



Microchip

PIC[®]17C42

High Performance 8-Bit CMOS EPROM Microcontroller

FEATURES

Powerful CPU

- Fully static design
- 8 bit wide data path
- 16 bit wide instructions
- All instructions are single word
- Most instructions are single cycle, a few are two cycle
- 250ns cycle time (at 16 MHz). 25 Mhz version is planned.
- 64K x 16 of addressable program memory space
- Direct, indirect (with auto increment and decrement), immediate and relative addressing
- Four modes of operation
 - Microcontroller mode
 - Secure microcontroller mode
 - Extended microcontroller mode (both internal and external program memory access)
 - Microprocessor mode (external only program memory access)

High level of Integration

- 2K x 16 on chip EPROM program memory
- 232 x 8 general purpose registers (SRAM)
- 48 special function registers
- 16 x 16 hardware stack
- 11 external/internal interrupts
- Up to 33 I/O pins
- Three 16-bit timer/counters
- Two 16-bit capture registers
- Two high speed PWM outputs (10 bit, 15.6 KHz)
- Full featured serial port (USART) with baud rate generator

Special microcontroller features

- Watchdog timer with its own on-chip RC oscillator for reliable operation
- Power saving SLEEP mode
- On-chip power-up timer and power on reset saves external circuitry
- On-chip oscillator start-up timer
- Fuse selectable oscillator options: standard crystal oscillator, low frequency crystal oscillator, RC oscillator or external clocking
- Code protection feature to protect on-chip EPROM program memory

Package Options

- 40L cerdip window, 40L PDIP, 44L PLCC and 44L PQFP

PIC17C42 OVERVIEW

PIC17C42 is the first member of a high performance EPROM based 8-bit CMOS microcontroller family. The PIC17C42 integrates a powerful CPU (250 ns instruction cycle) with an array of peripheral resources making it ideal for complex real-time control applications.

Microchip's EPROM technology allows the user to test and develop code on windowed cerdip package version and move into production with the cost effective One Time Programmable (OTP) plastic DIP package version.

The PIC17C42 is fully supported by a host of software and hardware development tools. These include an assembler/linker, a low cost in-circuit emulator, a high performance in-circuit emulator, a programmer and a programmer/development board. A C compiler is planned. All tools are supported by PC AT and compatible platforms.

1.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC17C42 can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC17C42 uses a modified Harvard architecture, in which, program and data are accessed from separate memories. This improves bandwidth over traditional Von-Neuman architecture where program and data are fetched from the same memory. Separating program and data memory further allows instructions to be sized differently than 8-bit wide data word. In PIC17C42, opcodes are 16-bit wide making it possible to have all single word instructions. Full 16 bit wide program memory access bus fetches a 16 bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions (55 in all) execute in a single cycle (250ns @ 16MHz) except for program branches and two special instructions that transfer data between program and data memory.

The PIC17C42 can address 64K x 16 program memory space. It integrates 2K x 16 EPROM program memory on-chip. Program execution can be internal only (microcontroller mode), external only (microprocessor mode) or both (extended microcontroller mode).

The PIC17C42 can directly or indirectly address 256 data memory locations (file registers). All special function registers including the program counter are mapped in the data memory. The PIC17C42 has a fairly orthogonal (symmetrical) instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC17C42 simple yet efficient. In addition, the learning curve is reduced significantly.

FIGURE A: PIC17C42 BLOCK DIAGRAM

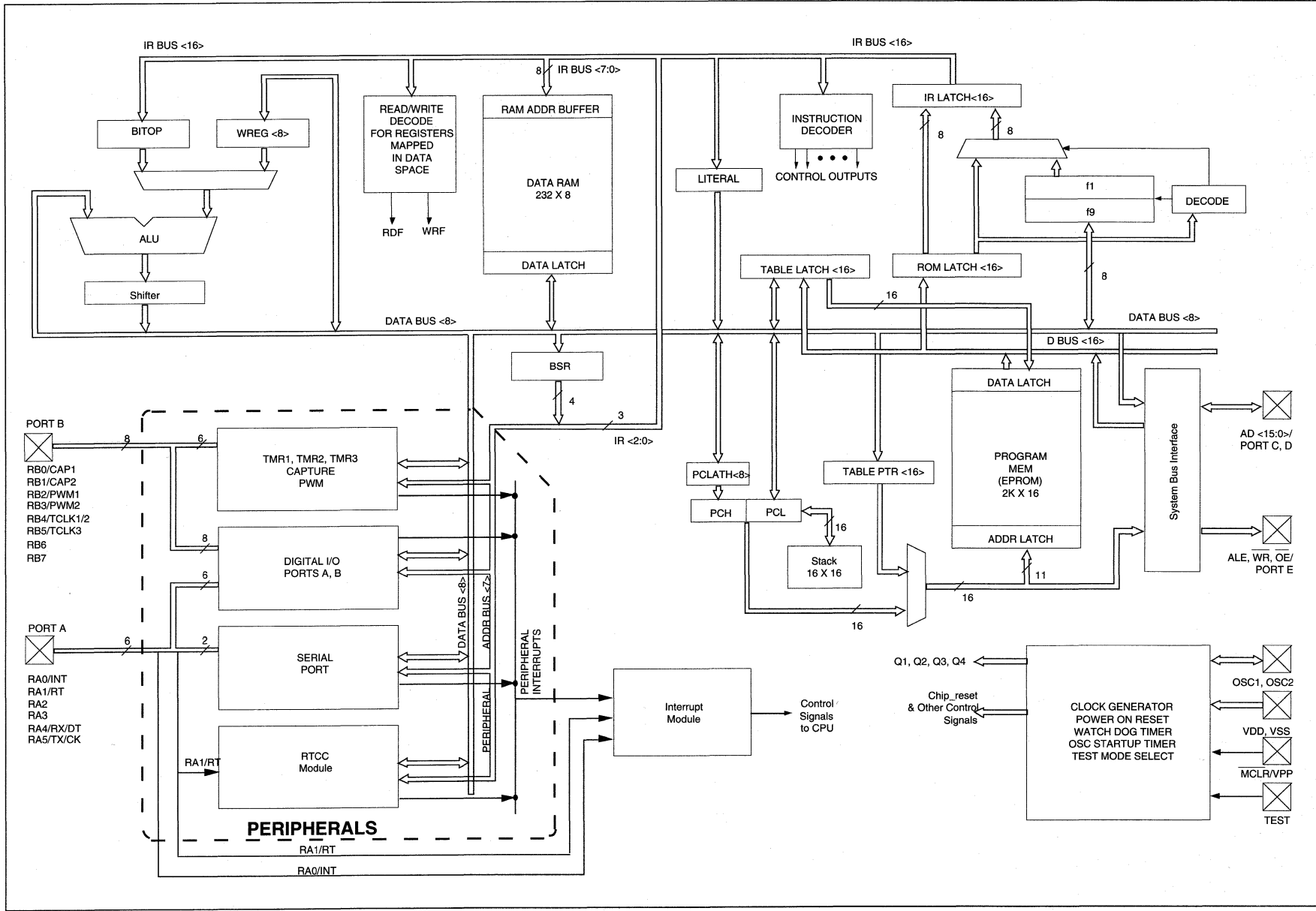


Table of Contents

1.0	Architectural Overview	1	7.1.2	Asynchronous Mode Reception	42
1.1	PIC17C42 Pinout Description	5	7.2	Synchronous Mode	43
1.2	Internal Clocking Scheme	6	7.2.1	Synchronous Mode Transmission	43
1.3	Instruction Flow/Pipelining	7	7.2.2	Synchronous Mode Reception	44
1.4	Memory Organization	7	7.2.3	Synchronous Slave Mode/SLEEP Mode	45
1.5	Different Program Memory Organization	7		Interaction:	45
1.5.1	External Program Memory Interface	8	7.3	Baud Rate Generator	45
1.6	Data Memory Organization	9	7.4	Serial Port Registers	48
1.6.1	Organization of Special Function Registers	9	7.4.1	Summary of Serial Port Registers	48
2.0	Instruction Set	11	7.5	Summary of Serial Port Pins	49
2.1	Special Function Registers as Source/ Destination	13	8.0	Timer/Counters: Overview	49
2.2	Instruction Description	14	8.1	Role of the Timer/Counters	49
3.0	Hardware Description of the CPU	23	8.2	RTCC Module	49
3.1	Indirect Addressing Registers (Files 00h & 08h)	23	8.2.1	RTCC Operation	50
3.2	File Select Registers (FSR0 and FSR1, Files 01h and 09h)	23	8.2.2	Read/Write Consideration for RTCC	50
3.3	Table Pointer (TBLPTRL Files and TBLPTRH, Files ODh and OEh)	23	8.2.3	External Clock Considerations	51
3.4	Table Latch (TBLATH, TBLATL)	23	8.2.4	Summary of RTCC Registers	52
3.5	Program Counter Module	23	8.3	Timer1 & Timer2	52
3.6	Stack	23	8.3.1	Timer1, Timer2 in 8 Bit Mode	52
3.6.1	Stack Available Status Bit (Bit 5, CPUSTA)	24	8.3.2	Timer1 & Timer2 in 16 Bit Mode	53
3.6.2	Using the STKAVL Bit	24	8.3.3	External Clock Input for Timer1, Timer2	53
3.7	Interrupt Logic	24	8.3.4	Summary of Timer1, Timer2 Registers	56
3.7.1	Interrupt Flag and Mask Bits	24	8.4	Timer/Counter 3	56
3.7.2	Peripheral Interrupts	24	8.4.1	External Clock Input for Timer3	57
3.7.3	INT and RT External Interrupts	26	8.4.2	Reading/Writing Timer3	57
3.8	ALU	27	8.4.3	Summary of Timer3 Registers	57
4.0	Special Features of the CPU	28	9.0	Capture Module	59
4.1	Reset	28	9.1	One Capture + Timer/Counter3 + Period	59
4.2	Oscillator	29		Register Mode	59
4.2.1	EC: External Clock Input Mode	29	9.2	Two Capture + Timer/Counter3 Mode	59
4.2.2	RC: RC Oscillator Mode	29	9.3	Summary of Capture Registers	60
4.2.3	XT: Crystal Oscillator Mode	29	10.0	Pulse Width Modulation (PWM) Outputs	60
4.2.4	LF: Low Frequency Oscillator Mode	29	10.1	Summary of PWM Registers	62
4.3	Oscillator Start-up Timer (OST)	30	11.0	Development Support	62
4.4	Power-up Timer (PWRT) and Power on Reset (POR)	30	11.1	PICASM-17: PIC17C42 Cross Assembler	62
4.5	Sleep Mode	32	11.2	PICPAK-17™: PIC17C42 Evaluation/ Development/Programmer Kit	62
4.5.1	Wake-up from SLEEP	32	11.3	PRO MASTER™ Programmer	62
4.5.2	Interrupt/SLEEP Interaction	33	11.4	PICMASTER-17™: High Performance Universal In-Circuit Emulator System	63
4.5.3	Minimizing current consumption in SLEEP Mode	33	11.4.1	Host System Requirements	63
4.6	Watchdog Timer	34	11.4.2	Emulator System Components	63
4.6.1	WDT as a Regular Timer	34	11.5	Ordering Development Tools	65
4.7	Code Protection and Write Protection	34	11.6	Application and Technical Support	65
4.8	Configuration Fuses	34	11.7	Programming Support	65
5.0	Overview of Peripherals	35	11.7.1	Prototype Programming	65
5.1	The Bank Select Register (BSR, Address 0Fh)	35	11.7.2	Production Volume Programming	65
6.0	Digital I/O Ports	36	11.7.3	Factory Programming	65
6.1	Port A	36	11.7.4	Distributor Programming Support	65
6.1.1	Using RA2, RA3 Pins as Output	37	12.0	Electrical Characteristics	66
6.1.2	Summary of Port A Registers	37	12.1	Absolute Maximum Ratings	66
6.2	Port B	37	12.2	DC Characteristics	67
6.2.1	Summary of Port B Registers	39	12.3	AC Characteristics	68
6.3	Port C	39	12.3.1	AC Characteristics: OSC/Reset/System Bus	68
6.3.1	Summary of Port C Registers	39	12.3.2	AC Characteristics: Serial Port	69
6.4	Port D	40	12.3.3	AC Characteristics: I/O Port	69
6.4.1	Summary of Port D Registers	40	12.3.4	AC Characteristics: RTCC and INT	70
6.5	Port E	40	12.3.5	AC Characteristics: Timer1, Timer2, Timer3, Capture and PWM:	70
6.5.1	Summary of Port E Registers	41	12.3.6	AC Test Load and Timing Conditions	71
7.0	Universal Synchronous Asynchronous Receiver Transmitter (USART)	41	12.4	Timing Diagrams	72
7.1	Asynchronous Mode	41	13.0	Package Information	78
7.1.1	Asynchronous Mode Transmission	41	13.1	Package Type: 40-Lead Ceramic Cerdip Dual In-Line with Window (.600 mil)	78
			13.2	Package Type: 40-Lead Plastic Dual In-Line (.600 mil)	79
			13.3	Package Type: 44-Lead Plastic Leaded Chip Carrier (Square)	80

13.4	Package Type: 44-Lead Metric Plastic Quad Fine Pitch (MQFP 10x10MM Body 1.6/.05MM Lead Form)	81
	Sales and Support	84
	Part Numbers	84

Table of Figures

A	PIC17C42 Block Diagram	2
B	PIC17C42 Pin-outs	5
1.2.1	Internal Clocks	6
1.3.1	Instruction Fetch/Execute Pipeline	7
1.5.1	Program Memory Map	8
1.5.2	Memory Map in Different Modes	8
1.5.1.1	External Program Memory Read and Write Timings	9
1.6.1	Data Memory Map	9
1.6.2	Register File Summary (PIC17C42)	10
2.0.1	Instruction Decode Map	13
3.7.1.1	Register INTSTA	25
3.7.1.2	PIR (Peripheral Interrupt Request) Register	25
3.7.1.3	PIE (Peripheral Interrupt Enable) Register	26
3.7.3.1	RTCSTA: RTCC Status/Control Register	26
3.8.1	ALUSTA (ALU Status) Register	27
4.1.1	Simplified Block Diagram of On-Chip Reset Circuit	28
4.2.1	Different Oscillator/Clockin Options	29
4.4.1	Brown Out Protection Circuit	30
4.4.2	Brown Out Protection Circuit	30
4.4.3	External Reset Pulse	31
4.4.4	Using On-chip POR	31
4.4.5	Internal Reset (V _{DD} and MCLR Tied Together)	31
4.4.6	Internal Reset (V _{DD} and MCLR Tied Together): Slow V _{DD} Rise Time	31
4.4.7	OST Start-up Timing Details	32
4.5.1	CPUSTA Register	33
4.8.1	Reading Fuse Location	34
6.0.1	I/O Port Read and Write Timing	36
6.1.1	Port A Block Diagrams	37
6.2.1	Port B Block Diagram	38
6.2.2	Port B Block Diagram	39
6.3.1	Block Diagram of Ports C, D and E	40
7.1.1.1	Asynchronous Transmission	42
7.1.1.2	Asynchronous Transmission (back to back)	42
7.1.2.1	Asynchronous Reception	43
7.2.1.1	Synchronous Transmission	44
7.2.1.2	Synchronous Transmission (through TXEN)	44
7.2.1.3	Synchronous Transmission (Slave)	44
7.2.2.1	Synchronous Reception (Master Mode, SREN)	45
7.2.2.2	Synchronous Master Mode Reception (CREN)	46
7.4.1.1	RCSTA: Receive Status & Control Register	48
7.4.1.2	TXSTA: Transmit Status & Control Register	48
8.2.1.1	RTCC Module Block Diagram	49
8.2.1.2	RTCSTA: RTCC Status/Control Register	50
8.2.2.1	RTCC Timing: Write High or Low Byte	50
8.2.2.2	RTCC Read/Write in Timer Mode	51
8.2.3.1	RTCC Timing With External Clock	52
8.3.1.1	Timer1/Timer2 Block Diagram	53
8.3.1.2	TMR1, TMR2, TMR3 Timing in Timer Mode	54
8.3.1.3	Timer/Capture/PWM Control Register 1 (TCON1)	55
8.3.3.1	TMR1, TMR2 and TMR3 in External Clock Mode	55
8.3.3.2	Timer/Capture/PWM Control Register 2 (TCON2)	56
8.4.1.1	Timer3/Capture Module Block Diagram	58
10.0.1	Simplified PWM Block Diagram	60
10.0.2	PWM Output	61

11.4.1	PICMASTER-17 Development System	64
11.4.2	PICMASTER-17 Development System Block Diagram	64
11.4.3	Sample Screen Layout for PICMASTER-17	65
12.3.6.1	Input Level Conditions	71
12.3.6.2	Output Level Conditions	71
12.3.6.3	Load Conditions	71
12.4.1	Timing Diagram - External Program Memory Read	72
12.4.2	Timing Diagram - External Program Memory Write	73
12.4.3	Timing Diagram - Interrupt Timing	73
12.4.4	Reset Timing	74
12.4.5	TABLRD Timing	74
12.4.6	TABLRD Timing (Consecutive TABLRD Instructions)	75
12.4.7	TABLWT Timing	75
12.4.8	TABLWT Timing (Consecutive TABLWT Instructions)	76
12.4.9	Sleep/Wake-up Through INT (LF, XT Modes)	76
12.4.10	Sleep/Wake-up Through INT (RC Mode)	77
12.4.11	Synchronous Transmission (master/slave)	77
12.4.12	Synchronous Receive (master/slave)	77
12.4.13	I/O Port Input/Output Timing (Port A, Port B)	77
13.1	Package Type: 40-Lead Ceramic Cerdip Dual In-Line with Window (.600 mil)	78
13.2	Package Type: 40-Lead Plastic Dual In-Line (.600 mil)	79
13.3	Package Type: 44-Lead Plastic Leaded Chip Carrier (Square)	80
13.4	Package Type: 44-Lead Metric Plastic Quad Fine Pitch (MQFP 10x10MM Body 1.6/.015MM Lead Form)	81

Table of Tables

1.1	PIC17C42 Pinout Description	5
1.5.1.2	Access Time Requirements for External Memory	9
3.7.1	Table of Interrupts	24
4.2.1	Oscillator Options	29
4.5.1.1	Wake-up and Reset Function Table	33
4.8.1	Configuration Fuses	34
6.1.1	Port A Functions	36
6.2.1	Port B Functions	38
6.5.1	Port E Functions	41
7.3.1	Baud Rates for Synchronous Mode	47
7.3.2	Baud Rates for Asynchronous Mode	47

Trademarks:

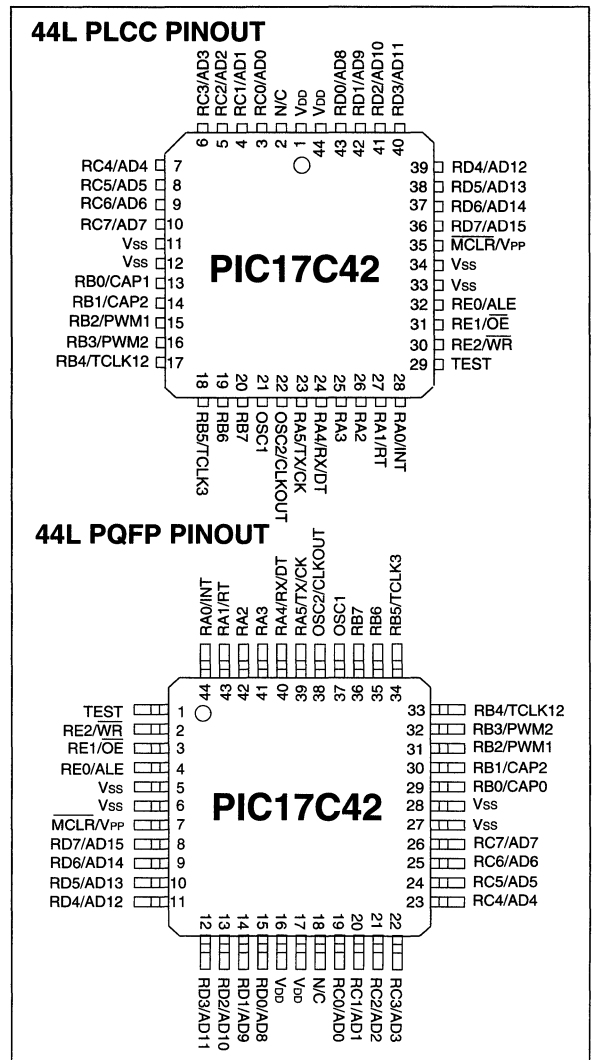
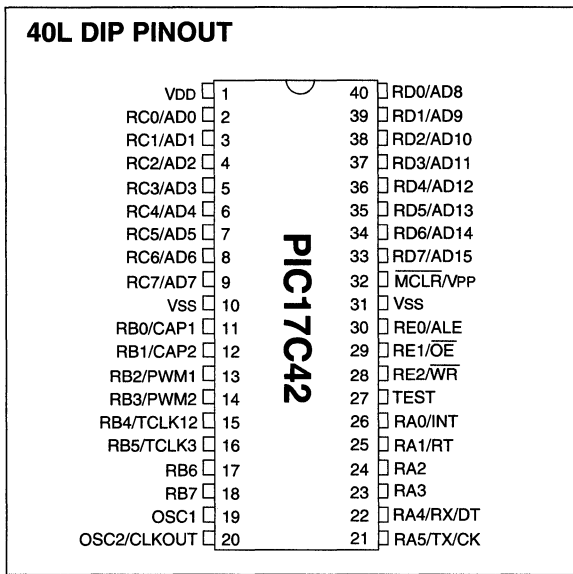
PIC is a registered trademark of Microchip Technology Incorporated.

PICMASTER, PICPRO, PICPAK and PRO MASTER are trademarks of Microchip Technology Incorporated.

IBM PC is a trademark of IBM Corporation.

MS DOS and Microsoft Windows are registered trademarks of Microsoft Corporation.

FIGURE B: PIC17C42 PIN-OUT



1.1 PIC17C42 PINOUT DESCRIPTION

Pin Name	Pin Type	Number of Pins	Pin Function
MCLR/VPP	I/P	1	Master clear (reset) input. This is the active low reset input to the chip. During Programming mode, it is the programming voltage (VPP) input.
OSC1	I	1	Oscillator input in crystal/resonator or RC oscillator mode. External clock input in external clock mode.
OSC2/CLKOUT	O	1	Oscillator output. Connects to crystal or resonator in crystal oscillator mode. In RC oscillator or external clock modes OSC2 pin outputs CLKOUT which has one fourth the frequency of OSC1 and denotes the instruction cycle rate.
RA0/INT	I	1	Input only port pin (bit 0 of Port A) and also external interrupt input. Interrupt can be configured to be on positive or negative edge.
RA1/RT	I	1	Input only port pin (bit 1 of Port A) and also an external interrupt input. Interrupt can be configured to be on rising or falling edge. It is also the external clock input for the RTCC timer/counter.
RA2,RA3	I/O	2	High voltage, high current open drain input/output port pins.
RA4/RX/DT	I/O	1	Input only port pin (bit 4 of Port A). If the serial port is enabled, in full duplex asynchronous serial communication mode this is the receive pin. In half duplex synchronous serial communication mode it is data input (during receive) or data output (during transmit).
RA5/TX/CK	I/O	1	Input only port pin (bit 5 of Port A). If the serial port is enabled, in full duplex asynchronous serial communication mode it is the transmit pin. In half duplex synchronous communication mode, it is shift clock input (slave mode) or clock output (master mode).

(cont.)

Pin Name	Pin Type	Number of Pins	Pin Function
RB0/CAP1	I/O	1	Port pin configurable as input or output in software, with Schmitt trigger input (bit 0 of Port B). It is also the capture1 input pin.
RB1/CAP2	I/O	1	Port pin configurable as input or output in software, with Schmitt trigger input (bit 1 of Port B). It is also the capture2 input pin.
RB4/TCLK12	I/O	1	Port pin configurable as input or output in software, with Schmitt trigger input (bit 4 of Port B). It is also the external clock input to timer1 and timer2.
RB5/TCLK3	I/O	1	Port pin configurable as input or output in software, with Schmitt trigger input (bit 5, of Port B). It is also the external clock input to timer3.
RB6, RB7	I/O	2	Port pins configurable as input or output in software, with Schmitt trigger input (bits 6 and 7 of Port B).
RC7/AD7-RC0/AD0	I/O	8	Eight bit wide Port C with each pin software configurable as input or output. Input is TTL compatible (and not CMOS Schmitt trigger type). This is also the lower half of the 16 bit wide system bus in microprocessor mode or extended microcontroller mode. In multiplexed system bus configuration, these pins are address output as well as data input or output.
RD7/AD15-RD0/AD8	I/O	8	Eight bit wide Port D with each pin software configurable as input or output. Input is TTL compatible (and not CMOS Schmitt trigger type). This is also the upper byte of the 16 bit system bus in microprocessor mode or extended microprocessor mode or extended microcontroller mode. In multiplexed system bus configuration these pins are address output as well as data input or output.
RE0/ALE	I/O	1	Port pin configurable as input or output in software, with TTL compatible input (bit 0 of Port E). In microprocessor mode or extended microcomputer mode, it is the Address Latch Enable (ALE) output. Address should be latched on the falling edge of ALE output.
RE1/ \overline{OE}	I/O	1	Port pin configurable as input or output in software, with TTL compatible input (bit 1 of Port E). In microprocessor or extended microcontroller mode, it is the Output Enable (\overline{OE}) control output (active low).
RE2/ \overline{WR}	I/O	1	Port pin configurable as input or output in software, with TTL compatible input (bit 2 of Port E). In microprocessor or extended microcontroller mode, it is the Write Enable (\overline{WR}) control output (active low).
TEST	I	1	Test mode selection control input. Always tie to Vss for normal operation.
VDD	P	1	Power
VSS	P	2	Ground. Both pins must be connected to system ground.

Legend: I = Input only; O = Output only; I/O = Input/output; P = Power.

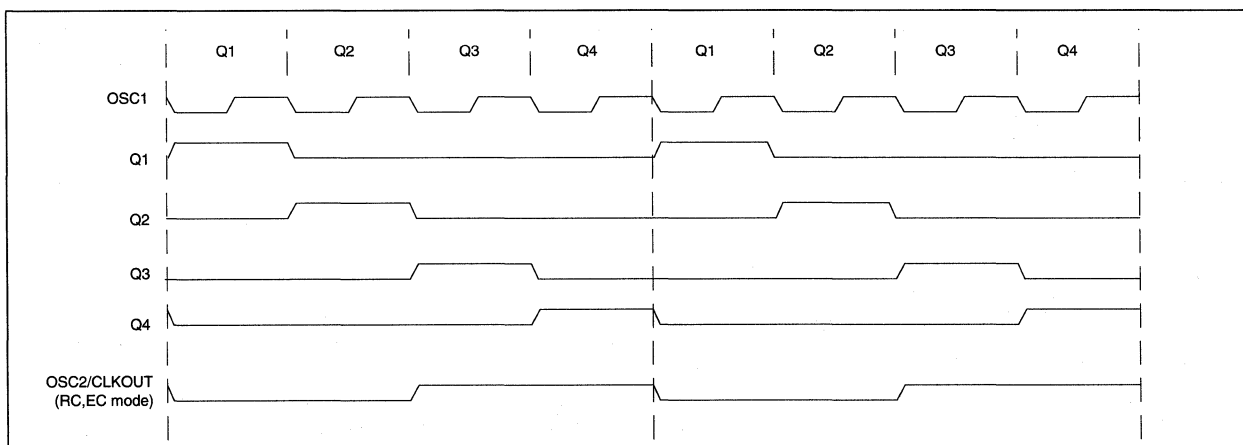
1.2 INTERNAL CLOCKING SCHEME

Internally, the clock input to OSC1 pin is divided by four to generate four phases (Q1, Q2, Q3 and Q4) each with a frequency equal to $f_{osc} / 4$ and duty cycle of 25%. If EC (external clock) or RC oscillator mode is selected,

the OSC2 pin provides a clock output, CLKOUT, which is high during Q3, Q4 and low during Q1, Q2.

As long as internal chip reset is active, the clock generator holds the chip in Q1 state. The CLKOUT pin is driven low (EC, RC mode).

FIGURE 1.2.1: INTERNAL CLOCKS



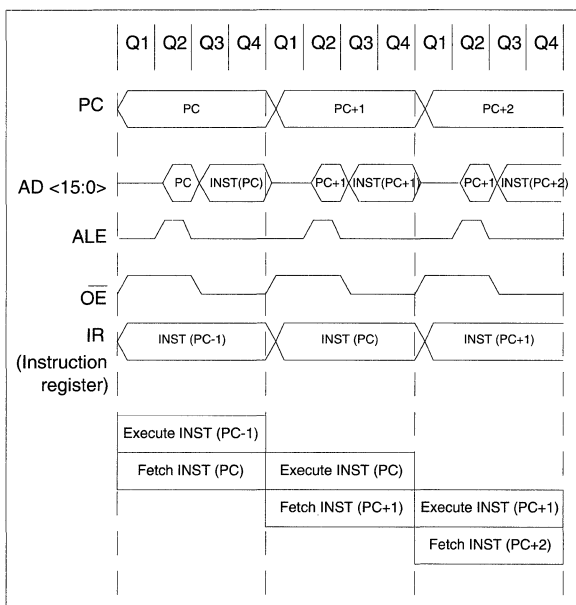
1.3 INSTRUCTION FLOW/PIPELINING

An "Instruction Cycle" in PIC17C42 consists of Q1, Q2, Q3 and Q4. Instruction fetch and execute are pipelined such that fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g. GOTO) then two cycles are required to complete the instruction. Additionally, there are two instructions, TABLRD and TABLWT which take two or more cycles to complete. These are explained in more details 'Instruction Set' description.

A fetch cycle begins with the program counter (PC) incrementing in Q1. In external execution, the address is presented on pins AD15 - AD0 during Q2. The instruction is latched on the falling edge of Q4.

The fetched instruction is latched into the "Instruction Register (IR)" which is decoded in Q1 and executed during Q2, Q3 and Q4. Data memory is read during Q2 (operand read), ALU operations are done in Q3 and result is written back during Q4 (destination write).

FIGURE 1.3.1: INSTRUCTION FETCH/ EXECUTE PIPELINE



1.4 MEMORY ORGANIZATION

The PIC17C42 employs a Harvard architecture, i.e. it has separate program and data memory space. In addition, there is a hardware stack separate from both data and program space. The data space is 256 bytes in size. Most of the data space is implemented as static RAM (address 18h to FFh). Special function registers, implemented as individual hardware registers make up the rest of the data space. Refer to section 1.6 for more details. Data memory "address" and "data" buses are not brought outside the chip. So the data memory can not be expanded externally. The user can, however,

create data segments in external program memory, use TABLWT and TABLRD instructions to move data between external program memory and the register file.

The program memory is 16 bits wide. It is addressed by the 16 bit program counter for instruction fetch. It is also addressed by the table pointer register (TBLPTR, also 16 bit wide) for data move to and from data space. Addressable program memory is 64K x 16. The PIC17C42 incorporates 2K x 16 EPROM program memory on chip.

1.5 DIFFERENT PROGRAM MEMORY ORGANIZATION

The PIC17C42 operates in one of four possible program memory configurations which are:

Microcontroller Mode: In this mode, only internal execution is allowed and therefore, only the on-chip 2K program memory is available. Any access to program memory beyond 2K reads 0000h (which is NOP). In addition to program memory, fuses, test memory, and boot memory (FE00h to FFFFh) are accessible.

Protected Microcontroller Mode: It is the same as microcontroller mode except that code protection is enabled. Refer to section 4.7 for details on code protection.

Extended Microcontroller Mode: In this mode, on chip program memory (0-2K) as well as external memory (2K - 64K) are available. Execution automatically switches to external if program memory address is greater than 07FFh. The fuses, test memory and the boot memory are not accessible in this mode.

Microprocessor Mode: In this mode the on-chip program memory is not used. The entire 64K program memory is mapped externally. The fuses, test memory and the boot memory are not accessible in this mode.

The different modes are selected by fuses FPMM0 and FPMM1. These fuses are mapped in the following program memory locations:

- FPMM0: FE04h
- FPMM1: FE06h

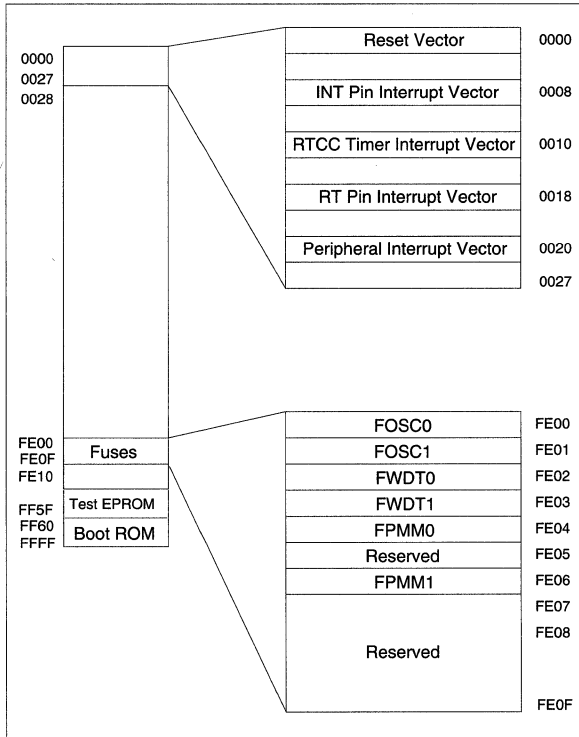
FPMM0	FPMM1	Mode
0	0	Microcontroller Mode (Code Protected)
0	1	Microcontroller Mode (Unprotected)
1	0	Extended Microcontroller Mode
1	1	Microprocessor Mode

Note: * 1 = fuse unprogrammed or erased,
0 = fuse programmed.

Refer to section 4.7 for information on code protection.

Test Memory, Boot Memory and Fuse Locations: Test memory space is used by the factory for testing purposes. The 'boot ROM' area holds programs used for programming and verification. The user need not be concerned about either of these. The fuse locations map configuration fuses used to select from various operating modes. The fuses are explained in detail, in section 4.8.

FIGURE 1.5.1: PROGRAM MEMORY MAP



1.5.1 External Program Memory interface

If external execution is selected, ports C, D and E are configured as a system bus for external program memory access. Ports D and C, together, constitute a 16 bit wide multiplexed address and data bus. The three bit E port outputs control signals \overline{ALE} (Address Latch Enable), \overline{OE} (Output Enable) and \overline{WR} (Write Enable). An external memory access cycle is comprised of four oscillator cycles (from Q1 rising edge to Q1 rising edge). During Q2, a 16 bit address is presented on ports C and D (RD7 = MSB, RC0 = LSB) and \overline{ALE} is asserted. The address output should be latched by the falling edge of \overline{ALE} . In an instruction fetch or data read cycle, the \overline{OE} is asserted during Q3 and Q4. The data is latched on the rising edge of \overline{OE} . One oscillator cycle separation between $\overline{OE} \uparrow$ and address output guarantees adequate time for external memories to shut off their output drivers before address is driven on to the bus.

In a data write cycle (only during TABLWT instruction), following address output during Q2, data is driven onto the bus during Q3 and Q4. \overline{WR} is asserted during Q4 and the data output is valid both on its falling and rising edge.

Figure 1.5.1.1 depicts read and write cycles and table 1.5.1.1 shows access time required of the external memory components. For complete timing information on the system bus, refer to AC characteristics section.

FIGURE 1.5.2: MEMORY MAP IN DIFFERENT MODES

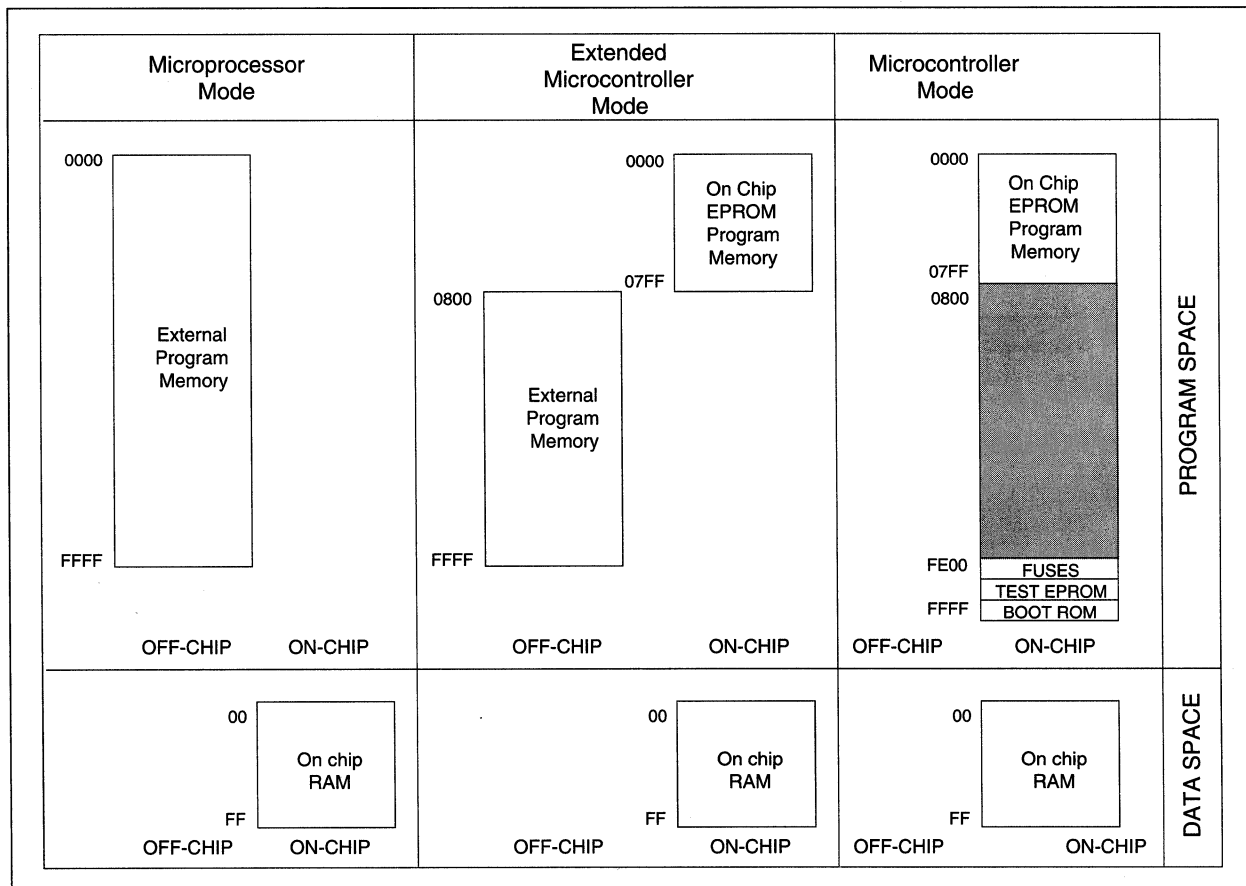


FIGURE 1.5.1.1: EXTERNAL PROGRAM MEMORY READ AND WRITE TIMINGS

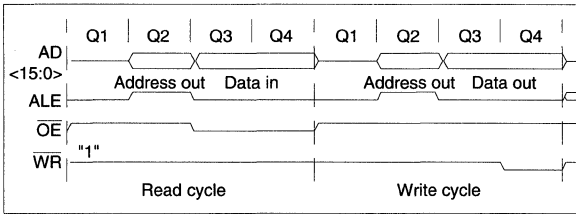


TABLE 1.5.1.2: ACCESS TIME REQUIREMENTS FOR EXTERNAL MEMORY

Osc frequency	Instruction cycle time (Tcy)	tacc	toe
8 MHz	500 ns	345 ns	115 ns
16 MHz	250 ns	157.5 ns	52.5 ns
20 MHz	200 ns	120 ns	40 ns
25 MHz	160 ns	90 ns	30 ns

Note: Estimated access time requirements. Exact number will be available after full characterization.

1.6 DATA MEMORY ORGANIZATION

Data memory in the PIC17C42 is organized as 256 x 8. It is accessed via an internal 8 bit data bus and an 8 bit data-memory-address-bus (derived from the instruction

register). Data memory can be addressed via direct addressing mode or through indirect addressing mode using file select registers FSR0 or FSR1 as pointer registers.

All special function registers (e.g. W, RTCC, Program Counter, Ports) are mapped in the data memory. The rest of the data memory is implemented as static RAM. A few special function registers such as Table Latches (TBLATH, TBLATL) are not mapped in data memory or any other memory space. Also not addressable are the watchdog timer and the stack pointer.

1.6.1 Organization of Special Function Registers

Figure 1.6.1 shows the data memory space in detail:

- a. Address 00h:0Fh are mostly special function registers related to the CPU.
- b. Address 10h:17h are ‘peripheral registers’ such as timer register or port data latch. Since there are many more peripheral registers than can be mapped into 8 address locations, a banking scheme is used. A bank select register, BSR (address 0Fh) is used to select one of many banks. Only the lower 4 bits of BSR are implemented in the PIC17C42, making it possible to address up to 16 banks.
- c. Address locations 18h:1Fh are general purpose file registers implemented as part of the static RAM. However, these locations have the added privilege of being source or destination of a MOVFP or MOVFP instruction respectively.
- d. Locations 20h:FFh are general purpose file registers implemented as static RAM.

FIGURE 1.6.1: DATA MEMORY MAP

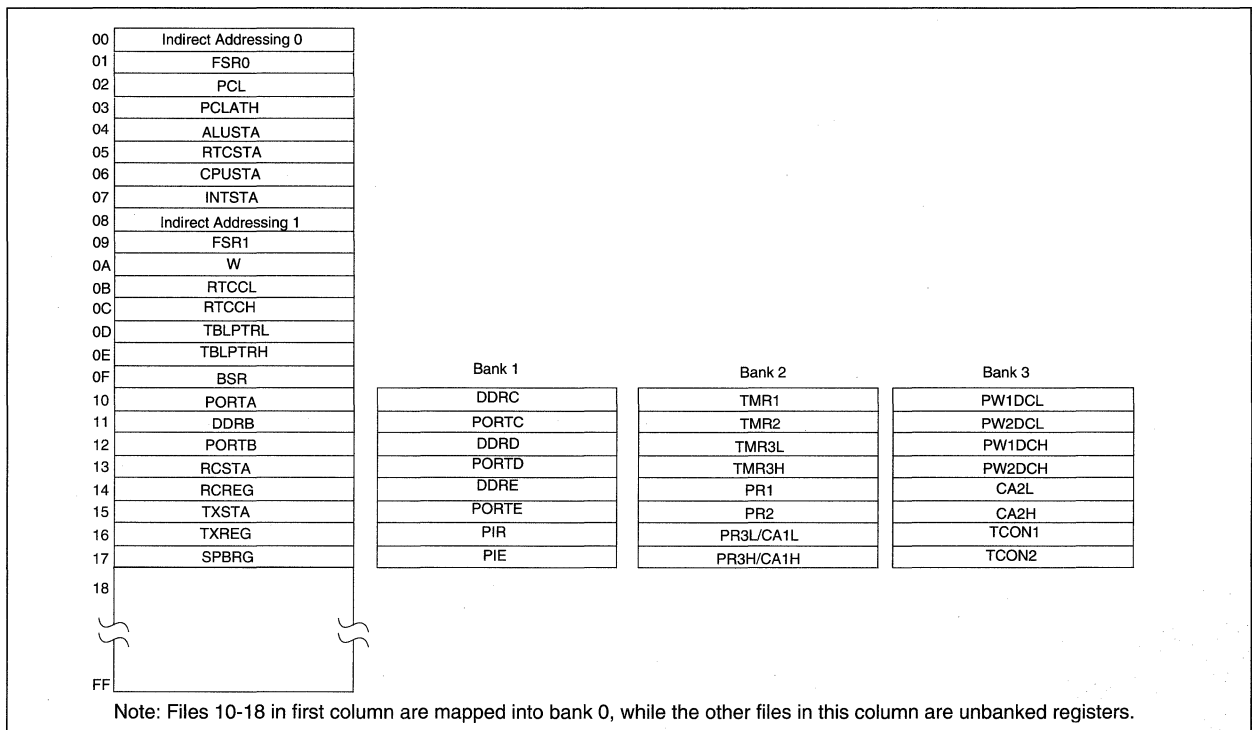


FIGURE 1.6.2: REGISTER FILE SUMMARY (PIC17C42)

Filename	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on power on reset	Value on all other reset (note3)		
UNBANKED:												
00	INDF0	Uses contents of F1 to address data memory (not a physical register)							00000000	00000000	Note 1 Note 2	
01	FSR0	Indirect data memory address pointer 0							XXXXXXXX	UUUUUUUU		
02	PCL	Low order 8 bits of PC							00000000	00000000		
03	PCLATH	Holding register for upper 8 bits of PC (Note 1)							XXXXXXXX	UUUUUUUU		
04	ALUSTA	FS3	FS2	FS1	FS0	OV	Z	DC	C	1111XXXX		1111UUUU
05	RTCSTA	INTEG	RTEDG	T/C	RTPS3	RTPS2	RTPS1	RTPS0	-	00000000		00000000
06	CPUSTA	-	-	STKAV	GLINTD	TO	PD	-	-	00111100		0011??00
07	INTSTA	PEIR	RTXIR	RTCIR	INTIR	PEIE	RTXIE	RTCIE	INTIE	00000000		00000000
08	INDF1	Uses contents of F9 to address data memory (not a physical register)							00000000	00000000		
09	FSR1	Indirect data memory address pointer 1							XXXXXXXX	UUUUUUUU		
0A	W	W register							XXXXXXXX	UUUUUUUU		
0B	RTCCL	Real time clock counter LS byte							XXXXXXXX	UUUUUUUU		
0C	RTCCH	Real time clock counter MS byte							XXXXXXXX	UUUUUUUU		
0D	TBLPTRL	Low byte of program memory table pointer							XXXXXXXX	UUUUUUUU		
0E	TBLPTRH	High byte of program memory table pointer							XXXXXXXX	UUUUUUUU		
0F	BSR	Bank select register							00000000	00000000		
BANK0:												
10	PORTA	PUEB	-	RA5	RA4	RA3	RA2	RA1/RT	RA0/INT	00XXXXXX	00UUUUUU	
11	DDRB	Data Direction Register for Port B							11111111	11111111		
12	PORTB	Port B data latch							XXXXXXXX	UUUUUUUU		
13	RCSTA	SPEN	RC8/9	SREN	CREN	-	FERR	OERR	RCD8	0000000X	0000000U	
14	RCREG	Serial Port Receive Register							XXXXXXXX	UUUUUUUU		
15	TXSTA	CSRC	TX8/9	TXEN	SYNC	-	-	TRMT	TXD8	0000001X	0000001U	
16	TXREG	Serial Port Transmit Register							XXXXXXXX	UUUUUUUU		
17	SPBRG	Baud Rate Generator							XXXXXXXX	UUUUUUUU		
BANK1:												
10	DDRC	Data Direction Register for Port C							11111111	11111111		
11	PORTC	Port C data latch							XXXXXXXX	UUUUUUUU		
12	DDRD	Data Direction Register for Port D							11111111	11111111		
13	PORTD	Port D data latch							XXXXXXXX	UUUUUUUU		
14	DDRE	Data Direction Register for Port E							00001111	0000000U		
15	PORTE	Port E data latch							000000XX	0000000U		
16	PIR	IRB	TM3IR	TM2IR	TM1IR	CA2IR	CA1IR	TBMT	RBFL	00000010	00000010	
17	PIE	IEB	TM3IE	TM2IE	TM1IE	CA2IE	CA1IE	TXIE	RCIE	00000000	00000000	
BANK2:												
10	TMR1	Timer1							XXXXXXXX	UUUUUUUU		
11	TMR2	Timer2							XXXXXXXX	UUUUUUUU		
12	TMR3L	Timer3 Low byte							XXXXXXXX	UUUUUUUU		
13	TMR3H	Timer3 High byte							XXXXXXXX	UUUUUUUU		
14	PR1	Timer1 Period Register							XXXXXXXX	UUUUUUUU		
15	PR2	Timer2 Period Register							XXXXXXXX	UUUUUUUU		
16	PR3L/CA1L	Timer3 Period Register, low byte/capture1 register, low byte							XXXXXXXX	UUUUUUUU		
17	PR3H/CA1H	Timer3 Period Register, High byte/capture1 register, high byte							XXXXXXXX	UUUUUUUU		
BANK3:												
10	PW1DCL	DC1	DC0	-	-	-	-	-	-	XX000000	UU000000	
11	PW2DCL	DC1	DC0	TM2PW2	-	-	-	-	-	XX000000	UU000000	
12	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	XXXXXXXX	UUUUUUUU	
13	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	XXXXXXXX	UUUUUUUU	
14	CA2L	Capture2 low byte							XXXXXXXX	UUUUUUUU		
15	CA2H	Capture2 high byte							XXXXXXXX	UUUUUUUU		
16	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	16/8	TMR3C	TMR2C	TMR1C	00000000	00000000	
17	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/PR3	TMR3ON	TMR2ON	TMR1ON	00000000	00000000	

x = unknown
u = unchanged

Notes:

- 1: The upper byte of the program counter is not directly accessible. f03 is a holding register for PC<15:8> whose contents are updated from or transferred to the upper byte of the program counter.
- 2: The "TO" and "PD" status bits in f06h are not affected by a "MCLR" reset. TO bit will be reset in the event of a WDT time-out reset.
- 3: Other (non power-up) resets include external reset through MCLR pin and watchdog timer timeout reset.

2.0 INSTRUCTION SET

The PIC17C42 instruction set consists of 55 instructions, each single word and 16 bit wide. Most instructions operate on a file register *f* and the working register *W* (accumulator). Depending on the instruction, the result may be directed to the file register, or the working register (*W*) or to both.

All instructions are executed in a single instruction cycle unless otherwise noted.

Any unused op-code is executed as a NOP.

The instruction set is highly orthogonal and is grouped into

- Data Move Operations
- Arithmetic and Logical Operations
- Bit Manipulation Operations
- Program Control Operations
- Special Control Operations

Data Move Instructions

Instruction Code Binary	Hex	mnemonic	Description	Function	Status bits Affected	Notes
011p pppp ffff ffff	6pff	MOVFP	f,p Move f to p	f → p	None	4
1011 1000 kkkk kkkk	B8kk	MOVLB	k Move literal to BSR	k → BSR	None	
010p pppp ffff ffff	4pff	MOVFP	p,f Move p to f	p → f	Z	4
0000 0001 ffff ffff	01ff	MOVWF	f Move W to f	W → f	None	
1010 10ti ffff ffff	A8ff	TABLWD	t,i,f Read data from table latch into file f, then update table latch with 16-bit contents of memory location addressed by the table pointer.	TBLATH → f if t = 1, TBLATL → f if t = 0; Prog Mem (TBLPTR) → TBLAT; TBLPTR + 1 → TBLPTR if i = 1;	None	8,10
1010 11ti ffff ffff	ACff	TABLWT	t,i,f Write data from file f to table latch and then Write 16-bit table latch to program memory location addressed by table pointer. It also initiates programming if on-chip EPROM program memory is addressed.	f → TBLATH if t = 1, f → TBLATL if t = 0; TBLAT → Prog Mem (TBLPTR); TBLPTR + 1 → TBLPTR if i = 1	None	6
1010 00tx ffff ffff	A0ff	TLRD	t,f Read data from table latch into file f (table latch unchanged).	TBLATH → f if t = 1, TBLATL → f if t = 0	None	
1010 01tx ffff ffff	A4ff	TLWT	t,f Write data from file f into table latch.	f → TBLATH if t = 1, f → TBLATL if t = 0	None	

Arithmetic and Logical Instructions

Instruction Code Binary	Hex	mnemonic	Description	Function	Status bits Affected	Notes
1011 0001 kkkk kkkk	B1kk	ADDLW	k Add literal to W	(W+k) → W	OV C DC Z	
0000 111d ffff ffff	0Efd	ADDWF	f,d ADD W to f	(W+f) → d	OV C DC Z	
0001 000d ffff ffff	10fd	ADDWFC	f,d ADD W and Carry to f	(W+f+C) → d	OV C DC Z	
1011 0101 kkkk kkkk	B5kk	ANDLW	k AND literal and W	(W.AND.k) → W	Z	
0000 101d ffff ffff	0Afd	ANDWF	f,d AND W with f	(W.AND.f) → d	Z	
0010 100d ffff ffff	28fd	CLRF	f,d Clear f and Clear d	“00h” → f, “00h” → d	None	3
0001 001d ffff ffff	12fd	COMF	f,d Complement f	f → d	Z	
0010 111d ffff ffff	2Efd	DAW	f,d Dec. adjust W, store in f,d	W adjusted → f and d	C	3
0000 011d ffff ffff	06fd	DECF	f,d Decrement f	(f-1) → d	OV C DC Z	
0001 010d ffff ffff	14fd	INCF	f,d Increment f	(f+1) → d	OV C DC Z	
1011 0011 kkkk kkkk	B3kk	IORLW	k Inclusive OR literal with W	(W.OR.k) → W	Z	
0000 100d ffff ffff	08fd	IORWF	f,d Inclusive OR W with f	(W.OR.f) → d	Z	
1011 0000 kkkk kkkk	B0kk	MOVLW	k Move literal to W	k → W	None	
0010 110d ffff ffff	2Cfd	NEGW	f,d Negate W, store in f and d	(W+1) → f, (W+1) → d	OV C DC Z	1,3
0001 101d ffff ffff	1Afd	RLCF	f,d Rotate left through Carry	f<n>→d<n+1>, f<7>→C, C→d<0>	C	
0010 001d ffff ffff	22fd	RLNCF	f,d Rotate left (no Carry)	f<n>→ d<n+1>, f<7>→ d<0>	None	
0001 100d ffff ffff	18fd	RRCF	f,d Rotate right through Carry	f<n>→d<n-1>, f<0>→C, C→d<7>	C	
0010 000d ffff ffff	20fd	RRNCF	f,d Rotate right (no Carry)	f<n>→ d<n-1>, f<0>→ d<7>	None	
0010 101d ffff ffff	2Afd	SETF	f,d Set f and Set d	“FFh” → f, “FFh” → d	None	3
1011 0010 kkkk kkkk	B2kk	SUBLW	k Subtract W from literal	(k-W) → W	OV C DC Z	
0000 010d ffff ffff	04fd	SUBWF	f,d Subtract W from f	(f-W) → d	OV C DC Z	1
0000 001d ffff ffff	02fd	SUBWFB	f,d Subtract W from f with borrow	(f-W-c)→ d	OV C DC Z	1
0001 110d ffff ffff	1Cfd	SWAPF	f,d Swap f	f<0:3>→ d<4:7>, f<4:7>→ d <0:3>	None	
1011 0100 kkkk kkkk	B4kk	XORLW	k Exclusive OR literal with W	(W.XOR.k) → W	Z	
0000 110d ffff ffff	0Cfd	XORWF	f,d Exclusive OR W with f	(W.XOR.f) → d	Z	

Program Control Instructions

Instruction Code Binary	Hex	mnemonic	Description	Function	Status bits Affected	Notes
111k kkkk kkkk kkkk	Ekkk	CALL	k Subroutine call (within 8K page boundary)	PC+1 → TOS, k → PC<12:0>; k<12:8> → f3<4:0>; PC<15:13> → f3<7:5>	None	8
0011 0001 ffff ffff	31ff	CPFSEQ	f Compare f/W skip if f=W	f - W, skip if f = W	None	7
0011 0010 ffff ffff	32ff	CPFSGT	f Compare f/W skip if f>W	f - W, skip if f > W	None	2,7
0011 0000 ffff ffff	30ff	CPFSLT	f Compare f/W skip if f<W	f - W, skip if f < W	None	2,7
0001 011d ffff ffff	16ff	DECFSZ	f,d Decrement f, skip if 0	(f-1) → d, skip if result =0	None	7
0010 011d ffff ffff	26ff	DCFNSZ	f,d Decrement f skip if not 0	(f-1) → d, skip if not 0	None	7
110k kkkk kkkk kkkk	Ckkk	GOTO	k Unconditional branch (within 8K page boundary)	k → PC<12:0>; k<12:8> → f3<4:0>; PC<15:13> → f3 <7:5>	None	8
0001 111d ffff ffff	1Efd	INCFSZ	f,d Increment f skip if 0	(f+1) → d, skip if result 0	None	7
0010 010d ffff ffff	24fd	INFSNZ	f,d Increment f skip if not 0	(f+1) → d, skip if not 0	None	7
1011 0111 kkkk kkkk	B7kk	LCALL	k Long Call (anywhere in 64K range)	(PC+1) → TOS; (f3) → PCH; k → PCL	None	5,8
0000 0000 0000 0101	0005	RETIE	Return from interrupt and enable interrupt	TOS → PC (f3 unchanged) "0" → GLINTD	GLINTD	8
1011 0110 kkkk kkkk	B6kk	RETLW	k Return literal to W	k → W, TOS → PC, f3 unchanged	None	8
0000 0000 0000 0010	0002	RETURN	Return from subroutine	TOS → PC (f3 unchanged)	None	8
0011 0011 ffff ffff	33ff	TSTFSZ	f Test f skip if 0	skip if f = 0	None	7

Bit Handling Instructions

Instruction Code Binary	Hex	mnemonic	Description	Function	Status bits Affected	Notes
1000 1bbb ffff ffff	8bff	BCF	f,b Bit clear f	0 → f(b)	None	4
1000 0bbb ffff ffff	8bff	BSF	f,b Bit set f	1 → f(b)	None	4
1001 1bbb ffff ffff	9bff	BTFSC	f,b Bit test, skip if clear	skip if f(b) = 0	None	4,7
1001 0bbb ffff ffff	9bff	BTFSS	f,b Bit test, skip if set	skip if f(b) = 1	None	4,7
0011 1bbb ffff ffff	3bff	BTG	f,b Bit Toggle f	f̄(b) → f(b)	None	4

Special control instructions

Instruction Code Binary	Hex	mnemonic	Description	Function	Status bits Affected	Notes
0000 0000 0000 0100	0004	CLRWDT	Clear watch dog timer	0 → WDT, 0 → WDT prescaler, 1 → PD, 1 → TO	PD, TO	
0000 0000 0000 0000	0000	NOP	No operation	None	None	
0000 0000 0000 0011	0003	SLEEP	Enter "sleep" mode	Stop oscillator, "power down" 0 → WDT, 0 → WDT prescaler, 1 → TO, 1 → PD	PD, TO	

Legend:

f	register file address (00h to FFh)
p	peripheral register file address (00h to 1Fh)
b	bit address with in 8 bit file register
i	table pointer control i = 0: do not change i = 1: increment after instruction execution
t	table byte select t = 0: perform operation on lower byte t = 1: perform operation on upper byte
k	literal field (constant data)
x	don't care
d	destination select; d=0 store result in W (f0A) d=1 store result in file register 'f'
C,DC,Z,OV	ALU status bits Carry, Digit Carry, Zero, Overflow
TO, PD	CPU status bits Time-out and Power-down
GLINTD	GLobal Interrupt Disable bit (bit 4, CPUSTA)
W	W-register
PC	Program counter
TBLPTR	Table Pointer (16 bit)
TBLAT	Table Latch (16 bit) consists of high byte (TBLATH) and low byte (TBLATL)
TBLATL	Table latch low byte
TBLATH	Table latch high byte
WDT	Watchdog timer
BSR	Bank Select Register
TOS	Top of Stack


Notes:

- 2's Complement method.
- Unsigned arithmetic
- If d=1, only the file is affected; If d=0, both W and the file are affected; If only W is required to be affected, then f=0Ah (File 0Ah) must be defined.
- The HEX representation is not accurate. The value of the bit to be modified has to be incorporated into the third digit.
- During an LCALL, the contents of file 03h are loaded into the MSB of the PC and kkkk kkkk is loaded into file 02h the LSB of the PC.
- Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event.
When writing to external program memory, it is a two cycle instruction.
- Two cycle instructions when condition is true, else single cycle instruction.
- Two cycle instruction except for TABLRD to f02h (Program Counter low byte) in which case it takes 3 cycles.
- A 'skip' means that instruction fetched during execution of current instruction is not executed. Instead a 'NOP' is executed.
- Any instruction that writes to PCL (f02) is a two cycle instruction, except for TABLRD to f02 is a 3 cycle instruction.

FIGURE 2.0.1: INSTRUCTION DECODE MAP

OPCODE <11:8>		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
OPCODE <15:12>	0	***	MOVWF	SUBWFB		SUBWF		DECF		IORWF		ANDWF		XORWF		ADDWF		
	1	ADDWFC		COMF		INCF		DECFSZ		RRCF		RLCF		SWAPF		INCFSZ		
	2	RRNCF		RLNCF		INFSNZ		DCFSNZ		CLRF		SETF		NEGW		DAW		
	3	CPFSLT	CPFSEQ	CPFSGT	TSTFSZ					BTG								
	4	MOVPPF																
	5	MOVFP																
	6	MOVFP																
	7	MOVFP																
	8	BSF								BCF								
	9	BTFSS								BTFSC								
	A	TLRD				TLWT				TABLRD				TABLWT				
	B	MOVLW	ADDLW	SUBLW	IORLW	XORLW	ANDLW	RETLW	LCALL	MOVLB								
	C	GOTO																
	D	GOTO																
	E	GOTO																
	F	CALL																

***:
 0000: NOP
 0001: unused
 0002: RETURN
 0003: SLEEP
 0004: CLRWDI
 0005: RETFIE
 0006: 00FF unused

 unused opcode (execute as NOP)

2.1 SPECIAL FUNCTION REGISTERS AS SOURCE/DESTINATION

PIC17C42's orthogonal instruction set allows read and write of all file registers, including special function registers such as PC and status registers. There are some special situations the user should be aware of:

ALUSTA as destination (file 04h): If an instruction writes to ALUSTA, the Z, C, DC and OV bits may be set or reset as a result of the instruction and overwrite the original data bits written. For example, executing CLRF 04 will clear register 04, and then set Z bit leaving 00000100b first in the register.

PCL as source or destination (file 02h): Read, write or read-modify-write on PCL (f02) have the following results:

- Read PCL (f02): PCH → PCLATH; PCL → d
- Write PCL (f02): PCLATH → PCH; 8 bit destination value → PCL
- Read-Modify-Write: PCL → ALU operand; PCLATH → PCH; 8 bit result → PCL

Where PCH = program counter high byte (not an addressable register), PCLATH = Program counter high holding latch (file f03), d = destination, W or f.

Bit Manipulation

All bit manipulation instructions are done by first reading the entire register, operating on the selected bit and writing the result back (read-modify-write). The user should keep this in mind when operating on special function registers, such as ports.

2.2 INSTRUCTION DESCRIPTION

ADDLW Add literal to W

Syntax: ADDLW k

Encoding:

1011	0001	kkkk	kkkk
------	------	------	------

Words: 1

Cycles: 1

Operation: $(W + k) \rightarrow W$

Status bits: OV, C, DC, Z

Description: The contents of the W register are added to the eight bit literal "k" and the result is placed in the W register.

ADDWF ADD W to f

Syntax: ADDWF f,d

Encoding:

0000	111d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: $(W + f) \rightarrow d$

Status bits: OV, C, DC, Z

Description: Add the contents of the W register to data memory location "f". If "d" is 0 the result is stored in the W register. If "d" is 1 the result is stored in data memory location "f".

ADDWFC ADD W and Carry to f

Syntax: ADDWFC f,d

Encoding:

0001	000d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: $(W + f + C) \rightarrow d$

Status bits: OV, C, DC, Z

Description: Add the W register and the Carry Flag to data memory location "f". If "d" is 0 the result is placed in the W register. If "d" is 1 the result is placed in data memory location "f".

ANDLW AND literal and W

Syntax: ANDLW k

Encoding:

1011	0101	kkkk	kkkk
------	------	------	------

Words: 1

Cycles: 1

Operation: $(W \text{ .AND. } k) \rightarrow W$

Status bits: Z

Description: The contents of W register are AND'ed with the eight bit literal "k". The result is placed in the W register.

ANDWF AND W with f

Syntax: ANDWF f,d

Encoding:

0000	101d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: $(W \text{ .AND. } f) \rightarrow d$

Status bits: Z

Description: AND the W register with data memory location "f". If "d" is 0 the result is stored in the W register. If "d" is 1 the result is stored in data memory location "f".

BCF Bit Clear f

Syntax: BCF f,b

Encoding:

1000	1bbb	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: $0 \rightarrow f(b)$

Status bits: None

Description: Bit "b" in data memory location "f" is reset to 0.

BSF Bit Set f

Syntax: BSF f,b

Encoding:

1000	0bbb	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: $1 \rightarrow f(b)$

Status bits: None

Description: Bit "b" in data memory location "f" is set to 1.

BTFSC Bit test, skip if clear

Syntax: BTFSC f,b

Encoding:

1001	1bbb	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1(2)

Operation: skip if $f(b) = 0$

Status bits: None

Description: If bit "b" in data memory location "f" is "0" then the next instruction is skipped.

If bit "b" is "0", the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed instead making this a 2 cycle instruction.

BTFSS Bit test, skip if set

Syntax: BTFSS f,b
 Encoding:

1001	0bbb	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1 (2)
 Operation: skip if f(b) = 1
 Status bits: None
 Description: If bit "b" in data memory location "f" is "1" then the next instruction is skipped.
 If bit "b" is "1", the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed instead making this a 2 cycle instruction.

BTG Bit Toggle f

Syntax: BTG f,b
 Encoding:

0011	1bbb	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: f(b) → f(b)
 Status bits: None
 Description: Bit "b" in data memory location "f" is inverted.

CALL Subroutine Call

Syntax: CALL k
 Encoding:

111k	kkkk	kkkk	kkkk
------	------	------	------

 Words: 1
 Cycles: 2
 Operation: PC + 1 → TOS, k → PC<12:0>, k<12:8> → PCLATH<4:0>; PC<15:13> → PCLATH<7:5>
 Status bits: None
 Description: Subroutine call within 8K page. First, return address (PC + 1) is pushed into the stack. The thirteen bit value is loaded into PC bits <12:0>. Then the upper eight bits of the PC is copied into PCLATH (f03). CALL is a two cycle instruction.

CLRF Clear f and Clear d

Syntax: CLRF f,d
 Encoding:

0010	100d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1

Operation: 00h → f, 00h → d
 Status bits: None
 Description: The contents of data memory location "f" are set to 0. If "d" is 0 the contents of both data memory location "f" and W register are set to 0. If "d" is 1 the only contents of data memory location "f" are set to 0.

CLRWDT Clear Watchdog Timer

Syntax: CLRWDT
 Encoding:

0000	0000	0000	1000
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: 00h → WDT, 0 → WDT prescaler,
 Status bits: 1 → TO, 1 → PD
 Description: CLRWDT instruction resets the watchdog timer. It also resets the prescaler of the WDT. Status bits TO and PD are set.

COMF Complement f

Syntax: COMF f,d
 Encoding:

0001	001d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: f → d
 Status bits: Z
 Description: The contents of data memory location "f" are complemented. If "d" is 0 the result is stored in W. If "d" is 1 the result is stored in data memory location "f".

CPFSEQ Compare f with W, skip if f = W

Syntax: CPFSEQ f
 Encoding:

0011	0001	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1 (2)
 Operation: f - W, skip if f = W
 Status bits: None
 Description: If the contents of data memory location "f" are equal to the contents of the W register, the next instruction is skipped.
 If f = W then the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed instead making this a 2 cycle instruction.

CPFSGT Compare f with W, skip if f > W

Syntax: CPFSGT f

Encoding:

0011	0010	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1 (2)

Operation: f - W, skip if f > W (unsigned comparison)

Status bits: None

Description: If the contents of data memory location "f" are greater than the contents of the W register, the next instruction is skipped. The subtraction is unsigned.

If f > W then the next instruction, fetched during the current instruction execution, is discarded. A NOP is executed instead making this a 2 cycle instruction.

CPFSLT Compare f with W, skip if f < W

Syntax: CPFSLT f

Encoding:

0011	0000	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1 (2)

Operation: f - W, skip if f < W (unsigned)

Status bits: None

Description: If the contents of data memory location "f" are less than the contents of the W register, the next instruction is skipped. The subtraction is unsigned.

If f < W then the next instruction, fetched during the current instruction execution, is discarded. A NOP is executed instead making this a 2 cycle instruction.

DAW Decimal Adjust W Register

Syntax: DAW f,d

Encoding:

0010	111d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: if [W<3:0> >9] .OR. [DC = 1] then W<3:0> + 6 → f<3:0>, d<3:0>;
if [W<7:4> >9] .OR. [C = 1] then W<7:4> + 6 → f<7:4>, d<7:4>;

Status bits: C

Description: DAW adjusts the eight bit value in the W register resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed

BCD result. If "d" is 0 the result is placed in the W register and data memory location "f". If "d" is 1 the result is placed only in data memory location "f".

The Decimal Adjust Algorithm is as follows:

Step 1: If the lower nibble of W is greater than 9 or if the DC flag (Digit Carry) is set from previous operations, then 06h is added to W.

Step 2: If upper nibble is greater than 9 or if C flag (Carry) is set following Step 1 operation, 60h is added to W

The Carry flag may be set as a result of Step 1 or Step 2 operation.

DECF Decrement f

Syntax: DECF f,d

Encoding:

0000	011d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: (f-1) → d

Status bits: OV, C, DC, Z

Description: Decrement data memory location "f". If "d" is 0 the result is stored in the W register. If "d" is 1 the result is stored in data memory location "f".

DECFSZ Decrement f, skip if 0

Syntax: DECFSZ f,d

Encoding:

0001	011d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1 (2)

Operation: (f - 1) → d; skip if result = 0

Status bits: None

Description: The contents of data memory location "f" are decremented. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is placed in data memory location "f". If the result is 0 the next instruction is skipped.

If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two cycle instruction.

DCFSNZ Decrement f, skip if not 0

Syntax: DCFSNZ f,d
 Encoding:

0010	011d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1 (2)
 Operation: (f-1) → d, skip if not 0
 Status bits: None
 Description: The contents of data memory location “f” are decremented. If “d” is 0 the result is placed in the W register. If “d” is 1 the result is placed in data memory location “f”.
 If the result is not 0, the next instruction, fetched during the current instruction execution is discarded. A NOP is executed instead making this a 2 cycle instruction.

GOTO Unconditional Branch

Syntax: GOTO k
 Encoding:

110k	kkkk	kkkk	kkkk
------	------	------	------

 Words: 1
 Cycles: 2
 Operation: k → PC<12:0>; k<12:8> → f3<4:0>, PC<15:13> → f3<7:5>
 Status bits: None
 Description: GOTO allows an unconditional branch anywhere within an 8K page boundary. The thirteen bit immediate value is loaded into PC bits <12:0>. Then the upper eight bits of PC are loaded into PCLATH (file 3). GOTO is always a two cycle instruction.

INCF Increment f

Syntax: INCF f,d
 Encoding:

0001	010d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: (f + 1) → d
 Status bits: OV, C, DC, Z
 Description: The contents of data memory location “f” are incremented. If “d” is 0 the result is placed in the W register. If “d” is 1 the result is place in data memory location “f”.

INCFSZ Increment f, skip if 0

Syntax: INCFSZ f,d
 Encoding:

0001	111d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1 (2)
 Operation: (f+1) → d, skip if result = 0
 Status bits: None
 Description: The contents of data memory location “f” are incremented. If “d” is 0 the result is placed in the W register.
 If “d” is 1 the result is placed in data memory location “f”. If the result is 0 the next instruction, fetched during the current instruction execution, is discarded. A NOP is executed instead making this the 2 cycle case.

INFSNZ Increment f, skip if not 0

Syntax: INFSNZ f,d
 Encoding:

0010	010d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1(2)
 Operation: (f+1) → d, skip if not 0
 Status bits: None
 Description: The contents of data memory location “f” are incremented. If “d” is 0 the result is placed in the W register.
 If “d” is 1 the result is placed in data memory location “f”. If the result is not 0 the next instruction, fetched during the current instruction execution, is discarded. A NOP is executed instead making this a 2 cycle instruction.

IORLW Inclusive OR literal with W

Syntax: IORLW k
 Encoding:

1011	0011	kkkk	kkkk
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: (W .OR. k) → W
 Status bits: Z
 Description: The contents of the W register are inclusively OR’ed with the eight bit literal “k”. The result is placed in the W register.

IORWF Inclusive OR W with f

Syntax: IORWF f,d

Encoding:

0000	100d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: (W .OR. f) → d

Status bits: Z

Description: Inclusive OR the W register with data memory location "f". If "d" is 0 the result is stored in the W register. If "d" is 1 the result is stored in data memory location "f".

LCALL Long Call

Syntax: LCALL k

Encoding:

1011	0111	kkkk	kkkk
------	------	------	------

Words: 1

Cycles: 2

Operation: PC+1 → TOS;
k → PCL, (PCLATH) → PCH

Status bits: None

Description: LCALL allows unconditional subroutine call to anywhere within the 64k program memory space. First, the return address (PC+1) is pushed onto the stack. A 16 bit destination address is then loaded into the program counter. The lower 8 bit of the destination address is embedded in the instruction. The upper 8 bit of PC is loaded from PC high holding latch, PCLATH. LCALL is a two cycle instruction.

Example: `MOVLW 56h ; W = 56h`
`MOVFP W,PCLATH ; PCLATH = 56h`
`LCALL 3Ah ; CALL 563Ah`

MOVFP Move f to p

Syntax: MOVFP f,p

Encoding:

011p	pppp	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: f → p

Status bits: None

Description: Move data from data memory location "p" to data memory location "f". Location "f" can be anywhere in the 256 word data space (00h to FFh) while "p" can be 00h to 1Fh.

Either "p" or "f" can be the W register (a useful special situation).

MOVFP is particularly useful to transfer a data memory location to a peripheral register (such as the transmit buffer or an I/O port). Both "f" and "p" can be indirectly addressed.

MOVLB Move Literal to BSR

Syntax: MOVLB k

Encoding:

1011	1000	kkkk	kkkk
------	------	------	------

Words: 1

Cycles: 1

Operation: k → BSR

Status bits: None

Description: The constant is loaded in Bank Select Register (BSR, 0Fh). Only the low 4 bits of the Bank Select Register are physically implemented.

MOVLW Move Literal to W

Syntax: MOVLW k

Encoding:

1011	0000	kkkk	kkkk
------	------	------	------

Words: 1

Cycles: 1

Operation: k → W

Status bits: None

Description: The eight bit literal "k" is loaded into W register.

MOVFP Move p to f

Syntax: MOVFP p,f

Encoding:

010p	pppp	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: p → f

Status bits: Z

Description: Move data from data memory location "p" to data memory location "f". Location "f" can be anywhere in the 256 byte data space (00h to FFh) while "p" can be 00h to 1Fh.

Either "p" or "f" can be the W register (an useful special situation)

MOVFP is particularly useful for transferring a peripheral register (e.g. the timer or an I/O port) to a data memory location.

MOVWF Move W to f

Syntax: MOVWF f
 Encoding:

0000	0001	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: W → f
 Status bits: None
 Description: Move data from W register to data memory location "f". Location "f" can be anywhere in the 256 word data space.

NEGW Negate W

Syntax: NEGW f,d
 Encoding:

0010	110d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: $\bar{W} + 1 \rightarrow f$; $\bar{W} + 1 \rightarrow d$
 Status bit: OV, C, DC, Z
 Description: The contents of the W register are negated using 2's complement. If "d" is 0 the result is placed in W register and data memory location "f". If "d" is 1 the result is placed only in data memory location "f".

NOP No Operation

Syntax: NOP
 Encoding:

0000	0000	0000	0000
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: No operation
 Status bits: None
 Description: No operation

RETFIE Return from Interrupt

Syntax: RETFIE
 Encoding:

0000	0000	0000	0101
------	------	------	------

 Words: 1
 Cycles: 2
 Operation: TOS → PC, 0 → GLINTD; PCLATH (f3) is unchanged
 Status bits: GLINTD
 Description: Return from Interrupt. Stack is popped and Top of the Stack (TOS) is loaded in PC. Interrupts are enabled by clearing GLINTD bit. GLINTD is global interrupt disable bit (bit 4, register CPUSTA). This is a two cycle instruction.

RETLW Return Literal to W

Syntax: RETLW k
 Encoding:

1011	0110	kkkk	kkkk
------	------	------	------

 Words: 1
 Cycles: 2
 Operation: k → W; TOS → PC; PCLATH (f03) is unchanged
 Status bits: None
 Description: The W register is loaded with the eight bit literal "k". The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged. This is a two cycle instruction.

RETURN Return from Subroutine

Syntax: RETURN
 Encoding:

0000	0000	0000	0010
------	------	------	------

 Words: 1
 Cycles: 2
 Operation: TOS → PC; PCLATH (f3) is unchanged
 Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.

RLCF Rotate Left f through Carry

Syntax: RLCF f,d
 Encoding:

0001	101d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: f<n> → d<n+1>; f<7> → C; C → d<0>
 Status bits: C
 Description: The contents of data memory location "f" are rotated one bit to the left through the Carry Flag. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is stored back in data memory location "f".

RLNCF Rotate Left f (no carry)

Syntax: RLNCF f,d
 Encoding:

0010	001d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: $f\langle n \rangle \rightarrow d\langle n+1 \rangle$; $f\langle 7 \rangle \rightarrow d\langle 0 \rangle$
 Status bits: None
 Description: The contents of data memory location "f" are rotated one bit to the left. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is stored back in data memory location "f".

RRCF Rotate Right f through Carry

Syntax: RRCF f,d
 Encoding:

0001	100d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: $f\langle n \rangle \rightarrow d\langle n-1 \rangle$; $f\langle 0 \rangle \rightarrow C$; $C \rightarrow d\langle 7 \rangle$
 Status bits: C
 Description: The contents of data memory location "f" are rotated one bit to the right through the Carry Flag. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is placed in data memory location "f".

RRNCF Rotate Right f (no carry)

Syntax: RRNCF f,d
 Encoding:

0010	000d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: $f\langle n \rangle \rightarrow d\langle n-1 \rangle$; $f\langle 0 \rangle \rightarrow d\langle 7 \rangle$
 Status bits: None
 Description: The contents of data memory location "f" are rotated one bit to the right. If "d" is 0 the result is placed in the W register. If "d" is 1 the result is placed in data memory location "f".

SETF Set f and Set d

Syntax: SETF f,d
 Encoding:

0010	101d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: $FFh \rightarrow f$, $FFh \rightarrow d$
 Status bits: None

Description: If "d" is 0 both the data memory location "f" and W register are set to FFh. If "d" is 1 the only the data memory location "f" is set to FFh.

SLEEP

Syntax: SLEEP
 Encoding:

0000	0000	0000	0011
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: $0 \rightarrow \overline{PD}$; $1 \rightarrow \overline{TO}$
 $00h \rightarrow WDT$; $0 \rightarrow WDT$ prescaler
 Status bits: \overline{TO} , \overline{PD}
 Description: The power down status bit (\overline{PD}) is cleared. Time-out status bit (\overline{TO}) is set. Watchdog Timer and its prescaler are cleared.

The processor is put into SLEEP mode with the oscillator stopped. See section on SLEEP mode for more details.

SUBLW Subtract W from literal

Syntax: SUBLW k
 Encoding:

1011	0010	kkkk	kkkk
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: $(k - W) \rightarrow W$
 Status bits: OV, C, DC, Z

Description: The contents of the W register are subtracted from the eight bit literal "k". The result is placed in the W register.

SUBWF Subtract W from f

Syntax: SUBWF f,d
 Encoding:

0000	010d	ffff	ffff
------	------	------	------

 Words: 1
 Cycles: 1
 Operation: $(f-W) \rightarrow d$
 Status bits: OV,C, DC, Z

Description: Subtract (2's complement method) the W register from data memory location "f". If "d" is 0 the result is stored in the W register. If "d" is 1 the result is stored back in data memory location "f".

SUBWFB Subtract W from f with Borrow

Syntax: SUBWFB f,d

Encoding:

0000	001d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: (f-W-C) → d

Status bits: OV, C, DC, Z

Description: Subtract (2's complement method) the W register and the carry flag (borrow) from data memory location "f". If "d" is 0 the result is stored in the W register. If "d" is 1 the result is stored in data memory location "f".

SWAPF Swap f

Syntax: SWAPF f,d

Encoding:

0001	110d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: f<0:3> → d<4:7>, f<4:7> → d<0:3>

Status bits: None

Description: The upper and lower nibbles of data memory location "f" are exchanged. If "d" is 0 the result is placed in W register. If "d" is 1 the result is place in data memory location "f".

TABLRD Table Read

Syntax: TABLRD t,i,f

Encoding:

1010	10ti	ffff	ffff
------	------	------	------

Words: 1

Cycles: 2 (3 cycle if f = 02h [PC])

Operation: If t = 1 then TBLATH → f
 else if t = 0 TBLATL → f;
 Prog Mem (TBLPTR) → TBLAT;
 if i = 1 then TBLPTR + 1 → TBLPTR

Status bits: None

Description: First, either the low byte (if t = 0) or the high byte (if t = 1) of the table latch (TBLAT) is moved to register file "f".
 Then the contents of the program memory location pointed to by the 16 bit Table Pointer (TBLPTR) is loaded into the 16 bit Table Latch (TBLAT). Finally table pointer is incremented if i = 1.

Example:

```
MOVLW 12h      ;
MOVWF  W, TBLPTRH ;
MOVLW 34h      ;
MOVWF  W,TBLPTRL ; TBLPTR = 1234h
TABLRD 0, 1, 50h ; TBLAT = Prog Mem
                    ; (1234h)
```

TBLPTR = 1235h
 ; low byte → 50h
 TABLRD 1,1, 51h ; high byte → 51h
 ; TBLAT = Prog Mem
 ; (1235h)
 ; TB1.PTR = 1236h
 TLRD 0, 52h ; low byte → 52h
 TLRD 1, 53h ; high byte → 53h

TABLWT Table write

Syntax: TABLWT t, i, f

Encoding:

1010	11ti	ffff	ffff
------	------	------	------

Words: 1

Cycles: 2 (Many if write is to on-chip EPROM program memory)

Operation: if t = 0 then f → TBLATL
 else if t = 1 then f → TBLATH;
 TBLAT → Prog Mem (TBLPTR);
 if i = 1 then TBLPTR + 1 → TBLPTR;

Description: First, contents of file register f is loaded in the low byte (if t = 0) or high byte (if t = 1) of Table Latch, TBLAT.
 If TBLPTR points to external program memory location then the contents of TBLAT is written to it and the instruction takes 2 cycles.
 If TBLPTR points to an internal EPROM location, then an EPROM write (program) sequence is initiated. It is terminated when an interrupt is received.

If the Global Interrupt Disable bit (GLINTD) is set, the interrupt will complete the TABLWT, but no interrupt sequence will be invoked. If GLINTD = 0, then interrupt will be acknowledged following the TABLWT.

For an interrupt to end programming, its corresponding mask bit must enable the interrupt. If the terminating interrupt is INTIR, RTCIR or RTXIR, the flag bit is automatically cleared. The clearing takes place for both short and long table writes. The user can protect against accidental clearing of an interrupt flag due to a TABLWT instruction by masking off the above mentioned interrupts before doing table write operations.

MCLR/V_{PP} pin must be at programming voltage for successful programming. If MCLR/V_{PP} = V_{CC} then the programming sequence will be executed, but will not be successful (although the location may be disturbed).

TLRD **Table Latch Read**

Syntax: TLRD t,f

Encoding:

1010	00tx	ffff	ffff
------	------	------	------

x= don't care

Words: 1

Cycles: 1

Operation: if (t = 0) then TBLATL → f else if (t = 1) then TBLATH → f

Status bits: None

Description: Read data from high byte (t = 1) or low byte (t = 0) of 16 bit Table Latch into file register "f". Table Latch is unaffected.

This instruction is used in conjunction with TABLRD to transfer data from program memory to data memory.

TLWT **Table Latch Write**

Syntax: TLWT t,f

Encoding:

1010	01tx	ffff	ffff
------	------	------	------

x= don't care

Words: 1

Cycles: 1

Operation: if (t=0) then f → TBLATL else if (t = 1) then f → TBLATH

Status bits: None

Description: Data from file register f is written into the low byte (t = 0) or the high byte (t = 1) of the 16 bit Table Latch.

This instruction is used in conjunction with TABLWT, to transfer data from data memory to program memory.

TSTFSZ **Test f, skip if 0**

Syntax: TSTFSZ f

Encoding:

0011	0011	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1 (2)

Operation: skip if f = 0

Status bits: None

Description: If the contents of data memory location "f" are 0 then the next instruction is skipped.

If "f" = 0, the next instruction, fetched during the current instruction execution, is discarded. A NOP is executed instead making this a 2 cycle instruction.

XORLW **Exclusive OR literal with W**

Syntax: XORLW k

Encoding:

1011	0100	kkkk	kkkk
------	------	------	------

Words: 1

Cycles: 1

Operation: (W .XOR. k) → W

Status bits: Z

Description: The contents of the W register are XOR'ed with the eight bit literal "k". The result is placed in the W register.

XORWF **Exclusive OR W with f**

Syntax: XORWF f,d

Encoding:

0000	110d	ffff	ffff
------	------	------	------

Words: 1

Cycles: 1

Operation: (W .XOR. f) → d

Status bits: Z

Description: Exclusive OR the contents of the W register with data memory location "f". If "d" is 0 the result is stored in the W register. If "d" is 1 the result is stored in data memory location "f".

3.0 HARDWARE DESCRIPTION OF THE CPU

3.1 INDIRECT ADDRESSING REGISTERS (FILES 00h & 08h)

These two register locations (not physically implemented) are used to implement indirect addressing of data memory space. An instruction using file address of 0 or 8 actually accesses the data memory location pointed to by the corresponding FSR register (file 1 or file 9). If file 00h (or file 08h) itself is read indirectly via an FSR, all zeroes are read. Similarly, if file 00h (or file 08h) is written to indirectly, the operation will be equivalent to a NOP.

Single cycle data transfers within the entire data space are possible with MOVFP and MOVFP instructions, when "p" is specified as "00h" and "q" as "08h", or vice versa.

3.2 FILE SELECT REGISTERS (FSR0 AND FSR1, FILES 01h AND 09h)

These two registers are 8 bit wide indirect address pointers for data memory. They can be auto-incremented, auto-decremented or left unchanged after each access as determined by the 4 control bits in the status register "ALUSTA" (File 04h bits 7:4). See figure 3.8.1.

3.3 TABLE POINTER (TBLPTRL FILES AND TBLPTRH, FILES 0Dh AND 0Eh)

File registers 0Dh and 0Eh form a 16 bit pointer to address the 64K program memory space. The table pointer is used by instructions TABLWT and TABLRD. The TABLRD and the TABLWT instructions allow transfer of data between program and data space. The table pointer serves as the 16 bit address of the data word within the program memory.

3.4 TABLE LATCH (TBLATH, TBLATL)

The table latch (TBLAT) is a 16 bit register, consisting of TBLATH and TBLATL refer to the high and low bytes of the register. It is not mapped into data or program memory. The table latch is used as a temporary holding latch during data transfer between program and data memory (see descriptions of instructions TABLRD, TABLWR, TLRD and TLWR).

3.5 PROGRAM COUNTER MODULE

The program counter (PC) is a 16 bit register. PCL, the low byte of the PC, is mapped in the data memory (file 02h). PCL is readable and writable just as any other register. PCH is the high byte of the PC and is not directly

addressable since PCH is not mapped in data or program memory. An 8 bit register PCLATH (PC high latch) is used as a holding latch for the high byte of the PC. PCLATH is mapped into data memory (file 03h). The user can read or write PCH through PCLATH.

The 16 bit wide PC is incremented after each instruction fetch during Q1 unless modified by GOTO, CALL, LCALL, RETURN, RETLW, or RETFIE instruction or interrupt response or due to destination write to PCL by an instruction. "Skip"s are equivalent to incrementing the PC twice.

The operations of the PC and PCLATH for different instructions are as follows:

- a) LCALL:
PCLATH → PCH, IR<7:0> → PCL (PCL is loaded with 8 bit destination address embedded in the instruction. PCLATH is unchanged.
- b) CALL, GOTO:
A 13 bit destination address is provided in the instruction
IR<12:0> → PC <12:0>
PC<15:13> → PCLATH<7:5>
- c) Read f03 (Any instruction that reads PCL):
PCL → data bus → ALU or destination
PCH → PCLATH
- d) Write f03 (Any instruction that writes to PCL):
8 bit data → data bus → PCL
PCLATH → PCH
- e) Read-Modify-Write (Any instruction that does a read-write-modify operation on f02, such as ADDWF f02)
Read: PCL → data bus → ALU
Write: 8 bit result → data bus → PCL
PCLATH → PCH

Note that read-modify-write only affects the PCL with the result. PCH is loaded with PCLATH. Thus, ADDWF f02, for example will result in a jump within the current page. If PC = 03F0h, W = 30h and PCLATH = 03h before instruction, PC = 0320h after the instruction. To accomplish a true 16 bit computed jump, the user needs to compute the 16 bit destination address, write the high byte to PCLATH and then write the low value to PCL.

The following PC related operations do not change f03h:

- a) LCALL, RETLW, RETURN, RETFIE instructions,
- b) Interrupt vector is forced onto the PC,
- c) Read-modify-write instructions (e.g. BSF 02) on f02h.

3.6 STACK

The PIC17C42 has a 16 word x 16 bit hardware stack which is not part of data or program space. The PC is pushed onto the stack if CALL or LCALL instructions are executed or if an interrupt is responded to by branching to the corresponding interrupt vector. The stack is POPed

into the PC if a RETURN, RETLW or RETFIE instruction is executed. The top of the stack is not addressable in any other way.

3.6.1 Stack Available Status Bit (Bit 5, CPUSTA)

STKAVL is a read only status bit that indicates any stack overflow error. STKAVL is set to '1' on reset and stays '1' unless the following situation occurs:

If stack is full: i.e. there are 16 entries in the stack the STKAVL is set to '0'. If, the stack is popped (by RETURN, RETLW or RETFIE instruction) then STKAVL is set to '1' again indicating 'stack availability'.

If, however, a push takes place instead (due to CALL, LCALL or interrupt), then stack overflow occurs. In this event the first entry is lost and STKAVL is permanently cleared to '0'. Under this condition, the only way STKAVL will set to '1' is via reset.

STKAVL usage caution: If the stack is empty, a POP (due to RETURN, RETLW or RETFIE) followed by a PUSH, will permanently clear STKAVL to '0'.

For a description of CPUSTA register, see figure 4.5.1.

3.6.2 Using the STKAVL bit

One way to use the STKAVL bit is to test it at the beginning of every subroutine or interrupt service routine. If STKAVL = 0, then all stack locations are used (and presumably no error has occurred yet). In such case, interrupts must be disabled in the subroutine. Also, no subroutine calls must be made unless software stack management is invoked.

3.7 INTERRUPT LOGIC

The PIC17C42 has 11 interrupt sources that are mapped into 4 interrupt vectors. The interrupt logic is controlled by the INTSTA register and the global interrupt disable bit (GLINTD) in CPUSTA register, file f06h. See figure 4.5.1 for a description of CPUSTA register. Four hard-wired vectors allow fast interrupt response time. Worst case latency is 3 instruction cycles when only one interrupt at a time is being serviced. Interrupt nesting to

3.7.1 TABLE OF INTERRUPTS

Interrupt flag	Flag location bit, Register	Interrupt mask bit	Mask bit location bit, Register	Interrupt Source	Priority	Vectors to
INTIR RTCIIR RTXIR PEIR	bit 4, INTSTA bit 5, INTSTA bit 6, INTSTA bit 7, INTSTA	INTIE RTCIE RTXIE PEIE	bit 0, INTSTA bit 1, INTSTA bit 2, INTSTA bit 3, INTSTA	External interrupt on INT pin RTCC overflow interrupt External interrupt on RT pin Any peripheral interrupt	Highest priority 2nd priority 3rd priority Lowest priority	0008h 0010h 0018h 0020h
IRB TM3IR TM2IR TM1IR CA2IR CA1IR TBMT RBFL	bit 7, PIR bit 6, PIR bit 5, PIR bit 4, PIR bit 3, PIR bit 2, PIR bit 1, PIR bit 0, PIR	IEB TM3IE TM2IE TM1IE CA2IE CA1IE TXIE RCIE	bit 7, PIE bit 6, PIE bit 5, PIE bit 4, PIE bit 3, PIE bit 2, PIE bit 1, PIE bit 0, PIE	Port B input change interrupt Timer/Counter3 interrupt Timer/Counter2 interrupt Timer/Counter1 interrupt Capture1 interrupt Capture2 interrupt Serial port transmit interrupt Serial port receive interrupt	lowest priority (All these peripheral interrupts are OR'ed together to generate PEIR)	0020h

multiple levels is possible by enabling interrupts within the service routine. When an interrupt occurs, the current PC value is pushed onto the stack and the vector corresponding to the interrupt source is loaded into the PC.

3.7.1 Interrupt Flag and Mask Bits

Each interrupt has a request flag bit and a mask bit associated with it. The registers that hold these bits are INTSTA (file 07h), PIR (Bank 1, file 16h) and PIE (Bank 1, file 17h). See table 3.7.1 for details.

Interrupt flag bits INTIR, RTCIR or RTXIR are cleared automatically in hardware. PEIR is not cleared automatically since it is not a latched bit. PEIR is simply the OR of all the individual peripheral interrupt flag bits such as IRB, TM3IR, etc. Therefore if PEIR is the source of the interrupt, the user must clear, in software, the actual peripheral interrupt flag bit. The global interrupt disable bit, GLINTD, is set, in any case, preventing any further interrupt. To enable interrupts from the service routine the user must clear GLINTD. The user, must first clear the current interrupt flag bit to prevent recursive vectoring to the same service routine.

The TABLWT instruction, in a long write situation (i.e. writing to on-chip EPROM location) must be terminated with an interrupt. On completion, TABLWT clears the interrupt flag in the same exact fashion as an interrupt response, i.e. INTIR, RTCIR or RTXIR flag will be cleared if responsible for ending the TABLWT.

3.7.2 Peripheral interrupts

All peripherals use the same interrupt vector, 0020h. The individual peripheral interrupt request bits are "OR-ed" together. When multiple peripheral interrupt sources are enabled, the priorities have to be determined by software. Each peripheral has its own interrupt enable and request bit(s). In addition, the PEIE (Peripheral Interrupt Enable) bit acts as a global enable bit for all peripheral interrupts. There is a common peripheral interrupt request status bit (PEIR, bit 7, register INTSTA) which is a logical OR of all the individual peripheral interrupt request flags. This is a read only status bit useful for quickly determining if any peripheral request is outstanding.

FIGURE 3.7.1.1: REGISTER INTSTA

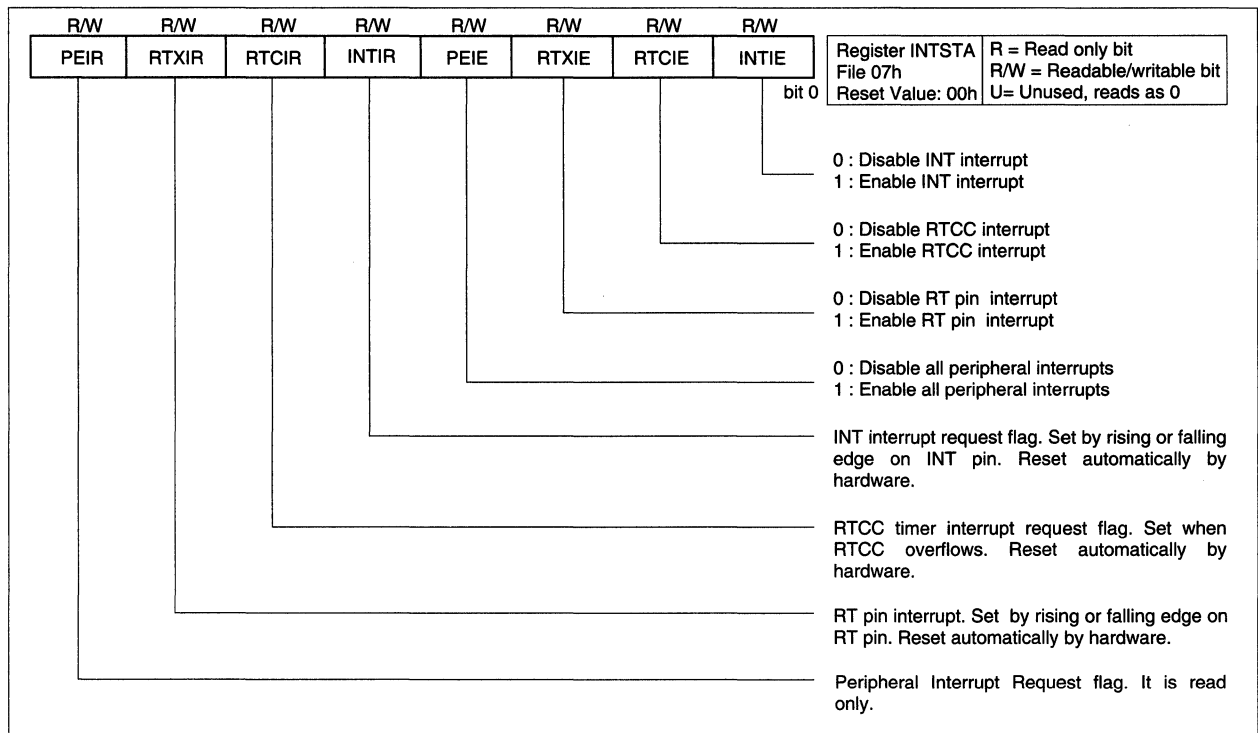


FIGURE 3.7.1.2: PIR (PERIPHERAL INTERRUPT REQUEST) REGISTER

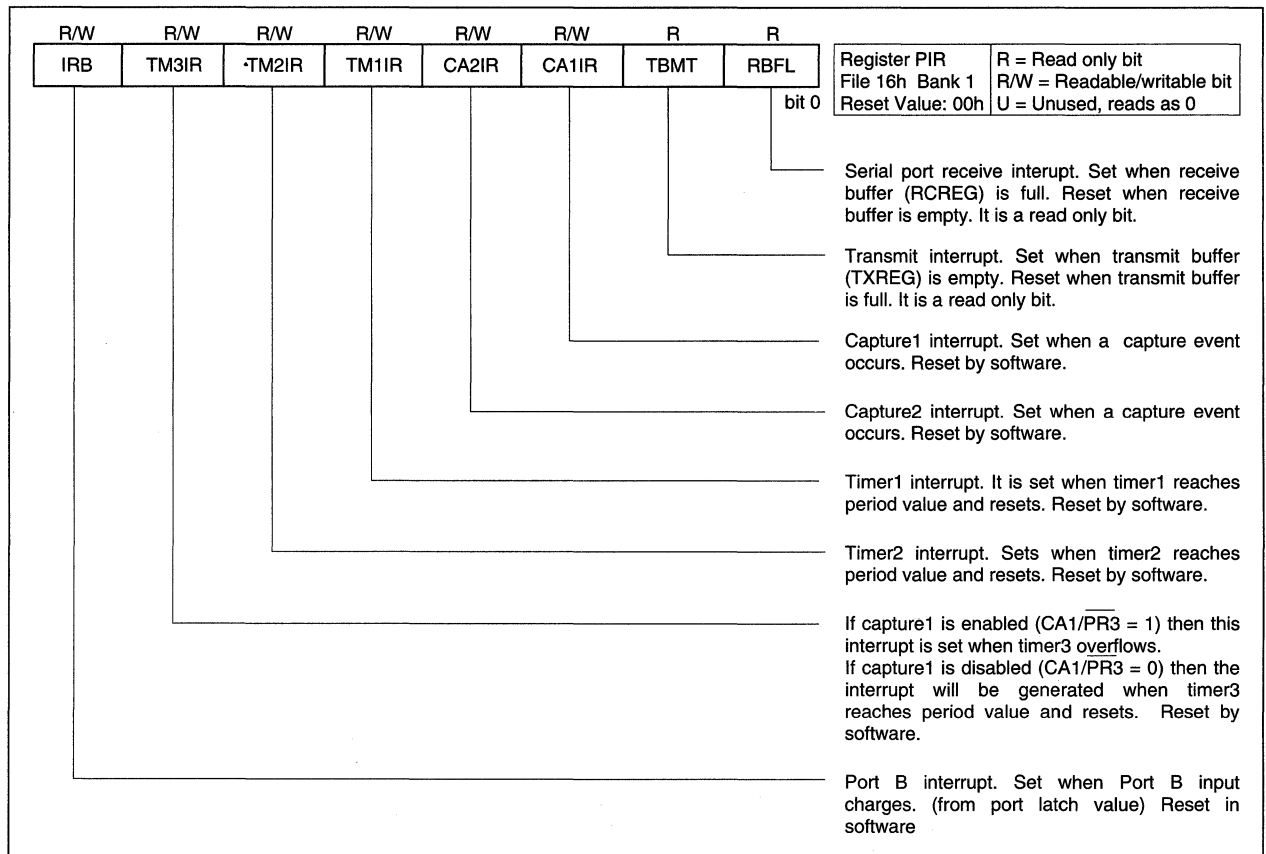
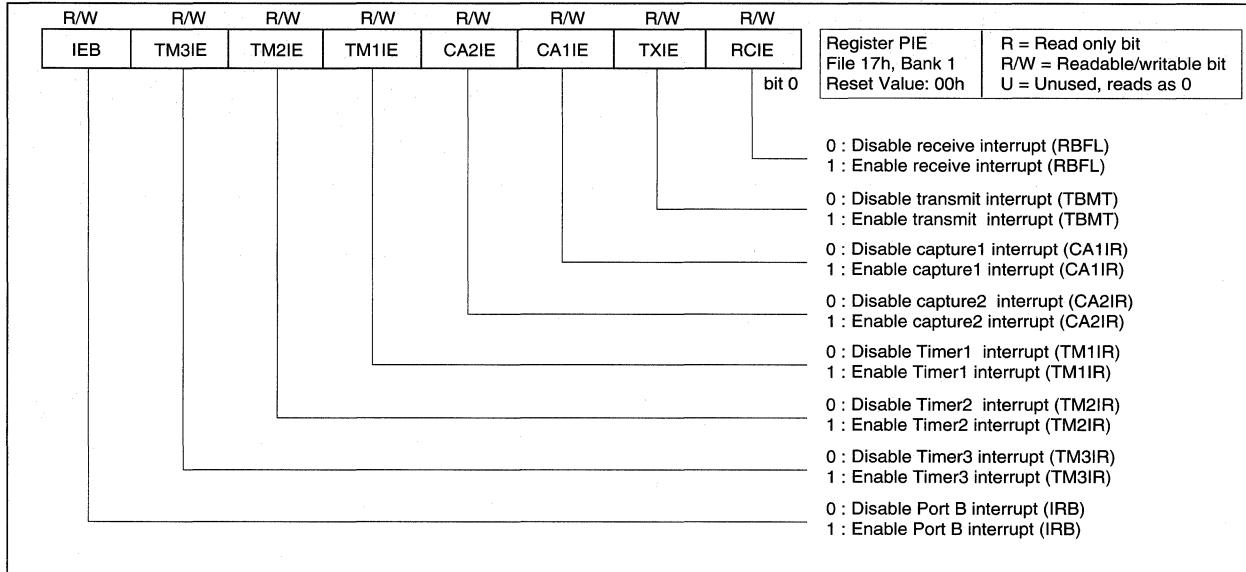


FIGURE 3.7.1.3: PIE (PERIPHERAL INTERRUPT ENABLE) REGISTER



3.7.3 INT and RT External Interrupts

INT and RT external interrupts can be positive or negative edge triggered, selectable in software. INT interrupt is generated on falling edge if INTEDG = '0' or on rising edge if INTEDG = '1'. Similarly, setting bit RTEDG = '0' will generate RT pin interrupt on falling edge whereas RTEDG = '1' will trigger RT interrupt on rising edge. The timing requirements on INT and RT inputs are as follows:

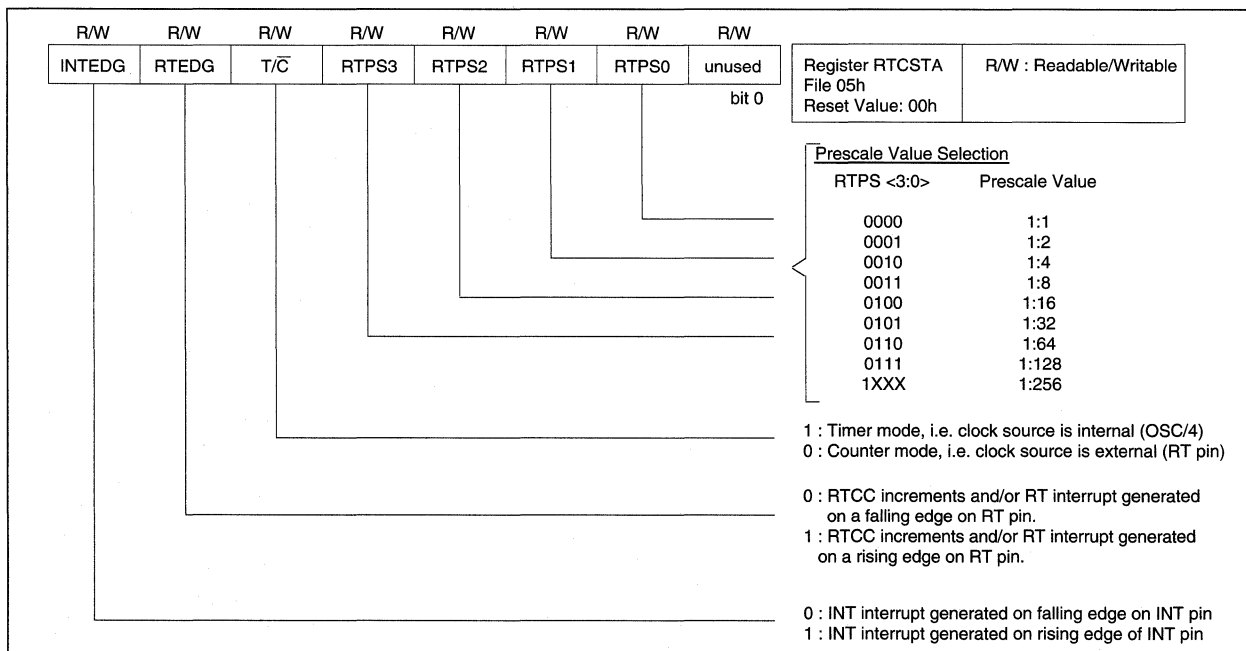
$$t_{INTH} = t_{RTIH} = \text{INT or RT high time} \geq 25\text{ns}$$

$$t_{INTL} = t_{RTL} = \text{INT or RT low time} \geq 25\text{ns}$$

Please note that changing edge selection for INT or RT pin may generate a false interrupt. The user should clear the INTIR or the RTXIR bit after changing edge setting.

See RTCSTA (register file 05h) for bit allocation.

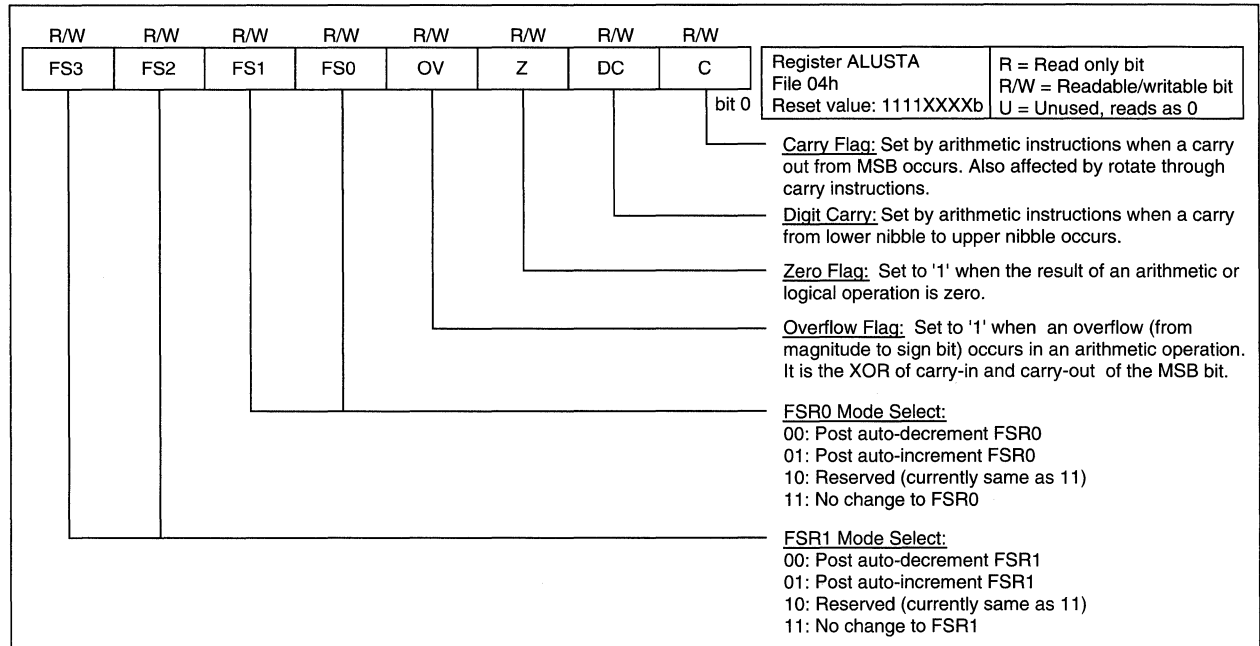
FIGURE 3.7.3.1: RTCSTA: RTCC STATUS/CONTROL REGISTER



3.8 ALU

The Arithmetic and Logic Unit of the PIC17C42 is capable of carrying out arithmetic or logical operations on two operands or a single operand. All single operand instructions operate either on the W register or a file register. For two operand instructions, one of the operands is the W register and the other one is either a file register or an 8 bit immediate constant.

FIGURE 3.8.1: ALUSTA (ALU STATUS) REGISTER



4.0 SPECIAL FEATURES OF THE CPU

What sets apart a microcontroller from other processors the most are special circuits to deal with the needs of real time applications. The PIC17C42 has a host of such features intended to maximize system reliability, minimize cost through elimination of costly external components, provide power saving operating modes and offer code protection.

The PIC17C42 has a watchdog timer which can be shut off only through EPROM fuses. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on power-up. One is the oscillator start-up timer (OST), intended to keep the chip in reset until the crystal oscillator is stable. The other is the power-up timer (PWRT), which provides a fixed delay of 80 ms nominal on power up only, designed to keep the part in reset while the power supply stabilizes. With these two timers on chip, most applications need no external reset circuitry. The SLEEP mode is designed to offer a very low current power-down mode. The user can wake up from SLEEP through external reset, watchdog timer time-out or through an interrupt. Several oscillator options are also made available to allow the part to fit the application. The RC oscillator option saves system cost while the LF (low frequency) crystal/resonator option saves power. A set of EPROM configuration bits (fuses) are used to select various options. Additional EPROM fuses are included for code-security.

4.1 RESET

The reset logic resets the complete PIC17C42 circuitry as follows:

- Oscillator buffer is enabled (i.e. oscillator is restarted if waking up from SLEEP through reset).
- Program Counter is reset to 0000h.
- All registers are reset as described in Table 1.6.2.
- Watchdog timer & its prescaler are cleared.
- Internal phase clock generator is held in Q1 state. If external execution is selected, ALE output is held low while OE and WR outputs are driven high.
- I/O ports B, C, D and E are configured as inputs. In case of port B, the weak pull-ups are activated. Ports RA2 and RA3 revert to high impedance state.

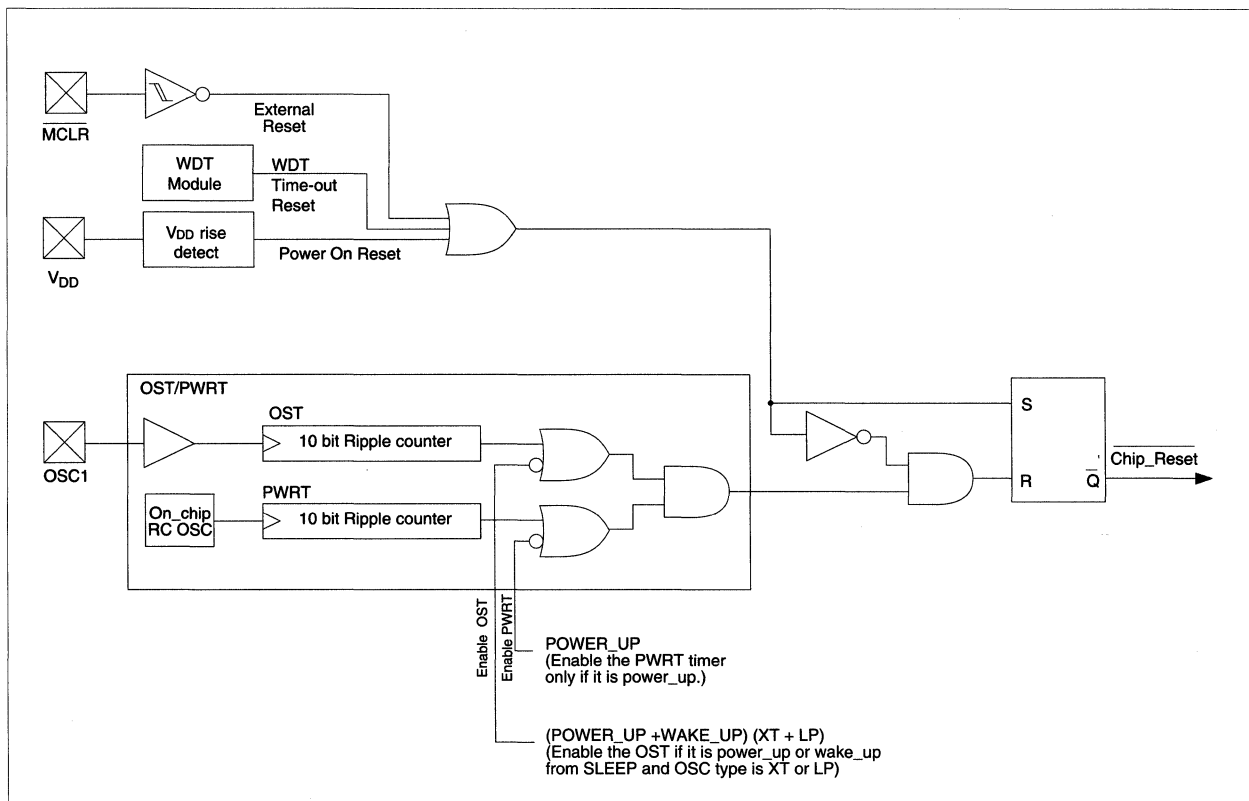
There are three events which can cause a device reset.

- a) Power On Reset :VDD rise is detected (1.2V - 2.0V range)
- b) External reset: "Low" level on the $\overline{\text{MCLR}}$ input
- c) WDT reset: Watchdog timer Time out

The RESET condition is maintained as long as

- a) the $\overline{\text{MCLR}}$ input is "low"
- b) $\overline{\text{MCLR}}$ has gone high but the Power-up timer (PWRT) is active, (i.e. has not timed out)
- c) $\overline{\text{MCLR}}$ has gone high but the oscillator start-up timer (OST) is active (i.e. has not timed out)

FIGURE 4.1.1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



4.2. OSCILLATOR

The PIC17C42 can accept an external clock input on OSC1 pin or will run off external crystal or ceramic resonator connected between OSC1 and OSC2 pins. It also has an RC oscillator mode in which an external R and C combination can be connected to OSC1 pin. The choice is made by EPROM fuses FOSC1 and FOSC0. These fuses are mapped in program memory locations FE01h and FE00h respectively. Refer to section 4.8 for details on the fuses.

TABLE 4.2.1: OSCILLATOR OPTIONS

Fosc1, Fosc0 Fuses	Mode	OSC1 Pin Function	OSC2 Pin Function	Freq. Range
11	EC: External Clock input	External clock input	CLKOUT output	DC-16Mhz
01	RC: RC oscillator mode	External RC oscillator connection	CLKOUT output	DC-4Mhz
10	XT: Crystal oscillator mode	Crystal connection	Crystal connection	0.2-16Mhz
00	LF: Low frequency crystal oscillator mode	Crystal connection	Crystal connection	32-200Khz

Note: 0 implies a programmed fuse.

4.2.1 EC: External Clock Input Mode:

The OSC1 input can be driven by CMOS drivers (figure 4.2.1A). In this mode, the OSC1 pin is a high impedance CMOS input. The OSC2 pin outputs CLKOUT (frequency = fosc/4). See Figure 1.2.1 for timing of CLKOUT.

4.2.2 RC: RC Oscillator Mode:

An external R and C combination can be connected to OSC1 pin (figure 4.2.1B). The RC oscillator mode provides a very cost effective solution. However, the frequency of oscillation will vary with V_{CC}, temperature and from chip to chip due to process variation. It is, therefore, not the right choice for timing sensitive applications where accurate oscillator frequency is desired. The OSC2 pin, in this mode, outputs CLKOUT (freq. = fosc/4). See Figure 1.2.1 for timing of CLKOUT.

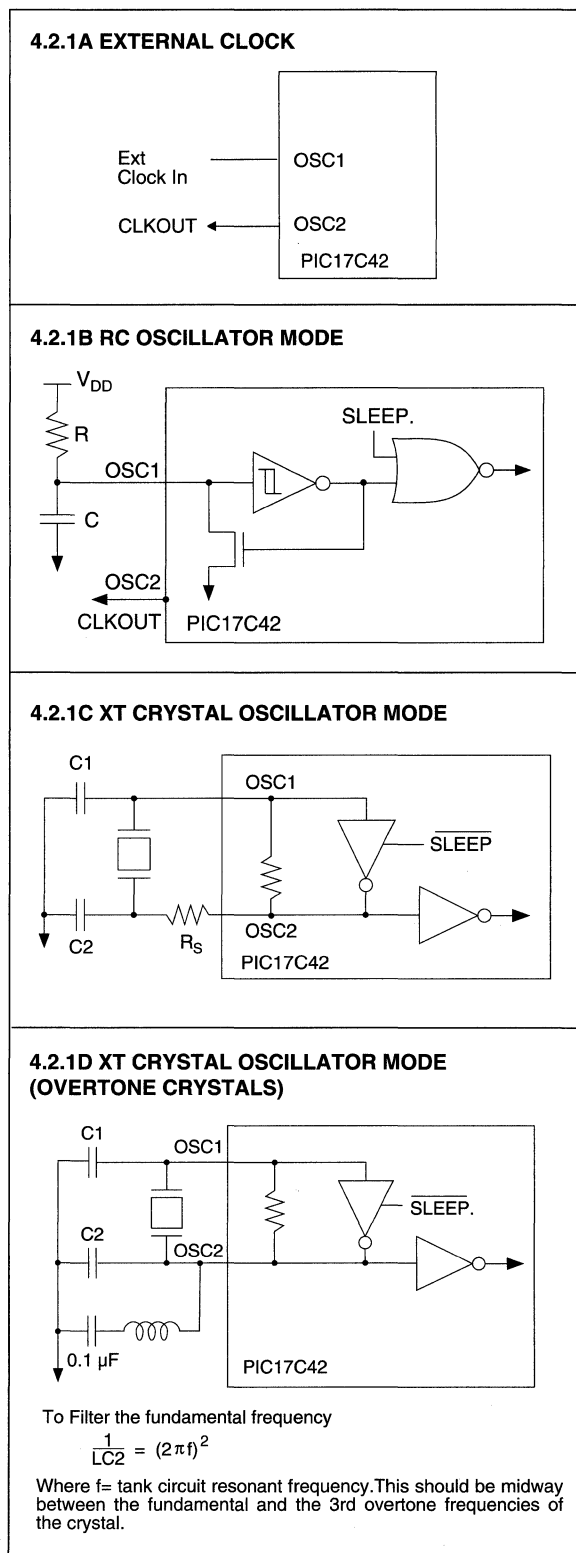
4.2.3 XT: Crystal Oscillator Mode:

In this mode a crystal or a ceramic resonator can be connected across OSC1 and OSC2. (figure 4.2.1C). The crystal must be of fundamental mode. If an overtone mode crystal is used (which is common above 20 MHz) then a tank circuit must be used to attenuate the gain at fundamental frequency (figure 4.2.1D)

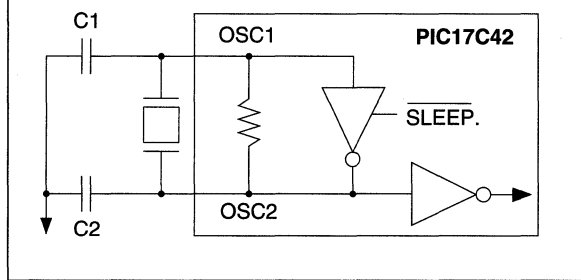
4.2.4 LF: Low Frequency Crystal Oscillator Mode:

This is same as the XT mode, (figure 4.2.1E) except that it is suitable for crystals of frequency range 32 KHz to 200 KHz.

FIGURE 4.2.1: DIFFERENT OSCILLATOR/ CLOCKIN OPTIONS



4.2.1E LF CRYSTAL OSCILLATOR MODE



4.3 OSCILLATOR START-UP TIMER (OST)

The OST provides a 1024 oscillator period delay on power-up and on wake up from SLEEP. This delay is provided by a 10 bit ripple counter. On power-up, the delay begins from the rising edge of \overline{MCLR} . On wake-up from SLEEP the time-out is counted from the time the wake-up event occurs. Since the OST counts oscillator signal on OSC1 pin, the counter only starts counting when amplitude on OSC1 pin reaches a certain acceptable limit. The OST time-out allows the crystal oscillator (or resonator) to stabilize before the chip is taken out of reset. The circuit will function with crystals of any frequency. This time-out is not invoked in RC oscillator mode or external clock (EC) mode.

4.4 POWER-UP TIMER (PWRT) AND POWER-ON RESET (POR)

The function of the PWRT timer is to provide a fixed 80 ms (typical) delay only on power-up. This is provided by a 10 bit ripple counter whose input clock comes from an on chip RC oscillator. The time-out is counted from the rising edge of \overline{MCLR} . The purpose of this time-out is to allow the V_{DD} supply to reach acceptable level before the part is taken out of reset.

An internal Power-on Reset pulse (POR) is generated when a V_{DD} rise is detected during initial power-up of the chip. (when $V_{DD} = 1.2V$ to $2.0V$ nominally). The POR signal resets internal registers as described in table 1.6.2. The user should note that the on-chip circuitry does not generate an internal reset when V_{DD} goes down, i.e., it does not provide brown out protection. Figure 4.4.1 and 4.4.2 shows possible external brown-out protection circuits. Also V_{DD} must come up from V_{SS} (nominal) for a POR signal to be generated. The PWRT timer and OST timer guarantee proper power-on reset without external components. This is done by simply tying the \overline{MCLR} pin to V_{DD} (figure 4.4.4). As V_{DD} comes up, POR is generated and \overline{MCLR} is sensed as '1' inside the chip, both OST and PWRT timers begin time-out. The 80 ms (nominal) delay of the PWRT allows V_{DD} to rise above V_{DD} min. spec. If the rise time of V_{DD} is much slower such that at the end of the time-out V_{DD} has not reached an acceptable level (as in figure 4.4.6) then external RC delay must be added on \overline{MCLR} pin.

The following table shows the time-outs for different oscillator types.

Oscillator Type	Power-up	Wake-up from SLEEP
EC	80 ms	—
RC	80 ms	—
XT	Greater of 80 ms and 1024 tosc	1024 tosc
LP	Greater of 80 ms and 1024 tosc	1024 tosc

FIGURE 4.4.1: BROWN OUT PROTECTION CIRCUIT

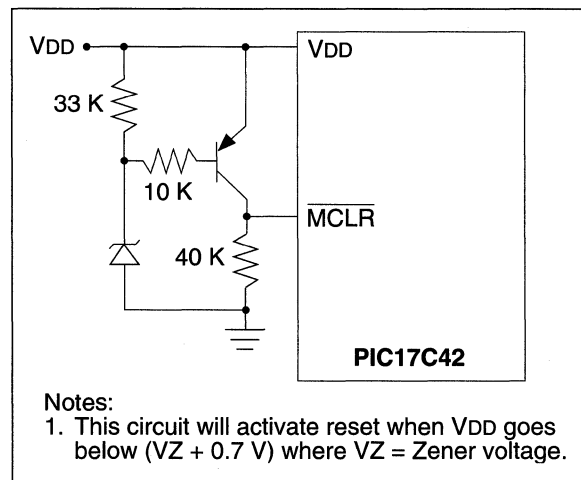


FIGURE 4.4.2: BROWN OUT PROTECTION CIRCUIT

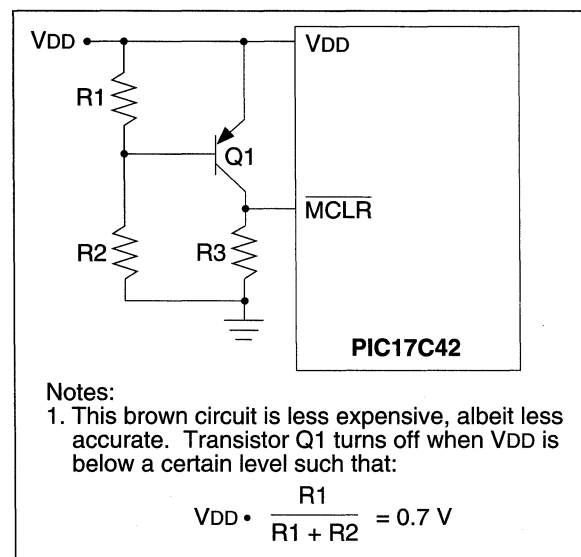


FIGURE 4.4.3: EXTERNAL RESET PULSE

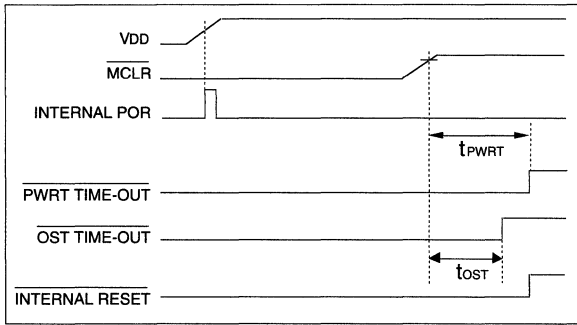


FIGURE 4.4.4: USING ON-CHIP POR

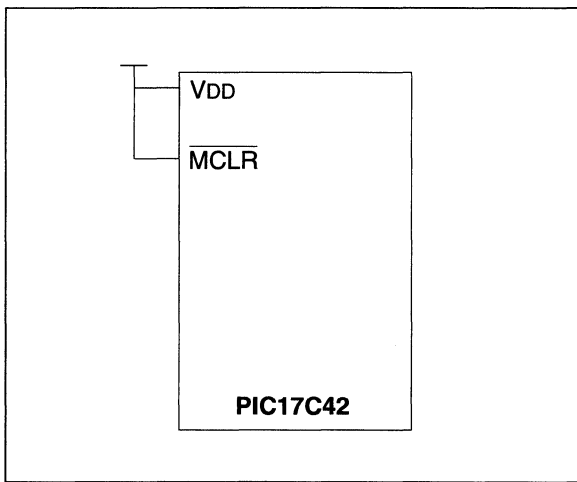
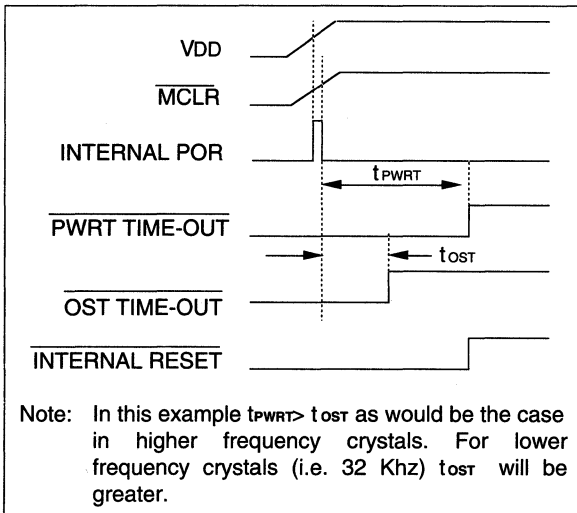
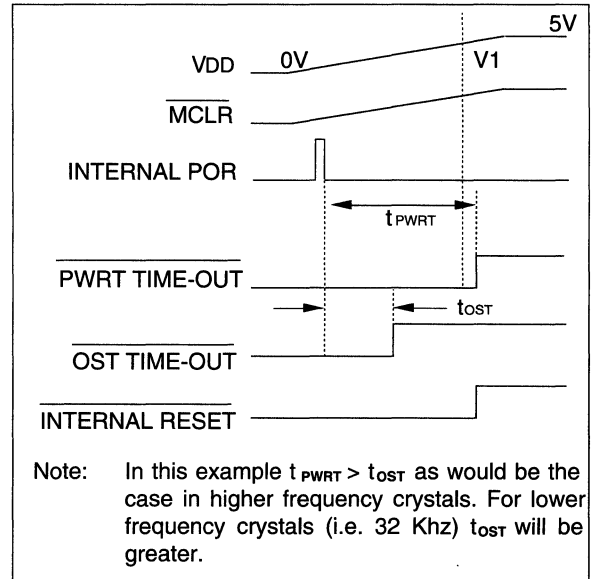


FIGURE 4.4.5: INTERNAL RESET (VDD AND MCLR TIED TOGETHER)



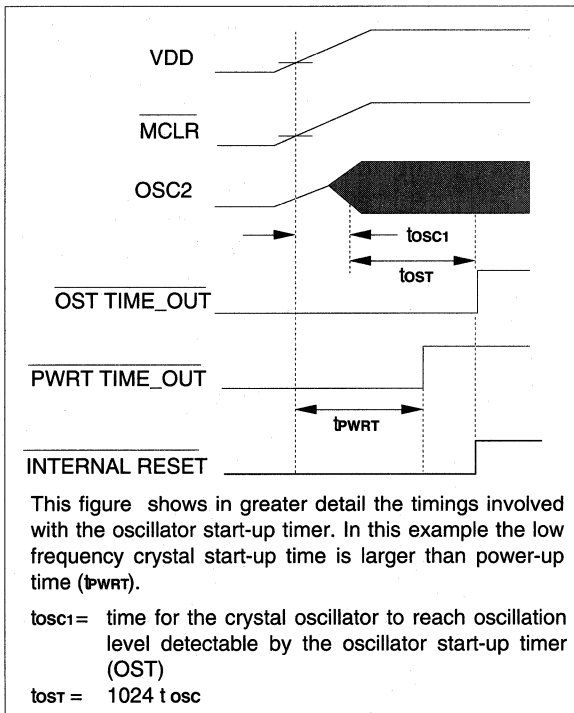
Note: In this example $t_{PWRT} > t_{OST}$ as would be the case in higher frequency crystals. For lower frequency crystals (i.e. 32 KHz) t_{OST} will be greater.

FIGURE 4.4.6: INTERNAL RESET (VDD AND MCLR TIED TOGETHER): SLOW VDD RISE TIME



Note: In this example $t_{PWRT} > t_{OST}$ as would be the case in higher frequency crystals. For lower frequency crystals (i.e. 32 KHz) t_{OST} will be greater.

FIGURE 4.4.7: OST START UP TIMING DETAILS



4.5 SLEEP MODE

The full static design of the PIC17C42 makes it possible to put the part in a power saving SLEEP (or power down) mode in which all on chip clocks are stopped.

The SLEEP mode, entered by executing a SLEEP instruction, shuts down the oscillator, sets \overline{TO} (bit3, CPUSTA), clears \overline{PD} (bit2, CPUSTA), the watchdog timer and its prescaler. In XT or LP mode, both OSC1 and OSC2 are placed into high-impedance state. In EC and RC modes, OSC1 pin is placed in high-impedance state while OSC2 is driven low. No clocks are presented to the internal logic even when an external clock is present on the OSC1 pin. The chip will remain in a completely static condition with the following exceptions:

- a) If the watchdog timer is enabled, it will keep running and will consequently wake up the chip on time-out.
- b) Signal edges on the RT pin (rising or falling whichever is defined to be the active edge by the RTEDG control bit) will increment the RTCC prescaler (an asynchronous ripple counter) if an external clock source is selected for RTCC. The RTCC itself will not increment.
- c) Any external interrupt event, such as RT, INT, capture1 or capture2 interrupt will wake the processor provided the corresponding interrupt mask bit was enabled when entering SLEEP mode. If global interrupt disable is off ($GLINTD=0$) then the chip will jump to corresponding interrupt vector on wake-up. Otherwise the chip will wake up and resume executing without responding to the interrupt (i.e. will not branch to interrupt vector).

- d) Any peripheral operating independent of the internal processor clock can change its status due to external events. Specifically, the serial port receive shift register will shift in data in synchronous slave (external clock) mode.

Besides the on-chip oscillator, any circuitry that consumes current is turned off in SLEEP mode. This includes the entire EPROM and, in particular, the EPROM fuses. The only fuses that will remain active are the WDT fuses (FWDT0, FWDT1). If minimal SLEEP current is desired, the user should consider turning off the watchdog timer. Since fuses consume current in '1' state. Turning WDT off not only saves the operating current it requires, but also saves fuse current due to FWDT1 or FWDT0 fuses. All I/O pins maintain their status during SLEEP.

4.5.1 Wake-up from SLEEP

Once the chip has entered the SLEEP mode it can only be awakened by one of the following events:

- a) Bringing VDD down to zero and back up to operational level will induce a power on reset and wake up the chip.
- b) Applying a "low" level on \overline{MCLR} pin
- c) A watchdog timer time-out (WDT must be enabled). "TO" status bit will be cleared in this case.
- d) The following interrupts can wake up the processor from SLEEP:
 1. External interrupt on RT pin
 2. External interrupt on INT pin
 3. Capture1 interrupt, due to a capture event on the RB0/CAP1 pin. The prescaler on the capture input will operate during SLEEP. The actual capture of the timer value will occur when execution resumes after wake-up (which is therefore, not meaningful).
 4. Capture2 interrupt.
 5. Input change on Port B interrupt
 6. Synchronous slave mode transmission interrupt: If synchronous transmission is in progress (using external clock) at the time the processor is put to SLEEP, a TBMT interrupt will be generated at the end of the transmission and wake the chip up.
 7. Synchronous slave mode reception interrupt: If synchronous reception is enabled ($CREN = 1$) before the chip goes into SLEEP, then RBFL interrupt will be set at the end of a reception (if a receive word came during SLEEP) which will wake the chip up.

If $GLINTD = 0$, the normal interrupt response takes place. If $GLINTD = 1$, the chip will resume execution starting with the instruction following the SLEEP instruction. It will not vector to interrupt service routine.

If selected oscillator type is XT or LP then the oscillator start-up timer (OST) is activated on wake-up. This will mean that the timer will keep the part in reset for $1024 t_{osc}$. The user needs to take this into account when considering interrupt response time coming out of SLEEP.

FIGURE 4.5.1: CPUSTA REGISTER

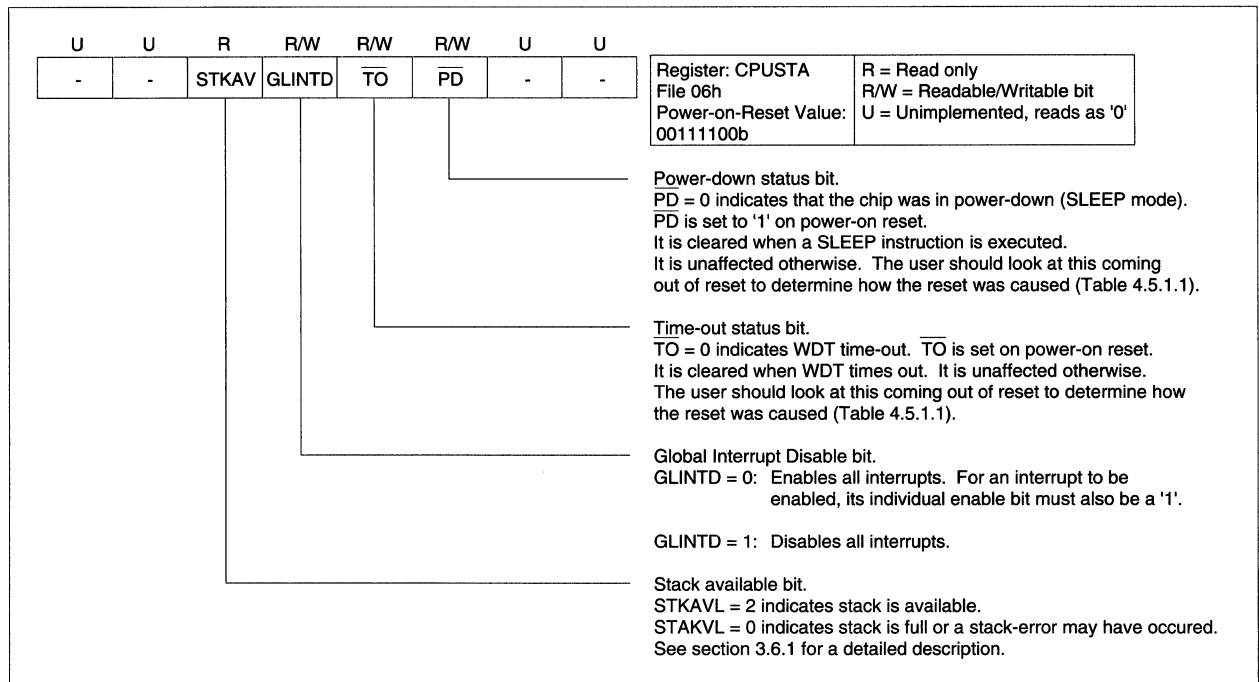


TABLE 4.5.1.1: WAKE -UP AND RESET FUNCTION TABLE

Event	Chip Status before event	Chip function after event					Notes
		PC	Oscillator Circuit	OST	TO	PD	
Power on reset	Don't care	0000	on	yes	1	1	
MCLR reset	Normal operation	0000	on	no	u	u	
SLEEP instruction	Normal operation	N+1	off	no	1	0	
MCLR wake-up	SLEEP	0000	on	yes(2)	u	u	
WDT time-out	Normal operation	0000	on	no	0	u	
WDT wake-up	SLEEP	0000	on	yes(2)	0	u	
Interrupt	Normal operation	Vector	on	no	u	u	
Interrupt wake-up	SLEEP, GLINTD=0	1. N+1 2. Vector	on	yes(2)	u	u	1
Interrupt wake-up	SLEEP, GLINTD=1	1. N+1 2. N+2	on	yes(2)	u	u	1

Legend

PC Program Counter contents after the event

TO Time Out status bit after the event

PD Power Down status bit after the event

N Address of SLEEP instruction

U No change takes place

Notes

Note 1: The instruction at "N+1" executed, after wake up. Step 2 depends on the status of the GLINTD bit at the time of the event. If GLINTD was "0", the program will vector to the interrupt routine.

Note 2: OST timer is activated only in XT and LP oscillator modes. (Sec. 4.4)

4.5.2 Interrupt/SLEEP Interaction

If an interrupt occurs during the very cycle a SLEEP instruction is fetched, it will be recognized in the following cycle (which is the execution cycle of the SLEEP instruction) preventing the processor from going into SLEEP. The SLEEP instruction will effectively execute as a single cycle NOP. The PD bit will not be cleared.

4.5.3 Minimizing Current Consumption in SLEEP Mode

The SLEEP mode is designed to reduce power consumption. To minimize current drawn during SLEEP mode, the user should turn-off output drivers that are sourcing or sinking current, if it is practical. Weak pull-ups on port-pins should be turned off, if possible. All inputs should be either at V_{SS} or at V_{DD} (or as close to rail as possible). An intermediate voltage on an input pin causes the input buffer to draw a significant amount of current.

4.6 WATCHDOG TIMER

The PIC17C42 has an on chip watchdog timer whose function is to recover from software malfunction. The watchdog timer is an 8 bit asynchronous ripple counter with an 8 bit prescaler (also an asynchronous ripple counter). The watchdog timer always runs off its own internal RC oscillator. The watchdog timer is not readable or writable. It is not mapped in data or program memory space. Two EPROM fuses provide four operating options for the watchdog timer:

FWDT1, FWDT0	WDT Clock Input Source	WDT Function Input Clock	WDT Period
1 0	RC osc	WDT runs with prescale = 256	4.6 sec
0 1	RC osc	WDT runs with prescale = 64	1.15 sec
1 1	RC osc	WDT runs with prescale = 1	18 ms
0 0	OSC/4	WDT runs as a regular timer with prescale = 256	65536 Tcy

Note: 0 implies a programmed fuse.

Fuses FWDT1 and FWDT0 are mapped in program memory locations FE03h and FE02h respectively. See section 4.8 for details on how to program fuses.

The watchdog timer and its prescaler are reset and the time-out bit, TO (bit3, CPUSTA) set to '1' if:

- A CLRWDT instruction is executed.
- A SLEEP instruction is executed.
- A power on reset occurs.

Under normal circumstances, the user program is expected to clear the watchdog timer on a regular interval. If the program fails to do so, the WDT will overflow and reset the chip. The watchdog timer and its prescaler are physically the same as the power-up timer (PWRT). They simply perform different roles in and outside reset condition.

4.6.1 WDT as a Regular Timer

Setting fuses FWDT1 and FWDT0 as 0's will configure the WDT as a simple timer. In this mode the timer increments on internal OSC/4 clock with a prescale of 256 (i.e. increments at OSC freq/1024 rate). On overflow TO bit is cleared, but the chip is not reset. In this mode the WDT is stopped during SLEEP. The TO bit is set when a CLRWDT instruction is executed.

4.7 CODE PROTECTION AND WRITE PROTECTION

The code in the user EPROM may be protected from piracy by selecting "code protected Microcontroller mode." This is done by blowing fuses FPMM1 and FPMM0 to "0". A TABLRD instruction, executed from the test EPROM attempting to read user EPROM will read encrypted data. However, if the instruction is executed from an address less than 2K (i.e. from user EPROM), it will read un-encrypted data.

Further, any TABLWT instruction executed from the test EPROM and attempting to write to the user EPROM, will

not result in programming of the destination. However, the instruction will still need to be terminated by an interrupt condition and the table latches will still be written. A TABLWT instruction, executed from an address less than 2K can program any user EPROM location regardless of code protection.

The above measures essentially prevent read, verify or programming of any user EPROM location from outside.

4.8 CONFIGURATION FUSES

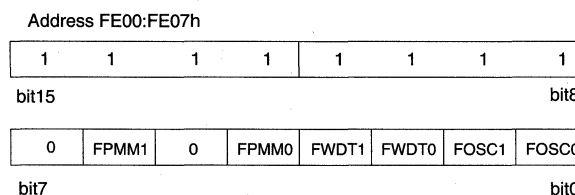
Configuration fuses are EPROM bits that can be programmed (reads '0') or left unprogrammed (reads '1') to select between options (e.g. operating modes). For simplicity of programming they are mapped into program memory. This also makes it possible to read the fuse values (only in microcontroller modes). Each fuse is assigned one program memory location. In erased condition a fuse will read as a '1'. To program (or "blow") a fuse, the user needs to write to the fuse address using a TABLWT instruction. Regardless of the data, a TABLWT to a fuse location will blow the fuse. The fuses and their addresses are shown in table 4.8.1.

Reading configuration fuses: Reading any fuse location in the address range FE00:FE07h will read all eight fuse values in the lower byte and all 1's in the upper byte. Fuse located at FE00h will show up in bit 0 and so on. The fuse locations are accessible only in microcontroller and secure microcontroller modes. In microprocessor and extended microcontroller modes, this section of the program memory is mapped external (see figure 1.5.2) making the fuse locations inaccessible.

TABLE 4.8.1: CONFIGURATION FUSES

Fuse	Address	Function
FOSC0 FOSC1	FE00h FE01h	FOSC1, FOSC0 : 00 : LP oscillator mode 01 : RC oscillator mode 10 : XT oscillator mode 11 : EC (external clock mode)
FWDT0 FWDT1	FE02h FE03h	FWDT1, FWDT0 : 10 : WDT prescale is 256 01 : WDT prescale is 64 11 : WDT prescale is 1 00 : WDT is a normal timer
FPMM0 FPMM1	FE04h FE06h	FPMM1, FPMM0 : 00 : Microcontroller mode (code protected) 10 : Microcontroller mode 01 : Extended microcontroller mode 11 : Microprocessor mode

FIGURE 4.8.1: READING FUSE LOCATION



5.0 OVERVIEW OF PERIPHERALS

An array of sophisticated, high speed peripherals are incorporated on chip to meet the demands of real time applications. All peripherals are highly intelligent and have their own interrupts and error handling to free up the CPU as much as possible. There are three 16 bit timer/counters one of which can be split into two eight bit timers creating up to four timer/counter resources. Two high speed captures are provided for efficient interface to shaft encoders and other high speed pulse train sources. Two high speed pulse-width-modulation (PWM) outputs with up to 10 bit resolution make it possible to control a motor through power drivers. There are two external and several internal interrupt sources. The capture pins can be used as interrupt pins making it possible to have up to four external interrupts. Finally, there are 33 I/O pins most of which can be configured as inputs or outputs in software. A number of the I/O pins are multiplexed with peripheral functions or the system bus. In microcontroller mode 23 I/O pin are un-multiplexed.

5.1 THE BANK SELECT REGISTER (BSR, ADDRESS 0Fh)

All the peripheral registers are mapped into the data memory space. In order to accommodate the large number of registers in the 256 byte data memory space without taking away from general purpose data RAM, a banking scheme has been used. A segment of the data memory, from address 10h to address 17h, is banked. A bank select register (BSR, address 0Fh) selects the currently active "peripheral bank". Effort has been made to group the peripheral registers of related functionality in one bank. However, it will still be necessary to switch from bank to bank in order to address all peripherals related to a single task. To alleviate this problem, a single cycle instruction, MOVLB (move literal value to BSR) is incorporated in the instruction set. In the PIC 17C42 only the low four bits of the BSR are physically implemented, making it possible to address up to sixteen banks. Only four banks are actually used (see data memory map in figure 1.6.1).

6.0 DIGITAL I/O PORTS

The PIC17C42 has five ports A, B, C, D and E. Together these add up to 33 port pins. Most port pins have an associated data direction bit which configures it as input (DDR bit='1') or output (DDR bit='0'). When a port pin is read as an input, the value on the pin (and not the data latch) is read.

Most port pins are multiplexed with the system bus or peripheral functions. These pins are configured as port pins or peripheral inputs/outputs by control bits in corresponding peripheral registers. Once a port pin is selected for an alternate function, its direction will be determined by the peripheral logic which will force the DDR bit to the required state.

Ports A, B, C, D and E and their associated DDR registers are mapped into the data-memory. Ports C, D and E multiplex with the system bus (AD <15:0>, ALE, WR and OE).

6.1 PORT A

File 10h in Bank 0 is PORTA, a 6 bit port. There is no Data Direction Register associated with this port. Port A is multiplexed with peripheral functions as described in table 6.1.1. See figure 6.1.1 for block diagram of Port A and 6.0.1 for read/write timing.

FIGURE 6.0.1: I/O PORT READ AND WRITE TIMING

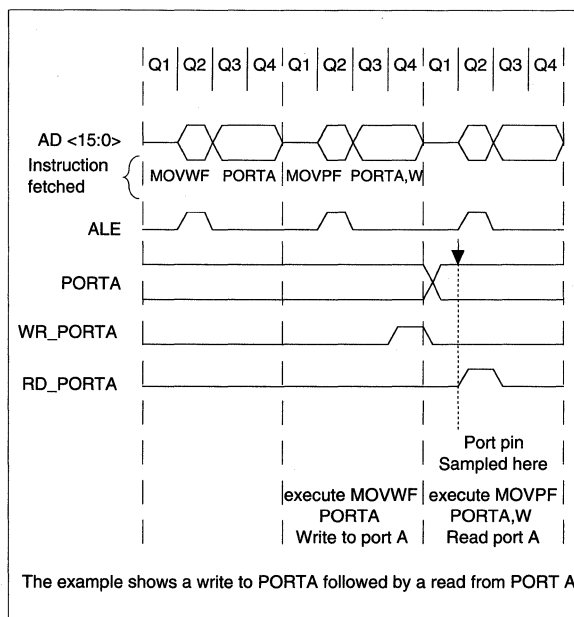


TABLE 6.1.1 : PORT A FUNCTIONS

Port Pin	Bit	Pin function	Alternate function
RA0/INT	bit 0	Input only (Schmitt Trigger) port pin	INT external interrupt input
RA1/RT	bit 1	Input only (Schmitt Trigger) port pin	RT external interrupt input. It is also the external clock input for the RTCC timer/counter.
RA2, RA3	bit 2,3	Input/output pins with Schmitt Trigger input and open-drain output. To use either of these two pins as an input, the user must write a '1' to the port data latch. If used as an output, external pull-up resistor must be provided. These pins can be pulled up to voltages higher than Vcc. Also, these two port pins provide higher current sink capability (See DC specs for details).	None
RA4/RX/DT	bit 4	Input only (Schmitt Trigger) port pin	If the SPEN bit (bit 7, RCSTA) is a '1' then this pin is configured by the serial port. In SYNC mode: It is data input or output (DT) In ASYNC mode: It is receive data input (RX).
RA5/TX/CK	bit 5	Input only (Schmitt Trigger) port pin	If the SPEN (bit 7, RCSTA) bit is a '1' then this pin is controlled by the serial port. In SYNC mode: It is either clock input (slave mode) or the clock output (master mode) in ASYNC mode: It is the transmit data output (TX).
	bit 6	This bit is unimplemented and reads as '0'.	
	bit 7	No pin associated	This is a control bit ($\overline{\text{PUEB}}$) for Port B. No port pin is associated with this bit. PUEB=0 enables weak pull-ups on Port B.

6.1.1 Using RA2, RA3 Pins as Output

PortA does not have an associated data direction register. When using them as outputs, read-modify-write instructions (such as BCF, BSF, BTF) are not recommended on PortA, since a read will read the port pins but a write will write to the port data latch. Such an operation may inadvertently cause RA2, RA3 to switch from output to input or vice-versa.

FIGURE 6.1.1: PORT A BLOCK DIAGRAMS

FIGURE 6.1.1A

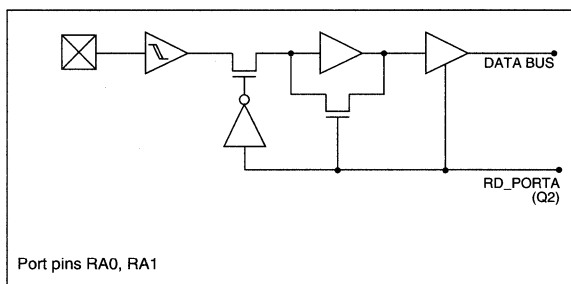


FIGURE 6.1.1B

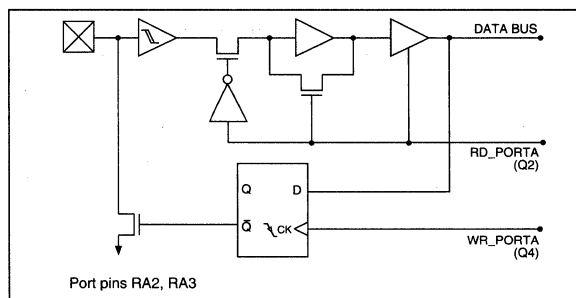
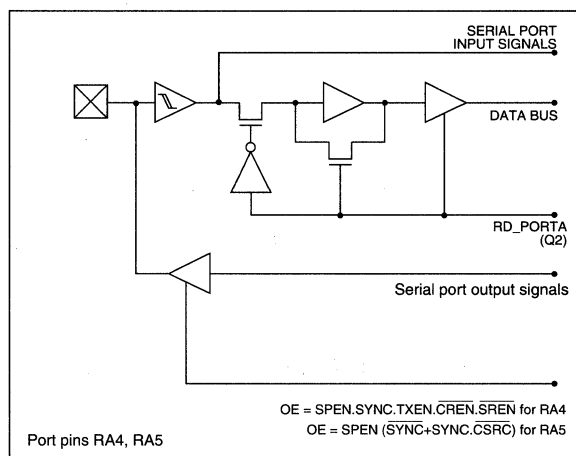


FIGURE 6.1.1C



6.1.2 SUMMARY OF PORT A REGISTERS

Register Name	Function	Address	Reset Value
PORTA	Port A pins when read, Port A latch when written (RA2/RA3 only)	Bank 0, File10h	00XXXXXb
RTCSTA	RTCC status/control register (configures RA0/INT & RA1/RT pins)	File 05h	0000000b
RCSTA	Serial port receive status/control register (configures RA4/RX/DT and RA5/TX/CK pins)	Bank0, File 13h	000000Xb

6.2 PORT B

Port B is an eight bit wide bidirectional port. It is mapped in Bank0, File 12h. Writing to this address writes to the port latch while reading it will read the port pins. An eight bit data direction register (DDRB, Bank 0, File 11h) configures each port pin as an input or output. A '0' in the 'DDR' register configures the port as an output. Each port pin also has a software configurable weak pull-up (~100 µA typical). A control bit PUEB (bit 7, Bank 0, File 10h, Register PORTA) can enable (PUEB = '0') or disable (PUEB = '1') the pull-ups. The weak pull-up is turned off for any pin configured as output.

Most of the pins of Port B are multiplexed with peripheral functions. Table 6.2.1 describes their alternate functions. When a pin is redefined to be a port pin from a peripheral pin, its data direction bit may be left in an unknown state. The user will need to re-initialize the DDR bit properly. See figures 6.2.1 and 6.2.2 for block diagrams of port B and figure 6.0.1 for read/write timing.

Port B also has an "interrupt on change" feature. When configured as input, its output data latch can be used as a compare latch. An active high output is generated on mismatch between the pin and the latch. The "mismatch" outputs of all the input pins are OR-ed together to generate the IRB interrupt. All the output pins are excluded from the comparison. Thus, an interrupt is generated when the port input changes. This interrupt can wake the chip up from SLEEP mode.

The interrupt is latched in the IRB bit (bit 7, Register PIR, Bank1, File 16h). IRB is readable and writable by the CPU. The user, in the interrupt service routine, can clear the interrupt in one of two ways:

- Disable the interrupt by clearing the corresponding interrupt enable bit, IEB.
- Read PortB and write back the pin value to the data latch. This will end mismatch condition and therefore the mismatch output. Next, the user must clear bit IRB.

TABLE 6.2.1: PORT B FUNCTIONS

Port Pin	Bit	Pin Function	Alternate Function
RB0/CAP1	bit 0	Input/Output port pin with Schmitt Trigger input	CAP1: Capture1 input
RB1/CAP2	bit 1	Input/Output port pin with Schmitt Trigger input	CAP2: Capture2 input
RB2/PWM1	bit 2	Input/Output port pin with Schmitt Trigger input	PWM1 output. This pin is configured as the PWM1 output if control bit PWM1ON (bit 4, Register TCON2, Bank 3, File 17h) is set to '1'.
RB3/PWM2	bit 3	Input/Output port pin with Schmitt Trigger input	PWM2 output. This pin is configured as the PWM2 output if the control bit PWM2ON (bit 5, Register TCON2, Bank 3, File 17h) is '1'
RB4/TCLK12	bit 4	Input/Output port pin with Schmitt Trigger input	TCLK12: external clock input for timer1 and timer2
RB5/TCLK3	bit 5	Input/Output port pin with Schmitt Trigger input	TCLK3: external clock input for timer3
RB6	bit 6	Input/Output port pin with Schmitt Trigger input	
RB7	bit 7	Input/Output port pin with Schmitt Trigger input	

FIGURE 6.2.1: PORT B BLOCK DIAGRAM

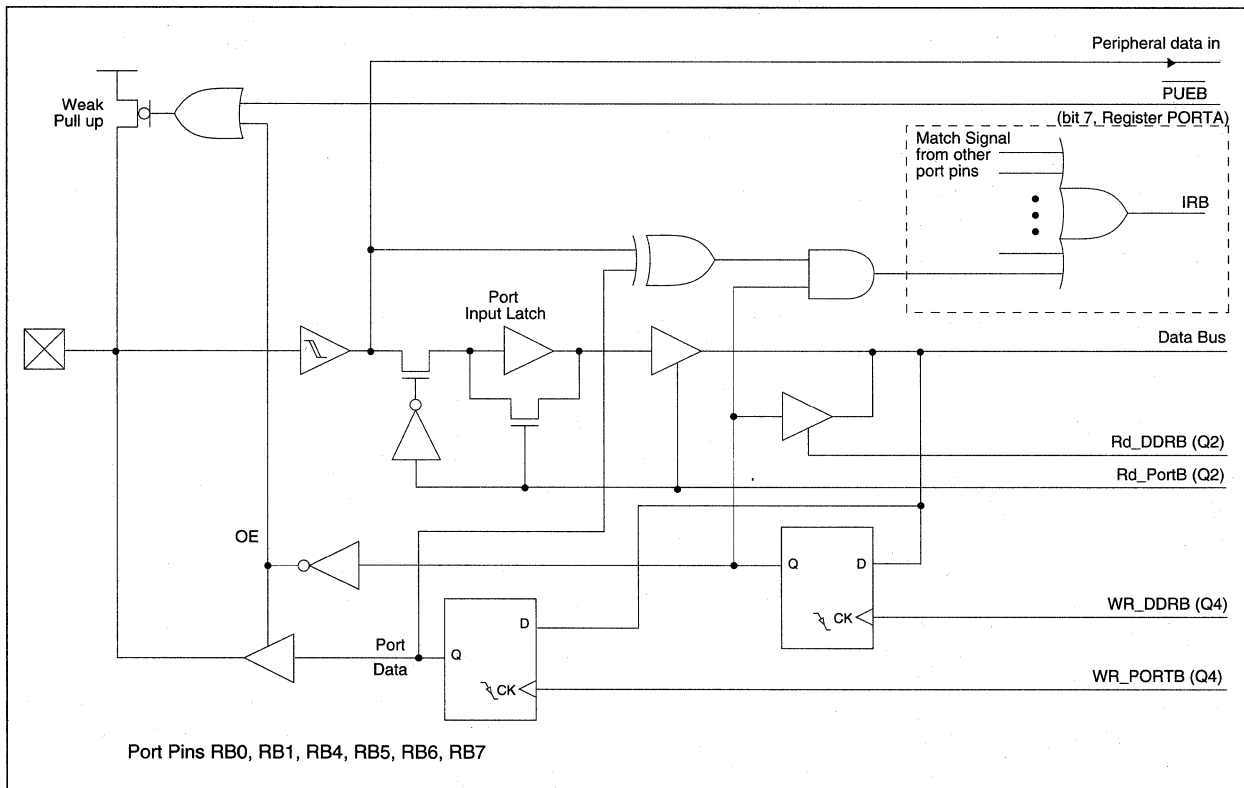
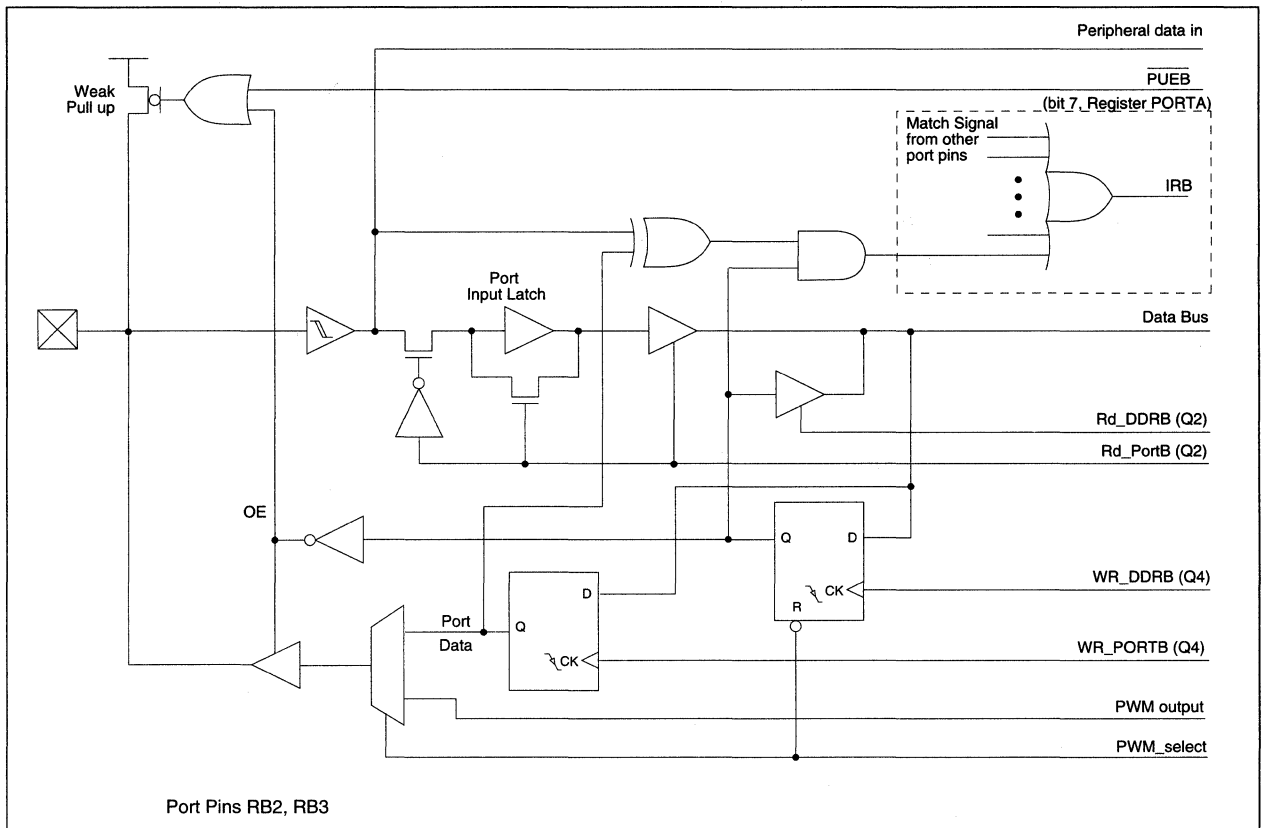


FIGURE 6.2.2: PORT B BLOCK DIAGRAM



6.2.1 SUMMARY OF PORT B REGISTERS

Register Name	Function	Address	Reset Value
PORTB	Port B pins when read Port B latch when written	Bank 0, File 12h	XXXXXXXXb
DDRB	Port B data direction register	Bank 0, File 11h	11111111b
PORTA (bit PUEB)	Port A data/ PUEB bit	Bank 0, File 10h	00XXXXXXXXb
PIR (bit IRB)	Peripheral interrupt register	Bank 1, File 16h	00000010b
PIE (bit IEB)	Peripheral interrupt enable register	Bank 1, File 17h	00000000b
INTSTA (bit PEIE)	Interrupt status register	File 07h	00000000b
CPUSTA (bit GLINTD)	CPU status register	File 06h	0011XX00b
TCON2	Timer/PWM/capture control registers	Bank 3, File 17h	00000000b

6.3 PORT C

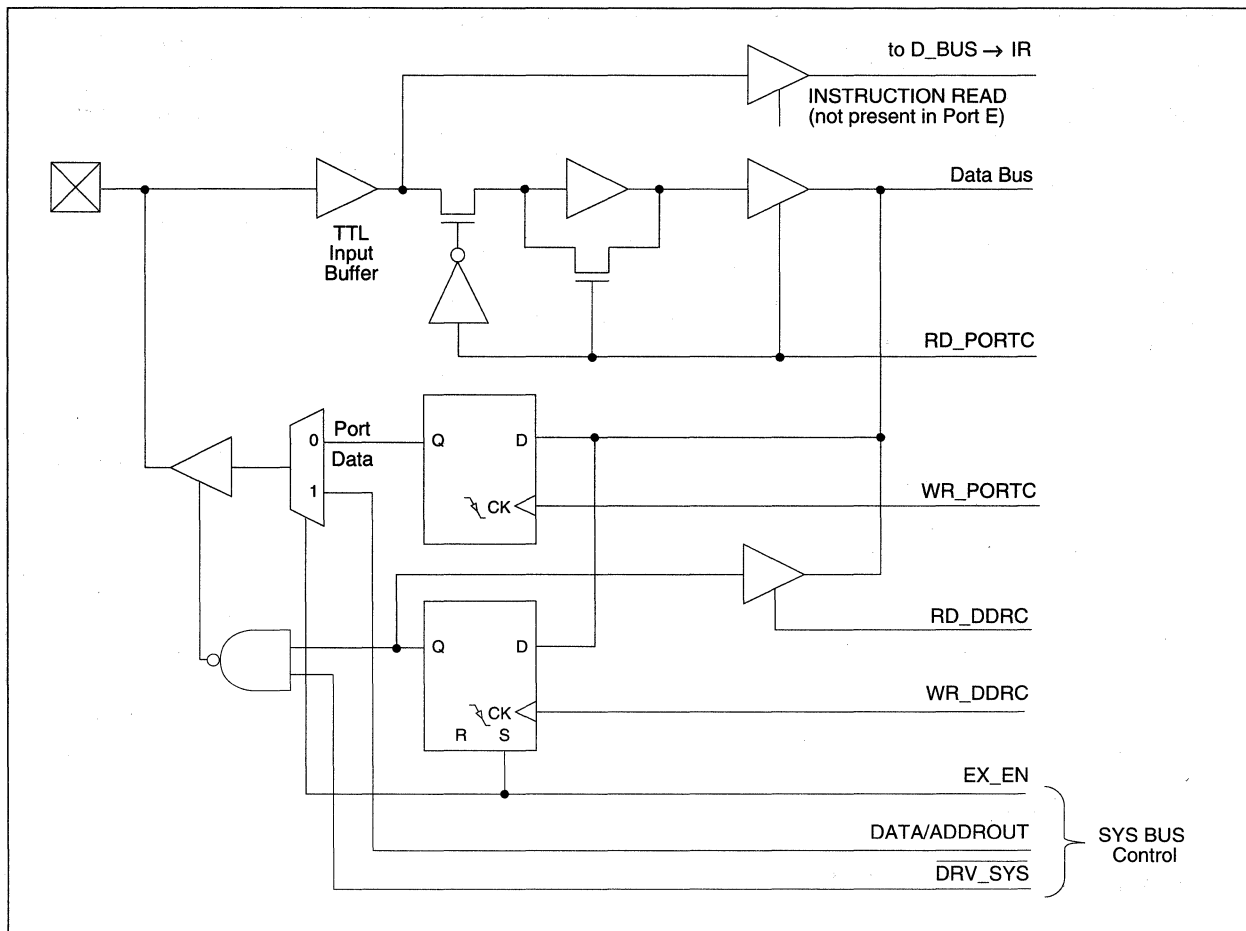
Port C is an eight bit wide bidirection port mapped in File 11h, Bank 1. The corresponding data direction register DDRC (file 10h, Bank 1) can configure each pin as an

input (if DDRC bit is '1') or output (if DDRC bit is '0'). This port is multiplexed with AD<7:0>, the lower byte of the Address/Data bus. Bit 0 of Port C is AD<0>. See figure 6.3.1 for block diagram of Port C and figure 6.0.1 for read/write timing.

6.3.1 SUMMARY OF PORT C REGISTERS

Register Name	Function	Address	Reset Value
PORTC	Port C pins when read Port C latch when written	Bank 1, File 11h	XXXXXXXXb
DDRC	Port C data direction register	Bank 1, File 10h	11111111b
INTSTA (bit PEIE)	Interrupt status register	File 07h	00000000b
CPUSTA (bit GLINTD)	CPU status register	File 06h	0011XX00b

FIGURE 6.3.1: BLOCK DIAGRAM OF PORTS C, D AND E



6.4 PORT D

Port D is an eight bit wide bidirection port mapped in File 13h, Bank 1. The corresponding data direction register DDRD (file 12h, Bank 1) can configure each pin as an input (if DDRD bit is '1') or output (if DDRD bit is '0'). This port is multiplexed with AD<15.8>, the higher byte of the Address/Data bus. Bit 0 of Port D is AD<8>. See figure 6.3.1 for block diagram of Port C and figure 6.0.1 for read/write timing.

6.5 PORT E

PORTE is a 3 bit wide bidirectional port mapped in data memory (file 15h, Bank1). The corresponding Data Direction Register, DDRE, is mapped at file 14h, Bank 1. Each port pin can be configured as an input (DDRE bit = '1') or an output (DDRE bit = '0'). Only the three lowest significant bits are physically implemented in 17C42. The unimplemented bits read as '0'. See Figure 6.3.1 for block diagram of Port E and Figure 6.0.1 for read/write timing. Port E is multiplexed with control outputs ALE, WR and OE in external execution mode.

FIGURE 6.4.1: SUMMARY OF PORT D REGISTERS

Register Name	Function	Address	Reset Value
PORTD	Port D pins when read Port D latch when written	Bank 1, File 13h	xxxxxxxxb
DDRD	Port D data direction register	Bank 1, File 12h	11111111b

6.5.1 SUMMARY OF PORT E REGISTERS

Register Name	Function	Address	Reset Value
PORT E	Port E pins when read Port E latch when written	Bank 1, File15h	00000xxx _b
DDRE	Port E data direction register	Bank 1, File14h	00000111 _b

TABLE 6.5.1 PORT E FUNCTIONS

Port Pin	Bit	Pin Function	System Bus Function (External execution)
RE0/ALE	bit 0	Input/output port. TTL input buffer.	ALE output
RE1/ÖE	bit 1	Input/output port. TTL input buffer.	ÖE output
RE2/WR	bit 2	Input/output port. TTL input buffer.	WR output

7.0 UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The serial port can operate either a full-duplex asynchronous mode or in a half-duplex clocked synchronous mode. Synchronous mode uses a bi-directional data pin and a bi-directional clock pin. In synchronous mode, the clock can be either internal (master mode) or external (slave mode). In asynchronous mode, the clock is always derived internally. A dedicated 8 bit Baud Rate Generator (BRG) is used for internal clock generation. In both modes, receiver and transmitter are double buffered, 8 or 9 data bits are supported and separate transmit and receive interrupts are available.

7.1 ASYNCHRONOUS MODE

The asynchronous mode is selected by setting the SYNC bit to '0' in the TXSTA register. Furthermore, SPEN bit (Serial Port Enable, bit 7, Register RCSTA, Bank 0) has to be set to enable RA4 and RA5 as serial port pins. SPEN=0 will configure these pins as port pins. In asynchronous mode the RX pin receives data and the TX pin transmits data in a full duplex mode. Data is transmitted and received least significant bit first. Both receive and transmit operate on the same internally generated clock which is derived from the Baud Rate Generator (Register SPBRG, Bank 0, File 17h). Data on the RX pin is sampled on the 7th, 8th and 9th pulses of a 16X (16 times the baud clock) internal clock. A majority of these three bits decide whether a one or a zero was received. In addition to the 8 or 9 data bits, one start bit and one stop bit are sent. Parity is not supported directly in hardware, but can easily be implemented in software. Asynchronous mode operation is stopped during SLEEP.

7.1.1 Asynchronous Mode Transmission

Once asynchronous mode is selected (SYNC=0, bit 4 Register TXSTA) and serial port outputs are enabled (SPEN=1, bit 7, Register RCSTA) transmission can be enabled by setting TXEN bit to '1' (bit 5, TXSTA register). Actual transmission will begin when a word is written to the transmit buffer register (TXREG, bank0, file 16h) and the Baud Rate Generator produces a shift clock (figure 7.1.1.1). A start bit is sent out first (logic '0'), followed by 8 or 9 data bits and a stop bit (logic '1'). Transmitted data appears on RA5/TX/CK pin. Transmission can also be started by first writing a word to the TXREG and then setting TXEN to '1'.

The transmit register (TXREG) is double buffered. As the user writes to TXREG, the data is transferred from the buffer to the transmit shift register (TSR), thus freeing up the buffer register. An interrupt is pending as long as TXREG is empty. Indicating that the transmit buffer register (TXREG) is free to accept another word. This interrupt request is bit 1 (TBMT) of PIR (peripheral interrupt request register; Bank 1, file 16h) register. This interrupt can be enabled or disabled by bit 1 (TXIE) of PIE (peripheral interrupt enable; Bank 1, file 17h) register. TXIE=1 enables the interrupt. Regardless of TXIE, the TBMT bit will always show the status of the TXREG buffer (can not be affected in software) and can be used as a status bit. The interrupt request bit (TBMT) is read only. Therefore, to avoid unwanted interrupts (say, at the end of a transmission) the user will need to mask off this interrupt.

In addition to TXIE bit, two other bits will affect the transmit interrupt. They are: PEIE (bit3, INTSTA register, file 07h) that enables (if='1') or disables (if='0') all peripheral interrupts, and GLINTD (Global Interrupt Disable, bit 4, CPUTA register, file 06h) bit that disables all interrupts if set to '1'.

While TBMT (Transmit Buffer Empty) indicates the status of the transmit buffer register, another bit TRMT (bit1, register TXSTA) indicates the status of the transmit shift register. It is a read only bit. TRMT=1 implies transmit shift register is empty. The user can determine exactly when transmission is completed by polling this bit. TRMT is set after stop bit is sent out.

CREN or SREN bits do not affect asynchronous transmission. Clearing TXEN during transmission aborts transmission, reverts TX pin to hi-impedance and resets the transmitter.

FIGURE 7.1.1.1: ASYNCHRONOUS TRANSMISSION

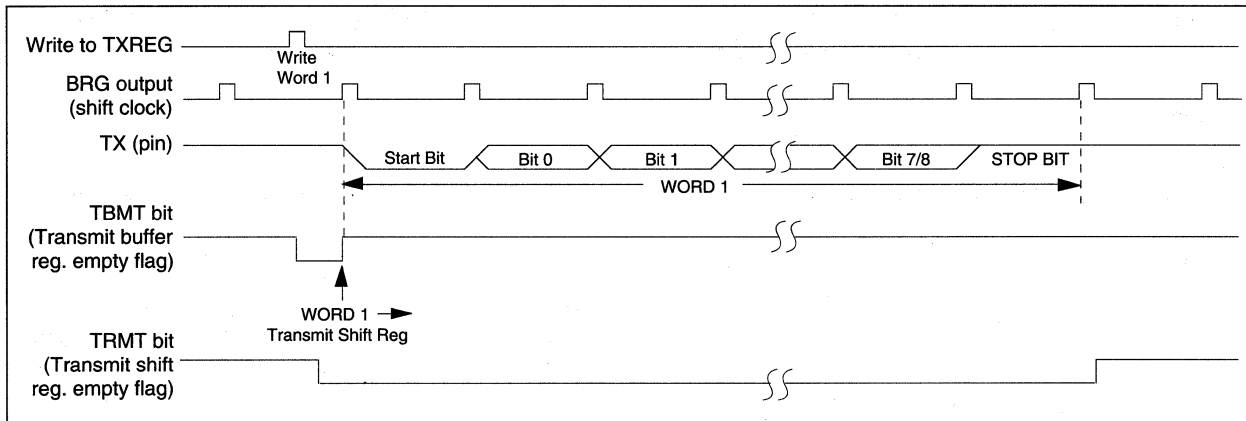
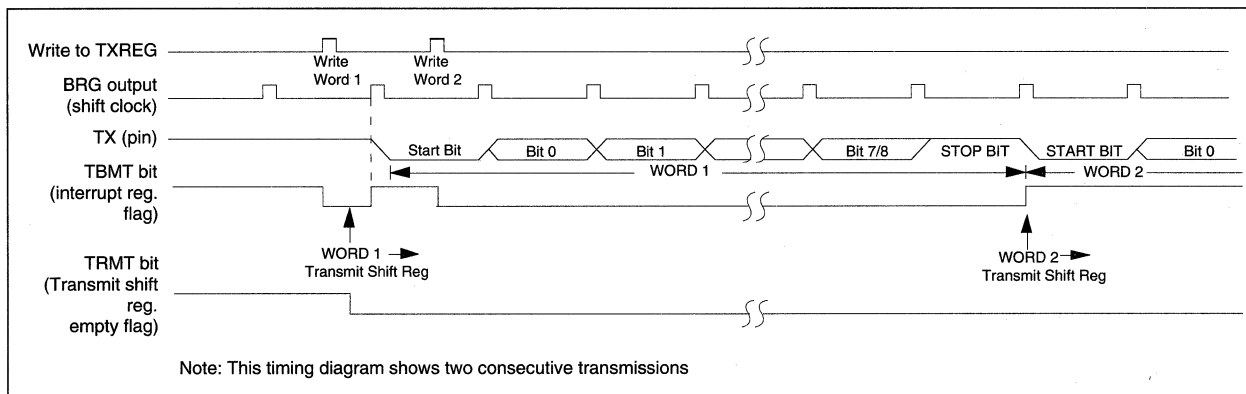


FIGURE 7.1.1.2: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)



If 9 bit transmission is selected (TX8/9=1, bit 6, register TXSTA) the 9th bit should be written to TXD8 (bit0, TXSTA). This bit is double buffered as well. The 9th bit must be written before writing the data word to TXREG, since the latter triggers the transfer of the entire word to the transmit shift register.

7.1.2 Asynchronous Mode Reception

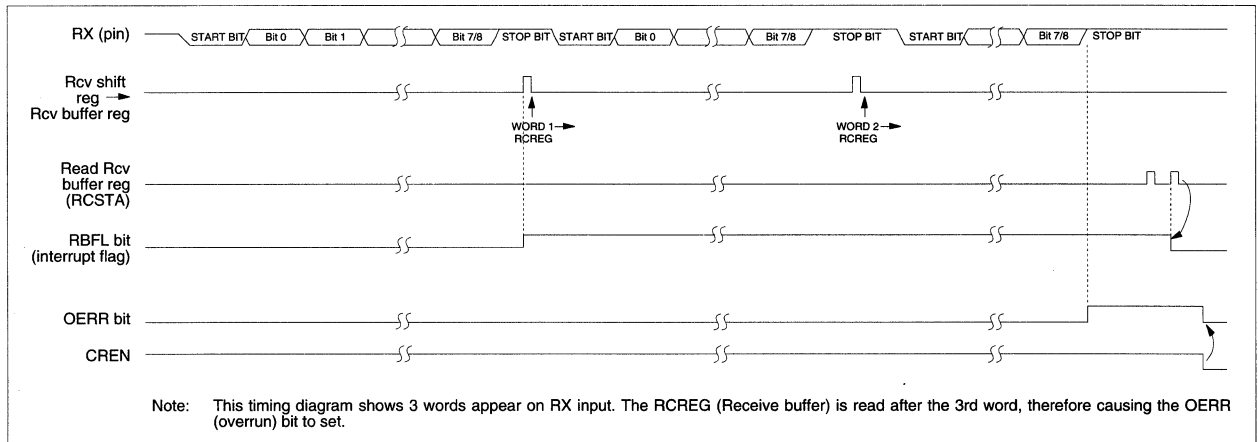
Data is received on RA4/RX/DT pin. Reception is enabled by setting the CREN bit (bit4, register RCSTA) to '1'. The SREN bit (bit5, RCSTA) has no function in asynchronous mode. Reception begins when a start-bit is detected on RX pin. The Baud Rate Generator internally generates a 16x clock. Every incoming bit is sampled on the 7th, 8th and the 9th time slot and a majority detection is done to determine the value of the bit. After sampling the stop bit (i.e. halfway through stop bit), the received data is transferred to the receive buffer register (RCREG) if the buffer register is empty. The RCREG is actually a two word deep FIFO. Therefore, it is possible to receive two words, transfer them to RCREG and begin receiving the 3rd word in the receive shift register (RSR). If at the time of reception of the last bit of the 3rd word, the RCREG has still not been read (and therefore is holding two words) then the receiver control logic will set the overrun error bit, OERR (bit1, register RCREG). In case of overrun, the word in the shift register

is lost (i.e. it can not be read). The RCREG can be read twice to retrieve the first two words. The user will need to clear OERR by resetting the receiver (by clearing CREN). Clearing OERR is essential since once the overrun flag is set, the receiver simply stops transferring RSR to RCREG.

The framing error bit, FERR (bit 2, Register RCSTA) and the 9th receive bit, RCD8 (bit 0, RCSTA) are buffered the same way as the receive data. Reading RCREG will load the RCD8 and FERR bits with new values. The user, therefore, must read the RCSTA register before reading the received data (RCREG) in order to obtain FERR and 9th data bit information. If the RCREG is read first, then the status register RCSTA will be loaded with new status information and the old information will be lost. The framing error bit, FERR, is set if the stop bit is detected to be a '0'.

A receive interrupt flag RBFL, is set (bit0, register PIR) when the receive shift register content is shifted to the receive buffer register. This interrupt can be enabled or disabled via the RCIE (Receive interrupt enable) bit (bit0, register PIE). RCIE=1 will enable the interrupt. The RBFL (receive buffer full interrupt flag) bit is a read only bit and is cleared when the receive buffer is read. However, if the receive shift register is full, it will transfer its contents to the receive buffer register and the RBFL

FIGURE 7.1.2.1: ASYNCHRONOUS RECEPTION



bit will be set again. To enable receive interrupt, the Peripheral Interrupt Enable bit, PEIE (bit3, INTSTA register, file 07h). must be set and the Global Interrupt Disable bit, GLINTD (bit4, CPUSTA register, file 06h), must be cleared.

7.2 SYNCHRONOUS MODE

The synchronous mode is selected by setting the SYNC bit (bit4, TXSTA register) to a '1'. In addition, the SPEN bit (bit7, RCSTA register) must be set to a '1' to configure the RA5/TX/CK and RA4/RX/DT pins as CK (synchronous clock) and DT (sync data) pins respectively. Synchronous mode is half duplex with the DT pin as data input during reception and data output during transmission. The CK pin is clock output if internal clock option (master mode) is selected by setting the CSRC (bit7, TXSTA register) bit to a '1'. If CSRC='0' then the CK pin is clock input (synchronous slave mode).

As in asynchronous mode, 8 or 9 data bits are transmitted or received. No start or stop bits are sent or received.

7.2.1 Synchronous Mode Transmission

Once the sync mode is selected (SYNC='1') and the serial port is enabled (SPEN='1', register RCSTA), transmission is enabled by setting the TXEN (transmit enable, bit5, TXSTA register) bit to a '1'. This will configure the TX pin as an output. Actual transmission will begin when a word is written to the transmit buffer register (TXREG). The transmitter is double buffered. If the transmit shift register (TSR) is empty then the word will be transferred from TXREG to TSR. The first data bit will be shifted out at the next available rising edge of the clock. Data out is stable around the falling edge of the sync clock. Transmission can also be started by first writing a data word to TXREG and then setting TXEN='1'. This method may be advantageous when slow baud rates are selected, since the Baud Rate Generator is kept under reset when TXEN=CREN=SREN=0. Setting the TXEN bit will start the BRG, creating a shift clock immediately.

The TBMT interrupt (bit1, PIR register) is pending whenever the transmit buffer is empty and ready to accept another word. The interrupt has a corresponding mask bit (TXIE, bit1, Register PIE). TXIE='1' enables the transmit interrupt while TXIE='0' disables it. Regardless of TXIE, TBMT will always show the status of the TXREG (not affected by software) and can be used as a status bit. To enable the transmit interrupt, Peripheral Interrupt Enable, PEIE (bit3, INTSTA register, file 07h) bit must be set and Global Interrupt Disable, GLINTD (bit4, CPUSTA register, file 06h) bit must be cleared.

While TBMT (Transmit Buffer Empty) indicates the status of the transmit buffer register, another bit TRMT (bit1, register TXSTA), indicates the status of the transmit shift register. It is a read only status bit. TRMT=1 implies that the transmit shift register is empty. The user can determine exactly when transmission is over by polling this bit. TRMT is set after the last bit is sent out.

If 9 bit transmission is selected, the 9th bit should be written to bit TXD8 (bit0, TXSTA). This bit is also double buffered. The 9th bit must be written prior to writing the data word to TXREG, since a write to the TXREG triggers the transfer of the entire word to the transmit shift register.

In sync master mode, the CK pin will output clocks only during actual transmission (figure 7.2.1.1). In sync slave mode clock input may be present on the pin at all times.

If TXEN is cleared during transmission of a word, transmission will be aborted and the DT and CK pins will revert to hi-impedance. If either the CREN or the SREN bit is set to a '1', transmission is also aborted and the DT pin will go into hi-impedance state (for reception). The CK pin will remain an output if CSRC=1 (internal clock). The transmitter logic, although disconnected from the pins, is not reset. The user must clear the TXEN bit to reset the transmitter. This is particularly important if the SREN was set to a '1' to interrupt an ongoing transmission. In this case, after reception of a single word, the SREN bit will reset and the serial port will revert back to transmit mode (since TXEN is still set). This means the DT pin will turn around and start driving. To avoid this, TXEN should be cleared.

FIGURE 7.2.1.1: SYNCHRONOUS TRANSMISSION

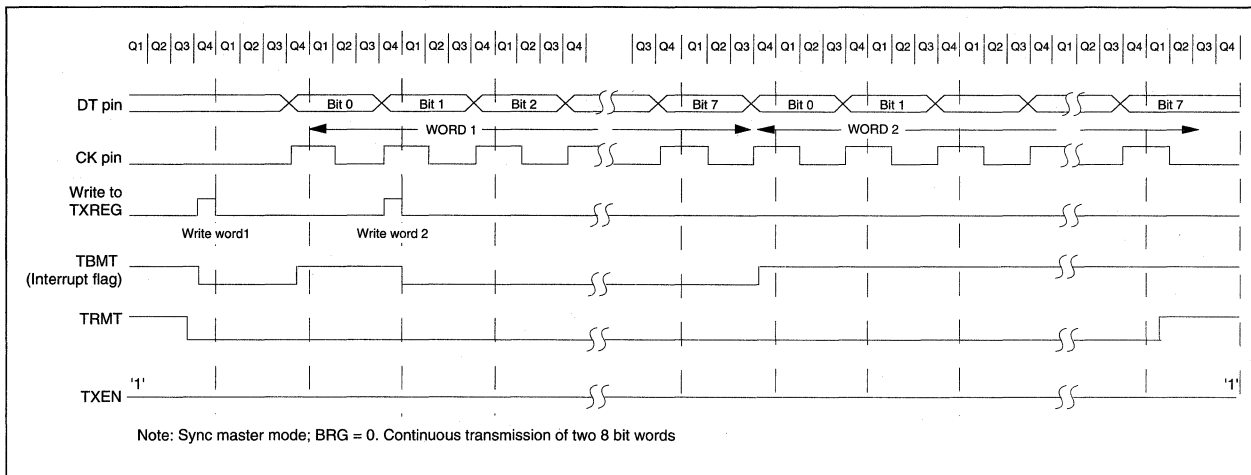


FIGURE 7.2.1.2: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

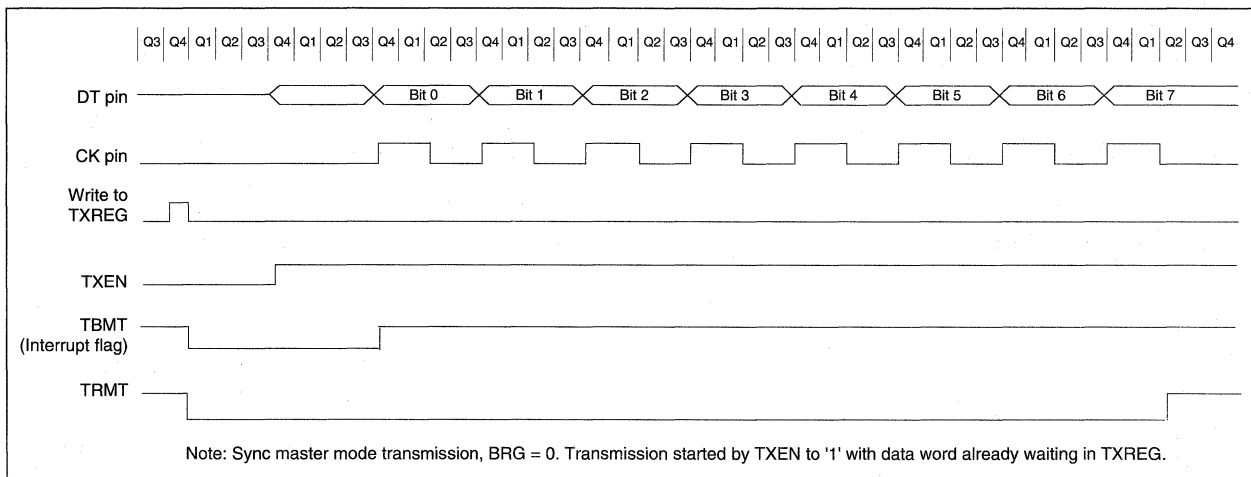
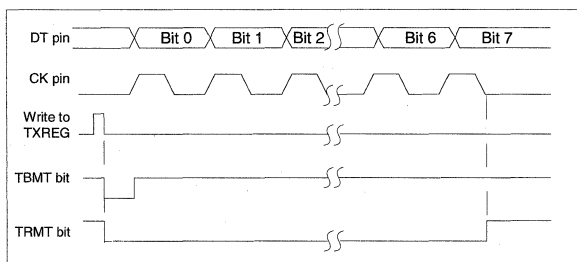


FIGURE 7.2.1.3: SYNCHRONOUS TRANSMISSION (SLAVE)



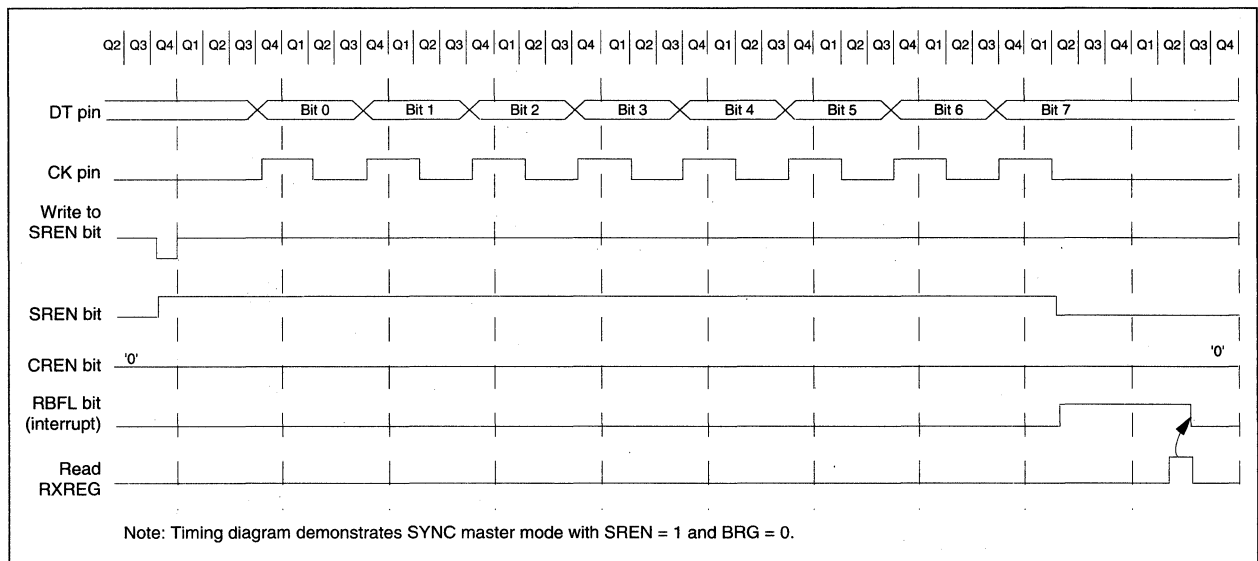
7.2.2 Synchronous Mode Reception

Data is sampled on the DT pin on the falling edge of the clock. Reception is enabled by either setting the SREN bit (Single Receive Enable, bit5, RCSTA register) or the CREN bit (Continuous Receive Enable, bit4, RCSTA register). If SREN is set, one word is received after which SREN is reset in hardware. If the CREN bit is set, words are received continuously (and read off by the CPU presumably) until CREN is reset by software. If both CREN and SREN are set, then CREN will take precedence.

After a word is received completely, it is transferred from the receive shift register (RSR) to the receive buffer register (RCREG) thus freeing up the RSR to receive the next word. With CREN=1, it is possible to receive consecutive data words without any discontinuity in between. This makes it possible to receive data words of larger size, e.g. 16 bit. In synchronous slave mode the SREN bit is a don't care.

The RCREG is actually a two word deep FIFO. Therefore, it is possible to receive two words, transfer them to RCREG and begin receiving the 3rd word in the receive shift register (RSR). If, at the time of reception of the last bit of the 3rd word, the RCREG has still not been read (and therefore is holding two words) then the receiver control logic will set the overrun error bit, OERR (bit1, register RCSTA). In case of an overrun, the word in the shift register is lost (i.e. it can not be read). The RCREG can be read twice to retrieve the first two words. The user will need to clear OERR by resetting the receiver (by clearing CREN). Clearing OERR is essential since once overflow flag is set the receiver simply stops transferring RSR to RCREG.

FIGURE 7.2.2.1: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



An interrupt is issued when RSR transfers a data word to receive buffer register, RCREG, indicating that RCREG is full. The interrupt flag (RBFL, bit0, register PIR) can be masked by interrupt mask bit RCIE (Receive interrupt enable, bit0, register PIE). RCIE=1 enables the receive interrupt.

The 9th bit of the received word is loaded into RCD8 (bit0, RCSTA). This bit is buffered the same way as the receive data. Reading the RCREG register will load the new 9th bit. Therefore, the user must read the RCSTA register before reading the received data word from RCREG.

7.2.3 Synchronous Slave Mode/SLEEP Mode Interaction:

When the part is put into SLEEP mode, all on chip phase clocks are stopped (part is held in Q1 state; see SLEEP section for details). In SLEEP, synchronous slave mode operation is possible because this mode uses external clock.

SLEEP/sync slave receive: If receive is enabled (SREN = '1') prior to invoking SLEEP mode, then a word may be received during SLEEP and at the completion of such reception the RSR will be transferred to RCREG (assuming it is empty). Simultaneously a receive interrupt will be generated which will wake the chip up, provided this interrupt was enabled (by setting RCIE = PEIE = '1'). If GLINTD = '0', then additionally the interrupt will be responded to by jumping to interrupt vector 0020h. If the receive interrupt is disabled, prior to invoking SLEEP mode, then words are received during SLEEP without waking up the processor. Overflow bit will be set if three words are received.

SLEEP/sync slave transmit: If two words are written to TXREG and then the chip is put into SLEEP the following sequence of events will occur. The first word will immediately transfer to the TSR. The second word will remain in TXREG. Transmit interrupt (TBMT) will stay inactive (low). As the first word is shifted out, the second word will transfer from TXREG to TSR and the transmit interrupt (TBMT) will be raised again. This will wake up the chip provided the interrupt was not masked (i.e. TXIE = PEIE = '1'). If GLINTD = 0, then branch to interrupt vector 0020h will take place as well.

7.3 BAUD RATE GENERATOR

The serial port is equipped with a dedicated 8 bit Baud Rate Generator (SPBRG, bank0, file 17h). The SPBRG register is readable and writable. The SPBRG register controls the period of a free running 8 bit timer. In synchronous mode the baud rate is $f_{osc}/4(x+1)$ where f_{osc} = oscillator or clock-in frequency and x = value written to SPBRG register. In asynchronous mode the baud rate is $f_{osc}/64(x+1)$. Tables 7.3.1 and 7.3.2 show baud rate values for different SPBRG value and clock-in frequency. SPBRG is unknown following power-on reset.

Writing a value to the SPBRG clears the timer. This guarantees that the timer does not go through an overflow cycle, before outputting the appropriate baud rate.

FIGURE 7.2.2.2: SYNCHRONOUS MASTER MODE RECEPTION, CREN

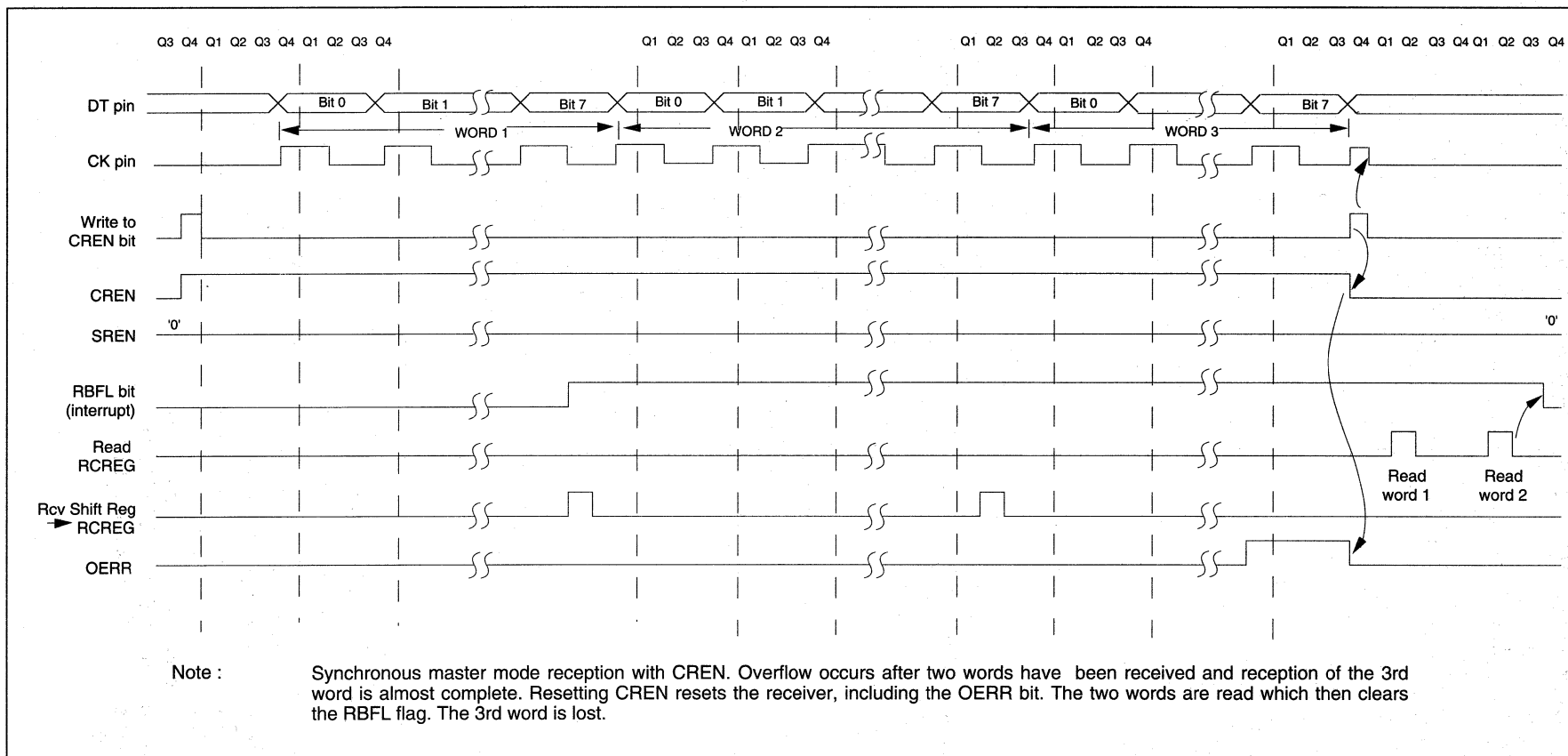


TABLE 7.3.1: BAUD RATES FOR SYNCHRONOUS MODE

BAUD RATE (K)	fosc = 20MHZ			16MHZ			10MHZ			7.15909MHZ		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	NA	-	-	9.766	+1.73	255	9.622	+0.23	185
19.2	19.53	+1.73	255	19.23	+0.16	207	19.23	+0.16	129	19.24	+0.23	92
76.8	76.92	+0.16	64	76.92	+0.16	51	75.76	-1.36	32	77.82	+1.32	22
96	96.15	+0.16	51	95.24	-0.79	41	96.15	+0.16	25	94.20	-1.88	18
300	294.1	-1.96	16	307.69	+2.56	12	312.5	+4.17	7	298.3	-0.57	5
500	500	0	9	500	0	7	500	0	4	NA	-	-
HIGH	5000	-	0	4000	-	0	2500	-	0	1789.8	-	0
LOW	19.53	-	255	15.625	-	255	9.766	-	255	6.991	-	255

BAUD RATE (K)	fosc = 5.0688MHZ			3.579545MHZ			1MHZ			32.768KHZ		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-	0.303	+1.14	26
1.2	NA	-	-	NA	-	-	1.202	+0.16	207	1.170	-2.48	6
2.4	NA	-	-	NA	-	-	2.404	+0.16	103	NA	-	-
9.6	9.6	0	131	9.622	+0.23	92	9.615	+0.16	25	NA	-	-
19.2	19.2	0	65	19.04	-0.83	46	19.24	+0.16	12	NA	-	-
76.8	79.2	+3.13	15	74.57	-2.90	11	83.34	+8.51	2	NA	-	-
96	97.48	+1.54	12	99.43	+3.57	8	NA	-	-	NA	-	-
300	316.8	+5.60	3	298.3	-0.57	2	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	1267	-	0	894.9	-	0	250	-	0	8.192	-	0
LOW	4.950	-	255	3.496	-	255	0.9766	-	255	0.032	-	255

TABLE 7.3.2: BAUD RATES FOR ASYNCHRONOUS MODE

BAUD RATE (K)	fosc = 20MHZ			16MHZ			10MHZ			7.15909MHZ		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	1.221	+1.73	255	1.202	+0.16	207	1.202	+0.16	129	1.203	+0.23	92
2.4	2.404	+0.16	129	2.404	+0.16	103	2.404	+0.16	64	2.380	-0.83	46
9.6	9.469	-1.36	32	9.615	+0.16	25	9.766	+1.73	15	9.322	-2.90	11
19.2	19.53	+1.73	15	19.23	+0.16	12	19.53	+1.73	7	18.64	-2.90	5
76.8	78.13	+1.73	3	83.33	+8.51	2	78.13	+1.73	1	NA	-	-
96	104.2	+8.51	2	NA	-	-	NA	-	-	NA	-	-
300	312.5	+4.17	0	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	312.5	-	0	250	-	0	156.3	-	0	111.9	-	0
LOW	1.221	-	255	0.977	-	255	0.6104	-	255	0.437	-	255

BAUD RATE (K)	fosc = 5.0688MHZ			3.579545MHZ			1MHZ			32.768KHZ		
	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)	KBAUD	%ERROR	SPBRG value (decimal)
0.3	0.31	+3.13	255	0.301	+0.23	185	0.300	+0.16	51	0.256	-14.67	1
1.2	1.2	0	65	1.190	-0.83	46	1.202	+0.16	12	NA	-	-
2.4	2.4	0	32	2.432	+1.32	22	2.232	-6.99	6	NA	-	-
9.6	9.9	+3.13	7	9.322	-2.90	5	NA	-	-	NA	-	-
19.2	19.8	+3.13	3	18.64	-2.90	2	NA	-	-	NA	-	-
76.8	79.2	+3.13	0	NA	-	-	NA	-	-	NA	-	-
96	NA	-	-	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	79.2	-	0	55.93	-	0	15.63	-	0	0.512	-	0
LOW	0.3094	-	255	0.2185	-	255	0.0610	-	255	0.0020	-	255

7.4. SERIAL PORT REGISTERS

7.4.1 Summary of Serial Port Registers

Register Name	Function	Address	Reset Value
RCSTA	Receive status/control register	Bank 0, File 13h	0000000Xb
RCREG	Receive buffer register	Bank 0, File 14h	XXXXXXXXXb
TXSTA	Transmit status/control register	Bank 0, File 15h	0000001Xb
TXREG	Transmit buffer register	Bank 0, File 16h	XXXXXXXXXb
SPBRG	Baud Rate Generator	Bank 0, File 17h	XXXXXXXXXb
PIR	Peripheral interrupt flag register	Bank 1, File 16h	00000010b
PIE	Peripheral interrupt enable register	Bank 1, File 17h	00000000b
INTSTA	Interrupt status register	File 07h	00000000b
CPUSTA	CPU status register	File 06h	0011XX00b

FIGURE 7.4.1.1 RCSTA: RECEIVE STATUS & CONTROL REGISTER

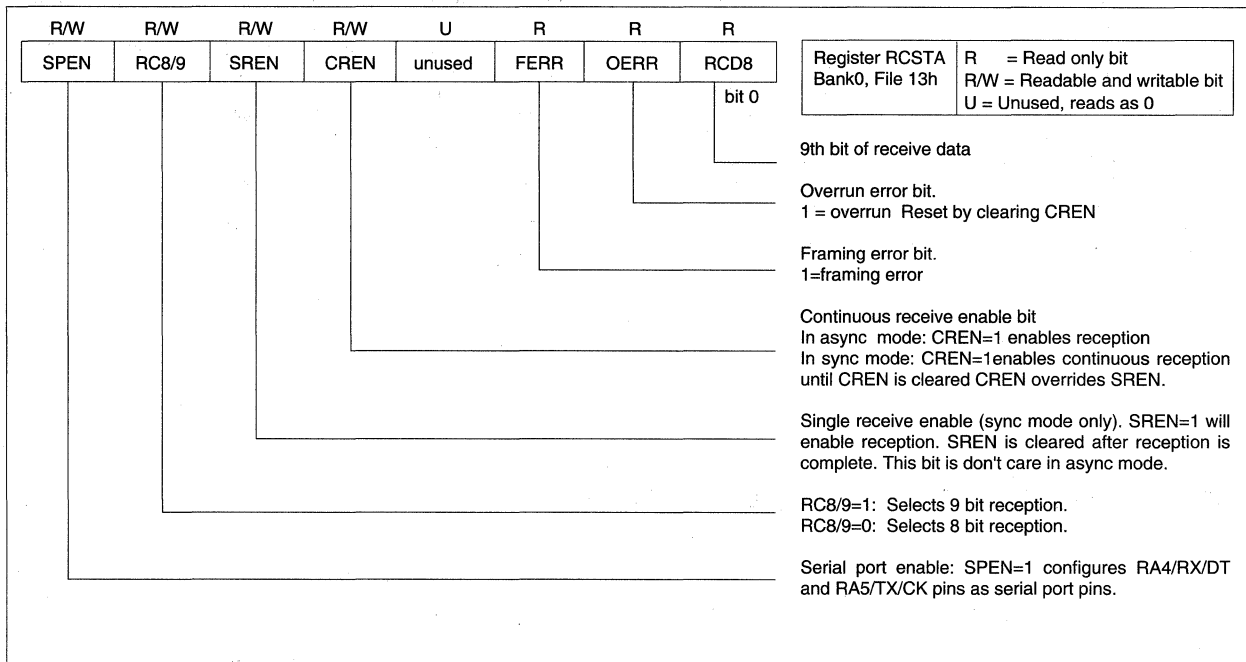
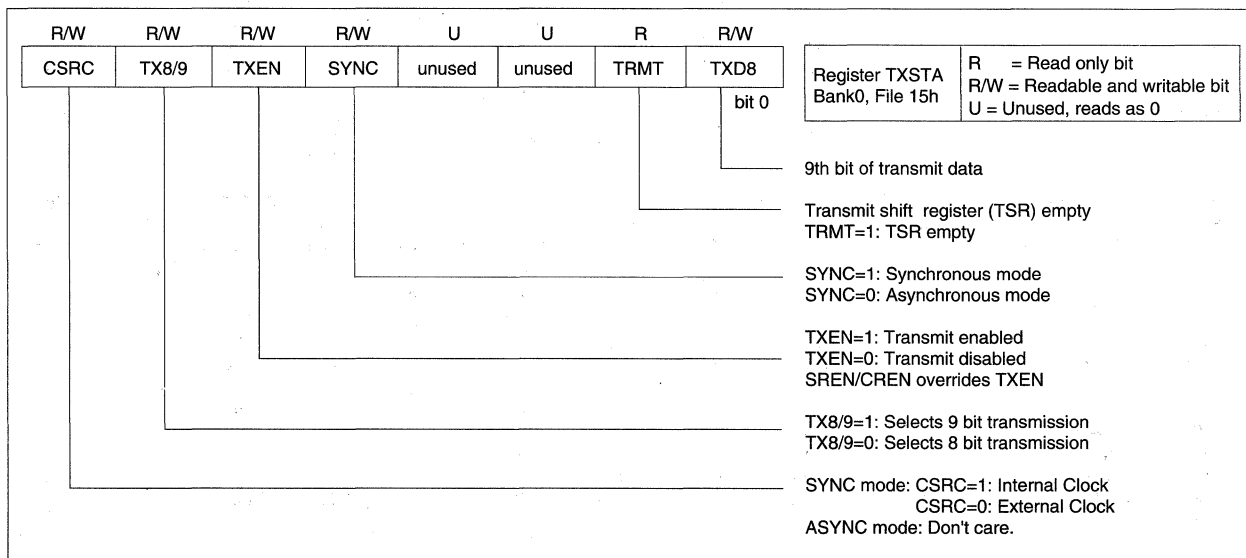


FIGURE 7.4.1.2 TXSTA: TRANSMIT STATUS & CONTROL REGISTER



7.5 SUMMARY OF SERIAL PORT PINS

The serial port uses two pins, RA4/RX/DT and RA5/TX/CK. If SPEN bit (bit 7, RCSTA) is set then these pins are controlled by the serial port. If SPEN=0, then they are configured as input only port pins. **(Both pins have Schmitt Trigger input buffer.)**

Pin Name	SPEN = 0	SPEN = 1		
		SYNC Master Mode	SYNC Slave Mode	ASYNC Mode
RA4/RX/DT	input only port pin	DT: Data in/out Data output if TXEN=1 and CREN=0 and SREN=0, Hi-impedance input otherwise		RX: Receive input, Always hi-impedance input.
RA5/TX/CK	input only port pin	CK: clock output Always a driven output	CK: clock input Always hi-impedance input	TX: Transmit Driven output if TXEN=1. Hi-impedance input if TXEN=0

8.0 TIMER/COUNTERS: OVERVIEW

The PIC17C42 has a rich set of timer/counters: Two 8 bit timer counters (also configurable as one sixteen bit timer/counter) and two 16 bit timer/counters. These can be configured as:

- Two 16 bit + two 8 bit timer/counters
- Three 16 bit timer/counters

A brief overview of these timer/counters is as follows:

RTCC <16>: RTCC <16> is a 16 bit timer/counter consisting of two 8 bit sections (RTCCH <8>, RTCCL <8>). It has a programmable 8 bit prescaler. RTCC can increment off internal clock (OSC/4) or external clock input on the RT pin. RTCC generates an interrupt on overflow.

TMR1 <8>, TMR2 <8>: These are two 8 bit timer/counters. They each have an eight bit period register (PR1 and PR2 respectively) and an interrupt. In counter mode, their clock comes from pin TCLK12 (shared between the two timer/counters). They can be configured as a 16 bit timer/counter with interrupt and a 16 bit period register.

TMR3 <16>: Timer3 is a 16 bit timer/counter consisting of two 8 bit sections TMR3H <8> and TMR3L <8>. It has a 16 bit period register (PR3H <8>, PR3L <8>), an interrupt and an external clock source (pin TCLK3) in counter mode.

8.1 ROLE OF THE TIMER/COUNTERS

The timer/counters are general purpose. However, they have special usage. RTCC is physically part of the 'core'. It is planned that future variations of the PIC17CXX family will include this timer. Therefore, time dependent code, e.g. real time operating system or clock/calender type software can be written using RTCC and ported to future PIC17CXX family members.

TMR3 is also used for 16 bit capture function as is described in capture section. Timers TMR1 and TMR2 can be used as time bases for PWM1 and PWM2 outputs respectively. Alternately, TMR1 can run both PWM outputs and thus free up TMR2 to be a general purpose timer.

These timers are not needed to do the following functions: Watchdog timer (it's a separate timer); Baud Rate generation for serial communication (serial port has its own 8 bit Baud Rate Generator).

8.2 RTCC MODULE

The RTCC (Real time clock/counter) module consists of a 16 bit timer/counter, RTCC (high byte RTCCH, file 0Ch and low byte RTCCL, file 0Bh), an 8 bit prescaler, and the RT pin as the source of external clock signal. The control bits for this module are in register RTCSTA (File 05h).

FIGURE 8.2.1.1: RTCC MODULE BLOCK DIAGRAM

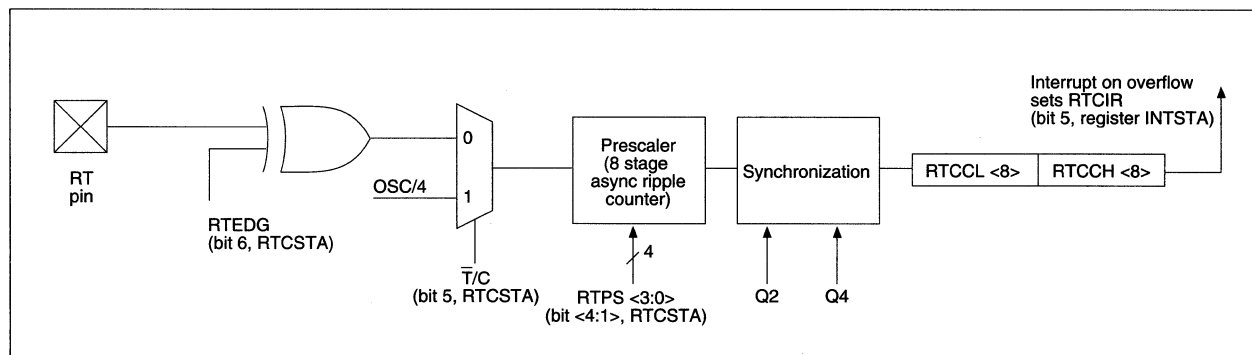
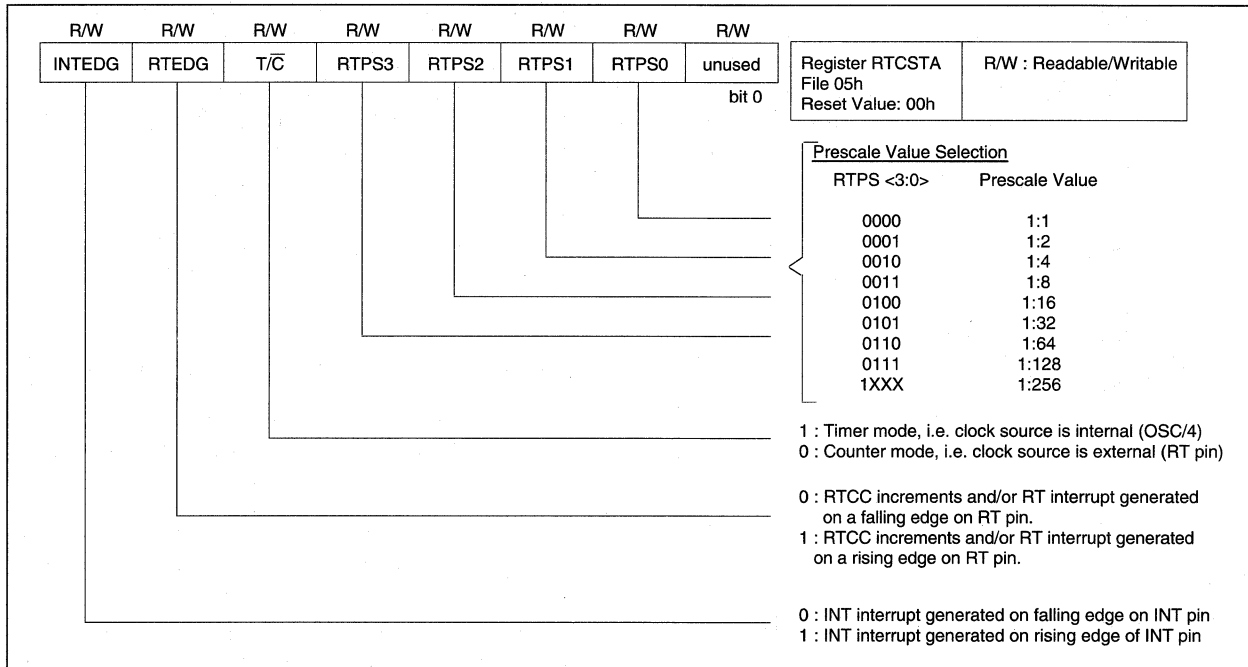


FIGURE 8.2.1.2: RTCSTA: RTCC STATUS/CONTROL REGISTER



8.2.1 RTCC Operation

RTCC increments either on internal clock, OSC/4 (if T/C = '1' in RTCSTA) or on external clock (counter mode) on RT pin (if bit T/C = '0' in RTCSTA). If external clock is chosen, increment can occur on either the rising edge (RTEDG = '1' in RTCSTA register) or the falling edge (RTEDG = '0' in RTCSTA register). The prescaler can be programmed to introduce a prescale of 1:1 to 1:256 in either timer or counter mode. The timers increment from 0000h to FFFFh and roll over to 0000h. On overflow, the RTCC interrupt request flag, RTCIR (bit 5, register INTSTA), is set. The RTCC interrupt can be masked off by clearing the corresponding interrupt mask bit, RTCIE (bit 1, INTSTA). The interrupt request flag, RTCIR, must be cleared in software.

8.2.2 Read/Write Consideration for RTCC

Although the RTCC is a 16 bit timer/counter, only 8 bits at a time can be read or written. This could create a problem unless care is taken.

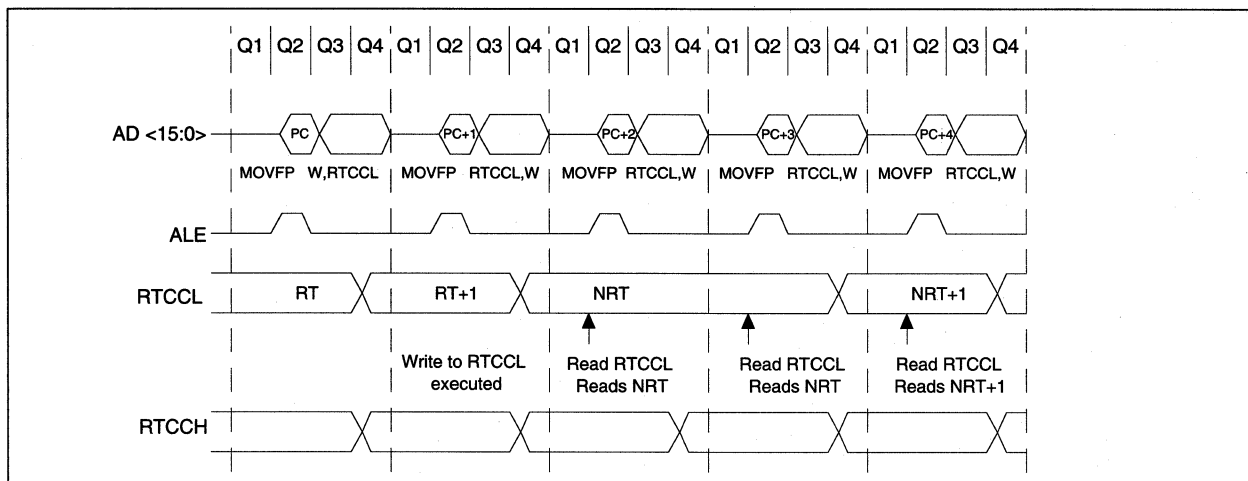
Reading 16 bit value: One problem in reading the entire 16 bit value is that after reading the low (or high) byte it may change from FFh to 00h. This can be handled in software as follows:

```

movpf   rtcc, tmplo   ;read lo rtcc
movpf   rtch, tmphi   ;read hi rtcc
movfp   tmplo, wreg   ;tmplo -> wreg
cpfslt  rtcc, wreg    ;rtcc < wreg?
retfie  ;no then return
movpf   rtcc, tmplo   ;read lo rtcc
movpf   rtch, tmphi   ;read hi rtcc
retfie  ;return
    
```

Interrupt must be disabled during this subroutine.

FIGURE 8.2.2.1: RTCC TIMING: WRITE HIGH OR LOW BYTE



Writing a 16 bit value to the RTCC: Since writing to either RTCCL or RTCCH will effectively inhibit increment of that half of the RTCC in the next cycle (following write), but not inhibit increment of the other half, the user must write to RTCCL first and RTCCH next in two consecutive instructions, as shown below:

```
BSF      CPUSTA, GLINTD      ; Disable interrupt
MOVFP   RAM_L, RTCCL        ;
MOVFP   RAM_H, RTCCH        ;
BCF      CPUSTA, GLINTD      ; Done, enable interrupt
```

Interrupt must be disabled. The user should note that a write to RTCCL or RTCCH will reset the prescaler.

8.2.3 External Clock Considerations

When the external clock input is used for RTCC, it is synchronized with the internal phase clocks. Therefore, the external clock input must meet certain requirements. Also, there is some delay from the occurrence of the external clock edge to the incrementing of RTCC. Referring to Figure 8.2.3.1, the synchronization is done after the prescaler. The output of the prescaler (PSOUT) is sampled twice in every instruction cycle to detect a rising or a falling edge. Therefore, it is necessary for PSOUT to be high for at least 2tosc or low for at least 2tosc where tosc= oscillator time period.

When no prescaler is used (i.e. prescale is 1:1): PSOUT is the same as the RTCC clock input and therefore the requirements are:

$$T_{RTH} = \text{RA1/RT high time} \geq 2tosc + 20 \text{ ns}$$

$$T_{RTL} = \text{RA1/RT low time} \geq 2tosc + 20 \text{ ns}$$

When prescaler is used: the RA1/RT input is divided by the asynchronous ripple counter-type prescaler and so the prescaler output is symmetrical. The requirements are then:

$$\text{PSOUT high time} = \text{PSOUT low time} = \frac{N \cdot T_{RT}}{2}$$

where T_{RT} = RA1/RT input period and

N = prescale value (2, 4, ..., 256).

$$\text{Therefore } \frac{N \cdot T_{RT}}{2} \geq 2tosc + 20 \text{ ns, or } T_{RT} \geq \frac{4tosc + 40 \text{ ns}}{N}$$

The user will notice that no requirement on RTCC high time or low time is specified. However, if the high time or low time on the RTCC input is too small then the pulse may not be detected, hence a minimum high or low time of 10 ns is required. In summary, the RTCC input requirements are:

$$T_{RT} = \text{RA1/RT period} \geq (4tosc + 40 \text{ ns})/N$$

$$T_{RTH} = \text{RA1/RT high time} \geq 10 \text{ ns}$$

$$T_{RTL} = \text{RTCC low time} \geq 10 \text{ ns}$$

Delay from external clock edge: since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time the RTCC is actually incremented. Referring to figure 8.2.3.1, the reader can see that this delay is between 3tosc and 7tosc. Thus, for example, measuring the interval between two edges (e.g. period) will be accurate within $\pm 4tosc$ ($\pm 250 \text{ ns}$ @ 16 MHz).

FIGURE 8.2.2.2: RTCC READ/WRITE IN TIMER MODE

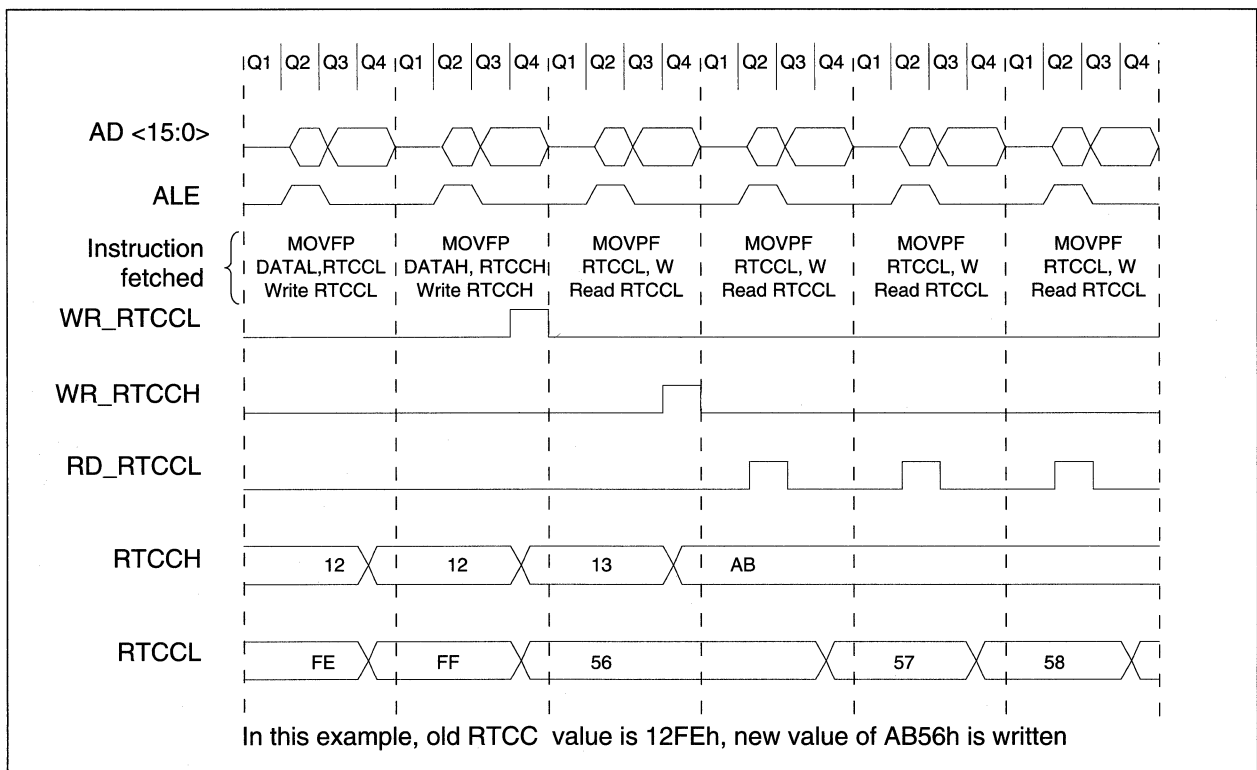
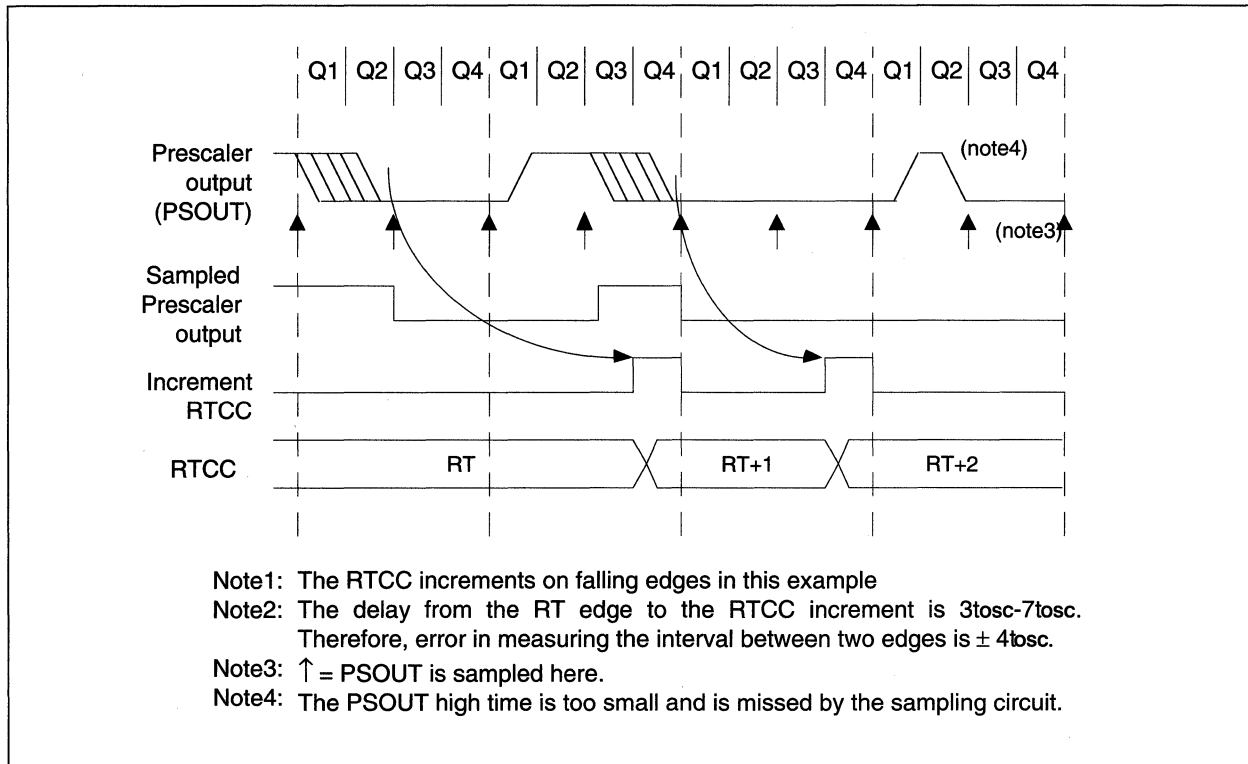


FIGURE 8.2.3.1: RTCC TIMING WITH EXTERNAL CLOCK



8.2.4 Summary of RTCC Registers

Register Name	Function	Address	Reset Value
RTCCCL	RTCC Timer/Counter low byte	File 0Bh	xxxxxxxxb
RTCCH	RTCC Timer/Counter high byte	File 0Ch	xxxxxxxxb
RTCSTA	RTCC Status/Control	File 05h	0000000b
INTSTA	Interrupt Status Register	File 07h	0000000b
CPUSTA	CPU Status Register	File 06h	0011xx00b

8.3 TIMER1 & TIMER2

Timer 1 (TMR1, Bank 2, File 10h) and Timer 2 (TMR2, Bank 2, File 11h) are two 8 bit incrementing timer/counters, each with a period register (PR1, Bank 2, File 14h and PR2, Bank 2, File 15h respectively) and separate overflow interrupt. They can operate as timers (increment on internal OSC/4 clock) or as counters (increment on falling edge of external clock on pin TCLK12). They can operate as two 8 bit timer/counters or as a single 16 bit timer counter. TMR1 and TMR2 are also used as the time base for the PWM (pulse width modulation) module.

8.3.1 Timer1, Timer2 in 8 Bit Mode

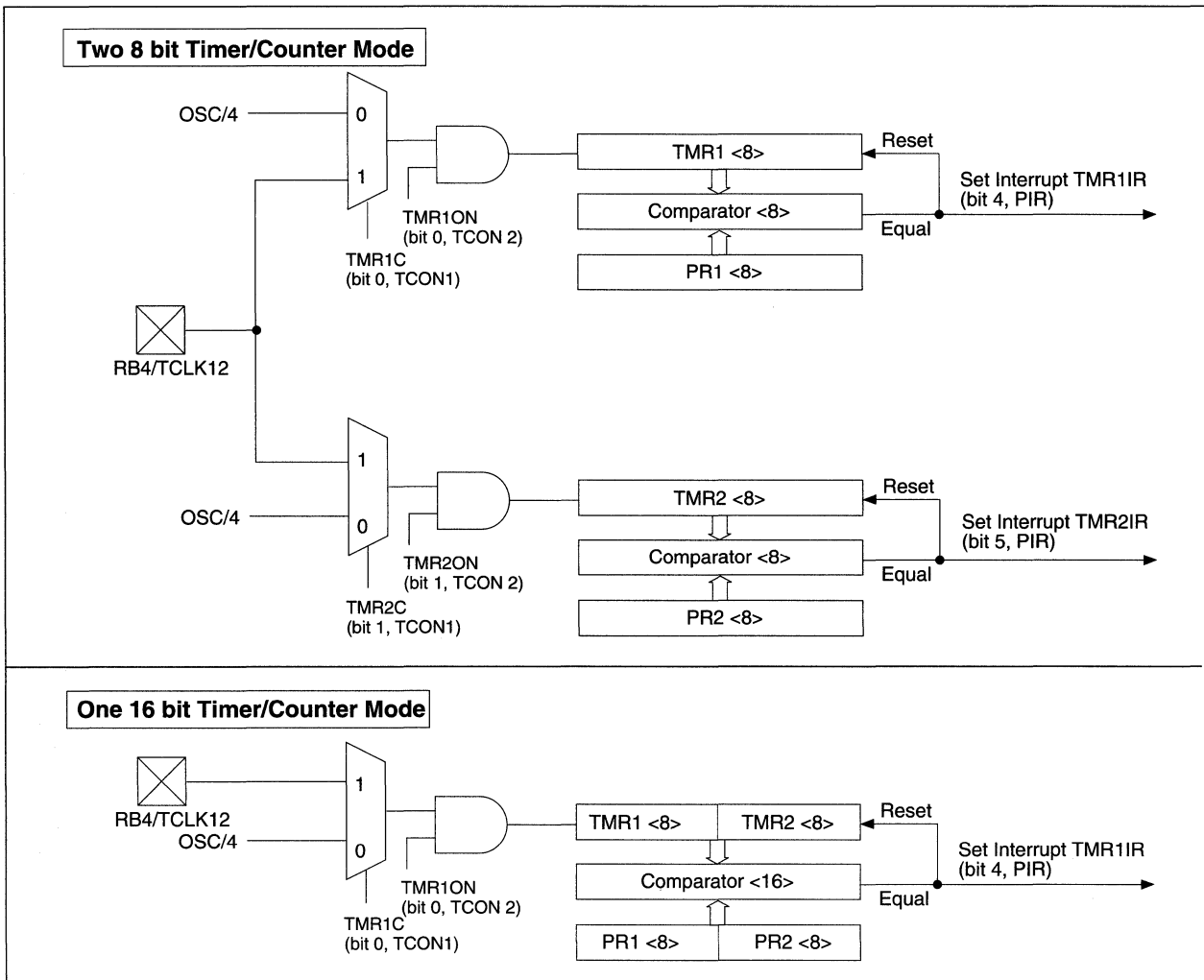
8 bit mode is selected by setting $16/\bar{8}$ (bit 3, register TCON1) to '0'. In this mode, TMR1 will be configured as a timer if control bit TMR1C (bit 0, register TCON1) is '0'

and increment once every instruction cycle (OSC/4). Setting bit TMR1C = '1' will configure TMR1 as a counter. As a counter, TMR1 will increment on every negative edge on pin TCLK12. Since TCLK12 input is synchronized with internal phase clocks, it has to satisfy certain requirements. TCLK12 must be high for at least $(0.5T_{cy} + 20)ns$ and low for at least $(0.5T_{cy} + 20)ns$ where $T_{cy} = 4t_{osc}$. TMR1 increments from 00h until it is equal to PR1 and then resets to 00h at the next increment cycle. An interrupt is generated when reset occurs which is latched in bit TM1IR (TMR1 Interrupt Request Flag, bit 4, PIR). This bit can be masked off by setting bit TM1IE (TMR1 Interrupt Enable) to '0'. In order for the TM1IR interrupt to be recognized, the Peripheral Interrupt Enable bit (PEIE, bit 3, register INTSTA) must be set to a '1' and the Global Interrupt Disable bit, GLINTD, must be '0'. TMR1 must be enabled by setting bit TMR1ON (bit 0, register TCON2) to a '1' and can be stopped any time by clearing bit TMR1ON to '0'. TMR1 and PR1 are both readable and writable registers.

TMR2, in 8 bit mode is identical in functionality as TMR1. The corresponding control bits for TMR2 are TMR2C (bit 1, TCON1), TM2IR (Timer 2 Interrupt-Request Flag, bit

5, PIR), TM2IE (Timer 2 Interrupt Enable Flag, bit 5, PIE) and TMR2ON (bit 1, TCON2). In counter mode, TMR2 also increments on falling edge on TCLK12 pin.

FIGURE 8.3.1.1: TIMER1/TIMER2 BLOCK DIAGRAM



8.3.2 Timer1 & Timer2 in 16 Bit Mode

16 bit mode is selected by setting bit 16/8 (bit 3, register TCON1) to '1'. In this mode TMR1 and TMR2 concatenate to form one 16 bit timer/counter (TMR2 = high byte). Timer mode is selected by setting TMR1C (bit0, register TCON1) to '0' where it increments once every instruction cycle (OSC/4). Counter mode is selected if TMR1C bit = '1' and it increments on every negative edge on pin TCLK12. Input clock on TCLK12 must have a high time $\geq (0.5T_{cy} + 20)ns$ and a low time $\geq (0.5T_{cy} + 20)ns$ where $T_{cy} = 4t_{osc}$. The 16 bit timer increments until it matches the 16 bit value in PR1, PR2 (PR2 = high byte) and then resets back to 0000h. An interrupt is generated at this time which is latched into the TM1IR bit (bit 4, PIR). In 16 bit mode, control bit TMR1C controls the entire 16 bit timer and bit TMR2C is a don't care. The TMR2ON bit must be always set to '1' in 16 bit mode. TMR1ON bit controls the entire 16-bit timer.

8.3.3 External Clock Input for Timer1, Timer2

When configured as a counter, TMR1 or TMR2 increments on the falling edge of clock input TCLK12. However, this input is sampled and synchronized by the internal phase clocks twice every instruction cycle. Therefore, the external clock must meet the following requirements:

$$\begin{aligned} \text{TCLK12 high time} &\geq 0.5 T_{cy} + 20 \text{ ns} \\ \text{TCLK12 low time} &\geq 0.5 T_{cy} + 20 \text{ ns} \end{aligned}$$

There is a delay from the time a falling edge appears on TCLK12 to the time TMR1 or TMR2 is actually incremented. The delay is between $2t_{osc}$ and $6t_{osc}$, where t_{osc} = oscillator period. See Figure 8.3.3.1 for a timing diagram.

FIGURE 8.3.1.2: TMR1, TMR2, TMR3 TIMING IN TIMER MODE

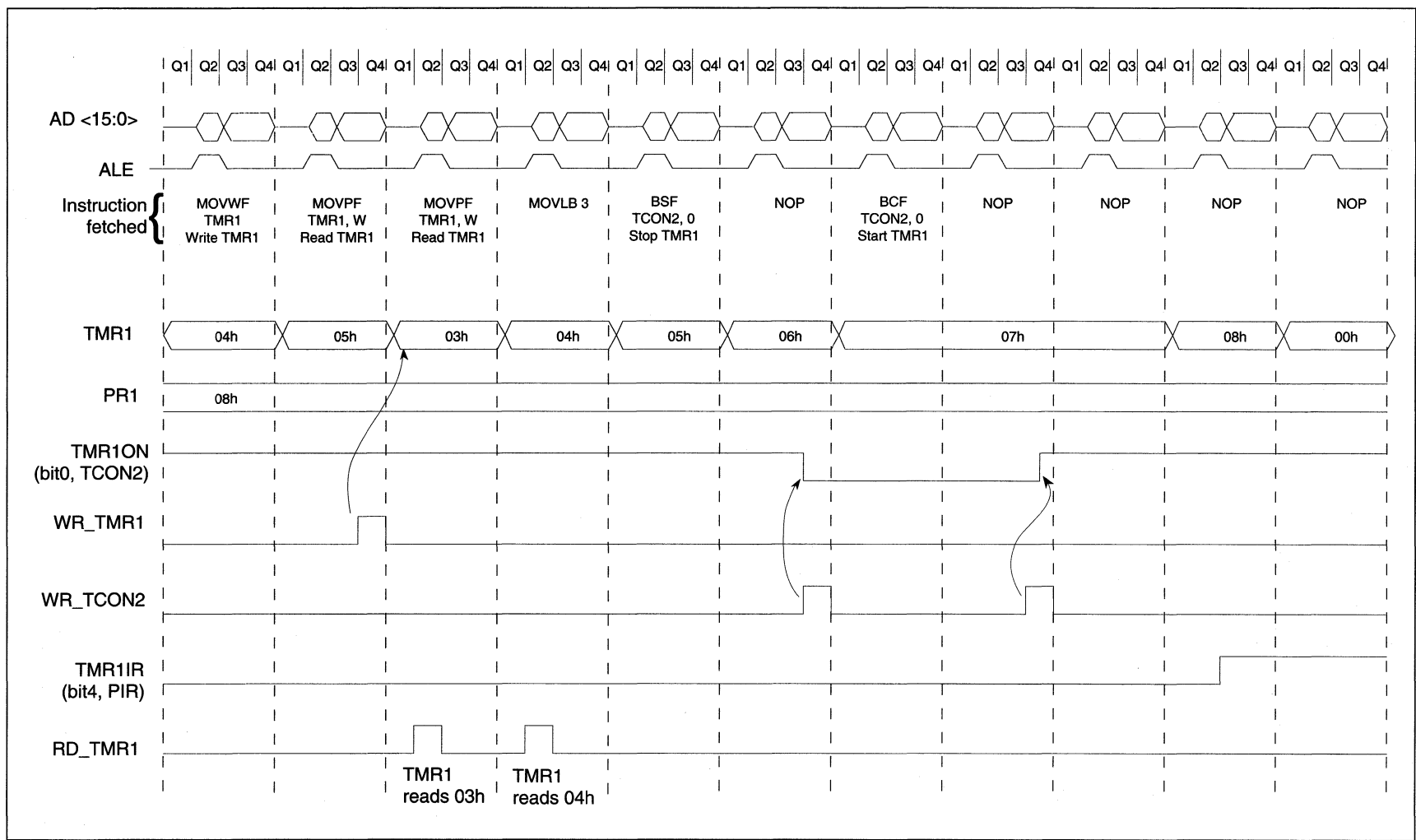


FIGURE 8.3.1.3: TIMER/CAPTURE/PWM CONTROL REGISTER 1 (TCON1)

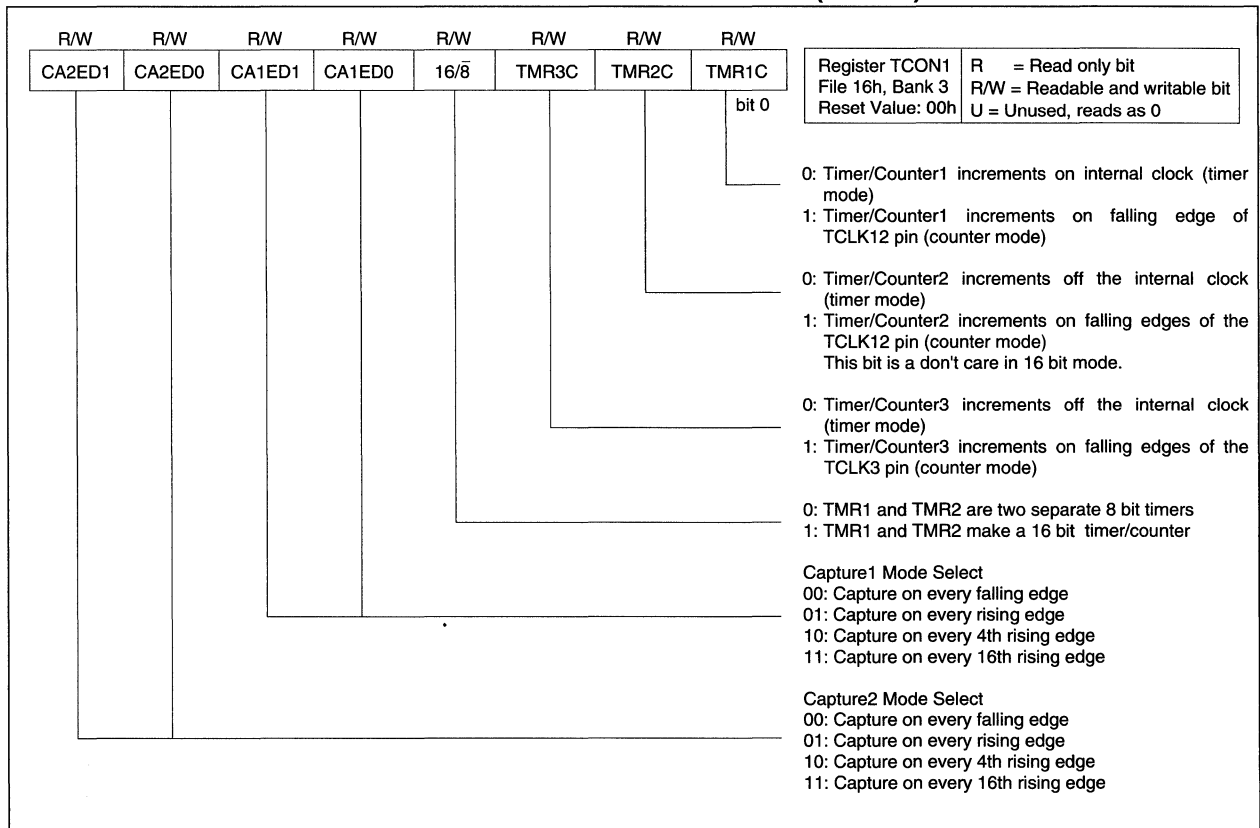


FIGURE 8.3.3.1: TMR1, TMR2 AND TMR3 IN EXTERNAL CLOCK MODE

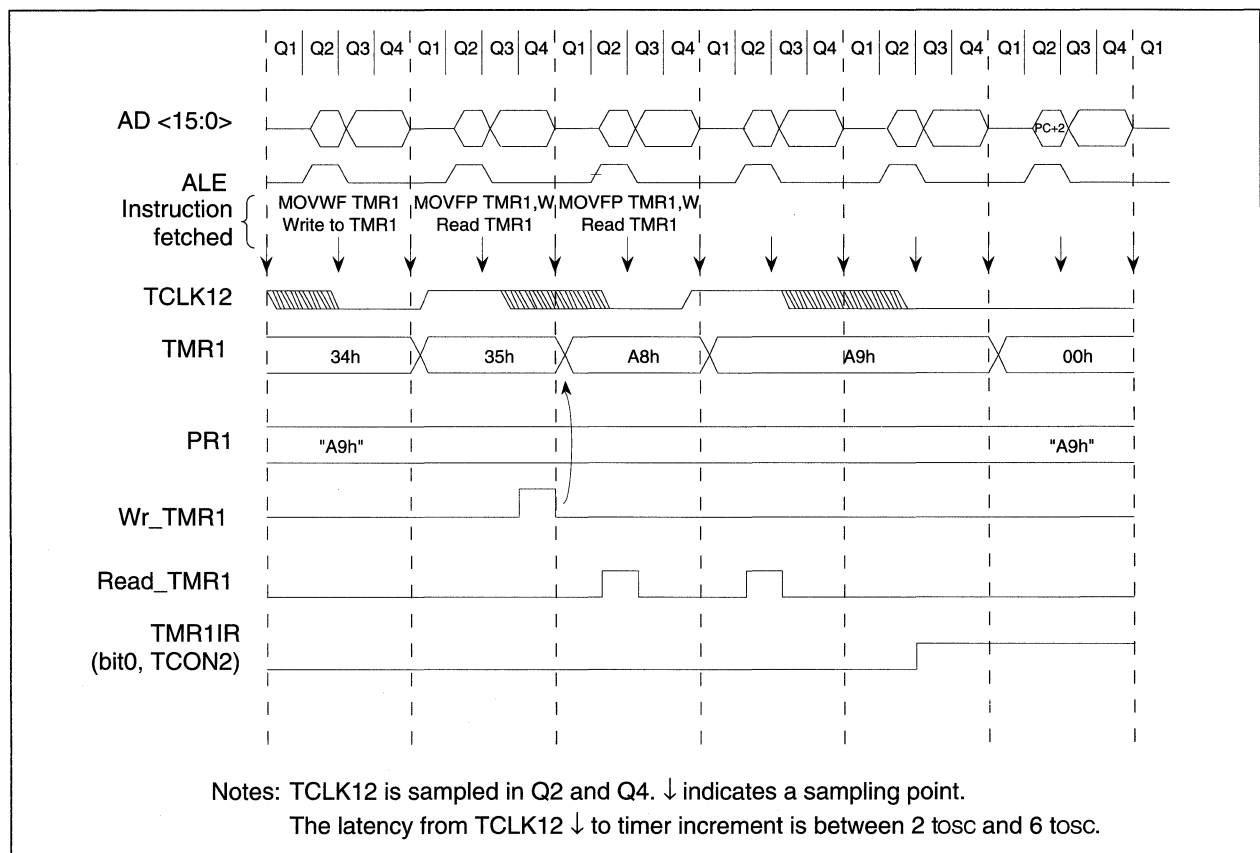
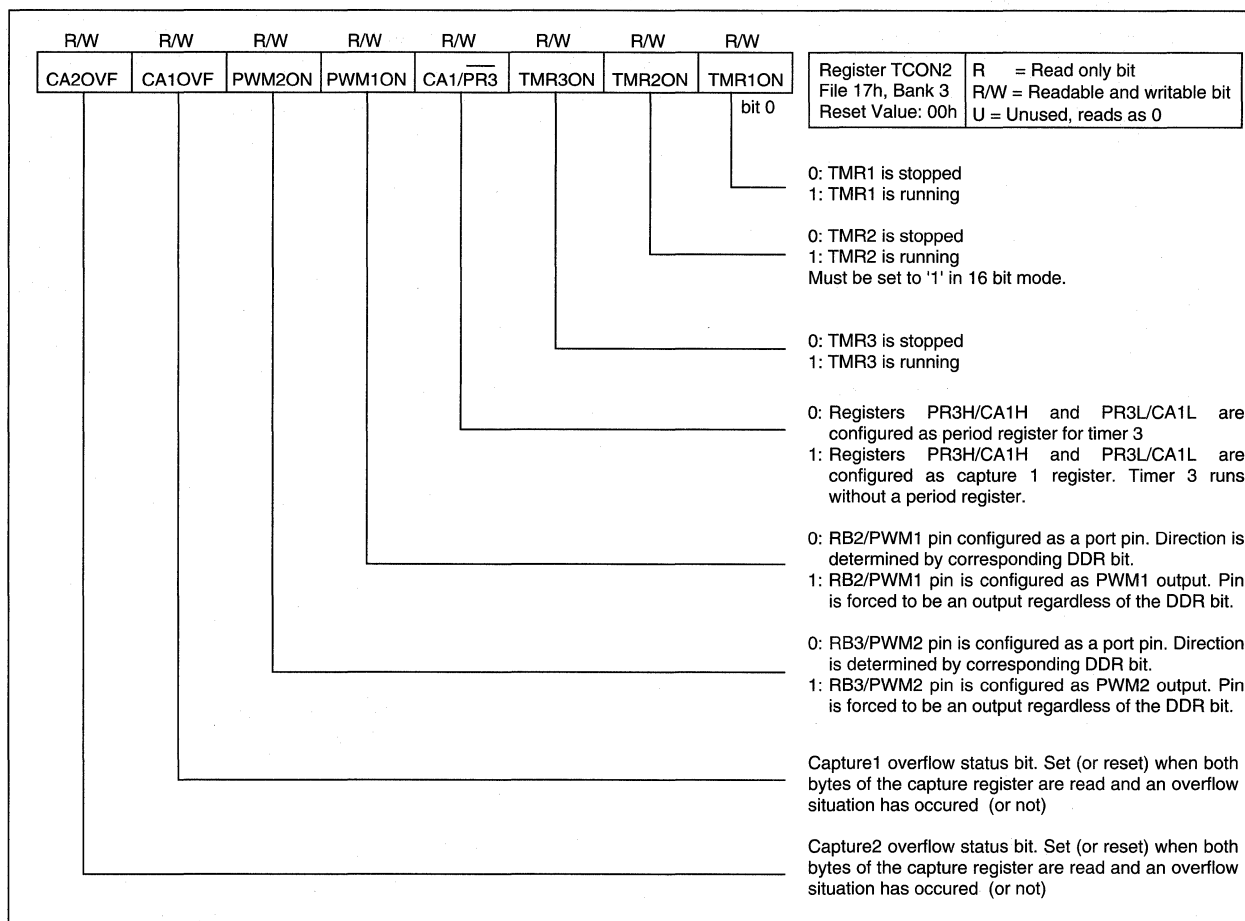


FIGURE 8.3.3.2: TIMER/CAPTURE/PWM CONTROL REGISTER 2 (TCON2)



8.3.4 Summary of Timer1, Timer2 Registers

Register Name	Function	Address	Reset Value
TMR1	Timer/Counter1	Bank 2, File 10h	XXXXXXXXb
TMR2	Timer/Counter2	Bank 2, File 11h	XXXXXXXXb
PR1	Period Register1	Bank 2, File 14h	XXXXXXXXb
PR2	Period Register2	Bank 2, File 15h	XXXXXXXXb
TCON1	Timer Control Register1	Bank 3, File 16h	00000000b
TCON2	Timer Control Register2	Bank 3, File 17h	00000000b
PIR	Peripheral Interrupt Register	Bank 1, File 16h	00000010b
PIE	Peripheral Interrupt Enable	Bank 1, File 17h	00000000b
INTSTA (bit PEIE)	Interrupt Status Register	File 07h	00000000b
CPUSTA (bit GLINTD)	CPU Status Register	File 06h	0011XX00b

8.4 TIMER/COUNTER 3

TMR3 is a 16 bit timer/counter consisting of TMR3L (file 12, Bank 2) as the low byte of the timer and TMR3H (file 13, Bank 2) as the high byte of the timer. It has an associated 16 bit period register consisting of PR3L/CA1L (file 16, Bank 2), the low byte, and PR3H/CA1H (file 17, Bank 2), the high byte. Timer3 is a timer if TMR3C = 0 (bit 2, Register TCON1) in which case it increments every instruction cycle (OSC/4). If TMR3C = 1, the timer 3 acts as a counter and increments on every

falling edge of TCLK3 pin input. In either mode, TMR3 increments if TMR3ON = 1 (bit 2, Register TCON2) and stops if TMR3ON = 0. TMR3 has two modes of operation: depending on bit CA1/PR3 (bit 3, Register TCON2) the period register can be configured as a period or a capture register (Refer to section 9.0 for details on capture operation).

Period register mode, CA1/PR3 = 0: In this mode registers PR3H/CA1H and PR3L/CA1L constitute a 16 bit period register. The timer increments until it equals the

period register and then resets to 0000h. Timer3 interrupt (TM3IR, bit 6, Register PIR) request flag is set at this point. This interrupt can be disabled by setting timer3 mask bit (TM3IE, bit 6, Register PIE) to '0'. TM3IR must be cleared in software.

Capture1 register mode, CA1/PR3 = 1: In this mode the PR3H/CA1H and PR3L/CA1L constitute a 16 bit capture register. The timer operates without a period register and increments from 0000h to FFFFh and rolls over to 0000h. A timer3 interrupt (TM3IR, bit 6, Register PIR) is generated on overflow. The TM3IR interrupt flag must be cleared in software.

8.4.1 External Clock Input for Timer3

Timer3 increments on the falling edges of the clock input on TCLK3 pin. However, this input is sampled and synchronized by the internal phases, twice every instruction cycle. Therefore, the external clock input must meet the following requirements:

TCLK3 high time $\geq 0.5T_{cy} + 20 \text{ ns}$

TCLK3 low time $\geq 0.5T_{cy} + 20 \text{ ns}$

There is a delay from the time an edge occurs on TCLK3 to the time the timer3 is actually incremented. This delay is between 2 tosc and 6tosc, where tosc = oscillator period. See Figure 8.3.3.1 for a timing diagram.

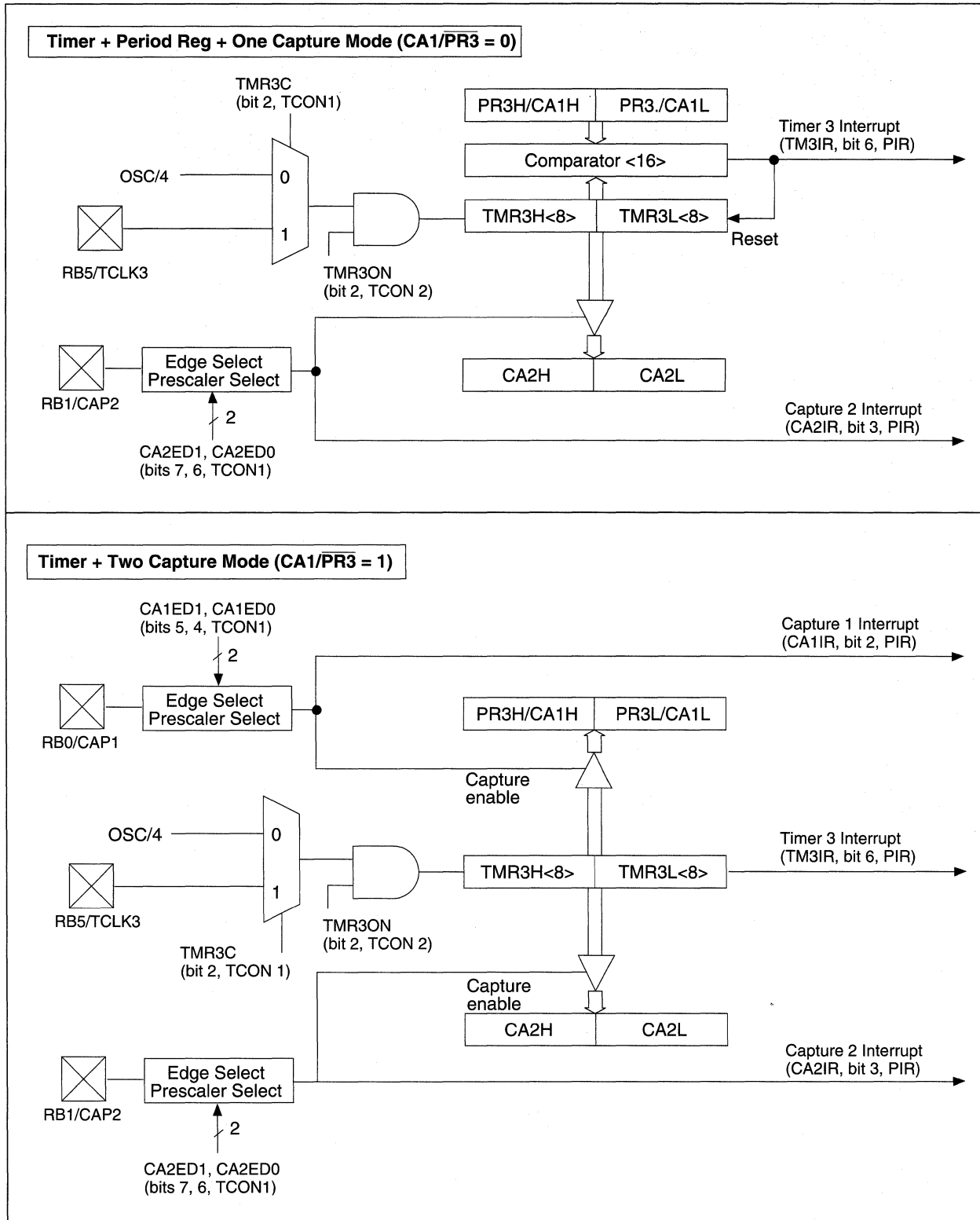
8.4.2 Reading/Writing Timer3

Since timer3 is a 16 bit timer and only 8 bits at a time can be read or written, the user should be careful about reading and writing when the timer is running. The safe and easy thing to do is to stop the timer, perform any read or write operation, and then restart timer3 (using the TMR3ON bit). If, however, it is necessary to keep timer3 free-running then certain suggested methods must be followed for reading and writing the timer. See section 8.2.3 for details.

8.4.3 Summary of Timer3 Registers

Register Name	Function	Address	Reset Value
TMR3L	Timer/Counter3 low byte	Bank 2, File 12h	XXXXXXXXb
TMR3H	Timer/Counter3 high byte	Bank 2, File 13h	XXXXXXXXb
CA2L	Capture2 low byte	Bank 3, File 14h	XXXXXXXXb
CA2H	Capture2 high byte	Bank 3, File 15h	XXXXXXXXb
PR3L/CA1L	Period Register3 low/capture 1 low	Bank 2, File 16h	XXXXXXXXb
PR3H/CA1H	Period Register3 high/capture 1 high	Bank 2, File 17h	XXXXXXXXb
TCON1	Timer Control Register1	Bank 3, File 16h	00000000b
TCON2	Timer Control Register2	Bank 3, File 17h	00000000b
PIR	Peripheral Interrupt Register	Bank 1, File 16h	00000010b
PIE	Peripheral Interrupt Enable	Bank 1, File 17h	00000000b
INTSTA (bit PEIE)	Interrupt Status Register	File 07h	00000000b
CPUSTA (bit GLINTD)	CPU Status Register	File 06h	0011XX00b

FIGURE 8.4.1.1: TIMER3/CAPTURE MODULE BLOCK DIAGRAM



9.0 CAPTURE MODULE

The PIC17C42 has two 16 bit capture registers that capture the 16-bit value of timer/counter3 (TMR3) when events are detected on capture pins. There are two capture pins (RB0/CAP1 and RB1/CAP2), one for each capture register. The capture pins are multiplexed with port B pins. An event can be a rising edge, a falling edge, 4 rising edges or 16 rising edges on the pin. Each capture register has an interrupt request flag associated with it which is set when a capture is made. The capture module is truly part of the timer/counter3/capture block. Refer to Figure 8.4.1.1 for a block diagram. The capture module can operate in one of two modes described below.

9.1 ONE CAPTURE + TIMER/ COUNTER3 + PERIOD REGISTER MODE

This mode is selected if control bit $CA1/\overline{PR3} = 0$ (bit 3, register TCON2). In this mode, the capture1 register, consisting of high byte (PR3H/CA1H, File 17, Bank 2) and low byte (PR3L/CA1L, File 16, Bank 2), is configured as the period control register for TMR3. Capture1 is disabled in this mode, and the corresponding interrupt bit CA1IR (bit 2, PIR) is never set. Timer/counter3 increments until it equals the value in the period register and then resets to 0000h. See Section 8.4 for details of TMR3 operation in this mode.

Capture2 is active in this mode. Control bits CA2ED1 and CA2ED0 (bits 7 & 6, Register TCON1) determine the event on which capture will occur. CA2ED1, CA2ED0 = 00 enables capture on every falling edge, 01 = capture on every rising edge, 10 = capture every 4th rising edge and 11 = capture every 16th rising edge. When a capture takes place, an interrupt is latched into CA2IR (capture 2 interrupt flag, bit 3, PIR). This interrupt can be enabled by setting the corresponding mask bit CA2IE (bit 3, PIE) to '1'. Also, peripheral interrupt enable bit PEIE (bit 3, INSTA) must be a '1' and the Global Interrupt Disable bit (GLINTD, bit 4, CPUSTA), should be '0' for the interrupt to be acknowledged. The CA2IR interrupt flag needs to be cleared in software.

When the capture prescale select is changed, the prescaler is not reset. Therefore, the first capture after such a change will be ambiguous. It, however, sets the basis for the next capture. The prescaler is reset upon chip reset.

The capture pin RB1/CAP2 is a multiplexed pin. When used as a port pin, capture2 is not disabled. However, the user can simply disable the capture2 interrupt by setting CA2IE = '0'. If RB1/CAP2 is used as an output pin,

some interesting possibilities arise. The user can activate a capture by writing to the port pin which may be useful during development phase to emulate a capture interrupt.

The input on capture pin, RB1/CAP2, is synchronized internally to internal phase clocks. This imposes certain restrictions on the input waveform. The minimum high time (T_{CPH}) and the minimum low time (T_{CPL}) on the capture input needs to be greater or equal to 10ns. The period (T_{CAP}) must be $>2T_{cy}/N$ where N = prescale value (1, 4, 16) and where T_{cy} = one instruction cycle time (= 4tosc).

Capture2 Overflow

The overflow status flag bit is double buffered. The master bit is set to '1' if one captured word is already residing in the capture2 register and another 'event' has occurred on RB1/CA2 pin. The new event will not transfer the timer3 value to the capture register, protecting the previous unread capture value. When the user reads both the high and the low bytes (in any order) of the capture2 register, the master overflow bit is transferred to the slave overflow bit (CA2OVF, bit 7, TCON2) and then the master bit is reset. The user can then read TCON2 to determine the value of CA2OVF.

The recommended sequence to read capture registers and overflow is as follows:

```

MOVLB  3           ; Select Bank 3
MOVPPF CA2L, LO_BYTE ; Read capture2 low byte,
                    ; store in LO_BYTE
MOVPPF CA2H, HI_BYTE ; Read capture2 high byte,
                    ; store in HI_BYTE
MOVPPF TCON2, STAT_VAL ; Read TCON2 into file
                    ; STAT_VAL
    
```

9.2 TWO CAPTURE + TIMER/ COUNTER3 MODE

This mode is selected by setting $CA1/\overline{PR3} = 1$ (bit 3, register TCON2). In this mode the timer (TMR3) runs without a period register and increments from 0000h to FFFFh and rolls over to 0000h. For details on TMR3 operation see section 8.4 Registers PR3H/CA1H (file 17h, Bank 2) and PR2L/CA1L (file 16h, Bank 2) make a 16 bit capture register (Capture1). It captures events on pin RB0/CAP1. Capture mode is set by control bits CA1ED1 and CA1ED0 (bit 5 & 4, Register TCON1). A capture1 interrupt is latched into the CA1IR (bit 2, PIR). The corresponding interrupt mask bit is CA1IE (bit 2, PIE). The capture1 overflow status bit is CA1OVF (bit 6, TCON2). Otherwise, capture1 operates identically to capture2. Capture2 operation is same as in the previous mode.

9.3 SUMMARY OF CAPTURE REGISTERS

Register Name	Function	Address	Reset Value
PR3L/CA1L	Period Register 3 low/capture 1 low	Bank 2, File 16h	XXXXXXXXb
PR3H/CA1H	Period Register 3 high/capture 1 low	Bank 2, File 17h	XXXXXXXXb
CA2L	Capture2 register low	Bank 3, File 14h	XXXXXXXXb
CA2H	Capture2 register high	Bank 3, File 15h	XXXXXXXXb
TMR3L	Timer/Counter 3 low	Bank 2, File 12h	XXXXXXXXb
TMR3H	Timer/Counter 3 high	Bank 2, File 13h	XXXXXXXXb
TCON1	Timer Control Register 1	Bank 3, File 16h	0000000b
TCON2	Timer Control Register 2	Bank 3, File 17h	0000000b
PIR	Peripheral Interrupt Register	Bank 1, File 16h	0000010b
PIE	Peripheral Interrupt Enable	Bank 1, File 17h	0000000b
INTSTA (bit PEIE)	Interrupt Status Register	File 07h	0000000b
CPUSTA (bit GLINTD)	CPU Status Register	File 06h	0011XX00b

10.0 PULSE WIDTH MODULATION (PWM) OUTPUTS

The PIC17C42 provides two high speed pulse-width modulation outputs on pins RB2/PWM1 and RB3/PWM2. Each PWM output has a maximum resolution of 10 bits. At 10 bit resolution, the PWM output frequency is 15.6 KHz (@ 16 MHz clock) and at 8 bit resolution the PWM output frequency is 62.5 KHz.

The user needs to set the PWM1ON control bit (bit 4, register TCON2) to enable the PWM1 output. Once the PWM1ON bit = '1', the RB2/PWM1 pin is configured as PWM1 output and forced as an output irrespective of the data direction bit. If PWM1ON = '0', then the pin behaves as a port pin and its direction is controlled by its data direction bit (bit2, DDRB). Similarly, the PWM2ON bit controls the configuration of the RB3/PWM2 pin.

The period of the PWM1 output is determined by timer1 (TMR1) and its period register (PR1). The period of the PWM2 output is determined by timer1 if control bit TM2PW2 = '0' (bit 5, register PW2DCL) or by timer2 if TM2PW2 = '1'.

Thus the PWM periods are:

$$t_{PWM1P} = \text{period of PWM1} = [(PR1) + 1] \times 4 \text{ tosc}$$

$$t_{PWM2P} = \text{period of PWM2} = [(PR1) + 1] \times 4 \text{ tosc}$$

or $[(PR2) + 1] \times 4 \text{ tosc}$

The duty cycle of PWM1 is determined by the 10 bit value DC1<9:0>. The upper 8 bits are from register PW1DCH (file 12, Bank 3) and the lower 2 bits are in register PW1DCL<1:0> (file 10, Bank 3). The PWM1 high time is as follows:

$$t_{PWM1H} = \text{PWM1 high time} = (DC1) \times \text{toscl}$$

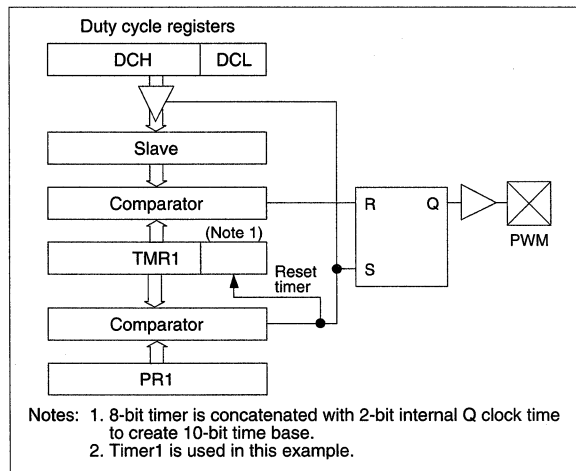
where DC1 represents the 10 bit value from PW1DCH, PW1DCL concatenated.

If DC1 = 0, then the duty cycle is zero. If t_{PWM1H} is equal to or higher than t_{PWM1P} then the duty cycle is 100%.

Similarly, PWM2 high time, $t_{PWM2H} = (DC2) \times \text{toscl}$.

The duty cycle registers for both PWM outputs are double buffered. When the user writes to these registers they are stored in master latches. When TMR1 (or TMR2) overflows, and a new PWM period begins the master latch values are transferred to the slave latches.

FIGURE 10.0.1 - SIMPLIFIED PWM BLOCK DIAGRAM



Using external clock for PWM will also cause jitter in the 'duty cycle' as well as the 'period' of the PWM output. This is because external TCLK12 input is synchronized internally (sampled once per instruction cycle). Therefore, from the time TCLK12 changes to the time timer increments will vary by as much as T_{cy} (one instruction cycle). Therefore, both the high time and the period of the PWM output will have a jitter of $\pm T_{cy}$, unless the external clock is in sync with the processor clock. The latter is the case when TCLK12 input itself is generated by the PIC17C42 (e.g. one PWM output is feedback as TCLK12).

In general therefore, when using external clock reference for PWM, its frequency should be much smaller compared to f_{osc} .

PWM interrupts: The PWM module makes use of timer1 or timer2 interrupts. A timer interrupt is generated when TMR1 or TMR2 equals its period register and is reset to zero. This interrupt, also marks the beginning of a PWM cycle. The user can write new duty cycle values before the next interrupt. The timer1 interrupt is latched into the TM1IR bit (bit4, PIR) and the timer2 interrupt is latched into the TM2IR bit (bit 5, PIR). These flags need to be cleared in software.

Using External clock: Timer1 or timer2, when used as the PWM time base, may be run off external clock only if the PWM output is being generated with 8 bit resolution or less. In this case, the PW1DCL and the PW2DCL registers must be kept at '0'. Any other value will distort the PWM output. Internal clock can be used for all

resolutions. The user should also note that the maximum attainable frequency is lower. Since the maximum possible external clock input frequency for a timer is $1/(T_c + 40)$ ns, (see AC specs) the PWM frequency at 8 bit resolution can be; at most, 13.47 KHz (@ 16 MHz osc clock).

Timer selection for PWM2: While PWM1 always runs based on TMR1, PWM2 can run off timer1 (if bit TM2PW2 = 0, bit 5, Register PW2DCL) or timer2 (if TM2PW2 = 1). Running two different PWM outputs on two different timers allow different PWM period.

Running both PWMs off timer1 allows the best utilization of resources. It frees timer2 to operate as an 8 bit timer/counter. Timer1 and timer2 can not be used as a 16 bit timer if either PWM is being used.

FIGURE 10.0.2 - PWM OUTPUT

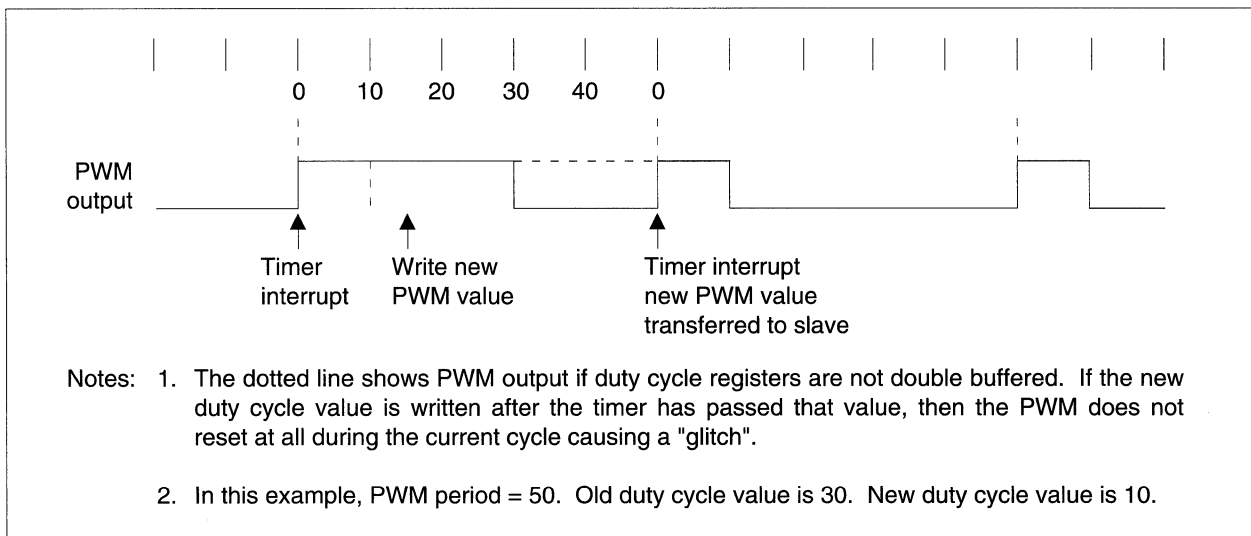


Figure 10.0.1 shows a simplified block diagram of the PWM module. The duty cycle register is double buffered for a glitch free operation. Figure 10.0.2 shows how a glitch could occur if duty cycle registers are not double buffered.

Operating on duty cycle registers: For PW1DCH, PW1DCL, PW2DCH and PW2DCL registers, a write operation writes to the "master latches" while a read operation reads the "slave latches". As a result, the user may not read back what was just written to the duty cycle registers.

The user should also avoid any "read-modify-write" operations on these registers, such as: ADDWF PW1DCH, may not work as intended.

10.1 SUMMARY OF PWM REGISTERS

<u>Register Name</u>	<u>Function</u>	<u>Address</u>	<u>Reset Value</u>
TMR1	Timer/Counter 1	Bank 2, File 10h	XXXXXXXXXb
TMR2	Timer/Counter 2	Bank 2, File 11h	XXXXXXXXXb
PR1	Period Register 1	Bank 2, File 14h	XXXXXXXXXb
PR2	Period Register 2	Bank 2, File 15h	XXXXXXXXXb
TCON1	Timer/Capture/PWM Control Register 1	Bank 3, File 16h	00000000b
TCON2	Timer/Capture/PWM Control Register 2	Bank 3, File 17h	00000000b
PW1DCL	PWM1 duty cycle, lower 2 bits	Bank 3, File 10h	XX000000b
PW1DCH	PWM1 duty cycle, upper 8 bits	Bank 3, File 12h	XXXXXXXXXb
PW2DCL	PWM2 duty cycle, lower 2 bits	Bank 3, File 11h	XX000000b
PW2DCH	PWM2 duty cycle, upper 8 bits	Bank 3, File 13h	XXXXXXXXXb
PIR	Peripheral Interrupt Register	Bank 1, File 16h	00000010b
PIE	Peripheral Interrupt Enable	Bank 1, File 17h	00000000b
INTSTA (bit PEIE)	Interrupt Status Register	File 07h	00000000b
CPUSTA (bit GLINTD)	CPU Status Register	File 06h	0011XX00b

11.0 DEVELOPMENT SUPPORT

The PIC17C42 is supported with a full range of development tools as well as several support programs. These tools and support programs help the user design the PIC17C42 into his or her system easily and quickly. The user can take advantage of the variety of tools from evaluation stage to complex design debug phase. Time to market is significantly reduced by the easy-to-use, PC based tools. All the available and planned tools and programs are described in this section.

11.1 PICASM-17 : PIC17C42 CROSS ASSEMBLER:

The PICASM-17 is a powerful two pass relocatable assembler with advanced MACRO capabilities, high level construct support and source line debug support. It runs on any PC compatible platform. A host of assembler directives support conditional assembly, data area definition and initialization, outputting customized error messages, formatting listing file etc. Advanced MACRO processing capabilities include nesting of macros, conditional macro expansion and parameterized macros. High level constructs such as WHILE and IF-THEN-ELSE permit readable and efficient code writing. ANSI-C style expressions and #define support further makes the assembler more like a high level language.

11.2 PICPAK-17™ : PIC17C42 EVALUATION/DEVELOPMENT/ PROGRAMMER KIT

The PICPAK-17 is a very low cost development kit containing an Evaluation/Development/Programmer PCB, PC-based assembler, and documentation. The EDP board operates in one of three modes:

- Programmer mode:** In this mode the PIC17C42 programs itself from two external 27C64 EPROMs. The user simply programs the EPROMs with the desired code using any standard EPROM programmer.
- External execution mode:** In this mode the PIC17C42 executes out of two external 8K x 8 EPROMs or SRAMs. The user may also plug in a ROM emulator instead of using EPROMs.
- Internal execution mode:** In this mode the PIC17C42 executes from its own internal memory. The external memories are disconnected.

The development board has a solder-less bread-board area with most PIC17C42 signals brought out for easy prototyping and evaluation. Only a single 5V supply is required for the board. Additionally, there is a PIC16C57 microcontroller, which primarily controls the mode selection but is also capable of providing complex stimuli to the PIC17C42 such as stream of capture, timer clock or interrupt pulses, asynchronous data stream or synchronous data stream. Various stimuli are easily selected through DIP switch settings.

11.3 PRO MASTER™ PROGRAMMER

The PRO MASTER programmer is a production quality programmer capable of operating in stand alone mode as well as PC-hosted mode.

The PRO MASTER has programmable VDD and VPP supplies which allows it to verify the PIC at VDD min and VDD max for maximum reliability. It has an LCD display for displaying error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand alone mode the PRO MASTER can read, verify or program a part. It can also set fuse configuration and code-protect in this mode. It's EEPROM memory holds data and parametric information even when powered down. It is ideal for duplicating a large number PIC17C42 for production.

In PC-hosted mode, the PRO MASTER connects to the PC via one of the COM (RS232) ports. PC based user-interface software makes using the programmer simple and efficient. The user interface is full-screen and menu-based. Full screen display and editing of data, easy selection of fuse configuration and part type, easy selection of VDD min, VDD max and VPP levels, load and store to and from disk files (intel hex format) are some of the features of the software. Essential commands such as read, verify, program, blank check can be issued from the screen. Additionally, serial programming support is possible where each part is programmed with a different serial number, sequential or random.

11.4 PICMASTER™-17: HIGH PERFORMANCE UNIVERSAL IN-CIRCUIT EMULATOR SYSTEM

The PICMASTER Universal In-Circuit Emulator System is intended to provide the product development engineer with a complete microcontroller design tool set for all microcontrollers in the PIC16C5X and PIC17CXX families. This system currently supports the PIC16C54, PIC16C55, PIC16C56, PIC16C57 and PIC17C42 processors.

Interchangeable target probes allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the PICMASTER allows expansion to support all new PIC16CXX and PIC17CXX microcontrollers.

The Emulator System is designed to operate on low-cost PC compatible machines ranging from 80286-AT class ISA-bus systems through the new 80486 EISA-bus machines. The development software runs in the Microsoft Windows® 3.1 environment, allowing the operator access to a wide range of supporting software and accessories.

Provided with the PICMASTER System is a high performance real-time In-Circuit Emulator, a microcontroller EPROM programmer unit, a macro assembler program, and a simulator program. Sample programs are provided to help quickly familiarize the user with the development system and the PIC microcontroller line.

Coupled with the user's choice of text editor, the system is ready for development of products containing any of Microchip's microcontroller products.

A "Quick Start" PIC Product Sample Pak containing user programmable parts is included for additional convenience.

Microchip provides additional customer support to developers through an electronic Bulletin Board System (EBBS). Customers have access to the latest updates in software as well as application source code examples. Consult your local sales representative for information on accessing the BBS system.

11.4.1 Host System Requirements:

The PICMASTER has been designed as a real-time emulation system with advanced features generally found on more expensive development tools. The AT platform and Windows 3.1 environment was chosen to best make these features available to you the end user. To properly take advantages of these features, PICMASTER requires installation on a system having the following minimum configuration:

- PC AT compatible machine: 80286, 386SX, 386DX, or 80486 with ISA or EISA Bus.
- EGA, VGA, 8514/A, Hercules graphic card (EGA or higher recommended).
- MSDOS / PCDOS version 3.1 or greater.
- Microsoft Windows® version 3.0 or greater operating in either standard or 386 enhanced mode).
- 1 Mbyte RAM (2 Mbytes recommended).
- One 5.25" floppy disk drive.
- Approximately 10 Mbytes of hard disk (1 Mbyte required for PICMASTER, remainder for Windows 3.0 system).
- One 8-bit PC AT (ISA) I/O expansion slot (half size)
- Microsoft® mouse or compatible (highly recommended).

11.4.2 Emulator System Components:

The PICMASTER Emulator Universal System consists primarily of 4 major components:

- **Host-Interface Card:** The PC Host Interface Card connects the emulator system to an IBM PC compatible system. This high-speed parallel interface requires a single half-size standard AT / ISA slot in the host system. A 37-conductor cable connects the interface card to the external Emulator Control Pod.
- **Emulator Control Pod:** The Emulator Control Pod contains all emulation and control logic common to all microcontroller devices. Emulation memory, trace memory, event and cycle timers, and trace/breakpoint logic are contained here. The Pod controls and interfaces to an interchangeable target-specific emulator probe via a 14" precision ribbon cable.
- **Target-specific Emulator Probe:** A probe specific to microcontroller family to be emulated is installed on the ribbon cable coming from the control pod. This probe configures the universal system for emulation of a specific microcontroller. Currently, the 16C5x family, and the new PIC17C42 microcontrollers are supported. Future microcontroller probes will be available as they are released.

PIC[®]17C42

- **PC Host Emulation Control Software:** Host software necessary to control and provide a working user interface is the last major component of the system. The emulation software runs in the Windows 3.0 environment, and provides the user with full display, alter, and control of the system under emulation. The Control Software is also universal to all microcontroller families.

The Windows 3.1 System is a multitasking operating system which will allow the developer to take full advantage of the many powerful features and functions of the PICMASTER system.

PICMASTER emulation can operate in one window, while a text editor is running in a second window. Dynamic Data Exchange (DDE), a feature of Windows 3.1, will be available in this and future versions of the software. DDE allows data to be dynamically transferred between two or more Windows programs. With this feature, data collected with PICMASTER can be automatically transferred to a spreadsheet or database program for further analysis.

Under Windows 3.1, two or more PICMASTER emulators can run simultaneously on the same PC making development of multi-microcontroller systems possible (e.g., a system containing a PIC16C5x processor and a PIC17Cxx processor).

FIGURE 11.4.1: PICMASTER-17 DEVELOPMENT SYSTEM



FIGURE 11.4.2: PICMASTER-17 DEVELOPMENT SYSTEM BLOCK DIAGRAM

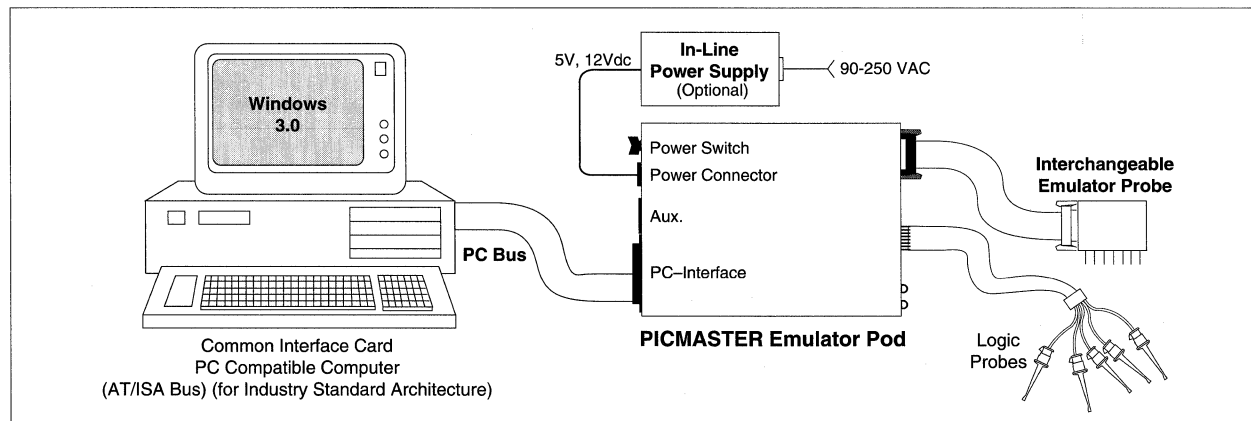
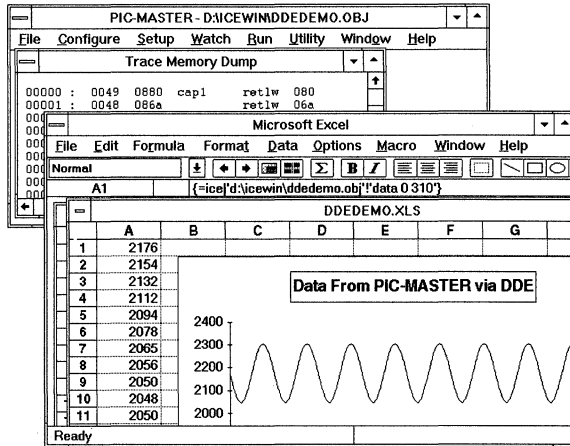


FIGURE 11.4.3: SAMPLE SCREEN LAYOUT FOR PICMASTER-17



11.5 ORDERING DEVELOPMENT TOOLS

The development tools are packaged as comprehensive systems for your convenience. Their description and planned availability dates are as follows:

System	Description	Available
PICPAK-17	Includes: PICASM-17 PIC EDP-17 manuals	Now
PICMASTER-17	Includes: PICASM-17 PRO MASTER PICMASTER PIC17C42 personality module manuals	Now
PRO MASTER™	Includes: PRO MASTER programmer DIP socket module manuals	Now

11.6 APPLICATION AND TECHNICAL SUPPORT

Microchip Technology has a number of sales offices in U.S., Europe, and Asia with highly trained Field Applications Engineers to give you prompt, hands on technical support. Please refer to the back cover page for the sales office and its number nearest to you. In addition, factory technical staff will be glad to help you over the phone (602-963-7373, Chandler, AZ, U.S.A). Application notes and software routines are being made available to give you a jump-start in your system development. These are usually available in printed as well as electronic format.

11.7 PROGRAMMING SUPPORT

The OTP microcontroller provides excellent time-to-market. It offers quick development, over-night code changes and easy to manage inventory. To support your programming needs Microchip offers various options.

11.7.1 Prototype Programming

Prototype programming can be done either using the low cost PICPAK-17 board or the PRO MASTER production quality programmer.

11.7.2 Production Volume Programming

High volume programming for production can be done using the PRO MASTER programmer. Microchip is working with industry leading programming companies to support the PIC17C42 on their programmers. Our low end 8-bit microcontroller family, the PIC16CXX, is now supported by DATA I/O, Logical Devices, BP Microsystems, Baradine and Stag most of which support handlers. Microchip is working to develop a similar level of support for the PIC17CXX family of products.

11.7.3 Factory Programming

High volume factory programming (QTP) is an available service from Microchip Technology. A small price adder and a minimum quantity requirement apply.

11.7.4 Distributor Programming Support

Some of our distributors will support your programming needs. Please contact your distributor for price and volume requirements.

12.0 ELECTRICAL CHARACTERISTICS

12.1 ABSOLUTE MAXIMUM RATINGS

Maximum temperatures

Ambient temperature under bias -55°C to 125°C

Storage temperature -65°C to 150°C

Maximum voltages

V_{DD} to V_{SS} 0V to +7.5V

MCLR to V_{SS} -0.6V to 12V

RA2 and RA3 to V_{SS} -0.6V to 12V

Any pin with respect to V_{SS} -0.6V to V_{DD} +0.6V
(except V_{DD}, MCLR, RA2, RA3)

Maximum currents

Into V_{DD} pin(s) total 150 mA

Out of all V_{SS} pins total 150 mA

Into any pin when configured as output
(except RA2, RA3) 35 mA

Into RA2, RA3 when configured as output 60 mA

Out of any pin when configured as output 20 mA

Into any pin when configured as input ±500 µA

Maximum power dissipation

Total power dissipation 1W

*Notice: Stresses above those listed under "Maximum Ratings" may cause permanent damage to the device. **This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied.** Exposure to maximum rating conditions for extended periods may affect device reliability.

Notes: 1. Total power dissipation should not exceed 1 W for the package. Power dissipation is calculated as follows:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{oh}\} + \sum \{(V_{DD} - V_{oh}) \times I_{oh}\} + \sum (V_{ol} \times I_{ol})$$

2. Voltage spikes below V_{SS} at the MCLR pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the MCLR pin rather than pulling this pin directly to V_{SS}.

12.2 DC CHARACTERISTICS

Operating Conditions: 4.5V ≤ V_{DD} ≤ 5.5V, -40°C ≤ T_A ≤ 85°C unless otherwise stated.

Characteristic	Symbol	Min	Typ†	Max	Unit	Conditions
Supply voltages and currents						
Supply Voltage	V _{DD}	4.5	5.0	5.5	V	
Supply Current (note 1)	I _{DD1}	-	3	6	mA	V _{DD} =5.5V, freq=4MHz
	I _{DD2}	-	95	-	µA	V _{DD} =4.5V, freq=32KHz
	I _{DD3}	-	6	12	mA	V _{DD} =5.5V, freq=8MHz
	I _{DD4}	-	11	24	mA	V _{DD} =5.5V, freq=16MHz
Standby current (notes 2,3)	I _{DDs1}	-	30	50	µA	V _{DD} =4.5V, WDT on
	I _{DDs2}	-	60	100	µA	V _{DD} =5.5V, WDT on
	I _{DDs3}	-	15	25	µA	V _{DD} =4.5V, WDT off
	I _{DDs4}	-	30	50	µA	V _{DD} =5.5V, WDT off
Programming voltage	V _{PP}	11.5	11.75	12.0	V	
Input voltage levels & hysteresis						
All inputs except C, D and E ports (Schmitt trigger inputs) including OSC1 (EC, RC modes) Ports C, D and E (TTL input)	V _{i1}	-	-	0.2 V _{DD}	V	
	V _{iH1}	0.8 V _{DD}	-	-	V	
	V _{hys1}	0.15 V _{DD} *	-	-	V	
	V _{i2}	-	-	0.8	V	
OSC1 (XT, LF modes)	V _{iH2}	2.0	-	-	V	
	V _{i13}	-	-	0.2V _{DD}	V	
	V _{iH3}	0.8 V _{DD}	-	-	V	
Input leakage current						
All pins except MCLR, RA2, RA3	I _{il1}	-	-	±1	µA	V _{SS} ≤ V _{PIN} ≤ V _{DD} (note 4)
MCLR pin	I _{il2}	-	-	±2	µA	V _{SS} ≤ V _{MCLR} ≤ V _{DD}
RA2, RA3 pin	I _{il3}	-	-	±2	µA	V _{SS} ≤ V _{RA2} , V _{RA3} ≤ 12V
MCLR pin	I _{il4}	-	-	10	µA	V _{MCLR} = V _{PP} (note 5)
Pin capacitance						
All pins except MCLR, V _{DD} , V _{SS}	C _{in}	-	10*	-	pF	
MCLR	C _{mclr}	-	20*	-	pF	
Output voltage levels						
RA2, RA3 (open collector)	V _{oh1}	-	-	12.0	V	(note 6)
	V _{ol1}	-	-	3.0	V	I _{ol1} = 60 mA, V _{DD} = 5.5V
PORT C, D & E (TTL)	V _{oh2}	2.4	-	-	V	I _{oh2} = -6 mA, V _{DD} = 4.5V
	V _{ol2}	-	-	0.4	V	I _{ol2} = 6 mA, V _{DD} = 4.5V
OSC2/CLKOUT (RC & EC modes)	V _{oh3}	2.4	-	-	V	I _{oh3} = -5 mA, V _{DD} = 4.5V
	V _{ol3}	-	-	0.4	V	I _{ol3} = 3 mA, V _{DD} = 4.5V
All Outputs except OSC2 (including C, D and E ports)	V _{oh4}	0.9 V _{DD}	-	-	V	I _{oh4} = -2 mA
	V _{ol4}	-	-	0.1 V _{DD}	V	I _{ol4} = 4 mA
Weak pull-up current (PortB)	I _{pu}	60	100	250	µA	Pull-up active, V _{PIN} = V _{SS}
RAM retention voltage	V _{ram}	1.5*	-	-	V	

- †: Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.
 *: Guaranteed by characterization and not tested.
 **: Guaranteed by Design.

- NOTE 1: Supply current is measured with PIC17C42 executing code (from internal test EPROM which is same as microcomputer mode) with all port pins configured as input and forced to V_{DD} or V_{SS}. External clock (rail to rail) is used. The user should note the following:
 a) The code executed from test memory attempts to exercise the chip to make more realistic measurements of I_{DD} (rather than in reset). However, depending on user's code, the current will vary.
 b) The user needs to add the current consumed by output drivers driving external capacitive or resistive load. For capacitive loads, this can be estimated for an individual output pin as: (C_L · V_{DD}) · f where C_L = total capacitive load, f = average frequency with which the pin switches.
 The current due to external capacitance load switching is most significant during external execution.
 c) The current consumed by the oscillator circuit needs to be considered as well. This will be especially significant for RC oscillator, where the current through the external pull up resistor can be estimated as: V_{DD}/(2·R)
 NOTE 2: Standby current is measured under the following conditions: Part in SLEEP, MCLR = V_{DD}, OSC1 and OSC2 pins driven or left floating (makes no difference). All port pins configured as input and tied to V_{SS} or V_{DD}. Standby current is not affected by oscillator type.
 NOTE 3: WDT off implies fuses FWDT1 = FWDT0 = 0 which configures the WDT as a normal timer that shuts off during SLEEP. WDT on implies that the WDT is configured as a watchdog timer (FWDT1, FWDT0 = 01, 10 or 11) which continues to run during SLEEP.
 NOTE 4: With any weak pull-up disabled.
 NOTE 5: When not programming
 NOTE 6: RA2 and RA3 are open collector outputs that will pull-up to externally applied voltage (through resistor pull-ups). Maximum allowable V_{OH} = 12V.

12.3 AC CHARACTERISTICS

12.3.1 AC Characteristics: OSC/Reset/System bus

Operating Conditions: 4.5V ≤ V_{DD} ≤ 5.5V, -40°C ≤ T_A ≤ 85°C unless otherwise stated.

Characteristic	Symbol	Min	Typ [†]	Max	Unit	Comments
Input clock and oscillator frequencies						
Oscillator frequency	FoscLf	DC	-	200	KHz	LF osc mode
	Foscxt	0.2	-	16	MHz	XT osc mode
RC mode frequency	Foscrc	DC	-	4	MHz	RC osc mode
Recommended limits:	R	2	-	50	Kohm	
	C	20	-	1000	pF	
External clock in frequency	Fextck	DC	-	16	MHz	EC mode (external clock)
Instruction cycle time	Tcy	-	4/Fosc	-	ns	Fosc = osc/clock-in frequency
Clock-in (OSC1) high or low time	TckHL	15*	-	-	ns	For external clock input in XT, LF or EC mode.
Clock-in (OSC1) rise or fall time	TckRF	-	-	15*	ns	For external clock input in XT, LF or EC mode.
Reset timing						
MCLR pulse width	tmcL	100*	-	-	ns	
MCLR ↓ to AD<15:0> high impedance	tmcL2adZ	-	-	50*	ns	
WDT, OST, PWRT and POR timings						
WDT period	twdt	-	20*	-	ms	Prescale = 1
Power up timer period	tpWRT	-	80*	-	ms	
Oscillator start-up timer (OST) period	tOST	-	1024 tOSC**	-	ns	tosc = oscillator period
V _{DD} rise time for POR to function properly	tVDDR	-	-	80*	ms	Time for V _{DD} to rise from 0V to 4.5V (Note 1)
V _{DD} start voltage to guarantee power on reset	V _{POR}	-	V _{SS} *	-	V	See section 4.4 for details
System bus timings						
Address out valid to ALE ↓ (address setup time)	tadV2aIL	0.25 Tcy-30	-	-	ns	with 100 pF load on all address/data and control (ALE,OE,WR) pins.
ALE ↓ to address out invalid (address hold time)	taIL2adI	5	-	-	ns	
AD <15:0> high impedance to OE ↓	tadZ2oeL	0	-	-	ns	
OE ↑ to AD<15:0> driven	toeH2adD	0.25 Tcy-15	-	-	ns	
Data in valid before OE ↑ (data setup time)	tadV2oeH	30	-	-	ns	
OE ↑ to data in invalid (data hold time)	toeH2adI	0	-	-	ns	
Data out valid to WR ↓ (data setup time)	tadV2wrL	0.25 Tcy-40	-	-	ns	
WR ↑ to data out invalid (data hold time)	twrH2adI	5	-	-	ns	
ALE pulse width	taIH	-	0.25 Tcy**	-	ns	
OE pulse width	toeL	0.5 Tcy-25**	-	-	ns	
WR pulse width	twrL	-	0.25 Tcy**	-	ns	
ALE ↑ to ALE ↑ (cycle time)	taIH2aIH	-	Tcy**	-	ns	
Capacitive load on output pins						
OSC2	Cload1	-	-	25	pF	(note 2)
ALE, WR, OE and AD<15:0>	Cload2	-	-	100	pF	(note 3)
All other pins, including C, D, E ports (when used as port)	Cload3	-	-	50	pF	(note 3)

See footnotes on next page.

- †: Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.
 - *: Guaranteed by characterization
 - ** : Guaranteed by design
- NOTE 1: V_{DD} must start from 0V for Power on reset to function properly. V_{DD} rise time can be longer but then external POR circuitry will be required.
- NOTE 2: In EC and RC oscillator modes when OSC2 pin is outputting CLKOUT, or in XT or LP mode when external clock is driven into OSC1 pin.
- NOTE 3: All AC specs are valid for these capacitive loadings

12.3.2 AC Characteristics: Serial Port

Operating Conditions: 4.5V ≤V_{DD} ≤5.5V, -40°C ≤T_A ≤85°C unless otherwise stated.

Characteristic	Symbol	Min	Typ†	Max	Unit	Comments
SYNC XMIT (MASTER & SLAVE)						
Clock high to data out valid	tckH2dtV	-	-	50	ns	
Clock out rise time and fall time (Master Mode)	tckrf	-	-	25	ns	
Data out rise time and fall time	tdtrf	-	-	25	ns	
SYNC RCV (MASTER & SLAVE)						
Data in valid before CK ↓ (DT setup time)	tdtV2ckL	15	-	-	ns	
Data in invalid after CD ↓ (DT hold time)	tckL2dtl	15	-	-	ns	

†: Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

12.3.3 AC Characteristics: I/O Port

Operating Conditions: 4.5V ≤V_{DD} ≤5.5V, -40°C ≤T_A ≤85°C unless otherwise stated.

Characteristic	Symbol	Min	Typ†	Max	Unit	Comments
CLKOUT ↑ to Port out valid	tckH2rxV	-	-	0.5T _{cy} +20	ns	note 1
Port A, B, C, D, E in valid before CLKOUT ↑ (RC and EC mode)	trxV2ckH	0.25 T _{cy} +25	-	-	ns	

†: Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

NOTE 1: Timings are valid for a maximum of 50pF total capacitive load on the port pins, and CLKOUT pin.

12.3.4 AC Characteristics: RTCC & INT

Operating Conditions: $4.5V \leq V_{DD} \leq 5.5V$, $-40^{\circ}C \leq T_A \leq 85^{\circ}C$ unless otherwise stated.

Characteristic	Symbol	Min	Typ†	Max	Unit	Comments
<u>RTCC in ext clock, prescale = 1</u>						
RT clock input high time	trtH1	$0.5 T_{cy} + 20^{**}$	-	-	ns	
RT clock input low time	trtL1	$0.5 T_{cy} + 20^{**}$	-	-	ns	
<u>RTCC in ext clock, prescale > 1</u>						
RT clock input high time	trtH2	10*	-	-	ns	
RT clock input low time	trtH2	10*	-	-	ns	
RT clock input period	trtP	-	$\frac{T_{cy} + 40^{**}}{N}$	-	ns	N = prescale value (2,4,8,...,256)
<u>RT and INT interrupt input</u>						
RT and INT input high time	triH	25*	-	-	ns	
RT and INT input low time	triL	25*	-	-	ns	

†: Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

* Guaranteed by characterization

** Guaranteed by design and characterization

12.3.5 AC Characteristics: Timer1, Timer2, Timer3, Capture and PWM

Operating Conditions: $4.5V \leq V_{DD} \leq 5.5V$, $-40^{\circ}C \leq T_A \leq 85^{\circ}C$ unless otherwise stated.

Characteristic	Symbol	Min	Typ†	Max	Unit	Comments
<u>Timer1, Timer2, Timer3</u>						
Input clock high time on pins TCLK12, TCLK3	tcH	$0.5 T_{cy} + 20^{**}$	-	-	ns	
Input clock low time on pins TCLK12, TCLK3	tcL	$0.5 T_{cy} + 20^{**}$	-	-	ns	
<u>Capture1, Capture2</u>						
Input high time on RB0/CAP1, RB1/CAP2	tcpH	10*	-	-	ns	
Input low time on RB0/CAP1, RB1/CAP2	tcpL	10*	-	-	ns	
Input period on RB0/CAP1, RB1/CAP2	tcpL	$\frac{2 T_{cy}^{**}}{N}$	-	-	ns	where N=capture prescale (1, 4, 16)

†: Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

* Guaranteed by characterization

12.3.6 AC TEST LOAD AND TIMING CONDITIONS

FIGURE 12.3.6.1 INPUT LEVEL CONDITIONS

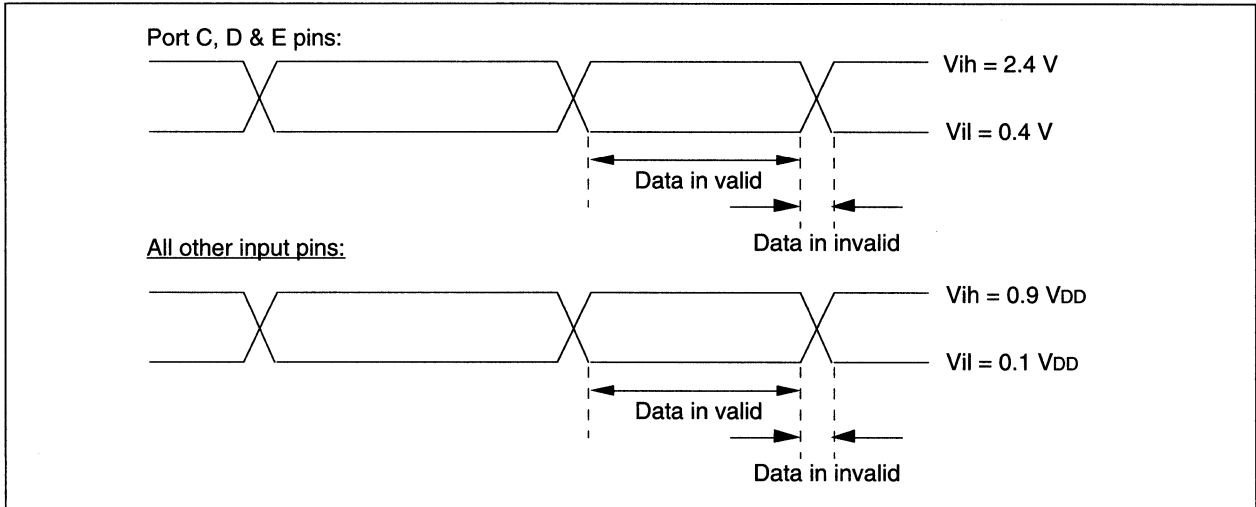


FIGURE 12.3.6.2 OUTPUT LEVEL CONDITIONS

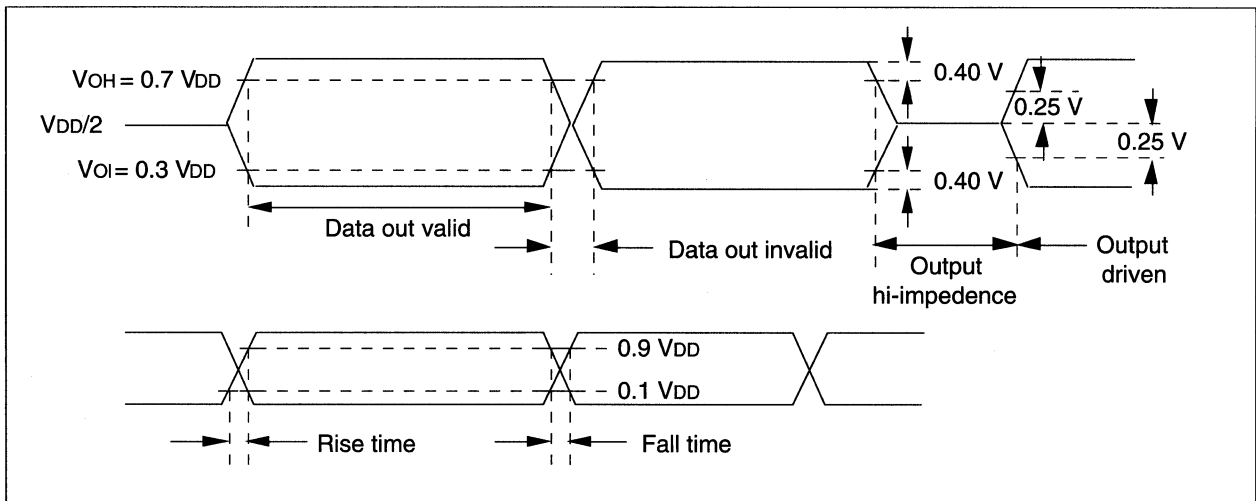
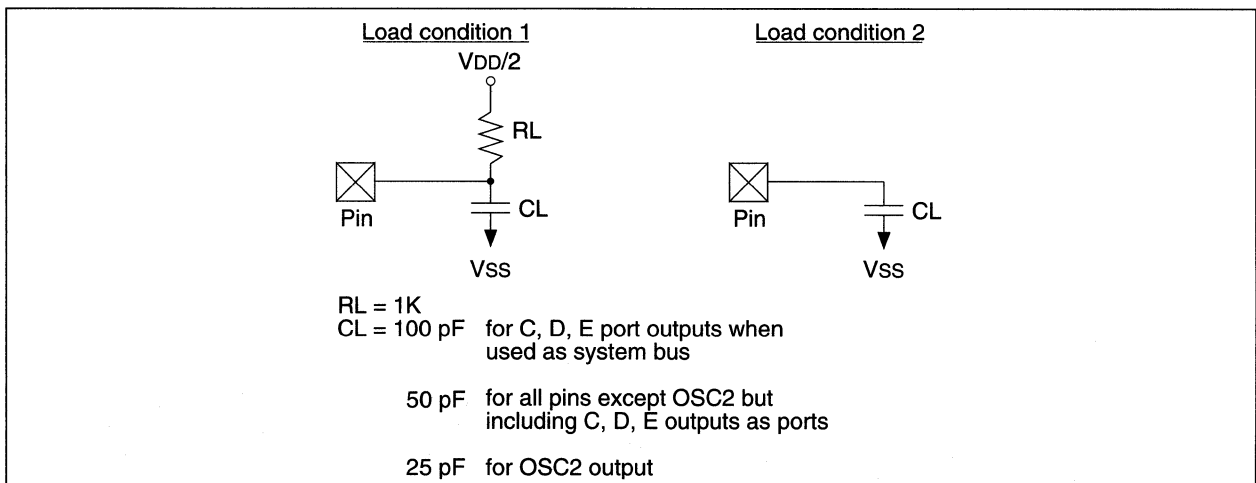


FIGURE 12.3.6.3 LOAD CONDITIONS



12.4 TIMING DIAGRAMS

FIGURE 12.4.1: TIMING DIAGRAM - EXTERNAL PROGRAM MEMORY READ

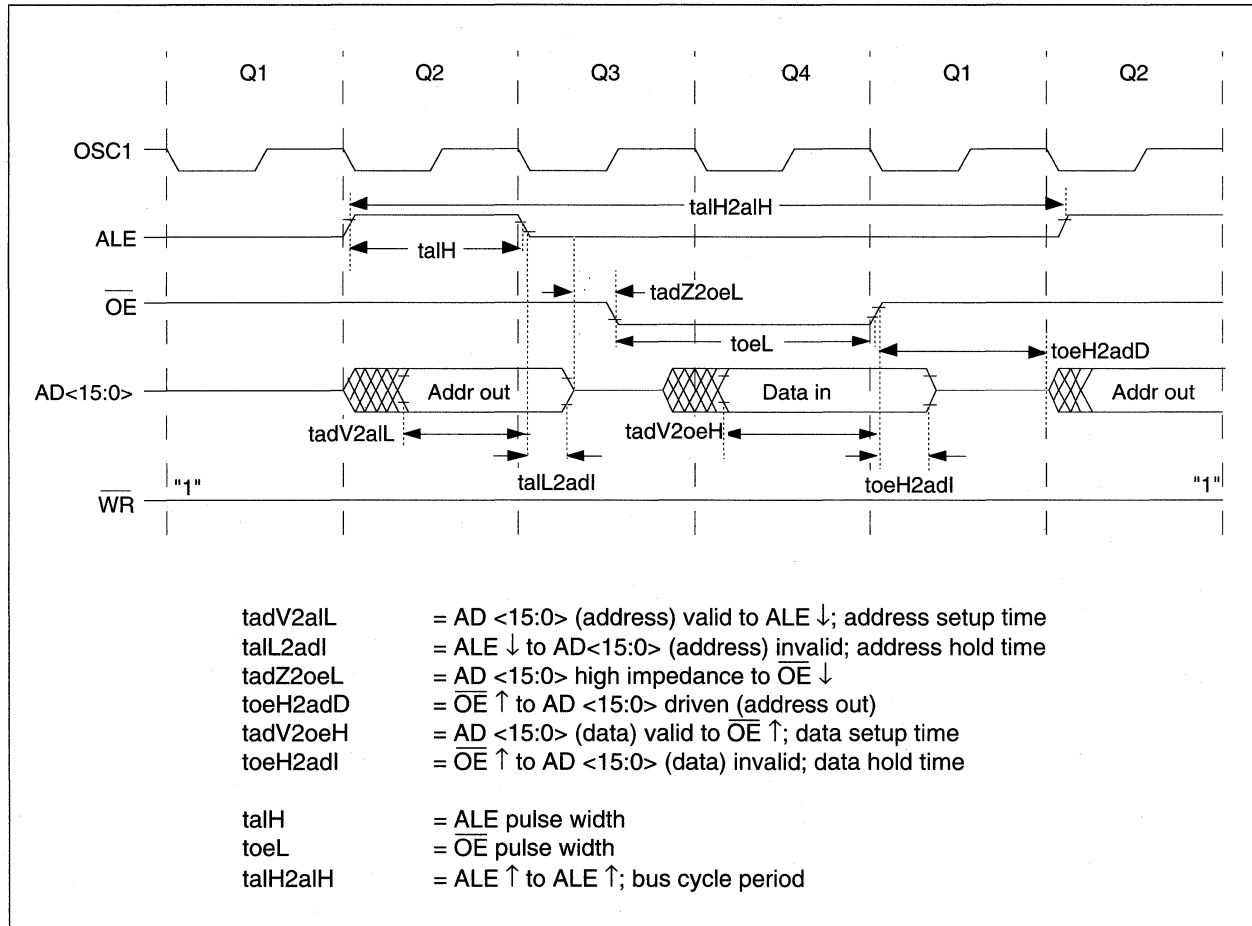


FIGURE 12.4.2: TIMING DIAGRAM - EXTERNAL PROGRAM MEMORY WRITE

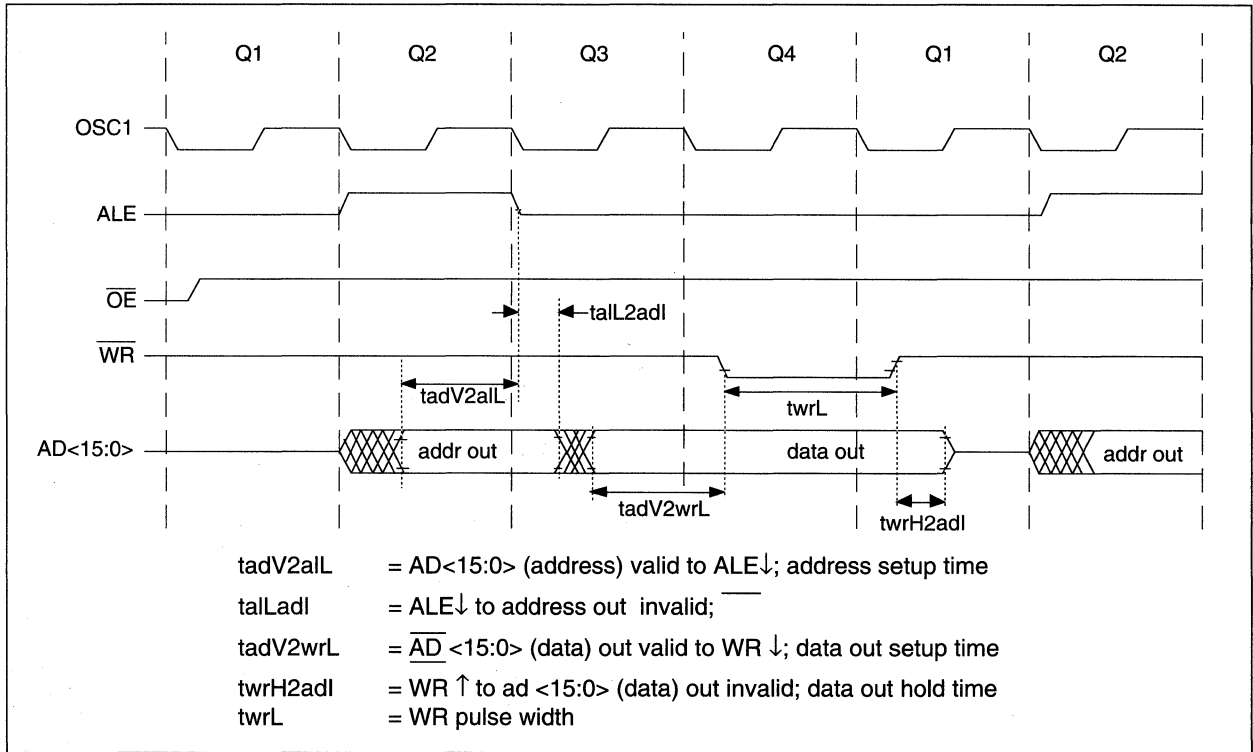


FIGURE 12.4.3: TIMING DIAGRAM - INTERRUPT TIMING

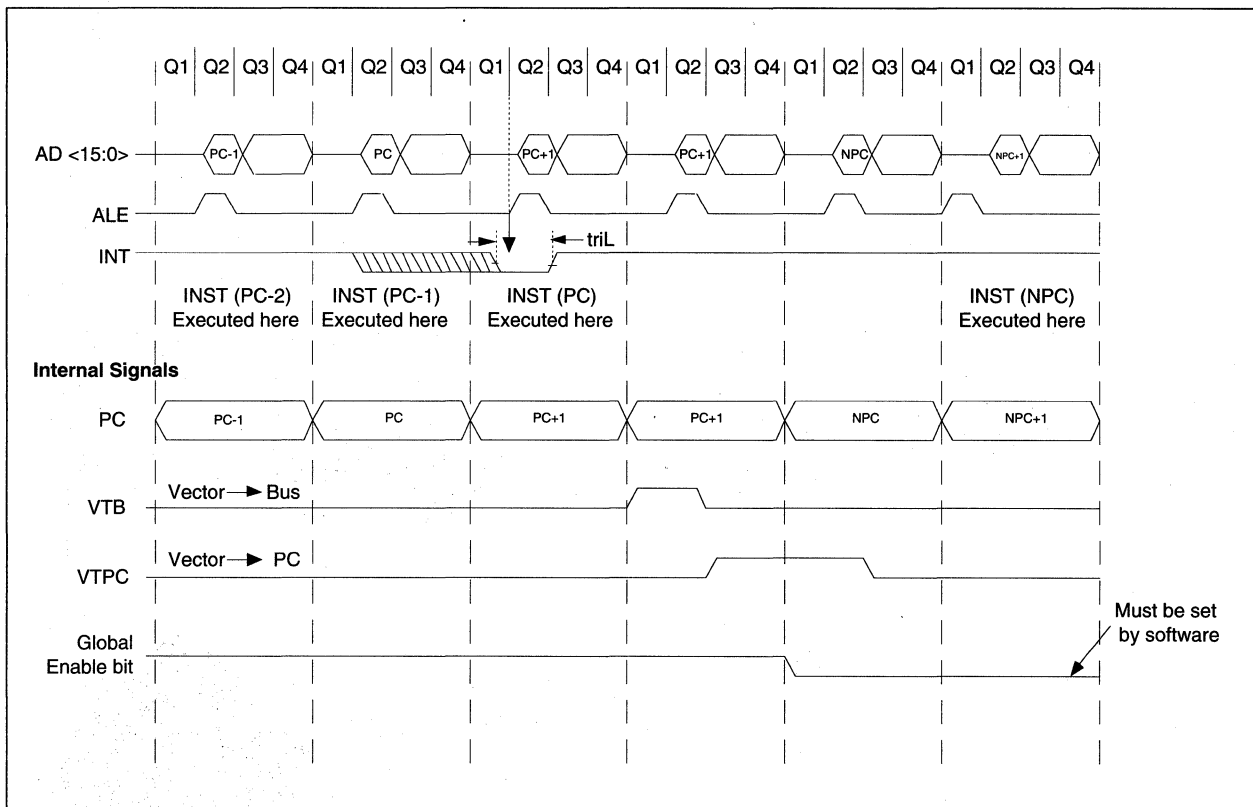


FIGURE 12.4.4: RESET TIMING

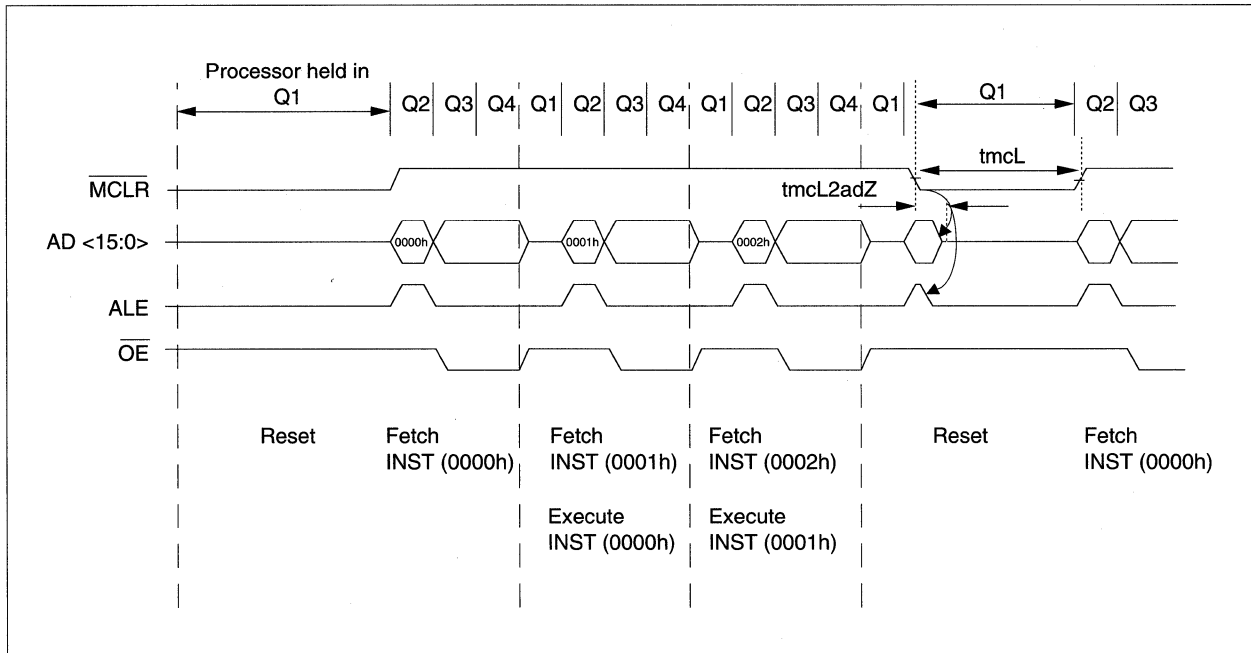


FIGURE 12.4.5: TABLRD TIMING

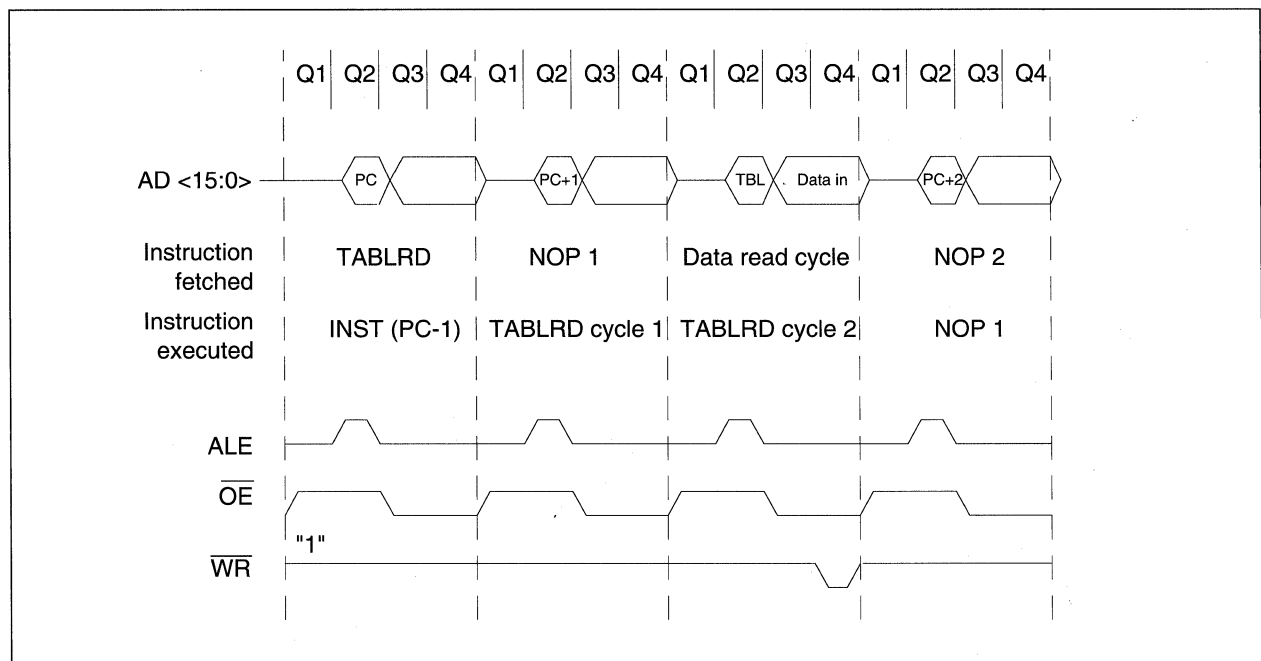


FIGURE 12.4.6: TABLRD TIMING (CONSECUTIVE TABLRD INSTRUCTIONS)

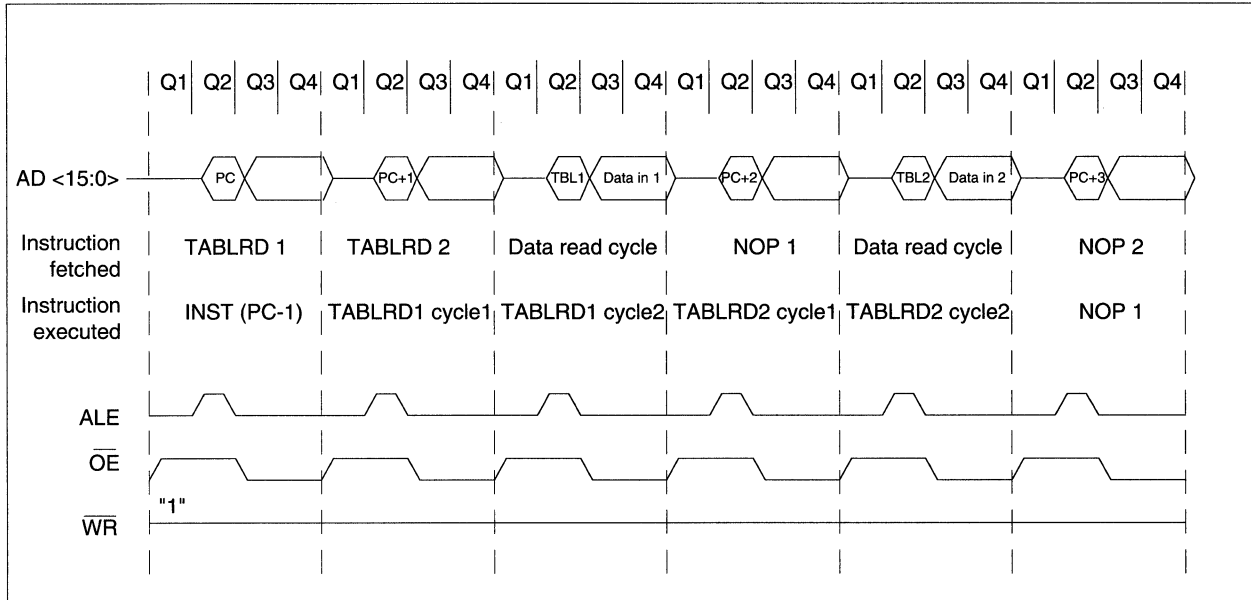


FIGURE 12.4.7: TABLWT TIMING

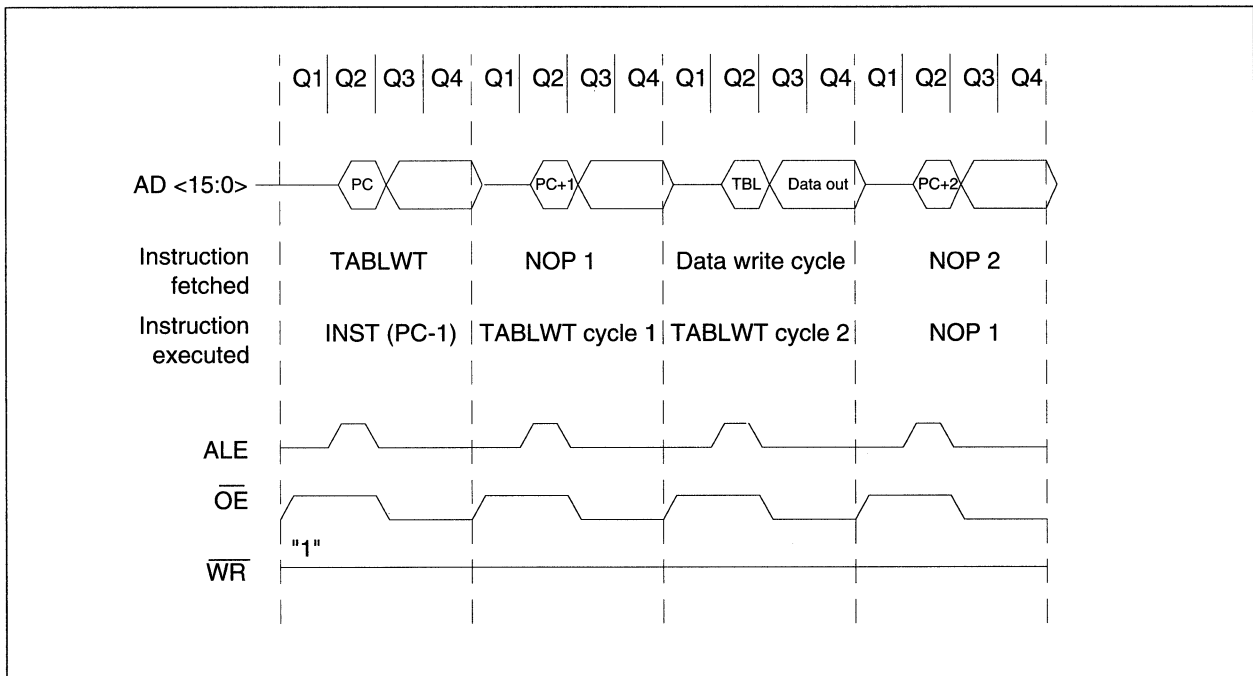


FIGURE 12.4.8: TABLWT TIMING (CONSECUTIVE TABLWT INSTRUCTIONS)

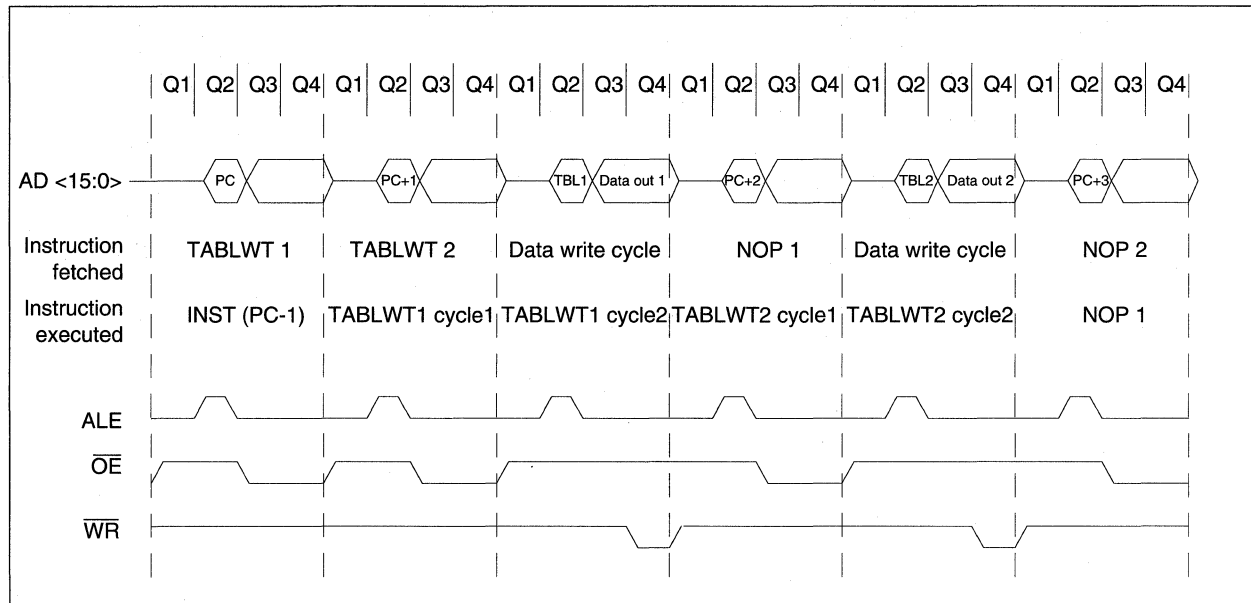


FIGURE 12.4.9: SLEEP/WAKE-UP THROUGH INT (LF, XT MODES)

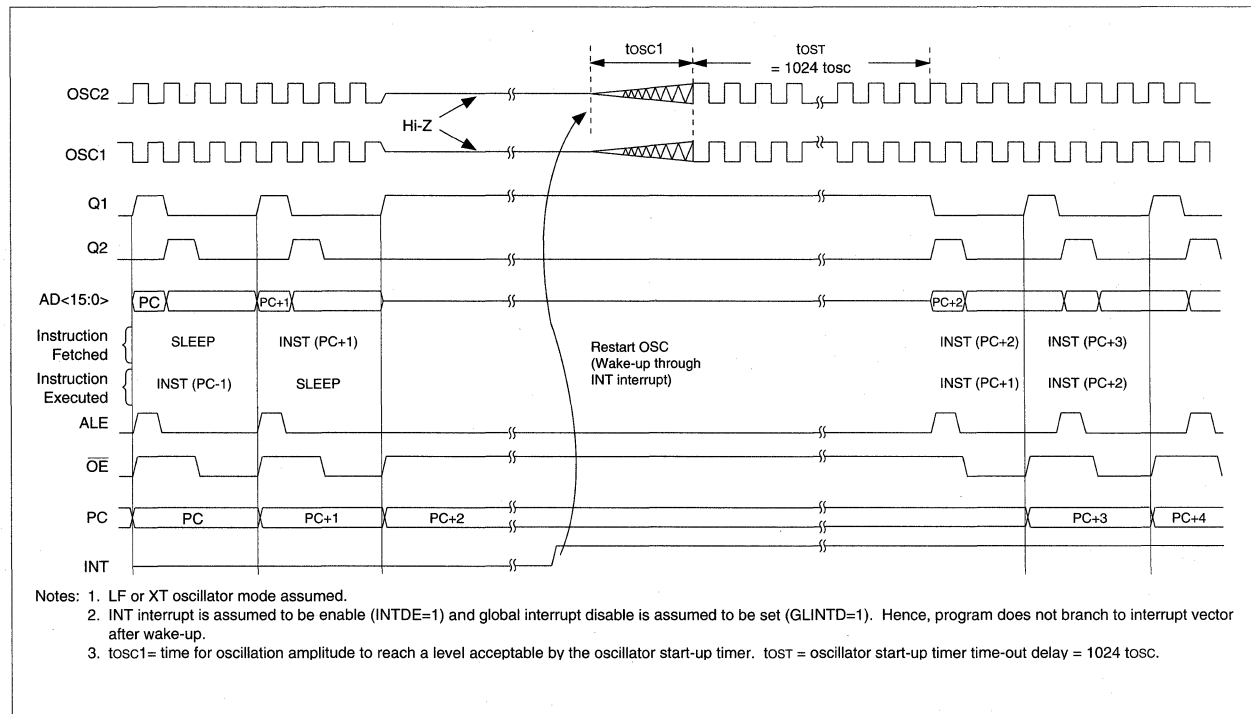


FIGURE 12.4.10: SLEEP/WAKE-UP THROUGH INT (RC MODE)

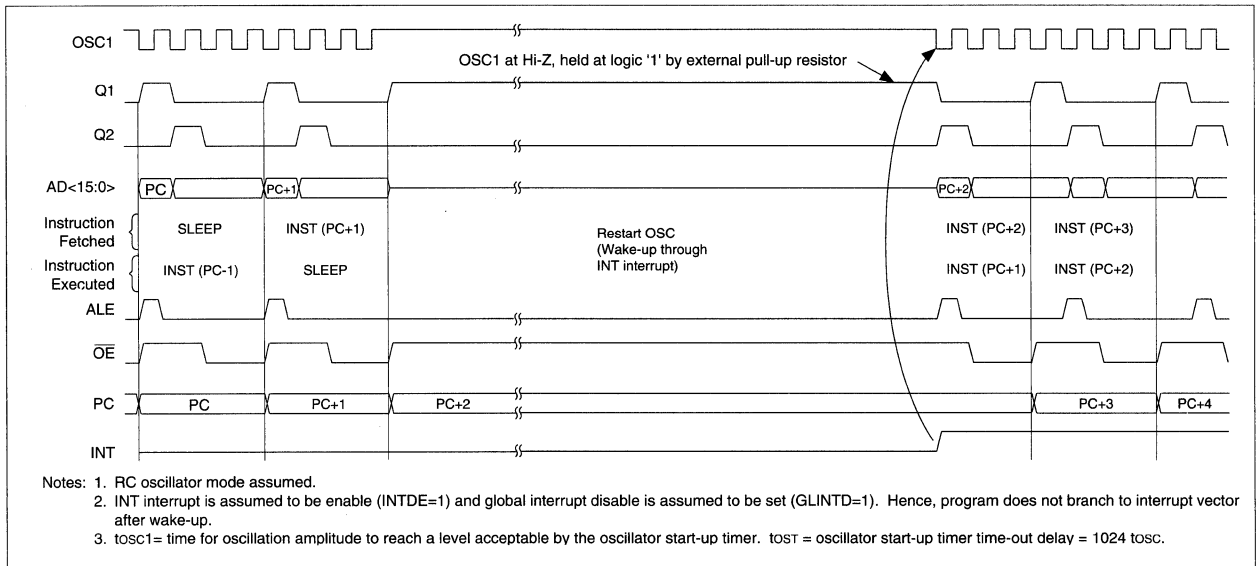


FIGURE 12.4.11 SYNCHRONOUS TRANSMISSION (MASTER/SLAVE)

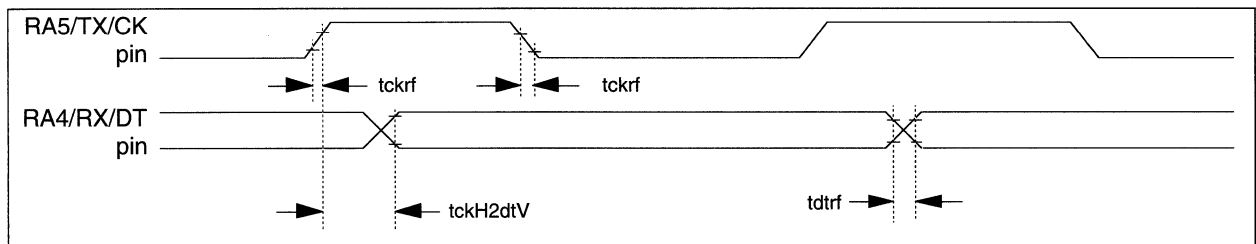


FIGURE 12.4.12 SYNCHRONOUS RECEIVE (MASTER/SLAVE)

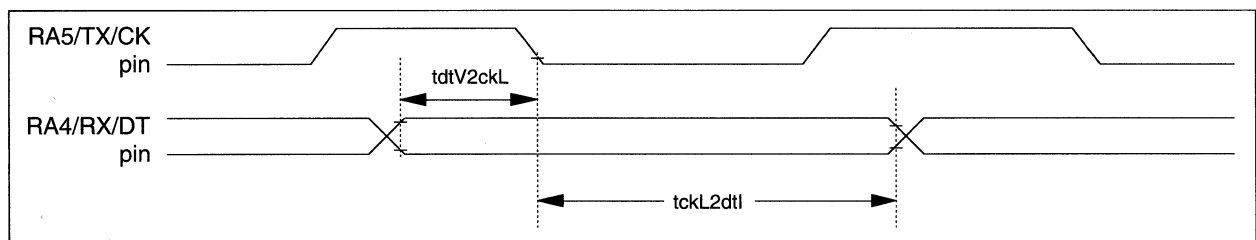
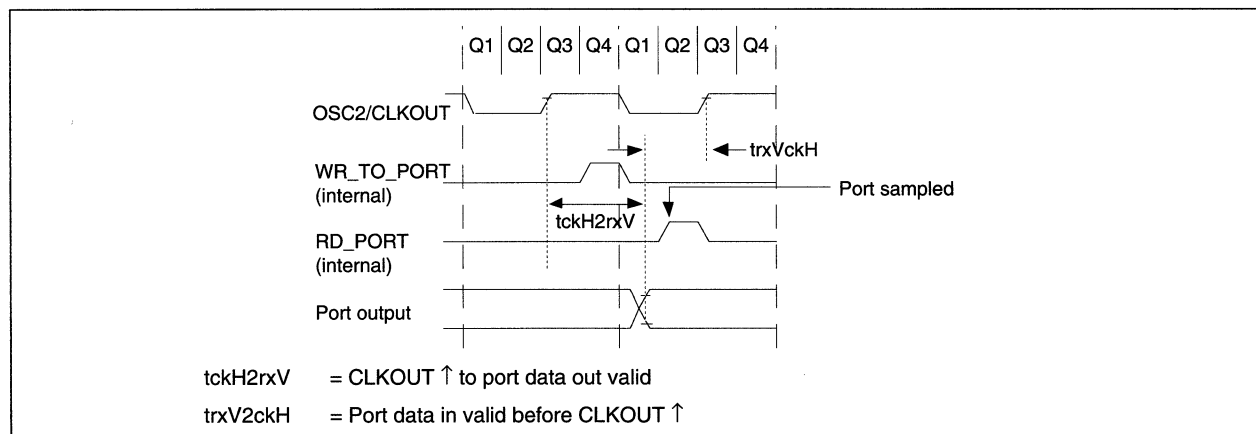
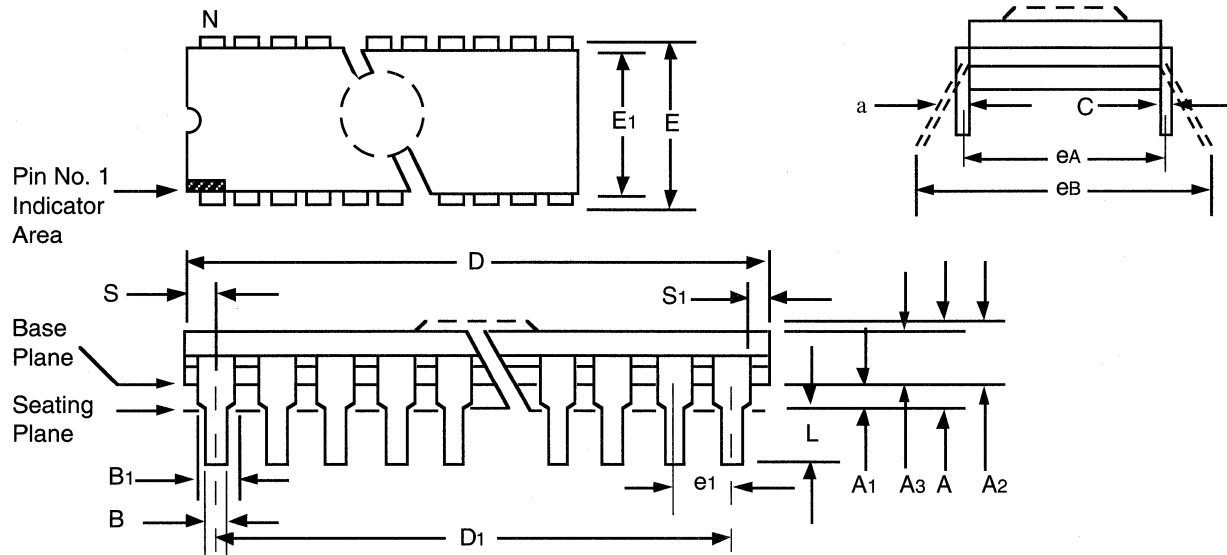


FIGURE 12.4.13: I/O PORT INPUT/OUTPUT TIMING (PORTA, PORTB)



13.0 PACKAGING INFORMATION

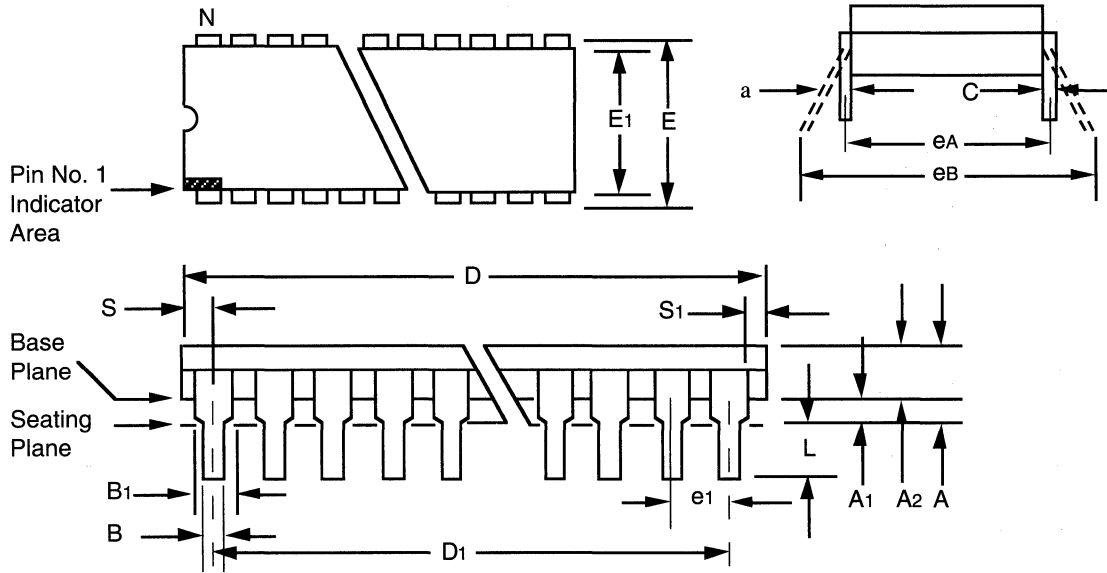
13.1 PACKAGE TYPE: 40-LEAD CERAMIC CERDIP DUAL IN-LINE WITH WINDOW (.600 MIL)



Package Group: Ceramic Cerdip Dual In-line (CDP)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
α	0°	10°		0°	10°	
A	4.318	5.715		0.170	0.225	
A1	0.381	1.778		0.015	0.070	
A2	3.810	4.699	Ref. A3	0.150	0.185	Ref. A3
A3	3.810	4.445		0.150	0.175	
B	0.356	0.584		0.014	0.023	
B1	1.270	1.651	Typical	0.050	0.065	Typical
C	0.203	0.381	Typical	0.008	0.015	Typical
D	51.435	52.705		2.025	2.075	
D1	48.260	48.260	Reference	1.900	1.900	Reference
E	15.240	15.875		0.600	0.625	
E1	12.954	15.240		0.510	0.600	
e1	2.540	2.540	Typical	0.100	0.100	Typical
eA	14.986	16.002	Reference	0.590	0.630	Reference
eB	15.240	18.034		0.600	0.710	
L	3.175	3.810		0.125	0.150	
N	40	40		40	40	
S	1.016	2.286		0.040	0.090	
S1	0.381	1.778		0.015	0.070	

13.0 PACKAGING INFORMATION (CONT.)

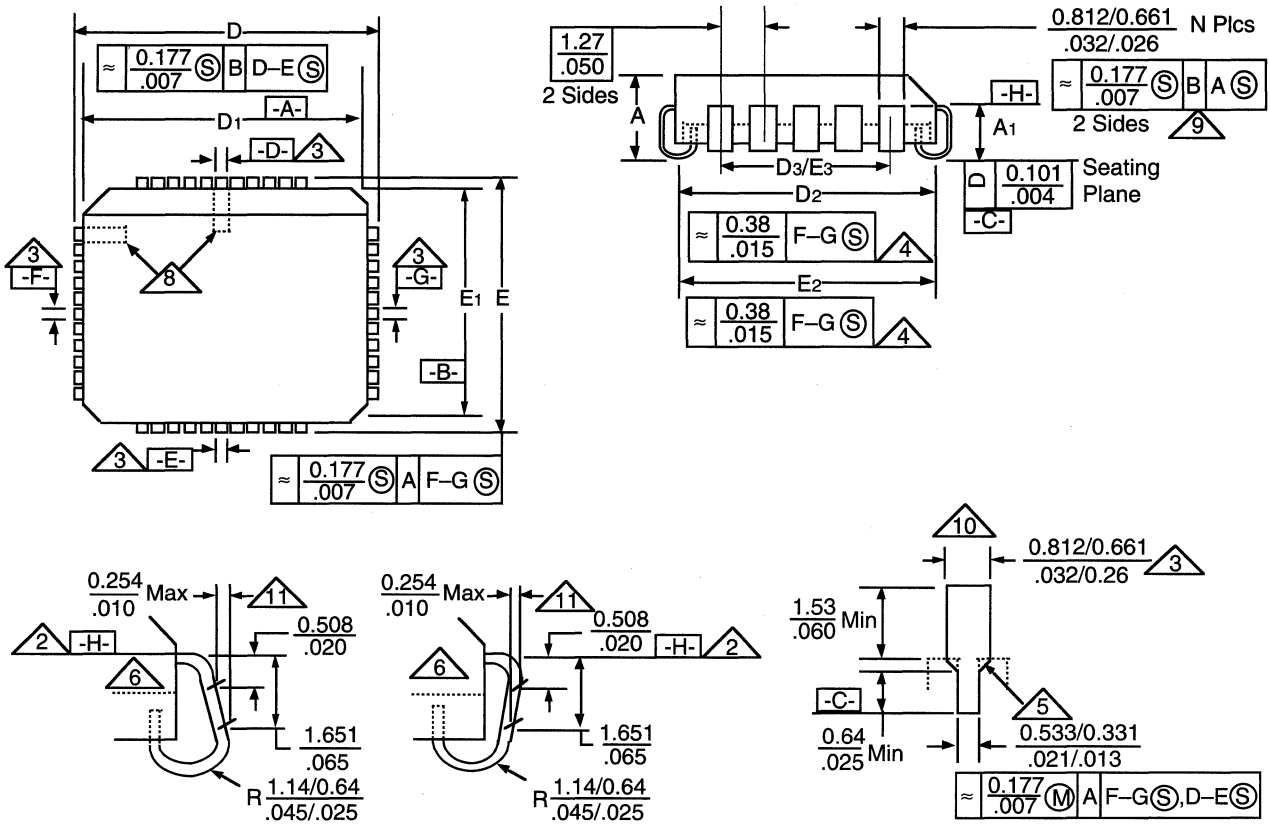
13.2 PACKAGE TYPE: 40-LEAD PLASTIC DUAL IN-LINE (.600 MIL)



Package Group: Plastic Dual In-line (PLA)						
Symbol	Millimeters			Inches		
	Min		Notes	Min	Max	Notes
α	0°			0°	10°	
A	—			—	0.200	
A1	0.381			0.015	—	
A2	3.175			0.125	0.160	
B	0.356			0.014	0.022	
B1	1.270		Typical	0.050	0.070	Typical
C	0.2032		Typical	0.008	0.015	Typical
D	51.181			2.015	2.055	
D1	48.260		Reference	1.900	1.900	Reference
E	15.240			0.600	0.625	
E1	13.462			0.530	0.550	
e1	2.489		Typical	0.098	0.102	Typical
eA	15.240		Reference	0.600	0.600	Reference
eB	15.240			0.600	0.680	
L	2.921			0.115	0.145	
N	40			40	40	
S	1.270			0.050	—	
S1	0.508			0.020	—	

13.0 PACKAGING INFORMATION (CONT.)

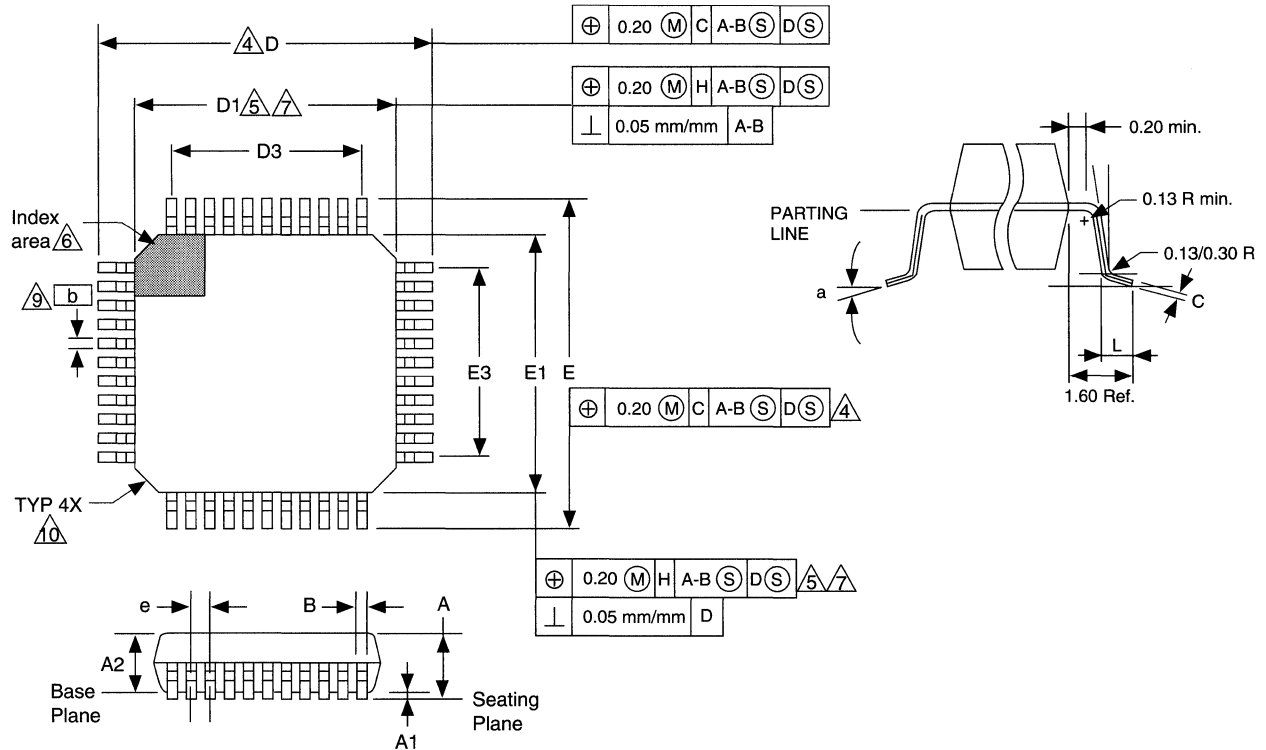
13.3 PACKAGE TYPE: 44-LEAD PLASTIC LEADED CHIP CARRIER (SQUARE)



Package Group: Plastic Leaded Chip Carrier (PLCC)						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	4.191	4.572		0.165	0.180	
A1	2.413	2.921		0.095	0.115	
D	17.399	17.653		0.685	0.695	
D1	16.510	16.662		0.650	0.656	
D2	15.494	16.002		0.610	0.630	
D3	12.700	12.700	Reference	0.500	0.500	Reference
E	17.399	17.653		0.685	0.695	
E1	16.510	16.662		0.650	0.656	
E2	15.494	16.002		0.610	0.630	
E3	12.700	12.700	Reference	0.500	0.500	Reference
N	44	44		44	44	
CP	-	0.1016		-	0.004	
LT	0.203	0.381		0.008	0.015	

13.0 PACKAGING INFORMATION (CONT.)

13.4 PACKAGE TYPE: 44-LEAD METRIC PLASTIC QUAD FINE PITCH
(MQFP 10X10MM BODY 1.6/.015MM LEAD FORM)



Package Group: Plastic MQFP						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
α	0°	7°		0°	7°	
A	2.00	2.35		0.0787	0.0925	
A ₁	0.05	0.25		0.0019	0.0098	
A ₂	1.95	2.10		0.768	0.0827	
b	0.30	0.45	Typical	0.0118	0.0177	Typical
C	0.15	0.18		.006	.007	
D	12.95	13.45		0.510	0.530	
D ₁	9.90	10.10		0.390	0.398	
D ₃	8.00	8.00	Reference	0.315	0.315	Reference
E	12.95	13.45		0.510	0.530	
E ₁	9.90	10.10		0.390	0.398	
E ₃	8.00	8.00	Reference	.315	.315	Reference
e	0.80	0.80		.0314	.0314	
L	0.65	0.95		.0256	.0374	
N	44	44		44	44	
CP	0.102			.004		

(Notes on following page)

Symbol List for Metric Plastic Quad Flat Pack Package Parameters	
Symbol	Description of Parameters
α	Angular spacing between min and max lead positions measured at the gauge plane
A	Distance between seating plane to highest point of body
A ₁	Distance between seating plane and base plane
A ₂	Distance from base plane to highest point of body
b	Width of terminals
C	Thickness of terminals
D ₁ /E ₁	Largest overall package parameter including leads
D/E	Largest overall package parameter including leads
D ₃ /E ₃	Center of end lead to center of end lead
e	Linear spacing of true minimum lead position center line to center line
L	Length of terminal for soldering to a substrate
N	Total number of potentially useable lead positions
CP	Seating plane coplanarity

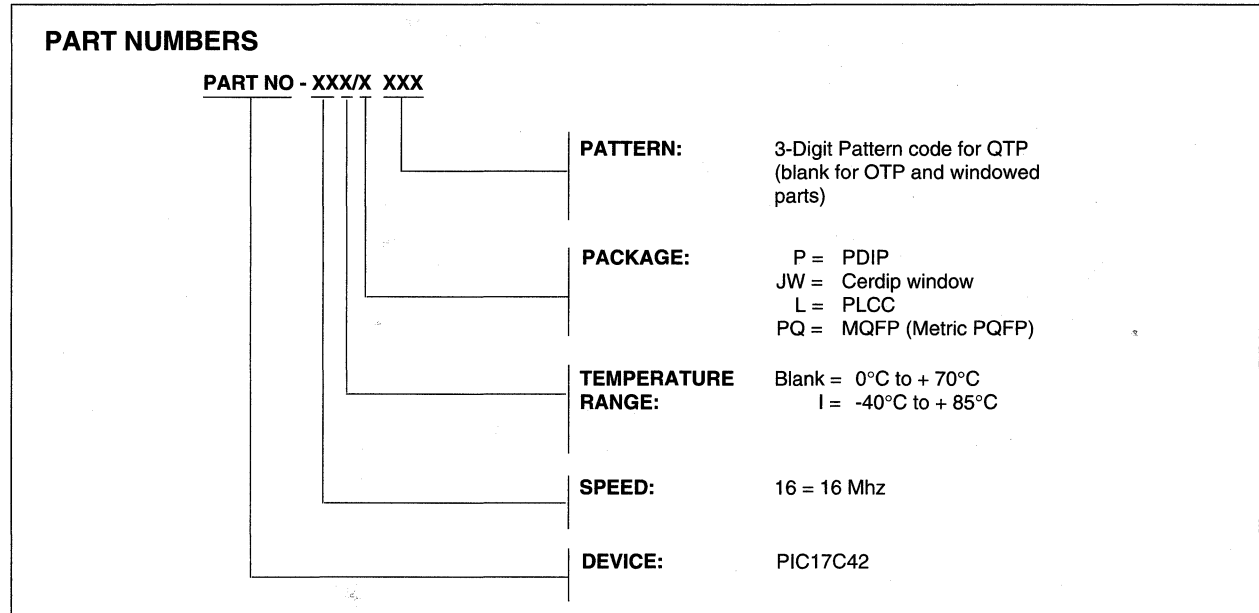
Notes

1. All dimensioning and tolerancing conform to ANSI Y14, BM-1582.
2. Datum Plane **-H-** is located at bottom of hold parting line and coincident with bottom of lead, where lead exits body.
3. Datums **A-B** and **-D-** to be determined at Datum plane **-H-**.
4. To be determined at seating plane **-C-**.
5. Dimensions D1 and E1 do not include hold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 do not include hold mismatch and are determined at Datum Plane **-H-**.
6. Details of pin 1 identifier are optional but must be located within the zone indicated.
7. These dimensions to be determined at Datum plane **-H-**.
8. All dimensions in millimeters.
9. Dimension b does not include Dambar protrusion. Allowable Dambar protrusion shall be 0.08mm total in excess of the b dimension at maximum material condition. Dambar cannot be located on the lower radius or the lead foot.
10. Exact shape of this feature is optional.
11. N is the number of leads.
12. Controlling parameters: millimeters
13. All packages are gull wing lead form.

Notes:

SALES AND SUPPORT

To order or to obtain information, e.g., on pricing or delivery, please use the listed part numbers, and refer to the factory or the listed sales offices.



Microchip Technology Incorporated

Printed in USA © 9207

ASIA / PACIFIC

Hong Kong

Microchip/Excel Associates Ltd.
Unit No. 2520-2525
Tower 1, Metroplaza
Hing Fong Road, Kwai Fong
N.T., Hong Kong
Tel: 852 418 0909 Fax: 852 418 1600

Japan

Microchip Technology Ltd.
Shinyokohama Gotoh Bldg. 8F, 3-22-4
Shinyokohama, Kohoku-Ku, Yokohama-Shi
Kanagawa 222 Japan
Tel: 45/471 6166 Fax: 45/471 6122

Korea

Microchip/Excel Associates Korea
4 Floor Myungdang Bldg.
Samsung-Dong
Kanghnam-Ku
Seoul, Korea
Tel: 82 2 563 5277 Fax: 82 2 563 5279

Singapore

Microchip/Excel Associates Ltd.
Singapore Representative Office
10 Anson Road #14-02
International Plaza
Singapore 0207
Tel: 65 222 4962 Fax: 65 222 4939

Taiwan

Microchip/Excel Associates Taiwan
1037 Min Sheng East Road
14FL-1, Hai Hwa Bldg.
Taipei, Taiwan
Tel: 886 2 7602028 Fax: 886 2 7651488

EUROPE

United Kingdom

Arizona Microchip Technology LTD.
Unit 3, Meadow Bank, Furlong Road
Bourne End, Bucks SL8 5AJ
Tel: 62-885-0303 Fax: 62-885-0178

Germany

Arizona Microchip Technology GMBH
Alte Landstrasse 12-14
D-8012 Ottobrunn, Germany
Tel: 089 609 6072 Fax: 089 609 1997

France

Arizona Microchip Technology SARL
2, Rue Du Buisson aux Fraises
F-91300 Massy, France
Tel: 1 69 30 90 90 Fax: 1 69 30 90 79

UNITED STATES

Corporate Office

Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 963-7373 Fax: 602 899-9210

North-East Region

Microchip Technology Inc.
Five The Mountain Road, Suite 120
Framingham, MA 01701
Tel: 508 820-3334 Fax: 508 820-4326

Mid-Atlantic Region

Microchip Technology Inc.
300 Wheeler Road, Suite 206
Hauppauge, NY 11788
Tel: 516 232-1930 Fax: 516 232-1935

South-East Region

Microchip Technology Inc.
1521 Johnson Ferry Road NE
Suite 170
Marietta, GA 30062
Tel: 404 509-8800 Fax: 404 509-8600

North-Central Region

Microchip Technology Inc.
665 Tollgate Road, Unit C
Elgin, IL 60123-9312
Tel: 708 741-0171 Fax: 708 741-0638

South-Central Region

Microchip Technology Inc.
17480 N Dallas Parkway, Suite 114
Dallas, TX 75287
Tel: 214 733-0391 Fax: 214 250-4631

North-West Region

Microchip Technology Inc.
2107 N First Street, Suite 410
San Jose, CA 95131
Tel: 408 436-7950 Fax: 408 436-7955

South-West Region

Microchip Technology Inc.
1411 W 190th Street, Suite 230
Gardena, CA 90248-4307
Tel: 310 323-1888 Fax: 310 323-1424



Microchip

"Information contained in this publication regarding device applications and the like is intended by way of suggestion only. No representation or warranty is given and no liability is assumed by Microchip Technology Inc. with respect to the accuracy or use of such information. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. The Microchip logo and name is a registered trademark of Microchip Technology Incorporated. PICPRO, PIC-ICE, PICPAK and PICMASTER are trademarks of Microchip Technology Inc. IBM PC is a trademark of IBM Corporation. All rights reserved."