STEbus-based Hardware for a Model Railway Control System

Paul Qualtrough
Robotics Laboratory
Department of Computer Science
University of Auckland
email: paulq@cs.auckland.ac.nz

April 1998

Chapter 1

Introduction

This report describes STEbus-based hardware that has been acquired and constructed to provide the basis for a system to control a model railway. The equipment in use may be divided into three basic functional areas: the model railway equipment itself, a computing facility, and special purpose STEbus boards which transform the low power digital logic of the computer(s) into the higher powered analogue signals which drive the train motors (i.e. interfacing circuitry between the other two system components).

This main purpose of this report is to document the two types of board which make up the interfacing circuitry, both of which were locally designed and produced. In order to understand fully the design and operation of these boards, the nature of the other parts of the system will first be described.

1.1 Model Railway Equipment

The model railway equipment presently in use in the Robotics Laboratory of the Department of Computer Science at the Univeristy of Auckland comprises a number of model trains, and a network of track as shown in Figure 1.1. The trains are powered by applying a voltage across the two rails of the track on which they run. An electrical contact is made between the rails and the pairs of "power pick-up wheels" on the trains — the motor connected between these wheels propels each train. Control of a train at the lowest level can only be achieved by controlling the voltage applied to the track. For example, reversing the polarity of this voltage reverses the direction of the train's motion.

In contrast, there are more modern (and expensive) model railways in which train control is achieved by sending a message to a train. The rails are supplied with a voltage of constant magnitude and polarity, upon which are superimposed small digital signals which address a train and command it to move. This approach provides a useful environment in which problems such as routing, scheduling, co-operative versus competitive control, control based on machine learning, and so forth may be investigated.

While this might be seen as an elegant approach to model railway control, it removes all

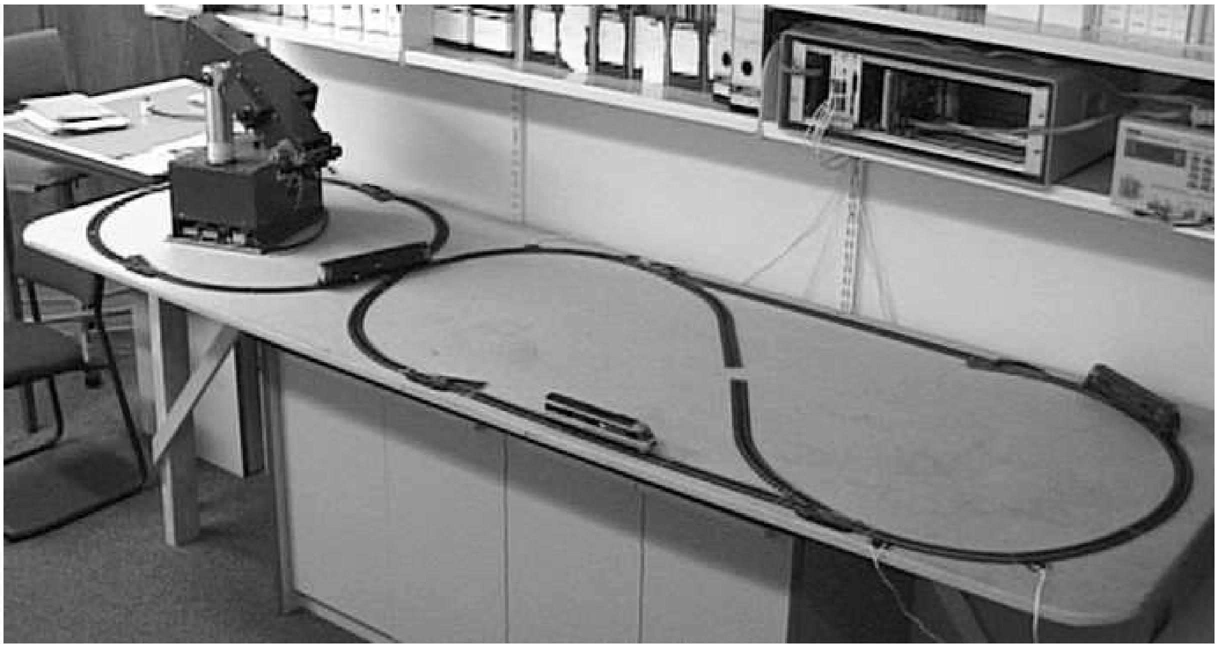


Figure 1.1: A photo of the model railway equipment in use in our Robotics Laboratory. Model trains and track are clearly evident and (with some imagination) the different types of track which make up the network can be discerned. Two cables can be seen in the bottom right hand corner of the photo supplying power to two of the segments. The model railway network is on a shelf with sufficient room to allow expansion of the network vertically, and with sufficient strength to support the weight of the small robot arm at the left hand end of the network. Above the network, a 19 inch rack can be seen. This rack houses the STEbus system which is used to control the network.

of the lower level problems by using a clever "trick" which does not necessarily generalise to the control of other devices and/or systems. The "cheap and nasty" nature of the equipment chosen provides a richer set of control problems to be solved (including those listed above), and is better suited to our educational aims.

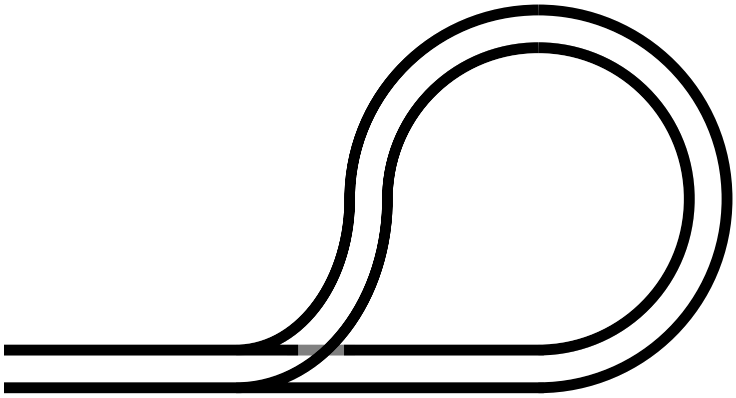
For the Lima trains which we have chosen to use, 12 volts d.c. applied across the track rails is sufficient to drive the train at a suitably fast maximum speed. Lower voltages than this will produce lower speeds, but in an inefficient manner: it is far better to pulse the motors with a square wave whose mark:space ratio is varied to set the speed, but whose amplitude remains constant at 12 volts. The Lima trains will run smoothly on a square wave of mark:space ratio 1:1 with a frequency of approximately 30Hz or more. This implies that the space interval should never exceed 30ms or so to maintain smooth travel. However, this limit depends on the length of pulse (mark) applied to the train, and will reduce proportionately as the mark length reduces.

The train motors are a brushed type of d.c. motor, and are notoriously electrically noisy. Each time the brushes cross the gap in the armature, there is momentarily no power applied to the motor windings. The magnetic field in the windings begins to collapse, and this can generate a large voltage in the opposite direction to which power was originally applied. These negative voltage "spikes" must be withstood by the circuitry powering the train. Spikes may also occur when contact between the wheels and the track rails is momentarily lost. This can be reduced (if not eliminated) by polishing the rails with emery paper.

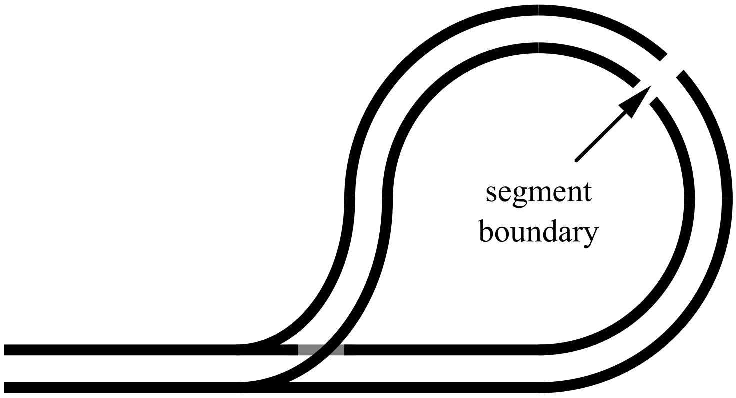
The network of track is built from track sections, some of which are straight, some are curved, some allow power input, some allow one track to cross another and some have switchable points in them to divert a train. This is the full diversity of the track sections available, and the network in Figure 1.1 uses all of them. Track sections must be connected with due respect for the fact that the curved sections are of a fixed curvature. Attempts to force track sections to connect when they are not laid out as designed will cause train derailments. Every few sections, the track must be joined with insulating joiners, with at least one section between insulated joints having power terminals. A powered group of sections between two neighbouring insulated joints is called a *segment*.

There are two reasons for segmentation of the network. First, it allows several trains to receive power independently from separate track segments. Second, it is possible for a loop of track to be connected via a set of points into a "teardrop" shape rather than a circle. Without insulation, this arrangement would cause one rail to connect directly to the other causing an electrical short circuit. Topologically, this effectively produces a Möbius strip. Such loops must be split into at least two segments, i.e. there must be at least two insulated joints in the loop for a train to succesfully traverse it (see Figure 1.2).

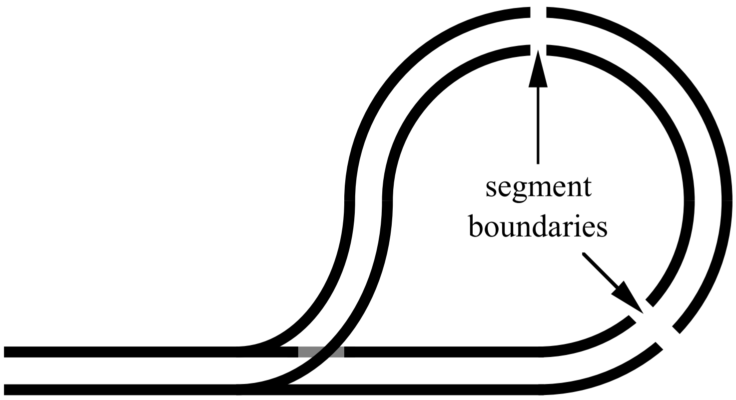
However, segmentation is not without its drawbacks: trains have at least four wheels, if not four sets of wheels, and, as evidenced by the Lima trains, tend not have the (sets of) power pick-up wheels directly across the track from each other. Figure 1.3 shows the source of the problem: in order to move in a given direction, one terminal of the train's motor must be connected to the positive rail via one or more of the wheels on that rail,



(a)



(b)



(c)

Figure 1.2: A network topology requiring segmentation. (a) Without segmentation, a permanent short circuit occurs since one rail connects directly to the other. (b) A single break in the loop will cause a short circuit when the train crosses this break. (c) At least two segments (and power switching) are required to ensure a short circuit does not occur.

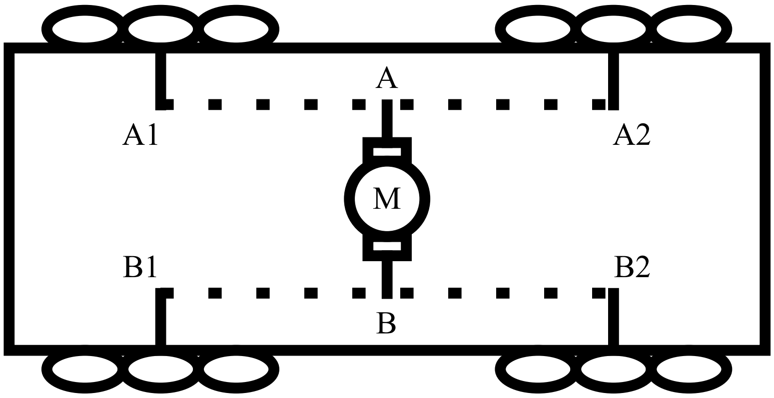


Figure 1.3: A train's motor terminals (A and B) may connect to the power pick-up wheels (A1 and/or A2, B1 and/or B2) in any one of eighteen different ways to produce the same external behaviour. Drive circuitry and control software must take account of these possibilities and their subtle implications.

while the other terminal must be connected to the neutral rail. There are eighteen ways of achieving this — each of the two sets of pickup wheels may be stationed at either or both ends of the train (three possibilities per pickup set, squared due to two pickup sets) and there are two choices of the rail which is the positive one (or, equivalently, two choices of the direction in which the train will move).

Some of these combinations are of particular concern. When the pick-up wheels are diagonally opposite each other (i.e. the motor is connected as A to A1 and B to B2, or A to A2 and B to B1 in Figure 1.3), then power must be applied simultaneously to two adjacent segments in order to power the train across the boundary between them. In such diagonally connected trains, the set of pick-up wheels which is the more positive of the two will always either lead the train as it moves, or trail it, regardless of whether the train is moving forwards or backwards. If the location of such a train is sensed electrically through the rails and is defined by the position of the *leading* set of pick-up wheels, there will be no electrical cues dictating how long one must apply power to the segment being vacated so that the train can move off it. The circuitry described in this report is able to sense the segment connected to the more positive of the pick-up wheels. More through luck than design, the Lima trains in use have their motors diagonally connected, and move with the positive wheels trailing the train.

A related issue is the occasional need to cope with track segments which are open at one end. An unpowered, isolated section of track may be used to ensure that the train does not derail itself at an open ended section of track, but once onto the unpowered section, the train has no power available to drive it back the way it came. A solution to this problem is to ensure a short circuit brings the train to a halt as it reaches the unpowered segment, but that it does not exist or persist when the power to the train is reversed to drive it away from the unpowered segment. A PTC thermistor across the rails of the unpowered segment provides the best solution known to this problem by ensuring the short circuit doesn't persist. [1] Use of diodes across the unpowered segment can only handle either leading or trailing positive power pick-up wheels depending on the direction the diode is placed, and cannot support trains whose power pick-up wheels are directly opposite each other.

Points have three terminals, the centre one being a common terminal to the circuits required to switch the points one way or the other. A short pulse (a few tens of milliseconds) at 12 volts d.c. applied between the common terminal and one of the other terminals is more than sufficient to switch the points in one direction. The same pulse applied between the common terminal and the remaining terminal will switch the points back the other way. Pulses of longer than a few tens of milliseconds may cause damage to the points, and pulses of a few seconds (e.g. applied manually) will burn the points out completely. The polarity of the voltage applied is not relevant, so a.c. voltages may also be used [2].

1.2 STEbus Computer System

The model railway equipment just described is modular, and can in principle be expanded indefinitely to meet the requirements of those using it. Accordingly, the interfacing circuitry to it must also be modular, and the computing facility which connects to the interface circuitry must be able to support its modularity. A bus-based system, which allows hardware modules to be inserted and removed easily, is strongly implied if not mandatory. In order to cope with the power requirements of the model railway equipment, make future expansions easy and economical, allow easy production of custom interfacing hardware, and potentially provide a useful environment for other control projects, the IEEE STEbus system [1] was chosen as a basis for the computing facility. A photo of the STEbus system in use is shown as Figure 1.4. The STEbus system shown comprises:

- a 3U high, 19 inch subrack mounted in a case;
- a 10-slot STEbus backplane;
- the six boards identified in Figure 1.4 which plug into the STEbus backplane; and

[1] A PTC thermistor may be thought of as a self-resetting fuse. Normally the thermistor presents a small resistance to a circuit, but when an excessive current is passed, it rapidly heats up and its positive temperature coefficent (hence PTC) causes its resistance to increase enormously and prevent any significant current flow. The PTC will reset itself when it cools back to its normal operating temperature. In this case, as soon as the train's leading wheels connect the unpowered segment to the powered one, the PTC will provide a short circuit which will halt the train. The short circuit will not persist due to the fuse effect, and allows the train to move back the way it came, provided its pick-up wheels are still in contact with the powered segment.

[2] In fact, it is preferable to use a.c. power to drive the coils since coils consume negligible power from an a.c. source. This reduces (if not eliminates) the possibility of points burn-out. However, generating an a.c. signal from a d.c. source increases the complexity of the circuitry, and it is easier to design hardware which generates a short and safe d.c. pulse.

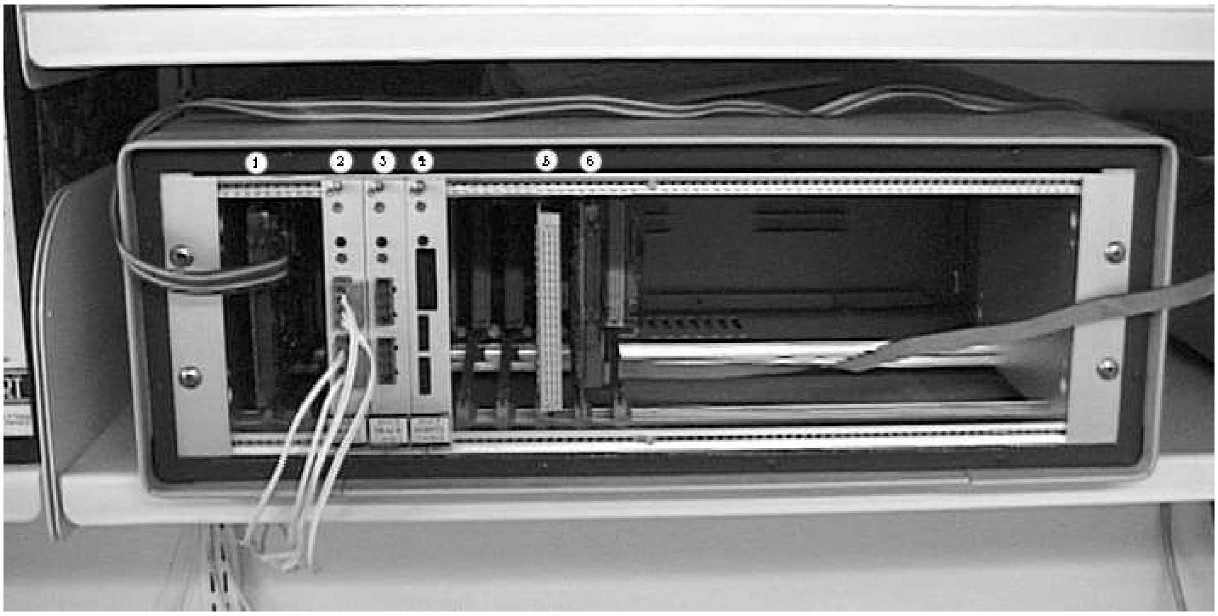


Figure 1.4: The Robotics Laboratory STEbus system. There are six boards in the chassis, numbered (minutely) from left to right in small spots above each board. The boards are: (1) an Arcom SCIM88 processor board, (2 and 3) track control boards (described in this report), (4) a points control board (also described here), (5) a passive bus extender board (used as an aid to hardware diagnosis), and (6) an Arcom SPCOM PC-compatible communications board. This arrangement of boards has been found to be the most practical thus far, although all boards may be placed anywhere on the STEbus backplane.

- a small, switch-mode power supply [3] mounted in the rear of the case behind the STEbus backplane.

The majority of STEbus equipment world wide is produced by Arcom Control Systems of Cambridge, England, who provide a comprehensive range of hardware and software products for the STEbus, mainly with application to industrial control applications [9]. Two STEbus boards from their range have been purchased: the SCIM88 microprocessor board [8] and the SPCOM PC-compatible communications board [5]. The SPCOM card is transparent for most purposes, and will not be discussed. Its purpose is to provide a serial link between a PC-based development system, on which code is developed, and a compatible STEbus processor board, on which code is exectuted.

The SCIM88 is based on a 16MHz Intel 80C188 microprocessor [3], has a fixed 64Kbytes of static RAM, and can support up to 512Kbytes of EPROM or Flash EPROM. When a program's combined code and data requirements exceed the 64Kb limit, its code

[3] At the time of writing, this could do with some attention. It has no switch controlling its mains power, and it uses a less than elegant resistor jammed into two pin receptors on one of the STEbus connectors to ensure that the manufacturer's recommended minimum of 0.35 amps is drawn from its -12 volt supply. Also, the right hand half of the subrack is vacant, potentially allowing people to touch the power supply. This almost certainly contravenes safety regulations. A rectangle of metal sheet mounted at the rear of the subrack alongside the STEbus backplane would fix this.

and static data must be placed in EPROM or Flash EPROM in order to run. Below this limit, code and data may be downloaded dynamically from a PC-based development environment to the SCIM88 via the serial link. The program running on the SCIM88 may be debugged remotely from a PC using Arcom's SourceView product [10],[7] in conjunction with Borland's Turbo Debugger [4]. Support for real-time multi-tasking applications is provided by Arcom's SourceTask package [6].

The 80C188 microprocessor, despite a model number hinting at a design from yesteryear, is quite modern, sophisticated and more than sufficient for a great deal of "real world" control tasks, since the real world operates on a time scale measured typically in milliseconds, not fractions of microseconds. Control problems requiring greater processing power may be implemented in a distributed and/or multiprocessor arrangement, using the SCIM88 to implement lower level functions. Additional processors may be tightly coupled to it through the STEbus, or loosely coupled via a serial link, and are employed to handle more demanding processing.

Interfacing the SCIM88 through the STEbus to other devices is straightforward from a programmer's perspective as all I/O devices are mapped into either STEbus memory or I/O space. The I/O space has a shorter address and is more convenient to use due to the Intel approach of constructing memory addresses from separate segment and offset components. Interfacing devices from a system's manager perspective is a matter of inserting them into the STEbus: there are no backplane jumpers to worry about as there are for some other bus systems, e.g. VMEbus [4]. Interfacing devices to the STEbus from a hardware designer's perspective is an issue better covered by referring to (for example) [11] and [12].

An item of note which is difficult to realise from a quick skim through the Arcom manuals, is that using the SPCOM for serial communications is not straight forward since it is already used by a ROM-resident loader program used to download programs to the SCIM88. Whenever the SPCOM receives a character, it generates an interrupt. For the loader, this interrupt should be the STEBUS ATNRQ0* interrupt which generates an NMI (non-maskable interrupt) on the SCIM88. The interrupt service routine on the SCIM88 aborts the program running at the time, and replaces that program with whatever it downloads.

To use the SPCOM for other purposes, one or more of the following must be changed:

- the serial port used on the SPCOM;
- the interrupt generated by the SPCOM serial port; or
- the SCIM88 NMI interrupt sevice routine.

The first of these options requires a different cable than used for downloading. The second can be achieved by changing a link on the SPCOM board after downloading has occurred and before the serial port is used e.g. change LK2 from position A to position B

[4]Note however that boards should not be inserted or removed while the STEbus system is powered up.

and use the vector for the STEbus ATNRQ2* interrupt). The third option option would be preferable, but is likely to require the most work to achieve since this routine exists only as compiled code in ROM. Ultimately, the loader and the other program would most sensibly both reside in ROM. The second program would be executed upon reset, while the interrupt service routine would be coded so that it switches to the loader when an unambiguous sequence of characters from the remote download program is sent to it.

Chapter 2

Track Control Board

This chapter describes the first prototype of the track control board, designated AUCS94.1 Rev 0.1.

2.1 Overview

A block diagram of the track control board AUCS94.1 Rev 0.1 is shown in Figure 2.1. The track control board independently controls four segments of track. Each segment should be connected by two wires from a track section with power terminals, to an appropriate pair of pins on the connectors P2 or P3 (see Table 2.1). The polarity of the connection made is not significant because the board has independent relays able to change the polarity of the power applied to each segment. An additional ceramic capacitor of around 0.1μF may be connected between the power terminals on the track to reduce noise from the train motor if required.

The track control board is used by reading from and writing to two 8-bit registers on the board. The addresses of the registers are set by the address jumpers J1-J19. Each address jumper, Jn, relates directly to the STEbus address line, An, and in turn to a bit in the address used by a program accessing the board. The CONTROL register is addressed when the address on the appropriate address lines matches those set by the jumpers, and the least significant address line (A0) is at logic 0. I.e. it is found at an

Segment	Connector	Pins
1	P2	3 and 4
2	P2	1 and 2
3	P3	3 and 4
4	P3	1 and 2

Table 2.1: Track segment power sources

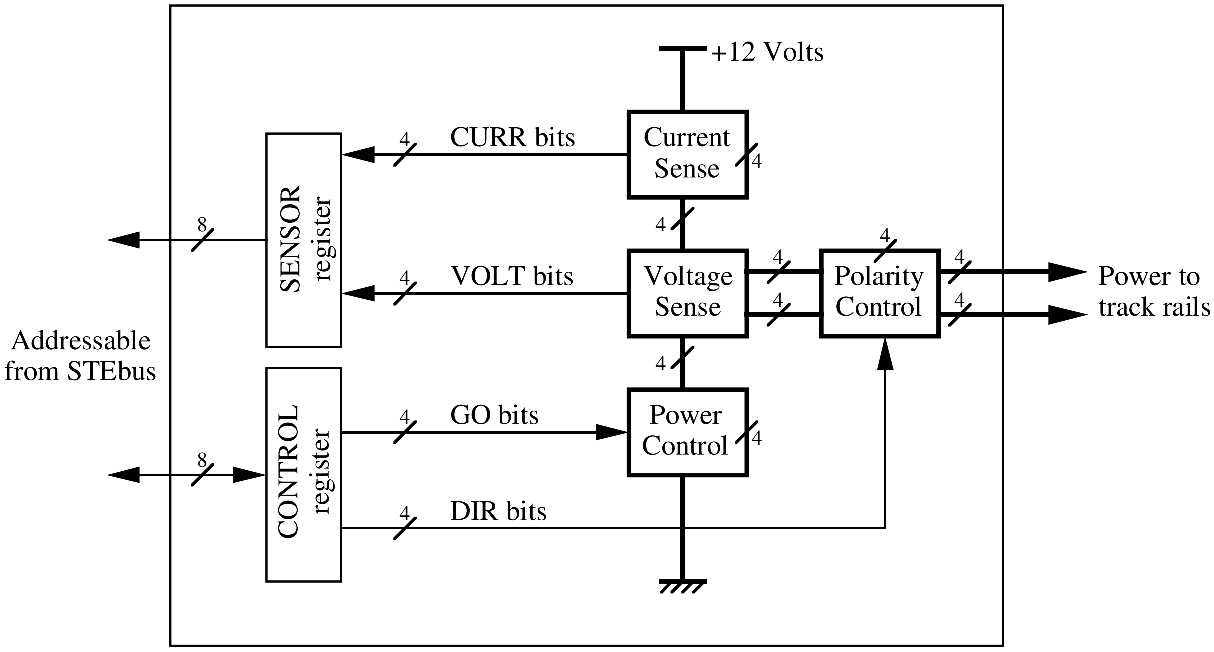


Figure 2.1: Block diagram of the AUCS94.1 Rev 0.1 track control board. The board independently controls four isolated segments of track through reading and writing bits in two onboard registers.

offset of 0 from the address set by the address jumpers. Similarly, the SENSOR register is addressed when the address lines match the jumper settings, and A0 is logic 1, i.e. at an offset of 1 from the jumpered address. Installed jumpers match address lines at logic 0; removed jumpers match address lines at logic 1.

Addressing mode jumpers J20 and J21, if removed, enable the board to respond to STEbus memory transfers and I/O transfers respectively. If both jumpers are installed, the board will not respond to any STEbus transfers. So, for example, if all jumpers were installed except J4-J9, J12-J15 and J21, the CONTROL register will appear at I/O address 3F0 hexadecimal, the SENSOR register will appear at I/O address 3F1, but neither register will appear in STEbus memory space (and the contributions of J12-J19, viz. 0F000 hexadecimal, are ignored). This is summarised in Table 2.2.

The CONTROL register is a read-write register arranged as four pairs of bits, each pair controlling a different track segment. One bit of the pair (GO) turns the power on (logic 1) or off (logic 0) to a segment, while the other (DIR) sets the direction a train will travel along the segment by setting the polarity of the voltage supplied to the track. A train may be powered by continually changing the state of the GO bit on its segment(s). Varying the lengths of time that the GO bit is held at logic 1 and logic 0 will vary the speed of the train. A read from the CONTROL register reflects the last value written to it, except under fault conditions (see circuit description).

The SENSOR register is a read-only register also arranged as four pairs of bits, each pair reflecting the state of a different segment. One bit of the pair (VOLT) indicates the

Jumper	Address Line	Address contribution (Hex)
J1	A1	2
J2	A2	4
J3	A3	8
J4	A4	10
J5	A5	20
J6	A6	40
J7	A7	80
J8	A8	100
J9	A9	200
J10	A10	400
J11	A11	800
J12	A12	1000
J13	A13	2000
J14	A14	4000
J15	A15	8000
J16	A16	10000
J17	A17	20000
J18	A18	40000
J19	A19	80000
J20	Enables STEbus memory transfers if removed	
J21	Enables STEbus I/O transfers if removed	

Table 2.2: Effects of address and addressing mode jumpers. All jumpers take effect when removed.

Register	Adress Offset	Bit	Signal
CONTROL	0	0 (LSB)	DIR1
CONTROL	0	1	GO1
CONTROL	0	2	DIR2
CONTROL	0	3	GO2
CONTROL	0	4	DIR3
CONTROL	0	5	GO3
CONTROL	0	6	DIR4
CONTROL	0	7 (MSB)	GO4
SENSOR	1	0 (LSB)	CURR1
SENSOR	1	1	VOLT1
SENSOR	1	2	CURR2
SENSOR	1	3	VOLT2
SENSOR	1	4	CURR3
SENSOR	1	5	VOLT3
SENSOR	1	6	CURR4
SENSOR	1	7 (MSB)	VOLT4

Table 2.3: CONTROL and SENSOR register bit assignments.

VOLT bit	CURR bit	Meaning
1	1	Train detected on segment
1	0	Nothing detected on segment
0	1	Short circuit detected on segment
0	0	Electrical fault detected on segment

Table 2.4: Interpretation of sensor bit combinations

presence (logic 1) or absence (logic 0) of significant voltage at the corresponding pins on P2 or P3 which connect to the relevant track segment, while the other bit (CURR) conveys the same information for significant current flowing through the corresponding pins on P2 or P3. A write to the SENSOR register will have no effect. The bit assignments in each register are shown in Table 2.3.

The four possible combinations of VOLT and CURR bits may be used to distinguish between the various conditions which may exist on a segment of track, as shown in Table 2.4. Correct voltage and current status will only be indicated a few tens of microseconds after the corresponding GO bit in the CONTROL register has been turned on (logic 1). Reading the SENSOR register too soon after corresponding GO bits in the CONTROL register have changed from logic 0 to logic 1 will usually result in a (false) fault condition being decoded from the VOLT and CURR bits (i.e. both at logic 0).

The circuitry is not able to tell where on a segment a train is, but can determine when the more positive of the pick-up wheel (sets) on a train crosses from one segment to the next using this information. When the trains being controlled have pick-up wheels diagonally opposite each other, this information is sufficient to detect either the train starting to cross from one segment to another, or completing the crossing between two segments, but not both. As noted, it is desirable that the positive pick-up wheels trail the train so that additional sensing or time estimates are not required in order to switch power off to the segment being vacated.

When a segment is powered (i.e. its corresponding GO bit is at logic 1), the SENSOR register must be checked at a rate sufficient to allow early detection of short circuit or fault conditions. Power should be immediately turned off (by setting the GO bit to logic 0) on any segment indicating such a condition. The green activity LED on the front panel is a visual indication of the rate at which the SENSOR register is being read. It should remain continuously lit while any segment is powered from the board. If it flickers, it is likely that the state of the segment is not being checked often enough to detect and act on these conditions which could cause damage to the circuit.

The red fail LED on the front panel is lit under two circumstances. When the STEbus SYSRST* signal goes low, the LED will be lit and will remain lit until this signal goes high. When power to the analogue part of the board fails, the LED will be lit and will remain lit until power is restored. The latter condition is not expected to occur in normal operation, and will be self-resetting if the PTC thermistor (T1) is installed (see circuit description). Either condition will cause the CONTROL register to be cleared, turning power off to all segments controlled by the board.

An example of how this board may be used in a C programming environment is illustrated by the following code fragments:

```
struct t
{
    int     t_polarity1:1;   /* Polarity of power on 1st segment */
    int     t_power1:1;      /* State of power on 1st segment */
    int     t_polarity2:1;   /* Polarity of power on 2nd segment */
    int     t_power2:1;      /* State of power on 2nd segment */
    int     t_polarity3:1;   /* Polarity of power on 3rd segment */
    int     t_power3:1;      /* State of power on 3rd segment */
    int     t_polarity4:1;   /* Polarity of power on 4th segment */
    int     t_power4:1;      /* State of power on 4th segment */
    int     t_current1:1;    /* State of current on 1st segment */
    int     t_voltage1:1;    /* State of voltage on 1st segment */
    int     t_current2:1;    /* State of current on 2nd segment */
    int     t_voltage2:1;    /* State of voltage on 2nd segment */
    int     t_current3:1;    /* State of current on 3rd segment */
    int     t_voltage3:1;    /* State of voltage on 3rd segment */
    int     t_current4:1;    /* State of current on 4th segment */
    int     t_voltage4:1;    /* State of voltage on 4th segment */
} track;

#define T_ON              1
```

```
#define T_OFF           0
#define T_HIGH          1
#define T_LOW           0
#define T_NORMAL        0
#define T_REVERSE       1
#define T_IO_ADDRESS    0x0C00
      .
      .
      .
{
    track.t_power1 = T_ON;
    track.t_polarity1 = T_REVERSE;
    track.t_power2 = T_OFF;
    track.t_power3 = T_OFF;
    track.t_power4 = T_OFF;
    set_track( T_IO_ADDRESS, &track );
      .
      .
    /* Insert code here to give a delay before reading SENSOR register */
      .
      .
    get_track( T_IO_ADDRESS, &track );
    if ( track.t_voltage1 == T_HIGH )    /* Situation normal */
        if ( track.t_current1 == T_HIGH )
            printf( "Train on segment 1\n" );
        else
            printf( "No train on segment 1\n" );
    else    /* A fault condition, or perhaps insufficient delay used */
        if ( track.t_current1 == T_HIGH )
            printf( "Short circuit on segment 1\n" );
        else
            printf( "Fault on segment 1\n" );
    track.t_power1 = T_OFF;
    set_track( T_IO_ADDRESS, &track );
}

void set_track( short addr, struct t *track )
{
    outportb( addr, *(char *)track );
}

void get_track( short addr, struct t *track )
{
    *(short *)track = inportw( addr );
}
```

Signal Name	Meaning
An	STEbus address line n (e.g. A2, A13)
CLK16	A cleaned up version of the 16Mhz STEbus SYSCLK signal
MTEN	STEbus memory transfers are enabled (J20 is removed)
IOTEN	STEbus I/O transfers are enabled (J21 is removed)
CM1	STEbus command modifier bit 1 (memory/io* selector)
CM0	STEbus command modifier bit 0 (read/write* selector)
HIVAL*	When low, the highest 8 STEbus address bits (A12 – A19) are valid
LOVAL*	When low, lower STEbus address bits A1 – A11 (but not A0) and STEbus control signals ADRSTB*, DATSTB* and CM2 are valid
GDCLK	The CONTROL register clock signal
VICLK	The SENSOR register clock signal
GDRDEN*	Enables a read from the CONTROL register (when low)
VIRDEN*	Enables a read from the SENSOR register (when low)
TFRERR	Indicates that an STEbus transfer error has occurred
DATACK	Indicates that an STEbus transfer has completed successfully
RESET*	Track control board reset signal (active low)
SYSRST*	The (active low) STEbus system reset signal
PWROK	The 12 volt power supply to the analogue circuitry is working

Table 2.5: Summary of Track Control Board Signals.

2.2 Circuit Description

The circuit schematics for the track control board are shown as Figure 2.2 and Figure 2.3. The track control board logically (and visibly) comprises two parts: a digital part and an analogue part. The digital part implements an STEbus slave interface to support the two registers described above. The analogue part boosts the output power to drive a train and senses the voltage and current drawn at its output.

2.2.1 Track Control Board: Digital Part

The digital part of the track control board (AUCS94.1 Rev 0.1) is functionally equivalent to the digital part of the points control board (AUCS94.2 Rev 0.1) in every respect. Both generate a number of SETbus and auxilliary signals. The names and meanings of these signals are summarised in Table 2.5. Any signal suffixed with an asterisk (*) indicates that the signal is "active low", that is, when active (or true or on), the signal assumes logic value 0 rather than logic 1.

The STEbus address, address strobe and data strobe lines are decoded by the three 688-style magnitude comparators [1]. One of these (U6) produces the HIVAL* signal indicating the validity of the 8 most significant address bits used only in STEbus memory

[1] By "688-style" I mean a 74LS688 or 74HCT688 chip. See the closing paragraph in this section for a

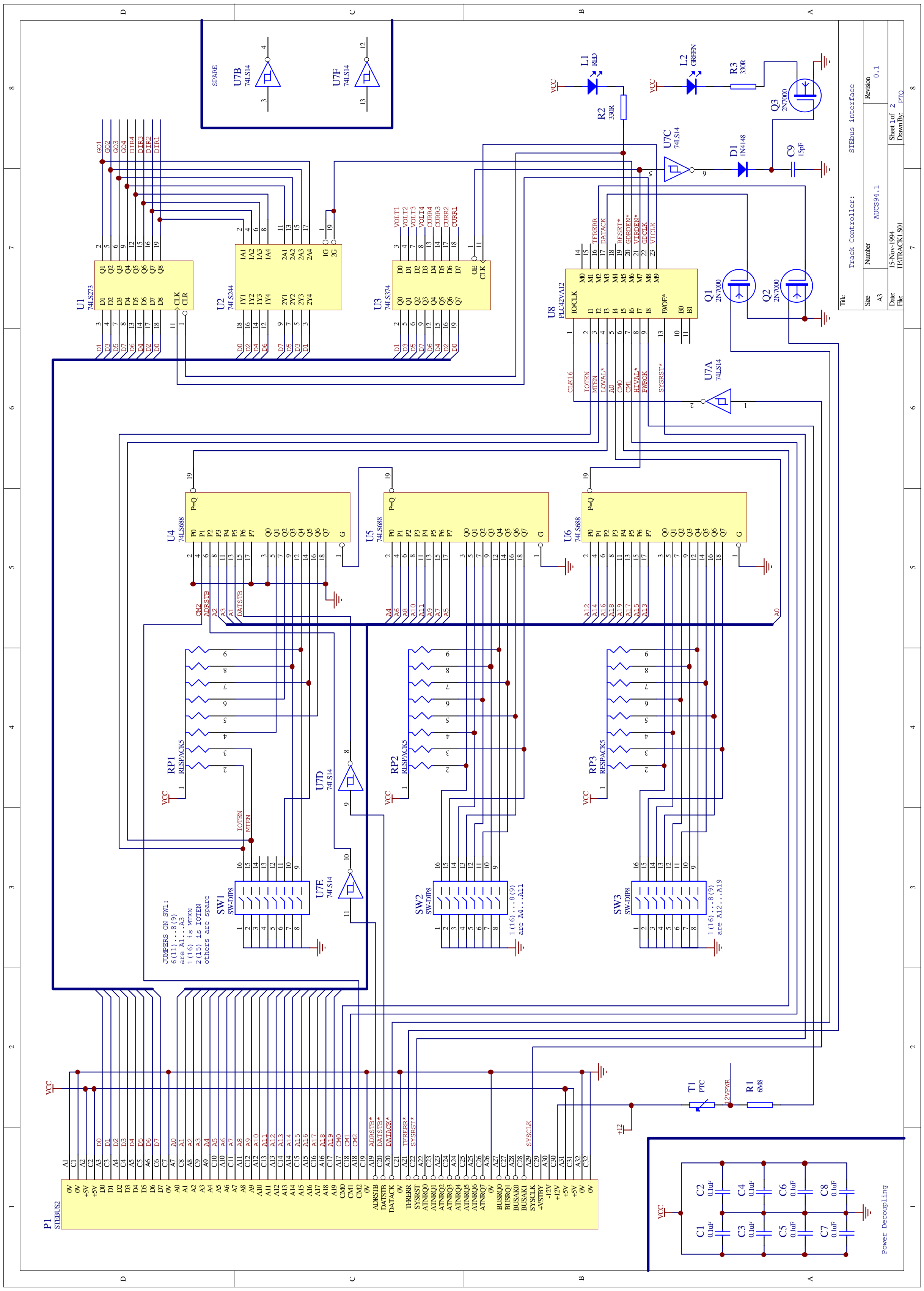


Figure 2.2: Track Control board circuit schematic: page 1 of 2

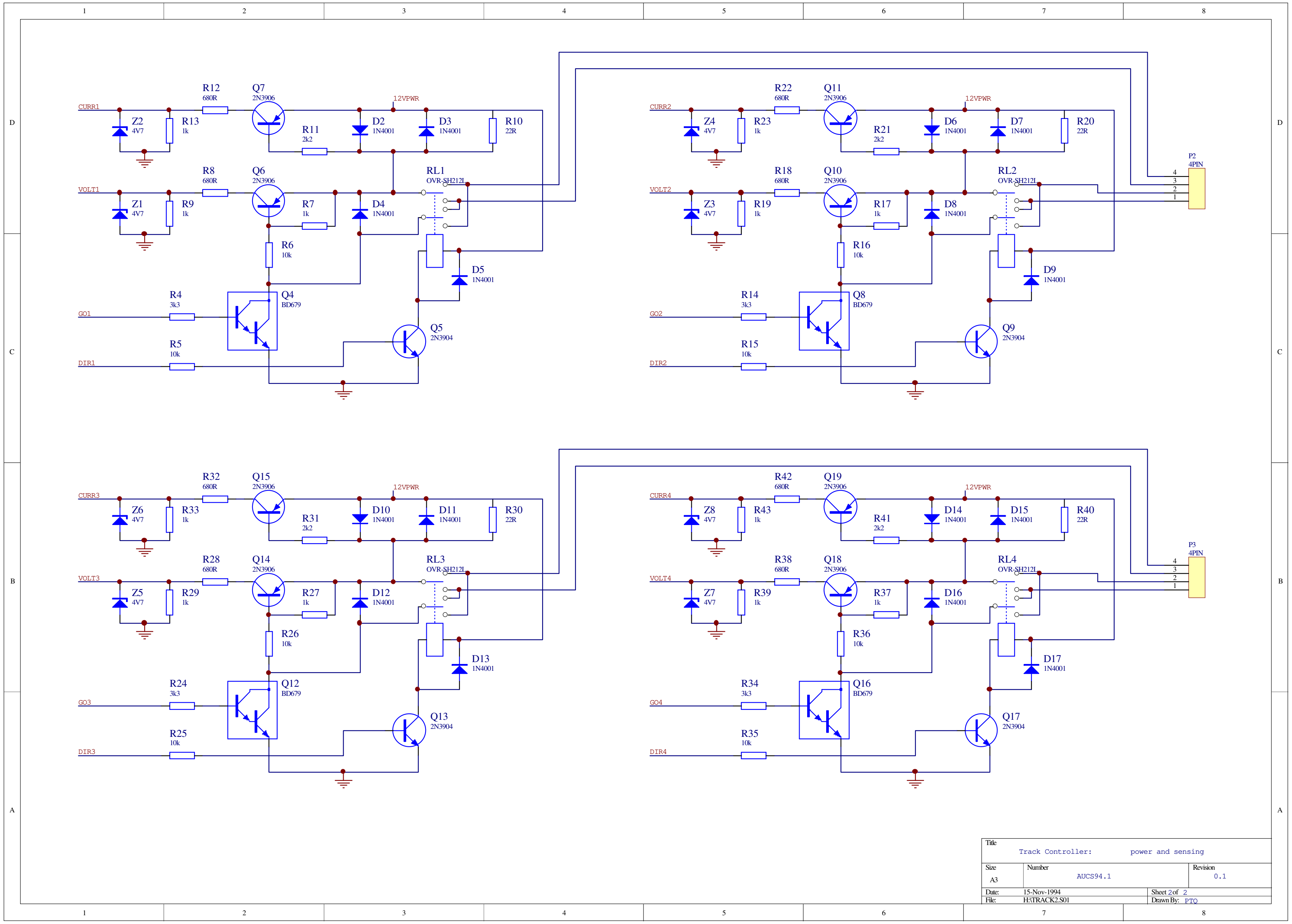


Figure 2.3: Track Control board circuit schematic: page 2 of 2

transfers. The other two (U4 and U5) are daisy chained and produce the LOVAL* signal used in both STEbus memory and I/O transfers to indicate validity [2] of the lower address bits and control signals. Memory and I/O transfers are enabled and disabled independently through jumpers J20 and J21, and give rise to the MTEN and IOTEN enabling signals respectively.

The pair of onboard registers are controlled by two pairs of signals, one for reading the register (the xxRDEN* signal) and one for writing to it (xxCLK signal). Reads from and writes to the CONTROL register are implemented in separate devices. A write is latched into a 273-style octal latch (U1) by the rising edge of the GDCLK signal. A read passes a copy of the value in U1 back to the STEbus by enabling the outputs of a 74LS244 tri-state buffer (U2) when the GDRDEN* signal goes low.

The SENSOR register is implemented in a single 74LS374 tri-state octal D-type flip-flop (U3). A read from it enables its outputs onto the STEbus when the VIRDEN* signal goes low. Data from the analogue part of the board is latched into U3 by the rising edge of the VICLK signal. A write to the SENSOR register from the STEbus has no effect due to the programming of U8 described below.

The PLC42VA12 programmable logic device (U8) produces the four register control signals just mentioned, and indirectly drives the two STEbus acknowledge lines DATACK* and TFRERR* via the FETs Q1 and Q2. It is programmed with the following logic equations written in the notation of the Philips SLICE(tm)package environment [2] (viz. / represents inversion, + represents logical OR, * represents logical AND). Active low signals on the schematic and in Table 2.5 which are suffixed with an asterisk (*) are instead prefixed with the letter N, since the asterisk is used as an operator by SLICE(tm). Two internal flip-flops (FF0 and FF1) are used to create a delay for the DATACK signal during a read cycle. The other signals require only combinational logic.

```
FF0.CLK   = CLK16;
FF0.D     = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL );
FF1.CLK   = /CLK16;
FF1.D     = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL );
GDCLK    = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*/CM0*/A0;
VICLK    = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*CM0*A0;
NGDRDEN  = /( /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*CM0*/A0 );
NVIRDEN  = /( /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*CM0*A0 );
TFRERR   = /NLOVAL*( /IOTEN*/CM1 + /MTEN*/NHIVAL*CM1 );
DATACK   = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*/CM0 + FF0*FF1*CM0;
NRESET   = PWROK*NSYSRST;
```

To be noted from these equations is that the VIRDEN* and VICLK signals are active under the same conditions, so that a read to the SENSOR register simultaneously latches

brief discussion of the constraints on the types of logic chips which may be used.

[2] "Validity" means that the address bits match the board's address as configured on J1-J19 (for both LOVAL* and HIVAL*), and the control signals are requesting a valid data transfer (explicit in the LOVAL* signal only).

the data into U3, and drives the STEbus data lines with it. Alternatively, the VICLK signal could have been programmed analogously to the GDCLK signal, so that data could be independently latched into the SENSOR register by writing a dummy value to it. No particular advantage was seen in doing this.

Of the components used, U8 is a CMOS device which drives TTL outputs, a fact which may be some cause for concern — it does not fall within STEbus standards for voltage and current. The other integrated circuits used must be Low Power Schottky TTL (LSTTL) devices if used to drive the STEbus, or can be either LSTTL or High Speed TTL-compatible CMOS (HCT) devices if they only receive STEbus signals. Although the Schmitt Trigger Inverter (U7) is used only to receive some signals from the STEbus, the SYSCLK signal is among them. This signal (produced by the SCIM88) is thought unlikely to be able to reliably switch an HCT device, hence the use of LSTTL.

2.2.2 Track Control Board: Analogue Part

The analogue part of the track control board has four duplicated sets of components for controlling and sensing the track, and two indicator LEDs. The fail LED (L1) is driven through a resistor (R2) from U8, and will light in the event of a failure of the +12 volt power supply (12VPWR) to the analogue part (lit until power is restored) or by an STEbus system reset (lit for as long as SYSRST* stays low). The activity LED (L2) is driven through a resistor (R3) from a FET (Q3) controlled by the voltage stored in a capacitor (C9). This capacitor is charged fast through a diode (D1) by the inverted read pulse to register 1 (VIRDEN*), and discharged slowly back through the diode, producing a flash of light on the LED with a duration of a few tens of milliseconds.

The PTC thermistor (T1), if available, is designed as a protection against short circuits of unreasonable length. Prolonged short circuits will cause T1 to heat up, increasing its resistance, and shutting down the +12 volt power supply to the analogue circuitry. In turn, this will cause the PWROK input to U8 to go low, driving the RESET* signal low, turning on the fail LED and clearing the CONTROL register as a result. The clearing of the CONTROL register stops the short circuit, allowing T1 to cool, restoring +12 volt power, driving RESET* high again and extinguishing the fail LED. While the fail LED is lit, any write to the CONTROL register will have no effect: it will remain cleared. Programmers may use this effect to detect for the restoration of power to the analogue circuitry, but are advised that their software should be written to prevent such conditions occurring at all.

The analogue circuitry used for powering and sensing the track segments corresponds quite directly to the bits in the two registers. The GO signals reflect the states of the GO bits and are used to drive a Darlington power transistor (e.g. Q4 for segment 1) as a saturated switch which turns power on to the track. A resistor (R4) limits the short circuit current to around 1.5A, while a reverse-biased diode (D4) protects against negative voltage transients from any connected train motors.

The DIR signals reflect the DIR bits and drive a lower powered transistor (Q5) as a saturated switch which in turn switches the DPDT relay (RL1) to set the polarity of the power supplied to the track. A resistor (R5) is chosen to ensure the transistor is saturated at the operating current of the relay, and a diode (D5) protects the transistor against the reverse voltage of the collapsing magnetic field in the relay.

The levels of the VOLT and CURR signals which are latched as the VOLT and CURR bits in the SENSOR register are determined by the states of two PNP transistors (Q6 and Q7). When on, these transistors will be at or near saturation, and will cause sufficient current to flow through their respective zener diodes (Z1 and Z2) to produce a high level on their corresponding signal line (VOLT and CURR). This current is limited by resistors (R8 and R12) to conserve power, and a further pair of resistors (R9 and R13) are used to ensure that a low signal level is produced when the transistors are cut off.

The voltage sensing transistor (Q6) is switched on when the voltage across the track (or across D4) exceeds approximately 7 volts. Two resistors set this threshold: the first (R6) is chosen large enough to limit the current drawn in the whole of the voltage sensing circuitry to below the threshold for the current sensing circuitry, and then the second resistor (R7) is chosen (with a high degree of trial and error) to give the desired threshold voltage.

Current is sensed by monitoring the voltage across a diode (D2) in series with the track power outputs. When this diode conducts, so too will the current-sensing transistor (Q7). The resistor (R11) connected to the base of this transistor is chosen mainly to protect it should the diode (D2) ever burn out due to a prolonged short circuit. This resistor can vary an order of magnitude either way without affecting the current threshold which is set by the other resistor (R10). This resistor effectively provides a bypass to the diode (D2) when only the small current through the voltage sensing circuitry is flowing. It is chosen to set a threshold of around 50mA. The trains in use draw around 200mA when moving freely, and more when starting or subjected to loads or friction.

The final diode (D3) in the circuit is to provide a reverse conduction path for when the train is crossing between two segments. Because train motors are connected to diagonally opposite sets of wheels, there is a short time where a train sources current from the circuitry of one segment, and sinks it to that of another. Without this extra diode, the diode used to sense current (D2) would block the reverse current from the motor, and potentially damage the circuit.

2.2.3 Component Layout and Connectors

Figure 2.4 shows the component overlay for the track control board. The numbering of the pins on connectors P1 to P3, as seen with the component side of the board facing upwards and from the perspective of the female connectors which connect to them, are shown in Figure 2.5 and Figure 2.6.

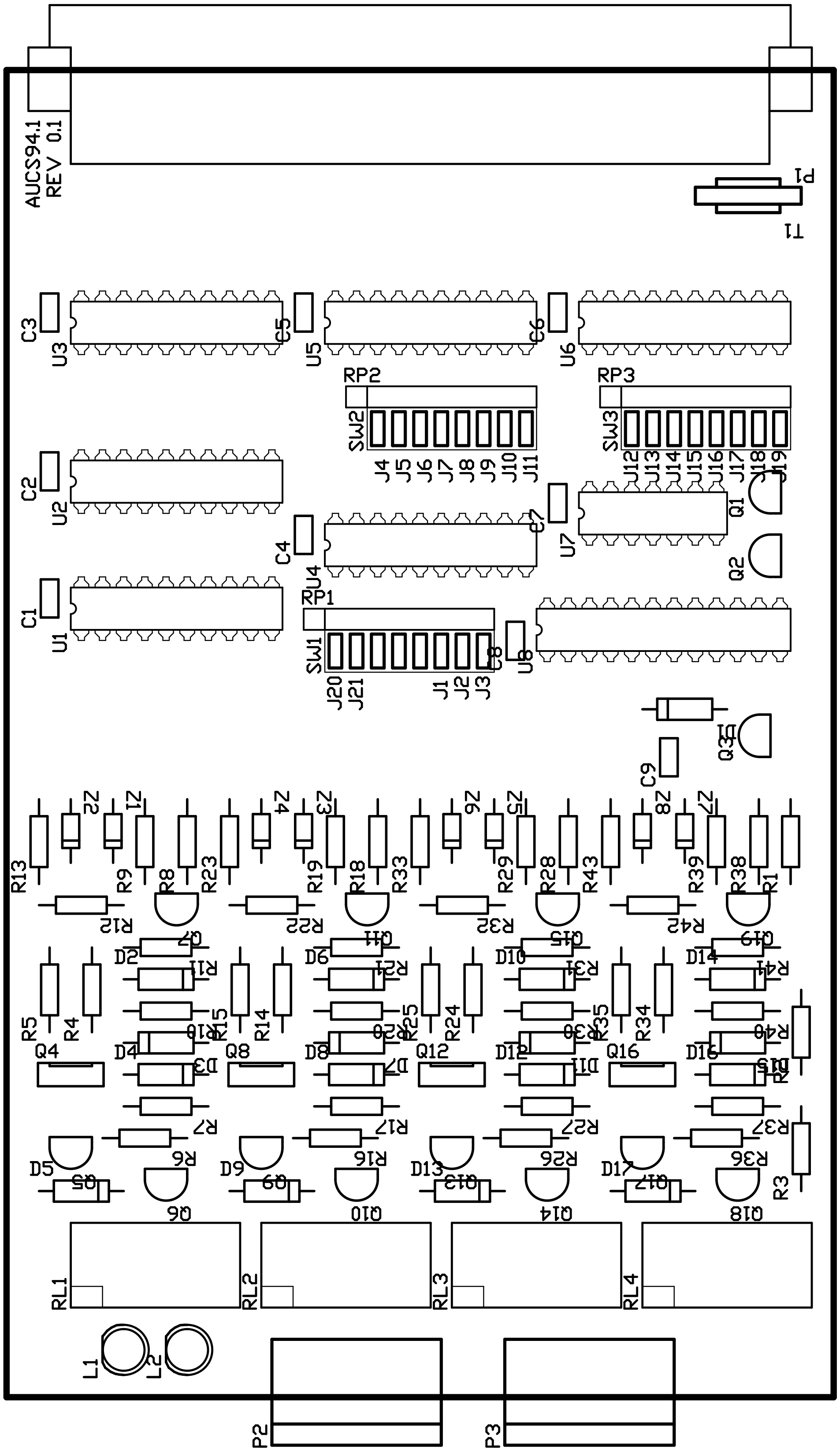


Figure 2.4: PCB component overlay for Track Control Board

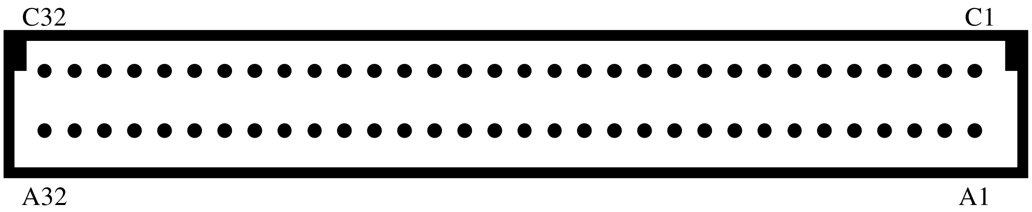


Figure 2.5: Pin assignments on STEbus connector P1

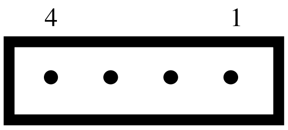


Figure 2.6: Pin assignments on track segment connectors P2 and P3

2.2.4 Deviations from Prescribed Design

Some modifications have been made to the circuit boards in use. They are:

- The SYSCLK signal has been put through an extra inversion by passing it first through the spare inverter between U7.13 and U7.12 before it goes into the input of the inverter U7.1.
- The PTC thermistor T1 is unavailable in New Zealand (but can be acquired in Europe, for example: Philips part 2322 661 54111). A wire link has been substituted in its place. A slow-blow fuse of approximately 500mA rating would provide the protection the PTC was initially designed for, but with the inconvenience of not being self-resetting. As mentioned, software should be written to ensure such protection devices are never required to perform.

2.3 Known Performance Problems

On the first board constructed, bus time-out errors occur frequently. Once they begin to occur, they do not stop without intervention. They can be alleviated by putting the board at the end of the STEbus farthest away from the SCIM88 processor board, or failing that, on a bus extender board.

This strongly suggests that there are signal propagation problems, probably caused by impedance mismatches. The STEbus specification requires that no STEbus signal travel more than 50mm into a board. Several signal lines fail to comply with this rule. On the other hand, the onset of bus time-out errors suggests a thermal component to the problem, and the PLC42VA12 runs suspiciously warm for a CMOS device. Attempts to isolate the cause of the problem, which has not been noted on the second board constructed, have so far failed.

Also, the green (activity) LED may remain lit for long periods when warming up. This is because the reverse leakage current through the 1N4148 diode (D1) is quite sensitive to temperature. In fact it is quite sensitive to quite a spectrum of radiation, e.g. from fluorescent tubes, and ought to be replaced by a diode with similarly low leakage current, but one which is less sensitive to its environment.

2.4 Advice for Future Revisions

The following points should be kept in mind when designing future revisions of this board:

- Address the deviations made from the prescribed design after the printed circuit boards were made.
- Address the performance problems outstanding with this revision.
- Under most circumstances, the voltage sense line represents an inversion of the voltage level at the collector of the darlington power transistor. If this is true in all cases, it may allow some circuit simplification.
- The entire functionality of the digital part of the circuit can be programmed into modern programmable logic devices (e.g. from XILINX). Provided the STEbus specifications are respected, which may require some additional Schmitt trigger devices between the bus and the programmable device, using such a device may result in a dramatic space saving on the board.

Chapter 3

Points Control Board

This chapter describes the first prototype of the points control board, designated AUCS94.2 Rev 0.1.

3.1 Overview

A block diagram of the points control board AUCS94.2 Rev 0.1 is shown in Figure 3.1. The points control board independently controls eight sets of points. Each set of points should be connected by three wires from the points terminals to appropriate pins on connectors P3 to P6 (see Table 3.1). Additionally, the points control board provides eight independent digital inputs for use with discrete sensors such as optical switches or reed relays. These sensors must use circuits which connect to an appropriate pair of pins on connector P2 (see Table 3.2).

The points control board is used by reading from and writing to two 8-bit registers on board. The addresses of the registers are set by the address jumpers J1-J19. Each address jumper, Jn, relates directly to the STEbus address line, An, and in turn to a

Points Set	Connector	Common Pin	Left Pin	Right Pin
1	P3	4	5	6
2	P3	3	1	2
3	P4	4	5	6
4	P4	3	1	2
5	P5	4	5	6
6	P5	3	1	2
7	P6	4	5	6
8	P6	3	1	2

Table 3.1: Points driving sources

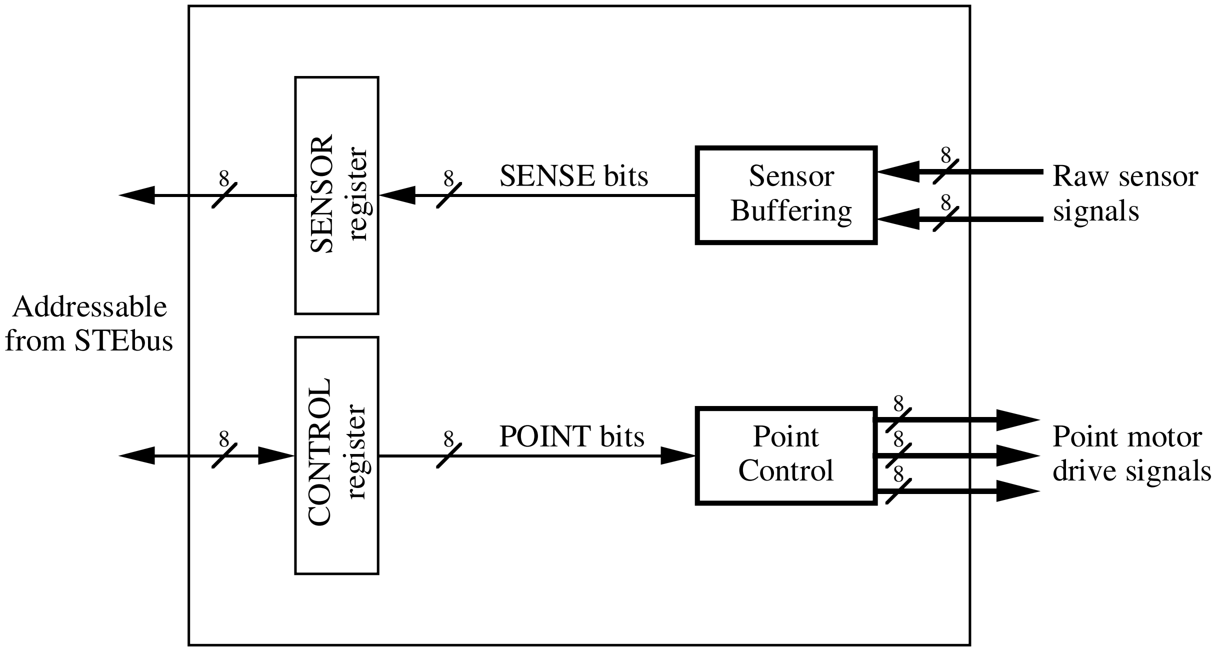


Figure 3.1: Block diagram of the AUCS94.2 Rev 0.1 points control board. The board independently controls eight points motors and buffers eight discrete sensors through two onboard registers.

Discrete Sensor	Postive Pin	Common Pin
1	15	16
2	13	14
3	11	12
4	9	10
5	7	8
6	5	6
7	3	4
8	1	2

Table 3.2: Discrete digital sensor inputs on connector P2

Jumper	Address Line	Address contribution (Hex)
J1	A1	2
J2	A2	4
J3	A3	8
J4	A4	10
J5	A5	20
J6	A6	40
J7	A7	80
J8	A8	100
J9	A9	200
J10	A10	400
J11	A11	800
J12	A12	1000
J13	A13	2000
J14	A14	4000
J15	A15	8000
J16	A16	10000
J17	A17	20000
J18	A18	40000
J19	A19	80000
J20	Enables STEbus memory transfers if removed	
J21	Enables STEbus I/O transfers if removed	

Table 3.3: Effects of address and addressing mode jumpers. All jumpers take effect when removed.

bit in the address used by a program accessing the board. The CONTROL register is addressed when the address on the appropriate address lines matches those set by the jumpers, and the least significant address line (A0) is at logic 0. I.e. it is found at an offset of 0 from the address set by the address jumpers. Similarly, the SENSOR register is addressed when the address lines match the jumper settings, and A0 is logic 1, i.e. at an offset of 1 from the jumpered address. Installed jumpers match address lines at logic 0; removed jumpers match address lines at logic 1.

Addressing mode jumpers J20 and J21, if removed, enable the board to respond to STEbus memory transfers and I/O transfers respectively. If both jumpers are installed, the board will not respond to any STEbus transfers. So, for example, if all jumpers were installed except J2-J7, J16-J19 and J20, the CONTROL register will appear at memory address F00FC hexadecimal, the SENSOR register will appear at memory address F00FD, but neither register will appear in STEbus I/O space. This is summarised in Table 3.3.

The CONTROL register is a read-write register arranged as eight individual bits, each bit controlling a single set of points. Points control is affected by setting the appropriate bit to logic 1 to switch the points one way (nominally left) or to logic 0 to switch them the other way (nominally right). The CONTROL register is read-write, but the value

Register	Adress Offset	Bit	Signal
CONTROL	0	0 (LSB)	POINT1
CONTROL	0	1	POINT2
CONTROL	0	2	POINT3
CONTROL	0	3	POINT4
CONTROL	0	4	POINT5
CONTROL	0	5	POINT6
CONTROL	0	6	POINT7
CONTROL	0	7 (MSB)	POINT8
SENSOR	1	0 (LSB)	SENSE1
SENSOR	1	1	SENSE2
SENSOR	1	2	SENSE3
SENSOR	1	3	SENSE4
SENSOR	1	4	SENSE5
SENSOR	1	5	SENSE6
SENSOR	1	6	SENSE7
SENSOR	1	7 (MSB)	SENSE8

Table 3.4: Hardware register bit assignments

read back only reflects the last value written to it. It is *not* an indication that the points are switched in the directions indicated by the bits in the register. The circuitry must be relied on to successfully switch the points under normal circumstances. Multiple sets of points should not be switched at once, nor should points be switched while there is a train or other object on the points section, as this may prevent successful points switching.

The SENSOR register is a read-only register also arranged as eight individual bits, each bit reflecting the state of the digital input on the corresponding pair of pins on P2. Allowing current to flow from the positive pin to the common pin in a pair will provide a logic 0 input to a bit in the SENSOR register; preventing current flow will provide a logic 1 input. The bit assignments in each register are shown in Table 3.4.

The red fail LED on the front panel is lit under two circumstances. When the STEbus SYSRST* signal goes low, the LED will be lit and will remain lit until this signal goes high. When power to the analogue part of the board fails, the LED will be lit and will remain lit until power is restored. The latter condition is not expected to occur in normal operation, and will be self-resetting if the PTC thermistor (T1) is installed (see circuit description). Either condition will cause the CONTROL register to be cleared, turning power off to all segments controlled by the board.

While the fail LED is lit, any write to the CONTROL register will have no effect: it will remain cleared. Programmers may use this effect to detect for the restoration of power to the analogue circuitry, but are advised that their software should be written to prevent fail conditions occurring at all. A CONTROL register cleared by a fail condition indicates that all points have attempted to switch to the nominally right position. Since there may not have been enough power to have successfully switched some points, all points should be switched individually immediately after a fail condition is cleared to ensure reliable operation thereafter. A fail condition will occur briefly at power-up, making this process a necessary part of initialisation for the board.

An example of how this board may be used in a C programming environment is illustrated by the following code fragments:

```
struct p
{
    int     p_point1:1;   /* State of 1st set of points */
    int     p_point2:1;   /* State of 2nd set of points */
    int     p_point3:1;   /* State of 3rd set of points */
    int     p_point4:1;   /* State of 4th set of points */
    int     p_point5:1;   /* State of 5th set of points */
    int     p_point6:1;   /* State of 6th set of points */
    int     p_point7:1;   /* State of 7th set of points */
    int     p_point8:1;   /* State of 8th set of points */
} *point;

struct s
{
    int     s_sense1:1;   /* State of 1st discrete sensor */
    int     s_sense2:1;   /* State of 2nd discrete sensor */
    int     s_sense3:1;   /* State of 3rd discrete sensor */
    int     s_sense4:1;   /* State of 4th discrete sensor */
    int     s_sense5:1;   /* State of 5th discrete sensor */
    int     s_sense6:1;   /* State of 6th discrete sensor */
    int     s_sense7:1;   /* State of 7th discrete sensor */
    int     s_sense8:1;   /* State of 8th discrete sensor */
} *sense;

#define P_ON            1
#define P_OFF           0
#define P_LEFT          1
#define P_RIGHT         0
#define P_IO_ADDRESS    0x0C00
    .
    .
    .
{
    get_point( P_IO_ADDRESS, &point );
    point.p_point3 = P_LEFT;
    set_point( P_IO_ADDRESS, &point );

    /* A short delay between switching points is advisable */
```

```
    get_point( P_IO_ADDRESS, &point );
    point.p_point6 = P_RIGHT;
    set_point( P_IO_ADDRESS, &point );

    get_sense( P_IO_ADDRESS, &sense );
    if ( sense.s_sense2 == P_ON )
        printf( "Discrete sensor number 2 is activated\n" );
    else
        printf( "Discrete sensor number 2 is not activated\n" );
}

void set_point( short addr, struct t *point )
{
    outportb( addr, *(char *)point );
}

void get_point( short addr, struct t *point )
{
    *(char *)point = inportb( addr );
}

void get_sense( short addr, struct t *sense )
{
    *(char *)sense = inportb( addr + 1 );
}
```

3.2 Circuit Description

The circuit schematics for the points control board are shown as Figure 3.2 and Figure 3.3. The points control board logically (and visibly) comprises two parts: a digital part and an analogue part. The digital part implements an STEbus slave interface to support the two registers described above. The analogue part produces pulses to switch the points and senses presence or absence of current flow through the discrete digital sensor circuitry.

3.2.1 Points Control Board: Digital Part

The digital part of the points control board (AUCS94.2 Rev 0.1) is functionally equivalent to the digital part of the track control board (AUCS94.1 Rev 0.1) in every respect. Both generate a number of SETbus and auxilliary signals. The names and meanings of these signals are summarised in Table 3.5. Any signal suffixed with an asterisk (*) indicates that the signal is "active low", that is, when active (or true or on), the signal assumes logic value 0 rather than logic 1.

The STEbus address, address strobe and data strobe lines are decoded by the three

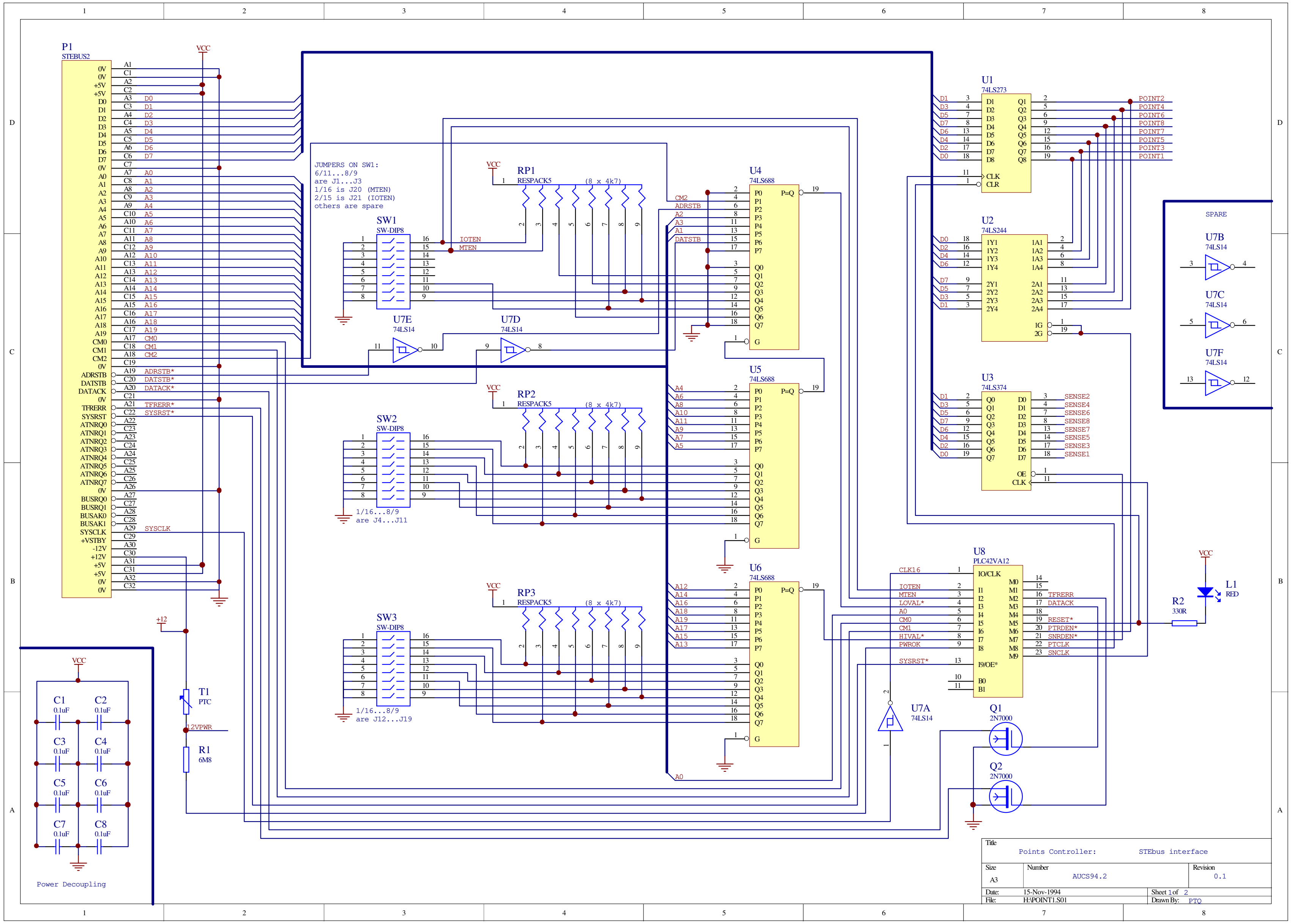


Figure 3.2: Points Control board circuit schematic: page 1 of 2

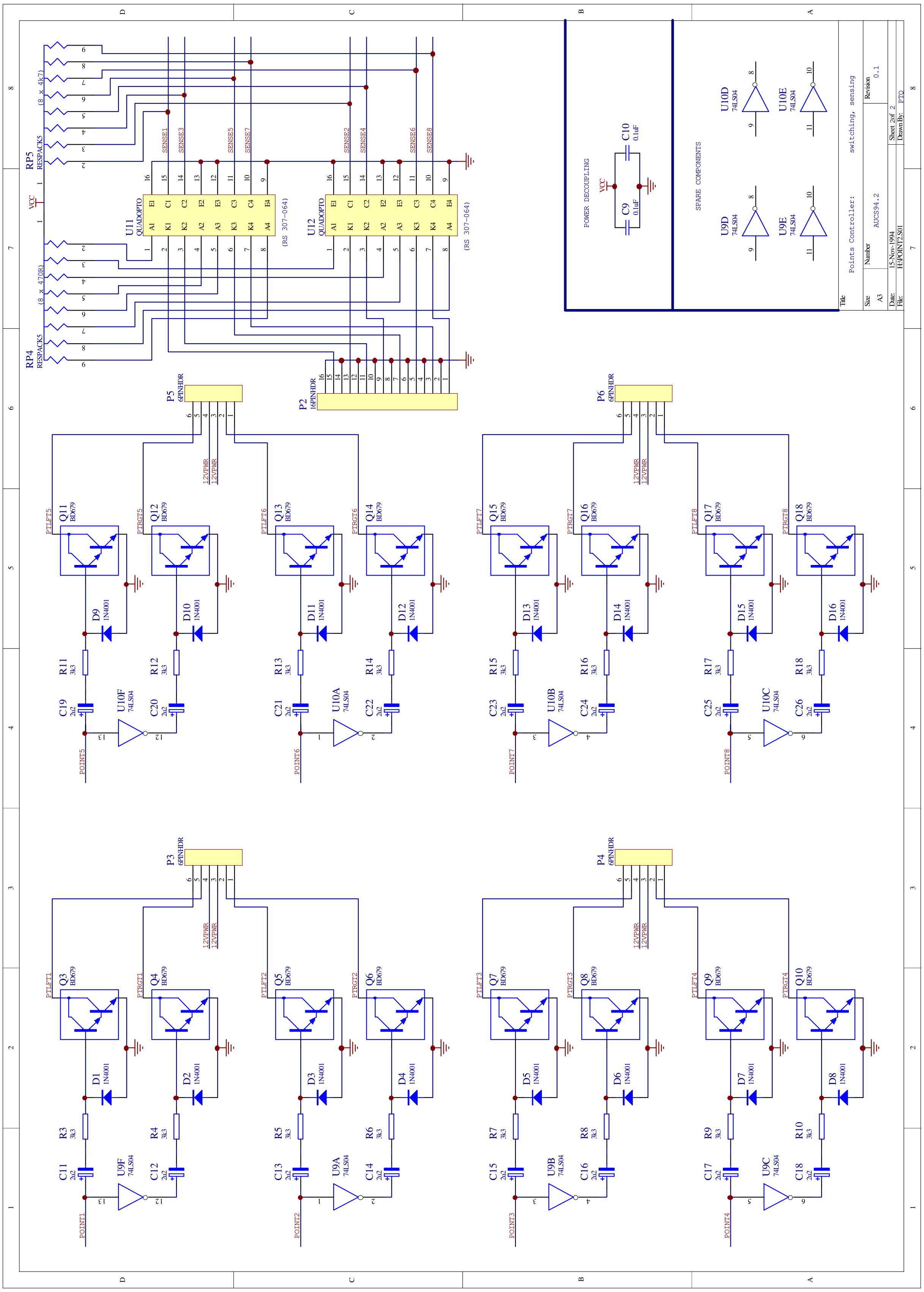


Figure 3.3: Points Control board circuit schematic: page 2 of 2

Signal Name	Meaning
An	STEbus address line n (e.g. A2, A13)
CLK16	A cleaned up version of the 16Mhz STEbus SYSCLK signal
MTEN	STEbus memory transfers are enabled (J20 is removed)
IOTEN	STEbus I/O transfers are enabled (J21 is removed)
CM1	STEbus command modifier bit 1 (memory/io* selector)
CM0	STEbus command modifier bit 0 (read/write* selector)
HIVAL*	When low, the highest 8 STEbus address bits (A12 – A19) are valid
LOVAL*	When low, lower STEbus address bits A1 – A11 (but not A0) and STEbus control signals ADRSTB*, DATSTB* and CM2 are valid
GDCLK	The CONTROL register clock signal
VICLK	The SENSOR register clock signal
GDRDEN*	Enables a read from the CONTROL register (when low)
VIRDEN*	Enables a read from the SENSOR register (when low)
TFRERR	Indicates that an STEbus transfer error has occurred
DATACK	Indicates that an STEbus transfer has completed successfully
RESET*	Points control board reset signal (active low)
SYSRST*	The (active low) STEbus system reset signal
PWROK	The 12 volt power supply to the analogue circuitry is working

Table 3.5: Summary of Points Control Board Signals.

688-style magnitude comparators [1]. One of these (U6) produces the HIVAL* signal indicating the validity of the 8 most significant address bits used only in STEbus memory transfers. The other two (U4 and U5) are daisy chained and produce the LOVAL* signal used in both STEbus memory and I/O transfers to indicate validity [2] of the lower address bits and control signals. Memory and I/O transfers are enabled and disabled independently through jumpers J20 and J21, and give rise to the MTEN and IOTEN enabling signals respectively.

The pair of onboard registers are controlled by two pairs of signals, one for reading the register (the xxRDEN* signal) and one for writing to it (xxCLK signal). Reads from and writes to the CONTROL register are implemented in separate devices. A write is latched into a 273-style octal latch (U1) by the rising edge of the PTCLK signal. A read passes a copy of the value in U1 back to the STEbus by enabling the outputs of a 74LS244 tri-state buffer (U2) when the PTRDEN* signal goes low.

The SENSOR register is implemented in a single 74LS374 tri-state octal D-type flip-flop (U3). A read from it enables its outputs onto the STEbus when the SNRDEN* signal goes low. Data from the analogue part of the board is latched into U3 by the rising edge

[1] By "688-style" I mean a 74LS688 or 74HCT688 chip. See the closing paragraph in this section for a brief discussion of the constraints on the types of logic chips which may be used.

[2] "Validity" means that the address bits match the board's address as configured on J1-J19 (for both LOVAL* and HIVAL*), and the control signals are requesting a valid data transfer (explicit in the LOVAL* signal only).

of the SNCLK signal. A write to the SENSOR register from the STEbus has no effect due to the programming of U8 described below.

The PLC42VA12 programmable logic device (U8) produces the four register control signals just mentioned, and indirectly drives the two STEbus acknowledge lines DATACK* and TFRERR* via the FETs Q1 and Q2. It is programmed with the following logic equations written in the notation of the Philips SLICE(tm)package environment [2] (viz. / represents inversion, + represents logical OR, * represents logical AND). Active low signals on the schematic and in Table 3.5 which are suffixed with an asterisk (*) are instead prefixed with the letter N, since the asterisk is used as an operator by SLICE(tm). Two internal flip-flops (FF0 and FF1) are used to create a delay for the DATACK signal during a read cycle. The other signals require only combinational logic.

```
FF0.CLK   = CLK16;
FF0.D     = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL );
FF1.CLK   = /CLK16;
FF1.D     = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL );
PTCLK    = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*/CM0*/A0;
SNCLK    = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*CM0*A0;
NPTRDEN  = /( /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*CM0*/A0 );
NSNRDEN  = /( /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*CM0*A0 );
TFRERR   = /NLOVAL*( /IOTEN*/CM1 + /MTEN*/NHIVAL*CM1 );
DATACK   = /NLOVAL*( IOTEN*/CM1 + MTEN*CM1*/NHIVAL )*/CM0 + FF0*FF1*CM0;
NRESET   = PWROK*NSYSRST;
```

To be noted from these equations is that the SNRDEN* and SNCLK signals are active under the same conditions, so that a read to the SENSOR register simultaneously latches the data into U3, and drives the STEbus data lines with it. Alternatively, the SNCLK signal could have been programmed analogously to the PTCLK signal, so that data could be independently latched into the SENSOR register by writing a dummy value to it. No particular advantage was seen in doing this.

Of the components used, U8 is a CMOS device which drives TTL outputs, a fact which may be some cause for concern — it does not fall within STEbus standards for voltage and current. The other integrated circuits used must be Low Power Schottky TTL (LSTTL) devices if used to drive the STEbus, or can be either LSTTL or High Speed TTL-compatible CMOS (HCT) devices if they only receive STEbus signals. Although the Schmitt Trigger Inverter (U7) is used only to receive some signals from the STEbus, the SYSCLK signal is among them. This signal (produced by the SCIM88) is thought unlikely to be able to reliably switch an HCT device, hence the use of LSTTL.

3.2.2 Points Control Board: Analogue Part

The analogue part of the points control board comprises eight sets of components for controlling sets of points, circuitry for supporting the digital inputs, and one fail indicator LED. This LED (L1) is driven through a resistor (R2) from U8, and will light in the event of a failure of the +12 volt power supply (12VPWR) to the analogue part (lit until power is restored) or by an STEbus system reset (lit for as long as SYSRST* stays low).

The PTC thermistor (T1), if available, is designed as a protection against excessive points switching currents being drawn, such as when multiple sets of points are accidentally or deliberately switched simultaneously. Excessive current drawn from the +12 volt power supply to the analogue part will cause T1 to heat up, increasing its resistance, and shutting down the power supply. In turn, this will cause the PWROK input to U8 to go low, driving the RESET* signal low, turning on the fail LED and clearing the CONTROL register as a result. T1 will cool of its own accord, restoring +12 volt power, driving RESET* high again and extinguishing the fail LED. Note that points may not be in the state indicated by the CONTROL register bits after such a failure.

Each set of points has two driver circuits: one for switching the points to the nominally left position, and one for switching it to the right. These circuits are identical and cause a decaying pulse of current to flow through the points coils when the corresponding bit in the CONTROL register changes from logic 0 to logic 1 (switch left) or from logic 1 to logic 0 (switch right). These individual points state bits are reflected by points signals which when switching left are used to charge one capacitor (e.g. C11 for points set 1) and simultaneously discharge another (C12) through a diode (D2). When switching right, C12 is charged by the inverted points signal while C11 discharges through a diode (D1). The charge and discharge rates are set by resistors (R3, R4). The current charging the capacitors is amplified by the respective transistors (Q3, Q4) producing the decaying pulse which drives the points motors.

The digital sensor inputs are implemented by two quad optoisolator devices (U11 and U12), biased by two resistor packs (RP4 and RP5). Short circuiting, or otherwise allowing current to flow, from the positive pin to the common pin of a pair on connector P2 completes the diode circuit in the optoisolator and drives its transistor circuit into saturation, giving a low level on the sense line into U3. Leaving the diode circuit open or off cuts off the transistor circuit, giving a high level on the corresponding sense line.

3.2.3 Component Layout and Connectors

Figure 3.4 shows the component overlay for the points control board. The numbering of the pins on connectors P1 to P6, as seen with the component side of the board facing upwards and from the perspective of the female connectors which connect to them, are shown in Figure 3.5, Figure 3.6 and Figure 3.7.

3.3 Deviations from Prescribed Design

Some modifications have been made to the circuit boards in use. They are:

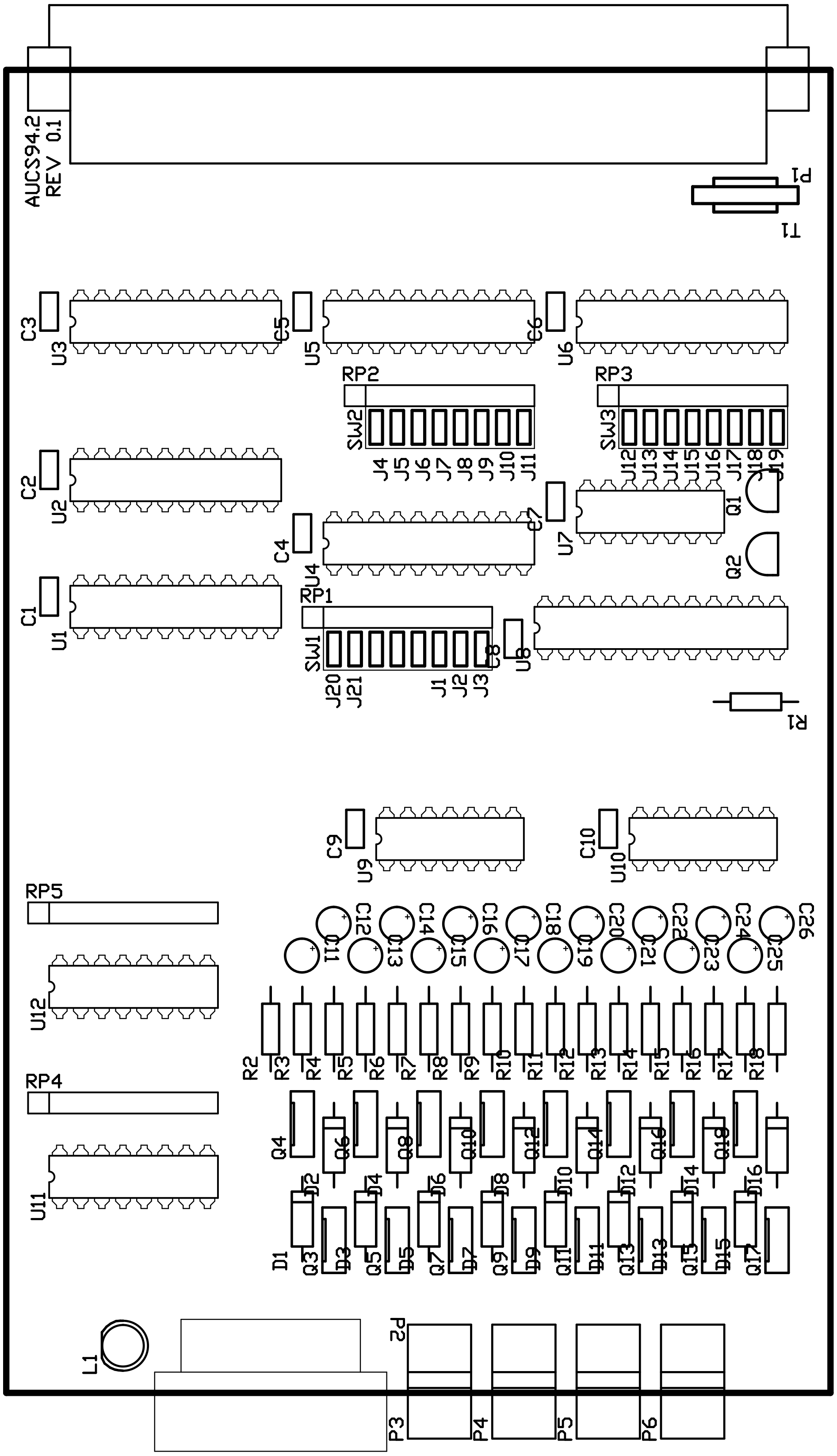


Figure 3.4: PCB component overlay for Points Control Board

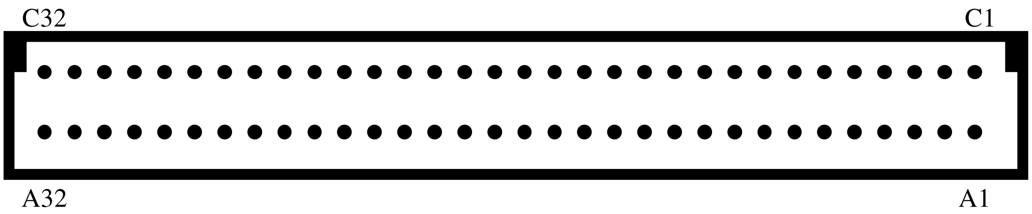


Figure 3.5: Pin assignments on STEbus connector P1

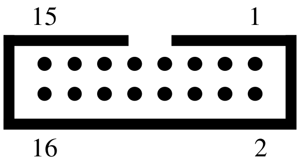


Figure 3.6: Pin assignments on digital input connector P2

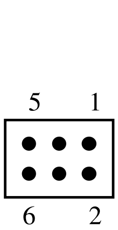


Figure 3.7: Pin assignments on points drive connectors P3, P4, P5 and P6

- The SYSCLK signal has been put through an extra inversion by passing it first through the spare inverter between U7.13 and U7.12 before it goes into the input of the inverter U7.1.
- The PTC thermistor T1 is unavailable in New Zealand (but can be acquired in Europe, for example: Philips part 2322 661 54111). A wire link has been substituted in its place. A more appropriate substitution would be to use a PNP transistor with its emitter connected to the STEbus +12V supply, its collector providing the 12VPWR supply, and its base connected via a resistor to 0 volts. An appropriate choice of resistor will make this circuit a current limiter, something far more appropriate to the needs of the analogue part. The PCB has been altered to allow this modification, but a suitable transistor has not been sourced.
- A large capacitor (4700μF) has been placed between the 12VPWR supply and 0 volts to supplement points switching current. If used in conjunction with the suggested current limiter, and the CONTROL register reads cleared rather than the last value written, this will be an indication that excessive switching was attempted, and the last sets of points switched should not be trusted to be in their indicated positions.

3.4 Known Performance Problems

No performance problems are known, but there is no diode protection for any of the transistors (Q3 – Q18) driving the points coils. However, due to the decaying current pulse generated by the circuitry, reverse voltage spikes may be reduced or eliminated. Until these boards have been in use for a significant period of time, it will be impossible to tell if this is true or not.

3.5 Advice for Future Revisions

The following points should be kept in mind when designing future revisions of this board:

- Address the deviations made from the prescribed design after the printed circuit boards were made.
- Address the performance problems outstanding with this revision.
- The entire functionality of the digital part of the circuit can be programmed into modern programmable logic devices (e.g. from XILINX). Provided the STEbus specifications are respected, which may require some additional Schmitt trigger devices between the bus and the programmable device, using such a device may result in a dramatic space saving on the board.

Bibliography

[1] ANSI/IEEE. *IEEE Standard for an 8-Bit Backplane Interface: STEbus*, 1988.

[2] Philips Components-Signetics. *SLICE: Signetics Logic Integration Computer Environment*, 1990.

[3] Intel Corporation and Zilog Inc. *Functional Descriptions of the Intel 80C188 Microprocessor and the Z85C30 Serial Communications Controller as used in the Arcom SCIM88 Processor Board*, 1994.

[4] Borland International Inc. *Turbo Debugger 3.0 User's Guide*, 1991.

[5] Arcom Control Systems Ltd. *SPCOM: STE PC Comunications Board User's Manual*, 1991.

[6] Arcom Control Systems Ltd. *SourceTASK: Multi-tasking Executive for 80188 Target Systems User's Manual*, 1992.

[7] Arcom Control Systems Ltd. *SVTP: SourceVIEW Turbo Pascal User's Manual*, 1992.

[8] Arcom Control Systems Ltd. *SCIM88: STEbus 80C188 CPU Board with SCIM Interface User's Manual*, 1993.

[9] Arcom Control Systems Ltd. *The Catalogue 5*, 1994.

[10] Arcom Control Systems Ltd. *SVBC: SourceVIEW Borland C User's Manual*, 1994.

[11] R. J. Mitchell. *Microcomputer Systems Using the STEbus.* Macmillan, 1989.

[12] Michael Tooley. *Bus-based Industrial Process Control.* Heinemann Newnes, 1988.