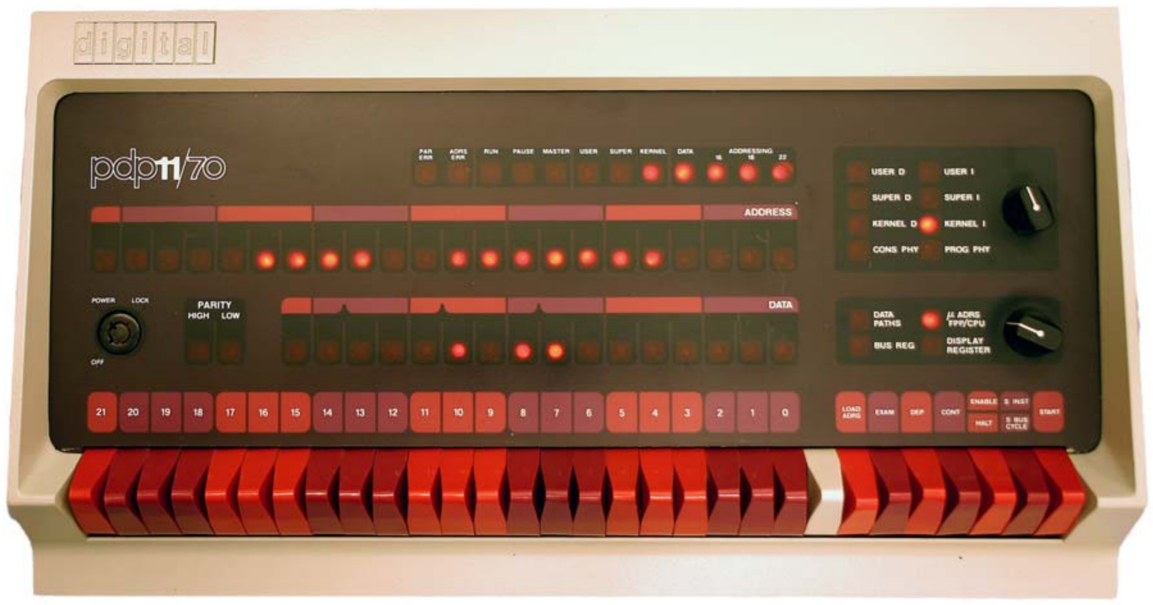
Abstract
Re-animating the PDP-11/70

©2004 John W. Peterson

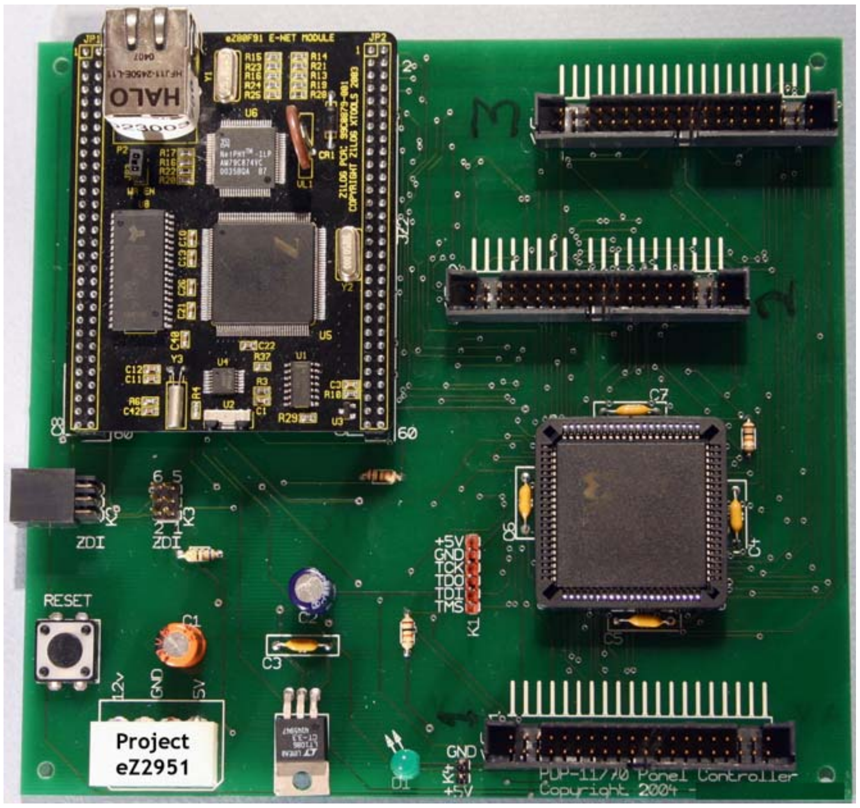
When computers first appeared, one of their most distinctive things about them was the complex array of blinking lights and switches. For many years the rows of mysterious blinking lights were the very stereotype of what a computer was.



PDP-11/70 Front Panel

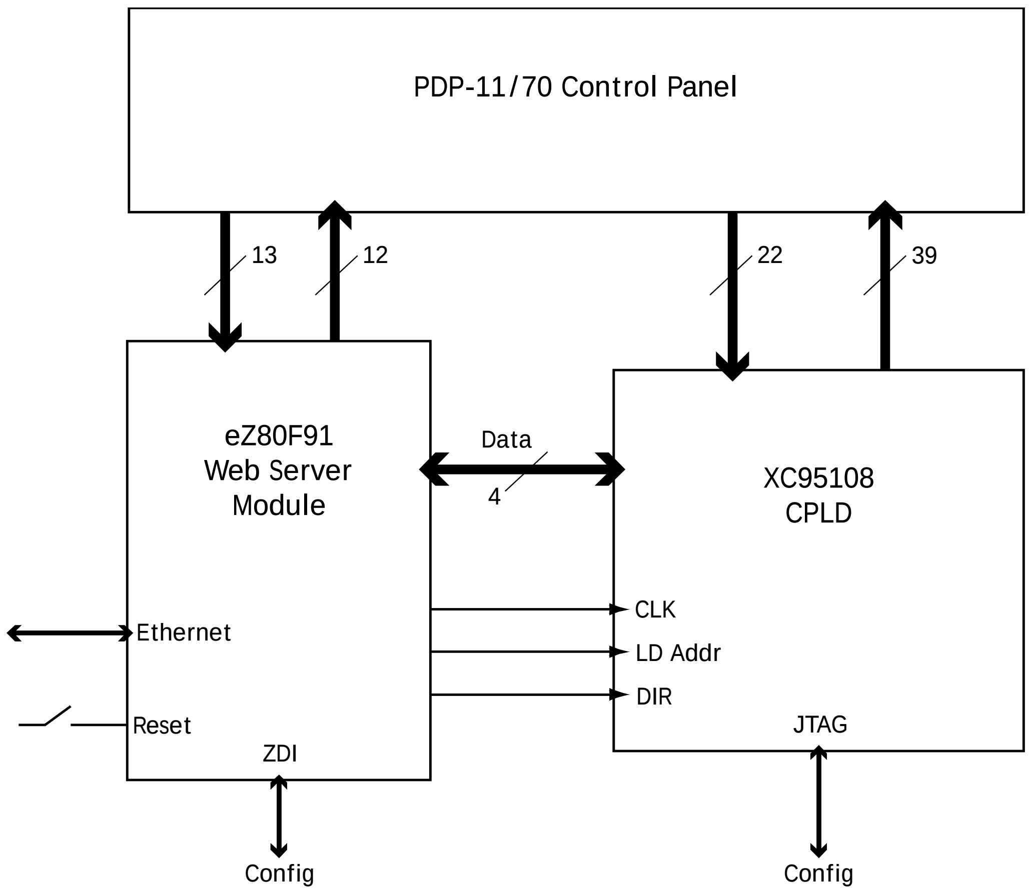
I've always missed the visual and tactile aspect of computers that's been lost since the advent of microprocessors. When a console for a PDP-11/70 came up for sale on Ebay, I couldn't resist the opportunity to bring back to life the magic of those switches and lights.

This project uses the eZ80F91 as a network read controller to re-animate the PDP-11/70 panel. Every switch and light is available to the hardware and under software control. The eZ80F91 module is a perfect fit for this application.



Panel Controller Hardware

While the eZ80F91 has a generous helping of parallel I/O, it clearly needs some more for this application. A Xilinx XC95108 CPLD is used to provide the additional parallel I/O. It's very flexible and provides 69 I/O lines in a single package. Sixty-one of these lines interface to the panel and seven to interface to the eZ80F91. Another nice feature of the XC95108 is it interfaces directly to both the 3.3v logic of the eZ80F91 and the 5v logic of the panel.



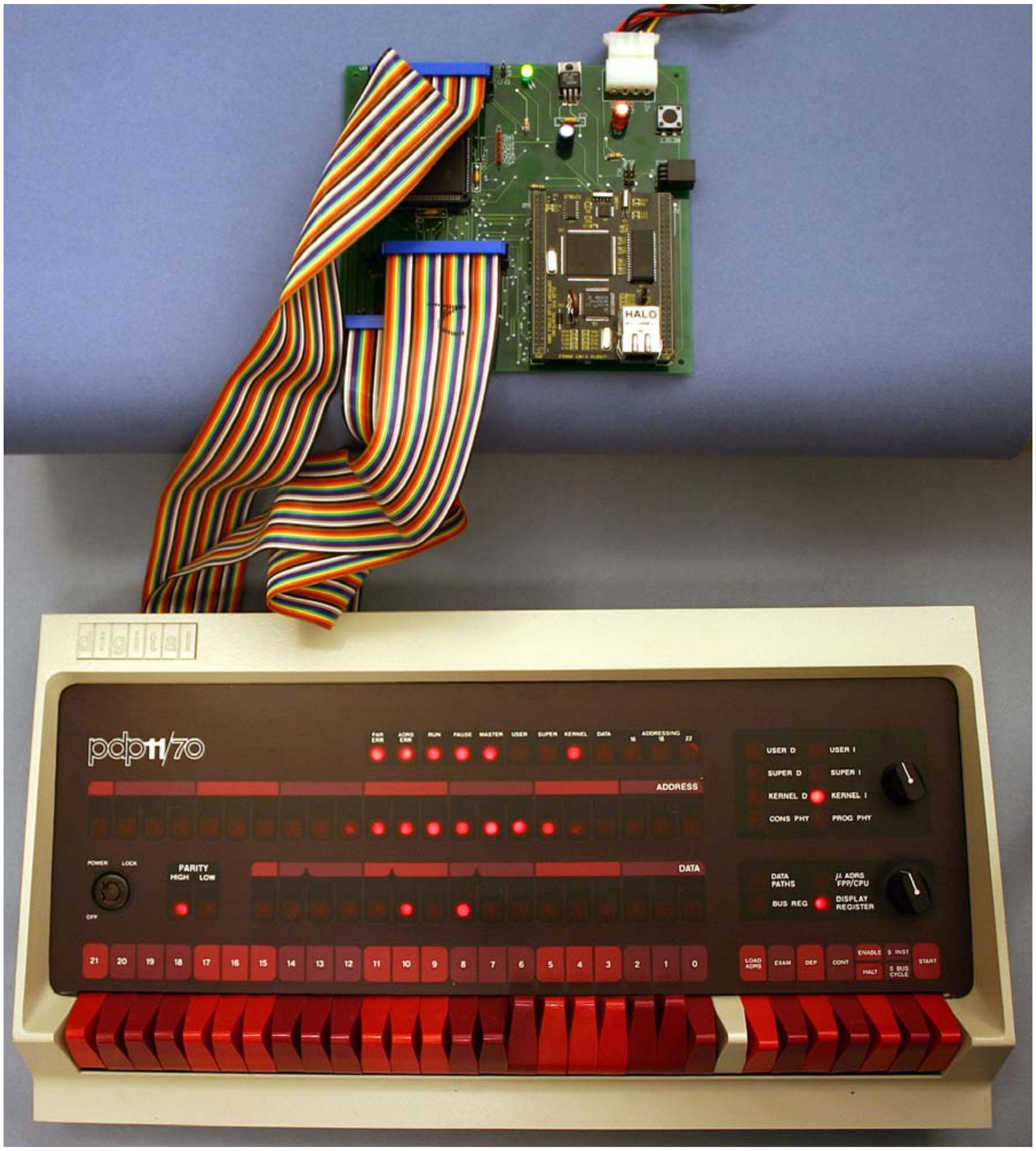
System Block Diagram

The eZ80F91 takes on the rest of the panel interface, driving 9 status lines (via 10K pull-up resisters) and handling 13 inputs. The inputs from the panel (driven by TTL gates) are connected directly to the eZ80F91, since it is 5v tolerant. To drive the TTL gate inputs, the eZ80F91 GPIO ports are configured in "open drain mode" and pulled up to +5v via 10KΩ resister networks. The remaining seven I/O lines are used to interface to the CPLD.

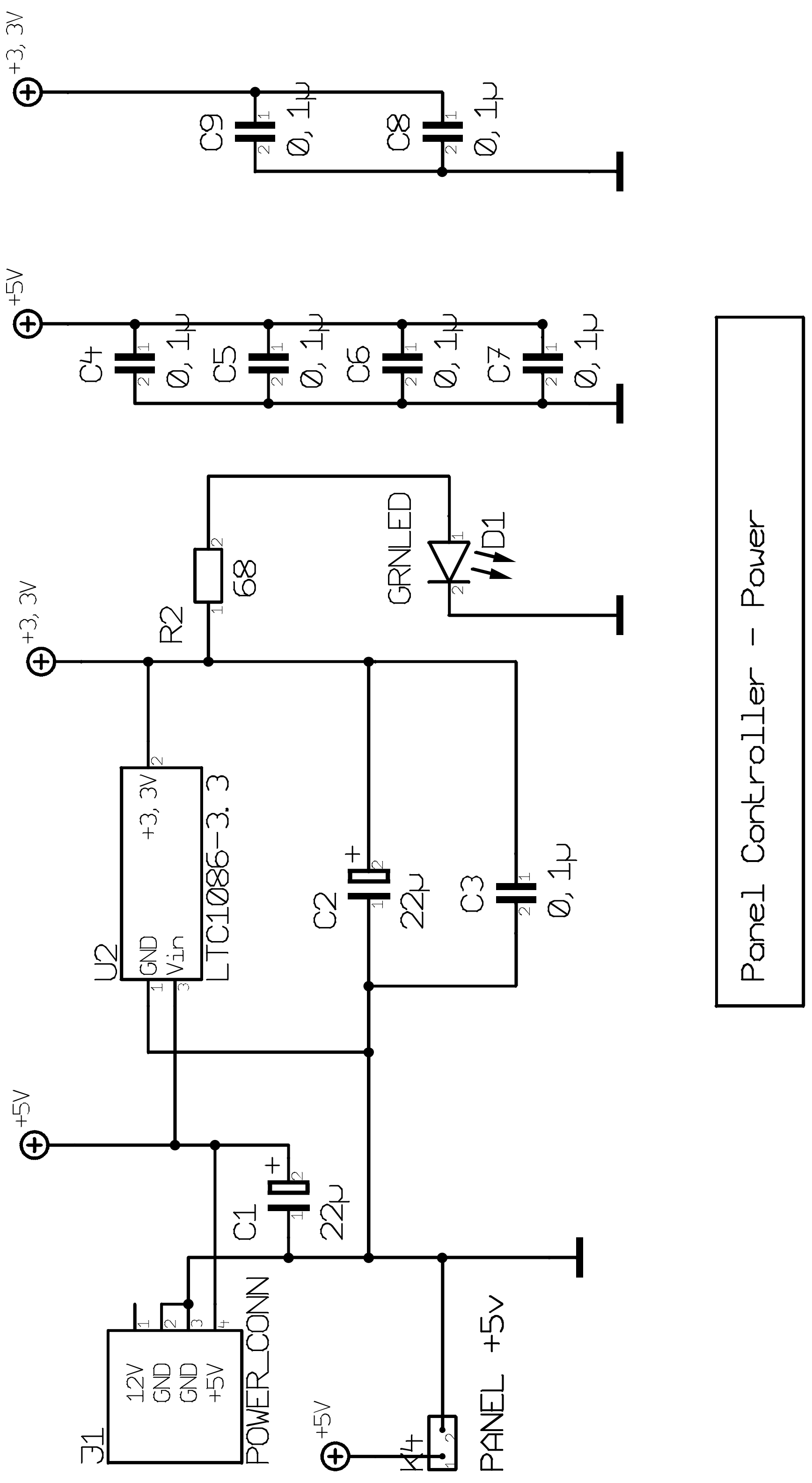
The I/O lines from the eZ80F91 and the CPLD are connected to the panel using the same ribbon connectors the panel originally used to connect to the PDP-11/70 CPU. These cables also supply some of the +5V power to the panel. The system is supplied with +5v from an external power supply. An LTC1086-3.3v regulator provides power for the eZ80F91 module. The only other components required are bypass capacitors, a reset switch, and connectors for programming the eZ80F91 and the CPLD.

Currently the software performs the basic, low-level access to the Panel. A library of routines (Panel_IO.c) interfaces with the GPIO ports to access all of the lights and switches. These are grouped by their placement on the panel, for example the switch register, the address and data lights, the status lights and the various control switches.

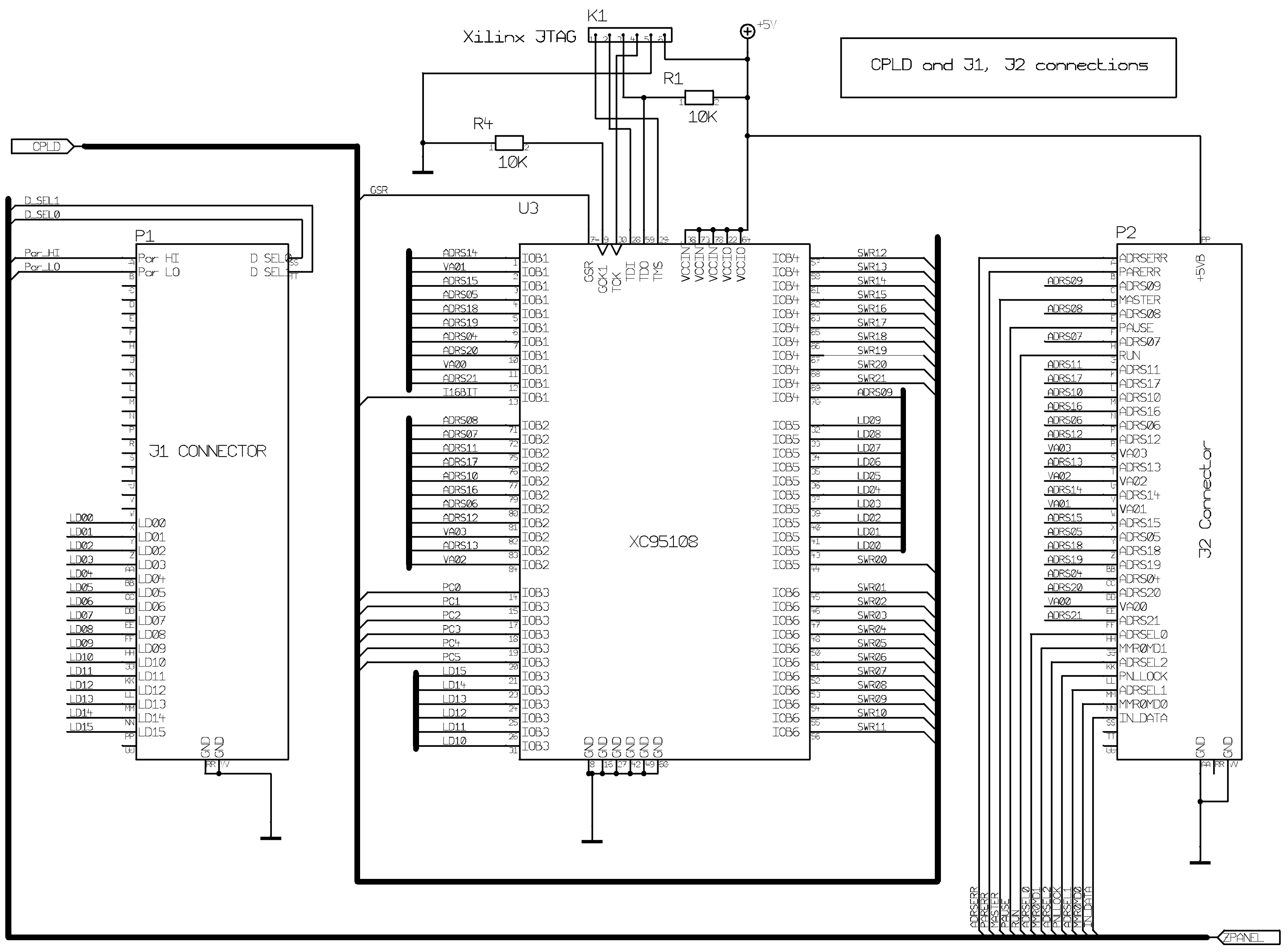
The next phase of the software (currently under development) is to make the switch and light information available over the Ethernet. The Panel controller will listen for connections on a specified port, and report the switch settings and accept bit patterns for the light display. Higher-level operations (such as installing a particular animation pattern) are the next step. A simple sequence of bit patterns for display can be downloaded to the panel controller for playback.

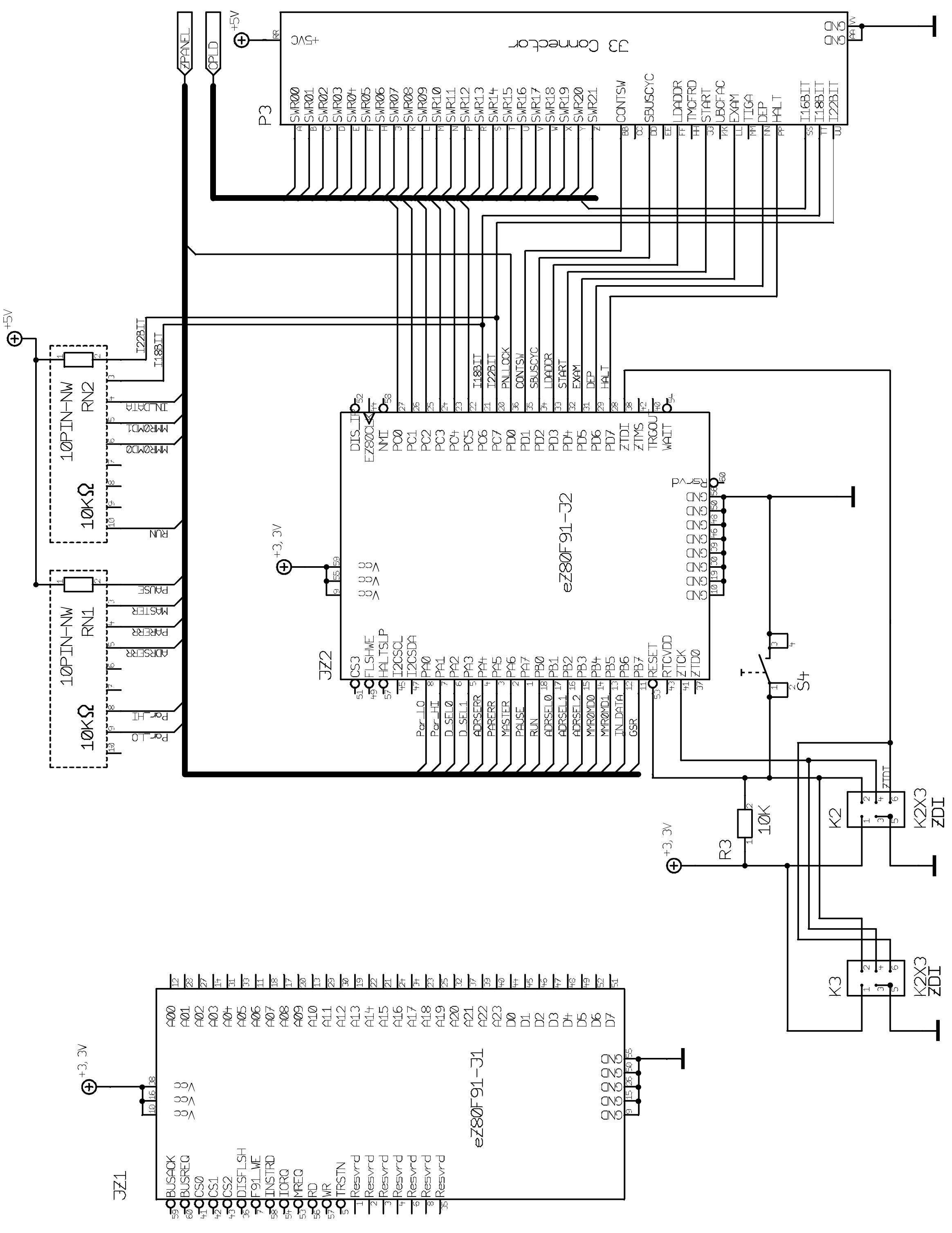


Schematics



Panel Controller - Power
+3,3V
C9
0,1µ
C8
0,1µ
+5V
C4
C5
C6
C7
R2
68
GRNLED
D1
U2
LTC1086-3.3
GND
Vin
C2
22µ
C3
C1
J1
POWER_CONN
12V
GND
GND
K4
PANEL +5v

CPLD and J1, J2 connections
Xilinx JTAG
K1
R1
10K
R4
10K
+5V
U3
XC95108
P1
J1 CONNECTOR
P2
J2 Connector
CPLD
ZPANEL
GSR
D_SEL1
D_SEL0
Par_HI
Par_LO


J3 Connector
P3
+5VC
+5V
ZPANEL
CPLD
SWR00
SWR01
SWR02
SWR03
SWR04
SWR05
SWR06
SWR07
SWR08
SWR09
SWR10
SWR11
SWR12
SWR13
SWR14
SWR15
SWR16
SWR17
SWR18
SWR19
SWR20
SWR21
CONTSW
SBUSCYC
LDADDR
TMCFRD
START
UBCFAC
EXAM
TIGA
DEP
HALT
I16BIT
I18BIT
I22BIT
GND
RN1
RN2
10PIN-NW
10KΩ
I22BIT
I18BIT
IN_DATA
MMR0MD1
MMR0MD0
RUN
PAUSE
MASTER
PARERR
ADRSERR
Par_HI
Par_LO
JZ2
eZ80F91-J2
+3.3V
Vcc
DIS_IR
EZ80CLK
NMI
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
ZTDI
ZTMS
TRGOUT
WAIT
I18BIT
I22BIT
PN_LOCK
CONTSW
SBUSCYC
LDADDR
START
EXAM
DEP
HALT
Pwr_V
CS3
FLSHWE
HALTSLP
I2CSCL
I2CSDA
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
RESET
RTCVDD
ZTCK
ZTDO
Par_LO
Par_HI
D_SEL0
D_SEL1
ADRSERR
PARERR
MASTER
PAUSE
RUN
ADRSEL0
ADRSEL1
ADRSEL2
MMR0MD0
MMR0MD1
IN_DATA
GSR
S4
R3
10K
K2
K3
K2X3
ZDI
ZTDI
JZ1
eZ80F91-J1
BUSACK
BUSREQ
CS0
CS1
CS2
DISFLSH
F91_WE
INSTRD
IORQ
MREQ
RD
WR
TRSTN
Resvrd
A00
A01
A02
A03
A04
A05
A06
A07
A08
A09
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
D0
D1
D2
D3
D4
D5
D6
D7

Code Sample

```
// Project eZ2951 - Panel Controller for the PDP-11/70

#include <ez80.h>
#include <stdio.h>

#include "CPLD.h"
#include "Panel_IO.h"
#include "Lights.h"
#include "Timer.h"

extern void _init_default_vectors( void );

// This simple demo shows how to access the Panel I/O
// library.

void main( void )
{
      uint16 controls;
      uint8 speedNum;
      uint8 speeds[4] = { 100, 75, 50, 25 };

#ifndef ZSL_DEVINIT
      _init_default_vectors();
#endif

      InitTimer( 1 );   // 1ms interval
      InitZ80Ports();
      InitLights();

      /* main demonstration loop */
      while( 1 )
      {
            // Read the control switches
            controls = ReadControls();
            WriteDataLEDs( controls );

            // Spin the lights
            if (TestControlBit(controls, kSBusCycl))
                  RevLights();
            else
                  SpinLights();

            // Set the speed according to
            // the data knob setting
            speedNum = TestControlBit( controls, kDSel0 );
            speedNum |= TestControlBit(controls, kDSel1 ) << 1;

            wait( speeds[speedNum] );     // 50
      }
}
```