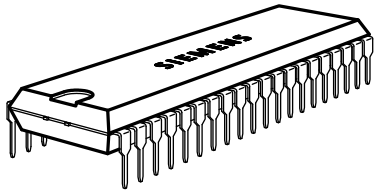
8-Bit Microcontroller with OSD

SDA 20Cxx50

Preliminary Data **CMOS IC**

1 Features



P-DIP-40

- **SAB 8051 Architecture**
  - On-chip oscillator and clock circuits
  - Binary or decimal arithmetic
  - Signed-overflow detection and parity computation
  - Integrated Boolean processor for control applications
  - Full depth stack for subroutine return linkage and data storage
  - Two priority level, nested interrupt structure

- **On-Screen Display Unit**
  - Up to six lines of 18 characters without software multiplexing
  - More than six lines when using software multiplexing
  - Interrupt generation by the OSD logic
  - 12 $\times$ 16 dot character matrix
  - 96 user defined characters (mask-programmable)
  - Two character sizes selectable individually for each line
  - Boxed or non-boxed mode selectable for each line
  - Character frame mode software selectable
  - One of eight colors selectable for each character
  - One of eight background colors selectable
  - LC oscillator for dot clock generation
  - Dot clock synchronization by sandcastle pulse
  - R/G/B/BLANK-outputs for colored text or symbols

- **On-Chip RAM**
  - Direct byte and bit addressability
  - Four register banks
  - Data memory with power down mode operation, including 128 user-defined software flags: 128 bytes for SDA 20C0850
    256 bytes for SDA 20C1650
    256 bytes for SDA 20C2450
    256 bytes for SDA 20C3250

– Data memory accessible with MOVX-instructions (XRAM):
0 bytes for SDA 20C0850
0 bytes for SDA 20C1650
128 bytes for SDA 20C2450
256 bytes for SDA 20C3250

- **On-Chip ROM**
  – Mask-programmable program memory:
  8192 bytes for SDA 20C0850
  16384 bytes for SDA 20C1650
  24576 bytes for SDA 20C2450
  32768 bytes for SDA 20C3250

- **26 Bidirectional I/O Lines**
  – One 8-bit port comprising up to eight programmable D/A outputs
  – One 8-bit multifunction port
  – One 8-bit port with open drain output for LED-driving
  – One 2-bit port with open drain output
  – Two additional input lines

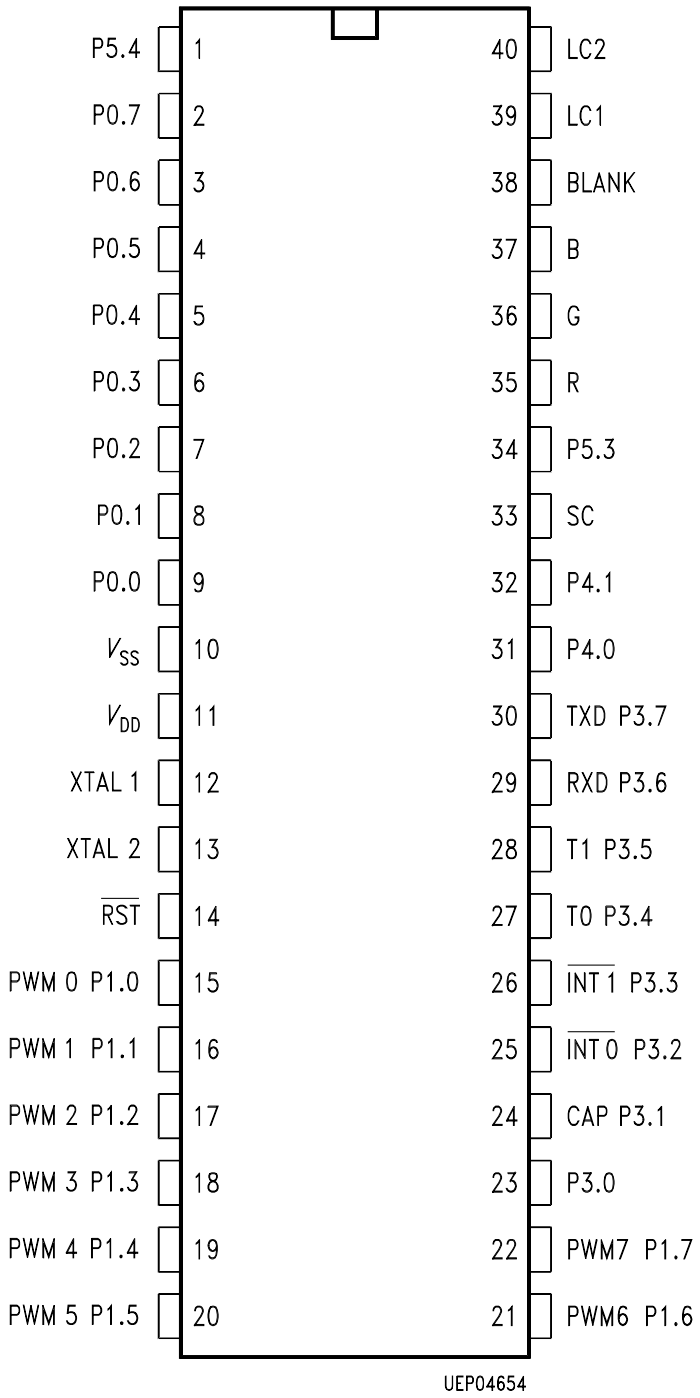
- **Pulse Width Modulation Unit**
  – Up to eight programmable PWM-output channels for low cost Digital-to-analog conversion (8-bit resolution)

- **Timers**
  – Two 16-bit general purpose timers/event counters
  – One 16-bit multi mode-timer with 2-bit prescaler and selectable watchdog, interrupt or capture function (not for SDA 20C0850).

- **Serial Interface**
  – Full duplex UART Interface

Type	Ordering Code	Package
SDA 20C0850	Q	P-DIP-40
SDA 20C1850	Q	P-DIP-40
SDA 20C2450	Q	P-DIP-40
SDA 20C3250	Q	P-DIP-40

Pin Configuration
(top view)



1.1 Pin Definitions and Functions

Pin No.	Symbol	Input (I) Output (O) Supply (S)	Function
1	P5.4	I	P5.4 is an input port only
2 3 4 5 6 7 8 9	P0.7 P0.6 P0.5 P0.4 P0.3 P0.2 P0.1 P0.0	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.
10	V_{SS}	S	Ground (0 V)
11	V_{DD}	S	Power supply voltage
12	XTAL1	I	Input to the inverting oscillator amplifier.
13	XTAL2	O	Output of the inverting oscillator amplifier. To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left open.
14	$\overline{\text{RST}}$	I	A low level on this pin resets the processor.
15 16 17 18 19 20 21 22	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)

Pin Definitions and Functions (cont'd)

Pin No.	Symbol	Input (I) Output (O) Supply (S)	Function
23 24 25 26 27 28 29 30	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 l s written to them are pulled high by the internal pullup resistors, and in that state can be used as inputs. It also contains the interrupt, timer, serial port and capture input pins. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. The secondary functions are assigned to the pins of port 3, as follows: – $\overline{\text{CAP}}$ (P3.1) : input for timer capture mode – $\overline{\text{INT0}}$ (P3.2) : interrupt 0 input/timer 0 gate control input – $\overline{\text{INT1}}$ (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
31 32	P4.0 P4.1	I/O I/O	Port 4 is a 2-bit bidirectional I/O port with no internal pullup. Port 4 pins that have 1 s written to them must be pulled high by an external pullup resistor, and in that state can be used as inputs.
33	SC	I	Sandcastle synchronization input
34	P5.3	I	P5.3 is an input port only
35	R	O	Red color signal output
36	G	O	Green color signal output
37	B	O	Blue color signal output
38	BLANK	O	Blanking output
39 40	LCIN LCOUT		LCIN and LCOUT are used to connect the external dot clock frequency reference.

1.2 Block Diagram

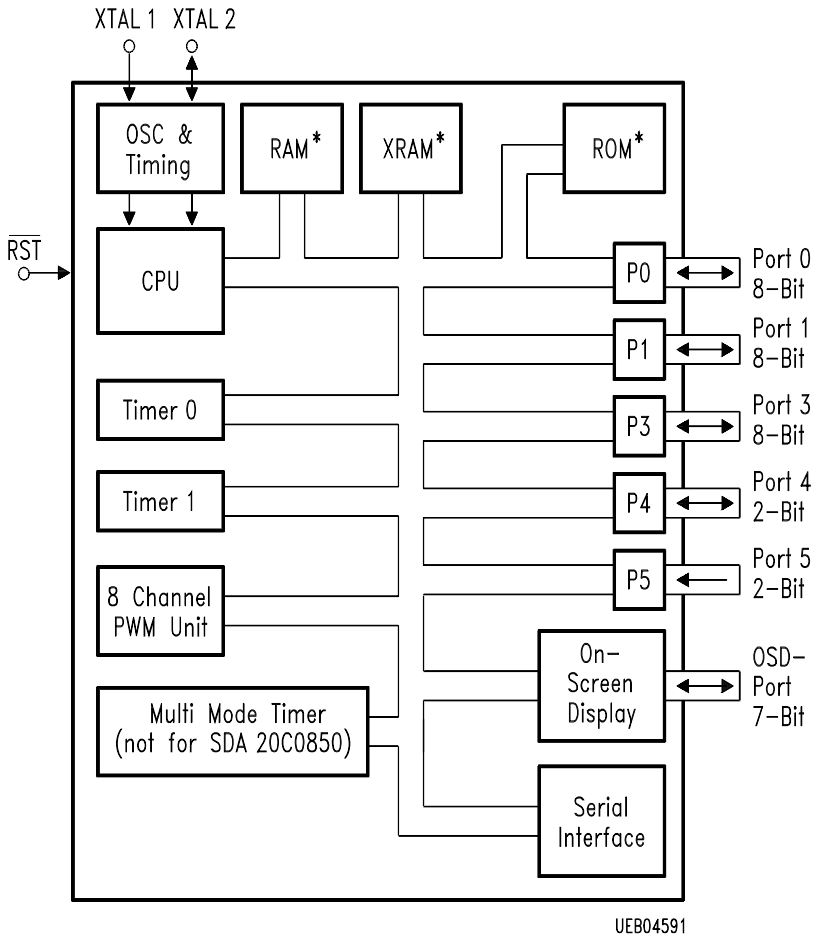


Figure 1
Block Diagram

RAM	XRAM	ROM	Chip
128 × 8	–	8 K × 8	SDA 20C0850
256 × 8	–	16 K × 8	SDA 20C1650
256 × 8	128 × 8	24 K × 8	SDA 20C2450
256 × 8	256 × 8	32 K × 8	SDA 20C3250

2 Functional Description

2.1 Architecture

The CPU manipulates operands in two memory spaces. These are the internal program memory and the internal data memory and extended data memory spaces. (No internal extended data memory in SDA 20C0850 and SDA 20C1650).

The internal data memory address space is divided into the 256-byte internal data RAM (128 byte for SDA 20C0850) 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 (SP). All registers except 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, timer, pulse width modulator, serial channel and OSD circuit. 128-bit locations in the SFR address space are addressable as bits. The extended data Memory is addressed by MOVX-instructions and addressed by the data pointer register DPTR.

(No internal extended data memory in SDA 20C0850 and SDA 20C1650).

Conditional branches are performed relative to the 16-bit 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 in the internal memory address space.

The processor has 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

The functional block diagram displayed in **figure 2** shows the hardware architecture of the CPU in detail.

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 Program Memory

The controller contains 8/16/24/32 Kbyte of internal program memory for SDA 20C08/16/24/32 resident on-chip. The memory is mask-programmable.

Internal Data RAM

The internal data RAM provides a 256-byte scratch pad memory (128 byte for SDA 20C0850), 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.

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 controls 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	$18_H – 1F_H$

MSB	SFR Address: $D0_H$						LSB
PSW: Program Status Word							
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 from 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. (128 bytes for SDA 20C0850).

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.

Port 0, Port 1, Port 3, Port 4, Port 5

These four ports provide 26 I/O lines to interface to the external world. These ports are both byte and bit addressable. Port 0 is used for binary l/O and offers eight open drain output lines. Port 1 provides up to eight PWM output channels as alternate functions while port 3 contains special control signals. Port 4 is a two bit open drain l/O port. P5 contains two additional input lines.

Interrupt Logic

Controlled by five special function registers (TCON, IE, IP, IFR, SCON) the interrupt logic provides six interrupt vectors. Each of them may be assigned to high or low priority (see chapter “Interrupt System”).

Timer/Counter 0/1

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

Multi Mode Timer (not for SDA 20C0850)

A 16-bit timer with a programmable 2-bit prescaler may be used as watchdog, capture or general purpose timer. It is controlled by the special function registers WDMOD, WDTC, WDCL, WDCH, WDRL, WDRH, CAPL and CAPH (see chapter “Multi Mode Timer”).

Pulse Width Modulation Unit

Up to eight lines of port 1 may be used as 8-bit PWM-outputs. The PWM logic is controlled by registers PWME, PWMC, PWCOUNT, PWCOMP0...7 (see chapter “Pulse Width Modulation Unit”).

On Screen Display Unit

Text and graphics to be inserted into a video picture are generated by the on-screen display unit. Control registers DER, DHR, DVR0...5, DSCR, DDR, DAR, DCPR are described in chapter “On-Screen Display”.

Serial Interface

To communicate with other UART (universal asynchronous receiver/transmitter) devices and for I/O-expansion, a full duplex serial interface has been provided. It consists of a serial port receive line RxD and a serial port transmit line TxD and can be programmed to function in different operating modes (see chapter “Serial Interface”).

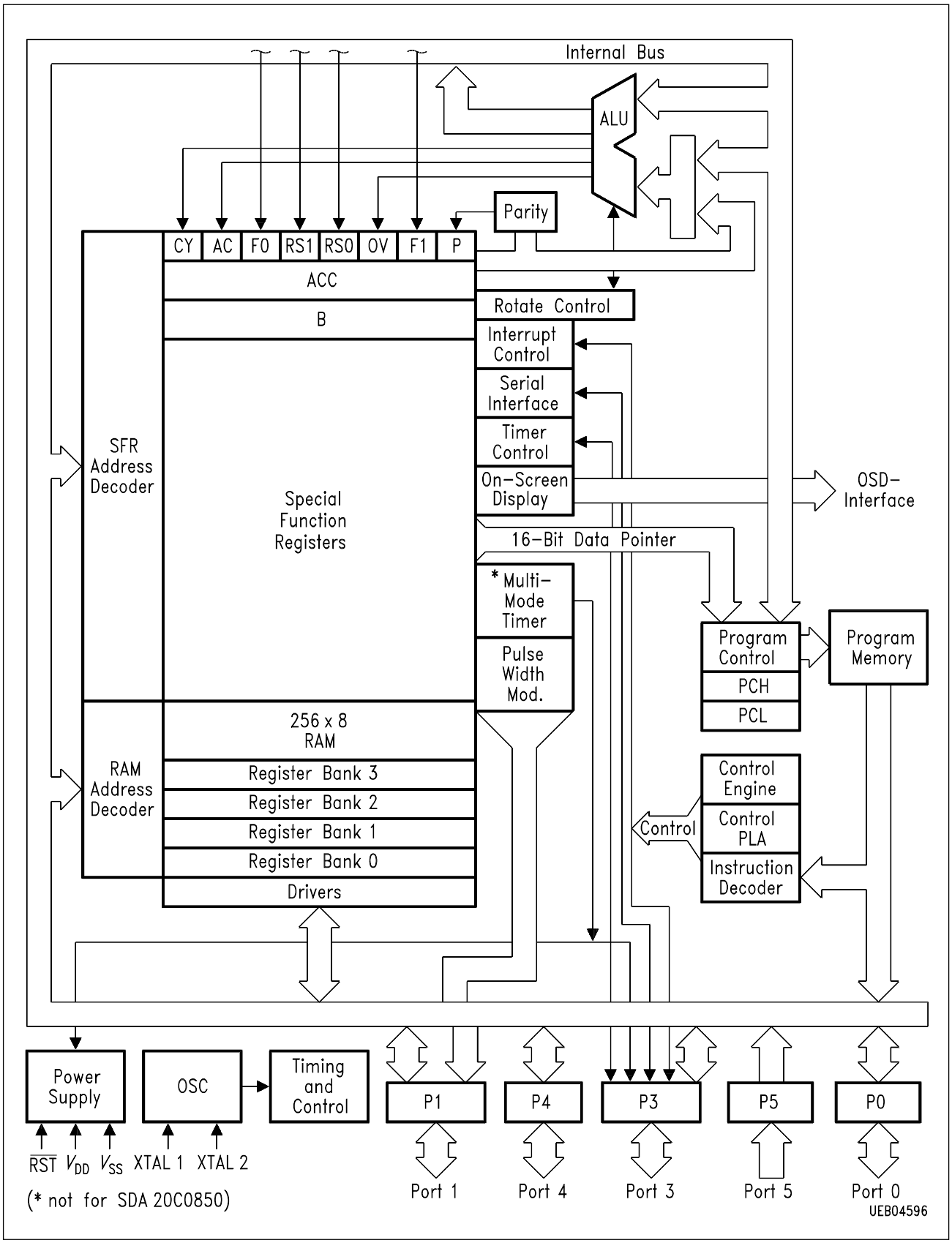


Figure 2
Functional Block Diagram

2.1.2 CPU Timing

The timing generation is completely self-contained, except for the frequency reference which can be a crystal or external clock source.

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 two address spaces. The memory spaces are:

– Internal program memory space (8/16/24/32 Kbyte for SDA 20C08/16/24/32)
– 256 byte (only 128 byte for SDA 20C08) plus 128 byte internal data memory address space
– Additional internal data memory (128 byte for SDA 20C2450, 256 byte for SDA 20C3250).

A 16-bit program counter provides the processor with its 64-Kbyte addressing capabilities.
The program counter allows the user to execute calls and branches to any location within the program memory space. There are no instructions that permit program execution to move from the program memory space to any of the data memory space.

2.2.1 Internal Program ROM

Certain locations in program memory are reserved for specific programs. Locations 0000 through 0002 are reserved for the initialization program. Following reset, the CPU always begins execution at location 0000. Locations 0003 through 0043 are reserved for the six interrupt-request service programs as indicated in the following table:

Source	Address
External Interrupt 0	03 (03_H)
Timer 0 Overflow	11 ($0B_H$)
External Interrupt 1 or On-Screen Display	19 (13_H)
Timer 1 Overflow	27 ($1B_H$)
Serial Interface	35 (23_H)
Pulse Width Modulator or Multi Mode Timer	43 ($2B_H$)

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 (not for SDA 20C0850), the 128-byte Special Function Register (SFR) area and for SDA 20C2450/20C3250 the 128/256 byte additional RAM (**figure 3**). 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 4**. 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.

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 1**.

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 5**.

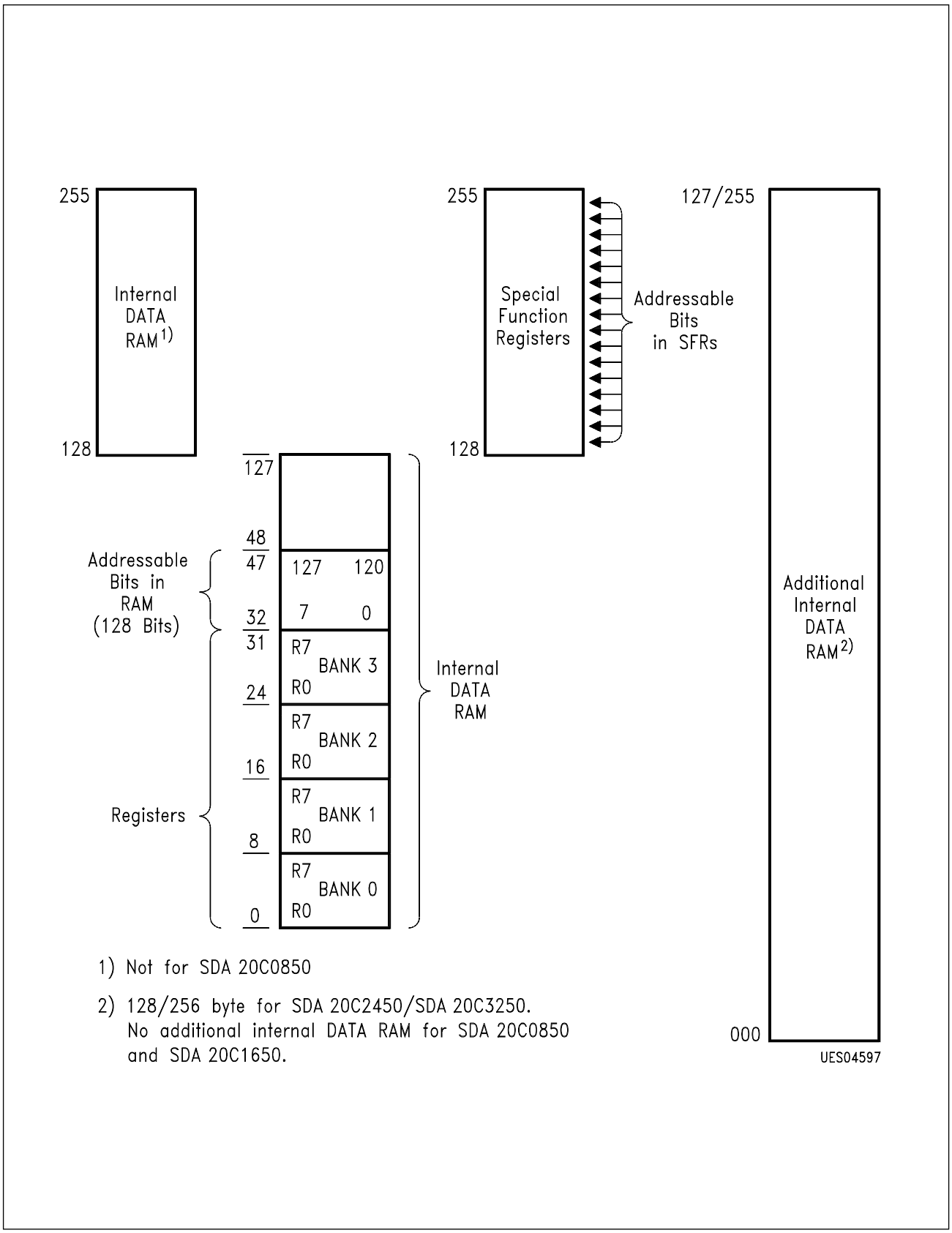


Figure 3
Internal Data Memory Address Space

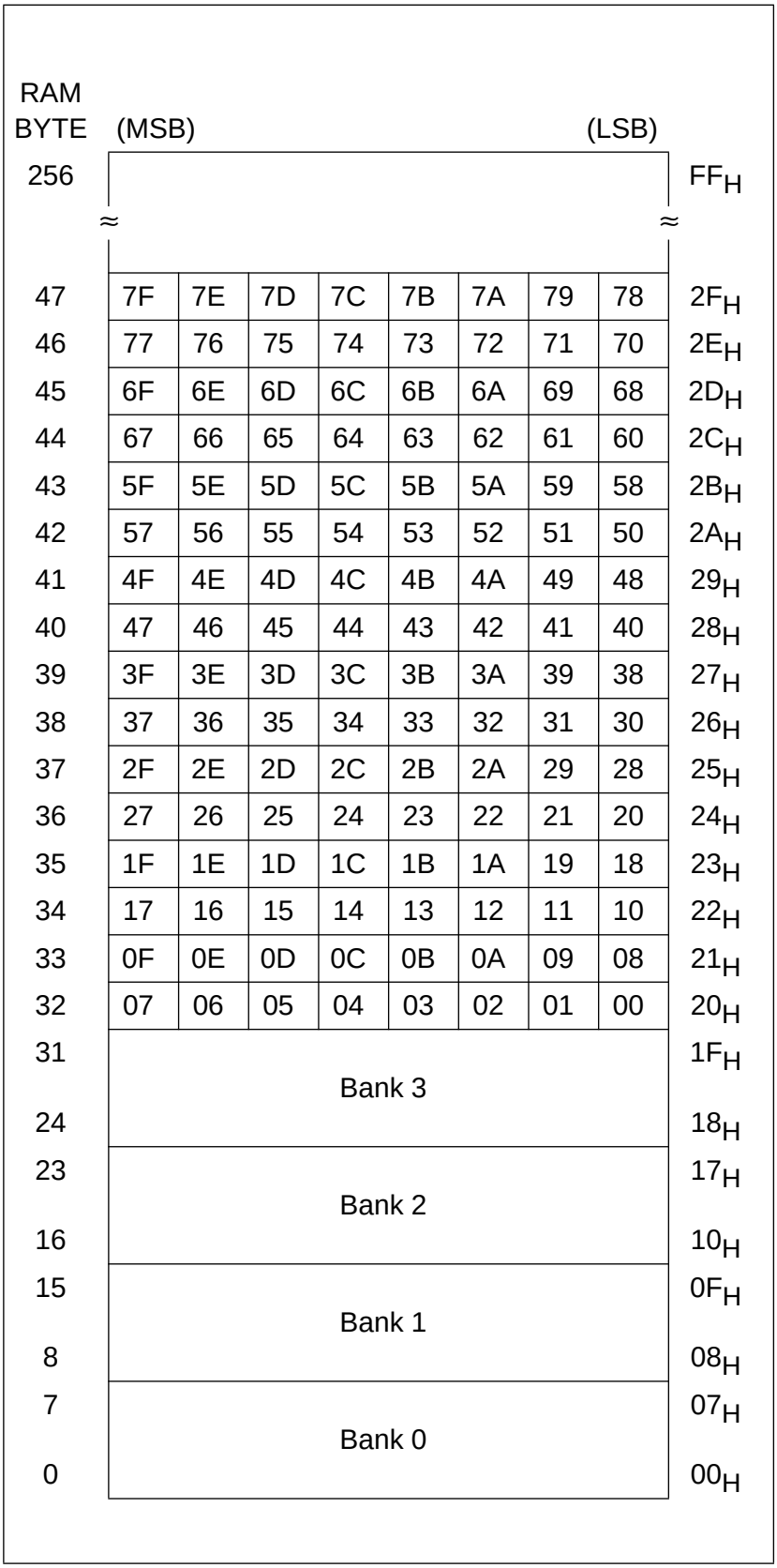


Figure 4
Internal RAM-Bit Addresses

Direct Byte Address (decimal)	Addressable SFR Bits								
248	P7	P6	P5	P4	P3	P2	P1	P0	PWCOMP7
240	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0	B
232							P4.1	P4.0	P4
224	A.7	A.6	A.5	A.4	A.3	A.2	A.1	A.0	ACC
216	1	HSY	VSY	EMU*	PRE	R	G	B	DSCR
208	CY	AC	F0	RS1	RS0	OV	F1	P	PSW
200	S	M1	M0	R	IR	C2	C1	C0	PWMC
192	E7	E6	E5	E4	E3	E2	E1	E0	PWME
184			PPW	PS	PT1	PX1	PT0	PX0	IP
176	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	P3
168	EA		EPW	ES	ET1	EX1	ET0	EX0	IE
152	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	SCON
144	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	P1
136	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	TCON
128	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	P0

* EMU bit is 0 for SDA 20Cxx50 and 1 for SDA 30C0050

Figure 5
Special Function Register Bit Addresses

Table 1
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.)
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
Pointer Registers					
Stack Pointer	SP	81	129	–	07
Data Pointer (high byte)	DPH	83	131	–	00
Data Pointer (low byte)	DPL	82	130	–	00
Power Control Register	PCON	87	135	–	00
I/O Port Registers					
Port 0	P0	80	128	87 – 80	FF
Port 1	P1	90	144	97 – 90	FF
Port 3	P3	B0	176	B7 – B0	FF
Port 4	P4	E8	232	E9 – E8	FF
Interrupt Control Registers					
Interrupt Priority Flags	IP	B8	184	BF – B8	XX000000 (bin.)
Interrupt Enable Flags	IE	A8	168	AF – A8	0X000000 (bin.)
Interrupt Flag Register	IFR	FA	250	–	00
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
Multi-Mode Timer Registers *)					
Timer Status Register	WDMOD	86	134	–	01
Timer Clear Register	WDTC	97	135	–	FF
Counter Register (low byte)	WDCL	84	132	–	FF
Counter Register (high byte)	WDCH	85	133	–	FF
Reload Register (low byte)	WDRL	8E	142	–	FF
Reload Register (high byte)	WDRH	8F	143	–	FF
Capture Register (low byte)	CAPL	91	145	–	00
Capture Register (high byte)	CAPH	92	146	–	00

X: undefined bit value

*) not for SDA 20C0850

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.)
Serial Interface					
Serial Control Register	SCON	98	152	9F – 98	00
Serial Interface Buffer Register	SBUF	99	153	–	00
Pulse Width Modulator Registers					
Control Register	PWMC	C8	200	CF – C8	80
Enable Register	PWME	C0	192	C7 – C0	00
PWM8-Counter Register	PWCOUNT	F9	249	–	00
Compare Register 0	PWCOMP0	F1	241	–	FF
Compare Register 1	PWCOMP1	F2	242	–	FF
Compare Register 2	PWCOMP2	F3	243	–	FF
Compare Register 3	PWCOMP3	F4	244	–	FF
Compare Register 4	PWCOMP4	F5	245	–	FF
Compare Register 5	PWCOMP5	F6	246	–	FF
Compare Register 6	PWCOMP6	F7	247	–	FF
Compare Register 7	PWCOMP7	F8	248	FF – F8	FF
On-Screen Display Registers					
Enable Register	DER	D1	209	–	00
Vertical Position of Row 0	DVR0	D2	210	–	XX
Vertical Position of Row 1	DVR1	D3	211	–	XX
Vertical Position of Row 2	DVR2	D4	212	–	XX
Vertical Position of Row 3	DVR3	D5	213	–	XX
Vertical Position of Row 4	DVR4	D6	214	–	XX
Vertical Position of Row 5	DVR5	D7	215	–	XX
Status and Control	DSCR	D8	216	DF – D8	XX
Address Register	DAR	D9	217	–	XX
Data Register	DDR	DA	218	–	XX
Horizontal Position	DHR	DB	219	–	XX
Color and Polarity Register	DCPR	DC	220	–	00
Accumulator Extension Register	ACEXT	E1	225	–	XX

X: undefined bit value

2.3 Interrupt System

External events and the real-time on-chip peripherals require CPU service asynchronous to the execution of any particular section of code. To couple the asynchronous activities of these functions to normal program execution, a sophisticated multiple-source, two-priority-level, nested interrupt system is provided.

2.3.1 Interrupt Sources

The processor acknowledges interrupt requests from eight sources, which are mapped on six vectors. One vector for external interrupt 0, one vector for external interrupt 1 or OSD interrupt, two vectors for timer 0 and timer 1, one vector for the serial interface and one vector for the multi-mode timer and PWM interrupts. Each of the six vectors can be assigned to either of two priority levels and can independently be enabled or disabled. Additionally, all enabled interrupts may be globally enabled or disabled.

Interrupts result in a transfer of control to a new program location. An interrupt vectors to a special 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 or On-Screen Display	19	(13_H)
Internal Timer/Counter 1	27	($1B_H$)
Serial Interface	35	(23_H)
Pulse Width Modulator or Multi-Mode Timer	43	($2B_H$)

2.3.2 Interrupt Control

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

TCON	Timer/Counter Control Register	88_H
IE	Interrupt Enable Register	$A8_H$
IP	Interrupt Priority Register	$B8_H$
IFR	Interrupt Flag Register	FA_H
SCON	Serial Control Register	98_H

All these registers except for IFR contain direct addressable bits.

The interrupt system is shown diagrammatically in **figure 6**.

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

Interrupt Source	Request Flag	Bit Location
External Request 0	IE0	TCON.1
Internal Timer/Counter 0	TF0	TCON.5
External Request 1 or On-Screen Display	IE1	TCON.3
Internal Timer/Counter 1	TF1	TCON.7
Serial Interface Receive	RI	SCON.0
Serial Interface Transmit	TI	SCON.1
Pulse Width Modulator or	IPWM	IFR.0
Multi-Mode Timer *)	IWDT	IFR.1

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

*) not for SDA 20C0850

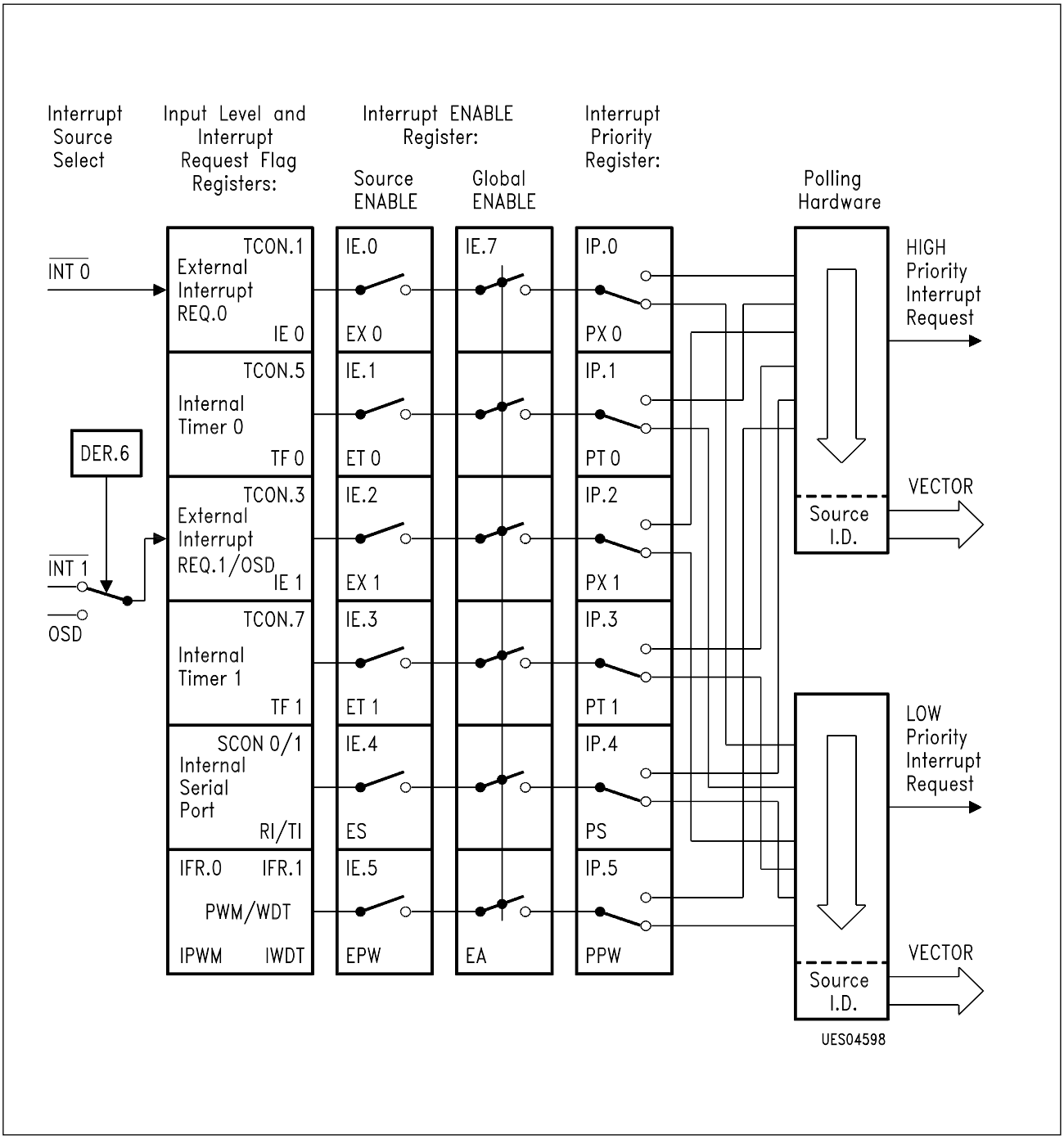


Figure 6
Interrupt System

- Six interrupt vectors
- Each interrupt can be individually enabled/disabled
- Enabled interrupts can be globally enabled/disabled
- Each interrupt can be assigned to either of two priority levels
- Each interrupt vectors to a separate location in program memory
- Interrupt nesting to two levels
- External interrupt requests can be programmed to be level- or transition-activated

The PWM/multi-mode timer interrupt is generated by the logical OR of IPWM and IWDT. To decide, which source (PWM or multi-mode timer) has requested the interrupt, the interrupt service routine located at address $02B_H$ has to check the bits IPWM (IFR.0) and IWDT (IFR.1). These bits are usually set by hardware but they can also be set by software. If a PWM/multi-mode timer interrupt is generated, the flags IPWM and IWDT will not be cleared by hardware, they will have to be cleared by software.The interrupt request will be acknowledged, if its interrupt enable bit in the Interrupt Enable register (IE) is set, and will be serviced according to the selected priority level in the Interrupt Priority register (IP).

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

MSB	SFR Address: FA_H						LSB
IFR: Interrupt Flag Register							
P5.3	P5.4	–	–	–	–	IWDT	IPWM

Default after reset:		00_H
IPWM	–	IPWM = 1 Interrupt request was generated by the PWM unit
IWDT *)	–	IWDT = 1 Interrupt request was generated by the multi-mode-timer
P5.3		Input Line P5.3
P5.4		Input Line P5.4
IFR.2 … IFR.5:		Reserved

Figure 7
Interrupt Flag Register IFR

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

*) not for SDA 20C0850

MSB	SFR Address: $A8_H$						LSB
IE: Interrupt Enable Register							
EA	–	EPW	ES	ET1	EX1	ET0	EX0

Default after reset:	$0X00\ 0000_B$ X: undefined bit value
EA	Enables or disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
IE.6	Reserved
EPW	Enables or disables the interrupt from the pulse width modulation unit and from the multi-mode-timer. If EPW = 0, this interrupt is disabled.
ES	Enables or disables the serial interface interrupt. If ES = 0, the serial interface interrupt is disabled.
ET1	Enables or disables the timer 1 overflow interrupt. If ET1 = 0, the timer 1 interrupt is disabled.
EX1	Enables or disables external interrupt 1 or OSD interrupt. If EX1 = 0, external interrupt 1/OSD interrupt is disabled.
ET0	Enables or disables the timer 0 overflow interrupt. If ET0 = 0, the timer 0 interrupt is disabled.
EX0	Enables or disables external interrupt 0. If EX0 = 0, external interrupt 0 is disabled.

Figure 8
Interrupt Enable Register IE

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

MSB	SFR Address: $B8_H$						LSB
IP: Interrupt Priority Register							
–	–	PPW	PS	PT1	PX1	PT0	PX0

Default after reset:	XX00 0000 X: undefined bit value
IP.7	Reserved
IP.6	Reserved
PPW	Defines the interrupt from the pulse width modulation unit or the multi-mode-timer. PPW = 1 programs it to the higher priority level.
PS	Defines the serial interface interrupt priority level. PS = 1 programs it to the higher priority level.
PT1	Defines the timer 1 interrupt priority level. PT1 = 1 programs it to the higher priority level.
PX1	Defines the external interrupt 1/OSD interrupt priority level. PX1 = 1 programs it to the higher priority level.
PT0	Defines the timer 0 interrupt priority level. PT0 = 1 programs it to the higher priority level.
PX0	Defines the external interrupt 0 priority level. PX0 = 1 programs it to the higher priority level.

Figure 9
Interrupt Enable Register IP

If two requests of different priority levels are received simultaneously, the request of higher priority level will be serviced. If requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence, as follows:

Source	Priority within Level
1. IE0	(highest)
2. TF0	
3. IE1/OSD	
4. TF1	
5. RI/TI	
6. IPWM/IWDT	(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 reenabled while further low-priority interrupts remain disabled.

2.3.4 External Interrupts

The external interrupt request inputs (INT0 and INT1) can be programmed for either transition-activated or level-activated operation. Control of the external interrupts is provided by the four low-order bits of TCON as shown in **figure 10**.

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

MSB	SFR Address: 88_H						LSB
TCON: Timer and Interrupt Control Register							
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Default after reset:	00_H
TCON.4 – TCON.7	See chapter “General Purpose Timers/Counters”
IE1	Interrupt 1 edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
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 10
Function of Lower Nibble Bits in TCON

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

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

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

– Level-Activated Interrupts (IT = 0, IT1 = 0)

The IE0, IE1 flags are set whenever INT0, INT1 are respectively sampled at low level. Sampling INT0, INT1 at high level clears IE0, IE1, respectively.

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

2.3.5 Interrupt Task Function

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

The sequence of events for an interrupt is:

– A source provokes an interrupt by setting its associated interrupt request bit to let the processor know an interrupt condition has occurred.

– The CPU's internal hardware latches the internal request in the tenth, twenty-second, thirty-fourth and forty-sixth oscillator period of the instruction in progress.

– The interrupt request is conditioned by bits in the interrupt enable and interrupt priority register.

– The processor acknowledges the interrupt by setting one of the two internal “priority-level active” flip-flops and performing a hardware subroutine call. This call pushes the PC (but not the PSW) onto the stack and, for most sources, clears the interrupt request flag.

– The service program is executed.

– Control is returned to the main program when the RETI instruction is executed. The RETI instruction also clears one of the internal “priority-level active” flip-flops.

Most interrupt request flags (IE0, IE1, TF0 and TF1) are cleared when the processor transfers control to the first instruction of the interrupt service program. The RI, TI, IPWM and IWDT interrupt request flags are exceptions and must be cleared as part of the respective interrupt service program. This is also the case for IE0, IE1, if INT0, INT1 are level activated.

2.3.6 Response Time

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

If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two cycles. Thus, a minimum of three complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine. lf the instruction in progress is not in its final cycle, the additional wait time cannot be more than 3 cycles, since the longest instructions (MUL and DIV) are only 4 cycles long, and if the instruction in progress is RETI or an access to IE or IP, the additional wait time cannot be more than 5 cycles (a maximum of one more cycle to complete the instruction in progress, plus 4 cycles to

complete the next instruction if the instruction is MUL or DIV). Thus, in a single-interrupt system, the response time is always more than 3 cycles and less than 8 cycles (approximately 7 µs at 12-MHz operation). Examples of the best and worst case conditions are illustrated in the following table.

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

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

2.4 Processor Reset and Initialization

Processor initialization is accomplished with activation of the RST pin, which is the input to a Schmitt Trigger. To reset the processor, this pin should be held low for at least four µs, while the oscillator is running. Upon powering up, RST should be held low for at least 10 ms after the power supply stabilizes to allow the oscillator to stabilize. 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). **Table 1** (in chapter "Memory Organization") shows the values to be read after the end of the initialization sequence.

After the processor is reset, all port latches are written with ones. Outputs are undefined until the reset period is complete.

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.

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.

MSB	SFR Address: 87_H						LSB
PCON: Power Control Register							
SMOD	PDS	IDLS	–	–	–	PDE	IDLE

Default after reset:	000xxx00
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.

2.5 Ports and I/O Pins

There are 26 I/O pins configured as three 8-bit ports and one 2-bit port. Each pin can be individually and independently programmed as input or output and each can be configured dynamically (i.e., on-the-fly) under software control. Moreover, two input-only lines are provided.

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 “ones” 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 (**figure 11**).

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.

Port 4.0 and 4.1 have also an open-drain configuration (**see figure 12**) and have to be pulled up by external resistors.

Ports 1 and 3 have “quasi-bidirectional” output drivers which comprise an internal pullup resistor of 10 kΩ to 40 kΩ as shown in **figure 13**. When configured as inputs they pull high and will source current when externally pulled low.

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

The port output drivers can sink or source the following TTL loads:

Port	LS TTL load
0	8 (only sink)
1	4
3	4
4	4 (only sink)

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

P3.1	CAP	(multi-mode timer capture input)
P3.2	$\overline{\text{INT0}}$	(external Interrupt 0)
P3.3	$\overline{\text{INT1}}$	(external Interrupt 1)
P3.4	T0	(timer/counter 0 external input)
P3.5	T1	(timer/counter 1 external input)
P3.6	RXD	(serial interface receive line)
P3.7	TXD	(serial interface transmit line)

Two read-only input lines P5.3 and P5.4 are available, the input status is read through special function register bits IFR.7 and IFR.6 (not bit-addressable).

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 2**.

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 2

Read-Modify-Write Instructions

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

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

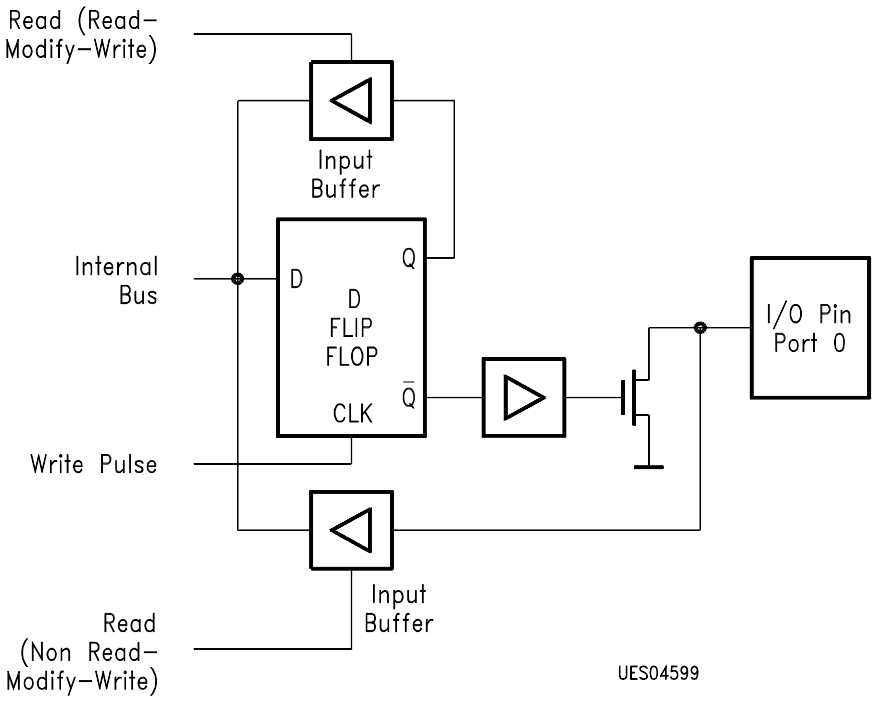


Figure 11
Port 0: Bidirectional Open Drain Bus Configuration

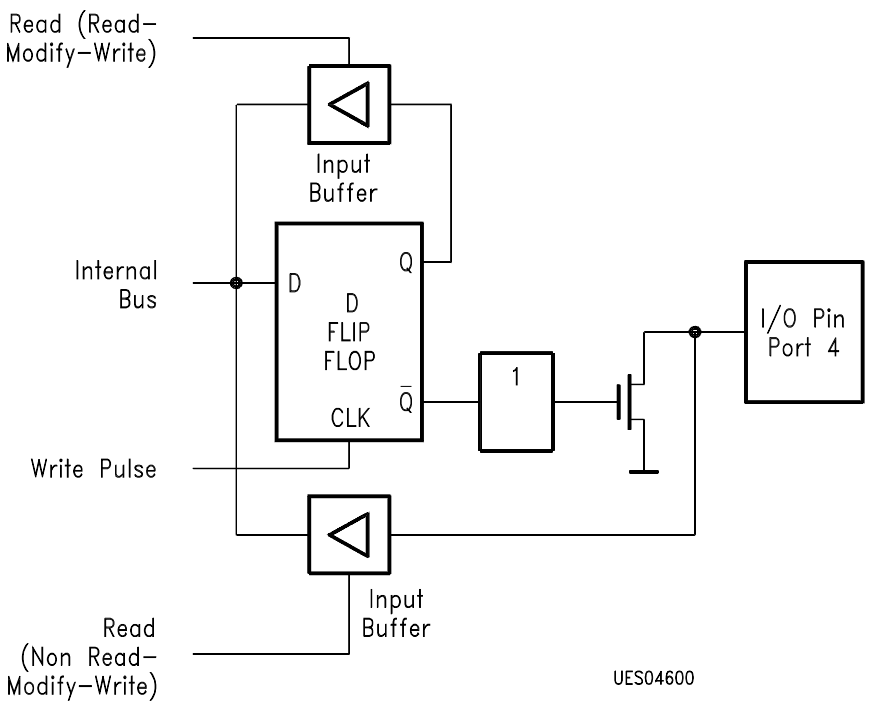


Figure 12
Port 4: Open Drain Configuration

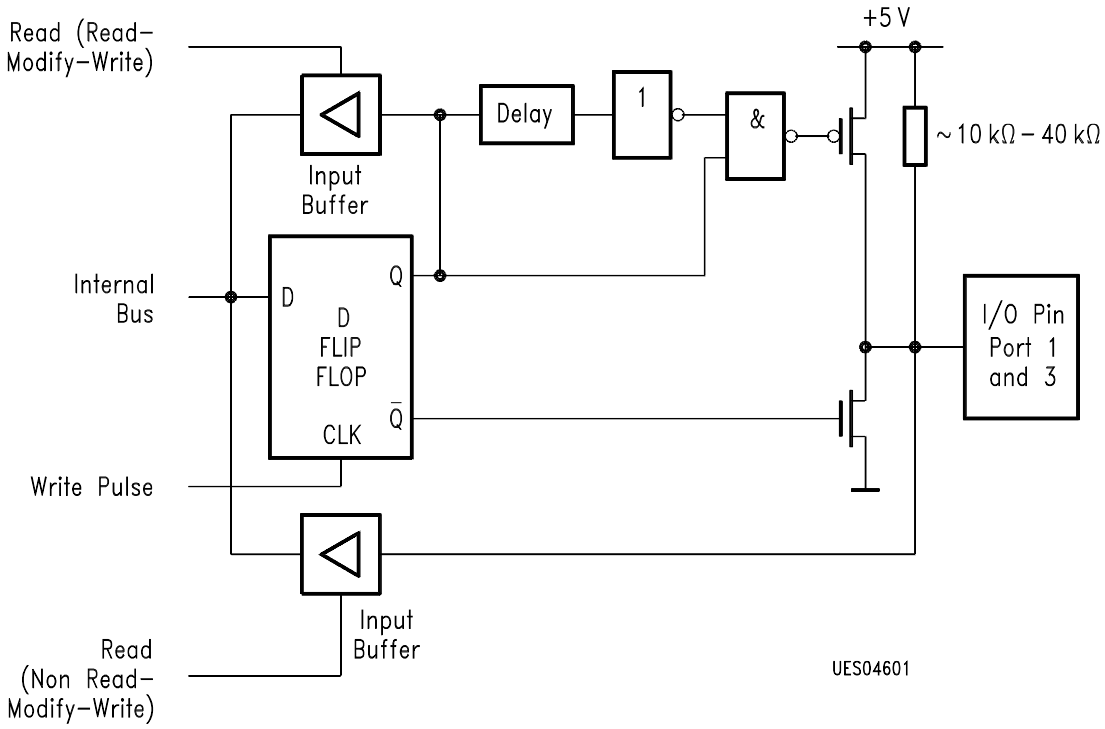


Figure 13
Ports 1 and 3: “Quasi-Bidirectional” Port Structure

2.6 General Purpose Timers/Counters

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

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

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

Timer/Counter 0: Mode Selection

Timer/counter 0 can be configured in one of four operating modes, which are selected by bit-pairs (M 1, M0) in TMOD register (**figure 14**).

– 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 16** shows the mode 0 operation as it applies to timer 0.

In this mode, the timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the timer interrupt flag TF0. The counted input is enabled to the Timer when TR0 = 1 and either GATE = 0 or INT0 = 1. (Setting GATE = 1 allows the Timer to be controlled by external input INT0, to facilitate pulse width measurements.) TR0 is a control bit in the special function register TCON (**figure 15**). GATE is contained in register TMOD (**figure 14**).

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 17**). 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 18**). TL0 uses the timer 0 control bits: C/T, GATE, TR0, INT0 and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from timer 1. Thus, TH0 now controls the “Timer 1” interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. With timer 0 in mode 3, the processor can operate as if it has three timers/counters. When timer 0 is in mode 3, timer 1 can be turned on and off by switching it out of and into its own mode 3, or can still be used in any application not requiring an interrupt.

Timer/Counter 1: Mode Selection

Timer/counter 1 can also be configured in one of four modes, which are selected by its own bit-pairs (M 1, M0) in TM0D 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 14** and **figure 15**. The input to the counter circuitry is from an external reference (for use as a counter), or from the on-chip oscillator (for use as a timer), depending on whether TMOD's C/T bit is set or cleared, respectively. When used as a time base, the on-chip oscillator frequency is divided by twelve (12) before being used as the counter input. When TMOD's gate bit is set (1), the external reference input (T1, T0) or the oscillator input is gated to the counter conditional upon a second external input (INT0), (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 INT0 and INT1 is not affected by the counter's operation. If enabled, an interrupt will occur when the input at INT0 or INT1 is low. The counters are enabled for incrementing when TCON's TR1- and TR0 bits are set. When the counters overflow, the TF1-and TF0 bits in TCON get set, and interrupt requests are generated.

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

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

MSB	SFR Address: 89_H						LSB
TMOD: Timer 0/1 Mode Register							
GATE	C/T	M1	M0	GATE	C/T	M1	M0

Timer 1 Timer 0

GATE	Gating control when set. Time/counter “x” is enabled only while “$\overline{\text{INT}}$x” 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).

M0	M1	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.

Default after reset: 00_H

Figure 14
Timer/Counter Mode Register

MSB	SFR Address: 88_H						LSB
TCON: Timer 0/1 Control Register							
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	Timer 1 overflow flag. Set by hardware on timer/counter owerflow. 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 owerflow. 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.
IE 1	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.
Default after reset:	00_H

Figure 15
Timer/Counter Control Register

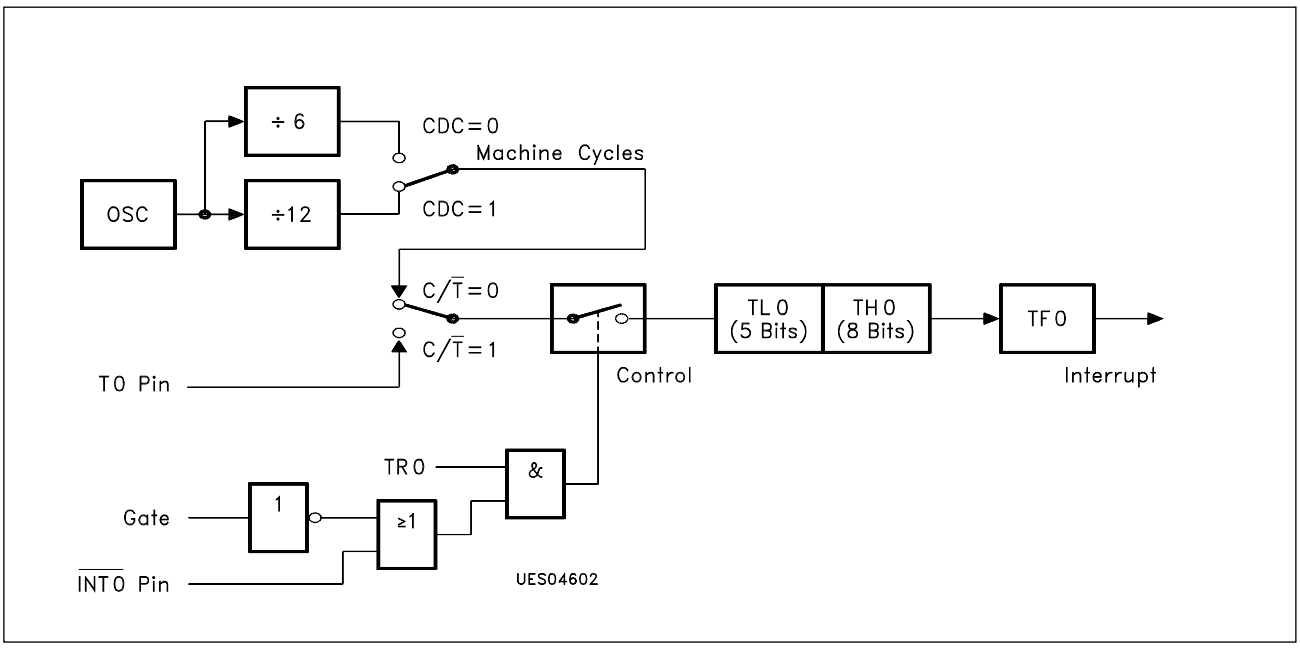


Figure 16
Timer/Counter 0 Mode 0: 13-Bit Counter

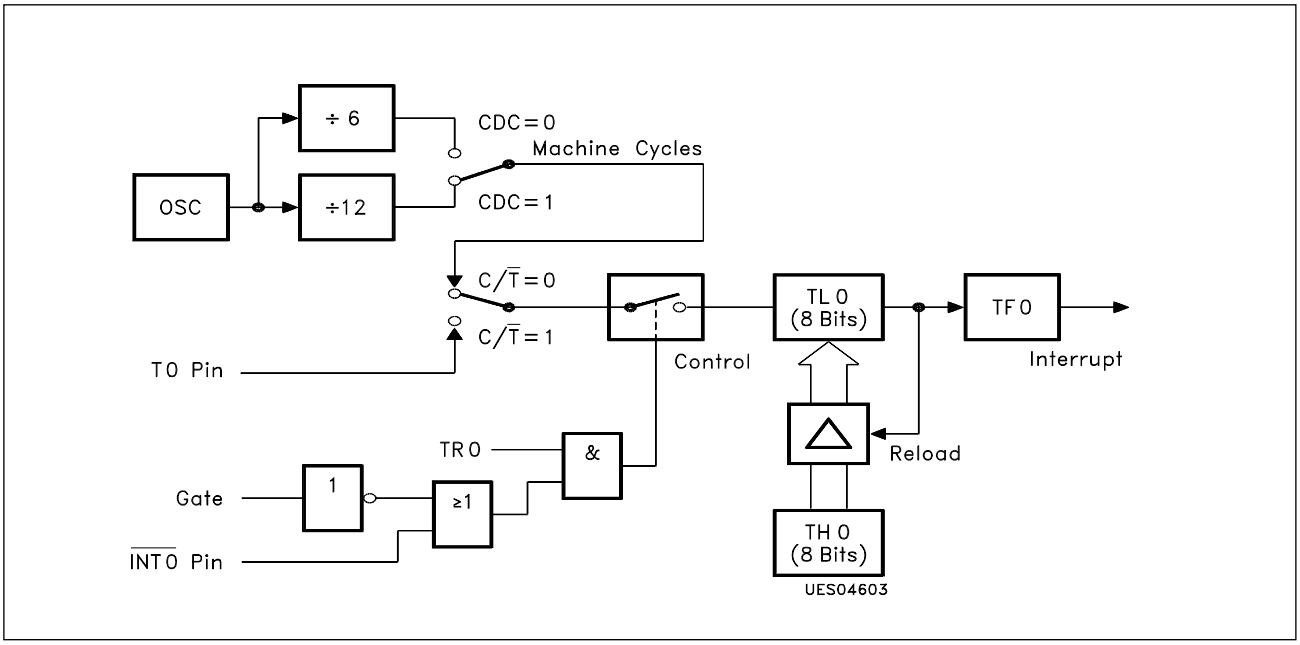


Figure 17
Timer/Counter 0 Mode 2: 8-Bit Auto-Reload

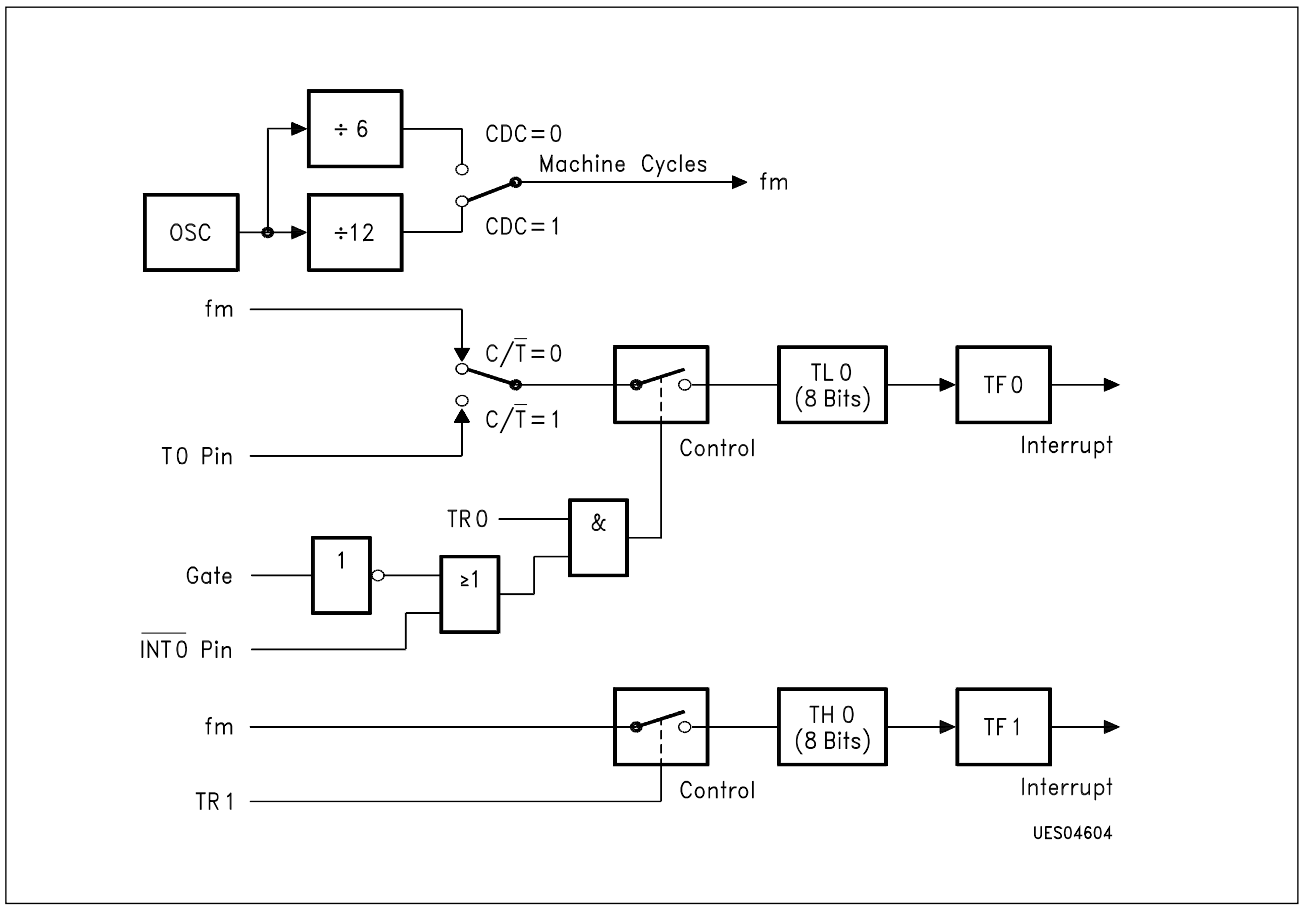


Figure 18
Timer/Counter 0 Mode 3: Two 8-Bit Counters

2.7 Multi-Mode Timer (not for SDA C0850)

The multi-mode timer is a 16-bit reloadable counter with a two-bit prescaler, which provides several functions. One of five different modes may be selected:

- **Watchdog Mode**
  for recovery from software or hardware upset
- **Timer Interrupt Mode**
  to generate an interrupt request at counter underflow
- **Timer Polling Mode**
  provides a 16-bit downward counter
- **Capture Mode 0** (no interrupt)
  to determine time difference between subsequent 1-to-0 transitions at pin P3.1 /CAP
- **Capture Mode 1** (with interrupt)
  to determine time difference between subsequent 1-to-0 transitions at pin P3.1/CAP and to generate an interrupt at every capture event or timer overflow

There is a variable prescaler (factor 1, 2 or 4) integrated, which enables the counter to be strobed by a timer clock that is 1/48,1/24 or 1/12 of the external oscillator frequency.

Independent of the chosen mode, the timer can be started, but cannot be stopped by software means. Only a clearing operation, which reloads and restarts the timer, can be done.

To ensure reliability (specially in watchdog mode), the clearing operation (i.e. reloading the counter) must be done by two subsequent MOV operations to different special function registers.

In watchdog, interrupt and polling mode, the timer decrements every timer clock cycle. The default reload value is $0FFFF_H$. The reload value might be changed by writing to the reload registers and is taken over by the counter, when a starting or clearing operation is performed. In capture mode, the timer increments and the default start value is 0000_H.

The cycle time of the multi-mode timer depends on the clock frequency, the prescaler factor and the reload value. It can be computed as:

$$T_{CYCLE} = (48/f_{OSC}) \times (WDRH \times 256 + WDRL + 1) / PRESC$$

$$(PRESC = 1, 2 \text{ or } 4)$$

Using an oscillator frequency of 12 MHz, a maximum timer cycle of 262.2 ms can be selected.

The prescale factor selection will be described in detail in the following.

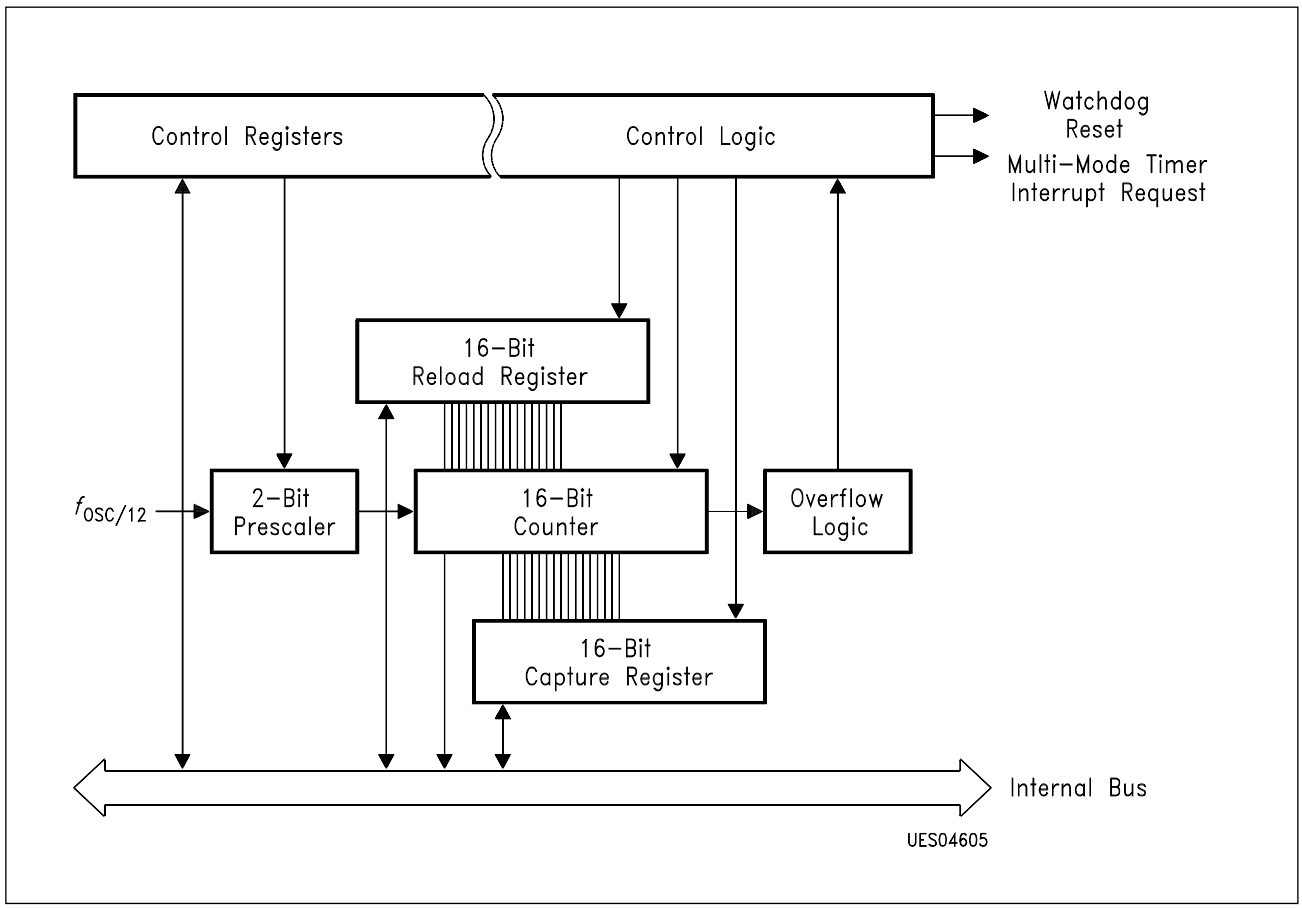


Figure 19
Block Diagram of the Multi-Mode Timer

Special Function Registers

MSB	SFR Address: 86_H						LSB
WDMOD: MMT Status Register							
WDTZ	PRS1	PRS0	M2	M1	M0	WDTS	SWDT

SWDT — Start Multi-Mode Timer. Setting SWDT = 1 will start the timer. This bit will be reset automatically; setting SWDT = 0 will not stop the timer! SWDT is a write-only bit. A read operation will always show a 1.

WDTS — Multi-Mode Timer Status WDTS = 1 flags that the timer has been started. This bit position can not be written to. The reset routine has to check this flag to differentiate between external and internal reset. This flag is not affected by an internal reset: WDTS = 1. External reset changes the flag: WDTS = 0.

M0/1/2 — Timer Mode Selection

M2	M1	M0	Timer Mode
0	0	0	Watchdog Mode
0	0	1	Timer Interrupt Mode
0	1	1	Timer Polling Mode
1	1	1	Capture Mode 0
1	0	1	Capture Mode 1

PRS0/1 — Prescaler Selection

PRS1	PRS0	Timer Input
0	0	$f_{osc}/48$
0	1	$f_{osc}/24$
1	X	$f_{osc}/12$

WDTZ — MMT Zero Flag. WDTZ is a read-only bit and indicates that the timer reached zero since the last read or write access to register WDTS. With every read or write access to register WDMOD the WDTZ flag is cleared automatically.

Note:

When using watchdog mode, M0, M1 and M2 have to be set simultaneously with starting the timer and can not be altered afterwards!

Default after reset: 01_H

MSB	SFR Address: 97_H						LSB
WDTC: MMT Clear Register							
—	CWDT	—	—	—	—	—	—

CWDT	Clear Multi-Mode Timer To clear the timer, bit CWDT has to be set to 1 immediately before setting SWDT. Like SWDT, CWDT will be cleared automatically.
Default after reset:	FF_H

Multi-Mode Timer Counter Register (WDCL, WDCH)

The low and the high byte of the 16-bit counter can be examined using the registers WDCL and WDCH. Only read access is possible.

Address Location WDCL, WDCH: 84_H, 85_H

Default Value WDCL, WDCH: FF_H, FF_H

Multi-Mode Timer Reload Register (WDRL, WDRH)

Using WDRL and WDRH, a 16-bit reload value may be written into the multi-mode timer reload register.

Address Location WDRL, WDRH: $8E_H$, $8F_H$

Default Value WDRL, WDRH: FF_H, FF_H

After starting the timer in watchdog mode, registers WDRL and WDRH can not be altered any more.

Multi-Mode Timer Capture Registers (CAPL, CAPH)

The 16-bit capture value will be stored in the two capture registers. These registers can be read from or written to.

Address Location CAPL, CAPH: 91_H, 92_H

Default Value CAPL, CAPH: 00_H, 00_H

Timer Modes

The five different timer modes are selected by the three control bits M0, M1 and M2 in status register WDMOD. Except for watchdog mode, modes can be changed by switching the control bits. Only the described mode selections may be chosen. Other combinations of the mode control bits must not be used.

Watchdog Mode

In watchdog mode, the multi-mode timer serves for recovering the processor from software or hardware upset. The timer can be started and cleared (which means reloaded) by software but it cannot be stopped. If the timer is once started and the software fails to clear it at least every timer cycle, a counter underflow will cause an internal reset. The timer cycle is determined by the reload value, the prescaler and f_{OSC}. The default reload value is $FFFF_H$ (the timer works decrementing). If a reload value different from $FFFF_H$ is to be used, the reload registers have to be written before entering watchdog mode; WDRL and WDRH will be set to a read-only state after the timer is started. The reload values are taken over to the timer at start time and with every clearing operation. In case of a reset, the reset routine can examine the reset cause (external of internal) by checking bit WDTS (WDMOD.1).

Internal reset released by the multi-mode timer does not affect the registers WDMOD, WDRL and WDRH. Bit WDTS in register WDMOD is set and bit WDTZ in register WDMOD is left undefined. The counter starts again from $FFFF_H$. So the reset routine will have time to run the application specific initialization.

An **external reset** will clear and stop the counter; registers WDRL, WDRH and WDMOD will be set to their default values and so bit WDTS in register WDMOD will be cleared.

Watchdog mode is entered by setting WDMOD as described in the following

PRS1	PRS0	Timer Clock	WDMOD Value
0	0	$f_{osc}/48$	01_H
0	1	$f_{osc}/24$	21_H
1	0	$f_{osc}/12$	41_H

Figure 20 shows a flow chart of a watchdog application.

Timer Interrupt Mode

The interrupt mode configures the multi-mode timer as an additional interrupt source.

If the interrupt is enabled (bit EPW and bit EA in register IE must be set), an interrupt request will be generated by every timer underflow. The interrupt service routine starting at location $02B_H$ will have to check the interrupt flag register IFR at SFR location $0FA_H$, whether the interrupt was initiated by the multi-mode timer or the PWM unit. If the multi-mode timer caused the interrupt, bit IFR.1 will be set. For detailed information about interrupt handling see chapter “Interrupt System”.

The timer itself shows the following features:

– the timer starts at the reload value

– it works decrementing

– at underflow an interrupt is requested and the timer simultaneously restarts at the reload value

– it is possible to modify the reload register while the timer is working

– the clearing operation (restart with reload values) can be performed

Timer interrupt mode is selected by setting WDMOD to one of the following values:

PRS1	PRS0	Timer Clock	WDMOD Value
0	0	$f_{osc}/48$	05_H
0	1	$f_{osc}/24$	25_H
1	0	$f_{osc}/12$	45_H

Polling Mode

Using polling mode, neither reset nor interrupt are initiated when the timer reaches zero.The user program has to read the counter status and act accordingly.

In polling mode the timer shows the following features:

– the timer starts at the reload value

– at underflow the timer restarts at the reload value

– it is possible to modify the reload register while the timer is working

– the clearing operation (restart with the reload value) can be performed

Polling mode is started with one of the following values written to WDMOD:

PRS1	PRS0	Timer Clock	WDMOD Value
0	0	$f_{osc}/48$	$0D_H$
0	1	$f_{osc}/24$	$2D_H$
1	0	$f_{osc}/12$	$4D_H$

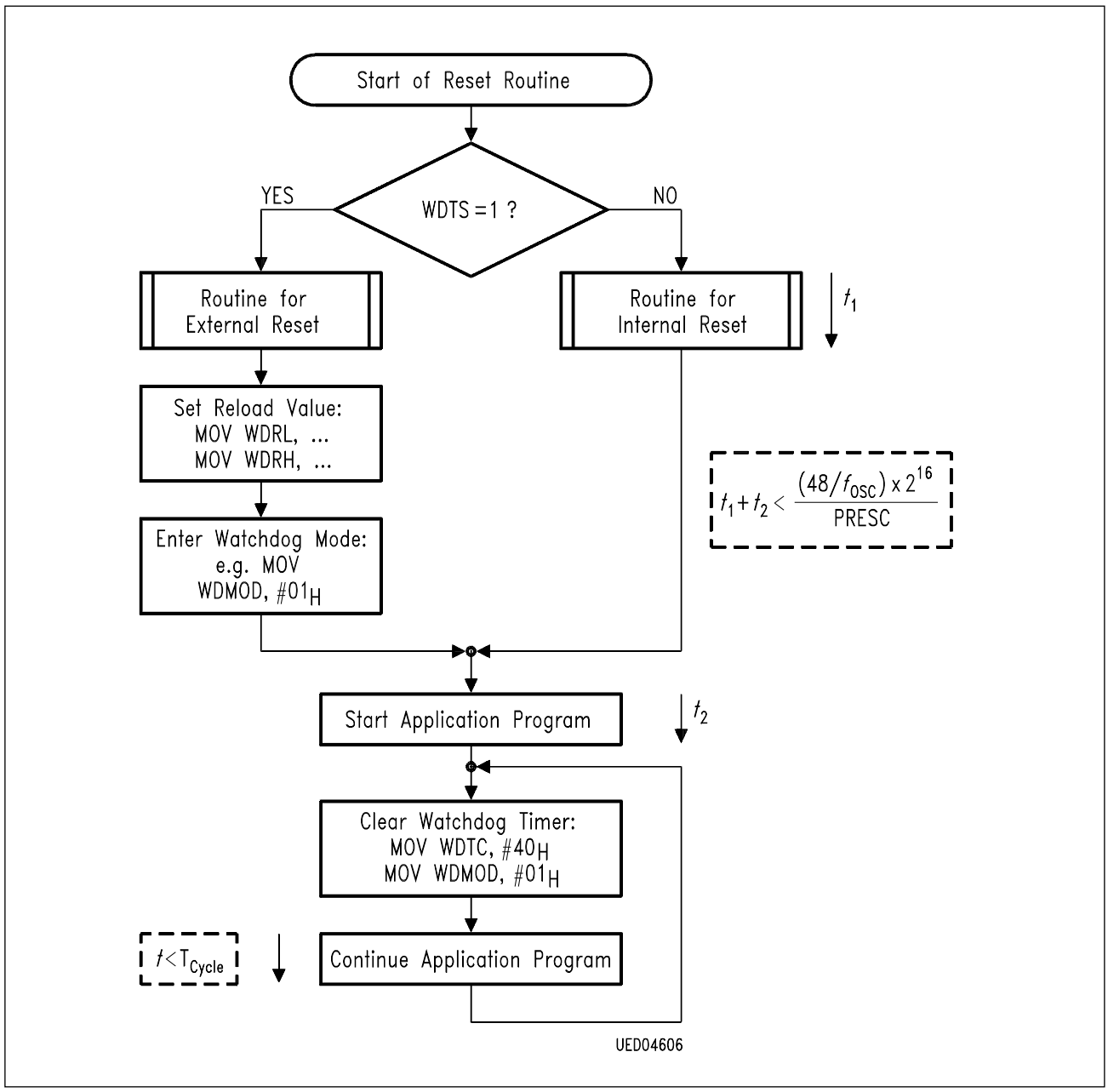


Figure 20
Watchdog Mode

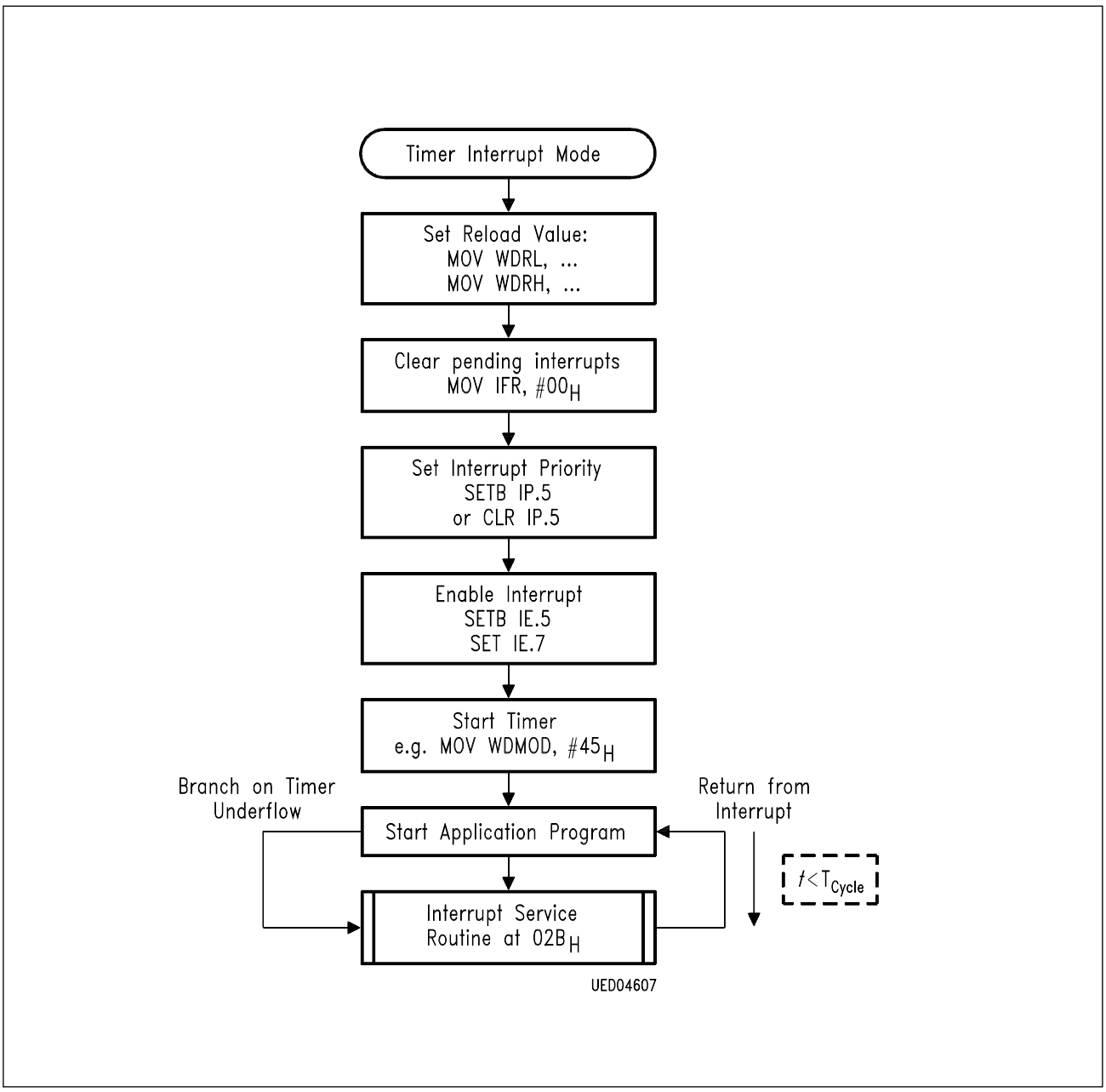


Figure 21
Timer Interrupt Mode (Main Program)

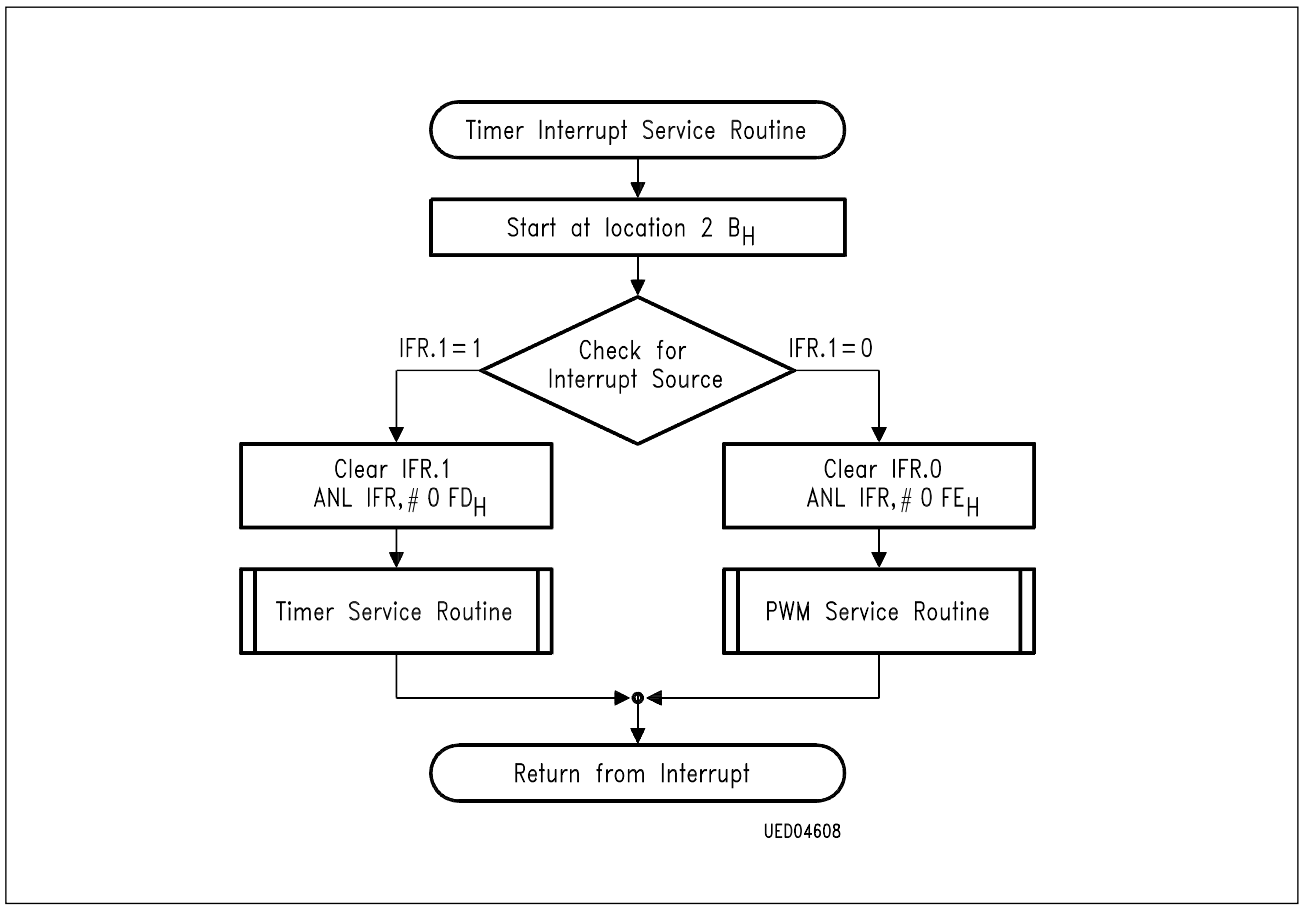


Figure 22
Timer Interrupt (Service Routine)

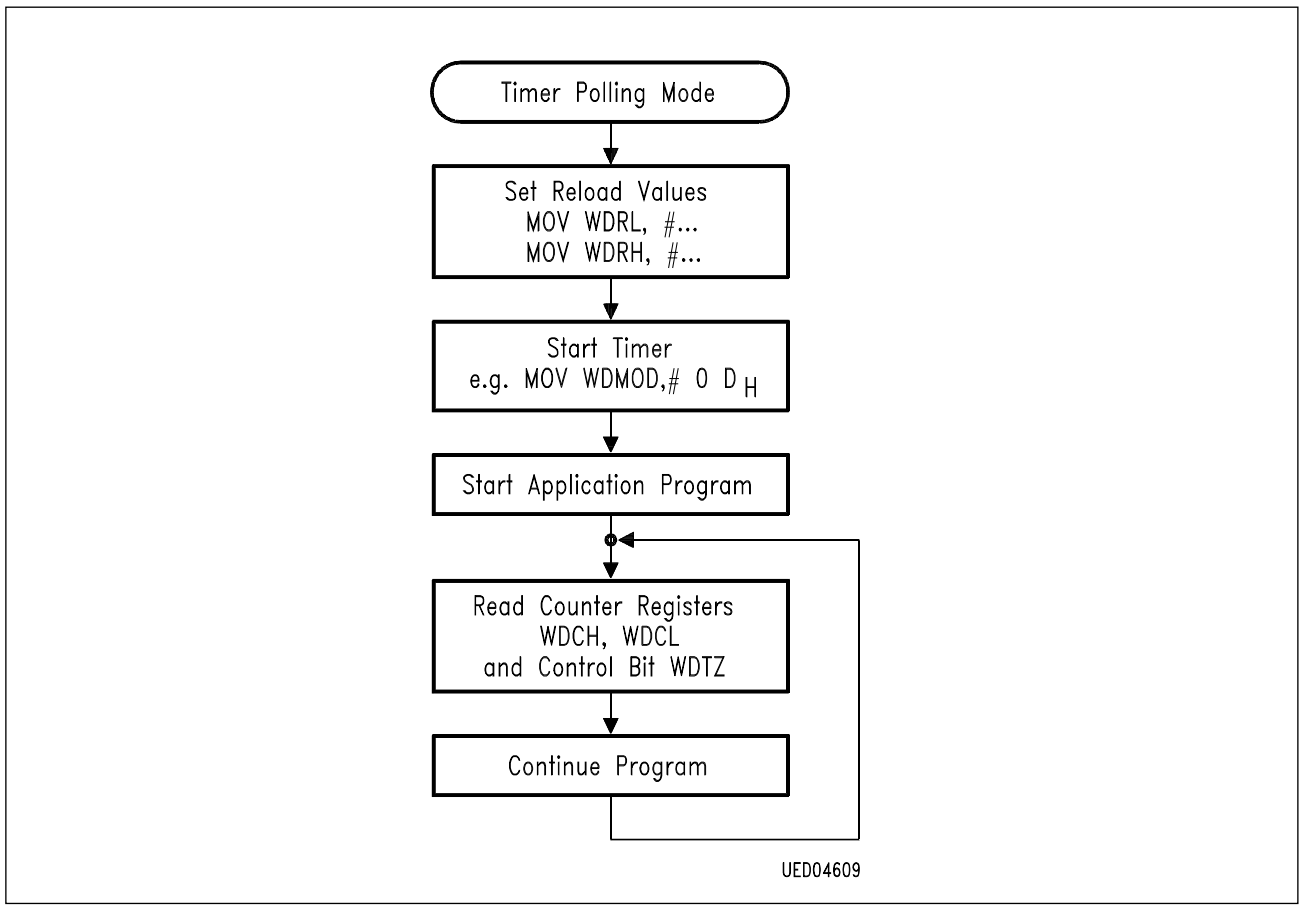


Figure 23
Timer Polling Mode

Capture Mode 0

In capture mode 0, the time difference between two subsequent falling edges at the capture input pin (P3.1/CAP) can be determined. The first falling edge after enabling capture mode and starting the counter resets the counter to the start value (e.g. 0000_H). The second and following edges will latch the counter value into the 16-bit capture register (CAPL, CAPH) and will restart the counter again (**see figure 24**). Capture mode can be terminated by changing mode or by an external reset.

In capture mode, the default start value might be changed; the counter will start from this value at the next 1-to-0 transition. For this purpose, the **complemented** start value has to be written into reload registers WDRL and WDRH. The default start value is 0000_H (WDRL, WDRH = FF_H).

When capture mode is terminated, the last capture value will remain in CAPL and CAPH registers. The time between two subsequent falling edges at P3.1/CAP is to be computed as:

$$T_C = (256 \times CAPH + CAPL + 1) \times PRESC \times \frac{12}{f_{OSC}}$$

(PRESC = 1,2 or 4)

Capture mode 0 is entered by setting WDMOD as described in the following:

PRS1	PRS0	Timer Clock	WDMOD Value
0	0	$f_{osc}/48$	$1D_H$
0	1	$f_{osc}/24$	$3D_H$
1	0	$f_{osc}/12$	$5D_H$

A flow chart of capture mode 0 is shown in **figure 25**.

If capture mode is not selected, the two capture registers can be used as additional scratch registers without disturbing any other multi-mode timer function.

Capture Mode 1

In addition to capture mode 0, every capture event or timer overflow causes a branch to the interrupt service routine at location $2B_H$, if bits EPW and EA in register IE are set.

Using control bit WDTZ in register WDMOD, the interrupt routine will be able to distinguish between these two cases:

WDTZ = 0 after interrupt: capture event
WDTZ = 1 after interrupt: timer overflow

WDTZ is to be cleared after starting capture mode by reading control register WDMOD once.

To start capture mode 1, WDMOD is to be set as follows:

PRS1	PRS0	Timer Clock	WDMOD Value
0	0	$f_{osc}/48$	15_H
0	1	$f_{osc}/24$	35_H
1	0	$f_{osc}/12$	55_H

A flow chart of capture mode 1 is shown in **figure 26**.

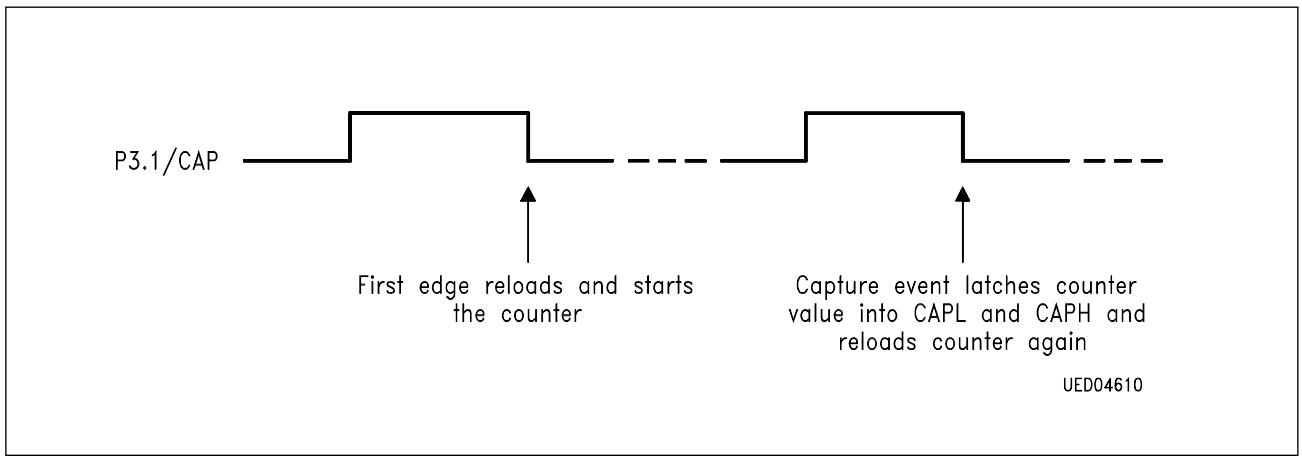


Figure 24
Capture Mode Timing Diagram

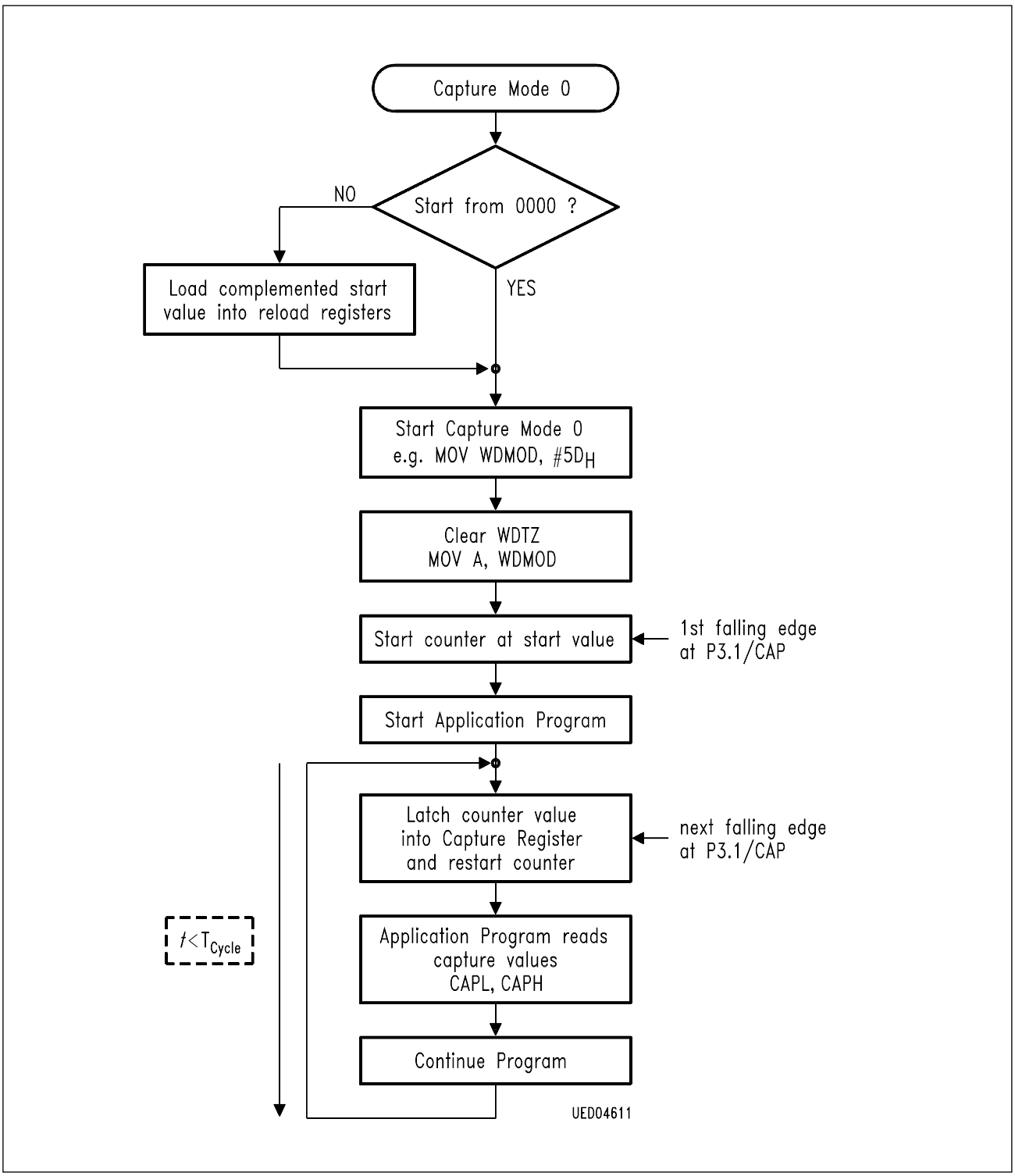


Figure 25
Capture Mode 0

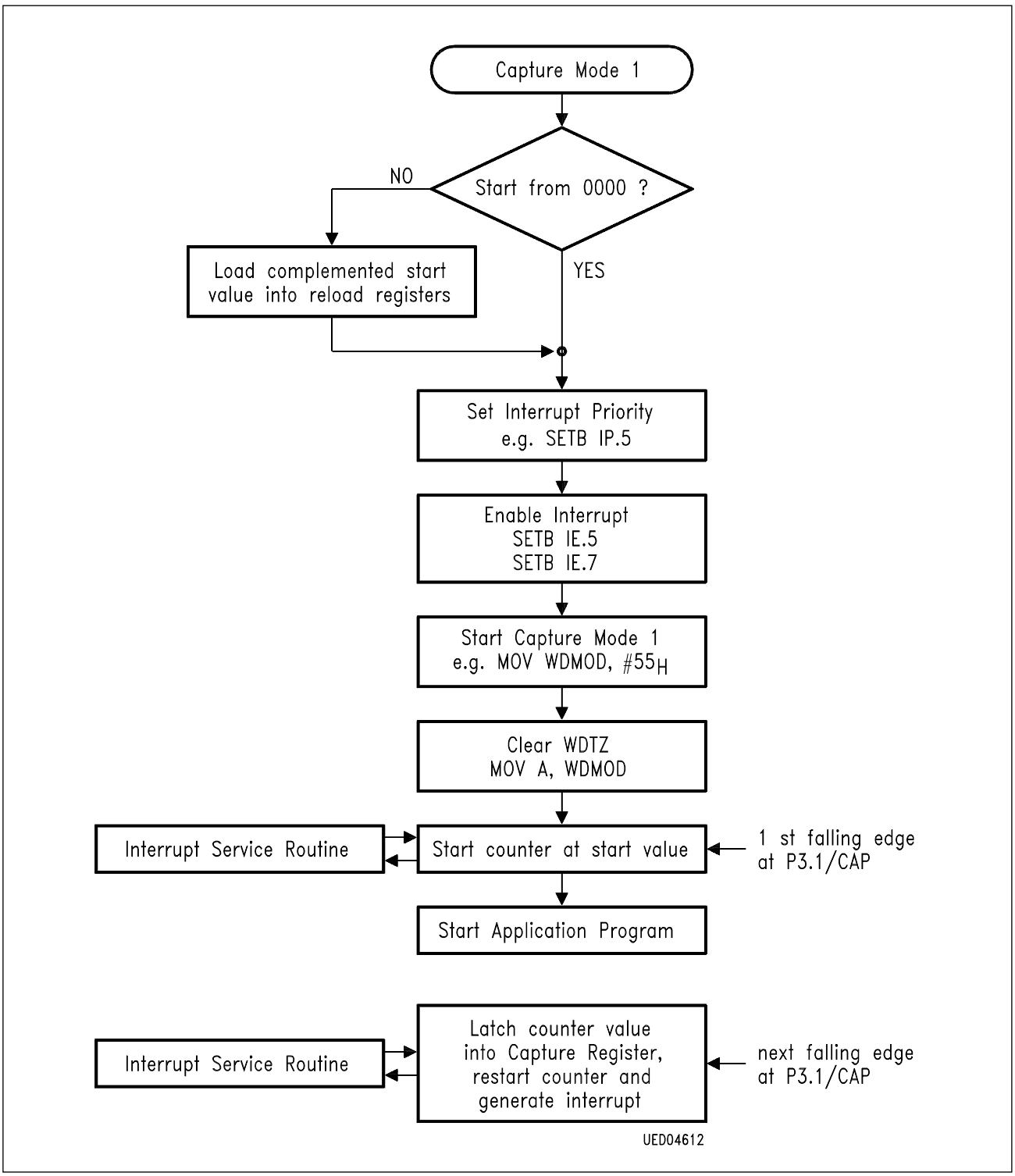


Figure 26
Capture Mode 1

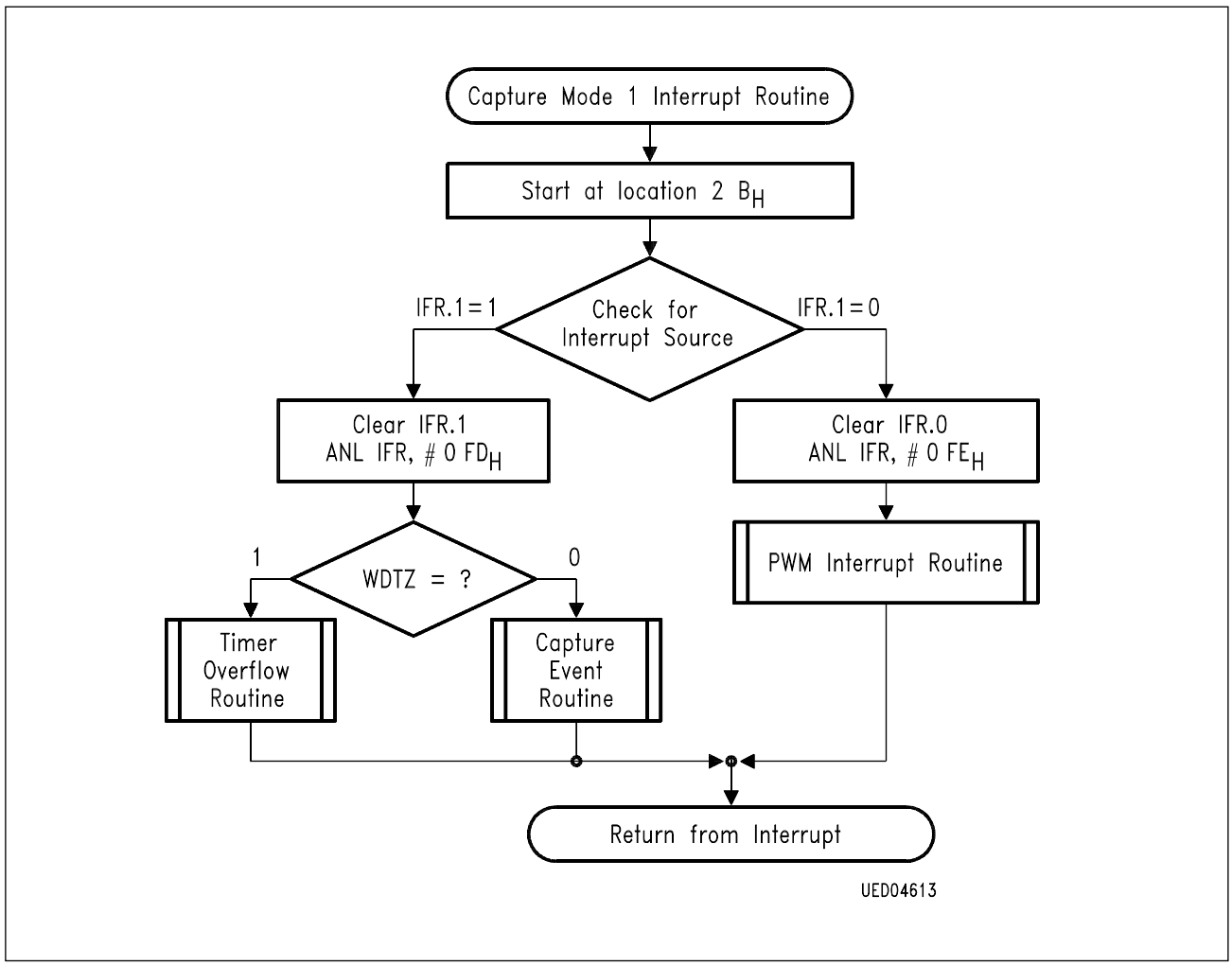


Figure 27
Capture Mode 1 (Interrupt Routine)

2.8 Pulse Width Modulation Unit

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

General Considerations

The PWM output channels are placed as alternate functions to the eight lines of port 1 (P1.0...P1.7). Each PWM channel can be individually switched between PWM function and port function. The PWM unit is controlled by the special function register PWMC located at address $0C8_H$. This register determines the counter's resolution (6, 7 or 8 bit) and starts or stops the counter. A counter status bit can be read and an interrupt enable flag can be set. Except for the status bit, read and write accesses are possible for this register. The PWMC register's lowest 3 bits are not employed and can be used as extra software flags (C0, C1, C2).

The eight 8-bit compare registers PWCOMP0 – PWCOMP7 located at SFR addresses $0F1_H$ – $0F8_H$ contain the modulation ratios of the output signals which are related to the maximum defined by the counter's resolution. These compare registers are double buffered and a new compare value will only be taken into the main register, if the PWM timer is stopped or after the next timer overflow. To avoid overwriting the desired compare value, the counter status bit should be checked before a new write operation to a compare register is done.

The PWM timer register located at SFR address $0F9_H$ contains the actual value of the PWM counter and can only be read by the CPU. Every compare register, which is not employed for the PWM output can be used as an additional register. This is not allowed for register PWME. If the PWM function is not activated, the PWM timer is available for any other timing purpose.

PWM Control Register PWMC

MSB	SFR Address: $C8_H$						LSB
PWMC: PWM Control Register							
S	M1	M0	R	IR	C2	C1	C0

Default after reset: 80_H

Function of the control bits:

R = 0 The PWM timer is stopped and reset to 00_H. All output latches (OL0 … OL7) are set to 1.

= 1 The PWM timer is set to RUN. At timer overflow, all output latches OL0...OL7 are set to 1. If the timer value meets the compare value of channel i, OLi is reset to 0.

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

C0, C1, C2 General purpose software flags.

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

S Shows the actual state of the PWM timer. S is set by PWM timer overflow and has to be reset by software.

This bit may be used to control whether a value selected for a compare channel was already written into the compare latch by a PWM timer overflow.

IR = 0 A PWM interrupt will not be requested.

= 1 At every timer overflow bit IPWM in register IFR will be set to “one”. This may initiate an interrupt request, if bit IE.5 in the interrupt enable register IE is set.

PWM Enable Register PWME

MSB	SFR Address: $C0_H$						LSB
PWME: PWM Enable Register							
E7	E6	E5	E4	E3	E2	E1	E0

Default after reset: 00_H

Ei = 0 The corresponding PWM channel is disabled.

(i = 0 … 7) P1.i functions as normal bidirectional I/O port.

= 1 The corresponding PWM channel is enabled. P1.i is automatically set to logic 1 and is connected to the output latch of the corresponding PWM channel (OLi).

PWM Compare Registers PWCOMPx

Each of the eight compare channels consists of

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

The SFR addresses are:

PWCOMP 0: $0F1_H$

PWCOMP 1: $0F2_H$

PWCOMP 2: $0F3_H$

PWCOMP 3: $0F4_H$

PWCOMP 4: $0F5_H$

PWCOMP 5: $0F6_H$

PWCOMP 6: $0F7_H$

PWCOMP 7: $0F8_H$

After reset, the register contents are $0FF_H$.

– an 8-bit compare latch, which is loaded with the value in the 8-bit Register, if the PWM timer overflows or stops.

– a comparator, which compares the value of the compare latch with the timer value. If (M1, MO) ≠ (0,0), only the 7 (or the 6) least significant bits will be compared.

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

PWM Timer Register PWCOUNT

(Address $0F9_H$, Reset value 00_H)

An 8-bit upwards counting binary counter (with an input frequency of $f_{OSC}/2$) is provided as PWM timer. The counter registers can be read by software at SFR address $0F9_H$, but cannot be written to. If in PWMC register R = 0, the PWM timer will be held at 00_H, i. e. at the reset value. If R = 1, the PWM timer will increment six times every CPU instruction cycle. M1 and M0 (in PWMC register) control the value, from which the PWM timer overflows to 00_H:

M1	M0	Overflow Value
0	0	$0FF_H$ (= 255)
0	1	$07F_H$ (= 127)
1	0	$03F_H$ (= 63)

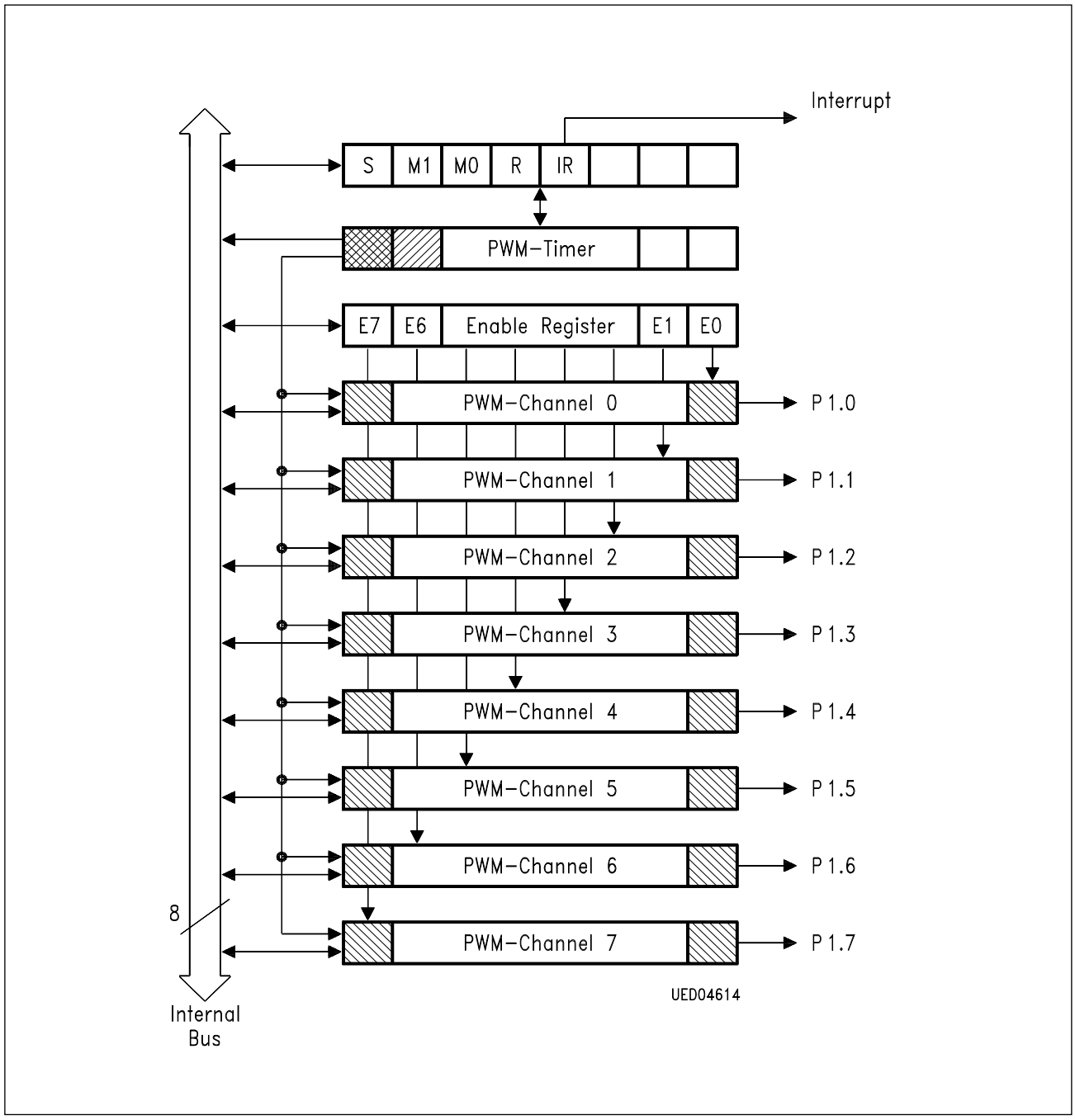


Figure 28
Block Diagram of Pulse Width Modulation Unit

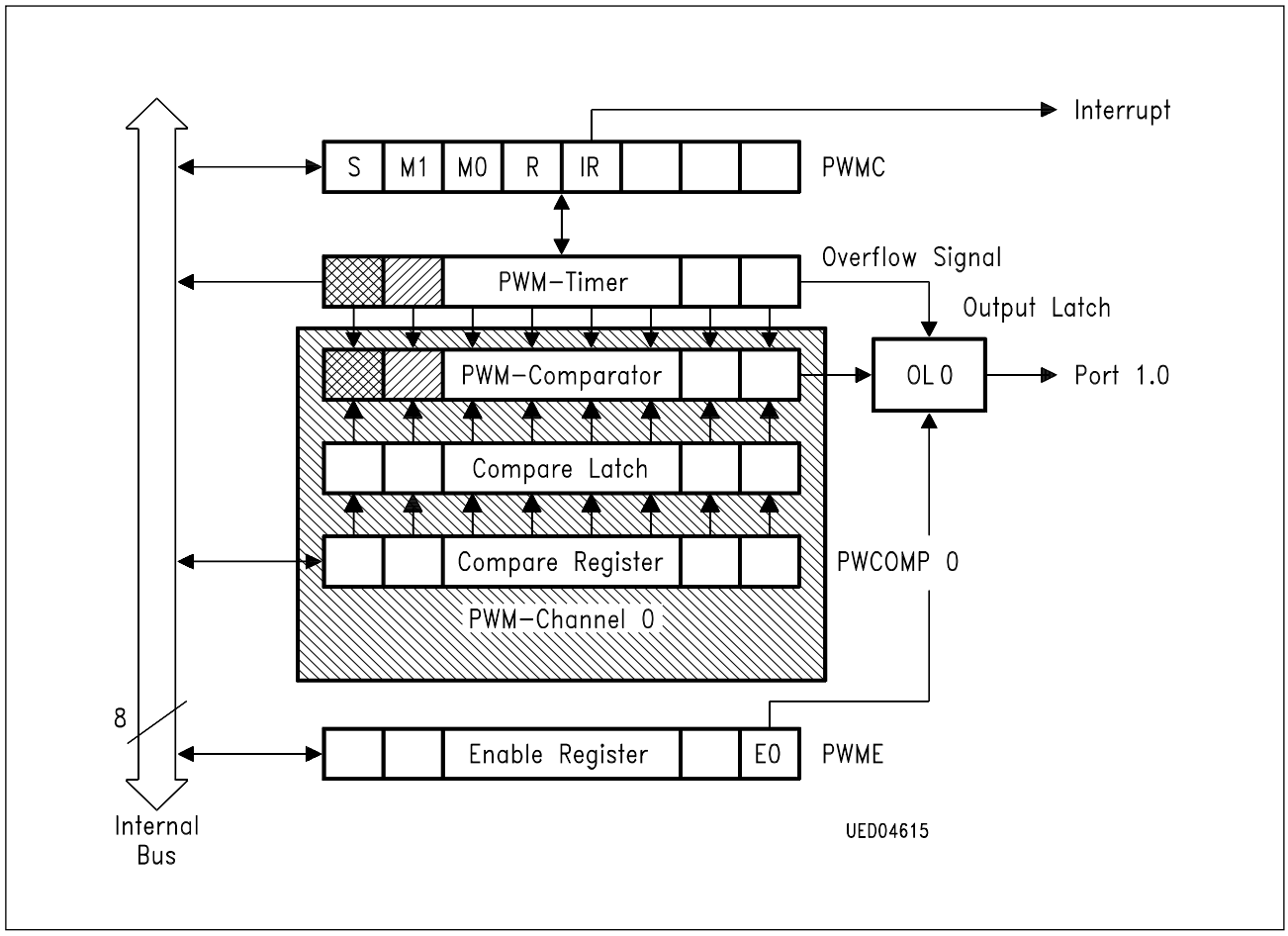


Figure 29
Block Diagram of One Pulse Width Modulation Channel (e.g. PWM0)

2.9 On-Screen Display

The OSD circuit generates video signals to display text or graphic symbols together with a video picture. The sandcastle signal (SC) is used to synchronize the circuit, which generates R-, G-, B- and BLANK signals to be mixed with an incoming RGB video signal. Internal sync signals HS and VS are derived from SC.

The dot clock is generated by an internal LC oscillator with external reference elements and synchronized by the rising edge of internal signal HS.

Each of the six rows of text holds up to 18 characters and may be positioned and enabled individually. Using the OSD interrupt function, even more than six lines can be used.

Boxed or non-boxed display mode and one of two text sizes may be selected for each row. One of eight background colors may be chosen globally and, additionally, full screen blanking is possible. In non-boxed mode, a colored frame may be activated to increase character readability.

A set of 96 characters, each defined by a 12×16 pixel matrix, is contained in a mask-programmable character ROM in the SDA 20Cxx50 devices and may be user-defined according to the specific application. The SDA 30C0050 comprises an internal pixel RAM for emulation purpose.

The OSD circuitry is designed to be used in a color TV application, where each character may be displayed in any of eight colors.

Principle diagrams of the OSD circuit are shown in **figure 30 to 32**.

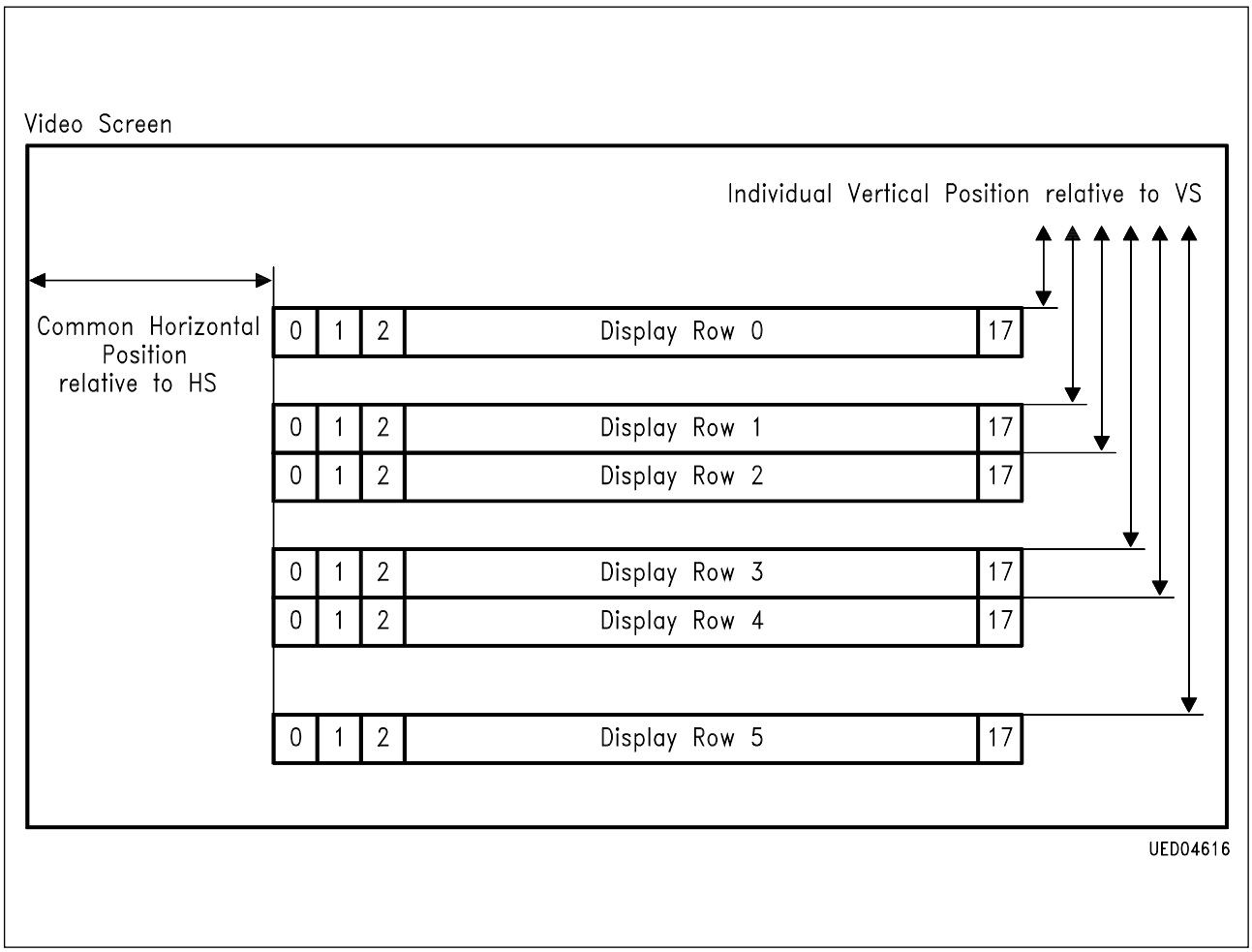


Figure 30
Display of Six Rows of Text on a Video Screen (without using interrupt)

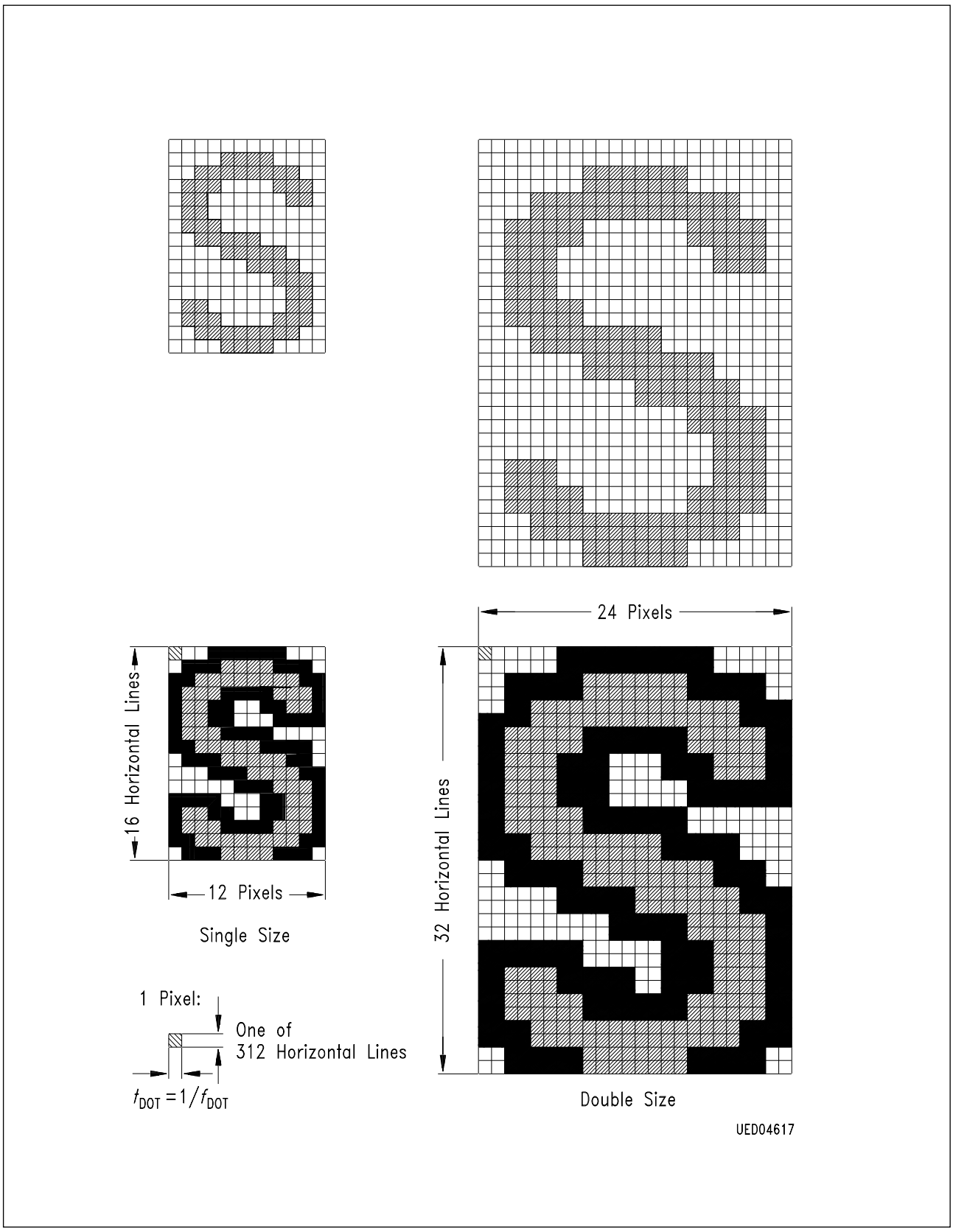


Figure 31
Single and Double Size Character Display with Programmable Frame Mode

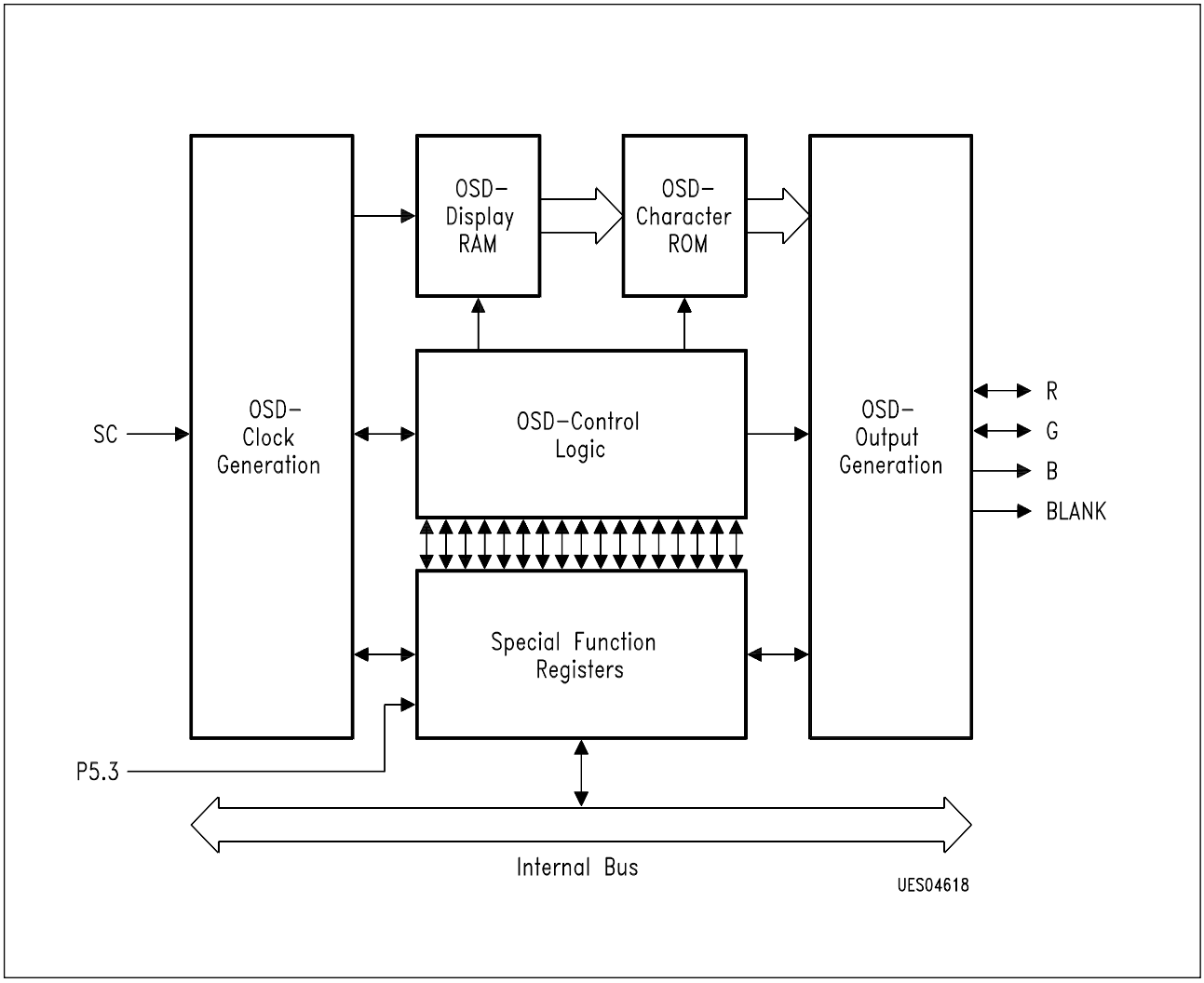


Figure 32
OSD Block Diagram

OSD Features and Programming

This chapter gives a general description of the on-screen display circuit and the various programming features. Detailed informations for programming the device are found in chapter “OSD Special Function Registers”.

Dot Clock

The OSD circuit contains an LC oscillator circuit for generation of a synchronized dot clock. The oscillator is synchronized by the active edge of internal signal HS, internal counters are reset with the falling edge of HS. HS is internally decoded from the synchronization input signal SC.

The dot clock frequency defines the pixel width on the screen (**figure 33**).

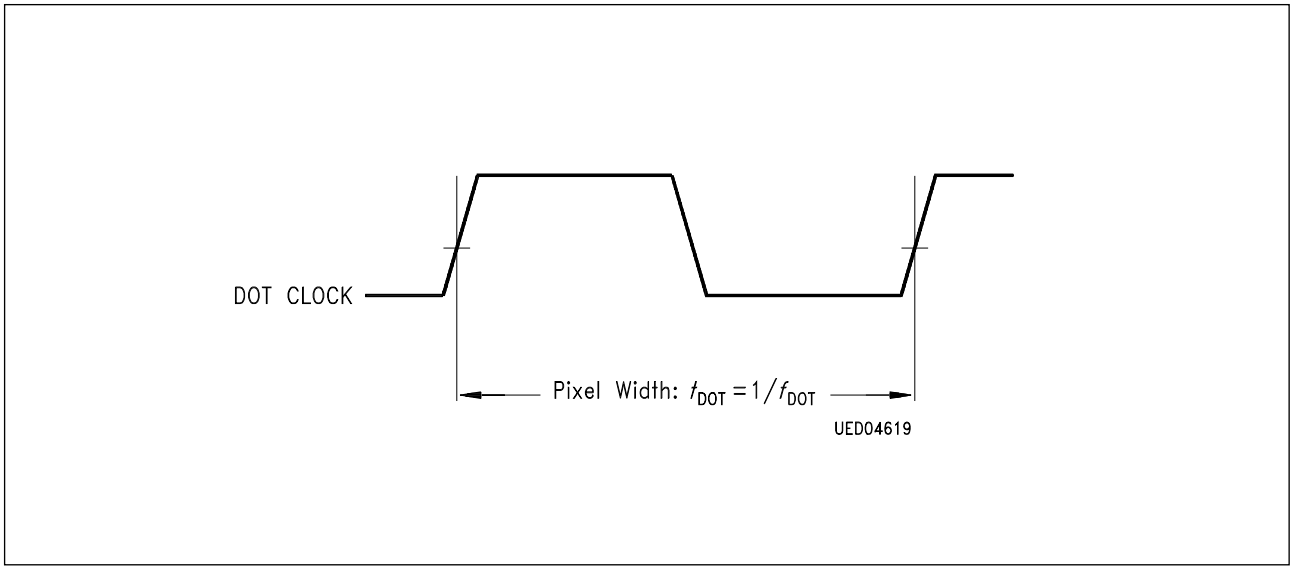


Figure 33
Pixel Width Definition

The dot clock frequency is determined by the external reference elements.

Horizontal Positioning

The horizontal start position is common to all lines of text and is independent of the selected text size. The time between the rising edge of SC and the first pixel to be displayed may be calculated as follows:

t_{HO} = (DHR + C) × t_{DOT}

$C \approx 17$ (depends on the dot clock frequency)

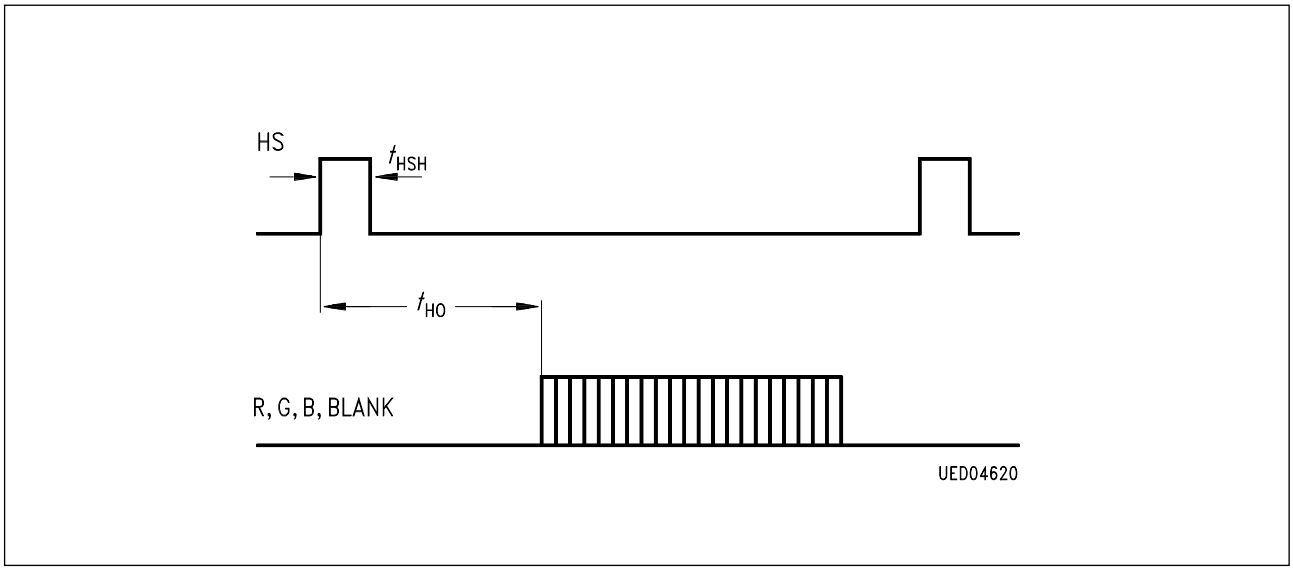


Figure 34
Horizontal Offset Timing Diagram

The value written to DHR may be set in steps of four, because DHR.0 and DHR.1 are always set to 0 internally. Time t_{HO} must be greater than the HS high time t_{HSH}.

Vertical Positioning and Line Modes

The vertical position of row i and the line mode may be chosen individually for each of the six rows of text. Each row is controlled by a vertical control register DVRi (i = 0,..., 5).

Bit Si defines the text size to normal (Si = 0) or double size (Si = 1), bit Bi enables (Bi = 1) or disables (Bi = 0) boxed mode. Bits Vi0 ... Vi5 define the vertical start position in steps of 8 horizontal lines.

The values of the vertical control registers DVRi are compared with the actual count value and a OSD output is started when the numbers match. If two or more lines are programmed to overlap, the line with the lowest index is displayed on top.

Display Enable

Each of the six display rows is enabled by one bit in OSD Enable Register DER. Additionally, a global enable bit in this register allows to enable or disable all individually enabled lines.

Background Colors

If boxed mode is selected, the respective rows of text are displayed on a colored background

The background color is common to all rows and is defined in OSD Color and Polarity Register DCPR. Moreover, if bit FBG (full background) in this register is set, the whole screen will be blanked with the selected background color.

Frame Mode

For all lines not displayed in boxed mode, a color frame around the characters may be activated. This will increase the character contrast to the video background.

OSD Interrupt

The OSD circuitry is designed for a main use of six lines of text. In some applications, however, more than six lines are to be displayed. This is possible, if a cyclic reload of display lines and a vertical repositioning is done via the OSD interrupt. Using this way of time multiplexing, 12 lines or even more can be displayed (**figure 35/36**) and background color switching is possible.

If enabled (by setting bit DER.6), every completion of a text row output will generate an interrupt, which is to be handled like the standard external interrupt 1. In this case, the state of pin P3.3/INT1 is ignored (bits EA and EX1 in register IE must be set).

Output Signal Polarity

The signal polarity for BLANK may be selected by setting a control bit in DCPR.

Sandcastle Operation

The principle of sandcastle synchronization is shown in **figure 37**.

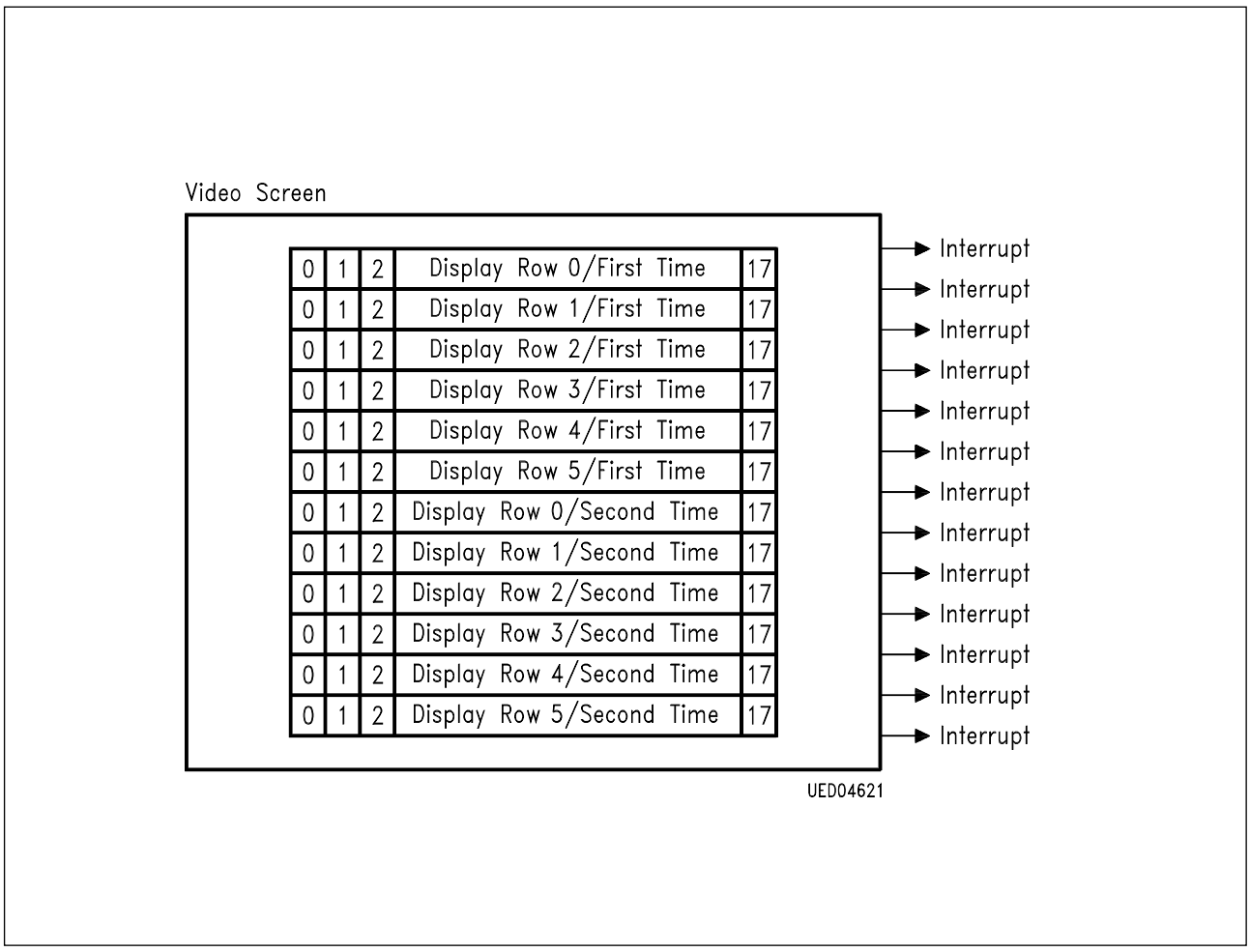


Figure 35
Display of More than Six Rows of Text on a Video Screen (using OSD-interrupt)

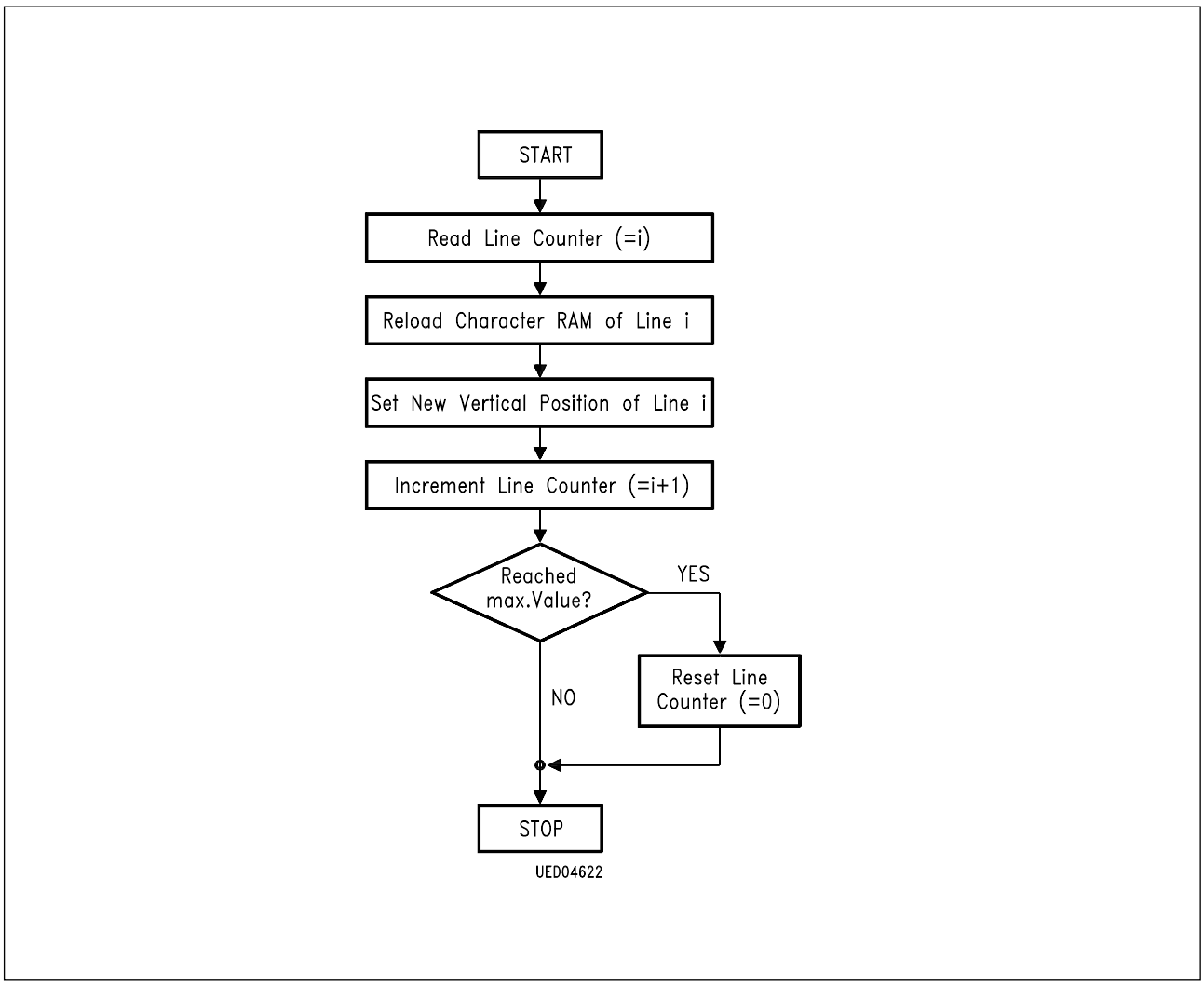


Figure 36
Flow Chart of OSD Interrupt Service Routine

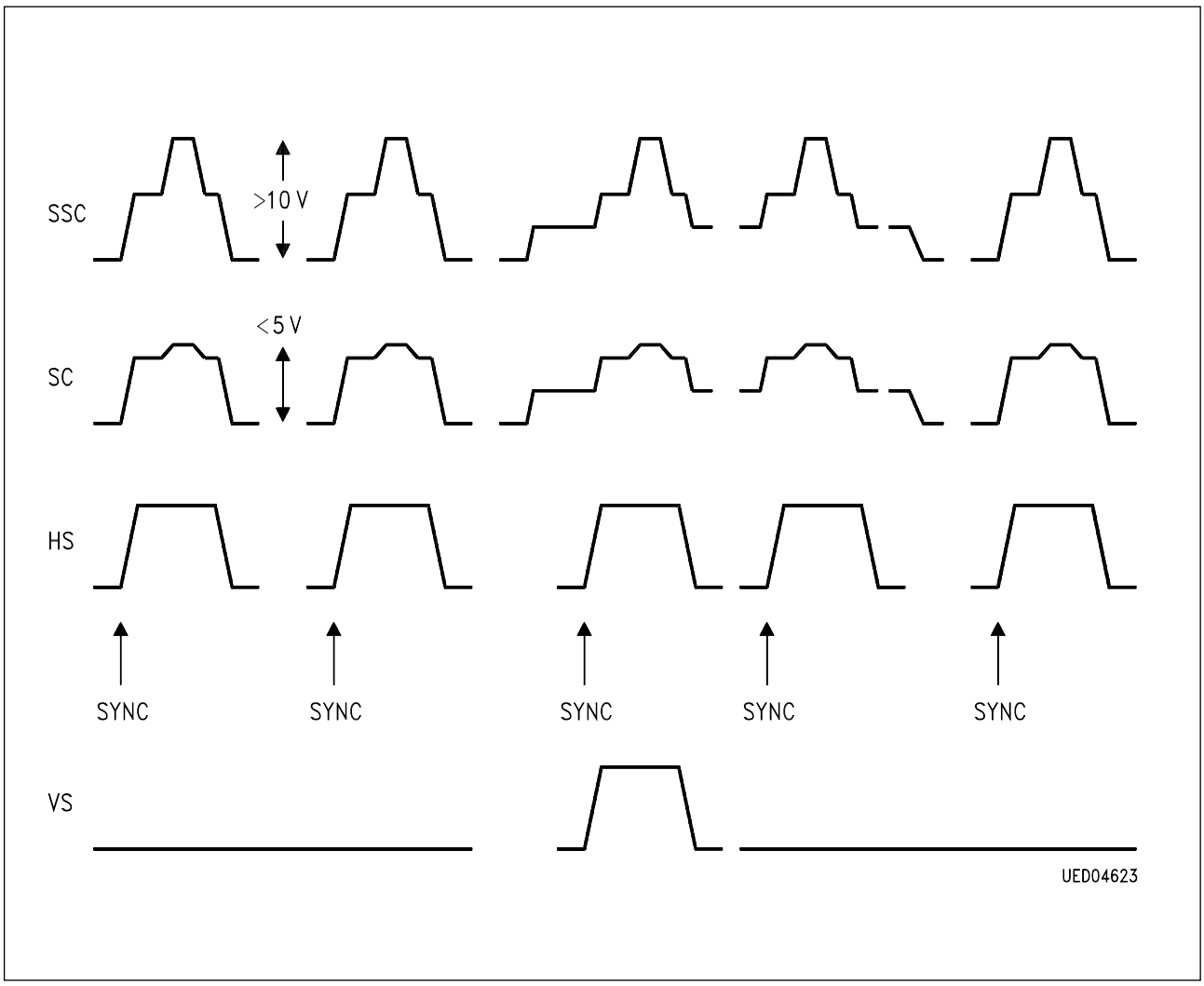


Figure 37
Sandcastle Synchronization Scheme

Display RAM Access

The information contained in the six display rows depends on the contents of the OSD display RAM. Each display position corresponds to one RAM location. Accesses to the display RAM are done in a two-step way. At first, the display RAM address is written to the OSD Address Register DAR, secondly the data and color information are written to or read from the OSD Registers DDR and DSCR, respectively. Note, that color information must be written first.

Using registers DAR, DDR and DSCR in this way, character and color codes can be written to and read from any of the $6 \times 18 = 108$ locations of display RAM, each location representing a 10-bit word.

Flow charts of read and write accesses are shown in the following figures.

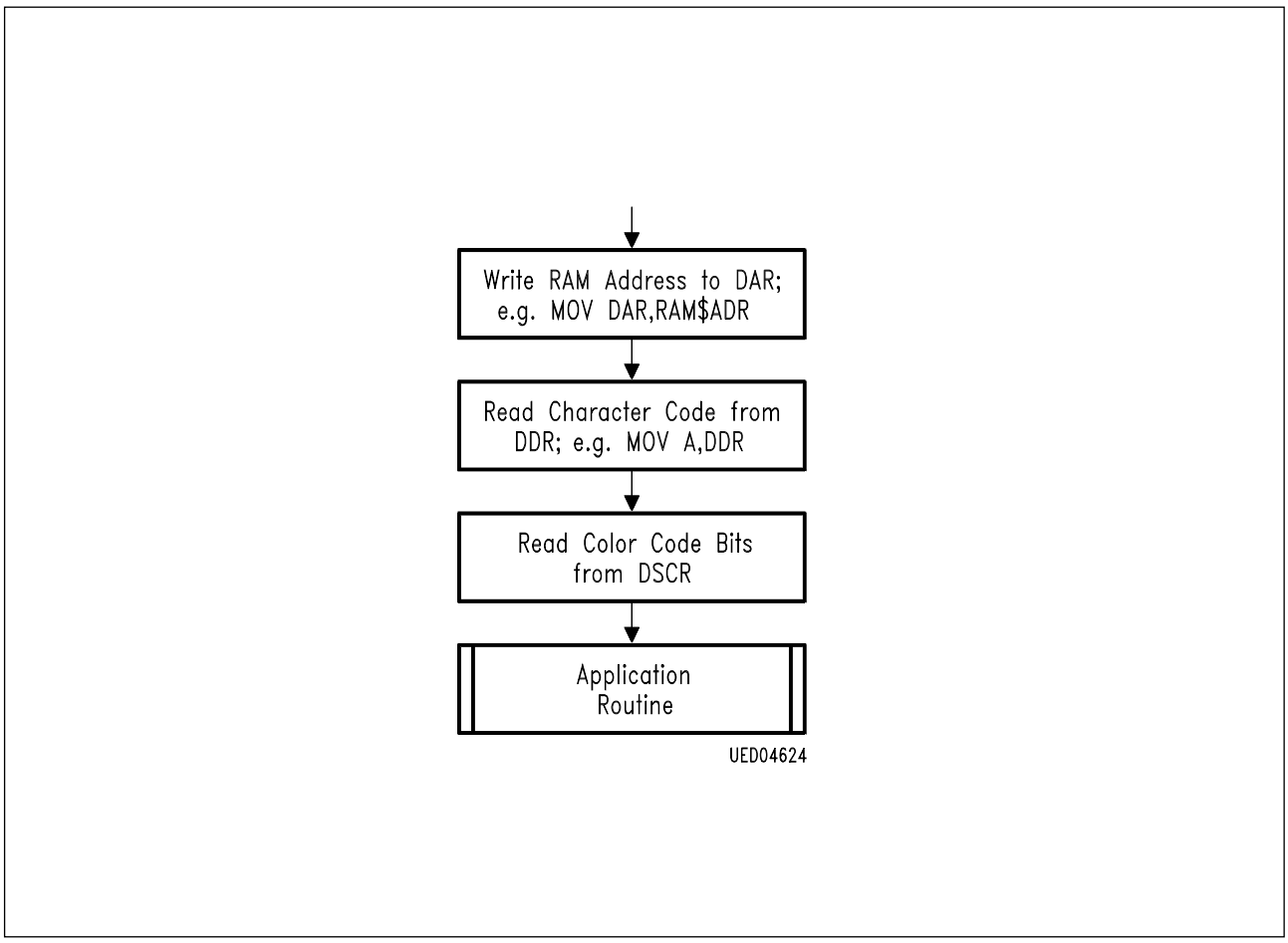


Figure 38
OSD RAM Read Access

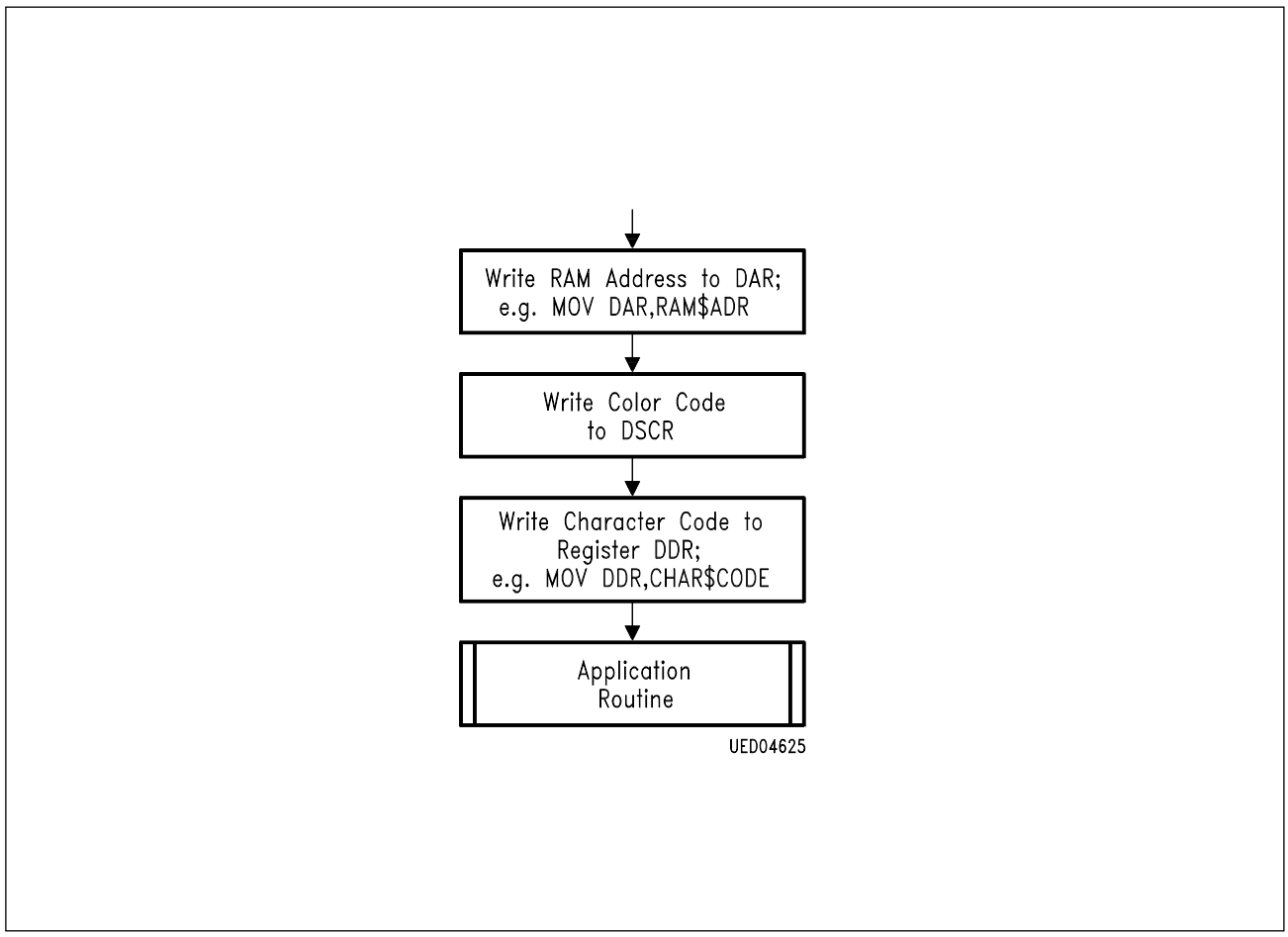


Figure 39
OSD RAM Write Access

OSD Special Function Registers

Bits notified **“unused”** in OSD special function registers may be used as additional software flags. Bits notified **“reserved”** must not be used.

Display Enable Register DER

MSB	SFR Address: $D1_H$						LSB
DER: Display Enable Register							
GE	DIE	EN5	EN4	EN3	EN2	EN1	EN0

GE:	Global Enable; enables (1) or disables (0) all individually enabled rows of text
DIE:	Display Interrupt Enable; if set to “1”, after every text row completion an interrupt will be generated and internally be switched to the “External Interrupt 1”. This interrupt should be programmed to edge triggered mode (see chapter “Interrupt System”).
ENi:	Enable Display Row i; individual enable bit for row i
ENi = 0:	OSD row i not displayed
ENi = 1:	OSD row i displayed, if GE = 1
Default after reset:	00_H

Display Vertical Control Registers DVRi

MSB	SFR Addresses: $D2_H$…$D7_H$						LSB
DVR0 … 5: Display Vertical Control Register							
Vi5	Vi4	Vi3	Vi2	Vi1	Vi0	Bi	Si

Vi0 ... 5:	Vertical position of row i in binary (000000_B ...100111_B)
Bi:	Row i background flag
Bi = 0:	No background (non boxed)
Bi = 1:	Colored background (boxed)
Si:	Character size of row i
Si = 0:	Normal size
Si = 1:	Double size
Default after reset:	undefined

Display Status and Control Register DSCR

This register is bit-addressable.

MSB	SFR Addresses: $D8_H$						LSB
DSCR: Display Status and Control Register							
1	HSY	VSY	–	–	R	G	B

DSCR.7: A constant value of “1” is read (for SW compatibility to SDA 2056X devices)

HSY: Horizontal synchronization status bit

VSY: Vertical synchronization status bit

R: Red color select

G: Green color select

B: Blue color select

Bits R, G and B define the color of the next character to be written to the display RAM via display data register DDR. When reading display RAM information, R, G and B are valid after DRW = 1.

The resulting color of any R-G-B combination is shown in table “OSD Color Definitions”.

DSCR.3: reserved
DSCR.4: reserved

Default after reset: undefined

OSD Color Definitions

R	G	B	Color
0	0	0	black
0	0	1	blue
0	1	0	green
0	1	1	cyan
1	0	0	red
1	0	1	magenta
1	1	0	yellow
1	1	1	white

Display Address Register DAR

MSB	SFR Address: $D9_H$						LSB
DAR: Display Address Register							
AV2	AV1	AV0	AH4	AH3	AH2	AH1	AH0

AV0 … 2: Row address in display RAM to be accessed next

AV2	AV1	AV0	Row
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5

AH0 … 4: Column address in display RAM to be accessed next

AH4	AH3	AH2	AH1	AH0	Column
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	2
		…			…
1	0	0	0	1	17

Default after reset: undefined

Display Data Register DDR

MSB	SFR Address: DA_H						LSB
DDR: Display Data Register							
FE	DB6	DB5	DB4	DB3	DB2	DB1	DB0

FE: Frame Enable;
if set to “1”, text rows in non-boxed mode will be displayed with a black frame.

DB0 … 6: Character code to be written to or read from display RAM

Default after reset: undefined

Display Horizontal Control Register DHR

MSB	SFR Address: DB_H						LSB
DHR: Display Horizontal Control Register							
H5	H4	H3	H2	H1	H0	–	–

HO … 5: Horizontal offset; common for all rows of text

DHR.1: Reserved for test purposes
DHR.0: Reserved for test purposes

Values written to DHR.0/1 are ignored.

Default after reset: undefined

Display Color and Polarity Register DCPR

MSB	SFR Address: DC_H						LSB
DCPR: Display Color and Polarity Register							
FBG	BCR	BCG	BCB	–	BP	–	–

FBG: If set to 1, the selected background color will be displayed over the entire screen; if set to 0, the background color will only appear behind active rows of text if boxed mode is selected

BCR: Background color red component

BCG: Background color green component

BCB: Background color blue component

DCPR.3: reserved

BP: BLANK polarity
BP = 0: BLANK = 0 during RGB out
BP = 1: BLANK = 1 during RGB out

DCPR.1: reserved

DCPR.0: reserved

Default after reset: 00_H

Character Codes and ROM Definition

The mask programmable devices SDA 20Cxx50 contain a character ROM for 96 user definable characters. The character codes are located in the range from 20_H to $7F_H$.

Character codes 00_H to $1F_H$ generate 'invisible' characters, which can be used to separate text in a single row. Leading blanks are to be filled with a user defined ‘blank’ character.

User Defined Character Sets

For applications using the mask-programmable program memory and character ROM, an application specific character set has to be defined.

For this purpose, a character definition file (standard ASCII format) has to be supplied to the manufacturer. From this file, the character ROM mask will be generated. The file format is:

$CODE = aa_H

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

XXXXXXXXXXXX

$CODE = …

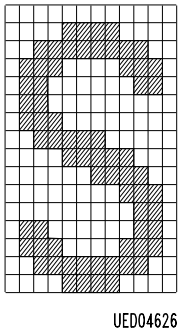
…

$END

(aa = 20 ... 7F; x = 0,1)

Empty lines or lines beginning with '#', '/', ';' or '*' are ignored and may be used to insert comments. '$CODE = ' and '$END' are used as keywords to mark the beginning of a character definition and the end of all definitions. An example for one character is given in the following.

Example of a Character Definition File
Character to be coded:




```
#  ------------------------------------
#  Example character definition file
#  ------------------------------------

$CODE = 20H
…
$CODE = 53H

000000000000
000011110000
001111111100
011100001110
011000000110
011000000000
011100000000
001111100000
000011111000
000000111100
000000001110
000000000110
011000000110
011100001110
001111111100
000011110000
…
$CODE = 7FH
…
$END
```

Character Definition File (ASCII format)

2.10 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 serial port can operate in 4 modes:

Mode 0: Serial data enters and exits through RxD (P3.6). TxD (P3.7) outputs the shift clock at 1/12 of the oscillator frequency.

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

Mode 2: 11 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmission, the 9th data bit (TB8 in SCON) can be assigned the value of 0 or 1. Or, for example, the parity bit (P, in the PSW) could be moved into TB8. On reception, the 9th data bit goes into RB8 in the special function register SCON, while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency.

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

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
SM0 SM1	SCON.7 SCON.6	Serial port mode selection, **see table 3**.
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.

Figure 40
Serial Port Control Register SCON (98_H)

Table 3
Serial Port Mode Selection

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift reg.	$f_{osc}/12$
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	$f_{osc}/64 – f_{osc}/32$
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 40**.

2.10.1 Multiprocessor Communication

Modes 2 and 3 of the serial interface of the controller have a special provision for multi-processor 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.

2.10.2 Baud Rates

The baud rate in mode 0 is fixed:

$$\text{Mode 0 baud rate} = \frac{\text{oscillator frequency}}{12}$$

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

$$\text{Mode 2 baud rate} = \frac{2^{SMOD}}{64} \times \text{osc. frequency}$$

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:

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

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:

$$\text{Modes 1, 3 baud rate} = \frac{2^{SMOD}}{32} \times \frac{\text{oscillator frequency}}{12 \times (256\text{-TH1})}$$

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 4 lists various commonly used baud rates and how they can be obtained from timer 1.

Table 4
Generated Commonly Used Baud Rates

Baud Rate		f_{osc} MHz	SMOD	Timer 1		
				CT	Mode	Reload Value
Mode 0 max.:	1.33 MHz	16.0	X	X	X	X
Mode 2 max.:	500 Kbaud	16.0	1	X	X	X
Mode 1, 3:	62.5 Kbaud	12.0	1	0	2	FF_H
	19.2 Kbaud	11.059	1	0	2	FD_H
	9.6 Kbaud	11.059	0	0	2	FD_H
	4.8 Kbaud	11.059	0	0	2	FA_H
	2.4 Kbaud	11.059	0	0	2	$F4_H$
	1.2 Kbaud	11.059	0	0	2	$E8_H$
	137.5 Baud	11.986	0	0	2	$1D_H$
	110 Baud	6.0	0	0	2	72_H
	110 Baud	12.0	0	0	1	$FEEB_H$

2.10.3 More about Mode 0

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

Figure 41 shows a simplified functional diagram of the serial port in mode 0, and associated timing.

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

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

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

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

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

As data bits come in from the right, 1s shift out to the left. When the 0 that was initially loaded into the farthest right position arrives at the farthest left 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.

2.10.4 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 42 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 devide-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 $1FF_H$ 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, 1s shift out to the left. When the start bit arrives at the farthest left 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.

2.10.5 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.

Figures 43 and 44 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 deactive $\overline{\text{SEND}}$ 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, 1s shift out to the left. When the start bit arrives at the farthest left 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.

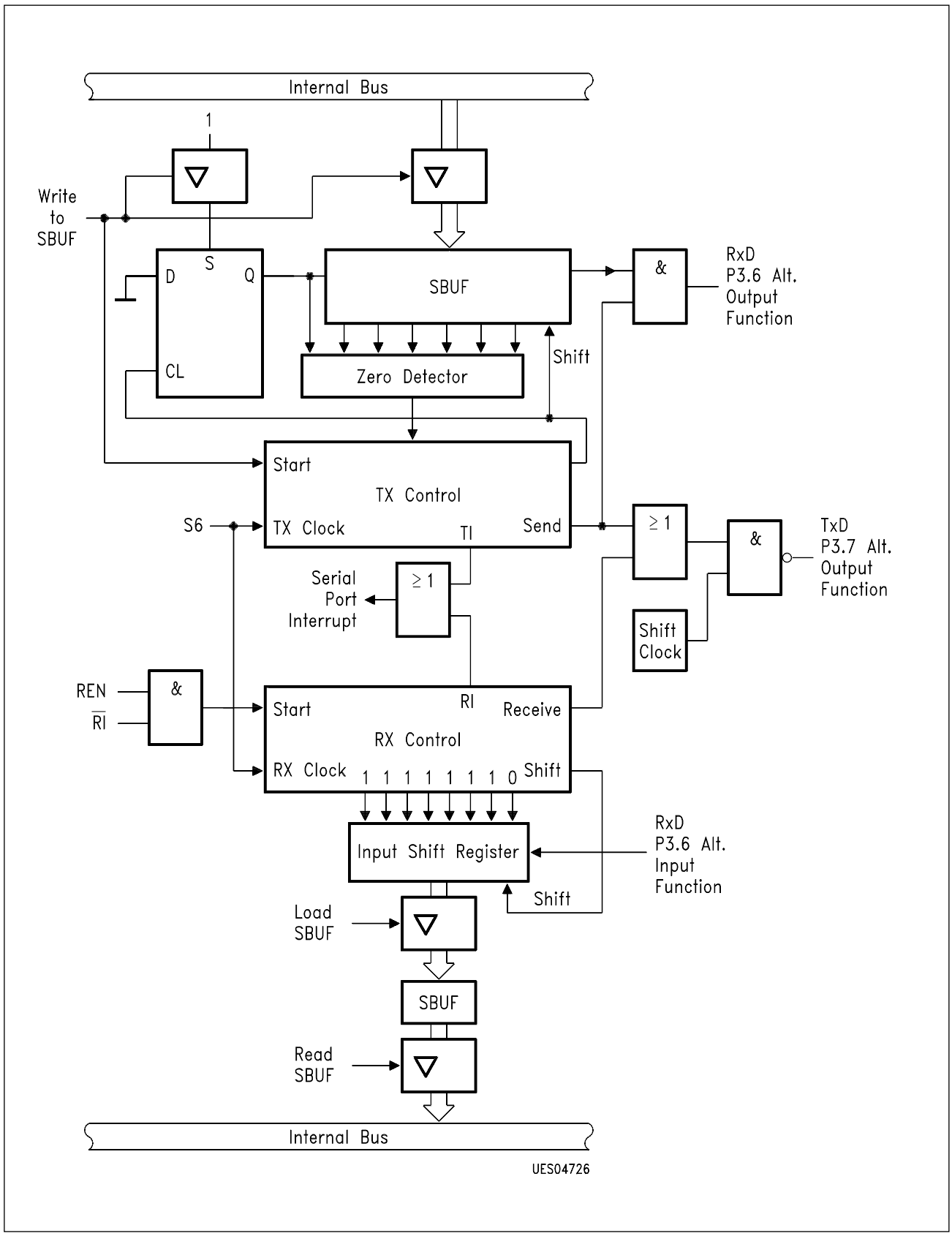


Figure 41a
Serial Port Mode 0, Functional Diagram

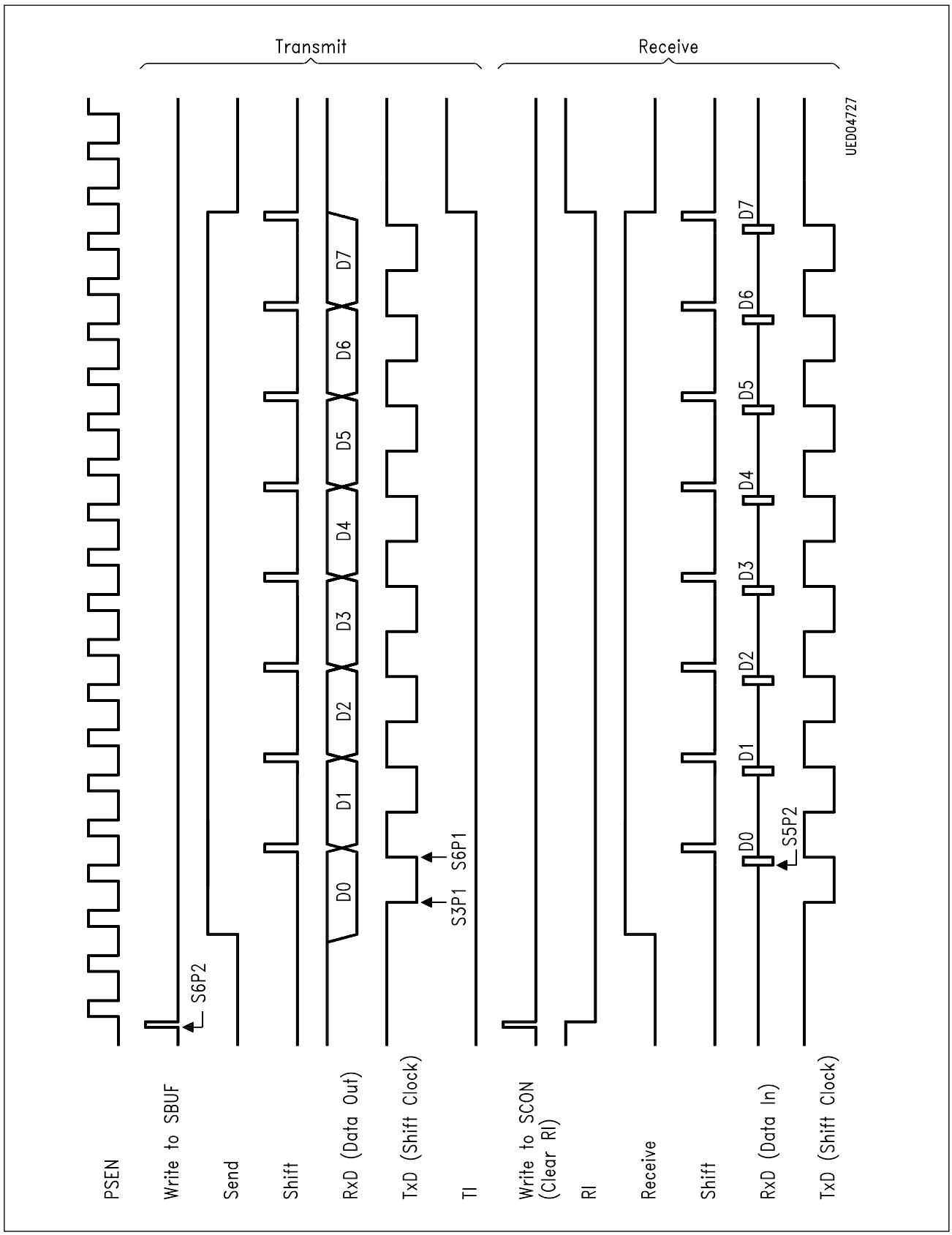


Figure 41b
Serial Port Mode 0, Timing

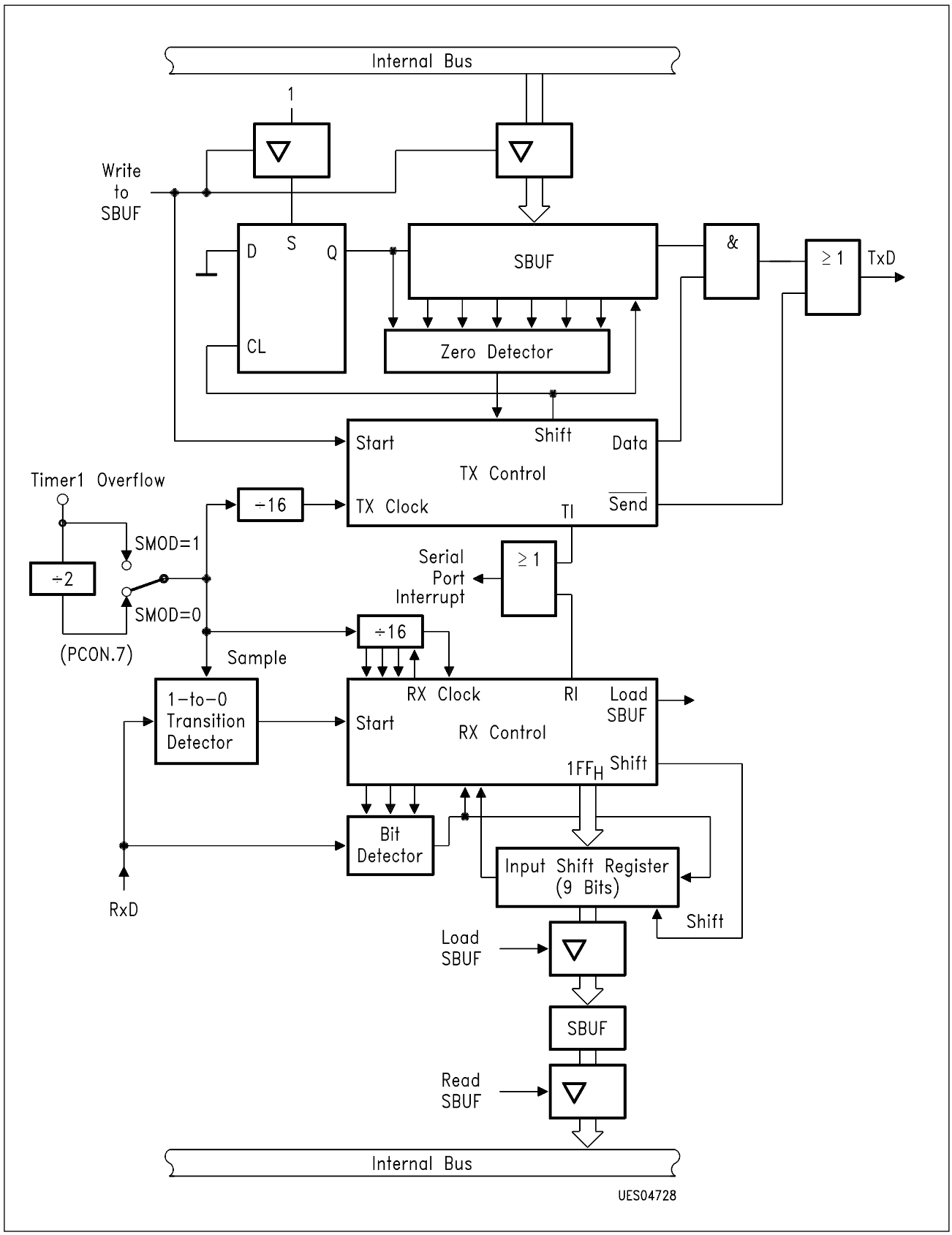


Figure 42a
Serial Port Mode 1, Functional Diagram

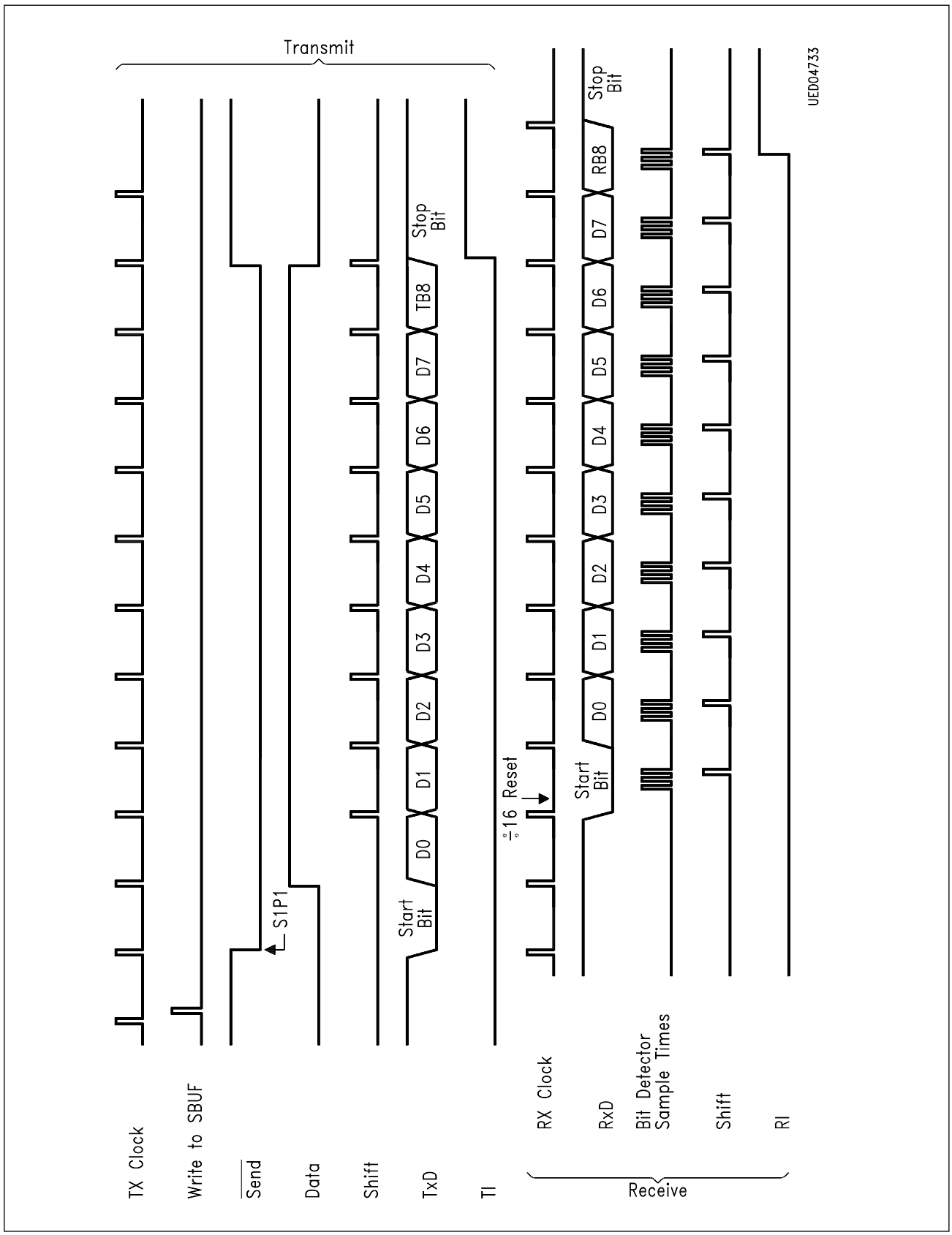


Figure 42b
Serial Port Mode 1, Timing

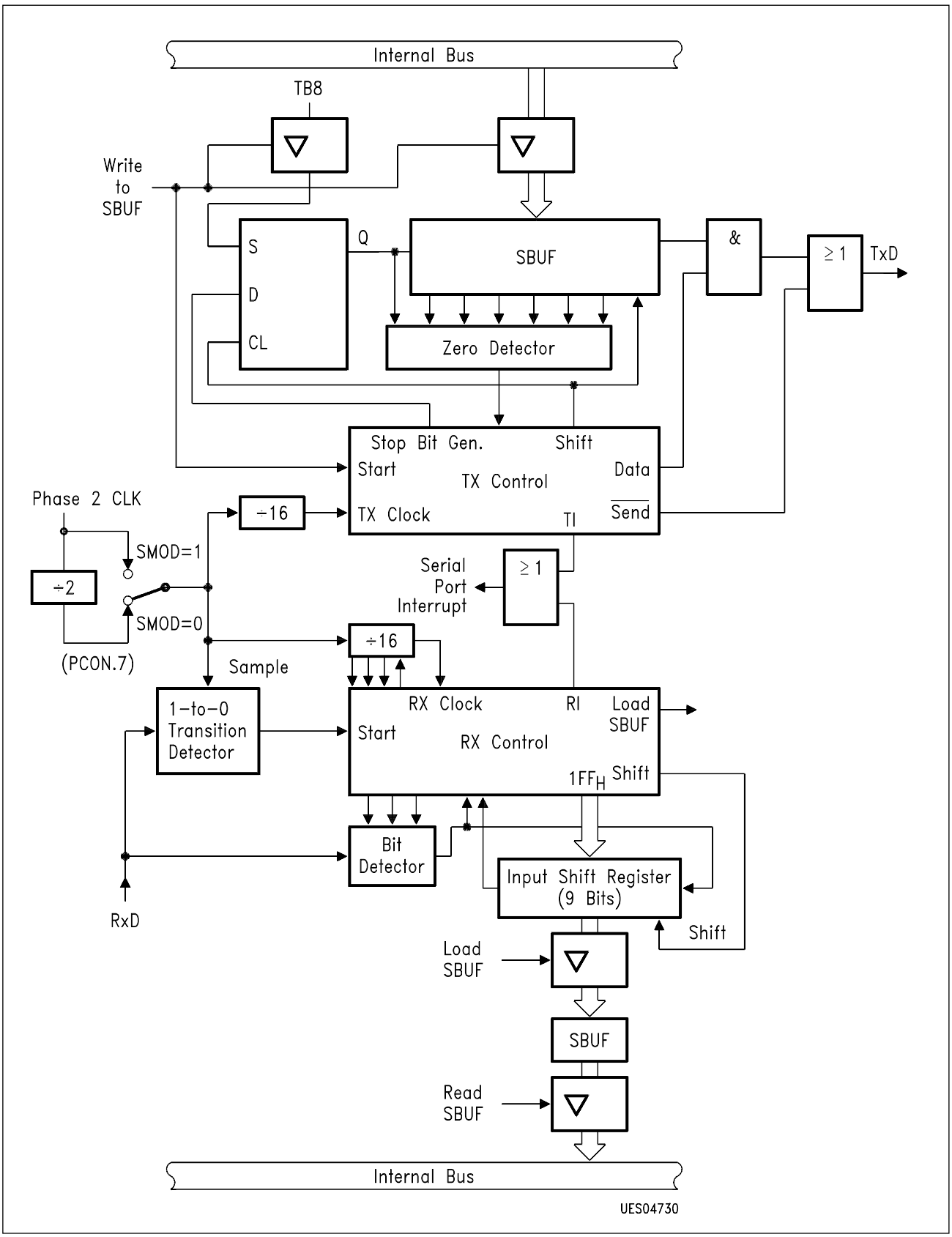


Figure 43a
Serial Port Mode 2, Functional Diagram

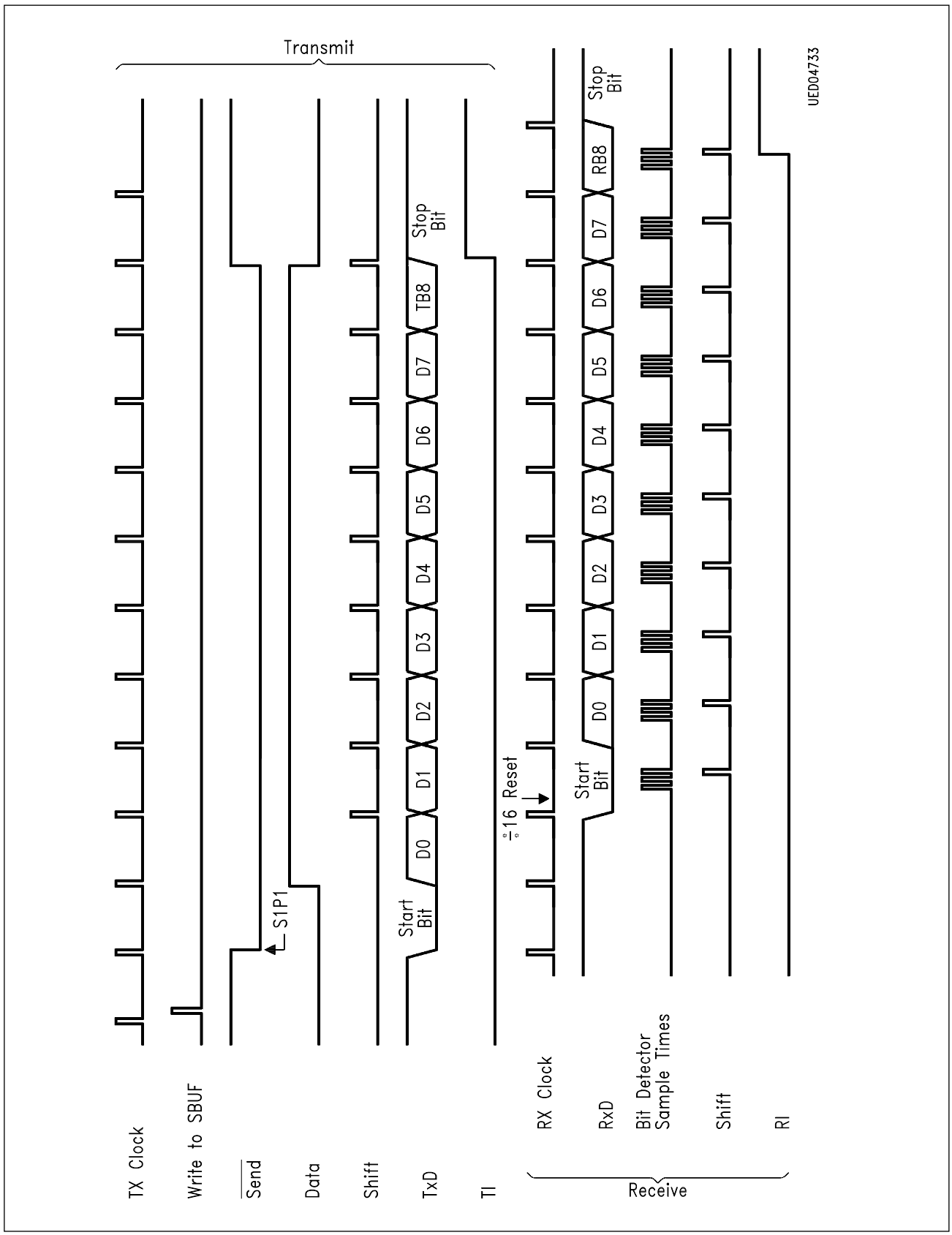


Figure 43b
Serial Port Mode 2, Timing

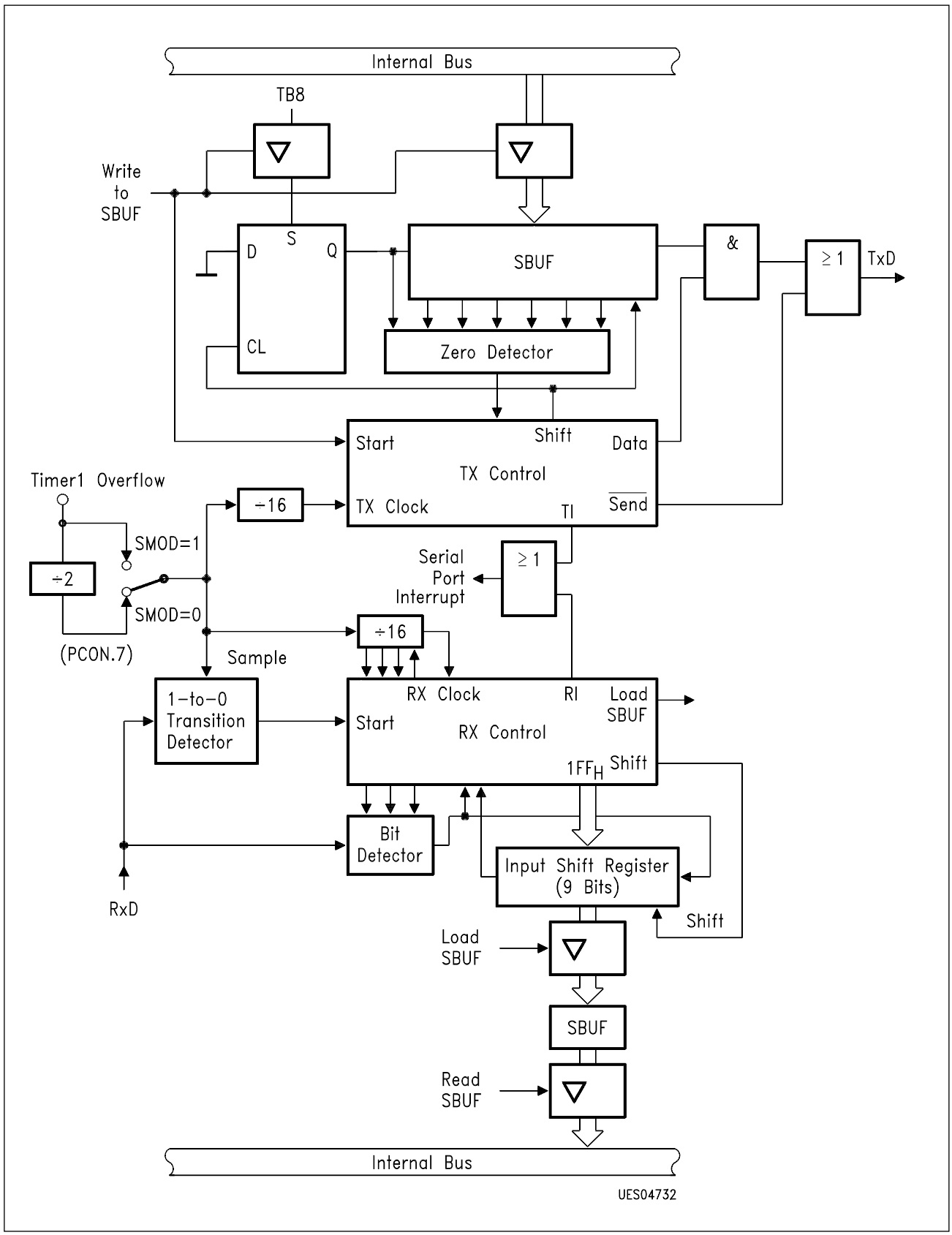


Figure 44a
Serial Port Mode 3, Functional Diagram

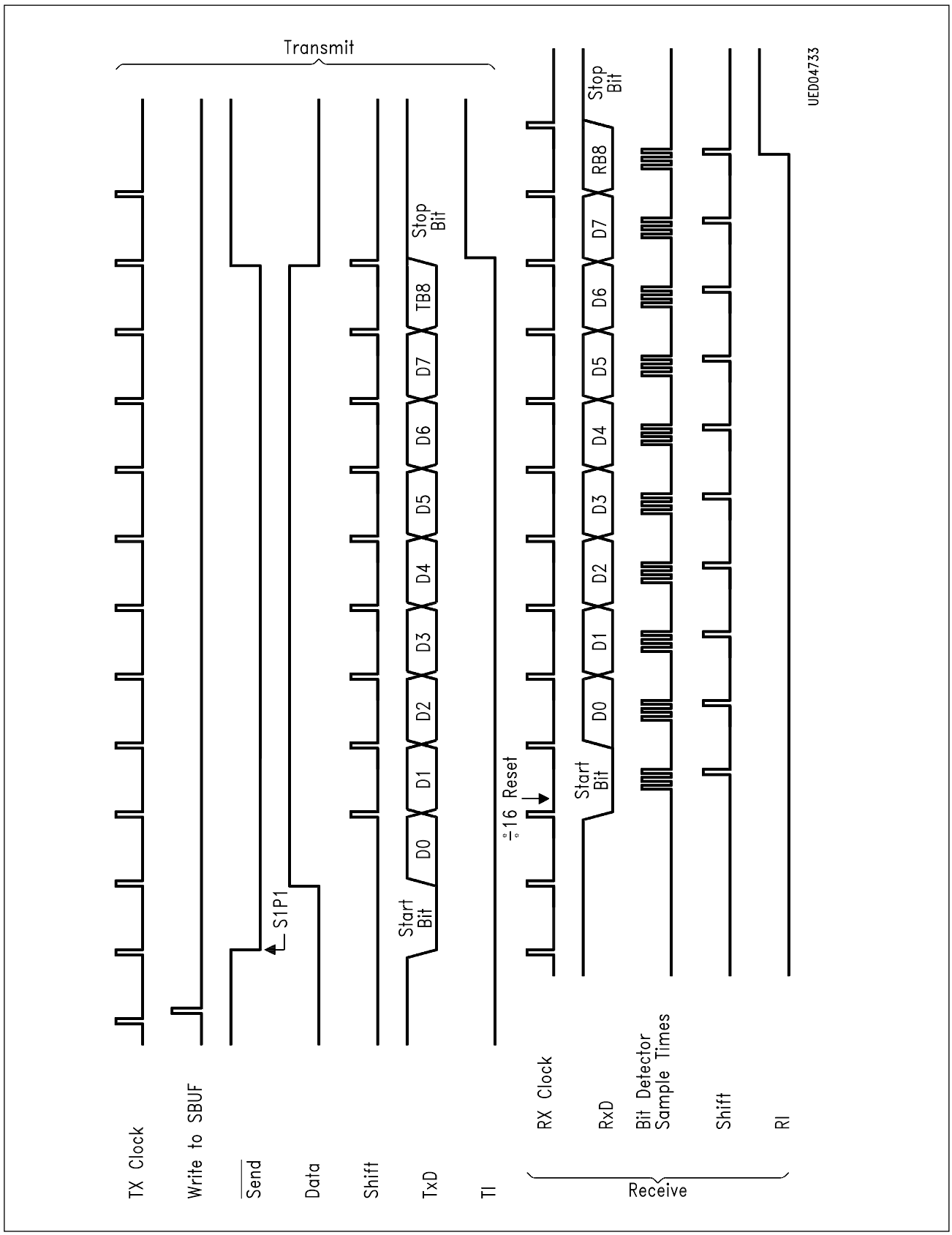


Figure 44b
Serial Port Mode 3, Timing

2.11 Advanced Function Register

The on-chip clock generator of the SDA 20Cxx50 contains the same clock divider, found in every 8051 compatible design. The clock divider divides the external clock frequency (oscillator frequency) by 2. To enhance clock performance by either doubling the internal clock frequency or by keeping the internal frequency constant and halving the external quartz-frequency, the divider can be switched off by software. As software-switch for the divider a new Special-Function-Register (SFR) has been defined:

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

Default after reset: FF_H

(MSB) (LSB)

CDC	**1**	**1**	**1**	**1**	**1**	**1**	**1**

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

AFR.0 – AFR.6 Reserved, always to be written with ‘1’.

Note:

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

Note:

All time values given in this specification apply for CDC = 1 only (clock divider is on).

2.12 Instruction Set

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

2.12.1 Notes on Data Addressing Modes

Rn – Working register R0-R7.

direct – 128 internal RAM locations, any I/O port, control or status register.

@Ri – Indirect internal RAM location addressed by register R0 or R1.

#data – 8-bit constant included in instruction.

#data 16 – 16-bit constant included as bytes 2 & 3 of instruction.

bit – 128 software flags, any l/O pin, control or status bit in special function registers.

Operations working on external data memory (MOVX ...) are used to access the additional bytes of the extended internal data RAM (128/256 bytes for SDA 20C2450/SDA20C3250).

2.12.2 Notes on Program Addressing Modes

addr 16 – Destination address for LCALL & LJMP may be anywhere within the program memory address space.

addr 11 – Destination address for ACALL & AJMP will be within the same 2 Kbyte of the following instruction.

rel – SJMP and all conditional jumps include an 8-bit offset byte. Range is + 127/– 128 bytes relative to first byte of the following instruction.

2.12.3 Instruction Set Description

Arithmetic Operations

Mnemonic		Description	Byte
ADD	A, Rn	Add register to accumulator	1
ADD	A, direct	Add direct byte to accumulator	2
ADD	A, @Ri	Add indirect RAM to accumulator	1
ADD	A, #data	Add immediate data to accumulator	2
ADDC	A, Rn	Add register to accumulator with carry flag	1
ADDC	A, direct	Add direct byte to A with carry flag	2
ADDC	A, @Ri	Add indirect RAM to A with carry flag	1
ADDC	A, #data	Add immediate data to A with carry flag	2
SUBB	A, Rn	Subtract register from A with borrow	1
SUBB	A, direct	Subtract direct byte from A with borrow	2
SUBB	A, @Ri	Subtract indirect RAM from A with borrow	1
SUBB	A, #data	Subtract immediate data from A with borrow	2
INC	A	Increment accumulator	1
INC	Rn	Increment register	1
INC	direct	Increment direct byte	2
INC	@Ri	Increment indirect RAM	1
DEC	A	Decrement accumulator	1
DEC	Rn	Decrement register	1
DEC	direct	Decrement direct byte	2
DEC	@Ri	Decrement indirect RAM	1
INC	DPTR	Increment data pointer	1
MUL	AB	Multiply A & B	1
DIV	AB	Divide A & B	1
DA	A	Decimal adjust accumulator	1

Logical Operations

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

Data Transfer Operations

Mnemonic		Description	Byte
MOV	A, Rn	Move register to accumulator	1
MOV	A, direct	Move direct byte to accumulator	2
MOV	A, @Ri	Move indirect RAM to accumulator	1
MOV	A, #data	Move immediate data to accumulator	2
MOV	Rn, A	Move accumulator to register	1
MOV	Rn, direct	Move direct byte to register	2
MOV	Rn, #data	Move immediate data to register	2
MOV	direct, A	Move accumulator to direct byte	2
MOV	direct, Rn	Move register to direct byte	2
MOV	direct, direct	Move direct byte to direct	3
MOV	direct, @Ri	Move indirect RAM to direct byte	2
MOV	direct, #data	Move immediate data to direct byte	3
MOV	@Ri, A	Move accumulator to indirect RAM	1
MOV	@Ri, direct	Move direct byte to indirect RAM	2
MOV	@Ri, #data	Move immediate data to indirect RAM	2
MOV	DPTR, #data16	Load data pointer with a 16-bit constant	3
MOVC	A@A + DPTR	Move code byte relative to DPTR to accumulator	1
MOVC	A@A + PC	Move code byte relative to PC to accumulator	1
MOVX	A, @Ri	Move external RAM (8-bit addr) to accumulator	1
MOVX	A, @DPTR	Move external RAM (16-bit addr) to accumulator	1
MOVX	@Ri, A	Move A to external RAM (8-bit addr)	1
MOVX	@DPTR, A	Move A to external RAM (16-bit addr)	1
PUSH	direct	Push direct byte onto stack	2
POP	direct	Push direct byte from stack	2
XCH	A, Rn	Exchange register with accumulator	1
XCH	A, direct	Exchange direct byte with accumulator	2
XCH	A, @Ri	Exchange indirect RAM with accumulator	1
XCHD	A, @Ri	Exchange low-order digital indirect RAM with A	1

Boolean Variable Manipulation

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

Program and Machine Control Operations

Mnemonic		Description	Byte
ACALL	addr11	Absolute subroutine call	2
LCALL	addr16	Long subroutine call	3
RET		Return from subroutine	1
RETI		Return from interrupt	1
AJMP	addr 11	Absolute jump	2
LJMP	addr 16	Long jump	3
SJMP	rel	Short jump (relative addr)	2
JMP	@A + DPTR	Jump indirect relative to the DPTR	1
JZ	rel	Jump if accumulator is zero	2
JNZ	rel	Jump if accumulator is not zero	2
JC	rel	Jump if carry flag is set	2
JNC	rel	Jump if carry flag is not set	2
JB	bit, rel	Jump if direct bit set	3
JNB	bit, rel	Jump if direct bit not set	3
JBC	bit, rel	Jump if direct bit is set and clear bit	3
CJNE	A, direct, rel	Compare direct to A and jump if not equal	3
CJNE	A, #data, rel	Compare immediate to A and jump if not equal	3
CJNE	Rn, #data, rel	Compare immediate to register and jump if not equal	3
CJNE	@Ri, #data, rel	Compare immediate to indirect and jump if not equal	3
DJNZ	Rn, rel	Decrement register and jump if not zero	2
DJNZ	direct, rel	Decrement direct and jump if not zero	3
NOP		No operation	1

2.12.4 Instruction Opcodes in Hexadecimal Order

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

Hex Code	Number of Bytes	Mnemonic	Operands
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, R7
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
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

Hex Code	Number of Bytes	Mnemonic	Operands
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
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

Hex Code	Number of Bytes	Mnemonic	Operands
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
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

Hex Code	Number of Bytes	Mnemonic	Operands
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
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

Hex Code	Number of Bytes	Mnemonic	Operands
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
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

Absolute Maximum Ratings

Parameter	Symbol	Limit Values	Unit
Voltage on any pin with respect to ground	V_S	– 0.5 to 7	V
Power dissipation	P_{tot}	1	W
Ambient temperature under bias	T_A	0 to 85	°C
Storage temperature	T_{stg}	– 65 to 125	°C

DC Characteristics

T_A = 0 to 85 °C; V_{DD} = 5 V ± 10 %, V_{SS} = 0 V

Parameter	Pins	Symbol	Limit Values		Unit	Test Conditions
			min.	max.		
Power supply voltage	V_{DD}	V_{DD}	4.5	5.5	V	f = 12 MHz
Power supply current	V_{DD}	I_{DD}		30	mA	V_{DD} = 5.5 V
Power down current	V_{DD}	I_{PD}		10	µA	
Input levels standard schmitt trigger	PORT 0 PORT 1 PORT 3	V_{IL1}		$0.2 \times V_{DD}$ – 100 mV	mV	V_{DD} = 5.5 V
	PORT 4 PORT 5 RST	V_{IH1}	$0.2 \times V_{DD}$	900	mV	
Input levels clock pins	XTAL1	V_{IL2}		$0.3 \times V_{DD}$	mV	V_{DD} = 4.5 V
	LC1	V_{IH2}	$0.7 \times V_{DD}$		mV	
Input levels sandcastle decoder	SC	V_{SCL}		$0.1 \times V_{DD}$	mV	V_{DD} = 4.5 V
		V_{SCM}	$0.1 \times V_{DD}$ + 1 V	$0.45 \times V_{DD}$	mV	
		V_{SCH}	$0.7 \times V_{DD}$		mV	
Leakage current at floating/input state	PORT 0 PORT 4 XTAL2 LC1 LC2 SC	I_{Leak}	– 10	10	µA	V_{DD} = 5.5 V V_{IL} = 0 V V_{IH} = 5.5 V

DC Characteristics (cont'd)

T_A = 0 to 85 °C; V_{DD} = 5 V ± 10 %, V_{SS} = 0 V

Parameter	Pins	Symbol	Limit Values		Unit	Test Conditions
			min.	max.		
Input current through standard pullup resistor	PORT 1 PORT 3	I_{IL1}	– 200		µA	V_{DD} = 5.5 V V_{IL} = 0 V
	PORT 5 RST	I_{IH1}		– 20	µA	V_{DD} = 4.5 V V_{IH} = 1.8 V
Low level output voltages of standard outputs	PORT 0 PORT 1 PORT 3 PORT 4 OSD	V_{OL1}		450	mV	V_{DD} = 4.5 V I_{OL} = 3.2 mA
High level output voltages of standard outputs	PORT 1 P 3.6 P 3.7 OSD	V_{OH1}	V_{DD} – 1 V			V_{DD} = 4.5 V I_{OH} = – 3.2 mA
Output levels oscillators	XTAL2 LC2	V_{OL2}		1.0	V	V_{DD} = 5.5 V I_{OL} = 500 µA
		V_{OH2}	V_{DD} – 1 V		V	V_{DD} = 4.5 V I_{OH} = – 500 µA

AC Characteristics
External Clock Drive XTAL1 / Quartz Clock Drive XTAL1-XTAL2

Parameter	Symbol	Limit Values Variable Clock 1.2 MHz - 12MHz		Unit
		min.	max.	
Oscillator cycle time	t_{CLCL}	83.3	833	ns
High time	t_{CHCX}	20	$t_{CLCL} - t_{CLCX}$	ns
Low time	t_{CLCX}	20	$t_{CLCL} - t_{CHCX}$	ns
Rise time	t_{CLCH}	–	20	ns
Fall time	t_{CHCL}	–	20	ns
External quartz frequency	f_Q	1.2	12	MHz

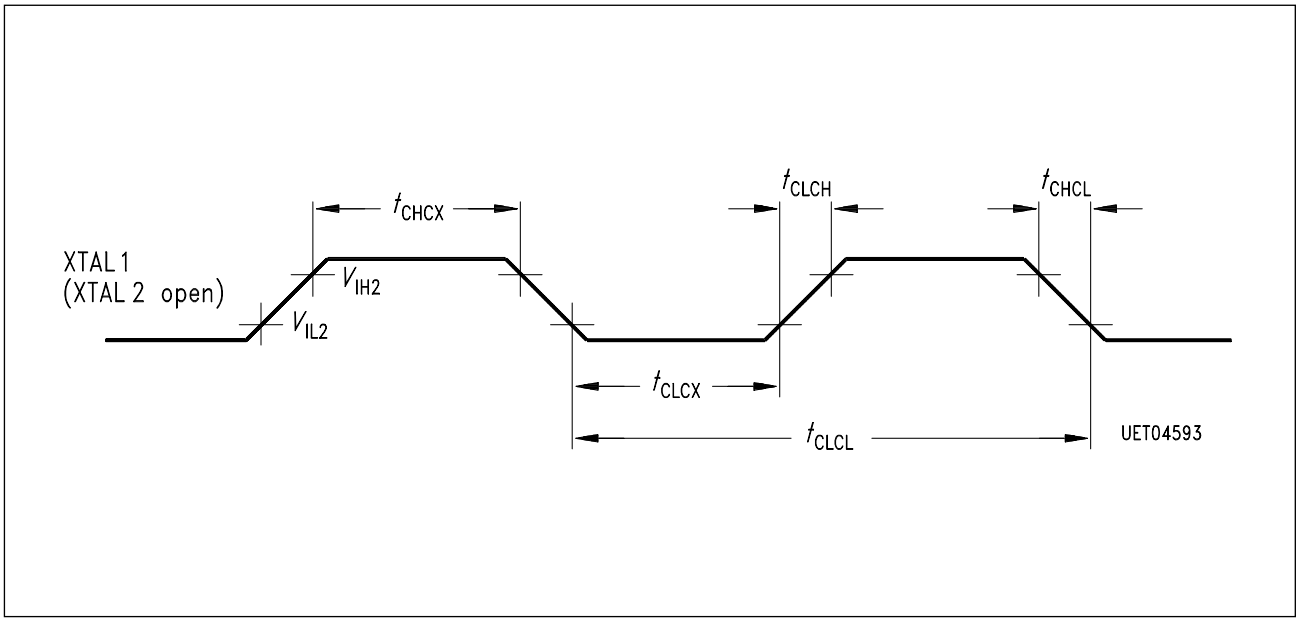


Figure 45
External Clock Cycle

OSD Input/Output Timing Characteristics

Parameter	Symbol	Limit Values Variable DOT Clock f_{DOT} = 6 MHz to 8 MHz		Unit
		min.	max.	
L-sandcastle time	t_{SCL}	15		µs
H-sandcastle time	t_{SCH}	3		µs
Horizontal offset	t_{HO}	t_{SCH}		µs
Pixel width	t_{DOT}	125	166	ns
Line width (= 216 × t_{DOT})	t_{LINE}	27	36	µs

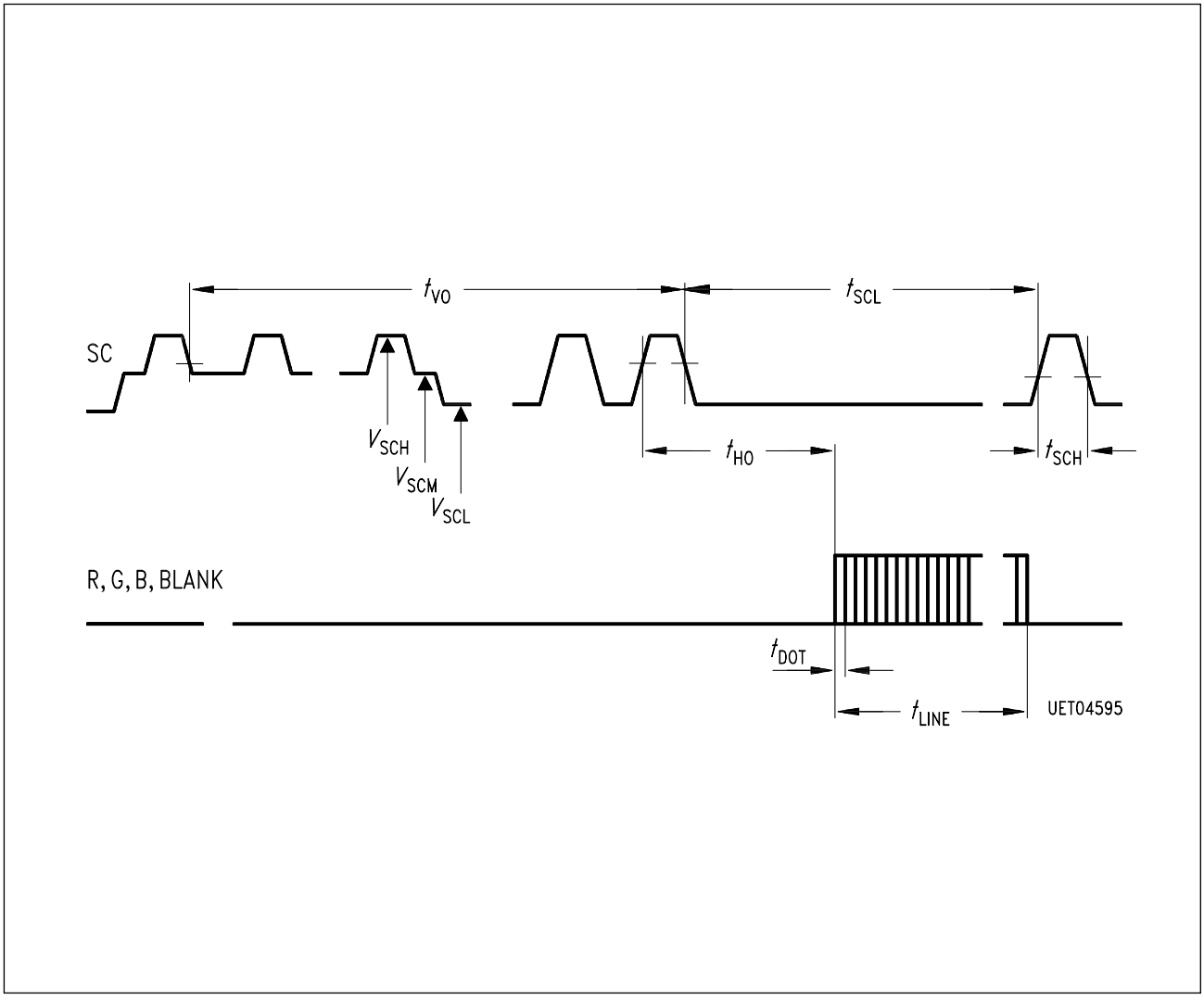


Figure 46
OSD Input/Output Timing

Capture Input Timing Characteristics

Parameter	Symbol	Limit Values Variable Clock Freq. up to 16 MHz		Unit
		min.	max.	
L-capture input time	t_{CAPL}	12 t_{CLCL}	–	ns
H-capture input time	t_{CAPH}	12 t_{CLCL}	–	ns

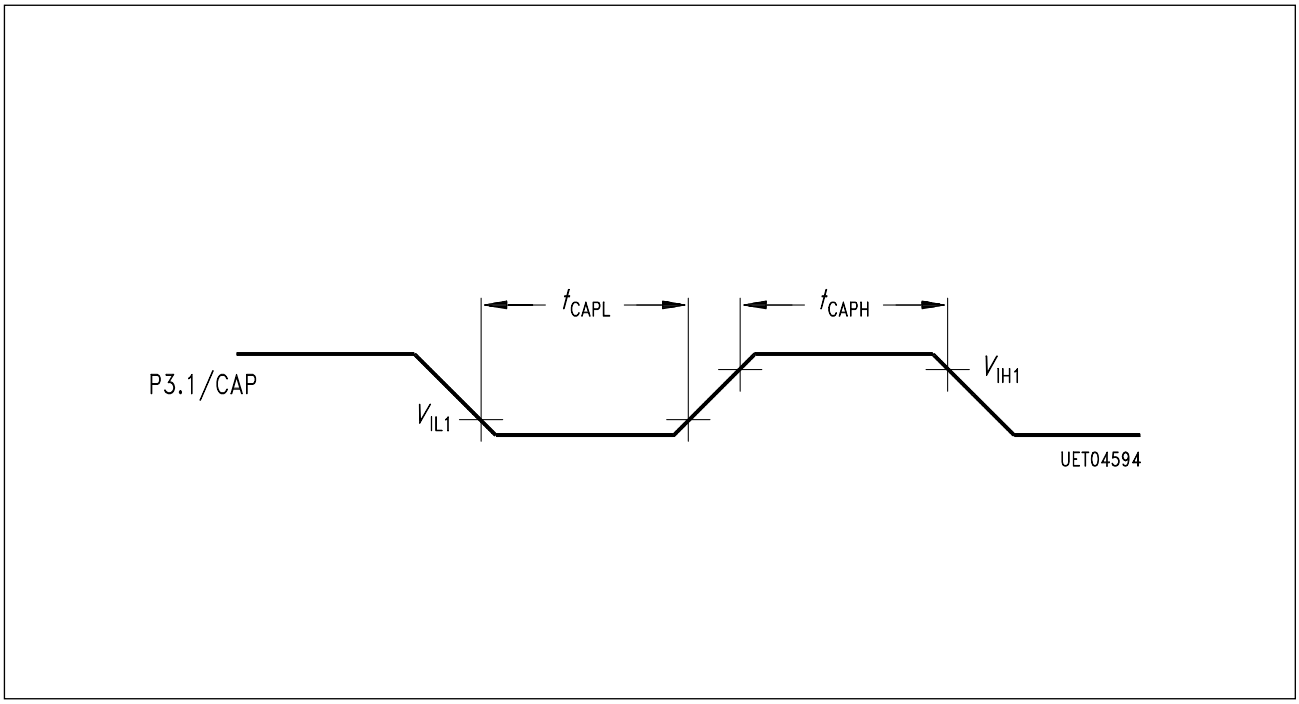
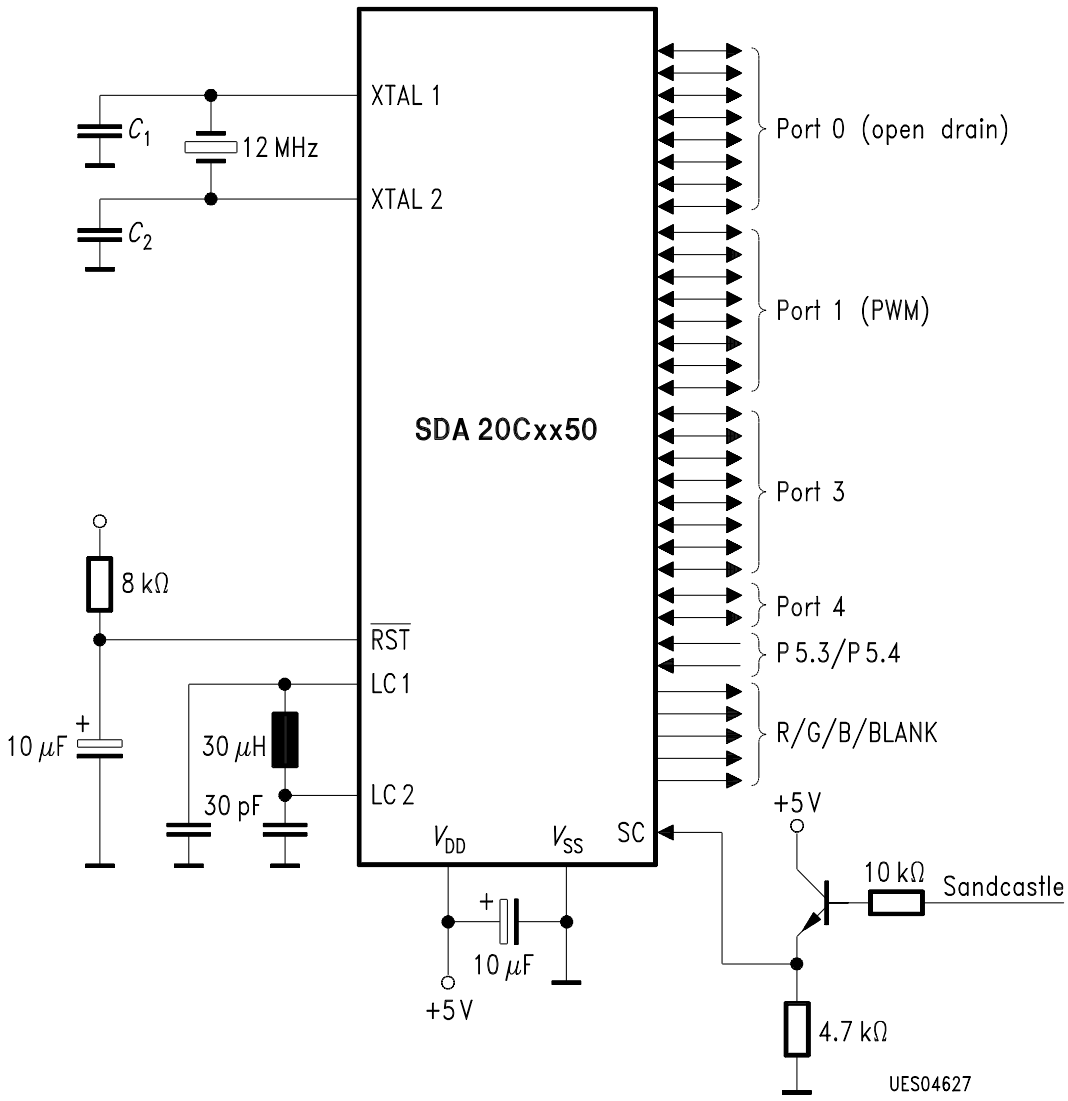


Figure 47
Capture Input Timing

4 Application Example



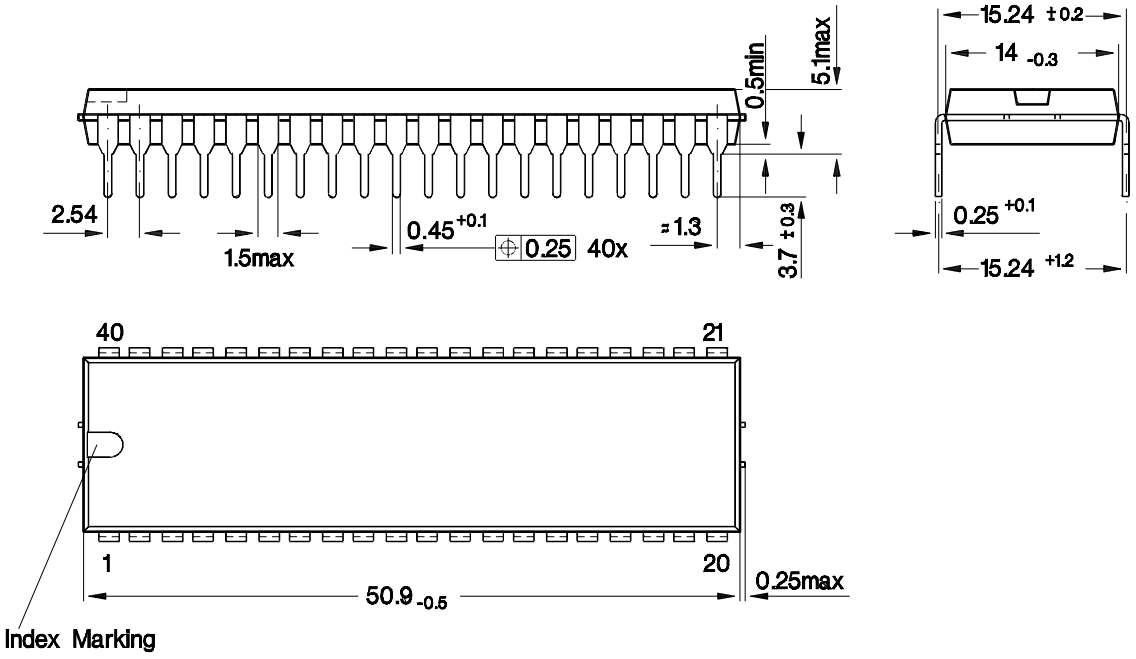
Recommended Capacitance Values

	Quartz Resonator	Ceramic Resonator	Unit
C_1	30 ± 10	40 ± 10	pF
C_2	30 ± 10	40 ± 10	pF

Figure 49
Color OSD Application

5 Package Outlines

Plastic Package, P-DIP-40
(Dual-in-Line)



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

Dimensions in mm