SYLVA
CONTROL SYSTEMS

AE-DAC - See Picture

The DAC analog expansion board offers four 8 bit analog outputs. Each Output channel has a common ground and can operate in voltage or current output modes.

Specifications:

Output

- four 8 bit analog outputs
- Range 0-20mA / 4-20mA / 0-10 Vdc.

Power

- 5V via I2C connection
- External 12-20 VDC required for output devices.

Communication

- Philips I2C bus

Physical

- 4.5" x 4.5" x 1"

Features:

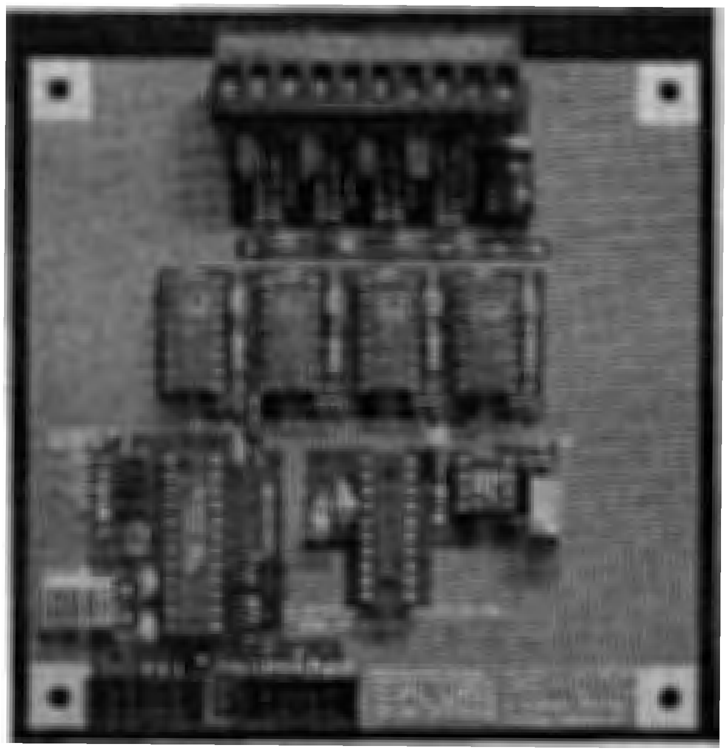
- Fully socketed circuit board
- Plug-on connector system
- Daily chain I2C bus connection
- AD694 output devices

[CATALOG] [ORDERING INFO] [PRICES&COSTS] [CONTACT US]

[TECH SUPPORT] [WHAT'S NEW]

LastEdited 08/10/97 by William H Mogk of ***WHM Software***

AE-DAC



Return to AE-DAC

SYLVA
CONTROL SYSTEMS

IO-8O8I - See Picture

The IO-8O8I provides eight discrete AC/DC opto-isolated inputs and eight FORM A relay outputs for Sylva's 552 controllers.

Specifications

Input/Output

- 8 AC/DC 12-24 Vdc opto-isolated inputs
- 8 FORM A 5 amp relay contacts.

Power

- 5V via I2C connection
- External 12 VDC for relay coil supply

Communication

- Philips I2C bus

Physical

- 4.5" x 4.5" x 1"

Features

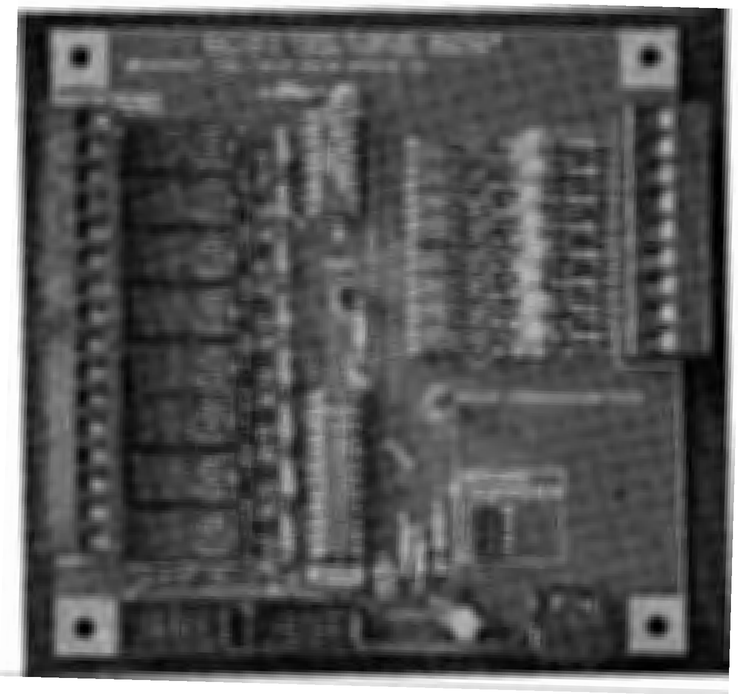
- Fully socketed circuit board
- Plug-on connector system
- Daily chain I2C bus connection
- LED indicators for all relay status

[CATALOG] [ORDERING INFO] [PRICES&COSTS] [CONTACT US]

[TECH SUPPORT] [WHAT'S NEW]

LastEdited 08/10/97 by William H Mogk of ***WHM Software***

IO-8O8I



Return to IO-8O8I

SYLVA
CONTROL SYSTEMS

IO-16I - See Picture

The IO-16I provides sixteen discrete AC/DC opto-isolated expansion inputs for Sylva's 552 controllers.

Specifications

Input

- 16 AC/DC 12-24 Vdc Opto-isolated inputs
- Separate grounds for each group of 8 inputs

Power

- 5V via I2C connection

Communication

- Philips I2C bus

Physical

- 4.5" x 4.5" x 1"

Features

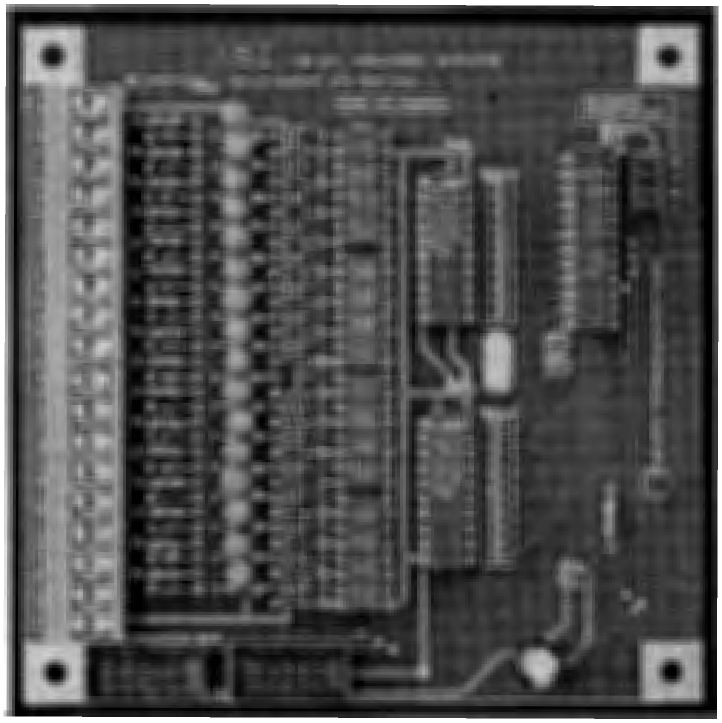
- Fully socketed circuit board
- Plug-on connector system
- Daily chain I2C bus connection
- LED indicators for all input status

[CATALOG] [ORDERING INFO] [PRICES&COSTS] [CONTACT US]

[TECH SUPPORT] [WHAT'S NEW]

LastEdited 08/10/97 by William H Mogk of ***WHM Software***

IO-16I



Back to IO-16I

SYLVA
CONTROL SYSTEMS

IO-16O - See Picture

The IO-16O provides sixteen FORM A relay expansion outputs for Sylva's 552 controllers.

Specifications

Output

- 16 FORM A relay expansion outputs for Sylva's 552 controllers.

Power

- 5V via I2C connection
- External 12 VDC for relay coil supply

Communication

- Philips I2C bus

Physical

- 4.5" x 4.5" x 1"

Features

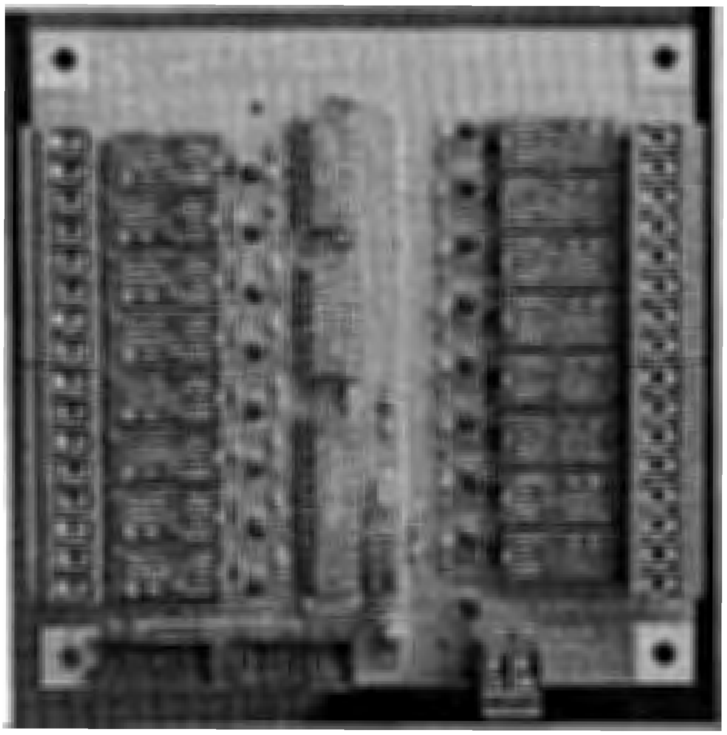
- Fully socketed circuit board
- Plug-on connector system
- Daily chain I2C bus connection
- LED indicators for all relay status
- Optional relay sockets

[CATALOG] [ORDERING INFO] [PRICES&COSTS] [CONTACT US]

[TECH SUPPORT] [WHAT'S NEW]

LastEdited 08/10/97 by William H Mogk of ***WHM Software***

IO-160



Back to IO-160

IO-16O OUTPUT EXPANSION BOARD

The IO-16O offers 16 form A relay outputs rated at 5A 120 VAC resistive. Each relay has a separate connection, there are no commons. The IO-16O has a 4 bit binary address selection header allowing addresses to be selected from 0 to 15. Each IO-16O is written to when the SC552 controller executes the EXIO [address],1 statement. The address value must always be a two digit integer (00-15) and the zero after the comma represents a IO-16O board. A EXIO Address LED will toggle on the IO-16O when the controller writes data. This LED is complemented at each valid write, therefore it takes two reads to flash the LED ON then OFF. Each board is supplied with 1-10 conductor ribbon cable with keyed ends and each board has 2 headers to allow chaining of additional boards. The IO-16O requires a 5 volt supply which is provided on the ribbon cable from the controller and a 12 VDC connection.

The SC552 controller will only write data to the IO-16O when the EXIO is executed, therefore the user must ensure enough EXIO executions to satisfy their requirements. The EXIO statement allows for no NOACKs (not acknowledges) from any external IO board before an EXIO communication error is generated. The user can then use the SC552s error trapping to take appropriate action.

The bits in the SC552s memory to where an input board is mapped can be calculated by the following formula:

SC552 bit = (Board Address*16)+32+Board Output

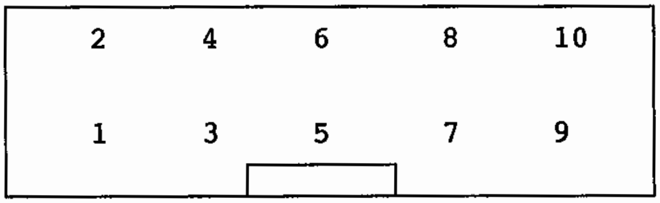
Example: Board Address = 01 and output 03 on IO-16O

SC552 bit = (1*16)+32+3 = 51

The statement OTL 051 or OTU 051 can be used to change the status of the output 03 on the IO-16O addressed 01.

A program TEST16O.BAS is provided on the SC552 demo disk.

EXPANSION IO BOARD CONNECTOR PIN-OUT



TOP VIEW

1,2,4 - +5Vdc supply
3,5,7,9 - Ground
6 - RESET (active high)
8 - SDA (IIC serial data)
10 - SCL (IIC serial clock)

Copyright © 1996 Sylva Energy Systems Inc.

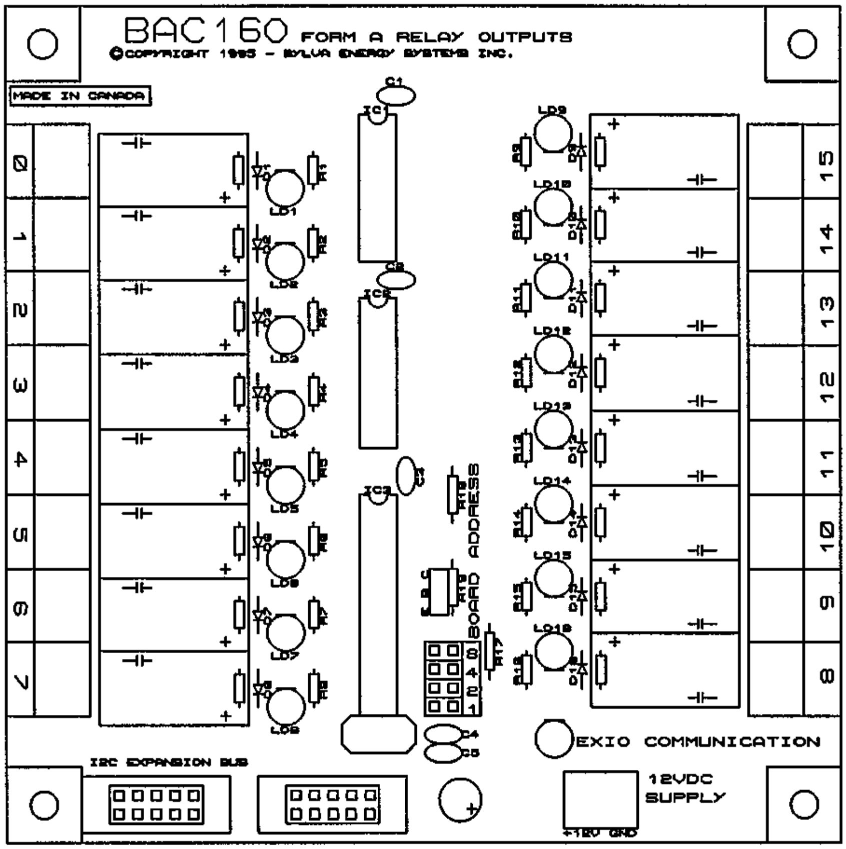
```
        REM * TEST160.BAS *
2       REM * This is a demo program for the IO-160 expansion unit *
3       REM * THIS PROGRAM IS FOR DEMONSTRATION ONLY !!!!!!!!!!!!!!!!!!!!!!! *
4       REM
5       REM * The IO-160 provides 16 additional relay outputs
6       REM * The BOARD ADDRESS jumpers select the base for where the outputs
7       REM * are mapped when the EXIO xx,y is executed.
8       REM * Where xx must equal the address set on the IO-160 and y must
9       REM * equal 0 for a input board. Example EXIO 01,1
10      REM
11      REM
12      REM * The bit address for the IO-160 outputs is calculated as follows:
13      REM * bit address = IO-160 output# + (Board Address * 16)+32
14      REM
15      REM * For example Board Address=01 (EXIO 01,0)
16      REM * for output 0       0+(01*16)+32=48
17      REM * for output 1       1+(01*16)+32=49
18      REM * for output 15      15+(01*16)+32=63
19      REM
20      REM * When a SCDTMF is used in a system it has a fixed address of 00
21      REM * therefore any other expansion boards CANNOT USE ADDRESS 00!
22      REM
50      CLEAR O: CLEAR R: CLEAR B: CLEAR J: CLEAR D: CLEAR I
200     REM LOOP
205     XIH 511: OST 000: CTU 000,016
210     XIL 511: ROS 000
 11     XCT 000,001: OTL 048: OTU 063
212     XCT 000,002: OTL 049: OTU 048
213     XCT 000,003: OTL 050: OTU 049
214     XCT 000,004: OTL 051: OTU 050
215     XCT 000,005: OTL 052: OTU 051
216     XCT 000,006: OTL 053: OTU 052
217     XCT 000,007: OTL 054: OTU 053
218     XCT 000,008: OTL 055: OTU 054
219     XCT 000,009: OTL 056: OTU 055
220     XCT 000,010: OTL 057: OTU 056
221     XCT 000,011: OTL 058: OTU 057
222     XCT 000,012: OTL 059: OTU 058
223     XCT 000,013: OTL 060: OTU 059
224     XCT 000,014: OTL 061: OTU 060
225     XCT 000,015: OTL 062: OTU 061
226     XCT 000,016: OTL 063: OTU 062: CTR 000
300     FLFP 012: LIO: EXIO 01,1: GOTO 200
```

64 + 32   96

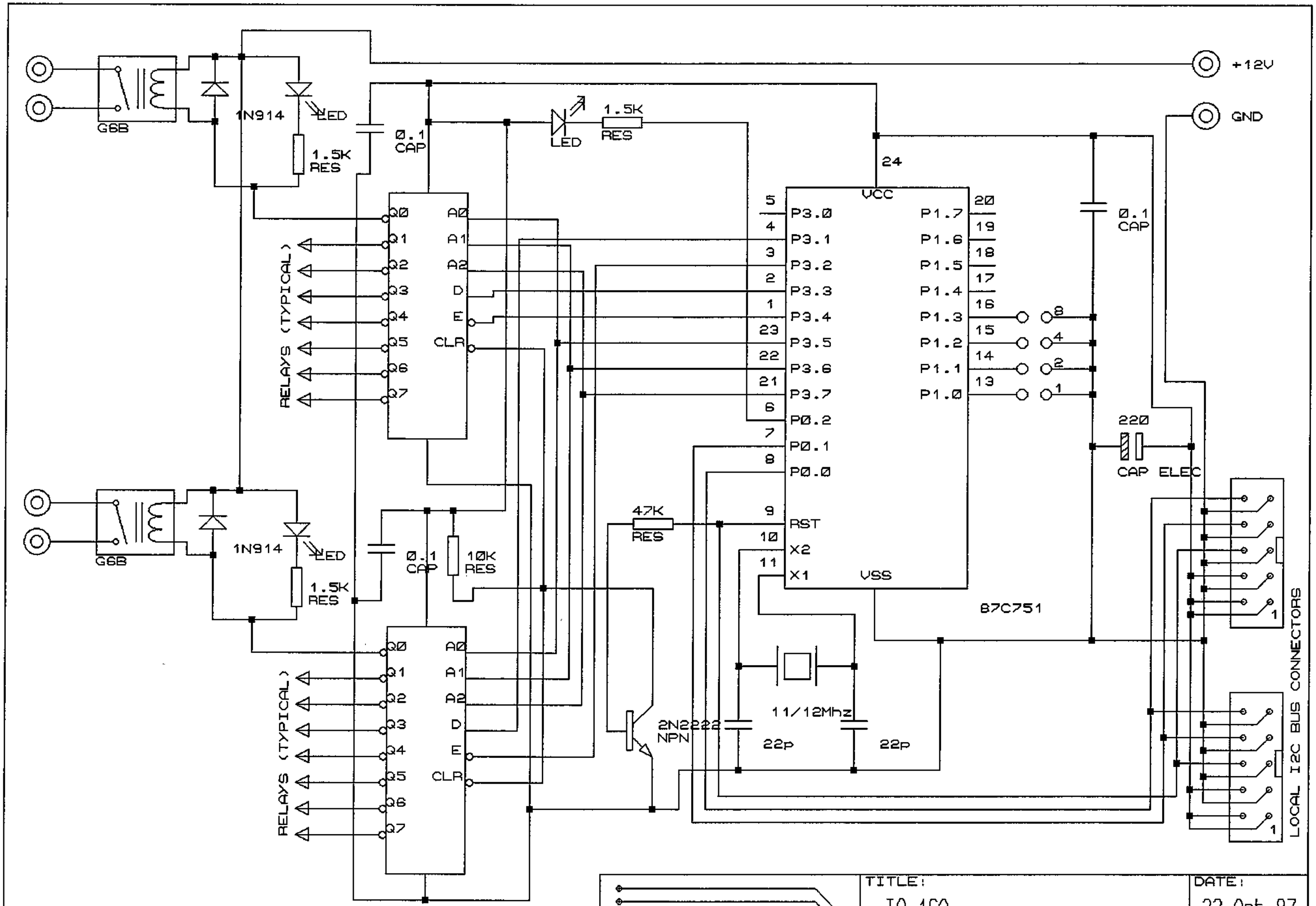
← BOARD ADDRESS 04

Copyright © 1996 Sylva Energy Systems Inc.

IO-160 RELAY EXPANSION BOARD



1,2,4-Vcc (5V)
3,5,7,9-Ground
6-Reset
8-SDA
10-SCL



SYLVA
CONTROL SYSTEMS

TITLE:

IO-160

RELAY EXPANSION

BY: LG

REV:1.Ø

DATE:

22-Oct-97

PAGE:

1/1

TEST 160.BAS

```
10      REM * SIMPLE PROGRAM FOR TESTING A BAC160 OUTPUT BOARD *
20      REM * THE BOARD ADDRESS IS = 01 *
30      REM
50      CLEAR O: CLEAR R: CLEAR B: CLEAR J: CLEAR D: CLEAR I
200     REM LOOP
205     XIH 511: OST 000: CTU 000,016
210     XIL 511: ROS 000
211     XCT 000,001: OTL 048: OTU 063
212     XCT 000,002: OTL 049: OTU 048
213     XCT 000,003: OTL 050: OTU 049
214     XCT 000,004: OTL 051: OTU 050
215     XCT 000,005: OTL 052: OTU 051
216     XCT 000,006: OTL 053: OTU 052
217     XCT 000,007: OTL 054: OTU 053
218     XCT 000,008: OTL 055: OTU 054
219     XCT 000,009: OTL 056: OTU 055
220     XCT 000,010: OTL 057: OTU 056
221     XCT 000,011: OTL 058: OTU 057
222     XCT 000,012: OTL 059: OTU 058
223     XCT 000,013: OTL 060: OTU 059
224     XCT 000,014: OTL 061: OTU 060
225     XCT 000,015: OTL 062: OTU 061
226     XCT 000,016: OTL 063: OTU 062: CTR 000
300     FLFP 012: LIO: EXIO 01,1: GOTO 200
```

SYLVA
CONTROL SYSTEMS

IO-Rack - see Picture

The IO-Rack interfaces Sylva's expansion I/O system to industry standard I/O module mounting rack systems manufactured by OPYO22, Grayhill and others. The IO-Rack also has extended 12C bus drivers for remote operation as well as the local 12C bus connection.

Specifications

Input / Output

- 24 Input/Output bits bia 50 pin Header

Power

- 5V via I2C sonnection for local Operation
- 12 VDC input for remote operation

Communication

- Philips I2C Bus local or extended

Physical

4.5" x 4.5" x 1"

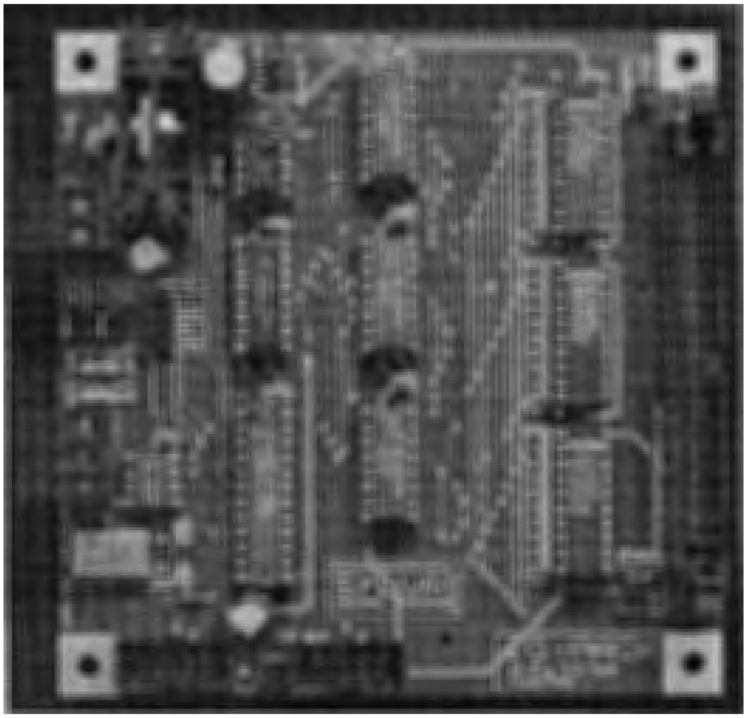
Features

- Fully socketed circuit board
- 50 pin locking header for rack connection
- Daisy chain I2C bus connection
- Hardware watchdog timer
- Extended I2C connections

[CATALOG] [ORDERING INFO] [PRICES&COSTS] [CONTACT US]

[TECH SUPPORT] [WHAT'S NEW]

IO-Rack



Return to IO-Rack

IO-RACK 24 BIT EXPANSION BOARD

The IO-RACK offers 24 bits of logic level inputs or outputs with high current sink ability. The IO-RACK board uses NE5090s open-collector relay drivers for sinking outputs and 74HC540s for logic inputs. The open-collector outputs are pulled-up to Vcc with 10K resistors and tied to the 540 inputs. This allows any mix of inputs or outputs and the true output state can be seen by reading all inputs. One disadvantage is that this arrangement uses 6 IO words. The IO-RACK board also has a I2C bus driver chip and a 12Vdc input so it can use an extended I2C bus and be remotely located away from the controller.
The IO-RACK was designed with a 50 pin 0.1" box header to connect to industry standard Io boards by OPTO 22, Grayhill, P&B and others.

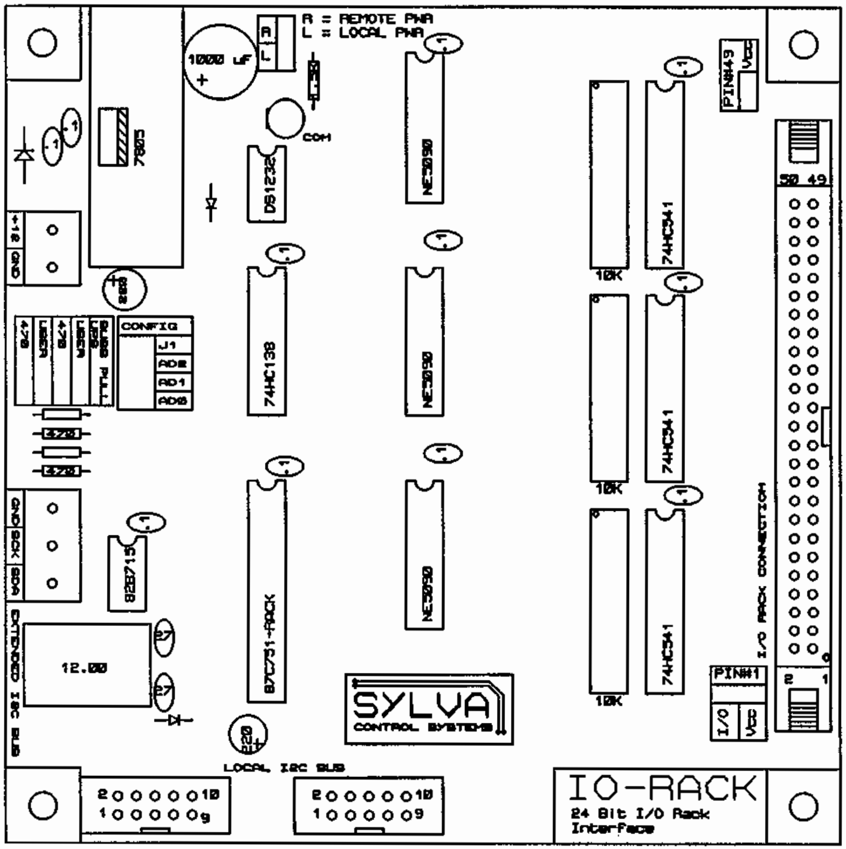
Each board is supplied with 1-10 conductor ribbon cable with keyed ends and each board has 2 headers to allow chaining of additional boards. The IO-RACK requires a 5 volt supply which is provided on the ribbon cable from the controller or 12 Vdc when located remotely.
The SC552 controller will only read and write data to the IO-RACK when the EXIO is executed, therefore the user must ensure enough EXIO executions to satisfy their requirements. It takes 3 consecutive EXIOs to read and write all 24 IO bits on the board. The EXIO statement allows for no NOACKs (not acknowledges) from any external IO board before an EXIO communication error is generated. The user can then use the SC552s error trapping to take appropriate action.

The IO-RACK addresses starts at a base of 60Hex for writes and 61Hex for reads. The first I/O word uses 60H write and 61H read. The next word uses 62H and 63H, with the last word using 64H and 65H. By shorting the AD0 jumper with AD1 and AD2 open the starting address is 66H. The address jumpers AD0, AD1 and AD2 move the address by 6 hex. The following table shows the 552 controllers memory address, EXIO address and IO-RACK board address.

I2C Address	IO-RACK Address	552 Controller Memory	EXIO Statement
60h	00	274h write 275h read 276h write 277h read 278h write 279h read	EXIO 00,2 EXIO 01,2 EXIO 02,2
66h	01	27Ah write 27Bh read 27Ch write 27Dh read 27Eh write 27Fh read	EXIO 03,2 EXIO 04,2 EXIO 05,2
6Ch	02	280h write 281h read 282h write 283h read 284h write 285h read	EXIO 06,2 EXIO 07,2 EXIO 08,2
72h	03	286h write 287h read 288h write 289h read 28Ah write 28Bh read	EXIO 09,2 EXIO 10,2 EXIO 11,2
78h	04	28Ch write 28Dh read 28Eh write 28Fh read 290h write 291h read	EXIO 12,2 EXIO 13,2 EXIO 14,2

Copyright © 1996 Sylva Energy Systems Inc.

IO-RACK EXPANSION BOARD

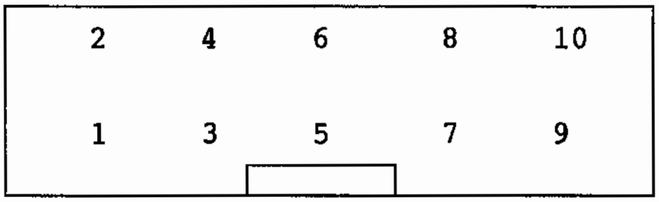


49-Vcc or NC
47-Word 0: Bit 0
45-0:1
43-0:2
41-0:3
39-0:4
37-0:5
35-0:6
33-0:7
31-Word 1: Bit 0
29-1:1
27-1:2
25-1:3
23-1:4
21-1:5
19-1:6
17-1:7
15-Word 2: Bit 0
13-2:1
11-2:2
9-2:3
7-2:4
5-2:5
3-2:6
1-Vcc or Word 2: Bit 7

Pins 2 to 50 are Grour

1,2,4-Vcc (5V)
3,5,7,9-Ground
6-Reset
8-I2C DATA
10-I2C CLOCK

EXPANSION IO BOARD CONNECTOR PIN-OUT

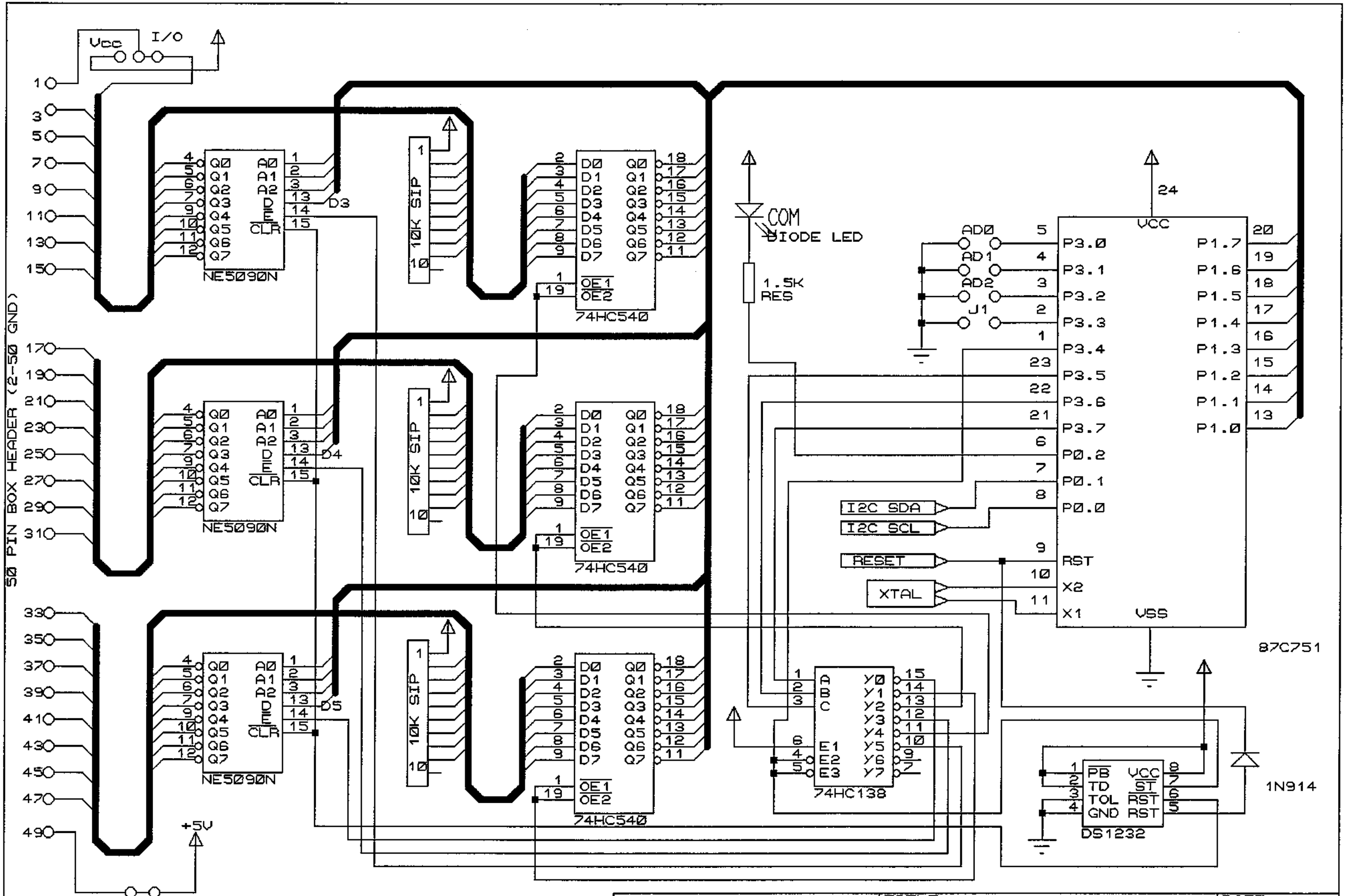


TOP VIEW

1,2,4 - +5Vdc supply
3,5,7,9 - Ground
6 - RESET (active high)
8 - SDA (IIC serial data)
10 - SCL (IIC serial clock)

```
3       REM IO-RACK DEMO PROGRAM
4       REM
5       CLEAR R
6      A=-1
10      PRINT "Output bytes",XBY(274H),XBY(276H),XBY(278H),"  Input bytes",
11      PRINT XBY(275H),XBY(277H),XBY(279H)
20      FOR T=1 TO 250: NEXT T
25     A=A+1: IF A>255 THEN A=0
26     XBY(274H)=A:XBY(276H)=A:XBY(278H)=A
30      EXIO 00,2: EXIO 01,2: EXIO 02,2
 )      GOTO 10
```

Copyright © 1996 Sylva Energy Systems Inc.



SYLVA

CONTROL SYSTEMS

TITLE:

IO-RACK
EXPANSION BAORD

BY:

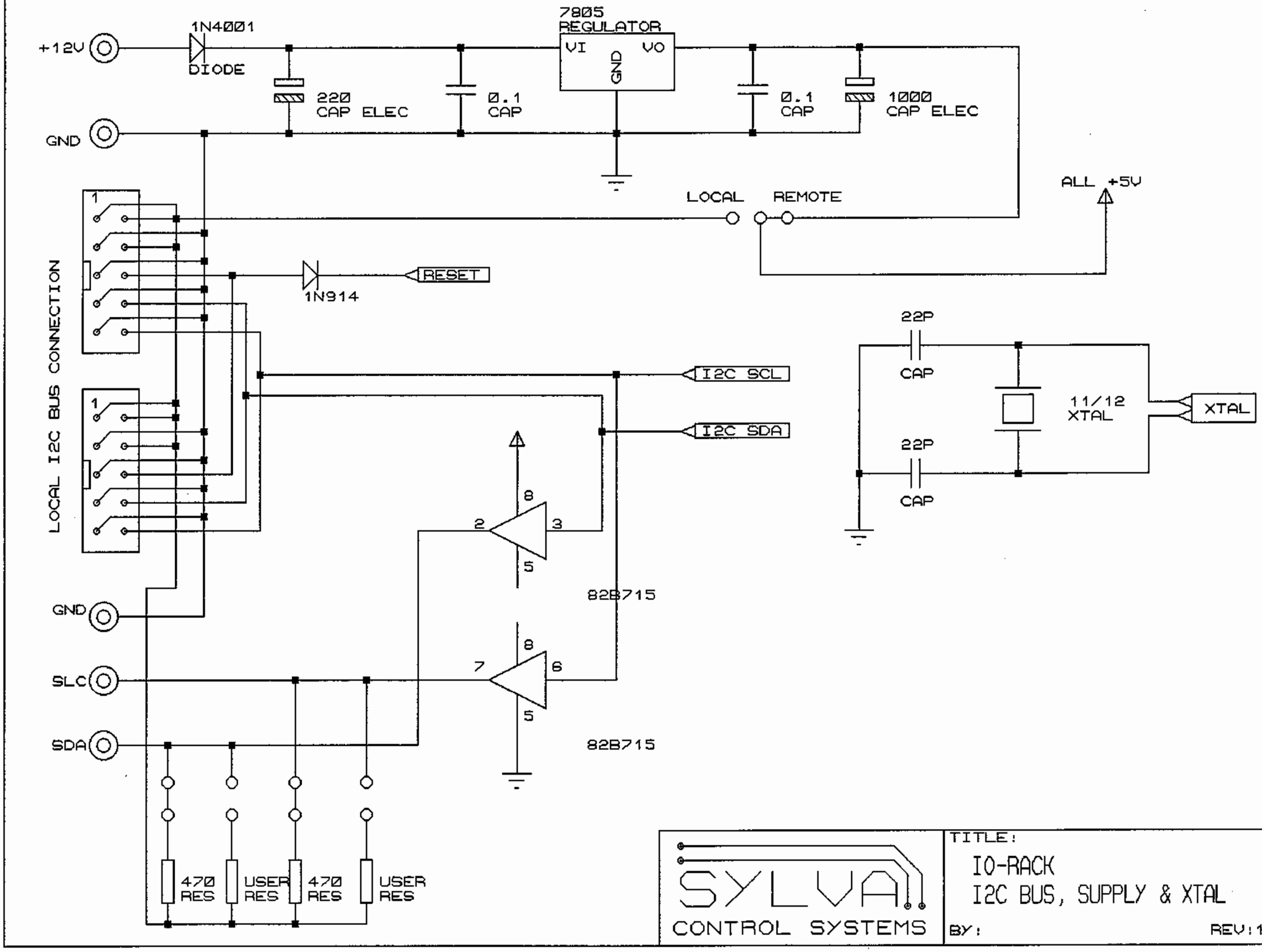
REV:1.0

DATE:

10-Mar-97

PAGE:

1/2



SYLVA
CONTROL SYSTEMS

TITLE:
IO-RACK
I2C BUS, SUPPLY & XTAL

BY:

REV:1.0

DATE:
10-Mar-97

PAGE:
2/2

RACK TEST · BAS

```
3       REM IO-RACK DEMO PROGRAM
4       REM
5       CLEAR R
6       REM this sets startup value of A so line 25 begins at zero
7       A=-1
10      PRINT "Output bytes",XBY(274H),XBY(276H),XBY(278H),
11      PRINT "Input bytes",XBY(275H),XBY(277H),XBY(279H)
20      FOR T=1 TO 250: NEXT T
25      A=A+1: IF A>255 THEN A=0
30      REM this stuffs the image table locations with the value of A
35      XBY(274H)=A:XBY(276H)=A:XBY(278H)=A
40      REM this outputs the values of A in the image table to the three bytes of the
rack
45      EXIO 00,2: EXIO 01,2: EXIO 02,2
50      GOTO 10
```

SYLVA
CONTROL SYSTEMS

IO-TERM - See Picture

The TERM is a simple terminal controller with a 1 x 16 to 4 x 40 LCD display connection. The TERMinal can connect to any RS232 or RS485 device operating at 9600 baud. The TERM can provide a very cost effective solution for implemementing an operator input/output device for controllers without that provision..

Specifications

Input/Output

- 4 x 4 matrix keypad
- 1 x 16 to 4 x 40 character LCD display
- O.C. output for back light control

Power

- 8-18 VDC input

Communication

- simple RS232 or RS485 at 9600 baud

Physical

- 4.0" x 3.0" x 1"

Features

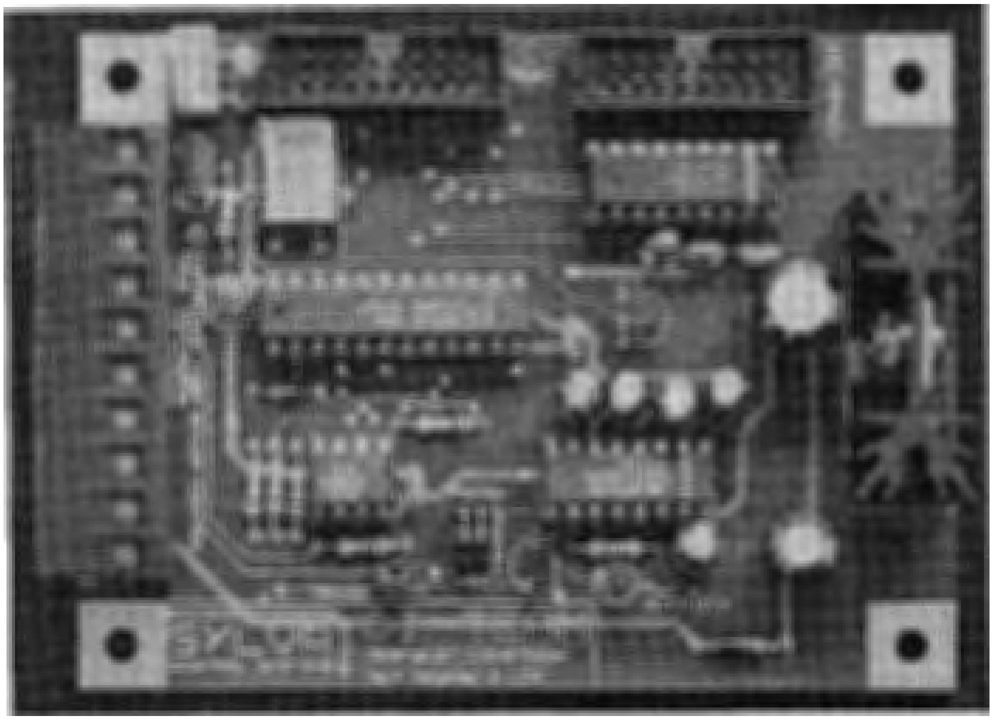
- Fully socketed circuit board
- Plug-on wiring connection
- Low cost

[CATALOG] [ORDERING INFO] [PRICES&COSTS] [CONTACT US]

[TECH SUPPORT] [WHAT'S NEW]

LastEdited 08/10/97 by William H Mogk of ***WHM Software***

IO-TERM



Back to IO-TERM

OI-TERM *User Manual/Data*

LCD Display / Keypad Controller

The OI-TERM provides a simple remote or local serial linked LCD display and keypad controller. The liquid crystal display controller responds to a simple command set which allows the clearing of the display and cursor positioning. Once the cursor is positioned the display operates in an auto increment mode advancing 1 position to the right as characters are printed. The OI-TERM will work with any LCD panel that uses a HITACHI 44780 controller. The 16 pin display connector allows direct connection to LCD units supplied by SYLVA Control Systems. The 4x40 LCD units use a straight ribbon cable and the 2x40 LCD units use a twist cable (two outside lines twisted). A pin-out is provided for direct wiring to other displays.

The keypad interface connects to a 4x4 matrix keypad via a 16 pin header connector. The controller sends the ASCII equivalent of the key pressed plus 128 (high bit set) and a CR. This is done to emulate the codes sent by older BAC-KEY keypad controllers. The 552ES controllers also have ASM routines (accessed by CALLs) to convert the ASCII keys to I/O image table bits. This allows older BACKEYs and BACLCDs to be easily replaced by the newer IO-TERM board.

OI-TERM Commands (shown as BASIC Statements)	
Cursor Position (upper display)	**PRINT #CHR(27),CHR(1),CHR(*);** *** = 0 to 79**
Cursor Position (lower display)	**PRINT #CHR(27),CHR(2),CHR(*);** *** = 0 to 79**
Clear display/Cursor home	**PRINT #CHR(27),CHR(3);**
Backlight ON	**PRINT #CHR(27),CHR(4);**
Backlight OFF	**PRINT #CHR(27),CHR(5);**
Clear upper display only	**PRINT #CHR(27),"U";**
Clear lower display only	**PRINT #CHR(27),"L";**
Reset controller	**PRINT #CHR(27),"R";**

Note: If the user connects a smaller display (ie 2x20) the first position on the bottom line always starts at 40.

Copyright © 1997 Sylva Control Systems

The OI-TERM transmits the ASCII equivalents of the key pressed plus 128. The reason for sending the ASCII value plus 128 is that the LCD controller ignores any received data which is greater than 127. Therefore no transmitted data will be displayed on the LCD if it's connected to a controller using the RS485 link. The keys are not repeated if held down. The ASCII transmission is initiated only on a key-down transition.

The SC552ES controllers have special ASM calls to convert the ASCII keycodes to the equivalent of the 2 bit code used by the old BACKEY controllers. The calling address depends on which of the 3 serial ports are connected to the OI-TERM board. The console port uses **361Ahex**, the AUX RS232 port uses **361Dhex** and the RS485 port uses **3620hex**.

The following is a table of 2 bit codes when the special ASM call is used in the COMINT routine:

```
Key 0 - bit 056 & 60  (XIH 056: XIH 060: PRINT "0")
Key 1 - bit 057 & 60  (XIH 057: XIH 060: PRINT "1")
Key 2 - bit 058 & 60  (XIH 058: XIH 060: PRINT "2")
Key 3 - bit 059 & 60  (XIH 059: XIH 060: PRINT "3")
Key 4 - bit 056 & 61  (XIH 056: XIH 061: PRINT "4")
Key 5 - bit 057 & 61  (XIH 057: XIH 061: PRINT "5")
Key 6 - bit 058 & 61  (XIH 058: XIH 061: PRINT "6")
Key 7 - bit 059 & 61  (XIH 059: XIH 061: PRINT "7")

Key 8 - bit 056 & 62  (XIH 056: XIH 062: PRINT "8")
Key 9 - bit 057 & 62  (XIH 057: XIH 062: PRINT "9")
Key A - bit 058 & 62  (XIH 058: XIH 062: PRINT "A")
Key B - bit 059 & 62  (XIH 059: XIH 062: PRINT "B")
Key C - bit 056 & 63  (XIH 056: XIH 063: PRINT "C")
Key D - bit 057 & 63  (XIH 057: XIH 063: PRINT "D")
Key E - bit 058 & 63  (XIH 058: XIH 063: PRINT "E")
Key F - bit 059 & 63  (XIH 059: XIH 063: PRINT "F")
```

Note: The special ASM call doesn't clear the bit state as in the old BACKEYs. The user should execute a CLW 07 after the bit testing.

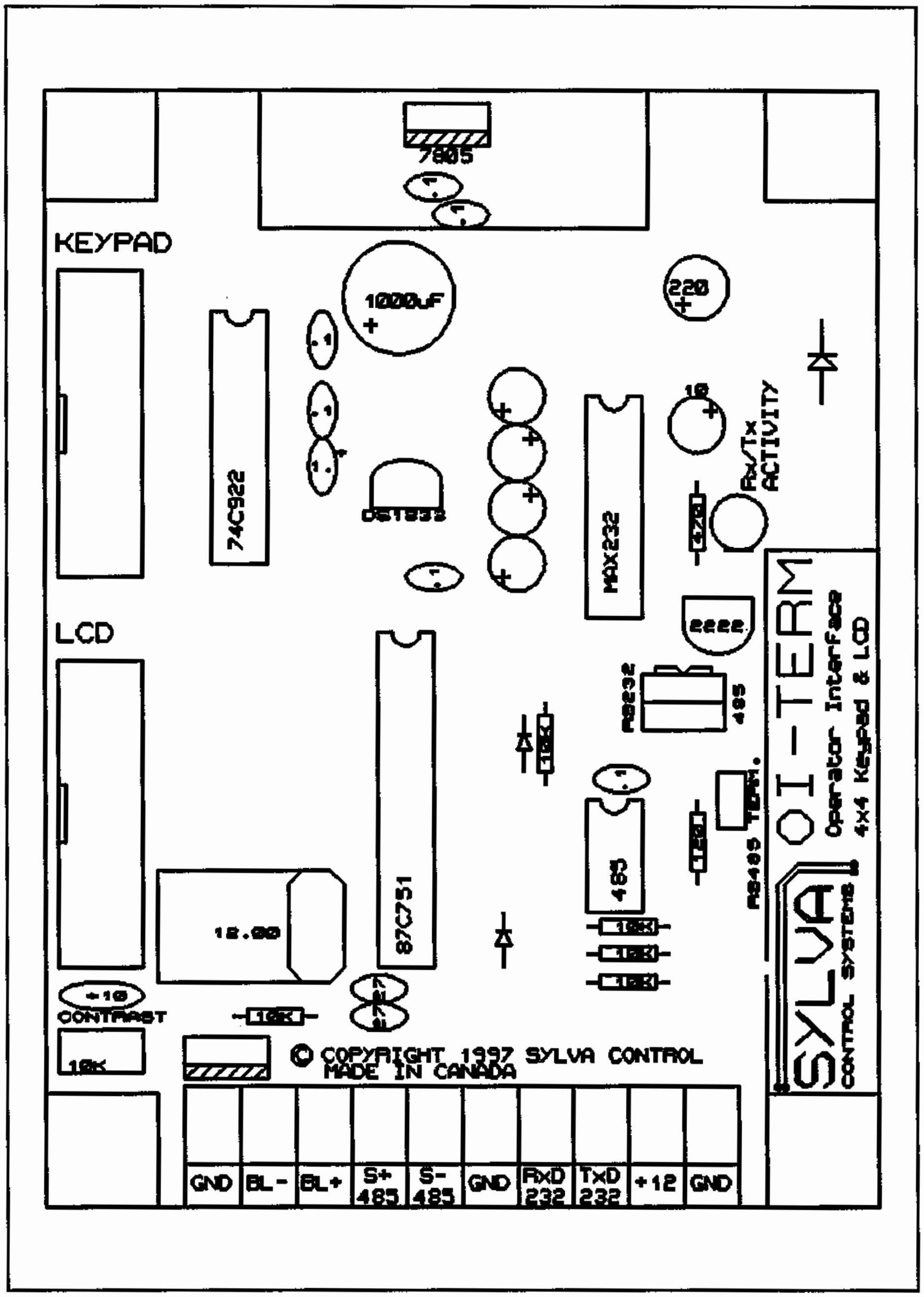
EXAMPLE:

```
100     XIH 059: XIH 063: GOSUB 1000: CLW 07
```

The bits set are cleared to zero after the execution of the subroutine at 1000.

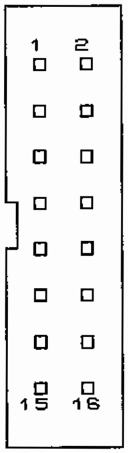
Copyright © 1997 Sylva Control Systems

OI-TERM Connections/Layout



KEYPAD & LCD Connections

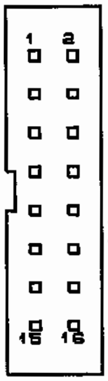
KEYPAD CONNECTOR PINOUT



1,2-ROW Y1
3,4-ROW Y2
5,6-ROW Y3
7,8-ROW Y4
9,10-ROW X1
11,12-ROW X2
13,14-ROW X3
15,16-ROW X4

TOP VIEW

LCD DISPLAY CONTECTOR PINOUT



1-data 6
2-data 7
3-data 4
4-data 5
5-data 2
6-data 3
7-data 0
8-data 1
9-R/W control
10-E enable (upper)
11-Vee (contrast supply)
12-RS register control
13-Vcc (+5V)
14-Vss (ground)
15-N.C.
16-E2 enable (lower)

TOP VIEW

Copyright © 1997 Sylva Control Systems

OI-TERM LCD Display / Keypad Controller

The OI-TERM provides a simple remote or local serial linked LCD display and keypad controller. The liquid crystal display controller responds to a simple command set which allows the clearing of the display and cursor positioning. Once the cursor is positioned the display operates in an auto increment mode advancing 1 position to the right as characters are printed. The OI-TERM will work with any LCD panel that uses a HITACHI 44780 controller. The 16 pin display connector allows direct connection to LCD units supplied by SYLVA Control Systems. The 4x40 LCD units use a straight ribbon cable and the 2x40 LCD units use a twist cable (two outside lines twisted). A pin-out is provided for direct wiring to other displays.
The keypad interface connects to a 4x4 matrix keypad via a 16 pin header connector. The controller sends the ASCII equivalent of the key pressed plus 128 (high bit set) and a CR. This is done to emulate the codes sent by older BAC-KEY keypad controllers. The 552ES controllers also have ASM routines (accessed by CALLs) to convert the ASCII keys to I/O image table bits. This allows older BACKEYs and BACLCDs to be easily replaced by the newer IO-TERM board.

OI-TERM Commands (shown as BASIC Statements)

Cursor Position (upper display)	PRINT #CHR(27),CHR(1),CHR(*); * = 0 to 79
Cursor Position (lower display)	PRINT #CHR(27),CHR(2),CHR(*); * = 0 to 79
Clear display/Cursor home	PRINT #CHR(27),CHR(3);
Backlight ON	PRINT #CHR(27),CHR(4);
Backlight OFF	PRINT #CHR(27),CHR(5);
Clear upper display only	PRINT #CHR(27),"U";
Clear lower display only	PRINT #CHR(27),"L";
Reset controller	PRINT #CHR(27),"R";

Note: If the user connects a smaller display (ie 2x20) the first position on the bottom line always starts at 40.

The OI-TERM transmits the ASCII equivalents of the key pressed plus 128. The reason for sending the ASCII value plus 128 is that the LCD controller ignores any received data which is greater than 127. Therefore no transmitted data will be displayed on the LCD if it's connected to a controller using the RS485 link. The keys are not repeated if held down. The ASCII transmission is initiated only on a key-down transition.

The SC552ES controllers have special ASM calls to convert the ASCII keycodes to the equivalent of the 2 bit code used by the old BACKEY controllers. The calling address depends on which of the 3 serial ports are connected to the OI-TERM board. The console port uses **361Ahex**, the AUX RS232 port uses **361Dhex** and the RS485 port uses **3620hex**.

The following is a table of 2 bit codes when the special ASM call is used in the COMINT routine:

```
Key 0 - bit 056 & 60  (XIH 056: XIH 060: PRINT "0")
Key 1 - bit 057 & 60  (XIH 057: XIH 060: PRINT "1")
Key 2 - bit 058 & 60  (XIH 058: XIH 060: PRINT "2")
Key 3 - bit 059 & 60  (XIH 059: XIH 060: PRINT "3")
Key 4 - bit 056 & 61  (XIH 056: XIH 061: PRINT "4")
Key 5 - bit 057 & 61  (XIH 057: XIH 061: PRINT "5")
Key 6 - bit 058 & 61  (XIH 058: XIH 061: PRINT "6")
Key 7 - bit 059 & 61  (XIH 059: XIH 061: PRINT "7")
Key 8 - bit 056 & 62  (XIH 056: XIH 062: PRINT "8")
Key 9 - bit 057 & 62  (XIH 057: XIH 062: PRINT "9")
Key A - bit 058 & 62  (XIH 058: XIH 062: PRINT "A")
Key B - bit 059 & 62  (XIH 059: XIH 062: PRINT "B")
Key C - bit 056 & 63  (XIH 056: XIH 063: PRINT "C")
Key D - bit 057 & 63  (XIH 057: XIH 063: PRINT "D")
Key E - bit 058 & 63  (XIH 058: XIH 063: PRINT "E")
Key F - bit 059 & 63  (XIH 059: XIH 063: PRINT "F")
```

Note: The special ASM call doesn't clear the bit state as in the old BACKEYs. The user should execute a CLW 07 after the bit testing.

EXAMPLE:

```
100     XIH 059: XIH 063: GOSUB 1000: CLW 07
```

The bits set are cleared to zero after the execution of the subroutine at 1000.

See APPLICATION PROGRAM NOTES for demo programs using the OI-TERM.

OI-TERM Addendum

This addendum applies to the new V1.2 OI-TERM controller chip.
The new V1.2 software will not allow the keypad character to be transmitted back to the controller until a CR is received by the OI-TERM.
It is recommended that users do all of the LCD printting in one subroutine with all CRs suppressed until the last bit of data is transmitted. By using this method the transmitted key data will not collide with data being transmitted to the LCD display. This is very important for a RS485 linked OI-TERM board. There are two example programs which use this method. They are OIRS232.BAS and OIRS485.BAS, both can be found at the back of this manual.

Also, OI-TERM boards have a modification which connects the collector of the back lighting switching transistor to pin #15 on the LCD display connector. All LCD panels supplied by Sylva have the back light LEDs connected to this pin so no additional wires are needed to operate the back lighting LEDs.

```
1       REM *OIRS232.BAS*
5       COMINT %,1000: CIC %,1
6       CLEAR R
10      PRINT%CHR(27),"U"
20      FOR T=1 TO 400: NEXT T
30      PRINT%CHR(27),"L"
40      FOR T=1 TO 400: NEXT T
100     REM LOOP
110     XIH 510: OST 000: GOSUB 2000
120     XIL 510: ROS 000
800     FLFP 012: LIO: GOTO 100

1000    CIC %,0: OTL 100
1001    IF XBY(0C00H)=176 FLFP 000
1002    IF XBY(0C00H)=177 FLFP 001
1003    IF XBY(0C00H)=178 FLFP 002
1004    IF XBY(0C00H)=179 FLFP 003
1005    IF XBY(0C00H)=180 FLFP 004
1006    IF XBY(0C00H)=181 FLFP 005
1007    IF XBY(0C00H)=182 FLFP 006
1008    IF XBY(0C00H)=183 FLFP 007
1009    IF XBY(0C00H)=184 FLFP 008
1010    IF XBY(0C00H)=185 FLFP 009
1020    FOR X=0C00H TO 0C02H: PRINT XBY(X),: NEXT X: PRINT
1099    CIC %,1: RETI

2000    XIH 100: PRINT% CHR(27),CHR(3);: FOR T=1 TO 50: NEXT T: OTU 100
2005    PRINT%CHR(27),CHR(1),CHR(0);
2010    PRINT%"0123456789012345678901234567890123456789";
2020    PRINT%CHR(27),CHR(1),CHR(40);
2030    PRINT%"ABCDEFGHIJKLMNOPQRSTUVWXYZ----0123456789";
2040    PRINT%CHR(27),CHR(2),CHR(0);
2050    PRINT%"This is just a test message to fill the ";
2060    PRINT%CHR(27),CHR(2),CHR(40);
2070    PRINT%"lcd display with stuff!";
2080    PRINT%CHR(27),CHR(2),CHR(70);: TIME %: REM here is the last CR
2099    RETURN
```

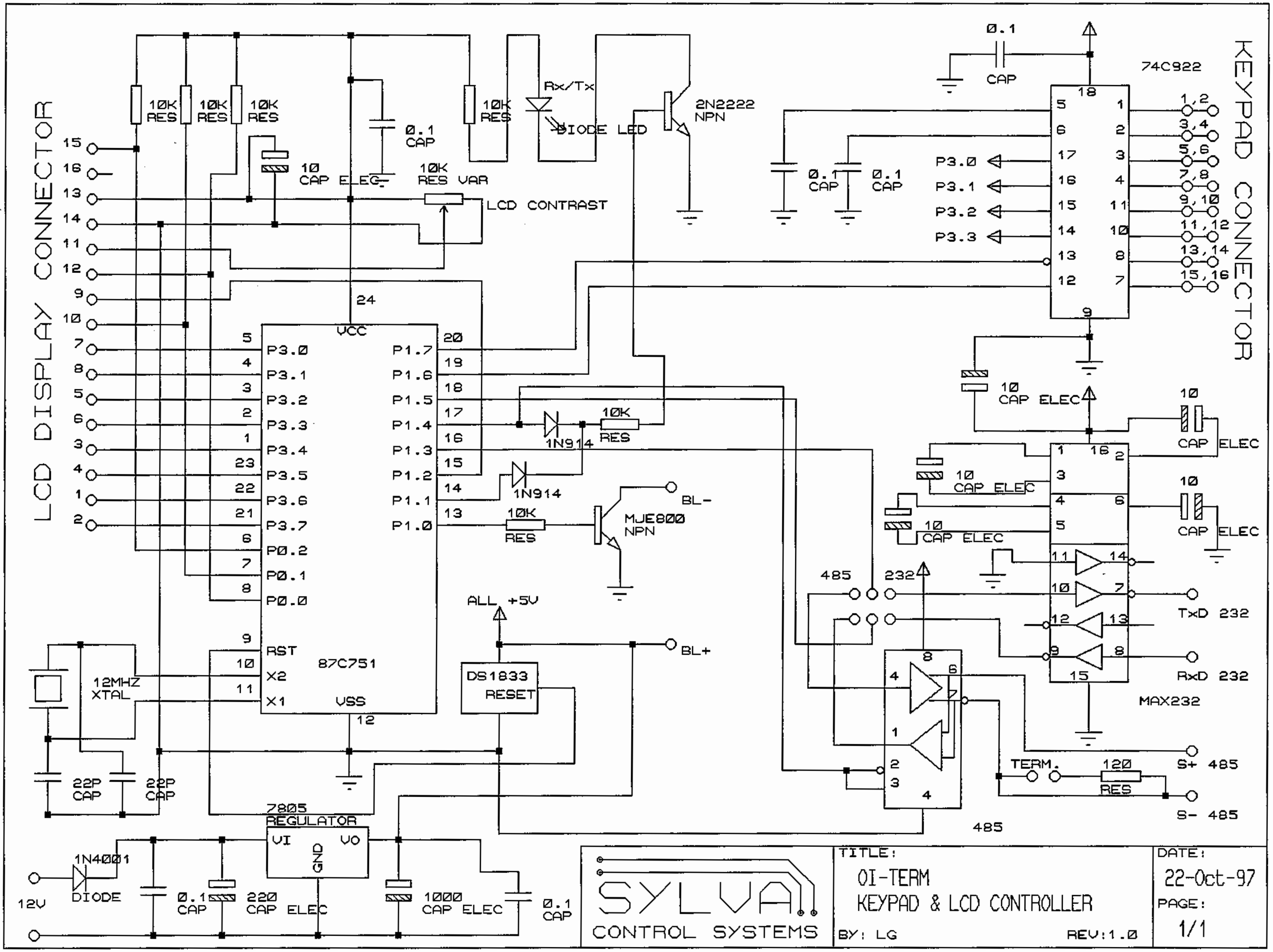
```
1       REM *OIRS485.BAS*
5       COMINT #,1000: CIC #,1
6       CLEAR R
10      PRINT#CHR(27),"U"
20      FOR T=1 TO 400: NEXT T
30      PRINT#CHR(27),"L"
40      FOR T=1 TO 400: NEXT T
100     REM LOOP
110     XIH 510: OST 000: GOSUB 2000
120     XIL 510: ROS 000
800     FLFP 012: LIO: GOTO 100

1000    CIC #,0: OTL 100
1001    IF XBY(0D00H)=176 FLFP 000
1002    IF XBY(0D00H)=177 FLFP 001
1003    IF XBY(0D00H)=178 FLFP 002
1004    IF XBY(0D00H)=179 FLFP 003
1005    IF XBY(0D00H)=180 FLFP 004
1006    IF XBY(0D00H)=181 FLFP 005
1007    IF XBY(0D00H)=182 FLFP 006
1008    IF XBY(0D00H)=183 FLFP 007
1009    IF XBY(0D00H)=184 FLFP 008
1010    IF XBY(0D00H)=185 FLFP 009
1020    FOR X=0D00H TO 0D02H: PRINT XBY(X),: NEXT X: PRINT
1099    CIC #,1: RETI

2000    XIH 100: PRINT# CHR(27),CHR(3);: FOR T=1 TO 50: NEXT T: OTU 100
2005    PRINT#CHR(27),CHR(1),CHR(0);
2010    PRINT#"0123456789012345678901234567890123456789";
2020    PRINT#CHR(27),CHR(1),CHR(40);
2030    PRINT#"ABCDEFGHIJKLMNOPQRSTUVWXYZ----0123456789";
2040    PRINT#CHR(27),CHR(2),CHR(0);
2050    PRINT#"This is just a test message to fill the ";
2060    PRINT#CHR(27),CHR(2),CHR(40);
2070    PRINT#"lcd display with stuff!";
2080    PRINT#CHR(27),CHR(2),CHR(70);: TIME #: REM here is the last CR
2099    RETURN
```

176
128
048

Copyright © 1997 Sylva Control Systems



SYLVA

CONTROL SYSTEMS

TITLE:

OI-TERM

KEYPAD & LCD CONTROLLER

BY: LG

REV:1.0

DATE:

22-Oct-97

PAGE:

1/1

KEY 485. BAS

```
1       REM The following program demonstrates how to use the RS485 keypad
2       REM interface.
3       REM If a 'C' is pressed then counter 000 is reset.
4       REM The next 4 keys will be tested to see if they match the code 05AF
5       REM If they match then bit 103 will be flipped and the console will
6       REM print "ALARM ENABLED"
7       REM The counter to track the sequence of key presses for the code.
8       REM Anytime the sequence fails the counter is cleared.
9       REM By using this method many combinations can be used. The first key
10      REM would be used to clear a specific counter and an extra
11      REM bit would have to be used to direct the next key to the correct
12      REM testing routine.
13      REM
14      REM THIS PROGRAM IS VERY SIMILAR TO THE I2CKEY.BAS PROGRAM. THE ONLY
15      REM DIFFERENCE IS THE SERIAL INTERRUPT ROUTINE, WHICH CONVERTS THE
16      REM KEY CODES TO A VALUE BETWEEN 0-15. THIS IS THEN USED TO SET THE
17      REM @JMP STATEMENT. JMPs BETWEEN 000 AND 015 WILL BE SET DEPENDING
18      REM WHICH KEY WAS PRESSED. THE LBL CAN THEN BE USED TO TEST WHICH KEY.
19      REM Remember that the LBL will clear itself after execution!
20      REM This gives us the onesot effect as in the I2C mode.
21      REM
22      COMINT 2000: CIC 1
25      CLEAR C: CLEAR O: CLW 00: CLW 01: CLEAR J
26      CLW 32: CLW 33
30      REM program loop
35      REM keys 0-3
40      LBL 000: FLFP 000: OTL 256
45      LBL 001: FLFP 001: OTL 257
50      LBL 002: FLFP 002: OTL 258
55      LBL 003: FLFP 003: OTL 259
60      REM keys 4-7
65      LBL 004: FLFP 004: OTL 260
70      LBL 005: FLFP 005: OTL 261
75      LBL 006: FLFP 006: OTL 262
80      LBL 007: FLFP 007: OTL 263
85      REM keys 8-B
90      LBL 008: FLFP 008: OTL 264
95      LBL 009: FLFP 009: OTL 265
100     LBL 010: PRINT "KEY A": OTL 266
105     LBL 011: PRINT "KEY B": OTL 267
106     REM keys C-F
110     LBL 012: PRINT "KEY C": CTR 000: OTL 268
115     LBL 013: PRINT "KEY D": OTL 269
120     LBL 014: PRINT "KEY E": OTL 270
125     LBL 015: PRINT "KEY F": OTL 271
130     REM now test for a 4 key code 0 A 5 F
133     REM Words 32(290h) and 33(291h) are tested for a key press condition
135     IF XBY(290H)>0 THEN GOSUB 1000
140     IF XBY(291H)>0 THEN GOSUB 1000
150     XIH 103: ROS 001: OST 000: CTR 000: PRINT "* ALARM ENABLED *": JMP 030
160     XIL 103: ROS 000: OST 001: CTR 000: PRINT " ALARM DISABLED ": JMP 031
170     LBL 030: PRINT #CHR(28),CHR(1);: REM led ON
175     LBL 031: PRINT #CHR(28),CHR(2);: REM led OFF
180     FLFP 012
190     LIO : GOTO 30
1000    CTU 000,005: GOTO 1060
1010    XCT 000,001: XIH 256: OTL 100: CLW 32: CLW 33: RETURN
1020    XCT 000,002: XIH 261: XIH 100: OTL 101: CLW 32: CLW 33: RETURN
1030    XCT 000,003: XIH 266: XIH 101: OTL 102: CLW 32: CLW 33: RETURN
```

```
1040   XCT 000,004: XIH 271: XIH 102: FLFP 103: CTR 000: JMP 020
1050   LBL 020: OTU 100: OTU 101: OTU 102: CLW 32: CLW 33: RETURN
1060   OTU 100: OTU 101: OTU 102: CTR 000: CLW 32: CLW 33: RETURN
1070   REM Words 32 & 33 are cleared to turn OFF any bits set ON by LBLs
2000   REM communication interrupt routine
2010   CIC 0:A=INKEY
2015   XSB "STOP": STOP
2020   FLFP 000: REM flip relay so we know when we are here
2030   REM convert key codes to a decimal number from 0-15
2040   IF A<175 THEN  CIC 1: RETI
2050   IF A>198 THEN  CIC 1: RETI
2055   REM BACKEY sends ASCII + 128 so BACLCD will not print any characters
2060  A=A-128: CODE=A-48: IF A>57 THEN CODE=A-55
2065   REM load memory equal to @000
2070  XBY(800H)=CODE: PRINT CODE
2080   JMP @000
2090   CIC 1
2099   RETI
```

□

KEY485.BAS

```
1       REM The following program demonstrates how to use the
RS485 keypad
2       REM interface.
3       REM If a 'C' is pressed then counter 000 is reset.
4       REM The next 4 keys will be tested to see if they match
the code 05AF
5       REM If they match then bit 103 will be flipped and the
console will
6       REM print "ALARM ENABLED"
7       REM The counter to track the sequence of key presses for
the code.
8       REM Anytime the sequence fails the counter is cleared.
9       REM By using this method many combinations can be used.
The first key
10      REM would be used to clear a specific counter and an
extra
11      REM bit would have to be used to direct the next key to
the correct
12      REM testing routine.
13      REM
14      REM THIS PROGRAM IS VERY SIMILAR TO THE I2CKEY.BAS
PROGRAM. THE ONLY
15      REM DIFFERENCE IS THE SERIAL INTERRUPT ROUTINE, WHICH
CONVERTS THE
16      REM KEY CODES TO A VALUE BETWEEN 0-15. THIS IS THEN USED
TO SET THE
17      REM @JMP STATEMENT. JMPs BETWEEN 000 AND 015 WILL BE SET
DEPENDING
18      REM WHICH KEY WAS PRESSED. THE LBL CAN THEN BE USED TO
TEST WHICH KEY.
19      REM Remember that the LBL will clear itself after
execution!
20      REM This gives us the one shot effect as in the I2C mode.
21      REM
22      COMINT 2000: CIC 1
25      CLEAR C: CLEAR O: CLW 00: CLW 01: CLEAR J
26      CLW 32: CLW 33
30      REM program loop
35      REM keys 0-3
40      LBL 000: FLFP 000: OTL 256
45      LBL 001: FLFP 001: OTL 257
50      LBL 002: FLFP 002: OTL 258
55      LBL 003: FLFP 003: OTL 259
60      REM keys 4-7
65      LBL 004: FLFP 004: OTL 260
70      LBL 005: FLFP 005: OTL 261
75      LBL 006: FLFP 006: OTL 262
80      LBL 007: FLFP 007: OTL 263
```

```
85      REM keys 8-B
90      LBL 008: FLFP 008: OTL 264
95      LBL 009: FLFP 009: OTL 265
100     LBL 010: PRINT "KEY A": OTL 266
105     LBL 011: PRINT "KEY B": OTL 267
106     REM keys C-F
110     LBL 012: PRINT "KEY C": CTR 000: OTL 268
115     LBL 013: PRINT "KEY D": OTL 269
120     LBL 014: PRINT "KEY E": OTL 270
125     LBL 015: PRINT "KEY F": OTL 271
130     REM now test for a 4 key code 0 A 5 F
133     REM Words 32(290h) and 33(291h) are tested for a key
press condition
135     IF XBY(290H)>0 THEN GOSUB 1000
140     IF XBY(291H)>0 THEN GOSUB 1000
150     XIH 103: ROS 001: OST 000: CTR 000: PRINT "* ALARM
ENABLED *": JMP 030
160     XIL 103: ROS 000: OST 001: CTR 000: PRINT " ALARM
DISABLED ": JMP 031
170     LBL 030: PRINT #CHR(28),CHR(1);: REM led ON
175     LBL 031: PRINT #CHR(28),CHR(2);: REM led OFF
180     FLFP 012
190     LIO : GOTO 30
1000    CTU 000,005: GOTO 1060
1010    XCT 000,001: XIH 256: OTL 100: CLW 32: CLW 33: RETURN
1020    XCT 000,002: XIH 261: XIH 100: OTL 101: CLW 32: CLW 33:
RETURN
1030    XCT 000,003: XIH 266: XIH 101: OTL 102: CLW 32: CLW 33:
RETURN
1040    XCT 000,004: XIH 271: XIH 102: FLFP 103: CTR 000: JMP 020
1050    LBL 020: OTU 100: OTU 101: OTU 102: CLW 32: CLW 33:
RETURN
1060    OTU 100: OTU 101: OTU 102: CTR 000: CLW 32: CLW 33:
RETURN
1070    REM Words 32 & 33 are cleared to turn OFF any bits set
ON by LBLs
2000    REM communication interrupt routine
2010    CIC 0:A=INKEY
2015    XSB "STOP": STOP
2020    FLFP 000: REM flip relay so we know when we are here
2030    REM convert key codes to a decimal number from 0-15
2040    IF A<175 THEN  CIC 1: RETI
2050    IF A>198 THEN  CIC 1: RETI
2055    REM SC-KEY sends ASCII + 128 so BACLCD will not print
any characters
2060   A=A-128: CODE=A-48: IF A>57 THEN CODE=A-55
2065    REM load memory equal to @000
2070   XBY(800H)=CODE: PRINT CODE
2080    JMP @000
2090    CIC 1
```

```
2099    RETI
```

I2CKEY.BAS

```
1       REM The following program demonstrates how to use the
I2C keypad
2       REM interface. The program will do something for every
key pressed.
3       REM If a 'C' is pressed then counter 000 is reset.
4       REM The next 4 keys will be tested to see if they match
the code 05AF
5       REM If they match then bit 103 will be flipped and the
console will
6       REM print "ALARM ENABLED"
7       REM The key 'C' is used to clear the counter to track
the sequence
8       REM It could be said that the actual code is = C05AF
9       REM By using this method many combinations can be used.
The first key
10      REM would be used to clear a specific counter and an
extra
11      REM bit would have to be used to direct the next key to
the correct
12      REM testing routine.
13      REM
14      REM AN IMPORTANT FEATURE OF THE I2C KEY INTERFACE IS
THAT THE IO BITS
15      REM THAT REPRESENT A KEY CLOSURE ARE ONLY SET ON UNTIL
THE NEXT
16      REM EXIO 01,2. SIMPLY, THE BAC552 WILL ONLY SEE THE KEY
CLOSURE AS A
17      REM MOMENTARY BIT AFTER THE EXIO. THAT IS WHY ALL BIT
TESTING IS DONE
18      REM AFTER THE EXIO 01,2. THIS GIVES A ONE-SHOT EFFECT
FOR KEYS PRESSED
19      REM
20      REM I2CKEY Program DEMO
25      CLEAR C: CLEAR O: CLW 00: CLW 01: CLEAR J
30      EXIO 01,2: REM read and write data to/from SC-KEY
35      REM keys 0-3
40      XIH 056: XIH 060: FLFP 000
45      XIH 057: XIH 060: FLFP 001
50      XIH 058: XIH 060: FLFP 002
55      XIH 059: XIH 060: FLFP 003
60      REM keys 4-7
65      XIH 056: XIH 061: FLFP 004
70      XIH 057: XIH 061: FLFP 005
75      XIH 058: XIH 061: FLFP 006
80      XIH 059: XIH 061: FLFP 007
85      REM keys 8-B
90      XIH 056: XIH 062: FLFP 008
```

```
95      XIH 057: XIH 062: FLFP 009
100     XIH 058: XIH 062: PRINT "KEY A"
105     XIH 059: XIH 062: PRINT "KEY B"
106     REM keys C-F
110     XIH 056: XIH 063: PRINT "KEY C": CTR 000
115     XIH 057: XIH 063: PRINT "KEY D"
120     XIH 058: XIH 063: PRINT "KEY E"
125     XIH 059: XIH 063: PRINT "KEY F"
130     REM now test for a 4 key code 0 A 5 F
135     REM the upper 4 bits 060-063 can be used to indicate a
key press
136     REM memory location 277H is the word for bits 056-063
137     REM Decimal 240 (11110000b) masks out the lower bits
140     IF XBY(277H).AND.240>0 THEN  GOSUB 1000
145     REM The bit 048 is an output on the SC-KEY which can
show our alarm
146     REM status at the keypad location (048 = Output0)
150     XIH 103: ROS 001: OST 000: CTR 000: OTL 048: PRINT "*
ALARM ENABLED *"
160     XIL 103: ROS 000: OST 001: CTR 000: OTU 048: PRINT "
ALARM DISABLED "
180     FLFP 012
190     LIO : GOTO 30
1000    CTU 000,005: GOTO 1060
1010    XCT 000,001: XIH 056: XIH 060: OTL 100: RETURN
1020    XCT 000,002: XIH 057: XIH 061: XIH 100: OTL 101: RETURN
1030    XCT 000,003: XIH 058: XIH 062: XIH 101: OTL 102: RETURN
1040    XCT 000,004: XIH 059: XIH 063: XIH 102: FLFP 103: CTR
000: JMP 000
1050    LBL 000: OTU 100: OTU 101: OTU 102: RETURN
1060    OTU 100: OTU 101: OTU 102: CTR 000: RETURN
```

ES485KEY.BAS

```
1       REM The following program demonstrates how to use the
RS485 keypad
2       REM for the 552ES Controller.
3       REM If a 'C' is pressed then counter 000 is reset.
4       REM The next 4 keys will be tested to see if they match
the code 05AF
5       REM If they match then bit 103 will be flipped and the
console will
6       REM print "ALARM ENABLED"
7       REM The counter to track the sequence of key presses for
the code.
8       REM Anytime the sequence fails the counter is cleared.
9       REM By using this method many combinations can be used.
The first key
10      REM would be used to clear a specific counter and an
extra
```

```
11      REM bit would have to be used to direct the next key to
the correct
12      REM testing routine.
13      REM
14      REM THIS PROGRAM IS VERY SIMILAR TO THE I2CKEY.BAS
PROGRAM. THE ONLY
15      REM DIFFERENCE IS THE SERIAL INTERRUPT ROUTINE, WHICH
CONVERTS THE
16      REM KEY CODES TO A VALUE BETWEEN 0-15. THIS IS THEN USED
TO SET THE
17      REM @JMP STATEMENT. JMPs BETWEEN 000 AND 015 WILL BE SET
DEPENDING
18      REM WHICH KEY WAS PRESSED. THE LBL CAN THEN BE USED TO
TEST WHICH KEY.
19      REM Remember that the LBL will clear itself after
execution!
20      REM This gives us the onesot effect as in the I2C mode.
21      REM
22      COMINT #,2000: CIC #,1
23      COMINT 3000: CIC 1
25      CLEAR C: CLEAR O: CLW 00: CLW 01: CLEAR J
26      CLW 32: CLW 33
30      REM program loop
35      REM keys 0-3
40      LBL 000: FLFP 000: OTL 256
45      LBL 001: FLFP 001: OTL 257
50      LBL 002: FLFP 002: OTL 258
55      LBL 003: FLFP 003: OTL 259
60      REM keys 4-7
65      LBL 004: FLFP 004: OTL 260
70      LBL 005: FLFP 005: OTL 261
75      LBL 006: FLFP 006: OTL 262
80      LBL 007: FLFP 007: OTL 263
85      REM keys 8-B
90      LBL 008: FLFP 008: OTL 264
95      LBL 009: FLFP 009: OTL 265
100     LBL 010: PRINT "KEY A": OTL 266
105     LBL 011: PRINT "KEY B": OTL 267
106     REM keys C-F
110     LBL 012: PRINT "KEY C": CTR 000: OTL 268
115     LBL 013: PRINT "KEY D": OTL 269
120     LBL 014: PRINT "KEY E": OTL 270
125     LBL 015: PRINT "KEY F": OTL 271
130     REM now test for a 4 key code 0 A 5 F
133     REM Words 32(290h) and 33(291h) are tested for a key
press condition
135     IF XBY(290H)>0 THEN  GOSUB 1000
140     IF XBY(291H)>0 THEN  GOSUB 1000
150     XIH 103: ROS 001: OST 000: CTR 000: PRINT "* ALARM
ENABLED *": JMP 030
```

```
160     XIL 103: ROS 000: OST 001: CTR 000: PRINT " ALARM
DISABLED ": JMP 031
170     LBL 030: PRINT#CHR(28),CHR(1);: REM led ON
175     LBL 031: PRINT#CHR(28),CHR(2);: REM led OFF
180     FLFP 012
190     LIO : GOTO 30
1000    CTU 000,005: GOTO 1060
1010    XCT 000,001: XIH 256: OTL 100: CLW 32: CLW 33: RETURN
1020    XCT 000,002: XIH 261: XIH 100: OTL 101: CLW 32: CLW 33:
RETURN
1030    XCT 000,003: XIH 266: XIH 101: OTL 102: CLW 32: CLW 33:
RETURN
1040    XCT 000,004: XIH 271: XIH 102: FLFP 103: CTR 000: JMP 020
1050    LBL 020: OTU 100: OTU 101: OTU 102: CLW 32: CLW 33:
RETURN
1060    OTU 100: OTU 101: OTU 102: CTR 000: CLW 32: CLW 33:
RETURN
1070    REM Words 32 & 33 are cleared to turn OFF any bits set
ON by LBLs
2000    REM communication interrupt routine
2010    CIC #,0:A=XBY(0D00H)
2020    FLFP 000: REM flip relay so we know when we are here
2030    REM convert key codes to a decimal number from 0-15
2040    IF A<175 THEN  CIC #,1: RETI
2050    IF A>198 THEN  CIC #,1: RETI
2055    REM SC-KEY sends ASCII + 128 so BACLCD will not print
any characters
2060   A=A-128:CODE=A-48: IF A>57 THEN CODE=A-55
2065    REM load memory equal to @000
2070   XBY(800H)=CODE: PRINT CODE
2080    JMP @000
2090    CIC #,1
2099    RETI
3000    CIC 0: IF INKEY=3 THEN  STOP
3005    CIC 1: RETI
```

SC-KEY on RS485 INTEGER INPUT ROUTINE with ES ROM V1.542

```
5       REM *** INTEGER INPUT FOR SC-KEY WITH 52ES ROM VERSION
1.542+ ***
20      COMINT #,2000: CIC #,1: A_P=0
100     FOR T=1 TO 100: NEXT T
120     FLFP 012: LIO: GOTO 100
1990    REM This COMINT subroutine gets keypad input data and
converts it to
1991    REM to a 16 BIT INTEGER value in variable KEY_PAD. The
max value is
1992    REM 65535 decimal. The ASM routine will drop extra input
if the value
1993    REM exceeds the 16 bit limit.
1994    REM Example 65536 will be cut to 6553
```

```
1995   REM The 'E' is ENTER. This routine is much faster than
the floating
1996   REM point CALL to 35F3H.

2000   CIC #,0:A=XBY(0D00H)
2010   IF A<175.OR.A>198 THEN CIC #,1: RETI
2030  A=A-128: REM strip bit 7 from data
2037   IF A=69.AND.A_P=0 THEN XBY(02E0H)=48:A_P=A_P+1: REM
force a zero
2040   IF A=69 THEN XBY(02E0H+A_P)=13: GOTO 2070
2050   IF A<58.AND.A>47 THEN XBY(02E0H+A_P)=A:A_P=A_P+1: PRINT
CHR(A),
2060   CIC #,1: RETI
2070   CALL 35F0H: POP KEY_PAD:A_P=0
2075   PRINT" The integer value input is ",KEY_PAD
2080   CIC #,1: RETI
```

SC-KEY on RS485 FLOATING POINT INPUT with ES ROM V1.542

```
5      REM *** FLOATING POINT FOR SC-KEY WITH BAC552ES ROM
VERSION 1.542 ***
20     COMINT #,2000: CIC #,1: A_P=0
100    FOR T=1 TO 100: NEXT T
120    FLFP 012: LIO: GOTO 100
1990   REM This COMINT subroutine gets keypad input data and
converts it to
1991   REM to a floating point value in variable KEY_PAD.
1992   REM The 'A' is the decimal, 'B' is a minus and the 'E'
is ENTER
2000   CIC #,0:A=XBY(0D00H)
2010   IF A<175.OR.A>198 THEN CIC #,1: RETI
2025  A=A-128: REM strip bit 7 from data
2030   IF A=65 THEN XBY(02E0H+A_P)=46: A_P=A_P+1: PRINT".",
2035   IF A=66 THEN XBY(02E0H+A_P)=45: A_P=A_P+1: PRINT"-",
2037   IF A=69.AND.A_P=0 THEN XBY(02E0H)=48:A_P=A_P+1: REM
force a zero
2040   IF A=69 THEN XBY(02E0H+A_P)=13: GOTO 2070
2050   IF A<58.AND.A>47 THEN XBY(02E0H+A_P)=A:A_P=A_P+1: PRINT
CHR(A),
2060   CIC #,1: RETI
2070   CALL 35F3H: POP KEY_PAD:A_P=0
2075   PRINT" The floating point input is ",KEY_PAD
2080   CIC #,1: RETI
```

ASC_I2C.BAS

```
5      REM *** FLOATING POINT INPUT FOR SC-KEY WITH ROM VERSION
1.542+ ***
20    A_P=0:XBY(0277H)=0: REM clear the received data location
100    FOR T=1 TO 10: NEXT T
105    OTL 055: EXIO 01,2: REM Force ASCII I2C MODE
110    IF XBY(0277H)>0 THEN  GOSUB 2000: REM if greater than 0
```

```
data is in
120    FLFP 012: LIO : GOTO 100

1990   REM This subroutine gets I2C ASCII keypad input data and
converts
1991   REM it to a floating point value in variable KEY_PAD.
1992   REM The 'A' is the decimal, 'B' is a minus and the 'E'
is ENTER
2000  A=XBY(0277H)
2030   IF A=65 THEN XBY(02E0H+A_P)=46:A_P=A_P+1: PRINT ".",
2035   IF A=66 THEN XBY(02E0H+A_P)=45:A_P=A_P+1: PRINT "-",
2037   IF A=69.AND.A_P=0 THEN XBY(02E0H)=48:A_P=A_P+1: REM
force a zero
2040   IF A=69 THEN XBY(02E0H+A_P)=13: GOTO 2070
2050   IF A<58.AND.A>47 THEN XBY(02E0H+A_P)=A:A_P=A_P+1: PRINT
CHR(A),
2060   RETURN
2070   CALL 35F3H: POP KEY_PAD:A_P=0
2075   PRINT " The floating point input is ",KEY_PAD
2080   RETURN
```