Model V581

50 to 68 - Channel Calibration Relay Module

INSTRUCTION MANUAL

February, 1996

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

CONTENTS

Tables

Figures

Calibration Relay Module

Connect 1 or 2 calibration sources to up to 68 differential channels

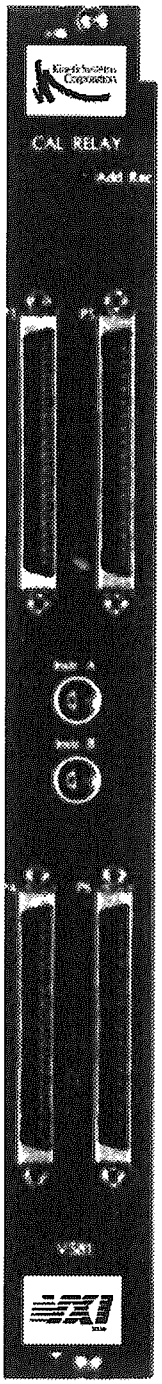
V581

Features

- **Up to 68 differential channels**
- **50P and 68P High Density connector options available**
- **Programmable on a per-channel basis**
- **Differential inputs and outputs**
- **Can route 1 of 2 instruments to any of 68 channels**

Typical Applications

- **Jet and rocket engine testing**
- **Data acquisition systems with per-channel calibration requirements**



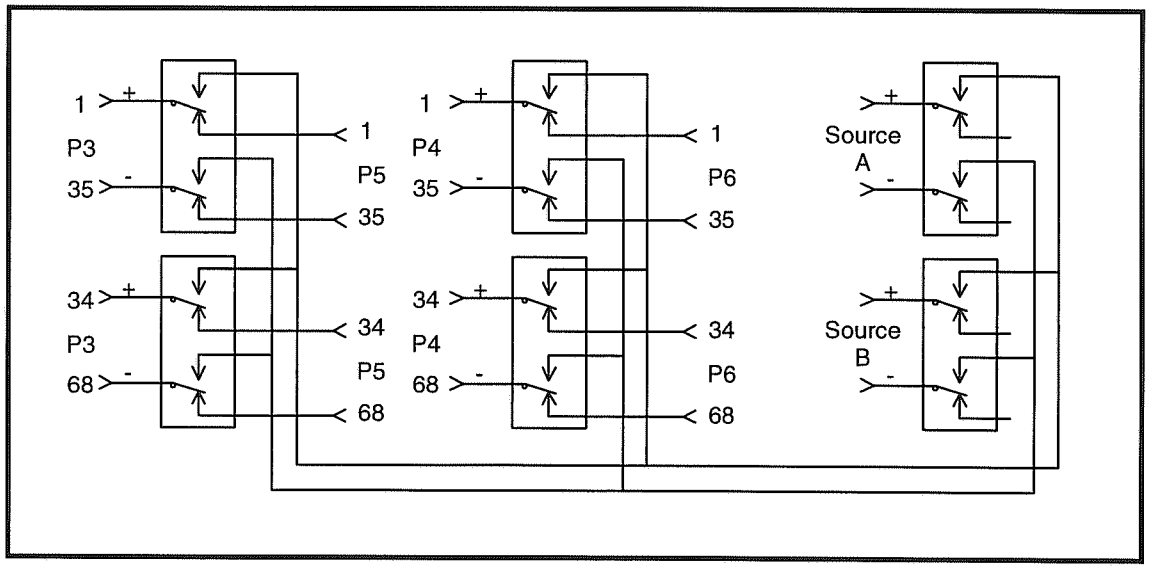
General Description (Product specifications and descriptions subject to change without notice.)

The V581 is a single-width, C-size, register-based, VXIbus module that contains 68 two-pole, double-throw relays for calibrating individual analog channels in data acquisition systems. Wiring from the transducers is brought in to the 50P or 68P High Density Connectors (P3 and P4) on the front panel. Two additional High Density Connectors (P5 and P6) provide a path to the data system. Any channel from the data system may be connected to either of two instruments for calibration, or to the field wiring, under program control. The calibration source is available through two balanced-pair, 2-contact LEMO connectors.

The V581 supports both static and dynamic configuration. It may be accessed using A24/A16, D16 data transfers.

Note that this module is intended for calibration of voltage input/output data acquisition channels (monitoring thermocouples, etc.) and should not be used between RTDs, strain gages, etc. and their associated bridge modules.

V581 Switching Diagram



Ordering Information

Model V581-ZA11 Calibration Relay Module, 50P High Density
Model V581-ZB11 Calibration Relay Module, 68P High Density

Related Products

Model 5857-Cxyz Cable—2-contact LEMO to Unterminated
Model 5857-Dxyz Cable—2-contact LEMO to 2-contact LEMO
Model 5857-Gxyz Cable—2-contact LEMO to BNC shielded
Model 5819-Bxyz Cable—50S High Density to Unterminated
Model 5819-Fxyz Cable—50S High Density to 50P High Density (V581-ZA11 to V765/V792)
Model 5819-Gxyz Cable—50S High Density to 50S High Density
Model 5868-Bxyz Cable—68S High Density to Unterminated
Model 5868-Dxyz Cable—68S High Density to 68P High Density (V581-ZB11 to V765/V792)
Model 5868-Exyz Cable—68S High Density to 68S High Density
Model 5868-Gxyz Cable—68S High Density to 50P High Density
Model 5868-Hxyz Cable—68S High Density to 50S High Density
Model V765-ZA11 Rack-mount Termination Panel
Model V792-ZA11 Rack-mount Isothermal Termination Panel

Model V581

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 V581 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 V581. 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 V581 will be dynamically configured by the resource manager. Logical address 255 is the factory default setting. (See Figure 1, page 1)

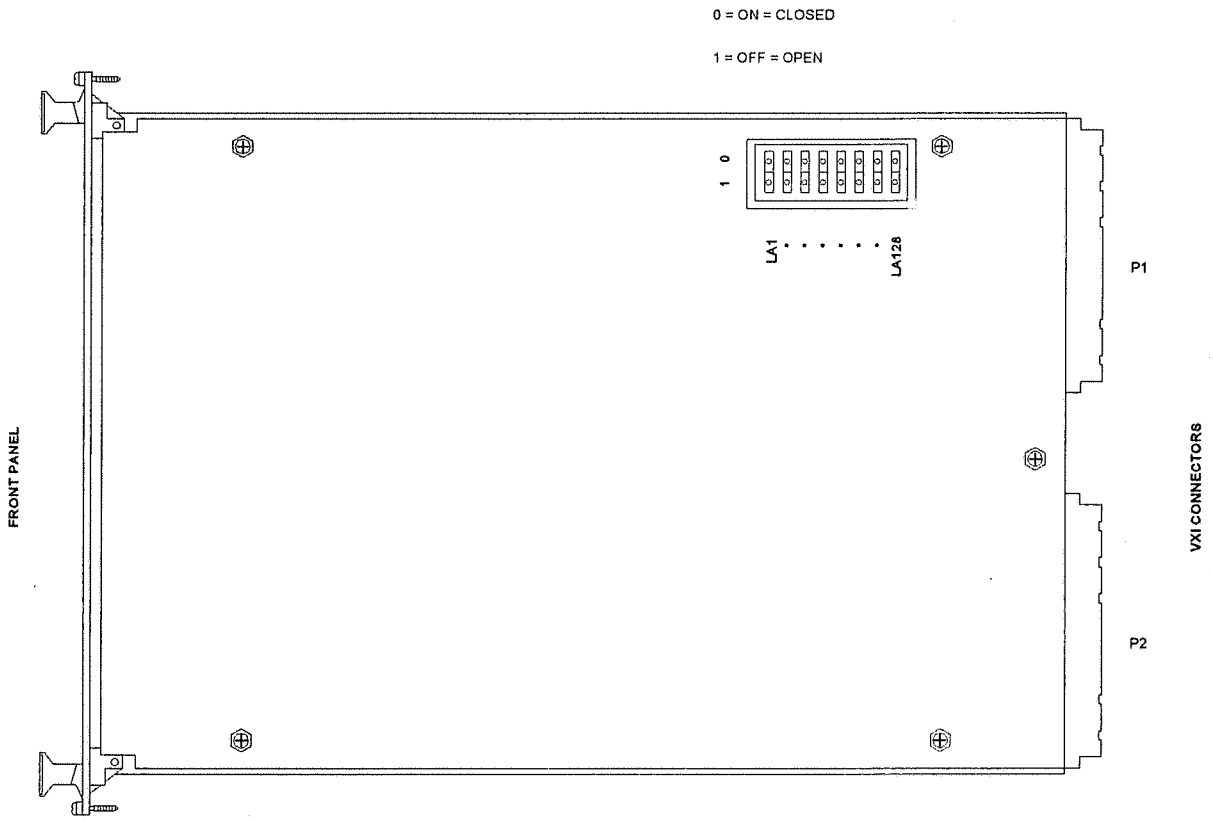


Figure 1 - V581 Switch Locations

Module Insertion

The V581 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 V581 is being accessed through VXI.

Connectors

External calibration signals are connected to the V581 via two 2-Contact LEMO's labeled "Instr A" and "Instr B". Either instrument may be configured to connect to any channel of the data system. Wiring from external transducers are connected to four high density connectors labeled P5 and P6. Connectors P3 and P4 provide a signal pathway from P5 and P6 to the data system. The V581-ZA11 option has four 50 pin, high density SCSI II type connectors. Beginning at the top of the module, connector P5 receives the differential input pairs for channels 1 through 25 and then routes the signals back out through connector P3. Connector P6 receives channels 26 through 50 and then routes the signals back out through connector P4 passes. The V581-ZB11 option has two 68 pin, high density SCSI II type connectors. Beginning at the top of the module, connector P5 receives the differential input pairs for channels 1 through 34 and then routes the signals back out through connector P3. Connector P6 receives channels 35 through 68 and then routesthe signals back out through connector P4. See Figures 2 and 3 and Tables 1, 2, 3, and 4 for the precise pinout descriptions.

PROGRAMMING INFO

VXIbus Addressing

The V581 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 V581 modules.

Appendix A includes an example in using a V581-ZA11 and a V581 ZB11.

Appendix B includes a C code example using NI-VXI routines with V581. It includes codes which will use the configuration registers to help identify the specific module and access both configuration and operational registers.

Resetting the V581

The V581 can be put into soft reset by setting the Reset bit of the Status Control Register.

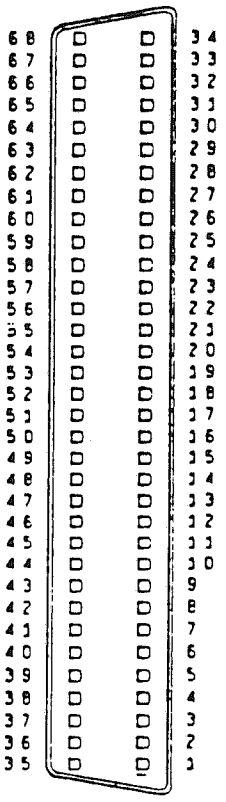


Figure 2 - 68 Pin SCSI II Connector

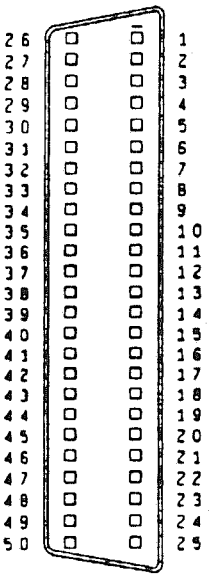


Figure 3 - 50 Pin SCSI II Connector

TABLE 1 - P3 and P5 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 and P6 68-Pin SCSI II Connector Pinout (Front View)

Pin #	Description	Pin #	Description
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 +

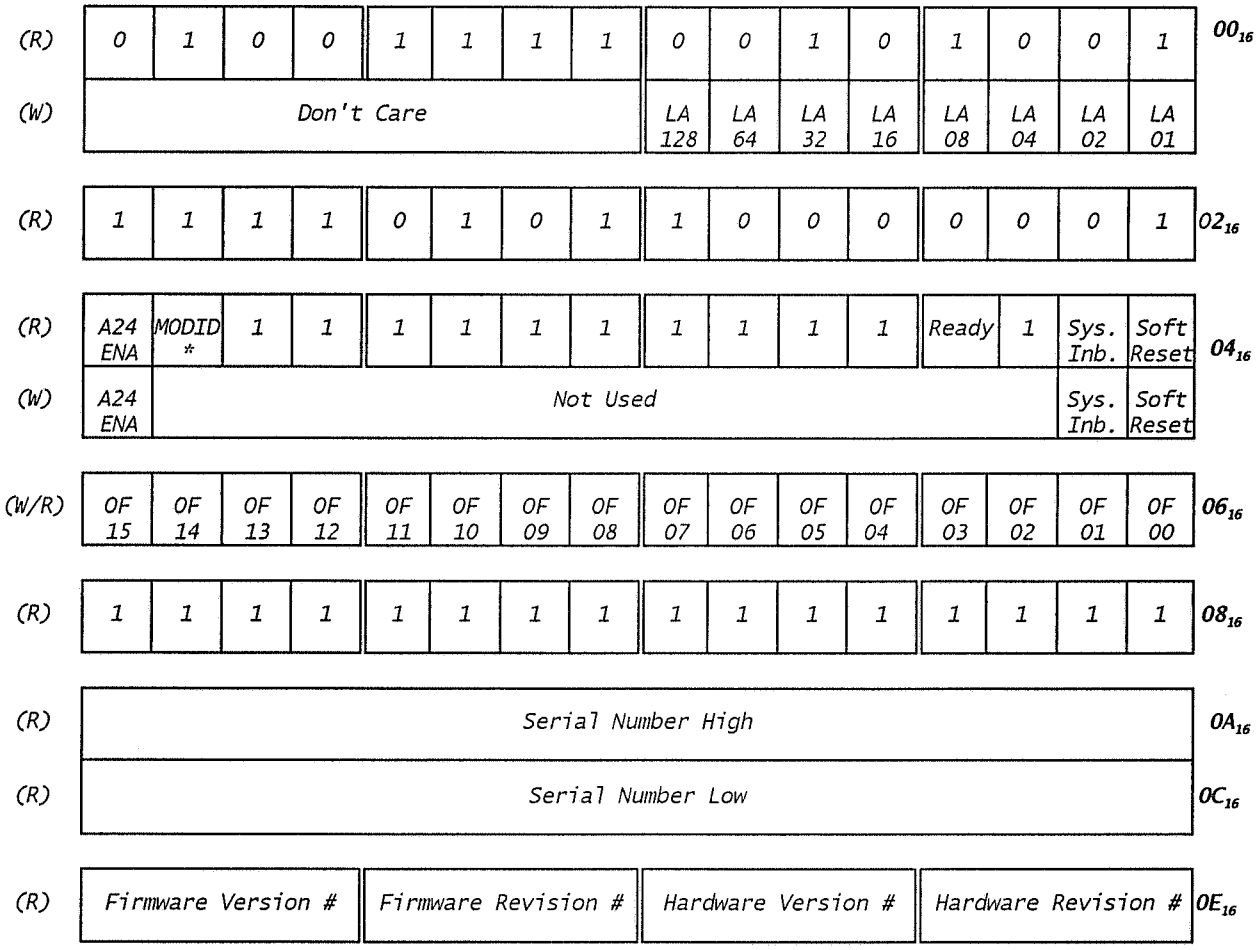
TABLE 3 - P3 and P5 50-Pin SCSI II Connector Pinout (Front View)

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 +

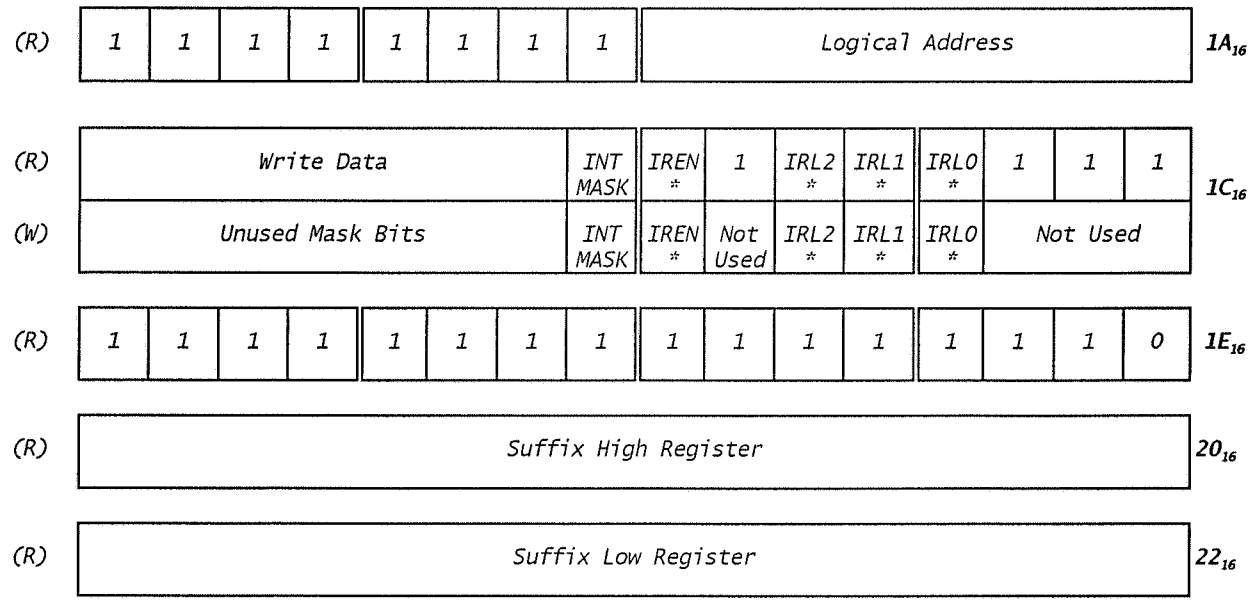
TABLE 4 - P4 and P6 50-Pin SCSI II Connector Pinout (Front View)

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 +

V581 Configuration Registers, A16 Space



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



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

V581 Configuration Registers, A16 Space

ID/Logical Address Register

00_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	Device Class		Address Space		Manufacturer's ID											
(R)	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	1
(W)	Not Used								Logical Address Register							
(W)	Don't Care								LA 128	LA 64	LA 32	LA 16	LA 08	LA 04	LA 02	LA 01

On READ transactions the V581 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

02_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	Required Memory				Model Code											
(R)	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	1

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

Bit(s)	Mnemonic	Meaning
15-12	Required Memory	The V581 requires 256 bytes of additional memory space.
11-00	Model Code	Identifies this device as a V581 (581_{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 not 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 V581 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	A15	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 V581 is not capable of generating interrupts.
01	IH*	This bit is read as a "1" and indicates the V581 is not capable of Interrupt Handler Control.
00	IS*	This bit is set to "1" to indicate the V581 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

$1A_{16}$	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	1	1	1	1	1	1	1	1	Logical Address							

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 V581.

Interrupt Control Register

$1C_{16}$	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R)	Write Data							INT MASK	IREN *	1	IRL2 *	IRL1 *	IRL0 *	1	1	1
(W)	Unused Mask Bits							INT MASK	IREN *	Not Used	IRL2 *	IRL1 *	IRL0 *	Not Used		

The V581 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 V581.
07	IREN*	Not used on the V581.
06	Not Used	This bit is reserved for use during interrupt handling. Since the V581 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 V581 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:

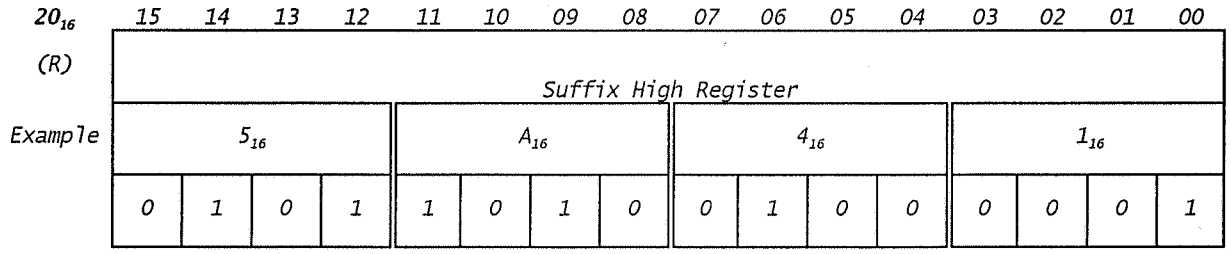
Bit(s)	Meaning
15	VXI E.D. = 1 indicates that the V581 is a VXIbus defined Extended Device.
14-00	Extended Register Based Device = $7FFE_{16}$ indicate that this is an Extended Register Based Device.

Model V581

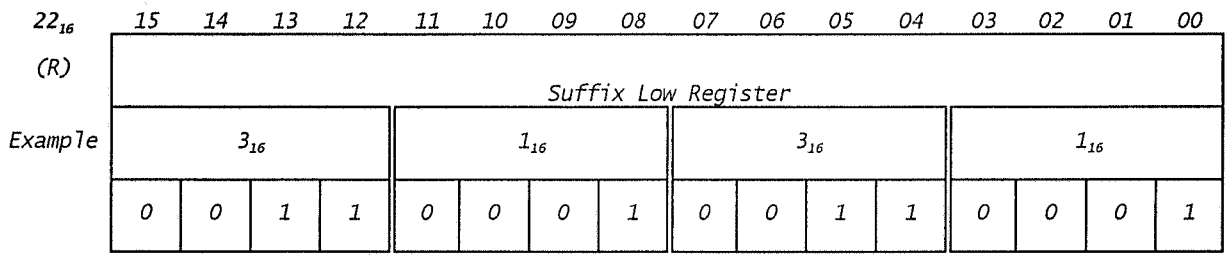
The following two registers are KineticSystems defined and hold the module's suffix. The suffix determines the particular option of the module. This information can be used remotely establish available channel count, filtering options, etc. of the module. For further information on each option, refer 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 V581-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.

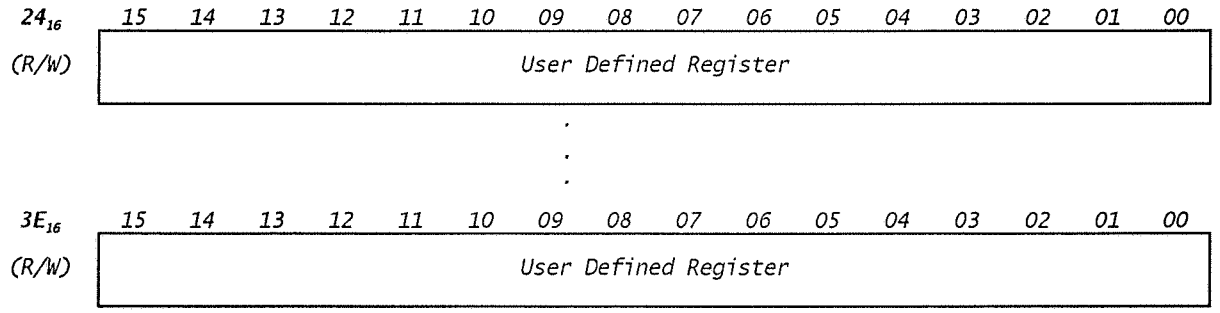
Suffix High Register



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.



V581 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)																Intr Path

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 A

Channel Group 1 Register (P5 to 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 (P5 to 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 (P5 to 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 (P6 to P4 Connector Channels 35 through 50)

0020_{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 (P6 to P4 Connector Channels 51 through 66)

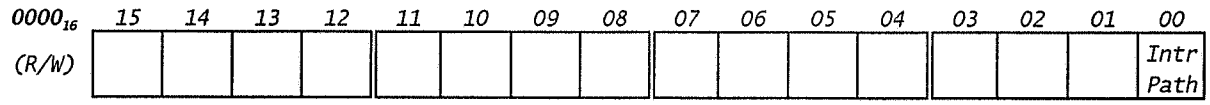
0022_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 66	CH 65	CH 64	CH 63	CH 62	CH 61	CH 60	CH 59	CH 58	CH 57	CH 56	CH 55	CH 54	CH 53	CH 52	CH 51

Channel Group 6 Register (P6 to P4 Connector Channels 67 through 68)

0024_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)															CH 68	CH 67

V581 Operational Registers, A24 Space

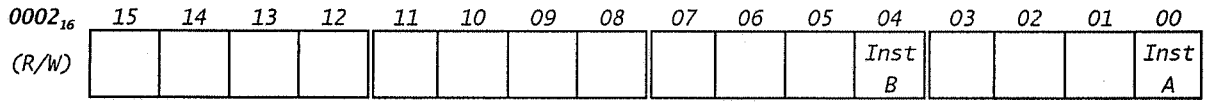
Path Grounding Configuration Register



The Path Grounding Configuration Register is used to monitor and control the grounding of the internal signal path on the V581.

Bit(s)	Meaning	
15-01	Not used.	
00	Internal Path	A "1" in this bit connects the internal signal path to ground. This in effect acts as a third calibration source which is connected to ground.

Instrument Path Connection Register



The Instrument Path Connection Register is used to monitor and control which instrument; "Instr A" or "Instr B" is to be connected to the internal signal path.

Note 1: Both instruments should not be connected to the internal signal path at the same time.

Note 2: If the internal signal path has been configured to connect to ground then do not connect either of the instruments to the pathway until ground has been removed by writing to the Path Grounding Register.

Bit(s)	Meaning	
15-05	Not used.	
04	Instrument B Control	"0" No connection to internal signal path. "1" Connect to internal signal path.
03-01	Not used	
00	Instrument A Control	"0" No connection to internal signal path. "1" Connect to internal signal path.

Model V581

The following six registers contain the module's Channel Groups. They are used to select any channel on the 50 or 68 position HD connector to connect to either of the 2-Contact LEMO's "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 25 are in Channel Group Registers 1 and 2 . Channels 26 through 50 are in Channel Group Registers 4 and 5.**

Channel Group 1 Register (P5 to 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 (P5 to 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 (P5 to 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 (P6 to P4 Connector Channels 35 through 50)

0020_{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 (P6 to P4 Connector Channels 51 through 66)

0022_{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 (P6 to P4 Connector Channels 67 through 68)

0024_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)															CH 68	CH 67

For all channel groups writing a "1" to a selected channel indicates that the channel is enabled and connected to the internal signal path.

APPENDIX A

Example 1:

The default power-up state of the V581 is no connection to the external calibration input LEMO's "Instr A" and "Instr B" . Thereby allowing the field wiring that is connected to the P6 and P5 connectors to be passes directly to the data system through connectors P4 and P3 respectively. The V581 allows the user four configuration choices: The default state which is connection from the field wiring to the data system. Choice two is connection to an external calibration source via a 2-contact LEMO labeled "Instr A". The third choice is connection to an external calibration source via a 2-contact LEMO labeled "Instr B". The final configuration is an internal connection to ground.

Lets look at an example of using a V581-ZB11 (68 channels) and calibrating all the data system channels using ground. Looking at the Path Grounding Configuration Register one can see that writing the register with data of $0001 would connect the internal signal path to ground.

Path Grounding Configuration Register

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

Bit(s)	Meaning	
15-01	Not used.	
00	Internal Path Control	A "1" in this bit connects the internal signal path to ground.

The next step in this example is to connect all the data system channels (connectors P3 and P4) to the internal signal path which is now grounded. Looking at the Channel Group Registers on page 19. One can see that if data of $FFFF was written to Channel Group Registers 1, 2 , 4 and 5 and data of $0003 was written to Channel Group Registers 3 and 6 then all the channels would be connected to the internal signal path.

Example 2:

Lets look at a second example using a V581-ZA11 (50 channels) with a calibration source connected to "Instr B" and calibrate channel 27.

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 A

The Instrument Path Connection Register is used to monitor and control which internal signal path; on the V581 is connected to an external instrument via the 2-Contact LEMO's labeled "Instr A " and "Instr B".

Bit(s)	Meaning	
15-05	Not used.	
04	Instrument B Control	"0" No connection to internal signal path. "1 " Connect to internal signal path
03-01	Not used	
00	Instrument A Control	"0" No connection to internal signal path. "1 " Connect to internal signal path

In this example writing data of $0010 to the Path Connection Register will connect "Instr B" to the internal signal path.

Caution: When using the V581 with a source connected to one of the 2-contact LEMO's and configuring one of the paths for internal ground connection there exists the potential for connecting a grounded path to the connected source.

The last step is to configure the channel registers with the desired channel number to connect to. Remembering that this example is for a ZA11 (50 channels) option and channels 26 through 50 are in Channel Registers 4 and 5. Then writing data of $0002 would connect channel 27 to the internal signal path which in turn is connected to "Instr B".

Channel Group 4 Register (P4 to P6 Connector Channels 26 through 41)

0020_{16}	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
(R/W)	CH 41	CH 40	CH 39	CH 38	CH 37	CH 36	CH 35	CH 34	CH 33	CH 32	CH 31	CH 30	CH 29	CH 28	CH 27	CH 26

APPENDIX B

```
/*****************************************************************/
/*                                                               */
/*                                                               */
/*  Program Name: module2.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_V581, serial_num;
   int16 error, channel;
   uint16 data,offset,V581,access,data2;
   int i,P3,P4,InstrA,InstrB,option,ground;
   char keybuf;

   system("resman -o");
   InitVXIlibrary();
   system("cls");
   access = A24SuperData;
   printf("\n\n              V581 Relay Multiplexer\n\n");

   /* Getting the Logical Address of the V581 */
   error = FindDevLA("",0xF29,0x581,-1,-1,-1,-1,&V581);
   if (error != 0)
   {
    printf(" Error: FindDevLA() returned %d\n", error);
    CloseVXIlibrary();
    exit(1);
   }

   /* Getting the A24 base address for the V581 */
   error = GetDevInfo(V581, 12, &A24_V581);
   if (error != 0)
```

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

offset = 0x20;
error = VXIinReg(V581,offset,&data);
if (error != 0)
{
 printf(" Error: VXIinReg returned %d\n",error);
 CloseVXIlibrary();
 exit(1);
}
data &= 0xFF;

offset = 0x22;
error = VXIinReg(V581,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(" V581-ZA11 found \n");
 option = 25;
}
else if (data == 'B' && data2 =='1'){
 printf(" V581-ZB11 found \n");
 option = 34;
}
else {
 printf(" Error: V581 option not found \n");
 printf("       Reread Suffix High Register\n");
 printf("       Expect ASCII 'ZA'(0x5A41) or 'ZB'(0x5A42)\n");
 offset = 0x20;
 error = VXIinReg(V581,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(V581,offset,&data2);
 if (error != 0)
```

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

  offset = 0xA;/* look at Serial Number Registers      */
  error = VXIinReg(V581,offset,&data);
  if (error !=0 )
  {
   printf(" Error: VXIinReg returned %d\n",error);
   CloseVXIlibrary();
   exit(1);
  }

  offset = 0xC;
  error = VXIinReg(V581,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_V581 + (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("\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("\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,V581,access;
    uint32 address,A24_V581;
    int16 error;
    int i,k;

    access = A24SuperData;
    /* Getting the Logic Address of V581 */
    error = FindDevLA("",0xF29,0x581,-1,-1,-1,&V581);
    if (error != 0)
    {
     printf(" Error: FindDevLA() returned %d\n", error);
     CloseVXIlibrary();
     exit(1);
    }
    /* Getting the A24 base address for the V581 */
    error = GetDevInfo(V581, 12, &A24_V581);
    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_V581 + 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_V581 + 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 = 0x20 + (i*2);
        address = A24_V581 + 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_V581 + 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_V581 + 0x20;
  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_V581 + 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_V581 + 0x22;
  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_V581 + 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_V581 + 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_V581 + 0x24;
   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_V581 + 0x24;
   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_V581 + 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 = 0x20 + (i*2);
      address = A24_V581 + 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 */
  address = A24_V581;
  error = VXIout(access,address,2,ground);
  if (error != 0)
  {
   printf(" Error: VXIout returned %d \n", error);
   CloseVXIlibrary();
   exit(1);
  }

  return;
}
```