

**SIEMENS**

## ICs for Consumer Electronics

8-Bit Microcontroller, ROMLESS

SDA 30C163-2

Advanced Data Sheet 05.94

■ 8235605 0080563 447 ■

<b>SDA 30C163-2</b>	
<b>Revision History:</b>	<b>Original Version: 05.94</b>
Previous Releases:	
Page	Subjects (changes since last revision)

### Data Classification

### Maximum Ratings

Maximum ratings are absolute ratings; exceeding only one of these values may cause irreversible damage to the integrated circuit.

### Characteristics

The listed characteristics are ensured over the operating range of the integrated circuit. Typical characteristics specify mean values expected over the production spread. If not otherwise specified, typical characteristics apply at  $T_A = 25^\circ\text{C}$  and the given supply voltage.

### Operating Range

In the operating range the functions given in the circuit description are fulfilled.

For detailed technical information about "**Processing Guidelines**" and "**Quality Assurance**" for ICs, see our "**Short Form Catalog**".

### Edition 05.94

This edition was realized using the software system FrameMaker®

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

© Siemens AG 1993. All Rights Reserved.

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.

Contents	Page
<b>Introduction</b> .....	5
<b>1 Features</b> .....	7
1.1 Pin Definitions and Functions .....	10
<b>2 Functional Description</b> .....	13
2.1 Architecture .....	13
2.1.1 CPU-Hardware .....	13
2.1.2 CPU-Timing .....	16
2.1.3 Addressing Modes .....	17
2.2 Memory Organization .....	18
2.2.1 External Program Memory .....	18
2.2.2 Internal Data RAM .....	24
2.2.3 Special Function Registers .....	25
2.3 Interrupt System .....	31
2.3.1 Interrupt Sources .....	31
2.3.2 Interrupt Control .....	31
2.3.3 Interrupt Nesting .....	36
2.3.4 External Interrupts .....	36
2.3.5 Interrupt Task Function .....	37
2.3.6 Response Time .....	38
2.4 Processor Reset and Initialization .....	39
2.5 Ports and I/O-Pins .....	41
2.6 General Purpose Timers/Counters .....	43
2.7 Watchdog Timer .....	49
2.8 Serial Interface .....	51
2.8.1 Multiprocessor Communication .....	53
2.8.2 Baud Rates .....	54
2.8.3 More about Mode 0 .....	56
2.8.4 More about Mode 1 .....	57
2.8.5 More about Mode 2 and 3 .....	58
2.9 Pulse Width Modulation Unit .....	67
2.10 Analog Digital Converter .....	72
2.11 Advanced Function Register .....	74
2.12 Instruction Set .....	75
2.12.1 Notes on Data Addressing Modes .....	75
2.12.2 Notes on Program Addressing Modes .....	75
2.12.3 Instruction Set Description .....	76
2.12.4 Instruction Opcodes in Hexadecimal Order .....	80
2.13 Operational Restrictions .....	87

Contents (cont'd)		Page
<b>3</b>	<b>Electrical Characteristics</b> .....	89
3.1	Absolute Maximum Ratings .....	89
3.2	DC-Characteristics .....	89
3.3	AC-Characteristics .....	91
<b>4</b>	<b>Applications</b> .....	94
<b>5</b>	<b>Package Outlines</b> .....	95

## Introduction

The SDA 30C163-2, a derivative of the SAB 8051, is a member of a family of single-chip computers, in which the emphasis is no longer placed on purely numeric computational performance, but on application-specific controller functions.

Architecture and instruction set are based upon that of the 8051 microcomputer. Like the 8051 it has many features which increase programming ease; extended internal data memory-space, variable manipulation in internal data memory, free stack location in data RAM, 4 register banks, special function registers, memory mapped I/O, individually addressable bits and a Boolean processor give the programmer the ability to improve the power of software development. Numerical problems can be processed with binary as well as with BCD-arithmetic. The many bit handling instructions also contribute to the computer's efficiency as a controller. Extended memory is controlled by an 8-bit data- and a 16-bit address bus without any additional devices such as latches or logic elements, even when all 512 K of the program address space is used. All this leads, in suitable applications, to a reduction in the peripheral hardware and to a simplification of the software and thus to reduced development and component costs. The controller, specially developed for entertainment electronic applications, can also be recommended where both lowest component costs and a large production volume are prime requirements.

The SDA 30C163-2 contains a 1024 + 256-byte data memory (RAM), two independent 16-bit timers/counters and a five-source, two-priority-level, nested interrupt structure, on-chip oscillator and clock circuits. The 34 digital I/O-lines include four 8-bit ports (P1 and P3 contain I/O-lines with multifunction options) and the serial interface with data and clock lines. The serial I/O-interface is compatible to the 8051 serial port. Included in the multifunction port P3 are two interrupt inputs and two counter inputs.

The second multifunction port consists of port P1, which alternatively can be used as up to eight independent pulse width modulated output channels (PWM). Controlled via special function registers, the PWM-circuitry provides flexibility in time resolution and system configuration.

Specially the realization of D/A-outputs using pulse width modulation will be a cost saving advantage in analog applications.

The internal ADC is a 8-bit, four channel converter. The input channels are P20 to P23, the analog supply are pins  $V_{AREF}$  and  $V_{AGND}$ .

Port 4 can be used as a standard port or as memory extension address bits.

Increased system reliability can be achieved by activating the integrated watchdog timer.

Efficient use of program memory results from an instruction set consisting of 49 single-byte, 46 two-byte and 16 three-byte instructions. When using a 16-MHz crystal, 64 instructions execute in 0.75  $\mu$ s and 45 instructions execute in 1.5  $\mu$ s. The remaining instructions (multiple and divide) require only 3  $\mu$ s. The number of bytes in each instruction and the number of oscillator periods required for execution are listed in the Instruction Set in chapter 2.11.3.

Based on the SDA 30C162, the SDA 30C163-2 comprises double stack size for the extension memory (32 byte) and seven additional data pointer registers.

PAGE(S) INTENTIONALLY BLANK

((VAKAT))

6

## 8-Bit Microcontroller ROMLESS

SDA 30C163-2

### Preliminary Data

CMOS IC

#### 1 Features

##### ● SAB 8051 Architecture

- On-chip oscillator and clock circuits
- Binary or decimal arithmetic
- Signed-overflow detection and parity computation
- Integrated Boolean processor for control applications
- Full depth stack for subroutine return linkage and data storage
- Two priority level, nested interrupt structure
- 16-MHz oscillator frequency, 0.75  $\mu$ s instruction cycle
- 8 data pointer registers

##### ● Serial Interface

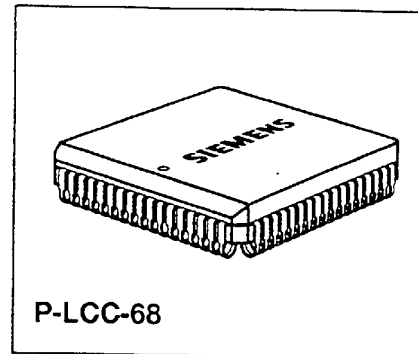
- Full duplex UART-interface

##### ● On-Chip RAM

- Direct byte and bit addressability
- Four register banks
- 256 bytes of data memory, including 128 user-defined software flags
- 1024 bytes of data memory accessible with MOVX-instructions

##### ● External Program Memory Interface

- 512 Kbytes of program memory may be addressed by a 8-bit data bus and a 16 + 3-bit address bus
- Extension stack depth 32 byte



● **34 Bidirectional I/O-Lines**

- Two 8-bit ports, one comprising up to eight programmable D/A-outputs
- One 8-bit multifunction port
- One 8-bit port with open drain output
- One 2-bit port with optional memory extension function

● **Pulse Width Modulation Unit**

- Up to eight programmable PWM-output channels for low cost digital-to-analog conversion

● **Timers**

- Two 16-bit general purpose timers/event counters
- Watchdog timer

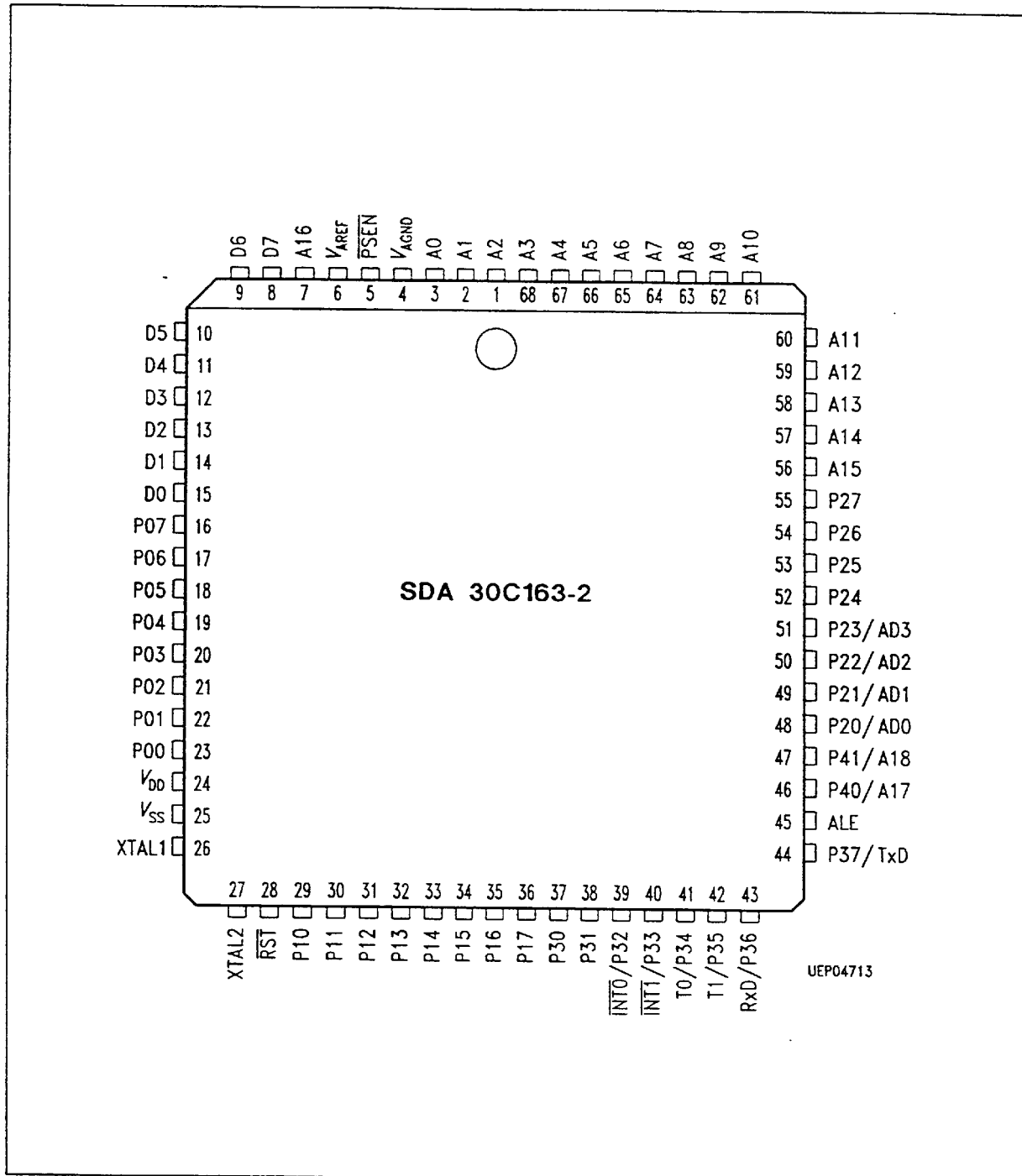
● **Analog-to-Digital Converter**

- Four multiplexed input channels to 8-bit resolution

Type	Ordering Code	Package
SDA 30C163-2	Q????-????	P-LCC-68 (SMD)



## Pin Configuration (top view)



## 1.1 Pin Definitions and Functions

Pin No.	Symbol	Input (I) Output (O) Supply (S)	Function
23	P0.0	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; in this state they can be used as high-impedance inputs.
22	P0.1	I/O	
21	P0.2	I/O	
20	P0.3	I/O	
19	P0.4	I/O	
18	P0.5	I/O	
17	P0.6	I/O	
16	P0.7	I/O	
29	P1.0	I/O	Port 1 is an 8-bit bidirectional I/O-port with internal pullup resistors. Port 1 pins that have 1 s written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. These eight bits also contain the output channels of the pulse width modulation unit. The secondary functions are assigned to the pins of port 1 as follows:  PWM <sub>i</sub> (P1.i): output of PWM-channel <i>i</i> ( <i>i</i> = 0, ..., 7).
30	P1.1	I/O	
31	P1.2	I/O	
32	P1.3	I/O	
33	P1.4	I/O	
34	P1.5	I/O	
35	P1.6	I/O	
36	P1.7	I/O	
48	P2.0	I	Port 2 is a multifunction port with P2.0 – P2.3 working as digital or analog inputs. Port bits P2.4 – P2.7 are bidirectional I/O-lines with internal pullup resistors.
49	P2.1	I	
50	P2.2	I	
51	P2.3	I	
52	P2.4	I/O	
53	P2.5	I/O	
54	P2.6	I/O	
55	P2.7	I/O	
37	P3.0	I/O	Port 3 is an 8-bit bidirectional I/O-port with internal pullup resistors. Port 3 pins that have 1 s written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. It also contains the interrupt, timer and serial port pins. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate.  The secondary functions are assigned to the pins of port 3, as follows:  <ul style="list-style-type: none"> <li>– <math>\overline{\text{INT0}}</math> (P3.2) : interrupt 0 input/timer 0 gate control input</li> <li>– <math>\overline{\text{INT1}}</math> (P3.3) : interrupt 1 input/timer 1 gate control input</li> <li>– T0 (P3.4) : counter 0 input</li> <li>– T1 (P3.5) : counter 1 input</li> <li>– RxD (P3.6) : serial port receive line</li> <li>– TxD (P3.7) : serial port transmit line.</li> </ul>
38	P3.1	I/O	
39	P3.2	I/O	
40	P3.3	I/O	
41	P3.4	I/O	
42	P3.5	I/O	
43	P3.6	I/O	
44	P3.7	I/O	

## Pin Definitions and Functions (cont'd)

Pin No.	Symbol	Input (I) Output (O) Supply (S)	Function
46 47	P4.0 P4.1	I/O I/O	Alternative outputs for port 2 quasi-bidirectional I/O or address bits A17/A18 for memory extension.
27	XTAL2	O	Output of the inverting oscillator amplifier. To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left open.
26	XTAL1	I	Input to the inverting oscillator amplifier.
28	$\overline{\text{RST}}$	I	A low level on this pin resets the processor.
24 25	$V_{\text{DD}}$ $V_{\text{SS}}$	S S	Power supply voltage Ground (0 V)
6 4	$V_{\text{AREF}}$ $V_{\text{AGND}}$	S S	Analog reference voltage Analog ground
5 45	$\overline{\text{PSEN}}$ ALE		Program Store Enable Address Latch Enable
3 2 1 68 67 66 65 64 63 62 61 60 59 58 57 56 7	A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16		Address bus for external memory
15 14 13 12 11 10 9 8	D0 D1 D2 D3 D4 D5 D6 D7		Data bus for external memory

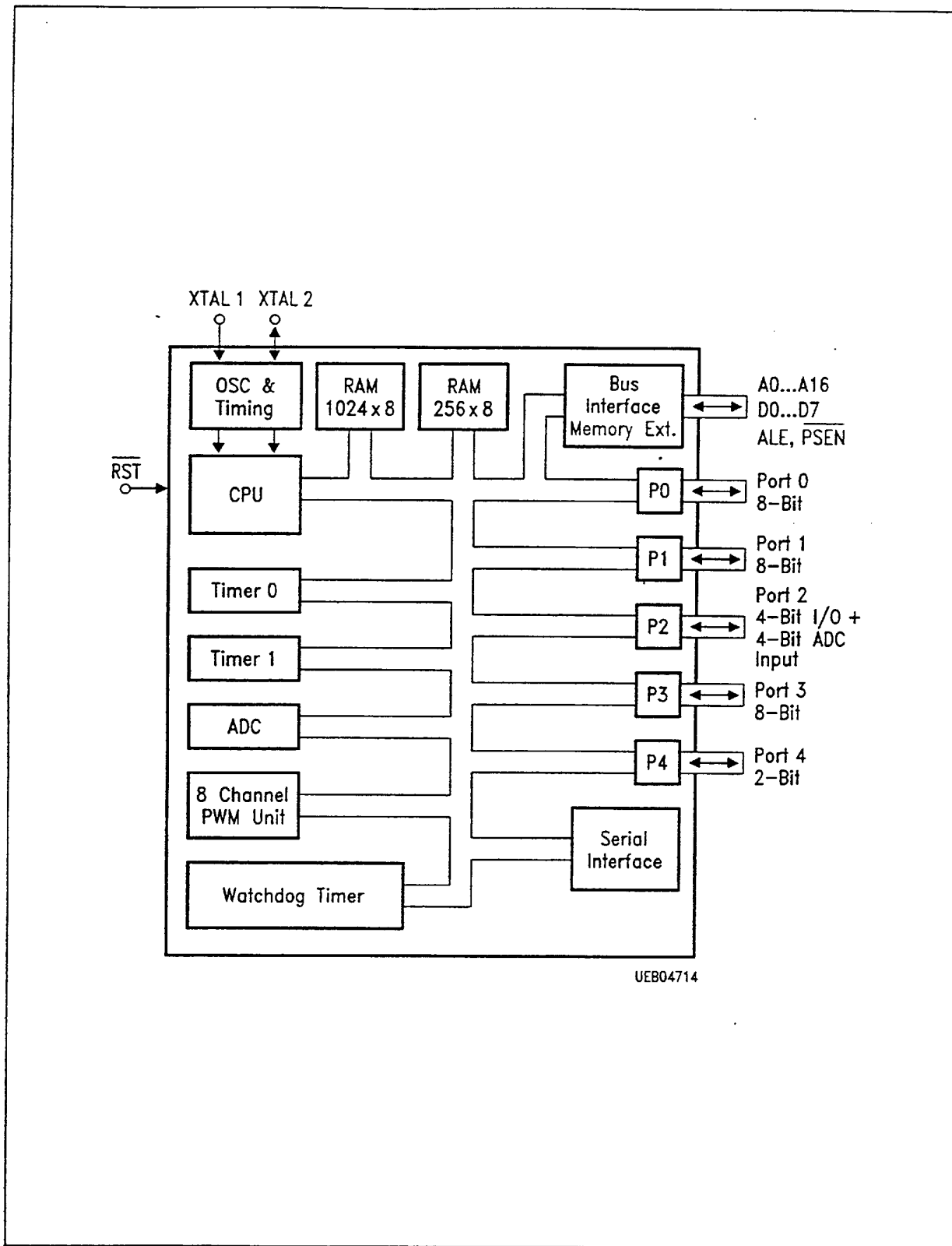


Figure 1  
Block Diagram

## 2 Functional Description

### 2.1 Architecture

The CPU manipulates operands in three memory spaces. These are the program memory (512 Kbyte) and (256 + 1024) byte internal data memory spaces. The program memory address space is provided to accommodate relocatable code.

The internal data memory address space is further divided into the 256-byte internal data RAM, 1024 bytes XRAM and the 128-byte Special Function Register (SFR) address spaces. Four register banks (each bank has eight registers), 128 addressable bits, and the stack reside in the internal data RAM. The stack depth is limited only by the available internal data RAM. Its location is determined by the 8-bit stack pointer. All registers except the program counter and the four 8-register banks reside in the special function register address space. These memory mapped registers include arithmetic registers, pointers, I/O-ports, registers for the interrupt system, timers, pulse width modulator and serial channel. Many locations in the SFR-address space are addressable as bits.

Note that reading from unused locations in internal data memory will yield undefined data.

Conditional branches are performed relative to the program counter. The register-indirect jump permits branching relative to a 16-bit base register with an offset provided by an 8-bit index register. Sixteen-bit jumps and calls permit branching to any location within one 64 K block of the 512 K program memory address space.

There are five methods for addressing source operands: register, direct, register-indirect, immediate, and base-register plus index-register indirect addressing.

The first three methods can be used for addressing destination operands. Most instructions have a "destination, source" field that specifies the data type, addressing methods and operands involved. For operations other than moves, the destination operand is also a source operand.

Registers in the four 8-register banks can be accessed through register, direct, or register-indirect addressing; the lower 128 bytes of internal data RAM through direct or register-indirect addressing, the upper 128 bytes of internal data RAM through register-indirect addressing; and the special function registers through direct addressing. Look-up tables resident in program memory can be accessed through base-register plus index-register indirect addressing.

#### 2.1.1 CPU-Hardware

##### Instruction Decoder

Each program instruction is decoded by the instruction decoder. This unit generates the internal signals that control the functions of each unit within the CPU-section. These signals control the sources and destination of data, as well as the function of the Arithmetic/Logic Unit (ALU).

##### Program Control Section

The program control section controls the sequence in which the instructions stored in program memory are executed. The conditional branch logic enables conditions internal and external to the processor to cause a change in the sequence of program execution. The 16-bit program counter holds the address of the instruction to be executed. It is manipulated with the control transfer instructions listed in chapter "Instruction Set".

**Internal Data RAM**

The internal data RAM provides a 256-byte scratch pad memory, which includes four register banks and 128 direct addressable software flags. Each register bank contains registers R0 – R7. The addressable flags are located in the 16-byte locations starting at byte address 32 and ending with byte location 47 of the RAM-address space.

In addition to this standard internal data RAM the processor contains additional 1024 bytes internal RAM. It can be considered as a part of an external data memory. It is located at addresses 63488 to 64511 of the external data memory address space and is referenced by MOVX-instructions (MOVX A, @DPTR).

**Arithmetic/Logic Unit (ALU)**

The arithmetic section of the processor performs many data manipulation functions and includes the Arithmetic/Logic Unit (ALU) and the A, B and PSW-registers. 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 of add, subtract, multiply, divide, increment, decrement, BCD-decimal-add-adjust and compare, and the logic operations of and, or, exclusive-or, complement and rotate (right, left, or nibble swap).

The A-register is the accumulator, the B-register is dedicated during multiply and divide and serves as both a source and a destination. During all other operations the B-register is simply another location of the special function register space and may be used for any purpose.

**Boolean Processor**

The Boolean processor is an integral part of the processor architecture. It is an independent bit processor with its own instruction set, its own accumulator (the carry flag) and its own bit-addressable RAM and I/O. The bit manipulation instructions allow the direct addressing of 128 bits within the internal data RAM and several bits within the special function registers. The special function registers which have addresses exactly divisible by eight contain directly addressable bits.

The Boolean processor can perform, on any addressable bit, the bit operations of set, clear, complement, jump-if-set, jump-if-not-set, jump-if-set then-clear and move to/from carry. Between any addressable bit (or its complement) and the carry flag it can perform the bit operation of logical AND or logical OR with the result returned to the carry flag.

**Program Status Word Register (PSW)**

The PSW-flags record processor status information and control the operation of the processor. The carry (CY), auxiliary carry (AC), two user flags (F0 and F1), register bank select (RS0 and RS1), overflow (OV) and parity (P) flags reside in the program status word register. These flags are bit-memory-mapped within the byte-memory-mapped PSW. The CY, AC, and OV flags generally reflect the status of the latest arithmetic operations. The CY-flag is also the Boolean accumulator for bit operations. The P-flag always reflects the parity of the A-register. F0 and F1 are general purpose flags which are pushed onto the stack as part of a PSW-save. The two register bank select bits (RS1 and RS0) determine which one of the four register banks is selected as follows:

RS1	RS0	Register Bank	Register Location
0	0	0	00H - 07H
0	1	1	08H - 0FH
1	0	2	10H - 17H
1	1	3	18H - 1FH

**Program Status Word PSW** **SFR-Address D0H**



**Stack Pointer (SP)**

The 8-bit stack pointer contains the address at which the last byte was pushed onto the stack. This is also the address of the next byte that will be popped. The SP is incremented during a push. SP can be read or written to under software control. The stack may be located anywhere within the internal data RAM address space and may be as large as 256 bytes.

**Data Pointer Register (DPTR)**

The 16-bit Data Pointer Register DPTR is the concatenation of registers DPH (high-order byte) and DPL (low-order byte). The DPTR is used in register-indirect addressing to move program memory constants and to access the extended data memory. DPTR may be manipulated as one 16-bit register or as two independent 8-bit registers DPL and DPH.

Eight data pointer registers are available, the active one is selected by a special function register (DPSEL).

### Port 0, Port 1, Port 2, Port 3, Port 4

The five ports provide 34 I/O-lines to interface to the external world. All five ports are both byte and bit addressable. Port 0 and port 2.4 – 2.7 are used for binary I/O only. Port 1 provides eight PWM-output channels as alternate functions while port 2.0 – 2.3 are digital or analog inputs. Port 3 contains special control signals. Port 4 will usually be selected as memory extension interface.

### Interrupt Logic

Controlled by two special function registers (IE, IP) the interrupt logic provides several interrupt vectors. Each of them may be assigned to high or low priority (see chapter "Interrupt System").

### Timer/Counter 0/1

Two general purpose 16-bit timers/counters are controlled by the special function registers TMOD and TCON (see chapter "General Purpose Timers/Counters").

### Serial Interface

A full duplex serial interface is provided where one of three operation modes may be selected. The serial interface is controlled by two special function registers (SCON, SBUF) as described in chapter "Serial Interface".

### Watchdog Timer

For software- and hardware security, a watchdog timer is supplied, which resets the processor, if not cleared by software within a maximum time period.

### Pulse Width Modulation Unit

Up to eight lines of port 1 may be used as PWM-outputs. The PWM-logic is controlled by registers PWME, PWMC, PWCOUNT, PWCOMP0 ... 7 (see chapter "Pulse Width Modulation Unit").

## 2.1.2 CPU-Timing

Timing generation is completely self-contained, except for the frequency reference which can be a crystal or external clock source. The on-board oscillator is a parallel anti-resonant circuit with a frequency range of 1.2 MHz to 16 MHz. There is a divide-by-12 internal timing which leads to a minimum instruction cycle of 0.75  $\mu$ s with a 16-MHz crystal. The XTAL2-pin is the output of a high-gain amplifier, while XTAL1 is its input. A crystal connected between XTAL1 and XTAL2 provides the feedback and phase shift required for oscillation. The 1.2 MHz to 16-MHz range is also accommodated when an external clock is applied to XTAL1 as the frequency source.

A machine cycle consists of 12 oscillator periods. Most instructions execute in one cycle. MUL (multiply) and DIV (divide) are the only instructions that take more than two cycles to complete. They take four cycles.

Normally, two code bytes are fetched from program memory during every machine cycle. The only exception to this is when a MOVX-instruction is executed. MOVX is a 1-byte 2-cycle instruction that accesses XRAM. During a MOVX, two fetches are skipped while the internal XRAM is being addressed.



### 2.1.3 Addressing Modes

There are five general addressing modes operating on bytes. One of these five addressing modes, however, operates on both bytes and bits:

- register
- direct (both bytes and bits)
- register indirect
- immediate
- base-register plus index-register indirect

The following table summarizes, which memory spaces may be accessed by each of the addressing modes:

#### Register Addressing

R0 – R7

ACC, B, CY (bit), DPTR

#### Direct Addressing

RAM (low part)

Special Function Registers

#### Register-Indirect Addressing

RAM (@R1, @R0, SP)

#### Immediate Addressing

Program Memory

#### Base-Register plus Index-Register Indirect Addressing

Program Memory (@DPTR + A, @PC + A)

#### Register Addressing

Register addressing accesses the eight working registers (R0 – R7) of the selected register bank. The PSW-register flags RS1 and RS0 determine which register bank is enabled. The least significant three bits of the instruction opcode indicate which register is to be used. ACC, B, DPTR and CY, the Boolean processor accumulator, can also be addressed as registers.

#### Direct Addressing

Direct byte addressing specifies an on-chip RAM-location (only low part) or a special function register. Direct addressing is the only method of accessing the special function registers. An additional byte is appended to the instruction opcode to provide the memory location address. The highest-order bit of this byte selects one of two groups of addresses: values between 0 and 127 (00H – 7FH) access internal RAM-locations, while values between 128 and 255 (80H – 0FFH) access one of the special function registers.

#### Register-Indirect Addressing

Register-indirect addressing uses the contents of either R0 or R1 (in the selected register bank) as a pointer to locations in the 256 bytes of internal RAM. Note that the special function registers are not accessible by this method.

Execution of *PUSH-* and *POP-*instructions also use register-indirect addressing. The stack pointer may reside anywhere in internal RAM.

### Immediate Addressing

Immediate addressing allows constants to be part of the opcode instruction in program memory. An additional byte is appended to the instruction to hold the source variable. In the assembly language and instruction set, a number sign (#) precedes the value to be used, which may refer to a constant, an expression, or a symbolic name.

### Base-Register plus Index Register-Indirect Addressing

Base-register plus index register-indirect addressing allows a byte to be accessed from program memory via an indirect move from the location whose address is the sum of a base register (DPTR or PC) and index register, ACC. This mode facilitates accessing to look-up-table resident in program memory.

## 2.2 Memory Organization

The processor memory is organized into two address spaces. The memory spaces are:

- 512-Kbyte external program memory address space
- 256 byte plus 128-byte internal data memory address space
- 1024-byte additional internal data memory

A 16-bit program counter and a dedicated banking logic provide the processor with its 512-Kbyte addressing capabilities (up to 19 address lines are available). The program counter allows the user to execute calls and branches to any location within the program memory space. There are no instructions that permit program execution to move from the program memory space to any of the data memory space.

### 2.2.1 External Program Memory

Certain locations in program memory are reserved for specific programs. Locations 0000 through 0002 are reserved for the initialization program. Following reset, the CPU always begins execution at location 0000. Locations 0003 through 0035 are reserved for the five interrupt-request service programs as indicated in the following table:

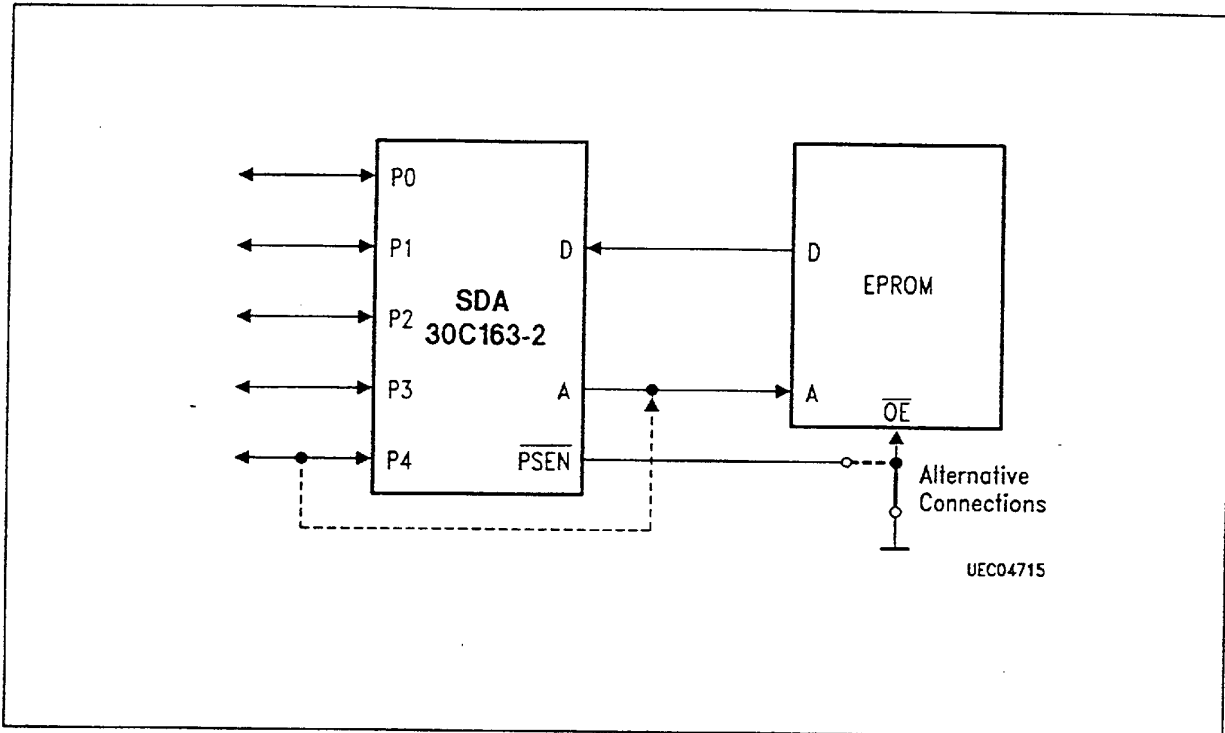
Source	Address	
External Interrupt 0	03	(03 <sub>H</sub> )
Timer 0 Overflow	11	(0B <sub>H</sub> )
External Interrupt 1	19	(13 <sub>H</sub> )
Timer 1 Overflow	27	(1B <sub>H</sub> )
Serial Interface	35	(23 <sub>H</sub> )

### Memory Extension

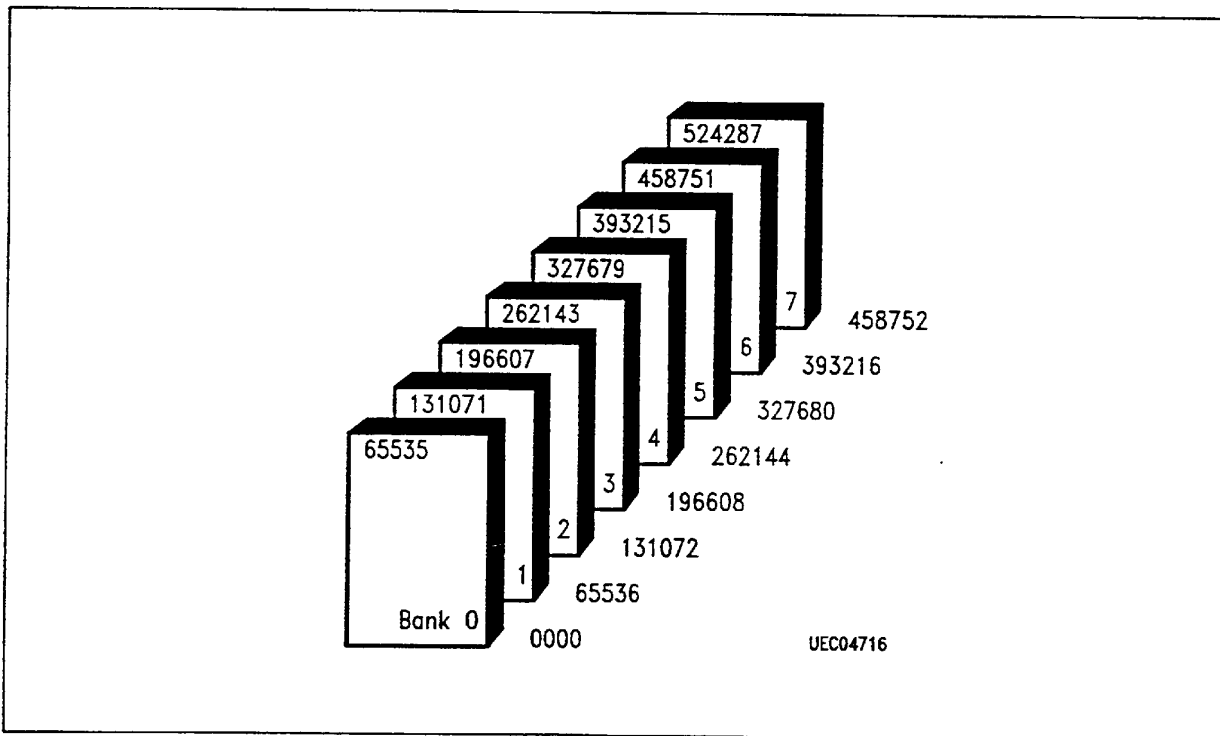
The processor is prepared to extend its external program memory space up to 512 Kbytes (figure 2, 3). For easy handling of existing software and assemblers this space is split into 8 banks of 64 Kbytes each. The extension concept, based on the standard 64 K addressing ability, is provided for high effective and easy memory access with minimum software overhead. There is also no need caring about bank organization during subroutine processing or interrupts. This is done through address bits A16 – 18, which are controlled by a special internal circuitry, performing a 'delayed banking'. The operations to the extended memory spaces are controlled by two additional special function registers called MEX1 and MEX2 (figure 4). The address bits A17 and A18 are implemented at Port 4. Programs, using only 128-Kbytes program memory space, may switch the address function off by setting bits NB, IB and bits MB to '1' followed by a LJMP. Then port 4 will work properly in port mode. Whenever full address mode is desired, port 4 bits have to be kept on '1' (table 1). After reset all CB are '0' and P4 latches are set to '1', resulting a '0' at the port 4 pins.

### Banking of Program Memory

After reset the bits for current bank (CB) and next bank (NB) are set to zero. This way the processor starts the same as any 8051 controller at address 00000<sub>H</sub>. Whenever a jump to another bank is required, the software has to change the bits NB16 – 18 for initializing the bank exchange (bits CB16 – 18 are read only). After operating the next LJMP instruction the NB16 – 18 bits (next bank) are copied to CB16 – 18 (current bank) and will appear at A16 – 18. Only LJMP will do this.



**Figure 2**  
Connecting External Program Memory



**Figure 3**  
Bank Organization

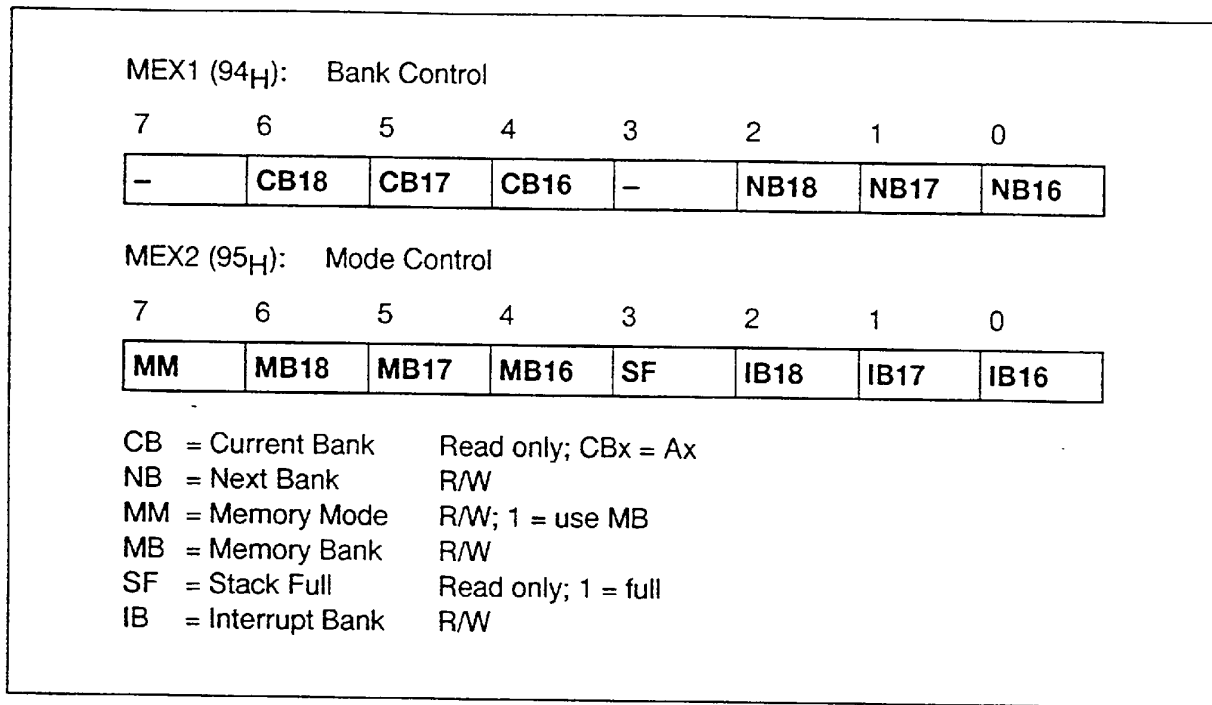


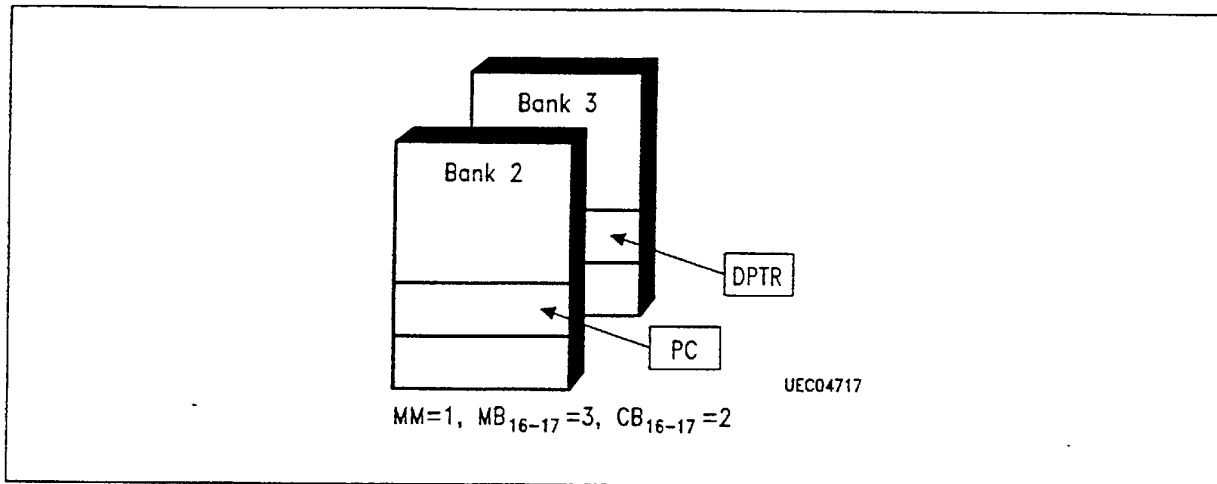
Figure 4  
Register Bits MEX1 / MEX2

Table 1  
Port 4 Configuration

CB	P4 Latch	P4 Out	Comment
0	0	0	x
0	1	0	Address
1	0	0	P4
1	1	1	Addr / P4

**MOVC-Handling**

MOVC-instructions may operate in two different modes, that are selected by bit MM in MEX2. On MM = 0 MOVC will access the current bank. On MM = 1 the bits MB16 – 18 will appear at A16 – A18 during MOVC.

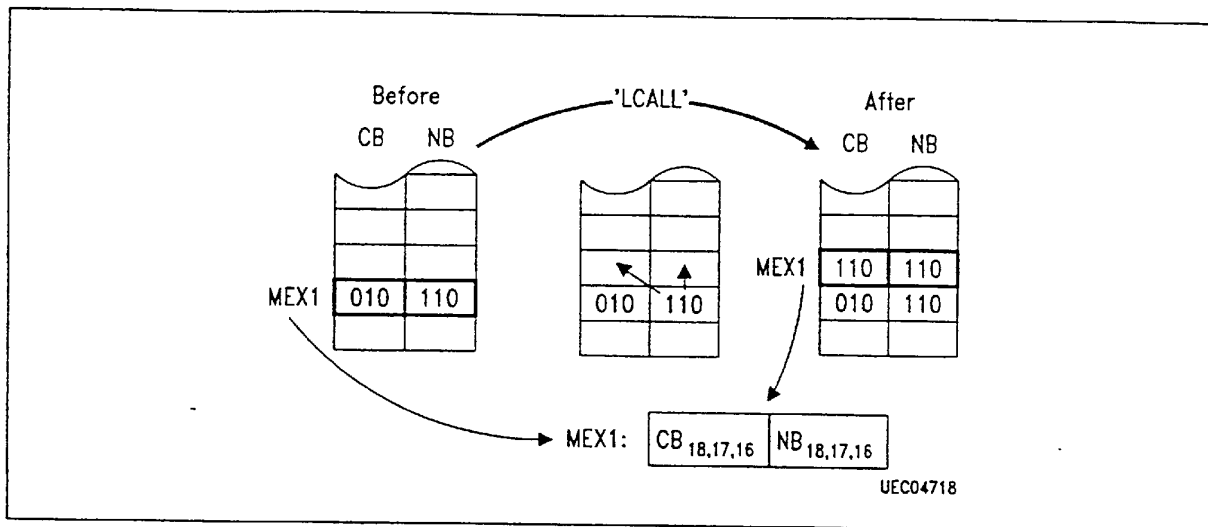


**Figure 5**  
PC and DPTR on Different Banks

### CALLS and Interrupts

For flexible use of CALL and interrupts the control logic holds an own 32 levels-six-bit-stack. Whenever a LCALL or ACALL occurs, CB16 – 18 and NB16 – 18 (MEX1) is copied to this stack and the memory extension stackpointer is incremented. Then NB16 – 18 is copied to CB16 – 18. Leaving subroutines through RET or RETI decrements the stack pointer and reads the old NB and CB contents from the stack. All six bits are required for saving to prevent conflicts on interrupt events. One additional feature simplifies the handling of interrupts: on occurrence the bits IB16 – 18 within MEX2 are copied to CB16 – 18 and NB16 – 18 after pushing their old contents on the stack. This way programmers can place their ISR (Interrupt Service Routine) on specific banks. After reset MM, MB16 – 18 and IB16 – 18 are set to zero.

In order to prevent loss of program control during deep subroutine nesting a warning bit 'SF' (Stack Full) is set in MEX2 whenever a memory extension stack depth overflow is imminent. For example **figure 6** shows the data flows at the memory extension stack during a LCALL. All three bits of NB are copied to the position CB and NB of the next higher stack level (now the current MEX1) while the last CB and NB are held on the stack. Returning from subroutine through RET the memory extension stack pointer decrements and CB and NB of MEX1 has the same contents as before LCALL.



**Figure 6**  
Processing LCALL (same as ACALL)

### Examples

The standard sequence jumping from one bank to another is simply preceding a 'MOV MEX1,#'-instruction to an 'LJMP / LCALL' as shown in figure 6. To operate programs up to 512 Kbytes with standard assemblers or from C the program can be split into sections, modules or files, that will each run in their own bank. Referencing banks to each other (jumps, calls, data moves) may be done by a simple preprocessing of the source programs or object files. Users, going to program a 512-Kbyte EPROM in assembler, may proceed like this:

- 1) build up to eight assembler source files (max. 64 K), inter bank operations will refer to dummy labels.
- 2) do assembler runs on each block and generate label lists.
- 3) preprocessing: substitute the inter bank labels in the source files with absolute 64 K addresses.
- 4) second and final assembler runs on each block, generate Hex files.
- 5) append the Hex files in right order.
- 6) program an EPROM.

More comfortable programming, e.g. based on C-programs, require similar processing of the source programs or object files with respect to special considerations of the compiler.

**Figure 7** shows an assembler program run, performing the following actions:

- 1) start at bank 0 at 00000.
- 2) set ISR-page to bank 2.
- 3) jump to bank 1 at address 25.
- 4) being interrupted to bank 2 ISR.
- 5) call a subprogram at bank 2 address 43.
- 6) after return read data from bank 2.

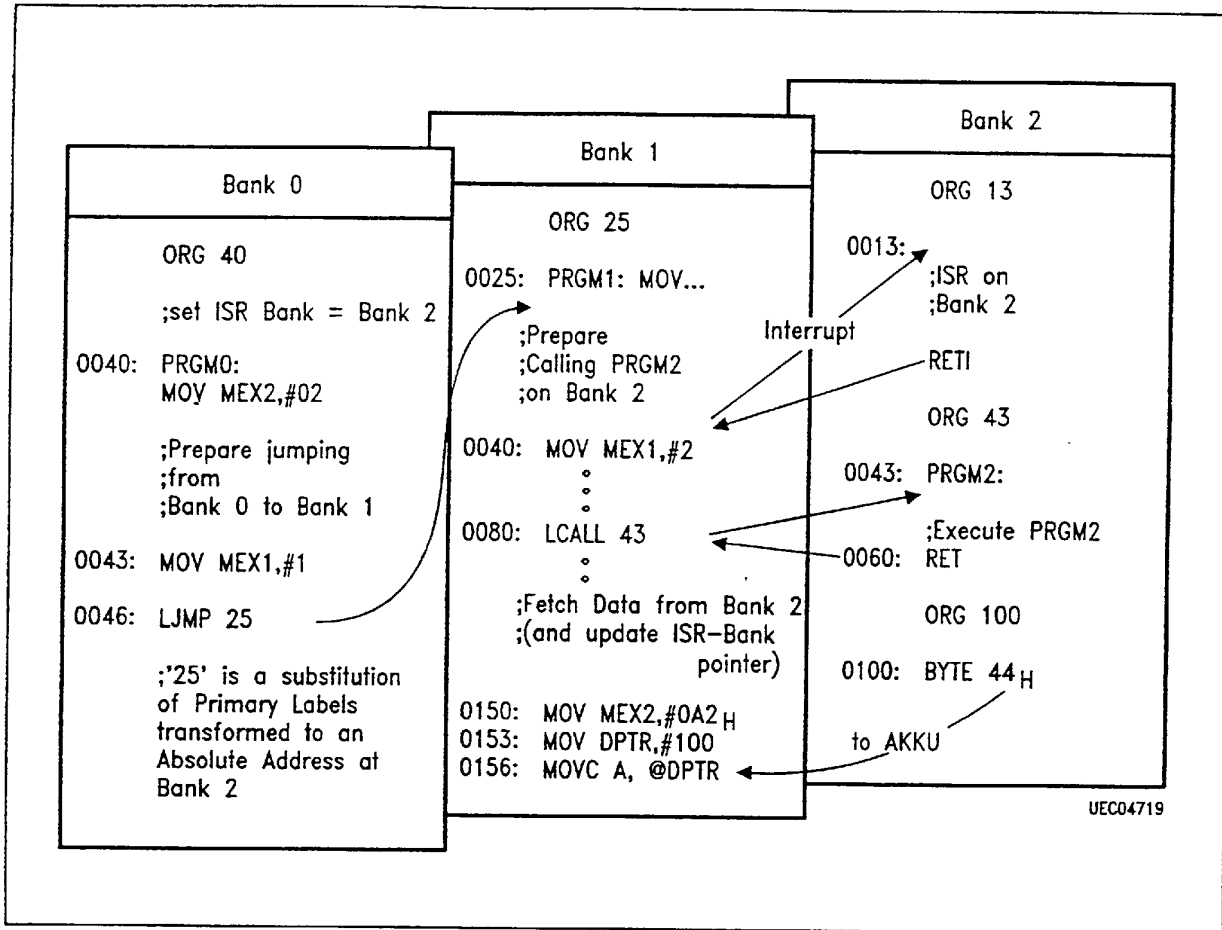


Figure 7  
Program Example

2.2.2 Internal Data RAM

The internal data memory is divided into four blocks: the lower 128 byte of RAM, the upper 128 byte of RAM, the 128-byte Special Function Register (SFR) area and the 1024-byte additional RAM (figure 8). Because the upper RAM-area and the SFR-area share the same address locations, they are accessed through different addressing modes.

The internal data RAM-address space is 0 to 255. Four banks of eight registers each occupy locations 0 through 31. Only one of these banks may be enabled at a time through a two-bit field in the PSW. In addition, 128-bit locations of the on-chip RAM are accessible through direct addressing.

These bits reside in internal data RAM at byte locations 32 through 47, as shown in figure 9. The lower 128 bytes of internal data RAM can be accessed through direct or register-indirect addressing, the upper 128 bytes of internal data RAM through register-indirect addressing and the special function registers through direct addressing.



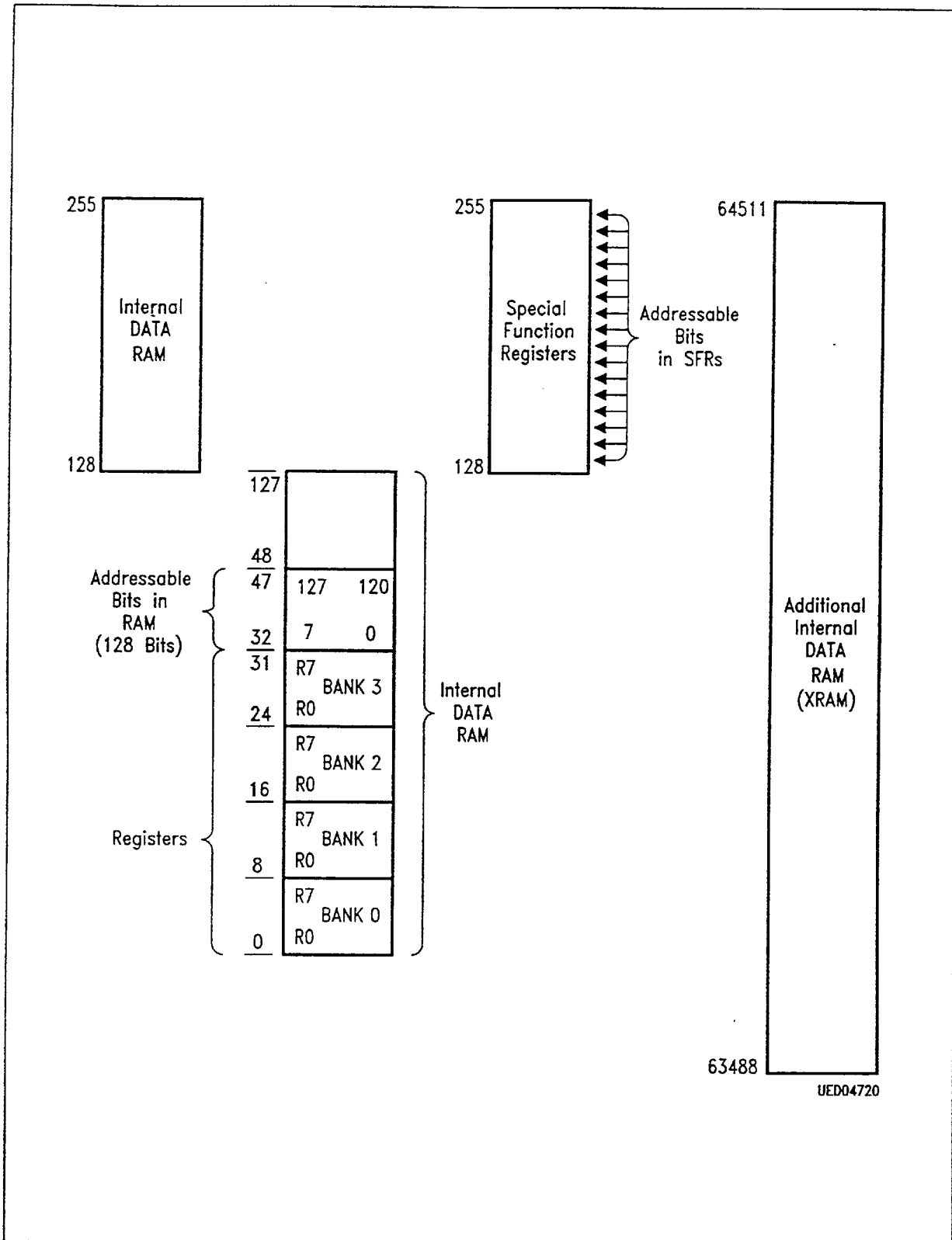
The stack can be located anywhere in the internal data RAM-address space. The stack depth is limited only by the available internal data RAM, thanks to an 8-bit reloadable stack pointer. The stack is used for storing the program counter during subroutine calls and may also be used for passing parameters. Any byte of internal data RAM or special function registers accessible through direct addressing can be pushed/popped.

An additional on-chip RAM-space called 'XRAM' extends the internal RAM-capacity up to 1280 bytes. The 1024 bytes of XRAM are accessed by MOVX @DPTR. XRAM is located in the upper area of the address space at 0F800<sub>H</sub> – 0FBFF<sub>H</sub>.

### 2.2.3 Special Function Registers

The special function register address space resides between addresses 128 and 255. All registers except the program counter and the four banks of eight working registers reside here. Memory mapping the special function registers allows them to be accessed as easily as the internal RAM. As such, they can be operated on by most instructions. A complete list of the special function registers is given in table 2.

In addition, many bit locations within the special function register address space can be accessed using direct addressing. These direct addressable bits are located at byte addresses divisible by eight as shown in figure 10.



**Figure 8**  
Internal Data Memory Address Space

RAM Byte	(MSB) (LSB)								
256									FF <sub>H</sub>
47	7F	7E	7D	7C	7B	7A	79	78	2F <sub>H</sub>
46	77	76	75	74	73	72	71	70	2E <sub>H</sub>
45	6F	6E	6D	6C	6B	6A	69	68	2D <sub>H</sub>
44	67	66	65	64	63	62	61	60	2C <sub>H</sub>
43	5F	5E	5D	5C	5B	5A	59	58	2B <sub>H</sub>
42	57	56	55	54	53	52	51	50	2A <sub>H</sub>
41	4F	4E	4D	4C	4B	4A	49	48	29 <sub>H</sub>
40	47	46	45	44	43	42	41	40	28 <sub>H</sub>
39	3F	3E	3D	3C	3B	3A	39	38	27 <sub>H</sub>
38	37	36	35	34	33	32	31	30	26 <sub>H</sub>
37	2F	2E	2D	2C	2B	2A	29	28	25 <sub>H</sub>
36	27	26	25	24	23	22	21	20	24 <sub>H</sub>
35	1F	1E	1D	1C	1B	1A	19	18	23 <sub>H</sub>
34	17	16	15	14	13	12	11	10	22 <sub>H</sub>
33	0F	0E	0D	0C	0B	0A	09	08	21 <sub>H</sub>
32	07	06	05	04	03	02	01	00	20 <sub>H</sub>
31	Bank 3								1F <sub>H</sub>
24									18 <sub>H</sub>
23	Bank 2								17 <sub>H</sub>
16									10 <sub>H</sub>
15	Bank 1								0F <sub>H</sub>
8									08 <sub>H</sub>
7	Bank 0								07 <sub>H</sub>
0									00 <sub>H</sub>

Figure 9  
Internal RAM-Bit Addresses

Direct Byte Address	Bit Address								Hardware Register Symbol
F8 <sub>H</sub>	FF	FE	FD	FC	FB	FA	F9	F8	PWCOMP7
F0 <sub>H</sub>	F7	F6	F5	F4	F3	F2	F1	F0	B
E8 <sub>H</sub>	-	-	-	-	-	-	E9	E8	P4
E0 <sub>H</sub>	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D8 <sub>H</sub>	DF	-	-	DC	DB	-	D9	D8	ADCON
D0 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	PSW
C8 <sub>H</sub>	CF	CE	CD	CC	-	-	-	-	PWMC
C0 <sub>H</sub>	C7	C6	C5	C4	C3	C2	C1	C0	PWME
B8 <sub>H</sub>	-	BE	-	-	-	-	-	-	WDSTART
B0 <sub>H</sub>	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8 <sub>H</sub>	AF	AE	AD	AC	AB	AA	A9	A8	IE
A0 <sub>H</sub>	A7	A6	A5	A4	A3	A2	A1	A0	P2
98 <sub>H</sub>	9F	9E	9D	9C	9B	9A	99	98	SCON
90 <sub>H</sub>	97	96	95	94	93	92	91	90	P1
88 <sub>H</sub>	8F	8E	8D	8C	8B	8A	89	88	TCON
80 <sub>H</sub>	87	86	85	84	83	82	81	80	P0

**Figure 10**  
Special Function Register Bit Address Space

**Table 2**  
**Special Function Register Overview**

Special Function Register Description	Symbolic Name	Address Location (hex.)	Address Location (dec.)	Bit Address MSB ... LSB (hex.)	Initial Value after Reset (hex./bin.)
<b>Arithmetic Registers</b>					
Accumulator	ACC, A	E0	224	E7 – E0	00
B-Register	B	F0	240	F7 – F0	00
Program Status Word	PSW	D0	208	D7 – D0	00
<b>System Control Registers</b>					
Stack Pointer	SP	81	129	–	07
Data Pointer (high byte)	DPH	83	131	–	00
Data Pointer (low byte)	DPL	82	130	–	00
Data Pointer Select	DPSEL	A2	162	–	xxxx x000
Power Control	PCON	87	135	–	FD
Memory Extension Bank	MEX1	94	148	–	88
Memory Extension Mode	MEX2	95	149	–	00
Advanced Function Register	AFR	A6	166	–	1xxx xxxx
<b>I/O-Port Registers</b>					
Port 0	P0	80	128	87 – 80	FF
Port 1	P1	90	144	97 – 90	FF
Port 2	P2	A0	160	A7 – A0	FF
Port 3	P3	B0	176	B7 – B0	FF
Port 4	P4	E8	232	E9 – E8	xxxx xx00
<b>Interrupt Control Registers</b>					
Interrupt Priority Flags	IP	A9	169	–	C0
Interrupt Enable Flags	IE	A8	168	AF – A8	00
<b>Timer 0/1 Registers</b>					
Timer 0/1 Mode Register	TMOD	89	137	–	00
Timer 0/1 Control Register	TCON	88	136	8F – 88	00
Timer 1 (high byte)	TH1	8D	141	–	00
Timer 0 (high byte)	TH0	8C	140	–	00
Timer 1 (low byte)	TL1	8B	139	–	00
Timer 0 (low byte)	TL0	8A	138	–	00
<b>Watchdog Timer Registers</b>					
Watchdog Start Register	WDSTART	B8	184	–	xx
Watchdog Reload Register	WDTREL	86	134	–	00
Watchdog Low Byte	WDTL	84	132	–	00
Watchdog High Byte	WDTH	85	133	–	80

## Special Function Register Overview (cont'd)

Special Function Register Description	Symbolic Name	Address Location (hex.)	Address Location (dec.)	Bit Address MSB ... LSB (hex.)	Initial Value after Reset (hex./bin.)
<b>Analog Digital Converter</b>					
ADC-Control Register	ADCON	D8	216	9F – 98	0xx00000
ADC-Data Register	ADDAT	D9	217	–	00
ADC-Start Register	DAPR	DA	218	–	xx
<b>Pulse Width Modulator Registers</b>					
Control Register	PWMC	C8	200	CF – C8	80
Enable Register	PWME	C0	192	C7 – C0	00
PWM-Counter Register	PWCOUNT	F9	249	–	00
Compare Register 0	PWCOMP0	F1	241	–	FF
Compare Register 1	PWCOMP1	F2	242	–	FF
Compare Register 2	PWCOMP2	F3	243	–	FF
Compare Register 3	PWCOMP3	F4	244	–	FF
Compare Register 4	PWCOMP4	F5	245	–	FF
Compare Register 5	PWCOMP5	F6	246	–	FF
Compare Register 6	PWCOMP6	F7	247	–	FF
Compare Register 7	PWCOMP7	F8	248	FF – F8	FF
<b>Serial Interface Registers</b>					
Serial Control Register	SCON	98	144	9F – 98	00
Serial Data Register	SBUF	99	145	–	xx

**2.3 Interrupt System**

External events and the real-time on-chip peripherals require CPU-service asynchronous to the execution of any particular section of code. To couple the asynchronous activities of these functions to normal program execution, a sophisticated multiple-source, two-priority-level, nested interrupt system is provided. Interrupt response delay ranges from 2  $\mu$ s to 5.25  $\mu$ s when using a 16-MHz crystal.

**2.3.1 Interrupt Sources**

The processor acknowledges interrupt requests from five sources: two from external sources via the INT0 and INT1 pins, one from each of the two internal counters and one from the serial I/O-port. Each of the five sources can be assigned to either of two priority levels and can be independently enabled and disabled. Additionally, all enabled sources can be globally disabled or enabled.

Interrupts result in a transfer of control to a new program location. Each interrupt vectors to a separate location in program memory for its service program. The program servicing the request begins at this address. The starting address (interrupt vector) of the interrupt service program for each interrupt source is shown in the following table:

Interrupt Source	Starting Address	
External Request 0	03	(03H)
Internal Timer/Counter 0	11	(0BH)
External Request 1	19	(13H)
Internal Timer/Counter 1	27	(1BH)
Serial Interface	35	(23H)

**2.3.2 Interrupt Control**

The information flags, which control the entire interrupt system, are stored in four special function registers:

TCON	Timer/Counter Control Register	88H
IE	Interrupt Enable Register	A8H
IP	Interrupt Priority Register	A9H
SCON	Serial Control Register	98H

The interrupt system is shown diagrammatically in figure 11.

A source requests an interrupt by setting its associated interrupt request flag in the TCON or IFR-register, as detailed in the following table:

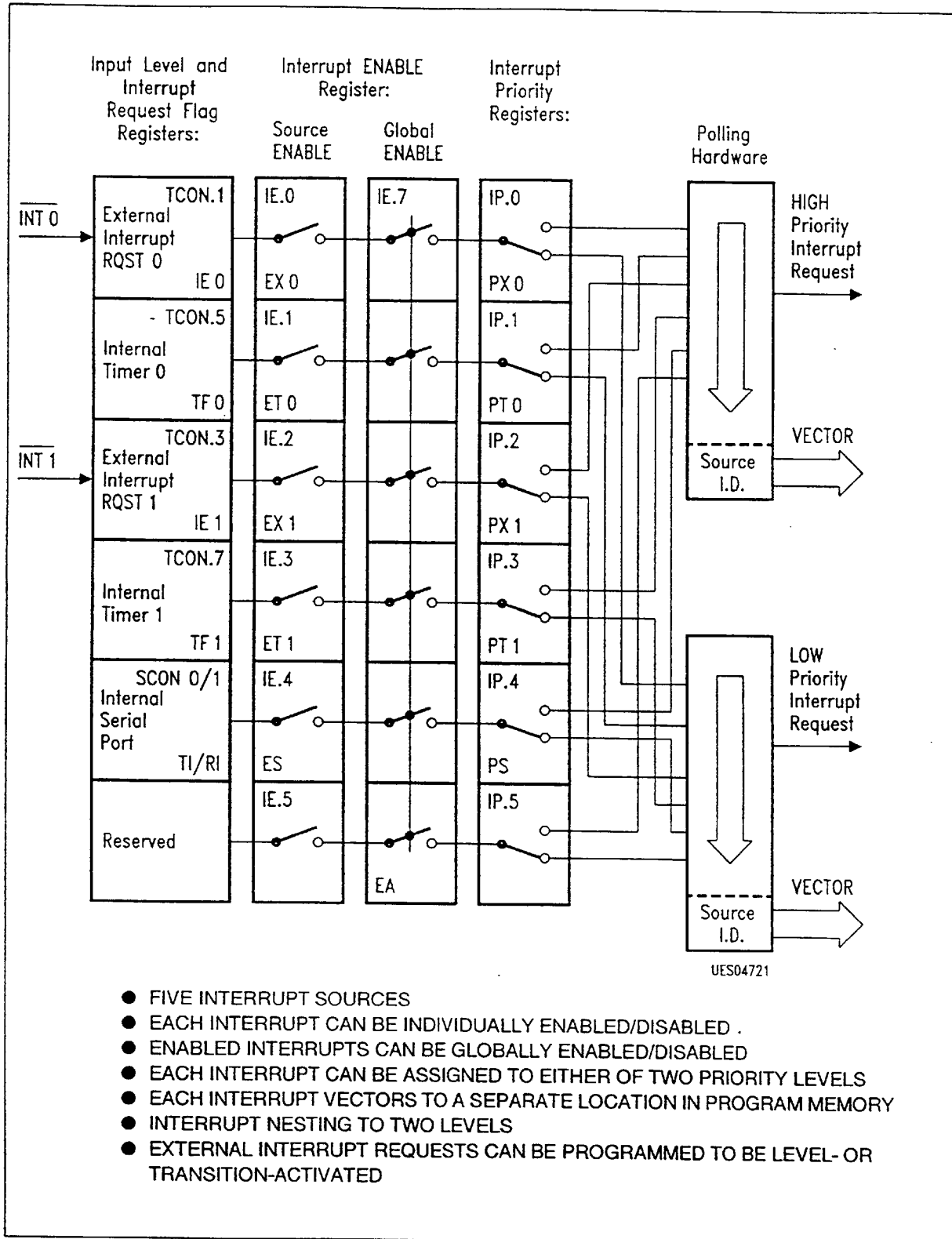
Interrupt Source	Request Flag	Bit Location
External Request 0	IE0	TCON.1
Internal Timer/Counter 0	TF0	TCON.5
External Request 1	IE1	TCON.3
Internal Timer/Counter 1	TF1	TCON.7
Serial Interface	RI/TI	SCON0/1

The timer 0 and timer 1 interrupts are generated by TF0 and TF1, which are set by a rollover in their respective timer/counter register, except for timer 0 in mode 3.

Within the IE-register there are seven addressable flags. Five flags enable/disable the five interrupt sources when set/cleared. Setting/clearing the seventh flag permits a global enable/disable of all enabled interrupt requests.

All the bits that generate interrupts can be set or cleared by software, with the same result as though they had been set or cleared by hardware. That is, interrupts can be generated or pending interrupt requests can be cancelled by software.





- FIVE INTERRUPT SOURCES
- EACH INTERRUPT CAN BE INDIVIDUALLY ENABLED/DISABLED .
- ENABLED INTERRUPTS CAN BE GLOBALLY ENABLED/DISABLED
- EACH INTERRUPT CAN BE ASSIGNED TO EITHER OF TWO PRIORITY LEVELS
- EACH INTERRUPT VECTORS TO A SEPARATE LOCATION IN PROGRAM MEMORY
- INTERRUPT NESTING TO TWO LEVELS
- EXTERNAL INTERRUPT REQUESTS CAN BE PROGRAMMED TO BE LEVEL- OR TRANSITION-ACTIVATED

Figure 11  
Interrupt System

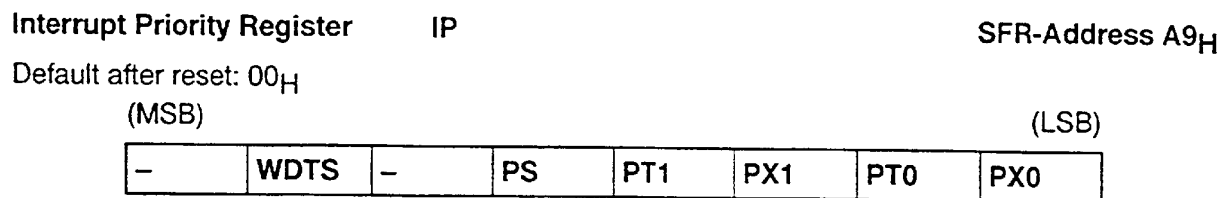
**Figure 12**  
**Interrupt Enable Register IE**

**Interrupt Enable Register IE** **SFR-Address A8H**  
 Default after reset: 00H  
 (MSB) (LSB)

EA	WDT	-	ES	ET1	EX1	ET0	EX0
----	-----	---	----	-----	-----	-----	-----

- EA**            Enables or disables 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.
- WDT**            Watchdog timer refresh flag (see chapter "Watchdog Timer").
- IE.5**            Reserved
- ES**             Enables or disables the serial interface interrupt. If ES = 0, this interrupt will be disabled.
- ET1**            Enables or disables the timer 1 overflow interrupt. If ET1 = 0, the timer 1 interrupt is disabled.
- EX1**            Enables or disables external interrupt 1. If EX1 = 0, external interrupt 1 is disabled.
- ET0**            Enables or disables the timer 0 overflow interrupt. If ET0 = 0, the timer 0 interrupt is disabled.
- EX0**            Enables or disables external interrupt 0. If EX0 = 0, external interrupt 0 is disabled.

**Figure 13**  
Interrupt Priority Register IP



- IP.7      Reserved
- WDTS     Watchdog timer interrupt flag (see chapter "Watchdog Timer").
- IP.5      Reserved
- PS        Defines the serial interface interrupt priority level. PS = 1 programs it to the higher priority level.
- PT1      Defines the timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
- PX1      Defines the external interrupt 1 priority level. PX1 = 1 programs it to the higher priority level.
- PT0      Defines the timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.
- PX0      Defines the external interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Setting/clearing a bit in the IP-register establishes its associated interrupt request as a high/low priority. If a low-priority level interrupt is being serviced, a high-priority level interrupt will interrupt it. However, an interrupt source cannot interrupt a service program of the same or higher priority level.

If two requests of different priority levels are received simultaneously, the request of higher priority level will be serviced. If requests of the same priority level 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, as follows:

Source	Priority within Level
1. IE0	(highest)
2. TF0	
3. IE1/OSD	
4. TF1	
5. RI/TI	
	(lowest)

Note that the "priority within level" structure is only used to resolve simultaneous requests of the same priority level.

**2.3.3 Interrupt Nesting**

The process whereby a high-level interrupt request interrupts a low-level interrupt service program is called nesting. In this case the address of the next instruction in the low-priority service program is pushed onto the stack, the stack pointer is incremented by two and processor control is transferred to the program memory location of the first instruction of the high-level service program. The last instruction of the high-priority interrupt service program must be a RETI-instruction. This instruction clears the higher "priority-level-active" flip-flop. RETI also returns processor control to the next instruction of the low-level interrupt service program. Since the lower "priority-level-active" flip-flop has remained set, high priority interrupts are re-enabled while further low-priority interrupts remain disabled.

**2.3.4 External Interrupts**

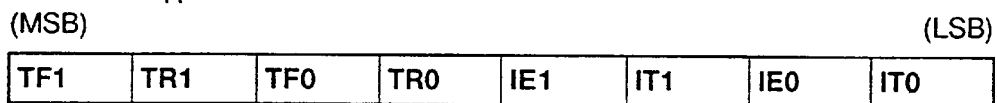
The external interrupt request inputs ( $\overline{INT0}$  and  $\overline{INT1}$ ) can be programmed for either transition-activated or level-activated operation. Control of the external interrupts is provided by the four low-order bits of TCON as shown in figure 14.

When IT0 and IT1 are set to one, interrupt requests on  $\overline{INT0}$  and  $\overline{INT1}$  are transition-activated (high-to-low), else they are low-level activated. IE0 and IE1 are the interrupt request flags. These flags are set when their corresponding interrupt request inputs at  $\overline{INT0}$  and  $\overline{INT1}$ , respectively, are low when sampled by the processor and the transition-activated scheme is selected by IT0 and IT1.

**Figure 14**  
**Function of Lower Nibble Bits in TCON**

**Timer and Interrupt Control Register TCON** **SFR-Address 88H**

Default after reset: 00H



- TCON.4 – TCON.7**    See chapter "General Purpose Timers/Counters"
- IE1**                    Interrupt 1 edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
- IT1**                    Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts. IT1 = 1 selects transition-activated (high-to-low) external interrupts.
- IE0**                    Interrupt 0 edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
- IT0**                    Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts. IT0 = 1 selects transition-activated (high-to-low) external interrupts.

- Transition-Activated Interrupts  
(IT0 = 1, IT1 = 1)

The IE0, IE1 flags are set by a high-to-low transition at  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ , respectively; they are cleared during entering the corresponding interrupt service routine.

For transition-activated operation, the input must remain low for more than twelve oscillator periods, but needs not to be synchronous with the oscillator. The upward transition of a transition-activated input may occur at any time after the twelve oscillator period latching time, but the input must remain high for twelve oscillator periods before reactivation.

- Level-Activated Interrupts  
(IT0 = 0, IT1 = 0)

The IE0, IE1 flags are set whenever  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$  are respectively sampled at low level. Sampling  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$  at high level clears IE0, IE1, respectively.

For level-activated operation, if the input is low during the sampling that occurs fourteen oscillator periods before the end of the instruction in progress, an interrupt subroutine call is made. The level-activated input needs to be low only during the sampling that occurs fourteen oscillator periods before the end of the instruction in progress and may remain low during the entire execution of the service program. However, the input must be deactivated before the service routine is completed to avoid invoking a second interrupt, or else another interrupt will be generated

### 2.3.5 Interrupt Task Function

The processor records the active priority level(s) by setting internal flip-flop(s). One of these non-addressable flip-flops is set while a low-level interrupt is being serviced. The other flip-flop is set while the high-level interrupt is being serviced. The appropriate flip-flop is set when the processor transfers control to the service program. The flip-flop corresponding to the interrupt level being serviced is reset when the processor executes a RETI-instruction

The sequence of events for an interrupt is:

- A source provokes an interrupt by setting its associated interrupt request bit to let the processor know an interrupt condition has occurred.
- The CPU's internal hardware latches the internal request in the tenth, twenty-second, thirty-fourth and forty-sixth oscillator period of the instruction in progress.
- The interrupt request is conditioned by bits in the interrupt enable and interrupt priority register.
- The processor acknowledges the interrupt by setting one of the two internal "priority-level active" flip-flops and performing a hardware subroutine call. This call pushes the PC (but not the PSW) onto the stack and, for most sources, clears the interrupt request flag.
- The service program is executed.
- Control is returned to the main program when the RETI-instruction is executed. The RETI-instruction also clears one of the internal "priority-level active" flip-flops.

Most interrupt request flags IE0, IE1, TF0 and TF1 are cleared when the processor transfers control to the first instruction of the interrupt service program. The RI/TI-interrupt request flag must be cleared as part of the respective interrupt service program. This is also the case for IE0, IE1, if  $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$  are level activated.

**2.3.6 Response Time**

The highest-priority interrupt request gets serviced at the end of the instruction in progress unless the request is made in the last fourteen oscillator periods of the instruction in progress. Under this circumstance, the next instruction will also execute before the interrupt's subroutine call is made.

If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the 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 an access to 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, in a single-interrupt system, the response time is always more than 3 cycles and less than 8 cycles (approximately 5.25  $\mu$ s at 16-MHz operation). Examples of the best and worst case conditions are illustrated in the following table.

Instruction	Time (Oscillator Periods)	
	Best Case	Worst Case
External interrupt generated immediately before (best) / after (worst) the pin is sampled (time until end of bus cycle).	$2 + \epsilon$	$2 - \epsilon$
Current or next instruction finishes in 12-oscillator periods	12	12
Next instruction is MUL or DIV	don't care	48
Internal latency for hardware subroutine call	24	24
	38	86

If an interrupt of equal or higher priority level is already in progress, the additional wait time obviously depends on the nature of the other interrupt's service routine.

## 2.4 Processor Reset and Initialization

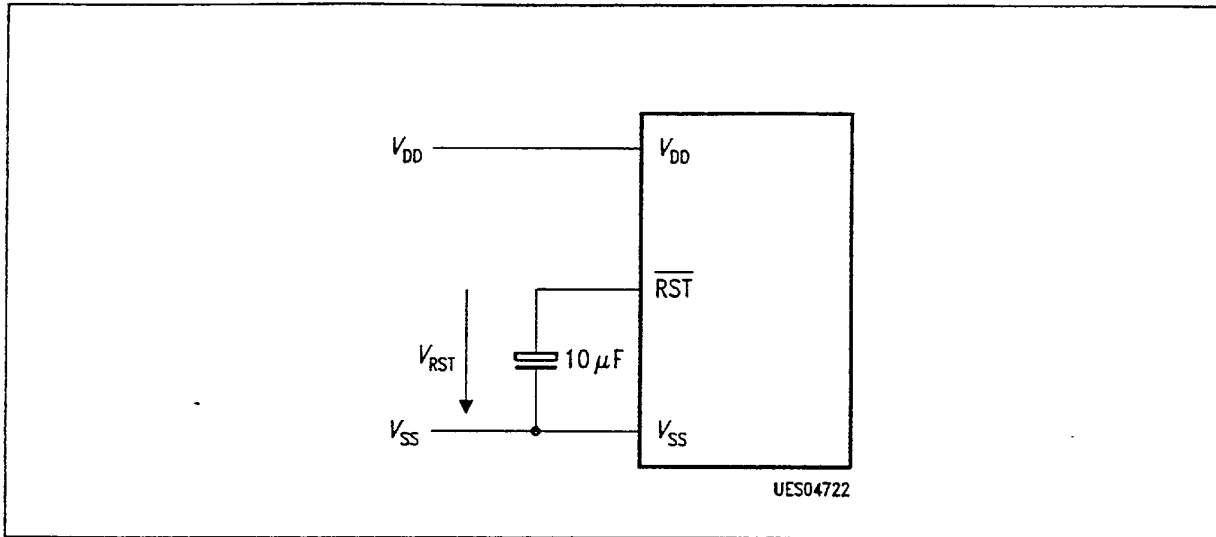
Processor initialization is accomplished with activation of the  $\overline{\text{RST}}$  pin, which is the input to a Schmitt Trigger. To reset the processor, this pin should be held low for at least two machine cycles, while the oscillator is running. Upon powering up,  $\overline{\text{RST}}$  should be held low for at least 10 ms after the power supply stabilizes to allow the oscillator to stabilize. Crystal operation below 6 MHz will increase the time necessary to hold  $\overline{\text{RST}}$  low. 24 oscillator periods after receiving of  $\overline{\text{RST}}$ , the processor ceases from instruction execution and remains dormant for the duration of the pulse. The high-going transition then initiates a sequence which requires approximately twelve oscillator periods to execute before normal operation commences with the instruction at absolute location 0000<sub>H</sub>. Program memory locations 0000<sub>H</sub> through 0002<sub>H</sub> are reserved for the initialization routine of the microcomputer. This sequence ends with registers initialized as shown in chapter 'Memory Organization'.

After the processor is reset, all ports are written with one (1) except Port 4, which is as an extended address output. Outputs are undefined until the reset period is complete.

An automatic reset can be obtained when  $V_{\text{DD}}$  is turned on by connecting the  $\overline{\text{RST}}$ -pin to  $V_{\text{SS}}$  through a 10  $\mu\text{F}$  capacitor, providing the  $V_{\text{DD}}$  rise time does not exceed a millisecond and the oscillator start-up time does not exceed 10 milliseconds. When power comes on, the current drawn by  $\overline{\text{RST}}$ -pin starts to charge the capacitor. The voltage  $V_{\text{RST}}$  at  $\overline{\text{RST}}$ -pin is the capacitor voltage, and increases to  $V_{\text{DD}}$  as the capacitor charges. The larger the capacitor, the more slowly  $V_{\text{RST}}$  decreases.  $V_{\text{RST}}$  must remain below the lower threshold of the Schmitt Trigger long enough to effect a complete reset. The time required is the oscillator start-up time plus 2 machine cycles.

**Attention:** While reset is active and at least two machine cycles after rising edge of  $\overline{\text{RST}}$ , ALE should not be pulled down externally.

If during powering off the supply voltage drops below  $V_{\text{DD min}}$  (4.5 V) and an external reset is not applied, the behaviour of the processor is not defined.



**Figure 15**  
**Power-On Reset Circuit**

**Power-Down Operations**

The controller provides two modes in which power consumption can be significantly reduced.

- Idle mode. The CPU is gated off from the oscillator. All peripherals are still provided with the clock and are able to work.
- Power-down mode. Operation of the controller is turned off. This mode is used to save the contents of internal RAM with a very low standby current.

Both modes are entered by software. Special function register PCON is used to enter one of these modes.

**Power Control Register PCON**

SFR-Address 87<sub>H</sub>

Default after reset: 000xxx00

(MSB)				(LSB)			
SMOD	PDS	IDLS	-	-	-	PDE	IDLE

**PDS** Power-down start bit. The instruction that sets the PDS-flag is the last instruction before entering the power down mode.

**IDLS** IDLE start bit. The instruction that sets the PDS-flag is the last instruction before entering the idle mode.

**PDE** Power-down enable bit. When set, starting the power-down mode is enabled.

**IDLE** Idle enable bit. When set, starting the idle mode is enabled.

**SMOD** Baud rate control for serial interface; if set, the baud rate is doubled.



The idle mode can be terminated by activation of any enabled interrupt (or a hardware reset). The CPU-operation is resumed, the interrupt will be serviced and the next instruction to be executed after RETI-instruction will be the one following the instruction that set the bit IDLS. The port state and the contents of SFRs are held during idle mode.

The only exit from power-down mode is a hardware reset. The reset will redefine all SFRs, but will not change the contents of internal RAM.

## 2.5 Ports and I/O-Pins

There are 30 I/O-pins configured as three 8-bit ports, one 4-bit-port (P2.4 – 2.7) and one 2-bit port (P4.0 – 4.1). Each pin can be individually and independently programmed as input or output and each can be configured dynamically.

An instruction that uses a port's bit/byte as a source operand reads a value that is the logical AND of the last value written to the bit/byte and the polarity being applied to the pin/pins by an external device (this assumes that none of the processor's electrical specifications are being violated). An instruction that reads a bit/byte, operates on the content, and writes the result back to the bit/byte, reads the last value written to the bit/byte instead of the logic level at the pin/pins. Pins comprising a single port can be made a mixed collection of inputs and outputs by writing a "one" to each pin that is to be an input. Each time an instruction uses a port as the destination, the operation must write "ones" to those bits that correspond to the input pins. An input to a port pin needs not to be synchronized to the oscillator.

All the port latches have "one" s written to them by the reset function. If a "zero" is subsequently written to a port latch, it can be reconfigured as an input by writing a "one" to it.

The instructions that perform a read of, operation on, and write to a port's bit/byte are INC, DEC, CPL, JBC, SETB, CLR, MOV P.X, CJNE, DJNZ, ANL, ORL, and XRL. The source read by these operations is the last value that was written to the port, without regard to the levels being applied at the pins. This insures that bits written to a "one" (for use as inputs) are not inadvertently cleared.

Port 0 has an open-drain output. Writing a "one" to the bit latch leaves the output transistor off, so the pin floats.

In that condition it can be used as a high-impedance input. Port 0 is considered "true bidirectional", because when configured as an input it floats.

Ports 1, 2.4 – 2.7, 3 and 4 have "quasi-bidirectional" output drivers which comprise an internal pullup resistor. When configured as inputs they pull high and will source current when externally pulled low (see DC Characteristics).

In ports 1, 2.4 – 2.7, 3 and 4 the output drivers provide source current for two oscillator periods if, and only if, software updates the bit in the output latch from a "zero" to an "one". Sourcing current only on "zero to one" transition prevents a pin, programmed as an input, from sourcing current into the external device that is driving the input pin.

Secondary functions can be selected individually and independently for the pins of port 1 and 3. Further information on port 1's secondary functions is given in chapter "Pulse Width Modulation Unit". P3 generates the secondary control signals automatically as long as the pin corresponding to the appropriate signal is programmed as an input, i. e. if the corresponding bit latch in the P3 special function register contains a "one".

The following alternate functions can be selected when using the corresponding P3 pins:

- P3.2  $\overline{\text{INT0}}$  (external interrupt 0)
- P3.3  $\overline{\text{INT1}}$  (external interrupt 1)
- P3.4 T0 (Timer/Counter 0 external input)
- P3.5 T1 (Timer/Counter 1 external input)
- P3.6 RxD (serial port receive line)
- P3.7 TxD (serial port transmit line)

### Read Modify-Write Feature

“Read-modify-write” commands are instructions that read a value, possibly change it, and then rewrite it to the latch. When the destination operand is a port or a port bit, these instructions read the latch rather than the pin. The read-modify-write instructions are listed in **table 3**.

The read-modify-write instructions are directed to the latch rather than the pin in order 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 “one” 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 and interpret it as a 0. Reading the latch rather than the pin will return the correct value of “one”.

**Table 3**  
**Read-Modify-Write Instructions**

Mnemonic	Description	Example
ANL	logical AND	ANL P1, A
ORL	logical OR	ORL P2, A
XRL	logical EX – OR	XRL P3, A
JBC	jump if bit = 1 and clear bit	JBC P1.1, LABEL
CPL	complement bit	CPL P3.0
INC	increment	INC P1
DEC	decrement	DEC P1
DJNZ	decrement and jump if not zero	DJNZ P3, LABEL
MOV PX.Y, C*	move carry bit to bit Y of Port X	MOV P1.7, C
CLR PX.Y*	clear bit Y of Port X	CLR P2.6
SET PX.Y*	set bit Y of Port X	SET P3.5

\* The instruction reads the port byte (all 8 bits), modifies the addressed bit, then writes the new byte back to the latch

## 2.6 General Purpose Timers/Counters

Two independent general purpose 16-bit timers/ counters are integrated for use in measuring time intervals, measuring pulse widths, counting events, and causing periodic (repetitive) interrupts. Either can be configured to operate as timer or event counter.

In the "timer" function, the registers TLx and/or THx ( $x = 0, 1$ ) are incremented every machine cycle. Thus, one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the "counter" function, the registers TLx and/or THx ( $x = 0, 1$ ) are incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 or T1. In this function, the external input is sampled during 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 the cycle following the one in which the transition was detected. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should be held for at least one full machine cycle.

### Timer/Counter 0: Mode Selection

Timer/counter 0 can be configured in one of four operating modes, which are selected by bit-pairs (M1, M0) in TMOD-register (figure 16).

#### – Mode 0

Putting timer/counter 0 into mode 0 makes it look like an 8048 timer, which is an 8-bit counter with a divide-by-32 prescaler. Figure 18 shows the mode 0 operation as it applies to timer 0.

In this mode, the timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the timer interrupt flag TF0. 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 (figure 17).  $GATE$  is contained in register TMOD (figure 16).

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 1

Mode 1 is the same as mode 0, except that the timer/counter 0 register is being run with all 16 bits.

#### – Mode 2

Mode 2 configures the timer/counter 0 register as an 8-bit counter (TL0) with automatic reload, as shown in figure 19. 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.

#### – Mode 3

Timer/counter 0 in mode 3 establishes TL0 and TH0 as two separate counters. The logic for mode 3 on timer 0 is shown in figure 20. TL0 uses the timer 0 control bits: C/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. With timer 0 in mode 3, the processor can operate as if it has three timers/counters. 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 in any application not requiring an interrupt.

### Timer/Counter 1: Mode Selection

Timer/counter 1 can also be configured in one of four modes, which are selected by its own bitpairs (M1, M0) in TMOD-register.

The serial port receives a pulse each time that timer/counter 1 overflows. This pulse rate is divided to generate the transmission rate of the serial port.

Modes 0 and 1 are the same as for counter 0.

#### – Mode 2

The "reload" mode is reserved to determine the frequency of the serial clock signal (not implemented).

#### – Mode 3

When counter 1's mode is reprogrammed to mode 3 (from mode 0, 1 or 2), it disables the increment counter. This mode is provided as an alternative to using the TR1 bit (in TCON-register) to start and stop timer/counter 1.

### Configuring the Timer/Counter Input

The use of the timer/counter is determined by two 8-bit registers, TMOD (timer mode) and TCON (timer control), as shown in **figure 16 and 17**. The input to the counter circuitry is from an external reference (for use as a counter), or from the on-chip oscillator (for use as a timer), depending on whether TMOD's C/T-bit is set or cleared, respectively. When used as a time base, the on-chip oscillator frequency is divided by twelve (12) before being used as the counter input. When TMOD's GATE bit is set (1), the external reference input (T1, T0) or the oscillator input is gated to the counter conditional upon a second external input ( $\overline{\text{INT0}}$ ,  $\overline{\text{INT1}}$ ) being high. When the GATE bit is zero (0), the external reference, or oscillator input, is unconditionally enabled. In either case, the normal interrupt function of  $\overline{\text{INT0}}$  and  $\overline{\text{INT1}}$  is not affected by the counter's operation. If enabled, an interrupt will occur when the input at  $\overline{\text{INT0}}$  or  $\overline{\text{INT1}}$  is low. The counters are enabled for incrementing when TCON's TR1 and TR0 bits are set. When the counters overflow, the TF1 and TF0 bits in TCON get set, and interrupt requests are generated.

The counter circuitry counts up to all 1's and then overflows to either 0's or the reload value. Upon overflow, TF1 or TF0 is set. When an instruction changes the timer's mode or alters its control bits, the actual change occurs at the end of the instruction's execution.

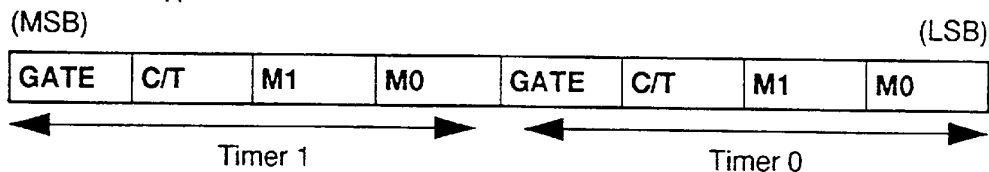
The T1 and T0 inputs are sampled near the falling-edge of ALE in the tenth, twenty-second, thirty-fourth and forty-sixth oscillator periods of the instruction-in-progress. Thus, an external reference's high and low times must each be a minimum of twelve oscillator periods in duration. There is a twelve oscillator period delay from the time when a toggled input (transition from high to low) is sampled to the time when the counter is incremented.

Figure 16  
Timer/Counter Mode Register

Timer 0/1 Mode Register TMOD

SFR-Address 89<sub>H</sub>

Default after reset: 00<sub>H</sub>



**GATE** Gating control when set. Timer/counter “x” is enabled only while  $\overline{\text{INTx}}$  pin is high and “TRx” control pin is set. When cleared, timer “x” is enabled, whenever “TRx” control bit is set.

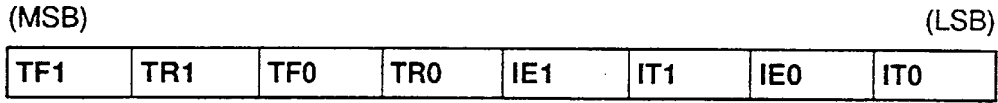
**C/T** Timer or counter selector. Cleared for timer operation (input from internal system clock). Set for Counter operation (input from “Tx” input pin).

M1	M0	Operating Mode
0	0	SAB 8048 timer: “TLx” serves as five-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) TLO is an eight-bit timer/counter controlled by the standard timer 0 control bits; TH0 is an eight-bit timer only controlled by timer 1 control bits. (Timer 1) timer/counter 1 is stopped.

**Figure 17**  
**Timer/Counter Control Register**

**Timer 0/1 Control Register      TCON      SFR-Address 88H**

Default after reset: 00H



- TF1**      Timer 1 overflow flag. Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.
- TR1**      Timer 1 run control bit. Set/cleared by software to turn timer/counter on/off.
- TF0**      Timer 0 overflow flag. Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.
- TR0**      Timer 0 run control bit. Set/cleared by software to turn timer/counter on/off.
- IE1**      Interrupt 1 edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
- IT1**      Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
- IE0**      Interrupt 0 edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
- IT0**      Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

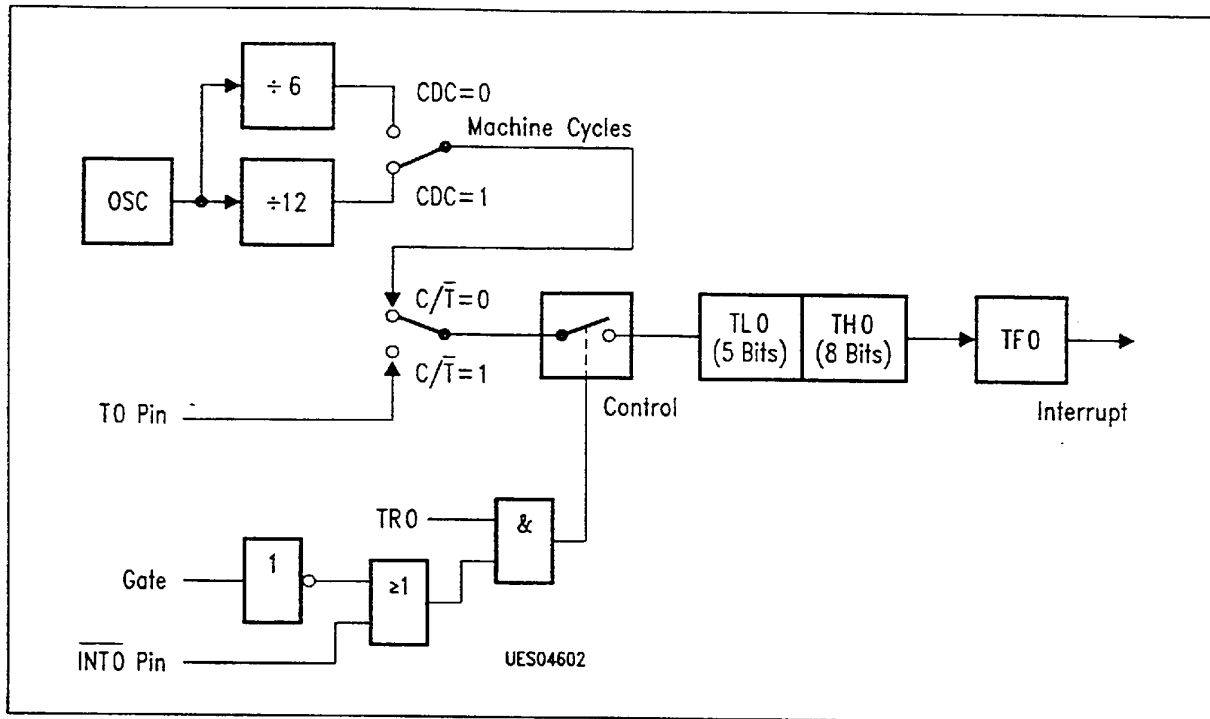


Figure 18  
Timer/Counter 0 Mode 0: 13-Bit Counter

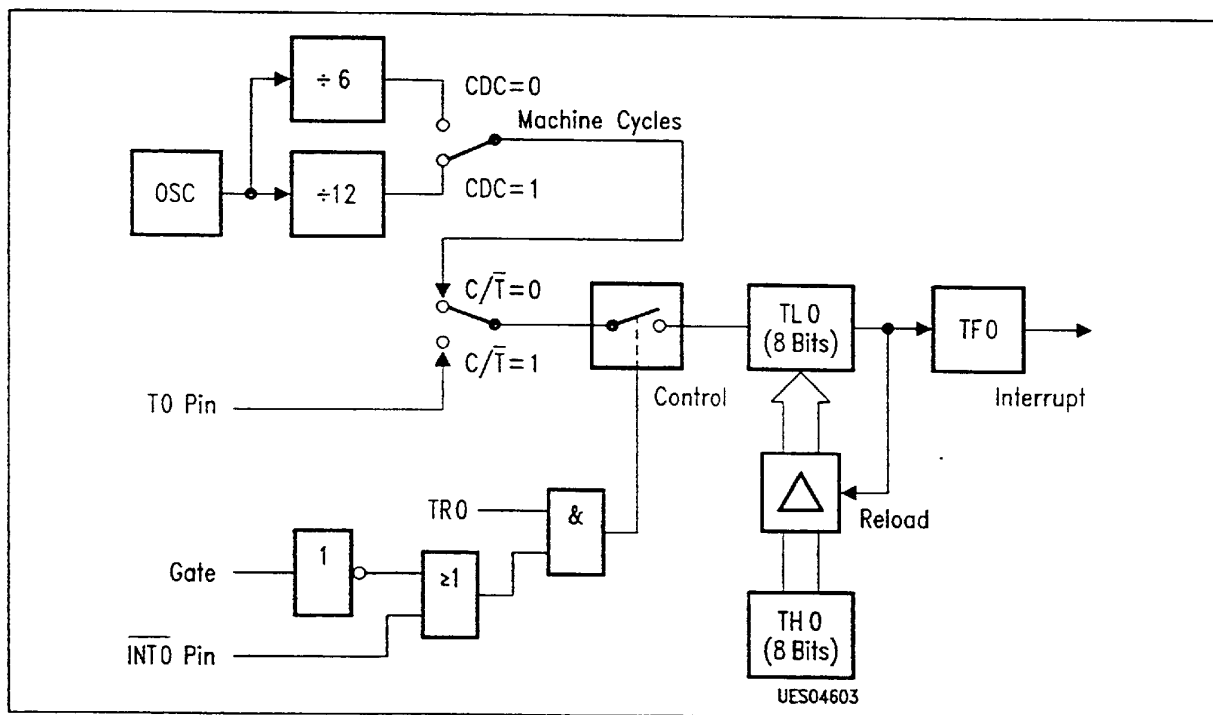


Figure 19  
Timer/Counter 0 Mode 2: 8-Bit Auto-Reload

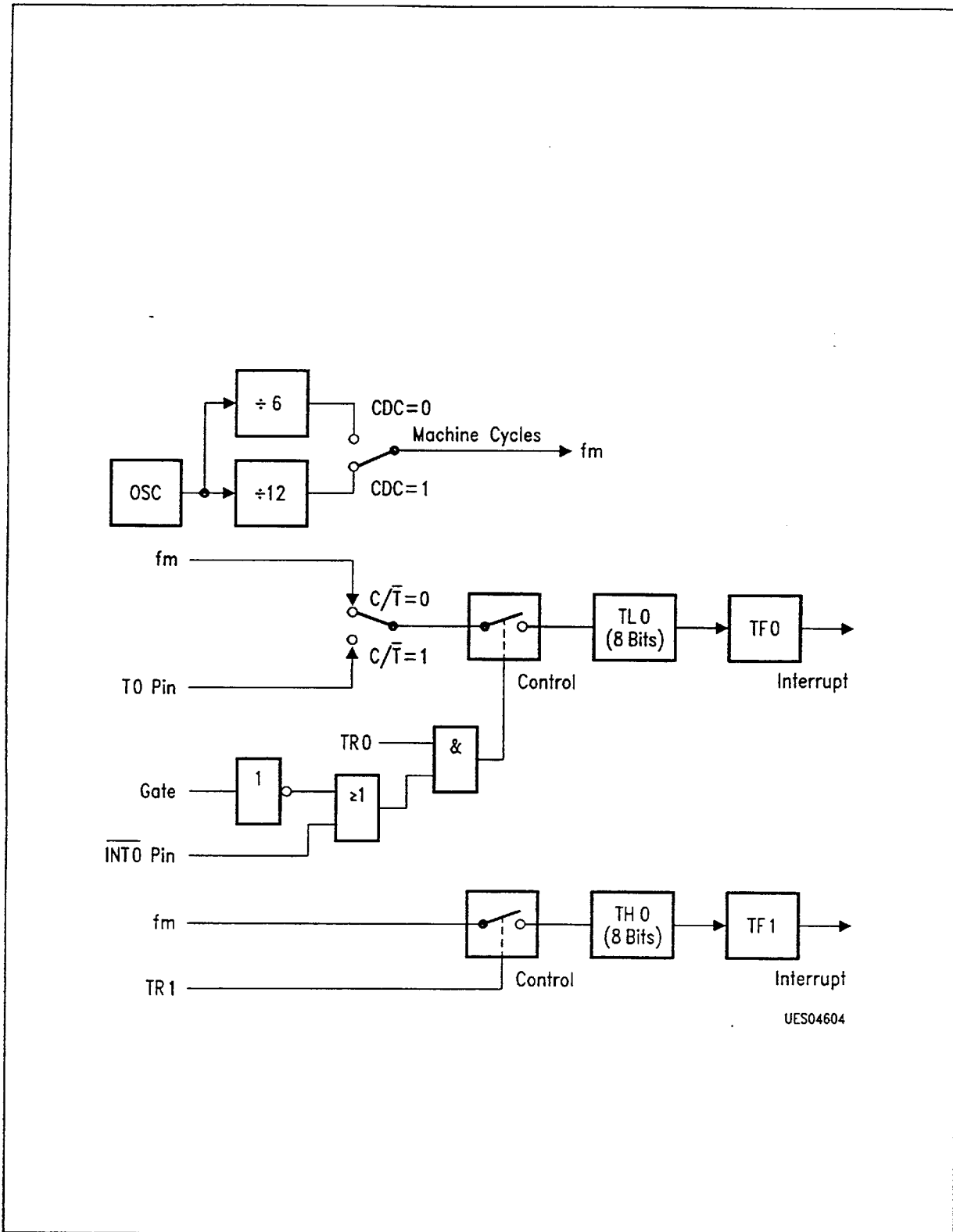


Figure 20  
Timer/Counter 0 Mode 3: Two 8-Bit Counters



**2.7 Watchdog Timer**

To protect the systems 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.

The watchdog timer is a 15-bit timer, which is incremented by a count rate of either  $f_{CYCLE/2}$  or  $f_{CYCLE/128}$ . The latter is enabled by setting bit WDTREL.7. Immediately after start, the watchdog timer is initialized to the reload value programmed to WDTREL.0 – WDTREL.6. After an external reset register WDTREL is cleared to 00H. The lower seven bits of WDTREL can be loaded by software at any time.

The watchdog timer is started by software by setting bit SWDT in special function register WDSTART (bit 6). If the counter is stopped, and WDTREL is loaded with a new value, WDTL (high-byte of the watchdog timer) is updated immediately. WDTL (low-byte of the watchdog timer) is always zero, if the counter is stopped. Once started the watchdog timer cannot be stopped by software but can only be refreshed to the reload value by first setting bit WDT (IE.6) and by the next instruction setting SWDT (WDSTART.6). 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.

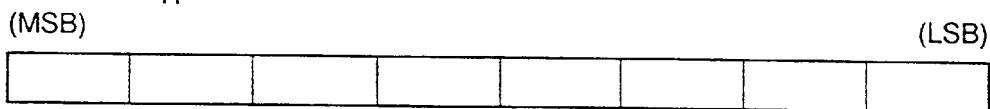
If the software fails to clear the watchdog in time, an internally generated watchdog reset is entered at the counter state 7FFCH. The duration of the reset signal then depends on the prescaler selection. This internal reset differs from an external reset only in so far as the watchdog timer is not disabled and bit WDTS (IP.6) is set. Bit WDTS allows the software to examine from which source the reset was activated. The watchdog timer status flag can also be cleared by software.

With WDTREL = 80H a maximum time period of about 3.1 s at 16-MHz oscillator frequency can be achieved.

**Watchdog Timer Control Bits**

**Watchdog Timer Reload Register WDTREL SFR-Address 86H**

Default after reset: 00H



**WDTREL.7** Prescaler bit. When set, the watchdog is clocked through an additional divide by 64 prescaler.

**WDTREL.0 – WDTREL.6** Seven bit reload value for the high-byte of the watchdog timer. This value is loaded to the WDT when a refresh is triggered by a consecutive setting of bits WDT and SWDT.

**Interrupt Enable Register      IE      SFR-Address A8<sub>H</sub>**

Default after reset: 00<sub>H</sub>



**WDT**      Watchdog timer refresh flag. Set to initiate a refresh of the watchdog timer.  
 Must be set directly before SWDT is set to the watchdog timer.

See chapter 'Interrupt System' for the description of the remaining bits.

**Watchdog Timer Start Register      WDSTART      SFR-Address B8<sub>H</sub>**

Default after reset: xx<sub>H</sub>



**SWDT**      Watchdog timer start flag. Set to activate the watchdog timer. When directly set after setting WDT, a watchdog timer refresh is performed.

**Interrupt Enable Register      IP      SFR-Address A9<sub>H</sub>**

Default after reset: xx<sub>H</sub>



**WDTS**      Watchdog timer reset flag. If bit WDTS is '1' after reset, the reset has been initiated by the watchdog timer.  
 After external reset, WDTS is reset to '0'.

See chapter 'Interrupt System' for the description of the remaining bits.

## 2.8 Serial Interface

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

The frequencies and baud rates described in this chapter depend on the internal system clock, used by the serial interface. The internal system clock frequency of the serial interface is defined by the oscillator frequency  $f_{OSC}$ , the setting of bit CDC in the Advanced Function Register AFR of the special function registers (see chapter "Advanced Function Register"), and the setting of bit PSC in the ADC Control Register ADCON of the special function registers (see chapter "Analog Digital Converter"). Both bits are software switches to activate or deactivate clock dividers by 2. The frequencies and baud rates specified in this chapter apply to bit CDC = 1 and bit PSC = 0. For other combinations of CDC and PSC see the following table:

CDC	PSC	Frequencies and Baud Rates of this Chapter
0	0	double the specified values
0	1	values as specified
1	0	values as specified
1	1	halve the specified values

The serial port can operate in 4 modes:

- Mode 0: Serial data enters and exits through RxD (P3.6). TxD (P3.7) outputs the shift clock at 1/12 of the oscillator frequency.
- Mode 1: 10 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On reception, the stop bit goes into RB8 in special function register SCON. The baud rate is variable.
- Mode 2: 11 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmission, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On reception, the 9th data bit goes into RB8 in the special function register SCON, while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 of the oscillator frequency.
- Mode 3: 11 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit and a stop bit (1). In fact, mode 3 is the same as mode 2 in all respects except the baud rate. The baud rate in mode 3 is variable.

**Figure 21**  
**Serial Port Control Register SCON (98H)**

<b>SM0</b>	<b>SM1</b>	<b>SM2</b>	<b>REN</b>	<b>TB8</b>	<b>RB8</b>	<b>TI</b>	<b>RI</b>	<b>Bit</b>
9FH	9EH	9DH	9CH	9BH	9AH	99H	98H	<b>Address</b>

<b>Symbol</b>	<b>Position</b>	<b>Function</b>
SM0 SM1	SCON.7 SCON.6	Serial Port Mode Selection, see table 4.
SM2	SCON.5	Enables the multiprocessor communication feature in modes 2 and 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0.
REN	SCON.4	Enables serial reception. Set by software to enable reception. Cleared by software to disable reception.
TB8	SCON.3	Is the 9th data bit that will be transmitted in modes 2 and 3. Set or cleared by software as desired.
RB8	SCON.2	In modes 2 and 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.
TI	SCON.1	Is the transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
RI	SCON.0	Is the receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through stop bit time in the other modes, in any serial reception. Must be cleared by software.

**Table 4**  
**Serial Port Mode Selection**

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift Reg.	$f_{OSC12}$
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	$f_{OSC164} - f_{OSC32}$
1	1	3	9-bit UART	Variable

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit if REN = 1. The control, mode, and status bits of the serial port in special function register SCON are illustrated in **figure 21**.

### 2.8.1 Multiprocessor Communication

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

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

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

### 2.8.2 Baud Rates

The baud rate in mode 0 is fixed:

Mode 0 baud rate =

$$\frac{f_{osc}}{12}$$

The baud rate in mode 2 depends on the value of bit SMOD in special function register PCON (bit 7). If SMOD = 0 (which is the value on reset), the baud rate is 1/64 of the oscillator frequency. If SMOD = 1, the baud rate is 1/32 of the oscillator frequency. Contrary to the SAB 8051 SMOD is placed on SFR-address 97H.

Mode 2 baud rate =

$$\frac{2^{SMOD}}{64} \times f_{osc}$$

The baud rates in modes 1 and 3 are determined by the timer 1 overflow rate or can be generated by the internal baud rate generator.

When timer 1 is used as the baud rate generator, the baud rates in modes 1 and 3 are determined by the timer 1 overflow rate and the value of SMOD as follows:

Modes 1,3 baud rate =

$$\frac{2^{SMOD}}{32} \times \text{Time 1 overflow rate}$$

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

Modes 1,3 baud rate =

$$\frac{2^{SMOD}}{32} \times \frac{f_{osc}}{12 \times (256 - TH1)}$$

One can achieve very low baud rates with timer 1 by leaving the timer 1 interrupt enabled, configuring the timer to run as a 16-bit timer (high nibble of TMOD = 0001B), and using the timer 1 interrupt to do a 16-bit software reload. Table 5 lists various commonly used baud rates and how they can be obtained from timer 1.

**Table 5**  
**Generated Commonly Used Baud Rates**

Baud Rate	$f_{osc}$ MHz	SMOD	Timer 1		
			CT	Mode	Reload Value
Mode 0 max: 1.33 MHz	16.0	X	X	X	X
Mode 2 max: 500 Kbaud	16.0	1	X	X	X
Mode 1, 3: 62.5 Kbaud	12.0	1	0	2	FF <sub>H</sub>
19.2 Kbaud	11.059	1	0	2	FD <sub>H</sub>
9.6 Kbaud	11.059	0	0	2	FD <sub>H</sub>
4.8 Kbaud	11.059	0	0	2	FA <sub>H</sub>
2.4 Kbaud	11.059	0	0	2	F4 <sub>H</sub>
1.2 Kbaud	11.059	0	0	2	E8 <sub>H</sub>
137.5 Baud	11.986	0	0	2	1D <sub>H</sub>
110 Baud	6.0	0	0	2	72 <sub>H</sub>
110 Baud	12.0	0	0	1	FE <sub>EB</sub> <sub>H</sub>

### 2.8.3 More about Mode 0

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

Figure 22 shows a simplified functional diagram of the serial port in mode 0, and associated timing.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write-to-SBUF" signal at S6P2 also loads a 1 into the 9th bit position of the transmit shift register and tells the TX-control block to commence a transmission. The internal timing is such that one full machine cycle will elapse between "write-to-SBUF" and activation of SEND. SEND enables the output of the shift register to the alternate output function line of P3.6, and also enables SHIFT CLOCK to the alternate output function, line of P3.7. SHIFT CLOCK is low during S3, S4 and S5 of every machine cycle, and high during S6, S1, and S2. At S6P2 of every machine cycle in which SEND is active, the contents of the transmit shift register is shifted one position to the right.

As data bits shift out to the right, zeros come in from the left. When the MSB of the data byte is at the output position of the shift register, then the 1 that was initially loaded into the 9th position, is just left of the MSB, and all positions to the left of that contain zeros.

This condition flags the TX-control block to do one last shift and then deactivate SEND and set TI. Both of these actions occur at S1P1 in the 10th machine cycle after "write-to-SBUF".

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

RECEIVE enables SHIFT CLOCK to the alternate output function line of P3.7. SHIFT CLOCK makes transitions at S3P1 and S6P1 in every machine cycle. At S6P2 of every machine cycle in which RECEIVE is active, the contents of the Receive Shift register are shifted one position to the left. The value that comes in from the right is the value that was sampled at the P3.6 pin at S5P2 in the same machine cycle.

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



### 2.8.4 More about Mode 1

Ten bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first) and a stop bit (1). On reception, the stop bit goes into RB8 in SCON.

The baud rate is determined by the timer 1 overflow rate.

**Figure 23** shows a simplified functional diagram of the serial port in mode 1, and associated timings for transmit and receive.

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

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

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

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

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

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

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

If either of these two conditions is not met, the received frame is irretrievably lost. If both conditions are met, the stop bit goes into RB8, the 8 data bits go into SBUF and RI is activated. At this time, no matter whether the above conditions are met or not, the unit goes back looking for a 1-to-0 transition in RxD.

### 2.8.5 More about Modes 2 and 3

11 bits are transmitted (through TxD), or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit, (1). On transmission, the 9th data bit (TB8) can be assigned the value of 0 or 1. On reception, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to either 1/32 or 1/64 of the oscillator frequency in mode 2. Mode 3 may have a variable baud rate generated from timer 1.

Figure 24 and 25 show a functional diagram of the serial port in modes 2 and 3 and associated timings. The receive portion is exactly the same as in mode 1. The transmit portion differs from mode 1 only in the 9th bit of the transmit shift register.

Transmission is initiated by any instruction that uses SBUF as a destination register. The "write-to-SBUF" signal also loads TB8 into the 9th bit position of the transmit shift register and flags the TX-control unit that a transmission is requested. Transmission commences at S1P1 of the machine cycle following the next rollover in the divide-by-16 counter (thus, the bit times are synchronized to the divide-by-16 counter, not to the "write-to-SBUF" signal).

The transmission begins with activation of  $\overline{\text{SEND}}$ , which puts the start bit to TxD. One bit time later, DATA is activated which enables the output bit of the transmit shift register to TxD. The first shift pulse occurs one bit time after that. The first shift clocks a 1 (the stop bit) into the 9th bit position of the shift register. Thereafter, only zeros are clocked in. Thus, as data bits shift out to the right, zeros are clocked in from the left. When TB8 is at the output position of the shift register, then the stop bit is just left of the TB8, and all positions to the left of that contain zeros.

This condition flags the TX-control unit to do one last shift and then deactivate  $\overline{\text{SEND}}$  and set TI. This occurs at the 11th divide-by-16 rollover after "write-to-SBUF".

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

At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RxD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back looking for another 1-to-0 transition. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed. As data bits come in from the right, 1s shift out to the left. When the start bit arrives at the leftmost position in the shift register (which in modes 2 and 3 is a 9-bit register), it flags the RX-control block to do one last shift, load SBUF and RB8, and set RI. The signal to load SBUF and RB8, and to set RI, will be generated if, and only if, the following conditions are met at the time the final shift pulse is generated:

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

If either of these two conditions is not met, the received frame is irretrievably lost, and RI is not set. If both conditions are met, the received 9th data bit goes into RB8, the first 8 data bits go into SBUF. One bit time later, no matter whether the above conditions are met or not, the unit goes back looking for a 1-to-0-transition at the RxD input.

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

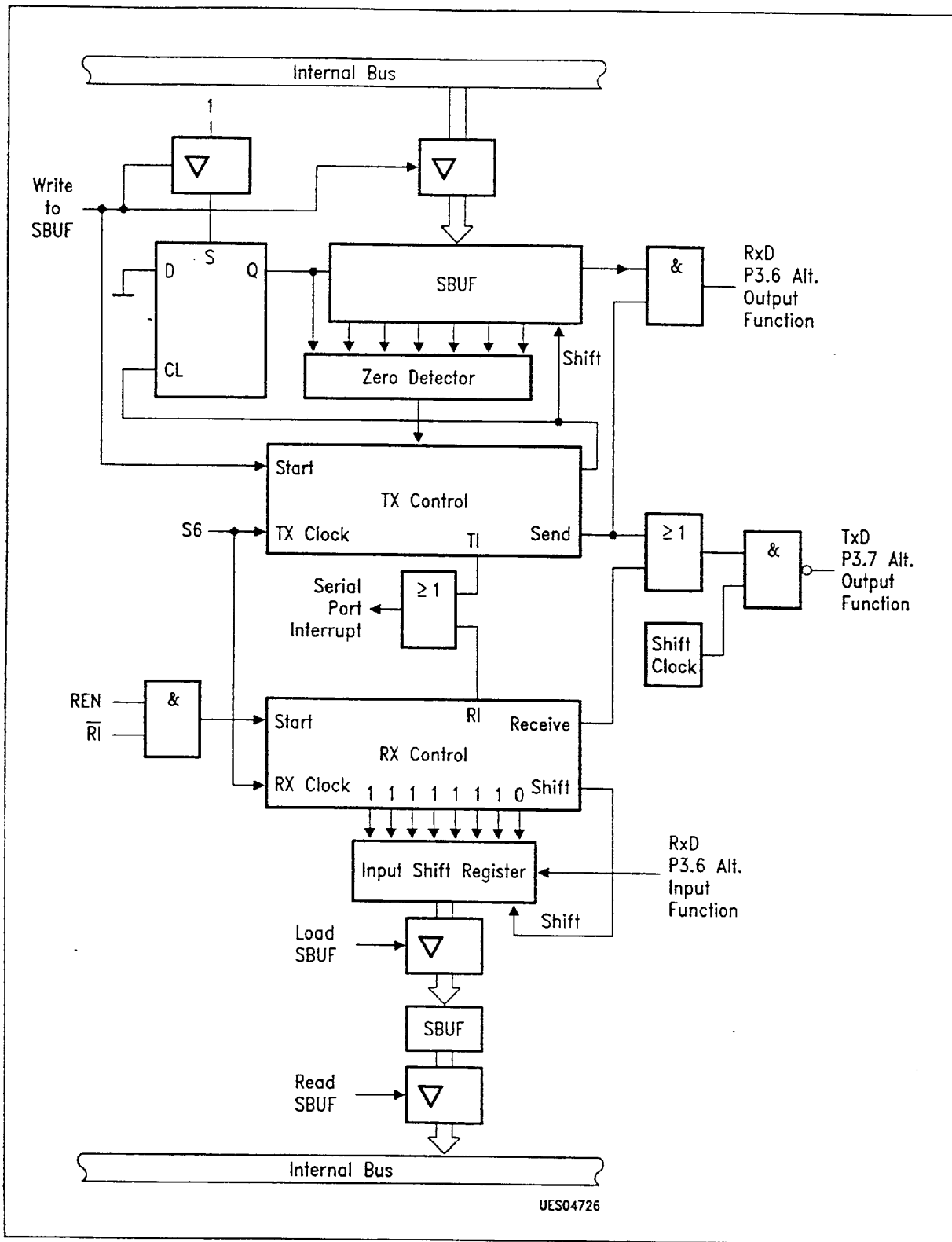


Figure 22 a  
Serial Port Mode 0, Functional Diagram

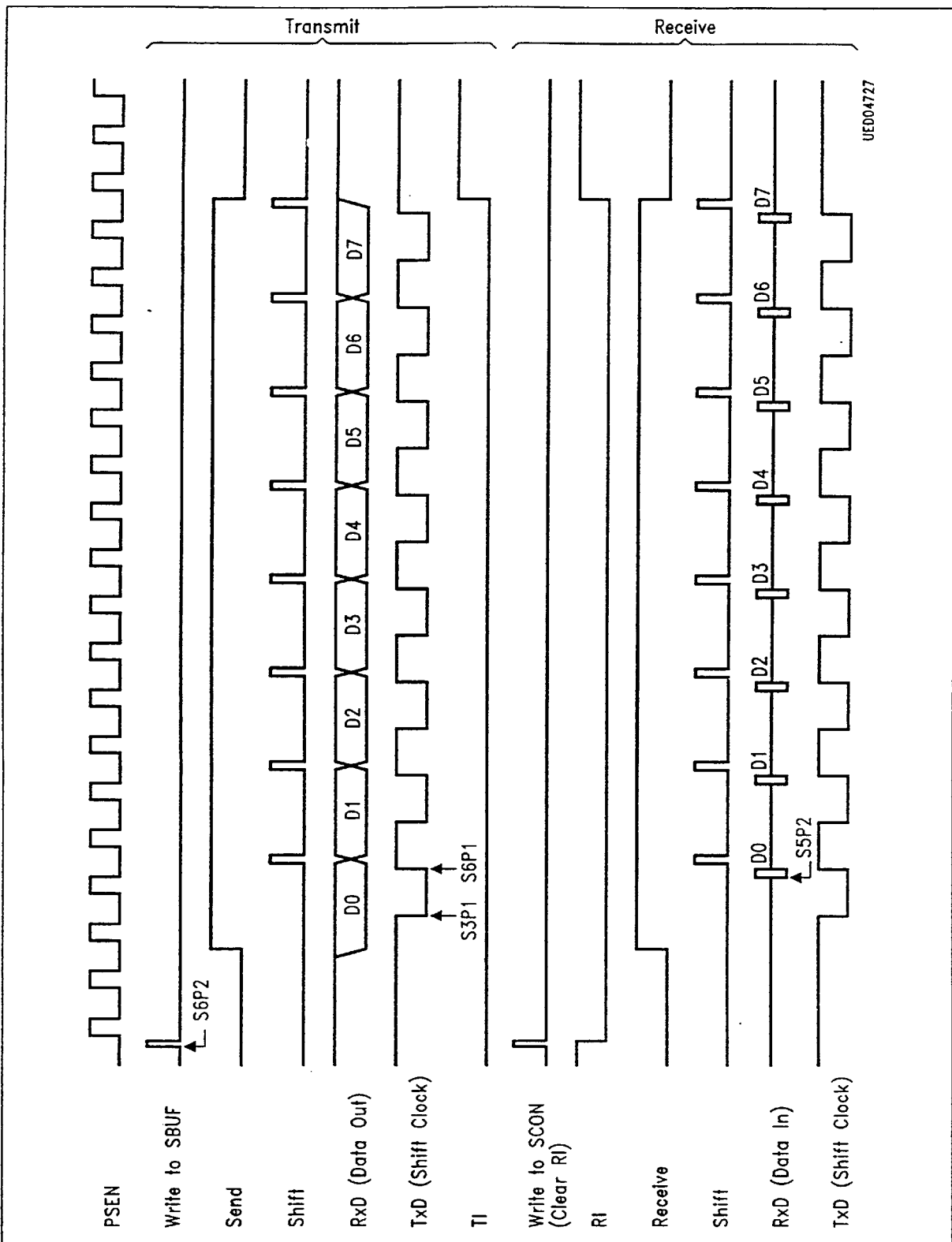


Figure 22 b  
Serial Port Mode 0, Timing

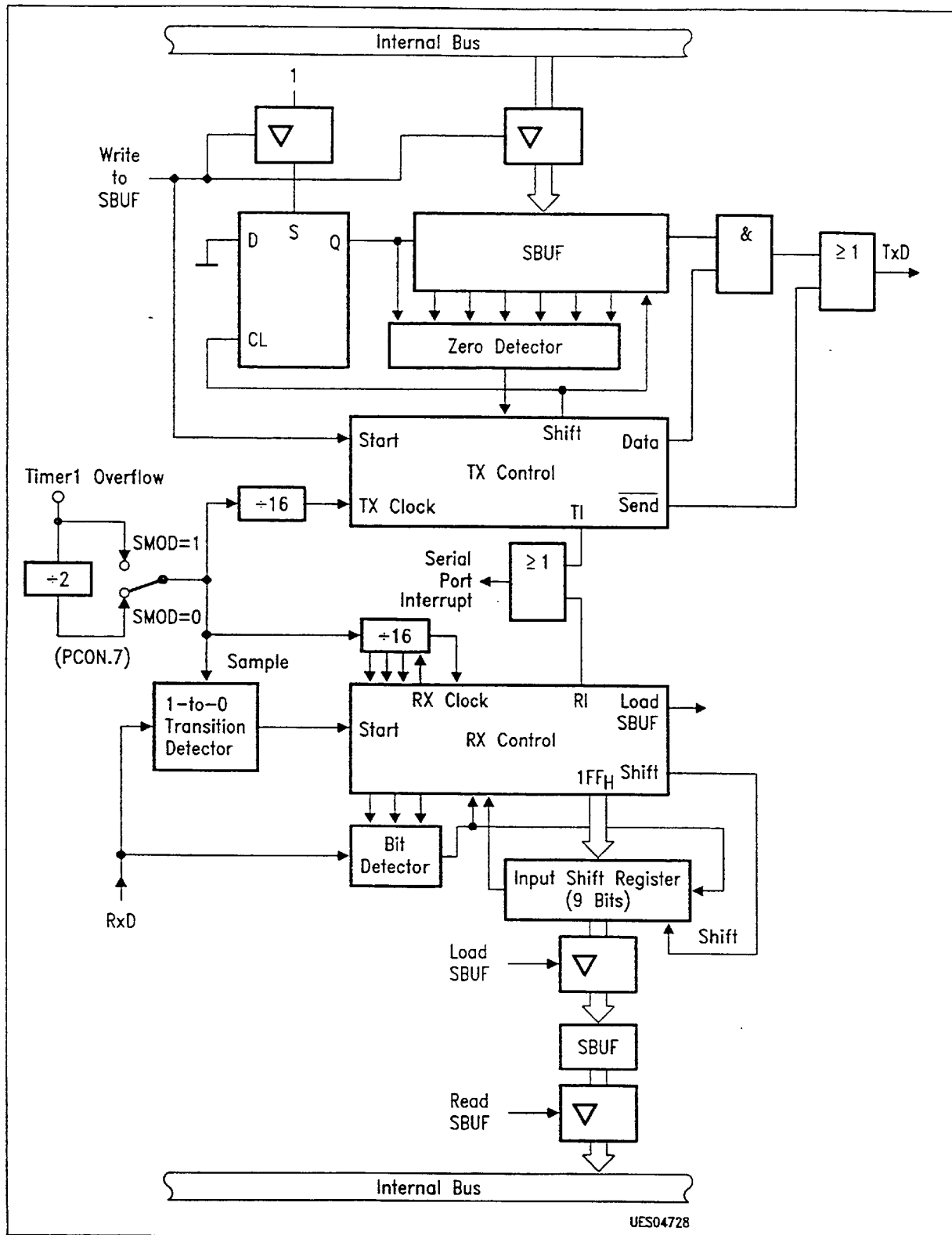


Figure 23 a  
Serial Port Mode 1, Functional Diagram

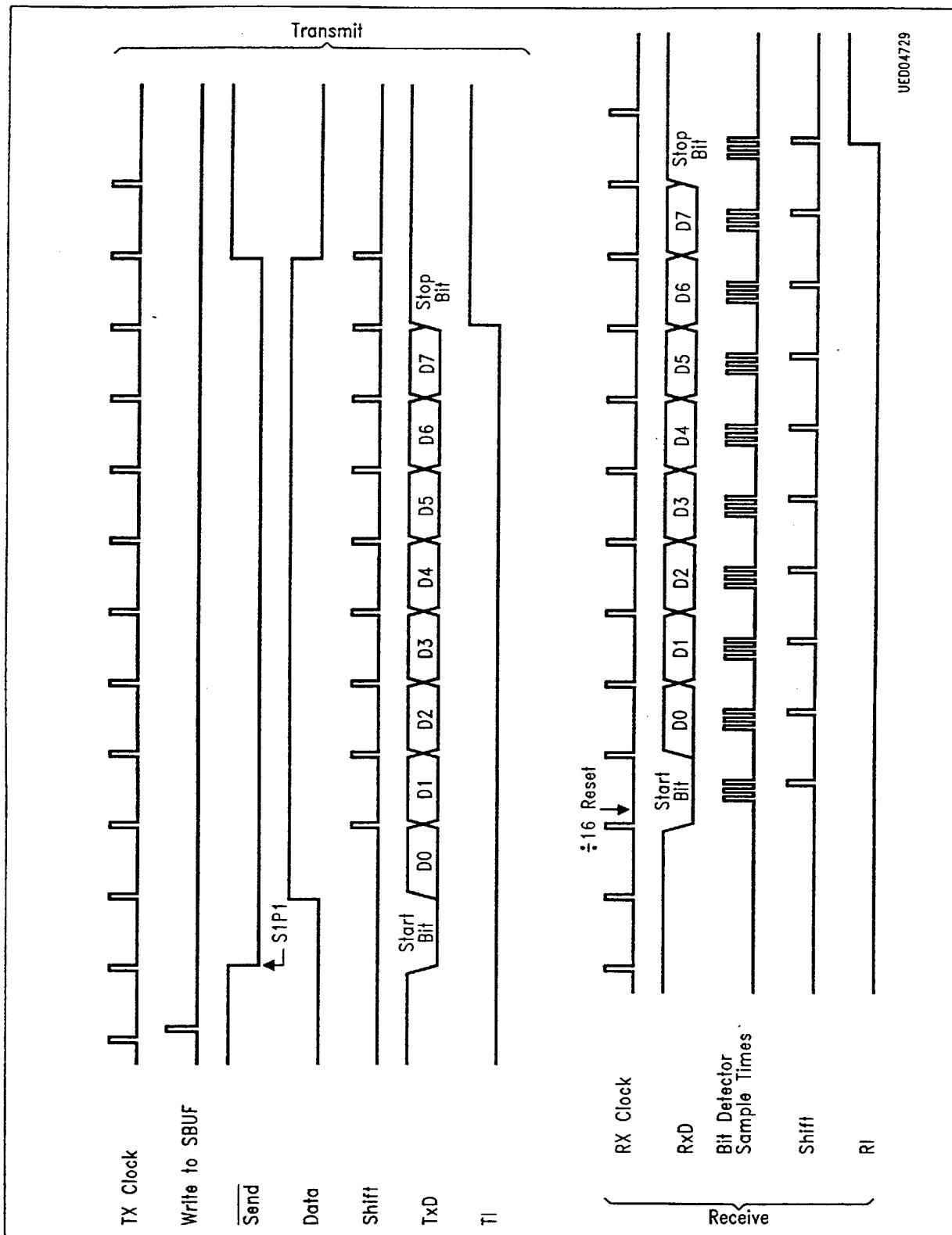


Figure 23 b  
Serial Port Mode 1, Timing

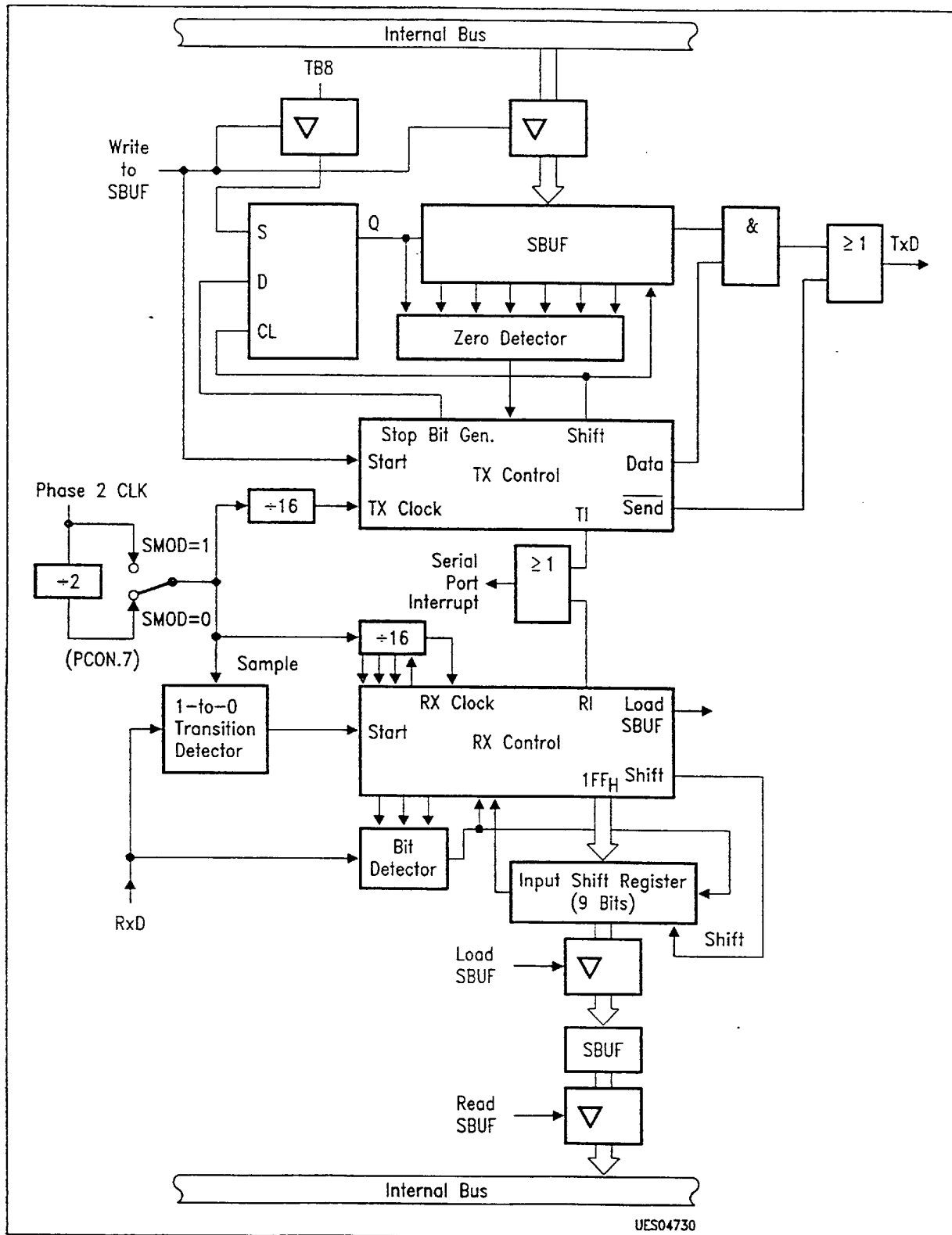


Figure 24 a  
Serial Port Mode 2, Functional Diagram

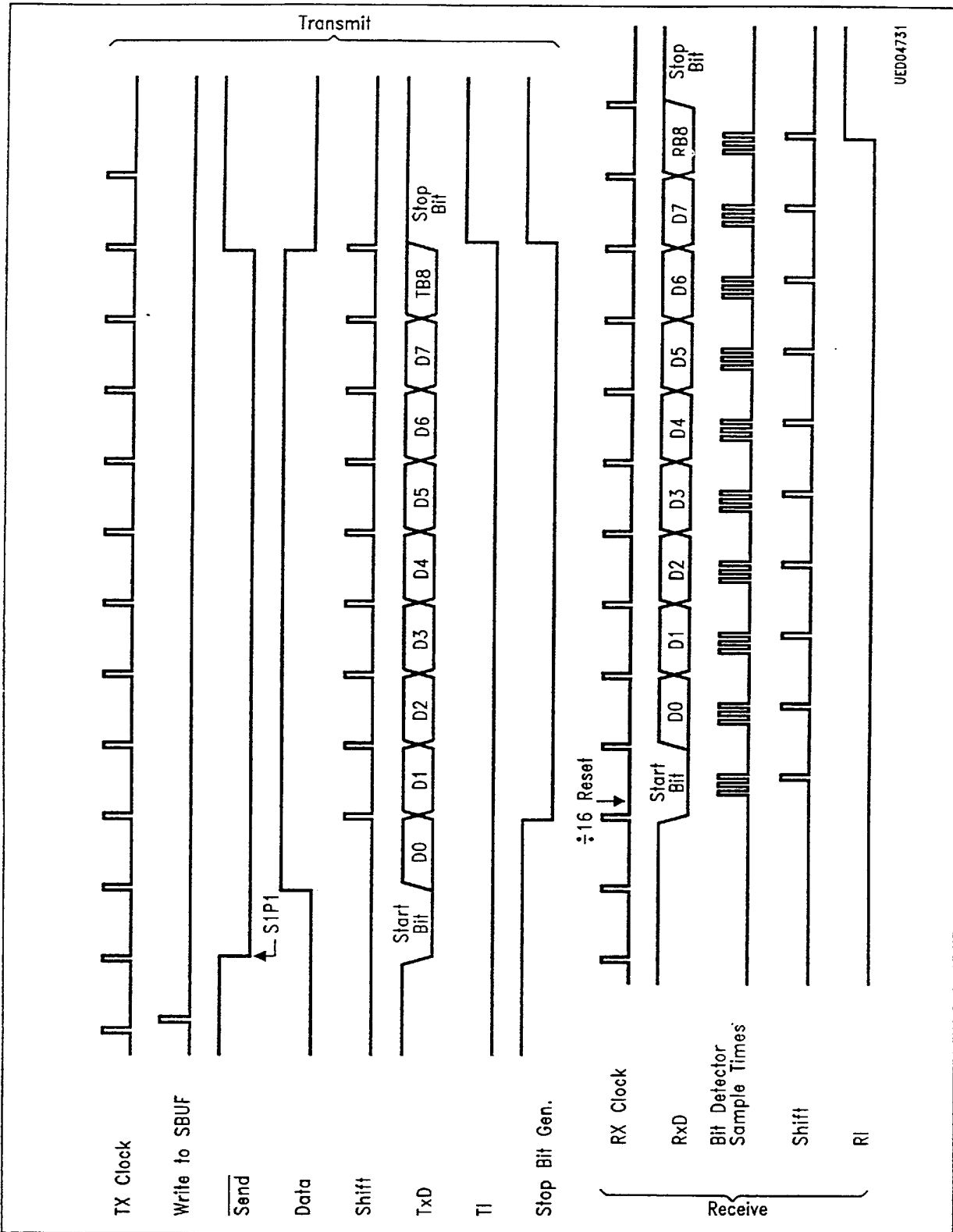


Figure 24 b  
Serial Port Mode 2, Timing



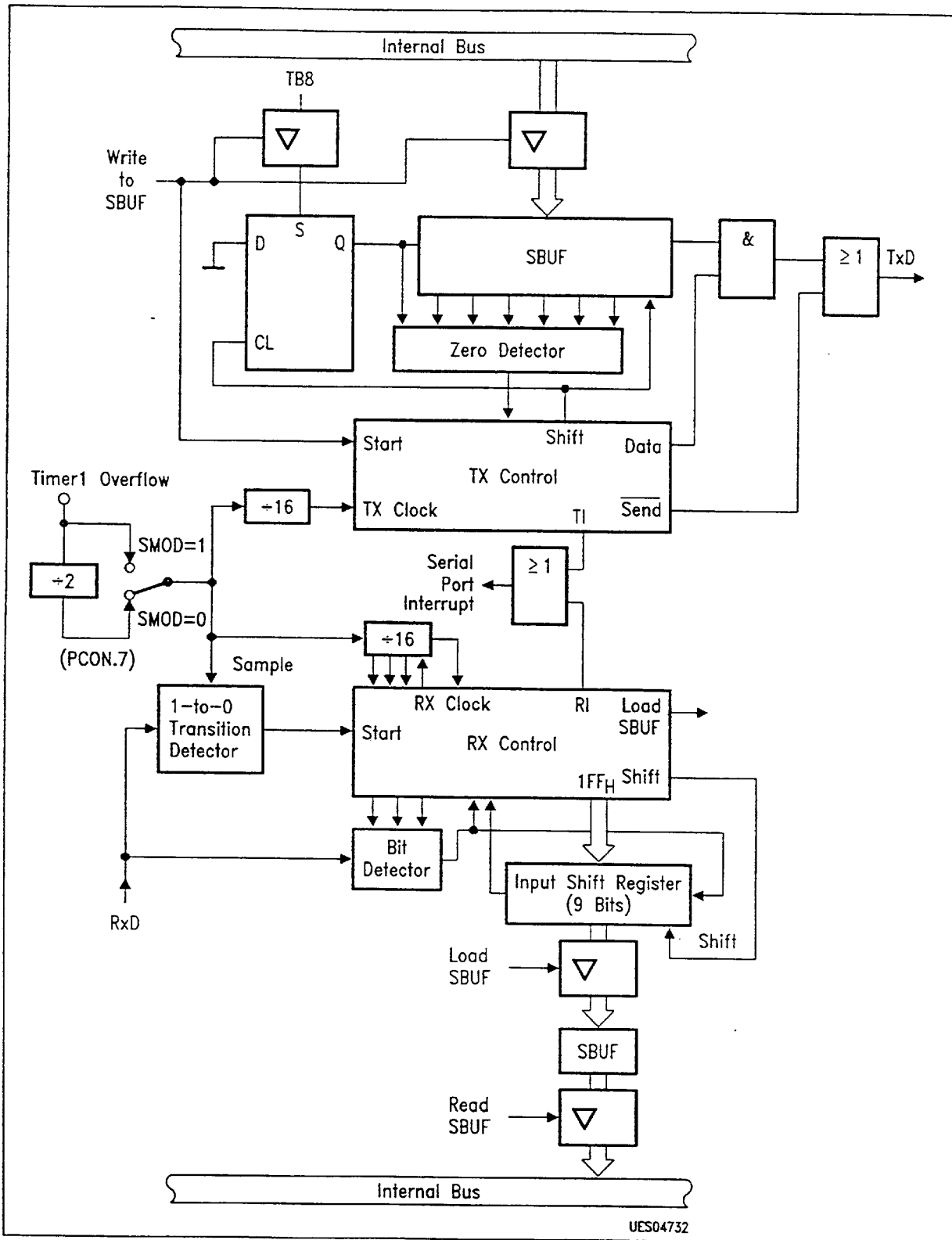


Figure 25 a  
Serial Port Mode 3, Functional Diagram

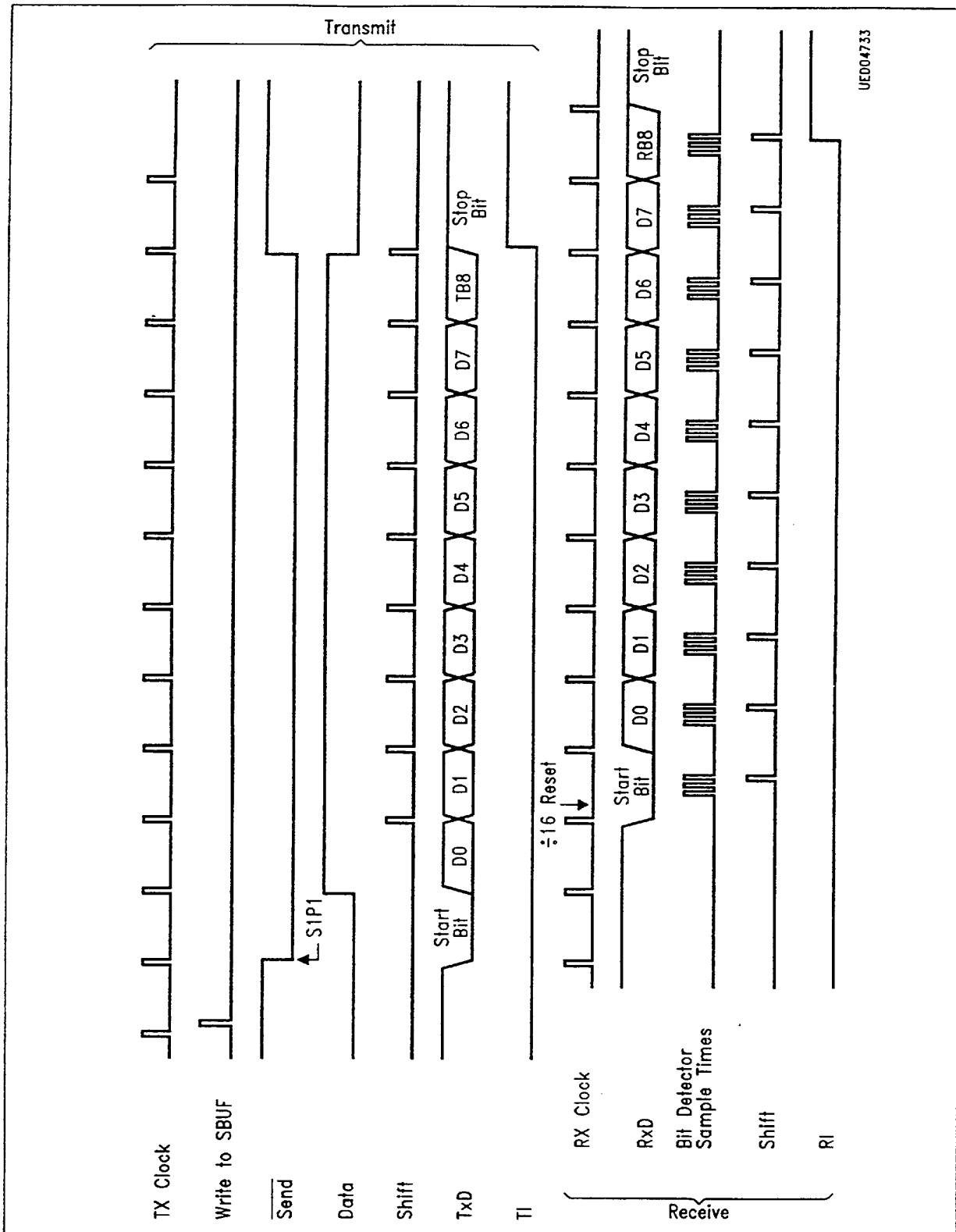


Figure 25 b  
Serial Port Mode 3, Timing

**2.9 Pulse Width Modulation Unit**

The PWM-unit provides eight independent digital to analog conversion channels, with helpful time resolution flexibility. Controlled via special function registers, each channel can be enabled individually. Due to the modulator's flexibility the output frequency can be switched to 23.4 kHz, 46.9 kHz and 93.8 kHz by reducing time resolution ( $f_{osc} = 12 \text{ MHz}$ ). This is done by decreasing the timer width from 8 to 7 or 6 bits.

**General Considerations**

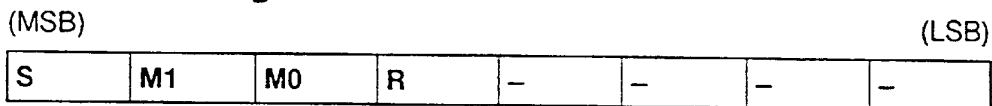
The PWM-output channels are placed as alternate functions to the eight lines of port 1 (P1.0 ... P1.7). Each PWM-channel can be individually switched between PWM-function and port function. The PWM-unit is controlled by the special function register PWMC located at address 0C8H. This register determines the counter's resolution (6, 7 or 8 bit) and starts or stops the counter. A counter status bit can be read and an interrupt enable flag can be set. Except for the status bit, read and write accesses are possible for this register. The PWMC-register's lowest 3 bits are not employed and can be used as extra software flags (C0, C1, C2).

The eight 8-bit compare registers PWCOMP0 – PWCOMP7 located at SFR-addresses 0F1H – 0F8H contain the modulation ratios of the output signals which are related to the maximum defined by the counter's resolution. These compare registers are double buffered and a new compare value will only be taken into the main register, if the PWM-timer is stopped or after the next timer overflow. To avoid overwriting the desired compare value, the counter status bit should be checked before a new write operation to a compare register is done.

The PWM-timer register located at SFR-address 0F9H contains the actual value of the PWM-counter and can only be read by the CPU. Every compare register, which is not employed for the PWM-output can be used as an additional register. This is not allowed for register PWME. If the PWM-function is not activated, the PWM-timer is available for any other timing purpose.

**PWM-Control Register PWMC** SFR-Address C8H

Default after reset: 1000xxxxB



Function of the control bits:

- R** = 0: The PWM-timer is stopped and reset to 00H. All output latches (OL0 ... OL7) are set to 1.
- = 1: The PWM-timer is set to RUN. At timer overflow, all output latches OL0 ... OL7 are set to 1. If the timer value meets the compare value of channel i, OLi is reset to 0.

**M1, M0** Control the output frequency and resolution of the PWM-unit.

M1	M0	Output Frequency	Resolution
0	0	$f_{osc} / (2 \times 256)$	8 bit
0	1	$f_{osc} / (2 \times 128)$	7 bit
1	0	$f_{osc} / (2 \times 64)$	6 bit

**S** Shows the actual state of the PWM-timer. S is set by PWM-timer overflow and has to be reset by software. This bit may be used to control whether a value selected for a compare channel was already written into the compare latch by a PWM-timer overflow.

**PWMC.0 – PWMC.3** reserved

**PWM-Enable Register PWME** SFR-Address C0<sub>H</sub>

Default after reset: 00<sub>H</sub>

(MSB)

(LSB)

E7	E6	E5	E4	E3	E2	E1	E0
----	----	----	----	----	----	----	----

**E<sub>i</sub>** (i = 0 ... 7) = 0: The corresponding PWM-channel is disabled.  
P1.i functions as normal bidirectional I/O-port.  
= 1: The corresponding PWM-channel is enabled. P1.i is automatically set to logic 1 and is connected to the output latch of the corresponding PWM-channel (OL<sub>i</sub>).

### PWM-Compare Registers PWCOMPx

Each of the eight compare channels consists of

- An 8-bit register with read and write access from the CPU.

The SFR-addresses are:

PWCOMP0: 0F1<sub>H</sub>  
PWCOMP1: 0F2<sub>H</sub>  
PWCOMP2: 0F3<sub>H</sub>  
PWCOMP3: 0F4<sub>H</sub>  
PWCOMP4: 0F5<sub>H</sub>  
PWCOMP5: 0F6<sub>H</sub>  
PWCOMP6: 0F7<sub>H</sub>  
PWCOMP7: 0F8<sub>H</sub>

After reset, the register contents are 0FF<sub>H</sub>.

- An 8-bit compare latch, which is loaded with the value in the 8-bit register, if the PWM-timer overflows or stops.
- A comparator, which compares the value of the compare latch with the timer value. If (M1, M0) ≠ (0, 0), only the 7 (or the 6) least significant bits will be compared.

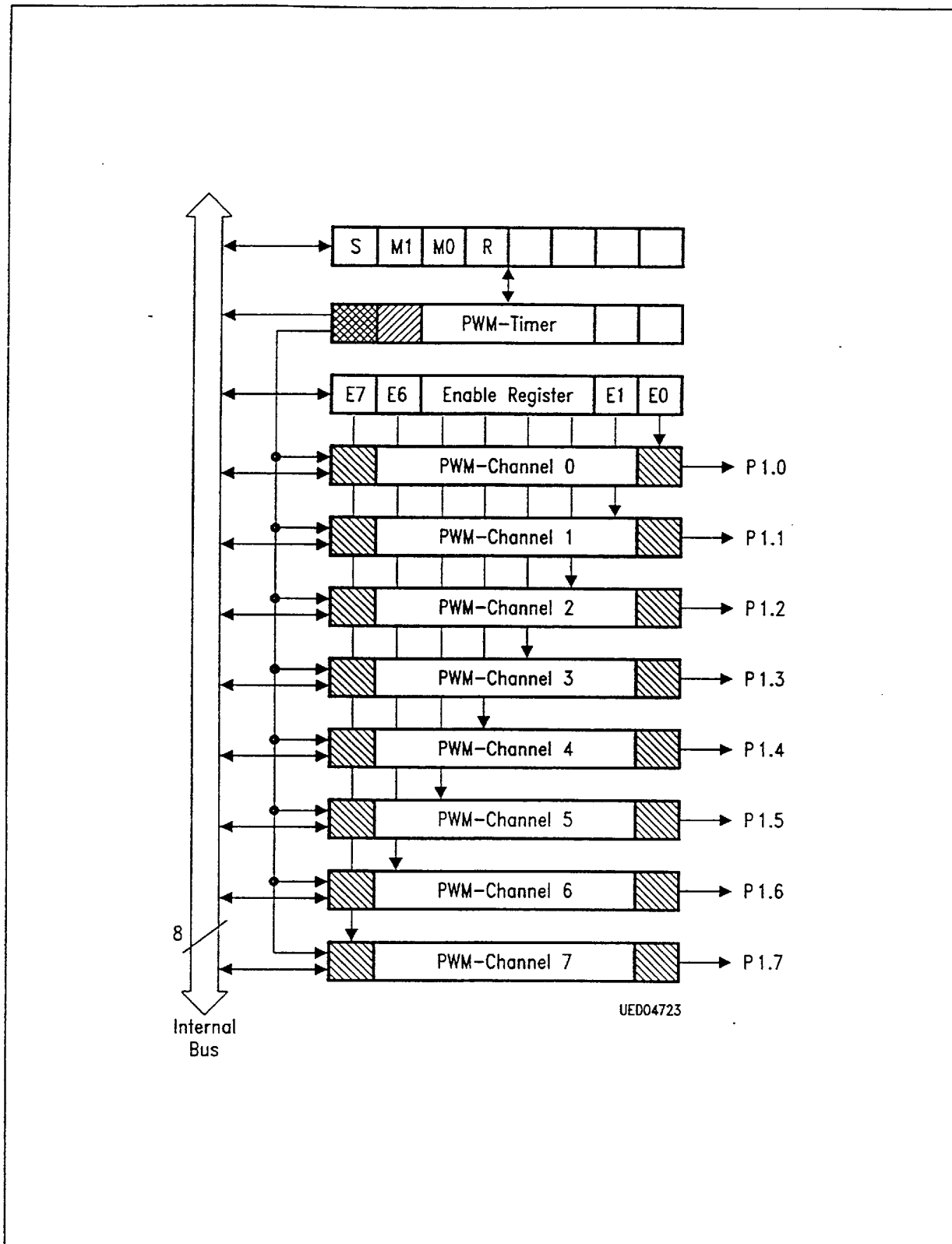
- A one bit output latch, which is set on PWM-timer overflow or stopped and reset on the compare event. The output latch controls the corresponding port pin, when the channel is enabled. RESET sets the output latch to 1.

### PWM-Timer Register PWCOUNT

(Address 0F9<sub>H</sub>, Reset value 00<sub>H</sub>)

An 8 bit upwards counting binary counter (with an input frequency of  $f_{osc}/2$ ) is provided as PWM-timer. The counter registers can be read by software at SFR-address 0F9<sub>H</sub>, but cannot be written to. If in PWMC-register R = 0, the PWM-timer will be held at 00<sub>H</sub>, i.e. at the reset value. If R = 1, the PWM-timer will increment six times every CPU-instruction cycle. M1 and M0 (in PWMC-register) control the value, from which the PWM-timer overflows to 00<sub>H</sub>:

M1	M0	Overflow Value
0	0	0FF <sub>H</sub> (= 255)
0	1	07F <sub>H</sub> (= 127)
1	0	03F <sub>H</sub> (= 63)



**Figure 26**  
Block Diagram of Pulse Width Modulation Unit

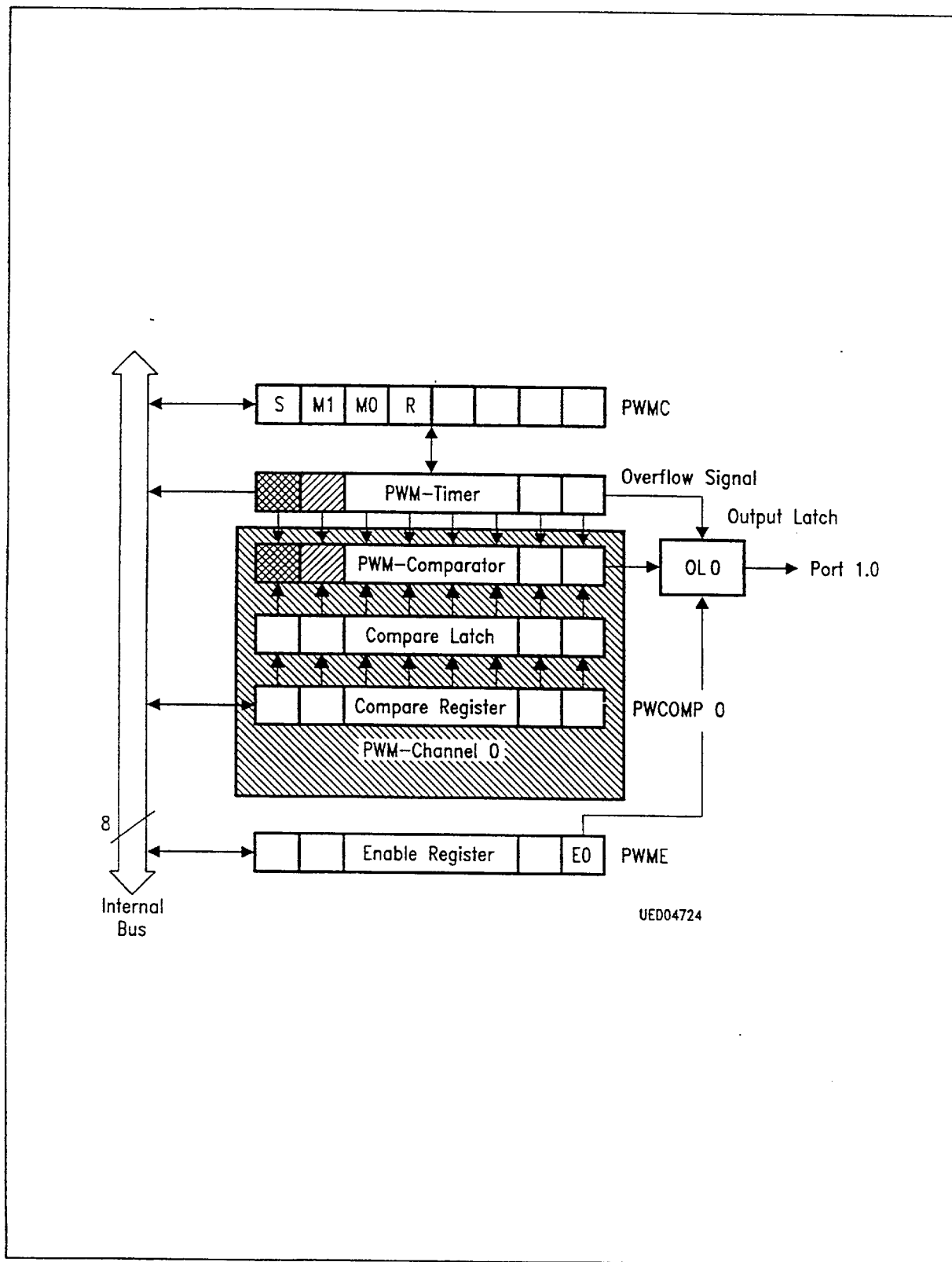


Figure 27  
Block Diagram of One Pulse Width Modulation Channel (e.g. PWM0)

**2.10 Analog Digital Converter**

The controller provides an A/D-converter with the following features:

- 4 multiplexed input channels, which can also be used as digital inputs
- 8-bit resolution
- 6.5 to 26 μs conversion time at 12 MHz oscillator frequency

The conversion time depends on the internal system clock, used by the ADC. The clock-frequency of the internal ADC system clock is defined by the external quartz (oscillator frequency  $f_{osc}$ ), the setting of bit CDC in the Advanced Function Register AFR of the special function registers (SFR) (see chapter "Advanced Function Register"), and the setting of bit PSC in the ADC Control Register ADCON (SFR). Both bits are software switches to activate or deactivate clock dividers by 2. The conversion time can be calculated by:

$$t_{conversion} = \frac{78 \times 2^{CDC} \times 2^{PSC}}{f_{osc}}$$

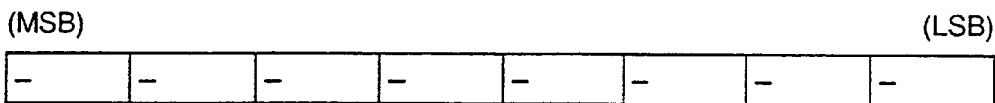
For the conversion, the method of successive approximation via capacitor array is used. There are three user accessible special function registers: ADCON, ADDAT and DAPR.

Special function register ADCON is used to set the operation modes, to check the status and to select one of four input channels. ADCON contains two mode bits. Bit ADM is used to choose the single or continuous conversion method. In single conversion mode (ADM = 0) only one conversion is performed after starting, while in continuous conversion mode (ADM = 1) after the first start a new conversion is automatically started on completion of the previous one.

The busy flag BSY (ADCON.4) is automatically set when a conversion is in progress. After completion of the conversion it is reset by hardware. This flag can be read only, a write has no effect. MX0 and MX1 are used to select one of 4 A/D-channels. With PSC a divide by two prescaler for the internal clock system of the ADC and Serial Interface only, can be activated. For PSC = 0 the internal chip-clock is used as master clock for the ADC and Serial Interface. For PSC = 1 the internal chip-clock is divided by two by the prescaler before being used as master clock for the ADC and Serial Interface.

The special function register ADDAT holds the converted digital 8-bit data result. The data remains in ADDAT until it is overwritten by the next converted data. ADDAT can be read or written under software control. A start of conversion is triggered by a write-to DAPR instruction. The data written must be 00H.

ADC-Start Register                      DAPR    SFR-Address DA<sub>H</sub>



Only the address of DAPR is used to decode a start-of-conversion signal. No bits are implemented. A read from DAPR might show random values.



### ADC-Control Register    **ADCON**

SFR-Address D8<sub>H</sub>

Default after reset: 0xx00000<sub>B</sub>

(MSB)				(LSB)			
<b>PSC</b>	-	-	<b>BSY</b>	<b>ADM</b>	0	<b>MX1</b>	<b>MX0</b>

This register is bit addressable.

- PSC**            Prescaler control: PSC = 0 for prescaler not active. Internal master clock of ADC and Serial Interface are equal the internal chip clock. PSC = 1 for prescaler active. Internal master clock of ADC and Serial Interface are at half of the internal chip clock.
- ADCON.6**      Reserved
- ADCON.5**      Reserved
- ADCON.2**      Always to be written with '0'
- BSY**            Busy flag; = 1, during conversion
- ADM**            ADC-conversion mode: ADM = 0 for single and ADM = 1 for continuous conversion.
- MX1, MX0**     ADC-channel select

<b>MX1</b>	<b>MX0</b>	<b>Selected Channel</b>
0	0	0
0	1	1
1	0	2
1	1	3

**Note:**

After changing the input channel, the input signal has to stabilize before a new conversion is started.

### ADC-Data Register    **ADDAT**

SFR-Address D9<sub>H</sub>

Default after reset: undefined

(MSB)				(LSB)			
<b>AD7</b>	<b>AD6</b>	<b>AD5</b>	<b>AD4</b>	<b>AD3</b>	<b>AD2</b>	<b>AD1</b>	<b>AD0</b>

8-Bit Analog Data Value

**2.11 Advanced Function Register**

The on-chip clock generator of the SDA 30C163-2 contains the same clock divider, found in every 8051 compatible design. The clock divider divides the external clock frequency (oscillator frequency) by 2. To enhance clock performance by either doubling the internal clock frequency or by keeping the internal frequency constant and halving the external quartz-frequency, the divider can be switched off by software. As software-switch for the divider a new Special-Function-Register (SFR) has been defined:

**Advanced Function Register AFR**

**SFR-Address A6H**

Default after reset: FF<sub>H</sub>

(MSB)

(LSB)

<b>CDC</b>	1	1	1	1	1	1	1
------------	---	---	---	---	---	---	---

**CDC** Clock divider control bit. If set, the clock divider is on. The internal clock frequency is half the external oscillator frequency. If cleared, the clock divider is off. The internal clock frequency is equal to the external oscillator frequency.

**AFR.0 – AFR.6** Reserved, always to be written with '1'.

**Note:**

The current implementation allows a write access to the AFR-register only!

## 2.12 Instruction Set

The assembly language uses the same instruction set and the same instruction opcodes as the 8051 microcomputer family.

### 2.12.1 Notes on Data Addressing Modes

- Rn – Working register R0 – R7.
- direct – 128 internal RAM-locations, any I/O-port, control or status register.
- @Ri – Indirect internal RAM-location addressed by register R0 or R1.
- #data – 8-bit constant included in instruction.
- #data 16 – 16-bit constant included as bytes 2 & 3 of instruction.
- bit – 128 software flags, any I/O-pin, control or status bit in special function registers.

Operations working on external data memory (MOVX ...) are used to access the additional 1024 bytes of the extended internal data RAM (XRAM).

### 2.12.2 Notes on Program Addressing Modes

- addr 16 – Destination address for LCALL & LJMP may be anywhere within the program memory address space.
- addr 11 – Destination address for ACALL & AJMP will be within the same 2 Kbyte of the following instruction.
- rel – SJMP and all conditional jumps include an 8-bit offset byte. Range is + 127/ – 128 bytes relative to first byte of the following instruction.

## 2.12.3 Instruction Set Description

## Arithmetic Operations

Mnemonic		Description	Byte
ADD	A, Rn	Add register to Accumulator	1
ADD	A, direct	Add direct byte to Accumulator	2
ADD	A, @Ri	Add indirect RAM to Accumulator	1
ADD	A, #data	Add immediate data to Accumulator	2
ADDC	A, Rn	Add register to Accumulator with Carry flag	1
ADDC	A, direct	Add direct byte to A with Carry flag	2
ADDC	A, @Ri	Add indirect RAM to A with Carry flag	1
ADDC	A, #data	Add immediate data to A with Carry flag	2
SUBB	A, Rn	Subtract register from A with Borrow	1
SUBB	A, direct	Subtract direct byte from A with Borrow	2
SUBB	A, @Ri	Subtract indirect RAM from A with Borrow	1
SUBB	A, #data	Subtract immediate data from A with Borrow	2
INC	A	Increment Accumulator	1
INC	Rn	Increment register	1
INC	direct	Increment direct byte	2
INC	@Ri	Increment indirect RAM	1
DEC	A	Decrement Accumulator	1
DEC	Rn	Decrement register	1
DEC	direct	Decrement direct byte	2
DEC	@Ri	Decrement indirect RAM	1
INC	DPTR	Increment Data Pointer	1
MUL	AB	Multiply A & B	1
DIV	AB	Divide A & B	1
DA	A	Decimal Adjust Accumulator	1

## Logical Operations

Mnemonic	Description	Byte
ANL    A, Rn	AND register to Accumulator	1
ANL    A, direct	AND direct byte to Accumulator	2
ANL    A, @Ri	AND indirect RAM to Accumulator	1
ANL    A, #data	AND immediate data to Accumulator	2
ANL    direct, A	AND Accumulator to direct byte	2
ANL    direct, #data	AND immediate data to direct byte	3
ORL    A, Rn	OR register to Accumulator	1
ORL    A, direct	OR direct byte to Accumulator	2
ORL    A, @Ri	OR indirect RAM to Accumulator	1
ORL    A, #data	OR immediate data to Accumulator	2
ORL    direct, A	OR Accumulator to direct byte	2
ORL    direct, #data	OR immediate data to direct byte	3
XRL    A, Rn	Exclusive-OR register to Accumulator	1
XRL    A, direct	Exclusive-OR direct byte to Accumulator	2
XRL    A, @Ri	Exclusive-OR indirect RAM to Accumulator	1
XRL    A, #data	Exclusive-OR immediate data to Accumulator	2
XRL    direct, A	Exclusive-OR Accumulator to direct byte	2
XRL    direct, #data	Exclusive-OR immediate data to direct	3
CLR    A	Clear Accumulator	1
CPL    A	Complement Accumulator	1
RL    A	Rotate Accumulator left	1
RLC    A	Rotate A left through the Carry flag	1
RR    A	Rotate Accumulator right	1
RRC    A	Rotate A right through Carry flag	1
SWAP   A	Swap nibbles within the Accumulator	1

## Data Transfer Operations

Mnemonic	Description	Byte
MOV A, Rn	Move register to Accumulator	1
MOV A, direct	Move direct byte to Accumulator	2
MOV A, @Ri	Move indirect RAM to Accumulator	1
MOV A, #data	Move immediate data to Accumulator	2
MOV Rn, A	Move Accumulator to register	1
MOV Rn, direct	Move direct byte to register	2
MOV Rn, #data	Move immediate data to register	2
MOV direct, A	Move Accumulator to direct byte	2
MOV direct, Rn	Move register to direct byte	2
MOV direct, direct	Move direct byte to direct	3
MOV direct, @Ri	Move indirect RAM to direct byte	2
MOV direct, #data	Move immediate data to direct byte	3
MOV @Ri, A	Move Accumulator to indirect RAM	1
MOV @Ri, direct	Move direct byte to indirect RAM	2
MOV @Ri, #data	Move immediate data to indirect RAM	2
MOV DPTR, #data 16	Load Data Pointer with a 16-bit constant	3
MOVC A@A + DPTR	Move Code byte relative to DPTR to Accumulator	1
MOVC A@A + PC	Move Code byte relative to PC to Accumulator	1
MOVX A, @Ri	Move External RAM (8-bit addr) to Accumulator	1
MOVX A, @DPTR	Move External RAM (16-bit addr) to Accumulator	1
MOVX @Ri, A	Move A to External RAM (8-bit addr)	1
MOVX @DPTR, A	Move A to External RAM (16-bit addr)	1
PUSH direct	Push direct byte onto stack	2
POP direct	Pop direct byte from stack	2
XCH A, Rn	Exchange register with Accumulator	1
XCH A, direct	Exchange direct byte with Accumulator	2
XCH A, @Ri	Exchange indirect RAM with Accumulator	1
XCHD A, @Ri	Exchange low-order digital indirect RAM with A	1

## Boolean Variable Manipulation

Mnemonic	Description	Byte
CLR C	Clear Carry flag	1
CLR bit	Clear direct bit	2
SETB C	Set Carry flag	1
SETB bit	Set direct bit	2
CPL C	Complement Carry flag	1
CPL bit	Complement direct bit	2
ANL C, bit	AND direct bit to Carry flag	2
ANL C, /bit	AND complement of direct bit to Carry	2
ORL C, bit	OR direct bit to Carry flag	2
ORL C, /bit	OR complement of direct bit to Carry	2
MOV C, bit	Move direct bit to Carry flag	2
MOV bit, C	Move Carry flag to direct bit	2

## Program and Machine Control Operations

Mnemonic	Description	Byte
ACALL addr 11	Absolute subroutine call	2
LCALL addr 16	Long subroutine call	3
RET	Return from subroutine	1
RETI	Return from interrupt	1
AJMP addr 11	Absolute jump	2
LJMP addr 16	Long jump	3
SJMP rel	Short jump (relative addr)	2
JMP @A + DPTR	Jump indirect relative to the DPTR	1
JZ rel	Jump if Accumulator is zero	2
JNZ rel	Jump if Accumulator is not zero	2
JC rel	Jump if Carry flag is set	2
JNC rel	Jump if Carry flag is not set	2
JB bit, rel	Jump if direct bit set	3
JNB bit, rel	Jump if direct bit not set	3
JBC bit, rel	Jump if direct bit is set and clear bit	3
CJNE A, direct rel	Compare direct to A and jump if not equal	3

## Program and Machine Control Operations (cont'd)

Mnemonic	Description	Byte
CJNE A, #data, rel	Compare immediate to A and jump if not equal	3
CJNE Rn, #data, rel	Compare immediate to register and jump if not equal	3
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal	3
DJNZ Rn, rel	Decrement register and jump if not zero	2
DJNZ direct, rel	Decrement direct and jump if not zero	3
NOP	No operation	1

### 2.12.4 Instruction Opcodes in Hexadecimal Order

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4



### Instruction Opcodes in Hexadecimal Order (cont'd)

Hex Code	Number of Bytes	Mnemonic	Operands
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A, #data
25	2	ADD	A, data addr
26	1	ADD	A, @R0
27	1	ADD	A, @R1
28	1	ADD	A, R0
29	1	ADD	A, R1
2A	1	ADD	A, R2
2B	1	ADD	A, R3
2C	1	ADD	A, R4
2D	1	ADD	A, R5
2E	1	ADD	A, R6
2F	1	ADD	A, R7
30	3	JNB	bit addr, code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A, #data
35	2	ADDC	A, data addr
36	1	ADDC	A, @R0
37	1	ADDC	A, @R1
38	1	ADDC	A, R0
39	1	ADDC	A, R1
3A	1	ADDC	A, R2
3B	1	ADDC	A, R3
3C	1	ADDC	A, R4
3D	1	ADDC	A, R5
3E	1	ADDC	A, R6
3F	1	ADDC	A, R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr., A
43	3	ORL	data addr, #data
44	2	ORL	A, #data
45	2	ORL	A, data addr
46	1	ORL	A, @R0
47	1	ORL	A, @R1

## Instruction Opcodes in Hexadecimal Order (cont'd)

Hex Code	Number of Bytes	Mnemonic	Operands
48	1	ORL	A, R0
49	1	ORL	A, R1
4A	1	ORL	A, R2
4B	1	ORL	A, R3
4C	1	ORL	A, R4
4D	1	ORL	A, R5
4E	1	ORL	A, R6
4F	1	ORL	A, R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr, A
53	3	ANL	data addr, #data
54	2	ANL	A, #data
55	2	ANL	A, data addr
56	1	ANL	A, @R0
57	1	ANL	A, @R1
58	1	ANL	A, R0
59	1	ANL	A, R1
5A	1	ANL	A, R2
5B	1	ANL	A, R3
5C	1	ANL	A, R4
5D	1	ANL	A, R5
5E	1	ANL	A, R6
5F	1	ANL	A, R7
60	2	JZ	code addr
61	2	AJMP	code addr.
62	2	XRL	data addr, A
63	3	XRL	data addr, #data
64	2	XRL	A, #data
65	2	XRL	A, data addr
66	1	XRL	A, @R0
67	1	XRL	A, @R1
68	1	XRL	A, R0
69	1	XRL	A, R1
6A	1	XRL	A, R2
6B	1	XRL	A, R3
6C	1	XRL	A, R4
6D	1	XRL	A, R5
6E	1	XRL	A, R6
6F	1	XRL	A, R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C, bit addr

## Instruction Opcodes in Hexadecimal Order (cont'd)

Hex Code	Number of Bytes	Mnemonic	Operands
73	1	JMP	@A + DPTR
74	2	MOV	A, #data
75	3	MOV	data addr, #data
76	2	MOV	@R0, #data
77	2	MOV	@R1, #data
78	2	MOV	R0, #data
79	2	MOV	R1, #data
7A	2	MOV	R2, #data
7B	2	MOV	R3, #data
7C	2	MOV	R4, #data
7D	2	MOV	R5, #data
7E	2	MOV	R6, #data
7F	2	MOV	R7, #data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C, bit addr
83	1	MOVC	A, @A + PC
84	1	DIV	AB
85	3	MOV	data addr, data addr
86	2	MOV	data addr, @R0
87	2	MOV	data addr, @R1
88	2	MOV	data addr, R0
89	2	MOV	data addr, R1
8A	2	MOV	data addr, R2
8B	2	MOV	data addr, R3
8C	2	MOV	data addr, R4
8D	2	MOV	data addr, R5
8E	2	MOV	data addr, R6
8F	2	MOV	data addr, R7
90	3	MOV	DPTR, #data 16
91	2	ACALL	code addr
92	2	MOV	bit addr, C
93	1	MOVC	A, @A + DPTR
94	2	SUBB	A, #data
95	2	SUBB	A, data addr
96	1	SUBB	A, @R0
97	1	SUBB	A, @R1
98	1	SUBB	A, R0
99	1	SUBB	A, R1
9A	1	SUBB	A, R2
9B	1	SUBB	A, R3
9C	1	SUBB	A, R4
9D	1	SUBB	A, R5

### Instruction Opcodes in Hexadecimal Order (cont'd)

Hex Code	Number of Bytes	Mnemonic	Operands
9E	1	SUBB	A, R6
9F	1	SUBB	A, R7
A0	2	ORL	C, /bit addr
A1	2	AJMP	code addr
A2	2	MOV	C, bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5	-	reserved	
A6	2	MOV	@R0, data addr
A7	2	MOV	@R1, data addr
A8	2	MOV	R0, data addr
A9	2	MOV	R1, data addr
AA	2	MOV	R2, data addr
AB	2	MOV	R3, data addr
AC	2	MOV	R4, data addr
AD	2	MOV	R5, data addr
AE	2	MOV	R6, data addr
AF	2	MOV	R7, data addr
B0	2	ANL	C, /bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A, #data, code addr
B5	3	CJNE	A, data addr, code addr
B6	3	CJNE	@R0, #data, code addr
B7	3	CJNE	@R1, #data, code addr
B8	3	CJNE	R0, #data, code addr
B9	3	CJNE	R1, #data, code addr
BA	3	CJNE	R2, #data, code addr
BB	3	CJNE	R3, #data, code addr
BC	3	CJNE	R4, #data, code addr
BD	3	CJNE	R5, #data, code addr
BE	3	CJNE	R6, #data, code addr
BF	3	CJNE	R7, #data, code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A, data addr
C6	1	XCH	A, @R0
C7	1	XCH	A, @R1
C8	1	XCH	A, R0

## Instruction Opcodes in Hexadecimal Order (cont'd)

Hex Code	Number of Bytes	Mnemonic	Operands
C9	1	XCH	A, R1
CA	1	XCH	A, R2
CB	1	XCH	A, R3
CC	1	XCH	A, R4
CD	1	XCH	A, R5
CE	1	XCH	A, R6
CF	1	XCH	A, R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr, code addr
D6	1	XCHD	A, @R0
D7	1	XCHD	A, @R1
D8	2	DJNZ	R0, code addr
D9	2	DJNZ	R1, code addr
DA	2	DJNZ	R2, code addr
DB	2	DJNZ	R3, code addr
DC	2	DJNZ	R4, code addr
DD	2	DJNZ	R5, code addr
DE	2	DJNZ	R6, code addr
DF	2	DJNZ	R7, code addr
E0	1	MOVX	A, @DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A, @R0
E3	1	MOVX	A, @R1
E4	1	CLR	A
E5	2	MOV	A, data addr
E6	1	MOV	A, @R0
E7	1	MOV	A, @R1
E8	1	MOV	A, R0
E9	1	MOV	A, R1
EA	1	MOV	A, R2
EB	1	MOV	A, R3
EC	1	MOV	A, R4
ED	1	MOV	A, R5
EE	1	MOV	A, R6
EF	1	MOV	A, R7
F0	1	MOVX	@DPTR, A
F1	2	ACALL	code addr
F2	1	MOVX	@R0, A
F3	1	MOVX	@R1, A

Instruction Opcodes in Hexadecimal Order (cont'd)

Hex Code	Number of Bytes	Mnemonic	Operands
F4	1	CPL	A
F5	2	MOV	data addr, A
F6	1	MOV	@R0, A
F7	1	MOV	@R1, A
F8	1	MOV	R0, A
F9	1	MOV	R1, A
FA	1	MOV	R2, A
FB	1	MOV	R3, A
FC	1	MOV	R4, A
FD	1	MOV	R5, A
FE	1	MOV	R6, A
FF	1	MOV	R7, A

### 2.13 Operational Restrictions

There is a problem when an internal or external event causes a write access to a special function register or a bit within a special function register and the CPU is executing a 2-cycle read-modify-write instruction on the same register or bit. The problem occurs when the access is concurrent to the 2nd cycle of the 2-cycle read-modify-write instruction. There is no problem when external RAM or ports are accessed.

An internal event for example can be a timer overflow. The corresponding overflow flag would be set. However, when the CPU is executing the 2nd cycle of a 2-cycle read-modify-write instruction at the same time, setting the overflow flag is suppressed because of reading a '0' modifying it to '0' and writing a '0' in the 2nd cycle of the 2-cycle read-modify-write instruction. A timer overflow interrupt will not be generated or a polling routine will miss this event.

A level triggered external interrupt is always recognized as an interrupt and can always be polled. An edge triggered external interrupt would only be recognized or correctly detected by polling if the workarounds are used. The timer overflow bits TF0 and TF1 are always lost by polling of TCON.

**Table 1** shows the affected special function registers or bits within a special function register when the CPU is executing a 2-cycle read-modify-write instruction on this register or bit. **Table 2** lists the instructions which are causing this problem.

**Table 1**  
**Affected Bits in Special Function Registers**

Special Function Register	Address	Affected Bits
TCON	88 <sub>H</sub>	TF0, TF1, IE0, IE1
SCON	98 <sub>H</sub>	TI, RI, RB8
SBUF	99 <sub>H</sub>	all bits of the receive buffer
ADCON	D8 <sub>H</sub>	BSY
THx, TLx		all bits, if timer is running

**Table 2**  
**2-Cycle Read-Modify-Write Instructions which are Causing this Problem**

ANL dadr, #const8	MOV badr, C
ORL dadr, #const8	DJNZ dadr, rel
XRL dadr, #const8	JBC badr, rel

**Workaround:**

Normally, the SFRs SBUF, TLx and THx are not accessed by a 2-cycle read-modify-write instruction, mentioned in **table 2**. The other SFRs are bit-addressable. If possible, replace accesses to these registers by bit-instructions (SETB, CLR). The logical operations could be replaced by the following instruction sequences:

ANL dadr, #const8	=	MOV A, #const8 ANL dadr, A
ORL dadr, #const8	=	MOV A, #const8 ORL dadr, A
XRL dadr, #const8	=	MOV A, #const8 XRL dadr, A
MOV badr, C	=	JC set CLR badr JMP ok set: SETB badr ok:
JBC badr, rel	=	JB badr, rel ..... rel: CLR badr

DJNZ could not be replaced by an instruction sequence. But this seems to be unnecessary.



### 3 Electrical Characteristics

#### 3.1 Absolute Maximum Ratings

Parameter	Symbol	Limit Values	Unit
Voltage on any pin with respect to ground ( $V_{SS}$ )	$V_S$	- 0.5 to 7	V
Power dissipation	$P_{tot}$	1	W
Ambient temperature under bias	$T_A$	- 20 to 85	°C
Storage temperature	$T_{sig}$	- 65 to 125	°C

#### 3.2 DC-Characteristics

$T_A = -20$  to  $85$  °C;  $V_{DD} = 5\text{ V} \pm 10\%$ ,  $V_{SS} = 0\text{ V}$   
 ( $C_L = 100\text{ pF}$  for port 0 and PSEN,  $80\text{ pF}$  for all other outputs)

Parameter	Symbol	Limit Values		Units	Test Condition
		min.	max.		
L-input voltage	$V_{IL}$	- 0.5	0.8	V	
H-input voltage (all except XTAL1)	$V_{IH}$	2.0	$V_{DD} + 0.5$	V	
H-input voltage (XTAL1)	$V_{IH1}$	$0.7 V_{DD}$	$V_{DD} + 0.5$	V	
L-output voltage	$V_{OL}$	-	0.45	V	$I_{OL} = 3.2\text{ mA}$
H-output voltage (ports 1, 3, 4 P24...P27)	$V_{OH}$	2.4	-	V	$I_{OH} = -40\text{ }\mu\text{A}$
H-output voltage (A0 ... A16, PWM-/UART-mode, ALE, PSEN)	$V_{OH1}$	2.4	-	V	$I_{OH} = -1.6\text{ mA}$
Logical 0 input current (D0...D7, ports 1,3,4, P24...P27)	$I_{IL}$	- 50	- 200	$\mu\text{A}$	$V_{IN} = 0.45\text{ V}$
Input leakage current (port 0, P20...P23)	$I_{LI}$		$\pm 10$	$\mu\text{A}$	$0.45\text{ V} \leq V_{IN} \leq V_{DD}$
Power supply current	$I_{DD}$		50	mA	$V_{DD} = 5\text{ V};$ $f_{osc} = 16\text{ MHz}$

## DC-Characteristics (cont'd)

$T_A = -20$  to  $85$  °C;  $V_{DD} = 5$  V  $\pm$  10 %,  $V_{SS} = 0$  V (unless noted otherwise)  
 ( $C_L = 100$  pF for port 0 and  $\overline{PSEN}$ , 80 pF for all other outputs)

Parameter	Symbol	Limit Values			Units	Test Condition
		min.	typ.	max.		
Power-down current	$I_{PD}$			70	$\mu$ A	$V_{DD} = 5$ V
Pin capacitance	$C_{IO}$			10	pF	$f_C = 1$ MHz
Analog input capacitance	$C_I$			45	pF	
ADC-input current	$I_{ANA}$			$\pm 50$	$\mu$ A	
ADC-differential non-linearity	$DNLE$		0.5	1	LSB	$T_A > 0$ °C
ADC-offset error	$OE$		0.5	1	LSB	$T_A > 0$ °C
ADC-gain error	$GE$		0.5	1	LSB	$T_A > 0$ °C
ADC-differential non-linearity	$DNLE$		1.5	t.b.d.	LSB	$T_A < 0$ °C
ADC-offset error	$OE$		1.5	t.b.d.	LSB	$T_A < 0$ °C
ADC-gain error	$GE$		1.5	t.b.d.	LSB	$T_A < 0$ °C
Analog ground voltage	$V_{AGND}$	$V_{SS}$		$0.5 \times V_{DD}$	V	
Analog reference voltage	$V_{AREF}$	$0.5 \times V_{DD}$		$V_{DD}$	V	
Analog supply difference voltage	$V_{ADELTA}$	$0.5 \times V_{DD}$		$V_{DD}$	V	
Analog input voltage	$V_{AI}$	$V_{AGND} - 0.2$		$V_{AREF} + 0.2$	V	
$V_{AREF}$ supply current	$I_{AREF}$			$\pm 50$	$\mu$ A	$V_{AREF} = 5$ V

## 3.3 AC-Characteristics

## Program Memory and External Clock Drive Characteristics

$T_A = -20$  to  $85$  °C;  $V_{DD} = 5$  V  $\pm$  10 %,  $V_{SS} = 0$  V

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

Parameter	Symbol	Limit Values		Unit
		Variable Clock $1/t_{CLCL} = 1.2$ MHz to 16 MHz		
		min.	max.	
Cycle time	$t_{CY}$	$12 t_{CLCL}$	–	ns
$\overline{PSEN}$ pulse width	$t_{PLPH}$	$3 t_{CLCL} - 35$	–	ns
$\overline{PSEN}$ low to valid instr in	$t_{PLIV}$	–	$1.6 \times t_{CLCL}$	ns
Input instruction hold after $\overline{PSEN}$ high	$t_{PXIX}$	0	–	ns
Address out to valid instr in	$t_{AVIV}$	–	$4.6 \times t_{CLCL}$	ns
Address valid after $\overline{PSEN}$ high	$t_{PXAV}$	0	–	ns
Oscillator period	$t_{CLCL}$	62.5	833.3	ns
External clock high time	$t_{CHCX}$	15		ns
External clock low time	$t_{CLCX}$	15		ns
External clock rise time	$t_{CLOH}$		15	ns
External clock fall time	$t_{CHCL}$		15	ns

AC-Testing Input, Output, Float Waveforms

AC testing inputs are driven at  $V_{DD} - 0.5 V$  for a logic "1" and at  $0.45 V$  for a logic "0". Timing measurements are made at  $V_{IHmin}$  for a logic "1" and at  $V_{IHmax}$  for a logic "0". For timing purposes a port pin is no longer floating, when a  $100 mV$  change from load voltage occurs.

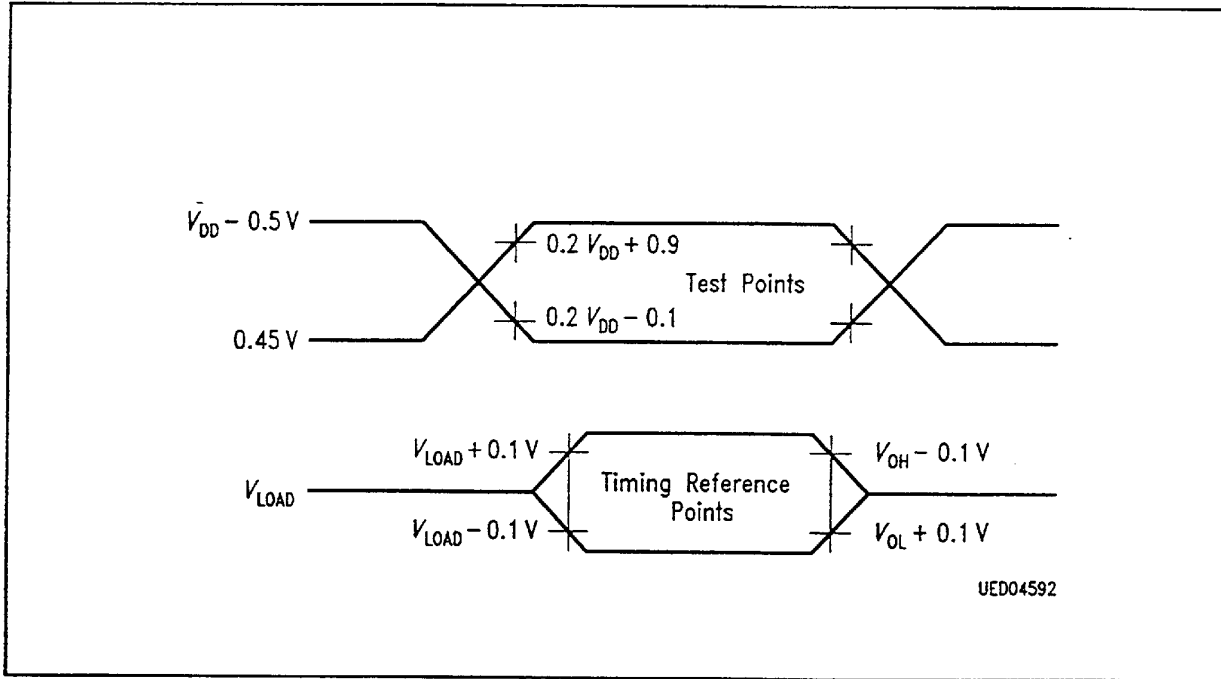


Figure 28  
I/O-Waveform for AC-Tests

Waveforms

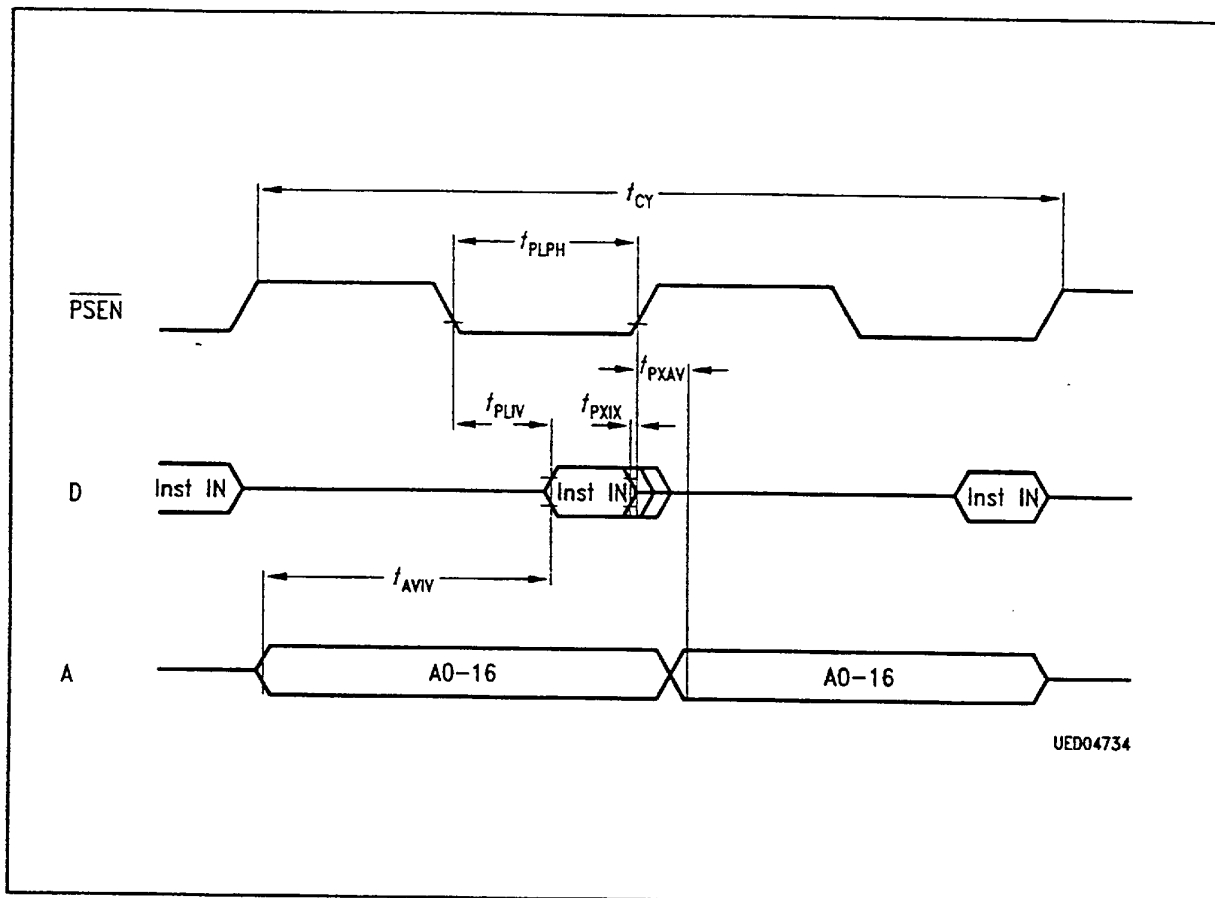


Figure 29  
Program Memory Read Cycle

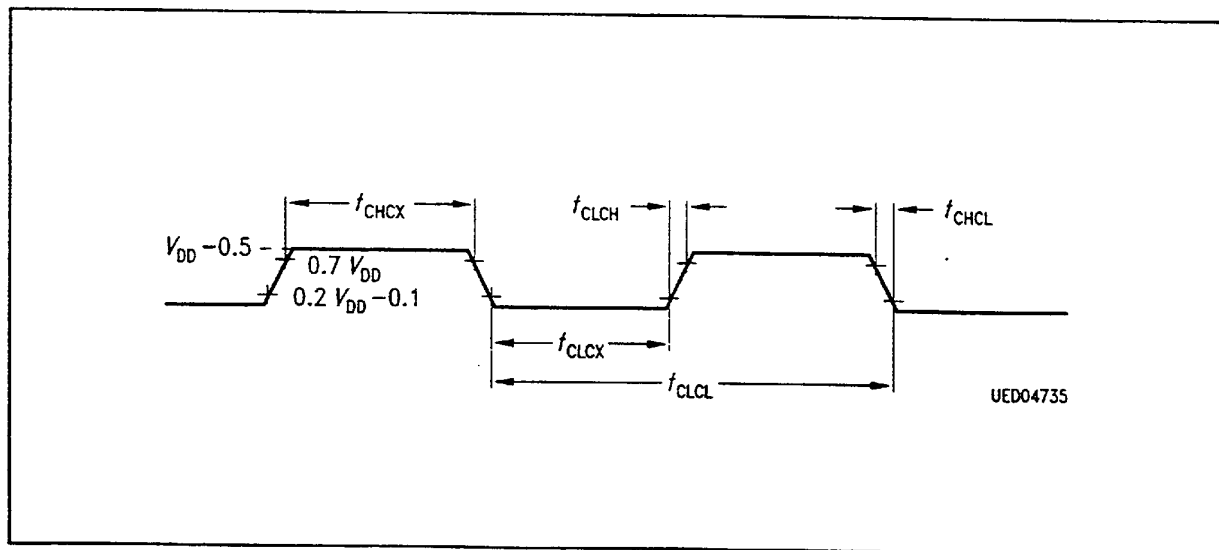
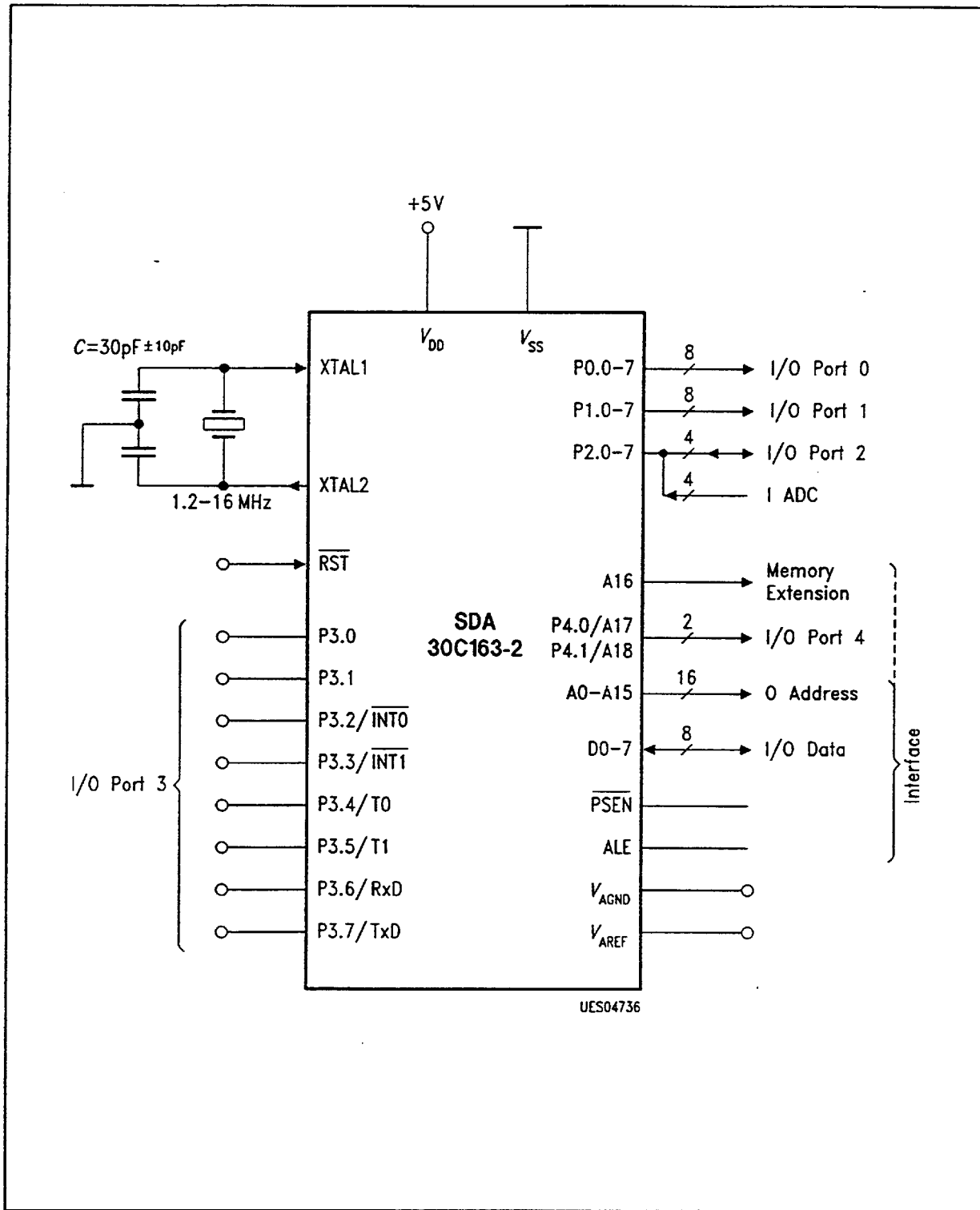


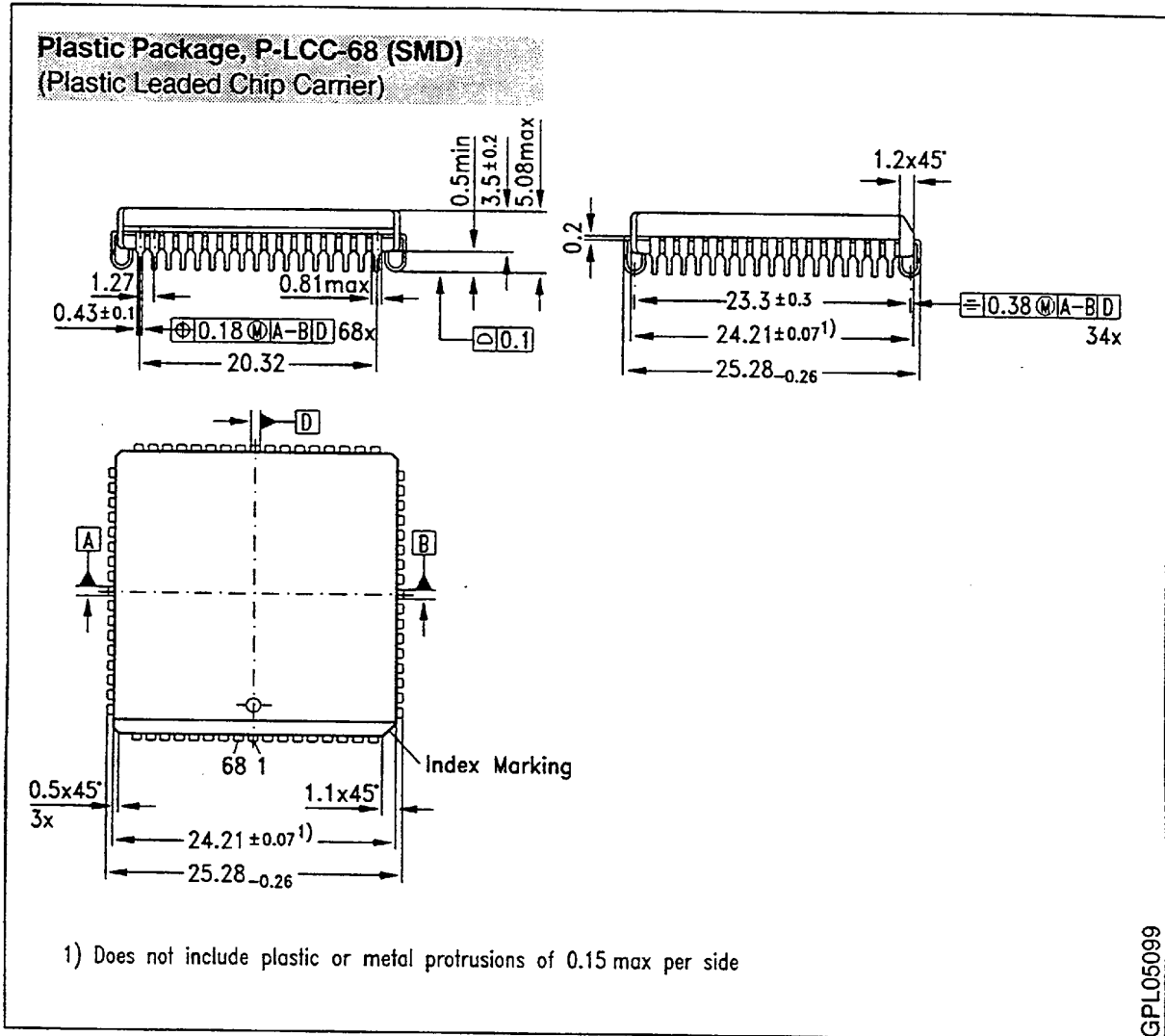
Figure 30  
External Clock Cycle

4 Applications



**Figure 31**  
Application Circuit

5 Package Outlines



**Sorts of Packing**

Package outlines for tubes, trays etc. are contained in our Data Book "Package Information"

SMD = Surface Mounted Device

Dimensions in mm