NXP 3725 Abstract

This project was the development of a controller for a three-axis milling machine for the specific purpose of cutting panels for electronic equipment. Modern electronic equipment often requires front panels with large cut-outs for LCD's, for meters and, in general, openings more complicated than can be made with a drill. It is tedious to do this by hand and difficult to achieve a nice finished appearance. This controller allows it to be done simply, quickly and to be replicated exactly.

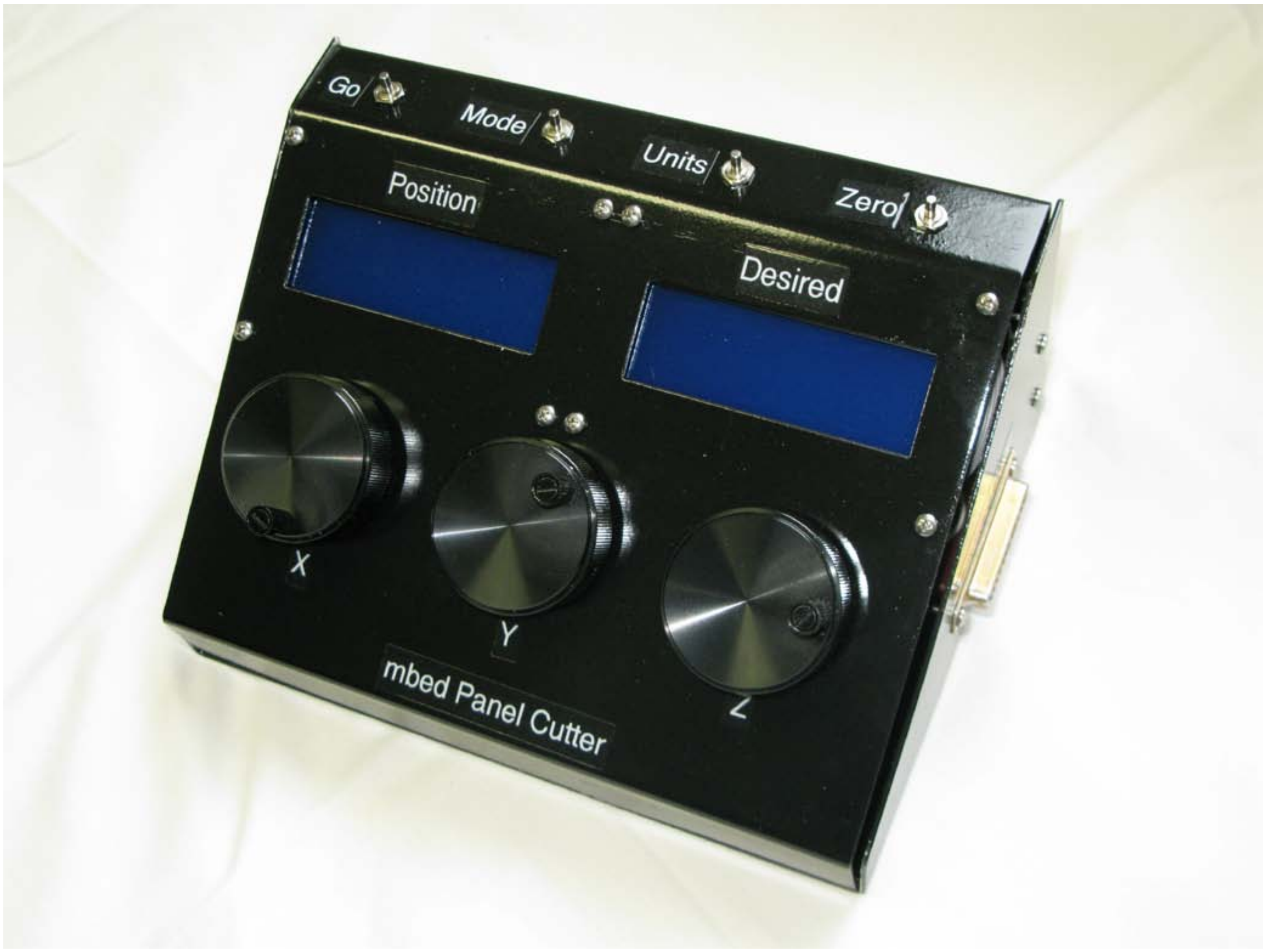

Figure 1 Photo of the completed controller. Its finished panel was cut using the system.

The controller is shown in Figure 1. It controls three stepping motors, one for each of the three axes of the milling machine. Inside the controller is a power supply and one printed-circuit board which carries the NXP mbed module plus necessary interface circuitry and a socket for an SD card which contains the instruction 'script' needed to cut any particular panel. In use, a piece of material for the panel is clamped onto the milling machine table and the cutting tool is moved to a starting position using the rotary encoders. Then the controller is switched to its 'automatic' mode and a script on the SD card is then followed to cut the panel. A very simple 'language' is used for the script; to go to any particular (x, y) position, to lift the cutting tool, to lower the cutting tool, to cut a rectangle of any dimension and to cut a circle of any dimension, etc. More complex instructions sequences such as those needed to cut the rectangular opening plus four mounting holes for a LCD are just combinations, called macros, of those simple instructions; every new device (meter mounting holes, LCD mounts, etc.) will have its own macro. The complete script for a particular panel can be any combination of simple commands plus macros.

The milling machine, a Taig ‘micro mill’, with stepping motors is shown in Figure 2. In its ‘manual’ mode, the system can be used as a conventional three axis mill controlled via the rotary encoders. The absolute position of the cutting tool is displayed in units of either inches, mm or thousandths of an inch.

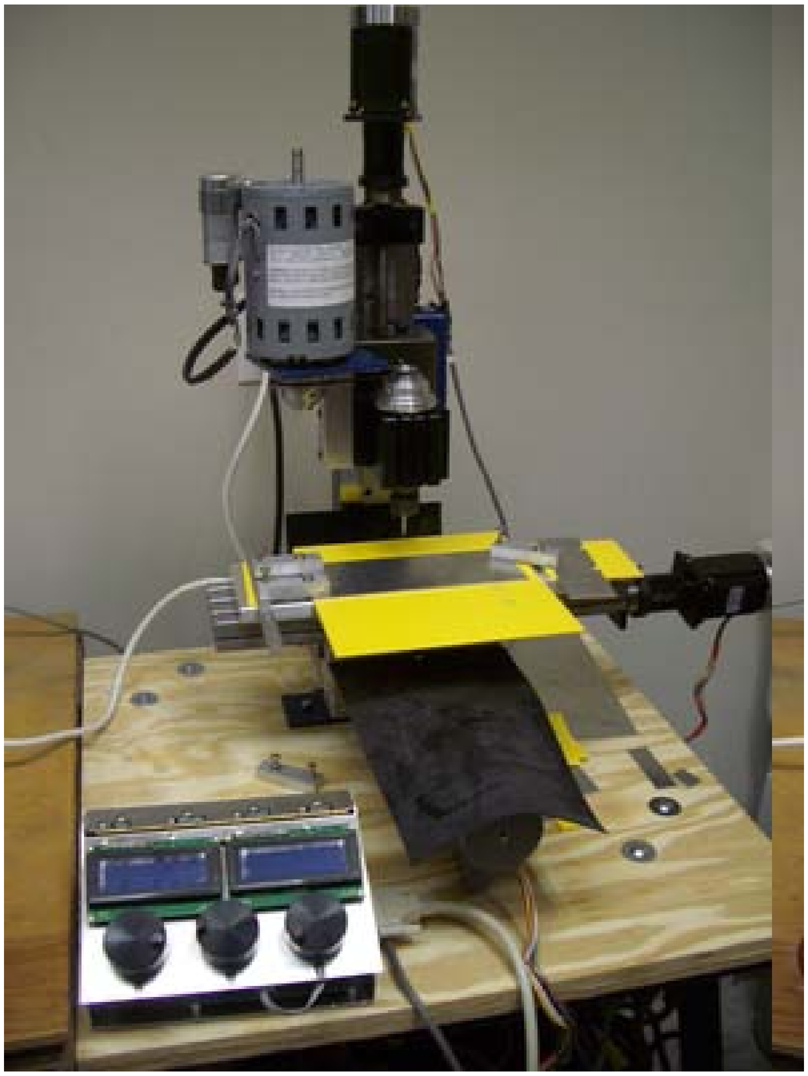

Figure 2 Milling machine with first prototype of the controller

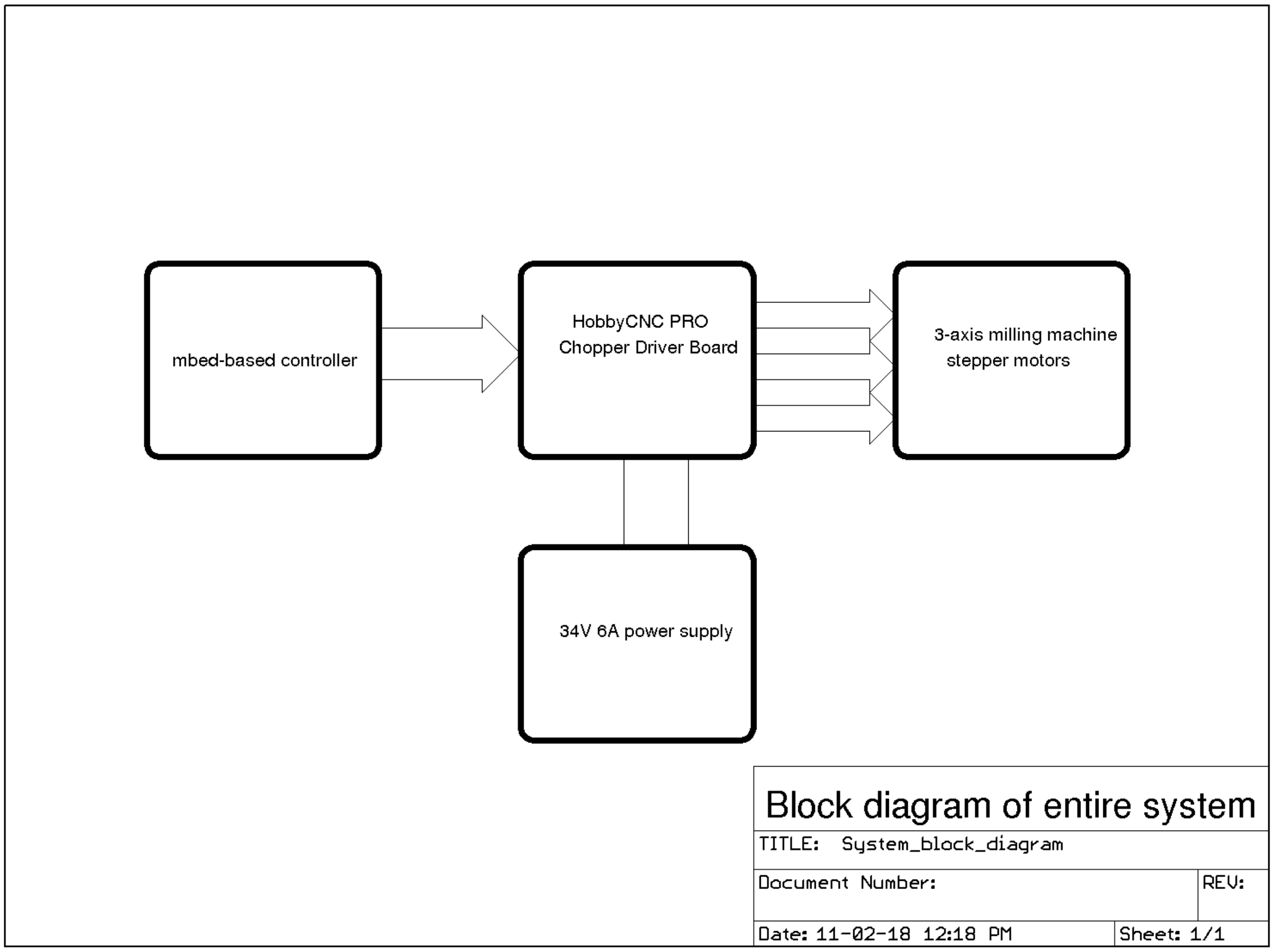

Figure 3 Block diagram of the whole system

The block diagram of the whole system is shown in Figure 3. The original manual milling machine was modified by the replacement of the lead screw handles by stepping motors. The driver for these was a HobbyCNC Pro driver board. The interface to the HobbyCNC board from the controller is via a 25-conductor cable with standard Cannon D connectors. The HobbyCNC board and stepper motors require a powerful DC source which was supplied by a Sorensen power supply able to provide 34 VDC at 6 Amperes.

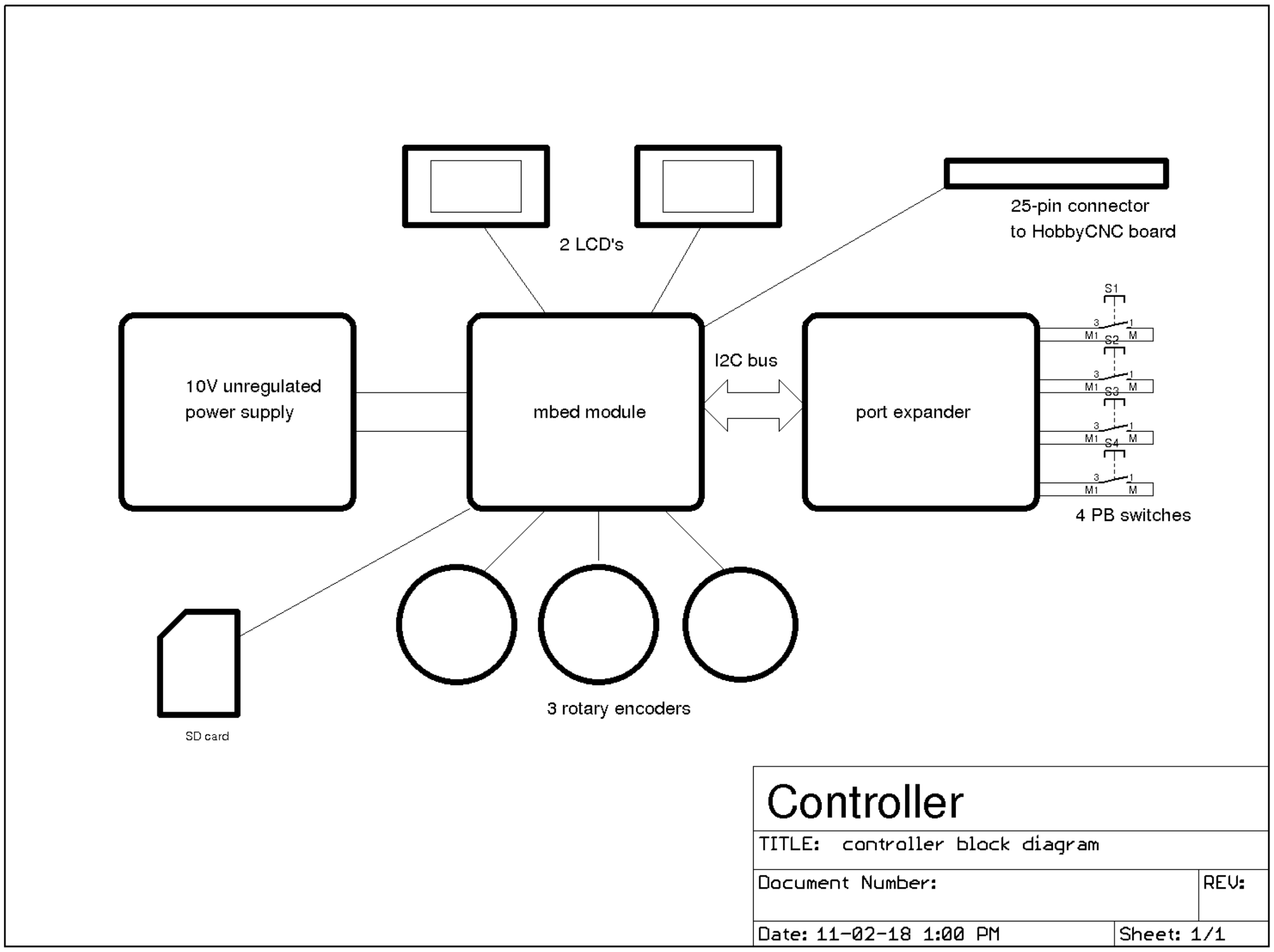

Figure 4 Block diagram of controller

The block diagram of the controller itself is shown as Figure 4. There were not enough general-purpose lines on the mbed module to provide all those needed so a port expander was used to connect to the four push-button switches. The mbed module was mounted on a custom printed-circuit board (designed using Eagle) which had connectors on it for the LCD's, the switches, the rotary encoders and an SD card socket.

The schematic diagram of the printed-circuit board holding the mbed module is shown in Figures 5a, 5b and 5c below. There is nothing unconventional in this circuit.

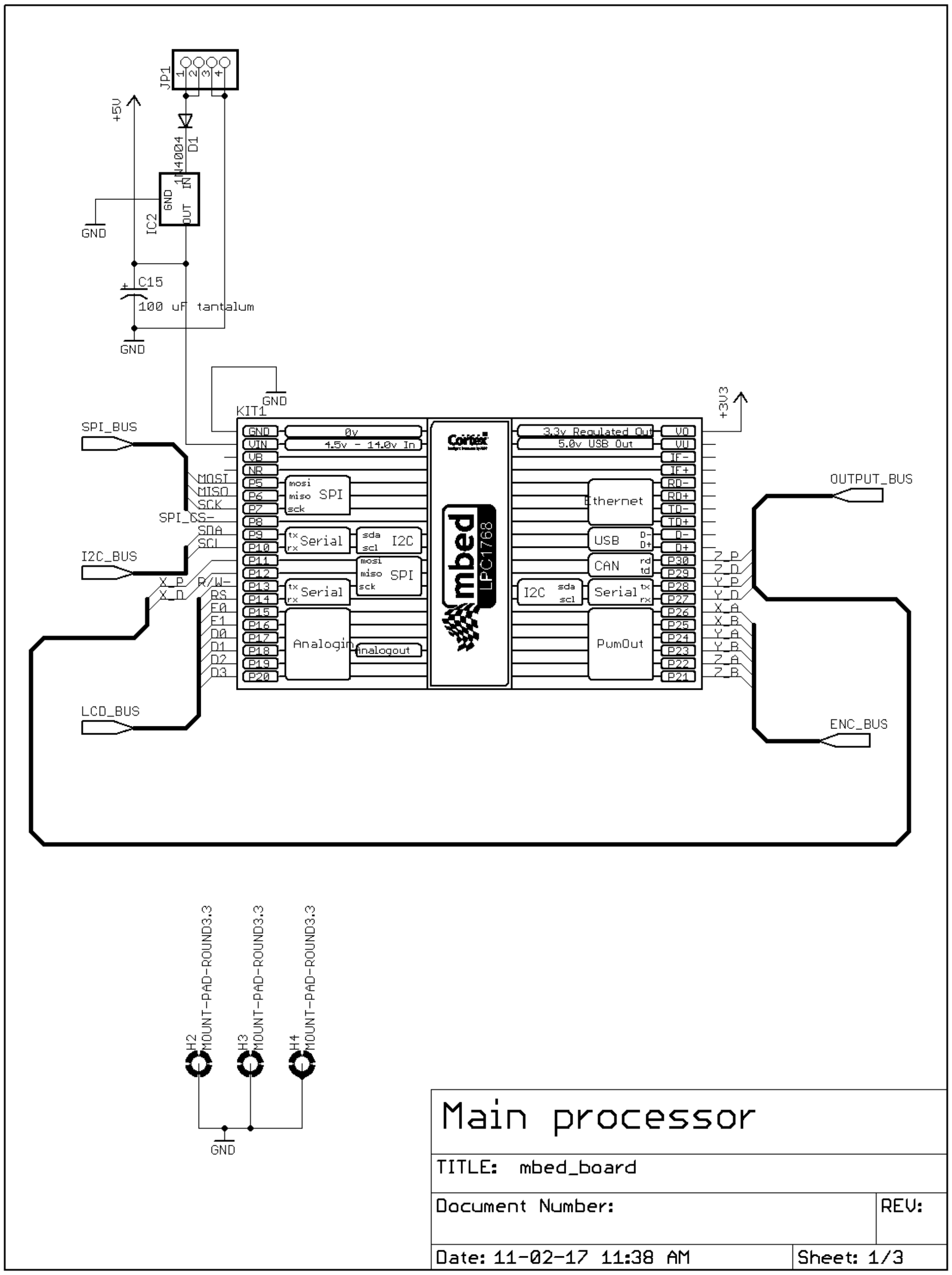

Figure 5a mbed module and interconnections

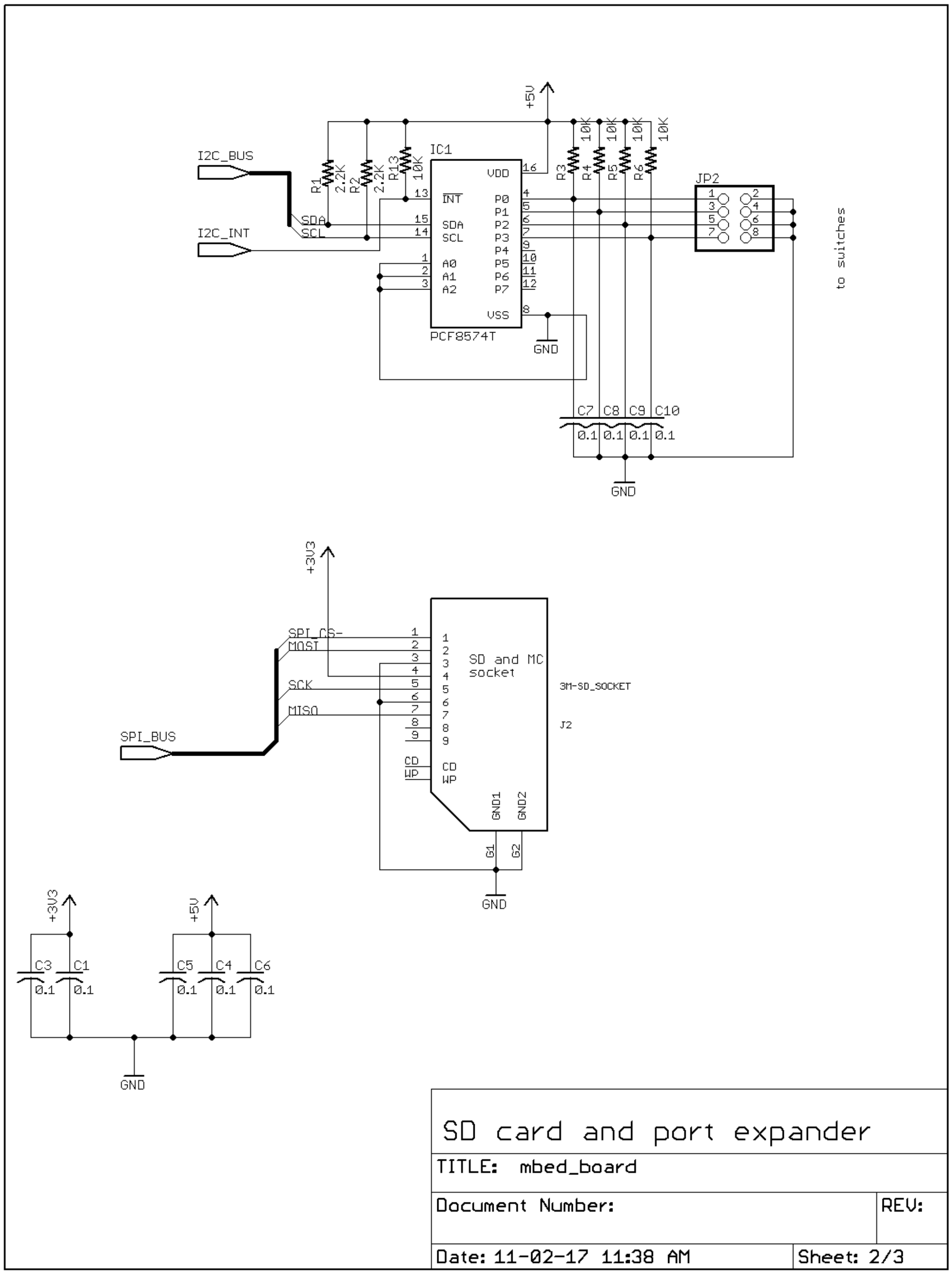

Figure 5b SD card and port expander

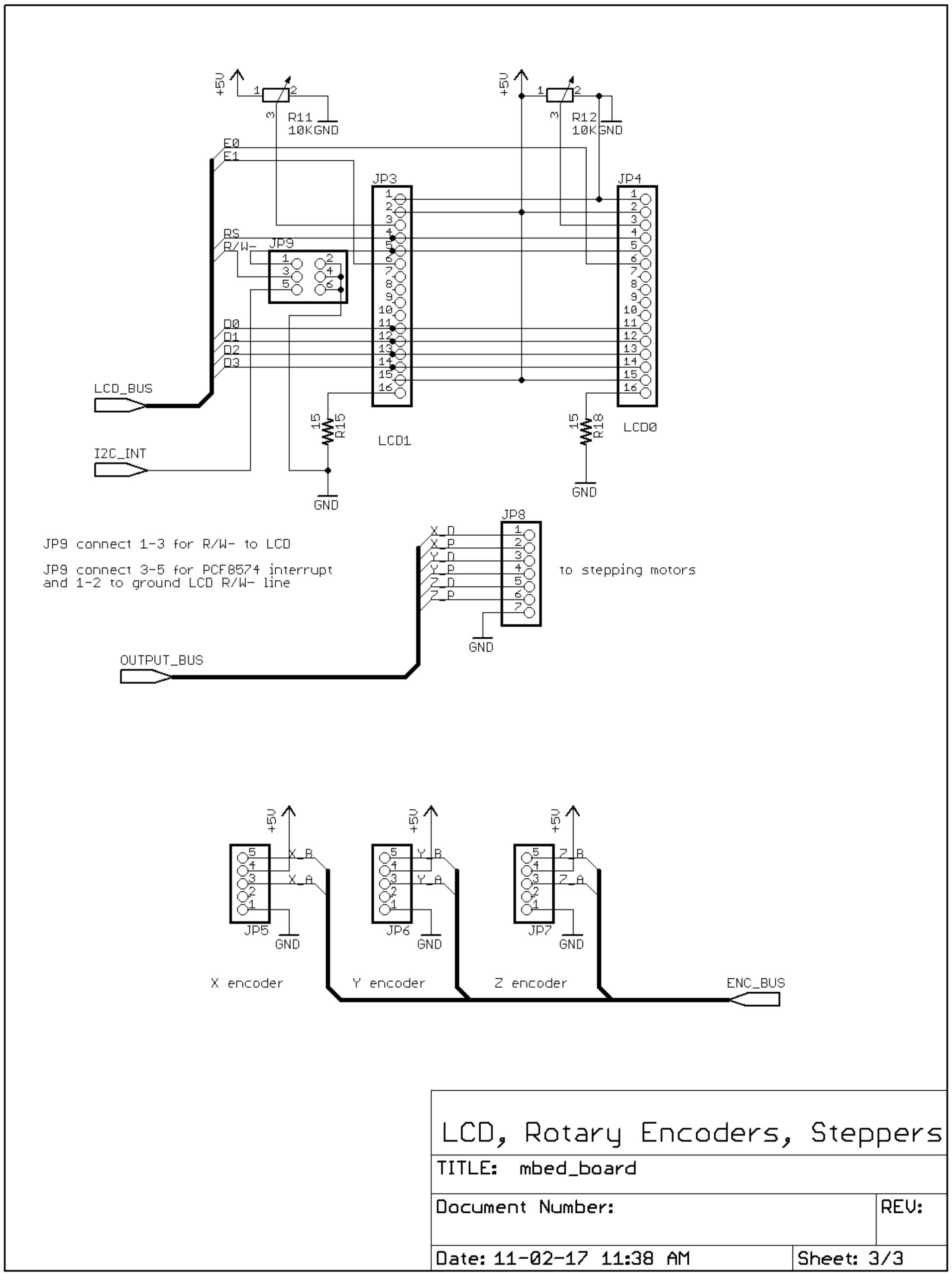

Figure 5c Encoder, output and LCD connections

Sample piece of code

This is the procedure which cuts a rectangle in the material. Internally, all the dimensional program variables, such as the passed variables, are in units of steps which are integers. For this compiler, an integer is a 32-bit variable. The variables, 'now' and 'desired' are global variables and are structures with three integer components; x, y and z. 'now' is the present position of the tool and 'desired' is the desired position; these are used by the basic go_to_xy() procedure.

```
void rectangle( int delta_x, int delta_y) {
//
// cuts a rectangle of dimensions (delta_x, delta_y) starting from the lower left corner
// it is assumed the tool is already up; if it isn't, it does nothing.
// delta_x and delta_y are in steps
//
   lcd2.cls();

   if (z_state == up) {

      desired.z = now.z;
      desired.x = now.x + tool_radius;    // tool is at lower left corner
      desired.y = now.y + tool_radius;
      go_to_xy();

      down_tool();

      desired = now;
      desired.y = (now.y + delta_y) - (2 * tool_radius);
      go_to_xy();

      desired.y = now.y;
      desired.x = (now.x + delta_x) - (2 * tool_radius);
      go_to_xy();

      desired.x = now.x;
      desired.y = (now.y - delta_y) +  (2 * tool_radius);
      go_to_xy();

      desired.y = now.y;
      desired.x = (now.x - delta_x) + (2 * tool_radius);
      go_to_xy();

      lift_tool();
   }

   display_string(desired_pos, 1, 0, "Done RECT");
}
```