transferred is still determined by MaxLen, which is not changed when bit DONE is set. The actual packet length (Len1 or Len2) is the number of bytes which have been written to the buffer before bit DONE is set.

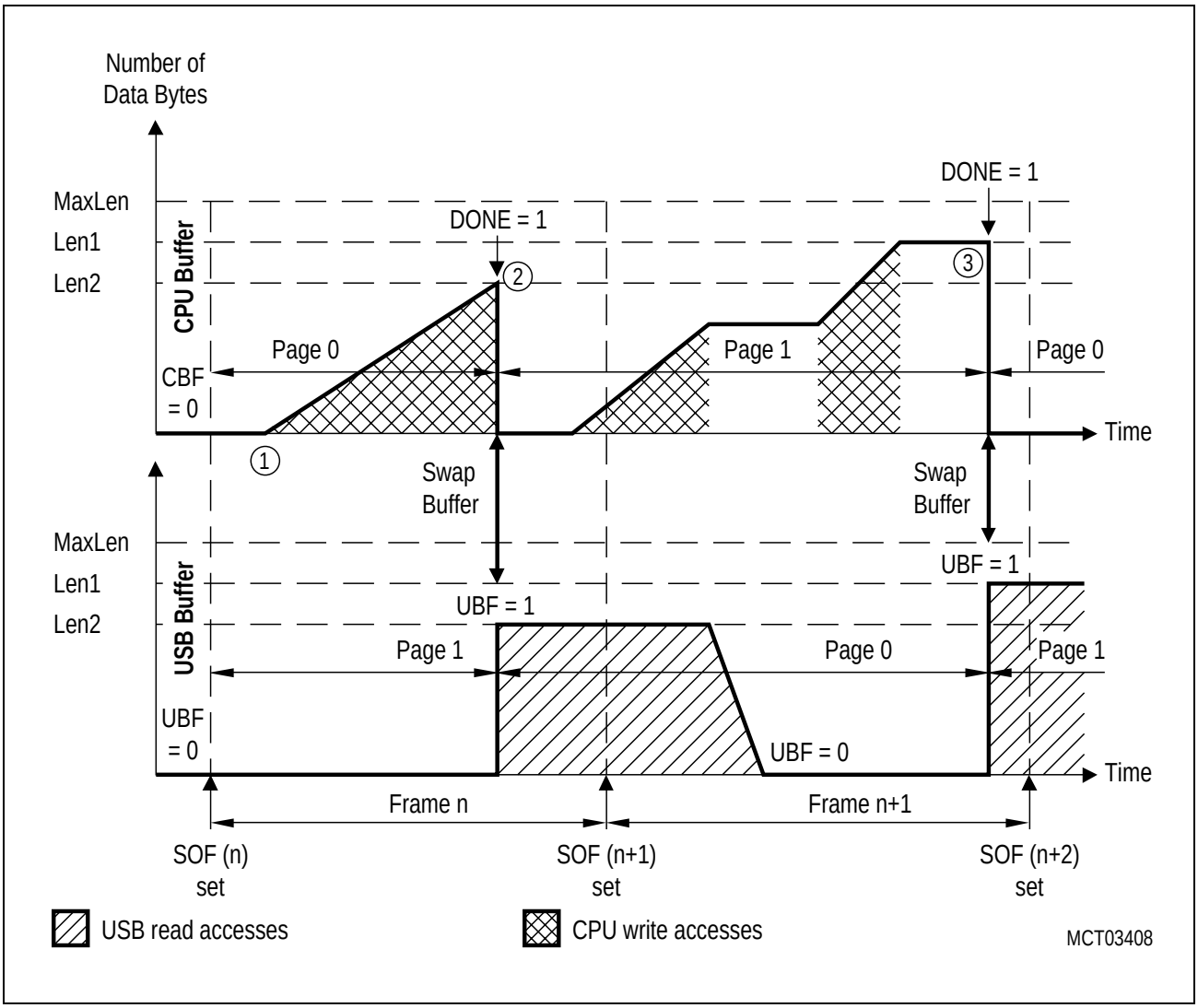


Figure 6-28
Dual Buffer Mode USB Read Access : Buffer Switching by Setting Bit DONE

If bit SOFDE is set, buffer switching is done automatically after SOF (start of frame) has been detected by the USB. **Figure 6-29** describes this functionality for USB read access for this case. The buffer which contains the latest data from the CPU is tagged valid for USB access (UBF=1) at ① and the buffers are swapped if the USB buffer is empty. After the USB read access has occured at ②, this buffer assigned to USB is empty again (UBF=0) and can be swapped again as soon as the CPU has filled its buffer (at ③). The number of bytes in the buffer is less or equal MaxLen. The MaxLen threshold is always active, but an occurrence of SOF (if SOFDE=1) or setting bit DONE by software are used to tag the CPU buffer full before reaching MaxLen.

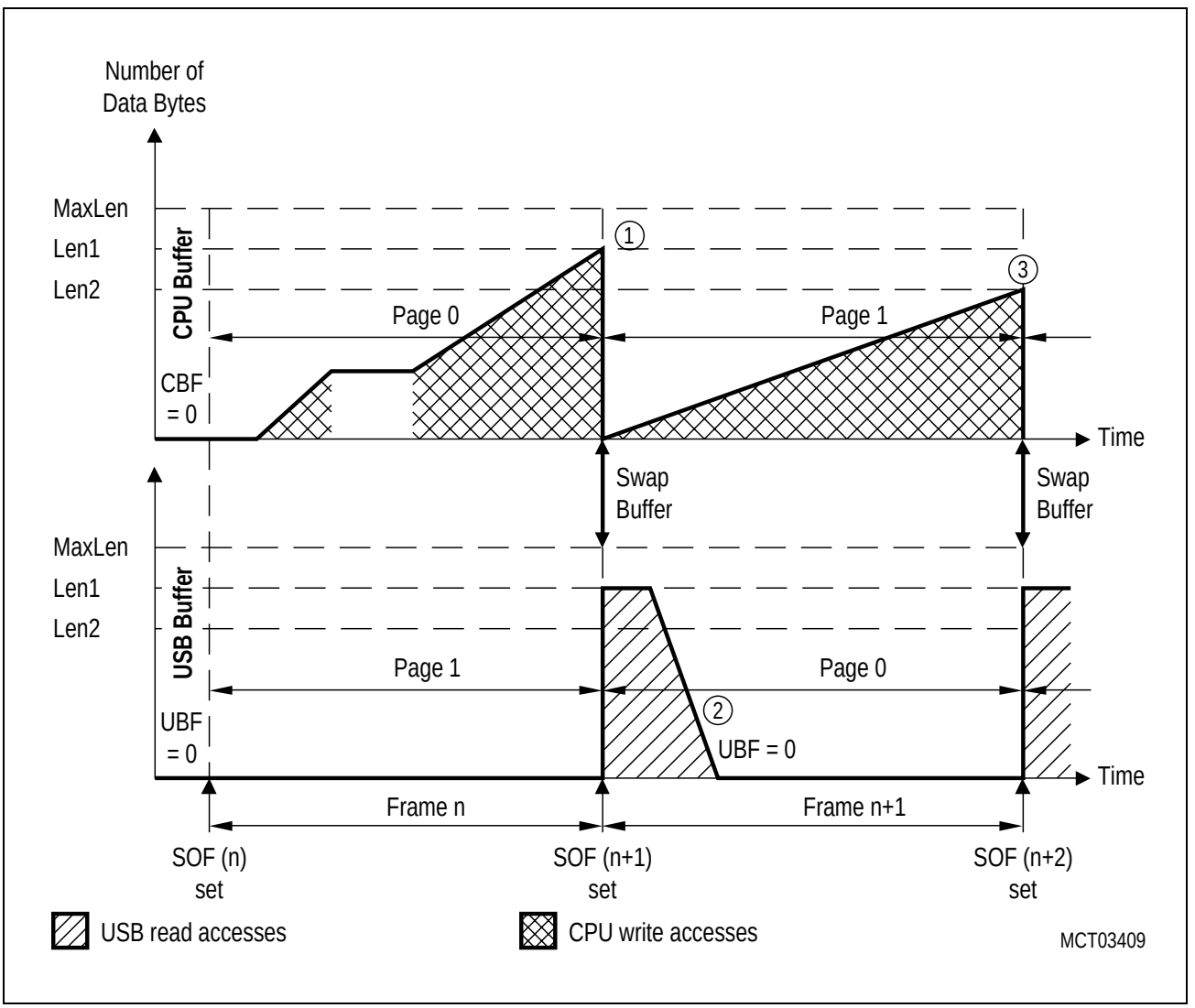


Figure 6-29
Dual Buffer Mode USB Read Access: Buffer Switching on SOF with SOFDE=1

If the number of data bytes to be transferred is greater than the maximum packet size (given by MaxLen), the data is split up automatically into packets, which are transferred one after the other. **Figure 6-30** gives an example of an USB read access, where data from the CPU is split up into two packets. When MaxLen is reached during the CPU write access, the currently active buffer is switched to USB side (UBF=1). The CPU continues writing data to the buffer. When the complete data packet has been written to the buffer by the CPU, bit DONE is set by software to indicate the end of the data packet (CBF=1). In the example, the USB buffer has not been read out. It is still full for the USB and can not be swapped (CBF=UBF=1). When the USB read access has occured (CBF=0), the buffers are automatically swapped and bit SOD is set.

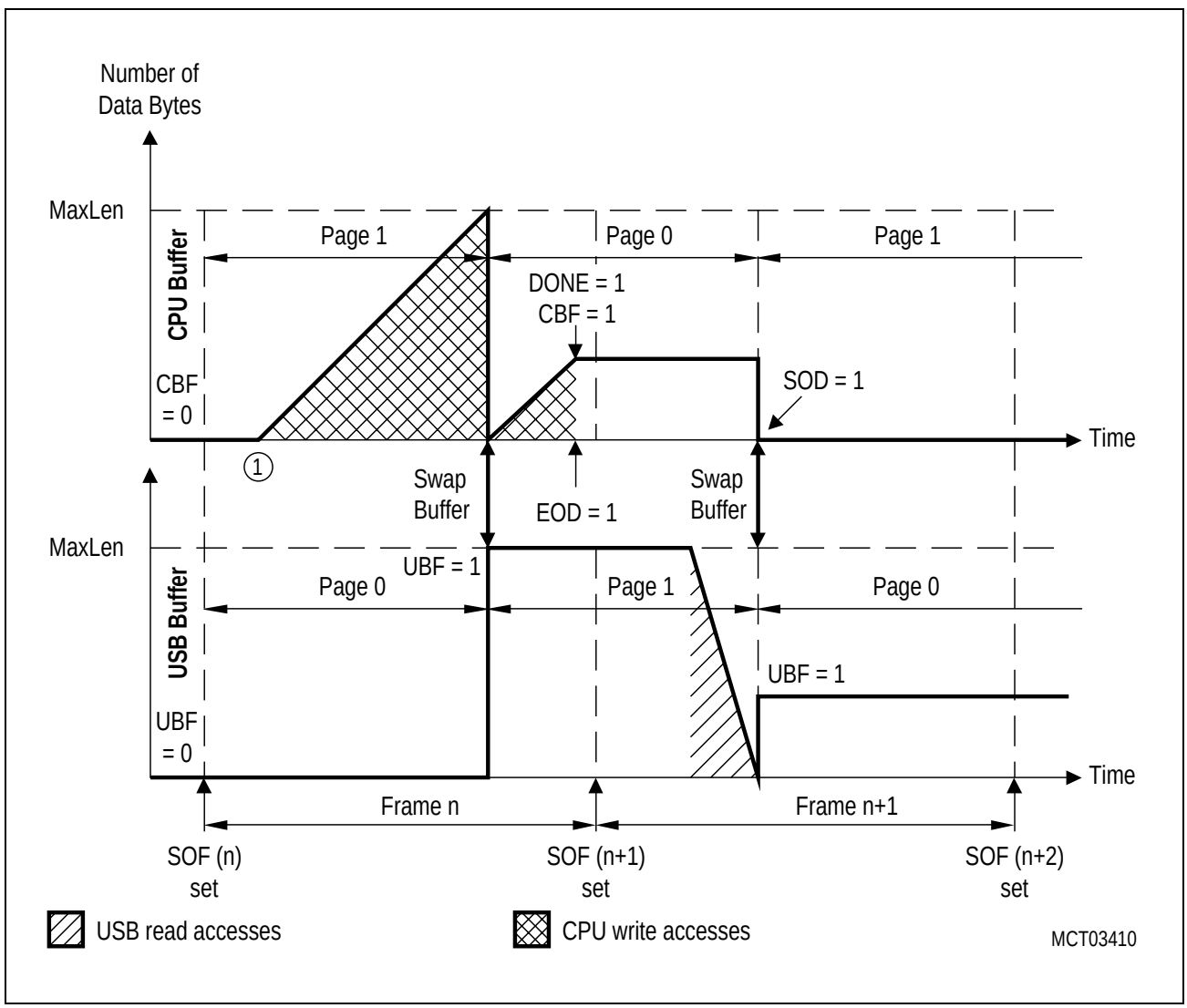


Figure 6-30
Double Buffer Mode USB Read Access: Data Length greater than Packet Length (MaxLen)

In general, three criteria for buffer switching are implemented in the USB module :

a) For sequential access, the address offset register ADROFF is automatically incremented after each read or write action of the CPU. The address offset value (before incrementing) represents the number of bytes stored in USB memory for a specific endpoint. If the address offset value (after incrementing) reaches the value stored in endpoint length register EPLENn, the currently active buffer is tagged full (USB read access - all bytes have been written by CPU, CBF=1) or empty (USB write access - all bytes have been read by CPU, CBF=0).

b) When Bit DONE, which is located in the endpoint buffer status register EPBSn, is set, software buffer switching is initiated. This action is independent from the number of bytes which have been handled by the CPU (possible in sequential access mode (INCE=1) and random access mode (INCE=0)).
On CPU read accesses, the buffer is declared empty and bit CBF is cleared. If the buffer assigned to the USB is full (UBF=1), the buffers are immediately swapped. In this case, register EPLENn contains the number of received bytes.
On CPU write accesses, two different cases must be distinguished. For random accesses, the number of bytes of one packet is fixed by the value in register EPLENn and does not change. For sequential accesses, the number of written bytes represents the packet size. In this case, the actual value of register ADROFF is transferred to register EPLENn when bit DONE is set.

c) The third criteria for buffer switching is the automatic buffer switching on detection of SOF (see **figure 6-30**). This feature can be individually enabled (SOFDE=1) or disabled (SOFDE=0) by software selectively for each endpoint.

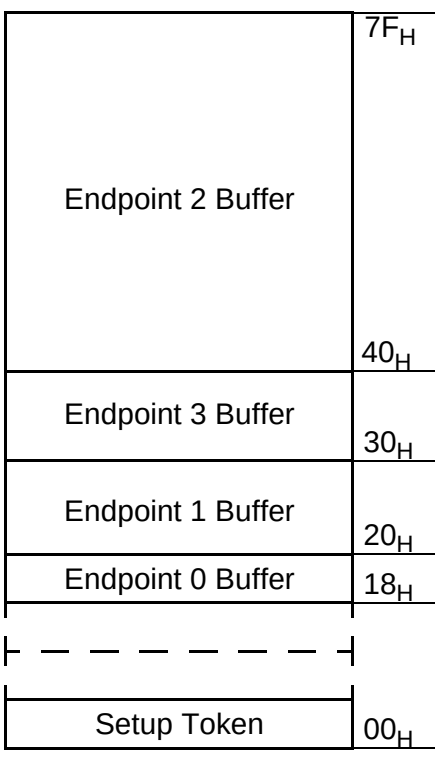
6.4.3 USB Memory Buffer Organization

The address generation of the USB memory buffer is based on address offset and base address pointer. This scheme allows flexible and application specific buffer allocation and management. The length of an endpoint buffer can be up to 8, 16, 32, or 64 bytes. The start address of each endpoint buffer can be located to memory locations according **table 6-5**.

Table 6-5
USB Buffer Length and Base Addresses Values

Buffer Length	Valid Buffer Base Addresses
8 bytes	08_H, 10_H, 18_H, 20_H, 28_H, 30_H, 38_H, 40_H, 48_H, 50_H, 58_H, 60_H, 68_H, 70_H, 78_H
16 bytes	10_H, 20_H, 30_H, 40_H, 50_H, 60_H, 70_H
32 bytes	20_H, 40_H, 60_H
64 bytes	40_H

In order to avoid unused memory space between two endpoint buffers, the largest buffer should be located at the highest address. This structure should be used to allocate USB memory for all endpoint buffers. The base address for the setup packet is always located at address 00_H. This leads to a typical USB buffer structure as shown in **figure 6-31** with a buffer length of 64 bytes for endpoint 2, 16 bytes for endpoints 1 and 3, 8 bytes for endpoint 0, and a predefined length of 8 bytes for endpoint 0 and the setup token.



Buffer Block	EPBAn	EPLENn
Endpoint 2	**EPBA2=08_H**	**EPLEN2=40_H**
Endpoint 3	**EPBA3=06_H**	**EPLEN3=10_H**
Endpoint 1	**EPBA1=04_H**	**EPLEN1=10_H**
Endpoint 0	**EPBA0=03_H**	**EPLEN0=08_H**
Setup Token	Adress 00_H on page **0**	8 Bytes

Figure 6-31
Endpoint Buffer Allocation (Example: 4 Endpoints)

6.4.4 USB Memory Buffer Address Generation

The generation of an USB memory address for USB access (read or write) depends on the EPNum (endpoint number) information, which has been transmitted to the USB module during the software initialization procedure. This EPNum information is used for the selection of an endpoint specific base address register. As the maximum data packet length of each endpoint can individually be programmed, there are some fixed start addresses for the endpoints. The user program defines the base address for the first data byte of the corresponding endpoint by writing the endpoint specific base address register EPBAn. The length depends on the amount of data to be read or written. The user must take care to assign a buffer space at least as large as the maximum packet size of the endpoint.

The address of the currently accessed byte in the USB memory area of the selected endpoint is defined by an address offset, which must be added to the endpoint base address in order to get the correct address for the USB memory buffer. The structure is shown in **figure 6-32**.

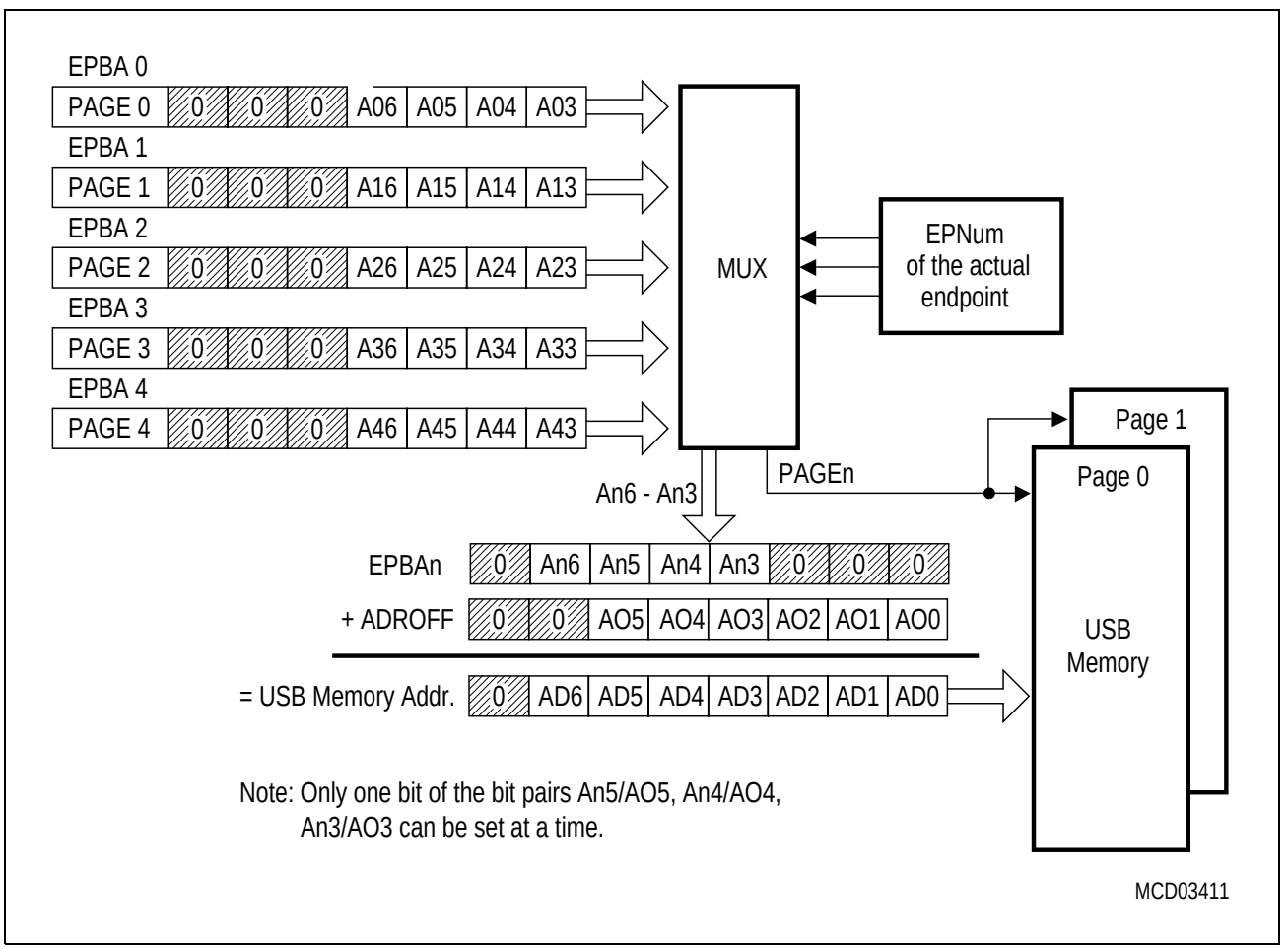


Figure 6-32
USB Memory Address Generation

With the software initialization of the USB module as described in section 6.4.5, each endpoint is initialized with the EPNum value which is used at the USB memory address generation to select the actual endpoint base address register. Further, in single buffer mode the bit PAGEn is used to select one of the USB memory pages In dual buffer mode bit PAGEn has no effect.

6.4.5 Initialization of USB Module

After a hardware reset operation bits PCLK, SPEED, and UCLK are set to 0. If full speed operation is required, a well defined procedure must be executed for switching on the clock for the USB module :

– USB PLL is switched on by setting bit PCLK
waiting 3 ms for PLL being locked
setting bit UCLK

This switch-on procedure after a hardware reset assures a proper operation of the USB clock system. When the USB clock system is switched on, a software initialization procedure must follow This procedure must execute the following steps :

– Setting bit SWR in register DCR starts the software reset operation for the complete USB module. Bit SPEED must be set together with SWR in the same instruction (write protection of SPEED bit).

– When the software reset is finished, bit SWR is cleared by hardware and bit DINIT is set to indicate the start of the initialization sequence.

– The USB module must be functionally initialized from the CPU by writing five configuration bytes for each endpoint to the USBVAL register. Thereafter, bit DONE0 in register EPBS0 must be set by software.

Figure 6-33 shows the 5-byte configuration block which must be transmitted by the CPU to the USB module via the USBVAL register for each endpoint. The gray shaded fields have a fixed “0“ or “1“ value for each endpoint while the white bitfields have to be filled by parameters according **table 6-6**.

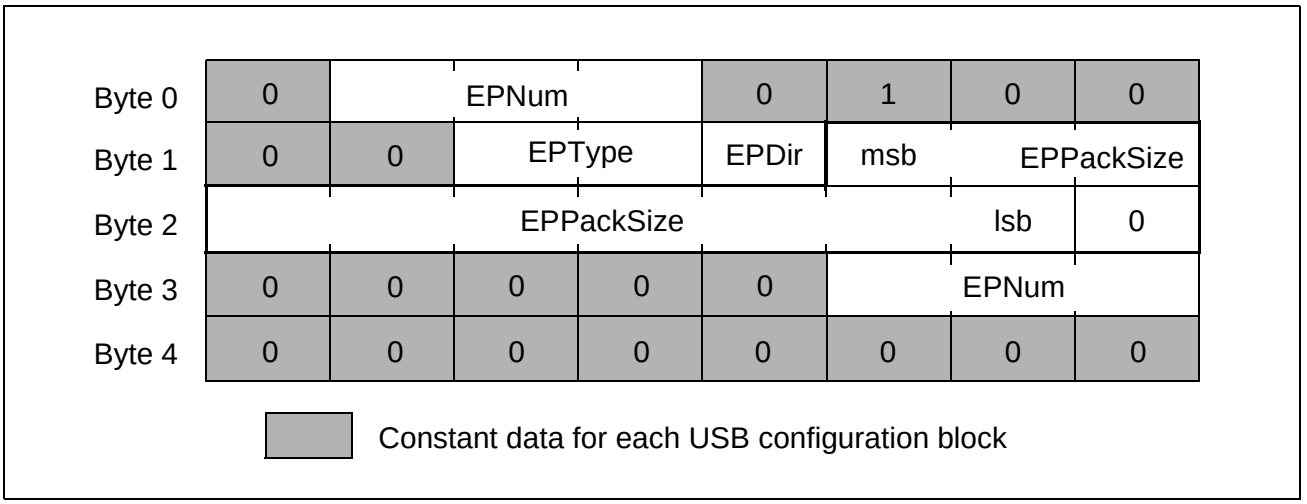


Figure 6-33
USB Configuration Block

The five byte USB configuration block must be transfered sequentially (byte 0 to byte 4) from the CPU to the USB module for each endpoint beginning with endpoint 0, followed by the USB configuration block for endpoint 1 and so on up to the USB configuration block for endpoint 4. EPNum is set to 000_B, 001_B, ... up to 100_B for endpoints 0 up to 4. After this action, bit DINIT is reset by hardware and the software reset and initialization sequence are finished.

Table 6-6
Bitfield Definition of USB Configuration Block

Bitfield	Description
EPNum	This 3-bit field specifies the number of the endpoint (0-4) for which the actual configuration byte block is valid. This 3-bit field must be referenced in byte 0 and byte 3 of a configuration byte block.
EPType	This 2-bit field defines the type of the endpoint 00 : Control endpoint 01 : Isochronous endpoint 10 : Bulk endpoint 11 : Interrupt endpoint Endpoint 0 must be setup for control endpoint-
EPDir	This bit defines the direction of the endpoint 0 : Out (packets to be transferred from Host to CPU) 1 : In (packets to be transferred from CPU to Host)
EPPackSize	This 10-bit field defines the maximum packet size to be transferred to this endpoint within the range from zero up to 1023 bytes. The configuration of EPPackSize has to be in harmony to the USB specification.

6.4.6 Control Transfer

A control transfer consists of at least two and perhaps three stages. This chapter gives a short description of these stages of a control transfer and the associated configuration of the control and status bits.

6.4.6.1 Setup Stage

Control transfer always begin with a setup stage that transfers information to a target device, defining the type of request being made to the USB device. The standard commands except the “set_descriptor“ and “get_descriptor“ command are handled by the USB module automatically without CPU interaction. If the command is not handled by the USB module automatically, a setup interrupt (bit SUI is set) indicates the end of a setup phase. Additionally, the status and control bits UBF, CBF and SOD are reset.

Since C541U only supports single device configuration and single interface, setting multiple device configuration through “set_configuration” and setting alternate interface through “set_interface” to the device will be ignored. In the case that the host sends multiple device configuration and/or alternate interface, the device request value interrupt can be used to capture the events.

6.4.6.2 Data Stage

This stage is defined only for requests that require data transfers. The direction of this data stage is always predicted to be from Host to Device (bit DIR is automatically cleared after the setup stage occured). The first data packet may immediately be send from the Host to the control endpoint according to this configuration of bit DIR, while NACK will be automatically returned from the Device to the Host in case of USB read access.

The configuration of bit DIR=0 predicts an USB write access, while an USB read access causes automatically a NACK (no acknowledge) to be generated and the direction bit to be changed (DIR=1, USB read access).
The direction of the next transfer can also be predicted under software control (bit SETWR is set) to be an USB read access (DIR=1). This feature is used, if the direction of the data stage is known and the data packet to be transferred from the CPU to the Host is set up before the next USB access occured. Therefore, the direction bit must be changed under software control, to be able to transfer the data packet within the first USB read access.

Status bit SOD is set under hardware control to indicate valid data to be read by CPU in case of USB write access, or data to be written by CPU in case of USB read access.

6.4.6.3 Status Stage

The status stage is always performed to report the result of the requested operation. A status stage initiated by the Host, but not terminated according to the configuration of ESP (ESP=0) is indicated by a status interrupt (bit STI is set). Bit ESP has to be set under software control to enable the acknowledge of the status stage.

6.4.7 Register Set

Two different kinds of registers are implemented for full speed operation in the USB module. The global registers (GEPIR, EPSEL, ADROFF, USBVAL) describe the basic functionality of the complete USB module and can be accessed via unique SFR addresses. For reduction of the number of SFR addresses which are needed to control the USB module inside the C541U, device registers and endpoint registers are mapped into an SFR address block of seven SFR addresses ($C1_H$ to $C7_H$). The endpoint specific functionality of the USB module is controlled via the device registers DCR, DPWDR, DIER, DIRR and the frame number registers. An endpoint register set is available for each endpoint (n=0..4) and describes the functionality of the selected endpoint. **Figure 6-34** explains the structure of the USB module registers.

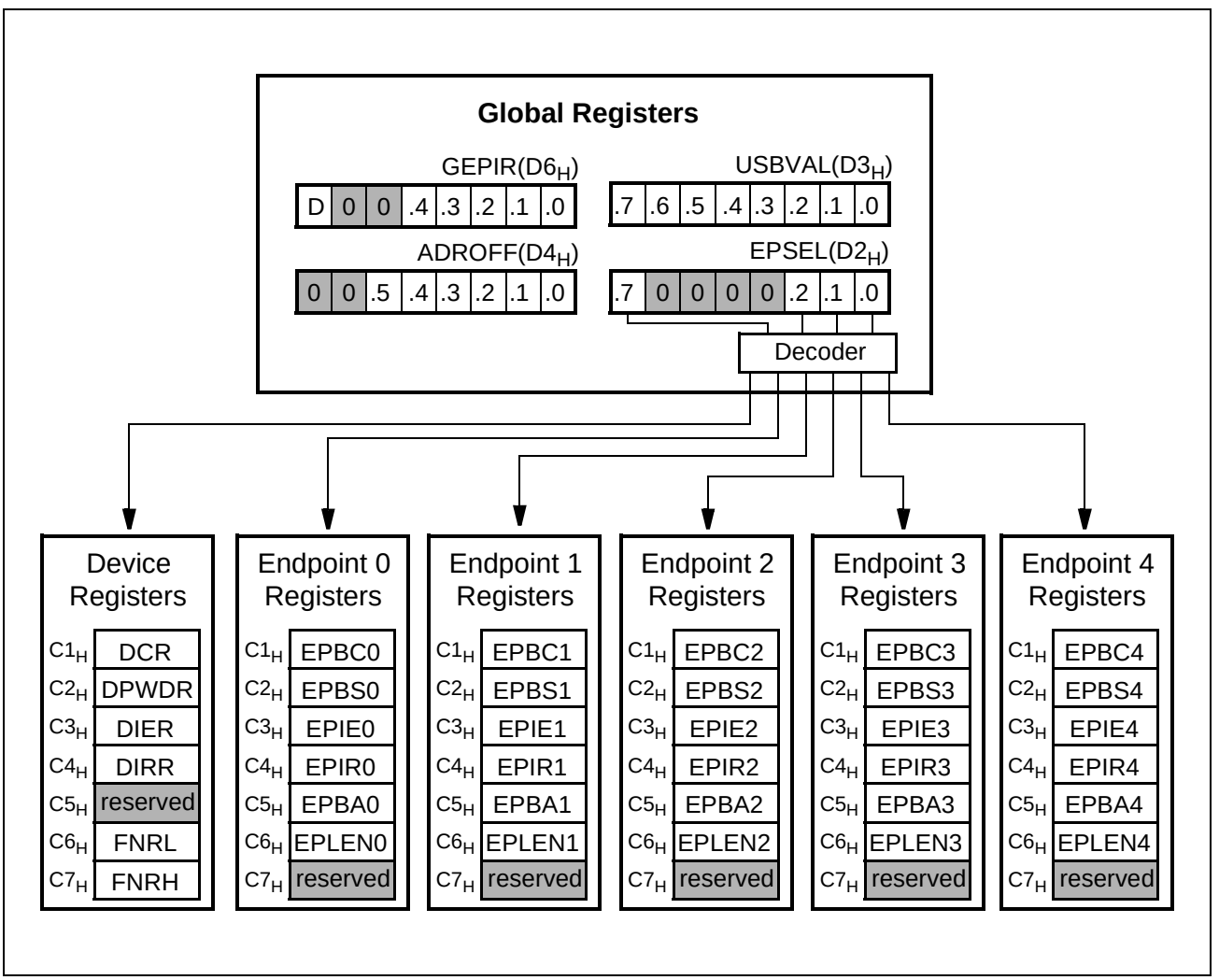


Figure 6-34
Register Structure of the USB Module

Note: In the description of the USB module registers bits are marked as “rw“, “r“, or “w“. Bits marked as “rw“ can be read and written. Bits marked as “r“ can be read only. Writing any value to “r“ bits has no effect. Bits marked as “w“ are used to execute internal commands which are triggered by writing a 1. Writing a 0 to “w“ bits has no effect. Reading “w“ bits returns a 0.

6.4.7.1 Global Registers

The global registers GEPIR, EPSEL, ADROFF, and USBVAL describe the global functionality of the USB module and can be accessed via unique SFR addresses.

The Endpoint Select Register EPSEL contains 4 bits which are used to select one of the register blocks of the five endpoint register blocks or the device register block. These register blocks are mapped to the same SFR address range of $C1_H$ to $C7_H$.

USB Endpoint Select Register EPSEL (Address $D2_H$) **Reset Value : 80_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$D2_H$	EPS7	0	0	0	0	EPS2	EPS1	EPS0	EPSEL
	rw	r	r	r	r	rw	rw	rw	

Bit	Function
EPS7 EPS2 EPS1 EPS0	**Endpoint / device register block select bits** These four bits select the active register block of endpoint or device registers.

EPS7	EPS2	EPS1	EPS0	Selected Register
1	X	X	X	Device register set selected
0	0	0	0	Endpoint 0 register set selected
0	0	0	1	Endpoint 1 register set selected
0	0	1	0	Endpoint 2 register set selected
0	0	1	1	Endpoint 3 register set selected
0	1	0	0	Endpoint 4 register set selected
0	1	0	1	reserved
0	1	1	X	reserved

Table 6-7 shows the register definitions of each endpoint or device register block in detail.

Table 6-7
Endpoint/Device Register Set Address Assignment

EPSEL	SFR Addr.	Selected Register
$1XXXXXXX_B$	$C1_H$	DCR : Device Control Register
	$C2_H$	DPWDR: Device Power Down Register
	$C3_H$	DIER : Device Interrupt Enable Register
(Device register	$C4_H$	DIRR : Device Interrupt Request Register
block selected)	$C5_H$	reserved address
	$C6_H$	FNRL : Frame Number Register (low byte)
	$C7_H$	FNRH : Frame Number Register (high byte)
$0XXXXnnn_B$	$C1_H$	EPBCn : Endpoint n Buffer Control Register (n=0-4)
(nnn =	$C2_H$	EPBSn : Endpoint n Buffer Status Register (n=0-4)
000_B to 100_B)	$C3_H$	EPIEn : Endpoint n Interrupt Enable Register (n=0-4)
	$C4_H$	EPIRn : Endpoint n Interrupt Request Register (n=0-4)
(Endpoint nnn_B	$C5_H$	EPBAn : Endpoint n Base Address Register (n=0-4)
register block	$C6_H$	EPLENn: Endpoint n Buffer Length Register (n=0-4)
selected)	$C7_H$	reserved address

The data transfers between USB memory and the CPU are handled via the SFR USBVAL. With a CPU write access to USBVAL, the value written into it is transferred to the USB memory location which is defined by the content of the endpoint specific endpoint base address register EPBAn and the content of the address offset register ADROFF. At USB memory read accesses from the CPU the data is transfered in reverse direction.
A write operation to USBVAL is only successfull if either DIR=0 and CBF=1 (write operation) or DIR=1 and CBF=0 (read operation).
Sequence of two or more write operations to USBVAL register using move-immediate instruction is unsupported. It is recommended to use either accumulator or a temporary register for storing an immediate data, before writing it to USBVAL register.

USB Data Register USBVAL (Address $D3_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$D3_H$	.7	.6	.5	.4	.3	.2	.1	.0	USBVAL
	rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Function
USBVAL.7-0	**USB data value** USBVAL stores the 8-bit data byte during transfers from CPU to USB memory and from USB memory to CPU. Bit NOD in the EPIRn register indicates when the CPU processes an USBVAL read operation with an empty USB buffer or a USBVAL write operation to a full USB buffer.

In most cases the CPU accesses only one endpoint buffer until it is full (CBF=1 at CPU write access) or empty (CBF=0 at CPU read access). As the USB memory size is 128 bytes per page, the maximum packet length is limited to 64 bytes. Therefore, only the lowest 6 bits of ADROFF (AO5..AO0) are required for offset definiton.
A write operation to ADROFF is only successful if either DIR=0 and CBF=1 (write operation) or DIR=1 and CBF=0 (read operation).

USB Address Offset Register ADROFF (Address $D4_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$D4_H$	0	0	AO5	AO4	AO3	AO2	AO1	AO0	ADROFF
	r	r	rw	rw	rw	rw	rw	rw	

Bit	Function
AO5-0	**USB address offset register** AO5-0 stores the 6-bit offset address for USB memory buffer addressing by the CPU.
0	Reserved for future use. For compatiblity, these bits have to be ignored in all read accesses and written with zero in all write accesses.

After each modification (automatical or by write action) of the address offset register ADROFF, the value pointed to is automatically read out of USB memory and transferred to register USBVAL.

The global endpoint interrupt request register GEPIRn (n=0-4) contaíns one flag for each endpoint which indicates whether one or more of the eight endpoint specific interrupt requests have become active. If a request flag in GEPIR is set, it is automatically cleared after a read operation of the corresponding endpoint specific EPIRn register.

This register contains an additional device interrupt request flag at bit-7. This bit is not automatically cleared by hardware, but has to be cleared by software. The rest of the device interrupt request flags can be found in DIRR register.

USB Global Endpoint Interrupt Request Register GEPIR (Address $D6_H$) Reset Value : 00_H

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$D6_H$	DRVI	0	0	EPI4	EPI3	EPI2	EPI1	EPI0	GEPIR
	rw	r	r	r	r	r	r	r	

Bit	Function
DRVI	**Device request value interrupt** Bit DRVI is set each time the host sends device request that contains one or more of the following : - Configuration Value (through SET_CONFIGURATION device request) - Alternate Setting (through SET_INTERFACE device request) - Interface (through SET_INTERFACE device request) This flag can only be cleared by writing ‘0’ to the bit. Writing ‘1’ to the bit will be ignored. The device interrupt has to be enabled by bit DRVIE in DPWDR (address $E6_H$).
0	Reserved for future use. For compatiblity, these bits have to be ignored in all read accesses and written with zero in all write accesses.
EPI4	**Endpoint 4 interrupt request flag** If EPI4 is set, an endpoint 4 interrupt request is pending.
EPI3	**Endpoint 3 interrupt request flag** If EPI3 is set, an endpoint 3 interrupt request is pending.
EPI2	**Endpoint 2 interrupt request flag** If EPI2 is set, an endpoint 2 interrupt request is pending.
EPI1	**Endpoint 1 interrupt request flag** If EPI1 is set, an endpoint 1 interrupt request is pending.
EPI0	**Endpoint 0 interrupt request flag** If EPI0 is set, an endpoint 0 interrupt request is pending.

Bit 4 (GEPIEn) of the specific endpoint buffer control register EPBCn must be set if EPIn should generate an interrupt. Additionally, bit EA (IEN0.7) and bit EUEI (IEN1.1) must be set when an endpoint interrupt should be triggered.

6.4.7.2 Device Registers

The device registers can only be accessed when global register EPSEL($D2_H$) = 80_H. The following SFRs are defined as device register :

- DCR Device Control Register
- DPWDR Device Power Down Register
- DIER Device Interrupt Enable Register
- DIRR Device Interrupt Request Register
- FNRL Frame Number Register (low byte)
- FNRH Frame Number Register (high byte)

The device control register includes control and status bits which indicate the current status of the USB module and the status of the USB bus.

USB Device Control Register DCR (Address $C1_H$) **Reset Value : $000X0000_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C1_H$	SPEED	DA	SWR	SUSP	DINIT	RSM	UCLK	PCLK	DCR
	rw	r	rw	r	r	rw	rw	rw	

Bit	Function
SPEED	**Low / full speed select** Bit SPEED configures the USB module in the C541U for full speed (12 MBaud) or low speed (1.5 MBaud) mode. This bit can only be written with bit SWR=1 (software reset). After hardware reset the USB module runs in low speed mode and the PLLx4 is automatically disabled. If SPEED=0, low speed mode selected (default after reset) If SPEED=1, full speed mode selected.
DA	**Device attached** Bit DA reflects the state of pin DADD, which can be used to indicate whether the device is attached to the USB bus or not in self-powered mode. If pin DADD is 0, bit DA=0. If pin DADD is 1, bit DA=1.
SWR	**Software reset** Setting bit SWR initiates a software reset operation of the USB device. This bit is cleared by hardware after successful reset operation. SWR can not be reset by software.
SUSP	**Suspend mode** This bit is set when the USB is idle for more than 3 ms. It will remain set until there is a non idle state on the USB cable or when bit RSM is set.

Bit	Function
DINIT	**Device initialization in progress** At the end of a software reset, bit DINIT is set by hardware. After software reset of the USB module, the USB module must be initialized by the CPU. When DINIT is set after a software reset, 5 bytes for each endpoint must be written to SFR USBVAL. After the 25th byte, bit DONE0 has to be set by software. Bit DINIT is reset by hardware after a successful initialization sequence.
RSM	**Resume bus activity** When the USB device is in suspend mode, setting bit RSM resumes bus activity of the device. In response to this action, the USB will disassert the suspend bit and will perform the remote wake-up operation. Writing 0 to RSM has no effect, the bit is reset if bit SUSP is 0.
UCLK	**UDC clock selection** Bit UCLK controls the functionality of the USB core clock in full speed mode (SPEED=1) as well as in low speed mode (SPEED=0).. If UCLK=0, the USB core clock (48 MHz or 6 MHz) is disabled. If UCLK=1, the USB core clock (48 MHz or 6 MHz) is enabled.
PCLK	**PLL clock select** Bit PCLK controls the 48 MHz PLL. If PCLK=0, the 48 MHz PLL is disabled (default after reset). If PLCK=1, the 48 MHz PLL is enabled. For power consumption and EMI reasons, the 48 MHz PLL should be disabled in low speed mode (SPEED=0).

The device power down register DPWDR includes two bits which allow to switch off the USB transmitter and receiver circuitry selectively for power down mode operation.

This register contains an additional device interrupt enable bit. The flag of the interrupt request is stored in GEPIR register.

USB Device Power Down Register DPWDR (Address $C2_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C2_H$	DRVIE	XVREG	0	0	0	0	TPWD	RPWD	DPWDR
	rw	rw	r	r	r	r	rw	rw	

Bit	Function
DRVIE	**Device request value interrupt enable** Setting bit DRVIE enables the generation of a device interrupt each time the host sends device request that contains one or more of the following : - Configuration Value (through SET_CONFIGURATION device request) - Alternate Setting (through SET_INTERFACE device request) - Interface (through SET_INTERFACE device request) If DRVIE=0, the device request value interrupt is disabled. If DRVIE=1, the device request value interrupt is enabled.
XVREG	**External USB voltage regulator** Setting bit XVREG disables the default on-chip voltage regulator. This is recommended if using an external USB voltage regulator is intended. If XVREG=0, the on-chip regulator is enabled. If XVREG=1, the on-chip regulator is disabled.
0	Reserved for future use. For compatiblity, these bits have to be ignored in all read accesses and written with zero in all write accesses.
TPWD	**USB Transmitter Power Down** Setting bit TPWD puts the USB transmitter into power down mode. After a wake-up from software power down mode operation, bit TPWD must be cleared by software to enable again data transmission. If TPWD=0, the transmitter is active (default after reset). If TPWD=1, the transmitter is in power down mode.
RPWD	**USB Receiver Power Down** Setting bit RPWD puts the USB receiver into power down mode. After a wake-up from software power down mode operation, bit RPWD must be cleared by software to enable again data reception. If RPWD is set, the USB bus cannot wake-up the C541U form power down mode. If RPWD=0, the USB receiver is active (default after reset). If RPWD=1, the USB receiver is in power down mode.

The device interrupt enable register DIER contains the enable bits for the different types of device interrupts. With these bits, the device interrupts can be individually enabled or disabled. The general device interrupt enable bit EUDI is located in SFR IEN1. A device interrupt can be only generated if EUDI and EA (global interrupt enable bit in IEN0) are set too.

USB Device Interrupt Enable Register DIER (Address $C3_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C3_H$	SE0IE	DAIE	DDIE	SBIE	SEIE	STIE	SUIE	SOFIE	DIER
	rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Function
SE0IE	**Single ended zero Interrupt Enable** Setting bit SE0IE enables the generation of a device interrupt each time a single ended zero is detected for more than 2.5µs (reset by host, not EOP). If SE0IE=0, the single ended zero interrupt is disabled. If SE0IE=1, the single ended zero interrupt is enabled.
DAIE	**Device attached interrupt enable** Setting bit DAIE enables the generation of a device interrupt when it is attached to the USB bus. If DAIE=0, the device attached interrupt is disabled. If DAIE=1, the device attached interrupt is enabled.
DDIE	**Device detached interrupt enable** Setting bit DDIE enables the generation of a device interrupt when it is detached from the USB bus. If DDIE=0, the device detached interrupt is disabled. If DDIE=1, the device detached interrupt is enabled.
SBIE	**Suspend begin interrupt enable** Setting bit SBIE enables the generation of a device interrupt if bit SBI is set, this means the suspend mode is entered. If SBIE=0, the suspend begin interrupt is disabled. If SBIE=1, the suspend begin interrupt is enabled.
SEIE	**Suspend change interrupt enable** Setting bit SEIE enables the generation of a device interrupt if bit SEI is set, this means the suspend mode is left. If SEIE=0, the suspend change interrupt is disabled. If SEIE=1, the suspend change interrupt is enabled.
STIE	**Status interrupt enable** Bit STIE enables the generation of a device interrupt at the end of the status phase of a control transfer. If STIE=0, the status interrupt is disabled. If STIE=1, the status interrupt is enabled.

Bit	Function
SUIE	**Setup interrupt enable** Bit SUIE enables the generation of a device interrupt on a succesful reception of a setup packet which must be processed by the CPU. If SUIE=0, the setup interrupt is disabled. If SUIE=1, the setup interrupt is enabled.
SOFIE	**Start of frame interrupt enable** bit SOFIE enables the generation of a device interrupt on the detection of a start of frame packet on the USB. If SOFIE=0, the start of frame interrupt is disabled. If SOFIE=1, the start of frame interrupt is enabled.

The device interrupt request register DIRR contains the interrupt request flags of the different types of device interrupts. All Interrupt request flags in DIRR are reset by hardware after DIRR has been read.

USB Device Interrupt Request Register DIRR (Address $C4_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C4_H$	SE0I	DAI	DDI	SBI	SEI	STI	SUI	SOFI	DIRR
	r	r	r	r	r	r	r	r	

Bit	Function
SE0I	**Single ended zero interrupt** Bit SE0I is set each time a single ended zero is detected for equal or greater than 2.5µs. EOP (2 bit times) is not detected.
DAI	**Device attached interrupt** Bit DAI is automatically set after detection of the USB device being attached to the USB bus.
DDI	**Device detached interrupt** Bit DDI is automatically set after detection of the device being detached from the USB bus.
SBI	**Suspend begin interrupt** Bit SBI is automatically set when the suspend mode is entered.
SEI	**Suspend end interrupt** Bit SEI is automatically set when the suspend mode is left.
STI	**Status interrupt** Bit STI is set if the host requests a status transfer and the device answers with NACK (if bit ESP is set, the device answers with ACK and STI is not set).
SUI	**Setup interrupt** Bit SUI is automatically set after a successful reception of a setup packet which is not handled by the USB module and must be forwarded to the CPU. The setup packet itself is limited to 8 bytes and is stored at USB memory adresses 00_H to 07_H and can be accessed with EPSEL.7=1. If a setup interrupt occurs (STI is set), the control and status bits UBF0, CBF0, SOD0, and DIR0 in the endpoint 0 registers EPBS0 and EPIR0 are cleared. DIR0=0 predicts the direction of the next USB access (data phase) to be from host to CPU. Bit DIR0 is automatically set (CPU to host), if the host tries to perform a read access in the first data packet.
SOFI	**Start of frame** Bit SOF is automatically set after detection of a start of frame packet on the USB.

The frame number registers stores an 11-bit value which defines the number of an USB frame. The frame number rolls over upon reaching its maximum value of $7FF_H$. The FNRH/FNRL registers are read only registers which are reset to 00_H by a hardware reset.

Frame Number Register High Byte FNRH (Address $C7_H$) **Reset Value : $00000XXX_B$**
Frame Number Register Low Byte FNRL (Address $C6_H$) **Reset Value : XX_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C7_H$	0	0	0	0	0	FNR10	FNR9	FNR8	FNRH
	r	r	r	r	r	r	r	r	
$C6_H$	FNR7	FNR6	FNR5	FNR4	FNR3	FNR2	FNR1	FNR0	FNRL
	r	r	r	r	r	r	r	r	

Bit	Function
FNR10-0	Frame number value FNRH.2-.0 and FNRL.7-.0 hold the current 11-bit frame number of the latest SOF token. FNRL holds the lowest 8 bits while FNRH holds the upper 3 bits of the frame number.

6.4.7.3 Endpoint Registers

Each of the five endpoints has its own endpoint register set which contains the following registers (n=0-4) :

- EPBCn Endpoint n Buffer Control Register
- EPBSn Endpoint n Buffer Status Register
- EPIERn Endpoint n Interrupt Enable Register
- EPIRRn Endpoint n Interrupt Request Register
- EPBAn Endpoint n Base Address Register
- EPLENn Endpoint n Buffer Length Register

The endpoint buffer control registers control the endpoint specific operations.

Endpoint 1 Buffer Control RegisterEPBCn, n=0-4 (Address $C1_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C1_H$	STALLn	0	0	GEPIEn	SOFDEn	INCEn	0	DBMn	EPBCn
	rw	r	r	rw	rw	rw	r	rw	

Bit	Function
STALLn	**Endpoint stall** Bit STALL can be set to indicate that the endpoint is stalled. If STALL=0, the endpoint n is active. If STALL=1, the endpoint n is stalled. Note : If the stall bit for endpoint 0 (STALL0) is set, the next incoming setup token will automatically clear it.
GEPIEn	**Global endpoint interrupt enable** Bit GEPIEn enables or disables the generation of the global endpoint interrupt n based on the endpoint specific interrupt request bits in register EPIRn. If GEPIE=0, the USB endpoint n interrupt is disabled. If GEPIE=1, the USB endpoint n interrupt is enabled.
SOFDEn	**Start of frame done enable** If bit SOFDE is set, the current CPU buffer in USB memory is automatically tagged full (data flow from the CPU to the USB) or empty (data flow from the USB to the CPU) on each detection of a start of frame on the USB (auto-done). If SOFDE=0, no action on SOF. If SOFDE=1, the automatical generation of DONE on SOF is enabled.
0	Reserved for future use. For compatiblity, these bits have to be ignored in all read accesses and written with zero in all write accesses.

Bit	Function
INCEn	**Auto increment enable** If bit INCE is set, the address offset register ADROFF for CPU access to USB memory is automatically incremented after each data write or data read action of the USBVAL register. This allows the user to handle the USB memory like a FIFO without modification of the address of the desired memory location by software. After each modification of ADROFF the data value pointed to is automatically read out of USB memory and transferred to the USBVAL register. If INCE=0, the auto-increment function is disabled. If INCE=1, the auto-increment function is enabled.
DBMn	**Dual buffer mode** Bit DBM allows the selection between single buffer mode and dual buffer mode. If DBM=0, single buffer mode is selected. If DBM=1, dual buffer mode is selected.

Note: For accessing the EPBCn registers SFR EPSEL($D2_H$) must be set with the appropriate value.

The bits of the endpoint buffer status registers indicate the status of the endpoint specific USB memory buffers and allows setting of certain USB memory buffer conditions.

Endpoint Buffer Status Register EPBSn, n=0-4 (Address $C2_H$) **Reset Value : 20_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C2_H$	UBFn	CBFn	DIRn	ESPn	SETRDn	SETWRn	CLREPn	DONEn	EPBSn
	r	r	r	w	w	w	w	w	

Bit	Function
UBFn	**USB buffer full** Bit UBFn indicates the status of the USB memory buffer for endpoint n. USB read access: If UBFn=0, the USB buffer for endpoint n is empty. If UBFn=1, the USB buffer for endpoint n is not empty. USB write access : If UBFn=0, the USB buffer for endpoint n is not full. If UBFn=1, the USB buffer for endpoint n is full.
CBFn	**CPU buffer full** Bit CBFn indicates the status of the CPU memory buffer for endpoint n. CPU read access: If CBFn=0, the CPU buffer for endpoint n is empty. If CBFn=1, the CPU buffer for endpoint n is not empty. CPU write access: If CBFn=0, the CPU buffer for endpoint n is not full. If CBFn=1, the CPU buffer for endpoint n is full.
DIRn	**Direction of USB memory access** Bit DIRn indicates the direction of the last USB memory access for endpoint n. If DIRn=0, the last data flow for endpoint n was from host to CPU. If DIRn=1, the last data flow for endpoint n was from CPU to host.
ESPn	**Enable status phase** If bit ESPn is set, the next status phase of endpoint n will automatically be acknowledged by an ACK except the endpoint n is stalled. If the status phase was successfully completed, bit ESPn is automatically reset by hardware and no status interrupt request (STI) is generated. If the CPU detects a corrupted control transfer (endpoint 0), bit STALL0 should be set by software instead of bit ESP0 in order to indicate an error condition which cannot be recovered by the USB device itself. Note : bit EPSn can only be set by software. Any read operation of register EPBSn returns ESPn=0.
SETRDn	**Set direction of USB memory buffer to read** Bit SETRDn is used to predict the direction of the next USB access for endpoint n as an USB read access. A faulty prediction causes no errors since the USB module determines the real direction. A change in the data direction is only executed, if both USB memory buffers are empty. SETRD cannot be set together with CLREPn because a change of bit DIRn during a transfer is not allowed. Note : bits SETRDn and SETWRn must not be set at a time.

Bit	Function
SETWRn	**Set direction of USB memory buffer to write** Bit SETWRn is used to predict the direction of the next USB access for endpoint n as an USB write access. A faulty prediction causes no errors since the USB module determines the real direction. A change in the data direction is only executed, if both USB memory buffers are empty. SETWR cannot be set together with CLREPn because a change of bit DIRn during a transfer is not allowed. Note : bits SETWRn and SETRDn must not be set at a time.
CLREPn	**Clear endpoint** Setting bit CLREPn will set the address offset register for a CPU access to USB memory to 0. The bits CBFn and UBFn will be reset when CLREPn is set. Bit CLREPn is reset by hardware. A read operation on this bit will always deliver 0. Setting of CLREPn does not change the direction of endpoint n. This means, bit DIRn is not changed. Note: When bits CLREPn and ESPn are set simultaneously with one instruction, bit ESPn remains set and the next status phase is enabled. If only CLREPn is set, bit ESPn is reset and the status phase is disabled. Setting bits CLREPn and SETRDn or SETWRn simultaneously with one instruction is not allowed. This means that the information of SETRDn or SETWRn is ignored in this case.
DONEn	**Buffer done by CPU** If bit DONE is set, the current USB memory buffer assigned to CPU is automatically tagged full (data flow from the CPU to the USB) or empty (data flow from the USB to the CPU). This bit is reset by hardware after it has been set. A read operation on this bit will deliver always 0. Note: If the direction of endpoint n is read (USB read access) and auto-increment is enabled (INCEn=1) and DONEn is set the content of register ADROFF is copied automatically to register EPLENn of the actual endpoint. Register EPLENn is not changed if the auto-increment capability is disabled (INCEn=0).

The endpoint interrupt enable registers contain the endpoint specific interrupt enable bits. With these bits, the endpoint specific interrupts can be individually enabled or disabled. Additionally to a bit in an EPIEn register, the global interrupt bit EPIn in GEPIR for endpoint n and the general endpoint interrupt bit EUEI in IEN1 and the general interrupt enable bit EA in IEN0 must be set for the interrupt becoming active.

Endpoint Interrupt Enable Register EPIEn, n=0-4 (Address $C3_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C3_H$	AIEn	NAIEn	RLEIEn	0	DNRIEn	NODIEn	EODIEn	SODIEn	EPIEn
	rw	rw	rw	rw	rw	rw	rw	rw	

For accessing EPIEn, SFR EPSEL must be $0n_H$.

Bit	Function
AIEn	**USB acknowledge interrupt enable** Bit AIEn enables the generation of an endpoint specific acknowledge interrupt when bit ACKn in register EPIRn is set. If AIEn=0, the USB acknowledge interrupt is disabled. If AIEn=1, the USB acknowledge interrupt is enabled.
NAIEn	**USB not acknowledged interrupt enable** Bit NAIEn enables the generation of an endpoint specific not acknowledged interrupt when bit NACKn in register EPIRn is set. If NAIEn=0, the USB not acknowledged interrupt is disabled. If NAIEn=1, the USB not acknowledged interrupt is enabled.
RLEIEn	**Read length error interrupt enable** Bit RLEIEn enables the generation of an endpoint specific read length error interrupt when bit RLEn in register EPIRn is set. If RLEIEn=0, the read length error interrupt is disabled. If RLEIEn=1, the read length error interrupt is enabled.
0	Reserved for future use. For compatiblity, these bits have to be ignored in all read accesses and written with zero in all write accesses.
DNRIEn	**Data not ready interrupt enable** Bit DNRIEn enables the generation of an endpoint specific data not ready interrupt when bit DNRn in register EPIRn is set. If DNRIEn=0, the data not ready interrupt is disabled. If DNRIEn=1, the data not ready interrupt is enabled.
NODIEn	**No data interrupt enable** Bit NODIEn enables the generation of an endpoint specific no data interrupt when bit NODn in register EPIRn is set. If NODIEn=0, the no data interrupt is disabled. If NODIEn=1, the no data interrupt is enabled.

Bit	Function
EODIEn	**End of data interrupt enable** Bit EODIEn enables the generation of an endpoint specific end of data interrupt when bit EODn in register EPIRn is set. If EODIEn=0, the end of data interrupt is disabled. If EODIEn=1, the end of data interrupt is enabled.
SODIEn	**Start of data interrupt enable** Bit SODIEn enables the generation of an endpoint specific start of data interrupt when bit SODn in register EPIRn is set. If SODIEn=0, the start of data interrupt is disabled. If SODIEn=1, the start of data interrupt is enabled.

The endpoint interrupt request register EPIRn contains the interrupt request flags of the different endpoint specific interrupts. In general, the bits in EPIRn are reset by hardware after a EPIRn read operation.

Endpoint Interrupt Request Register EPIRn, n=0-4 (Address $C4_H$) Reset Value EPIR0 : 11_H
Reset Value EPIR1 to EPIR4 : 10_H

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C4_H$	ACKn	NACKn	RLEn	–	DNRn	NODn	EODn	SODn	EPIRn
	r	r	r	r	r	r	r	r	

For accessing EPIRn, SFR EPSEL must be $0n_H$.

Bit	Function
ACKn	**USB acknowledge** Bit ACKn=1 indicates a succesful action on the USB.
NACKn	**USB not acknowledged** Bit NACKn is set for all unsuccessful actions on the USB.
RLEn	**Read length error** Bit RLEn is automatically set if the number of bytes read by the USB does not correspond to the packet length programmed by the CPU.
–	Reserved bit for future use. As this bit may return nonzero value, it must be ignored in all read accesses.
DNRn	**Data not ready** This bit is set by hardware if the UDC requires an access to USB memory, but no buffer is available. USB Read action : DNRn is set if UBF is not set. USB Write action : DNRn is set if UBF is set.
NODn	**No data** This bit indicates an incorrect CPU read or write access to USB memory. It is set if the CPU processes a read access to an empty USB buffer or a write access to a full buffer. NODn is also set if the direction is write (DIRn=0 for USB write access) and the CPU tries to write to the USB memory buffer.
EODn	**End of data** USB Read action : EODn is set if the CPU has written a programmable number (MaxLen) of bytes in the transmit buffer. As a result, the buffer is full and no more write actions from the CPU are allowed. USB Write action: EODn is set if the CPU has read a programmable number (USBLen) of bytes out of the receive buffer. As a result, the buffer is empty now and no more read actions from the CPU are allowed.

Bit	Function
SODn	**Start of data** USB Read action: SODn is set if the USB has read a fixed number (USBLen) of bytes from the transmit buffer. As a result, the buffer is empty now and the CPU can process write actions again. USB Write action: SODn is set if the USB has written a fixed number (USBLen) of bytes to the receive buffer. As a result, the buffer is full and the CPU can start read actions..

In dual buffer mode bits SODn and EODn can be set simultaneously if the corresponding buffer page is swapped.

The endpoint base address register defines the location and size (start address and length) of the endpoint specific buffers in the USB memory. See also **figure 6-31** for an example of EPBAn and EPLENn register setup.

Endpoint n Base Address Register EPBAn, n=0-4 (Address $C5_H$) **Reset Value : 00_H**
Endpoint n Buffer Length Register EPLENn, n=0-4 (Address $C6_H$) **Reset Val.: $0XXXXXX_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C5_H$	PAGEn	0	0	0	An6	An5	An4	An3	EPBAn
	r	r	r	r	rw	rw	rw	rw	
$C6_H$	0	Ln6	Ln5	Ln4	Ln3	Ln2	Ln1	Ln0	EPLENn
	r	rw	rw	rw	rw	rw	rw	rw	

Bit	Function
PAGEn	**Buffer page for endpoint n (single buffer mode only)** In single buffer mode, the endpoint n can be either located on USB memory buffer page 0 (PAGEn=0) or on USB memory buffer page 1 (PAGEn=1) by clearing or setting this bit. In dual buffer mode this bit has no effect. Note: The SETUP token is always stored on USB memory buffer page 0 at address 00_H to 07_H.
An6-An3	**Endpoint n buffer start address** The bits 0 to 3 of EPBAn are the address bits A6 to A3 of the USB memory buffer start address for endpoint n. A7 and A2-A0 of the resulting USB memory buffer start address are set to 0.
Ln6-Ln0	**Endpoint n buffer length** The bits 0 to 6 if EPLENn define the length of the USB memory buffer for endpoint n and cannot be written if DINIT=1.
0	Reserved for future use. For compatiblity, these bits have to be ignored in all read accesses and written with zero in all write accesses.

The USB buffer allocation and organization is described in detail in section 6.4.3.

6.4.8 Low Speed Mode

The features of this mode are listed below :

- Data rate of 1.5 Mbit/s
- Three endpoints : one bidirectional control endpoint, EP0
  two interrupt endpoints, EP1 and EP2
- Maximum data packet length of 8 bytes, fully direct accesible data packet
- Two supported transfer modes : control transfer, through EP0
  interrupt transfer, interval from 10 ms to 255 ms

The low speed operation require a different set of registers than the full speed operation. These registers are described in section 6.4.8.5.

6.4.8.1 Initialization

After a successful hardware reset, the device is by default in USB low speed mode. The initialization phase described in section 6.4.5 is required only for full speed mode.

6.4.8.2 Transfer Modes

According to the USB Specification Rev1.0 the low speed mode is only supported in control transfer mode and interrupt transfer mode. The two supported transfer modes are described here.

There are times when the hardware is busy updating the registers and accesses to USBDRn registers are blocked. Software has to wait until the TYPE bit field of USBDCR register is set to 0000_B or '**Empty**'.

6.4.8.3 Control Transfer

Control transfer is always directed to/from the control endpoint, EP0. This type of transfer consists of two or three stages. At the end of each stage an acknowledgement needs to be generated by writing 0000_B or '**Empty**' to the TYPE bit field.

6.4.8.3.1 Setup Stage

The host always initiates this stage by sending a SETUP packet to the device. SETUP interrupt request will be generated and the TYPE bit field of USBDCR contains 0010_B or '**SETUP packet**'. The data packet associated with this stage will be stored in the USBDRn registers, while the packet length is recorded in the LEN bit field of USBDCR register.

Software is then required to read the registers in order to detect the transfer. The SETUP interrupt request will be cleared by reading the USBDCR register. The stage is completed by writing '**Empty**' to the TYPE bit field.

Since C541U only supports single device configuration and single interface, setting multiple device configuration through "set_configuration" and setting alternate interface through "set_interface" to the device will be ignored.

6.4.8.3.2 Data Stage

This stage is optional and defined only for requests that require data transfer. The direction of the transfer is either from the host to the device (OUT packets) or from the device to the host (IN packets). When an OUT packet is received from the host, the OUT interrupt will be generated, the data stored in the USBDRn registers and the length recorded in the LEN bit field. Consequently the TYPE bit field will be set to 0011_B or '**OUT packet**'. Software is again required to read the registers and set TYPE to '**Empty**' at the end of this stage. The OUT interrupt is cleared by the reading of USBDCR register.

The device can send an IN packet to the host. In this case software has to write the data packet in the USBDRn registers, set the LEN bit field to the data length and set TYPE to 0101_B or '**IN packet**'. This will trigger the transmission of the packet over USB.

6.4.8.3.3 Status Stage

This stage is always performed to report the result of the requested operation. The direction is either from the host to the device (OUT packets) or from the device to the host (IN packets). The data length for this stage is always zero and the direction is always opposite to the direction of the previous data stage (if any), or setup stage (if no data stage).

6.4.8.4 Interrupt Transfer

The direction of this transfer is always from the device to the host. The usage is intended for device polling with programmable rate from 10 ms to 255 ms. A no acknowledge handshake (NACK) is automatically returned to the host, if the device has currently no interrupt data to transfer.

Software is required to set up the data packet in the USBDRn registers, to set the LEN bit field according to the data length and to set the TYPE bit field to either 0110_B (for EP1 interrupt endpoint) or 0111_B (for EP2 interrupt endpoint).

6.4.8.5 Register Set

A set of control and data registers are defined for the USB module in low speed operation.

USB Module Control Register USBDCR (Address $E7_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$E7_H$	TYPE3	TYPE2	TYPE1	TYPE0	LEN3	LEN2	LEN1	LEN0	USBDCR
	rw	rw	rw	rw	rw	rw	rw	rw	

Bit	Function
TYPE3 - 0	**Transfer Type**

TYPE3 - 0	Description
0000	**Empty** Software is required to write this value to generate an acknowledgement and trigger the next activity in the hardware.
0001	**Reserved**
0010	**SETUP packet** A SETUP packet has been received and the setup data is stored in the USBDRn registers and SETUP interrupt request is also generated. A read operation to register USBDCR resets the pending interrupt request, while TYPE bit field maintains its current value (0010_B).
0011	**OUT packet** An OUT packet has been received by endpoint zero and is stored in the USBDRn registers and OUT interupt request is generated. A read operation to register USBDCR resets the pending interrupt request, while TYPE bit field maintains its current value (0011_B).
0100	**Stall** The module is stalled including control endpoint and both interrupt endpoints.
0101	**IN packet** An IN packet has been set up for endpoint zero, which is directed to the host. Software is required to write this value to trigger the data packet to be sent to the host.

Bit	Function
TYPE3 - 0	**Transfer Type**

TYPE3 - 0	Description
0110	**Interrupt 1** Software is required to write this value for the host to read the data associated with EP1interrupt endpoint from the USBDRn registers.
0111	**Interrupt 2** Software is required to write this value for the host to read the data associated with EP2 interrupt endpoint from the USBDRn registers.
1000	**Suspend** The bus is in suspended mode, but the suspend interrupt request has been cleared by previous reading of this register.
1001	**Reset** The device has received a reset signal, but the reset interrupt request has been cleared by previous reading of this register.
1010	**Suspend interrupt pending** The bus is in suspended mode and the suspend interrupt is pending. Current reading of this register clears the interrupt request.
1011	**Reset interrupt pending** The device has received a reset signal and the reset interrupt is pending. Current reading of this register clears the interrupt request.
1100	**DADD falling** A falling edge was detected on DADD pin and the DADD interrupt has been cleared by previous reading of this register.
1101	**DADD rising** A rising edge was detected on DADD pin and the DADD interrupt has been cleared by previous reading of this register.
1110	**DADD falling interrupt pending** A falling edge has been detected on DADD pin and the DADD interrupt is pending. Current reading of this register clears the interrupt request.
1111	**DADD rising interrupt pending** A rising edge has been detected on DADD pin and the DADD interrupt is pending. Current reading of this register clears the interrupt request.

Bit	Function
LEN3 - 0	**Length Information** contains the number of bytes currently stored in the USB data registers. LEN is set to 1111_B when the hardware accesing the USBDRn registers. Bitfield LEN is updated either by USB or by CPU. **CPU:** The number of bytes (packet size) has to be written to bitfield LEN at the end of a CPU write access to the USB data registers. Note that setting TYPE to EMPTY after a CPU read access, will causes bitfield LEN to be reset. **USB:** The number of received bytes (the size of the received packet) is written to bitfield LEN after an USB write accesss, while LEN is reset after the data packet was read by the USB.

Move immediate data to USBDCR registers are unsupported. An example of such instructions is as the following :

```
MOV     USBDCR,#12H     ;The data 12H is just an example and can be replaced
                        ;       with any 8-bit immediate data.
```

It is recommended to use a temporary register or an accumulator to write to these registers. Examples of using accumulator and register are as the following :

```
MOV     A,#12H          ;Move immediate data to accumulator
MOV     USBDCR,A        ;Move the content of accumulator into USBDCR register

MOV     R2,#12H         ;Move immediate data to R2
MOV     USBDCR,R2       ;Move the content of R2 into USBDCR register
```

USB Module Power Down Register USBPWD (Address $E6_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$E6_H$	0	0	SUSPIE	DADDIE	SUSP	DADD	TPWD	RPWD	USBPWD
	r	r	rw	rw	r	r	rw	rw	

Bit	Function
SUSPIE	**Suspend Interrupt Enable/Disable bit** 0: Suspend interrupt generation is disabled. 1: Suspend interrupt generation is enabled.
DADDIE	**Device Attach/Detach Interrupt Enable/Disable bit** 0: Interrupt generation upon DADD/P3.1 pin status is disabled. 1: Interrupt generation upon DADD/P3.1 pin status is enabled.
SUSP	**Suspend mode** This bit is set when the USB has been idle for more than 3 ms. It will remain set until there is a non-idle state on the USB cable.
DADD	**Device attached or detached** Bit DADD monitors the state of the associated I/O pin DADD (see also TYPE bit field of USBDCR).
TPWD	**Transmitter Power Down** 0: Transmitter is active. 1: Transmitter is in power down state.
RPWD	**Receiver Power Down** 0: Receiver is active. 1: Receiver is in power down state.
0	Reserved for future use. For compatiblity, these bits have to be ignored in all read accesses and written with zero in all write accesses.

USB Module Data Register USBDRn, n=0..7 (Address $E8_H$ - EF_H)

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$E8_H$	.7	.6	.5	.4	.3	.2	.1	.0	USBDR0
	rw	rw	rw	rw	rw	rw	rw	rw	
$E9_H$	.7	.6	.5	.4	.3	.2	.1	.0	USBDR1
	rw	rw	rw	rw	rw	rw	rw	rw	
EA_H	.7	.6	.5	.4	.3	.2	.1	.0	USBDR2
	rw	rw	rw	rw	rw	rw	rw	rw	
EB_H	.7	.6	.5	.4	.3	.2	.1	.0	USBDR3
	rw	rw	rw	rw	rw	rw	rw	rw	
EC_H	.7	.6	.5	.4	.3	.2	.1	.0	USBDR4
	rw	rw	rw	rw	rw	rw	rw	rw	
ED_H	.7	.6	.5	.4	.3	.2	.1	.0	USBDR5
	rw	rw	rw	rw	rw	rw	rw	rw	
EE_H	.7	.6	.5	.4	.3	.2	.1	.0	USBDR6
	rw	rw	rw	rw	rw	rw	rw	rw	
EF_H	.7	.6	.5	.4	.3	.2	.1	.0	USBDR7
	rw	rw	rw	rw	rw	rw	rw	rw	

It is recommended to arrange the data by writing USBDR0 with the first byte. For the rest of the data, put the last byte in USBDR1 and the second last byte (if any) in USBDR2, the third last byte (if any) in USBDR3 and so on, until the second byte. So if the length of data is *n*, i.e. data[0], data[1], ... data[*n*-2], data[*n*-1], then the data should be arranged in the following manner:

USBDR0 <- data[0]
USBDR1 <- data[*n*-1]
USBDR2 <- data[*n*-2]
...
USBDR*n*-1 <- data[1]

6.4.8.6 USB Low Speed Interrupts

Each of the USB Low Speed specific interrupt share the vector address of $004B_H$. The general interrupt enable EUEI in IEN1 register has to be enabled, before using these interrupts. The interrupts are cleared by reading USBDCR register. The following table lists the USB Low Speed specific interrupts.

Table: USB Low Speed Interrupt Source and Vectors

Interrupt Request	Interrupt Enable Flags
SETUP packet interrupt	always enabled
OUT packet interrupt	always enabled
USB Suspend interrupt	SUSPIE
USB Reset interrupt	always enabled
DADD interrupt	DADDIE

6.4.9 On-Chip USB Transceiver

The C541U provides on-chip receiver and transmitter circuitries which allows to connect the C541U directly to the USB bus. The USB driver circuitry is shown in **figure 6-35**. The USB transceiver is capable of transmitting and receiving serial date at full speed (12 MBits/s) and low speed (1.5 Mbit/s) data rates. Transceiver and receiver can be separately disabled for power down mode operation. A single ended zero error condition (D+ and D- both at low level) can be detected.

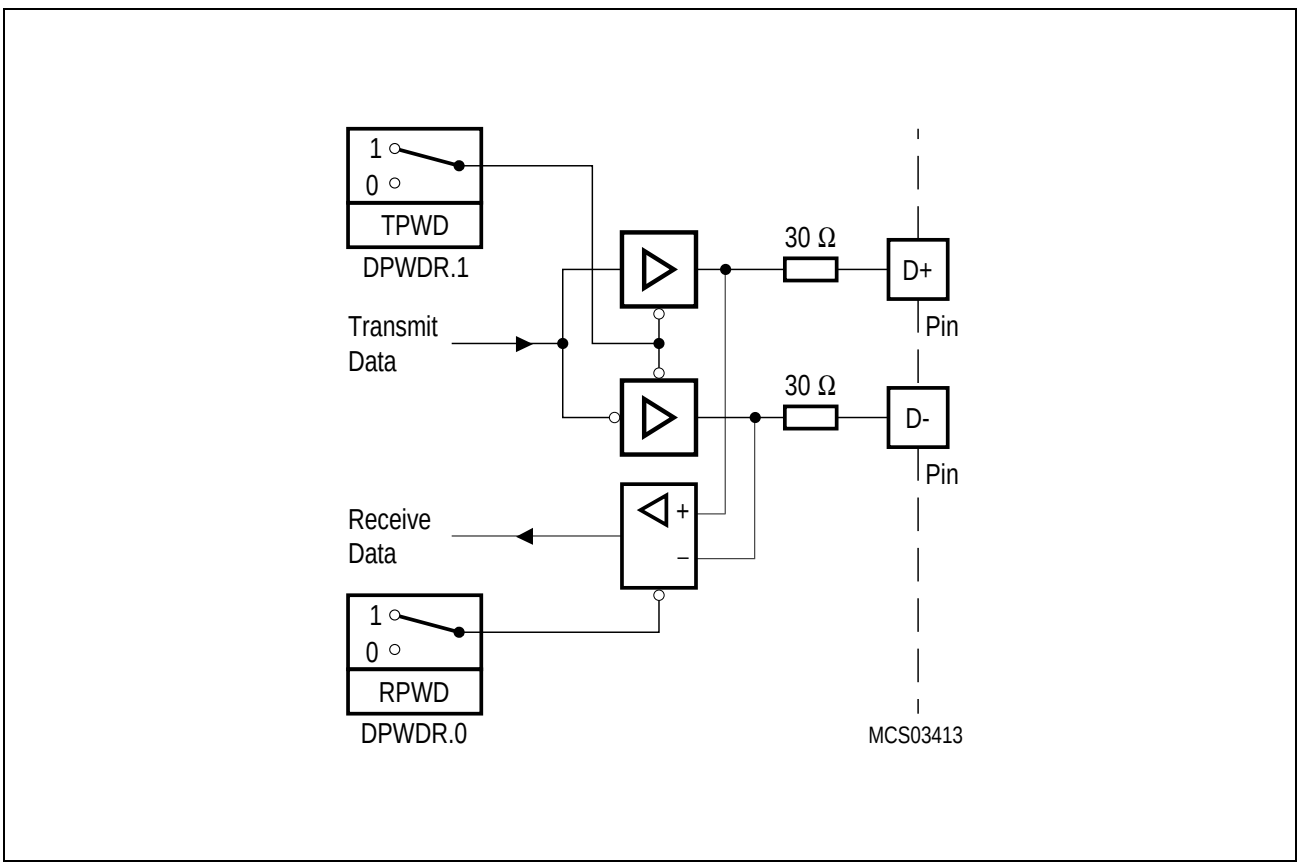


Figure 6-35
USB On-chip Driver Circuitry

The USB driver circuitry is a differential output driver which drives the USB data signal onto the cable of the USB bus. The static output swing of the transmitter is in low state below 0.3 V with a 1.5 kΩ load to 3.6 V and in high state above 2.8 V with a 15 kΩ load to V_{SS}. The driver outputs support tri-state operation to achieve bi-directional half duplex operation (control bits RPWD and TPWD). High impedance is also required to isolate the port from devices that are connected but powered down. The driver tolerates a voltage on the signal pins of -0.5 V to 3.8 V with respect to local ground reference without damage. This voltage is tolerated for 10.0 µs while the driver is active and driving, When the driver is in its high impedance state it tolerates this condition an indefinitely long time.

For a full speed USB connection the impedance of the USB driver must be between 29 Ω and 44 Ω. The data line rise and fall times are between 4 ns and 20 ns, smoothly rising or falling (monotonic), and are well matched to minimize RFI emissions and signal skew. **Figure 6-35** shows how the full speed driver is realized using two identical CMOS buffers. **Figure 6-36** shows the full speed driver signal waveforms.

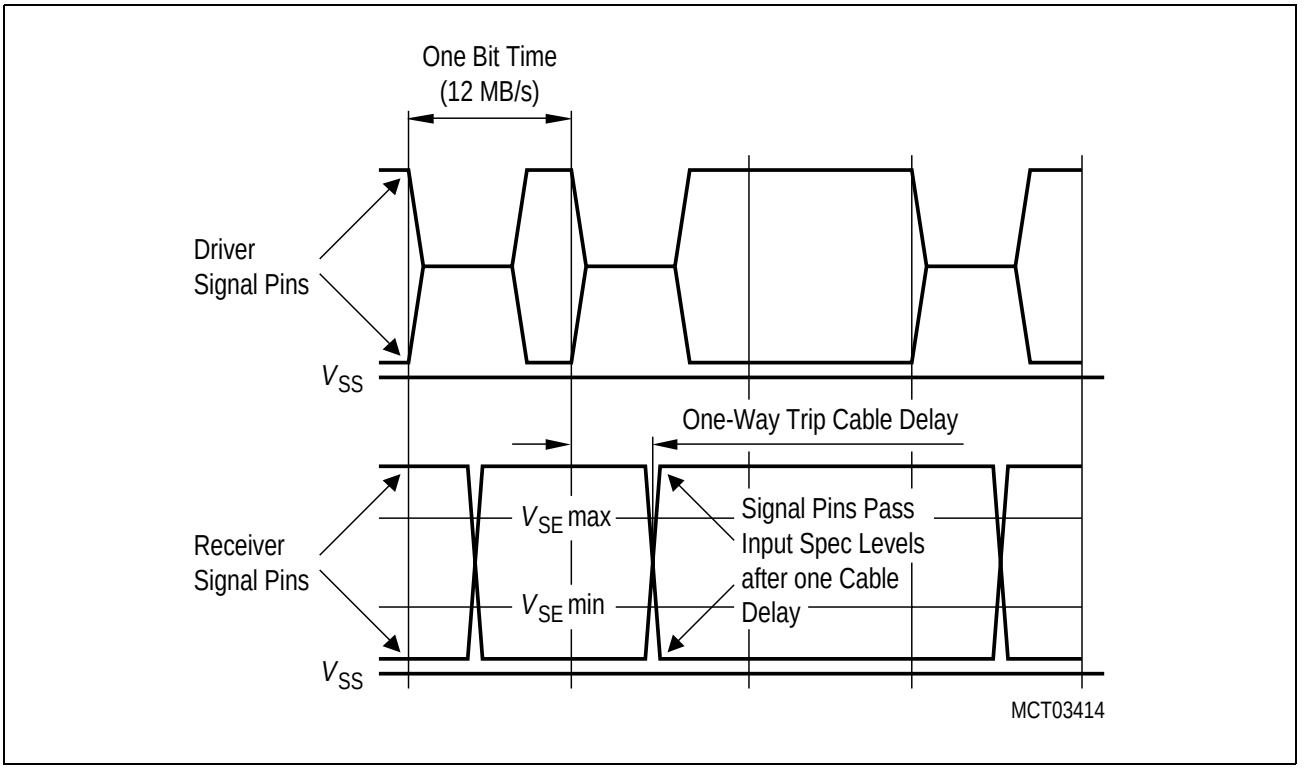


Figure 6-36
Full Speed USB Driver Signal Waveforms

For a low speed USB connection the rise and fall time of the signals are greater than 75 ns to keep RFI emissions under FCC class B limits, and less than 300 ns to limit timing delays and signaling skews and distortions. The driver reaches the specified static signal levels with smooth rise and fall times, and minimal reflections and ringing when driving the USB cable (see **figure 6-37**). This cable and driver are used only on network segments between low speed devices and the ports to which they are connected.

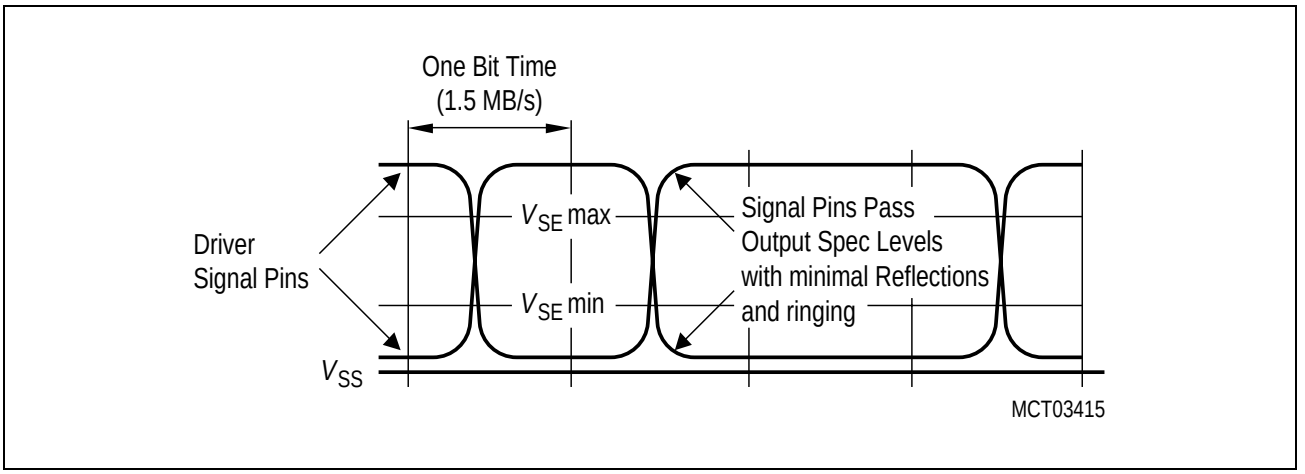


Figure 6-37
Low Speed USB Driver Signal Waveforms

6.4.10 Detection of Connected Devices

Full speed and low speed USB devices are differentiated by the position of the pullup resistor on the downstream end of the cable. Full speed devices are terminated with the pull-up on the D+ line and low speed devices are terminated with the pull-up in the D- line (see figure 6-). The pull-up terminator is a 1.5 kΩ resistor tied to a voltage source between 3.0 and 3.6 V referenced to the local ground.

When a device is attached to the host or hub, the data line with the pull-up is above 2.8 V and the other data line is near ground (idle state). A connect condition will be detected when one of the data lines is pulled above the single-ended high threshold level for more than 2.5 µs (30 full speed data bit times).

When a device is detached from the USB, the pull-down resistors will cause both D+ and D- lines to be pulled below the single-ended low threshold of the host or hub port. This creates a state called a single-ended zero (SE0) on the downstream port. A disconnect condition is indicated if an SE0 persists on a downstream port for more than 2.5 µs (30 full speed data bit times).

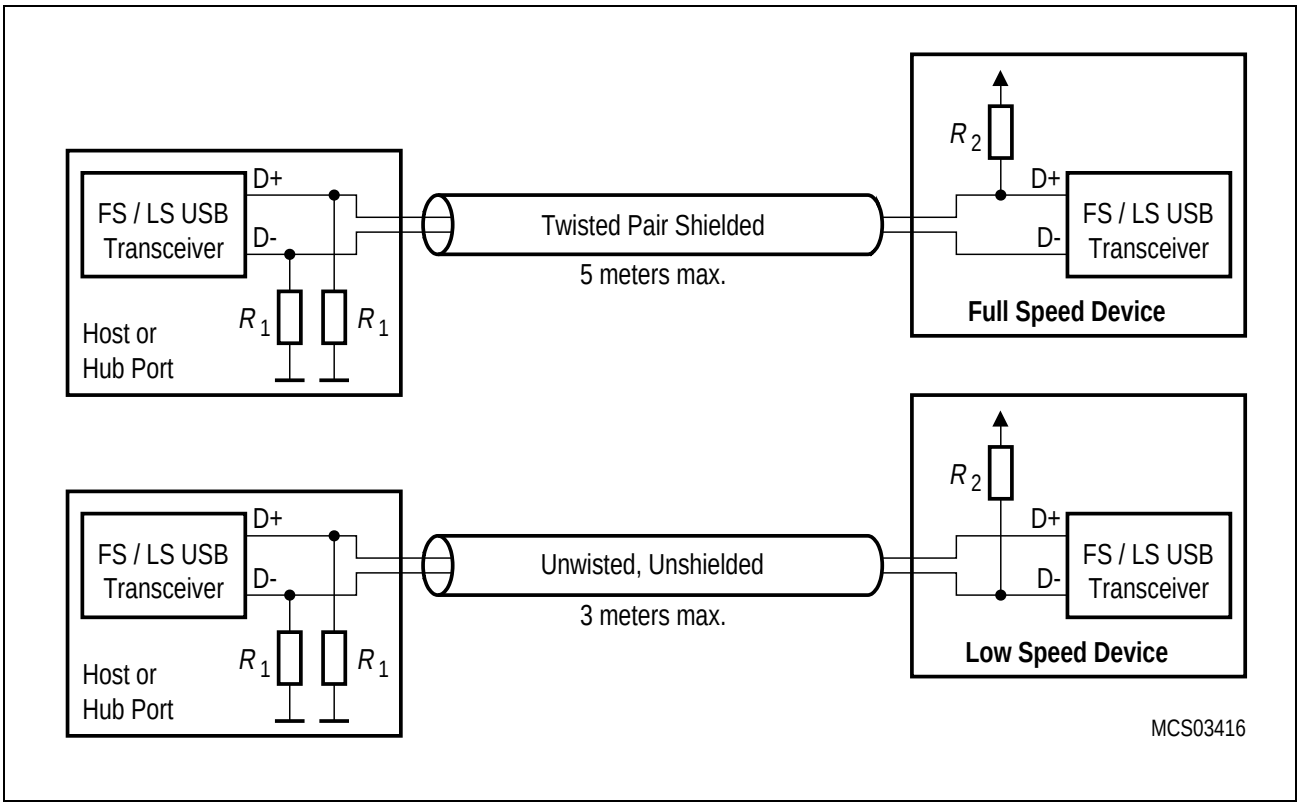


Figure 6-38
Low Speed / High Speed Device Cable and Resisitor Connection

Resistor R_2 has to be external. In some cases it might be necessary to hide the USB device from the host even when it is plugged in. To accomplish this, the pull-up resistor R_2 can be made switchable with a port a transistor.

6.4.11 Detach / Attach Detection

The USB device can be used in two different modes concerning its power supply, the bus-powered mode and the self-powered mode.

6.4.11.1 Self-Powered Mode

In self-powered mode, the USB device has its own power supply. The USB device has to detect whether it is connected to USB bus or not. This detection is done by hardware by using the Device-Attached Device-Detached pin DADD. When the device-attached condition is detected, bit DA in the device control register DCR is set and a device interrupt can be generated if required (bit DAI in SFR DIRR is set). The interrupt service routine of this device interrupt must completely initialize the USB device/module. The device-detached detection resets bit DA (and sets bit DDI in SFR DIRR) and can generate a device interrupt, too.

6.4.11.2 Bus-Powered Mode

In bus-powered mode, the USB device is driven by the power supply from the USB bus. The maximum power consumption is given by the USB specification. In order to respect this specification, the power consumption in suspend mode should not exceed 500 µA.

An explicit device-attached detection in this mode is not necessary. If the CPU is running, the device is attached, so the USB device/module has to be configured only after power-on. The device-detach action has no significance concerning software, because the device is no longer powered and the CPU stops. As a result, no attach-detach detection is needed. In this mode, pin DADD can be used as standard IO pin with bit DA monitoring its status and the interrupt generation on DA should not be used. If the interrupt generation on bit DA remains activated, a request must not be interpreted as attached-detached action, but as an external interrupt request on pin DADD, which is generating a device interrupt.

7 Interrupt System

The C541U provides seven interrupt sources with two priority levels. Five interrupts can be generated by the on-chip peripherals (timer 0, timer 1, SSC interface, and USB module), and two interrupts may be triggered externally (P3.2/$\overline{INT0}$ and P3.3/$\overline{INT1}$).

Figure 7-1 and **7-2** give a general overview of the interrupt sources and illustrate the request and control flags which are described in the next sections.

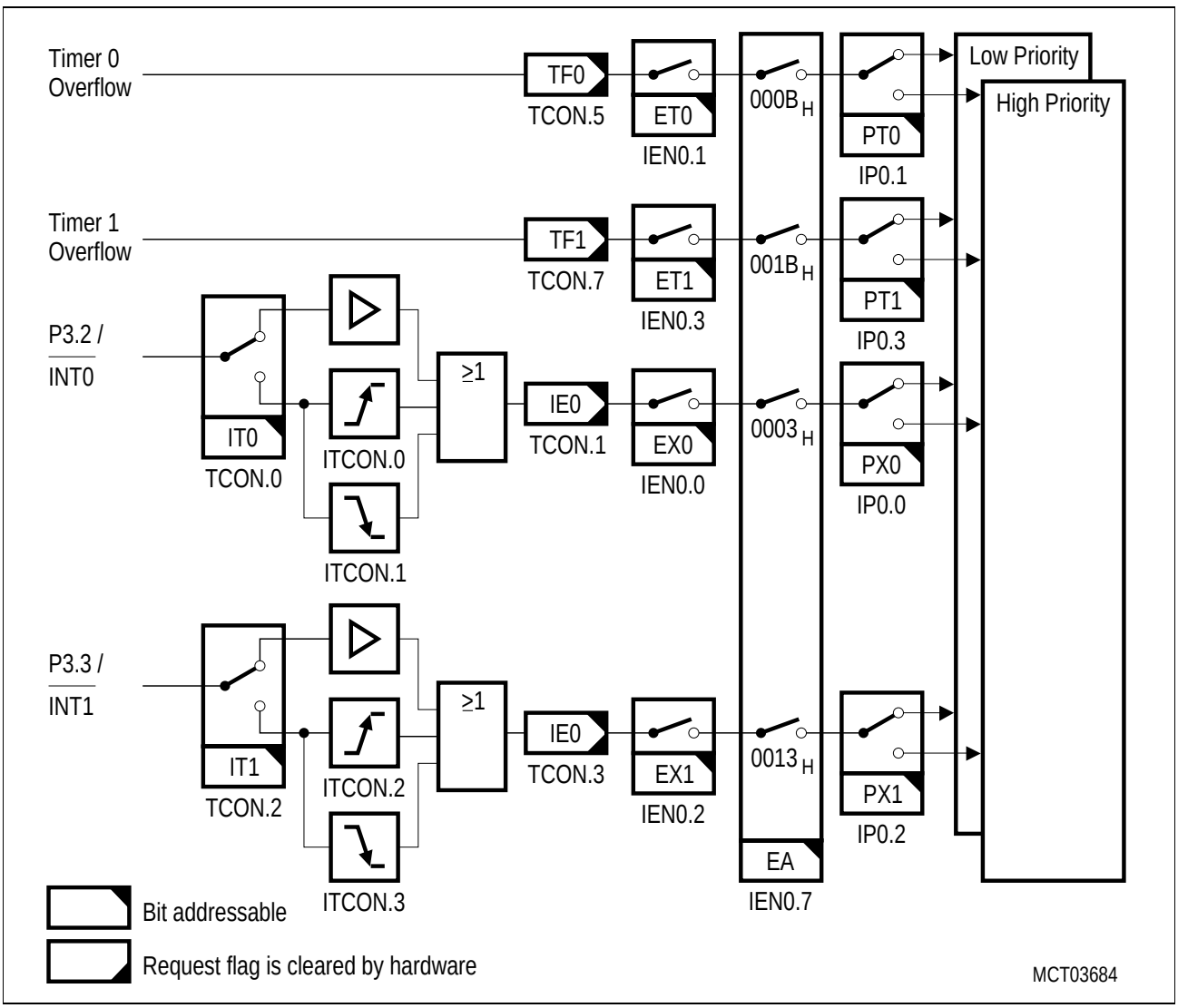


Figure 7-1
Interrupt Request Sources (Part 1)

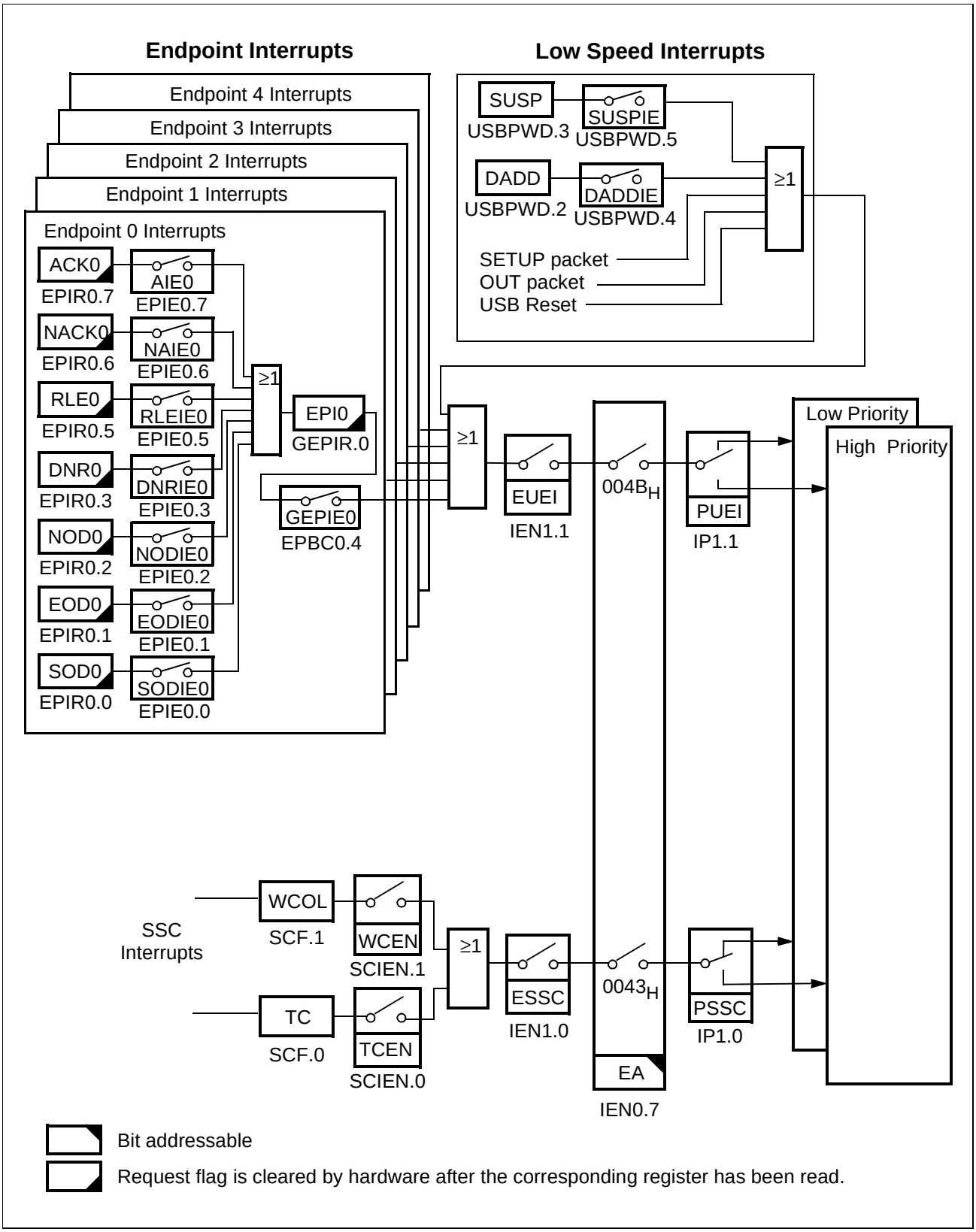


Figure 7-2
Interrupt Request Sources (Part 2)

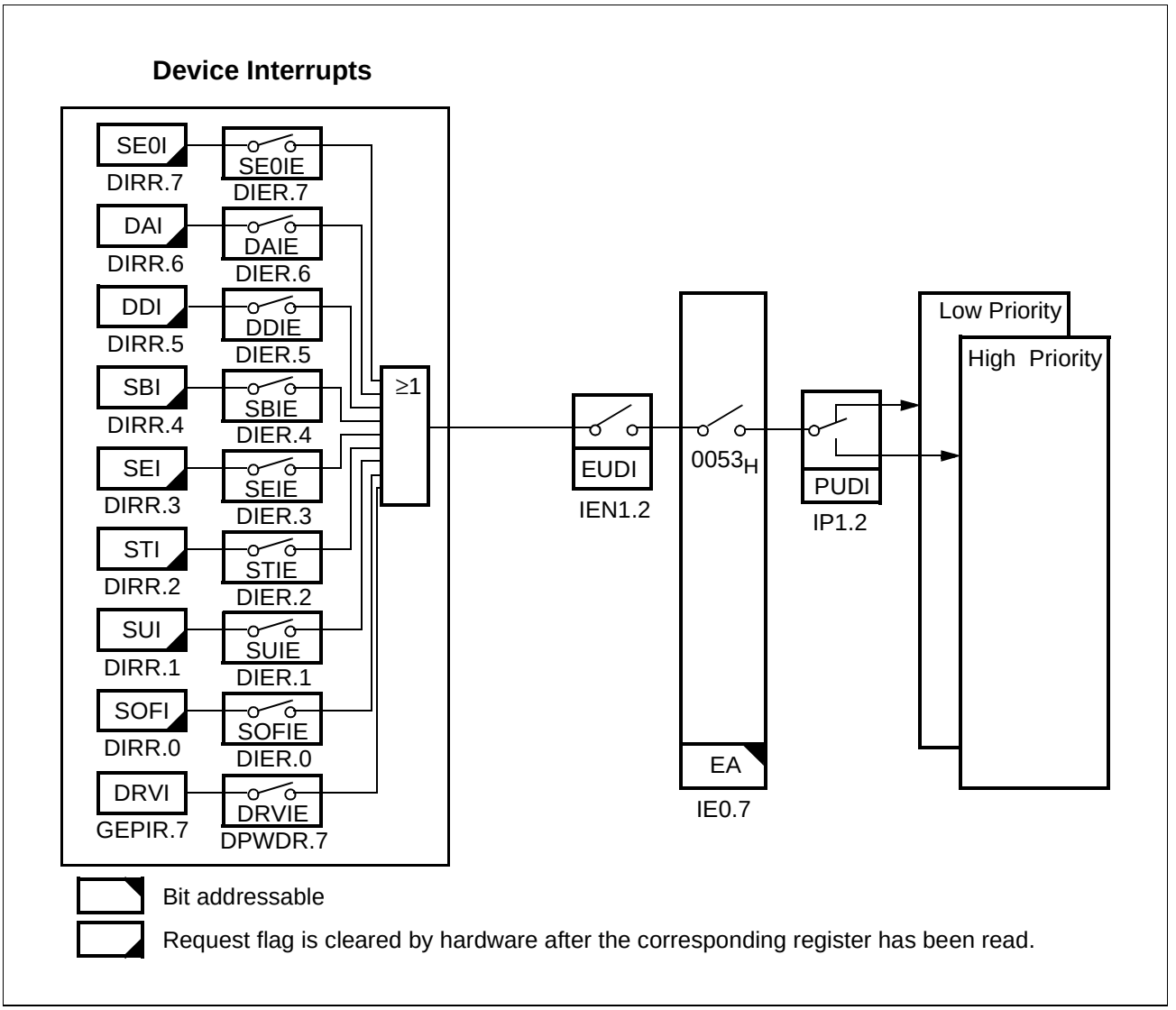


Figure 7-3
Interrupt Request Sources (Part 3)

7.1 Interrupt Registers

7.1.1 Interrupt Enable Registers

Each interrupt vector can be individually enabled or disabled by setting or clearing the corresponding bit in the interrupt enable registers IEN0 or IEN1. Register IEN0 also contains the global disable bit (EA), which can be cleared to disable all interrupts at once. The SSC and USB interrupts sources have further enable bits for individual interrupt control. Such interrupt enable bits are controlled by specific bits in the SFRs of the corresponding peripheral units.

The IEN0 register contains the general enable/disable flags of the external interrupts 0 and 1 as well as the timer interrupts. The SSC interrupt and the two USB interrupts are enabled/disabled by bits in the IEN1 register. After reset the enable bits of IEN0 and IEN1 are set to 0. That means that the corresponding interrupts are disabled.

Special Function Registers IEN0 (Address $A8_H$) **Reset Value : $0XXX0000_B$**

Bit No.	MSB							LSB	
	AF_H	AE_H	AD_H	AC_H	AB_H	AA_H	$A9_H$	$A8_H$	
$A8_H$	EA	–	–	–	ET1	EX1	ET0	EX0	IEN0

Bit	Function
EA	**Enable/disable all Interrupts** If EA=0, no interrupt will be acknowledged. If EA=1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	Not implemented. Reserved for future use.
ET1	**Timer 1 overflow interrupt enable** If ET1 = 0, the timer 1 interrupt is disabled If ET1 = 1, the timer 1 interrupt is enabled.
EX1	**External interrupt 1 enable** If EX1 = 0, the external interrupt 1 is disabled. If EX1 = 1, the external interrupt 1 is enabled.
ET0	**Timer 0 overflow interrupt enable** If ET0 = 0, the timer 0 interrupt is disabled If ET0 = 1, the timer 0 interrupt is enabled.
EX0	**External interrupt 0 enable** If EX0 = 0, the external interrupt 0 is disabled. If EX0 = 1, the external interrupt 0 is enabled.

Special Function Registers IEN1 (Address $A9_H$) **Reset Value : $XXXXX000_B$**
Special Function Registers SCIEN (Address AC_H) **Reset Value : $XXXXXX00_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$A9_H$	–	–	–	–	–	EUDI	EUEI	ESSC	IEN1
AC_H	–	–	–	–	–	–	WCEN	TCEN	SCIEN

Bit	Function
–	Not implemented. Reserved for future use.
EUDI	**Enable USB device interrupt enable** If EUDI = 0, the general USB device interrupt is disabled. If EUDI = 1, the general USB device interrupt is enabled.
EUEI	**Enablel USB endpoint interrupt** If EUEI = 0, the general USB endpoint interrupt is disabled. If EUEI = 1, the general USB endpoint interrupt is enabled.
ESSC	**SSC general interrupt enable** If ESSC = 0, the SSC general interrupt is disabled. If ESSC = 1, the SSC general interrupt is enabled.
WCEN	**SSC write collision interrupt enable** If WCEN = 0, the SSC write collision interrupt is disabled. If WCEN = 1, the SSC write collison interrupt is enabled. Additionally, bit ESSC must be set if the SSC write collision interrupt should be generated.
TCEN	**SSC transfer completed interrupt enable** If TCEN = 0, the SSC transfer completed interrupt is disabled. If TCEN = 1, the SSC transfer completed interrupt is enabled. Additionally, bit ESSC must be set if the SSC transfer completed interrupt should be generated.

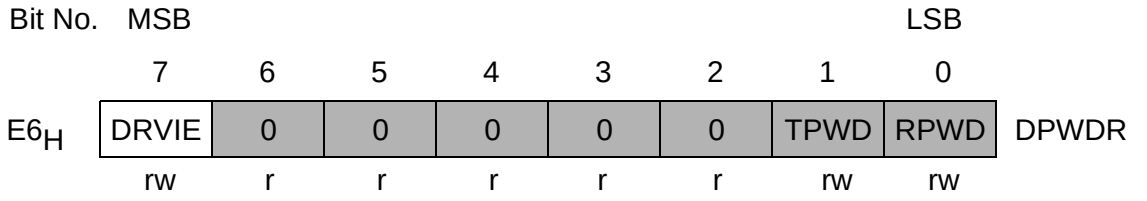
Special Function Registers DIER (Address $C3_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C3_H$	SE0IE	DAIE	DDIE	SBIE	SEIE	STIE	SUIE	SOFIE	DIER

For accessing DIER, SFR EPSEL must be 80_H.

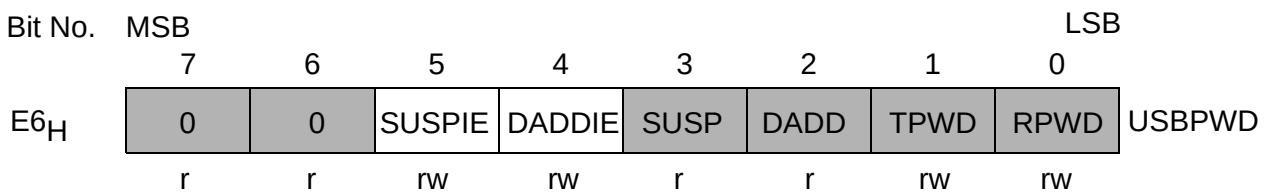
Bit	Function
SE0IE	**USB single ended zero interrupt enable** If SE0IE=0, the single ended zero interrupt is disabled. If SE0IE=1, the single ended zero interrupt is enabled
DAIE	**USB device attached interrupt enable** If DAIE=0, the USB device attached interrupt is disabled. If DAIE=1, the USB device attached interrupt is enabled.
DDIE	**USB device detached interrupt enable** If DDIE=0, the USB device detached interrupt is disabled. If DDIE=1, the USB device detached interrupt is enabled.
SBIE	**USB suspend begin interrupt enable** If SBIE=0, the USB suspend begin interrupt is disabled. If SBIE=1, the USB suspend begin interrupt is enabled.
SEIE	**USB suspend change interrupt enable** If SEIE=0, the USB suspend change interrupt is disabled. If SEIE=1, the USB suspend change interrupt is enabled.
STIE	**USB status interrupt enable** If STIE=0, the USB status interrupt is disabled. If STIE=1, the USB status interrupt is enabled.
SUIE	**USB setup interrupt enable** If SUIE=0, the USB setup interrupt is disabled. If SUIE=1, the USB setup interrupt is enabled.
SOFIE	**USB start of frame interrupt enable** If SOFIE=0, the USB start of frame interrupt is disabled. If SOFIE=1, the USB start of frame interrupt is enabled.

Special Function Register DPWDR (Address $C2_H$) **Reset Value : 00_H**



Bit	Function
DRVIE	**Device request value interrupt enable** Setting bit DRVIE enables the generation of a device interrupt each time the host sends device request that contains one or more of the following : - Configuration Value (through SET_CONFIGURATION device request) - Alternate Setting (through SET_INTERFACE device request) - Interface (through SET_INTERFACE device request) If DRVIE=0, the device request value interrupt is disabled. If DRVIE=1, the device request value interrupt is enabled.

Special Function Register USBPWD (Address $E6_H$) **Reset Value : 00_H**



Bit	Function
SUSPIE	**Suspend Interrupt Enable/Disable bit** 0: Suspend interrupt generation is disabled. 1: Suspend interrupt generation is enabled.
DADDIE	**Device Attach/Detach Interrupt Enable/Disable bit** 0: Interrupt generation upon DADD/P3.1 pin status is disabled. 1: Interrupt generation upon DADD/P3.1 pin status is enabled.

Special Function Registers EPIEn, n=0-4 (Address $C3_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C3_H$	AIEn	NAIEn	RLEIEn	–	DNRIEn	NODIEn	EODIEn	SODIEn	EPIEn

For accessing EPIEn, SFR EPSEL must be $0n_H$.

Bit	Function
AIEn	**USB acknowledge interrupt enable** If AIEn=0, the USB acknowledge interrupt is disabled. If AIEn=1, the USB acknowledge interrupt is enabled.
NAIEn	**USB not acknowledged interrupt enable** If NAIEn=0, the USB not acknowledged interrupt is disabled. If NAIEn=1, the USB not acknowledged interrupt is enabled.
RLEIEn	**USB read length error interrupt enable** If RLEIEn=0, the USB read length error interrupt is disabled. If RLEIEn=1, the USB read length error interrupt is enabled.
–	Reserved bit for future use.
DNRIEn	**USB data not ready interrupt enable** If DNRIEn=0, the USB data not ready interrupt is disabled. If DNRIEn=1, the USB data not ready interrupt is enabled.
NODIEn	**USB nodata interrupt enable** If NODIEn=0, the USB no data interrupt is disabled. If NODIEn=1, the USB no data interrupt is enabled.
EODIEn	**USB end of data interrupt enable** If EODIEn=0, the USB end of data interrupt is disabled. If EODIEn=1, the USB end of data interrupt is enabled.
SODIEn	**USB start of data interrupt enable** If SODIEn=0, the USB start of data interrupt is disabled. If SODIEn=1, the USB start of data interrupt is enabled.

Endpoint n Buffer Control RegisterEPBCn, n=0-4 (Address $C1_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C1_H$	STALLn	0	0	GEPIEn	SOFDEn	INCEn	0	DBMn	EPBCn
	rw	r	r	rw	rw	rw	r	rw	

The shaded bits are not used for interrupt control.

Bit	Function
GEPIEn	**Global endpoint interrupt enable** Bit GEPIEn enables or disables the generation of the global endpoint interrupt forr endpoint n based on the endpoint specific interrupt request bits in register EPIRn. If GEPIEn=0, the USB endpoint n interrupt is disabled. If GEPIEn=1, the USB endpoint n interrupt is enabled.

Note: For accessing the EPBCn registers SFR EPSEL($D2_H$) must be set with the appropriate value.

7.1.2 Interrupt Request / Control Flags

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

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

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

	MSB							LSB	
Bit No.	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H	
88_H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	TCON

The shaded bits are not used for interrupt control.

Bit	Function
TF1	**Timer 1 overflow flag** Set by hardware on timer/counter 1 overflow. Cleared by hardware when processor vectors to interrupt routine.
TF0	**Timer 0 overflow flag** Set by hardware on timer/counter 0 overflow. Cleared by hardware when processor vectors to interrupt routine.
IE1	**External interrupt 1 request flag** Set by hardware when external interrupt 1 edge is detected. Cleared by hardware when processor vectors to interrupt routine.
IT1	**External interrupt 1 level/edge trigger control flag** If IT1 = 0, low level triggered external interrupt 1 is selected. If IT1 = 1, edge triggered mode for external interrupt 1 is selected. The bits IE1TR and IE1TF in SFR ITCON (see section 7.4) further define which signal transition at pin $\overline{INT1}$ (rising and/or falling edge) generates an interrupt.
IE0	**External interrupt 0 request flag** Set by hardware when external interrupt 0 edge is detected. Cleared by hardware when processor vectors to interrupt routine.
IT0	**External interrupt 0 level/edge trigger control flag** If IT0 = 0, low level triggered external interrupt 0 is selected. If IT0 = 1, edge triggered mode for external interrupt 0 is selected. The bits IE0TR and IE0TF in SFR ITCON (see section 7.4) further define which signal transition at pin $\overline{INT0}$ (rising and/or falling edge) generates an interrupt.

Special Function Register SCF (Address AB_H) **Reset Value : $XXXXXX00_B$**

	MSB							LSB	
Bit No.	7	6	5	4	3	2	1	0	
AB_H	–	–	–	–	–	–	WCOL	TC	SCF

Bit	Function
–	Reserved bits for future use.
WCOL	**SSC write collision interrupt flag** WCOL set indicates that an attempt was made to write to the shift register STB while a data transfer was in progress and not fully completed. Bit WCEN in the SCIEN register must be set, if an interrupt request will be generated when WCOL is set.
TC	**SSC transfer complete interrupt flag** If TC is set it indicates that the last transfer has been completed. Bit TCEN in the SCIEN register must be set, if an interrupt request will be generated when TC is set.

The **SSC interrupt** is generated by a logical OR of flag WCOL and TC in SFR SCF. Both bits can be cleared by software when a “0“ is written to the bit location. WCOL is reset by hardware when after a preceeding read operation of the SCF register the SSC transmit data register STB is written with data. TC is reset by hardware when after a preceeding read operation of the SCF register the receive data register SRB is read the next time. The interrupt service routine will normally have to determine whether it was the WCOL or the TC flag that generated the interrupt, and the bit will have to be cleared by software.

The USB device interrupt request register contains the device specific interrupt flags of the USB module. These flags describe special events of the USB module. If a request flag is set, it is automatically cleared after a read operation of the DIRR register. The most significant bit in GEPIR register contains an additional device specific interrupt flag, but it needs to be cleared by software.

Special Function Registers DIRR (Address $C4_H$) **Reset Value : 00_H**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C4_H$	SE0I	DAI	DDI	SBI	SEI	STI	SUI	SOFI	DIRR

For accessing DIRR, SFR EPSEL must be 80_H.

Bit	Function
SE0I	**Single ended zero interrupt** Bit SE0I is set each time a single ended zero is detected for equal or greater than 2.5µs. EOP (2 bit times) is not detected.
DAI	**Device attached interrupt** Bit DAI is automatically set after detection of the USB device being attached to the USB bus.
DDI	**Device detached interrupt** Bit DDI is automatically set after detection of the device being detached from the USB bus.
SBI	**Suspend begin interrupt** Bit SBI is automatically set when the suspend mode is entered.
SEI	**Suspend end interrupt** Bit SBI is automatically set when the suspend mode is left.
STI	**Status interrupt** Bit STI is set if the host requests a status transfer and the device answers with NACK.
SUI	**Setup interrupt** Bit SUI is automatically set after a successful reception of a setup packet.
SOFI	**Start of frame** Bit SOF is automatically set after detection of a start of frame packet on the USB.

Special Function Register GEPIR (Address $D6_H$) **ResetValue : 00_H**

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
$D6_H$	DRVI	0	0	EPI4	EPI3	EPI2	EPI1	EPI0	GEPIR
	rw	r	r	r	r	r	r	r	

Bit	Function
DRVI	**Device request value interrupt** Bit DRVI is set each time the host sends device request that contains one or more of the following : - Configuration Value (through SET_CONFIGURATION device request) - Alternate Setting (through SET_INTERFACE device request) - Interface (through SET_INTERFACE device request). This flag can only be cleared by writing ‘0’ to the bit. Writing ‘1’ to the bit will be ignored.

Special Function Register USBPWD (Address $E6_H$) **Reset Value : 00_H**

Bit No.	MSB							LSB	
	7	6	5	4	3	2	1	0	
$E6_H$	0	0	SUSPIE	DADDIE	SUSP	DADD	TPWD	RPWD	USBPWD
	r	r	rw	rw	r	r	rw	rw	

Bit	Function
SUSP	**Suspend mode** This bit is set when the USB has been idle for more than 3 ms. It will remain set until there is a non-idle state on the USB cable.
DADD	**Device attached or detached** Bit DADD monitors the state of the associated I/O pin DADD (see also TYPE bit field of USBDCR).

The register EPIRn (n=0-4) contaíns USB endpoint specific interrupt request flags. This SFR is availble for each endpoint. If a request flag in EPIRn is set, it is automatically cleared after a read operation of the EPIRn register.

Endpoint Interrupt Request Register EPIRn, n=0-4 (Address $C4_H$) Reset Value EPIR0 : 11_H
Reset Value EPIR1 to EPIR4 : 10_H

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C4_H$	ACKn	NACKn	RLEn	–	DNRn	NODn	EODn	SODn	EPIRn
	r	r	r	r	r	r	r	r	

For accessing EPIRn, SFR EPSEL must be $0n_H$.

Bit	Function
ACKn	**USB acknowledge** Bit ACKn=1 indicates a succesful action on the USB.
NACKn	**USB not acknowledge** Bit NACK is set for all unsuccessful actions on the USB.
RLEn	**Read length error** Bit RLEn is automatically set if the number of bytes read by the USB does not correspond to the packet length programmed by the CPU.
–	Reserved bit for future use;
DNRn	**Data not ready** This bit is set by hardware if the USB module requires an access to USB memory, but no buffer is available.
NODn	**No data** This bit indicates an incorrect CPU read or write access to USB memory.
EODn	**End of data** During an USB read access EOD is set if the CPU has written a programmable number of bytes in the transmit buffer. During an USB write access EOD is set if the CPU has read a programmable number of bytes out of the receive buffer.
SODn	**Start of data** During an USB read access SOD is set if the USB has read a fixed number of bytes from the transmit buffer. During an USB write access SOD is set if the USB has written a fixed number of bytes to the receive buffer.

The global endpoint interrupt request register GEPIRn (n=0-4) contaíns one flag for each endpoint which indicates whether one or more of the seven endpoint specific interrupt requests has become active. If a request flag in GEPIR is set, it is automatically cleared after a read operation of the GEPIR register.

USB Global Endpoint Interrupt Request Register GEPIR (Address $D6_H$) Reset Value : 00_H

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$D6_H$	DRVI	0	0	EPI4	EPI3	EPI2	EPI1	EPI0	GEPIR
	r	r	r	r	r	r	r	r	

Bit	Function
EPI4	**Endpoint 4 interrupt request flag** If EPI4 is set, an endpoint 4 interrupt request is pending.
EPI3	**Endpoint 3 interrupt request flag** If EPI3 is set, an endpoint 3 interrupt request is pending.
EPI2	**Endpoint 2 interrupt request flag** If EPI2 is set, an endpoint 2 interrupt request is pending.
EPI1	**Endpoint 1 interrupt request flag** If EPI1 is set, an endpoint 1 interrupt request is pending.
EPI0	**Endpoint 0 interrupt request flag** If EPI0 is set, an endpoint 0 interrupt request is pending.

7.1.3 Interrupt Prioritiy Registers

Each interrupt source can also be individually programmed to one of two priority levels by setting or clearing a bit in the SFRs IP0 or IP1 (interrupt priority: 0 = low priority, 1 = high priority).

Special Function Registers IP0 (Address $B8_H$) **Reset Value : $XXXX0000_B$**
Special Function Registers IP1 (Address $B9_H$) **Reset Value : $XXXXX000_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$B8_H$	–	–	–	–	PT1	PX1	PT0	PX0	IP0
$B9_H$	–	–	–	–	–	PUDI	PUEI	PSSC	IP1

Bit	Function
–	Reserved for future use.
PT1	**Timer 1 overflow interrupt priority level** If PT1 = 0, the timer 1 interrupt has a low priority. If PT1 = 1, the timer 1 interrupt has a high priority
PX1	**External interrupt 1 priority level** If PX1 = 0, the external interrupt 1 has a low priority. If PX1 = 1, the external interrupt 1 has a high priority.
PT0	**Timer 0 overflow interrupt priority level** If PT0 = 0, the timer 0 interrupt has a low priority. If PT0 = 1, the timer 0 interrupt has a high priority.
PX0	**External interrupt 0 priority level** If PX0 = 0, the external interrupt 0 has a low priority. If PX0 = 1, the external interrupt 0 has a high priority.
PUDI	**USB device interrupt priority level** If PUDI = 0, the USB device interrupt has a low priority. If PUDI = 1, the USB device interrupt has a high priority.
PUEI	**USB endpoint interrupt priority level** If PUEI = 0, the USB endpoint interrupt has a low priority. If PUEI = 1, the USB endpoint interrupt has a high priority.
PSSC	**SSC interrupt priority level** If PSSC = 0, the SSC interrupt has a low priority. If PSSC = 1, the SSC interrupt has a high priority.

7.2 Interrupt Priority Level Structure

A low-priority interrupt can itself be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt cannot be interrupted by any other interrupt source.

If two requests of different priority level are received simultaneously, the request of higher priority is serviced. If requests of the same priority are received simultaneously, an internal polling sequence determines which request is serviced. Thus, within each priority level there is a second priority structure determined by the polling sequence (vertical and horizontal) as shown in **table 7-1** below. If e.g. the external interrupt 0 and the SSC interrupt have the same priority and if they are active simultaneously, the external interrupt 0 will be serviced first.

Table 7-1
Interrupt Source Structure

Interrupt Source		**Priority**
High Priority →	Low Priority	
External Interrupt 0	SSC Interrupt	High
Timer 0 Interrupt	USB Endpoint Interrupt	↓
External Interrupt 1	USB Device Interrupt	
Timer 1 Interrupt	–	Low

7.3 How Interrupts are Handled

The interrupt flags are sampled at S5P2 in each machine cycle. The sampled flags are polled during the following machine cycle. If one of the flags was in a set condition at S5P2 of the preceeding cycle, the polling cycle will find it and the interrupt system will generate a LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1) An interrupt of equal or higher priority is already in progress.
2) The current (polling) cycle is not in the final cycle of the instruction in progress.
3) The instruction in progress is RETI or any write access to registers IEN0/IEN1 or IP0/IP1.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress is completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any write access to registers IEN0/IEN1 or IP0/IP1, then at least one more instruction will be executed before any interrupt is vectored 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 7-4**.

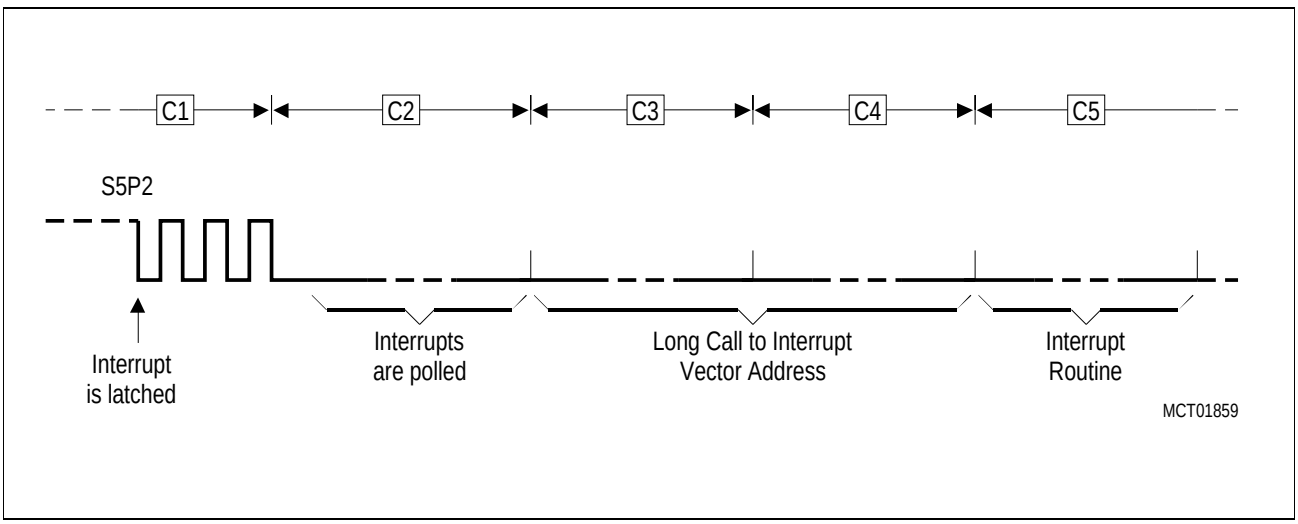


Figure 7-4
Interrupt Response Timing Diagram

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

Thus, the processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine. In some cases it also clears the flag that generated the interrupt, while in other cases it does not; then this has to be done by the user's software. The hardware clears the external interrupt flags IE0 and IE1 only if they were transition-activated. The hardware-generated LCALL pushes the contents of the program counter onto the stack (but it does not save the PSW) and reloads the program counter with an address that depends on the source of the interrupt being vectored too, as shown in the following **table 7-2**.

Table 7-2
Interrupt Source and Vectors

Interrupt Source	Interrupt Vector Address	Interrupt Request Flags
External Interrupt 0	0003_H	IE0
Timer 0 Overflow	$000B_H$	TF0
External Interrupt 1	0013_H	IE1
Timer 1 Overflow	$001B_H$	TF1
SSC Interrupt	0043_H	TC, WCOL
USB Endpoint Interrupt	$004B_H$	in SFRs EPIR0-4 and GEPIR
USB Device Interrupt	0053_H	in SFRs DIRR and GEPIR
Wake-up from power down	$007B_H$	–

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

A special interrupt source is the power-down mode interrupt. This interrupt is automatically enabled when the C541U is in power-down mode and bit EWPD (enable wake-up from power-down mode) in SFR PCON1 is set. If these two conditions are met and when the oscillator watchdog unit start-up phase after a wake-up condition ($\overline{INT0}$=0) is finished, the C541U starts with an interrupt at address $007B_H$. All other interrupts are now disabled until the RETI instruction of the power-down interrupt routine has been executed.

7.4 External Interrupts

The external interrupts 0 and 1 can be programmed to be level-activated or transition activated by setting or clearing bit IT0 or IT1 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. In edge-triggered mode (ITx =) two bits of the ITCON register define the type of signal transition for which the external interrupt inputs are sensitive. Edge-triggered interrupt can be activated for an interrupt input signal at the rising edge, at the falling edge or at both signal transitions. In edge-triggered mode, if successive samples of the $\overline{\text{INTx}}$ pin show a different logic level in two consequent machine cycles, the corresponding interrupt request flag IEx in TCON is set. Flag bit IEx=1 then requests the interrupt.

If the external interrupt 0 or 1 is level-activated, the external source has to hold the request active until the requested interrupt is actually generated. Then it has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated.

Since the external interrupt pins are sampled once in each machine cycle, an input high or low should be held for at least 6 oscillator periods to ensure sampling. If the external interrupt is transition-activated, the external source has to hold the request pin high for at least one cycle, and then hold it low for at least one cycle to ensure that the transition is recognized so that the corresponding interrupt request flag will be set (see **figure 7-5**). The external interrupt request flags will automatically be cleared by the CPU when the service routine is called.

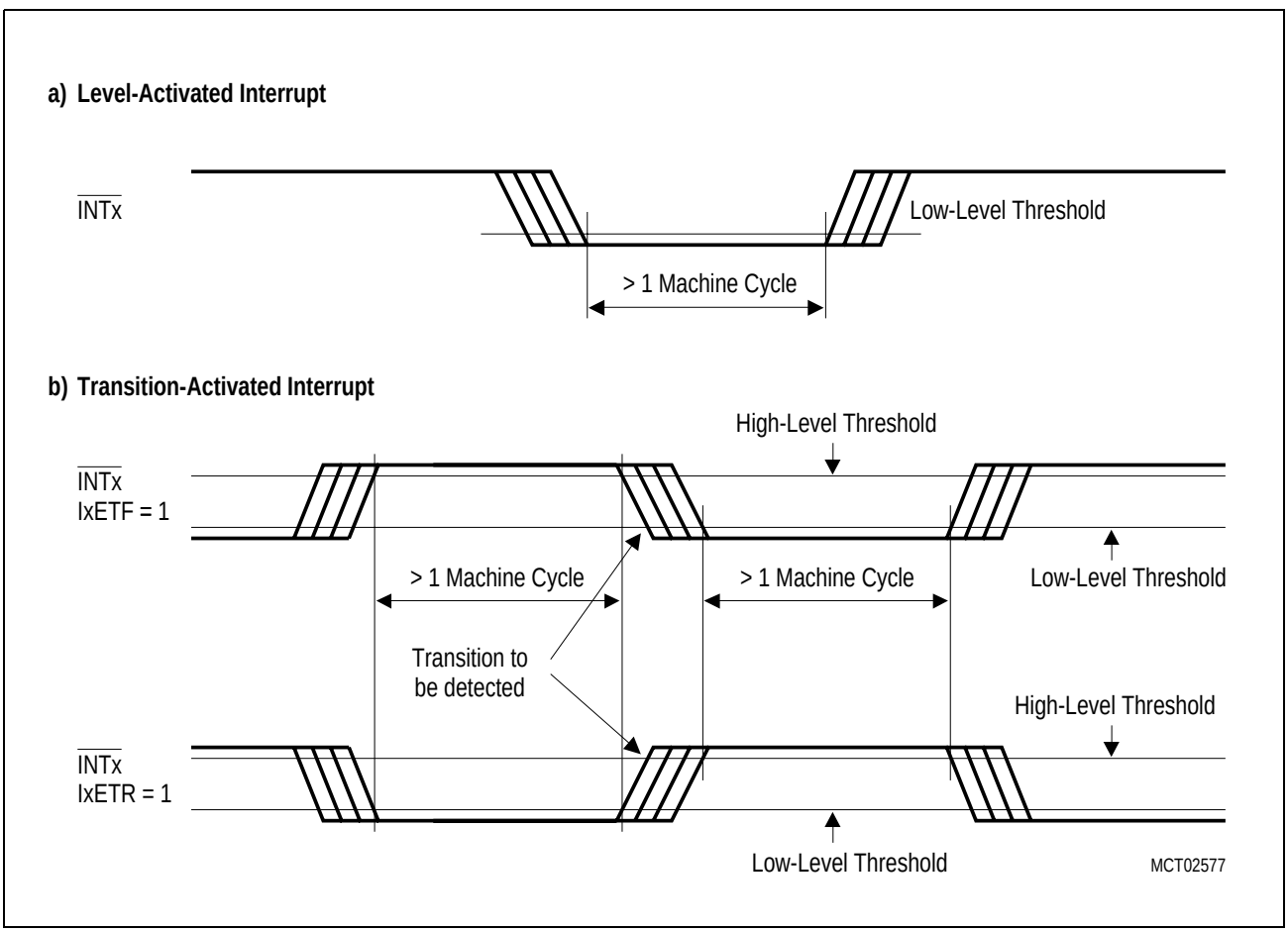
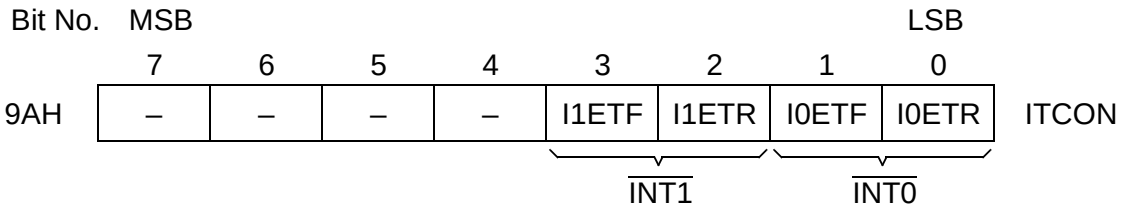


Figure 7-5
External Interrupt Detection

The edge-triggered interrupt mode selection for the external interrupts is selected by bits in SFR ITCON (External Interrupt Trigger Condition Register). The edge-trigger mode selection is defined in a way (default value of ITCON after reset), that their function is upward compatible to the basic external interrupt functionality of the C501.

Special Function Registers ITCON (Address $9A_H$) **Reset Value :** $XXXX1010_B$



Bit	Function
–	Reserved bit for future use.
IxETF IxETR	**External Interrupt Edge Trigger Mode Selection** (x=0,1 refers to $\overline{INT0}$, $\overline{INT1}$)

IxETF	IxETR	Function
0	0	$\overline{INTx}$ inputs are not sensitive for either rising or falling edge
0	1	$\overline{INTx}$ operates in rising edge-triggered mode
1	0	$\overline{INTx}$ operates in falling edge-triggered mode (default after reset)
1	1	$\overline{INTx}$ operates in falling and rising edge-triggered mode

7.5 Interrupt Response Time

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

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

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

8 Fail Safe Mechanisms

The C541U offers enhanced fail safe mechanisms, which allow an automatic recovery from software upset or hardware failure :

- a programmable watchdog timer (WDT), with variable time-out period from 256 µs up to approx. 0.55 µs at 12 MHz.
- an oscillator watchdog (OWD) which monitors the on-chip oscillator and forces the microcontroller into reset state in case the on-chip oscillator fails; it also provides the clock for a fast internal reset after power-on.

8.1 Programmable Watchdog Timer

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

The watchdog timer in the C541U is a 15-bit timer, which is incremented by a count rate of $f_{OSC}/12$ or $f_{OSC}/192$. The system clock of the C541U is divided by two prescalers, a divide-by-two and a divide-by-16 prescaler which are selected by bit WDTPSEL (WDTREL.7). 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.

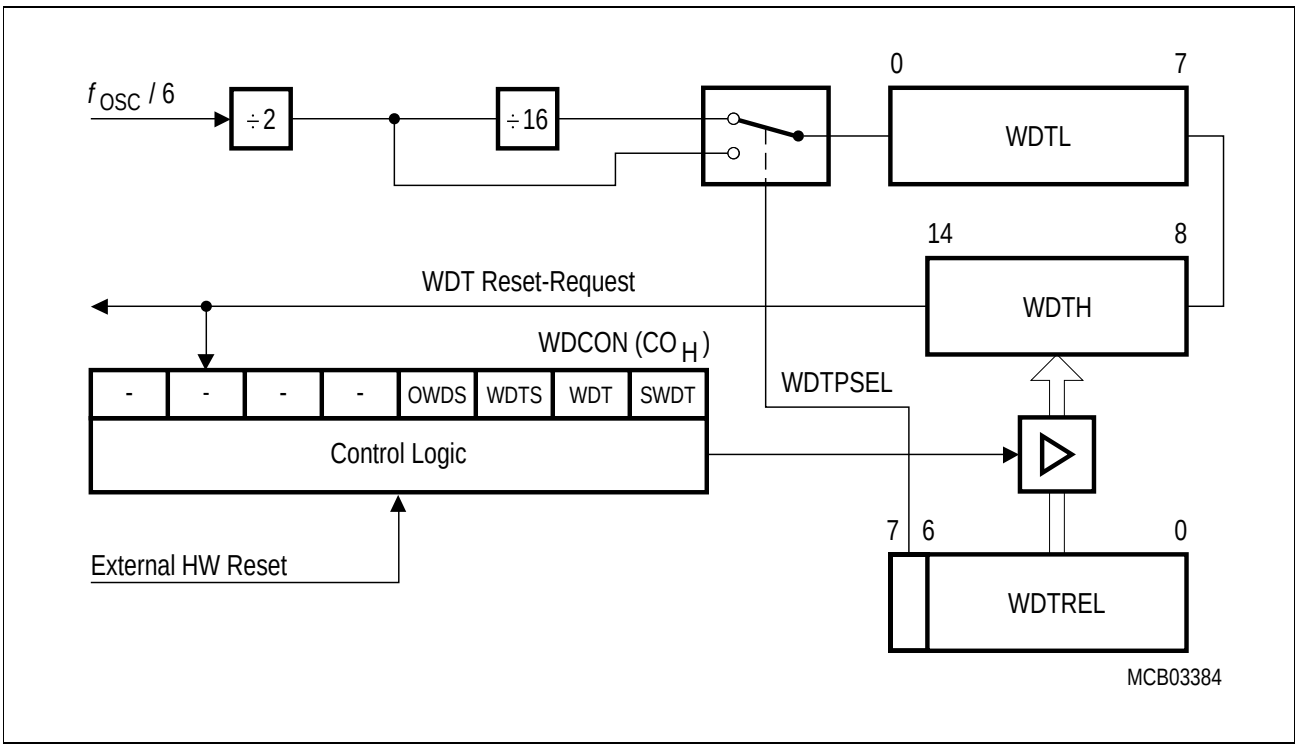


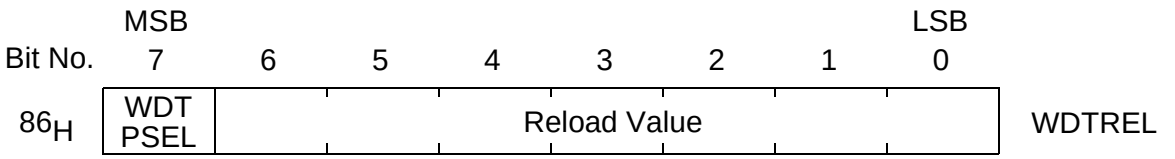
Figure 8-1
Block Diagram of the Programmable Watchdog Timer

Note : WDTH and WDTL cannot be accessed by software.

8.1.1 Input Clock Selection

The input clock rate of the watchdog timer is derived from the system clock of the C541U. There is a prescaler available, which is software selectable and defines the input clock rate. This prescaler is controlled by bit WDTPSEL in the SFR WDTREL. **Tabel 8-1** shows resulting timeout periods at f_{OSC} = 12 MHz.

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



Bit	Function
WDTPSEL	**Watchdog timer prescaler select bit.** If WDTPSEL=0, the watchdog timer is clocked by f_{OSC}/12 (default after reset). If WDTPSEL=1, the watchdog timer is clocked by f_{OSC}/192
WDTREL.6 - 0	**Seven bit reload value** for the high-byte of the watchdog timer. This value is loaded to WDTH when a refresh is triggered by a consecutive setting of bits WDT and SWDT.

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

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

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

8.1.2 Watchdog Timer Control / Status Flags

The watchdog timer is controlled by control and status flags which are located in SFR WDCON.

Special Function Register WDCON (Address $C0_H$) **Reset Value : $XXXX\ 0000_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C0_H$	–	–	–	–	OWDS	WDTS	WDT	SWDT	WDCON

Bit	Function
–	Reserved bits for future use.
OWDS	**Oscillator watchdog timer status flag.** Set by hardware when an oscillator watchdog reset occured. Can be set and cleared by software.
WDTS	**Watchdog timer status flag.** Set by hardware when a watchdog timer reset occured. Can be cleared and set by software.
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.

8.1.3 Starting the Watchdog Timer

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

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

8.1.4 Refreshing the Watchdog Timer

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

The reload register WDTREL can be written 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 WDTREL can be corrected by software.

Note : the watchdog timer registers WDTH and WDTL cannot be accessed by software.

8.1.5 Watchdog Reset and Watchdog Status Flag

If the software fails to clear the watchdog in time, an internally generated watchdog reset is entered at the counter state $7FFC_H$. The duration of the reset signal then depends on the prescaler selection (either 8 or 128 machine cycles). This internal reset differs from an external one in so far as the watchdog timer is not disabled and bit WDTS is set. The WDTS is a flip-flop, which is set by a watchdog timer reset and can be cleared by an external hardware reset. Bit WDTS allows the software to examine from which source the reset was activated. The bit WDTS can also be cleared by software.

8.2 Oscillator Watchdog Unit

The oscillator watchdog unit serves for three functions:

- **Monitoring of the on-chip oscillator's function**
  The watchdog supervises the on-chip oscillator's frequency; if it is lower than the frequency of the auxiliary RC oscillator in the watchdog unit, the internal clock is supplied by the RC oscillator and the device is brought into reset; if the failure condition disappears (i.e. the on-chip oscillator has a higher frequency than the RC oscillator), the part executes a final reset phase of typ. 1 ms in order to allow the oscillator to stabilize; then the oscillator watchdog reset is released and the part starts program execution again.
- **Fast internal reset after power-on**
  The oscillator watchdog unit provides a clock supply for the reset before the on-chip oscillator has started. The oscillator watchdog unit also works identically to the monitoring function.
- **Control of external wake-up from software power-down mode** (description see chapter 9)
  When the power-down mode is left by a low level at the $\overline{\text{INT0}}$ pin or by the USB, the oscillator watchdog unit assures that the microcontroller resumes operation (execution of the power-down wake-up interrupt) with the nominal clock rate. In the power-down mode the RC oscillator and the on-chip oscillator are stopped. Both oscillators are started again when power-down mode is released. When the on-chip oscillator has a higher frequency than the RC oscillator, the microcontroller starts operation after a final delay of typ. 1 ms in order to allow the on-chip oscillator to stabilize.

Note: The oscillator watchdog unit is always enabled.

8.2.1 Functionality of the Oscillator Watchdog Unit

Figure 8-2 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.

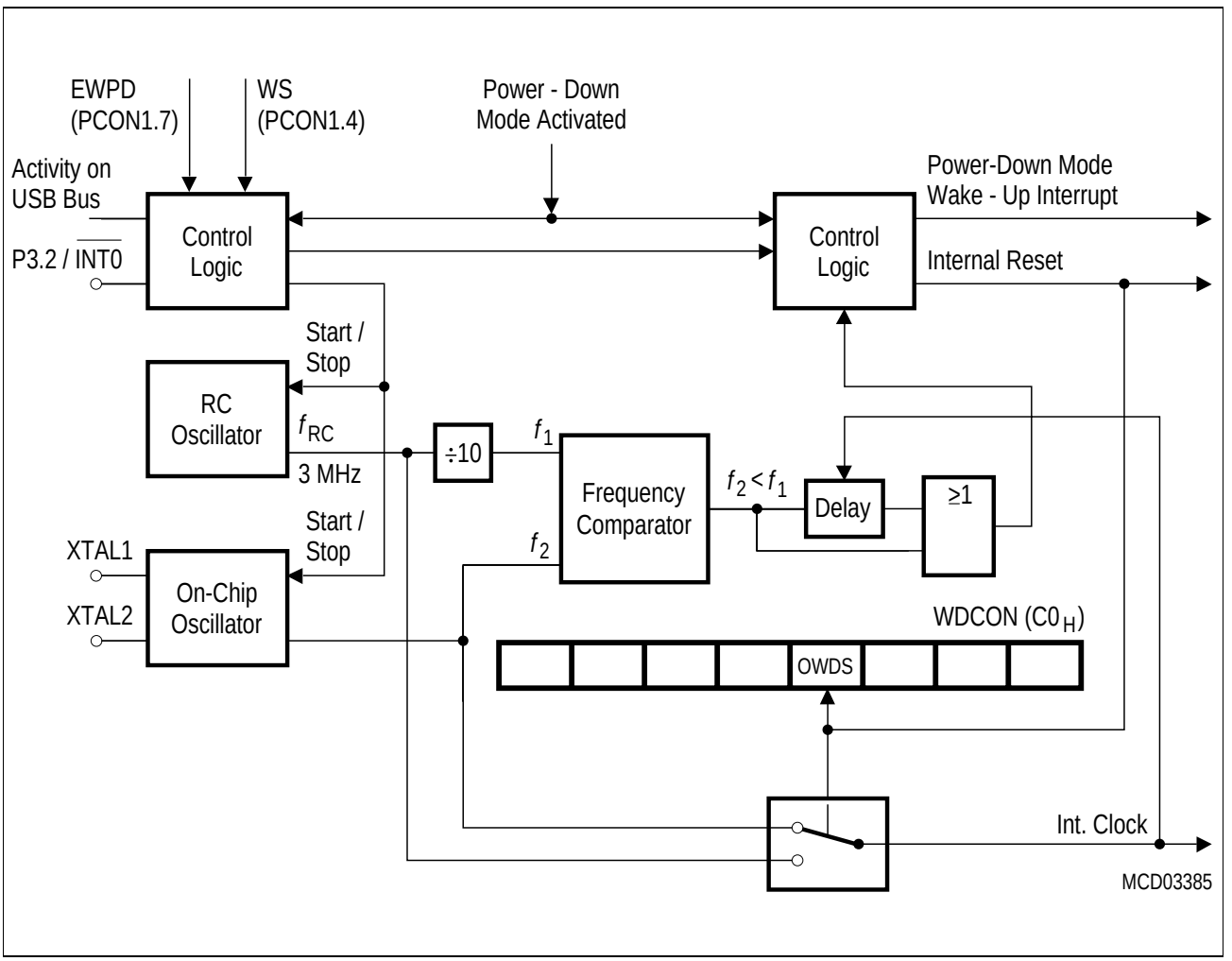


Figure 8-2
Functional Block Diagram of the Oscillator Watchdog

The frequency coming from the RC oscillator is divided by 10 and compared to the on-chip oscillator's frequency. If the frequency coming from the on-chip oscillator is found lower than the frequency derived from the RC oscillator the watchdog detects a failure condition (the oscillation at the on-chip oscillator could stop because of crystal damage etc.). In this case it switches the input of the internal clock system to the output of the RC oscillator. This means that the part is being clocked even if the on-chip oscillator has stopped or has not yet started. At the same time the watchdog activates the internal reset in order to bring the C541U in its defined reset state. The reset is performed because a clock is available from the RC oscillator. This internal watchdog reset has the same effects as an externally applied reset signal with the following exceptions: The watchdog timer status flag WDTS is not reset (the watchdog timer however is stopped) and bit OWDS is set. This allows the software to examine error conditions detected by the watchdog timer even if meanwhile an oscillator failure occured.

The oscillator watchdog is able to detect a recovery of the on-chip oscillator after a failure. If the frequency derived from the on-chip oscillator is again higher than the reference the watchdog starts a final reset sequence which takes typ. 1 ms. Within that time the clock is still supplied by the RC oscillator and the part is held in reset. This allows a reliable stabilization of the on-chip oscillator. After that, the watchdog toggles the clock supply back to the on-chip oscillator and releases the reset request. If no reset is applied in this moment the part will start program execution. If an external reset is active, however, the device will keep the reset state until also the external reset request disappears.

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

8.2.2 Fast Internal Reset after Power-On

The C541U can use the oscillator watchdog unit for a fast internal reset procedure after power-on.

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

In the C541U the oscillator watchdog unit avoids this situation. After power-on the oscillator watchdog's RC oscillator starts working within a very short start-up time (typ. less than 2 microseconds). In the following the watchdog circuitry detects a failure condition for the on-chip oscillator because this has not yet started (a failure is always recognized if the watchdog's RC oscillator runs faster than the on-chip oscillator). As long as this condition is valid the watchdog uses the RC oscillator output as clock source for the chip. This allows correct resetting of the part and brings all ports to the defined state. The delay time between power-on and correct reset state is max 34 µs. More details about the fast internal reset procedure after power-on are described in chapter 5 of this manual.

9 Power Saving Modes

The C541U provides two power saving modes :

- Idle mode
- Power down mode.

The functions of the power saving modes are controlled by bits which are located in the special function registers PCON und PCON1. PCON is located at address 87_H. PCON1 is located in the mapped SFR area and is accessed with RMAP=1. Bit RMAP is located in SFR SYSCON ($B1_H$) bit 4.

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

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

Special Function Register PCON (Address $87_{H)}$ **Reset Value : $X00X0000_B$**
Special Function Register PCON1 (Mapped Address $88_{H)}$ **Reset Value : $0XX0XXXX_B$**

Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
87_H	–	PDS	IDLS	–	GF1	GF0	PDE	IDLE	PCON
88_H	EWPD	–	–	WS	–	–	–	–	PCON1

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

Symbol	Function
PDS	**Power down start bit** The instruction that sets the PDS flag bit is the last instruction before entering the power down mode
IDLS	**Idle start bit** The instruction that sets the IDLS flag bit is the last instruction before entering the idle mode.
GF1	**General purpose flag**
GF0	**General purpose flag**
PDE	**Power down enable bit** When set, starting of the power down is enabled
IDLE	**Idle mode enable bit** When set, starting of the idle mode is enabled
EWPD	**External wake-up from powe down enable bit** Setting EWPD before entering power down mode, enables the external wake-up from power down mode capability via the pin P3.2/$\overline{INT0}$ or by the USB module.
WS	**Wake-up from software power down mode source select** WS = 0 : wake-up via pin P3.2/$\overline{INT0}$ selected (default after reset) WS = 1 : wake-up via USB bus selected
–	Reserved bits for future use.

9.1 Idle Mode

In the idle mode the main oscillator of the C541U continues to run, but the CPU is gated off from the clock signal. However, the interrupt system, the SSC, the USB module, and the timers with the exception of the watchdog timer are further provided with the clock. The CPU status is preserved in its entirety : the stack pointer, program counter, program status word, accumulator, and all other registers maintain their data during idle mode.

The reduction of power consumption, which can be achieved by this feature depends on the number of peripherals running. If all peripherals are disabled or stopped, the maximum power reduction can be achieved. This state is also the test condition for the idle mode I_{CC}.

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

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

Table 9-1
Status of External Pins During Idle and Power-Down Mode

Outputs	Last Instruction Executed from Internal Code Memory		Last Instruction Executed from External Code Memory	
	Idle	**Power-Down**	**Idle**	**Power-Down**
ALE	High	Low	High	Low
$\overline{\text{PSEN}}$	High	Low	High	Low
Port 0	Data	Data	Float	Float
Port 2	Data	Data	Address	Data
Port 1, 3	Data/alternate outputs	Data/last output	Data/alternate outputs	Data/last output

As in normal operation mode, the ports can be used as inputs during idle mode. Therefore, the timers can be used to count external events, and external interrupts will be detected.

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

9.1.1 Entering Idle Mode

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

PCON is not a bit-addressable register, so the above mentioned sequence for entering the idle mode is obtained by byte-handling instructions, as shown in the following example:

```
ORL     PCON,#00000001B     ;Set bit IDLE, bit IDLS must not be set
ORL     PCON,#00100000B     ;Set bit IDLS, bit IDLE must not be set
```

The instruction that sets bit IDLS is the last instruction executed before going into idle mode.

In idle mode, the USB module can be fully functional or can be switched off. If it is switched off in idle mode the following steps must be processed before entering the idle mode :

- USB module clock is switched off by software (resetting bit UCLK in SFR DCR)
- additionally in full speed mode : USB PLL is switched off (resetting bit PCLK in SFR DCR)

9.1.2 Exit from Idle Mode

There are two ways to terminate the idle mode:

- The idle mode can be terminated by activating any enabled interrupt. This interrupt will be serviced and normally the instruction to be executed following the RETI instruction will be the one following the instruction that sets the bit IDLS.
- The other way to terminate the idle mode, is a hardware reset. Since the oscillator is still running, the hardware reset must be held active only for two machine cycles for a complete reset.

After leaving the idle mode through e.g. an interrupt with a switched-off USB module, a well defined procedure must be executed foragain switching on the USB module :

- in full speed mode only USB PLL is switched on (setting bit PCLK in SFR DCR) and waiting 3 ms for PLL being locked
- USB module clock is switched on (setting bit UCLK in SFR DCR)

This switch off/on procedure assures a proper operation of the USB clock system. If the idle mode is terminated by a hardware reset, the USB module has to be reconfigured as defined for the hardware reset case.

9.2 Power Down Mode

In the power down mode, the RC osciillator and the on-chip oscillator which operates with the XTAL pins is stopped. Therefore, all functions of the microcontroller are stopped and only the contents of the on-chip RAM, XRAM and the SFR's are maintained. The port pins, which are controlled by their port latches, output the values that are held by their SFR's. The port pins which serve the alternate output functions show the values they had at the end of the last cycle of the instruction which initiated the power down mode. ALE and $\overline{\text{PSEN}}$ hold at logic low level (see **table 9-1**). The power down mode can be left either by an active reset signal or by a low signal at the P3.2/$\overline{\text{INT0}}$ pin or any activity on the USB bus. The USB module enters the suspend state when it detects no activity on the USB bus for more than 6 ms. The suspend state is left when bus activity is detected on the USB bus. Leaving the suspend state can (if selected and enabled) wake-up the power down mode.

Using reset to leave power down mode puts the microcontroller with its SFRs into the reset state. Using the $\overline{\text{INT0}}$ pin or USB bus for power down mode exit maintains the state of the SFRs, which has been frozen when power down mode is entered.

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

9.2.1 Entering Power Down Mode

The power down mode is entered by two consecutive instructions. The first instruction has to set the flag bit PDE (PCON.1) and must not set bit PDS (PCON.6), the following instruction has to set the start bit PDS (PCON.6) and must not set bit PDE (PCON.1). The hardware ensures that a concurrent setting of both bits, PDE and PDS, does not initiate the power down mode. Bits PDE and PDS will automatically be cleared after having been set and the value shown by reading one of these bits is always 0. This double instruction is implemented to minimize the chance of unintentionally entering the power down mode which could possibly ”freeze” the chip's activity in an undesired status. Important : the USB module must be switched off from the system clock prior to enabling the power down mode by software :

PCON is not a bit-addressable register, so the above mentioned sequence for entering the power-down mode is obtained by byte-handling instructions, as shown in the following example:

```
ANL     DCR,#11111101B      ;clear bit UCLK; USB clock is switched off
ANL     DCR,#11111110B      ;clear bit PCLK, stop PLL (required only in full speed mode)
ORL     PCON,#00000010B     ;set bit PDE, bit PDS must not be set
ORL     PCON,#01000000B     ;set bit PDS, bit PDE must not be set, enter power-down
```

The instruction that sets bit PDS is the last instruction executed before going into power down mode. When the double instruction sequence shown above is used and when bit EWPD in SFR PCON1 is 0, the power down mode can only be left by a reset operation.

If the wake-up from power down capability is required, its function must be enabled prior to executing the double instruction sequence shown above.

```
ORL     SYSCON,#00010000B   ;set RMAP
ORL     PCON1,#80H          ;enable wake-up from power-down by setting EWPD
                            ;80H = wake-up through pin P3.2/INT0
                            ;90H = wake-up through USB bus
ANL     SYSCON,#11101111B   ;reset RMAP (for future SFR accesses)
```

Note : Before entering the power down mode, the port latch of SFR P3.2 (P3.2/$\overline{\text{INT0}}$ pin) should contain a “1“ (pin operates as input). Otherwise, the wake-up sequence discussed in the next chapter will be started immediately when power down mode is entered.

If the wake-up from software power down mode through USB bus capability is selected, the USB receiver must be enabled in order to detect any activity on the USB bus lines. Therefore, bit RPWD in the USB device power down register DPWDR must be cleared before enering software power down mode.

The USB module enters the suspend state when it detects no activity on the USB bus for more than 6 ms.

9.2.2 Exit from Power Down Mode

If the power down mode is exit via a hardware reset, the microcontroller with its SFRs is put into the hardware reset state and the content of RAM and XRAM are not changed. The reset signal that terminates the power down mode also restarts the RC oscillator and the on-chip oscillatror. The reset operation should not be activated before V_{DD} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize (similar to power-on reset). The USB clock system must be cotrolled as described for the hardware reset in chapter 5.

Figure 9-1 shows the procedure which must is executed when the power down mode is left via the wake-up capability.

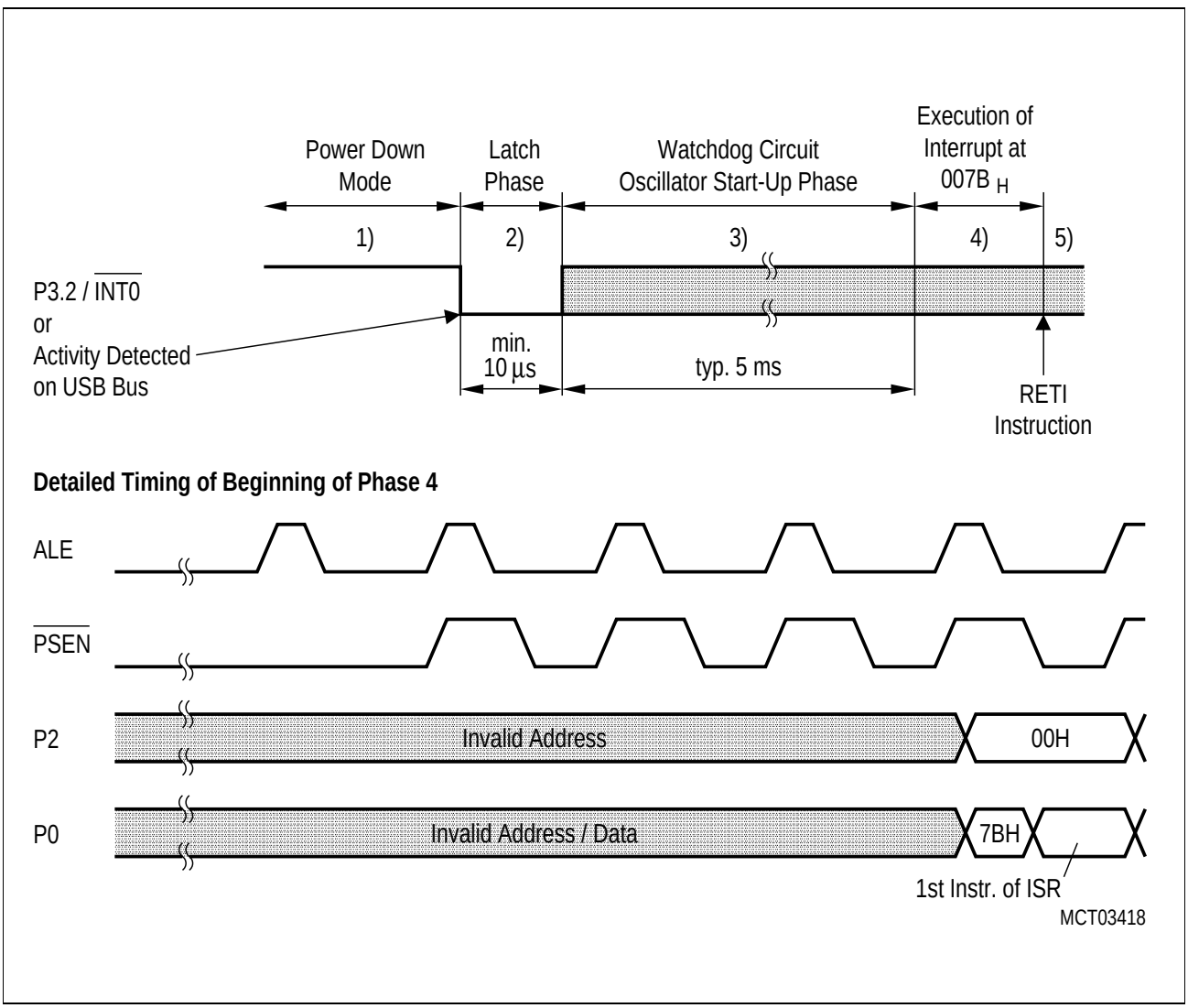


Figure 9-1
Wake-up from Power Down Mode Procedure

When the power-down mode wake-up capability has been enabled (bit EWPD in SFR PCON1 set) prior to entering power down mode, the power down mode can be exit either via P3.2/$\overline{INT0}$ or an activity on the USB bus.

9.2.2.1 Exit via Pin P3.2/$\overline{INT0}$

The following procedure :

1. In power down mode pin $\overline{INT0}$ must be held at high level.
2. Power down mode is left when $\overline{INT0}$ goes low. With $\overline{INT0}$ = low the internal RC oscillator is started. $\overline{INT0}$ is then latched by the RC oscillator clock signal. Therefore, $\overline{INT0}$ should be held at low level for at least 10 µs (latch phase). After this delay $\overline{INT0}$ can be set again to high level if required. Thereafter, the oscillator watchdog unit controls the wake-up procedure in its start-up phase.
3. The oscillator watchdog unit starts its operation. When the on-chip oscillator clock is detected for stable nominal frequency, the microcontroller further waits for a delay of typically 5 ms and then starts again with its operation initiating the power down wake-up interrupt. The interrupt address of the first instruction to be executed after wake-up is $007B_H$.
4. The clock system of the USB module must be setup again by software : USB PLL is switched on by setting bit PCLK in SFR DCR.(only required in full speed mode); Therafter, the PLL must be stabilized by waiting typically 3 ms. Now bit UCLK in SFR DCR can be set.
5. After the RETI instruction of the power down wake-up interrupt routine has been executed, the instruction which follows the initiating power down mode double instruction sequence will be executed. The peripheral units timer 0/1, SSC, and WDT are frozen until end of phase 4.

All interrupts of the C541U are disabled from phase 2) until the end of phase 4). Other Interrupts can be first handled after the RETI instruction of the wake-up interrupt routine.

Depending on the requirements, point 4 can also be executed in pahse 5) after the execution of the RETI instruction described in point 5 above.

9.2.2.2 Exit via UBS Bus

If the wake-up from software power down mode through USB bus capability has been selected, any activity on the USB bus causes the termination of the suspend state, triggers the watchdog unit and starts the wake-up procedure. After the start trigger by the USB bus activity, the actions 3. to 5. as described above are executed.

The wake-up trigger signal from the USB module can only be generated if the USB receiver circuitry was enabled in software power down mode.

10 OTP Memory Operation

The C541U contains a 8k byte one-time programmable (OTP) program memory. With the C541U fast programming cycles are achieved (1 byte in 100 µsec). Also several levels of OTP memory protection can be selected.

This chapter describes in detail the C541U programmimg interface.

10.1 Programming Configuration

For programming of the device, the C541U must be put into the programming mode. This typically is done not in-system but in a special programming hardware. In the programming mode the C541U operates as a slave device similar as an EPROM standalone memory device and must be controlled with address/data information, control lines, and an external 11.5V programming voltage.

In the programming mode port 0 provides the bidirectional data lines and port 2 is used for the multiplexed address inputs. The upper address information at port 2 is latched with the signal PALE. For basic programming mode selection the inputs RESET, $\overline{PSEN}$, $\overline{EA}/V_{PP}$, ALE, PMSEL1/0, and PSEL are used. Further, the inputs PMSEL1,0 are required to select the access types (e.g. program/verify data, write lock bits, ...) in the programming mode. In programming mode V_{DD}/V_{SS} and a clock signal at the XTAL pins must be applied to the C541U. The 11.5V external programming voltage is input through the $\overline{EA}/V_{PP}$ pin.

Figure 10-1 shows the pins of the C541U which are required for controlling of the OTP programming mode.

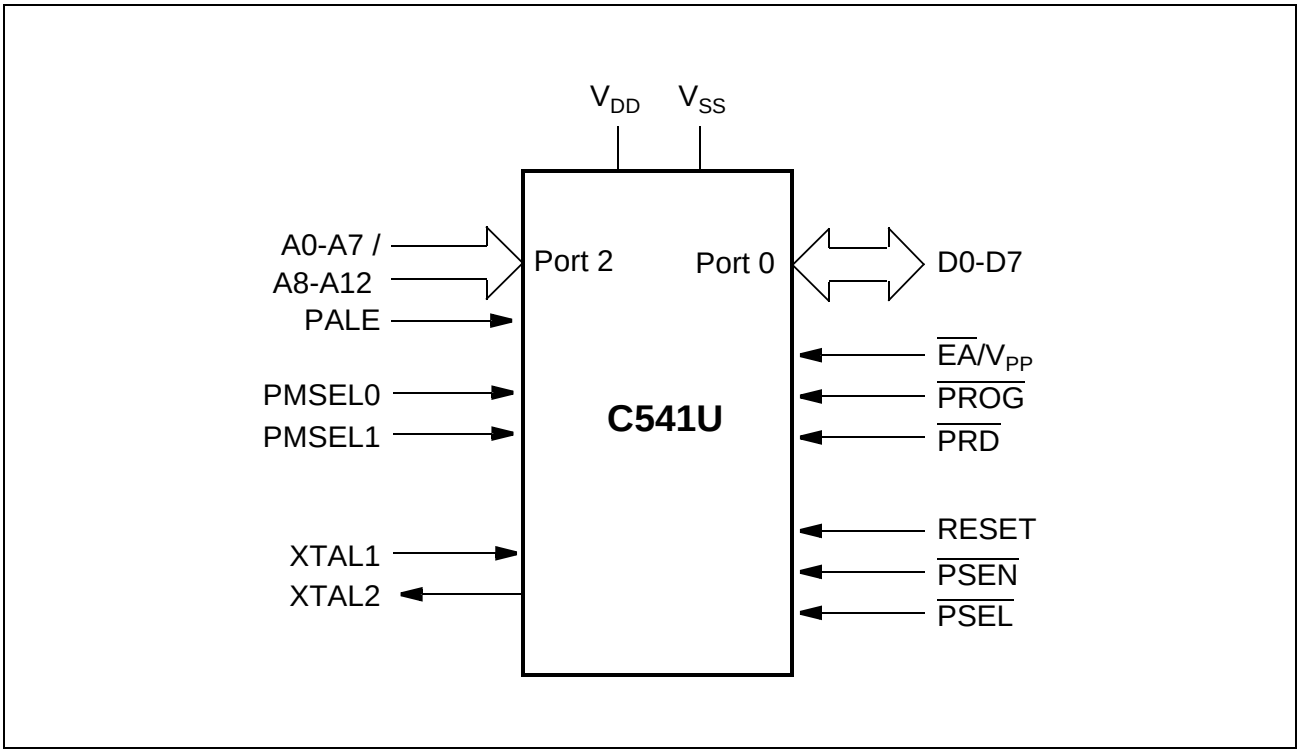


Figure 10-1
C541U Programming Mode Configuration

10.2 Pin Configuration

Figure 10-2 shows the detailed P-LCC-44 pin configuration of the C541U in programming mode.

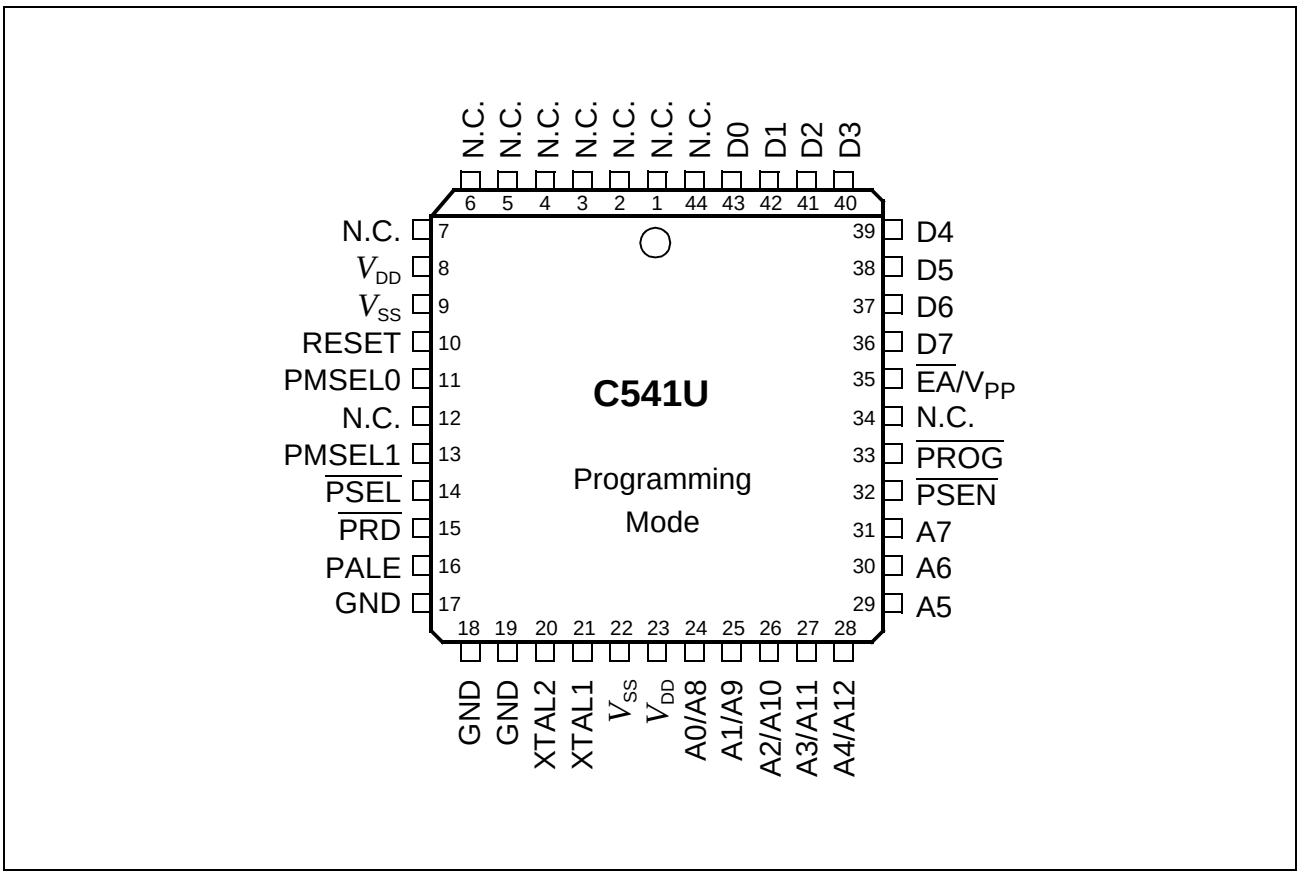


Figure 10-2
P-LCC-44 Pin Configuration of the C541U in Programming Mode (top view)

10.3 Pin Definitions

The following **table 10-1** contains the functional description of all C541U pins which are required for OTP memory programming

Table 10-1
Pin Definitions and Functions in Programming Mode

Symbol	Pin Numbers	I/O*)	Function
	P-LCC-44		
RESET	10	I	**Reset** This input must be at static “1“ (active) level during the whole programming mode.
PMSEL0 PMSEL1	11 13	I I	**Programming mode selection pins** These pins are used to select the different access modes in programming mode. PMSEL1,0 must satisfy a setup time to the rising edge of PALE. When the logic level of PMSEL1,0 is changed, PALE must be at low level.
$\overline{\text{PSEL}}$	14	I	**Basic programming mode select** This input is used for the basic programming mode selection and must be switched according **figure 10-3**.
$\overline{\text{PRD}}$	15	I	**Programming mode read strobe** This input is used for read access control for OTP memory read, version byte read, and lock bit read operations.
PALE	16	I	**Programming mode address latch enable** PALE is used to latch the high address lines. The high address lines must satisfy a setup and hold time to/from the falling edge of PALE. PALE must be at low level whenever the logic level of PMSEL1,0 is changed.
XTAL2	20	O	**XTAL2** Output of the inverting oscillator amplifier.
XTAL1	21	I	**XTAL1** Input to the oscillator amplifier.

PMSEL 1	PMSEL 0	Access Mode
0	0	Reserved
0	1	Read version bytes
1	0	Program/read lock bits
1	1	Program/read OTP memory byte

*) I = Input
O = Output

Table 10-1
Pin Definitions and Functions in Programming Mode (cont'd)

Symbol	Pin Numbers	I/O*)	Function
	P-LCC-44		
A0/A8 - A7	24 - 31	I	**Address lines** P2.0-7 are used as multiplexed address input lines A0-A7 and A8-A12. A8-A12 must be latched with PALE.
$\overline{\text{PSEN}}$	32	I	**$\overline{\text{Program store enable}}$** This input must be at static "0" level during the whole programming mode.
$\overline{\text{PROG}}$	33	I	**$\overline{\text{Programming mode write strobe}}$** This input is used in programming mode as a write strobe for OTP memory program and lock bit write operations During basic programming mode selection a low level must be applied to $\overline{\text{PROG}}$.
$\overline{\text{EA}}/V_{PP}$	35	I	**$\overline{\text{External Access}}$ / Programming voltage** This pin must be at 11.5 V (V_{PP}) voltage level during programming of an OTP memory byte or lock bit. During an OTP memory read operation this pin must be at high level (V_{IH}). This pin is also used for basic programming mode selection. At basic programming mode selection a low level must be applied to $\overline{\text{EA}}/V_{PP}$.
D0 - 7	43 - 36	I/O	**Data lines 0-7** During programming mode, data bytes are read or written from or to the C541U via the bidirectional D0-7 lines which are located at port 0.
V_{SS}	9, 22	–	**Circuit ground potential** must be applied to these pins in programming mode.
V_{DD}	8, 23	–	**Power supply terminal** must be applied to these pins in programming mode.
N.C.	1 - 7, 12,, 34, 44	–	**Not Connected** These pins should not be connected in programming mode.
GND	17 - 19	I	**Ground pins** In programming mode these pins must be connected to V_{IL} level.

*) I = Input
O = Output

10.4 Programming Mode Selection

The selection for the OTP programming mode can be separated into two different parts :

- Basic programming mode selection
- Access mode selection

With the basic programming mode selection the device is put into the mode in which it is possible to access the OTP memory through the programming interface logic. Further, after selection of the basic programming mode, OTP memory accesses are executed by using one of the access modes. These access modes are OTP memory byte program/read, version byte read, and program/read lock byte operations.

10.4.1 Basic Programming Mode Selection

The basic programming mode selection scheme is shown in **figure 10-3**.

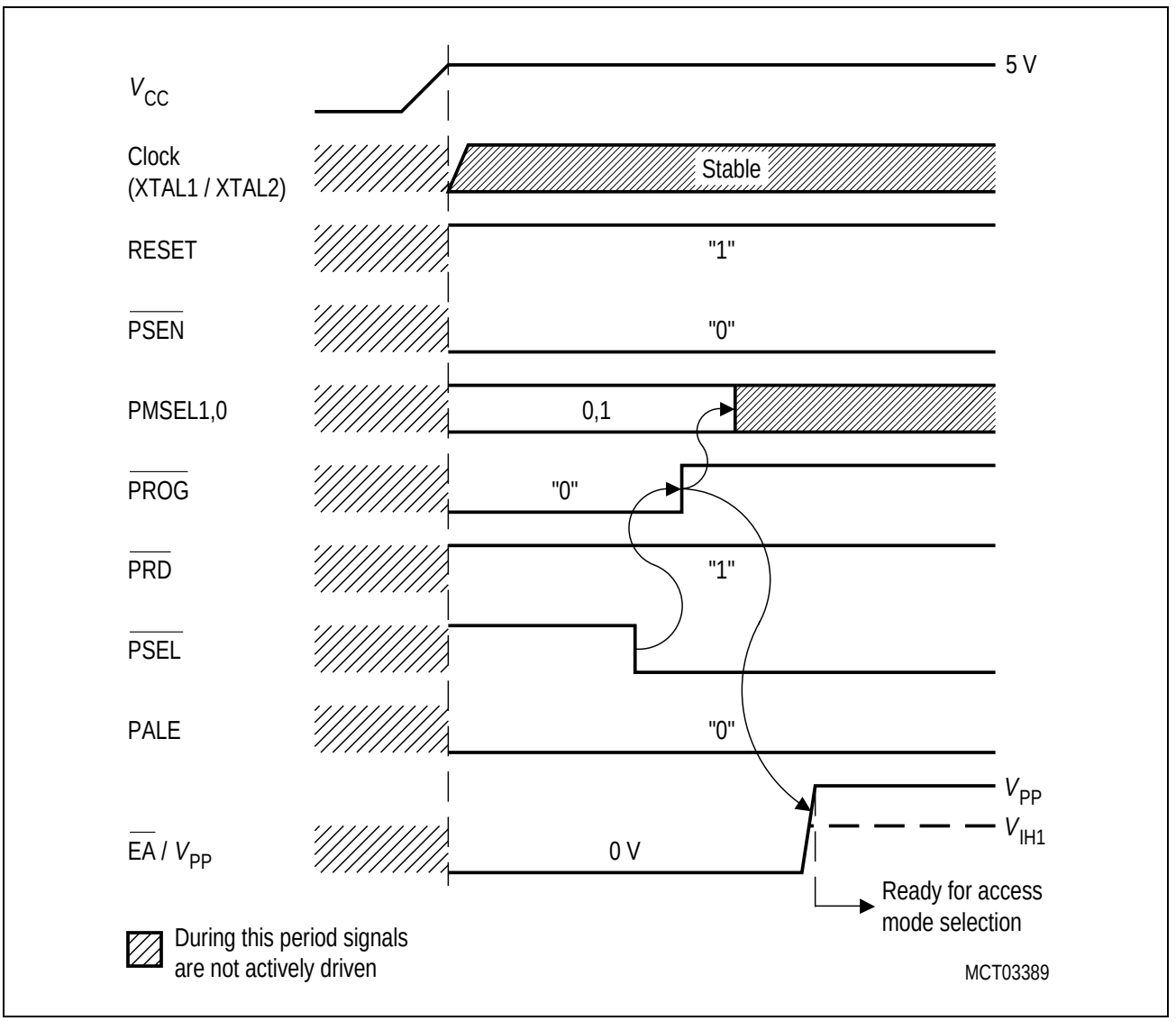


Figure 10-3
Basic Programming Mode Selection

The basic programming mode is selected by executing the following steps :

- With a stable V_{DD} a clock signal is applied to the XTAL pins; the RESET pin is set to “1“ level and the $\overline{PSEN}$ pin is set to “0“ level.
- $\overline{PROG}$, PALE, PMSEL1 and $\overline{EA}/V_{PP}$ are set to “0“ level; $\overline{PRD}$, $\overline{PSEL}$, and PMSEL0 are set to “1“ level.
- $\overline{PSEL}$ is set to from “1“ to “0“ level and thereafter $\overline{PROG}$ is switched to “1“ level.
- PMSEL1,0 can now be changed; after $\overline{EA}/V_{PP}$ has been set to V_{IH} high level or to V_{PP} the OTP memory is ready for access.

The pins RESET and $\overline{PSEN}$ must stay at “1“ respectively “0“ static signal level during the whole programming mode. With a falling edge of $\overline{PSEL}$ the logic state of ALE/$\overline{PROG}$ and V_{PP}($\overline{EA}$) is internally latched. These two signals are now used as programming write pulse signal ($\overline{PROG}$) and as programming voltage input pin V_{PP}. After the falling edge of $\overline{PSEL}$, $\overline{PSEL}$ must stay at “0“ state during all programming operations.

Note: If protection level 1 to 3 has been programmed (see section 10.6) and the programming mode has been left, it is no more possible to enter the programming mode !

10.4.2 OTP Memory Access Mode Selection

When the C541U has been put into the programming mode using the basic programming mode selection, several access modes of the OTP memory programming interface are available. The conditions for the different control signals of these access modes are listed in **table 10-2**.

Table 10-2
Access Modes Selection

Access Mode	$\overline{EA}$/ V_{PP}	$\overline{PROG}$	$\overline{PRD}$	PMSEL		Address (Port 2)	Data (Port 0)
				1	0		
Program OTP memory byte	V_{PP}	⊓	H	H	H	A0-7 A8-15	D0-7
Read OTP memory byte	V_{IH}	H	⊓				
Program OTP lock bits	V_{PP}	⊓	H	H	L	–	D1,D0 see **table 3**
Read OTP lock bits	V_{IH}	H	⊓				
Read OTP version byte	V_{IH}	H	⊓	L	H	Byte addr. of sign. byte	D0-7

The access modes from the table above are basically selected by setting the two PMSEL1,0 lines to the required logic level. The $\overline{PROG}$ and $\overline{PRD}$ signal are the write and read strobe signal. Data is transfered via port 0 and addresses are applied to port 2.

The following sections describes the details of the different access modes.

10.5 Program / Read OTP Memory Bytes

The program/read OTP memory byte access mode is defined by PMSEL1,0 = 1,1. It is initiated when the PMSEL1,0 = 1,1 is valid at the rising edge of PALE. With the falling edge of PALE the upper addresses A8-A12 of the 13-bit OTP memory address are latched. After A8-A12 has been latched, A0-A7 is put on the address bus (port 2). A0-A7 must be stable when $\overline{\text{PROG}}$ is low or $\overline{\text{PRD}}$ is low. If subsequent OTP address locations are accessed with constant address information at the high address lines A8-12, A8-A12 must only be latched once (page address mechanism).

Figure 10-4 shows a typical OTP memory programming cycle with a following OTP memory read operation. In this example A0-A12 of the read operation are identical to A8-A12 of the preceeding programming operation.

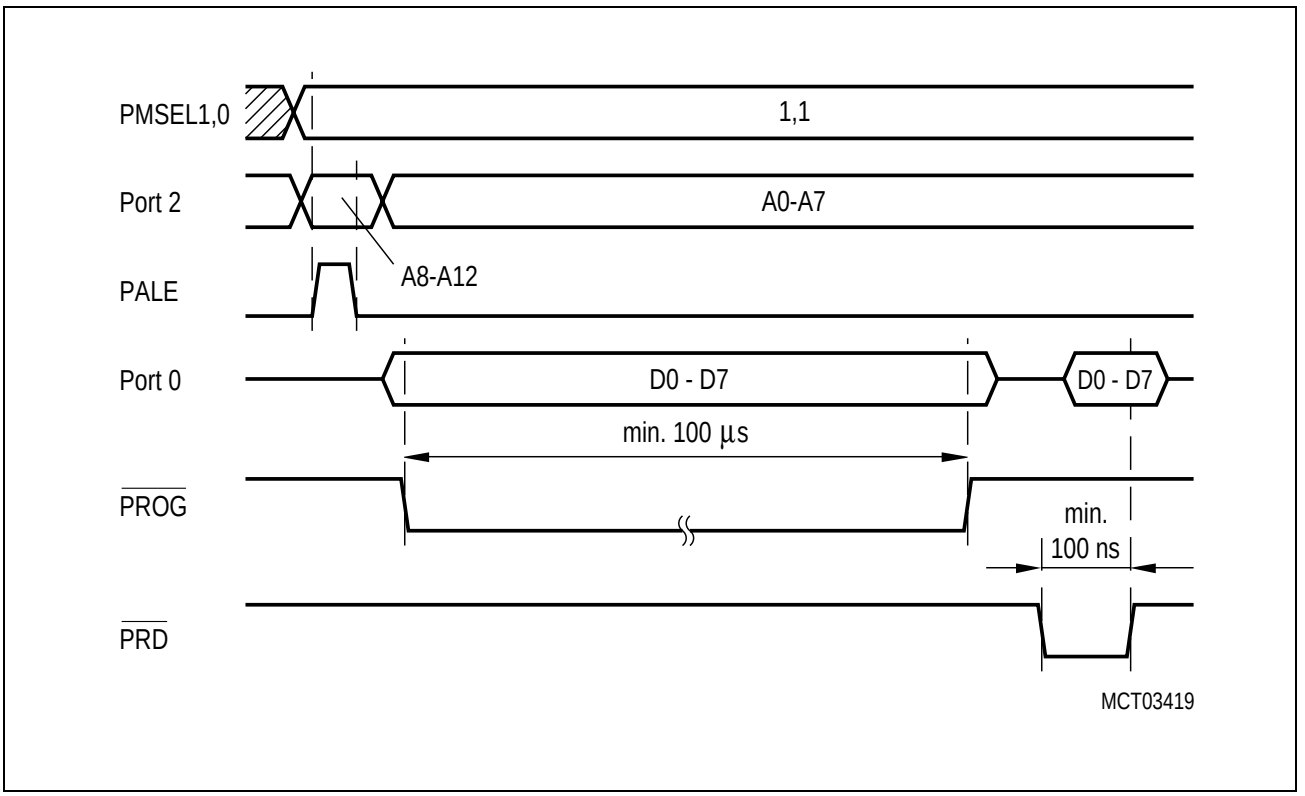


Figure 10-4
C541U Programming / VerifyOTP Memory Access Waveform

If the address lines A8-A12 must be updated, PALE must be activated for the latching of the new A8-A12 value. Control, address, and data information must only be switched when the $\overline{\text{PROG}}$ and $\overline{\text{PRD}}$ signals are at high level. The PALE high pulse must always be executed if a different access mode has been used prior to the actual access mode.

Figure 10-5 shows a waveform example of the program/read mode access for several OTP memory bytes. In this example OTP memory locations $3FD_H$ to 400_H are programmed. Thereafter, OTP memory locations 400_H and $3FD_H$ are read.

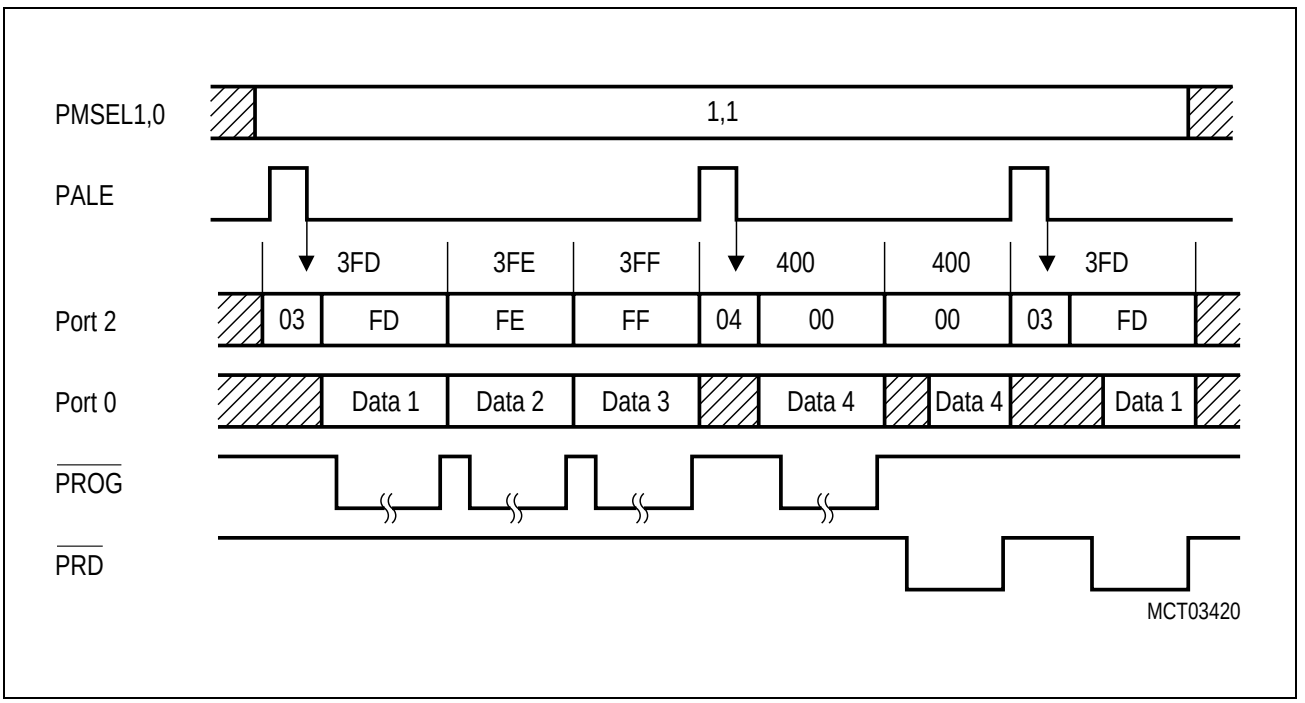


Figure 10-5
Typical OTP Memory Programming/Verify Access Waveform

10.6 Lock Bits Programming / Read

The C541U has two programmable lock bits which, when programmed according **tabie 10-3**, provide four levels of protection for the on-chip OTP code memory. The state of the lock bits can also be read.

Table 10-3
Lock Bit Protection Types

Lock Bits at D1,D0		Protection Level	Protection Type
D1	**D0**		
1	1	Level 0	The OTP lock feature is disabled. During normal operation of the C541U, the state of the $\overline{EA}$ pin is not latched on reset.
1	0	Level 1	During normal operation of the C541U, MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory. $\overline{EA}$ is sampled and latched on reset.An OTP memory read operation is only possible using the OTP verification mode for protection level 1. Further programming of the OTP memory is disabled (reprogramming security).
0	1	Level 2	Same as level 1, but also OTP memory read operation using OTP verification mode is disabled.
0	0	Level 3	Same as level 2; but additionally external code execution by setting $\overline{EA}$=low during normal operation of the C541U is no more possible. External code execution, which is initiated by an internal program (e.g. by an internal jump instruction above the OTP memory boundary), is still possible.

Note : A 1 means that the lock bit is unprogrammed. 0 means that lock bit is programmed.

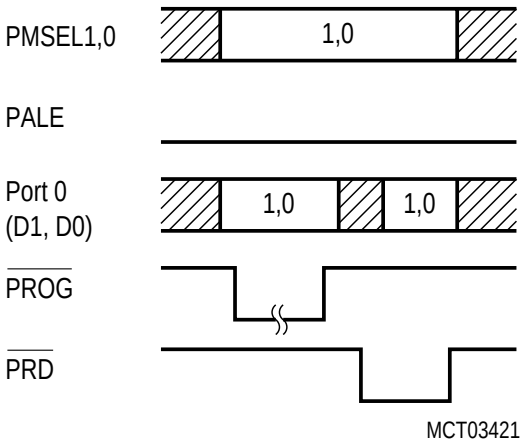
For a OTP verify operation at protection level 1, the C541U must be put into the OTP verification mode 2.

If a device is programmed with protection level 2 or 3, it is no more possible to verify the OTP content of a customer rejected (FAR) OTP device.

When a protection level has been activated by programming of the lock bits, the basic programming mode must be left for activation of the protection mechanisms. This means, after the activation of a protection level further OTP program/veriry operations are still possible if the basic programming mode is maintained.

The state of the lock bits can always be read if protection level 0 is selected. If protection level 1 to 3 has been programmed and the programming mode has been left, it is no more possible to enter the programming mode: In this case, also the lock bits cannot be read anymore.

Figure 10-6 shows the waveform of a lock bit write/read access. For a simple drawing, the $\overline{PROG}$ pulse is shortened. In reality, for Lock Bit programming, a 100µs $\overline{PROG}$ low puls must be applied.



The example shows the programming and reading of a protection level 1.

Figure 10-6
Write/Read Lock Bit Waveform

10.7 Access of Version Bytes

The C541U provides three version bytes at address locations FC_H, FD_H, and FE_H. The information stored in the version bytes, is defined by the mask of each microcontroller step, Therefore, the version bytes can be read but not written. The three version bytes hold information as manufacturer code, device type, and stepping code.

For reading of the version bytes the control lines must be used according **table 10-2** and **figure 10-7**. The address of the version byte must be applied at the port 1 address lines. PALE must not be activated.

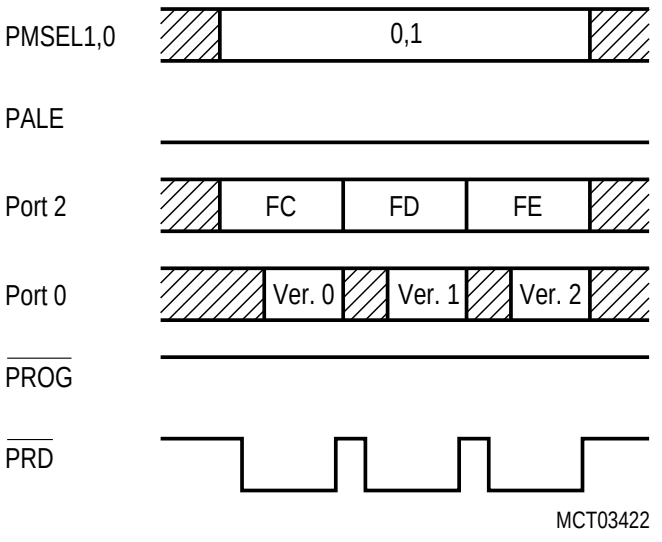


Figure 10-7
Read Version Byte(s) Waveform

Version bytes are typically used by programming systems for adapting the programming firmware to specifc device characteristics such as OTP size etc.

Note: The 3 version bytes are implemented in a way that they can be also read during normal program execution mode as a mapped register with bit RMAP in SFR SYSCON set. The addresses of the version bytes in normal mode and programming mode are identical and therefore they are located in the SFR address range.

The first step of the C541U will contain the following information at the version bytes :

Name	Address	Value
Version Byte 0	FC_H	$C5_H$
Version Byte 1	FD_H	$C1_H$
Version Byte 2	FE_H	01_H

Future steppings of the C541U will have a different version byte 2 (incremented value. version bytes 0 and 1 will remain unchanged for future steppings of the C541U.

10.8 OTP Verify with Protection Level 1

If the C541U OTP program memory is protected in protection level 1), an OTP verification as shown in **figure 10-8** is used to verify the content of the OTP. The detailed timing characteristics of this OTP verification mode is shown in the AC specifications (chapter 11).

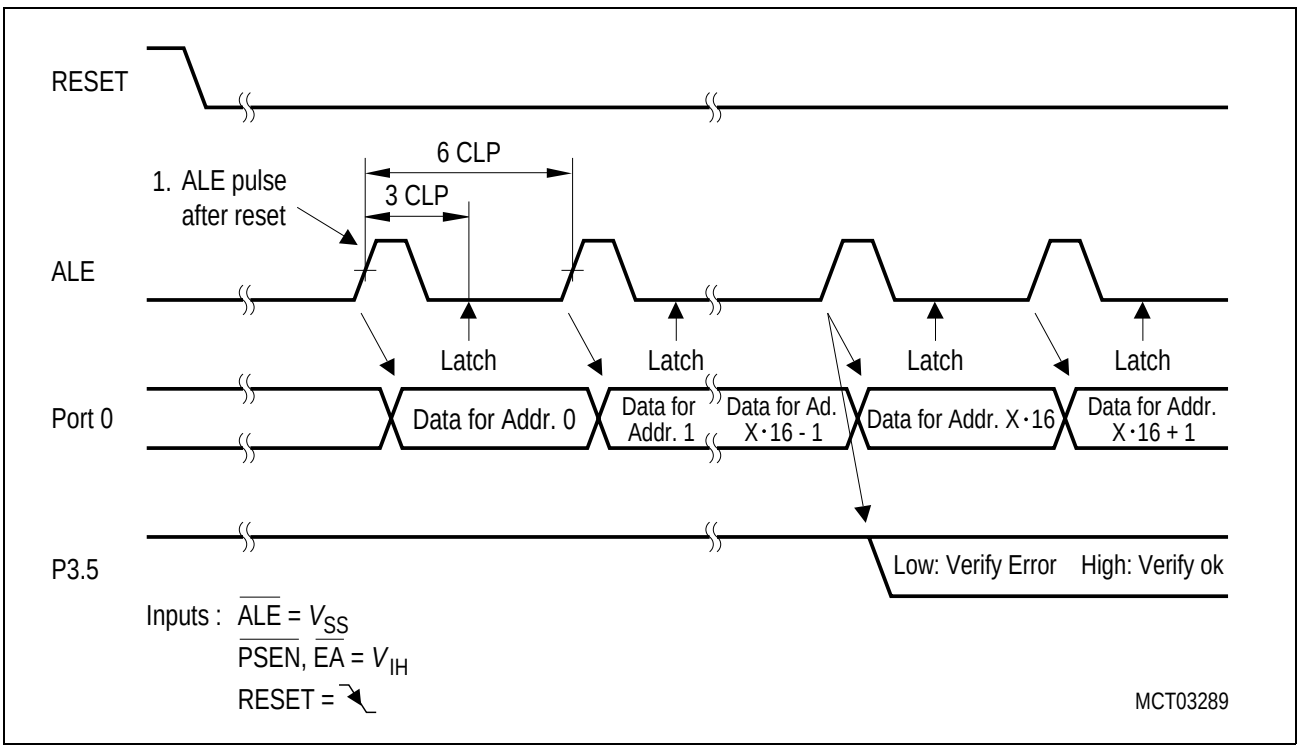


Figure 10-8
OTP Verification Mode Timing

The OTP verification mode is selected when the inputs $\overline{\text{PSEN}}$, $\overline{\text{EA}}$, and ALE are put to the specified logic levels. With RESET going inactive, the OTP verification mode sequence is started. The C541U outputs an ALE signal with a period of 3 CLPand expects data bytes at port 0. The data bytes at port 0 are assigned to the OTP addresses in the following way:

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

The C541U does not output any address information during the OTP verification mode. The first data byte to be verified is always the byte which is assigned to the internal OTP address 0000_H and must be put onto the data bus with the first ALE pulse after the falling edge of RESET. With each following ALE pulse the OTP address pointer is internally incremented and the expected data byte for the next OTP address must be delivered externally.

Between two ALE pulses the data at port 0 is latched (at 3 CLP after ALE rising edge) and compared internally with the OTP content of the actual address. If an verify error is detected, the error

condition is stored internally. After each 16th data byte the cumulated verify result (pass or fail) of the last 16 verify operations is output at P3.5. This means that P3.5 stays at static level (low for fail and high for pass) during the time when the following 16 bytes are checked. In OTP verification mode, the C541U must be provided with a system clock at the XTAL pins.

Figure 10-9 shows an application example of a external circuitry which allows to verify a protected OTP inside the. With RESET going inactive, the C541U starts the OTP verify sequence. Its ALE is clocking an 14-bit address counter. This counter generates the addresses for an external EPROM which is programmed with the content of the internal (protected) OTP. The verify detect logic typically displays the state of the verify error output P3.5. P3.5 can be latched with the falling edge of ALE.

When the last byte of the internal OTP has been handled, the C541U starts generating a $\overline{\text{PSEN}}$ signal. This signal or the CY signal of the address counter indicate to the verify detect logic the end of the internal OTP verification.

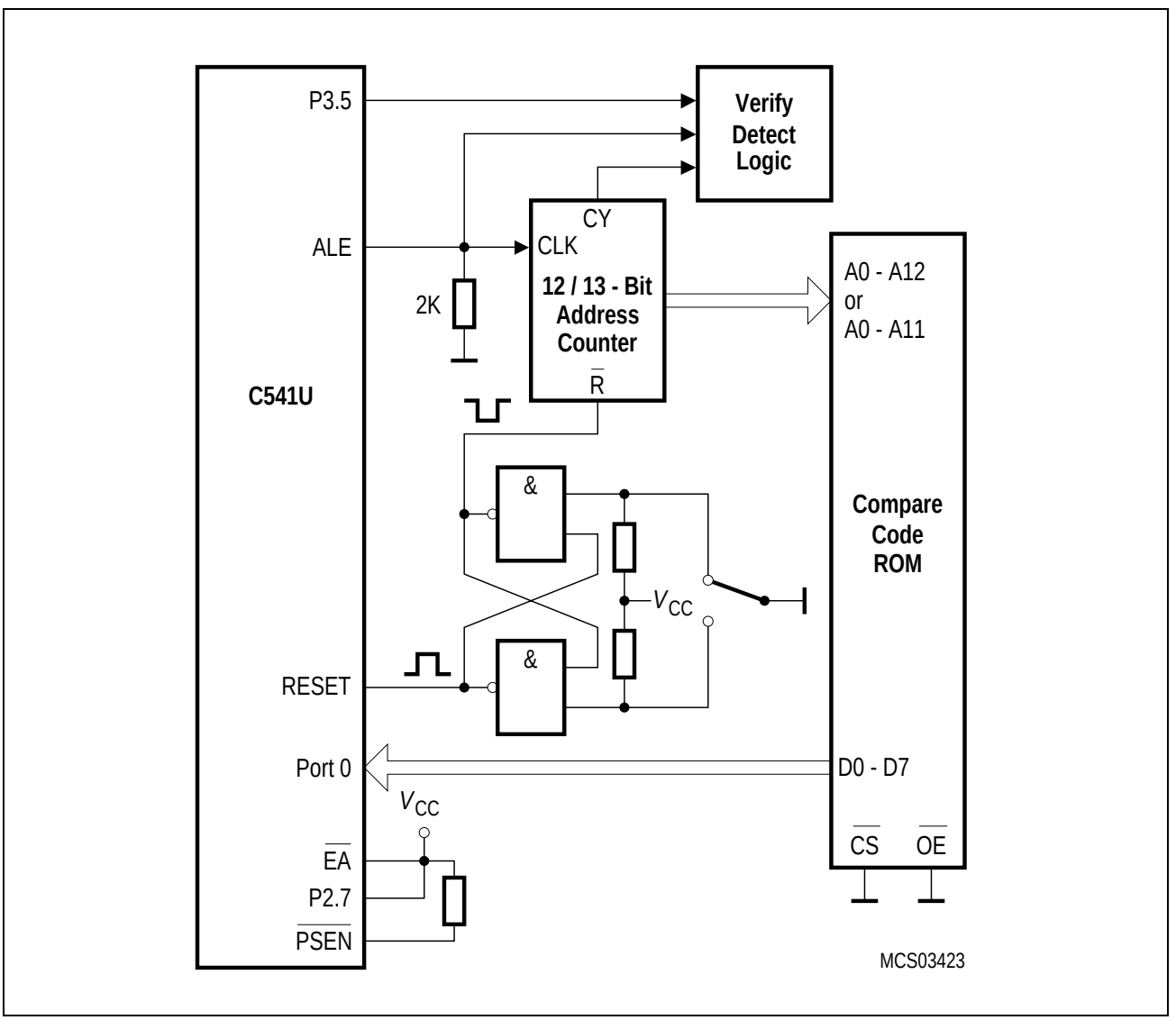


Figure 10-9
OTP Verification with Protection Level 1 - External Circuitry Example

11 Index

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

A

A06-A03 . . . . . 3-9, **6-74**
A16-A13 . . . . . 3-9, **6-74**
A26-A23 . . . . . 3-10, **6-74**
A36-A33 . . . . . 3-10, **6-74**
A46-A43 . . . . . 3-10, **6-74**
AC **2-3**, 3-7
ACC **2-2**, 3-4, 3-7
ACK0 3-9
ACK1 3-9
ACK2 3-10
ACK3 3-10
ACK4 3-10
ACKn **6-72**, 7-14
ADROFF. 3-5, 3-7, **6-57**
AIE0 3-9
AIE1 3-9
AIE2 3-10
AIE3 3-10
AIE4 3-10
AIEn **6-70**, 7-8
ALE signal 4-4
AO5-0. . . . . 3-7, **6-57**

B

B. **2-3**, 3-4, 3-8
Basic CPU timing 2-4
Block diagram. 2-1
BRS0 . . . . . 3-6, **6-30**
BRS1 . . . . . 3-6, **6-30**
BRS2 . . . . . 3-6, **6-30**

C

C/T 3-6, **6-18**
CBF0 3-9
CBF1 3-9
CBF2 3-10
CBF3 3-10
CBF4 3-10
CBFn **6-68**
CLREP0 3-9
CLREP1 3-9
CLREP2 3-10
CLREP3 3-10
CLREP4 3-10
CLREPn. **6-69**
CPHA. 3-6, **6-30**
CPOL 3-6, **6-30**
CPU
  Accumulator 2-2
  B register 2-3
  Basic timing 2-4
  Fetch/execute diagram. 2-5
  Functionality 2-2
  Program status word 2-2
  Stack pointer 2-3
CPU timing. 2-5
CY **2-3**, 3-7

D

DA 3-9, **6-59**
DADD. 3-7, **6-81**, **6-87**, 7-13
DADDIE 3-7, **6-81**, 7-7
DAI. 3-9, **6-64**, 7-12
DAIE 3-9, **6-62**, 7-6
DBM0. 3-9
DBM1. 3-9
DBM2. 3-10
DBM3. 3-10
DBM4. 3-10
DBMn. **6-67**
DCR 3-5, 3-9, **6-59**
DDI. 3-9, **6-64**, 7-12
DDIE 3-9, **6-62**, 7-6
DIER 3-5, 3-9, **6-62**, 7-6
DINIT 3-9, **6-60**
DIR0. 3-9
DIR1. 3-9
DIR2. 3-10
DIR3. 3-10
DIR4. 3-10
DIRn. **6-68**
DIRR 3-5, 3-9, **6-64**, 7-12
DNR0. 3-9
DNR1. 3-9
DNR2. 3-10
DNR3. 3-10
DNR4. 3-10
DNRIE0 3-9
DNRIE1 3-9
DNRIE2 3-10
DNRIE3 3-10
DNRIE4 3-10

DNRIEn **6-70**, 7-8
DNRn **6-72**, 7-14
DONE0 3-9
DONE1 3-9
DONE2 3-10
DONE3 3-10
DONE4 3-10
DONEn **6-69**
DPH 3-4, 3-6
DPL 3-4, 3-6
DPWDR 3-5, 3-9, **6-61**
DRVI 3-7, **6-58**, 7-13
DRVIE 3-9, **6-61**, 7-7

E

EA 3-6, **7-4**
EALE 3-7, **4-4**
Emulation concept 4-5
EOD0 3-9
EOD1 3-9
EOD2 3-10
EOD3 3-10
EOD4 3-10
EODIE0 3-9
EODIE1 3-9
EODIE2 3-10
EODIE3 3-10
EODIE4 3-10
EODIEn **6-71**, 7-8
EODn **6-72**, 7-14
EPBA0 3-9
EPBA1 3-9
EPBA2 3-10
EPBA3 3-10
EPBA4 3-10
EPBAn 3-5, **6-74**
EPBC0 3-9
EPBC1 3-9
EPBC2 3-10
EPBC3 3-10
EPBC4 3-10
EPBCn 3-5, **6-66**, 7-9
EPBS0 3-9
EPBS1 3-9
EPBS2 3-10
EPBS3 3-10
EPBS4 3-10
EPBSn 3-5, **6-68**
EPI4-0 . . . . . 3-7, **6-58**, 7-15
EPIE0 3-9
EPIE1 3-9
EPIE2 3-10
EPIE3 3-10
EPIE4 3-10
EPIEn 3-5, **6-70**, 7-8
EPIR0 3-9
EPIR1 3-9
EPIR2 3-10
EPIR3 3-10
EPIR4 3-10
EPIRn 3-5, **6-72**, 7-14
EPLEN0 3-9
EPLEN1 3-9
EPLEN2 3-10
EPLEN3 3-10
EPLEN4 3-10
EPLENn 3-5, **6-74**
EPS2-0 . . . . . 3-7, **6-55**
EPS7 . . . . . 3-7, **6-55**
EPSEL 3-5, 3-7, **6-55**
ESP0 3-9
ESP1 3-9
ESP2 3-10
ESP3 3-10
ESP4 3-10
ESPn **6-68**
ESSC 3-6, **7-5**
ET0 . . . . . 3-6, **7-4**
ET1 . . . . . 3-6, **7-4**
EUDI 3-6, **7-5**
EUEI 3-6, **7-5**
EWPD 3-6, **9-2**
EX0 . . . . . 3-6, **7-4**
EX1 . . . . . 3-6, **7-4**
Execution of instructions 2-4, 2-5
External bus interface . . . . . 4-1 to 4-4
ALE signal 4-4
ALE switch-off control 4-4
Overlapping of data/program memory 4-3
Program memory access 4-3
Program/data memory timing 4-2
PSEN signal 4-3
Role of P0 and P2 4-1

F

F0 . . . . . **2-3**, 3-7

F1 . . . . . **2-3**, 3-7
Fail save mechanisms 8-1–8-7
Fast power-on reset 5-3, 8-7
Features 1-2
FNR10-0 . . . . . 3-9, **6-65**
FNRH 3-5, 3-9, **6-65**
FNRL 3-5, 3-9, **6-65**
Functional units 1-1
Fundamental structure 2-1

G

GATE 3-6, **6-18**
GEPIE0 3-9
GEPIE1 3-9
GEPIE2 3-10
GEPIE3 3-10
GEPIE4 3-10
GEPIEn **6-66**, 7-9
GEPIR 3-5, 3-7, **6-58**, 7-15
GF0 . . . . . 3-6, **9-2**
GF1 . . . . . 3-6, **9-2**

H

Hardware reset 5-1

I

I/O ports . . . . . 6-1 to 6-14
I0ETF 3-6, **7-21**
I0ETR 3-6, **7-21**
I1ETF 3-6, **7-21**
I1ETR 3-6, **7-21**
IDLE 3-6, **9-2**
IDLS 3-6, **9-2**
IE0 . . . . . 3-6, **7-10**
IE1 . . . . . 3-6, **7-10**
IEN0 . . . . . 3-4, 3-6, **7-4**
IEN1 . . . . . 3-4, 3-6, **7-5**
INCE0 3-9
INCE1 3-9
INCE2 3-10
INCE3 3-10
INCE4 3-10
INCEn **6-67**
INT0 . . . . . 3-7, **7-20**
INT1 . . . . . 3-7, **7-20**
Interrupts 7-1–7-22
  Entry sequence timing 7-18
  External interrupts 7-20
  Handling procedure 7-18
  Registers . . . . . 7-4 to 7-16
  Request flags 7-10
  Response time 7-22
  Sources and vector addresses 7-19
IP0 . . . . . 3-4, 3-7, **7-16**
IP1 . . . . . 3-4, **7-16**
IT0 . . . . . 3-6, **7-10**
IT1 . . . . . 3-6, **7-10**
ITCON 3-4, 3-6, **7-21**

L

L06-L00 . . . . . 3-9, **6-74**
L16-L10 . . . . . 3-9, **6-74**
L26-L20 . . . . . 3-10, **6-74**
L36-L30 . . . . . 3-10, **6-74**
L46-L40 . . . . . 3-10, **6-74**
LED0 3-6
LED1 3-6
LED2 3-7
LEN3 - 0 . . . . . 3-7, **6-80**
Logic symbol 1-3
LOOPB 3-6, **6-33**
LSBSM 3-6, **6-33**

M

M0 . . . . . 3-6, **6-18**
M1 . . . . . 3-6, **6-18**
Memory organization 3-1
  Data memory 3-2
  General purpose registers 3-2
  Memory map 3-1
  Program memory 3-2
MSTR 3-6, **6-29**

N

NACK0 3-9
NACK1 3-9
NACK2 3-10
NACK3 3-10
NACK4 3-10
NACKn **6-72**, 7-14
NAIE0 3-9
NAIE1 3-9
NAIE2 3-10
NAIE3 3-10
NAIE4 3-10
NAIEn **6-70**, 7-8
NOD0 3-9
NOD1 3-9

NOD2 3-10
NOD3 3-10
NOD4 3-10
NODIE0 3-9
NODIE1 3-9
NODIE2 3-10
NODIE3 3-10
NODIE4 3-10
NODIEn **6-70**, 7-8
NODn **6-72**, 7-14

O

Oscillator operation . . . . . 5-6 to 5-8
External clock source 5-8
On-chip oscillator circuitry. 5-8
Recommended oscillator circuit 5-7
Oscillator watchdog . . . . . 8-5 to 8-7
Behaviour at reset. 5-3
Block diagram. 8-6
OTP memory operation . . . . . 10-1 to 10-13
Access mode selection. 10-6
Basic prog. mode selection. 10-5
Lock bit access 10-9
OTP memory protection levels 10-9
OTP protection
Level 1 verifiy timimg. 10-12
OTP verification example 10-13
Protection level 1. 10-12
Pin configuration
P-LCC-44 package 10-2
Pin definitions and functions . . 10-3, 10-4
Program/read operation 10-7, 10-8
Programming mode 10-1
Version byte access 10-11
OV **2-3**, 3-7
OWDS 3-7, **8-3**

P

P **2-3**, 3-7
P0 . . . . . 3-4, 3-6
P1 . . . . . 3-4, 3-6
P2 . . . . . 3-4, 3-6
P3 . . . . . 3-4, 3-7
PAGE0 3-9
PAGE1 3-9
PAGE2 3-10
PAGE3 3-10
PAGE4 3-10
PAGEn **6-74**
Parallel I/O . . . . . 6-1 to 6-14
PCLK 3-9, **6-60**
PCON 3-5, 3-6, **9-2**
PCON1 . . . . . 3-5, 3-6, **9-2**
PDE 3-6, **9-2**
PDS 3-6, **9-2**
Pin configuration 1-4
P-LCC-44 package. 1-4
Pin definitions and functions . . . . 1-5 to 1-8
Ports . . . . . 6-1 to 6-14
Alternate functions . . . . . 6-10 to 6-11
Port loading and interfacing 6-13
Port timing 6-12
Quasi-bidirectional port structure
Basic circuitry of port 1 to 3 6-3
Output driver circuitry 6-4
Port 0 circuitry. 6-8
Port 0/2 as address/data bus 6-9
SSC pins at port 1 6-6
Read-modify-write function 6-14
Power saving modes . . . . . 9-1 to 9-8
Behaviour of external pins 9-3
Idle mode . . . . . 9-3 to 9-4
Power down mode . . . . . 9-5 to 9-8
Entering 9-6
External wake-up timing 9-7
Functionality 9-5
Termination. 9-7
PSEN signal. 4-3
PSSC 3-7, **7-16**
PSW. **2-3**, 3-4, 3-7
PT0 . . . . . 3-7, **7-16**
PT1 . . . . . 3-7, **7-16**
PUDI 3-7, **7-16**
PUEI 3-7, **7-16**
PX0 . . . . . 3-7, **7-16**
PX1 . . . . . 3-7, **7-16**

R

RD 3-7
Reset . . . . . 5-1 to 5-5
Fast power-on reset 5-3
Hardware reset timing 5-5
of USB module 5-2
Power-on reset timing 5-4
Reset circuitries 5-2
RLE0 3-9
RLE1 3-9

RLE2 3-10
RLE3 3-10
RLE4 3-10
RLEIE0 3-9
RLEIE1 3-9
RLEIE2 3-10
RLEIE3 3-10
RLEIE4 3-10
RLEIEn **6-70**, 7-8
RLEn **6-72**, 7-14
RMAP **3-3**, 3-7
RPWD 3-7, 3-9, **6-81**
RS0 . . . . . **2-3**, 3-7
RS1 . . . . . **2-3**, 3-7
RSM 3-9, **6-60**

S

SBI 3-9, **6-64**, 7-12
SBIE 3-9, **6-62**, 7-6
SCEN 3-6, **6-29**
SCF 3-4, 3-6, **6-32**, 7-11
SCIEN 3-4, 3-7, **6-31**, 7-5
SCLK 3-6
SE0I 3-9, **6-64**, 7-12
SE0IE 3-9, **6-62**, 7-6
SEI 3-9, **6-64**, 7-12
SEIE 3-9, **6-62**, 7-6
SETRD0 3-9
SETRD1 3-9
SETRD2 3-10
SETRD3 3-10
SETRD4 3-10
SETRDn **6-68**
SETWR0 3-9
SETWR1 3-9
SETWR2 3-10
SETWR3 3-10
SETWR4 3-10
SETWRn **6-69**
SLS 3-6
SOD0 3-9
SOD1 3-9
SOD2 3-10
SOD3 3-10
SOD4 3-10
SODIE0 3-9
SODIE1 3-9
SODIE2 3-10
SODIE3 3-10
SODIE4 3-10
SODIEn **6-71**, 7-8
SODn **6-73**, 7-14
SOFDE0 3-9
SOFDE1 3-9
SOFDE2 3-10
SOFDE3 3-10
SOFDE4 3-10
SOFDEn **6-66**
SOFI 3-9, **6-64**, 7-12
SOFIE 3-9, **6-63**, 7-6
SP **2-3**, 3-4, 3-6
Special Function Registers 3-2
  Access with RMAP 3-3
  Table - address ordered . . . . . 3-6 to 3-8
  Table - functional order . . . . . 3-3 to 3-4
SPEED 3-9, **6-59**
SRB 3-4, 3-6, **6-33**
SRI 3-6
SSC interface . . . . . 6-23 to 6-33
  Baudrate generation 6-25
  Block diagram 6-23
  General operation 6-24
  Master mode timing 6-27
  Master/slave mode 6-26
  Registers . . . . . 6-29 to 6-33
  Slave mode timing 6-28
  Write collision detection 6-25
SSCCON 3-4, 3-6, **6-29**
SSCMOD 3-4, 3-6, **6-33**
STALL0 3-9
STALL1 3-9
STALL2 3-10
STALL3 3-10
STALL4 3-10
STALLn **6-66**
STB 3-4, 3-6, **6-33**
STI 3-9, **6-64**, 7-12
STIE 3-9, **6-62**, 7-6
STO 3-6
SUI 3-9, **6-64**, 7-12
SUIE 3-9, **6-63**, 7-6
SUSP 3-7, 3-9, **6-59**, **6-81**, 7-13
SUSPIE 3-7, **6-81**, 7-7
SWDT 3-7, **8-3**
SWR 3-9, **6-59**

SYSCON **3-3**, 3-4, 3-7, 4-4

T

T0 3-7
T1 3-7
TC 3-6, **6-32**, 7-11
TCEN 3-7, **6-31**, 7-5
TCON 3-4, 3-6, **6-17**, 7-10
TEN 3-6, **6-29**
TF0 . . . . . 3-6, **6-17**, 7-10
TF1 . . . . . 3-6, **6-17**, 7-10
TH0 . . . . . 3-4, 3-6, **6-16**
TH1 . . . . . 3-4, 3-6, **6-16**
Timer/counter 6-15
Timer/counter 0 and 1 . . . . . 6-15 to 6-22
Mode 0, 13-bit timer/counter 6-19
Mode 1, 16-bit timer/counter 6-20
Mode 2, 8-bit rel. timer/counter . . . 6-21
Mode 3, two 8-bit timer/counter . . . 6-22
Registers . . . . . 6-16 to 6-18
TL0 . . . . . 3-4, 3-6, **6-16**
TL1 . . . . . 3-4, 3-6, **6-16**
TMOD 3-4, 3-6, **6-18**
TPWD 3-7, 3-9, 6-61, **6-81**
TR0 . . . . . 3-6, **6-17**
TR1 . . . . . 3-6, **6-17**
TRIO 3-6, **6-33**
TYPE3 - 0 . . . . . 3-7, **6-78**

U

UBF0 3-9
UBF1 3-9
UBF2 3-10
UBF3 3-10
UBF4 3-10
UBFn **6-68**
UCLK 3-9, **6-60**
USB module . . . . . 6-34 to 6-86
Block diagram 6-34
Control transfer 6-53
Detach/attach detection 6-87
Detection of connected devices 6-86
Device registers . . . . . 6-59 to 6-65
Endpoint registers . . . . . 6-66 to 6-74
Global registers . . . . . 6-55 to 6-58
Initialization (full-speed) 6-51
Low-speed mode 6-75
Interrupts 6-83
Transfer Modes 6-76
Memory buffer address generation . 6-50
Memory buffer modes . . . . . 6-36 to 6-48
Double buffer mode . . . . . 6-42 to 6-48
Single buffer mode . . . . . 6-37 to 6-41
Memory buffer organization 6-49
On-chip USB transceiver 6-84
Register set . . . . . 6-54 to 6-74
Transfer modes 6-35
USBDCR 3-5, 3-7, **6-78**
USBDRn 3-5, 3-7, 3-8, **6-82**
USBPWD 3-5, 3-7, **6-81**
USBVAL 3-5, 3-7, **6-56**

V

Version bytes 10-11
Version registers 10-11
VR0 . . . . . 3-4, 3-8, **10-11**
VR1 . . . . . 3-4, 3-8, **10-11**
VR2 . . . . . 3-4, 3-8, **10-11**

W

Watchdog timer . . . . . 8-1 to 8-4
Block diagram 8-1
Control/status flags 8-3
Input clock selection 8-2
Refreshing of the WDT 8-4
Reset operation 8-4
Starting of the WDT 8-4
Time-out periods 8-2
WCEN 3-7, **6-31**, 7-5
WCOL 3-6, **6-32**, 7-11
WDCON 3-4, 3-7, **8-3**
WDT 3-7, **8-3**
WDTPSEL 3-6, **8-2**
WDTREL 3-4, 3-6, **8-2**
WDTS 3-7, **8-3**
WR 3-7
WS 3-6, **9-2**