Introducing Full_Atmega_Programmer_V_2.2

V_2.0 of this programmer suffered from three drawbacks.

First: It is too easy to inadvertently erase the flash of a target device.

Keypress 'P' in response to the "Press -P- to send a program file or -E- to send a text file" user prompt resulted in the flash memory of the target device being erased immediately.
In version 2.1 the flash is not erased until the user has initiated a hex file download and the first symbol (a ':') a has been received.
If 'P' is inadvertently pressed it is now simply necessary to press 'X' to escape.

Second: Data sent from the EEPROM programmer to the PC tends to get disorganized

Once the EEPROM is programmed its contents are echoed back to the screen for confirmation. Unless delays are introduced the flow of data can be more than the terminal program can handle without scrambling the data.
The need for key presses has therefore been introduced to slow the data down.

Third: Program failure due to poor default calibration factors

The default calibration factor supplied with a small number of devices fails to enable communication with a PC using the UART.
An auto cal routine has been added that only runs following the application of DC power. This routine determines suitable cal factors and writes them to EEPROM (locations 0x3FF and ox3FE).
For all other resets, i.e. external, watch dog and brown out the cal factors are copied from EEPROM. A device with a particularly poor default calibration factor will generate spurious characters when it is first released from reset by a programmer. If this is the case a POR with force the auto cal routine to run.

The auto cal routine assumes the presence of a watch crystal across the TOSC1 and TOSC2 pins. (Note: capacitors have not been found necessary.) The reason for only running the auto cal routine after a POR is the time recommended for the watch crystal to stabilise.

Please note.
It is hoped that parts of this program might be of interest to anyone wanting to incorporate a programmer into their product. The point about having the C code is that it can be amended to fit a particular requirement. For example:

The target device can be changed by making comparatively minor changes. See the "Atmel_powerup_and_target_detect" macro for example.

Any required target configuration bytes can be inserted into the program.

The host device optionally runs under the following configuration bytes:
Extended: 0xFF High: 0xD7 Low: 0xE2
However the only critical parameters are:
- Watch dog under program control
- EEPROM preserved at chip erase
- 8MHZ clock

The programming interface can of course be changed.
At the moment it is arranged so that the MOSI, MISO and SCK pins of the target are connected to the identical pins of the programmer.