A Beginners Guide to Programming the ATMEL ATtiny 45 and 85

(used with the SoftRock v9.0 RX or USB xtall interface)

by Bob G8VOI (April 2009)

Contents **Version 1.2**

1: Introduction

First of all, I am no expert in the programming these devices. At the time of writing this, I am the veteran of 8 weeks experience in this subject, so I am more than happy to be corrected for any errors I have made in writing this document. By programming, I do not mean the creation of the actual code, just the process of writing an existing HEX code file to the device.

This is meant for as a guide for those with no previous experience of the subject.

I wished to try out Fred PE0FKO's version of the SoftRock v9.0 firmware for myself, having evaluated the QRP2000 v2.0 firmware and found some limitations with that. I set about with no previous knowledge, acquiring as much information as I could obtain, and constructing my own programming hardware.

Since then I have helped a number of others follow the same path as I did, to successfully program their own devices. It was suggested to me that I make my findings and experience available as a guide to others. This is my attempt at that.

I must express my heartfelt thanks and gratitude to Fred PE0FKO, not only for his great work in enhancing the original firmware, but for providing advice and encouragement to me, thanks!

2: First Steps

One of the first difficulties you might encounter is some of the terminology associated with the programming of devices if you are not familiar with that environment when looking at the various websites associated with the subject.

Often it can be extremely difficult to find any references to the terms used as it is taken for granted that the reader is familiar with them.

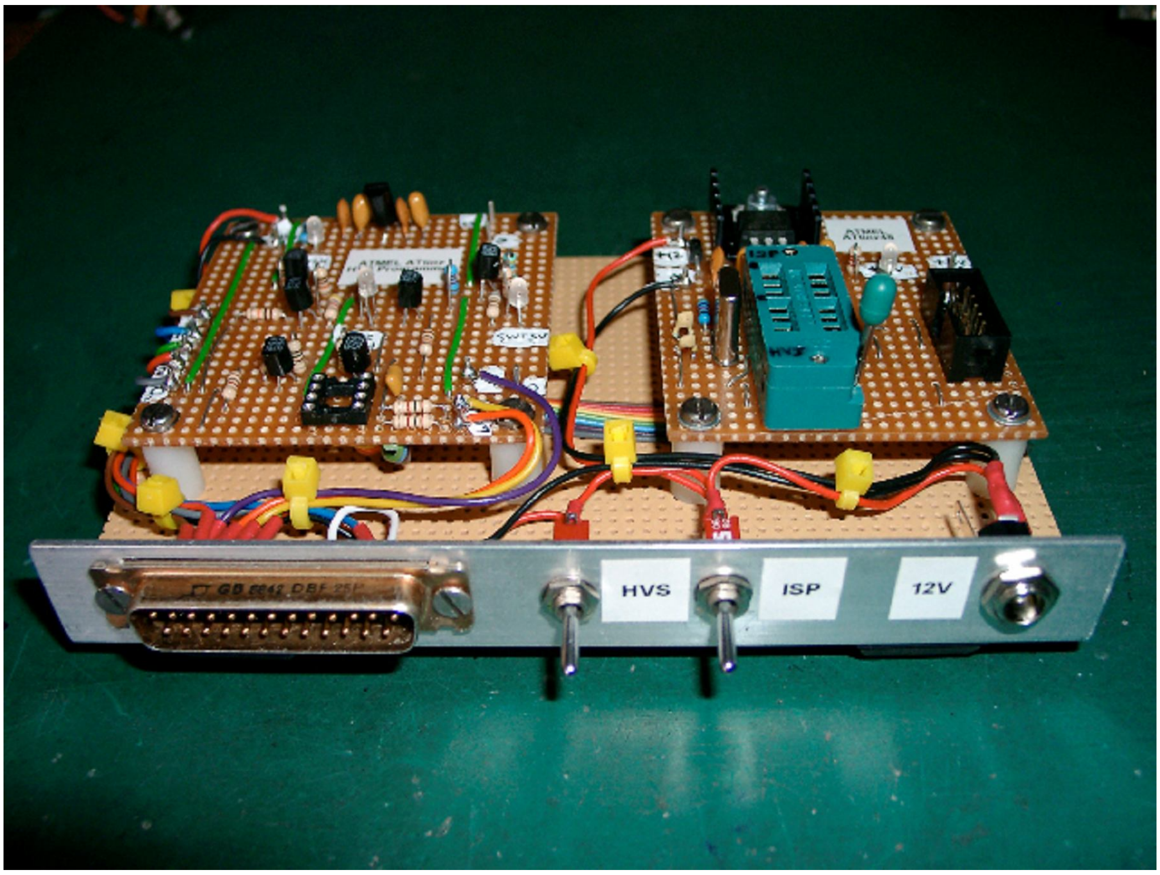
Hopefully I will try and avoid using any of these terms without providing an explanation when they are first used.

So here we go!

I found the following websites a very useful source for a mass of information to start with:

http://metku.net/index.html?path=articles/microcontroller-part-1/index_eng

http://elm-chan.org/works/avrx/report_e.html



This is the current set up I am using to program the Atmel devices. It is the combination of the two different types of programmers, built on strip board using junk box component.

On the right is my ‘Target’ circuit for ISP – In circuit Serial Programming

Originally I used a standard 8 pin IC socket for plugging in the device to be programmed, but decided that a zif type socket was a better method. The upper half of the 16 pin socket is used for the ISP circuit.

The ISP programmer is connected via the 10 way IDC socket to the right side of the zif socket.

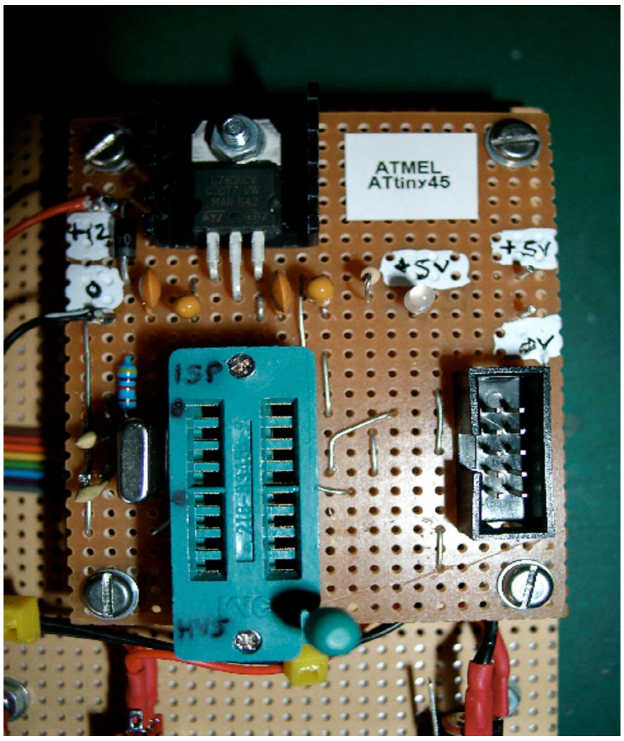
You can see a crystal fitted to the left of the zif socket, but this is not actually required during programming, something I only learnt after building the board.

The heatsink on the +5V 7805 voltage regulator is not needed, the current drawn is very small and a 100mA small 78L05 device is adequate.

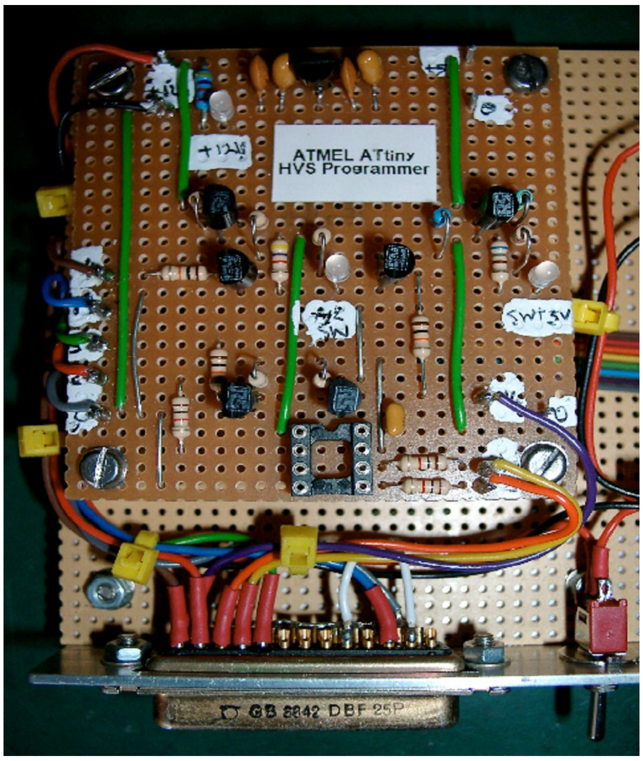
The board on the left side is my version of Loftur TF3LJ’s modified design for a HVS (High Voltage Serial) parallel port programmer.

Not much to say about it, except that the 8 pin IC socket is no longer used and a ribbon cable links it across to the lower half of the zif socket.

I was concerned about the lead length connecting the parallel port to the PC, but I have recently modified my set up to use a 1.5m (4 foot) extension lead and a 2 way parallel printer switch box. I have not had any problems at all with this, and it saves having to keep pulling the PC out to get to the rear connectors.



View of my ‘Target’ circuit used for ISP programming.



View of my version of the TF3LJ HVS programmer

3: ATMEL ATtiny 45 / 85 device pin outs

Both of these chips are pin compatible, the only difference is the program memory size

ATtiny45 – 4k byte
ATtiny85 – 8k byte

ATtiny 45 / 85 pin connections				
Pin	Function	I/O	SoftRock v9.0 Function	USB Xtall Interface
1	/RESET (controlled by RSTDISBL fuse bit)	PB5	J3 - 2 BPF control line	CW Key_1
2	XTAL1 - Crystal Oscillator input	PB3	Si570 - SCL control	Si570 - SCL control
3	XTAL2 - Crystal Oscillator output	PB4	J3 - 1 BPF control line	PTT_Out
4	0V Ground Vdd		0V Vdd Ground	0V Vdd Ground
5	MOSI - Master data Output / Slave data Input	PB0	USB-3 Input line	USB-3 Input line
6	MISO - Master data Input / Slave data Output	PB1	Si570 - SDA control	Si570 - SDA control / CW Key_2
7	SCK - Serial Clock Input	PB2	USB-2 Input line	USB-2 Input line
8	+5V Supply Vcc		+5V Vcc Supply	+5V Vcc Supply

You can download the datasheet for both devices from this site:

http://www.atmel.com/dyn/resources/prod_documents/doc2586.pdf

4: Programming modes

There are two different methods of programming the Atmel chips:

ISP – **I**n circuit **S**erial **P**rogramming

HVS – **H**igh **V**oltage **S**erial programming

Of the two types of programmer that can be used, the ISP type programmer is the simplest to construct, however this will restrict you to only programming new devices or those that have had the fuse bits restored following programming with the SoftRock firmware (an explanation is provided later).

Although a little more complicated to construct, my preferred choice would have to be the HVS type programmer. This will allow you to program new devices, reuse existing devices programmed with the SoftRock firmware or reset the fuse bits to allow programming again using the ISP method.

Your decision might have to be based on what PC hardware you have available, i.e. a parallel printer port or USB port, and also the availability of ATtiny devices where you are.

New motherboards are still currently available with legacy serial and parallel ports, or cheap PCI cards can be obtained to provide these.

If you only have the USB port option available, you might consider getting a cheap, older type PC with a printer port at a boot sale. The specification is not important, as long as it will run Win98se / Win2000 or XP. Any old Pentium 2 or 3 would be more than capable of doing the job.

I have tried a cheap USB to parallel printer port adapter, but this failed miserably. I believe some might be available that support bit manipulation, but do not know about any possible timing issues with this method.

5: ISP - In circuit Serial Programming

As the name implies, it is possible to program the device whilst it is physically mounted in the operating circuit. This method does however require direct access to the device pins, and steps must be taken in the circuit design to avoid the externally connected circuitry affecting the programming process. No facilities exist on the SoftRock v9.0 PCB to carry out his process.
You have to construct a simple ‘target’ circuit to allow you to program the chip. The target circuit simply provides a means of applying power (+5V) to the device, an 8 pin socket to mount the device in during the programming process, a pull-up resistor (4k7) on the /RESET line (pin 1), and a means of connecting the ISP programmer to the circuit.

There are two standard formats used for connecting the ISP programmer, either via a 6 IDC header or the more commonly used 10 way IDC header

Standard 10 Way ISP Connector

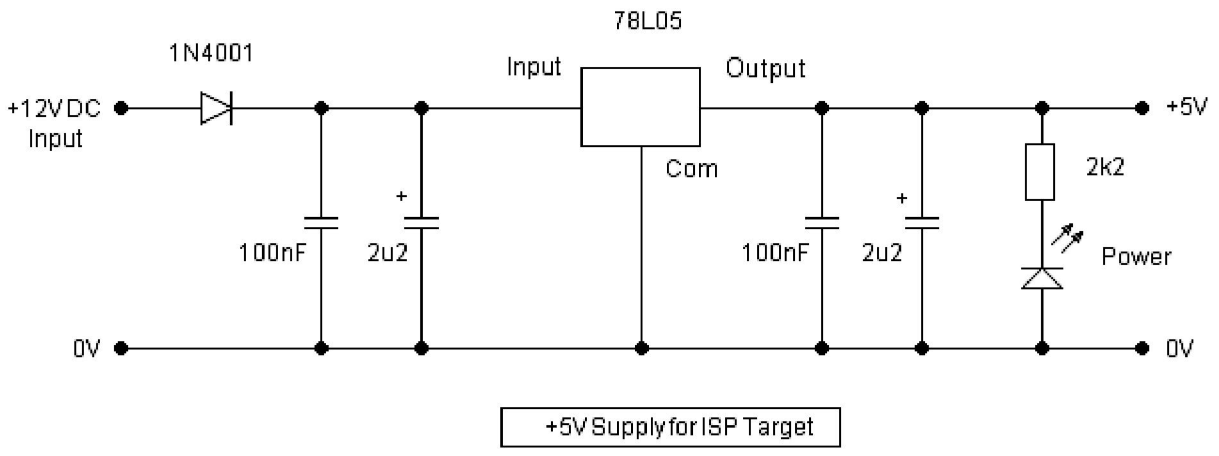
MOSI	Pin 1	Pin 2	+5V Vcc
LED	Pin 3	Pin 4	0V ground
RST	Pin 5	Pin 6	0V
SCK	Pin 7	Pin 8	0V
MISO	Pin 9	Pin10	0V

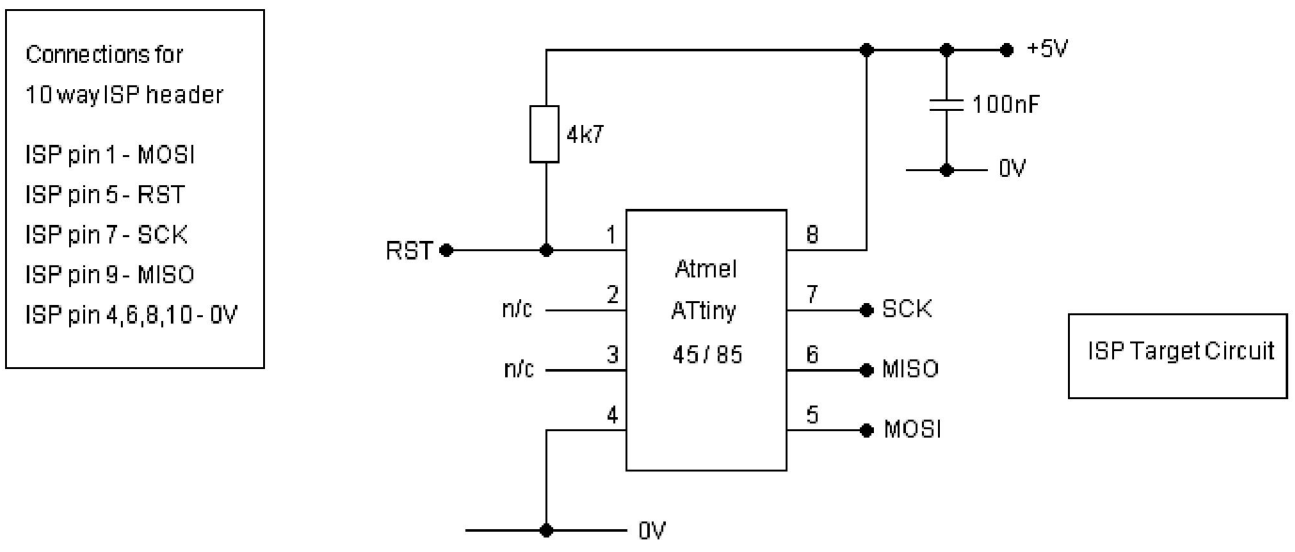
Standard 6 Way ISP Connector

MISO	Pin 1	Pin 2	+5V
SCK	Pin 3	Pin 4	MOSI
RST	Pin 5	Pin 6	0V

Depending on what type of programmer you use for ISP programming, you might wish to wire a compatible 10 way IDC socket on your board, or just hardwire it to the target circuit, as in the case of the simple parallel port described later.

Basic ISP (In circuit Programming) Target circuit





Connections for 10 way ISP header

ISP pin 1 - MOSI
ISP pin 5 - RST
ISP pin 7 - SCK
ISP pin 9 - MISO
ISP pin 4,6,8,10 - 0V

There is one limitation to using ISP programming. The device must either be new and unused or the fuse bits need to be reset to their original state if it has previously been programmed with a version of the SoftRock firmware.

The reason for the above is due to the limited number of pins available for use as inputs or outputs, the function of Pin 1 (/Reset) is changed in the final stage of programming to make it into an I/O (input / output pin). This is achieved by means of setting the RSTDISBL fuse bit to logic ‘0’. A fuller explanation of the fuse bytes will be given later.

Once the function of pin 1 has been changed, the /Reset line is no longer available, and that is essential for the ISP programming process.

Some ISP programmers can provide a +5V supply on the IDC header that can be used to provide power the target circuit, for example the Adafruit USBtinyISP has a link that can be fitted.

The Atmel devices are very cheap, so in the beginning you could treat them as 'one time programmable', and at a later stage use either a HVS programmer or a simple 'Fuse Restorer' to reset the devices.

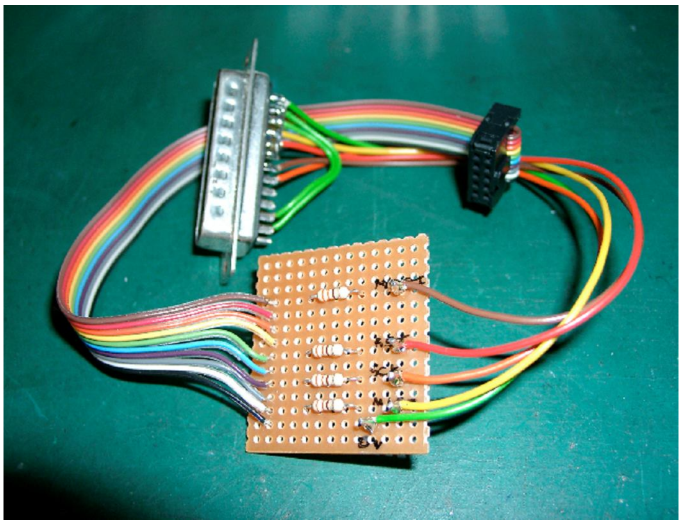
http://home.ict.nl/~fredkrom/pe0fko/Fuse-restore-ATtiny45/

It is worth noting that the 'Fuse Restorer' does require a programmed AVR chip, so it would be necessary to acquire this, or you will still need some programming hardware. An ISP programmer could be used for this purpose with a suitable target circuit.

The type of ISP programmer you choose to use will depend on what your PC hardware will support.

If you have a parallel printer port available, there are a number of very simple designs available to construct. The easiest just uses 4 resistors, often this type is also referred to as a 'stk-200' type programmer. An example shown here:

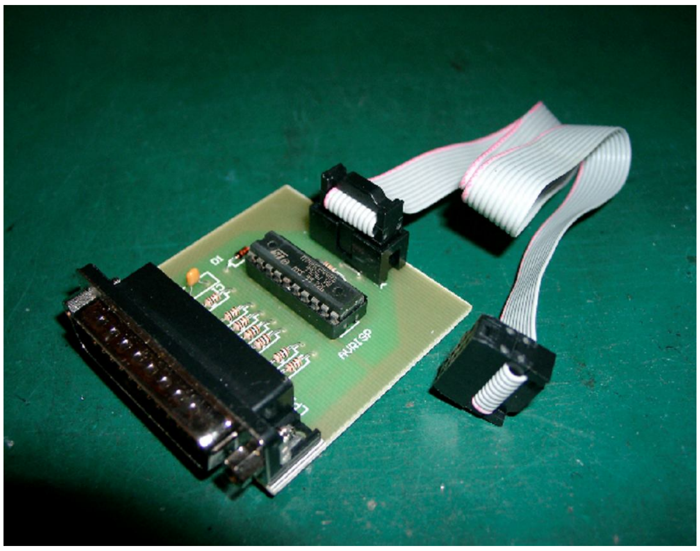
http://metku.net/index.html?path=articles/microcontroller-part-1/index_eng2



The simple ISP programmer I constructed to use with PonyProg2000

The ribbon cable and 10 way header are for connecting to my existing 'target' circuit, but you easily could combine both the target and programmer on a single board.

There are often cheap, simple parallel port programmers ready constructed available on ebay, similar to the one just mentioned, but usually with a buffer IC fitted to provide protection to the PC printer port.

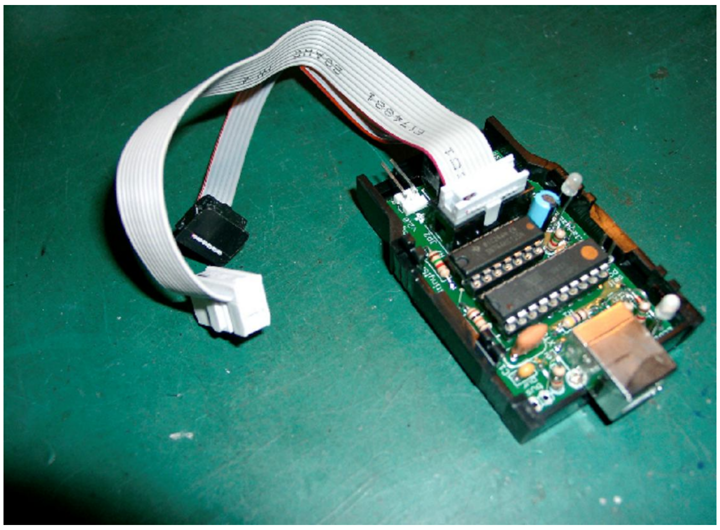


This is an example of a cheap parallel port programmer obtained from a Bulgarian company on ebay that will work with the PonyProg or Avrdude software.

If you do not have a parallel port, then there are cheap serial port or USB ISP programmers available.

One example of the USB type is the Adafruit USBTinyISP. This was the first device I obtained as a kit to build myself.

http://www.adafruit.com/index.php?currency=USD&main_page=index



Adafruit USBTinyISP programmer I built from their kit with the top cover removed.

I have used both the parallel port and USB ISP methods of programming on PC's using Win98se and Win2000, and have helped others run them with WinXP. I do not have any experience of Vista, so you are on your own there!

Much of the available free programming software packages is DOS based and does assume a reasonable understanding of using DOS. Later on I will describe using the common programs and explain some of the useful commands, plus how to create DOS batch files to simplify the process.

6: PonyProg2000

There is the option available to use the simple stk-200 type parallel interface with a Windows program by using the PonyProg2000 software, which can be downloaded from here:

http://www.lancos.com/prog.html

Initially it appeared that although you could program the Atmel ATtiny devices, the facility to program one of the fuse bits, critically the RSTSDISBL fuse, was greyed out and disabled, thus preventing it being used for the SoftRock firmware.

This is done in the program to prevent anyone accidentally changing that bit and preventing any further ISP programming.

Fred, PE0FKO spotted a 'script file' function in the help screens and suggested it might be of use. On investigation it was found it is possible to program all of the fuse bits from within the script file. I have produced a script file that will carry out the whole process of programming the chip using PonyProg, and that it is available to download from:

http://home.ict.nl/~fredkrom/pe0fko/SR-V9-Si570/

Please open the script file and read the comments contained within it.

They provide more information about using it.

It is just a basic text document and can be opened with Notepad or any text editor.

The script file is called 'peofko.e2s' and should be placed in the PonyProg2000 folder.

I would offer one suggestion, that whatever method of ISP programming you use that you follow a simple procedure to do it in steps, rather than programming the device all in one go.

First, program the hex code part, then the fuse bytes, but leave the RSTDISBL fuse set at its default state '1' to start with. This allows the device to be run in the v9.0 RX and verify that it is basically working apart from the one filter control line (pin 1) of the chip.

Then as a final step, program the fuse bits again, this time with the RSTDISBL bit set to '0' and then programming is complete and it should now fully control both BPF selection lines.

The above will be explained fully later on in the more detailed programming examples.

7: AVR-OSP II Program

There is another Windows program that supports some types of serial port ISP programmers that might be of interest. This can be obtained from:

http://www.esnips.com/web/AtmelAVR

8: WinAVR Software package

There is a suite of DOS based software for developing AVR code which includes the AVRDUDE program. This can be configured and used with either the simple parallel type ISP interface or the USBTinyISP.

http://sourceforge.net/project/showfiles.php?group_id=68108

Please note that this is a later version than I am currently running. This was only released on March 14th 2009 and I have not installed or run it yet.

Details on how to use the AVRDUDE program are included later in this document.

9: HVS - High Voltage Serial programming

The hardware required for a HVS programmer is a little more complicated than for ISP programming.

Essentially once the RSTDISBL fuse bit has been set to a ‘0’ as needed for the SoftRock firmware, this method of programming is required to reprogram the device, or clear the fuse bits. Once you have a working HVS programmer, the ISP one becomes redundant.

The programming involves switching a ‘High Voltage’ (12V) onto pin 1 of the chip during the process, hence HVS programming.

A simple design is available, modified by Loftur TF3LJ to use commonly available transistors works very well. It does however require a parallel port (a very good reason to hang onto an older PC).

http://sites.google.com/site/lofturj/tf3lj/AVR_HVS_Programmer.jpg

You need to pay careful attention to the 25 way male 'd' type connector pin layout and numbering. On the circuit diagram the layout is shown as a front view.

To make use of this programmer, you need the AVRX Programmer software, again a DOS based suite of programs. The file is called ‘avrxtool32.zip’

This can be obtained from here, from the link under the ‘Technical Data’ at the bottom of the page, described as ‘Win32-based control programs for Windows 9x/Me/NT/2k/XP’

http://elm-chan.org/works/avrx/report_e.html

Download this and unzip to a suitable folder.

Within that suite, it is the program AVRPP that is required to carry out all of the required operations.

Detailed notes on its use later on.

After constructing the HVS programmer, you should carry out a few basic tests on the hardware <u>before</u> connecting it to a PC and trying to use it.

I suggest the following:

Apply the 12V supply and check the output from the 7805 5V regulator is correct.

Connect a link from the +5V supply to pin 14 of the 25 way ‘d’ connector, and check pin 8 of the IC socket is approximately +5V (actually +5V – drop across transistor junction), also LED D3 should be ON.

Connect a link from the +5V supply to pin 1 of the 25 way ‘d’ connector, and check pin 1 of the IC socket is approximately +12V (actually the DC supply input – the drop across transistor junction), also LED D2 should be ON.

Although not exhaustive, hopefully these simple tests might prevent any damage to either the AVR chip or PC through incorrect construction of the HVS programmer circuit.

WARNING – DO NOT PUT ANY DEVICE IN THE IC SOCKET BEFORE RUNNING THE PROGRAM

The printer port default conditions turn both the +5V and +12V switched lines ON

I do not have experience of any USB controlled HVS programmers.

There is a design available called AVRDOPER which appears to provide the required facilities for HVS and ISP programming. Circuits and PCB layouts are available, but it appears to require a programmed Atmel controller itself, so either you would need to obtain that ready programmed, or construct an ICP programmer to do it yourself.

Details here:

http://www.obdev.at/products/avrusb/avrdoper.html

If money is no object (lucky you), then there is the official Atmel AVR Dragon development board, which provides both ISP and HVS facilities. There is a prototyping area provided on the board for a zif type socket to be fitted. Jumper links are used to connect to various headers to configure the socket for the desired device.

http://www.atmel.com/dyn/Products/tools_card.asp?tool_id=3891

10: Program HEX Code

In order to program your own devices you need to obtain the HEX code file for the firmware version and chip type you wish to use.

Note: there is a minor difference between versions for the ATtiny45 and ATtiny85 devices. Download the one you require. This is to do with the position of the 'stack pointer' at the 'top' of the memory of the two chips due to their differing capacity (4k and 8k respectively).

You can get download either version of Fred PE0FKO's firmware HEX files from here:

http://home.ict.nl/~fredkrom/pe0fko/SR-V9-Si570/ (latest version is v15.10 - 27th March 2009)

In Explorer or Firefox, use 'File' and 'Save Page as', this will create a file with the HEX extension in your chosen destination folder.

For simplicity, in all of the programming examples given later, using the different programming packages, this HEX file has been renamed, and is always called 'out.hex'. This saves any confusion with differing versions of the firmware. There is no reason however why it cannot be left as its original name, you just need to take this into account and modify the commands as necessary.

11: Fuse Byte Information

As well as programming the HEX 'code' part of the chip, it is necessary to program the fuse bytes.

These configure some basic parameters within the chip. If you do not do this, the firmware will not run.

It is not essential to understand the function of these bytes, only to know that there are essentially just two conditions that we are interested in for using with the SoftRock firmware. These concern the state of the RSTDISBL fuse bit:

'1' = safe state, not programmed, ISP programming possible
'0' = Pin 1 /Reset function changed to be used as an input or output line (PB5), ISP no longer possible.

Fuse bit information

```
Fuse high byte:
0xdd = 1 1 0 1  1 1 0 1    RSTDISBL disabled (ISP programming can be done)
0x5d = 0 1 0 1  1 1 0 1    RSTDISBL enabled (PB5 can be used as I/O pin)
       ^ ^ ^ ^  ^ \-+-/
       | | | |  |   +------ BODLEVEL 2..0 (brownout trigger level -> 2.7V)
       | | | |  +---------- EESAVE (preserve EEPROM on Chip Erase -> not preserved)
       | | | +-------------- WDTON (watchdog timer always on -> disable)
       | | +---------------- SPIEN (enable serial programming -> enabled)
       | +------------------ DWEN (debug wire enable)
       +-------------------- RSTDISBL (disable external reset -> disabled)

Fuse low byte:
0xe1 = 1 1 1 0  0 0 0 1
       ^ ^ \+/  \--+--/
       | |  |      +------- CKSEL 3..0 (clock selection -> HF PLL)
       | |  +-------------- SUT 1..0 (BOD enabled, fast rising power)
       | +----------------- CKOUT (clock output on CKOUT pin -> disabled)
       +------------------- CKDIV8 (divide clock by 8 -> don't divide)
```

There are three fuse bytes, but the third 'Fuse – extended byte' is left in its default state and not used.

Fuse byte – low:

This is always set to 1110 0001 (e1 hex, 0xe1)

Fuse byte – high:

The 'safe state' is set to 1101 1101 (dd hex, 0xdd) ISP programming possible

Final state for the SoftRock must be changed to 0101 1101 (5d hex, 0x5d)

12: Using PonyProg2000 and a simple parallel port programmer

This is perhaps the simplest possible method that you can use to program your own device, but it does require a PC with a parallel port, and a new or clean device. It is not possible to use this to reprogram a device previously programmed with a version of the SoftRock firmware.

In addition to obtaining and installing the PonyProg2000 software:

http://www.lancos.com/prog.html

You will also need to have constructed a simple 'target' circuit as previously described, and have parallel port ISP programmer.

Download the version of the firmware hex code you need and also my PonyProg script file from Fred PE0FKO firmware page:

http://home.ict.nl/~fredkrom/pe0fko/SR-V9-Si570/

Rename the hex code file as 'out.hex' and put it and the script file 'pe0fko.e2s' in the same folder as PonyProg is located.

When you first run PonyProg you need to set up the hardware interface for the parallel port programmer.

From the 'Command' bar, select 'Set Up', then 'Interface Set Up'.

Here you need to select 'Parallel Port' and the number of the printer port you wish to use, usually LPT1, (unless you have more fitted).

In addition you need to select the method the operating system uses for controlling the printer port.

This will must be set to 'Avr ISP I/O' for Win2000 and WinXP.

These operating systems do not support the use of the Windows API, 'Avr ISP API'. That can only be used with Windows98se (actually does not matter which you use with Win98se).

The only other thing needed is to run 'Calibration' from the set up. This is just a one time operation to determine the timing needed for your PC.

Select the device type you are going to program from either the pull down menu or command bar. ‘AVR micro’ and either leave as ‘AVR auto’ or set to ‘ATtiny45 or 85’ as required.

Once you have completed the above steps, you are ready to connect you programmer and the target circuit. At this stage do not apply power or fit the device to be programmed.

You are now ready to run the script file and program the device.

From the command bar, select ‘Script’ and ‘Load and Run’, select the script file to run ‘pe0fko.e2s’, and open. This will load the contents of the ‘out.hex’ file and this will be displayed on the screen, and you will then receive a series of prompts taking you through the programming and advised testing process. The final stage is to put the device back in the programmer and program the RSTDISBL fuse bit.

There is no problem restarting the process again from the start, other than if you have completed the last step, then ISP programming is not possible and you will receive an error message.

At each stage of the process, you have the option to continue ‘Yes’ or exit using the ‘No’ or ‘Cancel’ options.

The complete process is as follows:

Prompt > "Connect and power up the target, are you ready?'

If 'Yes', then continues to erase the device, write the 'out.hex file' and
verify it

Prompt > "Ready to program fuse bytes in 'safe mode'. RSTDISBL not
programmed now. This allows testing first"

If 'Yes', programs the fuse bytes, but leaves RSTDISBL at '1'

Prompt > "Remove target power and test firmware works, except for PB5 output
line, or continue to program the RSTDISBL bit"

If 'Yes', gives another prompt to make sure you really want to do that.

Prompt > "Are you really sure? Check target is powered and continue to
program RSTDISBL bit"

If 'Yes', programs fuse bits with RSTDISBL set to '0'

Prompt > "Programming completed, you will not be able to read or program in
ISP mode again, use HVS programmer to clear"

At the end of this process you should now have a fully programmed device.

13: Running DOS programs using the parallel port with Win2000 and XP

There should be no problem running any of the DOS based programming software under Win98se as this allows full control of the parallel printer port.

To make use of the parallel printer port under Win2000 or XP it is necessary to install and register a driver file called 'giveio.sys'.

This is included within the 'WinAVR' zipped package along with a DOS batch file 'install_giveio.bat' and 'loaddrv.exe' program file. Run the batch file from DOS command prompt in the c:\winavr\bin folder.

These files can also be used with the 'AVRPP' program. Copy the three files into the c:\avr\bin folder and run 'install_giveio.bat' from the DOS command prompt in c:\avr\bin

It has been found that the 'avrpp' program will actually work if the 'giveio.sys' driver has not been installed, but the file is present in the same directory.

More details are provided in the sections regarding operating these programs.

An alternative source of the 'giveio.sys' driver along with installation information is available to download from the following website:

http://www.cs.ucr.edu/~eblock/pages/pictools/giveio.html

Thanks to Ashok, VU2ASH for providing that link, and confirming the HVS programmer functioned correctly using Win XP.

14: Using AVRDUDE and a simple parallel port programmer

Download a copy of the WinAVR software package, and install this on your PC.

http://sourceforge.net/project/showfiles.php?group_id=68108

Please note this is a newer version of the one I am currently using. Released on March 14th 2009.

The version I have is: WinAVR-20081205

I have assumed that there are no major differences.

The default installation is to C:\WinAVR

There are user manuals in the above directory.

The program you need to use is called ‘avrdude.exe’ and is in the C:\WinAVR\bin folder.

Note: to use this program with Win2000 or WinXP you will need to install the driver ‘giveio.sys’ to allow control of the parallel port.

There is a batch file called ‘install_giveio.bat’ in the \bin folder, which will do this for you. Just double click on the file and it will install and register the ‘giveio.sys’ to the required system folder.

To run the ‘avrdude.exe’ program it is necessary to run this from a DOS window, to do this,click on ‘Start’ in the bottom left hand corner of your screen, select ‘Run’ and type ‘Command’ and enter.

This will open a DOS screen and take you to the DOS prompt, probably something like:

```
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-1999.

C:\DOCUME~1\ADMINI~1>
```

From the DOS prompt, type CD\ to get to the base directory, then CD\winavr\bin to get to the required folder containing avrdude.exe

It is possible to program the device with a series of manual commands from the DOS prompt.

A simpler method is to use a batch file that you can obtain if you download the full zipped version of the firmware from Fred’s website. The batch file called ‘avrdude.bat’ can be run to program the device in a single operation rather than manually typing in the commands.

You need to put this file batch file and the ‘out.hex ‘ file in the C:\winavr\bin folder.

Note: before using the batch file you will need to edit and modify it for your programmer hardware and the type of device you are using, use Notepad, or any text editor.

You need to change the following:

Device type either,	–p t45 for the ATtiny45	or	-p t85	for the ATtiny85
Programmer ID, use either	-c pony-stk200 (both work with the simple parallel interface)	or	-c stk-200	

Remove the Port command ‘–p com4’ completely, the program default is LPT1, no entry needed

You will also see that the high fuse byte is set as   hfuse:w:0xDD:m

This is the ‘default’ safe state and does not program the RSTDISBL bit, this gives you the opportunity to program the device and test it first. If all is ok, go back and edit the batch file again and change this entry to:

hfuse:w:0x5D:m   This will set the RSTDISBL bit to ‘0’ and program the chip again

After this, you will no longer be able to use ISP to program the device, if you try, you will receive an error message.

I have found it useful to make a desktop short cut from the ‘avrdude.bat’ file and then you can run it by double clicking on the icon. This saves opening ‘Explorer’ to get to the file, or going through the ‘Command’ and DOS box route.

15: Using AVRDUDE with the Adafruit USBTinyISP Programmer

The procedure for using the Adafruit USBTinyISP programmer is very similar to the previous one described for the simple parallel port programmer.

It is assumed that you have already followed the instructions for installing the USB driver for the USB device and it is working correctly.

Once you are at the DOS prompt C:\winavr\bin> you can type the command line:

avrdude –c usbtiny –p m8	this should recognise the USB adapter is connected, if not you need to investigate if the USB driver is loaded correctly.

Once this stage is working correctly, connect the USBTinyISP to your target circuit, plug in the device and apply power. Type the following at the DOS prompt:

avrdude –c usbtiny –p t45 or avrdude –c usbtiny –p t85

This should show communication is possible between the programmer and the device in the target circuit.

You need to modify the 'avrdude.bat' file as before, except now the Programmer ID becomes:

-c usbtiny

Once that is done, follow the procedure for using the parallel port programmer with avrdude.

I have found it useful to make a desktop short cut from the 'avrdude.bat' file and then you can run it by just double clicking on the icon. This saves opening 'Explorer' to get to the file, or going through the 'Command' and DOS box route.

16: Using the HVS programmer and the AVRX programming software

To make use of this programmer, you need the AVRX Programming software, again it is a DOS based suite of programs. The file to download is: avrxtool32.zip

This can be obtained from the link below under the 'Technical Data' at the bottom of the page, described as 'Win32-based control programs for Windows 9x/Me/NT/2k/XP'

http://elm-chan.org/works/avrx/report_e.html

Download this and unzip to a suitable folder, I suggest C:\avr for ease of use.

Within that suite, is the program 'avrpp.exe' that is used to carry out all of the required operations using the HVS programmer.

This will run happily under Win98se, however for Win2000 and WinXP you need to obtain the driver file 'giveio.sys' and install this.

There are some instructions in the 'differ_e.,txt' file in the C:\avr folder.

I suggest you follow the instructions in the AVRDUDE section to do this. Download the WINAVR package and copy the 'giveio.sys', 'install_giveio.bat' and 'loaddrv.exe' files into the following folder:

C:\avr\bin

You then just need to run 'install_giveio.bat' file once by double clicking on it to copy the driver to the correct system folder.

There is a basic help file for the commands called 'avrx32_e.txt in the C:\avr folder, and some additional information in the 'tips_e.txt' file.

<u>WARNING – DO NOT PUT YOUR CHIP IN THE SOCKET BEFORE RUNNING THIS PROGRAM</u>

The state the printer port defaults to means both the switched +5V and +12V supplies will be applied to the device. Once you run the program, these lines are set to the correct state and you will be prompted to plug the chip in and apply power.

These programming examples assume that the HEX code file has been renamed 'out.hex' and is located in the C:\avr\bin folder.

It is possible to program the device in a series of steps using commands from the DOS prompt.

To do this, you need to use the following commands (it is possible to combine multiple command on a single line)

The –p1 specifies the parallel port LPT 1 is used. This is the default, or after you have used it once, it can be omitted from subsequent lines.

It is not necessary to specify the device type, this is automatically detected from the device being programmed.

If you just run the avrpp program without any additions to the command, you get a simple help screen and list of supported devices.

```
C:\avr\bin\avrpp
```

To confirm the programmer is functioning correctly and recognising the device, at the DOS prompt:

C:\avr\bin>avrpp –p1 –r	Reads the chip
C:\avr\bin>avrpp –p1 –rf	Reads the state of the fuse bytes
C:\avr\bin>avrpp –p1 –e	Erases the chip

To program the chip for use in the v9.0 SoftRock:

C:\avr\bin>avrpp –p1 out.hex	Programs the HEX code part with the file 'out.hex'
C:\avr\bin>avrpp –p1 –v	Reads and verifies the code part has programmed correctly
C:\avr\bin>avrpp –p1 –fl11100001	Programs the low fuse byte with 'E1' hex (0xe1)
C:\avr\bin>avrpp –p1 –fh11011101	Programs the high fuse byte with 'DD' hex (0xDD) Note this is the 'safe' state with RSTDISBL fuse NOT set (also the state to reset a chip to allow ISP programming)
C:\avr\bin\avrpp –p1 –fh01011101	Programs the high fuse byte with '5D' hex (0x5D) This is the state needed for the v9.0 RX (ISP no longer possible after this)

You can use the above method, but I have produce two supporting DOS batch files which are called 'blank.bat' to reset a device to allow ISP programming, and 'burn.bat' to automate the complete programming sequence.

When run, these require the user to respond to the prompts to proceed with each step.

Create a new text document called ‘burn.bat’ using Notepad. Copy and paste the following text into the 'burn.bat' file and save. Put this in the C:\avr\bin folder.

```
=====================================================

@echo off
@cd\
@cd avr
@cd bin
echo.
@echo *** Erase device before programming ***
c:\avr\bin\avrpp.exe -p1 -e
echo.
@echo *** Clear Fuses before erasing ***
c:\avr\bin\avrpp.exe -p1 -fl11100001 -fh11011101
echo.
@echo *** Program Hex Code ***
c:\avr\bin\avrpp.exe -p1 out.hex
echo.
@echo *** Program Fuse bits ***
c:\avr\bin\avrpp.exe -p1 -fl11100001 -fh01011101
echo.
@echo *** Read Fuse bits ***
c:\avr\bin\avrpp.exe -p1 -rf
echo.
@echo *** Programming Completed ***
@pause
@exit

=====================================================
```

The above assumes the following:

Everything is in a folder called C:\avr\bin

The hex code is contained in a file called ‘out.hex’

You are using LPT 1 for the programmer.

Make any changes to suit as necessary your configuration if any deviation from the above.

If you make a desktop short cut from the file ‘burn.bat’ this can be run by double clicking on the icon, which simplifies the process.

I have also created a batch file called ‘blank.bat’ which can be used to erase the chip and clear the fuse bytes to allow it to be programmed using an ISP programmer again.

Create a new blank text document called ‘blank.bat’ and paste the text from the next page into this document and save it.

Copy and paste the following text into the file ‘blank.bat’

===

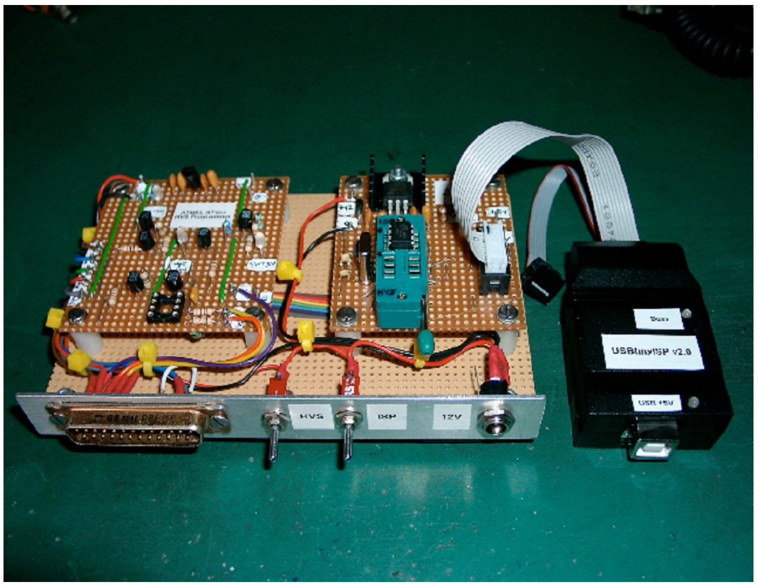
```
@echo off
@cd\
@cd avr
@cd bin
echo.
@echo *** Read device before erasing ***
c:\avr\bin\avrpp.exe -p1 -r
echo.
@echo *** Read  Fuses before erasing ***
c:\avr\bin\avrpp.exe -p1 -rf
echo.
@echo *** Erase device ***
c:\avr\bin\avrpp.exe -p1 -e
echo.
@echo *** Clear Fuse bits ***
c:\avr\bin\avrpp.exe -p1 -fl11100001 -fh11011101
echo.
@echo *** Read Fuse bits ***
c:\avr\bin\avrpp.exe -p1 -rf
echo.
@echo *** Device fully erased, ISP now possible ***
@pause
@exit
```

==

The above is a bit over kill, but demonstrates the process.

When the device is cleared, you will see the fuse high byte begins with a ‘1’, that is the RSTDISBL fuse bit in its default state.

Again, put the ‘blank.bat’ file in the C:\avr\bin folder and make a desk top short cut to it to simplify using it.



17: Disclaimer

Whilst I have taken reasonable steps to ensure the accuracy of the information supplied in this guide, I cannot accept any responsibility for any damage, loss or injury to equipment or persons as a result of using this information.

I cannot accept any responsibility for the contents and accuracy of websites where links are supplied.

You could have found these sites in the same way I did using the ‘Google’ search engine.

This guide is supplied in the spirit of amateur radio to perhaps answer a few common questions, nothing more, nothing less.

18: Copyright

I do not believe that I have used any material or information that is subject to copyright within this document.

I acknowledge the use of the names Atmel, AVR and ATtiny series of devices.

All photographs and drawings used are the property of Bob Reeves G8VOI.

I am more than happy for this guide to be freely used and distributed, providing no charge is made for it, and that I am given credit for it.

19: Errors or Suggestions (polite ones only please)!

If you find any errors or have suggestions of any additional items that are of interest, I would very much like to hear about them.

If you want further information or advice, I will try and provide assistance, please use the routes below.

My details are obtainable from the QRZ.com website, or make a posting on the Yahoo SoftRock forum.

Thank you to those who have provided feedback, your comments have been included in this latest version.

I hope you have found this of some use.

73, Bob Reeves G8VOI

3rd April 2009

20: Document Revisions

Any significant changes to this document will be listed here

Issue 1 - March 2009

A few copies were sent to individuals to check and provide comment and feedback, not generally released.

Issue 1.1 - March 2009

First version made public and available as a web page and in pdf format on Fred PE0FKO's website

Issue 1.2 - April 2009

Error in installation process for 'giveio.sys' driver corrected, plus minor changes reflecting comments received in feedback

Added index and contents at beginning, section and page numbering.