User Manual

CTC100

Cryogenic Temperature Controller

Firmware version 4.59
September 15, 2023

Certification
Stanford Research Systems certifies that this product met its published specifications at the time of shipment.

Warranty
This Stanford Research Systems product is warranted against defects in materials and workmanship for a period of one (1) year from the date of shipment.

Service
For warranty service or repair, this product must be returned to a Stanford Research Systems authorized service facility. Contact Stanford Research Systems or an authorized representative before returning this product for repair.

Information in this document is subject to change without notice.

Stanford Research Systems, Inc.
1290-C Reamwood Avenue
Sunnyvale, California 94089
Phone: (408) 744-9040
Fax: (408) 744-9049
www.thinkSRS.com

Printed in the USA

Contents

Safety and preparation for use

Line voltage

The CTC100 operates from an 88 to 264 VAC power source with a line frequency between 47 and 63 Hz.

Power entry module

A power entry module, labeled AC POWER on the back panel of the CTC100, provides connection to the power source and to a protective ground.

Power cord

The CTC100 package includes a detachable, three-wire power cord for connection to the power source and protective ground.

The exposed metal parts of the box are connected to the power ground to protect against electrical shock. Always use an outlet which has a properly connected protective ground. Consult with an electrician if necessary.

Grounding

The back panel of the CTC100 has a chassis grounding lug. Connect a heavy duty ground wire, #12AWG or larger, from the chassis ground lug directly to a facility earth ground to provide additional protection against electrical shock.

Line fuse

Use a 10 A/250 V 3AB Slo-Blo fuse.

Operate only with covers in place

To avoid personal injury, do not remove the product covers or panels. Do not operate the product without all covers and panels in place.

Serviceable parts

The CTC100 does not include any user serviceable parts inside. Refer service to a qualified technician.

Specifications

CTC100 temperature controller

Minimum sampling rate	1 Hz
Maximum sampling rate	50 or 60 Hz, depending on AC line frequency
Data logging rate	10 samples/second/channel – 1 sample/hour/channel
Numeric resolution	0.001 °C, °F, K, V, A, W, etc. if –1000 < displayed value < 1000; 6 significant figures otherwise
LCD display	320 × 240 pixel color touchscreen
Computer interface	USB, Ethernet, and RS-232; optional GPIB (IEEE488.2)
Power	10 A, 88 to 132 VAC or 176 to 264 VAC, 47 to 63 Hz or DC
Dimensions	8.5" wide × 5" tall × 16" deep
Weight	13 lbs.
Warranty	One year parts and labor on defects in material and workmanship

Thermistor, diode, and RTD inputs

Inputs	Four inputs for 2-wire or 4-wire thermistor, diode, or RTD	
Connectors	Two 9-pin D-sub sockets	
RTDs and thermistors		
Range	0 – 10, 30, 100, 300Ω; 1, 3, 10, 30, 100, 300 kΩ; 2.5 MΩ, or auto	
Excitation current	**Low power**	**High power**
10 Ω range	1 mA	3 mA
30 Ω range	300 µA	3 mA
100 Ω range	100 µA	2 mA
300 Ω range	30 µA	1 mA
1 kΩ range	10 µA	500 µA
3 kΩ range	3 µA	200 µA
10 kΩ range	1 µA	50 µA
30 kΩ range	300 nA	50 µA
100 kΩ range	100 nA	5 µA
300 kΩ range	30 nA	5 µA
2.5 MΩ range	1 µA	1 µA
Initial accuracy (AC current, at midrange)		
10 Ω range	±0.007 Ω	±0.005 Ω
30 Ω range	±0.03 Ω	±0.005 Ω
100 Ω range	±0.07 Ω	±0.008 Ω
300 Ω range	±0.25 Ω	±0.015 Ω (=±40 mK for Pt100 RTD at 25°C)
1 kΩ range	±0.6 Ω	±0.05 Ω
3 kΩ range	±2 Ω	±0.1 Ω
10 kΩ range	±6 Ω	±0.25 Ω
30 kΩ range	±25 Ω	±1 Ω
100 kΩ range	±150 Ω	±4 Ω
300 kΩ range	±1 kΩ	±13 Ω
2.5 MΩ range	±3 kΩ	±3 kΩ
Typical drift due to temperature (at midrange)		
10 Ω range	±0.0002 Ω/°C	±0.0001 Ω/°C
30 Ω range	±0.0004 Ω/°C	±0.0001 Ω/°C
100 Ω range	±0.002 Ω/°C	±0.0002 Ω/°C
300 Ω range	±0.004 Ω/°C	±0.0004 Ω/°C
1 kΩ range	±0.01 Ω/°C	±0.001 Ω/°C
3 kΩ range	±0.06 Ω/°C	±0.003 Ω/°C

10 kΩ range	±0.2 Ω/°C	±0.01 Ω/°C
30 kΩ range	±1 Ω/°C	±0.02 Ω/°C
100 kΩ range	±3 Ω/°C	±1 Ω/°C
300 kΩ range	±20 Ω/°C	±2 Ω/°C
2.5 MΩ range	±30 Ω/°C	±50 Ω/°C
RMS noise (DC current, at midrange)		
10 Ω range	0.0003 Ω	0.0001 Ω
30 Ω range	0.001 Ω	0.0001 Ω
100 Ω range	0.002 Ω	0.0002 Ω
300 Ω range	0.006 Ω	0.0003 Ω (= 1.4 mK for Pt100 RTD at 25°C)
1 kΩ range	0.02 Ω	0.0007 Ω
3 kΩ range	0.06 Ω	0.002 Ω
10 kΩ range	0.2 Ω	0.007 Ω
30 kΩ range	1.0 Ω	0.008 Ω
100 kΩ range	6 Ω	0.12 Ω
300 kΩ range	40 Ω	0.2 Ω
2.5 MΩ range	10 Ω	10 Ω

Diodes	
Excitation current	10 μA
Initial accuracy	± 100 ppm
Drift	±5 ppm/°C
Voltage input	0 – 2.5 V
Initial accuracy	10 μV + 0.01% of reading
Drift	±5 ppm/°C
RMS noise	3 μV

100W DC outputs

Output	Two unipolar DC current sources
Connector	#6 screw terminals. Accepts 12–22 AWG wire or #6 spade terminals up to 0.31" wide. Max torque 9 in-lb.
Range	50 V 2A, 50V 0.6A, 50V 0.2A, 20V 2A, 20V 0.6A, 20V 0.2A
Output resolution	16 bit
Accuracy	±1 mA (2 A range) ±0.5 mA (0.6 A range) ±0.2 mA (0.2 A range)
Noise (rms), 25 Ω load, DC–10 Hz	5 μA (2 A range) 1.5 μA (0.6 A range) 0.5 μA (0.2 A range)

Analog I/O

Inputs/outputs	4 voltage I/O channels, independantly configurable as inputs or outputs
Connector	4 BNC jacks
Range	±10 V
Resolution	24-bit input, 16-bit output
ADC noise	30 μV RMS = 100 μV p-p (at 10 samples/s)

Digital I/O

Digital I/O	
Inputs/outputs	8 optoisolated TTL lines, configurable as either 8 inputs or 8 outputs

Connector	One 25-pin D-sub socket

Relays

Outputs	4 independent SPDT relays
Connector	One 12-pin 3.5mm header
Maximum current	5 A
Maximum voltage	250 VAC

Introduction

The CTC100 is a high-performance cryogenic temperature controller that monitors RTD, thermistor, or diode temperature sensors with millikelvin resolution, and controls temperatures by providing power to resistive heaters. Features include:

4 temperature sensor inputs

Each of the CTC100's four temperature inputs can read any kind of resistive temperature sensor that has a resistance under 300 kΩ (including RTDs, thermistors, ruthenium oxide, carbon glass, etc.), or diodes with a voltage drop under 2.5V.

Temperature sensors can be calibrated in three ways:

- One of the 33 preloaded calibration curves for common sensor types can be selected.
- Steinhart-Hart, Callendar-van Dusen, or diode calibration factors can be entered.
- Custom calibration curves with up to 400 (temperature, resistance) or (temperature, voltage) data points each can be loaded, either with a communication port or a USB memory stick.

Each sensor input has an alarm that can be triggered when the input exceeds a predefined limit. When triggered, an alarm can play a sound, shut off a heater output, and/or trip a relay.

Sensor inputs can be lowpass filtered to reduce noise; the value of a second input can be subtracted; and the rate of change can be calculated.

Two 100W heater outputs

The CTC100 has two heater outputs that can be used with resistive heating elements having a resistance between 10 and 250 Ω. 25 Ω heaters are preferred because they produce the most power, up to 100W. Heaters with a higher or lower resistance will produce less power.

Each heater output has a PID feedback loop that continually adjusts the heater power to keep a sensor at a constant temperature.

General-purpose analog I/O, digital I/O, relays, and virtual channels

The CTC100 has four general-purpose ±10V analog I/O channels, each of which supports PID feedback and custom calibration curves.

The CTC100 also has eight digital I/O channels and four 5A relays.

Three virtual channels allow calculated values (such as the difference between two channels, or a value calculated by a user program) to be displayed, graphed, and logged.

Graphical touchscreen display

The CTC100 can display temperature measurements and heater outputs on graphs or numerically. Up to eight channels can be plotted at the same time, either on a single graph with a common Y axis or on separate graphs with independent Y axes.

Data logging

Up to one million readings per channel can be saved in the CTC100's internal memory.

Data can also be saved as a text file on USB memory sticks or hard drives. Readings can be logged at intervals between 0.1 s and 1 hour. A Windows application that graphs CTC100 log files can be downloaded from the SRS website.

Computer communications

The CTC100 can be controlled over USB, Ethernet, and either RS-232 or an optional GPIB interface. When the USB interface is used, the CTC100 appears on the computer as a standard COM port and can be controlled by any software that is compatible with an RS-232 port.

User programs

The CTC100's functions can be scripted so that, for example, a sequence of temperatures can be programmed. User programs can also add new capabilities; for example, a background program can make it possible for an analog input to control a user setting such as the feedback setpoint.

Connecting the inputs and outputs

Temperature sensor inputs

The CTC100 has four inputs, each of which can read resistive sensors having resistances between 1 Ω and 2.5 MΩ, and diode sensors having voltage drops of up to 2.5V.

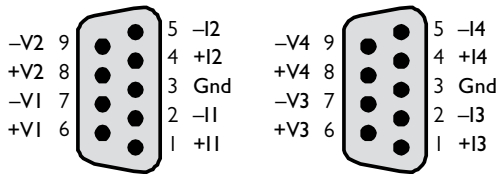
Standard calibration curves are included for the sensors shown in the following table. The "Range" column indicates the range of the standard calibration curve; if the sensor is outside this range, no reading appears. It may be possible to increase the range by uploading a custom calibration curve.

Sensor class	Manufacturer	Calibration type	Range, K
Diode	Scientific Instruments	Si410	1.0–450
		Si415	1.0–449
		Si430	1.0–400
		Si435	1.0–400
		Si440	1.0–500
		Si540	1.0–450
	LakeShore; Omega	DT-470 (=CY7)	1.4–475
		DT-670 (=CY670)	1.4–500
	Cryo-Con	S700	1.5–475
		S800	1.4–385
		S900	1.4–500
		S950	1.4–370
Ruthenium oxide	LakeShore	RX-102A	0.050–40
		RX-103A	1.2–40
		RX-202A	0.050–40
	Scientific Instruments	RO600	1.0–300
	Cryo-Con	R400	2.0–273
		R500	0.050–20
RTD	All	IEC751 (DIN43760)	48.15–1173.15
		US	48.15–1173.15
Thermistor	TE Connectivity (formerly Measurement Specialties, Inc. and YSI); Omega	100 Ω	193.15–373.15
		300 Ω	193.15–373.15
		1000 Ω	193.15–373.15
		2252 Ω	193.15–523.15
		3000 Ω	193.15–523.15
		5000 Ω	193.15–523.15
		6000 Ω	193.15–523.15
		10000 Ω type B (32.66 kΩ at 0°C)	193.15–523.15
		10000 Ω type H (29.49 kΩ at 0°C)	193.15–523.15
		30 kΩ	233.15–523.15
		100 kΩ	233.15–423.15
		300 kΩ	298.15–423.15
		1 MΩ	298.15–423.15

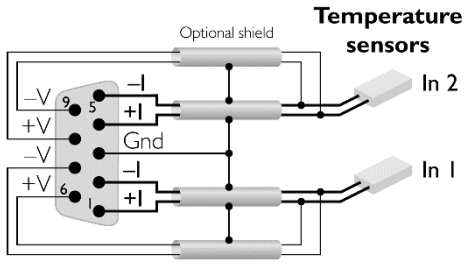
Other kinds of resistive and diode sensors can be used, but require custom calibration curves. For example, rhodium-iron, germanium, and carbon-glass sensors have too much sensor-to-sensor variability to use a standard curve, and therefore must be individually calibrated.

Connecting the sensor

The CTC100 has two 9-pin D-sub (DB9) sockets that mate with any standard DB9 plug, such as Amphenol L717SDE09P with backshell 17E-1657-09. Two plugs and backshells are provided with each CTC100. Here is the pinout of the two sockets, as they appear when looking at the CTC100's back panel:



Sensors In 1 and In 2, for example, should be connected as shown below.



The +I and –I pins provide an excitation current that should be routed to the temperature sensor through two wires, preferably a shielded twisted pair. The excitation current produces a voltage across the sensor that is measured with pins +V and –V. These pins should be connected to the sensor with two additional wires (preferably a second shielded twisted pair): +V should be connected to +I as close as possible to the temperature sensor, and likewise –I should be connected to –V as close as possible to the sensor. Because essentially no current flows through the V leads, they accurately transmit the sensor voltage to the CTC100. Using four wires instead of two ensures that the CTC100 measures the resistance of the sensor and not the wires going to the sensor.

The +V and –V pins are internally connected to the +I and –I pins with 1 MΩ resistors, so the sensor will still work if the +V and –V pins are not connected. However, the reading is more accurate when all four pins are connected.

Resistive sensors: Four-wire RTDs usually have two wires of one color attached to one side of the RTD, and two of a second color attached to the other side. In this case, the RTD should be wired to the CTC100 in one of the following two ways (assuming the leads are white and black):

	–V	–I	Ground	+V	+I
Option 1	White	White	Unconnected	Black	Black
Option 2	Black	Black	Unconnected	White	White

Two-wire sensors can be converted to four-wire sensors by soldering two additional wires to the existing leads, one on each side of the sensing element and as close to the sensing element as possible. A good-quality solder joint is essential; the wires should not be connected to the sensor by

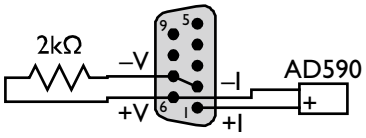
pressure alone (e.g. crimping or clamping), as any resistance within the joint becomes part of the measured sensor resistance.

The higher the resistance of an RTD or thermistor, the more sensitive it is to ambient electromagnetic noise and the greater the need for a shielded cable.

Diode sensors: Diode sensors can be connected in either direction. If the cathode is connected to –V and –I and the anode to +V and +I, *Channel*.Current should be set to Forward. If the diode is connected in the opposite direction, the current should be set to Reverse.

Diode sensors are especially susceptible to electromagnetic noise because the diode rectifies any noise picked up by the sensor leads, increasing the measured voltage. It may be necessary to put EMI filters not only the on the sensor leads but also on all other leads entering the Dewar. The filters should be located at the point where the wires enter the Dewar, and the Dewar itself should be grounded. D-sub and circular connectors with built-in filters can be obtained from Spectrum Advanced Specialty Products. We have found their 4000 pF pi filters to be effective. These filters include capacitors to ground, which should be connected to the Dewar.

AD590 sensors: The CTC100 can read AD590 sensors if the sensor is connected in series with a 2 kΩ resistor as shown below. Note that the diagram shows the sensor connected to channel A, but it can also be connected to channel B. The diagram shows the back of the DB9 connector, that is, the side that you solder to.



The 2 kΩ resistor must have a low temperature coefficient of resistance (TCR). Ordinary resistors have a TCR of about 100 ppm/°C, which means that the sensor reading will increase by about 30 mK for each 1°C rise in the ambient temperature. Thermal drift can be reduced substantially by using a 5 ppm/°C resistor available from SRS; ask for part number 4-02502-457. For even better stability, a 1 ppm/°C resistor such as the Riedon USR2G-2KX1, available from Digi-Key, can be used. In any case, to minimize noise and drift, the resistor should be soldered directly to the pins on the DB9 plug and covered up with the backshell.

Because AD590 sensors are highly sensitive to electromagnetic interference, the AD590 wires and package must be shielded, with the shield connected to pin 3 of the DB9 connector.

Use of threaded nuts on the sensor input connectors

To ensure accurate temperature readings, the DB9 sensor input connectors are held onto the chassis with #4-40 pan head Philips screws. These screws don't leave any place to screw the sensor cable to the CTC100.

Normally, DB9 connectors are held in place with M/F hex standoffs (screws with threaded nuts for heads). Such standoffs aren't used on the CTC100 because they often come loose when the user unscrews the connector. The D-sub sensor connector contains special EMI filters, so if the screws become loose, the EMI filters won't be properly grounded to the chassis, and high-frequency EMI noise will be emitted onto the sensor cable. Diode sensors rectify the noise, causing the temperature reading to be significantly higher than the correct value.

It's only OK to replace the pan head screws with hex standoffs if 1) diode sensors are never used with the instrument or 2) the standoffs are re-tightened every time the sensor is unplugged.

The screws should not be glued into the connector, because the connectors will be destroyed when the instrument is disassembled for servicing. Likewise, over-tightening the hex standoffs can destroy the connectors. The DB9 connectors are custom parts that cannot be replaced by the user without compromising the CTC100's accuracy.

Excitation current

The CTC100 measures the resistance of the sensor by passing an excitation current through it. The larger the excitation, the less noise the temperature reading will have. However, if the excitation is too large it will heat the sensor and cause higher than expected readings. Therefore, each channel can be configured for either "low power" or "high power" operation:

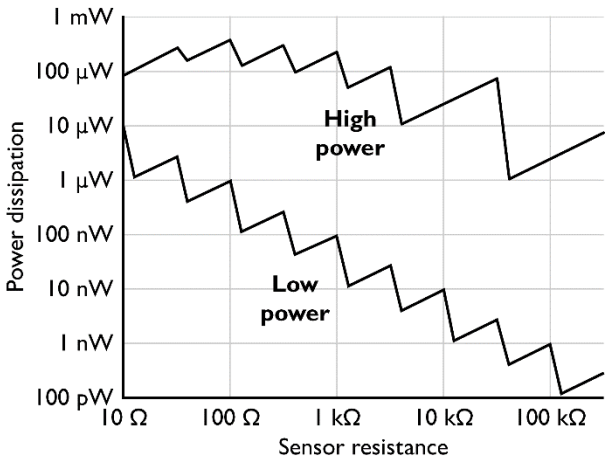
- **Low power**: minimizes sensor heating. This option is mainly for use with thermistors in cryogenic applications. To compensate for the fact that heat conductivity decreases (and thermistor resistance increases) as the temperature approaches absolute zero, the amount of power that the sensor dissipates decreases as the measurement range is increased.
- **High power:** minimizes noise. Power dissipation is kept roughly constant as the measurement range is increased. This option is for use with RTDs or with any kind of sensor at non-cryogenic temperatures.
- **Auto power:** uses low power if the sensor type is set to thermistor or ROX, or high power if the sensor type is set to RTD.

The CTC100 has 12 measurement ranges. Within any given range, it generates a constant excitation current as shown in the table below. Note that the range has to be greater than the sensor resistance, so if the sensor resistance is 10 kΩ, for example, the range should be 30 kΩ.

For diode sensors the range is always 2.5V and the excitation current is always 10 μA.

The graph below shows how the amount of power dissipated by the sensor depends on the range and power settings. Sensor heating (degrees above the ambient temperature) is proportional to power dissipation.

Measurement range	Excitation current	
	Low power	High power
10 Ω	1 mA	3 mA
30 Ω	300 μA	3 mA
100 Ω	100 μA	2 mA
300 Ω	30 μA	1 mA
1 kΩ	10 μA	500 μA
3 kΩ	3 μA	200 μA
10 kΩ	1 μA	50 μA
30 kΩ	300 nA	50 μA
100 kΩ	100 nA	5 μA
300 kΩ	30 nA	5 μA
2.5 MΩ	1 μA	1 μA
2.5 V	10 μA	10 μA



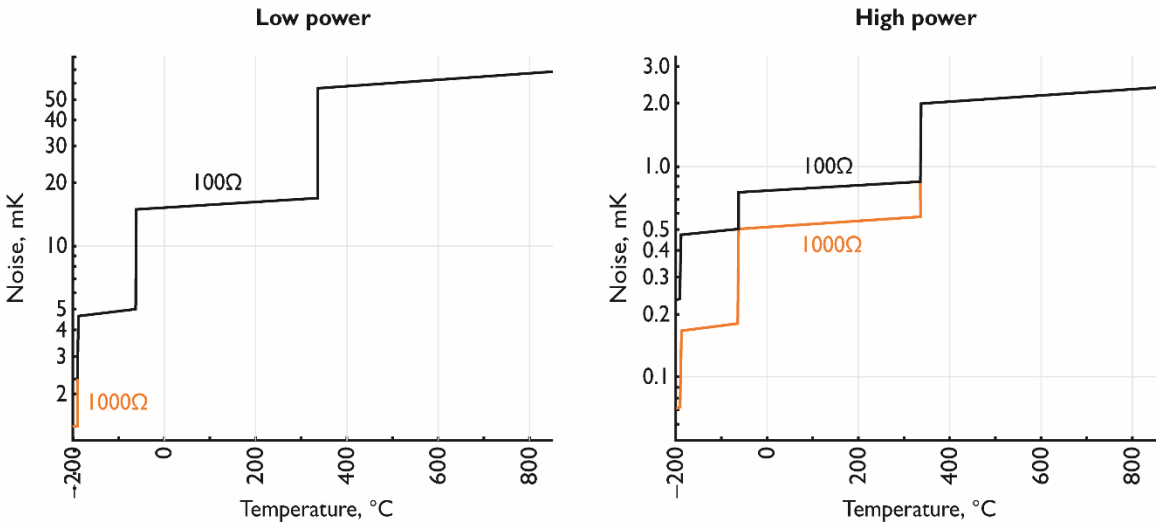
Left: the amount of current passed through the sensor by the CTC100; right: the amount of power that the sensor dissipates due to that current

The table below shows some representative noise, electronic accuracy, and self-heating values for free-standing sensors at room temperature. Note that the amount of self-heating can vary dramatically depending on the thermal conductivity of whatever the sensor is attached to or immersed in.

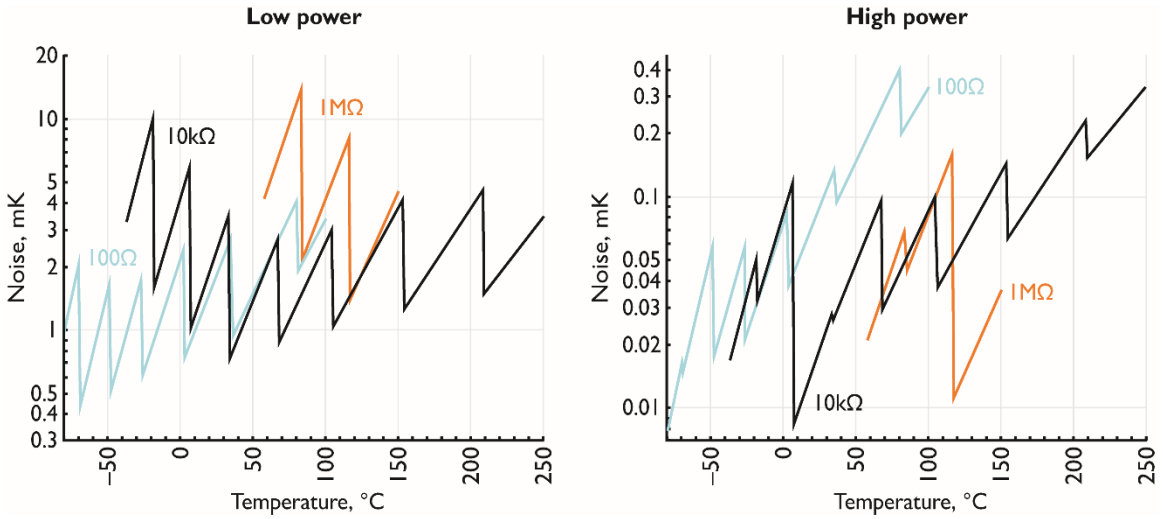
	Noise		Accuracy		Self-heating	
	Low power	High power	Low power	High power	Low power	High power
100Ω RTD	20 mK	0.8 mK	640 mK	40 mK	0.09 mK	100 mK
1 kΩ thermistor	2 mK	0.08 mK	60 mK	3 mK	0.009 mK	40 mK
10 kΩ thermistor	2 mK	0.02 mK	50 mK	2 mK	0.0009 mK	25 mK
100 kΩ thermistor	9 mK	0.04 mK	220 mK	3 mK	0.00009 mK	2.5 mK

Noise, accuracy, and amount of self-heating at 25°C for several sensors. "Accuracy" is the electronic accuracy of the CTC100 immediately after calibration and does not account for self-heating or the accuracy of the sensor. "Self-heating" is the rise above ambient temperature of a ~1 mm diameter sensor hanging by its leads in still air (dissipation constant 1 mW/°C).

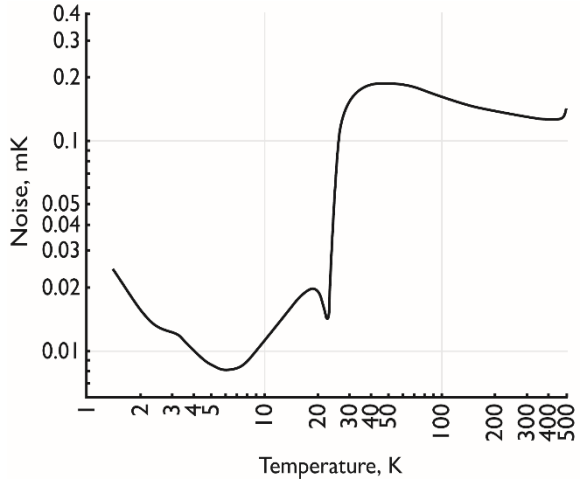
The graphs below show how electronic noise varies with temperature for several types of sensors.



RMS noise levels for 100Ω and 1000Ω platinum RTD sensors as a function of temperature. At low power, the 100Ω and 1000Ω sensors mostly have the same noise level.



RMS noise levels for 100Ω, 10kΩ, and 1MΩ (at 25 °C) thermistors as a function of temperature



RMS noise for DT-670 diode sensor as a function of temperature.

The direction of the excitation current can be set by the user to forward, reverse, or AC (switching between forward and reverse with each sample). AC current is recommended for resistive sensors to reduce noise and drift. AC current cannot be used with diode sensors. See the discussion of the Current setting on page 61.

100W heater outputs

The CTC100 has two outputs for resistive heaters. The output connectors are #6-32 wire clamp screws. Although the screws can be used with bare 12–22 AWG wire, for the most reliable connection it's recommended to crimp a #6 insulated spade terminal (such as TE Connectivity 34080 for 16–22 AWG wires or 35559 for 14–16 AWG wires) to the end of each heater wire. A crimp tool such as TE Connectivity 58433-3 should be used for this purpose.

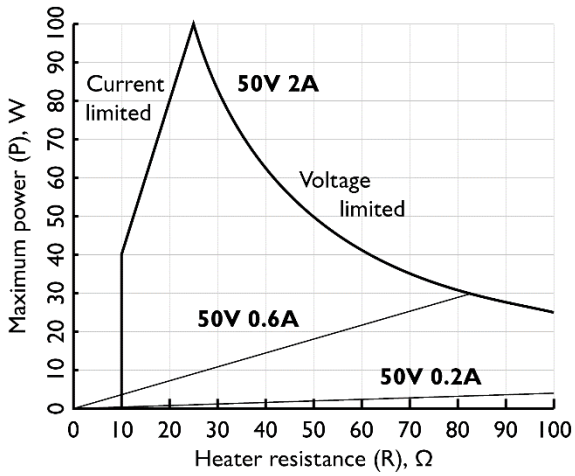
The CTC100 should always be switched off and unplugged when connecting the heater outputs.

Each output has two voltage ranges (50 V and 20 V) and three current ranges (2A, 0.6A, and 0.2A). An auto-range setting continuously adjusts the current and voltage ranges to the smallest values needed to reach the channel's Hi Lmt setting.

The 0.6A and 0.2A current ranges offer lower noise levels than the 2A range, but the 20V range has essentially the same performance as the 50V range.

The maximum power that each output can deliver depends on the resistance of the heater as shown below.

Output range	Heater resistance (*R*), Ω	Maximum power, W
50 V 2 A	<10	0
	10 – 25	4*R*
	25	100
	>25	2500/*R*
50 V 0.6 A	<83	0.36*R*
	83	30
	>83	2500/*R*
50 V 0.2 A	<250	0.04*R*
	250	10
	>250	2500/*R*



Maximum output power as a function of output range and heater resistance

Each of the two heater outputs is generated by a printed circuit board (PCB). If the temperature of either PCB exceeds 60°C, the CTC100 displays an "output card overheated" message and automatically shuts off the affected output until the temperature drops below 60°C. This can occur if the heater resistance is under 10 Ω, if the ambient temperature outside the chassis is above 30°C, and/or if the system fan is turned off or not working. The PCB temperatures can be monitored by going to the System Setup screen and setting the Monitors control to Show.

The heater output connectors should not be modified or replaced because they have built-in EMI filters.

±10V analog I/O channels

Each of the four BNC connectors on the back of the CTC100 can be a ±10V input (24-bit ADC) or output (16-bit DAC). The outer shell of each connector is grounded.

If used to drive a heater, each analog I/O channel can only supply up to 30 mA of current. Furthermore, because of their limited accuracy, when set to 0 V the analog outputs may still feed a small amount of power to the heater. This residual current can be eliminated by placing a diode in series with the heater and increasing the channel's Lo Lmt setting to about 0.5V to prevent integral windup.

Relays, digital I/O, and virtual channels

Relays

The CTC100 has four relays, each rated for up to 5A of current. The connector for the relays is a single 12-pin pluggable terminal block. The four relays are labeled "A" through "D", and each relay has three connections labeled "NC" (normally closed), "COM" (common), and "NO" (normally

open). The relay is in its "normal" or "deactivated" state when the CTC is turned off, when its outputs are not enabled, or when the relay is set to 0. In this state, the "NC" pin is connected to the "COM" pin and the "NO" pin is unconnected. When the relay is set to 1 and the outputs are enabled, the relay is activated: the "NO" pin is connected to the "COM" pin and the "NC" pin is unconnected.

The relays appear on the CTC100 display as a single 4-bit integer value between 0 and 15. If no relays are activated, the value is 0. Each relay, if activated, adds the following to the displayed value:

Relay	Value
A	1
B	2
C	4
D	8

Therefore, if the relay channel reads "6", relays B and C are activated. Conversely, setting the relay channel to 6 activates relays B and C, and deactivates the other relays. To set an individual relay from a macro or serial port without affecting the states of other relays, use a bitwise operator; for example, the remote command

```
relays |= 4
```

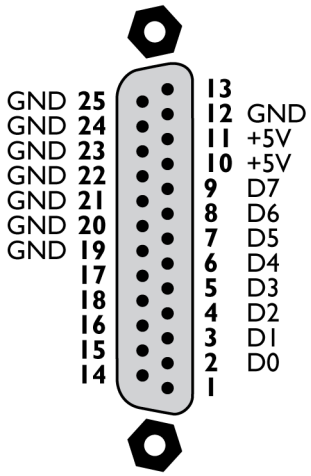
activates relay C, while the remote command

```
relays &= 11
```

deactivates relay C. See the "Remote programming" section of this manual for more information on remote commands.

Digital I/O

The CTC100 also has eight isolated TTL I/O lines on a 25-pin connector. The pinout of this connector is compatible with the standard PC parallel port. The TTL lines can be used as inputs or outputs, but all eight must have the same direction. The pinout follows (looking at the back panel of the CTC100):



All 25 pins on this connector are electrically isolated from the rest of the CTC100 and are floating with respect to earth ground. Therefore, to use the digital I/O lines, at least one of the "Gnd" pins must be connected to ground. Alternatively, if the digital I/O lines are configured as inputs, the value of D0 to D7 can be set by shorting them either to a +5V pin or to a Gnd pin.

The status of the eight digital I/O lines is reported as a single eight-bit integer value. Each I/O line is assigned an integer value as shown in the following table:

Bit	Value
D0	1
D1	2
D2	4
D3	8
D4	16
D5	32
D6	64
D7	128

The "DIO" value is the sum of the values of all set bits. For example, if only bits D1 and D3 are set, the CTC100 shows a DIO value of 2 + 8 = 10.

By using background macros, the digital I/O lines can be associated with most functions of the CTC100. The remote interface provides bitwise operators to set and query the relays and digital I/O lines.

The DIO lines can also be used to pass a single, 8-bit value into or out of the CTC. The CTC treats the DIO like any other channel; for example, its value can be plotted or used in a PID feedback loop.

Virtual channels

The digital I/O card has three virtual channels, by default named V1, V2, and V3. Macros can use these channels to perform simple real-time calculations (such as determining the average of several inputs) or to plot or log variables such as the setpoint.

Virtual channels can also be used without macros. For example, if the IO type of a virtual channel is set to "Input", the channel can follow the value of another channel (see the description of the Channel.Follow button in the Operation section), and in addition can be modified by applying a lowpass filter, subtracting a difference channel, taking its derivative with respect to time, or applying offset and gain factors. By doing these calculations on a virtual channel that has been configured to follow a sensor input (instead of doing them directly on the sensor input channel), the raw sensor input is preserved and can still be viewed.

If a virtual channel's IO type is set to "Meas out", the channel's value can be controlled by a PID feedback loop. A virtual channel can be used to implement cascade feedback control, for example; see the description of the Channel.PID.Casc button in the Operation section. Unlike other outputs, virtual outputs are not forced to zero when the CTC100's outputs are disabled with the Output Enable button. However, virtual PID feedback loops do stop running when the CTC100's outputs are disabled.

When the value of a virtual channel is changed by a macro or via the front panel, the new value does not become effective until an ADC conversion occurs. Therefore, if a macro sets the value of a virtual channel and then immediately reads the value back, the old value may be returned.

Operation

Quick start tutorial

Turn the instrument on

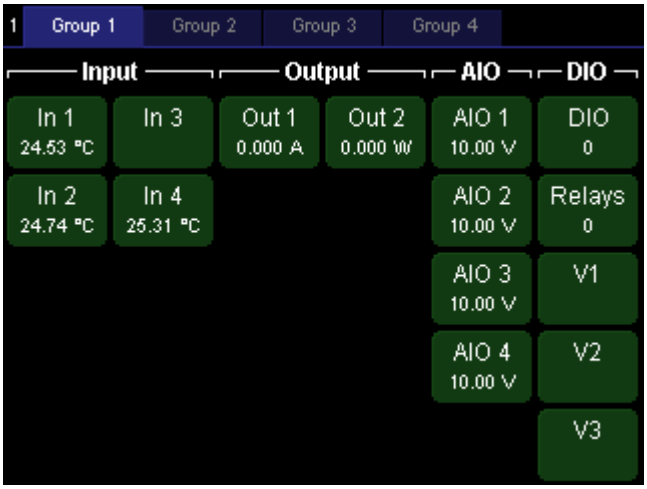
Plug the CTC100 in and turn it on with the power switch on the back of the instrument.

If the CTC100 does not turn on, a fuse may have blown. Two internal fuses can be accessed by unplugging the instrument and then removing its top cover.

If the instrument boots up but does not respond to button or touchscreen presses, the most likely cause is that the ground prong of the electrical socket is not connected to ground. A three-prong plug must be used and all three prongs must be connected.

The Select screen

The CTC100 boots up in the "Select" screen, which has a button for each physical input or output on the CTC's back panel, arranged in roughly the same order as the connectors on the back panel. There are also buttons for three virtual channels, V1 – V3, that don't correspond to back-panel connectors.



The AIO column shows the four analog I/O channels, while the DIO column shows the digital I/O channel, the alarm relays, and the three virtual channels (V1, V2, and V3) that can be used to perform real-time calculations.

The Select screen controls which channels appear on the Numeric, Plot, and Channel Setup screens. To select a channel, touch a button on the Select screen; the button becomes lighter, indicating that the channel is selected. Touch the button again to deselect the channel.

Configure the sensor inputs

Select the sensor type and calibration curve as follows:

1. Press the "Select" key and select one or more inputs.
2. Press the "Channel" key. The top of the screen has one tab for each selected channel. Touch one of the tabs to display the settings for that channel.
3. Touch the "Sensor" button and select the appropriate sensor type (RTD, thermistor, diode, etc.)

4. Set the Range to "Auto".
5. In the "Cal" column, touch the "Type" button and select the appropriate calibration curve.

If the sensor reading does not appear

The sensor reading is blank whenever it falls outside the limits of the calibration data or input circuitry. Normally, a blank reading means that no sensor is connected. If a sensor is in fact connected but the reading is still blank, try the following steps:

1. Make sure that the sensor is correctly connected. At a minimum, the sensor must be connected to the +I and –I pins.
2. Measure the resistance of the sensor with an ohmmeter to ensure that one of the wires is not broken.
3. Bring up the channel setup screen for the input channel and check the following settings:

Sensor: must agree with the type of sensor that is in use.
Range: set to Auto or, if a fixed range is selected, make sure it's larger than the sensor resistance.
Current: Forward, Reverse, or AC. If the current is off, no sensor reading will appear.
Cal **Type**: must agree with the type of sensor that is in use.
Cal **R0**: for RTDs only; must agree with the type of sensor that is in use.

4. Go to the System Setup screen and change the Units to "Sensor". Now the reading will appear in ohms or volts instead of degrees. Is the value correct?
5. If you're using a custom calibration table, make sure that the sensor resistance or voltage is within the range of the calibration table.
6. If you're using a resistive sensor and the reading in ohms is incorrect, remove the sensor and instead connect a resistor of about the same value to the CTC100. If the reading is still incorrect, the unit may need to be returned to SRS for recalibration.

Plot data

To plot data:

1. On the Select screen, touch the buttons to highlight the channels that you want to plot.
2. Press the Show Data key on the CTC100's front panel.

The Show Data screen has four blue tabs at the top that control the type of display:

- **Single**: all selected channels are plotted together on a single graph.
- **Multiple**: each selected channel is plotted in its own graph.
- **Ponytail**: all selected channels are plotted together on a single graph, and the traces are offset such that each channel starts at zero.
- **Custom**: selected channels are assigned to plots with the "Plot" button on the channel setup screen, described on page 68.
- **Numeric**: a numeric value is displayed for each selected channel.

To zoom in, touch anywhere within the right half of the plot. To zoom out, touch the left half of the plot (but not left of the Y axis).

You can also drag the plot left or right to see older or newer data; the words "X lock" appear in the bottom-left corner of the screen to indicate that the graph is no longer automatically scrolling to show new data. Touch the "X lock" indicator to return to viewing real-time data.

Optional: test the outputs

Before trying to run a PID feedback loop for the first time, it's helpful to verify that your heater is working by setting its current or power to a low value and seeing if any current flows. To do this, connect your heater to the CTC100's back panel and set the output as follows:

1. Enable the outputs by pressing the Output Enable key twice. The red Output Enable light should turn on.
2. Highlight the appropriate output channel on the Select screen, then press the Channel key to display the channel setup screen.
3. If you selected one of the ±10V analog I/O channels to drive your heater, make the channel an output by changing its IO type to "meas out".
4. Touch the Value button and enter a small value for the current or power (one that won't damage your system).
5. The Value button should display the value that you entered. If it's blank or displays zero, the CTC100 is not detecting the heater.
6. Use a temperature sensor to verify that your heater is warming up.
7. To turn the current off, touch the Off button on the channel setup screen.

If the heater doesn't start warming up, try the following:

- Verify that the heater leads are not shorted to ground or to each other.
- Measure the heater resistance with a multimeter and make sure that it's between 10Ω and about 1kΩ. Higher resistances are acceptable but the heater may not get very hot. Lower resistances may cause the CTC100 to overheat.
- Display the heater resistance: go to the System screen and, in the "Display" column, touch the "Monitors" button and select "Show". Return to the Select screen. Underneath the heater power there should now be buttons for heater current ("Out 1 I" and "Out 2 I"), voltage ("Out 1 V"), and resistance ("Out 1 R"). Turn the heater on again. Is the heater resistance the same as what you measured with a multimeter? Is the voltage pegged at 55V or the current at 2A?
- Verify that the PID mode is set to off.
- On the channel setup screen, make sure that the output's hi limit and range are both greater that the output value that you entered.

Optional: set the data logging rate

By default, the CTC100 logs three data points per second for each channel. To change this rate for all channels, press the Setup key on the front panel and then touch the blue "System" tab. Under the "Log" column, touch the "Interval" button and select from the list of available options. The log interval only affects how often data is recorded; it does not affect ADC sampling or PID feedback performance.

Each channel can be assigned its own data logging rate; see the description of the Channel.Logging control on page 68.

Optional: save data to and retrieve data from a USB memory device

If no USB memory stick or hard drive is present, the CTC100 only stores the most recent one million data points for each channel; older points are erased. Therefore, if the logging interval is 0.3 seconds per point, only the most recent 3 days of data can be displayed. In addition, all stored data is lost if the CTC100 is turned off.

A USB memory stick can be used to keep a much longer record of logged data that won't be lost if the CTC100 is turned off. Follow these steps to begin logging data to a USB storage device:

1. Plug a USB memory stick into the port on the back of the instrument. The memory stick should be freshly-formatted and completely empty.
2. Wait about 5–20 seconds until the message "Please wait while the USB drive is opened" appears on-screen. The message stays on-screen for several seconds while the log files are opened, then disappears.
3. Look for a small, dark-red circle in the upper-right corner of the screen. This is the USB logging indicator. Touch the circle. When the circle turns light, the CTC100 is saving data to the USB device. If the CTC100 is unable to write to the device, the USB logging indicator will remain dark.
4. Before turning the instrument off or removing the USB device, touch the USB logging indicator again and wait for it to turn dark. This very important step is needed to prevent damage to the USB device. If this step is skipped (e.g. if a power failure occurs while logging), the USB device should be re-formatted with a PC before using it again.

Once data has been logged to the USB memory stick, the stick will contain a log file named "\Log\00\Log00.csv" (the top-level directory name, "Log", can be changed from the front panel). The file is an ASCII CSV (comma-separated value) file that can be read with spreadsheet software.

A software package that can be downloaded from the SRS website (www.thinksrs.com, click Downloads > Software) includes a program that plots log files. No installation is required; just double-click the "FileGrapher" icon or drag a log file onto it.

The maximum size of a log file is 2GB for FAT-formatted USB devices or 4 GB for FAT32 devices. The file can reach this size in as little as 6 days, but more typically it should take about 30 days. When the file fills up, Log00.csv is closed and the instrument automatically begins writing data to a new file named Log01.csv. If Log99.csv fills up, the instrument begins logging to a new directory named "\Log\01" (where "Log" is the top-level directory name).

Optional: interface with a computer

The System setup menu, which can be displayed by pressing the "Setup" button on the CTC100's front panel and touching the blue "System" tab, has controls for setting up the CTC100's RS-232, USB, GPIB, and Ethernet interfaces (under the "COM" and "IP" columns).

The USB port uses an FTDI USB chip. If the PC asks for a driver, follow these instructions:

Install the USB driver

1. Download and install the appropriate FTDI Virtual Com Port (VCP) driver from:

   http://www.ftdichip.com/Drivers/VCP.htm

2. Optional: the default baud rate for the FTDI driver is 9600. For faster communication, change the baud rate to 230400 on both the PC and the CTC100. On the CTC100, press the Setup key, touch the System tag, and select the "USB" button. Select 230400. On the PC, change the baud rate setting in whatever software you're using; if there's no such setting,

open the Device Manager and expand the Ports line. Plug the CTC100 into the PC and look for a new port to appear. Double-click that port, select the Port Settings tab, and change "bits per second" to 230400.

3. The CTC100 should now appear as a COM port on your computer, and your programs can use the USB connection in the same way that they use an RS-232 connection.

Read data from the CTC100

All RS-232, GPIB, USB, and Ethernet messages sent to the CTC100 must end with a linefeed (decimal 10 = hex 0x0a = '\n'). The CTC100 will not process the message until the linefeed is received. Instructions are not case-sensitive.

The most recent value (i.e., the value read at the most recent ADC conversion) of a single channel can be queried by sending the name of the channel, followed by a question mark. Omit any spaces from the channel's name. For example, to query the value of channel "Out 1", send the command `Out1?`. The CTC100 replies with a value such as "0.00000":

```
Out1?
0.00000
```

The most recent value of all channels can be retrieved with a single `getOutput` instruction (the question mark is optional in this case):

```
getOutput?
29.98424, 25.86019, 27.49236, NaN, NaN, NaN, 0.000000, 10.04576,
10.04574, 10.04572, NaN, NaN, NaN, 0, 0
```

Sensors that are disconnected or out of range report a value of "NaN" (not a number). To determine the order in which the channels are listed, send a `getOutputNames` query:

```
getOutputNames?
In 1, In 3, In 2, In 4, Out 1, Out 2, AIO 1, AIO 2, AIO 3, AIO 4,
V1, V2, V3, DIO, Relays
```

Data can also be read with the `getLog` instruction, which returns logged data. Logged values are typically the average of three ADC conversions and have less noise than the values returned by `getOutput` and *`channel`*`?`, both of which return the result from the single most recent ADC conversion only. In addition, `getLog` makes it easier to retrieve data at consistent time intervals. For example, begin by sending this command, which retrieves the last point in channel In 1's log:

```
getLog "In 1", last
27.53936
```

Next, send the following command:

```
getLog "In 1", next
27.57375
```

Each time this command is sent, the CTC100 sends the next data point in the log, if necessary waiting for a new point to be added.

Control a temperature

The CTC100 can control the temperature of one or more external devices using a resistive heater and a temperature sensor.

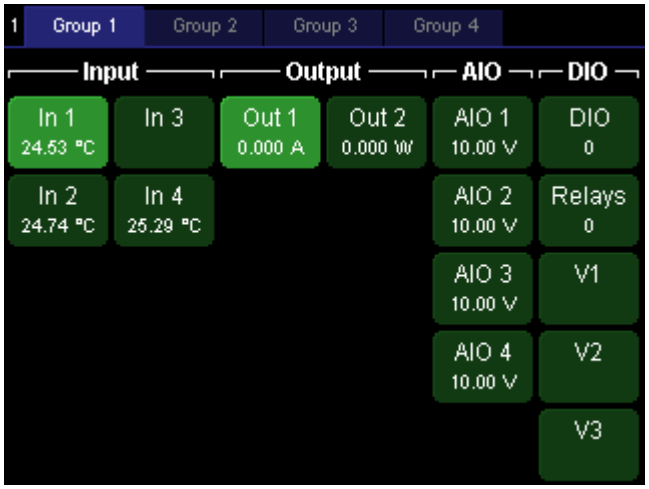
Each of the CTC100's output channels can use proportional-integral-differential (PID) feedback software to monitor a temperature sensor and determine how much power to send to the heater. The PID feedback has three adjustable gain factors that determine how much and how quickly the heater power is adjusted if the temperature deviates from its desired value. These gain factors must be properly set before the CTC100 can control the temperature of your system.

Start by plugging the heater and temperature sensor into the CTC100's back panel. The sensor must be in thermal contact with the heater — the better the thermal contact is, the more precise the temperature control will be.

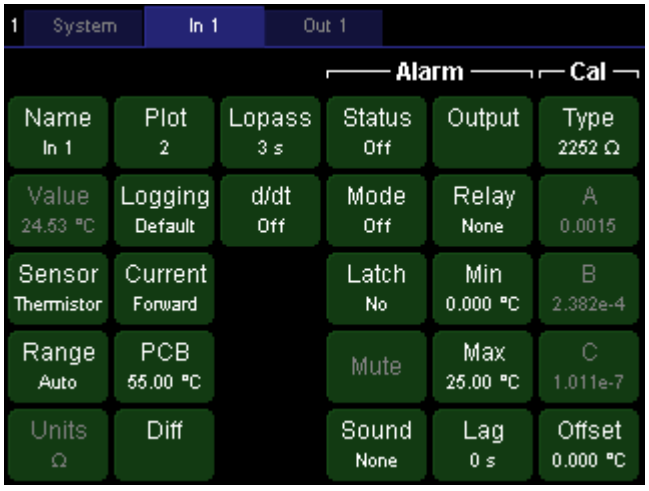
Optional: enable the lowpass filter

The feedback loop is usually much more stable if the temperature reading is lowpass filtered.

Select the channel to be filtered by pressing the "Select Channels" key, then touching the feedback temperature channel to highlight it. You could also select the output channel at this point, since you'll need to set it up next. If any other channels are highlighted, touch them to de-select them. In this example, temperature sensor "In 1" and heater "Out 1" have been selected:



Now press the "Setup" key. At the top of the screen are three blue tabs: a "System" tab for settings like the RS-232 baud rate and the display brightness, and a tab for each channel you selected. Touch the "In 1" tab.



A list of options for temperature input "In 1" is displayed. The upper-left green button, for example, shows the name of the channel; the name can be changed by touching the button. The "Value" button shows the sensor reading, but since the sensor is an input, its value can't be changed from the front panel and therefore the button is greyed out.

Touch the "Lopass" button in the third column. A list of lowpass filter time constants appears. To get more information about the Lopass setting, press the "Help" key, which displays a pop-up window with a brief description of whatever is showing on the screen. Touch the "OK" button or press the Help key again to dismiss the help window.

In the Lopass menu, select the value that is closest to your heater's response time. By reducing noise, the filter improves the accuracy of the PID tuning process and the performance of the tuned PID feedback loop. The larger the lowpass value is, the less noise there will be; however, a value larger than the heater's response time will slow down the feedback.

Optional: configure the alarm

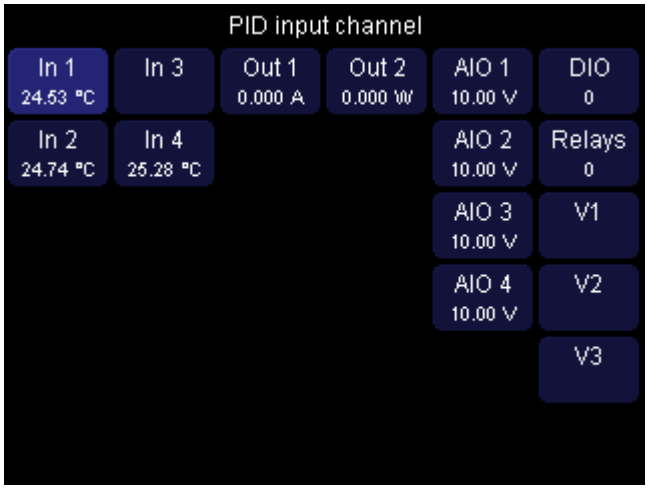
To protect your system from being damaged by excessive heater power (which can occur if, for example, the PID feedback is configured incorrectly or the sensor becomes disconnected), it's important to set up an alarm. The alarm automatically shuts off the heater whenever the temperature exceeds limits that you specify, whenever the sensor becomes disconnected, and whenever the temperature becomes too high or low for the sensor to measure. On the Setup screen for channel In 1, under the Alarm heading, set the options as follows:

- **Mode**: Set to "on" to enable the alarm.
- **Latch**: Set to "no". A latching alarm, once triggered, must be turned off manually.
- **Sound**: 1 beep.
- **Output**: select the heater output channel. Whatever channel you select will be forced to zero whenever the alarm is beeping.
- **Relay**: For the best possible security, the output should be routed through one of the four relays (A, B, C, or D) and the Relay button should be set to A, B, C, or D accordingly. The relay will physically disconnect the heater whenever the alarm is tripped.
- **Min**: Should be set well below the lowest temperature that could normally be observed; the min setting should only be exceeded if something is wrong with the sensor.
- **Max**: Set to the upper temperature limit of your system.
- **Lag**: Set to 1 s. This will prevent small glitches, such as those caused by autoranging, from triggering the alarm.

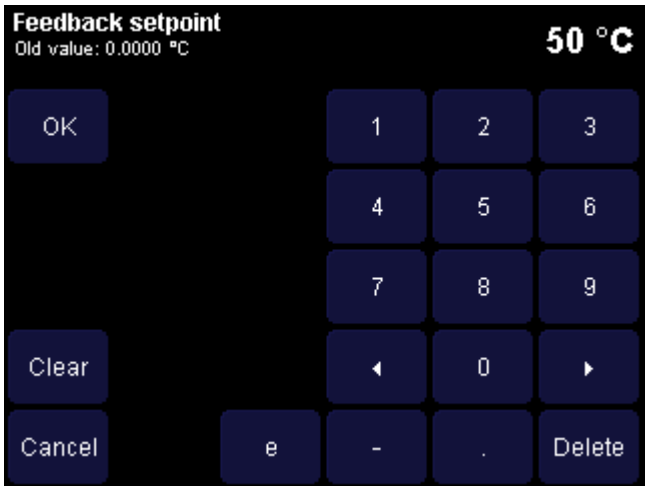
Configure the PID feedback loop

The next step is to tell the instrument which temperature sensor to control and the desired temperature of that sensor.

Make sure the "Channel" screen is still visible. Touch the "Out 1" tab to bring up the setup screen for channel Out 1. In the first "PID" column, touch the "Input" button. Then, on the list of channels that appears, touch the temperature input channel "In 1". This tells the CTC100 that we want heater Out 1 to control the temperature of sensor In 1.



Next, touch the "Setpoint" button and enter the desired temperature. Touch "OK" once you've entered the setpoint. Since the feedback is still disabled, the CTC100 won't actually turn the heater on yet.



Configure the feedback autotuner

The feedback tuner changes the heater output and measures how much the temperature changes in response. Before using the tuner, the CTC100 needs to be told how much the heater output should be changed and how long it should wait for the temperature to change.

Go to the Channel setup screen for the heater channel and look at the "Tune" column. If the output is increased to the value shown in "Step Y", would you expect to see a noticeable rise in temperature within the time shown in "Lag"? Would the amount of power shown in "Step Y" damage your system? Change these values if necessary.

Touch "D" and set the derivative gain to 1. Any nonzero value tells the tuner to enable derivative feedback, which makes the feedback faster but can also add noise. If D is set to zero, the tuner uses a different tuning algorithm that leaves derivative feedback disabled. This is sometimes necessary to reduce noise in the feedback output.

Start the feedback autotuner

If the system has never been tuned, start with the feedback turned off and the heater at ambient temperature. If the system has been tuned before, it's better to enable the feedback and wait for the temperature to stabilize at the setpoint. In either case, it's important to start with a stable temperature.

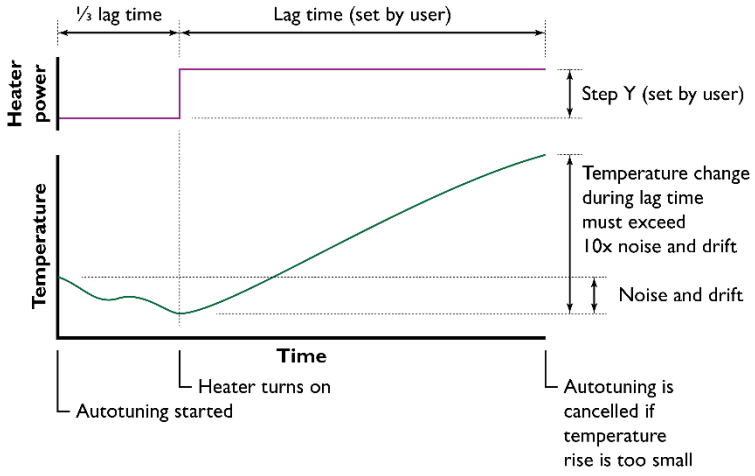
If starting at ambient temperature, ensure that the feedback is turned off before enabling the outputs: select the output channel on the "Select Channels" screen, then press the Setup key, select the tab for the output channel, and touch the "Off" button.

If the outputs are disabled, enable them by pressing the "Output Enable" key twice. The red Output Enable LED turns on and the CTC100 beeps (if pressed again, the Output Enable key immediately turns all the CTC100's outputs off; inputs are not affected).

In the PID menu, set the Mode to "auto" to start the autotuner. A status window appears and is updated every few seconds.

Before it begins the actual tuning process, the CTC100 verifies that the Step Y and Lag settings actually change the temperature by a significant amount, and cancels tuning if they don't. It first locks the heater output at its current value for one-third of the Lag time and measures how much the temperature changes. This is the noise and drift. It then increases the heater output by Step Y and waits for the Lag time to elapse. If the temperature doesn't change by at least ten times the noise and drift, autotuning is automatically cancelled, the heater output is returned to its original value, and no changes are made to the feedback gains. If the Status window is showing, it displays a message that says "Tuning was cancelled because the response was less than 10 times the noise and drift". The cause of the problem may be one or more of the following:

- Raising the heater output by Step Y didn't significantly increase the temperature, in which case Step Y should be increased;
- The Lag time wasn't long enough for the heater to respond, in which case it should be increased; or
- The temperature was drifting up or down as the test began, in which case the heater should be left to stabilize at room temperature before trying to tune again.



During the first stage of autotuning, the CTC100 compares how much the temperature changes before vs. after turning the heater on. Autotuning is cancelled if the temperature doesn't change 10x more after the heater is turned on.

It's OK to dismiss the status window or otherwise use the CTC100 during tuning; tuning will continue unless you turn tuning off or disable the outputs with the Output Enable key. In fact, it's a good idea to display a graph of heater output and temperature while autotuning (select the two channels, press the "Show Data" key, and select the "Multiple" tab). To see the status message again, press the "Setup" key, select the tab for the output channel, and then touch the "Status" button in the bottom-right corner.

When tuning is finished, the CTC100 beeps and the PID feedback is automatically enabled. If the temperature is still below the setpoint, the CTC100 starts increasing power to the heater. The temperature may overshoot the setpoint, but should eventually stabilize at the setpoint.

The feedback should be re-tuned when:

- There is a large change in the temperature
- The temperature units are changed (e.g. from °C to °F)
- The output units are changed (e.g. from W to A)
- The lowpass filter setting is changed

Acquiring and logging data

Input filters

The CTC100 has several filters that can be used to process sensor readings. Except for the sensor calibration, the filters are disabled by default and can be enabled by the user. In the order that they're applied, the filters are:

1. Sensor calibration (converts sensor reading in ohms, volts, etc. to temperature)
2. Follow filter (virtual channels only; makes the value equal to another channel)
3. Offset/gain (multiplies a channel by a gain and adds an offset)
4. Difference (subtracts the value of another channel)
5. Lowpass (filters out noise)
6. Derivative (takes the derivative of the signal with respect to time)

The filters can interact with each other. For example:

- If the settings of filters 1–4 are changed and the lowpass filter is enabled, the effect of the new setting on the sensor reading is lowpass filtered.
- Changing the gain may have unpredictable results if the difference filter is enabled, and changing the offset has no effect if the derivative filter is enabled.
- Custom calibration tables have no effect if the follow filter is enabled.

Custom calibration tables

If none of the built-in sensor calibration curves are appropriate for your sensors, a custom calibration table can be used to convert the raw sensor reading in ohms, volts, etc. to a temperature. Each channel can have its own custom calibration table.

Custom calibration tables can be loaded with a remote command or with a USB memory stick.

Format of custom calibration tables

Here's a calibration table for a platinum RTD that has a resistance of 100 Ω at 0°C:

```
units = °C
  0, 100.00
 10, 103.90
 20, 107.79
 30, 111.67
 40, 115.54
 50, 119.40
 60, 123.24
 70, 127.08
 80, 130.90
 90, 134.71
100, 138.51
```

Units: The table optionally starts with the word "units" followed by the units that this table converts readings to. If the units line is omitted, the units are assumed to be Kelvins. The units can be any string of 4 or fewer characters but must not contain any spaces, periods, plus or minus signs, or numbers. To type the degree sign on Windows computers, hold down the alt key and type "0176" on the number pad.

If the units are "°C", "°F", "K", or "mK", sensor readings will be converted to the units specified by the System.Display.Units setting. If any other units are specified, System.Display.Units is ignored and has no effect on the channel's value.

All other text before the first numeric value is ignored.

Calibration data: The data contains pairs of numeric values: the value to be displayed, followed by the raw sensor reading. For example, the sample table above tells the CTC100 to display 0 °C when the sensor resistance is 100 Ω.

The numeric values may be separated from each other with one or more commas, spaces, tabs, and/or newlines. You don't actually need to put each calibration point on a separate line as in the example.

The first value in each pair must be expressed in the units declared at the beginning of the calibration table, or in Kelvins if no units are declared.

The second value must be in ohms for resistive sensors or volts for diode sensors and analog I/O channels. For heater driver channels, the native units are by default watts, but can be changed to percent, volts, or amps with the "Units" control in the Channel menu. If in doubt, look at which units the reading is displayed in when System.Display.Units is set to "Sensor".

A calibration table must have at least two calibration points, and the entire file can't have more than 16384 characters (about 400–800 calibration points). Commas shouldn't be used within numeric values.

The data points don't have to be equally spaced; they can be closely spaced in critical temperature areas and more widely spaced in outlying areas. For RTDs, the interval between points should be 10°C or less to ensure the best possible (0.1 mK) interpolation accuracy. For thermistors, an interval of 1°C or less should be used.

The displayed value must constantly increase or decrease throughout the entire file, and there can't be two identical values. Likewise, the measured value must also increase or decrease monotonically. However, the displayed and measured values can go in opposite directions.

The calibration data must cover at least the entire expected range of measurements. When readings fall outside the range of the calibration file, no data appears on the display, and any PID feedback loops that use the affected channel are frozen.

Furthermore, while a cubic spline algorithm is normally used to interpolate between the calibration points, between the first and last two calibration points a less accurate linear interpolation algorithm is used. So the calibration table should ideally extend at least one point above and below the expected range of sensor measurements.

The order of the data points can be reversed (measured value first, displayed value second) by adding a tilde (~) to the beginning of the file. The tilde must be the first character in the file, appearing before the units declaration and any other header information.

How to load custom calibration tables

USB memory stick: create a calibration table as described above and save it as a text file. The name of the file should be the name of the channel, with any spaces removed, plus ".txt". The name of the channel cannot contain more than 8 characters, not including spaces. Create a directory named "cal" within the top-level directory of a USB storage device, and copy the .txt file into the directory. The calibration files are automatically loaded when the storage device is plugged into the instrument.

Remote command: send the command:

```
`Customcal "<channel>" "<calibration data>"
```

where <channel> is a channel name and <calibration data> is the calibration information as described above. The command is not case sensitive. The first character is a "backtick", which on US keyboards is to the left of the "1" key. If the channel name contains any spaces, the spaces cannot be omitted and the channel name must be in quotes. The calibration data must be in quotes. The data must be on a single line and the entire command not contain more than 4092 characters. For example:

```
`Customcal "In 2", "10.0, 0.012, 20.0, 0.024, ..."
```

How to verify that a custom calibration table is loaded

If a channel uses a custom calibration, the channel's name appears in bold type on the Select screen. For more details, go to the Channel Setup screen and press the Channel.Cal.Details button to see the first and last three data points in the custom calibration, or, if the calibration couldn't be read, a description of the problem.

If the calibration file can't be read, no readings appear for the affected channel. This occurs if the file has any values after the header with no numeric characters, if the values are not monotonically increasing or decreasing, or if the file ends with a temperature value.

Once a custom calibration table is loaded, it remains in effect until one of the following occurs:

- A different calibration table is loaded onto the same channel, using either a USB memory stick or remote command.
- The calibration type (set with the Channel.Cal.Type button) is changed to "standard".

The following have no effect on custom calibration tables:

- Unplugging the USB device with the calibration tables.
- Plugging in a USB device that does not contain a calibration file for the channel.
- Turning the CTC100 off and back on again.

Virtual channels

The CTC100 has three virtual channels named V1, V2, and V3. You can set these channels to any value or make their value follow the value of another channel. The value can then be graphed or saved to the log like any other channel. Here are some uses of virtual channels:

- Show the rate of change of a temperature by setting "Follow" to the desired temperature channel and "d/dt" to "On". You could just do this in the original channel, but using a virtual channel lets you display both the absolute temperature and the rate of change side-by-side.
- Show the difference between two temperatures by setting "Follow" to one temperature channel and "Diff" to the other. Again, this can be done without virtual channels, but using a virtual channel lets you keep the original data.
- Run this macro to continuously set the value of a virtual channel to a feedback setpoint:

  ```
  [waitForSample  V1=#Out3.PID.actual]-1
  ```

  Now the setpoint can be plotted on-screen and logged to USB.
- Run this macro to show the value of channel 3A divided by the value of channel 3B:

  ```
  [waitForSample  #a=#3A  #a/=#3B  V1=#a]-1
  ```

- Run this macro to show the average of channels 3A and 3B:

```
[waitForSample  #a=#3A  #a+=#3B  #a*=0.5  V1=#a]-1
```

Logging data to internal memory

The most recent one million readings from each channel are saved in internal RAM and can be plotted on-screen or retrieved using the getLog instruction. The amount of time that the internal log covers depends on the channel's log interval:

Log interval	Time span of internal log
0.1 second	1.2 days
0.3 second	3.6 days
1 second	12.1 days
3 seconds	36.4 days
10 seconds	121 days
30 seconds	1 year
1 minute	2 years
3 minutes	6 years
10 minutes	20 years
30 minutes	60 years
1 hour	120 years

All saved readings are erased when the log interval is changed.

Data from a single sensor can be logged at two different intervals by configuring one of the virtual channels (V1, V2, or V3) to follow an existing channel (set the IO type of the virtual channel to Input, then touch the Follow button) and then setting its log interval to a different value.

Logging data to USB

The CTC100 can store data on removable USB memory devices such as USB hard drives or flash memory keys. When a USB device is plugged in, it takes the CTC100 several seconds (normally about 5 seconds) to recognize the device and for the USB logging feature to become available.

The USB memory device must be newly formatted and must not contain any files other than CTC100 calibration, log, and macro files.

A small dark red dot (the USB logging indicator) appears in the upper-right corner of the screen whenever a USB device is present. The dot turns light red if data is being logged to USB. To log data to a USB device, plug the device into the CTC, touch the dot, and wait a few seconds until it turns light. Touch the dot again to stop logging.

The instrument can't read logged data from USB memory. Data shown on the plot screen and retrieved with the getLog instruction always comes from internal memory.

Do not unplug a USB device or switch the CTC100 off while USB logging is enabled. Either of these actions causes loss of data and corruption of the device's file system. If the USB logging indicator is light red, always touch it and wait for it to darken before unplugging the USB device or turning the CTC100 off.

If a USB device is unplugged while data is being logged to it, repair the device by inserting it into a PC and running a check disk program.

ADC sampling and logged data

The CTC100 has two different sampling rate settings: one controls how often data is acquired, and another controls how often it's stored.

A/D rate

The analog-to-digital conversion, or A/D, rate controls how often the feedback loop runs.

The A/D rate is set to 100 ms (10 samples per second) and should normally not be changed because the temperature readings are less accurate at other A/D rates. However, in some cases it may be necessary to use a different A/D rate.

All channels are read at the same rate. The A/D rate mainly affects the performance of feedback loops: a faster rate means the PID loops can respond more quickly to changing temperatures, but a slower rate means less noise.

Normally, A/D conversions are synchronized with the AC line voltage, and the A/D rate can only be set to multiples of the AC line period. For example, if the rate is set to 100 ms, conversions occur every six cycles of the AC voltage if the CTC100 is plugged into a 60 Hz AC wall socket, or every five cycles for 50 Hz AC. This prevents 60 Hz noise from aliasing into temperature readings, which would cause a slow sinusoidal variation in the readings. 60 Hz noise could still create a constant offset in the temperature readings, but the offset is usually too small to be of concern with thermocouples and can be removed from RTD readings using current reversal.

The A/D conversions can be uncoupled from the line frequency by moving the "Trigger source" jumper on the motherboard to the "1 MHz clock" position (the jumper should only be moved while the system is switched off). Then the A/D rate can be set to any value between 10 and 1000 ms with a resolution of 1 μs. However, A/D conversions will no longer be perfectly synchronized with the AC line voltage, even if the rate is set to a multiple of the line period. As a result, low-frequency sinusoidal noise may appear in your temperature sensor readings. The frequency of the noise is equal to the difference between the AC line frequency and the closest multiple of the A/C conversion rate (all expressed in Hertz). For example, if the A/D rate is 10.1 Hz and the AC line frequency is 60 Hz, a sine wave with a frequency of $60 - (6 \cdot 10.1) = 0.6$ Hz may be superimposed on your temperature readings.

Log rate

The log rate only controls how often channel readings are saved, and has no effect on the feedback performance. Since the plot screen only shows logged data, faster log rates result in more detailed but noisier graphs. If data is being logged to a USB memory stick, faster log rates result in larger log files.

The log rate can be set independently for each channel; the default is a point every 0.3 seconds. If the log rate is slower than the A/D rate, multiple A/D readings are averaged together to create each logged value. If the log rate is faster, each A/D reading is recorded more than once in the log.

Format of CTC100 log files

If a USB memory stick is present and logging is enabled, log files are saved to the memory stick as comma-separated ASCII text. The first line is a comma-separated list of channel names, while each following line is a comma-separated list of values recorded at a particular time:

```
Time (ms),In 1,In 2,PCB 1,In 3,In 4,PCB 2,Out 1,Out 1 I,Out 1 V,
Out 1 R,PCB 3,Out 2,Out 2 I,Out 2 V,Out 2 R,PCB 4,AIO 1,AIO 2,
AIO 3,AIO 4,DIO,Relays,V1,V2,V3
1635152672601,25.138351,,,,24.815240,,27.178291,,,,,0.0000000,,,,,
-0.0025160717,-0.0022943627,0.0037143015,0.0011772792,0.0000000,
```

```
0.0000000,,,
1635152672901,25.164306,,,,24.815740,,27.060104,,,,,0.0000000,,,,,
-0.0025366563,-0.0022110757,0.0037003612,0.0011041848,0.0000000,
0.0000000,,,
1635152673201,25.191293,,,,24.817480,,27.035020,,,,,0.0000000,,,,,
-0.0024720549,-0.0022719798,0.0037023611,0.0011656382,0.0000000,
0.0000000,,,
```

The first value in each line is the time in milliseconds since 1970. A utility program, PTCFileConverter, is available that can replace the time in milliseconds with time and date or elapsed time.

The data readings are saved in whatever units the CTC100 is set to. If the CTC100's units are changed while logging, there will be a discontinuity in the data.

The log file includes placeholders for PCB temperatures (PCB 1, PCB 2, etc.) as well as heater current, voltage, and resistance (Out 1 I, Out 1 V, Out 1 R, etc.). Normally these values are left empty; they're only recorded if the System > Monitors control is set to "Show" or "Hide".

If a sensor was disconnected, its reading was out of range, or no reading was acquired at the indicated time, then its value is empty and two consecutive commas appear.

The size of a log file is limited to 2 GB for FAT-formatted USB devices or 4 GB for FAT32. When a log file reaches its maximum size, the CTC100 automatically closes the file and starts a new one. The default names of the files are "\Log\00\Log00.csv", "\Log\00\Log01.csv", etc.

The system fan

The system fan regulates the temperature of the CTC100's input and output circuit boards.

Each of the two input cards has a Channel.PCB setting that sets the highest temperature it's allowed to reach before the fan turns on; if the temperature is below this setting, the card's temperature is unregulated.

The Channel.PCB setting of one card should ideally be just below its normal temperature, so that the fan is always running and the card's temperature is continuously kept at the Channel.PCB value.

The fan speed can be overridden with the System.Other.Fan remote command. However, if the fan speed is too low, the CTC100 can overheat. Input cards are less accurate when their temperature exceeds 35°C, while if an output card exceeds 60°C, its output is automatically shut off until the temperature drops below 60°C.

It's possible to display and log the temperatures of the four I/O cards by setting the System.Display.Monitors control to Show.

Besides the main system fan, the CTC100 also has an internal fan that periodically turns on to keep the main power supply cool. This fan runs even when the CTC100 is in system standby mode.

Rack mounting the CTC100

A 19-inch rack mount tray, SRS model number O100CTRM, is available for the CTC100. The tray will accept either one or two CTC100s or other half-rack instruments. Note that although the CTC100 chassis is 3U high, the rack mount tray is 4U high.

The CTC100's main vent is on the bottom of the chassis. When rack mounting the instrument, it's important to ensure that the vent is not blocked.

Using PID feedback

How stable is the CTC100's feedback?

Under ideal conditions, the CTC100's electronics and firmware can regulate temperature with a stability of 1 mK. But that's only possible if the CTC100 can both heat and cool the sample faster than any external factors can. That is:

- The heater has to produce enough heat to quickly change the temperature of the sample.
- The heat must rapidly reach and spread through the sample. The smaller the entire system is, the better.
- The sample must cool down as quickly as it can be heated up. Consider adding a fan to help cool the system or using a TEC device that can both heat and cool.
- The system must be shielded from external temperature variations. Consider using a two-stage design with the temperature-controlled system enclosed within a larger chamber that's also temperature controlled.

If the system is an environmental chamber, a flow-through configuration in which fresh air is heated as it enters the chamber (e.g. using a chassis-mount fan that has a heater attached) usually produces the best response times.

Basic concepts

The CTC100 controls temperature by supplying current, voltage, or power to a heater. The *PID feedback* algorithm continuously determines how much heater output (Y) is needed to keep the temperature at a predetermined "setpoint", even as outside factors such as the ambient temperature change how much heater output is needed to maintain that temperature.

PID feedback is a combination of three algorithms.

The **proportional** feedback algorithm determines the error E, i.e. the difference between the desired temperature (the setpoint) and the actual temperature T. The output Y_p of the proportional feedback algorithm is just the error multiplied by a constant, *P*:

$$E(t) = (\text{setpoint} - T(t))$$
$$Y_p(t) = P \cdot E(t)$$

As the actual temperature approaches the setpoint, the proportional output Y_p decreases to zero, at which point no power is supplied to the heater.

Normally, however, some power is required to keep the heater at the setpoint, which is why the **integral** feedback algorithm is needed. It multiplies the error by a constant (*I*) and adds the result to the previous integral output:

$$Y_i(t) = I \cdot E(t) + Y_i(t-1)$$

As the temperature approaches the setpoint, the rate of change of the integral output Y_i drops to zero.

Derivative feedback tries to predict what the temperature will be in the future by multiplying the rate of temperature change by a constant, *D*:

$$Y_d(t) = D * (T(t-1) - T(t))$$

If the temperature is increasing (and *D* is positive), derivative feedback reduces power to the heater; if the temperature is decreasing, derivative feedback increases power to the heater.

The output of the PID feedback loop (i.e., the heater power) is the sum of the three feedback algorithms:

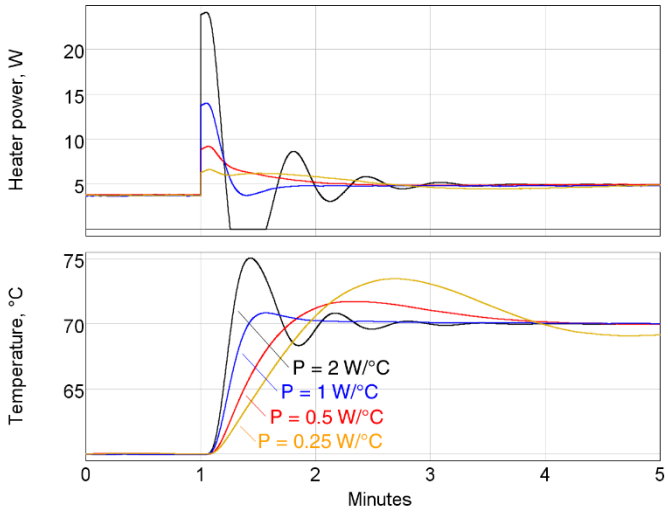
$$\text{Heater power} = Y_p(t) + Y_i(t) + Y_d(t)$$

The key challenge to using a PID feedback loop is determining the constants (or "gains") *P*, *I*, and *D,* which are different for every apparatus and must be determined experimentally. As a general rule, if the gains are too low, the feedback won't respond enough to temperature variations; if they are too high, the feedback responds too much and overshoots the setpoint, and both heater power and temperature may begin to oscillate. The faster the temperature changes in response to the heater, the larger the gains can be without causing oscillations.

Manual tuning

How do the three feedback parameters *P*, *I*, and *D* affect feedback performance?

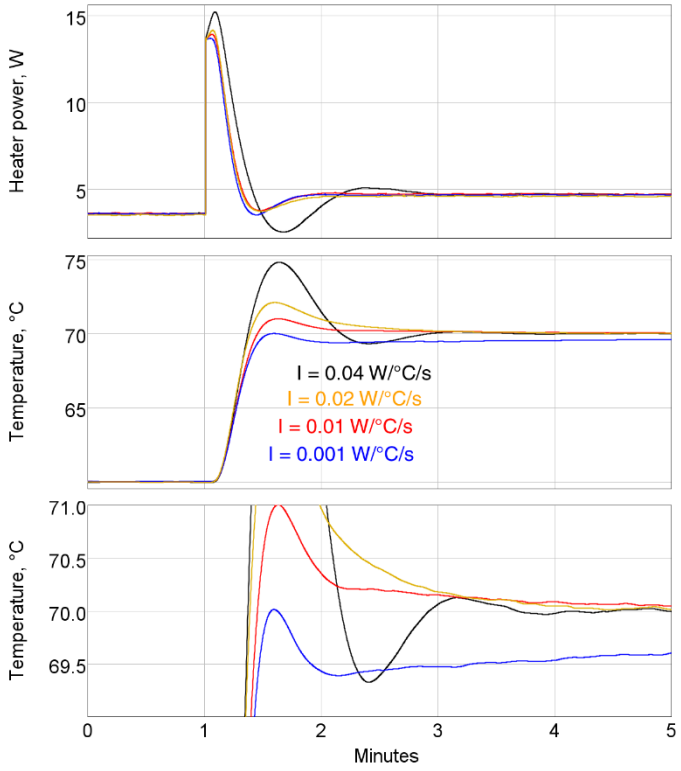
Proportional: the figure below shows the effect of changing the proportional gain *P*. The top graph shows the power being delivered to a heater by a PID feedback loop during four separate tests, while the bottom graph shows the temperature of the heater during the same tests. Each test is identical except for the value of *P*. At 1 minute, the setpoint is increased from 60 to 70°C. When *P* = 1 W/°C (second curve from top), the feedback loop exhibits a perfect response; that is, the temperature rapidly increases to 70°C with a slight overshoot that serves to minimize the settling time. If *P* is increased to 2 W/°C, the temperature responds more quickly but then overshoots the setpoint and oscillates.



Interestingly, decreasing the proportional gain to 0.5 or 0.25 W/°C also results in more overshoot and can even cause oscillations, despite the fact that the temperature rises more slowly. By reducing the proportional feedback response, we've forced the integral feedback to take more responsibility for raising the heater power — and as the next figure illustrates, the integral feedback has a greater tendency to overshoot and oscillate.

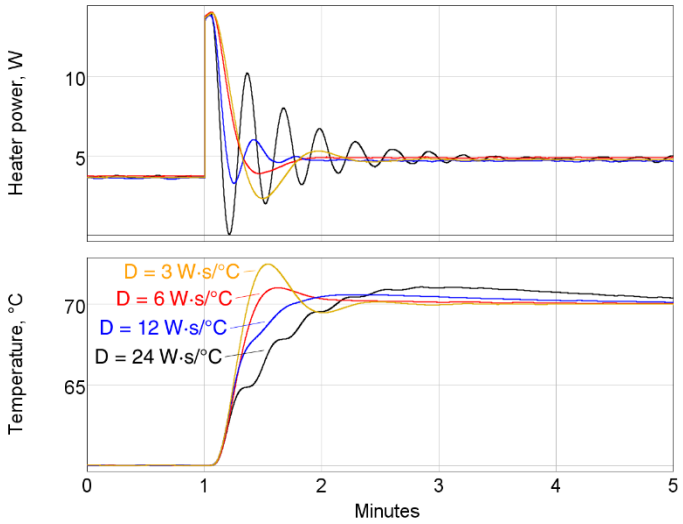
Integral: increasing the integral gain *I* also results in a larger heater response, but integral feedback is slow because it works by adjusting its previous output, rather than re-calculating its output from scratch at each feedback cycle. Therefore, integral feedback tends to overshoot the setpoint and cause oscillations.

Setting *I* to 0.001 W/°C/s minimizes overshoot. However, close inspection (see the lowest trace in the bottom graph) shows that the temperature doesn't actually reach the 70° setpoint within the time period shown. Without enough integral gain, temperature errors persist for a long period of time. As an approximate guide, the integral gain should be about one-tenth the proportional gain.



Derivative: derivative feedback reduces the heater output whenever the temperature rises rapidly. In the example below, when the derivative gain *D* is increased from 3 to 6 W·s/°C, the amount of overshoot and oscillation decreases. The temperature rise is also a little slower, but because there is less oscillation the system stabilizes at 70°C sooner.

However, if the derivative gain is too large, it can produce oscillations — because when the temperature is rising rapidly, derivative feedback reduces the heater output, which causes the temperature to rise more slowly, which makes the derivative feedback increase the heater output, and so on.

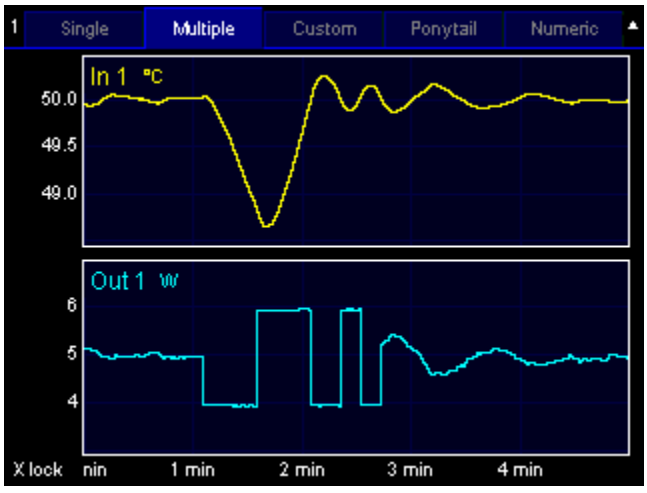


Derivative feedback allows *P* and *I* to be increased a little more, resulting in faster feedback control.

Automatic tuning algorithms

During automatic tuning, the CTC100 changes the heater power, measures how much and how quickly the temperature changes in response, and then estimates the optimum values of the gain factors *P*, *I*, and *D*. Two tuning algorithms are available: the relay tuner and the step response tuner.

Relay tuner



Temperature (top) and heater power (bottom) during relay autotuning. Step Y is 2 W, Lag is 30 s, feedback is initially on, and the system starts with the temperature stabilized at the 50°setpoint. After the tuning has finished, the feedback turns on and re-stabilizes the system at 50°C after a few cycles of oscillation.

The relay tuner creates a temperature oscillation by switching the heater between two output values:

$$\text{Output}_{high} = \text{Output}_{I} + (\text{Step Y})/2$$
$$\text{Output}_{low} = \text{Output}_{I} - (\text{Step Y})/2$$

where Output_{I} is the initial output and Step Y is the value specified in the "Step Y" control. Note that the relay tuner cannot be started unless the output is greater than (Step Y)/2. For best results, the output should be greater than Step Y.

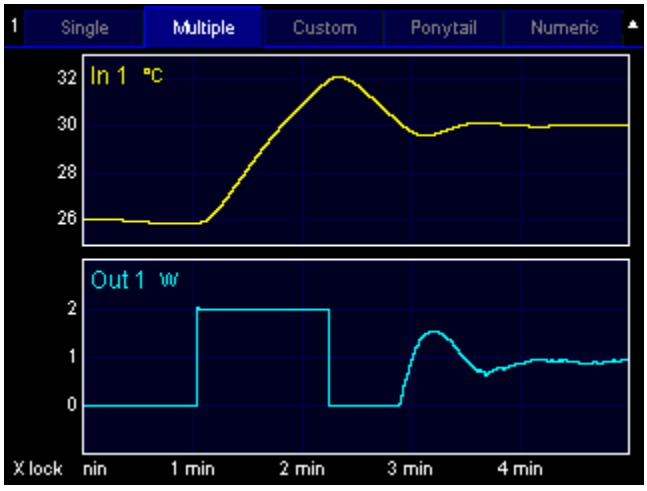
The relay tuner begins by disabling the feedback (if the feedback was on) and measuring the drift and noise of the feedback input signal in the absence of any changes to the feedback output. The drift-and-noise measurement continues for one-third the amount of time specified with the "Lag" control; the resulting drift-and-noise value is the difference between the largest and smallest input signal observed during this time.

After the drift and noise measurement, the relay tuner sets the heater output to Output_{low} for the Lag time to start the oscillation. If during this period the feedback input does not change by at least ten times the drift-and-noise value, an error message is displayed in the Status window and tuning is cancelled. If this occurs, either 1) ensure that the temperature is stable before starting the step response; 2) increase step Y; or 3) if it looks like the temperature didn't have enough time to respond, increase the Lag time.

The tuner then sets the output to the Output_{high} value. Then, each time the temperature crosses its initial value (50 °C in the figure above), the output is switched from high to low or low to high. This produces a temperature oscillation 180° out of phase with the output oscillation. The tuner performs two oscillation cycles, not including the kick start, and measures the period and amplitude of the second oscillation.

The relay tuner has to wait several times for the temperature to cross its initial value. If the temperature measurement is disturbed during this time (for example, if the temperature sensor is moved, or if the sensor is in an oven and the oven door is opened), the temperature may never cross its initial value and the tuner may run indefinitely without finishing.

Step response tuner



Temperature (top) and heater power (bottom) during step response autotuning. Step Y is 2 W, Lag is 30 s, feedback is initially off, and the system starts at room temperature. After the step response is complete, the feedback turns on and the temperature drops before stabilizing at the 30°C setpoint.

The step response tuner makes a single change to the amount of power delivered to the heater, and measures how much and how quickly the temperature changes in response.

The step response tuner begins by disabling the feedback (if the feedback was on), and measuring the drift and noise of the feedback input in the absence of any changes to the output. The drift-and-noise measurement takes one-third the period specified with the "Lag" control; the resulting drift-and-noise value is the difference between the largest and smallest input signal during this time.

Next, the step response tuner increases the output by the value specified with the "Step Y" control. The tuner then waits for the amount of time specified with the "Lag" control. If during this period the feedback input does not change by at least ten times the drift-and-noise value, an error message is displayed in the "Status" window and tuning is cancelled. If this occurs, either 1) ensure that the temperature is stable before starting the step response; or 2) increase step Y; or 3) if it looks like the temperature didn't have enough time to respond, increase the Lag time.

The tuner continuously measures the rate of temperature change. Tuning ends when the lag period has elapsed and the rate of temperature change is less than half of the fastest rate observed during the tuning process. The tuner then calculates the maximum rate of temperature change (the slope), the lag time, and the total response, and uses these values to calculate the PID gains.

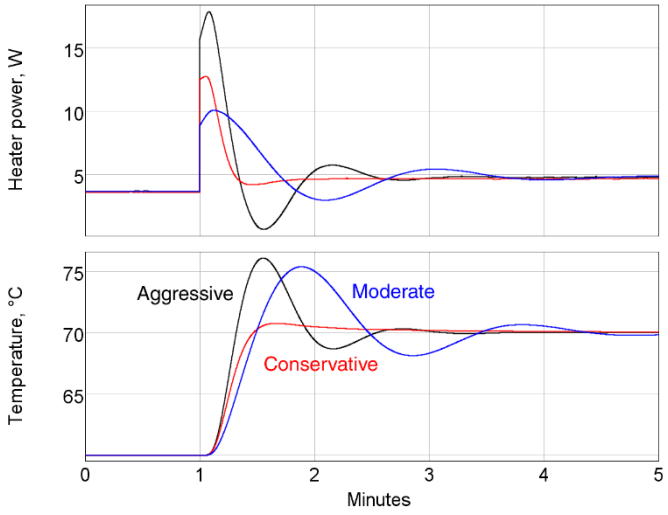
Because the slope calculation is sensitive to noise, it's important to enable the "lopass" filter on the feedback input channel to achieve accurate tuning results. Since the relay tuner does not require a slope measurement, it's less sensitive to noise than the step response tuner.

If the tuning mode is set to "Auto", the CTC100 selects the relay tuner if both its high and low outputs are within the heater's limits; otherwise, it selects the step response tuner. In particular, if the output is off when autotuning is started, the step response tuner runs because the relay tuner would require a negative out9put.

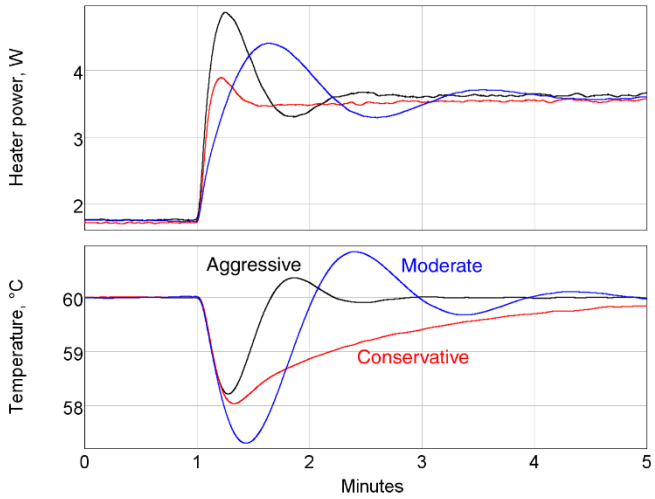
Aggressive, moderate, and conservative tuning

Both the step response and relay tuners offer aggressive, moderate, and conservative tuning options. Conservative tuning theoretically produces zero overshoot and is usually the best choice when the temperature needs to follow a changing setpoint. The aggressive tuning option theoretically produces 25% overshoot (although in fact it tends to be larger) and is usually the best choice for applications in which the setpoint is constant. Moderate tuning produces very stable feedback that behaves reasonably in a wide variety of situations.

The figure below compares the system's behavior when we change the setpoint from 60 to 70°C after relay tuning with the aggressive, moderate, and conservative options. In this case, the conservative tuning produces the best response.



However, the results are much different if we look at how the system responds to a change in the ambient temperature. The next figure shows how well the system recovers when we start blowing air over it with a fan. The setpoint is a constant 60°C. In this case, aggressive tuning produces the best response.



The actual behavior of your system might vary significantly from the behavior shown in these figures. In any event, the feedback gains determined by the automatic tuning algorithms should generally be regarded as only a starting point. In critical applications, the gains normally need to be manually adjusted to achieve good feedback performance.

Using the automatic tuner

Start with a stable temperature. Before the autotuner is started, the temperature must be stable. If the system hasn't been tuned before, the easiest way to get a stable temperature is to let the system sit undisturbed for a long period of time with the heater off. On the other hand, if the PID gains have been set before and just need to be re-optimized, it may be easier to turn the feedback loop on and let the feedback stabilize the temperature. The autotuner can be started with the feedback either on or off.

Disable or enable derivative feedback. Because derivative feedback can amplify sensor noise, it may sometimes be preferable to disable it. If the derivative feedback gain is set to zero before autotuning begins, derivative feedback is disabled and the autotuner calculates P and I feedback gains, leaving the derivative feedback set to zero. In contrast, if the derivative feedback gain is initially nonzero, the autotuner calculates P, I, and D feedback gains using a more aggressive algorithm. Therefore, setting D to a nonzero value (the exact value doesn't matter) before autotuning produces faster-acting feedback but more noise. If your temperature sensor is noisy or you're not using a lowpass filter, leave D set to zero.

Set the step size and lag time. "Step Y" controls how much the CTC100 increases the heater output, and "Lag" controls how long the CTC100 waits for a response. If either value is too small, the CTC100 may, after attempting to tune, display a message saying that there was an insufficient response. If the values are too large, tuning will take longer than necessary and your heater will get excessively hot.

Start tuning. To begin tuning, go to the channel setup screen and set the tuning mode to "Auto".

If the tuner finishes successfully, a high-pitched tone plays and the feedback loop starts running. If the tuner was unsuccessful (usually because Output Enable was off, the heater was unplugged, the

temperature sensor was unplugged, the heater was out of range, or the response was insufficient), a low-pitched tone plays and the feedback is turned off.

When PID tuning is started, a window with information about the autotuner's progress appears. This window can be dismissed by touching the "OK" button or any menu key. Dismissing the window does not cancel autotuning; to cancel autotuning, either 1) set the tuning Mode control to "Off"; 2) touch the output channel's "Off" button; or 3) disable all outputs by pressing the "Output Enable" key.

If the status window is dismissed, it can be shown again by touching the "Status" button in the output's "Channel" menu.

Automatic tuner error messages

One of the following messages appears in the Tuning Status window if tuning was unsuccessful. If you don't see a message, press the Channel > Tune > Status button to see it.

Tuning was cancelled because the response was less than 10 times the noise and drift

This message indicates that the heater didn't produce a large enough temperature change. It can result from any of these conditions:

- The temperature was not stable before the autotuner was started, or the temperature was changed by some external factor after the autotuner was started. In particular, after running the autotuner, it's necessary to wait for the temperature to re-stabilize before running the autotuner again.
- The autotuner disturbance size (Step Y) was not large enough to create a noticeable change in the temperature.
- The autotuner wait time (Lag) was not long enough for the heater to change the temperature.

To determine the source of the problem, look at a dual plot with the heater output on one plot and the sensor temperature on the other. Make sure that the temperature was stable before the heater turned on and that it changed significantly after the heater was turned on.

Autotuning was cancelled because the PID mode was set to "Off"

The user turned off PID feedback while the tuner was running. The tuner is unable to run when PID feedback is turned off.

Autotuning was cancelled because the PID mode was set to "Follow"

The user changed the PID mode to "Follow" after autotuning started. The tuner is unable to run in this mode.

Autotuning was cancelled because the tuning mode was set to "Off"

Indicates that the user turned off autotuning while it was running.

Tuning was cancelled because the input was disconnected

No sensor signal was detected during the tuning process. Ensure that a sensor is plugged in and that its reading is not blank. If the reading is blank, an incorrect sensor range, sensor type, or calibration may be selected.

Unable to tune feedback because the outputs are disabled. Press the Output Enable button to enable outputs.

The outputs must be enabled before autotuning, or else the CTC100 will not be able to provide any power to the heaters.

Unable to tune feedback because the heater is disconnected

This message appears when the heater is connected to channels Out 1 or Out 2 and the measured heater resistance is less than 1 Ω or greater than 10 kΩ.

Unable to tune feedback due to a hardware fault in the heater output

This message appears when the heater is connected to channels Out 1 or Out 2 and one of the following conditions occurs:

- The current at the + and – terminals is not equal
- Current was detected when the heater was supposed to be off
- The measured current differs from the expected current.
- The card's internal power supply is not at its specified voltage.

Make sure that the heater is not shorted to ground or to another power supply. In some cases, this error message may indicate that the output circuit has been damaged.

Unable to tune feedback because the heater is under range
Unable to tune feedback because the heater is over range

During step response tuning, the heater output is increased by the amount shown in the "Step Y" button. During relay tuning, the heater output is increased and then decreased by half of the Step Y value. If the heater output is expected to exceed its maximum or minimum values, an error message appears. The message appears before the heater output is changed.

Suggestions for best tuning results

- While tuning, use the "Plot" display to graph the heater output and the temperature on separate graphs. Make sure that you can see the temperature begin to rise or fall after the heater output changes.
- If tuning fails, let the temperature stabilize and try increasing the step Y or lag before attempting to tune again. You may also need to increase the lowpass filter time constant.
- The temperature must be stable when tuning is started. Either the feedback must be running and stabilized at the setpoint, or the heater must be off and the temperature stabilized at the ambient temperature.
- Set the lowpass filter on the input (temperature) channel to a value just below the expected response time of the system. The step response tuner in particular requires adequate lowpass filtering to produce accurate results.
- Make sure the system doesn't experience any temperature disturbances during the tuning process.
- Since the ideal feedback parameters usually vary with temperature, run the tuning algorithm at about the temperature at which the feedback will be used. If the system has never been tuned before you may need to tune at room temperature, then let the feedback bring the system to its working temperature, and re-tune at the working temperature.
- The autotuning algorithm assumes that the temperature is a linear function of heater power. In most cases it isn't, which means that the results produced by the algorithm may not be perfectly accurate and may need to be manually adjusted.

Using alarms with PID feedback loops

By default, the PID heater output is frozen whenever the sensor becomes disconnected or goes out of range. In some cases this can lead to uncontrolled heating or cooling of the sample. For example, if the feedback setpoint is set to 200 degrees but the sensor can only measure temperatures as high as 100 degrees, the CTC100 will continue heating the sample indefinitely.

Each input channel has an alarm that can be used to prevent such runaway heating. When properly configured, alarms set the heater output to zero whenever the sensor is disconnected, out of range, or the temperature exceeds limits that you specify. Alarms should be set up whenever the heater is capable of providing enough heat to damage your system.

Front-panel controls

The front panel has four menu keys labeled "Select Channels", "Show Data", "Program", and "Setup". These keys can be pressed at any time to display one of the four main screens. The front panel also has a "Help" button that displays help text for whatever is currently on-screen, and an "Output Enable" button that turns all the CTC100's outputs on and off.

Most front-panel controls have an equivalent remote command that performs the same function over RS-232, USB, GPIB, or Ethernet. For quick reference the equivalent remote command is listed after the name of each front-panel control. Parts of the command that are in italics should not be entered literally but should be replaced with an appropriate value. See the "Programming" section for more information on remote commands.

USB logging indicator

`System.log.logTo { RAM, USB, None }`

If a USB memory stick is plugged into the CTC100, a small dark-red dot appears in the upper-right corner of the screen. Touch the dot to turn USB logging on; the dot will turn light red to indicate that data is being logged to USB. Touch the dot again to turn USB logging off.

Removing the USB memory device or powering down the CTC100 while USB logging is on causes loss of data and corruption of the memory device. A corrupted device should be repaired by plugging it into a PC and running a program such as chkdsk (Windows), fsck (Linux), or Disk Utility (Macintosh).

"Help" key

Command`.help`

The "Help" key displays a popup screen that provides more information about whatever is currently visible on the CTC100's display. Some screens that don't accept any user input may not have any Help information.

"Output Enable" key

`OutputEnable { on, off }`

After the CTC is turned on, its inputs function normally but its outputs are disabled. This safety feature gives you a chance to adjust the CTC100's settings before it begins to provide power to the heaters. To turn on the outputs, press the "Output Enable" key twice. A red light next to the "Output Enable" key turns on to indicate that the outputs are active, and any PID feedback loops that were previously running begin to provide power to the heaters.

If the outputs are enabled, pressing the Output Enable key once disables all outputs, setting them to zero. Inputs continue to function normally. In an emergency situation, the Output Enable key is the quickest way to turn off the CTC100's outputs. Re-enabling the outputs immediately returns all outputs to their previous values.

It's possible to use a startup macro to make the CTC100 power up with the outputs enabled; see the "Startup macros" section.

The Output Enable key is not intended to prevent electric shocks. When handling exposed heater wires, always disconnect the wires from the CTC100 or unplug the CTC100 from the wall.

Press and hold the Output Enable key for 3 seconds to put the CTC100 into standby mode. In standby mode, the outputs are turned off, data acquisition and macros are paused, the front panel display and system fan are shut off, and the system does not respond to remote commands. RTD excitation currents are still on, and an internal cooling fan may switch on occasionally. Press the

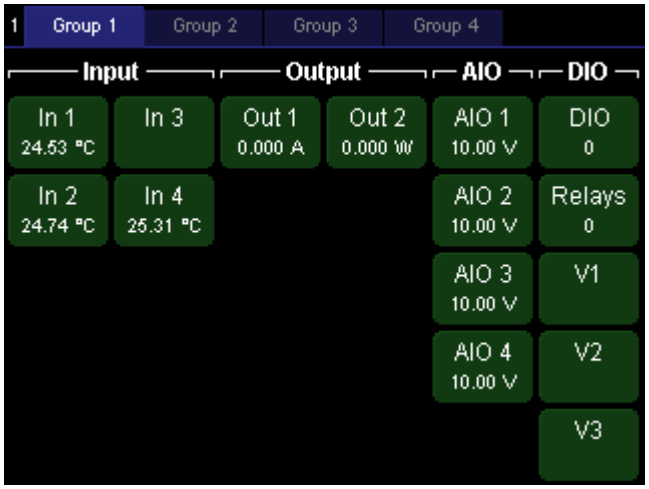
Output Enable key again to leave standby mode. There is no remote command for entering or exiting standby mode.

"Select Channels" screen

In 1, In 2, etc.

Channel.selected { on, off }

Each of the green buttons on the "Select Channels" screen represents one I/O channel. The buttons are arranged in roughly the same order as the connectors on the back of the CTC100. Each button shows the channel's name and current value. If no sensor or heater is connected, the value may be blank.



Buttons turn red whenever the channel's alarm is triggered. The name of the channel appears in bold if the channel uses a custom calibration table.

Touch one or more buttons to select channels to view on the Show Data and Setup screens.

Group 1, Group 2, etc.

Group { 1, 2, 3, 4 }

The top of the Select Channels screen has four blue tabs that can be used to save and recall groups of selected channels. Touch one of these Group tabs to select the channels in the group (all other channels are de-selected). To modify the definition of a group, simply select or de-select channels while the group is selected.

The group number also appears in the top-left corner of the screen. Touch the group number to advance to the next group. Repeatedly pressing the "Select Channels" key also advances to the next group.

"Show Data" screen

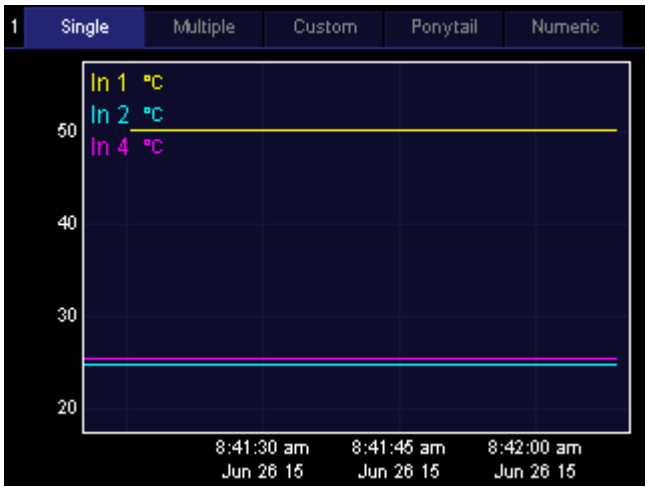
This screen displays data from the selected channels as graphs or as numbers. The tabs at the top of the screen control how data is displayed. Press the "Show Data" key repeatedly to change the selection group. Each of the four groups remembers its display format (single, multiple, etc.) as well as the plot's X and Y range. Therefore, when you change the selection group, the graph's range may also change.

Note that the graphs always show data recalled from the log. If data is deleted from the log, it no longer appears on the graph. In addition, if the log interval is sufficiently long, the graphs may have a "stairstep" appearance.

Single

`System.display.type single`

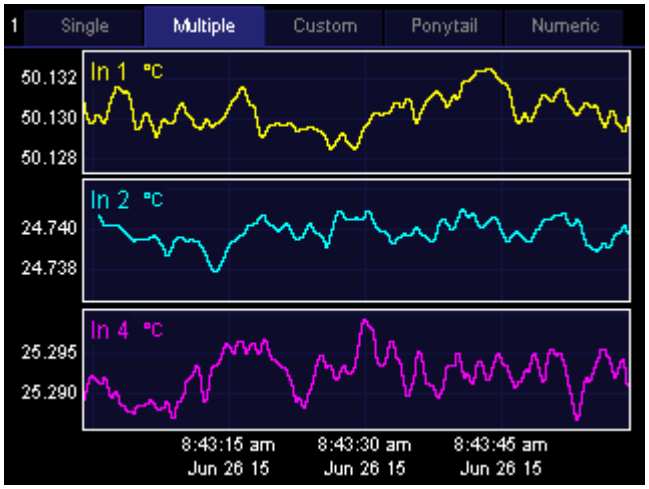
Touch the Single tab to see the selected channels plotted on a single graph with a common Y axis. Up to eight selected channels can be shown. If more than eight channels are selected, only the first eight are shown.



Multiple

`System.display.type multiple`

Touch the Multiple tab to see each selected channel plotted on its own graph. Each graph has its own Y axis scale. If more than eight channels are selected, only the first eight are shown.

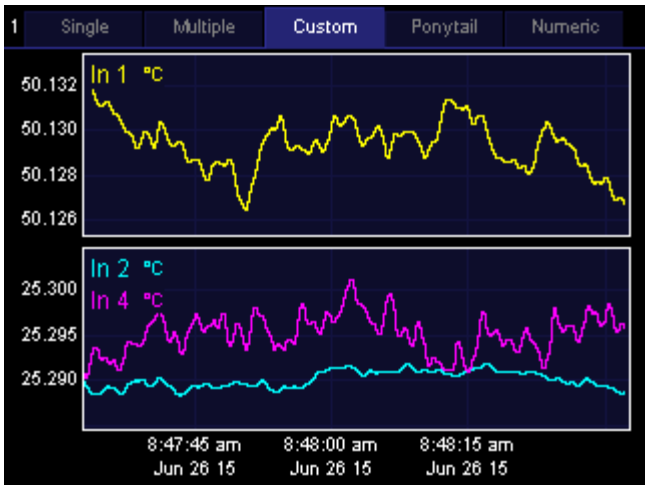


Custom

`System.display.type custom`

Touch the Custom tab to view a plot in which each channel can be assigned to one of up to eight graphs. To set the plot a particular channel appears in, display the Setup menu for that channel and touch the "Plot" button.

In the example below, channel "In 1" has been assigned to plot 1, while channels In 2 and 4 have both been assigned to plot 2.

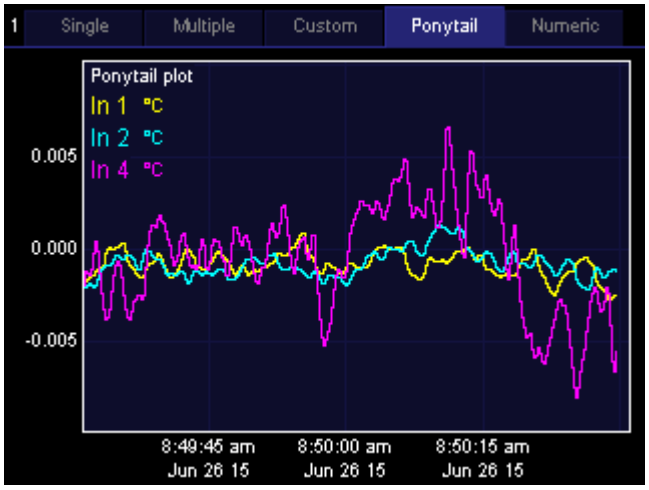


Ponytail

`System.display.type ponytail`

The Ponytail plot resembles the Single plot, except each trace is offset by its initial value so that the trace begins at zero. The offset is recalculated whenever you touch the graph to zoom or pan, or whenever you switch to another screen and back to the Plot screen. If you don't touch the CTC100's front panel, the offset is never recalculated.

Viewing the Ponytail plot does not cause offsets to be subtracted from logged data or feedback setpoints.



Numeric

```
System.display.type numeric
```

Touch the Numeric tab to see the current values of the selected channels as numbers. Up to 22 channels can be displayed. The more channels that are selected, the smaller the numbers are. If enough space is available, the type of sensor or output may be displayed, and an annunciator may appear that indicates whether the sensor or heater is disconnected ("N/A"), over range ("Hi"), under range ("Lo"), if Output Enable is off ("Off"), or if an internal error has occurred ("Err").

Channel displays turn red whenever the channel's alarm is triggered. The name of the channel appears in bold if the channel uses a custom calibration table.

Touch one of the channels to go to its setup menu.

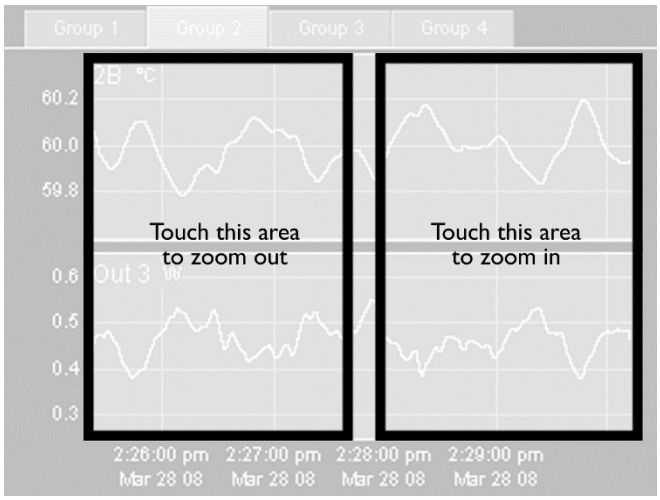
Zooming and panning

To change the X axis scale of a plot, touch anywhere inside the plot:

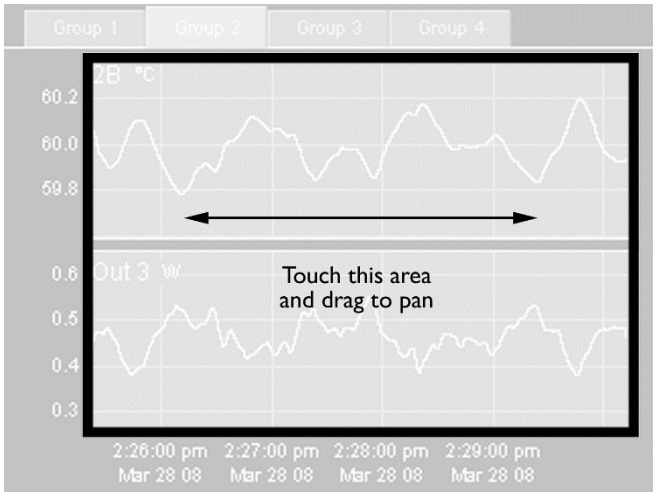
- Touch the right half of the plot to zoom in
- Touch the left half to zoom out
- Drag to pan.

Whenever the most recent data is visible on the graph, the graph automatically scrolls to keep the most recent data visible. If the most recent data is not visible, the words "X lock" appear in the bottom-left corner of the screen to indicate that scrolling is disabled. To show current data and resume scrolling, touch the words "X lock".

Graphs that appear together on a screen always have the same X axis range. However, each selection group has its own, independent X axis range.



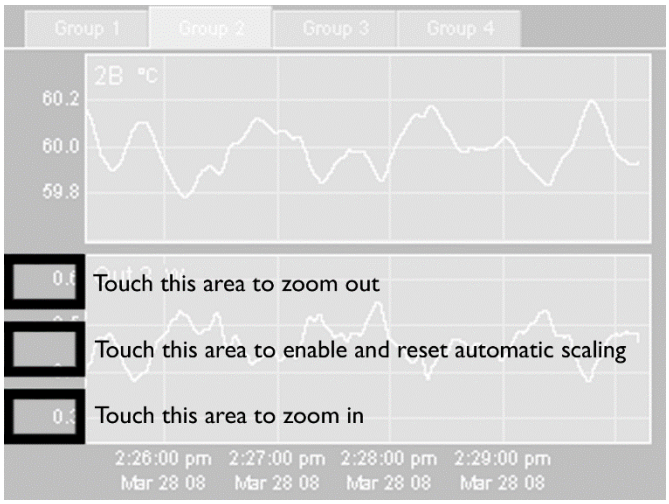
How to change the X axis scale



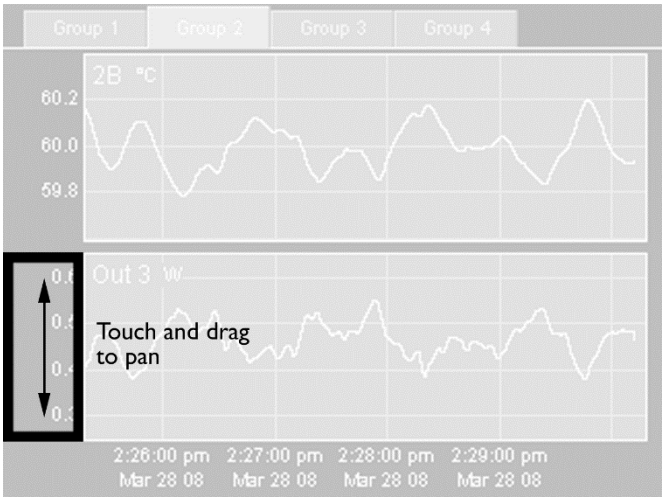
How to pan the graph horizontally

By default, the CTC100 continually adjusts the Y-axis scale to accommodate all the data on the graph. Each graph has its own, independent Y axis scale. To change the Y axis scale for a particular graph, touch the area to the left of its Y axis.

- Touch the top third of the Y axis to zoom out. Automatic scaling is disabled, so the Y axis scale no longer changes as new data is acquired.
- Touch the middle third to 1) re-enable automatic scaling and 2) reset automatic scaling, that is, ignore previously-acquired data and adjust the Y range to accommodate only new data.
- Touch the bottom third to zoom in. Automatic scaling is disabled.
- Drag to pan.



How to change the Y scale of the bottom graph



How to pan the bottom graph vertically

"Program" screen

A program is a set of one or more CTC100 commands in ASCII text format. Programs can be sent over the RS-232, GPIB, USB, or Ethernet interface, input from the program screen, or transferred as text files on a USB memory device. Regardless of how a program was input, its progress can be monitored from the program screen. Up to 10 programs from any one interface and up to 20 programs total can run at once.

The Program screen has an Input window, which shows the program as it was received; a Messages window, which shows responses and error messages from the CTC100; and a Progress window, which shows the individual instructions in the program, one instruction per line.

If a program is not running, you can compose or modify it by touching a line in the Progress window. Touching a blank line brings up a list of possible commands. Touching a line that already contains an instruction brings up a list of three options: you can add a new instruction on the line above the one that was touched; delete the instruction that was touched; or replace the instruction that was touched.

The Program screen has six buttons:

Play symbol

If a program is displayed but not running, press this button to start the program. If a program is running in the currently-selected tab, the button is highlighted and pressing it stops the program.

Pause symbol

Press this button to temporarily pause the program running in the currently-selected tab. Press the button again to resume the program.

Clear

Erases all text from the Input, Messages, and Progress windows. Unless it has previously been saved, the current program is lost. This button cannot be pressed while a program is running in the current tab.

Load

Touch this button and a list of programs stored in memory is displayed. Programs can be stored in memory with the Program.Save button, by sending a "define" instruction to a remote interface, or by attaching a USB device with text files contained in a "Macros" folder. Select a program from the list and its component instructions are displayed in the Progress window, replacing whatever was previously in the window.

The Program.Load button can be used to edit a previously-saved program: load the program, then edit it in the Progress window, and finally re-save the program with the Save button.

To call a previously-saved program as a subroutine from a program that you're composing, don't use the Program.Load button, since it would erase the rest of the program. Instead, touch the "Progress" window and select the saved program from the list of commands. The Program.Load button cannot be pressed while a program is running in the current tab.

Save

`define` *`macroName macroContent`*

Saves the current program, as shown in the Input window, to memory. You'll be asked to supply a name for the program. Up to 15 programs can be saved. If 15 programs are already saved, the Save button will have no effect.

Saved programs can be run using the "Load" button or called as subroutines by touching a line in the "Progress" window and then touching the name of the program. Saved programs can also be called by sending their name (like any other instruction) over one of the remote interfaces.

Delete

`delete` *`macroName`*

Touch this button to display a list of programs stored in memory. Select a program from the list and it will be deleted from memory. If the program is running, deleting it does not affect the running program.

Sending programs over RS-232, USB, GPIB, or Ethernet

Programs can be entered from any of the CTC100's communications ports: RS-232, USB, Ethernet (via Telnet or a raw UDP data stream), or the optional GPIB port. Each line of text sent to the CTC100 is run as a separate program (the entire program must be on a single line). If two or more lines are sent to the CTC100 in quick succession, the programs may run concurrently; the CTC100 may not finish running the first program before beginning the second. However, the first program sent will always begin running before the second program. If it's necessary to run programs sequentially, begin each line with the *PHO (port holdoff) instruction.

See the "remote interface" section of this manual for more details.

Preparing programs as files on USB memory devices

The CTC100 can also read programs that are stored as text files a USB memory device. This is the easiest way to import longer programs.

Create a "Programs" folder in the root directory of the memory device. Type the program in a word processing or text editor program, and save it as a .txt file in the "Programs" folder. Plug the memory device into the CTC100. To verify that the program is available, look for its name in the Macros column of the System Setup screen. The program can be run just as if it were saved in the CTC100's memory; however, after the USB device is unplugged, the program can no longer be started.

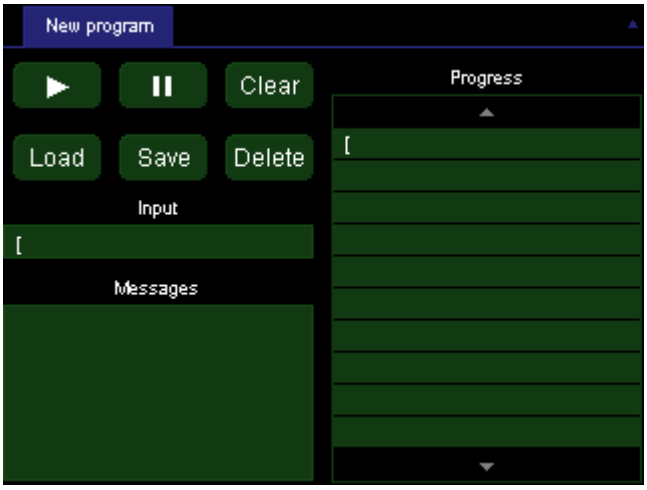
Program files can contain up to 16384 characters. Unlike programs sent over communications ports, program files can have more than one line; all extra whitespace is ignored. The program can also contain comments; an apostrophe, i.e. a single quote mark, indicates that the rest of the line is a comment.

Preparing programs from the front panel

Simple programs, for example, a series of temperature ramps, can be entered from the front panel.

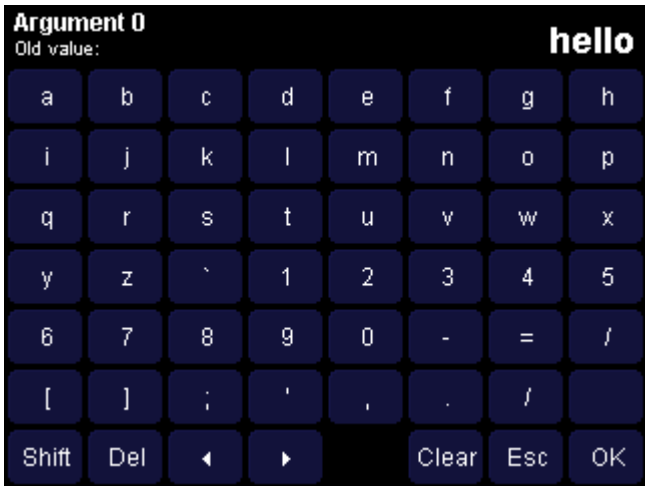
To enter a program from the front panel, first press the "program" key to show the Program screen. Touch the Progress window and a list of available top-level commands appears. A dot at the end of a command means that touching that button will bring up a sub-menu of instructions. For example, the command to change the feedback setpoint (channel.PID.setpoint) is accessed by first touching the "channel." button. See the "Programming" section of this manual for a full list of commands.

Touch the left square bracket (the button in the upper-left corner). Square brackets surround blocks of code to be repeated. The menu of instructions closes, and the first line in the "Progress" window is now a left square bracket.

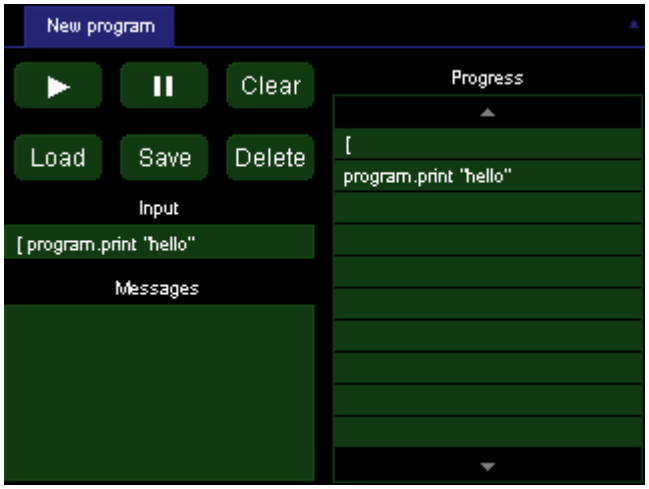


Touch the Progress window again, anywhere beneath the first line. The list of possible instructions appears again. Select "program." from the list. The sub-menu that appears contains a list of instructions that affect the program. For example, "cls" clears the Messages window; "name" assigns a name to the program; and "kill" ends a named program.

Select “print”. An alphanumeric input screen appears, on which you can enter an argument for the “program.print” instruction. Type “hello”.



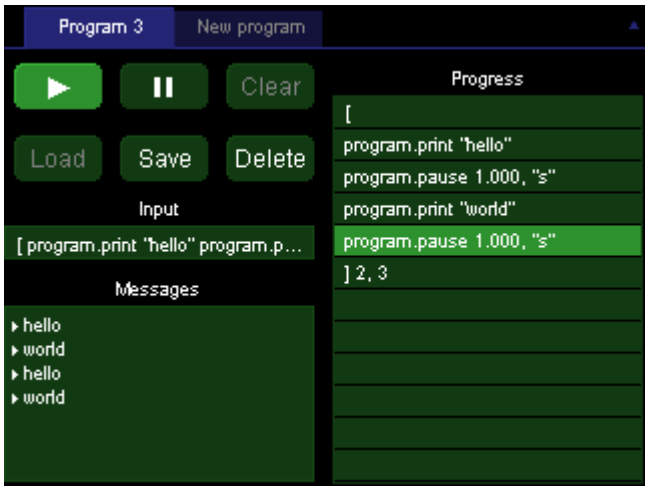
Touch the OK button. The Program screen re-appears and the instruction “program.print “hello”” appears in the second line of the Progress window.



Next, enter the instruction: “program.pause 1 s”. The pause instruction has two arguments that must be entered separately. First a numeric input screen appears where you can type “1”. Touch “OK” and a second menu appears where you can select the units (“s”). The completed instruction will pause the program for one second.

Next, enter the instruction “program.print world” followed by “program.pause 1 s”. Finally, enter the instruction “] 3”. This makes the program repeat everything between the square brackets three times.

Press the start button. While the program is running, the current instruction is highlighted and the number of repetitions remaining appears next to the right square bracket. In addition, while the program is running, a tab labeled “New program” appears at the top of the screen. By touching this tab, you can enter and start a second program while the first program is still running.



When the program is done, the messages “hello” and “world” should appear three times in the Messages window.

Once the program has finished it’s possible to press the start button to run the program again, the “Save” button to save the program, or the “Clear” button to erase the program and the Messages window.

Running concurrent macros

A macro can run for a long period of time or even indefinitely. Therefore, it’s possible to start a new macro before the previous macro has finished. It’s also possible to run multiple instances of a saved macro simultaneously.

The CTC can run up to ten concurrent macros started from the front panel. If an eleventh macro is started, a “Too many macros” assembly error is generated and the macro does not run.

When the CTC100 is turned on, it looks for a macro named “startup” and, if it exists, runs the macro. Any other macros that might have been running when the CTC100 was switched off are not re-started.

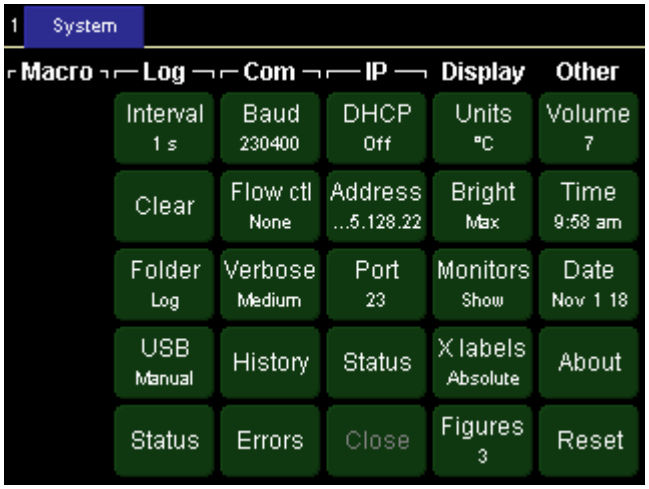
“Setup” screen

Press the “Setup” key to configure the CTC100. The Setup screen has between 1 and 5 blue tabs at the top, depending on how many channels are selected. Touch the “System” tab to configure parameters that affect the entire instrument, like the RS-232 baud rate, the display brightness, and

the time and date. The Setup screen also has one tab for each selected channel. Touch one of these tabs to set up a particular channel.

Repeatedly pressing the "Setup" button, or touching the group indicator in the top-left corner, cycles through the four selection groups.

"Setup" screen: System tab



The System setup screen includes controls for all settings that affect the entire instrument, including time and date, Ethernet and GPIB, and data logging parameters.

"Setup" screen: System tab: Macro column

MacroName

If any macros have been defined, buttons with their names appear in the left-hand column of the System setup screen. If more than five macros have been defined, buttons for only the first five appear. Touching one of these macro buttons runs the corresponding macro, and the button remains selected (i.e., highlighted) as long as the macro continues to run.

More generally, a macro button appears to be selected whenever a macro with the name shown on the button is running, whether the macro was started by touching the button, with a remote command, or from the Program screen. Touching a selected macro button stops all currently-running macros with that name, regardless of how the macros were started. See the Macro Names topic in the Remote Programming section for more information on how macro names are assigned.

"Setup" screen: System tab: Log column

Interval

`System.Log.Interval { off, 0.1 s, 0.3 s, 1 s, 3 s, 10 s, 30 s, 1 min, 3 min, 10 min, 30 min, 1 hr }`

Sets the default time between log points. If the interval is set, for example, to 1 s, the CTC saves a data point once per second, and each point represents the average reading over one second period.

Note that each channel has its own log interval setting (`Channel.Logging`) that can override the default interval.

Clear

`System.Log.clear { RAM, USB, all }`

Erases logged data. There are three options:

RAM deletes all logged data from the CTC100's internal memory and can be used to clear old data from the plot screen.

USB deletes all log files (not just the current file) from the log folder on the external USB memory stick. Specifically, all files named "\<folder>\XX\LogYY.csv" are deleted, where <folder> is the folder name specified by the Folder button, and XX and YY are two-digit numbers.

All is the same as selecting RAM and USB.

Folder

`System.Log.folder "`*`FolderName`*`"`

Sets the USB device folder into which data is logged. If the folder does not exist, it is created. If the folder does exist and already contains log files, data is appended to the existing files. Only data from the current folder appears on the plot screen. The default folder name is "./", which is the root directory of the USB device.

USB

`System.Log.USB { auto, manual }`

This setting determines whether or not the CTC automatically starts logging to USB memory devices when they are plugged in.

Auto: when a USB storage device is plugged into the instrument, the CTC immediately starts logging data to it.

Manual: when a USB storage device is plugged into the instrument, you must touch the grey dot in the upper-right corner of the screen to begin logging. If you unplug the device and plug it back in, data will no longer be logged to the device.

Status

`System.Log.Status`

Press this button to see which file data is being logged to and how full the file is. The information is updated every 5 seconds. When the file becomes 100% full, a new log file will be opened.

"Setup" screen: System tab: COM column

Baud

`System.COM.Baud { 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400 }`

Sets the baud rate for the RS-232 and USB-to-PC interfaces. The interface always has 8 bits, 1 stop bit, and no parity.

Flow ctl

`System.COM.Flow ctl { None, RTS/CTS }`

Enables or disables hardware flow control for the RS-232 and USB-to-PC interfaces.

GPIB

`System.COM.GPIB { 1, 2, 3,…30 }`

Sets the primary GPIB address. The address must be a value between 0 and 30, inclusive, but in most GPIB systems 0 is reserved for the controller-in-charge and should not be used.

Verbose

`System.COM.Verbose { Low, Medium, High }`

Determines how the CTC100 responds to RS-232, USB, GPIB, and Ethernet messages.

Low: the CTC100 only sends messages in response to queries. This mode should be selected for IEEE488.2 compatibility.

Medium: the CTC100 also sends error messages whenever a command could not be understood. Error messages always begin with the word "Error".

High: the CTC100 also sends messages whenever a parameter is set. Messages include the name of the parameter that was set or queried

Example responses are shown in the table below.

Verbose level	Response to instruction…		
	2A?	**xyz** (invalid instruction)	**2A = 37.47**
Low	37.4722	(no response)	(no response)
Medium	37.4722	Error: "xyz" is not a valid instruction	(no response)
High	2A.Value = 37.4722	Error: "xyz" is not a valid instruction	2A.Value = 37.47

History

`System.COM.History`

Pressing this button brings up a window that that shows the contents of the last twelve messages sent or received over the COM ports. The window is helpful for debugging communications issues.

Errors

`System.COM.Errors`

Pressing this button produces a window that shows the last six errors caused by COM port communications.

"Setup" screen: System tab: IP column

DHCP

`System.IP.DHCP { On, Off }`

Enables or disables the Dynamic Host Configuration Protocol. If DHCP is set to "on" and a DHCP server is present on the network, the other IP settings are automatically configured and are greyed out.

Address

`System.IP.Address` *`address`*

Sets the IP address in dot-decimal notation.

Port

`System.IP.Port` *`portNumber`*

Sets the telnet and UDP port for Ethernet communications. Remote commands can be sent to the CTC through a telnet connection or a raw UDP data stream addressed to the selected port. The port must be a value between 0 and 65535, inclusive, and should normally be either 23 (the default) or a value greater than 1024.

Status

Shows the status of the Ethernet interface.

- Link is up/down at 10/100/1000 Mbps: if the link is up, that means the CTC100 is physically connected to a network. If it's down, a network cable is unplugged or there's some sort of hardware problem.
- Max frames received: the CTC100 has a queue of unprocessed Ethernet packets, and this value is the longest that the queue has gotten in last 5 seconds. If this number reaches 40, that means Ethernet data is being received faster than the CTC100 can process it.
- Remote IP: shows the IP address of the remote system that the CTC100 is accepting commands from. If this is not the IP address of the PC that's sending commands, press the "close" button and then have the PC send a command.
- Protocol: TCP or UDP. Indicates which protocol the CTC100 is using to communicate with the PC. If this isn't the protocol that you want to use, touch the Close button and then have the PC send a command using a different protocol.
- TCP state: The status of the TCP connection. Normally the state is either "listen", which means the CTC100 is waiting for a connection, or "established", which means that there's an active connection. If it's stuck in some other state for more than a few seconds, that might mean there's some sort of network incompatibility.
- DHCP: The state of the process that automatically assigns an IP address to the CTC100. The state is normally "off" if DHCP is turned off or "bound" if it's turned on. If it's stuck in another state for more than a few seconds, that might mean that the CTC100 can't communicate with the DHCP server.

Close

`System.IP.Close`

The CTC100 only accepts messages from a single computer and only in a single protocol (Telnet or raw UDP stream). To change the computer or the protocol, press the Close button.

The Close button is greyed out if there's no open Telnet connection and the CTC100 hasn't received any UDP packets addressed to the active port (see the "Port" button above).

You don't need to press the Close button if a Telnet connection is being used and the computer closes it.

"Setup" screen: System tab: Display column

Units

`System.Display.Units { °C, K, mK, °F, Sensor }`

Sets the temperature units for the entire instrument. Temperature measurements are both displayed and logged in the specified units. If the units are changed in the middle of an experiment, there will appear to be a large jump in all of the temperature records.

Five options are available: °C, K, mK, °F, and Sensor. If the Sensor option is selected, sensor measurements are not converted to temperature and appear in the native units of the sensor, i.e. millivolts for thermocouples, volts for diode sensors, and ohms for resistive sensors. When Sensor units are selected, the Units button is blank and `System.display.units?` returns an empty string.

When the system temperature units are changed, other variables that are expressed in temperature units need to be manually updated by the user. These variables are not automatically updated because in some cases it is inappropriate to update them (for example, when inputs are set up to reflect temperature differences rather than absolute temperatures). The following settings may need to be updated:

Channel`.PID.Setpoint`
Channel`.PID.Ramp`
Channel`.PID.P`
Channel`.PID.I`
Channel`.PID.D`

Channel.Alarm.Min
Channel.Alarm.Max
Channel.Cal.Offset
All Zone Min, P, I, and D settings

Bright

System.Display.Bright { Off, 2, 3, 4, 5, 6, Max }

Sets the backlight brightness. If set to Off, the display turns completely off but turns on again for 2 seconds each time the screen is touched.

Monitors

System.Display.Monitors { Show, Hide, Off }

Most I/O cards have "monitor" channels that can show additional information about the card or the status of its I/O channels. For example, most cards have an onboard sensor that measures the temperature of the card itself. Cards that are intended to drive resistive heaters have a channel that shows the resistance of the heater, and may also have channels that show the heater voltage and current. While not typically needed for normal day-to-day use, these monitor channels can be very useful when debugging performance issues.

To save memory and processor time, the monitor channels are disabled by default. They can be enabled by touching the "Monitors" button and then selecting "Show" from the pop-up menu. The monitor channels will then appear on the Select screen like any other channel, will be logged to internal memory and to a USB stick (if USB logging is enabled), and can be queried by remote commands such as getOutput, getLog, and *Channel*?.

If "Monitors" is set to "Hide", the monitor channels don't appear on the Select screen, but are still logged and can be queried by remote commands.

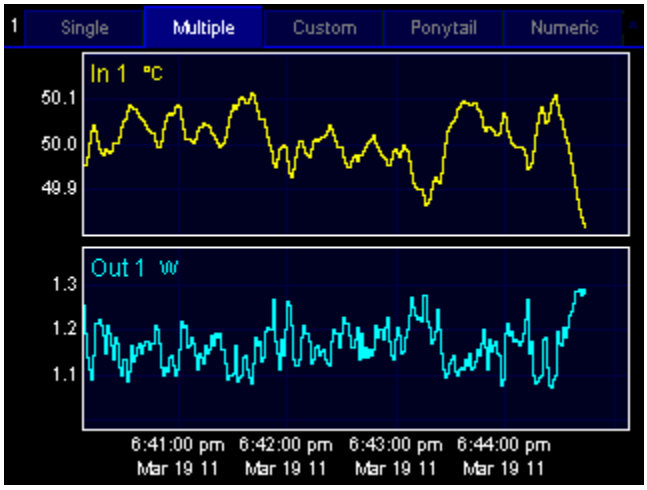
If "Monitors" is set to "Off", the monitor channels are removed from the response to getOutput and can't be queried by remote commands. The channels are not logged, although empty columns for the channels still appear in the log files.

X labels

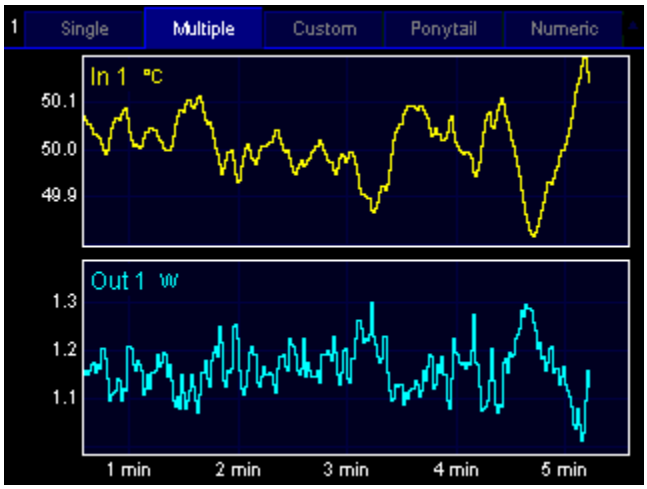
System.Display.XLabels { Absolute, Elapsed }

Absolute: the labels at the bottom of each graph show the full time and date.

Elapsed: the labels only indicate the amount of time between grid lines. The elapsed time labels do not actually reflect the amount of time elapsed since any particular event and reset to zero once per minute, hour, or day, depending on the X range of the graph.



Absolute X labels



Elapsed X labels

Figures

`System.Display.Figures { 0, 1, 2, 3, 4, 5, 6 }`

Sets the number of figures that are shown after the decimal point on the front panel display, and in values sent in response to remote queries. The front panel display may show fewer digits if the requested number of digits doesn't fit into the available space. This setting does not affect logged data, PID feedback, or plots.

Each channel also has its own Figures setting that can override this global value.

"Setup" screen: System tab: Other column

Volume

`System.Other.Volume { Off, 1, 2, 3, 4, 5, 6, 7, Max }`

Sets the speaker volume. The volume affects all sounds, including alarms.

Time

`System.Other.Time "time"`

Sets the time of day. Does not affect the time stamps of previously-acquired data points. For example, if the time is advanced by one hour, a one-hour gap appears in the plot. Conversely, if the time is set back by one hour, any data taken over the last hour is no longer plotted, and newly-acquired data appears in its place. The data is not erased from the USB log; it just doesn't appear on the plot.

Date

`System.Other.Date "date"`

Sets the date. Does not affect time stamps on previously-acquired data points.

About

`System.Other.About`

Displays a text box with information about the firmware version and installed I/O cards.

Reset

`System.Other.Reset { "Running macros", "Saved macros", Display, Ports, "Port settings", Channels, Log, All }`

Running macros: stops all running macros. Has no effect on saved macros.

Saved macros: deletes all saved macros from local memory. Does not delete macros from USB memory devices. Has no effect on running macros.

Display: Resets all settings in the System screen's Display column to their factory defaults. Returns the front panel to the Select menu, de-selects all channels in all groups, and erases locally-stored log data (data on USB drives is not affected). Returns all plots to autoscaled X and Y with a 1 minute X range and changes the plot location of all channels to 1. If a `*TRG` remote command was previously received, re-enables automatic A/D conversions. Hides the monitor channels.

Ports: Closes all I/O ports and re-opens them. USB and Telnet connections are lost. The port settings (baud rate, IP address, etc.) remain unchanged.

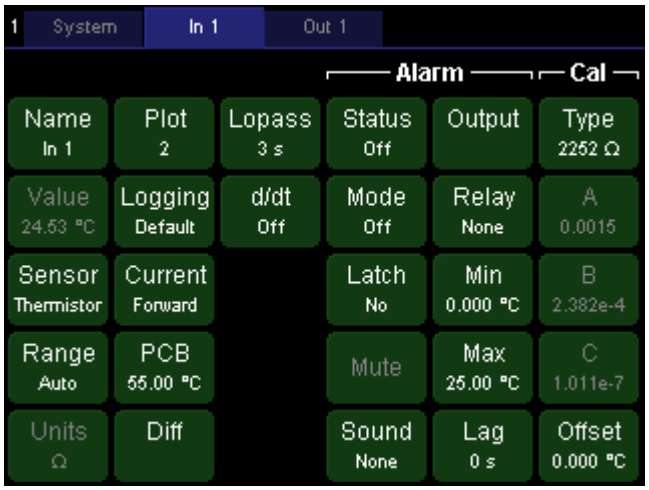
Port settings: Resets all I/O port settings to their factory defaults.

Channels: Resets the settings on the Channel menu for all channels to their factory defaults. Also sets the A/D rate to 100 ms.

Log: Resets the default log rate to 1 second, sets the log rate for each channel to the default, and enables automatic logging to USB. If a USB storage device is attached, erases log files in the root directory and begins logging to USB.

All: resets all of the above items.

Setup screen for channels In 1 – In 4



Except for the "PCB" setting, all the settings on this screen apply only to the channel in the highlighted tab and will not affect any other channels. When using the remote commands the word *Channel*, where it appears in italics, should be replaced with the channel name with spaces omitted. For example, the remote command to change the name of channel In 1 to "Sample A" would be:

```
In1.name = "Sample A"
```

Name

Channel`.Name = "`*New channel name*`"`

Touch this button to change the name of the channel. The name must have 10 or fewer characters.

Value

Channel`.Value?`
Channel`?`

This button shows the most recent sensor reading. The reading can't be changed by typing in a new value, so the button is greyed out.

Sensor

Channel.Sensor { Diode, ROX, RTD, Therm }

Touch this button to select a sensor class. Four options are available: diode, ROX (ruthenium oxide), RTD, and thermistor.

This setting controls how the sensor input hardware reads the input signal. It determines the excitation current, whether the CTC100 reads the sensor voltage or resistance, and the list of calibration options available in the Cal column. The Diode option sets the sensor excitation current to 10 μA and causes the hardware to read the voltage across the sensor. The ROX, Thermistor, or RTD option causes the hardware to read the sensor resistance. The ROX and thermistor modes are identical, except for the list of calibration options. The RTD mode results in a larger default sensor excitation current (and therefore lower noise) and places an RTD-specific list of calibration options in the Cal column.

Range

Channel.Range { 10ê, 30ê, 100ê, 300ê, 1kê, 3kê, 10kê, 30kê, 100kê, 300kê, 2.5V, Auto }

Sets the sensor measurement range. The default range is Auto. In general, a lower range produces a larger excitation current, less noise, and more accurate measurements. The range should be manually set if it is critical to limit sensor self-heating; otherwise the CTC100 may change the range and excitation current at unexpected times.

The Auto range function increases the range by one step whenever the sensor resistance rises above 95% of the current range, and decreases the range by two steps whenever the sensor resistance falls below 10% of the current range.

The CTC100 uses ASCII character 234 for the Ohms symbol. To type this character on a Windows computer, hold down the alt key and type 0234 on the number pad. On Windows computers the character appears as a letter "e" with a circumflex accent.

Units

Channel.Units?

This button shows the native units in which the sensor is read. It reads "volts" if the sensor type is "diode", or "ohms" if any other sensor type is selected. The units cannot be changed directly, but they change automatically if the sensor type is changed.

Plot

Channel.Plot { 1, 2, 3, 4, 5, 6, 7, 8 }

Indicates which plot this channel will appear in when the "Plot" screen is showing, the plot type is "Custom" (see the "Plot Screen" section above), and this channel is selected on the "Select" screen. Choose one of eight plots for the channel to appear in, where plot 1 is the uppermost plot. Empty plots won't appear on the "Plot" screen; for example, if all the selected channels have been assigned to plot 4, only one plot appears on the Custom plot screen.

Logging

Channel.Logging { Off, "0.1 s", "0.3 s", "1 s", "3 s", "10 s", "30 s", "1 min", "3 min", "10 min", "30 min", "1 hr", Default }

By default, readings from all channels are saved to the log at the global log rate, which is set on the System Setup screen (System.Log.Interval). However, exceptions can be made for individual channels by setting a different rate with the Logging button. To return the channel to the global log rate, set Logging to "Default".

Figures

Channel.Figures { 0, 1, 2, 3, 4, 5, 6, Default }

Sets the maximum number of digits that are shown after the decimal point for all channel values. Fewer digits may be shown if the value won't otherwise fit into the available space. This setting does not affect logged data, PID feedback, or plots.

If Default is selected, the number of digits is determined by the `System.Display.Figures` setting.

Current

Channel`.Current { Forward, Reverse, AC, Off }`

Forward: the polarity of the Sense and Excitation pins are as shown in the connector diagram.

Reverse: the polarities of the pins are switched and the sense current flows in the opposite direction through the sensor.Can be used to compensate for diodes that have been connected backwards.

AC: the current switches polarity with every reading. The measured resistance is the average of the last two readings. Recommended for resistive temperature sensors such as RTDs and thermistors, since it cancels out thermal EMFs, significantly improves accuracy, and reduces noise.

Off: disables the excitation current. The sensor cannot be read in this state and no sensor reading will appear. Prevents the CTC100 from "hunting" for the correct range when no sensor is connected, a process that generates audible clicks.

If two diodes are connected in parallel but with opposing polarities to a single pair of leads, one diode can be read with the forward current setting, the other with the reverse current setting, and their average with the AC setting. This technique reduces the number of leads and therefore the amount of heat transfer.

Power

Channel`.Power { Auto, Low, High }`

Determines the amount of excitation current used to measure resistive sensors. See the "excitation current" topic on page 5.

Low: minimizes sensor heating. This option is mainly for use with thermistors in cryogenic applications. To compensate for the fact that heat conductivity decreases (and thermistor resistance increases) as the temperature approaches absolute zero, the amount of power that the sensor dissipates decreases as the measurement range is increased.

High: minimizes noise. Power dissipation is kept roughly constant as the measurement range is increased. This option is for use with RTDs or with any kind of sensor at non-cryogenic temperatures.

Auto: uses low power if the sensor type is set to thermistor or ROX, or high power if the sensor type is set to RTD.

PCB

Channel`.PCB 0.0`

Sets the maximum printed circuit board (PCB) temperature. Since channels In 1 and In 3 are located on a single printed circuit board, their PCB setting is always the same. Likewise, channels In 2 and In 4 also share the same PCB value.

If the card's temperature exceeds the maximum and System.Other.Fan is set to "Auto", the CTC100 increases the fan speed to reduce the card's temperature.

The PCB temperature is always in °C, regardless of the System.Display.Units setting. The default setting is 35°C.

Reducing the maximum PCB temperature results in tighter regulation of the selected card's temperature, at the expense of the other cards and more fan noise.

The fan speed is also determined by the cooling needs of the DC outputs.

Diff

PositiveChannel`.Diff "`*Negative channel*`"`

This button lets you display the difference between two channels. Touching this button displays a list of available channels. Touch a channel and its value is then continuously subtracted from the channel indicated by the blue tab at the top of the Setup screen. If, for example, the Setup menu for

channel In 1 is showing, and you touch the "Diff" button and select channel "In 2" from the list, channel In 1's value becomes In 1 – In 2. The raw value of channel In 1 can no longer be seen.

To turn the difference feature off, touch "Diff", then touch whatever channel is currently selected. The "Diff" button then shows an empty value.

Difference readings can be used as the input for PID feedback loops, in which case the feedback maintains a constant temperature differential between two locations, rather than a constant absolute temperature.

Lopass

Channel.Lopass { Off, "1 s", "3 s", "10 s", "30 s", "100 s", "300 s" }

The lowpass option smooths the input signal with a 6th-order RC filter, reducing noise but also slowing down the sensor response. If, for example, the 10 s lowpass filter is selected, noise spikes less than 10 seconds long are removed, but it would also take the signal at least 10 seconds to respond to any sudden temperature changes.

The filter's time constant should ideally be just below the response time of your hardware. That is, the 10 second lowpass filter should only be used if you wouldn't expect to see temperature spikes shorter than 10 seconds anyway. In that case the lowpass filter only eliminates noise and doesn't slow down the system.

The lowpass filter should usually be enabled on the temperature inputs of PID control loops. This is especially true when using step response PID tuning or when derivative feedback is enabled (i.e., when the derivative gain is nonzero), since these algorithms calculate the change in temperature over time and therefore produce poor results if high-frequency noise is present.

When a sensor is disconnected and then reconnected to a lowpass-filtered channel, the CTC allows one second for the reading to settle. During this time, no reading appears. The output of the lowpass filter is then set equal to the next ADC reading so that you don't have to wait for the reading to gradually settle to its new value.

d/dt

Channel.d/dt { Off, On }

When d/dt is set to "On" the value of the selected channel is replaced with its rate of change, i.e. the difference between the previous ADC reading and the current reading divided by the time between the two readings. Since the derivative is normally somewhat noisy, the lowpass filter should be enabled.

Setup screen for channels In 1 – In 4: Alarm column

Each input channel has an alarm. If enabled, the alarm is triggered if any of the following conditions occur:

- The input (or its rate of change) exceeds the user-specified minimum and maximum values
- The input exceeds the measurement range of the I/O card
- The sensor is disconnected (except on analog I/O channels, which cannot detect disconnected sensors)

When an alarm is triggered, it can do any of the following:

- Play a sound
- Trigger a relay on the digital I/O card
- Shut off an output channel

The alarm can be programmed to remain triggered until it is manually shut off (latching alarm), or to shut itself off as soon as the input returns to a value within the alarm limits (non-latching alarm). The alarm can also be programmed to ignore momentary glitches.

To determine which alarms are currently triggered, look at the Select screen. Red channels indicate that the channel's alarm is in the triggered state.

It's very important to set at least one alarm if your heater can output enough power to damage your system. The alarm should be configured to disable the heater output when triggered. For additional protection, the heater output can be routed through one of the CTC100's relays and the relay associated with the alarm. Without such a safety mechanism, it's possible for the CTC100 to enter a "runaway feedback" condition if a sensor becomes unplugged or malfunctions, or if the PID feedback is incorrectly set up.

Status

Channel.Alarm.Status { Off, On }

Indicates if an alarm condition is currently present on this channel. If a latching alarm has been triggered, touch the Status control and set its status to "Off" to turn the alarm off. This control can also be used to artificially turn the alarm on to test the sound, output channel disabling, and GPIB status reporting.

To test an alarm, enable the alarm with the Mode control and then set its Status to "On". The alarm immediately turns on. If the alarm is non-latching, it turns off in less than a second; if it is latching, it stays on until the Status is set to "Off". The Lag setting has no effect on this test.

Mode

Channel.Alarm.Mode { Off, Level, "Rate /s" }

Off: the alarm never sounds.

Level: the alarm sounds whenever the channel's value exceeds the alarm Min and Max. The alarm also sounds whenever the input is disconnected or the sensor value exceeds the range of the input.

Rate /s: the alarm sounds whenever the channel's rate of change (in degrees per second) exceeds the alarm Min or Max. The alarm also sounds whenever the input is disconnected or the sensor value exceeds the range of the input.

Latch

Channel.Alarm.Latch { Yes, No }

If set to Yes, the alarm, once triggered, stays on until it is turned off with the Status or Mode control. If set to No, the alarm turns itself off once the input is again within the alarm limits.

Mute

Channel.Alarm.Mute { On, Off }

Temporarily silences the alarm sound without affecting the associated relays or output channels. Once this button is touched, the alarm stays muted until the alarm condition goes away or until the button is touched again.

Sound

Channel.Alarm.Sound { None, "1 beep", "2 beeps", "3 beeps", "4 beeps" }

Controls which sound plays when the alarm goes off.

Output

Channel.Alarm.Output "*Output name*"

The alarm, when triggered, can shut off one of the CTC100's output channels, setting the output to zero and temporarily disabling that channel's feedback loop. Once the alarm status returns to "Off", the output returns to its previous value and the feedback is re-enabled. This feature can be used to guard against runaway feedback loops or to otherwise protect equipment from damage due

to excessive temperatures. For example, one or more backup temperature sensors can be programmed to shut off a PID output to prevent damage in case the primary sensor fails.

Touching the "output" button brings up a list of output channels; from this list, select the channel to be shut off. If a channel is already selected, touching it again de-selects the channel and no channel will be shut off when the alarm triggers.

Relay

Channel.Alarm.Relay { None, A, B, C, D }

If a digital I/O card is installed in slot 6, the alarm can switch one of its four relays on. It's possible to assign more than one alarm to a given relay, in which case the relay will turn on if any one of the alarms is triggered.

Min

Channel.Alarm.Min *0.0*

The lowest permissible value of the input. The alarm is triggered if the input (or the rate of change of the input) becomes lower than this value.

Max

Channel.Alarm.Max *0.0*

The highest permissible value of the input. The alarm is triggered if the input (or the rate of change of the input) exceeds this value.

Lag

Channel.Alarm.Lag *0.0*

Prevents noise or glitches from inadvertently triggering the alarm. The alarm will not be triggered until the input has continuously exceeded the min or max setting for this number of seconds. The lag applies when the alarm is being switched and when it is being switched off.

Setup screen for channels In 1 – In 4: Cal column

The CTC100 offers four different ways to calibrate sensor readings:

- **Built-in calibration tables**: the easiest but least accurate way to calibrate your sensors. You select a sensor type and the CTC100 uses built-in calibration data that describes a typical sensor of that type. No experimental data is needed.
- **Calibration coefficients**: potentially more accurate. You enter 3–4 coefficients for an equation that is specific to your sensor type (the Callendar–van Dusen equation for RTDs, the Steinhart–Hart equation for thermistors, or a polynomial fit for diodes). The coefficients are typically provided by the sensor manufacturer or can be derived from several measurements at known temperatures.
- **Calibration tables**: you enter a table containing about 100–200 sensor readings over the entire working temperature range. This method can produce the most accurate results of all but also requires the most experimental data. The CTC100 can directly read the calibration tables provided by some sensor manufacturers. See the "Custom calibration tables" topic on page 24 for more information.
- **Offset/gain**: the temperature values produced by any of the above calibrations can be multiplied by a constant and then added to a second constant.

Type

Channel.Cal.Type { DT-470, DT-670, Si410, Si415, Si430, Si435, Si440, Si540, S700, S800, S900, Custom } (if Sensor = Diode)
Channel.Cal.Type { RX-102A, RX-103A, RX-202A, RO600, R400, R500 } (if Sensor = ROX)
Channel.Cal.Type { IEC751, US, Custom } (if Sensor = RTD)
Channel.Cal.Type { "100 ê", "300 ê", "1000 ê", "2252 ê", "3000 ê", "5000 ê", "6000 ê", "10000 ê B", "10000 ê H", "30 kê", "100 kê", "300 kê", "1 Mê", Custom } (if Sensor = Therm)
Channel.Cal.Type { File, Standard } (if a custom calibration table is loaded)

The Calibration Type control determines which of the CTC100's preloaded calibration tables is used to convert raw sensor readings to temperature values. If a diode, RTD, or thermistor is in use, the Type setting also has an option to produce a custom calibration table from user-entered calibration coefficients.

Changing the sensor type has no effect on how the raw sensor reading is acquired; for example, it does not affect the channel's input range or excitation current.

If the selected channel uses a custom calibration table that was loaded from a file on a USB device, its calibration type reads "File". To stop using the custom calibration, touch the Type button and select "Standard". The Type button then reverts to the normal list of calibration types supported by the I/O card. To return to using the calibration file, unplug the USB device with the file (if it is still plugged in) and then plug the device in again.

The available calibration types depend on the sensor type.

Diodes: Choose from the list of commercial cryogenic diodes. See the table on page 3 for more information on standard diode calibrations.

ROX: Choose from the list of commercial ruthenium oxide sensors. See the table on page 3 for more information on standard ROX calibrations.

RTDs: Choose "IEC751" for RTDs with an alpha of 0.00385; "US" for RTDs with an alpha of 0.00392; or "Custom" to enter your own Callendar–van Dusen calibration coefficients.

Thermistors: The available calibration types are named according to the resistance of the thermistor at 25°C. Thermistors from Omega, TE Connectivity (formerly Measurement Specialties, Inc. and, before that, YSI), and others that conform to the same calibration curve are supported. Note that there are no international standards for thermistors. Therefore, thermistors from different companies may not be compatible with each other or with the CTC100's built-in calibrations even though they have the correct resistance at 25°C.

The CTC100 uses ASCII character 234 for the Ohms symbol. To type this character on a Windows computer, hold down the alt key and type 0234 on the number pad. On Windows computers the character appears as a letter "e" with a circumflex accent.

A (Sensor = RTD, thermistor, and diode only)
B (Sensor = RTD, thermistor, and diode only)
C (Sensor = RTD, thermistor, and diode only)
R0 (Sensor = RTD only)

Channel.Cal.A *0.0*
Channel.Cal.B *0.0*
Channel.Cal.C *0.0*
Channel.Cal.R0 *0.0*

These settings allow you to enter Steinhart–Hart, Callendar–van Dusen, or polynomial fit coefficients for your RTD, thermistor, or diode sensor, respectively. The settings are only available if the Sensor control is set to one of these three sensor types and the Cal Type control is set to "Custom".

The use of calibration coefficients can result in more accurate measurements than the preloaded calibration tables. In many cases, commercial sensors come with these coefficients.

The values can only be changed if the calibration type is set to "Custom" and a custom calibration table is not in use.

RTDs: If the sensor is an RTD, A, B, C, and R0 are the constants for the Callendar–van Dusen equation. The temperature t is calculated from the RTD resistance R_t based on the following equation:

$$R_t = R_0(1 + At + Bt^2 + (t - 100)Ct^3) \text{ below } 0°C$$
$$R_t = R_0 (1 + At + Bt^2) \text{ above } 0°C$$

R_0 is the resistance of the RTD at 0°C, expressed in ohms; t is the temperature in °C.

When the calibration type is set to IEC751 or US, the A, B, and C settings are automatically changed to the values for that particular calibration, and the A, B, and C controls are greyed out and cannot be modified (to modify these values, select the "Custom" calibration type). The value of R0, however, is not preset and can still be modified.

The Callendar-van Dusen equation is not an exact representation of an RTD's characteristics, but is accurate to about 50 mK in the range -200 – 400°C. In contrast, class A commercial RTDs that have not been individually calibrated are accurate to 150 mK at 0°C and 950 mK at 400°C.

If you are calibrating your own sensor and the calibration points are separated by less than about 50°C, it's usually easier and more accurate to load the calibration in the form of a calibration table instead of calculating the Callendar–van Dusen coefficients.

Thermistors: If the sensor is a thermistor and the calibration type is set to "custom", the A, B, and C settings are the Steinhart–Hart coefficients. The temperature T (expressed in K) is calculated from the thermistor resistance R (in ohms) based on the following equation:

$$T = (A + B \cdot \ln(R) + C \cdot \ln^3(R))^{-1}$$

If a standard thermistor calibration is selected, the A, B, and C controls are automatically changed to show the best-fit coefficients for whichever curve is selected. These figures are approximations only and are not actually used to calculate the temperature unless the calibration type is changed to "Custom".

Diodes: If the sensor is a diode and the calibration type is set to "custom", the A, B, and C settings are a polynomial fit to the diode calibration curve:

$$T = A - BV - CV^2$$

where T is the temperature in Kelvins and V is the voltage across the diode in volts. Note that polynomial fits are only accurate within a limited temperature range.

If a standard diode calibration is selected, the A, B, and C controls show best-fit coefficients for whichever curve is selected. These figures are approximations only and may not produce the same results as the standard calibration curve.

A standard diode or bipolar junction transistor can be connected to the CTC100 and used as a low-cost temperature sensor. In this case a custom calibration must be used. If the voltage across the diode is measured at two known temperatures, the calibration coefficients can be calculated as follows:

$$B = -(T_1 - T_2) / (V_1 - V_2)$$
$$A = T_1 + (V_1 \cdot B) + 273.15$$
$$C = 0$$

where V_1 is the voltage (expressed in volts) at temperature T_1 (expressed in °C), and V_2 is the voltage at temperature T_2. The resulting calibration is a linear approximation. For greater accuracy, a custom calibration table should be used instead of the A, B, C coefficients; see page 24.

Offset
Gain
Channel.Cal.Offset *0.0*
Channel.Cal.Gain *1.0*

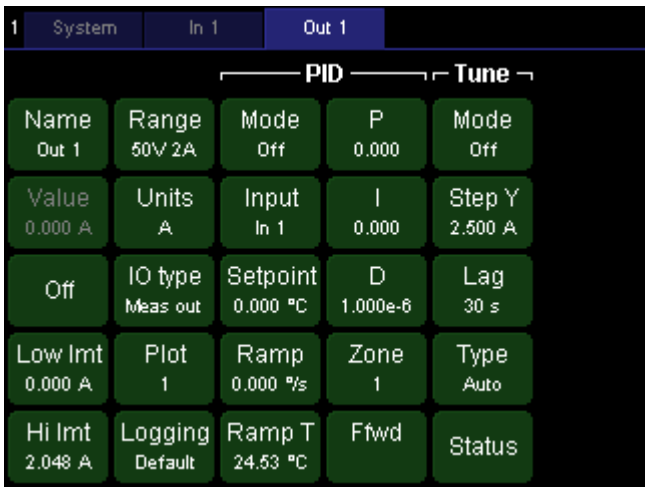
The offset/gain filter modifies the value of an input channel as follows:

$$\text{output} = (\text{input} \cdot \text{gain}) + \text{offset}$$

where input is the input to the offset/gain filter, and output is the output of the filter. This filter can be used as a simple way to adjust sensor calibrations.

The offset/gain filter is applied after the sensor calibration and after the "follow" filter, but before the difference, lowpass, and derivative filters.

Setup screen for channels Out 1 and Out 2



Name
Channel.Name = "*NewName*"

Touch this button to change the name of the channel. The name must have 10 or fewer characters.

Value
Channel.Value *0.0*

This button can be used to manually set the heater output. If the outputs have not enabled by pressing the Output Enable key, the channel value button is greyed out and its value can't be changed. If the outputs are enabled but the channel's PID feedback is turned on, changing the heater output will have no effect.

Off
Channel.Off

Pressing this button immediately sets the PID feedback mode to Off, cancels PID tuning, and sets the channel's output to zero or the "Low lmt" value, whichever is higher. Unlike the Output Enable key, which turns all of the CTC100's outputs off, the Off button only affects one channel.

Low lmt
Channel.LowLmt *0.0*

Touch this button to place a lower limit on the output. If the minimum is greater than zero, the output is still set to zero whenever outputs are disabled with the Output Enable key. Limits are

always expressed in the same units as the value. If the output units are changed, the limits are not automatically converted to the new units and must be updated by the user.

Hi lmt

Channel.HiLmt *0.0*

Touch this button to set an upper limit on the output, for example to prevent the PID feedback loop from delivering excessive power to a heater. If the high limit is less than the low limit, the low limit takes precedence.

Range

Channel.Range { "50V 2A", "50V .6A", "50V .2A", "20V 2A", "20V .6A", "20V .2A", Auto }

Using this control, the maximum heater voltage can be set to 50 or 20V, and the maximum heater current can be set to 2, 0.6, or 0.2 A. For safety the voltage should ideally be set to 20V if the full 50V is not needed, however, doing so does not improve the performance of the output. On the other hand, selecting a smaller current range does reduce the output noise and improve the accuracy. In the Auto setting, the range is selected based on the heater resistance and the Hi Lmt value.

Units

Channel.Units { W, A, V }

Determines whether the heater output is specified in W (watts), A (amps), or V (volts). The "W" setting generally works the best with resistive heaters, since the temperature of such heaters is roughly a linear function of the power supplied to the heater. If the output is connected to a fan or a thermoelectric cooler, the "A" setting is preferable. The default setting is "W".

IO type

Channel.IOType { "Meas out", "Set out" }

Each output channel has a DAC that produces the output and an ADC that measures it. The "IO type" setting determines whether the value displayed on the CTC100's screen is the one measured by the ADC ("Meas out") or the one that was sent to the DAC ("Set out"). The measured output can differ from the set output if, for example, the heater has become disconnected, or the output's compliance voltage has been exceeded. The default setting is "Meas out".

Plot

Channel.Plot { 1, 2, 3, 4, 5, 6, 7, 8 }

Indicates which plot this channel will appear in when the "Plot" screen is showing, the plot type is Custom (see the "Plot Screen" section above), and the channel is selected on the "Select" screen. Choose one of eight plots for the channel to appear in, where plot 1 is the uppermost plot. If no channels are assigned to a given plot, the plot won't appear on the "Plot" screen. For example, if all selected channels are assigned to plot 4, plot 4 will occupy the entire Custom plot screen.

Logging

Channel.Logging { Off, "0.1 s", "0.3 s", "1 s", "3 s", "10 s", "30 s", "1 min", "3 min", "10 min", "30 min", "1 hr", Default }

By default, each channel's value is written to the log at a global log rate that is set from the System Setup screen (System.Log.Interval). The Logging button makes it possible to override the global log rate for individual channels.

Setup screen for channels Out 1 and Out 2: PID column

Input
Channel.PID.Input "*Input channel*"

This setting determines which the temperature sensor the PID feedback loop tries to regulate. It's possible to use one temperature sensor as the input for more than one PID loop.

Mode
Channel.PID.Mode { Off, On, Follow }

Off: PID feedback is inactive and the output can be controlled with the "Value" button.

On: PID feedback actively controls the heater output, ideally maintaining the input channel at the setpoint.

Follow: the output mirrors the input channel. A gain and offset can be applied; see "Zero pt" and "Gain", below. There is no PID feedback in follow mode.

Setpoint
Channel.PID.Setpoint *0.0*

The Setpoint is the temperature at which the PID feedback tries to keep the input channel. The setpoint is expressed in the same units as the input channel.

Zero pt (Follow mode only)
Gain (Follow mode only)
Channel.PID.ZeroPt *0.0*
Channel.PID.Gain *0.0*

These controls are only available when the PID Mode is set to "Follow".

In "follow" mode, heater power is directly controlled by one of the CTC100's inputs, rather than by a feedback loop or from the front panel. The Zero Point and Gain settings allow the user to scale the output. The heater output is given by the following equation:

$$\text{Output} = (\text{Input} - \text{Zero pt})\text{Gain}$$

Note that the output is zero when the input is equal to the zero point.

Ramp
Channel.PID.Ramp *0.0*

This button is used to set the ramp rate in degrees per second, controlling how quickly the CTC100 heats or cools your system.

When the user changes the setpoint, the CTC100 gradually adjusts the ramp temperature (see the description of the "Ramp T" control, below), increasing or decreasing it at the ramp rate until it reaches the new setpoint. The PID feedback loop, in turn, attempts to control the sensor temperature such that it tracks the ramp temperature. Assuming the feedback is properly tuned and that your cryogenic hardware can respond quickly enough, the sensor temperature should rise or fall at the ramp rate until it reaches the new setpoint.

If Ramp is set to zero, ramping is disabled and the CTC100 heats or cools your system at the maximum possible rate.

Ramp T
Channel.PID.RampT *0.0*

The temperature that the PID feedback is trying to maintain. This is an internally-generated value that depends on the setpoint and the ramp rate.

When the feedback is disabled, Ramp T automatically tracks the sensor temperature. When the feedback is enabled, Ramp T gradually increases or decreases at the ramp rate until the setpoint is reached. This ensures that the temperature smoothly ramps from its initial value to the setpoint at a user-determined rate. If this behavior is undesirable (for example, if the ramp rate is very slow and

you want to reach the setpoint quickly), Ramp T can be manually set to another value, typically the setpoint.

Once it reaches the setpoint, Ramp T remains there until the setpoint is changed or the feedback is disabled. If the setpoint is changed, Ramp T increases or decreases at the ramp rate until it reaches the new setpoint. If the feedback is disabled, Ramp T immediately starts to track the sensor temperature.

The Ramp T button can be used to monitor the progress of temperature ramps. Although the sensor temperature could also be used for this purpose, it's subject to noise, external disturbances, and other artifacts that in some cases could make it difficult to determine the intended temperature.

P
I
D

Channel.PID.P *0.0*
Channel.PID.I *0.0*
Channel.PID.D *0.0*

Sets the proportional, integral, and derivative gain factors for PID feedback. The PID equation is:

$$\text{Output}_t = Pe_t + 0.5IT((e_0 + e_1) + (e_1 + e_2) + \ldots (e_{t-2} + e_{t-1}) + (e_{t-1} + e_t)) + (D/T)(e_t - e_{t-1})$$

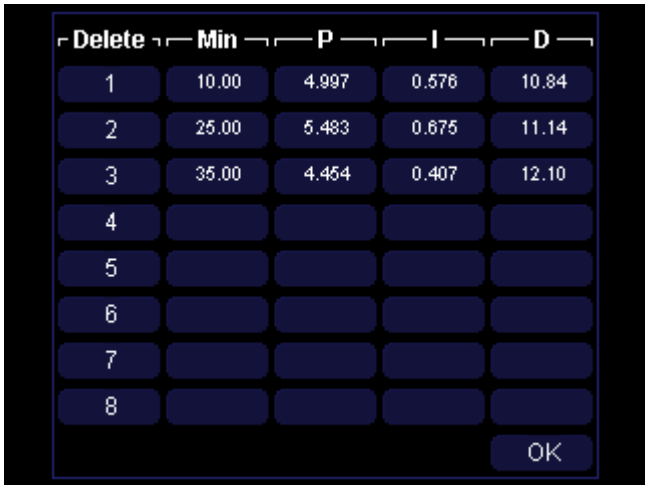
where P, I, and D are the derivative gains, e_t is the error (the difference between the setpoint and the PID input signal) at time t, and T is the ADC sampling time. Thus, larger values of P, I, or D produce a faster feedback response. Negative values of P, I, and D should be used if the output drives a fan or other device that cools the sample.

Zone

Channel.PID.Zone { 1, 2, 3, 4, 5, 6, 7, 8, Auto }

The Zone setting stores up to eight sets of feedback parameters. Each set can be associated with a temperature range and automatically recalled when the temperature of your experimental system enters that range. The zone can also be manually selected, ignoring the temperature.

To view a table of all stored feedback parameters, touch the 'Zone' button and then select 'Edit'. The table that appears has a row for each zone and columns for the zone's minimum temperature, the P, I, and D feedback gains, and the input channel. By default, zone 1 is selected and contains the current values of these parameters; the rest of the table is empty. Touch one of the parameter cells to modify its value. If a particular set of parameters is no longer needed, touch its zone number in the 'Delete' column to clear the entries for that location.



The PID zone editor

To manually select a zone, touch the 'Zone' button and select one of the zone numbers, 1–8. The feedback parameters immediately change to the values stored in the corresponding row of the Zone table. If the selected zone contains empty cells, the feedback parameters don't change and are copied into the empty cells.

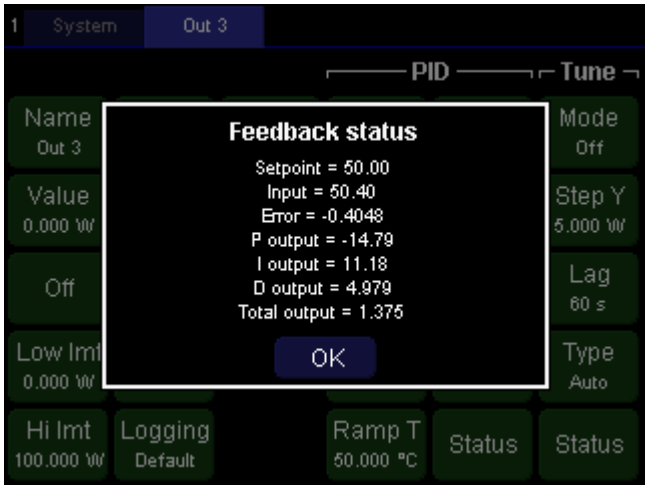
Whenever the feedback parameters change (for example, if the feedback is tuned), the selected zone is automatically updated with the new values.

To have the CTC100 automatically select zones based on the temperature, assign each zone a minimum temperature using the "Min" column of the memory table. The min temperatures can be in any order; they do not have to be monotonically ascending or descending. Next, set the zone to 'Auto'. The CTC100 automatically selects the zone with the largest Min value that is less than the ramp temperature ('Ramp T'). Memory locations without min values are never recalled in 'auto' mode.

Status

Channel.PID.Status

Touch the "Status" button to see the current PID inputs and outputs or, if the PID feedback loop is not functioning, a message explaining why. The information is updated once every second.



PID status display

Setup screen for channels Out 1 and Out 2: Tune column

This column is used to configure the PID autotuner. See the "Automatic PID Tuning" section for more details.

Step Y

Channel.Tune.StepY *0.0*

This setting determines how much the autotuner changes the heater power. It should be large enough to increase the temperature by several degrees, or significantly more than any noise or other temperature variations that would normally occur over several minutes. If Step Y is too small, autotuning will fail or may succeed but produce inaccurate feedback parameters. If Step Y is too large, the heater may become unacceptably hot.

If autotuning is in progress when Step Y is changed, the old value of Step Y is used.

Lag

Channel.Tune.Lag *0.0*

Controls how long the autotuner waits before it first checks the response of the system to the output disturbance. This time should be long enough for the temperature to rise noticeably after the output is increased by Step Y. If Lag is too small, the autotuner will mistake small noise spikes for the system's response to the output disturbance. If Lag is much larger than it needs to be, the autotuner will produce inaccurate results.

Changes to the Lag setting doesn't affect any autotuning algorithms that are currently in progress.

Status

Channel.Tune.Status?

Touch this button to view the progress of the autotuner.

Type

Channel.Tune.Type { Cons, Moderate, Aggr, Auto }

Controls the PID tuning rules used by the auto-tuner. Changing the tuning type immediately updates the PID feedback gains if the PID feedback has been auto-tuned since the CTC100 was turned on.

Cons (conservative): results in minimal overshoot (ideally, zero overshoot) but very slow response.

Aggr (aggressive): results in much faster feedback response but typically ~25% overshoot.

Moderate: provides intermediate results.

Auto: uses the conservative setting for step response tuning and aggressive for relay tuning. Works well if the step response tuner is used for an initial rough tuning at room temperature and the relay tuner is used for a final tuning once the system has reached its target temperature.

Regardless of the tuning type, the relay tuner always tunes more aggressively than the step response tuner. See "Aggressive, moderate, and conservative tuning" on page 36 for more information.

Setup screen for analog I/O and digital I/O channels

Depending on their "IO type" setting, the general-purpose analog and digital I/O channels have the same front-panel settings as either the sensor input or the heater output channels. These settings have been described above.

The general-purpose channels don't offer built-in sensor calibration curves, but they do accept custom calibration tables.

IO type

Channel.IOType { Input, "Set out", "Meas out" }

The analog and digital I/O channels can be inputs or outputs. Two output types are available: if "Set out" is selected, the desired output is shown, while if "Meas out" is selected, the actual output is measured and the measured value is reported.

PID feedback functionality is only available if the channel is an output, while alarm and calibration functions are only available if the channel is an input.

Polarity

Channel.Polarity { 0, 1 }

This setting only applies to the Relays channel on the digital I/O card. Changing the polarity reverses the state of all four relays.

The Polarity setting ensures that the relays are in an acceptable state when the CTC100 is switched off. When the Polarity is 0, the relays revert to the "alarm off" state when the CTC100 is switched off. When the Polarity is 1, they revert to the "alarm on" state.

The "Relays" value shown on the front panel does not change when the polarity is changed. This value is the sum of four individual relay values: relay A can have a value of 0 or 1, relay B can have a value of 0 or 2, relay C can have a value of 0 or 4, and relay D can have a value of 0 or 8. The meaning of these values depends on the Polarity setting as shown in the table below.

Relay state →	Default (power off)		Relay value = 0 (alarm off)		Relay value = 1, 2, 4, or 8 (alarm on)	
Back panel pin →	**NC**	**NO**	**NC**	**NO**	**NC**	**NO**
Polarity = 0	Closed	Open	Closed	Open	Open	Closed
Polarity = 1	Closed	Open	Open	Closed	Closed	Open

The "default" state is what the relays revert to when the CTC100 is switched off. If no alarms are configured, they will stay in that state when the CTC100 is turned back on again.

Setup screen for virtual channels

Depending on their "IO type" setting, the virtual channels V1, V2, and V3 have the same front-panel settings as either the sensor input or the heater output channels. These settings have been described above.

Virtual channels don't offer built-in sensor calibration curves, but they do accept custom calibration tables. When configured as an output, virtual channels also offer a cascade control setting.

IO type

`Channel.IOType { Input, "Meas out" }`

Virtual channels can be inputs or outputs. In either case, it is possible to set the channel's value from the front panel or by sending a remote command. However, PID feedback functionality is only available if the channel is an output, while alarm and calibration functions are only available if the channel is an input.

Casc

`Channel.PID.Casc "Output channel"`

Cascade control. A cascade control system consists of two (or more) PID loops. As in a normal PID system, a primary PID loop monitors a temperature that needs to be regulated (the primary temperature). However, instead of driving the physical output (heater, valve, etc.), the output of the primary loop becomes the setpoint for a secondary PID loop. The secondary loop monitors a secondary temperature reading and controls the physical output. The secondary temperature reading is typically a temperature that is not in and of itself critical to the application, but responds more quickly to the control output than the primary reading.

For example, the temperature of an incubator might need to be kept constant using a forced-air heater. In this case, the primary temperature is the air temperature inside the incubator, while the secondary temperature might be the temperature of the hot air entering the incubator (the vent temperature). The primary loop controls the air temperature in the incubator by telling the secondary loop how hot the vent air should be. The secondary loop regulates the temperature of the vent air by controlling the power to a heater coil. The advantage of cascade control is that variations in the vent temperature can be accounted for much more quickly than would be possible with a single PID loop.

To use cascade feedback, select one of the CTC100's virtual channels (V1, V2, or V3) to host the primary loop. Make sure the direction of the channel is "Set out" or "Meas out", and then touch the button labeled "Casc". You should see a list of output channels. Select the secondary channel from this list. Its PID setpoint will continuously track the value of the virtual channel.

To disable cascade control, touch the "Casc" button and then touch the selected channel to de-select it.

Firmware updates

The CTC100's firmware can be updated by copying a firmware file onto a USB stick, plugging the USB stick into the CTC100, and running a macro. Besides the CPU firmware, each of the six I/O cards and the front panel has its own firmware that can be upgraded.

In most cases, firmware updates do not affect your settings or I/O card calibration data. However, if the meaning of a particular setting has changed in the new firmware, that setting may revert to its default value.

CPU firmware updates

1. Copy the supplied update files to the root directory of a USB flash drive or hard drive.
2. Plug the USB device into the CTC100 and wait until a window that says "Opening USB drive" appears and then disappears.
3. The CTC100 erases the existing firmware and then loads the new firmware. The entire process should take about 6 minutes. When complete, the message "Firmware update was successful" should appear.
4. Remove the USB device and erase the files from it so that you don't accidentally reprogram the CTC100 again.
5. At this point, the old firmware is still running. Turn the CTC100 off and back on again to start using the new firmware.

I/O card firmware updates

1. Copy all files in the firmware update package to the root directory of a USB flash drive or hard drive.
2. Plug the USB device into the CTC100. The firmware update will begin automatically.
3. While the firmware is updating, the front-panel LCD and fan turn off. After about 15 seconds, power is restored to the front panel and a status message is displayed on-screen. This process may repeat several times, once for each I/O card that needs to be updated.
4. Remove the USB device and erase the files from it so that you don't accidentally reprogram the I/O cards again.
5. At this point the new firmware is already running; however, after all firmware updates are complete, the CTC100 should be turned off and back on again to ensure that the I/O card is properly initialized.

Replacing the clock battery

The CTC100 has a CR2032 battery that maintains the current time and date.

If the battery fails, the instrument's time and date will be incorrect. User settings and factory calibration data is stored in nonvolatile memory and are not affected.

The battery can be replaced as follows:

1. Unplug the CTC100 from the wall. This is important since otherwise live AC voltages may be present inside the chassis even if the CTC100 is switched off.
2. Remove the four black screws that secure the top cover. Lift the cover off of the instrument.
3. Looking at the front of the CTC100, the battery should be clearly visible. It is a 20 mm diameter coin cell located on the left side of the instrument 8 inches behind the front panel. The CTC100 only has one battery.
4. Remove the battery by pulling the coin cell toward you and sliding it to the left. It can be somewhat difficult to remove.
5. Install a new battery with the + side facing toward the rear of the instrument.
6. Replace the CTC100's lid.
7. After turning the CTC100 back on, reset the instrument's time and date and any other user settings.

A new battery should last for 6 years.

Removing an I/O or CPU card

If it's necessary to repair or replace an I/O card or the CPU card, the following process can be used to remove the card from the chassis:

1. Unplug the instrument and remove the top lid.
2. Carefully remove the bottom lid. After removing the four screws, pull the lid about an inch away from the case and look for the fan that's mounted on the lid and connected to the front panel. You'll need to unplug the fan before the lid can be completely removed. It's difficult to reach the plug since it's at the top of the front panel, behind the big silver power supply. For better access it may be necessary to remove two of the screws that hold the power supply in place (the top screw on the left and right sides of the case) so that the power supply can be rotated out of the way.
3. Remove the screws on the left and right sides that are normally covered by the bottom lid. These screws are only needed for shipping; if you're not going to be shipping the instrument anywhere, they don't need to be re-installed when you re-assemble the chassis.
4. Look at the bottom of the instrument. You should clearly see the bottom of the backplane card, which is the largest card in the instrument and the only one that's mounted horizontally. At the edge of the backplane card closest to the display are two screws. Remove these two screws. These screws are only needed for shipping and don't need to be re-installed when you re-assemble the chassis.
5. On the back panel, use a deep 9/16 inch socket wrench to remove the 4 BNC nuts on the analog I/O connectors.
6. Remove the 4 screws or nuts above and below the RTD/thermistor/diode inputs. It's especially important to re-install these screws when re-assembling the instrument. Without these screws, the input connectors will not be grounded and the CTC100's temperature readings will be inaccurate due to electromagnetic interference.
7. Remove the 2 screws at the top of the back panel above the big vent.
8. Remove the 1 screw just below the words "AC power 88–264V 4A".
9. Remove the 2 nuts above and below the DIO port.
10. If the instrument has a GPIB port, remove the 2 nuts above and below the GPIB port.
11. At this point, only the screws for the heater driver terminals and the RS-232 port (if the instrument has an RS-232 port) should remain on the back panel. Do not remove these screws.
12. Push the entire card and backplane assembly away from the back panel by about half an inch. Usually the best way is to gently push on the In 1 & 2 port with one thumb and on the DIO port with the other, but make sure not to apply so much force that you bend the connectors.
13. It should now be possible to pull the I/O or CPU card out.

Remote programming

The CTC100 can be remotely controlled over RS-232, USB, Ethernet, and the optional GPIB port. All of these ports are always enabled and accept the same commands. In addition, the front panel controls are always enabled.

To control the CTC100 remotely, you transmit lines of ASCII text to one of its ports. No action is taken until one of the following end-of-line characters is received:

- a linefeed (decimal 10 = hex A = '\n'), or
- a carriage return (decimal 13 = hex D = '\r') followed by a linefeed (decimal 10 = hex A = '\n').

The CTC100's replies always end with a carriage return followed by a linefeed.

Each line of text sent to the CTC100 is treated as a macro, meaning that it can contain one or more instructions as well as conditional statements and repeated blocks. The macro starts running immediately and, if it takes long enough to complete, its progress can be monitored on the Program screen. While the macro is running, more macros can be sent to the CTC100. Up to 10 macros can run at the same time, although only the first four are shown on the Program screen.

Any macro sent to one of the I/O ports must be written on a single line, otherwise it will be interpreted as several macros to be run concurrently. Each macro can have up to 4096 characters, while individual instructions or arguments can have up to 1024 characters each. Instructions and arguments are case insensitive and can be separated by one or more whitespace characters as well as by special characters such as parentheses, brackets, equals signs, etc.

Macros can also be stored as text files on a USB memory device. When the USB device is plugged into the CTC100, the macro can be run from the Program window or called from other macros, just like a saved macro. It's easier to edit long macros when they are saved as text files, since they can then include multiple lines and comments.

Macros can be saved under a name, and a macro can call other, saved macros by name (macros must not, however, call themselves recursively). If a macro is saved under a name that is the same as an instruction, the saved macro takes precedence if the command is issued with a capital first letter; the instruction takes precedence if the command has a lower-case first letter.

Most macro instructions correspond to buttons on the Setup screens, and the instruction names are the same as the button names. For example, the instruction to change the RS-232 baud rate is `System.COM.RS-232`; the corresponding button is found in the "System" tab of the Setup screen, in the "COM" column, and is named "RS-232".

Connecting to the CTC100

RS-232

The CTC100's RS-232 port is a 9-pin female D-sub connector. The connector is not present if the GPIB option is installed. The CTC100 is a DCE device and should be connected to a PC with a straight-through, DB9 male to DB9 female RS-232 cable (sometimes called a "modem cable", as opposed to a "null modem cable"). Depending on the capacitance of the cable, the maximum cable length is about 50 feet at 9600 baud and 4 feet at 115200 baud. The pin assignments are:

Pin	Description
1	Not connected
2	CTC100 data out
3	CTC100 data in
4	Not connected
5	Signal ground
6	Not connected
7	RTS (Request to Send; CTC flow control in)
8	CTS (Clear to Send; CTC flow control out)
9	Not connected

The RS-232 outputs (pins 2 and 8) are not active unless a voltage greater than +2.7 V or less than –2.7 V is present at the receive pin (pin 3). The outputs are ±5V instead of the more standard ±10V, and may therefore not work with some older computers. However, the CTC100 can still receive ±10V signals. The RS-232 interface does not echo characters back as they are received.

The RS-232 interface can use RTS/CTS hardware flow control; if enabled, the CTC100 pulls pin 8 high to indicate that the PC can send data, and low to indicate that the PC should not send data. Similarly, the CTC100 stops sending data whenever the PC pulls pin 7 low.

The RS-232 interface has no parity, 8 bits, and 1 stop bit. The baud rate can be selected from the System Setup menu.

USB device port

The CTC100 has a USB 1.1 device interface that can be connected to a PC with a standard USB A-to-B cable. The CTC100 appears on the PC as a virtual COM port, and any software that can communicate with an RS-232 port can be used to send remote commands to the CTC100.

The USB driver for Windows and Mac OS X can be downloaded from the FTDI website:

https://www.ftdichip.com/Drivers/VCP.htm

The port's baud rate and hardware flow control settings on the PC must match the settings on the CTC100. By default the baud rate is 9600 and flow control is off.

If the CTC100 is turned off and back on again while the PC application is running, the PC application must be closed and re-opened before it can communicate with the CTC100. This is a limitation of Windows that applies to all Communications Device Class (CDC) USB devices.

USB host port

Macros can be imported from a USB hard drive or flash key. Save the macro as a text file (its name must end with ".txt") and copy it to a folder named "macros" in the topmost directory of the USB device. When the device is plugged into the CTC100, up to six buttons with the names of the text files appear in the System.Macro menu. A macro can then be run by touching the button with its file name. If the USB device is unplugged, the corresponding macro buttons disappear but any running macros continue to run.

One macro can be saved in the topmost directory of the USB device under the name "autorun.txt". This macro automatically runs each time the USB device is plugged into the CTC100.

The USB storage device should have a FAT16 or, preferably, a FAT32 format. It should not contain any files other than the files for the CTC100.

GPIB

CTC100 units can be ordered with or without a GPIB port. If GPIB is requested, it replaces the RS-232 port. Although any standard GPIB cable can be used, due to space restrictions a single-ended cable (such as a National Instruments X5 cable) is recommended.

No more than three GPIB cables should be stacked on a single GPIB connector, and no more than 14 devices can be connected to a single GPIB interface. The total length of all GPIB cables must not exceed 2 meters per instrument or 20 meters, whichever is less.

Ethernet

Remote commands can be sent to the CTC100's Ethernet interface using the following methods. The CTC100 automatically determines which method is being used.

- Sending ASCII text in a raw TCP/IP stream to port 23
- Sending ASCII text in UDP packets addressed to port 23
- A telnet connection to port 23

The IP address must be set before the Ethernet interface can be used.

The CTC100 only accepts commands from one client. After a command addressed to port 23 is received, the CTC100 ignores commands from all other clients until the System.IP.Close button is pressed, the client closes the connection, the connection times out due to inactivity, or the instrument is rebooted.

It's not necessary to connect to your building's network to use an Ethernet connection; the CTC100 can be connected directly to your computer with a standard Cat 5 cable.

Use the following procedure to test an Ethernet connection:

1. Connect the CTC100 to your computer with a standard Cat5 Ethernet cable.
2. Enter a suitable IP address into the CTC100's System menu. If the CTC100 is directly connected to the computer (nothing else is connected to the network), assign 192.168.0.1 to the computer with a subnet mask of 255.255.0.0, and 192.168.0.2 to the CTC100.
3. Open a command line on your computer and try pinging the CTC100, e.g. type `ping 192.168.0.2` (or whatever the CTC100's IP address is).
4. Install a Telnet client, such as Tera Term, on your computer. Open a new connection with Host set to the CTC100's IP address and Service set to Telnet. If using Tera Term, select Setup > Terminal and change the Receive New-line to CR+LF.
5. If a connection was successfully made, the CTC100's System.IP.Close button should no longer be greyed out. If you touch the System.IP.Status button, you should see information about the connection.
6. Type `popup hello` and press Enter. A popup window should appear on the CTC100's screen.
7. Type `description` and press Enter. The CTC100 should return an instrument description string.

Troubleshooting communications problems

If the remote interface does not respond at all, try the following:

- If using RS-232, make sure the baud rate is set correctly.
- Make sure that each line of text sent to the CTC100 ends with a linefeed character (decimal 10 = hex 0x0a = '\n').
- Try viewing the System.COM.History window. This will tell you if the CTC100 is receiving anything at all.

- Try sending the command "popup hello". This command produces a response that is easy to see (a popup window appears) and only requires that the interface work in one direction. Next, send the command "description", which should generate a response.

Communication, assembly, and run-time errors

If the CTC100 is unable to receive a macro due to a problem with the I/O port, a communication error may be generated and the macro does not run.

If the macro is successfully received, the CTC analyzes it and any macros that it calls to ensure that:

- Each instruction is valid, and
- The arguments for each instruction are valid; for example, if the instruction takes an integer value, the argument must be an integer; if the instruction has a list of acceptable values, the argument must be one of those values. Numeric values are not tested at this time to see if they fall within acceptable limits, since those limits may change as the macro runs.

If the macro fails one of these tests, it doesn't run and an assembly error is reported. If the System.COM.Verbose setting is Medium or High, the CTC100 reports the error by sending an I/O port message that begins with the word "Error". If the Verbose setting is "Low", a message is placed on the error queue and can be retrieved with the "geterror" instruction.

At this point, the macro is displayed on the Program screen and starts to run. As each instruction is executed, errors can occur if:

- The instruction tries to change a value that can't be changed; for example, it tries to set the value of an input channel.
- The instruction existed at assembly time but not at run time; for example, the name of a channel was changed after assembly, and the instruction uses the old channel name.
- The instruction tries to set a parameter to a value outside the allowed limits.

If a run-time error occurs, the instruction in question is not executed, but the macro continues to run. If Verbose is set to Medium or High, an error message is sent to the I/O port; if Verbose is set to Low, a message is placed on the error queue.

Concurrent macros

Because macros can run for a long period of time or even indefinitely, when the CTC100 receives a macro over an I/O port, it may start running before the previous macro has finished. In addition, it's possible to run multiple instances of a saved macro simultaneously.

The CTC can run up to 10 concurrent macros received over any one I/O port and up to 50 concurrent macros from all sources combined (including the startup macro, macros received over all of the I/O ports, and macros started from the Program screen). If more than this number of macros is received, a "Too many macros" assembly error is generated and the macro does not run.

If the CTC is turned off and turned back on again, macros that were running when the CTC100 was turned off are not restarted.

Macros start running in the order that they were received. If each message sent to an I/O port contains only one instruction, the instructions always run sequentially in the order that they were sent.

Macro names

Each running macro has a name that can be used by the `kill` instruction to stop the macro and also appears in a tab on the Program screen. It is possible to have two or more macros with the same name running.

If a macro is started by a remote command with 32 or fewer characters, the macro name is the same as the remote command. If the command has more than 32 characters, the CTC100 assigns the name "Program XY", where XY is a two-digit number.

If a macro is started from the Program screen, its name is the text in the Input field. If the Input field contains more than 32 characters, the macro name is "Program XY", where XY is a two-digit number.

If the macro was started by pressing a macro button on the System screen, the macro name is the text on the macro button. If the name is too long for the button and has been truncated on screen, the full name of the macro is used.

A macro can change its own name with the `name` instruction.

Use "`kill.list`" to get the names of all currently-running macros.

Command syntax

instruction = argument
instruction += argument
***instruction*?**

Many instructions must be followed by some sort of value. The value must be separated from the instruction by whitespace and/or an optional equals sign. Numeric values can be incremented using the + operator. There is no – operator, but the + operator can be used with negative arguments.

If the argument is selected from a list, it can also be incremented using the + operator. An integer argument must be supplied that indicates how many places to advance in the list. If the value is incremented past the end (or beginning) of the list, it wraps back to the beginning (or end) of the list.

If a question mark follows the instruction, no argument should be provided, and the CTC100 replies with the current value of the setting. The reply also appears on the Program screen if the appropriate tab is selected.

For example,

```
"Out 1.value" = 5
```

sets the value of channel "Out 1" to 5 watts. The equals sign is optional and everything is case-insensitive. Since the channel name "Out 1" has a space, the instruction has to be in quotes or it will be interpreted as two separate instructions. The argument must be outside the quotes.

To reduce the need for quotes, spaces can be omitted from instructions. For example:

```
"Out 1.IO type" = "meas out"
```

is equivalent to:

```
Out1.IOtype = "meas out"
```

Spaces cannot be omitted from arguments.

The command:

```
"Out 1.value" += 1
```

increases the value of channel "Out 1" by 1 watt. The equals sign and the spaces before and after the + are optional. This command:

```
"Out 1.value" += -1
```

decreases the value of channel "Out 1" by 1 watt, while the query:

```
"Out 1.value?"
```

is a request for the value of channel "Out 1".

```
In1.lopass += 1
```

Since the lowpass filter setting must be chosen from a list of possible values ("1 s", "3 s", "10 s", etc.), this instruction sets the filter to the next setting on the list, rather than incrementing the lowpass time constant by one second. For example, if the filter setting was "3 s", it would be "10 s" after the above remote command was issued.

(*instruction*) (argument)
"*instruction*" "argument"

Instructions and arguments are normally separated by spaces. If an instruction contains spaces, the spaces can just be omitted, but if an argument contains spaces, the argument must be enclosed in parentheses or quotation marks.

Parentheses can be nested, but quotation marks cannot. Two quotation marks in a row before an instruction results in an "empty instruction" assembly error.

These two instructions are equivalent:

```
popup "Hello world!"
popup(Hello world!)
```

If an argument doesn't contain any spaces, it doesn't have to be enclosed in quotes or parentheses.

```
popup Hello!
```

Whitespace can be included before or after parentheses or quotes but is not necessary.

[*instructions*]*1*

A group of instructions can be repeated by enclosing it in square brackets and placing the number of repetitions after the right bracket.

```
[print Hello  pause 1 s  print world!  pause 1 s]3
```

Whitespace is not necessary before or after square brackets.

If the left bracket is omitted, all instructions from the beginning of the macro to the right bracket are repeated. If the right bracket is omitted, all instructions after the left bracket do not run.

A negative number after the right bracket causes the group of instructions to repeat indefinitely. Therefore,

```
[print Hello pause 1 s]-1
```

is equivalent to

```
while (1) { print Hello pause 1 s }
```

list
***menu*.list**
***instruction*.list**

List prints a list of top-level menus. If the .list suffix is appended to the name of any menu (System, Channel, System.COM, Channel.PID, etc.), the CTC100 lists the available instructions for the menu or submenu. If appended to an instruction, the .list suffix returns a list of arguments required for the instruction. A question mark after the .list query is optional. The .list suffix is only available for instructions that set some sort of variable and is not available for program flow instructions such as if, while, abort, and kill.

In the examples that follow, the first line is the remote command, while the remaining lines are the CTC100's reply.

```
Out1.list
pid., Name, Value, Off, Low lmt, Hi lmt, Units, IO type, Plot,
Logging, Stats, Points, Average, SD, Selected, Debug, Cycle,
Reset
```

In this case, the reply is a list of arguments that can be appended to the instruction Out1. The dot at the end of pid. indicates that Out1.pid. is another menu, not a complete instruction.

```
Out1.pid.list
Input, P, I, D, Setpoint, Mode, Step Y, Lag, Sq root, Ramp,
Memory, T min
```

Since Out1.pid is a menu, the reply lists the instructions available in the menu.

```
Out1.pid.setpoint.list
pid.Setpoint: float
```

Out1.pid.setpoint is an instruction, so the reply indicates that it takes a single floating-point argument.

```
Out1.value.list
Out 1.Value: float (0.000 - 1200)
```

If an argument has minimum and maximum values, these are shown in the reply. In this case, Out1.value takes a single floating-point instruction in the range 0 – 1200. Most arguments do not have minimum or maximum values.

```
pause.list
pause: float, { ms, s, min, hr }
```

The pause instruction requires two arguments: 1) a floating-point argument with no bounds, and 2) one of "ms", "s", "min", or "hr".

***instruction*.help**

Prints the help text for any instruction that sets a variable. The help suffix is not available for program flow instructions such as if, while, abort, and kill.

if (*condition*) { *instructions* }
while (*condition*) { *instructions* }
else { *instructions* }

Conditional statements consist of an if or while statement followed by a condition, one or more instructions in curly brackets, and possibly an else clause. The condition must be in parentheses if it contains spaces or if it compares two or more values.

The condition can contain numeric values, queries that do not require any arguments, and comparison operators ("!=", "=" "<", "<=", ">", and ">="). The condition can also include '||' (or) operators and '&&' (and) operators. For example, the following causes a macro to wait until temperature In1 is between 39 and 41 degrees:

```
while (In1 < 39 || In1 > 40) { pause 1 s }
```

The pause instruction is not necessary, but it helps to reduce the load on the CPU.

Conditional statements *must* be followed by curly brackets, otherwise the statement has no effect. There is no "else if" statement. Parentheses cannot be used within a conditional statement to affect the order in which parts of the statement are evaluated.

When the name of a channel is used within a conditional term, it is sometimes unclear whether it should be treated as a query of the channel's value or as a character string. In these cases, the channel name (or any other conditional term) can be preceded by a dollar ($) sign to ensure that it is treated as a string, or by a pound (#) sign to ensure that it is treated as a query. For example:

```
if (Out1.PID.Input==$In1) { Out1.PID.Input = In2 }
```

In this example, the dollar sign ensures that the PID input channel is compared with the string "`In1`", not the numeric value of channel In1. Dollar signs can only be used in this way within `if` or `while` conditions.

Conversely, a channel name (or any other conditional term) can be preceded with a hash (#) to force the CTC100 to treat it as a query. Since conditional terms are treated as queries by default, the pound sign is only required if you've changed a channel name to a numeric value that don't contain any letters. For example, if you've renamed one of the I/O channels "2", this statement:

```
while (#2<50) { pause 1 s }
```

pauses the macro until the value of channel "2", not the number 2, is greater than or equal to 50.

#variable 0.0

The hash character defines a variable and assigns it a floating-point value. The value can then be queried with `#variable?` and can also be used as an argument for any instruction that takes a numeric argument.

The `#variable` instruction consists of a hash (#) immediately followed by a variable name. The variable name can be any string up to 32 characters long, but spaces are not allowed within the variable name or between the pound sign and the variable name. Variable names are not case-sensitive. For example:

```
#x=10.2 #x?
```

defines a variable "x" and sets its value to 10.2, then queries the value of x. The equals sign is optional.

Variables can be used by the macro in which they are defined; by any macros called by that macro; and by the macro that called it. A macro cannot access variables defined by other, concurrently-running macros. In addition, once a macro finishes, all variables defined by the macro are deleted. The value of an undefined variable is zero.

When macros are sent over a serial port (as opposed to being loaded from a text file on a USB storage device), the macro can have at most one line, and therefore all variables must be defined and used on a single line. Therefore, if the command

```
#x=10.2
```

is sent over the serial port, and at a later time the command

```
#x?
```

is sent over the serial port, the response is "0" because the CTC runs each line of text as a separate macro, and the variable "x" has not been defined in the second macro.

The four basic arithmetic operations (+, -, /, *), power ('^'), bitwise 'and' ('&'), and bitwise 'or' ('|') can be applied to variables:

```
#x=2 #x+=8 #x-=1 #x*=2.6 #x/=7 #x^=2 #x&=2 #x|=2
```

Spaces are not allowed before the *, /, -, and ^ operators. The equals sign is optional and can be replaced with a space.

The CTC100 has three virtual channels that behave like much like variables. However, while a variable can only be used by the macro that defined it, the value of a virtual channel can be accessed by any macro. The value of a virtual channel also persists after the macro ends. Also, the value of a virtual channel is only updated when an ADC conversion occurs, but the value of a variable is updated without any lag when an instruction changes its value. Finally, virtual channels can be plotted on-screen and logged to USB, while variables cannot (except by assigning their value to a virtual channel).

Once defined, a variable can be substituted for any numeric argument. For example, the macro:

```
#y=5 Out1=#y
```

sets the value of channel "Out 1" to 5.

When `#variable` is used as an argument, a question mark can optionally be added after the variable name to indicate that the variable is being queried:

```
#y=5 Out1=#y?
```

Variables can be used within conditional statements. The macro:

```
#x=0 while (#x<3) { #x+=1 Out1=#x pause 1 s }
```

cycles through the "while" loop three times, setting channel "Out 1" to 1, then a second later to 2, and another second later to 3.

The CTC100's macro system does not support equations. For example, a statement of the form "#x = #y + #z" is not allowed. More generally, each CTC100 argument can only contain a single term.

`#instruction`

A single-instruction query with no arguments, if preceded by a pound sign, can be substituted for any numeric argument. The instruction cannot contain quotes, parentheses, or spaces. For example:

```
Out1.PID.setpoint = #Out2.PID.setpoint
```

sets Out 1's feedback setpoint equal to Out 2's setpoint. The CTC100 automatically appends a question mark to the argument (resulting in the query "Out2.PID.setpoint?"), and evaluates the resulting instruction at run time.

`#list?`

Prints a comma-separated list of variables that have been defined within the macro.

"Macro name"

Macros can be defined with the `define` instruction; by saving a macro from the Program screen; or by plugging in a USB drive containing macros in the form of text files. Once a macro has been defined, it can be called by including its name in another macro (the "parent macro"). When the parent macro is assembled, the macros it calls are expanded to their component instructions. Up to six levels of macro calls are allowed.

Variables declared in the parent macro are also valid in, and can be modified by, the child macro. For example, define a macro called "multiplyXY" by sending the following text to the CTC100:

```
define multiplyXY (#x*=#y)
```

Subsequently, "multiplyXY" can be called to modify the variables of a parent macro:

```
#x=3 #y=4 multiplyXY #x?
12.0000
```

A subroutine macro must consist of one or more complete instructions with arguments. Macro calls cannot be used to substitute text into arguments.

Like normal instructions, macro names are not case-sensitive. However, if a macro has the same name as a built-in instruction, the macro takes precedence if it is called with a capitalized first letter, while the instruction takes precedence if it is called with a lower-case first letter.

Errors: If the child macro contains any invalid instructions, an assembly-time error occurs and neither macro runs.

A macro cannot be both defined and called by another macro; either an assembly-time "not a valid instruction" error will occur, or an older version of the macro will be called instead of the new one.

Remote instructions

In the following listing, words in `Courier` font represent text that may be sent to or received from the CTC100 over RS-232, USB, GPIB, or Telnet, or via a text file on a USB memory device. Words in *`italicized Courier`* are placeholders that should be replaced with other names or values; for example, when the word *`Channel`* appears it should be replaced with the name of a data channel like `In1`. If an argument is enclosed in quotation marks and contains spaces, it must be enclosed in quotes or parentheses. If the argument doesn't contain any spaces, the quotes can be omitted.

Miscellaneous instructions

`abort`
Stops the macro. This instruction only affects its parent macro. Use the `kill` instruction to stop other, concurrently-running macros.

`customCal <channel>, <calibration table>`
Loads a custom calibration table. The calibration table must be formatted as described in the "Custom calibration table" section (page 32), except the table must be on a single line, it must be enclosed in quotes, and the maximum table length is 1024 characters. If the channel name contains a space, the space must be included. For example:

```
customCal "In 1", "units = °C 0, 100.00, 10, 103.90, 20, 107.79, 30, 111.67"
```

If the channel name is recognized, the command returns the content of the Cal > Detail screen with newlines removed.

`description`
Writes a string similar to the following to the I/O port:

```
CTC100 Cryogenic Temperature Controller, version: 0.135, S/N
92001
```

It's not necessary to use a question mark with this instruction.

`getLog[.xy][.reset][.v] "`*`channel`*`", `*`time`*
Gets a data point from the log. The first argument is the name of a channel. The second argument is one of the following:

- The desired time of the data point, in milliseconds since 1970. If the time is not available in the log, the point at the closest available time is returned.
- **`first`** to get the oldest point in the log.
- **`last`** to get the most recent point in the log.
- **`next`** to get the point after the one that `getLog` last fetched from the channel. If the next point has not been acquired yet, the CTC100 waits for it to be acquired. If `getLog` has not been used on this channel since the CTC100 was turned on or since `getLog.reset` was last issued, the last point in the log is returned.

If the `.xy` option is added to the instruction, both the time (in milliseconds since 1970) and the value of the point are reported; otherwise, only the value is reported.

The `.reset` option resets the "next" argument for all channels; the next time the instruction "`getLog channel, next`" is issued for any channel, the last point in that channel's log will be reported. No arguments should be used with the "reset" option.

The `.v` (verbose) option adds the name of the channel to the reply. The name of the channel is also added if the `System.Com.Verbose` setting is "High".

For example, the macro

```
getLog.reset   while (1) { getLog "In 1", next }
```

transmits the value of channel In 1 each time a new value is logged.

"`getLog? channel`" returns the number of data points that can be read with "`getLog channel, next`" before the end of the log is reached. For example, to read all logged data for channel 3A, first issue the following instructions:

```
getLog "In 1", first   getLog? "In 1"
```

Then, send the instruction "`getLog "In 1", next`" the indicated number of times.

Errors: if the channel does not exist, a run-time error occurs.

getOutput

Returns a single comma-separated string containing the current value of all channels. The values are always reported in the same order, which can be determined with the getOutput.names instruction.

Note that getOutput returns the most recent ADC reading, while getLog.last returns the most recent logged data point, which is usually the average of 1–100 ADC readings, depending on the logging interval.

getOutput.names

Returns a single comma-separated string containing the names of all channels.

getOutput.units

Returns a single comma-separated string containing the units of all channels.

group *1*

Changes the channel selection group. The group must be a number between 1 and 4, inclusive.

lasttouch

Indicates how many seconds have elapsed since the user last touched the touchscreen or pushed a button. If the user has not touched the touchscreen or a button since the CTC was turned on, the return value indicates how many seconds have elapsed since the CTC100 finished booting.

menu { Select Channels, Show Data, Program, Setup, Help, Output Enable }
menu *1*

Makes the CTC100 behave as if one of the front-panel buttons has been pressed. The argument can be the name of a front-panel button or a numeric value between 1 and 6, inclusive (1 for "Select Channels", 2 for "Show Data", etc). "Menu += 1" advances the CTC100 to the next menu; issuing the "Menu += 1" instruction while the Setup menu is showing brings up the Select Channels menu, not Help.

outputEnable { on, off }

Enables (`outputEnable = on`) or disables (`outputEnable = off`) all heater outputs and ±10V analog outputs. Issuing this instruction is the same as pressing the Output Enable button, but no pop-up window appears and the user doesn't have to confirm that the outputs should be enabled.

selectNone
Deselects all channels in all selection groups.

systemtime "*month day year hours:minutes*"
systemtime.dmy *day/month/year*
systemtime.hms *hours:minutes:seconds*
systemtime.mdy *month/day/year*
systemtime.ms *0*
systemtime.smh seconds minutes hours day month year
The `systemtime` instruction is similar to `System.Other.Time` and `System.Other.Date`, except that it 1) allows both time and date to be set or queried with a single instruction; 2) provides the time to the second instead of the minute; and 3) supports several different formats:

- "**Systemtime**" sets or reports the time and date in the same format as `System.Other.Time` and `System.Other.Date`, i.e. "Apr 7 2011 11:48 am".
- "**Systemtime.dmy**" sets the date in the format day/month/year or day-month-year.
- "**Systemtime.hms**" sets the time in the format hours:minutes:seconds, where hours is a value between 1 and 23.
- "**Systemtime.mdy**" sets the date in the format month/day/year or month-day-year.
- "**Systemtime.ms**" sets the time as the number of milliseconds since midnight on January 1, 1970 UTC.
- "**Systemtime.smh**" sets the time as six integers indicating the seconds, minutes, and hours since midnight, the day of the month, the number of the month, and the year.

IEEE 488.2 Instructions

The following instructions are intended for use with the GPIB interface, but can be issued through any of the CTC100's I/O ports. These instructions ignore the Verbose setting: a query instruction always returns the value only, while a set instruction always returns nothing. They also do not take the ".list" or ".help" suffixes.

Integer arguments can be supplied as hexadecimal values with the prefix "0x" (the number zero followed by a lower-case letter x); for example:

```
*ASE 0x10
```

sets the Alarm Status Register to hex 10 (decimal 16). Queries always return values in decimal format.

ASE *0
***ASE?**
Sets (or gets) the value of the Alarm Status Enable (ASE) register. If a bit of the ASR is set and the same bit of the ASE is also set, bit 0 of the Status Byte register is set.

***ASR?**
Returns the current value of the Alarm Status Register (ASR), and then clears the register. The Alarm Status Register is a 32-bit integer that indicates which alarms were triggered since the last time the `*ASR?` command was issued. Each of the CTC100's input channels is assigned a bit in the Alarm Status Register. When an alarm is tripped, the channel's bit in the Alarm Status Register is set. The bit is not cleared when the alarm turns off. Use the *`channel`*`.alarm.mask` instruction to determine which bit a particular channel is associated with.

***CLS**
Clear Status. Sets all status registers to zero, disabling all standard events.

***DMC "*Macro name*" "*Macro content*"**

Define Macro Command. Identical to the `define` instruction. Defines a macro, saving the text in the CTC100's local memory. Note that the macro content, like all instruction arguments, must be 1024 or fewer characters in length.

***EMC { 0, 1 }**
***EMC?**

Enable Macro Commands. Sending the command "`EMC 0`" disables macro expansion but does not affect macros that are already running. `*EMC?` queries whether macros are enabled and returns either 0 (macros disabled) or 1 (macros enabled). Since the state of `*EMC` does not persist when the CTC100 is rebooted, macros are always enabled when the CTC100 is turned on.

ESE *0
***ESE?**

Sets (or gets) the value of the Standard Event Status Enable (ESE) register. If a bit in the ESR register is set and the corresponding bit in the ESE register is also set, bit 5 of the Status Byte register is set.

***ESR?**

Returns the value of the Event Status Register (ESR), and then clears the register. The eight bits of the Event Status Register are assigned as follows:

Bit	Value	Description
7	128	Power On: set when the instrument is turned on.
6	64	User Request: set when the user touches the front panel or presses a menu key.
5	32	Command Error: set when an assembly error occurs in a GPIB macro.
4	16	Execution Error: set when a runtime error occurs in a GPIB macro.
3	8	Device Dependent Error: always 0.
2	4	Query Error: always 0.
1	2	Request Control: not used. always 0.
0	1	Operation Complete: set by the *OPC command.

***GMC? "*Macro name*"**

Get Macro Command. Prints out (to the I/O port that received the command) the text of a macro.

***IDN?**

Returns an identification string with the following format:

`Manufacturer, Model number, serial number, version`

where *serial number* is the instrument's five-digit serial number and *version* is the firmware version number.

***LMC?**

Learn Macro Command. Returns a comma-separated list containing the names of all available macros.

***NOP**

No Operation. Does nothing.

***OPC**

Operation Complete. Pauses the parent macro until all ongoing CTC100 operations have finished, then sets the Operation Complete bit in the Event Status register. `*OPC` is intended to indicate that all previous instructions in the macro have been completed.

*OPC is not generally required because most CTC100 instructions are fully processed before the next instruction in the macro is begun. The exceptions are PID autotuning (i.e., *channel*.tune.mode) and ramp-to-setpoint (*channel*.setpoint, if *channel*.ramp is nonzero). It's also possible to overlap instructions by sending a macro before the previous macro has finished.

*OPC waits for all autotuning processes to finish, regardless of whether they were started by the parent macro or not. It also waits for all setpoint ramps to finish, regardless of how those ramps were started. Finally, if two or more macros are running at the same time, *OPC waits until all other macros started by the source port have finished running before setting the Operation Complete bit. If the GPIB port starts two macros that contain *WAI?, *OPC?, or *OPC instructions, the result is a deadlock and both macros pause indefinitely. A deadlock does not occur if the two macros were received on different I/O ports or if one was started from the front panel.

While *OPC is waiting, new commands received over the source port are held in the input buffer. The commands are not processed until *OPC has finished waiting.

***OPC?**

Identical to *OPC, except that instead of setting the Operation Complete bit, *OPC? writes "1" to the I/O port once all tuning processes, setpoint ramps, and GPIB macros have finished.

***PHO**

Port holdoff. Prevents the I/O port that received this instruction from processing any incoming messages until the current macro (the macro that contains the *PHO instruction) has finished running. Once the current macro is finished, the I/O port returns to its normal state and *PHO has no further effect. Not a standard IEEE488.2 instruction.

***PMC**

Purge Macro Commands. Erases all locally-stored macros. Does not affect macros stored on USB memory devices.

***RST**

*RST is equivalent to turning the instrument off and back on again, except the Power On bit of the Event Status Register is not set. *RST has the following effects:

- Outputs are disabled (as if the "Output enable" button were pressed).
- All currently-running macros are stopped, regardless of whether the macros were started by the GPIB interface, another I/O port, or the Program screen.
- The instrument returns to the Select screen.
- Partially-received instructions on all I/O ports are cleared.
- All pending transmissions on all I/O ports are cancelled.
- The error queues for all I/O ports are cleared.
- The plot screen returns to showing the most recent data on autoscaled Y axes.
- The instrument automatically triggers at the rate set with the "A/D rate" control.
- All log data is stored in the CTC100's RAM is erased. Logs on USB devices are not affected. Unless data is being logged to a USB storage device, all graphs on the Plot screen are empty after a *RST command.

SRE *0
***SRE?**

Sets (or gets) the value of the Service Request Enable (SRE) register. If a bit of the Status Byte register is set and the same bit of SRE is also set, a GPIB Service Request is generated.

***STB?**

Returns the value of the Status Byte (STB) register. The 8 bits of the Status Byte are assigned as follows:

Bit	Value	Description
7	128	Unassigned. Always 0.
6	64	Requested Service: set when the CTC100 issues a GPIB service request.
5	32	Event Summary Bit: set when a bit is set in both the ESE and ESR registers.
4	16	Message Available: set when data is waiting to be read on the GPIB port.
3	8	Unassigned. Always 0.
2	4	Error Available: set when errors are waiting in the error queue. This bit will never be set unless System.COM.Verbose is set to Low.
1	2	Unassigned. Always 0.
0	1	Alarm: set when an alarm is triggered, if the bit that's set in the alarm's mask (see the *channel*`.alarm.mask` instruction) is also set in the ASE register.

***TRG**

Trigger command. Identical to the Group Execute Trigger (GET) bus message. Causes all channels to read their outputs. The amount of time that it takes to process this command is twice the value of the "A/D rate" setting.

After receiving a trigger command, the CTC100 stops automatically acquiring data. The inputs are only read, and PID feedback loops only update their outputs, when a `*TRG` command or a GET message is received. PID feedback outputs will not function properly unless the CTC receives `*TRG` commands or GET bus messages at the rate specified with the `System.Other.A/D rate` instruction.

To resume automatic sampling, set the A/D rate using `System.other.A/D rate`. For example,

```
"System.other.A/D rate" = 100
```

sets the CTC100 to automatically sample every 100 milliseconds.

***TST?**

Self-test. Returns a numeric error code that indicates whether data has been dropped and whether ADC conversions are occurring at the correct rate.

- First two digits: number of times ADC data has been dropped because of timing issues, or 30, whichever is smaller.
- Third digit: the lowest-numbered slot from which data was dropped; zero if no data has been dropped.
- Fourth and fifth digits: ADC conversion rate; 00 = 100% of the expected value (as set by `System.Other.A/D rate`); 01=101% of expected, 99=99%, etc. A value of 99 or 101% is usually not a problem and indicates that the line frequency is not exactly 50 or 60 Hz, or that the CTC100's clock is running slightly slow or fast. A value significantly different from 100% may indicate a problem with the circuit that synchronizes the ADC conversions to the AC line frequency.

The first 3 digits are cleared each time `*TST?` is issued.

For example, 13400 would indicate 13 read misses on slot 4 since the last time `*TST?` was issued, and that the system clock is operating normally.

***WAI**

Wait to Continue. Pauses the parent macro until other macros received on the same port, all PID tuning processes (regardless of how they were started), and all setpoint ramps have finished.

Identical to `*OPC`, but doesn't provide any explicit indication to the I/O port when the wait is complete.

Program menu

The `program.` prefix can be used but is not necessary for these instructions.

`abortMacro "`*`Macro content`*`"`
Defines an abort macro. The abort macro is run if the macro that defined it is aborted with an `abort` or `kill` instruction, or is stopped from Program or Setup screens. The abort macro is not run if the macro ends normally, if a `*RST` instruction is issued, if a `reset(running macros)` instruction is issued, or if a `reset(all)` instruction is issued. The abort macro also does not run if the macro is aborted before the `abortMacro` instruction is reached; therefore, this instruction should usually be at the beginning of the macro. The `abortMacro` instruction only affects the macro that called it, and has no effect on any other macros.

`clearerrors`
Erases all error messages for the port over which the instruction was transmitted. Also clears all messages from the System.Com.Errors window, regardless of which port generated them.

`cls`
Clears the "messages" window on the program screen, if the program is selected on the program screen's tab bar.
There is no `cls?` query

`define "`*`Macro name`*`", "`*`Macro content`*`"`
Saves a macro. The first argument is a file name under which to save the macro; the second argument is the content of the macro. Once a macro is saved, it can be called from another macro by issuing the file name like any standard instruction. The saved macro can also be started from the Program screen via the Load button or by touching the Progress window.
If a macro is already saved under the indicated name, the old macro is overwritten. If a file name conflicts with the name of a built-in instruction, the macro takes precedence if the command is issued with a capitalized first letter; the built-in instruction takes precedence if the command is issued with a lower-case first letter.
A single macro cannot both define a macro and call it. Calls to submacros are replaced with the full text of the submacro before the macro starts to run, but the define instruction doesn't actually define the macro until run time.
Example:

```
define Hello([print "Hello world!" pause 1 s]3)
```

The macro "Hello" can now be run by issuing the remote command:

```
Hello
```

Like all instruction arguments, the macro content must be 1024 or fewer characters in length. To define a macro longer than 1024 characters, instead of using the `define` instruction, save the macro on a USB memory device. First, use a text editor on a PC to compose the macro, and save it as a text file. The name of the text file should be the name of the macro plus the extension ".txt". Copy the text file to a folder named "macros" in the root directory of a USB memory device, and then plug the USB device into the CTC100. The macro should now be available for use as long as the USB device is plugged in.

Errors: If the macro name is longer than 32 characters, it is truncated to 32 characters. If the macro content is longer than 1024 characters, it is truncated to 1024 characters. "define" does not check the contents of the macro for syntax errors.

delete "*Macro name*"
delete.all

Deletes a saved macro. "Delete.all" deletes all macros saved in the CTC100's internal memory, but does not delete macros stored on attached USB devices. Deleting a macro has no effect on currently-running macros.

geterror

If verbose mode is set to "Low", error messages generated by remote commands are not transmitted over the remote interface. Instead, they are stored in an error buffer that can hold up to 20 messages. Each I/O port (USB, RS-232, etc) has its own error buffer. The "geterror" instruction returns the oldest message stored in the buffer, and then removes the message from the buffer. If the buffer is empty, "no errors" is returned.

Only errors generated by the port over which the geterror instruction was received are reported. If, for example, a geterror is received from the USB port, it only reports errors caused by messages that were received by the USB port.

Geterror does not remove messages from the System.Com.Errors window.

kill "*Macro name*"
kill.all

Stops all currently-running macros with the given runtime name. The runtime name is assigned with the name instruction and is not necessarily the same as the file name that a macro may be saved under.

kill.all stops all currently-running macros regardless of name or which port started the macro.

There is no kill? query.

name "*Macro name*"

Assigns a runtime name to the currently-running macro. A remote command or another macro can then use the kill instruction to stop the named macro. In addition, the name appears on the macro's tab in the Program screen. The name can be any alphanumeric string up to 32 characters long, and more than one macro can have the same name.

The name instruction does not affect the file name that a macro is defined under.

Errors**:** If the runtime name is more than 32 characters long, it is truncated to 32 characters.

pause *0.0* { ms, s, min, hr }

Pauses the program for the indicated amount of time. For example:

```
print hello  pause 2 s  print world!
```

prints the word "hello" on the program screen and also transmits "hello" to the serial port that the command was received from. After two seconds, the macro prints and sends "world!". The pause instruction only affects the macro that it's a part of. All other macros continue to run normally.

There is no pause? query.

popup "*Popup text*"
popup.close

Produces a popup window on the CTC100's screen with the supplied message. The message can be any alphanumeric string up to 128 characters long. If a help window or another popup message is already showing, it is closed and replaced with the new popup. The user has to press a menu button or the popup window's "ok" button to dismiss the window.

`Popup.close` closes any popup or help window currently visible, regardless of how the window was created.

If a popup window is visible on-screen, `popup?` returns the content of the popup window; otherwise, it returns the following text:

```
No popup window is present
```

portholdoff { on, off }

Prevents the IO port that received the parent macro from receiving any more macros until the parent macro has finished running or until a `portholdoff = off` instruction is encountered.

print "*message*"

Prints the indicated message. The message can be any alphanumeric string up to 128 characters long. If the program's tab is selected on the program screen, the message appears in the "Messages" area of the program screen. If the program was initiated from the remote interface, the message is also sent through the same remote interface that was used to transmit the program to the CTC.

There is no `print?` query.

redraw

Redraws the current top-level menu. This instruction closes all pop-up windows that may have been showing, including input windows, the Help window, windows produced with the "popup" instruction, the PID status window, COM port error and history windows, and warning message windows.

There is no `redraw?` query.

run "*Macro content*"

Starts a child macro that runs concurrently with the parent macro. The child macro runs invisibly in the background; any messages that it generates are not printed, and the macro has no effect on the `*OPC` and `*WAI` instructions. The parent macro continues to run while the child macro runs.

`run` should only be used when a child macro needs to run concurrently with the parent macro. Otherwise, macros should be called as subroutines, by including their name in the parent macro without the `run` instruction.

standby

Puts the CTC100 into standby mode, in which the outputs are turned off, data acquisition is paused, macros are paused, the front panel display and system fan are shut off, and the system does not respond to remote commands. The chassis cooling fan may switch on occasionally. Press the "Output Enable" key to exit standby. There is no remote command to exit standby mode.

waitForRamp

Pauses the macro until all PID setpoint ramps are complete. To wait for a particular channel's setpoint ramp to finish, use a `while` loop; for example:

```
while (Out1.PID.setpoint != Out1.PID.rampT) { pause 1 s }
```

waitForSample

Pauses the macro until an ADC conversion occurs.

waitForTune

Pauses the macro until all PID tuning processes are complete. To wait for a particular channel's tuning process to finish, use a `while` loop; for example:

```
while (Out1.Tune.Mode != Off) { pause 1 s }
```

System setup

system.com.baud { 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200, 250000 }

Sets the baud rate for the RS-232 and USB-to-PC interfaces. The interface always has no parity, 8 bits, and 1 stop bit.

system.com.flow ctl { None, RTS/CTS }

Enables or disables hardware flow control for the RS-232 and USB-to-PC interfaces.

system.com.verbose { Low, Medium, High }

Determines how the CTC100 replies when a remote instruction is processed.

Low: the CTC100 only replies when a query is processed.
Medium: messages are also produced whenever an error occurs.
High: messages are also sent whenever an instruction sets a parameter, and the messages include the names of parameters that are set or queried.

		Response to instruction…	
Verbose level	**2A?**	**xyz**	**2A = 37.47**
Low	37.4722	(none)	(none)
Medium	37.4722	Error: "xyz" is not a valid instruction	(none)
High	2A.Value = 37.4722	Error: "xyz" is not a valid instruction	2A.Value = 37.47

system.display.Bright { Min, 2, 3, 4, 5, 6, Max }

Sets the brightness of the front-panel display backlight.

system.display.Figures { 0, 1, 2, 3, 4, 5, 6 }

Sets the maximum number of figures that appear after the decimal point in the replies to remote queries of floating-point channel values, as well as on the front panel display. This setting does not affect how data is stored internally.

Each channel also has its own Figures setting that can override this global value.

system.display.Monitors { Show, Hide, Off }

If set to Show, the CTC100's internal monitor channels are displayed, logged, and reported by the getOutput command. These channels display printed circuit board (PCB) temperatures for the I/O cards as well as heater current, voltage, and resistance. If set to Hide, the monitor channels are not displayed on-screen but are still logged and still appear in the response to getOutput. If set to Off, the channels are completely disabled: they are not displayed, not logged, and don't appear in the response to getOutput.

system.display.Stats { Off, On }

Controls whether statistics are visible in the plot. If set to On and the plot type is single or multiple, the average and standard deviation for each channel for which statistics collection has been enabled (with *channel*.stats) is shown next to the channel name. Ponytail plots do not show the average and standard deviation, but instead show the offset of each channel, if stats display has been enabled.

"system.display.Monitors" { Show, Hide, Off }
If set to Show, the temperature of each I/O card that has a printed circuit board (PCB) temperature sensor is collected, stored, and shown on the Select screen. The system must be restarted before any changes are effective.

system.display.Type { Single, Multiple, Custom, Ponytail }
Controls the type of plot. On a Single plot, all selected channels appear on a single Y axis. "Multiple" generates a separate plot for each selected channel. "Custom" assigns each selected channel to a plot based on the channel's Plot setting. "Ponytail" produces a single plot with all selected channels, but each channel's trace is offset by its initial value. The offset is recalculated whenever the user scrolls or zooms the graph.

system.display.Units { °C, K, mK, °F, "" }
Sets the system units. Setting the units does not change previously-acquired data; that is, if a value of 22°C is recorded in the log and the units are then changed to °F, graphs and logs will appear to show that a value of 22°F was recorded. If the units are set to an empty string, thermocouple readings are shown in millivolts and RTD and thermistor readings are shown in ohms, even if the sensors have a custom calibration curve with declared units (see the "Custom calibration" section).

system.display.XLabels { None, Absolute, Elapsed }
Controls the type of label shown at the bottom of the plot. Absolute shows the time and date, while Elapsed shows the relative time in seconds, minutes, hours, or days.

system.display.XRange *0*
Sets the X range of the plot in milliseconds. Only the plot for the currently-selected group is affected.
Errors**:** a run-time error occurs if the argument is less than 10000 (10 seconds) or greater than 2592000000 (30 days).

system.IP.Address *0.0.0.0*
Sets the CTC100's IP address. The IP address should be in dotted-decimal notation, i.e. "172.16.0.0".
Errors: If part of the specified IP address is not in the correct format (i.e. contains a non-numeric character or a value that is not between 0 and 255), that portion of the IP address is set to zero. The IP address cannot be changed if `system.IP.DHCP` is set to `on`.

system.IP.DHCP { On, Off }
Enables or disables the Dynamic Host Configuration Protocol. If DHCP is enabled and a DHCP server is available on your network, the IP address is automatically set and cannot be changed manually.

system.IP.Port *23*
Sets the telnet port for Ethernet communications. Remote commands can be sent to the CTC via a telnet connection on the selected port. The port must be a value between 0 and 65535, inclusive, and should normally be set to either 23 (the default) or a value greater than 1024.

system.IP.Timeout *1440*
Sets the timeout period in minutes for Telnet connections. The CTC100 will close a Telnet connection if it doesn't receive any messages over the connection for this number of minutes. Any value between 0.1 and 65535 minutes can be used. The default is 1440 minutes (24 hours).

system.log.clear { yes, no }
Erases log files from the USB device.

- `system.log.clear USB` erases all logged data from internal memory and is mainly used to remove old data from the plot screen.
- `system.log.clear RAM` erases all log files from the current log folder on the USB device.
- `system.log.clear all` clears logged data from both USB and RAM.
- `system.log.clear?` always returns `no`.

system.log.folder "*Folder name*"
Determines which folder on the USB memory device receives the CTC100's logged data. If the folder does not exist, it is created. If the folder does exist and it already contains CTC100 logfiles, new data points are appended to the existing files.

system.log.interval { off, "0.1 s", "0.3 s", "1 s", "3 s", "10 s", "30 s", "1 min", "3 min", "10 min", "30 min", "1 hr" }
Sets the default log interval, which determines how often each channel's value is written to the log. Individual channels can override this value using the *channel*`.logging` instruction.

system.log.Log to { RAM, USB, None }
Set this parameter to "USB" to begin logging data to a USB memory device, if one is present. Set it to "RAM" to stop logging data to the USB device and store data in local memory, and to "None" to disable logging altogether. If set to "None", no data appears on the Plot screen.
Errors: if "USB" is selected and no USB storage device is present, this parameter automatically switches to "RAM".

system.log.USB { Auto, Manual }
If set to `Auto`, any time a memory device is plugged into one of the CTC100's USB ports, the CTC automatically begins logging to it. If set to `Manual`, the user must touch the USB logging dot in the upper-right corner of the screen to begin logging.

system.other.A/D rate { 20 ms, 40 ms, 60 ms, 80 ms, 100 ms, 120 ms, 140 ms, 160 ms, 180 ms, 200 ms, 220 ms, 240 ms, 260 ms, 280 ms, 300 ms, 400 ms, 500 ms, 600 ms, 700 ms, 800 ms, 900 ms, 1000 ms } (50 Hz line frequency)
system.other.A/D rate { 16.7 ms, 33.3, 50 ms, 66.7 ms, 83.3 ms, 100 ms, 150 ms, 200 ms, 250 ms, 350 ms, 400 ms, 500 ms, 600 ms, 700 ms, 800 ms, 900 ms, 1000 ms } (60 Hz line frequency)
system.other.A/D rate 0.0 (1 MHz trigger source)
Sets the A/D conversion time and determines how often PID feedback loops run. Different arguments are available depending on whether the line frequency is 50 or 60 Hz. If the "Trigger source" jumper on the CTC100's motherboard is moved to the "1 MHz clock" position, the A/D sampling can set to any value between 10 and 1000 ms.

system.other.fan { off, low, medium, high, max, auto }
Controls the system fan speed. Settings other than `auto` can reduce the accuracy of temperature measurements and cause the DC outputs to overheat and shut down.

system.other.date "*month day year*"
system.other.time "*hours:minutes*"
Sets the time and date. Note that setting the time and date can adversely affect the display of previously-acquired data. The time string should be in the form "10:57 am", while the date string should include the month, day, and year in that order, i.e. "Apr 7 2017" or "4/7/17".

system.other.reset { "Running macros", "Saved macros", "Front panel", Ports, "Port settings", Channels, Log, All }
Resets some or all user settings. The options have the following effects:

Running macros: stops all running macros. Has no effect on saved macros.
Saved macros: deletes all saved macros from the CTC100's internal memory, but does not delete macros from USB memory devices. Has no effect on running macros.
Display: Resets all `System.Display` settings to their factory defaults. Returns the front panel to the Select menu, de-selects all channels in all groups, and erases locally-stored log data (data on USB drives is not affected). Returns all plots to autoscaled X and Y with a 1 minute X range and changes the plot location of all channels to 1. If a `*TRG` remote command was previously received, re-enables automatic A/D conversions. Hides the monitor channels.
Ports: Closes all I/O ports and re-opens them. USB and Telnet connections will be lost. The port settings (baud rate, IP address, etc.) remain unchanged.
Port settings: Resets all I/O port settings to their factory defaults.
Channels: Resets the settings on the Channel menu for all channels to their factory defaults. Also sets the A/D rate to 100 ms.
Log: Resets the default log rate to 1 second, sets the log rate for each channel to the default (global) value, and enables automatic logging to USB. If a USB storage device is attached, erases log files in the root directory and begins logging to USB.
All: resets all of the above items.

system.other.Volume { off, 1, 2, 3, 4, 5, 6, 7, max }
Controls the volume of all tones and alarm sounds played through the CTC100's speaker.

Channel setup

Channel 0.0

To set the value of an output channel, send the name of the channel followed by the new value.

Attempting to set the value of an input channel, or attempting to set the value of an output channel when outputs are disabled, produces a run-time error.

Setting the value of an output channel that's under feedback control has no effect, but no error is generated.

<channel>.value? always returns the current value of the channel, regardless of whether the channel is an input or an output or whether outputs are enabled or disabled.

Examples:

```
Out1 = 1.0
```

Sets channel Out 1 to output 1 watt of power. Note that spaces in the channel name should be omitted.

```
2A?
```

Queries the output of channel 2A. If System.Com.Verbose is set to Low or Medium , the response is just a single value, such as:

```
37.4722
```

If System.Com.Verbose is set to High, the response is:

```
2A = 37.4722
```

If sensor 2A is not connected or is out of the range of its calibration data, the reported value is "NaN" (not a number).

For input channels and measured output channels, the current value reported by the CTC100 is the most recent ADC reading (after being calibrated and filtered). This value may be different than the most recently-logged point, which is the value that appears on the plot and in general corresponds to an average of several ADC readings.

Errors: attempting to set the value of an input channel, or of any channel when outputs are disabled, results in a "locked parameter" run-time error.

***Channel*.Average**

If statistics collection is enabled for the indicated channel (*channel*.Stats = on), this query returns the average over the most recent *n* A/D samples, where *n* is set with *channel*.Points.

***Channel*.Current { Forward, Reverse, AC, off }** (Temperature input channels only)

Selects the direction of the excitation current. In AC mode, the current direction is switched with each ADC reading and each measurement is the average of the two most recent readings, thereby eliminating errors caused by thermal EMFs. The excitation current can also be switched off entirely to control sensor self-heating; the sensor cannot be read while the excitation current is off.

***Channel*.d/dt { On, Off }** (Input channels only)

Enables or disables the derivative filter. If set to "On", the value of the channel is replaced with its rate of change expressed in units such as degrees/second, Watts/second, etc. Since the derivative is normally somewhat noisy, the lowpass filter should be enabled when the derivative filter is used.

Channels that have their derivative filter enabled can be used as inputs for PID feedback loops, in which case the feedback maintains a constant rate of temperature change rather than a constant absolute temperature.

***ChannelA*.Diff "*Channel B*"** (Input channels only)

Enables or disables the difference filter. If channel B is valid, the value of channel A is replaced with the difference between channel A and channel B (e.g., A–B). If channel B does not exist, the difference feature is disabled and channel A's output reverts to its normal value. Channel A must be an input.

Examples:

```
In1.diff(In 2)
```

Replaces the output of channel In 1 with the value (In 1 – In 2). Channel In 2 is unaffected.

```
In1.diff()
```

Removes the differencing function from channel In 1.

Channels that have their difference filter enabled can be used as the input for PID feedback loops, in which case the feedback maintains a constant temperature differential between two locations, rather than a constant absolute temperature.

Errors: if channel A is not an input, a "not a valid instruction" error is produced at assembly time.

***Channel*.Figures { 0, 1, 2, 3, 4, 5, 6, Default }**

Sets the maximum number of figures that appear after the decimal point in the replies to remote queries of floating-point channel values, as well as on the front panel display. This setting does not affect how data is stored internally.

If the Default option is selected, the number of figures is determined by the global System.Display.Figures setting.

***ChannelA*.Follow "*Channel B*"** (Virtual channels only)

This instruction is only available for virtual channels (channels V1, V2, and V3) with an IO type of Input. If Channel B is a valid channel name, the value of the virtual channel is updated with the value of Channel B each time an ADC conversion occurs. To exit follow mode, issue the *Channel*.Follow() instruction with an empty argument.

***Channel*.IOtype { Input, "Set out", "Meas out" }** (Output channels only)

This instruction, which is only available for output channels, determines the channel's direction. Not all options are available for every output channel.

If the IO type is set to "Input", the channel measures whatever value is present and does not produce an output; it becomes a high-impedance input.

If the IO type is "Set out" or "Meas out", the channel outputs either voltage, current, or power, depending on the *Channel*.Units setting. If "Set out" is selected, the CTC100 just displays whatever output was most recently requested by the PID feedback loop, remote interface, or front panel. If "Meas out" is selected, the displayed value is an ADC reading of the output.

Errors**:** If a channel's direction cannot be changed due to hardware limitations, attempting to set its IO type generates a run-time "locked parameter" error.

***Channel*.Logging { Off, "0.1 s", "0.3 s", "3 s", "10 s", "30 s", "1 min", "3 min", "10 min", "30 min", "1 hr", Default }**

Sets the log interval for this channel. "Default" makes this channel's log interval the same as the global default interval (see the System.Log.Interval instruction). "Off" disables logging for this channel.

***Channel*.Lopass { Off, "1 s", "3 s", "10 s", "30 s", "100 s", "300 s" }**

(Input channels only)

Sets the time constant for the lowpass filter. Input channels can be filtered with a 6th-order lowpass RC filter to remove noise. If enabled, the filter removes noise spikes and other transient signals that last for less than the selected time constant. The disadvantage is that the response speed of the sensor is also limited; that is, if your sensor can respond to temperature changes within 1 second and you select a 10 second lowpass filter, the sensor will now take 10 seconds to respond to temperature changes.

Errors: attempting to use the *Channel*.Lopass instruction on an output channel results in a compile-time "unrecognized instruction" error.

Channel*.LowLmt *0.0 (Output channels only)

Channel*.HiLmt *0.0 (Output channels only)

Constrains the minimum or maximum value of an output channel. These instructions can be used to prevent the PID loop, remote commands, or the front-panel controls from delivering excessive power to a heater.

The limits must be specified in the same units that the output is expressed in. The limits must normally be set again when the output units are changed, since the limits are not converted to the new units.

If the lower limit is greater than zero, it does not apply when the CTC100's outputs are disabled with the "Output Enable" key or the OutputEnable off instruction.

Errors: attempting to use one of these instructions on an input channel results in a compile-time "unrecognized instruction" error.

***Channel*.Name "*New channel name*"**

Changes the name of this channel. Once the name of a channel has been changed, the default channel name (In 1, In 2, Out 1, etc.) can no longer be used and all remote commands must address the channel by its new name.

To determine the current names of the CTC100's channels, use the `getOutput.names` instruction.

Errors: If a macro changes a channel's name, any attempts to address that channel again within the same macro will produce an "unrecognized instruction" error.

***Channel*.Off** (Output channels only)

Turns the selected channel off. The instruction cancels any active autotuning process, turns PID feedback off, and sets the channel's output to zero or the channel's lower limit (see the "*channel*.Low lmt" instruction), whichever is higher.

Errors: attempting to use this instruction on an input channel results in a compile-time "unrecognized instruction" error.

Channel*.PCB *0.0 (Temperature input channels only)

Sets the maximum allowable temperature of the input circuit (e.g., the printed circuit board or PCB). If the temperature of the circuit exceeds this value and `System.Other.Fan` is `Auto`, the CTC100 increases the fan speed to cool the air inside the chassis. The PCB temperature is always expressed in °C, regardless of the `System.Display.Units` setting.

***Channel*.Plot { 1, 2, 3, 4, 5, 6, 7, 8 }**

Indicates which plot the channel should appear in when the Plot screen is showing and the Custom plot tab is selected. Plot 1 is the topmost plot. If no channels are assigned to a plot, that plot does not appear.

Channel*.Points *0

Controls the maximum number of ADC readings used to calculate the average and standard deviation. The value refers to the number of ADC readings, not the number of log points. Each time the number of points is changed, the accumulated statistics are cleared.

Errors: if the number of points is not between 2 and 6000 inclusive, a run-time "parameter out of bounds" error occurs.

***Channel*.Polarity { 0, 1 }** (Digital I/O card relay channel only)

Sets the polarity of the relays. If both the polarity and the relay value are zero, the normally closed (NC) pins on the back panel are connected to the neighboring COM pins and the normally open (NO) pins are disconnected. If the polarity is 1, the reverse is true.

***Channel*.Range { 10ê, 30ê, 100ê, 300ê, 1kê, 3kê, 10kê, 30kê, 100kê, 300kê, 2.5V, Auto }** (Temperature input channels only)

Sets the sensor measurement range. The default range is Auto. In general, a lower range results in a larger excitation current, less noise, and more accurate measurements.

The CTC100 uses ASCII character 234 for the Ohms symbol. To type this character on a Windows computer, hold down the alt key and type 0234 on the number pad. On Windows computers the character appears as a letter "e" with a circumflex accent.

Errors**:** If this command is used with a channel that is not a sensor input channel, a "not a valid instruction" error is produced.

***Channel*.SD**

If statistics collection is enabled for this channel (with *channel*.Stats), this instruction returns the standard deviation over the most recent `points` A/D samples.

***Channel*.Selected { On, Off }**

Controls whether or not a channel is selected. Selected channels are added to the current selection group and appear on the Numeric, Plot, and Channel screens. Examples:

```
In1.selected(on)
```

adds channel In 1 to the current selection group, if it hasn't already been added.

```
In1.selected = off
```

removes channel In 1 from the current selection group.

***Channel*.Sensor { RTD, Thermistor, Diode, ROX }** (Temperature input channels only)

Selects the sensor type for a channel. Select `ROX` for a ruthenium oxide sensor, `Thermistor` for other NTC resistive sensors, `RTD` for PTC resistive sensors, and `Diode` for cryogenic diode sensors.

Some resistive cryogenic temperature sensors such as Rhodium-Iron, Germanium, and Carbon-Glass are not included in the list of available sensor types because they do not have standard calibration curves. To use these sensors, set the Sensor type to Thermistor or ROX and load a custom calibration table (see "Custom Calibration Tables" in the Introduction of this manual).

Changing the sensor type may affect how the CTC hardware acquires data from the sensor. In particular, if the sensor type is changed from `Thermistor` to `Diode`, the CTC acquires voltage instead of resistance readings and a special high-accuracy excitation current source is used. Also, the `RTD` setting results in larger excitation currents than the other settings.

The sensor type also affects which instructions are available in the *channel*`.Cal` menu. For example, if the sensor type is `RTD`, the *channel*`.cal.type` instruction offers a list of RTD types, and settings for the RTD's Callender-van Duzen coefficients appear in the *channel*`.cal` submenu.

***Channel*.Stats { on, off }**

Using the remote interface, the average and standard deviation of the most recent *n* ADC readings can be continuously calculated, where *n* is defined using the *channel*`.Points` instruction. The values can be displayed on the graph screen using the `System.Display.Stats` instruction or queried with the *channel*`.Average` and *channel*`.SD` instructions.

Channel`.Stats` turns sliding-window statistics collection on or off for a channel. When statistics collection is turned on, the average and standard deviation over the most recent *n* ADC readings are calculated at each ADC conversion and can be displayed on the single or multiple plot screens or queried via the Average and SD instructions. *n* is the smaller of 1) the number of ADC readings acquired since statistics collection was enabled; 2) the number defined with the *channel*`.Points` instruction; or 3) the number of ADC readings acquired since the Points instruction was last issued.

This command is only available through the remote interface.

***Channel*.Units { W, A, V }** (Heater output channels only)

By default, the outputs of the heater driver cards are measured in watts. Using the `Units` instruction, the output units of the output card can be changed to "A" (heater current) or "V" (heater voltage). Note that the `low lmt` and `hi lmt` settings are not automatically converted to the new units.

Channel*.Value *0.0

Identical to the Channel instruction. This command is only provided for consistency with the front-panel interface; the ".Value" is not necessary and can be omitted.

Channel.alarm submenu

All *channel*`.alarm` instructions can only be applied to input channels. Issuing any of the following instructions for an output channel results in an assembly-time "unrecognized instruction" error.

Channel*.alarm.lag *0

A non-zero lag prevents the alarm from triggering until the signal has continuously exceeded the alarm limits for the indicated number of seconds.

***Channel*.alarm.latch { No, Yes }**

Selecting `No` makes the channel's alarm momentary; `Yes` makes the alarm latching. A momentary alarm only sounds while the input signal exceeds the alarm limits; a latching alarm, once triggered, continues to sound until the status or mode is set to off.

Channel*.alarm.min *0.0
Channel*.alarm.max *0.0

These instructions set the alarm thresholds. The alarm is triggered whenever the signal exceeds these values. The thresholds are specified in the same units in which the channel value is displayed. If the channel's units are changed, the limits are not automatically updated.

***Channel*.alarm.mask?**

Returns a 32-bit integer that indicates which bit in the Alarm Status Register this alarm sets whenever it is tripped; for example, a mask value of 1 indicates that bit 0 is set; 2 indicates that bit 1 is set; 4 indicates that bit 2 is set; and so on. The Alarm Status Register is part of the GPIB status reporting system; see the IEEE488 commands section for more information.

Errors**:** attempting to change the value of the mask produces a run-time "locked parameter" error.

***Channel*.alarm.mode { Off, Level, Rate /s }**

Enables or disables the alarm. The alarm can be programmed to trigger either when the level of the signal or its rate of change exceeds the thresholds. The rate of change is calculated over two successive A/D conversions and is therefore susceptible to noise; if necessary, use the channel's lowpass filter to reduce noise.

***Channel*.alarm.mute { True, False }**

Turns off the alarm sound. Has no effect on the alarm relay. The alarm stays muted until the alarm condition disappears.

***Channel*.alarm.output { channel name }**

Associates an output channel with the alarm. This output is shut off whenever the alarm is triggered; that is, the output is set to zero and its feedback loop (if any) is disabled. Once the alarm status returns to "Off", the output returns to its previous value and the feedback loop resumes if it was running to begin with. This feature can protect equipment from the excessive temperatures that can occur if a PID feedback loop is poorly tuned.

To turn this feature off, issue the `alarm.output` command with an empty argument, i.e.:

```
In1.alarm.output()
```

***Channel*.alarm.relay { None, A, B, C, D }**

An alarm can trigger one of the CTC100's four relays. The `alarm.relay` instruction determines which, if any, of the four relays is triggered.

***Channel*.alarm.sound { None, 1 beep, 2 beeps, 3 beeps, 4 beeps }**

Controls which sound plays when the alarm goes off.

Channel.cal submenu

All `channel.cal` instructions are only available for input channels.

***Channel*.cal.A**
***Channel*.cal.B**
***Channel*.cal.C**
Channel*.cal.R0 *0.0

These instructions set custom calibration coefficients for RTD, thermistor, or diode inputs with a custom calibration type. See the description of the A, B, C, and R0 buttons on page 65 for more information.

Errors: If `cal.Type` is not set to `Custom`, attempting to set `cal.A`, `cal.B`, or `cal.C` produces a run-time "locked parameter" error. Attempting to use any of these instructions on a channel that is not a sensor input produces an assembly-time "unrecognized instruction" error.

Channel*.cal.Gain *0.0
Channel*.cal.Offset *0.0

These commands can be used to adjust a channel's calibration. The offset and gain are applied after the sensor signal is converted to temperature. These instructions provide an easy way to make adjustments to a sensor's calibration.

Errors: Attempting to set the offset or gain on a channel that is not an input produces an assembly-time "unrecognized instruction" error.

***Channel*.cal.Type { IEC751, US, Custom }** (RTD sensor type)
***Channel*.cal.Type { 100 ê, 300 ê, 1000 ê, 2252 ê, 3000 ê, 5000 ê, 6000 ê, 10000 ê B, 10000 ê H, 30 kê, 100 kê, 300 kê, 1 Mê, Custom }** (Thermistor sensor type)
***Channel*.cal.Type { DT-470, DT-670, Si410, Si415, Si430, Si435, Si440, Si540, S700, S800, S900, S950, Custom }** (Diode sensor type)
***Channel*.cal.Type { RX-102A, RX-103A, RX-202A, RO600, R400, R500 }** (ROX sensor type)
***Channel*.cal.Type { Custom, Standard }** (Channels with custom calibration tables)

Determines which calibration curve is used for a particular channel. The available arguments depend on the value of the *channel*`.Sensor` setting. See the description of the Type button on page 64 for more information.

The CTC100 uses ASCII character 234 for the Ohms symbol. To type this character on a Windows computer, hold down the alt key and type 0234 on the number pad. On Windows computers the character appears as a letter "e" with a circumflex accent.

Channel.PID submenu

All *channel*`.PID` instructions can only be used on output channels. Attempting to use a PID instruction on an input channel results in a "not a valid instruction" error.

By default, each PID loop has no assigned input channel. In this state, all PID settings are locked except for `channel.PID.input`. If a macro attempts to change the setpoint, the feedback gains, etc., a "locked setting" error is produced and the macro continues to run. An error message is only printed if Verbose is set to High.

Channel*.PID.D *0.0
Channel*.PID.I *0.0
Channel*.PID.P *0.0

These instructions set the PID gain factors. The PID equation is:

$$\text{Output}_t = Pe_t + 0.5IT((e_0 + e_1) + (e_1 + e_2) + \ldots (e_{t-2} + e_{t-1}) + (e_{t-1} + e_t)) + (D/T)(e_t - e_{t-1})$$

where P, I, and D are the derivative gains, e_t is the error (the difference between the setpoint and the PID input signal) at time t, and T is the ADC sampling time. Thus, larger values of P, I, or D

produce a faster feedback response. Increasing P or I tends to create oscillations, while increasing D reduces oscillations but adds noise. Negative values of P, I, and D should be used if the output drives a fan or other device that cools the sample.

Errors**:** Attempting to set P, I, or D when no PID input channel is selected produces a run-time "locked parameter" error. Attempting to set I or D when the PID mode is set to Follow also produces a run-time "locked parameter" error. Attempting to set P when the PID mode is set to Follow produces an assembly-time "Unrecognized instruction" error.

***Channel*.PID.Ffwd "*Channel name*"**

Selects a feedforward input channel. If a valid channel is selected and the PID mode is set to "on", the value the value of the feedforward channel is added to the PID output at each ADC conversion. To disable this feature, issue the *channel*.ffwd() instruction with an empty argument.

This feature can be used to implement feedforward control. The feedforward input should be some quantity with a known and predictable effect on the feedback system. The feedforward channel's cal.offset and cal.gain controls can be used to scale the feedforward effect.

Channel*.PID.Gain *0.0
Channel*.PID.ZeroPt *0.0

These instructions set the offset and gain for Follow mode. They are only available when the PID mode of *Channel* is set to Follow. In Follow mode, the value of an output channel follows the value of another channel with offset and gain applied:

$$\text{Output} = (\text{Input} - \text{Zero pt}) \times \text{Gain}$$

Note that "Input" can be either an input or an output channel. Also note that when the input is equal to the zero point, the output is zero.

Errors: Issuing a zero point or gain instruction when the PID mode is set to On or Off produces an assembly-time "Unrecognized instruction" error.

***Channel*.PID.Input "*Channel name*"**

Determines which temperature the PID feedback loop controls. If the channel name does not exist, any previously-selected input is deselected, leaving no PID input selected and the PID feedback disabled.

***Channel*.PID.Mode { Off, On, Follow }**

Enables or disables PID feedback. Turning feedback off freezes the output at its current value but does not set the output to zero. Setting the mode to "On" starts PID feedback using the current PID gains. In "Follow" mode, the output is continuously set to the value of the channel selected with the input instruction, adjusted by the Zero pt and Gain factors.

The input must be stable before either Step or Relay tuning is started. Furthermore, the output must be greater than half the step height before relay tuning is started. The best time to start a step response is when the system is first turned on at the beginning of the day, i.e. the heater is cold and its temperature stable. After the step response finishes, the feedback mode changes to manual and the heater ramps up to the setpoint. Once the temperature is stabilized at the setpoint, relay tuning can be used to produce more accurate PID parameters. When relay tuning is complete, the PID mode changes to manual.

Errors: Attempting to set the PID mode when no PID input channel is selected produces a run-time "locked parameter" error.

Channel*.PID.Ramp *0.0

Ramp rate. Determines the setpoint ramp rate in degrees per second. If the ramp rate is nonzero, whenever the feedback setpoint is changed the feedback will gradually ramp the

temperature to the new setpoint. If the ramp rate is set to zero, setpoint ramping is disabled and the CTC changes the temperature at the fastest possible rate.

Errors: Attempting to set the ramp rate when no PID input channel is selected produces a run-time "locked parameter" error.

Channel*.PID.RampT *0.0

Ramp temperature. The ramp temperature is an internally-generated setpoint for the PID feedback loop; it is the temperature that the CTC100 is trying to maintain at the present moment. If the feedback is not running, the ramp temperature always equals the sensor temperature, since the CTC100 has no control over the sensor temperature when the feedback is not running. When the feedback is started, the ramp temperature automatically increases or decreases at the ramp rate until it reaches the setpoint. This feature allows you to bring your system up to its operating temperature at a controlled rate. The actual temperature of your experimental system should ideally follow the ramp temperature, perhaps lagging a few seconds behind, depending on how quickly your system responds and how well the PID parameters have been tuned.

Once it reaches the setpoint, the ramp temperature remains at the setpoint as long as the feedback is running. If the setpoint is changed, the ramp temperature automatically increases or decreases at the ramp rate until it reaches the setpoint. If the feedback is disabled, the ramp temperature immediately begins to track the sensor temperature.

To start a temperature ramp, enable the feedback, set the ramp rate, and then change *Channel*.PID.Setpoint to the desired end point of the ramp. In general, the ramp temperature should not be directly set by the user, except perhaps as a way to cancel a ramp; for example,

```
Out1.PID.RampT = #Out1.PID.setpoint
```

tells the CTC100 to stop gradually ramping the temperature and instead proceed as quickly as possible to the setpoint. On the other hand,

```
Out1.PID.setpoint = #Out1.PID.RampT
```

stops ramping by freezing the temperature at its current value.

The following line can be used to pause a macro until the ramp is complete:

```
while (Out1.PID.RampT != Out1.PID.setpoint){ pause 1 s }
```

Errors**:** Attempting to set the ramp temperature when no PID input channel is selected produces a run-time "locked parameter" error.

Channel*.PID.Setpoint *0.0

Determines the PID setpoint. The PID loop attempts to keep the input at this value by changing the output.

Errors**:** Attempting to set the setpoint when no PID input channel is selected produces a run-time "locked parameter" error. Issuing a setpoint instruction when the PID mode is set to Follow produces an assembly-time "Unrecognized instruction" error.

Channel*.PID.TMin *0.0

Each of the eight PID memory locations can be assigned a temperature range for zoned feedback. If zoned feedback is enabled by setting *Channel*.PID.Zone to Auto, any given memory location is automatically recalled whenever the PID input temperature enters its temperature range.

The *Channel*.PID.TMin instruction determines that temperature range. It assigns a lower temperature bound to whichever memory location is currently selected. There is no TMax instruction; the upper end of the temperature range is the next-highest TMin value in the memory table.

Errors: Attempting to set the minimum zone temperature when no PID input channel is selected produces a run-time "locked parameter" error.

***Channel*.PID.Zone { 1, 2, 3, 4, 5, 6, 7, 8, Auto }**

Sets the PID temperature zone. A set of PID gains, an input sensor, and a minimum temperature can be assigned to each of eight temperature zones. If the zone is set to `Auto`, a set of stored feedback parameters is automatically recalled whenever the ramp temperature (*Channel*`.PID.RampT`, which is usually the same as the feedback setpoint) enters one of the temperature zones. Any changes to P, I, D, or the input sensor are automatically reflected in the zone definition for the current zone.

All eight PID zones can be viewed as a table on the front panel; see the description of the Zone button on page 66. If you don't already know the feedback parameters to be loaded into the table, it's usually easier to use the front panel rather than remote commands to determine the correct parameters and load them into the table. However, if the feedback parameters are already known, they can be loaded into the table with a macro such as the following:

```
Out1.PID.Zone 1       ' select the first line of the table
                      '   and disable zoned feedback
Out1.PID.Tmin 25      ' fill in the first line of the table…
Out1.PID.P 1.5
Out1.PID.I 0.13
Out1.PID.D 0.04
Out1.PID.Input In1

Out1.PID.Zone 2       ' select the second line of the table
Out1.PID.Tmin 35
Out1.PID.P 0.75
Out1.PID.I 0.05
Out1.PID.D 0.03
Out1.PID.Input In3

Out1.PID.Zone 3       ' select the third line of the table
Out1.PID.Tmin 1000    ' ensure that this line is never used

Out1.PID.Zone Auto    ' enable zoned feedback
```

Errors: Attempting to change the zone when no PID input channel is selected produces a run-time "locked parameter" error.

Channel.Tune submenu

See the "Automatic PID Tuning" section of this manual for more information on using these instructions.

Channel*.Tune.Lag *0.0
Channel*.Tune.StepY *0.0

These parameters provide the PID autotuners with initial guesses of the system's response magnitude and time. *Channel*`.Tune.StepY` controls the height of the step response or relay disturbance, while *Channel*`.Tune.Lag` determines how long the tuner waits before it first evaluates the effect of the disturbance. If either value is too small, the autotuning algorithm will be susceptible to noise. The Y step size should be high enough to produce a temperature rise of several degrees, and the lag should be long enough for the temperature to rise noticeably.

Errors: Attempting to set step Y or Lag when no PID input channel is selected results in a run-time "locked parameter" error.

***Channel*.Tune.Mode { Off, Auto, Step, Relay }**

Starts or stops PID autotuning. `Step` starts the step response tuning algorithm; `Relay` starts the relay tuning algorithm. In `Auto` mode, the CTC100 begins a step response if the PID output is less than half of *`Channel`*`.Tune.StepY`; otherwise it begins the relay tuning procedure. `Off` cancels any PID autotuning that's currently in progress.

***Channel*.Tune.Type { Cons, Moderate, Aggr, Auto }**

Determines how the PID tuner sets the feedback gains. `Cons` results in slow feedback response rates with little overshoot of the setpoint. `Aggr` results in fast response, but much more overshoot. `Moderate` produces intermediate results. `Auto` uses the conservative setting with the step response tuner and the aggressive setting with the relay tuner.

Error codes

The CTC100 reports an error code if there's an error in a macro and `System.COM.Verbose` is set to `High`. The error code appears in the error message sent to the communications port that initiated the macro, and is accompanied by an explanation of the error and which instruction caused the error. For example, if you send the word `Hello` to the CTC100 over the RS-232 port when `System.COM.Verbose` is set to `High`, the CTC100 replies with the following message:

```
Error: "hello" is not a valid instruction (assembly error -113)
```

-100 – -199: assembly errors

Assembly errors are produced before the macro starts to run. If an assembly error is reported, the macro was cancelled before it began running.

-102: Empty instruction. The instruction consisted of two quotes or parentheses in a row, with no text in between.
-113: Invalid instruction. The instruction was not recognized.
-109: Multiple argument error. Two or more arguments were expected, and the arguments provided did not conform to the types of arguments expected.
-121: Numeric argument error. A numeric value was expected, but a non-numeric argument, or no argument, was provided.
-158: List argument error. The argument must be chosen from a list of possible values, but the argument provided is not on the list.
-180: Too many macros. The maximum number of macros (10 from any one I/O port or 50 from all sources combined) is already running. At least one macro must finish before any new macros can be started.
-185: Excessive recursion. A macro may call another macro, which can call another macro, and so on, but only 6 levels of recursion are allowed. This error is always generated if a macro calls itself.
-186: Assembled macro exceeds 4096 lines. When a macro is assembled, all of its subroutine calls are expanded into their component instructions (thus, the assembled macro only contains native instructions). The assembled macro cannot be longer than 4096 lines.

-200 – -299: runtime errors

Runtime errors are produced after the macro starts running. After a runtime error occurs, the macro continues to run.

-221: Locked parameter. The parameter is locked (on the front panel, the control is greyed out) and cannot be changed.

-222: Argument out of range. The argument was a numeric value and was too large or too small.
-224: Bad argument. The argument must be chosen from a list of possible values, and the argument provided is not on the list.
-225: Out of memory. At attempt was made to define a macro, but ten macros are already defined in RAM.

Startup macros

To make a macro run automatically whenever the CTC100 boots up, enclose the macro in the following statement:

```
define Startup(macro)
```

where *macro* is the text of the macro to be run. Send this line over a serial port or run it from a USB stick. The macro doesn't run when the define statement is issued, but subsequently, it will run each time the CTC100 boots up. Note that the macro must be less than 1024 characters long and must not call any macros stored on USB devices.

For example, the following remote command defines a startup macro that displays a message each time the CTC boots up:

```
define Startup(popup "Power has cycled")
```

If the Startup macro contains any errors, the macro won't run and no error messages will appear. Therefore, it's a good idea to test startup macros by running them normally, i.e. by sending the remote command `Startup` or using the Program screen to start the macro.

Similar functionality can be obtained by saving a macro as a file called "autorun.txt" in the root directory (not the macros directory) of a USB device and keeping the device plugged in to the CTC100. This macro will automatically run each time the USB device is plugged into the CTC100 and each time the CTC100 is turned on with the USB device plugged in. Autorun macros are not limited to 1024 characters and can call other macros stored on the same USB device.

Sample macros

This section presents several sample macros to illustrate the capabilities of the CTC100. The macros are shown on multiple lines for clarity and can only be run as shown if they are input with a USB memory stick, as follows:

1. Enter the macro into a text editor such as Notepad. Save the macro as an ASCII text file with the extension ".txt". Copy the file into a directory named "macros" on a USB memory stick or hard drive.
2. Plug the USB stick or drive into the CTC100.
3. Press the CTC100's "System" key. A button with the macro's file name should appear in the "Macros" column. Touch the button to start running the macro. The button remains highlighted as long as the macro is running. Touch the highlighted button to stop the macro.

If the sample macros are sent to the CTC100 via the RS-232, GPIB, USB device, or Ethernet port, each macro must be formatted as a single line with the comments removed, otherwise each line will be treated as a separate macro, and the lines will all run at the same time instead of sequentially.

Temperature profiles

The following macro ramps the temperature controlled by channel Out 1 to 100°C at a rate of 1°C/second. Once the ramp is complete, the system pauses for 1 minute at 100°C and then ramps the temperature down to 80°C. After another 1 minute pause, the system is allowed to cool back to room temperature by changing the feedback setpoint to 0 degrees (without ramping):

```
Out1.PID.ramp = 1         ' set the ramp rate to 1 degree/s
Out1.PID.setpoint = 100   ' start a ramp to 100 degrees
waitForRamp               ' wait for the ramp to finish
pause 1 min               ' wait for 1 minute
Out1.PID.setpoint = 80
waitForRamp
pause 1 min
Out1.PID.ramp = 0         ' disable ramping
Out1.PID.setpoint = 0
```

The equals signs are optional and are shown for clarity. If this macro is entered from the "Program" screen, the "channel." and "program." prefixes must be included:

```
channel.Out1.PID.ramp 1
channel.Out1.PID.setpoint 100
program.waitForRamp
program.pause 1 min
channel.Out1.PID.setpoint 80
program.waitForRamp
program.pause 1 min
channel.Out1.PID.ramp 0
channel.Out1.PID.setpoint 0
```

This is the easiest way to run a temperature profile, but it doesn't work if two or more PID feedback loops are ramping at the same time. The problem is that `waitForRamp` actually waits for all setpoint ramps to end, whether or not they were started by the macro.

A more elaborate version eliminates this issue by comparing the current value of the ramp (`Out1.PID.rampT`) with the endpoint of the ramp (`Out1.PID.setpoint`):

```
Out1.PID.ramp = 1
Out1.PID.setpoint = 100
while (Out1.PID.rampT!=Out1.PID.setpoint) { pause 1 s }
pause 1 min
Out1.PID.setpoint = 80
while (Out1.PID.rampT!=Out1.PID.setpoint) { pause 1 s }
pause 1 min
Out1.PID.ramp = 0
Out1.PID.setpoint = 0
```

A third option is to wait for the measured temperature to reach the endpoint of the ramp:

```
Out1.PID.ramp = 1
Out1.PID.setpoint = 100
while (In1 < 99.5 || In1 > 100.5) { pause 1 s }
pause 1 min
Out1.PID.setpoint = 80
while (In1 > 80) { pause 1 s }
pause 1 min
Out1.PID.ramp = 0
Out1.PID.setpoint = 0
```

The `pause 1 s` instructions aren't strictly necessary, but reduce the load on the CPU.

Control a feedback setpoint with an analog input

The following macro controls the setpoint of channel Out 1 based on the voltage at analog input 1. The macro converts the ±10V analog voltage to a temperature between 0 and 100 degrees; another way to scale the analog voltage would be to use channel 5A's offset and gain controls. The contents of the macro are placed in an infinite-repeat block (square brackets followed by a negative number). The `waitforSample` ensures that the block doesn't run any more often than necessary (i.e., once per ADC sample).

```
[
  waitforSample
  if (Out1.PID.Mode==on) {
    #x = #AIO1
    #x+=10
    #x*=5          ' note: spaces are not allowed before the '*'
    Out1.PID.setpoint = #x
    }
 ]-1
```

The setpoint is only updated when the feedback is turned on. Although not necessary, this precaution keeps the macro from generating run-time errors when the setpoint is locked.

Show channels with tripped alarms on the Numeric screen

This macro turns selection group 1 into a display of channels with tripped alarms. Once per second, if group 1 is selected, all channels whose alarm mode is "on" are selected; all other channels are deselected. The macro is best used with the Numeric screen visible, but also works with the Select or Plot screens.

```
[
  if (group==1) { selectAlarmed }
  pause 1 s
]-1
```

Show the PID setpoint in a virtual channel

Virtual input channels have a "follow" control that can be used to make the channel echo the value of any other channel. With a macro, the virtual channel can also be made to echo any CTC100 parameter — not just channel values. The following macro uses a virtual channel to echo a feedback setpoint. This macro makes it possible, for example, to graph the setpoint on the "Plot" screen alongside other variables, or (using the "Diff" button) to graph the difference between the setpoint and the actual temperature:

```
[waitForSample  V1=#Out1.PID.rampT]-1
```

Each time an ADC conversion occurs, this macro sets channel V1 to the setpoint of channel "Out 1" (if the setpoint is ramping toward a given value, Out1.PID.rampT returns the current value of the ramp, while Out1.PID.setpoint returns the endpoint of the ramp). Because the macro is contained within a `[…]-1` statement it repeats indefinitely, running as a background task.

Using the "diff" function of channel V1, the difference between the actual temperature and the feedback setpoint can be plotted. This can be helpful for monitoring the accuracy of setpoint ramps.

Linearizing outputs when interfacing with external power supplies

If you need more heater power than the CTC100 can deliver, you can use the CTC100's analog outputs to control a programmable power supply. Since the analog input on programmable power supplies usually sets the voltage or current supplied to the heater, the temperature rise of the heater roughly depends on the square of the CTC100's output. For example, if a 1 V output increases the temperature by 1 degree over ambient, a 2 V output would increase the temperature by about 4 degrees. As a result, you may notice sluggish response at low output values and/or feedback oscillations at high outputs. Feedback performance can be made more consistent by linearizing the PID output vs. temperature response curve.

One way to linearize the PID output is to apply a custom calibration table to the output channel (see page 24 for a description of how to make and upload calibration tables). In this case, the calibration table is a file containing comma-separated data in the format "X1, Y1, X2, Y2, …", where Xn is the analog output, in volts, to be produced when the PID algorithm requests output Yn. To produce such a table experimentally, set the analog output to a series of different voltages. At each analog IO voltage Xn, measure the temperature Yn at which the system stabilizes.

Another way to linearize the PID output is by using a macro to apply a simple equation to the PID output. Use a virtual channel such as channel V1 to host the PID feedback loop. Set the IO type of channel V1 to "Meas out", then configure channel V1's PID loop with the appropriate input sensor and temperature setpoint. Set the IO type of analog I/O channel A to "Set out" or "Meas out" and disable channel AIO A's PID feedback loop. Next, run the following macro, which sets AIO A to the square root of channel V1 each time an ADC conversion occurs.

```
[
  waitForSample
  #x = #V1
  #x^=0.5
  AIOA = #x
]-1
```

Control instrument functions with the digital IO lines

This macro enables the feedback for channel "Out 1" whenever bit 0 of the digital I/O is high, and disables the feedback whenever the bit is low. The program runs indefinitely.

```
' start with the feedback turned off
Out1.PID.mode = off

' this loop repeats indefinitely
while (1) {
  ' wait for DIO bit 0 to go high, then turn feedback on
  while (DIO & 0x01 = 0) { pause 0.25 s }
  Out1.PID.mode = manual

  ' wait for DIO bit 0 to go low, then turn feedback off
  while (DIO & 0x01 = 1) { pause 0.25 s }
  Out1.PID.mode = off
}
```

The next macro lets DIO bit 1 control which temperature sensor serves as the input for channel Out 1's feedback loop:

```
[
  #x = DIO
  #x &= 2

  ' if bit 1 is clear and the PID input channel is not In 1,
  ' set the PID input channel to In 1
  if (#x==0 && Out1.PID.input!=$In 1) { Out1.PID.input=(In 1) }

  ' if bit 1 is set and the PID input channel is not In 2,
  ' set the PID input channel to In 2
  if (#x==2 && Out1.PID.input!=$In 2) { Out1.PID.input=(In 2) }

  pause 0.25 s
]-1
```

Within an `if` or `while` statement, the "$" prefix prevents the following text from being treated as a query. If the $ prefix were left out, the statement would attempt to compare the name of the PID input channel to the value of channel In 1, rather than to the string "In 1".

Drive a solid state relay with the digital IO lines

In some high-power applications, the current to a heating or cooling unit is provided by an external power supply and modulated with an external solid state relay (SSR). To modulate the heater or cooler power and obtain accurate temperature control, a variable duty cycle square wave, similar to pulse width modulation but typically with a cycle time of several seconds, is required from the CTC100. For example, to supply half of the maximum power to the heater, the CTC100 would need to turn the relay on for 5 seconds, off for 5 seconds, on for 5 seconds, etc.

The following procedure transforms the output of a PID feedback loop into a variable duty cycle square wave that can be output on the CTC100's digital IO lines and used to drive a solid state relay. The macro works well as long as a period of about 10 seconds or longer and a resolution of 0.1 seconds is acceptable. If a much shorter period or greater resolution is needed, it would be better to fabricate an external analog-to-PWM circuit and drive it with an analog I/O channel.

First, make channel V1 the feedback output, and make it produce a value between 0 and 100. To do this, select channel V1 and set the following parameters:

- Low lmt: 0
- Hi lmt: 100
- IO type: Meas out
- PID input: select the temperature channel to be controlled
- PID mode: set to "off" for now

• PID setpoint: set to the desired temperature

Next, select channel DIO and set the following parameters:

• IO type: set out
• PID input: should be blank; or, the PID mode should be off.

Now run the following macro by sending it over a serial port (in which case it all has to be on one line) or by copying it onto a USB stick (save it as a .txt file in a directory named "macros"):

```
[
 waitForSample
 #d = 0
 if (#V1>#t) { #d = 1 }
 DIO = #d
 #t += 1
 if (#t>100) { #t = 0 }
]-1
```

To test the macro, set V1's value to 50 and plot channel DIO. You should see a square wave with a duty cycle of 50% and a period of 10 seconds: high for 5 seconds, low for 5 seconds, high for 5 seconds, etc. Reduce V1 to 25 and the duty cycle should go to 25%.

Before the feedback can be used, the PID gain factors will need to be set by using the automatic tuning feature on channel V1. If tuning is successful, the feedback should now operate normally.

If more than one feedback loop is required, set up channels V2 and/or V3 as described for channel V1, and add these lines after the { #d = 1 } statement:

```
if (#V2>#t) { #d+=2 }
if (#V3>#t) { #d+=4 }
```

The macro can automatically run every time the CTC100 is turned on; just send the command "define Startup (…)", replacing the … with the macro contents.

PC applications

SRS offers a package of PC applications for displaying and reformatting CTC100 log files. The applications can be downloaded free of charge from the SRS website at www.thinksrs.com; click on Downloads > Software. Once unzipped, the applications can be run by double-clicking the .exe icon or dragging CTC100 log files to the .exe icon. It is not necessary to run an installation program.

PTCFileConverter

PTCFileConverter is a Windows utility that modifies log files. It can:

- Make timestamps easier to read by changing them from milliseconds since 1970 to date and time, or elapsed seconds, minutes, hours, or days.
- Change the format from CSV to tab or space-separated text or an HTML table.
- Make large files more manageable by increasing the amount of time between data points (resampling).
- Combine two or more consecutive log files into a single file.

Double-click the program icon to open the setup window, which has six input fields and two buttons. Once the fields have been filled in, files can be converted by clicking the "Start" button. Or, click the "Close" button and then drag one or more files onto the PTCFileConverter icon. In this drag-and-drop mode, the setup window is not displayed and the files are immediately converted using the most recently-saved settings (the "Input folder or file" setting is ignored).

Input folder or file

Select the log file or files that you'd like to convert. If you select a directory, when the "Start" button is pressed PTCFileConverter will convert all log files in the directory (but not in its subdirectories) and attempt to combine them into a single output file.

Files that do not contain any data (empty log files or files that are not log files) are ignored and do not appear in the output file.

Files to be combined must all have identical headers.

Output file

If a Text or HTML output format is selected, this field determines the name of the output file. If a directory isn't specified, it will be the same as the input file. If an extension isn't specified, ".txt", ".csv", or ".html" will be appended to the file name when the file is saved.

If Binary output format is selected, this field determines the output folder. The output files are saved to this folder and have the same name as the input files. The output folder must be different from the input folder.

Output format

The converted data can be saved as a text file, an HTML file, or a binary file. In all cases, the output is a table with a timestamp column plus one column per channel, and one line per sampling period. Text files can be saved with a tab, comma, or space between the entries on each line.

HTML files are useful because they are easily viewed and are also easily imported into many application programs; however, this format should only be used for short datasets (less than a thousand points) because HTML browsers are very slow when displaying large tables.

The "Binary" output format is included for backward compatibility with older CTC100 log files.

Timestamp

When converting data to a text or HTML file, this setting determines how the time of each data point is recorded:

- "Date and Time" records the time to the nearest second in the format "March 26, 2018 6:43:11 PM".
- "Milliseconds since 1970" is a single 64-bit decimal value that indicates how many milliseconds have elapsed since midnight on January 1, 1970.

- "Elapsed seconds", "Elapsed minutes", "Elapsed hours", and "Elapsed days" record the time as a single floating-point value that indicates how much time has elapsed since the first point in the log.

Resample

Check the "Resample" box to downsample or upsample log files. If "Resample" is checked, PTCFileConverter either averages points together or duplicates points so that the log rate of the output file is the value in the "Resample period" field. For example, if the input log has one point per second and the "Resample Period" is set to 10 seconds, checking the "Resample" box produces an output file in which each point is the average of 10 input points.

The resample feature is useful for reducing the number of data points in very large logs. Also, different CTC channels can be logged at different intervals and it's often useful to resample the data so that data points appear at the same interval for all channels.

Resample period (seconds)

If the "Resample" control is checked, the "Resample period" field controls how many seconds each line of data in the output file represents. If the "Resample" control is not checked, the "Resample period" field has no effect.

Start

Press the Start button to begin the conversion.

Close

Press the Close button to save all settings and close PTCFileConverter.

Clicking the "X" button in the upper-right corner of the window closes the program without saving any settings.

FileGrapher

FileGrapher is a Windows utility that plots CTC100 log files.

To plot a file, either drag a log file onto the File Grapher icon or double-click the FileGrapher icon and then select "Open" from the "File" menu. Once the file has been plotted, a file selection window appears that shows all of the CTC100 files in the same directory as the plotted file.

Click on a file in the file selection window to plot it. Shift-click or Control-click to plot two or more files at the same time. The first file listed in the selection window always appears as a black trace; the second file is always red, the third blue, the fourth orange.

To zoom in on a graph, draw a rectangle around the area that you'd like to zoom in on. To zoom out to the previous zoom area, double-click on the graph. Triple-click on the graph to show all data.

When FileGrapher opens a file, it reads the entire file into a buffer in RAM. Very large files may not fit in the program's memory or may take a long time to load and display. If this occurs, use PTCFileConverter to downsample the file before opening it with FileGrapher.

File menu

Open

Opens a directory for plotting. All log files in the directory are shown in the selection window and the selected file is plotted. All unsaved changes to data in the old directory are lost.

Close

Closes the selected directory. All unsaved changes to data are lost and the selection window closes.

Save GIF

Saves the graph as a GIF file.

Save data

Saves a trace as a text file or a binary (.PTC) file. See the PTCFileConverter documentation for more information on data saving options.

Exit:

Quits the program.

Edit menu

Items in the Edit menu may affect how data buffers are graphed, but do not affect the contents of the buffers.

Plot options

Opens a window that controls the appearance of the graph. Click "Apply" to update the graph with the new settings; "OK" to update the graph and close the window; "Cancel" to undo all changes since the last time the graph was updated and close the window.

- **Automatically scale X**: if checked, the graph is automatically scaled to show the full time span of the data.
- **X minimum**: if "Automatically scale X" is checked, this box indicates the time at the left-hand edge of the graph; any values entered here by the user are ignored. If "Automatically

scale X" is not checked, the time entered here determines the time at the left-hand edge of the graph.
- **X maximum**: if "Automatically scale X" is checked, this box indicates the time at the right-hand edge of the graph; any values entered here by the user are ignored. If "Automatically scale X" is not checked, the time entered here determines the time at the right-hand edge of the graph.
- **Automatically scale Y**: if checked, the graph is automatically scaled to show the full vertical span of the data. The graph is automatically rescaled as necessary whenever the data is modified.
- **Y minimum**: if "Automatically scale Y" is checked, this box indicates the lower limit of the graph; any values entered here by the user are ignored. If "Automatically scale Y" is not checked, the value entered here determines the lower limit of the graph.
- **Y maximum**: if "Automatically scale Y" is checked, this box indicates the upper limit of the graph; any values entered here by the user are ignored. If "Automatically scale Y" is not checked, the value entered here determines the upper limit of the graph.
- **Suppress X axis label**: if checked, the graph's X axis is not labeled. This option is intended for use when two or more graphs with the same X range are stacked on top of each other.
- **Number of X divisions**: controls the number of vertical gridlines. The value entered is approximate; the program may draw slightly more or fewer gridlines in order to put the gridlines on round time values.
- **Number of Y divisions**: controls the number of horizontal gridlines. The value entered is approximate; the program may draw slightly more or fewer gridlines in order to put the gridlines on round Y values.
- **Y axis label**: The text entered here is displayed to the left of the graph.
- **Annotation**: The text entered here is displayed inside the plot area. Enter the string "<names>" to display a list of the plotted files, each shown in the color in which it is plotted.
- **Annotation position**: Controls where on the plot the annotation appears.

The following options appear when the "More options" button is clicked:
- **Subtract baseline**: if checked, "baseline" data is subtracted from every plot in the graph. To set the baseline data, display a graph and select "Set as baseline" from the Edit menu.
- **Subtract average**: if checked, each trace is offset such that its average value is 0.
- **Y offset between traces**: can be used to separate traces that are on top of each other. One times this constant is added to trace 2; two times this constant is added to trace 3; three times this constant is added to trace 4; and so on.
- **Colors**: the colors used in the graph can be defined in this section. Each color is a set of three numbers between 0 and 255 for red, green, and blue brightness. Enter "255,255,255" (without the quotes) for white and "0,0,0" for black.
- **Show tick mark**s: some data files can include "tick marks" to mark events. If the "show tick marks" box is checked, the tick marks are shown as small spikes in the graph.
- **Antialias**: if checked, the plot is drawn with antialiased lines. This improves the appearance of the graph but also significantly increases the amount of time that it takes to draw the graph.
- **Axis linewidth**: the width of the box surrounding the plot, in pixels.
- **Grid linewidth**: the width of the plot gridlines, in pixels.
- **Plot linewidth**: the width of the plot traces, in pixels. Values other than 1 may significantly increase the amount of tie that it takes to draw the graph.

Show statistics
Shows information such as the average, minimum, and maximum values for all data within the graph's X range. Only information for the buffer plotted in black is shown.

Linear regression
The linear regression feature can be used to determine how much one temperature sensor is miscalibrated compared to another. You are asked to choose an X and a Y buffer (the log files for two different temperature sensors). The software then determines the average offset and gain of the X buffer relative to the Y buffer. Check the "Apply equation to X buffer" box to multiply the X buffer by the gain factor and then add the offset.

Command line
Opens a File Grapher command line window. The commands described in the table below can be typed into the command line. Sequences of commands can be stored as macros and then recalled either from the command line or the Special menu.

Align X axes
Sets the X axis range of all graphs to be equal to the X axis range of the selected graph.

Add graph
Adds another graph to the display. When more than one graph is displayed, you can select a graph by clicking on it. Most operations only apply to the selected graph.

Overall plot size
Changes the size of the entire plot window and all the graphs in the window.

Set as baseline
When this option is selected, the channel that is currently displayed is subsequently subtracted from all displayed data until "Clear baseline" is selected. Selecting this option does not change the actual data that's stored in the program; it just changes how the data is displayed.

Clear baseline
Disables the baseline feature. This option is grayed out if no baseline is currently set.

Subtract average
When selected, each file's data is displayed with its average subtracted. Selecting this option does not modify the data stored in the program, just the way the data is displayed.

Process menu

The process menu lets you modify data. The operations are applied to an internal copy of the data (i.e., a buffer) and do not affect log files on disk. When you select an item from the process menu, a dialog may appear asking which of the currently-plotted buffers you'd like to apply the operation to.

Add buffer
Adds two buffers together. You're asked to select two buffers from among the buffers that are currently plotted: the buffer to be modified (buffer 1) and the buffer to add (buffer 2). When you click "Apply" or "OK", each point in buffer 1 is added to the first point in buffer 2 that has a time equal to or greater than the time of the point in buffer 1.

Subtract buffer
Subtracts one buffer from another.

Multiply by buffer
Multiplies two buffers together.

Divide by buffer
Divides one buffer by another.

Add constant
Adds a constant to each point in a buffer. You're asked to select one of the currently-plotted buffers and to provide a numeric value. When you click "Apply" or "OK", the value is added to each point in the selected buffer.

Subtract constant
Subtracts a constant from each point in a buffer.

Multiply by constant
Multiplies each point in a buffer by a constant.

Divide by constant
Divides each point in a buffer by a constant.

Kelvin to Celsius
Assuming the contents of a buffer are expressed in Kelvins, converts the data to °C.

Celsius to Kelvin
Assuming the contents of a buffer are expressed in °C, converts the data to Kelvins.

Celsius to Fahrenheit
Assuming the contents of a buffer are expressed in °C, converts the data to °F.

Fahrenheit to Celsius
Assuming the contents of a buffer are expressed in °F, converts the data to °C.

Align start time
Shifts one buffer in time so that its earliest time matches the earliest time of another buffer. Useful for comparing results from two different experiments.

Average plotted buffers
Replaces the contents of whichever buffer is plotted in black with the average of all plotted buffers.

Copy
Creates a new buffer that contains a copy of all data from an existing buffer.

Crop
Creates a new buffer that contains a copy of data from an existing buffer. Only points that falls within the graph's X range are copied.

Derivative
Replaces each data point with the difference between it and the succeeding point.

Downsample
Reduces the number of points in a buffer by averaging two or more neighboring points together and storing the result in a single point. You're asked to provide a "downsampling constant", which is the number of neighboring points to average together. A downsampling constant of 3, for example, reduces the number of points in the buffer to one-third of its previous value.

Lowpass
Removes noise by emulating an analog RC lowpass filter. Similar to the CTC100's lowpass filter, except it's first-order rather than sixth-order.

Median filter
Removes single-point noise spikes with a sliding-window median filter. The filter replaces each data point with the median value of itself, the previous point, and the next point.

Normalize
Subtracts a constant from a buffer, then multiplies the buffer by another constant, such that the minimum value in the buffer is zero and the maximum value is one.

Revert to saved
Re-loads a buffer from disk, discarding the effects of all operations performed with the Process menu.

Smooth
Removes noise using a sliding-window Gaussian filter. Smoothing replaces each data point with a weighted average of data acquired before, during, and after the point.

Subtract average
Subtracts the average value of a buffer from all data points in the buffer.

Subtract initial
Subtracts the value of the first point in a buffer from all data points in the buffer.

Subtract slope
Subtracts the overall slope from a buffer.

Undo
Undoes the last operation performed with the Process menu.

Special menu

This menu contains macros that each consist of some combination of operations from the other menus. The list of macros is defined in the file Resource\SpecialMenu.rsc, and the individual macros are defined by files in the Resource directory.

Small plot size
Displays a single graph in a 200 x 375-pixel window.

Medium plot size
Restores the default single graph in a 294 x 486-pixel window.

Large plot size
Displays a single graph in a 600 x 1000-pixel window.

Add small header
Adds a 50-pixel-tall graph with no X axis labels above the current graph.

Add large header
Adds a 100-pixel-tall graph with no X axis labels above the current graph.

Remove headers
Removes all graphs except for the bottom graph.

Command line and macro instructions

Instruction	Description
add "*buffer1*", "*buffer2*"	add two buffers: buffer1 = buffer1 + buffer2
addGraph[b] 350	add a new graph to the display; specify height; option b=put new graph below current graph
addx "*buffer*", 0.0	add constant: buffer = buffer + constant
alignAll	align the start times of all buffers
annotation "*annotation*"	adds the specified text to the corner of the graph specified with annotationPosition
annotationPosition "*position*"	sets the position of the annotation to "top left", "bottom center", etc.
antialias on/off	set antialiasing on or off; off by default
autoscale[XY] on/off	sets automatic X and Y axis scaling on or off; on by default
axisDivisions 4,4	set the number of X and Y grid lines
break[Pos][Neg] "*sourceBuffer*", "*resultBase*"	break buffer at marks [positive/negative marks only]
cp[n] "*buffer*" [, "*buffer2*",...]	clear the plot, then plot the indicated n buffers
clearMark 1	clears the indicated mark (use drawMarks to see mark numbers)
clearMarks	clears all stored marks
clearPlot	remove all buffers from the plot
copy "*sourceBuffer*", "*destinationBuffer*"	make a copy of a buffer
crop "*sourceBuffer*", "*destinationBuffer*"	crop sourceBuffer to the time segment currently visible on the plot
directory "*directoryName*"	set the current directory
div "*buffer1*", "*buffer2*"	divide one buffer by another: buffer1 = buffer1 / buffer2
diva "*buffer1*"	divide a buffer by its average
divx "*buffer*", 1.0	divide by constant: buffer = buffer / constant
drawMarks	draws a vertical red line on the plot at the location of each stored mark
fontSize 10	set the size of the font used to label the graph axes
hideMarks	hide tick marks
level "*buffer*"	subtract the average slope from a buffer
linewidth[pga] 1	set the width of the plot (option p), grid (g), and/or axis (a) lines in pixels
load "*buffer*", "*fileName*"	load a file into a new buffer; specify a name for the buffer and the name of the file to load
lowpass "*buffer*", 1.0	lowpass-filter a buffer; specify the time constant in seconds

markLevel "*buffer*", 1.0, 1.0	stores marks that indicate when the specified buffer enters a level plus or minus a tolerance
median "*buffer*"	median filter a buffer
moveMark 1, 0.0	moves the indicated mark (use drawMarks to see mark numbers) forward 0.0 seconds
mpy "*buffer1*", "*buffer2*"	multiply two buffers: buffer1 = buffer1 * buffer2
mpyx "*buffer*", 0.0	multiply by constant: buffer = buffer * constant
norm "*buffer*"	normalizes a buffer, i.e. performs linear scaling such that all y values are between 0 and 1
normAll	normalizes all buffers
plot[n] "*buffer*" [, "*buffer2*",...]	add the current contents of *n* buffers to the plot
plotAll	clear plot, then plot all currently-existing buffers
remove "*buffer*"	delete a buffer
removeAll	delete all buffers
removeGraph	remove the currently-selected graph
removeTrace *traceColor*	remove a trace from the plot, by plot color (black, red, blue, orange, green, or cyan)
rep	replot the currently-plotted buffers, reflecting all changes made since the last plot
rev "*buffer*"	revert a buffer to the most recently saved version
riseStats[column] "*buffer*"	display rise statistics for a buffer; "column" option defines columns
roundYAxis on/off	if set to on, automatically-scaled y axes will be set to a round number of units; off by default
saveData "*buffer*", "*fileName*"	save a buffer as a text file
savePlot "*fileName*"	save the current plot as a GIF in the current directory
selectGraph 0	selects the indicated graph; 0=first graph to be added, 1=second graph, etc.
selectionWindow[add/remove] "buffer"	add or remove a graph from the graph selection window
setDefaultBounds	sets the current size and position of the FileGrapher window as the default
setMarks[Pos][Neg] "*buffer*"	store [positive/negative] marks from the specified buffer (for use with break and riseStats)
setSize 500,350	set the x and y size of the plot in pixels; -1 = no change
showBuffers	list names of all currently-existing buffers
showMarks	show tick marks on all plotted buffers
smooth "*buffer*", 0	apply a Gaussian smoothing filter; specify radius in data points
sub "*buffer1*", "*buffer2*"	subtract two buffers: buffer1 = buffer1 - buffer2
subx "*buffer*", 0.0	subtract constant: buffer = buffer - constant
suba "*buffer*"	subtract average: buffer = buffer - ave(buffer)
subi "*buffer*"	subtract initial: buffer = buffer - buffer[0]
undo "*buffer*"	undoes the last operation that modified the indicated buffer
wave "*buffer1*", "*buffer2*", 1.0	weighted average: buffer1 = (buffer1 + buffer2*weighting factor) / (1 + weighting factor)

xLabel "*state*"	Sets the X-axis label to "dateTime" (date and time), "elapsedTime" (elapsed time), or "off" (none),
yLabel "*text*"	Label the Y axis of the graph with the indicated text

Circuit description

Each of the six I/O cards (two sensor input, two heater output, analog I/O, and digital I/O) includes an Atmel ATmega microcontroller (U110). The microcontroller has onboard flash and SRAM. Its clock signal comes from an external 16 MHz oscillator located on the CTC100's backplane. The microcontroller controls the ADCs or DACs on each I/O card.

Each I/O card has a status LED that turns on or off each time an ADC conversion occurs; that is, if 10 ADC conversions are occurring each second, the LED blinks 5 times per second. If the status LED does not blink while the CTC100 is running, or does not blink at the same rate as the status LEDs on the other I/O cards, the card has a hardware or software problem.

I/O cards are calibrated at the factory, and the microcontroller's built-in EEPROM holds the card's calibration data. Input cards produce calibrated readings in the "native units" of the sensor; for example, the RTD card provides calibrated resistance readings, while the thermocouple provides calibrated voltage readings. The CPU card converts these readings to temperatures using calibration data for the particular sensor. Output cards provide calibrated outputs in watts.

The microcontroller is interfaced to the backplane bus with a transceiver (U120). An RS-232 port is available but only used for debugging. The backplane bus uses a proprietary synchronous communication protocol.

CPU board

The CPU (U1) is a Xilinx Zynq XC7Z020-1CLG484C with two ARM Cortex-A9 cores running at 667 MHz. One core operates the I/O cards and the PID feedback loop, while the other operates the user interface.

The CPU has 512 MB of SDRAM (U9, U18), plus 32 MB of flash (U8) used for storing the software and user settings.

Four different voltages are provided by switching power supplies (the 5V supply is not populated, since it's provided by the backplane). As required by the Zynq, the 5V supply turns on first; then power supply sequencer U2 enables the 1.0V, 1.8V, 1.5V, and finally the 3.3V supply. Each voltage must reach a preset minimum value before the next supply is enabled. If the voltage does not reach the minimum within 40 ms, all the power supplies are switched off and the red "power bad" LED is illuminated. Therefore, if one supply is shorted out, none of the supplies will power up.

The USB host port is provided by an FTDI Vinculum-II (U3) which is programmed with custom firmware. LED D2 indicates that the Vinculum-II's microprocessor is running. If the Vinculum-II firmware crashes, the LED stops blinking for 6 seconds before a watchdog timer restarts the Vinculum.

The USB device port is an FTDI FT230XQ USB-to-RS232 converter (U13).

The LCD controller is implemented in the FPGA fabric of the Zynq chip.

The I/O cards connect to GPIO pins on the Zynq through two 5V-tolerant transceivers.

The Ethernet controller is built into the Zynq chip and is connected to an Ethernet PHY (U11) and its magnetics (L2).

Backplane

The backplane uses a proprietary parallel bus to connect the CPU card to the six I/O cards and the front panel. The bus has six I/O card slots. All six slots are equivalent; only the chassis cutouts constrain which cards are plugged into which slots.

The backplane also includes +5V and +3.3V switching power supplies for the CTC100's digital components, and +10, +20, and -20V switching supplies for the low-noise analog circuitry. Jumper J203 connects the analog supply ground to the system ground; if removed, the analog supplies operate with a floating ground.

A circuit is available (U201, U202) to synchronize the switching frequency of the various switching supplies with each other, potentially reducing noise. The circuit is normally not used since it doesn't have a noticeable effect on noise levels.

An AC power bus (J100–J104) distributes 120 or 220VAC power to any CTC420 AC output cards that are installed. The AC power connectors have a lifetime of 25 mate/unmate cycles.

Connected to the AC bus is a line trigger circuit that synchronizes the A/D sampling (actually the CONV* signal; see the description of pin C18 below) with the 50 or 60 Hz line frequency. If this circuit fails, the CTC100 may become unresponsive. Jumper J160 can be used to synchronize the CONV* signal to a 1 MHz clock instead of the line frequency; in this case, the A/D sampling period can be set to any integer multiple of 1 µs rather than being limited to an integer multiple of the line period, but 60 Hz interference is inevitable. Jumper J160 should not be moved while the CTC100 is turned on.

The pinout of the I/O card connectors on the backplane bus is described below. The pin numbers and some pin names are printed next to each I/O card's backplane connector.

Power

A31–A32: 8V. An analog supply used to generate +5V.
B31–B32: +20V. An analog supply used to generate +15V.
C31–C32: -20V. An analog supply used to generate -15V.
A29–A30, B29–B30, C29–C30: AGND. Ground for the analog supplies. May be floating relative to digital ground.
A27–A28: +3.3V. Powers the ColdFire CPU and other components on the CPU card.
B27–B28, B12–B22, A3, A12, A14, A18, B3, C3, C19: DGND. Ground for the +3.3V and +5V supplies.
C27–C28: +5V. Powers the Atmel microcontrollers and all other digital components on the I/O cards.
A25–A26, B25–B26, C25–C26: +24V. Connects directly to the CTC100's 24V "brick" power supply. Used for all high-current outputs.
A23–A24, B23–B24, C23–C24: 24VR. Ground return for the +24V supply.
A1, A2, B1, B2, C1, C2: 24VGND. Ground for +24V.

Parallel bus

This proprietary 8-bit data bus is used for communication between the CPU card and I/O cards.

A4–A11: ADD[0:7]. The address lines. ADD0–ADD3 are used to select a specific card. ADD4–ADD7 are not used.
A13. CLK (Clock). A 16 MHz clock signal used for the Atmel microcontrollers.
A15: RESET*. When pulled low, the Atmel microcontrollers on all I/O cards are reset, regardless of whether or not CS* is active. Used to upload firmware onto the microcontrollers.
B4–B11, C4–C11: D[0:15]. The data lines. Only D0–D7 are currently used.

C13: CS* (Card Select). Each I/O card has its own active-low select line. An address decoder on the backplane decodes a 4-bit address provided by the CPU and pulls the appropriate select line low. Addresses 0–5 select the I/O cards; 6 selects the front panel; 7 is not used; and addresses 8 and above select none. When low, the I/O card can send and receive messages from the CPU, during which time the card stops all other activity.

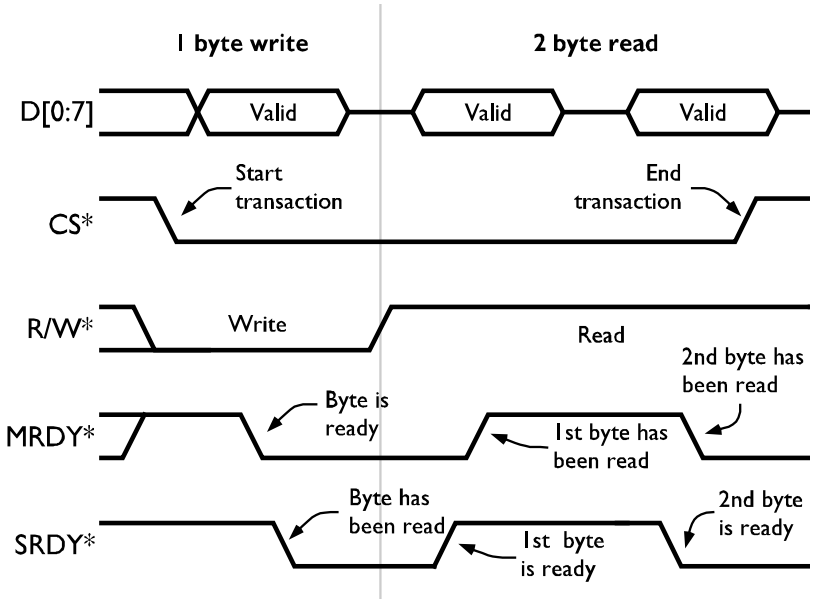
C14: SRDY* (Slave Ready). The I/O card inverts the state of this line after reading data or placing data on the bus. Each bus transaction starts with SRDY* in a high state.

C15: MRDY* (Master Ready). The CPU inverts the state of this line when it places data on the bus (in write mode) or after it has read data (in read mode). Each bus transaction starts with MRDY* in a high state.

C16: R/W* (Read/Write). If high, the selected I/O card takes control of the data lines. If the CPU holds the R/W* line high when CS* is pulled low, the I/O card immediately sends its most recent reading from each channel. Otherwise, the I/O card waits to receive data from the CPU.

C17: SIZ16* (Transfer size 16). Can be used to enable 16-bit data transfers. Currently not used.

C18: CONV* (Convert). A rising or falling edge on this line puts the I/O card into a "standby" state for 5 ms, during which the I/O card is inactive. The CPU card normally requests the I/O card's ADC readings during this period. 5 ms after the falling edge, the I/O card exits the standby state and begins an ADC conversion. If it does not receive the CONV* signal, the I/O card never performs any ADC conversions.



Parallel bus timing diagram. For simplicity only a 1-byte write and 2-byte read are shown, but reads and writes generally transfer at least 3 bytes each.

SPI bus

The SPI bus is used to reprogram the Atmel microcontrollers on the I/O cards. The card's Card Select (CS*) line must be pulled low for its SPI bus to become active.

C20: SCK (SPI Clock).
C21: MOSI (Master out, slave in).
C22: MISO (Master in, slave out).

UART

Connected to the CTC100's back-panel RS-232 port. The I/O cards do not use and are not connected to the backplane UART.

A19: CTS (Clear to Send).
A20: RTS (Request to Send).
A21: RXD (Receive Data).
A22: TXD (Transmit Data).

Front panel

The front panel connects to the same backplane bus as the I/O cards. An Atmel ATmega162 microcontroller on the front panel PCB detects touchscreen touches and button presses, controls the system fan, generates sounds, and manages the LCD power supplies.

All sounds are generated by the Atmel microcontroller and output as an 8-bit, 60 kHz PWM signal. The speaker driver amplifies this signal, providing 250 mW of power to drive the speaker.

Touchscreen and button presses are detected by touchscreen controller U201, which is connected to the microcontroller with an SPI interface. The Atmel microcontroller automatically plays "click" sounds and illuminates the front-panel LEDs (except for the Output Enable LED) when the buttons or touchscreen are pressed.

The LCD display is illuminated by three strings of built-in LEDs. The LCD backlight supply has three independent constant-current sources that each produce 62.5 mA of current to power one string of LEDs. The BACKLIGHT_ON* signal is driven by one of the Atmel microcontroller's PWM outputs. The LCD display can be dimmed by rapidly switching the backlight LEDs on and off.

The fan driver converts a PWM signal from the Atmel microcontroller into a constant-current output. The microcontroller can vary the fan speed by changing its PWM output. The front panel has provisions for a fan tachometer that are currently not used.

The RS-232 port is provided for debugging and is not used.

GPIB card

The GPIB interface is based on a National Instruments TNT4882 GPIB chip. Since the GPIB chip uses a +5V supply, while the other CPU bus components use a +3.3V supply, 5V-tolerant transceivers are needed to interface the chip with the CPU bus. A glue logic chip, U160, resolves incompatibilities between the GPIB's data bus and the CPU bus.

Sensor input cards

Each of the two sensor input cards has two input channels. The cards measure the resistance of thermistors and RTDs by passing an excitation current through both the sensor and a reference resistor in series with the sensor. An ADC measures the ratio between the sensor and reference voltages.

Diode sensor voltages are measured with a similar technique, except a 5V reference is used instead of the reference resistor.

Because each card has two independent channels, it has two copies of each of the following analog circuits. Part references are given for one circuit only.

Variable current source: generates the excitation current. A 10V reference (U610), resistor ladder, and 8:1 multiplexer (U620) produce one of eight voltages: 200 mV, 300 mV, 500 mV, 1 V, 2V, 3V, 5V, or 10V. Op amp U650A provides the excitation current, keeping the voltage across a sense resistor equal to the selected voltage. Multiplexers U630 and U670 select one of

three sense resistors (1 kΩ, 100kΩ, or 10MΩ). The voltage across the sense resistor is measured by a unity-gain instrumentation amplifier (U660).

Fixed 10 µA current source: generates a high-accuracy excitation current for diode sensors. Voltage reference U640 maintains a 5V potential across R642, thereby producing the 10 µA current. Op amp U650B provides a virtual ground for the reference; the virtual ground voltage is the same as the voltage at the bottom of R642. Zener diode D641 prevents this voltage from exceeding 5V, which is the maximum value that can be read by the ADCs.

Reference resistors: Mechanical relays are used to select one of four reference resistors. Mechanical relays are needed because the input protection diodes of semiconductor switches would leak current and produce unacceptable errors.

When reading diode sensors, the excitation current still passes through a reference resistor, although the reference resistor voltage is not actually used. In this case, the lowest-resistance reference is selected.

Select current source and forward/reverse current: Multiplexer U230 controls the direction of current flow through both the temperature sensor and the reference resistor. The current bypass, U210, is engaged while switching between different excitation current values and when the excitation current is set to "off". It keeps the excitation circuit from developing a large voltage (which could be damaging to diode sensors) when it is disconnected.

Multiplexer U210 shunts the excitation current through either D200 or U233. D200 adds a 300 mV offset, ensuring that the voltages at the inputs of op amps U260A–D are above the minimum value. Current is shunted through U233 when the 30x gain circuit is enabled; it adds a 2.5V offset.

ADC input buffers: These op amps isolate the signal and reference resistors from the current produced or drawn by the ADC input pins. The buffers are equipped with RC networks that allow them to drive 1 µF capacitors. Multiplexers U250A–B activate a 30x gain circuit used when necessary to keep the reference voltage above the 100 mV minimum required by the ADC. The gain applies to both the signal and the reference voltage.

Compensate for current direction: When reverse current is selected, a multiplexer ensures that the voltage at the ADC's REF+ pin is more positive than the voltage at the REF- pin. The multiplexer creates a significant voltage drop because it has a 4 Ω resistance and the ADC's REF+ and REF- pins draw a few microamps of current. To compensate for this voltage drop, the feedback network of each ADC input buffer is connected through the multiplexer to a point as close as possible to the actual ADC input pins.

ADC: a 24-bit, delta-sigma ADC, the LTC2440's input range is -0.5·Vref – +0.5·Vref, where Vref is the difference between the voltages at pins ref+ and ref–.

Temperature sensor: U140 reports the temperature of the board's analog section and is used to regulate the CTC100's fan speed. The sensor is read by a built-in ADC on the microprocessor.

Heater driver cards

Each heater driver card outputs 2 A of current with a compliance voltage of up to 55V.

+24V to +50V 2A boost regulator: this switching regular boosts the power supply voltage to the level required by the constant-current heater driver. Although the output of the regulator is labeled "+50V" on the schematics, in fact it can be adjusted to any value between 28 and 55V, or it can be disabled, in which case the heater drivers receive a +23.8V supply. The card's microcontroller sets the regulator output via a PWM signal such that it is always slightly above the heater voltage.

Switching regulator U210 regulates the supply such that the voltage at its feedback pin (FB, pin 3) is equal to 1.26 V. The feedback pin voltage is produced by a voltage divider between the power supply output and op amp U220A. When op amp U220A outputs 0V, the voltage across the heater (OUT+) is 55V; when U220A outputs 0.8V, OUT+ is 24V. Diode D221 protects the feedback pin from an over-voltage condition during start-up. R214 sinks current when the op amp output is near its lower rail.

If 50VSHDN is low, regulator U210 is shut down and the heater voltage is limited to 23.8V.

A 10W 120 Ω internal load resistor, R331, is connected between the +50V supply and ground. The microcontroller can switch the current through this resistor on or off to keep the power supply voltage from oscillating at low output voltages and/or currents.

Constant-current heater driver: the card has three independent current-output heater driver circuits which output 1.333A, 0.465A, and 0.200A. If the user has selected a 2A range, all three circuits operate in parallel. If a 0.6A range is selected, the 1.333A circuit is disabled. If a 0.2A range is selected, only the 0.2A circuit is enabled. The three circuits are identical except for the sense resistor. The microcontroller enables each circuit by pulling one of the three lines 2000MA_ONOFF, 200MA_ONOFF, or 20MA_ONOFF low.

A 16-bit DAC, U240, sets the desired output current. The DAC outputs a value between 0 V (no output current) and 4.0 V (highest possible current for the selected range).

Considering the 2.0A circuit, current from the +50V supply flows through sense resistor R251, then through FET Q251, which throttles back the current to the desired level, then to the user's heater.

This high-side configuration is safer than the more common low-side current source, but given the 55V range, it requires a special high-side-sense IC, U290A. The output of this chip is a voltage proportional to the voltage across sense resistor R251 multiplied by 20. When the maximum current is flowing (2A in this case), the output is 4.0 V.

While the current source is enabled, op amp U250A drives FET Q251 such that the output of U233 is equal to the output of the current control DAC, U240. FET Q233 is needed so that the gate of Q251 can be driven with a high voltage (up to +50V).

FET Q251 can dissipate up to 10 W of power. If it is not kept sufficiently cool, it may fail in the "on" position. Therefore a temperature sensor, U140, measures the temperature of the heatsink. The sensor outputs a voltage of 1 mV/°F which is read by one of the microcontroller's ADC inputs. The microcontroller requests increasing cooling from the system fan as the heatsink temperature rises above 35°C. If the heatsink temperature exceeds 60°C, the microcontroller causes an error message to appear on the CTC's front panel and disables the output.

A pair of automatically-resetting fuses (F221, F222) cuts off the output current if it exceeds 2 A. The current passes through the user's heater, which is connected to J200. A second sense resistor, R208, is used to measure the return current. If the return current differs from the output current by more than 0.25A, the microcontroller requests that an error message be displayed on the CTC's front panel.

Voltage and current monitor: a multiplexed 16-bit ADC, U280, monitors the heater current, the voltage across the heater, and the return current. The ADC has a range of 0–4V. The heater current is monitored by measuring the voltage across the sense resistor, which is 0.2V when no current is flowing and 4.0V when the maximum current for the selected range is flowing.

Analog I/O card

The analog I/O card has four channels that can be used as DAC outputs or ADC inputs.

On-card regulators produce +5, +15, and -15V analog supply voltages.

A 4-channel DAC, U202, produces four 0–5V outputs, which are converted to ±10V by U203A-D. Switches U204A-D can disconnect any of the DAC outputs from the card's BNC connectors, changing the affected channels from DAC outputs to ADC inputs.

The outputs of the four switches are connected to the card's four BNC connectors. A self-resetting fuse, F301-4, temporarily shuts off the current if it exceeds 200 mA. The normal resistance of the fuse is about 1.5 Ω. D301 protects the card from electrostatic discharge and excessive voltages.

U206 multiplexes the four channels into a 24-bit ADC. Since the ADC has a 0–5V range while the inputs are specified for a ±10V range, the input voltage is divided by 4 and offset by 2.5V.

The microcontroller communicates with the analog section through an optoisolated SPI bus. A two-bit address (SPI_ADD0, SPI_ADD1) provided to an address decoder (U302) selects one of three chips on the bus: an SPI-to-parallel adapter (U340), the ADC, or the DAC. The SPI-to-parallel adapter controls the ADC's multiplexer and the direction (input or output) of each channel. The ADC's BUSY signal, which is high while the ADC is performing a conversion, is also connected to the microcontroller through an optoisolator; this signal tells the microcontroller when an ADC conversion is complete and without it the microcontroller freezes up.

Digital I/O card

The CTC's eight digital I/O (DIO) lines can be user-configured to serve as inputs or outputs. All eight lines must have the same direction.

The DIO lines are presented on a 25-pin D connector, J200. Resistors RN200 and RN201 terminate the lines. Capacitors C200–C207 provide ESD protection, while D200, D202, D204, and D206 provide overvoltage protection. The parallel-to-SPI converter, U210, reads the inputs, while the SPI-to-parallel converter U220 produces the outputs. When DOUT_EN* is high, the outputs of U210 are placed into a high-impedance state and the DIO lines serve as inputs. When DOUT_EN* is low, U210 is enabled and the DIO lines serve as outputs.

Since the digital I/O lines are optically isolated and have a floating ground, U210 and U220 are powered by an isolated 5V power supply.

The DIO card also includes four non-latching relays, K401–K404. Each relay is double throw Pins 2, 3, and 4 serve as a monitoring relay. If the monitoring relay fails to switch as expected, XOR gates U410 notify the microcontroller by pulling one of OUT1MON, OUT2MON, etc. high.

Parts List

Chassis (assembly 199)

0-00048	6-32 KEP	Kep nuts for fan
0-00079	4-40X3/16 M/F	Black hex head screws for RS-232 and DIO connectors
0-00120	16 #18	Wire
0-00149	4-40X1/4PF	Mounting screws for front panel and power supply
0-00159	FAN GUARD	
0-00167	6-32X1/2RP	Mounting screws for backplane bracket
0-00179	RIGHT FOOT	
0-00180	LEFT FOOT	
0-00185	6-32X3/8PP	Screws for front feet
0-00204	REAR FOOT	
0-00238	6-32X1/4PF	Backplane mounting screws
0-00288	7 #24	Wire
0-00315	6-32X7/16 PP	Screws for rear feet
0-00328	8 #18 RED	Wire
0-00329	8 #18 BLACK	Wire
0-00371	4-40X3/16PF	Bezel mounting screws
0-00457	6-32X1/4PP/LW	Speaker and backplane mounting screws
0-00517	BINDING POST	Ground lug
0-00525	8-1/4 #18	Wire
0-00901	0.25	Ring terminal for ground lug
0-00978	M4 X 6MM	Power supply mounting screws
0-01013	18GREEN W/YELL	Wire
0-01014	4GREEN W/YELL	Wire
0-01212	6-32X1/4 BLACK	Wire
0-01226	16 BROWN	Wire
0-01227	36151	"PIDG ring terminal, 22-18 guage"
0-01228	696366-1	Blue faston terminal 0.25 in
0-01246	51864	"PIDG ring terminal, 16-14 AWG, #6, for ground wire"
0-01320	37CFM / 24V	Fan
0-01323	74-IFH5	Fuse holder
0-01324	10EAS1	AC power inlet
0-01325	#4-40	I/O card support screws
0-01327	3-520412-2	0.188 inch faston terminal
0-01329	10-32 x 1/4 TR	Black Phillips screws for top and bottom covers
1-00340	10 COND DIL 13	10 PIN DIL CNCTR to 9 PIN D-SUB
1-00472	"2 PIN, 24AWG/WH"	Plug for 24V board-to-board wiring
1-00496	6 POS 18GA ORNG	Plug for 24V supply-to-board wiring
1-01093	"3 PIN, 640441-3"	3 pin white plug for fan
1-01182	3 PIN	AC power plug for motherboard
1-01183	43375-0001	Crimp contacts for AC power plug
1-01187	60 COND.	Front panel ribbon cable
1-01252	050R33-075B	White LCD display cable
2-00068	RC1083BBLKBLKFF	Power switch
6-00076	2 SPKR	2 inch speaker
6-01012	24V / 240W	Main switching power supply
6-01013	5TT 4-R	4A 120V slow blow fuse
6-01014	34.312	2A 250V slow blow fuse
7-00122	BAIL	Stand
7-01002	IGC BEZEL	Plastic bezel for front panel
7-01286	"IGC, RCK SHELF"	Rack mount tray
7-02180	CTC100 CHASSIS	Chassis metal
7-02181	CTC100 F/P	Front panel
7-02182	CTC100 LEXAN	Plastic front panel overlay
7-02184	CTC100 M/B BRKT	Motherboard bracket
7-02185	CTC100 BRKT P/S	Power supply mounting bracket

	7-02186	CTC100 BOT COVE	Bottom cover
	7-02187	CTC100 CPU BRKT	CPU card angle bracket
	7-02201	CTC100 DISPLAY	Paper spacer; goes between LCD display and front panel PCB
	7-02206	CTC100 TOP COVE	Top cover
	7-02225	CTC100 COVER	Cover for GPIB cutout, with RS-232 connector mount
	8-00021	TOUCH PANEL	Touch-sensitive overlay for LCD display
	8-00098	LCD DISPLAY	QVGA display
	9-00267	GENERIC	Serial number label

CPU card *(assembly 383)*

J3	0-00985	ENETLED (G/Y)	Conn RJ-45 F 8 POS 2.03mm Solder RA Thru-
BT1	0-01089	1065	Battery Holder Snap in Automotive
Z	0-01325	Mtg Bckt #4-40	
JDR301	1-00236	120 PIN RT ANGL	3 Row, Right Angle Mount
J5	1-00350	4 PIN, USB	Conn USB 2.0 Type
J202	1-01290	40 PIN	Conn Board Stacker HDR 40 POS
J16	1-01347	87831-1420	2 mm DIP header, 14-pin vertical
J4	1-01351	951106-8622	2 mm SIP header, 6-pin vertical
J630	1-01396	USB-A1HSB6	CONN USB TYPE A R/A BLACK
Q1	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q2	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q4	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q5	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q6	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q7	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q8	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q10	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q11	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q12	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q36	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
U7	3-01203	SN74LVC1G08DBVR	AND Gate 1-Element 2-IN CMOS 5-Pin SOT-23 T/R
U12	3-01203	SN74LVC1G08DBVR	AND Gate 1-Element 2-IN CMOS 5-Pin SOT-23 T/R
U6	3-01233	DS1672S-33	Real-time clock
U302	3-01235	74LCX04M	Low voltage hex inverter with 5V tolerant inputs
U303	3-01236	74LCX16245MTD	Bus XCVR Dual 16-CH 3-ST 48-Pin TSSOP W T/R
U304	3-01236	74LCX16245MTD	Bus XCVR Dual 16-CH 3-ST 48-Pin TSSOP W T/R
D3	3-01342	STZ5.6NT146	Diode Zener Dual Common Anode 5.6V 5% 300mW
U4	3-01836	TPS2042BD	USB Power Switch Dual 5.5V 0.75A to 1.25A 8-Pin SOIC Tube
U24	3-01890	LTC1326CS8#PBF	Processor Supervisor 3.118V/2.363V/Adj
Q21	3-02210	BSS84	P-channel MOSFET, SOT-23, Rds(on)=10 ohms
Q16	3-02211	BSZ042N04NS G	N-channel MOSFET, Vds=40V, Rds=4.2 mOhm, Id=40A
U3	3-02215	VNC2-48L1B	Vinculum-II USB controller, LQFP-48 package
U10	3-02233	74LVC1T45	Bus Driver
U5	3-02242	SE95D	I2C temperature sensor
Q9	3-02247	2N7002	N-channel MOSFET
D1	3-02250	HSMG-C150	1206 surface mount LED, green
D2	3-02250	HSMG-C150	1206 surface mount LED, green
D4	3-02250	HSMG-C150	1206 surface mount LED, green
D10	3-02250	HSMG-C150	1206 surface mount LED, green
D11	3-02250	HSMG-C150	1206 surface mount LED, green
D12	3-02250	HSMG-C150	1206 surface mount LED, green
D13	3-02250	HSMG-C150	1206 surface mount LED, green
D14	3-02250	HSMG-C150	1206 surface mount LED, green
D15	3-02250	HSMG-C150	1206 surface mount LED, green
D16	3-02250	HSMG-C150	1206 surface mount LED, green
D17	3-02250	HSMG-C150	1206 surface mount LED, green
D18	3-02250	HSMG-C150	1206 surface mount LED, green
D39	3-02250	HSMG-C150	1206 surface mount LED, green
Q18	3-02252	BSZ105N04NS G	N-channel MOSFET, Vds=40V, Rds=10 mOhm, Id=40A
Q20	3-02252	BSZ105N04NS G	N-channel MOSFET, Vds=40V, Rds=10 mOhm, Id=40A
Q24	3-02252	BSZ105N04NS G	N-channel MOSFET, Vds=40V, Rds=10 mOhm, Id=40A
D36	3-02254	BAT54T1G	Schottky diode, 200 mA, 0.35V@10 mA

D37	3-02254	BAT54T1G	Schottky diode, 200 mA, 0.35V@10 mA
D38	3-02254	BAT54T1G	Schottky diode, 200 mA, 0.35V@10 mA
D42	3-02254	BAT54T1G	Schottky diode, 200 mA, 0.35V@10 mA
U19	3-02255	LM25117	Switching regulator
U46	3-02255	LM25117	Switching regulator
U47	3-02255	LM25117	Switching regulator
U48	3-02255	LM25117	Switching regulator
Q15	3-02257	BSZ165N04NS G	N-channel MOSFET, Vds=40V, Rds=16.5 mOhm, Id=31A
Q17	3-02257	BSZ165N04NS G	N-channel MOSFET, Vds=40V, Rds=16.5 mOhm, Id=31A
Q19	3-02257	BSZ165N04NS G	N-channel MOSFET, Vds=40V, Rds=16.5 mOhm, Id=31A
Q23	3-02257	BSZ165N04NS G	N-channel MOSFET, Vds=40V, Rds=16.5 mOhm, Id=31A
U49	3-02270	TPS51200DRCT	DDR3 termination regulator
U8	3-02287	S25FL256SAGMFI001	S25FL256S, 256 Mbit serial Flash Memory
U13	3-02314	FT230XQ	USB to UART converter
D5	3-02363	1SMB5914BT3G	Zener diode, 3.6V 3.0W
U9	3-02364	MT41J128M16JT	DDR3 SDRAM, 256 MB, x16, 800 MHz
U18	3-02364	MT41J128M16JT	DDR3 SDRAM, 256 MB, x16, 800 MHz
D23	3-02383	S5AC-13-F	5A SMC diode, forward voltage=1.15V@5A
D24	3-02383	S5AC-13-F	5A SMC diode, forward voltage=1.15V@5A
D8	3-02384	RED	LED, red 1206
U2	3-02385	LTC2924CGN	Quad power supply sequencer
U11	3-02386	88E1116RA0	10/100/1000BASE-T Ethernet PHY, RGMII
U1	3-02387	XC7Z020-1CLG484	Zynq FPGA, 484-pin package
RN301	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
R87	4-01406	0	Resistor, Thick Film, 5%, 300 ppm, SMT
R112	4-01670	20K 1%	MPM Series Resistive Divider, Thin Film, 10.0K x 2, 0.1W, 1%
R6	4-01796	0	Resistor, Thick Film, 5%, 300 ppm, 0603 Chip
R7	4-01796	0	Resistor, Thick Film, 5%, 300 ppm, 0603 Chip
R9	4-01796	0	Resistor, Thick Film, 5%, 300 ppm, 0603 Chip
R12	4-01796	0	Resistor, Thick Film, 5%, 300 ppm, 0603 Chip
R24	4-01829	22	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R25	4-01829	22	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R32	4-01829	22	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R38	4-01829	22	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R39	4-01829	22	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R40	4-01829	22	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R41	4-01829	22	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R108	4-01830	24	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R27	4-01831	27	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R30	4-01831	27	Resistor, Thick Film, 5%, 200 ppm, 0603 Chip
R2	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R3	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R42	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R46	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R74	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R82	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R85	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R103	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R106	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R107	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R109	4-01885	4.7K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R20	4-02040	60.4	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R21	4-02040	60.4	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R26	4-02040	60.4	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R288	4-02040	60.4	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R160	4-02075	140	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R163	4-02075	140	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R166	4-02075	140	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R174	4-02075	140	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R178	4-02075	140	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R179	4-02075	140	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R182	4-02075	140	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R185	4-02075	140	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R188	4-02075	140	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip

R195	4-02080	158	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R71	4-02098	243	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R234	4-02098	243	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R286	4-02100	255	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R14	4-02111	332	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R15	4-02111	332	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R81	4-02111	332	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R177	4-02111	332	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R104	4-02157	1.00K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R105	4-02157	1.00K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R281	4-02158	1.02K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R282	4-02159	1.05K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R372	4-02159	1.05K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R199	4-02165	1.21K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R283	4-02165	1.21K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R285	4-02165	1.21K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R304	4-02174	1.50K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R37	4-02186	2.00K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R83	4-02195	2.49K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R97	4-02203	3.01K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R284	4-02212	3.74K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R98	4-02213	3.83K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R273	4-02220	4.53K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R84	4-02222	4.75K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R91	4-02224	4.99K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R92	4-02224	4.99K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R137	4-02224	4.99K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R344	4-02224	4.99K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R345	4-02224	4.99K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R298	4-02242	7.68K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R101	4-02246	8.45K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R13	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R22	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R23	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R28	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R29	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R33	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R34	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R35	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R44	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R79	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R86	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R110	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R111	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R113	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R114	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R115	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R116	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R353	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R354	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R271	4-02258	11.3K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R296	4-02260	11.8K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R99	4-02262	12.4K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R278	4-02265	13.3K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R270	4-02267	14.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R36	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R45	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R100	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R136	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R167	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R168	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R170	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R171	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R200	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip

R231	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R232	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R269	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R274	4-02282	20.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R279	4-02297	28.7K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R302	4-02297	28.7K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R73	4-02303	33.2K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R102	4-02320	49.9K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R280	4-02320	49.9K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R355	4-02349	100K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R31	4-02416	499K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R88	4-02416	499K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R89	4-02416	499K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R90	4-02416	499K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R80	4-02567	100	100 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R275	4-02568	0.020 ohm	Current sense resistor, 0.5W 1%
R276	4-02568	0.020 ohm	Current sense resistor, 0.5W 1%
R277	4-02568	0.020 ohm	Current sense resistor, 0.5W 1%
R300	4-02568	0.020 ohm	Current sense resistor, 0.5W 1%
R47	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R48	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R50	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R51	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R52	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R53	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R54	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R55	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R56	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R57	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R58	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R59	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R60	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R61	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R62	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R63	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R64	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R65	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R66	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R67	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R68	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R69	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R70	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R72	4-02593	40.2	40.2 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R1	4-02594	80.6	80.6 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R4	4-02594	80.6	80.6 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R49	4-02594	80.6	80.6 ohm 0402 resistor, 1/16W 100 ppm/C 1%
R16	4-02596	75	75.0 ohm 0603 resistor, 1/8W 100ppm/C 1%
R17	4-02596	75	75.0 ohm 0603 resistor, 1/8W 100ppm/C 1%
R18	4-02596	75	75.0 ohm 0603 resistor, 1/8W 100ppm/C 1%
R19	4-02596	75	75.0 ohm 0603 resistor, 1/8W 100ppm/C 1%
C314	5-00692	10P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C35	5-00702	27P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C36	5-00702	27P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C513	5-00702	27P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C20	5-00703	30P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C21	5-00703	30P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C38	5-00708	47P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C39	5-00708	47P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C43	5-00708	47P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C87	5-00708	47P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C313	5-00708	47P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C312	5-00718	120P	CAP CER 120PF 50V C0G/NP0 0603
C324	5-00738	820P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603
C325	5-00738	820P	Capacitor, Mono, 50V, ±0.25pF or 5%, NPO, 0603

C326	5-00738	820P	Capacitor, Mono, 50V, ª0.25pF or 5%, NPO, 0603
C510	5-00738	820P	Capacitor, Mono, 50V, ª0.25pF or 5%, NPO, 0603
C394	5-00740	1000P	Capacitor, Mono, 50V, ª0.25pF or 5%, NPO, 0603
C320	5-00744	2200P	Capacitor, Mono, 50V, ±10%, X7R, 0603
C101	5-00752	10000P	Capacitor, Mono, 50V, ±10%, X7R, 0603
C102	5-00752	10000P	Capacitor, Mono, 50V, ±10%, X7R, 0603
C315	5-00752	10000P	Capacitor, Mono, 50V, ±10%, X7R, 0603
C317	5-00761	56000P	Capacitor, Mono, 50V, ±10%, X7R, 0603
C511	5-00761	56000P	Capacitor, Mono, 50V, ±10%, X7R, 0603
C1	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C4	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C17	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C22	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C23	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C24	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C25	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C30	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C31	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C32	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C33	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C34	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C37	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C40	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C41	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C42	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C45	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C46	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C48	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C81	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C82	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C89	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C91	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C92	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C93	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C94	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C95	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C96	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C97	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C99	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C100	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C303	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C304	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C305	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C306	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C307	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C308	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C321	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C322	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C323	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C494	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C505	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C509	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C537	5-00764	0.1UF	Capacitor, Mono, 50V, ±10%, X7R, 0603
C149	5-00790	1UF - 0603	Cap Ceramic 1µF 25V X5R 10% Pad SMD 0603 85??C T/R
C150	5-00790	1UF - 0603	Cap Ceramic 1µF 25V X5R 10% Pad SMD 0603 85??C T/R
C451	5-00790	1UF - 0603	Cap Ceramic 1µF 25V X5R 10% Pad SMD 0603 85??C T/R
C11	5-00870	10 µF	10 µF 1206 capacitor, 6.3V X7R 10%
C16	5-00870	10 µF	10 µF 1206 capacitor, 6.3V X7R 10%
C337	5-00870	10 µF	10 µF 1206 capacitor, 6.3V X7R 10%
C344	5-00870	10 µF	10 µF 1206 capacitor, 6.3V X7R 10%
C349	5-00870	10 µF	10 µF 1206 capacitor, 6.3V X7R 10%
C395	5-00870	10 µF	10 µF 1206 capacitor, 6.3V X7R 10%
C483	5-00871	0.01 µF	0.01 µF 0402 capacitor, 6.3V X5R 10%
C485	5-00871	0.01 µF	0.01 µF 0402 capacitor, 6.3V X5R 10%

C44	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C79	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C85	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C155	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C159	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C195	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C199	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C328	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C329	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C452	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C456	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C467	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C471	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C477	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C495	5-00872	100 µF	100 µF 1210 capacitor, 6.3V X5R 20%
C334	5-00873	4.7 µF	4.7 µF 0805 capacitor, 6.3V X5R 10%
C388	5-00873	4.7 µF	4.7 µF 0805 capacitor, 6.3V X5R 10%
C389	5-00873	4.7 µF	4.7 µF 0805 capacitor, 6.3V X5R 10%
C300	5-00874	0.47µF	0.47 µF 0603 capacitor, 25V X7R 10%
C301	5-00874	0.47µF	0.47 µF 0603 capacitor, 25V X7R 10%
C302	5-00874	0.47µF	0.47 µF 0603 capacitor, 25V X7R 10%
C316	5-00874	0.47µF	0.47 µF 0603 capacitor, 25V X7R 10%
C318	5-00874	0.47µF	0.47 µF 0603 capacitor, 25V X7R 10%
C319	5-00874	0.47µF	0.47 µF 0603 capacitor, 25V X7R 10%
C507	5-00874	0.47µF	0.47 µF 0603 capacitor, 25V X7R 10%
C508	5-00874	0.47µF	0.47 µF 0603 capacitor, 25V X7R 10%
C26	5-00876	1000 pF	1000 pF 1206 capacitor, 2000V X7R 10%
C2	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C3	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C5	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C6	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C7	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C8	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C9	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C10	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C12	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C13	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C14	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C15	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C18	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C19	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C27	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C28	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C47	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C49	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C50	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C51	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C52	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C53	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C54	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C55	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C56	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C57	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C58	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C59	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C60	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C61	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C62	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C63	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C64	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C65	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C66	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C67	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C68	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%

C69	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C70	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C71	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C72	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C73	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C74	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C75	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C76	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C352	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C353	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C420	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C421	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C422	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C423	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C424	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C425	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C426	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C427	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C428	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C486	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C487	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C488	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C489	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C490	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C491	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C492	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C493	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C498	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C499	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C500	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C501	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C502	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C503	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C504	5-00877	0.1 µF	0.1 µF 0402 capacitor, 6.3V X5R 10%
C108	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C109	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C110	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C111	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C112	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C115	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C151	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C152	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C153	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C154	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C163	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C164	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C165	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C166	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C167	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C168	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C169	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C170	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C171	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C172	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C179	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C181	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C182	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C183	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C184	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C185	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C186	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C187	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C191	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C192	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%

C193	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C194	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C224	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C225	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C226	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C335	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C417	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C418	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C419	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C460	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C461	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C462	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C463	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C464	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C465	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C466	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C469	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C470	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C474	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C475	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C476	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C480	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C481	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C482	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C512	5-00878	0.47 µF	0.47 µF 0402 capacitor, 6.3V X5R 10%
C309	5-00879	22 µF	22 µF 2220 capacitor, 25V, X7R, 20%
C310	5-00879	22 µF	22 µF 2220 capacitor, 25V, X7R, 20%
C311	5-00879	22 µF	22 µF 2220 capacitor, 25V, X7R, 20%
C506	5-00879	22 µF	22 µF 2220 capacitor, 25V, X7R, 20%
C78	5-00880	47 µF	47 µF 1206 capacitor, 6.3V X5R 20%
C188	5-00880	47 µF	47 µF 1206 capacitor, 6.3V X5R 20%
C327	5-00880	47 µF	47 µF 1206 capacitor, 6.3V X5R 20%
C330	5-00880	47 µF	47 µF 1206 capacitor, 6.3V X5R 20%
C331	5-00880	47 µF	47 µF 1206 capacitor, 6.3V X5R 20%
C80	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C83	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C84	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C86	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C88	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C90	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C156	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C157	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C158	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C160	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C161	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C162	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C174	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C175	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C176	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C177	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C178	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C189	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C190	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C196	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C197	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C198	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C200	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C209	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C457	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C458	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C459	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C468	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C472	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C473	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor

C478	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C479	5-00903	4.7 µF	4.7 µF 6.3V 0402 capacitor
C98	5-00904	10 µF	10 µF 4V 0603 capacitor
C180	5-00904	10 µF	10 µF 4V 0603 capacitor
C173	5-00905	330 µF	330 µF tantalum capacitor, 2917 (V-case), 2.5V
C29	5-00927	0.22U	0.22 µF 0603 capacitor, 50V X7R 10%
L3	6-00236	FR47	Ferrite Bead, SMD, Type 43/44, 1812
L4	6-00236	FR47	Ferrite Bead, SMD, Type 43/44, 1812
L6	6-00236	FR47	Ferrite Bead, SMD, Type 43/44, 1812
Y4	6-00762	32.768KHZ - 6PF	Crystal 0.032768MHz ±20ppm (Tol) 6pF FUND
Z	6-00789	CR2032 W/OUT PN	Lithium Battery Coin 3V 225mAh Primary Automotive
Y2	6-01062	25 MHz crystal	Crystal 25MHz ±30ppm 18pF FUND
Y3	6-01063	FOXSDLF/120-20	Crystal 12MHz ±30ppm 20pF FUND
L18	6-01066	4.7 uH	SMD power inductor, 5A, 30%, 0.0123 ohms
L20	6-01066	4.7 uH	SMD power inductor, 5A, 30%, 0.0123 ohms
L1	6-01093	BLM18PG121SN1D	0603 Ferrite bead, 2000 mA
L7	6-01093	BLM18PG121SN1D	0603 Ferrite bead, 2000 mA
L10	6-01093	BLM18PG121SN1D	0603 Ferrite bead, 2000 mA
L2	6-01096	TG1G-S002NZRL	12-core 1000Base-T magnetics
L19	6-01098	8.2uH	SMD power inductor, 8.2 uH, 6.3A
Y8	6-01124	33.33333 MHz	33.33333 MHz oscillator
L17	6-01125	2.2 uH	SMD power inductor, 6.6A, 30%, 0.0072 ohms
L5	6-01135	220 OHM@100MHZ	Ferrite bead, 0603, 450 mA, 220 ohms@100 MHz, 0.45 ohm@DC
Z	7-01773	BRACKET PTC10	
PC1	7-02485	CPU board	Zynq CPU board

Backplane *(assembly 209)*

C111	5-00601	0.1UF - 16V X7R	
C121	5-00601	0.1UF - 16V X7R	
C131	5-00601	0.1UF - 16V X7R	
C141	5-00601	0.1UF - 16V X7R	
C142	5-00601	0.1UF - 16V X7R	
C143	5-00601	0.1UF - 16V X7R	
C144	5-00601	0.1UF - 16V X7R	
C150	5-00601	0.1UF - 16V X7R	
C160	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C161	5-00393	3300P	Capacitor, Mono, 50V, 5%, NPO, 1206
C201	5-00601	0.1UF - 16V X7R	
C202	5-00601	0.1UF - 16V X7R	
C211	5-00375	100P	Capacitor, Mono, 50V, 5%, NPO, 1206
C212	5-00472	4.7U/T35	SMD TANTALUM, D-Case
C213	5-00395	4700P -5%	Capacitor, Mono, 50V, 5%, X7R, 1206
C214	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C215	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C216	5-00329	120U	Capacitor, Electrolytic, 35V, 20%, Rad
C217	5-00384	560P	Capacitor, Mono, 50V, 5%, NPO, 1206
C218	5-00318	2.2U/T35	SMD TANTALUM, C-Case
C221	5-00318	2.2U/T35	SMD TANTALUM, C-Case
C223	5-00318	2.2U/T35	SMD TANTALUM, C-Case
C224	5-00318	2.2U/T35	SMD TANTALUM, C-Case
C227	5-00041	220U	Capacitor, Electrolytic, 50V, 20%, Rad
C228	5-00041	220U	Capacitor, Electrolytic, 50V, 20%, Rad
C229	5-00041	220U	Capacitor, Electrolytic, 50V, 20%, Rad
C241	5-00375	100P	Capacitor, Mono, 50V, 5%, NPO, 1206
C242	5-00628	22U - 35V	
C244	5-00399	.01U - 5%	Capacitor, Mono, 50V, 5%, X7R, 1206
C245	5-00640	100U - 10V	
C246	5-00640	100U - 10V	
C251	5-00375	100P	Capacitor, Mono, 50V, 5%, NPO, 1206
C252	5-00628	22U - 35V	

C253	5-00628	22U - 35V	
C254	5-00399	.01U - 5%	Capacitor, Mono, 50V, 5%, X7R, 1206
C255	5-00640	100U - 10V	
C256	5-00640	100U - 10V	
D161	3-00204	1N5230	1N5230, 4.7V, 500mW DO-35 ZENER DIODE
D211	3-00380	1N5248	1N5248, 18V, 500mW, DO-35 ZENER DIODE
D221	3-00479	MUR410	MUR410, 100V, 4A ULTRA FAST DIODE
D222	3-00479	MUR410	MUR410, 100V, 4A ULTRA FAST DIODE
D223	3-00479	MUR410	MUR410, 100V, 4A ULTRA FAST DIODE
D224	3-00479	MUR410	MUR410, 100V, 4A ULTRA FAST DIODE
D225	3-00479	MUR410	MUR410, 100V, 4A ULTRA FAST DIODE
D226	3-00479	MUR410	MUR410, 100V, 4A ULTRA FAST DIODE
D227	3-00479	MUR410	MUR410, 100V, 4A ULTRA FAST DIODE
D228	3-00479	MUR410	MUR410, 100V, 4A ULTRA FAST DIODE
D231	3-00012	GREEN	LED, Rectangular, 0.1÷ x 0.3÷
D232	3-00012	GREEN	LED, Rectangular, 0.1÷ x 0.3÷
D233	3-00012	GREEN	LED, Rectangular, 0.1÷ x 0.3÷
D234	3-00012	GREEN	LED, Rectangular, 0.1÷ x 0.3÷
D235	3-00012	GREEN	LED, Rectangular, 0.1÷ x 0.3÷
D236	3-00012	GREEN	LED, Rectangular, 0.1÷ x 0.3÷
D241	3-01859	B540C-13-F	
D251	3-01859	B540C-13-F	
J100	1-01181	431602103	
J106	1-00166	60 PIN DIL	Header, DIM, Latching Clips
J150	1-00251	10 PIN DIL	Header, DIM, Locking Clips
J160	1-00086	3 PIN SI	Header, SIM
J201	1-00111	6 PIN WHITE	Header, SIM, Polarized
J205	1-00260	4 PIN, WHITE	Header, SIM, Polarized
J206	1-00250	2 PIN, WHITE	Header, SIM, Polarized
J207	1-00471	4 PIN, WHITE	Header, SIM, Polarized
J208	1-00471	4 PIN, WHITE	Header, SIM, Polarized
J209	1-00250	2 PIN, WHITE	Header, SIM, Polarized
J211	1-00006	2 PIN DI	Header, SIM
J241	1-00006	2 PIN DI	Header, SIM
J251	1-00006	2 PIN DI	Header, SIM
JD100	1-00235	96 PIN VERTICAL	3 Row, Vertical
JD101	1-00235	96 PIN VERTICAL	3 Row, Vertical
JD102	1-00235	96 PIN VERTICAL	3 Row, Vertical
JD103	1-00235	96 PIN VERTICAL	3 Row, Vertical
JD104	1-00235	96 PIN VERTICAL	3 Row, Vertical
JD105	1-00235	96 PIN VERTICAL	3 Row, Vertical
JD107	1-00237	120 PIN VERTICA	3 Row, Vertical Mount
L241	6-00691	22UH - SMT	
L251	6-00691	22UH - SMT	
PC1	7-02178	CTC100 BACKPLAN	
Q211	3-00283	IRF530/IRF532	100V,14A, N Channel MOSFET, R(DS)on = 0.140 ohms
Q212	3-00283	IRF530/IRF532	100V,14A, N Channel MOSFET, R(DS)on = 0.140 ohms
R121	4-01439	22	Resistor, Thick Film, 5%, 200 ppm, SMT
R160	4-00082	470K	Resistor, Carbon Film, 1/4W, 5%
R161	4-00082	470K	Resistor, Carbon Film, 1/4W, 5%
R162	4-01510	20K	Resistor, Thick Film, 5%, 200 ppm, SMT
R163	4-01503	10K	Resistor, Thick Film, 5%, 200 ppm, SMT
R164	4-01459	150	Resistor, Thick Film, 5%, 200 ppm, SMT
R165	4-01510	20K	Resistor, Thick Film, 5%, 200 ppm, SMT
R166	4-01448	51	Resistor, Thick Film, 5%, 200 ppm, SMT
R201	4-01439	22	Resistor, Thick Film, 5%, 200 ppm, SMT
R202	4-01406	0	Resistor, Thick Film, 5%, 300 ppm, SMT
R203	4-01406	0	Resistor, Thick Film, 5%, 300 ppm, SMT
R204	4-01406	0	Resistor, Thick Film, 5%, 300 ppm, SMT
R205	4-01406	0	Resistor, Thick Film, 5%, 300 ppm, SMT
R206	4-01406	0	Resistor, Thick Film, 5%, 300 ppm, SMT
R207	4-01406	0	Resistor, Thick Film, 5%, 300 ppm, SMT
R211	4-01479	1.0K	Resistor, Thick Film, 5%, 200 ppm, SMT
R212	4-01158	2.67K	Resistor, Thin Film, 1%, 50 ppm, MELF

R213	4-01455	100	Resistor, Thick Film, 5%, 200 ppm, SMT
R214	4-01455	100	Resistor, Thick Film, 5%, 200 ppm, SMT
R215	4-01021	100	Resistor, Thin Film, 1%, 50 ppm, MELF
R216	4-01021	100	Resistor, Thin Film, 1%, 50 ppm, MELF
R217	4-01001	61.9	Resistor, Thin Film, 1%, 50 ppm, MELF
R218	4-00436	0.1	Resistor, Wire-wound
R231	4-01458	130	Resistor, Thick Film, 5%, 200 ppm, SMT
R232	4-01466	300	Resistor, Thick Film, 5%, 200 ppm, SMT
R233	4-01472	510	Resistor, Thick Film, 5%, 200 ppm, SMT
R234	4-00029	1.8K	Resistor, Carbon Film, 1/4W, 5%
R235	4-00029	1.8K	Resistor, Carbon Film, 1/4W, 5%
R236	4-00048	2.2K	Resistor, Carbon Film, 1/4W, 5%
R241	4-01479	1.0K	Resistor, Thick Film, 5%, 200 ppm, SMT
R251	4-01479	1.0K	Resistor, Thick Film, 5%, 200 ppm, SMT
RN111	4-01727	22X4	
RN112	4-01727	22X4	
RN131	4-01727	22X4	
RN132	4-01727	22X4	
RN143	4-00905	82X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN144	4-00905	82X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN145	4-00905	82X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN146	4-00905	82X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
T211	6-00774	PTC220	
U110	3-01345	74ABT541CSC	
U120	3-01346	74HC4040M	
U130	3-00795	74AC138	74AC138, 3-to-8 Line Decoder
U140	3-01498	74ABT16245CMTD	
U150	3-01239	MAX3233ECWP	
U160	3-00094	LM311	LM311 Voltage Comparator
U201	3-00742	74HC74	74HC74, Dual D-Type Flip-Flop With Clear and Preset, SO-14
U202	3-00782	74HC02	74HC02, Quad 2-Input NOR Gate
U210	3-00919	3525A	3525A, POWER SUPPLY CONTROLLER
U240	3-01347	LM2670S-3.3	
U250	3-01348	LM2670S-5	
Y110	6-00692	16MHZ - SMT	
Z0	1-00470	4 PIN, 24AWG/WH	Non board mount, Female, 24 AWG
Z1	1-00087	2 PIN JUMPER	2 PIN JUMPER ON J160 & J203
Z2	1-00254	2 PIN, 22AWG/RD	Non board mount, Female, Seperate wire, 22 AWG
Z3	1-00259	4 PIN, 18AWG/OR	Non board mount, Female, Seperate wire, 18 AWG
Z4	0-00043	4-40 KEP	
Z5	0-00129	5 #24	
Z6	0-00187	4-40X1/4PP	
Z7	0-00390	1-72X1/4	TO HOLD CNCTRS. DOWN
Z8	0-00391	1-72X5/32X3/64	TO HOLD CNCTRS. DOWN
Z9	0-01015	11RED #18	
Z10	0-01016	11 BLK #18	
Z11	0-01093	563002B00000	

***Front panel** (assembly 210)*

C101	5-00601	0.1UF - 16V X7R
C102	5-00601	0.1UF - 16V X7R
C103	5-00601	0.1UF - 16V X7R
C105	5-00601	0.1UF - 16V X7R
C106	5-00601	0.1UF - 16V X7R
C107	5-00601	0.1UF - 16V X7R
C108	5-00601	0.1UF - 16V X7R
C201	5-00601	0.1UF - 16V X7R
C202	5-00601	0.1UF - 16V X7R
C203	5-00601	0.1UF - 16V X7R
C205	5-00601	0.1UF - 16V X7R
C211	5-00604	0.01UF / 16V
C212	5-00604	0.01UF / 16V

C213	5-00604	0.01UF / 16V	
C214	5-00604	0.01UF / 16V	
C301	5-00519	.33U/T35	SMD TANTALUM, Y-Case
C302	5-00513	1U-16V A-CASE	SMT Tantalum, 16V, A-case (1206, but NEEDS POLARITY mark)
C303	5-00513	1U-16V A-CASE	SMT Tantalum, 16V, A-case (1206, but NEEDS POLARITY mark)
C304	5-00389	1500P	Capacitor, Mono, 50V, 5%, NPO, 1206
C331	5-00407	.047U	Capacitor, Mono, 50V, 10%, X7R, 1206
C360	5-00519	.33U/T35	SMD TANTALUM, Y-Case
C361	5-00519	.33U/T35	SMD TANTALUM, Y-Case
C362	5-00519	.33U/T35	SMD TANTALUM, Y-Case
C370	5-00601	0.1UF - 16V X7R	
D101	3-00576	RED MINI	LED, Subminiature, 1.8mm (T 3/4)
D201	3-00010	GREEN	LED, T1 Package, 3mm diameter
D202	3-00010	GREEN	LED, T1 Package, 3mm diameter
D203	3-00010	GREEN	LED, T1 Package, 3mm diameter
D204	3-00010	GREEN	LED, T1 Package, 3mm diameter
D205	3-00009	YELLOW	LED, T1 Package, 3mm diameter
D206	3-00011	RED	LED, T1 Package, 3mm diameter
D341	3-01253	B270-13	
ISO350	3-01414	MOC213-M	MOC213, Transistor Output Optocoupler, CTR = 100% min, SO-8
J106	1-01253	60 PIN DIL	
J201	1-00559	1.00MM FFC -SMT	
J301	1-00473	2 PIN, WHITE	Header, SIM, Polarized
J341	1-00045	3 PIN STRAIGHT	Header, SIM, w/ Friction Lock
J360	1-01250	SM06B-SHLS-TF	
J370	1-01251	XF2M-3315-1A	
PC1	7-02183	CTC100 F/P PCB	
Q342	3-01989	IRFR3410	
Q360	3-01989	IRFR3410	
Q361	3-01989	IRFR3410	
Q362	3-01989	IRFR3410	
R104	4-01466	300	Resistor, Thick Film, 5%, 200 ppm, SMT
R105	4-01519	47K	Resistor, Thick Film, 5%, 200 ppm, SMT
R201	4-01519	47K	Resistor, Thick Film, 5%, 200 ppm, SMT
R202	4-01431	10	Resistor, Thick Film, 5%, 200 ppm, SMT
R301	4-01510	20K	Resistor, Thick Film, 5%, 200 ppm, SMT
R302	4-01514	30K	Resistor, Thick Film, 5%, 200 ppm, SMT
R331	4-01479	1.0K	Resistor, Thick Film, 5%, 200 ppm, SMT
R360	4-00954	20	Resistor, Thin Film, 1%, 50 ppm, MELF
R361	4-00954	20	Resistor, Thin Film, 1%, 50 ppm, MELF
R362	4-00954	20	Resistor, Thin Film, 1%, 50 ppm, MELF
R363	4-01423	4.7	Resistor, Thick Film, 5%, 300 ppm, SMT
R364	4-01423	4.7	Resistor, Thick Film, 5%, 300 ppm, SMT
R365	4-01423	4.7	Resistor, Thick Film, 5%, 300 ppm, SMT
R371	4-01495	4.7K	Resistor, Thick Film, 5%, 200 ppm, SMT
R374	4-01495	4.7K	Resistor, Thick Film, 5%, 200 ppm, SMT
RN101	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN102	4-00905	82X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN103	4-00905	82X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN104	4-00905	82X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN105	4-01707	47KX4D	
RN202	4-01707	47KX4D	
RN204	4-00908	270X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN205	4-00908	270X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
S201	2-00065	12MM TACT SWITC	
S202	2-00065	12MM TACT SWITC	
S203	2-00065	12MM TACT SWITC	
S204	2-00065	12MM TACT SWITC	
S205	2-00065	12MM TACT SWITC	
S206	2-00065	12MM TACT SWITC	
U101	3-01497	ATMEGA162-16AI	
U102	3-01498	74ABT16245CMTD	
U201	3-01215	MAX1234EGI	
U202	3-00741	74HC04	74HC04, Hex Inverter

U203	3-01216	HEF4794BTD	
U301	3-00939	LM4882M	LM4882M, AUDIO POWER AMP
U360	3-01977	LM317MABDTG	
U361	3-01977	LM317MABDTG	
U362	3-01977	LM317MABDTG	
U370	3-01235	74LCX04M	
Z0	7-02036	BLK CAP	
Z1	7-02037	RED CAP	
Z2	1-01252	050R33-075B	

GPIB option *(assembly 289)*

C111	5-00601	0.1UF - 16V X7R	
C112	5-00601	0.1UF - 16V X7R	
C113	5-00601	0.1UF - 16V X7R	
C114	5-00601	0.1UF - 16V X7R	
C121	5-00601	0.1UF - 16V X7R	
C122	5-00601	0.1UF - 16V X7R	
C123	5-00601	0.1UF - 16V X7R	
C124	5-00601	0.1UF - 16V X7R	
C131	5-00601	0.1UF - 16V X7R	
C132	5-00601	0.1UF - 16V X7R	
C133	5-00601	0.1UF - 16V X7R	
C134	5-00601	0.1UF - 16V X7R	
C135	5-00601	0.1UF - 16V X7R	
C136	5-00601	0.1UF - 16V X7R	
C137	5-00601	0.1UF - 16V X7R	
C138	5-00601	0.1UF - 16V X7R	
C139	5-00601	0.1UF - 16V X7R	
C140	5-00601	0.1UF - 16V X7R	
C150	5-00601	0.1UF - 16V X7R	
C161	5-00601	0.1UF - 16V X7R	
C162	5-00601	0.1UF - 16V X7R	
C163	5-00601	0.1UF - 16V X7R	
J140	1-00160	IEEE488/STAND.	Connector, IEEE488, Standard, R/A, Female
J160	1-00251	10 PIN DIL	Header, DIM, Locking Clips
J202	1-01291	40 PIN	
PC1	7-01892	PTC240, GPIB	
R131	4-01406	0	Resistor, Thick Film, 5%, 300 ppm, SMT
U110	3-01236	74LCX16245MTD	
U120	3-01236	74LCX16245MTD	
U130	3-01019	TNT4882-BQ	GPIB
U140	3-01742	74VCX245WM	
U150	3-00741	74HC04	74HC04, Hex Inverter
U160	3-01743	ISPGAL22V10AV	
Y101	6-00756	40 MHZ	
Z0	0-00500	554043-1	
Z1	7-01736	PTC BRKT	

2-channel thermistor/RTD/diode reader *(assembly 310)*

Z0	0-00079	4-40X3/16 M/F	
Z1	0-00089	4	
J300	1-00071	8 PIN, WHITE	Header, SIM, Polarized
JDR121	1-00234	96 PIN RT ANGLE	3 Row, Right Angle Mount
J200	1-00281	10 PIN DI	Header, DIM
J500	1-00281	10 PIN DI	Header, DIM
Z2	1-00468	8 PIN, 24AWG/WH	Non board mount, Female, 24 AWG
Z3	1-01067	9 PIN	
Z4	1-01068	9 PIN	
J260	1-01331	1201-066	
D111	3-00011	RED	LED, T1 Package, 3mm diameter

D351	3-00489	1N5232	1N5232, 5.6V, 500 mW, DO-35 ZENER DIODE
U610	3-00542	AD587JR	High precision 10 volt reference
U140	3-00656	LM34DM	LM34DM, TEMP SENSOR
U300	3-00663	74HC08	74HC08, Quad 2-Input AND Gate
U302	3-00743	74HC138D	74HC138, 3-to-8 line decoder/demultiplexer; inverting
U340	3-00787	74HC595	74HC595, 8 Bit Serial Input, Parallel Output Shift Register
U341	3-00787	74HC595	74HC595, 8 Bit Serial Input, Parallel Output Shift Register
U342	3-00787	74HC595	74HC595, 8 Bit Serial Input, Parallel Output Shift Register
U345	3-00787	74HC595	74HC595, 8 Bit Serial Input, Parallel Output Shift Register
U350	3-00814	78M05	78M05
U233	3-01133	TL431CD5	TL431C, Adjustable Shunt Voltage Regulator, 100 mA, SOT23-5
U380	3-01133	TL431CD5	TL431C, Adjustable Shunt Voltage Regulator, 100 mA, SOT23-5
U385	3-01133	TL431CD5	TL431C, Adjustable Shunt Voltage Regulator, 100 mA, SOT23-5
U533	3-01133	TL431CD5	TL431C, Adjustable Shunt Voltage Regulator, 100 mA, SOT23-5
U360	3-01175	78M15	
U370	3-01176	79M15	
U430	3-01302	NUD3105DMT1	
U432	3-01302	NUD3105DMT1	
U434	3-01302	NUD3105DMT1	
U436	3-01302	NUD3105DMT1	
U438	3-01302	NUD3105DMT1	
U440	3-01302	NUD3105DMT1	
U442	3-01302	NUD3105DMT1	
U444	3-01302	NUD3105DMT1	
K430	3-01316	G6SK-2F-DC5	
K431	3-01316	G6SK-2F-DC5	
K432	3-01316	G6SK-2F-DC5	
K433	3-01316	G6SK-2F-DC5	
K434	3-01316	G6SK-2F-DC5	
K435	3-01316	G6SK-2F-DC5	
K436	3-01316	G6SK-2F-DC5	
K437	3-01316	G6SK-2F-DC5	
K438	3-01316	G6SK-2F-DC5	
K439	3-01316	G6SK-2F-DC5	
K440	3-01316	G6SK-2F-DC5	
K441	3-01316	G6SK-2F-DC5	
K442	3-01316	G6SK-2F-DC5	
K443	3-01316	G6SK-2F-DC5	
K444	3-01316	G6SK-2F-DC5	
K445	3-01316	G6SK-2F-DC5	
D205	3-01319	MMBD1503A	
D206	3-01319	MMBD1503A	
D207	3-01319	MMBD1503A	
D208	3-01319	MMBD1503A	
D501	3-01319	MMBD1503A	
D502	3-01319	MMBD1503A	
D503	3-01319	MMBD1503A	
D504	3-01319	MMBD1503A	
D505	3-01319	MMBD1503A	
D506	3-01319	MMBD1503A	
D507	3-01319	MMBD1503A	
D508	3-01319	MMBD1503A	
ISO310	3-01320	HCPL-2630	
ISO311	3-01320	HCPL-2630	
ISO330	3-01320	HCPL-2630	
D641	3-01357	MMBZ5230	4.7V ZENER 5%
D741	3-01357	MMBZ5230	4.7V ZENER 5%
U620	3-01386	DG408DY	Analog mux, 8-to-1, +/-15V okay, TTL compat.
U720	3-01386	DG408DY	Analog mux, 8-to-1, +/-15V okay, TTL compat.
U650	3-01398	OPA2131UJ	FET-input dual opamp, 4 MHz GBW,
U750	3-01398	OPA2131UJ	FET-input dual opamp, 4 MHz GBW,
U270	3-01469	MAX6250BCSA	+5V Reference
U640	3-01469	MAX6250BCSA	+5V Reference
U740	3-01469	MAX6250BCSA	+5V Reference

U120	3-01498	74ABT16245CMTD	
U290	3-01500	LTC2440CGN	
U590	3-01500	LTC2440CGN	
U210	3-01695	MAX4635EUB+	
U250	3-01695	MAX4635EUB+	
U281	3-01695	MAX4635EUB+	
U510	3-01695	MAX4635EUB+	
U550	3-01695	MAX4635EUB+	
U581	3-01695	MAX4635EUB+	
U110	3-01696	ATMEGA64-16AC	
U260	3-01900	LT6012ACS#PBF	Quad unity-stable opamp, rail/rail output
U560	3-01900	LT6012ACS#PBF	Quad unity-stable opamp, rail/rail output
U230	3-01941	MAX339CSE	
U530	3-01941	MAX339CSE	
U630	3-01941	MAX339CSE	
U670	3-01941	MAX339CSE	
U730	3-01941	MAX339CSE	
U770	3-01941	MAX339CSE	
U410	3-01944	74HC238	
U420	3-01944	74HC238	
U660	3-01945	INA121UA	
U760	3-01945	INA121UA	
U280	3-01963	MAX4674ESE+	
U580	3-01963	MAX4674ESE+	
R641	4-00016	10K	Pot, Multi Turn, Side Adjust
R741	4-00016	10K	Pot, Multi Turn, Side Adjust
R633	4-00139	10.0M	Resistor, Metal Film, 1/8W, 1%, 50PPM
R733	4-00139	10.0M	Resistor, Metal Film, 1/8W, 1%, 50PPM
RN292	4-00442	1.2K 1206 MINI	Network, scalloped edge 1206,
RN592	4-00442	1.2K 1206 MINI	Network, scalloped edge 1206,
R432	4-00678	6.040K	Resistor, Metal Film, 1/8W, 0.1%, 5ppm
R440	4-00678	6.040K	Resistor, Metal Film, 1/8W, 0.1%, 5ppm
RN310	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN312	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN330	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN332	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN670	4-00910	1.0KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN770	4-00910	1.0KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN291	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN345	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN591	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN200	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN201	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN202	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN203	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN500	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN501	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN502	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN503	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
R631	4-01117	1.00K	Resistor, Thin Film, 1%, 50 ppm, MELF
R731	4-01117	1.00K	Resistor, Thin Film, 1%, 50 ppm, MELF
R632	4-01309	100K	Resistor, Thin Film, 1%, 50 ppm, MELF
R732	4-01309	100K	Resistor, Thin Film, 1%, 50 ppm, MELF
R112	4-01466	300	Resistor, Thick Film, 5%, 200 ppm, SMT
RN111	4-01707	47KX4D	
RN112	4-01707	47KX4D	
RN121	4-01707	47KX4D	
R430	4-01733	604	Resistor, Metal Film, 1/8W, 0.1%, 5ppm
R438	4-01733	604	Resistor, Metal Film, 1/8W, 0.1%, 5ppm
R113	4-01869	1.0K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R643	4-01869	1.0K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R651	4-01869	1.0K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R743	4-01869	1.0K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R751	4-01869	1.0K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip

R644	4-01917	100K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R682	4-01917	100K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R744	4-01917	100K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R782	4-01917	100K	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R281	4-01948	2.0M	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R282	4-01948	2.0M	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R581	4-01948	2.0M	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R582	4-01948	2.0M	Resistor, Thick Film, 5%, 200 ppm, 1/16W, 0603 Chip
R617	4-02061	100	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R652	4-02061	100	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R717	4-02061	100	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R752	4-02061	100	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R616	4-02090	200	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R618	4-02090	200	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R716	4-02090	200	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R718	4-02090	200	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R383	4-02107	301	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R385	4-02107	301	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R615	4-02128	499	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R715	4-02128	499	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R613	4-02157	1.00K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R614	4-02157	1.00K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R713	4-02157	1.00K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R714	4-02157	1.00K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R612	4-02186	2.00K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R712	4-02186	2.00K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R233	4-02195	2.49K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R533	4-02195	2.49K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R381	4-02217	4.22K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R386	4-02217	4.22K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R382	4-02224	4.99K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R387	4-02224	4.99K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R611	4-02224	4.99K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R711	4-02224	4.99K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R260	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R261	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R560	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R561	4-02253	10.0K	Resistor, Thin Film, 1%, 50 ppm, 1/16W 0603 Chip
R434	4-02483	60K	
R442	4-02483	60K	
R250	4-02519	MAX5491WC30000	
R251	4-02519	MAX5491WC30000	
R550	4-02519	MAX5491WC30000	
R551	4-02519	MAX5491WC30000	
R436	4-02520	634K / 5PPM	
R444	4-02520	634K / 5PPM	
R642	4-02524	500K / 0.1%	
R742	4-02524	500K / 0.1%	
C352	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C353	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C362	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C363	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C372	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C373	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C612	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C271	5-00526	22U-T16	SMD TANTALUM, C-Case
C111	5-00601	0.1UF - 16V X7R	
C112	5-00601	0.1UF - 16V X7R	
C113	5-00601	0.1UF - 16V X7R	
C121	5-00601	0.1UF - 16V X7R	
C122	5-00601	0.1UF - 16V X7R	
C123	5-00601	0.1UF - 16V X7R	
C124	5-00601	0.1UF - 16V X7R	
C210	5-00601	0.1UF - 16V X7R	

C230	5-00601	0.1UF - 16V X7R	
C231	5-00601	0.1UF - 16V X7R	
C250	5-00601	0.1UF - 16V X7R	
C260	5-00601	0.1UF - 16V X7R	
C261	5-00601	0.1UF - 16V X7R	
C270	5-00601	0.1UF - 16V X7R	
C280	5-00601	0.1UF - 16V X7R	
C281	5-00601	0.1UF - 16V X7R	
C300	5-00601	0.1UF - 16V X7R	
C302	5-00601	0.1UF - 16V X7R	
C310	5-00601	0.1UF - 16V X7R	
C311	5-00601	0.1UF - 16V X7R	
C330	5-00601	0.1UF - 16V X7R	
C331	5-00601	0.1UF - 16V X7R	
C332	5-00601	0.1UF - 16V X7R	
C340	5-00601	0.1UF - 16V X7R	
C341	5-00601	0.1UF - 16V X7R	
C342	5-00601	0.1UF - 16V X7R	
C345	5-00601	0.1UF - 16V X7R	
C411	5-00601	0.1UF - 16V X7R	
C421	5-00601	0.1UF - 16V X7R	
C510	5-00601	0.1UF - 16V X7R	
C530	5-00601	0.1UF - 16V X7R	
C531	5-00601	0.1UF - 16V X7R	
C550	5-00601	0.1UF - 16V X7R	
C560	5-00601	0.1UF - 16V X7R	
C561	5-00601	0.1UF - 16V X7R	
C580	5-00601	0.1UF - 16V X7R	
C581	5-00601	0.1UF - 16V X7R	
C611	5-00601	0.1UF - 16V X7R	
C620	5-00601	0.1UF - 16V X7R	
C621	5-00601	0.1UF - 16V X7R	
C622	5-00601	0.1UF - 16V X7R	
C630	5-00601	0.1UF - 16V X7R	
C631	5-00601	0.1UF - 16V X7R	
C640	5-00601	0.1UF - 16V X7R	
C650	5-00601	0.1UF - 16V X7R	
C651	5-00601	0.1UF - 16V X7R	
C660	5-00601	0.1UF - 16V X7R	
C661	5-00601	0.1UF - 16V X7R	
C670	5-00601	0.1UF - 16V X7R	
C671	5-00601	0.1UF - 16V X7R	
C672	5-00601	0.1UF - 16V X7R	
C682	5-00601	0.1UF - 16V X7R	
C720	5-00601	0.1UF - 16V X7R	
C721	5-00601	0.1UF - 16V X7R	
C722	5-00601	0.1UF - 16V X7R	
C730	5-00601	0.1UF - 16V X7R	
C731	5-00601	0.1UF - 16V X7R	
C740	5-00601	0.1UF - 16V X7R	
C750	5-00601	0.1UF - 16V X7R	
C751	5-00601	0.1UF - 16V X7R	
C760	5-00601	0.1UF - 16V X7R	
C761	5-00601	0.1UF - 16V X7R	
C770	5-00601	0.1UF - 16V X7R	
C771	5-00601	0.1UF - 16V X7R	
C772	5-00601	0.1UF - 16V X7R	
C782	5-00601	0.1UF - 16V X7R	
C295	5-00654	.01UF X 4	
C595	5-00654	.01UF X 4	
C305	5-00716	100P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C306	5-00716	100P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C307	5-00716	100P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C200	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603

C201	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C202	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C203	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C204	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C205	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C206	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C500	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C501	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C502	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C503	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C504	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C505	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C506	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C641	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C741	5-00740	1000P	Capacitor, Mono, 50V, 0.25pF or 5%, NPO, 0603
C652	5-00752	10000P	Capacitor, Mono, 50V,+/-10%, X7R, 0603
C752	5-00752	10000P	Capacitor, Mono, 50V,+/-10%, X7R, 0603
C272	5-00841	1UF/16V/ X5R	
C290	5-00841	1UF/16V/ X5R	
C291	5-00841	1UF/16V/ X5R	
C292	5-00841	1UF/16V/ X5R	
C293	5-00841	1UF/16V/ X5R	
C294	5-00841	1UF/16V/ X5R	
C590	5-00841	1UF/16V/ X5R	
C591	5-00841	1UF/16V/ X5R	
C592	5-00841	1UF/16V/ X5R	
C593	5-00841	1UF/16V/ X5R	
C594	5-00841	1UF/16V/ X5R	
C642	5-00841	1UF/16V/ X5R	
C742	5-00841	1UF/16V/ X5R	
L300	6-01030	FT-87-W	
PCB	7-02172	PTC100 PCB	
PCB	7-02172	PTC100 PCB	

100W DC output card (assembly 206)

C111	5-00601	0.1UF - 16V X7R	
C112	5-00601	0.1UF - 16V X7R	
C113	5-00601	0.1UF - 16V X7R	
C121	5-00601	0.1UF - 16V X7R	
C122	5-00601	0.1UF - 16V X7R	
C123	5-00601	0.1UF - 16V X7R	
C124	5-00601	0.1UF - 16V X7R	
C203	5-00606	1U / 100V	
C204	5-00389	1500P	Capacitor, Mono, 50V, 5%, NPO, 1206
C205	5-00850	0.33U PEN 16V	
C211	5-00607	10U / 50V SMT	
C212	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C213	5-00840	10 UF / X7S	
C214	5-00840	10 UF / X7S	
C216	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C217	5-00298	.01U	Capacitor, Mono, 50V, 10%, X7R, 1206
C220	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C222	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C223	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C232	5-00629	1000P X 4	
C233	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C234	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C235	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C240	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C245	5-00627	0.1U X 4	
C250	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C251	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206

C261	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C270	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C271	5-00798	2.2U	
C272	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C273	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C274	5-00654	.01UF X 4	
C275	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C276	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C277	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C281	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C290	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C311	5-00035	47U	Capacitor, Electrolytic, 25V, 20%, Rad
C312	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C313	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C320	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C321	5-00035	47U	Capacitor, Electrolytic, 25V, 20%, Rad
C323	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C331	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C500	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C501	5-00601	0.1UF - 16V X7R	
C510	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C511	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C530	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C540	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C550	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
D111	3-00011	RED	LED, T1 Package, 3mm diameter
D211	3-00403	1N459A	1N459A, 175V, 0.5A, LOW LEAKAGE DIODE
D212	3-02066	12CWQ10FNPBF	
D214	3-00626	MUR1100E	MUR1100E, 1000V, 1A ULTRA FAST DIODE
D221	3-00896	BAV99	BAV99, DUAL SERIES DIODE, 70V BREAKDOWN
D251	3-00457	1N5241B	1N5241B, 11V, 500mW, DO-35 ZENER DIODE
D252	3-00457	1N5241B	1N5241B, 11V, 500mW, DO-35 ZENER DIODE
D253	3-00457	1N5241B	1N5241B, 11V, 500mW, DO-35 ZENER DIODE
F221	6-00644	1A 60V	
F222	6-00644	1A 60V	
ISO500	3-00446	6N137	Hi Speed Optocoupler
ISO510	3-01320	HCPL-2630	
ISO511	3-01320	HCPL-2630	
ISO530	3-01320	HCPL-2630	
J111	1-00251	10 PIN DIL	Header, DIM, Locking Clips
JDR121	1-00234	96 PIN RT ANGLE	3 Row, Right Angle Mount
L211	6-01005	22 UH	
L310	6-00684	10UH	Inductor, SMD, Type R, 23MHz, 240mA, 10%, 1210
L311	0-00000	UNDECIDED PART	
L320	6-00684	10UH	Inductor, SMD, Type R, 23MHz, 240mA, 10%, 1210
L321	6-00174	6611 TYPE 43	Ferite Bead, Thru-hole, Type 43
L351	6-00174	6611 TYPE 43	Ferite Bead, Thru-hole, Type 43
PCB1	7-02177	PTC431 2A DC OU	
Q210	3-02065	IRLR3110ZPBF	
Q220	3-01254	BSS123LT1	
Q233	3-01254	BSS123LT1	
Q234	3-01254	BSS123LT1	
Q235	3-01254	BSS123LT1	
Q251	3-02075	IRF6218S	
Q252	3-02075	IRF6218S	
Q253	3-02075	IRF6218S	
Q330	3-02065	IRLR3110ZPBF	
R112	4-01466	300	Resistor, Thick Film, 5%, 200 ppm, SMT
R202	4-01186	5.23K	Resistor, Thin Film, 1%, 50 ppm, MELF
R203	4-01309	100K	Resistor, Thin Film, 1%, 50 ppm, MELF
R206	4-01186	5.23K	Resistor, Thin Film, 1%, 50 ppm, MELF
R207	4-01309	100K	Resistor, Thin Film, 1%, 50 ppm, MELF
R208	4-00436	0.1	Resistor, Wire-wound
R211	4-01157	2.61K	Resistor, Thin Film, 1%, 50 ppm, MELF

R212	4-01292	66.5K	Resistor, Thin Film, 1%, 50 ppm, MELF
R213	4-01130	1.37K	Resistor, Thin Film, 1%, 50 ppm, MELF
R214	4-01029	121	Resistor, Thin Film, 1%, 50 ppm, MELF
R215	4-01285	56.2K	Resistor, Thin Film, 1%, 50 ppm, MELF
R216	4-01096	604	Resistor, Thin Film, 1%, 50 ppm, MELF
R217	4-02546	0.01 ohm	
R222	4-00645	4.7K	Resistor, Thick Film, 1/8W, 5%, 1206
R223	4-00645	4.7K	Resistor, Thick Film, 1/8W, 5%, 1206
R224	4-00645	4.7K	Resistor, Thick Film, 1/8W, 5%, 1206
R225	4-00645	4.7K	Resistor, Thick Film, 1/8W, 5%, 1206
R226	4-01151	2.26K	Resistor, Thin Film, 1%, 50 ppm, MELF
R231	4-01175	4.02K	Resistor, Thin Film, 1%, 50 ppm, MELF
R232	4-01155	2.49K	Resistor, Thin Film, 1%, 50 ppm, MELF
R233	4-01117	1.00K	Resistor, Thin Film, 1%, 50 ppm, MELF
R234	4-01050	200	Resistor, Thin Film, 1%, 50 ppm, MELF
R235	4-01175	4.02K	Resistor, Thin Film, 1%, 50 ppm, MELF
R236	4-01155	2.49K	Resistor, Thin Film, 1%, 50 ppm, MELF
R237	4-01117	1.00K	Resistor, Thin Film, 1%, 50 ppm, MELF
R238	4-01050	200	Resistor, Thin Film, 1%, 50 ppm, MELF
R239	4-01175	4.02K	Resistor, Thin Film, 1%, 50 ppm, MELF
R240	4-01155	2.49K	Resistor, Thin Film, 1%, 50 ppm, MELF
R241	4-01117	1.00K	Resistor, Thin Film, 1%, 50 ppm, MELF
R242	4-01050	200	Resistor, Thin Film, 1%, 50 ppm, MELF
R251	4-02525	0.1 ohm	
R252	4-02526	1 ohm	
R253	4-00925	10	Resistor, Thin Film, 1%, 50 ppm, MELF
R255	4-01050	200	Resistor, Thin Film, 1%, 50 ppm, MELF
R256	4-01050	200	Resistor, Thin Film, 1%, 50 ppm, MELF
R257	4-01050	200	Resistor, Thin Film, 1%, 50 ppm, MELF
R263	4-01404	976K	Resistor, Thin Film, 1%, 50 ppm, MELF
R264	4-01296	73.2K	Resistor, Thin Film, 1%, 50 ppm, MELF
R271	4-01670	20K 1%	Divider, Thin Film, 10.0K x 2, 0.1W, 1%, 2ppm ratio, SOT23
R273	4-01242	20.0K	Resistor, Thin Film, 1%, 50 ppm, MELF
R274	4-01117	1.00K	Resistor, Thin Film, 1%, 50 ppm, MELF
R331	4-02547	120 OHM / 10W	
R332	4-00645	4.7K	Resistor, Thick Film, 1/8W, 5%, 1206
R501	4-01471	470	Resistor, Thick Film, 5%, 200 ppm, SMT
R502	4-01471	470	Resistor, Thick Film, 5%, 200 ppm, SMT
RN111	4-01707	47KX4D	
RN112	4-01707	47KX4D	
RN113	4-00910	1.0KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN121	4-01707	47KX4D	
RN272	4-01764	10X4D	
RN273	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN291	4-00912	10KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN292	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN510	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN512	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN530	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN532	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
U110	3-01696	ATMEGA64-16AC	
U120	3-01498	74ABT16245CMTD	
U200	3-00656	LM34DM	LM34DM, TEMP SENSOR
U210	3-02064	LM3478MM/NOPB	
U220	3-00773	LM358	LM358 Dual low power 1 MHz GBW op amp
U221	3-01133	TL431CD5	TL431C, Adjustable Shunt Voltage Regulator, 100 mA, SOT23-5
U233	3-01821	LTC6102HMS	
U234	3-01821	LTC6102HMS	
U235	3-01821	LTC6102HMS	
U240	3-00675	LTC1655	16 bit Rail-Rail DAC
U250	3-01257	LMC6484AIM	
U260	3-01386	DG408DY	Analog mux, 8-to-1, +/-15V okay, TTL compat.
U270	3-01257	LMC6484AIM	
U271	3-01186	MAX6241BCSA	

U280	3-01258	LTC2433-1CMS	
U290	3-01366	DG333ADW	Quad SPDT 175nsec, 25ohms-ONres
U310	3-00814	78M05	78M05,
U320	3-01979	79M05CDT/RK	
U500	3-00663	74HC08	74HC08, Quad 2-Input AND Gate
U540	3-00787	74HC595	74HC595, 8 Bit Serial Input, Parallel Output Shift Register
U550	3-00743	74HC138D	74HC138, 3-to-8 line decoder/demultiplexer; inverting
Z0	0-00772	1.5 WIRE	
Z1	0-01318	7109DG	
Z2	0-01349	8-32 X 5/8 PP	

Analog I/O card *(assembly 297)*

C101	5-00601	0.1UF - 16V X7R	
C102	5-00601	0.1UF - 16V X7R	
C103	5-00601	0.1UF - 16V X7R	
C105	5-00601	0.1UF - 16V X7R	
C106	5-00601	0.1UF - 16V X7R	
C107	5-00601	0.1UF - 16V X7R	
C108	5-00601	0.1UF - 16V X7R	
C201	5-00470	2.2U/T16	SMD TANTALUM, Y-Case
C202	5-00525	1U	CAP 1UF 25V CERAMIC Y5V 1206 +80/ -20%
C203	5-00470	2.2U/T16	SMD TANTALUM, Y-Case
C204	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C205	5-00471	10U/T16	SMD TANTALUM, C-Case
C206	5-00527	.47U	CAP .47UF 16V CERAMIC X7R 1206
C207	5-00601	0.1UF - 16V X7R	
C208	5-00601	0.1UF - 16V X7R	
C209	5-00527	.47U	CAP .47UF 16V CERAMIC X7R 1206
C210	5-00527	.47U	CAP .47UF 16V CERAMIC X7R 1206
C211	5-00527	.47U	CAP .47UF 16V CERAMIC X7R 1206
C212	5-00601	0.1UF - 16V X7R	
C213	5-00601	0.1UF - 16V X7R	
C214	5-00601	0.1UF - 16V X7R	
C215	5-00601	0.1UF - 16V X7R	
C216	5-00627	0.1U X 4	
C217	5-00601	0.1UF - 16V X7R	
C225	5-00601	0.1UF - 16V X7R	
C226	5-00601	0.1UF - 16V X7R	
C227	5-00381	330P	Capacitor, Mono, 50V, 5%, NPO, 1206
C232	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C233	5-00471	10U/T16	SMD TANTALUM, C-Case
C302	5-00601	0.1UF - 16V X7R	
C310	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C311	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C330	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C331	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C340	5-00601	0.1UF - 16V X7R	
C351	5-00035	47U	Capacitor, Electrolytic, 25V, 20%, Rad
C352	5-00519	.33U/T35	SMD TANTALUM, Y-Case
C353	5-00513	1U-16V A-CASE	SMT Tantalum, 16V, A-case (1206, but NEEDS POLARITY mark)
C361	5-00035	47U	Capacitor, Electrolytic, 25V, 20%, Rad
C362	5-00519	.33U/T35	SMD TANTALUM, Y-Case
C363	5-00513	1U-16V A-CASE	SMT Tantalum, 16V, A-case (1206, but NEEDS POLARITY mark)
C371	5-00035	47U	Capacitor, Electrolytic, 25V, 20%, Rad
C372	5-00519	.33U/T35	SMD TANTALUM, Y-Case
C373	5-00513	1U-16V A-CASE	SMT Tantalum, 16V, A-case (1206, but NEEDS POLARITY mark)
D101	3-00011	RED	LED, T1 Package, 3mm diameter
D201	3-00544	BAV70LT1-ROHS	BAV70LT1
D202	3-00544	BAV70LT1-ROHS	BAV70LT1
D203	3-00544	BAV70LT1-ROHS	BAV70LT1
D204	3-00544	BAV70LT1-ROHS	BAV70LT1
D246	3-01384	MMBZ5232BLT1	5.6V Zener

D301	3-01880	SMBJ12CA	
D302	3-01880	SMBJ12CA	
D303	3-01880	SMBJ12CA	
D304	3-01880	SMBJ12CA	
D341	3-00011	RED	LED, T1 Package, 3mm diameter
D342	3-00011	RED	LED, T1 Package, 3mm diameter
D343	3-00011	RED	LED, T1 Package, 3mm diameter
D344	3-00011	RED	LED, T1 Package, 3mm diameter
F301	6-00773	1206L020	
F302	6-00773	1206L020	
F303	6-00773	1206L020	
F304	6-00773	1206L020	
IS310	3-01320	HCPL-2630	
IS311	3-01320	HCPL-2630	
IS330	3-01320	HCPL-2630	
J101	1-00251	10 PIN DIL	Header, DIM, Locking Clips
J301	1-00233	RT ANGLE	BNC, PCB Panel Mount, Right Angle, Isolated
J302	1-00233	RT ANGLE	BNC, PCB Panel Mount, Right Angle, Isolated
J303	1-00233	RT ANGLE	BNC, PCB Panel Mount, Right Angle, Isolated
J304	1-00233	RT ANGLE	BNC, PCB Panel Mount, Right Angle, Isolated
JD101	1-00234	96 PIN RT ANGLE	3 Row, Right Angle Mount
L351	6-00174	6611 TYPE 43	Ferite Bead, Thru-hole, Type 43
L352	6-00684	10UH	Inductor, SMD, Type R, 23MHz, 240mA, 10%, 1210
L361	6-00174	6611 TYPE 43	Ferite Bead, Thru-hole, Type 43
L362	6-00684	10UH	Inductor, SMD, Type R, 23MHz, 240mA, 10%, 1210
L371	6-00174	6611 TYPE 43	Ferite Bead, Thru-hole, Type 43
L372	6-00684	10UH	Inductor, SMD, Type R, 23MHz, 240mA, 10%, 1210
PC1	7-01709	PTC	
R101	4-01466	300	Resistor, Thick Film, 5%, 200 ppm, SMT
R201	4-01230	15.0K	Resistor, Thin Film, 1%, 50 ppm, MELF
R202	4-01213	10.0K	Resistor, Thin Film, 1%, 50 ppm, MELF
R203	4-01213	10.0K	Resistor, Thin Film, 1%, 50 ppm, MELF
R204	4-01155	2.49K	Resistor, Thin Film, 1%, 50 ppm, MELF
R205	4-01213	10.0K	Resistor, Thin Film, 1%, 50 ppm, MELF
R206	4-01155	2.49K	Resistor, Thin Film, 1%, 50 ppm, MELF
R207	4-01213	10.0K	Resistor, Thin Film, 1%, 50 ppm, MELF
R208	4-01155	2.49K	Resistor, Thin Film, 1%, 50 ppm, MELF
R209	4-01213	10.0K	Resistor, Thin Film, 1%, 50 ppm, MELF
R210	4-01155	2.49K	Resistor, Thin Film, 1%, 50 ppm, MELF
R227	4-01163	3.01K	Resistor, Thin Film, 1%, 50 ppm, MELF
R228	4-01117	1.00K	Resistor, Thin Film, 1%, 50 ppm, MELF
R230	4-01139	1.69K	Resistor, Thin Film, 1%, 50 ppm, MELF
R231	4-01110	845	Resistor, Thin Film, 1%, 50 ppm, MELF
R234	4-01156	2.55K	Resistor, Thin Film, 1%, 50 ppm, MELF
RN101	4-01704	100Kx4D 5%	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN102	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN103	4-00910	1.0KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN105	4-01707	47KX4D	
RN206	4-00910	1.0KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN310	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN312	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN330	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN332	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN341	4-00908	270X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
U101	3-01497	ATMEGA162-16AI	
U102	3-01498	74ABT16245CMTD	
U201	3-01469	MAX6250BCSA	+5V Reference
U202	3-01499	DAC8534IPW	
U203	3-01838	MC33079D	
U204	3-01365	DG411DY	Quad SPST 25ohms-ONRes
U205	3-01838	MC33079D	
U206	3-01369	DG409DY	Diff Analog MUX 4-ch
U209	3-01500	LTC2440CGN	
U302	3-00743	74HC138D	74HC138, 3-to-8 line decoder/demultiplexer; inverting

U331	3-00749	74HC541	74HC541, Octal 3-State Buffer / Line Driver/Receiver
U340	3-00787	74HC595	74HC595, 8 Bit Shift Register w Latched 3-state Outputs
U350	3-00814	78M05	78M05
U360	3-01175	78M15	
U370	3-01176	79M15	
Z0	0-00472	1-329631-2	Jam nut for BNC connectors
Z1	0-00306	4-40X3/16PP	
Z1	0-00772	1.5 WIRE	
Z2	0-00306	4-40X3/16PP	
Z2	7-01734	PTC ANALOG.IO BRKT	

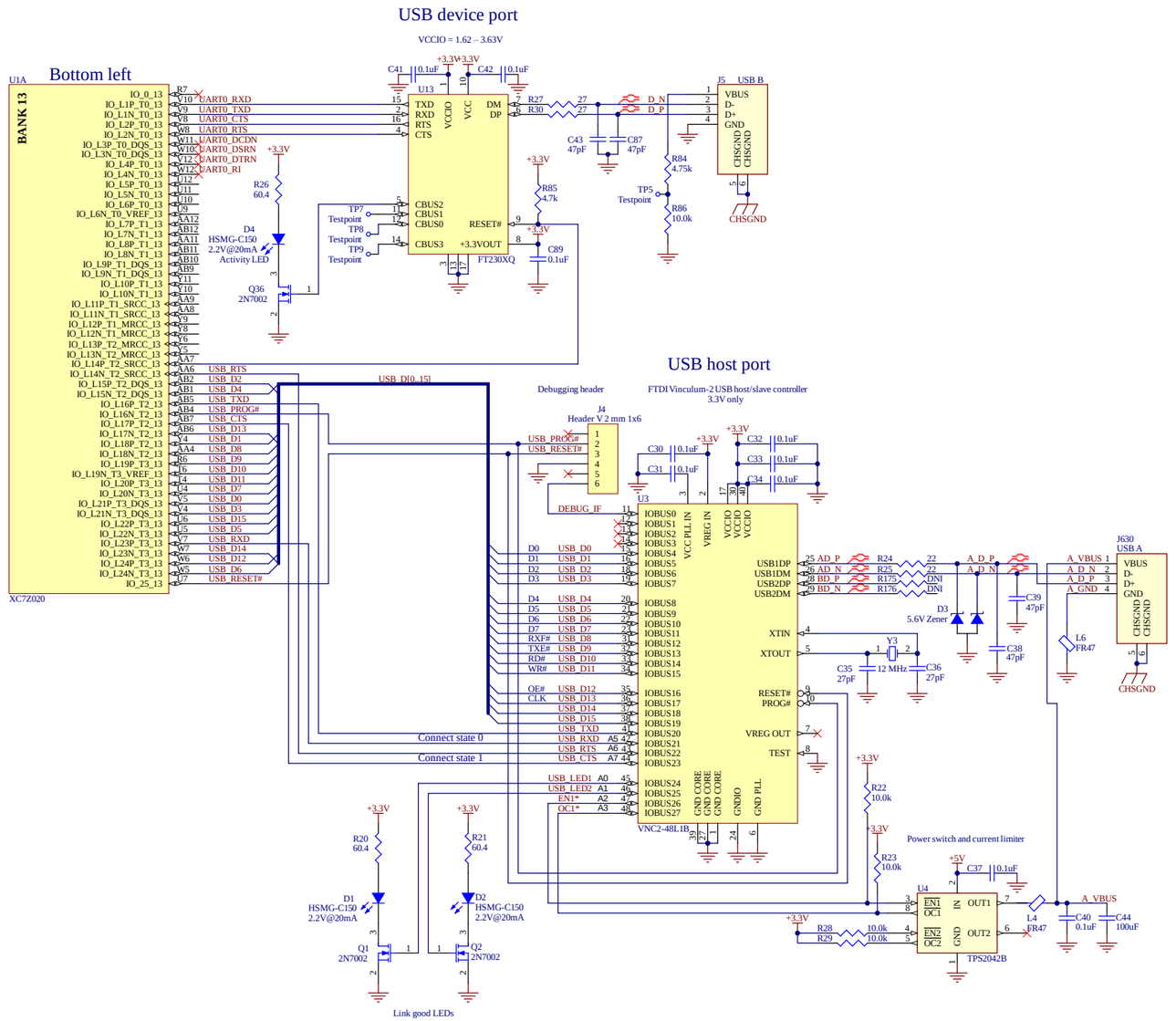
Digital I/O card *(assembly 298)*

C111	5-00601	0.1UF - 16V X7R	
C112	5-00601	0.1UF - 16V X7R	
C113	5-00601	0.1UF - 16V X7R	
C121	5-00601	0.1UF - 16V X7R	
C122	5-00601	0.1UF - 16V X7R	
C123	5-00601	0.1UF - 16V X7R	
C124	5-00601	0.1UF - 16V X7R	
C200	5-00378	180P	Capacitor, Mono, 50V, 5%, NPO, 1206
C201	5-00378	180P	Capacitor, Mono, 50V, 5%, NPO, 1206
C202	5-00378	180P	Capacitor, Mono, 50V, 5%, NPO, 1206
C203	5-00378	180P	Capacitor, Mono, 50V, 5%, NPO, 1206
C204	5-00378	180P	Capacitor, Mono, 50V, 5%, NPO, 1206
C205	5-00378	180P	Capacitor, Mono, 50V, 5%, NPO, 1206
C206	5-00378	180P	Capacitor, Mono, 50V, 5%, NPO, 1206
C207	5-00378	180P	Capacitor, Mono, 50V, 5%, NPO, 1206
C210	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C220	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C230	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C240	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C250	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C260	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C270	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C310	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C311	5-00319	10U/T35	SMD TANTALUM, D-Case
C312	5-00387	1000P	Capacitor, Mono, 50V, 5%, NPO, 1206
C313	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C314	5-00381	330P	Capacitor, Mono, 50V, 5%, NPO, 1206
C316	5-00519	.33U/T35	SMD TANTALUM, Y-Case
C360	5-00513	1U-16V A-CASE	SMT Tantalum, 16V, A-case (1206, but NEEDS POLARITY mark)
C361	5-00519	.33U/T35	SMD TANTALUM, Y-Case
C362	5-00628	22U - 35V	
C364	5-00381	330P	Capacitor, Mono, 50V, 5%, NPO, 1206
C410	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
C420	5-00299	.1U	Capacitor, Mono, 50V, 10%, X7R, 1206
D111	3-00576	RED MINI	LED, Subminiature, 1.8mm (T 3/4)
D200	3-01342	STZ5.6NT146	
D202	3-01342	STZ5.6NT146	
D204	3-01342	STZ5.6NT146	
D206	3-01342	STZ5.6NT146	
D314	3-00380	1N5248	1N5248, 18V, 500mW, DO-35 ZENER DIODE
D315	3-00926	MBR0540T1	MBR0540T1, Power Rectifier
D361	3-00010	GREEN	LED, T1 Package, 3mm diameter
D362	3-01303	B340LA-13-F	
D401	3-00806	BAV170LT1	BAV170LT1, DUAL DIODE COMMON CATHODE
D403	3-00806	BAV170LT1	BAV170LT1, DUAL DIODE COMMON CATHODE
D421	3-00011	RED	LED, T1 Package, 3mm diameter
D422	3-00011	RED	LED, T1 Package, 3mm diameter
D423	3-00011	RED	LED, T1 Package, 3mm diameter
D424	3-00011	RED	LED, T1 Package, 3mm diameter
IS230	3-01320	HCPL-2630	

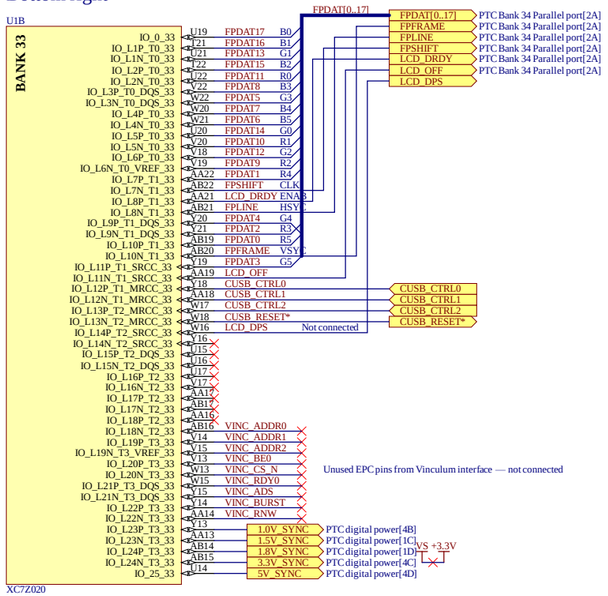
IS240	3-00446	6N137	Hi Speed Optocoupler
IS250	3-01320	HCPL-2630	
IS260	3-00446	6N137	Hi Speed Optocoupler
J111	1-00251	10 PIN DIL	Header, DIM, Locking Clips
J200	1-00371	25 PIN D RS232	DB Female, Right Angle, .318
J400	1-01090	1615490000	
JD121	1-00234	96 PIN RT ANGLE	3 Row, Right Angle Mount
K401	3-01056	24VDC DPDT	
K402	3-01056	24VDC DPDT	
K403	3-01056	24VDC DPDT	
K404	3-01056	24VDC DPDT	
PC1	7-01710	PTC	
Q411	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q412	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q413	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
Q414	3-00601	MMBT3904LT1	MMBT3904LT1, 3904 NPN
R112	4-01466	300	Resistor, Thick Film, 5%, 200 ppm, SMT
R311	4-01270	39.2K	Resistor, Thin Film, 1%, 50 ppm, MELF
R312	4-01210	9.31K	Resistor, Thin Film, 1%, 50 ppm, MELF
R313	4-01163	3.01K	Resistor, Thin Film, 1%, 50 ppm, MELF
R314	4-01009	75	Resistor, Thin Film, 1%, 50 ppm, MELF
R361	4-01466	300	Resistor, Thick Film, 5%, 200 ppm, SMT
R362	4-01455	100	Resistor, Thick Film, 5%, 200 ppm, SMT
R412	4-01406	0	Resistor, Thick Film, 5%, 300 ppm, SMT
RN111	4-01707	47KX4D	
RN112	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN113	4-00910	1.0KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN121	4-01707	47KX4D	
RN200	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN201	4-00916	47X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN202	4-01765	0KX4D	
RN231	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN232	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN251	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN252	4-00909	470X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN410	4-01707	47KX4D	
RN411	4-01707	47KX4D	
RN412	4-00911	4.7KX4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
RN421	4-00908	270X4D	Network, DIP, Isolated, 1/16W, 5%, Tiny
T300	6-00683	VP1-0190	
U110	3-01497	ATMEGA162-16AI	
U120	3-01498	74ABT16245CMTD	
U210	3-01343	74HC166D	
U220	3-00787	74HC595	74HC595, 8 Bit Shift Register w Latched 3-state Outputs
U270	3-00749	74HC541	74HC541, Octal 3-State Buffer / Line Driver/Receiver
U310	3-01322	LT1425CS	
U321	3-01460	MC7815ACD2T	7815, 3 Terminal, 15V, 1A Regulator
U360	3-00814	78M05	78M05,
U410	3-01375	74HC86AD	Quad XOR gate
U420	3-00741	74HC04	74HC04, Hex Inverter
Z0	7-01738	PTC DIG.I/O BRK	
Z1	0-00306	4-40X3/16PP	
Z2	0-00306	4-40X3/16PP	
Z3	0-01093	563002B00000	Heat sink
Z4	1-01186	1690520000	Relay connector

Schematics

Circuit board	Page count
PTC212 CPU board	9
PTC222 Backplane	3
PTC232 Front panel	3
PTC240 GPIB card	1
PTC323 2-channel thermistor/diode/RTD reader	6
PTC431 100W DC output card	3
PTC510 Analog IO card	3
PTC520 Digital IO card	4


USB device port
VCCIO = 1.62 – 3.63V
Bottom left
U1A
BANK 13
XC7Z020
U13
FT230XQ
TXD
RXD
RTS
CTS
DM
DP
RESET#
CBUS2
CBUS1
CBUS0
CBUS3
+3.3VOUT
J5 USB B
VBUS
D-
D+
GND
CHSGND
TP5 Testpoint
TP7 Testpoint
TP8 Testpoint
TP9 Testpoint
D4 HSMG-C150 2.2V@20mA Activity LED
Q36 2N7002
USB host port
Debugging header
FTDI Vinculum-2 USB host/slave controller 3.3V only
J4 Header V 2 mm 1x6
DEBUG_IF
U3
VNC2-48L1B
J630 USB A
D3 5.6V Zener
Y3 12 MHz
L6 FR47
L4 FR47
Connect state 0
Connect state 1
Link good LEDs
D1 HSMG-C150 2.2V@20mA
D2 HSMG-C150 2.2V@20mA
Q1 2N7002
Q2 2N7002
Power switch and current limiter
U4
TPS2042B
CPU card: Bank 13 USB
PTC Bank 13 USB.SchDoc
8/23/2023 4:33:42 PM
Sheet 1 of 9

Bottom right



CPU card: Bank 33 LCD

PTC Bank 33 LCD.SchDoc

8/23/2023 4:33:43 PM

Right

U1C

BANK 34

XC7Z020

PTC Bank 33 LCD[4B] FPDAT[0..17]
PTC Bank 33 LCD[4B] FPFRAME
PTC Bank 33 LCD[4B] FPLINE
PTC Bank 33 LCD[4B] FPSHIFT
PTC Bank 33 LCD[4B] LCD_DRDY
PTC Bank 33 LCD[4B] LCD_OFF

R93 DNI
R91 4.99k

PPD[0..15]

C75 0.1uF
C76 0.1uF

BPPCLK
PPCLK 5V
Only rising edge is accurate

U10 74LVC1T45

5V tolerant transceivers

U303 MC74LCX16245DT

3–10.4 ns R/W* delay to prevent data line contention

U302A 74LCX04M
U302B 74LCX04M

Write = PPR/W* = low = B to A
Read = PPR/W* high = A to B

U302E 74LCX04M

Ground unused U302 inputs

U302C 74LCX04M
U302D 74LCX04M
U302F 74LCX04M

RN301A 4.7k x 4
RN301B 4.7k x 4
RN301C 4.7k x 4
RN301D 4.7k x 4

PPA[0..7]

U304 MC74LCX16245DT

R92 4.99k

ADD[0..7]

SPI mode is used to program I/O card processors
ADD4-7 are not used

C82 0.1uF
C92 0.1uF
C94 0.1uF
C96 0.1uF
C91 0.1uF
C93 0.1uF
C95 0.1uF
C97 0.1uF

Bypass caps for transceivers

120-pin DIN connector

FPDAT[0..17]

JDR301A
JDR301B
JDR301C
DIN120_ABC/R

60 Hz PCLK

R87 0

24V supply pins are not connected on rev A+B backplanes

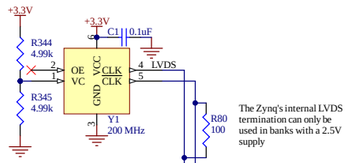
DATA[0..7]
PPD[0..15]

R2 4.7k

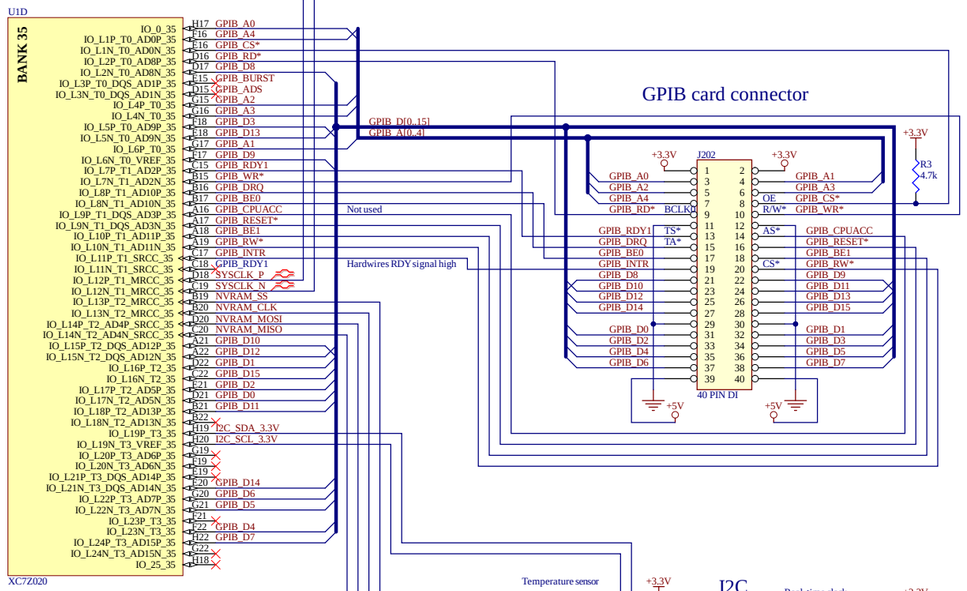
CPU card: Bank 34 Parallel port	
PTC Bank 34 Parallel port.SchDoc	
8/23/2023 4:33:44 PM	Sheet 3 of 9

Main FPGA clock

Not used. CPU clock is used instead.

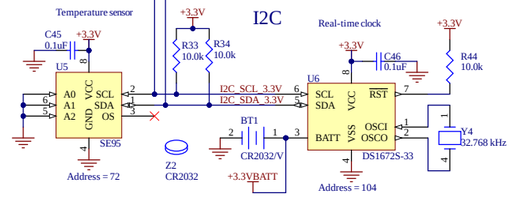


Top right



GPIB card connector

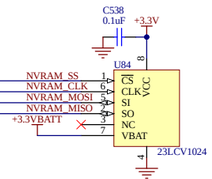
I2C



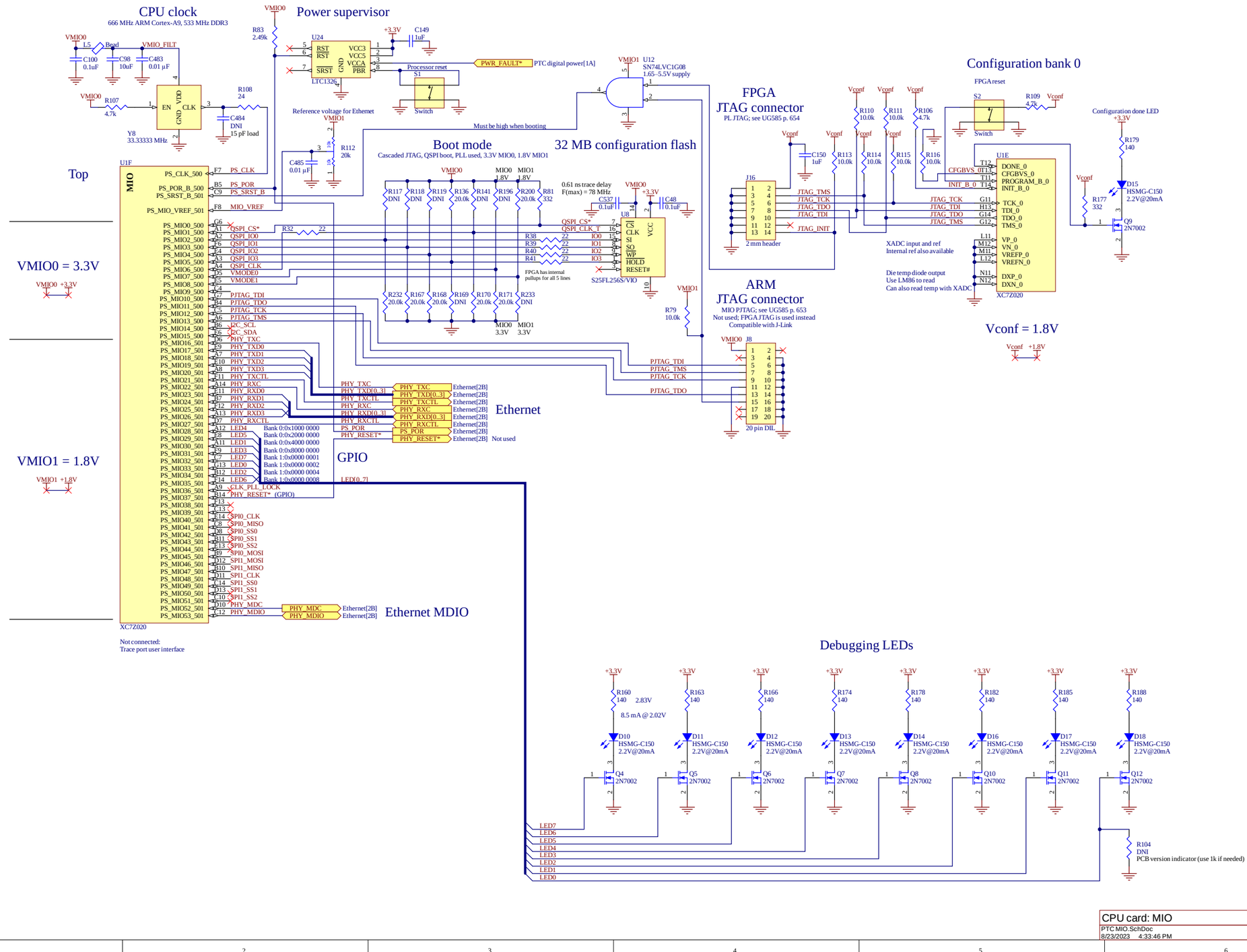
128 kB SRAM w/battery backup

(Nonvolatile RAM)

3.3–5V



CPU card: Bank 35 GPIB	
PTC Bank 35 GPIB.SchDoc	
8/23/2023 4:33:45 PM	Sheet 4 of 9


CPU clock
666 MHz ARM Cortex-A9, 533 MHz DDR3
Power supervisor
Boot mode
Cascaded JTAG, QSPI boot, PLL used, 3.3V MIO0, 1.8V MIO1
32 MB configuration flash
Top
VMIO0 = 3.3V
VMIO1 = 1.8V
U1F
MIO
XC7Z020
Not connected:
Trace port user interface
Processor reset
Switch
PWR_FAULT*
PTC digital power[1A]
Reference voltage for Ethernet
Must be high when booting
U12
SN74LVC1G08
1.65–5.5V supply
FPGA
JTAG connector
PL JTAG; see UG585 p. 654
2 mm header
ARM
JTAG connector
MIO PJTAG; see UG585 p. 653
Not used; FPGA JTAG is used instead
Compatible with J-Link
20 pin DIL
Configuration bank 0
FPGA reset
Configuration done LED
XADC input and ref
Internal ref also available
Die temp diode output
Use LM86 to read
Can also read temp with XADC
Vconf = 1.8V
Ethernet
GPIO
Ethernet MDIO
Debugging LEDs
0.61 ns trace delay
F(max) = 78 MHz
FPGA has internal
pullups for all 5 lines
S25FL256S/VIO
8.5 mA @ 2.02V
PCB version indicator (use 1k if needed)
CPU card: MIO
PTC MIO.SchDoc
8/23/2023 4:33:46 PM
Sheet 5 of 9

2 x 2 Gb (=512 MB) DDR3 SDRAM

CLK = 667 MHz

Left

U1G

DDR

XC7Z020

U9 MT41J128M16JT-125:K

U18 MT41J128M16JT-125:K

Net ties

SDRAM1_x, SDRAM2_x, and SDRAM3_x are electrically connected but are divided into separate nets so their lengths can be individually measured

SDRAM1_A0	SDRAM2_A0	SDRAM3_A0
SDRAM1_A1	SDRAM2_A1	SDRAM3_A1
SDRAM1_A2	SDRAM2_A2	SDRAM3_A2
SDRAM1_A3	SDRAM2_A3	SDRAM3_A3
SDRAM1_A4	SDRAM2_A4	SDRAM3_A4
SDRAM1_A5	SDRAM2_A5	SDRAM3_A5
SDRAM1_A6	SDRAM2_A6	SDRAM3_A6
SDRAM1_A7	SDRAM2_A7	SDRAM3_A7
SDRAM1_A8	SDRAM2_A8	SDRAM3_A8
SDRAM1_A9	SDRAM2_A9	SDRAM3_A9
SDRAM1_A10	SDRAM2_A10	SDRAM3_A10
SDRAM1_A11	SDRAM2_A11	SDRAM3_A11
SDRAM1_A12	SDRAM2_A12	SDRAM3_A12
SDRAM1_A13	SDRAM2_A13	SDRAM3_A13
SDRAM1_A14	SDRAM2_A14	SDRAM3_A14
SDRAM1_BA0	SDRAM2_BA0	SDRAM3_BA0
SDRAM1_BA1	SDRAM2_BA1	SDRAM3_BA1
SDRAM1_BA2	SDRAM2_BA2	SDRAM3_BA2

0.75V DDR termination regulator

U49 TPS51200

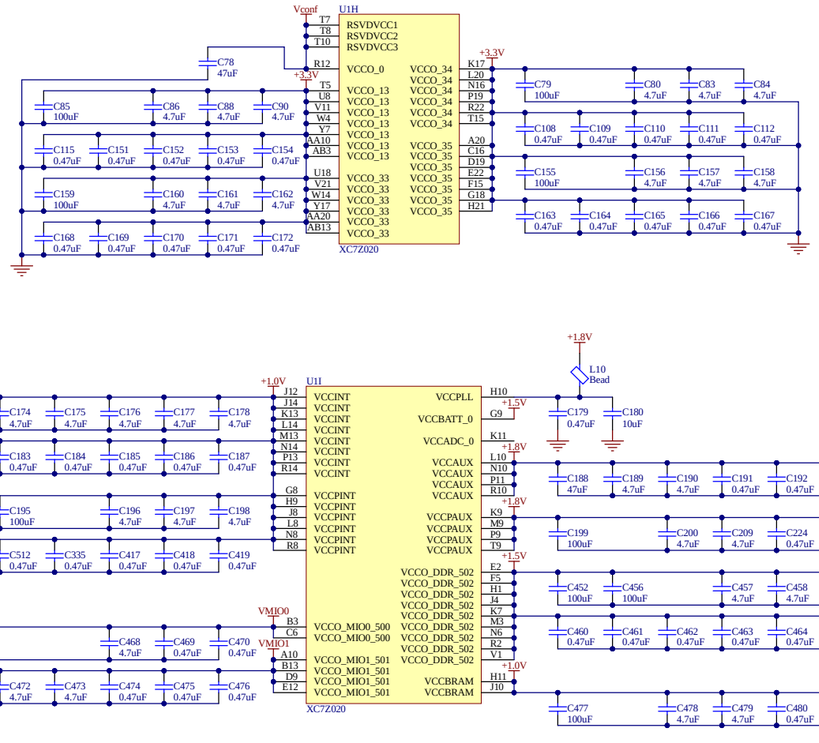
Not connected

DDR3_PGOOD

Optional decoupling capacitors

Value of R49 must be 2 x PCB impedance

Place near VOSNS pin



CPU card: FPGA power

PTC FPGA power.SchDoc

8/23/2023 4:33:48 PM

Sheet 7 of 9

Digital power supplies

PTC MIO[3A] PWR_FAULT*

For correct power-off sequencing, ON should go low before the other supply voltages start to drop.

Sequencer
Alternate: MAX16050

U2 LTC2924

	+5V	+1.0V	+1.8V	+1.5V	+3.3V
R36 / R37 / R45 / R231 / R73	20.0k	2.00k	20.0k	20.0k	33.2k
R97 / R98 / R99 / R100 / R101	3.01k	3.83k	12.4k	20.0k	8.45k
C99 / C101 / C102	0.1uF	0.01uF	0.01uF		
V(on)	4.75V	0.93V	1.6V	1.2V	2.98V
V(off)	4.55V	0.91V	1.4V	1.0V	2.65V

C4 0.1uF; C17 0.1uF; C29 0.22uF
20 ms delay 200 ms/µF
40 ms timeout 200 ms/µF
Must be <100*C(TMR)

R13 10.0k; R35 10.0k; R102 49.9k

Power fault LED
Q21 BSS84
R195 158
D8 HSMH-C150 1.8V@20mA

1.0V 4A FPGA supply
248 kHz

U46 LM25117
C306 0.1uF; C309 22 uF; C316 0.47uF
D36 BAT54T1G; C321 0.1uF
Q15 BSZ165N04NS G; Q16 BSZ042N04NS G
L17 2.2uH
R278 13.3k; R286 255; R281 1.02k
C324 820 pF; C317 0.056uF; C312 120 pF; R372 1.05k; R275 0.020
C331 47uF; C330 47uF; C327 47uF
TP10 Testpoint ENA_1.0V
R31 499k; C300 0.47uF; C303 0.1uF; C449 150 pF; R269 20.0k
Range 0.985–1.015V
Estimated ripple 7.86 mV
788–812 mV
1.0Vsupply
TP3 Testpoint

PTC Bank 33 LCD[4C] 1.0V_SYNC
R43 4.7k; C356 0.1uF; C390 0.1uF
U29 74LVC1T45
B_1.0V_SYNC
Sync signals must be 5V to be effective

1.5V 4A SDRAM supply
348 kHz

U47 LM25117
C307 0.1uF; C310 22 uF; C318 0.47uF
D37 BAT54T1G; C322 0.1uF
Q17 BSZ165N04NS G; Q18 BSZ105N04NS G
L18 4.70uH
R279 28.7k; R282 1.05k; R283 1.20k
C325 820 pF; C328 100uF
C301 0.47uF; C304 0.1uF; C448 150 pF; R270 14.0k
C315 0.01uF; R273 4.53K; C313 47pF; R276 0.020
R88 499k; R89 499k; R90 499k
TP12 Testpoint ENA_1.5V
Range 1.477–1.522V
Estimated ripple 3.95 mV
788-812 mV
1.5Vsupply
TP1 Testpoint

Sync circuits are not used
1.2–5.5V
R75 4.7k; C444 0.1uF; C445 0.1uF
PTC Bank 33 LCD[4C] 1.5V_SYNC
U77 74LVC1T45
B_1.5V_SYNC

3.3V 4A peripherial supply
424 kHz

U48 LM25117
C308 0.1uF; C311 22 uF; C319 0.47uF
D38 BAT54T1G; C323 0.1uF
Q19 BSZ165N04NS G; Q20 BSZ105N04NS G
L19 8.2uH
R280 49.9k; R284 3.74k; R285 1.20k
C326 820 pF; C320 2200pF; C314 10 pF; R274 20.0k; R277 0.020
C329 100uF; R288 60.4
D5 1SMB5914BT3G 3.6V; D23 S5AC-13-F; D24 S5AC-13-F; D39 HSMG-C150 2.2V@20mA
Power good LED
Accuracy ±1.5%
Estimated ripple 2.5 mV
3.3Vsupply
TP2 Testpoint
+1.8V

TP13 Testpoint ENA_3.3V
C302 0.47uF; C305 0.1uF 8 ms soft start; C350 150 pF; R271 11.3k
PTC Bank 33 LCD[4C] 3.3V_SYNC
R76 4.7k; C442 0.1uF; C443 0.1uF
U76 74LVC1T45
B_3.3V_SYNC

1.8V 4A FPGA supply
408 kHz

U19 LM25117
C505 0.1uF; C506 22 uF; C507 0.47uF
D42 BAT54T1G; C494 0.1uF
Q23 BSZ165N04NS G; Q24 BSZ105N04NS G
L20 4.70uH
R302 28.7k; R304 1.50k; R199 1.20k
C510 820 pF; C495 100uF
C508 0.47uF; C509 0.1uF 8 ms soft start; C77 150 pF; R296 11.8k
C511 0.056uF; R298 7.68K; C513 27pF; R300 0.020
TP11 Testpoint ENA_1.8V
Allowable range 1.71–1.89V
1.8Vsupply
TP17 Testpoint

PTC Bank 33 LCD[4C] 1.8V_SYNC
R78 4.7k; C496 0.1uF; C497 0.1uF
U79 74LVC1T45
B_1.8V_SYNC

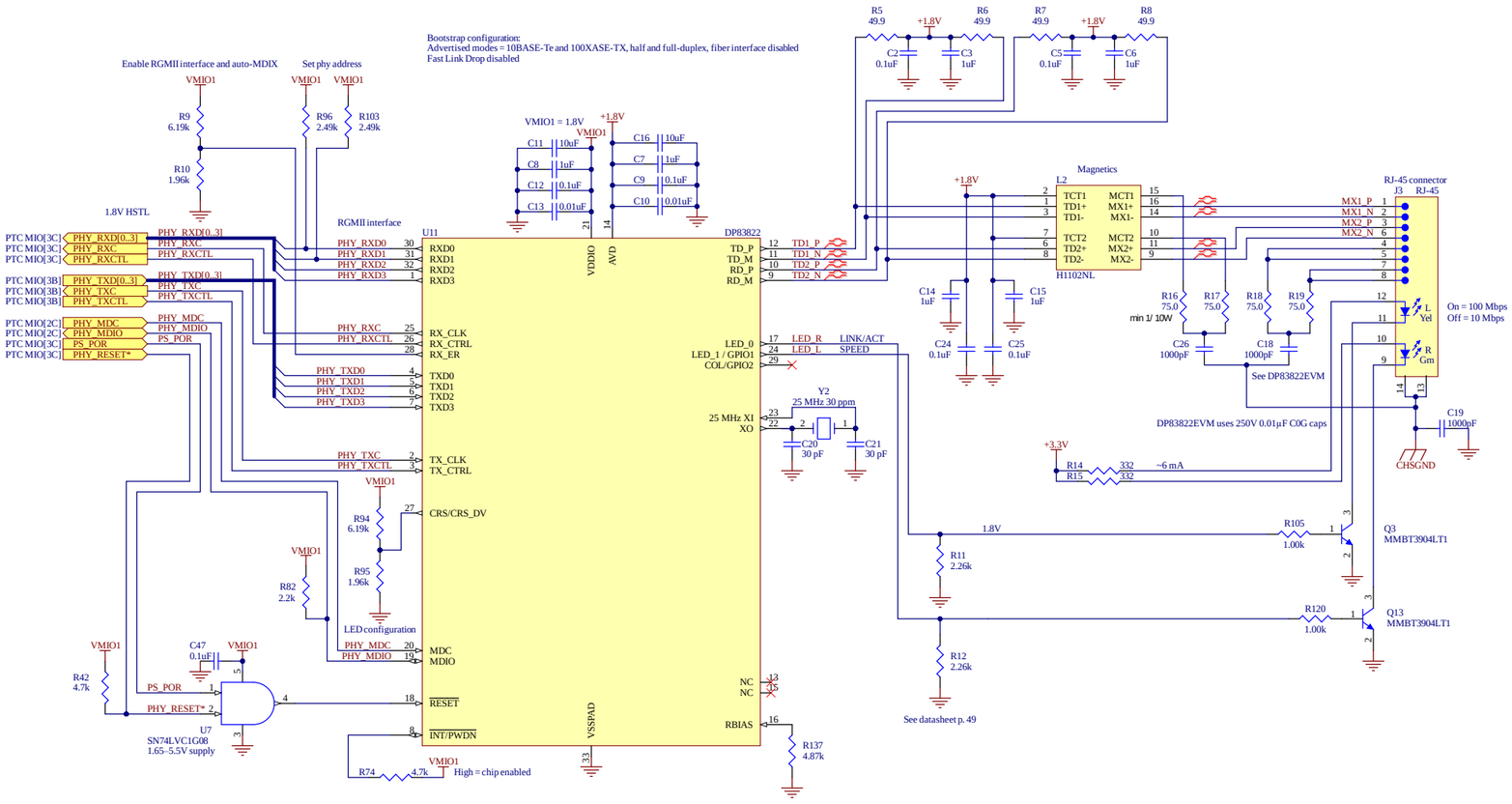
5V 2A USB and GPIB supply
Not in use; using backplane supply instead
424 kHz

U51 LM25117
C339 0.1uF; C341 10 uF; C348 0.47uF
D43 BAT54T1G; C351 0.1uF
Q25 CSD19537Q3; Q26 CSD19537Q3
L21 12.0uH
R303 48.7k; R306 6.04k; R307 1.15k
C355 820 pF; C345 3900pF; C343 39 pF; R299 22.1k; R301 0.030
C357 22uF; C541 22uF
5Vsupply
TP6 Testpoint

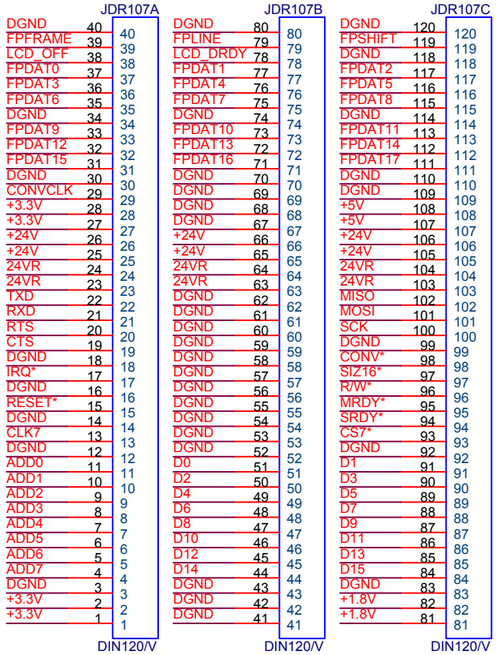
R295 54.9k; R294 3.74k
1.25V under voltage lockout
C333 0.47uF; C336 0.1uF 8 ms soft start; C450 150 pF; R297 11.3k
PTC Bank 33 LCD[4C] 5V_SYNC
R77 4.7k; C446 0.1uF; C447 0.1uF
U78 74LVC1T45
B_5V_SYNC

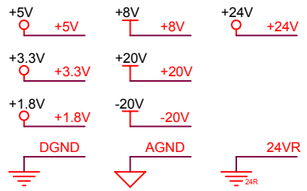
CPU card: Digital power
PTC digital power.SchDoc
8/23/2023 4:33:50 PM
Sheet 8 of 9

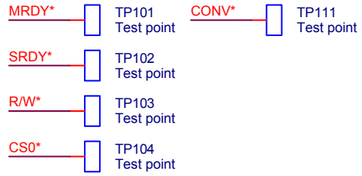
100 Mbps Ethernet



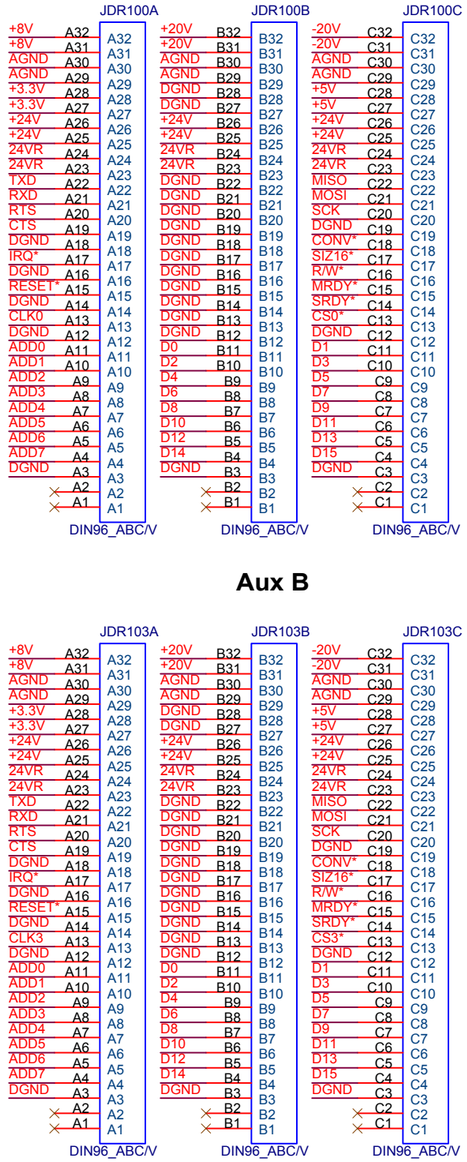
CPU



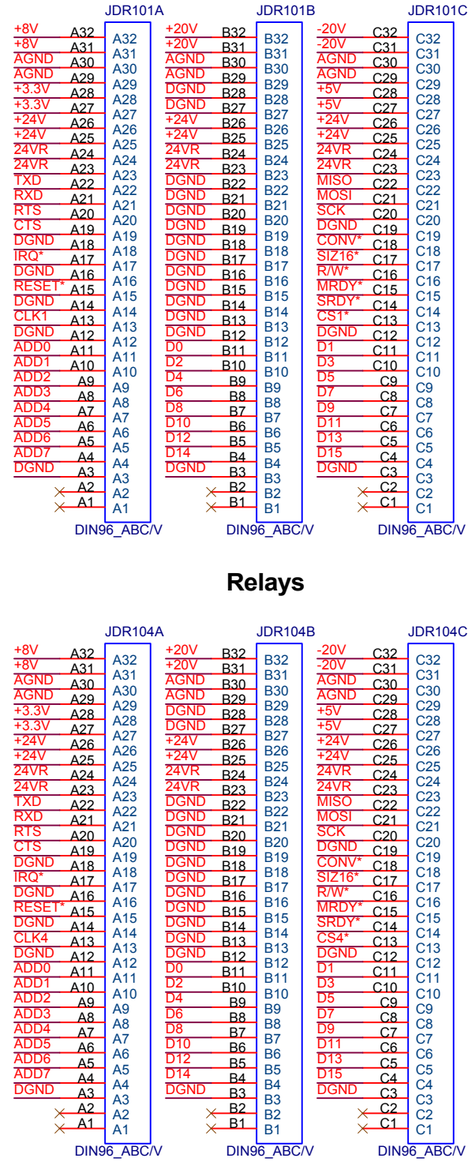




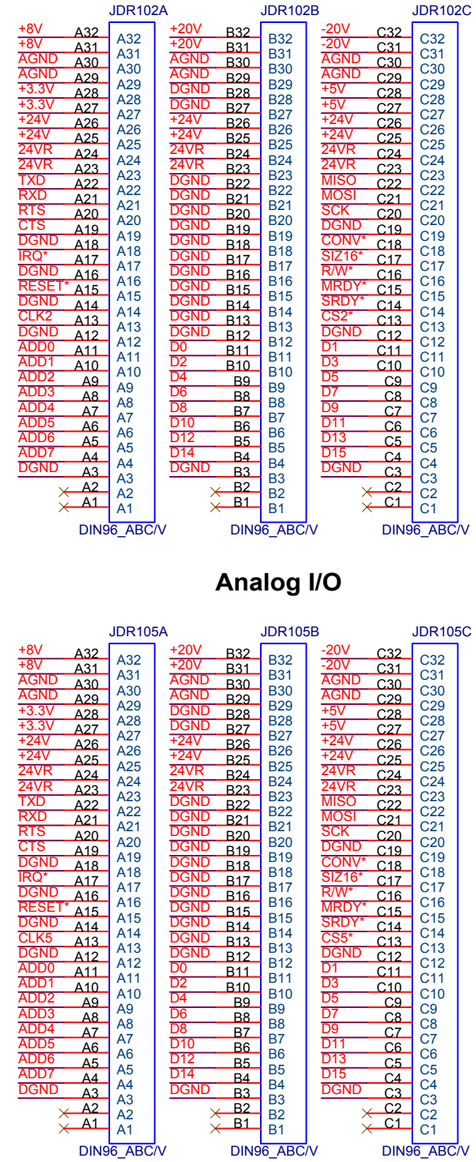
Input



Output



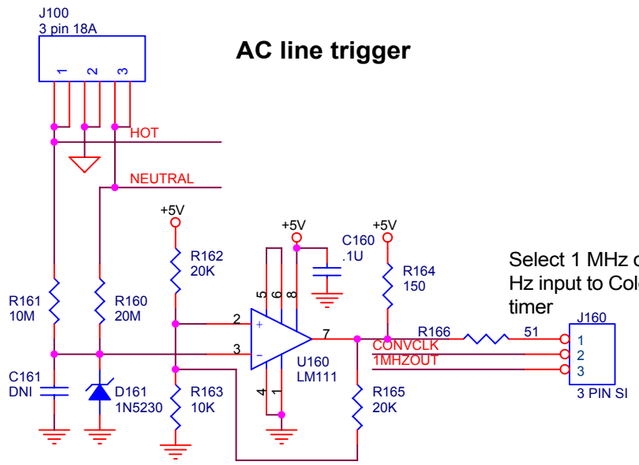
Aux A



Aux B

Relays

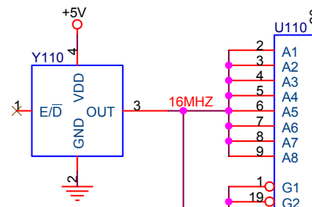
Analog I/O



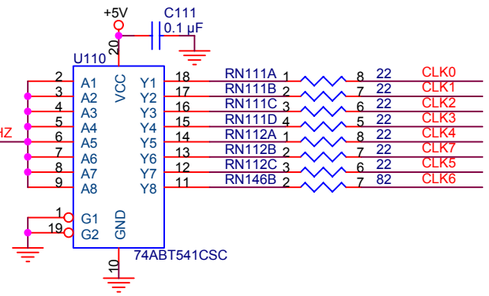
AC line trigger

Select 1 MHz or 60 Hz input to ColdFire timer

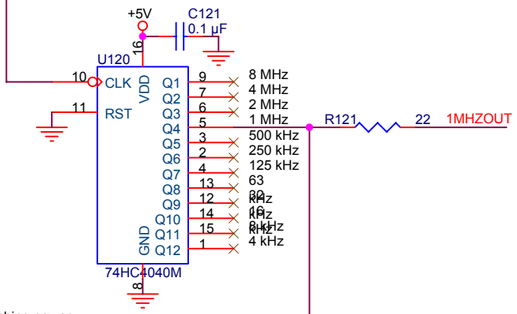
16 MHz oscillator



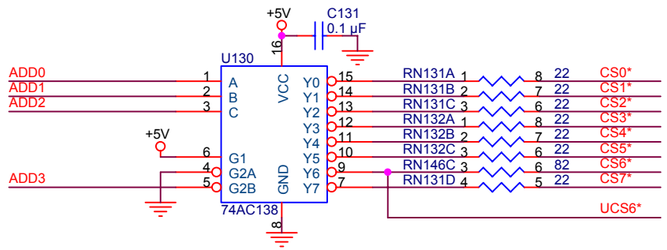
Clock driver



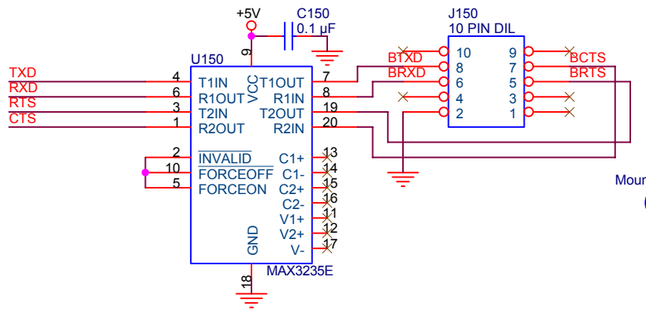
Clock divider



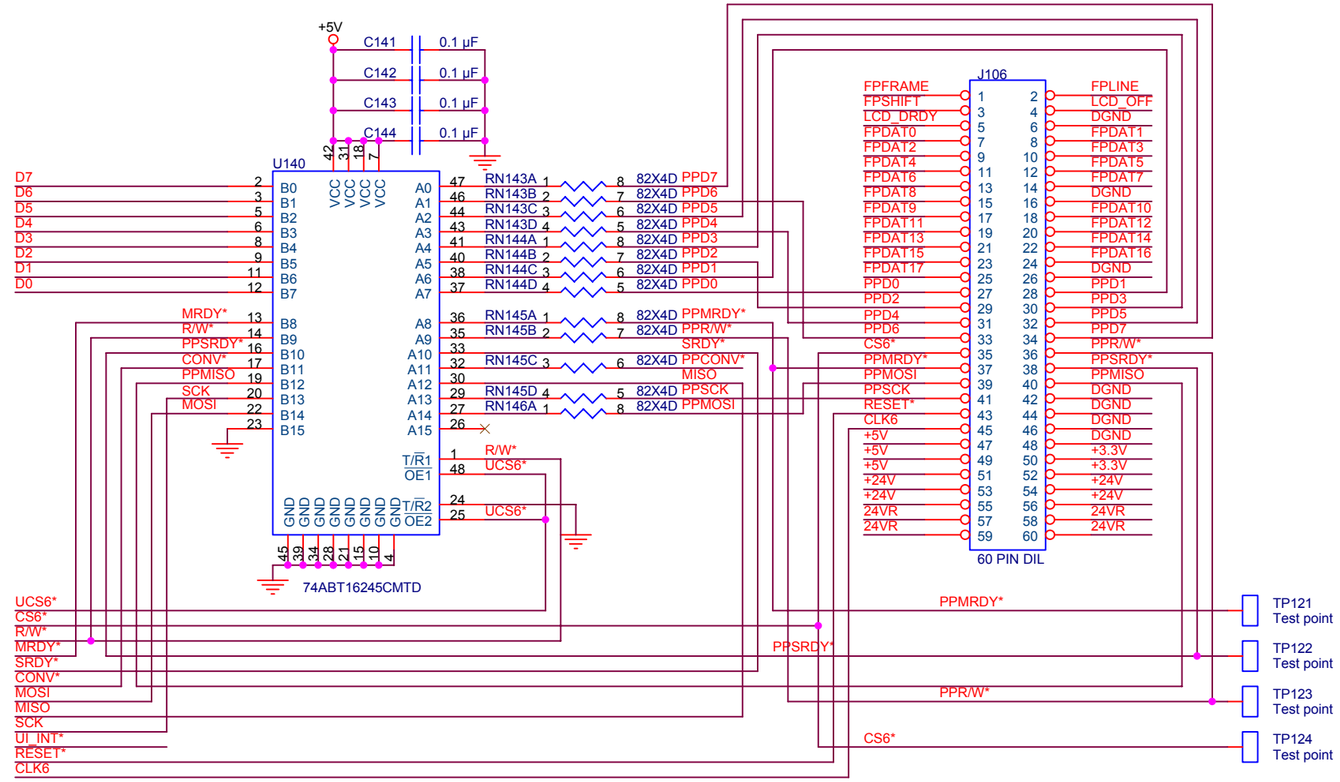
Address decoder

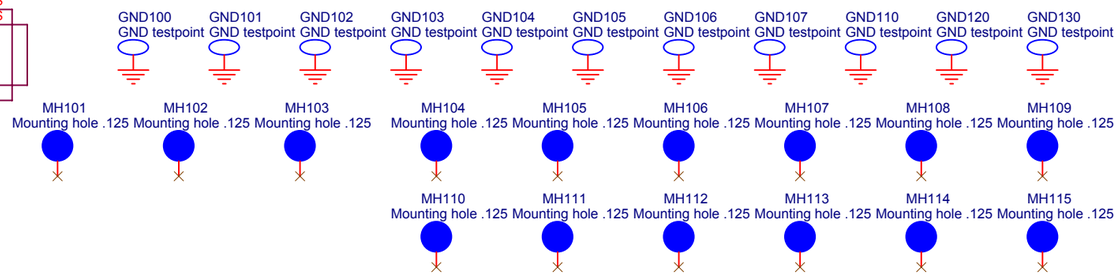


RS-232 transceiver



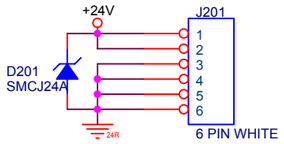
Front panel connector



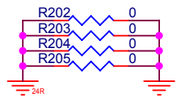


	Stanford Research Systems	
Title	PTC2222 Backplane: Cardslots	
Size	Document Number	Rev D
Date: Thursday, November 01, 2018	Sheet 1 of 3	

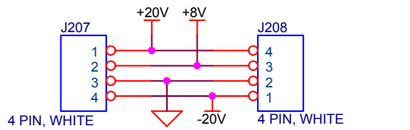
Power input



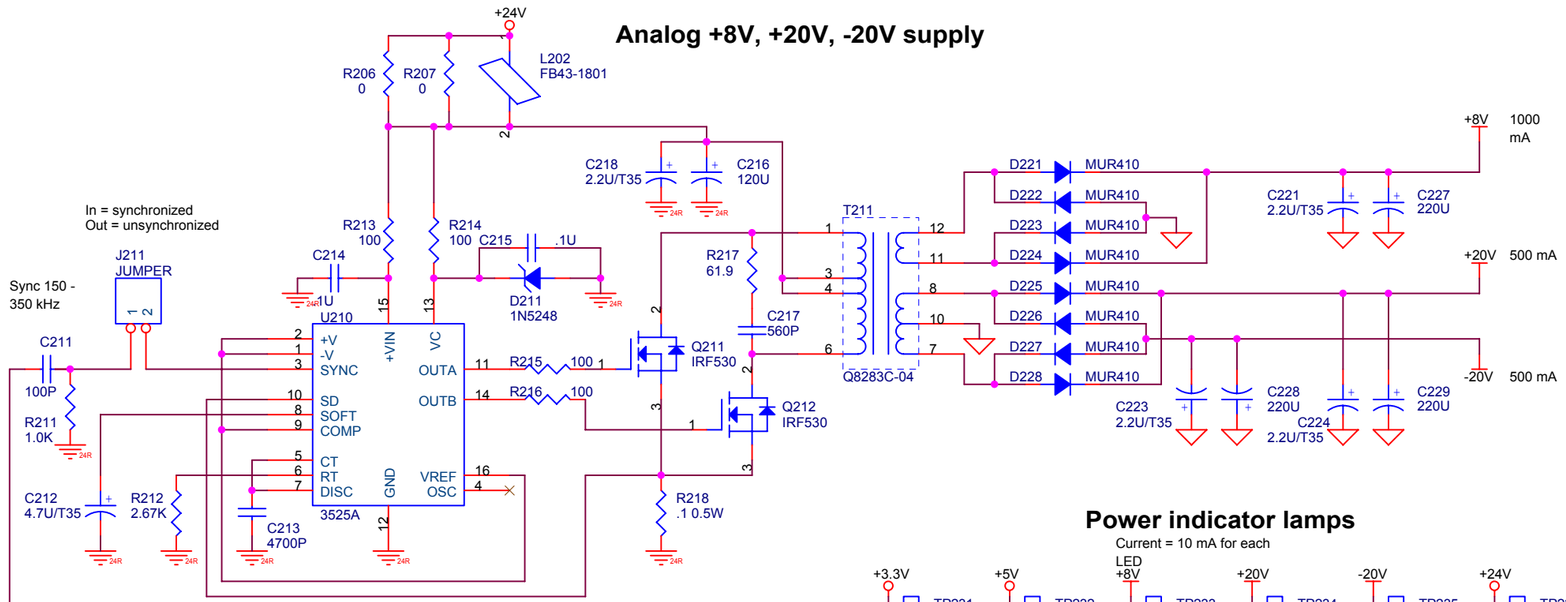
Ground interconnect



Power interconnects

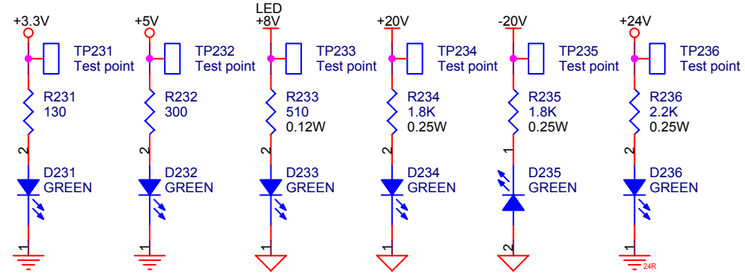


Analog +8V, +20V, -20V supply

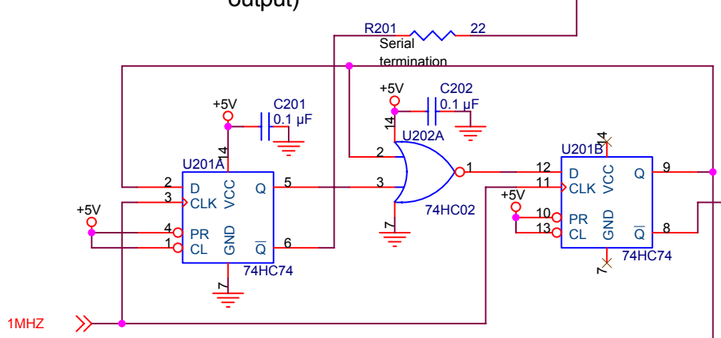


Power indicator lamps

Current = 10 mA for each LED

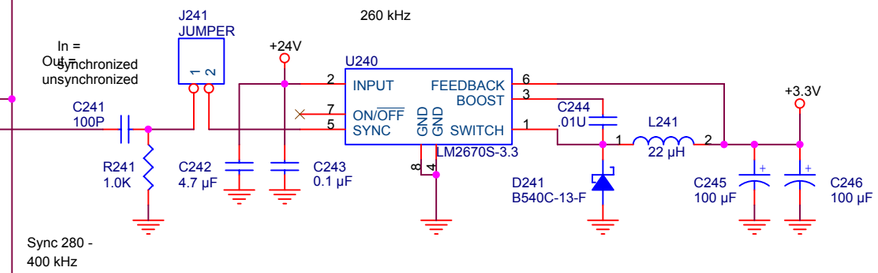


Divide by 3 (four-phase output)

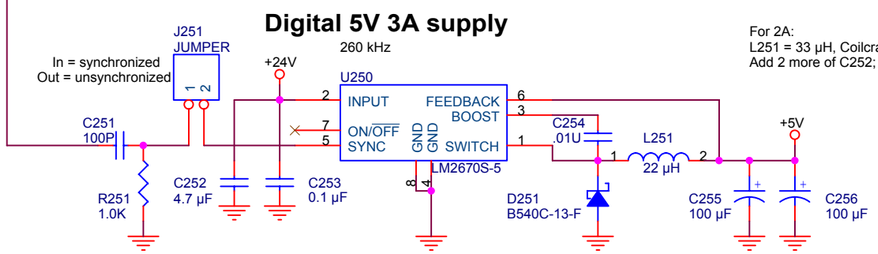


Digital 3.3V 2A supply

Replace with LM1117 800 mA linear regulator?

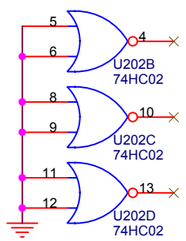


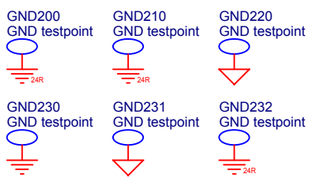
Digital 5V 3A supply



For 2A:
L251 = 33 µH, Coilcraft DO5022P-333
Add 2 more of C252; delete C253?

Connect unused inputs to ground





Stanford Research Systems		
Title: PTC2222 Backplane: Switching supplies		
Size	Document Number	Rev D
Date: Thursday, November 01, 2018	Sheet 3 of 3	

FPFRAME
FPLINE
FPSHIFT
LCD_DRDY
FPDAT[0..18]
B[0..7]

Microcontroller

JTAG port
J101
TCK
TDO
TMS
TDI
GND
VTREF
PPRESET*
GND
+3.3V
10 pin DIL

Omit RN105
RN105C 47k x 4
RN105B 47k x 4
RN105A 47k x 4
RN105D 47k x 4
RN202A 47k x 4

C101 0.1uF
C102 0.1uF
C103 0.1uF

U101
ATmega162
CLK
PPRESET*
XTAL1
XTAL2
RESET
PA0 / AD0 / PCINT0
PA1 / AD1 / PCINT1
PA2 / AD2 / PCINT2
PA3 / AD3 / PCINT2
PA4 / AD4 / PCINT4
PA5 / AD5 / PCINT5
PA6 / AD6 / PCINT6
PA7 / AD7 / PCINT7
PB0 / OC0 / T0
PB1 / OC2 / T1
PB2 / RXD1 / AIN0
PB3 / TXD1 / AIN1
PB4 / SS / OC3B
PB5 / MOSI
PB6 / MISO
PB7 / SCK
PC0 / A8 / PCINT8
PC1 / A9 / PCINT9
PC2 / A10 / PCINT10
PC3 / A11 / PCINT11
PC4 / A12 / PCINT12 / TCK
PC5 / A13 / PCINT13 / TMS
PC6 / A14 / PCINT14 / TDO
PC7 / A15 / PCINT15 / TDI
PD0 / RXD0
PD1 / TXD0
PD2 / INT0 / XCK1
PD3 / INT1 / ICP3
PD4 / TOSC1 / OC3A / XCK0
PD5 / TOSC2 / OC1A
PD6 / WR
PD7 / RD
PE0 / INT2 / ICP1
PE1 / ALE
PE2 / OC1B
D0 D1 D2 D3 D4 D5 D6 D7
B0 B1 B2 B3 B4 B5 B6 B7
FAN
SPEAKER
BUSY*
LED_CS*
TS_CS*
MOSI
MISO
SCK
CS*
PPCONV*
MRDY*
R/W*
SRDY*
TACH
LED*
BACKLIGHT_ON*

RN101A 8 1 4.7k x 4
RN101B 7 2 4.7k x 4

R104 158
D101 HSMH-C150 1.8V@20mA

Bidirectional transceiver

C105 0.1uF
C106 0.1uF
C107 0.1uF
C108 0.1uF

U102
74LCX16245
A0 A1 A2 A3 A4 A5 A6 A7
B0 B1 B2 B3 B4 B5 B6 B7
D0 D1 D2 D3 D4 D5 D6 D7
MRDY* R/W* PSRDY* CS* PMISO PMOSI PSCK
PPMRDY* PPR/W* SRDY* PPCS* MISO PPMOSI PPSCK
PPR/W* PPCS*
T/R1 OE1
T/R2 OE2
Read = high = A to B
Write = low = B to A

Series terminators
RN102D 4 5 82 x 4 PPD0
RN102C 3 6 82 x 4 PPD1
RN102B 2 7 82 x 4 PPD2
RN102A 1 8 82 x 4 PPD3
RN103D 4 5 82 x 4 PPD4
RN103C 3 6 82 x 4 PPD5
RN103B 2 7 82 x 4 PPD6
RN103A 1 8 82 x 4 PPD7

RN104A 1 8 82 x 4 PPMISO
RN104C 3 6 82 x 4 PPSRDY*
RN104B 2 7 82 x 4
RN104D 4 5 82 x 4

0.1uF C110
C111 0.1uF
U110 74LVC1T45
A B DIR
CLK
PPCLK

TP102 Testpoint
TP103 Testpoint
TP104 Testpoint
TP101 Testpoint

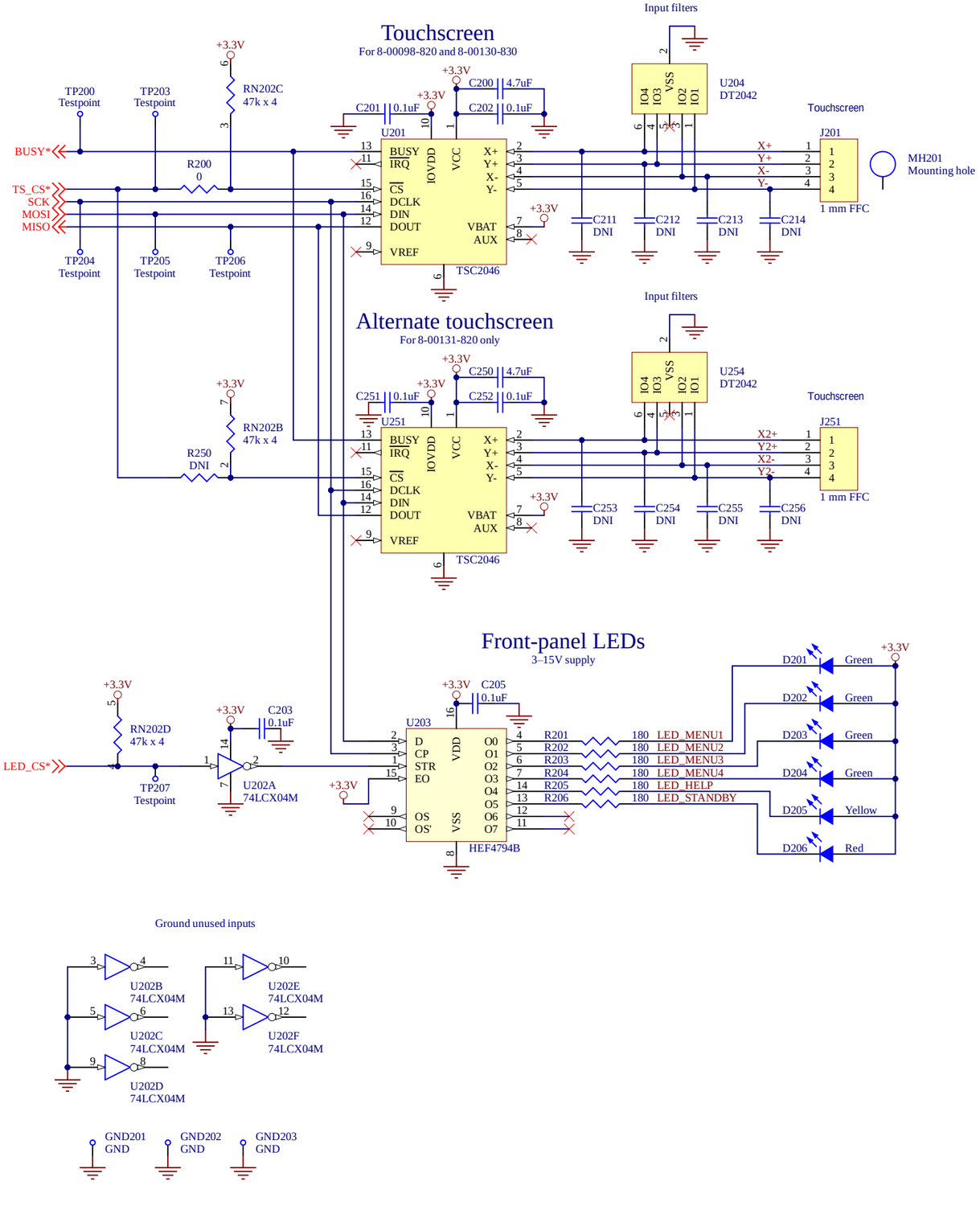
Parallel port (ribbon cable connector)

PPD[0..7]
FPDAT[0..18]
J106
60 pin DIL
FPFRAME FPSHIFT LCD_DRDY FPDAT0 FPDAT2 FPDAT4 FPDAT6 FPDAT8 FPDAT9 FPDAT11 FPDAT13 FPDAT15 FPDAT17 PPD0 PPD2 PPD4 PPD6 PPCS* PPMRDY* PPMOSI PPSCK PPRESET* PPCLK
FPLINE MCF_LCD_OFF FPDAT1 FPDAT3 FPDAT5 FPDAT7 FPDAT10 FPDAT12 FPDAT14 FPDAT16 PPD1 PPD3 PPD5 PPD7 PPR/W* PPSRDY* PPMISO
GND +5V +24V 24R +3.3V

MH101 Mounting hole
MH102 Mounting hole
MH103 Mounting hole
MH104 Mounting hole
MH105 Mounting hole
MH106 Mounting hole
MH107 Mounting hole
MH108 Mounting hole
MH109 Mounting hole
MH110 Mounting hole
GND101 GND
GND102 GND
GND103 GND

SRS ART101 SRS logo
PC1 PCB

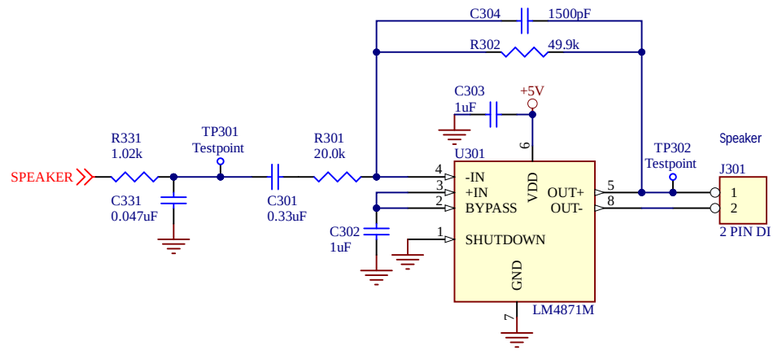
PTC2328 front panel: CPU
CPU.SchDoc
8/18/2023 3:12:04 PM
Sheet1 of 3



PTC2328 front panel: Input	
Input.SchDoc	
8/18/2023 3:12:05 PM	Sheet2 of3

Speaker driver

Supply = 2.0–5.5V, Gain=1.5x, cutoff=20 kHz



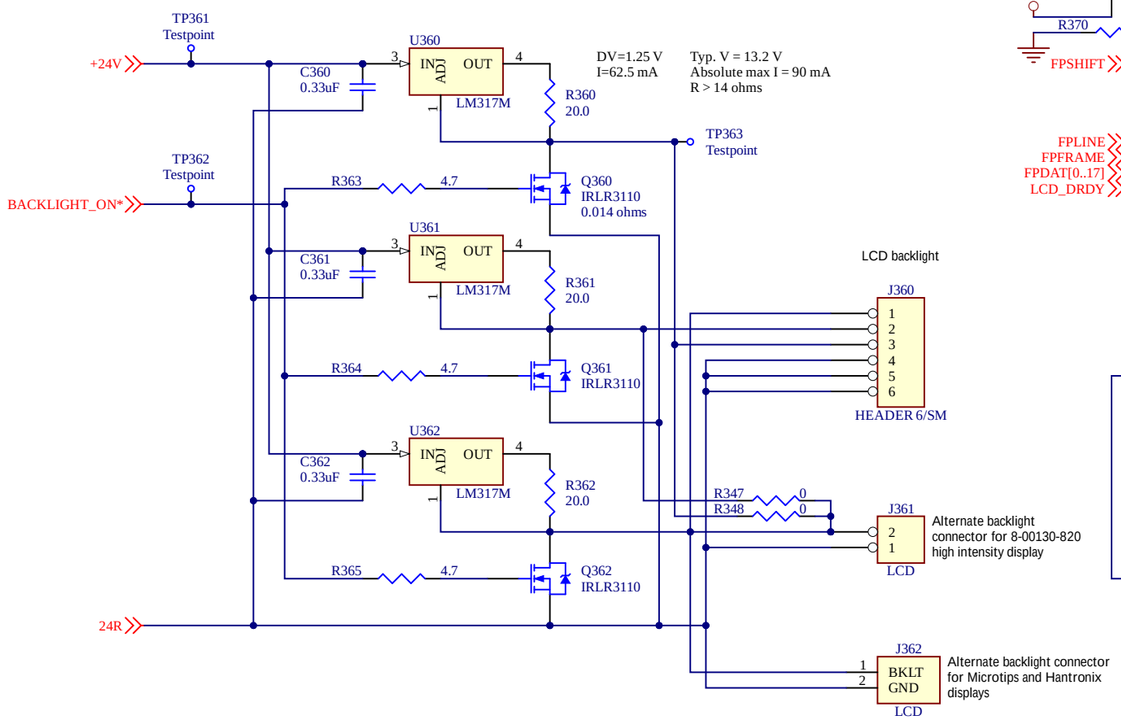
LCD displays:

8-98: Remove R347 and R348 to use this LCD.
Old version: T-55265GD057J-LW-ADN (obsolete) : install jumper J371
New version: T-55265GD057J-LW-AMN: omit jumper J371
Possible alternate: Microtips MTF-TQ57SN721AV. Change R360-R362 to 12.5 ohms. Smaller screen might require new Lexan.

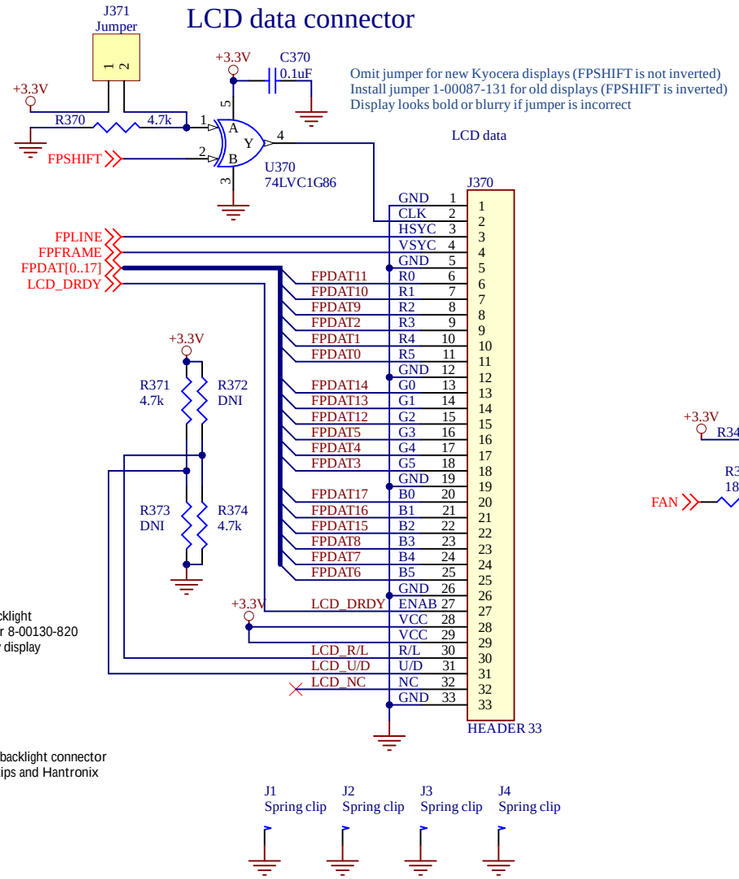
8-130: high intensity LED. Same as 8-98 but with a different LED connector. Schematic is compatible with this display as drawn.
Old version: T-55520GD057J-LW-ACN (obsolete): install jumper J371
New version: none.

8-131: includes a resistive touch panel. Equivalent to 8-98 + 8-21, except touch connector is in a different location. Remove R200, R347, and R348 and install R250 to use this LCD.
Old version: T-55265GD057J-LW-ACN (obsolete) : install jumper J371
New version: T-55265GD057J-LW-AKN: remove jumper J371

LCD backlight power supply

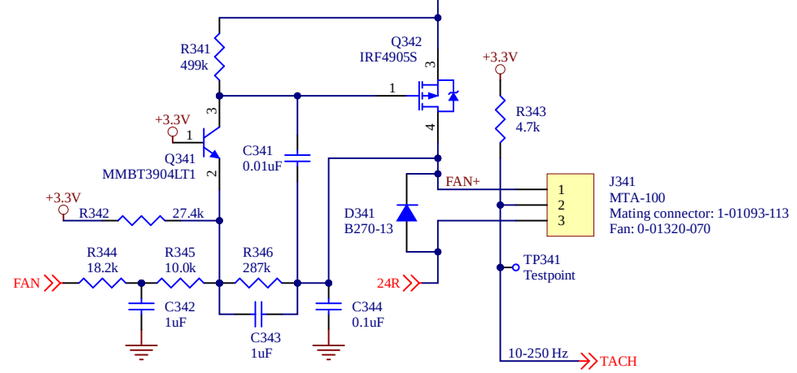


LCD data connector



Variable-speed fan driver

Fan in: 30 or 60 kHz PWM
Fan out: 24W (24V 1A), min load 24 ohms
Polarity is reversed relative to rev F
See Maxim APP 3530



GND301 GND
GND302 GND
GND303 GND

PTC2328 front panel: Output	
Output.SchDoc	
8/18/2023 3:12:06 PM	Sheet3 of 3

Expansion connector

A[0..23]

J202

+3.3V	1	2	+3.3V
A0	3	4	A1
A2	5	6	A3
A4	7	8	CS*
RD*	9	10	WR*
BRDY1	11	12	BCPUACC
BDRQ	13	14	RESET*
BE0	15	16	BE1
BINTR	17	18	RW*
D8	19	20	D9
D10	21	22	D11
D12	23	24	D13
D14	25	26	D15
D0	27	28	D1
D2	29	30	D3
D4	31	32	D5
D6	33	34	D7
	35	36	
	37	38	
+5V	39	40	+5V

40 PIN DI

D[0..15]

P/N in BOM is 1-00291-000
Should be 1-01291-131

5V tolerant transceivers

Propagation delay 7 ns
Input capacitance 7 pF

+3.3V
C111 0.1uF C113 0.1uF
C112 0.1uF C114 0.1uF

U110 74LCX16245

D8	47	A0	BD8
D9	46	A1	BD9
D10	44	A2	BD10
D11	43	A3	BD11
D12	41	A4	BD12
D13	40	A5	BD13
D14	38	A6	BD14
D15	37	A7	BD15
D0	36	A8	BD0
D1	35	A9	BD1
D2	33	A10	BD2
D3	32	A11	BD3
D4	30	A12	BD4
D5	29	A13	BD5
D6	27	A14	BD6
D7	26	A15	BD7
RD*	1	T/R1	
CS*	48	OE1	
RD*	24	T/R2	
CS*	25	OE2	

VCC: 7, 18, 31, 42
GND: 4, 10, 15, 21, 28, 34, 39, 45

BD[0..15]

+3.3V
C121 0.1uF C123 0.1uF
C122 0.1uF C124 0.1uF

U120 74LCX16245

RESET*	2	B0	A0 47 BRESET*
INTR	3	B1	A1 46 BINTR
RDY1	5	B2	A2 44 BRDY1
BE0	6	B3	A3 43 BABUS*
WR*	8	B4	A4 41 BWR*
BE1	9	B5	A5 40 BBBUS*
CS*	11	B6	A6 38 BCS*
RD*	12	B7	A7 37 BRD*
A4	13	B8	A8 36 BA4
A3	14	B9	A9 35 BA3
A2	16	B10	A10 33 BA2
A1	17	B11	A11 32 BA1
A0	19	B12	A12 30 BA0
DRQ	20	B13	A13 29 BDRQ
CPUACC	22	B14	A14 27 BCPUACC
RW*	23	B15	A15 26
		T/R1	1
		OE1	48
		T/R2	24
		OE2	25

VCC: 42, 31, 18, 7
GND: 45, 39, 34, 28, 21, 15, 10, 4

GPIB chip

+5V
C131 0.1uF C135 0.1uF
C132 0.1uF C136 0.1uF
C133 0.1uF C137 0.1uF
C134 0.1uF C138 0.1uF
C139 0.1uF

Y101 40 MHz

U130 TNT4882

BRESET*	67	RESETN	
	95	XTAL0	
	96	XTAL1	
BD15	2	D15	
BD14	3	D14	
BD13	5	D13	
BD12	6	D12	
BD11	7	D11	
BD10	9	D10	
BD9	10	D9	
BD8	11	D8	
BD7	50	D7	
BD6	49	D6	
BD5	47	D5	
BD4	46	D4	
BD3	44	D3	
BD2	43	D2	
BD1	42	D1	
BD0	39	D0	
BA4	19	ADDR4	
BA3	18	ADDR3	
BA2	17	ADDR2	
BA1	16	ADDR1	
BA0	15	ADDR0	
BCS*	55	CSN	
BABUS*	14	ABUSN	
BBBUS*	62	BBUSN	
BRD*	63	RDN	
BWR*	64	WRN	
RDY1	38	RDY1	
INTR	34	INTR	
	26	PAGED	
	29	SWAPN	
	31	BURST_RDN	
	33	DACKN	
	53	MODE	
	20	ABUS_OEN	
	1	BBUS_OEN	
	21	TADCS	
	66	LADCS	
CPUACC	22	CPUACC	
	23	TRIG	
	28	REM	
	30	FIFO_RDY	
DRQ	32	DRQ	
	51	DCAS	
		NC	52

VDD: 12, 24, 36, 40, 58, 60, 69, 87, 94
GND: 4, 8, 13, 25, 27, 35, 37, 41, 45, 48, 54, 56, 57, 59, 61, 65, 68, 72, 75, 78, 83, 86, 90, 93, 97

R131 0

MODE low = 9914 mode

RENN	70	17	REN
IFCN	79	9	IFC
NDACN	81	8	NDAC
NRFDN	82	7	NRFD
DAVN	84	6	DAV
EOIN	85	5	EOI
ATNN	73	11	ATN
SRQN	76	10	SRQ
KEYCLKN	98		
KEYDQ	99		
KEYRSTN	100		

J140 IEEE488

DIO1N	92	1	DIO1
DIO2N	91	2	DIO2
DIO3N	89	3	DIO3
DIO4N	88	4	DIO4
DIO5N	80	13	DIO5
DIO6N	77	14	DIO6
DIO7N	74	15	DIO7
DIO8N	71	16	DIO8
		12	SHIELD
		24	GND L
		21	GND 9
		20	GND 8
		19	GND 7
		18	GND 6
		23	GND 11
		22	GND 10

SHELL: 25, 26

R141 DNI
R142 DNI

Z2
Screw PP 4-40 x 5/16

Z3
Screw PP 4-40 x 5/16

Screws to hold GPIB connector to PCB

ART101
SRS
Logo

GND101 GND
GND102 GND
GND103 GND
GND104 GND

MH101 Mounting hole
MH102 Mounting hole
MH103 Mounting hole
MH104 Mounting hole

PC1
PCB

GPIB card	
C:\Temperature controller\PTC2411 GPIB card\1 GPIB.SchDoc	
8/8/2018 11:01:13 AM	Sheet 1 of 1

Microcontroller

Bidirectional transceivers

Parallel port (96-pin connector)

+8V
+20V
-20V
+24V
24R
AGND
PPCLK

D[0..7]

JTAG port
J111
TCK
TDO
TMS
+5V
TDI
GND
+5V
PPRESET*
GND
10 PIN DIL

Temperature sensor
U140
+Vs
GND
Vout
LM34DM

C111 0.1 µF
C112 0.1 µF
C113 0.1 µF

C121 0.1 µF
C122 0.1 µF
C123 0.1 µF
C124 0.1 µF

U110
XTAL1
XTAL2
PEN
RESET
PPCLK
PPRESET*
PA0 / AD0
PA1 / AD1
PA2 / AD2
PA3 / AD3
PA4 / AD4
PA5 / AD5
PA6 / AD6 / PCINT6
PA7 / AD7
PB7 / OC2 / OC1C
PB6 / OC1B
PB5 / OC1A
PB4 / OC0
PB0 / SS
PB2 / MOSI
PB3 / MISO
PB1 / SCK
PC0 / A8
PC1 / A9
PC2 / A10
PC3 / A11
PC4 / A12
PC5 / A13
PC6 / A14
PC7 / A15
AREF
PF0 / ADC0
PF1 / ADC1
PF2 / ADC2
PF3 / ADC3
PF4 / ADC4 / TCK
PF5 / ADC5 / TMS
PF6 / ADC6 / TDO
PF7 / ADC7 / TDI
PD0 / SCL / INT0
PD1 / SDA/INT1
PD2 / RXD1 / INT2
PD3 / TXD1 / INT3
PD4 / TOSC1 / ICP1
PD5 / XCK1
PD6 / T131
PD7 / T2
PE0 / RXD0 / PDI
PE1 / TXD0 / PDO
PE2 / XCK0 / AIN0
PE3 / OC3A / AIN1
PE4 / OC3B / INT4
PE5 / OC3C / INT5
PE6 / T3 / INT6
PE7 / ICP3 / INT7
PG0 / WR
PG1 / RD
PG2 / ALE
PG3 / TOSC2
PG4 / TOSC1
ATmega64

U120
D0..D7
MRDY*
R/W*
PPSRDY*
CS*
PPMISO
PMOSI
PSCK
PPR/W*
PPCS*
T/R1
OE1
T/R2
OE2
A0..A15
B0..B15
PPD0..PPD7
PPMRDY*
PPR/W*
SRDY*
PPCS*
PMISO
PPMOSI
PPSCK
74ABT16245

JDR121A
JDR121B
JDR121C
DIN96_ABC/R

PPD[0..7]

RN111A 47k
RN112A 47k
RN111C 47k
RN112B 47k
RN111B 47k
RN112C 47k
RN111D 47k
RN112D 47k
RN121A 47k
RN121B 47k
RN121C 47k
RN121D 47k

PWM_OUT
MOSI
MISO
SCK
SPI_ADD0
SPI_ADD1
PAR_CS
ADC_BUSY1*
PMOSI
PMISO
ADC_BUSY1*
PPCONV*
CS*
MRDY*
R/W*
SRDY*
LED*
PPCS*
PPMOSI
PPSCK

TP110
Test point
C115
1U

Status LED
R112
300
D111
RED MINI

R113
1.0K

ART101
SRS
Logo
PC1
PCB

PTC3238: CPU	
1 CPU.SchDoc	
11/1/2018 3:15:00 PM	Sheet 1 of 6

Select current source and forward/reverse current

RTD/thermistor connectors

Calibration port

J200

V1+ V1- REF1+ REF1- REF_IN1 — 1 2 3 4 5 6 7 8 9 10 — ADCIN1+ ADCIN1- ADCREF1+ ADCREF1- REF_OUT1

10 PIN DIL

ANA_GAIN1

REF_OUT1

REF_IN1

J260 9 PIN D

RN200A RN200B 8 47X4D 47X4D

RN200D 47X4D RN200C 47X4D

C200 1000 pF

I1+

V1+

R261 1.0M

RN203D 47X4D RN203C 47X4D

C205 1000 pF

RN203B 47X4D RN203A 47X4D

C206 1000 pF

I1-

V1-

R260 1.0M

I2- V2- I2+ V2+

C230 0.1 µF +15V -15V C231 0.1 µF

U230

NO1A NO2A NO3A NO4A NO1B NO2B NO3B NO4B EN A0 A1 GND V+ V- COMA COMB

IVAR1

IFIXED1

ANA_CAL*

ANA_CAL* ANA_F/R1* ANA_CURR_SRC1

MAX339CSE 50 pA, 500 ohms, 5 pC

ANA_EC_SHORT1*

+5VANA C210 0.1 µF

U210A

MAX4635 4.75 ohms, 0.35 nA

Current bypass

Set ANA_CAL* to 0 to isolate reference resistors so their resistance can be measured through the calibration port

ADC input buffers

RN201A 47X4D RN201B 47X4D

C201 1000 pF

U260A LT6012ACS

C295D 0.01 µF

RN292D 1.2kX4D

RN291D 4.7KX4D

RN291A 4.7KX4D

C295A 0.01 µF

RN292A 1.2kX4D

RN201C 47X4D RN201D 47X4D

C202 1000 pF

U260B LT6012ACS

RN202A 47X4D RN202B 47X4D

C203 1000 pF

U260C LT6012ACS

C295C 0.01 µF

RN292C 1.2kX4D

RN291C 4.7KX4D

R281 2.0M

RN291B 4.7KX4D

C295B 0.01 µF

RN292B 1.2kX4D

R282 2.0M

RN202C 47X4D RN202D 47X4D

C204 1000 pF

U260D LT6012ACS

Bias current 20 pA typ., 450 pA max

A0: Low=forward=NC High=reverse=NO

Forward current: 2 µA @ 0.4V, 2 mA @ 0.7V

x30 gain

+5VANA C250 0.1 µF

U250A

MAX4635 4.75 ohms, 0.35 nA

R250 MAX5491

D205 MMBD1503/A

D206 MMBD1503/A

-RAIL +RAIL

C291 1uF

C292 1uF

Compensate for current direction

+15V C260 0.1 µF

U260E OPA4277UA

C261 0.1 µF -15V

ADC

+5VANA C290 1 µF

U290

IN+ IN- REF+ REF- VCC F0 EXT CS SCK SDI SDO BUSY GND GND GND GND

LTC2440

D207 MMBD1503/A

U250B

MAX4635 x30 gain

R251 MAX5491

D208 MMBD1503/A

C280 0.1 µF +5VANA

U280

NC1 NO1 NC2 NO2 NC3 NO3 NC4 NO4 A0 EN V+ GND COM1 COM2 COM3 COM4

ADCREF1+ ADCREF1-

MAX4674 4 ohms, 1 nA

C281 0.1 µF +5VANA

U281

NO1 NC1 NO2 NC2 IN1 IN2 V+ GND COM1 COM2

ADCREF1+ ADCREF1-

C293 1uF C294 1uF

MAX4635 4 ohms, 0.3 nA

5V reference

+15V C270 0.1 µF

U270

NR IN OUT NC1 TRIM NC2 NC3 GND

C272 1 µF

5VREF

C271 22U-T16

MAX6250BCSA

U210B MAX4635

R233 2.49K

2.495V_A

U233 TL431CD5

2.5V offset voltage is needed for x30 gain in reverse current mode

0.1 - 10 Hz noise = 6 µV p-p

Low=forward High=reverse

ANA_F/R1* ANA_REF_SRC1 Low=reference resistor=NC High=voltage reference=NO

REF1+ REF1- ANA_F/R1* ANA_REF_SRC1 ANA_CURR_SRC1

ANA_ADC_CS1* ANA_SCK ANA_MOSI ANA_MISO ANA_BUSY1*

5VREF

PTC3238: Channel A	
2 Channel A.SchDoc	
11/1/2018 3:15:00 PM	Sheet 2 of 6

Optoisolators

SPI-to-parallel converter

Analog regulators

ADC input protection rails

SPI_ADD0
SPI_ADD1
SCK
PAR_CS
MOSI
MISO
ADC_BUSY1*

RN310B 470X4D
RN310A 470X4D
RN310D 470X4D
RN310C 470X4D
RN332B 470X4D
RN332C 470X4D

ISO310A HCPL2630
ISO310B HCPL2630
ISO311A HCPL2630
ISO311B HCPL2630
ISO330A HCPL2630
ISO330B HCPL2630

RN312B 4.7KX4D
RN312A 4.7KX4D
RN312C 4.7KX4D
RN312D 4.7KX4D
RN330B 470X4D
RN330A 470X4D

C310 0.1 µF
C311 0.1 µF
C300 0.1 µF
C330 0.1 µF
C302 0.1 µF
C340 0.1 µF
C345 0.1 µF
C341 0.1 µF
C342 0.1 µF

U300A 74HC08
U300B 74HC08
U300C 74HC08
U300D 74HC08

TP306 Test point
TP307 Test point

Address decoder
U302 SN74HC138D
A B C G1 G2A G2B VCC GND Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7

ANA_ADC_CS2*
ANA_PAR_CS*
ANA_ADC_CS1*
ANA_SCK
ANA_MOSI
ANA_MISO
ANA_BUSY1*

C305 100P
C306 100P
C307 100P

Select excitation current
U340 74HC595
SER SRCLK SRCLR RCLK G VCC GND QA QB QC QD QE QF QG QH SQH

ANA_EC1_0
ANA_EC1_1
ANA_EC1_3
ANA_EC1_4
ANA_F/R1*
ANA_CURR_SRC1
ANA_REF_SRC1
ANA_GAIN1

Select reference resistor
U345 74HC595

ANA_RR0
ANA_RR1
ANA_RR2
ANA_CAL*
ANA_EC1_2
ANA_EC_SHORT1*
ANA_SET*
ANA_RST*

RN345A 4.7KX4D
RN345B 4.7KX4D
RN345C 4.7KX4D
RN345D 4.7KX4D

Select excitation current
U341 74HC595

ANA_EC2_0
ANA_EC2_1
ANA_EC2_3
ANA_EC2_4
ANA_F/R2*
ANA_CURR_SRC2
ANA_REF_SRC2
ANA_GAIN2

Select reference resistor
U342 74HC595

ANA_EC2_2
ANA_EC_SHORT2*

ANA_ADC_CS1*
ANA_ADC_CS2*
ANA_MOSI
ANA_SCK
ANA_MISO
ANA_BUSY1*

Ferrite toroid
L300 Ferrite toroid
J300 8 PIN, WHITE

AGND
+8V
+20V
-20V

Analog +5V regulator (0.2 A)
U350 MC78M05CDT/RK
C352 1U
C353 1U
TP350 Test point
D351 1N5232

+15V regulator (0.2 A)
U360 MC78M15CDT/RK
C362 1U
C363 1U
TP360 Test point

-15V regulator (0.2 A)
U370 MC79M15CDT/RK
C372 1U
C373 1U
TP370 Test point

5VREF
Q380 BSS123LT1
R381 11.5K
R382 1.00K
R383 301
C380 0.1 µF
U380 LMC7111
0.4V
-RAIL
R385 301
+RAIL 4.6V
R386 4.22K
R387 4.99K
U385 TL431CD5

MH331 Mounting hole .125
MH332 Mounting hole .125
C331 0.1 µF
C332 0.1 µF

Ground interconnect
R391 DNI
R392 DNI
R393 DNI
R394 DNI

GND300 GND testpoint
GND301 GND testpoint
GND302 GND testpoint
GND303 GND testpoint
GND304 GND testpoint
GND305 GND testpoint
GND306 GND testpoint
GND307 GND testpoint
GND308 GND testpoint
GND309 GND testpoint
GND310 GND testpoint
GND311 GND testpoint

PTC3238: Isolators and regulators
3 Isolators and regulators.SchDoc
11/1/2018 3:15:00 PM
Sheet 3 of 6

Reference resistors

Address decoders

REF1+
REF1-
REF_IN1
REF_OUT1
REF_IN2
REF_OUT2
REF2+
REF2-
ANA_RR0
ANA_RR1
ANA_RR2
ANA_RST*
ANA_SET*
+5VANA
C411 0.1 µF
C421 0.1 µF
U410
U420
CD74HC238M
K430
K431
K432
K433
K434
K435
K436
K437
K438
K439
K440
K441
K442
K443
K444
K445
G6SK
U430
U432
U434
U436
U438
U440
U442
U444
NUD3105DMT1
R430 600 1/8W, 0.1%, 5 ppm
R432 6k
R434 60k
R436 600k
R438 600
R440 6k
R442 60k
R444 600k
600_SET1
600_RST1
6K_SET1
6K_RST1
60K_SET1
60K_RST1
600K_SET1
600K_RST1
600_SET2
600_RST2
6K_SET2
6K_RST2
60K_SET2
60K_RST2
600K_SET2
600K_RST2

PTC3238: Reference resistors
4 Reference resistors.SchDoc
11/1/2018 3:15:01 PM
Sheet 4 of 6

Select current source and forward/reverse current

RTD/thermistor connectors

Calibration port

J500

10 PIN DIL

ADC input buffers

ADC

Compensate for current direction

Current bypass

Set ANA_CAL* to 0 to isolate reference resistors so their resistance can be measured through the calibration port

Low=forward
High=reverse

2.5V offset voltage is needed for x100 gain in reverse current mode

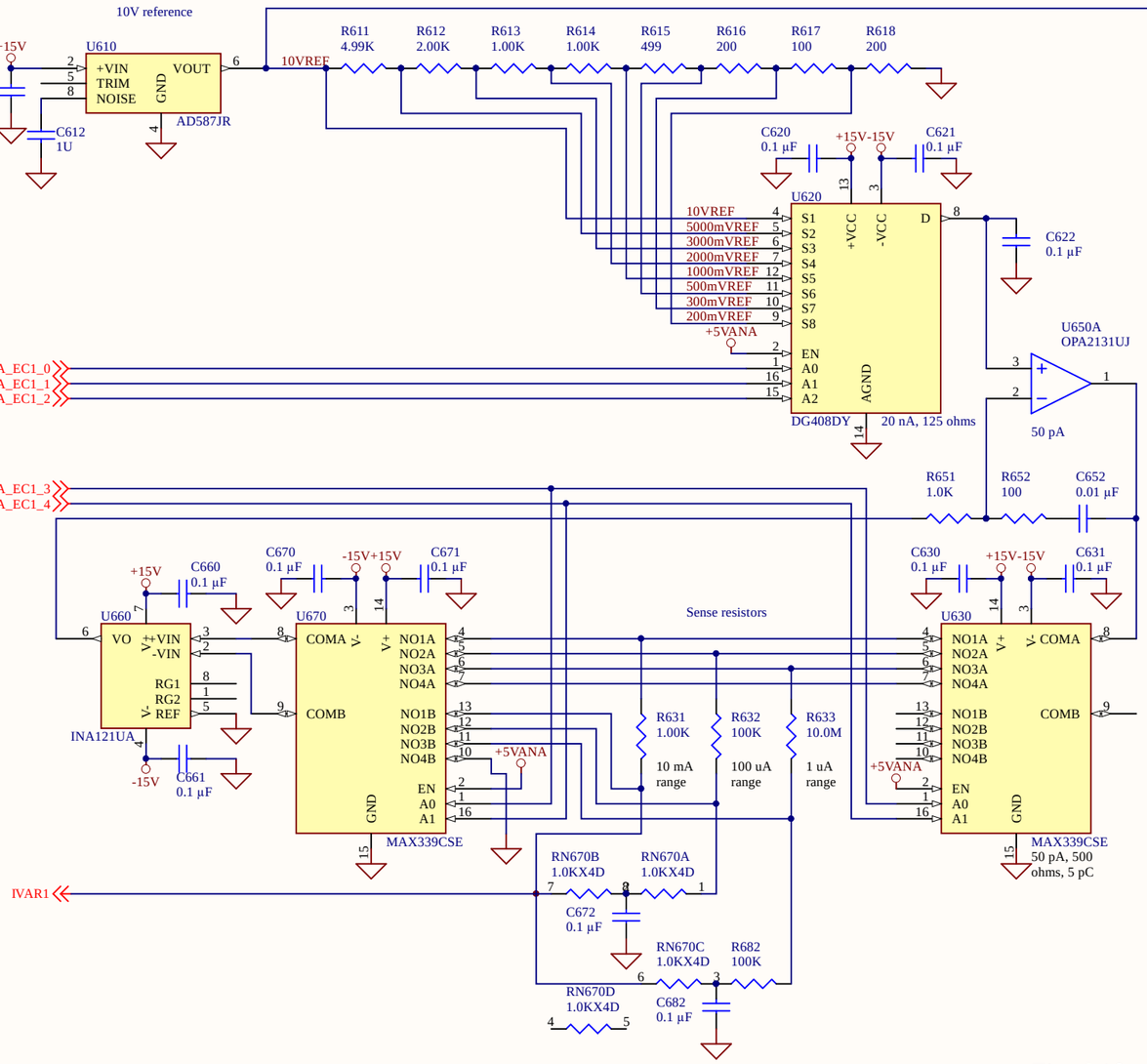
A0:
Low=forward=NC
High=reverse=NO

ANA_F/R2*
ANA_REF_SRC2
Low=reference resistor=NC
High=voltage reference=NO

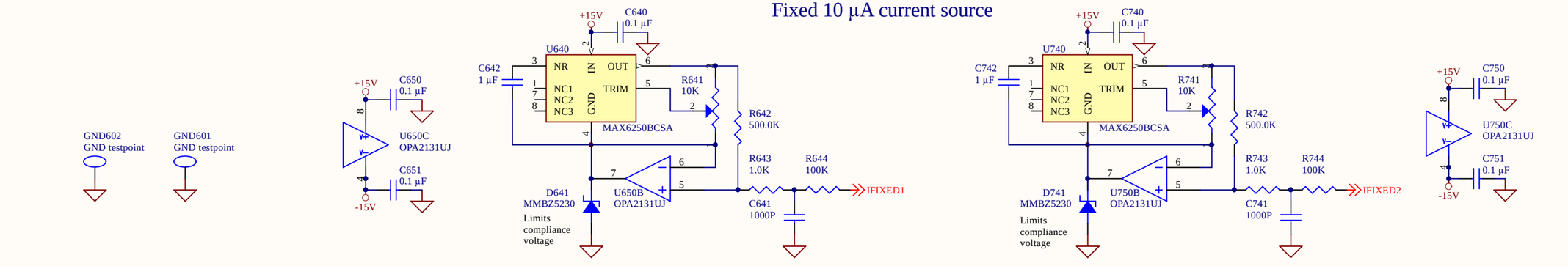
PTC3238: Channel B

5 Channel B.SchDoc
11/1/2018 3:15:01 PM
Sheet 5 of 6

Variable current source



Fixed 10 µA current source



PTC3238: Variable current source

6 Variable current source.SchDoc

11/1/2018 3:15:01 PM

Sheet 6 of 6

Microcontroller

Bidirectional transceivers

Parallel port (96-pin connector)

JTAG port

-20V
+8V
+24V
24R
PPCLK
D[0:7]

J111
10 PIN DIL
TCK
TDO
TMS
+5V
TDI
GND
VTREF
PPRESET*
GND

RN111A 47k
RN111B 47k
RN111C 47k
RN111D 47k
RN112A 47k
RN112B 47k
RN112C 47k

C111 0.1 µF
C112 0.1 µF
C113 0.1 µF
C121 0.1 µF
C122 0.1 µF
C123 0.1 µF
C124 0.1 µF

U110
ATmega64
XTAL1
XTAL2
PEN
RESET
PA0 / AD0
PA1 / AD1
PA2 / AD2
PA3 / AD3
PA4 / AD4
PA5 / AD5
PA6 / AD6 / PCINT6
PA7 / AD7
PB7 / OC2 / OC1C
PB6 / OC1B
PB5 / OC1A
PB4 / OC0
PB0 / SS
PB2 / MOSI
PB3 / MISO
PB1 / SCK
PC0 / A8
PC1 / A9
PC2 / A10
PC3 / A11
PC4 / A12
PC5 / A13
PC6 / A14
PC7 / A15
AREF
PF0 / ADC0
PF1 / ADC1
PF2 / ADC2
PF3 / ADC3
PF4 / ADC4 / TCK
PF5 / ADC5 / TMS
PF6 / ADC6 / TDO
PF7 / ADC7 / TDI
PD0 / SCL / INT0
PD1 / SDA/INT1
PD2 / RXD1 / INT2
PD3 / TXD1 / INT3
PD4 / TOSC1 / ICP1
PD5 / XCK1
PD6 / T131
PD7 / T2
PE0 / RXD0 / PDI
PE1 / TXD0 / PDO
PE2 / XCK0 / AIN0
PE3 / OC3A / AIN1
PE4 / OC3B / INT4
PE5 / OC3C / INT5
PE6 / T3 / INT6
PE7 / ICP3 / INT7
PG0 / WR
PG1 / RD
PG2 / ALE
PG3 / TOSC2
PG4 / TOSC1

U120
74ABT16245
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14 B15
T/R1
OE1
T/R2
OE2

JDR121A
JDR121B
JDR121C
DIN96_ABC/R

PWM
LOAD_EN
ADC_F0
MOSI
MISO
SCK
PAR_CS
ADC_BUSY*
SPI0
SPI1
50VSHDN

RN113A 1.0KX4D
RN113B 1.0KX4D
RN121A 47k
RN121B 47k
RN121C 47k
R112 300
RED MINI D111
Status LED

PPD[0:7]
PPCS*
PPMOSI
PPSCK

GND101 GND testpoint
GND102 GND testpoint
GND103 GND testpoint
GND104 GND testpoint
GND105 GND testpoint
GND106 GND testpoint

	Stanford Research Systems	
Title	PTC4314 DC output card: CPU	
Size	Document Number	Rev D
Date: Friday, June 24, 2011	Sheet 1 of 4	

Constant-current heater driver

MON_A0
MON_A1
MON_A2

+5VANA
U200
VS+
GND
Vout
LM34
Heatsink temperature sensor
+50V
R265 56.2K
R266 3.92K

Heater connector

J200
2 PIN, WHITE
OUT-
OUT+
50 V
D214 MUR1100E
Inductive kickback protection
C205 10 µF
R206 5.23K
R207 100K
C204 1500P
R202 5.23K
R203 100K
U270A LMC6484IM
0 - 3.824V
R208 .1 Low side sense resistor
24R
F221 1.1A fuse
F222 1.1A fuse
0.38 ohm
R263 976K
R264 73.2K

Voltage and current monitor

C261 .1U
+5VANA
U260
P_MON
N_MON
2000MA_MON
200MA_MON
20MA_MON
PCB_TEMP
50V_MON
S1 S2 S3 S4 S5 S6 S7 S8
EN A0 A1 A2
+VCC -VCC AGND D
DG408DY
RN273D 4.7KX4D
C274D 0.01 µF x 4
RN272D 10X4D
U270B LMC6484IM
4 pA, 1 µV/°C
0.18 - 4.8V
C275 1U
RN273A 4.7KX4D
C274A 0.01 µF x 4
RN272A 10X4D
U270C LMC6484IM
C276 1U
+5VANA
C281 .1U
U280
IN+ IN- VCC CS SCK SDO REF+ REF- GND F0
ADC_CS*
SCK
MISO
ANA_ADC_F0
LTC2433-1CMS
16 bit ADC
0 - 4V range
60 µV resolution
4VREF
2VREF

SWITCH A[0:2]
0: Heater + voltage
1: Heater - voltage
2: Heater current (2 A range)
3: Heater current (0.5 A range)
4: Heater current (0.1 A range)
5: Heater current (dither)

Unused components
RN272C 10X4D
C274C 0.01 µF x 4
RN273C 4.7KX4D
RN272B 10X4D
C274B 0.01 µF x 4
RN273B 4.7KX4D

Current control DAC

+5VANA
C240 .1U
U240
CLK DIN CS DOUT REF VCC VOUT GND
LTC1655CS8
TP240 Test point
TP280 Test point
TP201 Test point
TP202 Test point
TP203 Test point
DAC_CS*
ANA_ADC_F0
ADC_CS*
ANA_SCK
ANA_MISO
ANA_MOSI

Constant-current circuits

1.333 A

Output enable
U290A DG333A
IN1 NO1 COM1 NC1
0.0V = 0A
4.02 V = 2A
0.75 nA on-leakage
Substitute MAX394 for lower leakage
+5VANA
U233 LTC6102
OUT V+ VREG -INS -INF +IN V- V-
0-4V
R231 4.02K
High side current sense w/20.1x gain
C233 .1U
+50V
R234 200
1 mA
R255 200
R232 2.49K
D251 1N5241B
R251 0.15 1W Sense resistor 50 ppm
VGS=20V
Q251 IRF6218S
36 kHz BW
RN291A 10KX4D
RN292D 4.7KX4D
C232A 1000 pF x 4
U250A LMC6484IM
C245D 0.1 µF x 4
Q233 BSS123LT1
VGS=20V
R233 1.00K
max 3V

0.465 A

U290B DG333A
IN2 NO2 COM2 NC2
+5VANA
U234 LTC6102
R235 4.02K
C234 .1U
+50V
R238 200
R256 200
R236 2.49K
D252 1N5241B
R252 0.43 1W Sense resistor
Q252 IRF6218S
RN291B 10KX4D
RN292C 4.7KX4D
C232B 1000 pF x 4
U250B LMC6484IM
C245C 0.1 µF x 4
Q234 BSS123LT1
R237 1.00K

0.200 A

+5VANA
U235 LTC6102
U290C DG333A
IN3 NO3 COM3 NC3
R239 4.02K
C235 .1U
+50V
R242 200
R257 200
R240 2.49K
D253 1N5241B
R253 1.0 1W Sense resistor
Q253 IRF6218S
RN291C 10KX4D
RN292B 4.7KX4D
C232C 1000 pF x 4
U250C LMC6484IM
C245B 0.1 µF x 4
Q235 BSS123LT1
R241 1.00K

All three circuits are used in parallel when the 2.0A range is selected; the 0.465A and 0.2A circuits are used in parallel when the 0.6A range is selected

U250D LMC6484IM
U290D DG333A

2000MA_ONOFF
200MA_ONOFF
20MA_ONOFF
2000MA_MON
200MA_MON
20MA_MON

Reference voltages

+8VFILT
C271 2.2 µF
U271
NR IN OUT TRIM NC1 NC2 NC3 GND
MAX6241BCSA
C272 1U
C273 1U
4VREF
R273 20.0K
R274 1.00K
R271 MPM200
10k
2VREF
U270D LMC6484IM
0.2VREF
Not used

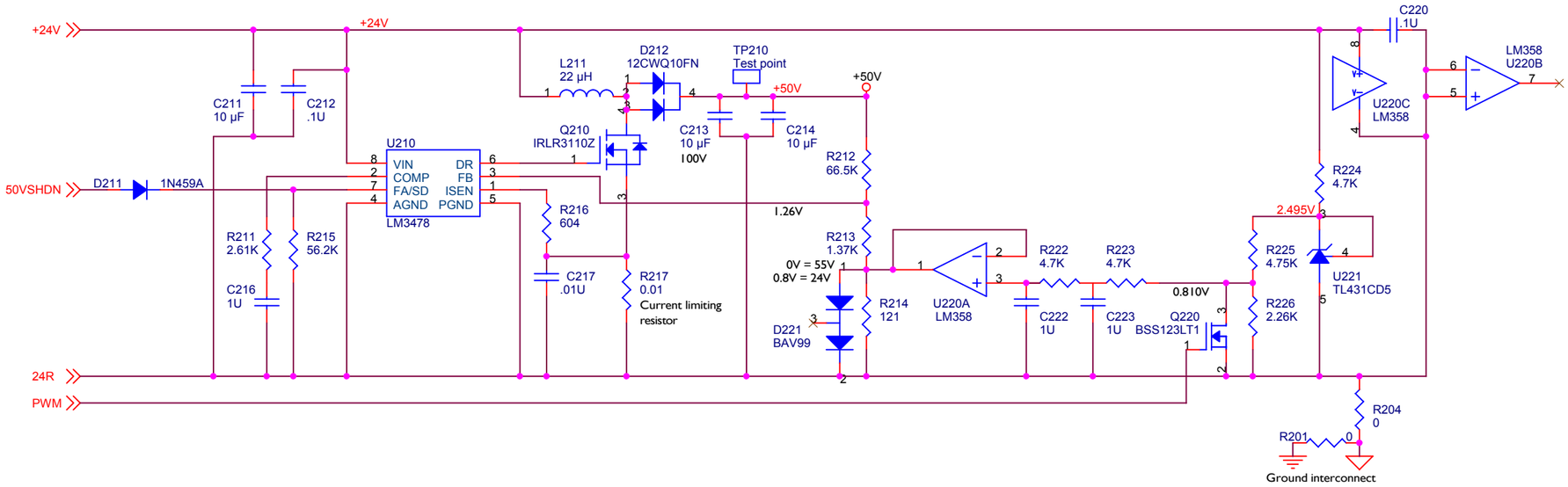
+5VANA
C250 .1U
U250E LMC6484IM
C251 .1U
-5VANA
+5VANA
C270 .1U
U270E LMC6484IM
C277 .1U
-5VANA
+5VANA
C290 .1U
U290E DG333A

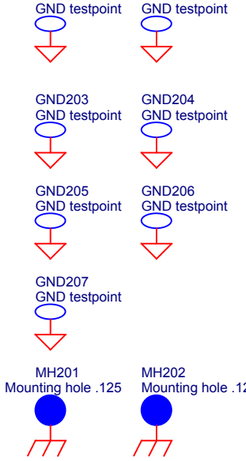
	Stanford Research Systems		
Title	PTC4314 DC output card: Analog		
Size	Document Number		Rev D
Date:	Friday, June 24, 2011	Sheet 2 of 4	

+24V to +50V 2 A boost regulator

Output voltage is controlled by
microcontroller's PWM output

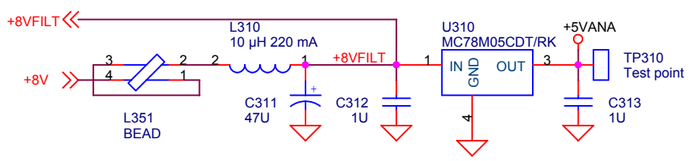
Switching frequency = 300.9 kHz
Duty cycle = 0.583
Input capacitor ripple current = 0.59A
Output capacitor ripple current = 2.34A
Input current = 4.8A
Output ripple voltage = 0.11V



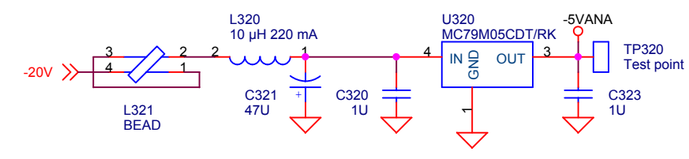


Analog supplies

Analog +5V regulator (0.2 A)

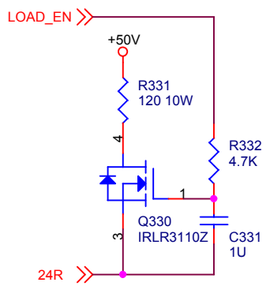


Analog -5V regulator (0.2 A)



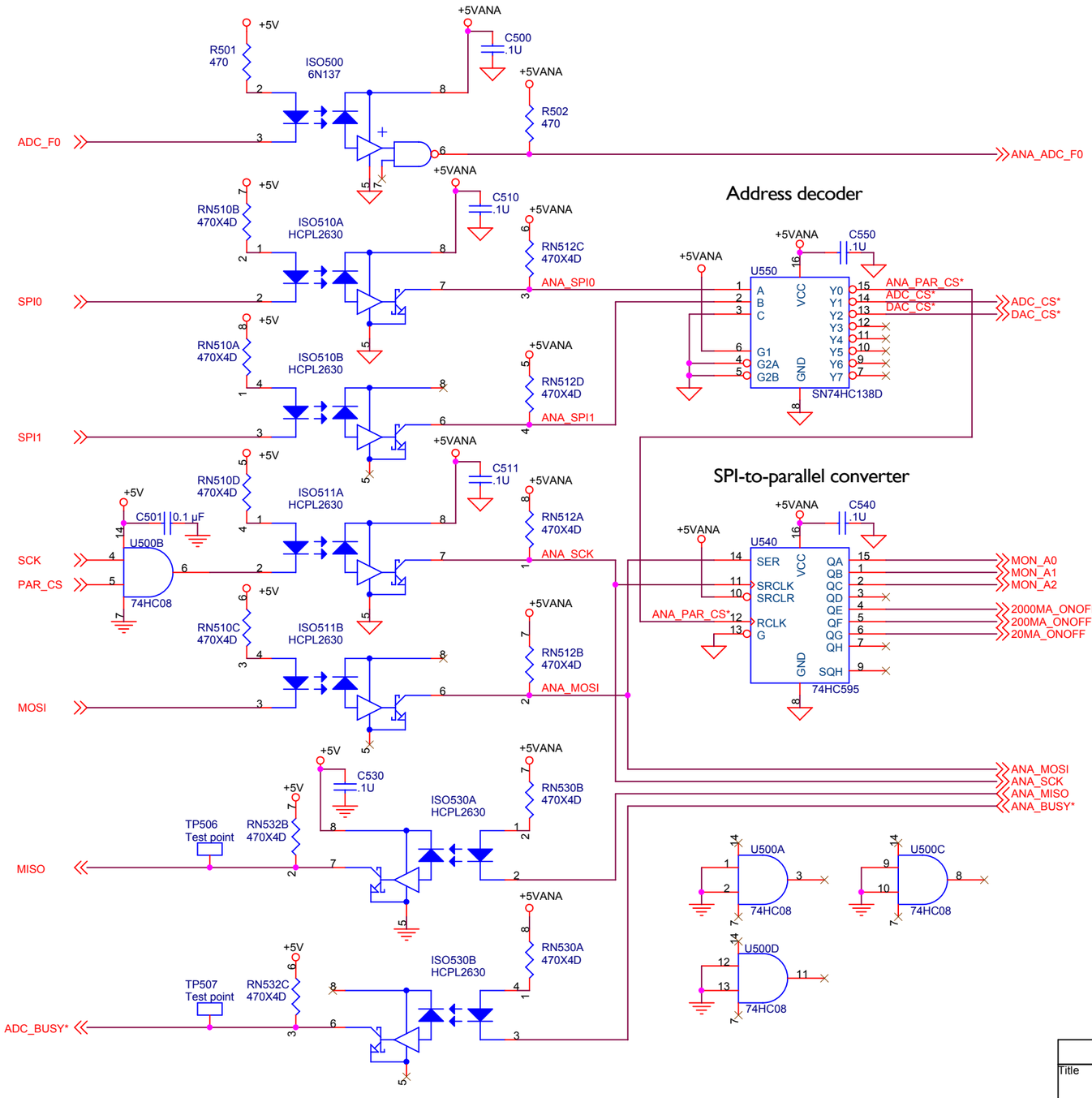
Internal load

Ensures minimum current
requirements are met



Stanford Research Systems		
Title: PTC4314 DC output card: Power		
Size	Document Number	Rev D
Date: Friday, June 24, 2011	Sheet 3 of 4	

Optoisolators



Address decoder

SPI-to-parallel converter

Title	PTC4314 DC output card: Isolators	
Size: Custom<Doc>	Document Number	Rev: D
Date: Friday, June 24, 2011	Sheet 4 of 4	

Microcontroller

Bidirectional transceivers

Parallel port (96-pin connector)

+8V
+20V
-20V
PPRESET*
PPCLK

JTAG port

J101
TCK
TDO
TMS
+5V
TDI
GND
VTREF
PPRESET*
GND
10 PIN DIL

U101
ATmega162-16AI
XTAL1
XTAL2
RESET
PPCLK
PPRESET*
PA0 / AD0 / PCINT0
PA1 / AD1 / PCINT1
PA2 / AD2 / PCINT2
PA3 / AD3 / PCINT2
PA4 / AD4 / PCINT4
PA5 / AD5 / PCINT5
PA6 / AD6 / PCINT6
PA7 / AD7 / PCINT7
PB0 / OC0 / T0
PB1 / OC2 / T1
PB2 / RXD1 / AIN0
PB3 / TXD1 / AIN1
PB4 / SS / OC3B
PB5 / MOSI
PB6 / MISO
PB7 / SCK
PC0 / A8 / PCINT8
PC1 / A9 / PCINT9
PC2 / A10 / PCINT10
PC3 / A11 / PCINT11
PC4 / A12 / PCINT12 / TCK
PC5 / A13 / PCINT13 / TMS
PC6 / A14 / PCINT14 / TDO
PC7 / A15 / PCINT15 / TDI
PD0 / RXD0
PD1 / TXD0
PD2 / INT0 / XCK1
PD3 / INT1 / ICP3
PD4 / TOSC1 / OC3A / XCK0
PD5 / TOSC2 / OC1A
PD6 / WR
PD7 / RD
PE0 / INT2 / ICP1
PE1 / ALE
PE2 / OC1B
C101 0.1 µF
C102 0.1 µF
C103 0.1 µF

U102
74ABT16245CMTD
D0 D1 D2 D3 D4 D5 D6 D7
CS*
MRDY*
R/W*
PPSRDY*
PPMISO
PMOSI
PSCK
PPR/W*
PPCS*
T/R1
OE1
T/R2
OE2
B0 PPD0
B1 PPD1
B2 PPD2
B3 PPD3
B4 PPD4
B5 PPD5
B6 PPD6
B7 PPD7
B8 PPCS*
B9 PPMRDY*
B10 PPR/W*
B11 SRDY*
B12 MISO
B13 PPMOSI
B14 PPSCK
B15
C105 0.1 µF
C106 0.1 µF
C107 0.1 µF
C108 0.1 µF

D[0:7]
PPD[0:7]

JDR101A
JDR101B
JDR101C
DIN96_ABC/R

RN101A 100Kx4D 5%
RN101C 100Kx4D 5%
RN101D 100Kx4D 5%
SPI_EN*
MOSI
MISO
SCK
SPI_ADD0
SPI_ADD1
ADC_BUSY*
LED*
RN103A 4.7KX4D
RN103B 4.7KX4D
RN102A 4.7KX4D
RN102B 4.7KX4D

Status LED

R101 300
D101 RED

RS-232 (for development purposes only)

U104
MAX3235ECWP
C113 .1U/T35
J103
9 PIN D
TXD
RXD
TX
RX
T1IN
R1OUT
T2IN
R2OUT
INVALID
FORCEOFF
FORCEON
T1OUT
R1IN
T2OUT
R2IN
C1+
C1-
C2+
C2-
V1+
V2+
V-

RN105A 47k
RN105B 47k
RN105C 47k
RN105D 47k
PPSRDY*
PPSCK
PPCS*
PPMOSI

	Stanford Research Systems
Title	PTC5106 analog IO card: CPU
Size	Document Number — Rev F
Date:	Tuesday, July 19, 2011 — Sheet 1 of 3

5 V reference
±15 mA

+15V
C201 2.2U/T16
U201
C202 1U
NR
IN
OUT
NC1
NC2
NC3
GND
TRIM
C203 2.2U/T16
MAX6250BCSA
TP201 1 PIN
+5VREF
R201 15.0K
R202 10.0K

Convert 0 – 5V to ±10V
34 Hz rolloff

C206 .47U
R203 10.0K
R204 2.49K
U203C ¼ MC33079D
C209 .47U
R205 10.0K
R206 2.49K
U203B ¼ MC33079D
C210 .47U
R207 10.0K
R208 2.49K
+15V
C207 0.1 µF
U203A ¼ MC33079D
C208 0.1 µF
-15V
C211 .47U
R209 10.0K
R210 2.49K
U203D ¼ MC33079D

Switches
45 ohms, 30 mA

MAX4661: 2.5 ohm, 200 mA, different package

MAX4664: 5 ohm, 100 mA, same package

ANA_I/O*1
U204D DG411DY
ANA_I/O*2
U204C DG411DY
ANA_I/O*3
U204B DG411DY
ANA_I/O*4
U204A DG411DY

Deglitching filters

Prevents charge injection from ADC multiplexer from introducing glitches on DAC output
Also acts as a 2 kHz lowpass filter; source impedance must be under 100 kOhm to keep lowpass cutoff above 100 Hz

RN206D 1.0KX4D
RN206C 1.0KX4D
RN206B 1.0KX4D
RN206A 1.0KX4D
C216D 0.1 µF x 4
C216C 0.1 µF x 4
C216B 0.1 µF x 4
C216A 0.1 µF x 4

Multiplexer

+15V -15V
D203 BAV70LT1
D204 BAV70LT1
C225 0.1 µF
C226 0.1 µF
U206
S1A S2A S3A S4A
SB1 SB2 SB3 SB4
V+ V-
EN
A0 A1
DA DB
GND
+5VANA
DG409/SO
Alternative: MAX339CSE

Input buffer

TP211 1 PIN
+15V
C215 0.1 µF
U205A ¼ MC33079D
C217 0.1 µF
-15V

ANA_SEL0
ANA_SEL1

Convert ±10V to 0 – 5V
3 MHz rolloff

+5VREF
R230 1.69K
R231 845
R234 2.55K
R227 3.01K
R228 1.00K
3.33V
C227 330P
U205C ¼ MC33079D
U205B ¼ MC33079D
TP212 1 PIN
D246 MMBZ5232BLT1 ADC input protection

Single-channel ADC

Set In- to 2.5 V, since (In+) - (In-) must be between -2.5V and +2.5V

U205D ¼ MC33079D
2.5V
+5VREF
+5VANA
C232 .1U
C233 10U/T16
U209
IN+ IN-
REF+ REF-
VCC
F0 EXT
CS SCK SDI SDO BUSY
GND GND GND GND
ANA_ADC_CS*
ANA_SCK
ANA_MOSI
ANA_MISO
ANA_BUSY*
LTC2440CGN

4-channel DAC

+5VANA
C204 .1U
C205 10U/T16
U202
SYNC SCLK DIN
AVDD IOVDD
VOUTA VOUTB VOUTC VOUTD
A0 A1
LDAC ENABLE
VREFH VREFL
GND
DAC8534IPW
ANA_DAC_CS*
ANA_SCK
ANA_MOSI
+5VREF

D201 BAV70LT1
D202 BAV70LT1
+15V -15V
+5VANA
C212 0.1 µF
C213 0.1 µF
C214 0.1 µF
V+ V- VL GND
Power input for switches
U204E DG411DY

to BNCs
V1
V2
V3
V4

to isolators
ANA_DAC_CS
ANA_I/O*[1:4]
ANA_BUSY*
ANA_MISO
ANA_MOSI
ANA_SCK
ANA_ADC_CS
ANA_SEL0
ANA_SEL1
ANA_SYNC

ANA_I/O*[1:4]

Stanford Research Systems		
Title: PTC5106 analog IO card: Data converters		
Size	Document Number	Rev F
Date: Tuesday, July 19, 2011	Sheet 2 of 3	

BNC connectors

with overvoltage protection

All PTC fuses
0.2A 15V
2.5 ohms

J301 BNC — F301 1206L020YRT — D301 12V — R301 10M — V1
J302 BNC — F302 1206L020YRT — D302 12V — R302 10M — V2
J303 BNC — F303 1206L020YRT — D303 12V — R303 10M — V3
J304 BNC — F304 1206L020YRT — D304 12V — R304 10M — V4

Optoisolators

ISO310A HCPL2630 — RN310B 470X4D — C310 .1U — RN312C 470X4D — SPI_ADD0 — ANA_SPI0
ISO310B HCPL2630 — RN310A 470X4D — RN312D 470X4D — SPI_ADD1 — ANA_SPI1
ISO311A HCPL2630 — RN310D 470X4D — C311 .1U — RN312A 470X4D — SCK/MOSI — ANA_SCK
ISO311B HCPL2630 — RN310C 470X4D — RN312B 470X4D — SPI_EN* — ANA_MOSI
ISO330A HCPL2630 — RN332B 470X4D — C330 .1U — RN330B 470X4D — TP306 — ANA_MISO
ISO330B HCPL2630 — RN332C 470X4D — RN330A 470X4D — TP307 — ADC_BUSY* — ANA_BUSY*

U331 74HC541 — C331 .1U — TP300 Test point

Tri-state buffer ensures that MISO is in high-Z state during SPI programming.

SPI_EN*: High = disabled, Low = enabled

TP301 Test point — TP302 Test point

Address decoder

U302 SN74HC138D — C302 0.1 µF

TP303 Test point — TP304 Test point — TP305 Test point

ANA_PAR_CS*
ANA_ADC_CS*
ADA_DAC_CS*

ANA_ADC_CS*
ANA_DAC_CS*

SPI-to-parallel converter

U340 74HC595 — C340 0.1 µF

ANA_SEL0
ANA_SEL1
ANA_I/O*1
ANA_I/O*2
ANA_I/O*3
ANA_I/O*4
ANA_I/O*[1:4]

ANA_I/O*[1:4]

Input/output status indicators

10 mA, 1.8V high efficiency red LEDs

D341 RED — RN341D 270X4D
D342 RED — RN341C 270X4D
D343 RED — RN341B 270X4D
D344 RED — RN341A 270X4D

ANA_MOSI
ANA_SCK
ANA_MISO
ANA_BUSY*

Analog regulators

Debug power connector

For development purposes only

J308 — 10 PIN, WHITE
+8VFILT
+20VFILT
AGND
-20VFILT

Analog +5V regulator (0.2 A)

+8V — L351 BEAD — L352 10 µH 220 mA — C351 47U — C352 .33U/T35 — +8VFILT
U350 MC78M05CDT/RK — C353 1U-16V — TP350 Test point — +5VANA

+15V regulator (0.2 A)

+20V — L361 BEAD — L362 10 µH 220 mA — C361 47U — C362 .33U/T35 — +20VFILT
U360 MC78M15CDT/RK — C363 1U-16V — TP360 Test point — +15V

-15V regulator (0.2 A)

-20V — L371 BEAD — L372 10 µH 220 mA — C371 47U — C372 .33U/T35 — -20VFILT
U370 MC79M15CDT/RK — C373 1U-16V — TP370 Test point — -15V

Also try 470 µF caps for C351, C361, and C371

R391 DNI
R392 DNI
R393 DNI
R394 DNI

GND300 GND testpoint
GND301 GND testpoint
GND302 GND testpoint
GND303 GND testpoint
GND304 GND testpoint
GND305 GND testpoint
GND306 GND testpoint
GND307 GND testpoint
GND308 GND testpoint
GND309 GND testpoint

	Stanford Research Systems
Title	PTC5106 analog IO card: Inputs, regulators, & isolators
Size	Document Number — Rev F
Date:	Tuesday, July 19, 2011 — Sheet 3 of 3

Microcontroller

Bidirectional transceivers

Parallel port (96-pin connector)

Debug power connector

For development purposes only

J100
10 PIN, WHITE

JTAG port

J111
10 PIN DIL

U110
ATmega162-16AI

U120
74ABT16245

JDR121A
DIN96_ABC/R

JDR121B
DIN96_ABC/R

JDR121C
DIN96_ABC/R

RS-232 (for development purposes only)

U130
MAX3235ECWP

J131
9 PIN D

Status LED

D111
RED MINI

GND101 GND testpoint
GND102 GND testpoint
GND103 GND testpoint
GND104 GND testpoint
GND105 GND testpoint
GND106 GND testpoint

Stanford Research Systems

Title: PTC5204 Digital IO card: CPU

Size | Document Number | Rev D

Date: Monday, December 27, 2010 | Sheet 1 of 4

Termination, filtering, and ESD protection

J200 — D0 = LSB, D7 = MSB

25 PIN D RS232

STROBE* 1
AUTOFD* 14
D0 2
ERROR* 15
D1 3
INIT* 16
D2 4
SELECTIN* 17
D3 5
D4 18
D5 6
D6 19
D7 7
ACK* 20
BUSY 8
PAPEREND 21
SELECT* 22
23
24
25

Should really be 33 ohms

RN200D 4 — 5 47X4D
RN200C 3 — 6 47X4D
RN200B 2 — 7 47X4D
RN200A 1 — 8 47X4D
RN201D 4 — 5 47X4D
RN201C 3 — 6 47X4D
RN201B 2 — 7 47X4D
RN201A 1 — 8 47X4D
RN202B 2 — 7 4.7KX4D
RN202A 1 — 8 4.7KX4D

+5VISO

C200 180P
C201 180P
C202 180P
C203 180P
C204 180P
C205 180P
C206 180P
C207 180P

D200 5.6V Zener
D202 5.6V Zener
D204 5.6V Zener
D206 5.6V Zener

GND201 GND testpoint
GND202 GND testpoint
GND203 GND testpoint
GND204 GND testpoint

Parallel-to-SPI converter

U210 — 74HC166

+5VISO
C210 .1U
+5VISO

RD0 2 — A
RD1 3 — B
RD2 4 — C
RD3 5 — D
RD4 10 — E
RD5 11 — F
RD6 12 — G
RD7 14 — H

VCC 16
CS 6
CLK 7
SEROUT 13
SH/LD 15
CLR 9
GND 8
SERIN 1

TP201 Test point — ISCK
TP202 Test point — IMISO
TP203 Test point — IMOSI
TP204 Test point — IDIN_CS*
TP205 Test point — IDOUT_CS*

SPI-to-parallel converter

U220 — 74HC595

+5VISO
C220 .1U
+5VISO

QA 15
QB 1
QC 2
QD 3
QE 4
QF 5
QG 6
QH 7
SQH 9

VCC 16
SER 14
SRCLK 11
SRCLR 10
RCLK 12
G 13
GND 8

IDOUT_CS*

Optoisolators

ISCK
IDIN_CS*
IDOUT_CS*
IMOSI
IMISO

+5VISO
C230 .1U
RN231B 470X4D
ISO230A HCPL2630
+5V
RN232B 470X4D
BSCK
TP211 SCK

+5VISO
RN231C 470X4D
ISO230B HCPL2630
+5V
RN232A 470X4D
BMOSI
TP213 MOSI
TP212 MISO

+5VISO
RN231A 470X4D
ISO240 6N137
+5V
C240 .1U
R241 4.7K
BMISO
74HC04
U420F

+5VISO
C250 .1U
RN251B 470X4D
ISO250A HCPL2630
+5V
RN252B 470X4D
DIN_CS*

+5VISO
RN251C 470X4D
ISO250B HCPL2630
+5V
RN252A 470X4D
DOUT_CS*

+5VISO
C260 .1U
ISO260 6N137
+5V
RN252C 470X4D
DOUT_EN*

+5VISO
R261 4.7K
+5VISO
R262 4.7K

TP214 DIN_CS*
TP215 DOUT_CS*
DIN_CS*
DOUT_CS*

+5V
C270 .1U
U270 — 74HC541

Y1 18
Y2 17
Y3 16
Y4 15
Y5 14
Y6 13
Y7 12
Y8 11
VCC 20
A1 2
A2 3
A3 4
A4 5
A5 6
A6 7
A7 8
A8 9
GND 10
G1 1
G2 19

SCK
MOSI
High = disabled
Low = enabled
SPI_EN*

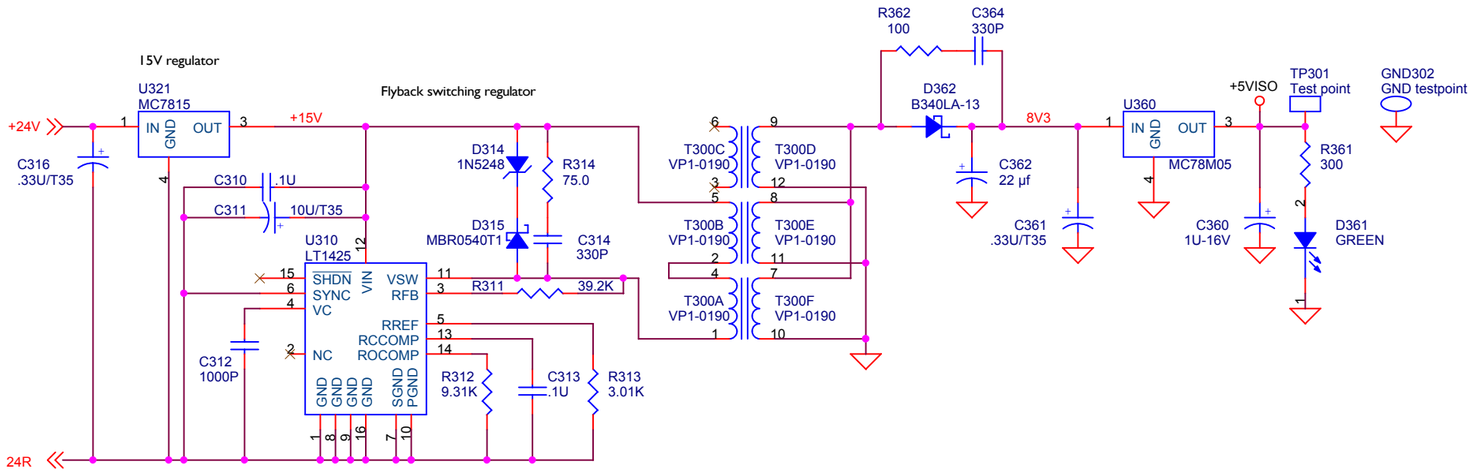
Tri-state buffer ensures that MISO is in high-Z state during SPI programming.

MISO
DOUT_EN*

GND211 GND testpoint

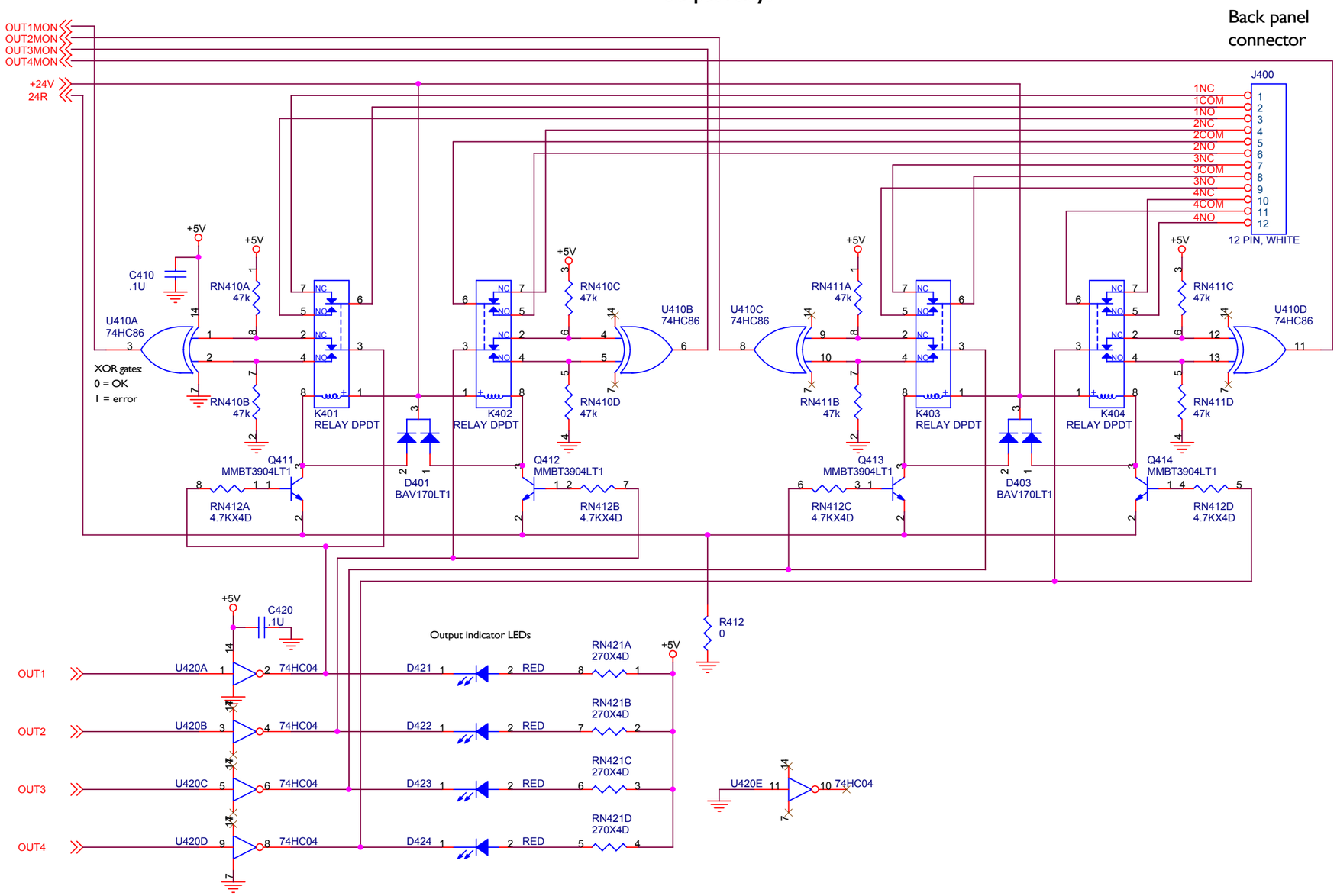
	Stanford Research Systems	
Title	PTC5204 digital IO card: DIO	
Size	Document Number	Rev D
Date: Monday, December 27, 2010	Sheet 2 of 4	

Isolated +5V power supply



	Stanford Research Systems	
Title	PTC5204 Digital I/O card: Power	
Size	Document Number	Rev D
Date: Monday, December 27, 2010	Sheet 3 of 4	

Output relays



	Stanford Research Systems	
Title	PTC5204 digital IO card: Relays	
Size	Document Number	Rev D
Date: Monday, December 27, 2010	Sheet 4 of 4	