# SC88T

## STE 80188 CPU

Technical Manual

## Contents

# Revision History

| Manual | PCB | Comments | |
|---|---|---|---|
| V1 Iss 2 | V1 Iss 2 | 870414 | First Published in this format. |
| Addendum | | | |
| Issue A | V2 Iss 3 | 970818 | ECO2630. |
| Issue B | V2 Iss 3 | 971103 | ECO2674 |



The choice of board or systems is the responsibility of the buyer, and the use to which they are put cannot be the liability of Arcom Control Systems Ltd. However, Arcom's sales team is always available to assist you in making your decision.

### Product Information

Full information about other Arcom products is available via the Fax-on-Demand System, (Telephone numbers are listed below), or by contacting our Website at: www.arcom.co.uk, *or for the US,* www.arcomcontrols.com

### Additional useful contact information:

**Customer Support**: (tel) +44 (0)1223 412428, (fax) +44 (0)1223 403400, (email) support@arcom.co.uk
**Sales**: (tel) +44 (0)1223 411200, (fax) +44 (0)1223 410457, (email) sales@arcom.co.uk, *or for the US*, icpsales@arcomcontrols.com

**United Kingdom**
Arcom Control Systems Ltd
Clifton Road
Cambridge
CB1 4WH
UK
tel: +44 (0)1223 411200
fax: +44 (0)1223 410457
FoD: 01223 240600

**United States**
Arcom Control Systems Inc
13510 South Oak Street
Kansas City, MO 64145
USA
tel: (toll free) 888-941-2224
fax: 816-941-7807
FoD: 800-747-1097

**France**
Arcom Control Systems
Centre d'affaires SCALDY
23, rue Colbert
7885 SAINT QUENTIN
Cedex, FRANCE
tel:(Numero Vert) 0800 90 84 06
fax:(Numero Vert) 0800 90 84 12
FoD:(Numero Vert) 0800 90 23 80

**Germany**
(Kostenlose Infoline:)
tel: 0130 824 511
fax: 0130 824 512
FoD: 0130 860 449

**Italy**
(Numeroverde:)
FoD: 1678 73600

**Belgium**
(Groen Nummer:)
tel: 0800 7 3192
fax: 0800 7 3191

**Netherlands**
(Gratis 06 Nummer:)
tel: 06022 11 36
fax: 06022 11 48

# Section 1. Introduction

The SC88T is a powerful and expandable single-board computer, based on the Intel 80188 microprocessor. It is designed as a CPU board for CPU in a large control system. It uses the industry-standard STEbus, so a large range of peripheral boards are available.

Software allows programs written in Borland C to be put into an EPROM and run on the SC88T without disks, and support for development on a PC. Contact us for details.

The STEbus interface is an important feature of the SC88T. The bus is being ratified by the IEEE (IEEE designation is P1000), and it is designed as a processor- and manufacturer-independent, asynchronous, multiprocessing bus. The asynchronous nature of the bus means that a slave board (memory or I/O) must acknowledge commands by the SC88T, which will wait until it does, so that any speed of peripheral board can be accessed by any speed of CPU board. The multiprocessing ability lets you run several CPU boards on the same bus. This is explained in greater detail later on.

# Section 2. Circuit Description

The 16MHz oscillator module drives the 80188 CPU (IC12) and optionally the STEbus system clock. The 8MHz output clock from the CPU is halved by IC11, to produce the 4MHz clock to drive the SCC (IC13). The SCC is wired to connector PL3 via RS232 buffer IC18 and IC19.

IC9 is a logic array which detects bus accesses and generates wait states, based on the address, the state of the on-board chip selects and the CPU status. IC8 latches the top 4 bits of the CPU address bus and IC7 latches the lowest eight bits. IC11 controls chip-selects for the RAMs (IC15,17) and generates a one bit output port for use as an attention request on the STEbus.

The EPROMs (IC14 and 16) can be 8, 16, 32 or 64k each and there are 2 jumpers to set the EPROM size.

Address and data lines on the bus are driven by IC1,2,4 and 5, with IC5 driving the strobes. Incoming bus signals are buffered by IC3 and 6. Reset can come in from the bus (via LK5A) to reset the CPU, or it can go to the bus from the on-board reset circuit, in which case a reset switch can be wired between the pins of connector PL2.

# Section 3. STEbus Interface

The SC8TT has a STEbus interface with the IEEE number P1000. Here is a summary of the features of the bus with some notes on their implementation on the SC88T board. The pinout of the bus is given in Appendix B.

Table 1. STEbus interface on the SC88T

| Signal | In-Out | Type | Description | Implementation |
|---|---|---|---|---|
| A0 - A19 | 0 | 3s | 20-bit memory address | Buffered from the CPU 20-bit address. |
| A0 - 11 | 0 | 3s | 12-bit I/O address | As above. |
| A0 - 2 | 0 | 3s | 3-bit acknowledge address | The SC88T does not handle bus-vectored interrupts, so it does not generate acknowledge cycles. It does respond to bus non-vectored interrupts on lines ATNRQ0*-ATNRQ3* if they have been appropriately jumpered. |
| D0 - 7 | I/O | 3s | 8-bit data bus | Driven by the CPU on write cycles, or by the slave on read cycles. |
| ADRSTB* | 0 | 3s | Address Strobe | Addresses, data and command modifiers are valid before the falling edge of ADRSTB* and DATSTB*. Both strobes are active at the same time on the SC88T. |
| DATSTB* | 0 | 3s | Data Strobe | On the write cycle, valid data and CM0 are present before this is asserted. On a read cycle, CM0 height is present before it is asserted and DATSTB* then indicates to the slave that the CPU is ready to accept |
| CM2<br>CM1<br>CM0 | 0<br>0<br>0 | 3s<br>3s<br>3s | | Command modifiers indicating the type of bus cycle, according to the following table:<br>*CM 2 1 0*<br>*0 0 0* \<br>*0 0 1* } *reserved*<br>*0 1 0* /<br>*0 1 1* *acknowledge*<br>*1 0 0* *I/O write*<br>*1 0 1* *I/O read*<br>*1 1 0* *memory write*<br>*1 1 1* *memory read*<br>(Note: the acknowledge cycle is NOT generated by the SC88T.) |

Table 1. (continued)

| Signal | In-Out | Type | Description | Implementation |
|---|---|---|---|---|
| BUSRQ0-1* | 0 | o/c | Bus requests | Potential (temporary) bus masters request the bus from the arbiter on either of these lines. BUSRQ0* has a higher priority than BUSRQ1*. The SC88T acts as temporary master, and can be jumpered to ignore arbitration as a single master. |
| BUSAK0-1* | I | in | Bus Acknowledge | The arbiter acknowledges a request from either of the two potential masters on these lines. A potential master may only drive the bus when it has received an acknowledge on the line corresponding to its request. |

The SC88T can be jumpered either to act as a potential master which requests the bus from the permanent master or a separate arbiter, or it can be set up to ignore arbitration. If the SC88T is set to ignore arbitration it must be the only master on the bus. If you wish to use more than one master on the bus, then you must have one and only one arbiter. A suitable arbiter is available on the SYSCON board. See 'Links and Options' for details of how to set up the board.

Key: * = signal is active low
3s = tri-dtate
tp = totem-pole
o/c = open-collector

It is quite easy to use the SC88T on the STEbus. You will need a terminated backplane, and one or more slave boards, such as A/D converters, for example. In order to comply fully with the specification of the impedance of each backplane line should be 60 ohms ±10%. However, a short backplane is not likely to cause any malfunction even if its impedance varies considerably from this. A terminator is necessary because some of the lines are open-collector, and timing is critical on the strobes.

A SC88T configured as standard will generate all necessary bus signals. All that is required to generate a bus access is that you try to read from or write to a memory or I/O location which the on-board logic defines as on the bus. See the 'Memory Map' and 'I/O Devices' sections for details on which addresses are on-board and which are not.

**Note:** The internal timers on the SC88T can be programmed so that if the slave board which you are attempting to access does not respond, for example if you used the wrong address, a bus timeout will be generated. This will release the SC88T from its wait-state and generate an interrupt.

See section 7. for more details.

If the SC88T 'HANGS' on a bus access, or a bus timeout occurs, check the following:

Are the address and data lines enabled ('HUNG' state)?

> If they are not, the SC88T has not gained access to the bus. This may be because some other master already has access to the bus and has not released it. Check the state of the BUSRQ* and BUSAK* lines, and jumpers LK2C and D.
>
> If they are, check that DATACK* or TFRERR* has been received.

If DATACK* or TFRERR* have not been received (Bus timeout)

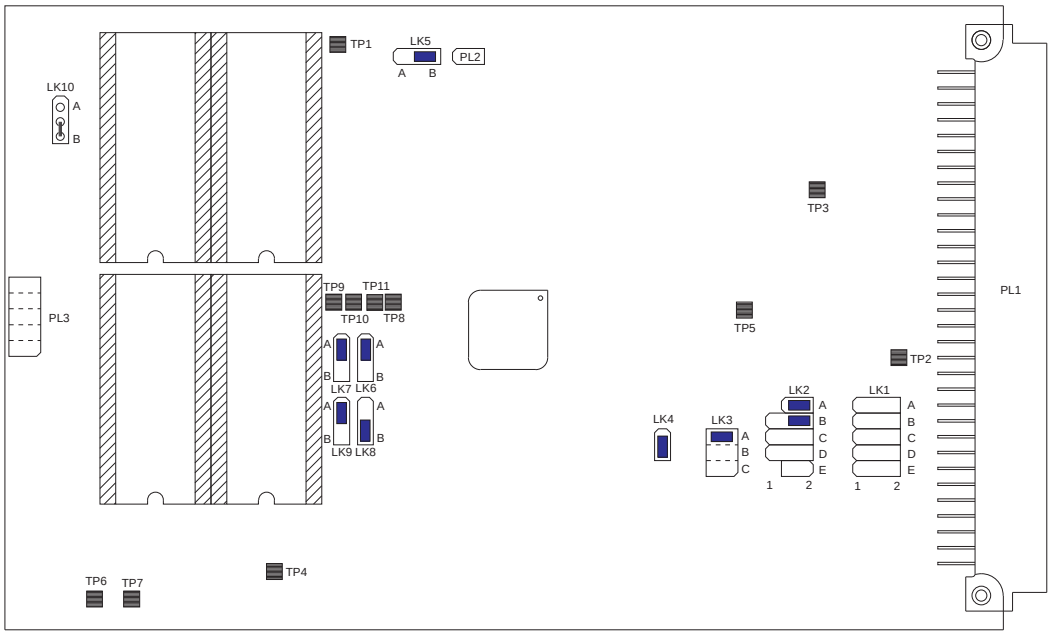
> Check that the slave board can respond to the bus access which the SC88T sent out.
>
> If it can, check the SYSCLK is present on the bus, as many slave boards require this for timing, and check also that SYSCLK is coming from only one source, if you have more than one CPU on the bus.

Check also that your software has allowed bus timeouts to occur, and that it can deal with interrupts from bus timeouts if they happen. It may sometimes be useful to disable bus timeouts and check the bus in the 'HUNG' state if you are confused.

# Section 4. Links and Options

There are many different ways in which you can configure the SC88T; they are selected by jumpering across the links on the board. A '+' indicates a default link setting. The board is described as seen from the component side of the board, with the 64-way bus connector to the right.

Figure 1. Link Positions




**Link area 1 Attention Requests in and out**

| | |
|---|---|
| LK1A1 | /ATOUT to ATNRQ1* |
| LK1A2 | ATNRQ1* goes to CPU INT0 |
| LK1B1 | /ATOUT to ATNRQ2* |
| LK1B2 | ATNRQ2* goes to CPU INT1 |
| LK1C1 | /ATOUT to ATNRQ3* |
| LK1C2 | ATNRQ3* goes to CPU INT3 |
| LK1D1 | /ATOUT to ATNRQ5* |
| LK1D2 | ATNRQ5* goes to CPU DRQ0 |
| LK1E1 | ATNRQ7* goes to CPU DRQ1 |
| LK1E2 | ATNRQ6* goes to CPU DRQ1 |
| LK1DE2 | ATNRQ6* goes to CPU DRQ0 |

Figure 2.
Link area 1.

o A1 o A2 o

o B1 o B2 o

o C1 o C2 o

o D1 o D2 o
DE2
o E1 o E2 o

## Link area 2 Reset, Bus Requests/Acknowledges, Clock.

| | |
|---|---|
| + LK2A | SYSRST* to bus from this board |
| LK2B1 | Accept bus acknowledges from external arbiter |
| + LK2B2 | Ignore arbitration, single master only |
| LK2C1 | Bus requests on BUSRQ1* |
| LK2C2 | Bus requests on BUSRQ0* |
| LK2D1 | Bus acknowledges on BUSACK1* |
| LK2D2 | Bus acknowledges on BUSACK0* |
| LK2E | SYSCLK from this board |

Figure 3.
Link area 2.

o A o
o B1 o B2 o
o C1 o C2 o
o D1 o D2 o
o E o

**Notes:**
If LK2B2 is made, the state of LK2C and D is irrelevant.
If LK2B2 is made, an external arbiter cannot be used, and the SC88T must be the only master on the STEbus.
The request and acknowledge must be on the same level if an external arbiter is used. This means that if LK2C1 is made, LK2D1 must also be made. Alternatively, if LK2C2 is made, LK2D2 must also be made.

## Link area 3 Non-Maskable Interrupts (NMIs)

| | |
|---|---|
| + LK3A | NMI disabled |
| LK3B | NMI from ANRQ0* |
| LK3C | NMI from TFRERR* |

Figure 4.
Link area 3.

o A o
o B o
o C o

## Link area 4 RAM size

| | |
|---|---|
| + LK4 open | 8k RAM chips installed (e.g. 6264 or similar) |
| LK4 made | 32k RAM chips installed (e.g. 55257 or similar) |

Figure 5.
Link area 4.

o
o

## Link area 5 Board reset

| | |
|---|---|
| LK5A | This board is reset from the STEbus |
| + LK5B | This board is reset from a switch across PL2 |

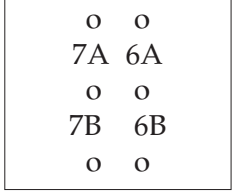
Figure 6.
Link area 5.

o A o B o

## Link area 6 and 7 EPROM pins

| | |
|---|---|
| + LK6A | EPROMs pin 1 to +5V (all except 27512) |
| LK6B | EPROMs pin 1 to A15 (27512 only) |
| | |
| + LK7A | EPROMs pin 27 to +5V (2764 & 27128) |
| LK7B | EPROMs pin 27 to A14 (27256 & 27512) |

Figure 7. Link areas 6 and 7

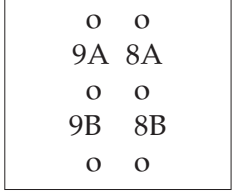



## Link area 8 and 9 RAM pins

| | |
|---|---|
| LK8A | RAM power is from STEbus VSTBY line |
| + LK8B | RAM power is from on-board +5V |
| | |
| + LK9A | RAM pin 26 to +5V (8k RAMs or VSTBY) |
| LK9B | RAM pin 26 to A13 (32k RAMs) |

NB: LK9 Dependant on LK8

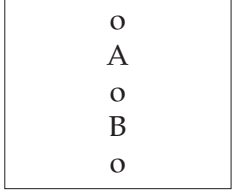
Figure 8. Link areas 8 and 9




## Link area 10 Serial Input level on channel A

| | |
|---|---|
| LK10A | Serial input is at TTL levels |
| + LK10B | Serial input is at RS232 levels |

Figure 9. Link area 10




***Note**: This link is hardwired in the B position. You must take care if the A position is used: If RS232 levels are input in the A position, SCC will be destroyed.*

# Section 5. I/O Devices

The chip-selects for I/O devices are generated by the 80188, and it is vital to set the 80188 up in software as part of the initialisation sequence. If this is not done the board will not function correctly. See section 7 for information on how to do this.

There are three types of I/O devices which the 80188 can access.

1. I/O boards on the STEbus

The STEbus 4k I/O space is accessed in the lower 32k of the 80188's 64k I/O address space. The selection is automatic, and the programmer should not relocate any of the programmable I/O chip selects in this space.

2. The on-board SCC for serial communications

The on-board SCC is controlled by two programmable chip-selects. PCS1 is the chip-select which accesses the registers. Accessing PCS2 will assert INTA*, the SCC interrupt acknowledge input. This must be done to acknowledge the SCC interrupt generated on INT2 of the 80188, because the 80188 in its most useful interrupt mode does not generate an acknowledge. A read from PCS2 will read the interrupt vector from the SCC and reset the SCC ready for the next interrupt.

3. The 80188 on-chip peripherals (counter/timers, DMA etc)

These integrated peripherals (and the address of PCS1 and PCS2) are controlled by the Internal Control Block. Even the I/O base location of this block of peripherals is controlled by the relocation register within the block. It is recommended that you do not alter this block unless you are sure what you are doing.

## Internal block control register map

After a reset, the 256-byte Internal Control Block base address will be at FF00 (Hex) in I/O space. The individual control registers reside within the block, at fixed-offsets from the base address. Each register is 16-bits wide.

| Base +offset | | Control Block register |
|---|---|---|
| FF00 +20 to +3E | | Interrupt Controller |
| +50 to +56 | | Timer 0 Control |
| +58 to +5E | | Timer 1 Control |
| +60 to +66 | | Timer 2 Control |
| +A0 | UMCS | Upper Memory Chip Select |
| +A2 | LMCS | Lower Memory Chip Select |
| +A4 | PACS | Peripheral Base Address Control |
| +A6 | MMCS | Middle Memory Chip Select |
| +A8 | MPCS | Middle Peripheral Chip Select |
| +C0 to +CA | | DMA Descriptors Channel 0 |
| +D0 to +DA | | DMA Descriptors Channel 1 |
| +FE | | Relocation register |

Note that programming the control block is not a trivial task. More detailed information on how to do this may be found in the Intel 80188 data book. This manual covers values for most useful memory maps for the SC88T.

If PACS is loaded with OFBE and MPCS is loaded with 80B8 then the Peripheral Chip Select map appears as below.

| Base +offset | | Peripheral Chip Select I/O map |
|---|---|---|
| F800 +000 | PCS0 | not used |
| +080 | PCS1 | SCC channel B control |
| 081 | | SCC channel A control |
| 082 | | SCC channel B data |
| 083 | | SCC channel A data |
| +100 | PCS2 | Read to generate INTA* and fetch vector from the SCC (i.e. the interrupt acknowledge address) |
| +180 | PCS3 | Writing a byte with D0 = 1 asserts /ATOUT (low) Do not read, as D0 will not be predictable. |
| +200 | PCS4 | not used |
| +280 | PCS5 | not used |
| +300 | PCS6 | not used |
| +380 | PCS7 | not used |

# Section 6. Memory Map

The 80188 CPU has several programmable chip-select lines, which are asserted when the CPU address falls within the chip-select boundaries defined in the CPUs internal chip select registers. Upon reset, the 80188 asserts the Upper Memory-Chip Select (UCS) to select EPROM1 (IC16), and jumps to location 0FFFF0H. The first instructions must be to re-set the Upper Memory Chip Select register (UMCS) as in the table below, then jump to some code lower in (IC16). This code can then set up the MMCS and MPCS registers, by which time both EPROMS will be properly selected. An example of this is given at the end of the assembly language program in Appendix D.

If EPROM0 (IC14) is not to be used then it will not be enabled if the MMCS register is left unaccessed. If IC14 is to be used, then using the values in the table below , EPROM0 will occupy memory directly below IC16, forming a contiguous block. This also imposes the restriction that EPROM0 and EPROM1 must be the same size.

Because UCS overlaps MCS3, the number of programmable wait states must be identical for the UCS and MCSO-3 . Two wait-states will be required by slow EPROMs. If STEbus memory overlaps MCS1 and MCS0 then MCS0-3 must accept externally generated wait-states until DATACK* is received from the bus . Both the above conditions are satisfied by the contents of the table below. Slow EPROMs are those of 250ns access time and above. Note that Arcom use two wait-states in their initialisation code. This allows the use of any EPROM currently available.

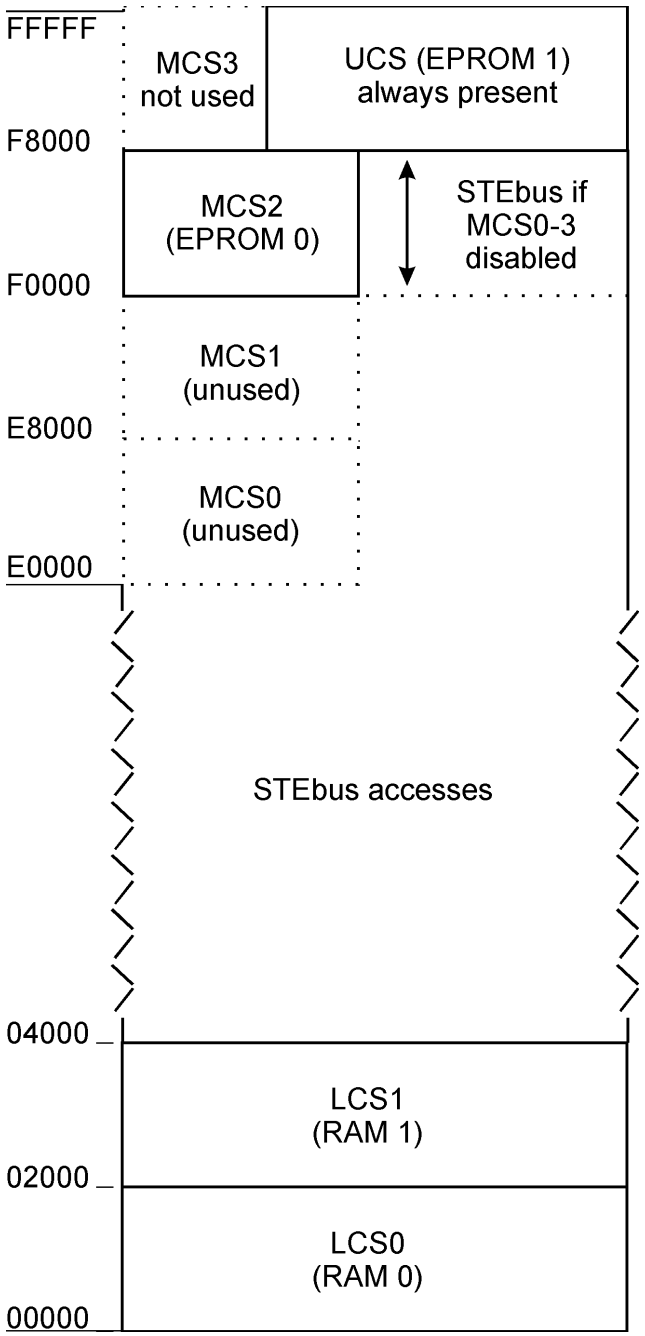
The RAM is selected by the Lower Memory Chip-Select (LCS), which may be set from 8k (one 8k RAM) to 64k (two 32k RAMs). Wait-states are not usually required for fast static RAMs.

If a memory access is made to an address which is within neither UCS, MCS2 or LCS, the STEbus will be selected automatically. This may result in a bus timeout if no actual memory exists on the bus. The tables below contain chip select register values for various RAM and EPROM sizes. EPROM IC16 and IC14 must be the same size.

| Total EPROM capacity | Individual EPROM type, size | Register contents (hex) | | |
|---|---|---|---|---|
| | | UMCS | MMCS | MPCS |
| 16k | 2764 8k | FE3E | F802 | 8438 |
| 32k | 27128 16k | FC3E | F002 | 8838 |
| 64k | 27256 32k | F83E | E002 | 9038 |
| 128k | 27512 64k | F03E | C002 | A038 |

| Total RAM | IC15 | IC17 | LMCS |
|---|---|---|---|
| 8k | 6264 | ---- | 01F8 |
| 16k | 6264 | 6264 | 03F8 |
| 32k | 55257 | ----- | 07F8 |
| 64k | 55257 | 55257 | 0FF8 |

Example memory map using two 32k EPROMs. If EPROM 0 is not used, and MCS0-3 disabled, then STEbus accesses may continue up to the base address of EPROM1.




The memory blocks of the 80188 cannot be mapped at any arbitrary location. The Middle Memory Chip-Selects are all the same size, and must be mapped at a base address which is a multiple of the middle memory block size. Hence this is the only way of creating a chip select area contiguous with UCS.

MCS0, MCS1 and MCS3 are not used. To ensure that they do not interfere with STE memory space, set UCS and MCS0-3 to include externally -generated wait states. (So that accesses to memory in the MCS0 and MCS1 address ranges wait for the DATACK* signal from the STEbus.)

Example shown with two 8k RAMs (as supplied) plus two 32k EPROMs.

# Section 7. Using the SC88T

Because the SC88T can be configured in so many different ways, it is difficult to make any general recommendation about using it, mainly because the software will determine how it is used. There are two obvious situations where it can be employed, and a summary of them may suggest other possibilities.

## As a target system

Here the SC88T runs software created on another computer, which is transferred either by blowing an EPROM or by downloading machine-code into RAM. A useful approach is to have a machine-code monitor in the EPROM socket, so that the code can be debugged.

If you want to start from scratch the following notes may help.

The chip-select for EPROM (UCS) should be set up first, so that you can jump to the start of EPROM and then continue initialising.

The LCS is required before any stack or memory operations are done with the on-board memory.

The peripheral chip-selects must be programmed to allow access to the on-board SCC.

Timers 0 and 1 may be programmed to produce a bus timeout function. The input to T0 is the bus driver enable line; T0 output is wired to T1 input, and T1 output going low will take the CPU out of the wait-state. For this to happen it is necessary to set T0 up as a free-running timer which is stopped by the active-low bus-driver enable signal: T1 must be set up to be retriggered by pulses on its input (which is connected to T0 output). If T1 then fails to be retriggered to T0 it will time out. It can also generate an interrupt when it times out so that the CPU is aware that a timeout has occurred.

The SCC requires some initialisation code, and it is strongly recommended that you obtain the SCC Technical Manual before doing anything unusual.

If 80188 instructions are used in 'test and set' operations for communication between processors on the bus, they should be prefixed with the LOCK prefix. This will prevent the bus request from going away until the instruction is complete.

# Appendix A. Component List

| | | | |
|---|---|---|---|
| LK10B | 1 | Cable Misc, Solid Wire | Tin Copper 22swg |
| PL1 | 1 | Conn Multi, 41612 | 64PCB 964-053 |
| LK1-1 (A-E> | 0.2 | Conn Multi, Inter-PCB | Bergstick 1 row |
| LK2-1 (B-D) | 0.1 | Conn Multi, Inter-PCB | Bergstick 1 row |
| LK4-9 | 0.5 | Conn Multi, Inter-PCB | Bergstick 1 row |
| PL2 | 0.1 | Conn Multi, Inter-PCB | Bergstick 1 row |
| MISC | 13 | Conn Multi, Inter-PCB | Bergjump |
| LK1-2 (A-E) | 0.2 | Conn Multi, Inter-PCB | Bergstick 2 row |
| LK2-2 (A-E) | 0.2 | Conn Multi, Inter-PCB | Bergstick 2 row |
| LK3 | 0.1 | Conn Multi, Inter-PCB | Bergstick 2 row |
| PL3 | 0.2 | Conn Multi, Inter-PCB | Bergstick 2 row |
| TP1-11 | 11 | Conn Terminals/Test Lead | PCB Test point loop |
| MISC | 2 | Fastening, Rivet | Brass Hot Tin |
| IC14-17 | 4 | Socket IC TH, Misc | 28 Pin TP |
| XM1 | 1 | XTAL Th, Module | 16 MHz Half Size |
| C2, 3, 33, 34 | 4 | CAP SM, TANT | 22U 16V 10% CASE-D |
| C35 | 1 | CAP SM, TANT | 3U3 16V 10% CASE-B |
| C36, 37 | 2 | CAP SM, TANT | 10U 16V CASE-B |
| IC13 | 1 | IC Interface SM, Periph | 8530 SCC 6MHz ZILOG PLCC |
| IC18 | 1 | IC Interface SM, Periph | 14C89 |
| IC19 | 1 | IC Interface SM, Periph | 14C88 Texas Only |
| IC12 | 1 | IC Processor Th, STD | 80C188-16 CMOS |
| IC3, 6, 10 | 3 | IC STD Logic SM, TTL | S04 |
| IC1, 2, 4, 5 | 4 | IC STD Logic SM, TTL | LS245 |
| IC7, 8 | 2 | IC STD Logic SM, HCT | HCT373 |
| L1-20 | 20 | Inductor SM , Misc | 1206 CASE 31R at 100MHz |
| L21, 22 | 2 | Inductor SM , Misc | 4.7uH 900Ma CASE 2220 |
| RP3+6 | 2 | RES Fixed SM, Network | 10K DIL 4 IND ROHM MNR34 |
| RP1+4, RP2+5 | 4 | RES Fixed SM, Network | 10K DIL 4 IND ROHM MNR34 |
| D1, 2 | 2 | Semi SM, Diode | Single 1N4148 |
| D3 | 1 | Semi SM, Diode | BAR43S 2x SCHOTTKY SOT23 |
| TR1, 2 | 2 | Semi SM, Trans | NCHANNEL MOSFET2N7002 |
| C1 | 1 | CAP SM, Ceramic | 1N0 10% 50V 0805 |
| C14-32, 38, 39 | 21 | CAP SM, Ceramic | 100N 10% 50V 0805 |
| C4 | 1 | CAP SM, Ceramic | 180P 10% 0805 |
| C5-13 | 9 | CAP SM, Ceramic | 220P 10% 50V 0805 |
| R9-11 | 3 | RES Fixed SM, 1206 | 33R 5% 1/8W |
| R4, 7 | 2 | RES Fixed SM, 1206 | 1K0 5% 1/8W |
| R1 | 1 | RES Fixed SM, 1206 | 4K7 5% 1/8W |
| R13-15 | 3 | RES Fixed SM, 1206 | 10K 5% 1/8W |
| R12 | 1 | RES Fixed SM, 1206 | 330R 5% 1/8W |
| R6 | 1 | RES Fixed SM, 1206 | 47K 5% 1/8W 1206 |
| R5, 16, 17 | 3 | RES Fixed SM, 1206 | 100R 5% 1/8W |
| R2, 3, 8 | 3 | RES Fixed SM, 1206 | 2K2 5% 1/8W |
| IC11 | 1 | IC PLD SM, Misc | 16V8-25 PLCC |
| IC9 | 1 | IC PLD SM, Misc | 16V8-25 PLCC |

# Appendix B. Connections

There are three connectors on the board. Pin 1 of each connector is marked, and they are shown here as they appear on the board.

PL1
STEbus 64-way a/c DIN41612

| Pin | Row a | | Row c |
|---|---|---|---|
| 1 | gnd | o | o | gnd |
| 2 | +5V | o | o | +5V |
| 3 | D0 | o | o | D1 |
| 4 | D2 | o | o | D3 |
| 5 | D4 | o | o | D5 |
| 6 | D6 | o | o | D7 |
| 7 | A0 | o | o | gnd |
| 8 | A2 | o | o | A1 |
| 9 | A4 | o | o | A3 |
| 10 | A6 | o | o | A5 |
| 11 | A8 | o | o | A7 |
| 12 | A10 | o | o | A9 |
| 13 | A12 | o | o | A11 |
| 14 | A14 | o | o | A13 |
| 15 | A16 | o | o | A15 |
| 16 | A18 | o | o | A17 |
| 17 | CM0 | o | o | A19 |
| 18 | CM2 | o | o | CM1 |
| 19 | ADRSTB* | o | o | gnd |
| 20 | DATACK* | o | o | DATSTB* |
| 21 | TRFERR* | o | o | gnd |
| 22 | ATNRQ0* | o | o | SYSRST* |
| 23 | ATNRQ2* | o | o | ATNRQ1* |
| 24 | ATNRQ4* | o | o | ATNRQ3* |
| 25 | ATNRQ6* | o | o | ATNRQ5* |
| 26 | gnd | o | o | ATNRQ7* |
| 27 | BUSRQ0* | o | o | BUSRQ1* |
| 28 | BUSAK0* | o | o | BUSAK1* |
| 29 | SYSCLK | o | o | VSTBY |
| 30 | -12V | o | o | +12V |
| 31 | +5V | o | o | +5V |
| 32 | gnd | o | o | gnd |

PL2

Reset switch connector - 2-pin header
switch o o gnd

PL3

I/O connector 10-way ribbon-cable

| | | | | |
|---|---|---|---|---|
| channel B data in | 9 | o o | 10 | channel B data out |
| channel B CTS in | 7 | o o | 8 | channel B RTS out |
| gnd | 5 | o o | 6 | n/c |
| channel A RTS out | 3 | o o | 4 | channel A CTS in |
| channel A data out | 1 | o o | 2 | 2 channel A data in |

# Appendix C. Specification

| | |
|---|---|
| **Operating temperature** | 0C to 55C |
| **Power Consumption (typ)** | 5V + /- 0.25V 0.7A<br>+12V +/- 1V 30mA<br>-12V +/- 1V 30mA |
| **Microprocessor** | 80188 8MHz |
| **On-board memory** | 64k RAM maximum<br>128k EPROM maximum<br><br>supplied with 4k RAM |
| **Off-board memory** | 1MB via STEbus |
| **on-board I/O** | 2 DMA controllers<br>3 counter-timers<br><br>Zilog 8530 SCC<br><br>Two RS232 channels with optional handshaking |
| **Off-board I/O** | 4096 locations via STEbus |
| **Bus** | STEbus |
| **Bus connector** | 64 a/c DIN41612 |
| **Format** | Single Eurocard |
| **Dimensions** | 167mm x 100mm x 12mm |
| **Weight** | 160g |

# Appendix D. Example Programs.

```
        TITLE   'SC88T EXAMPLE PROGRAM'

; This is an example program for the SC88T board. A real time
; clock is set up and the time is sent to a terminal continuously.

; Users are free to copy and adapt this code for programs
; that run on the SC88T. Most applications will not need
; all the code here.

; The code performs the following operations:

;       Initialises the SCC and includes input and output routines

;       Initialises timer 2 of the 80188 as a interrupt source and
;       sets up its interrupt vector. The interrupt handler
;       increments a real time clock.

;       Sets up the chip select for this EPROM for a 8K, 16K or 32K EPROM

; This program was assembled using Digital Research's ASM86 as supplied
; with Concurrent DOS

; This assembler is an 8086 assembler and so does not include the
; extra instructions available on the 80188 so the following
; library (also supplied with Concurrent) must be included:

        INCLUDE 186.LIB
```

```
;       ***************************************************
;       *                                                 *
;       *               Definitions                       *
;       *                                                 *
;       ***************************************************
;
; Set baud rate. The allowed values are
; 1=50 baud, 2=75 baud, 3=110 baud, 4=134.5 baud, 5=150 baud,
; 6=300 baud, 7=600 baud, 8=1200 baud, 9=1800 baud, 10=2400 baud,
; 11=3600 baud, 12=4800 baud, 13=7200 baud, 14=9600 baud.
BDEF    EQU     14              ; Console baud rate
PDEF    EQU     8               ; Printer baud rate

; Set to real-time clock ticks in ms
TICK    EQU     20              ; 50 Hz
;
WAITI   EQU     2               ; 8 MHz 2 wait states for peripherals
WAITE   EQU     2               ; 8 MHz 2 wait states for EPROM
WAITR   EQU     0               ; 8 MHz 0 wait states for RAM
;
INTREG  EQU     0FF00H          ; internal register address (I/O)
EOIREG  EQU     INTREG+022H     ; end-of-interrupt register
TINTCR  EQU     INTREG+032H     ; timer interrupt control register
TCNT2   EQU     INTREG+060H     ; timer 2 count register
TMAXA2  EQU     INTREG+062H     ;       maximum count A register
TMCN2   EQU     INTREG+066H     ;       mode/control word
UMCS    EQU     INTREG+0A0H     ; upper memory chip select register
LMCS    EQU     INTREG+042H     ; lower memory chip select register
MPCS    EQU     INTREG+0A8H
PACS    EQU     INTREG+0A4H     ; peripheral chip select register
RELREG  EQU     INTREG+0FEH     ; relocation register
;
PBA     EQU     0FB00H          ; peripheral base address (I/O)
SCC     EQU     PBA+128         ; SCC
SCCBC   EQU     SCC             ; SCC channel B control register
SCCAC   EQU     SCC+1           ; SCC channel A control register
SCCBD   EQU     SCC+2           ; SCC channel B data register
SCCAD   EQU     SCC+3           ; SCC channel A data register

LF      EQU     10
CR      EQU     13
```

```
; ************************************************
; *                                              *
; *     Storage declarations                     *
; *                                              *
; ************************************************


; Define stack segment and stack usage

        SSEG    003OH                   ; initialisation stack top is 03FFH
        ORG     010OH
STACK   EQU     $

; Define data segment and usage

        DSEG    0040H                   ; data area is 0400H upwards
DATA    EQU     $

; A real-time clock is set up on initialisation
TIMEF   RB      1                       ; time fraction in 20ms (binary)
TIMES   RB      1                       ; time seconds (BCD)
TIMEM   RB      1                       ; time minutes (BCD)
TIMEH   RB      1                       ; time hours   (BCD)

;
```

```
; ************************************************
; *                                              *
; *              Start of Code                   *
; *                                              *
; *             Serial port drivers              *
; *                                              *
; ************************************************

; Users code will probably fit into the top 2K of the EPROM. In this case
; use CSEG OFF8OH and change the ORG statement for the top 16 bytes of
; the EPROM

        CSEG    0FE00H                  ; 8k PROGRAM
        ORG     0

; SCC initialisation data table
; All data consists of two bytes - the SCC register number
; followed by the value to be written
SCCTA   DB      009H,080H               ; reset channel A
        DB      004H,04CH               ; clock x16, 2 stop bits
        DB      001H,000H               ; disable external/status interrupts
        DB      003H,0C0H               ; Rx 8 bits
        DB      005H,062H               ; Tx 8 bits, RTS
        DB      009H,000H               ; clear reset
        DB      00AH,000H
        DB      008H,056H               ; clocks use baud rate generator
SCCLA1  EQU     OFFSET $ - OFFSET SCCTA
        DB      00EH,002H               ; BRG source
        DB      00EH,003H               ; BRG enable
        DB      003H,0C1H               ; Rx enable
        DB      005H,0EAH               ; Tx enable, DTR, RTS
        DB      00FH,000H               ; disable external/status interrupts
        DB      000H,010H               ; Reset external/status interrupts
        DB      000H,010H               ; twice!
SCCLA2  EQU     OFFSET $ - OFFSET SCCTA - OFFSET SCCLA1
SCCTB   DB      009H,040H               ; reset channel B
        DB      004H,04CH               ; clock x16, 2 stop bits
        DB      003H,0C0H               ; Rx 8 bits, auto enables
        DB      005H,062H               ; Tx 8 bits, RTS
        DB      009H,000H               ; clear reset
        DB      00AH,000H
        DB      008H,056H               ; clocks use baud rate generator
SCCLB1  EQU     OFFSET $ - OFFSET SCCTB
        DB      00EH,002H               ; BRG source
        DB      00EH,003H               ; BRG enable
        DB      003H,0C1H               ; Rx enable
        DB      005H,0EAH               ; Tx enable, DTR, RTS
        DB      00FH,000H               ; Disable external/status interrupts
        DB      000H,010H               ; Reset external/status interrupts
        DB      000H,010H               ; Twice!
SCCLB2  EQU     OFFSET $ - OFFSET SCCTB - OFFSET SCCLB1

; SCC baud rate divisors for x16 clock, PCLK = 4 MHz
DIVIS   DW      12500,2500,1665,1134, 924, 831, 415, 206
        DW        102,  67,  50,  33,  24,  15,  11,   5
```

```
; I/O routines
;
; Initialise channel A
;
CONINT: PUSHA                           ; save registers
        MOV     DX,SCCAC
        MOV     BX,BDEF*2
        MOV     SI,OFFSET SCCTA
        PUSH    DS
        PUSHW   SCCLA2                  ; Length of second part of table
        MOV     CX,SCCLA1               ; Length of first part of table
        JMPS    CHINT
;
; Initialise channel B
;
LISINT: PUSHA
        MOV     DX,SCCBC
        MOV     BX,PDEF*2
        MOV     SI,OFFSET SCCTB
        PUSH    DS
        PUSHW   SCCLB2                  ; Length of second part of table
        MOV     CX,SCCLB1               ; Length of first part of table
CHINT:  PUSH    CS                      ; set DS register
        POP     DS
CHIN1:  OUTSB                           ; access recovery time of SCC prevents
        LOOP    CHIN1                   ; use of REP OUTS instruction
        MOV     AL,12
        OUT     DX,AL
        MOV     AX,OFFSET DIVIS[BX]
        OUT     DX,AL
        MOV     AL,13
        OUT     DX,AL
        MOV     AL,AH
        OUT     DX,AL
        POP     CX                      ; Length of second part of table
CHIN2:  OUTSB
        LOOP    CHIN2
        POP     DS                      ; restore registers
        POPA
        RET
;
; Input status of channel A
; Returns AL = 0FFH if char available else 0
;
CONIST: PUSH    DX
        MOV     DX,SCCAC
        JMPS    CHIST
;
; Input status of channel B
;
LISIST: PUSH    DX
        MOV     DX,SCCBC
CHIST:  IN      AL,DX
        AND     AL,01
        JZ      CHIS1
        OR      AL,0FFH
CHIS1:  POP     DX
        RET
;
; Input a character from channel A
; Returns AL = char
;
CONIN:  CALL    CONIST
        JZ      CONIN                   ; Wait until char available
        PUSH    DX
        MOV     DX,SCCAD
        JMPS    CHIN
;
; Input a character from channel B
;
LISIN:  CALL    LISIST
        JZ      LISIN
        PUSH    DX
        MOV     DX,SCCBD
CHIN:   IN      AL,DX
        AND     AL,07FH
        POP     DX
        RET
;
; Output status of channel A
; Returns AL = 0FFH if char can be sent
;
CONOST: PUSH    DX
        MOV     DX,SCCAC
        IN      AL,DX
        AND     AL,04
        JZ      CONOS1
        OR      AL,0FFH
CONOS1: POP     DX
        RET
;
```

```
; Output status of channel B
; Returns AL = 0FFH if SCC ready to send char and CTS line asserted
;
LISOST: PUSH    DX
        MOV     DX,SCCBC
        IN      AL,DX
        AND     AL,024H
        CMP     AL,024H
        JNZ     LISOS1                  ; If  NOT (EMPTY AND CTS)
        OR      AL,0FFH
        POP     DX
        RET
LISOS1: XOR     AL,AL
        POP     DX
        RET
;
; Output a character to channel A
; Call with AL = char
;
CONOUT: PUSH    AX
CONOTW: CALL    CONOST
        JZ      CONOTW                  ; Wait until ready
        POP     AX
        PUSH    DX
        MOV     DX,SCCAD
        OUT     DX,AL
        POP     DX
        RET
;
; Output a character to channel B
; Call with AL = char
;
LISOUT: PUSH    AX
LISOTW: CALL    LISOST
        JZ      LISOTW
        POP     AX
        PUSH    DX
        MOV     DX,SCCBD
        OUT     DX,AL
        POP     DX
        RET
;
```

```
; ***********************************************
; *                                             *
; *            Interrupt Handler                *
; *                                             *
; ***********************************************
; Timer 2 is set up as a real time clock
; On each interrupt ( every 20ms ) the following locations are incremented
;       TIMEF           fractions of a second (in 20ms) in binary
;       TIMES           seconds in BCD
;       TIMEM           minutes in BCD
;       TIMEH           hours in BCD

TIMINT: PUSH    AX                      ; save registers
        PUSH    DX
        PUSH    BX
        PUSH    DS
        MOV     AX,SEG DATA
        MOV     DS,AX
        XOR     AX,AX
        MOV     BX,OFFSET TIMEF
        INC     BYTE PTR [BX]           ; increment fraction
        CMP     BYTE PTR [BX],50        ; 1 second?
        JB      TIMIN1
        MOV     [BX],AH
        INC     BX
        MOV     AL,[BX]                 ; increment seconds
        INC     AL
        DAA
        MOV     [BX],AL
        CMP     AL,60H                  ; 1 minute?
        JB      TIMIN1
        MOV     [BX],AH
        INC     BX
        MOV     AL,[BX]                 ; increment minutes
        INC     AL
        DAA
        MOV     [BX],AL
        CMP     AL,60H                  ; 1 hour?
        JB      TIMIN1
        MOV     [BX],AH
        INC     BX
        MOV     AL,[BX]                 ; increment hours
        INC     AL
        DAA
        MOV     [BX],AL
TIMIN1: POP     DS
        POP     BX
        MOV     DX,EOIREG               ; issue a non-specific
        MOV     AX,08000H               ;   end-of-interrupt command
        OUT     DX,AX
        POP     DX                      ; restore registers
        POP     AX
        IRET
```

```
; ***********************************************
; *                                             *
; *            Messages and utilities           *
; *                                             *
; ***********************************************

SIGNON  DB      CR,LF,LF
        DB      '        -----*****<<<<< The SC88T >>>>>*****-----'
        DB      CR,LF,LF,LF,0

TIMEIS  DB      CR,'The time is ',0
MINS    DB      ' minutes and ',0
SECS    DB      ' seconds',0

; Utility routines

; print a string terminated by a zero byte
;
WRITES: MOV     AL,CS:[BX]
        TEST    AL,AL
        JZ      WRITER
        CALL    CONOUT
        INC     BX
        JMPS    WRITES
WRITER: RET
;
; print CX hex digits from AX
;
WRITEH: DEC     CX                      ; if CX > 1
        JZ      WRITE1
        PUSH    AX                      ;    then WRITEH(AX>>4,CX-1)
        DB      0C1H,0E8H,4             ; SHR AX,4
        CALL    WRITEH
        POP     AX
WRITE1: AND     AL,00FH
        DAA                             ; if A-F then add 6
        ADD     AL,0F0H                 ;    and set carry else subtract 10
        ADC     AL,040H                 ; convert to ASCII
        CALL    CONOUT
        RET
;
; print CX decimal digits from AX by recursion
;
PDEC:   MOV     BL,10
        DIV     BL
        DEC     CX                      ; if CX > 1
        JZ      PDEC1
        PUSH    AX                      ;    then PDEC(AX/10,CX-1)
        XOR     AH,AH
        CALL    PDEC
        POP     AX
PDEC1:  MOV     AL,AH
        AND     AL,00FH
        ADD     AL,'0'
        CALL    CONOUT
        RET

; print AX in hex
;
PWORDH: PUSH    CX
        MOV     CX,4
        CALL    WRITEH
        POP     CX
        RET
;
; print AL in hex
;
PBYTEH: PUSH    CX
        MOV     CX,2
        CALL    WRITEH
        POP     CX
        RET
```

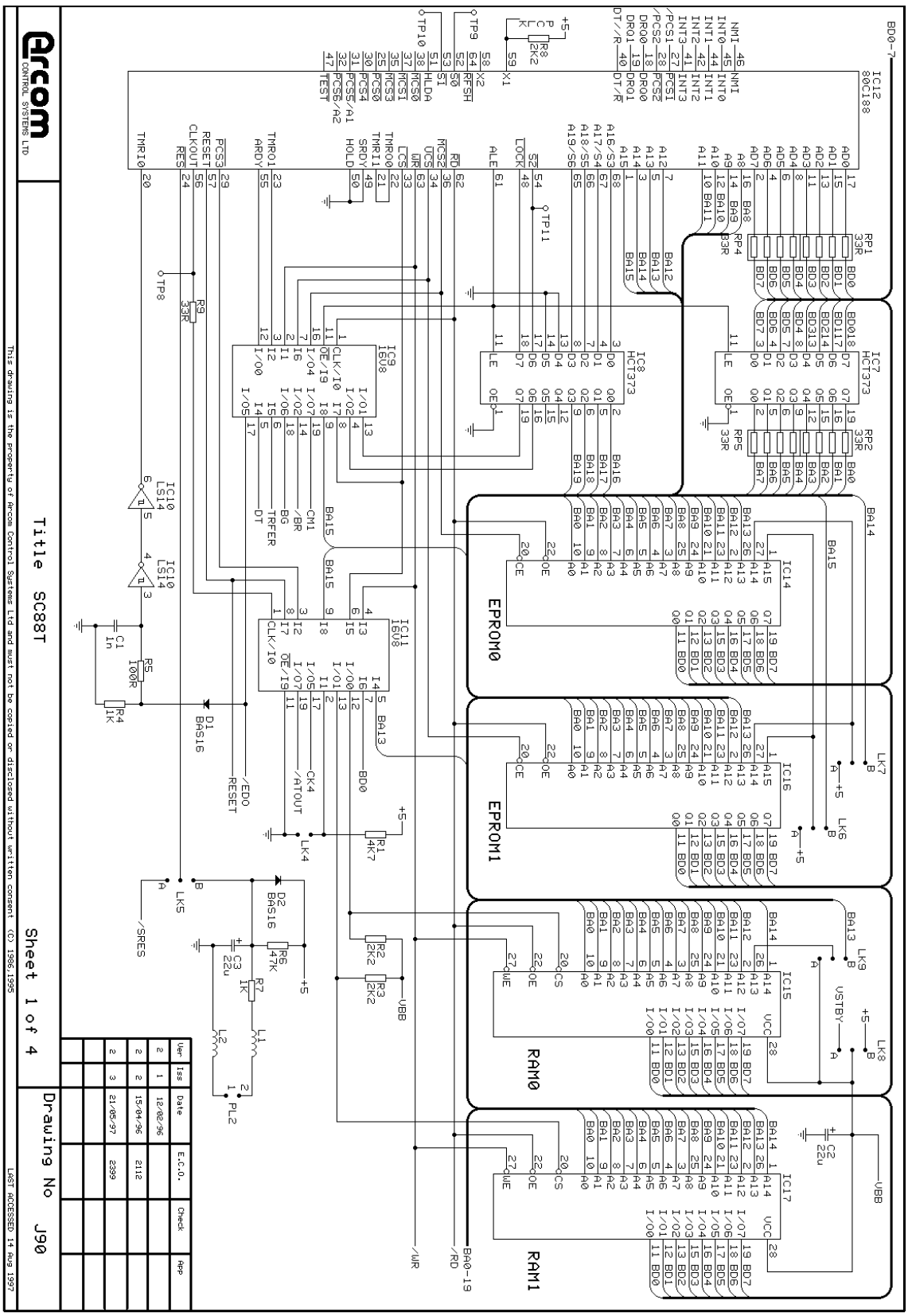
```
;*************************************************
;*                                               *
;*            Initialisation code                *
;*                                               *
;*************************************************
;
START:  CLI                             ; ensure interrupts are disabled
        MOV     DX,LMCS                 ; LMCS
        MOV     AX,03FFBH + WAITR       ; 256k RAM
        OUT     DX,AX
        MOV     DX,MPCS
        MOV     AX,080BBH               ; PCS are in I/O space
        OUT     DX,AX
        MOV     DX,PACS
        MOV     AX,00FBDH + WAITI       ; PBA=FF80
        OUT     DX,AX
        OUT     DX,AL
        MOV     AX,SEG STACK            ; set up stack pointer
        MOV     SS,AX
        MOV     SP,OFFSET STACK
        MOV     AX,SEG DATA
        MOV     DS,AX
        CLD                             ; clear direction
        STI
        CALL    CONINT                  ; Initialise serial ports
        CALL    LISINT
        MOV     BX,OFFSET SIGNON        ; Print greetings
        CALL    WRITES
;
```

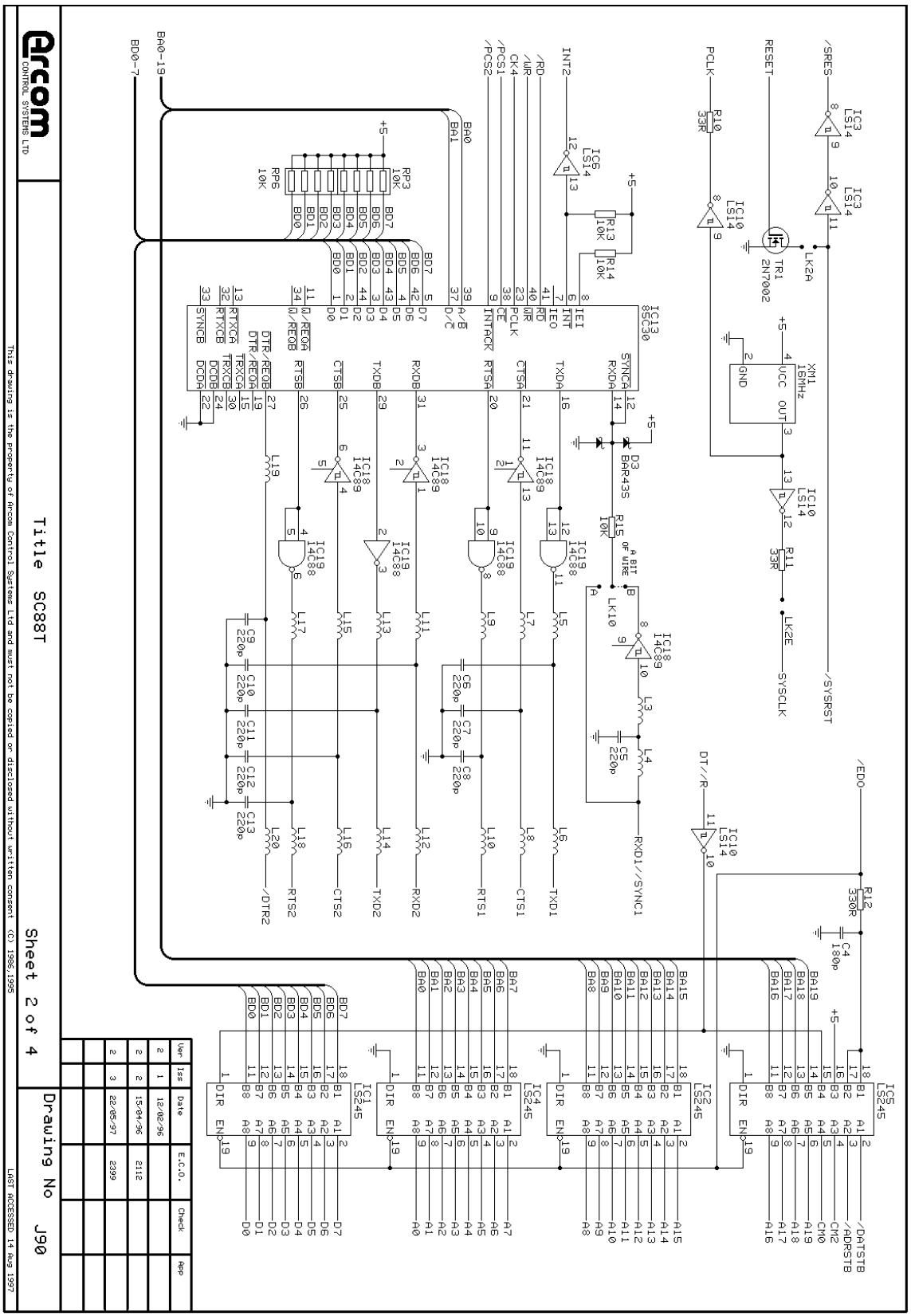
```
; Set up interrupt handler for Timer 2
;
        XOR     AX,AX                   ; Write address of interrupt
        MOV     ES,AX                   ; handler into interrupt table
        MOV     AX,OFFSET TIMINT
        MOV     BX,76
        MOV     ES:[BX],AX              ; Offset into address 0004CH
        MOV     BX,78                   ; Segment into address 0004EH
        MOV     AX,SEG TIMINT
        MOV     ES:[BX],AX

        MOV     AX,SEG DATA             ; Zero time and date
        MOV     ES,AX
        XOR     AX,AX
        MOV     CX,3
        MOV     DI,OFFSET TIMEF
        REP STOS AX
        MOV     DX,TCNT2                ; Set up timer 2
        XOR     AX,AX                   ; clear count
        OUT     DX,AX
        INC     DX
        INC     DX
        MOV     AX,2000*TICK            ; real-time clock constant
        OUT     DX,AX
        MOV     DX,TMCW2
        MOV     AX,0E001H               ; EN, INT, CONT
        OUT     DX,AX
        MOV     DX,TINTCR               ; enable timer interrupts
        XOR     AX,AX                   ; priority = 0
        OUT     DX,AX
```

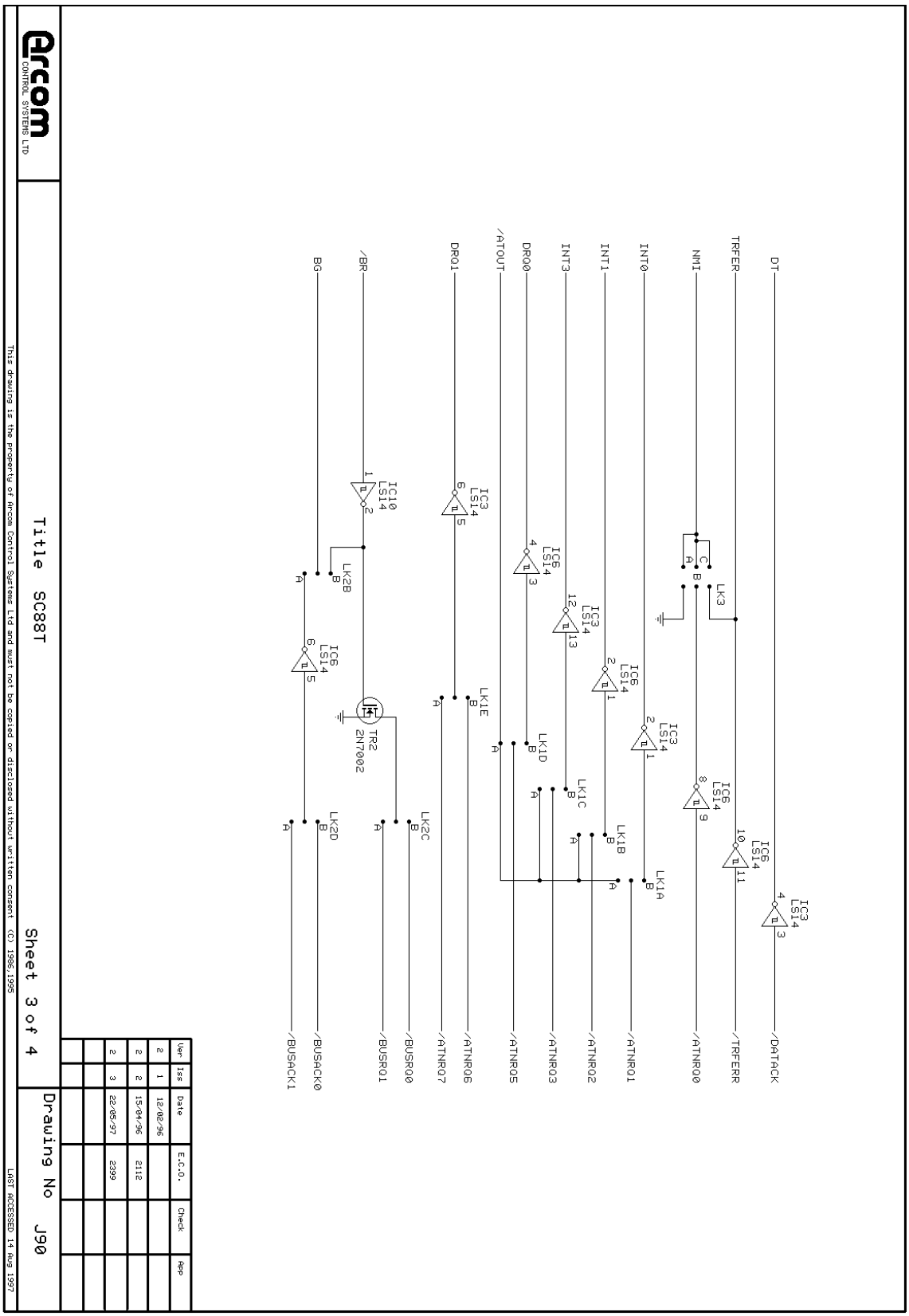
```
; ***********************************************
; *                                             *
; *            Example Program                  *
; *                                             *
; ***********************************************

        PUSHW   SEG DATA                ; Set up DS register
        POP     DS
PROG:   MOV     BX,OFFSET TIMEIS                        ; 'The time is'
        CALL    WRITES
        CLI
        MOV     AL,TIMEF        ; Read time while ints are disabled
        PUSH    AX
        MOV     AL,TIMES
        PUSH    AX
        MOV     AL,TIMEM
        STI
        CALL    PBYTEH          ; Print number of minutes
        MOV     BX,OFFSET MINS                          ; 'minutes and'
        CALL    WRITES
        POP     AX              ; Print number of seconds
        CALL    PBYTEH
        MOV     AL,'.'                                  ; '.'
        CALL    CONOUT
        POP     AX              ; Fraction in 20ms
        XOR     AH,AH
        SHL     AL,1            ; * 2
        MOV     CX,2
        CALL    PDEC            ; Print hundredths of a second
        MOV     BX,OFFSET SECS                          ; 'seconds'
        CALL    WRITES
        JMP     PROG
```

```
; ***********************************************
; *                                             *
; *            Reset entry point                *
; *                                             *
; ***********************************************

; The 80188 processor jumps to segment OFFFFH offset 00000H on reset
; This corresponds to the top 16 bytes of the EPROM in IC16
; Thus these bytes must set up the UMCS register so that the whole
; EPROM is enabled and then jump to the rest of the initialisation
; code.

        CSEG    0FFFFH
        ORG     1FF0H                   ; This positions this code in the
                                        ; top 16 bytes of a 8K EPROM

        MOV     DX,UMCS                 ; Chip select register for EPROM
;       MOV     AX,0FE3CH + WAITE       ; 8k    Enable ONE of these statements
;       MOV     AX,0FC3CH + WAITE       ; 16K   depending on your size of
        MOV     AX,0F83CH + WAITE       ; 32K   EPROM
        OUT     DX,AX                   ; Set up UMCS
        JMPF    START                   ; Jump to code in rest of EPROM

        END
```

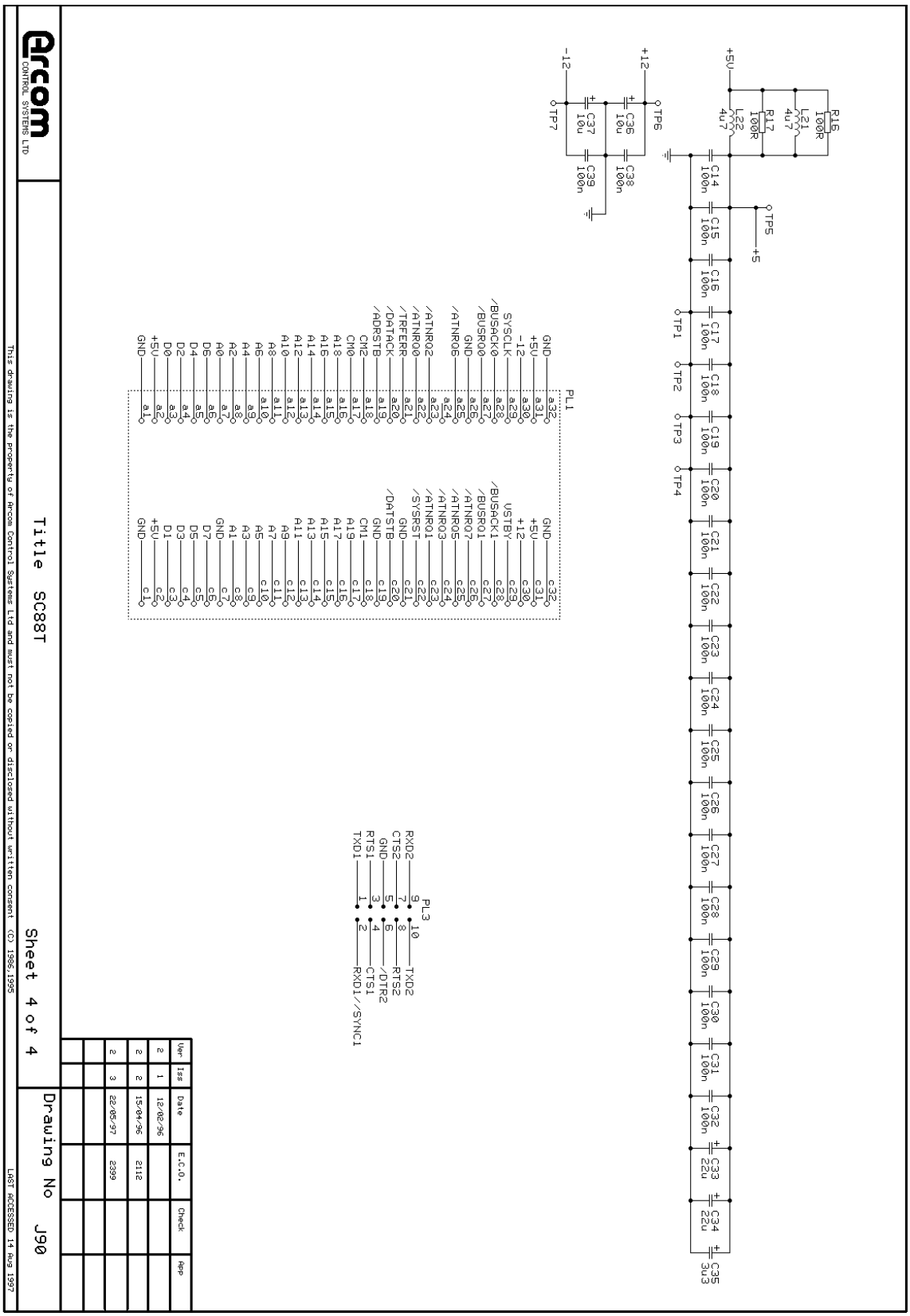
# Appendix E. Circuit Diagrams




Arcom CONTROL SYSTEMS LTD
Title SC88T
Sheet 2 of 4
Drawing No J90
This drawing is the property of Arcom Control Systems Ltd and must not be copied or disclosed without written consent (C) 1996,1997
LAST ACCESSED 14 Aug 1997
Ver Iss Date E.C.O. Check App
2 1 12/02/96
2 2 15/04/96 2112
2 3 22/05/97 2399
/SRES
RESET
PCLK
/SYSRST
/SYSCLK
/EDO
DT/R
/DATSTB
/ADRSTB
CM2
CM0
INT2
/RD
/WR
CK4
/PCS1
/PCS2
BA0-19
BD0-7
IC3 LS14
IC10 LS14
IC5 LS14
IC13 85C30
IC18 I4C89
IC19 I4C88
IC1 LS245
IC2 LS245
IC4 LS245
IC5 LS245
R10 33R
R11 33R
R12 330R
R13 10K
R14 10K
R15 10K
RP3 10K
RP6 10K
C4 180p
C5 220p
C6 220p
C7 220p
C8 220p
C9 220p
C10 220p
C11 220p
C12 220p
C13 220p
TR1 2N7002
XM1 16MHz
D3 BAR43S
LK2A
LK2E
LK10
A BIT OF WIRE
RXD1/SYNC1
TXD1
CTS1
RTS1
RXD2
TXD2
CTS2
RTS2
/DTR2




| Ver | Iss | Date | E.C.O. | Check | App |
|---|---|---|---|---|---|
| 2 | 1 | 12/02/96 | | | |
| 2 | 2 | 15/04/96 | 2112 | | |
| 2 | 3 | 22/05/97 | 2399 | | |


Arcom CONTROL SYSTEMS LTD
Title SC88T
Sheet 4 of 4
Drawing No J90
This drawing is the property of Arcom Control Systems Ltd and must not be copied or disclosed without written consent. (C) 1986,1995
LAST ACCESSED: 14-Aug-1997
Ver Iss Date E.C.O. Check App
2 1 12-02-96
2 2 15-04-96 2112
2 3 22-05-97 2339
+5V
+12
-12
GND
TP1
TP2
TP3
TP4
TP5
TP6
TP7
R15 100R
L21 4u7
R17 100R
L22 4u7
C14 100n
C15 100n
C16 100n
C17 100n
C18 100n
C19 100n
C20 100n
C21 100n
C22 100n
C23 100n
C24 100n
C25 100n
C26 100n
C27 100n
C28 100n
C29 100n
C30 100n
C31 100n
C32 100n
C33 22u
C34 22u
C35 3u3
C36 10u
C37 10u
C38 100n
C39 100n
PL1
PL3
TXD2
RTS2
/DTR2
CTS1
RXD1/SYNC1
RXD2
CTS2
GND
RTS1
TXD1