DEVELOPING APPLICATIONS AROUND THE PIC ARCHITECTURE

PART 2

Pickled Processors

by Stephen Waddington

Last month, we took a first look at the elements of a generic microprocessor. Here, Stephen Waddington talks through the basic characteristics of the PIC microprocessor

The PIC microprocessor family is available in three key flavours of increasing processing capability, as shown along the performance axis in Figure 1. These range from the low-end 20MHz PIC16C5X, typically with 12 I/O lines, a 12-bit instruction set and no interrupt resources, through to the 25MHz PC17CXX with up to 33 I/O lines, a 16-bit instruction set and up to 11 interrupts.

Figure 1 also shows another type of segmentation, in terms of local memory technology ranging from conventional ROM through to EPROM (Erasable Programmable Read Only Memory) and EEPROM (Electrically Erasable Programmable Read Only Memory). Pure ROM-based devices are the least versatile, since they can only be programmed once. By contrast, EEPROM based devices can be repeatedly programmed and re-programmed with ease.

The conventional way to develop a microprocessor-based system, is to determine the scope of the application and then select a microprocessor capable of handling the job. In many ways, we decided to turn this process on its head when we opted to focus purely on PIC-based microprocessors, rather than taking a wider outlook. The bad news for purists is that we're going to go one stage further. In order to get to grips with developing PIC systems from the ground up, this article will focus on a single member of the PIC family – the PIC16C84. Table 1 shows the key characteristics of this device, while Figure 2 shows a pin-out diagram.

CPU Features

- ◆ Only 35 instructions to learn
- ◆ All instructions are single cycle (400ns) except for program branches which are two cycle
- ◆ 10MHz clock
- ◆ 14-bit wide instructions
- ◆ 1,024 × 14 on-chip EEPROM program memory
- ◆ 36 × 8 general purpose registers (SRAM)
- ◆ 15 special function hardware registers
- ◆ 64 × 8 EEPROM data memory
- ◆ Eight level deep hardware stack
- ◆ Direct, indirect and relative addressing modes
- ◆ Multiple interrupt sources
- ◆ 1,000,000 ERASE/WRITE cycles
- ◆ Data retention greater than 40 years

Peripheral Features

- ◆ 13 I/O pins with individual direction control
- ◆ High current (25mA per pin) for direct LED drive
- ◆ 8-bit real time clock/counter with 8-bit programmable pre-scaler

Microprocessor Features

- ◆ Power-On Reset
- ◆ Power-up Timer
- ◆ Oscillator Start-up Timer
- ◆ Self clocked Watchdog Timer (WDT)
- ◆ Security EEPROM fuse for code protection
- ◆ Power saving sleep mode
- ◆ User selectable oscillator options
- ◆ Serial in-system programming capability

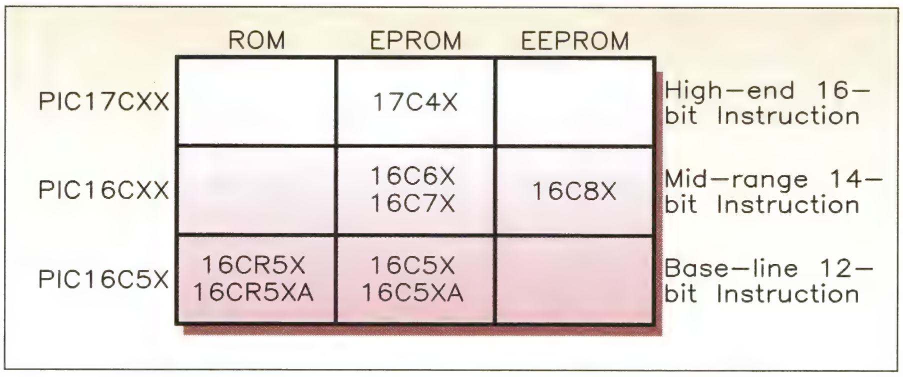
Electrical Characteristics

- ◆ Low power high speed CMOS technology
- ◆ Fully static design
- ◆ Wide operation voltage range (2·0 to 6·0V)
- ◆ Low power consumption: 2mA at 5V, 4MHz; 15μA typically at 2V

Table 1. Key characteristics of the PIC16C84.

However, it's not all bad news. The PIC16C84 is an excellent device to learn the basics of microprocessor and PIC development. It uses the standard PIC instruction set of 33 instructions, with two additions to address registers which are unique to the device. This means that once you've got to grips with programming the PIC16C84, you'll be able to work with any of the other devices in the PIC family. Perhaps even more attractive is the fact that it has 1K of local EEPROM. This makes it an ideal device to learn techniques on, since it can literally be programmed and re-programmed within 20 seconds. Figure 3 shows a block diagram of the PIC16C84.

For performance details of other members of the PIC family in comparison to the PIC16C84, check the Microchip Databook – see further reading at the end of this article. In later parts of this series, we will take a look in greater detail at other members of the PIC family.



ROM
EPROM
EEPROM
PIC17CXX
17C4X
High–end 16–bit Instruction
PIC16CXX
16C6X
16C7X
16C8X
Mid–range 14–bit Instruction
PIC16C5X
16CR5X
16CR5XA
16C5X
16C5XA
Base–line 12–bit Instruction

Figure 1. PIC 16/17 microcontroller families.

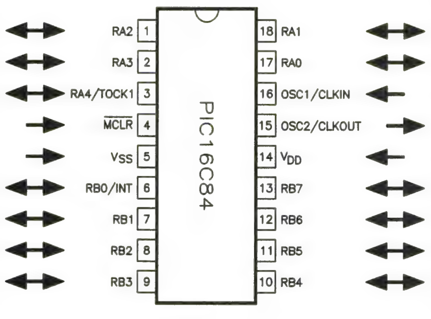


Figure 2. Pin-out diagram for PIC16C84.

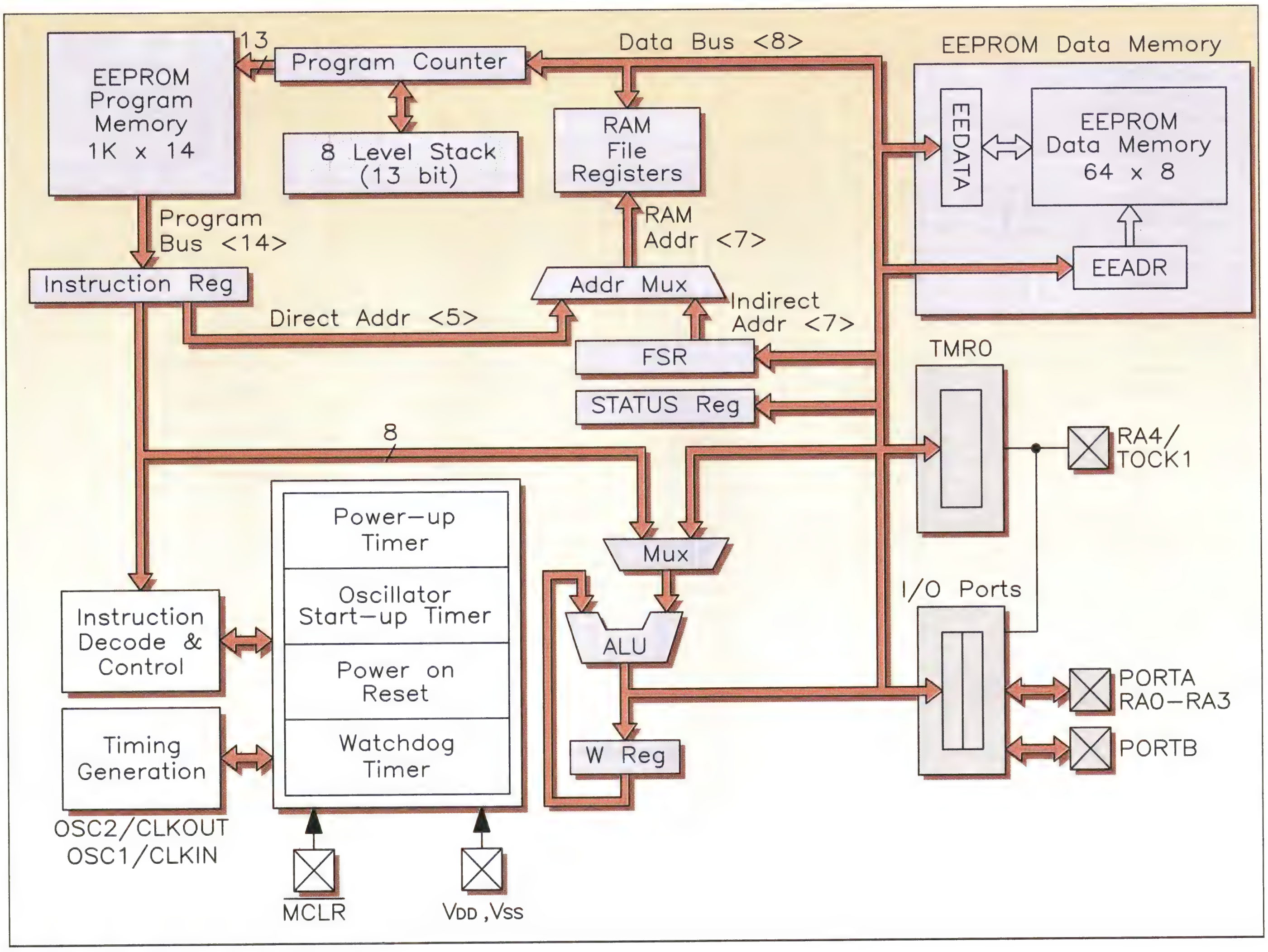
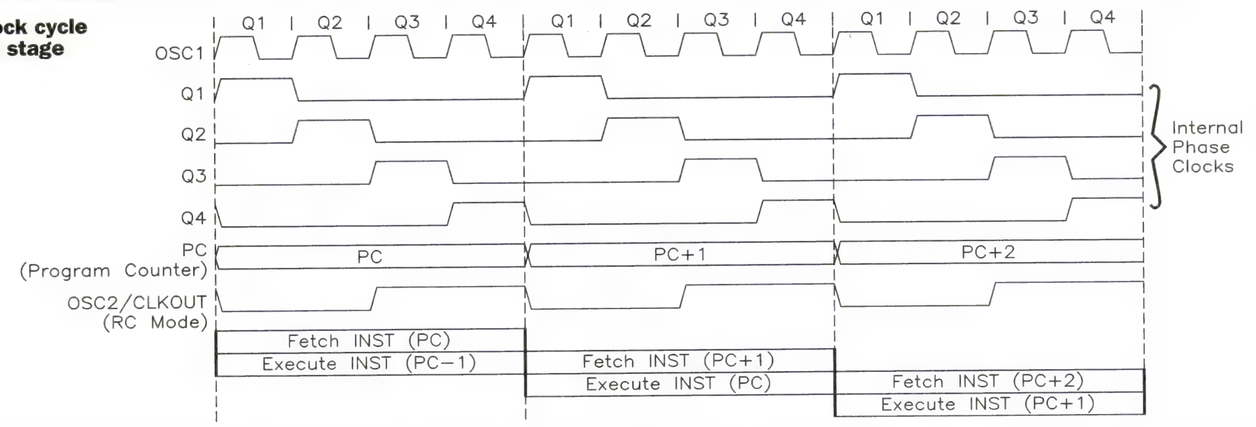


Figure 3. Block diagram of PIC16C84.

Figure 4. Clock cycle showing two stage pipeline.



Architectural Overview

The PIC16C84 is equipped with special features such as an internal clock, sleep function, and software watchdog to reduce external components and thus cost, while enhancing system reliability and power consumption. There are four oscillator options which can be optimised by the designer for cost or power savings. The power down or SLEEP mode enables the microprocessor to be shut down when not in use, providing power savings. The user can wake up the chip from SLEEP using external interrupts or via a reset function. Meanwhile, a watchdog timer with its own on-chip RC oscillator provides protection against software malfunction.

The PIC family is based on the Harvard architecture, which means programme and data are held in separate memories. This improves the performance of the device over the Von Neuman approach, where memory is shared. It also means that the instruction length or op-code is not tied to the length of the data word. Op-codes can be greater than the 8-bit data word because they occupy their own memory.

Clock/Instruction Cycle

In the PIC16C84, all op-codes are 14-bit wide. On-chip, 1K × 14-bit is reserved for program memory. A 14-bit wide program memory means that all instructions can be fetched in a single cycle. A two stage pipeline, as shown in Figure 4, is used to overlap the fetch, decode and execute cycle. This means that all instructions execute with a single clock cycle, with the exception of program jumps or branches.

The PIC16C84 can directly or indirectly address its 48 register files or data memory. All special function registers including the program counter are mapped in the data memory. The instruction set is fairly orthogonal, which 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 PIC16C84 simplistic, yet efficient.

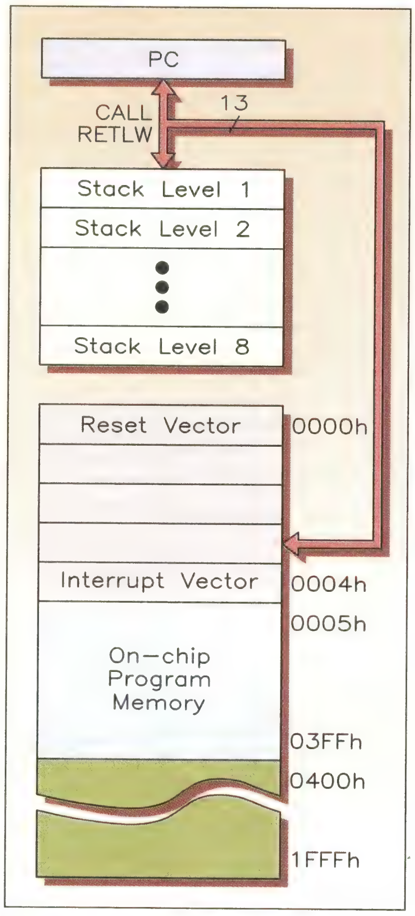


Figure 5. Program memory map and stack.

The clock input is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4, as shown in Figure 4. Internally, the program counter is incremented every Q1. An instruction is fetched from the program memory and latched into the instruction register in Q4.

An Instruction Cycle consists of Q1, Q2, Q3 and Q4 cycles. The Instruction fetch and execute cycles are pipelined such that a fetch takes one instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change, such as GOTO or JUMP, then two cycles are required to complete the instruction. A fetch cycle begins with the program counter (PC) incrementing in Q1. The fetched instruction is latched into the Instruction Register (IR) which is decoded and executed during Q2, Q3 and Q4. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

Memory Organisation

The PIC16C84 has a 13-bit wide program counter, capable of addressing an 8K × 14-bit program memory space, as shown in Figure 5. Only the first 1K × 14-bit (0000h to 03FFh)* are physically implemented. Accessing a location above 03FFh will cause a wraparound within the first 1K × 14-bit space. The reset vector is at 0000h and the interrupt vector is at 0004h.

* xxxxh refers to a memory address in hexadecimal.

The EEPROM program memory of the PIC16C84 is rated for limited erase/write cycles. To program the program memory, the device must be put into a special mode by raising MCLR pin to high voltage. Also, V_{DD} must be 4·5 to 5·5V during programming. While this feature means the PIC16C84 is not suitable for applications where program memory is updated in the user application frequently, it does make it ideal for the design development.

Program Counter

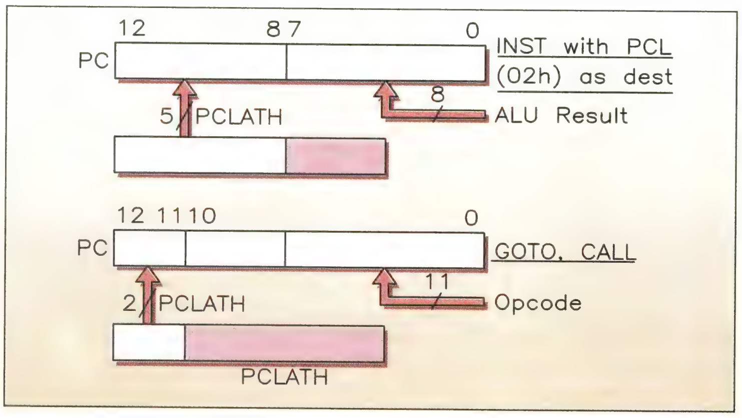
The program memory can be programmed serially using two data/clock pins, which makes in-system programming possible. This enables the designer to customise the system during final testing or upgrade a system in location.

The program counter (PC) is shown in Figure 6. The program counter low byte (PCL) is a readable and writable register. The high byte of the program counter (PCH) is not directly addressable or writable. However, it can be written through the PCLATH register (0Ah) when the PC is loaded with a new value during a CALL, GOTO or write to PCL.

Stack

The PIC16C84 has an 8 × 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The value of the program counter is placed on the stack when a CALL instruction is executed or an interrupt is acknowledged. The stack is cleared in the event of a RETURN, RETLW, or RETFIE instruction execution.

Figure 6. Program counter organisation.



Registers

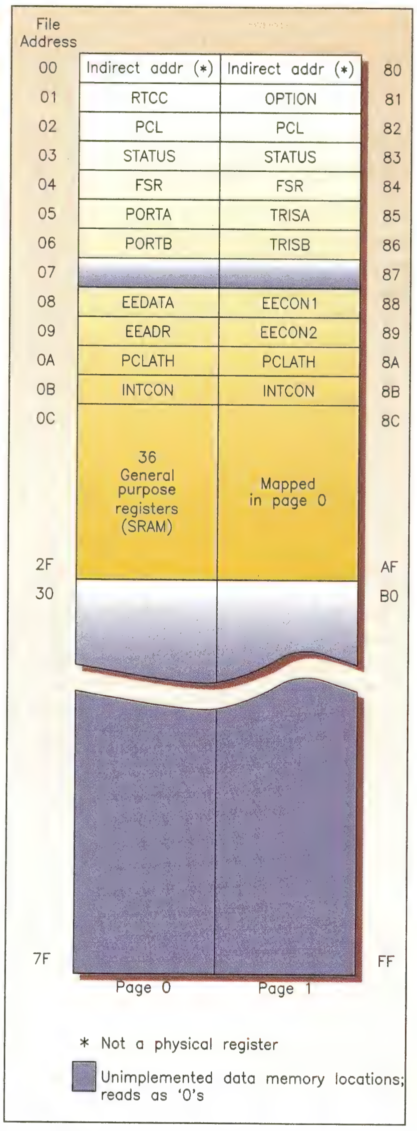
The register file is organised as a 128 × 8-bit block, as shown in Figure 7. It is accessed either directly or indirectly through the file select register (FSR). It is also referred to as the data memory. There are several register file page select bits in the STATUS register allowing up to four pages. However, data memory extends only up to 2Fh. The first 12 locations are used to map special function registers. Locations 0Ch to 2Fh are general-purpose registers implemented as static RAM. Some special function registers are mapped in page 1. When in page 1, accessing locations 8Ch to AFh will enable access the RAM in page 0.

There are two possible ways to address the register file, either directly using a physical number to represent an address, or indirectly using the result of a mathematical operation or by referencing the contents of another register. In both modes, up to 512 register locations can be addressed:

Direct addressing mode

An effective 9-bit address is obtained by concatenating 7-bits of direct address from the op-code and two bits (RP1, RP0) from the status register, as shown in Figure 8.

Figure 7. Register file map.



File Address	Page 0	Page 1	
00	Indirect addr (*)	Indirect addr (*)	80
01	RTCC	OPTION	81
02	PCL	PCL	82
03	STATUS	STATUS	83
04	FSR	FSR	84
05	PORTA	TRISA	85
06	PORTB	TRISB	86
07			87
08	EEDATA	EECON1	88
09	EEADR	EECON2	89
0A	PCLATH	PCLATH	8A
0B	INTCON	INTCON	8B
0C–2F	36 General purpose registers (SRAM)	Mapped in page 0	8C–AF
30–7F			B0–FF

* Not a physical register

▨ Unimplemented data memory locations; reads as '0's

ndirect addressing mode

ndirect addressing is achieved by using the le address 00h (INDF). Any instruction sing INDF as file register actually accesses lata pointed to by the file select register (FSR). Reading INDF directly will produce 00h. Writing to indirectly results in no operation, lthough the status bits may be affected. n effective 9-bit address is obtained by concatenating the 8-bit FSR from the status egister, as shown in Figure 8.

Event	TO	PD
Power-up	1	1
WDT time-out	0	Unchanged
SLEEP instruction	1	0
CLR WDT instruction	1	1

Table 2. Events affecting $\overline{TO}/\overline{PD}$ status bits.

Reset Event	TO	PD
WDT wake-up from sleep	0	0
WDT time-out	0	1
MCLR wake-up from sleep	Unchanged	0
Power-up	1	1
MCLR reset during normal operation	Unchanged	U

Table 3. $\overline{TO}/\overline{PD}$ status after reset.

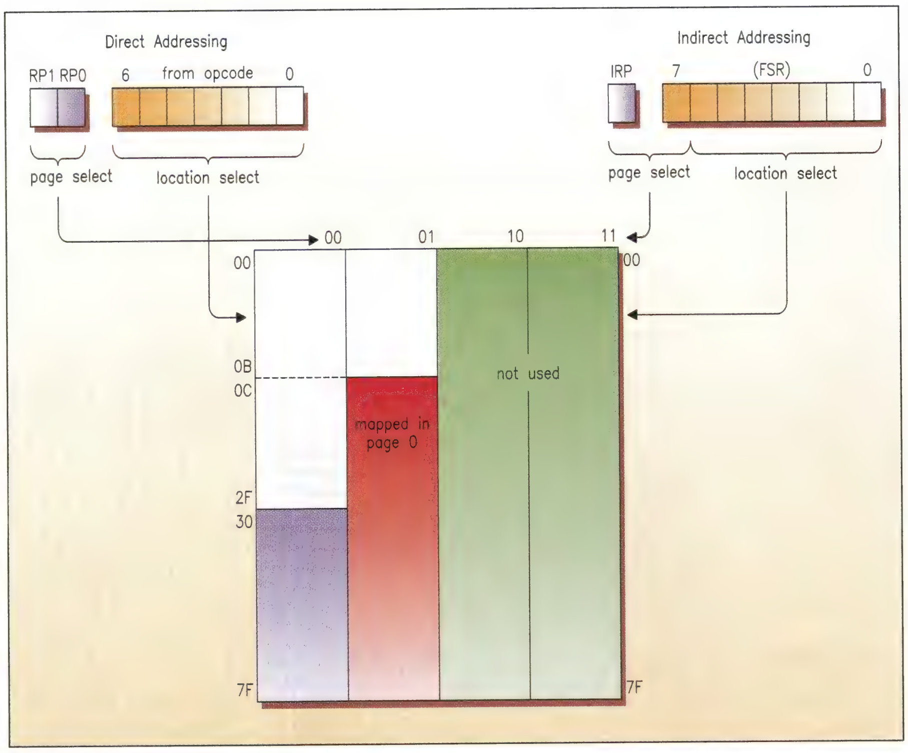
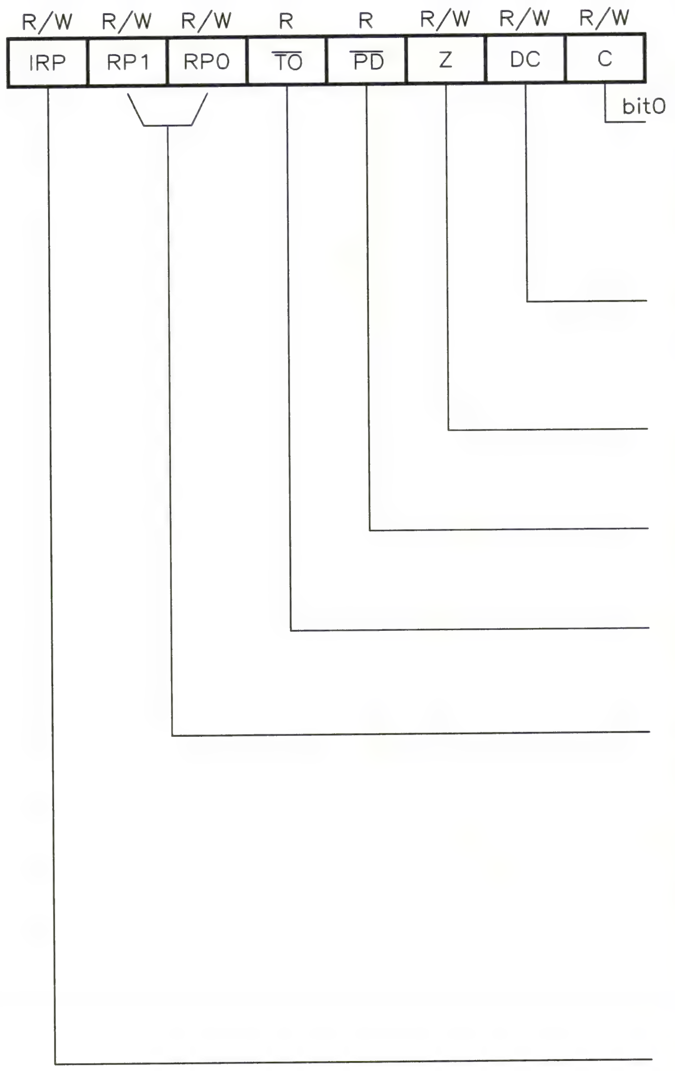


Figure 8. Direct and indirect addressing.



ADDRESS: 03h
RESET CONDITION: 000??XXX

CARRY/$\overline{\text{BORROW}}$ BIT:
For ADDWF, SUBWF, ADDLW and SUBLW instructions, this bit is set if there is a carry out from the most significant bit of the resultant. Note that a subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

DIGIT CARRY/$\overline{\text{BORROW}}$ BIT:
For ADDWF, SUBWF, ADDLW and SUBLW instructions, this bit is set if there is a carry out from the 4th low order bit of the resultant.

ZERO BIT:
Set if the result of an arithmetic or logic operation is zero.
Reset otherwise. Additionally, MOVF instruction will affect the Z bit.

POWER DOWN BIT:
Set to "1" during power up or by a CLRWDT command.
This bit is cleared to "0" by a SLEEP instruction.

TIME-OUT BIT:
Set to "1" during power up and by the CLRWDT and SLEEP command.
This bit is cleared to "0" by a watchdog timer time out.

REGISTER PAGE SELECT BITS FOR DIRECT ADDRESSING:

RP1,0 :	00 :	page 0 (00h-7Fh)
	01 :	page 1 (80h-FFh)
	10 :	page 2 (100h-17Fh)
	11 :	page 3 (180h-1FFh)

Each page is 128 bytes.
Only RP0 is useful in PIC16C84. Bit RP1 can be used as a general purpose read/write bit. However, this may affect upward compatibility with future products.

REGISTER PAGE SELECT BITS FOR INDIRECT ADDRESSING:

IRP0 :	0 :	page 0,1 (00h-FFh)
	1 :	page 2,3 (100h-1FFh)

This bit is effectively not used in the PIC16C84.

Figure 9. Status register.

Status Register

The STATUS register shown in Figure 9 contains the arithmetic status of the ALU, the RESET status, and the page preselect bits for data memory. Like any other register, the STATUS register can be the destination for any instruction. However, the status bits are set following the write operation (Q4). The and bits are not writable, and therefore, the result of an instruction with the STATUS register as its destination may be different than necessarily intended. For example, the command, CLRF STATUS will clear all bits except and and then set the z bit, leaving the STATUS register as 000XX100 – where X represents a bit unchanged. Tables 2 and 3 show how the and status bits change following a set or reset.

Microchip recommend that only BCF, BSF and MOVWF instructions are used to alter the STATUS registers, because these instructions do not affect any status bit.

Carry/Borrow Bits

The carry bit is a carry out in addition operations such as ADDWF and ADDLW, and a borrow out in subtract operations such as SUBWF and SUBLW. The digit carry operates in the same way as the carry bit. n this case, however, it is a borrow in subtract operations.

Arithmetic Logic Unit

The Arithmetic Logic Unit (ALU) is 8-bits wide and capable of addition, subtraction, shift and logical operations. Arithmetic operations are generally two's complement in nature. In two-operand instructions, typically, one operand is the working register (W register) or the accumulator. The other operand is a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

The W register is an 8-bit working register or accumulator used for ALU operations. It is not in the data memory.

Interrupts

The PIC16C84 has four sources of interrupt:

- External interrupt from RB0/INT pin.
- TMR0 timer/counter overflow interrupt.
- End of data EEPROM write.
- Interrupt on change on RB.

The interrupt control register (INTCON, addr0Bh), shown in Figure 10, records individual interrupt requests in flag bits. It also has individual local and global interrupt enable bits. A global interrupt enable (GIE) bit enables – if set – all unmasked interrupts or disables – if cleared – all interrupts. Individual interrupts can be disabled through their corresponding enable bits in the INTCON register. GIE is cleared on reset.

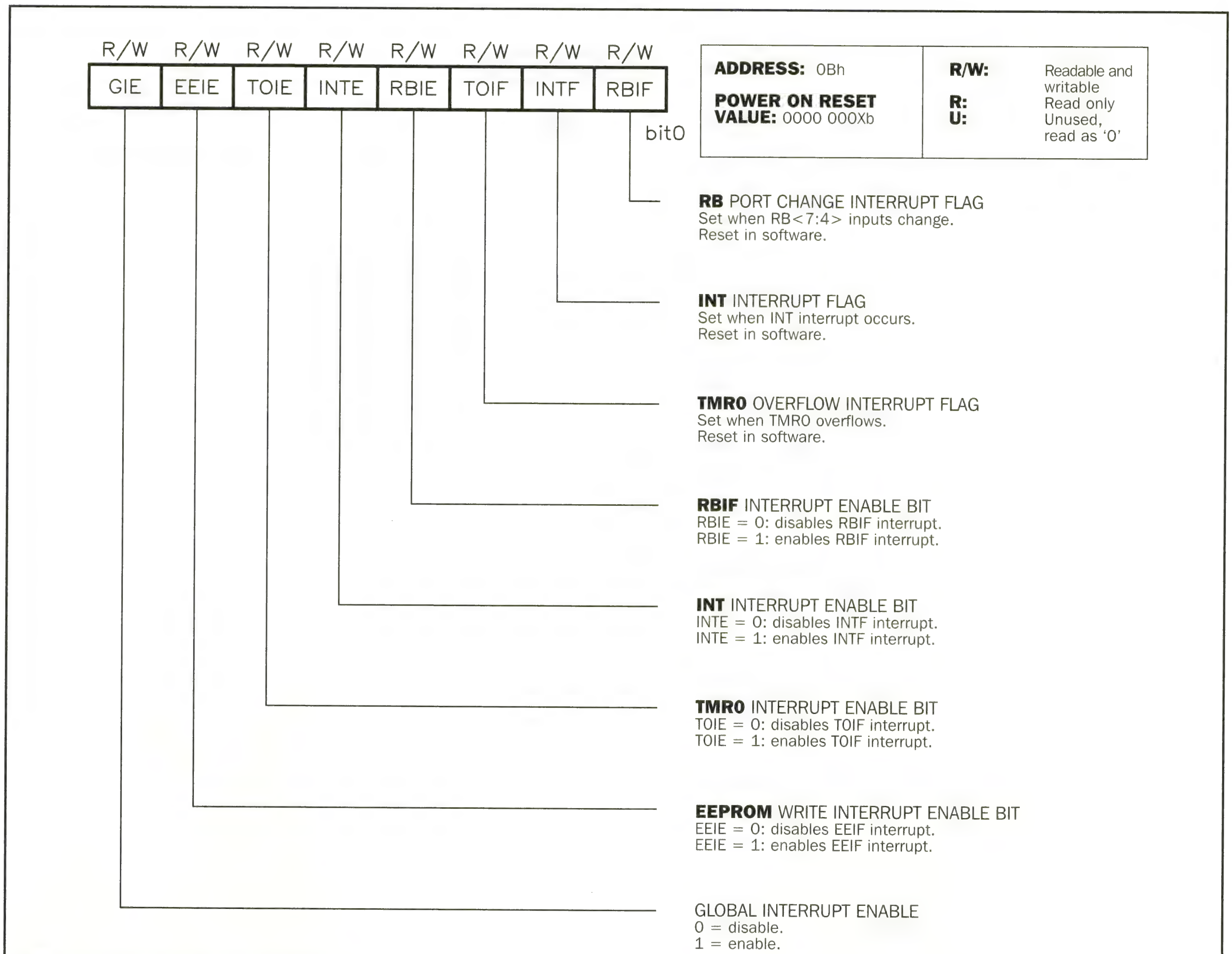
When an interrupt is responded to, the GIE is cleared to disable any further interrupt, the return address is pushed into the stack and the PC is loaded with 0004h. Once in the interrupt service routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flagbit(s) must be cleared in software before enabling interrupts to avoid recursive interrupts.

Next month, we'll focus on some of the special features and I/O capabilities of the PIC16C84. In Part 4 of the series, we will start building code for the PIC16C84, and begin to look at application development.

Further Reading

Data sheets containing further details of each of the PIC family are available from Maplin. Refer to the catalogue for details of individual devices.

Check the T.A.K. DesignS home page http://www.takdesign.demon.co.uk\ for FAQ's and PIC links.



R/W R/W R/W R/W R/W R/W R/W R/W
GIE EEIE TOIE INTE RBIE TOIF INTF RBIF
bit0
ADDRESS: 0Bh
POWER ON RESET VALUE: 0000 000Xb
R/W: Readable and writable
R: Read only
U: Unused, read as '0'
RB PORT CHANGE INTERRUPT FLAG
Set when RB<7:4> inputs change.
Reset in software.
INT INTERRUPT FLAG
Set when INT interrupt occurs.
Reset in software.
TMR0 OVERFLOW INTERRUPT FLAG
Set when TMR0 overflows.
Reset in software.
RBIF INTERRUPT ENABLE BIT
RBIE = 0: disables RBIF interrupt.
RBIE = 1: enables RBIF interrupt.
INT INTERRUPT ENABLE BIT
INTE = 0: disables INTF interrupt.
INTE = 1: enables INTF interrupt.
TMR0 INTERRUPT ENABLE BIT
TOIE = 0: disables TOIF interrupt.
TOIE = 1: enables TOIF interrupt.
EEPROM WRITE INTERRUPT ENABLE BIT
EEIE = 0: disables EEIF interrupt.
EEIE = 1: enables EEIF interrupt.
GLOBAL INTERRUPT ENABLE
0 = disable.
1 = enable.

Figure 10. Interrupt control register.