Model V580

50 to 68 - Channel Multiplexer Module

INSTRUCTION MANUAL

February, 1996

(C) 1996
Copyright by
KineticSystems Corporation
Lockport, Illinois
All rights reserved

CONTENTS

Tables

Figures

UNPACKING AND INSTALLATION

At KineticSystems, static precautions are observed during all phases of production, test, and packaging of each module. This includes using static proof mats and wrist straps. Please observe these same precautions when unpacking and installing the module whenever possible.

The Model V580 is shipped in an anti-static bag within a Styrofoam packing container. Carefully remove the module from its static-proof bag and prepare to set the logical address switches to the appropriate value.

Configuration

There is one set of user configurable switches on the V580. All eight switches, #1 (msb) to #8 (lsb), are for the logical address. The logical address may be set from 1 to 254 as a statically configured device. If the module is set for logical address 255 (all switches open), then the V580 will be dynamically configured by the resource manager. Logical address 255 is the factory default setting. (See Figure 1, page 1)

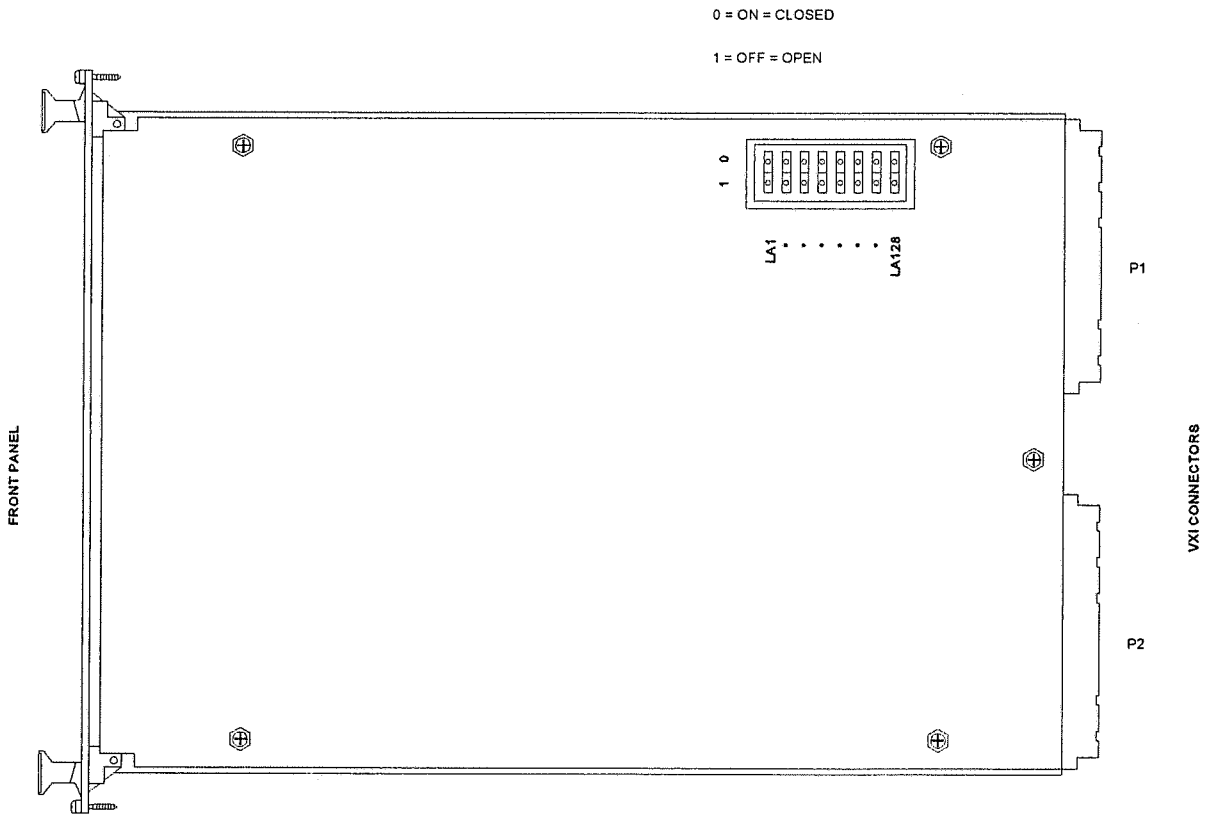


Figure 1 - V580 Switch Locations

Module Insertion

The V580 is a C-sized, single width, VXIbus module. Except for Slot 0, it can be mounted in any unoccupied slot in a C-size VXIbus main frame.

CAUTION:	TURN MAINFRAME POWER OFF WHEN INSERTING OR REMOVING MODULE

FRONT PANEL DESCRIPTION

LED

The Add Rec LED lights to indicate that the V580 is being accessed through VXI.

Connectors

External signals are connected to the V580 via two 2-Contact LEMOs labeled "Instr A" and "Instr B" and two high density connectors labeled P3 and P4. The V580-ZA11 option has two 50 pin, high density SCSI II type connectors. Beginning at the top of the module, connector P3 receives the differential input pairs for channels 1 through 25, connector P4 receives channels 26 through 50. The V580-ZB11 option has two 68 pin, high density SCSI II type connectors. Beginning at the top of the module, connector P3 receives the differential input pairs for channels 1 through 34, connector P4 receives channels 35 through 68. See Figures 3 and 4 and Tables 1, 2, 3, and 4 for the precise pinout descriptions .

The signals on the P3 connector are routed internally to what will be referred to in this manual as path A. Which should not be confused with the 2-pin front panel LEMO labeled "Instr A". The P4 connector is routed internally to path B again not to be confused with the 2-pin front panel LEMO labeled "Instr B".

PROGRAMMING INFO

VXIbus Addressing

The V580 is classified as an extended register device which means it has registers that occupy A16 and A24 space.

The configuration registers are located in A16 space and include the standard registers defined by VXI as well as additional registers to help identify the module. An additional register defined by KineticSystems is the Suffix Register. The Suffix Register includes the model suffix number which indicates the model option. The User Defined Registers at the end of the configuration space can also be used to identify the module (i.e., with an internal identification number). The operational registers are located in A24 space and include the registers specific to V580 modules.

Each option of the V580 may be setup in several modes of operation. The ZA11 option may be configured as two 1x25 or one 1x50 multiplexer. The ZB11 option may be configured as two 1x34 or as one 1x68 multiplexer. The 2-contact LEMO's are connected to test equipment such as a function generator, variable voltage source, or DVM.

Appendix A includes an example in using a V580-ZA11 and a V580 ZB11.

Appendix B includes a C code example using NI-VXI routines with V580. It includes codes which will use the configuration registers to help identify the specific module and access both configuration and operational registers.

Resetting the V580

The V580 can be put into soft reset by setting the Reset bit of the Status Control Register.

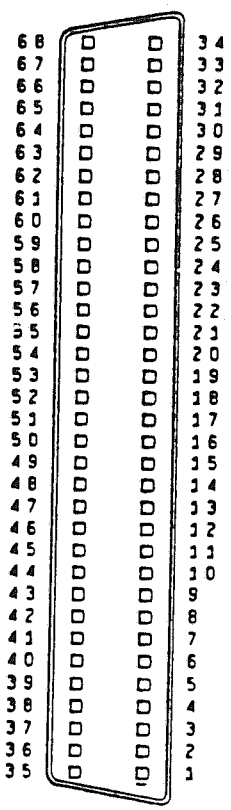


Figure 2 - 68 Pin SCSI II Connector

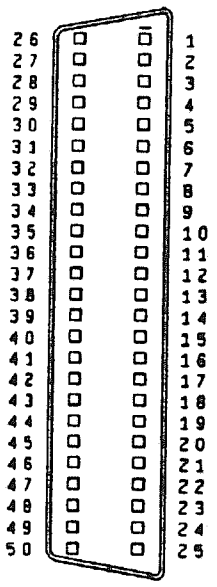


Figure 3 - 50 Pin SCSI II Connector

TABLE 1 - P3 68-Pin SCSI II Connector Pinout (Front View)

Pin #	Description	Pin #	Description
35	Channel 1 Out -	1	Channel 1 Out +
36	Channel 2 Out -	2	Channel 2 Out +
37	Channel 3 Out -	3	Channel 3 Out +
38	Channel 4 Out -	4	Channel 4 Out +
39	Channel 5 Out -	5	Channel 5 Out +
40	Channel 6 Out -	6	Channel 6 Out +
41	Channel 7 Out -	7	Channel 7 Out +
42	Channel 8 Out -	8	Channel 8 Out +
43	Channel 9 Out -	9	Channel 9 Out +
44	Channel 10 Out -	10	Channel 10 Out +
45	Channel 11 Out -	11	Channel 11 Out +
46	Channel 12 Out -	12	Channel 12 Out +
47	Channel 13 Out -	13	Channel 13 Out +
48	Channel 14 Out -	14	Channel 14 Out +
49	Channel 15 Out -	15	Channel 15 Out +
50	Channel 16 Out -	16	Channel 16 Out +
51	Channel 17 Out -	17	Channel 17 Out +
52	Channel 18 Out -	18	Channel 18 Out +
53	Channel 19 Out -	19	Channel 19 Out +
54	Channel 20 Out -	20	Channel 20 Out +
55	Channel 21 Out -	21	Channel 21 Out +
56	Channel 22 Out -	22	Channel 22 Out +
57	Channel 23 Out -	23	Channel 23 Out +
58	Channel 24 Out -	24	Channel 24 Out +
59	Channel 25 Out -	25	Channel 25 Out +
60	Channel 26 Out -	26	Channel 26 Out +
61	Channel 27 Out -	27	Channel 27 Out +
62	Channel 28 Out -	28	Channel 28 Out +
63	Channel 29 Out -	29	Channel 29 Out +
64	Channel 30 Out -	30	Channel 30 Out +
65	Channel 31 Out -	31	Channel 31 Out +
66	Channel 32 Out -	32	Channel 32 Out +
67	Channel 33 Out -	33	Channel 33 Out +
68	Channel 34 Out -	34	Channel 34 Out +

TABLE 2 - P4 68-Pin SCSI II Connector Pinout (Front View)

Pin #	Description	Pin #	Descrip-tion
35	Channel 35 Out -	1	Channel 35 Out +
36	Channel 36 Out -	2	Channel 36 Out +
37	Channel 37 Out -	3	Channel 37 Out +
38	Channel 38 Out -	4	Channel 38 Out +
39	Channel 39 Out -	5	Channel 39 Out +
40	Channel 40 Out -	6	Channel 40 Out +
41	Channel 41 Out -	7	Channel 41 Out +
42	Channel 42 Out -	8	Channel 42 Out +
43	Channel 43 Out -	9	Channel 43 Out +
44	Channel 44 Out -	10	Channel 44 Out +
45	Channel 45 Out -	11	Channel 45 Out +
46	Channel 46 Out -	12	Channel 46 Out +
47	Channel 47 Out -	13	Channel 47 Out +
48	Channel 48 Out -	14	Channel 48 Out +
49	Channel 49 Out -	15	Channel 49 Out +
50	Channel 50 Out -	16	Channel 50 Out +
51	Channel 51 Out -	17	Channel 51 Out +
52	Channel 52 Out -	18	Channel 52 Out +
53	Channel 53 Out -	19	Channel 53 Out +
54	Channel 54 Out -	20	Channel 54 Out +
55	Channel 55 Out -	21	Channel 55 Out +
56	Channel 56 Out -	22	Channel 56 Out +
57	Channel 57 Out -	23	Channel 57 Out +
58	Channel 58 Out -	24	Channel 58 Out +
59	Channel 57 Out -	25	Channel 59 Out +
60	Channel 60 Out -	26	Channel 60 Out +
61	Channel 61 Out -	27	Channel 61 Out +
62	Channel 62 Out -	28	Channel 62 Out +
63	Channel 63 Out -	29	Channel 63 Out +
64	Channel 64 Out -	30	Channel 64 Out +
65	Channel 65 Out -	31	Channel 65 Out +
66	Channel 66 Out -	32	Channel 66 Out +
67	Channel 67 Out -	33	Channel 67 Out +
68	Channel 68 Out -	34	Channel 68 Out +

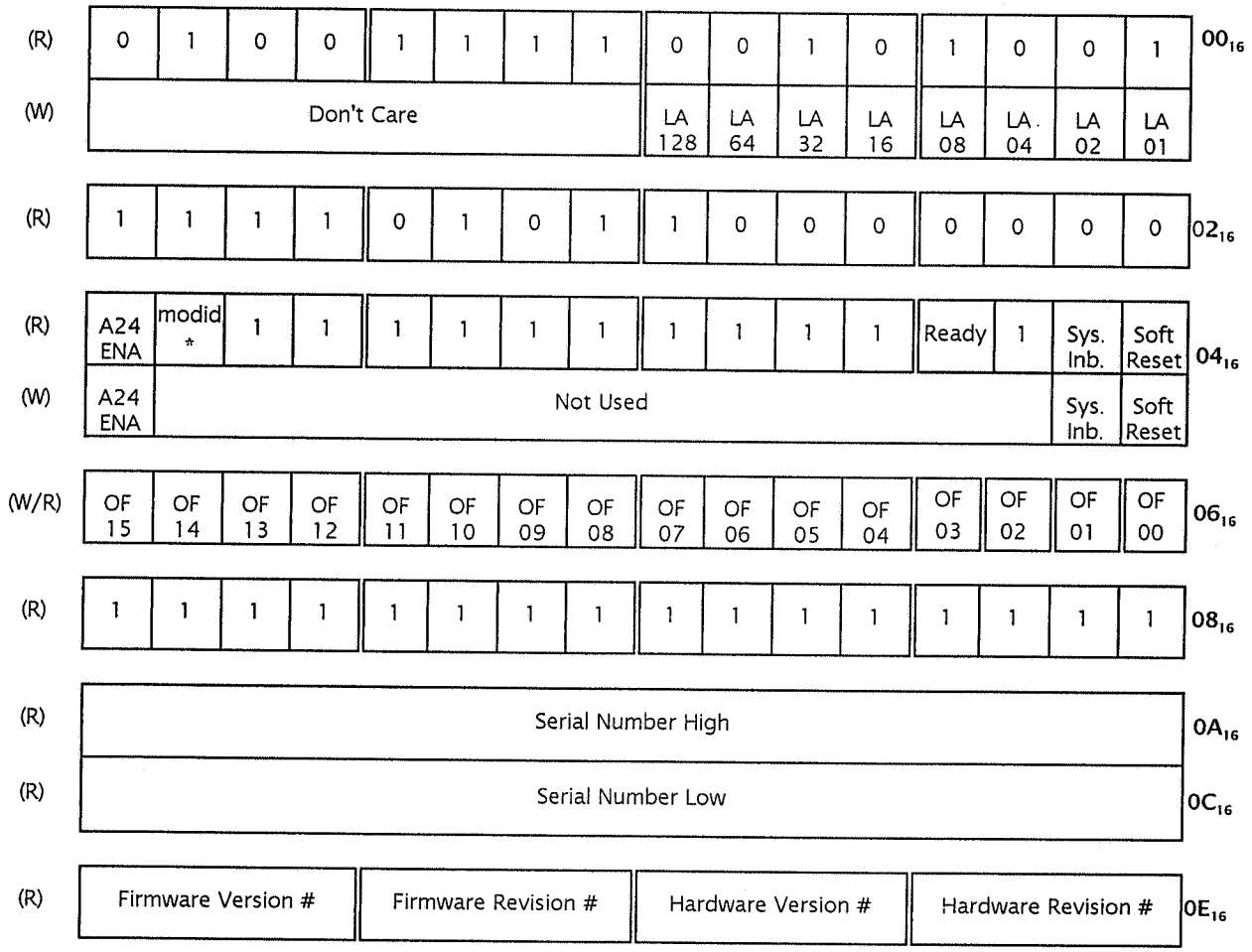
TABLE 3 - P3 50-Pin SCSI II Connector Pinout (Front View)

Pin #	Description	Pin #	Description
26	Channel 1 Out -	1	Channel 1 Out +
27	Channel 2 Out -	2	Channel 2 Out +
28	Channel 3 Out -	3	Channel 3 Out +
29	Channel 4 Out -	4	Channel 4 Out +
30	Channel 5 Out -	5	Channel 5 Out +
31	Channel 6 Out -	6	Channel 6 Out +
32	Channel 7 Out -	7	Channel 7 Out +
33	Channel 8 Out -	8	Channel 8 Out +
34	Channel 9 Out -	9	Channel 9 Out +
35	Channel 10 Out -	10	Channel 10 Out +
36	Channel 11 Out -	11	Channel 11 Out +
37	Channel 12 Out -	12	Channel 12 Out +
38	Channel 13 Out -	13	Channel 13 Out +
39	Channel 14 Out -	14	Channel 14 Out +
40	Channel 15 Out -	15	Channel 15 Out +
41	Channel 16 Out -	16	Channel 16 Out +
42	Channel 17 Out -	17	Channel 17 Out +
43	Channel 18 Out -	18	Channel 18 Out +
44	Channel 19 Out -	19	Channel 19 Out +
45	Channel 20 Out -	20	Channel 20 Out +
46	Channel 21 Out -	21	Channel 21 Out +
47	Channel 22 Out -	22	Channel 22 Out +
48	Channel 23 Out -	23	Channel 23 Out +
49	Channel 24 Out -	24	Channel 24 Out +
50	Channel 25 Out -	25	Channel 25 Out +

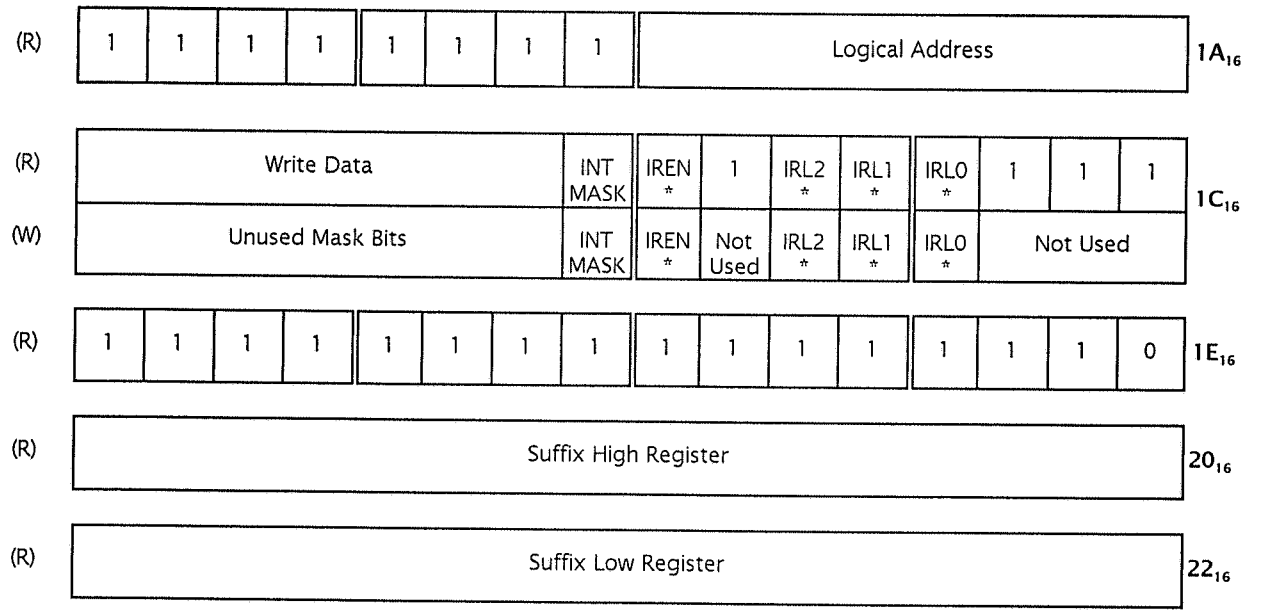
TABLE 4 - P4 50-Pin SCSI II Connector Pinout (Front View)

Pin #	Description	Pin #	Description
26	Channel 26 Out -	1	Channel 26 Out +
27	Channel 27 Out -	2	Channel 27 Out +
28	Channel 28 Out -	3	Channel 28 Out +
29	Channel 29 Out -	4	Channel 29 Out +
30	Channel 30 Out -	5	Channel 30 Out +
31	Channel 31 Out -	6	Channel 31 Out +
32	Channel 32 Out -	7	Channel 32 Out +
33	Channel 33 Out -	8	Channel 33 Out +
34	Channel 34 Out -	9	Channel 34 Out +
35	Channel 35 Out -	10	Channel 35 Out +
36	Channel 36 Out -	11	Channel 36 Out +
37	Channel 37 Out -	12	Channel 37 Out +
38	Channel 38 Out -	13	Channel 38 Out +
39	Channel 39 Out -	14	Channel 39 Out +
40	Channel 40 Out -	15	Channel 40 Out +
41	Channel 41 Out -	16	Channel 41 Out +
42	Channel 42 Out -	17	Channel 42 Out +
43	Channel 43 Out -	18	Channel 43 Out +
44	Channel 44 Out -	19	Channel 44 Out +
45	Channel 45 Out -	20	Channel 45 Out +
46	Channel 46 Out -	21	Channel 46 Out +
47	Channel 47 Out -	22	Channel 47 Out +
48	Channel 48 Out -	23	Channel 48 Out +
49	Channel 49 Out -	24	Channel 49 Out +
50	Channel 50 Out -	25	Channel 50 Out +

V580 Configuration Registers, A16 Space



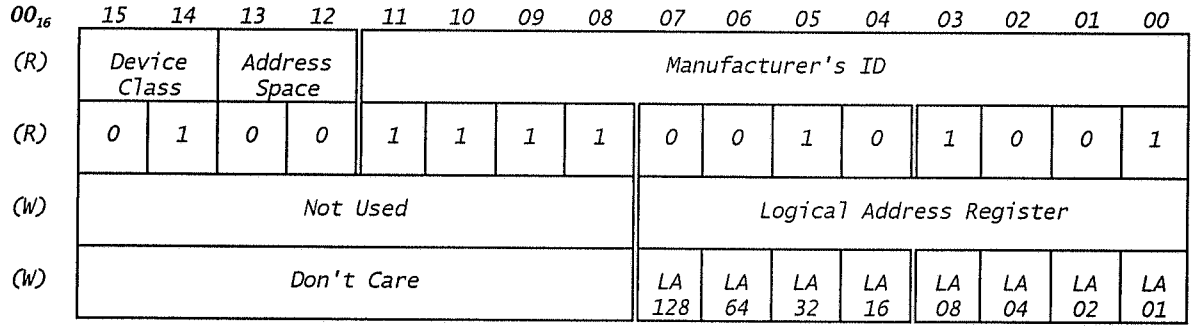
Offsets 10_{16} - 18_{16} Reserved



User Defined Registers 24_{16} - $3E_{16}$

V580 Configuration Registers, A16 Space

ID/Logical Address Register

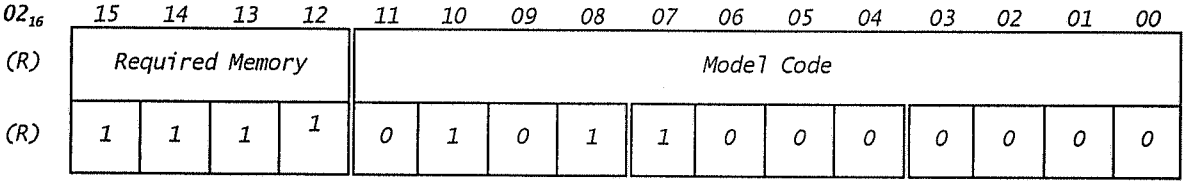


On READ transactions the V580 returns $4F29_{16}$.

Bit(s)	Mnemonic	Meaning
15,14	Device Class	This is an Extended Register-Based Device.
13,12	Address Space	This module requires the use of A16/A24 address space.
11-00	Manufacture's ID	3881 ($F29_{16}$) for KineticSytems.

For WRITE transactions, bits fifteen through eight are not used. These bits may be written with any data pattern. In Dynamically Configured systems (and the Logical Address switches were set to a value of 255), bits seven through zero are written with the Logical Address value.

Device Type Register



This READ ONLY register returns $E580_{16}$.

Bit(s)	Mnemonic	Meaning
15-12	Required Memory	The V580 requires 256 bytes of additional memory space.
11-00	Model Code	Identifies this device as a V580 (580_{16}).

Status/Control Register

04_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	A24 ENA	MODID *	1	1	1	1	1	1	1	1	1	1	Ready	1	Sys. Inb.	Soft Reset
(W)	A24 ENA	Not Used													Sys. Inb.	Soft Reset

The bit assignments for the Status/Control register are defined as follows:

Bit(s)	Mnemonic	Meaning
15	A24 ENA	This bit is written with a "1" to enable A24 addressing and reset to "0" to disable these registers. **This bit must be set to allow access to the module's Operational Registers**. Reads of this bit indicate its current state. This bit is reset to "0" by the assertion of SYSRESET*.
14	MODID*	This read only bit is set to a "1" if the module is <u>not</u> selected with the MODID line on P2. A "0" in this bit location indicates the device is selected via a high state on its P2 MODID line.
13-04	Not Used	These bits are not used and are read as "1s".
03	Ready	A "1" in this bit indicates the successful completion of register initialization.
02	Not Used	This bit is not used and is read as "1".
01	Sys. Inb.	(Sysfail Inhibit) Writing a "1" to this bit disables the V580 from driving the SYSFAIL* line. Reads of this bit indicate its current state.
00	Soft Reset	Writing a "1" to this bit forces the device into the Soft Reset State. While in this state, the module will only allow access to its configuration Registers. **This bit must be cleared along with the Pass and Ready bits set before any access to the Operational Registers is allowed.**

Offset Register

06_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(W/R)	OF 15	OF 14	OF 13	OF 12	OF 11	OF 10	OF 09	OF 08	OF 07	OF 06	OF 05	OF 04	OF 03	OF 02	OF 01	OF 00
Address Mapping	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A09	A08

After SYSRESET* all bits are reset to "0". Otherwise, a read or write defines the base address of the module's A24 registers. As shown above bits 15-00 map directly onto VME address lines A23-A08. For example, if bits OF15-0F00 contain 2430_{16} the base address for the module's Operational Registers becomes 243000_{16}.

Attribute Register

08_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	Reserved													IR*	IH*	IC*
(R)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

This read only register returns $FFFF_{16}$ on READ transactions. Write transactions to this register have no effect and its usage is reserved for future definition.

Bit(s)	Mnemonic	Meaning
15-03	Reserved	These bits are read as "1s" and reserved for future definition.
02	IR*	This bit is read as a "1" to signify that the V580 is not capable of generating interrupts.
01	IH*	This bit is read as a "1" and indicates the V580 is not capable of Interrupt Handler Control.
00	IS*	This bit is set to "1" to indicate the V580 has no Interrupt Status Reporting capability.

The following two READ ONLY registers indicate the serial number of the module. Each module is given a unique serial number. The serial number is represented by a 32-bit unsigned integer. The least significant bits (LSBs) reside in the Serial Number Low register while the most significant bits (MSBs) are in the Serial Number High register. Writing to these registers will have no effect and its use is reserved. For example, assume the module's serial number is 10064_{16} (65636). A read of the Serial Number High register returns 0001_{16} (1 => 1 * 65536); and the Serial Number Low register returns 0064_{16} (100). This example is illustrated below.

Serial Number High

$0A_{16}$	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	Serial Number High															
Example	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Serial Number Low

$0C_{16}$	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	Serial Number Low															
Example	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

Version Number Register

$0E_{16}$	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	Firmware Version #				Firmware Revision #				Hardware Version #				Hardware Revision #			
Example	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1

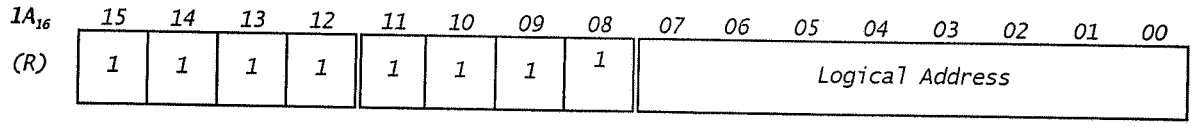
This READ ONLY register indicates the hardware and firmware revision number of the module. A write to this register has no effect on its contents. The fields of this register are explained as follows:

Bits	Mnemonic	Meaning
15-12	Firmware Version #	Firmware Version Number
11-08	Firmware Revision #	Firmware Revision Number
07-04	Hardware Version #	Hardware Version Number
03-00	Hardware Revision #	Hardware Revision Number

The combination of Firmware Version Number and Firmware Revision Number indicate the module's firmware version level. These two fields contain two four bit integers and are joined to form the level. An example of data returned for a firmware version number of 1.0 is shown above.

The combination of Hardware Version Number and Hardware Revision Number indicate the module's hardware version level. These two fields contain two four bit integers and are joined to form the level. An example of data returned for a hardware version number of 1.9 is shown above.

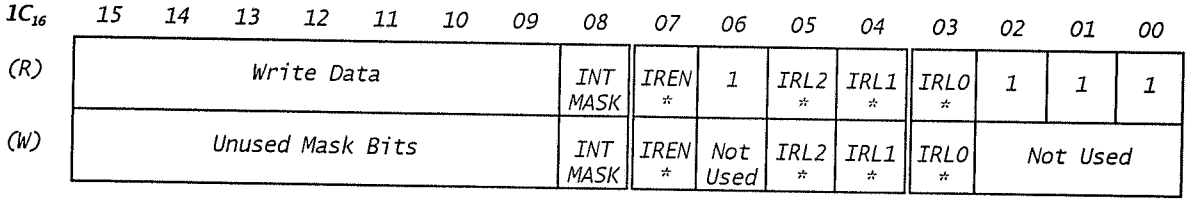
Interrupt Status Register



This READ ONLY register is defined as follows:

Bit(s)	Mnemonic	Meaning
15-08	Not Used	These bits are not used and read as "1s".
07-00	Logical Address	the Logical Address of the V580.

Interrupt Control Register



The V580 has no interrupt capability.

Bits(s)	Mnemonic	Meaning
15-09	Not Used	These bits are reserved for use as interrupt mask bits. Although their function is currently unimplemented, they should be written with "1s" to prevent incompatibility with future enhancements.
08	INT MASK	Not used on the V580
07	IREN*	Not used on the V580.
06	Not Used	This bit is reserved for use during interrupt handling. Since the V580 is not capable of interrupt handling, this bit should always be written with a "1".

05-03 IRL2*-IRL0* This 3-bit field selects the VXIbus interrupt line associated with the interrupt according to the following table:

Bit			Interrupt Request Line
IRL2* (D05)	IRL1* (D04)	IRL0* (D03)	
0	0	0	IRQ7
0	0	1	IRQ6
0	1	0	IRQ5
0	1	1	IRQ4
1	0	0	IRQ3
1	0	1	IRQ2
1	1	0	IRQ1
1	1	1	Disconnected

02-00 Not Used

These bits are reserved for selecting an interrupt handler line. Since the V580 does not have interrupt handler capabilities, these bits should always be written with "1s".

All bits in this register are set to "1" on the assertion of SYSRESET* or if the SOFT RESET bit in the Status/Control register is written with a "1".

Subclass Register

$1E_{16}$	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	VXI E.D.	Extended Register Based Device														
(R)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

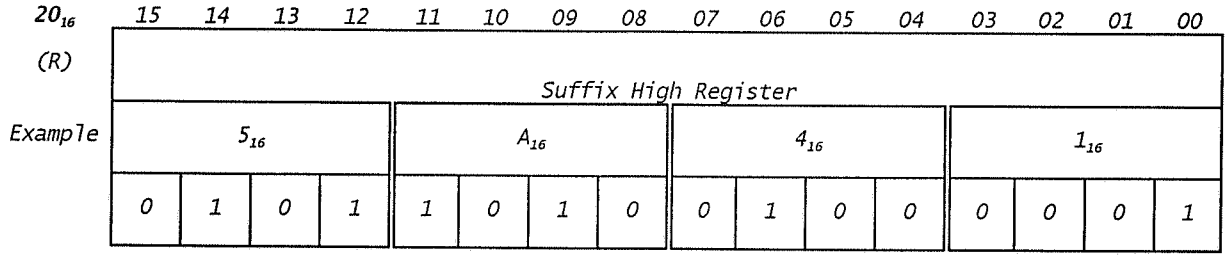
Reads of this register return $FFFE_{16}$. Writes to this register have no effect.
The read contents are defined as follows:

Bit(s)	Meaning
15	VXI E.D. = 1 indicates that the V580 is a VXIbus defined Extended Device.
14-00	Extended Register Based Device = $7FFE_{16}$ indicate that this is an Extended Register Based Device.

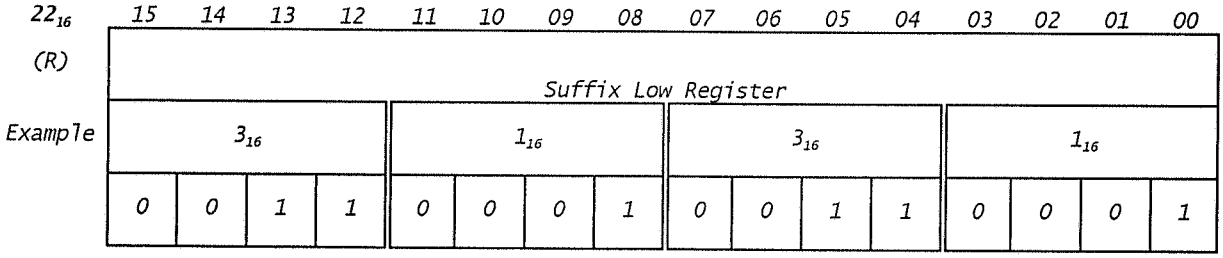
The following two registers are KineticSystems defined and hold the module's suffix. The suffix determines the particular option of the module. This information can be used remotely establish available channel count, filtering options, etc. of the module. For further information on each option, refer to the Ordering Information section of this manual.

The module's suffix is always composed of four ASCII characters. The Suffix High register contains the first two characters; while, the last two characters are in the Suffix Low register. For instance, assume the module is a model V580-ZA11. The module's suffix is "ZA11". Converting this to ASCII yields $5A413131_{16}$. This value is divided among the upper and lower registers as shown below.

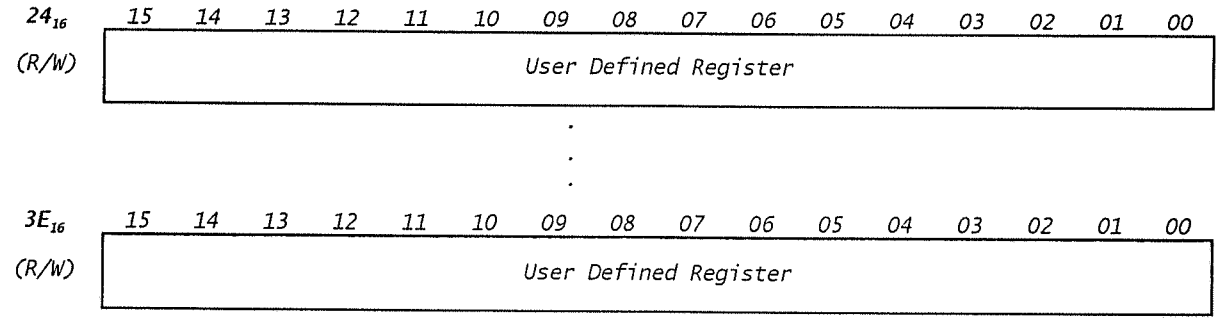
Suffix High Register



Suffix Low Register



Offsets 24_{16} through $3E_{16}$ are READ/WRITE registers and may be used to store user defined data. These registers are contained in non-volatile EEPROM. A typical use for these registers would be to hold calibration information such as date, time, etc.



V580 Operational Registers, A24 Space

Path Grounding Configuration Register

0000_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)															Path B	Path A

Instrument Path Connection Register

0002_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)											Inst B	Inst B			Inst A	Inst A

Channel Group 1 Register (P3 Connector Channels 1 through 16)

0010_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 16	CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1

Channel Group 2 Register (P3 Connector Channels 17 through 32)

0012_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 32	CH 31	CH 30	CH 29	CH 28	CH 27	CH 26	CH 25	CH 24	CH 23	CH 22	CH 21	CH 20	CH 19	CH 18	CH 17

Channel Group 3 Register (P3 Connector Channels 33 through 34)

0014_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)															CH 34	CH 33

Channel Group 4 Register (P4 Connector Channels 35 through 50)

0026_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 50	CH 49	CH 48	CH 47	CH 46	CH 45	CH 44	CH 43	CH 42	CH 41	CH 40	CH 39	CH 38	CH 37	CH 36	CH 35

Channel Group 5 Register (P4 Connector Channels 51 through 66)

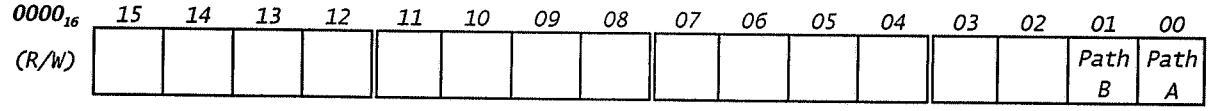
0028_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 66	CH 65	CH 64	CH 63	CH 62	CH 61	CH 60	CH 59	CH 58	CH 57	CH 56	CH 55	CH 54	CH 53	CH 52	CH 51

Channel Group 6 Register (P4 Connector Channels 67 through 68)

$002A_{16}$	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)															CH 68	CH 67

V580 Operational Registers, A24 Space

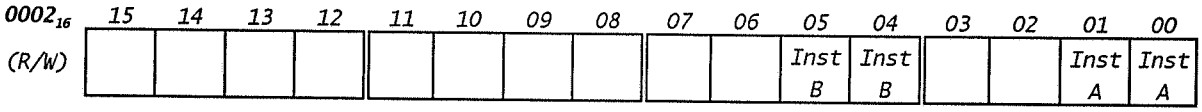
Path Grounding Configuration Register



The Path Grounding Configuration Register is used to monitor and control which internal signal path on the V580 is connected to ground.

Bit(s)	Meaning	
15-02	Not used.	
01	Path B Control	A "1" in this bit connects path B internally to ground.
00	Path A Control	A "1" in this bit connects path A internally to ground.

Instrument Path Connection Register



The Instrument Path Connection Register is used to monitor and control which internal signal path; path A (P3 connector containing channels 1 through 34) or path B (P4 connector containing channels 35 through 68) on the V580 is connected to an external instrument (DVM, Function Generator, ect) via the 2-Contact LEMO's labeled "Instr A " and "Instr B".

Bit(s)	Meaning	
15-06	Not used.	
05-04	Instrument B Control	"00" No connection to either internal path. "01 " Connect to internal path A (P3 connector) "10" Connect to internal path B (P4 connector) "11" Connect to both path A and B
03-02	Not used	
01-00	Instrument A Control	"00" No connection to either internal path. "01 " Connect to internal path A (P3 connector) "10" Connect to internal path B (P4 connector) "11" Connect to both path A and B.

The following six registers contain the module's Channel Groups. They are used to select any channel on the 50 or 68 position connector to connect to the internal signal path. The internal signal path may then be connected to ground or to either of the 2-Contact LEMOs "Instr A" or "Instr B".

Note: The registers show the channel numbers for a ZB11 option (68 position HD connectors) which gives the user 68 differential channels. The ZA11 option (50 position connectors) uses the same registers where channels 1 through 16 are in register 1 and channel 17 through 25 are in register 2. Channels 26 through 41 are in registers 4 and channels 42 through 50 are in register 5.

Channel Group 1 Register (P3 Connector Channels 1 through 16)

0010_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 16	CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1

Channel Group 2 Register (P3 Connector Channels 17 through 32)

0012_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 32	CH 31	CH 30	CH 29	CH 28	CH 27	CH 26	CH 25	CH 24	CH 23	CH 22	CH 21	CH 20	CH 19	CH 18	CH 17

Channel Group 3 Register (P3 Connector Channels 33 through 34)

0014_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)															CH 34	CH 33

For channel groups 1, 2, and 3, a "1" indicates that the channel is enabled and connected to path A.

Channel Group 4 Register (P4 Connector Channels 35 through 50)

0026_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 50	CH 49	CH 48	CH 47	CH 46	CH 45	CH 44	CH 43	CH 42	CH 41	CH 40	CH 39	CH 38	CH 37	CH 36	CH 35

Channel Group 5 Register (P4 Connector Channels 51 through 66)

0028_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 66	CH 65	CH 64	CH 63	CH 62	CH 61	CH 60	CH 59	CH 58	CH 57	CH 56	CH 55	CH 54	CH 53	CH 52	CH 51

Channel Group 6 Register (P4 Connector Channels 67 through 68)

$002A_{16}$	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)															CH 68	CH 67

For channel groups 4, 5, and 6, a "1" indicates that the channel is enabled and connected to path B.

APPENDIX A

Lets take a quick look at a configuration of a V580-ZA11 as two 1x25 muxs. The input signals would be connected to the high density connectors P3 and P4. The P3 connector contains channels 1 through 25 routed internally to path A. These channels are controlled by Channel Group Registers 1 and 2. The P4 connector contains the second set of 25 channels which will be referred to as channels 26 through 50 routed internally to path B. These channels are controlled by Channel Group Registers 4 and 5. The output of the two muxs may then be routed to either of four choices, the 2-contact LEMO "Instr A", 2-contact LEMO "Instr B", internal ground connection, or no connection.

Lets look at a more in-depth example of using a V580-ZB11 as two 1x34 muxs with "Instr A" connected to a DVM and to path A which is channels 1 through 34 on the P3 connector. And "Instr B" connected to a variable voltage source and to path B which is channels 35 through 68 on the P4 connector. Then configuring in A24 space the following operational registers:

Path Grounding Configuration Register

0000_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)															Path B	Path A

The Path Grounding Configuration Register is used to monitor and control which internal signal path on the V580 is connected to ground.

Bit(s)	Meaning	
15-02	Not used.	
01	Path B Control	A "1" in this bit connects path B internally to ground.
00	Path A Control	A "1" in this bit connects path A internally to ground.

In this example neither path should be grounded; therefore data of $00 should be written to the path grounding register.

Caution: When using the V580 with a source connected to one of the 2-contact LEMOs and configuring one of the paths for internal ground connection there exists the potential for connecting a grounded path to the connected source.

Instrument Path Connection Register

0002_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)											Inst B	Inst B			Inst A	Inst A

The Instrument Path Connection Register is used to monitor and control which internal signal path; path A (P3 connector containing channels 1 through 34) or path B (P4 connector containing channels 35 through 68) on the V580 is connected to an external instrument (DVM, Function Generator, ect) via the 2-Contact LEMOs labeled "Instr A" and "Instr B".

Bit(s)	Meaning	
15-06	Not used.	
05-04	Instrument B Control	"00" No connection to either internal path. "01 " Connect to internal path A (P3 connector) "10" Connect to internal path B (P4 connector) "11" Connect to both path A and B
03-02	Not used	
01-00	Instrument A Control	"00" No connection to either internal path. "01 " Connect to internal path A (P3 connector) "10" Connect to internal path B (P4 connector) "11" Connect to both path A and B.

In this example "Instr A " is connected to path A and "Instr B " is connected to path B; therefore data $21 should be written to the instrument path register.

The last step is to configure the channel registers with the desired channel number to connect to path A which in this case is a DVM and path B which is a voltage source. Lets look at a channel 30 (P3 connector) which we will connect to the DVM and channel 38 (P4 connector) which we will connect to the voltage source.

Channel Group 2 Register (P3 Connector Channels 17 through 32)

0012_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 32	CH 31	CH 30	CH 29	CH 28	CH 27	CH 26	CH 25	CH 24	CH 23	CH 22	CH 21	CH 20	CH 19	CH 18	CH 17

Channel Group 4 Register (P4 Connector Channels 35 through 50)

0026_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 50	CH 49	CH 48	CH 47	CH 46	CH 45	CH 44	CH 43	CH 42	CH 41	CH 40	CH 39	CH 38	CH 37	CH 36	CH 35

Write data $2000 to Channel Group 2 Register to connect channel 30 to "Instr A".
Write data $80 to Channel Group 4 Register to connect channel 38 to "Instr B".

APPENDIX B

```
/*****************************************************************/
/*                                                               */
/*                                                               */
/*  Program Name: module1.c                                      */
/*                                                               */
/*                                                               */
/*  This program is a C/C++ code example for the National        */
/*  Instruments controller.  This program will set certain       */
/*  relays specified by the user.                                */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
#include<stdlib.h>
#include<stdio.h>
#include<time.h>

typedef unsigned short uint16;
typedef unsigned long uint32;
typedef short int16;

uint16 const          A24SuperData = 0x6;

void set_channel(int,int,int,int,int,int);

void main()
{
   uint32 address, A24_V580, serial_num;
   int16 error, channel;
   uint16 data,offset,V580,access,data2;
   int i,P3,P4,InstrA,InstrB,option,ground;
   char keybuf;

   system("resman -o");
   InitVXIlibrary();
   system("cls");
   access = A24SuperData;
   printf("\n\n              V580 Relay Multiplexer\n\n");

   /* Getting the Logical Address of the V580 */
   error = FindDevLA("",0xF29,0x580,-1,-1,-1,-1,&V580);
   if (error != 0)
   {
    printf(" Error: FindDevLA() returned %d\n", error);
    CloseVXIlibrary();
    exit(1);
   }

   /* Getting the A24 base address for the V580 */
   error = GetDevInfo(V580, 12, &A24_V580);
   if (error != 0)
```

```
{
 printf(" Error: GetDevInfo returned %d\n", error);
 CloseVXIlibrary();
 exit(1);
}

offset = 0x20;
error = VXIinReg(V580,offset,&data);
if (error != 0)
{
 printf(" Error: VXIinReg returned %d\n",error);
 CloseVXIlibrary();
 exit(1);
}
data &= 0xFF;

offset = 0x22;
error = VXIinReg(V580,offset,&data2);
if (error != 0)
{
 printf(" Error: VXIinReg returned %d\n",error);
 CloseVXIlibrary();
 exit(1);
}
data2 >> 8;
data2 &= 0xFF;

if (data == 'A' && data2 == '1') {
 printf(" V580-ZA11 found \n");
 option = 25;
}
else if (data == 'B' && data2 =='1'){
 printf(" V580-ZB11 found \n");
 option = 34;
}
else {
 printf(" Error: V580 option not found \n");
 printf("       Reread Suffix High Register\n");
 printf("       Expect ASCII 'ZA'(0x5A41) or 'ZB'(0x5A42)\n");
 offset = 0x20;
 error = VXIinReg(V580,offset,&data);
 if (error != 0)
 {
    printf(" Error: VXIinReg returned %d\n",error);
    CloseVXIlibrary();
    exit(1);
 }

 printf("       Reread Suffix Low Register\n");
 printf("       Expect ASCII '11'(0x3131) or '21'(0x3231)\n");
 offset = 0x22;
 error = VXIinReg(V580,offset,&data2);
 if (error != 0)
```

```
    {
       printf(" Error: VXIinReg returned %d\n",error);
       CloseVXIlibrary();
       exit(1);
    }
  }

  offset = 0xA;/* look at Serial Number Registers      */
  error = VXIinReg(V580,offset,&data);
  if (error !=0 )
  {
   printf(" Error: VXIinReg returned %d\n",error);
   CloseVXIlibrary();
   exit(1);
  }

  offset = 0xC;
  error = VXIinReg(V580,offset,&data2);
  if (error !=0 )
  {
   printf(" Error: VXIinReg returned %d\n",error);
   CloseVXIlibrary();
   exit(1);
  }
  serial_num = ((uint32)data << 16) | (uint32)data2;
  printf(" Serial number = %ld \n\n",serial_num);
  repeat:
  /* Reset all Registers */
  data = 0;
  for (i=0;i<=21;i++)
  {
   address = A24_V580 + (i*2);
   error = VXIout(access,address,2,data);
   if (error != 0)
   {
      printf(" Error: VXIout() returned %d\n", error);
      CloseVXIlibrary();
      exit(1);
   }
  }
  printf(" Options: 0     No channel \n");
  printf("          1-%d  Channel number \n",(option));
  printf("          -1    Set all channels \n");

  printf("\n\n Enter the option to set P3: ");
  scanf("%d", &P3);
  printf(" Enter the option to set P4: ");
  scanf("%d", &P4);

  printf("\n\n Options: 0     Not connected \n");
  printf("          1     Connect to Path A \n");
  printf("          2     Connect to Path B \n");
  printf("          3     Connect to Path A and Path B \n");
```

```
    printf("\n\n Enter the option to set Instr A: ");
    scanf("%d", &InstrA);
    printf(" Enter the option to set Instr B: ");
    scanf("%d", &InstrB);

    printf("\n\n Options: 0    No path grounded \n");
    printf("          1    Ground Path A \n");
    printf("          2    Ground Path B \n");
    printf("          3    Ground Path A and Path B \n");

    printf("\n\n Enter the grounding option: ");
    scanf("%d",&ground);

    set_channel(P3, P4, InstrA, InstrB, ground, option);
    printf("\n Do you wish to continue [y/n]? ");
    while (!kbhit());           /* wait for keyboard to send char */
    keybuf = getch();
    keybuf = (char)(toupper( (int)keybuf ));
    if (keybuf == 'N' || keybuf == 'n')
    {
     exit(1);
    }
    if (keybuf == 'Y' || keybuf == 'y')
    {
     system("cls");
     printf("\n\n");
     goto repeat;
    }

    CloseVXIlibrary();
    return;
}

void set_channel(int P3,int P4,int InstrA,int InstrB,int ground,int option)
{
    uint16 offset,pin,data,V580,access;
    uint32 address,A24_V580;
    int16 error;
    int i,k;

    access = A24SuperData;
    /* Getting the Logic Address of V580 */
    error = FindDevLA("",0xF29,0x580,-1,-1,-1,&V580);
    if (error != 0)
    {
     printf(" Error: FindDevLA() returned %d\n", error);
     CloseVXIlibrary();
     exit(1);
    }
    /* Getting the A24 base address for the V580 */
    error = GetDevInfo(V580, 12, &A24_V580);
    if (error != 0)
    {
```

```
    printf(" Error: GetDevInfo returned %d\n", error);
    CloseVXIlibrary();
    exit(1);
  }

  if(option == 25)
  {
    pin = 25;
    k=1;
  }
  if(option == 34)
  {
    pin = 32;
    k=2;
  }

  /* Writes data to Instr A and B */
  address = A24_V580 + 0x2;
  InstrB *= 0x10;
  data = InstrA + InstrB;
  error = VXIout(access,address,2,data);
  if (error != 0)
  {
    printf(" Error: VXIout() returned %d \n",error);
    CloseVXIlibrary();
    exit(1);
  }

/***** Writes data to P3 or P4 ************************************/

  if (P3 == -1)
  {
    data = 0xFFFF;
    for(i=0; i<=k; i++)
    {
      offset = 0x10 + (i*2);
      address = A24_V580 + offset;
      error = VXIout(access,address,2,data);
      if (error !=0)
      {
        printf(" Error: VXIout() returned %d\n",error);
        CloseVXIlibrary();
        exit(1);
      }
    }
  }

  if (P4 == -1)
  {
    data = 0xFFFF;
    for(i=0; i<=k; i++)
    {
      offset = 0x26 + (i*2);
```

```
        address = A24_V580 + offset;
        error = VXIout(access,address,2,data);
        if (error !=0)
        {
            printf(" Error: VXIout() returned %d\n",error);
            CloseVXIlibrary();
            exit(1);
        }
    }
}

if (P3 >=1 && P3 <=16)
{
  data = 1 << (P3 - 1);
  address = A24_V580 + 0x10;
  error = VXIout(access,address,2,data);
  if (error != 0)
  {
     printf(" Error VXIout() returned %d\n",error);
     CloseVXIlibrary();
     exit(1);
  }
}

if (P4 >=1 && P4 <=16)
{
  data = 1 << (P4 - 1);
  address = A24_V580 + 0x26;
  error = VXIout(access,address,2,data);
  if (error != 0)
  {
     printf(" Error VXIout() returned %d\n",error);
     CloseVXIlibrary();
     exit(1);
  }
}

if (P3 >=17 && P3 <= pin)
{
  data = 1 << (P3 - 17);
  address = A24_V580 + 0x12;
  error = VXIout(access,address,2,data);
  if (error != 0)
  {
     printf(" Error VXIout() returned %d\n",error);
     CloseVXIlibrary();
     exit(1);
  }
}

if (P4 >=17 && P4 <= pin)
{
  data = 1 << (P4 - 17);
```

```
    address = A24_V580 + 0x28;
    error = VXIout(access,address,2,data);
    if (error != 0)
    {
       printf(" Error VXIout() returned %d\n",error);
       CloseVXIlibrary();
       exit(1);
    }
  }

  if (P3 == 33)
  {
    data = 1;
    address = A24_V580 + 0x14;
    error = VXIout(access,address,2,data);
    if (error != 0)
    {
       printf(" Error VXIout() returned %d\n",error);
       CloseVXIlibrary();
       exit(1);
    }
  }
  else if (P3 == 34)
  {
    data = 2;
    address = A24_V580 + 0x14;
    error = VXIout(access,address,2,data);
    if (error != 0)
    {
       printf(" Error VXIout() returned %d\n",error);
       CloseVXIlibrary();
       exit(1);
    }
  }

  if (P4 == 33)
  {
    data = 1;
    address = A24_V580 + 0x2A;
    error = VXIout(access,address,2,data);
    if (error != 0)
    {
       printf(" Error VXIout() returned %d\n",error);
       CloseVXIlibrary();
       exit(1);
    }
  }
  else if (P4 == 34)
  {
    data = 2;
    address = A24_V580 + 0x2A;
    error = VXIout(access,address,2,data);
    if (error != 0)
```

```
    {
      printf(" Error VXIout() returned %d\n",error);
      CloseVXIlibrary();
      exit(1);
    }
  }

  if (P3 == 0)
  {
   data = 0;
   for (i=0;i<=2;i++)
   {
      offset = 0x10 + (i*2);
      address = A24_V580 + offset;
      error = VXIout(access,address,2,data);
      if (error != 0)
      {
            printf(" Error: VXIout() returned %d\n", error);
            CloseVXIlibrary();
            exit(1);
      }
   }
  }

  if (P4 == 0)
  {
   data = 0;
   for (i=0;i<=2;i++)
   {
      offset = 0x26 + (i*2);
      address = A24_V580 + offset;
      error = VXIout(access,address,2,data);
      if (error != 0)
      {
            printf(" Error: VXIout() returned %d\n", error);
            CloseVXIlibrary();
            exit(1);
      }
   }
  }
/*****************************************************************/

  /* Writes data to ground Path A or Path B */
  address = A24_V580;
  error = VXIout(access,address,2,ground);
  if (error != 0)
  {
   printf(" Error: VXIout returned %d \n", error);
   CloseVXIlibrary();
   exit(1);
  }
  return;
}
```