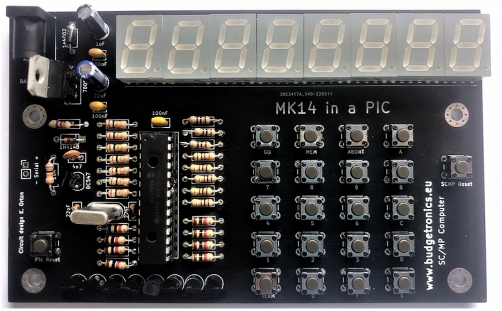
BudgeTronics

CAMBRIDGE MK14 in a PIC

An affordable MK14 for everyone to enjoy!



Go back to the seventies and program this computer as was done more then 40 years ago.

INTRODUCTION

The MK14 (Microcomputer Kit 14) was a computer kit sold by Science of Cambridge in the United Kingdom, first introduced in 1977. The price was very low for a complete computer system at the time, and Science of Cambridge eventually sold over 15.000 kits.

Few of these kits have survived and if you want to own an original MK14 to play with you have to pay very high prices for them. Because this is not in everyones reach here we introduce the MK14 in a PIC designed by the late Karen Orton. It is a very affordable building kit which you can program like a real MK14. Nice to experiment with and learn how to program the SC/MP CPU like in the old days.

Contents building kit

1x PCB
1x programmed PIC16F876
1x IC socket 28 pins
8x 7 segment LED display
22x pushbutton switch
8x 100 Ohms resistor (brown, black, brown)
8x 1K resistor (brown, black, red)
1x 4K7 resistor (yellow, violet, red)
8x 10K resistor (brown, black, orange)
3x 100nf ceramic capacitor
2x 27pf ceramic capacitor
1x 4n7 ceramic capacitor
1x 1uF electrolytic capacitor

1x 22uF electrolytic capacitor
1x quartz 20 Mhz
9x BC547 transistor
1x 4002 Diode or equivalent
1x 1N4148 diode
1x 7805 voltage regulator
1x power barrel jack

BUILDING INSTRUCTIONS

On the PCB you can see exactly where all the components go. We will give you a step by step instruction here and tips what to watch out for.

1. First solder all the resistors.
2. Solder the two Diode and place the stripe of the diode at the same side as printed on the PCB.
3. Solder all ceramic capacitors and take care to place the right value in the right spot.
4. Solder the 2 electrolytic capacitors and watch the polarity!! So the negative sign on the capacitor at the negative side on the PCB. Short leg is most of the time negative.
5. Solder the IC socket and make sure the little notch on top is placed as shown on the PCB. So notch up.
6. Solder all the BC547 transistors in place and also watch that the flat side is aligned with the silk screen on the PCB.
7. Place the barrel jack and use enough

solder.

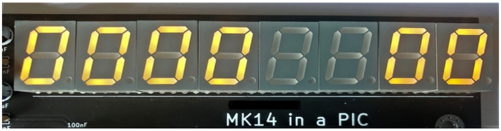
8. Place the voltage regulator. Align the ironside with the screen print on the PCB. So the iron side faces the barrel jack.
9. Solder the quartz in its place.
10. Place all the LED display's and take care the **<u>DOT</u>** on the is pointing down to the right side otherwise the displays will not work. Look at the screen print of the PCB.
11. Finally you can solder all the 22 switches and place the PIC in its socket with the little notch pointing UP.

Examine all solder connections for solder bridges or bad connection and if all is well you are finished.

STARTING UP THE MK14

Having selected a suitable 9 Volts power supply the most important precaution to
observe is that of correct polarity with a centre +. You will not damage the circuit if polarity is wrong because of the protecting diode but the MK14 will not work.

After plugging in the power supply the display stays dark. Press the SC/MP RESET button and if all is well the following display is shown:



The left-hand group of four digits is called the address field and the right-hand two digit group is the data field.

If this works, congratulations you now have a working MK14!

ENTERING AND RUNNING A PROGRAMS

Here we enter a simple program to see how well the MK14 is working and to learn how to enter programs the old fashioned way.

The program is very simple. We want to perform the following calculation : Add 29 to 42 and store the result in a memory location.

The program is as follows:

ccl	clear the carry/link bit so it isn't added in
ldi $42	load 42 in hex into the accumulator
dai $29	add hex 29 to it
st [result]	store it in the result memory location
xppc 3	return to the SCIOS ROM
[result] byte 0	this memory location will contain the result of the calculation

In the past, the next stage was to convert this to machine code. This was done by hand. This program will start at memory location $F20, as $F00..$F11 are used by the SCIOS ROM "Monitor Program".

```
0f20   ccl        02       Clear the Carry/Link bit
                           Load the accumulator with
0f21   ldi $42    c4 42    42 hexadecimal
0f23   dai $29    ec 29    decimal add 29
0f25   st 0f27    c8 02    save 2 on from PC
0f27   xppc 3     3f       return to SCIOS
0f28   [result]   00       result goes here
```

The only thing requiring explanation is the store instruction. This is "PC Relative", so +2 means 2 on from the Program Counter. 0F26 (where the offset is) +2 gives you 0F28, where the result goes.

<u>Technical Note:</u> Unlike almost all other CPUs, the SC/MP CPU increments the PC (Program Counter) ***before*** the instruction is fetched, so when the ST (STORE) instruction is being executed, the PC points to 0F26 and not 0F27 as you would expect on most other processors.

<u>Entering and running the program</u>

Power up the MK14 and press the SC/MP RESET button to start fresh.

1. If you type 0 F 2 0 the current display will change to 0F20.
2. Press TERM to enter data entry mode
3. Enter the first opcode 0 2
4. Press MEM to advance the address with one
5. Enter the second opcode C 4
6. Press MEM to advance the address again
7. Enter the next opcode 42 at address 0f22 as shown on the display
8. Press MEM to advance to the next address 0F23
9. Repeat these steps until each of the bytes has been entered, including the 'result' byte. At the end, the screen should display the address 0F29

To check it, press ABORT to return to Address mode, and enter 0 F 2 0 again. You can use the MEM button to step through the program showing each byte in turn.

To run the program, press ABORT to go into address mode. Enter the start program address 0 F 2 0. The display should read 0F20 02. Press GO to run the program. The instructions are run. On exit, the address is set to the address after the XPPC 3 instruction, which is the result byte at 0F28. If all has worked correctly it should display 0F28 71 (which is the result of 29+42).

So in short:

TERM= enter data entry mode

MEM= advance address with 1 step

ABORT= exit and return to address mode

GO= run program from address as shown in display

<u>Unlike the real MK14 the “MK14 in a PIC” remembers the programs which are entered even if you power down. The code will be remembered after a fresh startup so you do not have to type it in every time you pull the power.</u>

Further reading

You can find the manual of the Cambridge MK14 for download on the Internet if you look for it. Make sure you have the manual for the MK14 that states on page 3, last sentence, “Note: This manual applies to kits with Issue 4 or 5 boards and the revised SCIOS monitor”. There is an older version and there are some differences. In this manual there are some programs to type in. The games section is a nice place to start with.

If you look for SC/MP with Google you will find other

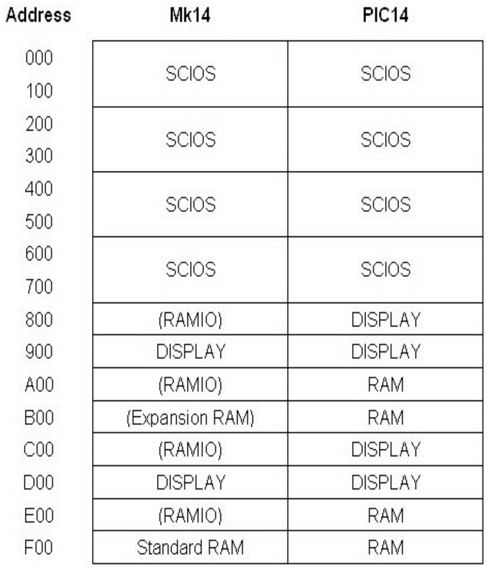
PDF manuals for download which will explain the SC/MP programming in more detail if you want to learn it.

Differences between the real MK14 and the "MK14 in a PIC" kit

The MK14 in a PIC was developed by the late Karen Orton who described the differences as follows.

The MK14 in a PIC is a 'cycle perfect' emulation insofar as every SC/MP instruction executes in the same time as it would on a real SC/MP chip. However, the emulation departs in a number of ways from true Mk14 behaviour:

The memory map of the PIC14 is not identical to the Mk14 although it is broadly compatible insofar as the ROM, display and standard RAM are to be found in their correct Mk14 locations. The PIC14 memory map is shown in an associated table which includes the Mk14 memory map alongside for comparison. Note that items not present in the unexpanded Mk14 are shown in brackets (Note also that, although the PIC14 RAM features several times in the memory map, they are just copies – there is only 256 bytes of RAM in total).



Address	Mk14	PIC14
000	SCIOS	SCIOS
100		
200	SCIOS	SCIOS
300		
400	SCIOS	SCIOS
500		
600	SCIOS	SCIOS
700		
800	(RAMIO)	DISPLAY
900	DISPLAY	DISPLAY
A00	(RAMIO)	RAM
B00	(Expansion RAM)	RAM
C00	(RAMIO)	DISPLAY
D00	DISPLAY	DISPLAY
E00	(RAMIO)	RAM
F00	Standard RAM	RAM

PIC14 Memory Map

Source:Website Karen Orton

Sense A interrupts are not implemented.

SC/MP paging is not implemented. Both program counter increment and pointer arithmetic will carry normally into the top four bits of any address calculation. Mk14 behaviour follows from the fact that these four bits play no part in address decoding. Nonetheless, if a program examines these bits, they may be found to be different to what they would be in an Mk14.

The display column latch is not updated by reads of the display. This aspect of Mk14 behaviour required programs to exercise discipline when scanning the display and keyboard, otherwise 'ghost' digits could be generated. This discipline is not needed with the PIC14. Problems can occur with programs that implement column latch update using display read cycles, though the existence of such a program is never seen.

The emulation is of a 4MHz SC/MP. The Mk14 had a 4.43MHz crystal and clock. In practise this will just cause programs to run 11% slower than they would on an Mk14. For those who want a true 4.43MHz emulation, they could try fitting a 22.15MHz crystal to the PIC. This is driving the PIC beyond its specification but might work for some chips.

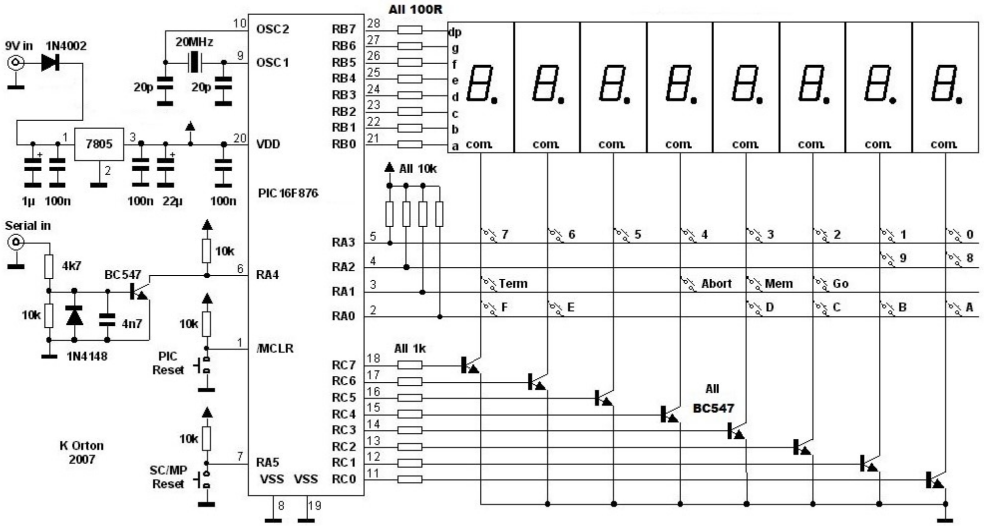
llegal SC/MP instructions cause a reset. This is actually an advantage – in the Mk14 a program crash went completely undetected and often lead to total RAM obliteration.
The emulation is cycle perfect only for programs executing in RAM. ROM execution runs slightly slower due to approximately an extra cycle

overhead on program fetches from ROM (note however, that data reads from ROM by programs running in RAM execute at the correct speed).

The MK14 in a PIC has a couple of additional features over the Mk14:

The RAM, as programmed through the SCIOS monitor, is non-volatile. This means that programs are retained through power cycles. It also means that a program can be recovered if it overwrites itself.

A serial download facility allows Intel hex files to be sent to the PIC14. This feature is entered automatically following a **PIC reset**. If this feature is not required, then it can be exited by pressing the SC/MP reset key, in which event the RAM is loaded from non-volatile memory. If you want to try this you can use the serial connection on the left side of the PCB. Remember this is only one way. The MK14 only receives and can not sent. Files should be transferred at 9600 baud, 8 data bits, 1 stop bit and no parity. As the link is one way, no flow control or handshake should be enabled for file transfer.



9V in
1N4002
20MHz
20p
20p
7805
1
2
3
1µ
100n
100n
22µ
100n
10
9
20
OSC2
OSC1
VDD
PIC16F876
Serial in
4k7
10k
BC547
4n7
1N4148
10k
6
RA4
10k
1
/MCLR
PIC Reset
10k
7
RA5
SC/MP Reset
VSS
VSS
8
19
K Orton 2007
All 100R
RB7 28
RB6 27
RB5 26
RB4 25
RB3 24
RB2 23
RB1 22
RB0 21
dp
g
f
e
d
c
b
a
com.
All 10k
RA3 5
RA2 4
RA1 3
RA0 2
7
6
5
4
3
2
1
0
9
8
Term
Abort
Mem
Go
F
E
D
C
B
A
All 1k
RC7 18
RC6 17
RC5 16
RC4 15
RC3 14
RC2 13
RC1 12
RC0 11
All BC547

RA0	Active low keyboard row sense (as per D4 data bus assignment on Mk14)
RA1	Active low keyboard row sense (as per D5 data bus assignment on Mk14)
RA2	Active low keyboard row sense (as per D6 data bus assignment on Mk14)
RA3	Active low keyboard row sense (as per D7 data bus assignment on Mk14)
RA4	Serial download input (9600 baud)
RA5	Active low SC/MP reset
RB0	Active high segment a drive
RB1	Active high segment b drive
RB2	Active high segment c drive
RB3	Active high segment d drive
RB4	Active high segment e drive
RB5	Active high segment f drive
RB6	Active high segment g drive
RB7	Active high segment dp drive
RC0	Active high column 0 drive (rightmost)
RC1	Active high column 1 drive
RC2	Active high column 2 drive
RC3	Active high column 3 drive
RC4	Active high column 4 drive
RC5	Active high column 5 drive
RC6	Active high column 6 drive
RC7	Active high column 7 drive (leftmost)

Port A, B, C assignments

EXAMPLE GAME - DUCK SHOOT

Shoot ducks flying over the LED display by hitting key with number corresponding to their position: 7 leftmost, 0 rightmost. If you miss another duck appears.
Duck=061 – segment pattern
Disp=0d00 – display address

```
0F12 C4
0F13 0D
0F14 35
0F15 C4
0F16 00
0F17 31
0F18 C4
0F19 01
0F1A C8
0F1B F4
0F1C C4
0F1D 10
0F1E C8
0F1F F1
0F20 C4
0F21 00
0F22C8
0F23 EE
0F24 C4
0F25 08
0F26 01
0F27 C0
0F28 E7
0F29 1E
0F2A C8
0F2B E4
0F2C 94
0F2D 04
0F2E C4
0F2F 61
0F30 90
0F31 02
0F32 C4
0F33 00
0F34 C9
0F35 80
0F36 8F
0F37 01
0F38 C0
0F39 D8
0F3A 9C
0F3B 0E
0F3C C1
0F3D 80
0F3E E4
0F3F FF
```

```
0F40 98
0F41 08
0F42 C8
0F43 CE
0F44 C0
0F45 CA
0F46 E4
0F47 80
0F48 C8
0F49 C6
0F4A 40
0F4B 03
0F4C FC
0F4D 01
0F4E 94
0F4F D6
0F50 B8
0F51 BF
0F52 98
0F53 C8
0F54 C4
0F55 07
0F56 90
0F57 CE
```

www.budgetronics.eu