An AVR programmer development base on the UNO

Introducing the UNO programmer

An Atmega 328 device pre-loaded with the Arduino bootloader and the UNO hardware make a very effective combination for developing and running C applications. It is also possible to program the UNO using a range of external programmers or even with another UNO. The purpose of the development described here is to extent this flexibility allowing the UNO to be used as a general purpose AVR programmer.

An Arduino application "UNO_programmer" is introduced that is hosted on the UNO and enables it to be used to program Atmega devices as follows:

Flash with a hex file loaded at address zero.
Flash with a text file loaded from the maximum address.
The configuration bytes.
The EEPROM with text and numbers.

Two Arduino applications are provided for an Atmega 328 target device.

The first, "Cal_auto_plus_manual" is used to calibrate the target device.
The second, "text_reader" contains the subroutines needed to read text from the flash.
Sample text files are also provided.

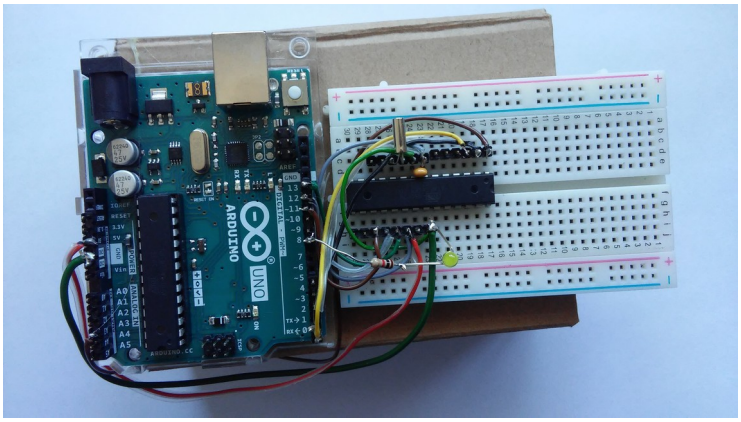
Once programming is complete, the target device is automatically put into the run state and the UNO reset button must be pressed to reinitialise the programmer.

UNO programmer Hard Ware.

A photograph of the HW is shown below. The target device is loaded into some plug-in prototyping breadboard and the following connections between the UNO and target device are made:

UNO output		Atmega 328 target pin and pin number	
5V		5V	7 & 20
0V		0V	8 and 22
Rx		USART TXD	3
Tx		USART RXD	2
Digital 11	MOSI	MOSI	17
Digital 12	MISO	MISO	18
Digital 13	SCK	SCK	19

In addition the following extra components are used;
A watch crystal between pins 9 and 10 (TOSC1 and TOSC2) of the target device
A 100nF capacitor between pins 7 and 8 of the target device
A led and 1K resistor between digital output 8 (PB0) of the UNO and 0V.



UNO Programmer HW:

The UNO and breadboard are mounted on a cardboard box.

Note especially the 100nF capacitor beside the crystal. This is particularly important for obtaining an accurate user OSCCAL value.

A user interface for the UNO programmer

A PC application normally used for sending hex files to the target processor hardware has not been written. Instead a terminal program is used and the entire hex file is sent to the bootloader. Arduino comes with a basic terminal program and there are several others on the web.

My favourite can be downloaded from https://sites.google.com/site/terminalbpp/. But take care to download the version 20130820, other versions may have an issue with the "scroll" button.

Settings for the terminal program are: Baud rate: 38400, 8 data bits, no parity, 1 stop bit and no handshaking.

Running the "UNO_programmer": Atmega 328 target devices only

Connect the UNO to the PC and open the Arduino application "UNO_programmer". It should upload OK. Check the comm port number if there are problems.

Open the terminal program.
User prompt "s s s s ….." should be generated.
Left click the mouse in the grey area at the bottom of its window and press 's'.
The programmer should respond with the message "Target_not_detected".

Connect the UNO to the bread board plus target device and reconnect it to the PC. This time the response should be "Press -p- to program flash, -e- for EEPROM or -x- to escape."

<u>Programming hex files</u>

Press 'p' to program a hex file only.
Click the "Send File" box and send a hex file when requested.
Press '0' at the "Integer(0-FF)?" prompt.
The file size and config bytes should be printed out.

Note: The config bytes have not been programmed.

<u>Programming text files</u>

If 'y' is pressed at the "Text_file? y or n" prompt a text file will be requested and immediately echoed back to the terminal program.
If 'n' is pressed at the "Text_file? y or n" prompt the program will run immediately.

Which ever is pressed the program will automatically be launched.
Note that the baud rate must temporarily be reduced while the text is being uploaded.

Target device calibration The application "Cal_auto_plus_manual" stores calibration bytes in the top two addresses of the target device eeprom. User applications can do without the crystal. Instead they read these bytes and use them to replace the default value of the OSCCAL register.
There is of course no need to restrict the target device to using its internal RC oscillator. However it is the default selection and is in many cases very convenient to do so.

Running the "Cal_auto_plus_manual" applications
This is supplied as an Arduino project however uploading it to the target device requires the "UNO programmer" rather than the Arduino bootloader. Its hex file is supplied but if changes are introduced a new hex file can be obtained by clicking on "Sketch/Export compiled Binary" and selecting "UNO_programmer.ino.standard"

To run the application
Connect the programmer to the PC and open the terminal program.
Download the hex file as described above.
There is no text file.
Auto cal will start running automatically.
The user will then be asked to confirm the device type.
Press '0' when requested assuming an Atmega 328 or 328P device.
Press 'x' to select the new calibration value or any other key to check it.

Reading text from the Flash

Upload the "text reader application" followed by the text file.
Note the temporary change in baud rate required by the text programmer.
The text strings are echoed to the terminal program in numbered format.
The user application calls two subroutines:

string_counter(): This counts the number of strings programmed to the flash.
Each is terminated by a null character and the final string is terminated by two.

print_string_num(string_num): This prints out any numbered string.

Both subroutines also need to know the size of the flash in words (3FFF for the Atmega 328).

Programming the EEPROM

At the "s s s……." prompt press 's', 'e' and 'w'.
Press '0' at the "Integer(0-FF)?" prompt and send the file.

Note that the file has to be sent several times however there is no need to change the baud rate.

Each string is echoed to the screen together with its address.
User applications need to know these addresses so that they can print the strings out.

Programming devices other than the Atmega 328/P

For all programs supplied here it is assumed that the target device is either an Atmega 328 or 328P and it is believed that the Arduino compiler only generates hex files for these devices.

For other target devices, all projects including “Cal_auto_plus_manual” and “text_reader” must be compiled using Atmel Studio or WinAVR.

Configuring the programmer for target devices other than the Atmega 328/P.

For the UNO programmer two files are used to specify the target device:

The header file of “UNO_programmer” which contains the following line:

```
const char *Device = "328";
```

Subroutine “set_up_target_parameters()” which can be found in:

“File Resources_UNO_programmer/Device_characteristics.c”

The main routine calls “set_up_target_parameters()” which reads the header file and then defines the flash and EEPROM sizes appropriately and all the other parameters required by the hex programmer.

As supplied “Device_characteristics.c” only has data for the Atmega328/P and 168/P devices. These are the ones with which this programmer has been developed. It is up to the user to ensure that this subroutine contains entries with all relevant data for their chosen device.

Target device data required by the programmer

In addition to memory size the “Device_characteristics.c” file also gives values for the following variables.

PageSZ PAmask FlashSZ EE_size target_type_M target_type target_type_P

These are all given in the data sheets apart from PAmask which is equal to FlashSZ – PageSZ.

Programming the Config bytes

Keypress ‘p’ programs the flash with a hex file. It does not program the configuration or lock bytes but does read them out. For an unused device this gives the default values.

Key press ‘P’ reads new configuration and lock bytes from the “Device_characteristics.c” file which must have already been entered by the user.

Setting up user programs

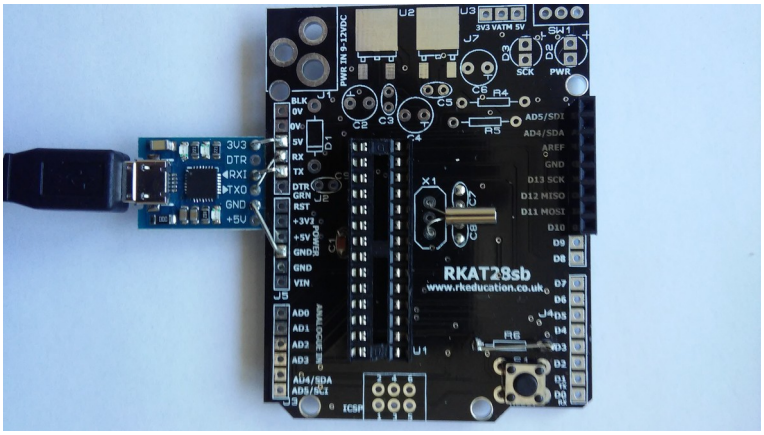
The user program header must contain the following line:

```
const char *Device = "168";
```

This assumes that an Atmega 168 or 168P is to be used as the target device.

A cut down version of the subroutine “set_up_target_parameters()” is also required that just gives the EPROM and flash sizes. See “text_reader_app” for an example.
This data enables the program to locate the calibration bytes and program the EEPROM and flash with text if required.

Test jig for programmed devices



Consists of an RKAT28sb Arduino shield
plus
CP2102 micro USB bridge.

Both available on ebay.

18pF trimmer capacitors are used with the watch crystal.

A device having been programmed on the UNO programmer HW was transferred to the test jig where it functioned satisfactorily. A crystal was also added to confirm that the user calibration byte obtained using breadboard was compatible with a double sided pcb.

NOTE: if reloading the UNO_programmer the target device Rx/Tx ports will usually have to be temporarily disconnected.

Devices with which the UNO_programmer has been tested
klkl
Atmega 328/P
Atmega 168/P