

Table of Contents

1.	Introduction	
	MAX765x Features	1 - 1
	Performance Overview	1 - 2
	Software Compatibility	1 - 3
	803x/805x Comparision	1 - 4
2.	MAX765x Architectural Overview	
	Block Diagram	2 - 2
	Pinout	2 - 3
	Signal Descriptions	2 - 4
	Memory Organization	2 - 6
	Program Memory	2 - 7
	Internal FLASH Memory	2 - 8
	External ROM	2 - 8
	External RAM	2 - 8
	Internal RAM	2 - 9
	Instruction Set	2 - 11
	Instruction Timing	2 - 17
	CPU Timing	2 - 18
	Stretch Memory Cycles	2 - 19
	Dual Data Pointers	2 - 21
	Special Function Registers Overview	2 - 22

3. MAX765x Hardware Description

Timers/Counters	
Timers 0 & 1	3 - 2
Mode 0	3 - 3
Mode 1	3 - 6
Mode 2	3 - 7
Mode 3	3 - 8
Timer Rate Control	3 - 9
Timer 2	
Timer 2 Mode Control	3 - 12
16-Bit Mode with Capture	3 - 14
16-Bit Mode with Auto-Reload .	3 - 15
Baud Rate Generator Mode	3 - 16
Serial Interface	
Mode 0	3 - 19
Mode 1	
Mode 1 Baud Rate	3 - 25
Mode 1 Transmit	3 - 29
Mode 1 Receive	3 - 29
Mode 2	
Mode 2 Transmit	3 - 32
Mode 2 Receive	3 - 33
Mode 3	3 - 35
Multiprocessor Communications	3 - 37
A/D Converter Registers	3 - 38
Watchdog Timer/Registers	3 - 42
FLASH Programming Registers	3 - 46
Pulse-Width Modulators	3 - 49
SFR Registers	3 - 53

Interrupts	3 - 54
803x/805x Compatibility	3 - 55
Interrupt SFR Registers	3 - 56
Interrupt Processing	3 - 60
Interrupt Masking	3 - 61
Interrupt Priorities	3 - 62
Interrupt Sampling	3 - 64
Interrupt Latency	3 - 65
Single-Step Operation	3 - 65
Reset	
Power-On Reset	3 - 66
Power Saving Modes	
Idle	3 - 68
Stop	3 - 70
4. MAX765x Interfacing	
External Memory	4 - 2
Adding Stretch Cycles	4 - 4
FLASH Programming	4 - 10

1

Maxim MAX765x Features

The MAX765x is a complete 12-bit data acquisition system combined with an industry-standard 8051 microprocessor core. Features include:

- 4-clock instruction cycle
- Space-saving 64-pin TQFP package
- DC to 12Mhz clock speed
- 8 Analog Input Channels – 50ksps @ 12-bit resolution
- Dual 8K-byte FLASH memories
- Dual 8-bit PWM outputs
- Operates from +2.7V (MAX7652) to +5.5V (MAX7651)
- Dual data pointers
- Programmable Watchdog Timer

Performance Overview

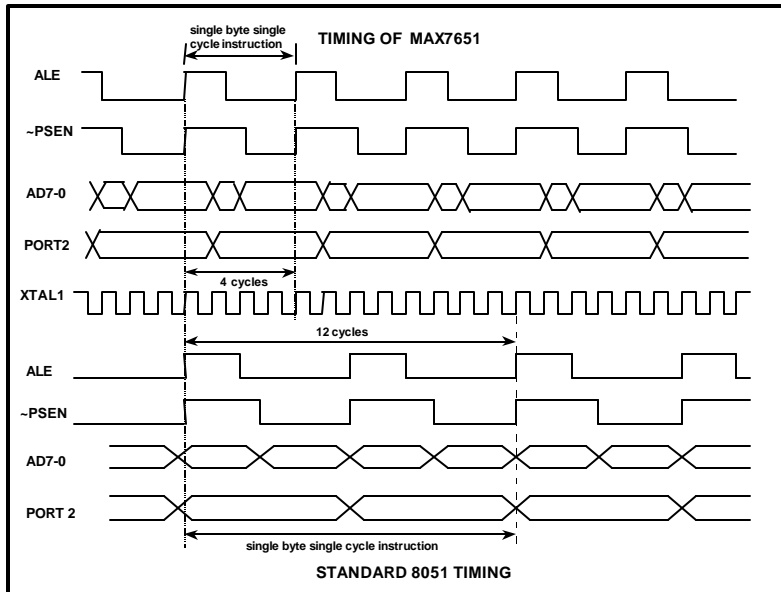
The MAX765x processor core provides increased performance by executing instructions in a 4-clock bus cycle, as opposed to the 12-clock bus cycle in the standard 8051 (see Figure 1-1). The shortened bus timing improves the instruction execution rate (for most instructions) by a factor of three over the standard 8051 architectures.

Some instructions require a different number of instruction cycles on the MAX765x than they do on the standard 8051. In the standard 8051, all instructions except for `MUL` and `DIV` take one or two instruction cycles to complete. In the MAX765x architecture, instructions can take from one to five instruction cycles to complete.

The average speed improvement for the entire instruction set is approximately 2.5X, calculated as follows:

Number of Opcodes	Speed Improvement
150	3.0X
51	1.5X
43	2.0X
2	2.4X
Total: 246	Average: 2.5X
Note: Comparison is for MAX765x and standard 8051 running at the same clock frequency.	

Figure 1 – 1: Comparative Timing of MAX765x and Standard 8051



Software Compatibility

The MAX765x is object code compatible with the industry standard 8051 microcontroller. That is, object code compiled with an industry standard 8051 compiler or assembler will execute on the MAX765x and be functionally equivalent. However, because the MAX765x uses a different instruction timing than the standard 8051, code with timing loops may require modification.

The “Instruction Set” section in Chapter 2 of this databook lists the number of instruction cycles required to perform each instruction on the MAX765x.

803x/805x Feature Comparison

Table 1-1 provides a feature-by-feature comparison of the MAX765x and several common 803x/805x configurations. Similar hardware feature comparisons appear in the detailed hardware descriptions throughout this databook.

Table 1-1: Feature Summary of MAX765x and 803x/805x Types

Feature	Industry-Standard				Maxim MAX765x
	8031	8051	80C32	80C52	
Clocks per instruction cycle	12	12	12	12	4
Internal ROM *		4 KB		8 KB	16K
Internal RAM	128 bytes	128 bytes	256 bytes	256 bytes	256 bytes
Data Pointers	1	1	1	1	2
Serial Ports	1	1	1	1	2
16-bit Timers	2	2	3	3	3
Interrupt sources (internal and external)	5	5	6	6	7
Stretch memory cycles	no	no	no	no	yes
Internal watchdog timer	no	no	no	no	yes

* MAX765x contains two banks of 8K FLASH.

2

This chapter provides a technical overview and a description of the MAX765x architecture. The following topics discussed are:

- Input/Output Signals
- RAM and ROM Memory
- Instruction Set

Figure 2-1 illustrates the MAX765x internal architecture. Table 2-1 describes the function of each signal.

Figure 2-1: MAX765x Block Diagram

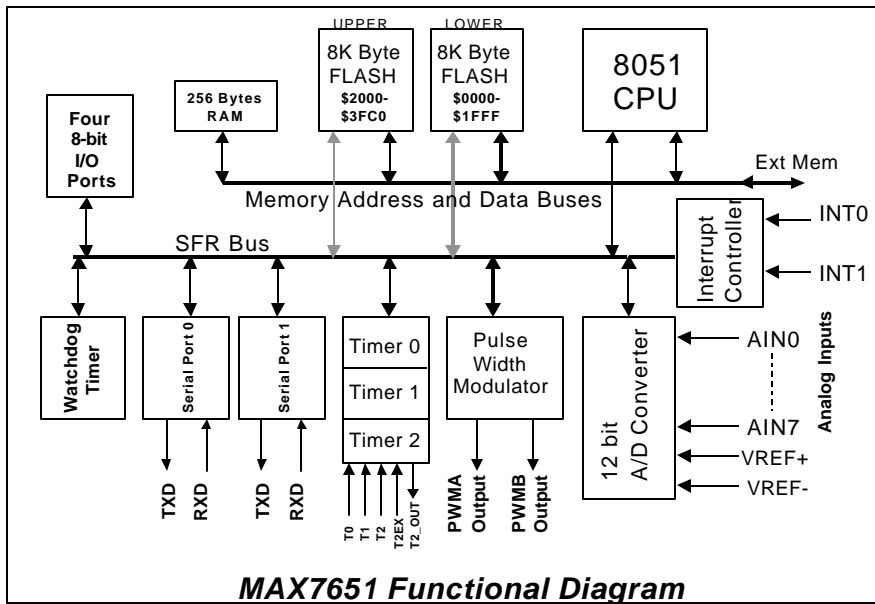
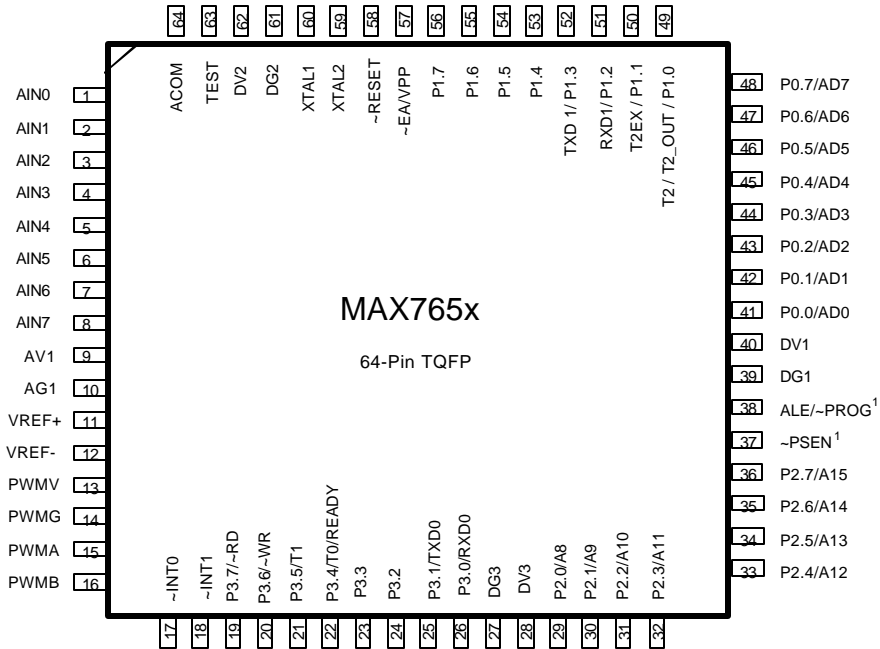


Figure 2-2: MAX765x Pinout



NOTES-

1. ALE and ~PSEN are weakly pulled HIGH (10K ohm) during RST assertion. To insure FLASH data integrity R_{oad} on ALE or ~PSEN must be greater than 200K ohm to ensure a valid HIGH level on ALE and ~PSEN during RST assertion. Additionally, RST must follow DV during power-up.
2. TEST pin is always grounded.
3. Signals which are active LOW are indicated by a tilde (~) in front of the signal name.

Table 2-1: Signal Descriptions

Pin Name	Pin #	Type	Function
AIN0 – AIN7	1 - 8	Input	Analog inputs 0 – 7.
AV1	9	Input	Analog power supply for the A/D converter.
AG1	10	Input	Analog ground for the A/D converter.
VREF+	11	Input	Positive reference for the A/D converter (VREF- to AVDD).
VREF-	12	Input	Ground reference for the A/D converter (0V to VREF+).
PWMV	13	Input	Pulse-width modulator power. Usually connected to AV1.
PWMG	14	Input	Pulse-width modulator ground. Usually connected to AG1.
PWMA, PWMB	15 – 16	Output	PWM outputs.
~INT0, ~INT1	17 - 18	Input	External interrupts 0 and 1, active LOW.
P3.7/~RD	19	I/O	Port 3, bit 7 I/O or ~RD (read strobe for accessing external data memory, active LOW).
P3.6/~WR	20	I/O	Port 3, bit 6 or ~WR (write strobe for writing to external data memory, active LOW).
P3.5/T1	21	I/O	Port 3, bit 5 I/O or Timer 1 input
P3.4/T0/READY	22	I/O	Port 3, bit 4 I/O or Timer 0 input. In external FLASH programming mode, acts as the READY line.
P3.3	23	I/O	Port 3, bit 3 I/O.
P3.2	24	I/O	Port 3, bit 2 I/O.
P3.1/TXD0	25	I/O	Port 3, bit 1 I/O or Serial Port 0 Transmit Data.
P3.0/RXD0	26	I/O	Port 3 bit 0 I/O or Serial Port 0 Receive Data.
DG3,2,1	27, 61, 39	Input	Digital ground.
DV3, 2, 1	28, 62, 40	Input	Digital power supply. 2.7V – 5.5V.
P2.0/A8	29	I/O	Port 2, bit 0 I/O or Address line 8.
P2.1/A9	30	I/O	Port 2, bit 1 I/O or Address line 9.
P2.2/A10	31	I/O	Port 2, bit 2 I/O or Address line 10.

P2.3/A11	32	I/O	Port 2, bit 3 I/O or Address line 11.
P2.4/A12	33	I/O	Port 2, bit 4 I/O or Address line 12.
P2.5/A13	34	I/O	Port 2, bit 5 I/O or Address line 13.
P2.6/A14	35	I/O	Port 2, bit 6 I/O or Address line 14.
P2.7/A15	36	I/O	Port 2, bit 7 I/O or Address line 15.
~PSEN	37	Output	Program Store Enable, active LOW. Used to qualify reads from external devices.
P0.0/AD0 – P0.7/AD7	41 – 48	I/O	Port 0, bits 0-7 I/O or multiplexed address/data bits 0-7.
P1.0/T2/T2_OUT	49	I/O	Port 1, bit 0 I/O or Timer 2 Input/Timer 2 Output
P1.1/T2EX	50	I/O	Port 1, bit 1 I/O or Timer 2 Capture/Reload Trigger.
P1.2/RXD1	51	I/O	Port 1, bit 2 I/O or Serial Port 1 Receive Data.
P1.3/TXD1	52	I/O	Port 1, bit 3 I/O or Serial Port 1 Transmit Data.
P1.4 – P1.7	53-56	I/O	Port 1, bits 4 to 7 I/O.
~EA/Vpp	57	Input	When tied LOW, enables external memory fetches. When in FLASH programming mode, is the FLASH programming voltage (+5V).
RESET	58	Input	When HIGH, resets the CPU. Weakly pulled down with approx. 120K.
XTAL2	59	Output	Crystal oscillator output.
XTAL1/CLK IN	60	Input	Crystal oscillator input. Accepts external clock input.
TEST	63	Input	CONNECT TO DIGITAL GROUND.
ACOM	64	Input	Analog Common. Used in single-ended analog input mode only.

Memory Organization

Memory organization in the MAX765x is similar to that of the industry standard 8051. There are three distinct memory areas: program memory (ROM), data memory (external RAM), and registers (internal RAM).

Figure 2-3 illustrates the memory map. Program memory and data memory can be up to 64 KB each and share the same address range (0000H–FFFFH), but are accessed differently. The internal RAM addresses overlap the lower external RAM addresses, but are accessed through different instruction types.

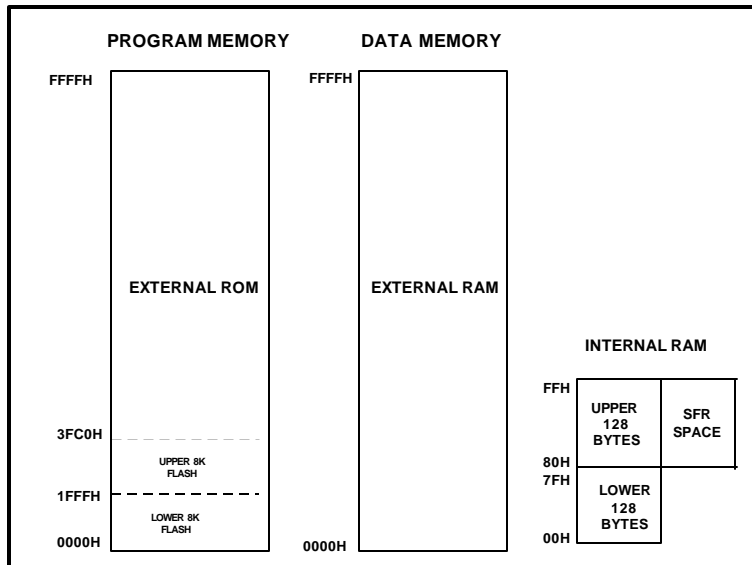
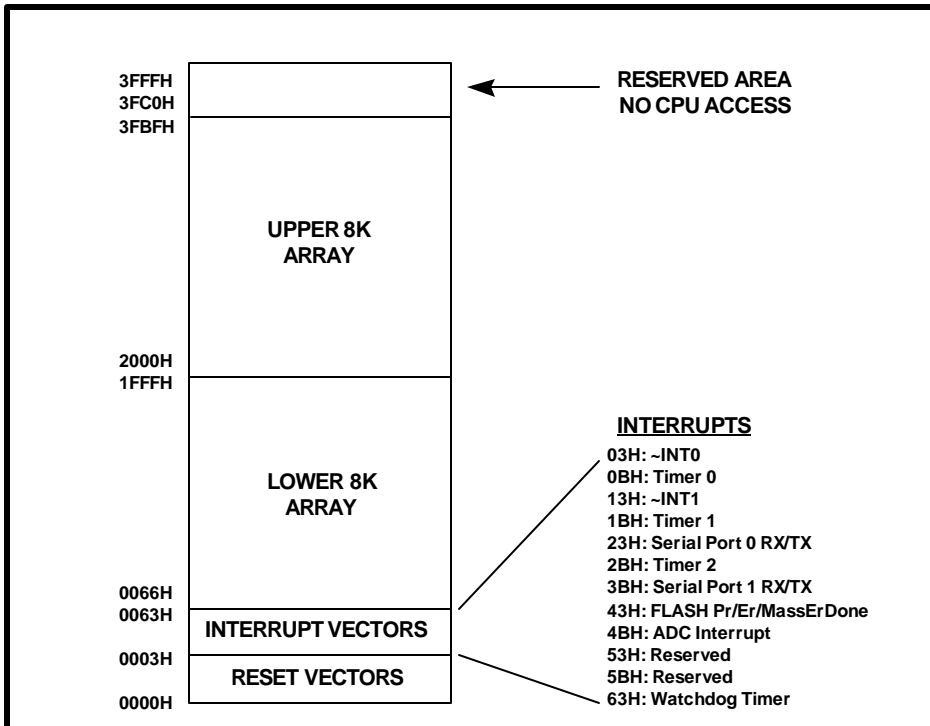


Figure 2-4: FLASH Memory Map



Program Memory

The MAX765x can address up to 64 KB of program memory at addresses 0000H–FFFFH. Program memory can be implemented as internal ROM, external ROM, or a combination of both.

INTERNAL FLASH ROM

The MAX765x contains two 8K byte blocks of CPU-writeable FLASH memory called UPPER and LOWER. Code execution can take place out of either 8K FLASH block. In addition, both blocks can be written, erased or read via the EESTCMD, EEAH, EEAL, and EEDAT registers in the SFR map. The user can execute code out of one FLASH block while writing/erasing/reading the other block. Read operations occur as fast as the CPU can write the address registers (EEAH/EEAL). FLASH write and page erases are *much slower* due to FLASH technology writing speeds.

EXTERNAL ROM

External ROM is read-only and begins at address 4000H unless the \sim EA pin is pulled low. When \sim EA is low, all CPU program fetches will be from external ROM including reset and interrupt vectors.

The MAX765x accesses external ROM in the industry-standard fashion using the P0, P2, and \sim PSEN signals.

EXTERNAL RAM

The MAX765x can access up to 64KB of external RAM, at addresses 0000H – FFFFH, using the MOVX instruction. The MAX765x accesses external RAM using ports P0 and P2 and the \sim WR and \sim RD control signals. Access to external data memory can use either a 16-bit address (MOVX @DPTR) or an 8-bit address (MOVX @Ri). Whenever a 16-bit address is used, the high byte of the address comes out on P2, where it is held for the

duration of the read or write cycle. Note that P2 drivers use strong internal pullups during the entire time that they are emitting address bits that are 1's. This is during the execution of `MOVX @DPTR`, and the P2 latch does not have to contain 1's, and the contents of the P2 SFR are **not** modified.

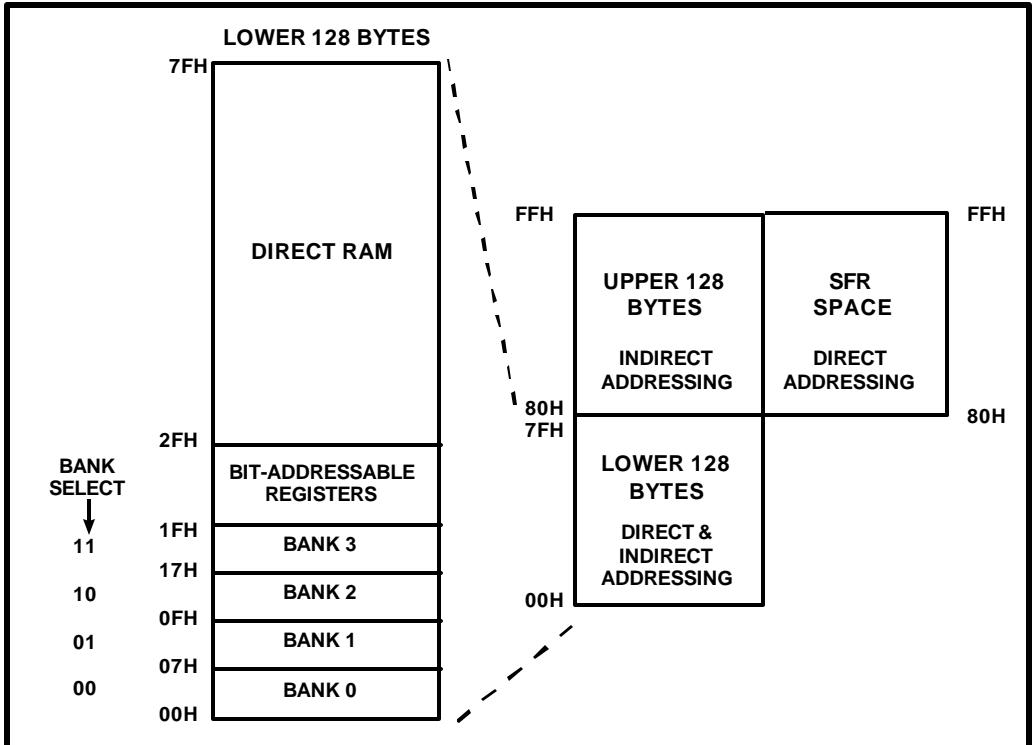
If the external memory cycle is not immediately followed by another external memory cycle, the undisturbed contents of the P2 SFR will reappear in the next cycle. If an 8-bit address is used (`MOVX @Ri`), the contents of the P2 SFR remain at the P2 pins throughout the entire memory cycle.

Internal RAM

The internal RAM (Figure 2-5) consists of the following:

- 128 bytes of registers and scratchpad memory accessible through direct or indirect addressing (addresses 00H–7FH).
- Upper 128 bytes of scratchpad memory accessible through indirect addressing (addresses 80H–FFH).
- 128 special function registers (SFRs) accessible through direct addressing (addresses 80H–FFH).

Figure 2-5: Internal RAM Organization



The lower 128 bytes are organized as shown in Figure 2-5. The lower 32 bytes form four banks of eight registers (R0–R7). Two bits of the program status word (PSW) select which bank is in use. The next 16 bytes form a block of bit-addressable memory space at bit addresses 00H–7FH.

The SFRs and the upper 128 bytes of RAM share the same address range (80H–FFH). However, the actual address space is separate and is differentiated by the type of addressing.

Most SFRs are reserved for specific functions as described in the “Special Function Registers” section of this chapter.

Instruction Set

All MAX765x instructions are binary code compatible and perform the same functions that they do in the 8051. The effects of these instructions on bits, flags, and other status functions is identical to the 8051. However, the timing of the instructions is different, both in terms of number of clock cycles per instruction cycle and timing within the instruction cycle.

Table 2-4 lists the instruction set and the number of instruction cycles required to complete each instruction. Table 2-3 defines the symbols and mnemonics used in Table 2-4.

Table 2-3: Legend for Instruction Set Table

Symbol	Function
A	Accumulator
Rn	Register R0–R7
direct	Internal register address
@Ri	Internal register pointed to by R0 or R1 (except MOVX)
rel	Two's complement offset byte
bit	Direct bit address
#data	8-bit constant
#data 16	16-bit constant
addr 16	16-bit destination address
addr 11	11-bit destination address

Table 2-4: MAX765x Instruction Set

Mnemonic	Description	Byte	Instr. Cycles	Hex Code
Arithmetic				
ADD A, Rn	Add register to A	1	1	28–2F
ADD A, direct	Add direct byte to A	2	2	25
ADD A, @Ri	Add data memory to A	1	1	26–27
ADD A, #data	Add immediate to A	2	2	24
ADDC A, Rn	Add register to A with carry	1	1	38–3F
ADDC A, direct	Add direct byte to A with carry	2	2	35
ADDC A, @Ri	Add data memory to A with carry	1	1	36–37
ADDC A, #data	Add immediate to A with carry	2	2	34
SUBB A, Rn	Subtract register from A with borrow	1	1	98–9F
SUBB A, direct	Subtract direct byte from A with borrow	2	2	95
SUBB A, @Ri	Subtract data memory from A with borrow	1	1	96–97
SUBB A, #data	Subtract immediate from A with borrow	2	2	94
INC A	Increment A	1	1	04
INC Rn	Increment register	1	1	08–0F
INC direct	Increment direct byte	2	2	05
INC @Ri	Increment data memory	1	1	06–07
DEC A	Decrement A	1	1	14
DEC Rn	Decrement register	1	1	18–1F
DEC direct	Decrement direct byte	2	2	15
DEC @Ri	Decrement data memory	1	1	16–17
INC DPTR	Increment data pointer	1	3	A3
MUL AB	Multiply A by B	1	5	A4
DIV AB	Divide A by B	1	5	84
DA A	Decimal adjust A	1	1	D4

Logical				
ANL A, Rn	AND register to A	1	1	58–5F
ANL A, direct	AND direct byte to A	2	2	55
ANL A, @Ri	AND data memory to A	1	1	56–57
ANL A, #data	AND immediate to A	2	2	54
ANL direct, A	AND A to direct byte	2	2	52
ANL direct, #data	AND immediate data to direct byte	3	3	53
ORL A, Rn	OR register to A	1	1	48–4F
ORL A, direct	OR direct byte to A	2	2	45
ORL A, @Ri	OR data memory to A	1	1	46–47
ORL A, #data	OR immediate to A	2	2	44
ORL direct, A	OR A to direct byte	2	2	42
ORL direct, #data	OR immediate data to direct byte	3	3	43
XRL A, Rn	Exclusive-OR register to A	1	1	68–6F
XRL A, direct	Exclusive-OR direct byte to A	2	2	65
XRL A, @Ri	Exclusive-OR data memory to A	1	1	66–67
XRL A, #data	Exclusive-OR immediate to A	2	2	64
XRL direct, A	Exclusive-OR A to direct byte	2	2	62
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3	63
CLR A	Clear A	1	1	E4
CPL A	Complement A	1	1	F4
SWAP A	Swap nibbles of A	1	1	C4
RL A	Rotate A left	1	1	23
RLC A	Rotate A left through carry	1	1	33
RR A	Rotate A right	1	1	03
RRC A	Rotate A right through carry	1	1	13
Data Transfer				
MOV A, Rn	Move register to A	1	1	E8–EF
MOV A, direct	Move direct byte to A	2	2	E5
MOV A, @Ri	Move data memory to A	1	1	E6–E7
MOV A, #data	Move immediate to A	2	2	74

MOV Rn, A	Move A to register	1	1	F8–FF
MOV Rn, direct	Move direct byte to register	2	2	A8–AF
MOV Rn, #data	Move immediate to register	2	2	78–7F
MOV direct, A	Move A to direct byte	2	2	F5
MOV direct, Rn	Move register to direct byte	2	2	88–8F
MOV direct, direct	Move direct byte to direct byte	3	3	85
MOV direct, @Ri	Move data memory to direct byte	2	2	86–87
MOV direct, #data	Move immediate to direct byte	3	3	75
MOV @Ri, A	MOV A to data memory	1	1	F6–F7
MOV @Ri, direct	Move direct byte to data memory	2	2	A6–A7
MOV @Ri, #data	Move immediate to data memory	2	2	76–77
MOV DPTR, #data	Move immediate to data pointer	3	3	90
MOVC A, @A+DPTR	Move code byte relative DPTR to A	1	3	93
MOVC A, @A+PC	Move code byte relative PC to A	1	3	83
MOVX A, @Ri	Move external data (A8) to A	1	2–9*	E2–E3
MOVX A, @DPTR	Move external data (A16) to A	1	2–9*	E0
MOVX @Ri, A	Move A to external data (A8)	1	2–9*	F2–F3
MOVX @DPTR, A	Move A to external data (A16)	1	2–9*	F0
PUSH direct	Push direct byte onto stack	2	2	C0
POP direct	Pop direct byte from stack	2	2	D0
XCH A, Rn	Exchange A and register	1	1	C8–CF
XCH A, direct	Exchange A and direct byte	2	2	C5
XCH A, @Ri	Exchange A and data memory	1	1	C6–C7
XCHD A, @Ri	Exchange A and data memory nibble	1	1	D6–D7

* Number of cycles is user-selectable.

Boolean				
CLR C	Clear carry	1	1	C3
CLR bit	Clear direct bit	2	2	C2
SETB C	Set carry	1	1	D3
SETB bit	Set direct bit	2	2	D2
CPL C	Complement carry	1	1	B3
CPL bit	Complement direct bit	2	2	B2
ANL C, bit	AND direct bit to carry	2	2	82
ANL C, /bit	AND direct bit inverse to carry	2	2	B0
ORL C, bit	OR direct bit to carry	2	2	72
ORL C, /bit	OR direct bit inverse to carry	2	2	A0
MOV C, bit	Move direct bit to carry	2	2	A2
MOV bit, C	Move carry to direct bit	2	2	92
Branching				
ACALL addr 11	Absolute call to subroutine	2	3	11–F1
LCALL addr 16	Long call to subroutine	3	4	12
RET	Return from subroutine	1	4	22
RETI	Return from interrupt	1	4	32
AJMP addr 11	Absolute jump unconditional	2	3	01–E1
LJMP addr 16	Long jump unconditional	3	4	02
SJMP rel	Short jump (relative address)	2	3	80
JC rel	Jump on carry = 1	2	3	40
JNC rel	Jump on carry = 0	2	3	50
JB bit, rel	Jump on direct bit = 1	3	4	20
JNB bit, rel	Jump on direct bit = 0	3	4	30
JBC bit, rel	Jump on direct bit = 1 and clear	3	4	10
JMP @A+DPTR	Jump indirect relative DPTR	1	3	73
JZ rel	Jump on accumulator = 0	2	3	60
JNZ rel	Jump on accumulator /= 0	2	3	70
CJNE A, direct, rel	Compare A, direct JNE relative	3	4	B5
CJNE A, #d, rel	Compare A, immediate JNE relative	3	4	B4
CJNE Rn, #d, rel	Compare reg, immediate JNE relative	3	4	B8–BF

CJNE @Ri, #d, rel	Compare ind, immediate JNE relative	3	4	B6–B7
DJNZ Rn, rel	Decrement register, JNZ relative	2	3	D8–DF
DJNZ direct, rel	Decrement direct byte, JNZ relative	3	4	D5
Miscellaneous				
NOP	No operation	1	1	00
There is an additional reserved opcode (A5) that performs the same function as NOP.				
All mnemonics are copyright © Intel Corporation 1980.				

Instruction Timing

Instruction cycles in the MAX765x are 4 clock cycles in length, as opposed to the 12 clock cycles per instruction cycle in the 8051. This translates to a 3X improvement in execution time for most instructions. Note that the CPU *always* runs at 4 clock cycles per instruction: there is no “backwards compatibility” mode. However, the counter/timers can be programmed to use 12 cycles (default) or 4 cycles per increment.

Some instructions on the MAX765x require a different number of instruction cycles than the 8051. In the 8051, all instructions except for `MUL` and `DIV` take one or two instruction cycles to complete. In the MAX765x architecture, instructions can take from one to five instruction cycles to complete.

For example, in the 8051, the instructions `MOVX A, @DPTR` and `MOV direct, direct` each take two instruction cycles (24 clock cycles) to execute. In the MAX765x architecture, `MOVX A, @DPTR` takes two instruction cycles (8 clock cycles) and `MOV direct, direct` takes three instruction cycles (12 clock cycles). Both instructions execute faster on the MAX765x than

they do on the standard 8051, but require a different number of clock cycles.

For timing of real-time events, use the numbers of instruction cycles from Table 2-4 to calculate the timing of software loops. The bytes column in Table 2-4 indicates the number of memory accesses (bytes) needed to execute the instruction. In most cases, the number of bytes is equal to the number of instruction cycles required to complete the instruction. However, as indicated in Table 2-4, there are some instructions (for example, `DIV` and `MUL`) that require a greater number of instruction cycles than memory accesses.

By default, the MAX765x timers/counters run at 12 clock cycles per increment so that timer-based events have the same timing as with the 8051. The timers can be configured to run at four clock cycles per increment to take advantage of the higher speed.

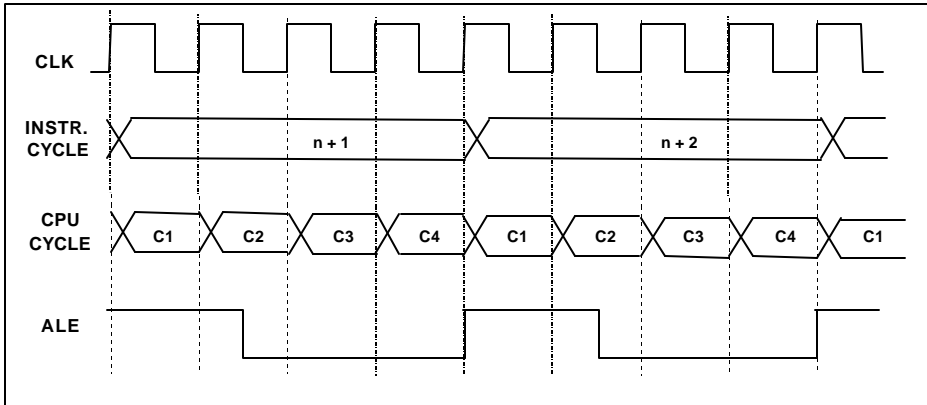
CPU Timing

As previously stated, a MAX765x instruction cycle consists of four *clk* cycles. Each *clk* cycle forms a CPU cycle. Therefore, an instruction cycle consists of four CPU cycles: C1, C2, C3, and C4, as illustrated in Figure 2-6. Various events occur in each CPU cycle, depending on the type of instruction being executed. Throughout this databook, the labels C1, C2, C3, and C4 in timing descriptions refer to the four CPU cycles within a particular instruction cycle.

The execution for instruction n is performed during the fetch of instruction $n+1$. Data writes occur during fetch of instruction $n+2$.

The level-sensitive interrupts are sampled with the rising edge of *clk* at the end of cycle C3.

Figure 2-6: CPU Timing for Single-Cycle



Instruction

Stretch Memory Cycles

The stretch memory cycle feature enables application software to adjust the speed of data memory access. The MAX765x can execute the `MOVX` instruction in as little as two instruction cycles. However, it is sometimes desirable to stretch this value. For example, to access slow memory or slow memory-mapped peripherals such as UARTs or LCDs.

The three LSBs of the Clock Control Register (at SFR location 8EH) control the stretch value. You can use stretch values

between zero and seven. A stretch value of zero adds zero instruction cycles, resulting in `MOVX` instructions executing in two instruction cycles. A stretch value of seven adds seven instruction cycles, resulting in `MOVX` instructions executing in nine instruction cycles. The stretch value can be dynamically changed under program control.

By default, the stretch value resets to one (three cycle `MOVX`). For full-speed data memory access, the software must set the stretch value to zero. The stretch value only affects data memory access. The only way to reduce the speed of program memory (ROM) access is to use a slower clock.

The stretch value affects the width of the read/write strobe and all related timing. Using a higher stretch value results in a wider read/write strobe, which allows the memory (or peripheral) more time to respond.

Table 2-5 lists the data memory access speeds for stretch values zero through seven. MD2–0 are the three LSBs of the Clock Control Register (CKCON.2–0).

Table 2-5: Data Memory Stretch Values

MD2	MD1	MD0	Machine Cycles	Read/Write Strobe Width (Clocks)
0	0	0	2	2
0	0	1	3 (default)	4
0	1	0	4	8
0	1	1	5	12
1	0	0	6	16
1	0	1	7	20
1	1	0	8	24
1	1	1	9	28

Dual Data Pointers

The MAX765x uses dual data pointers to accelerate data memory block moves. The standard 8051 data pointer (DPTR) is a 16-bit value used to address external data RAM or peripherals. The MAX765x maintains the standard data pointer as DPTR0 at SFR locations 82H and 83H. It is **not** necessary to modify code to use DPTR0.

The MAX765x adds a second data pointer (DPTR1) at SFR locations 84H and 85H. The SEL bit in the DPTR Select register, DPS (SFR 86H), selects the active pointer. When bit SEL = 0, instructions that use the DPTR will use DPL0 and DPH0. When SEL = 1, instructions that use the DPTR will use DPL1 and DPH1. SEL is the bit 0 of SFR location 86H. No other bits of SFR location 86H are used.

All DPTR-related instructions use the currently selected data pointer. To switch the active pointer, toggle the SEL bit. The fastest way to do so is to use the increment instruction (`INC DPS`). This requires only one instruction to switch from a source address to a destination address, saving application code from having to save source and destination addresses when doing a block move.

Using dual data pointers provides significantly increased efficiency when moving large blocks of data.

The SFR locations related to the dual data pointers are:

82H	DPL0	DPTR0 low byte
83H	DPH0	DPTR0 high byte
84H	DPL1	DPTR1 low byte
85H	DPH1	DPTR1 high byte
86H	DPS	DPTR Select (LSB)

Special Function Registers

The Special Function Registers (SFRs) control several of the features of the MAX765x. Most of the MAX765x SFRs are identical to the 8051 SFRs. However, there are additional SFRs that control features that are not available in the 8051.

Table 2-6 lists the MAX765x SFRs and indicates which SFRs are not included in the standard 8051 SFR space. When writing software for the MAX765x, use equate statements to define the SFRs that are specific to the MAX765x and custom peripherals.

In Table 2-6, SFR bit positions that contain a 0 or a 1 cannot be written to and, when read, always return the value shown (0 or 1). SFR bit positions that contain a hyphen (-) are available but not used. Table 2-7 lists the reset values for the SFRs.

Microprocessor SFR Register Map

The following table merges the standard 8051 SFR register map with the custom registers added by Maxim for PWM, A/D, FLASH control, etc. Unshaded areas are 8051 SFR standard register definitions.

Table 2-6: MAX765x SFR Registers

Hex Addr.	0 / 8	1 / 9	2 / A	3 / B	4 / C	5 / D	6 / E	7 / F
F8	EIP						PWMC	
F0	B							
E8	EIE		EEAL	EEAH	EEDAT	EESTCM D		
E0	ACC							
D8	EICON		PWPS	PWDTA	PWDTB	WDT		
D0	PSW							
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2		
C0	SCON1	SBUF1	ADDAT0	ADDAT1	ADDAT2	ADCON		
B8	IP							
B0	P3		VERSION					
A8	IE							
A0	P2							
98	SCON0	SBUF0						
90	P1	EXIF						
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	SPC_FNC
80	P0	SP	DPL0	DPH0	DPL1	DPH1	DPS	PCON

Table 2-7: MAX765x SFR Reset Values

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Addr
SP	0	0	0	0	0	1	1	1	81h
DPL0	0	0	0	0	0	0	0	0	82h
DPH0	0	0	0	0	0	0	0	0	83h
DPL1	0	0	0	0	0	0	0	0	84h
DPH1	0	0	0	0	0	0	0	0	85h
DPS	0	0	0	0	0	0	0	0	86h
PCON	0	0	1	1	0	0	0	0	87h
TCON	0	0	0	0	0	0	0	0	88h
TMOD	0	0	0	0	0	0	0	0	89h
TL0	0	0	0	0	0	0	0	0	8Ah
TL1	0	0	0	0	0	0	0	0	8Bh
TH0	0	0	0	0	0	0	0	0	8Ch
TH1	0	0	0	0	0	0	0	0	8Dh
CKCON	0	0	0	0	0	0	0	1	8Eh
SPC_FNC	0	0	0	0	0	0	0	0	8Fh
EXIF	0	0	0	0	1	0	0	0	91h
MPAGE	0	0	0	0	0	0	0	0	92h
SCON0	0	0	0	0	0	0	0	0	98h
SBUF0	0	0	0	0	0	0	0	0	99h
IE	0	0	0	0	0	0	0	0	A8h
IP	1	0	0	0	0	0	0	0	B8h
SCON1	0	0	0	0	0	0	0	0	C0h
SBUF1	0	0	0	0	0	0	0	0	C1h
T2CON	0	0	0	0	0	0	0	0	C8h
RCAP2L	0	0	0	0	0	0	0	0	CAh
RCAP2H	0	0	0	0	0	0	0	0	CBh
TL2	0	0	0	0	0	0	0	0	CCh
TH2	0	0	0	0	0	0	0	0	CDh
PSW	0	0	0	0	0	0	0	0	D0h
EICON	0	1	0	0	0	0	0	0	D8h



ACC	0	0	0	0	0	0	0	0	E0h
EIE	1	1	1	0	0	0	0	0	E8h
B	0	0	0	0	0	0	0	0	F0h
EIP	1	1	1	0	0	0	0	0	F8h

The following SFRs are related to CPU operation and program execution:

81H	SP	Stack Pointer
D0H	PSW	Program Status Word
E0H	ACC	Accumulator Register
F0H	B	B Register

Table 2-8 lists the functions of the bits in the PSW SFR. Detailed descriptions of the remaining SFRs appear with the associated hardware descriptions in Chapter 3.

Table 2-8: PSW Register - SFR D0H

Bit	Function															
PSW.7	CY Carry flag. Set to 1 when the last arithmetic operation resulted in a carry (during addition) or borrow (during subtraction), otherwise cleared to 0 by all arithmetic operations.															
PSW.6	AC Auxiliary carry flag. Set to 1 when the last arithmetic operation resulted in a carry into (during addition) or borrow from (during subtraction) the high order nibble, otherwise cleared to 0 by all arithmetic operations.															
PSW.5	F0 User flag 0. Bit-addressable, general purpose flag for software control.															
PSW.4	RS1 Register bank select bit 1. Used with RS0 to select a register bank in internal RAM.															
PSW.3	RS0 Register bank select bit 0, decoded as: <table border="1" data-bbox="282 1006 981 1197"> <thead> <tr> <th>RS1</th> <th>RS0</th> <th>Bank Selected</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Register bank 0, addresses 00H-07H</td> </tr> <tr> <td>0</td> <td>1</td> <td>Register bank 1, addresses 08H-0FH</td> </tr> <tr> <td>1</td> <td>0</td> <td>Register bank 2, addresses 10H-17H</td> </tr> <tr> <td>1</td> <td>1</td> <td>Register bank 3, addresses 18H-1FH</td> </tr> </tbody> </table>	RS1	RS0	Bank Selected	0	0	Register bank 0, addresses 00H-07H	0	1	Register bank 1, addresses 08H-0FH	1	0	Register bank 2, addresses 10H-17H	1	1	Register bank 3, addresses 18H-1FH
RS1	RS0	Bank Selected														
0	0	Register bank 0, addresses 00H-07H														
0	1	Register bank 1, addresses 08H-0FH														
1	0	Register bank 2, addresses 10H-17H														
1	1	Register bank 3, addresses 18H-1FH														
PSW.2	OV Overflow flag. Set to 1 when the last arithmetic operation resulted in a carry (addition), borrow (subtraction), or overflow (multiply or divide). Otherwise, the bit cleared to 0 by all arithmetic operations.															
PSW.1	F1 User flag 1. Bit-addressable, general purpose flag for software control.															
PSW.0	P Parity flag. Set to 1 when the modulo-2 sum of the 8 bits in the accumulator is 1 (odd parity), cleared to 0 on even parity.															

3

MAX765x Hardware Description

This chapter provides technical data about the MAX765x hardware operation and timing. The following topics are discussed:

- Timers/Counters
- Serial Interface
- SFR Registers
- A/D Converter
- Pulse-Width Modulator
- Watchdog Timer
- Interrupts
- Reset
- Power Saving Modes

Timers/Counters

The MAX765x includes three 16-bit timer/counters: Timer 0, Timer 1 and Timer 2. Each timer/counter can operate as either a timer with a clock rate based on the *clk* pin, or as an event counter clocked by the *t0* pin [Timer 0], *t1* pin [Timer 1], or the *t2* pin [Timer 2].

In timer mode, internal clock pulses increment the registers. In counter mode, external pulses applied to specific pins increment the registers. CLK/8 is the maximum input frequency.

Each timer/counter consists of a 16-bit register that is accessible to software as a SFR pair.

- Timer 0 – TL0 and TH0
- Timer 1 – TL1 and TH1
- Timer 2 – TL2 and TH2

Where THx and TLx are the high/low bytes of the associated 16-bit timer register.

Table 3-1: Timer/Counter Implementation Comparison

Feature	8051	MAX765x
Number of timers	2	3
Timer 0/1 overflow available	not implemented	<i>t0_out</i> , <i>t1_out</i> (one clk pulse)
Timer 2 overflow available	n/a	<i>t2_out</i> (one clk pulse)

Timers 0 and 1

Timers 0 and 1 each operate in four modes, as controlled through the TMOD SFR (Table 3-2) and the TCON SFR (Table 3-3). The four modes are:

- Table 3-1: 13-bit timer/counter (mode 0)
- Table 3-2: 16-bit timer/counter (mode 1)
- Table 3-3: 8-bit counter with auto-reload (mode 2)
- Table 3-4: Two 8-bit counters (mode 3, Timer 0 only)

Mode 0

Mode 0 operation (Figure 3-1) is the same for Timer 0 and Timer 1. In mode 0, the timer is configured as a 13-bit counter that uses bits 0–4 of TL0 (or TL1) and all 8 bits of TH0 (or TH1). The timer enable bit (TR0/TR1) in the TCON SFR starts the timer. The C/T bit selects the timer/counter clock source, *clk* or *t0/t1*.

The timer counts transitions from the selected source as long as the GATE bit is 0, or the GATE bit is 1 and the corresponding interrupt pin (\sim INT0 or \sim INT1) is deasserted.

When the 13-bit count increments from 1FFFH (all ones), the counter rolls over to all zeros, the TF0 (or TF1) bit is set in the TCON SFR, and the *T0* (or *T1*) pin goes high for one clock cycle.

The upper three bits of TL0 (or TL1) are indeterminate in mode 0 and must be masked when the software evaluates the register.

Table 3-2: TMOD Register - SFR 89H

Bit	Function															
TMOD.7	GATE – Timer 1 gate control. When GATE = 1, Timer 1 will clock only when $\sim INT1 = 1$ and TR1 (TCON.6) = 1. When GATE = 0, Timer 1 will clock only when TR1 = 1, regardless of the state of $\sim INT1$.															
TMOD.6	C/T – Counter/Timer select. When C/T = 0, Timer 1 is clocked by $clk/4$ or $clk/12$, depending on the state of T1M (CKCON.4). When C/T = 1, Timer 1 is clocked by the T1 pin.															
TMOD.5	M1 – Timer 1 mode select bit 1.															
TMOD.4	M0 – Timer 1 mode select bit 0, decoded as: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>M1</th> <th>M0</th> <th>Mode</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Mode 0 : 13-bit counter</td> </tr> <tr> <td>0</td> <td>1</td> <td>Mode 1 : 16-bit counter</td> </tr> <tr> <td>1</td> <td>0</td> <td>Mode 2 : 8-bit counter with auto-reload</td> </tr> <tr> <td>1</td> <td>1</td> <td>Mode 3 : Two 8-bit counters</td> </tr> </tbody> </table>	M1	M0	Mode	0	0	Mode 0 : 13-bit counter	0	1	Mode 1 : 16-bit counter	1	0	Mode 2 : 8-bit counter with auto-reload	1	1	Mode 3 : Two 8-bit counters
M1	M0	Mode														
0	0	Mode 0 : 13-bit counter														
0	1	Mode 1 : 16-bit counter														
1	0	Mode 2 : 8-bit counter with auto-reload														
1	1	Mode 3 : Two 8-bit counters														
TMOD.3	GATE – Timer 0 gate control. When GATE = 1, Timer 0 will clock only when $\sim INT0 = 1$ and TR0 (TCON.4) = 1. When GATE = 0, Timer 0 will clock only when TR0 = 1, regardless of the state of $\sim INT0$.															
TMOD.2	C/T – Counter/Timer select. When C/T = 0, Timer 0 is clocked by $clk/4$ or $clk/12$, depending on the state of T0M (CKCON.3). When C/T = 1, Timer 0 is clocked by the t0 pin.															
TMOD.1	M1 – Timer 0 mode select bit 1.															
TMOD.0	M0 – Timer 0 mode select bit 0, decoded as: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>M1</th> <th>M0</th> <th>Mode</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Mode 0 : 13-bit counter</td> </tr> <tr> <td>0</td> <td>1</td> <td>Mode 1 : 16-bit counter</td> </tr> <tr> <td>1</td> <td>0</td> <td>Mode 2 : 8-bit counter with auto-reload</td> </tr> <tr> <td>1</td> <td>1</td> <td>Mode 3 : Two 8-bit counters</td> </tr> </tbody> </table>	M1	M0	Mode	0	0	Mode 0 : 13-bit counter	0	1	Mode 1 : 16-bit counter	1	0	Mode 2 : 8-bit counter with auto-reload	1	1	Mode 3 : Two 8-bit counters
M1	M0	Mode														
0	0	Mode 0 : 13-bit counter														
0	1	Mode 1 : 16-bit counter														
1	0	Mode 2 : 8-bit counter with auto-reload														
1	1	Mode 3 : Two 8-bit counters														

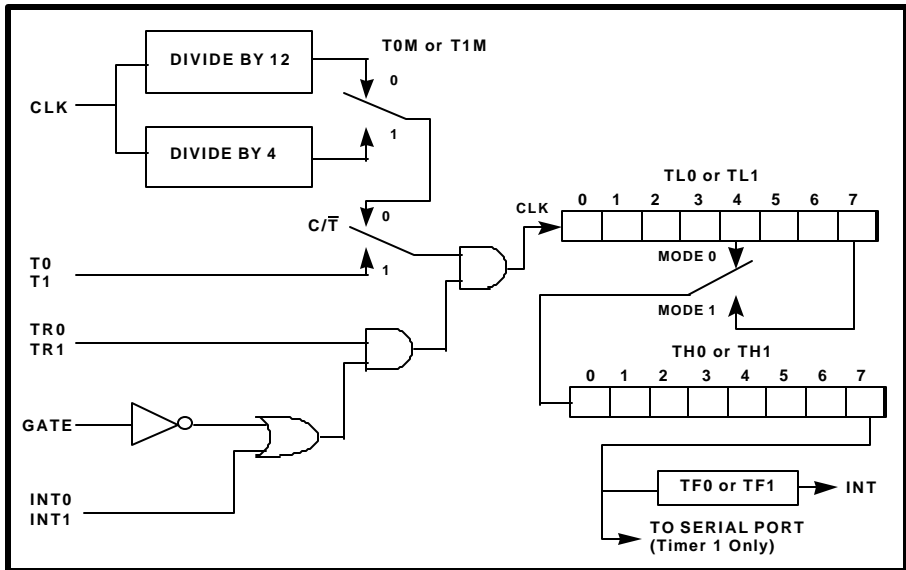
Table 3-3: TCON Register – SFR 88H

Bit	Function
TCON.7	TF1 – Timer 1 overflow flag. Set to 1 when the Timer 1 count overflows and cleared when the CPU vectors to the interrupt service routine.
TCON.6	TR1 – Timer 1 run control. Set to 1 to enable counting on Timer 1.
TCON.5	TF0 – Timer 0 overflow flag. Set to 1 when the Timer 0 count overflows and cleared when the CPU vectors to the interrupt service routine.
TCON.4	TR0 – Timer 0 run control. Set to 1 to enable counting on Timer 0.
TCON.3	IE1 – Interrupt 1 edge detect. If external interrupt 1 is configured to be edge-sensitive (IT1 = 1), IE1 is set by hardware when a negative edge is detected on the \sim INT1 pin and is automatically cleared when the CPU vectors to the corresponding interrupt service routine. In edge-sensitive mode, IE1 can also be cleared by software. If external interrupt 1 is configured to be level-sensitive (IT1 = 0), IE1 is set when the \sim INT1 pin is low and cleared when the \sim INT1 pin is high. In level-sensitive mode, software cannot write to IE1.
TCON.2	IT1 – Interrupt 1 type select. When IT1 = 1, the MAX765x detects \sim INT1 on the falling edge (edge-sensitive). When IT1 = 0, the MAX765x detects \sim INT1 as a low level (level-sensitive).
TCON.1	IE0 – Interrupt 0 edge detect. If external interrupt 0 is configured to be edge-sensitive (IT0 = 1), IE0 is set by hardware when a negative edge is detected on the \sim INT0 pin and is automatically cleared when the CPU vectors to the corresponding interrupt service routine. In edge-sensitive mode, IE0 can also be cleared by software. If external interrupt 0 is configured to be level-sensitive (IT0 = 0), IE0 is set when the \sim INT0 pin is low and cleared when the \sim INT0 pin is high. In level-sensitive mode, software cannot write to IE0.
TCON.0	IT0 – Interrupt 0 type select. When IT1 = 1, the MAX765x detects \sim INT0 on the falling edge (edge-sensitive). When IT1 = 0, the MAX765x detects \sim INT0 as a low level (level-sensitive).

Mode 1

Mode 1 operation is the same for Timer 0 and Timer 1. In mode 1, the timer is configured as a 16-bit counter. As illustrated in Figure 3-1, all 8 bits of the LSB register (TL0 or TL1) are used. The counter rolls over to all zeros when the count increments from FFFFH. Otherwise, Mode 1 operation is the same as Mode 0.

Figure 3-1: Timer 0/1 – Modes 0 and 1

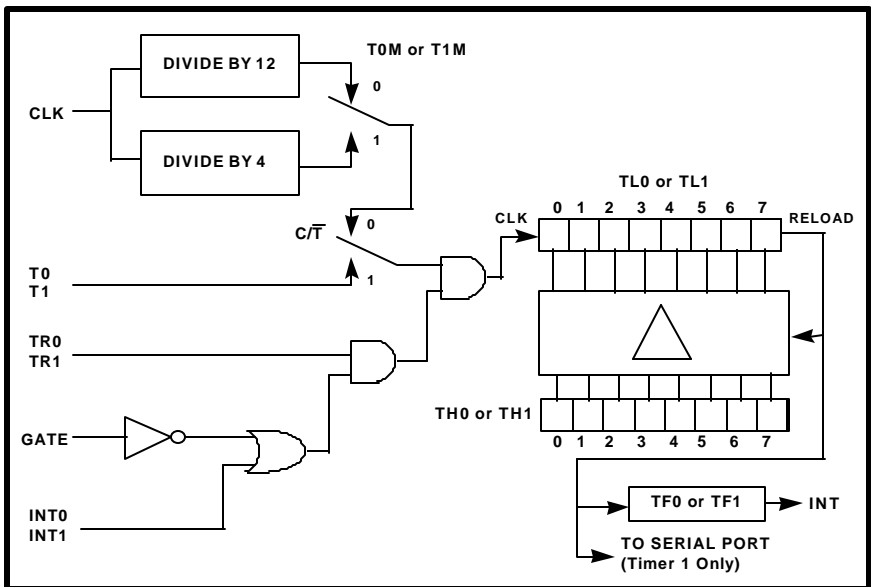


Mode 2

Mode 2 operation is the same for Timer 0 and Timer 1. In Mode 2, the timer is configured as an 8-bit counter, with automatic reload of the start value. The LSB register (TL0 or TL1) is the counter and the MSB register (TH0 or TH1) stores the reload value.

As illustrated in Figure 3-2, Mode 2 counter control is the same as for Mode 0 and Mode 1. However, in Mode 2, when TL_n increments from FFH, the value stored in TH_n is reloaded into TL_n.

Figure 3-2: Timer 0/1 – Modes 0 and 1



Mode 3

In Mode 3, Timer 0 operates as two 8-bit counters and Timer 1 stops counting and holds its value.

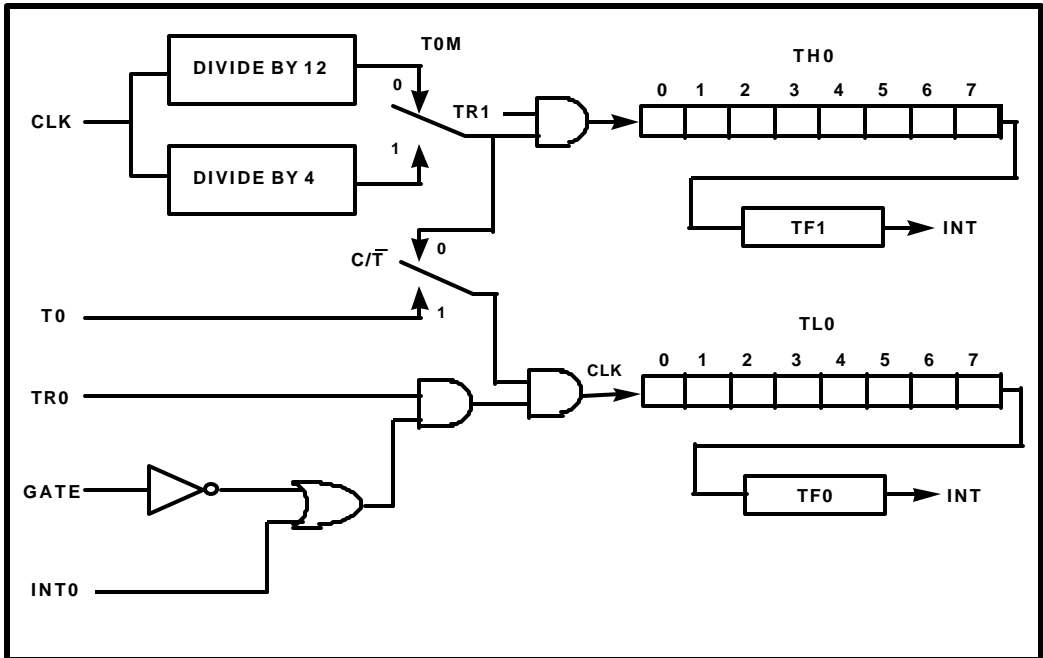
As illustrated in Figure 3-3, TL0 is configured as an 8-bit counter controlled by the normal Timer 0 control bits. TL0 can either count *clk* cycles (divided by 4 or 12) or high-to-low transitions on *T0*, as determined by the C/T bit. The GATE function can be used to give counter enable control to the \sim INT0 signal.

TH0 functions as an independent 8-bit counter. However, TH0 can only count *clk* cycles (divided by 4 or 12). The Timer 1 control and flag bits (TR1 and TF1) are used as the control and flag bits for TH0.

When Timer 0 is in Mode 3, Timer 1 has limited usage because Timer 0 uses the Timer 1 control bit (TR1) and interrupt flag (TF1). Timer 1 can still be used for baud rate generation and the Timer 1 count values are still available in the TL1 and TH1 registers.

Control of Timer 1 when Timer 0 is in Mode 3 is through the Timer 1 mode bits. To turn Timer 1 on, set Timer 1 to mode 0, 1, or 2. To turn Timer 1 off, set it to Mode 3. The Timer 1 C/T bit and T1M bit are still available to Timer 1. Therefore, Timer 1 can count *clk/4*, *clk/12*, or high-to-low transitions on the *T1* pin. The Timer 1 GATE function is also available when Timer 0 is in Mode 3.

Figure 3-3: Timer 0 – Mode 3



Timer Rate Control

The default timer clock scheme for the MAX765x timers is 12 *clk* cycles per increment, the same as in the 8051. However, in the MAX765x, the instruction cycle is 4 *clk* cycles.

Using the default rate (12 clocks per timer increment) allows existing application code with real-time dependencies, such as baud rate, to operate properly. However, applications that require fast timing can set the timers to increment every 4 *clk* cycles by

setting bits in the Clock Control register (CKCON) at SFR location 8EH (see Table 3-4).

The CKCON bits that control the timer clock rates are:

<u>CKCON bit</u>	<u>Counter/Timer</u>
5	Timer 2
4	Timer 1
3	Timer 0

When a CKCON register bit is set to 1, the associated counter increments at 4 *clk* intervals. When a CKCON bit is cleared, the associated counter increments at 12 *clk* intervals. The timer controls are independent of each other. The default setting for all three timers is 0 (12 *clk* intervals). These bits have no effect in counter mode.

Table 3-4: CKCON Register – SFR 8EH

Bit	Function
CKCON.7,6	WD1,0 – selects 1 of 4 possible WDT timeout periods.
CKCON.5	T2M – Timer 2 clock select. When T2M = 0, Timer 2 uses <i>clk/12</i> (for compatibility with 80C32); when T2M = 1, Timer 2 uses <i>clk/4</i> . This bit has no effect when Timer 2 is configured for baud rate generation.
CKCON.4	T1M – Timer 1 clock select. When T1M = 0, Timer 1 uses <i>clk/12</i> (for compatibility with 80C32); when T1M = 1, Timer 1 uses <i>clk/4</i> .
CKCON.3	T0M – Timer 0 clock select. When T0M = 0, Timer 0 uses <i>clk/12</i> (for compatibility with 80C32); when T0M = 1, Timer 0 uses <i>clk/4</i> .
CKCON.2–0	MD2, MD1, MD0 – Control the number of cycles to be used for external MOVX instructions. See “Stretch Memory Cycles” in Chapter 2 for details.

Timer 2

Timer 2 runs only in 16-bit mode and offers several capabilities not available with Timers 0 and 1. The modes available with Timer 2 are:

- 16-bit timer/counter
- 16-bit timer with capture
- 16-bit auto-reload timer/counter
- Baud-rate generator

The SFRs associated with Timer 2 are:

- T2CON – SFR C8H.
- RCAP2L – SFR CAH. Used to capture the TL2 value when Timer 2 is configured for capture mode, or as the LSB of the 16-bit reload value when Timer 2 is configured for auto-reload mode.
- RCAP2H – SFR CBH. Used to capture the TH2 value when Timer 2 is configured for capture mode, or as the MSB of the 16-bit reload value when Timer 2 is configured for auto-reload mode.
- TL2 – SFR CCH. Lower 8 bits of the 16-bit count.
- TH2 – SFR CDH. Upper 8 bits of the 16-bit count.

Timer 2 Mode Control

Table 3-5 summarizes how the SFR bits determine the Timer 2 mode.

Table 3-5: Timer 2 Mode Control Summary

RCLK	TCLK	CP/RL2	TR2	Mode
0	0	1	1	16-bit timer/counter with capture
0	0	0	1	16-bit timer/counter with auto-reload
1	X	X	1	Baud rate generator
X	1	X	1	Baud rate generator
X	X	X	0	Off

X = Don't care.

Table 3-6: T2CON Register - SFR C8H

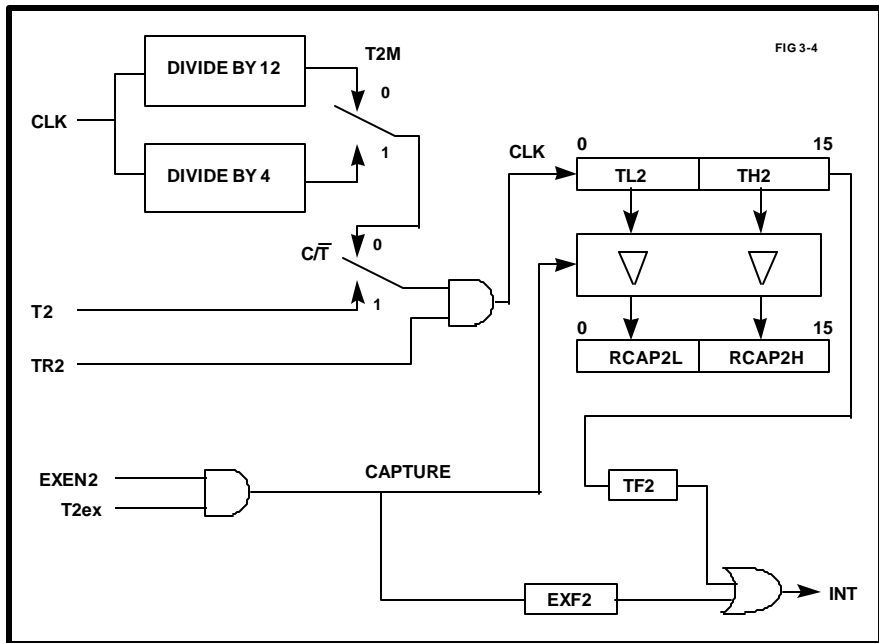
Bit	Function
T2CON.7	TF2 – Timer 2 overflow flag. Hardware will set TF2 when Timer 2 overflows from FFFFH. TF2 must be cleared to 0 by the software. TF2 will only be set to a 1 if RCLK and TCLK are both cleared to 0. Writing a 1 to TF2 forces a Timer 2 interrupt if enabled.
T2CON.6	EXF2 – Timer 2 external flag. Hardware will set EXF2 when a reload or capture is caused by a high-to-low transition on the <i>T2EX</i> pin, and EXEN2 is set. EXF2 must be cleared to 0 by the software. Writing a 1 to EXF2 forces a Timer 2 interrupt if enabled.
T2CON.5	RCLK – Receive clock flag. Determines whether Timer 1 or Timer 2 is used for Serial Port 0 timing of received data in serial mode 1 or 3. RCLK =1 selects Timer 2 overflow as the receive clock. RCLK = 0 selects Timer 1 overflow as the receive clock.
T2CON.4	TCLK – Transmit clock flag. Determines whether Timer 1 or Timer 2 is used for Serial Port 0 timing of transmit data in serial mode 1 or 3. TCLK =1 selects Timer 2 overflow as the transmit clock. TCLK = 0 selects Timer 1 overflow as the transmit clock.
T2CON.3	EXEN2 – Timer 2 external enable. EXEN2 = 1 enables capture or reload to occur as a result of a high-to-low transition on <i>T2EX</i> , if Timer 2 is not generating baud rates for the serial port. EXEN2 = 0 causes Timer 2 to ignore all external events at <i>T2</i> .
T2CON.2	TR2 – Timer 2 run control flag. TR2 = 1 starts Timer 2. TR2 = 0 stops Timer 2.
T2CON.1	C/T2 – Counter/timer select. C/T2 = 0 selects a timer function for Timer 2. C/T2 = 1 selects a counter of falling transitions on the <i>T2EX</i> pin. When used as a timer, Timer 2 runs at 4 clocks per increment or 12 clocks per increment as programmed by CKCON.5, in all modes except baud rate generator mode. When used in baud rate generator mode, Timer 2 runs at 2 clocks per increment, independent of the state of CKCON.5.
T2CON.0	CP/RL2 – Capture/reload flag. When CP/RL2 = 1, Timer 2 captures occur on high-to-low transitions of <i>T2EX</i> , if EXEN2 = 1. When CP/RL2 = 0, auto-reloads occur when Timer 2 overflows or when high-to-low transitions occur on <i>T2EX</i> if EXEN2 = 1. If either RCLK or TCLK is set to 1, CP/RL2 will not function and Timer 2 will operate in auto-reload mode following each overflow.

16-Bit Timer/Counter Mode with Capture

The Timer 2 capture mode (Figure 3-4) is the same as the 16-bit timer/counter mode, with the addition of the capture registers and control signals.

The CP/RL2 bit in the T2CON SFR enables the capture feature. When CP/RL2 = 1, a high-to-low transition on T2 when EXEN2 = 1 causes the Timer 2 value to be loaded into the capture registers (RCAP2L and RCAP2H).

Figure 3-4: Timer 2 – Timer/Counter with Capture

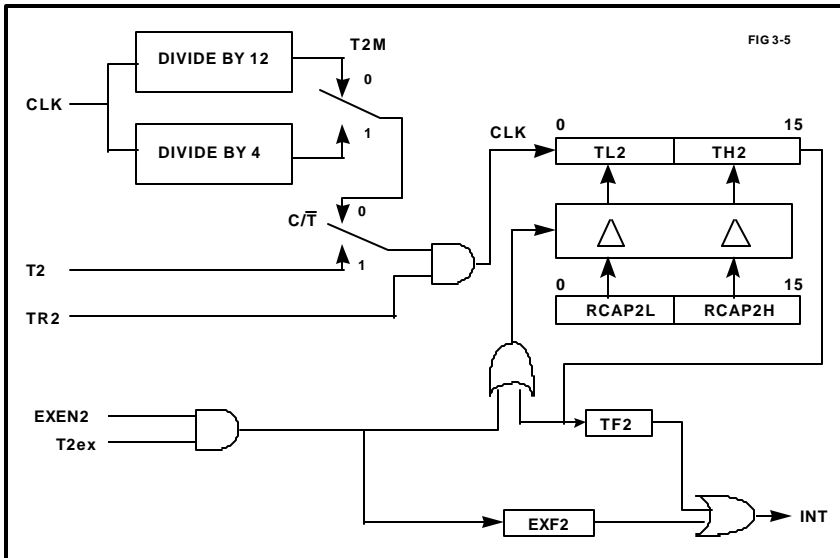


16-Bit Timer/Counter Mode with Auto-Reload

When $CP/RL2 = 0$, Timer 2 is configured for the auto-reload mode (Figure 3-5). Control of counter input is the same as for the other 16-bit counter modes. When the count increments from FFFFH, Timer 2 sets the TF2 flag and the starting value is reloaded into TL2 and TH2. The software must preload the starting value into the RCAP2L and RCAP2H registers.

When Timer 2 is in auto-reload mode, a reload can be forced by a high-to-low transition on the T2 pin, if enabled by setting $EXEN2 = 1$.

Figure 3-5: Timer 2 – Timer/Counter with Auto-Reload



Baud-Rate Generator Mode

Setting either RCLK or TCLK to 1 configures Timer 2 to generate baud rates for Serial Port 0 in serial mode 1 or 3. In baud rate generator mode, Timer 2 functions in auto-reload mode.

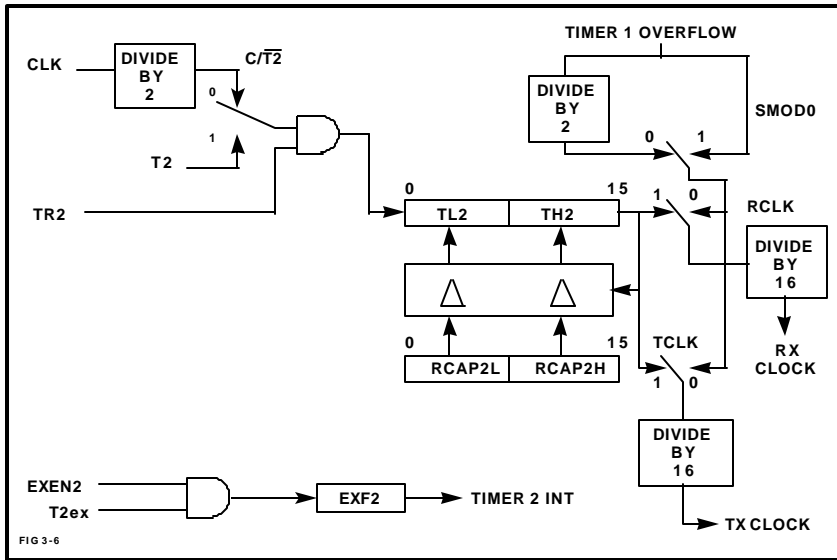
However, instead of setting the TF2 flag, the counter overflow generates a shift clock for the serial port function. As in normal auto-reload mode, the overflow also causes the preloaded start value in the RCAP2L and RCAP2H registers to be reloaded into the TL2 and TH2 registers.

When either TCLK = 1 or RCLK = 1, Timer 2 is forced into auto-reload operation, regardless of the state of the CP/RL2 bit.

When operating as a baud-rate generator, Timer 2 does not set the TF2 bit. In this mode, a Timer 2 interrupt can only be generated by a high-to-low transition on the T2 pin setting the EXF2 bit, and only if enabled by EXEN2 = 1.

The counter time base in baud-rate generator mode is $clk/2$. To use an external clock source, set C/T2 to 1 and apply the desired clock source to the T2 pin.

Figure 3-6: Timer 2 –Timer/Counter with Auto-Reload



Serial Interface

The MAX765x provides two serial ports. Serial Port 0 is identical in operation to the 8051 serial port. Serial Port 1 is identical to Serial Port 0, except that Timer 2 cannot be used as the baud-rate generator for Serial Port 1.

Each serial port can operate in synchronous or asynchronous mode. In synchronous mode, the MAX765x generates the serial clock, and the serial port operates in half-duplex mode. In asynchronous mode, the serial port operates in full-duplex mode. In all modes, the MAX765x buffers receive data in a holding register, enabling the UART to receive an incoming word before the software has read the previous value.

Each serial port can operate in one of four modes, as outlined in Table 3-7 .

Figure 3-7: Serial Port Modes

Mode	Sync/Async	Baud Clock	Data Bits	Start/Stop	9th Bit Function
0	Sync	<i>clk/4</i> or <i>clk/12</i>	8	None	None
1	Async	Timer 1 or Timer 2 ⁽¹⁾	8	1 start, 1 stop	None
2	Async	<i>clk/32</i> or <i>clk/64</i>	9	1 start, 1 stop	0, 1, parity
3	Async	Timer 1 or Timer 2 ⁽¹⁾	9	1 start, 1 stop	0, 1, parity

(1) Timer 2 available for Serial Port 0 only.

The SFRs associated with the serial ports are:

- SCON0 - SFR 98H. Serial Port 0 control (Table 3-9)
- SBUF0 - SFR 99H. Serial Port 0 buffer.
- SCON1 - SFR C0H. Serial Port 1 control (Table 3-10).
- SBUF1 - SFR C1H. Serial Port 1 buffer.

Table 3-8: Serial Interface Implementation Comparison

Feature	8051	MAX765x
Number of serial ports	1	2
Framing error detection	not implemented	not implemented
Slave address comparison for multiprocessor communication	not implemented	not implemented

Mode 0

Serial mode 0 provides synchronous, half-duplex serial communication. For Serial Port 0, serial I/O occurs on the RXD0 pin. The shift clock is provided on the TXD0 pin. Note that whether transmitting or receiving, the serial clock is generated by the MAX765x. Serial Port 1 uses RXD1 and TXD1 in a similar manner.

The serial mode 0 baud rate is either $clk/12$ or $clk/4$, depending on the state of the SM2_0 bit (or SM2_1 for Serial Port 1). When SM2_0 = 0, the baud rate is $clk/12$; when SM2_0 = 1, the baud rate is $clk/4$.

Mode 0 operation is identical to the 8051. Data transmission begins when an instruction writes to the SBUF0 (or SBUF1) SFR. The UART shifts the data out, LSB first, at the selected baud rate until the 8-bit value has been shifted out.

Mode 0 data reception begins when the REN_0 (or REN_1) bit is set and the RI_0 (or RI_1) bit is cleared in the corresponding SCON SFR. The shift clock is activated and the UART shifts data in on each rising edge of the shift clock until eight bits have been received. One machine cycle after the 8th bit is shifted in, the RI_0 (or RI_1) bit is set and reception stops until the software clears the RI bit.

Figure 3-7 through Figure 3-10 illustrate Serial Port Mode 0 transmit and receive timing for both low-speed ($clk/12$) and high-speed ($clk/4$) operation.

Table 3-9: SCON0 Register – SFR 98H

Bit	Function															
SCON0.7	SM0_0 – Serial Port 0 mode bit 0.															
SCON0.6	SM1_0 – Serial Port 0 mode bit 1, decoded as: <table border="1"> <thead> <tr> <th>SM0_0</th> <th>SM1_0</th> <th>Mode</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>2</td> </tr> <tr> <td>1</td> <td>1</td> <td>3</td> </tr> </tbody> </table>	SM0_0	SM1_0	Mode	0	0	0	0	1	1	1	0	2	1	1	3
SM0_0	SM1_0	Mode														
0	0	0														
0	1	1														
1	0	2														
1	1	3														
SCON0.5	SM2_0 – Multiprocessor communication enable. In modes 2 and 3, SM2_0 enables the multiprocessor communication feature. If SM2_0 = 1 in mode 2 or 3, RI_0 will not be activated if the received 9th bit is 0. If SM2_0 = 1 in mode 1, RI_0 will only be activated if a valid stop is received. In mode 0, SM2_0 establishes the baud rate: when SM2_0 = 0, the baud rate is $clk/12$. When SM2_0 = 1, the baud rate is $clk/4$.															
SCON0.4	REN_0 – Receive enable. When REN_0 = 1, reception is enabled.															
SCON0.3	TB8_0 – Defines the state of the 9th data bit transmitted in modes 2 and 3.															
SCON0.2	RB8_0 – In modes 2 and 3, RB8_0 indicates the state of the 9th bit received. In mode 1, RB8_0 indicates the state of the received stop bit. In mode 0, RB8_0 is not used.															
SCON0.1	TI_0 – Transmit interrupt flag. Indicates that the transmit data word has been shifted out. In mode 0, TI_0 is set at the end of the 8th data bit. In all other modes, TI_0 is set when the stop bit is placed on the <i>txd0</i> pin. TI_0 must be cleared by the software.															
SCON0.0	RI_0 – Receive interrupt flag. Indicates that a serial data word has been received. In mode 0, RI_0 is set at the end of the 8th data bit. In mode 1, RI_0 is set after the last sample of the incoming stop bit, subject to the state of SM2_0. In modes 2 and 3, RI_0 is set at the end of the last sample of RB8_0. RI_0 must be cleared by the software.															

Table 3-10: SCON1 Register – SFR C0H

Bit	Function															
SCON1.7	SM0_1 – Serial Port 1 mode bit 0.															
SCON1.6	SM1_1 – Serial Port 1 mode bit 1, decoded as: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>SM0_1</th> <th>SM1_1</th> <th>Mode</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>2</td> </tr> <tr> <td>1</td> <td>1</td> <td>3</td> </tr> </tbody> </table>	SM0_1	SM1_1	Mode	0	0	0	0	1	1	1	0	2	1	1	3
SM0_1	SM1_1	Mode														
0	0	0														
0	1	1														
1	0	2														
1	1	3														
SCON1.5	SM2_1 – Multiprocessor communication enable. In modes 2 and 3, SM2_1 enables the multiprocessor communication feature. If SM2_1 = 1 in mode 2 or 3, RI_1 will not be activated if the received 9th bit is 0. If SM2_1 = 1 in mode 1, RI_1 will only be activated if a valid stop is received. In mode 0, SM2_1 establishes the baud rate: when SM2_1 = 0, the baud rate is clk/ 12; when SM2_1 = 1, the baud rate is clk/4.															
SCON1.4	REN_1 – Receive enable. When REN_1 = 1, reception is enabled.															
SCON1.3	TB8_1 – Defines the state of the 9th data bit transmitted in modes 2 and 3.															
SCON1.2	RB8_1 – In modes 2 and 3, RB8_1 indicates the state of the 9th bit received. In mode 1, RB8_1 indicates the state of the received stop bit. In mode 0, RB8_1 is not used.															
SCON1.1	TI_1 – Transmit interrupt flag. Indicates that the transmit data word has been shifted out. In mode 0, TI_1 is set at the end of the 8th data bit. In all other modes, TI_1 is set when the stop bit is placed on the txd1 pin. TI_1 must be cleared by the software.															
SCON1.0	RI_1 – Receive interrupt flag. Indicates that a serial data word has been received. In mode 0, RI_1 is set at the end of the 8th data bit. In mode 1, RI_1 is set after the last sample of the incoming stop bit, subject to the state of SM2_1. In modes 2 and 3, RI_1 is set at the end of the last sample of RB8_1. RI_1 must be cleared by the software.															

Figure 3-7: Serial Port Mode 0 Receive Timing – Low Speed Operation

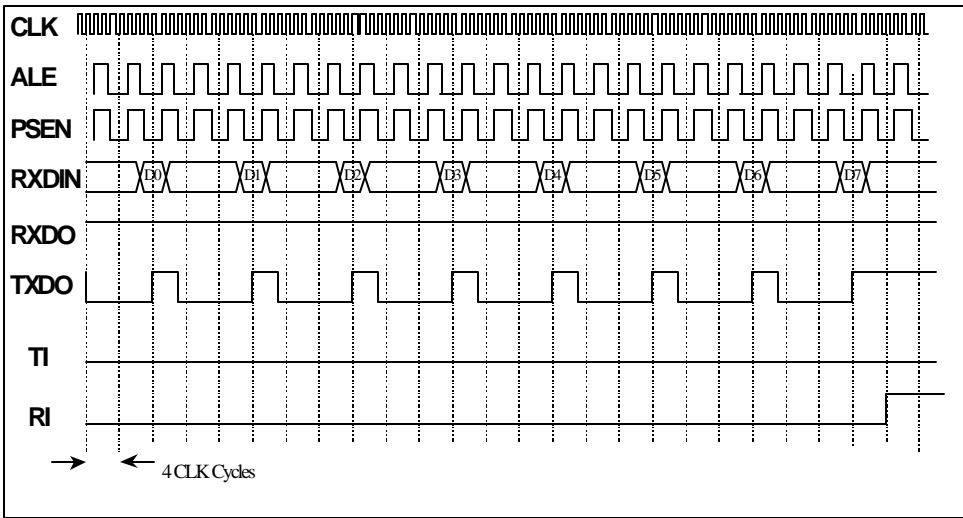


Figure 3-8: Serial Port Mode 0 Receive Timing – High Speed Operation

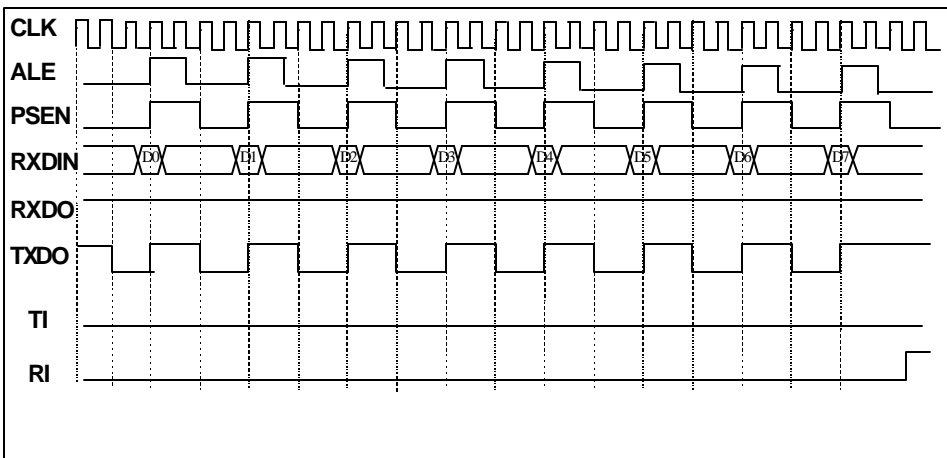


Figure 3-9: Serial Port Mode 0 Transmit Timing – Low Speed Operation

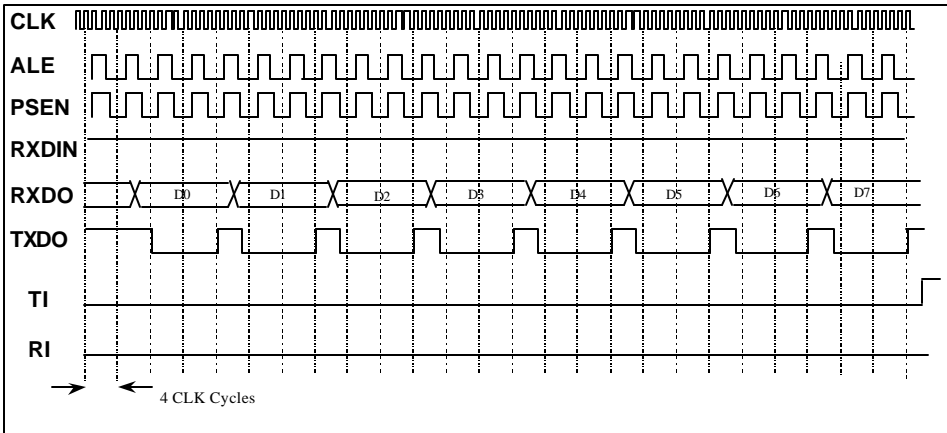
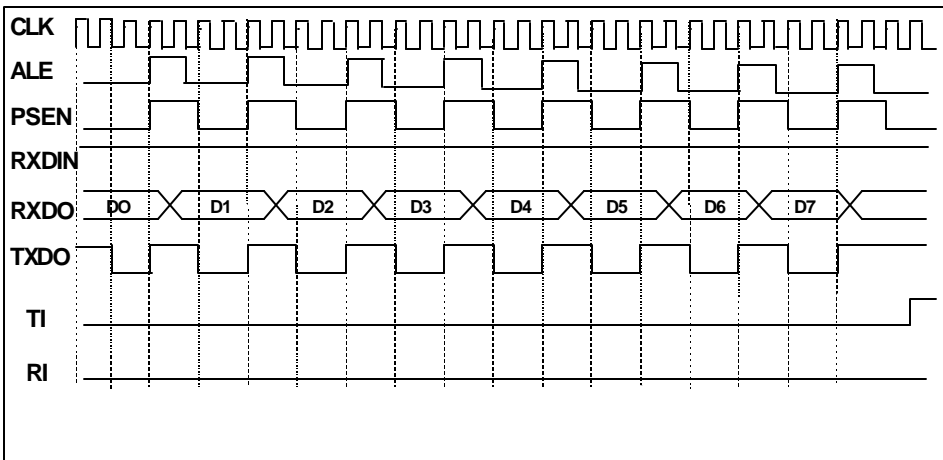


Figure 3-10: Serial Port Mode 0 Transmit Timing – High Speed Operation



Mode 1

Mode 1 provides standard asynchronous, full-duplex communication, using a total of 10 bits: 1 start bit, 8 data bits, and 1 stop bit. For receive operations, the stop bit is stored in RB8_0 (or RB8_1). Data bits are received and transmitted LSB first.

Mode 1 Baud Rate

The mode 1 baud rate is a function of timer overflow. Serial Port 0 can use either Timer 1 or Timer 2 to generate baud rates. Serial Port 1 can only use Timer 1. The two serial ports can run at the same baud rate if they both use Timer 1, or different baud rates if Serial Port 0 uses Timer 2 and Serial Port 1 uses Timer 1.

Each time the timer increments from its maximum count (FFH for Timer 1 or FFFFH for Timer 2), a clock is sent to the baud-rate circuit. The clock is then divided by 16 to generate the baud rate.

When using Timer 1, the SMOD0 (or SMOD1) bit selects whether or not to divide the Timer 1 rollover rate by 2. Therefore, when using Timer 1, the baud rate is determined by the equation:

$$\text{Baud_Rate} = \frac{2^{\text{SMODx}}}{32} \times \text{Timer1_Overflow}$$

SMOD0 is SFR bit PCON.7; SMOD1 is SFR bit EICON.7.

When using Timer 2, the baud rate is determined by the equation:

$$\text{Baud_Rate} = \frac{\text{Timer2_Overflow}}{16}$$

To use Timer 1 as the baud rate generator, it is best to use Timer 1 mode 2 (8-bit counter with auto-reload), although any counter mode can be used. The Timer 1 reload value is stored in the TH1 register, which makes the complete formula for Timer 1:

$$\text{Baud_Rate} = \frac{2^{\text{SMODx}}}{32} \times \frac{\text{clk}}{12 \times (256 - \text{TH1})}$$

The 12 in the denominator of the above equation can be changed to 4 by setting the T1M bit in the CKCON SFR. To derive the required TH1 value from a known baud rate (when TM1 = 0), use the equation:

$$\text{TH1} = 256 - \frac{2^{\text{SMODx}}}{384 \times \text{Baud_Rate}} \times \text{clk}$$

You can also achieve very low serial port baud rates from Timer 1 by enabling the Timer 1 interrupt, configuring Timer 1 to mode 1, and using the Timer 1 interrupt to initiate a 16-bit software reload. Table 3-11 lists sample reload values for a variety of common serial port baud rates.

Table 3-11: Timer 1 Reload Values for Common Serial Port Mode 1 Baud Rates

Desired Baud Rate	SMODx	C/T	Timer 1 Mode	TH1 Value for 11.0592-MHz clk
57.6 Kb/s	1	0	2	FFH
19.2 Kb/s	1	0	2	FDH
9.6 Kb/s	1	0	2	FAH
4.8 Kb/s	1	0	2	F4H
2.4 Kb/s	1	0	2	E8H
1.2 Kb/s	1	0	2	D0H

To use Timer 2 as the baud rate generator, configure Timer 2 in auto-reload mode and set the TCLK and/or RCLK bits in the T2CON SFR. TCLK selects Timer 2 as the baud rate generator for the transmitter; RCLK selects Timer 2 as the baud rate generator for the receiver. The 16-bit reload value for Timer 2 is stored in the RCAP2L and RCA2H SFRs, which makes the equation for the Timer 2 baud rate:

$$\text{Baud_Rate} = \frac{\text{clk}}{32 \times (65536 - [\text{RCAP2H}, \text{RCAP2L}])}$$

where RCAP2H,RCAP2L is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned number.

The 32 in the denominator is the result of the *clk* being divided by 2 and the Timer 2 overflow being divided by 16. Setting TCLK or RCLK to 1 automatically causes the *clk* to be divided by 2, as

shown in Figure 3-6, instead of the 4 or 12 determined by the T2M bit in the CKCON SFR.

To derive the required RCAP2H and RCAP2L values from a known baud rate, use the equation:

$$\text{RCAP2H, RCAP2L} = 65536 - \frac{\text{clk}}{32 \times \text{Baud_Rate}}$$

Table 3-12 lists sample values of RCAP2L and RCAP2H for a variety of desired baud rates.

Table 3-12: Timer 2 Reload Values for Common Serial Port Mode 1 Baud Rates

Baud Rate	C/T2	11.0592-MHz clk	
		RCAP2H	RCAP2L
57.6 Kb/s	0	FFH	FAH
19.2 Kb/s	0	FFH	EEH
9.6 Kb/s	0	FFH	DCH
4.8 Kb/s	0	FFH	B8H
2.4 Kb/s	0	FFH	70H
1.2 Kb/s	0	FEH	E0H

When either RCLK or TCLK is set, the TF2 flag will not be set on a Timer 2 rollover, and the *T2EX* reload trigger is disabled.

Mode 1 Transmit

Figure 3-11 illustrates the Mode 1 transmit timing. In Mode 1, the UART begins transmitting after the first rollover of the divide-by-16 counter, after the software writes to the SBUF0 (or SBUF1) register. The UART transmits data on the *TXD0* (or *TXD1*) pin in the following order: start bit, 8 data bits (LSB first), stop bit. The *TI_0* (or *TI_1*) bit is set 2 *clk* cycles after the stop bit is transmitted.

Mode 1 Receive

Figure 3-12 illustrates the Mode 1 receive timing. Reception begins at the falling edge of a start bit received on *RXD0* (or *RXD1*), when enabled by the *REN_0* (or *REN_1*) bit. For this purpose, *RXD0* (or *RXD1*) is sampled 16 times per bit for any baud rate. When a falling edge of a start bit is detected, the divide-by-16 counter used to generate the receive clock is reset to align the counter rollover to the bit boundaries.

For noise rejection, the serial port establishes the content of each received bit by a majority decision of three consecutive samples in the middle of each bit time. This is especially true for the start bit. If the falling edge on *RXD0* (or *RXD1*) is not verified by a majority decision of three consecutive samples (low), then the serial port stops reception and waits for another falling edge on *RXD0* (or *RXD1*).

At the middle of the stop bit time, the serial port checks for the following conditions:

- RI_0 (or RI_1) = 0
- If $SM2_0$ (or $SM2_1$) = 1, the state of the stop bit is 1. If $SM2_0$ (or $SM2_1$) = 0, the state of the stop bit doesn't matter.

If the above conditions are met, the serial port writes the received byte to the SBUF0 (or SBUF1) register, loads the stop bit into RB8_0 (or RB8_1), and sets the RI_0 (or RI_1) bit. If the above conditions are not met, the received data is lost, the SBUF register and RB8 bit are not loaded, and the RI bit is not set.

After the middle of the stop bit time, the serial port waits for another high-to-low transition on the *RXD0* or *RXD1* pin.

Mode 1 operation is identical to that of the 8051 when Timers 1 and 2 use *clk/12* (the default value).

Figure 3-11: Serial Port Mode 1 Transmit Timing

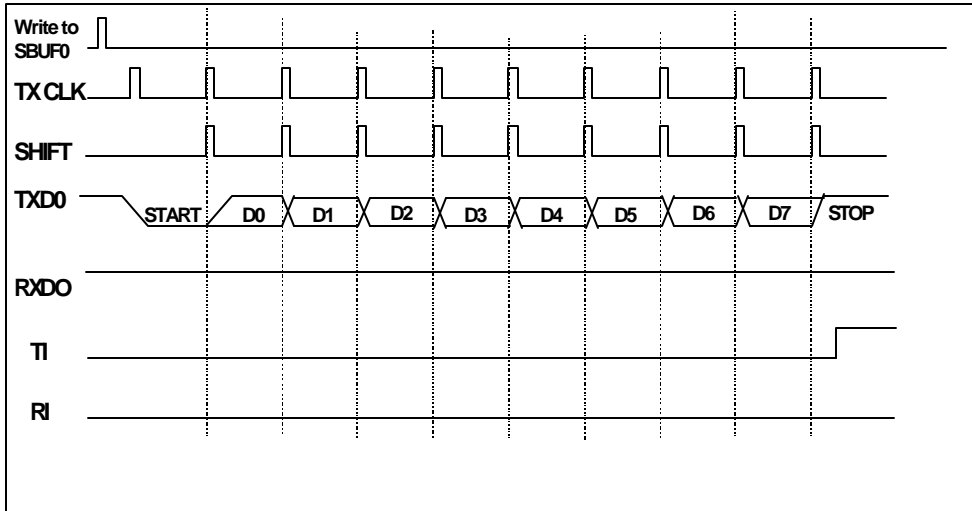
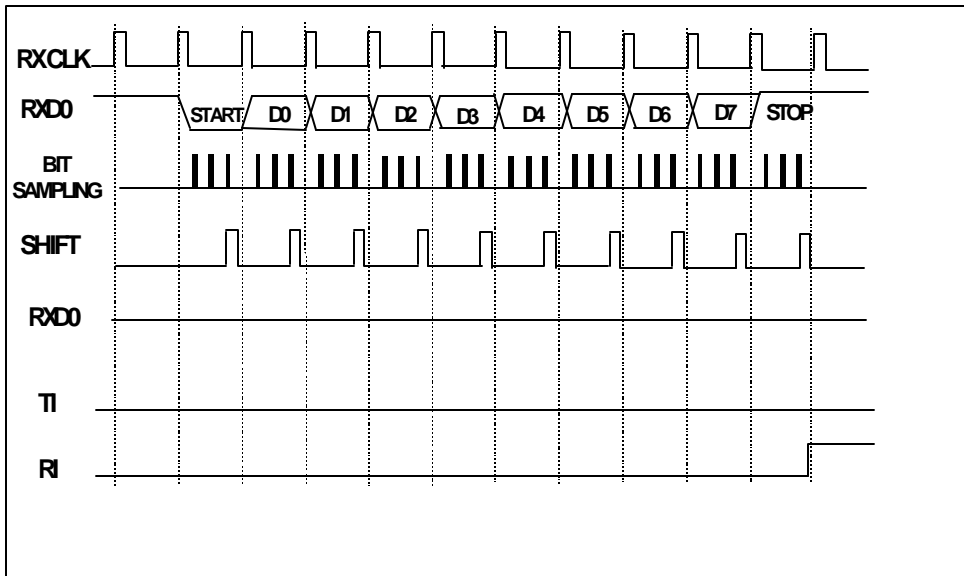


Figure 3-12: Serial Port 0 Mode 1 Receive Timing



Mode 2

Mode 2 provides asynchronous, full-duplex communication, using a total of 11 bits: 1 start bit, 8 data bits, a programmable 9th bit, and 1 stop bit. The data bits are transmitted and received LSB first. For transmission, the 9th bit is determined by the value in TB8_0 (or TB8_1). To use the 9th bit as a parity bit, move the value of the P bit (SFR PSW.0) to TB8_0 (or TB8_1).

The mode 2 baud rate is either $clk/32$ or $clk/64$, as determined by the SMOD0 (or SMOD1) bit. The formula for the mode 2 baud rate is:

$$\text{Baud_Rate} = \frac{2^{\text{SMODx}}}{64} \times \text{clk}$$

Mode 2 operation is identical to the standard 8051.

Mode 2 Transmit

Figure 3-13 illustrates the mode 2 transmit timing. Transmission begins after the first rollover of the divide-by-16 counter following a software write to SBUF0 (or SBUF1). The UART shifts data out on the TXD0 (or TXD1) pin in the following order: start bit, data bits (LSB first), 9th bit, stop bit. The TI_0 (or TI_1) bit is set when the stop bit is placed on the TXD0 (or TXD1) pin.

Mode 2 Receive

Figure 3-14 illustrates the Mode 2 receive timing. Reception begins at the falling edge of a start bit received on *RXD0* (or *RXD1*), when enabled by the *REN_0* (or *REN_1*) bit. For this purpose, *RXD0* (or *RXD1*) is sampled 16 times per bit for any baud rate. When a falling edge of a start bit is detected, the divide-by-16 counter used to generate the receive clock is reset to align the counter rollover to the bit boundaries.

For noise rejection, the serial port establishes the content of each received bit by a majority decision of three consecutive samples in the middle of each bit time. This is especially true for the start bit. If the falling edge on *RXD0* (or *RXD1*) is not verified by a majority decision of three consecutive samples (low), then the serial port stops reception and waits for another falling edge on *RXD0* (or *RXD1*).

At the middle of the stop bit time, the serial port checks for the following conditions:

- RI_0 (or RI1) = 0
- If SM2_0 (or SM2_1) = 1, the state of the stop bit is 1.
If SM2_0 (or SM2_1) = 0, the state of the stop bit doesn't matter.

If the above conditions are met, the serial port writes the received byte to the SBUF0 (or SBUF1) register, loads the 9th received bit into RB8_0 (or RB8_1), and sets the RI_0 (or RI_1) bit. If the above conditions are not met, the received data is lost, the SBUF register and RB8 bit are not loaded, and the RI bit is not set.

After the middle of the stop bit time, the serial port waits for another high-to-low transition on the *RXD0* or *RXD1* pin.

Figure 3-13: Serial Port 0 Mode 2 Transmit Timing

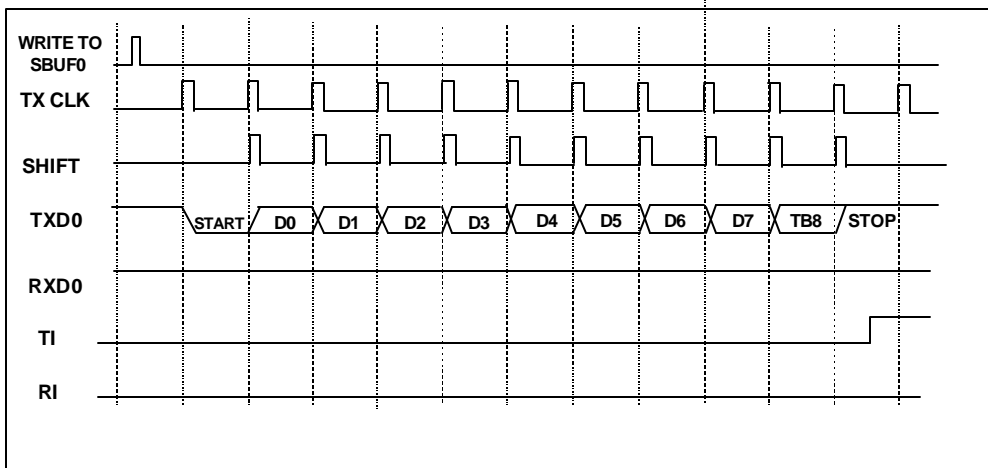
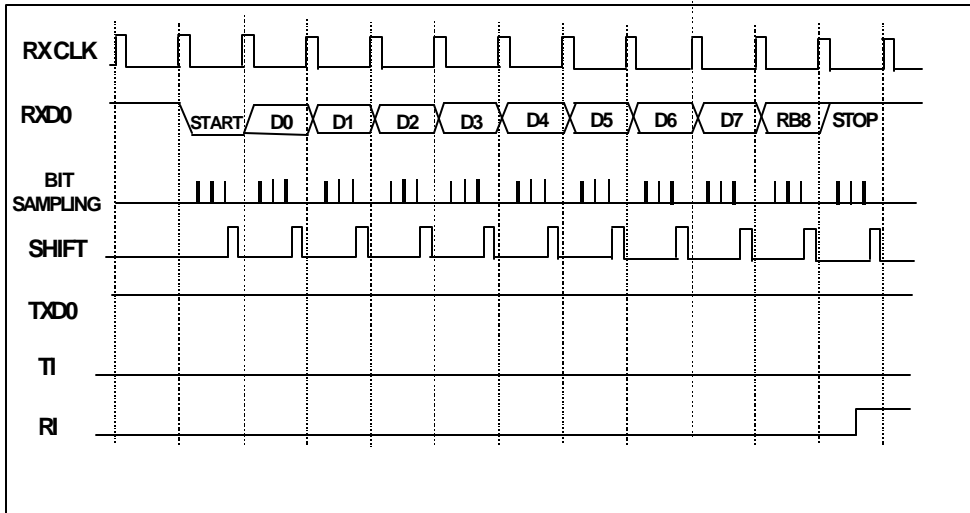


Figure 3-14: Serial Port 0 Mode 2 Receive Timing



Mode 3

Mode 3 provides asynchronous, full-duplex communication, using a total of 11 bits: 1 start bit, 8 data bits, a programmable 9th bit, and 1 stop bit. The data bits are transmitted and received LSB first.

The Mode 3 transmit and operations are identical to Mode 2. The Mode 3 baud-rate generation is identical to Mode 1. Mode 3 is a combination of Mode 2 protocol and Mode 1 baud-rate. Figure 3-15 illustrates the mode 3 transmit timing. Figure 3-16 illustrates the Mode 3 receive timing.

Mode 3 operation is identical to that of the standard 8051 when Timers 1 and 2 use $clk/12$ (the default).

Figure 3-15: Serial Port 0 Mode 3 Transmit Timing

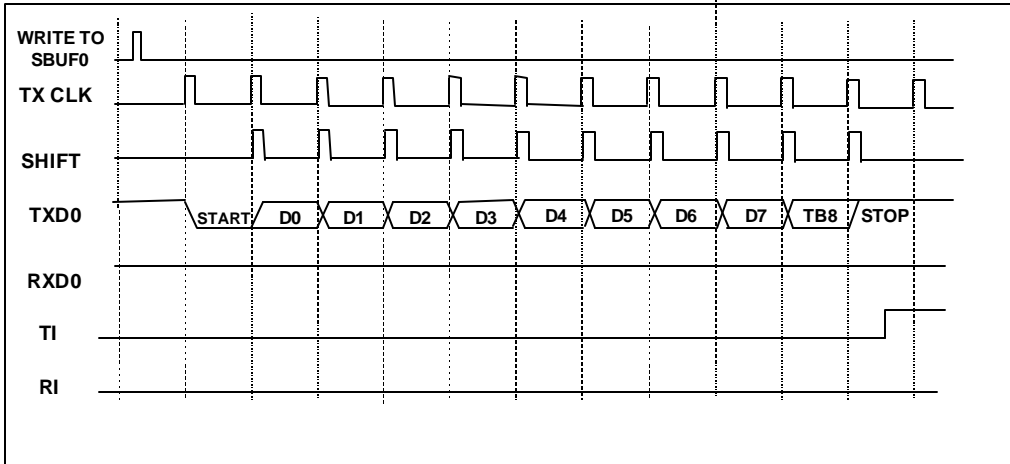
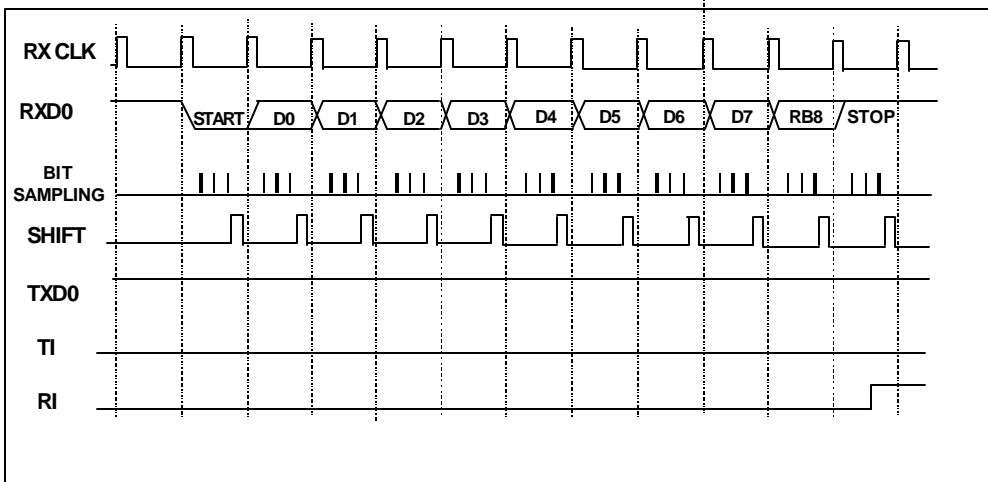


Figure 3-16: Serial Port 0 Mode 3 Receive Timing



Multiprocessor Communications

The multiprocessor communication feature is enabled in modes 2 and 3 when the SM2 bit is set in the SCON SFR for a serial port (SM2_0 for Serial Port 0, SM2_1 for Serial Port 1). In multiprocessor communication mode, the 9th bit received is stored in RB8_0 (or RB8_1) and, after the stop bit is received, the serial port interrupt is activated only if RB8_0 (or RB8_1) = 1.

A typical use for the multiprocessor communication feature is when a master wants to send a block of data to one of several slaves. The master first transmits an address byte that identifies the target slave. When transmitting an address byte, the master sets the 9th bit to 1; for data bytes, the 9th bit is 0.

When SM2_0 (or SM2_1) = 1, no slave will be interrupted by a data byte. However, an address byte interrupts all slaves so that each slave can examine the received address byte to determine whether that slave is being addressed. Address decoding must be done by software during the interrupt service routine. The addressed slave clears its SM2_0 (or SM2_1) bit and prepares to receive the data bytes. The slaves that are not being addressed leave the SM2_0 (or SM2_1) bit set and ignore the incoming data bytes.

A/D Converter SFR Registers

Figure 3-17 illustrates the equivalent input networks for the external reference pins and the analog inputs, as well as the switched-capacitor timing for these pins. The external reference pins see a 1.84 pF load which is reversed every eight CPU clock periods as shown by waveforms 1 and 2. This load condition is present whenever the CPU clock is running and reset is not asserted, and represents the dominant loading of the external reference pins. There is also a negligible per-sample load seen at the external reference pins (see following description).

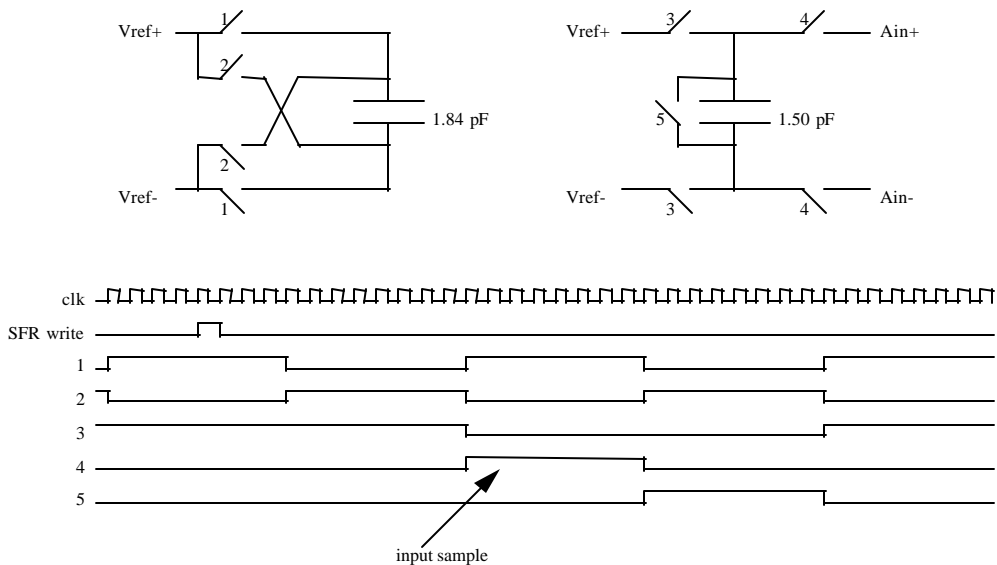
Both the reference pins and the analog inputs drive a 1.50 pF load on each ADC conversion. For the external reference, the 1.50 pF capacitor is always initially discharged, and for the analog input, the 1.50 pF capacitor is always initially charged to the full reference potential. This operation is shown by waveforms 3, 4, and 5.

The analog inputs are connected to the 1.50 pF load for eight CPU clock periods. This sampling operation begins in the 12th CPU clock cycle following the write to ADCON (SFR address C5H, shown as “SFR write” in figure 3-17) which initiates the conversion. In the case of continuous conversion (CCVT set in ADCON), the sampling process is repeated every 224 CPU clock periods. The conversion complete flag (CC bit of ADCON) is asserted in the 215th CPU clock cycle following the SFR write which initiates the conversion. Interrupt 3, if enabled, is generated at the same time as CC is asserted. If continuous conversion is

enabled, CC and interrupt 3 (if enabled) will be asserted every 224 CPU clock periods.

In Figure 3-17, Ain+ and Ain- represent the positive and negative analog inputs appropriate for the analog input multiplexer setting. In the case of single-ended conversions, Ain- is connected to ACOM.

Figure 3-17: A/D Converter Equivalent Input/Timing



ADCON - (SFR address C5H) A/D Control and status register. A *write* to the ADCON register will initiate an A/D conversion cycle, or start continuous conversions if bit 6 (CCVT) is set. If continuous conversions are active, a write with bit 6 clear will cause conversions to stop after the current conversion.

Bit 7				Bit 0			
CC	CVT	CCIE	OVRN	M3	M2	M1	M0

CC - Conversion Complete Flag; read only. A one in Bit 7 indicates the A/D converter has latched valid data into the ADDAT0-ADDAT1 data registers and is available for reading by the CPU. The CC bit remains set until ADDAT1 (SFR address C3H) is read, at which point it is cleared.

CCVT - Continuous Conversion enable; read/write. When CCVT=1 continuous conversions will be initiated. Conversions will continue at the maximum rate until the part is reset, or until CCVT is cleared, in which case the current conversion will be completed and then conversions will stop.

CCIE - Conversion Complete Interrupt Enable; read/write. When CCIE=1 interrupt 3 will be generated each time the A/D converter has completed a conversion cycle and latched valid data into ADDAT0-ADDAT2 registers.

OVRN - Overrun Flag; read only. A one in Bit 4 indicates that the A/D converter has latched new valid data into the ADDAT0-ADDAT1 data registers before ADDAT1 of the previous conversion result was read. The OVRN bit remains set until ADDAT1 is read, at which point it is cleared.

ADDAT1 - (SFR address C3H) bits 11 through 4 of A/D converter output. The upper byte of the (11 bit + sign) 12-bit A/D conversion result. Reading this register clears the CC and OVRN flags in ADCON (SFR address C5H).

Bit 7							Bit 0
Sign	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4

ADDAT0 - (SFR address C2H) bits 3 through 0 of A/D converter output. The result is zero-padded on the right.

Bit 7							Bit 0
bit 3	bit 2	bit 1	bit 0	0	0	0	0

Watchdog Timer/Registers

For applications that cannot afford to run out-of-control the MAX765x incorporates a programmable Watchdog Timer (WDT) feature. The WDT is programmed to generate an interrupt after counting clock cycles (it also sets flag WDIF). The software then has 512 clocks (about 60 instruction cycles) to reset the WDT, else a CPU RESET will occur. It is important to note that the WDT can **only** be reset **after** timeout: writing to the flags before a timeout **will not reset the WDT**.

Regardless of whether the user enables this interrupt, *there are 512 oscillator clocks remaining until a processor RESET occurs*. Time-out values can be one of four sub-divides off the crystal oscillator. The table below summarizes the time-out delay values for a crystal frequency of 12 MHz:

WD 1	WD 0	Interrupt timeout	Time	Reset timeout	Time
0	0	2^{16} clocks	5.461 ms	$2^{16} + 512$ clocks	5.474 ms
0	1	2^{19} clocks	43.691 ms	$2^{19} + 512$ clocks	43.734 ms
1	0	2^{22} clocks	349.52 ms	$2^{22} + 512$ clocks	349.567 ms
1	1	2^{25} clocks	2796.0 ms	$2^{25} + 512$ clocks	2796.04 ms

There are five control bits in special function registers (SFR's) that affect the Watchdog Timer and two status flags that report to the user.

The following table summarizes all status and control bits associated with the WDT and their respective register locations:

WDT-related bit name	SFR Location	Function:	SFR Address
WDIF (WDTI)	EICON.3	Status	D8
WTRF	WDT.2	Status/Control	DD
EWT	EICON.1	Control	D8
RWT	EICON.0	Control	D8
WD1	CKCON.7	Control	8E
WD0	CKCON.6	Control	8E
EWDI	EIE.4	Control	E8

The WDIF bit is a readable status bit indicating that the Watchdog counter has reached the final count and has generated a Watchdog interrupt.

The WTRF bit is a status/control bit indicating that the Watchdog counter has counted an additional 512 clocks past the WDT interrupt and has generated a processor RESET. The reset routine should check the WTRF flag to determine the source of the reset. The Watchdog Timer will be reset when a zero is written to the WTRF flag. This allows the processor to re-gain synchronization with the WDT. The WTRF flag is also cleared when a zero is written to it.

The EWT bit is a writeable control bit that enables the WDT. A one enables the WDT. The RWT bit is a writeable control bit that, when set to one, will reset the WDT count *after timeout*. The WDT will respond to RWT assertion only after the selected time interval has expired and WDIF=1.

Analog Input Mux

There are eight analog input pins on the MAX765x, labeled AN7 to AN0. The eight pins can also be selected as four pairs of differential inputs: AN1 & AN0, AN3 & AN2, etc. There is also a selection to measure the differential voltage of the two reference inputs, VREF+ and VREF-. Single-ended conversions are referenced to an analog common (ACOM) which is the bipolar “zero” reference pin.

The mux is controlled by 4 bits M3-M0 in the SFR ADCON (address C5H). Bit M3 determines if the inputs are single-ended (M3=0) or differential (M3=1). See Table 3-13.

These four bits select the analog input to be digitized on the next A/D conversion cycle. When M3=1 the analog inputs are configured as three sets of input *pairs*. The A/D converter will convert the difference in voltage between the specified input pairs, i.e. AIN1-AIN0, AIN3-AIN2, up to AIN7-AIN6 thus accomodating differential input signals. The ACOM pin is the bipolar zero reference pin. The differential signal input range is +-VREF (where VREF is the differential voltage between the VREF+ and VREF- pins). The single-ended signal range is +-VREF/2 above and below ACOM giving signed binary output from the A/D converter (VREF is again the differential value).

Table 3-13: Analog MUX Control Bits

M3	M2	M1	M0	Active analog inputs
0	0	0	0	AIN0-ACOM
0	0	0	1	AIN1-ACOM
0	0	1	0	AIN2-ACOM
0	0	1	1	AIN3-ACOM
0	1	0	0	AIN4-ACOM
0	1	0	1	AIN5-ACOM
0	1	1	0	AIN6-ACOM
0	1	1	1	AIN7-ACOM
1	0	0	0	AIN1-AIN0 (differential)
1	0	0	1	AIN3-AIN2 (differential)
1	0	1	0	AIN5-AIN4 (differential)
1	0	1	1	AIN7-AIN6 (differential)
1	1	0	0	(VREF+) - (VREF-)

FLASH Programming Registers

The MAX765x contains two 8K byte blocks of CPU-writeable FLASH memory (respectively called UPPER and LOWER). Code execution can take place out of *either* 8K FLASH array. In addition both arrays can be written/erased/read via the EESTCMD, EEAH, EEAL, and EEDAT registers in the SFR map **only at supply voltages of 3.0V or greater**. The user can execute code out of one FLASH block while writing, erasing or reading from the other FLASH array.

Read operations occur as fast as the CPU can write the address registers EEAH & EEAL while FLASH writes and page erases are *much slower* due to FLASH technology writing speeds. Also, you cannot write to a FLASH location **more than twice** without page/bulk erasing first.

EESTCMD- (SFR address EDH) FLASH command/status register. When read, EESTCMD contains ready flags which indicate that each FLASH array is ready to accept user commands. EESTCMD accepts three written commands: *read*, *write*, and *page erase*. Both FLASH memories are organized as 128 64-byte *pages*. To set up a FLASH *write* cycle the user writes the high byte and low byte address to EEAH and EEAL respectively. (Bit 7 of EEAH selects either the UPPER or LOWER FLASH array). Then the program writes the desired data to EEDAT. To invoke a FLASH *write* cycle, 55H is written to EESTCMD. This action causes the RDY flag in question to be cleared (zero). When the on-board FLASH write logic has completed the user-initiated write cycle RDYHI/RDYLO is

returned to a “1” signifying that the addressed FLASH is ready for another command. *Write cycles typically require $T_{clk} + 72\mu s$ to complete.*

Reading of FLASH takes place by writing the address bytes to EEAL and EEAH, followed by writing a “read” command byte (AAH) to EESTCMD. The Data FLASH read data will then be available in the next CPU instruction cycle in the EEDAT register. As a result, RDYHI/RDYLO need not be tested for FLASH *read* cycles. To *erase* a FLASH page the user only writes EEAH (effectively the FLASH “page” register) and then writes 5AH to EESTCMD. This action causes the DYHI/RDYLO bit to be cleared (depending of the value of EEAH7). Again EEAH7 selects which 8K FLASH will receive the command while EEAH6-EEAH0 selects the specific page within the FLASH array. When the page erase cycle is completed the RDYHI/RDYLO flag will return to a “1”. In general, **never** write a new command until RDY returns to a “1”.

Also the user **must not attempt** to read/write/page erase the FLASH block in which the CPU is currently executing program instructions. For example, to program the UPPER FLASH the CPU must be executing code from the LOWER FLASH and vice versa. For ease of use, either RDYHI or RDYLO will generate an “Interrupt 2” (priority level 8) with vector at address 43H. Interrupt 2 sets flag EXIF.4, is enabled by EIE.0, and has priority controlled by EIP.0. EXIF.4 must be manually cleared in software upon entering the interrupt routine. (See the EXIF, EIE, and EIP registers for more detail).

Bit 7							Bit 0
RDYHI	RDYLO	0	0	0	0	0	0

EEAL - (SFR address EAH) FLASH *low byte address*. The user-written program writes the *lower* address byte to this register for FLASH SFR access cycles. Effectively the address within the FLASH page. There are 64 bytes in a FLASH page.

Bit 7							Bit 0
-	-	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0

EEAH - (SFR address EBH) FLASH *high byte address*. The user written program writes the *high* address byte to this register for FLASH SFR access cycles. This register selects both the UPPER/LOWER array as well as the page within the array. There are a total of 128 FLASH pages. EEAH7 selects either the UPPER or LOWER array and EEAH6-EEAH0 selects the page within the array. For security reasons this register is “frozen” until the RDYHI/RDYLO bit returns to a “1”.

Bit 7							Bit 0
EEAH7	EEAH6	EEAH5	EEAH4	EEAH3	EEAH2	EEAH1	EEAH0

EEDAT - (SFR address ECH) FLASH data register. EEDAT either accepts (SFR write to EEDAT) or contains (SFR read from

EEDAT) FLASH memory data after either a write or read command has been written to the EESTCMD register. Do not write to this register if the RDY flag is not “1”. This register is not used during a *page erase* cycle.

Bit 7**Bit 0**

EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
--------	--------	--------	--------	--------	--------	--------	--------

Pulse-Width Modulators

The MAX765x contains two independent pulse-width modulators (PWM). A PWM output is a fixed-frequency, variable duty-cycle clock. SFR registers program the PWM frequency and the duty-cycle of the waveform.

PWM is mainly used for generating control voltages, where absolute precision is not required (such as LCD contrast). In order to generate a voltage, the user must low-pass filter the output of the PWM modulator. The filter can be as simple as a R-C network, or an active LPF using opamps. Tradeoffs include cost, board area, power, and ripple of the output voltage. In some cases, the PWM clock is used directly (to drive a speaker, for example).

SFR Register Definitions for Pulse Width Modulator

PWPS - (SFR address DAH, W). Pulse-width modulator prescaler register (8 bits). The pulse-width prescaler register divides down the crystal oscillator by (2 times value+1) loaded into PWPS and provides this prescaled clock to the modulo 255 PWM counter (See Figure 3-18). *This SFR is write-only!*

Bit 7							Bit 0
PWPS7	PWPS6	PWPS5	PWPS4	PWPS3	PWPS2	PWPS1	PWPS0

PWDA - (SFR address DBH, R/W), Pulse-width modulator A data (8 bits). Writing a 00H will result in a 100% duty cycle pulse at the PWMA output pin (always 1). Writing FFH will result in a 0% (always 0) at the PWM output pin P1.6. Writing any value between 01H and FEH will result in a pulse train of duty cycle proportional to the written value (see Figure 3-18).

Bit 7							Bit 0
PWDA7	PWDA6	PWDA5	PWDA4	PWDA3	PWDA2	PWDA1	PWDA0

PWDB- (SFR address DCH, R/W). Pulse-width modulator B data (8 bits). Writing a 00H will result in a 100% duty cycle pulse at the PWMB output pin (always 1). Writing FFH will result in a 0% (always 0) at the PWM output pin P1.7. Writing any value between 01H and FEH will result in a pulse train of duty cycle proportional to the written value. (see Figure 3-18).

Bit 7							Bit 0
PWDB7	PWDB6	PWDB5	PWDB4	PWDB3	PWDB2	PWDB1	PWDB0

PWMC- (SFR address FEH, W). Pulse-width modulator control register. The PWMA and PWMB outputs are enabled by the PWENA and PWENB bits. The divide by two, PWPS prescaler, and modulo-255 counter clock can be stopped by setting the PWON bit to zero. *Register is write-only.*

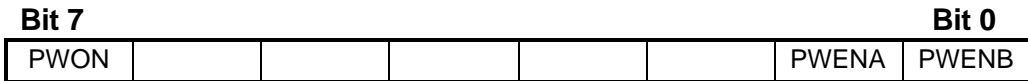


Figure 3-18: PWM Duty Cycle Calculation

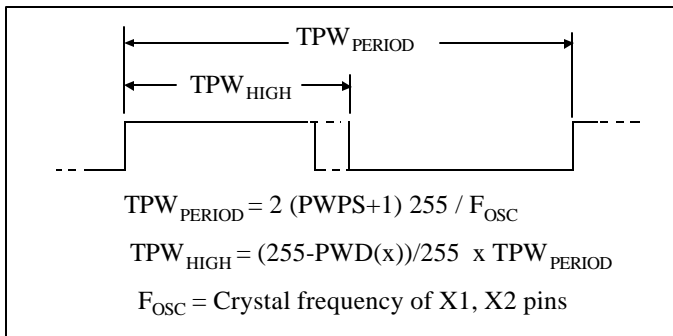
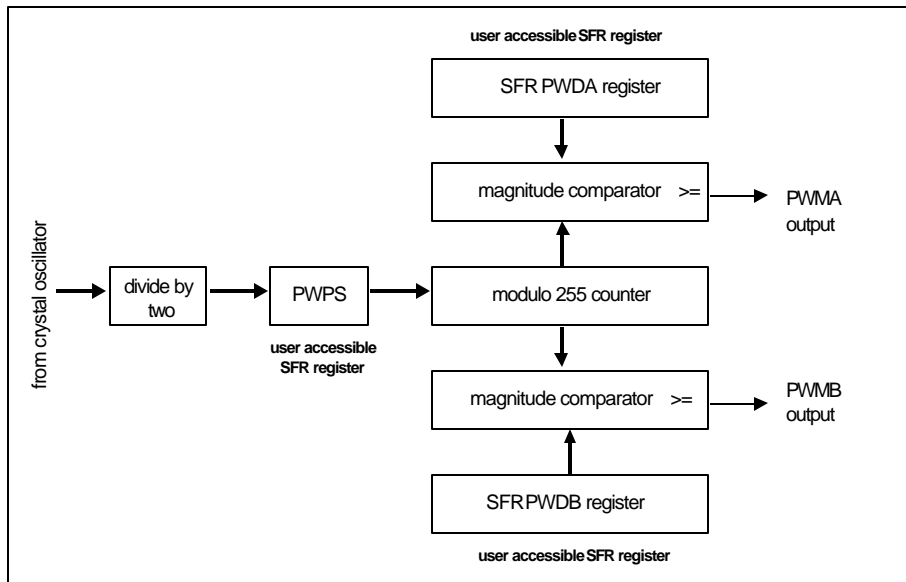


Figure 3-19: PWM Register Block Diagram



SFR Registers

The MAX765x accesses internal and external peripherals by using Special Function Registers (SFRs). The internal peripherals include the timer/counters, SFR RAM, interrupt unit, and serial ports.

Bit Addressing

Bit addressing in SFR space is available at all SFR addresses that end with 0 or 8 **except EICON**. Table 3-14 lists the bit-addressable SFRs.

Table 3-14: Bit-Addressable SFRs

SFR Address	Usage
88H	TCON
98H	SCON0
A8H	IE
B8H	IP
C0H	SCON1
C8H	T2CON
D0H	PSW
D8H	EICON (<i>not bit addressable</i>)
E0H	ACC
E8H	EIE
F0H	B
F8H	EIP

Interrupts

Sources of interrupts on the MAX765x are as follows:

- $\sim INT0$ – External interrupt, active low, configurable as edge- sensitive or level-sensitive
- $\sim INT1$ – External interrupt, active low, configurable as edge- sensitive or level-sensitive
- TF0 – Timer 0 interrupt
- TF1 – Timer 1 interrupt
- TF2 or EXF2 – Timer 2 interrupt
- TI_0 or RI_0 – Internal receive/transmit interrupt from Serial Port 0
- TI_1 or RI_1 – Internal receive/transmit interrupt from optional Serial Port 1
- ADC End-of-conversion interrupt
- FLASH Write/Page Erase interrupt
- WDTI – Internal Watchdog-timer interrupt, edge-sensitive, active high

803x/805x Compatibility

Table 3-15: Interrupt Compatibility for MAX765x

Feature	8051	MAX765x
External interrupt 0	implemented	implemented
Timer 0 interrupt	implemented	implemented
External interrupt 1	implemented	implemented
Timer 1 interrupt	implemented	implemented
Serial port 0 interrupt	implemented	implemented
Timer 2 interrupt	not implemented	implemented
Serial port 1 interrupt	not implemented	implemented
ADC interrupt	not implemented	implemented
FLASH interrupt	not implemented	implemented
Watchdog timer interrupt	not implemented	implemented

Interrupt SFRs

The following SFRs are associated with interrupt control:

- IE – SFR A8H (Table 3-15)
- IP – SFR B8H (Table 3-16)
- EXIF – SFR 91H (Table 3-17)
- EICON – SFR D8H (Table 3-18)
- EIE – SFR E8H (Table 3-19)
- EIP – SFR F8H (Table 3-20)

The IE and IP SFRs provide interrupt enable and priority control. Additionally, these SFRs provide control bits for the Serial Port 1 interrupt.

The EXIF, EICON, EIE, and EIP registers provide flags, enable control, and priority control for the extended interrupts in the MAX765x.

Table 3-16: IE Register – SFR A8H

Bit	Function
IE.7	EA – Global interrupt enable. Controls masking of all interrupts. EA = 0 disables all interrupts (EA overrides individual interrupt enable bits). When EA = 1, each interrupt is enabled or masked by its individual enable bit.
IE.6	ES1 – Enable Serial Port 1 interrupt. ES1 = 0 disables Serial Port 1 interrupts (TI_1 and RI_1).
IE.5	ET2 – Enable Timer 2 interrupt. ET2 = 0 disables Timer 2 interrupt (TF2). ET2 = 1 enables interrupts generated by the TF2 or EXF2 flag.
IE.4	ES0 – Enable Serial Port 0 interrupt. ES0 = 0 disables Serial Port 0 interrupts (TI_0 and RI_0). ES0 = 1 enables interrupts generated by the TI_0 or RI_0 flag.
IE.3	ET1 – Enable Timer 1 interrupt. ET1 = 0 disables Timer 1 interrupt (TF1). ET1 = 1 enables interrupts generated by the TF1 flag.
IE.2	EX1 – Enable external interrupt 1. EX1 = 0 disables external interrupt 1 (\sim INT1). EX1 = 1 enables interrupts generated by the \sim INT1 pin.
IE.1	ET0 – Enable Timer 0 interrupt. ET0 = 0 disables Timer 0 interrupt (TF0). ET0 = 1 enables interrupts generated by the TF0 flag.
IE.0	EX0 – Enable external interrupt 0. EX0 = 0 disables external interrupt 0 (\sim INT0). EX0 = 1 enables interrupts generated by the \sim INT0 pin.

Table 3-17: IP Register – SFR B8H

Bit	Function
IP.7	Reserved. Read as 1.
IP.6	PS1 – Serial Port 1 interrupt priority control. PS1 = 0 sets Serial Port 1 interrupt (TI_1 or RI_1) to low priority. PS1 = 1 sets Serial Port 1 interrupt to high priority.
IP.5	PT2 – Timer 2 interrupt priority control. PT2 = 0 sets Timer 2 interrupt (TF2) to low priority. PT2 = 1 sets Timer 2 interrupt to high priority.
IP.4	PS0 – Serial Port 0 interrupt priority control. PS0 = 0 sets Serial Port 0 interrupt (TI_0 or RI_0) to low priority. PS0 = 1 sets Serial Port 0 interrupt to high priority.
IP.3	PT1 – Timer 1 interrupt priority control. PT1 = 0 sets Timer 1 interrupt (TF1) to low priority. PT1 = 1 sets Timer 1 interrupt to high priority.
IP.2	PX1 – External interrupt 1 priority control. PX1 = 0 sets external interrupt 1 (\sim INT1) to low priority. PT1 = 1 sets external interrupt 1 to high priority.
IP.1	PT0 – Timer 0 interrupt priority control. PT0 = 0 sets Timer 0 interrupt (TF0) to low priority. PT0 = 1 sets Timer 0 interrupt to high priority.
IP.0	PX0 – External interrupt 0 priority control. PX0 = 0 sets external interrupt 0 (\sim INT0) to low priority. PT0 = 1 sets external interrupt 0 to high priority.

Table 3-18: EXIF Register – SFR 91H

Bit	Function
EXIF.7	Reserved.
EXIF.6	Reserved.
EXIF.5	IE3 – A/D converter EOC (end of conversion). IE3 = 1 indicates that a ADC conversion has completed. IE3 must be cleared by software. Setting IE3 in software generates an interrupt, if enabled.
EXIF.4	IE2 – FLASH Write/Page Erase Done. IE2 = 1 indicates that a FLASH Write or a Page Erase has completed. IE2 must be cleared by software. Setting IE2 in software generates an interrupt, if enabled.
EXIF.3	Reserved. Read as 1.
EXIF.2–0	Reserved. Read as 0.

Table 3-19: EICON Register – SFR D8H

Bit	Function
EICON.7	SMOD1 – Serial Port 1 baud rate doubler enable. When SMOD1 = 1, the baud rate for Serial Port 1 is doubled.
EICON.6	Reserved. Read as 1.
EICON.5	Reserved.
EICON.4	Reserved.
EICON.3	WDTI – Watchdog timer interrupt flag. WDTI = 1 indicates a Watchdog timer interrupt was detected. WDTI must be cleared by software before exiting the interrupt service routine. Otherwise, the interrupt occurs again. Setting WDTI in software generates a watchdog timer interrupt, if enabled.
EICON.2	Reserved.
EICON.1	EWT - Enable WDT. EWT = 1 enables the watchdog timer.
EICON.0	RWT - Reset WDT. RWT = 1 resets the WDT only if a WDT interrupt has occurred . If the RWT flag is not set within 512 clock cycles of a WDT interrupt, the WRTF flag will be set and

the CPU will reset.

Table 3-20: EIE Register – SFR E8H

Bit	Function
EIE.7–5	Reserved. Read as 1.
EIE.4	EWDI – Enable watchdog timer interrupt. EWDI = 0 disables watchdog timer interrupt (WDTI). EWDI = 1 enables interrupts.
EIE.3	Reserved.
EIE.2	Reserved.
EIE.1	EX3 – Enable A/D converter EOC interrupt. EX3 = 0 disables the interrupt.
EIE.0	EX2 – Enable FLASH interrupt. EX2 = 0 disables the interrupt.

Table 3-21: EIP Register – SFR F8H

Bit	Function
EIP.7–5	Reserved. Read as 1.
EIP.4	PWDI – Watchdog timer interrupt priority control. WDPI = 0 sets watchdog timer interrupt (WDTI) to low priority. PS0 = 1 sets watchdog timer interrupt to high priority.
EIP.3	Reserved.
EIP.2	Reserved.
EIP.1	PX3 – A/D converter EOC interrupt priority control. PX3 = 0 sets ADC interrupt to low priority. PX3 = 1 sets interrupt to high priority.
EIP.0	PX2 – FLASH interrupt priority control. PX2 = 0 sets interrupt to low priority. PX2 = 1 sets interrupt to high priority.

Interrupt Processing

When an enabled interrupt occurs, the CPU vectors to the address of the interrupt service routine (ISR) associated with that interrupt, as listed in Table 3-22. The CPU executes the ISR to completion unless another interrupt of higher priority occurs. Each ISR ends with a `RETI` (return from interrupt) instruction. After executing the `RETI`, the CPU returns to the next instruction that would have been executed if the interrupt had not occurred.

An ISR can only be interrupted by a higher priority interrupt. That is, an ISR for a low-level interrupt can only be interrupted by high-level interrupt. An ISR for a high-level interrupt can only be interrupted by the power-fail interrupt (extended interrupt unit only).

The MAX765x always completes the instruction in progress before servicing an interrupt. If the instruction in progress is `RETI` or a write access to any of the IP, IE, EIP, or EIE SFRs, the MAX765x completes one additional instruction before servicing the interrupt.

Interrupt Masking

The EA bit in the IE SFR (IE.7) is a global enable for all interrupts. When EA = 1, each interrupt is enabled/masked by its individual enable bit. When EA = 0, all interrupts are masked.

Table 3-23 provides a summary of interrupt sources, flags, enables, and priorities.

Table 3-22: Interrupt Natural Vectors and Priorities

Interrupt	Description	Natural Priority	Interrupt Vector
~INT0	External interrupt 0	1	03H
TF0	Timer 0 interrupt	2	0BH
~INT1	External interrupt 1	3	13H
TF1	Timer 1 interrupt	4	1BH
TI_0 or RI_0	Serial Port 0 transmit or receive	5	23H
TF2 or EXF2	Timer 2 interrupt	6	2BH
TI_1 or RI_1	Serial Port 1 transmit or receive	7	3BH
FLASH	FLASH Write/Page Erase	8	43H
ADC	A/D converter EOC	9	4BH
N/A	Reserved	10	53H
N/A	Reserved	11	5BH
WDTI	Watchdog timer interrupt	12	63H

Interrupt Priorities

There are two stages of interrupt priority assignment, interrupt level and natural priority. The interrupt level (highest, high, or low) takes precedence over natural priority. Interrupts can be assigned either high or low priority.

In addition to an assigned priority level (high or low), each interrupt has a natural priority, as listed in Table 3-22.

Simultaneous interrupts with the same priority level (for example, both high) are resolved according to their natural priority. For example, if \sim INT0 and T2 are both programmed as high priority, \sim INT0 takes precedence.

Once an interrupt is being serviced, only an interrupt of higher priority level can interrupt the service routine of the interrupt currently being serviced.

Table 3-23: Interrupt Flags, Enables, and Priority Control

Interrupt	Description	Flag	Enable	Priority Control
\sim INT0	External interrupt 0	TCON.1	IE.0	IP.0
TF0	Timer 0 interrupt	TCON.5	IE.1	IP.1
\sim INT1	External interrupt 1	TCON.3	IE.2	IP.2
TF1	Timer 1 interrupt	TCON.7	IE.3	IP.3
TI_0 or RI_0	Serial Port 0 transmit or receive	SCON0.0 (RI_0), SCON0.1 (TI_0)	IE.4	IP.4
TF2 or EXF2	Timer 2 interrupt	T2CON.7 (TF2), T2CON.6 (EXF2)	IE.5	IP.5
TI_1 or RI_1	Serial Port 1 transmit or receive	SCON1.0 (RI_1), SCON1.1 (TI_1)	IE.6	IP.6
FLASH	FLASH Write/Page Erase	EXIF.4	EIE.0	EIP.0
ADC	A/D converter EOC	EXIF.5	EIE.1	EIP.1
WDTI	Watchdog timer interrupt	EICON.3	EIE.4	EIP.4

Interrupt Sampling

The internal timers and serial ports generate interrupts by setting their respective SFR interrupt flag bits. The MAX765x samples

external interrupts once per instruction cycle, at the rising edge of *clk* at the end of cycle C4.

$\sim INT0$ and $\sim INT1$ are both active low and can be programmed to be either edge-sensitive or level-sensitive, through the IT0 and IT1 bits in the TCON SFR. For example, when IT0 = 0, $\sim INT0$ is level-sensitive and the MAX765x sets the IE0 flag when the $\sim INT0$ pin is sampled low. When IT0 = 1, $\sim INT0$ is edge-sensitive and the MAX765x sets the IE0 flag when the $\sim INT0$ pin is sampled high then low on consecutive samples.

The watchdog timer (WDTI) interrupt is sampled once per instruction cycle.

To ensure that edge-sensitive interrupts are detected, the corresponding ports should be held high for 4 *clk* cycles and then low for 4 *clk* cycles. Level-sensitive interrupts are not latched and must remain active until serviced.

Interrupt Latency

Interrupt response time depends on the current state of the MAX765x. The fastest response time is five instruction cycles: one to detect the interrupt, and four to perform the `LCALL` to the ISR.

The maximum latency (13 instruction cycles) occurs when the MAX765x is currently executing a `RETI` instruction followed by a `MUL` or `DIV` instruction. The 13 instruction cycles in this case are: 1 to detect the interrupt, 3 to complete the `RETI`, 5 to execute the `DIV` or `MUL`, and 4 to execute the `LCALL` to the ISR. For the maximum latency case, the response time is $13 \times 4 = 52 \text{ clk}$ cycles.

Single-Step Operation

The MAX765x interrupt structure provides a way to perform single-step program execution. When exiting an ISR with an `RETI` instruction, the MAX765x will always execute at least one instruction of the task program. Therefore, once an ISR is entered, it cannot be re-entered until at least one program instruction is executed.

To perform single-step execution, program one of the external interrupts (for example, `~INT0`) to be level-sensitive and write an ISR for that interrupt that terminates as follows:

```
JNB  TCON.1,$      ; wait for high on ~INT0
JB   TCON.1,$      ; wait for low on ~INT0
RETI                          ; return for ISR
```

The CPU enters the ISR when $\sim INT0$ goes low, then waits for a pulse on $\sim INT0$. Each time $\sim INT0$ is pulsed, the CPU exits the ISR, executes one program instruction, and then re-enters the ISR.

Reset

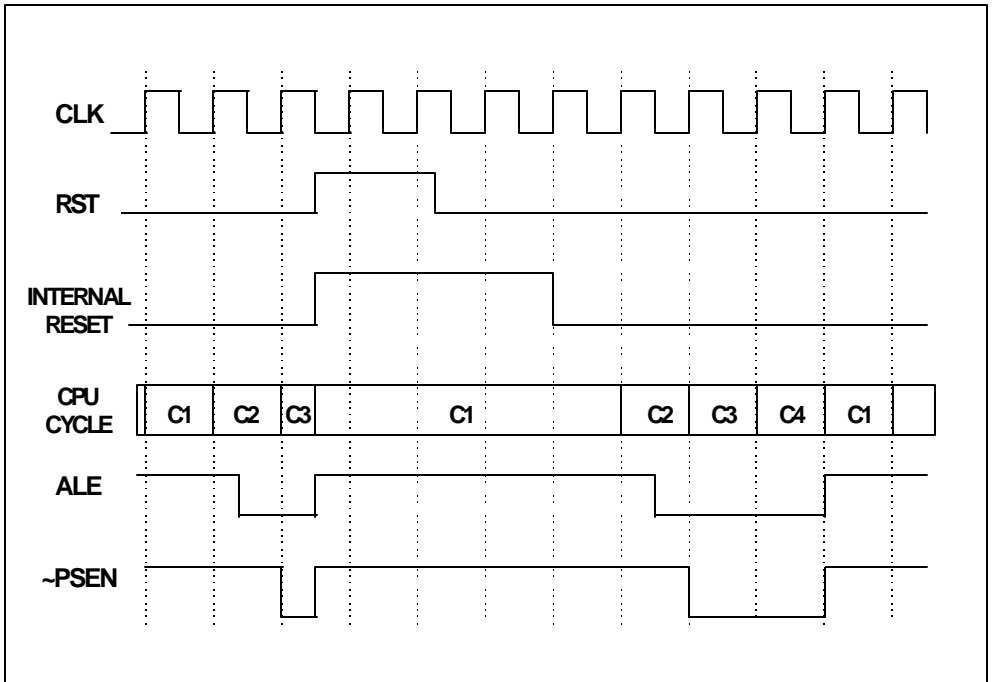
The MAX765x provides one reset input, *RST* (Power-On Reset). The *RST* pin is internally pulled-down with 120K, and in normal operation a capacitor is tied from the pin to the positive supply. The input is active HIGH, and can be driven from external logic. The MAX765x remains in the reset state until the external reset signal is removed, and initializes the SFRs to their reset values (see Table 2-7). The internal RAM is not affected. When the reset signal is removed, the MAX765x exits the reset state and begins program execution at the standard reset vector address 0000H.

Power-On Reset

The *RST* input must be driven HIGH for at least two *clk* cycles to ensure proper initialization. There is no need to externally synchronize the *RST* signal to *clk*. An internal 2-stage synchronizer synchronizes *RST* to *clk*.

Asserting *RST* immediately aborts the current operation, forces all internal logic into the reset state. In normal operation, an electrolytic capacitor of 2.2mfd is tied to the pin. The positive lead of the cap is connected to *Vcc*, and the negative lead is connected to the *RST* pin.

Figure 3-20: Reset Timing



Power Saving Modes

The MAX765x provides two power saving modes: idle mode and stop mode. Table 3-24 summarizes the differences in power saving features.

The bits that control entry into idle and stop modes are in the PCON register at SFR address 87H (see Table 3-25).

Table 3-24: Power Saving Modes Compatibility Summary

Feature	8051	MAX765x
Idle mode	Clock gated internally. Exit idle mode by interrupt or reset.	Clock not gated internally. Exit idle mode by interrupt or reset.
Stop mode	Clock gated internally. Exit stop mode by interrupt or reset.	Clock not gated internally. Exit stop mode by power-on reset only.

Idle Mode

An instruction that sets the IDLE bit (PCON.0) causes the MAX765x to enter idle mode when the instruction completes. In idle mode, CPU processing is suspended and internal registers maintain their current data. However, unlike the 8051, the *clk* is not disabled internally.

Figure 3-18 illustrates the timing relationships of MAX765x signals when entering idle mode.

There are two ways to exit idle mode: activate any enabled interrupt or assert *RST*. Activation of any enabled interrupt causes the hardware to clear the IDLE bit and terminate idle mode. The CPU executes the ISR associated with the received interrupt. The `RETI` instruction at the end of the of ISR returns the CPU to the instruction following the one that put the MAX765x into idle mode.

Figure 3-21 illustrates the timing relationships of MAX765x signals when exiting idle mode due to an interrupt.

Activating *RST* causes the MAX765x to exit idle mode, reset internal modules, and begin program execution at the standard reset vector address 0000H. See the reset timing diagrams for timing information.

Table 3-25: PCON Register – SFR 87H

Bit	Function
PCON.7	SMOD0 – Serial Port 0 baud-rate doubler enable. When SMOD0 = 1, the baud rate for Serial Port 0 is doubled.
PCON.6–4	Reserved.
PCON.3	GF1 – General purpose flag 1. Bit-addressable, general purpose flag for software control.
PCON.2	GF0 – General purpose flag 0. Bit-addressable, general purpose flag for software control.
PCON.1	STOP – Stop mode select. Setting the STOP bit places the MAX765x in stop mode.
PCON.0	IDLE – Idle mode select. Setting the IDLE bit places the MAX765x in idle mode.

Stop Mode

An instruction that sets the STOP bit causes the MAX765x to enter stop mode when that instruction completes (PCON.1, see Table 3-24). In stop mode, CPU processing is suspended and internal registers maintain their current data. The internal clock is disabled, internal analog circuitry is powered down, and the chip enters a low-current power-down state.

In stop mode, the internal cycle counter is reset. Because most internal operations are controlled by the cycle counter, internal flip-flops do not change state in stop mode. Therefore, there is a significant reduction in power consumption.

The only way to exit stop mode is by asserting *RST*. The MAX765x executes its reset sequence and begins program execution at the standard reset vector address 0000H.

Figure 3-23 illustrates the MAX765x stop mode timing.

Figure 3-21: Idle Mode Entry Timing

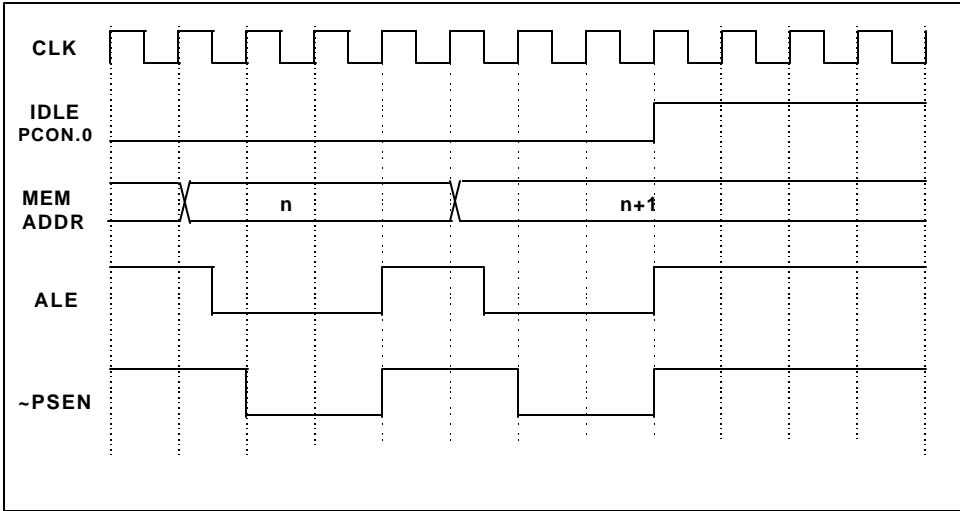


Figure 3-22: Idle Mode Exit Timing

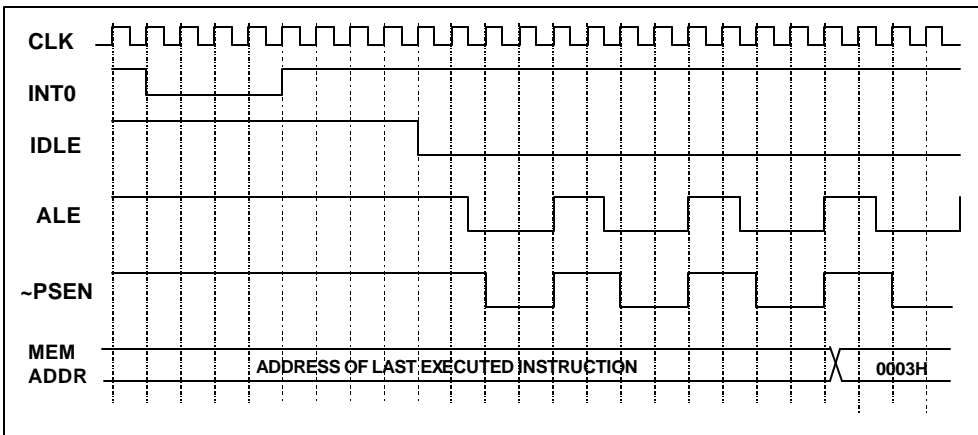
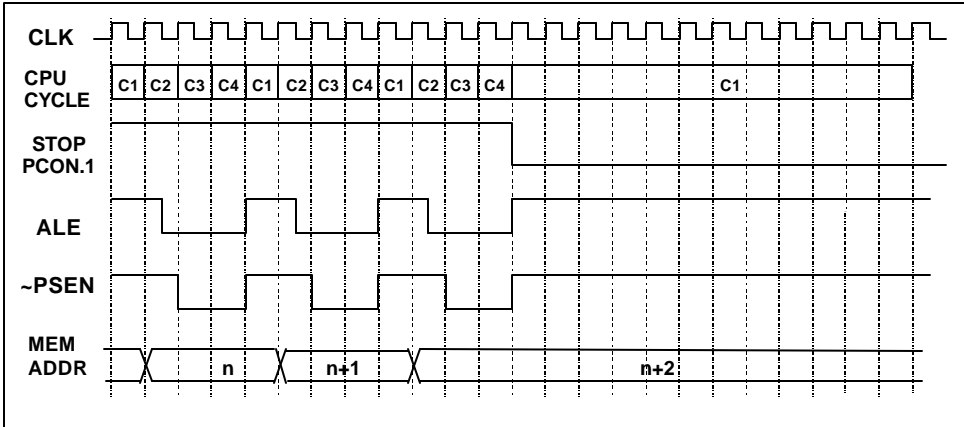


Figure 3-23: Stop Mode Timing



4

Maxim MAX765x Interfacing

This chapter details how to connect the MAX765x to various external peripherals, and how to program the FLASH array.

- Interfacing to External ROM/RAM
- External FLASH programming

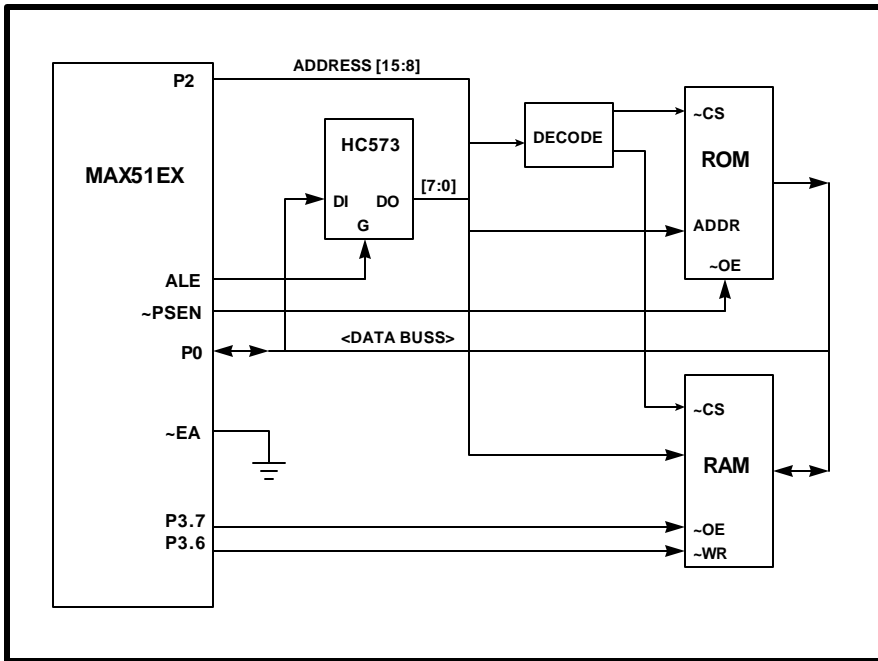
Interfacing to External Memory

Most applications using the MAX765x will only need to use the internal FLASH ROM (program storage) and RAM (data storage). Executing from external memory requires:

- Tie the \sim EA pin to ground.
- Stretch bits (CKCON.2-0) to set the interface timing between the MAX765x and the access time of the external memory.
- Additional hardware is required to generate the address and data signals, and to generate chip selects for the memory.
- Using external memory remaps the I/O ports for the MAX765x, and additional hardware may be required to compensate for the loss of I/O (depending on the design).

The additional hardware required to use external memory is illustrated in Figure 4-1.

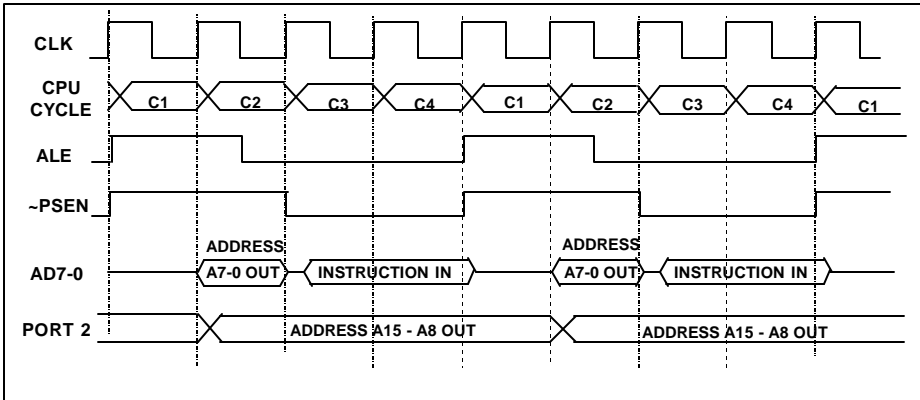
Figure 4 - 1: Interfacing to External Memory



When \sim EA is tied low, Port 2 outputs the upper eight address lines (A15-A8). The data lines and lower eight address lines are multiplexed out the Port 0 pins. An external transparent latch (such as a 74HC573) is used to separate the address lines. When ALE goes low in the middle of cycle C2, the address lines are latched. The MAX765x then uses Port 0 for the bi-directional data port. Unlike the 8051, the MAX765x does not require external pullups on Port 0.

See Figure 4-2 for the MAX765x timing when \sim EA is low.

Figure 4 -2: MAX765x I/O Timing for External Memory



To access RAM, the signals \sim RD and \sim WR are generated by using port bits P3.7 and P3.6, respectively. External ROMS/EPROMs are accessed using \sim PSEN.

Adding Stretched Memory Cycles

The width of the \sim RD and \sim WR pulses can be increased by setting the appropriate bits in the register CKCON. Because data is sampled on the low-to-high transition of \sim RD and \sim WR, stretching the width of these signals gives slower memory devices more time for data access.

The default state of the MAX765x after power-up or application of RESET is CKCON = 01H. This inserts one stretch cycle (also referred to as a 'wait state') for external memory accesses.

WARNING!

Adding Stretch cycles only affects RAM timing. It DOES NOT affect ROM timing: the \sim PSEN signal is NOT stretched. The only way to use slower ROMS is to reduce the CLK frequency. At 12Mhz with Stretch = 0, the ROM access time needed is 200ns (from \sim CS to data valid, and 150ns from \sim OE to data valid). A 150ns part is recommended for operation at 12Mhz, Stretch = 0. In the default configuration, 250ns devices can be used at 12Mhz.

The following table describes RAM timing for Read and Write cycles with different Stretch bit settings.

Opertation	Timing Diagram
Read with Stretch = 0	Figure 4-3
Write with Stretch = 0	Figure 4-4
Read with Stretch = 1	Figure 4-5
Write with Stretch = 1	Figure 4-6
Write with Stretch = 2	Figure 4-7

Figure 4 - 3: RAM Read Timing with Stretch = 0

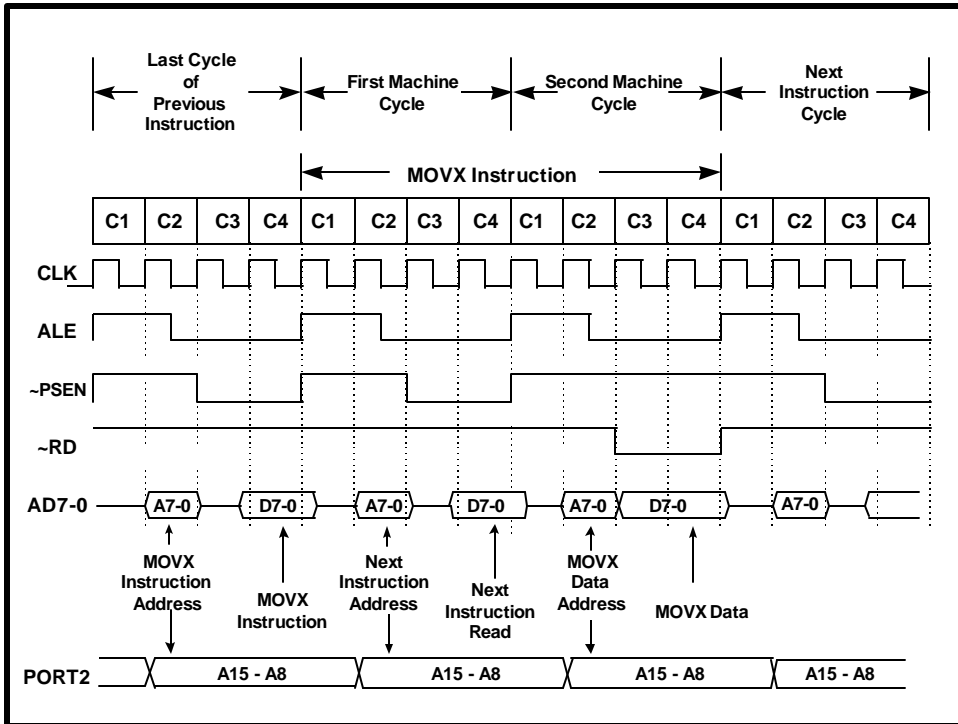


Figure 4 - 4: RAM Write Timing with Stretch = 0

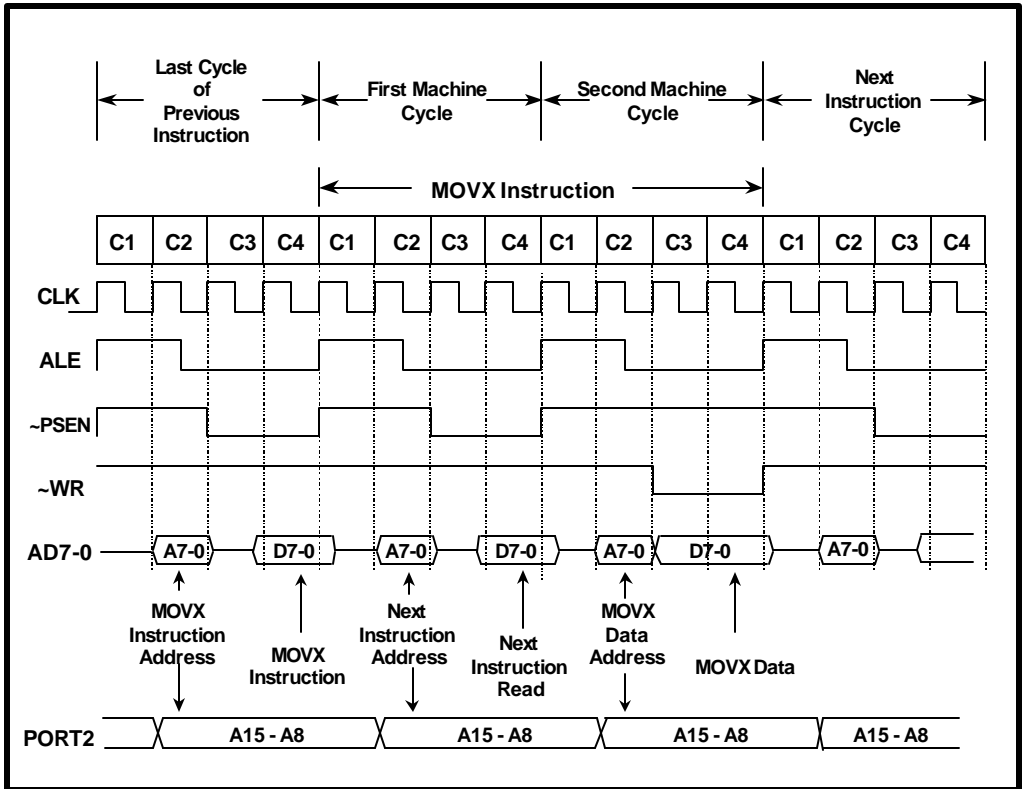


Figure 4 - 5 RAM Write Timing with Stretch = 1

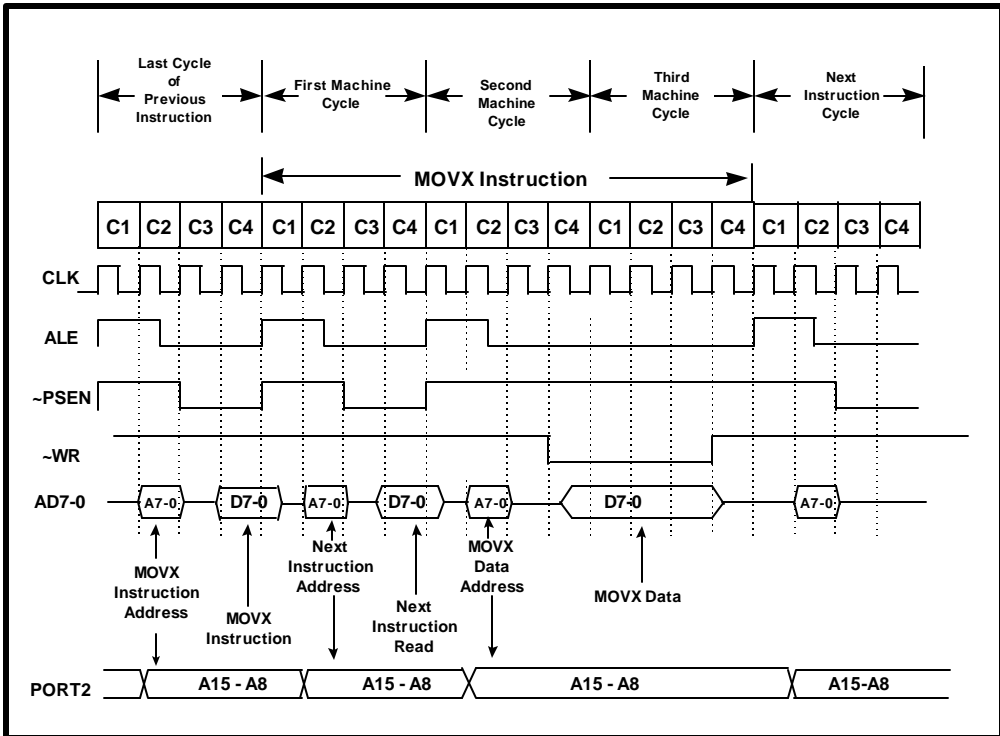
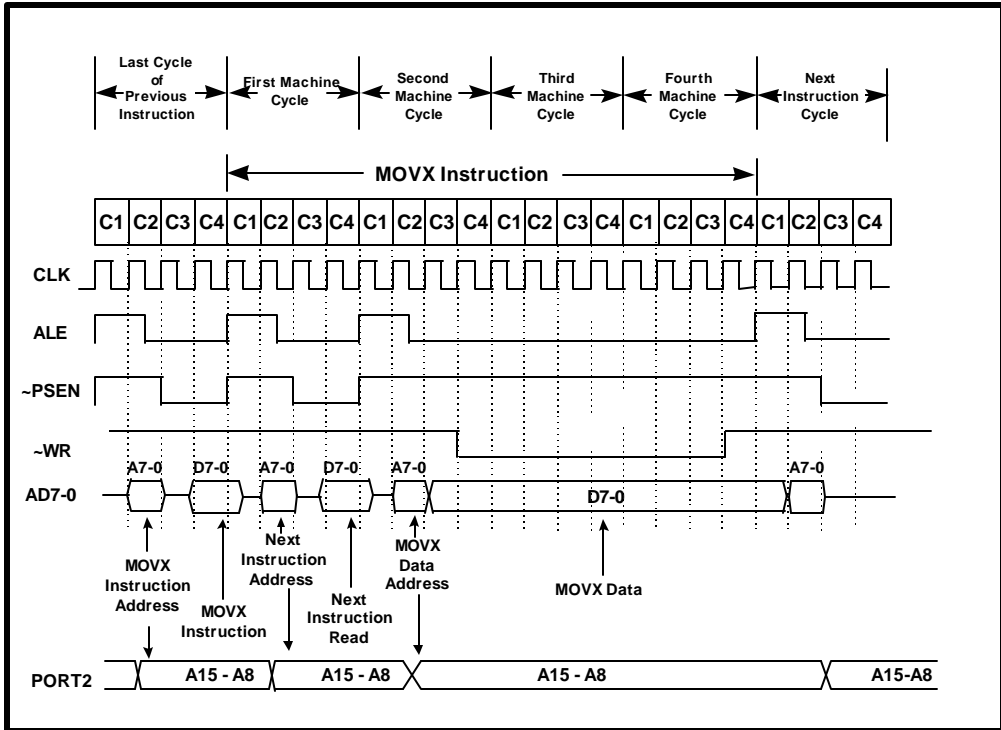


Figure 4 - 6: RAM Write Timing with Stretch = 2



External Programming of the Internal FLASH Memory

The MAX765x is normally shipped with the on-chip FLASH arrays in the erased state (contents = FFH) and ready to be programmed.

External Programming Algorithm

Before programming the MAX765x, address, data, and control signals should be set up according to the FLASH programming mode table shown below. *Failure to follow the recommended programming/verify algorithms can result in loss of FLASH data integrity.* To program the MAX765x perform the following steps:

External data / lockbit programming and power-up sequence:

1. Power-up the chip with RST asserted and allow ALE and ~PSEN to float to the “1” state (they will be internally pulled-up during RST assertion). Ports 0,1,2, & 3 go to weak resistive pullup mode. Wait 10 milliseconds for internal bandgap and oscillator to stabilize.
2. Input the desired memory location on the address lines.
3. Input the appropriate data byte on the data lines.

4. Raise \sim EA/Vpp to 5V.

5. Pulse ALE/ \sim PROG low once to program a byte in either FLASH array as per the programming mode table below. The byte-write cycle is self-timed and takes about 120 microseconds per byte. Repeat steps 2-5 for each additional byte programmed. During a write cycle the P3.4 pin functions as a READY/ \sim BUSY signal to indicate the status of the byte-write cycle. P3.4 will go low upon assertion of the ALE/ \sim PROG pulse. P3.4 will go high again after completion of the write cycle. Return ALE/ \sim PROG to logic 1.

External programming and power-down sequence

1. Remove drive from and allow \sim PSEN and ALE/ \sim PROG to float high, pull \sim EA low
2. Tri-state all other pins and remove power from all power pins.

External Verify

If lock bits LB1 and LB2 have not been programmed, the programmed FLASH array(s) can be read back using the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

External verify (readback) power-up sequence

1. Power-up chip with RST asserted and allow ALE and ~PSEN to float to the “1” state (they will be internally pulled-up during RST assertion). Wait 10 milliseconds for internal bandgap and oscillator to stabilize.

2. Pull ~PSEN LOW, ~EA HIGH, ALE HIGH, and set P2.6 P2.7 P3.6 P3.7 P2.5 as per the Flash Programming Modes table below for reading either LOWER or UPPER Flash memory block.

Note that P2.7 is cycled low/high to perform a FLASH read operation (see table below). Minimum low time for P2.7 is 10 Tck cycles.

External verify power-down sequence

(follow same steps as external programming power-down sequence)

External Chip Erase:

Both FLASH arrays can be simultaneously mass-erased electrically by using the proper combination of control signals as shown in the programming table below. The erase operation must be executed before either memory can be reprogrammed. Lock bits are also erased (cleared).

External chip erase power-up sequence

1. (same as step 1. for External verify sequence)
2. Pull \sim PSEN LOW, \sim EA HIGH, set P2.6, P2.7, P3.6, P3.7 and P2.5 as per Mass Erase mode in the Flash Programming Modes table below.
3. P3.4 will be LOW during mass erase cycle and return HI at the end of mass erase cycle.

External chip erase power-down sequence

(follow same steps as external programming power-down sequence)

Mode:	RST	~PSEN	ALE / /~PROG	~EA/ Vpp	P2.6	P2.7	P3.6	P3.7	P2.5 lower=L upper=H
Write lower FLASH	H	L	↓↑	H	L	H	H	H	L
Read lower FLASH	H	L	H	H	L	↓↑	H	H	L
Write Lock Bit 1	H	L	↓↑	H	H	H	H	H	H
Write Lock Bit 2	H	L	↓↑	H	H	H	L	L	H
Write Lock Bit 3	H	L	↓↑	H	H	L	H	L	H
Mass Erase	H	L	↓↑	H	H	L	L	L	H
Read Sig Bytes	H	L	H	H	L	L	L	L	L
Write upper FLASH	H	L	↓↑	H	L	H	H	H	H
Read upper FLASH	H	L	H	H	L	↓↑	H	H	H

Figure _ Programming the FLASH

Figure _ Verifying (reading) the FLASH

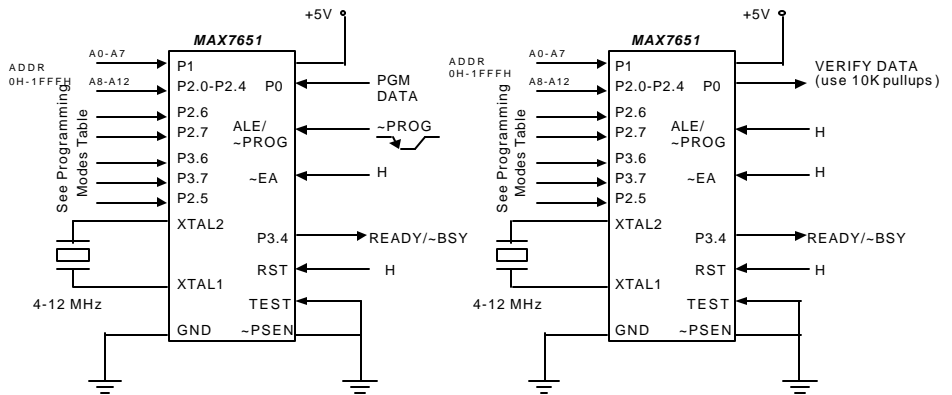
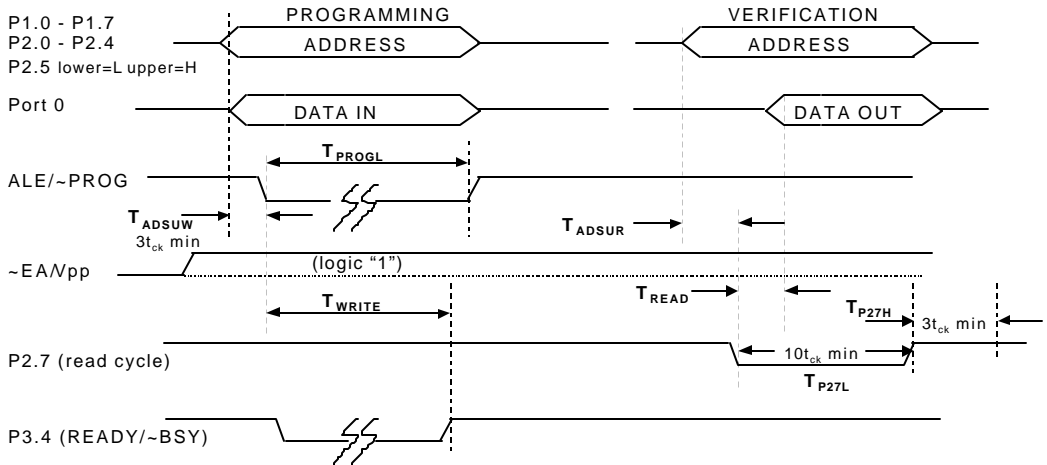


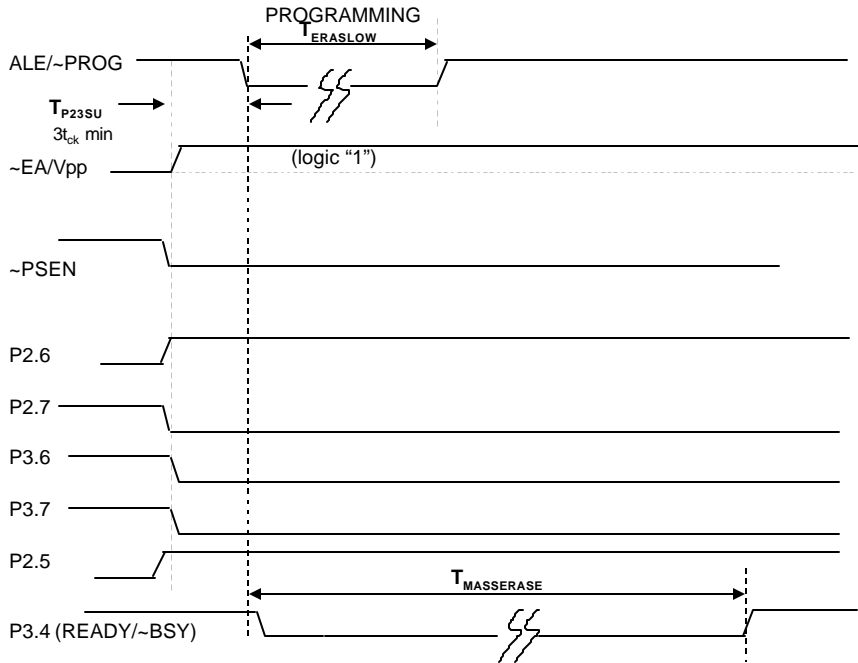
Figure 4 - 7: External I/O for FLASH Programming

FLASH External Data Programming and Verification Waveforms



Parameter	Min	Max	Comments
T_{PROGL}	$10 T_{CK}$		T_{PROGL} must equal T_{WRITE} during lockbit writes
T_{ASUW}	$3 T_{CK}$		
T_{WRITE}	$7 T_{CK} + 54\text{usec}$	$7 T_{CK} + 72\text{usec}$	Test limits. P3.4 low for 63.25 1MHz periods +/- 15%
T_{ADSUR}	$3 T_{CK}$		
T_{READ}		$8 T_{CK} + 50\text{nsec}$	Read access time
T_{P27L}	$10 T_{CK}$		
T_{P27H}	$3 T_{CK}$		
T_{CK}	83nsec	250nsec	

FLASH External Mass Erase Waveforms



Parameter	Min	Max	Comments
T_{P23SU}	$3 T_{CK}$		
$T_{ERASLOW}$	$10 T_{CK}$		
$T_{MASSERASE}$	$1 T_{CK} + 8.29ms$	$1 T_{CK} + 10.97ms$	Test Limits. P3.4 low for 9543 1MHz periods +/- 15%
T_{CK}	83ns	250ns	

Note- Timing assumes inputs are set up before next rising edge of CK