CNC Spot Welder Project.

I'm Simon Daniels, Director of S&T Controls Ltd, a small company whose core business is making one-off jigs for industry. My background is in analogue electronics together with some mechanical and electrical experience. I've never been a fan of coding in the traditional sense and being dyslexic doesn't help.

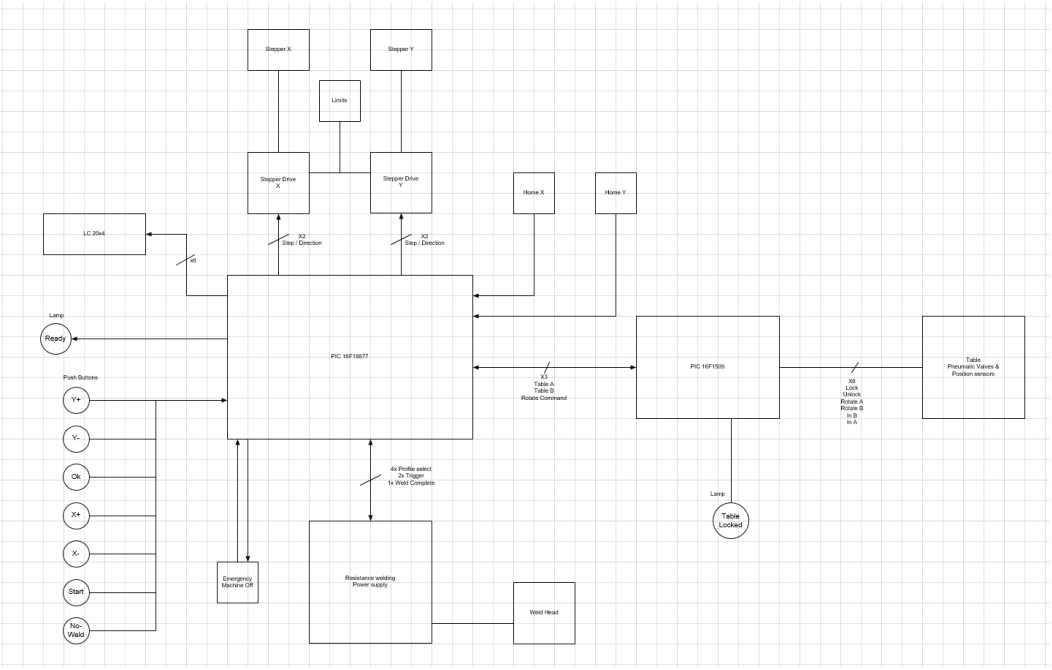
Around fifteen years ago a Doctor of Electronics recommended that I try Flowcode, so I purchased my first licence, which I believe was Version 3! Since that purchase I've designed and completed multiple projects, from a simple little box with a couple of lamps to relatively complex controllers.

In March 2020 the country went into the first Covid 19 lockdown. Huge demands were placed on both the NHS and the private medical sector. One of my customers, who makes sensors used in the medical world, needed to double their production capacity within a couple of months.

One of the machines on their production line is a precision CNC (Computer Numeric Control) spot welding machine. The customer also had another spot welding machine, which was stored in their warehouse as it no longer had any CNC equipment, which had failed some time earlier. I'd had a similar project to upgrade some other equipment for them five years before, and for that I'd used a PC running Mach3 CNC together with some added Visual Basic code. This time, however, the specification stated that there were to be no computers. It was, however, agreed that I could use a PIC.

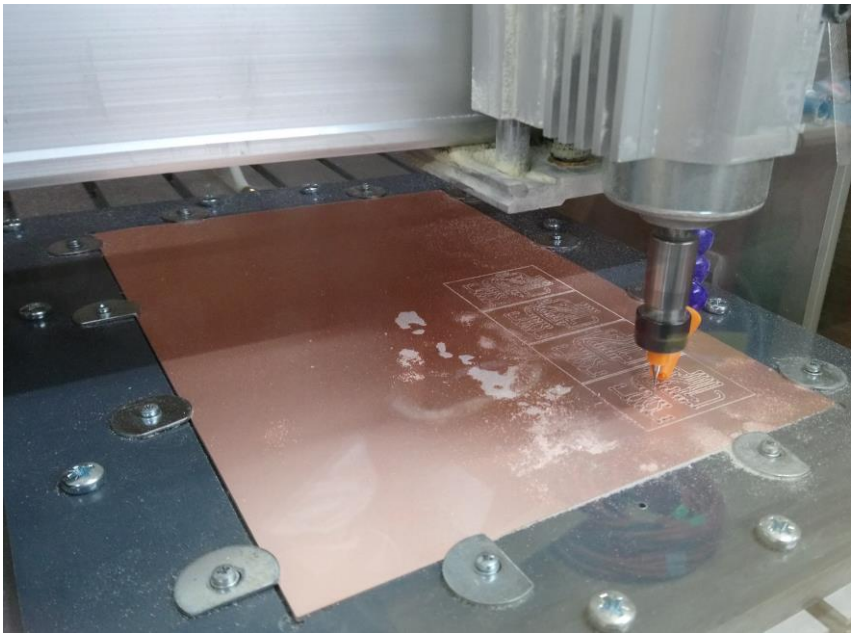
The non-CNC spot welding machine had two different fixtures on a rotary table and used a small X-Y table with stepper motors to position six spot welds of varying profiles for each fixture. A part is loaded into the machine and welded while the operator adds additional parts to the second fixture. A button is pressed, the table turns through 180º and welds the second set of parts, while the operator loads the first set again, and so the cycle continues.

I began searching for a forty pin, through hole PIC, with the largest program memory I could find and decided to use the PIC 16F18877. I drew a block diagram and decided I would need two microcontrollers, due to a lack of I/O. The second, smaller one, controls the rotary table.

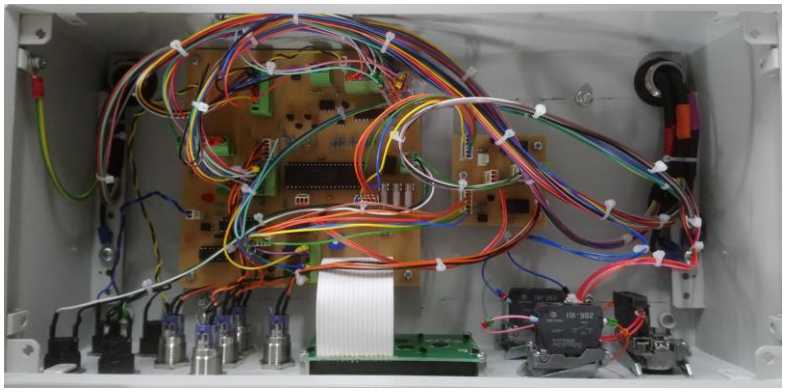



Next, I set about designing and making a PCB. I have a small CNC engraving machine set up with a vacuum fixture and, with a bit of software manipulation, I can create G code that mills/engraves islands for tracks and then drills the holes. This machine has some shortcomings, single sided and through holes are the limit, but

it's great for fast one-off boards and for proving a design before I bear the cost of having multiple boards made professionally.



The component list included a handful of optical isolators, some relays and as many decoupling capacitors as I could fit on the board to suppress a noisy resistance welder with pneumatic solenoids everywhere. Once I had a PCB a small amount of 'on the bench' testing, probing the PIC socket pins to make sure the PCB did what it should, was required. The PCB was then put in a box together with an LCD, assorted buttons and stringent measures were taken to keep the noise out of this area.

A second control panel, pre-existing on the machine, was modified to house the 24V power supply, contactors, stepper drives and such.

My next task was the PIC software. I decided to break into it gently and got the control table working first, checking it all operated correctly, unlocking the table, rotating it, and then locking and sending the confirmation back as to which part was locked into the welder jaws.

I then set about the main software. I have no idea how much longer this would have taken to write in C++, but it was more than two weeks' worth of work using Flowcode (including late days and weekends bearing in mind the urgency). Twelve macros later, and the main loop, I had code that was showing signs of working.

Whilst creating the flow chart I realised I'd missed out an output and I needed the PIC to be able to trip the EMO (Emergency Machine Off) circuit. This was because after setting up, and driving the machine around in manual mode, I decided it would be a good idea to restart the machine completely and re-datum.

I'd managed to leave the ISCP lines free for programming, so I made a little sub PCB with a relay and buffer transistor. This now meant that, if the sub PCB was not plugged into the programming port, the EMO tripped off which was a nice, if inadvertent, safety win.

The machine can take two different products so needed to be able to handle six coordinates, plus their weld profiles, for four different fixtures. This was accomplished by storing five bytes to the EEPROM for each move, the X axis before and after the decimal place, the same with the Y axis and the weld profile. I could then use a formula to point the program to read the correct bytes from the EEPROM at each move.

I did the same with the setup menu, writing the move coordinates to the address locations in the EEPROM. The program knows how many pulses are required to move the X-Y table 1mm and can then calculate the distance from one move to the next and then how many pulses to output and in which direction.

I played around with pulse timings and picked a delay that meant I didn't need an acceleration/deceleration ramp, safely. I didn't want any risk of losing step pulses and creating a distance error. My customer confirmed that the welder was still faster than the operator's loading ability to load the next part, so this was plenty good enough.

I secured the services of a furloughed friend to come and test my code in the machine, programming coordinates, etc. checking every aspect we could conceive, picking out the bugs and there were a few! Following some bug-fixes and further testing the result was a successful project which proved to reliably hold a 10um position repeatability. After some conformity paperwork and EMC testing (I have a pre-compliance grade setup in my workshop) the machine was delivered back to my customer where it went straight into production.

I have since written a software update to add a third product. One bug was reported, in one of the setup menus. It turned out I had forgotten to flash the LCD with a "parameters saved" message for a few seconds before reverting to normal display, although they had saved. I fixed this in the update with a "don't worry, it's saved" message appearing on the LCD, luckily to the amusement of the customer's staff.


No further bugs have been reported! 🙄