RADCOM TECHNICAL FEATURE

DIY Micro-Processors

By A A Mockford, G8ZGK*

HAVE YOU EVER started a project then found that the functions required were not readily available from the chips that you could buy? Have you had to abandon a project because the number of logic chips or the board size became unmanageable?

THE ONE CHIP MPU

AN EMBEDDED MICRO-CONTROLLER or one-chip micro-processor (MPU) could be the answer. They offer amateurs an ideal medium to design, experiment and try their own unique ideas. To get started with micro-processors requires less design knowledge than analogue or digital hardware design.

There is a range of one-chip MPUs readily available. They are relatively cheap, simple to program and use, and are fully backed by development equipment and manuals. With one MPU chip you can design and build any logic circuit you desire. Just a few examples of circuits that can be constructed using MPUs are listed below:

- Controller for an antenna switcher
- LED driver
- PLL controller
- Keyboard controller
- Serial communication unit

Very few other components are required to make the MPU operate.

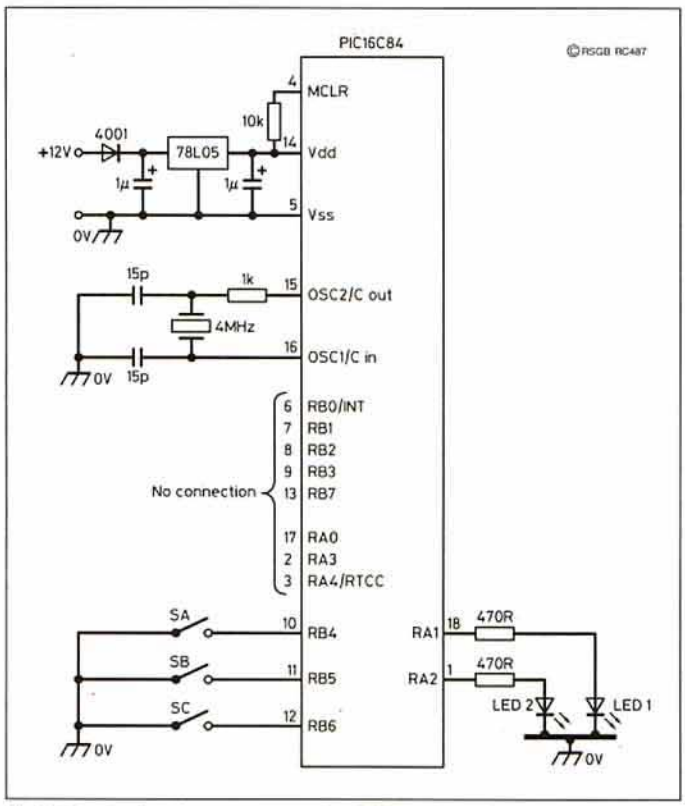

Fig 1: Simple microprocessor control of LEDs using switches.

PIC MPU, 16C84

ALL PARTS DESCRIBED in this article may be obtained from any Maplin store. The MPUs described here are from Microchip Technology Inc and are part of their PIC range. They have been available for several years and the range is slowly being added to. Microchip also supply cost effective, well engineered and well documented development systems.

There are several MPUs in the PIC range offering different speeds, number of connections and different facilities inside. The particular PIC MPU, 16C84, described in this article has an EEPROM memory. You can program it, try its operation, change something and try it again, all within two minutes. It can be reprogrammed thousands of times, therefore is ideal for development. No UV erase light is needed to wipe the memory. In fact a project can be designed developed and put into operation with just one chip costing less than £6.50.

* 47 Kendalls Close, High Wycombe, Bucks, HP13 7NN.

DESIGNING A PROJECT

THERE ARE FOUR STAGES to go through to make any MPU do what you want it to do.

1. Write the instructions telling it what to do.
2. Load these instructions into the MPU.
3. Put the MPU chip in a circuit.
4. Test its operation.

(Repeat steps 1 to 4 until satisfied with the result)

To take these steps in turn:

Step 1. Writing the instructions is easiest done on a PC computer, using a text editor or word processor. Embedded microcontrollers are small MPUs that can not usually be expanded to bigger memories, their instructions are written in Low level assembler language. High level languages like Basic, Algol and C are generally, not very efficient on these small controllers. If you wish, you can plan the instructions and write then on a writing pad, then type them into the PC later. The PIC 16C84 language is very simple. It is called the 16C84 assembler language and consists of only 35 different instructions. To write a program, you use the instructions in the manual, one after the other, so that the MPU obeys them one at a time. The manuals are included in the Picstart-16B development system and give all necessary details of the hardware and software parameters of all the PIC MPUs.

Step 2. Loading the instructions into the MPU requires a PIC programmer. This is included in the Picstart-16B development system. This will cost about £170, but it may be reused for any number of projects. Maybe your local club would consider purchase so all members could use it. It connects to the PC computer and is supplied with manuals, power supply, connecting leads and software. You plug your PIC 16C84 MPU into the Picstart 16B socket, then use the PC keyboard to tell it to load the MPU with the list of instructions. You will need a PC with a serial port, and preferably a hard disk, ie practically any 286, 386 or 486 PC. The Picstart-16B has no requirement for Windows, additional memory or a colour screen.

Step 3. The PIC 16C84 has 18 pins, of which 13 can be used for input and output purposes. **Fig 1** shows the PIC 16C84 in a typical circuit. It has three switches as inputs, and 2 LEDs as outputs. Very few other components are needed to make it work. Its 13 pins can source or sink up to 20mA each, so it can drive the LEDs directly. Pull-up resistors are not necessary on the inputs as some pins can have these internally. For applications where accurate timing is not necessary a simple capacitor and resistor will operate the on-board oscillator. If your design needs accurate generation of pulses or timing, a ceramic resonator or crystal can be used with the oscillator, as shown in Fig 1. A stable 5 volt supply is required. This is easily supplied by a 7805 regulator. The low power type is ideal as the PIC 16C84 uses less than 2mA, mak-

Label	Instruction		Notes
light1	BTFSS	portB,rb4	;is the rb4/SWA pin high or low
	GOTO	light1on	;it is low, turn the LED 1 on
	BCF	portA,ra1	;it is high, turn the LED 1 off
	GOTO	light2	
light1on	BSF	portA,ra1	;turn the LED 1 on
light2	BTFSS	portB,rb5	;is the rb5/SWB pin high or low
	GOTO	light2on	;it is low, turn the LED 2 on
	BCF	portA,ra2	;it is high, turn the LED 2 off
	GOTO	light1	
light2on	BSF	portA,ra2	;turn the LED 2 on
	GOTO	light1	;go back to the 1st instruction

Table 1: A simple list of instructions to turn LED 1 & 2 on or off in response to switches A & B in Fig 1.

ing it perfect for battery powered equipment.

Step 4. When you decide the need for the device, you need to prepare a well defined list of the things the unit will do (and not do). It sounds obvious, but the testing is to ensure that everything on the list has been achieved, no more and no less. Microprocessors are simple beasts and do exactly what they are told to do. The difference between what was meant to happen and what actually happens is called a software 'bug'. Bugs can be harmless or stop the unit from working completely, therefore testing every function of the unit is very important. Once written and de-bugged, the MPU will continue to obey its instructions perfectly every time.

THE PROGRAM

A SIMPLE LIST of instructions is shown in **Table 1** (a program) to turn LED 1 and 2 on or off in response to switches A and B in Fig 1. When the switches are closed, 0 volts or low on the MPU input pins make the output pins high or +5 volts, and the LEDs are on. The instructions are obeyed one after the other. Some instructions are identified by labels that may give some indication to their purpose and allow a jump or GOTO to that instruction.

Some instructions check the state of something, eg a high or low voltage on an input pin, or the number in a register. Other instructions change things, eg the number in a register, the voltage sent to an output pin, or the next instruction to be obeyed. After each instruction a plain language note has been added after a semicolon, to explain what the instruction is doing. Good programming is normally written in small modules (typically, no more than 100 instructions), and includes many comments. This helps writing, testing and changes which often are required at a later date.

SIMPLE APPLICATION

THIS IS A SIMPLE application and is ideal to start learning about an MPU. Each BTFSS (bit test, skip if set) instruction makes the MPU look at an input pin (from a switch) and either do the next instruction or the next but one (skip one instruction). In this way the MPU can make decisions and can take different actions by skipping the next instruction. In this example the different actions are to turn the corresponding LED on with the BSF (bit set) instruction, or off with the BCF (bit clear) instruction. BSF makes the output pin go high to +5 volts, and BCF makes it go low to 0 volts. When all the instructions are obeyed, the MPU starts to do them all over again by returning to the light1 label. Because the example is using a crystal at 4MHz, each instruction takes 1µS to work. Therefore when a switch is operated, it may be up to 10µS before the BTFSS question is asked and the LED responds. This delay is not normally noticed, so response appears to be instant. This example does not need exact timing so an R-C oscillator could have been used, saving on cost and components.

The example shown in Table 1 uses eleven instructions, the PIC 16C84 has space inside for up to 1024 instructions. In addition there are a further 64 memory locations for data, a counter that can count the internal oscillator signal or the signal coming in on pin 3, and 15 other general purpose registers. The example in Fig 1 uses only two of the general purpose registers portA and portB. These are the actual input and output pins on the chip. Only 11 instructions were used and four different ones from the 35 possible, so there is plenty of opportunity to use this controller chip for quite complex controlling functions.

By changing the instructions as shown in **Table 2**, the LEDs 1 or 2 will latch on whenever switch A or B is closed. They will remain on until switch C is closed, this will switch both LEDs off again. It is very easy therefore, to change the function of a unit by a software change without having to change any hardware.

One important feature of the 16C84 controller is an 'interrupt structure'. This enables the MPU to respond immediately to a change on one or more of its input pins. For example, Table 1 constantly checks the state of the input switches A and B and does little else, so in this case, an interrupt routine would have no advantage. If there were many more instructions it could be some time before the questions were asked about the state of the input pins. This response delay could be unacceptable in some circuits where timing is critical.

In this case, an interrupt routine could be beneficial. An interrupt works by the change on an input pin causing the next instruction that would normally be obeyed, to be stored, and a separate set of instructions (the interrupt instructions) to be obeyed instead. It would be the interrupt instructions that would do whatever was necessary, now that the change is detected. When all the interrupt instructions have been obeyed, the RETFIE instruction recovers the stored instruction, it is obeyed, then the sequence through the instructions continues as before. An example is given in **Table 3**.

Label	Instruction		Notes
light1	BTFSS	portB,rb4	;is the rb4/SWA pin high or low
	BSF	portA,ra1	;it is low, turn the LED 1 on
	BTFSS	portB,rb5	;is the rb5/SWB pin high or low
	BSF	portA,ra2	;it is low, turn the LED 2 on
	BTFSC	portB,rb6	;is the rb6/SWC pin high or low
	GOTO	light1	;its high so no action
	BCF	portA,ra1	;its low, so turn LED 1 & 2 off
	BCF	portA,ra2	
	GOTO	light1	;go back to 1st instruction

Table 2: By changing the instructions, shown in Table 1, the LEDs 1 or 2 will latch on whenever switch A or B is closed.

SOFTWARE SIMULATOR

WHEN YOU WRITE many instructions there are many opportunities for a bug to be included, sometimes many bugs. To assist testing, a 'software simulator' is included in the Picstart-16B development system. This is a PC simulation of the PIC 16C84. By loading the instructions to the simulator program instead of to the Picstart programmer, it is possible to see all the registers and internal operation of the PIC 16C84 and to step through the instructions one at a time. This enables you to see each register and the changes, at each step of the program. The simulator software is included with the Picstart-16B development system.

An embedded microcontroller is a very adaptable controlling device. **Fig 2** shows examples of a 4-digit, 7-segment display connected to the PIC 16C84, a high current 12V

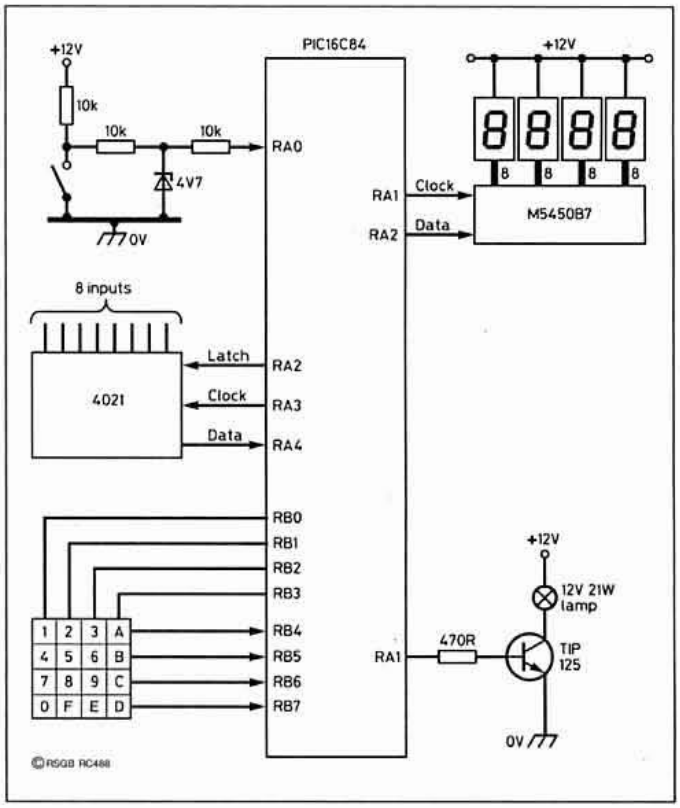

Fig 2: Microprocessor control of a seven segment displays using a keypad.

IDEAS FROM ABROAD

TRANSLATED AND EDITED BY ERWIN DAVID, G4LQI

A full-size **7MHz HB9CV** beam is not beyond the ability of the serious home constructor provided adequate mechanical precautions are observed. **Daniel Caudroy, F6BXC** described how he did it in *Radio-REF* 10/93.

THE ELECTRICAL PROPERTIES of the HB9CV can be found in earlier amateur publications [1]. These also contain mechanical designs for VHF and UHF and, less frequently, for the higher HF bands [2]. The HB9CV is just as useful for 7MHz DX. Home construction was a challenge, which I met as described. [The mechanical design of this antenna is applicable to other full-size two-element close-spaced 7MHz arrays. Available space does not permit reproduction of detail drawings and photographs - *G4LQI*].

SIZE AND WEIGHT

THE ANTENNA FITS in a rectangle of 21.5 x 5.5m. See **Fig 1**. An uncluttered assembly area of that size is required, preferably near the tower. Once aloft, the antenna has a turning radius of 11.5m; make sure it does not infringe on your neighbour's property. [In the UK, the advice of the local planning authority should be sought to avoid a planning application being rejected out of hand - *G4LQI*]

The antenna elements are made of dural tubing; the boom is a galvanized steel tube of 50 x 47mm (OD x ID), as used for central heating installations; if the boom were to be made of aluminium alloy as well, a larger diameter would be required. The assembled weight of the antenna is near 60kg. The maximum wind area is approx $2m^2$.

For a rotary array of this size, commercial hardware sold for beams for 14MHz and above is definitely inadequate. The weights and dimensions given will serve as advance notice of the magnitude of the project. If the following guidelines are adhered to, however, you will not have to worry during every gale. My beam has been up for two and a half years now and has survived winds up to 120km/h.

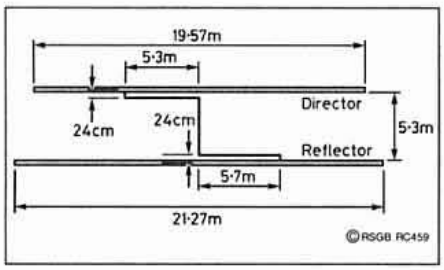

Fig 1: General layout and nominal dimensions of the 7MHz HB9CV. F6BXC got best results with elements of equal length and phasing lines as shown.

THE TOWER

FOR THE DESIRED LOW-ANGLE radiation, the antenna height should be at least a half wave-length, ie 21m. My tower is 16.5m high.

A self-supporting lattice tower should have a bottom section of 70 or 80cm across and be set on a concrete base of $2m^3$ [3].

For a guyed lattice tower, **Fig 2**, a cross section of 35 - 40cm is suitable; it should be solidly anchored to a concrete base 80cm square and 50cm deep. Its guys must take not only the wind load on the tower and antenna, but also the torque exerted by the antenna on the tower through the rotator. I use 8mm steel guys, electrically broken up by egg insulators; the guy anchors are appropriately massive [3]. I must warn against the use of nylon rope for guying. Fibreglass is OK if not stressed beyond 20% of its rated breaking strength.

To reduce the effect of the torque, an anti-twist crossbar is used. It is a 6m long tube, of the same material as the boom of the antenna, which is rigidly bolted to the base plate of the rotator. Guyed to two anchors, **Fig 3**, this crossbar has the effect of increasing the tower cross section at the level of maximum torque. Other, smaller antennas may be mounted on the cross bar.

ROTATOR AND STUBMAST

BECAUSE COMMERCIAL AMATEUR rotators are inadequate, a second-hand industrial motor with a built-on reduction gear was used. 1/3 - 1/2 RPM is about right. If too fast, an additional reduction gear may be used. The output shaft, which is directly coupled to the stub mast, should be at least 30mm dia.

The control box should provide a delay to prevent the motor from reversing direction before the antenna has stopped completely. When the antenna is not used, a servo, coupled to a wind vane, parks the antenna in the position presenting the minimum area to winds over 40km/h.

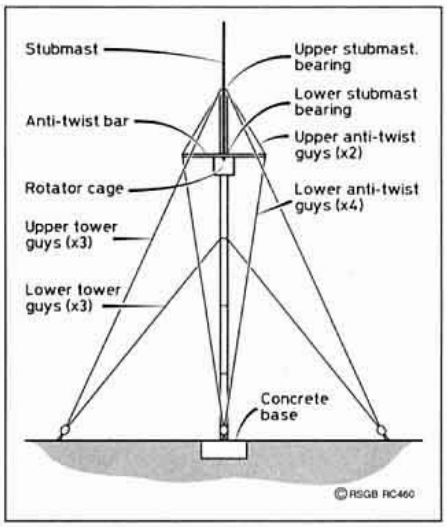

Fig 2: Diagram of the guyed tower, rotator, stub mast, and anti-twist crossbar.

---

lamp driver connection, the connection to a 16-key keypad, a serial shift register to provide eight inputs, but using only three pins on the MPU, and an input from a switched 12V source.

There have been many cases of badly designed microprocessors radiating noise over bands and disrupting communications. It is worth noting that single chip microcontrollers have all their high frequency switching elements internal. There are no bus and control lines to other chips to radiate noise, therefore they are inherently quiet. The cost of the Pic start-16B is £170 including VAT from Maplins.

Label	Instruction		Notes
normal	GOTO	normal	;this simulates normal operation ;instructions that are obeyed ;repeatedly.
interrupt	BTFSS	portB,rb4	;is the rb4/SWA pin high or low
	BSF	portA,ra1	;it is low, turn the LED 1 on
	BTFSS	portB,rb5	;is the rb5/SWB pin high or low
	BSF	portA,ra2	;it is low, turn the LED 2 on
	BTFSC	portB,rb6	;is the rb6/SWC pin high or low
	RETFIE		;its high so end the interrupt
	BCF	portA,ra1	;its low so turn LED 1 & 2 off
	BCF	portA,ra2	
	RETFIE		;end the interrupt

Table 3: Program using interrupt instructions.

COMPONENTS AVAILABLE

IT IS IMPOSSIBLE in a few pages to provide an 'introduction to programming' course, however, I hope that this article has whetted your appetite to experiment. All parts described in this article may be obtained from any Maplin store. The mail order address is Maplins, PO Box 3, Rayleigh, Essex SS6 8LR or telephone 01702 554161. If you wish to try your hand at programming PIC microcontrollers an 'Aid To Easier Programming' disk is available from the author (see below) that includes many maths routines that have been tried and tested. It is much easier to use routines that have been written and debugged than to re-invent the wheel. The disk also includes several useful programming suggestions and has a simple layout format to help your first attempts at programming to be successful.

NOTES

MICROCHIP AND PIC are trademarks of Arizona Microchip Inc. Diagrams and data are reproduced by kind permission of Arizona Microchip Inc.

SOFTWARE AVAILABILITY

FOR THE *Aid To Easier Programming* disk send a cheque for £5 to A A Mockford, 47 Kendalls Close, High Wycombe, Bucks HP13 7NN. ♦