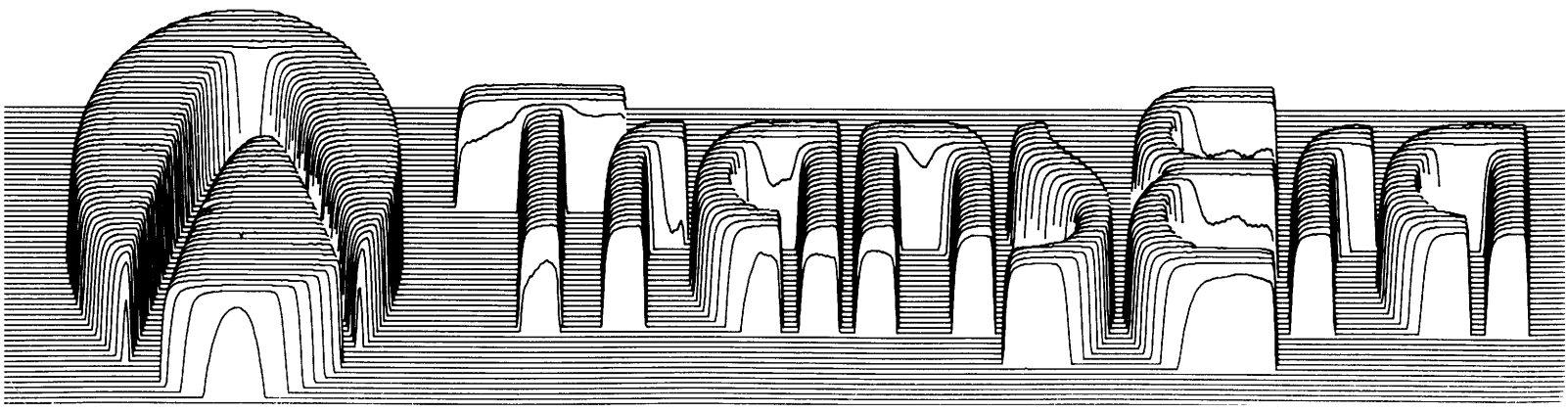
TransEra

7000-MDAS

Reference Manual



TransEra

7000-MDAS

Reference Manual

Manual Part No. 7000-070-02

TransEra Corporation
3707 North Canyon Road Bldg. 4
Provo, Utah 84604

Phone 801-224-6550

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 512 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 385 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 Aquainted 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 D 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 conditoining 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 opional 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 outide 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 asyncronous, 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 No 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 commmand) 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)
*ERRORS DEV	Errors Device	(ERR)
*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 four menu entries are used to assign the four information types, Command, Output, Errors and Input, to the devices that suit the needs of the system. These four 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, *ERRORS 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 indicatied 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
LED (*ERRORS DEV only)

Table 2.1
RS-232 Parameter Options

Baud Rates		Parity	Echo	Xon/Xoff	Delimeter
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 coeffients 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 number 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 nine 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

If the Error Device is set to the LED display, the error message is displayed instead of the error number. 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 catagories: 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 catagory 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 catagory 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 transfered but the commands are all executed within MDAS. Square root, scaling and addition are some of the commands in the Math catagory.

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 paramters 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 deivce 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 posision 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 multiplied by the gain whereas in the integer format, the data is not gain adjusted.

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, there are three minimum period values depending on the chosen parameters. If the trigger type is 1,2 or 3 and pre-trigger samples are requested, then the minimum period is 6.4E-6 seconds. If the trigger type is 1, 2 or 3 and no pre-trigger samples are requested, then the minimum period is 3.2E-6 seconds. Finally, if the trigger type is 0 and no pre-trigger samples are requested, the minimum period is 2.6E-6 seconds.

For sequences with more than one channel, the minimum period is derived from the following equation: minimum period = (channels * 7.2E-6)+6.8E-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 7.2E-6 seconds. The channels in a sequence are sampled in the order that they appear in the command.

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 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:"a0 6.1100 b2 1 2c 3;"
```

The first 1100 numbers in buffers 1 and 3 are output to channels b2 and 2c. This is repeated 6 times.

Count Input

Counts the number of TTL pulses during an interval

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

where: <reps> number of succesive 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.

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 occuring 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.

Copy buffer

Copies contents of one buffer to another

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

where: <bufA> buffer containing input data
<bufX> buffer to copy data into
<pnts> number of points to copy

Description

COPY copies the number of points (<pnts>) from <bufA> to <bufX>. 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>.

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.

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 miminum 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. <Reps> must be sent in integer format (no decimal point or scientific notation).

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 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 miminum 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.

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 Ouput 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.

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 Digital

Reads digital data and sends it out the Output Port

Syntax: GD <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 DIGITAL sends the digital level of each channel specified in the command to the Output Port (see Ouput Device command). The data is sent in ASCII format; a '0' is sent for a low level TTL signal and a '1' is sent for 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:

```
640 PRINT #1:"gd a5 b5 c5 d5;"
650 INPUT #1:W,X,Y,Z
```

The digital information on channels a5, b5, c5, and d5 is read and output in ASCII format as either 0 or 1 for each channel. The variables W, X, Y and Z are set to 0 or 1 according to the digital information present at the respective input channels.

Get Error

Sends the last error message to the defined data port

Syntax: GE;

format: <errnum> <dlm>

where: <err> error number
<dlm> delimiter, either <CR> or <CR> <LF>

Description

The GET ERROR command sends the last error number to the port that is defined as the data output port. (See the Define Output command). The error number is then cleared. If the LED display was set as the error device, the error message that accompanies the error number will also be cleared by the GET ERROR command.

Examples:

900 PRINT #1:"ge;"

The most recent error number is sent to the output data port and the error number is set to 0.

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.

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.

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.)

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 specifed 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 specifed 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.

Read

Load a buffer with ACSII 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 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.

Write

Output a buffer in ASCII format

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

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

where: <bufA> name of buffer to output
<size> the number of points to be sent
<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.

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.

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 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.

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

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.

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>;

where: <port> the port that MDAS will send error messages to

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	PORT 1
2	PORT 2
3	GPIB

Examples:

```
670 PRINT #1:"err 3;"
```

The GPIB is 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.

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.

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.

Output Device

Sets the port for the device that data is written to

Syntax: OUT <port>;

where: <port> the port that MDAS will send data to

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

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 expessed 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 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 ommitted:

dd mmm yy
hh:mm:ss
yy
ss

If one of the above fields is ommitted, 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> [<level>]];

where: <type> a digit between 0 and 3 indicating the type of trigger
<chan> the channel number to trigger on
<level> the level to cause triggering of conversion

Description

SET TRIGGER sets the type of trigger used to start A/D or D/A conversions. <type> is interpreted according to the following table:

<type>	explanation
0	No trigger, commands executed immediately
1	trigger when the sample values on <channel> exceed <level> for at least one period
2	trigger when the sample values on <channel> fall below <level> for at least one period
3	TTL, falling edge triggered

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 1 0;"
```

The trigger is true when the value on channel 1 is less than 0 Volts.

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

The trigger is true when a falling TTL singal is deteced on the digital trigger input.

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.

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 decible 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 trasform.

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 trasform.

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 symmetrial 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 minumum 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 commmad.)

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, MULTIPY 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 ouputs 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 progammable 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 B1. 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 B1 1;"         ! data output on channel B1 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 obtianed. 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 DIMENSION 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 ouput 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 DIMENSION 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 ouput 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 number 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 nine 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

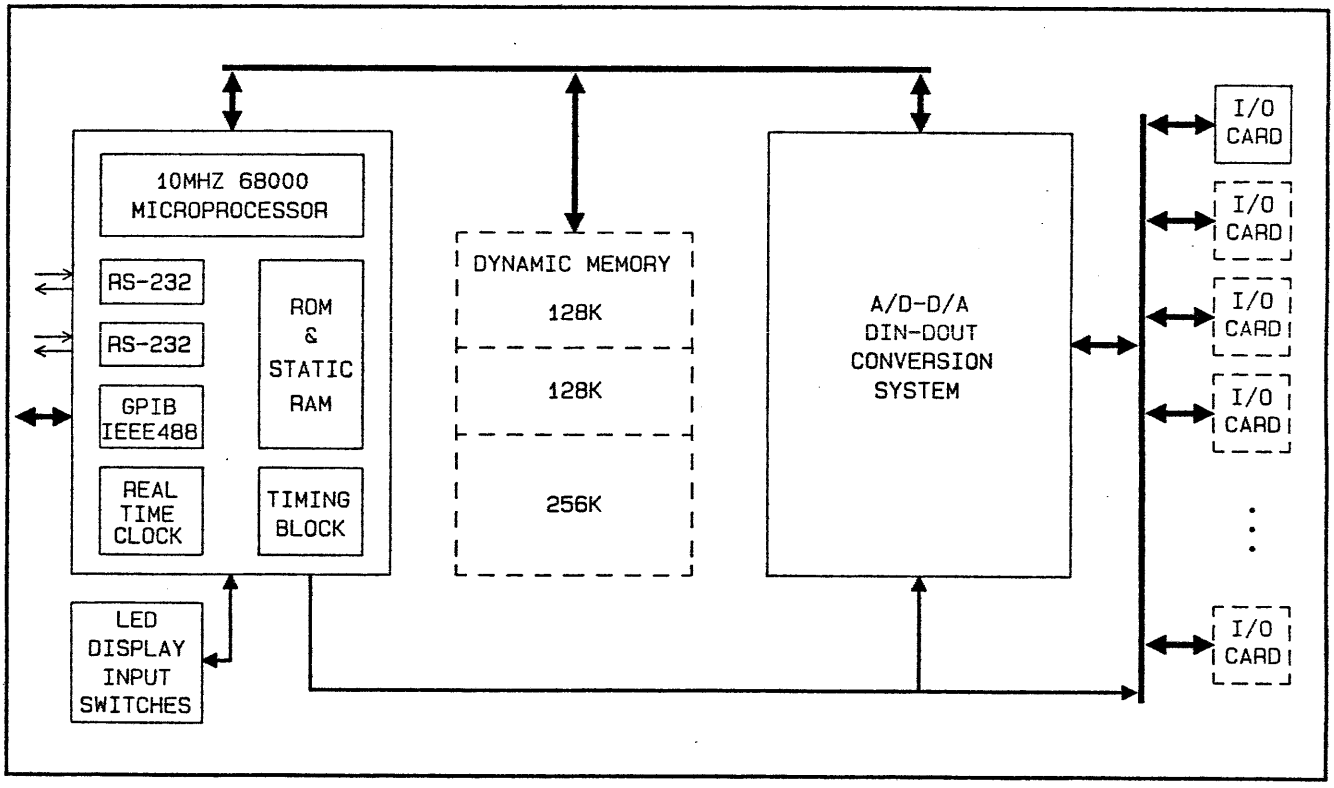
If the Error Device is set to the LED display, the error message is displayed instead of the error number. The format for error messages is:

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.

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

Muiti-channel analysis

Voice synthesis/recognition

Optical input/scanning

Pattern recognition

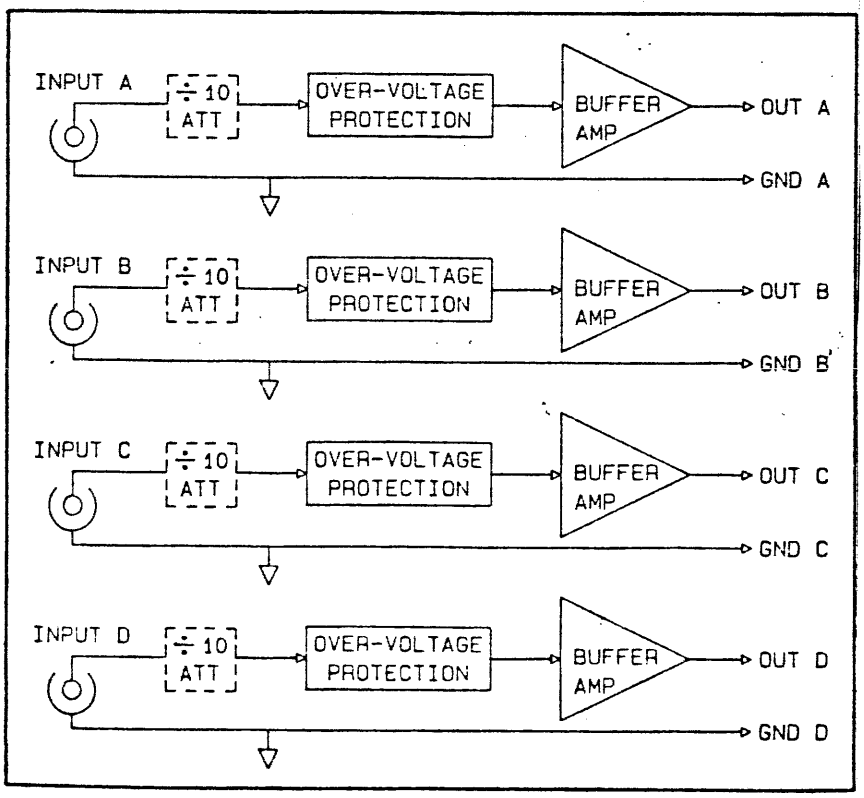
Acoustic and vibration testing

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:

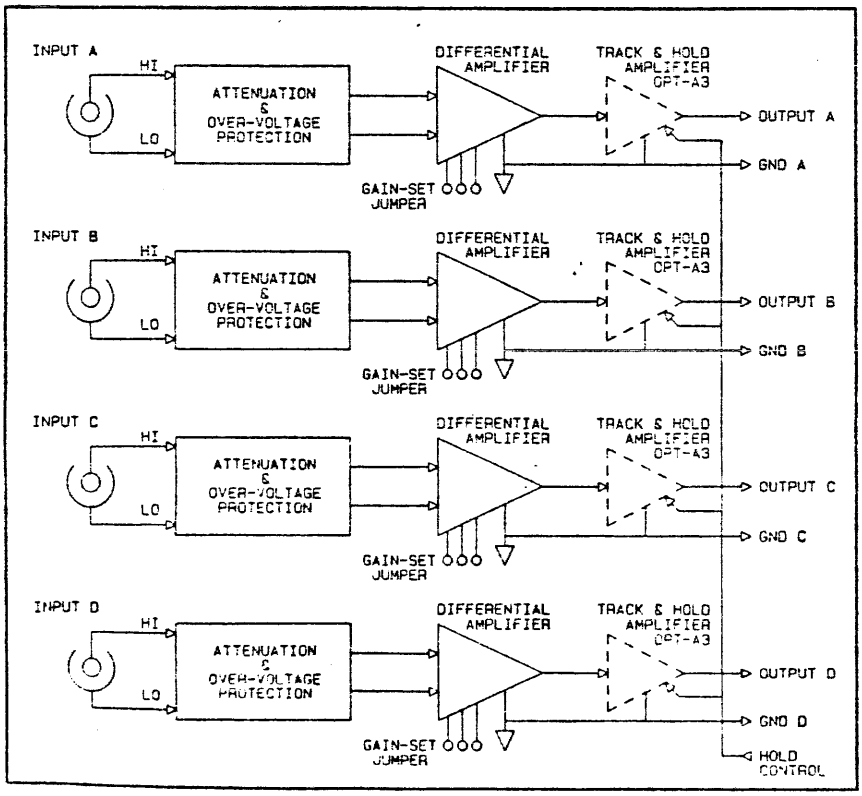
- General purpose differential input
- Transient recording
- Process control
- Transducer interface
- Multi-channel analysis
- Voice synthesis/recognition
- Acoustic and vibration testing
- Seismic data acquistion and analysis
- Signal processing

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:

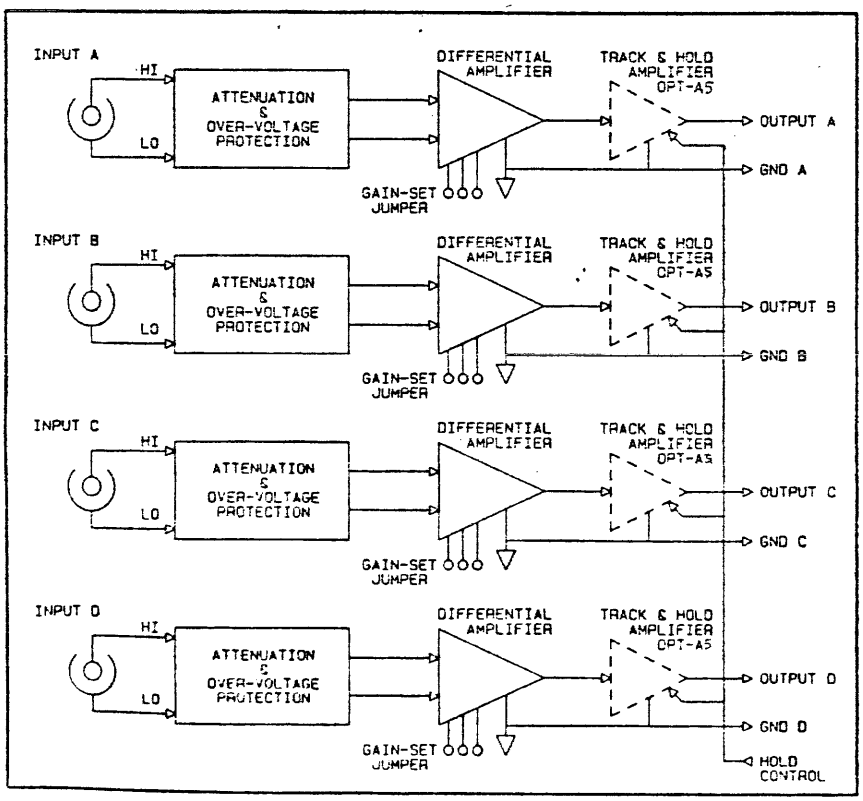
- General purpose differential input
- Transient recording
- Process control
- Transducer interface
- Multi-channel analysis
- Voice synthesis/recognition
- Acoustic and vibration testing
- Seismic data acquistion and analysis
- Signal processing

SPECIFICATIONS:

Input channels	4
Type	Differential
Manually selectable ranges (volts)	±10
Input impedance (Mohms)	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:

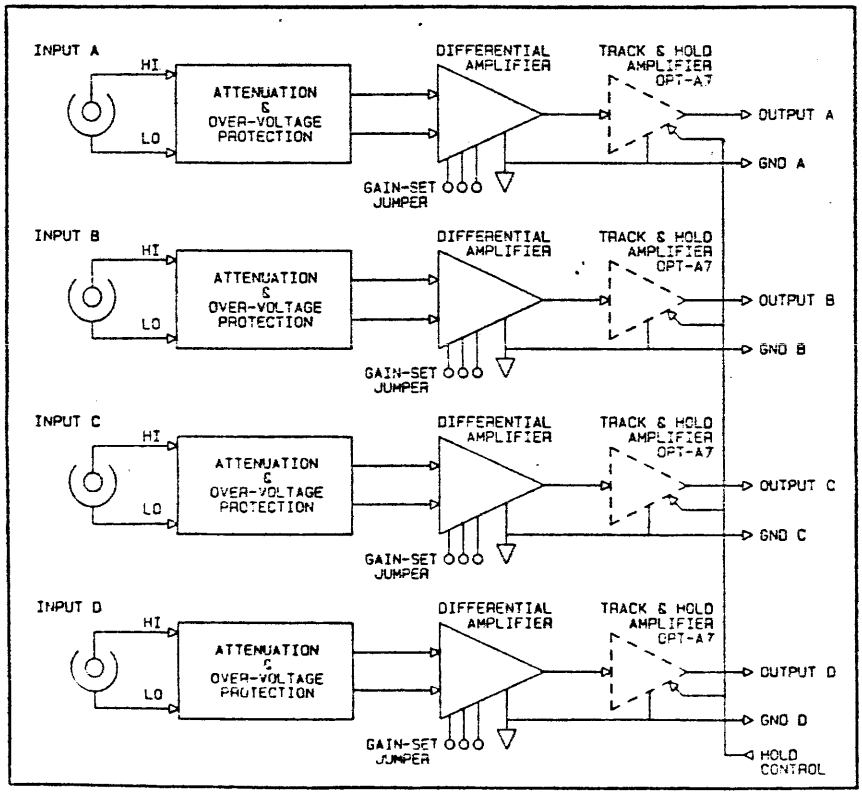
- Low level differential input
- Low level transducer input
- Process control
- Multi-channel analysis
- Voice synthesis/recognition
- Acoustic and vibration testing
- Seismic data acquistion and analysis
- Signal processing

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

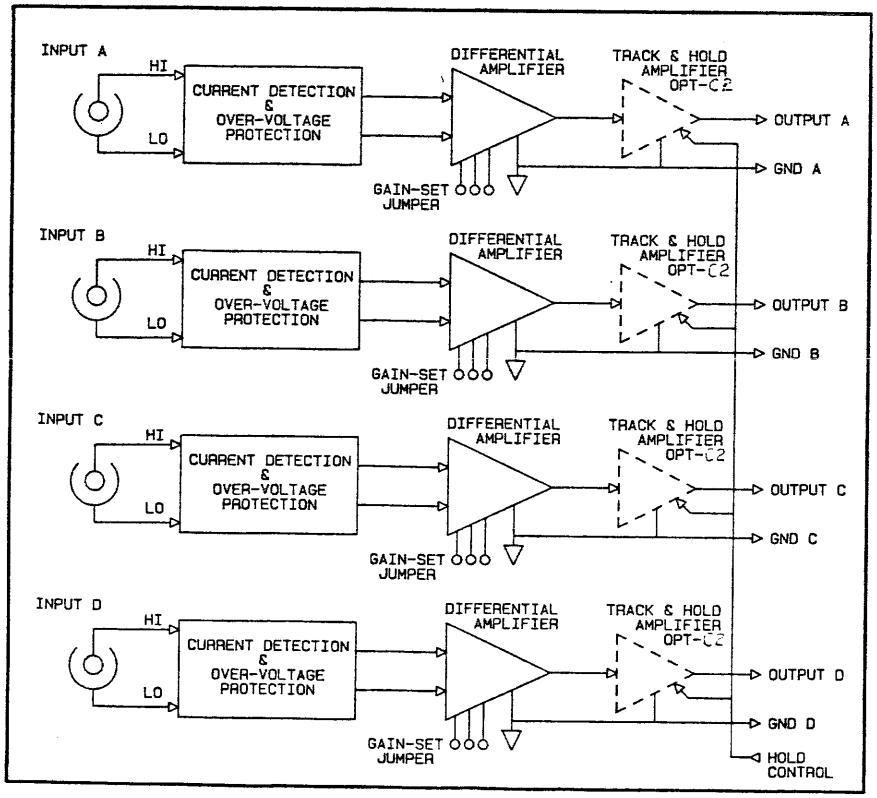
Acoustic and vibration testing

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

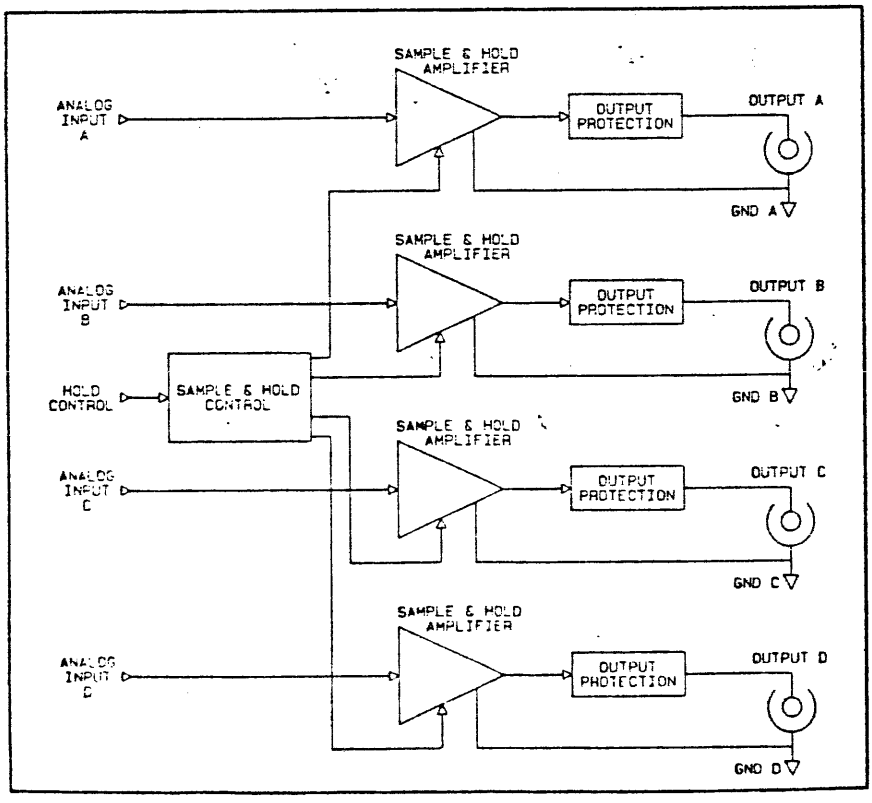
Industrial process control

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

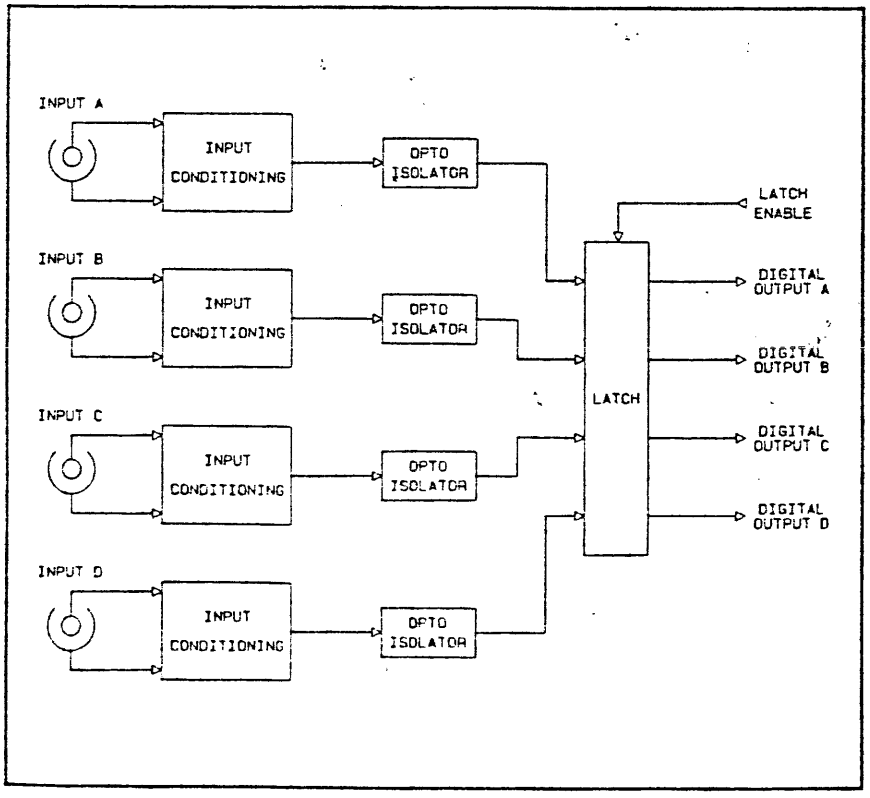
Process control

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

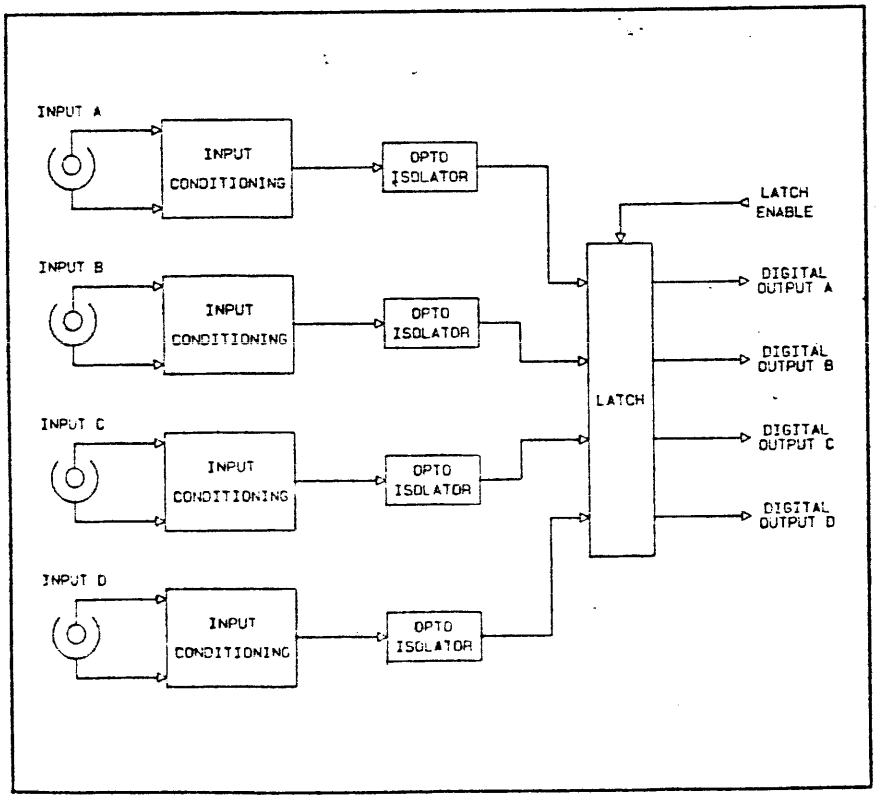
Trigger input for data acquisition

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:

4 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

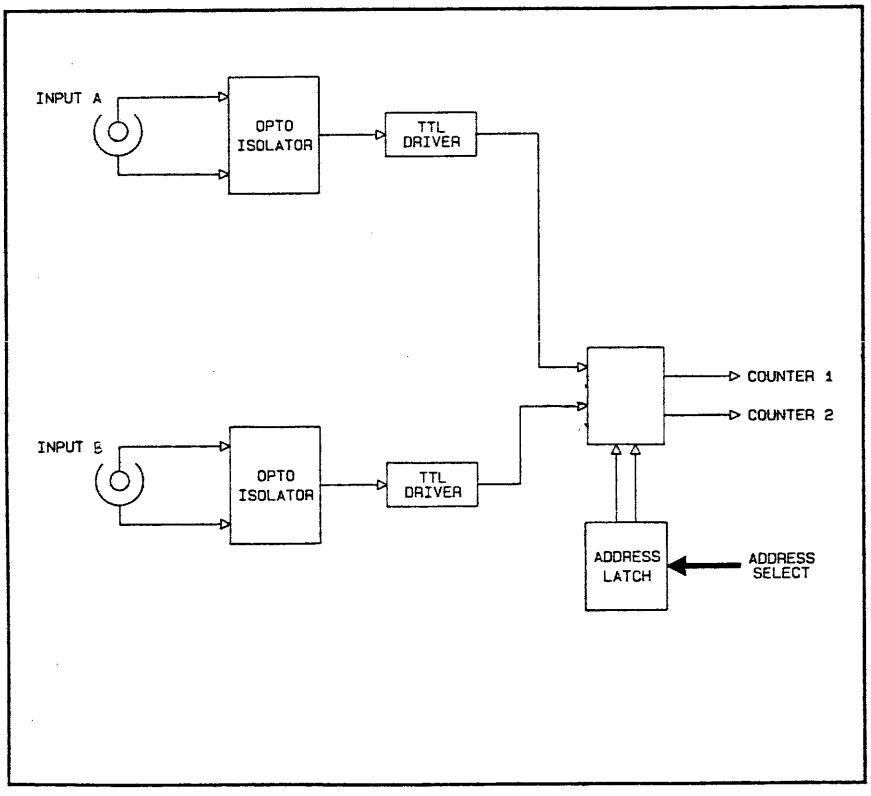
Differential frequency measurement

Fiber optic data transfer

Process control

SPECIFICATIONS:

Input channels	4
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

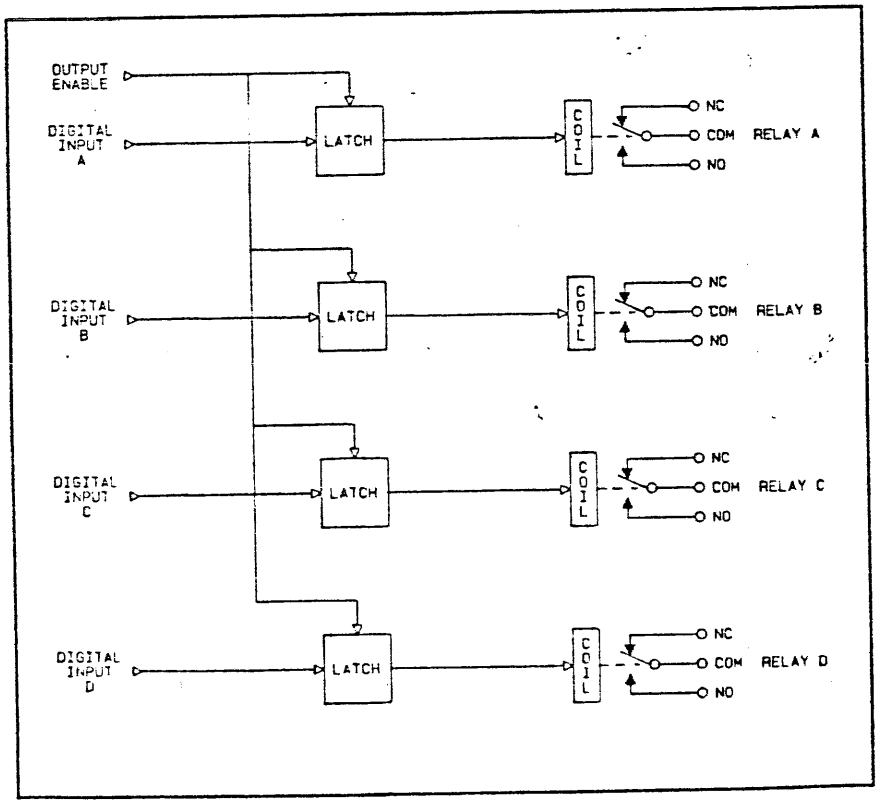
Signal routing

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

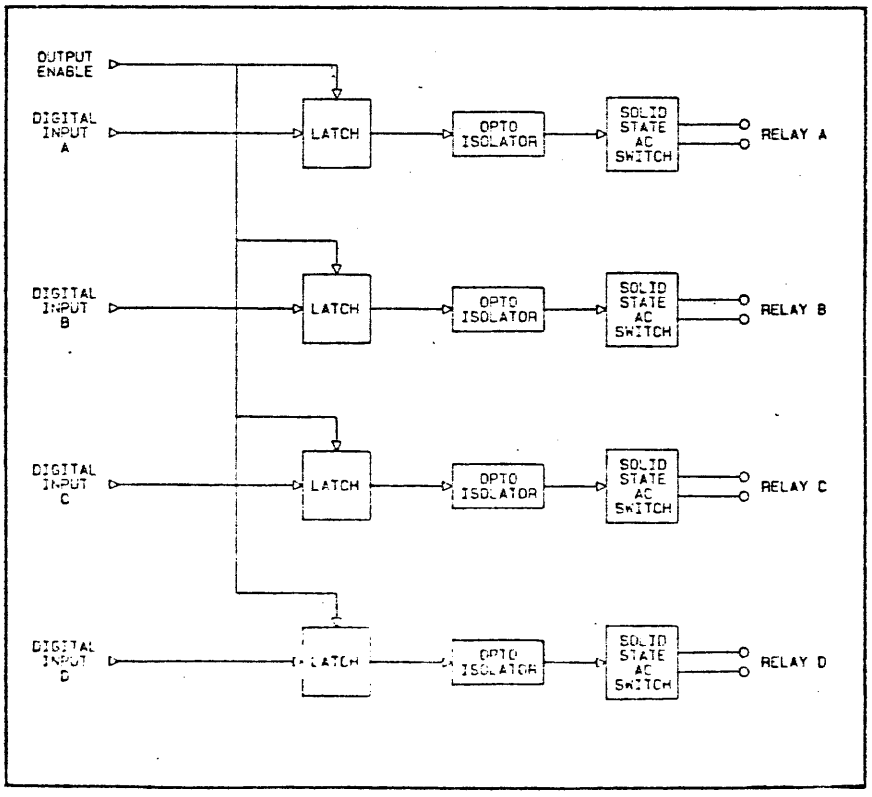
Motor control

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

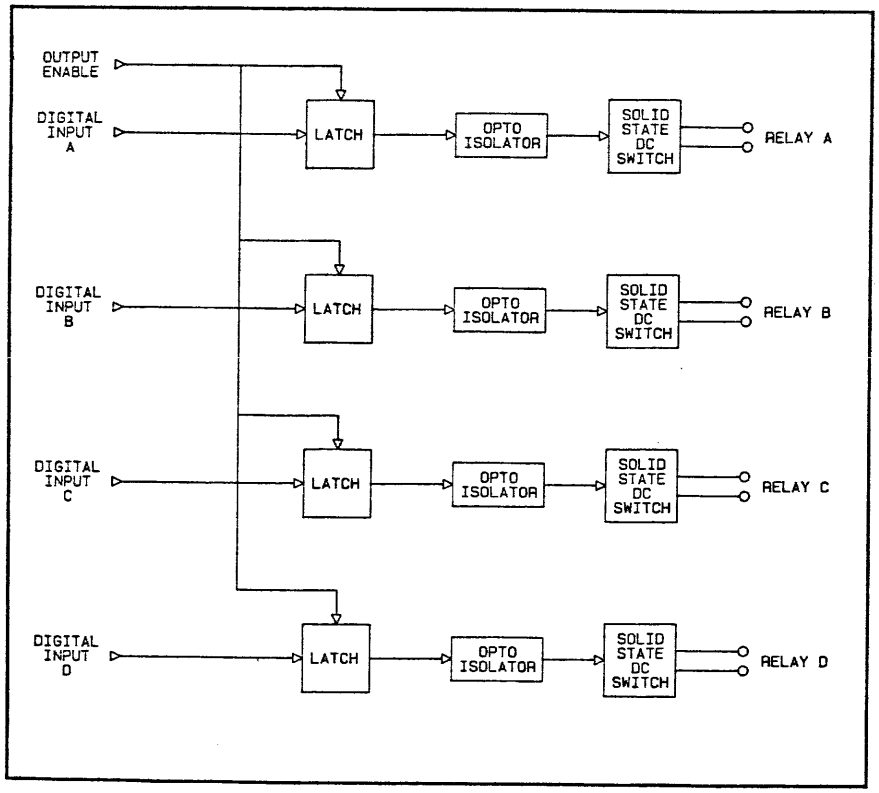
Motor control

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

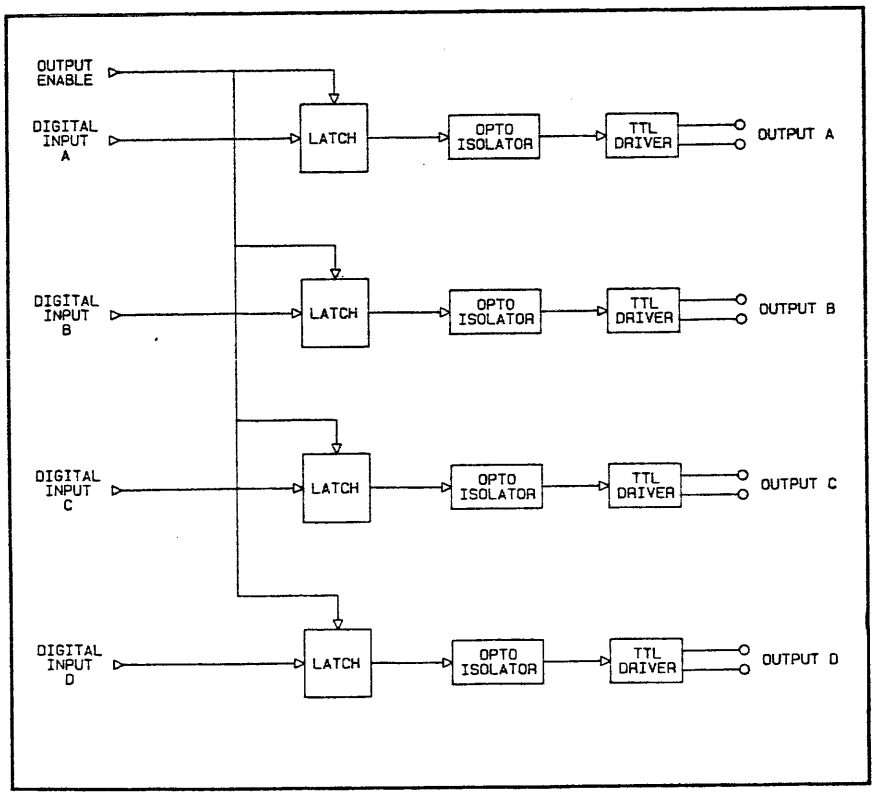
Motor control

Control actuation

Alarm activation

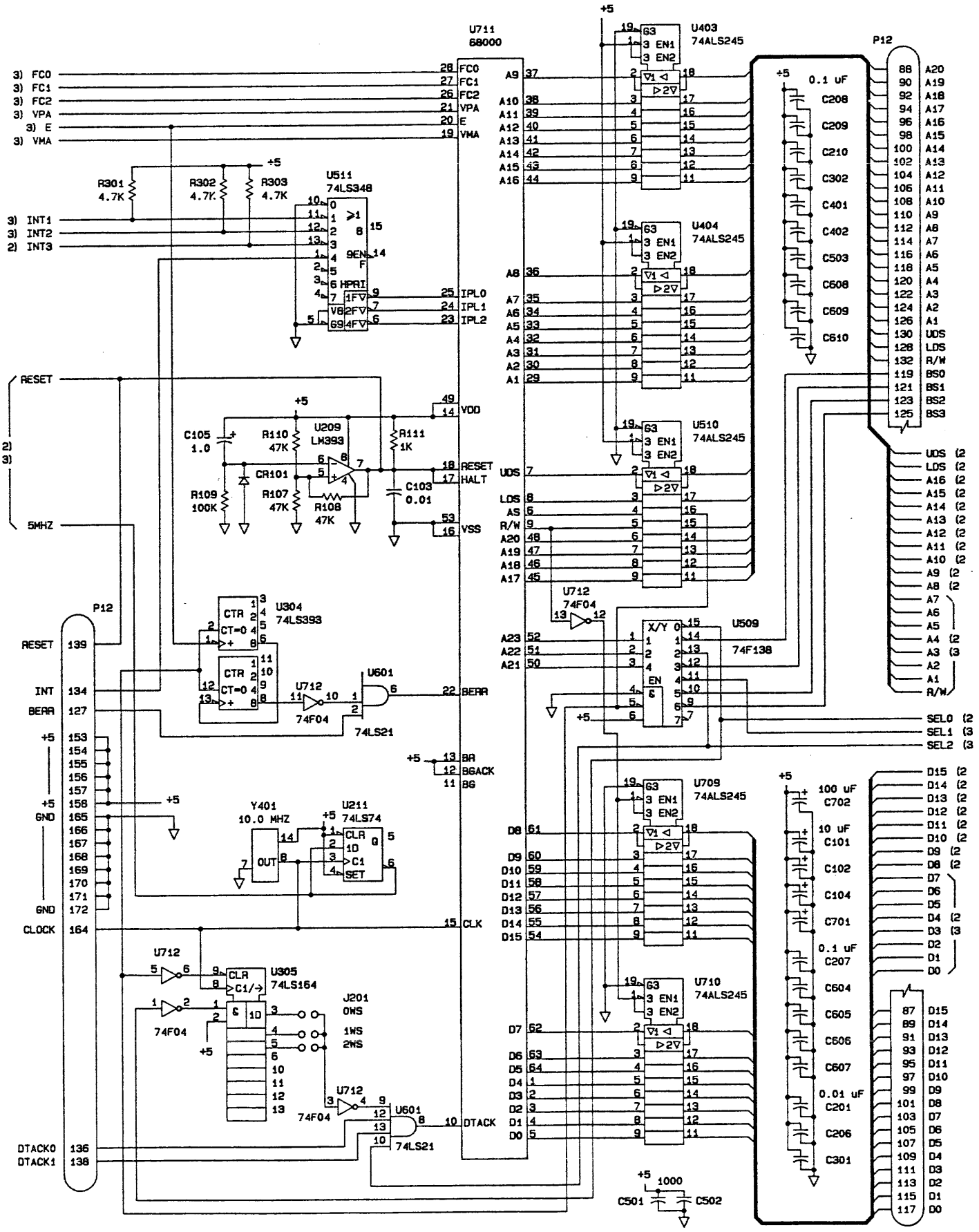
SPECIFICATIONS:

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

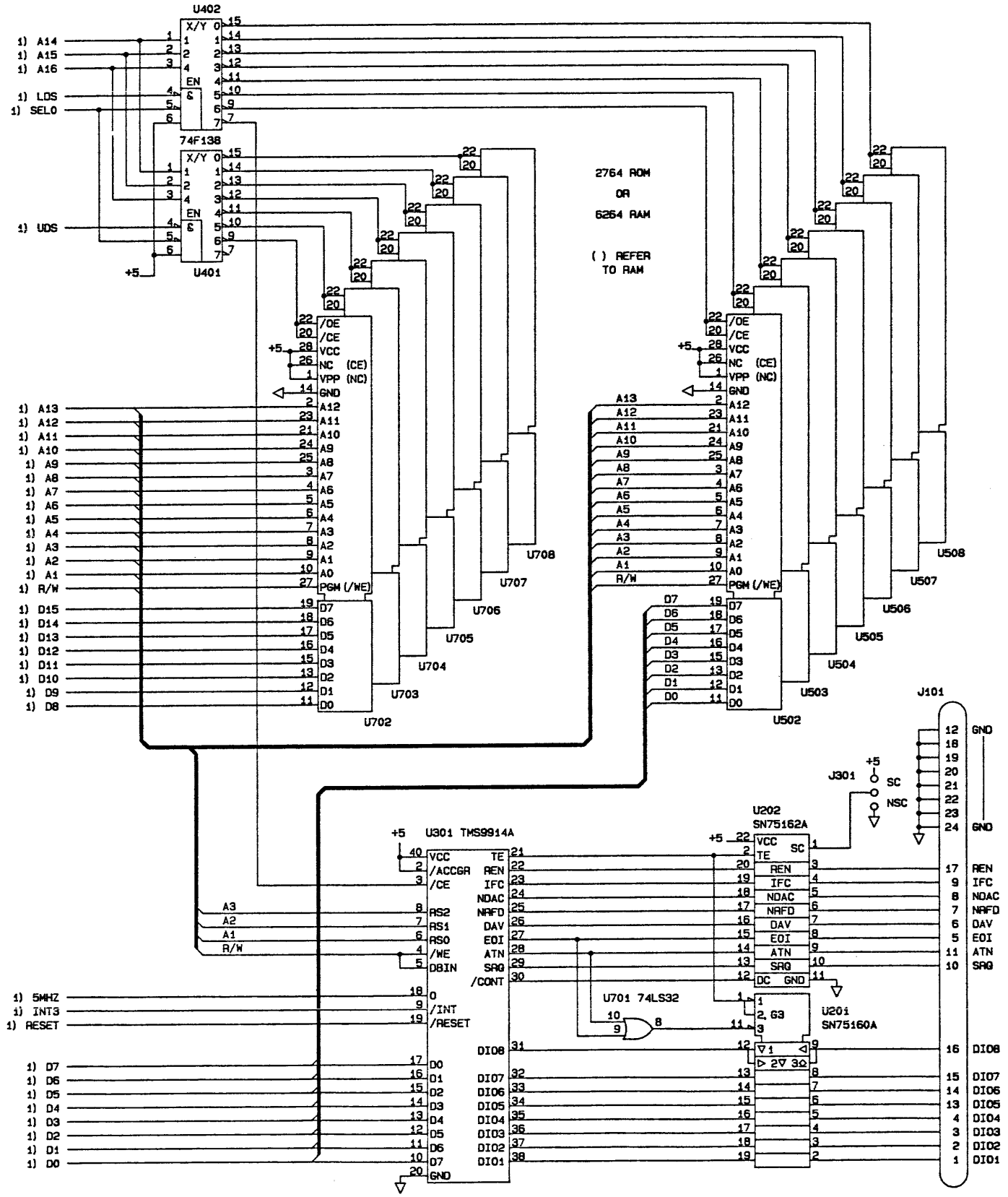


TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604

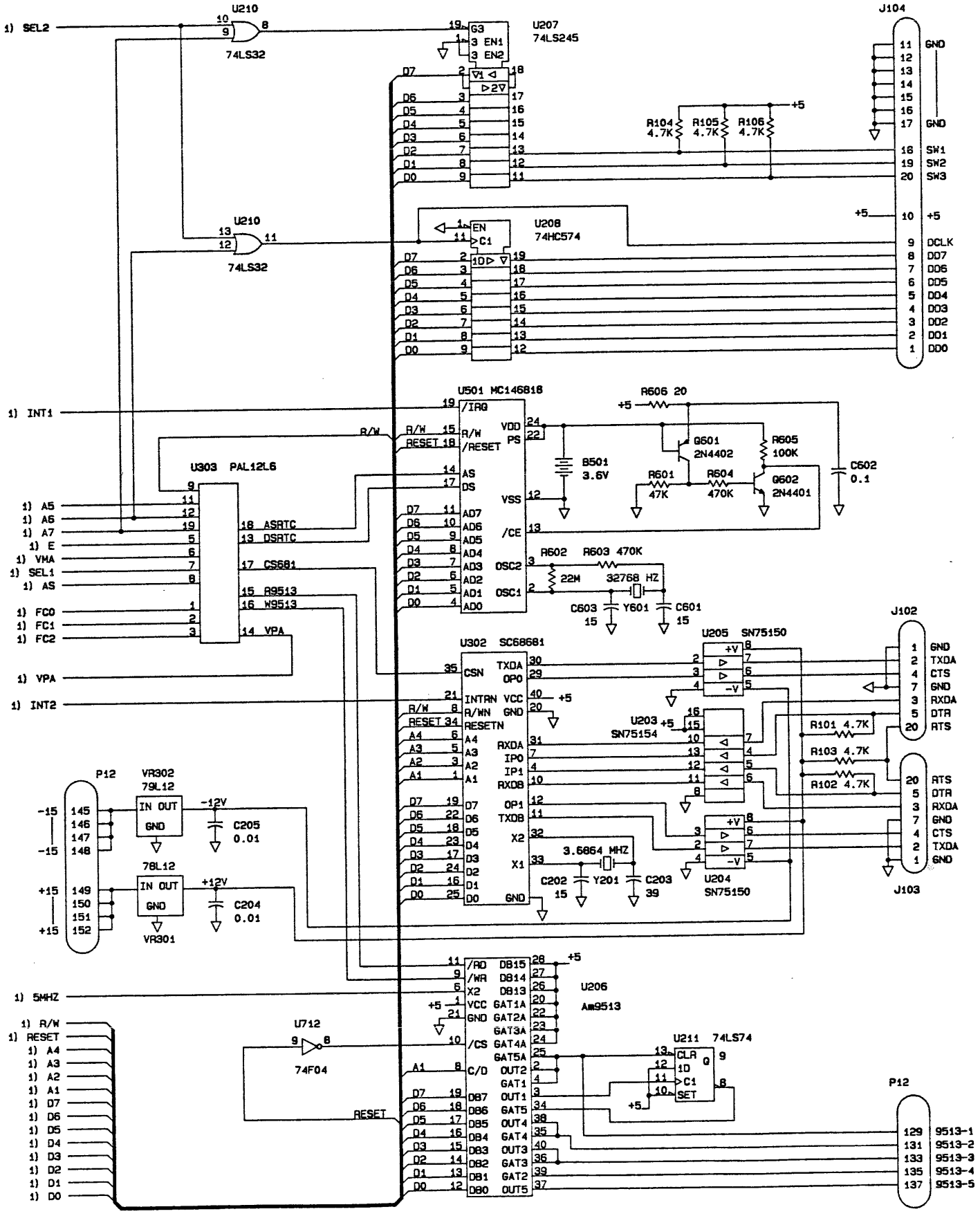
Phone 801-224-6550
Telex 296438



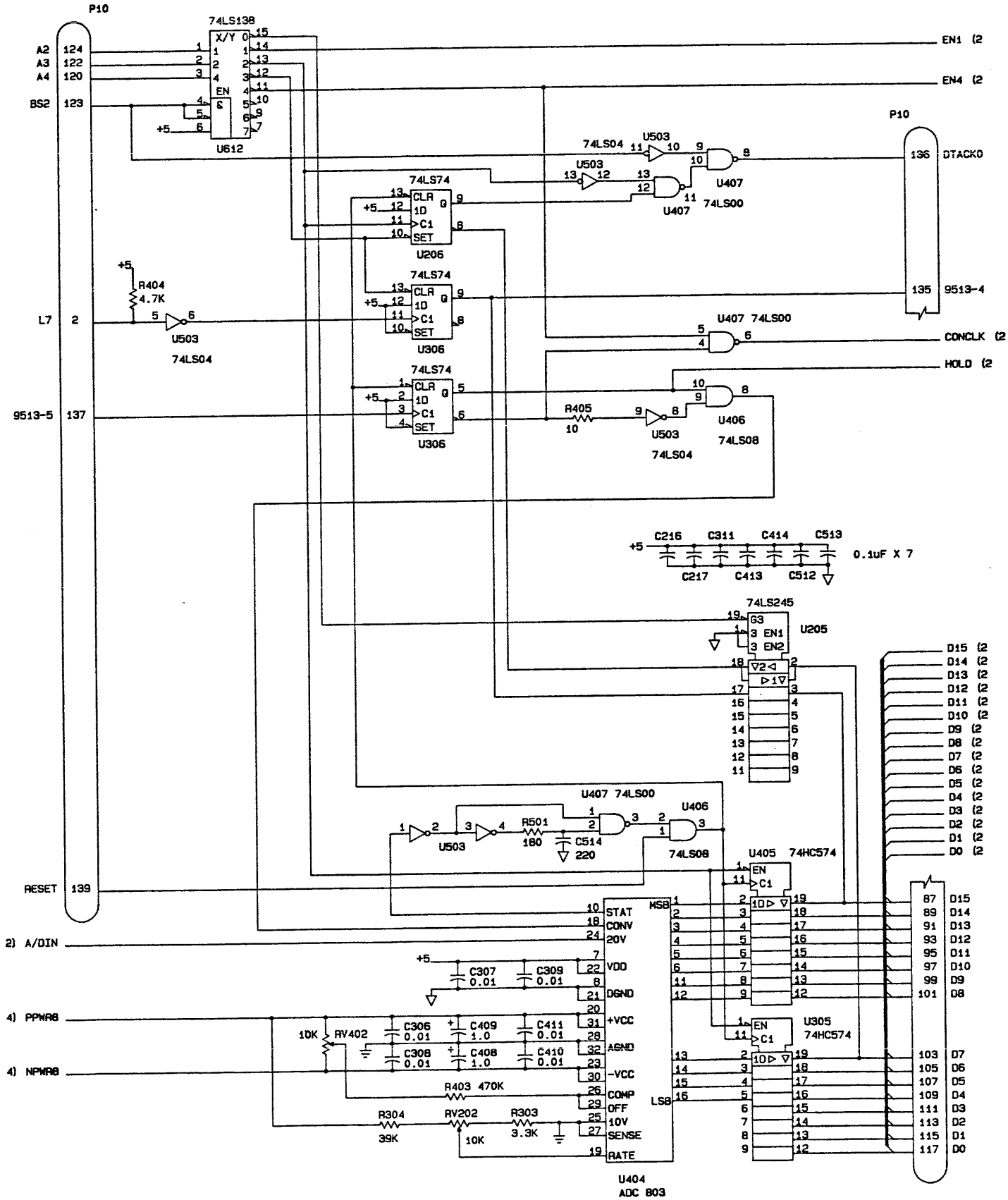
PRODUCT	NAME 7000-670-02-C	VERSION	PAGE
7000 MDAS	PROCESSOR BOARD	C	1 OF 3



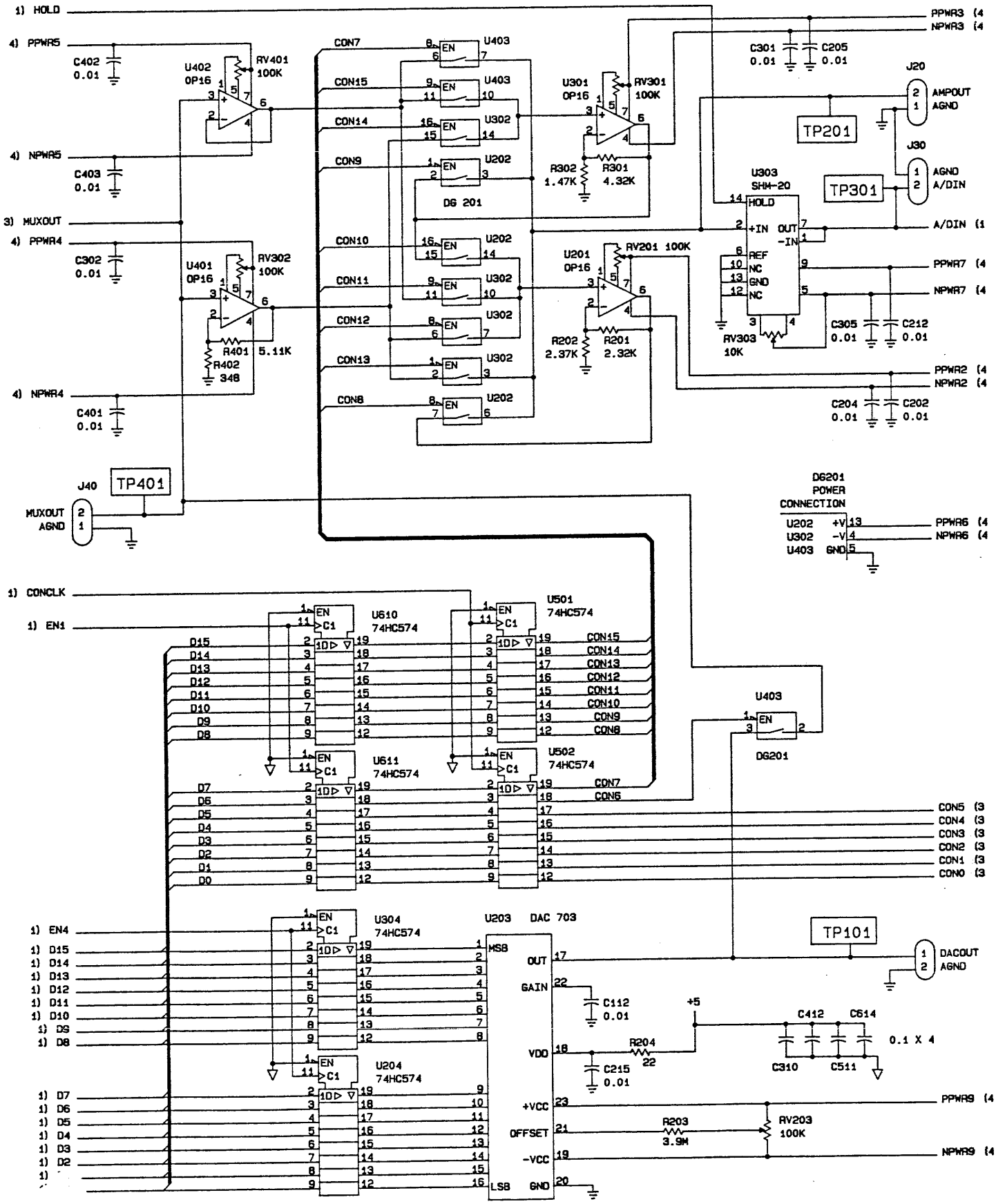
PRODUCT	NAME 7000-670-02-C	VERSION	PAGE
7000 MDAS	PROCESSOR BOARD	C	2 OF 3



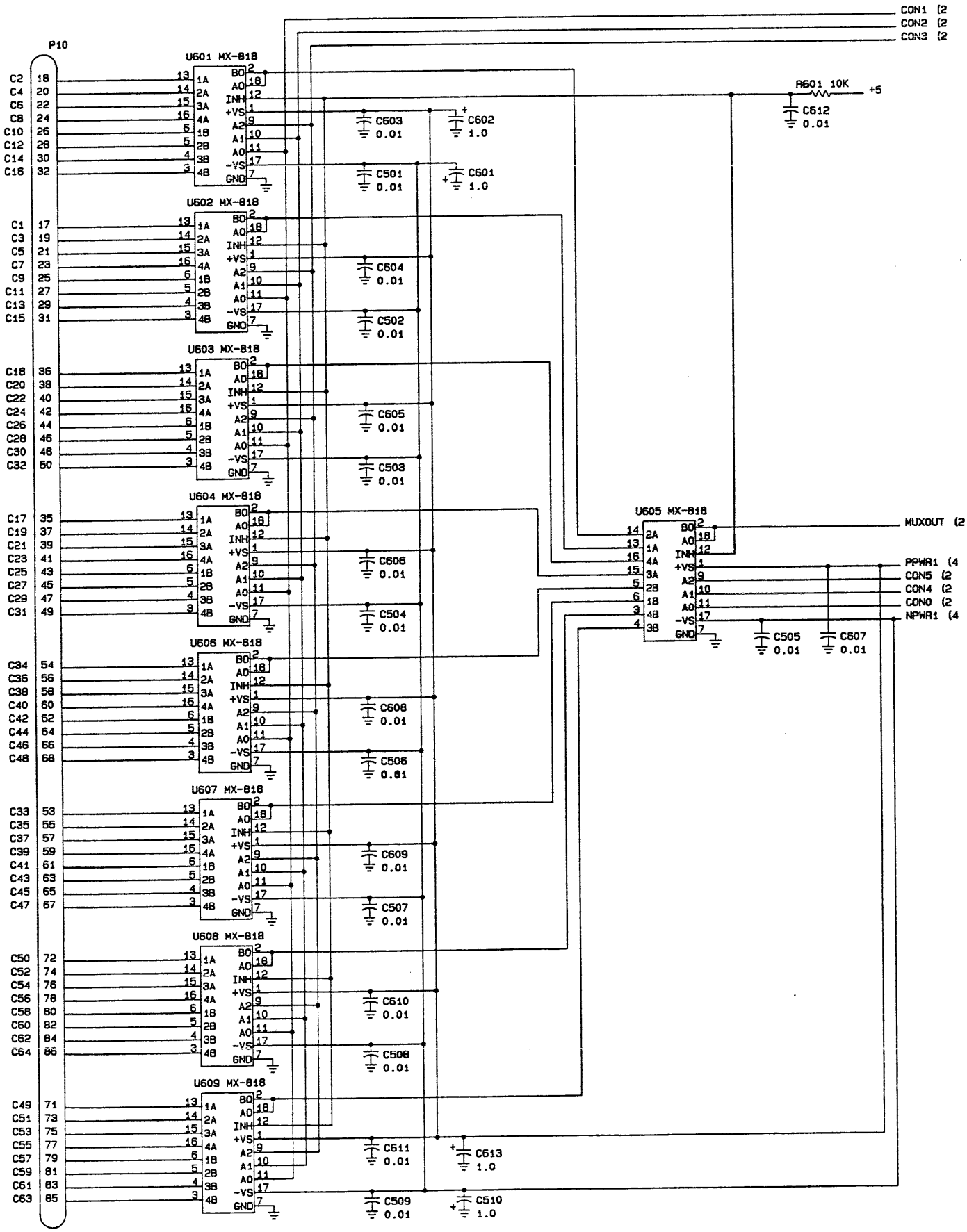
PRODUCT	NAME 7000-670-02-C	VERSION	PAGE
7000 MDAS	PROCESSOR BOARD	C	3 OF 3



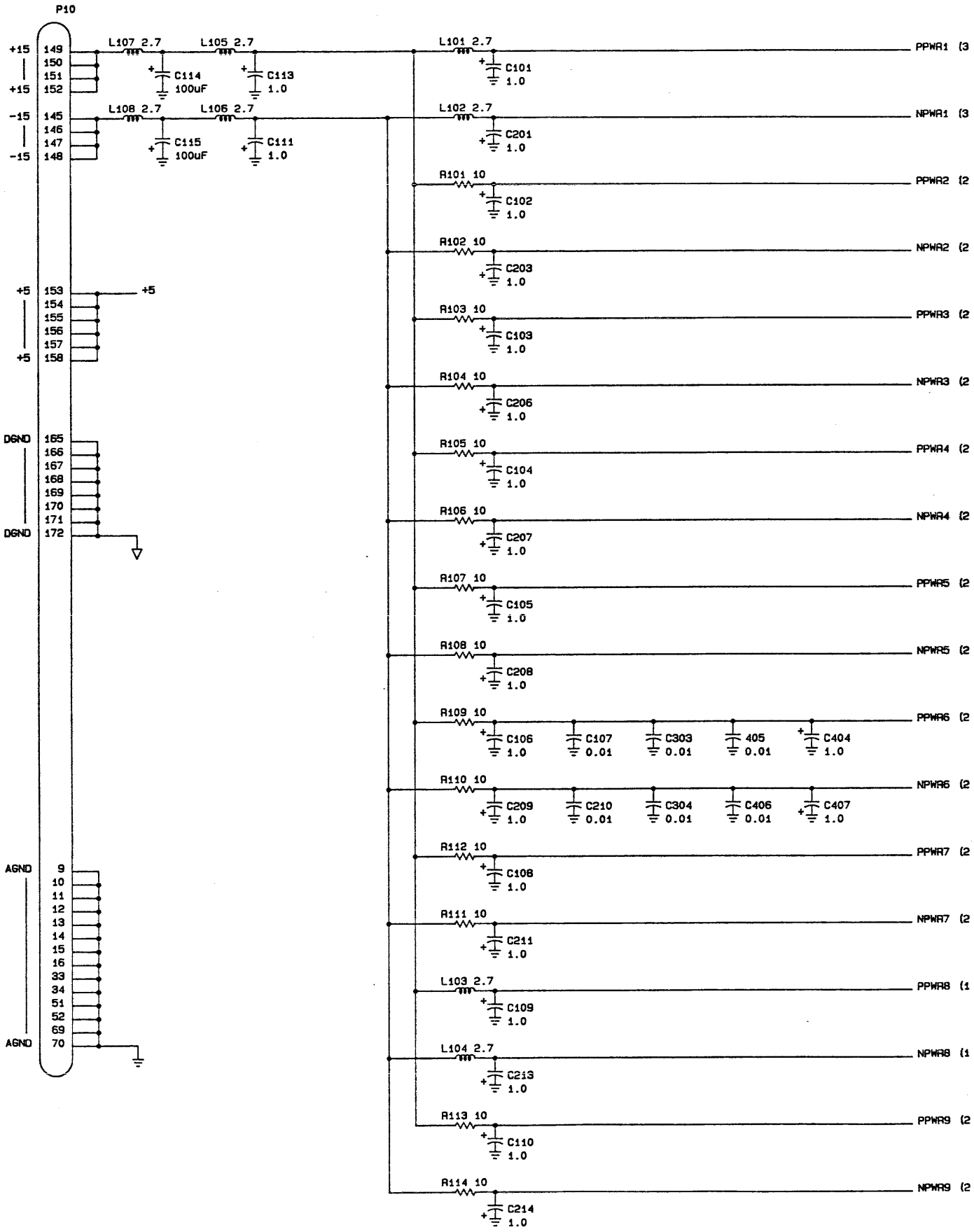
PRODUCT	NAME 7000-670-03-C	VERSION	PAGE
7000 MDAS	12 BIT 400KHZ ANALOG BOARD	CS4-A	1 OF 4



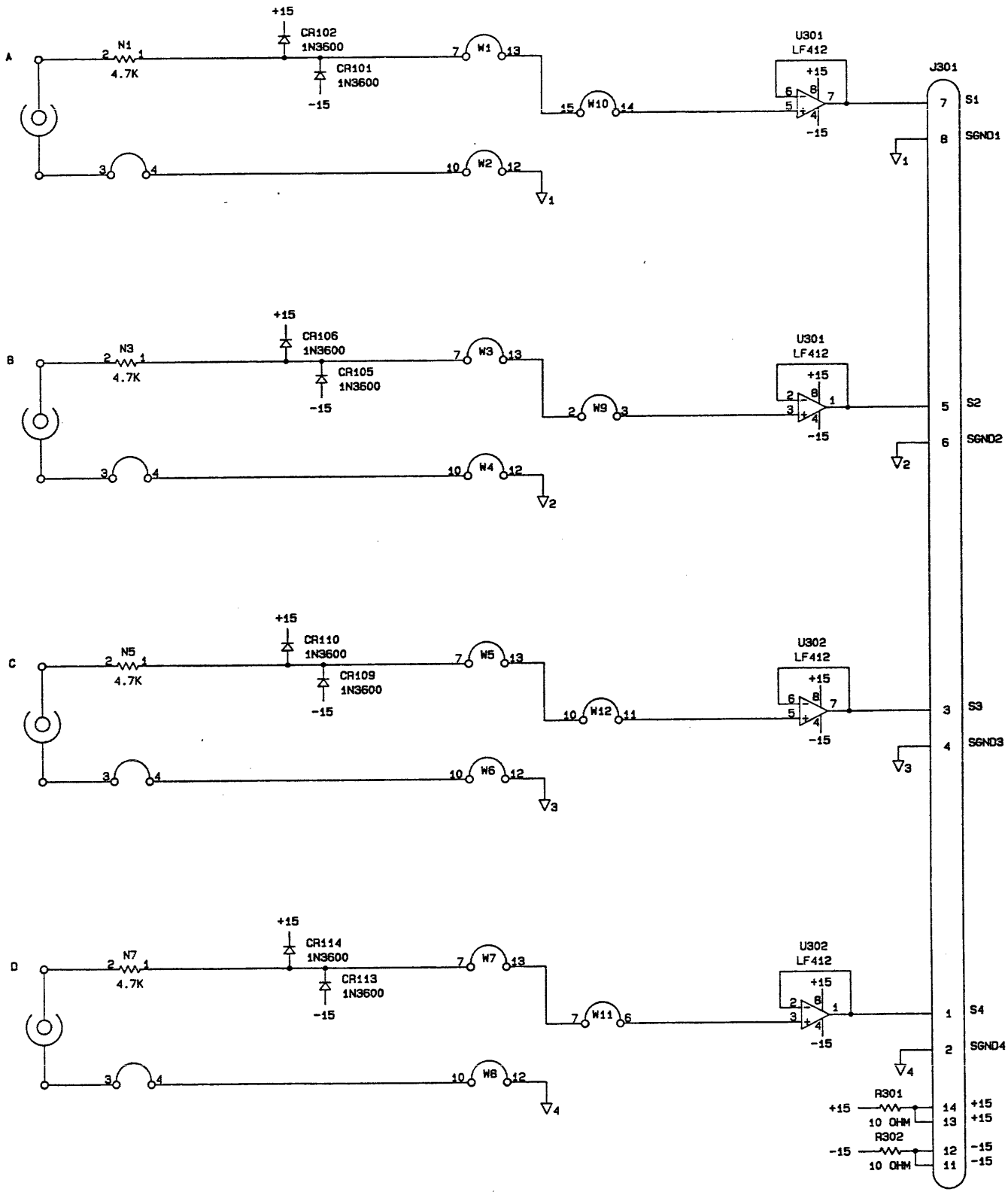
PRODUCT	NAME 7000-670-03-C	VERSION	PAGE
7000 MDAS	12 BIT 400KHZ ANALOG BOARD	CS4-A	2 OF 4



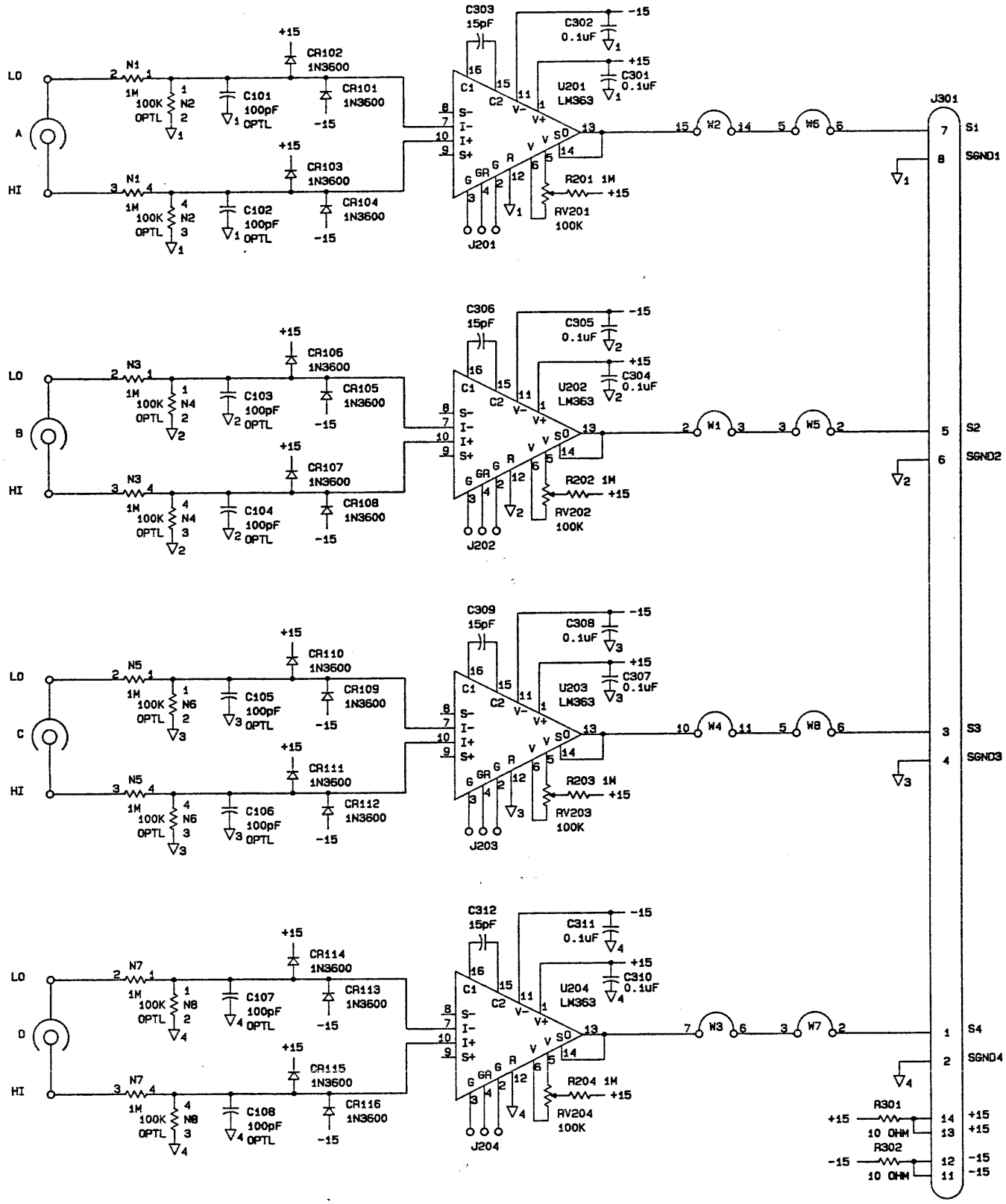
PRODUCT	NAME 7000-670-03-C	VERSION	PAGE
7000 MDAS	12 BIT 400KHZ ANALOG BOARD	CS4-A	3 OF 4



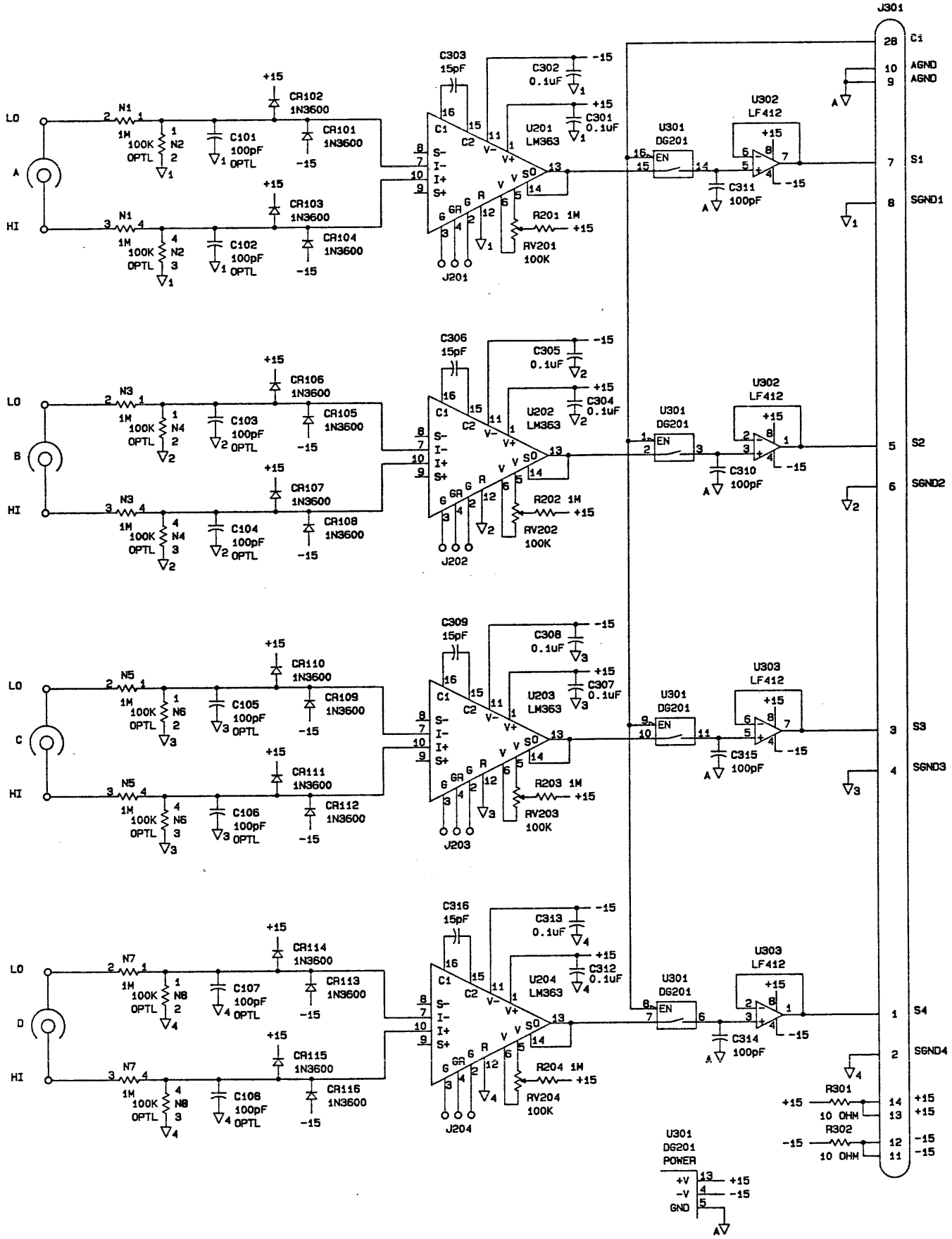
PRODUCT	NAME 7000-670-03-C	VERSION	PAGE
7000 MDAS	12 BIT 400KHZ ANALOG BOARD	CS4-A	4 OF 4



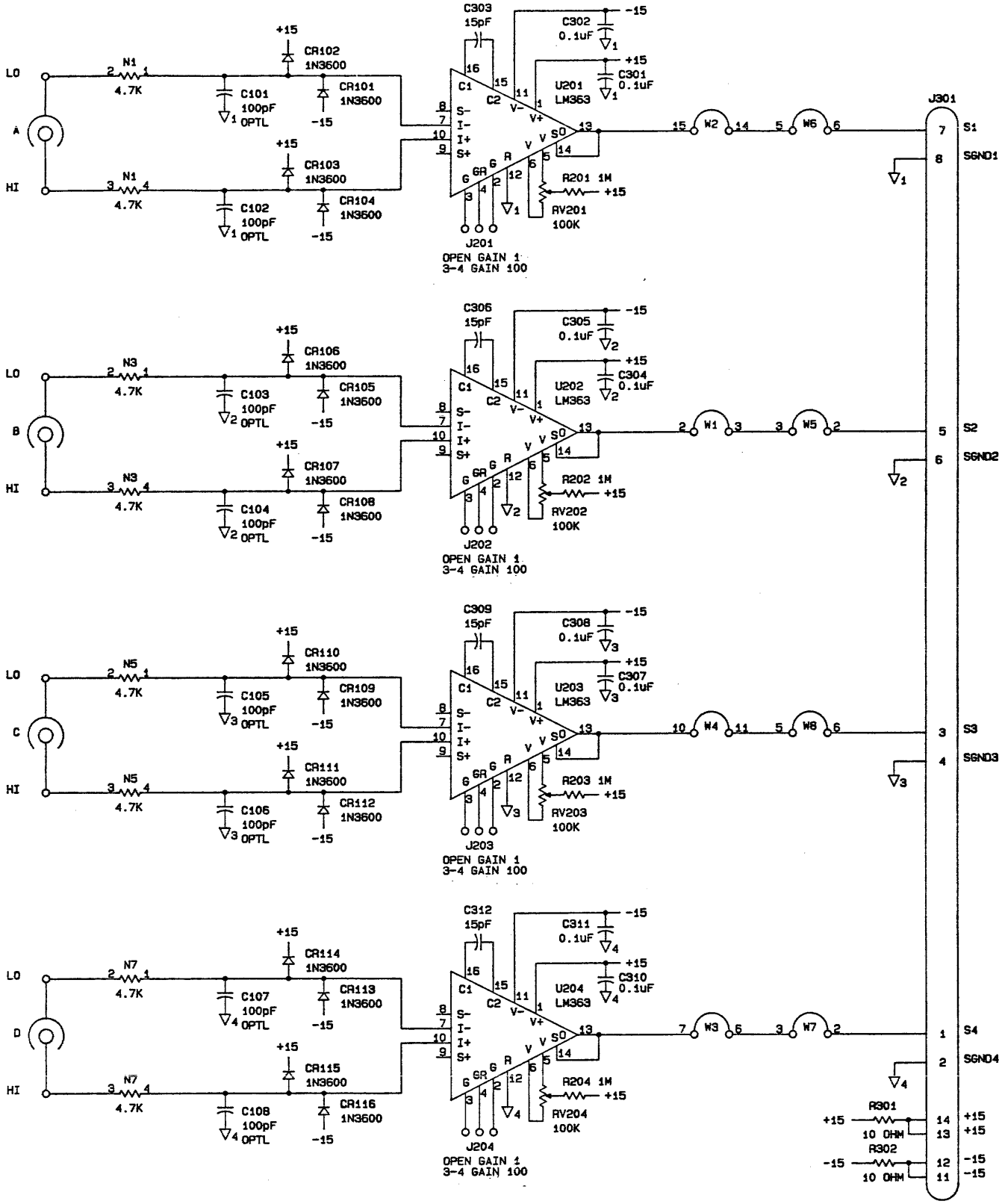
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	SINGLE ENDED ANALOG INPUT BOARD	A1	1 OF 1



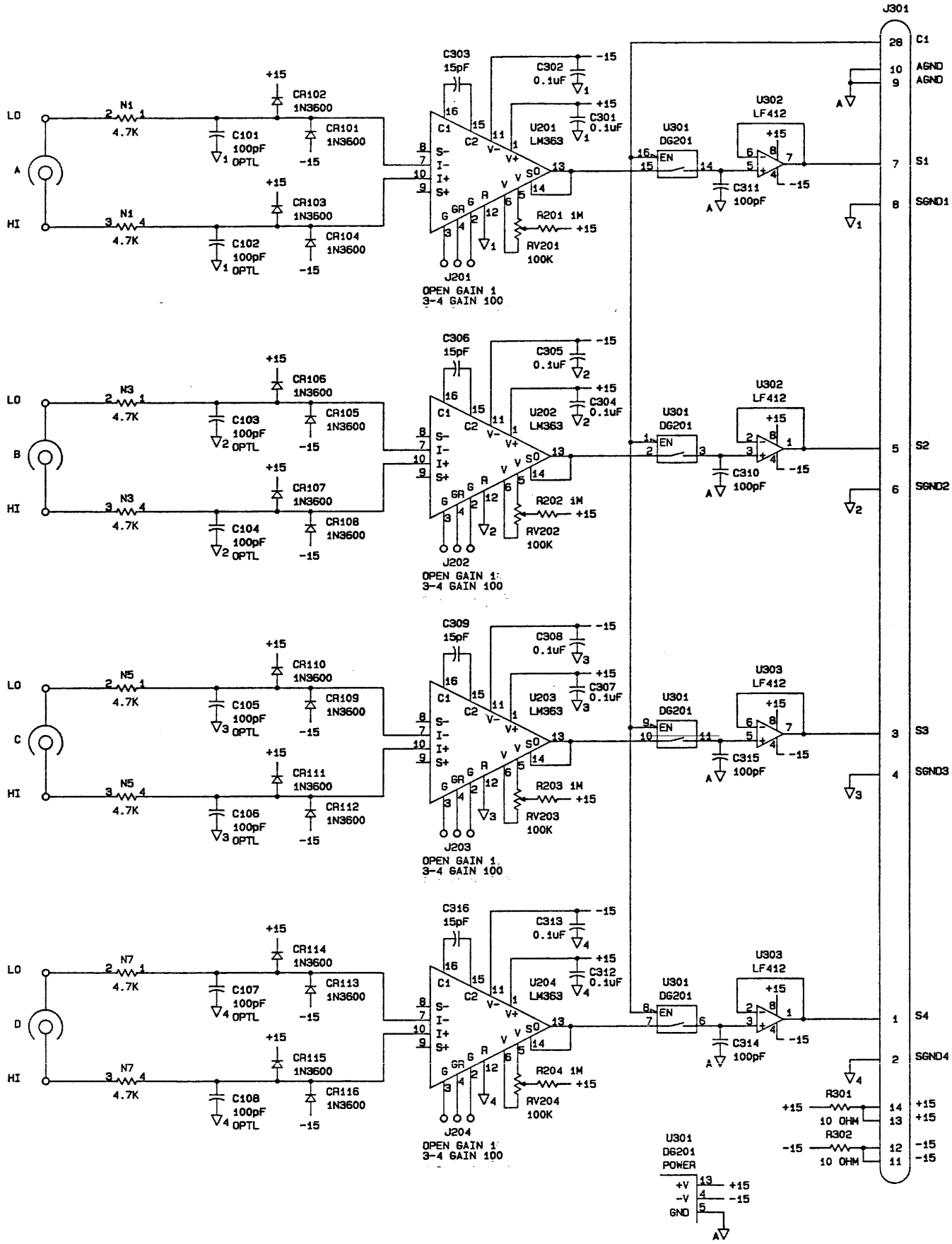
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFFERENTIAL ANALOG INPUT BOARD	A2	1 OF 1



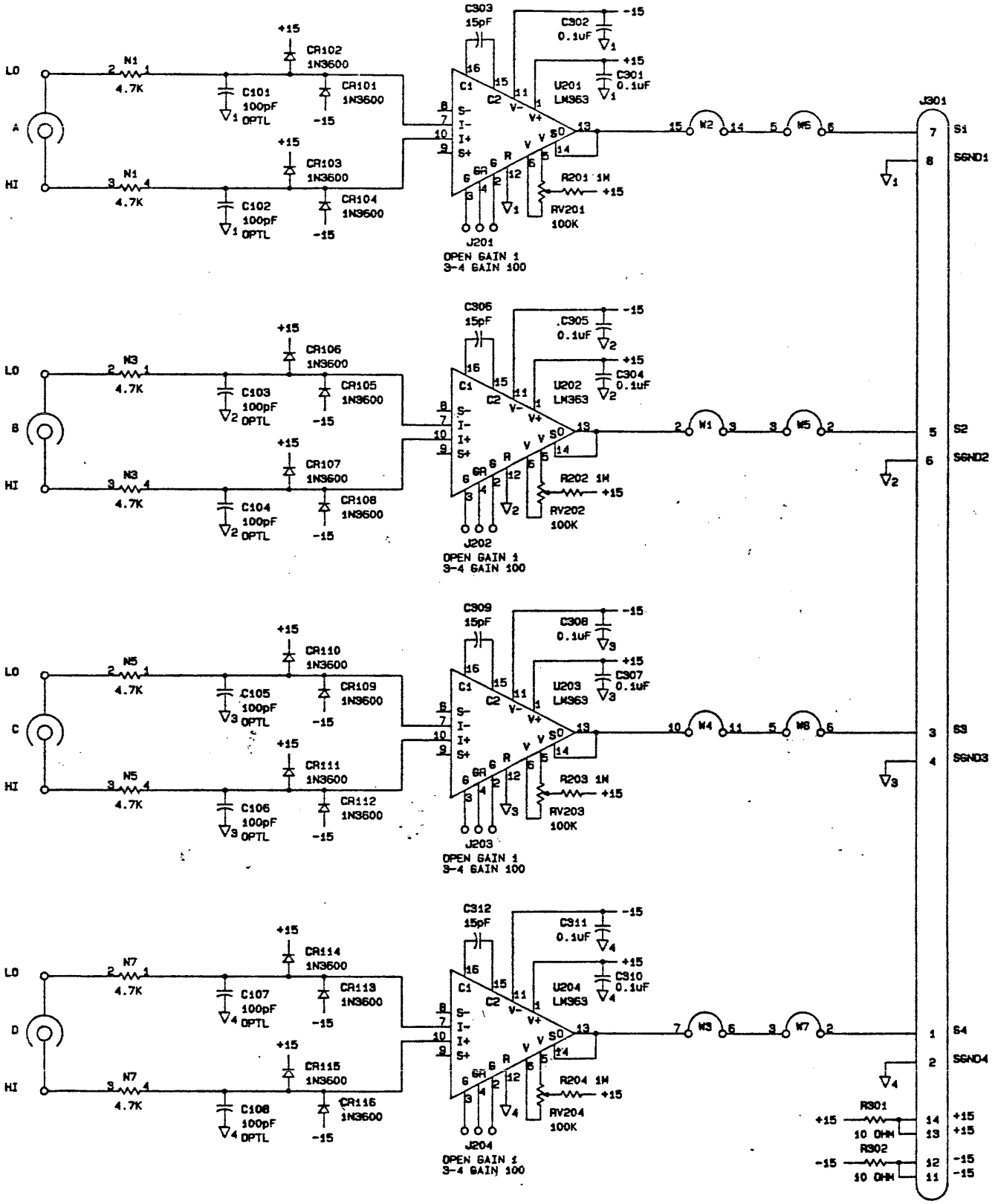
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFF ANALOG INPUT BOARD W/HOLD	A3	1 OF 1



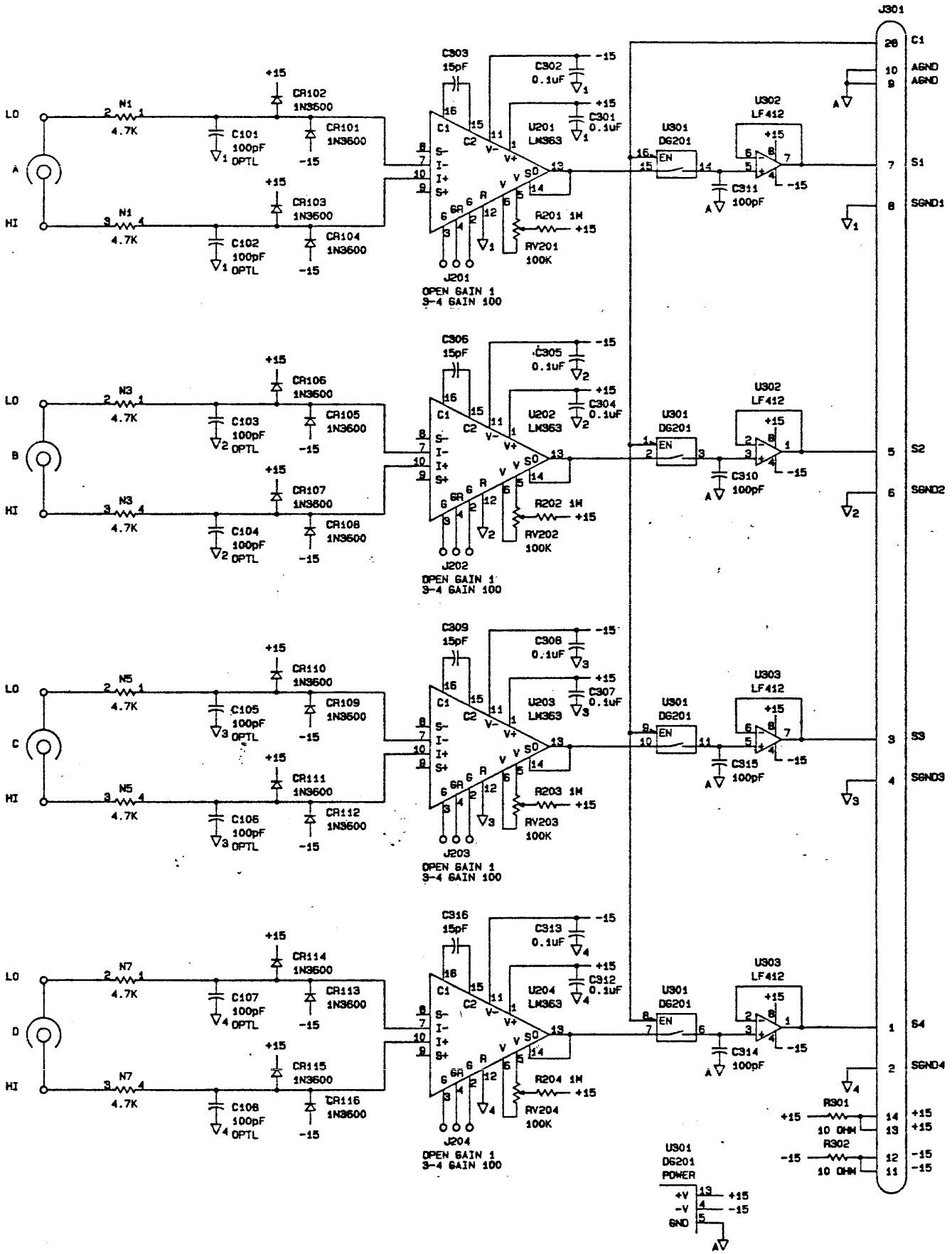
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFFERENTIAL ANALOG INPUT BOARD	A4	1 OF 1



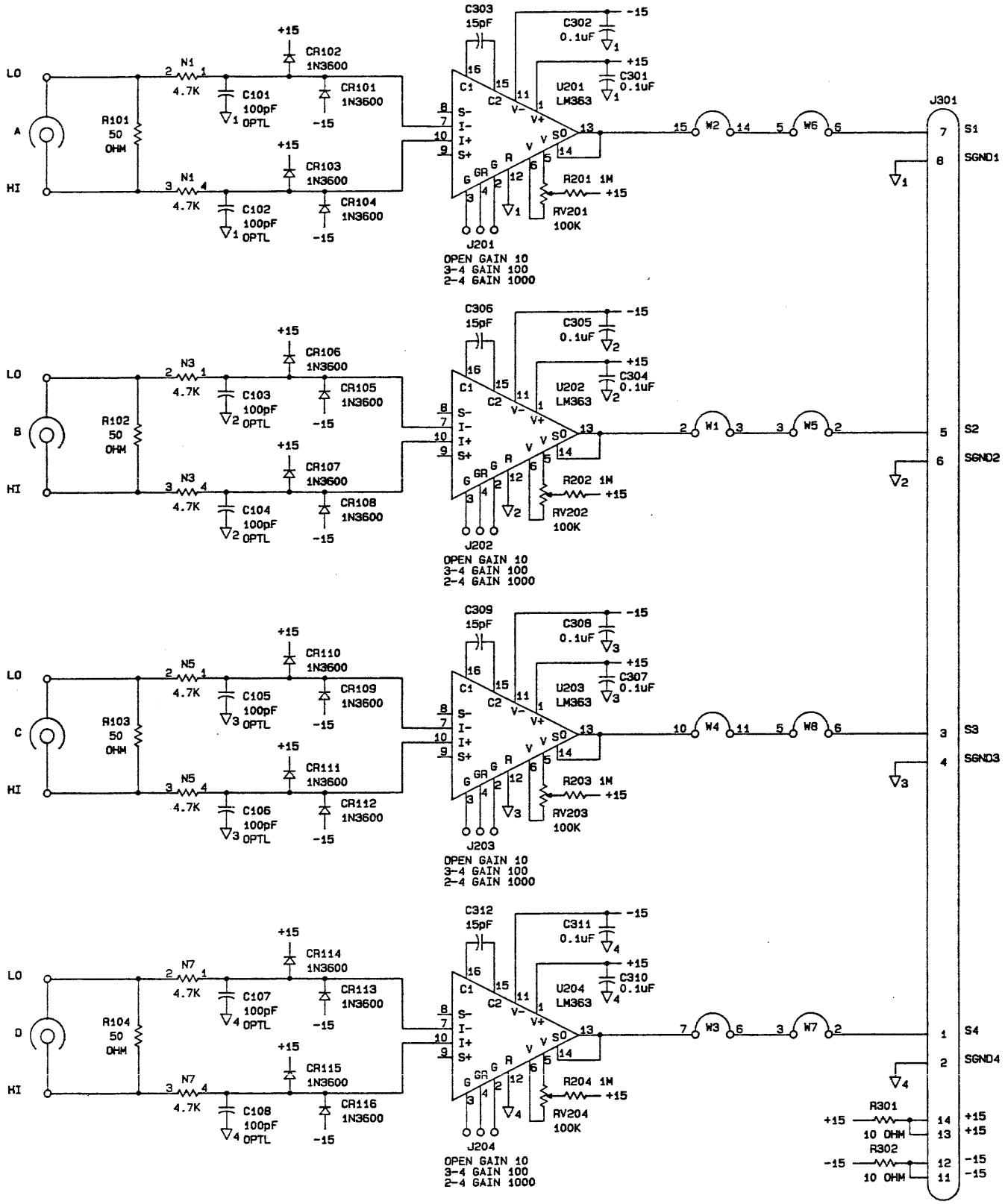
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFF ANALOG INPUT BOARD W/HOLD	A5	1 OF 1



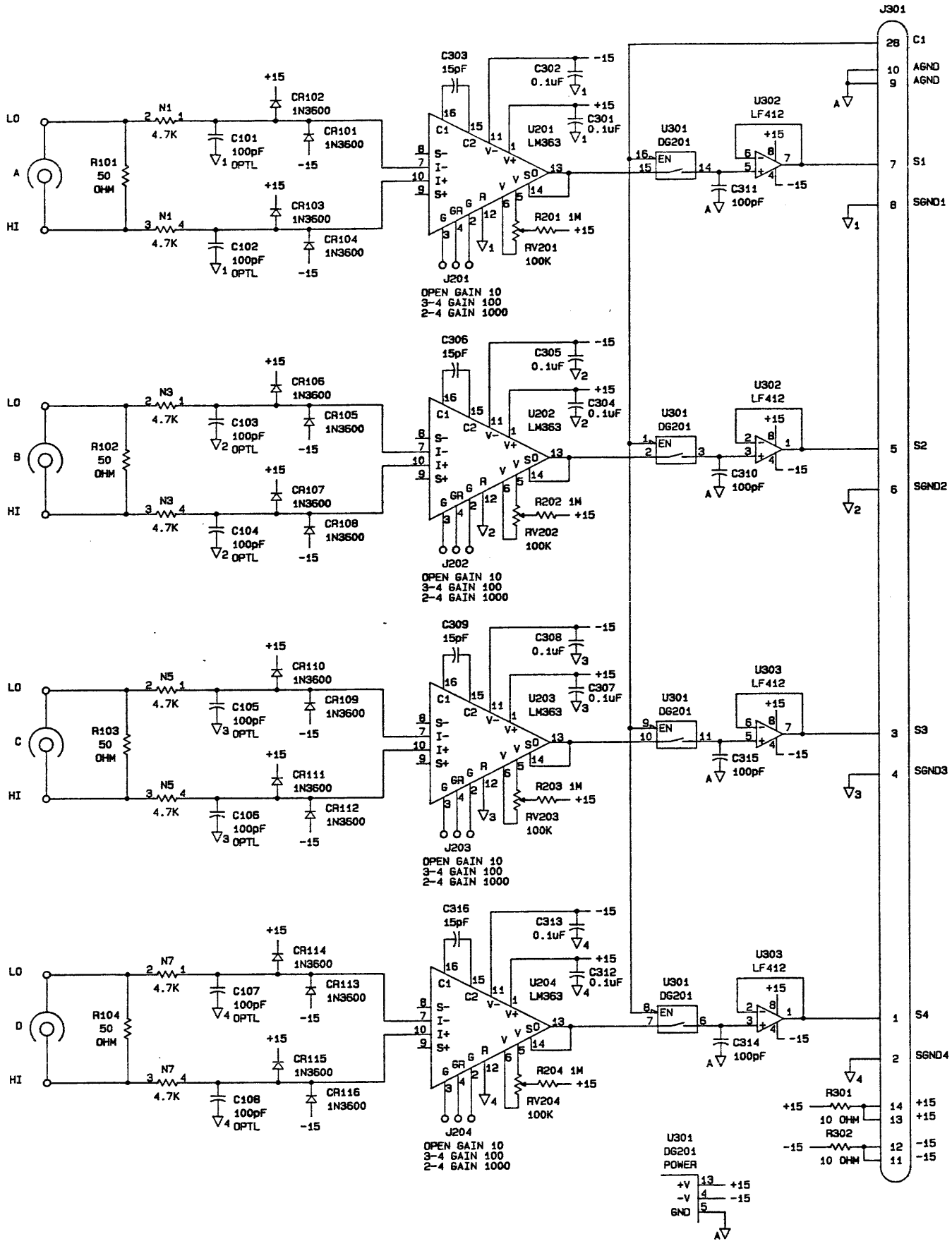
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFFERENTIAL ANALOG INPUT BOARD	A6	1 OF 1



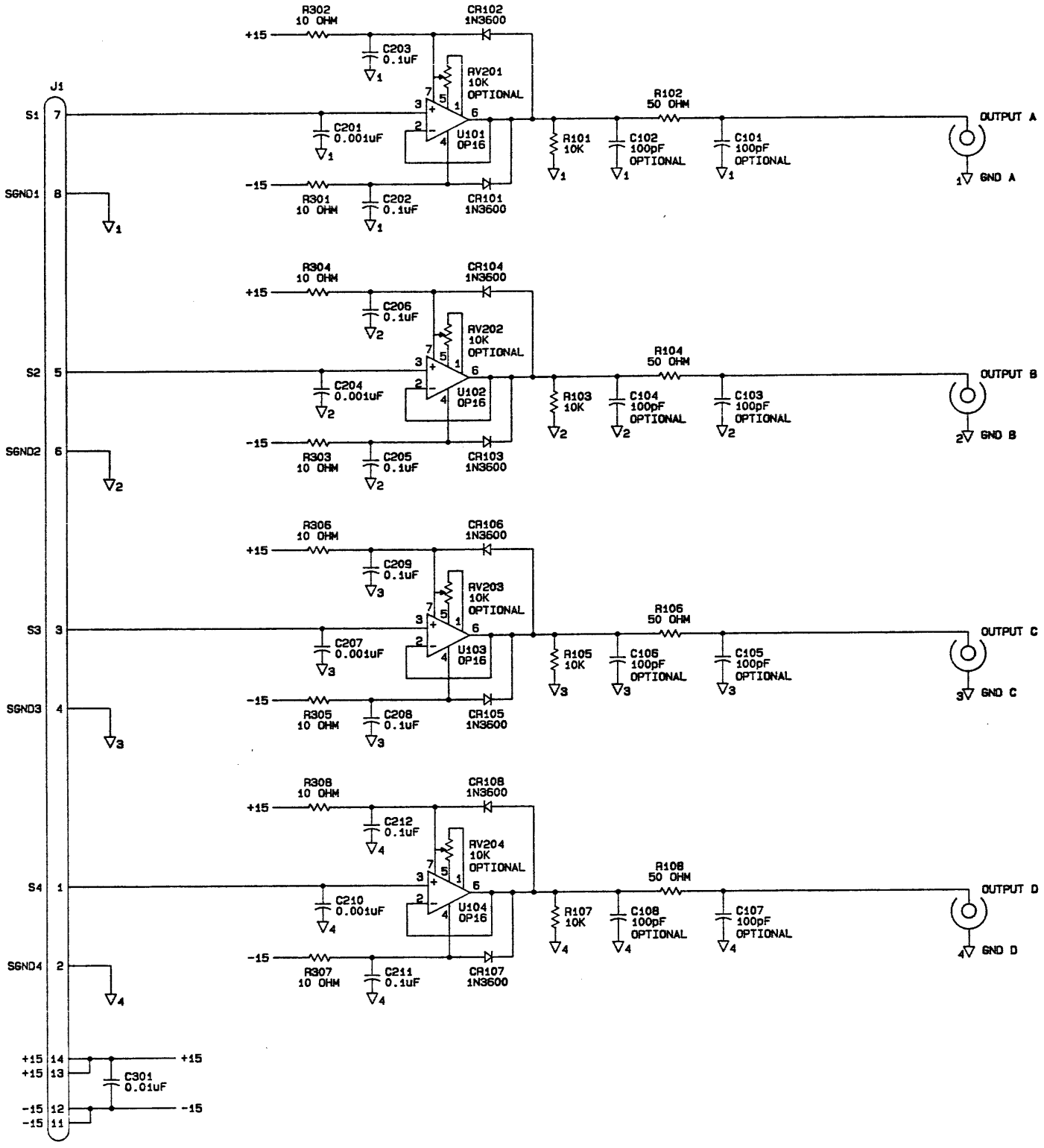
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	DIFF ANALOG INPUT BOARD W/HOLD	A7	1 OF 1



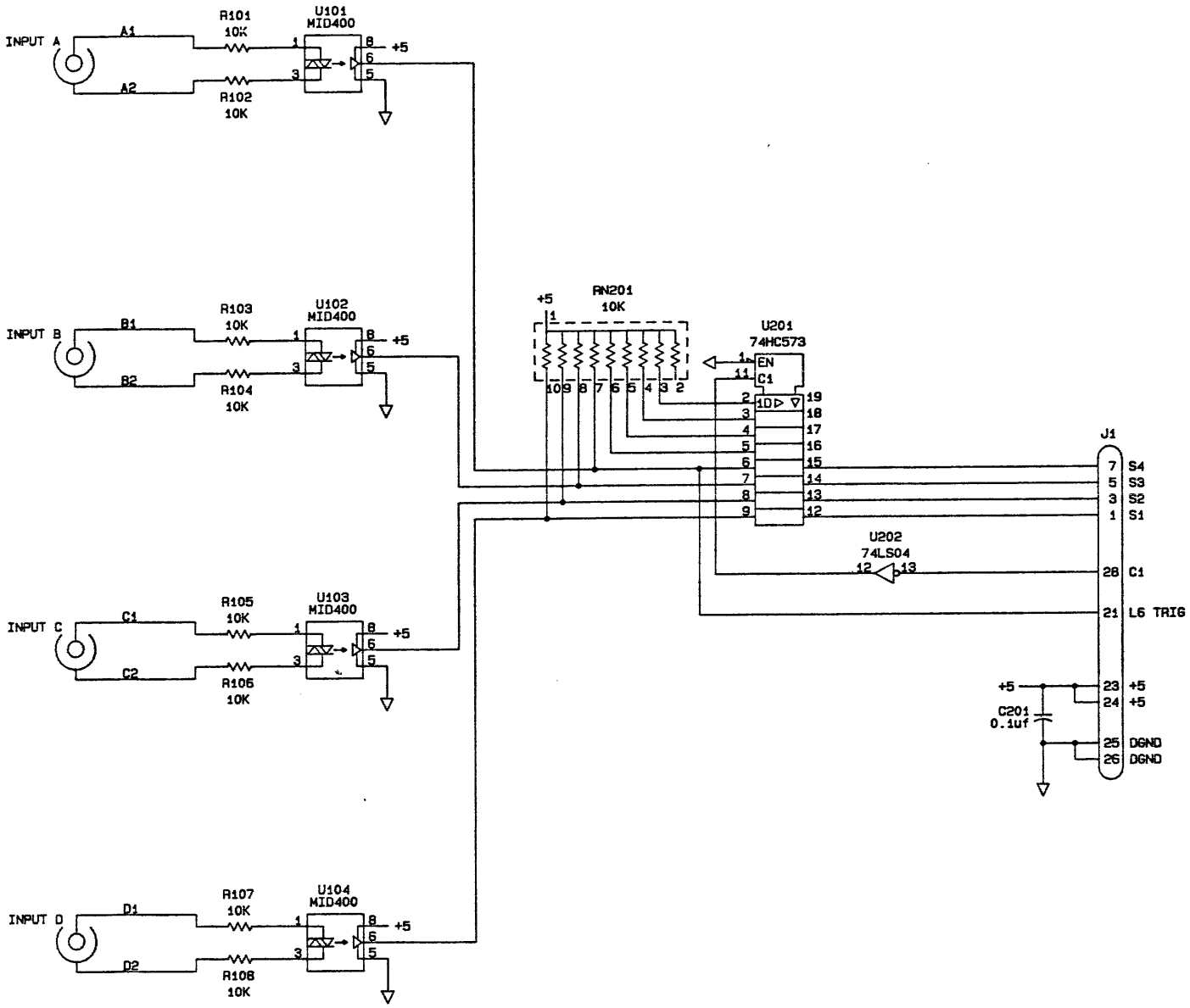
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	4-20 mA CURRENT INPUT BOARD	C1	1 OF 1



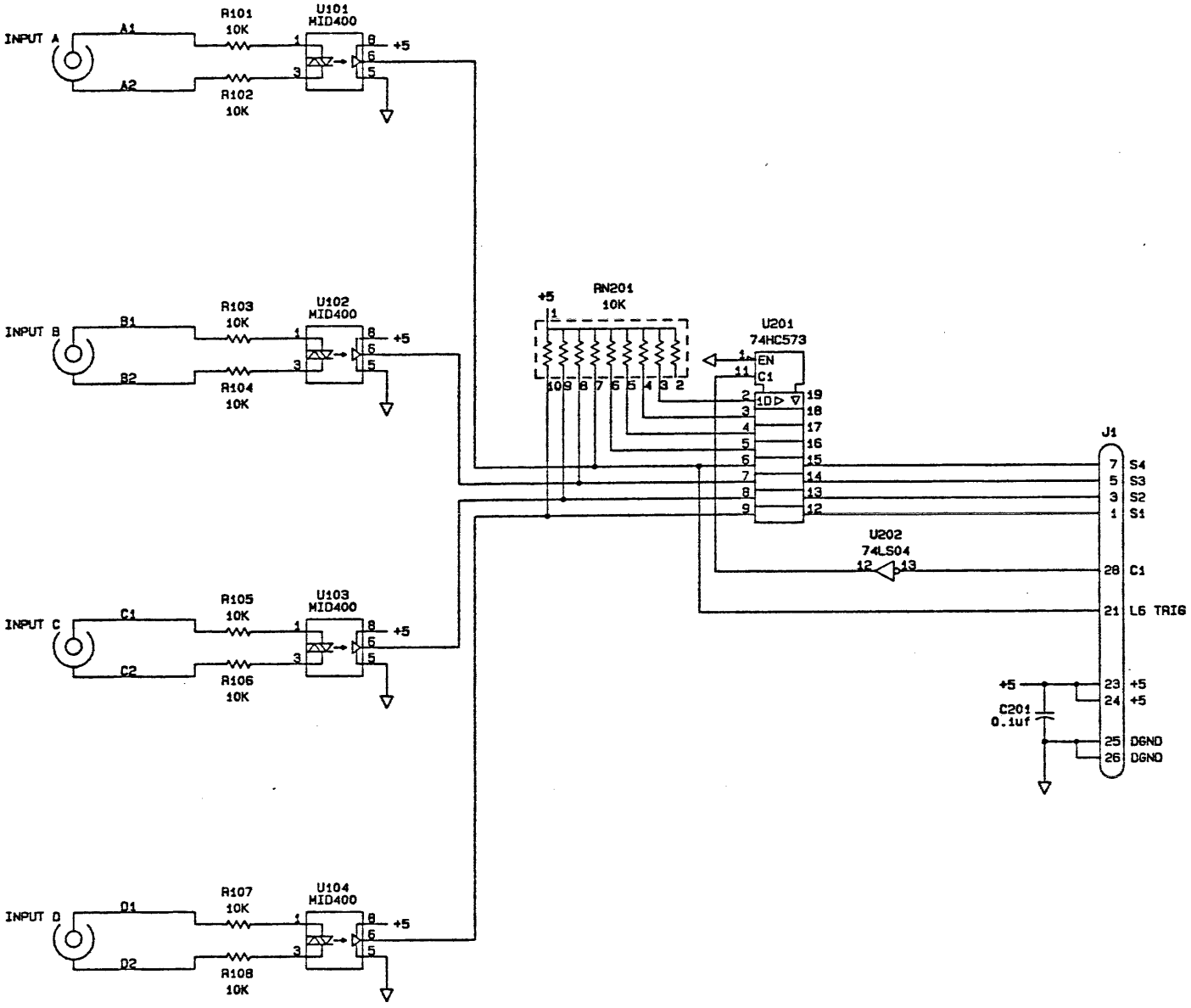
PRODUCT	NAME 7000-670-21-A	VERSION	PAGE
7000 MDAS	4-20 mA CURRENT INP BD W/HOLD	C2	1 OF 1



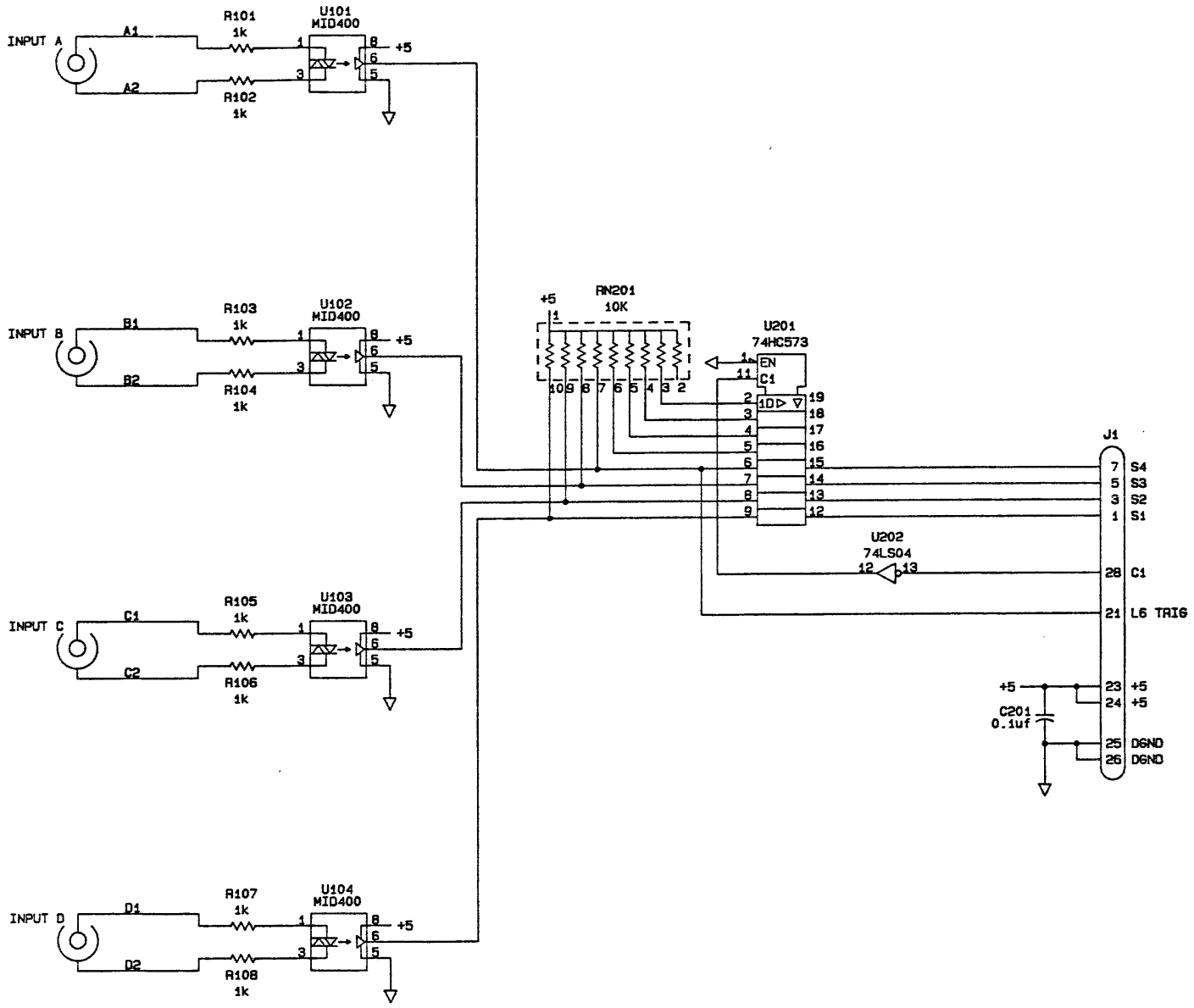
PRODUCT	NAME 7000-670-22-A	VERSION	PAGE
7000-MDAS	±10 VOLT ANALOG OUTPUT BOARD	S1	1 OF 1



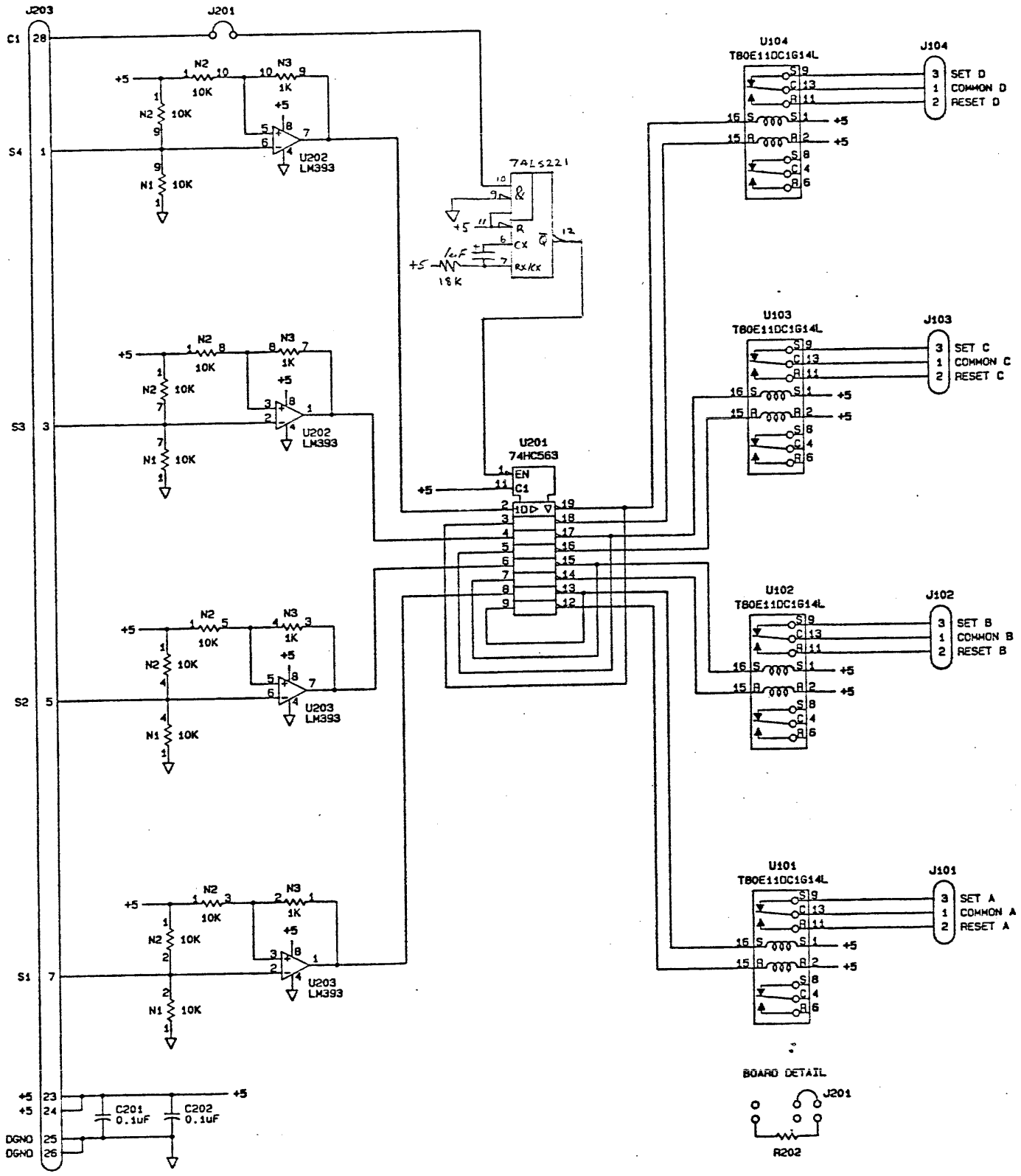
PRODUCT	NAME 7000-670-23-A	VERSION	PAGE
7000 MDAS	110/220 AC DETECT D-INPUT BD	R1	1 OF 1



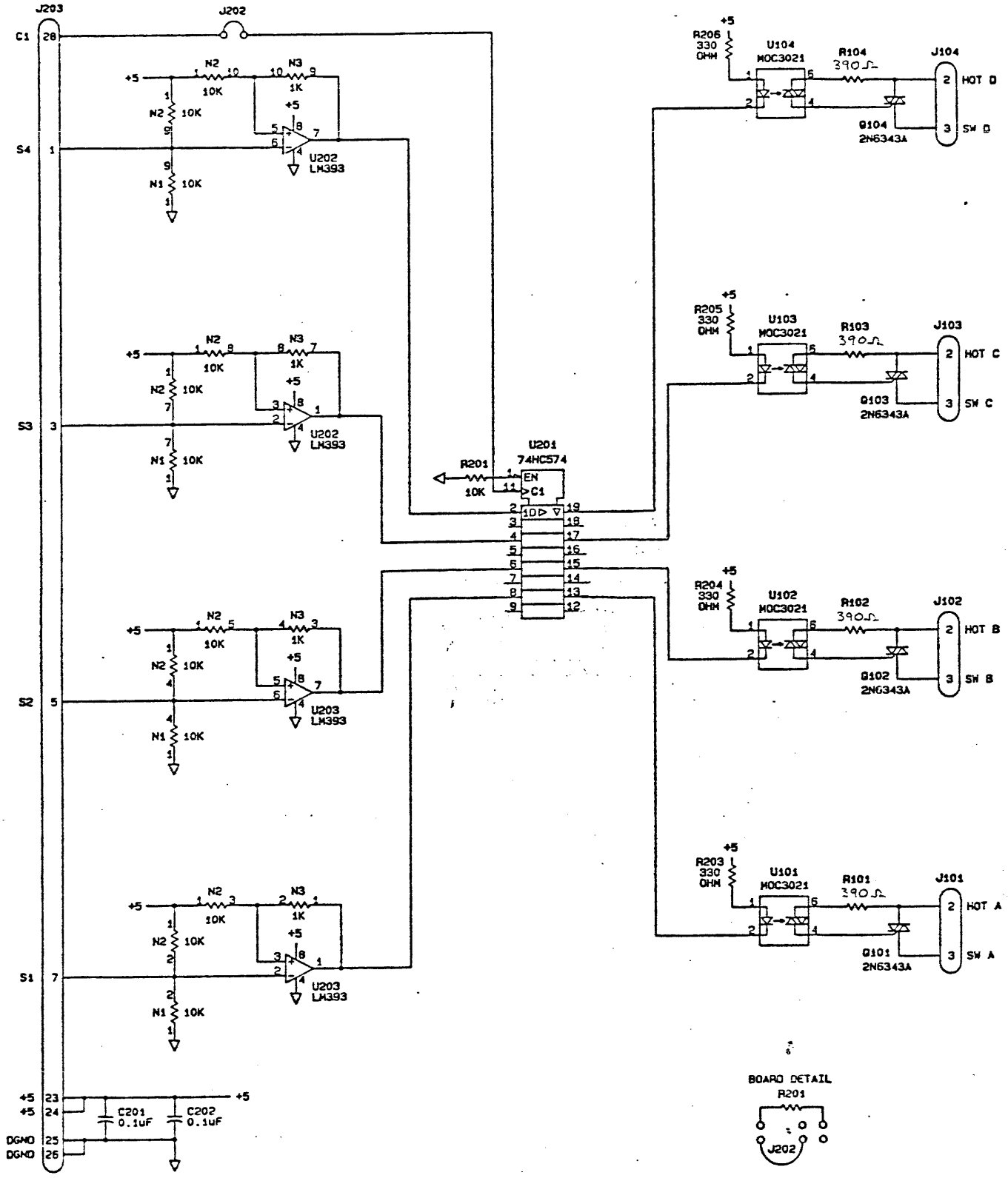
PRODUCT	NAME 7000-670-23-A	VERSION	PAGE
7000 MDAS	HI'TTL-100V DETECT D-INPUT BD	R2	1 OF 1



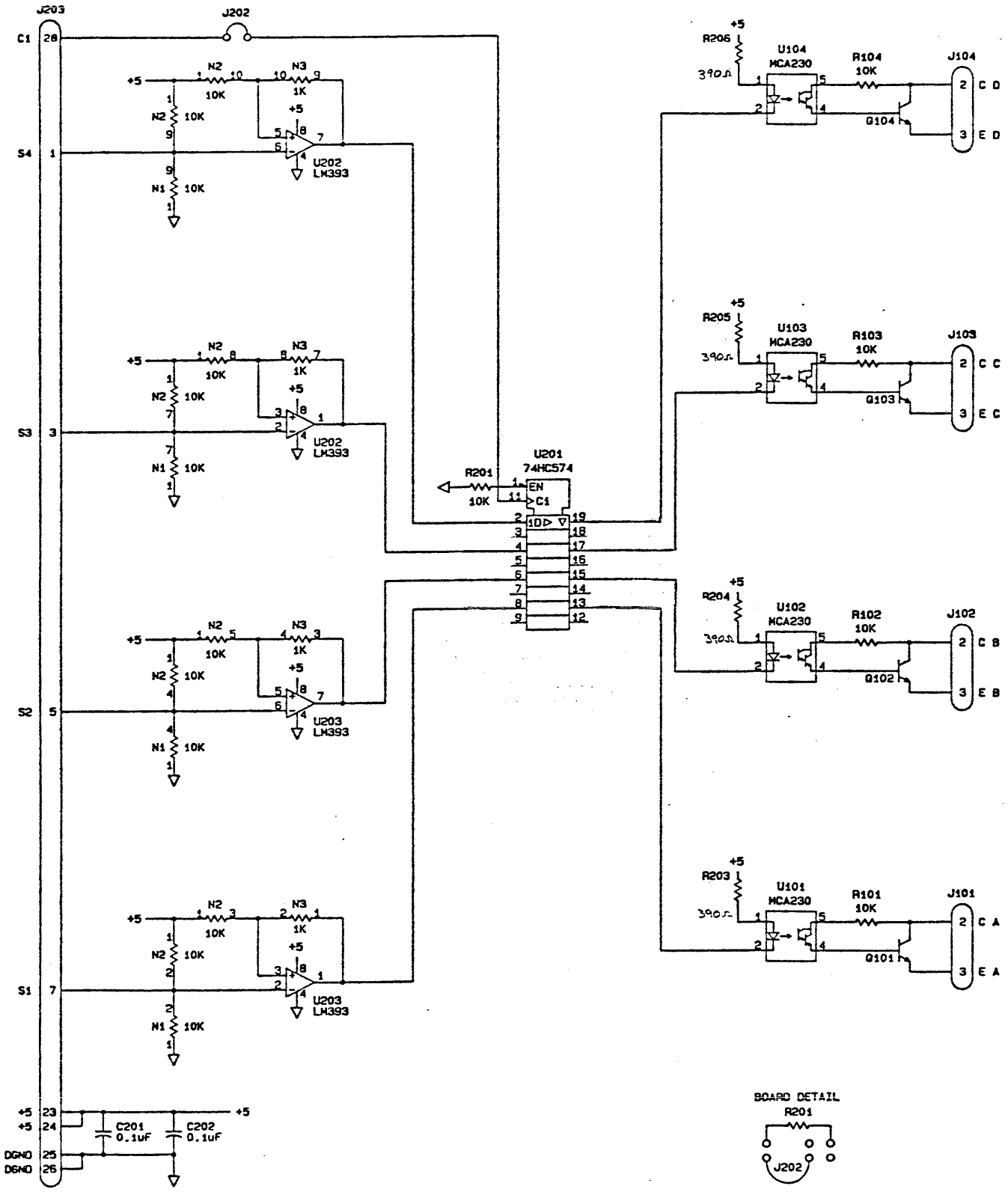
PRODUCT	NAME 7000-670-23-A	VERSION	PAGE
7000 MDAS	LOW TTL DETECTION D-INPUT BD	R3	1 OF 1



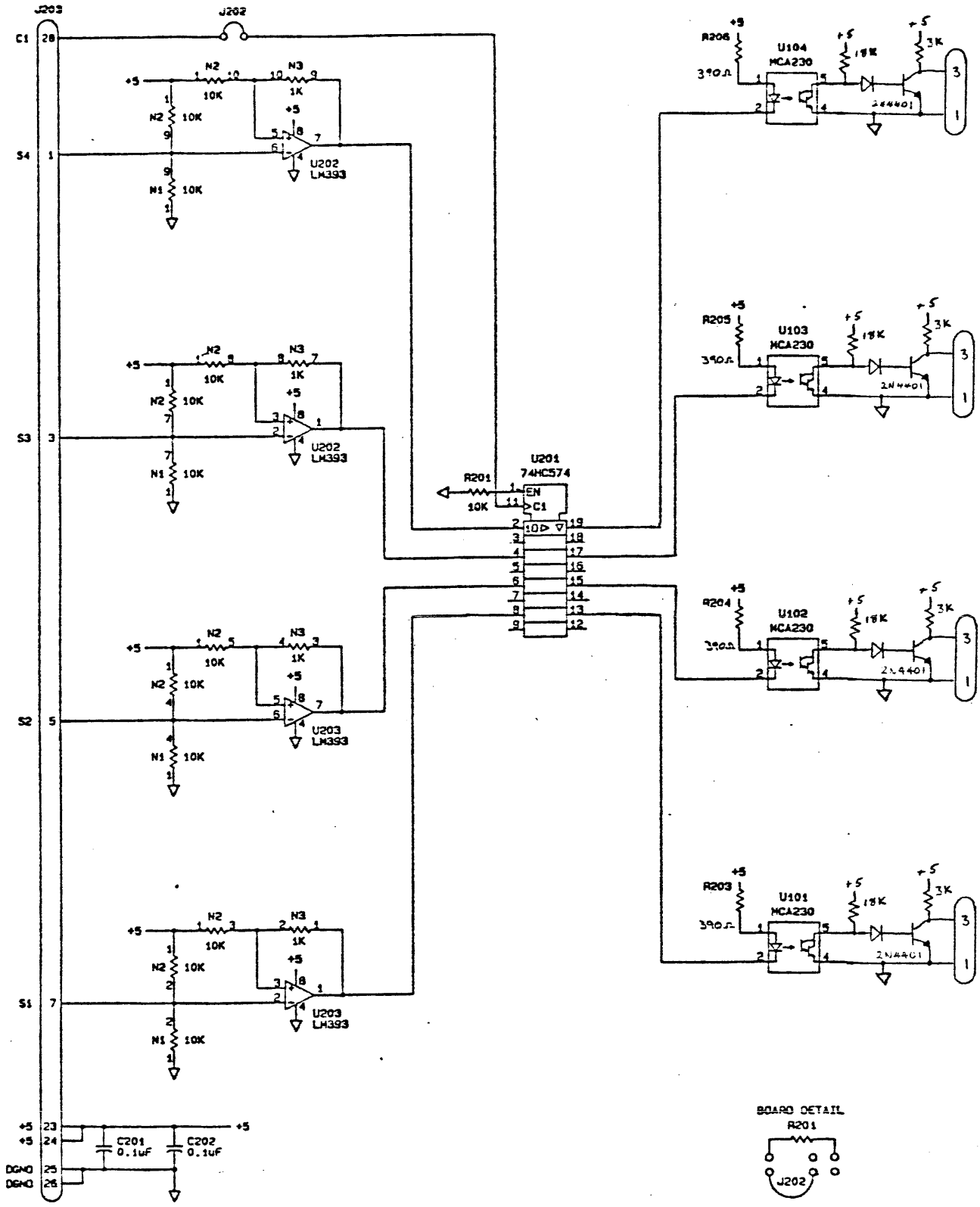
PRODUCT	NAME 7000-670-24-A	VERSION	PAGE
7000 MDAS	LATCHING RELAY D-OUTPUT BOARD	D1	1 OF 1



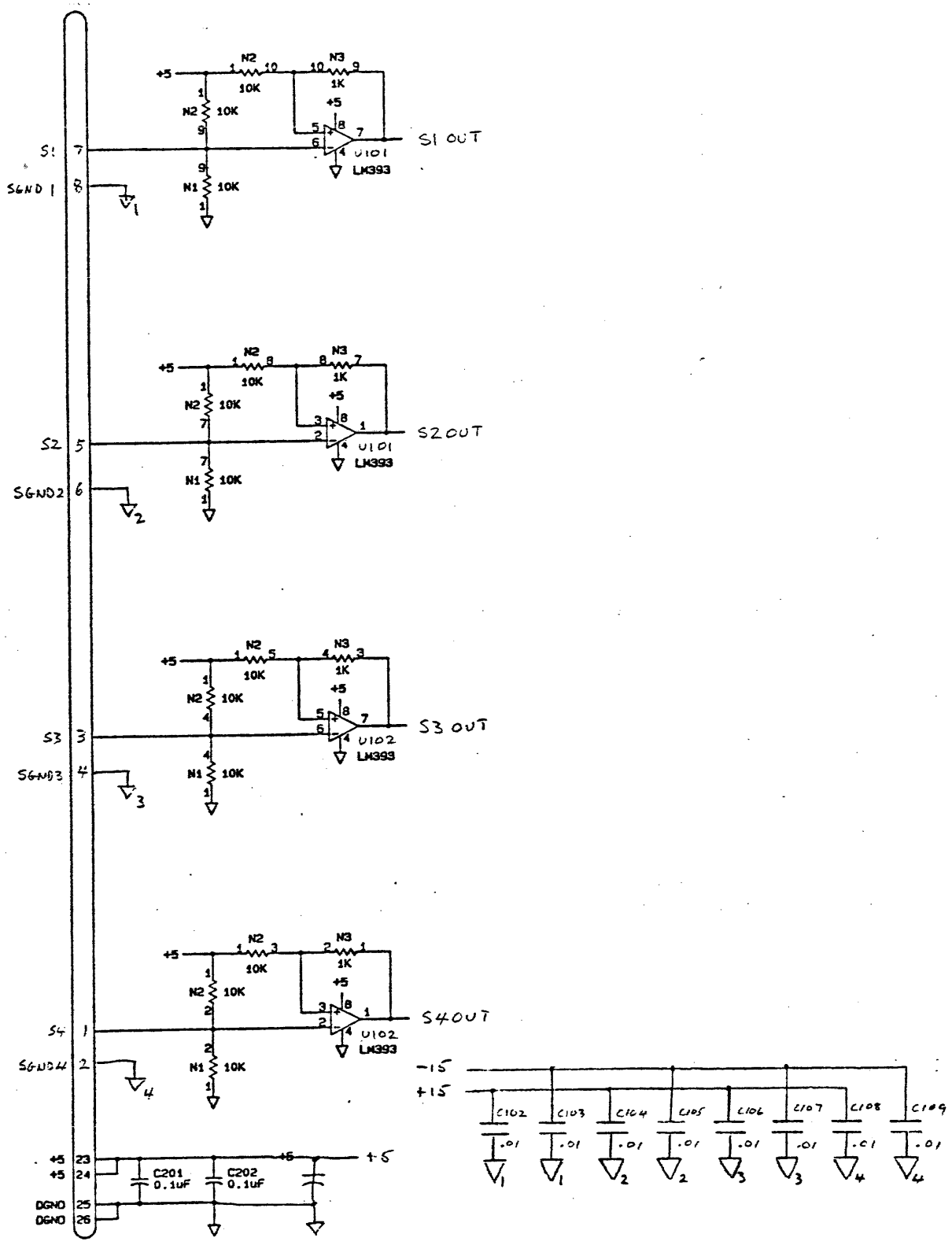
PRODUCT	NAME 7000-670-24-A	VERSION	PAGE
7000 MDAS	SOLID STATE AC RELAY D-OUT BD	D2	1 OF 1



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

MDAS 7000 Upgrade Information

September 2, 1987

MDAS Firmware Upgrades

The table below lists the minimum firmware version required by some of the newer I/O cards and the SM0 option.

I/O Card	Firmware
D3 AC/DC Relay	Ver. 2.4
F1 Anti-Aliasing Input	Ver. 2.6
R4 Optical Encoder	Ver. 2.3
R6 24 Bit TTL I/O	Ver. 2.5
R8 Frequency I/O ("B" or later)	Ver. 2.1
S3 Analog Output	Ver. 2.4
T3 Thermocouple ("B" or later)	Ver. 1.6
SM0 option (on-board TBASIC)	Ver. 3.0

The version number of the MDAS firmware is displayed on the LED panel when the MDAS powers up. If your current version number is lower than the version number indicated below for any card that you wish to use, you must have a firmware upgrade (Model SAPP) before your new card will function.

The version number of an I/O card is the last letter of the part number. The part number can be found on the back of the circuit board opposite the 30-pin connector to the MDAS backplane.

All firmware upgrades involve replacing EPROM's on the MDAS processor board. In addition, some firmware upgrades require hardware modifications:

1.x -> 4.x requires new converter (Model PCA), new processor (Model PCP), backplane fix, led display fix, power supply fix
2.x -> 4.x requires new processor (Model PCP), backplane fix, led display fix, power supply fix
3.x -> 4.x requires modification of the processor
4.x -> 4.x requires firmware EPROM upgrade only (Model SAPP)

Firmware Upgrade Kit Model Numbers	Price
Model SAPP (MDAS Firmware Upgrade Kit)	$100
Model PCA (MDAS Converter Card Upgrade Kit)	$400
Model PCP (MDAS Processor Card Upgrade Kit)	$400

When ordering either a SAPP, PCA, or PCP, be sure to include your current version number, the number of the version to which you wish to upgrade, and whether or not you have SM0. All firmware upgrades can be user installed. All prices based on return of old components. A new manual set is included with all firmware upgrades.

Memory Upgrades

To:	M512K	M1M	M2M	
From:				
M128K	$600	$1100	$2000	Memory upgrades can be user installed.
M256K	$450	$950	$1850	Prices based on return of old memory board.
M512K	----	$700	$1600	
M1M	----	----	$1100	

System Upgrade

7000-I Basic System to 7000-II Disk-based System $1500
Entire system must be returned to factory.

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 4.4 LANGUAGE ADDITIONS AND CHANGES

1. APPL 4.4 must be run with MNIX 3.2 to run correctly. MNIX 3.2 has the space available to store the power up parameters for PWRS.

2. New command PWRS was added. PWRS allows a user to specify the following output cards to an initial value on power up: (1) R6 (2) Regular Digital (3) D3 (4) S3, S4, S5 (5) S1. The command is specified as follows:
   PWRS <type> <chan> <value> [<type> <chan> <value>] [...];
   Where <type> is the type of output (1-5).

   In conjunction with the R6 card, please note that if more than one channel (of the possible three channels) is going to be used, the channels must be sequentially specified.

   On power up, these output channels are initially set and will be set in the order they are specified. The whole sequence must be specified in one command.

3. New command GPWS was added. GPWS allows a user to see what channels have been defined with the PWRS command. The <type> <chan> and <value> will printed out just as they were specified in the PWRS command. A NULL string will be sent to signify the end of the sequence.

MDAS Release Notes
Version: 4.3
Date: 1-Jan-88

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 4.3 LANGUAGE ADDITIONS AND CHANGES

1. New command STMO was added to stream data directly from hard disk or streaming tape to D/A channels. The syntax is:
   STMO <fname> <reps.offset> <num> <chan> [<chan>] [...];

2. The GW command to generate a sine wave was upgraded to use the precise phase instead of just shifting the points after the sine wave was generated. A problem with odd period numbers for buffer type 2 was fixed.

3. An addition was made to the RD command to read data into an ASCII buffer (type 6).

4. New command RCNT was added to count the period of a signal. It is used with an R8 card and the syntax is: RCNT <chan> <buf> [<#of periods>]; Valid buffer types include 2, 3, and 4. If buffer is a type 4 buffer, the period is stored; otherwise the number of 200e-9 periods is stored.

5. The AOAI command was upgraded to a maximum speed determined by: minperiod = (#analog channels) * 5.4e-6 + 15.0e-6. If a 16-bit analog board is used, change the 5.4e-6 to 25e-6.

6. The CLVL command was upgraded so one can specify channels A and B to set the count level for these channels (A and B). One can still specify channels G and F to set the levels for A and B.

7. A problem with transferring data while using resident TBASIC was fixed. When using resident TBASIC and calling M_GET with a fifo buffer an error was generated because the data types didn't match.

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 4.2 LANGUAGE ADDITIONS AND CHANGES

1. Changes originally made with 3.F have been included.

2. A problem when using the SPN (spawn processes) command was fixed. It was an erratic error that happened when interrupting a command that was being sent over the communication lines.

3. The problem with having more than 16 LABLs in an executable buffer creating a system hard error was fixed. It will only give a SYNTAX error when one tries to specify more than 16 LABLs in one executable buffer.

4. A problem with redirecting the OUTPUT DEVICE to a type 6 buffer and doing a DIR was fixed.

5. Integer math routines ADDI, SUBI, MULI, DIVI, and SCAI were upgraded to be used with long word integer buffers (type 3). The routines will work with type 2 (integer) and type 3 (long) buffers.

6. IFBD was upgraded to allow for an optional bits parameter. The command now looks like IFBD <cons> <op> <buff> [<bits>];. The bits parameter specifies which bits to use in the comparison. Only the bits in <bits> will be used in the comparison (ie the contents of <buff> is ANDed with the number in <bits> and that result is used to compare against <cons>).

7. IFBA was upgraded to allow buffer types 1 and 3. IFBA can now be used with buffer types 1, 2, 3, and 4.

8. New command BIT was added to perform bit operations on a buffer. BIT has the following parameters: BIT <bufA> <bufX> <type> <const>. Where <bufA> is the input buffer and <bufX> is the output buffer. The <type> parameter specifies the type of bit operation to perform and is given by:
   0 - AND
   1 - OR
   2 - EOR
   3 - SHIFT RIGHT
   4 - SHIFT LEFT
   The <const> parameter specifies the bits or number of bits to be used in the operation.

9. New command IFB was added to do comparisons on buffers. IFB format: IFB <bufA> <op> <bufB> [<pts>];. IFB takes the average of <bufA> and compares it with the average of <bufB> and set the condition flag accordingly. If <pts> is not specified it defaults to a value of 1. IFB will work on buffers types 1, 2, 3, and 4 and <bufA> and <bufB> do not have to be of the same type.

10. When using full bridges (B1S2 card), the results were turning out negative; the sign problem has been fixed.

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 4.1 LANGUAGE ADDITIONS AND CHANGES

1. Changes originally made with 3.E have been included.

2. A problem with the SCAN command when using a floating point buffer was fixed.

3. Due to the inherent limitations of the SCSI port, a check on STRM was added because the maximum number of samples that can be written to the hard disk at one time is 16,776,960.

4. A problem with the DOSB command was fixed. This timing was not functioning correctly and the command would never finish.

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 4.0 LANGUAGE ADDITIONS AND CHANGES

1. The commands are the same as those found in version 3.B except that the type of EPROM hardware changed from a 256K EPROM to a 512K EPROM. Version 4.0 includes the changes of versions 3.C and 3.D except that those commands that were deleted in 3.C and 3.D are included.

2. Changes made with 3.E will be added at a later date.

3. Changes made with 3.F will be added at a later date.

MDAS Release Notes
Version: 3.F
Date: 21-Oct-87

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 3.F LANGUAGE ADDITIONS AND CHANGES

1. A problem with transferring data while using resident TBASIC was fixed. When using resident TBASIC and calling M_GET with a fifo buffer an error was generated because the data types didn't match.

2. A problem when using SPN (Spawn Processes) was found. It was an erratic error that happend when interrupting a command that was being sent over the communication lines.

3. The problem with having more than 16 LABLs in an executable buffer creating a system hard error was fixed. It will only give a SYNTAX error when one tries to specify more than 16 LABLs in one executable buffer.

4. A problem with redirecting the OUTPUT DEVICE to a type six buffer and doing a DIR was fixed.

5. Integer math routines ADDI, SUBI, MULI, DIVI, and SCAI were upgraded to be used with long word integer buffers (type 3). The routines will work with type 2 (integer) and type 3 (long) buffers.

6. IFBD was upgraded to allow for an optional bits parameter. The command now looks like IFBD <cons> <op> <buff> [<bits>];. The bits parameter specifies which bits to use in the comparison. Only the bits in <bits> will be used in the comparison (ie the contents of <buff> is ANDed with the number in <bits> and that result is used to compare against <cons>).

7. IFBA was upgraded to allow buffer types 1 and 3. IFBA can now be used with buffer types 1, 2, 3, and 4.

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 3.E LANGUAGE ADDITIONS AND CHANGES

1. A problem with the SCAN command when using a floating point buffer was fixed.

2. Due to the inherent limitations of the SCSI port, a check on STRM was added because the maximum number of samples that can be written to the hard disk at one time is 16,776,960.

3. A problem with the DOSB command was fixed. The timing was not functioning correctly and the command would never finish.

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 3.D LANGUAGE ADDITIONS AND CHANGES

1. MDAS commands SCAN and STRM were changed to include up to 128 channels (changed from 64) of input.

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 3.C LANGUAGE ADDITIONS AND CHANGES

1. The following commands removed from the MDAS command set: CI, AOAI, DFF, DOF. They were taken out to make room for the upgraded version of the streaming tape command STRM. STRM will now take data in background and write it to tape in foreground until a control-C is sent to the command port.

2. SPN will function endlessly if the <count> parameter is specified as 0.

3. New command SPNI was added. SPNI allows for interrupts from RS232 to occur while a SPN process is running. It is reset on control-C.

4. New command IFC was added. IFC will verify if a character has been sent to a MDAS input PORT. IFC sets the condition flag.

5. New command RDC was added. RDC will read the characters sent to a MDAS input PORT.

6. For analog cards using the AIF command, the sample and hold circuitry is enabled at the beginning of the command to allow all the channels in the sequence to be sampled at effectively the same time.

7. CPY was modified to allow one to copy a regular buffer into a fifo buffer.

8. GT was modified to allow one to get the time into a buffer. The time is written as 8 bytes in the format: sec, min, hr, mday, mon, yr, wday, 1/100.

MDAS Release Notes
Version: 3.B
Date: 19-Aug-87

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 3.B LANGUAGE ADDITIONS AND CHANGES

1. New commands ADDI, SUBI, MULI, DIVI, and SCAI were added to perform integer math on word integer buffers (type 2). They are much faster than the floating point math routines but have the inherent limitations of doing integer math.

2. New command WRFM was added. WRFM will allow the user to specify the number of characters that are displayed on an output that uses floating point notation. This allows for faster output of floating point data, but less number of significant digits.

3. New command MVHD was added. MVHD allows for the user to move the heads of a hard disk to a shipping zone. The heads will move back to track 0 when power is reapplied to the system. The system power may have to be be applied two times before the hard disk is recognized.

4. An append flag has been added to the command WRD for the streaming tape unit. This allows the user to append a file at the end of the tape. The format is: WRD <bufA> <E:filename> [<points> [<offset> [<appendflag>]]].

5. When using commands RUN or SPN an error now generates the printing of the command that caused the error instead of "RUN" or "SPN". This is useful when debugging executable buffers.

6. The error "SYSTEM HARD" now prints the command that caused the system hard error. Example "AI: SYSTEM HARD"

7. The position of the parameter that caused an error in an MDAS command should now be displayed on the LEDs.

8. AO and AOS were rewritten to function more efficiently and to reduce code requirements.

9. RDP (Read Pairs) was taken out to make room for the new commands.

MDAS Release Notes
Version: 3.A
Date: 10-Aug-87

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 3.A LANGUAGE ADDITIONS AND CHANGES

1. New command TPRW was added to rewind a cassette tape in conjunction with the streaming tape drive. The syntax is: TPRW [<iflag>]. When the argument of <iflag> is "0", the tape will rewind only after the end of the tape is encountered. When the argument of <iflag> is "1", the tape will rewind immediately after the TPRW command is enacted.

2. New command STST was added to do a test of the system EPROMs. Subsequently, GC (Get Condition) will verify operational EPROMs by displaying a "1" or "0" if the EPROMs are defective.

3. The SFRQ command was modified to allow specification of the cut-off frequency instead of the clock frequency. Specification of the whole frequency range is necessary if the jumper is set on the F1 board.

4. For improved accurate calibration, the CALB command has increased the delay between switching in the relays.

5. The RES command will now redefine all channels back to type 0 with a gain of 1. For further info, refer to the DFNC command.

MDAS Release Notes
Version: 3.9
Date: 12-May-87

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 3.9 LANGUAGE ADDITIONS AND CHANGES

1. APPL version 3.9 must run with MNIX 2.9 or higher because it has a check sum at the end of the ROMs that is verified on power-up. The jump table for resident TBASIC has been moved down 4 bytes to accomodate for the check sum, therefore, resident TBASIC must be a version manufactured after 12-May-87.

2. All type 1 character buffers will be printed out in unsigned format.

3. New command GNER was added. GNER returns the gain error of each of the 8 programmable gains. Syntax: GNER <gain>; where <gain> is 1 ,2, 4, 8, 16, 32, 64, or 128.

4. An interleave factor can be added to RDD for a tape. This allows read out of data from one channel at a time. Syntax:
RDD <buf> <e:filename> [<points> [<offset> [<interleave>]]];

MDAS Release Notes
Version: 3.8A
Date: 04-May-87

TransEra Corporation
3707 North Canyon Road
Provo, UT 84604

MDAS APPL RELEASE 3.8 LANGUAGE ADDITIONS AND CHANGES

1. The GA (Get Analog) command will perform job auto ranging more efficiently.

2. The GW command will not limit the waveform to + or - 10 volts.

3. In conjunction with the R8 card, MIX & MIXG commands will not reset the 9513 counter on type 5.

4. Commands RDCH, RDRG & RDST may store results in an optional buffer (example: RDCH <chan> [<buf>];).

5. Fix (on an odd number of characters) in setting up the arguments for resident TBASIC completed.

6. Minor change completed in conjunction to the method of calculation of bridge card command CALB.