The Practical Wireless IBP Beacon Clock (PIC Version) part 1

Phil Cadman G4JCP introduces the PW IBP Clock (PIC version): "Back in December 2001/January 2002, *PW* published a design for an International Beacon Project (IBP) Clock. It was a single band design, which used l.e.d.s to indicate, in real time, which one of the eighteen IBP beacons was due to transmit in its allocated time slot on 14.1, 18.110, 21.150, 24.930 or 28.2MHz. Despite its simple function, the design required numerous i.c.s and was laborious to wire. A multiple band version that had been developed, was deemed too complex for publication but since then a programmable integrated circuit (PIC) version has been developed and is presented here in Part 1."

Important copyright information: The terms PIC and PICmicro are registered trademarks of Microchip Technology Inc. in the USA and other countries. (**Microchip Technology Inc. 2355 West Chandler Boulvard, Chandler, Arizon, AZ 85224, USA**).

Editorial note: As this major PIC project is the very first of its type in *PW*, Phil Cadman G4JCP and the Editorial team consider that the following introduction to the device is entirely appropriate. Everyone involved in preparing and producing the article on behalf of readers hope it will provide encouragement to anyone thinking of building the project! **Rob Mannion G3XFD/EI5IW**

Introducing The PIC

Just what is a PIC? Phil G4JCP provides the answers: As briefly mentioned above in the copyright notice, Microchip Technology of Chandler, Arizona, make a range of semiconductor products but the product most familiar to electronics enthusiasts is their PIC range of microcontrollers.

A microcontroller is a microprocessor, which has, on the same piece of silicon, a clock generator, program memory, data memory and various peripheral devices. Microchip manufacture a wide variety of PIC microcontrollers, from 8-bit models, through to 16-bit models to high performance 16-bit Digital Signal Controllers. The 8-bit models are by far the most popular – and a number of the Mid-Range flash versions have become highly favoured by electronics enthusiasts.

Flash Memory

Flash memory is electrically erasable and programmable non-volatile memory. Storing the microcontroller's program in flash memory means that the same PIC can be reprogrammed many times over. In addition, flash PICs have a simple, two wire serial programming interface, making them easy to physically program. The devices themselves are relatively inexpensive and Microchip provides a free development environment,which runs on the *Windows* operating system.

There's a bewildering array of PICs to choose from, mainly because commercial users of PICs always want maximum performance at minimum cost. Having a large number of different PICs available helps them achieve that objective. Electronics enthusiasts on the other hand, don't have commercial pressures and neither do they have the buying power of big companies.

So, constructional enthusiasts like us have settled on a relatively small number of PICs, which form a logical progression in terms of size and performance and are readily available. That's not to say there's no progress! Many enthusiasts are moving on to the higher performance 8-bit PICs and even on to the 16-bit ranges.

The 'original' Flash PIC was the 18-pin PIC16F84. Now available in its suffix 'A' guise, it's an ideal entry level PIC due to its simplicity. However, there are now cheaper and more powerful PICs available, so it's nowhere near as popular as it once was. The PIC used in the IBP Clock is a 16F871. This is a 40-pin device, which has 2,048 words of program memory, 128 bytes of data memory and 64 bytes of EEPROM. As it's quite typical of the devices used by enthusiasts, it's worth a closer look.

Inside The 16F871

The diagram, **Fig. 1**, shows a simplified block diagram of what's inside the 16F871. Central to its operation is an 8-bit Reduced Instruction Set (RISC) microprocessor. Most microprocessors have so called complex instruction sets (CISC processors) and make use of a large number of instructions, which can vary in length, perform simple or complex operations and take varying lengths of time to execute.

Conversely, RISC processors use a small number of simple, fixed length instructions, which execute in short, fixed lengths of time. The RISC processor cores have a definite advantage over their CISC counterparts in that they can be implemented with quite small amounts of silicon. This makes them cheap, power efficient and fast.

The RISC processors often use the Harvard architecture, in which program and data are stored in separate memories and accessed using separate buses. This improves performance over the traditional Von Neumann architecture where program and data share the same memory. Midrange PIC instructions are all single, 14-bit wide words.

The PIC fetches each instruction in a single instruction cycle, and a two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions execute in a single instruction cycle, except for program branches, which execute in two cycles.

An instruction cycle occupies four oscillator cycles, so a PIC with a 4MHz clock can execute one million (inline) instructions each second. In practice, PICs can be quite fast, especially when clocked at their maximum speed, which in the case of the 16F871 is 20MHz. That equates to five million instructions per second, or one instruction every 200nS.

As can be see from Fig. 1, the Flash program memory is accessed independently of the internal 8-bit data bus. However, there's a 'back way' into the program memory, which allows it to be read in a similar way to EEPROM data memory. It's also possible for the 16F871 (and some other PICs) to reprogram the Flash memory while actually running a program. By the way – don't get too concerned about the small amount of program memory, the code for the IBP l.e.d. clock, the l.c.d. clock and the l.e.d. repeater, all fits quite easily in 2K words!*

There are 128 bytes of RAM data memory. This is static memory, which retains its contents as long as there's power supplied to the PIC. Incidentally, one particularly nice thing about the PIC is the inclusion of EEPROM in many devices. The EEPROM is nonvolatile data memory and it allows the PIC to 'remember things' even in the absence of power, thereby allowing customised settings and the like to be retained through power down.

The 16F871 has three timer/counters. Timer 0 and the more versatile Timer 2 are both 8-bit timers, while Timer 1 is a 16-bit timer. Allied to the timers is the CCP Capture/Compare/PWM module, used for timing events and for generating a pulsewidth modulated output. This is very useful for engine management systems and the like. Most PICs also incorporate a 'Watchdog Timer'. This timer, which has its own oscillator, runs independently of the rest of the PIC.

If the Watchdog Timer ever times-out whilst enabled, it instructs the Reset circuitry to reset the PIC. To prevent a timeout, the PIC must periodically execute a special 'Clear Watchdog Timer' instruction. The idea being that this instruction is placed where it should get repeatedly executed, such as in the main program loop. If the PIC ever gets stuck somewhere else in its program, the Watchdog will timeout, reset the PIC and restart the program!

Note: Readers who are not familiar with 'computer speak' may be confused with the upper case K. Many of us understand that it refers to Kelvin, a term related to temperature used in science. The term is also used to represent the nearest computer equivalent to 1,000, which is 1024 (or 2^{10}). **Editor*

Useful USART

Another very useful peripheral is the universal synchronous/asynchronous receiver/transmitter or USART. This supports synchronous serial communications (like those used between the PC's keyboard and motherboard), and the more familiar asynchronous serial communications used by modems and v.d.u.s.

Even if an application doesn't use serial communications, the availability of a serial interface is very useful during program development and debugging. As you may appreciate, it's awkward to 'see' what's happening to the program flow within a PIC (although there are ways and means), so sending a byte out of the serial port at strategic places in your program whilst monitoring them on a v.d.u., can help keep track of what's happening.

Given the low cost of PICs, it's tempting to use them as intelligent peripherals in more powerful systems. To this end, some PICs have a Parallel Slave Port or PSP. This module allows parallel communications between the PIC and another processor to be controlled by the other processor without the need for additional logic. Finally, many PICs incorporate analogue to digital converters and in the case of the 16F871, the A/D is a 10-bit, 8-channel converter.

Most of the peripherals I've mentioned need inputs and/or outputs (I/O). Rather than dedicate pins on the i.c. package to each peripheral (which would be wasteful if a peripheral wasn't used), all input/output pins on a PIC can be used as general purpose I/O. They're arranged in groups of up to eight I/O lines, and referred to as Ports A, B, C, D and E. However, pins that are also allocated to a peripheral cannot be used as general purpose I/O if that peripheral is in use. This must be borne in mind at the hardware design stage and is the reason why the beacon l.e.d.s in Figure 3 skip certain port pins.

The diagram, **Fig 2**, shows the pin-outs of the 16F871 and I've included the peripheral function of those pins, which are of interest in the IBP Clock project. For example, pin 6 – RA4/T0CKI – is I/O line 4 on Port A or alternatively it can be the external input to Timer 0.

Smaller Sibling

As I mentioned earlier, certain PICs have become very popular with electronics enthusiasts, the 16F871 being one of them. It has a smaller sibling in the 16F870, which is the same as the 16F871, except that it has no Parallel Slave Port. It's available in a 28-pin 'Skinny DIP' (0.3 inch pin spacing) package which occupies significantly less space than the usual 0.6 inch spaced package.

Also in a 28-pin package is the 16F872, which is effectively a 16F870. However, this has an enhanced synchronous communications module instead of a USART.

If more memory is needed, then the 16F874 has twice the memory of the 16F871 and the 16F877 four times the memory. The 16F873 and 16F876 are the corresponding larger memory versions of the 16F870.

The nice thing about these PICs is that they're pin compatible – providing they have the same number of pins, of course! In fact, many PICs are pin compatible, which allows designers to change processors without necessarily having to redesign p.c.b.s.

Down In Size

Going down in size, the 16F628 is an 18-pin device with many of the peripherals of the 16F870 but with fewer I/O pins. The 16F627 is found in some PIC kits and that's just a 16F628 but has 1,024 words of program memory instead of 2,048 words.

Owing to their low cost and pin compatibility with the 16F84, the 16F627 and 16F628 have become the new entry level PIC processors. Even though the 16F628 is quite small for a microcontroller, Microchip make even smaller - and cheaper - devices. Four of these are of interest to us.

First, on the choice list the 16F676, a 14-pin device with two timers, an analogue comparator and an 8-channel A/D converter. It has 1,024 words of program memory, 64 bytes of RAM and 128 bytes of EEPROM. For applications that don't need to handle analogue inputs, the 16F630 is the same as the 16F676 but without the A/D converter.

Finally on this topic, the smallest pair of PICs I'm going to mention are the 12F675 and its A/D-less sibling, the 12F629. These devices have just eight pins and yet contain the same peripherals as their 14-pin counterparts I've already described. All necessary support circuitry is inside these little PICs - even a 1% accurate 4MHz oscillator – so all but two of their pins (remember PICs still need power) can be used as inputs or outputs.

Given their low cost, ease of programming and high integration, PICs open up the possibility of enthusiasts using microcontrollers in situations where previously they would not have been practicable. And I take the view that using PICs in Amateur Radio is little different to using a 741 op amp in place of a bunch of transistors! We're just trading hardware for software, that's all.

Now we're ready to tackle the main project! - turn the page!

The Practical Wireless IBP Beacon Clock (PIC Version) part 1

Don't start unless you've read the introduction on the previous two pages!

Back in December 2001/January 2002, *PW* published a design for an International Beacon Project Clock. It was a single-band design, which used l.e.d.s to indicate, in real time, which of the eighteen IBP beacons should be transmitting. Despite its simple function, the design required numerous i.c.s and was laborious to wire. A five-band version, designed at the same time, was deemed too complex for publication.

A microprocessor based design was also considered but using universally available CD4000 series logic chips was felt to be the better option. However, since the logic based design was published, Arizona Microchip's PIC(tm) microcontrollers have enjoyed increasing popularity and numerous variants are now both inexpensive and widely available. In addition, there are several inexpensive kits currently obtainable, which allow electronics enthusiasts to easily experiment with – and physically program – a range of popular PIC devices.

This month a PICbased l.e.d. IBP Clock is described, which uses just two i.c. packages (albeit one has 40 pins). In part 2, a version with a liquid crystal display will be featured, together with an associated l.e.d. repeater. (The latter allowing a l.e.d. display to be added to the l.c.d. unit.).

The Complete Circuit

The complete circuit of the l.e.d. clock is shown in **Fig. 3**. Both the PIC – a PIC16F871 – and the CD/HEF4060 oscillator/divider i.c. run from a +5V d.c. supply, provided by a standard 7805 three terminal regulator (IC2). The values of C5 and C6 are not critical, but C6 must be at least 330nF, while anything between 10μF and 100μF will suffice for C5. Diode D1 offers protection from a reversed supply connection.

Just as in the original wired logic based design, the timebase for the clock uses a 32KHz (See note in PIC introduction article) – actual frequency 32768Hz – miniature watch crystal and a CD/HEF4060 oscillator/divider i.c. The stability of this oscillator is critical to the long term accuracy of the clock, so I've included R32 and C7 to provide extra supply decoupling. The oscillator section of the CD/HEF4060 is connected internally to a 15-stage binary counter and an output at 2,048Hz is taken from pin 7 to drive the PIC's Timer 0.

Trimmer capacitor C9 should be adjusted so that X2 resonates at exactly 32,768Hz. This can be checked in two ways. First, there's a buffered 32,768Hz output available at pin 9, but do try to use a frequency counter with a 10 second (or longer) gate time. Alternatively, the period of the 2Hz output at pin 3 can be measured. It should, of course, be exactly 500,00μS. Please make sure your test equipment can measure to an accuracy of 1μS or better.

The buffered test points are essential because connecting anything to the oscillator circuit itself will cause a frequency shift due to the capacitance of the probe. Alternatively, simply set C9 to maximum and run the clock over a period of a few hours. See how it keeps time and adjust C9 accordingly.

The values of C8 and C9 are suitable for a crystal requiring a load capacitance of 12pF (a common value for 32KHz (see PIC intro note) watch crystals, as used in the prototype). If you use a crystal that requires a different load capacitance, then alter the values appropriately. Both C8 and C9 should be around twice the specified load capacitance of the crystal, when adjusted. If C9 can't quite pull a '12pF' crystal exactly on to frequency, then a 10pF ceramic capacitor connected in parallel with the trimmer should solve the problem.

Many PIC-based designs that maintain real time use either the PIC's own crystal (X1 in our case), or a 32KHz watch crystal connected to Timer 1's internal oscillator. While both methods are valid, I prefer a separate oscillator.

Eighteen Beacon LEDs

As you can see from the diagram, the 18 beacon l.e.d.s (catering for the three minute IBP cycle) are driven directly by the PIC and resistors R1 to R18 should be chosen to give each l.e.d. the required brightness. Having a non-multiplexed array with individual resistors allows the use of different colour l.e.d.s, even if they have widely varying efficiencies. To dim the l.e.d.s for say, night time use, all you have to do is simply switch in one or more forward-biased diodes (1N4002 or similar) in series with the l.e.d.s' +5V supply.

According to the data sheet, PIC pins configured as outputs can source or sink up to 25mA. But like virtually all logic chips, PICs are more efficient at sinking current than sourcing current. To keep the internal voltage drop to less than 0.5V (10% of supply), I only allow PIC pins to source up to 7mA and sink up to 15mA. So, it's wise to limit the l.e.d. current to 15mA.

Most of the other resistors associated with the PIC are 10k, a value that's not critical; anything between 4.7 and 15k is perfectly acceptable. I should mention that resistors that merely serve to pull inputs high or low (like R28 and R24 respectively) are not strictly necessary. However, all PIC I/O pins can be configured, under software control, to be either inputs or outputs.

Accidentally configuring an input that happens to be tied directly to +5V or ground, as an output, can result in a large flow of current. Using an input resistor limits any unexpected current to a safe value. Resistor R35 performs a similar function, this time protecting the output of the CD/HEF4060.

Although input resistors should not be necessary once a design has been thoroughly tested, I still like to

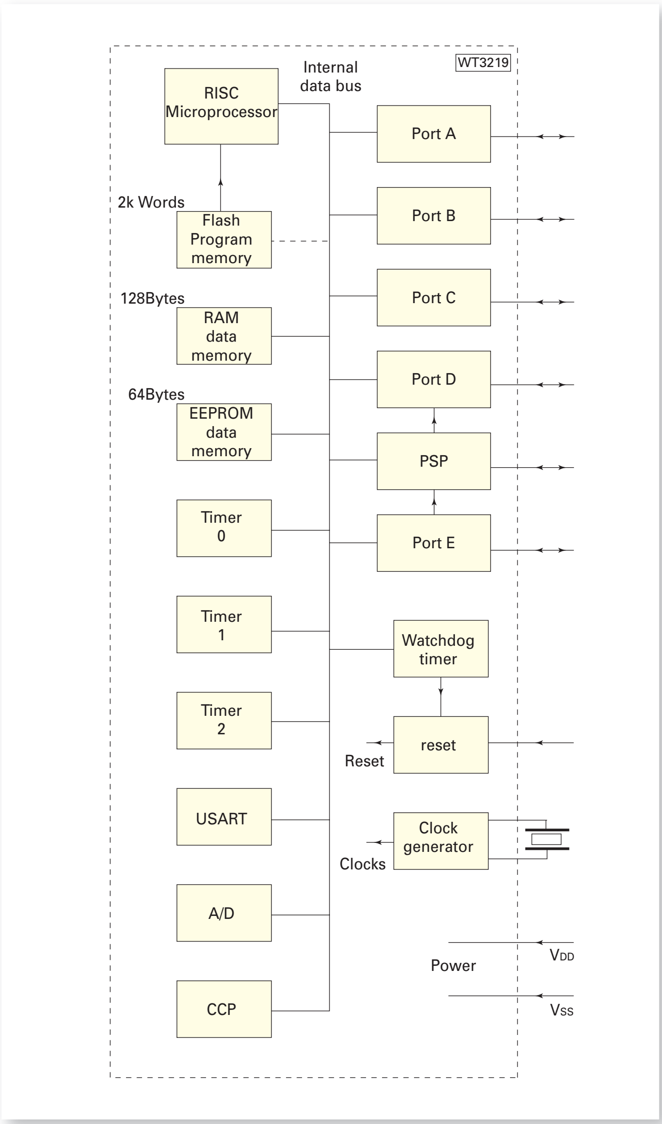

Fig. 1: The internal structure of the PIC family of microcontrollers.

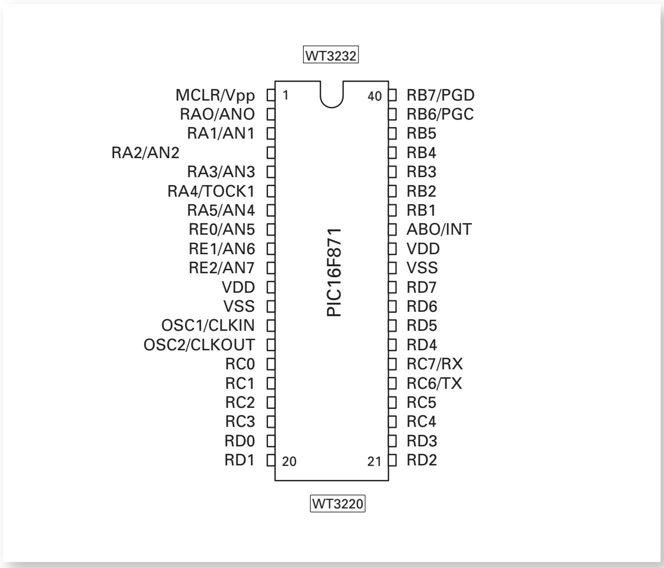

Fig. 2: The pin-out for the PIC used for this project.

keep them in circuit. Given that the IBP Clock may be used in places where there's a significant amount of r.f. energy floating around, it's not impossible for this energy to disturb a PIC. In particular, most modern v.h.f./u.h.f. hand-held transceivers can run up to 5W output and that kind of power at close range can interfere with many electronic devices, particularly if they're not adequately shielded as we know to our cost with TV and radio receivers!

Resistors R19 to R23 are, of course, necessary, as switch S1 selects the required band by connecting the appropriate band input (normally pulled high) to 0V. The sixth position is used for turning all the beacon l.e.d.s off. Incidentally, if a two-pole switch is used here, the other half of the switch can be used with more l.e.d.s to provide illuminated band indications.

Switch S2 is the reset switch, which sets the clock to the beginning of a three minute cycle. The software effectively de-bounces S2 by providing a one second hold off period during which time further resets are inhibited. Try to press SW2 as close as possible to the start of a three minute cycle or maybe a fraction of a second early, which some people prefer to do*****.

Editorial comment: *Phil's advice is very sound and I have had a prototype of the original wired logic beacon clock at home for some years and follow his tip by having the clock running fractionally (half a second or so) ahead of the correct time. It works well in practice and means the l.e.d. is illuminated just before the allotted time slot for the beacon the band you are monitoring. This technique helps the user recognise the high speed (22w.p.m.) Morse and alerts you what to expect (as the beacon l.e.d. is labelled with its callsign).* ***G3XFD***

The LED Repeater

Even though the l.e.d. repeater (featured in Part 2) is primarily designed to provide l.e.d. indications for the l.c.d. clock, it can also be driven from the l.e.d. clock. To prevent ground loops, the repeater has an optocoupler at its input. Consequently, no damaging fault currents can flow thorough the 0V rails.

Twin screened cable should be used to connect the repeater to the l.e.d. or l.c.d. clock, with the screen connected to protective earth rather than 0V, if possible. Only a single 2kΩ resistor is needed to limit the current fed to the optocoupler but I've split it into two equal resistors – R30 and R31 – for safety reasons. You'll notice that a short circuit between either of the two repeater terminals to either 0V or +5V, will not cause more than 5mA to flow.

The l.e.d. and l.c.d. clocks communicate with the repeater using the PIC's USART. Now although the PIC's crystal frequency (X1) isn't critical as far as running the l.e.d. clock is concerned, it does drive the USART's baud rate generator and so directly affects the serial port's transmission (and reception) speed. With the 4MHz crystal specified, I've programmed the baud rate to be 4,808. That's as close to 4,800 as the combination of crystal frequency and baud rate generator divisors will allow. Please, do remember that for the clock to communicate with the repeater, the PICs' crystal frequencies must be the same, or at least very close. Why 4,800? The reason will be revealed next time!

The 4MHz crystal used in the prototype clock required a load capacitance of 20pF, hence the value of 22pF for C3 and C4. Actually, 22pF is on the low side as the series connection of C3 and C4 ought be close to 20pF. Many crystals specify a load capacitance of 30pF, in which case use 33pF or 47pF ceramic capacitors for C3 and C4. If in doubt, use 33pF.

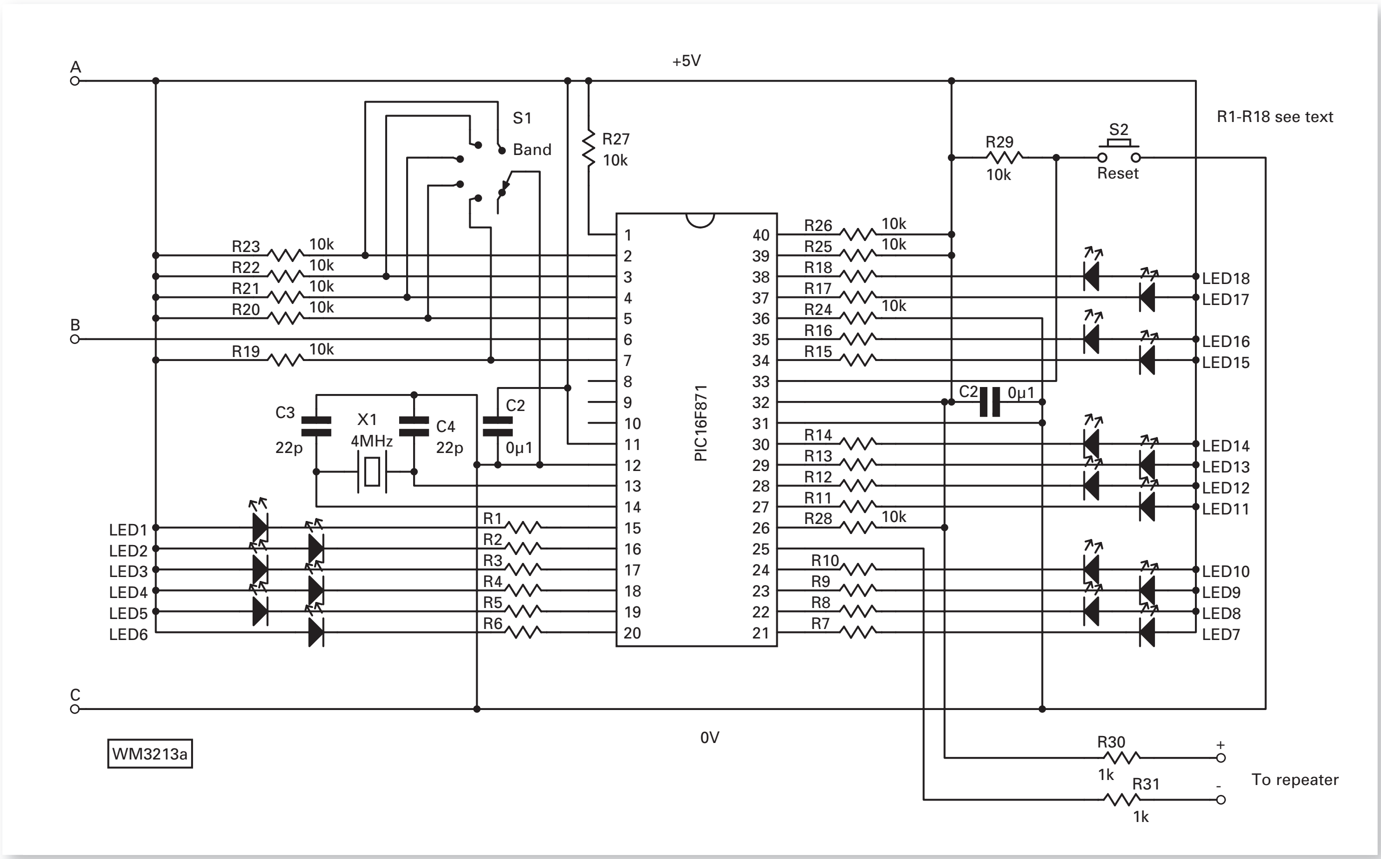

Fig. 3: The PIC clock and its external clock tick generator.

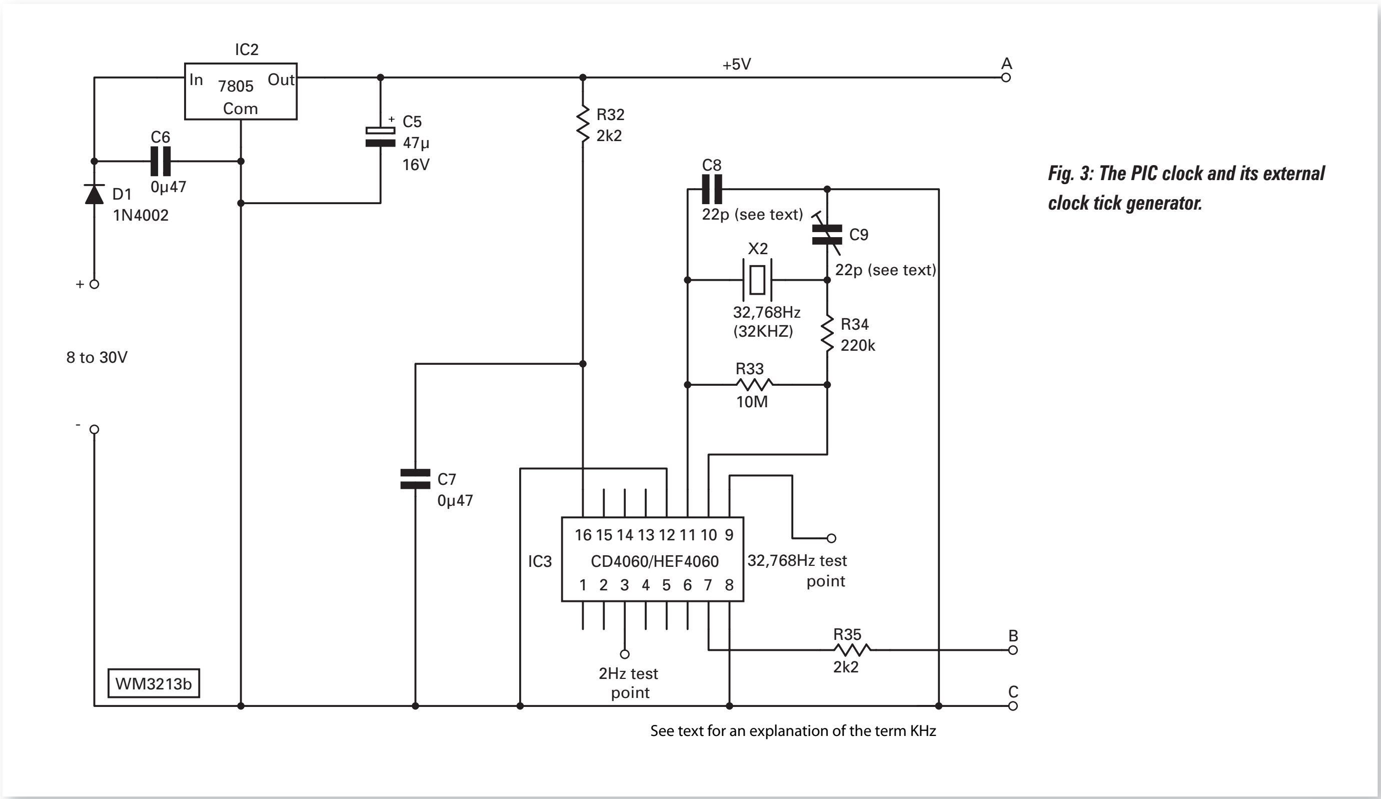

Fig. 3: The PIC clock and its external clock tick generator.

Pin Functions

The function of some of the PIC's individual pins isn't obvious, so I'll run through them now. Pin 1 on the PIC is the microcontroller's external reset input (and high voltage programming input), which is normally held at +5V. We have to momentarily short pin 1 to 0V to reset the PIC's c.p.u. and peripherals.

Pin 26 is the USART's receive input, and because it's not used in this design, it's held inactive (high) by R28. Pins 39 and 40 are used when physically programming the PIC. So, to allow in circuit programming (more about that later), it's useful to configure these pins as inputs if at all possible.

At switch on, the IBP Clock's software checks the

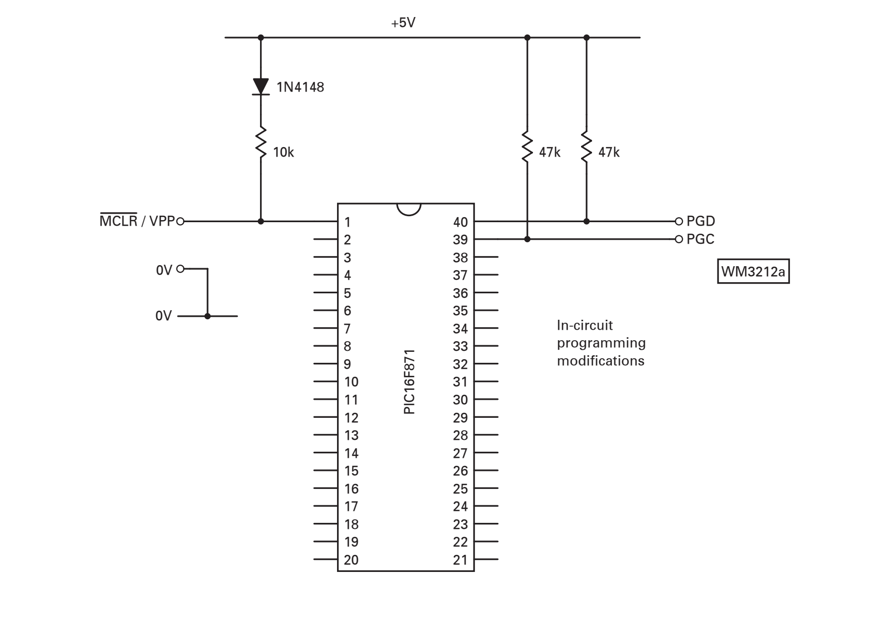

Fig. 4: The PIC can be programmed in-situ with these circuit modifications.

voltages on pins 39 and 40 and these voltages are then used to determine the function of the PIC. The following table should make things clear.

Pin 40	Pin 39	PIC Function
+5V	+5V l.e.d.	Clock
+5V	0V l.c.d.	Clock
0V	+5V l.e.d.	Repeater
0V	0V(Undefined - defaults to l.e.d. Repeater)	

Now to pin 36. This pin is pulled low by R24, and serves two purposes; one Microchip's and one mine. First, pin 36 can be used as a low voltage programming input. This function is active on brand new devices as supplied by the factory. However, most of the PIC kits on the market use the PIC's high voltage programming mode, which involves applying +12V to the reset input pin 1.

To prevent the low voltage programming mode being accidentally activated, it's essential that pin 36 be held at 0V the very first time a new PIC is programmed. Unless specifically required, it's advisable to disable the PIC's low voltage programming option. (Please see the 16F871's data sheet for more information).

In this design, pin 36 is used as a l.e.d. active high/low select pin. The circuit of Fig. 3 shows the l.e.d.s connected between the PIC pins and +5V. The l.e.d. drive is thus said to be active low, in that the PIC pin switches from +5V to 0V to illuminate the required l.e.d. However, if resistor R24 is taken to +5V instead of 0V (remember the warning above though), then the l.e.d. drive is reversed and the PIC pins will switch from 0V to +5V to illuminate the required l.e.d.

I've included this option, not to drive l.e.d.s but to directly drive ULN2804 and similar, open collector buffers. These devices (as used on the original hardware IBP Clock) can switch up to 500mA to ground from a 50V supply. So, if 15mA at 5V isn't enough, then ULN2804s should provide more than sufficient current. Of course, active high outputs can also drive discrete transistors and that's another option. Just remember to use appropriate base resistors!

Finally, the three pins which belong to Port E (Port E only has three I/O pins) are unused and are left unconnected. During initialisation, the IBP Clock software configures them as outputs and sets them low. Anyone modifying the software can use these pins for their own purposes.

Programming The 16F871

April's issue of *PW* featured a review of the **Velleman K8048 PIC Programmer and Experimentation Board** by G4JCP. This kit is eminently suitable for programming the 16F871.

If using this kit, you have two options. Either solder a 40pin d.i.l. socket (preferably turned pin) onto a piece of stripboard, and wire it as described in the kit's documentation. Or you can program the PIC while it's actually in circuit, in the IBP Clock. To do the latter, you'll need to make a few simple modifications, as shown in **Fig. 4**.

First, resistors R25 and 26 need to be increased to 47kΩ. Next, wire a 1N4148 diode in series with R27. Four connections are then required between the IBP Clock and the K8048 board. These connections don't have to be permanent, and in fact, the K8048 p.c.b. is laid out for a 5-pin connector that carries the required programming signals. Only four pins are used, the fifth is a +5V supply, which isn't needed as the IBP Clock has its own +5V supply.

The actual programming procedure is described in a 'help file' supplied with the K8048 kit. If you want to program your own PIC chip and are unfamiliar with the procedure, then I recommend you gain experience by first programming the PIC supplied with the kit. There are two example programs and if both work as they should, then you can try the 16F871. (Please note my comments concerning computers made in the K8048 review, April *PW*).

Source Code & Hex File

The source code for the IPB beacon clock PIC project and the HEX file, which is used by the programming software, is available from **http://www.g4jcp.freeserve.co.uk/** The HEX file should also be compatible with any commercial PIC programmer. Alternatively, ready programmed 16F871 PICs are available from the **Kit Radio Company** - see the components list for details. Blank PIC chips are available from a number of sources, including Maplin. A search of the Internet will reveal others.

Comprehensive data on all PIC devices, development software and debugging tools, application notes and other documentation is freely available from Microchip's website at: **http://www.microchip.com/**

Note: if you want to know more about any particular PIC device, then download its data sheet. And for additional information on mid-range PICs (like those mentioned in the introduction), get the **PICmicro(tm) Mid-Range MCU Family Reference Manual**.

Prototype LED Clock

The prototype l.e.d. clock was built on a piece of stripboard, as shown in **Fig. 5**. Please note that the physical layout of the PIC and its surrounding components closely follow the layout of the circuit diagram. Of course, the l.e.d.s are mounted elsewhere, and pins are fixed in the stripboard at places corresponding to the cathodes of the l.e.d.s on the circuit diagram. Also notice that the l.e.d.s run in sequence, anticlockwise from pin 15 on the PIC.

When building the clock, please make sure you do double check everything! It's a relatively simple circuit, but it's also very easy to make a mistake on stripboard. When testing, make sure you have a stable +5V rail before inserting the i.c.s. Also, ensure those pins that are connected to either 0V or +5V (via resistor or not), really are connected to the proper voltage rail.

Assuming the PIC has been programmed correctly, the clock should work first time. Ruling out wiring errors and faulty components, the only thing that can stop the clock working is the failure of one or both of the crystal oscillators.

The 32KHz oscillator can be checked by wiring a l.e.d. (plus a series resistor) between IC1 pin 3 to 0V. It should flash at 2Hz. To check the 4MHz PIC oscillator, either use a logic probe or an oscilloscope. Or simply use a general coverage receiver tuned to 4MHz. You may have to put the clock quite close to the receiver's antenna to get a good signal.

As with all software projects, there may be bugs in the code. Always use the latest version, which will be made available at the web address mentioned above.

There may also be more features added to the code over time, so it's worth checking the website even if you buy a ready programmed PIC. One last point, please don't expect me to modify the code just because you want a particular feature included. The source code is on the 'Net, so have a go yourself – It's great fun! ●

Components

Resistors (All resistors 0.25W)
R1-18 300Ω minimum (see text)
R19-29 10kΩ
R30,31 1kΩ
R32,35 2.2kΩ
R33 10MΩ
R34 220kΩ
All resistors 0.25W

Capacitors

C1,2	100nF 50V ceramic
C3,4,8	22pF 50V ceramic
C5	47μF 16V electrolytic
C6,7	470nF 50V ceramic
C9	22pF film dielectric trimmer
X1	4MHz HC-49/U (or low profile) crystal
X2	32KHz miniature watch crystal
D1	1N4002
LED1-18	l.e.d.s to suit (see text)
IC1	PIC16F871-I/P [*]
IC2	7805
IC3	CD4060 or HEF4060
S1	Single-pole, 6-way rotary switch
S2	Single-pole, non-latching push-to-make switch

Stripboard, pins, wire, case to suit.

[*] Programmed PIC16F871-I/P microcontrollers for the PIC IBP Clock are available from: **KRC, Unit 11, Marlbrough Court, Westerham, Kent. TN16 1EU. Tel: (01959) 563023, E-mail: kitradioco@aol.com**. Price £7.99 all inclusive to UK addresses. (Overseas 'phone or E-mail for quote.)

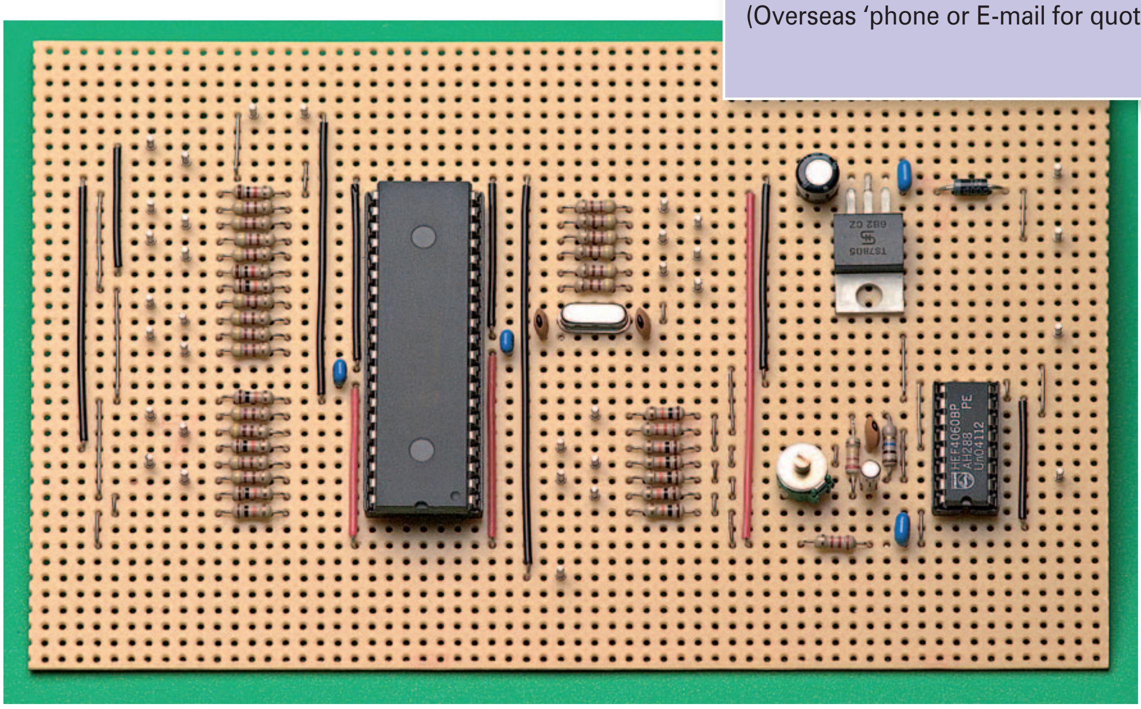
Fig.5: The prototype PIC clock controller.

In part 2, a version with a liquid crystal display will be featured, together with an associated l.e.d. repeater. (The latter allowing a l.e.d. display to be added to the l.c.d. unit.).

Phil Cadman G4JCP writes: "The *PW* team and I are planning to present Part 2 of this project in the July issue of the magazine. In the meantime, as this is our first PIC-based project, everyone involved in preparing it for you will be most interested to hear your comments and what you think of the idea. Please contact the PW offices or myself via **g4jcp@btinternet.com** or at **21 Scotts Green Close, Scotts Green, Dudley, West Midlands DY1 2DX.**