SETHER

STEbus Ethernet Network Board

User's Manual

Contents

Section			Page
	1.	Introduction	3
		The SETHER Board and Ethernet	3
		Features	3
	2.	Ethernet (IEEE 802.3)	4
		Introduction	4
		Ethernet Packet Structure	5
	3.	Hardware Installation	7
		Storage and Handling	7
		Links and Options	7
		Installation	10
		Network Cabling and Connectors	10
	4.	Circuit Description	13
	5.	Troubleshooting	16
Appendix			
	A.	Component List	17
	B.	STEbus Connector Pin Assignments	18
	C.	Specification	19
	D.	Example Program	20
	E.	AUI Connector Pin Assignments	28
	F.	Circuit Diagrams	29

Revision History

Manual	PCB	Comments
V1 Iss 2	V1 Iss 2	900807 First Full Release
V2 Iss 1	V2 Iss 1	910807 Released in new format
		[ECO 418] Various edits
		New example program
V2 Iss 1a		920809 [ECO837,915]
V2 Iss 1b		940809 New AUI Connector Diagram

The choice of boards 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.

Section 1. Introduction

The SETHER board and Ethernet

The SETHER board provides a high performance interface between STEbus based processor systems and the IEEE 802.3 Ethernet network. Ethernet is a 10 Mbits/s high-performance networking system, allowing many different types of computer (PCs, STEbus, VMEbus) to interchange messages and share files, hard disks, and other peripherals. This manual provides a brief overview of Ethernet and shows how the SETHER board is implemented. The SETHER utilises the National Semiconductor DP8390 Ethernet chip set which consists of the DP8390 Network Interface Chip (NIC), DP8391 Serial Network Interface (SNI) and DP8392 Coaxial Transceiver Interface (CTI).

A number of network driver packages are available from Arcom for DOS based, OS/9 systems and standalone target systems. For detailed hardware and programming information relating to the 8390 chip set refer to the National Semiconductor's "Data Comms, Local Area Network, UART Handbook".

The board includes two useful LED's to indicate when data is being received or transmitted by the 8390 NIC.

Features include:

- Full STEbus slave interface
- Full Ethernet interface
- Thin-wire Ethernet (Cheapernet) interface
- National Semiconductors' DP8390 controller chip
- Software compatible with PCbus Ethernet board
- Ethernet BIOS ROM socket
- Dual-port RAM with local DMA
- LED indication of Ethernet Rx and Tx traffic
- Interrupts on any level (ATNRQ0-7)
- SETHER board network 'burned in address' stored in PROM
- Complete TCP/IP protocol package available
- Software drivers available from Arcom for various operating systems such as OS-9

Section 2. Ethernet (IEEE 802.3)

Introduction

The DP8390 chip set is designed to provide the physical and media access control layer functions of the local area network as defined by the IEEE 802.3 standard. The standard is based on the access method known as carrier-sense multiple access with collision detect (CSMA/CD). The operational principle is such that if a network node wants to transmit, it first listens to the network for other transmitted traffic. If the medium is quiet the network controller will begin transmission while monitoring for a possible collision with another transmitter. If two or more stations should simultaneously transmit, they will detect the collision and back off for a random amount of time before making another attempt.

The physical layer of Ethernet may be implemented in two forms according to the performance required from the system. Thick wire Ethernet (referenced as 10BASE5) uses high specification coaxial cable which is connected to the network node via a transceiver unit (Medium Attachment Unit, MAU) and drop cable (Attachment Unit Interface, AUI). The AUI cable consists of four individually shielded twisted pairs fitted with D15 connectors. Thin wire Ethernet or Cheapernet (referenced as 10BASE2) uses a more flexible coaxial cable with BNC connectors. The SETHER board supports interfaces for both Cheapernet and a D15 AUI cable connector for connection to standard Ethernet. A comparison of performance characteristics is shown below:-

Parameter	Ethernet	Cheapernet
Data Rate	10 Mbits	10 Mbits
Segment Length	500m	185m
Network Span	2500m	925m
Nodes per Segment	100	30
Node Spacing	2.5m (cable marked)	0.5m
Cable Type	Belden 9880 50Ω Double Shielded Rugged N Series Connectors	Cheapernet 50Ω(RG58A/U) Single Shielded Flexible BNC Connectors
Transceiver Drop Cable	Belden 9892 multi-way cable with D15 connectors (max drop length 50m)	Not needed due to high flexibility of RG59A/U cable

Each Ethernet board has a unique 6 byte identifying code (stored in PROM and referred to as the burned in address) which is issued by the IEEE. This identifier is used to indicate the source and the destination of packets transferred on the network, the Arcom burned in addresses start at 008066000001

Ethernet Packet Structure

The structure of an Ethernet packet is as follows:

6 bytes	destination address
6 bytes	source address
2 bytes	type/length field
46-1500 bytes	user data

If the destination address has all bits set, the packet is broadcast and will be received by all nodes that have broadcast reception enabled.

The source address is normally the sender's Burned In Address, copied from the on-board PROM into the 8390 chip. This ensures that every Ethernet card has a unique identity, no matter how large the network.

The type/length field is used by some protocols to indicate packet length, in others it serves as an identifier of the protocol used e.g 0806H for ARP, 0800H for IP. When creating a new protocol, it is advisable to choose a unique value for this field, to try to avoid clashes with other protocols. In accordance with Ethernet standards, this field is in high-low byte order.

The user data may be in any format, and will be passed across the network unaltered. If it is less than 46 bytes long, it must be padded with dummy bytes.

The hardware performs automatic CRC generation and checking, this operation is transparent to the programmer.

To transmit the packet, it is placed at the base of the SETHER dual-port RAM, and a transmit command is sent to the 8390 Ethernet chip. The 8390 will then put the packet onto the network. The 8390 returns error codes if it is unable to send the packet (e.g. the retry limit has been exceeded). There is no built-in mechanism for checking whether the packet has arrived at its destination: the higher-level protocols are responsible for ensuring an acknowledgement is returned.

When receive has been enabled, the 8390 will automatically stack up incoming packets in the dual-port RAM without any processor intervention. The 8390 treats the RAM area as a

circular buffer, and will only cease reception if the buffer is full. The format of the received packet is as described above, but with a 4-byte hardware header:-

1 byte	packet status
1 byte	next hardware page
2 bytes	packet length

The status byte mirrors the Receive Status Register at the time of reception: if bit 0 is set, the packet is error-free.

The 'next hardware page' refers to the next 256-byte block of dual-port RAM used by the 8390.

The packet length is in low-high byte order, and includes the length of the 4-byte hardware header.

A simple Ethernet monitor program written in Aztec 'C' has been included in Appendix D of this manual. It uses 'promiscuous mode' which means that all network packets are received, irrespective of destination address. This mode is useful for network diagnosis, though it will result in a very high data throughput if used on a heavily-loaded network.

Section 3. Hardware Installation

Storage and Handling

The SETHER is supplied in static protective packing. It is important that normal precautions against static should be observed at all times.

The boards should always be kept in dry conditions at a controlled temperature.

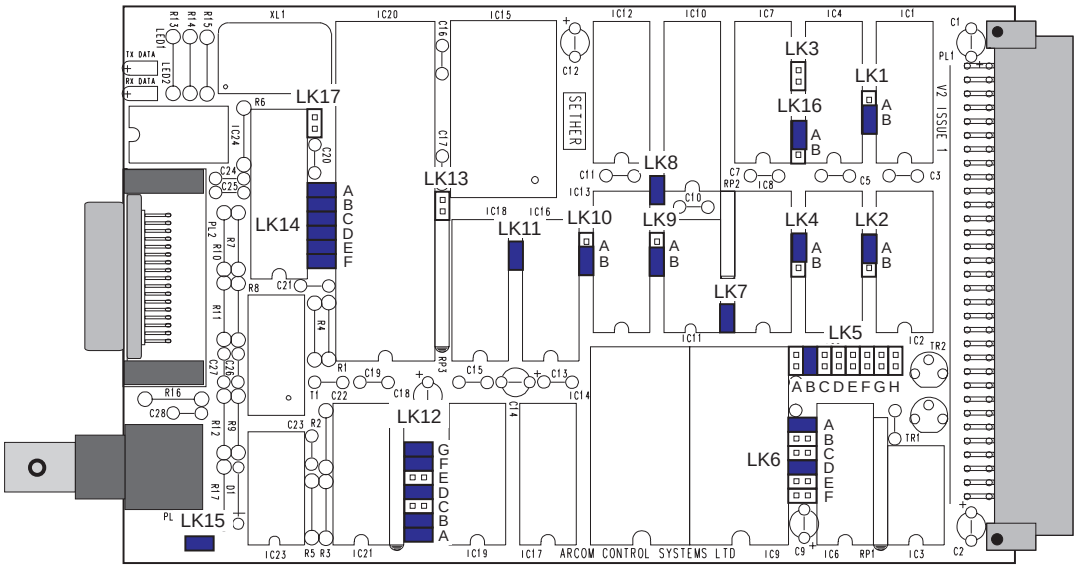
Under the terms of Arcom's warranty, any boards returned with damage attributed to incorrect handling or storage will not be repaired, nor replaced free of charge.

See also the specifications in Appendix C.

Links and Options

This section describes how to set the link defined features. The link positions are shown below. '+' indicates a standard setting. Note that all, links are viewed with the pcb edge connector to the right

Link Position Diagram



= Default Link Position

Links 1 and 16. RAM buffer size select.

	LK1	LK16	
+	B	A	8K RAM
	A	B	32K RAM

Links 2 and 4. ROM decode size select.

	LK2	LK4		
+	A	A	2764	(8k)
	A	A	27128	(16k)
	B	A	27256	(32k)
	B	B	27512	(64k)

Link 3. RAM base address A19

LK3 Insert to respond if A19 low

The RAM can appear at any multiple of its size in STEbus memory space. A18-13 are software programmable, but this link defines A19.

On reset the base address is set to80000H and the RAM is disabled

By default the software will generally set the RAM base-address to be D0000H and RAM will be enabled.

Link 5. ATNRQ select.

	LK5A	Interrupt on ATNRQ0*
+	LK5B	Interrupt on ATNRQ1*
	LK5C	Interrupt on ATNRQ2*
	LK5D	Interrupt on ATNRQ3*
	LK5E	Interrupt on ATNRQ4*
	LK5F	Interrupt on ATNRQ5*
	LK5G	Interrupt on ATNRQ6*
	LK5H	Interrupt on ATNRQ7*

Link 6. ROM socket base address.

+	LK6A	Respond if A14 low
	LK6B	Respond if A15 low
	LK6C	Respond if A16 low
+	LK6D	Respond if A17 low
	LK6E	Respond if A18 low
	LK6F	Respond if A19 low

The ROM can be mapped to any 16K boundary.The default link settings map the ROM at address D8000H

Link 7. Disable ROM space.

+	LK7	Insert to disable ROM.

Link 8. Ethernet 'burned in address' PROM high/low select

+	LK 8	Omit to select the top half of the Ethernet 'burned in address' PROM

Note: This link should be inserted to make the boards compatible with the standard drivers supplied on the distribution disk.

Link 9 and 10. ROM device size select.

	LK 9	LK 10		
+	B	B	2764	(8k)
	B	B	27128	(16k)
	B	A	27258	(32k)
	A	A	27512	(64k)

Link 11. RAM device size select.

+	LK 11	Insert for 62256 (32k), Omit for 6264 (8k)

Link area 12. I/O base address

+	LK12A	Respond if A5 low
+	LK12B	Respond if A6 low
	LK12C	Respond if A7 low
+	LK12D	Respond if A8 low
	LK12E	Respond if A9 low
+	LK12F	Respond if A10 low
+	LK12G	Respond if A11 low

The I/O address can be mapped to any 32-byte boundary of the STEbus I/O space (000 to FE0). Inserting a link in this area sets the corresponding register base address bit low. The default link settings map the I/O at address 280H.

Link 13. Connect +12V to AUI cable.

	LK13	Insert to connect +12v to an external transceiver unit (MAU).

Link 14. Enable Thin Wire Ethernet Transceiver Interface.

+	LK 14A	TX+
+	LK 14B	TX-
+	LK 14C	RX+
+	LK 14D	RX-
+	LK 14E	CD-
+	LK 14F	CD+

These links transfer the outputs from the Serial Network Interface chip DP8391 to the isolation transformer and on to the Coaxial Transceiver Interface, DP8392. If the D15 connector is used to drive an external thick/thin wire transceiver unit, all LK14 links should be removed.

Link 15. Network Segment Length Option.

+	LK15	Insert for standard IEEE 802.3 networks. Omit for extended segment lengths.

Each standard Ethernet segment has a maximum length of 185m. If it is necessary to expand this up to 300m, then Link 15 should be inserted.

IMPORTANT : If this option is selected then the network is limited to a single segment, ie no repeaters may be used and also all other SETHER cards on the segment must be set with Link 15 made. This configuration does not comply with the electrical specification of IEEE 802.3.

The state of Link 15 is only relevant if Thin Wire Ethernet is used, ie all Link area 14 are made.

Link 17. Network Version Select.

Omit for Ethernet Version 2, IEEE 802.3 networks, Cheapernet.

Insert for older version 1 networks.The network version is determined by the MAU transceiver used for the network. (This will be indicated on the MAU).

Installation

It is important to follow normal good practice associated with the installation and removal of boards from the backplane. Make sure all power to the backplane is off. Follow standard anti-static precautions, including earthed wrist straps and conductive carpets, when installing the board. **If in doubt, consult your supplier.**

Network Cabling and Connectors

Thin-wire Ethernet (Cheapernet) requires RG58A/U co-axial cable and 50 Ohm BNC connectors. Each BNC connector should be fitted with a BNC 'T' piece such that the network cabling can continue to the next node. If the SETHER board is the last node in the network chain, one side of the 'T' piece should be fitted with a 50 ohm terminator. (The network will not work without terminations fitted at both ends.) A few important parameters should be noted :-

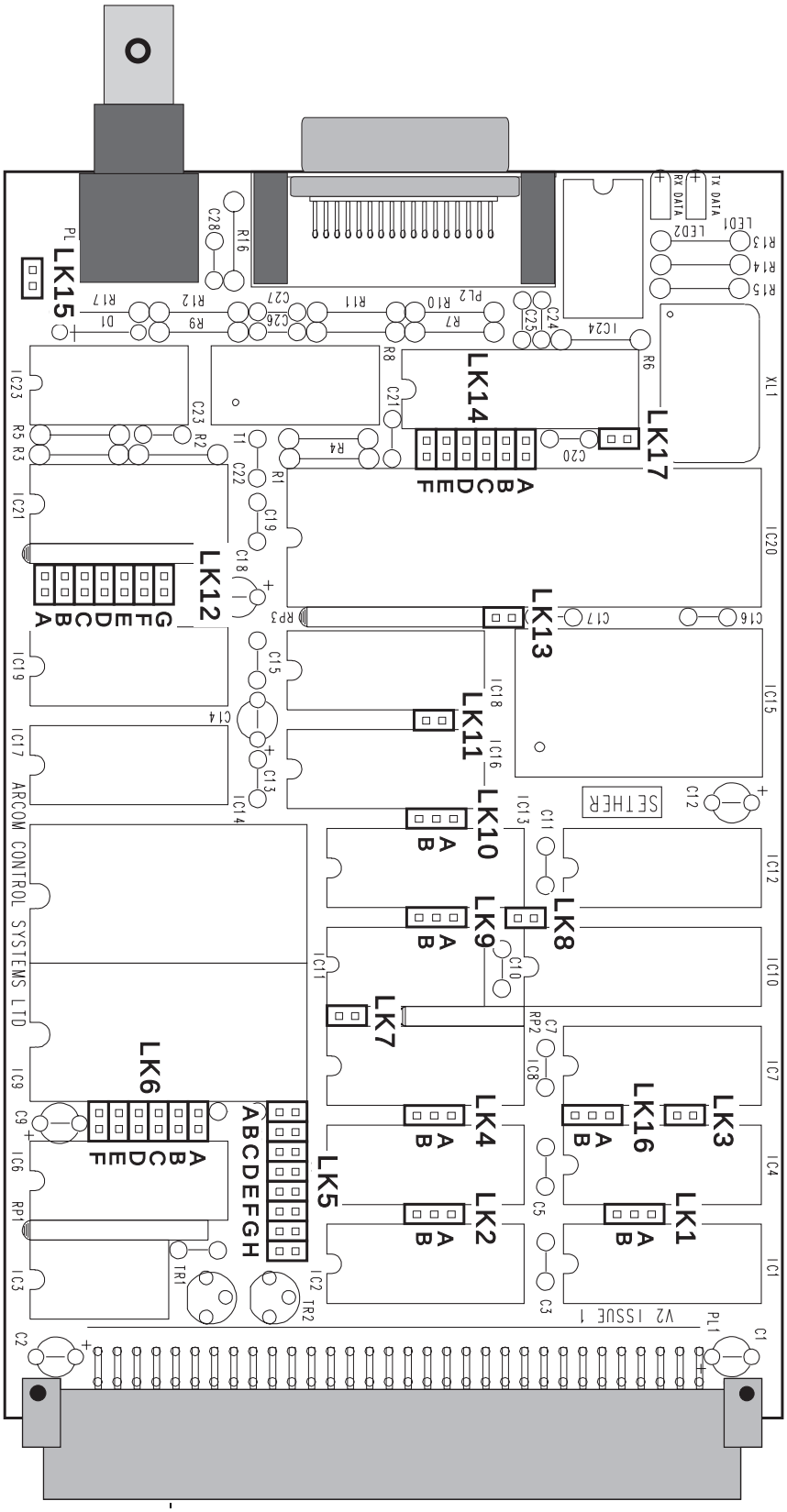
Minimum separation between nodes	500mm
Minimum bend radius of cable	30mm
Maximum length of any one segment	185m
Maximum number of nodes per segment	30

IMPORTANT NOTE: the network cable screen is independent from chassis ground, and that ALL CONNECTORS, BULK-HEAD or LINE, MUST BE ISOLATED FROM CHASSIS GROUND. Failure to observe this requirement may result in loss of noise immunity. Also the BNC 'T' pieces should be connected directly onto the SETHER board BNC connector, ie do not use any drop cable between these two.

Thick-wire Ethernet uses special multi-wire co-axial Ethernet cable, and can be up to 500m long. The SETHER board should be connected to a transceiver unit via a multiple twisted pair (78 ohm impedance) drop cable (such as Belden 9892 or similar) fitted with D15 connectors.This cable should be a maximum of 50m long. The network should also be terminated with 50 ohm terminators

User Configuration Record Sheet

This sheet may be duplicated.



Section 4. Circuit Description

The core of the SETHER is the DP8390 Network Interface Chip (IC20) which handles all the media access control to the Ethernet. The NIC interchanges packets of information via dual port memory (IC14) using an on chip DMA controller.

The SETHER also includes 28 pin EPROM socket (IC9) for network BIOS software, a PROM socket (IC11) for the Ethernet 'burned in address' and a control register (IC7). The control register is used to set the base address of the RAM buffer, control the local reset and enable or disable the RAM (bit assignments shown below).

Bit 7	-	SETHER board reset (high)
Bit 6	-	RAM Enabled (high)
Bit 5	-	A18
Bit 4	-	A17
Bit 3	-	A16
Bit 2	-	A15
Bit 1	-	A14
Bit 0	-	A13

The base addresses for the I/O registers and EPROM space are decoded by 74HCT688's (IC21 and IC6 respectively) and Link areas 12 and 6. A part of the RAM buffer base address (A18-A13) is selected by the value written to the control register (and compared by IC4). Bit A19is selected by LK3. Bit 7 of the control register serves as a software invoked reset. The board is reset when this bit is set high, this board reset will also clear the control register thus removing the local reset. Bit 6 is used to enable access (when high) to the RAM from the STEbus.

Access to the EPROM, PROM and control register is independent of NIC accesses to dual port memory. The RAM is double buffered and is accessed under the control of the dual port arbiter.

The arbitration is performed by the 16R4 PAL (IC18) and is programmed to give priority to the NIC in preference to a STEbus access. Access to the I/O registers with in the NIC are arbitrated by the NIC, ie when /CS and /SRD or /SWR are asserted the control circuit will wait for an /ACK to be asserted by the NIC. At this time the data buffers from STEbus to the NIC are enabled and the STEbus access is terminated.

Device decodes, strobe signals and STEbus /DATACK are generated by the 22V10 PAL (IC10) and 16L8 PAL (IC12). These also use the arbitration signals /STE GNT and NIC GNT to enable the appropriate buffers and direction controls. STEbus /DATACK is driven by NMOS FET TR1.

The EPROM base address is set by Link area 6 and compared with the STEbus address lines using IC6. Similarly the I/O base address (PROM, control reg. and NIC registers) is selected by Link area 12 and IC 21. The base address of RAM set in the control register is compared by IC4.

The 'burned in address' PROM, 74S288 (IC11) is a 32byte one time programmable device which is programmed by ARCOM with a valid IEEE Ethernet address. This information is mapped twice into the 16 bytes above the I/O base address. Only the higher image should be used (ie I/O base +08H to I/O base +0FH) for compatibility with comparable PC-bus boards.

The format of the 'burned in address' is shown by the following I/O register map :

Offset	Write	Read
0	Board Control Register	Do Not Use
1-7		Do Not Use
8		Burned in address byte 0
9		Burned in address byte 1
A		Burned in address byte 2
B		Burned in address byte 3
C		Burned in address byte 4
D		Burned in address byte 5
E		Type Byte
F		Checksum Byte
10 - 1F		8390 Controller Registers

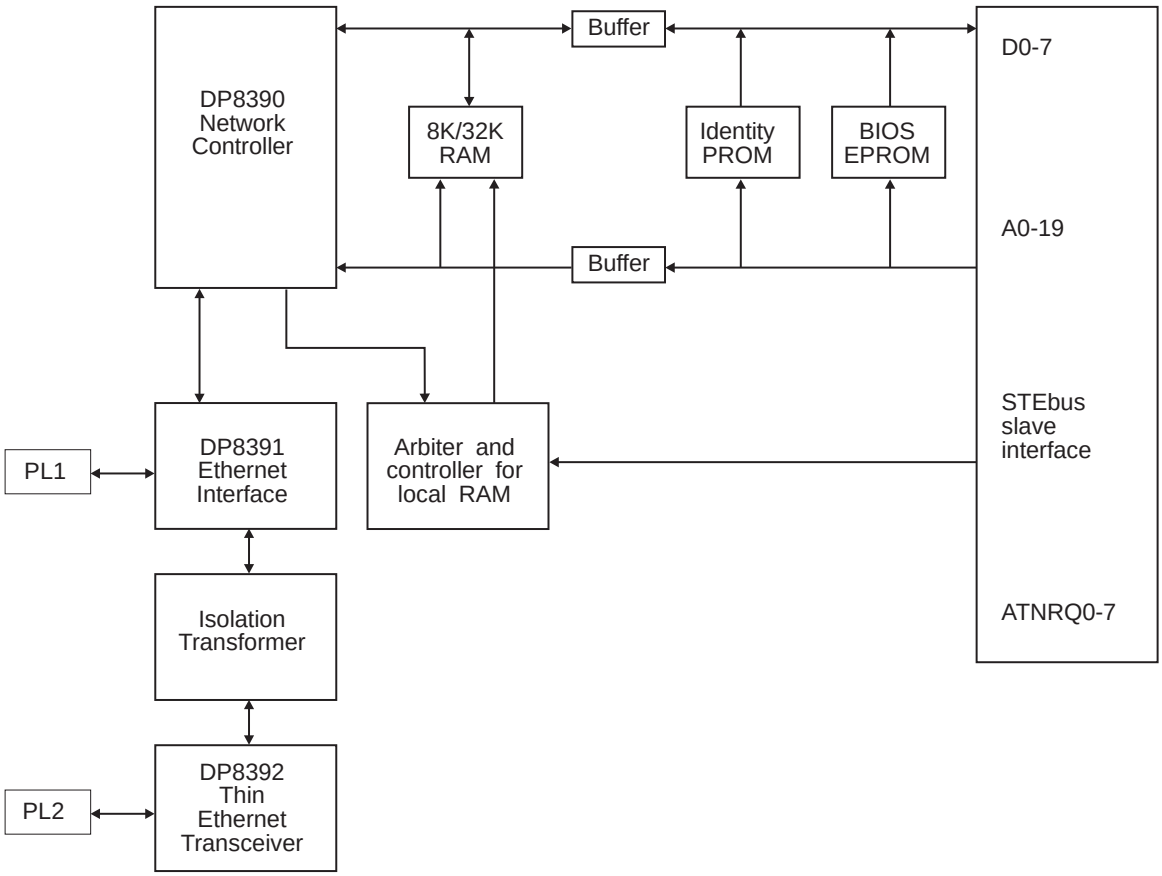
The DP8390 NIC and DP8391 SNI both require a 20MHz clock input to drive the 10 Mbits/s Ethernet transmission. The clock has a high specification in order to comply with the IEEE 802.3 limit of 0.01% absolute accuracy. The tolerance must be +-0.001% @ 25deg.C with a stability of +-0.005% in the range 0-70 deg.C. This is achieved by using the crystal module, XL1. The 8391 SNI provides the Manchester data encoding and decoding for the IEEE802.3 Ethernet. The device includes the differential Tx data line drivers (with 270R pull-down resistors on the outputs, R1, R4) and differential line receivers for the Rx and collision detect (CD) signals. These input lines must be terminated if used with the standard 78 ohm AUI cable with 78 ohm loads. This is provided by two 39R resistors in series which are then by-passed to GND.

The AUI cable output can be used to power the external MAU transceiver unit. This +12v (@ 300mA typ.) supply is connected if Link 13 is made.

The two LED's are useful for monitoring Ethernet traffic through the board. LED1 (red) and LED2 (green) indicate activity on the Tx and Rx data lines respectively. The RxD and TxD lines of the NIC are connected to two pulse stretching monostables (IC24) configured from four high drive current nand gates.

In order to satisfy the IEEE 802.3 specification for Cheapernet, the 8392 Coaxial Transceiver Interface must be fully isolated from the main Ethernet interface. This is achieved using pulse transformers (T1) to connect the signal lines (CD,Tx and Tx) and a DC/DC converter (IC15) to provide the isolated -9v supply. This device is a 2W, +5V to -9V converter.

Functional Block Diagram



Section 5. Troubleshooting

Board does not appear in memory or appears at the wrong address	Check address jumpers LK 12 locates the I/O block LK 3 defines the 8K RAM A19 LK6,7 locates the 8K ROM socket LK2 ,4,9,10 defines the ROM size Ensure that the correct RAM base address has been written to the control register. When setting the address jumpers :- Install a jumper for a zero address bit Remove a jumper for a one address bit
Interrupts do not work correctly	Check that the interrupt links are correctly set, note that SETHER does not support vectored interrupts.
Network will not configure	1. Check that at least one other node is connected to the network and it is switched on. 2. Check that the network cabling is not faulty. 3. Check that 50Ω terminators are fitted at both ends of the network.
Network configures but board does not appear on the network	Check that the node ID is correctly set .The identify PROM - IC11 can be read in I/O SPACE.

Appendix A. Component List

IC1,16	ACT245
IC2,5,8,13,19,24	HCT244
IC3	HCT14
IC7	HCT273
IC6,4,21	HCT688 Comparators
IC12	PAL16L8
IC10	PAL22V10 DATACK functions
IC9	BIOS ROM Socket
IC11	74S288 Ethernet Identity PROM
IC18	PAL16R4
IC14	6264/55257 8K/32K RAM (32K fitted as standard)
IC15	NMB0509 +5V to -9V DC-DC converter
IC20	DP8390 Ethernet Network Interface Controller
IC24	ACT00
IC22	DP8391 Ethernet Serial Network Interface
IC17	ACT373
IC23	DP8392 Ethernet Co-axial Transceiver Interface (for thin-wire networks)
TR1,2	2N7000 N-channel Enhancement MOSFET
D1	1N4148
LED1	0.1 inch Red LED
LED2	0.1 inch Green LED
R6,15	820K
R14	10k
R13	330R
R17	130R
R16	1M0
R7,8,10,11	39R
R2	1k0
R1,4	270R
RP1-4	4k7 C SIL
R3,5,9,12	510R
C1,2,9,14,18	22µF 16V Tantalum
C3-8,10-13,15,17,19, 21-25,29	100nF 25V Ceramic
C12	47µF 16V Aluminium
C20,26,27,28	10nF 25V Ceramic
C16	10nF 1kV Ceramic
T1	Quad 1:1 DIL-package isolation transformers
XL1	20MHz Oscillator Module

Appendix B. STE Bus Connections

PL1 is the 64-way STEbus connector.

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

Appendix C. Specification

Operating temperature	5°C to 55°C
Power consumption	+5v 1.0A (typical) +12v (depends on the MAU transceiver)
Bus	STEbus
I/O	32 bytes
Memory	8K/32K RAM packet buffer 8K/16K/32K/64K BIOS extension EPROM Recommended EPROM type: 27128 Maximum access time: 250ns
Interrupts	ATNRQ0-7*, no bus-vectored acknowledge

Appendix D. Example Program

```
/* SETHMON.C: Simple Ethernet Monitor  JPB 9/8/91
   This utility takes in packets from the network and displays them.
   Written in Aztec/Borland C (small model), low-level drivers in ETHER.C
   For use on an Arcom PC with SETHER card.
   Compile & link using:
         Aztec:   c sethmon.c ether.c
         Borland: bcc sethmon.c ether.c
   To use on a target system, replace 'while(!kbhit())' with 'while (1)', and
   link in the standard target files.

   Copyright (c) Arcom Control Systems Ltd 1991 */

#define ADRLEN 6                        /* No of bytes in ethernet address */
#define MAXDATA 1500           /* Maximum Ethernet data size */

/* Data type definitions */
typedef unsigned long   uint32;         /* 32-bit unsigned */
typedef unsigned short  uint16;         /* 16-bit unsigned */
typedef unsigned char   uint8;          /* 8-bit unsigned */

/* Data structures */ struct dlinkh {          /* Datalink (Ethernet) header */
        uint8   dadr[ADRLEN],   /* Destination Enet addr */
                sadr[ADRLEN];   /* Source Enet addr */
        uint16  type;           /* Enet packet type/length field */
        };                      /* MOST SIGNIFICANT BYTE IS FIRST! */
typedef struct dlinkh DHEAD;

struct dlinkp {                 /* Datalink (Ethernet) packet */
        DHEAD   d;              /* Datalink header */
        uint8   data[MAXDATA];  /* Data area */
        };
typedef struct dlinkp DPKT;

DPKT rxd;                       /* Rx datalink packet buffer */
extern int promisc;               /* Promiscuous physical addr flag */
extern unsigned char myeth[];   /* My ether addr (from ROM) */

/* Prototypes */
void swap(uint16 *p);
void pr6byt(uint8 *str);
void hexdump(uint8 *buff, int len);
int getpack(void *buff);

void main() {
    uint16 type, len;
    printf("Simple Ethernet Monitor\n");
    promisc = 1;                                /* Enable promiscuous mode */
```

```
    if (! etinit()) {                          /* Initialise hardware */
        printf("ERROR: can't access Ethernet card\n");
        exit(0); }
    printf("My Ethernet address is ");         /* Print Burned In Addr */
    pr6byt(myeth);
    printf("\nWaiting for packet... press any key to exit\n");
    while (!kbhit()) {                         /* Loop until key hit */
        if ((len = getpack(&rxd))0) {          /* If packet recd.. */
            printf("From "); pr6byt(rxd.d.sadr);   /* Print header */
            printf(" to "); pr6byt(rxd.d.dadr);
            type = rxd.d.type; swap(&type);    /* Byte-swap type.. */
            printf(" type/len %04X hex\n", type);  /* ..before printing */
            hexdump(rxd.data, len-sizeof(DHEAD));  /* Dump rest of packet */
        }
    }
}

void swap(p)            /* Swap bytes in int */
uint16 *p; {
    uint8 c, *q;
    q = (uint8 *)p;
    c = *q; *q = *(q+1); *(q+1) = c;
}

void pr6byt(str)        /* Print a 6-byte ethernet address */
uint8 *str; {
    int i;
    if (*(str+5) == 0xff) {
        printf("----BROADCAST----"); return; }
    printf("%02x", *str++);
    for (i=1; i<ADRLEN; i++) printf(":%02x", *str++);
}

#ifndef __BORLANDC__
kbhit() {
                       /* Return true if key hit */
    return(bdos(0xb, 0, 0));
}
#endif

void hexdump(buff,len)          /* Print hex dump of buffer */
uint8 *buff; int len; {
    int i, j, m, n;
    uint8 c, str[17];
    n = len/16;
    for (i=0; i<=n; i++) {
        if (len>0) {
            printf("  %04x: ", i*16);
            for (j=0; j<16; j++) {
                if (len>0) {
                    if (j==8) printf(" ");
```

```
                    printf("%02x ", c=buff[i*16+j]);
                    str[j] = (c>=' ' && c<='~') ? c : '.'; }
                else {
                    printf("   "); str[j] = 0; }
                len--; }
            str[j] = 0; printf(" %s\n", str); }
    }
}
```

```
/* SETHER/WD8003E/WD8003EB Ethernet hardware drivers   JPB 9/8/91 */
/* Suitable for Aztec or Borland 'C' compilers (small model) */
/* Copyright (c) Arcom Control Systems Ltd. 1991 */

#include "ETHER.H"

#define IOBASE 0x280       /* I/O base address for Ethernet card */
#define RSEG 0xd000        /* Segment for dual-port RAM buffer */

#ifdef __BORLANDC__             /* If Borland 'C'... */
        /* ..all segments and offsets in opposite order to Aztec! */
unsigned char peekb(unsigned seg, unsigned off);
#define peekb_(off,seg) peekb(seg,off)
#define peekw_(off,seg) peek(seg,off)
#define movblock(o1,s1,o2,s2,len) movedata(s1,o1,s2,o2,len)
#define _dsval _DS

#else                   /* If Aztec 'C'... */
#define peekb_ peekb
#define peekw_ peekw
extern int _dsval;         /* Data seg value from linker */
#endif

#define MAXPACK 1514       /* Maximum packet size (excl hardware bytes) */
#define MINPACK 60         /* Minimum packet size */

unsigned char myeth[6];    /* My Ethernet addr (from ROM) */
int promisc=0;             /* Flag to enable promiscuous mode */

/* Prototypes */
void etopen(unsigned char s[], unsigned addr);
int getpack(unsigned char buff[]);
int putpack(unsigned char pack[], unsigned len);
int egetb(int port);
void eputb(int port, int byt);
void rpage(int n);
int wrapp(int page);

int etinit() {             /* Initialise card, return 0 if error */
    int i, sum=0;
    for (i=0; i; i++)                      /* Get 6 byte h/ware addr.. */
      sum += (myeth[i] = egetb(AROM+i));      /* ..keeping sum of bytes */
    sum += egetb(AROM+6);                    /* Add on 7th, 8th bytes */
    sum += egetb(AROM+7);
    if ((sum & 0xff) != 0xff) return(0);      /* ..total should be FF */
eputb(W83CREG, MSK_RESET);                /* Reset card */
    eputb(W83CREG, 0);
    etopen(myeth, RSEG);               /* Set up card */
    return(1); }
```

```
void etopen(s, addr)          /* Set up hardware */
unsigned char s[];            /* Ethernet addr */
unsigned addr;                /* Mem addr */
{
    int i;
    eputb(W83CREG, addr>9);           /* Set dual-port RAM addr */
    rpage(0);                         /* Reg page 0, abort remote DMA */
    eputb(DCR,  0x48);                /* FIFO 8 bytes, no loopback */
    eputb(RBCR0,0); eputb(RBCR1,0);   /* Clear remote byte count */
    eputb(RCRW,  0x20);               /* Monitor mode */
    eputb(TCR,  0);
    eputb(PSTOP,STOP_PG); eputb(PSTART,STRT_PG);   /* Mem stop, start */
    eputb(BNRY, STRT_PG);             /* Mem boundary */
    eputb(ISR,  -1); eputb(IMR,0);    /* Clear & mask all interrupts */
    rpage(1);                         /* Reg page 1 */
    for (i=0; i; i++) eputb(PAR0+i, s[i]);   /* Set Phys addr */
    for (i=0; i; i++) eputb(MAR0+i, 0);      /* Set Mcast addr to 0 */
    eputb(CURR, STRT_PG+1);           /* Current page = start+1 */
    rpage(0); eputb(CMDR, 0x22);      /* Reg page 0, start 8390 */
    eputb(RCRW, promisc ? 0x14 : 0x04);/* Accept Bcast, also promiscuous? */
    }

/* Get packet into buffer, return length (0 if none received) */
getpack(buff)             /* Length includes dest addr, srce addr, type */
unsigned char buff[]; {
    unsigned curr, bdry, raddr;
    unsigned stat, plen, p1, p2;
    rpage(1); curr = egetb(CURR);  /* Page after the incoming data */
    rpage(0); bdry = egetb(BNRY);  /* Page of data already read */
    bdry = wrapp(bdry+1);          /* Move up to unread page */
    if (curr == bdry) return(0);   /* Return 0 if nothing received */
    raddr = bdry<8;                /* Calculate addr of unread page */
    stat = peekb_(raddr, RSEG);    /* Get status and length of packet */
    plen = peekw_(raddr+2,RSEG)-HWB;   /* Deduct 4 hardware bytes from len */
    if (!(stat & 1) || plenMAXPACK)   /* If error in packet, ignore it */
      plen = 0;
    else {
                            /* No error in packet, copy it */
        if (plen+raddr  STOP_RAM) {   /* If packet wraps around end of RAM */
            p1 = STOP_RAM-raddr-HWB;   /* ..copy it in 2 parts */
            p2 = plen - p1;            /* Only 1st part has hardware bytes */
movblock(raddr+HWB, RSEG, buff, _dsval, p1);
            movblock(STRT_RAM, RSEG, buff+p1, _dsval, p2); }
        else {                         /* If no wrap, copy whole packet */
            movblock(raddr+HWB, RSEG, buff, _dsval, plen); }
        }
    bdry = wrapp(peekb_(raddr+1,RSEG)-1);  /* Move pointer to next packet */
    eputb(BNRY, bdry);
    return(plen); }                       /* Return packet length, or 0 */
```

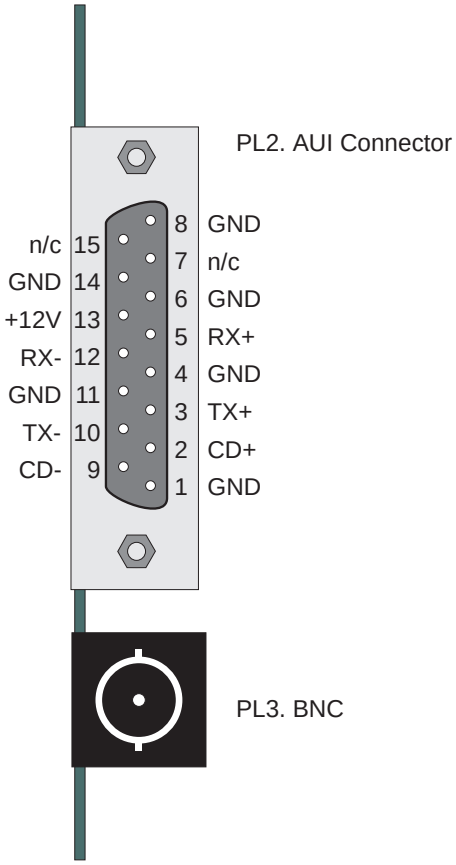
```
putpack(pack, len)     /* Send Ethernet packet, return 0 if error */
unsigned char pack[];  /* See above for packet format */
unsigned len; {        /* Length includes dest addr, srce addr, type */
    unsigned timout = 60000;               /* Arbitrary timout value */
    while(egetb(CMDR) & 4)                 /* Ensure transmitter is idle */
      if (! timout--) return(0);           /* ..return 0 if timout */
    if (len>MAXPACK) return(0);            /* Error if packet too long */
    movblock(pack, _dsval, 0, RSEG, len);  /* Copy packet into RAM page 0 */
    if (len<MINPACK) len = MINPACK;        /* Must be = >60 bytes */
    eputb(TBCR0, len); eputb(TBCR1, len>>8);/* Set length regs */
    eputb(TPSR, 0);                        /* Select transmit page 0 */
    eputb(CMDR, 0x24);                     /* Transmit */
    return(len); }

egetb(port) {                  /* Read byte from card register */
   return(inportb(IOBASE+port)); }

void eputb(port,byt) {         /* Write byte to card register */
    outportb(IOBASE+port,byt); }

void rpage(n) {                /* Select 8390 register page */
    outportb(CMDR+IOBASE, (n<6)+0x20); } /* Abort remote DMA */

wrapp(page)                    /* Return a wrapped value of Rx memory page */
int page; {
    if (page = >STOP_PG) return(STRT_PG);      /* Wrap if overflow */
    if (page <  STRT_PG) return(STOP_PG-1);    /* Wrap if underflow */
    return(page); }
```

```
/* Equates for SETHER/WD8003E/WD8003EB Ethernet cards   JPB 23/6/90 */
/* Copyright (c) Arcom Control Systems Ltd. 1990 */

/* SETHER/8003E/8003EB control register operations */
#define W83CREG         0       /* Control reg offset from base */
#define MSK_RESET       0x80    /* Control bit to reset LAN controller */
#define MSK_ENASH       0x40    /* Control bit to enable shared mem */
#define MSK_DECOD       0x3Fh   /* Bits to set mem address lines A18-A13 */
                                /* Address line A19 assumed to be 1 */

/* On-board Ethernet address ROM */
#define AROM    8               /* ROM offset from I/O base addr */
/* To keep compatibility with all 3 boards, do not use ROM image at base+0 */

/* 8390 LAN Controller (page0) register offset for read and write */
#define CMDR    0x10            /* command register for read & write */
#define CLDA0   0x11            /* current local dma addr 0 for read */
#define PSTART  0x11            /* page start register for write */
#define CLDA1   0x12            /* current local dma addr 1 for read */
#define PSTOP   0x12            /* page stop register for write */
#define BNRY    0x13            /* boundary reg for rd and wr */
#define TSR     0x14            /* tx status reg for rd */
#define TPSR    0x14            /* tx start page start reg for wr */
#define NCR     0x15            /* number of collision reg for rd */
#define TBCR0   0x15            /* tx byte count 0 reg for wr */
#define FIFO    0x16            /* FIFO for rd */
#define TBCR1   0x16            /* tx byte count 1 reg for wr */
#define ISR     0x17            /* interrupt status reg for rd and wr */
#define CRDA0   0x18            /* current remote dma address 0 for rd */
#define RSAR0   0x18            /* remote start address reg 0  for wr */
#define CRDA1   0x19            /* current remote dma address 1 for rd */
#define RSAR1   0x19            /* remote start address reg 1 for wr */
#define RBCR0   0x1A            /* remote byte count reg 0 for wr */
#define RBCR1   0x1B            /* remote byte count reg 1 for wr */
#define RSR     0x1C            /* rx status reg for rd */
#define RCRW    0x1C            /* rx configuration reg for wr */
#define CNTR0   0x1D            /* tally cnt 0 for frm alg err for rd */
#define TCR     0x1D            /* tx configuration reg for wr */
#define CNTR1   0x1E            /* tally cnt 1 for crc err for rd */
#define DCR     0x1E            /* data configuration reg for wr */
#define CNTR2   0x1F            /* tally cnt 2 for missed pkt for rd */
#define IMR     0x1F            /* interrupt mask reg for wr */

/* 8390 LAN Controller (page1) register offset for read and write */
#define PAR0    0x11            /* physical addr reg 0 for rd and wr */
#define CURR    0x17            /* current page reg for rd and wr */
#define MAR0    0x18            /* multicast addr reg 0 for rd and WR */
```

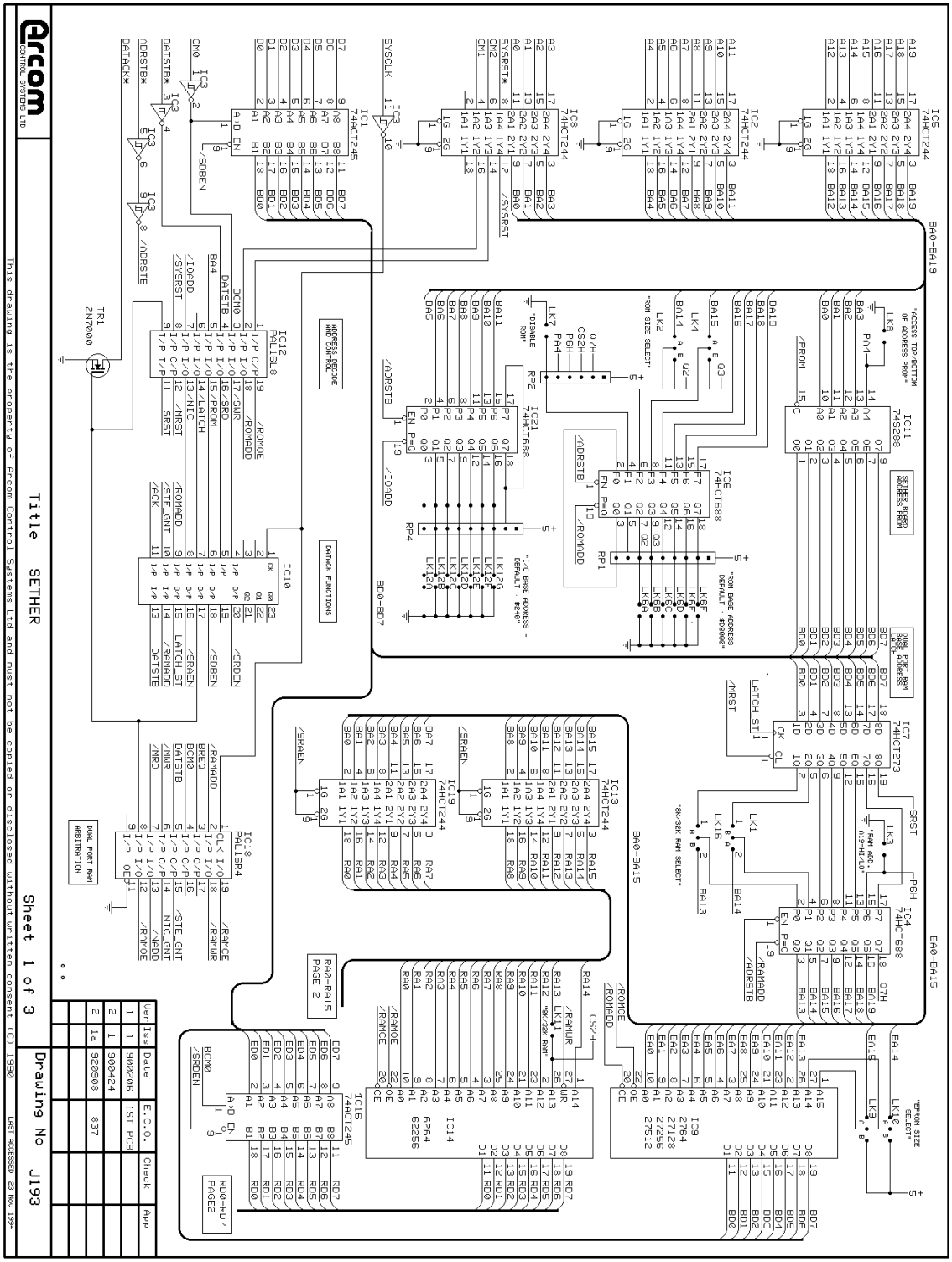
```
/* Buffer Length and Field Definition Info */
#define HWB     4               /* Number of bytes in hardware header */
#define STRT_PG 6               /* Receive buffer start page */
#define STOP_PG 32              /* Receive buffer end page */
#define STRT_RAM (STRT_PG<8)    /* Receive buffer start addr */
#define STOP_RAM (STOP_PG<8)    /* Receive buffer end addr */
```

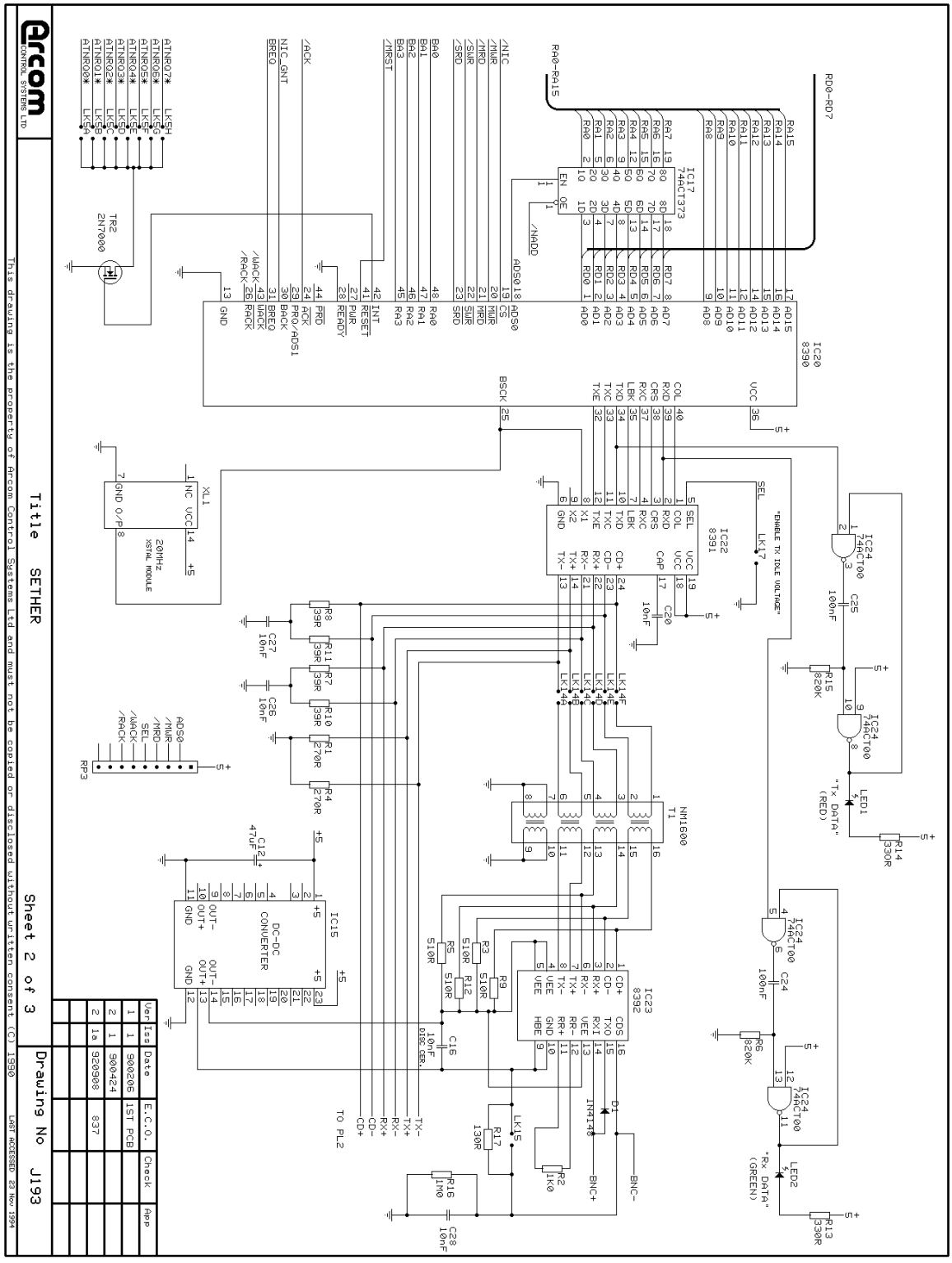
Appendix E. AUI Connector Pin Assignments

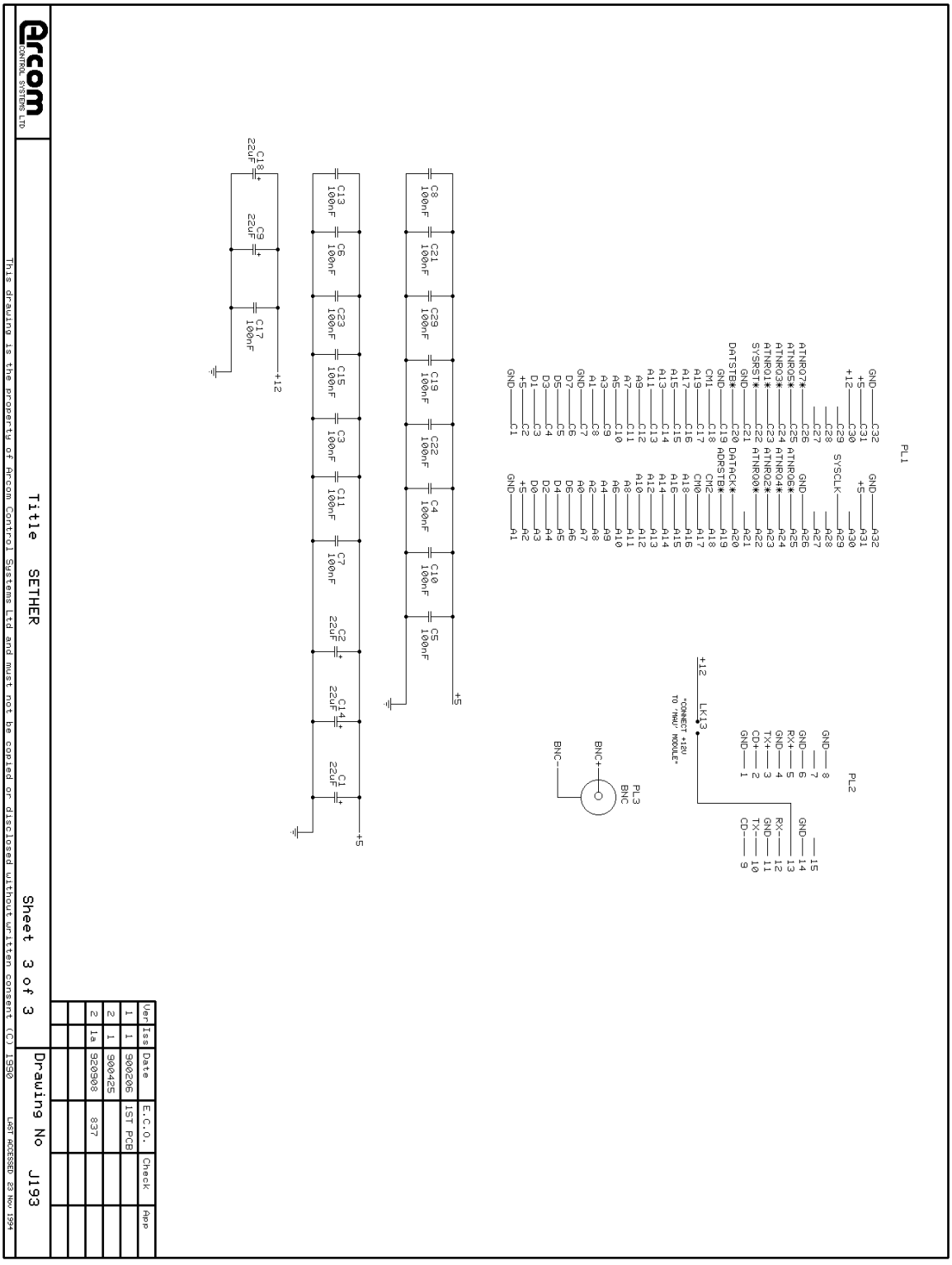
PL2. AUI Connector



Appendix E. Circuit Diagrams




Arcom CONTROL SYSTEMS LTD
Title SETHER
Sheet 2 of 3
Drawing No J193
This drawing is the property of Arcom Control Systems Ltd and must not be copied or disclosed without written consent (C) 1990
LAST ACCESSED 23 Nov 1994
Ver Iss Date E.C.O. Check App
1 1 900206 1ST PCB
2 1 900424
2 1a 920908 837
RA0-RA15
RD0-RD7
IC17 74HCT373
IC20 8390
IC22 8391
IC23 8392
IC24 74HCT00
IC15 DC-DC CONVERTER
NM1600 T1
TR2 2N7000
XL1 20MHz XSTAL MODULE
RP3
LED1 "Tx DATA" (RED)
LED2 "Rx DATA" (GREEN)
"ENABLE TX IDLE VOLTAGE"
TO PL2
/ACK
NIC_GNT
BREQ
/MRST
/NIC
/MWR
/MRD
/SWR
/SRD
/WACK
/RACK
/NADD
BSCK
BA0
BA1
BA2
BA3
RA0
RA1
RA2
RA3
INT
RESET
PWR
READY
PRD
PRO/ADS1
BACK
ADS0
CS
SEL
VCC
GND
BNC+
BNC-
D1 1N4148
C16 10nF DISC CER.
C12 47uF
C20 10nF
C24 100nF
C25 100nF
C26 10nF
C27 10nF
C28 10nF
R1 270R
R2 1K0
R3 510R
R4 270R
R5 510R
R6 820K
R7 39R
R8 39R
R9 510R
R10 39R
R11 39R
R12 510R
R13 330R
R14 330R
R15 820K
R16 1M0
R17 130R
LK1-LK5
LK7


Arcom CONTROL SYSTEMS LTD
Title SETHER
Sheet 3 of 3
Drawing No J193
This drawing is the property of Arcom Control Systems Ltd and must not be copied or disclosed without written consent (C) 1990
LAST ACCESSED 22 Nov 1994
Ver Iss Date E.C.O. Check App
1 1 900206 1ST PCB
2 1 900425
2 1a 920908 837
PL1
PL2
PL3 BNC
LK13
"CONNECT +12V TO 'MRU' MODULE"
C1 22uF
C2 22uF
C3 100nF
C4 100nF
C5 100nF
C6 100nF
C7 100nF
C8 100nF
C9 22uF
C10 100nF
C11 100nF
C12 100nF
C13 100nF
C14 22uF
C15 100nF
C17 100nF
C18 22uF
C19 100nF
C20 DATACK*
C21 100nF
C22 100nF
C23 100nF
C24 ATNR04*
C25 ATNR06*
C26
C27
C28
C29 100nF
C30
C31
C32
+5
+12
GND