DX/256x256
DIGITAL
MATRIX
with IF-12 Control Module

This information is proprietary to CYTEC Corp., and is not to be used, caused to be used, reproduced, published or otherwise used/or disclosed in any way that might be detrimental or compromising to CYTEC Corp.

Table of Contents

1.0 ADDENDUM 6
2.0 GETTING STARTED 7
3.0 GENERAL 8
3.1 CHASSIS DESCRIPTION 9
3.2 SPECIFICATIONS 10
3.3 POWER SUPPLY 11
3.4 FRONT PANEL 12
4.0 INPUT and OUTPUT BUFFER MODULES 13
5.0 IF-12 GPIB / RS232 / LAN CONTROL MODULE 16
5.1 LAN INTERFACE 16
5.2 RS232 INTERFACE 17
5.3 IEEE488 INTERFACE 19
5.3.1 IEEE488.2 SPECIFIC MATRIX COMMANDS 19
5.4 CONFIGURING TCP/IP PARAMETERS FROM A SERIAL CONNECTION 20
5.5 RUNNING THE CYTEC FACTORY APPLICATION ON THE IF12 22
5.6 COMMAND FORMAT/COMPLETION 22
5.6.1 END OF LINE CHARACTER (EOL) 23
5.7 SETUP COMMANDS 24
5.8 SWITCH COMMANDS 25
5.9 STATUS AND INTERROGATE COMMANDS 27
5.10 I – INTERROGATE COMMAND 27
5.11 OTHER COMMANDS 28
5.12 INPUT / OUTPUT vs MODULE / SWITCH NOMENCLATURE 29
5.13 LIST MANAGEMENT 30
5.14 MATRIX COMMAND SUMMARY 30
5.15 IF-12 (RS232/LAN/GPIB) DEFAULT CONFIGURATION SETTINGS 31
5.16 LCD DISPLAY/KEYPAD MANUAL CONTROL OPTION 31
5.17 MEMORY SANITATION PROCEDURE 33
APPENDIX - EXAMPLE PROGRAMS 34
Java LAN Programming Example: 34
C LAN Programming Example 36
LabWindows RS232 Programming Example 39
LabWindows GPIB Programming Example 45
LabView Drivers 52

DRAWINGS

DRWG #	DESCRIPTION
11-21-50	IF-12 LAN/RS232/GPIB Control Module Schematic
17-08-50	64 Channel RS422 H68RT Output Buffer Schematic
17-09-50	64 Channel RS422 H68RT Input Buffer Schematic
17-12-50	64 Channel RS232 H68RT Output Buffer Schematic
17-13-50	64 Channel RS232 H68RT Input Buffer Schematic
17-16-51	64 Channel RS422 D78S Output Buffer Schematic
17-17-51	64 Channel RS422 D78S Input Buffer Schematic
48-06-22	DX/256x256 Front Panel
48-06-50	5V To 3.3V Voltage Regulator Board
48-07-00	DX/256x256 Redundant Power Supply Option
48-12-50	DX256x256 Schematic
48-34-00	DX/256x256 Chassis Overview
48-34-21	DX/256x256 Rear Panel
48-34-30	DX/256x256 Standard Data/Control Wiring Diagram
6260-01-50	64 Channel 75 Ohm TTL D37S Input Buffer Schematic
6260-02-50	32 Channel 75 Ohm TTL D37S Output Buffer Schematic
99-23-50	MC-2 LCD Keypad Display Assembly Schematic
99-76-54	9V-36V IN +5V/-5V/+3.3V OUT DC/DC Converter Schematic

1.0 ADDENDUM

This page is intentionally left blank

2.0 GETTING STARTED

Unpack the unit and make sure it has arrived undamaged. Inspect for dents, bent handles, major scratches and missing or loose parts. Note that many of the items listed individually on the packing list are already installed within the chassis, rather than being packed separately.

Compare the Shipped Configuration List on the last page of the Quick Start Guide that shipped with the unit with the included packing slip to verify that all components and ordered parts have been received. If any purchased items are missing please contact your Sales Representative at 585-381-4740 or sales@cytec-ate.com. Utilize the Shipped Configuration List to identify which drawings and diagrams refer to the specific unit ordered.

Next, set up the chassis on either a bench or rack. The front handles allow the unit to be bolted to a standard 19 inch rack. No special setup tools are needed.

For AC powered units, a Power Cord should be included in the box. Plug one end into the chassis and the other into a three prong commercial AC outlet. The unit will operate from one of two AC voltage ranges: 100/140 or 200/260. There is a fuse holder built into the AC Input Module. The switching power supply on some units is auto-ranging.

Install the appropriate remote control cable to the controlling computer: RS232, IEEE488 (GPIB) or Ethernet. Cytec provides a one to one RS232 D9 cable but does not provide Ethernet or GPIB cables with the unit.

Turn the unit ON via the toggle switch located on the AC Input Module. The front panel Power LED should illuminate.

Study the sections of this manual which deal with your control interface (RS232, GPIB or Ethernet), as well as the controlling command syntax. A group of programming examples are included in appendices at the end of the manual and provide a good structure to work from. Example driver programs may be included on a disc if requested. Drivers can also be downloaded from Cytec's web site at: http://www.cytec-ate.com/downloads

You should now be able to begin writing useful code. **Always write and debug code thoroughly before hooking up live signals to the matrix!** This equipment gives you full control over what is switched to where and will not prevent you from making potentially harmful connections. That is, nothing in the system prevents the switching of excessive voltage, current, or power, which can damage or destroy the relay contacts or digital switches.

3.0 GENERAL

The DX/256x256 Digital Matrix is a solid state switch matrix designed to switch high data rate differential digital signals. The basic system is intended to switch TTL or CMOS signal levels in a number of different possible configurations. These configurations usually equal an NxM (N by M) matrix where N + M add up to 512 total input and outputs. The heart of the system is a 512 port switch module **(Drwg. #48-12-50)** containing an Altera Cyclone 2 FPGA configured as a Digital Crosspoint Switch. When used without buffer modules, the system conforms to all TTL/LVTTL specifications.

Input and Output configurations occur in 64 channel increments. The system is completely modular by adding the desired number of Input and Output Buffer Modules. All configurations are nonblocking and full fan-out. (These terms are defined immediately below.)

In a switching system, "nonblocking" is defined as allowing any input to be connected to any output without disturbing the connectivity of previously established paths. The term "full fan-out" implies that any one input may be connected to any or all outputs without attenuating or diminishing the quality of the signals. The system will not mix or combine inputs so you cannot mix or connect multiple inputs to one output.

This system incorporates an advanced solid state switch fabric that helps restore signal levels, corrects deterministic jitter and cleans up rise time problems associated with wiring and cable runs. Input and output transition modules are also available that perform signal conversions in applications that do not require switching.

3.1 CHASSIS DESCRIPTION

The DX/256x256 **(Drwg. #48-34-00)** is a single, stand-alone chassis which may be controlled either remotely via computer or locally via the optional keypad manual control. All chassis are standard 19" rack mounting, 20" deep and 7" high. An example rear panel view of the DX/256x256 is shown in **Drwg. #48-34-21.**

The chassis' power supply operates from the AC line supply via a fused line cord adapter on the rear panel. This adapter also houses the ON/OFF switch.

Refer to the "Shipped Configuration" sheet to determine which of the chassis assembly drawings pertain to the purchased system. This assembly drawings will show the locations of the various modules.

Each DX/256x256 Chassis Contains*:*

- (One) High Density 256x256 Switch Fabric Module **(See Drwg. #48-12-50)**
- (Up to four) Output Buffer Modules (**Section 4.0)**
- (Up to four) Input Buffer Module **(Section 4.0)**
- (One) Control Module **(Section 5.0)**
- (One) LS100-5 Power Supply **(Section 3.3)**
- (One) Universal Voltage Adaptor Module **(Section 3.3)**
- (One) Front Panel Chassis Fan
- (One) Internal Fan
- (Optional) LCD/Keypad Manual Control **(Section 5.9)**
- (Optional) Redundant Power Supply **(Section 3.3.1)**

When used with input and/or output buffers, the specifications for the entire system are determined by the buffer stages.

Input Voltage High (V_{IH})	2.1V Min	5.25V Max
Input Voltage Low (V_{IL})	-0.3V Min	0.8V Max
Output Voltage High (V_{OH})	2.4V Min	3.7V Max
Output Voltage Low (V_{OL})	-------	0.4V Max
Data Rate		150Mb/s Max
Output Current		80mA Max

Buffer modules are optionally available to allow for either 75 ohm TTL, RS-422 or RS-232 inputs and/ or outputs. See **Section 4.0** for more information.

3.2 SPECIFICATIONS

Dimensions:	19" Rack Mounting x 7" High x 20" Deep
Weight:	Maximum weight (with full complement of modules and redundant power supply) less than 45 lbs.
Power:	Less than 160 W @ 100-130 VAC or@ 200-260 VAC
Environment:	
Operating:	0˚C to 50˚C @ 95% Relative Humidity
Storage:	-25˚C to 65˚C @ 95% Relative Humidity
Capacity:	Up to 256x256 size (512 cross points) matrix configuration
Display:	One Power LED Indicator (if no manual control and no redundant power supply); No Power LED Indicator (if manual control is included and no redundant power supply); Two Power LED Indicators (if redundant power supply is included)
Control Mode:	IEEE488, RS232 and Ethernet TCP/IP standard; LCD/Keypad as an option
Signal Connections:	Specific to transition modules
Switch Fabric:	Altera Cyclone 2 FPGA configured as a Digital Crosspoint Switch

3.3 POWER SUPPLY

The standard DX/256x256 chassis is built with an LS100-5 power supply, which has a +5 Volt, 16 Amp output. **Drwg. #48-34-30** shows the power/control wiring diagram. The supply output is routed to the Universal Voltage Adapter Module (**Drwg. #48-06-50**), which converts the 5V supply to a 3.3V source for the Cyclone 2 FPGA while the +5V output is used to power the fans, the control module, the DX/256x256 switch module **(Drwg. #48-12-50)** and the input and output buffer modules **(See Section 4.0)**. Two separate power supplies may be included if the system is built with the redundant supply option **(See Section 3.3.1)**.

The standard configuration using the Voltage Adapter Module with one power supply is sufficient for RS-232 input/output buffers **(Drwg. # 17-13-10 and 17-12-10)** or RS-422 input/output buffers **(Drwg. # 17-17-11 and 17-16-11)** however for 75 Ohm TTL applications **(Drwg. # 6260-01-10 and 6260-02-10)** the system requires two LS100-12 power supplies which have +12 Volt, 8 Amp outputs. In this case the power/control wiring diagram is shown in **Drwg. # 99-76-54** and the adapter module converts the 12V supply to the required 5V and 3.3V sources.

The supply is wired out to the Selectable AC Input Module on the rear panel, which also holds the chassis's ON/OFF Switch. The user can select one of two AC voltage ranges: 110/120 Volts or 220/240 volts AC. To change the selected voltage, remove the fuse cartridge using a small blade screw driver or a similar tool. Select the desired voltage by matching the arrow on the fuse cartridge to the arrow located on the Input Module's lower right corner. *Replace the fuse cartridge making sure the voltage selection arrow aligns with the arrow located on the Input Module*.

Two fuses are held in the fuse cartridge, with 220/240 VAC fused separately from 110/120 VAC. See the "Shipped Configuration" section for fuse sizes.

3.3.1 REDUNDANT POWER SUPPLY OPTION

This option allows the system to continue to operate, when one of the two power supplies fails. See **Drwg. #48-07-00**. Front panel LEDs show the power supply operating status **(See Section 3.4)**. **An unlit LED is the only indication you will receive that a supply has failed! LEDs should be checked on a regular basis!**

Both supplies are wired through a Diode Separation Module that isolates the supplies from each other during normal use and allows a single supply to operate the system. In the case where a redundant 5V power supply is used as a backup for the primary 5V supply, protection diodes are required to protect against any short circuiting that may arise from minor voltage differences between the two supplies. A Diode Separation Module (utilizing the 18-02-10 voltage regulator board) is used for this purpose. The two 5V supplies (trimmed up to 5.5V to compensate for the diode drop) enter the board separately. Each passes through a high current power diode, and then both are tied together at the output. This results in a single, 5V source to drive the control logic and any buffer modules. It also feeds the 3.3V voltage regulator board described above. With this setup, if either of the 5V power supplies fails for any reason, the remaining power supply will be able to drive all the necessary circuitry.

3.4 FRONT PANEL

Three different Mainframe Front Panels are supplied, depending on whether the unit is built with or without the optional Manual Controls or the optional Redundant Power Supplies features.

If the optional LCD/Keypad Manual Control is NOT included and the system does NOT have Redundant Power Supply, the chassis will have a single Power LED. See **Drwg. #48-06-22.**

If the DX/256x256 chassis HAS the optional LCD/Keypad Manual Control but NOT the Redundant Power Supplies option, the Front Panel has no Power LED. Also see **Drwg. #48-06-22.**

If the system HAS Redundant Power Supply the Front Panel will have two Power LED Indicators. These LEDs are labeled “5V POWER - A” and “5V POWER - B”, and they show the power supplies’ operating status. See Drwg. **#48-07-00.**

The front panel of every DX/256x256 also includes a +5V fan that cools the unit.

4.0 INPUT and OUTPUT BUFFER MODULES

DX/256x256 Matrices that switch RS-422 or RS-232 level signals may require the use of Input and/or Output Buffer Modules. Buffer Modules convert the input and output signals to levels that can be safely handled by the Solid State Crosspoint IC at the heart of the matrix.

Each Buffer Module supplies 64 channels, therefore a fully buffered 256x256 system has four Input Buffers and four Output Buffers. When Buffers are required, system specifications are determined by the Buffer Modules themselves.

RS-422 Input and Output Modules have two connector options: 68 pin male headers **(See Drwg. #17-08-50 and Drwg. #17-09-50)** or 78 pin female D connectors **(See Drwg. #17-16-51 and Drwg. #17-17-51-1)**.

RS-422 Input Buffer Module (Drwg. #17-09-50 and Drwg. #17-17-51-1)

Common Mode Voltage (V_{CM})	$-7V<V_{CM}<+7V$
Differential Voltage Swing	200mV Min
Input Resistance	6.8kΩ Typical
Data Rate	20Mb/s Max

RS-422 Output Buffer Module (Drwg. #17-08-50 and Drwg. #17-16-51)

Common Mode Output Voltage	1.8V Typical	
Output Voltage High (V_{OH})	2.5V Min	3.4V Typical
Output Voltage Low (V_{OL})	0.3V Typical	0.5V Max
Data Rate	20Mb/s Max	
Output Current	150mA Max	

RS-232 Input Buffer Module (Drwg. #17-13-50)

Input Voltage Range	-25V Min	+25V Max
Input Voltage High (V_{IH})	2.17V Min	
Input Voltage Low (V_{IL})	1.06V Max	
Data Rate	200kb/s Typical	
Input Resistance	5.0kΩ Typical	

RS-232 Output Buffer Module (Drwg. #17-12-50)

Output Voltage High (V_{OH})	+5V Min	+8V Typical
Output Voltage Low (V_{OL})	-8V Typical	-5V Max
Data Rate	120kb/s Typical	
Output Current	±10mA Typical	

If the user wishes to switch normal TTL signals, the system will be furnished with the DX/64-TTL to TTL Thru Module. This module serves to only to provide a rear panel connection to the internal switch module **(Drwg. #48-05-50).** If this board is included in the system it will be referred to as **Drwg. #17-12-11**.

75 ohm TTL Input and Output Buffer Modules

DX/64-TTL-75-IN-D37S 75 Ohm TTL Input Buffers (Board # 6260-01-10)

These modules are designed to accept 64 standard TTL signals of 0 to 5 Volts and are terminated in 75 ohms. These modules are shown in **Schematic # 6260-01-50** pages 1 through 4. The 75 ohm terminating resistors are rated to withstand +15 Volts on the input as a maximum signal level. The modules have four D37F female connectors with 16 channels each. The pin out is shown on the schematic.

Input buffer specs:	
TTL Low	0.8 V Max
TTL High	2.0 V Min
Maximum Input without damage	+15 V not continuous for long duration. +8.5 V continuous long term. -15 V not continuous for long duration.

DX/32-TTL-75-OUT-D37S 75 Ohm TTL Output Buffer Modules (Board # 6260-02-10-1)

There modules are designed to output standard TTL levels of 0 to 5 volts. A schematic of the output buffers is shown in **Drawing # 6260-02-50-1** pages 1 through 4. The output buffers are capable of driving 32 TTL levels into a 75 ohm load over 200 feet of typical 75 ohm coaxial cable.

There are two D37F Female connectors on each module for a total of 32 outputs per module.

The output amps are short circuit protected by a 75 ohm series load on

the output. Output Buffer Specifications:

	At Patch Panel	After 200' Cable
TTL High	+ 4.8V*	+ 4.4V*
TTL Low	200 mV*	500 mV*
Maximum output current	133 mA	133 mA
Maximum output voltage	5.2 V*	5.2 V*
Maximum Output frequency	40 MHz	35 MHz
Rise time into 75 ohm load	2.65 nS	7.0 nS

* 75 Ohm load to ground

5.0 IF-12 GPIB / RS232 / LAN CONTROL MODULE

Introduction

CYTEC's IF-12 RS232/LAN Control Module is designed to control single chassis mainframes. Three forms of remote control are available on the module: IEEE488 (GPIB), RS232 and Ethernet LAN. An optional manual control is also available. All four interfaces may be active and used simultaneously.

Interface options: GPIB, RS232 and LAN are standard, but the Manual Control must be specified when purchasing the system. On some systems where panel space is limited, only two of the three interface connectors may be included.

5.1 LAN INTERFACE

Dynamic IP Address (DHCP): The Cytec IF12 is set at the factory to attempt to obtain an address from a DHCP server when the application boots. If you are connected to a network with a DHCP server, then the device IP address, network mask and gateway should be configured automatically. If your PC is on the same DHCP network, you will be able to communicate with the device after a short boot period of less than 10 seconds.

Static IP Address: If the module is plugged in to a network that does not have a DHCP server, you must provide a static IP address, network mask and gateway. These addresses should be provided by your network administrator.

Auto IP Address: The factory application contains an auto IP negotiation system. This allows the device to automatically configure its address in the absence of a central DHCP server, and without the need for a static IP address. This scheme is utilized as a fallback that will activate when both dynamic and static IP addresses fail to initialize. In order to communicate with a device in auto IP mode, the host system must support auto IP. Auto IP support is included in both Windows and OS X operating systems. By default, auto IP addressing starts in the 169.XXX.XXX.XXX address range.

Find Your Device: Our recommended option to locate the device is to use a local discover utility. You can do this by navigating to the Cytec web site and downloading the tool localdiscover.exe from https://cytec-ate.com/discover-cytec-local. The executable sends out a request to all Cytec devices on the local network. It opens a browser page on the first device to respond that lists all of the discovered devices, or a page that show that no devices were found.

Note: If these options are failing, there may be a firewall issue blocking the applications from sending the UDP broadcast that is used to locate Cytec devices. Always grant Cytec applications the ability to get through your OS firewall and ensure that UDP port 20034 is open for use.

5.2 RS232 INTERFACE

Signal Connections
The control module is pre-configured at the factory to operate as Data Communications Equipment (DCE) per the EIA RS232D Standard. In this configuration, the module transmits on the RxD Pin and receives on the TxD Pin. RTS is required to be high for the control module to transmit and CTS is output high by the control module to indicate a ready for data state and low when busy. The RS232 rear panel connector is a D9P (male) and can be run directly from a D9 computer COM port with a straight through (one to one) D9S to D9S cable. A null modem cable will not work with the factory default settings! Adaptors are available at any computer store to convert from D25 to D9. Do not use any adaptor that also acts as a null modem converter. If you are building your own cables, consult CYTEC Corp., for D25 to D9 pin out conversion.

D9P (male) PIN OUTS

Pin	Signal	Function
1	DCD	Not Used.
2	RxD	Data out of Control Module.
3	TxD	Data in to Control Module.
4	DTR	Not Used
5	Common	Signal Ground.
6	DSR	Not Used
7	RTS	Control Module requires + V to transmit.
8	CTS	Control Module provides +V when ready
9	RI	Not Used

Find Your Device

Open Device Manager on Windows computers and navigate to Ports. The COM port number will be bracketed next to the device description.

Configure Your Device

The RS232 interface can be accessed using any standard terminal emulation program such as PuTTY which can be downloaded from putty.org. Enter the COM port number in the field for the Serial line to connect to. The default values set at the factory are:

- Speed(baud): 9600
- Data bits: 8
- Stop bits: 1
- Parity: None
- Flow control: RTS/CTS (Hardware)

The first thing you should do is turn on Echo. This will enable you to see what you are typing. Make sure you turn Echo back off when you are done with the terminal session. Echo being left on will normally interfere with programs written specifically to control the switch.

Echo
Echos the characters back to your screen while you type them so you can see what you type.

Command:	“E 0 73”	Turns Echo Off
	“E 1 73”	Turns Echo On

Answerback
Answerback allows the Control Module to return information to the COM port. Answerback should almost always be left on. If Answerback is enabled, the Answerback byte **must** be read back by the requesting device. Failure to do so could have unpredictable results.

Command:	“A 0 73”	Turns Answerback Off
	“A 1 73”	Turns Answerback On

Verbose
Verbose causes the system to return more specific information when you request status or read answerback characters. It is sometimes helpful when troubleshooting but it slows the interface down a lot. While there may occasionally be a good reason to turn on Verbose during a puTTY or Hyperterm session, it is almost never used in a programmatic interface. All of the same information can be generated in code based on the non-verbose responses without slowing down the RS232 interface.

Command:	“V 0 73”	Turns Verbose Off
	“V 1 73”	Turns Verbose On

Baud Rate
Baud rate is set at the factory at 9600 Baud. Change is under software control and the control module must be connected to a serial interface to effect the change.

Baud	Baud# n
2400	4
4800	5
9600	6
19200	7
38400	8
57600	9
115200	10
230400	11 (untested, you should consider LAN)
460800	12 (untested, you should consider LAN)

Command:	"P19 n 73"	
	"P19 7 73"	sets baud rate to 19200.

If the Baud rate is inadvertently set to an unknown rate, the default value may be restored. See the section on Setting Defaults for the procedure.

Obviously as soon as you reset the Cytec baud rate you will no longer be able to communicate with the switch until you reset the baud rate on your controlling computer or communication device.

CTS/RTS Handshake

The Clear to Send (CTS) and Request To Send (RTS) hardware handshaking functions may be modified by the 'P6' command.

Command:"P6 handshake 73"

handshake = 0	Handshaking off
handshake = 1	Handshaking on (default)

Example

"P6 0 73"	Turn handshaking off.

5.3 IEEE488 INTERFACE

Also known as GPIB (General Purpose Interface Bus), IEEE-488 is the international standard for a parallel interface used for attaching sensors and programmable instruments to a computer. When connecting IEEE-488 cables, some rules apply. The total number of devices should be 15 or less. The total length of all cables should not exceed 2 meters multiplied by the number of connected devices, up to a maximum of 20 meters. And no more than three connectors should be stacked together.

Find Your Device

Our recommended option to locate the device is to use NI Measurement & Automation Explorer (NI MAX), which can be downloaded from their website. Search for instruments in the application and the Cytec device should be found at default GPIB address 7.

Configure Your Device

GPIB Address:

Command syntax: “P14 n 73”.

For example, “P14 8 73” sets the GPIB address to 8.

5.3.1 IEEE488.2 SPECIFIC MATRIX COMMANDS

These commands are ignored by the RS232 interface.

***IDN? - Revision Number** (Same as Cytec “N” - Revision Command)

Syntax: “*IDN?”

The 'IDN?' command will cause the matrix to return its current revision number followed

by an end of line.

Send:	"*idn?"	Request revision number.
Receive:	"Cytec VDX/32x32 11-01-13 1.0" eol	Text string indicating rev.

***RST - Reset** (same as C - Clear command)

The '*RST' command will clear (open all switches) in the matrix.

Send:	"*rst"	Reset.
Receive:	"0" eol	Returns ‘0'.

5.4 CONFIGURING TCP/IP PARAMETERS FROM A SERIAL CONNECTION

To change parameters you will need to access the serial interface using any standard terminal emulation program from the COM port on your computer. Once you have established a serial connection the following commands can be used for configuration:

D command returns a list of current settings:

A1, E1, V0 Answerback = ON, Echo = ON, Verbose = OFF
Baudnumber = 6, RS Handshaking = 1
IP Address = 10.0.0.144
Netmask = 255.255.255.0
Gateway = 0.0.0.0
Port0 = 8080, Port1 = 8081
TCP idle = 60
Telnetlock = 0, Telnet Echo = 0
Battery Ram = 0, Default List = 0

IFConfig command is used to set the static IP address. The syntax for this command is:

ifconfig aaa.aaa.aaa.aaa nnn.nnn.nnn.nnn

a = ip address in dotted decimal format n = subnet mask in dotted decimal format

Example: ifconfig 10.0.0.100 255.0.0.0

Typing ifconfig and hitting the enter key will return the current settings.

Since you may be connected via Telnet to do this, **the IP address will not actually change until you reboot the Cytec switch**. This helps prevent anyone from mistakenly setting the IP to an

unknown address by accident. It is a good idea to double check the settings with the D command before you reboot.

HOSTS command sets the gateway for TCP/IP sockets. The syntax for this command is:

HOSTS xxx.xxx.xxx.xxx

Example: hosts 10.0.0.100

Typing hosts and hitting the enter key will return the current settings.

SNET TCP PORT command sets the Port number for TCP/IP sockets. The syntax for this command is:

SNET TCP PORT n m where n = equals one of two sockets and m is the port number

Example:

snet tcp port 0 8088 socket 0 is port #8088
snet tcp port 1 8089 socket 1 is port #8089

Port numbers must be between 1024 and 65535.
The Telnet port (23) may also be available. See TELNETLOCK command.

SNET TCP Idle command sets the socket life for the connection. The syntax for this command is:

SNET TCP Idle n (n=seconds) (1 to 3600 sec)
Default = 60 sec
SNET TCP Idle (display)
TCP Idle = 60
SNET TCP Idle 0 Socket never dies until the computer that established the socket kills it.

Setting the TCP Idle to 0 will force the socket to stay alive until the program that established the socket kills it.

WARNING: This can lead to issues if there is a network disconnect or the computer that established the socket locks up. If the computer that establishes the socket cannot kill the socket, no one will be able to connect to the switch until the Cytec unit is rebooted.

TCPAnswerback – Answerback

Syntax: TCPANSWERBACK n n = 0, 1 or 2

Answerback will enable or disable the transmission of a single character followed by an end of line upon the completion of all commands. The Answerback character will be a 1 or 0 depending on what command is sent. It is used to verify that the command was accepted and can verify completion of relay control commands. See **Section 5.5.2**

Eg.	"TCPANSWERBACK 0 "	Turn answerback off.
	"TCPANSWERBACK 1 "	Turn answerback on
	"TCPANSWERBACK 2 "	Turn answerback plus terminator on

Note: TCPANSWERBACK 2:
This setting appends a set of square brackets to the answerback byte.

Eg.	Send:	"L0 0"	Latch Module 0 Switch 0.
	Receive:	"1[]"	End of line follows the terminator

5.5 RUNNING THE CYTEC FACTORY APPLICATION ON THE IF12

The factory application that is included with the IF12 Control module includes:

- System Parameter Settings
- Matrix Parameter Settings
- Remote Switch Control
- List/Config Management
- File Management
- Custom Labeling

The URL request in the browser should look like the following:

http://<Device IP>

Where <Device IP> is replaced with the corresponding IP address. For more information on finding the IP address of your device, please see the device discovery section of this manual.

5.6 COMMAND FORMAT/COMPLETION

COMMAND FORMAT

All commands consist of at least one ASCII character indicating the command followed by optional values. After the command string is sent, an End of Line Character must be sent to affect the command.

If values are included with the command, the first value does not need to be separated from the command; all subsequent values MUST be separated by spaces or commas, eg. L1 2.

Multiple commands may also be sent on one line. Commands must be separated by a semi-colon character. Command line length is limited to 19 characters so avoid abusing this feature.

Examples:	"L2 7;C"	Connects Input 2 to Output 7 then clear
	"U4 7;L 1 2"	Unlatch Mod 4, Sw 7 then Latch Mod 1, Sw 2

COMMAND COMPLETION

A code representing the last requested switch point status (open or closed) and command completion will be stored by the matrix.

If the LAN or RS232 answerback function is enabled, a single character followed by end of line will be sent upon completion of all commands. Answerback may also include a termination character.

Note: Command Completion is NOT updated until the matrix finishes the requested operation.

Command Completion Codes – See Section 5.8 for error and completion codes

5.6.1 END OF LINE CHARACTER (EOL)

A received end of line character will cause the control module to execute the ASCII command string. The end of line character may be sent as a carriage return (CR) or New Line / Line Feed (NL/LF) for RS232 interfaces and a New Line / Line Feed (NL/LF) for IEEE488 interfaces or LAN interfaces. The IEEE488 also allows for the END control line being true with the last data character to initiate the command.

Valid end of Lines:

CR, LF or NL	LAN, RS232 or IEEE488
CR and END	IEEE488
LF/NL and END	IEEE488

Note that the terms New Line and Line Feed are often used to mean the same thing.
Both are expressed as \n in most programming languages and are shown on the ASCII table as "LF".
LF = Line Feed / New Line represented as \n, on ASCII table it is Decimal 10, or Hex A (0xA).
CR = Carriage Return represented as \r, on ASCII table it is = Decimal 13 or Hex D (0xD).

When any data is returned from the switch, the data will also be followed by an End Of Line character (EOL).

Notes - All Interfaces: Upon requesting status output characters MUST be received by the requesting device. Failure to do this will prevent further use of the matrix.

Access Code
Some commands require an access code number to be included with the command. This code prevents inadvertent operation of system modifying commands. The access code is 73.

5.7 SETUP COMMANDS

Matrixsize command sets the matrix size. The syntax for this command is:

matrixsize mtx# #mods #rlys

mtx#: For mainframes this is 0, for Mesa expansion systems this is the matrix number for the expansion chassis.

#mods: The maximum number of modules for the chassis.

#rlys: The maximum number of relays per module.

Example: matrixsize 0 16 8 Sets the # of modules to 16 and the maximum number of relays per module to 8 for a mainframe chassis (mtx is 0).

Typing matrixsize and hitting the enter key will return the current settings and chassis type (for Mesa systems all of the expansion chassis settings will be returned as a list).

P Commands (Except for communications settings these are set at the factory to the correct value for your system and should not need to be altered)

- P0 n 73 Set maximum number of matrices to 'n'. For Mainframes n = 1, for Mesa Control n = number of expansion chassis
- P6 n 73 n can be 1 or 0. 1 turns RTS/CTS handshaking on, 0 turns RTS/CTS handshaking off. This setting only applies to serial communication.
- P7 n 73 n can be 1 or 0. 1 turns Use RAM on, 0 turns Use RAM off.
- P8 n 73 n can be 0 to 6. Sets the default list (configuration) to load at power up if Use RAM is on.
- P10 n 73 Set maximum number of inputs to 'n' for a Mainframe system. For Mesa Systems sets the maximum number of inputs for Matrix 0;
- P11 n 73 Set maximum number of inputs in Matrix 1 of a Mesa System to 'n'.
- P12 n 73 Set maximum number of inputs in Matrix 2 of a Mesa System to 'n'.
- P13 n 73 Set maximum number of inputs in Matrix 3 of a Mesa System to 'n'.
- P14 n 73 Set GPIB address to 'n'. n can be 0 to 31.

- P19 n 73 Set the baud number to ‘ n’. See RS232 configuration section for corresponding baud rate to baud number.

- P20 n 73 Set maximum number of outputs to ‘n’ for a Mainframe system. For Mesa Systems sets the maximum number of outputs for Matrix 0;

- P21 n 73 Set maximum number of outputs in Matrix 1 of a Mesa System to ‘n’.

- P22 n 73 Set maximum number of outputs in Matrix 2 of a Mesa System to ‘n’.

- P23 n 73 Set maximum number of outputs in Matrix 3 of a Mesa System to ‘n’.

- P90 n 73 Set the system ID number to ‘n’. Used in large systems to differentiate between chassis.

5.8 SWITCH COMMANDS

General Notes

For LAN and RS232, after sending any command the Cytec control will return an integer Answerback character if Answerback (TCPAnswerback for LAN) is ON. Answerback/TCPAnswerback is turned on by default and is Cytec’s preferred operation since it allows you to verify commands are accepted before continuing.

If the command was a switch operation command such as Latch (L) or Unlatch (U), the character will be a meaningful status response where 1 = switch latched and 0 = switch unlatched. This may be used to verify that the command was received correctly.

Any other commands sent will also generate an answerback character which may be either a 1 or 0 and either character will indicate the command was received but the value is meaningless so either is acceptable.

Answerback may be turned off when using LAN or RS232 although it is not recommended. Answerback can also include a termination character for the LAN or RS232 interface.

Error Characters

If a command is sent incorrectly, an error character will be generated and added to the answerback character. Since the answerback character may be a 1 or 0, there may be two values for error characters as described below.

Answerback returned:

Dec	Hex	
1	30	Latch completed without errors.
0	31	Unlatch completed without errors.

2 or 3	32 or 33	Unknown command, first character unrecognizable.
4 or 5	34 or 35	Incorrect entries, number or type of entries incorrect.
6 or 7	36 or 37	Entries out of limits, switch point out of usable range.
8 or 9	38 or 39	Invalid access code, number 73 not included when required.

Delays to Prevent Errors

It is important to recognize that with modern computers and control interfaces, it is possible to stream commands to the switch matrix faster than the relays can physically operate. Many electro-mechanical relays may take between 2 to 20 ms to close or open. This can result in unpredictable results if certain operations are streamed together without considering this delay.

A good example of this type of problem occurs if a Latch command is sent and is immediately followed by a status request. Many of Cytec's products actually base status on current flow through the relay drives so it is possible to send a command and request status before the relay has physically operated, resulting in incorrect status feedback.

Typically, a 5 to 20 ms delay between commands requiring feedback can ensure that this is never an issue.

L,U,X – Latch, Unlatch, Multiplex Commands

Syntax: Cmd Output
Cmd Input, Output
Cmd Matrix, Input, Output
The specified switchpoint is operated on. Note: For mainframe systems the matrix number will be 0.

(Cmd = 'L', 'U' or 'X')
L = Latch = Turn switch ON Closes the specified point, all others unaffected.
U = Unlatch = Turn switch OFF Opens the specified point, all others are unaffected.
X = Multiplex = Clear + Latch Opens all points, then Latches the specified point.

E.g. "U0 2 3" Matrix 0, Input 2, Output 3 is disconnected. (OFF)
"L0 1 3" Matrix 0, Input 1, Output 3 is connected. (ON)
"L0 1" Input 0, Output 1 is connected. (ON)
"L2 3 7" Matrix 2, Input 3, Output 7 is connected. (ON)
"X0 3 0" Clear all switch points (turn them all OFF) then Latch Matrix 0, Input 3, Output 0.

If a single integer value is sent, the control module assumes it is an output value and defaults to the last module value sent. If two integers are sent, the control module assumes they are a input and output value and defaults to the last matrix value sent.

E.g. "L3 2 3" Matrix 3, Input 2, Output 3 is connected (ON). Then, "L1 4" Assumes Matrix 3. Matrix 3, Input 1, Output 4 is connected (ON). Then,"L5" Assumes Matrix 3, Input 1. Matrix 3, Input 1, Output 5 is closed (ON).

Some Cytec programming examples may refer to Mod #, Rly # (Relay #). The terms Switch (Sw) and Relay (Rly) mean the same thing. For Unidirectional matrix switches the Module # may be thought of as Input #, and the Switch or Relay # may be thought of as the Output #.

C - Clear Command

Syntax: C
All points in the chassis are opened.

E.g. "C" All switches in the chassis are opened.

For IEEE488.2, The C command is the same as the *RST (reset) function.

5.9 STATUS AND INTERROGATE COMMANDS

DX/256x256 systems do not make use of the S command but instead use the I command to return information.

Some of these systems would return up to 65,536 characters if they accepted the S command which becomes very difficult to deal with. Instead, they use the "I" command which only returns the address of switch points which are on. This limits the amount of returned data to a maximum of 256 addresses which is much easier and quicker to deal with.

These systems will interpret any "S" command sent as an "I" command and respond accordingly.

See the Interrogate Command Section for details.

5.10 I – INTERROGATE COMMAND

Syntax: I

The Interrogate function will return a list of all connected (ON) switch points. Each switch point will be followed by an "end of line" (EOL). The switch point is listed as the Input# and then Output#.

<I> Request interrogation.
Receive: <Module#><comma><space><Switch#><EOL>

Eg.	Send	"I"	
	Receive	"0, 0" eol	Input 0, Output 0 Connected. End of line.
		"1, 6" eol	Input 1, Output 6 Connected. End of line.
		"3, 2" eol	Input 3, Output 2 Connected. End of line.
		"0" eol	Answerback character (if enabled). End of line.

For system such as a DX/256x256 the "I" command may return up to 256 addresses. Be sure your buffer size can handle the amount of returned data.

5.11 OTHER COMMANDS

F - Front Panel
Syntax: F n 73 n = 0 or 1

Front panel lock-out will be initiated by the receipt of a 0 character and enabled by the receipt of a 1 character followed by the access code. The access code prevents inadvertent lock-out from occurring. Lock-out will prevent any operation of the system from the front panel until it is terminated from the remote (F 1) or power is turned off then on. Preset to panel enabled at power on.

Eg.	"F 0,73"	Lock-out local operation.
	"F 1 73"	Enable local operation.

P - Program
Syntax: P n1,n2,73

The program command allows the operator to setup matrix dependent variables. These include matrix switch configuration and certain interface functions. Use of the P commands is complicated and varies greatly between systems. Your system should have been provided with the correct P command set-up.

If you need to change the matrix configuration, number of allowed modules, or other obscure set-up configurations on your system we recommend you contact Cytec and we can walk you through the P commands needed for your specific system. Please provide the serial # of your system when you contact us.

N - Revision Number (Same as IEEE488 *IDN?)

Syntax: N

The 'N' command will cause the matrix to return its current revision number followed by an integer identifier, followed by an end of line.

Eg.	Send:	"N"	Request revision Number
	Receive:	" Cytec, 11-21-60, DX256, IF-12, 2.12 " eol	Text string returned.

Where: “Cytec” = manufacturer.
“11-21-60" = control module board number.
“DX256” = model number.
“IF-12" = control module name.
“2.12” = Firmware Revision # (example).
“0" = Integer identifier.

Note: When requesting the Revision number, all characters must be received before the system can be resumed.

**NOTE - The text string received from the 'N' Command will vary depending on the type of system.*

Integer identifier

The N command now includes a single byte which can be used as an identifier for Cytec systems. The identifier is a single byte integer so it may 0 to 255. WE do not assign this and it has no meaningful relationship to any product. It is simply a number which may be assigned to a chassis so that the end user can acknowledge that a specific Cytec chassis is communicating. It is up to the customer to assign the number and keep track of it. It allows them to poll multiple chassis and know that the one they are talking to is, for example, the JX/256 that they assigned the identifier "13" to.

Command to enter or change the number:

P90 n 73 where n is the number from 0 to 255

5.12 INPUT / OUTPUT vs MODULE / SWITCH NOMENCLATURE

Most of the switching systems sold by Cytec are completely bi-directional and can be used in a variety of ways by the customer so it is impossible for us to use the terms Input and Output, even though it is what probably makes the most sense to the end user when connecting signals to the switch.

We label and control the switches using Module# and Switch# to avoid this confusion since for most systems either can be considered an Input or an Output.

However some Cytec systems are uni-directional and therefore do have assigned Inputs and Outputs:

Uni-directional Systems with defined Input and Output ports:

DX and DXM high speed digital switch matrix systems
VDX and VDM analog or digital switch matrix systems
TX analog systems
FX fiber optic systems

For all of these systems:

Module # = Input
Switch # = Output

Note that these systems will all allow multiple Outputs to be connected to a single Input, but will never allow multiple Inputs to be connected to one Output. There are internal controls to prevent this and attempting to do it will simply disconnect a previously set path.

5.13 LIST MANAGEMENT

Lists can be set most easily through the device webpage, which can be accessed by typing the IP address for the system into any browser address bar. Currently, Cytec switches allow only nine saved lists and list 0 is always the current latched points. Valid values for n are 1-9.

- BS n 73: Saves the current latched switchpoints in List n
- BL n 73: Clears the switch and loads the switch points in List n.
- BD n 73: Displays the switch points in List n.
- BC n 73: Clears List n.

5.14 MATRIX COMMAND SUMMARY

COMMAND	FUNCTION
L output L input, output	Latch switch point.
U output U input, output	Unlatch switch point.
X output X input, output	Multiplex switch point.
C	Clear entire system.
S S input, output	Return status.
I	Interrogate Closed Points.
F 0/1 73	Disable/Enable Front Panel.
P parameter value 73	Program parameter.
N	Revision Number

RS232 Specific Commands

R baud, RTS/CTS 73	Baud Rate, RTS/CTS operation.
A 0/1 73	Disable/Enable Answerback.
E 0/1 73	Disable/Enable Echo.

V 0/1 73 Disable/Enable Verbose.

TCP/IP Specific Commands

TCPANSWERBACK 0/1/2	Disable/Enable Answerback/Termination
IFCONFIG aaa.aaa.aaa.aaa nnn.nnn.nnn.nnn	a = ip address in dotted decimal format n = subnet mask in dotted decimal format
SNET TCP PORT n m	Where n = equals one of two sockets and m is the port number

5.15 IF-12 (RS232/LAN/GPIB) DEFAULT CONFIGURATION SETTINGS

System parameters can be set most easily through the device webpage, which can be accessed by typing the IP address for the system into any browser address bar.

Default Values

TCP Settings:

Port 0	8080
Port 1	8081
Socket Timeout	60 seconds
TCPAnswerback	1 (on)
TelNet Lock	0 (off)

Serial Settings:

Answerback	1 (on)
Verbose	0 (off)
Echo	0 (off)
Baudrate	9600
RS Handshake	1 (RTS/CTS)

GPIB Settings:

GPIB Address	7

Front Panel Settings:

Mux Config	0 (Single 32) only for 16 or 32 channel systems
Front Panel	1 (on)

Miscellaneous Settings:

Use RAM (startup)	0 (off)
Default List	0 (currently latched switchpoints)
Sys ID Number	0

5.16 LCD DISPLAY/KEYPAD MANUAL CONTROL OPTION

The Keypad/Display option allows manual control of the matrix from the front panel. Keypad

operation is always enabled at power on but may be disabled by the remote command, 'F 0 73'.

Display
The display contains two lines with sixteen characters per line. The top line displays matrix commands and numeric entry. The bottom line displays the status of the entry or operation. The display will also show the last command entered from the remote computer interface when the front panel is enabled.

Keypad
The keypad consists of ten numeric keys, four function keys, a space key and an enter key.

Key	Function
0-9	Numeric entries.
space	Delimits between numeric entries.
L	Latch operation.
U	Unlatch operation.
X	Multiplex operation.
C	Clear operation.
ENTR	Execute displayed operation.

Operation
A matrix command key, **L**, **U**, **X** or **C**, MUST be pressed before numeric entry keys. Pressing any key except a matrix command key causes the message **Enter Cmd First** to be displayed. After pressing a matrix command key the command and a cursor are displayed. The switch point to be operated on may now be entered with the numeric and space keys. The entry format is the same as described in the MATRIX OPERATION section and described briefly by the following table:

Command Key	Display Line 1	Line 2
L	Lat _	Enter Point
U	Unl _	Enter Point
X	Mux _	Enter Point
C	Clr _	Enter Matrix

The numeric keypad now allows selection of the Module and Relay (Input and Output) to be operated on. Each entry may be multiple digits and a space must be pressed between selections.

Key	Line 1 Display	Line 2 Message
L	Lat _	Enter Point
1	Lat 1_	
Space	Lat 1 _	
2	Lat 1 2_	
3	Lat 1 23_	
Enter Key	Lat 1 23_	1

The **ENTR** key may now be pressed to execute the displayed operation. If the displayed entry is incorrect or the operation is not desired, pressing any matrix command key will clear the display and restart the entry.

Status Display
After the **ENTR** key is pressed, the displayed operation is attempted to be executed by the

control module. If the execution is successful, a **Point Closed** or **Point Open** message will be displayed on line 2. If the operation cannot be executed, an error message will be displayed.

Line 2 Message	Status
Ready	Displayed after power on.
Enter Point	The ENTR key has not been pressed, command and selection mode.
Point Closed	The selected point was closed.
Point Open	The selected point was opened.
Points Open	All points opened, Clear operation.
***Err: limits	The selected point is outside the programmed size of the matrix.
***Err: entry	An incorrect entry was selected.

Front Panel Disable
The 'F' command allows enabling or disabling front panel operation. If the front panel is disabled, no operation can be performed from the keypad.

Remote Command	Line 1	Line 2
F 0 73	Panel	Disabled
F 1 73	Panel	Enabled

Contrast and LED Backlight Adjustment
Controls are provided to adjust the LCD contrast and LED backlight level. These controls should only need adjustment in extremely bright or dim environments or for acute viewing angles. Both LCD and LED circuits have temperature sensing elements that will automatically adjust the output level for changes in the ambient temperature.

5.17 MEMORY SANITATION PROCEDURE

Cytec's IF-12 uses the NXP MCF54415 microprocessor with 32 MB of non-volatile flash memory which is used to store user lists and labels for the webpage factory application and 128kB of user parameter storage. If the unit is ever removed from service or needs to be sanitized for disposal the memory can be erased using one of the following methods.

1) Easiest with least damage. Contact Cytec for factory application .bin file at: sales@cytec-ate.com or 1-585-381-4740.

2) Permanent. Remove cover. Locate IF-12. Remove the NetBurner core board by disconnecting the LAN cable and prying the module off the IF-12 board. Destroy the NetBurner board. Unit is non functional until IF-12 has been replaced.

APPENDIX - EXAMPLE PROGRAMS

Java LAN Programming Example:

```
import java.net.*;  // for Socket
import java.io.*;   // for IOException and Input/OutputStream

public class if12_lantester
{
        static final int N_MODS = 4;
        static final int N_RLYS = 12;

        /**----------------test if12 utility functions--------------------------------------*/

        public static void main(String[] args) throws IOException, InterruptedException
        {
            if  (args.length != 2) // Test for correct # of args. IP Address and Port
                throw new IllegalArgumentException("Parameter(s): <IP Address> [<Port>]");

            String server = args[0];       // Server name or IP address
            int servPort = Integer.parseInt(args[1]);  // Port Number

            // Create socket that is connected to server on specified port
            Socket socket = new Socket(server, servPort);
            System.out.println("Connected to server...sending string");
            InputStream in = socket.getInputStream();
            OutputStream out = socket.getOutputStream();

            if12_lan if12 = new if12_lan();

            // Initialize Device: Turn Verbose & Echo off, Answerback on
            if (if12.init_LAN(in,out) < 0)
                throw new SocketException("Error Initializing Device");

            // Clear Device: Unlatch all relays
            if (if12.matrix_clear(in,out) != 48)
                throw new SocketException("Error clearing Device");

            // Latch and Unlatch Relays
            for (int mod =0; mod < N_MODS; mod++)
            {
                for (int rly=0;rly<N_RLYS;rly++)
                {
                    if (if12.point_ops(in,out,'L',0,mod,rly) != 49)
                    {
                        System.out.printf("Error latching Mod %d Rly %d\n",mod,rly);
                        break;
                    }
                    System.out.printf("Latched Mod %d Rly %d\n",mod,rly);
```

```
                    if (if12.point_ops(in,out,'U',0,mod,rly) != 48)
                    {
                        System.out.printf("Error unlatching Mod %d Rly %d\n",mod,rly);
                        break;
                    }
                    System.out.printf("Unlatched Mod %d Rly %d\n",mod,rly);
                }
            }

            socket.close();  // Close the socket and its streams
        }
}

public class if12_lan
{
        private int bytesRcvd,bytesSent;
        private byte[] rcvBuffer = new byte[256];

        public if12_lan()
        {
            bytesRcvd = 0;
            bytesSent = 0;
        }
        /**------------------Initialize Device-----------------------------------*/
        public int init_LAN(InputStream in, OutputStream out) throws IOException,
                InterruptedException
        {
            String str =new String("E0 73;V0 73;TCPANSWERBACK 1\n");
            // Convert string to bytes for writing to output stream
            byte[] byteBuffer = str.getBytes();
            // Send the encoded string to the if12
            out.write(byteBuffer);
            Thread.sleep(1000); //Wait one second
            // Receive the response from the device
            if ((bytesRcvd = in.read(rcvBuffer,0,9)) != 9)
                return -1;
            return 0;
        }
        /**--------------------clear matrix-----------------------------------*/
        public int matrix_clear(InputStream in,OutputStream out) throws IOException
        {
            String str = new String("C\n");
            // Convert string to bytes for writing to output stream
            byte[] byteBuffer = str.getBytes();
            // Send the encoded string to the if12
            out.write(byteBuffer);
            // Receive the response from the device
            if ((bytesRcvd = in.read(rcvBuffer,0,3)) == -1)
                return -1;
```

```
            return rcvBuffer[0];
        }
        /**------------------switchpoint operations-----------------------------*/
        public int point_ops(InputStream in, OutputStream out,char cmd,int mtx,
            int mod,int rly) throws IOException, InterruptedException
        {
            //Format command string to send to device
            String cmd_line = String.format("%c%d %d %d\n",cmd,mtx,mod,rly);
            // Convert string to bytes for writing to output stream
            byte[] byteBuffer = cmd_line.getBytes();
            // Send the encoded string to the if12
            out.write(byteBuffer);
            Thread.sleep(100); //Wait 1/10 second
            // Receive the response from the device
            if ((bytesRcvd = in.read(rcvBuffer,0,3)) == -1)
                return -1;
            return rcvBuffer[0];
        }
}
```

C LAN Programming Example

```
/* Cytec Matrix Test Program for LAN  */
/* This program uses Microsoft's WS2_32 Library */
/* and winsock2.h. These are available in the */
/* Microsoft SDKs and can be downloaded from */
/* Microsoft's Developer Network */
/* https://msdn.microsoft.com/en-us/default.aspx */

#include <stdio.h>
#include <winsock2.h>
#include <stdlib.h>   /* for exit() */

int init_LAN(int sock);
int point_ops(int sock,int cmd, int mtx, int mod, int rly);
int matrix_clear(int sock);
void DieWithError(char *errorMessage);


#define MAX_MTX   1
#define MAX_MOD  4
#define MAX_RLY 12

int main(int argc, char *argv[])
{
        int sock;
        char *servIP = "10.0.0.144";  /*Default IP Address*?
        struct sockaddr_in servAddr; /* IP address */
```

```
unsigned short servPort = 8080;  /* Port */
int mtx, mod, rly,status;

if (argc == 3)
{
     servIP = argv[1];
     servPort = atoi(argv[2]);
}


WSADATA wsaData;          /* Structure for WinSock setup communication */
WSAStartup(0x202, &wsaData);       /* Load Winsock 2.2 DLL */

/* Create a reliable, stream socket using TCP */
if ((sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))<0)
     DieWithError("socket() failed");

/* Construct the server address structure */
memset(&servAddr, 0, sizeof(servAddr));     /* Zero out structure */
servAddr.sin_family      = AF_INET;         /* Internet address family */
servAddr.sin_addr.s_addr = inet_addr(servIP);   /* Server IP address */
servAddr.sin_port        = htons(servPort); /* Server port */

/* Establish the connection to the server */
if (connect(sock, (struct sockaddr *) &servAddr, sizeof(servAddr))<0)
          DieWithError("connect() failed");

/* Initialize Device using init_LAN Function */
init_LAN(sock);

/* Send Clear Command to Device with matrix_clear Function*/
if ((status = matrix_clear(sock)) != 48)
     printf("Error clearing device/n");

/* Simple looping through switchpoints */

for(mtx=0; mtx<MAX_MTX; mtx++)
{
  for (mod=0; mod<MAX_MOD; mod++)
  {
     for (rly=0; rly<MAX_RLY; rly++)
     {
     if (((status = point_ops(sock,'L',mtx,mod,rly))) !=49)
          printf("Error point %d %d %d not closed\n",mtx,mod,rly);
          else
          printf("Latched point %d %d\n",mod, rly);

          if (((status = point_ops(sock,'U',mtx,mod,rly))) !=48)
          printf("Error point %d %d %d not open\n",mtx,mod,rly);
```

```
                else
                printf("Unlatched point %d %d\n",mod, rly);
            }
          }
        }
        closesocket(sock);
        WSACleanup();  /* Cleanup Winsock */
        return 0;
}
/*--------------Initialize--------------------------*/

int init_LAN(int sock)
{
        char rcvString[40];     /* Buffer for device response */
        int rcvStringLen;       /* Length of device response */
        void DieWithError(char *errorMessage);
        /* Initialize Device */
        if ((send(sock, "E0 73;V0 73;TCPANSWERBACK 1\n", 28, 0)) != 28)
        {
            DieWithError("send() failed");
        }
        Sleep(1000);     /* Wait for Response from Device */
        if ((rcvStringLen = recv(sock, rcvString, 9, 0)) < 9)
        {
            DieWithError("recv() failed or connection closed prematurely");
        }
        rcvString[rcvStringLen] = '\0';
        return 0;
}

/*-----------------Clear Matrix----------------------*/

int matrix_clear(int sock)
{

        char rcvString[8];     /* Buffer for device response */
        int rcvStringLen;      /* Length of device response */

        if ((send(sock,"C\n",2,0)) != 2)
            DieWithError("send() failed");

        Sleep(200);   /* Wait for Response

        /* Receive Response from Device */
        if ((rcvStringLen = recv(sock, rcvString, 10, 0)) <= 0)
            DieWithError("recv() failed or connection closed prematurely");
        rcvString[rcvStringLen] = '\0';
        int status = rcvString[0] & 0x3f;
```

```
	return status;
}
/*----------Switchpoint Operation----------------  */

int point_ops(int sock,int cmd, int mtx, int mod, int rly)
{
	char cmd_str[40];	/* Formatted command string */
	char rcvString[8];	/* Buffer for device response */
	int rcvStringLen;	/* Length of device response */

	/* Format String */
	sprintf(cmd_str,"%c%d %d %d\n",cmd,mtx,mod,rly);

	/* Send Command to Device */
	if ((send(sock,cmd_str,strlen(cmd_str),0)) != strlen(cmd_str))
		DieWithError("send() failed");

	Sleep(200);   /* Wait for Response

	/* Receive Response from Device */
	if ((rcvStringLen = recv(sock, rcvString, 10, 0)) <= 0)
	    DieWithError("recv() failed or connection closed prematurely");
	rcvString[rcvStringLen] = '\0';
	int status = rcvString[0] & 0x3f;

	return status;
}

/*------------Error Handling Function----------------------*/

 void DieWithError(char *errorMessage)
{
	fprintf(stderr,"%s: %d\n", errorMessage, WSAGetLastError());
	getchar();
	exit(1);
}
```

LabWindows RS232 Programming Example

```
*== Cytec Main Frame Control Include File rs232.h
==================================*/

int RS232port;

/*== GLOBAL FUNCTION DECLARATIONS
========================================*/

int CYRS232Initialize (int com_port, int baud_rate);
int CyIf3_read (char *buf);
```

```
int CyIf3_write (char *buf);
int CyIf3_close (void);


/*================================ END */

#include <ansi_c.h>
#include <utility.h>
/*============================================================================*/
/* Cytec Main Frame RS232 LabWindows/CVI Driver Module                        */
/*============================================================================/*

#include <rs232.h>
#include <formatio.h>
#include "CYRS232.h"

/*= STATIC VARIABLES
=========================================================*/
/* port contains the number of the port opened for the instrument module.  */
/* cmd is a buffer for RS-232 I/O strings.                                 */
/* rscnt contains the number of bytes transferred during a read or write.  */
/* CyIf3_err: the error variable for the instrument module                 */
/*============================================================================*/
//static int port;
static char cmd[26];
static int rscnt;
static int CyIf12_err;

/*= UTILITY ROUTINES
=========================================================*/
int CyIf12_invalid_short_range (short val, short min, short max, int err_code);
int CyIf12_invalid_integer_range (int val, int min, int max, int err_code);
int CyIf12_invalid_longint_range (long val, long min, long max, int err_code);
int CyIf12_invalid_real_range (double val, double min, double max, int err_code);
int CyIf12_read_data (char *buf, int cnt, int term);
int CyIf12_write_data (char *buf, int cnt);
int CyIf12_device_closed (void);
void CyIf12_setup_arrays (void);

int main()
{
        CYRS232Initialize(12,9600);

}

/*============================================================================*/
/* This function opens a com port for the instrument module, queries for   */
/* ID, and initializes the instrument to a known state.                    */
/*============================================================================*/
```

```
int CYRS232Initialize(int com_port, int baud_rate)
{
        char s[40];

   if (CyIf12_invalid_integer_range (baud_rate, 110, 19200, -2) != 0)
      return -14;


   CyIf12_err = OpenComConfig (com_port, "", baud_rate, 0, 8, 1, 512, 512);
   if (CyIf12_err<0) {
      return CyIf12_err;
   }
   CyIf12_err = SetComTime (com_port, 1.0);
   if (CyIf12_err<0) {
      return CyIf12_err;
   }


/*
   Set port to the number of the port just opened.
*/

   RS232port = com_port;

/* Initialize communication, Answerback ON, Verbose, Echo OFF */
        Fmt (s, "A1,73;V0,73;E0,73\r");
        CyIf12_err = (ComWrt (RS232port, s, StringLength(s)));
        if (CyIf12_err<0)  {
            return CyIf12_err;
        }
        Delay(.1);

        CyIf12_err = ComRdTerm(RS232port, s, 40, '\r');
        if (CyIf12_err<0) {
            return CyIf12_err;
        }
        Delay (1.0);

        FlushInQ (com_port);
        FlushOutQ (com_port);

   return CyIf12_err;
}

/*============================================================*/
int CyIf12_read (char *buf)
{
        return(CyIf12_read_data (buf, 40, '\r'));
```

```
}

int CyIf12_write (char *buf)
{
        return (CyIf12_write_data (buf, StringLength(buf)));

}


/*=============================================================*/
/* This function closes the port for the instrument module and sets the    */
/* port to zero.                                               */
/*=============================================================*/
int CyIf12_close (void)
{
/*  Check for device closed */

   if (CyIf12_device_closed())
      return CyIf12_err;

/*
   Close the com port.  If error, set CyIf3_err = rs232err+300.
*/

   CloseCom(RS232port);
   if (rs232err != 0) {
      CyIf12_err = rs232err+300;
      return CyIf12_err;
   }

   RS232port = 0;
   return CyIf12_err;
}

/* = UTILITY ROUTINES ==========================================*/

/*=============================================================*/
/* Function: Invalid Short Range                               */
/* Purpose:  This function checks a short to see if it lies between a      */
/*          minimum and maximum value.  If the value is out of range, set */
/*          the global error variable to the value err_code.  If the       */
/*          value is OK, error = 0.                            */
/*=============================================================*/
int CyIf12_invalid_short_range (short val, short min, short max, int err_code)
{
  if ((val < min) || (val > max)) {
    CyIf12_err = err_code;
    return -1;
  }
```

```
  return 0;
}
/*=================================================================*/
/* Function: Invalid Integer Range                                 */
/* Purpose:  This function checks an integer to see if it lies between a   */
/*           minimum and maximum value.  If the value is out of range, set */
/*           the global error variable to the value err_code.  If the      */
/*           value is OK, error = 0.                                       */
/*=================================================================*/
int CyIf12_invalid_integer_range (int val, int min, int max, int err_code)
{
  if ((val < min) || (val > max)) {
    CyIf12_err = err_code;
    return -1;
  }
  return 0;
}
/*=================================================================*/
/* Function: Invalid Long Integer Range                            */
/* Purpose:  This function checks a long integer to see if it lies between */
/*           a minimum and maximum value.  If the value is out of range,   */
/*           set the global error variable to the value err_code.  If the  */
/*           value is OK, error = 0.  The return value is equal to the     */
/*           global error value.                                           */
/*=================================================================*/
int CyIf12_invalid_longint_range (long val, long min, long max, int err_code)
{
   if (val < min || val > max) {
      CyIf12_err = err_code;
      return -1;
   }
   return 0;
}
/*=================================================================*/
/* Function: Invalid Real Range                                    */
/* Purpose:  This function checks a real number to see if it lies between  */
/*           a minimum and maximum value.  If the value is out of range,   */
/*           set the global error variable to the value err_code.  If the  */
/*           value is OK, error = 0.                                       */
/*=================================================================*/
int CyIf12_invalid_real_range (double val, double min, double max, int err_code)
{
  if ((val < min) || (val > max)) {
    CyIf12_err = err_code;
    return -1;
  }
  return 0;
}
/*=================================================================*/
```

```
/* Function: Device Closed                                          */
/* Purpose:  This function checks to see if the module has been     */
/*           initialized.  If the device has not been opened, a 1 is */
/*           returned, 0 otherwise.                                 */
/*==================================================================*/
int CyIf12_device_closed (void)
{
   if (RS232port == 0)  {
      CyIf12_err = 232;
      return -1;
   }
   return 0;
}

/*==================================================================*/
/* Function: Read Data                                              */
/* Purpose:  This function reads a buffer of data from the instrument. The */
/*           return value is equal to the global error variable.    */
/*==================================================================*/
int CyIf12_read_data (char *buf, int cnt, int term)
{
   rscnt = ComRdTerm(RS232port, buf, cnt, term);
         FlushInQ (RS232port);

   return rscnt;
}

/*==================================================================*/
/* Function: Write Data                                             */
/* Purpose:  This function writes a buffer of data to the instrument. The  */
/*           return value is equal to the global error variable.    */
/*==================================================================*/
int CyIf12_write_data (char *buf, int cnt)
{
   rscnt = ComWrt (RS232port, buf, cnt);

   return rscnt;
}

/*==================================================================*/
/* This function is called by the init routine to initialize global arrays  */
/* This routine should be modified for each instrument to include   */
/* instrument-dependent commmand arrays.                            */
/*==================================================================*/
void CyIf12_setup_arrays (void)
{
}
/*= THE END
====================================================================*/
```

LabWindows GPIB Programming Example

```
/*============================================================================*/

/*= Cytec IF-11 IEEE488 Control Module Include File ========================*/

/*== GLOBAL CONSTANT DECLARATIONS
==========================================*/

/* Replace 10 with the maximum number of devices of this type being used.  */
#define IF12_MAX_INSTR    10

/*== GLOBAL FUNCTION DECLARATIONS
==========================================*/
int if12_init (int, int, int *);

/** INSERT INSTRUMENT-DEPENDENT FUNCTION DECLARATIONS HERE **/

int if12_operate(int, int, int, int, int *);
int if12_write (int, char *);
int if12_read (int, int, char *, int *);
int if12_close (int);

/*=== END INCLUDE FILE
=================================================*/


/*============================================================================*/

#include <gpib.h>
#include <utility.h>
#include <formatio.h>
#include "cy_if12.h"


/*= INSTRUMENT TABLE
=================================================*/
/* address array: contains the GPIB addresses of opened instruments.       */
/* bd array: contains the device descriptors returned by OpenDev.          */
/* instr_cnt: contains the number of instruments open of this model type.  */
/*============================================================================*/
static int address[IF12_MAX_INSTR + 1];
static int bd[IF12_MAX_INSTR + 1];
static int instr_cnt;

/*= STATIC VARIABLES
=================================================*/
```

```
/* cmd is a buffer for GPIB I/O strings.                              */
/* if12_err: the error variable for the instrument module            */
/* ibcnt: contains the number of bytes transferred by GPIB reads and */
/*        writes.  See the GPIB library I/O Class for more information */
/*====================================================================*/
static char cmd[50];
static int if12_err;

/*= UTILITY ROUTINES
====================================================*/
int if12_open_instr (int, int *);
int if12_close_instr (int);
int if12_invalid_integer_range (int, int, int, int);
int if12_device_closed (int);
int if12_read_data (int, char *, int);
int if12_write_data (int, char *, int);
int if12_set_timeout (int, int, int *);
void if12_setup_arrays (void);

/*====================================================================*/
/* Function: Initialize                                               */
/* Purpose:  This function opens the instrument, queries the instrument */
/*           for its ID, and initializes the instrument to a known state.  */
/*====================================================================*/
int if12_init (addr, rest, instrID)
int   addr;
int   rest;
int * instrID;
{
        int  ID;

        if (if12_invalid_integer_range (addr, 0, 30, -1) != 0)
            return if12_err;
        if (if12_invalid_integer_range (rest, 0, 1, -3) != 0)
            return if12_err;

        if (if12_open_instr (addr, &ID) != 0)
            return if12_err;

        if (rest) {
            if (if12_write_data (ID, "C", 1) != 0) {
                if12_close_instr (ID);
                return if12_err;
            }
            Delay(0.01);
        }
        if12_setup_arrays ();
        *instrID = ID;
```

```
	return if12_err;
}

/*============================================================*/
/* - Operations: Latch, Unlatch, Multiplex, Clear and Status --- */

int if12_operate (instrID, Operation, Module, Relay, Status)
int instrID, Operation, Module, Relay;
int *Status;
{
	char s[20];

	*Status = -1;
	if (Operation == 'C') {
	    Fmt(s,"C");
	    if (if12_write_data(instrID, s, StringLength(s)) != 0)
	        return if12_err;
	    Delay(0.01);
	}
	else {
	    Fmt(s,"%c %d %d", Operation, Module, Relay);
	    if (if12_write_data(instrID, s, StringLength(s)) != 0)
	        return if12_err;
	}
	if (if12_read_data(instrID, s, 2) != 0)
	    return if12_err;
	*Status = s[0] & 0xf;
	return if12_err;
}

/*============================================================*/
/* Function: Write To Instrument                              */
/* Purpose:  This function writes a command string to the instrument.      */
/*============================================================*/
int if12_write (instrID, cmd_string)
int   instrID;
char *cmd_string;
{
	if (if12_invalid_integer_range (instrID, 1, IF12_MAX_INSTR, -1) != 0)
	    return if12_err;
	if (if12_device_closed(instrID) != 0)
	    return if12_err;

	Fmt (cmd, "%s<%s", cmd_string);
	if (if12_write_data (instrID, cmd, NumFmtdBytes()) != 0)
	    return if12_err;

	return if12_err;
}
```

```
/*=============================================================*/
/* Function: Read Instrument Buffer                            */
/* Purpose:  This function reads the output buffer of the instrument.      */
/*=============================================================*/
int if12_read (instrID, numbytes, in_buff, bytes_read)
int   instrID;
int   numbytes;
char *in_buff;
int  *bytes_read;
{
        if (if12_invalid_integer_range (instrID, 1, IF12_MAX_INSTR, -1) != 0)
            return if12_err;
        if (if12_device_closed(instrID) != 0)
            return if12_err;

        *bytes_read = 0;
        if (if12_read_data (instrID, in_buff, numbytes) != 0)
            return if12_err;

        *bytes_read = ibcnt;

        return if12_err;
}

/*=============================================================*/
/* Function: Close                                             */
/* Purpose:  This function closes the instrument.                      */
/*=============================================================*/
int if12_close (instrID)
int instrID;
{
        if (if12_invalid_integer_range (instrID, 1, IF12_MAX_INSTR, -1) != 0)
            return if12_err;
        if (if12_device_closed (instrID))
            return if12_err;

        if12_close_instr (instrID);

        return if12_err;
}

/*= UTILITY ROUTINES ========================================*/

/*=============================================================*/
/* Function: Open Instrument                                   */
/* Purpose:  This function locates and initializes an entry in the        */
/*           Instrument Table and the GPIB device table for the            */
/*           instrument.  The size of the Instrument Table can be changed  */
```

```
/*          in the include file by altering the constant                */
/*          IF12_MAX_INSTR.  The return value of this function is equal */
/*          to the global error variable.                               */
/*============================================================*/
int if12_open_instr (addr, ID)
int  addr;
int *ID;
{
        int i, instrID;

        instrID = 0;
        if12_err = 0;

/* Check to see if the instrument is already in the Instrument Table. */

        for (i = 1; i <= IF12_MAX_INSTR; i++)
            if (address[i] == addr) {
                instrID = i;
                i = IF12_MAX_INSTR;
             }

/* If it is not in the instrument table, open an entry for the instrument. */

        if (instrID <= 0)
            for (i = 1; i <= IF12_MAX_INSTR; i++)
                if (address[i] == 0) {
                    instrID = i;
                    i = IF12_MAX_INSTR;
                 }

/* If an entry could not be opened in the Instrument Table, return an error.*/

        if (instrID <= 0) {
            if12_err = 220;
            return if12_err;
        }

/* If the device has not been opened in the GPIB device table (bd[ID] = 0),*/
/*  then open it.                                                         */

        if (bd[instrID] <= 0) {
            if (instr_cnt <= 0)
                CloseInstrDevs("if12");
            bd[instrID] = OpenDev ("", "if12");
            if (bd[instrID] <= 0) {
                if12_err = 220;
                return if12_err;
            }
            instr_cnt += 1;
```

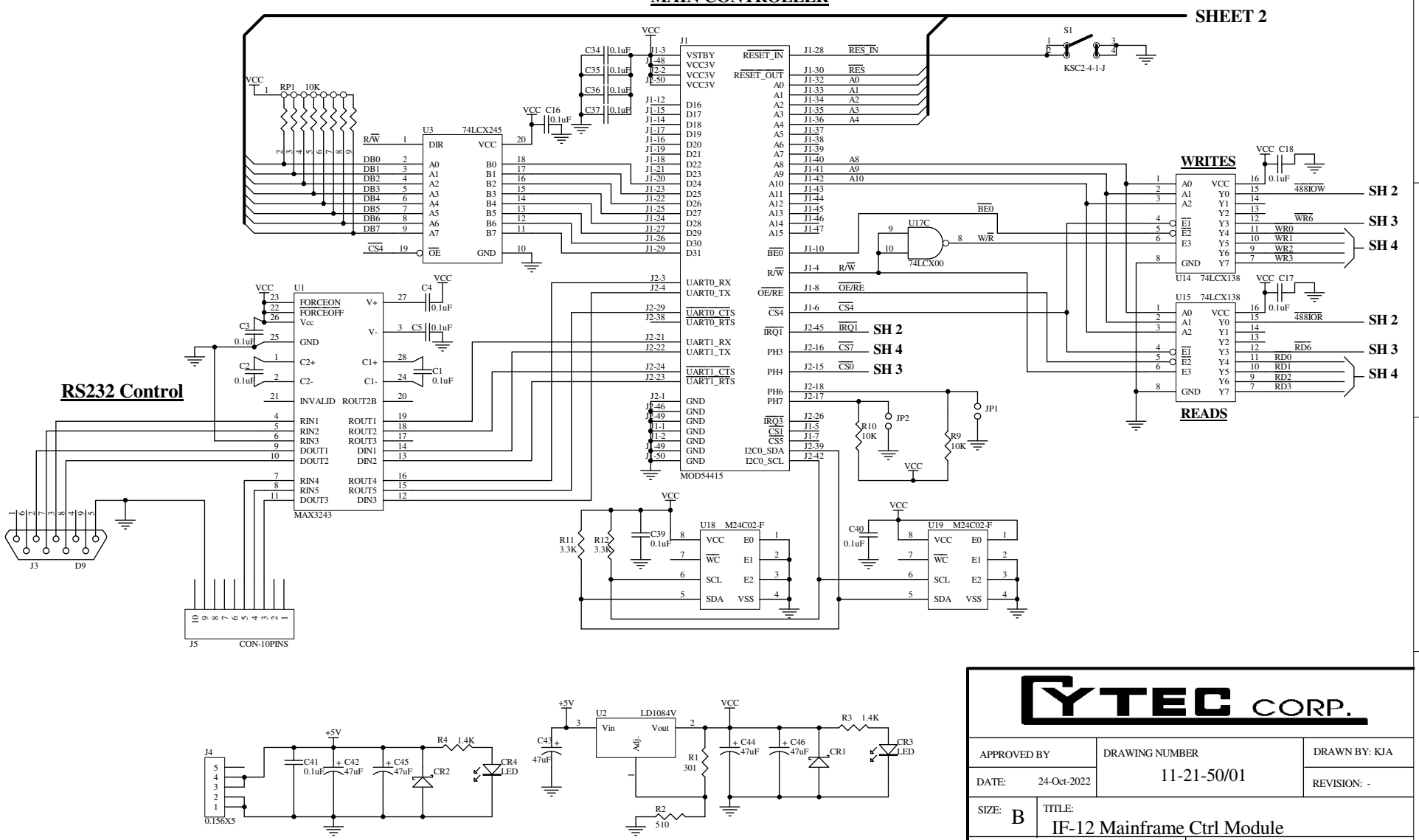
```
            address[instrID] = addr;
         }

/*  Change the primary address of the device    */

        if (ibpad (bd[instrID], addr) < 0) {
            if12_err = 233;
            return if12_err;
        }

        *ID = instrID;
        return if12_err;
}

/*=============================================================*/
/* Function: Close Instrument                                  */
/* Purpose:  This function closes the instrument by removing it from the   */
/*          GPIB device table and setting the address and bd[instrID] to  */
/*          zero in the Instrument Table.  The return value is equal to   */
/*          the global error variable.                         */
/*=============================================================*/
int if12_close_instr (instrID)
int instrID;
{
        if (bd[instrID] != 0) {
            CloseDev (bd[instrID]);
            bd[instrID] = 0;
            address[instrID] = 0;
            instr_cnt -= 1;
        }
        else
            if12_err = 221;

        return if12_err;
}

/*=============================================================*/
/* Function: Invalid Integer Range                             */
/* Purpose:  This function checks an integer to see if it lies between a   */
/*          minimum and maximum value.  If the value is out of range, set */
/*          the global error variable to the value err_code.  If the      */
/*          value is OK, error = 0.  The return value is equal to the     */
/*          global error value.                                */
/*=============================================================*/
int if12_invalid_integer_range (val, min, max, err_code)
int val;
int min;
int max;
int err_code;
```

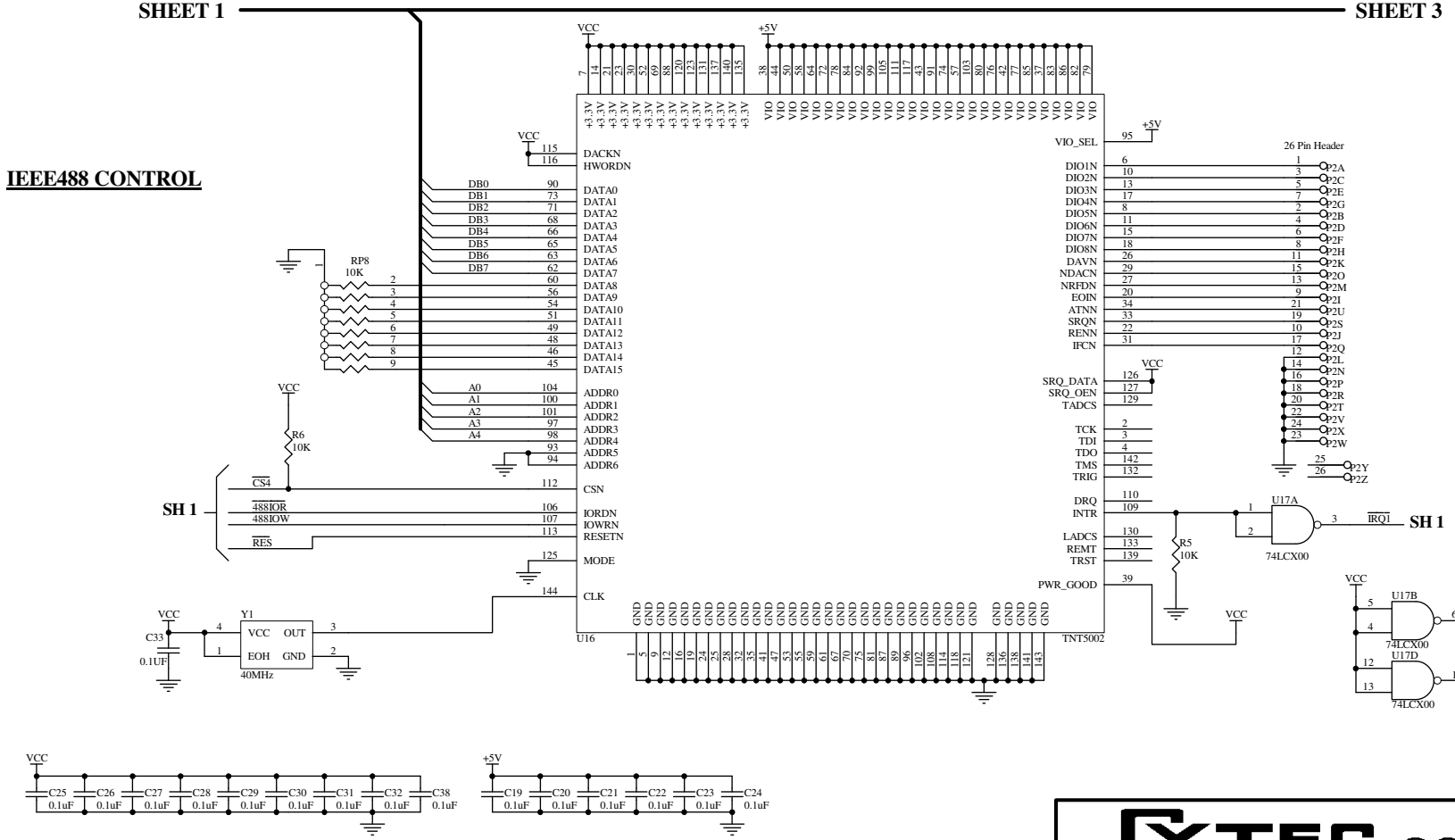
```
{
        if (val < min || val > max)
            if12_err = err_code;
        else
            if12_err = 0;

        return if12_err;
}

/*=============================================================*/
/* Function: Device Closed                                     */
/* Purpose:  This function checks to see if the module has been */
/*          initialized.  If the device has not been opened, set the */
/*          global error variable to 232, 0 otherwise. The return value */
/*          is equal to the global error value.                */
/*=============================================================*/
int if12_device_closed (instrID)
int instrID;
{
        if (bd[instrID] <= 0)
            if12_err = 232;
        else
            if12_err = 0;

        return if12_err;
}

/*=============================================================*/
/* Function: Read Data                                         */
/* Purpose:  This function reads a buffer of data from the instrument. The */
/*          return value is equal to the global error variable.  */
/*=============================================================*/
int if12_read_data (instrID, buf, cnt)
int   instrID;
char *buf;
int   cnt;
{
        if (ibrd(bd[instrID], buf, (long)cnt) <= 0)
            if12_err = 231;
        else
            if12_err = 0;

        return if12_err;
}

/*=============================================================*/
/* Function: Write Data                                        */
/* Purpose:  This function writes a buffer of data to the instrument. The  */
/*          return value is equal to the global error variable.  */
```

```
/*=============================================================*/
int if12_write_data (instrID, buf, cnt)
int   instrID;
char *buf;
int   cnt;
{
        if (ibwrt(bd[instrID], buf, (long)cnt) <= 0)
            if12_err = 230;
        else
            if12_err = 0;

        return if12_err;
}

/*=============================================================*/
/* Function: Set Timeout                                       */
/* Purpose:  This function changes or disables the timeout of the device.  */
/*           Refer to the LabWindows Standard Libraries Reference Manual   */
/*           for timeout codes. The return value is equal to the global    */
/*           error variable.                                   */
/*=============================================================*/
int if12_set_timeout (instrID, tmo_code, old_timeout)
int  instrID;
int  tmo_code;
int *old_timeout;
{
        *old_timeout = ibtmo (bd[instrID], tmo_code);
        if (ibsta <= 0)
            if12_err = 239;
        else
            if12_err = 0;

        return if12_err;
}

/*=============================================================*/
/* Function: Setup Arrays                                      */
/* Purpose:  This function is called by the init routine to initialize     */
/*           static arrays.                                    */
/*           This routine should be modified for each instrument to        */
/*           include instrument-dependent commmand arrays.                 */
/*=============================================================*/
void if12_setup_arrays ()
{
}
/*=== THE END ===============================================*/
```

LabView Drivers

Labview Drivers are available for download at https://cytec-ate.com/downloads/drivers/


MAIN CONTROLLER
SHEET 2
RS232 Control
WRITES
READS
MOD54415
MAX3243
74LCX245
74LCX138
74LCX00
M24C02-F
LD1084V
KSC2-4-1-J
CON-10PINS
SH 2
SH 3
SH 4
488IOW
488IOR
WR0
WR1
WR2
WR3
WR6
RD0
RD1
RD2
RD3
RES_IN
RESET_IN
RESET_OUT
UART0_RX
UART0_TX
UART0_CTS
UART0_RTS
UART1_RX
UART1_TX
UART1_CTS
UART1_RTS
I2C0_SDA
I2C0_SCL
APPROVED BY
DRAWN BY: KJA
DATE: 24-Oct-2022
DRAWING NUMBER
11-21-50/01
REVISION: -
SIZE: B
TITLE:
IF-12 Mainframe Ctrl Module
11-21-50-01.sch
SHEET 1 OF 4

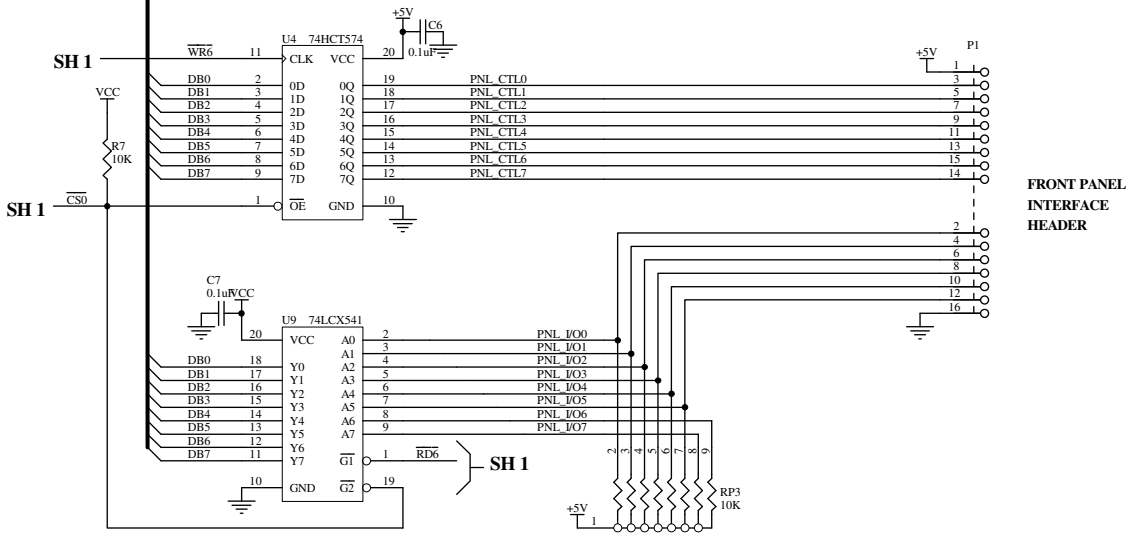


APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 24-Oct-2022	11-21-50/02	REVISION: -
SIZE: B	TITLE: IF-12 Mainframe Ctrl Module	
11-21-50-02.sch	SHEET 2 OF 4	

SHEET 2

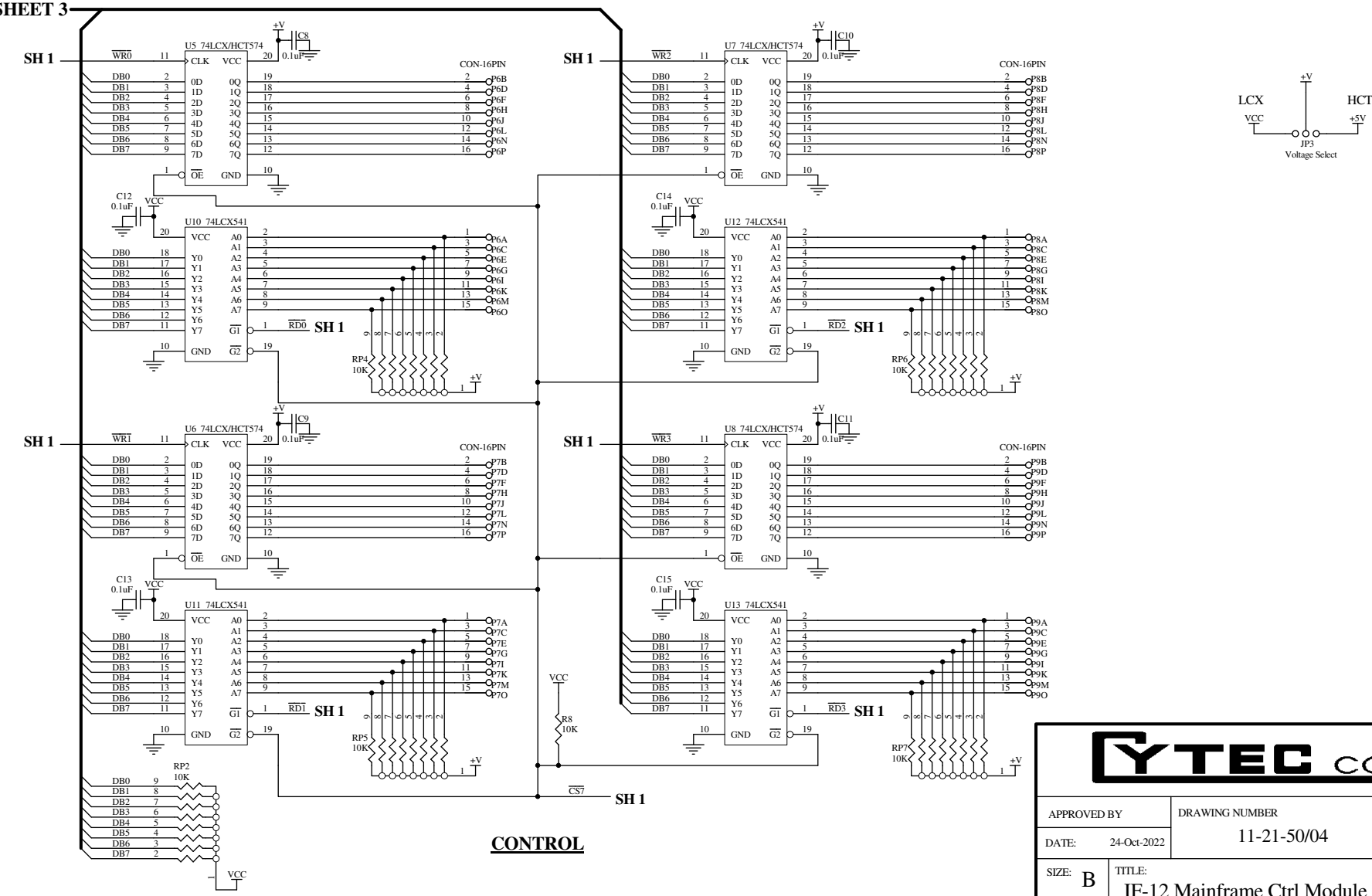
SHEET 4

LCD CTRL



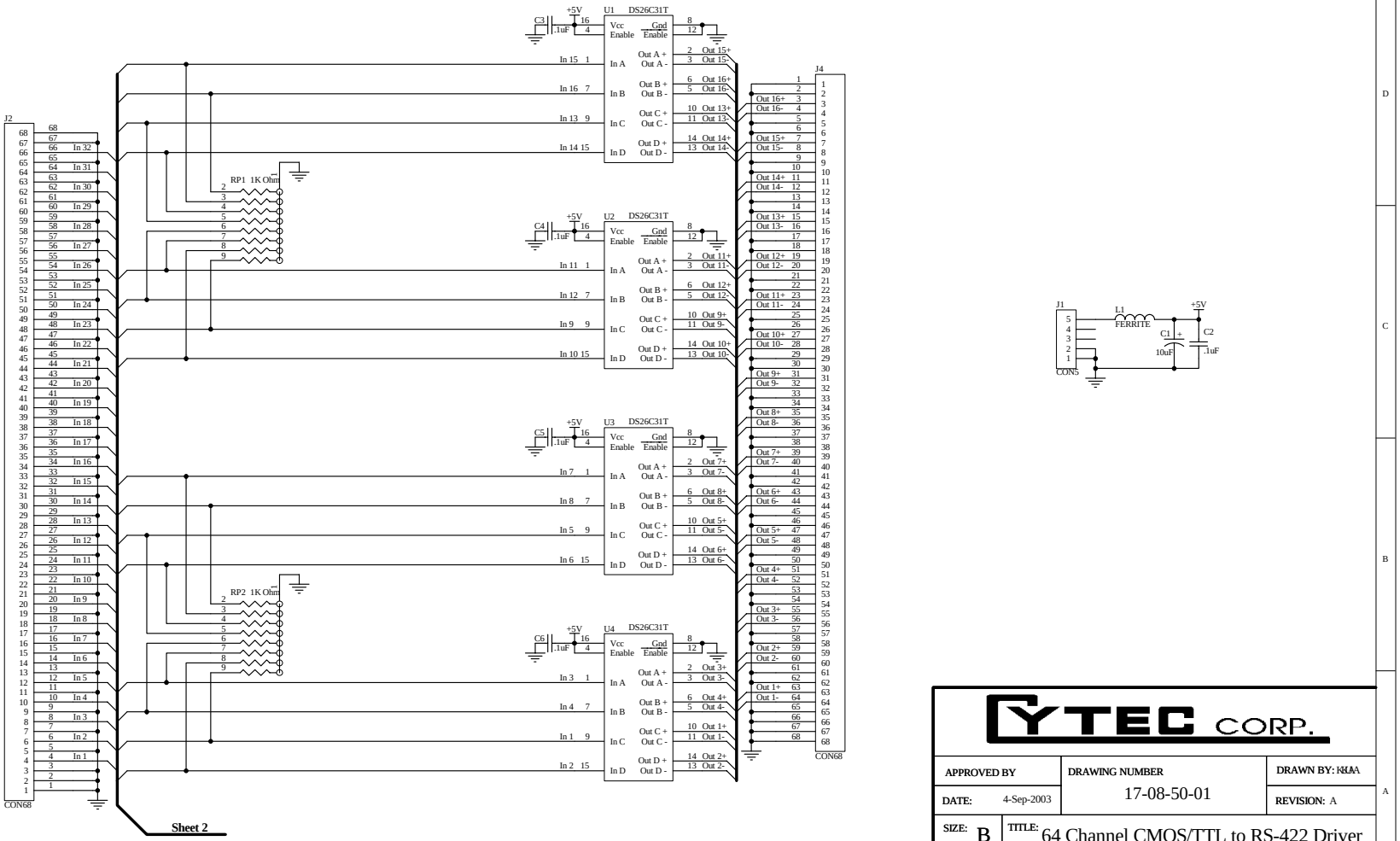
CYTEC CORP.		
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 24-Oct-2022	11-21-50/03	REVISION: -
SIZE: B	TITLE: IF-12 Mainframe Ctrl Module	
11-21-50-03.sch	SHEET 3 OF 4	

SHEET 3

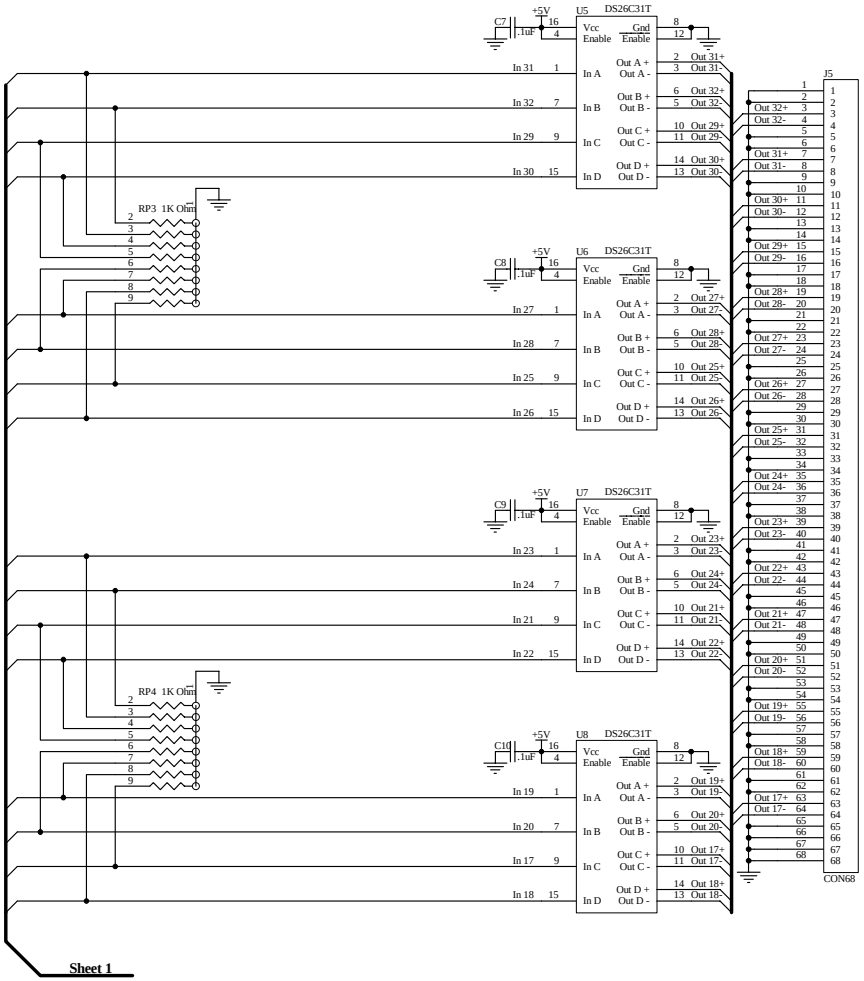


CONTROL

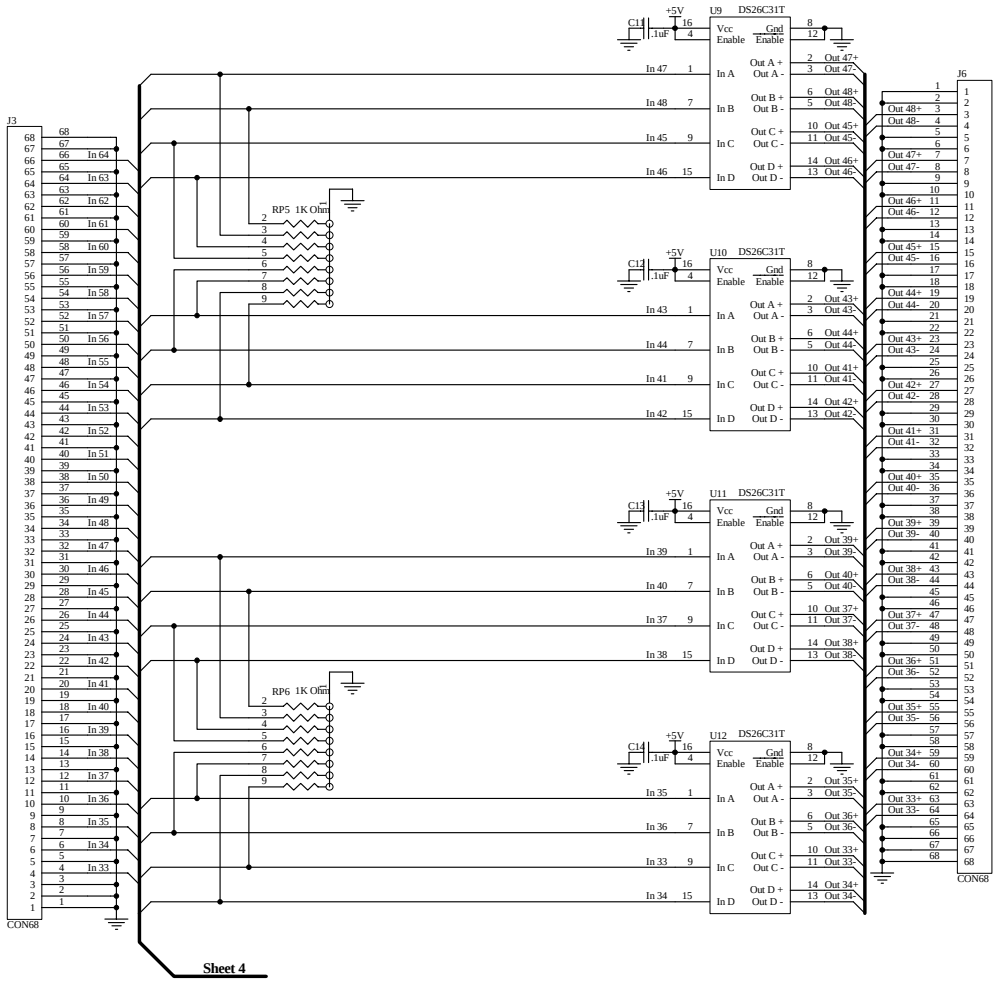
CYTEC CORP.			
APPROVED BY		DRAWING NUMBER	DRAWN BY: KJA
DATE:	24-Oct-2022	11-21-50/04	REVISION: -
SIZE: B	TITLE: IF-12 Mainframe Ctrl Module		
11-21-50-04.sch		SHEET 4 OF 4	



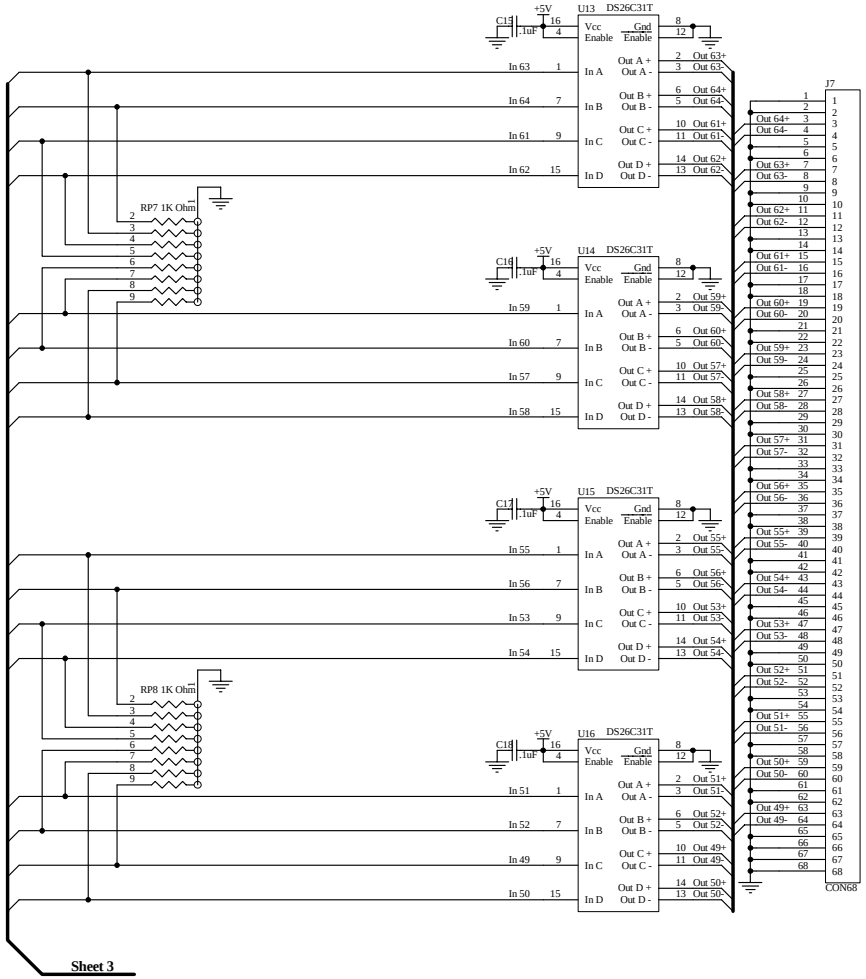
APPROVED BY		DRAWING NUMBER	DRAWN BY: KUA
DATE:	4-Sep-2003	17-08-50-01	REVISION: A
SIZE: B	TITLE: 64 Channel CMOS/TTL to RS-422 Driver		
17-08-50-01.sch		SHEET 1 OF 4	



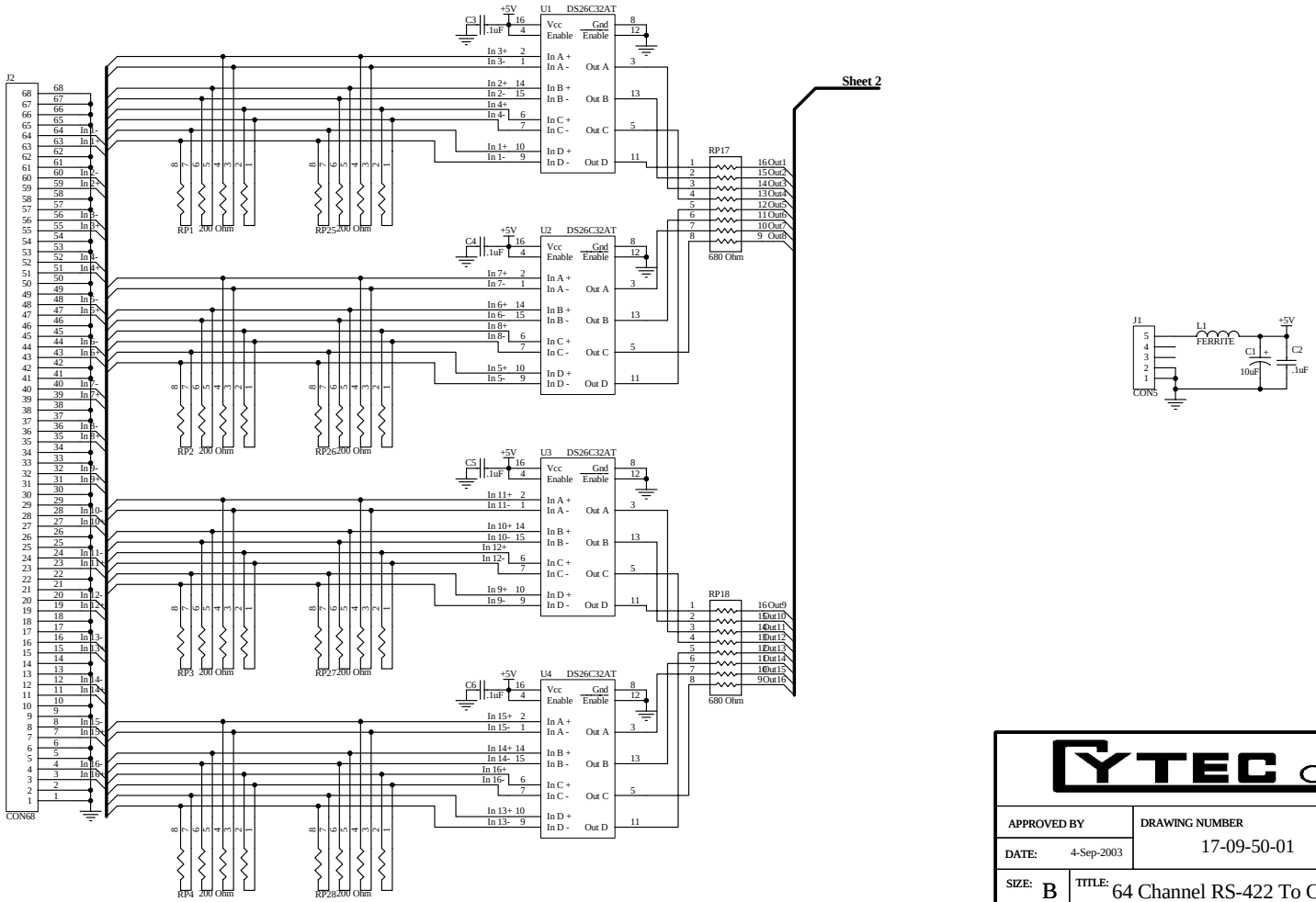
APPROVED BY	DRAWING NUMBER	DRAWN BY: KUA
DATE: 4-Sep-2003	17-08-50-02	REVISION: A
SIZE: B	TITLE: 64 Channel CMOS/TTL to RS-422 Driver	
17-08-50-02.sch		SHEET 2 OF 4



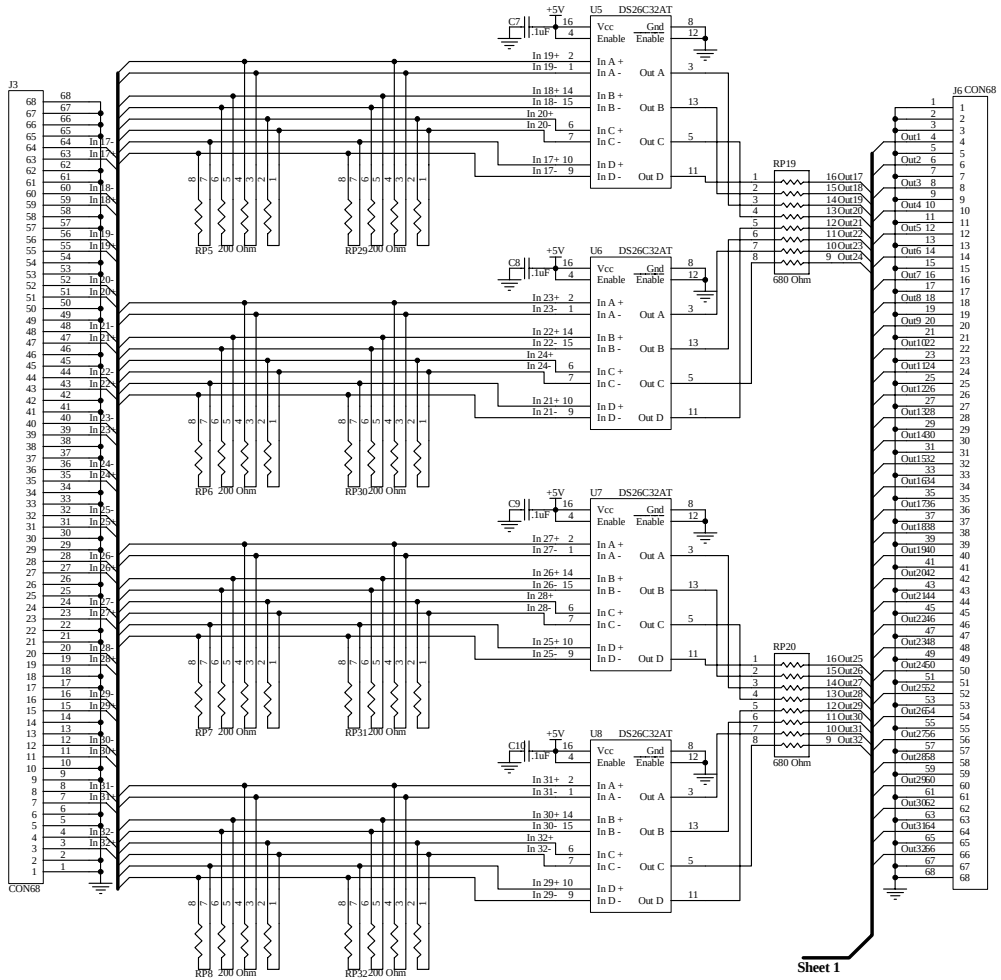
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 4-Sep-2003	17-08-50-03	REVISION: A
SIZE: B	TITLE: 64 Channel CMOS/TTL to RS-422 Driver	
17-08-50-03.sch		SHEET 3 OF 4



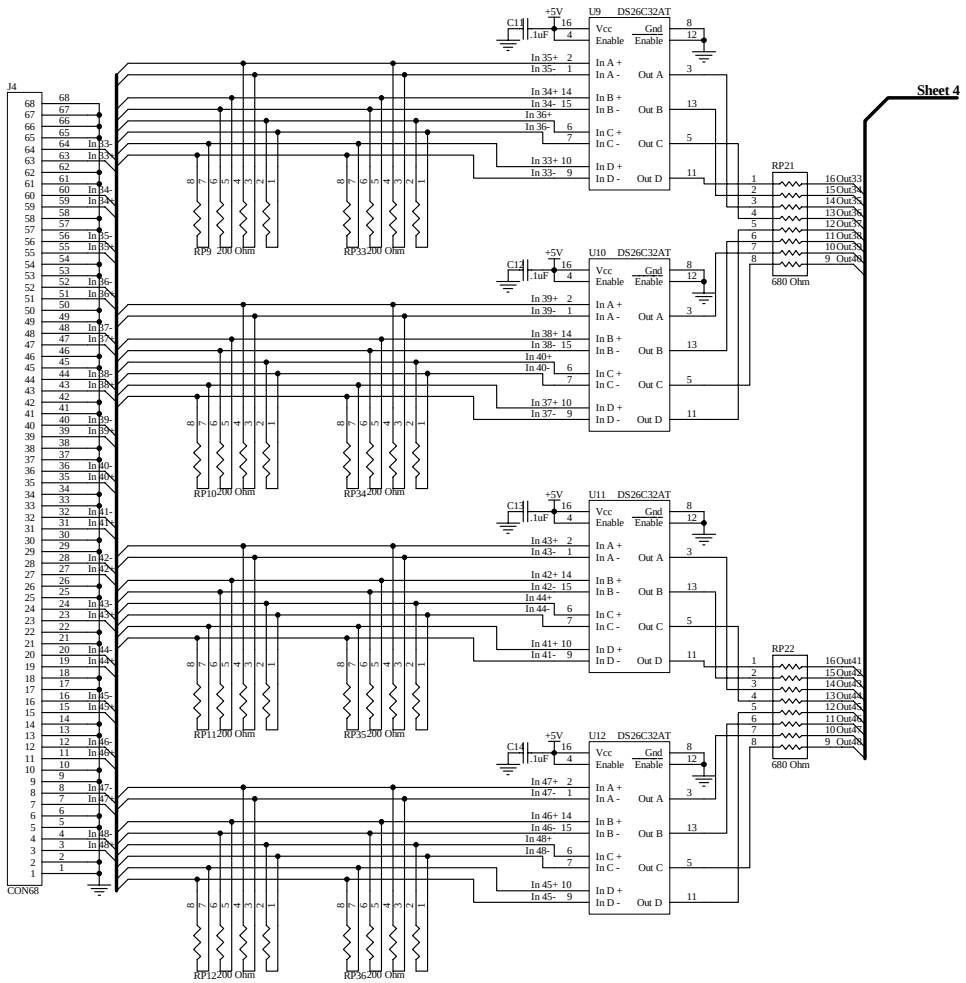
CYTEC CORP.		
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 4-Sep-2003	17-08-50-04	REVISION: A
SIZE: B	TITLE: 64 Channel CMOS/TTL to RS-422 Driver	
17-08-50-04.sch		SHEET 4 OF 4



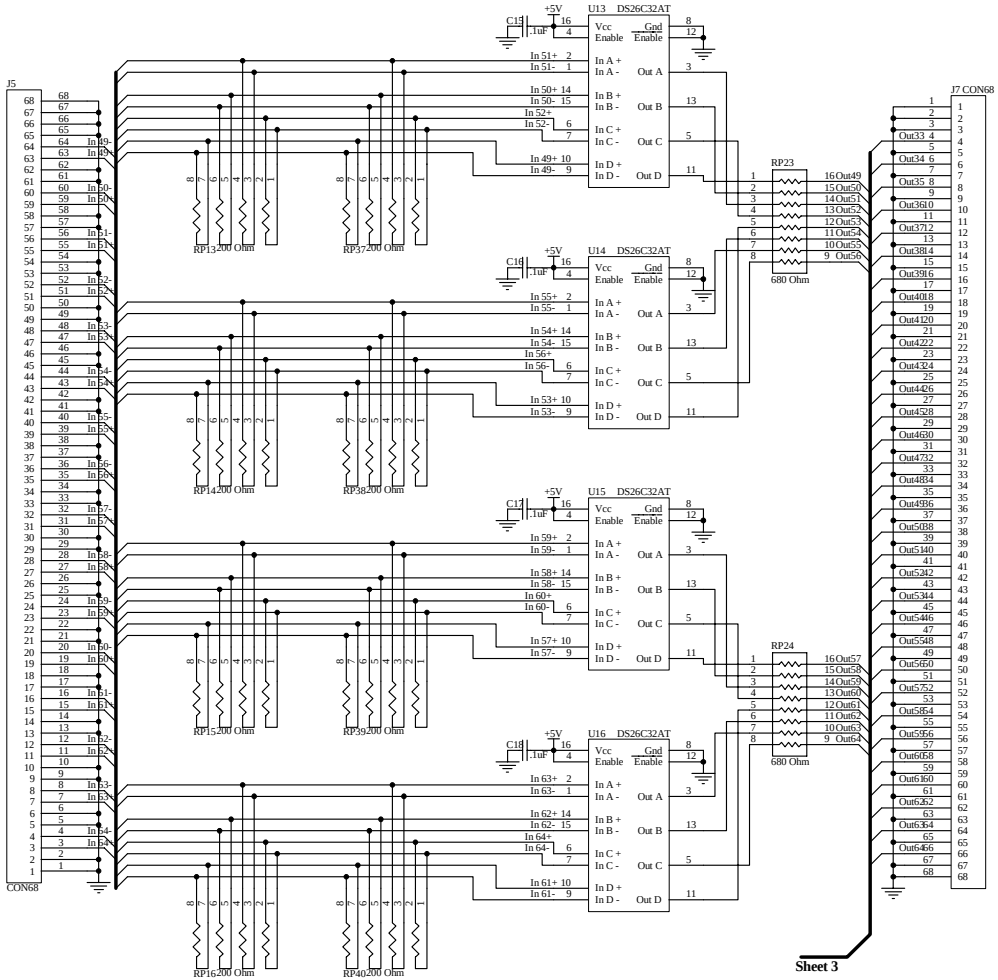
CYTEC CORP.			
APPROVED BY	DRAWING NUMBER		DRAWN BY: KJA
DATE: 4-Sep-2003	17-09-50-01		REVISION: A
SIZE: B	TITLE: 64 Channel RS-422 To CMOS/TTL Receiver		
17-09-50-01.sch			SHEET 1 OF 4



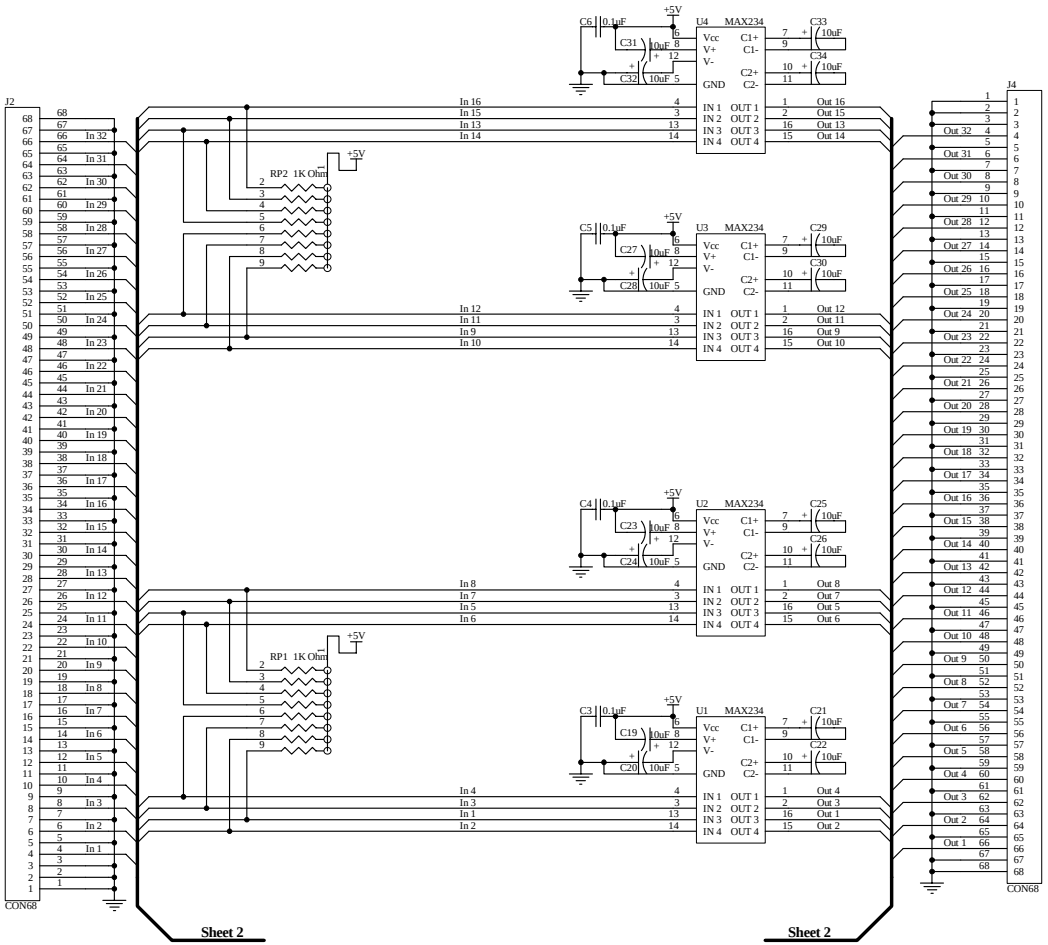
APPROVED BY		DRAWING NUMBER	DRAWN BY: KUA
DATE:	4-Sep-2003	17-09-50-02	REVISION: A
SIZE: B	TITLE: 64 Channel RS-422 To CMOS/TTL Receiver		
17-09-50-02.sch		SHEET 2 OF 4	



APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 4-Sep-2003	17-09-50-3	REVISION: A
SIZE: B	TITLE: 64 Channel RS-422 To CMOS/TTL Receiver	
17-09-50-03.sch		SHEET 3 OF 4

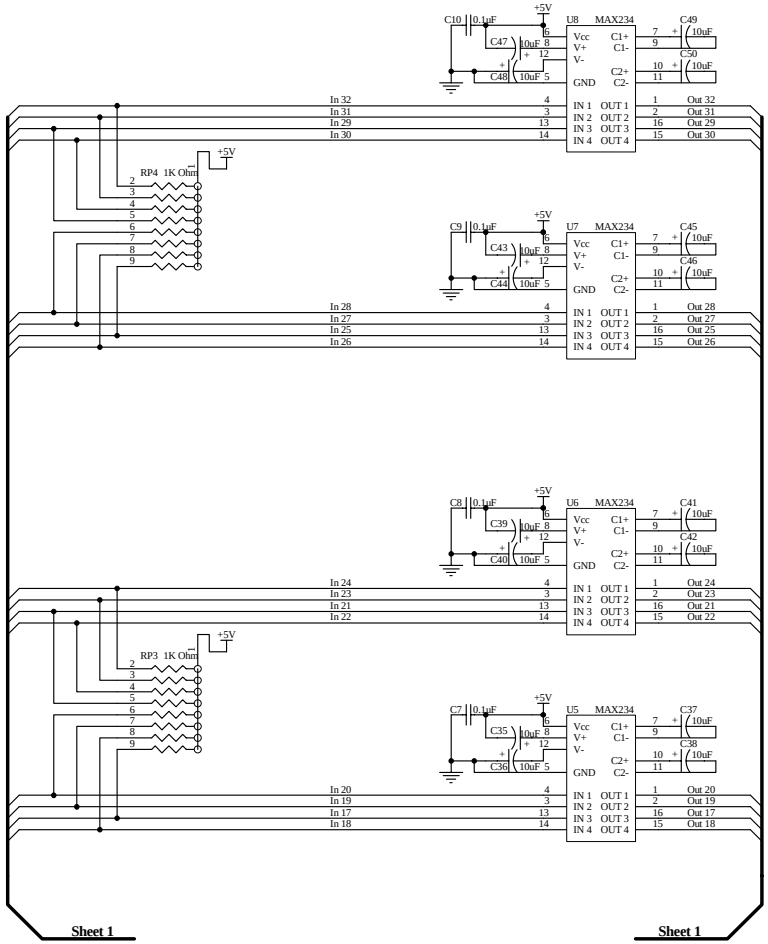


APPROVED BY		DRAWING NUMBER	DRAWN BY: KJA
DATE:	4-Sep-2003	17-09-50-4	REVISION: A
SIZE:	B	TITLE: 64 Channel RS-422 To CMOS/TTL Receiver	
17-09-50-04.sch			SHEET 4 OF 4

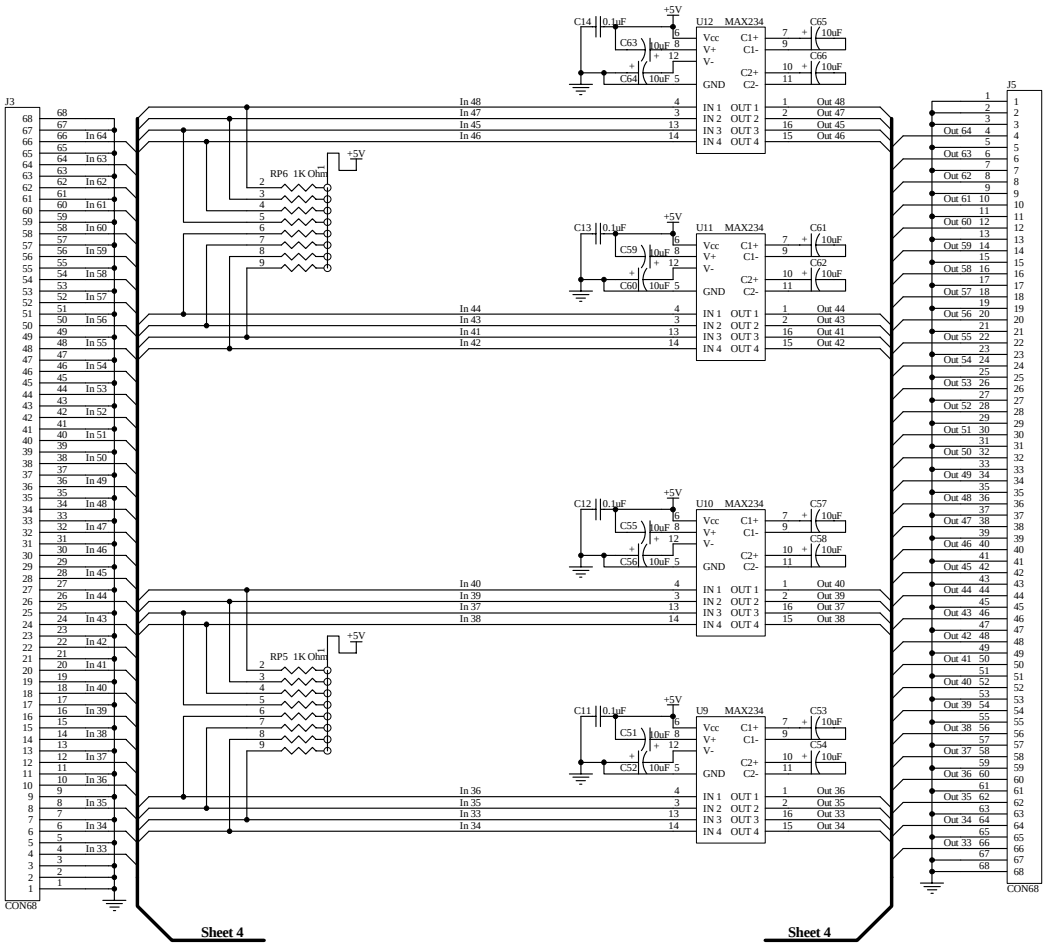


CYTEC CORP.

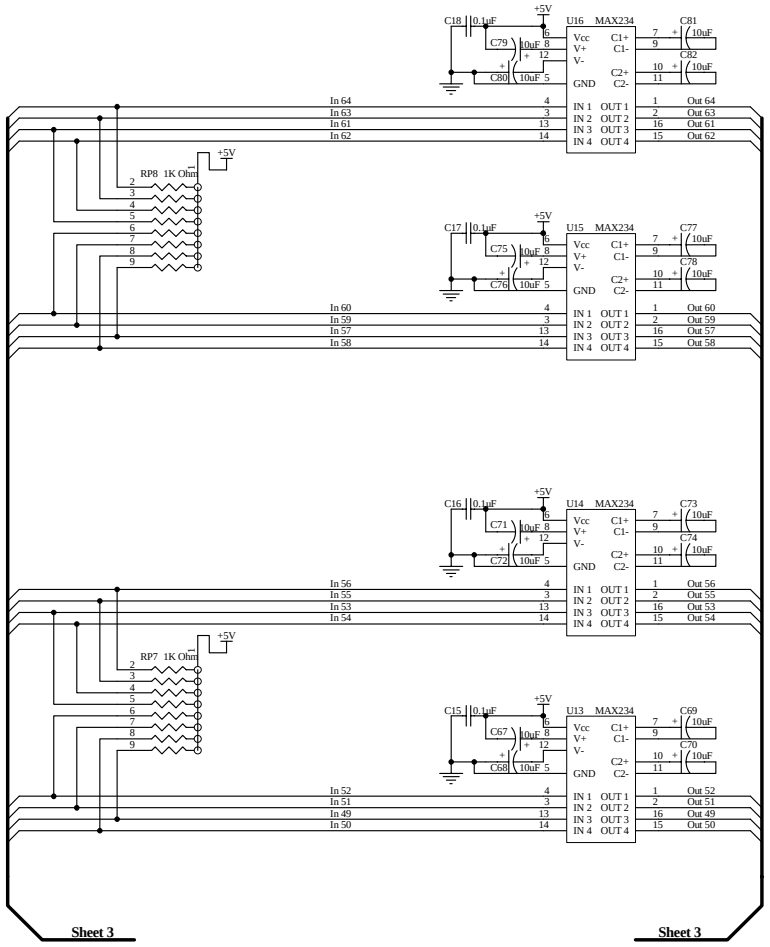
APPROVED BY	DRAWING NUMBER	DRAWN BY: KUA
DATE: 4-Sep-2003	17-12-50-01	REVISION: A
SIZE: B	TITLE: 64 Channel CMOS/TTL To RS-232 Driver	
17-12-50-01.sch		SHEET 1 OF 4



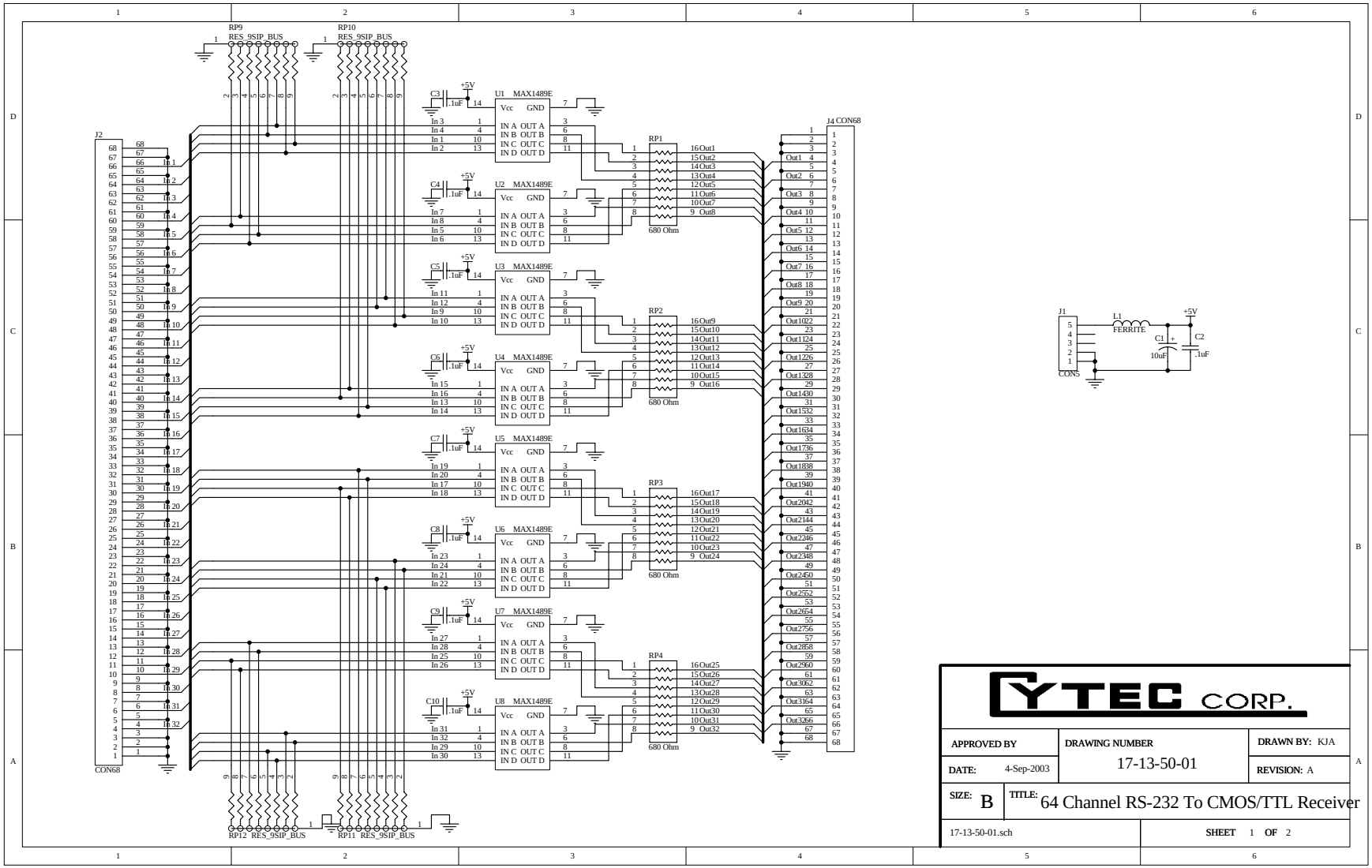
APPROVED BY	DRAWING NUMBER	DRAWN BY: KUA
DATE: 4-Sep-2003	17-12-50-02	REVISION: A
SIZE: B	TITLE: 64 Channel CMOS/TTL To RS-232 Driver	
17-12-50-02.sch		SHEET 2 OF 4



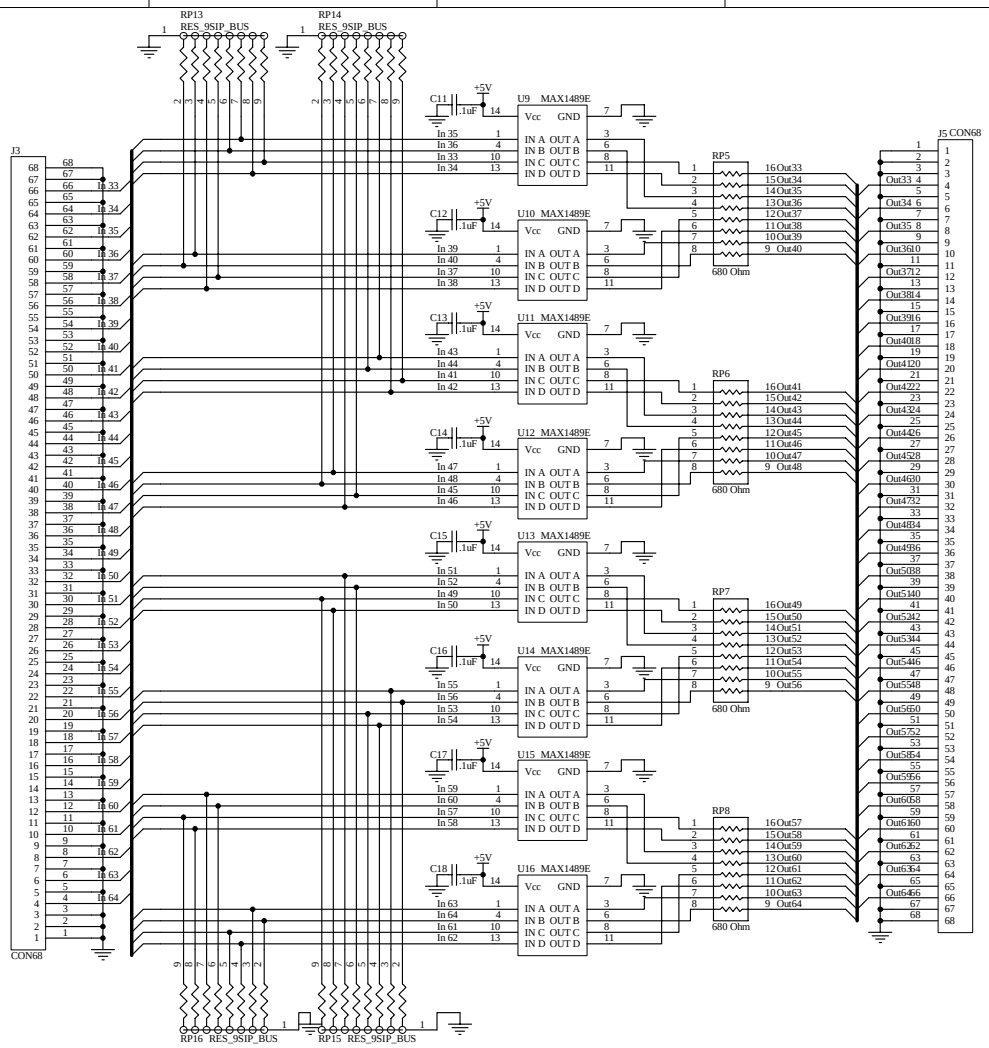
CYTEC CORP.			
APPROVED BY		DRAWING NUMBER	DRAWN BY: KUA
DATE:	4-Sep-2003	17-12-50-03	REVISION: A
SIZE: B	TITLE: 64 Channel CMOS/TTL To RS-232 Driver		
17-12-50-03.sch		SHEET 3 OF 4	



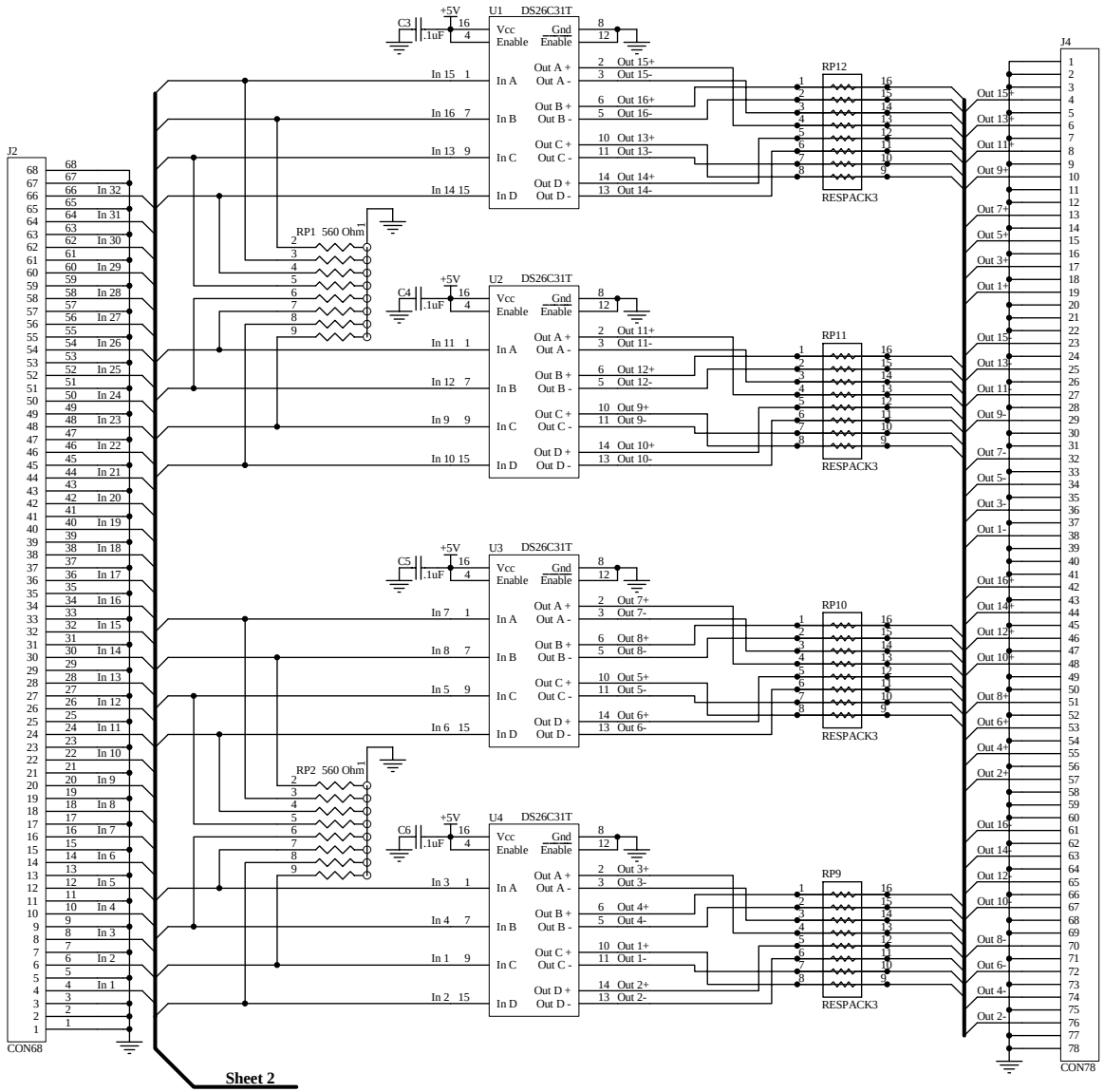
APPROVED BY	DRAWING NUMBER	DRAWN BY: KUA
DATE: 4-Sep-2003	17-12-50-04	REVISION: A
SIZE: B	TITLE: 64 Channel CMOS/TTL To RS-232 Driver	
17-12-50-04.sch		SHEET 4 OF 4



CYTEC CORP.			
APPROVED BY		DRAWING NUMBER	DRAWN BY: KJA
DATE:	4-Sep-2003	17-13-50-01	REVISION: A
SIZE: B	TITLE: 64 Channel RS-232 To CMOS/TTL Receiver		
17-13-50-01.sch		SHEET 1 OF 2	

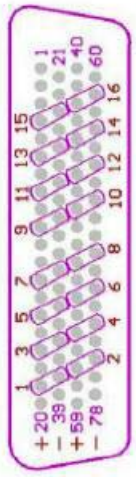


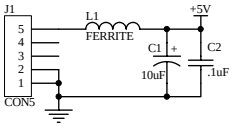
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 4-Sep-2003	17-13-50-02	REVISION: A
SIZE: B	TITLE: 64 Channel RS-232 To CMOS/TTL Receiver	
17-13-50-02.sch		SHEET 2 OF 2



D78 Pinout

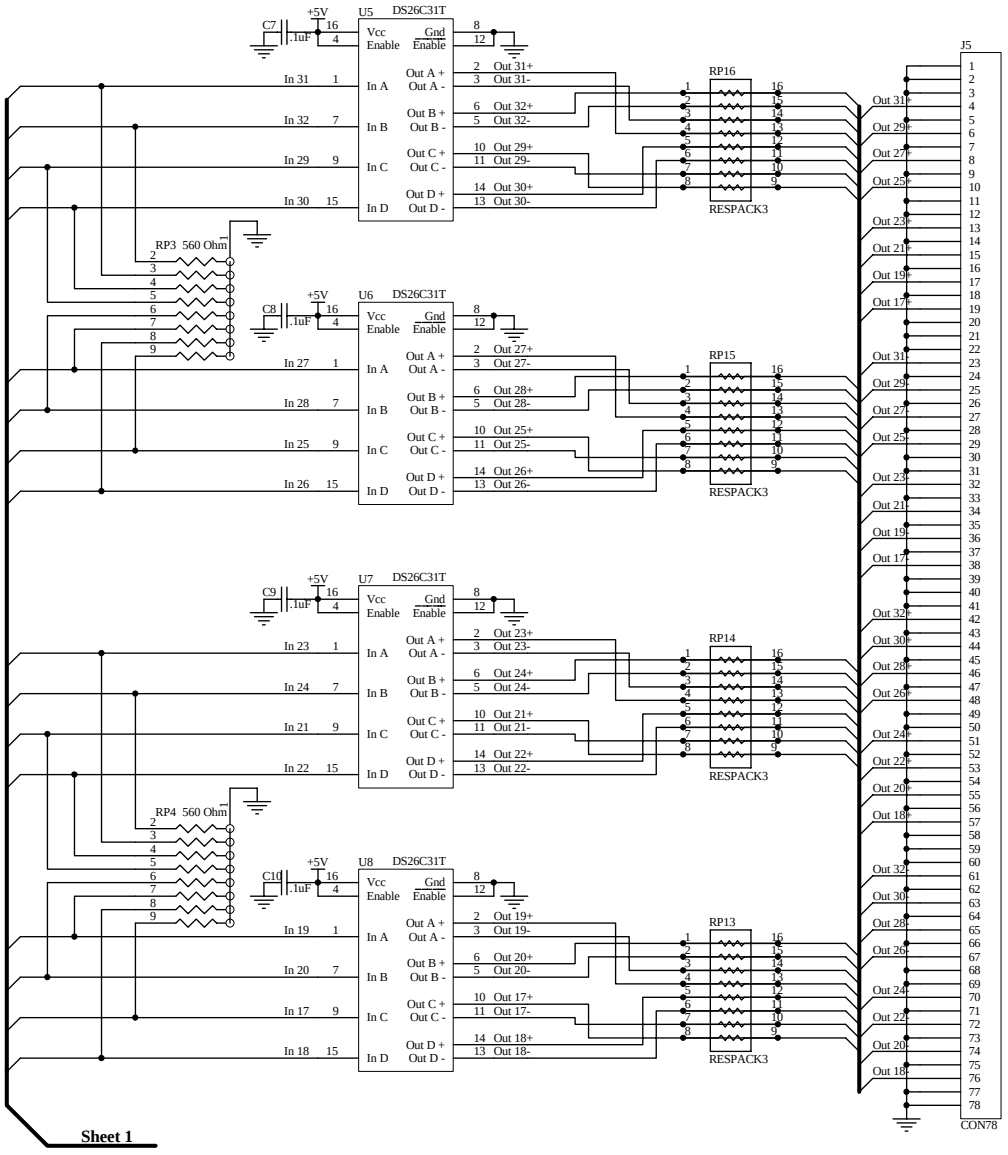
D78 Pinout





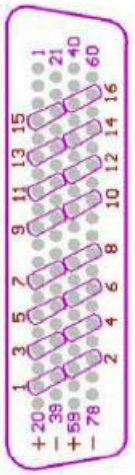
CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: KUA
DATE: 29-Jul-2010	17-16-51/01	REVISION:
SIZE: B	TITLE: DX/64-RS422-OUT D78 Output Buffer	
17-16-51-01.sch		SHEET 1 OF 4

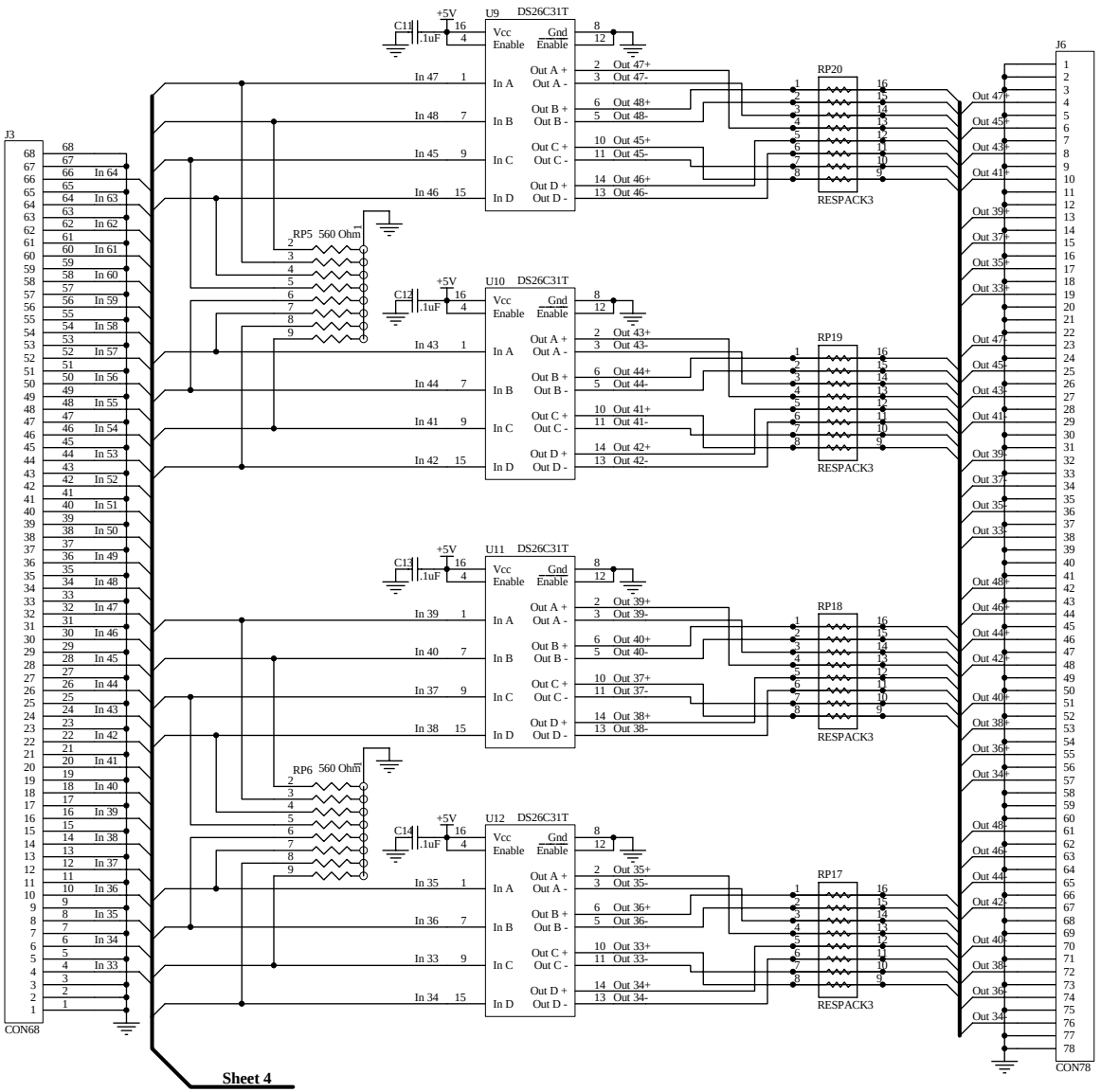


D78 Pinout

D78 Pinout

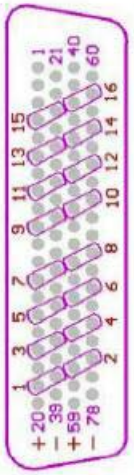


CYTEC CORP.			
APPROVED BY		DRAWING NUMBER	DRAWN BY: KUA
DATE:	29-Jul-2010	17-16-51/02	REVISION:
SIZE: B	TITLE: DX/64-RS422-OUT D78 Output Buffer		
17-16-51-02.sch		SHEET 2 OF 4	

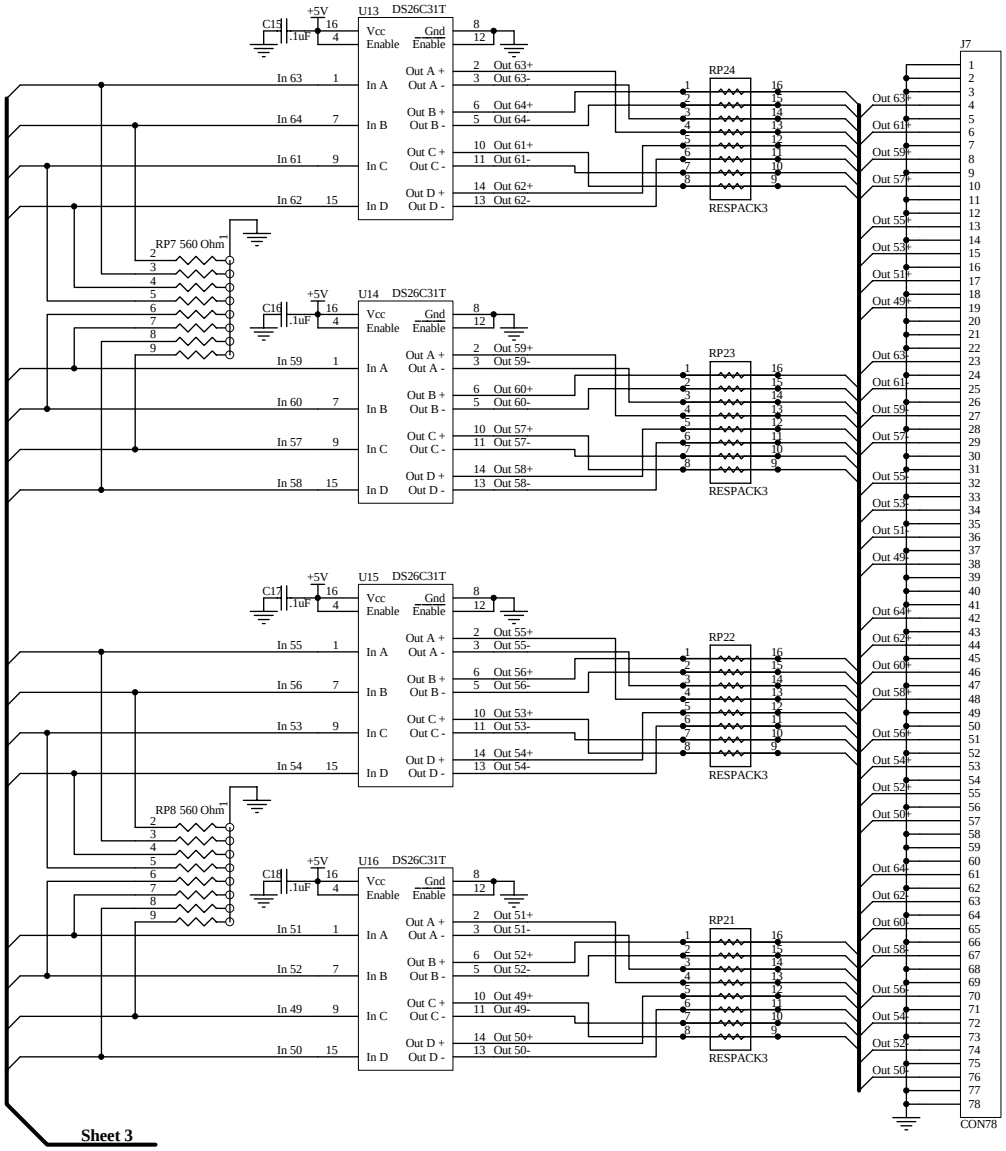


D78 Pinout

D78 Pinout

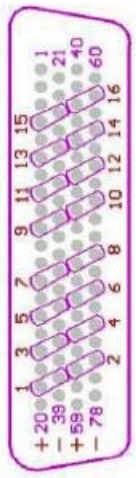


CYTEC CORP.			
APPROVED BY	DRAWING NUMBER		DRAWN BY: KJA
DATE: 29-Jul-2010	17-16-51/03		REVISION:
SIZE: B	TITLE: DX/64-RS422-OUT D78 Output Buffer		
17-16-51-03.sch		SHEET 3 OF 4	



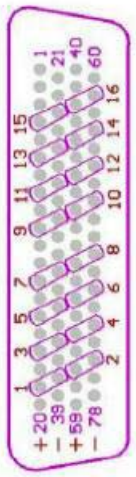
D78 Pinout

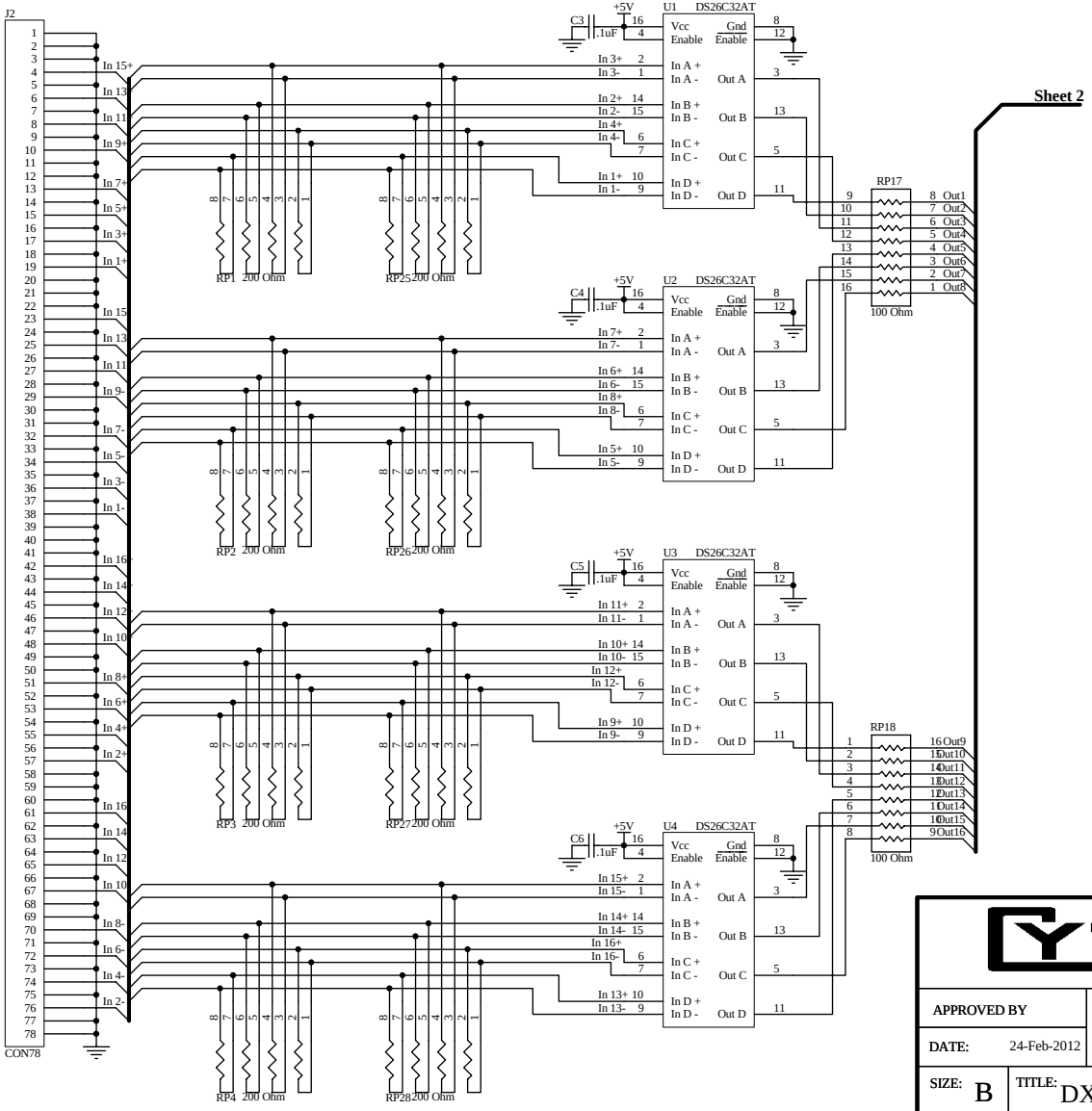
D78 Pinout

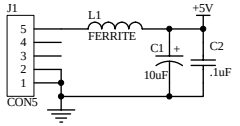


CYTEC CORP.		
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 29-Jul-2010	17-16-51/04	REVISION:
SIZE: B	TITLE: DX/64-RS422-OUT D78 Output Buffer	
17-16-51-04.sch		SHEET 4 OF 4

D78 Pinout



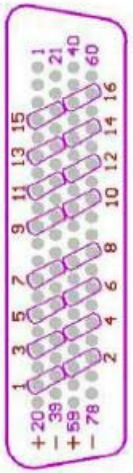


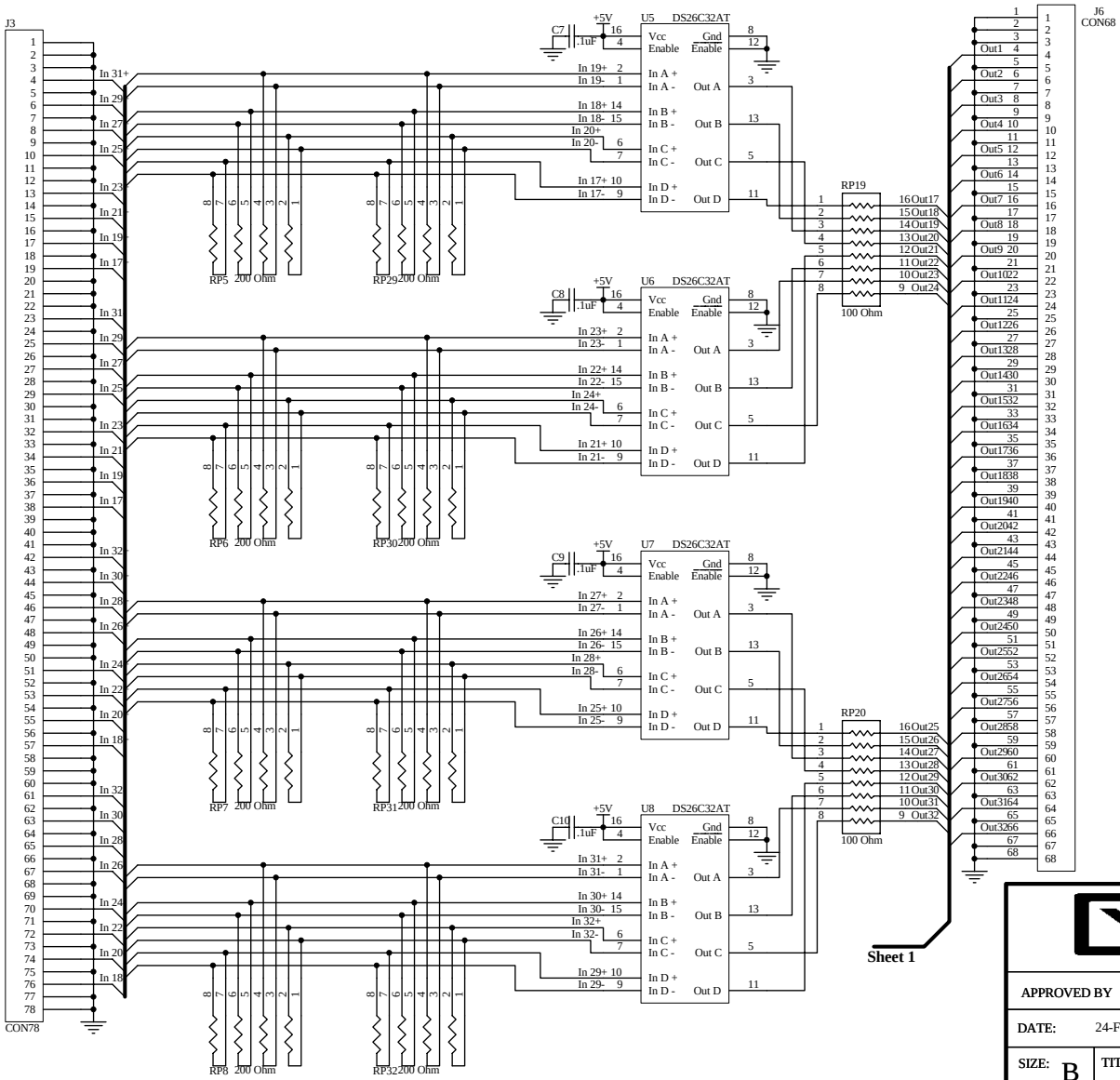


CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 24-Feb-2012	17-17-51-1/01	REVISION:
SIZE: B	TITLE: DX/64-RS422-IN D78 Input Buffer	
17-17-51-1-01.sch		SHEET 1 OF 4

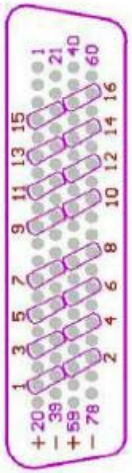
D78 Pinout

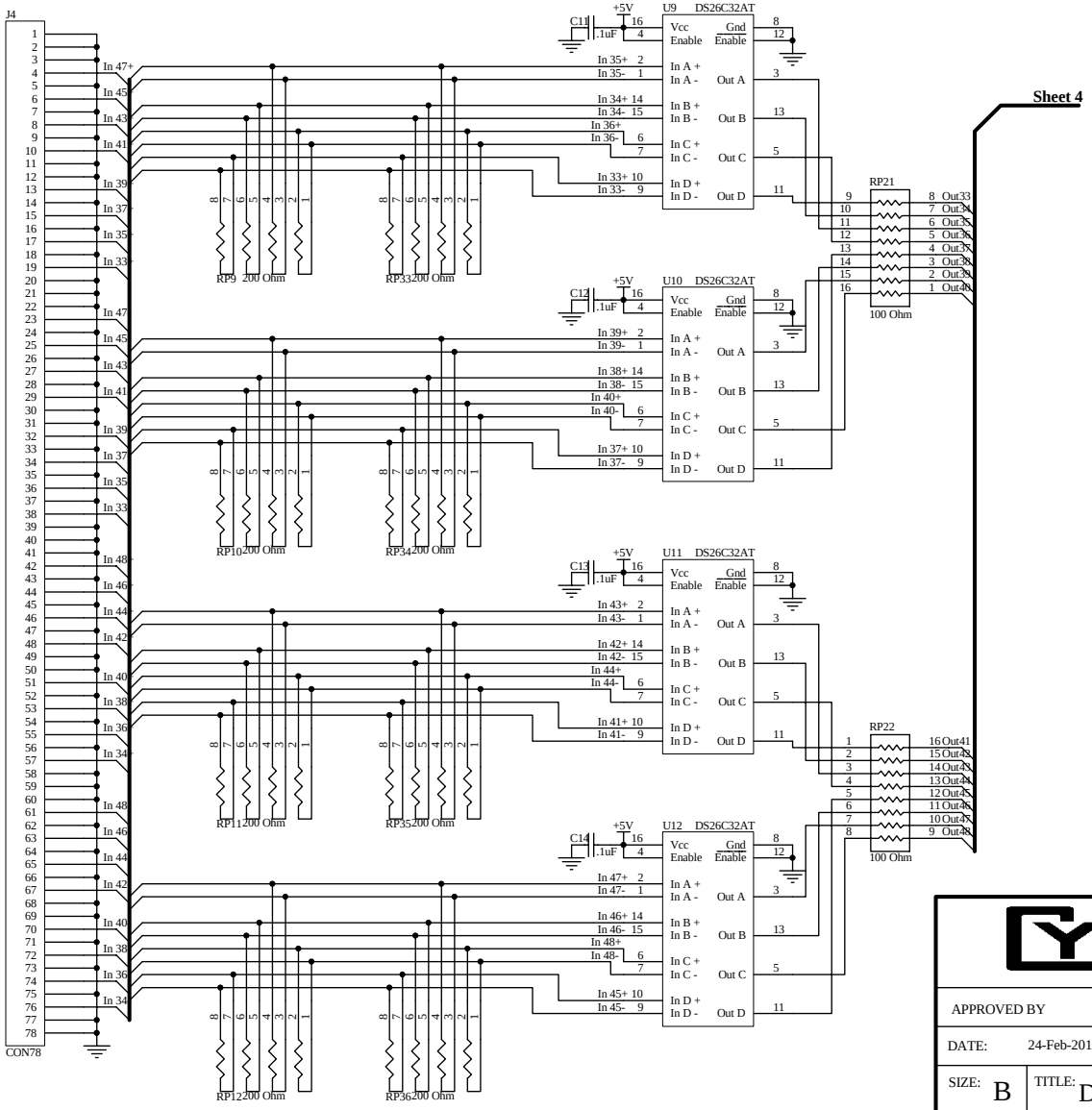




CYTEC CORP.			
APPROVED BY		DRAWING NUMBER	DRAWN BY: KUA
DATE:	24-Feb-2012	17-17-51-1/02	REVISION:
SIZE: B	TITLE: DX/64-RS422-IN D78 Input Buffer		
17-17-51-1-02.sch		SHEET 2 OF 4	

D78 Pinout

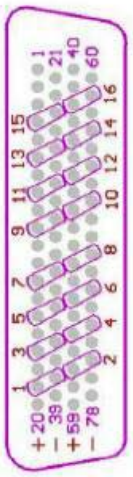




CYTEC CORP.

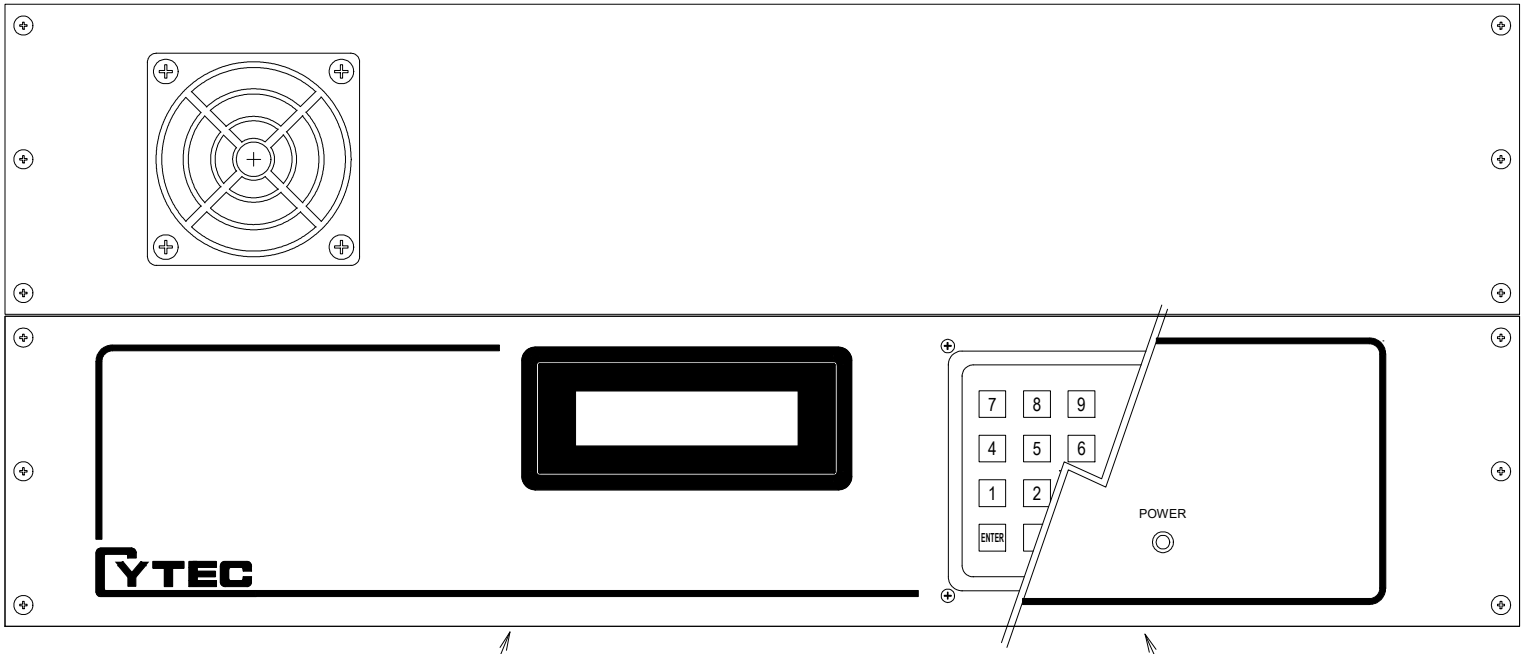
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 24-Feb-2012	17-17-51-1/03	REVISION:
SIZE: B	TITLE: DX/64-RS422-IN D78 Input Buffer	
17-17-51-1-03.sch		SHEET 3 OF 4

D78 Pinout



Sheet 3

APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 24-Feb-2012	17-17-51-1/04	REVISION:
SIZE: B	TITLE: DX/64-RS422-IN D78 Input Buffer	
17-17-51-1-04.sch		SHEET 4 OF 4



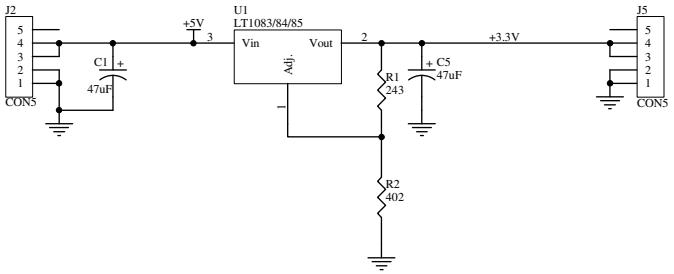
OPTIONAL 3.5" LCD KEYPAD FRONT PANEL
(REPLACES STANDARD PANEL)

STANDARD 3.5" FRONT PANEL

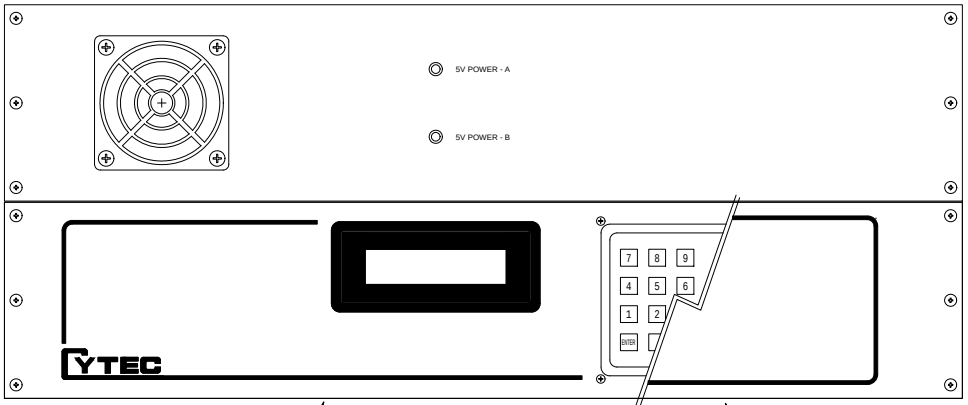
YTEC CORP.

TOL. +/- .010" U.O.S.

DATE: 5/8/03	CAD FILE: 48-06-22.dw2	DRAWN BY: DR	
SIZE: C	TITLE: DX/256x256 FRONT PANEL ASSY		
COMMENTS:		DRAWING NUMBER: 48-06-22	

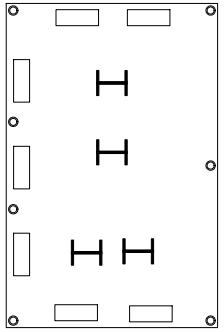


CYTEC CORP.		
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 9-Jul-2021	48-06-50	REVISION:
SIZE: B	TITLE: 5V To 3.3V Voltage Regulator Board	
48-06-50.sch		SHEET 1 OF 1

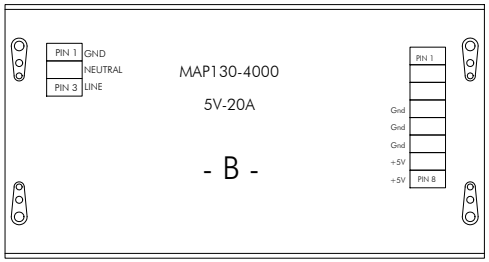


OPTIONAL 3.5" LCD KEYPAD FRONT PANEL
(REPLACES STANDARD PANEL)

STANDARD 3.5" FRONT PANEL



REDUNDANT SUPPLY



* SET BOTH SUPPLIES AT 5.5V *

REPLACE 48-06-72 FRONT PANEL w/ 48-07-72 SHOWN

ADD 2nd MAP130-4000 (5V-20A) POWER SUPPLY

ADD 46-06-11 P.S. MANAGEMENT BOARD

-SEE 48-07-140 B.O.M. FOR PARTS LIST

-SEE 48-07-30 FOR WIRING CHANGES

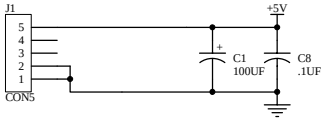
C	11/19/12	DR	CHANGED TO MAP130-4000
			(availability issues)
A	9/4/03	DR	WAS MAP130-1005 P.S.

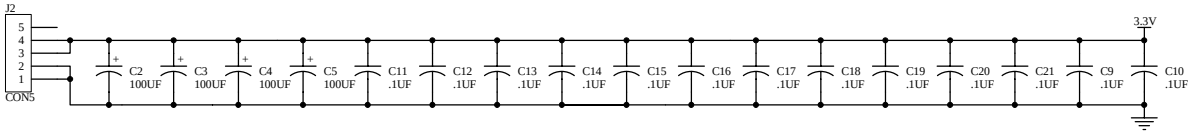
CYTEC CORP. TOL. +/- .010" U.O.S.

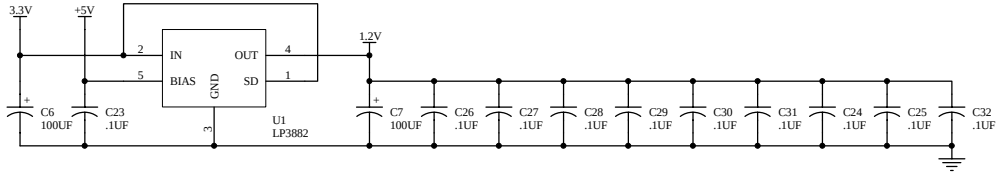
DATE: 7/2/03 | CAD FILE: 48-07-00.dw2 | DRAWN BY: DR

SIZE: D | TITLE: DX/256x256 REDUNDANT P.S. OPTION

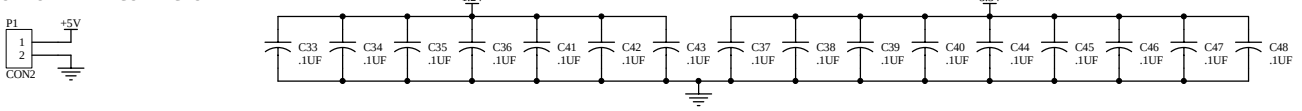
COMMENTS: | DRAWING NUMBER: 48-07-00







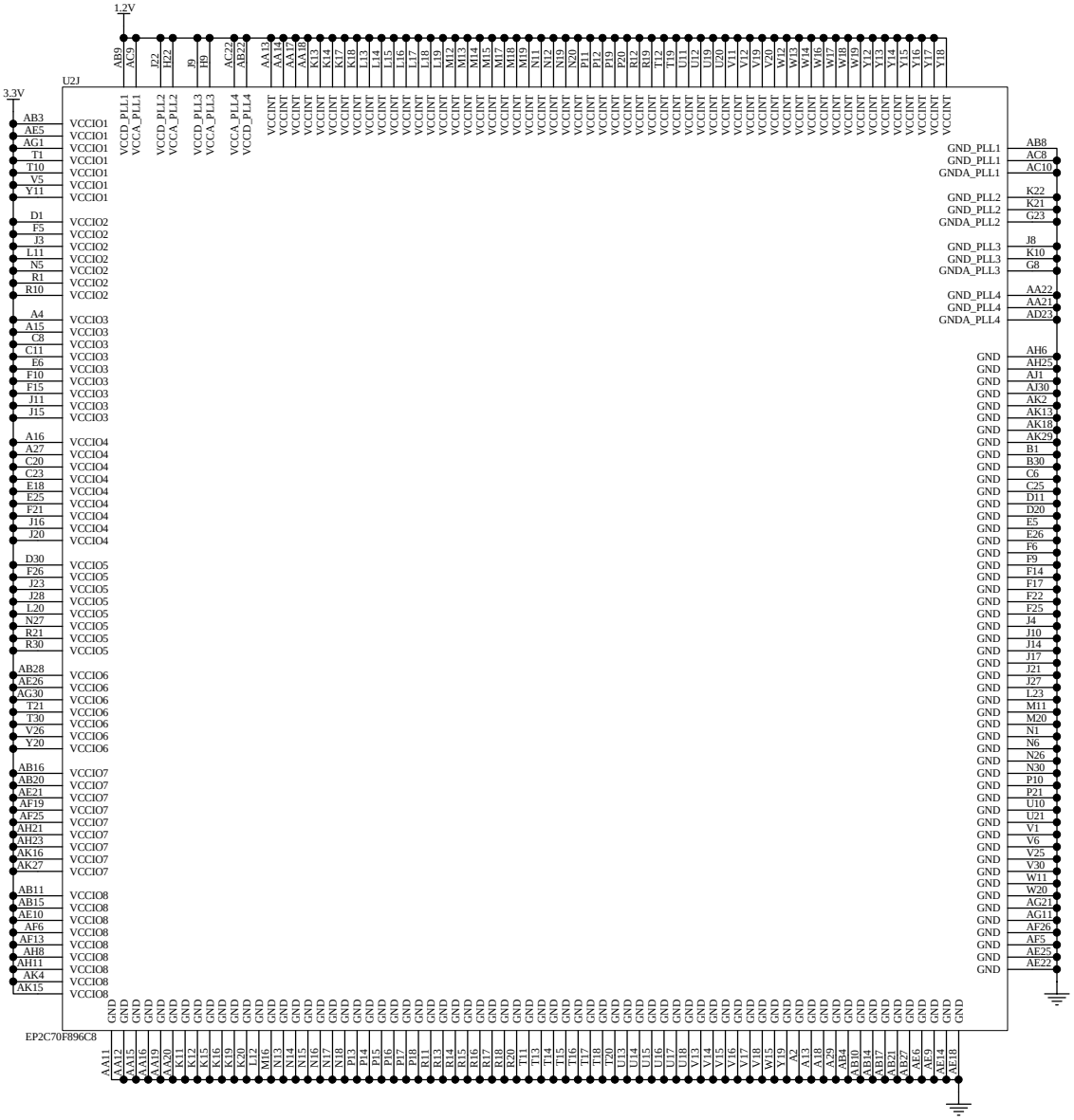
OPTIONAL FAN CONNECTOR



REVISED FROM:

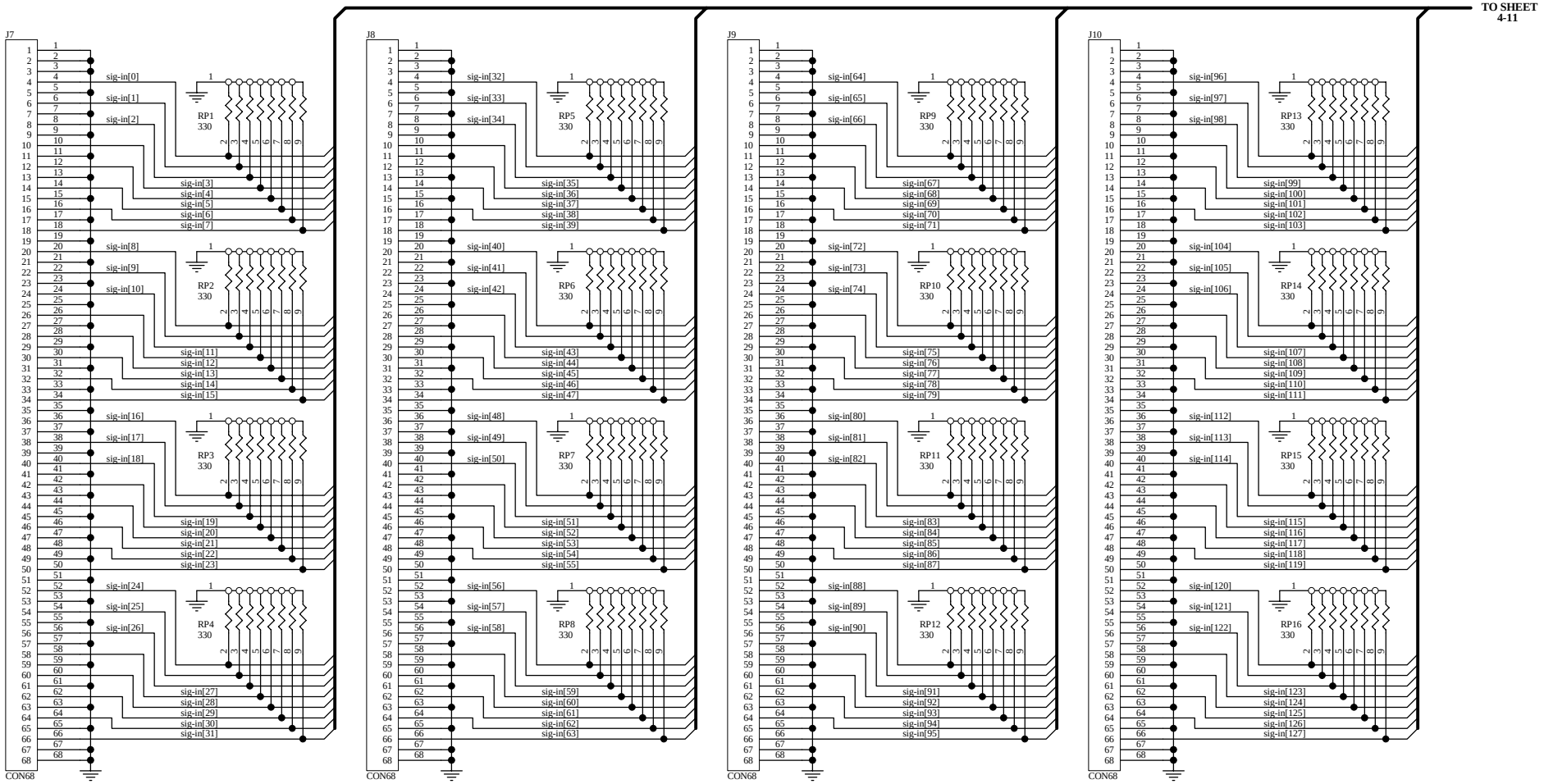
CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/1	REVISION:
SIZE: A3	TITLE: DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/1		SHEET 1 OF 11



REVISED FROM:

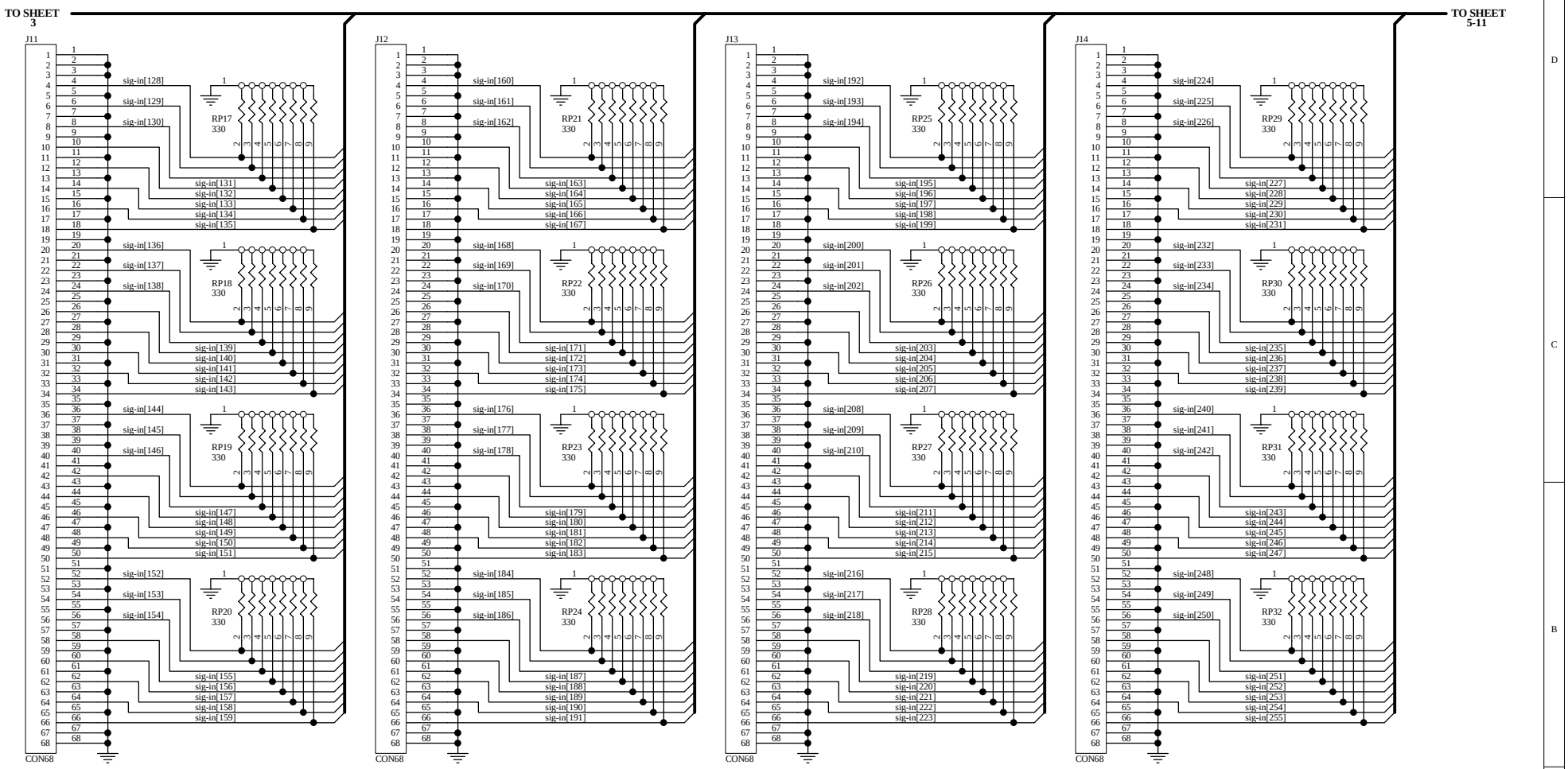
APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/2	REVISION:
SIZE: A3	TITLE: DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/2		SHEET 2 OF 11



REVISED FROM:

CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/3	REVISION:
SIZE: A3	**TITLE:** DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/3		SHEET 3 OF 11



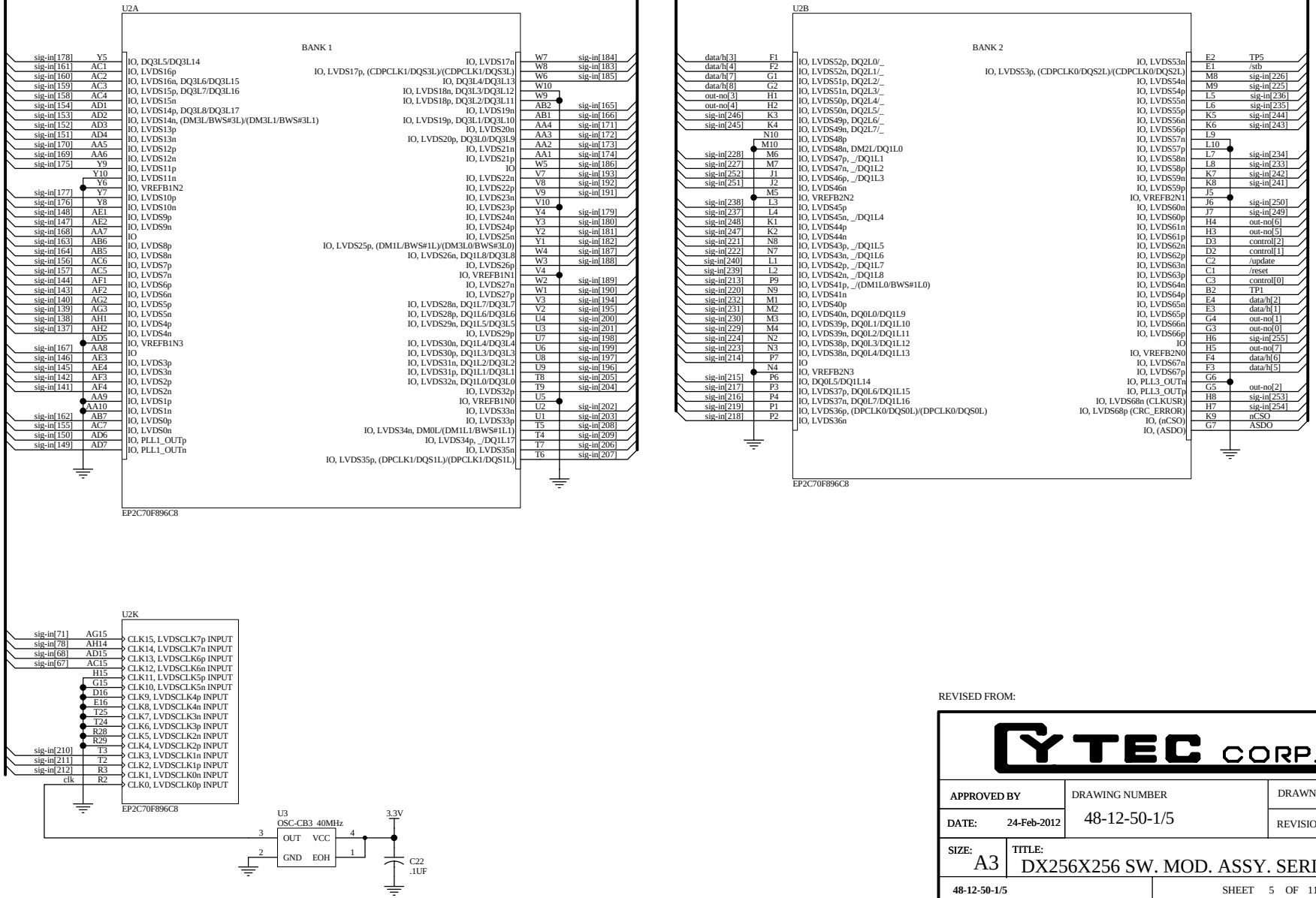
REVISED FROM:

CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/4	REVISION:
SIZE: A3	TITLE: DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/4		SHEET 4 OF 11

TO SHEET 3,4

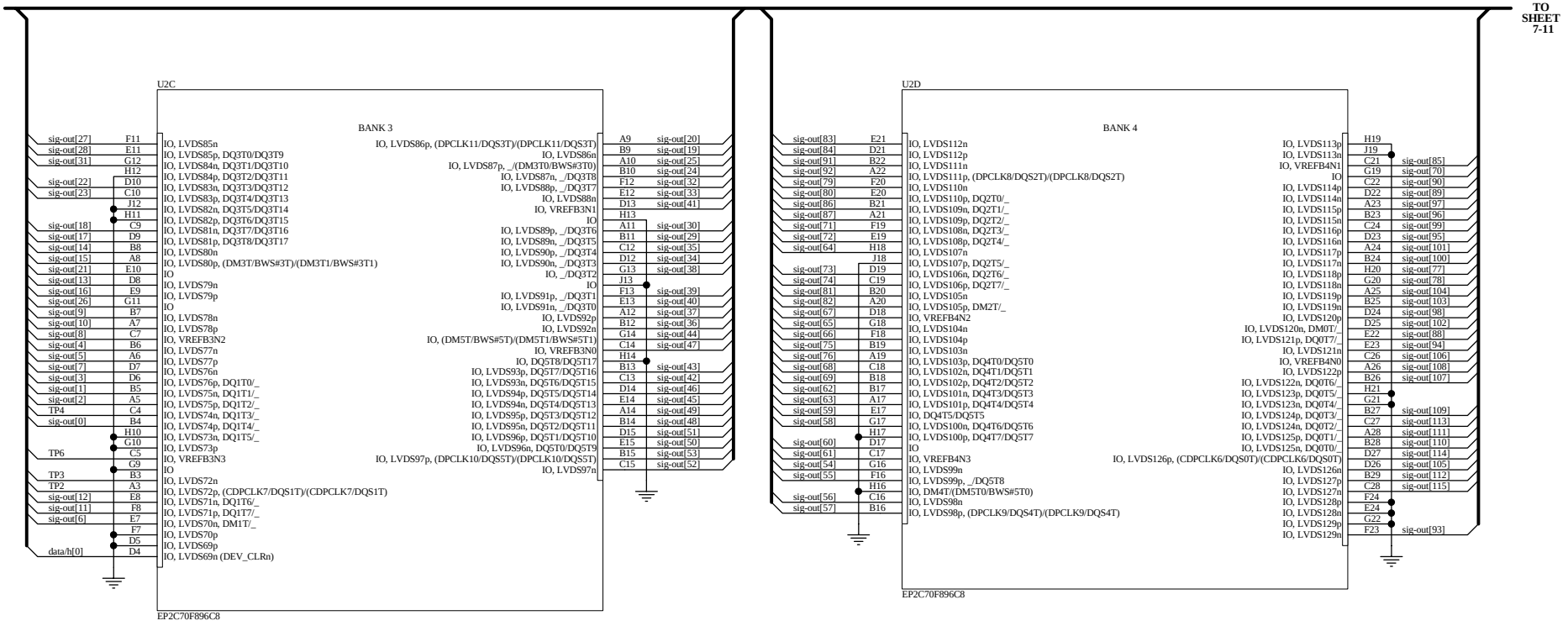
TO SHEET 6-11



REVISED FROM:

CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/5	REVISION:
SIZE: A3	TITLE: DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/5		SHEET 5 OF 11



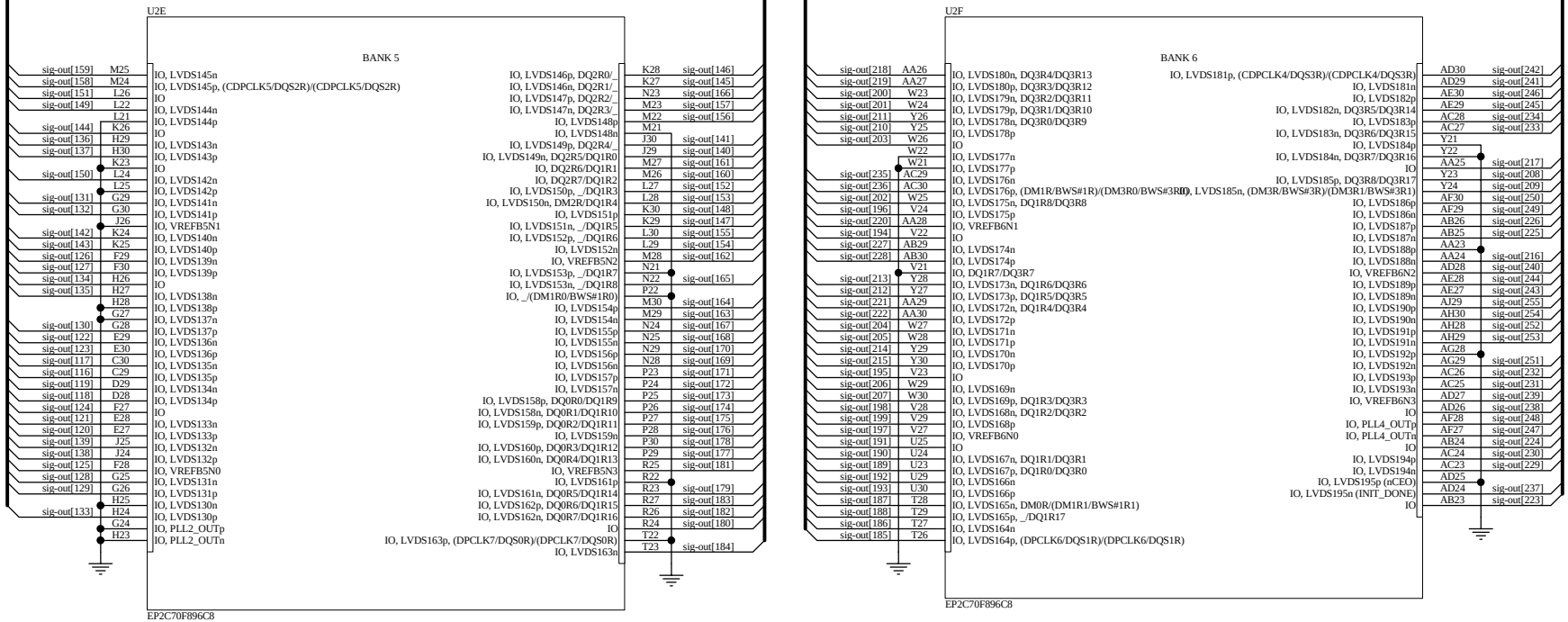
REVISED FROM:

CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/6	REVISION:
SIZE: A3	TITLE: DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/6		SHEET 6 OF 11

TO SHEET 3-6

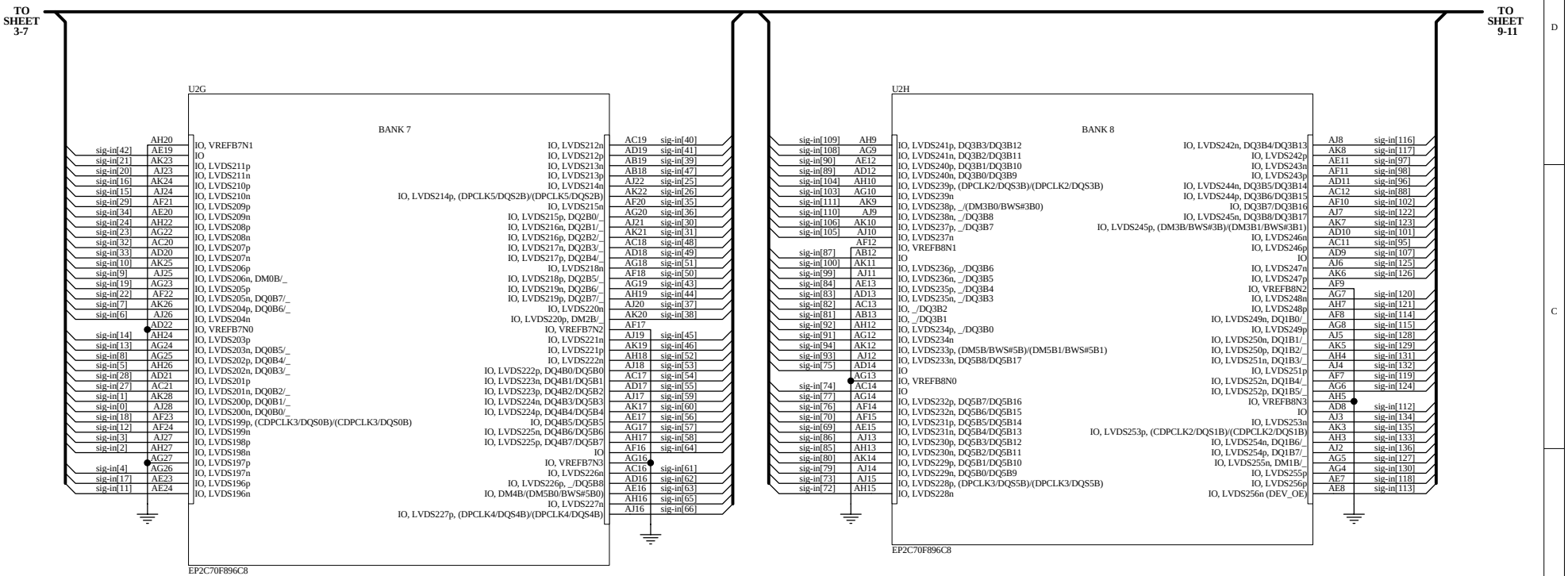
TO SHEET 8-11



REVISED FROM:

CYTEC CORP.

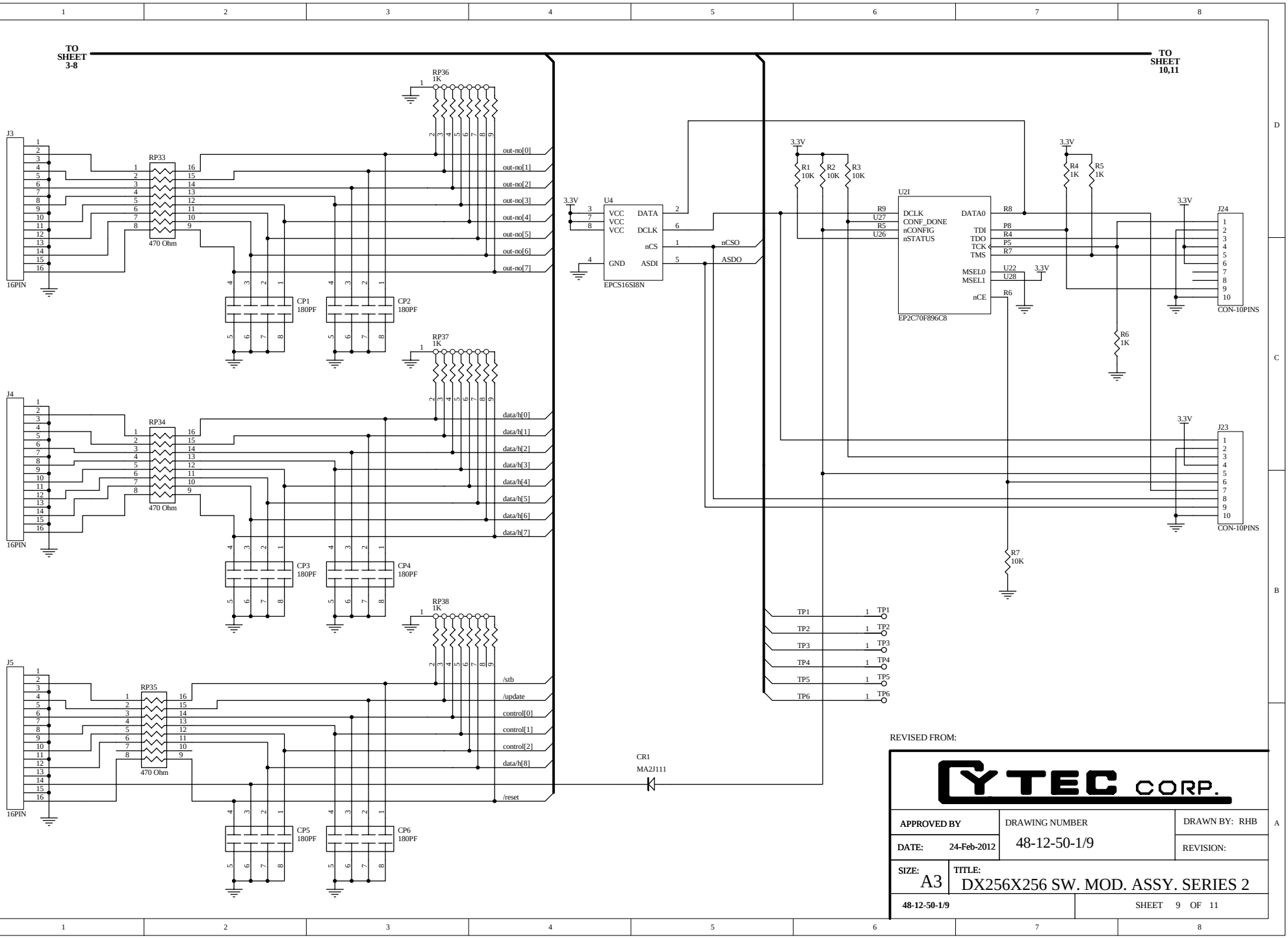
APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/7	REVISION:
SIZE: A3	TITLE: DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/7		SHEET 7 OF 11



REVISED FROM:

CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/8	REVISION:
SIZE: A3	TITLE: DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/8		SHEET 8 OF 11



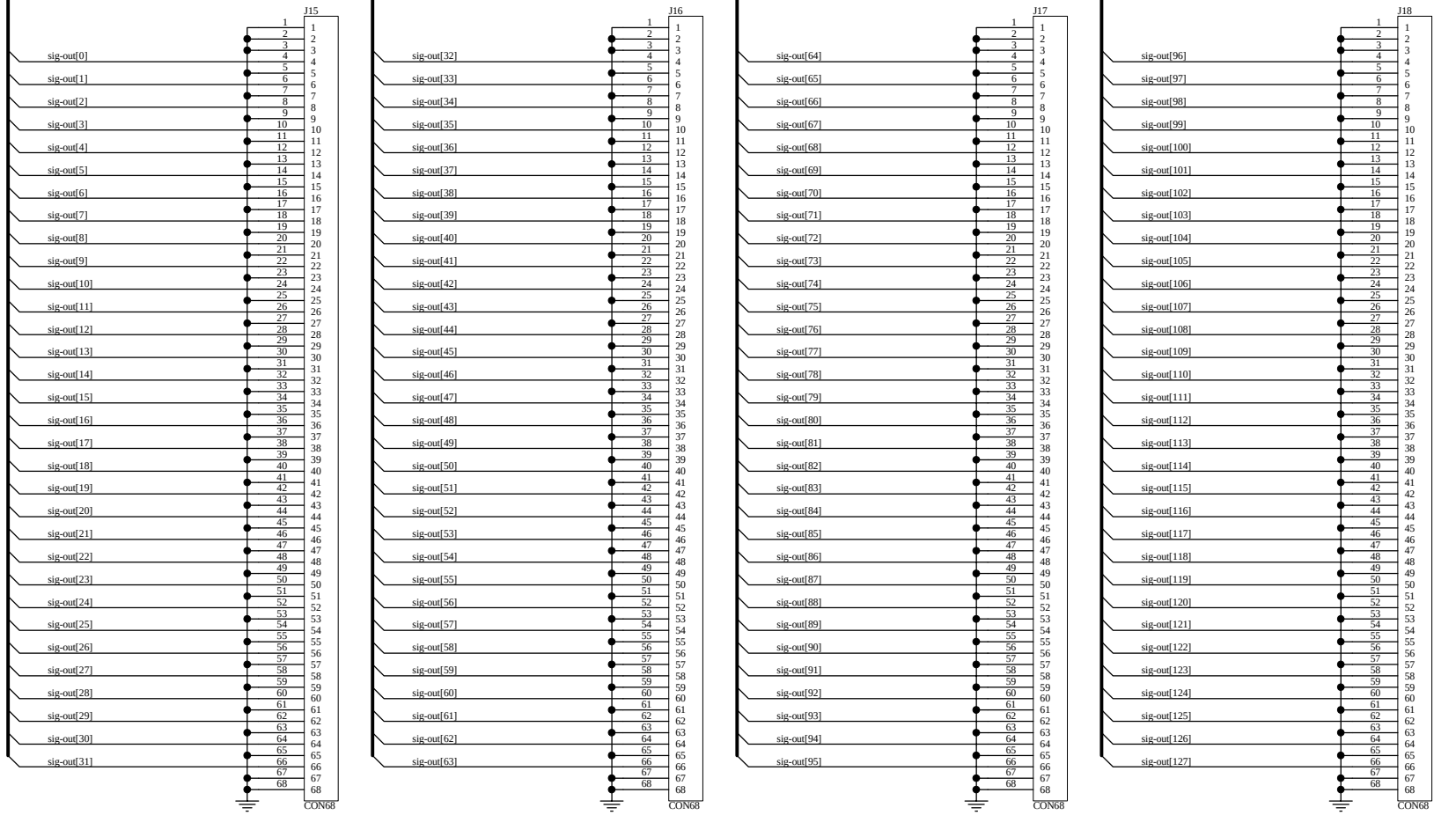
REVISED FROM:

CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/9	REVISION:
SIZE: A3	TITLE: DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/9		SHEET 9 OF 11

TO SHEET 3-9

TO SHEET 11

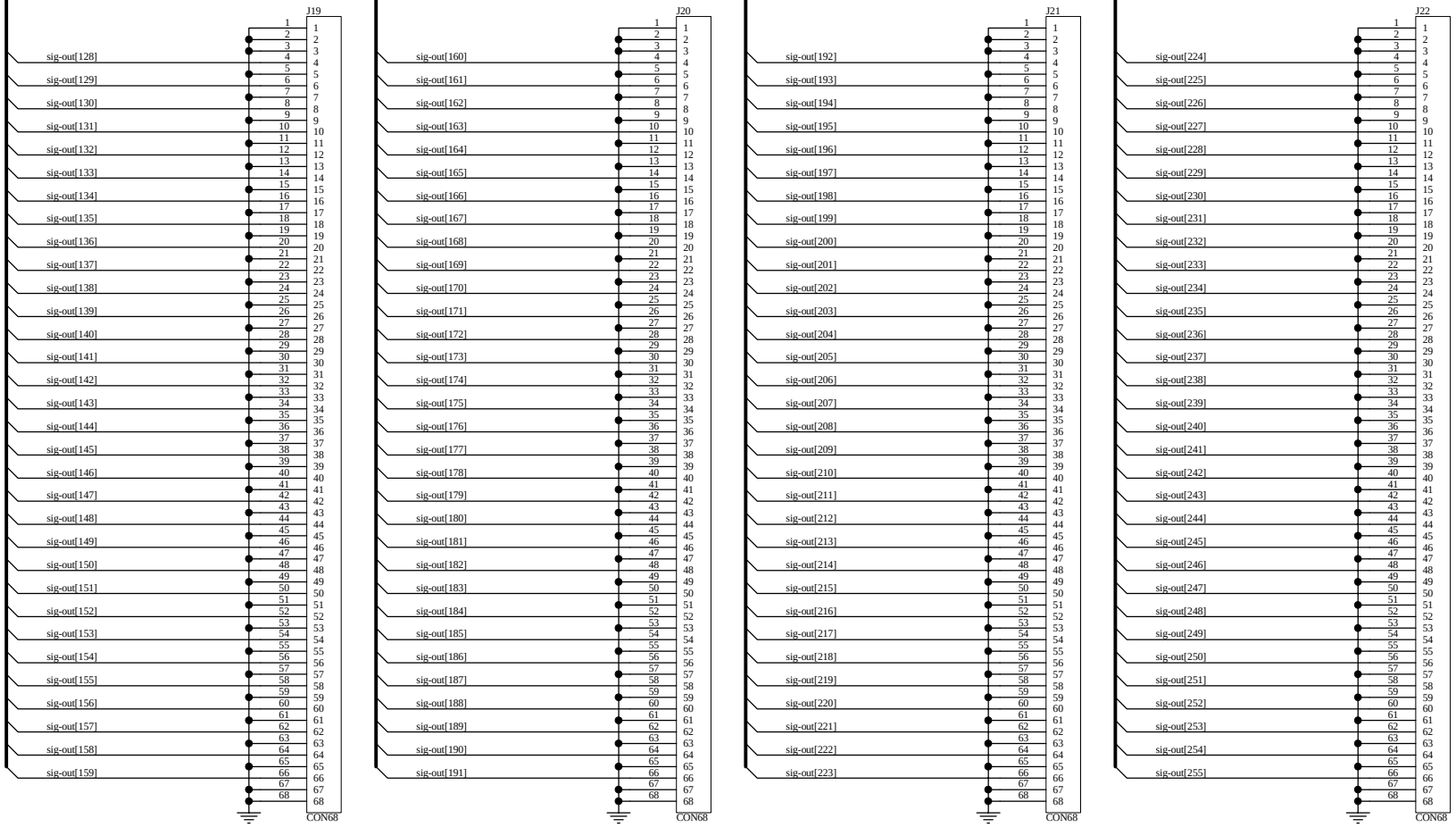


REVISED FROM:

CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/10	REVISION:
SIZE: A3	**TITLE:** DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/10		SHEET 10 OF 11

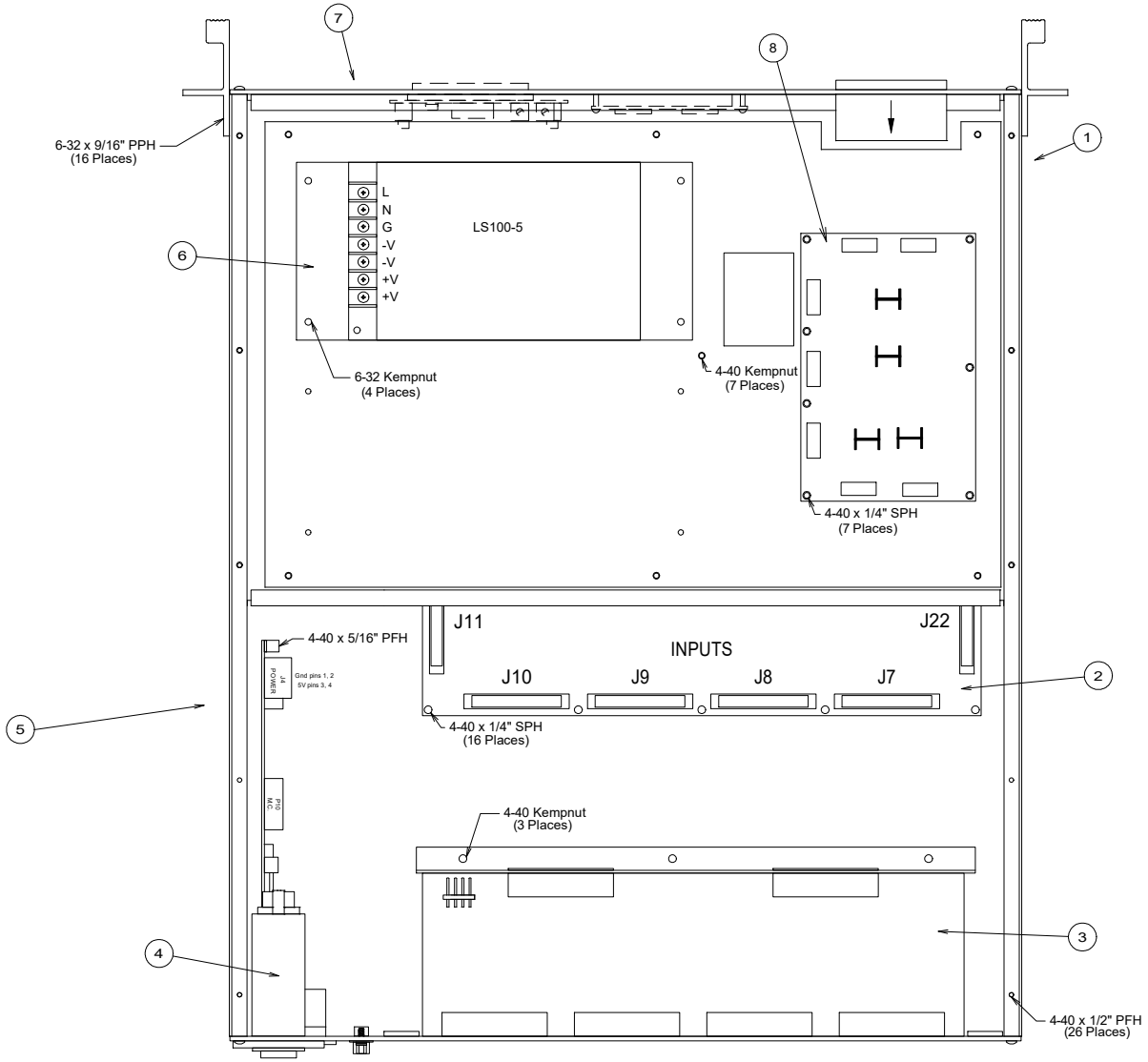
TO SHEET
3-10



REVISED FROM:

CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: RHB
DATE: 24-Feb-2012	48-12-50-1/11	REVISION:
SIZE: A3	**TITLE:** DX256X256 SW. MOD. ASSY. SERIES 2	
48-12-50-1/11		SHEET 11 OF 11



1. 14-30-20 x 4U CHASSIS
   20" DEEP x 19" WIDE x 7" HIGH
2. DX/256x256 SWITCH MOD
3. INPUT / OUTPUT MODS
4. AC SELECTABLE POWER ENTRY
5. CONTROL INTERFACE PER ORDER
   IF-11 (11-14-XX)
6. LS100-5V POWER SUPPLY
7. FRONT PANEL w/ FAN & FILTER GUARD
   & OPTIONAL LCD KEYPAD MANUAL CONTROL
8. 48-06-10 UNIVERSAL VOLTAGE ADAPTOR

Cytec P/N	Description	Qty
FAS1022	4-40 x 1/4" Slotted Panhead	23
FAS1006	4-40 x 1/2" Phillips Flathead	26
FAS1010	4-40 x 5/16" Phillips Flathead	1
FAS1013	4-40 Kempnut	7
FAS1012	6-32 Kempnut	4
FAS1007	6-32 x 9/16" Phillips Panhead	16

A	11/19/19	PV	Changed to LS Power Supply

CYTEC CORP.

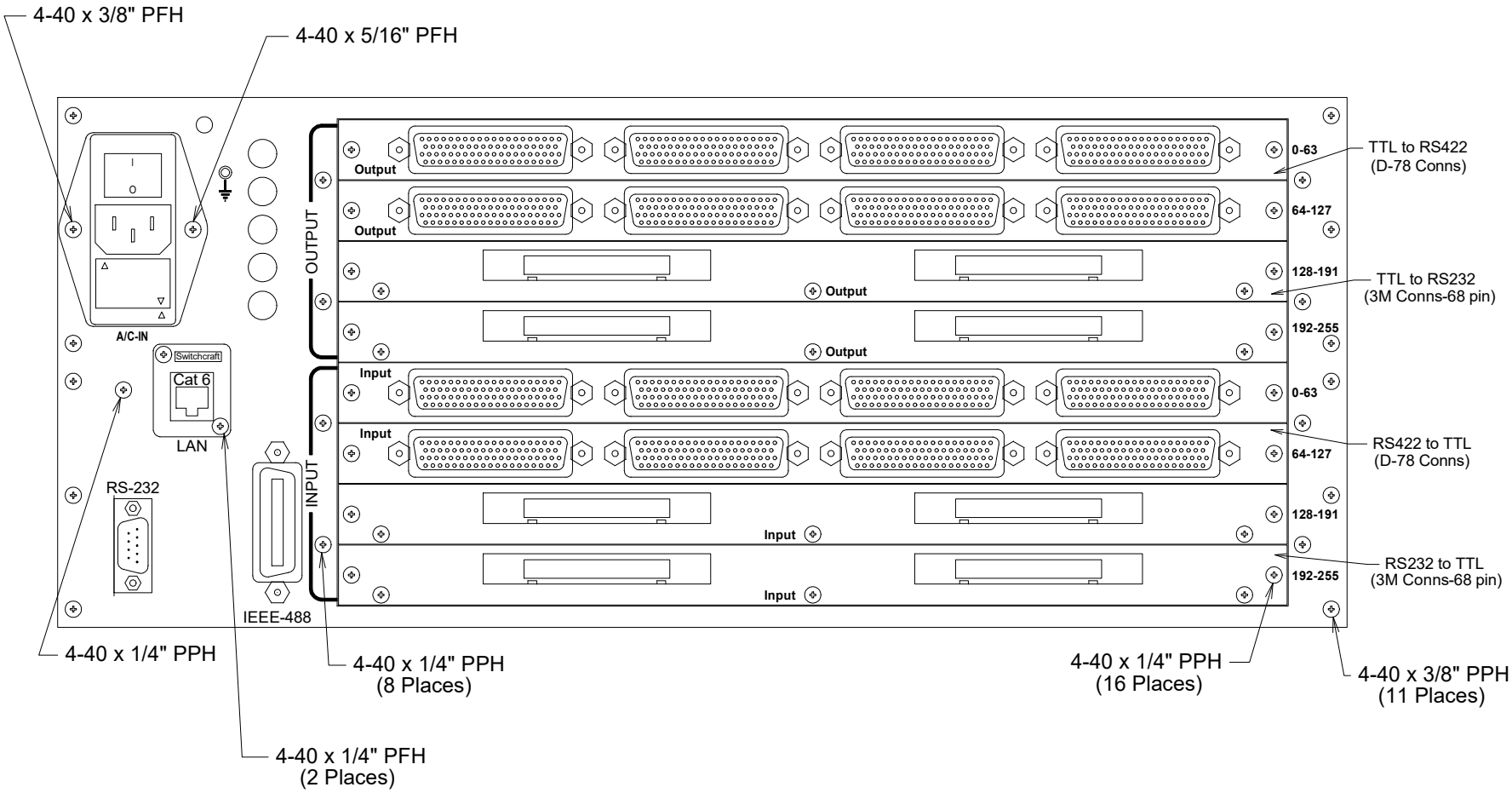
TOL. +/- .010" U.O.S.

DATE: 9/25/17 | CAD FILE: 48-34-00.dcd | DRAWN BY: PV

SIZE: D | TITLE: DX/256x256 CHASSIS

COMMENTS:

DRAWING NUMBER: 48-34-00



Cytec P/N	Description	Qty
FAS1014	4-40 x 3/8" Phillips Panhead	11
FAS1017	4-40 x 1/4" Phillips Panhead	19
FAS1010	4-40 x 5/16" Phillips Flathead	1
FAS1081	4-40 x 3/8" Phillips Flathead	1

DEEP ETCH & CLEAR ANODIZE

A	9/22/17	PV	Drawing Start

CYTEC CORP.

TOL. +/- .010" U.O.S.

DATE: 9/22/17 | CAD FILE: 48-34-21.dcd | DRAWN BY: PV

SIZE: C | TITLE: DX/256x256 Rear Panel

COMMENTS:

DRAWING NUMBER: 48-34-21

Control to Switch Mod

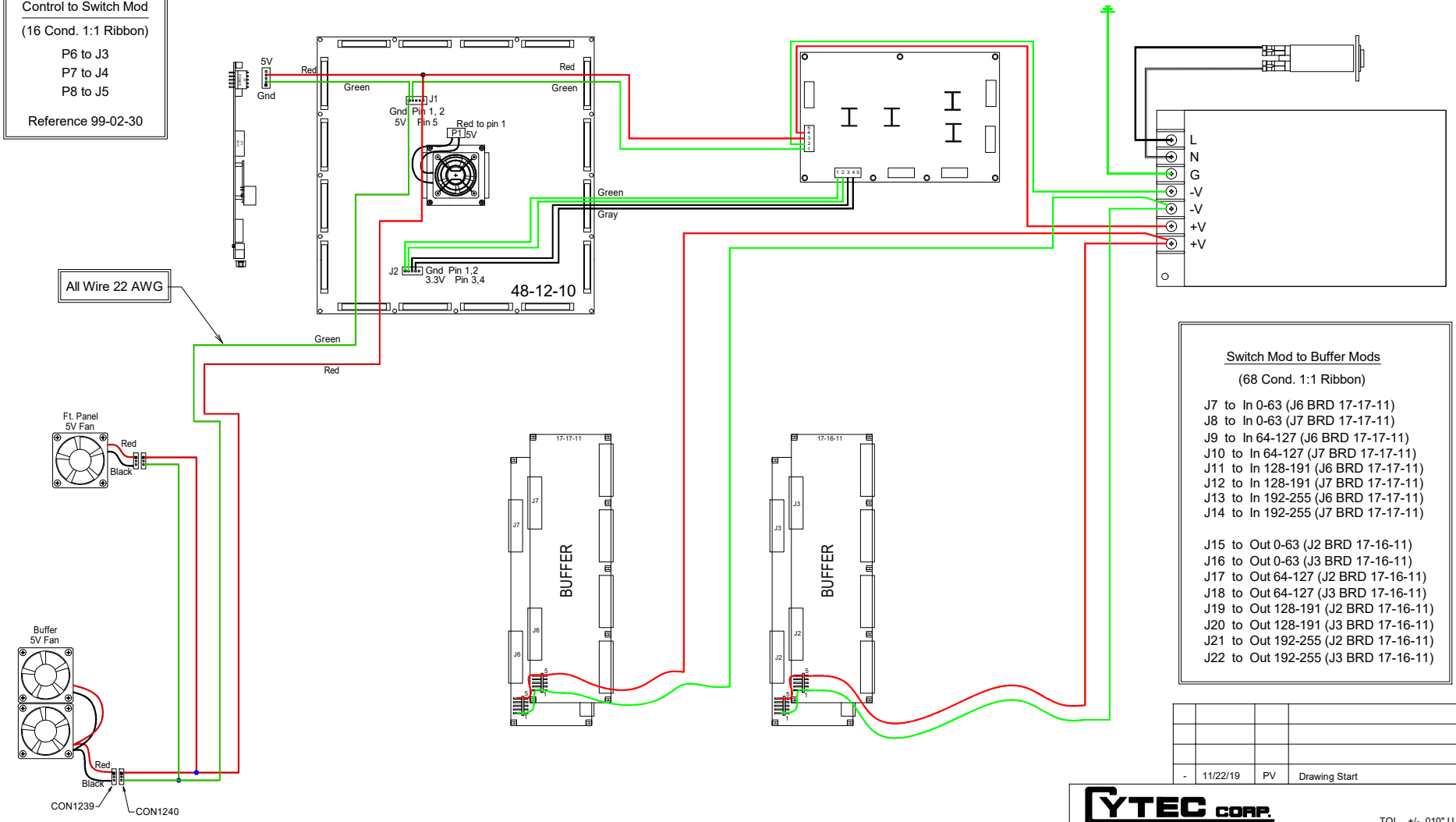
(16 Cond. 1:1 Ribbon)

P6 to J3

P7 to J4

P8 to J5

Reference 99-02-30



Switch Mod to Buffer Mods

(68 Cond. 1:1 Ribbon)

J7 to In 0-63 (J6 BRD 17-17-11)
J8 to In 0-63 (J7 BRD 17-17-11)
J9 to In 64-127 (J6 BRD 17-17-11)
J10 to In 64-127 (J7 BRD 17-17-11)
J11 to In 128-191 (J6 BRD 17-17-11)
J12 to In 128-191 (J7 BRD 17-17-11)
J13 to In 192-255 (J6 BRD 17-17-11)
J14 to In 192-255 (J7 BRD 17-17-11)

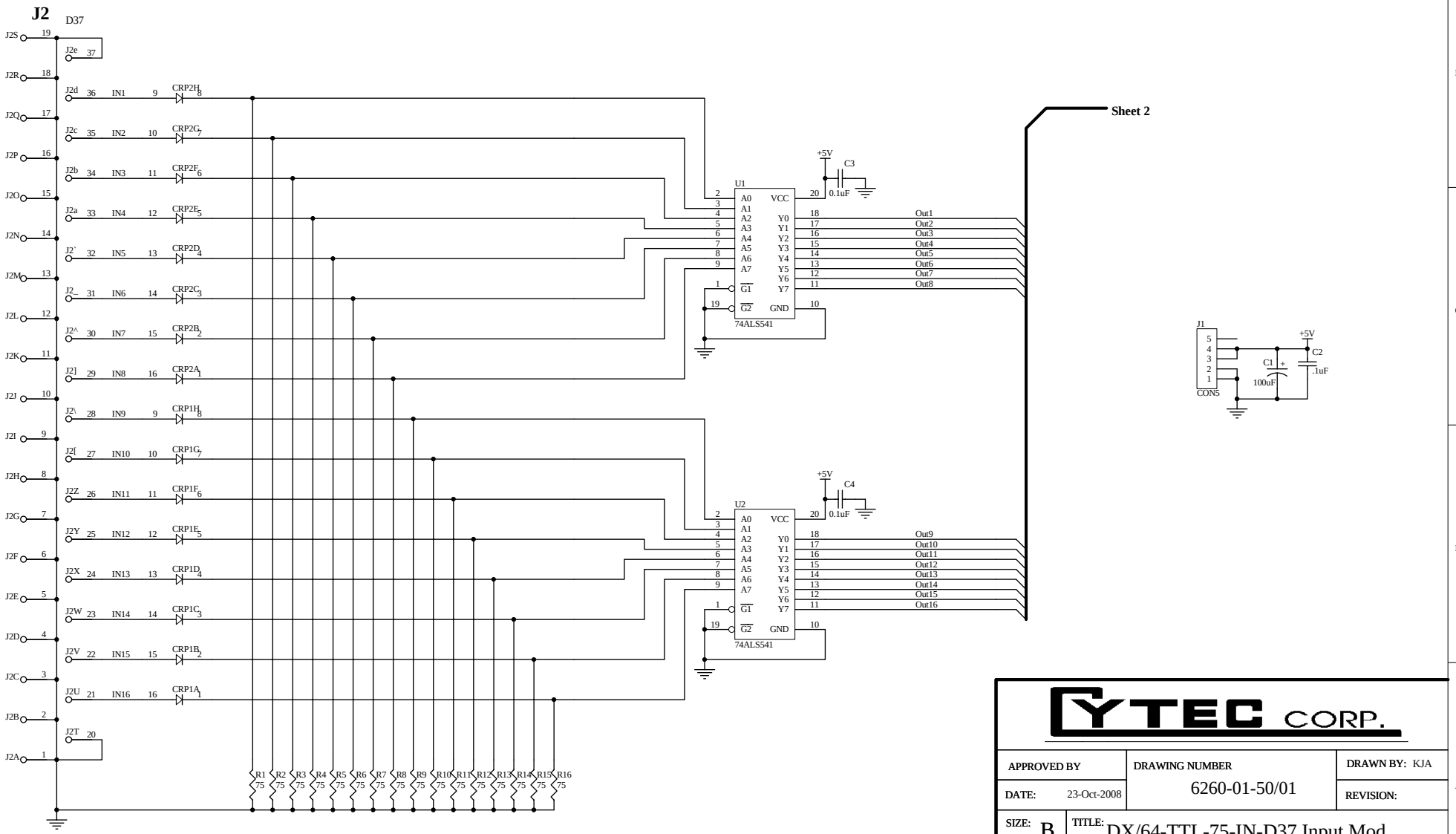
J15 to Out 0-63 (J2 BRD 17-16-11)
J16 to Out 0-63 (J3 BRD 17-16-11)
J17 to Out 64-127 (J2 BRD 17-16-11)
J18 to Out 64-127 (J3 BRD 17-16-11)
J19 to Out 128-191 (J2 BRD 17-16-11)
J20 to Out 128-191 (J3 BRD 17-16-11)
J21 to Out 192-255 (J2 BRD 17-16-11)
J22 to Out 192-255 (J3 BRD 17-16-11)

-	11/22/19	PV	Drawing Start

CYTEC CORP.

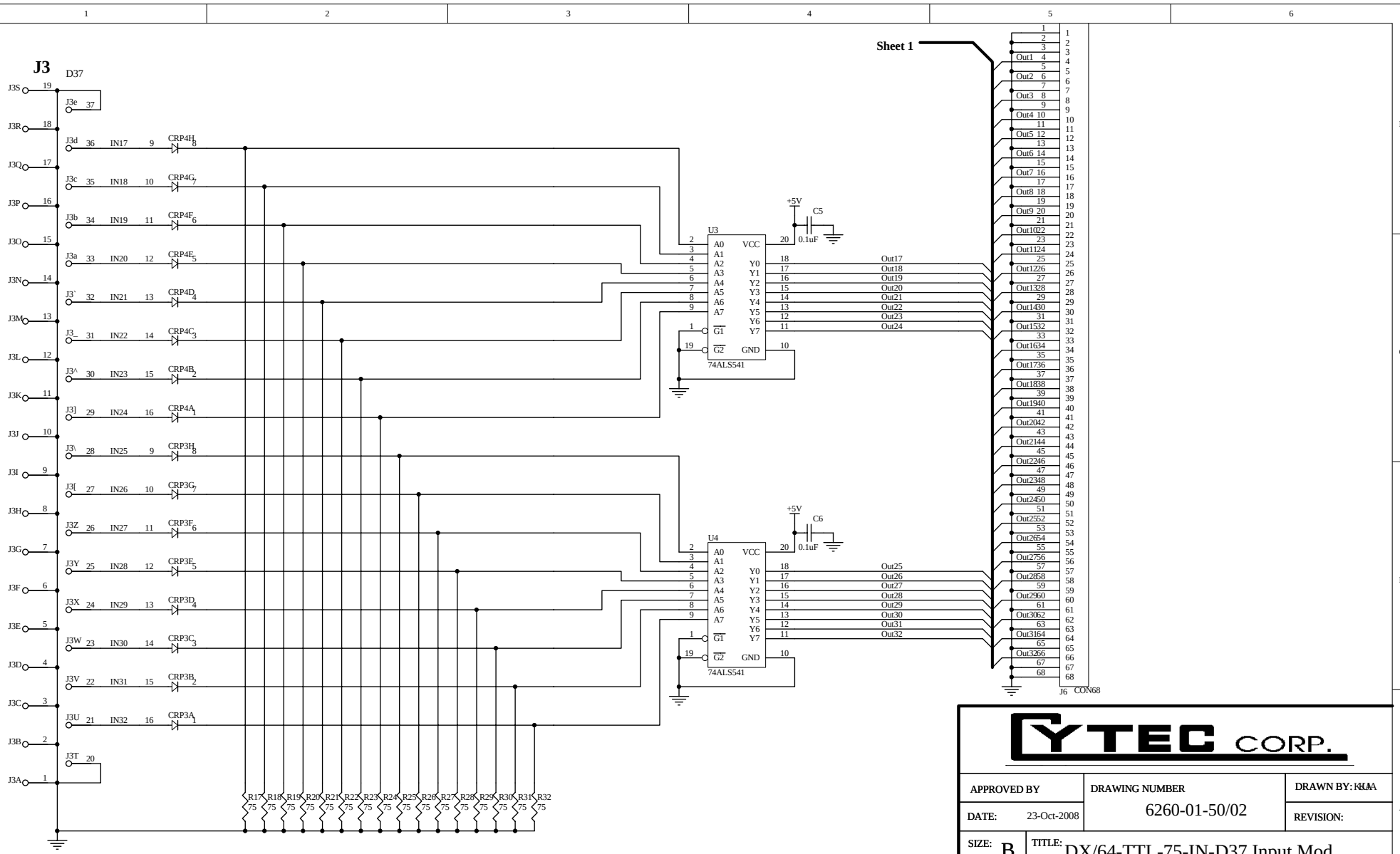
TOL. +/- .010" U.O.S.

DATE: 11/22/19	CAD FILE: 48-34-30.dcd	DRAWN BY: PV	
SIZE:	TITLE: DX/256x256 Power/Control Wiring		
COMMENTS:		DRAWING NUMBER: 48-34-30	

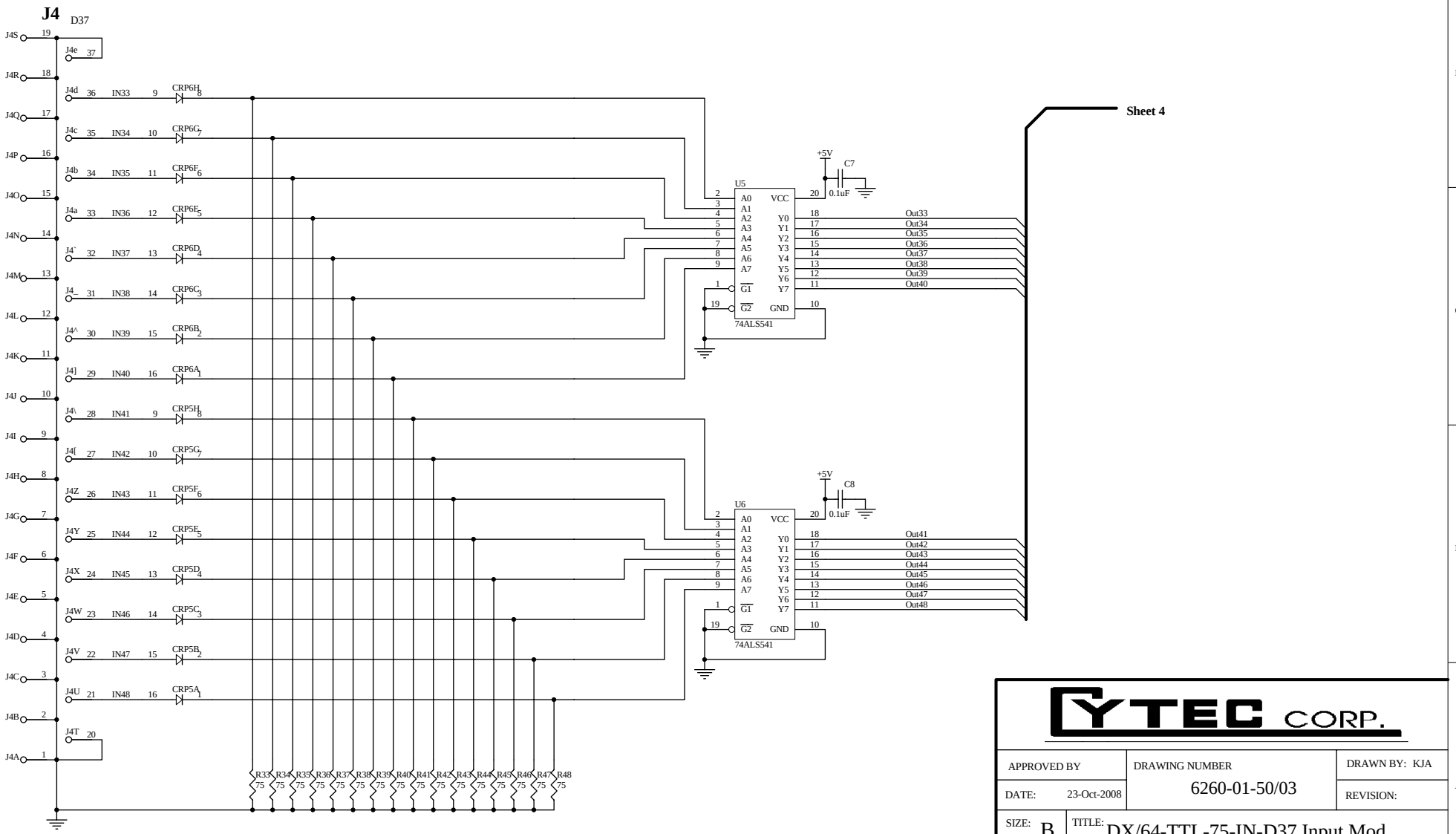


CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 23-Oct-2008	6260-01-50/01	REVISION:
SIZE: B	TITLE: DX/64-TTL-75-IN-D37 Input Mod.	
6260-01-50-01.sch	SHEET 1 OF 4	

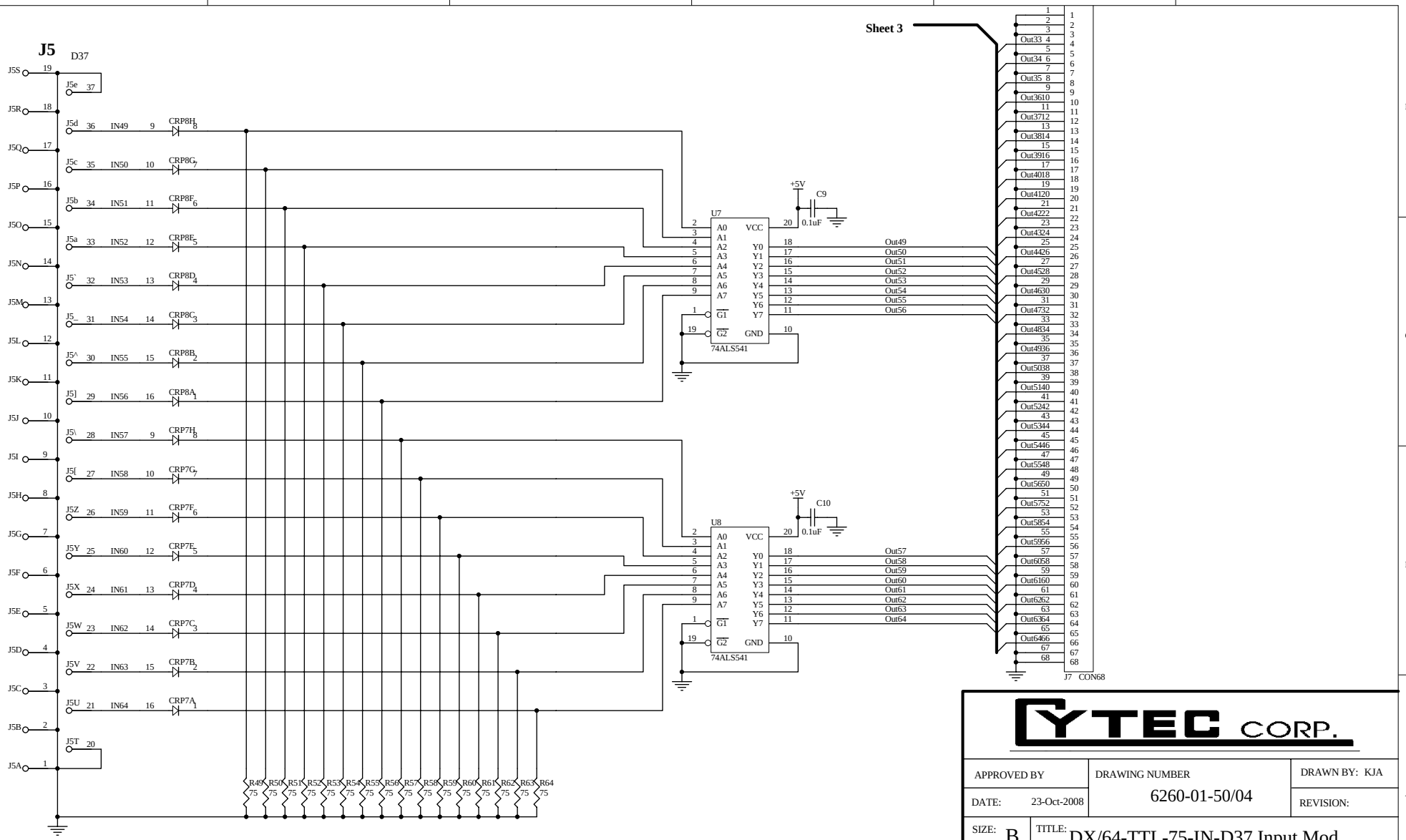


APPROVED BY	DRAWING NUMBER	DRAWN BY: KUA
DATE: 23-Oct-2008	6260-01-50/02	REVISION:
SIZE: B	TITLE: DX/64-TTL-75-IN-D37 Input Mod.	
6260-01-50-02.sch	SHEET 2 OF 4	

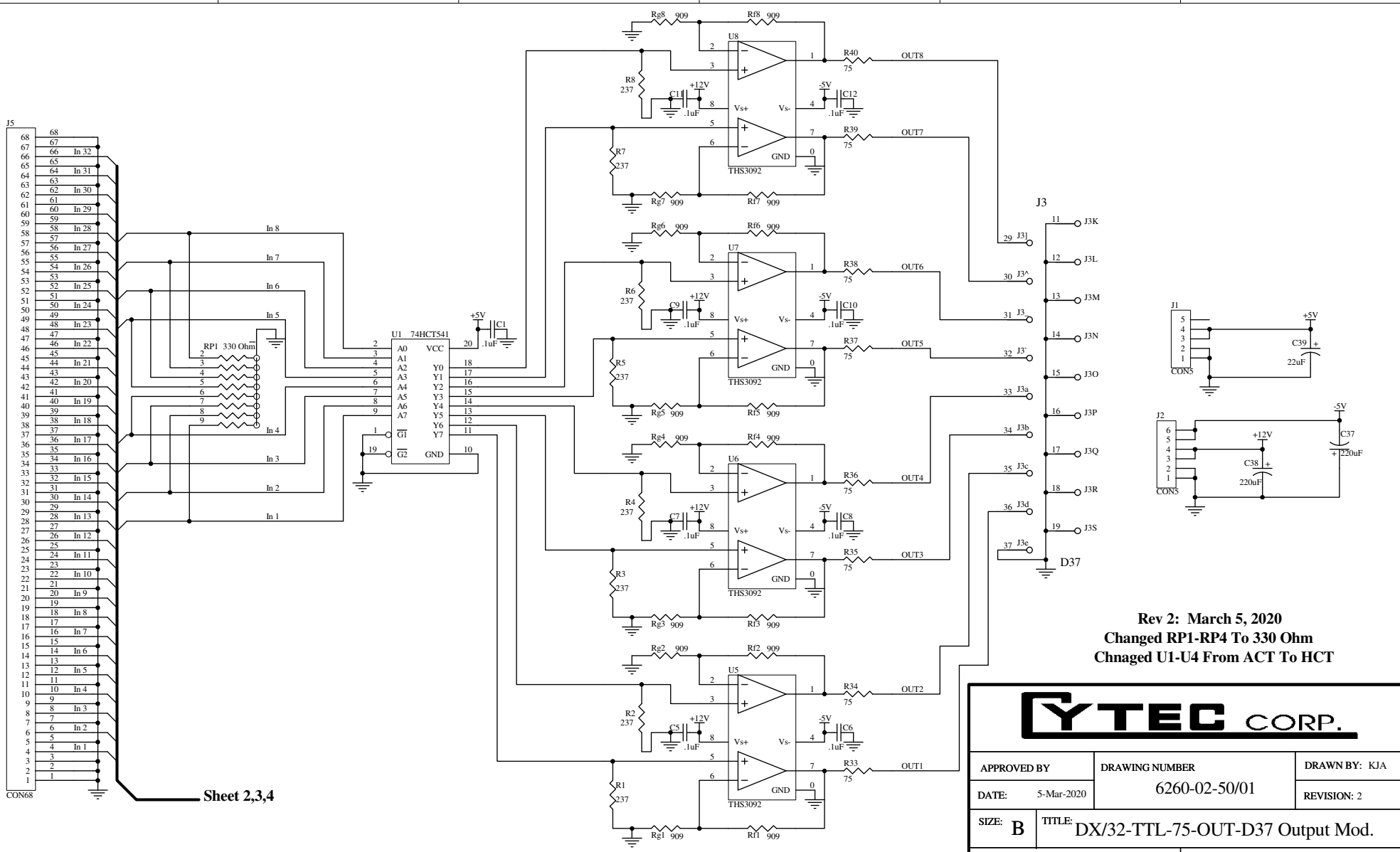


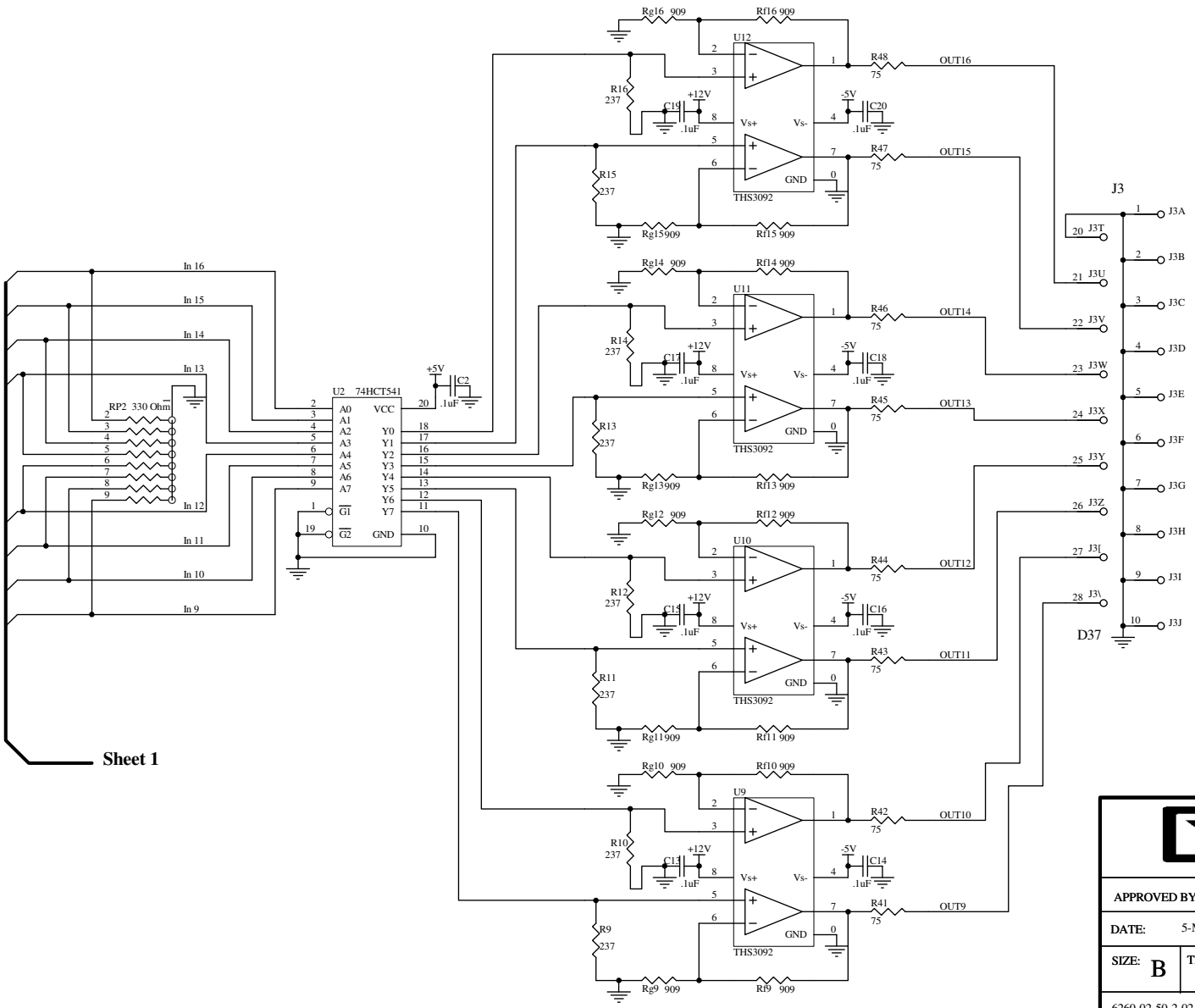
CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 23-Oct-2008	6260-01-50/03	REVISION:
SIZE: B	TITLE: DX/64-TTL-75-IN-D37 Input Mod.	
6260-01-50-03.sch		SHEET 3 OF 4

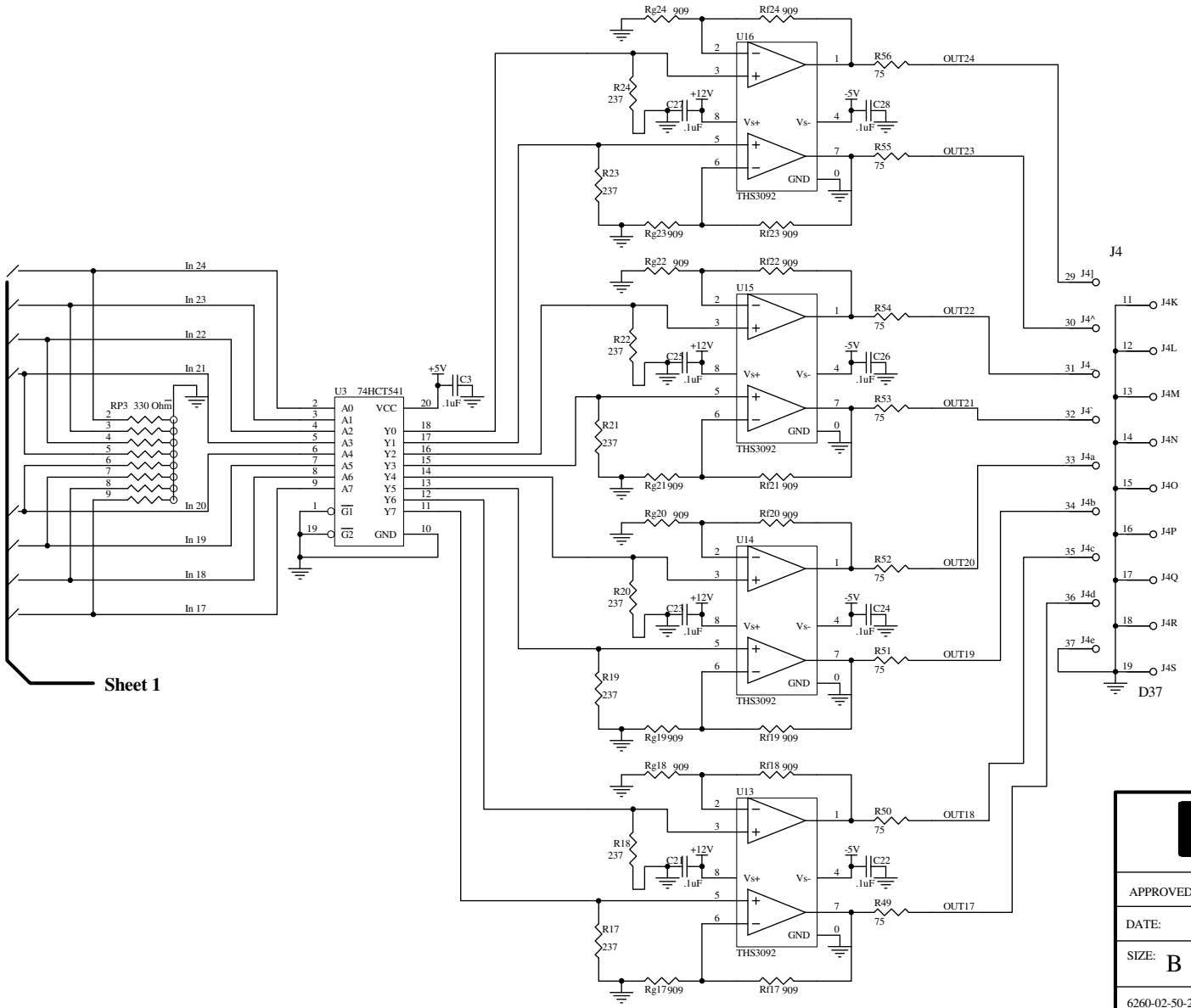


APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 23-Oct-2008	6260-01-50/04	REVISION:
SIZE: B	TITLE: DX/64-TTL-75-IN-D37 Input Mod.	
6260-01-50-04.sch		SHEET 4 OF 4

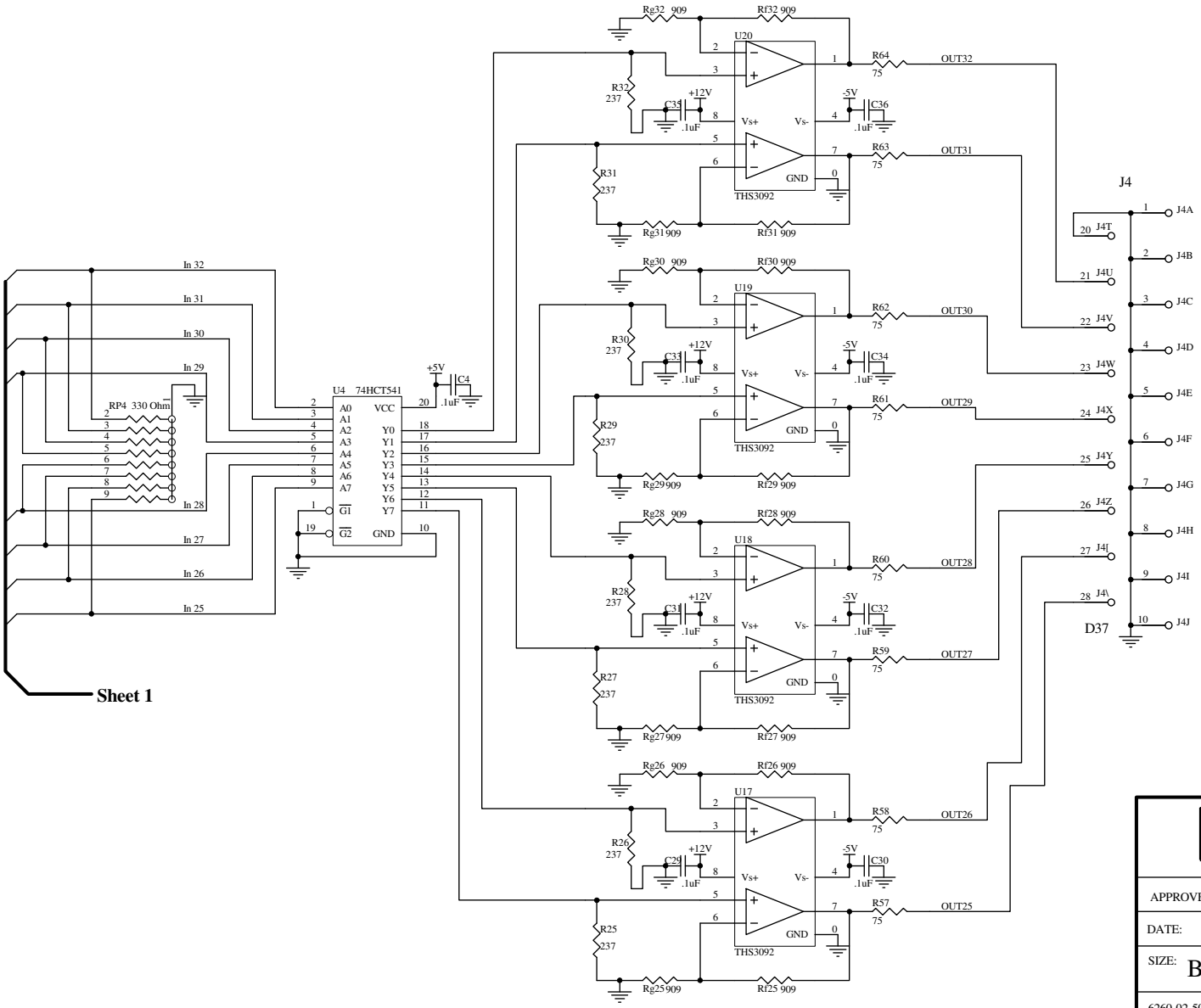

J5
CON68
In 1
In 2
In 3
In 4
In 5
In 6
In 7
In 8
In 9
In 10
In 11
In 12
In 13
In 14
In 15
In 16
In 17
In 18
In 19
In 20
In 21
In 22
In 23
In 24
In 25
In 26
In 27
In 28
In 29
In 30
In 31
In 32
Sheet 2,3,4
RP1 330 Ohm
U1 74HCT541
A0 A1 A2 A3 A4 A5 A6 A7
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
VCC GND
+5V C1 .1uF
U8 U7 U6 U5
THS3092
Vs+ Vs- GND
Rg8 909 Rf8 909
Rg7 909 Rf7 909
Rg6 909 Rf6 909
Rg5 909 Rf5 909
Rg4 909 Rf4 909
Rg3 909 Rf3 909
Rg2 909 Rf2 909
Rg1 909 Rf1 909
R8 237
R7 237
R6 237
R5 237
R4 237
R3 237
R2 237
R1 237
C11 C12 C9 C10 C7 C8 C5 C6
+12V -5V .1uF
R40 R39 R38 R37 R36 R35 R34 R33
75
OUT8 OUT7 OUT6 OUT5 OUT4 OUT3 OUT2 OUT1
J3
J3K J3L J3M J3N J3O J3P J3Q J3R J3S
J3I J3^ J3` J3a J3b J3c J3d J3e
D37
J1
CON5
C39 22uF
J2
CON5
C38 220uF
C37 220uF
Rev 2: March 5, 2020
Changed RP1-RP4 To 330 Ohm
Chnaged U1-U4 From ACT To HCT
CYTEC CORP.
APPROVED BY
DRAWING NUMBER
6260-02-50/01
DRAWN BY: KJA
DATE: 5-Mar-2020
REVISION: 2
SIZE: B
TITLE: DX/32-TTL-75-OUT-D37 Output Mod.
6260-02-50-2-01.sch
SHEET 1 OF 4



CYTEC CORP.			
APPROVED BY		DRAWING NUMBER	DRAWN BY: KJA
DATE: 5-Mar-2020		6260-02-50/02	REVISION: 2
SIZE: B	TITLE: DX/32-TTL-75-OUT-D37 Output Mod.		
6260-02-50-2-02.sch			SHEET 2 OF 4

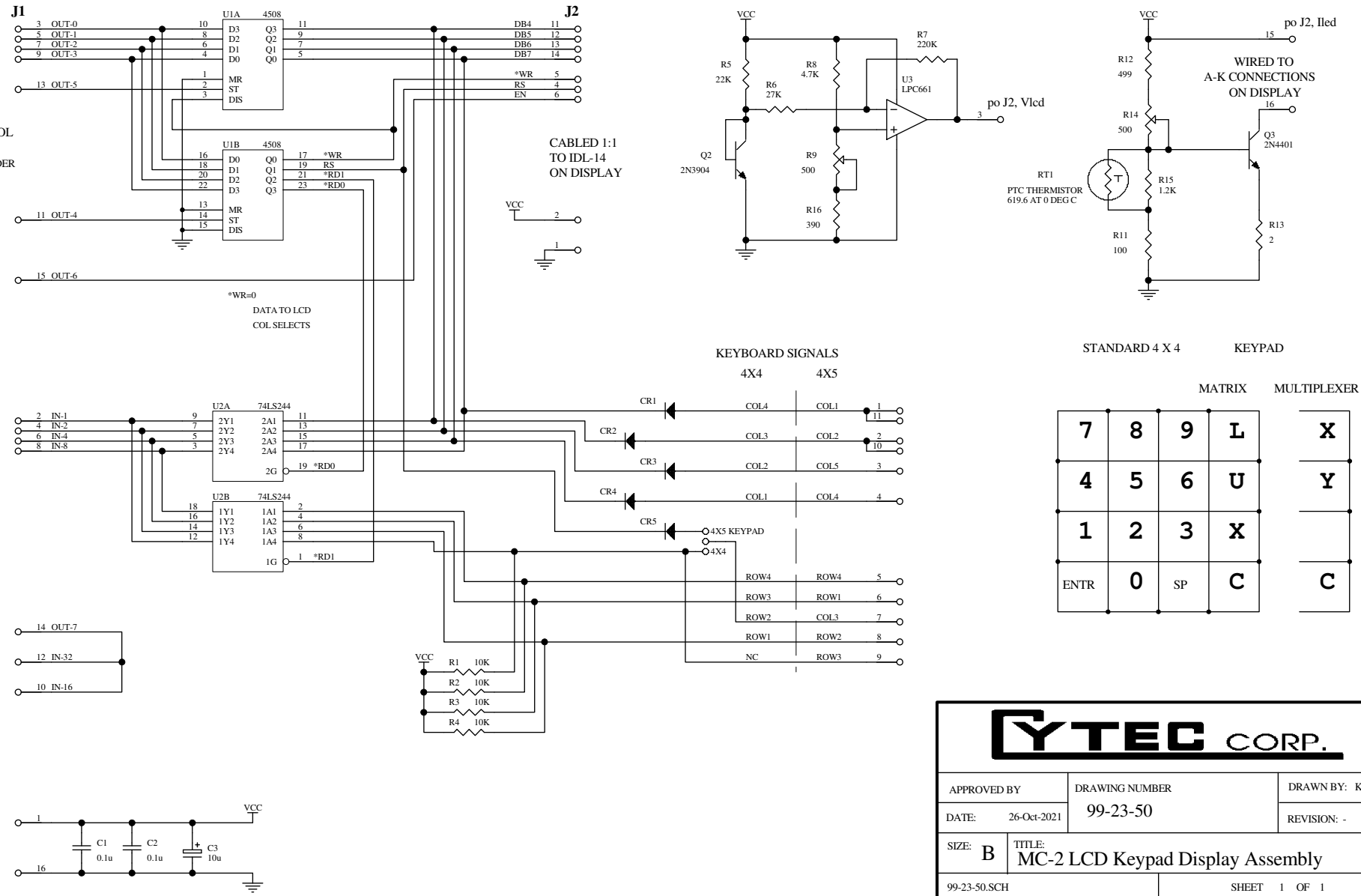


CYTEC CORP.		
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 5-Mar-2020	6260-02-50/03	REVISION: 2
SIZE: B	TITLE: DX/32-TTL-75-OUT-D37 Output Mod.	
6260-02-50-2-03.sch		SHEET 3 OF 4

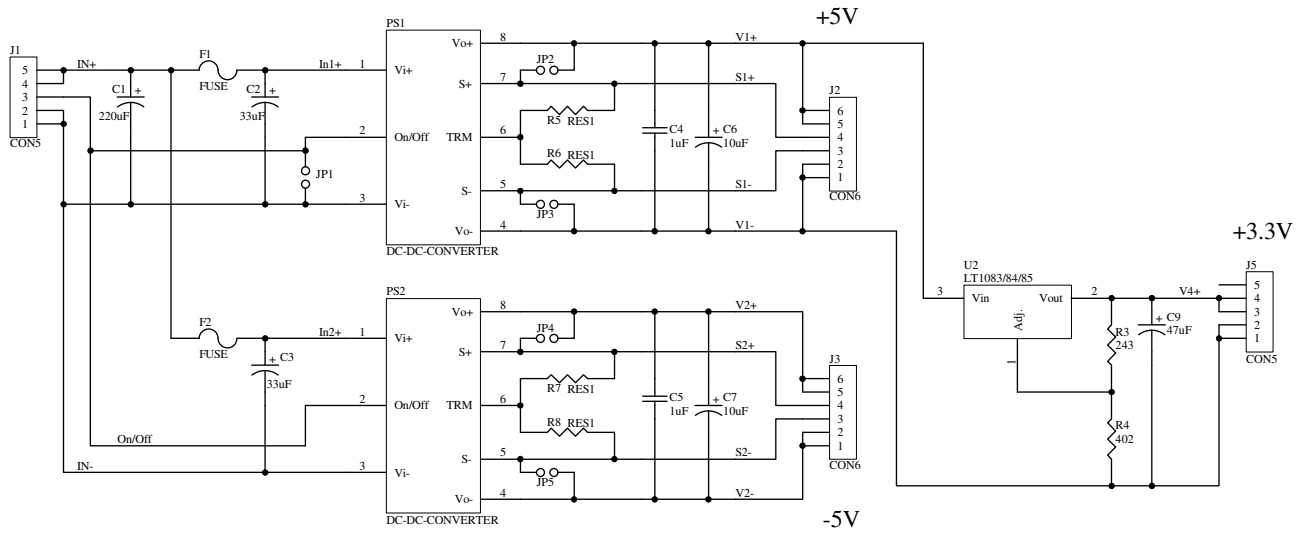


CYTEC CORP.

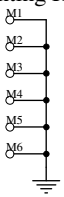
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 5-Mar-2020	6260-02-50/04	REVISION: 2
SIZE: B	TITLE: DX/32-TTL-75-OUT-D37 Output Mod.	
6260-02-50-2-04.sch		SHEET 4 OF 4



CYTEC CORP.		
APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 26-Oct-2021	99-23-50	REVISION: -
SIZE: B	TITLE: MC-2 LCD Keypad Display Assembly	
99-23-50.SCH		SHEET 1 OF 1



Mounting Holes



CYTEC CORP.

APPROVED BY	DRAWING NUMBER	DRAWN BY: KJA
DATE: 2-Jun-2022	99-76-54	REVISION: -
SIZE: B	TITLE: 9-36v IN +5v/-5v/+3.3v OUT DC/DC Mod.	
99-76-54.sch		SHEET 1 OF 1