A PIC Microcontroller Introduction

What can you do with a microcontroller?

Answer: **Virtually Any Project**
Easily!...

Microcontrollers give you a fantastic way of creating projects. A PIC microcontroller is a processor with built in memory and RAM and you can use it to control your projects (or build projects around it). So it saves you building a circuit that has separate external RAM, ROM and peripheral chips.

What this really means for you is that you have a very powerful device that has many useful built in modules e.g.

- EEPROM.
- Timers.
- Analogue comparators.
- UART.

Even with just these four modules (note these are just example modules - there are more) you can make up many projects e.g.:

*** Frequency counter** - using the internal timers and reporting through UART (RS232) or output to LCD.

*** Capacitance meter** - analogue comparator oscillator.

*** Event timer -** using internal timers.

*** Event data logger** -capturing analogue data using an internal ADC and using the internal EEPROM for storing data (using an external I2C for high data storage capacity.

*** Servo controller** (Control through UART) - using the internal PWM module or using a software created PWM.

The PIC Micro is one of the most popular microcontrollers and in case you were wondering the difference between a microprocessor and a microcontroller is that a microcontroller has an internal bus with in built memory and peripherals.

In fact the 8 pin (DIL) version of the 12F675 has an amazing number of internal peripherals. These are:

- Two timers.
- One 10bit ADC with 4 selectable inputs.
- An internal oscillator (or you can use an external crystal).
- An analogue comparator.
- 1024 words of program memory.
- 64 Bytes of RAM.
- 128 Bytes of EEPROM memory.
- External interrupt (as well as interrupts from internal peripherals).
- External crystal can go up to 20MHz.
- ICSP : PIC standard programming interface.

And all of these work from within an 8 pin DIL package!

In the mid-range devices the memory space ranges from 1k to 8k (18F parts have more) - this does not sound like a lot but the processor has an efficient instruction set and you can make useful projects even with 1k e.g. LM35 temperature sensing project that reports data to the serial port easily fits within 1k.

Features

In fact a PIC microcontroller is an amazingly powerful fully featured processor with internal RAM, EEROM FLASH memory and peripherals. One of the smallest ones occupies the space of a 555 timer but has a 10bit ADC, 1k of memory, 2 timers, high current I/O ports a comparator a watch dog timer... I could go on as there is more!

Programming

One of the most useful features of a PIC microcontroller is that you can re-program them as they use flash memory (if you choose a part with an F in the part number e.g. 12F675 not 12C509). You can also use the ICSP serial interface built into each PIC Microcontroller for programming and even do programming while it's still plugged into the circuit!

You can either program a PIC microcontroller using assembler or a high level language and I recommend using a high level language such as C as it is much easier to use (after an initial learning curve). Once you have learned the high level language you are not forced to use the same processor e.g. you could go to an AVR or Dallas microcontroller and still use the same high level language.

Input / Output - I/O

A PIC Microcontroller can control outputs and react to inputs e.g. you could drive a relay or read input buttons.

With the larger devices it's possible to drive LCDs or seven segment displays with very few control lines as all the work is done inside the PIC Micro.

Comparing a frequency counter to discrete web designs you'll find two or three chips for the microcontroller design and ten or more for a discrete design. So using them saves prototype design effort as you can use built in peripherals to take care of lots of the circuit operation.

Many now have a built in ADC so you can read analogue signal levels so you don't need to add an external devices e.g. you can read an LM35 temperature sensordirectly with no interface logic.

Peripherals

The PIC microcontroller has many built in peripherals and this can make using them quite daunting at first which is why I have made this introductory page with a summary of each major peripheral block.

At the end is a short summary of the main devices used in projects shown on this site.

The best way to start is to learn about the main features of a chip and then begin to use each peripheral in a project. I think learning by doing is the best way.

PIC microcontroller Feature	PIC microcontroller feature description
Flash memory	Re-programmable program storage.

RAM	Memory storage for variables.
EEPROM	Long term stable memory : Electrically Erasable Programmable Read Only Memory.
I/O ports	High current Input/Output ports (with pin direction change).
Timers/Counters	Typically 3.
USART	Built in RS232 protocol (only needs level translator chip).
CCP	Capture/Compare/PWM module.
SSP	I2C and SPI Interfaces.
Comparator	An analogue comparator and internal voltage reference.
ADC	Analogue to digital converter.
PSP	Parallel Slave Port (for 8 bit microprocessor systems).
LCD	LCD interface.
Special features	ICSP,WDT,BOR,POR,PWRT,OST,SLEEP
ICSP	Simple programming using In Circuit Serial Programming.

Note:these are some of the main features
(some chips have all of these and some don't).

Flash memory

This is the program storage area and gives you the **most important benefit** for using a PIC microcontroller - You program the device many times. Since when does anyone get a program right first time ?

Devices used in projects on this site can be re-programmed up to **100,000** times (probably more) as they use Flash memory - these have the letter F in the part name. You can get cheaper (OTP) devices but these are One-Time-Programmable; once programmed you can't program it again!

ICSP

In Circuit Serial Programming (ICSP) is the **next most important** benefit. Instead of transferring your chip from the programmer to the development board you just leave it in the board. By arranging the programming connections to your circuit correctly you won't need to remove the chip!

You can **re-program** the device while it's still **in the circuit** so once your programmer is setup you can leave it on the bench and test your programs without moving the chip around and it makes the whole process much easier.

I/O Ports

Input / Output ports let you **communicate** with the outside world so you can control leds, LCDs or just about anything with the right interface. You can also set them as inputs to gather information.

Pin direction

Most PIC microcontroller pins can be set as an input or and output and this can be done on the fly e.g. for a dallas 1 wire system a pin can be written to generate data and read at a later stage. The TRIS register controls the I/O direction and setting a bit in this register to **zero** sets the pin as **output** while setting it as **one** sets the pin as **input**.

This allows you to use a pin for multiple operations e.g. the Real Time clock project uses RA0, the first pin of PORTA, to output data to a seven segment display and at a later point in the program read the analogue value as an input.

Current

The PIC I/O ports are high current ports capable of directly driving LEDs (up to 25ma output current) - the total current allowed usually ~200mA this is often for the whole chip (or specified for several ports combined together).

Timer / Counters

Each PIC microcontroller has up to three timers that you can either use as a timer or a counter (Timer 1 & 2) or a baud clock (Timer 2).

Timer 0

The original timer: Timer 0 was the first timer developed and you can find it in all the earliest devices e.g. 16F84 up to the most current e,g, 16F877A.

It is an 8 bit timer with an 8 bit prescaler that can be driven from an internal (Fosc/4) or external clock. It generates an interrupt on overflow when the count goes from 255 to zero.

Timer 0 always synchronizes the input clock (when using external clock).

Note: You can read and write timer 0 but you can not read the prescaler.

Note: The prescaler changes its effect depending on whether it is a timer prescaler or a watch dog prescaler - so the same prescaler setting may prescale by 2 or by 1 depending on its use!

Timer 1

This is a 16 bit timer that generates an overflow interrupt when it goes from 65535 to zero. It has an 8 bit programmable prescaler and you can drive it from the internal clock (Fosc/4) or an external pin.

To eliminate false triggering it also has an optional input synchronizer for external pin input.

This timer can be used in sleep mode and will generate a wakeup interrupt on overflow.

Timer 1 is also read by the CCP module to capture an event time.

Note: Using this timer in sleep mode will use more current.

In addition it can be used to drive a low power watch crystal. This is something that sounds good but I don't recommend you do it as watch crystals are extremely difficult to drive correctly. You should only use it if you are going to make a pcb and follow all the guidelines in making it noise free. I used a DS1307 in the Real Time clock project which drives the crystal directly but even this is difficult to get operating accurately.

Timer 2

This is an 8 bit timer with an 8 bit prescaler and an 8 bit postscaler. It takes its input only from the internal oscillator (Fosc/4).

This timer is used for the timebase of a PWM when PWM is active and it can be software selected by the SSP module as a baud clock.

It also has a period register that allows easy control of the period. When timer 2 reaches the PR2 register value then it resets. This saves having to check the timer value in software and then reset

the timer and since it is done in hardware the operation is much faster - so you can generate fast clocks with periods that are multiples of the main clock.

USART

The USART is a useful module and saves having to code up a software version so it saves valuable program memory. You can find more information on RS232here and how to make it work.
Look here for pin outs.

All you need to interface it to a PC serial port is a MAX232 chip (or equivalent).

Note: An equivalent MAX232 chip is the SP202ECP that has the same pinout as the MAX232 but lets you use 100nF capacitors - so you don't need the large 1uF caps.

Baud Rates

You have to be careful using the baud rates as they depend on the main clock in use and normal oscillator values in general do not fit very well with 'real' baud rates.

There is a table of baud rates in microchip data sheet DS33023A which indicates the expected percentage error for a specific clock rate and in general the higher the main clock the lower the error.

You sometimes have to play around with the register settings to get a better fit with your clock rate and the baud rate you want. An example is for an 8MHz clock - if you use BRGH=1 and an 8MHz clock (see the 16F88 datasheet) you get accurate baud rates up to 38.4kbaud. You have to force this to work e.g. in mikroC the built in USART routines use BRGH=0 so at 8MHz the baud rate is only accurate to 9.6kbaud.

If you want a super-accurate baud rate the best way is to use a clock crystal that ends up giving you that baud rate i.e. work back through the baud rate equations to find the crystal you need.

CCP

The Capture/Compare/PWM module has three modes of operation:

- Capture - Capture the time of an event.
- Compare - Generate an output when Timer 1 reaches a value.
- PWM - Pulse Width Modulation.

Capture

Capture mode is used to capture the value of Timer 1 when a signal at the CCP pin goes high (or low depending on how the CCP is set up). The CCP can accurately capture the arrival time of a signal at the CCP pin so it can be used for pulse time measurement.

Compare

Compare mode is used to generate an output when Timer 1 reaches a value you put into CCPR1.
One special event trigger mode lets you start the ADC when the compare mode triggers.

PWM

PWM gives you one Pulse Width Modulation output with 10 bit resolution and with no software overhead - once started it operates all by itself unless you want to change the duty cycle.

It uses Timer 2 to define its operation using Timer 2 period register to define the frequency of the PWM.

Note: The duty cycle is not a percentage it is the number of periods of the PWM clock that the output is high!

SSP

The Synchronous Serial Port lets you communicate with devices that use either the SPI (Serial Peripheral Interface) or I2C (Inter IC communication) protocols. Note that for full Master mode I2C operation you need to choose a PIC device that has the MSSP device (Master Synchronous Serial Port).

SPI and I2C are shared so you can only use one at a time (or you could use the I2C bit banged routines in the Real Time Clock project to have both at the same time).

You can find a project that uses I2C here and you can find more information on I2C here.

Comparator and comparator voltage reference

The comparator is module that has two analogue comparators which can be set up in one of 8 different ways. Either digital or analogue inputs can be compared to reference voltages.

In one mode an internally generated voltage reference is used as an input to both comparators and in the same mode multiplexing lets you monitor up to four different input pins.

You can even send the output of the comparator to a pin so that it is used independently from the microcontroller e.g. in a circuit where you need a comparator you don't need an extra chip!

The analogue level must be between Vdd and Vss as protection diodes won't allow anything else.

The module will generate an interrupt if the comparator output changes.

You can use it in sleep mode and the interrupt will wake it up.

The source impedance of the analogue signal must be smaller than 10k.

ADC

The single 10 bit Analogue to Digital Converter can have up to 8 inputs for a device multiplexed from input pins.

The ADC can be used during sleep but you have to use the RC clock mode. One benefit of this is that there will be no digital switching noise so you will get better conversion accuracy.

For the 16F877A you can not just choose to use an analogue input if you feel the need as there are only a specific and limited number of ways that the analogue input pins can be enabled. It is best to start with AN0 and add more as necessary - see the datasheet for which analogue inputs can be enabled e.g. if you started a design using only AN5 you would find that you may have to enable a few more analogue inputs as well!

The 16F675 can measure 4 analogue input pins!

PSP

The Parallel Slave Port lets you to connect the PIC microcontroller directly into a microprocessor system. It provides an 8 bit read/write data bus and RD (read) WR (write) and CS (chip select) inputs - all active low.

This will let you add a PIC microcontroller to a system so that the PIC microcontroller can be treated as a memory mapped peripheral. It will let the microcontroller behave just as though it was another microprocessor building block e.g. some memory or ram but in this case you have full control over exactly what the building block is i.e. you can re-program the PIC microcontroller to do just about anything.

This provides an easy route to adding a PIC microcontroller to an 8 bit system that already exists.

LCD

The LCD interface lets you directly interface to an LCD saving you having to use an LCD module such as the HD44780. I have not used this feature as it is another commercial requirement where removing a chip (HD44780) saves money in a production run. I think it is capable of driving a graphic LCD.

Special Features

ICSP	In Circuit Serial Programming	click here (jumps to ICSP section).
WDT	Watch dog timer	This is a software error protector.
BOR	Brown Out reset	This detects if the power supply dips slightly and resets the device if so.
POR	Power on reset	This starts microcontroller initialization.
PWRT	PoWeR up Time	A time delay to let Vdd rise.
OST	Oscillator start up timer	Wait for 1024 cycles after PWRT.
SLEEP	PIC microcontroller sleepmode	Enter low power mode.

WDT

If your software goes haywire then this timer resets the processor. To stop the reset the well behaved software must periodically issue the CLRWDT instruction to stop a resert. The WDT runs using its own oscillator. It runs during sleep and shares Timer 0 prescaler.

POR

Power On Reset starts PIC microcontroller initialization when it detects a rising edge on MCLR.

PWRT

If you enable this then 72ms after a POR the PIC microcontroller is started.

OST

Oscillator Startup Timer delays for 1024 oscillator cycles after PWRT (if PWRT is enabled) ensuring that the oscillator has started and is stable. It is automatic and only used for crystal oscillator modes and is active after POR or wake from sleep.

SLEEP

Sleep mode (or low power consumption mode) is entered by executing the 'SLEEP' command. The device can wake from sleep caused by an external reset, Watch Dog Timer timeout, INT pin RB port change or peripheral interrupt.

Project device overview

This site mainly uses three PIC devices out of the hundreds of different chips that microchip produces. This does not sound like a lot but you can use the devices in almost any project and they have so many built in peripherals that you can make hundreds of projects with them.

The other microchip devices are all useful in different situations - perhaps they have more memory or different peripherals - this is useful if you want to tailor your designs to the system you build - but probably more useful in a commercial environment where every cent counts in a production run.

All three devices are extremely powerful and the main difference is that they have different numbers of pins and memory size.

Note: There are differences in using the devices i.e. there are some registers that are different but in the generally you can interchange them - this is made easier using a high level language.

The devices used in this site are:

PIC microcontroller Device	PIC microcontroller No. Pins	PIC microcontroller Flash memory WORDS
12F675	8	1k
16F88	18	4k
16F877A	40	8k

Note : When looking at the microchip site the memory size is kwords - ignore kbytes - you need the kword size as this is what each instruction occupies - the kbyte size is for comparison to other types of micros (probably). But the microcontroller data bus is 8 bits wide so it is an 8 bit microcontroller (different program memory and data memory due to using Harvard architecture).

(Note: that all of them have the letter F in - this means it is a Flash re-programmable part - don't go and buy a part with O in as its OTP - programmable only once! - only do that if you are really really sure it's the final design).

PIC Microcontroller Flash Memory size

You may think that 1k or even 8k is so tiny that it won't be useful but each PIC microcontroller uses RISC (Reduced Instruction Set Computing) which simply means that it has a cleverly arranged instruction set that only has a few instructions. The mid range parts have 35 instructions.

If you use the high level language as recommended in this site then you won't need to be too aware of the instruction set it just means you can do a lot with a small amount of memory. Most of the projects on this site although they are fully working projects fit within 2k words!

Note: If you need more memory you can always move to the 18F series of PIC microcontrollers. Another option is to add an I2C serial eprom.

PIC microcontroller RAM and EEPROM size

The PIC microcontroller **RAM size** is also important as it stores all your variables and intermediate data.

Note: You can usually alter the program to **use less RAM** by choosing the right variable sizes or

changing how your program works

For example **don't use floating point** alter it to use a different variable type e.g. you can use long integers with fixed point operation to avoid floating point.

PIC microcontroller EEROM : Electrically Erasable ROM is used to store data that must be saved between power up and power down.

This area is readable and writable and has a much longer life than the main program store i.e. it has been designed for more frequent use.