Architecture des systèmes informatiques

LSINF1252_11 - Marc Lobelle

Pic 18F97J60 development in C on UNIX howto

David Janssens, Sébastien Barré, Laurent Lesage, Marc Lobelle. v2.06, 18 March. 2010

How to compile and run C programs on PIC development boards under UNIX (tested on Linux and SOLARISx86).

1. Introduction

This howto shows how to produce executable code for a PIC 18F97J60 processor and how to load it on a development board such as the Pic-maxi-web board from Olimex trough the network using TFTP

2. Fetching development tools

To install these tools, you should be root on your computer. Besides, if you are on SOLARIS, since the names of many gnu tools are prefixed with "g" on this system, you should type gmake instead of make and replace the string "ar -S" by "gar -s" in the Makefiles. You will be instructed below when you have to do this replacement in Makefiles

Install GPUTILS, which contains the PIC assembler, gpasm (gputils-0.13.7.tar.gz). gpasm, the gnu Pic assembler is compatible with the microchip Pic assembler mpasm documented in mpasm.pdf

```
wget http://sourceforge.net/projects/gputils/files/gputils/0.13.7/gputils-0.13.7.tar.gz/download
tar zxvf gputils-0.13.7.tar.gz
cd gputils-0.13.7
./configure
make
make install
```

Install SDCC, which contains the PIC C compiler sdcc-src-20091215-5595.tar.bz2). Documentation is available as sdccman.pdf

```
wget http://sdcc.sourceforge.net/snapshots/sdcc-src/sdcc-src-20091215-5595.tar.bz2
bunzip2 sdcc-src-20091215-5595.tar.bz2
tar xvf sdcc-src-20091215-5595.tar
cd sdcc
./configure
<if on SOLARIS, replace "ar -S" by "gar -S" in Makefiles in device/lib and subfolders >
make                    build the compiler
make install
cd device/lib/pic16
make                    compile the libraries with the new compiler
make install            install libraries and include files in
                        /usr/local/lib/pic16  and
                        /usr/local/share/sdcc/include/pic16
```

After this you should check the contents of/usr/local/lib/pic16 and if many have a name ending in .a (and not .lib), you should give the names the linker is expecting to those that shall be used:

```
cd /usr/local/lib/pic16
ln -s libc18f.a libc18f.lib
ln -s libdev18f97j60.a libdev18f97j60.lib
ln -s libio18f97j60.a libio18f97j60.lib
ln -s libm18f.a libm18f.lib
ln -s libsdcc.a libsdcc.lib
```

3. Compiling your PIC C program on UNIX and executing it on the PIC development board

Compile the program using SDCC. Small example programs are provided.

- led.c just blinks periodically one of the leds. It shows how to program a simple peripheral interface and how to use naïvely a timer (active waiting): in real programs, this should be avoided because it wastes processor cycles.
- test.c (download and unpack the archive testPIC.tgz including the testPic directory) uses the lcd screen, push buttons, and also reads and writes in the 8K external ram called Ethernet buffer. To write or read in this memory, one must initialise pairs sfr registers holding the low and high parts of addresses in the Ethernet buffer where to write(EWRPTL, EWRPTH) or read (ERDPTL, ERDPTH). Each pair constitutes an auto increment indirect adressing register. Then, to write or read series of data in the buffer, one must put or get the data in or from the EDATA sfr. Because we use autoincrement indirect addressing, write-after-read instructions such as "increment", "test-and-set", etc. should be avoided in this buffer, because the two pointers will be incrmented.
  For test.c, a few specific header files and library members are needed.

```
sdcc -mpic16 -p18f97j60 led.c
sdcc -mpic16 -p18f97j60 -L/usr/local/lib/pic16 test.c
```

or, if you downloaded the testPIC directory, "make led" or "make test".

This produces the .hex file that you can upload to the PIC, as explained in the next section. Other files are also produced. Have a look at the .lst file, that shows the program as translated in assembly language by sdcc.
The program memory, on the PIC boards is flash memory. Thus the bootloaders download programs and flash them in program memory.

4. Transferring and flashing the .hex program to the PIC

The TFTP bootloader, preinstalled in the top of the pic-maxi-web flash memory, is transferring data over the network. When running (i.e. for a few seconds each time you reboot the pic-maxi-web), it sets the IP address to 192.168.97.60, which means that tour computer should set its address to 192.160.97.. If you connect both to the yellow RJ45 sockets of the nat router provided with the pic-maxi-web board, you will get such an address automatically.

4.3 Loading programs through the network with the TFTP bootloader (pic-maxi-web)

A router is provided in projects based on the pic-maxi-web board. It is needed to set up a small network to which you will connect the PIC board and your computer on the LAN side (4 yellow Ethernet sockets).

When the board is powered up, the TFTP boot-loader waits for 4 seconds for a tftp client to send the right ".hex" file. After that, the boot-loader will run the application on the board, if any.

When running the TFTP bootloader,the board has the fixed IP address 192.168.97.60/24. It does not have any other IP tool (e.g. no ICMP, so it will not respond to ping). A DHCP server runs in the router. It is set up to give adresses in the range 192.168.97.100-200 to DHCP clients connected to the LAN side. This way, if you conect a computer to the LAN side, it will get an address in the right range to communicate with the pic-maxi-web board.

To communicate with the TFTP bootloader, you will have to use a TFTP client on your computer. On UNIX (LINUX, SOLARIS), there is a TFTP client called `tftp`. Change to the directory where the "prog.hex" is located. On the command line, type :

```
my_computer $ tftp 192.168.97.60
my_computer $ tftp> binary       // the program must be sent as bytes
                                 // (see "man tftp"!)
my_computer $ tftp> trace        // to see what happens
my_computer $ tftp> verbose      // to see more...
my_computer $ tftp> put progr.hex// before pressing "enter",
                                 // power up or reset the card (tiny red
                                 // button behind the ethernet plug the card),
                                 // count one, two, three and, at three,
                                 // press "enter"
...
...                              //trace of what happens...
my_computer $ tftp> quit         // you're done!
```

Just after that, your program will start running. Normally, there's no need to "reboot" the card, unless missed the 4 seconds to send your program, then push the reset button (the tiny red button near the ethernet socket) and try again the whole download procedure.

A few remarks

- The tftp server on the board will check the flashing process on the fly. If errors are detected, errors messages will be displayed on the tftp console on your computer. In that case, you may have to redo the downloading and flashing operation.
- Flashing the card with tftp is very fast (10 to 20 seconds for the whole 128Ko). Do not abuse of it : the number of flashing cycle is limited - do not shorten unnecessarily the life time of the card.
- The bootloader protects itself against rewriting. It is located in the highest part of the program memory (from 0x1DC00 to 0x1FC00). If you try to put a program in this area, you will get an error during the flashing operation.
