PCB_111000_V2: Getting a new pcb up and running

1. Introduction

PCB_111000_V2 is a development of PCB_111000.
Both pcbs are loaded with

An Atmega 168 which hosts user projects

An Atmega 328. This hosts a mini-OS which:

- Drives an 8 digit display
- Hosts a pcb bootloader that programs the Atmega 168
- Provides a variety of services for use by user projects

Differences are:

For the HW

- The FETS used to drive the display are no longer fitted

For the SW

- User projects have been revised and are now compiled using Arduino
- System programs are compiled using Studio 7
- PCB initialisation is carried out using the following Arduino sketches rather than bespoke HW
  - 8_UNO_AVR_programmer_V3
  - 5_Project_pcb_168_V2.30_Arduino_V2

2. Setting up PCB-111000_V2

Step 1 The PCB is loaded with all components except the displays and the Atmega devices

Step 2
Open the sketch “8_UNO_AVR_programmer_V3”. Under Tool/Board select Arduino UNO.
Connect the UNO to the PC check the port number and upload the sketch.

Step 3
Close the sketch assemble the HW (see section 3) and with the aid of the Br@y++ terminal emulator use the UNO to:

- Upload “5_Project_pcb_168_V2.30_Arduino_V2” to the Atmega 168
- Calibrate it’s internal RC oscillator
- Allow the Atmega 168 to run so that it can upload string data to its EEPROM.

Note:
Terminal emulator runs at 38.4KB and uses default settings.
See section 4 for more details.

Step 4 Load the Atmega 168 device onto the PCB111000 and test, then add the Atmega 328.

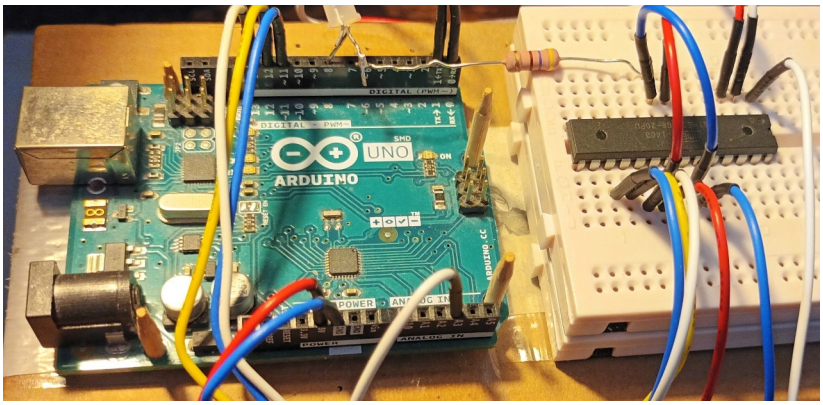
Step 5
Run “5_Project_pcb_168_V2.30_Arduino_V2” under the control of Br@y++ to setup the Atmega 328 (See section 5 for more details) by:

- Uploading the mini-OS to its flash
- Checking the calibration of its internal RC oscillator.
- Uploading the “Hello world” strings to its EEPROM

Step 6 Overwrite “5_Project_pcb_168_V2.30_Arduino_V2” with user projects. (See sections 6 and 7.)

Step 7 Upgrade the mini_OS if necessary (See section 8).

3. Hardware (HW) details



UNO used to program the Atmega168 before it is soldered onto the PCB111000_V2

Note: Atmega328 pin numbers given for a DIP28 device
Upload “8_UNO_AVR_programmer_V3” sketch before placing the Atmega168

UNO			Plug-in pcb
UNO Pin	Atmega 328		Atmega168
	Pin	Function	Pin
0 (Rx)	2	RXD	2
1 (Tx)	3	TXD	3
11	17	MOSI	17
12	18	MISO	18
13	19	SCK	19
A3	26	PC3	1 (Reset)
GND			8, 22
5V			7,20
8	14	PB0	LED driver

4. Driving “8_UNO_AVR_programmer_V3”

Having uploaded the sketch to a UNO open Br@y++ and set it up as follows:
8 data bits, 1 stop bit, no parity or handshaking and a baud rate of 38.4KB

Click on “scan” to identify the comm port and then click on connect:

The following user prompt should be generated:
“s s s s s s s……”

Step 1 Click mouse pointer in the text box (at the bottom of the Br@y++ window) and press -s- to get
“Atmega 168 detected.
Press -p- to program flash, -r- to run target, -d- to clear the target EEPROM or -x- to escape.”

Step 2 Press -p- and send “5_Project_pcb_168_V2.30_Arduino_V2.ino.eightanaloginputs”

Step 3 Follow instructions to Cal device then get user prompt “P/S P/S P/S……”

Step 4 Press -s-, change bit rate when asked and send Text_files\on-chip_strings.txt

A bug Sometimes random text following programming. Power cycle and reprogram.

Optional steps
At P/S prompt press -v- to get program name plus version "Project_pcb_168_V2.30_Arduino"

Reset UNO and change the BR to 38.4kB to restore user prompt "s s s s s s s……."
Press -s- then -d- to clear "on-chip_strings" and cal bytes from Atmega168 EEPROM

Reset UNO to restore user prompt "s s s s s s s……."
Press -s- then -t- to recalibrate the Atmega 328 and restore user prompt "P/S P/S…...…"

Reset UNO to restore user prompt "s s s s s s s……."
Press -s- then -r- to restore user prompt "P/S P/S P/S……."

Press -s- and reprogram the on-chip strings

Reset UNO to restore user prompt "s s s s s s s……."
Press -s- then -x- to escape

Reprogram the Atmega168 and note that it now offers a recalibration

5 Driving "5_Project_pcb_168_V2.30_Arduino_V2"

5.1 Add the Atmega 168
Solder the Atmega 168 to the PCB-111000_V2.
Press the DPDT switch left
Connect the pcb to a PC running Br@y++ running at 57.6KB.
Check the port and the user prompt "P/S P/S P/S……..." should be generated.
Press -p- and the response TTND (Target not detected) should also be generated.
This confirms that the USB port is working correctly.

5.2 Add the Atmega 328
Solder on the Atmega 328 and press -p- again to generate the prompt:
"ATMEGA 328 detected
Press -P- to send a program file -E- to send a text file (or X to escape).
p/e p/e p/e p/e p/e p/e….."

5.3 Upload the mini-OS to the Atmega 328
Press -p- and send
Mini-OS_V2\Hex_files\CC_files\Mini-OS_V26_CC.hex
When the LED stops flashing send 2_pcb_Bootloader_V4_28_CC.hex

Note: This assumes that Common Cathode displays are to be used. Similar files are present for Common Anode displays.

5.4 Auto cal the Atmega 328
Follow instructions to auto cal the Atmega238 and restore the "P/S……" prompt.

5.5 Upload the "Hello_world" file to the Atmega 328 EEPROM
Press -p- then -e- then -w- and send "Text_files\Hello_world.txt several times as requested.
Finally press any key to read back the file from EEPROM.

6. Run simple User Project

Press the DPDT switch right to get the "p/r p/r p/r ….." (May need to press the pcb reset switch.)
Press -p- and send

"9_Nano_Projects\System_programs\Proj_9F_Test_Proj_1\
Proj_9F_Test_Proj_1.ino.eightanaloginputs"

At the "p/r p/r p/r ….." prompt press -r- and operate the DPDT switch.
A column of numbers will be printed out.
Press sw1 to repeat.

Operate the DPDT switch and press the reset switch to restore the "p/r p/r p/r ….." prompt.
This confirms that the Atmega328 is being programmed properly and the displays can be added.

7. Check the displays

Displays are driven using a multiplexer which can make fault finding difficult.
Repeat as for section 6 but send "Proj_9G_Test_PCB_Assembly.ino.eightanaloginputs". This turns off the multiplexer and illuminates digits one at a time.
Check for dry joints or solder bridges if any segments fail to work.

8. To upgrade the mini_OS

Push the DPDT switch right and press the reset switch.
Press -p- at the "p/r p/r p/r p/r p/r…." user prompt
and send 5_Project_pcb_168_V2.30_Arduino_V2.ino.eightanaloginputs
Run the project to get the "P/S P/S P/S……."
Press S and reload file "on-chip_strings.txt" containing programmer strings
Press P then upload the new mini-OS as described in section 5.3

Note: Remember to delete line :00000001FF from end of new mini-OS file

9. Optional system files

Mini-OS_Resources\Eeprom_subroutines_PRN_alt.c:

In early designs PRN generator write to the EEPROM was too frequent and EEPROM burnout occurred. In this file the PRN uses a different location. It should be used if the Message from the OS is always the same.

Bootloader_resources\Bootloader_main_with_OS_reset.c

This enables user projects to reset the mini-OS which is necessary if the I2C bus crashes which it does with some routines, especially I2C_Tx_2_integers().
This optional file has only been adopted for use with some boards because of the time needed for testing. **See section 14 for more details.**

10 Default User Project

At the p/r p/r prompt press -x- in place of -r-. This will give a series of options depending upon the project. These include as simple default project to check that the displays are being driven

correctly, a print out from the “Hello World” file and details about the soft ware including the version.

11 SW downloads

There are two essential software applications that must be downloaded onto this PC. They are:

The Arduino development environment. Go to https://www.arduino.cc/en/software and download it from the Microsoft store.

Bray++ version 20130820 which can be downloaded from
https://www.sites.google.com/site/terminalbpp/

Almost as important are

The portable version of Programmers notepad which can be downloaded from
http://www.pnotepad.org/download/.

An excellent text written by Joe Pardue that can be downloaded from
https://epdf.pub/c-programming-for-microcontrollers.html

The Atmega data sheet that can be downloaded from
https://www.docdroid.net/Q6jfzd3/atmega48a-pa-88a-pa-168a-pa-328-p-ds-ds40002061a-pdf

12 Getting the PCB manufacturing files

The Eagle pcb design software is required
Place the Eagle circuit design and layout drawings in a convenient folder
Open the Eagle pcb application. This takes the user to the Eagle control panel.
Click on (expand)
- CAM Jobs
- Examples and open example_2_layer.cam

Find and click on the Select Board button (Bottom left of the window)
- Navigate to the layout drawing and select it
- Its name will be confirmed at the bottom of the window

Click on
- Process Job
- Open folder (Eagle names this CAMOutputs)
- Only the DrillFiles and GerberFiles are required for pcb fabrication
- Use windows to send them to a compressed (zipped) folder.
- This can be E-mailed to a pcb manufacturer.

13. PCB assembly

Eagle circuit design and layout drawings can be found under
PCB-111000_V2\PCB 111000_Eagle_drawings together with a parts list.
The Eagle pcb design software is required to view these

Photographs of the pcb are shown below.

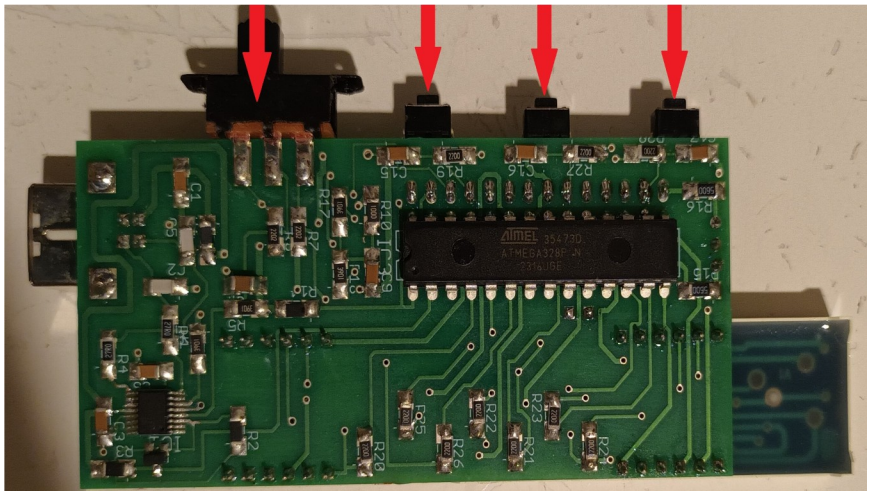
Note the locations of the following components

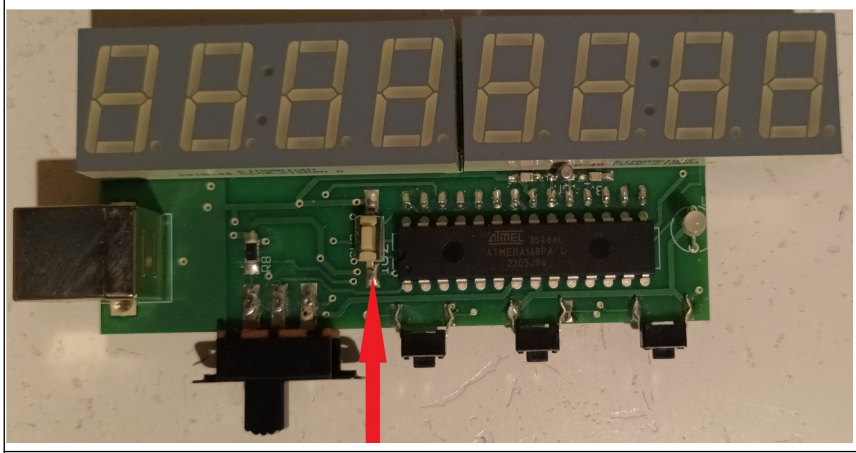
The user switches sw1, sw2 and sw3
The double pole double throw switch (DPDT)
The reset switch

The Atmega 168 which is in the top side of the pcb (note also the location of pin 1)
The Atmega 328 which is in the under side of the pcb (note also the location of pin 1)

Because of the way the two Atmega devices are assembled the pins of one of them must be soldered from the device side of the board.

Note also the orientation of the two display modules.

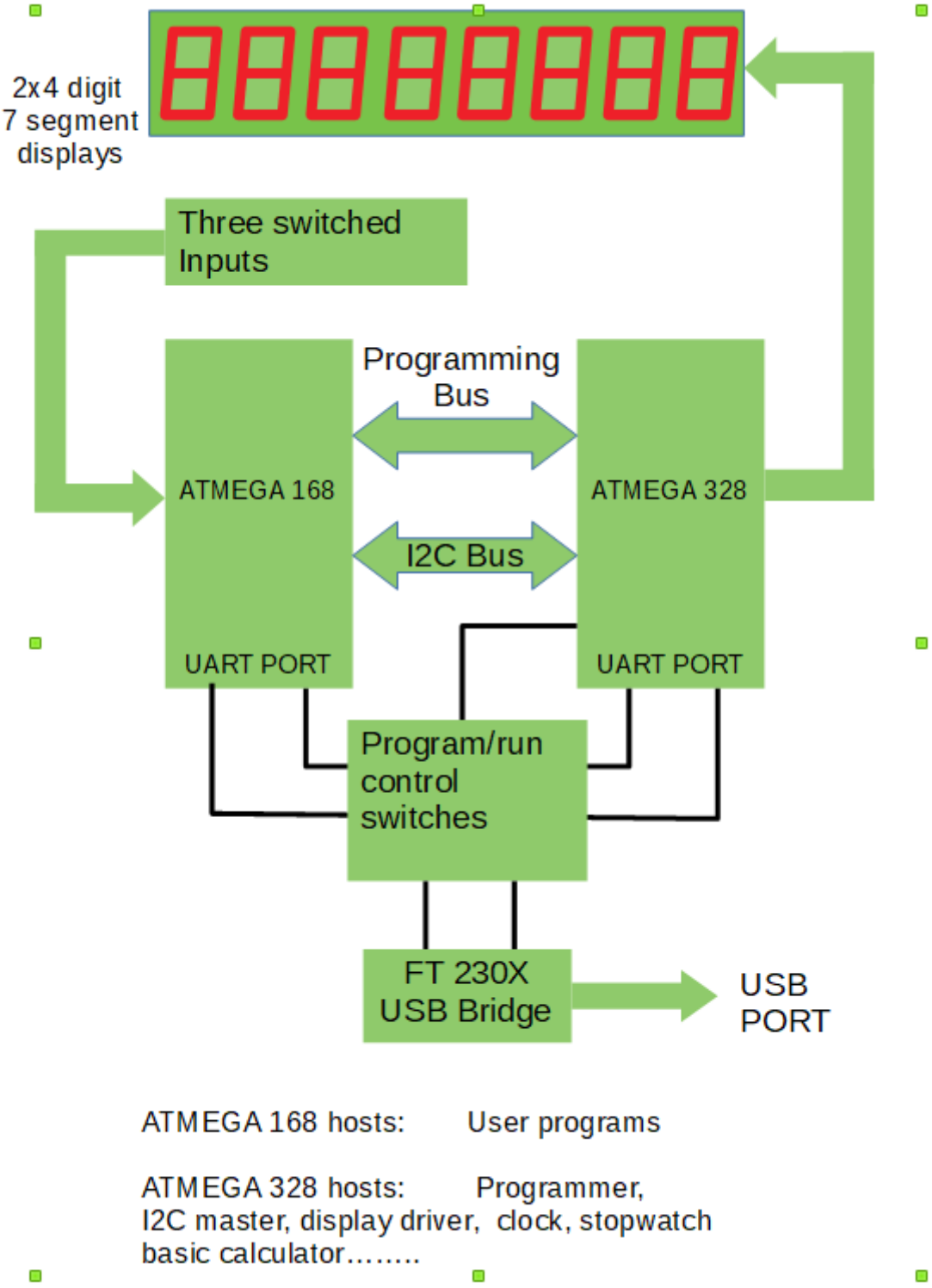
<table>
<tr>
<td>DPDT switch sw1 sw2 sw3
  </td>
<td>Photo of PCB-111000V2 under side.

Solder the Atmega328 to this side carefully noting its placement and the location of pin 1.</td>
</tr>
<tr>
<td>  </td>
<td>Photo of PCB-111000V2 top side.

Solder the Atmega168 to this side carefully noting its placement and the location of pin 1.

Note also the orientation of the display modules</td>
</tr>
<tr>
<td>Reset switch</td>
<td></td>
</tr>
</table>

14 System Block diagram



15 I2C bus crash

Projects designed to create patterns on the display are liable to crash if left running indefinitely or even after say 5 minutes.

A major cause of this is the I2C bus: slave and master somehow get out of step with the result that the slave finds itself waiting for a signal from the master that has already gone or for some reason never comes. The watch dog timer with interrupt is used to detect this situation. The interrupt service routine puts a reset pulse on the mini-OS IC. This alone Causes program control to jump to the programmer (location 0x7000). Therefore immediately before the reset pulse a low is put on one of the programming bus lines (PB4). The programmer detects this and sends program control to location 0x0000 (i.e. the mini-OS).
The user project "waiting_for_I2C_master" macro must be modified as follows:

activate interrupts to enable a jump to the watch dog ISR
Ensure that a time out is generated after a short (but unknown) wait as follows:
Replace the line

```
while (!(TWCR & (1 << TWINT)));
```

or

```
while (!(TWCR & (1 << TWINT)))wdr();
```

with the lines

```
{int m = 0; while((!(TWCR & (1 << TWINT))) && (m++ < 5000))wdr();\
if (m >= 4999){sei(); while((!(TWCR & (1 << TWINT))));}}\
```

These changes have only been made in projects

5I_Noise_waveform
5J_Gravity
5J_pendulum

They may easily generate unexpected side effects in other projects.

To indicate that the special version of the programmer is being used
the "p/r p/r p/r…….. user prompt in file

Bootloader_main_with_OS_reset.c

Has been changed to "p/g p/g p/g……."

16 Programmer Bug

If -p- is pressed at the p/r prompt the first task of the programmer is to switch the target device into programming mode after which its programming memory is reset and the user is requested to send the code. Occasionally the device fails to enter programming mode however this condition has never persisted long enough to be investigated. This may be because there is a dry joint connecting the Atmega reset pins high via the 10K resistors.

In some versions of PCB111000, programming memory has only been reset after the first character of the new code has been detected. This was not found to work reliably in PCB111000_V2 so once -p- has been selected there is no escape and programming memory must be refreshed if anything is to work even the default program.

17 Mini-OS bug

The Mini-OS was originally developed using winAVR with the compiler set for full optimisation, in other words to make the executable file as small as possible.
WinAVR has subsequently been replaced by Atmel Studio 7. For this compiler full optimisation generates an executable file that is even smaller than the one generated by

WinAVR. Unfortunately the mini-OS failed to function with this level of optimisation. When optimisation was turned off normal operation of the mini-OS was restored.

It is now believed that it is the I2C bus that fails with full optimisation. Ideally this should be investigated and the fault remedied so that with full optimisation and using Studio 7 everything works correctly.
This investigation has not been carried out. Instead use of the watch dog timer has been investigated. In many ways this is the more useful approach since the watch dog has widespread applications.