Writing Machine Code with Asm80.com

Tutorial

For the Small Computer Monitor v0.3 on RC2014

Software and Documentation by Stephen C Cousins Edition 1.0.0

Writing machine code programs directly with the Small Computer Monitor is possible, but soon gets tedious when the program grows beyond a handful of bytes in length.

At this point it becomes desirable to use a separate assembler program to generate the machine code.

Here we look at using the on-line assembler at www.Asm80.com.

This site provides an assembler which can compile code for a number of different processors and also has some emulators built in to test your code.

In this document we look at how to use the site to write machine code programs for RC2014 Z80 systems running the Small Computer Monitor (v0.3).

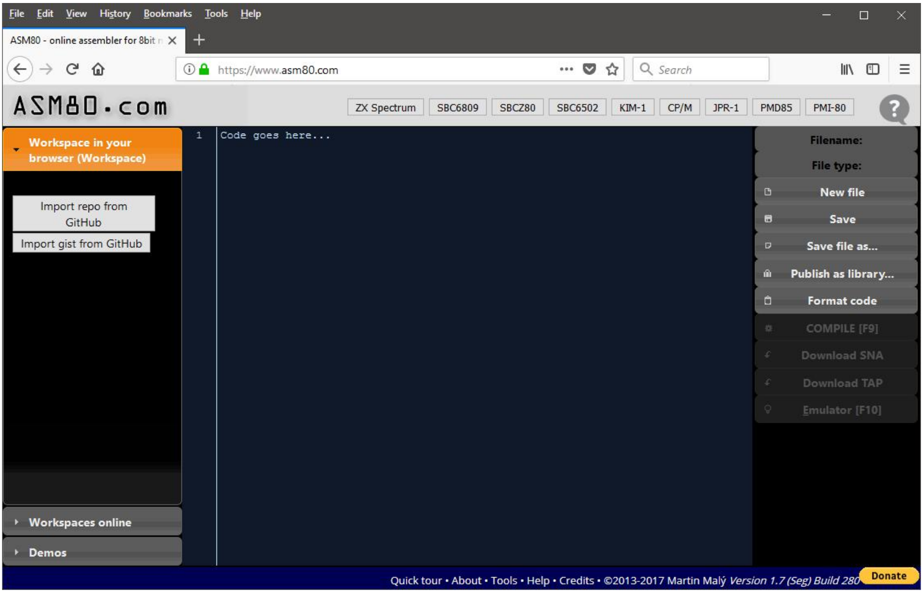
The RC2014 system used is a fairly basic set up with the optional digital I/O module to give us some LEDs to control.

We will use the terminal program Tera Term v4.96 to communicate with the RC2014 system.

The example program will simply turn ON or OFF the specified LEDs.

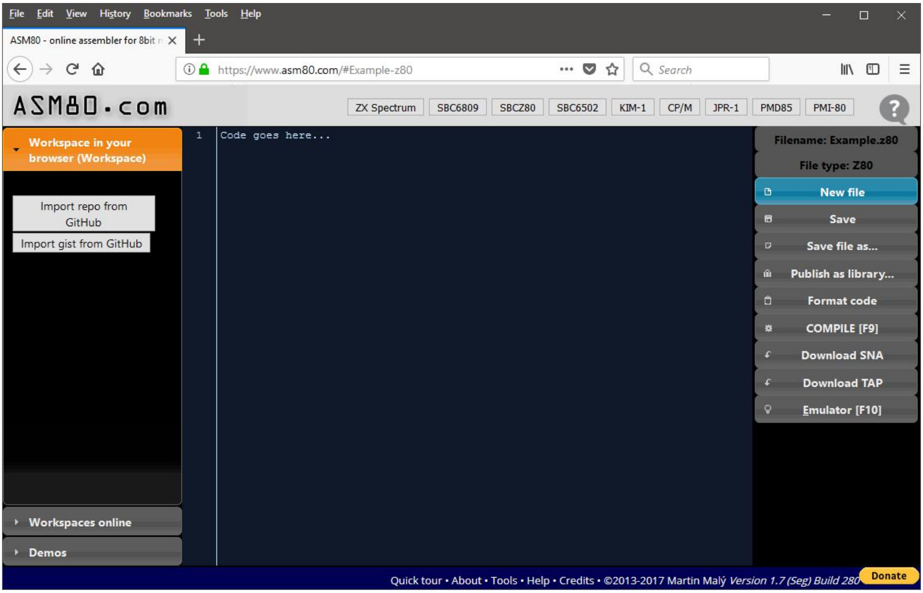
To start just use your web browser to visit www.asm80.com

The screen should look like this.



```
Code goes here...
```

Select the button “New file” and enter the file name “Example.z80”.



```
Code goes here...
```

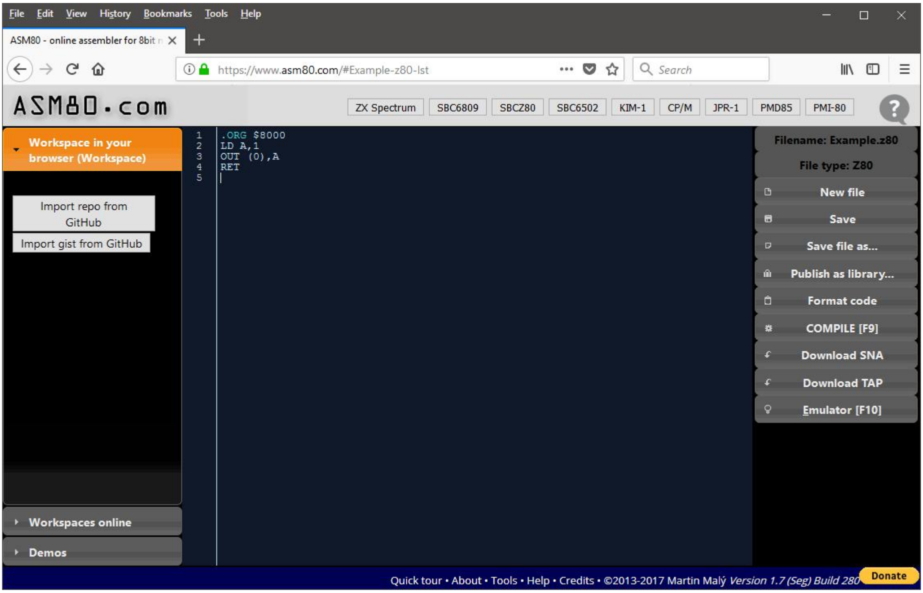
Filename: Example.z80

File type: Z80

The file name has the file type “.z80” to tell the assembler the program is to be written in Z80 assembly language.

Now type the following Z80 assembly language instructions in the main code area.

```
.ORG  $8000
LD  A,1
OUT  (0),A
RET
```



```
.ORG $8000
LD A,1
OUT (0),A
RET
```

Filename: Example.z80

File type: Z80

The above assembly language instructions are as follows:

.ORG $8000 This tells the assembler where the program is to reside in memory
The address can be the start of any unused area of RAM

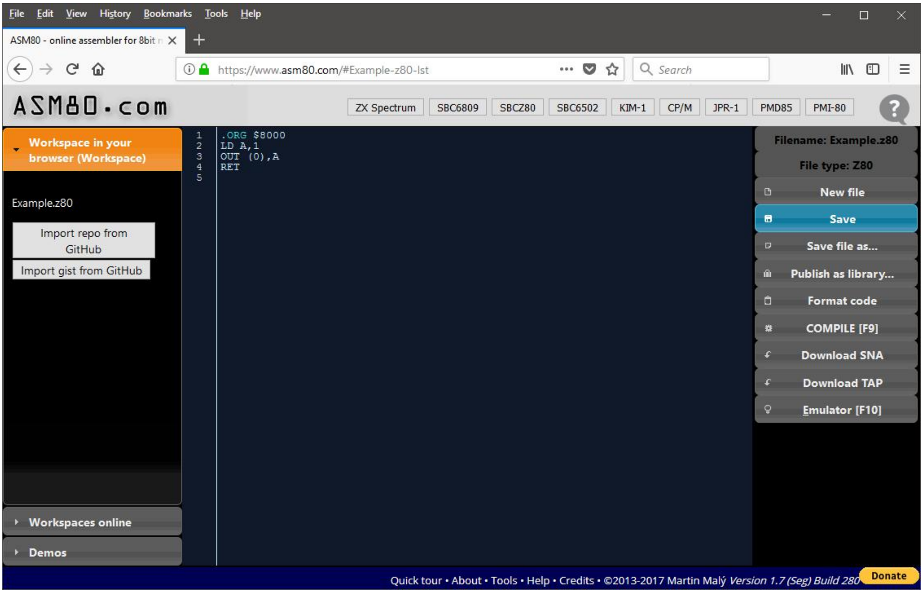
LD A,1 Loads the A register with the value 1
This will be the data byte written to the LEDs

OUT (0),A Outputs the value in the A register to I/O port zero
The RC2014 target hardware has 8 LEDs at output port zero

RET The return instruction ends the program
A program on the Small Computer Monitor is written as a subroutine

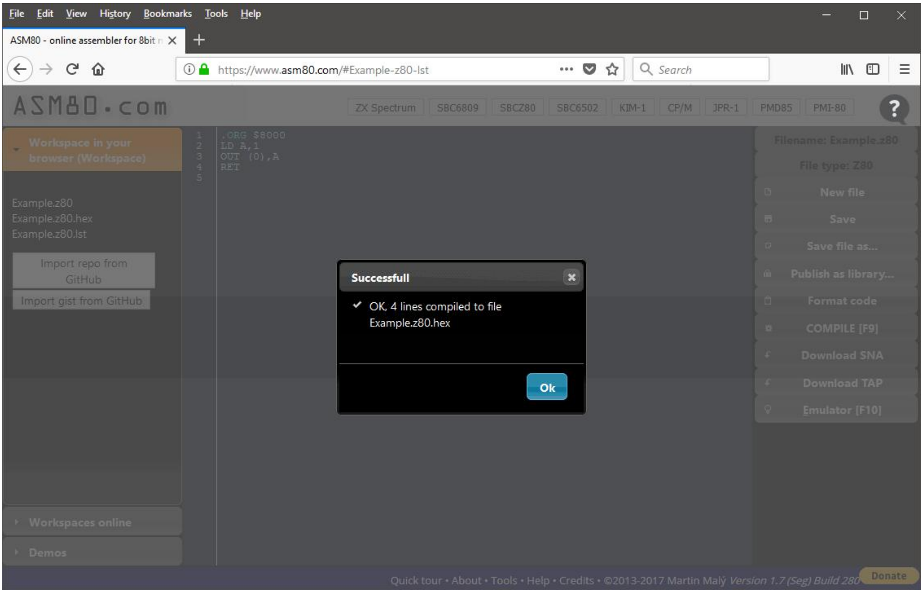
Select the button “Save”.

The file name will be shown in the panel on the left of the screen.



```
.ORG $8000
LD A,1
OUT (0),A
RET
```

Now select the button “COMPILE [F9]” to compile (or assemble) the code.



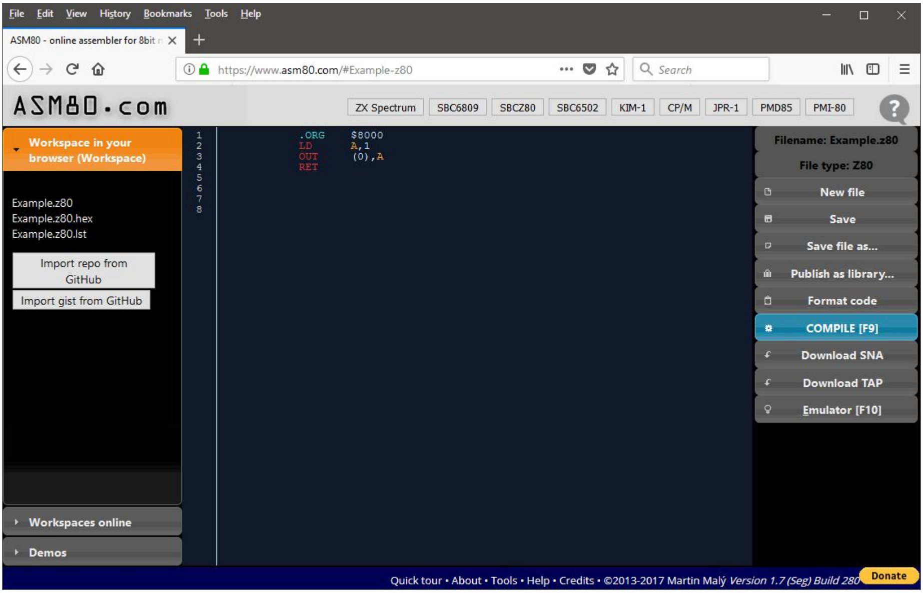
```
.ORG $8000
LD A,1
OUT (0),A
RET
```

Successfull

✓ OK, 4 lines compiled to file Example.z80.hex

The panel to the left should now show two additional files:

- Example.z80.hex
- Example.z80.lst



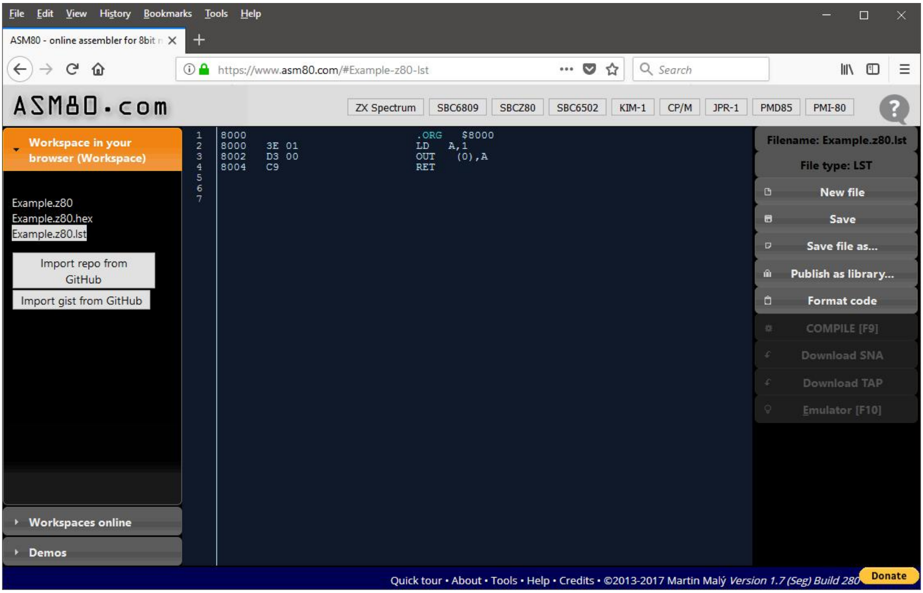
Example.z80
Example.z80.hex
Example.z80.lst

```
        .ORG    $8000
        LD      A,1
        OUT     (0),A
        RET
```

Filename: Example.z80

File type: Z80

Select the file “Example.z80.lst” to see the compiled code.



Filename: Example.z80.lst

File type: LST

```
8000                    .ORG   $8000
8000    3E 01           LD   A,1
8002    D3 00           OUT   (0),A
8004    C9              RET
```

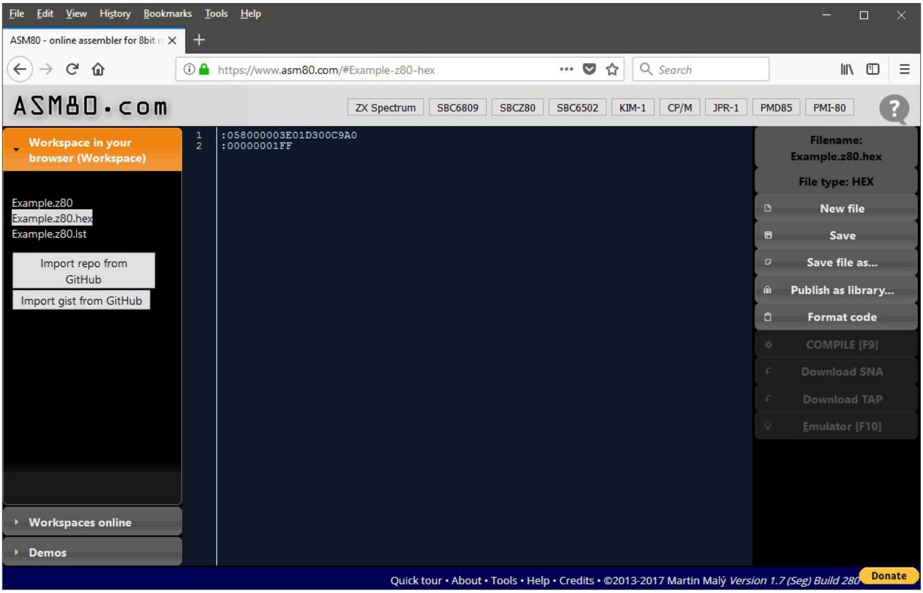
The list file shows the machine code program generated from the Z80 assembly language statements entered earlier.

The first column is the address in memory shown in hexadecimal.

The next column(s) are the machine code instruction bytes shown in hexadecimal.

The final column shows the assembly language instructions entered earlier.

Select the file "Example.z80.hex" to see the hex file.



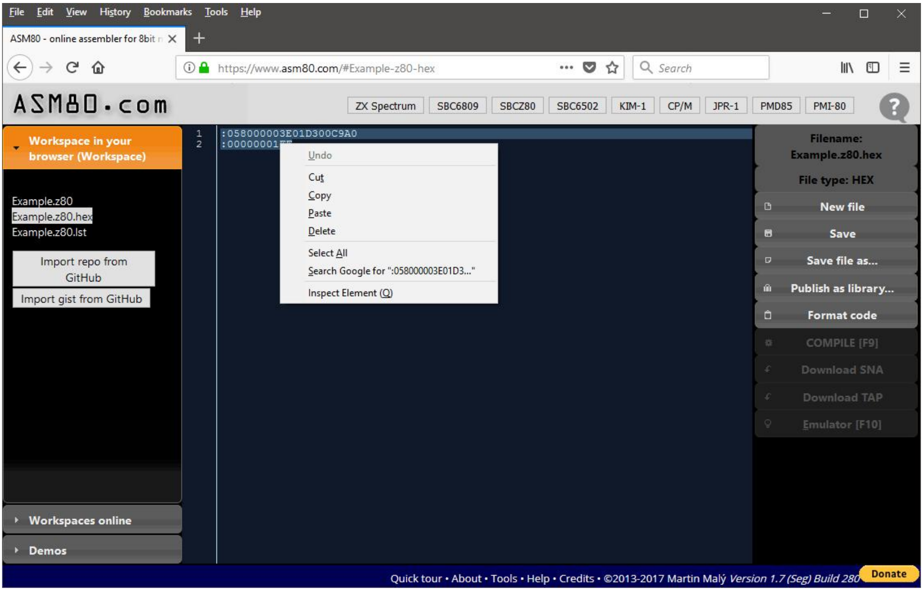
```
:058000003E01D300C9A0
:00000001FF
```

Filename: Example.z80.hex

File type: HEX

The hex file contains the machine code program bytes in Intel Hex format. This is a commonly used format for storing and transferring machine code programs. It can be opened by most EPROM programmers and can also be sent directly to the Small Computer Monitor.

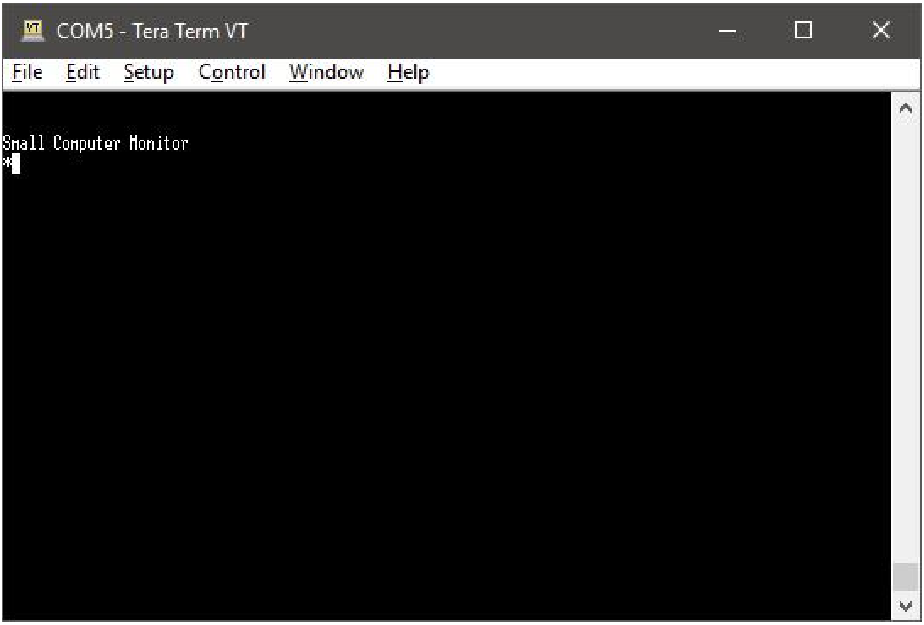
Select (highlight) the hex file text and copy it to the clipboard.



```
:058000003E01D300C9A0
:00000001FF
```

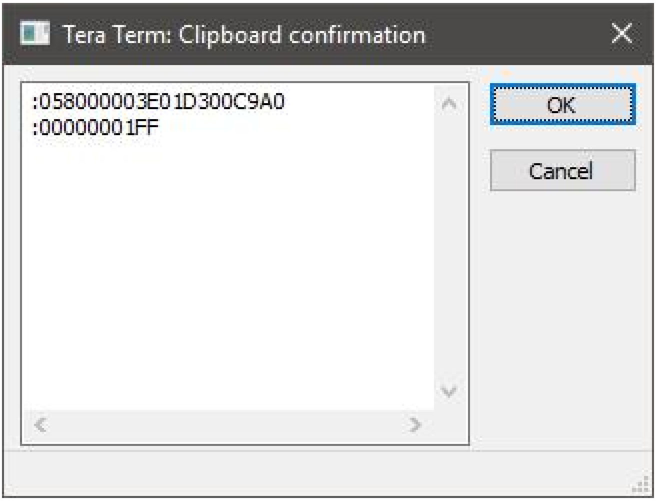
We now have the Intel Hex file text on the clipboard ready to transfer to the target RC2014 system via a terminal program.

Now switch to a terminal window with the RC2014 connected and running the Small Computer Monitor. In this example we are using Tera Term.



```
Small Computer Monitor
*
```

Select the menu item: Edit, Paste and select OK.

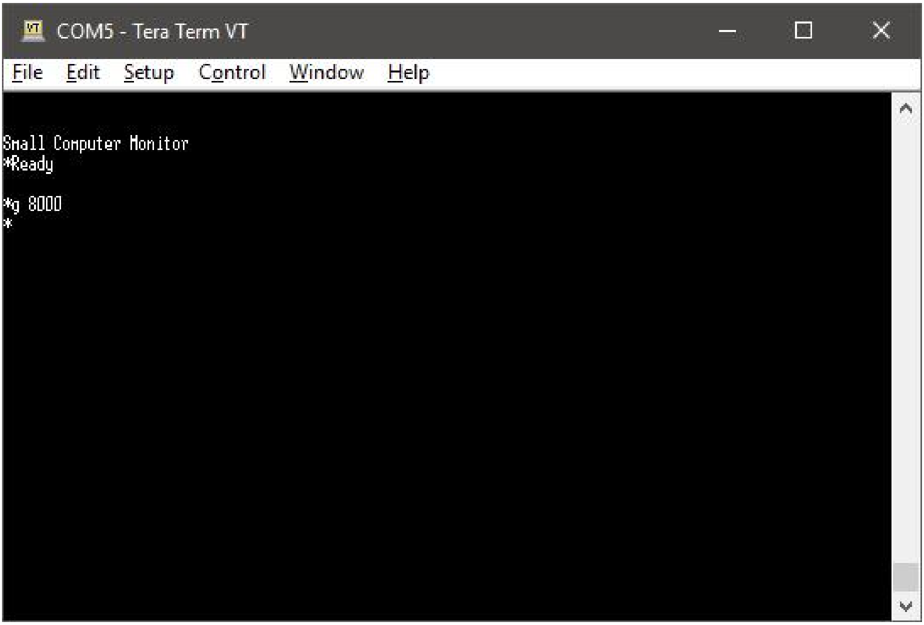


```
:058000003E01D300C9A0
:00000001FF
```

The terminal window should then show the "Ready" message indicating the hex file has been accepted.

Now type the command:

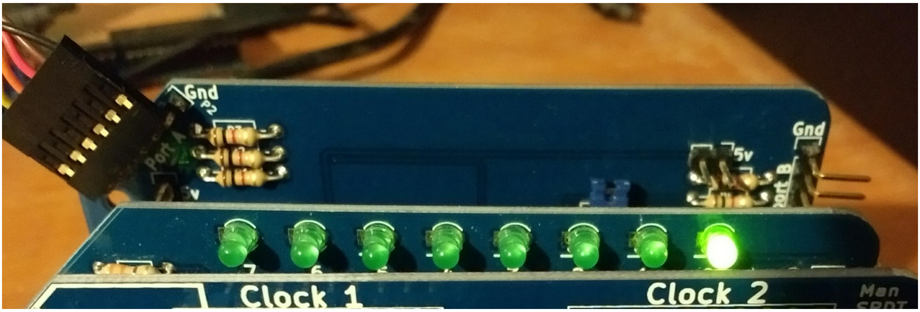
```
    G 8000
```



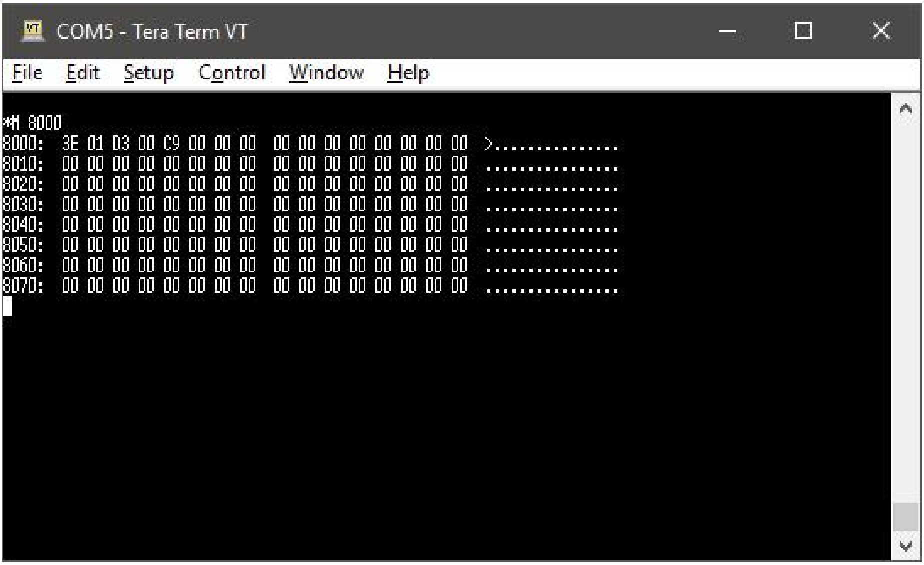
```
Small Computer Monitor
*Ready

*g 8000
*
```

And the LED turns ON...

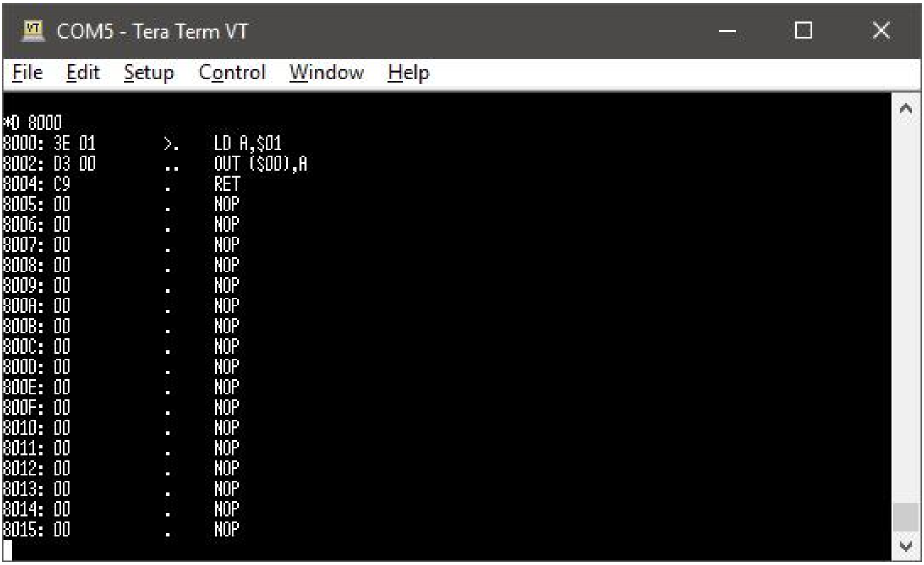


We can see the program in the RC2014's memory with the command "M 8000".



```
*M 8000
8000:  3E 01 D3 00 C9 00 00 00  00 00 00 00 00 00 00 00  >...............
8010:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
8020:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
8030:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
8040:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
8050:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
8060:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
8070:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

We can also disassemble the program with the command "D 8000"



```
*D 8000
8000: 3E 01       >.    LD A,$01
8002: D3 00       ..    OUT ($00),A
8004: C9          .     RET
8005: 00          .     NOP
8006: 00          .     NOP
8007: 00          .     NOP
8008: 00          .     NOP
8009: 00          .     NOP
800A: 00          .     NOP
800B: 00          .     NOP
800C: 00          .     NOP
800D: 00          .     NOP
800E: 00          .     NOP
800F: 00          .     NOP
8010: 00          .     NOP
8011: 00          .     NOP
8012: 00          .     NOP
8013: 00          .     NOP
8014: 00          .     NOP
8015: 00          .     NOP
```

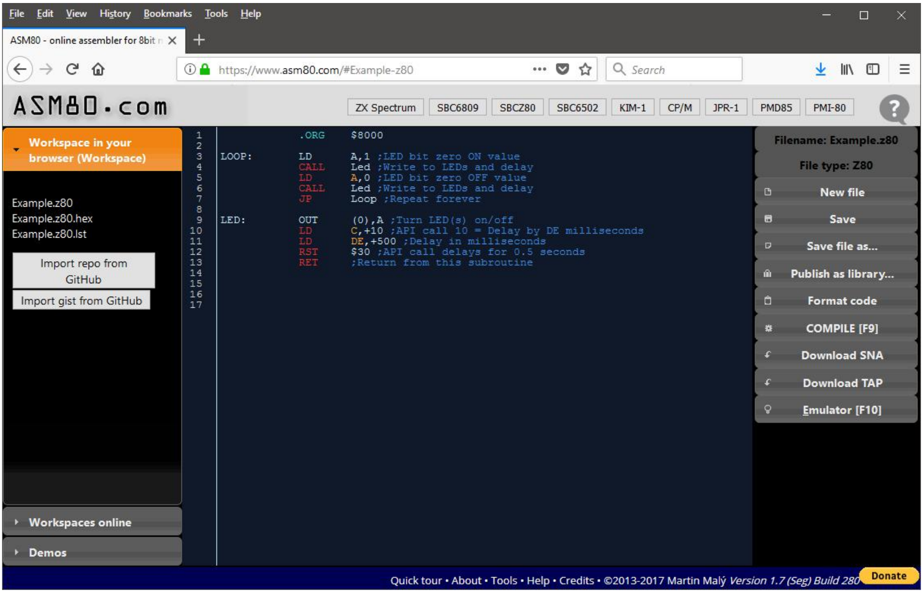
Here's a slightly more sophisticated example, taken from the Small Computer Monitor Tutorial, which actually flashes an LED!

```
            .ORG    $8000

LOOP:       LD      A,1         ;LED bit zero ON value
            CALL    Led         ;Write to LEDs and delay
            LD      A,0         ;LED bit zero OFF value
            CALL    Led         ;Write to LEDs and delay
            JP      Loop        ;Repeat forever

LED:        OUT     (0),A       ;Turn LED(s) on/off
            LD      C,+10       ;API call 10 = Delay by DE milliseconds
            LD      DE,+500     ;Delay in milliseconds
            RST     $30         ;API call delays for 0.5 seconds
            RET                 ;Return from this subroutine
```

This is the display when the "Format code" button is select.



```
            .ORG    $8000

LOOP:       LD      A,1 ;LED bit zero ON value
            CALL    Led ;Write to LEDs and delay
            LD      A,0 ;LED bit zero OFF value
            CALL    Led ;Write to LEDs and delay
            JP      Loop ;Repeat forever

LED:        OUT     (0),A ;Turn LED(s) on/off
            LD      C,+10 ;API call 10 = Delay by DE milliseconds
            LD      DE,+500 ;Delay in milliseconds
            RST     $30 ;API call delays for 0.5 seconds
            RET     ;Return from this subroutine
```

Use the same method to send the hex file to the RC2014 system.

Contact Information

If you wish to contact me regarding the Small Computer Monitor please use the contact page at www.scc.me.uk (or smallcomputercentral.wordpress.com).

Issues related to the RC2014 can be posted on the RC2014-Z80 google grooup.

Stephen C Cousins, Chelmsford, Essex, United Kingdom.