kontron
... always a Jump ahead!

VSIO

STANDART I/O MODULE

FOR THE VMEbus

ID: 02363 Index: 0100

MANUAL

SERIES

PRODUCT HISTORY:

INDEX:	COMMENT:	DATE:
01/05	ADDRESS MODIFIER ENHANCEMENTS	02-01-87
01/06	TIMING ENHANCEMENTS ADDRESS MODIFIER CHANGE ADDED JUMPER INFO	29-10-87
0100	TITLEPAGE UPDATED	SEPTEMBER 1997

Unpacking and Special Handling Instructions

This PEP product is carefully designed for a long and fault-free life; nonetheless, its life expectancy can be drastically reduced by improper treatment during unpacking and installation.

Observe standard anti-static precautions when changing piggybacks, ROM devices, jumper settings etc. If the product contains batteries for RTC or memory back-up, ensure that the board is not placed on conductive surfaces, including anti-static plastics or sponges. These can cause shorts and damage to the batteries or tracks on the board.

When installing piggybacks, switch off the power mains.

Furthermore, do not exceed the specified operational temperature ranges of the board version ordered. If batteries are present, their temperature restrictions must be taken into account.

Keep all the original packaging material for future storage or warranty shipments. If it is necessary to store or ship the board, re-pack it as it was originally packed.

TABLE OF CONTENTS

CHAPTER 1
USING THIS MANUAL

1.1 MANUAL CONTENTS

This manual gives a physical and functional description of the VSIO module, including specifications, a block diagram, example jumper settings, and gives instructions for configuring the jumpers and installing the module.

Chapter Objectives:

Chapter 1: Using This Manual - describes the contents and organization of this manual. Also, this chapter refers you to related publications.

Chapter 2: Identifying The VSIO - gives an overview of the VSIO and describes its specific features such as: board specifications and access times.

Chapter 3: Configuring and Installing The VSIO - provides a step-by-step procedure for setting the jumpers for your specific operation and instructs on how to install the module.

Appendix

PAL Listings
Connector Pin Assignments
Jumper Configurations
Parts List
Ordering Information
Layout and Schematics

We assume that you have some knowledge of the VMEbus, however, no expert knowledge is required for you to prepare and use the VSIO.

1.2 RELATED PUBLICATIONS

For more information related to the VSIO, please refer to:

VMEbus Specifications, PEP Modular Computers

CHAPTER 2

IDENTIFYING THE VSIO

This chapter includes a discussion of the VSIO's capabilities and features. It also includes a physical description including specifications and a block diagram. You should read this chapter to familiarize yourself with the specific characteristics of the VSIO.

2.1 PRODUCT OVERVIEW

The VSIO was designed as a cost-effective VMEbus solution for your serial and parallel I/O needs.

The VSIO serves as a serial and parallel interface module to the VMEbus. It provides standard interfaces such as RS232C, RS422, current loop or fiber optic. The parallel interface allows the adaption of a centronics printer. Programmable transfer rates, independent for each channel, serve a wide range of applications. The module is designed for real-time applications with individual interrupt generation.

The VSIO is part of our PEP family of VME modules designed for industrial applications in harsh environments. All PEP modules are in single-height Eurocard format using standard P1/J1 connector to reduce volume and increase ruggedness.

2.2 VSIO FEATURES

The VSIO supplies you with the following features:

* Two independent serial interfaces with high modularity when using piggybacks for individual I/O requirements: RS232C, RS422, 20mA current loop or fiber optic

* Parallel interface with two 8-bit TTL I/O ports and four control lines using a PIA

* Powerful Z8530 multi-protocol serial controller (SCC)

* Full duplex operation on both serial ports (bi-directional/simultaneous)

* Asynchronous and synchronous (bit and byte) operation

* Synchronization logic allows baud rates up to 307 KBaud (with piggybacks up to 38 KBaud).

* Programmable baud rate from 50 baud up to 38,400 KBaud (with piggybacks, up to 38 KBaud).

* External baud rate input for both serial ports

* Asynchronous mode of operation:
  - Character length: 5 to 8 bit
  - Parity: odd, even, none
  - Stop bits: 1, 1.5, or 2

* Byte synchronous operation:
  - Character synchronization
  - One or two sync characters
  - Automatic CRC generation and checking (CRC-16)
  - IBM bisync compatible

* Bit synchronous operation:
  - SDLC/HDLC flag generation and recognition
  - 8 bit address recognition
  - Automatic zero bit insertion and deletion
  - Automatic CRC-generation and checking (CCITTT-16)
  - CCITT X.25 compatible

* Error detection:
  - Framming
  - Overrun
  - Parity

* Programmable interrupt level IRQ1* to IRQ7*

* Programmable interrupt vector address

* Optional driver Centronics interface for printer capabilities

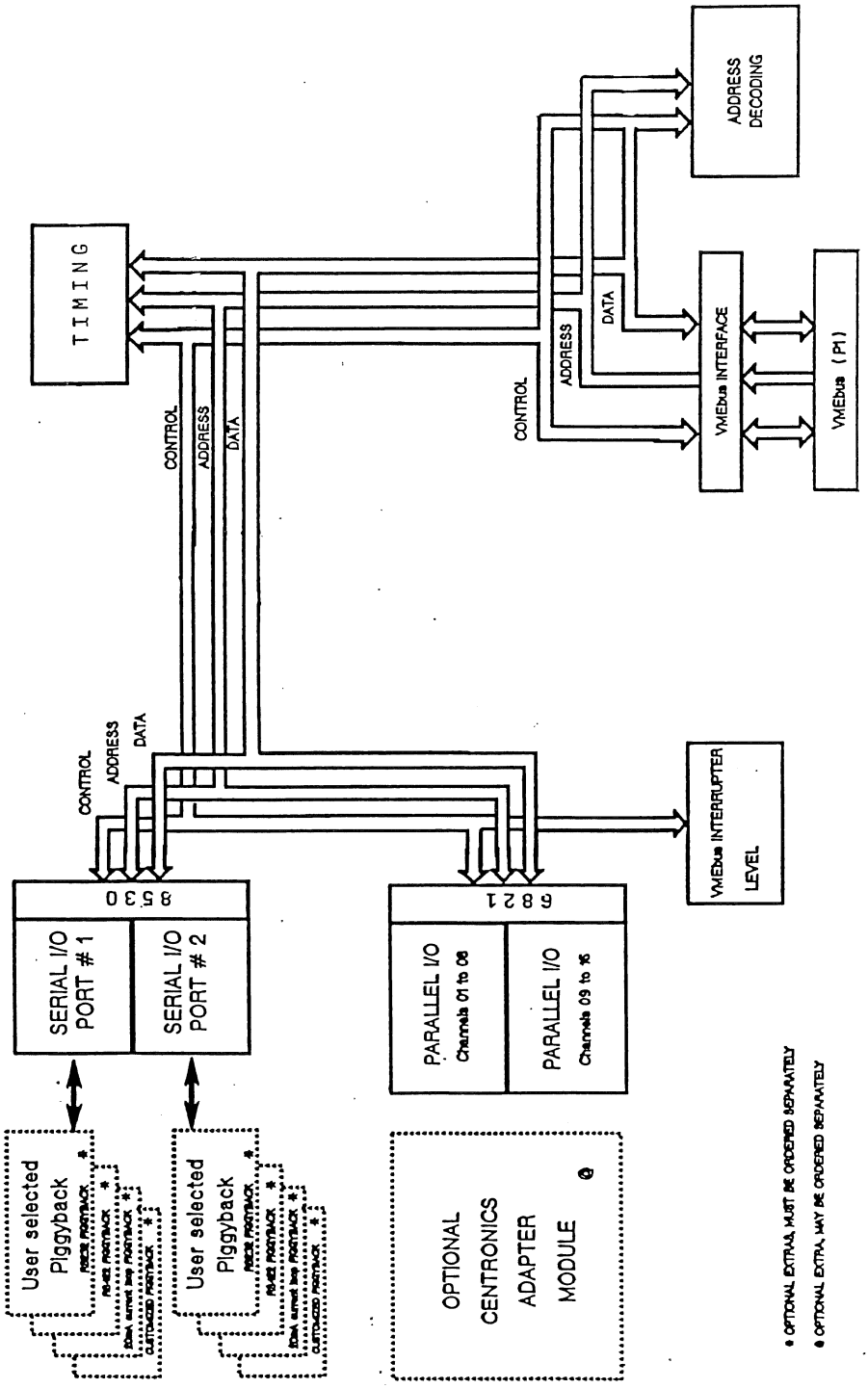
* Address modifier combinations for AM0 to AM5 selectable via jumpers

2.3 VSIO SPECIFICATIONS

VMEbus interface:	Slave A24/D16, A16/D16, A16/D8, A24/D8
Address Modifier:	jumper selectable Standard/User Data or Short I/O
Serial I\O:	Z8530 2 independent serial channels with handshake lines
Transfer rates Synchronous: Asynchronous:	 307 KBaud Depends on Piggybacks chosen 50 - 38,400 KBaud
Protocols:	SDLC/HDLC, NRZ, NRZI
Serial I\O Piggybacks:	user selectable, mixable, any two from the PEP piggyback family: RS232C (50 to 19200 Baud) RS422 (50 to 38400 Baud) 20mA Current Loop (50 to 9600 B)
Parallel I\O:	MC6821 2x8-bit digital I/O 4 handshake lines
Interrupt Structure:	Jumper selectable, IL1 - 7
Transfer rate:	Up to 307 KBaud
Operating temperature:	0 to 70 degrees C (standard) -40 to 85 degrees C (extended)
Operating humidity:	0 to 95%, non condensing
Power requirements:	1A typ. at +5VDC 50mA typ. at +12VDC 50mA typ. at -12VDC
Physical configuration:	SINGLE
Board size:	Single Eurocard, 100x160mm

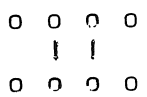
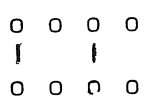
VMEbus connector:	DIN41612 style C, 96 contacts
Frontpanel width:	4 TE (20.3 mm)
Frontpanel connector:	2x9 pin sub-D connector for serial I/O

2.4 VSIO BLOCK DIAGRAM



2.5 ADDRESS MODIFIER SELECTION

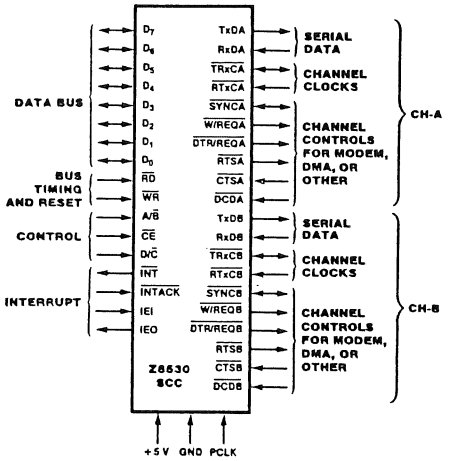
The VSIO is capable of implementing two different address modifier modes, therefore providing full functionality for all VME systems. The "PEP Old" mode is compatible to previous versions of this module and therefore allows you to replace the previous versions.

Address Modifier Function	AM codes	Jumper B3 B6
Standard Supervisory/ Non-privileged Data Access PEP Old	39/3D 00	
Short Supervisory/ Non-privileged Data Access	29/2D	

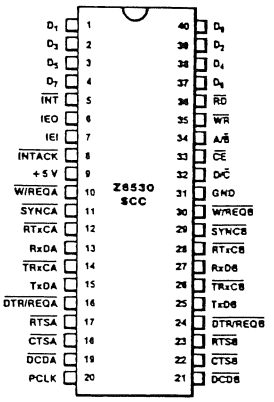
You may not use any other combinations than those shown above. By reprogramming the appropriate IC, you may change the address modifiers to fit your needs.

2.6 SERIAL INTERFACE

The Z8530 (SCC) is a fully programmable multi-protocol serial controller allowing you, in conjunction with an on-board synchronization logic, to program baud rates from 50 Baud up to 307 KBaud. For details of the baud rate generator of the Z8530 refer to chapter 3.9 of the Z8530 manual.



Z8530 SCC Pin Functions



40-pin Dual-In-Line Package (DIP) (Pin Assignments)

2.6.1 Clock Select Register: Transmission Rates (Frequency in MHz)

For maximum flexibility, the VSIO allows you to select multiple clock sources and various baud rates up to 1.2288 MBaud by programming the clock select register in base address +9 of the VSIO. However, to ensure a safe transmission, we recommend only synchronous rates of up to 307 KBaud be used. For asynchronous data, see the serial piggyback section of the specifications.

Divider programmed for the SCC				Bit pattern in Clock Select register (Base address+9)	
x1	x16	x32	x64	MSB Channel B D7...D5	LSB Channel A D4...D0
9.8304	0.6144	0.3072	0.1536	0 1 1 0	0 1 1 0
4.9152(*)	0.3072	0.1536	0.0768	0 1 0 1	0 1 0 1
2.4576	0.1536	0.0768	0.0384	0 1 0 0	0 1 0 0
1.2288	0.0768	0.0384	0.0192	0 0 1 1	0 0 1 1
0.6144	0.0384	0.0192	0.0096	0 0 1 0	0 0 1 0
0.3072	0.0192	0.0096	0.0048	0 0 0 1	0 0 0 1
0.1536	0.0096	0.0048	0.0024	0 0 0 0	0 0 0 0
EXT.	EXT.	EXT.	EXT.	0 1 1 1	0 1 1 1

EXT. means that the clock source is supplied via input lines CAEx (connector St6 pin 3) and CABx (connector St5 pin 3).

(*) When the baud rate generator source of the Z8530 is selected as the PCKL input, the transmission rate is 4.9152 MHz. This clock frequency must be considered also when calculating the time constant or baud rates according to table 3-7 in the Z8530 programming manual. Setting the bit 2**3 for channel A respective bit 2**7 for channel B to logical "1" enables the onboard-synchronisation logic for bit-synchronisation (HDLC). For synchronous data transfer it is recommended to use this logic or external synchronous clock. The divider in the SCC must be set to x1.

The interface type for each of the serial ports is determined by the selected piggy-back module. The piggy back module also determines the maximum baud rate which should be programmed.

(2**n means 2 power n)

2.7 PARALLEL INTERFACE

The VSIO parallel interface (PIA 6821) may be used in either the polling mode or with one dedicated interrupt source on CA or CB. You may set the interrupt level for the VMEbus in the control byte.

2.7.1 Control Byte (Base address +8)

Programming the control byte allows the preselection of the interrupt-level of the board.

Bit	7 (MSB)	6	5	4	3	2	1	0 (LSB)
	0	0	0	0	X	V3	V2	V1

X = not used

2.7.2 Programming the Interrupt-level (V1...V3 of Control byte)

V3	V2	V1	Interrupt level
0	0	0	no interrupt
0	0	1	IRQ 1*
0	1	0	IRQ 2*
0	1	1	IRQ 3*
1	0	0	IRQ 4*
1	0	1	IRQ 5*
1	1	0	IRQ 6*
1	1	1	IRQ 7*

2.7.3 Interrupt-Vector-Address (Base address +11)

The VSIO is capable of generating two different interrupt vectors to distinguish whether interrupts are coming from the SCC or the PIA. One vector may be programmed to the interrupt-handler-routine for the SCC. For programming details refer to the SCC datasheet.

An additional vector may be programmed to the interrupt handler routine for PIA. Both interrupts, IRQA and IRQB, use the same vector-address.

2.7.4 PIA - Interrupt-Vector

Bit	7	6	5	4	3	2	1	0
	MSB			Interrupt Vector				LSB

CHAPTER 3

CONFIGURING and INSTALLING THE VSIO

Chapter Objectives:

This chapter describes the necessary jumper settings for proper operation of your VSIO module. It also includes instructions for installing the module in your system

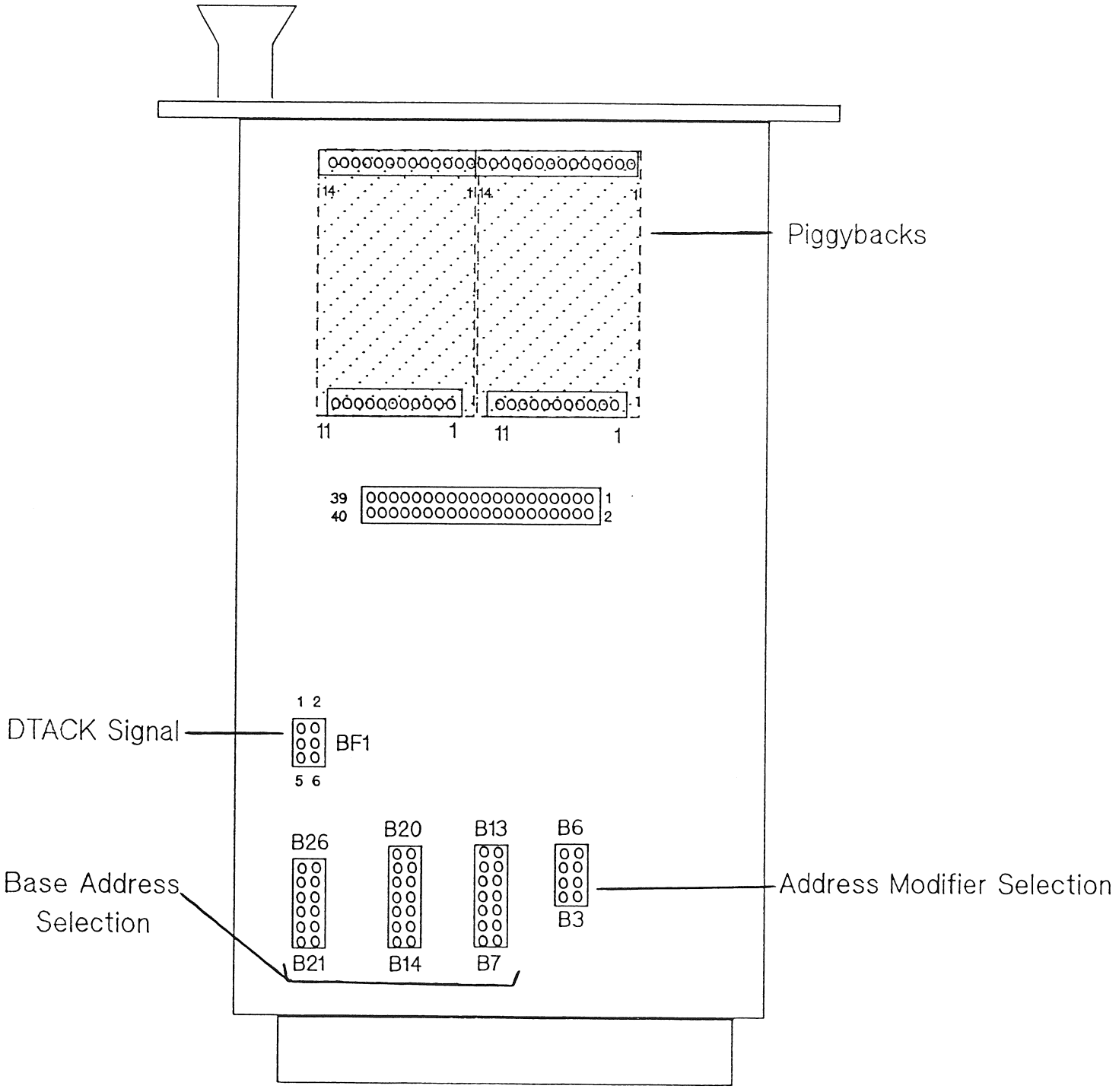
3.1 VSIO JUMPERS

The VSIO has a total of 26 jumpers. These jumpers, along with their functions, are listed in the table below.

Jumper	Function
BF 1	DTACK Selection
B3 - 6	Address Modifier Selection
B7 - 26	Base Address Selection

For a detailed description of each step, please read further in this chapter.

FIGURE 3-1: VSIO JUMPER LOCATIONS



3.2 CONFIGURING THE VSIO

In the following sections, we provide you with the procedures necessary for proper operation of the VSIO. Please read over the summary of these procedures now to get a feel of how to go about configuring the VSIO.

To configure the VSIO for use, follow these steps:

1. Set the **Base Address** of your VSIO (B7 - 26)

2. Choose your **Address Modifier** (B3 - 6)

3. Choose your **DTACK Timing** (BF 1). To assure that your VSIO operates properly with the VMEbus, set BF to 1-2.

4. Initialize your **Software**

The table below describes the terms and abbreviations used in conjunction with setting jumpers.

Jumper Setting	**Abbreviation**	**Meaning**
Jumper closed	set	= log 0 (connection)
Jumper open	open	= log 1 (no connection)
Jumper closed or open	x	= don't care

3.3 SETTING THE BASE ADDRESS

The base address of the VSIO module is selected via jumpers B7 to B26. Beginning at this address the module occupies 12 bytes of addressing space. In addition to the base address a pattern for the address modifiers AM0 to AM6 must be selected via jumpers B1 to B6. The pattern selected must match the pattern which the VMEbus master (i.e. CPU module) is generating when accessing the VSIO module.

Address	Write operation	Read operation	Device
Base address Base address +1	Ch.A data Ch.A CMD/Par.	Ch.A data Ch.A status	serial Chann.A
Base address +2 Base address +3	Ch.B data Ch.B CMD/Par	Ch.B data Ch.B status	serial Chann.B
Base address +4 Base address +5	Port A data Port A control	Port A data Port A control	PIA Port A
Base address +6 Base address +7	Port B data Port B control	Port B data Port B control	PIA Port B
Base address +8 Base address +9	Control Byte Clock select	X X	IRQ-lev. trans- mission rate
Base address +10 Base address +11	X Interrupt vector address	X X	PIA

X = not used, but occupied by the board.

3.3.1 B7...B2: The Base Address

Jumper	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16
Address	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14
Jumper	B17	B18	B19	B20	B21	B22	B23	B24	B25	B26
Address	A13	A12	A11	A10	A09	A08	A07	A06	A05	A04

Jumper set = address bit '0'
Jumper open = address bit '1'

3.3.2 Selecting Your Address Modifier

The VSIO address modifier capabilities enable either Short I/O or Standard I/O access in conjunction with the address lines A23 to A4 (jumper B7 to B26). The PEP Old mode may be used in order that previous versions of the VMPM68KA and VMPM68KB modules can be used.

Jumpers B3 B4 B5 B6	AM Codes	AM Functions
open set set open	00 3D 39	Supervisory Data PEP Old
set open set open	2D 29	Short I/O

Example of Short I/O selection with:

For VM 20: $85000020	--> $85	B3: set.	B4: open.	B5: set.	B6: open.
	00	B07 to B14: set.			
	00	B15 to B22: set.			
	20	B23: set.	B24: set.	B25: open.	B26: set.
For VSBC-4: $fc0020	--> AM code	B3: set.	B4: open.	B5: set.	B6: open.
	$FC	B07 to B12: open.		B13 and B14: set.	
	00	B15 to B22: set.			
	20	B23: set.	B24: set.	B25: open.	B26: set.

Short I/O can be executed with the VSIO if the CPU drives A16 to A23 with a defined pattern during access.

If these address lines are floating during Short I/O, pullup resistors should be used at the inputs (A16 to A23) of the address comparator and jumpers B7 to B14: open.

3.3.3 BF1 Access Time (DTACK)

The access time of the VSIO is the time it takes for a DTACK* signal to be generated and sent to the VMEbus. It is selected via the jumpers of BF1.

	Default		
Access time (ns)	*437*	375	312
Jumper	*1-2*	3-4	5-6

3.2.4 Programming Your Software

The last step in preparing your VSIO for use is the programming of your software. First, we recommend that you program your interrupt vector; secondly, program your bus request.

CHAPTER 4

CONFIGURING/INSTALLING THE VSIO MODULE

Chapter Objectives

This chapter instructs you on how to install the VSIO module in your system.

4.1 CONFIGURING THE VSIO

The VSIO has a variety of possible applications detrmined by software settingd.

4.2 INSTALLING YOUR VSIO

This section describes how to install the VSIO for system operation. Before installing, check that you have the jumpers configured to suit your particular use.

WARNING!

Before installing or removing any boards, always turn off the power to the bus and any external power supplies which may be connected.

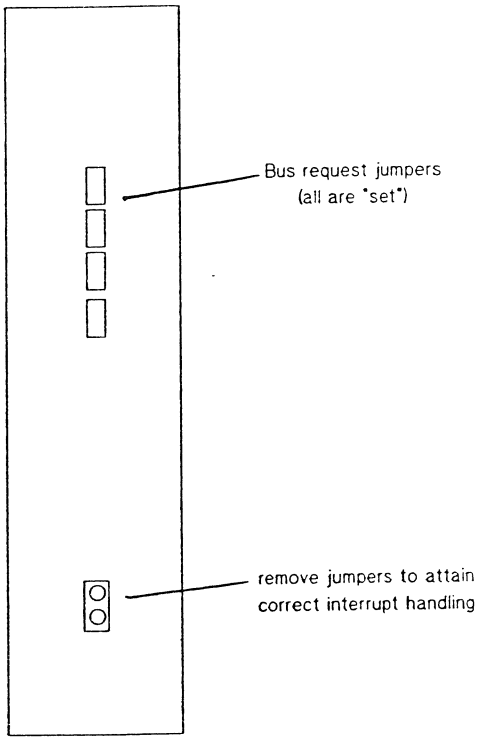
Inserting or withdrawing modules without observing these precautions could result in damage to this or other modules in your system.

You may install the VSIO in any appropriate vacant slot (other than slot # 1) on your VMEbus system. Please ensure observation of the following installation procedure.

1. Inspect the 96 way PCB connector for clean straight pins.
2. Check jumper configuration and clock generator frequency suit your particular application.
3. Ensure all power is off.
4. Choose the desired slot for the VSIO module (other than slot 1) and if necessary reposition other modules carefully.
5. Remove all jumpers (for that particular slot) from the VMEbus backplane, should be 5 in all, 4 at top and 1 in the middle.
6. Placing PCB into card guides push board firmly into place and tighten front panel securing screws.
7. Connect serial I/O to the appropriate socket(s) on the front panel.
8. Restore power only when you are satisfied that all modules are correctly fitted and all connections have been properly made.

If you experience any difficulty in incorporating this module into your system or have any queries regarding this or any other PEP product please do not hesitate to contact your local representative.

An installed VSIO module -
viewed from the VMEbus backplane



APPENDIX

This appendix contains the following information in the following order:

Pinouts of connectors
Glossary of terms
Jumper configurations
Layout and Schematics
Palette record forms for CLUT users

VMEbus CONNEÇTORS

A.1 VMEbus connector P1

Pin No.	Row A signal	Row B signal	Row C signal
1	D00		D08
2	D01		D09
3	D02		D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	
11	GND	BG3OUT*	
12	DS1*		SYSRESET*
13	DS0*		LWORD*
14	WRITE*		AM5
15	GND		A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*		A17
22	IACKOUT*		A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12 V		+12 V
32	+ 5 V	+ 5 V	+ 5 V

A.2 SERIAL INTERFACE

The signals of the two channels of the serial interface are connected to two 15-pinned DSUB connectors. Dependent on the interface-board used (RS232C or RS422), the pinout of the connector is different. On the fiber optic interface there are only the transmit and receive-lines available.

RS232C		
Pin	Signal	Direction
1	GND	out
2	TXD	out
3	RXD	in
4	CTS	in
5	RTS	out
6	NC	-
7	GND	out
8	DCD	in
9	NC	-
10	DTR	out
11	NC	-
12	NC	-
13	NC	-
14	NC	-
15	CEX	-

RS422		
Pin	Signal	Direction
1	GND	out
2	+CEX	in
3	-CEX	in
4	+DCD	in
5	-DCD	in
6	+DTR	out
7	-DTR	out
8	+TXD	out
9	-TXD	out
10	+RXD	in
11	-RXD	in
12	+CTS	in
13	-CTS	in
14	+RTS	out
15	-RTS	out

Current Loop		
Pin	Signal	Polarity
1	GND	
2	RTS	+
3	RTS	-
4	RXD	-
5	RXD	+
6	CTS	-
7	CTS	+
8	+12V ext	
9	+12V int	
10	TXD	+
11	TXD	-
12	DTR	-
13	DTR	+
14	DCD	-
15	DCD	+

The maximum baud rate which is possible for proper data transfer depends on the interface type selected by the piggyback module. The maximum baud rates which should not be exceeded depend also on the line quality / length and the environment.

The specifications are:

20mA current loop 9600 Baud

20mA	------
RS232C	19200 "
RS422	76800 "
Fiber Optic	1.2 MBaud

A.3 PARALLEL INTERFACE

The 16 I/O-lines and additionally 4 control lines are connected to a 40-pin flat ribbon connector St4.

Signal	Pin
CA 1	2
CA 2	4
PA 0	6
PA 1	8
PA 2	10
PA 3	12
PA 4	14
PA 5	16
PA 6	18
PA 7	20

Signal	Pin
PB 0	22
PB 1	24
PB 2	26
PB 3	28
PB 4	30
PB 5	32
PB 6	34
PB 7	36
CB 1	38
CB 2	40

1.39 connected to + 5 Volt, 3-37 connected to GND

Ordering information

Product	Description	Order No.
VSIO	Standard I/O Module for VMEbus, 2x DSUB conn.	504-0
RS232-PB	RS232C Piggy-back	504-01
RS422-PB	RS422 Piggy-back	504-02
FIBER-PB	Fiber Optic Piggy-back	504-03
CLOOP-PB	20mA Current Loop Piggy-back	504-04
ACCESSORY	Centronics adapter for parallel port (flat cable without front panel)	800-0
ACCESSORY	" " " (includes separate front panel)	800-1

Each serial port requires one of the piggy-backs in addition to the VSIO module. If only parallel ports are used, no piggy-back is required.

A Centronics adapter allows the connection of a printer to the parallel port of the VSIO module.

SCC 8530 Programming Example:

```
PAGE     1    LIST VER  122084 4  10/27/87  14:25:37  X:2000.DEMO.SCCDEMO.TX

*****************************************************************************
*
*        Program example for SCC 8530:
*
*        - Initialization of SCC 8530
*        - Interrupt handling
*
*****************************************************************************


* HARDWARE PENDING EQUATES SCC 8530 (MPSC)

SCCBASE  EQU       $FE0020   SCC base address
MPSCDAT  EQU       0         Channel data register
MPSCCOM  EQU       1         Channel command register

WR1      EQU       1         Write register 1
WR2      EQU       2           "       "    2
WR3      EQU       3           "       "    3
WR4      EQU       4           "       "    4
WR5      EQU       5           "       "    5
WR6      FQU       6           "       "    6
WR7      EQU       7           "       "    7
WR8      EQU       8           "       "    8
WR9      FQU       9           "       "    9
WR10     EQU       10          "       "    10
WR11     FQU       11          "       "    11
WR12     FQU       12          "       "    12
WR13     EQU       13          "       "    13
WR14     EQU       14          "       "    14
WR15     FQU       15          "       "    15

RR0      EQU       0         Read register 0
RR1      EQU       1          "       "    1
RR2      FQU       2          "       "    2
RR3      EQU       3          "       "    3

INITWR3  EQU       $C0       Init.-value of WR3
INITWR4  FQU       $4C       Init.-value of WR4
INITWR5  FQU       $62       Init.-value of WR5
SCCVECT  EQU       $40       SCC interrupt vector

RESCHA   EQU       $80       Reset channel A
RESCHB   EQU       $40       Reset channel B
RESEXT   EQU       $10       Reset external/status interrupts
RESIUS   EQU       $38       Reset highest IUS
RESTIRQ  EQU       $28       Reset T-IRQ pending
RESERR   FQU       $30       Reset error

TENBL    EQU       3         WR5-Bit position: enable transmitter
RENBL    EQU       0         WR3-Bit position: enable receiver
RDRDY    EQU       0         RR0-Bit position: receive buffer full
TDRDY    EQU       2         RR0-Bit position: transmit buffer empty
```

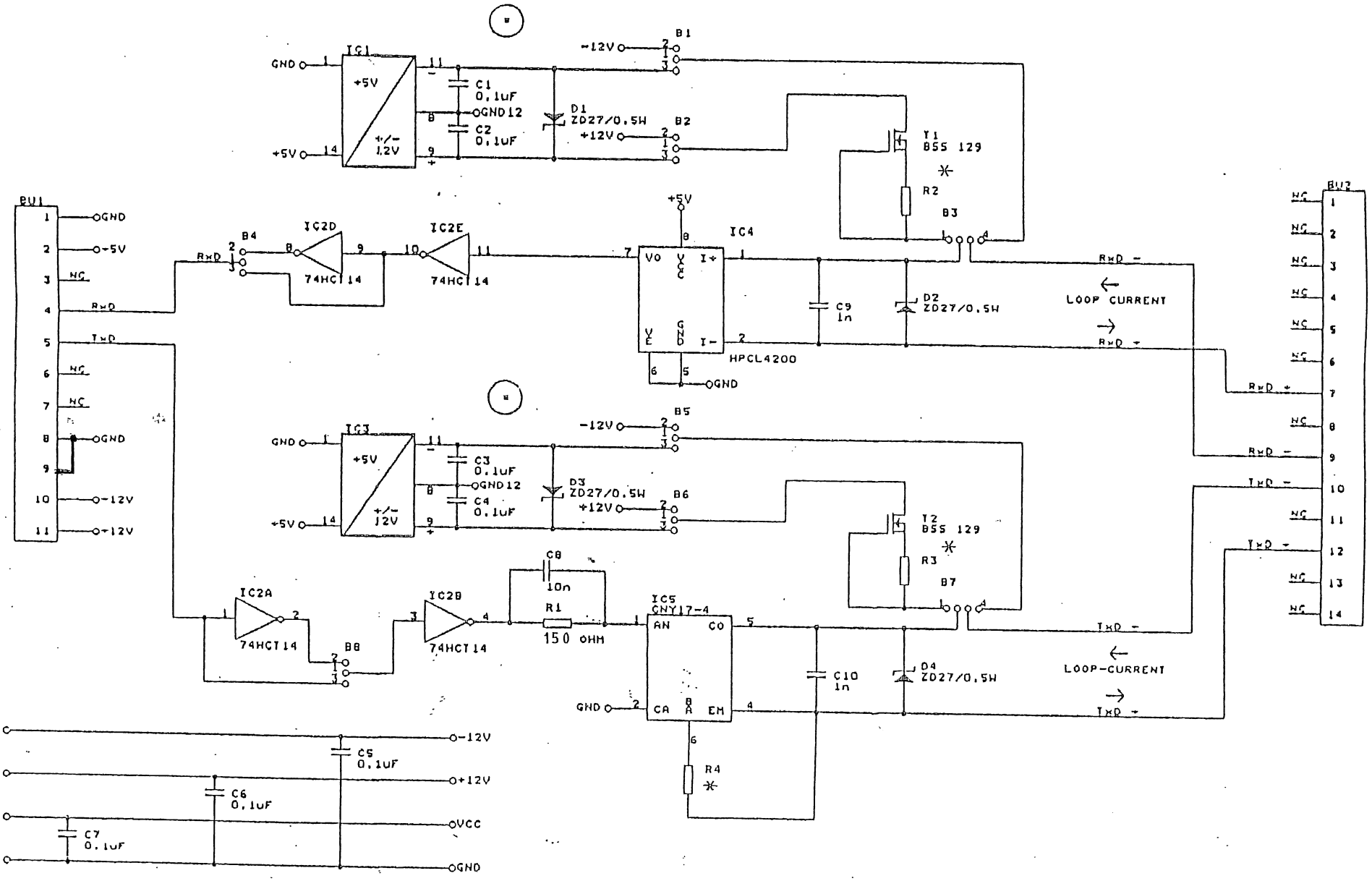
```
* MACRO

WTIME   MACRO
        NOP
        NOP
        NOP
        ENDM



* Variable definitions

SCCRES  DS.B    1       Reset code of channel A or B
SCCRRØ  DS.B    1       Storage for SCC status register RRØ
SCCRR1  DS.B    1       Storage for SCC status register RR1
SCCRR3  DS.B    1       Storage for SCC status register RR3
SCCCHAR DS.B    1       Storage for transmit or receive character


*****************************************************************************
*
*       Initialization of SCC 8530 channel A and B
*
*       Asynchronous 9600 Baud
*       Interrupt on receive buffer full
*       Interrupt on transmit buffer empty
*       2 stop bits
*       No parity
*
*****************************************************************************

        MOVE.L  #SCCBASE,AØ         Get SCC base address of channel A
        MOVE.B  #RESCHA,SCCRES      Reset code for channel A
        BSR     SCCINIT             Initialization of channel A


        MOVE.L  #SCCBASE+2,AØ       Get SCC base address of channel B
        MOVE.B  #RESCHB,SCCRES      Reset code for channel B
        BSR     SCCINIT             Initialization of channel B


SCCINIT:
        MOVE.B  #WR9,MPSCCOM(AØ)    Select WR9
        WTIME                       Wait time
        MOVE.B  SCCRES,MPSCCOM(AØ)  Channel reset, IRQ vector
        WTIME

        MOVE.B  #WR15,MPSCCOM(AØ)   Select WR15
        WTIME
        MOVE.B  #Ø,MPSCCOM(AØ)      External/status interrupts disabled
        WTIME

        MOVE.B  #RESEXT,MPSCCOM(AØ) Reset external/status interrupts
        WTIME
```

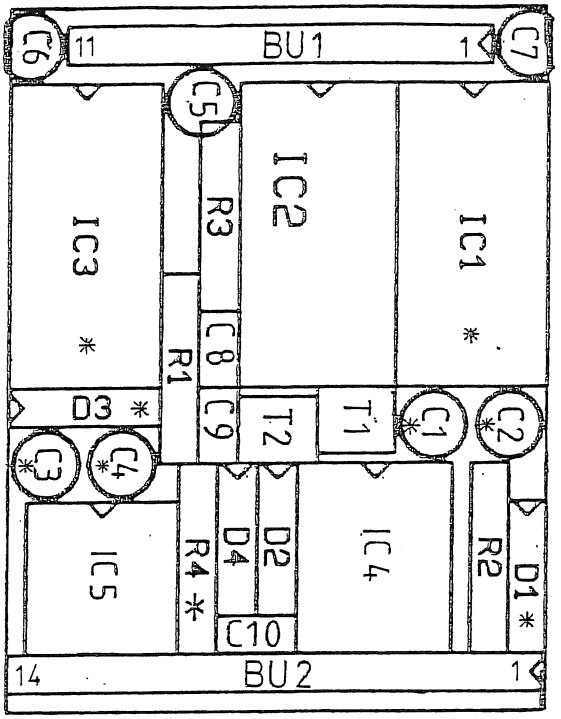
```
        MOVE.B  #RESERR,MPSCCOM(A0) Reset error
        WTIME

        MOVE.B  #WR2,MPSCCOM(A0)    Select WR2
        WTIME
        MOVE.B  #SCCVECT,MPSCCOM(A0)  IRQ vector number
        WTIME

        MOVE.B  #WR4,MPSCCOM(A0)    Select WR4
        WTIME
        MOVE.B  #INITWR4,MPSCCOM(A0)  X16 clock mode, 2 stop bits, no
        WTIME                         parity

        MOVE.B  #WR3,MPSCCOM(A0)    Select WR3
        WTIME
        MOVE.B  #INITWR3,MPSCCOM(A0)  R-bits/char, receiver disabled
        WTIME

        MOVE.B  #WR5,MPSCCOM(A0)    Select WR5
        WTIME
        MOVE.B  #INITWR5,MPSCCOM(A0)  T-bits/char, TXD disabled, RTS low
        WTIME

        MOVE.B  #WR1,MPSCCOM(A0)    Select WR1
        WTIME
        MOVE.B  #0,MPSCCOM(A0)      no WAIT/DMA REQ, no R-IRQ, no T-IRQ
        WTIME

        MOVE.B  #WR10,MPSCCOM(A0)   Select WR10
        WTIME
        MOVE.B  #0,MPSCCOM(A0)      NRZ mode
        WTIME

        MOVE.B  #WR11,MPSCCOM(A0)   Select WR11
        WTIME
        MOVE.B  #$50,MPSCCOM(A0)    RXC/TXC = BRG output
        WTIME

        MOVE.B  #WR12,MPSCCOM(A0)   Select WR12
        WTIME
        MOVE.B  #$E,MPSCCOM(A0)     Lower time constant of baud rate 9600
        WTIME

        MOVE.B  #WR13,MPSCCOM(A0)   Select WR13
        WTIME
        MOVE.B  #0,MPSCCOM(A0)      Upper time constant of baud rate 9600
        WTIME

        MOVE.B  #WR14,MPSCCOM(A0)   Select WR14
        WTIME
        MOVE.B  #$02,MPSCCOM(A0)    DTR = status bit, BRG-src = PCLK
        WTIME

* Enable functions

        MOVE.B  #WR14,MPSCCOM(A0)   Select WR14
        WTIME
```

```
        MOVE.B    #$03,MPSCCOM(A0)     Enable BRG
        WTIME

        MOVE.B    #INITWR3,D0
        BSET.L    #RENBL,D0
        MOVE.B    #WR3,MPSCCOM(A0)     Select WR3
        WTIME
        MOVE.B    D0,MPSCCOM(A0)       Enable receiver
        WTIME

        MOVE.B    #INITWR5(A5),D0
        BSET.L    #TENBL,D0
        MOVE.B    #WR5,MPSCCOM(A0)     Select WR5
        WTIME
        MOVE.B    D0,MPSCCOM(A0)       Enable transmitter
        WTIME

        MOVE.B    #WR1,MPSCCOM(A0)     Select WR1
        WTIME
        MOVE.B    #$12,MPSCCOM(A0)     Enable R-IRQ or all Rxcharacters,
        WTIME                          enable T-IRQ

        MOVE.B    #WR9,MPSCCOM(A0)     Select WR9
        WTIME
        MOVE.B    #$8,MPSCCOM(A0)      Enable master interrupt
        WTIME

        RTS


*************************************************************************
*       Load vector address with address of SCC interrupt
*       service routine
*************************************************************************

        MOVE.L    #SCCVECT*4,A1        Get vector address of SCC vector = $40
        LEA       SCCISR(PC),A2        Load vector address of SCC vector with
        EXG       D0,A2                address of SCC interupt service
        MOVE.L    D0,(A1)              routine


*************************************************************************
*       Interrupt service routine
*************************************************************************

SCCISR:
        MOVE.L    #SCCBASE,A0          Get SCC base address

        MOVE.B    #RR3,MPSCCOM(A0)     Select RR3
        WTIME
        MOVE.B    MPSCCOM(A0),SCCRR3   Save RR3 status in SCCRR3

        BTST.B    #5,SCCRR3            R-IRQ channel A pending ?
        BNE       RECEIVE              Branch if yes
```

```
        BTST.B    #4,SCCRR3           T-IRQ channel A pending ?
        BNE.S     TRANS               Branch if yes

        MOVE.L    #SCCBASE+2,A0       Get SCC channel B base address

        BTST.B    #2,SCCRR3           R-IRQ channel B pending ?
        BNE       RECEIVE             Branch if yes

        BTST.B    #1,SCCRR3           T-IRQ channel B pending ?
        BNE.S     TRANS               Branch if yes

        RTE

*******************************************************************************
*       Output of a character
*******************************************************************************

TRANS:
        MOVE.B    #RR0,MPSCCOM(A0)    Get SCC status register RR0
        WTIME
        MOVE.B    MPSCCOM(A0),SCCRR0
        BTST.B    #TDRDY,SCCRR0       Test for busy
        BEQ.S     TRANSERR            Branch if busy

        MOVE.B    SCCCHAR,MPSCDAT(A0) Send data byte to device
        MOVE.B    #RESIUS,MPSCCOM(A0) Reset highest IUS pending

        RTE                           Finish up IRQ service routine

*******************************************************************************
*       Input of a character
*******************************************************************************

RECEIVE:
        MOVE.B    #RR0,MPSCCOM(A0)    Get SCC status register RR0
        WTIME
        MOVE.B    MPSCCOM(A0),SCCRR0
        BTST.B    #RDRDY,SCCRR0       Test for receive buffer full
        BEQ.S     RECERR              Branch if no receive buffer full

        MOVE.B    #RR1,MPSCCOM(A0)    Get SCC status register RR1
        WTIME
        MOVE.B    MPSCCOM(A0),SCCRR1
        AND.B     #$70,SCCRR1         Error in receive sequence ?
        BNE.S     RECERR              Branch if error

        MOVE.B    MPSCDAT(A0),SCCCHAR  Get character from SCC
        MOVE.B    #RESIUS,MPSCCOM(A0) Reset highest IUS

        RTE                           Finish up IRQ service routine

*******************************************************************************
*       Error handling
*******************************************************************************
```

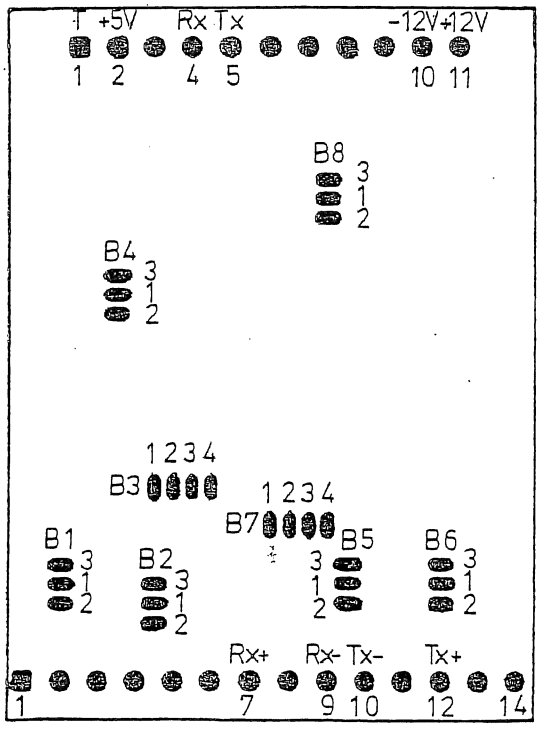
```
RECERR:
        MOVE.B    MPSCDAT(A0),D1      dummy read to clear pending IRQ

        MOVE.B    #RESERR,MPSCCOM(A0) Reset SCC error status bits
        WTIME
        MOVE.B    #RESIUS,MPSCCOM(A0) Reset highest IUS

***     Error message to mark error in receive sequence     ***
        :
        :
        :


TRANSERR:

        MOVE.B    #RESTIRQ,MPSCCOM(A0)  Reset T-IRQ pending
        WTIME
        MOVE.B    #RESERR,MPSCCOM(A0) Reset SCC error status bits
        WTIME
        MOVE.B    #RESIUS,MPSCCOM(A0) Reset highest IUS

***     Error message to mark error in transmit sequence     ***
        :
        :
        :
```


Modular Computers®
CLOOP-PB
31.062-1070.1
Date 10.10.2000
Index 01/2
Page 1 of 1
IC1
+5V
+/- 12V
GND
+5V
C1 0.1uF
C2 0.1uF
GND12
D1 ZD27/0.5W
-12V
+12V
B1
B2
T1 BSS 129
R2
B3
BU1
BU2
B4
IC2D
IC2E
74HCT14
RxD
TxD
IC4
HPCL4200
VO
VCC
VEE
I+
I-
C9 1n
D2 ZD27/0.5W
LOOP CURRENT
IC3
C3 0.1uF
C4 0.1uF
D3 ZD27/0.5W
B5
B6
T2 BSS 129
R3
B7
IC2A
IC2B
B8
C8 10n
R1 150 OHM
IC5 CNY17-4
AN
CA
CO
EM
C10 1n
D4 ZD27/0.5W
R4
C5 0.1uF
C6 0.1uF
C7 0.1uF
VCC
* entspr. Bestell-Nr. / according to order-no. 504-04/x



B-side

solder-jumpers on L-side



L-side

Modular Computers®	CLOOP-PB		31.062-1070.1
	Date	10.10.2000	Index 01 / 2
			Page 1 of 1