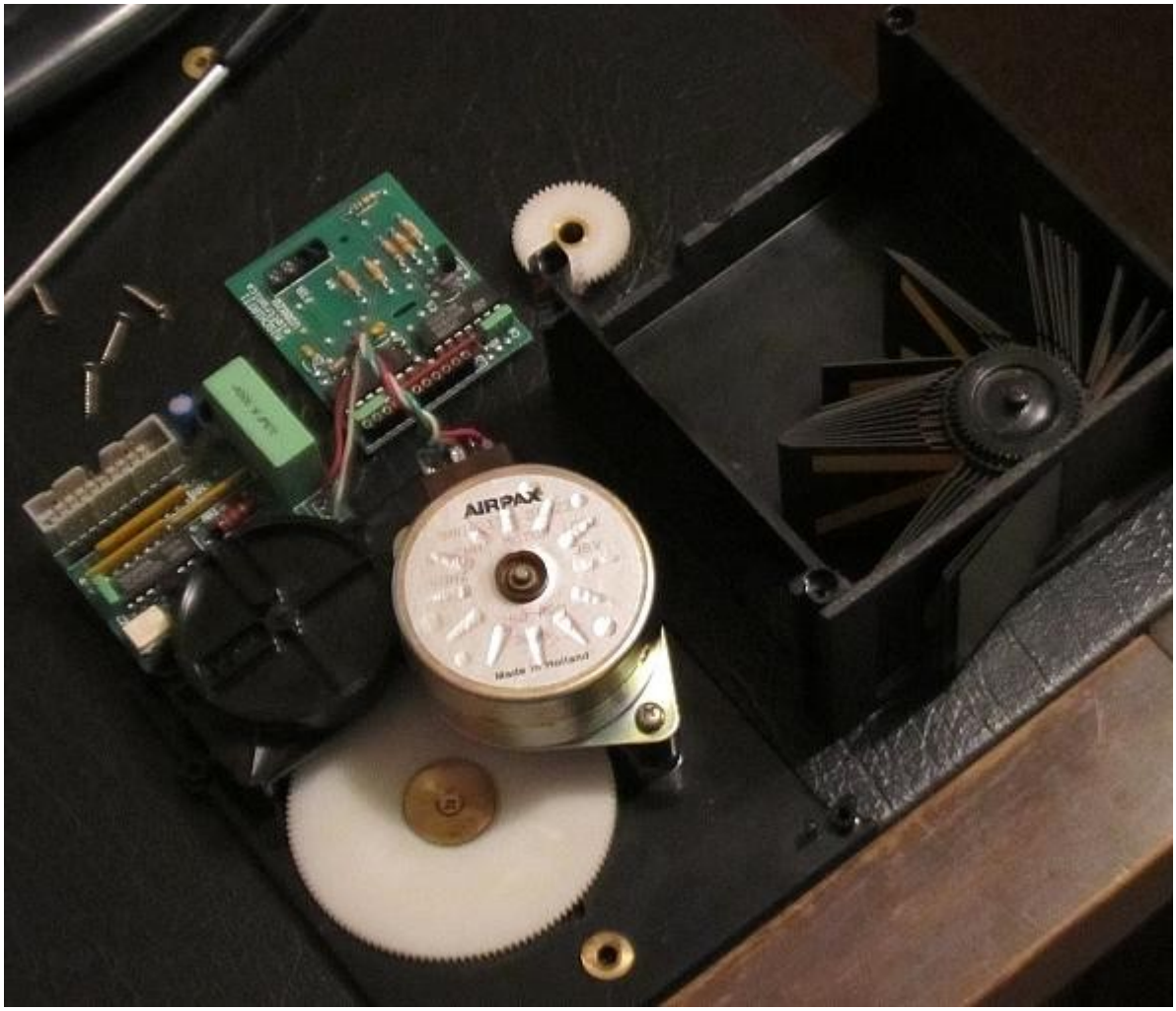
Reverse engineering and creating a controller for Solari soft flap display

What follows is a very rough documented account of how I reverse engineered some of the above solari soft-flap display units and produced a more upto date controller for my client who bought 20 of them at a recent on-line auction

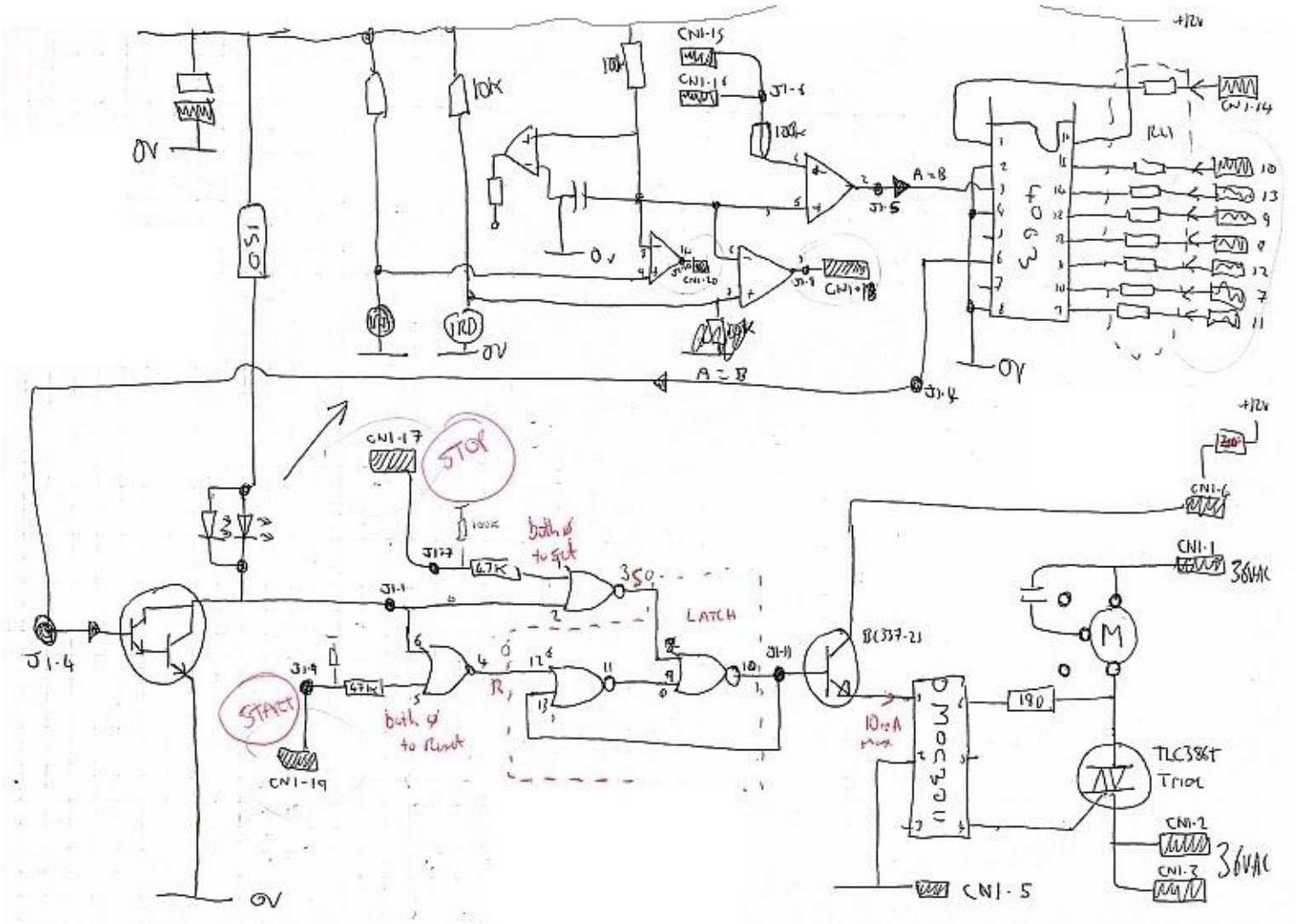
Armed with a screwdriver the first thing I did was pull the unit apart to see how it ticks. The dismantled unit is below, as you can see it consists of a 36VAC synchronous motor with two PCB's containing the motor control logic. I have had very little dealings with synchronous motors in the past, mostly old belt drive record players back in the eighties which worked at the 50hz frequency of the mains supply.



Synchronous motors are only usually found these days in higher power industrial stuff and most small motors have been replaced by stepper motors as these are simpler to control from a microprocessor and very cheap, Synchronous motors however provide a very accurate RPM rate and by reducing the gearing as they have done here it takes a leisurely 3 seconds to do a full revolution of the flapper display.

Since we had no luck contacting the current maintenance providers for the units, the operation of the control circuitry had to be reverse engineered from scratch :-(

The display unit has two PCB's connected together as shown, the smaller PCB has the manufacturer and part number, but a search through google for MISCHIATTI ELETTONICA U2802B provided nothing of interest. So began the arduous work of re-creating the circuit diagram using a magnifying glass and multi-meter. My first (and only) effort was hand-drawn and is shown below. (I'm not 100% its correct so I'm not going to waste time re-drawing it on a computer)

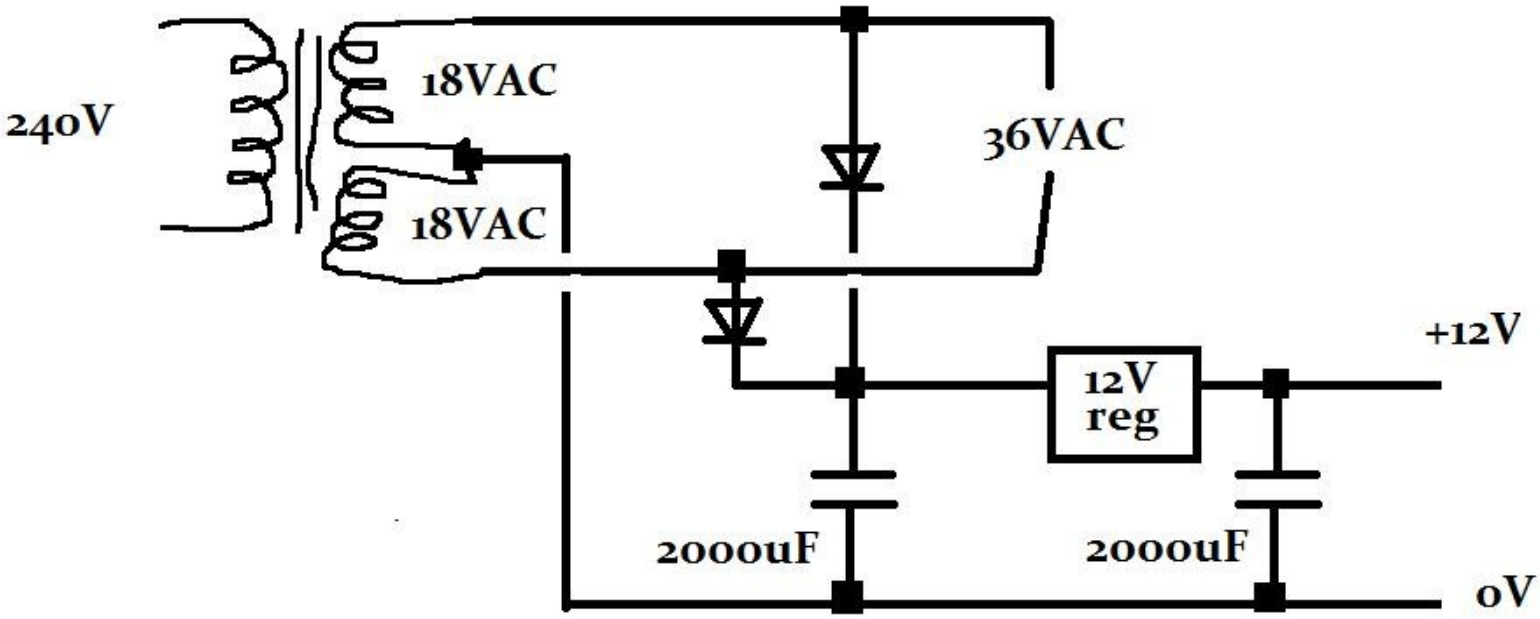


From this you can deduce several things, firstly (except for the obvious fact I cannot draw!) there is no real control logic on the board. I had sort of assumed these units would be daisy chained back to a controller on some sort of multi-drop bus such as RS422, but as you can see each board would need an individual 20 pin cable back to another controller or a bigger mother board. The 4063 comparator is obviously part of this bigger picture and looked far too complicated to even guess how they had done this on the real control board, so I knew from first base I was never going to try and reproduce the way it was originally done.

The other parts of the system included 4 analogue comparators and photo-diodes which form an optical shaft encoder with the LED in the bottom right and a latch to make the motor start and stop using the NOR gate, optical isolator and triac in the bottom.

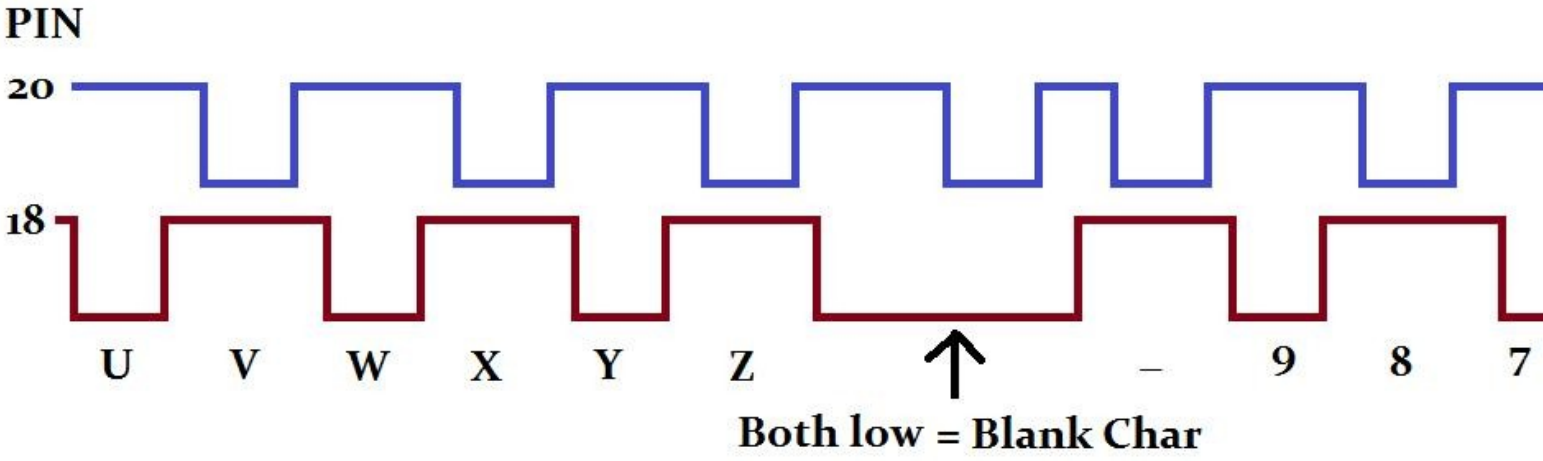
First problem was what power did this run on? CMOS can do up to 15V so I guessed it would be fine on anything from 5 to 15V, I thought 5V might be a bit low but had a handy 12V regulator so assumed this would be as good a voltage as any (it definitely wasn't going to damage the board).

My first purchase was a suitable transformer that would supply 36VAC, I chose a toroid which provided 2 x 18VAC 3A winding outputs, this was NZ$58 from RS components, with it (and again I apologise for the really bad paintbrush sketch!) I knocked up a power supply below. The only reason I show this was so you can see it was easy to extract both the AC and DC from the same transformer. The transformer was housed in an old ATX power supply case to keep the mains voltages safely at bay this worked quite well so I did the same on the final version



With power applied and CN15 or CN16 of the 20 pin connector held low the latch and optical shaft encoder are enabled (called E in the diagram below) To start the thing flapping we just need to set the latch by holding CN19 low (`M on` in the diagram below) , to stop flapping hold CN17 low (Moff).

When flapping the scope produces a square wave out of pin 1 of the comparator output (we will call Sync1 for ease of reference) and what seemed to be the opposite pulse out of pin 14 (Sync2) of another comparator. Both these signals and the three control signals above are brought out to the 20 pin connector which is good since we wont be using the 4063 in my version of the controller. The first thing we notice when we analyse the waveforms is that there is a point at the blank char position where both pulses go low, this is our index position and our first breakthrough in creating a new controller for the unit. There are twenty blue pulses and twenty red pulses each revolution (40 characters on our solari but see later some other solari boards only have numeric or text values)
We can use these pulses to count flips and so in theory should always know which character position we are at.

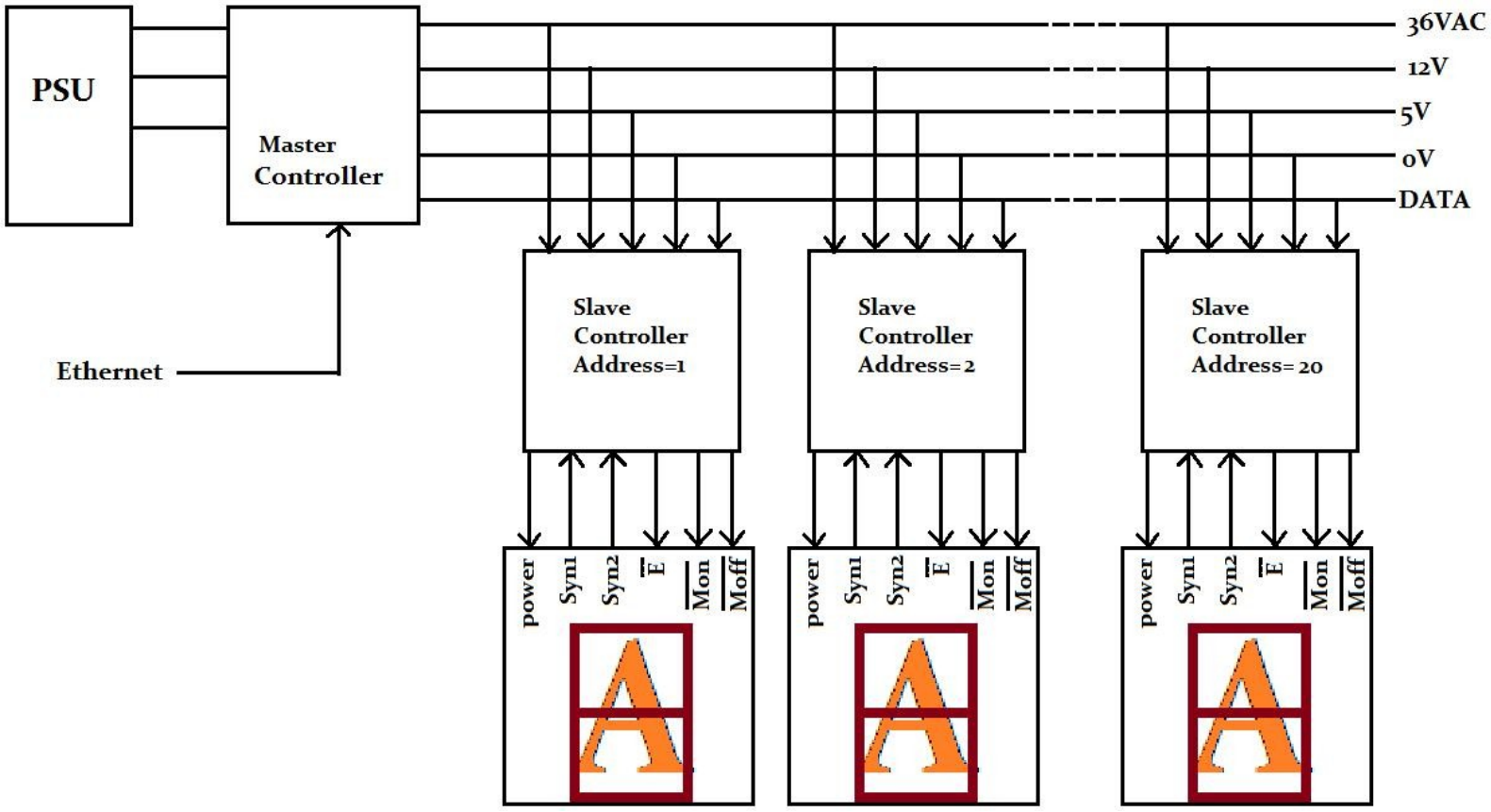


So enough theory, how do you wire the buggers up and control them?

Ignoring the 4063 comparator circuit, to control one unit you need to be able to read the two SYNC pulses above and pull low the correct control lines of the latch, In Indevin's (my customers) situation, they had bought 20 modules, making a total of 5 x 20 or a hundred I/O lines to control! If we decided to do it using a single board with twenty connectors the circuit would be horrendous! There is a much easier way.

How I decided to approach this was to use a very dumb slave control board for each solari module which can control the latch, provide power, be addressed individually and just count the flapping until we get to the correct character. A Master controller will address this slave and send it the character to display. To make this simple to wire it would be nice if they were on some sort of chained bus as below, as I needed

to order some 20 pin connectors anyway I decided to use a 20 wire bus, mainly so I could distribute the 36V, 12V and 0V power onto 4 wires each, this will allow most of the flappers to flap simultaneously without fear of drawing too much current down one wire. Using 20 way ribbon cable I estimate about 3A can be safely drawn which is about 20 solaris (ideal!) the 5V will use very current little as we will use PIC controllers which draw about 5mA each, still I used 2 wires for the 5V supply just to be safe.



The current figures drawn by each solari were very roughly measured with a cheapo multimeter (and then I added a bit to be safe). Obviously the bus could accommodate 100's of slave boards but we need to be careful we dont go over the limits of both the supply and the bus cable. There is a very elegant solution to this and that is to put a delay between writing each character (say 150ms) this means we send the character to slave 1 and start it flapping, wait 150mS then goto slave 2 and start it flapping, all the way to slave 20 a whole 3 seconds later, by that time it is very likely the earlier solaris have finished and can be disabled, this would limit the current on a larger system and also produces a nice ripple effect of the characters changing.

Like I said the slave controller can be dumb, we are counting 20mS pulses and setting a couple of lines, looking above it needs 6 I/O lines for DATA,SYNC1, SYNC2, Mon, Moff and Enable, I had 25+ spare PIC 18F2221 28 pin controllers so I used them (even though they were total overkill), but if I diddnt have any I could have got a nice $1.00 16 pin 16F device to do the same. The master controller needs to be a bit smarter as I need to connect to the network. There are millions of examples of PIC 18F using ENC28J60 Ethernet controllers and as I had both on my development board I decided to go this path, though an 8051 or AVR or even an ARM (if your a sadist), would work just as sweet.

Initially I was going to use I2C to control the slaves, but the only library I had for this was a blocking one and I would need to over-complicate the slave to do the responses where none was required (the master will fire off the data to the slaves and not expect a response) I then thought to use the UART functionality built into all the PICS and link all the receive lines up, this would work nicely as PICS have a large FAN OUT and FAN IN on there I/O pins. With this method you could probably get the system working quite easily by using just a PC serial port and not bother to use a controller at all.

As the system was well away from a PC and me being me I decided to implement a separate controller

board and my own much simpler protocol by bit-banging the data onto the wires as below, this way I could use any pin and simplify the PCB layouts. (I tend to over-think these things when I'm on a roll!)

Daughter Mk1 prototype board

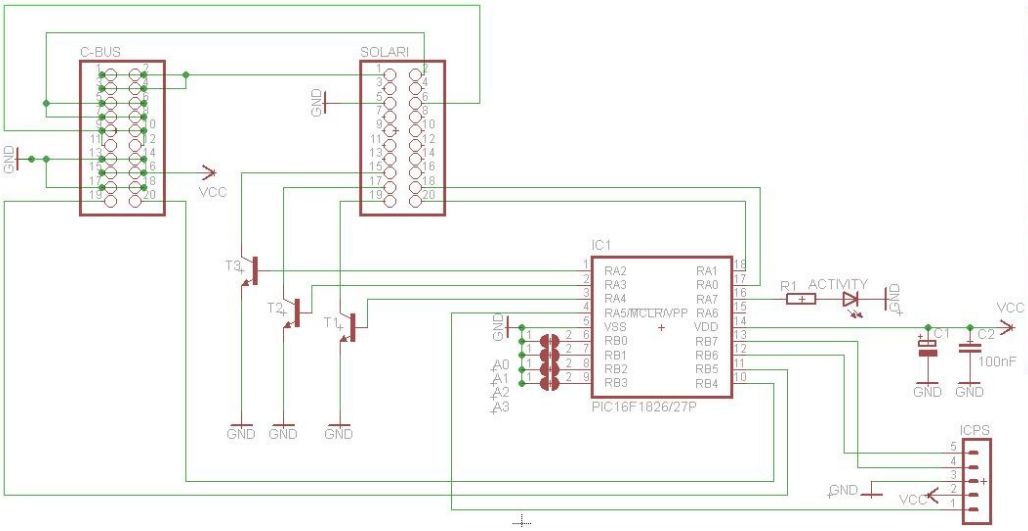
The mark 1 daughter board above is designed to plug directly into the 20 pin solari board socket, this holds the daughter board securely and allows us to expose our own 20 pin connector for the ribbon cable which I will make to be the BUS (simply crimping 20x connectors onto a ribbon cable - like the older IDE or floppy drive cables on a PC) , I used basically what I had in my parts bin and used the internal oscillator to clock the PIC at its default 4MHz. The MK1 circuit is similar to the MK2 version but using a few more components. BEWARE: While NPN transistors are shown in all my diagrams I actually used 2N7000 MOSFETS to pull down the Latch and enable lines, I used these as a) I diddnt have the 2N7000 MOSFET to hand in my eagle parts library, b) I have loads of them and c)they connect the same as the more usual NPN transistor but present a higher resistance for the PIC I/O pins not loading it as much as a transistor. As the two sync lines are open collector a pull up resistor is used, remember to do this or your SYNC lines will just look dead (caused me several frustrating hours this!)

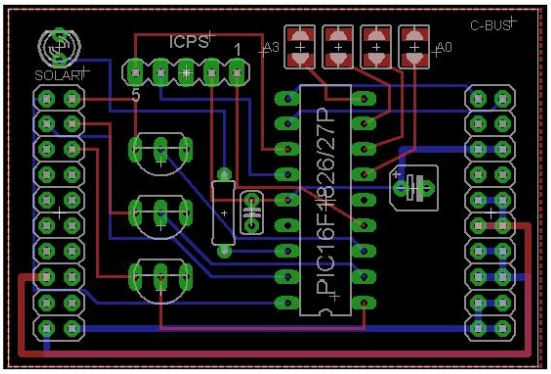
nothing really to go wrong here and it all works sweet! The only real issue is the tolerance of the mechanical units, some of ours were out by a character on some passes and had to be taken apart to be adjusted, the system self corrects itself as it goes past the index point so the tolerance of your code can be quite bad. My first attempt worked so I never changed the code much in the mk1 version despite seeing a few obvious improvements.

Daughter Mk2

The mark 2 design below is smaller and sweeter still removing the address switch and using internal pull-ups for the open collector Sync signals on RA1 and RA2. Again I used 2N7000 MOSFETS NOT NPN transistors. I priced it up at under $5 for the PCB and parts in qtys of 50 or more, a nice little earner possibly considering there are 480 more solari boards around NZ after the auction and how many more in

the hands of enthusiasts !





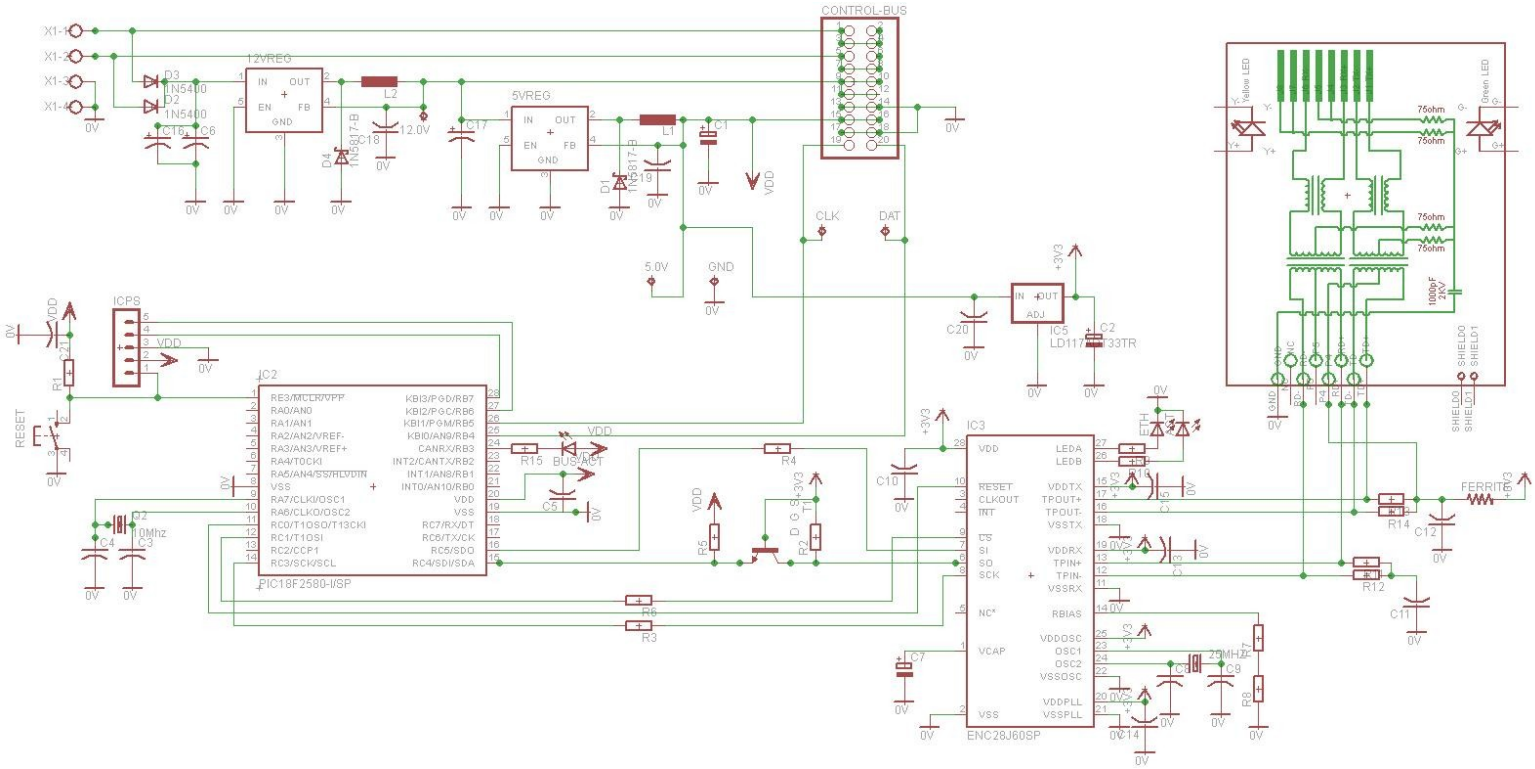
The PSUDO CODE for the slave board is below, Its really simple so I'll leave the challenge to write it to the reader

1. Read slave address (0..31 in this case)
2. RA2, RA3, RA4 = 0 (Transistors/MOSFETS off) - Stop flapping and disable solari
3. Wait for DATA Line activity
4. Read DATA word (16 Bits)
5. if bits 0-7 match the address then set the desired flap position 0..39 (using bits 8-15 of the DATA word)
6. RA2,RA4 = 1, RA3 = 0 (Enable and start motor)
7. if RA0 OR RA1 = 0 Increment flap position counter
8. if RA0 AND RA1 = 0 set Flap position counter = 0
9. if Flap position counter <> desired flap position goto step 7
10. RA2,RA3 = 1, RA4 = 0 (Enable and stop motor)
11. Wait 1mS
12. RA2,RA3,RA4 = 0 (All off again)
13. Goto 3

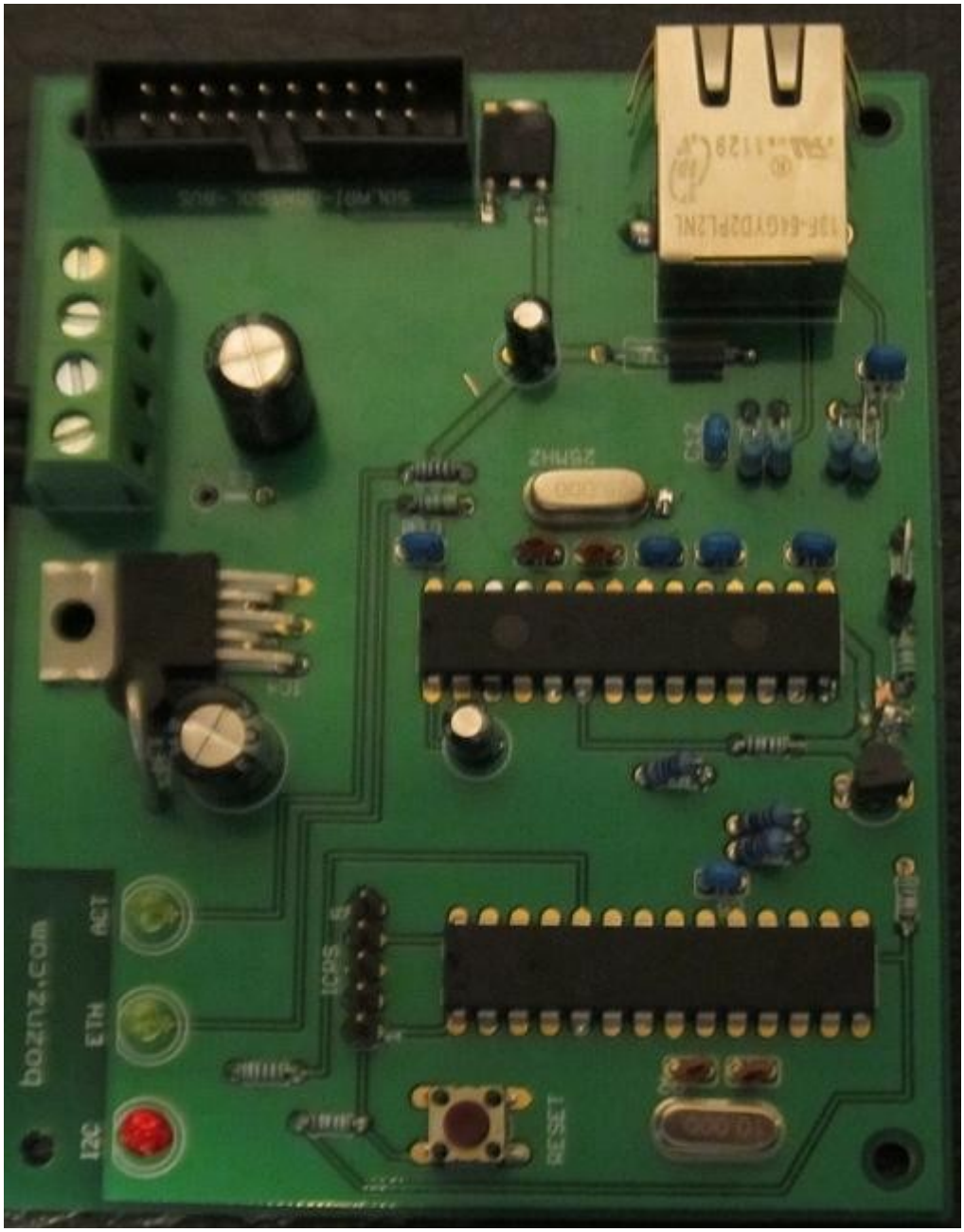
Master board

A bit more trickier here as we have ethernet to control, again both Microchip and Mikoe who developed my development board and compiler have a TCP stack to control the ENC28J60 chip and it is really well documented everywhere. Again we have a mark 1 and a mark2 board, the picture shows the MK1 prototype which has a 5V regulator on-board, the only difference between the mk1 and mk2 versions is

we have moved the 12V regulator on-board too. The regulators are LM2576 switching regulators and run quite cool without a heatsink, additional smothing and decoupling has been added on the MK2 version too as I noticed the MK1 version was reluctant to work without a stable supply of 4.6-5V. As the ENC28J60 is a 3.3V device we use a 3.3V regulator off the 5V supply



Firstly before anything else the NPN transistor is again a 2N7000 MOSFET (I really should add this to my eagle library, I'll get there eventually but I'm a bit new to eagle as my old CAD software - which I wont embarrass the company by naming - hasnt had a decent free version for years!), this does the 3V3 to 5V level conversion for the serial data out of the ENC28J60 device, the rest of the design is standard and on a million places on the Internet. As you can see we use all the 20 pins on the solari bus connector to distribute the power load on the cable. You **MUST** key this cable as if you get this or the slave cables the wrong way around.. KABOOM! 36VAC will definitely fry most of the chips!



The MK2 version is now in production and should be ready in 3-4 weeks, I doubt I will write anything more up here though, this is about as much as I can cope with and is more than enough for my reference!

The code for the master is basically read ethernet (I made a 200 character web page with an input component) convert ASCII to solari flap position (see table below) and output the characters one at a time to the appropriate slave on the data line, again I wont go into the code, as I'm still messing with it and will probably improve it for the mk2 controller.

ASCII to Flap Position conversion table (You will need to implemment this in whatever controller you use, note some solari displays have only numeric or even words)

ASCII Char	**Flap Position**
Space (and all unused)	0
_ (Underscore)	1

0	2
9	3
8	4
...	...
1	11
-	12
/	13
Z	14
Y	12
...	...
A	39

The prototype unit is currently with a carpenter who is creating a housing, hopefully a picture will follow once everything is finished

here is the video of 8 of the units working on the bench (apologies for the first word we spelt, we tend to be a little childish at work) and here are the eagle files. and here are a few more pictures and a video of the solari unit itself for anyone interested in trying to make one

Finally thanks to my customer Indevin who paid for the 3 days of nostalgia to do this little project and for all the parts :-)

Note you can find loads more information from Tom at http://unknowndomain.co.uk/blog/category/split-flap-display/ who got in touch with me after the post on hackaday.com and is in the process of making one from scratch (google wasn't my friend when I originally searched for all this!)

Home (boznz.com)