Building the Opencockpits COMM Radio

With this tutorial I hope to partly address the lack of adequate instruction available for assembly of the Opencockpits radios. This is a shame as the Opencockpits hardware and software offer some of the best value for the home cockpit builder on a budget. There are a number of websites that proved invaluable to my knowledge and understanding of Opencockpits hardware and software and I would highly recommend researching these sites prior to undertaking this build. You'll find a full list at the end of this document.

This tutorial assumes that the user already owns the Opencockpits Mastercard as the associated 7-segment Display cards need to connect to this unit in order to operate. You should also have an understanding of the software SIOC which is used to program the different functions of the radios. A sample file is included in the Appendix.

The COMM panel comprises a front plate laser cut and ready for backlighting, backplate, switch mounting plate and buttons.

In addition you will need the following to complete the radio:

- 2 x OMROM B3F 4050 keyswitches
- 12 x 7-segment display digits
- 1 x CTS288 rotary encoder
- 2 x PCB's for 6 digit display
- 1 x Gray knob
- 1 x Opencockpits Display Card
- Lots of multipin plugs and cables
- Plenty of time and patience
- Coffee, wine, beer or a mix of all

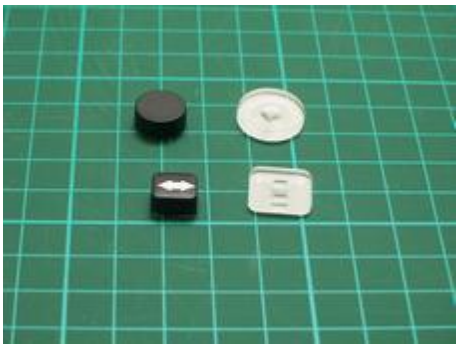
NOTE: The 7-segment displays need to be the Common Cathode variety and all are available from the Opencockpits Shop

Other options you might consider are:

- Dual concentric rotary encoder or alternatively use two rotary encoders to achieve High & Low digit selection (this is what I did).

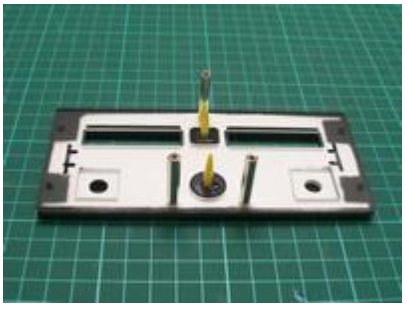
- 3mm White LED's for backlighting. (Installation details at the end of this document)

The Transfer and Comm Test buttons come in two pieces. Simply glue the black button onto the clear base with superglue. The backing plate is clear to allow light through when backlit.



Here's the result. It will only take a few minutes using superglue. Edges need to be painted black to avoid light leakage when backlit. Use acrylic paint available from any craft shop.

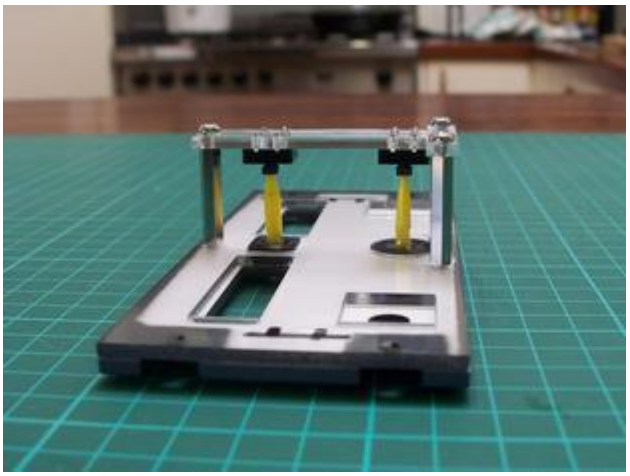
Then some sort of pins need to be installed that will push on the switch mounted above it. I used plastic knitting needles cut to size and glued in place. Be sure to get the length correct. The pointed ends insert into the small hole in the Omron momentary switch for button centering.



Buttons fit into the backplate and 25mm threaded standoff shafts hold the frontplate to the back and the switches above the buttons as can be seen in the next photo.

NOTE: You might be wondering what the extra hole is in the right side of my panel. I decided to modify the radio to incorporate 2 rotary encoders providing both High & Low digit selection. Other options include installing a dual concentric encoder or using just 1 encoder with High & Low selection activated by the encoder's push button. You decide depending on your level of realism and access to the various components.

Glue the Omron switches onto the clear mounting plate and align with the needle ends.



Here you can clearly see how the needle ends insert into the switches and stop them flailing about.

Encoders come with long shafts that need to be cutdown. Measure the required length which will depend on your choice of knob.

Cut with a hacksaw.

Note that these encoders available from Opencockpits incorporate a pushbutton switch. The 3 top legs are for the encoder (known as "A", "C" & "B" legs) and the 2 bottom legs are for the pushbutton. The outer "A" & "B" legs of the encoder connect to two **consecutive** inputs of the Mastercard and the centre "C" leg connects to the corresponding Ground.

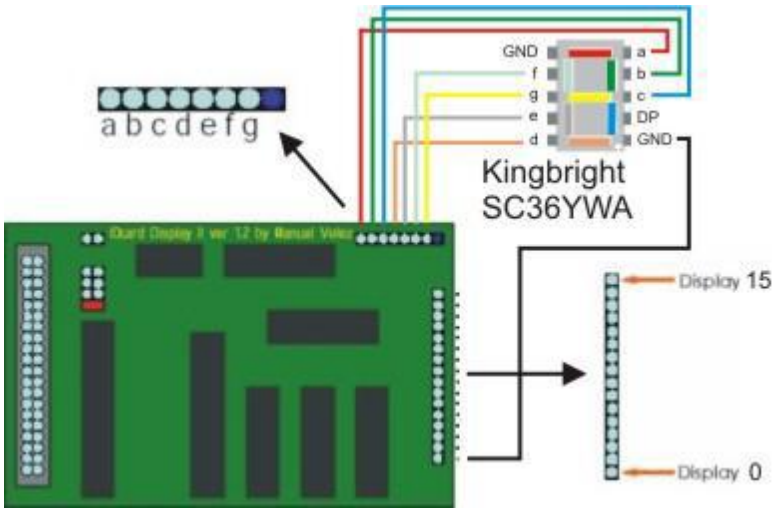
For my radio two encoders provide high and low digit selection. You may choose to use the more authentic dual concentric encoder or a single encoder with the encoder's pushbutton alternating between High & Low selection.

NOTE: Pushbutton selection of high and low selection is done within the code of SIOC. Search the Opencockpits forum for example code to achieve this.

This is what it should look like all ready for the electronics to be installed.

Now onto the heart of the system, the displays board. The Opencockpits display board controls up to 16 x 7-segment displays connected as shown in this diagram. The displays PCB caters for all the associated connections to the card via multipin cables that you will need to make.

More on that later.

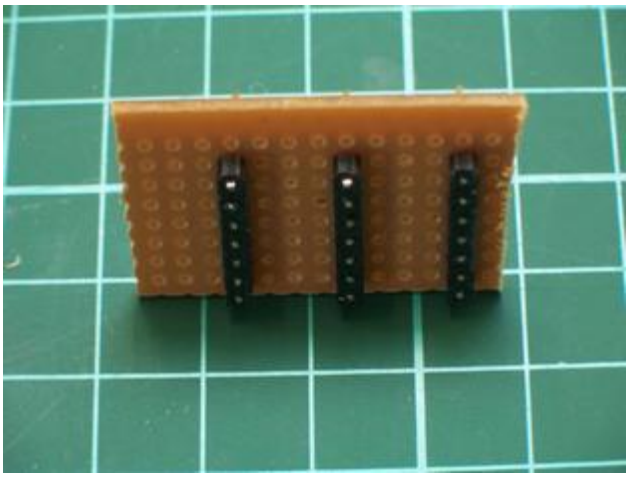


As you can see from the picture above, the 7-segments of each display PCB connects to the board via a 7pin connector and you may have to connect several 7 pin connectors to one board. Some use splitter cables but I decided to take a pcb approach and solder a mulitpin board to the card using strip veroboard.

The Common Cathodes connect to the pins numbered 0 – 15.

Only takes a few minutes and provides easier multipin connection.

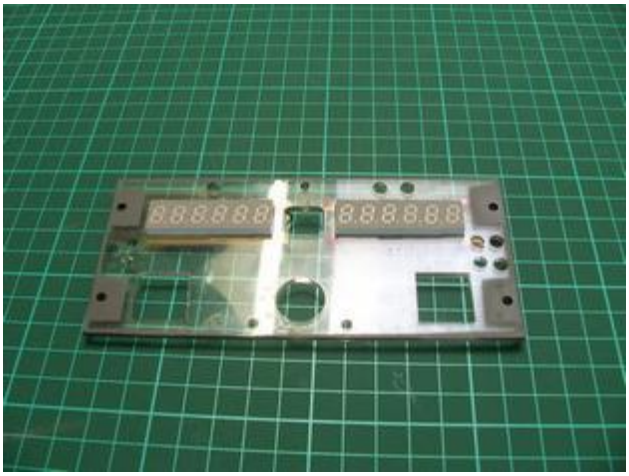
Standard pcb pins were used but due to the need to push them through from the underside I modified them by pushing the plastic support down to the base of the pins.

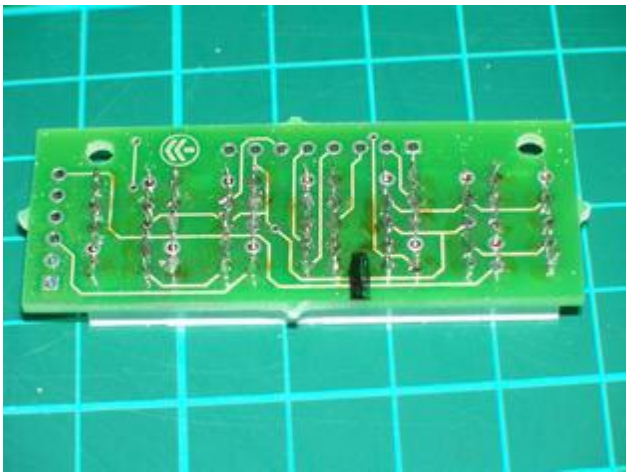


Here you can see the rows of pins inserted from the underside of the veroboard.

Now onto the construction of the 7-segment displays. Again always use the mounting PCB's. They only cost a few dollars each and make it a breeze to install.

Place the displays onto the PCB's then mount the backplate over the displays prior to soldering. Turn the whole lot over and solder the displays onto the PCB's while mounted in the backplate. This will allow the displays to more easily insert through the backplate later on during final assembly. These backplates are laser cut to very fine tolerance and if you solder without doing this you'll end up needing to file the backplate holes to fit the displays. (Ask me how I know this!)

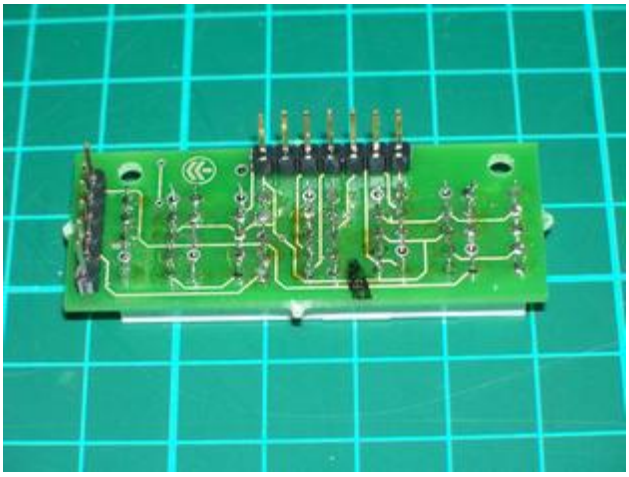




This is the rear of the 6 digit display card. The black mark indicates which display has the decimal point.

Make sure you place the 7-segs onto the front of the PCB and solder to the rear. The front is indicated by the Opencockpits logo and writing.

NOTE: You will only need to solder pins 2(f), 3(g), 4(e), 5(d), 6(GND), 8(c), 9(b), 10(a). Pin 7 is the decimal point and will need a separate wire connecting to a Mastercard output. Pin 1 can be left unsoldered as it's just another Ground pin.

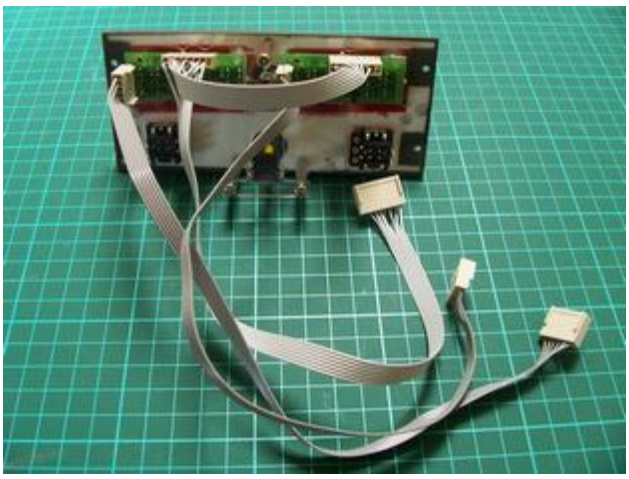


Solder connection pins onto the rear of the PCB to provide multipin connectors. The top row of 7 pins are for the 7 segments of each display (a,b,c,d,e,f,g,GND) and the 6 pins to the left are for the common cathode of each display. In this case there are 6. The 5 display PCB will have 5 pins and the 3 display PCB has 3. Clever hey!

Secure your completed displays PCB's onto your panel with hot glue at each corner.

HOT TIP! To connect the Decimal Point (pin #7) wire it to a spare output of the Mastercard but don't connect a GND cable from the display to the output card. The output will utilize the display card GND and therefore its own buffering and you won't need a resistor in circuit. The DP will also be the same brightness as the other digits. I connected both stby & act DP to a single Mastercard output.

NOTE: Some builders advocate hard wiring the first digit (#1) to a Mastercard output as it never changes, however Nico Kaan recommends wiring as a fully operational digit driven from the display card. The has the advantage of complete control over the digit via SIOC therefore allowing display dimming and full display testing by writing "8" to the display when pressing the Test switch. This approach does consume one extra display card output per frequency; however it also offers greater flexibility. It also simplifies soldering and you don't have to cut the circuit tracks in order to isolate the first digit which you need to do if you hard wire it.



Your next major work is making up cables to connect your displays to your boards. I highly recommend making up multipin connectors. Don't try soldering directly to the display pins. Whatever method you end up using you will have a lot of pin connections to make, it's unavoidable!!

As you can see the 7-segment cables can be paralleled together but the Cathode connections need separate cables to connect to their respective pins on the display board. One COMM radio consists of 12 digits so you'll need one display board. **See Appendix 1 for the circuit diagram of my complete pedestal.**

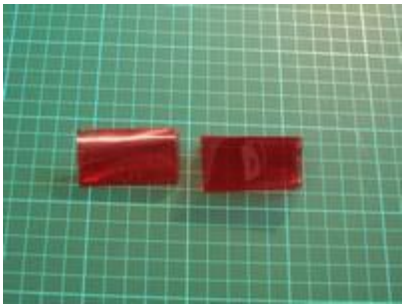
NOTE: If you intend building multiple units as I did you will need to decide which displays connect to which board, remembering there is a maximum of 16 displays per board. This will dictate how many cables you'll need and how many 7-segment cables can be paralleled together. The cable lengths will also be dependent upon where your display boards are located in relation to the radio, but I would strongly advise making them longer than expected to allow for ease of access. If you end up building a complete pedestal setup you'll have a lot of cables to contend with.

The digital displays look more aesthetically appealing with a coloured lens over them. I used standard red cellophane sandwiched between the front and back plates and it works well.

The only criticism I have is that it's a little flimsy and prone to tearing but if you take care it is very cheap and easy and quite effective.

Cut two strips about 50mm x 100mm.

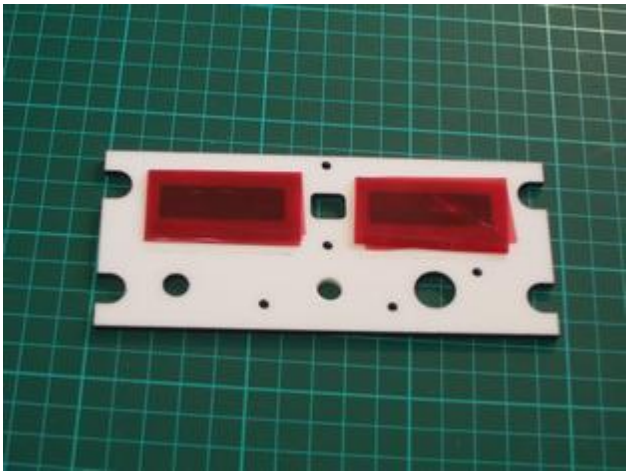
And fold into 3.

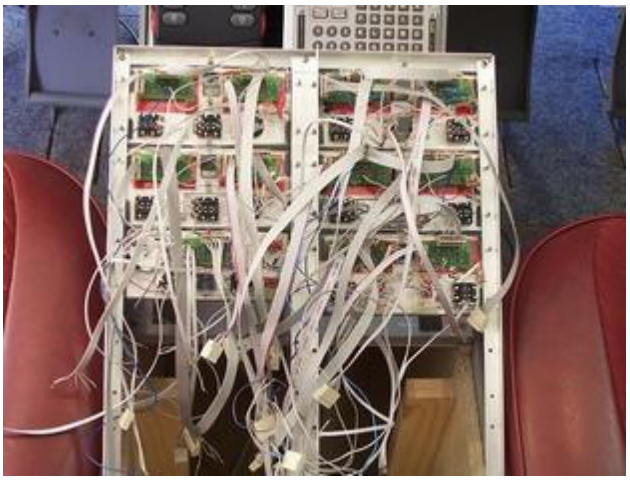


Use very thin double sided tape to stick the cellophane in place. You should be able to get this tape at office stationary stores. Don't use the foam style double sided mounting tape, it's far too thick.

Note: This shows the Nav panel but the principle is exactly the same for all panels.

And here is the result. One advantage of cellophane is that it is very thin and doesn't provide too much bulk between the front and back plates.





Here's an example of the final wiring mess for the full set of radios in the pedestal. It's really important therefore to take care making up cables to the length required and keeping good records of connections.

BACKLIGHTING

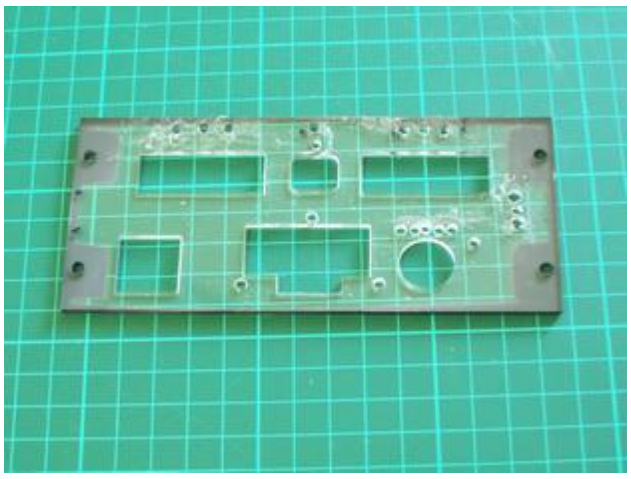
You may already have a preference for backlighting but I found after my own research and testing that I preferred the whiter glow and lower current draw of LED's. I used 3mm White LED's with a brightness of 15000mcd purchased on ebay from China. These are VERY bright and I could easily have gone for a lower brightness, therefore I would recommend something around 10000mcd. You can even go for the warm white variety if you prefer the warmer look similar to incandescent globes. There's nothing fancy about this method and therefore it can be applied simply and cheaply by anyone with the most meagre of skills.

I'm driving my backlighting from a 12v DC power supply and connecting them in a series configuration. The maximum number of these LED's that can be driven from 12v in series is 3, so most of my LED's are grouped this way along with a resistor. These LED's have a working voltage of 3.4v and current draw of 20mA which results in a resistor requirement of 100 ohms per 3 LED's.

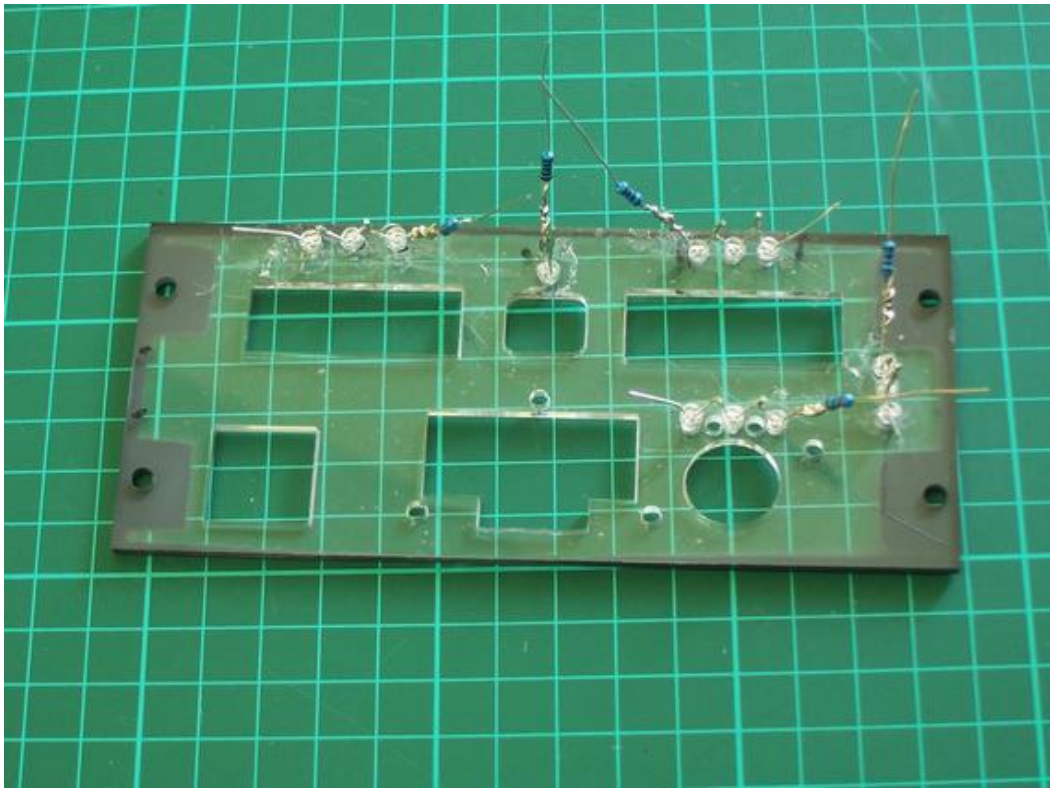
NOTE: Take a look at http://led.linear1.org/led.wiz for a very useful LED array resistor calculator. Also be sure to connect the LED's with the correct polarity i.e. -ve leg of one LED to the +ve leg of the next and so on. The resistor can be connected to either the -ve or +ve end of the array but for convention stick with one end throughout your arrays. I connected my resistors to the –ve end of all arrays.

Drill 3mm holes in the backplate to mount the LED's. Use the front plate as a guide to mark the position of the holes depending on where the writing is that you want to backlight.

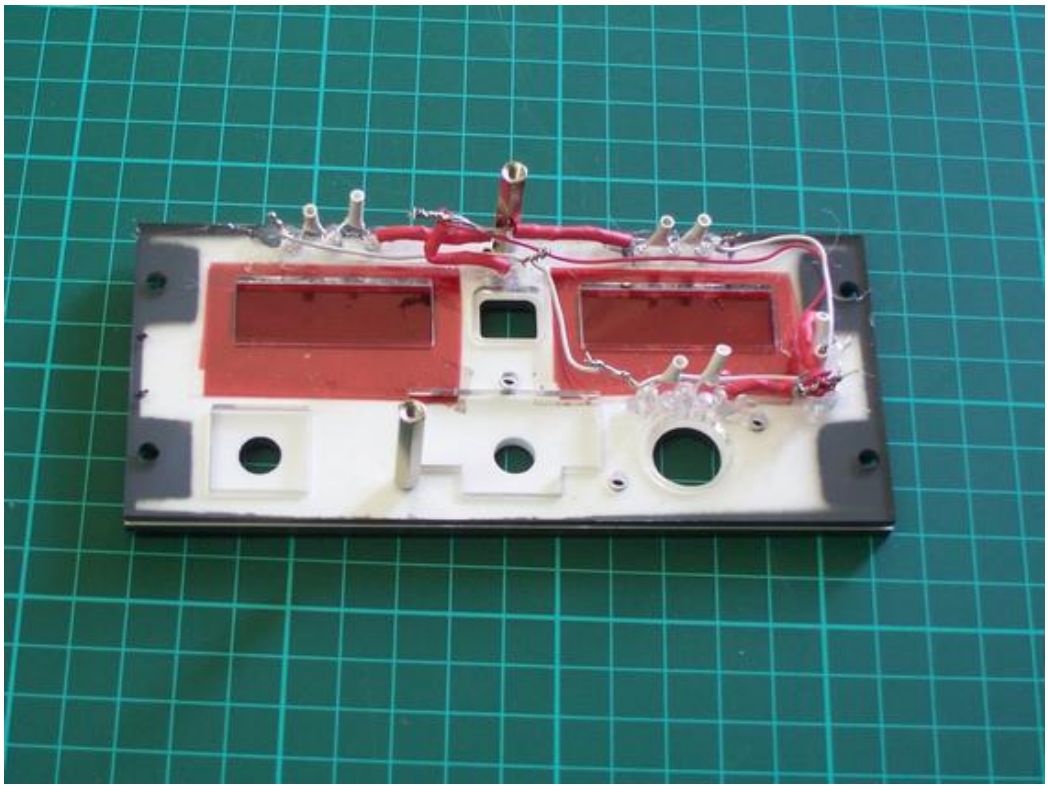
Again I'm using the Nav panel to demonstrate but the principle is the same for all panels only the hole locations will differ.



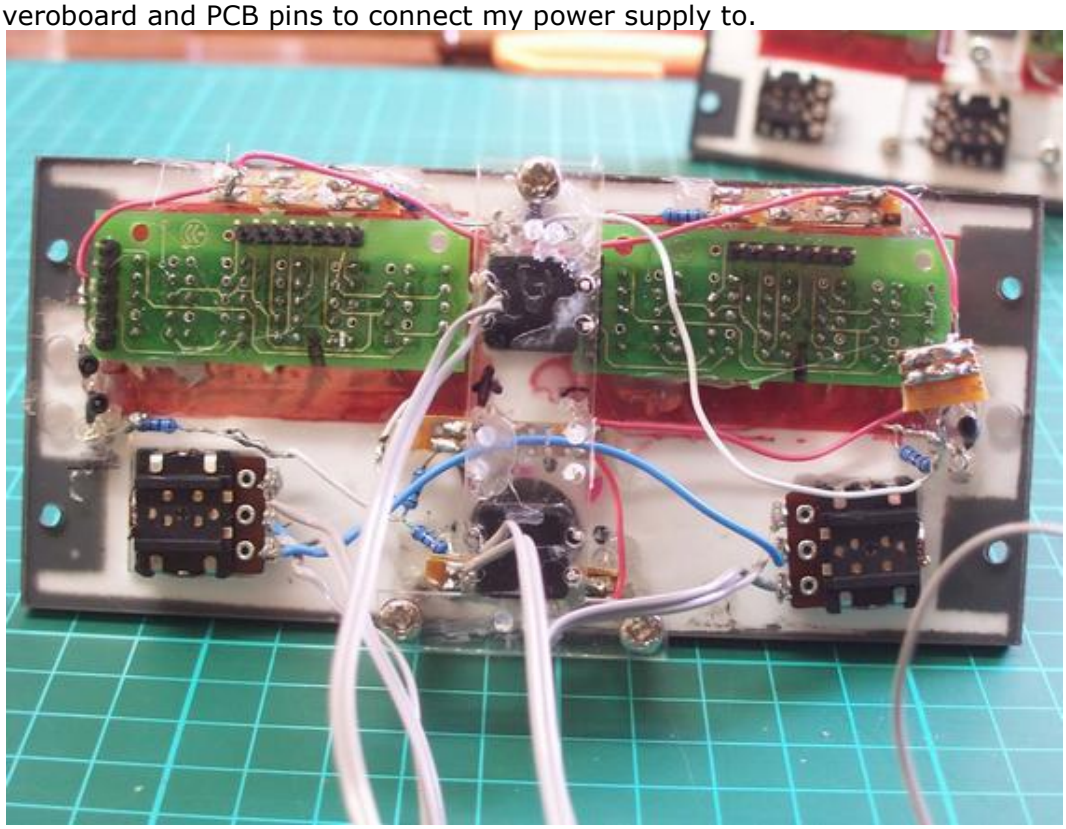
Place the LED's in the holes and adjust for correct placement then use hot glue to hold them in place. Try to group –ve and +ve ends in close proximity as it will be easier to connect them together later.



Use heatshrink to protect resistor legs and other wires that you don't want contacting.



Now join all -ve ends of each LED group together and all +ve ends. Your 12v supply connects to the respective -ve & +ve legs of the completed circuit. I used a piece of strip veroboard and PCB pins to connect my power supply to.



And here is the result. Not bad hey?

This process of backlighting should take approximately 1 hour per panel from start to finish.

The final process of getting the radio working is programming the operations in SIOC. I'm not going into detail here about this process but I would point you to Nico Kaan's website at http://www.lekseecon.nl/sioc.html for an explanation of this very powerful software. You can also download examples of code to adapt and use for your radio. Appendix 2 contains an example file that works with my radio.

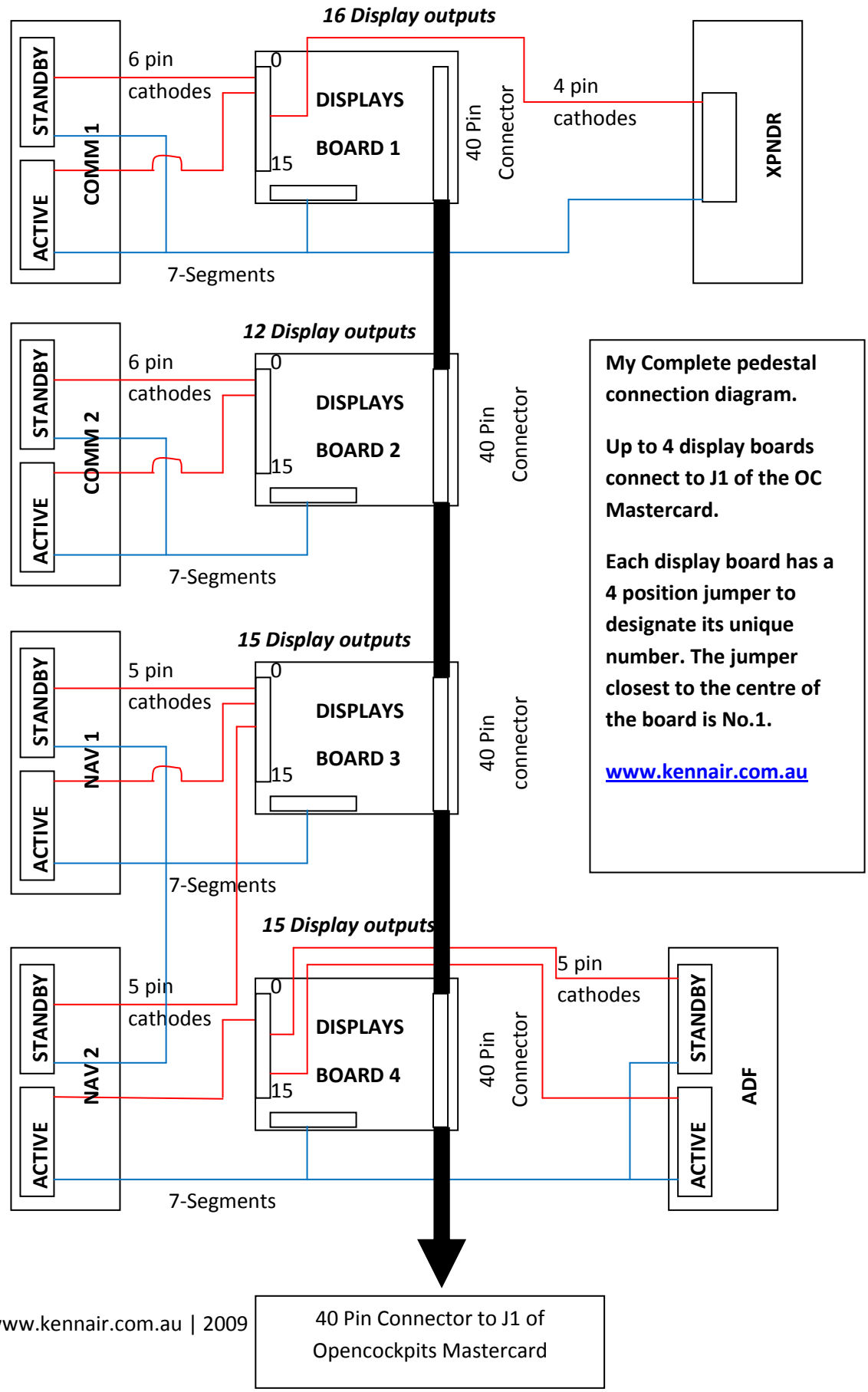
Weblinks List:

- http://www.opencockpits.com/catalog/index.php
- http://www.opencockpits.com/
- http://personales.ya.com/micabina737/indexi.htm.
- http://www.lekseecon.nl/
- http://led.linear1.org/led.wiz

For more detailed photos used in this guide, go to my website at http://www.kennair.com.au/avionics.html.

Ken Brand.

Appendix 1.



My Complete pedestal connection diagram.

Up to 4 display boards connect to J1 of the OC Mastercard.

Each display board has a 4 position jumper to designate its unique number. The jumper closest to the centre of the board is No.1.

www.kennair.com.au

Appendix 2.

This is a comm1 & 2 file courtesy of Nico Kaan (http://www.lekseecon.nl/) and works with this radio build. Copy this text into a Notepad file, change the input/output and display digit numbering to comply with your hardware and import into SIOC. Delete the COMM2 section if not required.

NOTE: This file caters for a 5 digit frequency display and not the full 6 this tutorial helps to build which is no problem; you will just have a frequency to 2 decimal places instead of 3 and an unused last digit. It utilizes software control over the fixed digit "1" and decimal point and it also gathers the current radio frequency setting for the aircraft loaded and writes it to the displays at startup. There is also the facility to incorporate a master avionics switch. This function has been commented out but can be restored if required.

```
// ********************************************************************************
// * Config_SIOC ver 1.98  -  By Manolo Vélez  -  www.opencockpits.com
// ********************************************************************************
// * FileName : Comm1_2.txt
// * Author Nico Kaan
// * Date : 10/11/2007
// * modified by Nico Kaan, April 18th, 2008
// * also incorporates a display test function
//
//*********************************************
//***** COMM 1 ********************************
//*********************************************

Var 1000 name Com1_act

Var 1002 name FS_C1_SB_BCD Link FSUIPC_INOUT Offset $311A Length 2 // FS COM1 standby frequency in BCD format
Var 1003 name FS_C1_ACT_BCD Link FSUIPC_INOUT Offset $034E Length 2 // FS COM1 active frequency in BCD format
{
&Com1_act = FROMBCD &FS_C1_ACT_BCD
CALL &OutCom1_Act
CALL &C1_LH_ROT_VAL // Align rotary with new C1 SB integer val
CALL &C1_RH_ROT_VAL // Align rotary with new C1 SB decimal val
CALL &C1_UPDATE_LHS
CALL &C1_UPDATE_RHS
CALL &OutCom1_SB
}

Var 1004 name FS_C1_SWAP Link FSUIPC_OUT Offset $3123 Length 1 // FS COM1 active standby swap

Var 1005 name C1_SB_AS_DEC // COM1 standby frequency in decimal format
Var 1006 name C1_SB_LHS // COM1 standby frequency integers
Var 1007 name C1_SB_RHS // COM1 standby frequency decimals
Var 1008 name C1_RHS_ROT_VAL // COM1 right hand rotary encoder ref value
Var 1009 name C1_LHS_ROT_VAL // COM1 left hand rotary encoder ref value

Var 1010 name ROT_C1_LHS Link IOCARD_ENCODER Input 18 Aceleration 1 Type 2
{
L0 = &ROT_C1_LHS * -1
&C1_LHS_ROT_VAL = ROTATE 18 36 L0
CALL &C1_UPDATE_LHS
CALL &SET_FS_C1_SB
CALL &OutCom1_SB
}

Var 1011 name ROT_C1_RHS Link IOCARD_ENCODER Input 20 Aceleration 1 Type 2
{
L0 = &ROT_C1_RHS * -1
&C1_RHS_ROT_VAL = ROTATE 0 39 L0
CALL &C1_UPDATE_RHS
CALL &SET_FS_C1_SB
CALL &OutCom1_SB
```

```
}

Var 1012 name SW_C1_SWAP Link IOCARD_SW Input 26 Type 1 // Active/Standby swap button
{
&FS_C1_SWAP = TOGGLE 3
}

Var 1015 name SW_C1_TEST Link IOCARD_SW Input 28 // COMM TEST button sends 8's to Displays
{
IF &SW_C1_TEST = 1
  {
   &DISP_C1_ACT = 88888
   &DISP_C1_SB = 88888
  }
ELSE
  {
   CALL &OutCom1_Act
   CALL &OutCom1_SB
  }
}

Var 1020 name DISP_C1_SB Link IOCARD_DISPLAY Digit 1 Numbers 5 // Standby decimals
Var 1021 name DISP_C1_ACT Link IOCARD_DISPLAY Digit 7 Numbers 5 // active display
Var 1022 name C1_DP Link IOCARD_OUT Output 15 // Decimal Point

Var 1030 name C1_LH_ROT_VAL Link SUBRUTINE // Aligns LH rotary ref with COM1 SB freq
{
L0 = FROMBCD &FS_C1_SB_BCD
L1 = L0 / 100
&C1_LHS_ROT_VAL = TRUNC L1
}

Var 1031 name C1_RH_ROT_VAL Link SUBRUTINE // Aligns RH rotary ref with COM1 SB freq
{
L0 = FROMBCD &FS_C1_SB_BCD
L1 = &C1_LHS_ROT_VAL * 100
L2 = L0 - L1
&C1_RHS_ROT_VAL = L2 / 2.487179
}

Var 1032 name C1_UPDATE_LHS Link SUBRUTINE // Update COM1 stanby integers
{
&C1_SB_LHS = &C1_LHS_ROT_VAL
}

Var 1033 name C1_UPDATE_RHS Link SUBRUTINE // Update COM1 standby decimals
{
L0 = &C1_RHS_ROT_VAL * 2.5
&C1_SB_RHS = TRUNC L0
}

Var 1034 name SET_FS_C1_SB Link SUBRUTINE // Update COM1 standby frequency in FS
{
L0 = &C1_SB_LHS * 100
L1 = L0 + &C1_SB_RHS
&FS_C1_SB_BCD = TOBCD L1
}

Var 1040 name OutCom1_Act Link SUBRUTINE
{
//IF &MAST_SW = 1
//{
L0 = &Com1_act
&DISP_C1_ACT = L0 + 10000
}
//ELSE
//{
//&DISP_C1_ACT = -999999 // blank
//}
//}

Var 1041 name OutCom1_SB Link SUBRUTINE
```

```
{
//IF &MAST_SW = 1
//{
&C1_DP = 1
L0 = &C1_SB_LHS * 100
L1 = L0 + &C1_SB_RHS
&DISP_C1_SB = L1 + 10000
}
//ELSE
//{
//&C1_DP = 0
//&DISP_C1_SB = -999999 // blank
//}
//}
//***********************************************
//***** COMM 2 *********************************
//***********************************************

Var 1050 name Com2_act

Var 1052 name FS_C2_SB_BCD Link FSUIPC_INOUT Offset $311C Length 2 // FS COM2 standby frequency in BCD format
Var 1053 name FS_C2_ACT_BCD Link FSUIPC_INOUT Offset $3118 Length 2 // FS COM2 active frequency in BCD format
{
&Com2_act = FROMBCD &FS_C2_ACT_BCD
CALL &OutCom2_Act
CALL &C2_LH_ROT_VAL // Align rotary with new C2 SB integer val
CALL &C2_RH_ROT_VAL // Align rotary with new C2 SB decimal val
CALL &C2_UPDATE_LHS
CALL &C2_UPDATE_RHS
CALL &OutCom2_SB
}

Var 1054 name FS_C2_SWAP Link FSUIPC_OUT Offset $3123 Length 1 // FS COM2 active standby swap

Var 1055 name C2_SB_AS_DEC // COM2 standby frequency in decimal format
Var 1056 name C2_SB_LHS // COM2 standby frequency integers
Var 1057 name C2_SB_RHS // COM2 standby frequency decimals
Var 1058 name C2_RHS_ROT_VAL // COM2 right hand rotary encoder ref value
Var 1059 name C2_LHS_ROT_VAL // COM2 left hand rotary encoder ref value


Var 1060 name ROT_C2_LHS Link IOCARD_ENCODER Input 22 Aceleration 1 Type 2
{
L0 = &ROT_C2_LHS
&C2_LHS_ROT_VAL = ROTATE 18 36 L0
CALL &C2_UPDATE_LHS
CALL &SET_FS_C2_SB
CALL &OutCom2_SB
}

Var 1061 name ROT_C2_RHS Link IOCARD_ENCODER Input 24 Aceleration 1 Type 2
{
L0 = &ROT_C2_RHS
&C2_RHS_ROT_VAL = ROTATE 0 39 L0
CALL &C2_UPDATE_RHS
CALL &SET_FS_C2_SB
CALL &OutCom2_SB
}

Var 1062 name SW_C2_SWAP Link IOCARD_SW Input 34 Type 1
{
&FS_C2_SWAP = TOGGLE 2
}

Var 1065 name SW_C2_TEST Link IOCARD_SW Input 36 // COMM TEST button sends 8's to Displays
{
IF &SW_C2_TEST = 1
  {
   &DISP_C2_ACT = 88888
   &DISP_C2_SB = 88888
  }
ELSE
```

```
  {
   CALL &OutCom2_Act
   CALL &OutCom2_SB
  }
}

Var 1070 name DISP_C2_SB Link IOCARD_DISPLAY Digit 17 Numbers 5 // Standby decimals
Var 1071 name DISP_C2_ACT Link IOCARD_DISPLAY Digit 23 Numbers 5 // active display
Var 1072 name C2_DP Link IOCARD_OUT Output 17 // Decimal Point

Var 1080 name C2_LH_ROT_VAL Link SUBRUTINE // Aligns LH rotary ref with COM2 SB freq
{
L0 = FROMBCD &FS_C2_SB_BCD
L1 = L0 / 100
&C2_LHS_ROT_VAL = TRUNC L1
}

Var 1081 name C2_RH_ROT_VAL Link SUBRUTINE // Aligns RH rotary ref with COM2 SB freq
{
L0 = FROMBCD &FS_C2_SB_BCD
L1 = &C2_LHS_ROT_VAL * 100
L2 = L0 - L1
&C2_RHS_ROT_VAL = L2 / 2.487179
}

Var 1082 name C2_UPDATE_LHS Link SUBRUTINE // Update COM2 stanby integers
{
&C2_SB_LHS = &C2_LHS_ROT_VAL
}

Var 1083 name C2_UPDATE_RHS Link SUBRUTINE // Update COM2 standby decimals
{
L0 = &C2_RHS_ROT_VAL * 2.5
&C2_SB_RHS = TRUNC L0
}

Var 1084 name SET_FS_C2_SB Link SUBRUTINE // Update COM2 standby frequency in FS
{
L0 = &C2_SB_LHS * 100
L1 = L0 + &C2_SB_RHS
&FS_C2_SB_BCD = TOBCD L1
}

Var 1090 name OutCom2_Act Link SUBRUTINE
{
//IF &MAST_SW = 1
//{
L0 = &Com2_act
&DISP_C2_ACT = L0 + 10000
}
//ELSE
//{
//&DISP_C2_ACT = -999999 // blank
//}
//}

Var 1091 name OutCom2_SB Link SUBRUTINE
{
//IF &MAST_SW = 1
//{
&C2_DP = 1
L0 = &C2_SB_LHS * 100
L1 = L0 + &C2_SB_RHS
&DISP_C2_SB = L1 + 10000
}
//ELSE
//{
//&C2_DP = 0
//&DISP_C2_SB = -999999 // blank
//}
//}
```