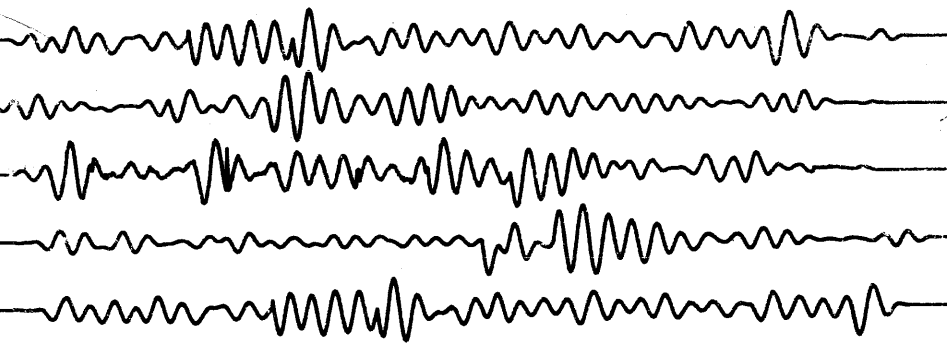
7000-MDAS
Modular Data Acquisition System



REFERENCE MANUAL

TransEra

First Printing November 1984

Table of Contents

Section 1

Overview

This manual contains programming information and hardware specifications for the TransEra model 7000 Modular Data Acquisition System. As you become familiar with the contents of this manual you will have at your fingertips a very powerful, easy to use data acquisition and analysis system.

At the heart of MDAS is a 68000 microprocessor running at 10 MHz. with up to 2048 Kbytes of random access memory. The MDAS operating system resides in 64 Kbytes of ROM. MDAS also includes a real time clock that is battery-powered and a timing circuit that generates precise timing signals.

Analog to digital and digital to analog conversions are performed under control of the microprocessor at rates up to 540 KHz. Access to the data converters is made through signal conditioning cards (up to 16) that plug into the front of MDAS. The signal conditioning cards accommodate a wide variety of signal types that include low level voltages (thermocouples and strain gauges), solid-state relay control for line voltages, current loops and digital output and input control. If your signal conditioning needs are not exstensive, a minimally configured access card is available.

MDAS has the flexibility to support a "dumb" terminal in a stand-alone mode or MDAS can be a peripheral to a host computer, from a mainframe computer to a personal computer. In addition, MDAS can do both at the same time; in a terminal pass-through mode, only one serial link to a host computer is needed to drive a terminal and control MDAS.

Another feature of MDAS that gives added versatility is the choice of communication links to a host. You have your choice of two RS-232 serial ports or an IEEE-488 (GPIB) interface. Having both of these standard interfaces means you can communicate with just about any computer.

Programming MDAS is independent of the computer you use because the commands sent to MDAS are composed of ASCII standard characters. All of the MDAS commands consist of a 2, 3 or 4 letter mnemonic command code followed by parameters that the command may require. Because the commands use standard characters, programs that you develop for MDAS can be easily moved to different computers as your needs may dictate.

MDAS uses several different data formats internally including different length integers and a 32-bit floating point format. Binary format is used by the commands involving analog data and the floating point format is used by the MDAS math routines to maintain precision. For your ease of programming, MDAS automatically converts from one data format to another as needed.

MDAS is designed to serve you as a high performance data aquisision and analysis system. The versatility of this system gives you the freedom to configure your system with the front-end signal conditioning that you need and allows you to tie into the computer of your choice.

Section 2

Getting Acquainted with MDAS

The MDAS Hardware

The MDAS hardware consists of two card cages interconnected by a common backplane. The card cage visible at the front of MDAS receives the various analog and digital signal conditioning cards. The second card cage is at the back of the MDAS chassis, behind the back panel. This card cage receives the power supply, the processor board, the analog board and (optionally) the memory board.

The card cage at the front of MDAS will hold up to 16 signal conditioning cards. Each conditioning card interfaces with 4 analog data channels that are connected to the analog card. Identification of the channels is as follows: the slots are numbered from 1 through 16, starting at the left, and the individual channels for a specific slot are 'numbered' A through H starting at the top and going down. For example, the second channel down on a signal conditioning card plugged in the fifth slot is channel B5. Any signal conditioning card can be inserted into any slot; however the channel number referencing a certain signal conditioning card depends on the location of the card.

Signal conditioning cards are inserted along the card guides until the connector has been firmly seated. The black keeper knob at the bottom of the card provides a positive lock to prevent accidental removal of the card. The black keeper knob must be pulled to the extended position before the signal conditioning card can be removed.

The card cage at the back of MDAS has 4 positions. The top is reserved for the power supply module and the bottom position is reserved for one of the several analog cards available. The processor card and the optional memory card occupy either of the remaining middle positions. The back panel must be removed before any of these cards can be installed or taken out. The back panel is removed by unscrewing the four screws along the outside edges of the back panel.

Care must be taken to ensure that the correct line voltage is applied to MDAS. The line voltage selector card is located under the fuse holder and can be removed without removing the back panel. The two RS-232 and GP-IB connectors are an integral part of the processor card and stay in place if the back panel is removed. The display and input switches are part of the power supply assembly and connection to the processor card is made via a ribbon cable.

The MDAS Communication Ports

MDAS can communicate with a variety of devices through the three communication ports located on the back panel. Two of the ports are standard RS-232 serial interfaces. These are referred to as PORT1 and PORT2. The third port is an IEEE-488 standard parallel port also known as GPIB (General Purpose Interface Bus). This port is simply referred to as GPIB. The MDAS operating system also considers the switch/display module, located on the back panel of MDAS, as a communication device even though its capabilities are somewhat limited compared to devices generally attached to PORT1, PORT2 or GPIB.

The information transferred through the data ports can be categorized into four types: Commands, Errors, Input and Output. Commands are sent to MDAS from a controlling device and govern the operation of MDAS. Errors are output from MDAS, indicating a problem with the execution of a given command. The data needed by MDAS or generated by MDAS is categorized by the direction of data flow, Input for data going into MDAS and Output for data going out of MDAS. Any of the information categories can be directed to any of the three communication ports but only Errors information can be directed to the switch/display module.

The following example illustrates one of the numerous system configurations possible. A computer attached to GPIB could send Commands to MDAS; a source of Input data could be attached to PORT2; a terminal attached to PORT1 could display Output data; any Errors could be displayed on the switch/display module on the back panel of MDAS. On the other hand, a system may consist of a single device (most probably a computer) that is assigned to all four of the information groups, Commands, Errors, Input and Output.

The RS-232 Ports

There are two RS-232 ports on MDAS. These are referred to as PORT1 and PORT2. RS-232 is an asynchronous, bit-serial standard that is supported by most computers and peripheral devices. There are a number of parameters associated with RS-232 interfaces; these parameters must be set the same in both MDAS and the device interfaced via RS-232. Table 2.1 (page 2-4) shows the various parameter options available on MDAS.

The baud rate is the rate at which data is transmitted over the RS-232 interface. Since this interface is a bit-serial standard, the baud rate is the number of bits per second that are transmitted.

Parity is a method of verifying that data is correctly received. On the transmitting end, a parity bit is sent after the bits that make up the transmitted byte. The parity bit is set such that the number of '1's transmitted is either even or odd. On the receiving end, the parity is checked and the parity bit discarded. If there is a parity discrepancy, an error message is generated.

The Echo option determines if MDAS will echo data that it receives. This is another method of data verification; if the echo option is active, every character that is received is transmitted back to the sending device. Thus if MDAS did not receive a character correctly, the sending device would receive back a different character than it sent.

Xon/xoff is a method of controlling the data flow over an RS-232 interface. If xon/xoff is active and the receiving device of a data transfer cannot process any more data, the receiving device sends an ASCII control character (^S) back to the sending device. Upon receipt of a ^S, the sending device suspends data transfer until the receiving device sends a ^Q, indicating that it is ready to receive more data. If xon/xoff is not active, then control of data transfer is done with the dedicated control lines established by the RS-232 protocol.

The Delimiter option is not technically an RS-232 parameter but it controls the type delimiter between values sent from MDAS. A Carriage Return (<CR>) can be specified or both a Carriage Return and Line Feed (<CR><LF>) may be used to delimit values. The format needed depends on the requirements of the device that MDAS is sending data to.

Some RS-232 parameters are pre-set and not alterable. The number of bits per character is 7. The number of stop bits is 1 for all baud rates between and including 75 and 4800. For baud rates 9600 and 19200, the number of stop bits is 2. Each channel operates in full-duplex mode.

The GPIB Port

GPIB (General Purpose Interface Bus) is a standard designated by IEEE-488. There is one GPIB port on MDAS and it is referred to simply as GPIB. It is a byte-parallel port accompanied by several control and handshake lines. There are only two parameter options for GPIB, address and delimiter.

The address option is used to select the GPIB device address that MDAS should respond to. The range of allowed addresses is 0 to 30 inclusive. There are no secondary addresses required; if any secondary addresses are received, they are ignored.

The delimiter option is the same as for PORT1 and PORT2. Depending on the requirements of a receiving device, MDAS can delimit values with either a Carriage Return (<CR>) or both a Carriage Return and Line Feed (<CR><LF>).

MDAS is not able to be a GPIB bus controller. The GPIB function subsets to which MDAS conforms are the following:

SH1 Source Handshake capability
AH1 Acceptor Handshake capability
T8 Basic Talker with no serial poll
TE0 No Talker with address Extension capability
L4 Basic Listener
LE0 No Listener with address Extension capability
SR0 Service Request capability
RL0 No Remote Lockout capability
PP0 No Parallel Poll capability
DC0 No Device Clear capability
DT0 No Device Trigger capability
C0 No Controller capability

The Switch-Display Module

The switch/display module is located on the back panel and consists of a 12 character display and three input switches. The three switches, SCROLL, RESET and ENTER are used to configure the communication ports according to the needs of the system. The display is normally off unless the configuration set-up mode has been entered or if an error occurred while the switch/display module was set up as the Errors device. The operation of the switch/display module is explained in detail in the next section.

How to Configure the MDAS Communication Ports

The configuration of the MDAS communication ports is done by entering the configuration set-up mode. Normally the MDAS operating system is either idling (waiting for a command) or executing a command. In both cases, the display is off, unless an error has occurred. By pressing the ENTER switch, MDAS will enter the configuration set-up mode and the first entry of the set-up menu will appear on the display. When in this mode, port assignments for commands, error messages and data transfers can be made. Also, the communication parameters for the three ports as required by the system configuration can be selected. All of the set-up information can be saved in non-volatile memory. When MDAS is turned on, the communication ports are automatically configured to the set-ups stored in the non-volatile memory.

In the configuration set-up mode, the various assignments are presented as entries on the set-up menu. Each menu entry appears one at a time on the display. To move through the menu entries, press the SCROLL switch and the next entry will appear. When the end of the menu is reached, the SCROLL switch returns to the top, to the first menu entry. A list of the menu entries follows:

*COMMAND DEV	Command Device	(CMD)
*OUTPUT DEV	Output Device	(OUT)
*INPUT DEV	Input Device	(INP)
*PORT1	RS-232 Port 1	
*PORT2	RS-232 Port 2	
*GPIB	GPIB Port (IEEE-488)	
*SAVE	SAVE set-up	

The first three menu entries are used to assign the three information types, Command, Output and Input, to the devices that suit the needs of the system. These three entries are also MDAS commands; this means that under program control assignment of the information types can be changed to different devices. The next three entries allow for the configuration of the communication ports to the requirements of the system devices. The last entry provides for the current set-up to be saved in the non-volatile memory.

When all the required modifications have been made, the RESET switch will return the MDAS operating system back to what it was doing before entering the set-up mode.

Making Port Assignments

The port assignments direct the four types of information to the desired devices. An example may serve best in explaining how to make the port assignments. First the information type is selected from the menu by pressing the SCROLL switch until the type is displayed. This will be one of the four first menu entries, *COMMAND DEV, *INPUT DEV, or *OUTPUT DEV.

Each of these menu entries has a sub-menu from which the port to be assigned is selected. To enter the sub-menu, the ENTER switch is pressed. The current port assignment will then be displayed. Entrance into the sub-menu is indicated by the absence of the '*' in the display. Once the sub-menu has been entered, the switches are used to modify the port assignment and/or exit the sub-menu. The switches perform the following functions in these sub-menus:

SCROLL Selects the next port assignment option.

RESET Selects the next menu entry without affecting the current port assignment.

ENTER Makes the displayed port the current port assignment and then selects the next menu entry.

The sub-menu options are the same for these four entries with the exception that *ERRORS DEV includes the switch/display module (referred to as LED). The options are the following:

PORT1
PORT2
GPIB

Table 2.1
RS-232 Parameter Options

Baud Rates		Parity	Echo	Xon/Xoff	Delimiter
75	110	no parity	no echo	no xon/xoff	<CR>
134.5	150	even parity	echo	xon/xoff	<CR><LF>
300	600	odd parity			
1200	1800				
2000	2400				
4800	9600				
19200					

Configuring the Ports

The menu entries *PORT1, *PORT2 and GPIB are used to set port parameters to the correct option in order to establish communication to the devices that make up the system. The sub-menus are made up of the options that are available for the selected port. After selecting one of the ports (*PORT1, *PORT2 or *GPIB) pressing the ENTER switch will cause the current value of the first option to be displayed. From then on the panel switches perform the following functions:

SCROLL Selects the next option of the sub-menu.

RESET Selects the next menu entry without affecting the current option assignment.

ENTER Makes the displayed option the current option and then selects the next sub-menu entry or next menu entry if there are no more sub-menus.

The sub-menus for the RS-232 ports are baud rate, parity, echo, xon/xoff and delimiter. The options available for these sub-menus are given in Table 2.1.

The sub-menus for the GPIB port are address and delimeter. The options for the address sub-menu are 0 through 30 and the options for the delimeter sub-menu are the same as for the RS-232 ports.

Saving the Current Set-Up

The *SAVE entry in the menu is used to store all the current options in non-volatile memory; to store the current options the ENTER switch must be pressed. When MDAS is turned on, the communication parameters are all set according to what was stored in the non-volatile memory.

The MDAS Command Syntax

MDAS commands are divided into several fields. Each field contains letters, digits, "+", "-", and ".". All other characters with the exception of the semicolon (;), are treated as field delimiters. The semicolon is used to terminate a command. Valid delimiters include Carriage Return and Line Feed characters which means that MDAS commands may span more than a single line.

The first field in an MDAS command is a two-, three- or four-letter code that identifies the command. The mnemonic nature of the command code is intended to make command code easier to remember. This field is followed by delimiters and as many other fields as required by the command. Parameter fields are generally numeric, and may be either floating point or fixed point, and either signed or unsigned.

MDAS Data Formats

Several data formats are used with MDAS. Internally, MDAS uses a 32-bit floating point format as well as 3 binary formats. Data transmission is done in either a floating point format or a fixed point format using ASCII characters.

Internal Floating Point Format

The floating point format consists of a sign bit, a seven bit exponent and a 24 bit mantissa. Each floating point number occupies four bytes of memory. The range of numbers that this format handles follows:

Largest + number:	9.22337177E+18
Smallest + number:	5.42101070E-20
Largest - number:	-9.22337177E+18
Smallest - number:	-5.42101070E-20

Transmission of floating point data is done using ASCII characters and according to this format: -123.456E-12, where the minus signs show the position of optional plus signs.

All of the math commands use the floating point format. If the data in an input buffer is not in floating point format, it is converted to floating point automatically as the math is performed; the input data buffers themselves are not altered by the math commands. A buffer must be defined large enough to accommodate the floating point format if any math commands are to be used on that buffer.

Internal Binary Format

There are three binary formats used internally; these are byte (8 bits), word (16 bits) and long-word (32 bits). In each case the binary data is represented in 2's complement notation. All of the data conversion routines use the word format with one exception; if a buffer is defined as floating-point, the ANALOG INPUT (AI) command will store gain-adjusted voltage in floating-point format.

The long-word format is always used with the Count Input command. This allows for maximum resolution and versatility. The long-word format may also be used with the Digital Input and Digital Output commands for configurations with more than 16 digital channels.

The byte format is only used with the Digital Input and Digital Output commands. If the number of digital channels is eight or less, the byte format will make more efficient use of memory.

FFT Frequency Format

MDAS uses two formats to represent frequency data. The first format, FFT Real Format, is used to represent frequency data derived from signals having no imaginary components. The second format, FFT Complex Format, is used for data derived from signals having both real and imaginary components. In the FFT Real Format, the frequency data for a real signal occupies the two output buffers, each 1/2 the length of the input buffer that contains the real signal. The first output buffer contains the real coefficients of the frequency data. The second output buffer contains the imaginary coefficients of the frequency data, with the exception of the first term. This term is 1/2 the real coefficient of 1/2 the sampling frequency; the imaginary portion of the 0th or DC frequency term which would normally occupy that position is, of course, zero. This format is produced by the FFT real command and is used as input by the INVerse fft command. The format of these buffers is illustrated in Table 2.2 (N is the length of the input buffer.)

In the FFT Complex Format, the frequency data for a complex signal occupies the outputs buffers, each the same length as the input buffers that contain the complex time-domain signal. The first output buffer contains the real coefficients of the frequency data and the second output buffer contains the imaginary coefficients of the frequency data. This format is produced by the FFT complex command and is used as input by the INVerse fft command. The format of these buffers is illustrated in Table 2.3.

If a spectrum of a signal having no imaginary components is represented in the second format, the coefficients of the imaginary portion of the DC term is zero, as is the coefficient of the imaginary portion of the -N/2 frequency term. All of the coefficients of terms whose position is greater than or equal to N/2 (that is, the negative frequency terms) are the complex conjugates of the corresponding positive frequency terms.

If a spectrum of a signal having no imaginary components is represented in both formats, the data have the relation illustrated in Table 2.4.

Table 2.2
FFT Real Format

POSITION in buffers (0 is first element)	MEANING
0 (in first buffer)	real coefficient of DC or 0 frequency term
0 (in second buffer)	1/2 real coefficient at 1/2 sampling freq.
1 (both buffers)	coefficient of 1/N * sampling frequency term
2 (both buffers)	coefficient of 2/N * sampling frequency term
.	.
.	.
.	.
N/2 - 2	coefficient at (N-2)/2N * sampling freq.
N/2 - 1	coefficient at (N-1)/2N * sampling freq.

Table 2.3
FFT Complex Format

POSITION in buffers (0 is first element)	MEANING
0 (in real buffer)	d. c. or 0 frequency term
1 (both buffers)	1/N * sampling frequency term
2 (both buffers)	2/N * sampling frequency term
.	.
.	.
.	.
N/2 - 2	coefficient at (N-2)/2N * sampling freq.
N/2 - 1	coefficient at (N-1)/2N * sampling freq.
N/2	coefficient at 1/2 * sampling freq.
N/2 + 1	coefficient at -(N-1)/2N * sampling freq.
N/2 + 2	coefficient at -(N-2)/2N * sampling freq.
.	.
.	.
.	.
N-2	coefficient at -2/2N * sampling freq.
N-1	coefficient at -1/2N * sampling freq.

Table 2.4
FFT Real and FFT Complex Format Comparison for Input with No Imaginary Components

POSITION in first format	EQUIVALENT in second format
0 (in real buffer)	0 (in real buffer)
0 (in imaginary buffer)	-N/2 (in real buffer)
1 (both buffers)	data at pos. 1 + complex conjugate of data at pos N-1
2 (both buffers)	data at pos. 2 + complex conjugate of data at pos N-2
.	.
.	.
.	.
N/2-1 (both buffers)	data at pos. N/2-1 + complex conjugate of data at pos N/2+1

Error Messages

If MDAS encounters a problem executing a command, an error message is sent to the device currently assigned to receive error information. A device can be assigned either with the *ERRORS DEV command or by using the switch/display module. The twelve errors are:

1.	"VALUE "	Value of parameter invalid
2.	"RANGE "	Parameter out of range
3.	"WIDTH "	Invalid word width or unmatching word sizes
4.	"CNFLCT"	Two output buffers have the same name
5.	"SYNTAX"	Incorrect number of parameters in command line
6.	"COMAND"	Invalid command
7.	"MEMORY"	Not enough memory for requested operation
8.	"SPEED "	Period too fast for sequence
9.	"MATH "	Divide by zero
10.	"LABEL "	Incorrect label definition
11.	"STACK "	Stack overflow
12.	"SPAWN "	Error in a spawned process

If the Error Device is set to the LED display, the error message is displayed on the LED display.

The format for error messages is:

CC:mmmmmm [n]

CC is the command code of the command that MDAS tried to execute. mmmmmm is the error message as printed in the above list. [n] is the position of the bad parameter in the command line. If the command code itself is not 2, 3 or 4 characters, then APPL is displayed in place of a command code (APPLication error).

When a GET ERROR command is issued, only the error number is sent to the data output device, the error message on the LED display is cleared and the error number is set to 0. The CLEAR ERROR command sets the error number to 0 and does not affect the LED display.

Section 3

MDAS Commands

The descriptions of the MDAS commands have been organized alphabetically into three categories: Information Transfer commands, MDAS Environment commands, and Math commands.

The Input/Output commands include all commands that involve the movement of data, analog or digital, in or out of an MDAS. This category includes such commands as those to read analog data, read the real-time clock and send data to a host.

The MDAS Environment commands include commands that control the manner in which MDAS operates. Commands in this category include defining buffers, defining output device and setting the trigger mode.

The Math commands perform various mathematical functions on data in MDAS. There is no data transferred but the commands are all executed within MDAS. Square root, scaling and addition are some of the commands in the Math category.

In order for MDAS to correctly interpret commands, the syntax described in this section must be followed. Each command statement consists of three parts: the command code, command parameters and the command terminator. The command code is the two, three or four letter code at the beginning of a command statement and the command parameters are shown enclosed by angle brackets, <>. The command terminator is the semicolon.

Some commands have optional parameters; these are indicated by brackets, []. Any parameters enclosed in brackets do not have to be included in the command statement. Any parameters not enclosed by brackets must be present in the command statement. Additionally, some optional parameters or parameter groups can be repeated. This is indicated by [...] and means the contents of the previous optional parameters may be repeated.

Each command description is accompanied by an example in BASIC. The examples assume that host device #1 has been opened and is either an RS-232 or GPIB device connected to MDAS. It is also assumed that device #1 is connected to an MDAS port that has been set as the command port, the input port, the output port and the error port. The examples are described from the MDAS point of view. More extensive examples can be found in Appendix A.

Identification of the MDAS channels depends on the slot where the conditioning card is located and the position of the channel on the card. The slots are numbered from 1 through 16, starting at the left, and the individual channels for a specific slot are 'numbered' A through D starting at the top and going down. For example, the second channel down on the 4th card is channel b4.

Analog Input

Reads analog data and stores it in MDAS memory

Syntax: AI <reps> <chan> <bufA> <gain> [<chan> <bufn> <gain>] [...];

where:
- <reps> number of repetitions of the sequence
- <chan> name of channel to take data from
- <bufA,n> name of buffer to receive data
- <gain> gain factor for channel

Description

ANALOG INPUT performs an analog to digital conversion on each channel specified in the command and stores the digital data in the specified buffer. Before a conversion is performed, the internal gain is set to the value specified by <gain>. The values of amplification are 1, 2, 4, 8, 16, 32, 64 and 128.

Buffers that are intended to be used with this command must be defined either as floating point or word integer (type 2 or 4). For buffers that are defined as floating point, the value stored in the buffer is divided by the gain whereas in the integer format, the data is not gain adjusted. Stored values in integer buffers have a range of -32768 to 32767. Any data conversion specified by the DeFiNe Channel instruction is used in computing the final result if a floating point buffer is used.

Conversion ends when the number of repetitions (<reps>) of the specified sequence are taken. If <reps> contains a decimal point, the portion to the right of the decimal point represents the number of sequences to retain before the trigger becomes true.

The same channel may be specified more than once in the list; if this is done, alternate samples from that channel will be placed in two or more different buffers. The same buffer may not be specified more than once in the list; if this is done, a "CNFLCT" error will be generated.

If the period specified by the SET PERIOD command is not long enough to take the requested number of channels in the sequence, a "SPEED" error will be generated. The minimum period allowed depends on the number of channels in the sequence, the trigger type and whether or not any pre-trigger samples have been requested.

For a sequence of just one channel and no pre-trigger samples, there are four minimum period values depending on the chosen parameters. If the trigger type is 1-3 or 5-8 and the number of samples is less than 65535 the minimum period is 2.6E-6 seconds. If the trigger type is 1-3 or 5-8 and the number of samples is greater than 65535 the minimum period is 3.6E-6 seconds. If the trigger type is 4 the minimum period is 18E-6 seconds. If the trigger type is 0 the minimum period is 1.6E-6 seconds.

If the trigger type is 0, the period is less than 2.6E-6 and the buffer type is 2 (word integer), the buffer size must be defined 2 points larger than the desired number of points.

For sequences with more than one channel, the minimum period is derived from the following equations: trigger type 0: minimum period = (channels * 6.2E-6) + 4.4E-6 seconds; trigger type 1-4, and 7-8: minimum period = ((channels + 1)* 6.2E-6 + 5.6E-6) seconds; trigger type 5-6: minimum period = ((channels + 2)* 6.2E-6 + 5.6E-6) seconds. Channels is the number of channels in the sequence and the number of pre-trigger samples does not enter into this equation.

All channels in a multiple channel sequence are sampled once each period; the time interval between samples is 6.2E-6 seconds. The channels in a sequence are sampled in the order that they appear in the command.

The preceeding vaules for minimum periods are only valid for 12-bit converter cards. If a 16-bit converter is used the minimum period is increased. The time interval between samples is 25E-6 for the 16-bit converter card. Thus, to calculate the minimum period for a sequence of channels, simply substitute 25E-6 for 6.2E-6 in the preceeding equations.

All channels that have the sample and hold option are put in the hold mode at the beginning of each repetition of the sequence and are returned to sample mode after all the channels in the sequence have been read. The same amount of time is required to read these channels as those that do not have the sample and hold option.

Examples:

```
100 PRINT #1:"ai 235 a1 1 8 b1 2 4;"
```

After the trigger condition is satisfied, channel a1 is read with the gain set at 8 and the result placed in buffer 1. Then channel b1 is read with the gain set to 4 and the result is placed in buffer 2. This sequence is repeated until a total of 235 sequences have been executed.

```
140 PRINT #1:"ai 3000.2000 c1 3 1 d1 4 2;"
```

Channel c1 is read with the gain set to 1 and placed into buffer 3. Channel d1 is read with the gain set to 2 and placed into buffer 4. This sequence is repeated continuously keeping the latest 2000 sequences in buffers 3 and 4. When the trigger condition is satisfied, the sequence continues with the readings being placed in the remaining buffer space until a total of 3000 sequences have been saved in the buffers.

Analog Input to Fifo

Reads analog data and stores it in a fifo buffer

Syntax: AIF <chan> <bufA> <gain> [<chan> <bufn> <gain>] [...];

where: <chan> name of channel to take data from
<bufA,n> name of fifo buffer to receive data
<gain> gain factor for channel

Description

ANALOG INPUT to FIFO performs an analog to digital conversion on each channel specified in the command and stores the digital data in the specified fifo buffer. Before a conversion is performed, the internal gain is set to the value specified by <gain>. The values of amplification are 1, 2, 4, 8, 16, 32, 64 and 128.

Buffers that are intended to be used with this command must be defined either as floating point or word integer (type 2 or 4). For buffers that are defined as floating point, the value stored in the buffer is divided by the gain whereas in the integer format, the data is not gain adjusted. Stored values in integer buffers have a range of -32768 to 32767. Any data conversion specified by the DeFiNe Channel instruction is used in computing the final result if a floating point buffer is used.

The same channel may be specified more than once in the list; if this is done, alternate samples from that channel will be placed in the specified buffers. The same buffer may also be specified more than once in the list.

The channels in a sequence are sampled in the order that they appear in the command. All channels in a multiple channel sequence are sampled once each time the command is issued. All channels that have the sample and hold option are put in the hold mode as execution of the command begins and are returned to sample mode after all the channels in the command have been read.

The period and trigger parameters that control the Analog Input command are not in effect with the ANALOG INPUT to FIFO command. If a buffer overflow occurs, a "range" error is issued.

Examples:

```
100 PRINT #1:"aif a1 1 8 b1 2 4;"
```

Channel a1 is read with the gain set at 8 and the result placed in fifo buffer 1. Then channel b1 is read with the gain set to 4 and the result is placed in fifo buffer 2.

Analog Output

Sets channel(s) to analog values stored in MDAS memory

Syntax: AO <reps> <chan> <bufA> [<chan> <bufn>] [...];

where: <reps> number of repetitions of the sequence
<chan> name of channel to receive data
<bufA,n> name of buffer(s) that sources data

Description

ANALOG OUTPUT outputs samples from the digital to analog converter to each channel in the sequence. The data to be converted is found in the buffer that corresponds to each channel. The time between sequences is determined by the SET PERIOD command. The minimum time required to output data for a single channel sequence is 5.0E-6 seconds for each data point. For a multiple channel sequence, the time is 10.0E-6 seconds for each channel plus 10.0E-6 seconds overhead for each sequence.

Conversion ends when the number of repetitions specified are taken. The repeat factor (<reps>) causes the entire command to be repeated the number of times indicated. If the number of repetitions (<reps>) contains a decimal point, the number to the right of the decimal point is the number of samples to output starting at the beginning of the buffer.

Buffers that are intended to be used with this command must be defined either as floating point or word integer (type 2 or 4.) The same channel may be specified more than once in the list; if this is done, samples from two or more different buffers will be alternately output to that channel.

The same buffer may be specified more than once in the list; if this is done, samples from a single buffer will be output alternately to two or more channels.

Examples:

```
500 PRINT #1:"ao 1.622 b2 1;"
```

The first 622 numbers in buffer 1 are output to channel b2.

```
280 PRINT #1:"ao 6.1100 b2 1 c2 3;"
```

The first 1100 numbers in buffers 1 and 3 are output to channels b2 and c2. This is repeated 6 times.

Analog Output Analog Input

Sets a channel to an analog value and read a sequence of analog channels

Syntax: AOAI <reps> <chanA> <bufA> <chanX> <bufX> <gain>
[<chann> <bufn> <gain>] [...];

where:
- <reps> number of samples on each input channel
- <chanA> name of channel to output analog value
- <bufA> name of buffer that sources output data
- <chanX,n> name of channel to input analog value
- <bufX,n> name of buffer to store input value
- <gain> gain factor for channel

Description

ANALOG OUTPUT ANALOG INPUT outputs samples from the digital to analog converter to <chanA>. The output data is found in <bufA>. After the first sample has been output on <chanA> the input channels are sampled and the values are stored in the corresponding buffers. The time between sequences is determined by the SET PERIOD command. The minimum period can be calculated using the following equation: ((n + 2)*6.4E-6 + 5E-6) seconds, where n is the number of input channels in the sequence.

Conversion ends when the number of repetitions <reps> specified are taken. If <bufA> has less points than <reps> then <bufA> is repeated starting at the beginning of the buffer.

Buffers that are intended to be used with this command must be defined either as word integer or floating point (type 2 or 4.) The same channel may be specified more than once in the list; if this is done, samples from that channel will be alternately stored in the different buffers. The same buffer may not be specified more than once in the list; if this is done a CONFLCT error will be generated.

No triggers are used and the command is executed when the it is received.

Examples:

```
280 PRINT #1:"per 1e-3;"
500 PRINT #1:"aoai 100 d2 1 a4 10 1 b4 11
```

The values in buffer 1 are output to channel d2, while the values on channels a4 and b4 are stored in buffers 10 and 11. The time between sequences is set up by the set PERiod command to one milisecond.

Analog Output from Fifo

Sets channel(s) to analog values stored in MDAS memory

Syntax: AOF <chan> <bufA> [<chan> <bufn>] [...];

where: <chan> name of channel to receive data
<bufA,n> name of buffer(s) that source(s) data

Description

ANALOG OUTPUT from FIFO outputs samples from the digital to analog converter to each channel in the sequence. The data to be converted is found in the buffer that corresponds to each channel. A single point is output to each specified channel each time this command is executed. The time lapse between the output to successive channels in the command is 10.0E-6 seconds.

The source buffers can be either tagged as fifo buffers or can be regular buffers. If an underflow occurs in a fifo buffer, a "range" error is generated. If the source buffer is a regular buffer and the end of the buffer is reached, the buffer contents are repeated as this command is executed.

Buffers that are intended to be used with this command must be defined either as floating point or word integer (type 2 or 4). The same channel may be specified more than once in the list; if this is done, samples from two or more different buffers will be alternately output to that channel.

The same buffer may be specified more than once in the list; if this is done, samples from a single buffer will be output alternately to two or more channels.

Examples:

```
500 PRINT #1:"aof b2 1;"
```

The next number in buffer 1 is output to channel b2.

```
280 PRINT #1:"aof b2 1 2c 3;"
```

The next number in buffers 1 and 3 are output to channels b2 and 2c.

Count Input

Counts the number of TTL pulses during an interval

Syntax: CI <reps> <time> <chan> <bufA> [<type>];

where: <reps> number of successive intervals to count
<time> counting interval in no. of periods
<chan> number of channel to take data from
<bufA> name of buffer to store counts
<type> the direction of edge to count on

Description

COUNT INPUT counts the number of times a TTL compatible input satisfies the <type> condition during a time interval equal to the <time> times the length of the period set by SET PERIOD. <Reps> is the number of successive time intervals that are counted and must be sent in integer format (no decimal point or scientific notation). The number of counts is stored in <bufA> with an entry for each time interval counted. This command is to be used with the Pulse Frequency card (R8B2). The channels C and D are used as inputs.

The maximum input frequency is 5 MHz. and the highest number that be counted is 4,294,967,295.

<type> is interpreted according to the following table:

<type>	explanation
0	each rising edge of input counted
1	each falling edge of input counted

Default Value

<type> = 1

Examples:

```
160 PRINT #1:"ci 100 6 c2 3 0;"
```

The number of rising edges occurring on channel c2 during 6 periods is saved in buffer 3. This continues for 100 times. At the end, buffer 3 will contain 100 numbers corresponding to the number of transitions during each interval.

Count Multiple Inputs

Counts TTL pulses from multiple channels during an interval

Syntax: CNT <reps> <chan> <bufA> [<chan> <bufn>] [...];

where:		
	<reps>	number of successive intervals to count
	<chan>	name of channel to take data from
	<bufA,n>	name of buffer to store counts

Description

COUNT MULTIPLE INPUTS counts the number of TTL compatible input pulses that occur on the specified channels during the specified time interval; the specified time interval is the time defined by the Set Period command. <Reps> is the number of successive time intervals that are counted and must be sent in integer format (no decimal point or scientific notation). The number of counts is stored in <bufA> or <bufn> with an entry for each time interval counted.

This is a software counting routine in contrast to the Count Input command which is a hardware counting routine. The maximum input frequency for this command is therefore lower and depends on the number of channels requested in the command. The amount of time it takes to scan each channel once is 12E-6 * (chan) + 7E-6 seconds, where chan is the number of channels requested. Both the input low and high times must be longer than the time required to scan all the channels.

Examples:

```
160 PRINT #1:"cnt 100 c2 6 c3 7;"
```

The number of TTL pulses at the inputs to channels c2 and c3 during the current time interval is saved in buffers 6 and 7. This is repeated for a total of 100 consecutive times.

Copy buffer

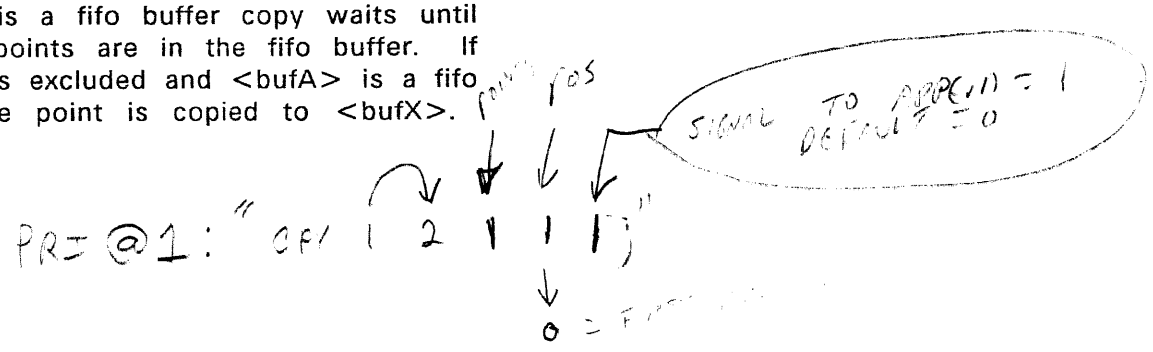
Copies contents of one buffer to another

Syntax: CPY <bufA> <bufX> [<pnts> [<pos>]]; [AMP]

where: <bufA> buffer containing input data
<bufX> buffer to copy data into
<pnts> number of points to copy
<pos> position in buffer to start copying
<APPEND> APPENDS TO RIGHT OF BUFFER

Description

COPY copies the number of points (<pnts>) from <bufA> to <bufX> starting at <pos>. The buffer type of <bufX> is set to the same type as <bufA>. The number of points may be left off in which case each point of <bufA> is copied to <bufX>. The offset may be left off in which case the copy starts with the first point in <bufA>. A zero for <offset> corresponds to the first point in <bufA>. <bufA> may be a fifo buffer but in this case, <pos> cannot be included. If <bufA> is a fifo buffer copy waits until <pnts> points are in the fifo buffer. If <pnts> is excluded and <bufA> is a fifo buffer one point is copied to <bufX>.



Examples:

```
740 PRINT #1:"cpy 1 2;"
```

Buffer 1 is copied to buffer 2.

```
430 PRINT #1:"cpy 1 2 100;"
```

The first 100 points of buffer 1 are copied to buffer 2.

Delete Disk File

Deletes a file from disk

Syntax: DELD <filename>;

where: <filename> name of floppy disk file to delete

Description

DELETE DISK FILE deletes a file from the floppy disk.

Examples:

```
380 PRINT #1:"deld test.run"
```

The file named test.run is deleted from the floppy disk.

Digital Input to Fifo

Reads TTL digital data from I/O channel(s) to a fifo buffer

Syntax: DFF <bufA> <chan> [<chan>] [...]

where: <bufA> destination fifo buffer for input data
<chan> channels to input

Description

DIGITAL INPUT to FIFO inputs digital signals from the specified channels to a buffer that has been properly defined and tagged as a fifo buffer (see Fifo command). The first channel in the list is assigned to the least significant bit in the data word and each succeeding channel is assigned to the next most significant bit. The maximum number of channels is 32. Any unused bits in the data word are set to 0.

The size of the data word is determined by the defined width of the buffer. (See Define Buffer command). The floating point format is not allowed and the buffer width must be large enough for the number of channels specified in the command. One data word is written to the fifo buffer each time this command is executed. If a buffer overflow occurs, a "range" error is issued.

Examples:

```
540 PRINT #1:"dff 2 a4 b4 c4 d4 a5 b5;"
```

The listed channels are read into a word in buffer 2 (previously defined as a byte-wide buffer and tagged as a fifo buffer) with channel a4 being the least significant bit.

Digital Input

Reads TTL digital data from I/O channel(s)

Syntax: DI <reps> <bufA> <chan> [<chan>] [...]

where: <reps> number of repetitions
<bufA> destination buffer for input data
<chan> channels to input

Description

DIGITAL INPUT inputs digital signals on the channels specified. The first channel in the list is assigned to the least significant bit in the data word and each succeeding channel is assigned to the next most significant bit. The maximum number of channels is 32. Any unused bits in the data word are set to 0.

The size of the data word is determined by the defined width of the buffer. (See Define Buffer command). The floating point format is not allowed and the buffer width must be large enough for the number of channels specified in the command.

The minimum period that can be used depends on the number of channels specified in the command. For 1 to 16 channels, the minimum period is (n)*7.2E-6 + 11.8E-6 seconds; for 16 to 32 channels, the minimum period is (n)*7.2E-6 + 12.3E-6 seconds where (n) is the number of channels. <Reps> must be sent in integer format (no decimal point or scientific notation).

The preceeding vaules for minimum periods are only valid for 12-bit converter cards. If a 16-bit converter is used the minimum period is increased. The time interval between samples is 25E-6 for the 16-bit converter card. Thus, to calculate the minimum period for a sequence of channels, simply substitute 25E-6 for 7.2E-6 in the preceeding equations.

If a trigger type of 4 (See TRI) is specified then the internal sample period will be ignored and the samples will be taken on each falling edge of the external trigger. The channel for this trigger is also specified by the Trigger command.

Examples:

```
540 PRINT #1:"di 48 2 a4 b4 c4 d4 a5 b5;"
```

The listed channels are read into a word in buffer 2 with channel a4 being the least significant bit. The command is repeated for a total of 48 times.

Digital Input Word

Reads digital data from I/O channels in byte or word format

Syntax: DIW <reps> <bufA> <chan> [<wflag> [<mask>];

where: <reps> number of repetitions
<bufA> destination buffer for input data
<chan> channels to input
<wflag> word flag
<mask> mask unwanted bits

Description

DIGITAL INPUT WORD inputs digital signals on the channel specified in byte or word format. The <chan> parameter indicates the slot that the data will be read from. For example, if <chan> is d4 then the data will be sampled on the fourth slot. Channel A is always assigned the least significant bit in the data word and each succeeding channel is assigned to the next most significant bit. If the <wflag> is 1 or not included then the data will be sampled in word format (16 bits) and stored into an integer buffer (type 2). If the <wflag> is 0 then the data will be sampled in byte format (8 bits) and stored into a character buffer (type 1). The <mask> parameter is used to mask off unwanted bits (Set to 0).

For a 12-bit converter board the minimum period for sampling a word is 145E-6 seconds and the minimum period for sampling byte data is 65E-6 seconds. For a 16-bit converter board the minimum period for sampling a word is 550E-6 seconds and the minimum period for sampling byte data is 250E-6 seconds.

The DIW command was written to be used with the R5P8 card. This card has a total of 16 TTL level inputs. The top bit is the least significant bit of the data word and the bottom is the most significant bit. If the <wflag> is a 0 then only one byte is read. The top 8 bits can be read by setting channel G on the same slot to a 1 (See SD command) before the DIW command is executed. Similiarly the bottom 8 bits can be read by setting channel G on the same slot to a 0 (example, "SD G1 0;").

If a trigger type of 3 (See TRI) is specified then the samples will start to be stored on the falling edge of the external trigger. If a trigger type of 4 is specified then the internal sample period will be ignored and the samples will be taken on each falling edge of the external trigger.

The channel for these trigger types are also specified by the TRIgger command.

Examples:

```
530 PRINT #1:"per .01;"
540 PRINT #1:"diw 100 1 a3;"
```

The 16 bits in slot 3 are sampled every 1/100 of a second and stored into buffer 1. The command is repeated for a total of 100 times.

```
950 PRINT #1:"dfn 3 50 1;"
955 PRINT #1:"sd a1 0;"
960 PRINT #1:"diw 50 3 a1 0 63;"
```

The bottom 8 bits of slot 1 are read into buffer 3. The 3 most significant bits are mask out (set to 0) and the 5 least signigicant bits remain unchanged.

Digital Input Word Fifo

Stores digital data from I/O channels in byte or word fifo buffers

Syntax: DIWF <bufA> <chan> [<wflag> [<mask>];

where: <bufA> destination buffer for input data
<chan> channels to input
<wflag> word flag
<mask> mask unwanted bits

Description

DIGITAL INPUT WORD FIFO inputs digital signals to byte or word fifo buffers. The <chan> parameter indicates the slot that the data will be read from. For example, if <chan> is d4 then the data will be sampled on the fourth slot. Channel A is always assigned the least significant bit in the data word and each succeeding channel is assigned to the next most significant bit. If the <wflag> is 1 or not included then the data will be sampled in word format (16 bits) and stored into an integer fifo buffer (type 2). If the <wflag> is 0 then the data will be sampled in byte format (8 bits) and stored into a character fifo buffer (type 1). The buffers used must be defined to be fifo by the FIFO command. The <mask> parameter is used to mask off unwanted bits (Set to 0). Only 1 (byte or word) sample is stored for each command given.

The DIWF command was written to be used with the R5P8 card. This card has a total of 16 TTL level inputs. The top bit is the least significant bit of the data word and the bottom is the most significant bit. If the <wflag> is a 0 then only one byte is read. The top 8 bits can be read by setting channel G on the same slot to a 1 (See SD command) before the DIWF command is executed. Similiarly the bottom 8 bits can be read by setting channel G on the same slot to a 0 (example, "SD G1 0;").

Default Values

<wflag> = 1
<mask> = FFFF (Hex)

Examples:

```
5530 PRINT #1:"dfn 1 100 2;fifo 1;"
5540 PRINT #1:"diwf 1 a3;"
```

The 16 bits in slot 3 are stored in buffer 1.

```
9950 PRINT #1:"dfn 3 50 1, 10 100 5;fifo 3;
9960 PRINT #1:"load 10;diwf 3 a1 0;end;
9970 PRINT #1:"sd a1 1;sint 0.5;spn 10 30;"
9980 INPUT #1:snum
```

The top 8 bits of slot 1 are read into fifo buffer 3. The 8 bits are sampled 30 times at a rate of twice a second by the spn command.

Directory

List the files on a floppy disk

Syntax: DIR [<typ>];

format: <num> <dlm> [<str> <dlm>] [...]

where: <num> number of directory entries that will be output
<dlm> delimiter, either <CR> or <CR> <LF>
<typ> either long or short format
<str> string containing the directory information

Description

DIRECTORY outputs a listing of the files on a floppy disk to the data output device (see OUT). The number of entries is sent first, followed by the file listings. If the long option is requested then the size, and time and date the files were created are sent to the output port. If no type is given the default is short format. If there is no internal disk drive a COMMAND error will occur. The time and date format that is output is compatible with the format used on the IBM PC not the format used with the Get Time and Set Time commands. The format for the file string is as follows.

<type>	format	<file string>
0	short	<filename> <del>
1	long	<filename> <size> <tim

Examples:

```
670 PRINT #1:"dir 1;"
680 INPUT #1:num
690 FOR i = 1 to num
700     INPUT #1:str$
710     PRINT str$
720 NEXT i
```

A list of all the files on a floppy are printed in the long format.

Digital Output

Writes TTL digital data to an I/O channel

Syntax: DO <reps> <bufA> <chan> [<chan>] [...]

where: <reps> number of repetitions
<bufA> source buffer for data to be output
<chan> channels to output to

Description

DIGITAL OUTPUT outputs digital signals to the specified channels. The first channel in the list is assigned to the least significant bit in the data word and each succeeding channel is assigned to the next most significant bit. The maximum number of channels is 32.

The size of the data word is determined by the defined width of the buffer. (See Define Buffer command). The floating point format is not allowed and the buffer width must be large enough for the number of channels specified in the command.

The minimum period that can be used depends on the number of channels specified in the command. For 1 to 16 channels, the minimum period is (n)*7.6E-6 + 11.8E-6 seconds; for 16 to 32 channels, the minimum period is (n)*7.6E-6 + 12.3E-6 seconds where (n) is the number of channels.

Conversion ends when the number of repetitions specified are taken. The repeat factor (<reps>) causes the entire command to be repeated the number of times indicated. If the number of repetitions (<reps>) contains a decimal point, the number to the right of the decimal point is the number of samples to output starting at the beginning of the buffer.

Examples:

```
780 PRINT #1:"do 3.90 2 a4 b4 c4 d4 a5 b
```

The first 90 points in buffer 2 are output to listed channels, channel a4 being the least significant bit. The command is repeated 3 times.

Digital Output from Fifo

Writes TTL digital data to I/O channels

Syntax: DOF <bufA> <chan> [<chan>] [...]

where: <bufA> source buffer for data to be output
<chan> channels to output to

Description

DIGITAL OUTPUT from FIFO outputs digital signals to the specified channels. The first channel in the list is assigned to the least significant bit in the data word and each succeeding channel is assigned to the next most significant bit. The maximum number of channels is 32. Each channel is written to once each time the command is executed.

The source buffers can be either tagged as fifo buffers (see Fifo command) or can be regular buffers. If an underflow occurs in a fifo buffer, a "range" error is generated. If the source buffer is a regular buffer and the end of the buffer is reached, the buffer contents are repeated as this command is executed.

The size of the data word is determined by the defined width of the buffer (see Define Buffer command). The floating point format is not allowed and the buffer width must be large enough for the number of channels specified in the command.

Examples:

```
780 PRINT #1:"dof 2 a4 b4 c4 d4 a5 b5;"
```

The next points in buffer 2 are output to listed channels, channel a4 being the least significant bit.

End Count Input

End a sequence of TTL edge counts

Syntax: ECI <chan>;

where: <chan> number of channel data is comming from

Description

END COUNT INPUT ends a sequence of TTL edge counts issued by the SCI command. See SCI and RCI commands. ECI must be given to end a counting sequence.

This command is to be used with the Pulse Frequency card (R8B2). Channels C and D are used as inputs.

Examples:

```
9000 PRINT #1:"eci d1;"
```

The counting of TTL edges on channel d1 is ended.

Get Analog

Reads analog data and sends it out the Output Port

Syntax: GA <chanA> [<chanN>] [...];

format: <point> <del> [<point> <del>] [...]

where:
<chanA,N> name of channel to take data from
<point> the value of a data point
<dlm> delimiter, either <CR> or <CR> <LF>

Description

GET ANALOG performs an analog to digital conversion on each channel specified in the command and the data is sent to the Output Port (see Output Device command.) The data is sent in ASCII format. A given channel may be specified more than once and the total number of channels that can be specified is 64.

Autoranging is performed on each channel; this means that the gain is automatically set to the highest level possible to obtain the most precise result. There are no trigger options with this command; the data conversion begins as soon as the command is received.

Any data conversion specified by the DeFiNe Channel instruction is used in computing the final result.

The time elapsed between samples is not fixed due to the autoranging routine. Roughly, the time between samples is between 20E-6 and 80E-6 seconds.

Channels with the sample and hold option are not put into the hold mode at the beginning of the command. These channels are sampled as they appear in the command sequence.

Examples:

```
430 PRINT #1:"ga a1 a4 b1 b2;"
440 INPUT #1:W,X,Y,Z
```

Channels a1, a4, b1, and b2 are read. The results are output in floating-point format and stored in variables W, X, Y and Z.

Get Buffer information

Sends information concerning a buffer to the defined data port

Syntax: GBUF <buff>;

format: <buff> <size> <type> <fifo> <dlm>

where: <buff> buffer number in question
<size> number of points currently in <buff>
<type> type the buffer is defined as
<fifo> 0 if not a fifo buffer, 1 if it is
<dlm> delimiter, either <CR> or <CR> <LF>

Description

GET BUFFER INFORMATION sends the information concerning a buffer to the I/O port defined as the data output port. (See the define OUTput command). A buffer is defined using the DFN command. The <buff> <size> <type> and <fifo> are sent as one string to the output port. See DFN command for a description of <type>. This command may be helpful during debugging of programs.

Examples:

```
460 PRINT #1:"gbuf 4;"
470 INPUT #1:B$
```

The information of buffer 4 is returned in B$.

Get Condition

Sends the condition flag to the defined data port

Syntax: GC;

format: <con> <dlm>

where: <con> 1 = True, 0 = False
<dlm> delimiter, either <CR> or <CR> <LF>

Description

GET CONDITION sends the condition flag to the I/O port defined as the data output port. (See the define OUTput command). The condition flag is set true (1) or false (0) by the if instructions (IFA, IFBA, IFD, IFBD). The condition flag is set false (0) by the CLear Condition command.

Examples:

```
900 PRINT #1:"ifa a1 > 2.5;"
910 PRINT #1:"gc;"
920 INPUT #1:RES
```

The condition flag is set to 1 if the value on channel a1 is greater than 2.5. Otherwise the condition flag is cleared. The condition flag is stored in variable RES.

Get Channel Information

Sends the definition of a channel to the defined data port

Syntax: GCHN <chan>;

format: <chan> <type> <gain> <dlm>

where: <chan> channel in question
<type> type the channel is defined as
<gain> gain the channel is defined to have
<dlm> delimiter, either <CR> or <CR> <LF>

Description

GET CHANNEL INFORMATION sends the definition of a channel to the I/O port defined as the data output port. (See the define OUTput command). A channel may be defined using the DFNC command. The <chan> <type> and <gain> are sent as one string to the output port. See DFNC command for a description <type> and <gain>.

Examples:

```
900 PRINT #1:"dfnc b4 2 100;;"
910 PRINT #1:"gchn b4;"
920 INPUT #1:C$
```

Channel b4 is defined as a J type thermocouple with a gain of 100. The gchn command will return the information sent in the dfnc command.

Get Period

Sends the current period to the defined data port

Syntax: GPER;

format: <peroid> <dlm>

where: <period> current period in seconds
<dlm> delimiter, either <CR> or <CR> <LF>

Description

GET PERIOD sends the current period to the I/O port defined as the data output port. (See the define OUTput command). The period is set up using the PERiod command.

Examples:

```
8300 PRINT #1:"gper;"
9310 INPUT #1:P$
```

The period is return in P$.

Get Time

Outputs the time in ASCII characters

Syntax: GT;

format: <day> <mon> <year> <hour>:<min>:<sec> <dlm>

where: <dlm> delimiter, either <CR> or <CR> <LF>

Description

GET TIME outputs the date and time from the MDAS real time clock to the port assigned by the "OUTput device" command. The format of the output is as follows:

dd mmm yy hh:mm:ss

The month is expressed as a three letter abbreviation as follows: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC.

The date and time data is followed by a Carriage Return (and a Line Feed if the port was set for Line Feed.)

Examples:

```
880 PRINT #1:"gt;"
```

The MDAS time is sent to the host.

Get Trigger Information

Sends the trigger information to the defined data port

Syntax: GTRI;

format: <type> [<chan> <valueA> [<valueB>]] <dlm>

where: <type> type of trigger
<chan> trigger channel
<valueA> level used by all triggers
<valueB> level used by trigger types 5 and 6
<dlm> delimiter, either <CR> or <CR> <LF>

Description

GET TRIGGER INFORMATION sends the trigger information to the I/O port defined as the data output port. (See the define OUTput command). The triggers are used in the AI and SCAN commands. For a description of the paramaters see the TRIgger command.

Examples:

```
220 PRINT #1:"gtri;"
230 INPUT #1:T$
```

The trigger information is input into T$.

If Analog

Compares the value of a channel to a constant

Syntax: IFA <chan> <op> <cons>;

where: <chan> channel number to take a sample from
<op> operator to use in the comparison
<cons> constant used in comparison

Description

IF ANALOG compares the value on channel <chan> against the constant <cons>. The result of the comparison sets the condition flag to 1 if the result of the comparison is true and 0 if the result if false. The result can be obtained by using the GET CONDITION command. The comparison can also be used to create a conditional branch in an executable buffer (see BRA). The operator <op> used in the comparison corresponds to one of the following:

<op>	operator
=	equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
<>	not equal to

Autoranging is performed on the channel. this means that the gain is automatically set to the highest level possible to obtain the most precise result.

Any data conversion specified by the DEFINE CHANNEL command is used in computing the final result used in the comparison. This makes it possible to test against units other than voltage (temperature, micro-strains, amperes).

Examples:

```
330 PRINT #1:"ifa b9 < 3.8;"
340 PRINT #1:"gc;"
```

A sample is taken on channel b9 and compared to the constant 3.8. The Get Condition returns the result of the comparison.

```
530 PRINT #1:"load 5;"
540 PRINT #1:"labl repeat;;"
550 PRINT #1:"ifa c2 < 2.5;"
560 PRINT #1:"bra repeat;"
570 PRINT #1:"gt;"
580 PRINT #1:"end;"
590 PRINT #1:"run 5;"
```

Buffer 5 is loaded with commands used to create a while loop. Samples on channel c2 will continue to be taken while the value of channel c2 is less than 2.5. The time is sent to the output device when the comparison becomes false.

If Analog Buffer

Compares the average value of a buffer to a constant

Syntax: IFBA <buff> <op> <cons> [<pnts>];

where: <buff> buffer containing the sample points
<op> operator to use in the comparison
<cons> constant to compare against
<pnts> number of points to average

Description

IF ANALOG BUFFER compares the average value of a buffer against a constant. If <pnts> is not included only one point is used in the comparison. If <buff> is a fifo buffer the comparison will not be made until the number of points requested <pnts> have been averaged. The result of the comparison sets the condition flag (see Get Condition) to 1 if the comparison result is true and 0 if the result is false. The result can be obtained by using the GET CONDITION command. The comparison can also be used to create a conditional branch in an executable buffer (see BRA). The operator <op> used in the comparison corresponds to one of the following:

<op>	operator
=	equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
<>	not equal to

Examples:

```
220 PRINT #1:"dfn 3 100 4; fifo 3;"
230 PRINT #1:"aif a2 3 1 b2 3 1"
240 PRINT #1:" c2 3 1 d2 3 1;"
250 PRINT #1:"ifba 3 > 5 4;"
260 PRINT #1:"gc;"
```

Four samples are taken on channels a2, b2, c2, and d2 and are stored in the fifo buffer number 3. If the average of the four samples is greater than 5, a 1 is sent to the output device, otherwise a 0 is sent.

```
520 PRINT #1:"dfn 10 100 4;"
530 PRINT #1:"load 10;"
540 PRINT #1:"labl repeat;;"
545 PRINT #1:"ai 100 a12 1;"
550 PRINT #1:"ifba a12 >= -1.5 100;"
560 PRINT #1:"bra repeat;"
570 PRINT #1:"gt;"
580 PRINT #1:"end;"
590 PRINT #1:"run 5;"
```

Buffer 10 is loaded with commands used to create a while loop. 100 Samples are taken on channel a12. While the average of the 100 samples is greater than or equal to -1.5, samples will continue to be taken. The time is sent to the output device when the comparison becomes false.

If Digital Buffer

Compares the bits of a point in a buffer to a constant

Syntax: IFBD <cons> <op> <buff>;

where: <cons> constant bit pattern to compare against
<op> operator to use in the comparison
<buff> buffer containing the sample point

Description

IF DIGITAL BUFFER compares the bit pattern of a point in a buffer to a constant. Only the first point in the buffer is used in the comparison. If <buff> is a fifo buffer the comparison will not be made until there is a point in the buffer. The result of the comparison sets the condition flag to 1 if the comparison result is true and 0 if the result is false. The result can be obtained by using the GET CONDITION command. The comparison can also be used to create a conditional branch in an executable buffer (see BRA). The operator <op> used in the comparison corresponds to one of the following:

<op>	operator
=	equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
<>	not equal to

Examples:

```
220 PRINT #1:"dfn 20 100 1; fifo 20;"
230 PRINT #1:"dfn 5 100 5 6 100 5;"
240 PRINT #1:"load 5;"
250 PRINT #1:"dff 20 a5 b5;end;"
260 PRINT #1:"load 6;"
270 PRINT #1:"labl again; ifbd 3 <> 20;"
280 PRINT #1:"bra again; end;"
290 PRINT #1:"sint 1.0;"
300 PRINT #1:"spn 5 1000;"
310 INPUT #1:SPNNUM
320 PRINT #1:"run 6;"
```

Buffer 20 is defined to be a fifo buffer of size char. Buffer 5 is loaded with the command to do a digital input on a fifo buffer. Buffer 6 is loaded with the commands to test a point in buffer 20 to see if the two least significant bits are set. Buffer 5 is run in background every second for 1000 seconds. The spawned process number is read into variable SPNNUM. Buffer 6 is run testing the points in fifo buffer 20 until the comparison becomes false (the value in buffer 20 is equal to 3).

If Digital

Compares the bits from a sequence of channels to a constant

Syntax: IFBA <cons> <op> <chan> [<chan>] [...];

where: <cons> constant bit pattern to compare against
<op> operator to use in the comparison
<chan> channels to input

Description

IF DIGITAL compares the bit pattern from a sequence of channels to a constant. The first channel in the list is assigned to the least significant bit in the data word and each succeeding channel is assigned to the next most significant bit. The maximum number of channels is 32. Any unused bits in the data word are set to 0. The result of the comparison sets the condition flag to 1 if the comparison result is true and 0 if the result is false. The result can be obtained by using the GET CONDITION command. The comparison can also be used to create a conditional branch in an executable buffer (see BRA). The operator <op> used in the comparison corresponds to one of the following:

<op>	operator
=	equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
<>	not equal to

Examples:

```
310 PRINT #1:"ifd 7 = a1 b1 c1;"
320 PRINT #1:"gc;"
330 INPUT #1: temp
```

The three digital channels (a1, b1, c1) are tested to see if they are set. If all three are set the GET CONDITION command will return a 1 otherwise it will return a 0.

Memory Space

Outputs the memory space available

Syntax: MEM;

format: <mem> <dlm>

where: <mem> memory space available
<dlm> delimiter, either <CR> or <CR> <LF>

Description

MEM returns the number of bytes of memory that are available for new buffer use. This does not include memory space used by currently defined buffers.

Examples:

```
880 PRINT #1:"mem;"
890 INPUT #1:S
```

The memory space in bytes is put in variable S.

Pulses Output

Sends TTL pulses to a Pulse Output card

Syntax: PO <reps> <low> <high> <chan>;

where: <reps> number of pulses to output
<low> duration of low portion of pulse in seconds
<high> duration of high portion of pulse in seconds
<chan> name of channel to output pulses to

Description

PULSES OUTPUT sends the number of pulses (<reps>) requested to a <chan>nel on a Pulse Output card. The duration of the low and high levels of the pulse are sent in ASCII format and are in seconds. The range of <reps> is from 0 to 65535 and must be sent in integer format (no decimal point or scientific notation). If <reps> is 0, the pulse train continues until the next OUTPUT PULSES command overrides the <reps> parameter. Only one channel at a time can be set up in the continuous output mode.

The output begins at a low TTL level and stays for the duration specified by <low>. The output then goes to a high TTL level for the duration specified by <high> and then returns to a low TTL level. The duration time for either level ranges from a minimum of 400 nsec. to a maximum of 858.98 seconds.

Examples:

```
300 PRINT #1:"po 21000 1e-4 2e-4 c4;"
```

21000 pulses are generated at channel c4 with a low duration of 100 usec. and a high duration of 200 usec.

Pulse Digital

Sends TTL pulses to a selected Digital Out channel

Syntax: PULD <reps> <low> <high> <chan> [<chan>] [...];

where: <reps> number of pulses to output
<low> duration of low portion of pulse in seconds
<high> duration of high portion of pulse in seconds
<chan> name of channel to output pulses to

Description

PULSES DIGITAL sends the number of pulses (<reps>) requested to a <chan>nel on a Digital Output card. The duration of the low and high portions of the pulse are sent in ASCII format and are in seconds. The output begins at a low TTL level and stays for the duration specified by <low>. The output then goes to a high TTL level for the duration specified by <high>; then the output returns to the low level.

The range of <reps> is from 1 to 2147482752 and must be sent in integer format (no decimal point or scientific notation). The minimum duration time for either level is 3.4E-6 seconds for one channel and (n*3.6 + 6.6)E-6 seconds for multiple channels (n being the number of channels). The maximum duration time for either level is 858.98 seconds.

Examples:

```
310 PRINT #1:"pul 14000 1e-4 2e-4 c4 ;"
```

14000 pulses are generated at channel c4 with a low duration of 100 usec. and a high duration of 200 usec.

Ramp Digital Pulses

Ramp output pulse rate for Stepper Motor

Syntax: RAMP <chan> <steps> <vel> <acc> [<dir> [<h/f>]];

where: <chan> channel number (slot)
<steps> number of steps
<vel> final velocity
<acc> acceleration
<dir> direction
<h/f> step mode (half or full)

Description

The RAMP command is used to drive a stepper motor with a ramping velocity. The <chan> parameter is the number of the slot that the stepper motor card is plugged in. The number of steps is specified by the <steps> parameter. The final velocity is given in steps per second and the acceleration in steps per second per second. The <dir> and <h/f> parameters remain unchanged if not specified. The direction of rotation is given by the <dir> parameter; 1 is clockwise and 0 is counter-clockwise. The mode of operation, full- or half-step, is selected by the <h/f> parameter; 1 is full-step mode and 0 is half-step mode.

The RAMP command parameters are expressed in terms of "steps". The user must equate "steps" to radians, degrees, or revolutions, taking into account the number of degrees per step of the motor being used and the mode (half- or full-step). For example, if the motor being used has 200 steps per revolution (1.8 degrees per step) and the mode is set to half step, then each requested step will cause the motor to turn 0.9 degrees; in this case, 100 steps per second = 90 degrees per second (or 15 RPM).

A "dual-sloped" trapazoidal velocity profile is generated by the RAMP command according to the acceleration and velocity parameters. The first and last 10 steps of a step sequence ramp the velocity at the rate requested by the acceleration parameter. The acceleration for the remaining ramping steps is approximatly 80% of the request acceleration. The two part acceleration on the ramp-up and ramp-down give rise to the "dual-sloped" velocity profile. This feature accounts, to some degree, for the decreasing amount of torque available as the speed increases.

For more information on the stepper motor card see appendex E.

Examples:

```
2000 PRINT #1:"ramp a2 25000 300 500 1
```

The step mode is set to 0 (half-step) and the direction is set to 1 (clockwise). Then 25,000 step pulses are sent, accelerating at a 500 steps per second squared rate until the maximum velocity of 300 steps per second is reached. Since the mode is half-step, the number of steps, the velocity and acceleration are motor half-steps.

```
1000 PRINT #1:"ramp a5 10000 100 200;"
```

10,000 steps are output to the stepper motor card in slot 5. The steps will ramp up to 100 steps per second at a rate of 200 steps per second squared and will also ramp down at the same rate. The direction and mode remain as they were.

Read Count Input

Read contents of a counter

Syntax: RCI <chan> [<rflag> [<bufX>]];

where: <chan> number of channel data is comming from
<rflag> reset flag
<bufX> buffer to store count in

Description

READ COUNT INPUT reads the the number of TTL edges that have occured on the specified channel. The count must have been started by the SCI command. RCI will read the contents of a counter with out altering the counter if <rflag> is a 0. If <rflag> is a 1 then the counter will be reset to 0 after the contents have been read. No matter what the value of <rflag>, the counter will continue to count until the ECI command is given.

The value read from the counter will be sent to the OUT device unless a <bufX> is specified. The buffer may be either a fifo or regular buffer. The maximum input frequency is 5 MHz. If a 16 bit count was specified in the SCI command then <bufX> must be defined to be type 2. If a 32 bit count was used the <bufX> must be defined to be type 3.

This command is to be used with the Pulse Frequency card (R8B2). Channels C and D are used as inputs.

<rflag> is interpreted according to the following table:

<rflag>	explanation
0	do not reset counter after reading
1	reset count to 0 after reading

Default Value

<rflag> = 0
<bufX> = Send reading to OUT Device

Examples:

```
1000 PRINT #1:"dfn 1 100 3, 5 100 5; fifo 1
1010 PRINT #1:"load 5; rci c1 1 1; end;"
1020 PRINT #1:"sci c1 0 1;"
1030 PRINT #1:"spn 5 5;"
1040 INPUT #1:pnum
1050 temp = 1
1060 DO WHILE temp = 1
1070     PRINT # 1:"ifpr ";pnum;" ; gc;"
1080     INPUT #1: temp
1090 LOOP
1100 PRINT #1:"ECI c1;"
```

A 32 bit count of rising TTL edges is started on channel c1. The counter is read into a fifo buffer every second for five seconds. Then the count is ended with the ECI command.

Read

Load a buffer with ASCII data

Syntax: RD <bufA>;

format: <size> <point> [<point>] [...]

where:
<bufA> name of buffer to input
<size> the number of points in the buffer
<point> the value of a point

Description

READ causes MDAS to receive ASCII data and store it in the buffer identified by <bufA>. The first ASCII number that is input is <size> and it indicates the number of points that follow. The source must send the number of points specified by <size>. Each data point is converted to the data type that was defined for <bufA>.

The source may use any ASCII character as a delimiter except '0 to 9', '.', '+', '-', 'E' or 'e'. The Carriage Return character may be used as a delimiter and must be used at the end of data transmission.

Examples:

```
630 PRINT #1:"rd 3;"
```

Buffer 3 is loaded with data from the host. The first value sent from the host is the number of data points that follow.

Read Disk

Read a file from floppy disk and put it in a buffer

Syntax: RDD <buff> <filename>;

where: <buff> buffer to put data in
<filename> name of file to read

Description

READ DISK reads a file from floppy disk and stores it in a MDAS buffer. An error occurs if the buffer is not big enough to hold the file. If an executable buffer is loaded it must already be defined to be an executable buffer (See DFN). If there is no internal disk drive, a COMMAND error will occur.

Filenames consist of one to eight characters, except for the characters . [ ] ? \ = * : ; - < >. These twelve characters are the only ones that cannot be used. Additionally, a file extension can be added to the file name. A file extension is one to three valid characters and is separated from the file name by a period. A floppy disk file can be read only if it is stored using the format described in the WRD command.

Examples:

```
970 PRINT #1:"dfn 3 200 5;"
980 PRINT #1:"rdd 3 test.run"
990 PRINT #1:"run 3;"
```

Buffer 3 is loaded with a test program and executed one time.

Read Pairs

Load two buffers with pairs of ASCII data

Syntax: RDP <bufA> <bufB>;

format: <size> <point> <point> [<point> <point>] [...]

where:
<bufA,B> names of buffers to fill
<size> the number of pairs of points to be sent
<point> the value of a point

Description

READ PAIRS causes MDAS to fill two buffers with the points sent. Every other point is placed in <buffer A> starting with the first data point after <size>; the other data points are placed in <buffer B>. <Size> specifies the number of points to follow. Each data point is converted to the data type that was defined for the input buffers.

The source may use any ASCII character as a delimiter except '0 to 9', '.', '+', '-', 'E' or 'e'. The Carriage Return character may be used as a delimiter and must be used at the end of data transmission.

Examples:

```
510 PRINT #1:"rdp 1 2;"
```

Buffers 1 and 2 are loaded with data pairs from the host. The first value sent is the number of data pairs that follow.

Set Analog

Sets analog output channel(s) to voltage values received

Syntax: SA <chanA> <volt> [<chanN> <volt>] [...];

where: <chanA,N> name of channel to receive data
<volt> the voltage to be output

Description

SET ANALOG performs a digital to analog conversion on each voltage value and sets the specified channel to that voltage. The data is sent in ASCII format. A given channel may be specified more than once and the total number of channels that can be specified is 64.

There are no trigger options with this command; the data conversion begins as soon as the command is received.

The time elapsed between samples is not fixed because the timing module is not used to control the acquisition of the data. Roughly, the time between samples is between 20E-6 and 80E-6 seconds.

Examples:

```
260 PRINT #1:"sa a3 2.2 b3 -4.75;"
```

Channel a3 is set to 2.2 Volts and channel b3 is set to -4.75 Volts.

Scan Analog Channels

Reads analog data and stores it in 1 buffer

Syntax: SCAN <reps> <buf> <chan> <gain> [<chan> <gain>] [...];

where:		
	<reps>	number of repetitions of the sequence
	<buf>	name of buffer to receive data
	<chan>	name of channel to take data from
	<gain>	gain factor for channel

Description

SCAN performs an analog to digital conversion on each channel specified in the command and stores the digital data in the specified buffer. Before each conversion is performed, the internal gain is set to the value specified by <gain>. The values of amplification are 1, 2, 4, 8, 16, 32, 64 and 128.

The buffer that is intended to be used with this command must be defined either as floating point or word integer (type 2 or 4). For a buffer that is defined as floating point, the value stored in the buffer is divided by the gain. For word integer buffers, the data is not gain adjusted. Stored values in integer buffers have a range of -32768 to 32767.

Conversion ends when the number of repetitions (<reps>) of the specified sequence are taken. If <reps> contains a decimal point, the portion to the right of the decimal point represents the number of sequences to retain before the trigger becomes true.

The same channel may be specified more than once in the list; if this is done, alternate samples from that channel will be placed in the buffer.

If the period specified by the SET PERIOD command is not long enough to take the requested number of channels in the sequence, a "SPEED" error will be generated.

The minimum period that can be used depends on the number of channels being scanned and the type of trigger. For trigger type 0 (no trigger) the minimum period is (n)*4.0E-6 + 3.6E-6 seconds. For trigger types 1,2,3,4,7, and 8 the minimum period is (n + 1)*4.0E-6 + 4.8E-6 seconds. For trigger types 5 and 6 the minimum period is (n + 2)*4E-6 + 4.8E-6 seconds. (n) in the previous equations is the total number of channels in the sequence. The minimum period equations do not change if pre-sample points are requested.

If trigger type 4 is used, data is sampled at a rate determined by the external trigger (See the TRI command). However the minimum time between samples must conform to the minimum times described above.

For a sequence of just one channel and no pre-trigger samples, there are four minimum period values depending on the chosen parameters. If the trigger type is 1-3 or 5-8 and the number of samples is less than 65535 the minimum period is 2.6E-6 seconds. If the trigger type is 1-3 or 5-8 and the number of samples is greater than 65535 the minimum period is 3.6E-6 seconds. If the trigger type is 4 the minimum period is 12.8E-6 seconds. If the trigger type is 0 the minimum period is 1.6E-6 seconds.

If the trigger type is 0, the period is less than 2.6E-6 and the buffer type is 2 (word integer), the buffer size must be defined 2 points larger than the desired number of points.

All channels in a multiple channel sequence are sampled once each period; the time interval between samples is 4.0E-6 seconds. The channels in a sequence are sampled in the order that they appear in the command.

The preceeding vaules for minimum periods are only valid for 12-bit converter cards. If a 16-bit converter is used the minimum period is increased. The time interval between samples is 25E-6 for the 16-bit converter card. Thus, to calculate the minimum period for a sequence of channels, simply substitute 25E-6 for 4.0E-6 in the preceeding equations.

All channels that have the sample and hold option are put in the hold mode at the beginning of each repetition of the sequence and are returned to sample mode after all the channels in the sequence have been read. The same amount of time is required to read these channels as those that do not have the sample and hold option.

Examples:

```
100 PRINT #1:"scan 235 3 a1 2 b1 8"
110 PRINT #1:" c1 128 d1 32;"
```

After the trigger condition is satisfied, channel a1 is read with the gain set at 2, channel b1 is read with the gain set to 8, channel c1 is read with the gain set to 128 and channel d1 is read with the gain set to 32. The data samples are all put into buffer 3 in the order in which they are read. This sequence is repeated until a total of 235 sequences have been executed.

```
140 PRINT #1:"scan 3000.200 3 c1 1 d1 2;"
```

Channel c1 is read with the gain set to 1 and channel d1 is read with the gain set to 2. The data is placed into buffer 3. This sequence is repeated
continuously keeping the latest 200 sequenc trigger condition is satisfied, the sequence continues, with the readings being placed in the remaining buffer space until a total of 3000 sequences have been saved in the buffer.

Start Count Input

Starts a Count of TTL Pulses

Syntax: SCI <chan> [<type> [<lflag>]];

where: <chan> number of channel to take data from
<type> the direction of edge to count on
<lflag> flag for 16 or 32 bit count

Description

START COUNT INPUT initializes a counter to start counting TTL compatible inputs that satisfy the <type> condition. The <lflag> determines if a 16 or 32 bit count is used. The SCI command is similar to the CI command except that once the SCI command is issued, MDAS is free to do other things. The count can be read using the RCI command or ended using the ECI command. This command is to be used with the Pulse Frequency card (R8B2). Channels C and D are used as inputs.

The maximum input frequency is 5 MHz. and the highest number that can be counted using 16 bits is 65535. The highest number that can be counted using 32 bits is 4,294,967,295.

<type> and <lflag> are interpreted according to the following table:

<type>	explanation
0	each rising edge of input counted
1	each falling edge of input counted

<lflag>	explanation
0	16 bit count
1	32 bit count

Default Values

<type> = 0
<lflag> = 0

Examples:

```
1260 PRINT #1:"sci d1 0 1;"
```

A 32 bit count of rising TTL edges is started on channel d1. This continues until a ECI command is issued.

Set Digital

Sets digital channel(s) to values received

Syntax: SD <chanA> <val> [<chanN> <val>] [...];

where: <chanA,N> name of channel to take data from
<val> value of the digital data

Description

SET DIGITAL sets the output of the specified channel to the value sent for that channel. The data is sent in ASCII format and can be either '0' or '1'. A '0' will output a low level TTL signal and a '1' will output a high level TTL signal. A given channel may be specified more than once and the total number of channels that can be specified is 64.

There are no trigger options with this command; the data conversion begins as soon as the command is received.

The time elapsed between samples is not fixed because the timing module is not used to control the acquisition of the data. Roughly, the time between samples is between 20E-6 and 80E-6 seconds.

Examples:

```
730 PRINT #1:"sd a6 0 b6 1;"
```

Channel a6 is set to a low-level TTL signal (0 Volts) and channel b6 is set to a high-level TTL signal (5 Volts.)

Service Request

Assert the GPIB Service Request line until a serial poll is performed.

Syntax: SRQ;

Description

The Service Request line on the GPIB bus is asserted until a serial poll is performed. When MDAS is addressed during a serial poll a status byte will be returned and the Service Request line will be deasserted. The value of the status byte is -31 (DF hexa-decimal). This informs the GPIB controller that MDAS is requesting service.

Examples:

```
390 PRINT #1:"dfn 1 100 5;"
400 PRINT #1;"load 1;"
410 PRINT #1;"labl begin;"
420 PRINT #1;"ifa a1 < 5;bra begin;"
430 PRINT #1;"srq;end;"
440 PRINT #1;"run 1;"
```

Executable buffer 1 is loaded with a program which loops testing the value on channel a1 until the value is above 5 at which time the srq instruction will be executed.

Write

Output a buffer in ASCII format

Syntax: WR <bufA> [<size> [<offset> [<interl>]]];

format: <size> <dlm> [<point> <dlm>] [...]

where:
<bufA> name of buffer to output
<size> the number of points to be sent
<offset> offset from beginning of <bufA>
<interl> interleave factor
<point> the value of a point
<dlm> delimiter, either <CR> or <CR> <LF>

Description

WRITE causes MDAS to output the contents of a buffer to the port assigned by the Output Device command. Optionally, a portion of a buffer may be sent by specifying the size in the command. In this case, the first N points of the buffer are sent (N = <size>). If the size of the buffer is 0, then no <points> are sent but <size> and <del> are sent. If the <offset> parameter is included, the first point that is sent is an offset from the beginning of the buffer. The default value for <offset> is 0 (which is the beginning of the buffer). If the <interl> parameter is included, every <interl> point is sent to the OUTput device. For example if <interl> is 5 then every fith point will be sent. The default value for <interl> is 1, which will send every point in sequence.

The first line output is the number of points that will follow. This number is in ASCII format. Each subsequent output by MDAS contains a single point followed by a Carriage Return (and Line Feed, if the port is set for Line Feeds). MDAS terminates the output when the number of points output in the first line have been sent. The number of points and all of the values are sent in ASCII format.

If the buffer that is written is type 6 then the first line output is the number of strings to be output. A string is a series of ASCII characters followed by a carriage return or carriage return/line feed. Each subsequent output is a string. The <size> parameter is ignored if a type 6 buffer is being output.

The WRITE command is also used to extract data from a fifo buffer (see the Fifo command). If the <size> parameter is left off, it defaults to 1 and if the fifo buffer is empty, the WRITE command waits until a point is loaded into the buffer.

Examples:

```
390 PRINT #1:"wr 2;"
```

The contents of buffer 2 are sent to the host. The first value sent is the number of points that follow.

Write Binary

Output a buffer in Binary format

Syntax: WRB <bufA> [<size> [<offset> [<interl> [<order>]]]];

format: <high byte> <low byte> [...] [<high byte> <low byte with EOI>]

where:

<bufA>	name of buffer to output
<size>	the number of points to be sent
<offset>	offset from beginning of <bufA>
<interl>	interleave factor
<order>	order of bytes in an integer buffer transfer
<high byte>	high byte
<low byte>	low byte

Description

WRITE BINARY causes MDAS to output the contents of a buffer to the GPIB port. The GPIB address is set up using the LED display and buttons in the back of MDAS. WRITE BINARY only uses the GPIB to output the contents of a buffer and does not use the port assigned by the OUTput Device command. Optionally, a portion of a buffer may be sent by specifying the size in the command. In this case, the first N points of the buffer are sent (N = <size>). If the size of the buffer is 0, then an error will generated. If the <offset> parameter is included, the first point that is sent is an offset from the beginning of the buffer. The default value for <offset> is 0 (which is the beginning of the buffer). If the <interl> parameter is included, every <interl> point is sent to the OUTput device. For example if <interl> is 5 then every fith point will be sent. The default value for <interl> is 1, which will send every point in sequence.

Each point in an integer buffer is sent as two bytes. The first byte that is sent is the high byte of the integer and the second byte is the low byte. The format of the two byte integer is 2's complement. If <order> is a 1 then the low byte is sent first followed by the high byte. The <order> parameter will only affect integer buffer transfers. MDAS terminates the output when all the bytes have been sent. An EOI is sent with the last byte to signal that the transmission is complete.

The WRITE BINARY command is also used to extract data from a fifo buffer (see the Fifo command). If the <size> parameter is left off, it defaults to 1 and if the fifo buffer is empty, the WRITE BINARY command waits until a point is loaded into the buffer.

MDAS can maintain a transfer rate of 110K bytes per second.

Examples:

```
390 PRINT #1:"wrb 8 5000;"
```

The first 5000 points of buffer 8 are sent over the GPIB, in binary format, to the host. An EIO is sent with the last byte to signal that the transmission has completed.

Write Disk

Writes a data or executable buffer to the floppy disk file in ASCII format

Syntax: WRD <buff> <filename>;

where: <buff> buffer to write to floppy disk
<filename> name of floppy diks file to write

Description

WRITE DISK writes a buffer to a floppy disk file in ASCII format. If there is no internal disk drive a COMMAND error will occur.

Filenames consist of one to eight characters, except for the characters . [ ] ? \ = * : ; - < >. These twelve characters are the only ones that cannot be used. Additionally, a file extension can be added to the file name. A file extension is one to three valid characters separated from the file name by a period. The data written to the floppy disk file will be organized as follows.

<number of points in ASCII format> <dlm>
<type of buffer in ASCII format> <dlm>
<data in ASCII format> <dlm>
.
.
.

where <dlm> is <CR> <LF>.

Examples:

```
180 PRINT #1:"wrd 1 data.4"
```

Buffer 1 is written to disk and stored as a file named data.4 .

Write Pairs

Output two buffers as pairs of ASCII data

Syntax: WRP <bufA> <bufB> [<size>];

format: <size> <dlm> <point> <point> <dlm> [<point> <point> <dlm>] [...]

where:
- <bufA,B> names of buffers to fill
- <size> the number of pairs of points to be sent
- <point> the value of a point
- <dlm> delimiter, either <CR> or <CR> <LF>

Description

WRITE PAIRS causes MDAS to output the contents of two buffers to the port assigned by the Output Device command. Optionally, a portion of each buffer may be sent by specifying the size in the command. In this case, the first N points of each buffer are sent (N = <size>).

The first line output contains the number of pairs of points that will follow. Each subsequent output by MDAS contains two points separated by a space and followed by a Carriage Return (and Line Feed, if the port is set for Line Feeds). The odd numbered points are sent from <buffer A> and the even numbered points are sent from <buffer B>. MDAS terminates the output when the number of pairs of points output in the first line have been sent. The number of points and all of the values are sent in ASCII format.

Examples:

```
220 PRINT #1:"wrp 1 2 512;"
```

The first 512 data pairs in buffers 1 and 2 are sent to the assigned output port; the first data point in each pair comes from buffer 1.

Branch on Condition

Branches in an executable buffer if the condition flag is set to 1

Syntax: BRA <label>

where: <label> any alphanumeric sequence

Description

BRA continues program execution at the specified label if the condition flag is true. Program execution is continued at the next instruction if the condition flag is false. A label can be any sequence of 1 to 7 alphanumeric characters. The destination of the BRA command must be defined with the LABL command. The BRA command can only be used in an executable buffer.

Examples:

```
690 PRINT #1:"dfn 60 100 5;"
700 PRINT #1:"load 60;"
710 PRINT #1:"labl begin;"
720 PRINT #1:"ifa d12 <= 3.0;"
730 PRINT #1:"bra begin; end;"
```

Executable buffer 60 is loaded with a program that tests the value on channel d12 until the value is greater than 3.0.

Calibrate Bridge

Enters bridge parameters for linearization

Syntax: CALB <chan> <K> <GF> [<CF>];

where: <chan> name of channel
<K> bridge configuration constant
<GF> gage factor
<CF> calibration factor

Description

CALIBRATE BRIDGE enters the linearization constants to be used for the specified bridge input channel. CALB should be executed when the bridge is in the quiescent state.

K is a constant that identifies the type of bridge configuration being used. GF is the gage factor and is a property of the sensing resistors. CF is the calibration factor that is the amount of change expected for the given sensing and calibration resistors.

The bridge configuration constant indicates the bridge configuration being used. R1 is the resistor located between the positive exitation voltage and the input node and R2 is between the input node and ground. The valid values of K are:

1	full bridge
2	half bridge
-4	quarter bridge, R1 active
+4	quarter bridge, R2 active

The gage factor is a property of the sensing resistor and indicates the amount of change in resistance for a given change in the parameter of interest. For a strain gage,

GF = (deltaR / R) / (deltaL / L);

deltaL / L is by definition, strain.

The calibration factor indicates the fractional change in a sensing resistor that the calibration resistor should cause. This is used to calibrate the gain of the bridge amplifier and is expressed in parts per million. CF can be determined by the formula:

CF = Rs*1000000/(Rs+RC)

where Rs is the sensing resistor (R1 or R2) and RC is the calibration resistor.

The bridge measurement technique uses the ratio of the input voltage to the excitation voltage. Measurement of strain is done by taking the difference between the unstrained input to excitation ratio and the strained input to excitation ratio. Calling this difference of ratios D, the linearization equations for measured strain (e) in the different bridge configurations are:

quarter e = -K*D / (GF*(1+2*D))

half, full e = -K*D / GF.

One advantage to using this technique is that the bridge does not have to be balanced. If another bridge measurement technique is required, the bridge channel can be defined as an analog input channel and can be read like any other channel. Additional information on the bridge input card can be found in Appendix C.

Examples:

```
400 PRINT #1:"calb a3 -4 2.02 1424.3;"
```

Channel a3 is a quarter bridge with R1 the active resistor. The gage factor of the sensing resistor is 2.02. The calibration resistor is 100516 ohms and the sensing resistor is 120.6 ohms; these values give 1424.3 as the calibration factor.

Clear Condition

Clears the Condition Flag

Syntax: CLC;

Description

The CLEAR CONDITION command clears the conditional flag that contains the result of the last conditional test.

Examples:

```
610 PRINT #1:"clc;"
```

The conditional flag is cleared.

Clear Error

Clears the last error number

Syntax: CLE;

Description

The CLEAR ERROR command clears the last error number.

Examples:

```
610 PRINT #1:"cle;"
```

The last error number is cleared. If the LED display is set as the error device, the error message that accompanies the error number is not cleared. The error message is cleared by executing a GET ERROR command.

Command Device

Sets the port for the device that will receive commands

Syntax: CMD <port>;

where: <port> the port that MDAS will accept commands from

Description

COMMAND DEVICE defines which port will receive the commands. <port> corresponds to one of the following devices:

<port>	port identified
1	PORT 1
2	PORT 2
3	GPIB

Examples:

```
940 PRINT #1:"cmd 3;"
```

The GPIB port is set as the command port.

Define Buffer

Assigns memory space to a buffer

Syntax: DFN <bufA> <size> <type> [<bufn> <size> <type>] [...];

where: <bufA,n> name of buffer(s) to define
<size> number of points in buffer
<type> number of bytes per sample

Description

DEFINE BUFFER allocates memory for data buffers that can be used with the conversion, math and data transfer commands. Valid buffer names are the integers 1 through 64. Each buffer is created according to the <size> and <type> attributes. If a buffer already exists, it is cleared and given the new <size> and <type>. <type> is interpreted according to the following table:

<type>	explanation
1	byte integer (1 byte)
2	word integer (2 bytes)
3	long-word integer (4 bytes)
4	floating point (4 bytes)
5	executable buffer
6	ASCII for errors or data output

The buffer is not actually created until an attempt is made to write data into it.

Default Values

<buffer> = 1
<size> = 600
<type> = 2

Examples:

```
400 PRINT #1:"dfn 1 10000 2 3 5000 4;"
```

Buffer 1 is defined for 10,000 points of word integer data and buffer 3 is defined for 5,000 points of floating point data.

Define Channel

Defines a channel for external gain or linearization

Syntax: DFNC <chan> <type> [<gain>];

where: <chan> name of channel to define
<type> number indicating type of channel
<gain> gain external to converter card

Description

DEFINE CHANNEL is used to identify channels with external gain or channels that require linearization. External gain refers to the gain on an I/O card and any other amplification outside of the MDAS system. Internal gain is specified by the ANALOG INPUT command and refers to the amplifiers on the converter card.

The target buffer for any channel with external gain and any channel requiring linearization must be a floating point buffer (buffer type 4). See DEFINE BUFFER command. If the target buffer is not a floating point buffer the data remains unmodified.

The following table identifies the type number with the input type.

<type>	explanation
0	normal with external gain
1	E type thermocouple
2	J type thermocouple
3	K type thermocouple
4	undefined
5	T type thermocouple
6	solid state temperature sensor
7	bridge input

After data is entered into a buffer from a channel that requires thermocouple linearization, the data is manipulated using built in tables to yield degrees Centigrade. Data from a solid state temperature sensor is scaled to yield degrees Centigrade.

The bridge type input requires additional parameters that are entered using the CALIBRATE BRIDGE command; a channel must be defined as a bridge input using the DFNC command before the CALB command can be executed for that channel. Only channels 'a' and 'e' are valid bridge channels. The bridge input returns values expressed as parts per million (ppm), corresponding to the changes in resistance measured by the bridge.

At power-up all channels default to type = 0 and gain = 1. If the gain parameter is not included, the following are the default gains for the different types:

<type>	default gain
0	1
1-5	100
6	1
7	100

Examples:

```
400 PRINT #1:"dfnc d3 0 500;"
```

Channel d3 has an amplifier that is set to a gain of 500. All readings from this channel will be automatically divided by 500.

```
350 PRINT #1:"dfnc a1 2;"
```

Channel a1 is a thermocouple channel using a J type of thermocouple. The gain of the I/O card amplifier is set to the default value of 100. The readings from this channel will be linearized according to the J type tables.

Delete Buffer

Deletes a buffer from MDAS memory

Syntax: DEL <bufA> [<bufn>] [...];

where: <bufA,n> buffer(s) to delete from the MDAS memory

Description

DELETE BUFFER deletes one or more buffers from the MDAS memory. If the buffer didn't exist, DELETE BUFFER does nothing.

Examples:

```
720 PRINT #1:"del 1 2 3;"
```

Buffers 1, 2 and 3 are deleted from memory.

Error Device

Sets the port for the device to which errors messages are sent

Syntax: ERR <port> [<buffer> [<append flag>]];

where: <port> the port that MDAS will send error messages to
<buffer> buffer to output error messages to if port is 1
<append flag> append or not to a buffer

Description

ERROR DEVICE defines which port to send the error messages to. <port> corresponds to one of the following ports:

<port>	port identified
0	Switch/Display panel
1	Internal Buffer

If port 1 is used an internal buffer of type 6 must have been previously defined using the DeFiNe buffer command. All output data will then be redirected to this buffer. The data in the internal buffer can be output by changing the output device to 1, 2, or 3 and using the WRite buffer command. The <append flag> specifies whether output data is appended at the end of the specified buffer or whether the buffer is reset so that any data that was in the buffer is deleted. <append flag> corresponds to the following table:

<append flag>	explanation
0	no append
1	append

Default Values

<append flag> = 0

Examples:

```
670 PRINT #1:"err 1 10 0;"
```

Buffer 10 is cleared and set as the error port.

End

Terminates loading of an executable file

Syntax: END;

Description

END is used to close an executable buffer that was opened with a LOAD command. All commands received after the END command are executed as they are received.

Examples:

```
500 PRINT #1:"load 4;"
510 PRINT #1:"ai 30000 a3 8 1;"
520 PRINT #1:"sca a3 a3 2.25 1;"
530 PRINT #1:"end;"
540 PRINT #1:"run 4;"
```

Buffer 4 is opened to receive commands. The commands sent in line 110 and 120 are not executed but placed in buffer 4. Line 130 closes buffer 4 and line 140 causes the commands stored in buffer 4 to be executed.

Fifo Buffer

Tags a previously defined buffer as a fifo buffer

Syntax: FIFO <bufA> [<bufn>] [...];

where: <bufA,n> name of buffer(s) to tag as fifo

Description

FIFO tags a previously defined buffer as a fifo buffer (see Define Buffer command). Fifo stands for "First In, First Out". A buffer that is tagged as a fifo buffer can be written to and read from asynchronously. Using fifo buffers in the background mode (see Spawn command) allows MDAS to process a very large amount of data over a relatively long period of time.

The WRITE command is used to extract data from a fifo buffer. If the <size> parameter is left off, it defaults to 1 and if the fifo buffer is empty, the WRITE command waits until a point is loaded into the buffer.

When a buffer is tagged as a fifo buffer, that buffer is cleared of any data.

Examples:

```
400 PRINT #1:"dfn 1 10000 2 3 5000 4;"
410 PRINT #1:"fifo 1 3;"
```

In line 400 buffer 1 is defined for 10,000 points of word integer data and buffer 3 is defined for 5,000 points of floating point data. Line 410 tags both buffers as fifo buffers.

Format Disk

Format a floppy disk

Syntax: FMT;

Description

FORMAT formats a floppy disk. The format used is 9 sector/track, double density, double sided. The format is the same used by the IBM-PC which allows MDAS files to be accessed by a PC compatible floppy drive. If MDAS does not have a floppy drive then a command error will occur.

Examples:

```
380 PRINT #1:"fmt;"
```

The floppy diskette in the MDAS floppy disk drive is formatted.

Goto Label

Branches unconditionally in an executable buffer

Syntax: GOTO <label>

where: <label> any alphanumeric sequence

Description

GOTO continues program execution at the specified label. A label can be any sequence of 1 to 7 alphanumeric characters. The destination of the GOTO command must be defined with the LABL command. The GOTO command can only be used in an executable buffer.

Examples:

```
690 PRINT #1:"dfn 4 100 5;"
700 PRINT #1:"load 4;"
710 PRINT #1:"labl repeat;"
720 PRINT #1:"ifa a1 < 3;"
730 PRINT #1:"bra stop;"
740 PRINT #1:"ifa a1 > 6;"
750 PRINT #1:"bra stop;"
760 PRINT #1:"goto repeat;"
770 PRINT #1:"labl stop;"
740 PRINT #1;"end;"
```

Buffer 4 is defined to be an executable buffer. Buffer 4 is loaded with a program which will continually test the value on channel a1. Program execution will stop when the value on a1 is not between 3 and 6.

If Error

Checks if an error has occured

Syntax: IFE;

Description

IF ERROR checks to see if an error has occured. If an error has occured then the condition flag is set to 1 otherwise it is cleared. The result of the test can be obtained using the Get Condition command.

Examples:

```
520 PRINT #1:"bad command;"
530 PRINT #1:"ife;"
540 PRINT #1:"gc;"
550 INPUT #1:c
```

A bad command is sent in line 520. Ife sets the condition flag to 1 because an error has occured. The result of the test is entered into c.

If Process Running

Checks if a spawned process is running

Syntax: IFPR <process num>;

where: <process num> number of the spawned process

Description

IF PROCESS RUNNING checks to see if a spawned process is still running. The <process num> parameter is the process number that was sent back when the SPAWN command was executed. The condition flag is set to 1 if the process indicated by <process num> is still running, otherwise the condition flag is set to 0. The result of the test can be obtained by using the GET CONDITION command. The test can also be used to create a conditional branch in an executable buffer (see BRA).

Examples:

```
330 PRINT #1:"ifpr ",NUM," ;"
340 PRINT #1:"gc;"
350 INPUT #1:RES
```

The process indicated by the variable NUM is tested to see if it is still running. The result of the test is obtained by the GET CONDITION command and stored in variable RES.

```
510 PRINT #1:"spn 4 10;"
520 INPUT #1:NUM
530 PRINT #1:"load 5;"
540 PRINT #1:"labl repeat;;"
550 PRINT #1:"ifpr ",NUM," ;"
560 PRINT #1:"bra repeat;"
570 PRINT #1:"gt;"
580 PRINT #1:"end;"
590 PRINT #1:"run 5;"
600 INPUT #1:T
```

Executable buffer 4 is run is background 10 times (See SPN command). The number of the spawned process is input into the variable NUM. Buffer 5 is loaded with commands used to create a while loop. The LABL, IFPR and BRA commands create a loop that continues until the spawned process NUM has finished. When the process has completed the time stored in variable T.

Input Device

Sets the port for the device that receives data

Syntax: INP <port>;

where: <port> the port that MDAS will accept data from

Description

INPUT DEVICE defines which port to receive input data from. <port> corresponds to one of the following ports:

<port>	port identified
1	PORT 1
2	PORT 2
3	GPIB

Examples:

```
330 PRINT #1:"inp 2;"
```

Port 2 (RS-232) is set as the input data port.

Increase Settling Time

Increases the time between samples

Syntax: INST [<iflag>];

where: <iflag> flag to increase settling time

Description

INCREASE SETTLING TIME lengthens the time between samples of a multiple channel input. INST command only affects the commands AI and SCAN. If iflag is 1 then the time between multiple samples is lengthened to 20e-6. If iflag is 0 then the time between multiple samples is specified in the command (AI uses 6.2e-6 and SCAN uses 4.0e-6). This command will only affect conversions using the 12 bit converter cards. The time between samples of a 16 bit converter card remains at 25e-6. The INST command should be used when doing inputs with thermocouple cards. The default value of iflag is 0.

Examples:

```
7730 PRINT #1:"inst 1;"
7740 PRINT #1:"ai 100 a1 1 1 b1 2 1;"
```

The time between samples of channels a1 and b1 is increased from 6.2e-6 (specified by the AI command) to 20e-6.

Kill Process

Kills a process (executable buffer) in background mode

Syntax: KILL <prcA> [<prcn>] [...];

where: <prcA,n> name of process(es) to kill

Description

KILL terminates a process that is running in background.

Examples:

```
900 PRINT #1:"kill 4;"
```

Process number 4 is terminated.

Label

Label used in executable files

Syntax: LABL <label>;

where: <label> label name

Description

LABL marks the position in an executable buffer where the GOTO and BRA commands transfer program control. This command can only be used while an executable buffer is being loaded. A label consist of 1 to 7 alphanumerics.

Examples:

```
700 PRINT #1:"load 4;"
710 PRINT #!:"labl begin;
720 PRINT #1:"ai 30000 a3 8 1;"
730 PRINT #1:"goto begin;"
740 PRINT #1:"end;"
```

Executable buffer 4 is loaded. When buffer 4 is run (see RUN) the AI command will be executed. When the GOTO command is executed program control will be transferred to the matching label command. This causes the AI command to be executed in an endless loop.

Load Commands

Opens a buffer to receive MDAS commands

Syntax: LOAD <bufA>;

where: <bufA> the target buffer for commands

Description

The LOAD command opens a buffer to receive commands. The commands sent after the LOAD command are saved in the specified buffer and are not executed. The END command terminates the LOAD command and closes the executable buffer. The RUN command is used to execute the commands saved in an executable buffer.

Examples:

```
700 PRINT #1:"load 4;"
710 PRINT #1:"ai 30000 a3 8 1;"
720 PRINT #1:"sca a3 a3 2.25 1;"
730 PRINT #1:"end;"
740 PRINT #1:"run 4;"
```

Buffer 4 is opened to receive commands. The commands sent in line 110 and 120 are not executed but placed in buffer 4. Line 130 closes buffer 4 and line 140 causes the commands stored in buffer 4 to be executed.

Logical Trigger

Sets the logical trigger sequence used with trigger type 9

Syntax: LTRI <chan> <cond> <levlA> <logical> <chan> <op> <levlB>
[<logical> <chan> <cond> <levln>] [...];

where: <chan> the channel number to test
<cond> conditional operator
<levln> level used to compare against
<logical> logical operator

Description

LOGICAL TRIGGER sets the sequence of tests used to start A/D conversions. In order to use this command the trigger type defined by the TRI command must be type 9 (See TRI command). The <cond> operator determines the type of test made on each channel. The two types of <cond> operaters are ">" and "<", meaning greater than and less than, respectfully. The <levln> parameter indicates the value used for each comparison. The value read on each channel is compared with <levln> using the <cond> operator. The <logical> operator describes the combination of tests that will be used. The two possible logical operators are "&", which indicates the AND operator, and "|" which indicates the OR operator. There can be as many as 16 tests that can be made using the LTRI command. An explation of the possible values for some of the parateters follows:

<parameter>	valid values
<cond>	">" or "<"
<levln>	between -10 and 10
<logical>	"&" or "|"

Examples:

```
5630 PRINT #1:"tri 9;"
5640 PRINT #1:"ltri a1 > 5 | b1 > 4;"
```

The trigger is true when the value on channel a1 is greater than 5 Volts OR the value on channel b1 is greater than 4 Volts.

```
1320 PRINT #1:"tri 9;"
1330 PRINT #1:"ltri a1 < -3 & b1 < -5 | c
```

The trigger is true when (the value on channel a1 is less than -3 AND the value on channel b1 is less than -5) OR (the value on channel c1 is greater than 3 AND the value on channel d1 is greater than 5).

Output Device

Sets the port for the device that data is written to

Syntax: OUT <port> [<buffer> [<append flag>]];

where: <port> the port that MDAS will send data to
<buffer> the buffer to send data to if port is 4
<append flag> append or not to internal buffer

Description

OUTPUT DEVICE defines which port to output data to. <port> corresponds to one of the following ports:

<port>	port identified
1	PORT 1
2	PORT 2
3	GPIB
4	INTERNAL BUFFER

If port 4 is used an internal buffer of type 6 must have been previously defined using the DeFiNe buffer command. All output data will then be redirected to this buffer. The data in the internal buffer can be output by changing the output device to 1, 2 , or 3 and using the WRite buffer command. The <append flag> specifies whether output data is appended at the end of the specified buffer or whether the buffer is reset so that any data that was in the buffer is deleted. <append flag> corresponds to the following table:

<append flag>	explanation
0	no append
1	append

Default Values

<append flag> = 0

Examples:

```
750 PRINT #1:"out 1;"
```

Port 1 (RS-232) is set as the output data port.

Set Period

Sets the time between conversion command sequences

Syntax: PER <period>;

where: <period> the number of seconds between samples

Description

SET PERIOD sets the time between conversion command sequences. The period is expressed in seconds and ranges from 2.0E-6 seconds to 859 seconds. The actual period is a multiple of the period increment for the range that the requested period falls into. These values are shown in the following table:

period range in seconds		increment
0.000002 to	0.013107	0.0000002
0.013107 to	0.209712	0.0000032
0.209712 to	3.355392	0.0000512
3.355392 to	53.686272	0.0008192
53.686272 to	858.980352	0.0131072

The actual period can be calculated as follows:

A = I*INT(0.5+R/I)

where: A = Actual Period
R = Requested <period>
I = period increment for the range <period> falls into
INT is the integer function

Default Value

0.001 seconds

Examples:

410 PRINT #1:"per 1e-4;"

The period is set to 0.1 millisecond.

Restore Parameters

Restores parameters to default values

Syntax: RES;

Description

RESTORE PARAMETERS sets the parameters for the following functions to their default values:

TRIGGER	<type>=0 (no trigger)
PERIOD	0.001 seconds

Examples:

```
850 PRINT #1:"res;"
```

The trigger and period parameters are restored to their default values.

Run

Executes the commands in a buffer

Syntax: RUN <bufA>;

where: <bufA> executable buffer

Description

The RUN command begins execution of MDAS commands that were previously loaded into the specified executable buffer.

Examples:

```
300 PRINT #1:"load 4;"
310 PRINT #1:"ai 30000 a3 8 1;"
320 PRINT #1:"sca a3 a3 2.25 1;"
330 PRINT #1:"end;"
340 PRINT #1:"run 4;"
```

Buffer 4 is opened to receive commands. The commands sent in line 110 and 120 are not executed but placed in buffer 4. Line 130 closes buffer 4 and line 140 causes the commands stored in buffer 4 to be executed.

Set Interrupt Period

Sets the time between process interrupts

Syntax: SINT [<period>];

where: <period> the number of seconds between interrupts

Description

SET INTERRUPT PERIOD sets the time between process interrupts. This period is used in conjunction with the <mult> parameter of the Spawn command to determine the time between background process activations.

The period given in the command is rounded off to the nearest valid period. The valid periods are given in the following list:

0.1221E-3 seconds
0.2441E-3 seconds
0.4883E-3 seconds
0.9766E-3 seconds
1.9531E-3 seconds
3.9063E-3 seconds
7.8125E-3 seconds
15.6250E-3 seconds
31.2500E-3 seconds
62.5000E-3 seconds
125.0000E-3 seconds
250.0000E-3 seconds

n * 0.5 seconds
n=1 to 65535

Default Value

0.5 seconds

Examples:

```
410 PRINT #1:"sint 0.1;"
```

The process interrupt time is set to 0.125 seconds, the nearest valid period.

```
550 PRINT #1:"sint 12;"
```

The process interrupt time is set to 12 seconds.

Spawn Process

Spawns a process (executable buffer) in background mode

Syntax: SPN <bufA> [<reps> [<mult>]]

where: <bufA> name of executable buffer
<reps> number of repetitions
<mult> multiples of interrupt period

Description

SPN runs an executable buffer in background mode. Background mode can be used to gather data at a relatively slow rate but for long periods of time. The background mode allows MDAS to perform other tasks in foreground (normal running mode) while still running the background routines.

A SPAWNed process (the executable buffer) is activated periodically. The <reps> parameter indicates the number of times the process is to be run. The time interval between each running of the process is controlled by the <mult> parameter and the interrupt period (see Set Interrupt command). The Set Interrupt command sets the time between interrupts; this time is multiplied by <mult> and becomes the process activation period. Both <reps> and <mult> must be integers greater than 0.

When a process is SPAWNed, one activation period elapses before the first running of the process. When the SPAWN command is executed, a process number is sent back to the host. This number is used to terminate the process using the KILL command.

Default Values

<reps> = 1
<mult> = 1

Examples:

```
400 PRINT #1:"spn 4;"
410 INPUT #1:P
```

The process described by executable buffer 4 is spawned. The process number is stored in variable P. The number of repetitions and interrupt multiples both default to 1.

```
440 PRINT #1:"sint 0.25;"
450 PRINT #1:"spn 6 40 8; spn 7 125 3;"
460 INPUT #1:Q
470 INPUT #1:R
```

The interrupt period is set to 0.25 seconds in line 440. Then the processes in executable buffers 6 and 7 are spawned. The process in buffer 6 is activated every 2 seconds and is run 40 times. The process in buffer 7 is activated every 0.75 seconds and is run 125 times. The process number identifying executable buffer 6 is stored in variable Q and the process number identifying executable buffer 7 is stored in variable R.

Set Time

Set the real time clock to new date and time

Syntax: ST [<day> <mon> [<year>]] [<hour>:<min>:[<sec>]]

Description

SET TIME sets the MDAS real time clock to the value in <time>. The full date and time format is:

dd mmm yy hh:mm:ss

The month is expressed as a three letter abbreviation as follows: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC.

The following fields may be optionally omitted:

dd mmm yy
hh:mm:ss
yy
ss

If one of the above fields is omitted, the current value of that field is not changed.

Examples:

```
640 PRINT #1:"st 1 jan 85 8:00;"
```

The time is set to 8 AM January 1, 1985.

Set Trigger

Sets the type of trigger to be used.

Syntax: TRI [<type> <chan> [<levlA> [<levlB>]]];

where: <type> a digit between 0 and 8 indicating the type of trigger
<chan> the channel number to trigger on
<levln> levels used to determine trigger condition

Description

SET TRIGGER sets the type of trigger used to start A/D or D/A conversions. When type 5 or 6 is specified, <levlA> must be less than <levlB>. When type 7 or 8 is used, <levlA> is specified in volts per micro-second and the voltage difference can be calculated by multiplying <levlA> by the PERiod. When trigger type 9 is specified the parameters used are set up using the LTRI command. The trigger <type> is interpreted according to the following table:

<type> explanation

0 No trigger, commands executed immediately

1 Trigger when the sample values on <chan> exceed <levlA> for at least one period

2 Trigger when the sample values on <chan> fall below <levlA> for at least one period

3 TTL, falling edge triggered

4 TTL, each sample is taken on the falling edge of the trigger (Internal sample period is ignored)

5 Trigger when the sample values on <chan> fall between <levlA> and <levlB>

6 Trigger when the sample values on <chan> fall outside the range specified by <levlA> and <levlB>

7 Trigger when the change of a rising signal is greater than the value specified by <levlA>. <levlA> is specified in volts per micro-second.

8 Trigger when the change of a falling signal is greater than the value specified by <levlA>. <levlA> is specified in volts per micro-second.

9 Trigger when the sequence of logical trigger combination becomes true. (See the LTRI command).

If no parameters are included with the TRIGGER command, <type> is set to the default value.

Default Value

<type> = 0 (no trigger)
<level> = 0 Volts

Examples:

```
530 PRINT #1:"tri 2 a1 0;"
```

The trigger is true when the value on channel a1 is less than 0 Volts.

```
320 PRINT #1:"tri 3 d4;"
```

The trigger is true when a falling TTL signal is detected on channel d4, a digital input channel.

Unpack Buffer

Reassigns points in a buffer to other buffers

Syntax: UNPK <interl> <bufA> <bufX> [<bufn>] [...];

where:	<interl>	interleave factor
	<bufA>	buffer to unpack
	<bufX,n>	output buffers to store points in

Description

UNPACK BUFFER distributes points in a buffer to the specified output buffers. The first point in <bufA> is copied into the first position of <bufX>, the second point in <bufA> is copied into the first position of <bufn>, and so forth until <interl> number of points have been distributed. The sequence is repeated until all the points in <bufA> have been copied. The interleave factor <interl> specifies the number of channels that were scanned using the SCAN command.

The UNPACK BUFFER command enables data acquired using the SCAN command to be distributed to separate buffers. The points copied from <bufA> are stored in the output buffers in the same format as they are stored in <bufA> (ie. integer data <2 bytes> in <bufA> is stored in integer format in the output buffers).

If the number of receiving buffers is less than <interl>, the first <interl> number of points in each scan are distributed. If a certain channel is of interest, the ROTATE command can be used to move that channel to the beginning of the buffer. The UNPACK command will then extract only that channel of interest.

Examples:

```
350 PRINT #1:"dfn 1 300 4, 2 100 4;"
360 PRINT #1:"dfn 3 100 4, 4 100 4;"
370 PRINT #1:"scan 100 1 a5 8 b5 8 c5 8;"
380 PRINT #1:"unpk 3 1 2 3 4;"
```

Buffer 1 is defined for 300 points of floating point data and buffers 2, 3, and 4 are defined for 100 points of floating point data. A SCAN of channels a5, b5, and c5, each set at gain 8, are stored in buffer 1. The 300 points stored in buffer 1 are unpacked and copied to buffers 2, 3, and 4. Each buffer contains the points read on the specified channels (ie. buffer 2 contains the points read on channel a5, buffer 3 contains points from channel b5, and buffer 4 contains points from channel c5).

```
150 PRINT #1:"dfn 1 300 4, 2 100 4;"
160 PRINT #1:"scan 100 1 a5 2 b5 2 c5 2;"
170 PRINT #1:"unpk 3 1 2;"
```

Buffer 1 is defined for 300 points of floating point data and buffer 2 is defined for 100 points of floating point data. A SCAN of channels a5, b5, and c5, each set at gain 2, are stored in buffer 1. The UNPK command in line 170 says to unpack buffer 1 and put the result in buffer 2. Buffer 1 contains points from three channels (specified by <interl>) but only every third point is copied into buffer 2. So, buffer 2 contains the points that were input on channel a5 during the SCAN command and buffer 1 remains unchanged.

Unfifo Buffer

Restores fifo buffer to regular buffer

Syntax: UNFI <bufA> [<bufn>] [...];

where: <bufA,n> name of buffer(s) to restore

Description

The UNFIFO command restores a fifo buffer to a regular buffer. The data is not destroyed; the last item entered into the fifo buffer is located in the first position of the regular buffer.

Examples:

```
460 PRINT #1:"unfi 1 3;"
```

Buffers 1 and 3 are restored from fifo buffers to regular buffers.

Add

Adds two buffers together

Syntax: ADD <bufA> <bufB> <bufX>;

where: <bufA,B> input buffers
<bufX> buffer containing results

brief: X = A+B

Description

ADD adds each point in <bufA> to the corresponding point in <bufB> and places the result in the corresponding point in <bufX>. Any or all or the buffers may be the same. If any of the buffers are not the same length, ADD will continue until the end of the shortest buffer has been reached.

Examples:

```
450 PRINT #1:"add 1 2 8;"
```

The contents of buffer 1 are added to buffer 2 and the results are placed in buffer 8.

```
930 PRINT #1:"add 5 5 5;"
```

The contents of buffer 5 are doubled.

Add Complex

Adds two complex number buffers together

Syntax: ADD <bufA> <bufB> <bufC> <bufD> <bufX> <bufY>;

where:	<bufA,B,C,D>	buffers containing input data
	<bufX,Y>	buffers containing results
brief:	X = A+C	Y = B+D

Description

ADD COMPLEX adds each point in <buffer A> to the corresponding point in <buffer C> and places the result in <buffer X>. Then it adds each point in <buffer B> to the corresponding point in <buffer D> and places the result in <buffer Y>. Any or all of the buffers may be the same, except for the two output buffers. If any of the buffers are not the same length, ADD COMPLEX will continue until the end of the shortest buffer has been reached.

Examples:

```
640 PRINT #1:"add 1 2 4 5 1 2;"
```

The complex numbers in buffers 1 and 2 are added to the complex numbers in buffers 4 and 5; the resulting complex numbers are stored back in buffers 1 and 2, overwriting the former contents.

Average

Returns the average value of the buffer

Syntax: AVE <bufA> <num>;

format: <average value> <dlm>

where: <bufA> buffer whose average value is calculated
<num> number of points to average
<dlm> delimiter, either <CR> or <CR> <LF>

Description

AVE computes the average value of the specified buffer and sends that value out the port selected by the "OUTput device" command. The average value is followed by a Carriage Return and optionally a Line Feed if the output port was so defined. (See the Output Device command.)

If the buffer is a regular buffer, the first <num>ber of points are averaged; if the buffer is a fifo buffer, the remaining <num>ber of points are averaged. If the fifo buffer doesn't have the required number of points, the command waits until there are enough points.

Examples:

```
300 PRINT #1:"ave 3;"
310 INPUT #1:B
```

The average value of buffer 3 is put into variable B.

Decibels

Converts a buffer to decibels

Syntax: DB <bufA> <bufX>;

where: <bufA> buffer containing input data
<bufX> buffer containing results

brief: X = 10*LOG(A)

Description

DECIBELS converts the data in the input buffer to decibels and places the resulting data in the output buffer.

A value of 0 in the input buffer returns the largest negative number (-9.22337177E+18) and does **not** issue a "MATH" error.

Examples:

```
490 PRINT #1:"db 3 3;"
```

The data in buffer 3 is converted to decibel representation.

Differentiate

Performs a differentiation on a buffer

Syntax: DIF <bufA> <bufX>;

where: <bufA> buffer containing input data
<bufX> buffer containing output data

Description

DIFFERENTIATE sets each point in the output buffer to the difference between the corresponding point in input buffer and the next point in the input buffer. The output buffer has one fewer point than input buffer.

Examples:

820 PRINT #1:"dif 3 4;"

The result of differentiating the data is buffer 3 is placed in buffer 4.

Divide

Divides one buffer into another

Syntax: DIV <bufA> <bufB> <bufX>;

where: <bufA,B> buffers containing input data
<bufX> buffer containing results

brief: X = A/B

Description

DIVIDE divides each point in <buffer A> by the corresponding point in <buffer B> and places the result in the corresponding point in <buffer X>. Any or all or the buffers may be the same. If any of the buffers are not the same length, DIVIDE will continue until the end of the shortest buffer has been reached.

Division by 0 will give a "MATH" error and the size of the output buffer is set to 0 (empty buffer).

Examples:

```
970 PRINT #1:"div 1 2 1;"
```

The contents of buffer 1 are divided by the contents of buffer 2; the results are placed back in buffer 1.

FFT Real

Computes frequency domain from real time domain

Syntax: FFT <size> <bufA> <bufX> <bufY>;

where:		
	<size>	the size of the Fourier transform
	<bufA>	buffer containing input data
	<bufX,Y>	buffers containing output data

Description

FFT REAL produces a real Fourier transform of the data in <buffer A>. The real portion of the output is placed in <buffer X> and the imaginary portion of the output is placed in <buffer Y>. Even though, when the real transform is complete, the number of points in each output buffer is half of the size of the transform, <buffer X> must be defined to the size of the transform or larger; the additional buffer area is needed by the transform for intermediate calculations. <Buffer A> may be the same as one of the output buffers but the two output buffers cannot be the same. The <size> of the transform must be an integer between 8 and 65536, inclusive, and must be a power of 2. <Buffer A> must be defined at least as large as the size of the transform.

The formats of the data in the output buffer are discussed in Chapter 2.

Examples:

```
780 PRINT #1:"fft 1024 1 8 9;"
```

An FFT is performed on the contents of buffer 1; the results are stored in buffers 8 and 9, the real portion in 8 and the imaginary in 9.

FFT Complex

Computes frequency domain from complex time domain

Syntax: FFT <size> <bufA> <bufB> <bufX> <bufY>;

where:		
	<size>	the size of the Fourier transform
	<bufA,B>	buffers containing input data
	<bufX,Y>	buffers containing output data

Description

FFT COMPLEX produces a complex Fourier transform of the data in the input buffers. <Buffer A> contains the real portion of the input and <buffer B> contains the imaginary portion. The real portion of the output is placed in <buffer X> and the imaginary portion in <buffer Y>. The input buffers may be the same as the output buffers but the output buffers cannot be the same. The <size> of the transform must be an integer between 8 and 65536, inclusive, and must be a power of 2. <Buffer A> must be defined at least as large as the size of the transform.

The formats of the data in the output buffer are discussed in Chapter 2.

Examples:

```
860 PRINT #1:"fft 2048 1 2 1 2;"
```

An FFT is performed on the complex data points in buffers 1 and 2; the results are stored back in buffers 1 and 2, the real portion in 1 and the imaginary in 2.

Generate Waveform

Generates waveform data in a buffer

Syntax: GW <bufA> <size> <code>
[<bias> [<ampl> [<peri> [<phas>]]]];

where: <bufA> buffer number to load
<size> the number of points in the buffer
<code> a function code selected from the table below
<bias> the bias to apply to the waveform
<ampl> the scaling factor to apply to the waveform
<peri> the period of the periodic waveform
<phas> the phase of the periodic waveform in degrees

Description

GENERATE WAVEFORM generates a waveform in <bufA>. The function <code> is one of the codes listed in the table below:

code	function
0	a constant equal to <bias>
1	a square wave function symmetrical about <bias>
2	a triangular wave function symmetrical about <bias>
3	a rising sawtooth wave function symmetrical about <bias>
4	a falling sawtooth wave function symmetrical about <bias>
5	a sine function symmetrical about <bias>

The <bias> is applied to the function after the function is multiplied by <ampl>. The period <peri> is expressed in sample points, and must be an integer. <Phas> is in degrees.

Default Values

<bias>	0
<ampl>itude	1
<peri>od	128 points
<phas>e	0 degrees

Examples:

```
580 PRINT #1:"gw 1 12000 3 4 3 100;"
```

Buffer 1 is loaded with a rising sawtooth wave form with minimum value 1 and maximum value 7 (bias is 4 and amplitude is 3). Each sawtooth period is composed of 100 points with a total of 120 cycles in the buffer. The phase angle is 0 by default.

Integrate

Performs an integration on a buffer

Syntax: INT <bufA> <bufX> <init>;

where: <bufA> buffer containing input data
<bufX> buffer containing output data
<init> value of first point (constant of integration)

Description

INTEGRATE sets the first point in the output buffer equal to <init>. Then the first point of the input buffer is added to the first point of output buffer and the result is placed in the second point of the output buffer. The procedure of adding corresponding points of the two buffers continues until all of the input buffer points have been summed with the output buffer points. The number of points in the output buffer is one greater than the number of points in the input buffer.

Examples:

```
490 PRINT #1:"int 3 4 0.5"
```

Buffer 4 contains the result of integration of buffer 3; 0.5 is the constant of integration.

Inverse FFT Real

Computes real time domain data from frequency domain data

Syntax: INV <size> <bufA> <bufB> <bufX>;

where:		
	<size>	the size of the Fourier transform
	<bufA,B>	buffers containing input data
	<bufX>	buffer containing output data

Description

INVERSE FFT REAL produces a real inverse Fourier transform of the data in the input buffers. <Buffer A> contains the real frequency information and <buffer B> contains the imaginary information. The output placed in <buffer X> contains real time domain data. <Buffer B> is used by the transform for intermediate calculations, and its contents are destroyed. <Buffer A> may be the same as <Buffer X>. The <size> of the transform must be an integer between 8 and 65536, inclusive, and must be a power of 2. The output buffer must be defined at least twice as large as <size>.

The format of the data in the input buffers must be the same as the format of the data output from the "FFT real" command. This format is discussed in Chapter 2.

Examples:

```
390 PRINT #1:"inv 512 1 2 1;"
```

Buffers 1 and 2 contain real and complex frequency domain data; real time domain data from the inverse transform is placed back in buffer 1.

Inverse FFT Complex

Computes complex time domain data from frequency domain data

Syntax: INV <size> <bufA> <bufB> <bufX> <bufY>;

where:		
	<size>	the size of the Fourier transform
	<bufA,B>	buffers containing input data
	<bufX,Y>	buffers containing output data

Description

INVERSE FFT COMPLEX produces a complex inverse Fourier transform of the data in the input buffers. <Buffer A> contains the real frequency information and <buffer B> contains the imaginary information. The output placed in <buffer X> contains real time domain data and <buffer Y> contains the imaginary portion of the time domain data. The input buffers may be the same as the output buffers. The <size> of the transform must be an integer between 8 and 65536, inclusive, and must be a power of 2.

The format of the data in the input buffers must be the same as the format of the data output from the "FFT complex" command. This format is discussed in Chapter 2.

Examples:

```
730 PRINT #1:"inv 2048 1 2 4 5;"
```

The frequency domain data in buffers 1 and 2 is transformed to time domain data (real and complex) and stored in buffers 4 and 5.

Linear

Converts decibel data to linear data

Syntax: LIN <bufA> <bufX>;

where: <bufA> buffer containing input data
<bufX> buffer containing results

brief: X = 10 ^ (A/10)

Description

LINEAR converts the decibel data in the input buffer to linear form and outputs the results to the output buffer.

Examples:

```
180 PRINT #1:"lin 3 3;"
```

The data in buffer 3 is converted to linear form.

Maximum

Returns the maximum value in the buffer

Syntax: MAX <bufA>;

format: <maximum value> <dlm>

where: <bufA> buffer where the maximum value is located
<dlm> delimiter, either <CR> or <CR> <LF>

Description

MAX locates the maximum value in <buffer A> and sends that value out the port selected by the "OUTput device" command. The maximum value is followed by a Carriage Return and optionally a Line Feed if the output port was so defined. (See the Output Device command.)

Examples:

```
300 PRINT #1:"max 3;"
310 INPUT #1:B
```

The maximum value in buffer 3 is located and put into variable B.

Minimum

Returns the minimum value in the buffer

Syntax: MIN <bufA>;

format: <minimum value> <dlm>

where: <bufA> buffer where the minimum value is located
<dlm> delimiter, either <CR> or <CR> <LF>

Description

MIN locates the minimum value in <buffer A> and sends that value out the port selected by the "OUTput device" command. The minimum value is followed by a Carriage Return and optionally a Line Feed if the output port was so defined. (See the OUTput device command.)

Examples:

```
900 PRINT #1:"min 3;"
910 INPUT #1:A
```

The minimum value in buffer 3 is located and put into variable A.

Multiply

Multiplies two buffers together

Syntax: MUL <bufA> <bufB> <bufX>;

where: <bufA,B> buffers containing input data
<bufX> buffer containing results

brief: X = A*B

Description

MULTIPLY multiplies each point in <buffer A> by the corresponding point in <buffer B> and places the result in the corresponding point in <buffer X>. Any or all or the buffers may be the same. If any of the buffers are not the same length, MULTIPLY will continue until the end of the shortest buffer has been reached.

Examples:

```
770 PRINT #1:"mul 1 1 2;"
```

Buffer 2 is set to the square of each number in buffer 1.

Multiply Complex

Performs complex multiplication on two buffers

Syntax: MUL <bufA> <bufB> <bufC> <bufD> <bufX> <bufY>;

where: <bufA,B,C,D> buffers containing input data
<bufX,Y> buffers containing results

brief: X = A*C - B*D Y = B*C + A*D

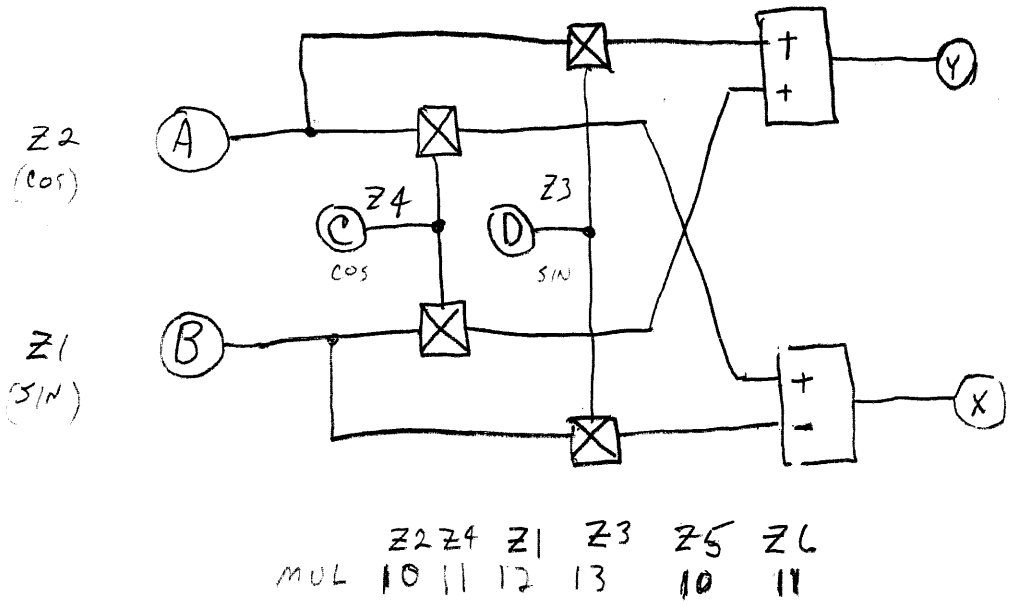
Description

MULTIPLY COMPLEX uses complex number multiplication to multiply the complex series whose real part is in <buffer A> and whose imaginary part is in <buffer B> by the complex series whose real part is in <buffer C> and whose imaginary part is in <buffer D>. MULTIPLY COMPLEX places the real portion of the result in <buffer X> and the imaginary portion of the result in <buffer Y>. Any or all or the buffers may be the same, except for the two output buffers. If any of the buffers are not the same length, MULTIPLY COMPLEX will continue until the end of the shortest buffer has been reached.

Examples:

```
590 PRINT #1:"mul 1 2 3 4 5 6;"
```

The complex numbers in buffers 1 and 2 are multiplied by the complex numbers in buffers 3 and 4; the results are placed in buffers 5 and 6.



Multiply by Conjugate

Multiplies a complex buffer pair with the conjugate of another pair

Syntax: MULC <bufA> <bufB> <bufC> <bufD> <bufX> <bufY>;

where: <bufA,B,C,D> buffers containing input data
<bufX,Y> buffer containing results

brief: X = A*C + B*D Y = B*C - A*D

Description

MULTIPLY BY CONJUGATE uses complex number multiplication to multiply the complex series whose real part is in <buffer A> and whose imaginary part is in <buffer B> by the complex conjugate of the series whose real part is in <buffer C> and whose imaginary part is in <buffer D>. MULTIPLY BY CONJUGATE places the real portion of the result in <buffer X> and the imaginary portion of the result in <buffer Y>. Any or all or the buffers may be the same, except for the two output buffers. If any of the buffers are not the same length, MULTIPLY BY CONJUGATE will continue until the end of the shortest buffer has been reached.

Examples:

```
480 PRINT #1:"mulc 1 2 1 2 3 4;"
490 PRINT #1:"sqrt 3 3;"
```

The complex numbers in buffers 1 and 2 are multiplied by their complex conjugate; the results are placed in buffers 3 and 4. Buffer 4 will be all zeros and buffer 3 will contain the square of the magnitude of the complex vectors described in buffers 1 and 2. Line 490 calculates the square root of buffer 3; buffer 3 contains the magnitude of the complex pairs contained in buffers 1 and 2. For a more compact program, both lines could have been put in one line as follows:

```
480 PRINT #1:"mulc 1 2 1 2 3 4;sqrt 3 3;"
```

Negate

Multiplies a buffer by -1

Syntax: NEG <bufA> <bufX>;

where: <bufA> buffer containing input data
<bufX> buffer containing results

brief: X = -A

Description

NEGATE negates each point in <buffer A> and places the result in <buffer X>. The input buffer and the output buffer may be the same.

Examples:

```
420 PRINT #1:"neg 4 4;"
```

The contents of buffer 4 are negated.

Polar

Converts rectangular data to polar data

Syntax: POL <bufA> <bufB> <bufX> [<bufY>];

where: <bufA,B> buffers containing input data
<bufX,Y> buffer(s) containing results

brief: X = SQR(A*A + B*B) Y = ATN(B/A)

Description

POLAR converts the data in the input buffers to magnitude and angle data. <Buffer A> contains the X coordinate and <buffer B> contains the Y coordinate information. The output magnitude data is stored in <buffer X> and the output angle data in <buffer Y>. If <buffer Y> is not specified, no angle data is produced. The output buffers may be the same as the input buffers, but both output buffers, if present, may not be the same.

Examples:

```
760 PRINT #1:"pol 1 2 5;"
```

The magnitude of the vectors described by the data in buffers 1 and 2 is stored in buffer 5.

Rectangular

Converts polar data to complex data

Syntax: REC <bufA> <bufB> <bufX> [<bufY>];

where: <bufA,B> buffers containing input data
<bufX,Y> buffers containing results

brief: X = A*COS(B) Y = A*SIN(B)

Description

RECTANGULAR converts the magnitude and angle data in the input buffers to rectangular form. <Buffer A> contains the magnitude data and <buffer B> contains the angle data. X and Y rectangular data are stored in the respective output buffers. If <buffer Y> is not specified, no Y rectangular data is produced. The output buffers may be the same as the input buffers, but both output buffers, if present, may not be the same.

Examples:

```
240 PRINT #1:"rec 1 2 1 2;"
```

The polar data (magnitude and angle) in buffers 1 and 2 is converted to rectangular data back into buffers 1 and 2.

Rotate

Rotates buffer points to right

Syntax: ROT <bufA> <bufX> <psns>;

where: <bufA> buffer containing input data
<bufX> buffer containing results
<psns> number of positions to rotate data

Description

ROTATE rotates the points in <buffer A> to the right and stores the result in <buffer X>. The number of positions to rotate is given by the <psns> parameter. The first point in the buffer is considered the left end and the last point in the buffer is considered the right end. As a point is rotated out of the buffer at the right end, that point is inserted into the buffer at the left end. The input buffer and the output buffer may be the same.

Examples:

```
160 PRINT #1:"rot 1 2 5;"
```

The contents of buffer 1 are rotated 5 positions and saved in buffer 2. If the data in buffer 1 was 0,1,2,3,4,5,6,7,8,9,10,11,12 then the data in buffer 2 would be 8,9,10,11,12,0,1,2,3,4,5,6,7.

Scale

Scales and/or adds offset to a buffer

Syntax: SCA <bufA> <bufX> <mltl> <addJ>;

where: <bufA> buffer containing input data
<bufX> buffer containing results
<mltl> scale factor
<addJ> offset factor

brief: X = A*I + J

Description

SCALE first multiplies every point in <buffer A> by <mltl>, and then adds <addJ> to the product. The result is stored in the corresponding point of <buffer X>. The input buffer and the output buffer may be the same.

Examples:

```
510 PRINT #1:"sca 3 4 8 24;"
```

The contents of buffer 3 are multiplied by 8; then 24 is added and the results are put in buffer 4.

Shift

Shifts buffer points to right inserting 0 at left

Syntax: SHF <bufA> <bufX> <psns>;

where: <bufA> buffer containing input data
<bufX> buffer containing results
<psns> number of positions of to shift

Description

SHIFT shifts the points of <buffer A> to the right and stores the result in <buffer X>. Zeros are inserted into <buffer X> at the left. The first point in the buffer is considered the left end and the last point in the buffer is considered the right end. The input buffer and the output buffer may be the same.

Examples:

```
840 PRINT #1:"shf 1 1 5;"
```

The contents of buffer 1 are shifted 5 positions and saved back in buffer 1. If the data in buffer 1 was 1,2,3,4,5,6,7,8,9,10,11,12 then the data after shifting would be 0,0,0,0,0,1,2,3,4,5,6,7.

Subtract

Subtracts one buffer from another

Syntax: SUB <bufA> <bufB> <bufX>;

where: <bufA,B> buffers containing input data
<bufX> buffer containing results

brief: X = A-B

Description

SUBTRACT subtracts all the points in <buffer B> from the corresponding points in <buffer A> and places the result in the corresponding points of <buffer X>. Any or all or the buffers may be the same. If any of the buffers are not the same length, SUB will continue until the end of the shortest buffer has been reached.

Examples:

```
380 PRINT #1:"sub 1 2 1;"
```

The contents of buffer 1 are reduced by the amounts in buffer 2.

Subtract Complex

Subtracts one complex number buffer from another

Syntax: SUB <bufA> <bufB> <bufC> <bufD> <bufX> <bufY>;

where:	<bufA,B,C,D>	buffers containing input data
	<bufX,Y>	buffer containing results
brief:	X = A-C	Y = B-D

Description

SUBTRACT COMPLEX subtracts <buffer C> from <buffer A> and places the results in <buffer X>. It then subtracts <buffer D> from <buffer B> and places the result in <buffer Y>. Any or all or the buffers may be the same, except for the two output buffers. If any of the buffers are not the same length, SUBTRACT COMPLEX will continue until the end of the shortest buffer has been reached.

Examples:

```
330 PRINT #1:"sub 1 2 3 4 3 4;"
```

The complex numbers in buffers 3 and 4 are subtracted from the complex numbers in buffers 1 and 2; the resulting complex numbers are stored back in buffers 3 and 4, overwriting the former contents.

Square Root

Computes the square roots of a buffer

Syntax: SQRT <bufA> <bufX>;

where: <bufA> buffer containing input data
<bufX> buffer containing results

brief: X = SQRT(A)

Description

SQUARE ROOT computes the square root of <buffer A> and places the results in <buffer X>. The input and output buffers may be the same.

If a negative value is encountered in the input buffer, no error is generated but the value 0 is returned as the result.

Examples:

```
650 PRINT #1:"sqrt 1 1;"
```

The contents of buffer 1 are replaced by the square root of the contents.

Appendix A

Example Programs

Input then Output an Analog Signal

This routine inputs an analog signal through an analog input card and then outputs the signal through an analog output card. The DFN command allocates 2000 bytes of memory for buffer number 1 (1000 points * 2 bytes/point). The PER command sets the time between sample points to 10 micro-seconds. It is assumed that an analog input card is plugged into slot 1 and that an analog output card is plugged into slot 2. The AI command converts 1000 points of the analog input to digital values and stores the results in buffer 1. The programmable gain is set to 1. The sample time will last for 10 mili-seconds (1000 points * 10 micro-seconds per point). The AO command outputs the signal stored in buffer 1 to channel B2. The output signal will repeat the input signal 500 times for a total of duration of 5 seconds (500 repetitions * 1000 points * 10 microseconds per point).

```
100 OPEN #1:"COM1:","f"          ! communication port opened
110 PRINT #1:"DFN 1 1000 2;"     ! data buffer defined for 1000 words
120 PRINT #1:"PER 1e-5;"         ! period set to 10 microseconds
130 PRINT #1:"AI 1000 A1 1 1;"   ! 1000 points are converted from A1
140 PRINT #1:"AO 1000 B2 1;"     ! data output on channel B2 repeated
150 END                          !      1000 times
```

Output a Sine Wave

This routine creates a sine wave in a buffer and outputs it on an analog channel; the analog output card is plugged into slot 4. The DFN command allocates 2000 bytes of memory for buffer number 1 (1000 points * 2-bytes). The GW command generates a sine wave in buffer 1 which has a bias of 0 volts and a peak of 5 volts. Each period of the sine wave contains 100 points, which means there are total of 10 periods in the buffer (1000 points / 100 points per period). The PER command sets the time required to output a point to 10 micro-seconds. Thus the period of the sine wave is 1 millisecond (100 points/period * 10 microseconds/point) and the sine wave frequency is 1 KHertz. The AO command repeatedly (2000 times) outputs the sine wave in buffer 1 to channel A4. The sine wave will last for 20 seconds (2000 repetitions * 1000 points * 10 microseconds/point).

```
100 OPEN #1:"COM2:","f"          ! open communication port
110 PRINT #1:"DFN 1 1000 2;"     ! buffer 1 defined for 1000 words
120 PRINT #1:"GW 1 1000 5 0 5 100 0;"  ! sine wave generated
130 PRINT #1:"PER 1e-5;"         ! period set to 10 microseconds
140 PRINT #1:"AO 2000 A4 1;"     ! buffer 1 output 2000 times
150 END
```

Spectral Power Density (Autocorrelation)

This routine determines the spectral power density of sampled data and sends both the time and frequency data back to the host. Buffers 1 and 2 are defined as floating point buffers because they will be involved with math operations. The length of each buffer is set to 1024 in line 120. The period is set to 100 microseconds which corresponds to a sampling rate of 10 KHertz. After the data is acquired, it is sent to the host and stored in the TIMEDAT array. Then an FFT is performed on the data, the results being placed in buffers 1 and 2. By multiplying the FFT data by its own complex conjugate, the spectral power density is obtained. The contents of buffer 2 are zero after the complex conjugate multiplication. The frequency power data is then sent to the host and saved in array variable FREQDAT.

```
100 OPEN #1:"COM1:","f"                    ! communication port opened
110 DIM TIMEDAT(1024), FREQDAT(1024)
120 PRINT #1:"DFN 1 1024 4 2 1024 4;"      ! define buffers 1 and 2
130 PRINT #1:"PER 1E-4;"                   ! set the period to 100 microsecond
140 PRINT #1:"AI 1024 A1 1 4;"             ! input analog data
150 PRINT #1:"WR 1;"                       ! initiate the output of time data
160 INPUT #1:N                             ! input the number of data points
170 FOR I = 1 TO N
180    INPUT #1:TIMEDAT(I)                 ! load up the time data array
190 NEXT I
200 PRINT #1:"FFT 1024 1 1 2;"             ! perform an FFT on the time data
210 PRINT #1:"MULC 1 2 1 2 1 2;"           ! multiply by complex conjugate
220 PRINT #1:"WR 1;"                       ! initiate the output of data
230 INPUT #1:N                             ! input the number of data points
240 FOR I = 1 TO N
250    INPUT #1:FREQDAT(I)                 ! load up the frequency data array
260 NEXT I
270 END
```

Using an Executable Buffer

This program uses the previous example program to obtain time and spectral power density data at different sampling rates using an executable buffer. The three sample rates are calculated in line 320 and will be 1 milliseconds, 0.1 milliseconds and 0.01 milliseconds. The executable buffer contains the commands to take the data, send the time data, find the power spectral density and send that data. The host stores the data in the two-dimensional arrays TIMEDAT and FREQDAT.

```
100 OPEN #1:"COM1:","f"                    ! communication port opened
110 DIM TIMEDAT(3,1024), FREQDAT(3,1024)
120 PRINT #1:"DFN 1 1024 4 2 1024 4;"      ! define buffers 1 and 2 for data
130 PRINT #1:"DFN 9 500 5;"                ! define buffer 9 for execution

140 PRINT #1:"LOAD 9;"                     ! open buffer 9 for commands
150   PRINT #1:"AI 1024 A1 1 4;"           ! input analog data
160   PRINT #1:"WR 1;"                     ! initiate the output of time data
170   PRINT #1:"FFT 1024 1 1 2;"           ! perform an FFT on the time data
180   PRINT #1:"MULC 1 2 1 2 1 2;"         ! multiply by complex conjugate
190   PRINT #1:"WR 1;"                     ! initiate the output of data
200 PRINT #1:"END;"                        ! executable buffer closed

300 P = 0.01
310 FOR J = 1 TO 3                         ! start loop for 3 data readings
320   P = P * 0.1                          ! calculate the period
330   PRINT #1:"PER ",P,";"                ! set the sampling period
340   PRINT #1:"RUN 9;"                    ! run the executable buffer
350   INPUT #1:N                           ! input the number of data points
360   FOR I = 1 TO N
370     INPUT #1:TIMEDAT(J,I)              ! load up the time data array
380   NEXT I
390   INPUT #1:N                           ! input the number of data points
400   FOR I = 1 TO N
410     INPUT #1:FREQDAT(J,I)              ! load up the frequency data array
420   NEXT I
430 NEXT J
440 END
```

Appendix B

Error Messages

If MDAS encounters a problem executing a command, an error message is sent to the device currently assigned to receive error information. A device can be assigned either with the *ERRORS DEV command or by using the switch/display module. The twelve errors are:

1.	"VALUE "	Value of parameter invalid
2.	"RANGE "	Parameter out of range
3.	"WIDTH "	Invalid word width or unmatching word sizes
4.	"CNFLCT"	Two output buffers have the same name
5.	"SYNTAX"	Incorrect number of parameters in command line
6.	"COMAND"	Invalid command
7.	"MEMORY"	Not enough memory for requested operation
8.	"SPEED "	Period too fast for sequence
9.	"MATH "	Divide by zero
10.	"LABEL "	Incorrect label definition
11.	"STACK "	Stack overflow
12.	"SPAWN "	Error in a spawned process

If the Error Device is set to the LED display, the error message is displayed on the LED display.

The format for error messages is:

CC:mmmmmm [n]

CC is the command code of the command that MDAS tried to execute. mmmmmm is the error message as printed in the above list. [n] is the position of the bad parameter in the command line. If the command code itself is not 2, 3 or 4 characters, then APPL is displayed in place of a command code (APPLication error).

When a GET ERROR command is issued, only the error number is sent to the data output device, the error message on the LED display is cleared and the error number is set to 0. The CLEAR ERROR command sets the error number to 0 and does not affect the LED display.

Appendix C

Bridge Card Option B1P2

Theory of Operation

The bridge input option uses an unbalanced bridge technique for measuring sensors that are in a bridge configuration. Quarter, half and full bridge configurations can be used. Figure C-1 is a schematic of the bridge and input amplifier for one channel. Changes in bridge resistance values due to external paramaters cause the input voltage from the bridge to change. The relationship between resistance change and input voltage change depends on the bridge configuration used.

The unbalanced bridge technique has several advantages. One is that precise bridge balancing by the user is not neccessary. Each time a bridge channel is calibrated, the excitation voltage and unstrained input voltage are recorded. After bridge data has been collected, each reading is corrected by using the ratio of input to excitation voltage and subtracting out the unbalanced input voltage to excitation voltage ratio. This technique simplifies the use of bridges by correcting for imbalance in firmware.

Another advantage to the unbalanced bridge technique is the increased range of operation that can be achieved. An example may serve best to describe this feature. For best results the gain of the amplifier on the bridge card should be set as high as possible. Suppose that the test of interest causes the input voltage from the bridge to change by +15 millivolts. If the bridge is initially balanced, the highest gain that could be used to stay below 10 volts is 667 (0.015/10). If the bridge is intentionally unbalanced so that the unstrained input from the bridge is -8 millivolts, then a gain of 1000 can be used.

If desired, the unbalanced bridge technique does not have to be used. The bridge may be balanced manually and the input voltages retrieved without any adjustments.

In the quarter bridge configuration the sensing resistor should be either RG1 or RG2. In the half bridge configuration, the sensing resistors should be RG1 and RG2. This allows the calibration resistor to be shunted across a sensing resistor. The bridge may be completed either on the bridge card or externally.

Resistors RG1, RG2, RG3, RG4, RB and RC must be supplied by the user. Solder terminals are provided on the bridge board for soldering these resistors in place.

The CALB command measures the unstrained or nominal bridge voltage and also calibrates the bridge by automatically switching in the bridge calibration resistor. If a calibration resistor is not used, the gain passed with the DFNC command is used and the CF term must be set to 0 or not included in the command. After the CALB command has been executed, all readings from that channel are adjusted for imbalance and calibration using the parameters obtained during the CALB routine.

The bridge may be balanced or intentionally unbalanced by soldering in a balancing resistor and adjusting the balance trimpot. The balance trimpot is accessible from the front of the card. The ouput of the amplifier is also available at the front of the card. A standard male pin connector can be used to connect to the amplifier output.

For additional information on bridge measurements, Application Note 290-1 **Practical Strain Gage Measurements** by The Hewlett Packard Company is an excellent reference.

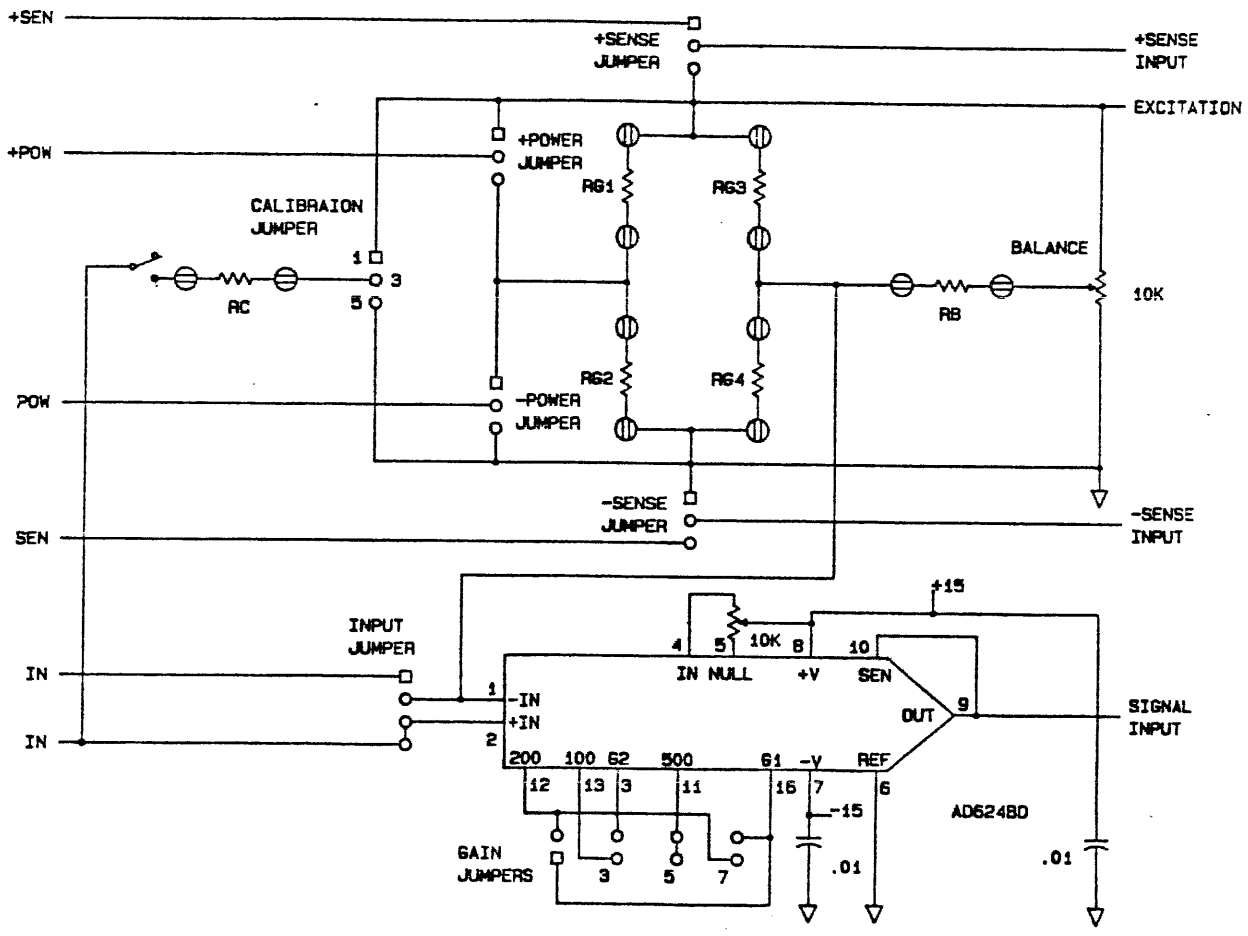


Figure C-1.

Bridge Configuration

Figure C-2 shows the parts placement and connector labeling for the bridge card. The 'A' bridge is the upper circuit and the 'E' bridge is the lower circuit. Each bridge circuit is identified using the prefixes 'A' and 'E'.

The connectors are used with shorting blocks to configure each bridge circuit. All connectors are labled with a J prefix an the pin numbers are assigned according to Figure C-3. Shorted connections are indicated with a dash: 1-2 means pins 1 and 2 of the connector are jumpered (shorted) together.

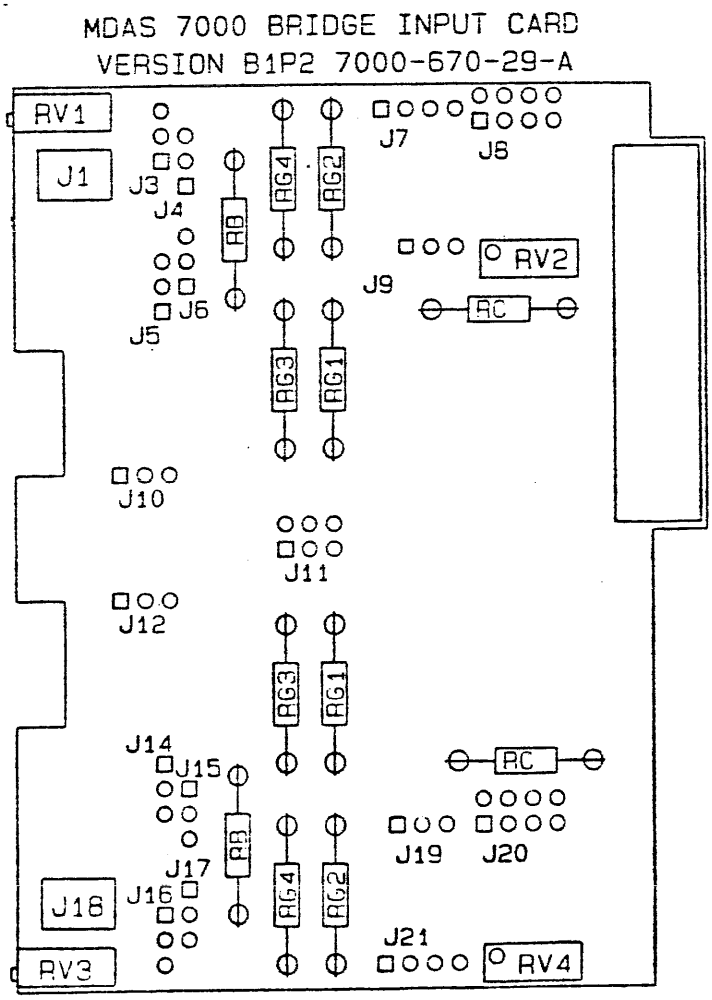


Figure C-2.

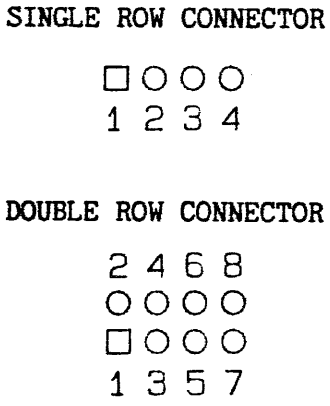


Figure C-3.

Excitation Voltage Selection

The excitation voltage is either 4 or 8 Volts and is jumper selectable. In both cases the current is limited to 50 mA. Due to maximum loading constraints on the power supplies the minimum bridge resistances are 250 Ohms for 8 Volt excitation and 100 Ohms for 4 Volt excitation.

	A Channel	E Channel
4 Volt Excitation	J11 4-6 J10 1-2	J11 3-5 J12 1-2
8 Volt Excitation	J11 2-4 J10 2-3	J11 1-3 J12 2-3

Amplifier Gain Selection

The instrumentation amplifier has a number of jumper selectable gains. The gain selection is the same for both amplifiers.

Gain	A Channel (J8), E Channel (J20)
1000	4-6, 1-2, 3-5
831	4-6, 1-2
688	4-6, 1-3, 5-7
624	4-6, 1-3
500	4-6, 5-7
375	2-4, 1-3
333	2-4, 6-8
250	2-4, 3-5
200	2-4, 5-6
186.5	3-4, 6-8, 5-7
137	3-4, 5-7
125	3-4, 6-8
100	3-4, 5-6
1	1-2, 5-6, 7-8

Unused jumper blocks can be stored by plugging one end of the jumper block to pin 7 or 8 with no connection to the other end of the jumper block.

Calibration Resistor

The calibration resistor can be connected to shunt either RG1 or RG2.

	A Channel	E Channel
RC shunt RG1	J9 1-2	J19 1-2
RC shunt RG2	J9 2-3	J19 2-3

Configuration of Excitation Leads

The excitation signal is carried on the two wires labled +POWER and -POWER. The +POWER signal can be connected to either the excitation voltage or to the RG1-RG2 node of the bridge. The -POWER signal can be connected to either ground or the RG1-RG2 node of the bridge.

For all full and half bridge configurations as well as quarter bridges that are completed externally, +POWER is connected to the excitation voltage and -POWER is connected to ground. For internally completed quarter bridges with RG1 the active resistor, +POWER is connected to the excitation voltage and -POWER is connected to the RG1-RG2 node. For internally completed quarter bridges with RG2 the active resistor, +POWER is connected to the RG1-RG2 node and -POWER is connected to ground.

	A Channel		E Channel	
+POWER to excitation	J6	1-2	J15	1-2
+POWER to RG1-RG2 node	J6	2-3	J15	2-3
-POWER to RG1-RG2 node	J4	1-2	J17	1-2
-POWER to ground	J4	2-3	J17	2-3

Configuration of Sense Leads

The excitation sense signal is carried on the two wires labled +SENSE and -SENSE. The +SENSE signal can be connected internally to the excitation voltage or can be connected to the the +SENSE line provided at the external connector. The -SENSE signal can be connected internally to ground or can be connected to the the +SENSE line provided at the external connector.

The configuration of the SENSE lines depends on the type of bridge and the number of conductors in the cable. All full bridge configurations can be connected internally as well as externally providing the lead resistance is the same for +POWER and -POWER. If at all possible, half and quarter bridges SENSE lines should be connected externally. This minimizes the effect lead resistance may have on the measurements. Generally the SENSE lines should be connected similarly, that is, both external or both internal.

	A Channel		E Channel	
+SENSE to external	J5	1-2	J14	1-2
+SENSE to excitation	J5	2-3	J14	2-3
-SENSE to ground	J3	1-2	J16	1-2
-SENSE to external	J3	2-3	J16	2-3

Configuration of -Input Lead

The input signal is carried on the two wires labled +INPUT and -INPUT. The +INPUT signal is always connected to the +INPUT wire. The -INPUT signal can be connected to the -INPUT wire or if the bridge is conpleted internally, the -INPUT signal is only connected to the RG3-RG4 node of the bridge.

	A Channel	E Channel
-INPUT to external	J7 1-2	J21 1-2
-INPUT to internal	J7 3-4	J21 3-4

Output Wire Assignments

The ouput cable assignments are as follows:

-SENSE 1
-POWER 2
+INPUT 3
-INPUT 4
+POWER 5
+SENSE 6

Further information on cable and connector wiring is found at the end of this Appendix.

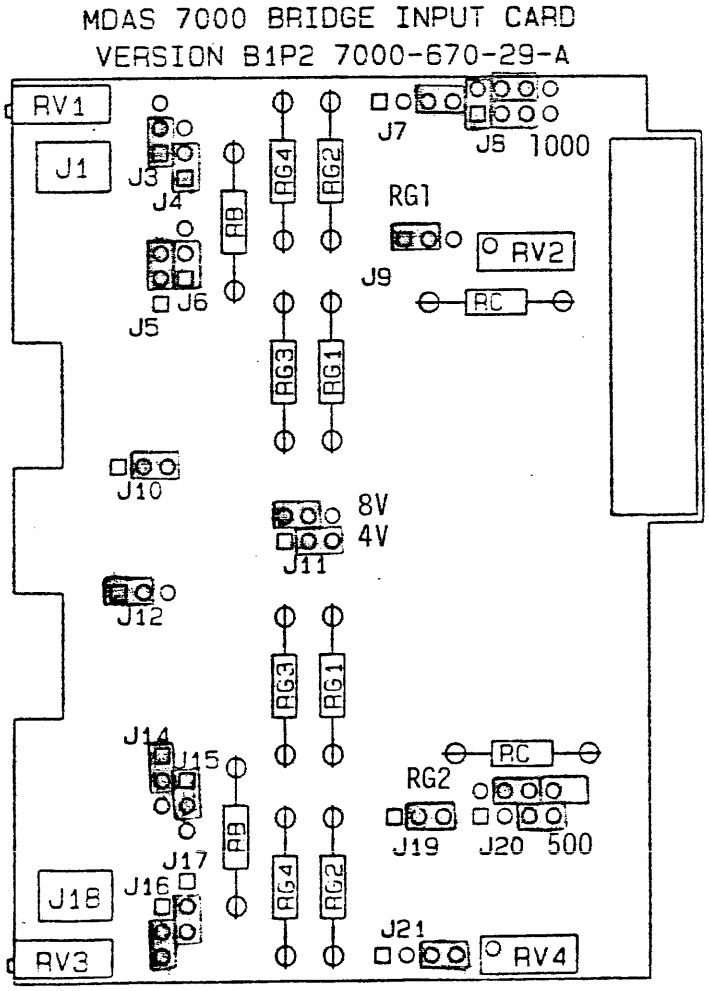
Configuration Examples

The following diagrams show several different bridge card configurations. The positions of the shorting blocks are indicated by the darkened rectangles.

	A Channel	E Channel
Bridge configuration	quarter	quarter
Bridge completion	internal	external
Excitation voltage	8 Volts	4 Volts
Amplifier gain	1000	500
RC shunted	RG1	RG2
+POWER	excitation	excitation
-POWER	RG1-RG2 node	ground
+SENSE	internal	external
-SENSE	internal	external
-INPUT	internal	internal

1/4 bridge internal

1/4 bridge external

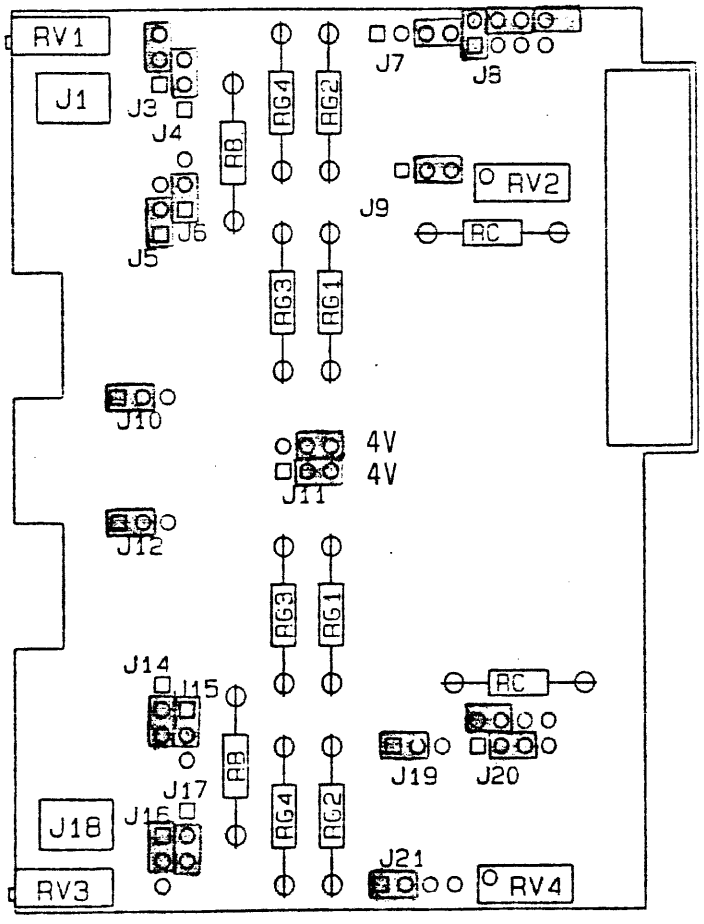


	A Channel	E Channel
Bridge configuration	half	full
Bridge completion	internal	external
Excitation voltage	4 Volts	4 Volts
Amplifier gain	831	250
RC shunted	RG2	RG1
+POWER	excitation	excitation
-POWER	ground	ground
+SENSE	external	internal
-SENSE	external	internal
-INPUT	internal	external

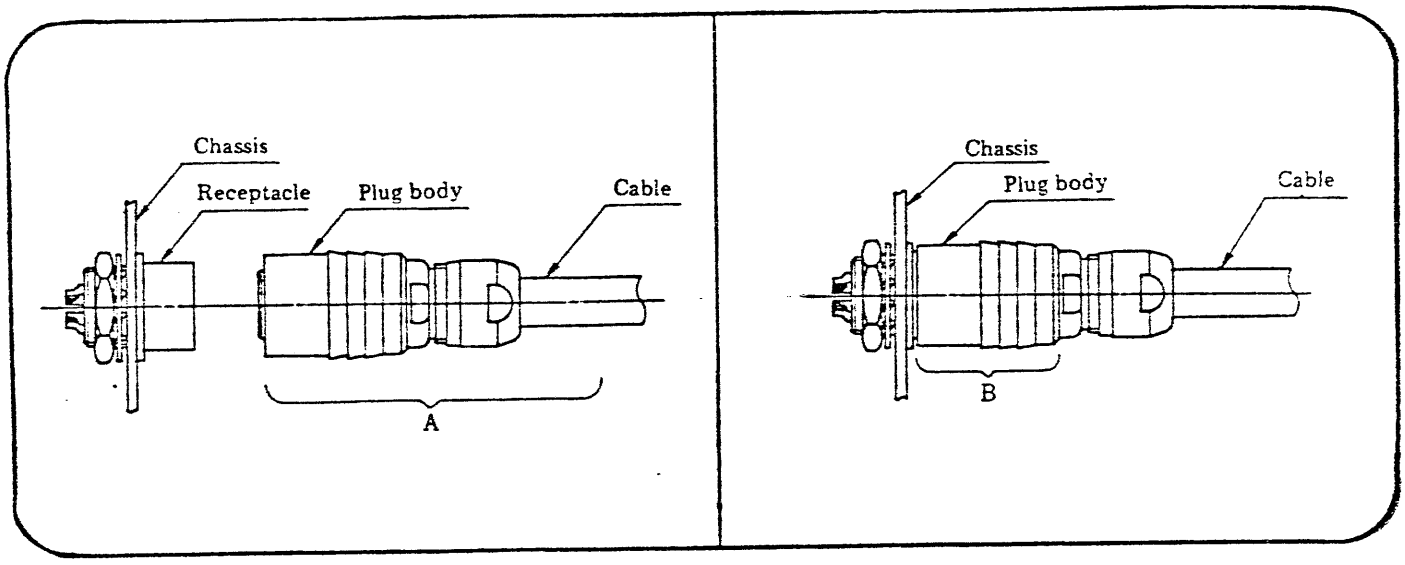
MDAS 7000 BRIDGE INPUT CARD
VERSION B1P2 7000-670-29-A

1/2 bridge external

full bridge



MDAS 7000 MULTIPLE PIN CONNECTOR DESCRIPTION:



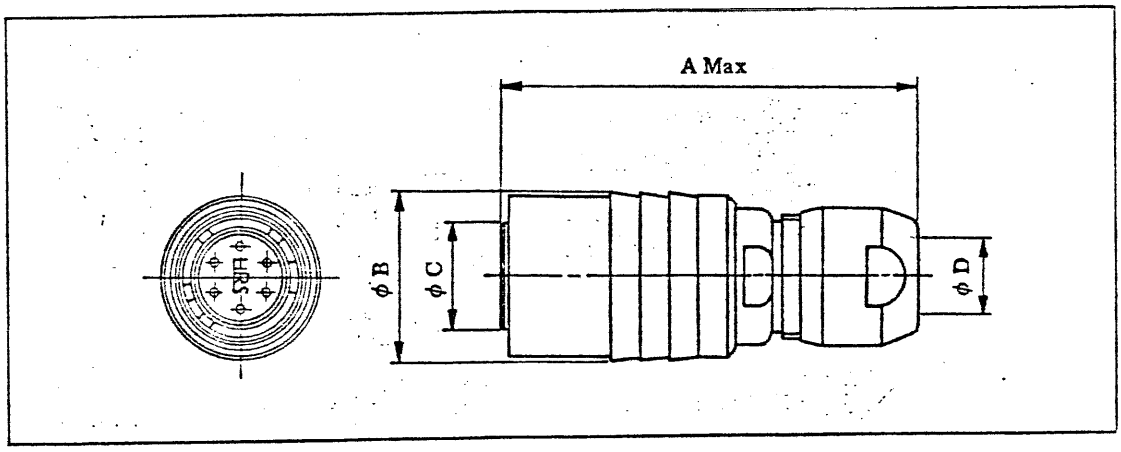
PIN-OUT DIAGRAM
(FOR CABLE SOLDERING)

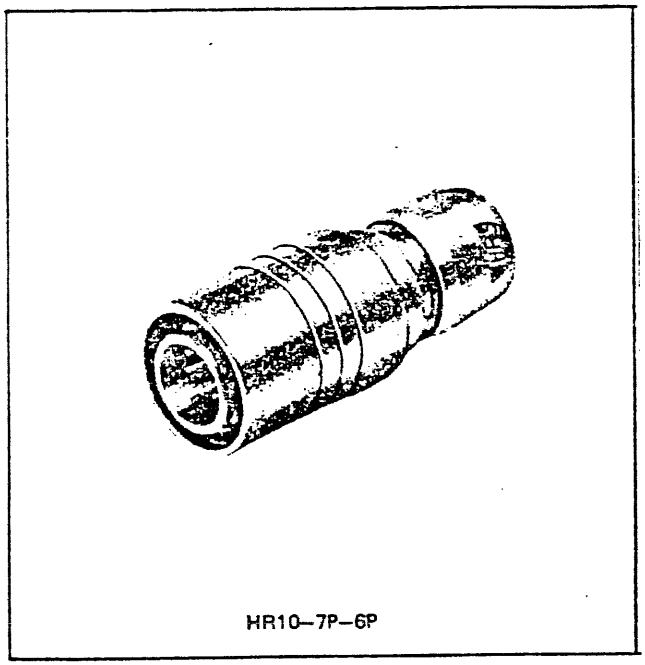
BLACK	**1**	**-SENSE**
BROWN	**2**	**-POWER**
RED	**3**	**+INPUT**
GREEN	**4**	**-INPUT**
BLUE	**5**	**+POWER**
WHITE	**6**	**+SENSE**

PLUG WITH PIN INSERT

mm (inch)

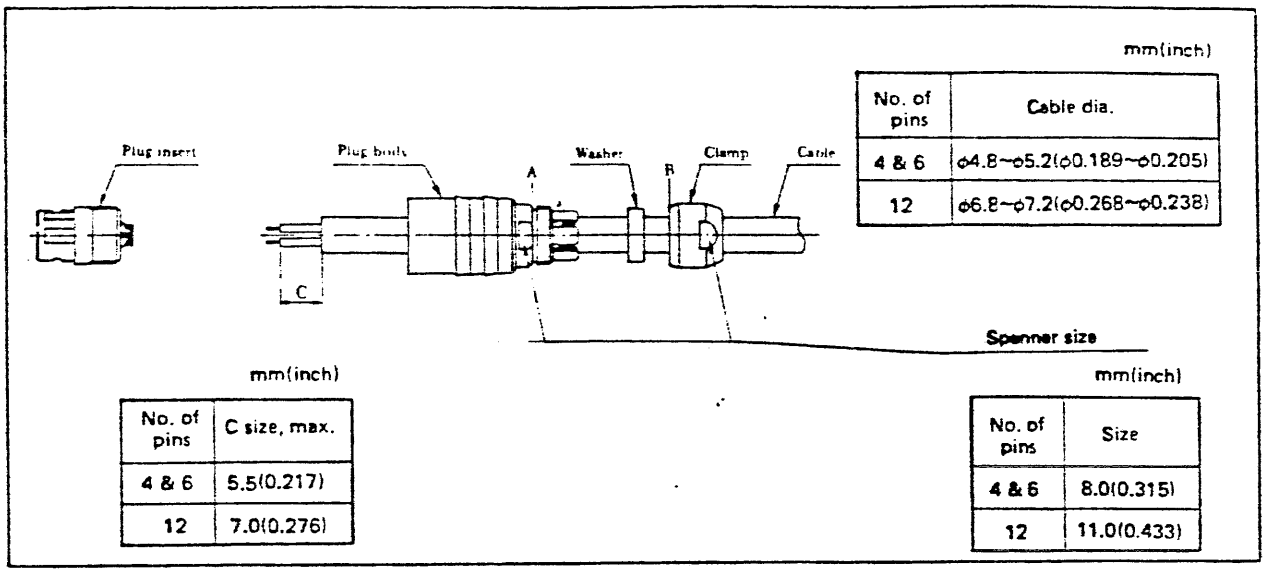
No. of contact	Part No.	A	B	C	D	Weight
6	HR10–7P–6P	28.5 (1.122)	11 (0.433)	7 (0.276)	5.2 (0.205)	8 grs.





HR10–7P–6P

Wiring Procedure (for reference)



mm(inch)

No. of pins	Cable dia.
4 & 6	ϕ4.8~ϕ5.2(ϕ0.189~ϕ0.205)
12	ϕ6.8~ϕ7.2(ϕ0.268~ϕ0.238)

mm(inch)

No. of pins	C size, max.
4 & 6	5.5(0.217)
12	7.0(0.276)

mm(inch)

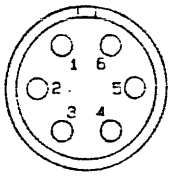
No. of pins	Size
4 & 6	8.0(0.315)
12	11.0(0.433)

1. The cable should be AWG26 or smaller, and the outer diameter must meet the specifications shown in the table above.
2. Pass the cable through the cable clamp, washer, and plug body one after another as shown above.
3. Solder cable to plug insert.
4. Fasten plug body with spanner wrench.
5. Slide washer against plug body.
6. Tighten cable clamp until surface "B" touches surface "A".

PIN-OUT DIAGRAM
(FOR CABLE SOLDERING)

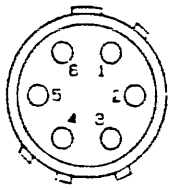
RECEPTICAL WITH SOCKET INSERT

(ON I/O CARD)



PLUG WITH PIN INSERT

(ON CABLE)



1 -SENSE
2 -POWER
3 +INPUT
4 -INPUT
5 +POWER
6 +SENSE

Appendix D

Digital I/O Card Option R5P8

Theory of Operation

The digital I/O card, Option R5P8, consists of 16 digital (TTL compatible) input lines and 8 latched digital (TTL compatible) output lines. Three of the output lines can be used as control lines; the remaining 5 output lines are uncommitted. The 16 input lines are multiplexed to one 8 bit bus. A trigger line is made available to control the flow of data.

The 16 input lines, the 8 output lines and the trigger line are available on the 50 pin connector at the front of the I/O card. Viewing the I/O card in the upright position from the front, all of the connector pins in the left row are grounded and the active signals are on the pins in the right row. The trigger line is top-most pin on the right. The 8 output lines occupy the next 8 pins and the 16 input lines are on the remaining pins. In each case, D0 is the top-most pin of each byte. In the 16 bit format, D0 of the top input byte is the least significant bit and D7 of the bottom input byte is the most significant byte.

The output bits are latched and are controlled using any of the digital output commands. Channel A is assigned to D0 and channel H is assigned to D7. The input bits are automatically read in as bytes or words if the Digital Input Word (DIW) command is used. These bits can also be read using any of the digital input commands in conjuction with the multipexing bit, output bit D6. The channel assignments for the input bits is the same as for the output bits; channel A is assigned to D0, channel H is assigned to D7.

Control Bit Configuration

The output bits D0 through D4 are uncommitted. The other three bits are also connected to provide some control fuctions. D5 can be used to control the trigger input. D6 is the multiplexor bit for the input data bytes. D7 can be disconnected as an output and used as an interrupt input. Figure D-1 shows the location of the connectors used to configure the contol functions. Connector position 1 is indicated by a square.

By placing a jumper in position 1-2 of connector J3, the D5 bit becomes an enable for the trigger input. A high written to D5 will enable the trigger and a low will disable the trigger. If a jumper is placed in position 2-3 of connector J3, then D5 is uncommitted and the trigger is disabled. With no jumper on connector J3, the trigger is enabled and D5 is an uncommitted output.

Output bit D6 is used to control the multiplexing of the input data. When D6 is set high, the top 8 input data bits can be read. When D6 is set low, the bottom data bits can be read. This data bit is always connected to the multiplexing circuitry and since D6 is used by the Digital Input Word command, care must be taken when using this bit as an output.

The use of output bit D7 as a control line involves two connectors, J1 and J2. To use D7 as an uncommitted output, a jumper must be placed in the 2-3 positon of connector J1 and a jumper must also be placed in the 2-3 position of connector J2. To use D7 as an interrupt input, jumpers must be in the 1-2 position of connectors J1 and J2. In this mode, the output bit D7 is used internally to clear an interrupt by setting it low and then setting it high. An interrupt is generated on the falling edge of the interrupt input.

The following is a summary of the control functions as defined by the placement of the jumpers.

Function	Connector and jumper position
Trigger enabled by high D5 Trigger disabled by low D5	J3 1-2
Trigger always disabled	J3 2-3
Trigger always enabled	J3 all pins open
Output D7 uncommitted No interrupt capability	J1 2-3 J2 2-3
Interrupt at output D7 bit	J1 1-2 J2 1-2

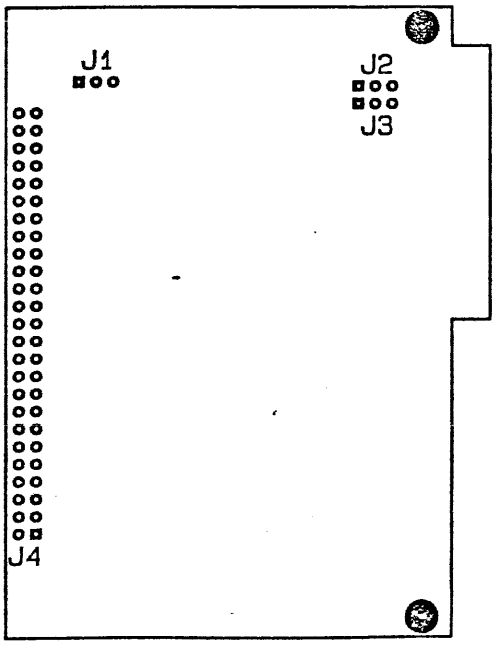


Figure D-1.

Appendix E

Stepper Motor Card Option D5P1

Theory of Operation

OVERVIEW

The stepper motor card option D5P8 allows a 7000 MDAS to control the operation of a stepper motor configured in the unipolar mode. The required power to run a motor may come from the MDAS +5 Volt supply or may be supplied externally by the user. Also, a boost voltage, supplied externally by the user or internally from the MDAS +15 Volt supply, may be used to increase the performance of the stepper motor. Direction of rotation, step rate, step acceleration, number of steps and full or half step mode are all selected under software control.

The nature of the unipolar motor configuration requires the dissipation of the inductive energy of each winding when it is turned off. This is accomplished by directing this energy into 15 Volt transient suppressors that are rated at 5 Watts. The amount of energy that can be dissipated depends on the winding inductance, the winding current and the rate at which the winding is turned on and off. To stay within the power rating of the devices, the maximum full steps per second should be limited to 40 / (L * I*I) where L is the winding inductance and I is the winding current. The winding current is the hold voltage divided by the winding resistance.

CONTROLLING THE STEPPER MOTOR CARD

The RAMP command is the most convenient way to control the stepper motor card. (See the description of the RAMP command.) However, the stepper motor card can be controlled by commands other than the RAMP command. Channels B, C and D are used to control the stepper card. Given 'n' as the slot number where the stepper card is located, Bn is the direction bit, Cn is the mode bit, and Dn is the clock bit.

The direction bit (Bn) is set high (+5 volts) for clockwise rotation and is set low (0 volts) for counter-clockwise rotation. The mode bit (Cn) is set high for full-step mode and is set low for half-step mode. The mode and direction bits are latched and can be set by the SD command or the SA command. The rising edge of the clock bit (Dn) causes the control logic to advance to the next step. This bit can be controlled by the PULD command or by forcing the bit low and then high using the SD or SA commands. Effective ramping of the motor velocity can only be achieved by using the RAMP command.

MOTOR AND POWER CONNECTIONS

The stepper motor card applies the drive voltage to the center tap of a unipolar motor and grounds each leg as needed. There are two power connections, one for the "hold" voltage and the other for the "boost" voltage. The boost voltage connector is second from the top and the hold voltage connector is third from the top. Each power connector is a three terminal connector; the middle contact is the positive voltage and the two outer contacts are connected to earth ground.

It is possible to use the MDAS 5 Volt supply for the "hold" voltage and the MDAS +15 Volt supply for the "boost voltage. The available current from the MDAS supplies is 1.5 Amperes from the 5 Volt supply and 0.8 Amperes from the +15 Volt supply. For higher current needs, the user must supply the required voltage sources. The jumpers for connecting the MDAS supplies are shown in Figure E-2.

The top and bottom connectors are the motor winding connectors. The center terminal of each connector should be connected to the center tap of the stepper motor. When the motor is at rest the "hold" voltage is present at this connector. When the motor is being stepped, the "boost" voltage is present as each step is initiated and then returns to the "hold" voltage after a short time fixed by jumpers on the stepper card. The outer connectors are the four motor phases. Phases 1 and 2 are on the top connector, phase 1 the top terminal and phase 2 is the bottom terminal. Phases 3 and 4 are on the bottom connector, phase 3 is the top terminal and phase 4 is the bottom terminal.

The direction of rotation depends on the connection of the winding wires. The motor driver is set up to generate the pulse sequence shown in figure E-1. Each phase is sequenced according to the direction requested. The '+' sign indicates that the phase is energized and the '-' sign indicates that the phase is not energized. A phase is energized by switching the phase lead to ground, thus applying the "hold" and/or "boost" voltage across the phase. This is done internally with power transistors.

Direction Table for Half and Full Step

	clockwise -->>					<<-- counter clockwise								
phase 1	+	+	-	-	+	+	+	+	-	-	-	-	-	+
phase 2	-	-	+	+	-	-	-	-	-	+	+	+	-	-
phase 3	+	-	-	+	+	+	-	-	-	-	-	+	+	+
phase 4	-	+	+	-	-	-	-	+	+	+	-	-	-	-
	full step mode					half step mode								

Figure E-1.

THE BOOST VOLTAGE

The boost voltage is optional and does not have to be used. If the boost voltage is present, it is applied to the center tap for a fixed amount of time each time the motor steps to the next position. After the fixed amount of time has elapsed, the hold voltage is again applied to the motor. The increased potential from the boost voltage causes the current in the motor to build up faster than by just applying the hold voltage. The performance of the motor depends largely on how fast the motor currents can be switched. The larger the boost voltage, the faster the motor currents will change.

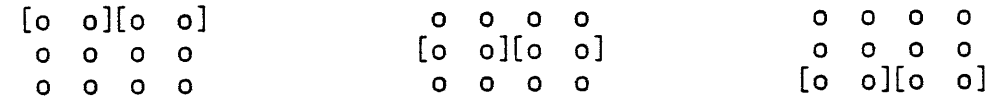
The use of a boost voltage allows for the use of a hold voltage lower than the rated winding voltage. This reduces the overall power disipated in the motor. Since the holding torque is generally quite a bit higher than the running torque, motor performance is not sacrificed by the reduction of the hold voltage.

The amount of time that the boost voltage is applied is controlled by jumper connections on the card. For optimum performance, the duration of the boost voltage should be close to the natural time constant of the winding. The natural time constant of the winding is found by dividing the winding inductance by the winding resistance.

BOOST ON-TIME CONFIGURATION

The boost voltage on-time is selected by a set of jumpers at the top edge of the stepper card. There are four on-times that can be selected: 3.3, 2.1, 1.3, and 0.8 milliseconds. The 3.3 millisecond on-time is selected by leaving off both of the jumpers. The other on-times are selected according to the diagram in Figure E-2. Both jumpers should be at the same position.

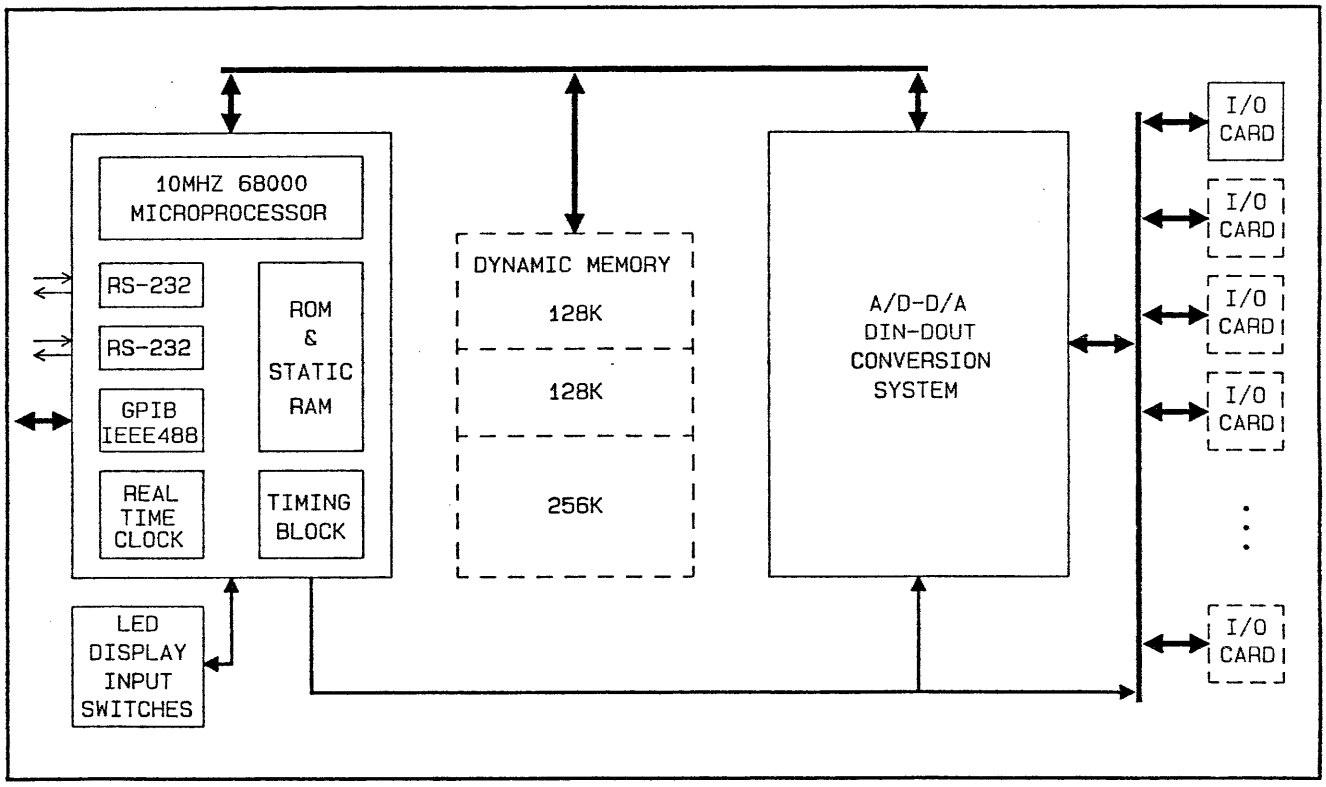
Boost On-time Jumper Configuration



0.8 milliseconds	1.3 milliseconds	2.1 milliseconds

Figure E-2.

SYSTEM OVERVIEW



MDAS-Opt A1B4 SINGLE ENDED ANALOG INPUT

General purpose input...

MAJOR FEATURES:

4 single ended analog input channels

4 BNC type connectors

Ground/shield connection

Input impedance of 10E+12

Input range of +/-10 V

Fast settling time

Excellent linearity and accuracy

Careful grounding considerations

APPLICATIONS:

General purpose input

Transient recording

Peak detection

Multi-channel analysis

Voice synthesis/recognition

Optical input/scanning

Pattern recognition

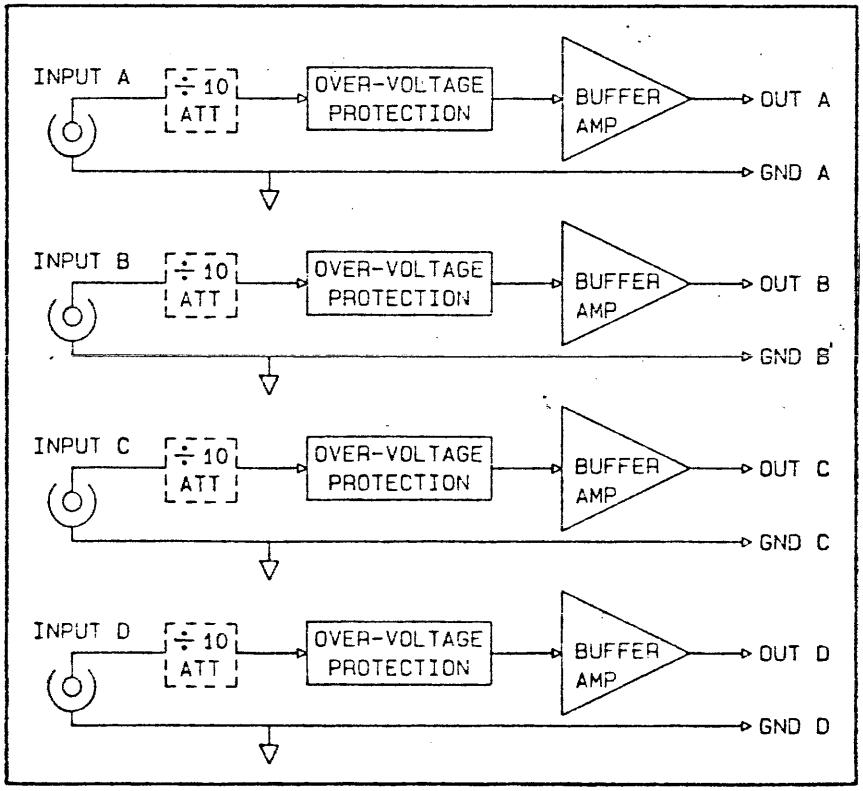
Acoustic and vibration testing

Seismic data acquistion and analysis

Signal processing

SPECIFICATIONS:

Input channels	4
Type	single ended
Range	±10 V
Input impedance	10E+12 ohm
Small signal BW	100 kHz
Settling time (10V step)	2 usec.
Linearity	±0.015%



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt A2B4/A2S4 HIGH LEVEL DIFFERENTIAL ANALOG INPUT
MDAS-Opt A3B4/A3S4 HIGH LEVEL DIFFERENTIAL ANALOG INPUT W/HOLD

Precision instrumentation amplifier...

MAJOR FEATURES:

- 4 differential analog input channels
- 4 floating screw lug type connectors
- Ground/shield connection
- Low input noise
- High CMRR
- Manually selectable input ranges
- Excellent linearity and accuracy
- Careful grounding considerations

APPLICATIONS:

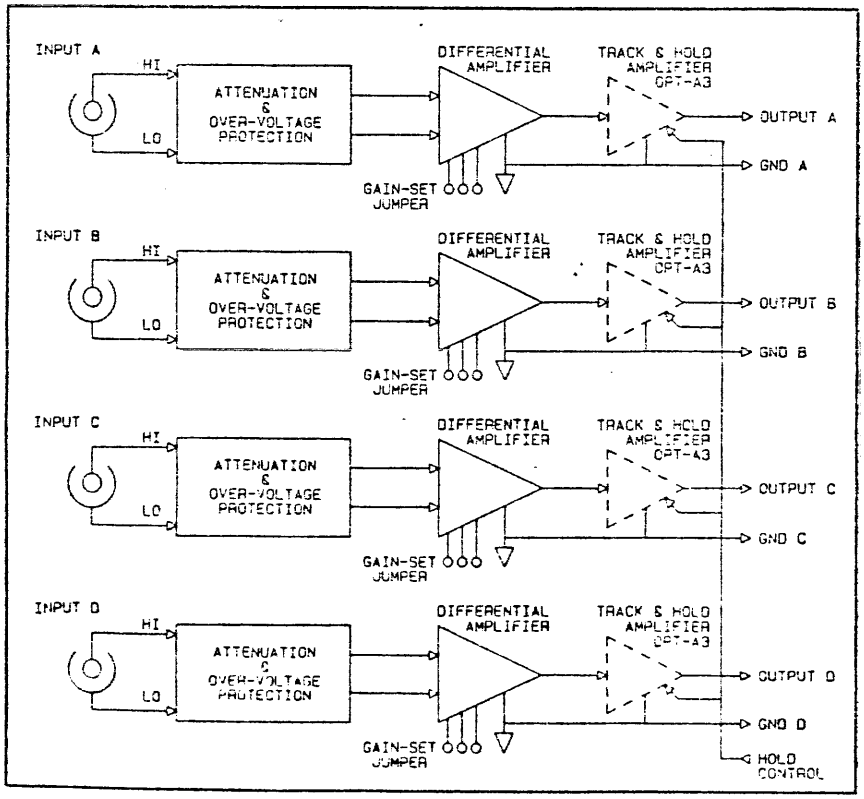
- General purpose differential input
- Transient recording
- Process control
- Transducer interface
- Multi-channel analysis
- Voice synthesis/recognition
- Acoustic and vibration testing
- Seismic data acquistion and analysis
- Signal processing

SPECIFICATIONS:

Input channels	4
Type	Differential
Manually selectable ranges (volts)	±100
Input impedance (Mohms)	1
CMRR (dB)	74
Small signal BW (kHz)	1000
Settling time (10V step, usec)	15
Common mode range	±100 Volts
Linearity	±0.003%

MDAS-Opt A3 Group hold feature

Aperture delay	200 nsec.
Aperture time	50 nsec.
Aperture uncertainty	5 nsec.



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt A4B4/A4S4 HIGH LEVEL DIFFERENTIAL ANALOG INPUT
MDAS-Opt A5B4/A5S4 HIGH LEVEL DIFFERENTIAL ANALOG INPUT W/HOLD

Precision instrumentation amplifier...

MAJOR FEATURES:

- 4 differential analog input channels
- 4 floating screw lug type connectors
- Ground/shield connection
- Low input noise
- High CMRR
- Manually selectable input ranges
- Excellent linearity and accuracy
- Careful grounding considerations

APPLICATIONS:

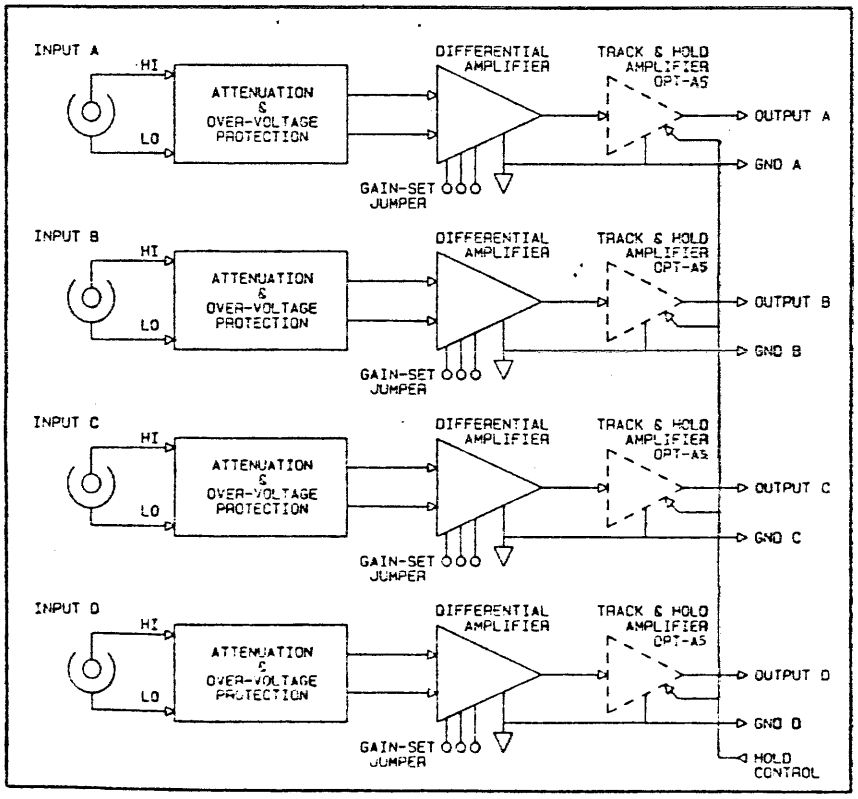
- General purpose differential input
- Transient recording
- Process control
- Transducer interface
- Multi-channel analysis
- Voice synthesis/recognition
- Acoustic and vibration testing
- Seismic data acquistion and analysis
- Signal processing

SPECIFICATIONS:

Input channels	4
Type	Differential
Manually selectable ranges (volts)	±10
Input impedance (Ω ohms)	1
CMRR (dB)	74
Small signal BW (kHz)	1000
Settling time (10V step, usec)	15
Common mode range	±10 Volts
Linearity	±0.003%

MDAS-Opt A3 Group hold feature

Aperture delay	200 nsec.
Aperture time	50 nsec.
Aperture uncertainty	5 nsec.



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt A6B4/A6S4 LOW LEVEL DIFFERENTIAL ANALOG INPUT
MDAS-Opt A7B4/A7S4 LOW LEVEL DIFFERENTIAL ANALOG INPUT W/HOLD

Precision instrumentation amplifier...

MAJOR FEATURES:

- 4 differential analog input channels
- 4 floating screw lug type connectors
- Ground/shield connection
- Low input noise
- High CMRR
- Manually selectable input ranges
- Excellent linearity and accuracy
- Careful grounding considerations

APPLICATIONS:

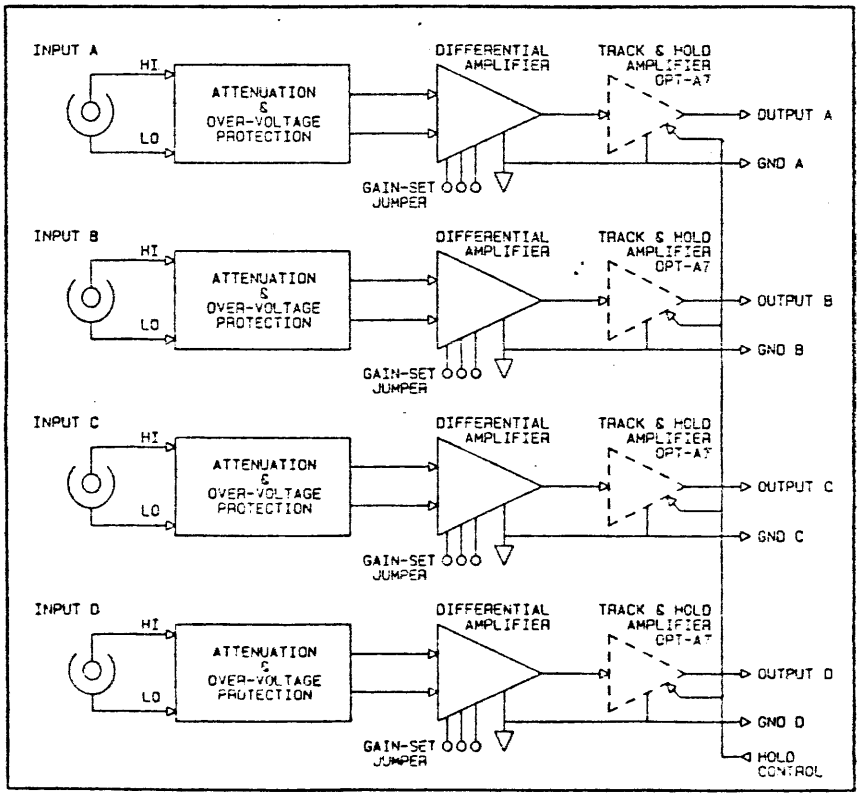
- Low level differential input
- Low level transducer input
- Process control
- Multi-channel analysis
- Voice synthesis/recognition
- Acoustic and vibration testing
- Seismic data acquistion and analysis
- Signal processing

SPECIFICATIONS:

Input channels	4
Type	Differential
Manually selectable ranges (volts)	±0.1
Input impedance (Gohms)	1
CMRR (dB)	105
Small signal BW (kHz)	150
Settling time (10V step, usec)	15
Common mode range	±10 Volts
Linearity	±0.003%

MDAS-Opt A3 Group hold feature

Aperture delay	200 nsec.
Aperture time	50 nsec.
Aperture uncertainty	5 nsec.



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt C1B4/C1S4 4-20 mA CURRENT TYPE ANALOG INPUT
MDAS-Opt C2B4/C2S4 4-20 mA CURRENT TYPE ANALOG INPUT W/HOLD

Precision instrumentation amplifier with 4-20 mA sense capability...

MAJOR FEATURES:

4 differential analog input channels

4-20 mA current input

Low input noise

High CMRR

Excellent linearity and accuracy

Careful grounding considerations

APPLICATIONS:

4-20 mA low noise reciever

Remote sensors

Temperature measurement

Pressure measurement

Factory automation

Energy management

Industrial process control

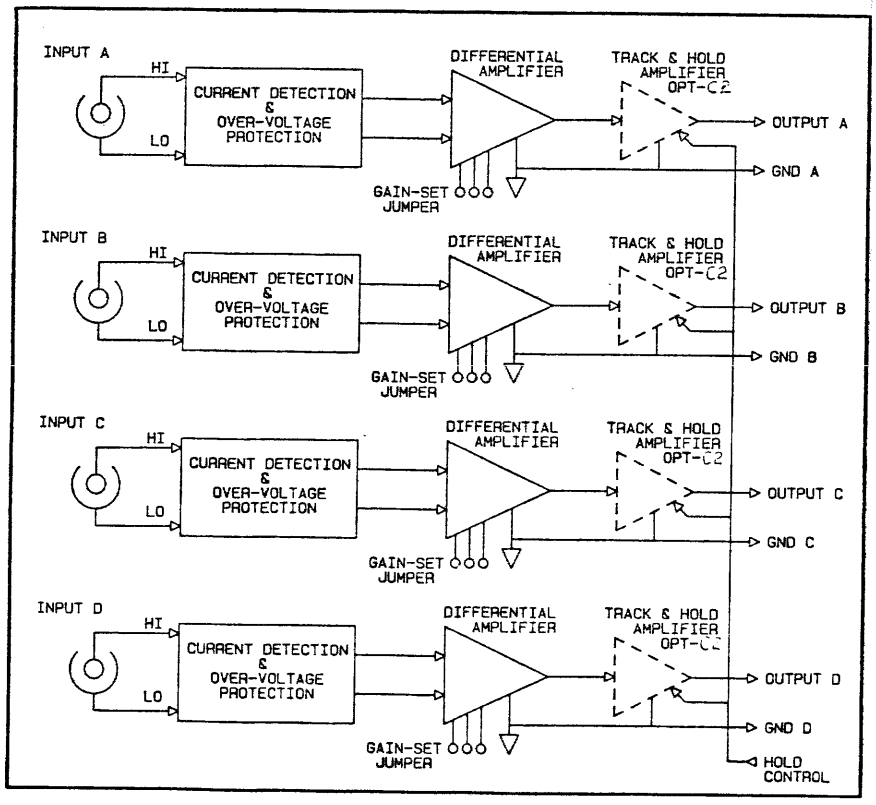
Acoustic and vibration testing

SPECIFICATIONS:

Input channels	4
Type	Differential input with current to voltage converter
Range	4-20 mA
Input impedance (ohms)	50
CMRR (dB)	94
Small signal BW (kHz)	200
Settling time (16 mA step, usec)	20
Common mode range	+12 to -11 Volts
Linearity	±0.015%

MDAS-Opt A3 Group hold feature

Aperture delay	200 nsec.
Aperture time	50 nsec.
Aperture uncertainty	5 nsec.



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt S1B4 ±10 VOLT ANALOG OUTPUT

12 bit resolution...

MAJOR FEATURES:

4 analog output channels

4 BNC type connectors

Ground/shield connection

Excellent linearity and accuracy

APPLICATIONS:

General purpose setpoint control

Waveform generation

Laboratory

Motor control

Motion control

Medical

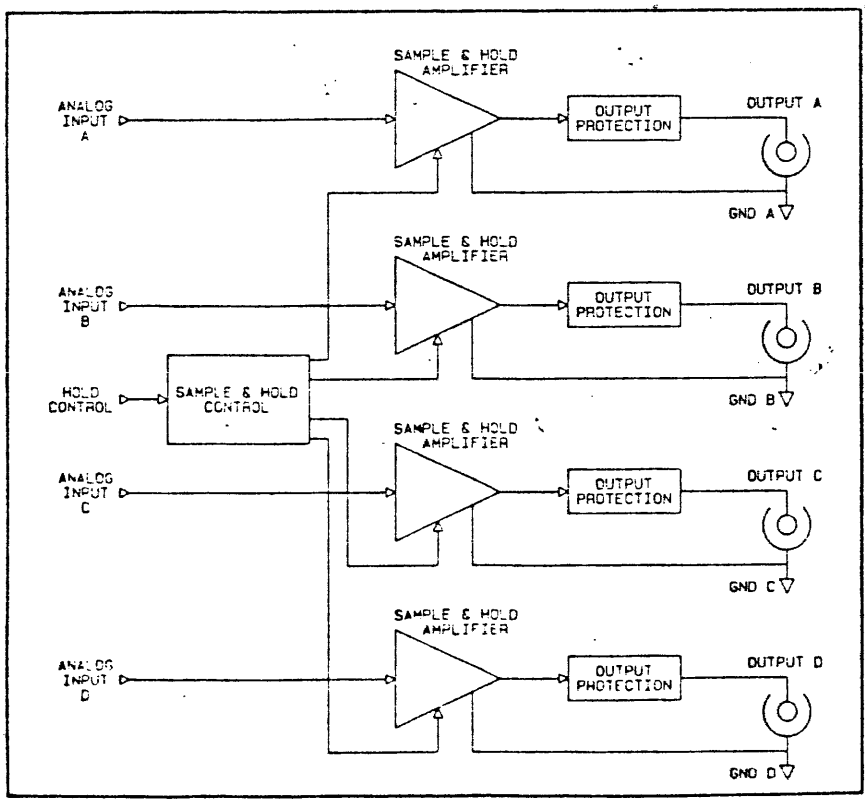
Industrial process control

Voice synthesis

Signal processing output

SPECIFICATIONS:

Output channels	4
Type	Voltage
Range (volts)	±10
Resolution	12 bits (0.3 mV)
linearity	±.003%
Droop rate	0.1 mV/msec.
Output impedance (ohms)	100
Settling time	
(10V step, .002%)	10 usec.
(10V step, .02%)	2 usec.



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt R1S4 110/220 AC DETECTION

Digital monitoring...

MAJOR FEATURES:

4 digital input channels

4 floating BNC type connectors

Ground/shield connection

APPLICATIONS:

AC voltage detection

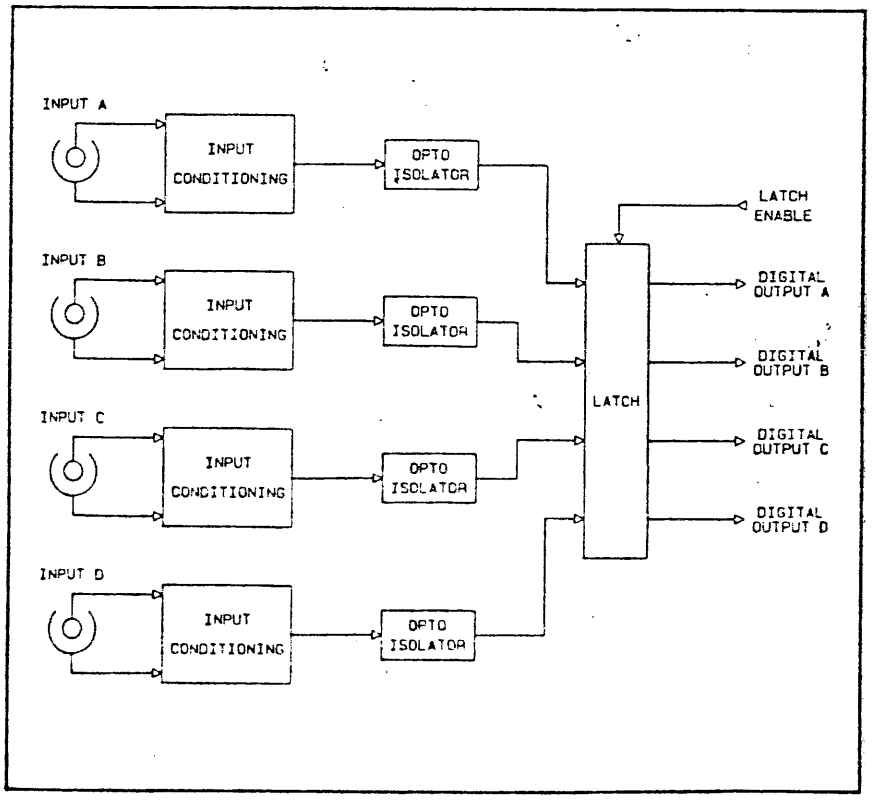
Process control

Power failure testing

Switched circuit testing

SPECIFICATIONS:

Input channels	4
Type	Floating digital inputs with parallel latches
Range	±250 Volts
Input impedance (kohms)	20
Frequency range	DC to 1 MHz



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt R3B4 TTL DIGITAL INPUT

Digital input...

MAJOR FEATURES:

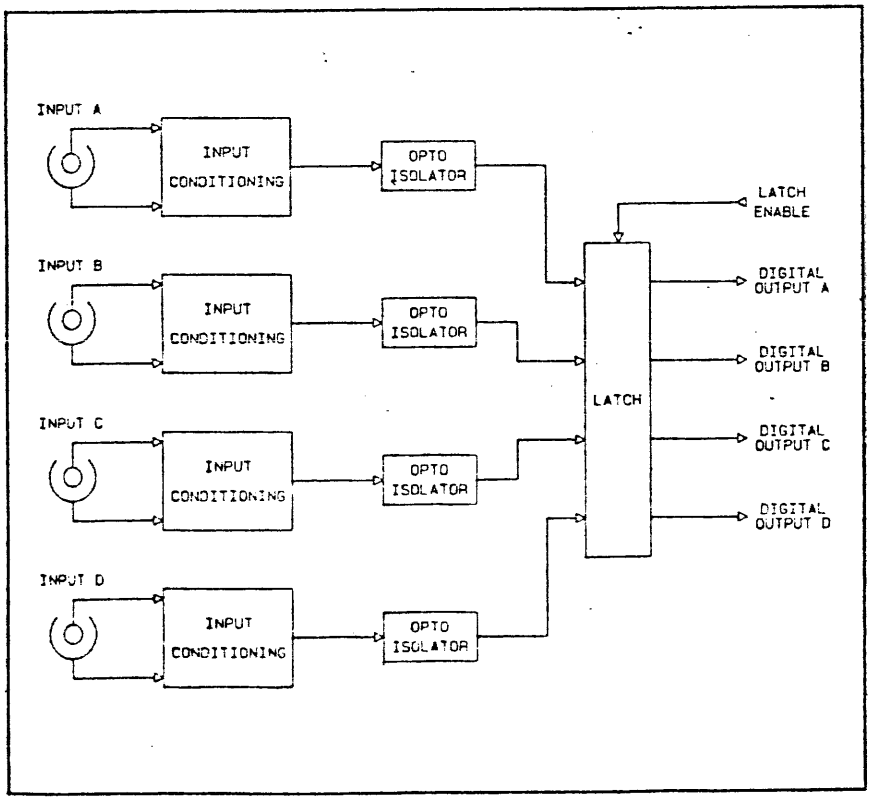
- 4 digital input channels
- 4 floating BNC type connectors
- Optical isolation
- Parallel latching
- Ground/shield connection

APPLICATIONS:

- Digital input
- TTL compatible
- Trigger input for data acquisition
- Interrupt input
- Process control

SPECIFICATIONS:

Input channels	4
Type	Floating digital inputs with parallel latches TTL compatible
Optical isolation	2500 volts
Range	0 to +25 Volts
Input impedance	2k ohms
Square wave input frequency range	DC to 50 kHz
Pulse input min high or low time	10 usec



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt R8B2 PULSE/FREQUENCY MEASUREMENT

Frequency counter/totalizer...

MAJOR FEATURES:

2 input channels

Floating BNC type connectors

Optical isolation

DC to 5 MHz input detection range

Software adjustable period

Ground/shield connection

APPLICATIONS:

Frequency measurement

Pulse measurement

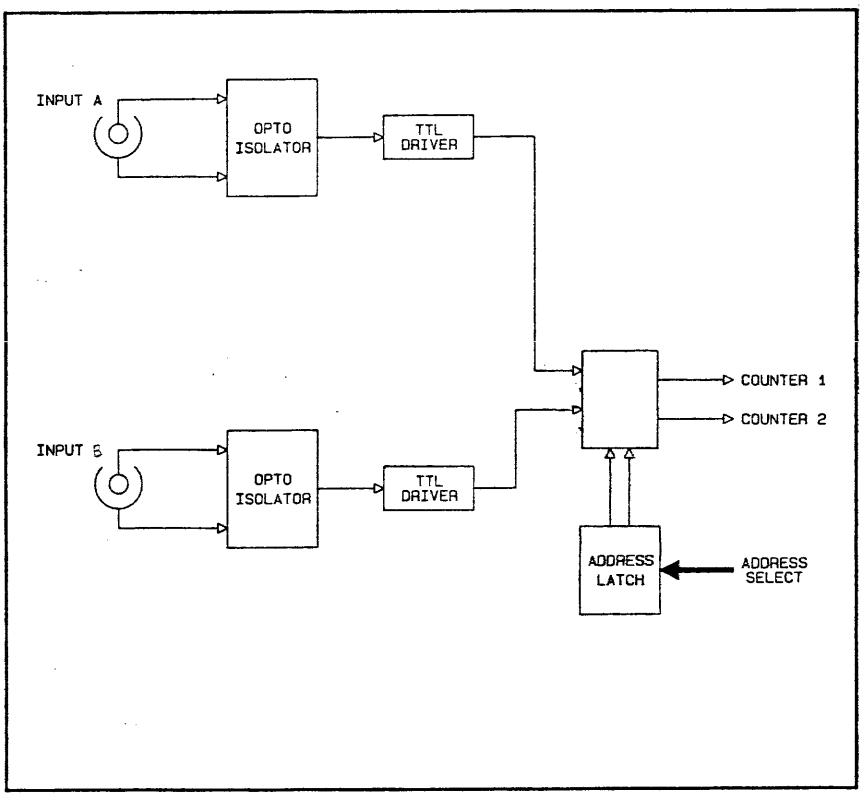
Differential frequency measurement

Fiber optic data transfer

Process control

SPECIFICATIONS:

Input channels	2
Type	Floating digital inputs TTL compatible
Optical isolation	2500 volts
Input voltage range	+2 to +25 Volts
Input impedance	100 ohms
Input threshold	2.5 V
Square wave input frequency range	DC to 5 MHz
Pulse width min high or low time	100 nsec



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt D1S4 MECHANICAL RELAY

Versatile mechanical relays...

MAJOR FEATURES:

4 SPDT mechanical relays

4 screw terminal connectors

500 V isolation

Low impedance contacts

APPLICATIONS:

General purpose switching

Process control

Motor control

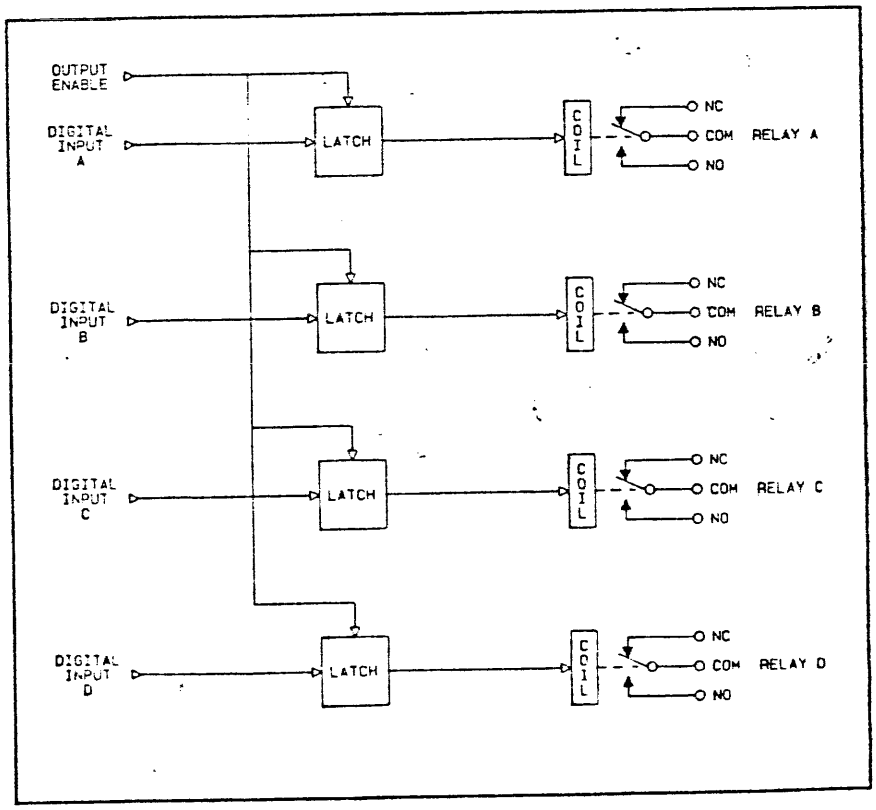
Signal routing

Control actuation

Alarm activation

SPECIFICATIONS:

Relays	4
Type	Single-pole, double-throw Parallel output strobe capability
Contact data	
Max switching voltage	220 VDC, 250 VAC
Max switching power	60 WDC. 120 VAAC
Max switching current	2 A AC or DC
Max carrying current	3 A AC or DC
Resistance	10 mohm initial
Isolation	500 V
Rated life	10E+8 operations
Attenuation	less than 0.1 dB to 1 MHz



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt D2S4 SOLID STATE AC RELAY

AC control...

MAJOR FEATURES:

4 SPST solid state relays

4 screw terminal connectors

2500 V isolation

APPLICATIONS:

AC power switching

Process control

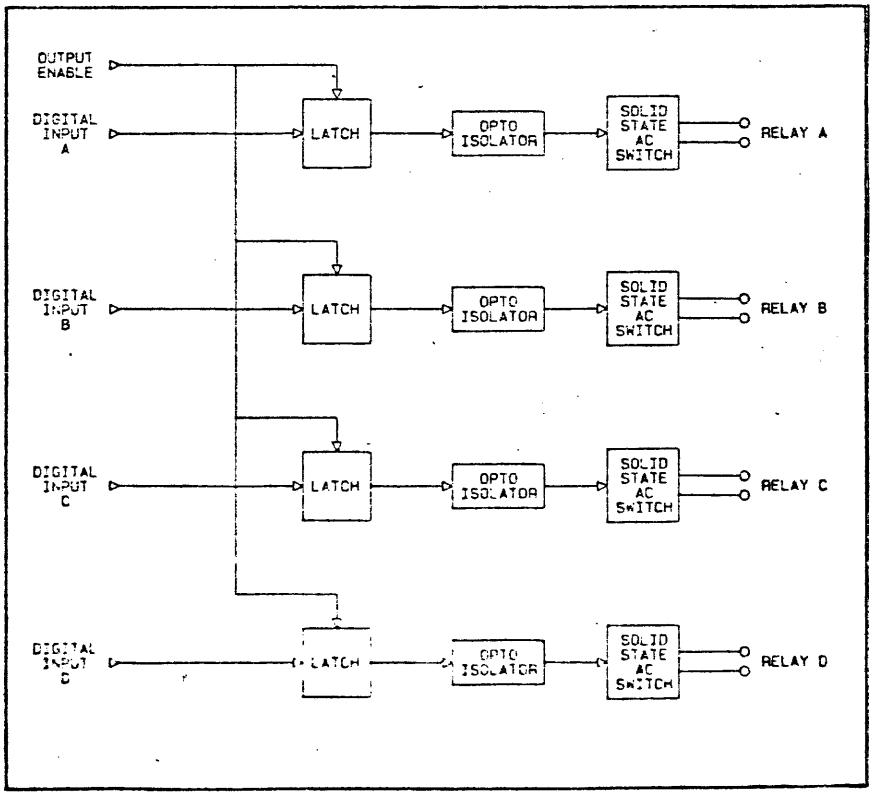
Motor control

Control actuation

Alarm activation

SPECIFICATIONS:

Relays	4
Type	Single-pole, single-throw Parallel output strobe capability
Max switching voltage	250 VAC
Max carrying current	8 A AC
Optical isolation	2500 V



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt D3S4 SOLID STATE DC RELAY

DC control...

MAJOR FEATURES:

4 SPST solid state relays

4 screw terminal connectors

50 kHz maximum output frequency

2500 V isolation

APPLICATIONS:

DC power switching

Process control

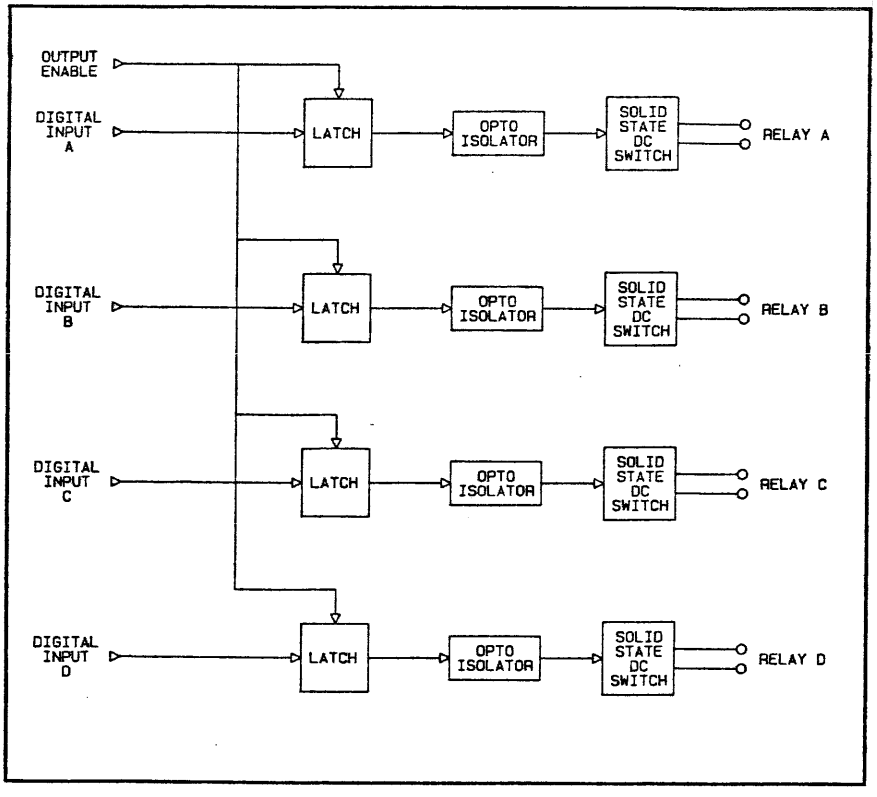
Motor control

Control actuation

Alarm activation

SPECIFICATIONS:

Relays	4
Type	Single-pole, single-throw
	Parallel output strobe capability
Max switching voltage	80 VDC
Max carrying current	1 A DC
Max output frequency	50 kHz
Min high or low time	10 usec
Optical isolation	2500 V



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

MDAS-Opt D4B4 SOLID STATE DC RELAY TTL COMPATIBLE

TTL driver...

MAJOR FEATURES:

4 SPST solid state relays

Screw terminal connectors

TTL compatible drivers

50 kHz maximum output frequency

2500 V isolation

APPLICATIONS:

DC switching

Logic enables and controls

Data transfer

Process control

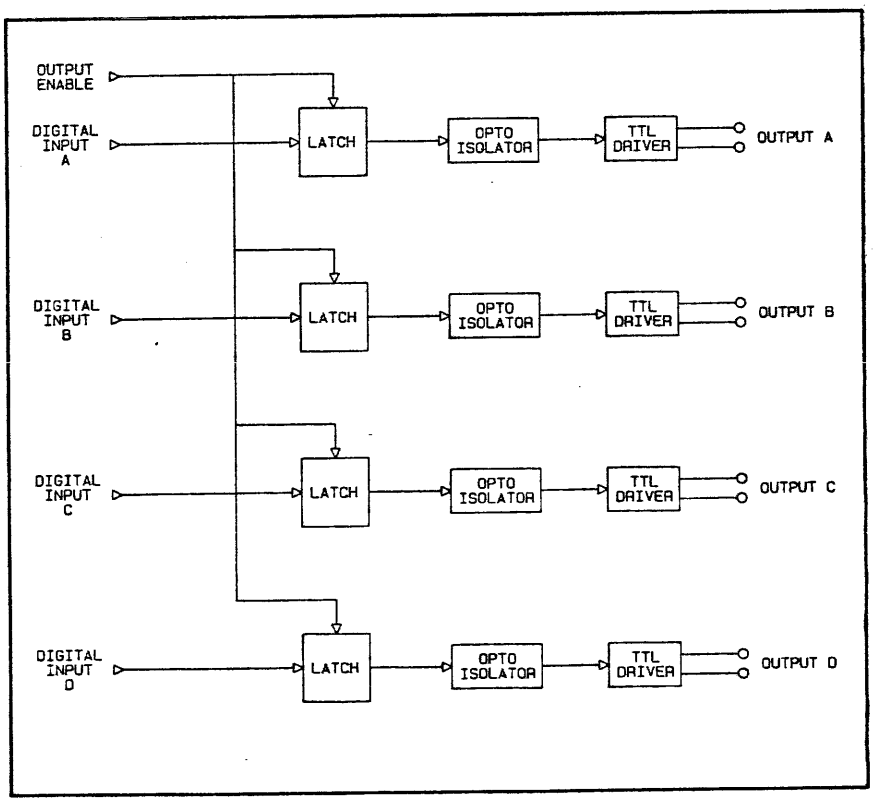
Motor control

Control actuation

Alarm activation

SPECIFICATIONS:

Relays	4
Type	Single-pole, single-throw
	Parallel output strobe capability
Max switching voltage	80 VDC
Max carrying current	1 A DC
Max output frequency	50 kHz
Min high or low time	10 usec
Optical isolation	2500 V



TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

Phone 801-224-6550
Telex 296438

Note to MDAS floppy disk users.

With a floppy disk it is possible to load an executable buffer from the floppy disk and have this file automatically run at power-up. The procedure to do this is as follows:

1. Store an executable buffer on the floppy disk with the file name "AUTO.RUN".

2. At power-up hold the RESET button down. When the RESET button is released the "AUTO.RUN" file (if it exists) on the floppy disk will be loaded into buffer 0 and run.

TRANSERA CORPORATION
PRODUCT RELEASE INFO

DATE: 26-MAR-85

MODEL NUMBER: GU1

MODEL DESCRIPTION: USER CONFIGURABLE CARD

MESSAGE: PRELIMINARY DOCUMENTATION FOR MODEL NUMBER GU1. MORE DETAILED INFORMATION WILL BE SENT WHEN AVAILABLE.

TRANSERA CORPORATION
PRODUCT RELEASE INFO

DATE: 26-MAR-85

MODEL NUMBER: R8B2

MODEL DESCRIPTION: PULSE OR FREQUENCY MEASUREMENT

MESSAGE: PRELIMINARY DOCUMENTATION FOR MODEL NUMBER R8B2.

A = TRIGGER INPUT

B = NOT USED

C = COUNTER INPUT

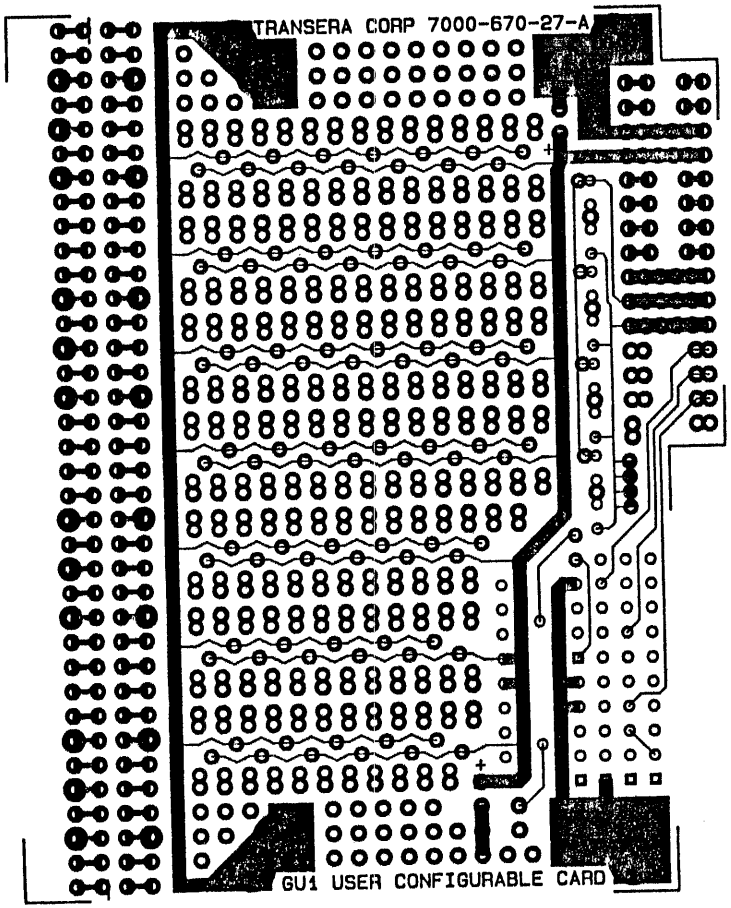
D = COUNTER INPUT

CHANNELS C&D USE AN INTERNAL 9513 COUNTER CHIP. FOR SLOWER GENERAL COUNTING, SEE OPERATORS MANUAL.

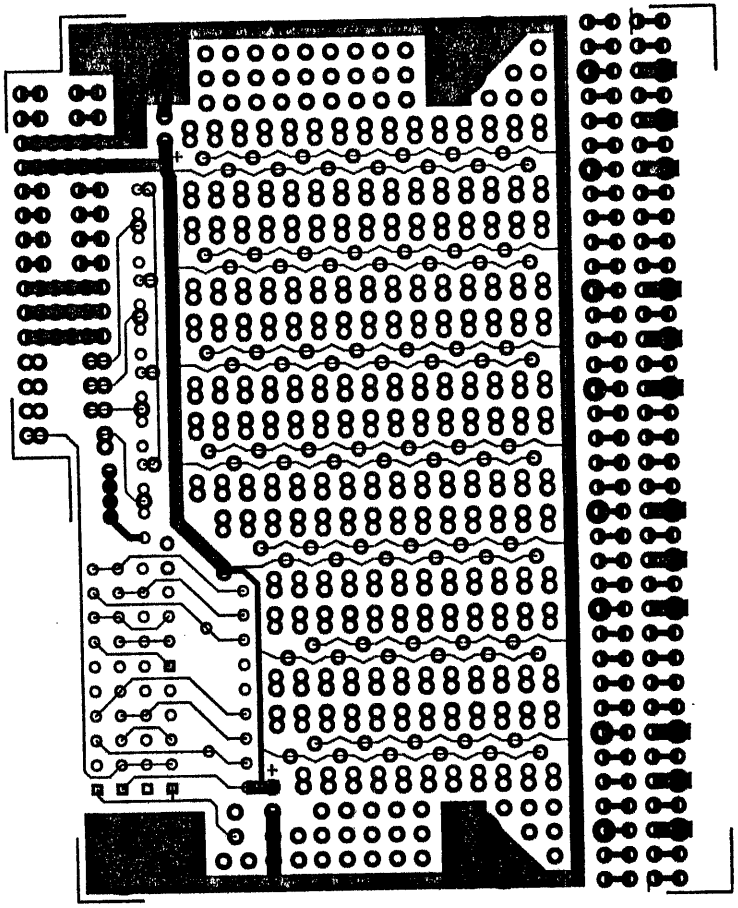
MORE DETAILED INFORMATION WILL BE SENT WHEN AVAILABLE.

I/O CONNECTOR DESIGNATION MARCH 7,1985

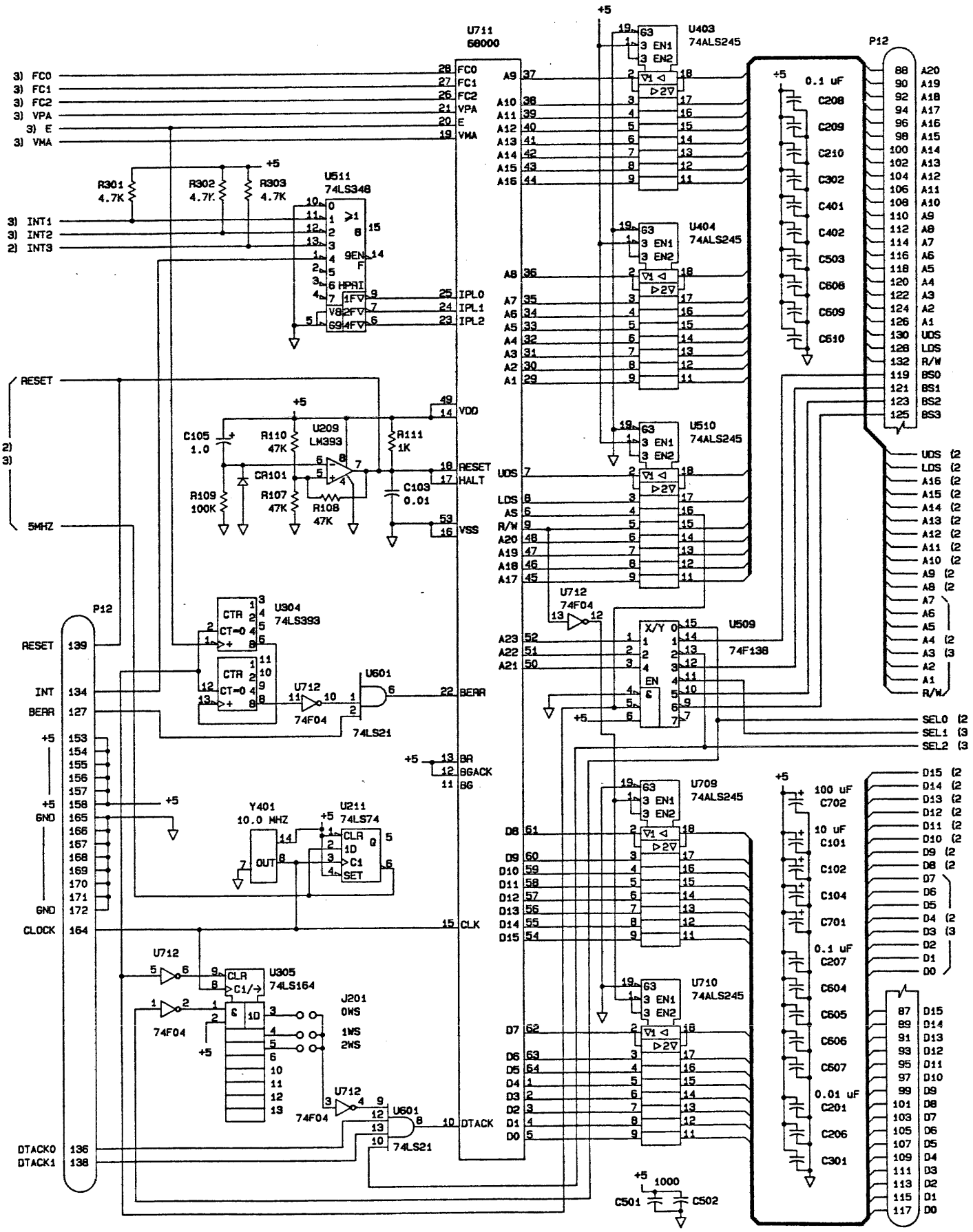
ODD PINS				EVEN PINS
29	TRIGGER OUT	o o	CNTR3	30
27	CNTR4	o o	HOLD(CNTR2)	28
25		o--o	DGND	26
23		o--o	+5 V	24
21	TRIGGER IN	o o		22
19		o o	D/A OUT	20
17	ADD1	o o	INTERRUPT	18
15	ADD0	o o	ADD6	16
13		o--o	+15 V	14
11		o--o	-15 V	12
9		o--o	AGND	10
7	S1	o o	SGND	8
5	S2	o o	SGND	6
3	S3	o o	SGND	4
1	S4	o o	SGND	2



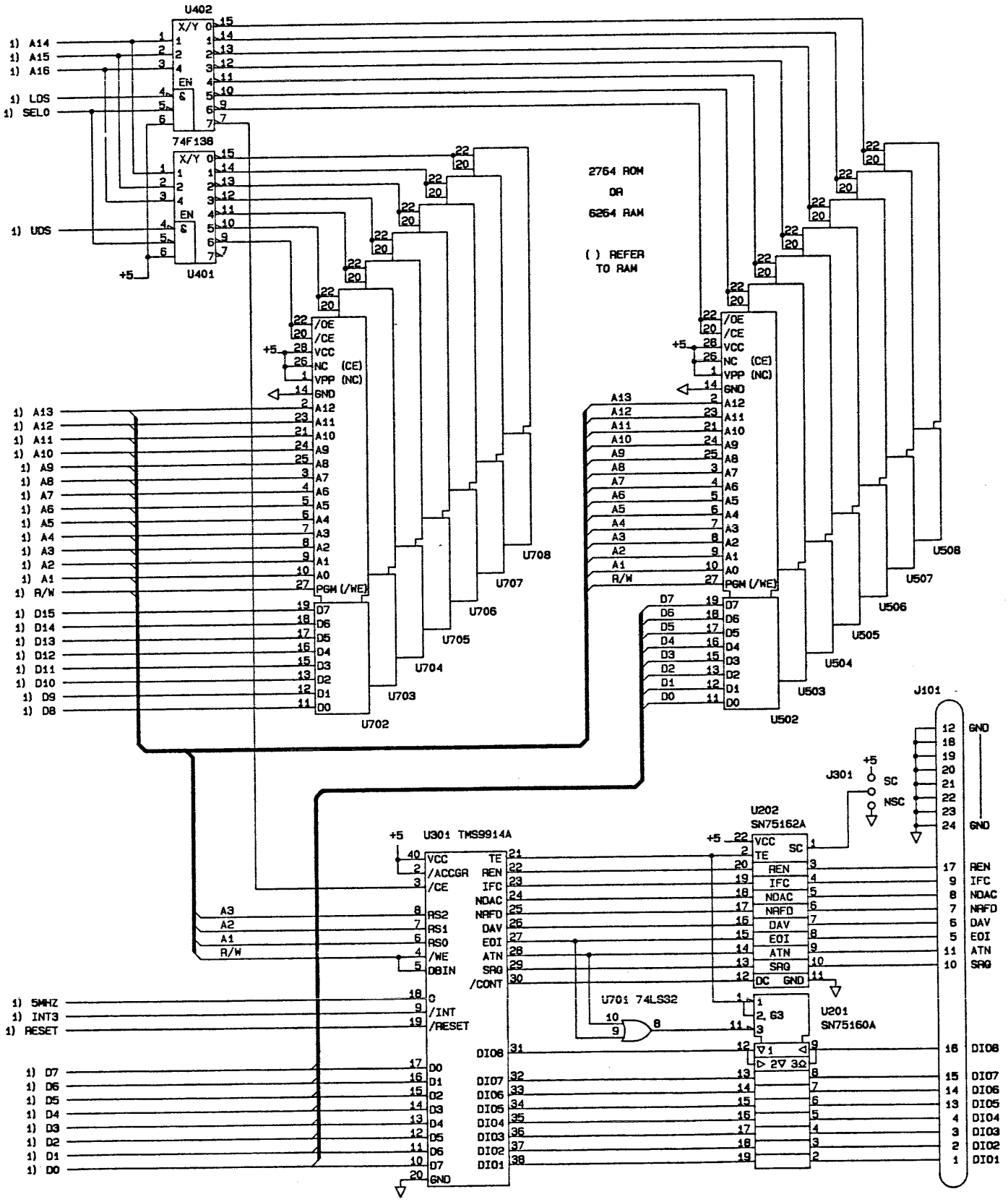
USER CONFIGURABLE BOARD TOP



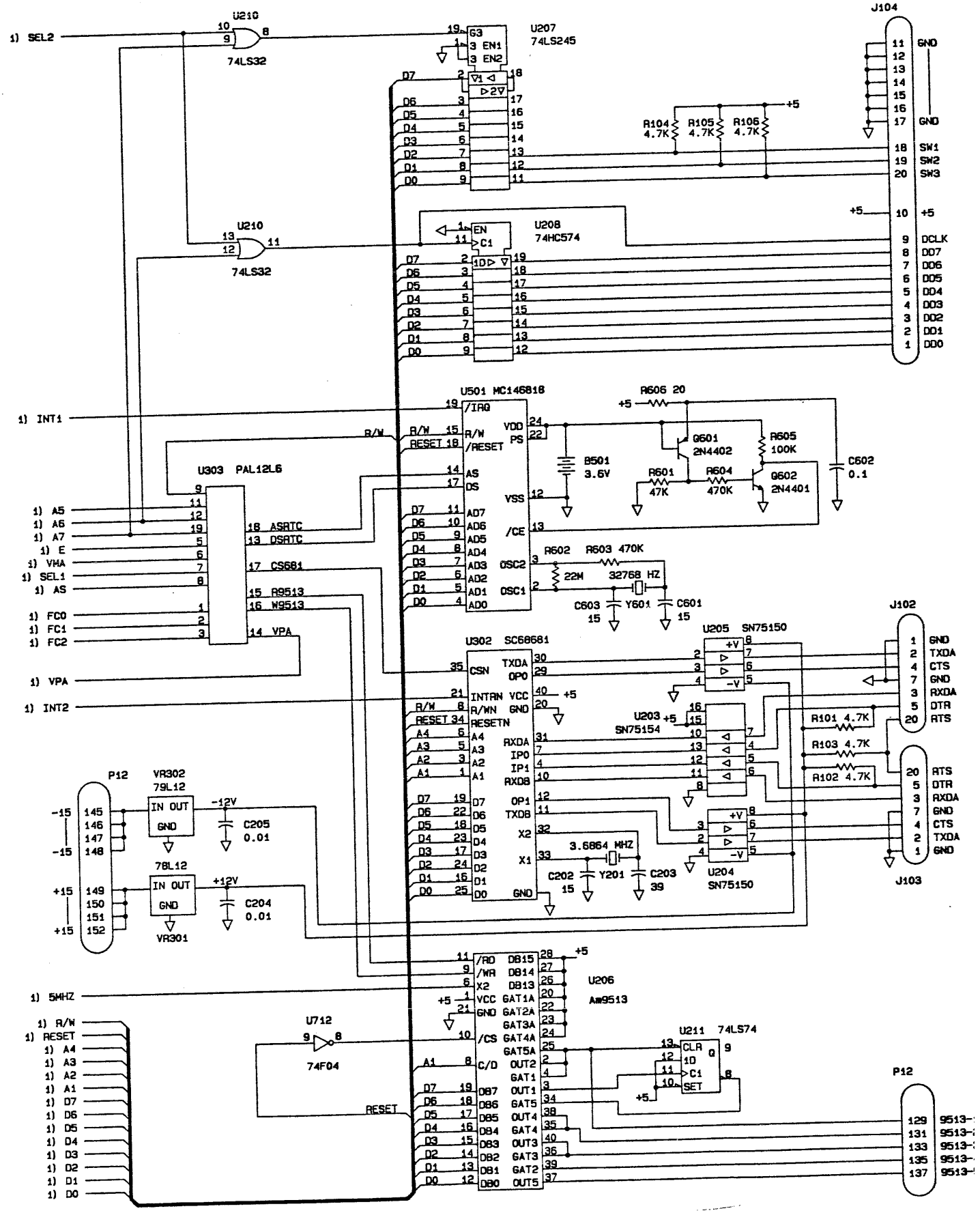
USER CONFIGURABLE BOARD BOTTOM



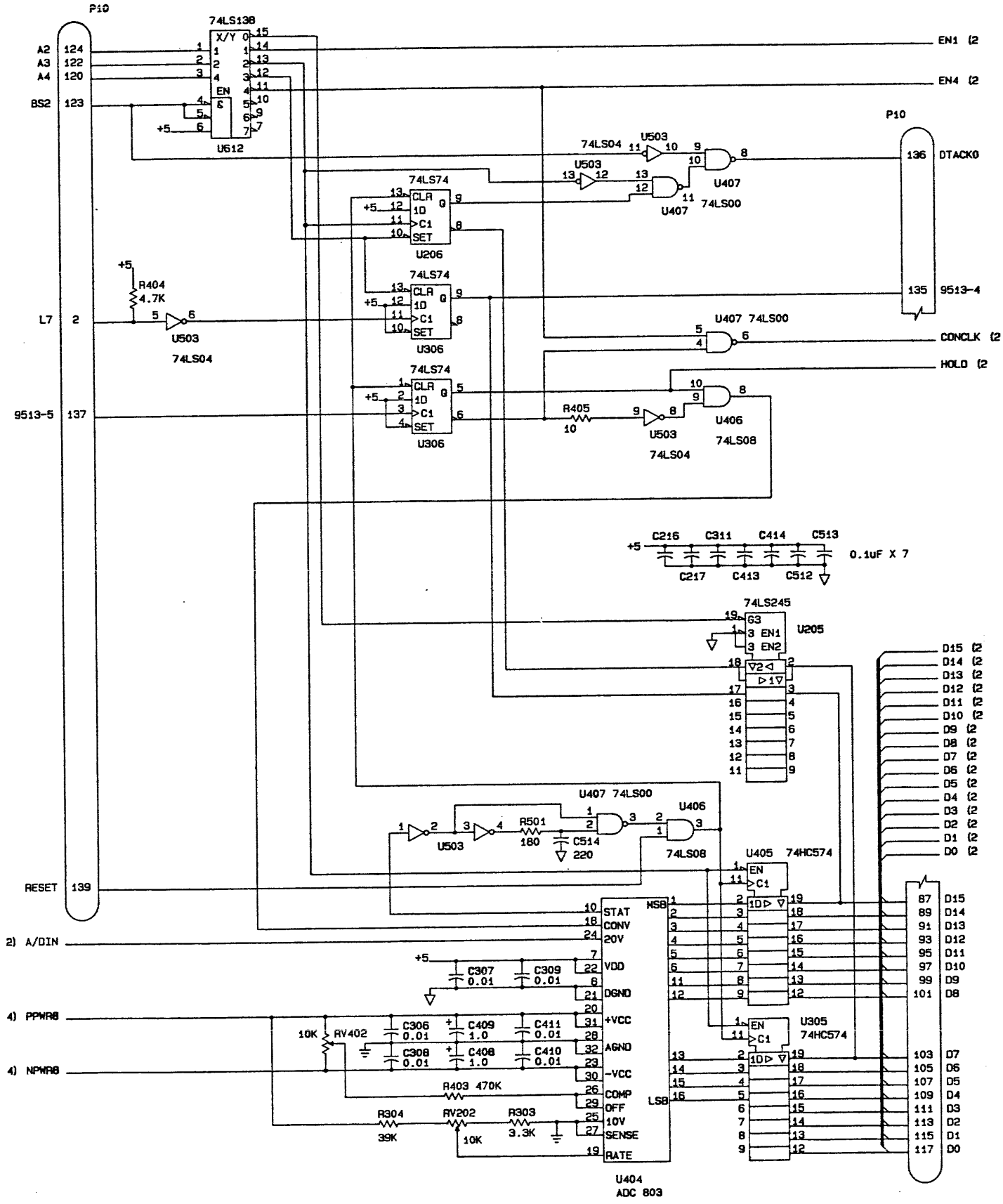
PRODUCT	NAME 7000-670-02-C	VERSION	PAGE
7000 MDAS	PROCESSOR BOARD	C	1 OF 3



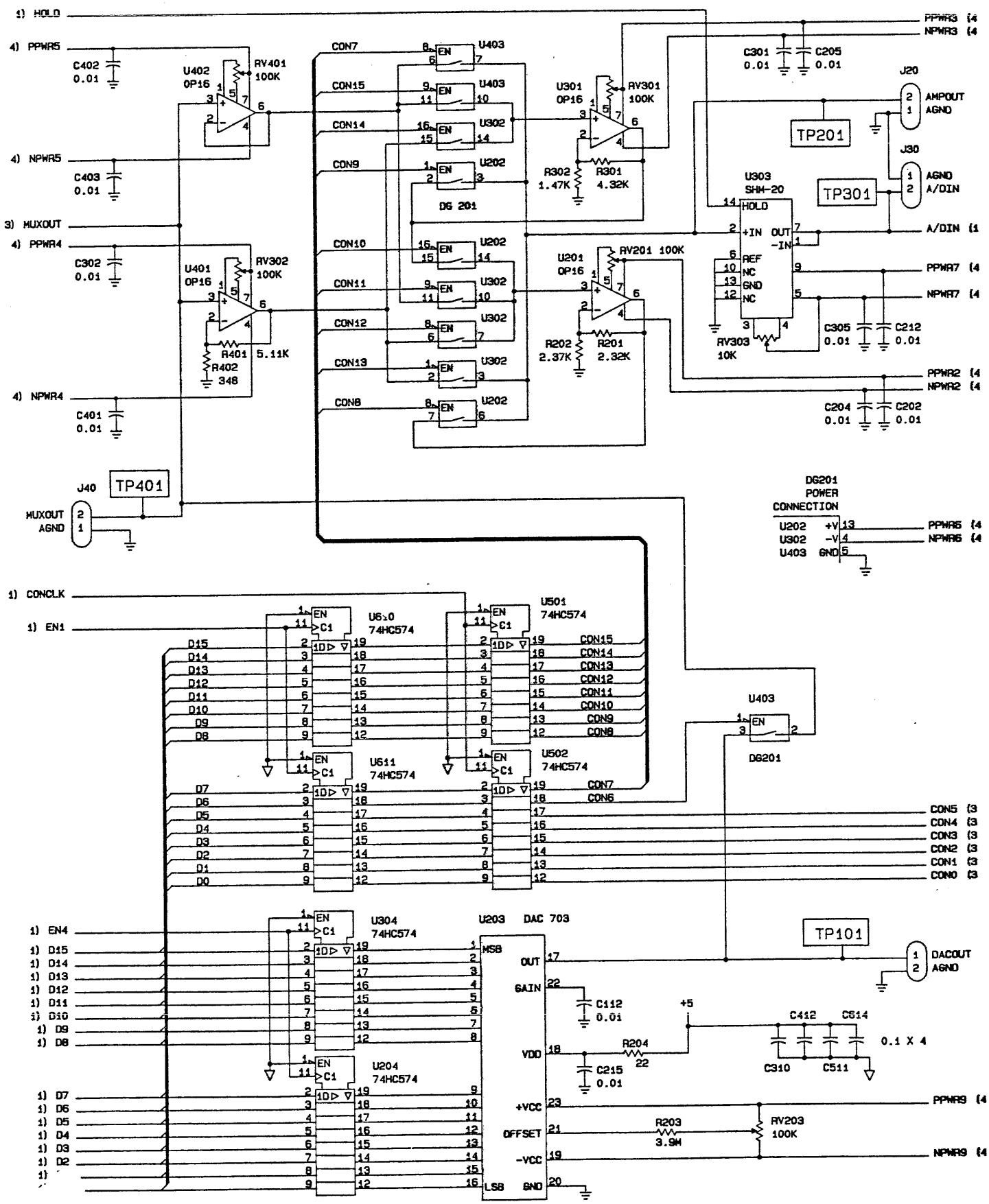
PRODUCT	NAME 7000-670-02-C	VERSION	PAGE
7000 MDAS	PROCESSOR BOARD	C	2 OF 3



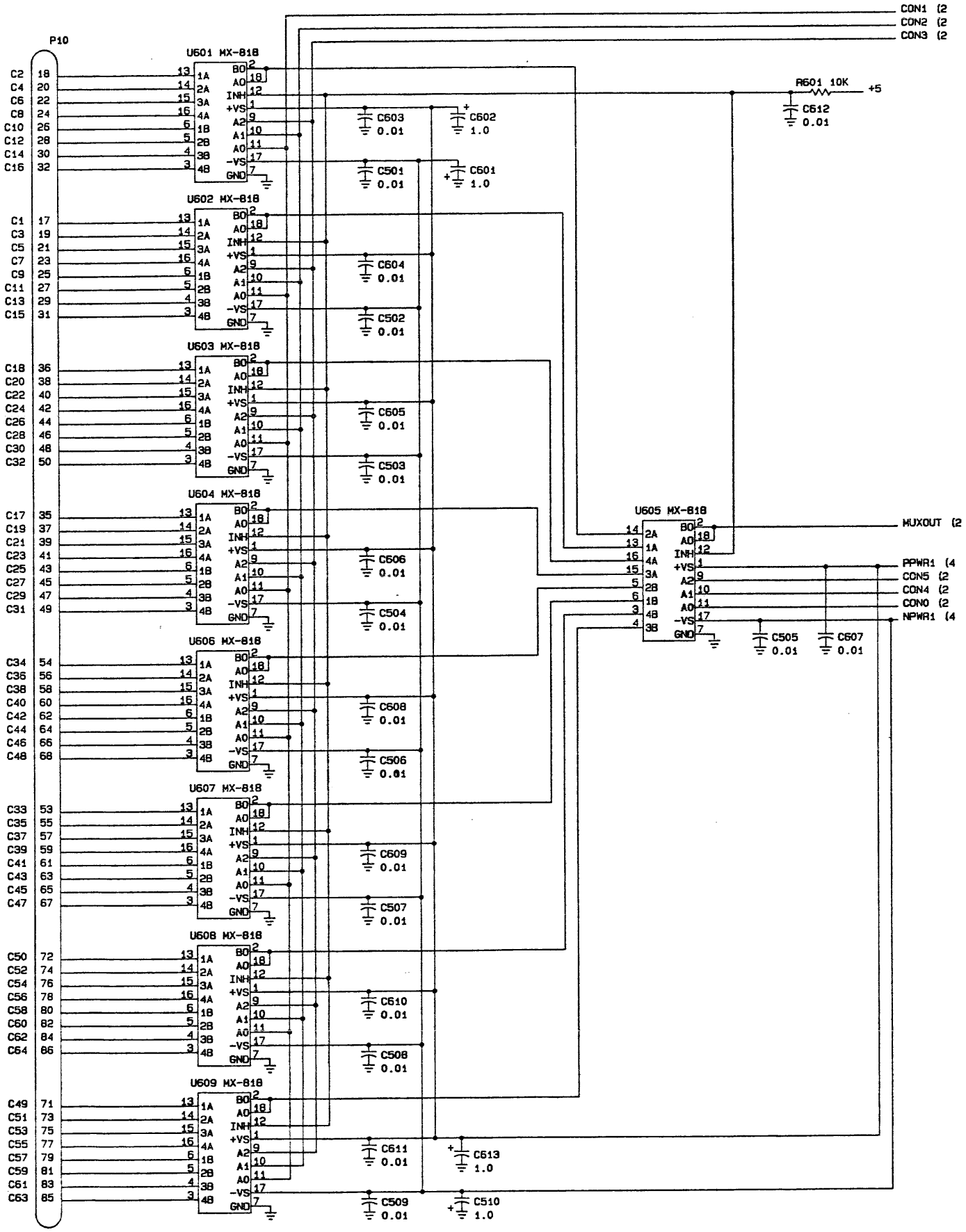
PRODUCT	NAME 7000-670-02-C	VERSION	PAGE
7000 MDAS	PROCESSOR BOARD	C	3 OF 3



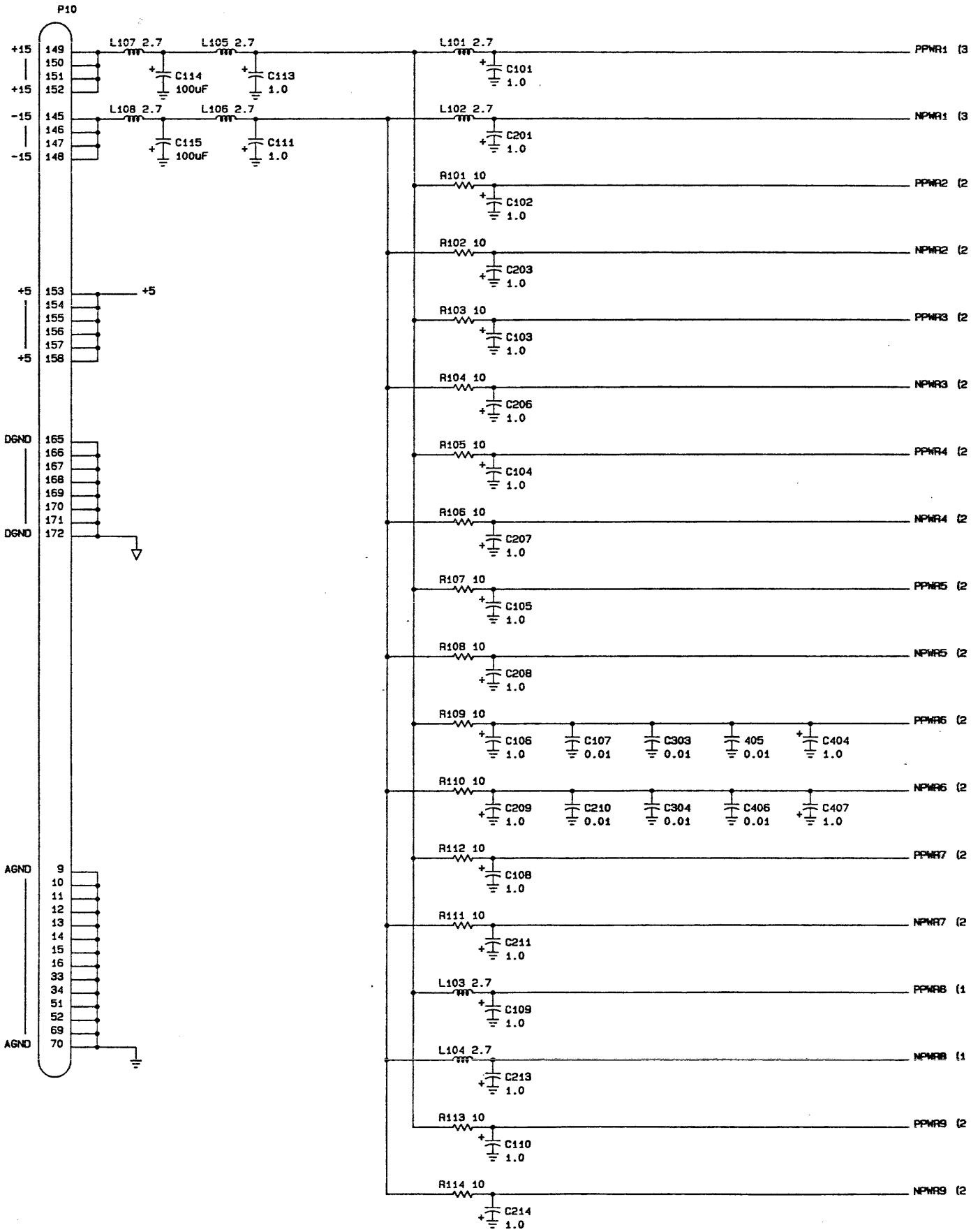
PRODUCT	NAME 7000-670-03-C	VERSION	PAGE
7000 MDAS	12 BIT 400KHZ ANALOG BOARD	CS4-A	1 OF 4



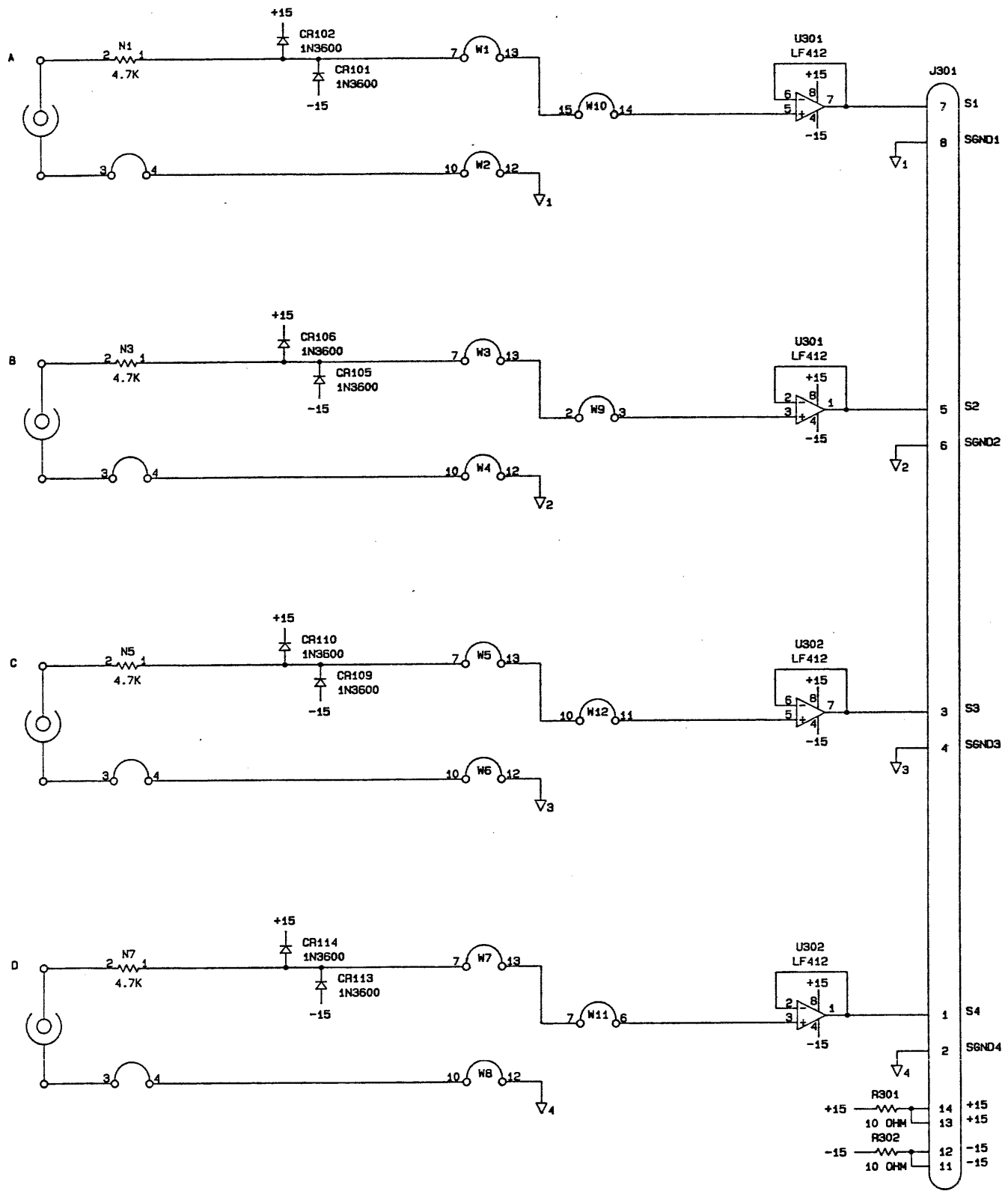
PRODUCT	NAME 7000-670-03-C	VERSION	PAGE
7000 MDAS	12 BIT 400KHZ ANALOG BOARD	CS4-A	2 OF 4



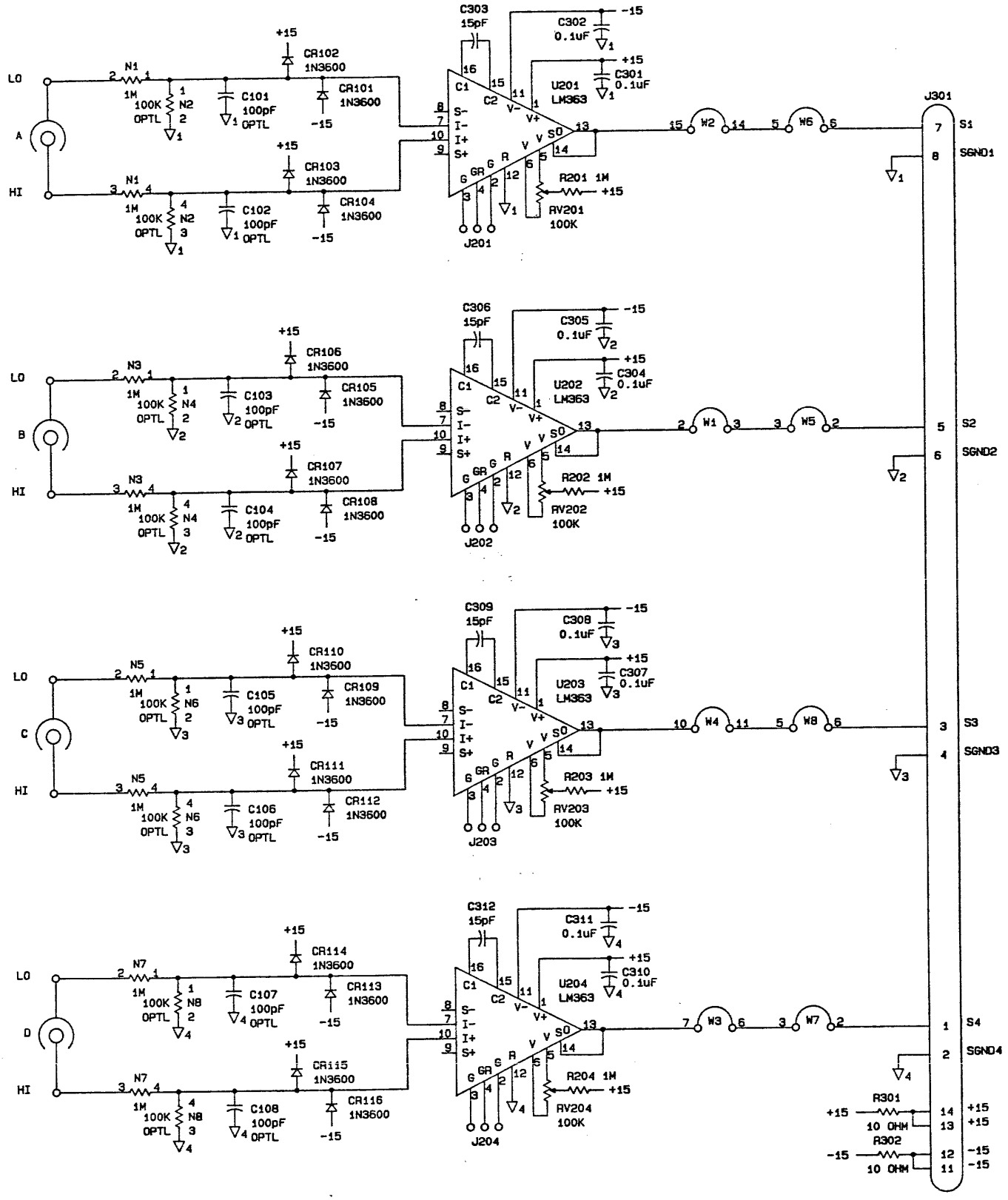
PRODUCT	NAME 7000-670-03-C	VERSION	PAGE
7000 MDAS	12 BIT 400KHZ ANALOG BOARD	CS4-A	3 OF 4



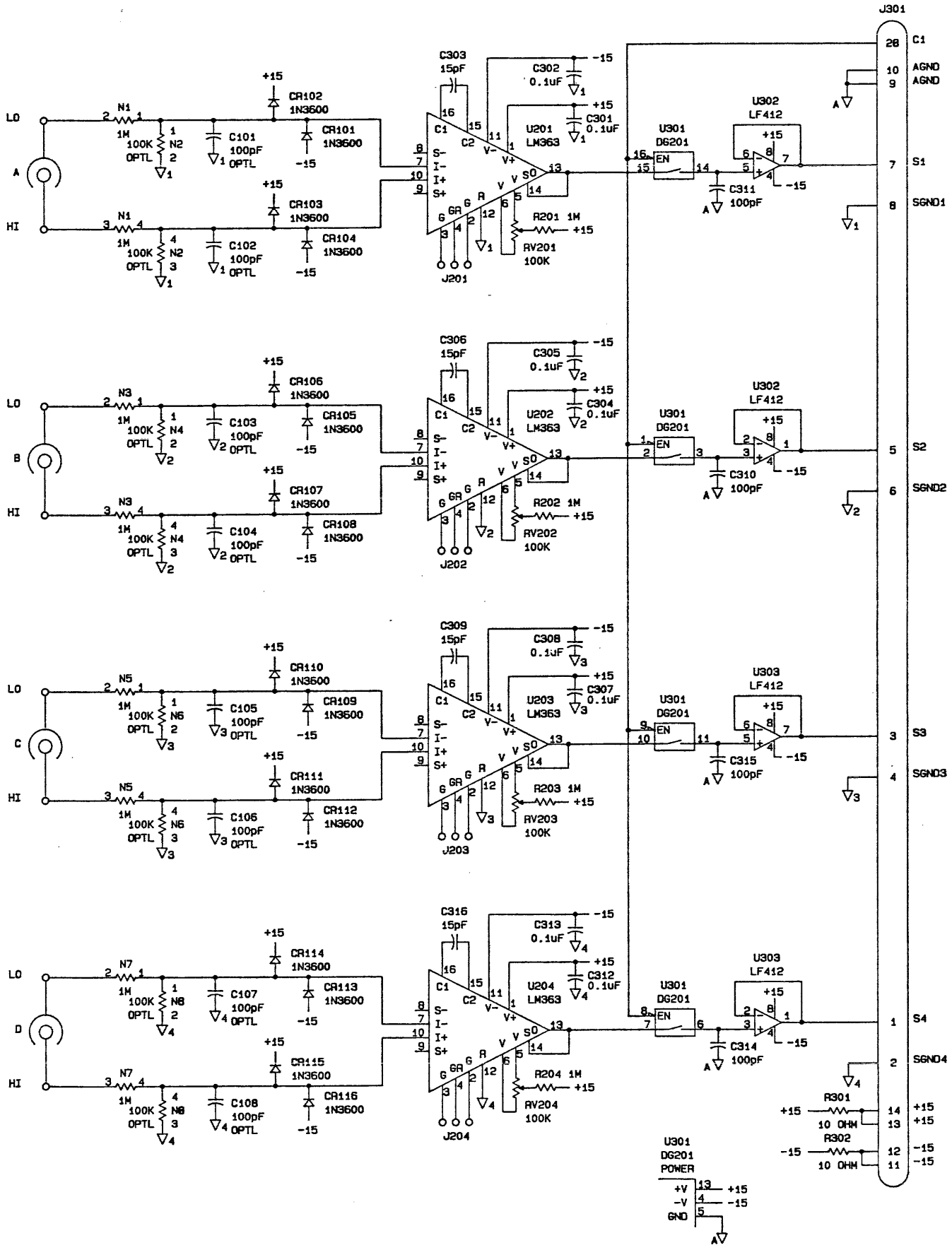
PRODUCT	NAME 7000-670-03-C	VERSION	PAGE
7000 MDAS	12 BIT 400KHZ ANALOG BOARD	CS4-A	4 OF 4



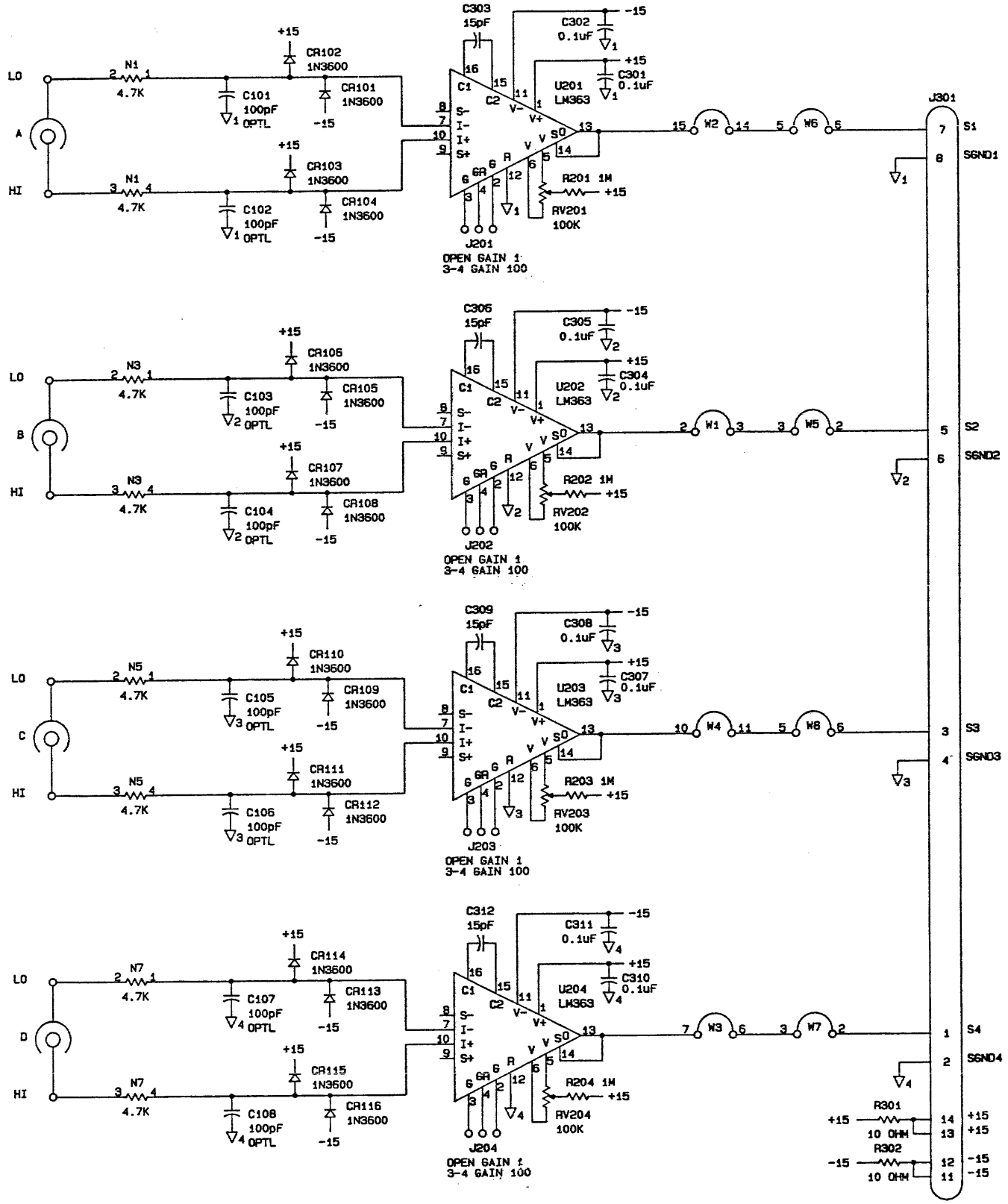
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	SINGLE ENDED ANALOG INPUT BOARD	A1	1 OF 1



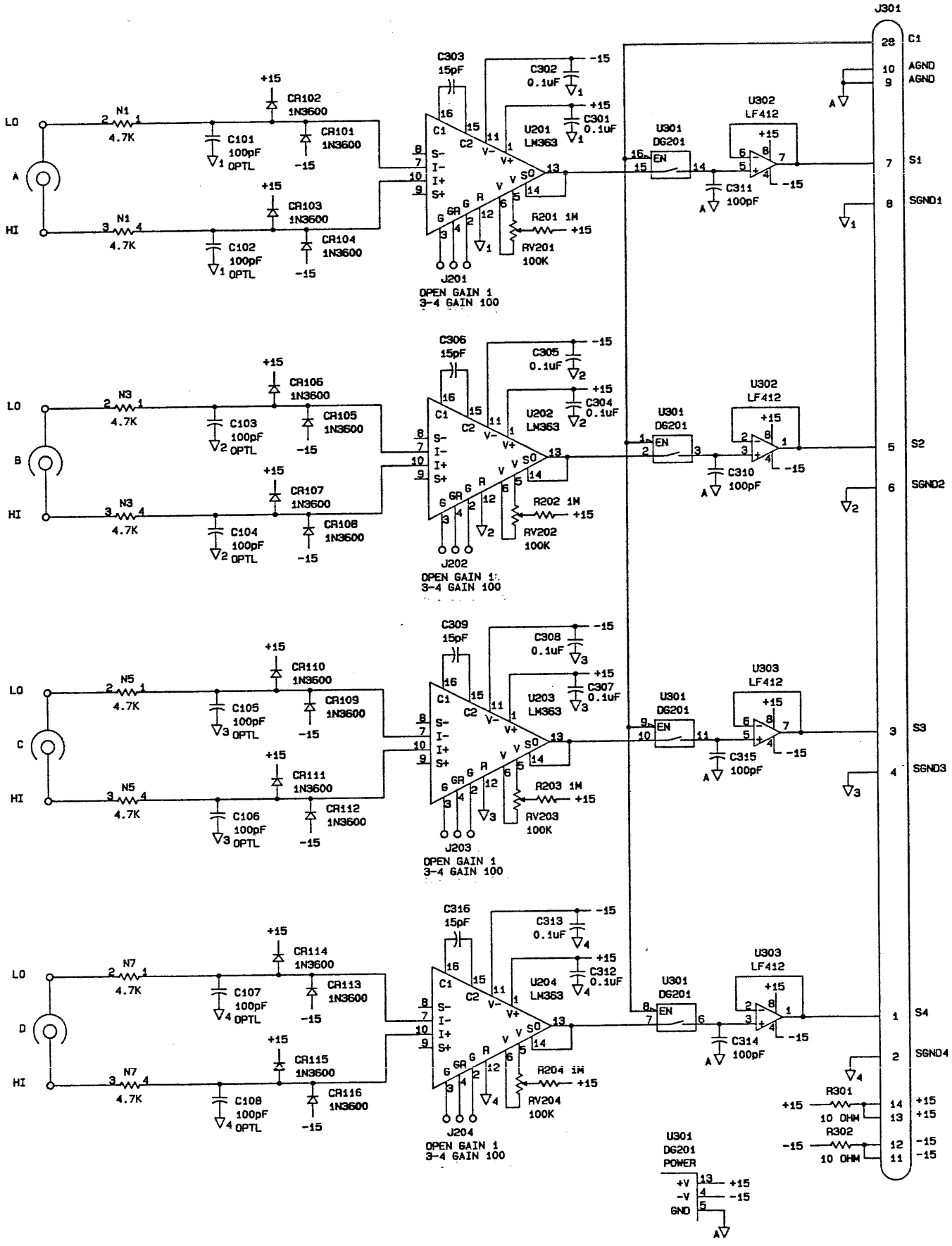
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFFERENTIAL ANALOG INPUT BOARD	A2	1 OF 1



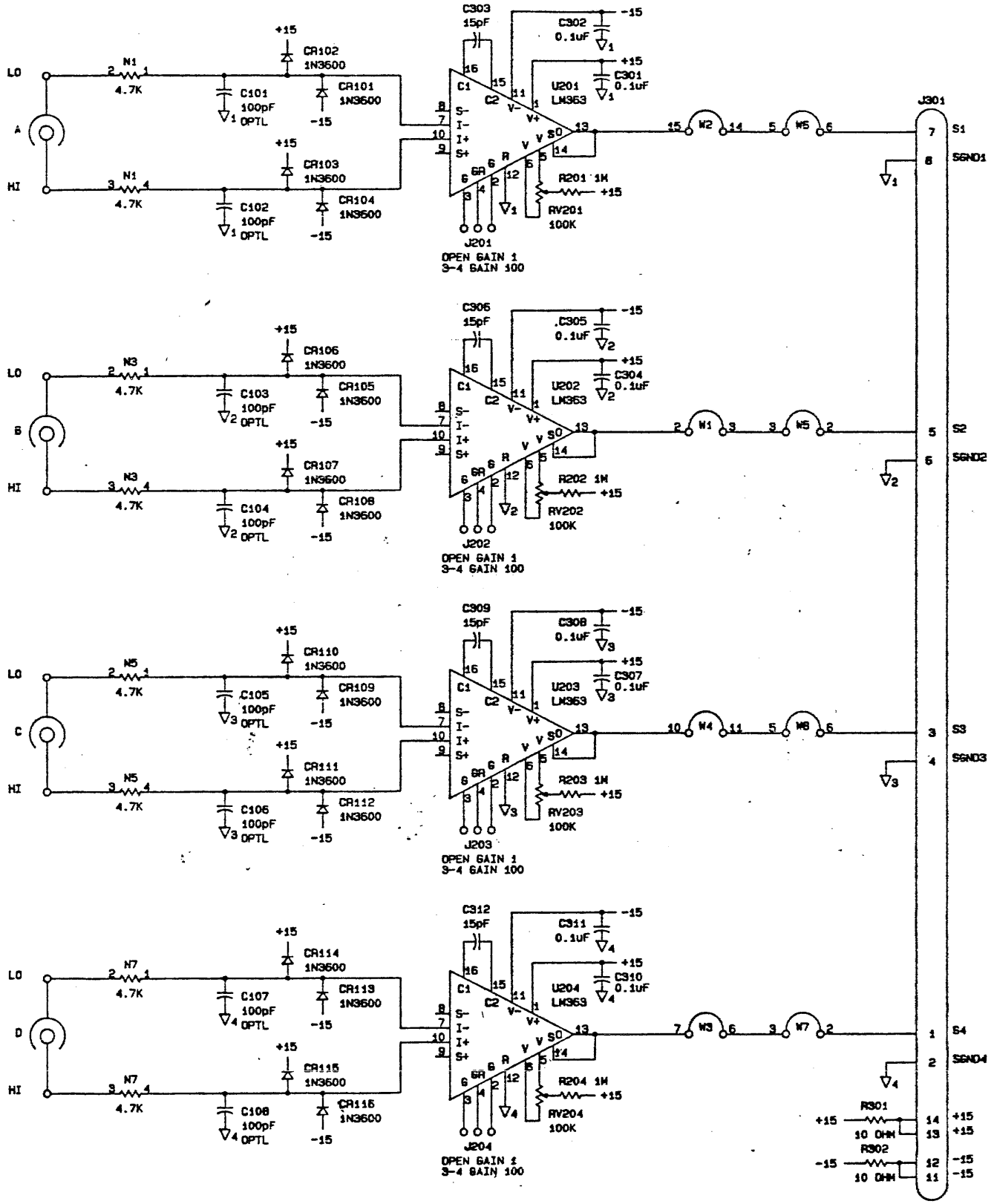
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFF ANALOG INPUT BOARD W/HOLD	A3	1 OF 1



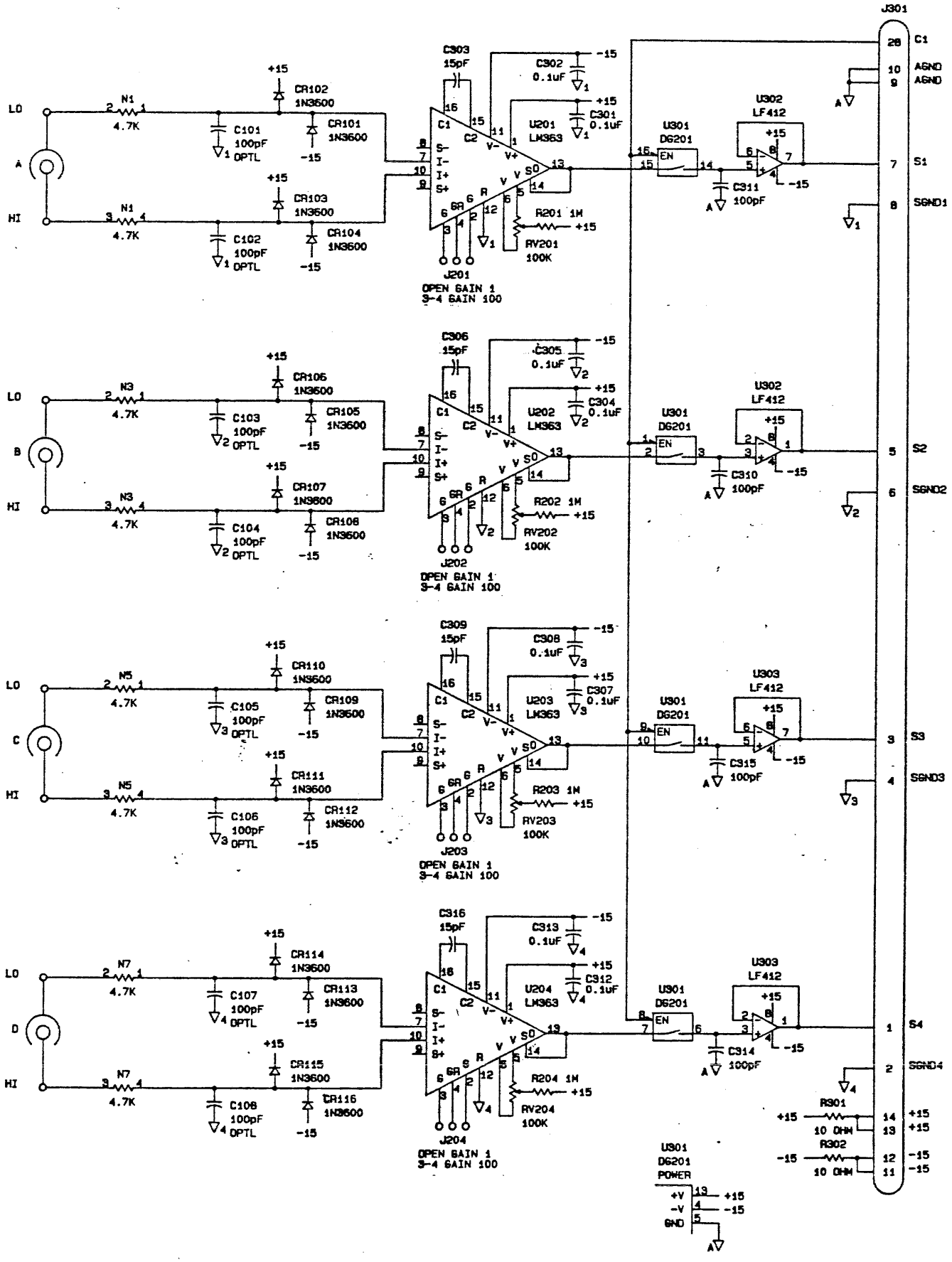
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFFERENTIAL ANALOG INPUT BOARD	A4	1 OF 1



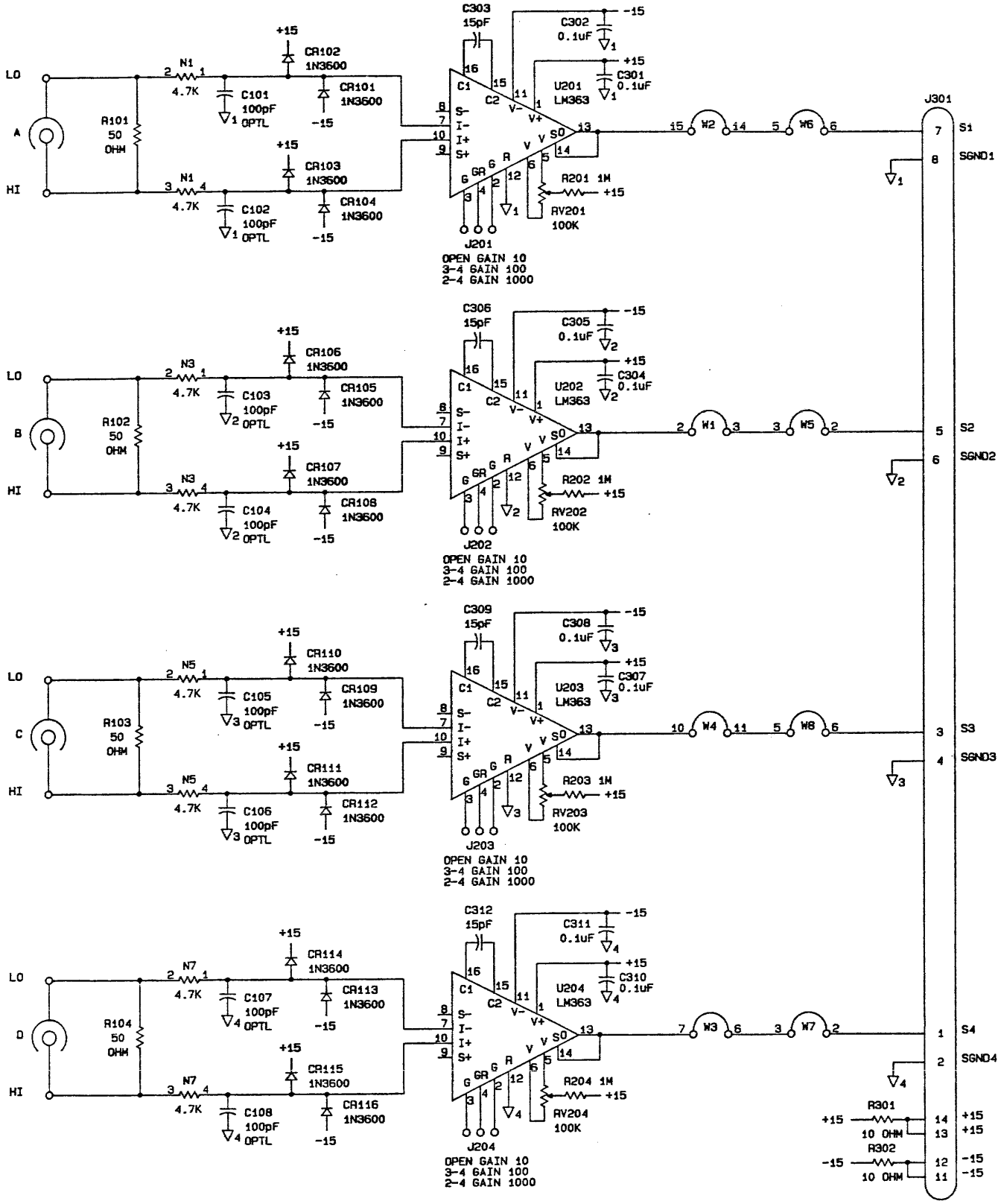
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFF ANALOG INPUT BOARD W/HOLD	A5	1 OF 1



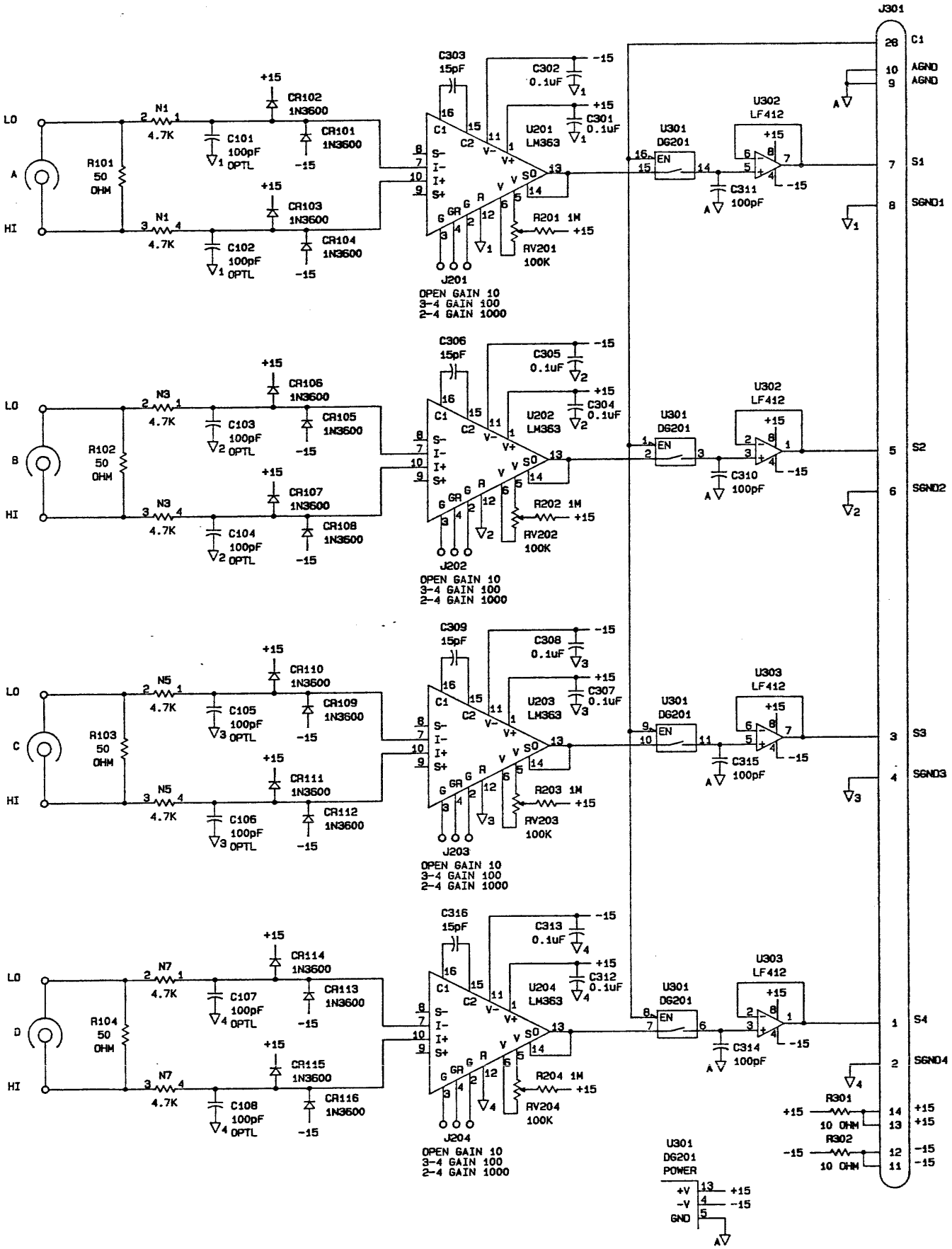
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFFERENTIAL ANALOG INPUT BOARD	A6	1 OF 1



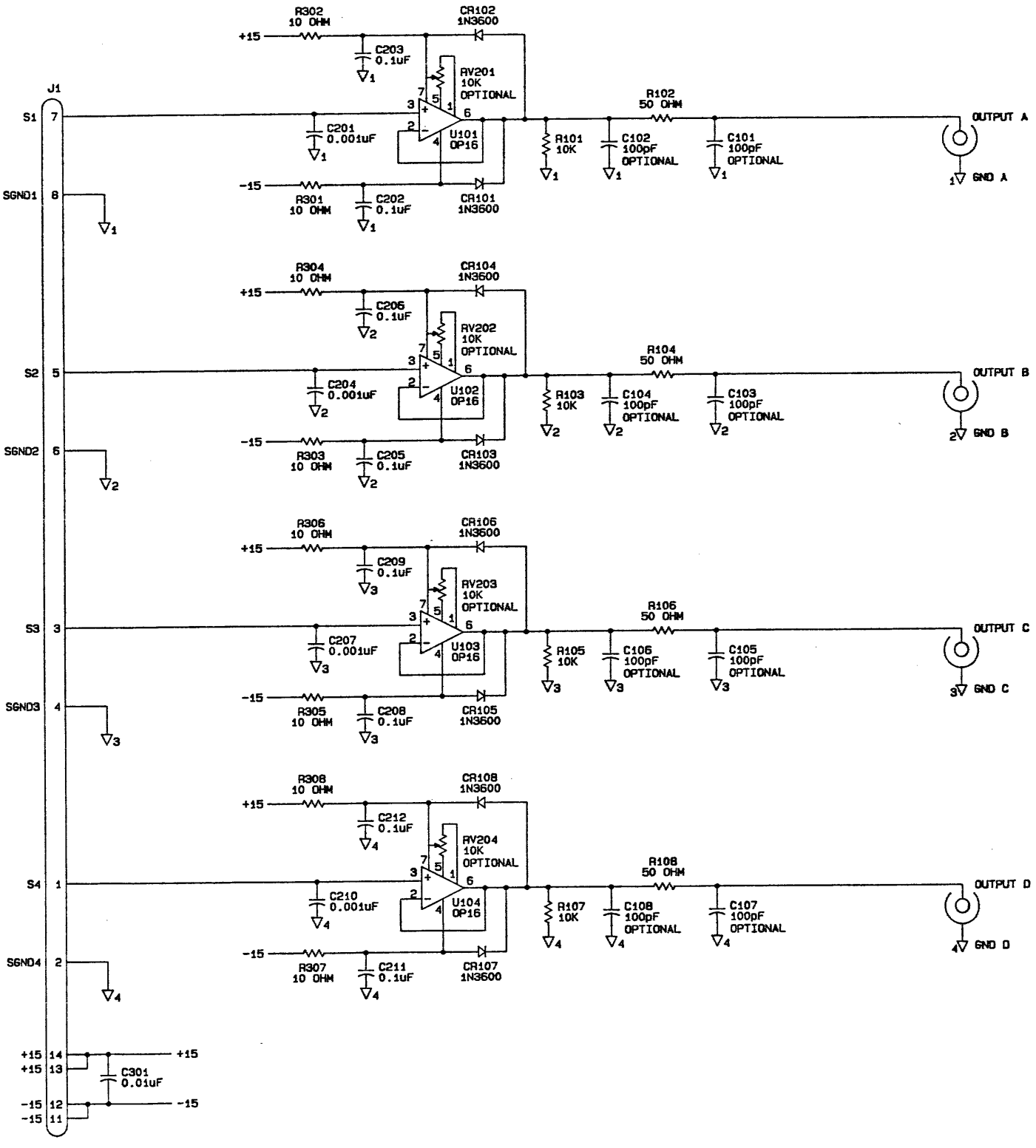
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFF ANALOG INPUT BOARD W/HOLD	A7	1 OF 1



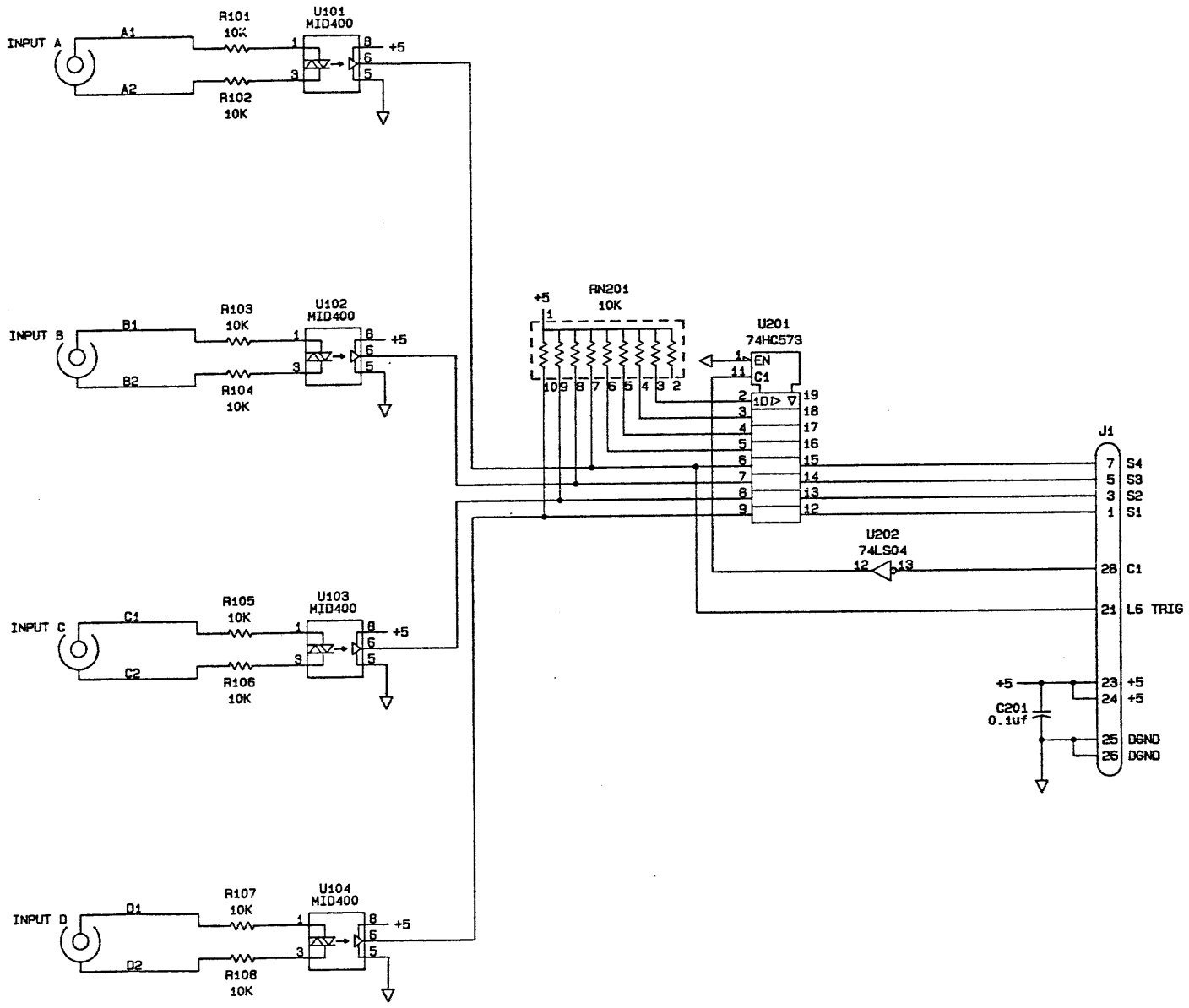
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	4-20 mA CURRENT INPUT BOARD	C1	1 OF 1



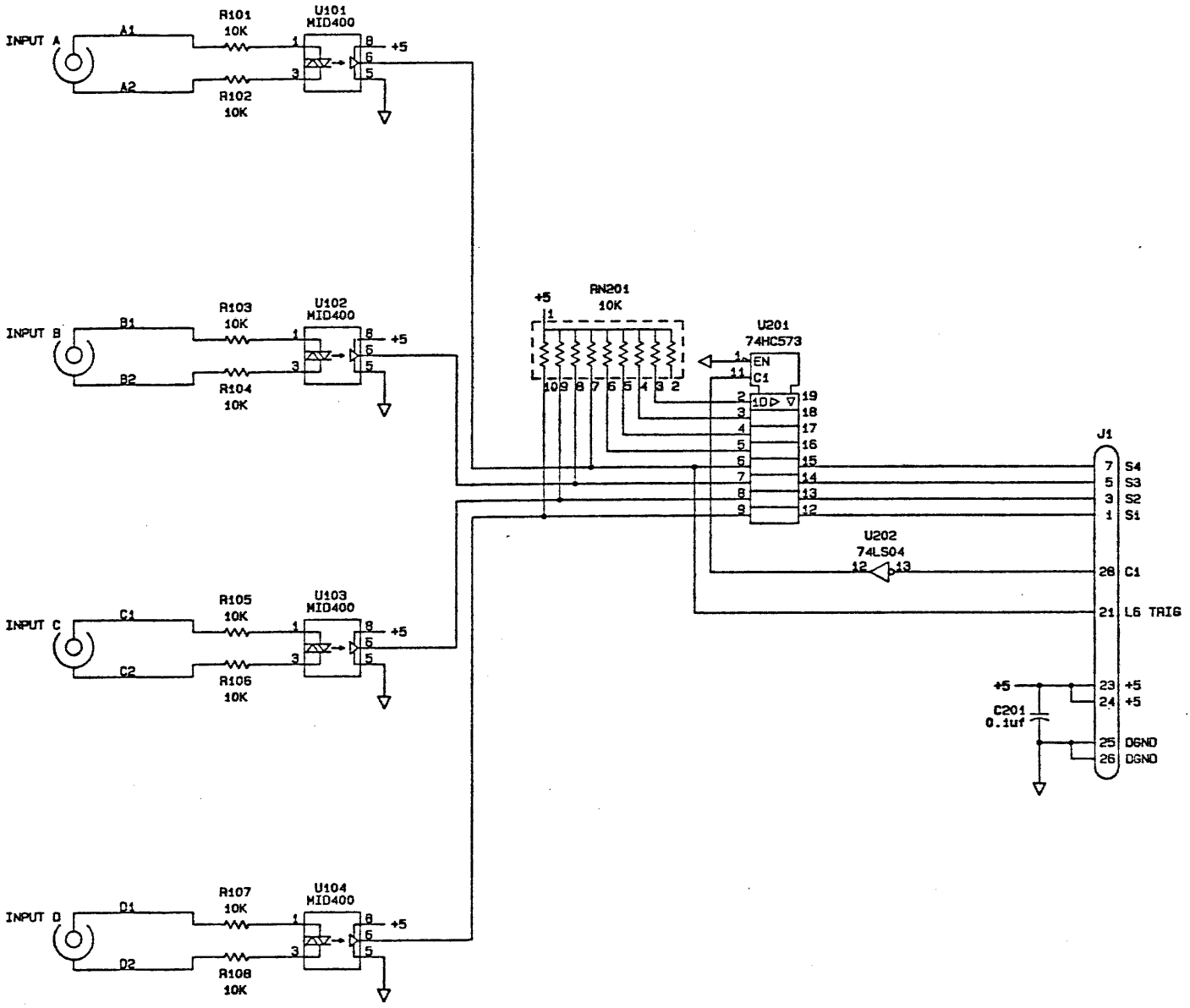
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	4-20 mA CURRENT INP BD W/HOLD	C2	1 OF 1



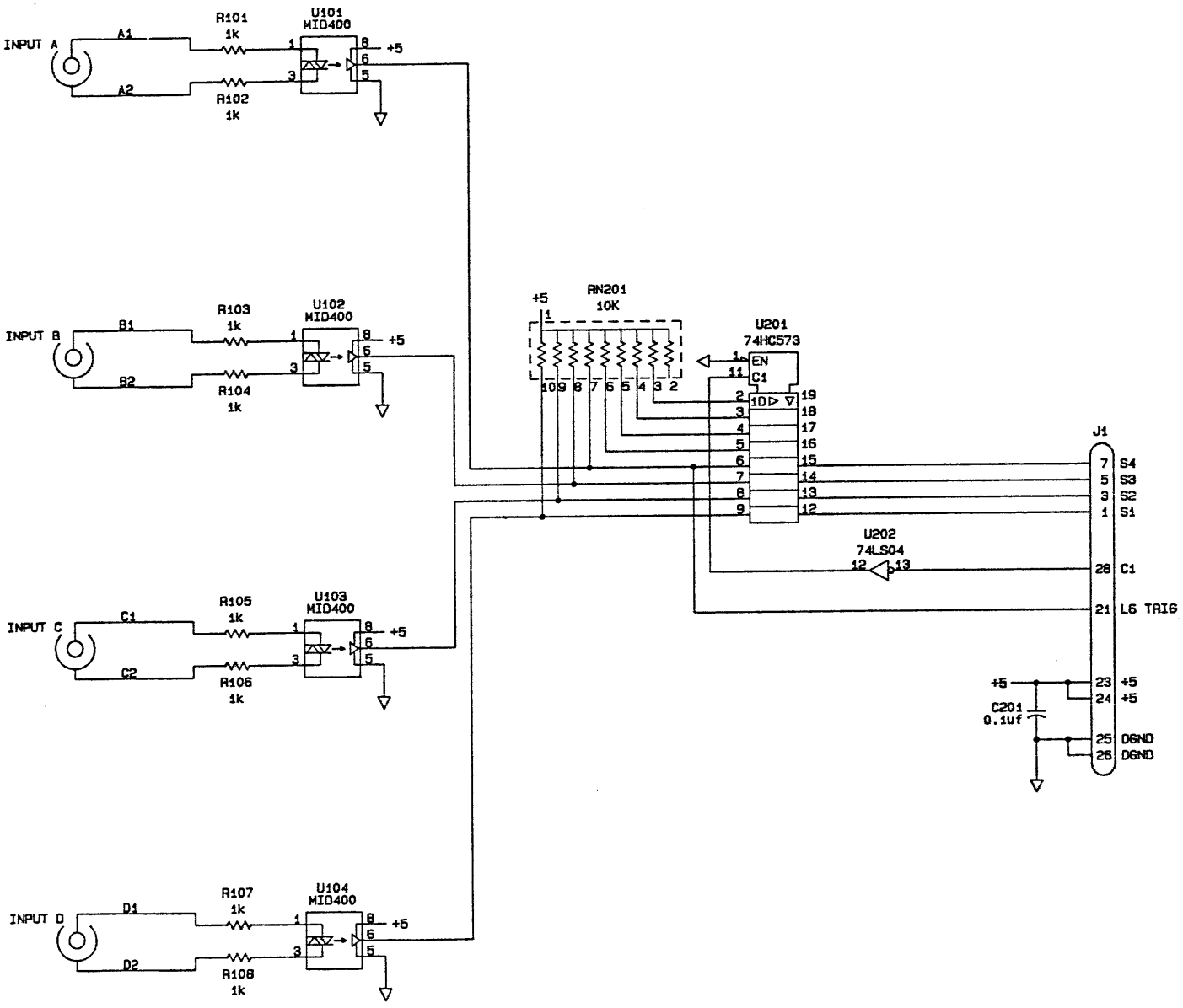
PRODUCT	NAME 7000-670-22-A	VERSION	PAGE
7000-MDAS	±10 VOLT ANALOG OUTPUT BOARD	S1	1 OF 1



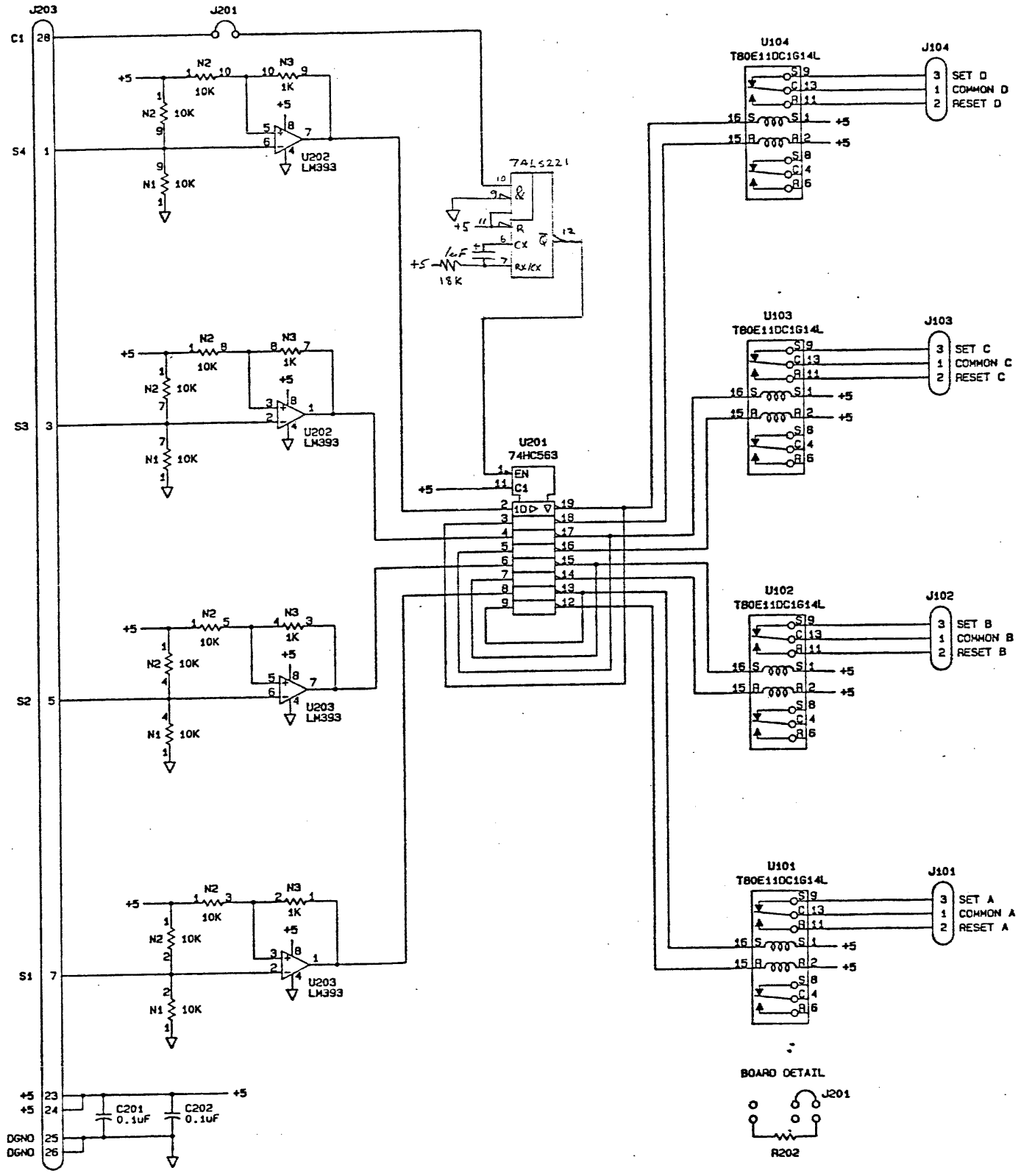
PRODUCT	NAME 7000-670-23-A	VERSION	PAGE
7000 MDAS	110/220 AC DETECT D-INPUT BD	R1	1 OF 1



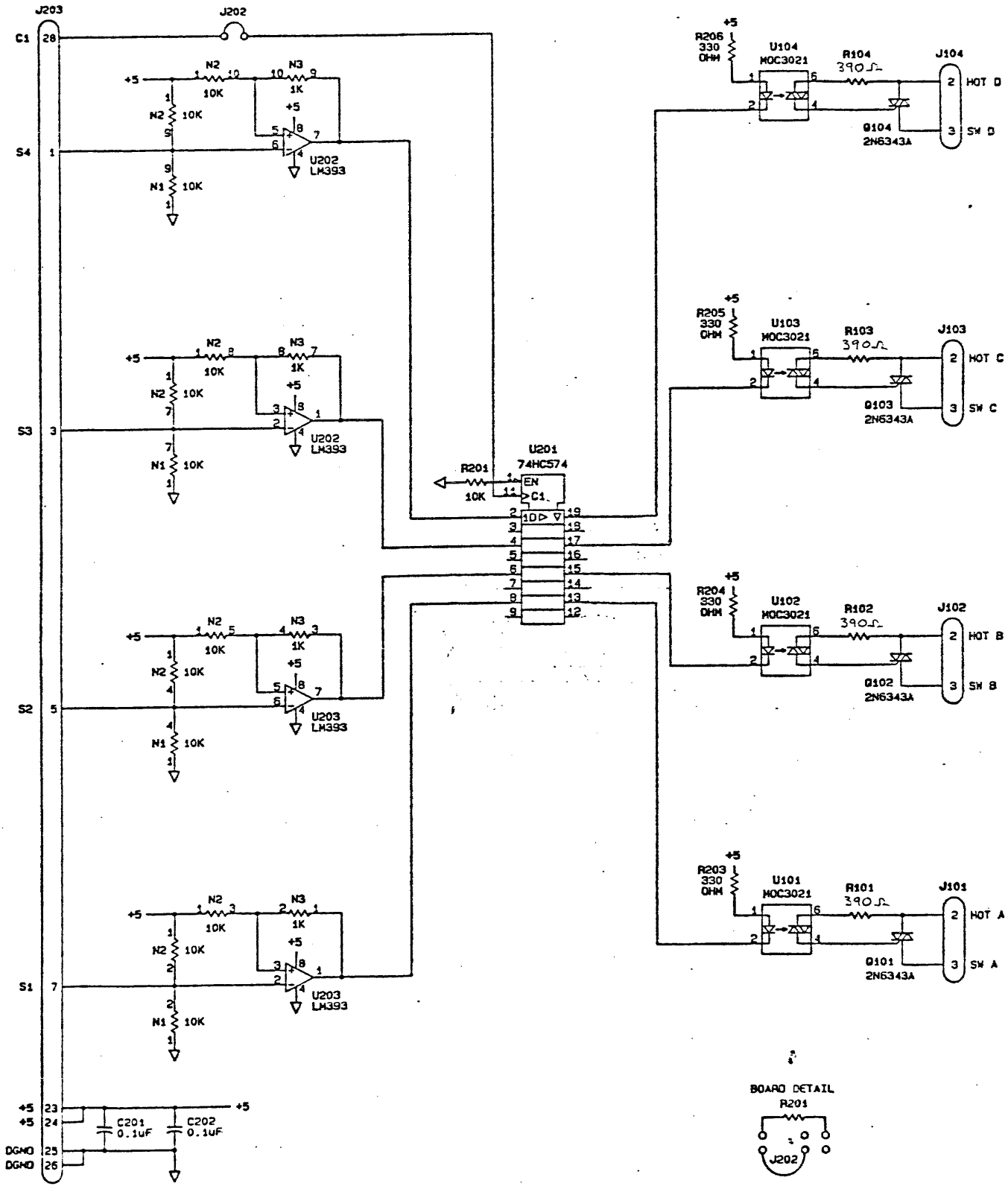
PRODUCT	NAME 7000-670-23-A	VERSION	PAGE
7000 MDAS	HI TTL-100V DETECT D-INPUT BD	R2	1 OF 1



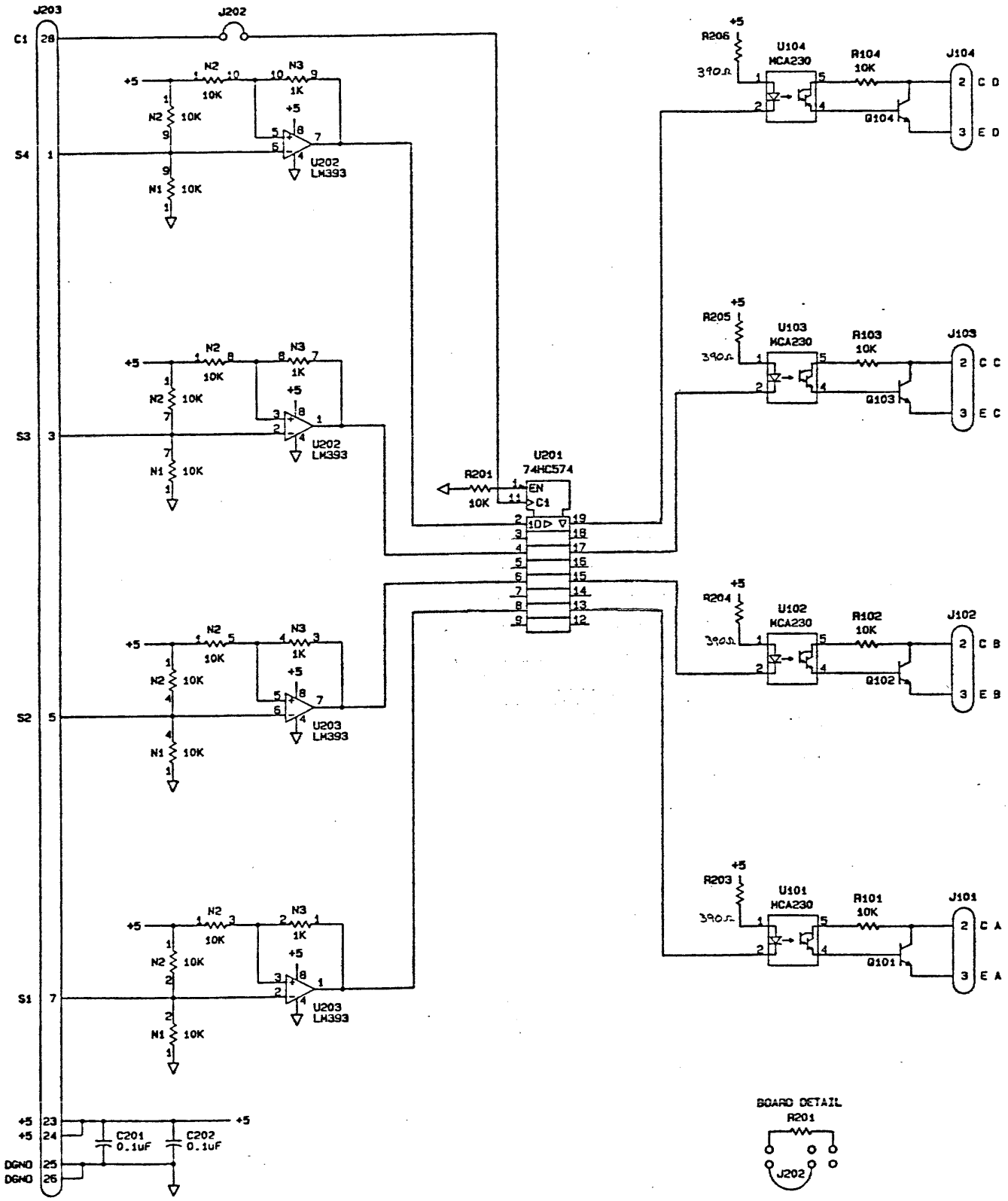
PRODUCT	NAME 7000-670-23-A	VERSION	PAGE
7000 MDAS	LOW TTL DETECTION D-INPUT BD	R3	1 OF 1



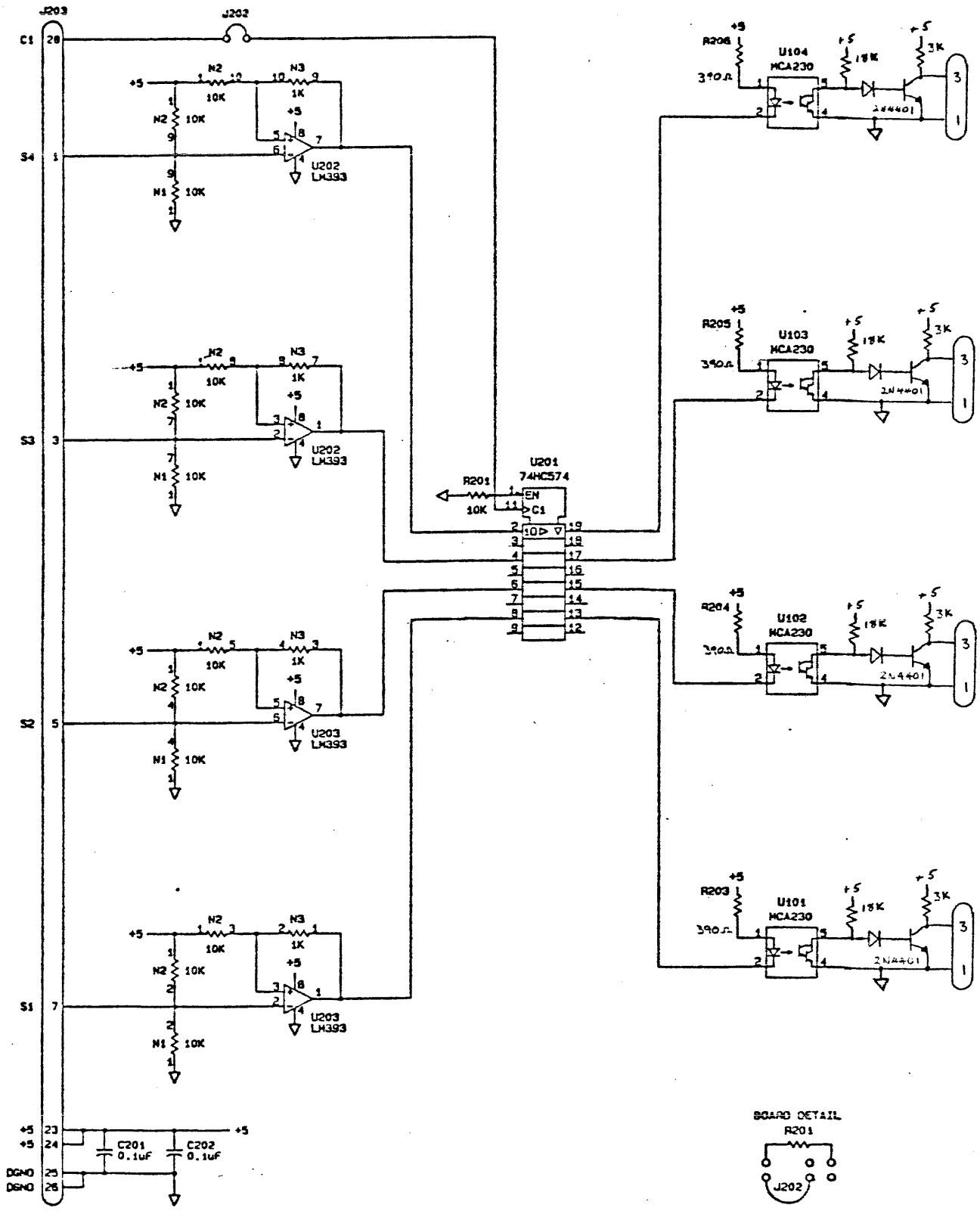
PRODUCT	NAME	VERSION	PAGE
7000 MDAS	7000-670-24-A LATCHING RELAY D-OUTPUT BOARD	D1	1 OF 1



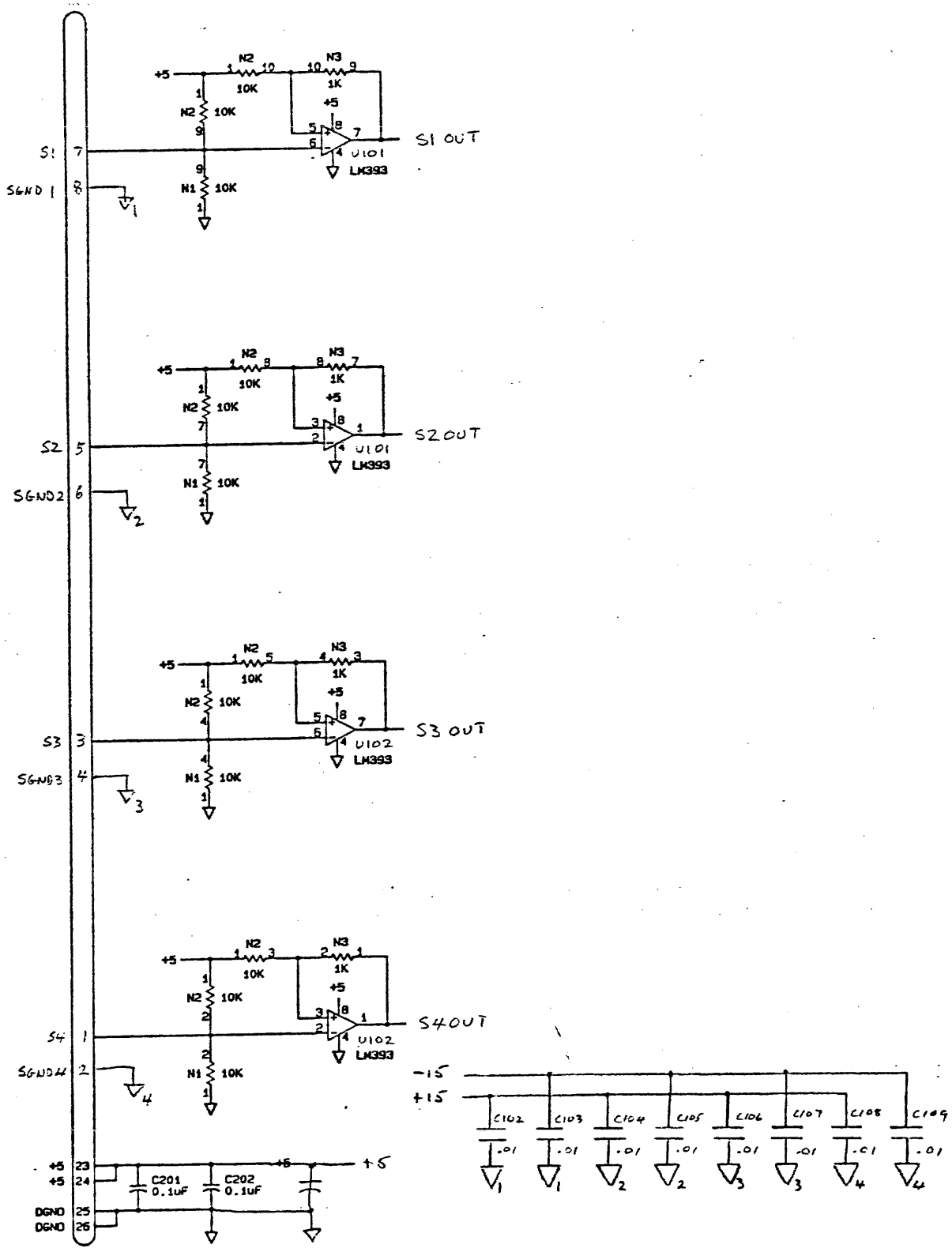
PRODUCT	NAME 7000-670-24-A	VERSION	PAGE
7000 MDAS	SOLID STATE AC RELAY D-OUT BD	D2	1 OF 1



PRODUCT	NAME 7000-670-24-A	VERSION	PAGE
7000 MDAS	SOLID STATE DC RELAY D-OUT BD	D3	1 OF 1



PRODUCT	NAME	VERSION	PAGE
7000 MDAS	7000-670-24-A TTL DRIVER	D4	1 OF 1



PRODUCT	NAME	VERSION	PAGE
7000 MDAS	7000-670-24-A USER CONFIGURABLE BOARD	GU1	1 OF 1