SIEMENS

ICs for Consumer Electronics

8-Bit Microcontroller, ROMLESS

SDA 30C164

Preliminary Data Sheet 11.94

SDA 30C164 Revision History:	Original Version: 11.94
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 specified mean values expected over the production spread. If not otherwise specified, typical characteristics apply at $T_A = 25\,°C$ 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 11.94
This edition was realized using the software system FrameMaker®

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

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

Contents (cont'd) Page

Introduction

The SDA 30C164, a derivative of the SAB-C501, 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+3)-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 30C164 contains a 2048 + 256-byte data memory (RAM), two independent 16-bit timers/ -counters and a seven-source, four-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 one 2-bit port. Two serial interfaces are included, one behaves like the 8051 UART, the other is a I^2C bus interface.

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 an 8-bit, four channel converter. The input channels are P20 to P23, the analog supply are pins V_{AREF} and V_{AGND}. A flexible overvoltage / undervoltage detector is included.

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 12-MHz crystal, 64 instructions execute in 0.5 µs and 45 instructions execute in 1.0 µs. The remaining instructions (multiple and divide) require only 2 µs. The number of bytes in each instruction and the number of oscillator periods required for execution are listed in the Instruction Set in chapter 2.13.3.

Based on the SDA 30C163-2, the SDA 30C164 comprises double stack size for the extension memory (32 byte) and seven additional data pointer registers.

SIEMENS

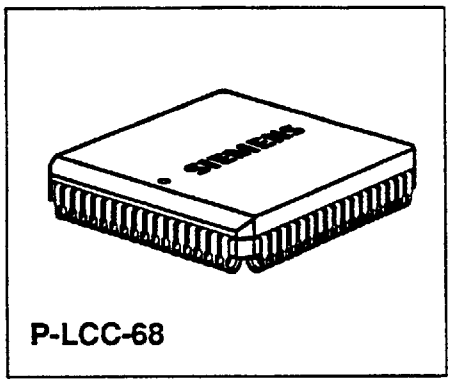
8-Bit Microcontroller ROMLESS

SDA 30C164

Preliminary Data

CMOS IC

1 Features



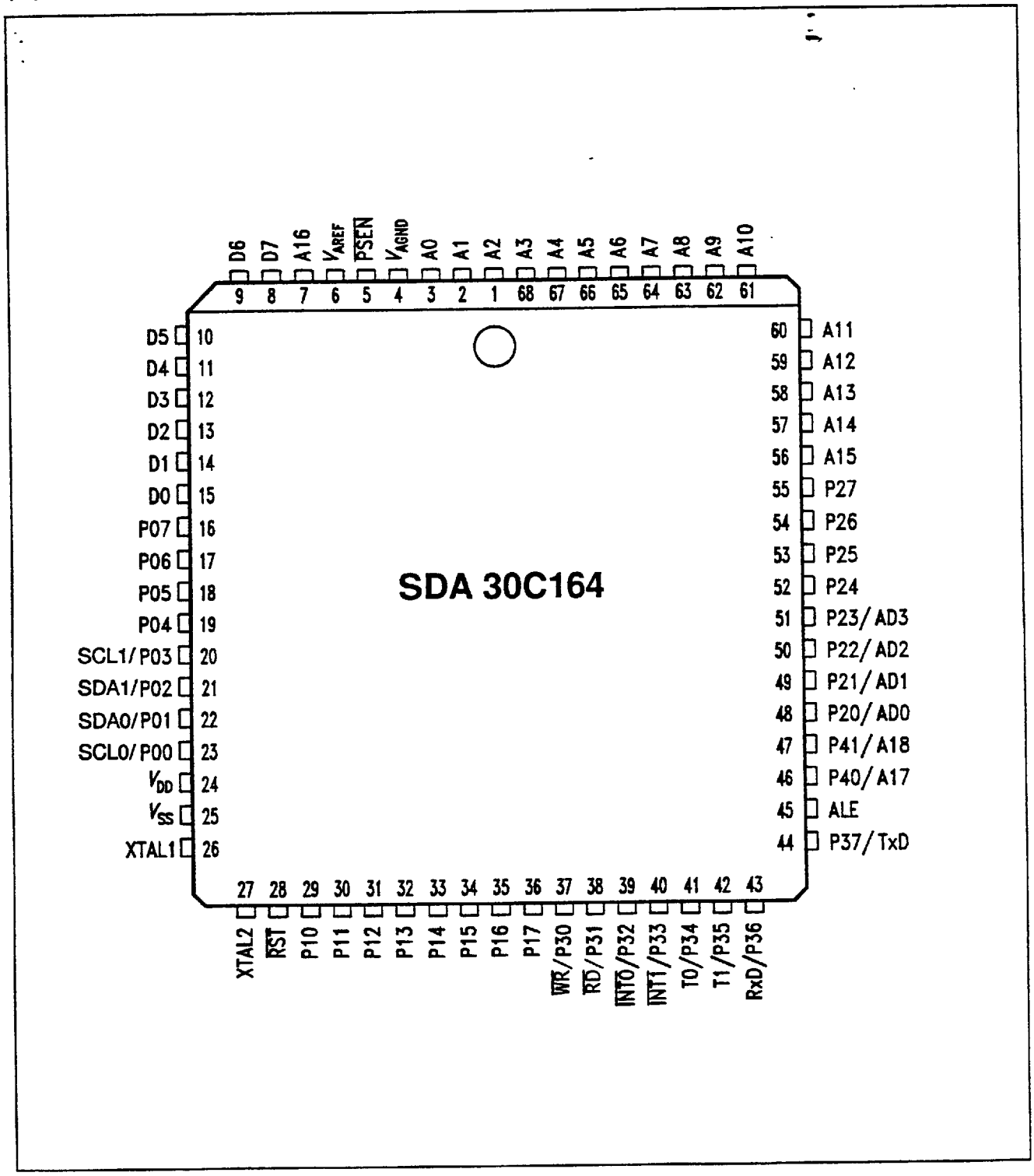
P-LCC-68

- **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
  - Four priority level, nested interrupt structure
  - 8-MHz oscillator frequency, 0.75 µs instruction cycle
  - 8 data pointer registers
- **Serial Interface**
  - Full duplex UART-interface
  - I^2C compatible interface
- **On-Chip RAM**
  - Direct byte and bit addressability
  - Four register banks
  - 256 bytes of data memory, including 128 user-defined software flags
  - 2048 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 with 8-bit resolution
  - Overvoltage/Undervoltage Detector with interrupt capability

Type	Ordering Code	Package
SDA 30C164	Q 67127 - D6346 -	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 22 21 20 19 18 17 16	P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	I/O I/O I/O I/O I/O I/O I/O 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. The secondary functions are assigned to the pins of port 0 as follows: – SCL0 (P0.0) : I^2C Bus Clock 0 – SDA0 (P0.1) : I^2C Bus Data 0 – SDA1 (P0.2) : I^2C Bus Data 1 – SCL1 (P0.3) : I^2C Bus Clock 1
29 30 31 32 33 34 35 36	P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7	I/O I/O I/O I/O I/O I/O I/O 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: PWMi (P1.i): output of PWM-channel i (i = 0, ..., 7).
48 49 50 51 52 53 54 55	P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7	I I I I I/O I/O I/O I/O	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.
37 38 39 40 41 42 43 44	P3.0 P3.1 P3.2 P3.3 P3.4 P3.5 P3.6 P3.7	I/O I/O I/O I/O I/O I/O I/O 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. The secondary functions are assigned to the pins of port 3, as follows: – $\overline{WR}$ (P3.0) : Write for MOVX – $\overline{RD}$ (P3.1) : Read for MOVX – $\overline{INT0}$ (P3.2) : interrupt 0 input/timer 0 gate control input – $\overline{INT1}$ (P3.3) : interrupt 1 input/timer 1 gate control input – T0 (P3.4) : counter 0 input – T1 (P3.5) : counter 1 input – RxD (P3.6) : serial port receive line – TxD (P3.7) : serial port transmit line

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 26	XTAL2 XTAL1	O I	Output of the inverting oscillator amplifier. To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left open. Input to the inverting oscillator amplifier.
28	$\overline{RST}$	I	A low level on this pin resets the processor.
24 25	V_{DD} V_{SS}	S S	Power supply voltage Ground (0 V)
6 4	V_{AREF} V_{AGND}	S S	Analog reference voltage Analog ground
5 45	$\overline{PSEN}$ 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

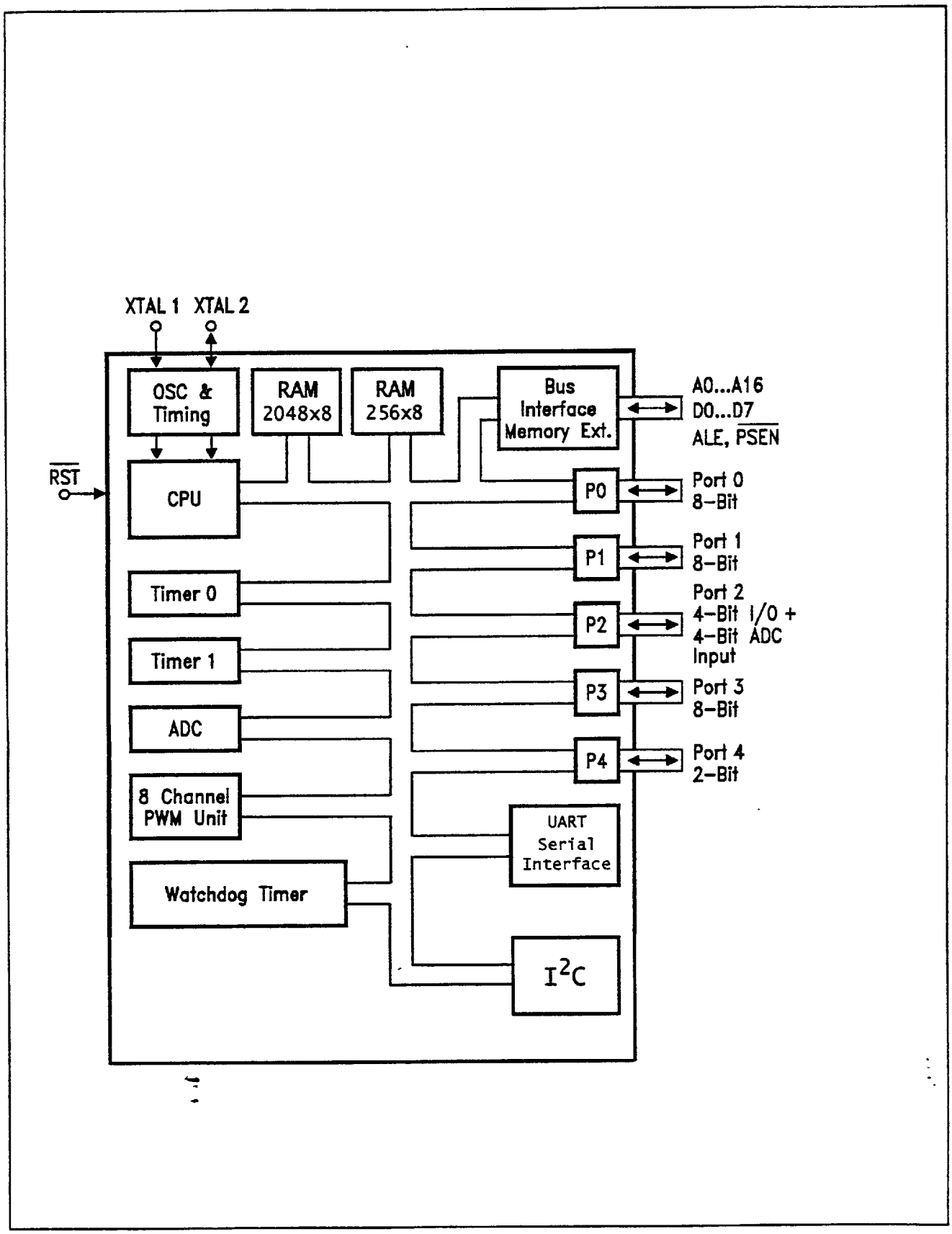


Figure 1
Block Diagram

2 Functional Description

2.1 Architecture

The CPU manipulates operands in three memory spaces. These are the program memory (512 Kbyte) and (256 + 2048) 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, 2048 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 2048 bytes internal RAM. It can be considered as a part of an external data memory. It is located at addresses 63488 to 65535 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	$00_H - 07_H$
0	1	1	$08_H - 0F_H$
1	0	2	$10_H - 17_H$
1	1	3	$18_H - 1F_H$

Program Status Word PSW **SFR-Address $D0_H$**

Default after reset: 00_H

(MSB) (LSB)

CY	AC	F0	RS1	RS0	OV	F1	P

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).

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 MHz to 12 MHz. There is a divide-by-6 internal timing which leads to a minimum instruction cycle of 0.5 µs with a 12-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 MHz to 12-MHz range is also accommodated when an external clock is applied to XTAL1 as the frequency source.

In this specification all timings are referenced to an internal clock. The relationship between the internal clock and the oscillator frequency depends on the setting of CDC in special function register AFR. If CDC is set to 1 the internal clock is half the external oscillator frequency, if CDC is reset to 0 (which is the reset value) the internal clock is equal to the external oscillator frequency.

A machine cycle consists of 6 internal clocks. 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 (00_H – $7F_H$) access internal RAM-locations, while values between 128 and 255 (80_H – $0FF_H$) 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 accessable 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 four address spaces. The memory spaces are:

- 512-Kbyte external program memory address space
- 256 byte plus 128-byte internal data memory address space
- 2048-byte additional internal data memory
- 62 Kbyte additional external 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 the data memory space.

2.2.1 External Program Memory

Certain locations in program memory are reserved for specific programs. Locations 0000_H through 0002_H are reserved for the initialization program. Following reset, the CPU always begins execution at location 0000. Locations 0003_H through 0033_H are reserved for the seven interrupt service routines.

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_H. 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. If enabled by DJMP in SFR AFR, JMP@A+DPTR will perform the same operation.

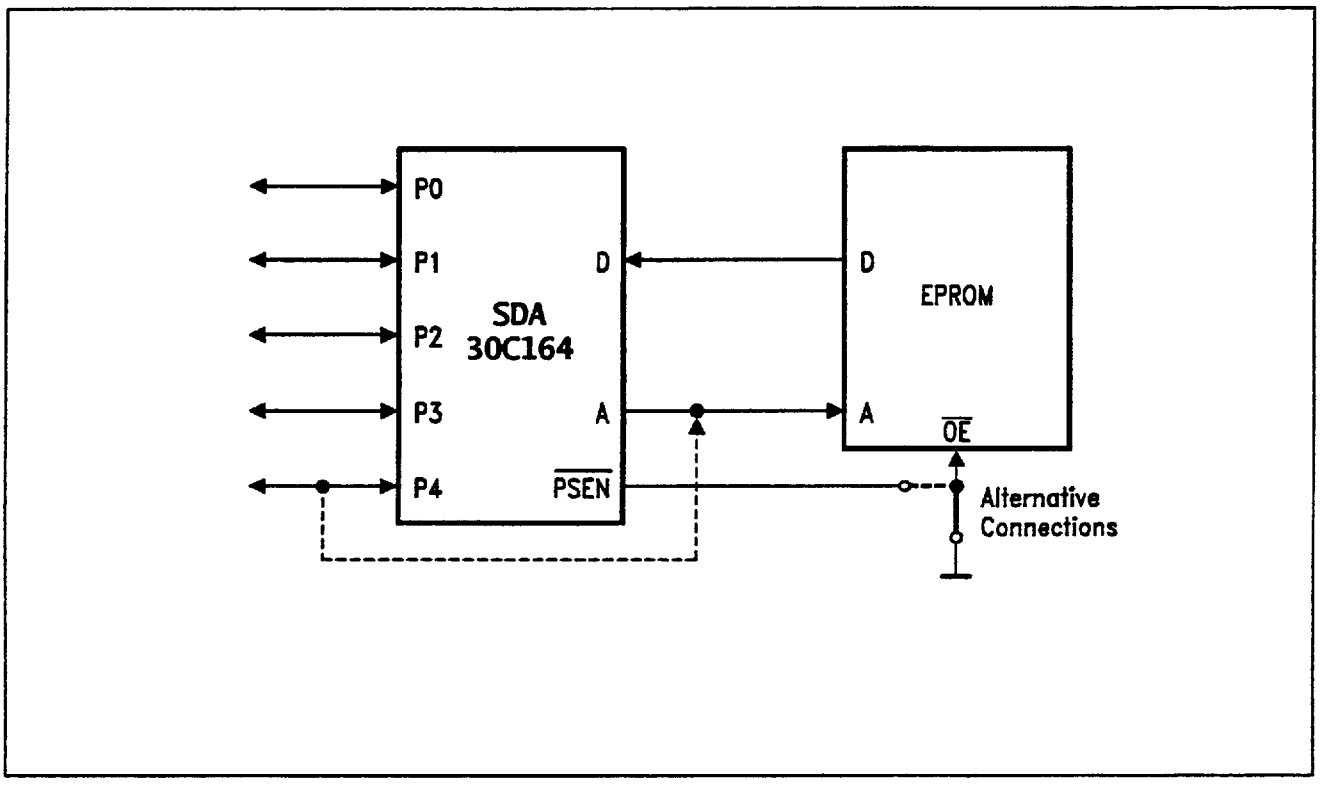


Figure 2
Connecting External Program Memory

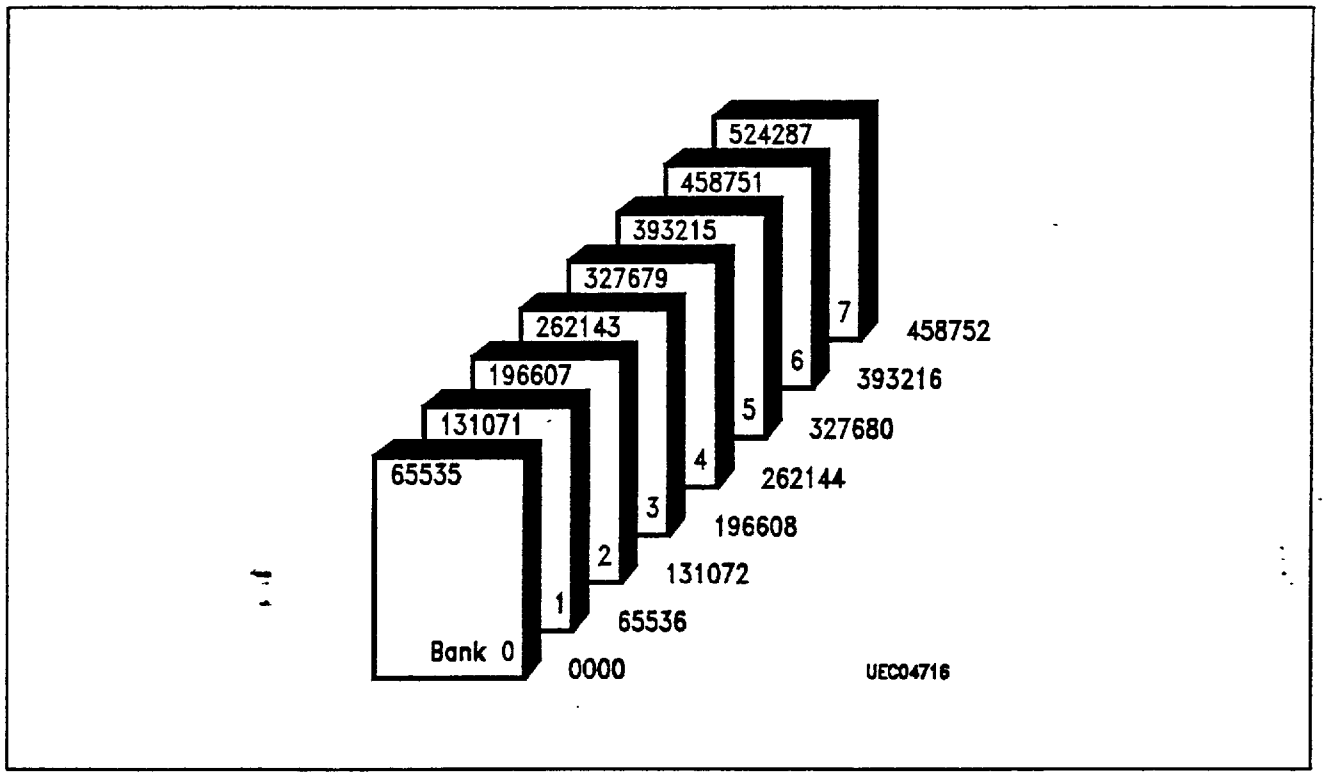


Figure 3
Bank Organization

Figure 4
Register Bits MEX1 and MEX2

Memory Extension Bank MEX1 **SFR-Address 94_H**

Default after reset: X000 $X000_B$

(MSB) (LSB)

–	CB18	CB17	CB16	–	NB18	NB17	NB16

Memory Extension Mode MEX2 **SFR-Address 95_H**

Default after reset: 00_H

(MSB) (LSB)

MM	MB18	MB17	MB16	SF	IB18	IB17	IB16

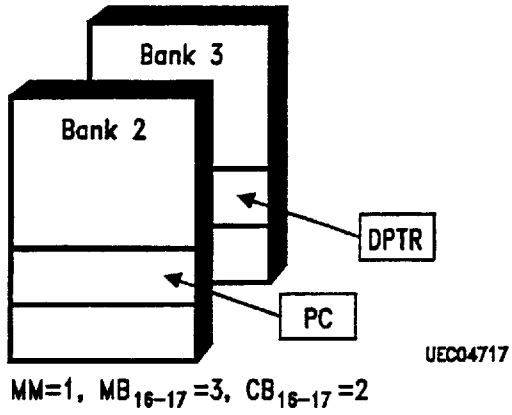
CB = Current Bank Read only; CBx = Ax
NB = Next Bank R/W
MM = Memory Mode R/W; 1 = use MB
MB = Memory Bank R/W
SF = Stack Full Read only; 1 = full
IB = Interrupt Bank R/W

Table 1
Port 4 Configuration

CB	P4 Latch	P4 Out	Comment
0	0	0	
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.



FigFigure 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.

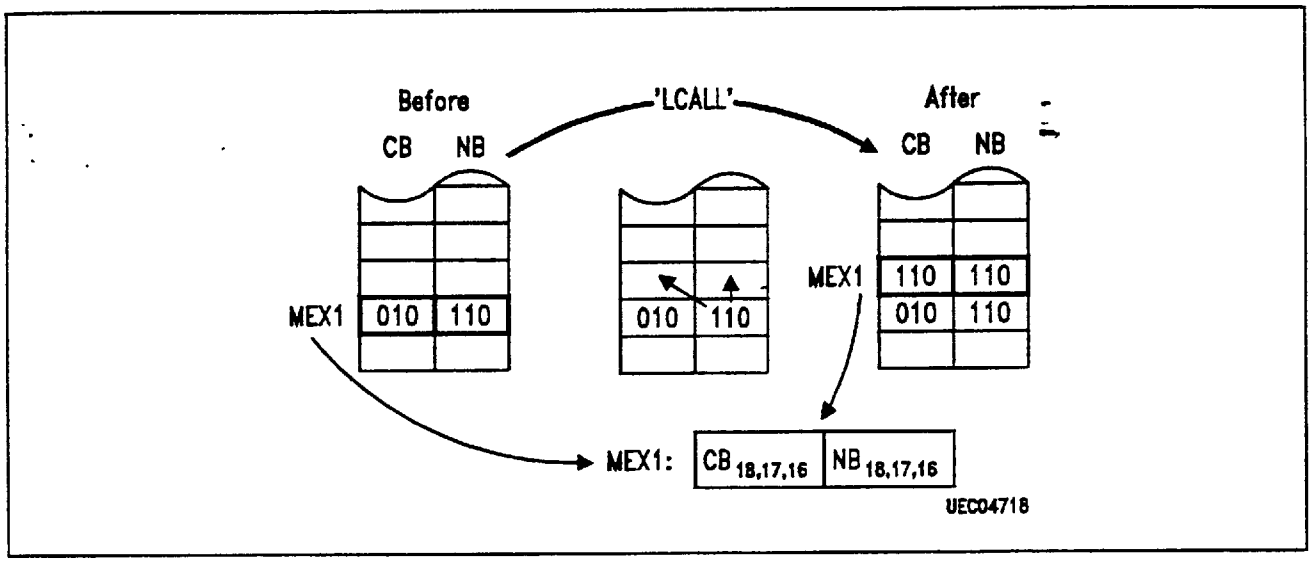


Figure 6
Processing LCALL (same as ACALL)

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.

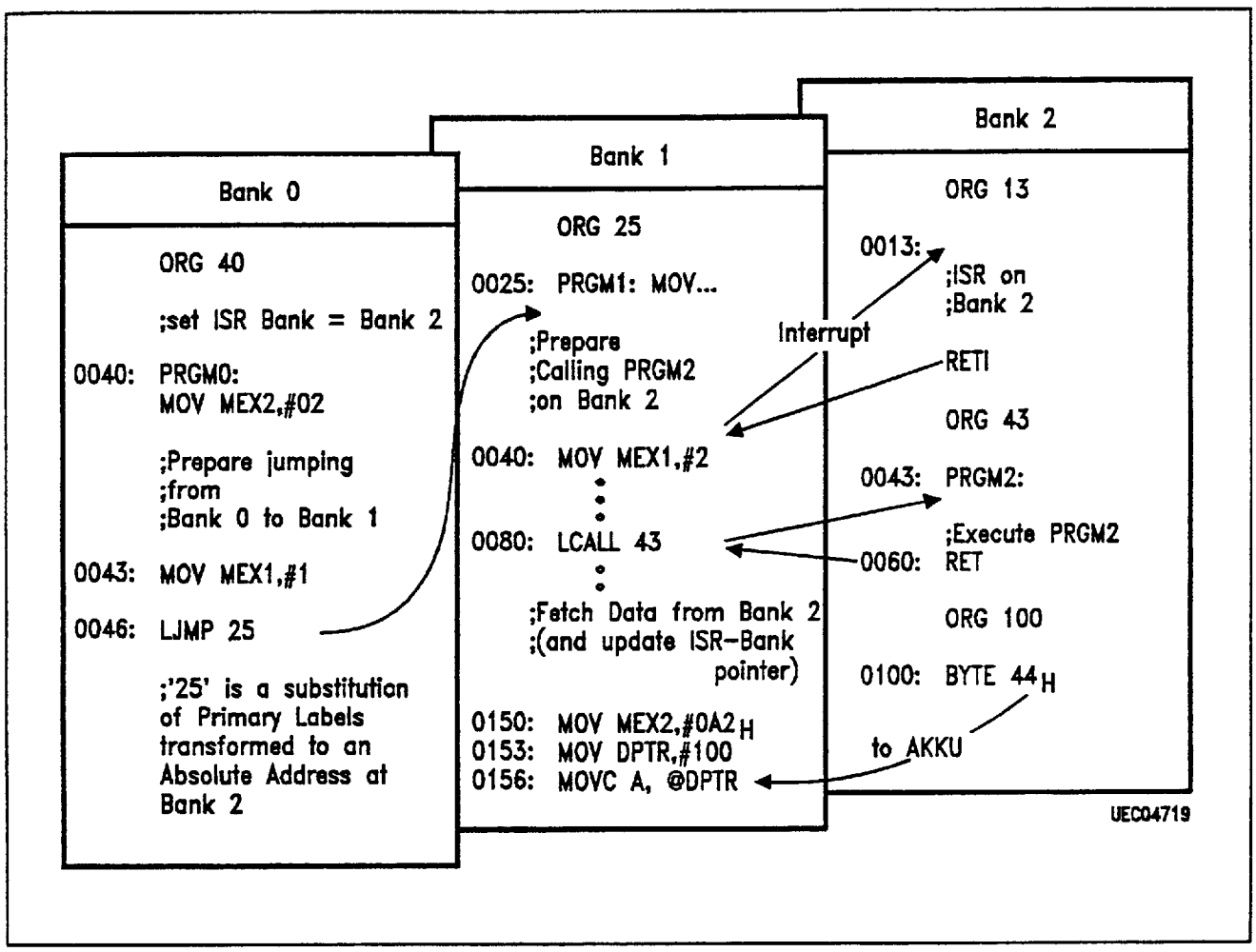


Figure 7
Program Example

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 2048-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 2304 bytes. The 2048 bytes of XRAM are accessed by MOVX @DPTR. XRAM is located in the upper area of the address space at $0F800_H$ – $0FFFF_H$.

Page Registers **PAGE 0,1** **SFR-Address $A4_H$,$A5_H$**

Default after reset: XX_H

(MSB) (LSB)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7...Bit 0 : Upper 8 bits of memory address used in MOVX@Ri instructions. PAGE0 is used together with R0, PAGE1 is used together with R1.

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**.

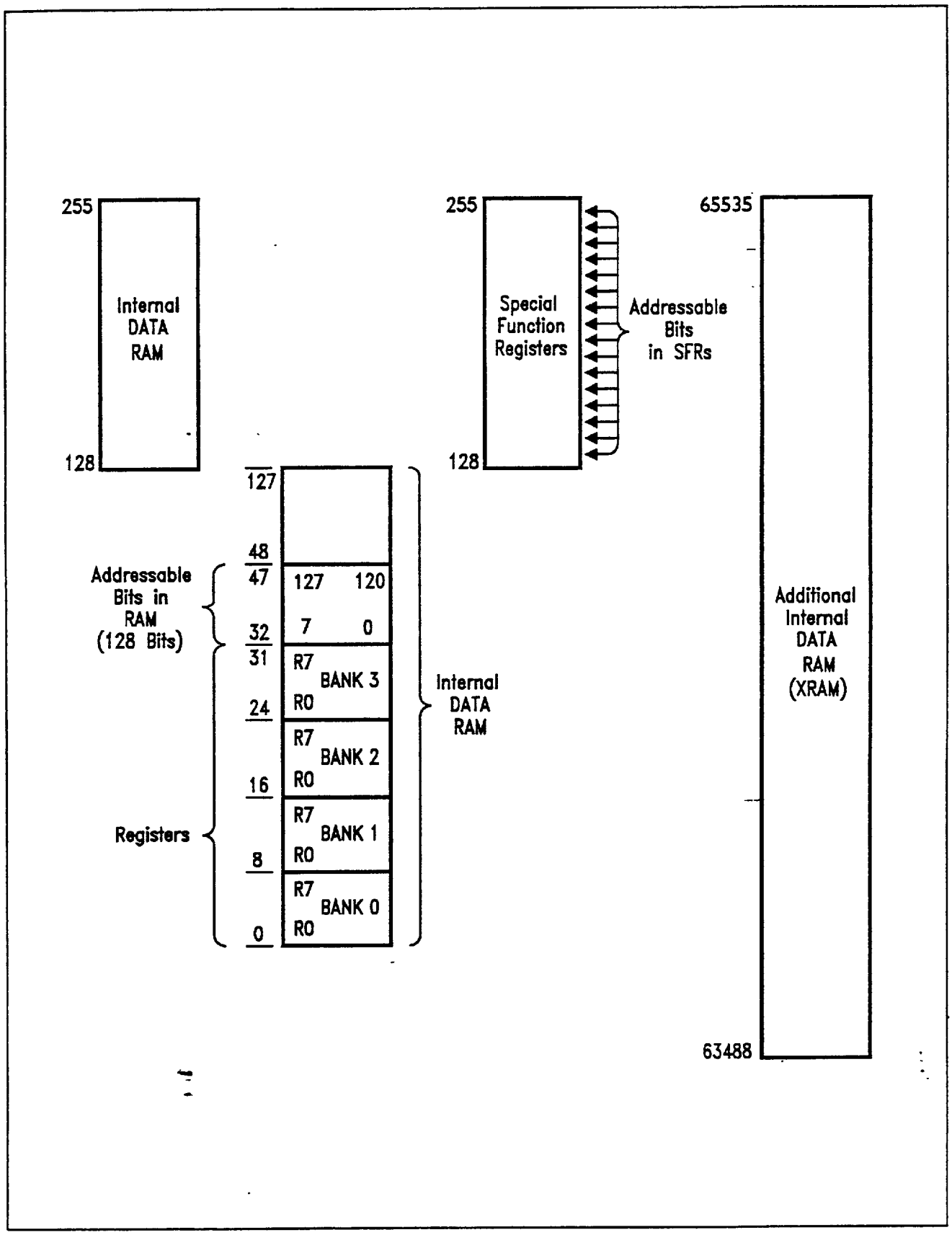


Figure 8
Internal Data Memory Address Space

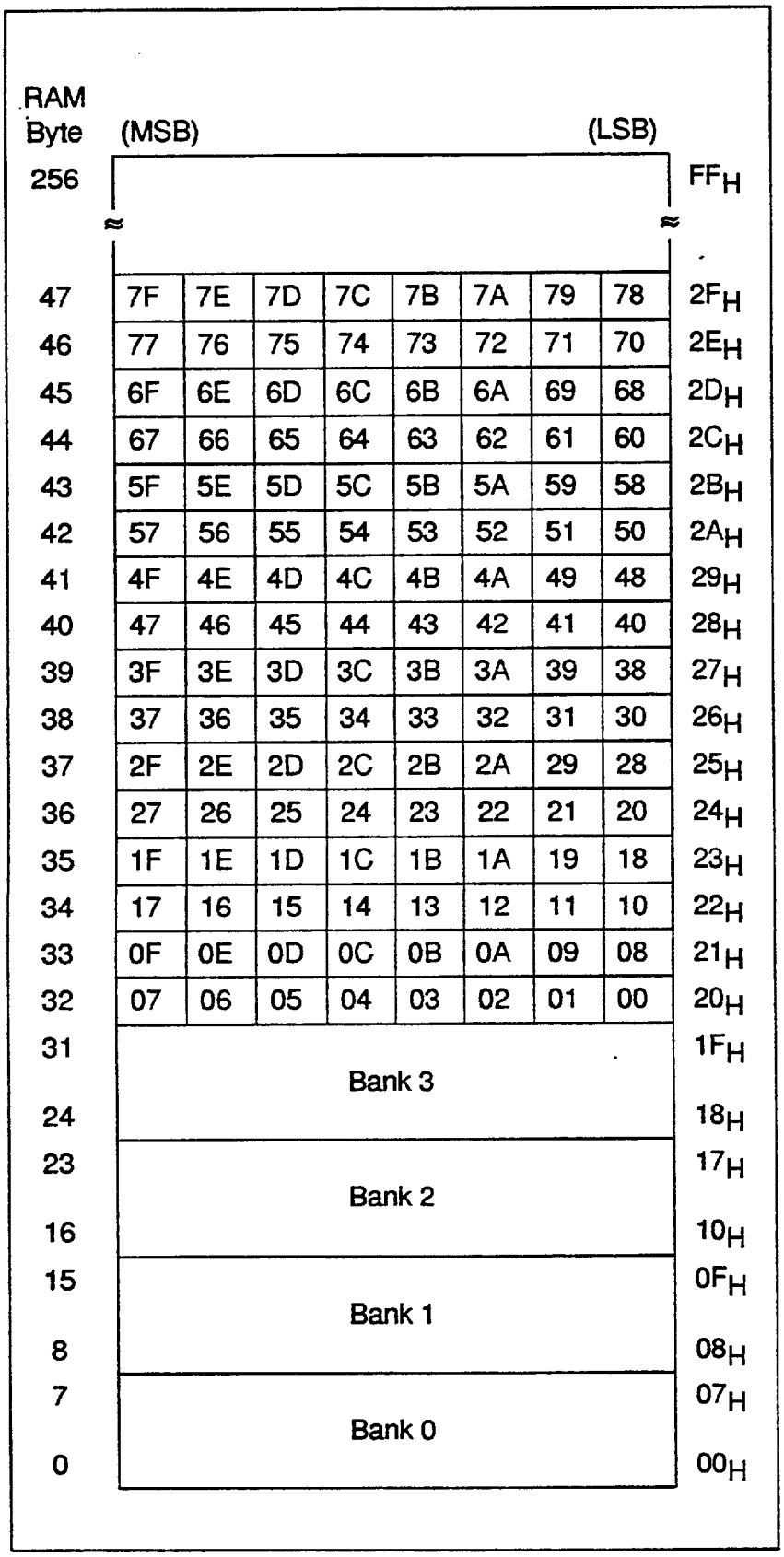


Figure 9
Internal RAM-Bit Addresses

Direct Byte Address	Bit Address								Hardware Register Symbol
$F8_H$	FF	FE	FD	FC	FB	FA	F9	F8	PWME
$F0_H$	F7	F6	F5	F4	F3	F2	F1	F0	B
$E8_H$	–	–	–	–	–	–	E9	E8	P4
$E0_H$	E7	E6	E5	E4	E3	E2	E1	E0	ACC
$D8_H$	DF	DE	–	DC	DB	–	D9	D8	ADCON
$D0_H$	D7	D6	D5	D4	D3	D2	D1	D0	PSW
$C8_H$	–	–	–	–	–	–	–	–	–
$C0_H$	–	–	–	–	–	–	–	–	–
$B8_H$	BF	BE	BD	BC	BB	BA	B9	B8	ICCON
$B0_H$	B7	B6	B5	B4	B3	B2	B1	B0	P3
$A8_H$	AF	AE	AD	AC	AB	AA	A9	A8	IE
$A0_H$	A7	A6	A5	A4	A3	A2	A1	A0	P2
98_H	9F	9E	9D	9C	9B	9A	99	98	SCON
90_H	97	96	95	94	93	92	91	90	P1
88_H	8F	8E	8D	8C	8B	8A	89	88	TCON
80_H	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.)
Arithmetic Registers					
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
System Control Registers					
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	–	00
Memory Extension Bank	MEX1	94	148	–	x000 x000
Memory Extension Mode	MEX2	95	149	–	00
Page Register 0	PAGE 0	A4	164	–	xx
Page Register 1	PAGE 1	A5	165	–	xx
Advanced Function Register	AFR	A6	166	–	0000 0xx1
I/O-Port Registers					
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	see note
Interrupt Control Registers					
Interrupt Enable Flags	IE	A8	168	AF – A8	00
Interrupt Priority Flags 0	IP0	A9	169	–	00
Interrupt Priority Flags 1	IP1	AA	170	–	00
Interrupt Request Control	IRCON	AB	171	–	x5
Timer 0/1 Registers					
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
Watchdog Timer Registers					
Watchdog Control Register	WDCON	A7	167	–	0xxx xxxx
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.)
Analog Digital Converter					
ADC-Control Register	ADCON	D8	216	9F – 98	00
ADC-Data Register	ADDAT	D9	217	–	xx
ADC-Start Register	DAPR	DA	218	–	xx
Detector Control Register	DECON	DC	220	–	0000 0xxx
Detector Select Register	DESEL	DD	221	–	xx
Detector Lower Register	DELOW	DE	222	–	xx
Detector Upper Register	DEUP	DF	223	–	xx
Pulse Width Modulator Registers					
Enable Register	PWME	F8	248	FF – F8	00
Counter Register (low byte)	PWCL	F7	247	–	00
Counter Register (high byte)	PWCH	F9	249	–	c0
Compare Register 0	PWCOMP0	F1	241	–	00
Compare Register 1	PWCOMP1	F2	242	–	00
Compare Register 2	PWCOMP2	F3	243	–	00
Compare Register 3	PWCOMP3	F4	244	–	00
Compare Register 4	PWCOMP4	F5	245	–	00
Compare Register 5	PWCOMP5	F6	246	–	00
PWM 14 Compare Reg. 0	PWCOMP6	FB	251	–	00
PWM 14 Extension Reg. 0	PWEXT6	FA	250	–	02
PWM 14 Compare Reg. 1	PWCOMP7	FD	253	–	00
PWM 14 Extension Reg. 1	PWEXT7	FC	252	–	02
Serial Interface Registers					
Serial Control Register	SCON	98	144	9F – 98	00
Serial Data Register	SBUF	99	145	–	xx
I^2C Registers					
I^2C Control Register	ICCON	B8	184	BF – B8	0000 0x00
I^2C Mode Register	ICMOD	B9	185	–	000x 00xx
I^2C Shifter Register	ICSHI	BA	186	–	xx
I^2C Address Register	ICADR	BB	187	–	xx
I^2C Baud Register 0	ICBD0	BC	188	–	00xx xxxx
I^2C Baud Register 1	ICBD1	BD	189	–	00xx xxxx

Note: The P4 internal latches are set to 1, the P4 external pins are reset to 0.

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, four-priority-level, nested interrupt system is provided. Interrupt response delay ranges from 1.5 µs to 3.5 µs when using a 12-MHz internal clock.

2.3.1 Interrupt Sources

The processor acknowledges interrupt requests from seven sources: two from external sources via the $\overline{INT0}$ and $\overline{INT1}$ pins, one from each of the two internal counters, one from the serial I/O-port, one from the I^2C-interface and one from the analog digital converter. Each of the seven sources can be assigned to either of four 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	(03_H)
Internal Timer/Counter 0	11	($0B_H$)
External Request 1	19	(13_H)
Internal Timer/Counter 1	27	($1B_H$)
Serial Interface	35	(23_H)
I^2C Interface	43	($2B_H$)
Analog Digital Converter	51	(33_H)

2.3.2 Interrupt Control

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

TCON	Timer/Counter Control Register	88_H
IE	Interrupt Enable Register	$A8_H$
IP0	Interrupt Priority Register 0	$A9_H$
IP1	Interrupt Priority Register 1	AA_H
SCON	Serial Control Register	98_H
ICCON	I^2C Control Register	$B8_H$
DECON	ADC Control Register	DC_H
IRCON	Interrupt Request Control Register	AB_H

The interrupt system is shown diagrammatically in **figure 11**.

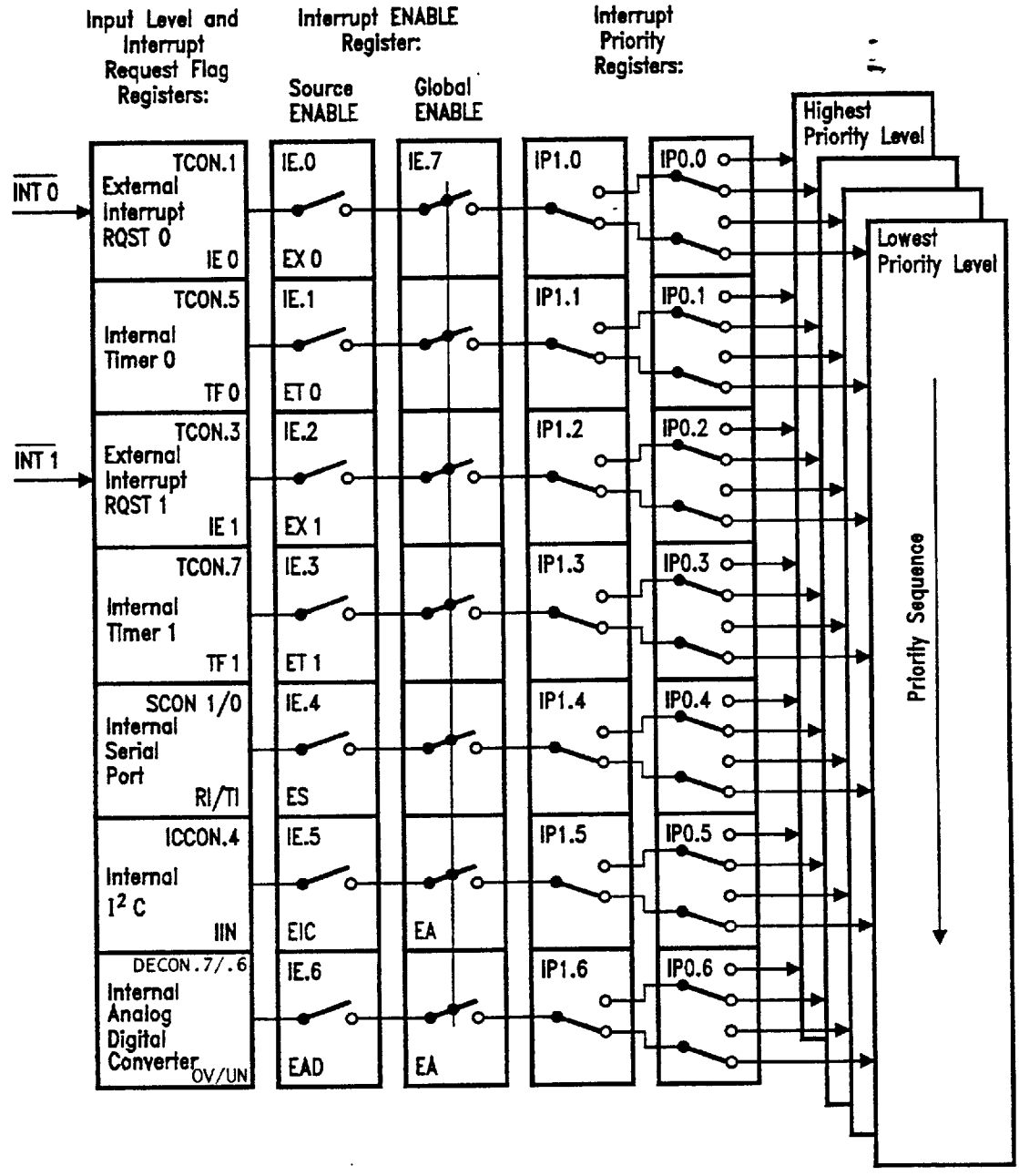
A source requests an interrupt by setting its associated interrupt request flag in the TCON-, SCON-, ICCON- and DECON-register, as detailed in the following table:

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	SCON.0/.1
I^2C Interface	IIN	ICCON.4
Analog Digital Converter	OV/UN	DECON.7/.6

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 eight addressable flags. Seven flags enable/disable the seven interrupt sources when set/cleared. Setting/clearing the eighth 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. This means, interrupts can be generated or pending interrupt requests can be cancelled by software.



- SEVEN INTERRUPT SOURCES
- EACH INTERRUPT CAN BE INDIVIDUALLY ENABLED/DISABLED
- ENABLED INTERRUPTS CAN BE GLOBALLY ENABLED/DISABLED
- EACH INTERRUPT CAN BE ASSIGNED TO EITHER OF FOUR PRIORITY LEVELS
- EACH INTERRUPT VECTORS TO A SEPARATE LOCATION IN PROGRAM MEMORY
- INTERRUPT NESTING TO FOUR 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 $A8_H$**

Default after reset: 00_H

(MSB) (LSB)

EA	EAD	EIC	ES	ET1	EX1	ET0	EX0

EA Enables or disables all interrupts. If EA = 1, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.

EAD Enables or disables the analog digital converter interrupt. If EAD=1, this interrupt will be enabled.

EIC Enables or disables the I^2C interface interrupt. If EIC=1, this interrupt will be enabled.

ES Enables or disables the serial interface interrupt. If ES = 1, this interrupt will be enabled.

ET1 Enables or disables the timer 1 overflow interrupt. If ET1 = 1, the timer 1 interrupt is enabled.

EX1 Enables or disables external interrupt 1. If EX1 = 1, external interrupt 1 is enabled.

ET0 Enables or disables the timer 0 overflow interrupt. If ET0 = 1, the timer 0 interrupt is enabled.

EX0 Enables or disables external interrupt 0. If EX0 = 1, external interrupt 0 is enabled.

Figure 13
Interrupt Priority Register IP0 and IP1

Interrupt Priority Register IP0 **SFR-Address $A9_H$**

Default after reset: 00_H

(MSB)							(LSB)
–	IP0.6	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0

Interrupt Priority Register IP1 **SFR-Address AA_H**

Default after reset: 00_H

(MSB)							(LSB)
–	IP1.6	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0

Corresponding bit-locations in both registers are used to set the interrupt priority level of an interrupt.

IP1.X	IP0.X	Function
0	0	Set priority level 0 (lowest)
0	1	Set priority level 1
1	0	Set priority level 2
1	1	Set priority level 3 (highest)

Bit	Corresponding Interrupt
IP1.0 / IP0.0	IE0
IP1.1 / IP0.1	TF0
IP1.2 / IP0.2	IE1
IP1.3 / IP0.3	TF1
IP1.4 / IP0.4	RI / TI
IP1.5 / IP0.5	IIN
IP1.6 / IP0.6	OV/UN

Setting/clearing a bit in the IP-register establishes its associated interrupt request priority level. If a low-priority level interrupt is being serviced, a higher-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	
4.	TF1	
5.	RI/TI	
6.	IIN	
7.	OV/UN	(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 ($\overline{\text{INT0}}$ and $\overline{\text{INT1}}$) 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 and the four low-order bits of IRCON as shown in **figure 15**.

When IT0 and IT1 are set to one, interrupt requests on $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ are transition-activated, 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 $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$, 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 88_H**

Default after reset: 00_H

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

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 edge/low level triggered external interrupts. IT1 = 1 selects transition-activated 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 edge/low level triggered external interrupts. IT0 = 1 selects transition-activated external interrupts.

Figure 15
Interrupt Request Control Register IRCON **SFR-Address AB_H**

Default after reset: $X5_H$

(MSB) (LSB)

–	–	–	–	ERI1	EFA1	ERI0	EFA0

ERI1 Enable rising edge Interrupt 1.
ERI1=1 enables interrupt generation if a rising edge is detected at $\overline{INT1}$.

EFA1 Enable falling edge Interrupt 1.
EFA1=1 enables interrupt generation if a falling edge is detected at $\overline{INT1}$.

ERI0 Enable rising edge Interrupt 0.
ERI0=1 enables interrupt generation if a rising edge is detected at $\overline{INT0}$.

EFA0 Enable falling edge Interrupt 0.
EFA0=1 enables interrupt generation if a falling edge is detected at $\overline{INT0}$.

Note: The reset value maintains compatibility to Intel 8051.

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

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

For transition-activated operation, the input must remain stable for more than six oscillator periods, but needs not to be synchronous with the oscillator. In this mode, the register IRCON adds more flexibility. Two bits are available for each external interrupt to control interrupt generation on the rising and falling edge.

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

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

For level-activated operation, if the input is low during the sampling that occurs seven 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 seven 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.

If an interrupt is level-activated, its corresponding bits in IRCON should be set to 0 for future compatibility.

2.3.5 Interrupt Task Function

The processor records the active priority level(s) by setting internal flip-flop(s). Each interrupt level has its own flip-flop. 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 5th, 11th, 17th and 23th 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 four 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.

The 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, IIN and OV/UN request flag must be cleared as part of the respective interrupt service program.

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 seven 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 IP0 or IP1, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to complete the next instruction if the instruction is MUL or DIV). Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 8 cycles (approximately 5.25 µs at 8-MHz operation). Examples of the best and worst case conditions are illustrated in the following table.

Instruction	Time (Internal Clocks)	
	Best Case	Worst Case
External interrupt generated immediately before (best) / after (worst) the pin is sampled (time until end of bus cycle).	1 + ε	1 − ε
Current or next instruction finishes in 6-oscillator periods	6	6
Next instruction is MUL or DIV	don't care	24
Internal latency for hardware subroutine call	12	12
	19	43

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 $\overline{RST}$ 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, $\overline{RST}$ should be held low for at least 10 ms after the power supply stabilizes to allow the oscillator to stabilize. Crystal operation below 3 MHz will increase the time necessary to hold $\overline{RST}$ low. 12 internal clocks after receiving of $\overline{RST}$, 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 internal clocks to execute before normal operation commences with the instruction at absolute location 0000_H. Program memory locations 0000_H through 0002_H 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.

If $\overline{RST}$ is left open, the pin is held low by an internal pull-down resistor.

An automatic reset can be obtained when V_{DD} is turned on by connecting the $\overline{RST}$-pin to V_{DD} through a 3V Zener diode, providing the V_{DD} rise time does not exceed a millisecond and the oscillator start-up time does not exceed 10 milliseconds. V_{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 $\overline{RST}$, ALE should not be pulled down externally.

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

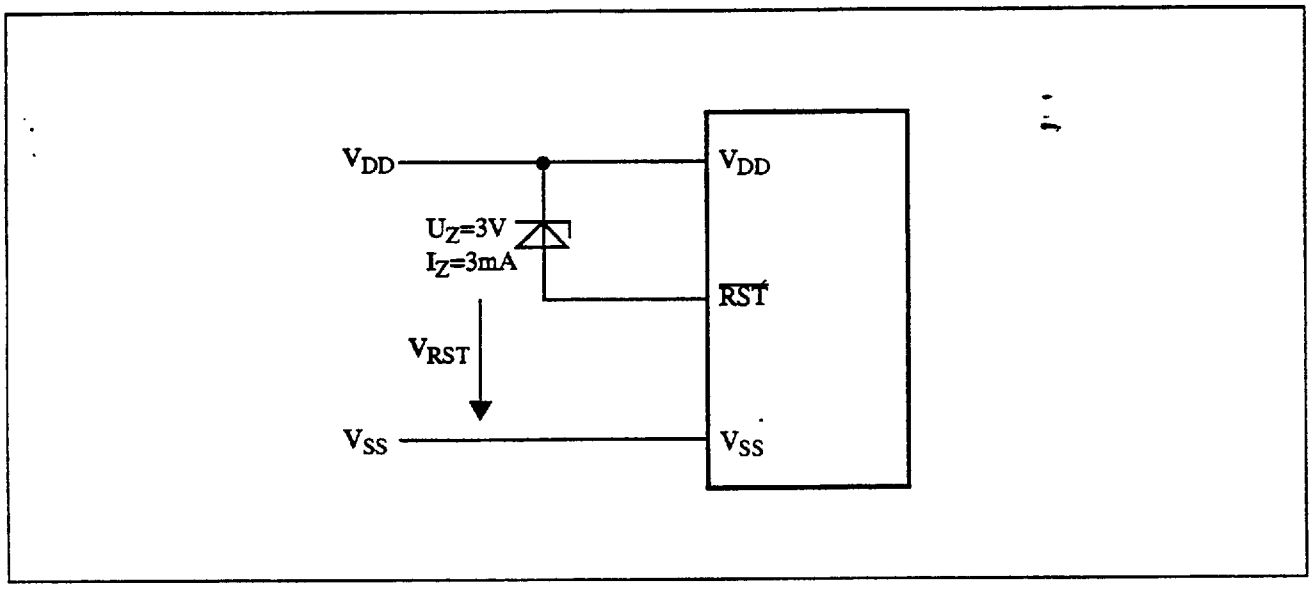


Figure 15
Power-On Reset Circuit

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_H**

Default after reset: 00_H

(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 one internal clock 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.

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

P3.0	$\overline{WR}$	(Write for MOVX)
P3.1	$\overline{RD}$	(Read for MOVX)
P3.2	$\overline{INT0}$	(external interrupt 0)
P3.3	$\overline{INT1}$	(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
SETB PX.Y*	set bit Y of Port X	SETB 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 6 internal clocks, the count rate is 1/6 of the internal clock.

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 to recognize a 1-to-0 transition, the maximum count rate is 1/12 of the internal clock. 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 1 s to all 0 s, 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.

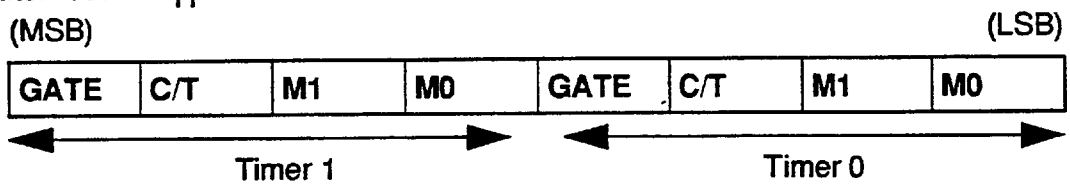
- Mode 3

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.

Figure 16
Timer/Counter Mode Register

Timer 0/1 Mode Register TMOD **SFR-Address 89_H**

Default after reset: 00_H



GATE Gating control when set. Timer/counter "x" is enabled only while "$\overline{INTx}$" 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) TL0 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.

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 internal clock is divided by six 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 ($\overline{INT0}$), ($\overline{INT1}$) 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 $\overline{INT0}$ and $\overline{INT1}$ is not affected by the counter's operation. If enabled, an interrupt will occur when the input at $\overline{INT0}$ or $\overline{INT1}$ 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 5th, 11th, 17th and 23th internal clock of the-instruction-in-progress. Thus, an external reference's high and low times must each be a minimum of 6 internal clock periods in duration. There is a 6 internal clock delay from the time when a toggled input (transition from high to low) is sampled to the time when the counter is incremented.

Figure 17
Timer/Counter Control Register

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

Default after reset: 00_H

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

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.

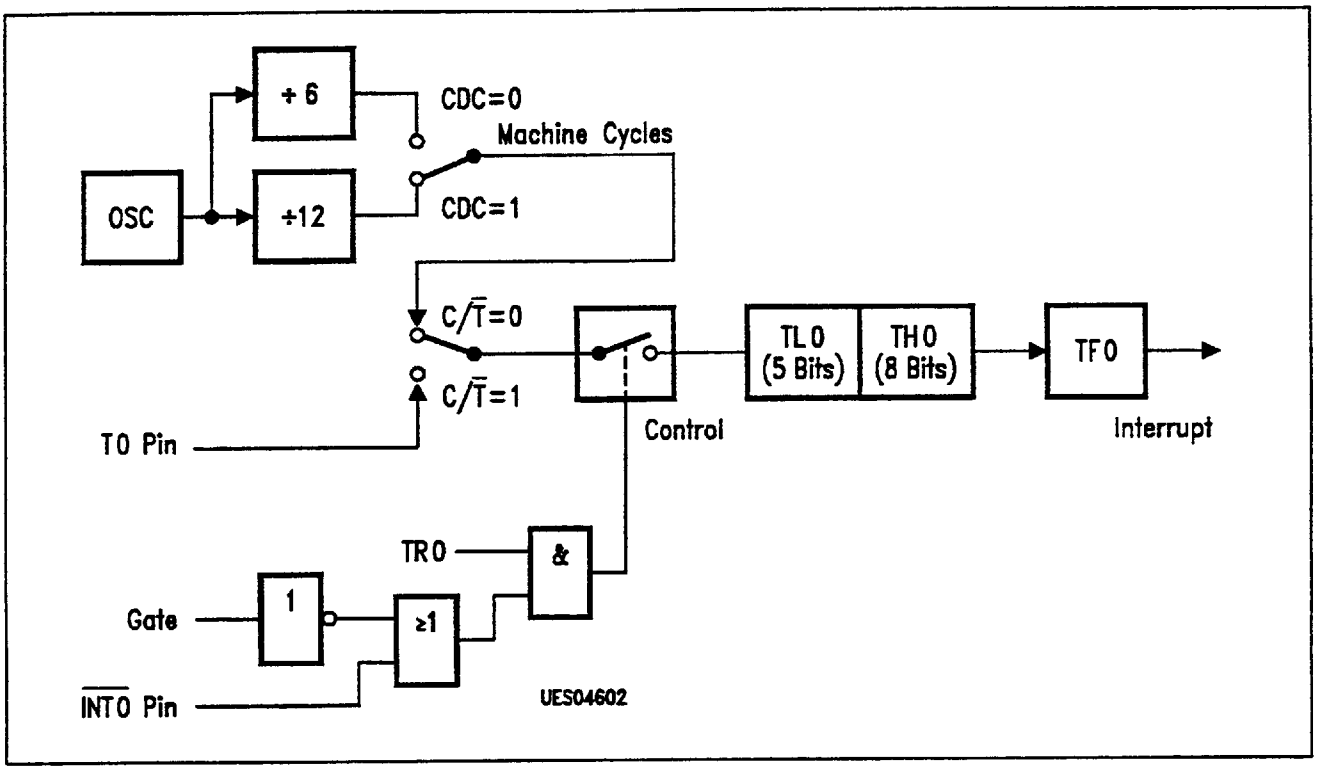


Figure 18
Timer/Counter 0 Mode 0: 13-Bit Counter

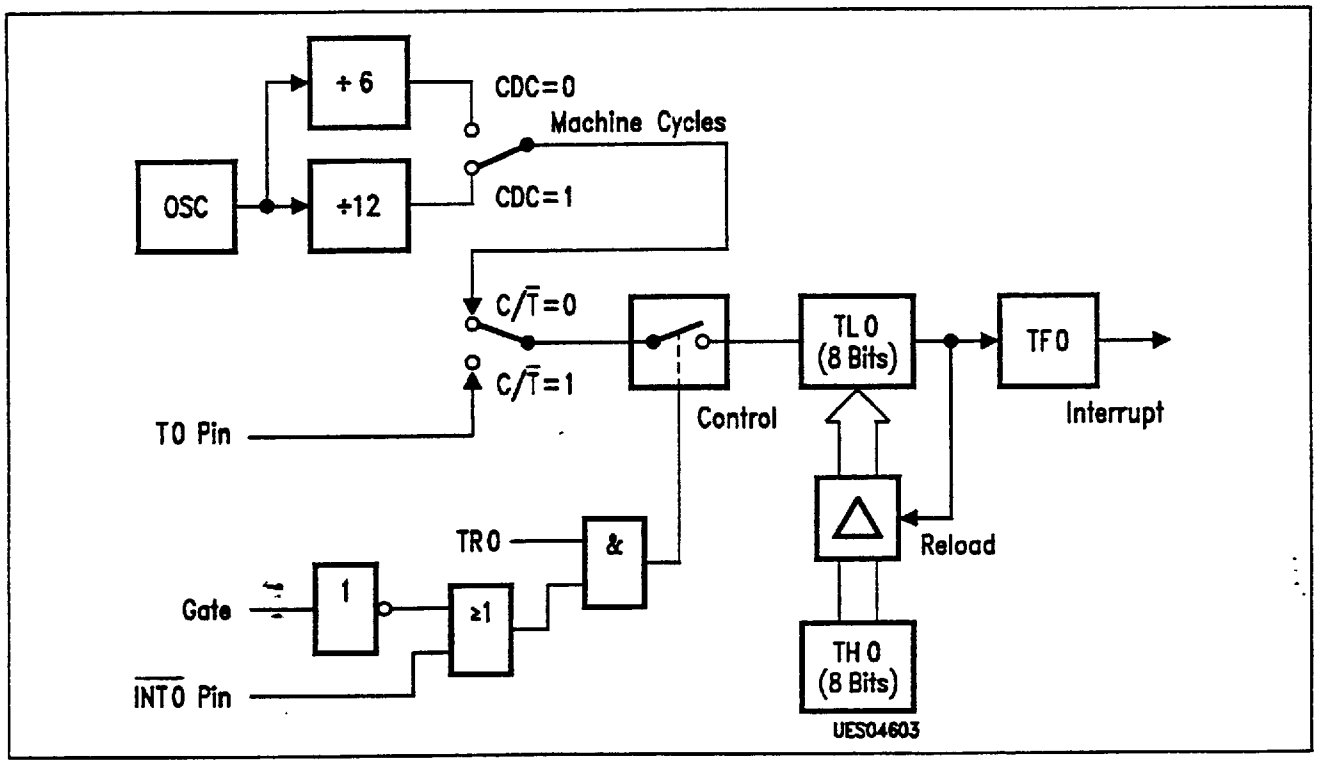


Figure 19
Timer/Counter 0 Mode 2: 8-Bit Auto-Reload

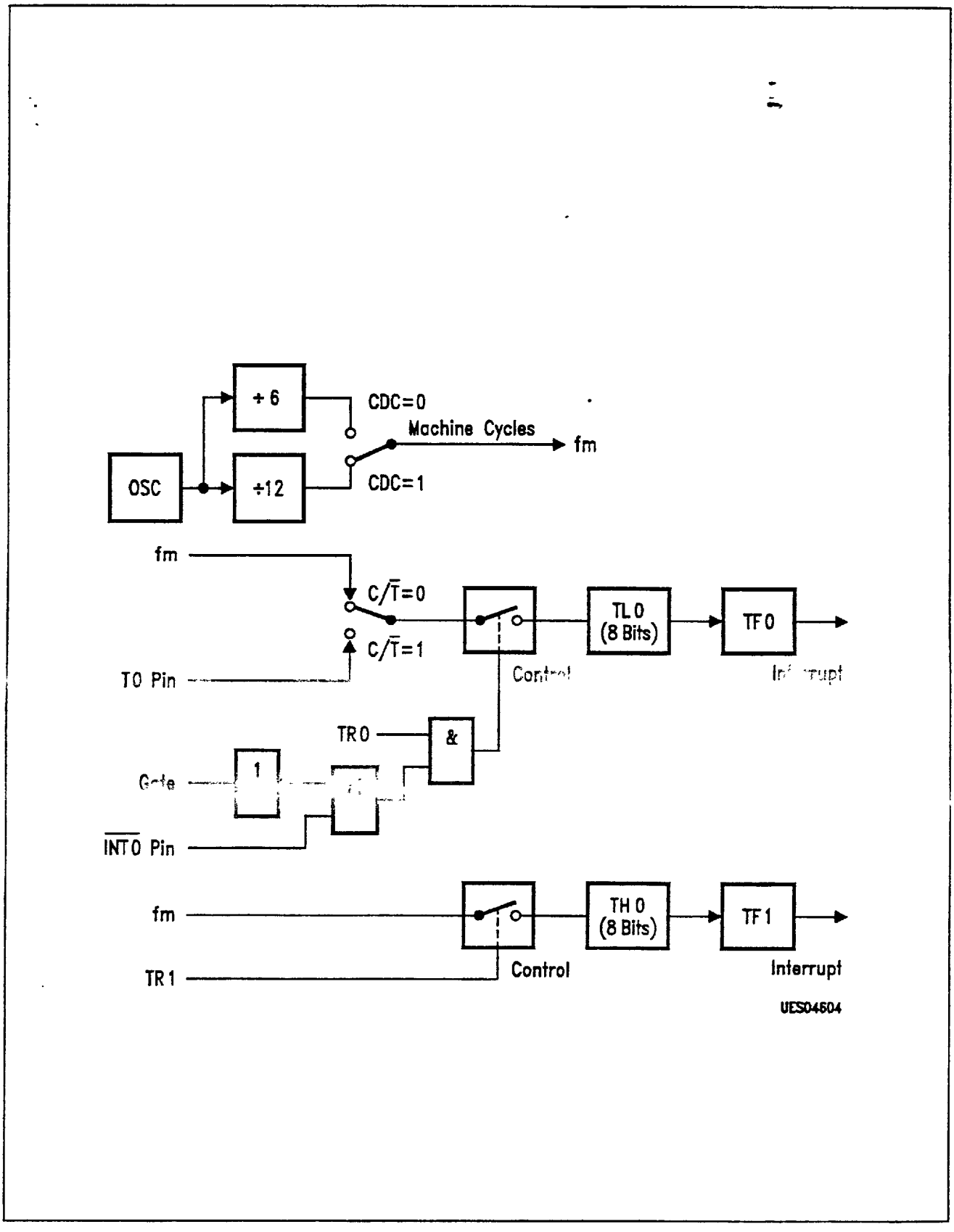


Figure 20
Timer/Counter 0 Mode 3: Two 8-Bit Counters

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 counter with an 8 bit prescaler. Without the prescaler the counter is incremented every 12 internal clocks. If the prescaler is active the counter is incremented every 12*256 = 3072 internal clocks. The watchdog timer is initialized to the reload value programmed to WDTREL.6 – WDTREL.0. After an external reset register WDTREL is cleared to 00_H. 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 WDCON (bit 6). If the counter is stopped, and WDTREL is loaded with a new value, WDTH (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 (WDCON.2) and by the next instruction setting SWDT (WDCON.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 $7FFC_H$. 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 (WDCON.7) 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 = 80_H a maximum time period of about 8 s at 12 MHz internal clock can be achieved.

Watchdog Timer Control Register WDCON **SFR-Address $A7_H$**

Default after reset: $0XXX\ XXXX_B$

(MSB) (LSB)

WDTS	SWDT	-	-	-	WDT	-	-

WDTS Watchdog timer status flag: If bit WDTS is 1 after reset, the reset has been initialized by the watchdog timer. After external reset, WDTS is reset to 0. This bit can be written by software too.

SWDT Start watchdog timer: A write to WDCON with SWDT=1 and WDT=0 starts the WDT operation.

WDT Watchdog timer refresh flag: A write to WDCON with WDT=1 and SWDT=0 initialize a refresh cycle. The next write to WDCON must follow immeadiately with WDT=0 and SWDT =1 to execute the refresh cycle.

Note: SWDT and WDT can only be written, so read-modify-write opcodes are not useful.

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

Default after reset: 00_H

(MSB) (LSB)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

WDTREL.7 Prescaler bit. When set, the watchdog is clocked through an additional divide by 256 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.

Watchdog High Byte **WDTH** **SFR-Address 85_H**

Default after reset: 80_H

(MSB) (LSB)

–	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

WDTH.7 Not implemented.

WDTH.6...0 These are the upper 7 bits of the 15 bit watchdog counter. These bits are only readable.

Watchdog Low Byte **WDTL** **SFR-Address 84_H**

Default after reset: 00_H

(MSB) (LSB)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7...Bit 0 These are the lower 8 bits of the 15 bit watchdog counter. They are only readable.

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 clock, used by the serial interface.

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 1/6 of the internal clock.

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 1/16 or 1/32 of the internal clock.

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

Serial Port Control Register **SCON** **SFR-Address 98_H**

Default after reset: 00_H

(MSB) (LSB)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI	Bit
$9F_H$	$9E_H$	$9D_H$	$9C_H$	$9B_H$	$9A_H$	99_H	98_H	Address

Symbol	Position	Function
SMO 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.	internal clock / 6
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	internal clock / 16 or 32
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

For the following calculations f_{int} is equal to the internal clock.

The baud rate in mode 0 is fixed:

Mode 0 baud rate =

$$\frac{f_{int}}{6}$$

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 1/32 of the internal clock. If SMOD = 1, the baud rate is 1/16 of the internal clock. Contrary to the SAB 8051 SMOD is placed on SFR-address 87_H.

Mode 2 baud rate =

$$\frac{2^{SMOD}}{32} \times f_{int}$$

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 =

$$\frac{2^{SMOD}}{16} \times \text{Time 1 overflow rate}$$

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 =

$$\frac{2^{SMOD}}{32} \times \frac{f_{int}}{6 \times (256 - TH1)}$$

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:	2 MHz	12.0	X	X	X	X
Mode 2 max:	750 Kbaud	12.0	1	X	X	X
Mode 1, 3:	62.5 Kbaud	6.0	1	0	2	FF_H
	19.2 Kbaud	5.529	1	0	2	FD_H
	9.6 Kbaud	5.529	0	0	2	FD_H
	4.8 Kbaud	5.529	0	0	2	FA_H
	2.4 Kbaud	5.529	0	0	2	$F4_H$
	1.2 Kbaud	5.529	0	0	2	$E8_H$
	137.5 Baud	5.993	0	0	2	$1D_H$
	110 Baud	3.0	0	0	2	72_H
	110 Baud	6.0	0	0	1	$FEEB_H$

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 1/6 of the internal 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, 1 s shift out to the left. When the 0 that was initially loaded into the rightmost position arrives at the leftmost position in the shift register, it flags the RX-control block to do one last shift and load 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 FF_H 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, 1 s 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 $1FF_H$ is written to the input shift register.

At the 7th, 8th, and 9th counter states of each bit time, the bit detector samples the value of RxD. The value accepted is the value that was seen in at least 2 of the 3 samples. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back 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, 1 s 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.

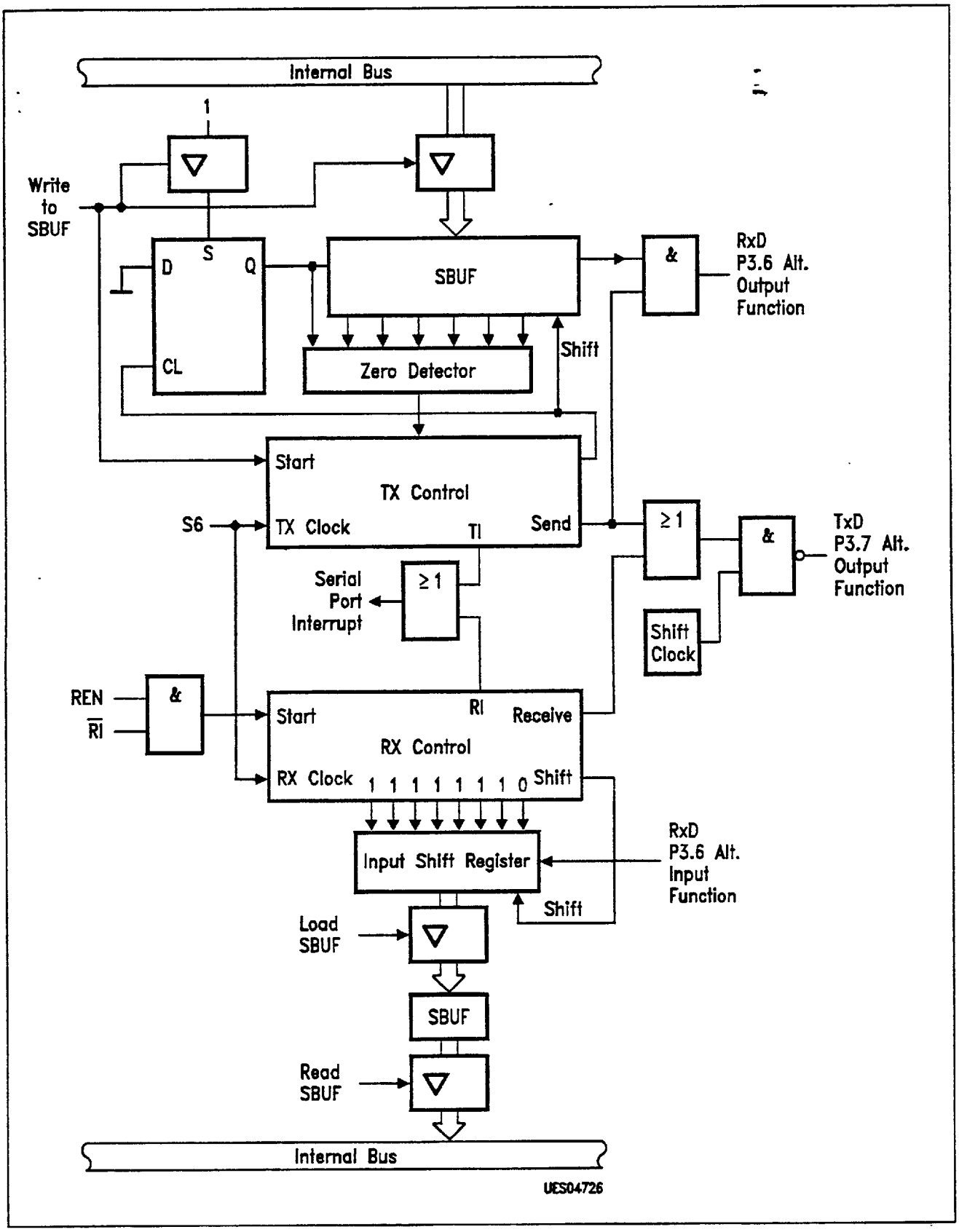


Figure 22 a
Serial Port Mode 0, Functional Diagram

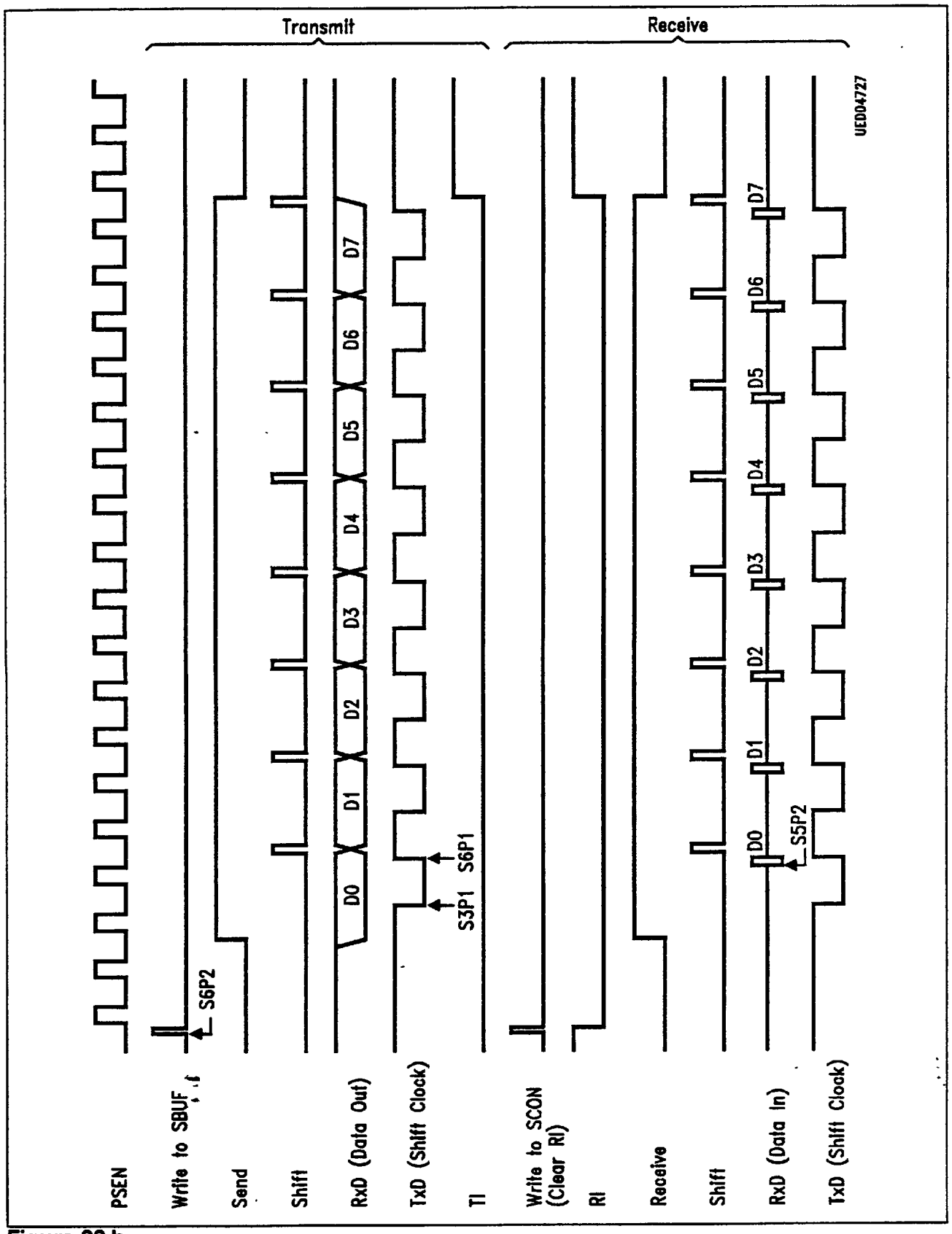


Figure 22 b
Serial Port Mode 0, Timing

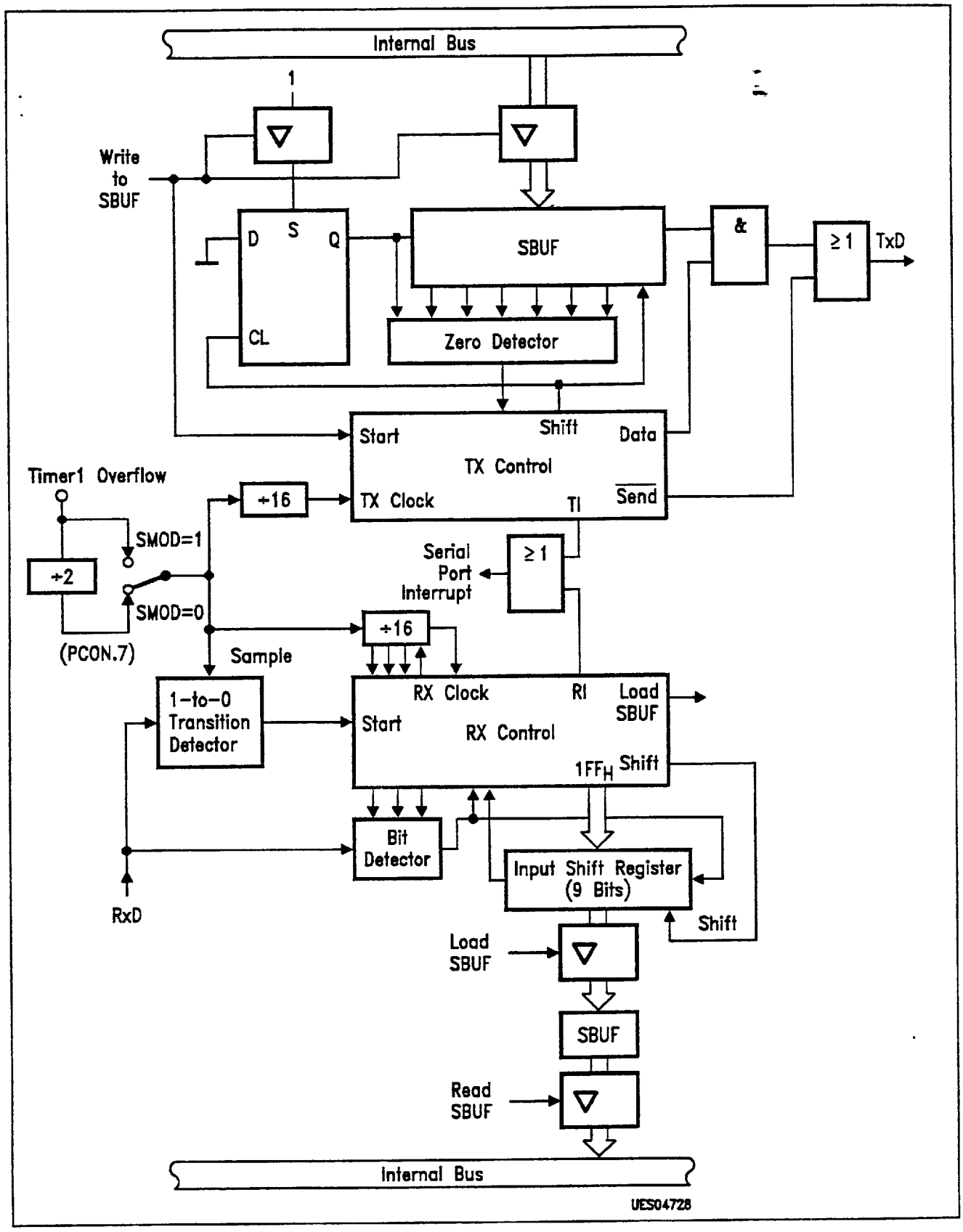


Figure 23 a
Serial Port Mode 1, Functional Diagram

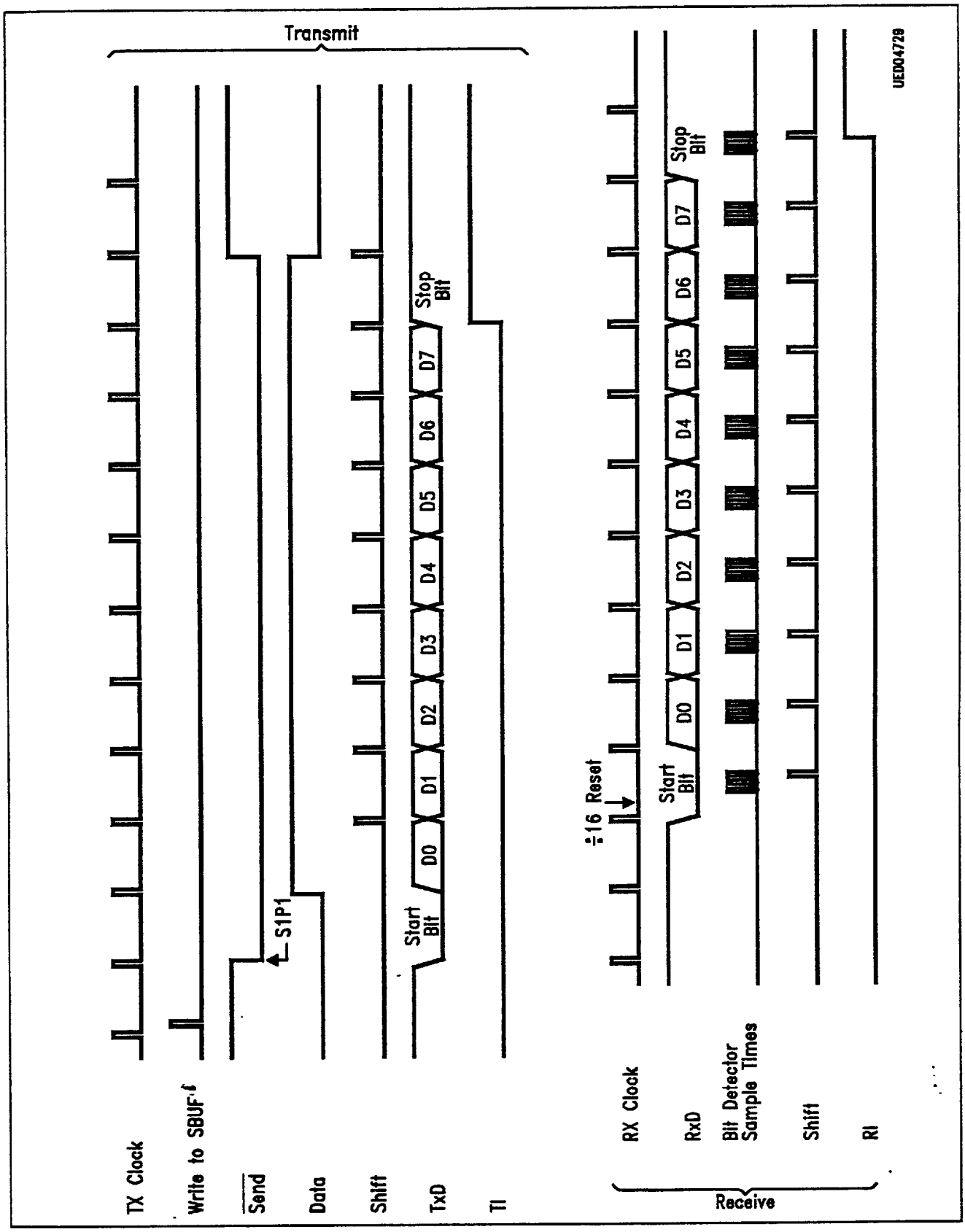


Figure 23 b
Serial Port Mode 1, Timing

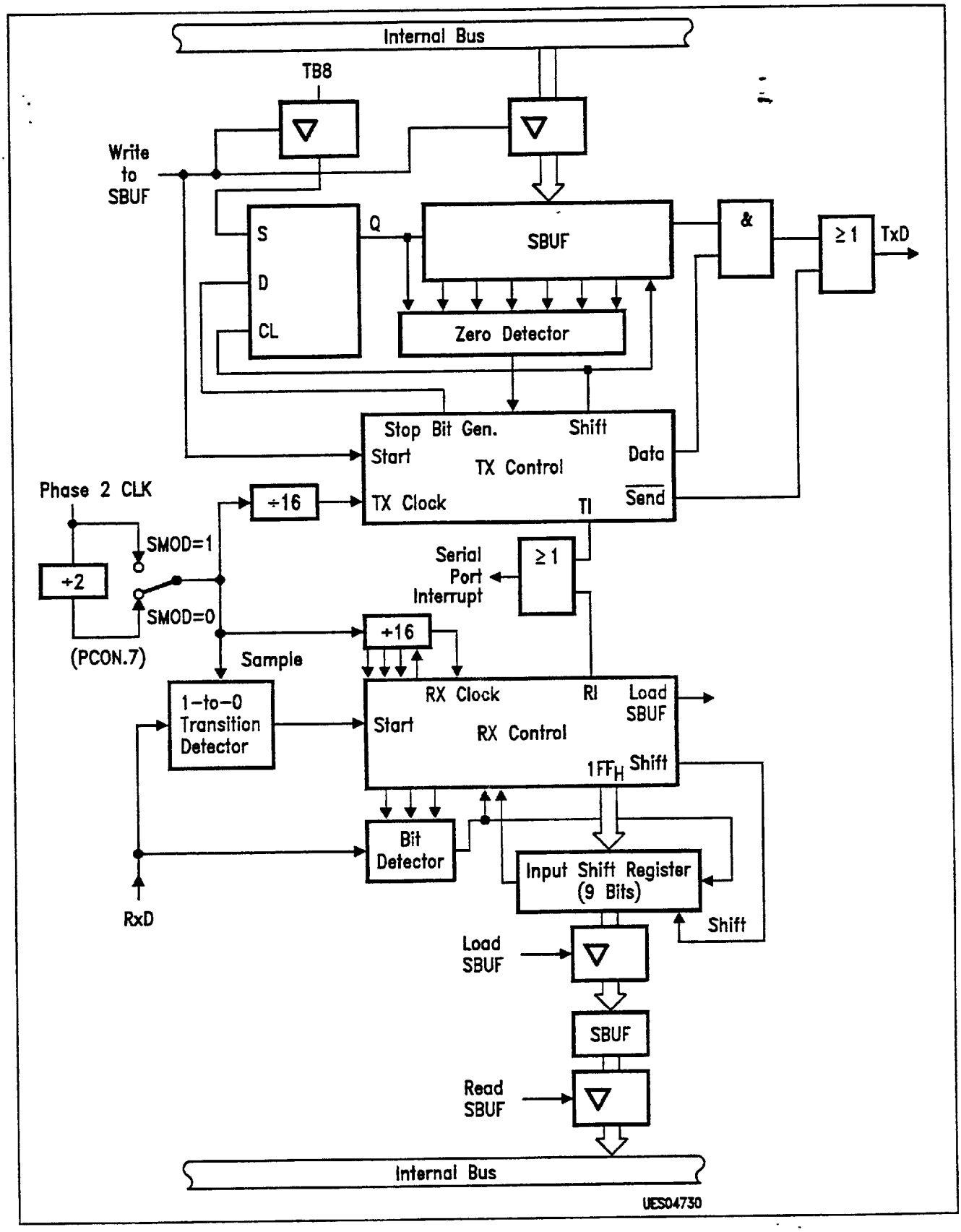


Figure 24 a
Serial Port Mode 2, Functional Diagram

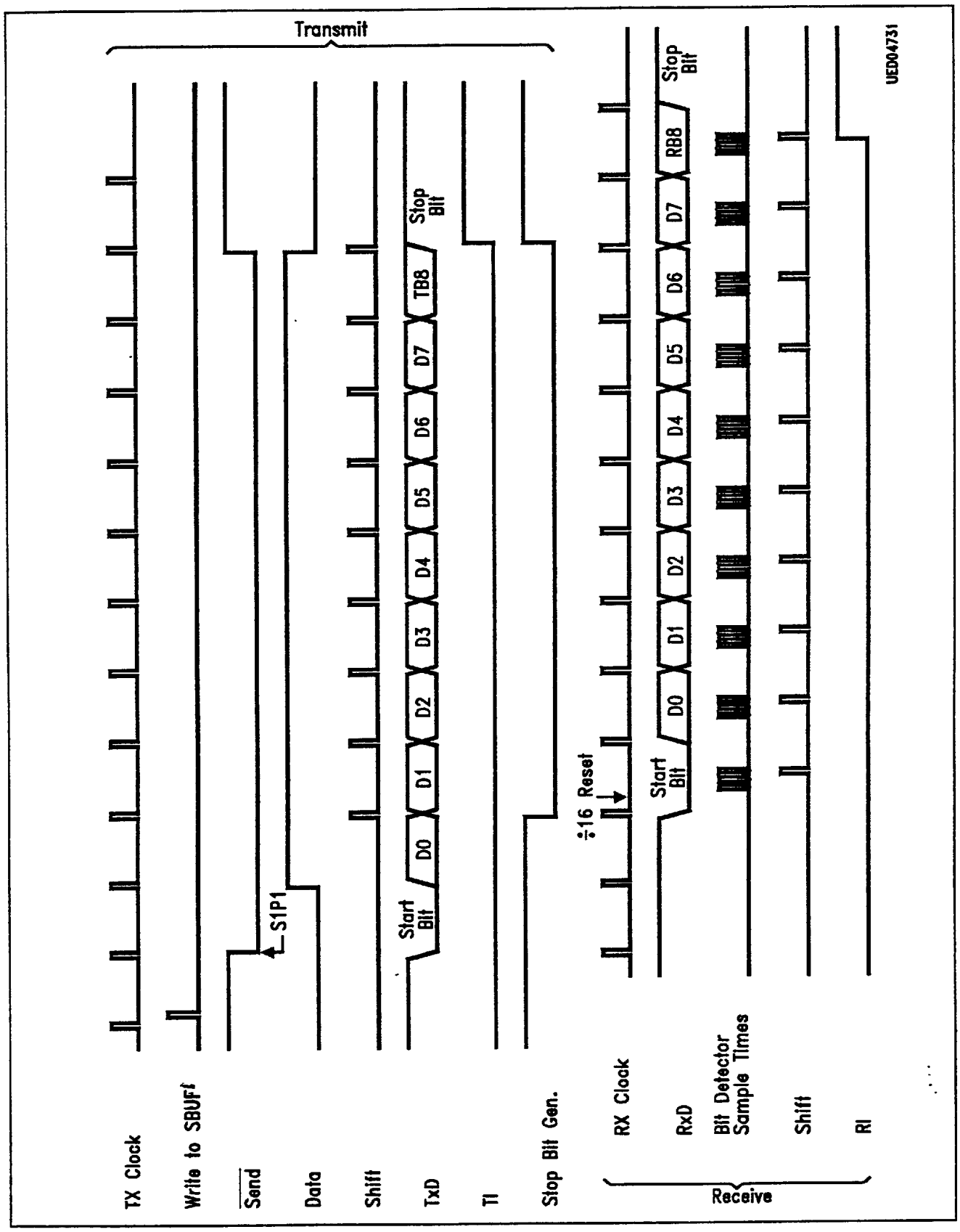


Figure 24 b
Serial Port Mode 2, Timing

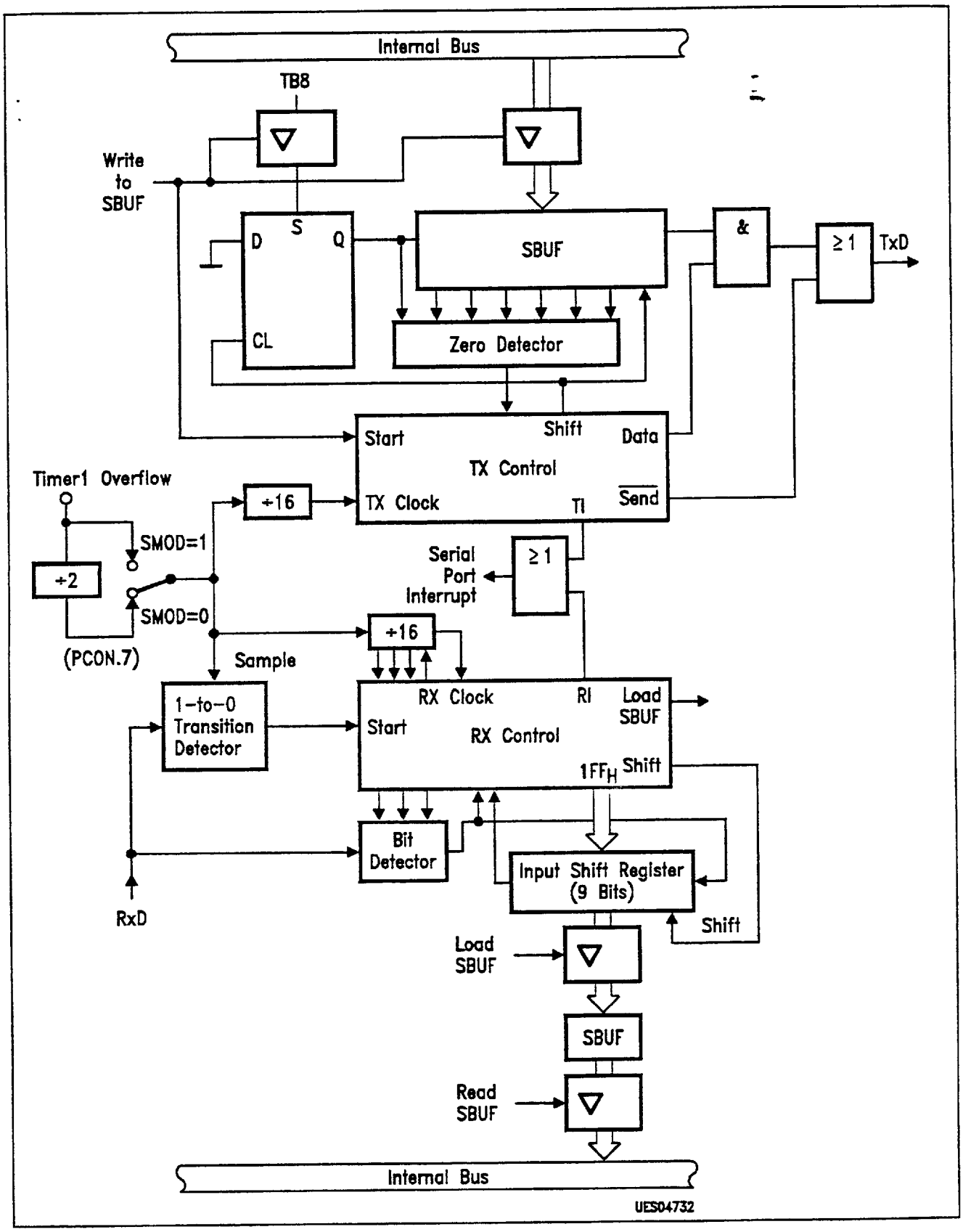


Figure 25 a
Serial Port Mode 3, Functional Diagram

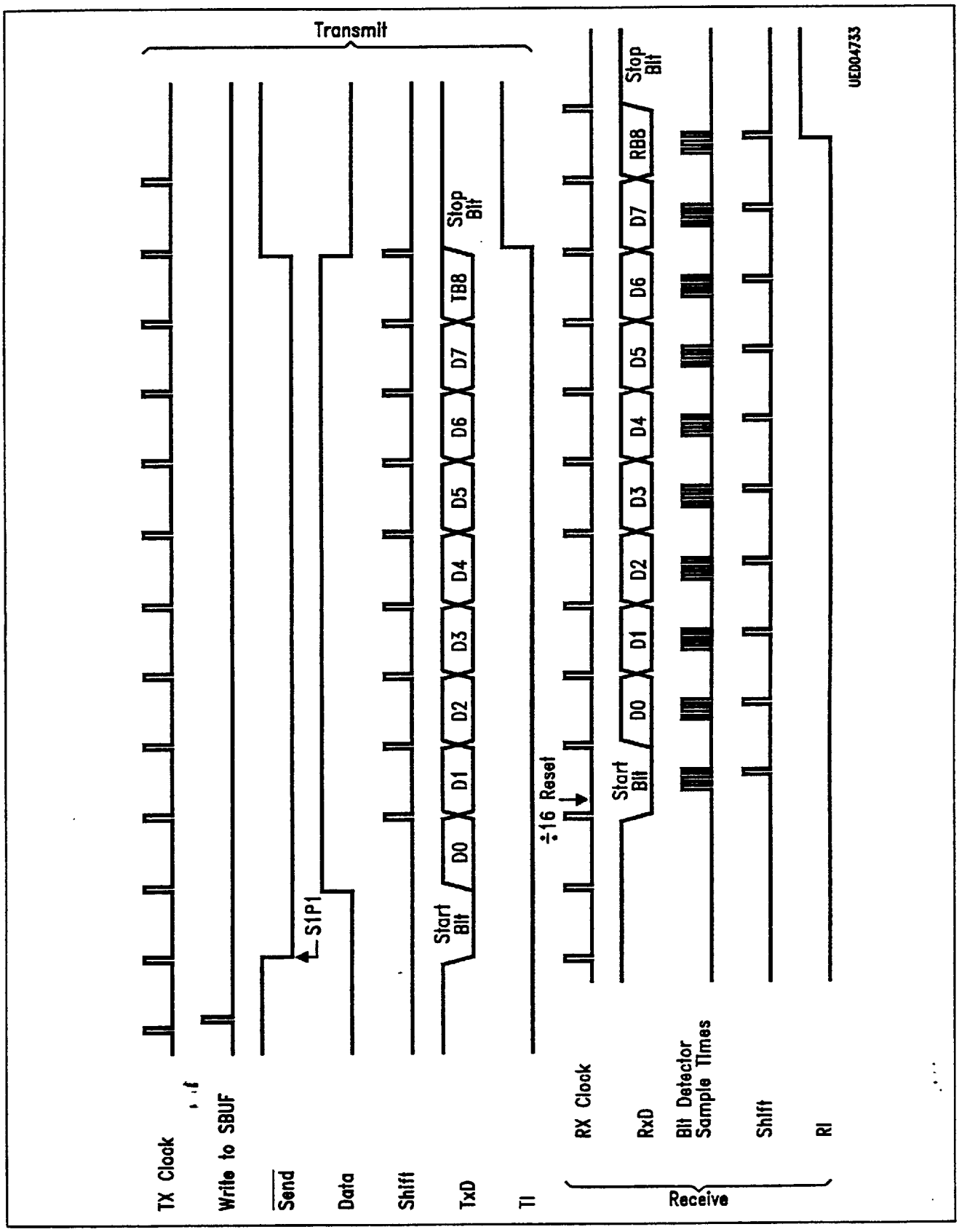


Figure 25 b
Serial Port Mode 3, Timing

2.9 Pulse Width Modulation Unit

The PWM-unit provides eight independent digital to analog conversion channels: six output channels with 8 bit resolution and two output channels with 14 bit resolution. Controlled via special function registers, each channel can be enabled individually. The base frequency of each channel depends on its resolution. The 8 bit channels have a cycle time of 64 internal clocks. The 14 bit channels have a cycle time of 256 internal clocks. For an internal clock of 8 MHz, this results in 187 kHz output frequency for 8 bit channels and 47 kHz for 14 bit channels.

The 8 bit channels use the 6 most significant bits of PWCOMPx for generating the base high/low ratio of the output signal. With the 2 least significant bits of PWCOMPx the high/low ratio is modified for an overall resolution of 8 bits.

The 14 bit channels use PWCOMPx for generating the base high/low ratio of the output signal. With the 6 most significant bit of PWEXTx the high/low ratio is modified for an overall resolution of 14 bits.

General Considerations

The PWM-output channels are placed as alternate functions to the eight lines of port 1. P1.0...P1.5 contain the 6 output channels with 8-bit resolution and P1.6...P1.7 the 2 output channels with 14-bit resolution. Each PWM-channel can be individually switched between PWM-function and port function.

The six 8-bit compare registers PWCOMP0 - PWCOMP5 are located at SFR-addresses $0F1_H$ - $0F6_H$. The two 14-bit compare registers consist each of an 8-bit register PWCOMP6 or PWCOMP7 and of a six-bit extension register PWEXT6 or PWEXT7, all located at SFR-addresses $0FA_H$-$0FD_H$. They 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 after the next timer overflow or if the PWM-timer is stopped.

The PWM-timer register located at SFR-address F7 and $F9_H$ contain the actual value of the PWM-counter low byte and high byte 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.

The internal timer of the PWM unit is running as long as at least one PWM-channel is enabled by the PWM-Enable Register PWME. At timer overflow of the 8-bit [14-bit] timer, all output latches OL0...OL5 [OL6 and OL7] are set to 1. If the timer value meets the compare value of channel i, OLi is reset to 0.

PWM-Enable Register PWME **SFR-Address $F8_H$**

Default after reset: 00_H

(MSB) (LSB)

E7	E6	E5	E4	E3	E2	E1	E0

E7...E0 = 0: The corresponding PWM-channel is disabled.
P1.i functions as normal bidirectional I/O-port.
= 1: The corresponding PWM-channel is enabled. E0...E5 are channels with 8-bit resolution, while E6 and E7 are channels with 14-bit resolution.

PWM Compare Registers PWCOMP 0-5 **SFR-Address $F1_H$-$F6_H$**

Default after reset: 00_H

(MSB) (LSB)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7...Bit 2 : This bits define the high time of the output. If all bits are 0, the high time is 0 internal clocks. If all bits are 1, the high time is 63 internal clocks.

Bit 1 : If this bit is set, every second PWM-Cycle is stretched by one internal clock, regardless of the settings of Bit7...Bit2.

Bit 0 : If this bit is set, every fourth PWM-Cycle is stretched by one internal clock, regardless of the settings of Bit7...Bit2.

Note: The stretch operation is interleaved between PWM-Cycles.

PWM Compare Registers PWCOMP 6,7 **SFR-Address FB_H,FD_H**

Default after reset: 00_H

(MSB) (LSB)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7...Bit 0 : This bits define the high time of the output. If all bits are 0, the high time is 0 internal clocks. If all bits are 1, the high time is 255 internal clocks.

PWM Extension Registers PWEXT 6,7 **SFR-Address FA_H,FC_H**

Default after reset: 02_H

(MSB) (LSB)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7 : If this bit is set, every second PWM-Cycle is stretched by one internal clock.
Bit 6 : If this bit is set, every fourth PWM-Cycle is stretched by one internal clock.
Bit 5 : If this bit is set, every eighth PWM-Cycle is stretched by one internal clock.
Bit 4 : If this bit is set, every 16th PWM-Cycle is stretched by one internal clock.
Bit 3 : If this bit is set, every 32th PWM-Cycle is stretched by one internal clock.
Bit 2 : If this bit is set, every 64th PWM-Cycle is stretched by one internal clock.
Bit 1, Bit 0 : This bits must always be set to 0.

Note: **1)** The described operation is independent of the setting of PWCOMP6 or PWCOMP7.
2) The stretch operation is interleaved between PWM-Cycles.

PWM Low Counter Registers PWCL **SFR-Address $F7_H$**

Default after reset: 00_H

(MSB) (LSB)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7...Bit 0 : This bits are the low order 8 Bits of the 14 Bit PWM-Counter. This register can only be read.

PWM High Counter Registers PWCH **SFR-Address $F9_H$**

Default after reset: $c0_B$

(MSB) (LSB)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7, Bit 6 : This bits are undefined.
Bit 5...Bit 0 : This bits are the high order 6 Bits of the 14 Bit PWM-Counter. This register can only be read.

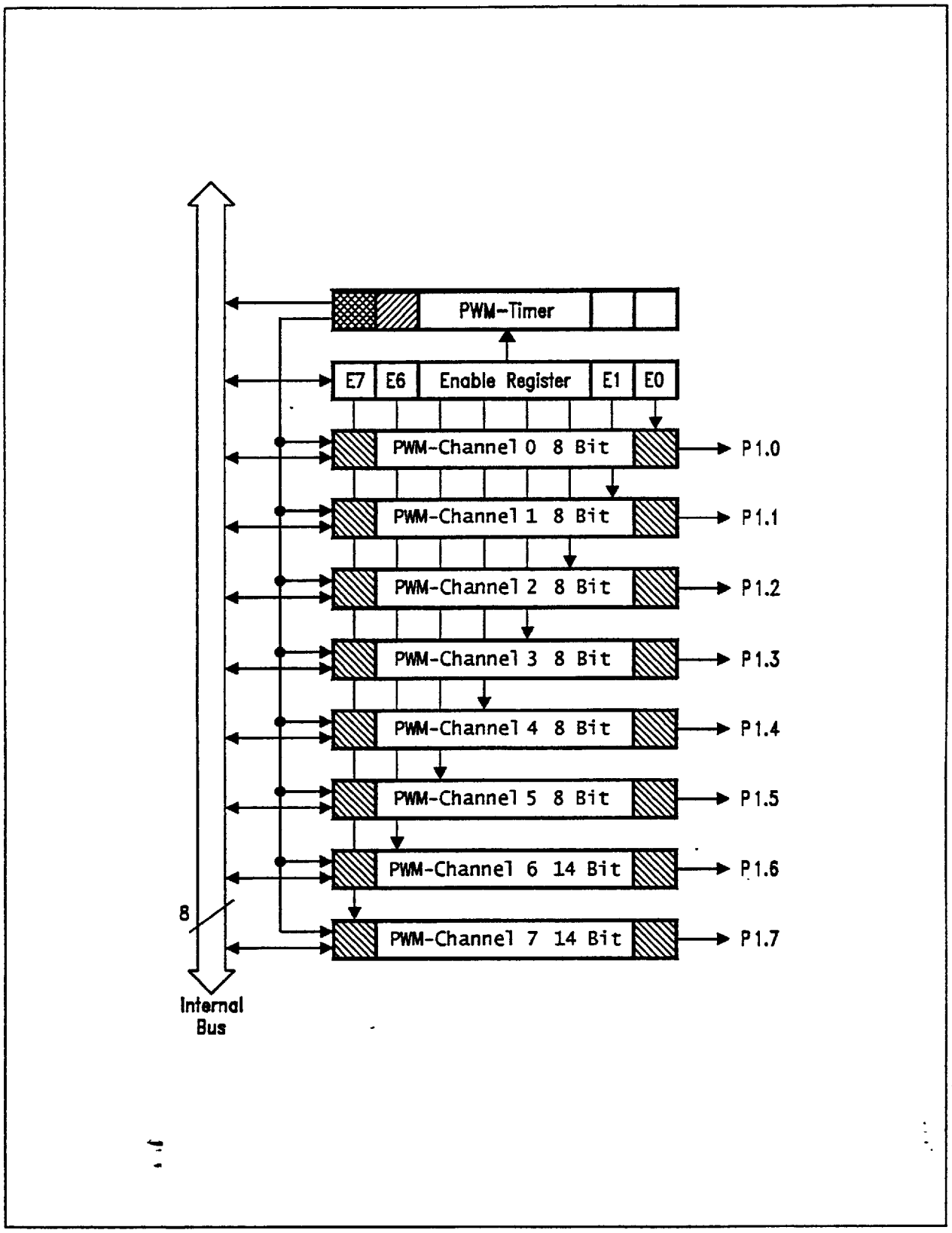


Figure 26
Block Diagram of Pulse Width Modulation Unit

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
- 6.7 to 21.3 µs conversion time at 12 MHz internal clock

The conversion time depends on the internal clock and the setting of PSC and SAMP in ADCON. If PSC and SAMP are 0 , the sample period is 16 internal clocks and the approximation time is 64 internal clocks. If SAMP is set to 1 the sample time is increased fourfold to 64 internal clocks. If PSC is set to 1 the selected sample time and the approximation time are doubled.

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

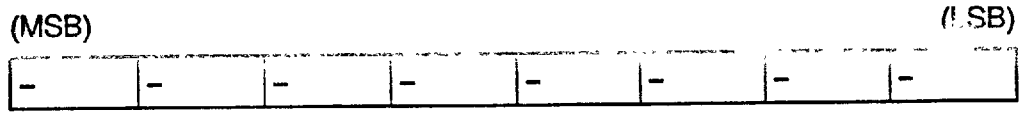
Special function register ADCON is used to set the operation modes, to check the status and to select one of four input channels. ADCON contains two mode bits. Bit ADM is used to choose the single or continuous conversion method. In single conversion mode (ADM = 0) only one conversion is performed after 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.

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. A new conversion can be started only after reading ADDAT. 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 00_H.

ADC-Start Register **DAPR** **SFR-Address DA_H**

Default after Reset: XX_H



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

ADC-Control Register ADCON **SFR-Address $D8_H$**

Default after reset: 00_H

(MSB) (LSB)

PSC	SAMP	–	BSY	ADM	0	MX1	MX0

This register is bit addressable.

PSC Prescaler control: PSC = 0 for prescaler not active, internal master clock of the ADC is equal to the internal clock. PSC = 1 for prescaler active, internal master clock of ADC is half of the internal clock.

SAMP Sample time : If SAMP=1 the sample time is 64 internal clocks, if SAMP=0 the sample time is 16 internal clocks.

ADCON.5 Reserved

ADCON.2 Always to be written with '0'

BSY Busy flag: BSY = 1 during conversion

ADM ADC-conversion mode: ADM = 0 for single and ADM = 1 for continuous conversion.

MX1, MX0 ADC-channel select

MX1	MX0	Selected Channel
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 $D9_H$**

Default after reset: XX_H

(MSB) (LSB)

AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

ADDAT.7-.0 8-Bit Analog Data Value

2.10.1 Analog Detector

The ADC contains special logic to simplify the task of supervising an analog voltage. For this purpose, several registers are implemented.

Detector-Control Register DECON **SFR-Address DC_H**

Default after reset: $0000\ 0XXX_B$

(MSB) (LSB)

OV	UN	0	REQ	END	TI2	TI1	TI0

OV OVERFLOW - Flag : OV = 1 if DEUP - ADDAT < 0.

UN UNDERFLOW - Flag : UN = 1 if ADDAT - DELOW < 0.

DECON.5 Always to be written with '0'

REQ Request Flag : REQ = 1 if CPU conversion request is pending. Reset to 0 after start of conversion.

END Enable Detector : if END = 1 the detector is active.

TI2,TI1,TI0 Timing Intervall select

TI2	TI1	TI0	Selected Timing Intervall in internal machine cycles
0	0	0	512
0	0	1	1024
0	1	0	2048
0	1	1	4096
1	0	0	8192
1	0	1	16384
1	1	0	32768
1	1	1	65536

Detector-Select Register DESEL **SFR-Address DD_H**

Default after reset: undefined

(MSB) (LSB)

-	-	-	-	-	-	DX1	DX0

DESEL.7 - .2 Reserved

DX1, DX0 ADC-channel select for monitoring

DX1	DX0	Selected Channel
0	0	0
0	1	1
1	0	2
1	1	3

Detector-Lower Register DELOW **SFR-Address DE_H**

Default after reset: undefined

(MSB) (LSB)

DL7	DL6	DL5	DL4	DL3	DL2	DL1	DL0

DELOW.7 - .0 8-Bit lower Limit

Detector-Upper Register DEUP **SFR-Address DF_H**

Default after reset: undefined

(MSB) (LSB)

DU7	DU6	DU5	DU4	DU3	DU2	DU1	DU0

DEUP.7 - .0 8-Bit upper Limit

Operation description

The detector implements an "out-of-range" supervising function. This can be used to avoid software polling of the ADC in certain applications. 4 registers are available to control the detector. The lower 2 bits of DESEL selects the input channel for the detector independent of MX1 and MX0 in ADCON. The input switch is done automatically. After the detector cycle has finished, MX1 and MX0 control the input channel again. DEUP holds the upper limit, DELOW holds the lower limit. The register DECON contains status and control bits. OV and UN are updated by hardware at the end of a detector cycle. OV is set to 1 if DEUP-ADDAT< 0. UN is set to 1 if ADDAT-DELOW< 0. All values are interpreted as zero-extended integers. The flags can be set and reset by software too. END enables the detector. TIx selects one of 8 timing intervals. After each timing intervall a request for conversion is generated. The detector has its own 16 bit counter, which is incremented every machine cycle.

The operation-sequence starts after the selected timing interval. The ADC starts a conversion of the channel selected by DESEL. If a conversion is in progress, the detector-request will be stored and serviced after the CPU has read ADDAT. This insures that the CPU has received the result of the conversion. The result of the detector cycle is stored in ADDAT. Now the comparator operations are performed and the coresponding flags are set or reset. An interrupt is generated if EAD is set to 1 in IE and OV or UN is 1. During the detector cycle the BSY bit is set to show that a conversion is in progress. If the CPU makes a request for conversion during this time, the request will be stored and serviced after the running conversion. The bit REQ in DECON shows the pending CPU request. It is automatically cleared after the conversion has started. The request can be cancelled, if REQ is set to 0. If an "out-of-range" value is detected, the detector will disable further conversions until the processor has cleared the bits OV and UN. This prevents overwriting of ADDAT until the CPU has checked the value, under the assumption that a conversion request is pending.

If DELOW is set to 0 , an underflow is impossible. If DEHI is set to 255 , an overflow is impossible. If DELOW = DEHI a certain value can be supervised (This implies noise-limited analog signals !)

2.11 I^2C Serial Interface

The I^2C serial interface has been designed to satisfy the multi-master requirements of the I^2C Bus Specification. All enhancements like 10 bit addressing and 400 kbit/s data rate are included.

This interface detects, receives and converts the serial data stream to parallel format without interrupting the execution of the current program. An interrupt request is generated when the complete byte has been received; the microcomputer can then read the data byte in a single instruction. Likewise, for transmission, the serial interface performs the parallel to serial conversion and subsequent serial output of the data while the microcomputer continues with execution of its programmed tasks. After completion of the transmission an interrupt is generated.

The interface also provides simple mechanisms for acknowledgement of data reception.

The serial interface works with a serial bus consisting of two bi-directional lines SDA and SCL : one for data signals and one for clock signals, respectively. A protocol has been defined to allow reliabel and efficient operation of this bus. During data transfer, the SDA line must remain stable whenever the SCL line is high. Changes of the SDA line while the SCL line is high will be interpreted as control signals.

Accordingly, the following bus conditions have been defined :

a) Bus not busy : both SDA and SCL lines remain high.

b) Start data transfer : a change in the state of the SDA line, from high to low, while the SCL line is high, defines the start condition.

c) Stop data transfer : a change in the state of the SDA line, from low to high, while the SCL line is high, defines the stop condition.

d) Data valid : the state of the SDA line represents valid data when, after a start condition, the SDA line is stable for the duration of the high period of the clock signal. The data on the line may be changed during the low period of the clock signal. There is one clock pulse per bit of data.

Figure 2.11.1 illustrates the sequence of events involved in data transfer on the serial bus. Each data transfer is initiated with a start condition and terminated with a stop condition; the number of bytes transferred between the start and stop conditions is not limited; one byte consists of eigth bits.

2.11.1 Registers and Hardware-Interface

The I^2C interface in the SDA 30C164 has multiple operation options. Two complete SDA/SCL line pairs are available : channel 0 and channel 1. In a special mixed mode, one SCL line could be shared for both SDA lines. The advantage of such an arrangement is that more addresses are available and different speeds can be used, one interface with 100kbit/s original specification, the other with the new 400kbit/s specification. Therefore the baud rate for each channel is programmable. A total of 6 registers is available for the programmer.

I^2C-Control Register ICCON SFR-Address $B8_H$

Default after reset: $00000X00_B$

(MSB) (LSB)

SLA	TRX	BB	IIN	AL	LRB	ACK	BUM

This register is bit addressable.

SLA Slave : If SLA = 1 the interface is active as a slave on the selected bus. SLA is set to 1 after receiving the correct device address. SLA is reset to 0 after a stop condition. SLA = 0 has two meanings : 1. the bus is not busy or 2. the interface is in master mode.

TRX Transmit select : In all modes, TRX = 1 selects transmission of data, TRX = 0 selects reception of data.

BB Bus Busy : This bit can only be read. If the I^2C interface is active, this bit monitors the I^2C bus status. After a start condition on the selected I^2C bus, BB is set to 1. After a stop condition on the selected I^2C bus, BB is reset to 0. If the I^2C interface is inactive, this bit is always 0.

IIN I^2C Interrupt : After transmitting or receiving the acknowledge bit, IIN is set to 1. If the state of SLA or AL changes from 0 to 1 IIN is also set to 1.IIN can be set or reset by software.If IIN is 1 and the device is master or selected as a slave SCL is held low. Resetting IIN to 0 releases SCL and starts a data transmission.

AL Arbitration lost : If this bit is set, the interface has tried to become a master on the bus, but has lost the arbitration. The interface continues operation until the 9th clock pulse. IIN is set after the 9th clock pulse. Software has to reset this bit to 0.

LRB Last received bit : If the interface is in transmitter mode, this bit represents the acknowledge bit of the receiver of the last transmission.

ACK Acknowledge Pulse : If the interface is in the receiver mode and ACK = 0, an acknowledge pulse is generated; if ACK = 1 no pulse is generated.

BUM Busy Master : in master or multimaster mode this bit can be set to 1. Otherwise it is always 0. A 0-to-1 transition of BUM generates a start condition, a 1-to-0 transition generates a stop condition. If BB = 1 and BUM is set to 1 an arbitration lost situation occurs. BUM is reset to 0 and AL ist set to 1.

I^2C-Mode Register ICMOD **SFR-Address $B9_H$**

Default after reset: $000X00XX_B$

(MSB) (LSB)

CSEL	EMA	ES	ADR	RSC	M10	ICA9	ICA8

CSEL Channel select : CSEL = 0 selects channel 0 for the following operation, CSEL = 1 selects channel 1. Note : If CSEL is changed during BB = 1 operation becomes unpredictable !

EMA Enable Master Mode : see table below for definition of operation modes.

ES Enable Slave Mode : see table below for definition of operation modes.

ADR Address : After a start condition the first byte in 8 bit mode is the address and ADR is set to 1. In 10 bit mode the first two bytes received have ADR set to 1.

RSC Repeated Start Condition : If the interface is active as a master, setting this bit generates a repeated start condition. RSC is automatically reset to 0 after sending the repeated start condition. RSC can not be set in slave mode.

M10 10 bit addressing : if M10=1, the interface is selected by the 10 bit address defined by IA9 to IA0. If M10=0 only the higher 7 bits IA7 to IA1 are used.

ICA9,ICA8 I^2C Address Bits 9/8 : this are the two MSBs of the device address in 10 bit mode.

Possible Operation Modes :

EMA	ES1	Selected Operating Mode
0	0	interface disabled and initialized
0	1	slave mode
1	0	master mode
1	1	multimaster mode

I^2C-Shifter Register ICSHI **SFR-Address BA_H**

Default after reset: undefined

(MSB) (LSB)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

ICSHI.7 - .0 8-Bit transmit and receive shifter

Note : Write transfers to ICSHI are disabled by hardware during a transmission. Read operations are always possible.

I^2C-Address Register ICADR **SFR-Address BB_H**

Default after reset: undefined

(MSB) (LSB)

ICA7	ICA6	ICA5	ICA4	ICA3	ICA2	ICA1	ICA0

ICA7 - ICA0 8-Bit address register: ICA7-ICA1 define the device address in 8 bit mode, in 10 bit mode all bits are used.

I^2C-Baud Register 0 ICBD0 **SFR-Address BC_H**

Default after reset: $00XX\ XXXX_B$

(MSB) (LSB)

EDA0	ECL0	BD05	BD04	BD03	BD02	BD01	BD00

I^2C-Baud Register 1 ICBD1 **SFR-Address BD_H**

Default after reset: $00XX\ XXXX_B$

(MSB) (LSB)

EDA1	ECL1	BD15	BD14	BD13	BD12	BD11	BD10

EDA[1/0] Enable Data Pin : EDAx = 1 enables data output for I^2C operation.

ECL[1/0] Enable Clock Pin : ECLx = 1 enables clock output for I^2C operation.

BD[1/0]5..0 Baud Rate : the Baud Rate is 1/(4*n) internal clock cycles. n is the binary number in ICBDi5.. ICBDi0 plus 1, so n is in the range 1.. 64.
For example : 100 kBaud, 6 MHz internal clock -> n=15 -> ICBD0=CE_H .

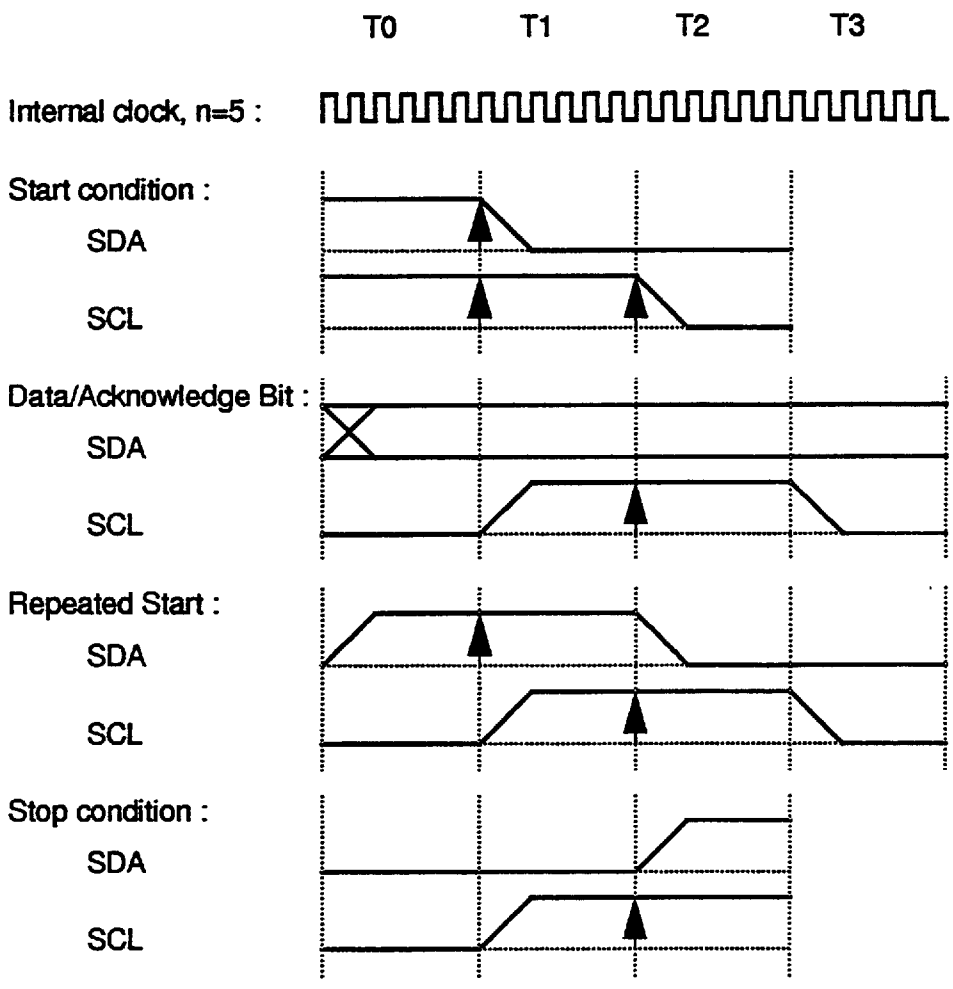
Possible Port Configurations :

EDA1	ECL1	EDA0	ECL0	P0.3	P0.2	P0.1	P0.0
1	1	0	X	SCL1	SDA1	PORT	PORT
1	1	1	0	SCL1/0	SDA1	SDA0	PORT
0	X	1	1	PORT	PORT	SDA0	SCL0
1	0	1	1	PORT	SDA1	SDA0	SCL1/0
1	1	1	1	SCL1	SDA1	SDA0	SCL0

Notes : 1) any other combination results in PORT mode for P0.3-P0.0!
2) It is always possible to read the status of SDAx/SCLx by usual I/O-commands.

2.11.2 Operation Description

After reset the port configuration must be defined. If the baud rates for both channels never change, they can be defined after reset once. Otherwise, they can be changed at any time.The desired operation mode can be selected at any time. In slave mode the device address has to be defined, 7 bit or 10 bit addressing can be selected. The interrupt enable bit EIC in IE may be set. ES has to be set to 1 to allow slave operation of the interface, CSEL is set to the desired channel. If the device is selected by another master, SLA changes from 0 to 1, an interrupt is generated and the CPU has to start the desired data transmission. If the device is receiver, TRX must be reset to 0. Resetting IIN to 0 starts the reception phase. At the same time, SCL is released to allow the data transfer. If the device is transmitter, the CPU first writes the data to ICSHI and then sets TRX to 1, selects the state of ACK and reset IIN to 0. This can be done with one byte write operation to ICCON. The device remains in slave mode until a stop condition is detected. This will also reset SLA to 0. If master mode is required, the interface must have EMA set to 1. Before any transmission, the CPU has to select the desired channel by setting or clearing CSEL. If the channel remains the same, no action is required. To become master on the bus is simple, if the device is the only possible master in the system. Generation of start and stop conditions is done by setting or clearing BUM. Transfer operations are controlled in the same manner like in slave mode. After setting BUM, the data contained in ICSHI is shifted out automatically after the start condition. A repeated start condition can be generated by setting RSC to 1. RSC is reset to 0 automatically at the end of the repeated start condition. The CPU must write the slave address to ICSHI before setting RSC. The interface will automatically send the slave address after the repeated start condition. A stop condition is generated when BUM is reset to 0. In Multimaster mode all the operations described above remains the same. The only exception is, that the bit AL in ICCON has to be checked by the CPU. If the interface lost the arbitration, AL is set to 1 and the interface switches to slave mode. If BB is set to 1 by a start condition on the bus and the CPU tries to set BUM, the request is suppressed and AL is set to 1. AL can only be cleared by software.



▲ the high level of the signal is verified. If it is low, Ti is repeated.

T0..T3 each Ti has a length of 1...64 internal clocks as defined in ICBD0 and ICBD1.

Figure 27 I^2C Bus events

2.12 Advanced Function Register

This register contains some control bits, which enable and disable special functions or enhancements to previous controller generations. Therefore it is strongly recommended to use and to program only the described values !

The on-chip clock generator of the SDA 30C164 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 on or off by software.

Advanced Function Register AFR **SFR-Address $A6_H$**

Default after reset: $0000\ 0XX1_B$

(MSB)							(LSB)
CDC	0	0	0	ERW	0	0	DJMP

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. After reset CDC is 0.

AFR.7 – .5 Reserved, always to be written with 0.

AFR.2 ,.1 Reserved, always to be written with 0.

ERW Enable Read/Write : If set to 1, the Port Pins P31 and P30 are used for $\overline{RD}$ and $\overline{WR}$ respectively. This can be used to increase the available XRAM address space externally. MOVX write operations output the internal data [illegible] read operations read data from D7-D0, if the address is below $FC00_H$. After reset ERW is 0.

DJMP Disable Jump Mode : This bit controls the behaviour of the banking logic for the opcode JMP@A+DPTR. If this bit is set to 1, no banking operation is performed. If this bit is set to 0, JMP@A+DPTR switches in the same way as LJMP to the desired bank. After reset DJMP is 1.

2.13 Instruction Set

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

2.13.1 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 2048 bytes of the extended internal data RAM (XRAM).

2.13.2 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.

2.13.3 Instruction Set Description

Arithmetic Operations

Mnemonic		Description	Byte	Cycle
ADD	A, Rn	Add register to Accumulator	1	1
ADD	A, direct	Add direct byte to Accumulator	2	1
ADD	A, @Ri	Add indirect RAM to Accumulator	1	1
ADD	A, #data	Add immediate data to Accumulator	2	1
ADDC	A, Rn	Add register to Accumulator with Carry flag	1	1
ADDC	A, direct	Add direct byte to A with Carry flag	2	1
ADDC	A, @Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A, #data	Add immediate data to A with Carry flag	2	1
SUBB	A, Rn	Subtract register from A with Borrow	1	1
SUBB	A, direct	Subtract direct byte from A with Borrow	2	1
SUBB	A, @Ri	Subtract indirect RAM from A with Borrow	1	1
SUBB	A, #data	Subtract immediate data from A with Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment Data Pointer	1	2
MUL	AB	Multiply A & B	1	4
DIV	AB	Divide A & B	1	4
DA	A	Decimal Adjust Accumulator	1	1

Logical Operations

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

Data Transfer Operations

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

Boolean Variable Manipulation

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

Program and Machine Control Operations

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

Program and Machine Control Operations (cont'd)

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

2.13.4 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

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_{stg}	– 65 to 125	°C

3.2 DC-Characteristics

T_A = – 20 to 85 °C; V_{DD} = 5 V ± 10 %, V_{SS} = 0 V
(C_L = 100 pF for port 0 and $\overline{PSEN}$, 80 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 mA
L-output voltage (P03...P00 only)	V_{OL1}	–	0.6	V	I_{OL} = 5.0 mA
H-output voltage (ports 1, 3, 4 P24...P27)	V_{OH}	2.4	–	V	I_{OH} = – 40 µA
H-output voltage (A0 ... A16, PWM-/UART-mode, ALE, $\overline{PSEN}$)	V_{OH1}	2.4	–	V	I_{OH} = – 1.6 mA
Logical 0 input current (D0...D7, ports 1,3,4, P24...P27)	I_{IL}	- 50	– 200	µA	V_{IN} = 0.45 V
Input leakage current (port 0, P20...P23)	I_{LI}		± 10	µA	0.45 V ≤ V_{IN} ≤ V_{DD}
Logical 1 input current ($\overline{RST}$)	$I_{\overline{RST}}$	2	4	mA	$V_{\overline{RST}}$ = 1.5 V
Reset Voltage ($\overline{RST}$)	$V_{\overline{RST}}$	0	1	V	
Power supply current	I_{DD}		50	mA	V_{DD} = 5 V; f_{osc} = 8 MHz

DC-Characteristics (cont'd)

T_A = – 20 to 85 °C; V_{DD} = 5 V ± 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	µA	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}			± 50	µA	
ADC-differential non-linearity	*DNLE*		0.5	1.0	LSB	T_A > 0 °C
ADC-offset error	*OE*		0.5	1.0	LSB	T_A > 0 °C
ADC-gain error	*GE*		0.5	1.0	LSB	T_A > 0 °C
ADC-differential non-linearity	*DNLE*		1.0	2.0	LSB	T_A < 0 °C
ADC-offset error	*OE*		1.0	2.0	LSB	T_A < 0 °C
ADC-gain error	*GE*		1.0	2.0	LSB	T_A < 0 °C
Analog ground voltage	V_{AGND}	V_{SS}		0.5 × V_{DD}	V	
Analog reference voltage	V_{AREF}	0.5 × V_{DD}		V_{DD}	V	
Analog supply difference voltage	V_{ADELTA}	0.5 × 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}			± 50	µA	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 ± 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 $1/t_{CLCL}$ = 1.0 MHz to 12 MHz		
		min.	max.	
Cycle time	t_{CY}	6 t_{CLCL}	–	ns
$\overline{PSEN}$ pulse width	t_{PLPH}	1.5 t_{CLCL}	–	ns
$\overline{PSEN}$ low to valid instr in	t_{PLIV}	–	0.8 × t_{CLCL}	ns
Input instruction hold after $\overline{PSEN}$ high	t_{PXIX}	0	–	ns
Address out to valid instr in	t_{AVIV}	–	2.3 × t_{CLCL}	ns
Address valid after $\overline{PSEN}$ high	t_{PXAV}	0	–	ns
Oscillator period	t_{CLCL}	83.3	1000	ns
External clock high time	t_{CHCX}	25		ns
External clock low time	t_{CLCX}	25		ns
External clock rise time	t_{CLCH}		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.

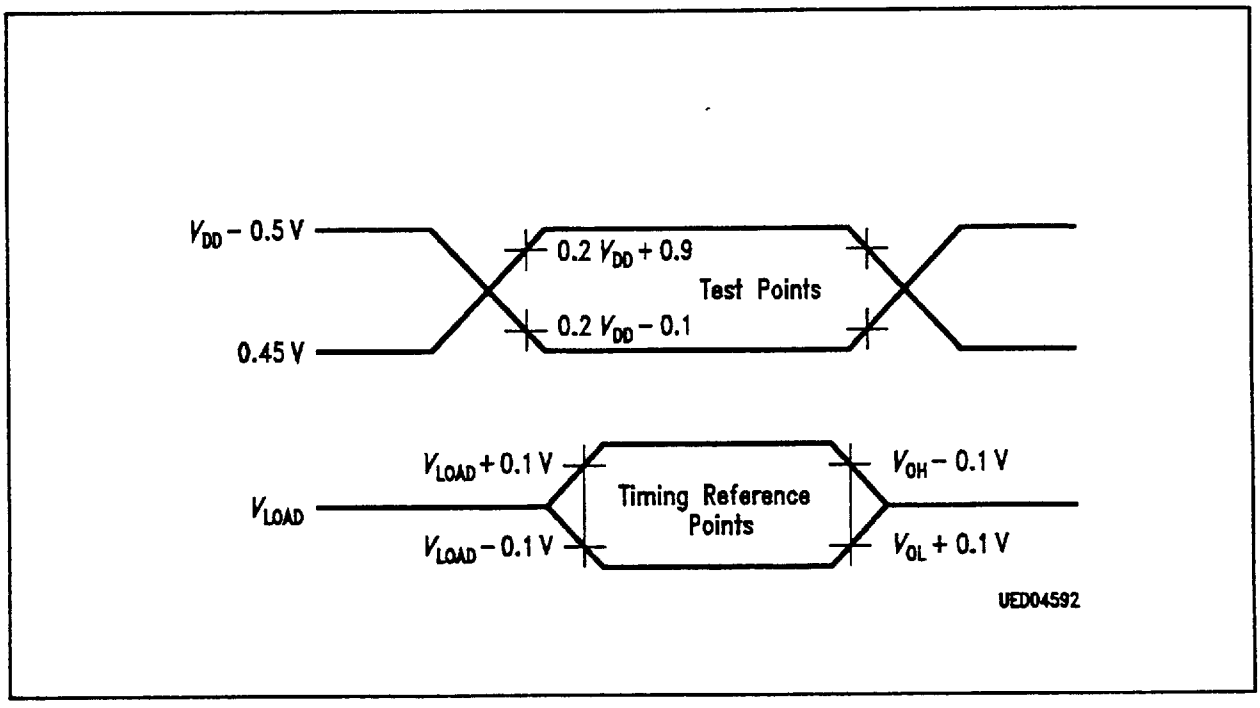


Figure 28
I/O-Waveform for AC-Tests

Waveforms

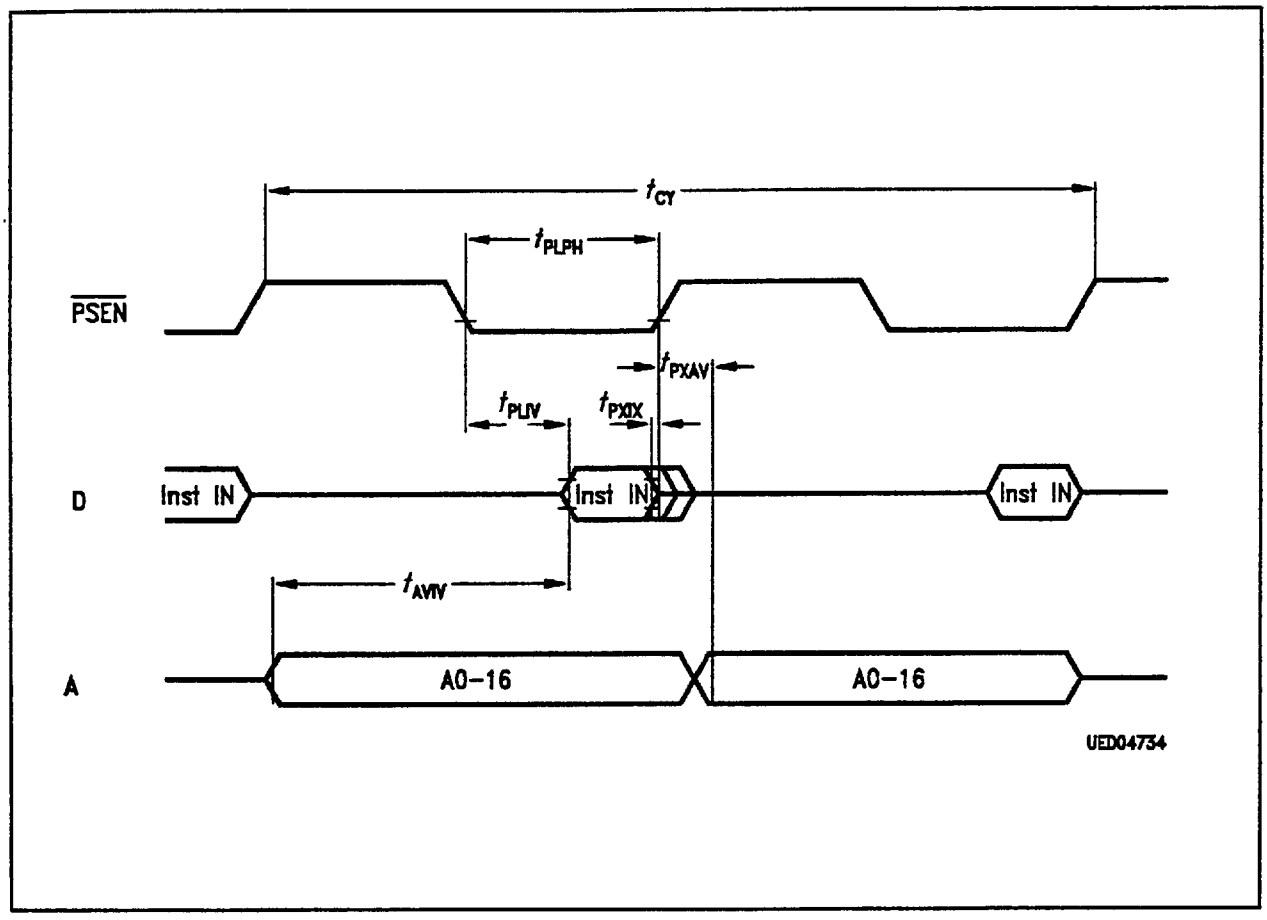


Figure 29
Program Memory Read Cycle

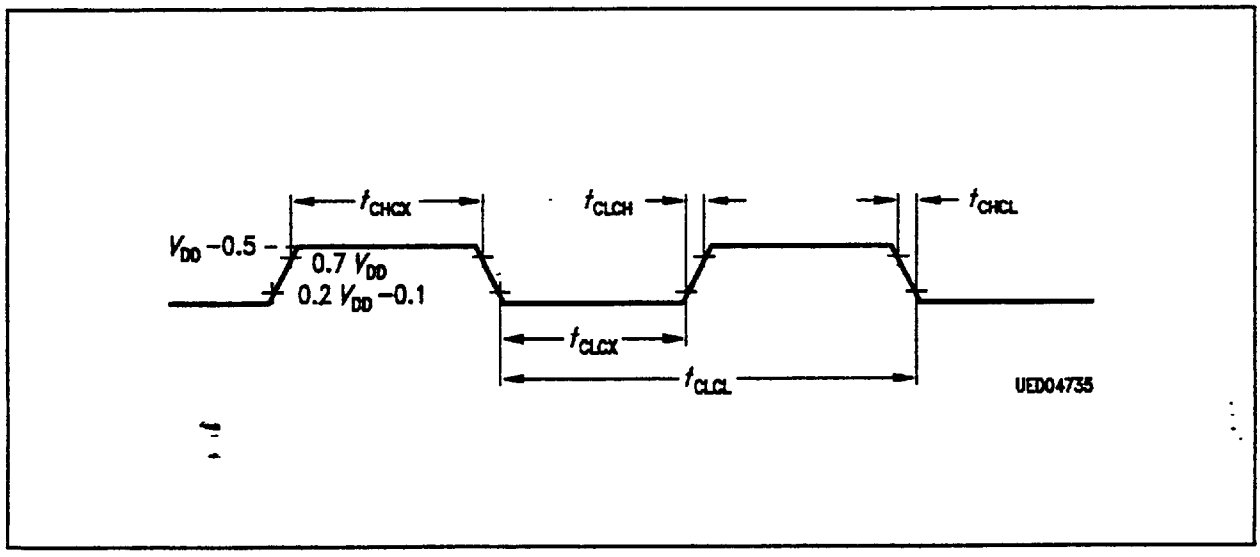


Figure 30
External Clock Cycle

4 Applications

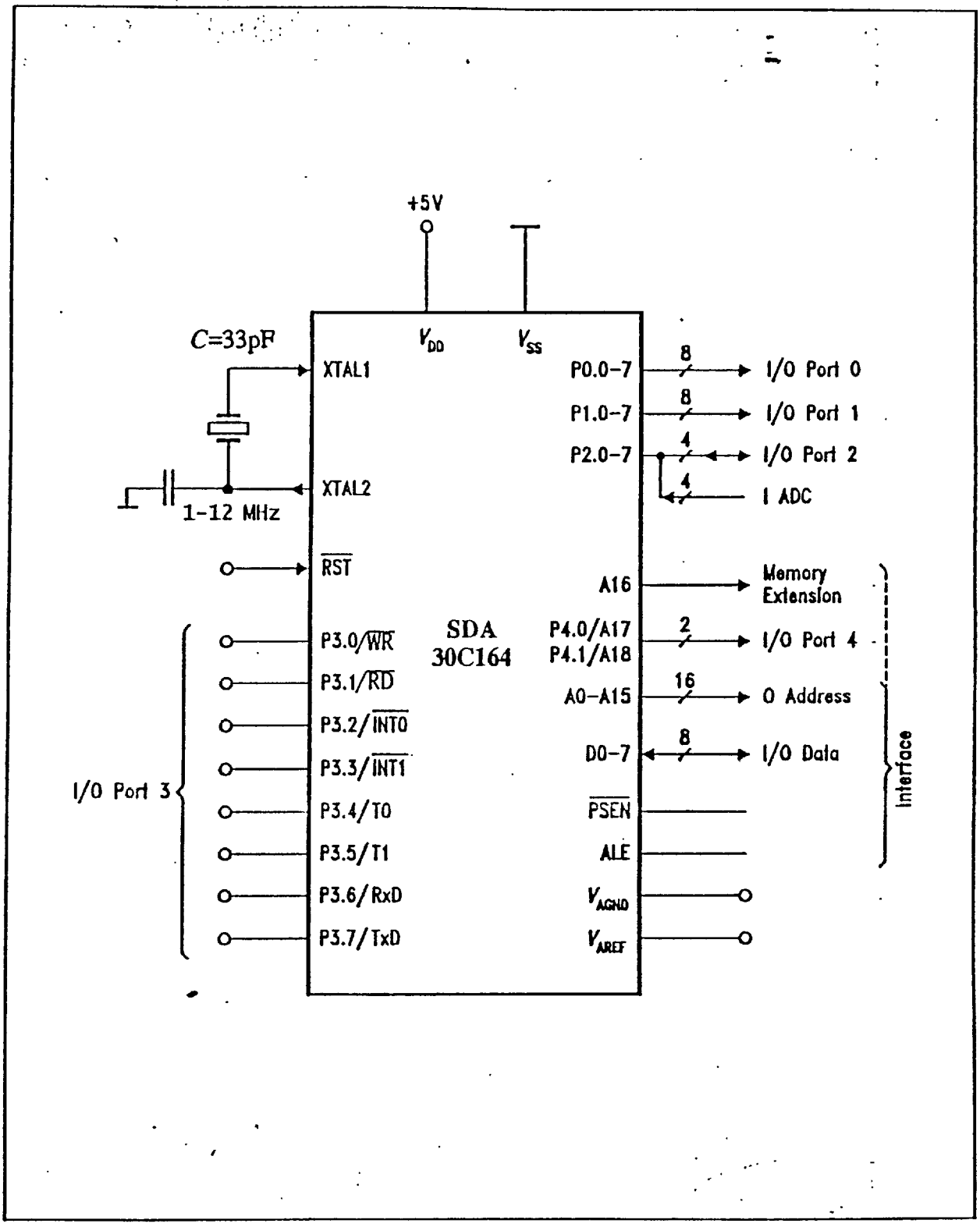


Figure 31
Application Circuit