PS200/CH200 12 V Charging Regulators

User Guide

Issued 22.1.13

CSL 910

Guarantee

This equipment is guaranteed against defects in materials and workmanship. This guarantee applies for twelve months from date of delivery. We will repair or replace products which prove to be defective during the guarantee period provided they are returned to us prepaid. The guarantee will not apply to:

- Equipment which has been modified or altered in any way without the written permission of Campbell Scientific
- Batteries
- Any product which has been subjected to misuse, neglect, acts of God or damage in transit.

Campbell Scientific will return guaranteed equipment by surface carrier prepaid. Campbell Scientific will not reimburse the claimant for costs incurred in removing and/or reinstalling equipment. This guarantee and the Company's obligation thereunder is in lieu of all other guarantees, expressed or implied, including those of suitability and fitness for a particular purpose. Campbell Scientific is not liable for consequential damage.

Please inform us before returning equipment and obtain a Repair Reference Number whether the repair is under guarantee or not. Please state the faults as clearly as possible, and if the product is out of the guarantee period it should be accompanied by a purchase order. Quotations for repairs can be given on request. It is the policy of Campbell Scientific to protect the health of its employees and provide a safe working environment, in support of this policy a "Declaration of Hazardous Material and Decontamination" form will be issued for completion.

When returning equipment, the Repair Reference Number must be clearly marked on the outside of the package. Complete the "Declaration of Hazardous Material and Decontamination" form and ensure a completed copy is returned with your goods. Please note your Repair may not be processed if you do not include a copy of this form and Campbell Scientific Ltd reserves the right to return goods at the customers' expense.

Note that goods sent air freight are subject to Customs clearance fees which Campbell Scientific will charge to customers. In many cases, these charges are greater than the cost of the repair.

Campbell Scientific Ltd,
Campbell Park, 80 Hathern Road,
Shepshed, Loughborough, LE12 9GX, UK
Tel: +44 (0) 1509 601141
Fax: +44 (0) 1509 601091
Email: support@campbellsci.co.uk
www.campbellsci.co.uk

PLEASE READ FIRST

About this manual

Please note that this manual was originally produced by Campbell Scientific Inc. primarily for the North American market. Some spellings, weights and measures may reflect this origin.

Some useful conversion factors:

Area: 1 in^2 (square inch) = 645 mm^2

Length: 1 in. (inch) = 25.4 mm
1 ft (foot) = 304.8 mm
1 yard = 0.914 m
1 mile = 1.609 km

Mass: 1 oz. (ounce) = 28.35 g
1 lb (pound weight) = 0.454 kg

Pressure: 1 psi (lb/in^2) = 68.95 mb

Volume: 1 UK pint = 568.3 ml
1 UK gallon = 4.546 litres
1 US gallon = 3.785 litres

In addition, while most of the information in the manual is correct for all countries, certain information is specific to the North American market and so may not be applicable to European users.

Differences include the U.S standard external power supply details where some information (for example the AC transformer input voltage) will not be applicable for British/European use. *Please note, however, that when a power supply adapter is ordered it will be suitable for use in your country.*

Reference to some radio transmitters, digital cell phones and aerials may also not be applicable according to your locality.

Some brackets, shields and enclosure options, including wiring, are not sold as standard items in the European market; in some cases alternatives are offered. Details of the alternatives will be covered in separate manuals.

Part numbers prefixed with a "#" symbol are special order parts for use with non-EU variants or for special installations. Please quote the full part number with the # when ordering.

Recycling information

At the end of this product's life it should not be put in commercial or domestic refuse but sent for recycling. Any batteries contained within the product or used during the products life should be removed from the product and also be sent to an appropriate recycling facility.

Campbell Scientific Ltd can advise on the recycling of the equipment and in some cases arrange collection and the correct disposal of it, although charges may apply for some items or territories.

For further advice or support, please contact Campbell Scientific Ltd, or your local agent.

CAMPBELL SCIENTIFIC
WHEN MEASUREMENTS MATTER

Campbell Scientific Ltd, Campbell Park, 80 Hathern Road, Shepshed, Loughborough, LE12 9GX, UK
Tel: +44 (0) 1509 601141 Fax: +44 (0) 1509 601091
Email: support@campbellsci.co.uk
www.campbellsci.co.uk

Contents

PDF viewers note: These page numbers refer to the printed version of this document.
Use the Adobe Acrobat® bookmarks tab for links to specific sections.

List of Figures

List of Tables

PS200/CH200 12 V Charging Regulators

1. Precautions and Tips

Under normal charging conditions with sealed VRLA batteries, hydrogen and oxygen gasses are produced in relatively small quantities, most of which later recombines back into water. Aggressive overcharging produces excess hydrogen and oxygen gasses, resulting in gas venting by means of a pressure activated valve. Hydrogen gas emitted from VRLA batteries must not be allowed to accumulate, as it could form an explosive mixture. Fortunately, hydrogen gas is difficult to contain in anything but a metal or glass enclosure.

WARNING **Never put VRLA batteries in an enclosure that does not allow emitted hydrogen gas to be dispersed.**

VRLA batteries are capable of providing high surge currents. The 12 V output terminals of the PS200/CH200 are fused with a 4 A self-resettable thermal fuse, but there is no fusing for inadvertent bridging of the battery terminals. Accidental shorting of battery terminals by metallic objects, such as watchbands, can cause severe burns due to rapid heating and is also a fire hazard.

VRLA battery manufacturers state that "Heat Kills Batteries". While the PS200/CH200 can operate from –40°C to +60°C, optimum battery life is achieved with battery operating temperatures ranging from 5°C to 35°C[1], per manufacturer's recommendations[1]. The PS200/CH200 offer temperature compensation of the battery charging voltage based on a temperature measurement inside the PS200/CH200 cases. The CH200 internal temperature measurement likely will not accurately represent battery temperature for charge voltage compensation unless the battery is in close proximity to the CH200. The PS200/CH200 serial interface can be used to input an independently measured battery temperature for improved charging temperature compensation when the charger temperature is different than the battery temperature (see Section 5.2.3.1 Write Remote Battery Temperature to PS200/CH200 and Section 5.2.3.2 Restore Internal Battery Temperature Measurement for SDI-12 programming examples and Section 5.3.3.2 Write Remote Battery Temperature to PS200/CH200 and Section 5.3.3.3 Restore Internal Battery Temperature Measurement for RS-232 programming examples).

With rechargeable batteries a charge → discharge → re-charge event is termed a cycle. In general the most important factor for the service life of a battery is depth of discharge[1]. For example, decreasing the depth of each discharge from 100% to 50% approximately doubles the number of useful cycles available from the battery[1].

CAUTION Leaving a lead-acid battery in a discharged state for prolonged periods of time results in the undesirable growth of large sulphate crystals (sulphating) that are detrimental to battery performance.

VRLA batteries self-discharge at approximately 3% of rated capacity per month at room temperature[1]. A 3% of rated capacity per month self-discharge results in

[1] Genesis Application Manual – Genesis NP and NPX Series US-NP-AM-002, June 2006.

100% discharge in approximately 33 months (≈ 3 years) for a battery stored at room temperature. Self-discharge increasing with increasing storage temperature.

NOTE Periodic recharging of stored batteries every few months is recommended to prevent irreversible sulphation due to prolonged time in a discharged state.

2. Quick Start

The PS200/CH200 modules are designed to handle extreme conditions and have the ability to transmit charging, load, and battery voltage and current information directly to a datalogger using special SDI-12 or RS-232 commands and associated communication cables. Using datalogger programming the data from the PS200/CH200 can be used to calculate a power budget for the entire system and help to remotely pin point any power problems. These cables are not required for normal operation and the modules are ready to use right out of the box. The modules have been designed with mounting holes on one inch centres for mounting to a standard Campbell Scientific enclosure back plate – see the enclosure manual for mounting suggestions. See Figures 2-1 through 2-3 for typical enclosure installations using a PS200/CH200.

By default the CH200 module is programmed with a battery capacity of zero Amp-hours (Ah). This sets the charger to charge at a lower current rate. A battery capacity must be configured into the CH200 to enable the more aggressive two-step constant voltage charging scheme. See Section 5.1.2 Settings Editor Tab for making changes using Device Configuration Utility and Appendix A.2 Change Battery Capacity using the datalogger.

NOTE Figures 2-1 through 2-3 show PS200/CH200 installations using a SDI-12 communication cable. A SDI-12 communication cable is not required for normal operation.

Figure 2-1. The PS200 connected to a CR1000 and AC power.

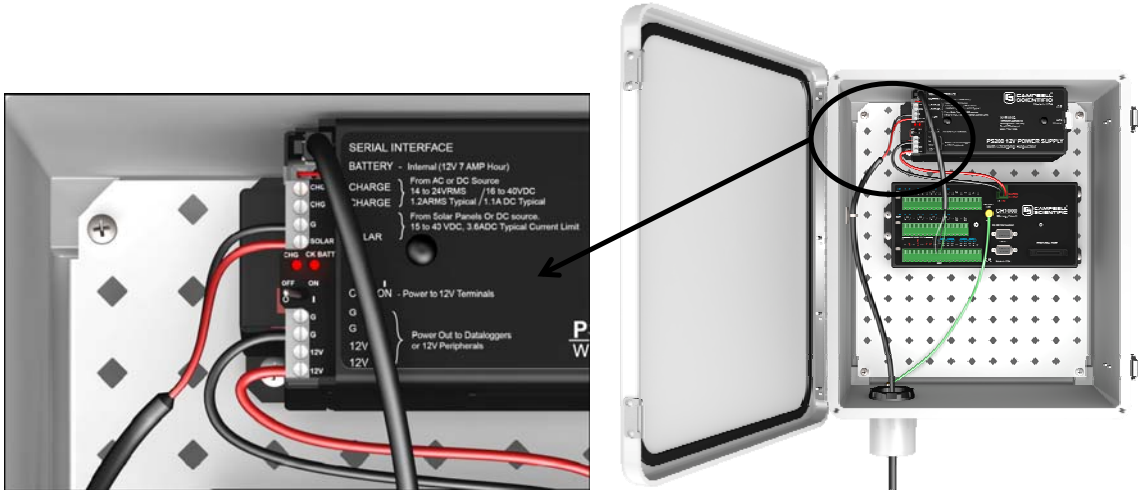


Figure 2-2. The PS200 connected to a CR1000 and solar panel.

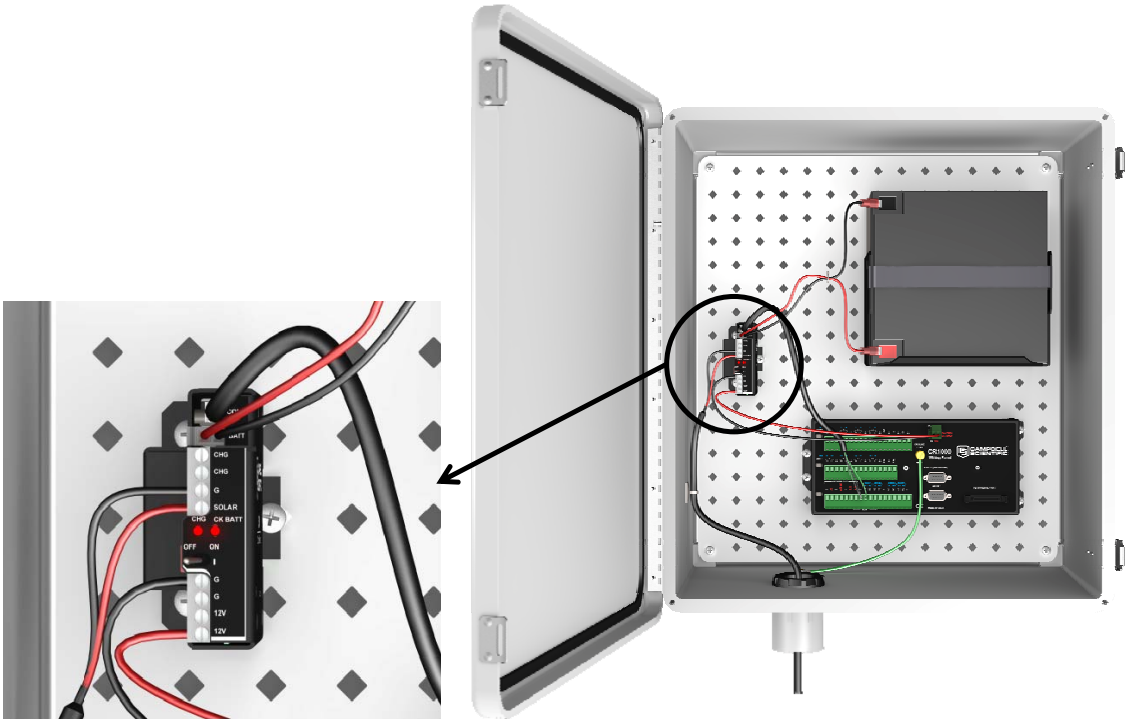


Figure 2-3. CH200 connected to BP24 battery pack and CR1000.

2.1 Connecting Power

WARNING **Although the power supply and battery are low voltage they do have the ability to supply a high current and could potentially heat up a metal ring, watch band, or bracelet enough to burn skin or melt metal when shorted. Remove rings, watches, or bracelets before hooking up power and connecting a battery.**

Unlike earlier Campbell Scientific power supplies, the CH/PS200 can have both solar and AC power hooked up simultaneously.

Flip the power supply switch to "Off" before hooking up power to the power supply.

NOTE

The switch on the CH/PS200 only controls power going to the "12V" and "G" terminal blocks. The battery is continuously charged regardless of the switch setting as long as a charging voltage is present.

2.1.1 Solar Panel

WARNING

To prevent sparking while wiring up the solar panel, either lay the solar panel face down on its packing box or cover it with something fairly opaque to block the sunlight while wiring up the panel.

Connect the BLACK (negative) lead from the solar panel to the terminal block marked "G" that is directly adjacent to the "SOLAR" terminal block. Connect the RED (positive) lead from an unregulated solar panel to the terminal block marked "SOLAR". See Figure 2-4.

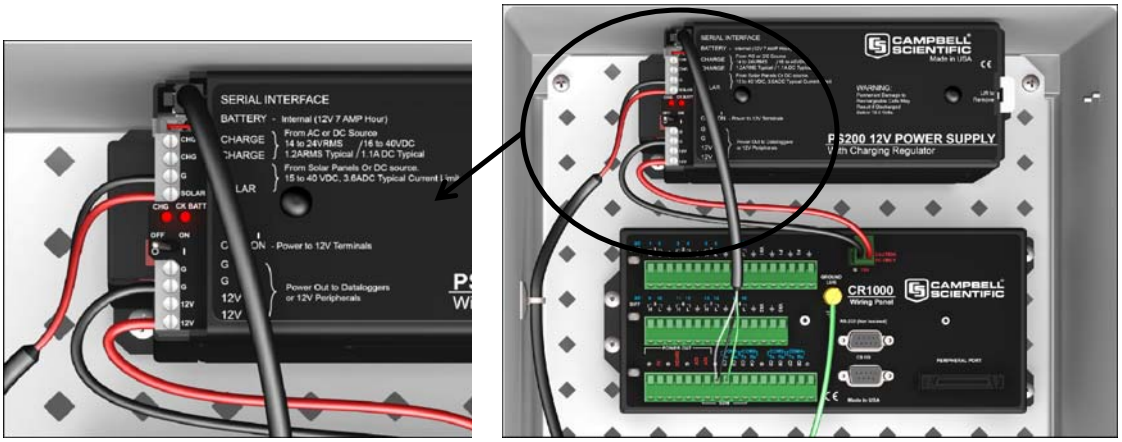


Figure 2-4. Solar Panel Connections on PS200

2.1.2 AC/DC Power

Double check the input voltages coming in to the charger/regulator with a volt meter.

AC Input Voltage: 14 to 24 VAC RMS
DC Input Voltage: 15 to 40 VDC

WARNING

Exceeding the voltages listed above will damage the power supply.

Disconnect the primary side of the AC/DC power before connecting wires to the PS200.

Connect the secondary power supply leads to the two terminal blocks marked "CHG". There is no polarity on the "CHG" terminal blocks, so it does not matter which wire goes to which "CHG" terminal block, but make sure there is only ONE wire per block. See Figure 2-5.

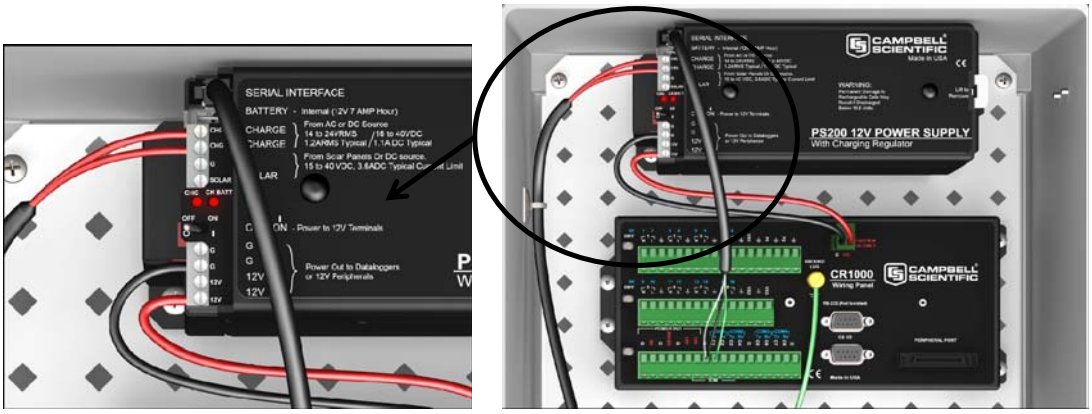


Figure 2-5. AC Power Connections on PS200

2.2 Plug In the Battery

The battery used with the PS200 is shipped inside of the PS200 case if the power supply is NOT installed inside an enclosure. If the PS200 is mounted inside an enclosure then the battery will be located separately packed in one of the packing boxes. This is done to minimize any damage that could occur if the power supply should get loose from its mounts inside the enclosure during shipment. The battery will NOT be plugged into the PS200. This is done to minimize discharging the battery.

To remove the lid from the PS200, pull up on the PS200 lid latch and slide the lid off as shown in Figures 2-6 and 2-7.

WARNING

Do not remove the tape holding the battery wiring harness to the top of the battery! The tape is used to keep the battery wiring harness out of the way of the rubber bumpers on the inside of the lid.

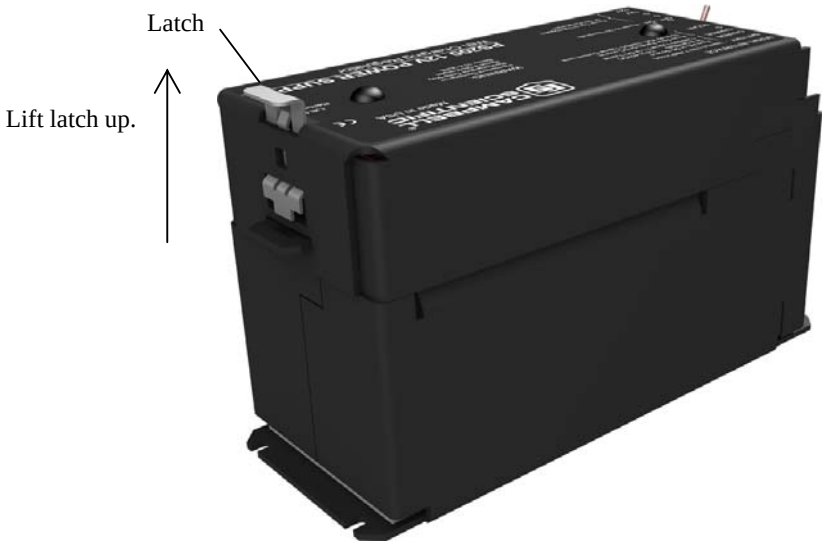


Figure 2-6. Lift latch up on PS200.

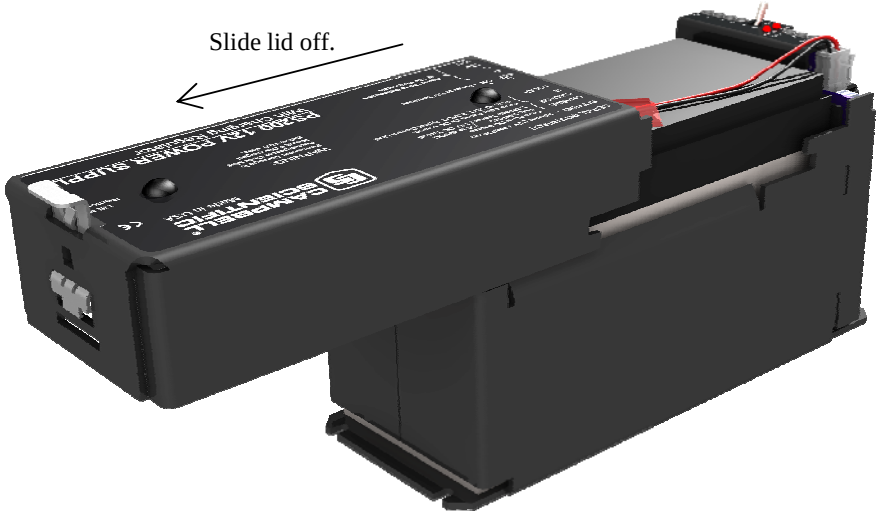


Figure 2-7. Slide PS200 lid off.

Plug the battery into the connector marked “BATT”. This connector is polarized and will only allow the mating connector to be plugged in one way. Push the connector all the way in until it locks in place.

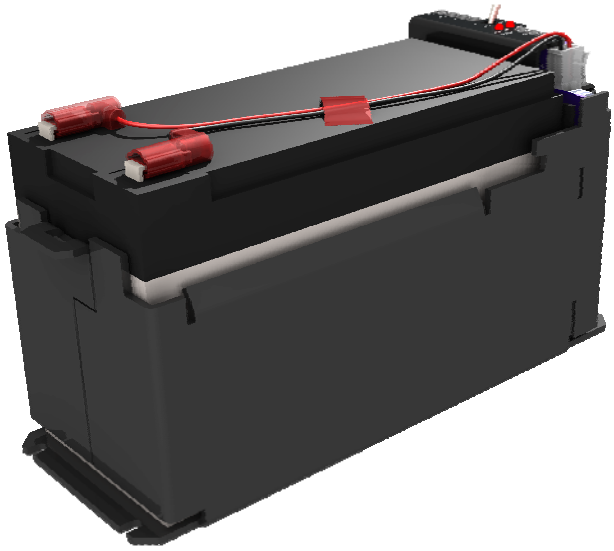


Figure 2-8. Wiring Harness Plugged into Battery Connector

NOTE When connecting the battery the "CHG" LED will briefly flash red and then go out.

2.3 Hook Up Power to Datalogger

Both the PS200 and the CH200 come with a 1 foot black wire attached to one of the **G** terminal blocks and a 1 foot red wire attached to one of the **12V** terminal blocks. Attach the red wire from the power supply to the datalogger Power terminal block marked 12V (Campbell Scientific part number 3768). Attach the black wire from the power supply to the datalogger Power terminal block marked "G".

2.4 Hook Up Communication Cable to Datalogger If Used

Skip this step if a CH/PS200 communication interface cable is not being used. Plug the power supply SDI-12 or RS-232 communication cable to the connector marked "COMM". This cable is polarized and will only plug in one way. Push the connector all the way into the mating connector until it locks in place. Wire the leads into the appropriate datalogger channels. An example of SDI-12 wiring is shown in Figure 2-9. An example of RS-232 wiring is shown in Figure 2-10.

NOTE This cable is NOT required for normal operation of the power supply! It is only required if the datalogger has been programmed to collect information from the power supply. See Section 5.2 SDI-12 and Section 5.3 RS-232 Interface for how to use these cables.

2.5 Turn On the Charging Source

Turn on the power going to the charging source or uncover the solar panel. The "CHG" LED should flash green approximately every 4 to 5 seconds if all incoming connections are correct and there is an adequate charging voltage present.

2.6 Turn On Power to the Datalogger

Flip the switch on the PS200/CH200 supply to "On". Verify voltage to the datalogger with a volt meter, or use a key pad display, or connect to the datalogger with a laptop or PDA to make sure the datalogger is running correctly.

2.7 LED Indicators

As previously mentioned, the PS200/CH200 has two LED indicators, the CHARGE LED and the CHECK BATTERY LED. The following tables illustrate the various conditions and associated colours for the CHARGE and CHECK BATTERY LEDs.

Table 2-1. CHARGE LED

Condition	Colour
No Valid Charge Source	Off
Valid Charge Source & Charging Battery	Flashing Green
Valid Charge Source but Battery being Discharged	Flashing Orange
Regulator Fault Detected	Flashing Red
Waiting for New Operating System (Section 5.2.2 SDI-12 Measurements)	Solid Red
Transferring Operating System	Solid Green

Table 2-2. CHECK BATTERY LED

Condition	Colour
Battery Voltage > 11.5 V	Off
10.5 V ≤ Battery Voltage ≤ 11.5 V	Flashing Orange
Battery Fault Detected OR Battery Voltage < 10.5 V	Flashing Red

3. General Description

The PS200 is a 12 volt power supply that includes a rechargeable 7 amp hour valve-regulated lead-acid (VRLA) battery and charging regulator. The CH200 is a charging regulator for an external rechargeable 12 V VRLA battery such as the BP12 or BP24 offered by Campbell Scientific, Inc. Charging power for these charging regulators is typically supplied by an unregulated solar panel, AC/AC transformer, or AC/DC converter.

The PS200/CH200 are smart chargers that provide two-step constant voltage charging with temperature compensation for optimal charging and battery life. A maximum power point tracking algorithm is incorporated for solar inputs to maximize available solar charging resources.

The PS200/CH200 are compatible with the A100 null-modem adapter and the A105 adapter for additional 12 V output terminals. The A100 Null Modem Adapter connects and powers two Campbell Scientific peripherals via two CS I/O 9-pin connectors configured as a null modem. This is useful in linking different communications technologies; e.g., telephone to radio, at sites that do not have a datalogger. The A105 Additional 12 V Terminals Adapter may be used to provide additional 12 V and ground terminals where the power supply is used to power several devices.

The PS200/CH200 charging regulators are termed series regulators, because the regulators are placed in series between the charging source and the load. As batteries become closer to fully charged, series regulators reduce the current drawn from the charging source, to where the charging source may be completely unloaded if full-charge is reached. While this unloading of the charging source is acceptable for solar panels, AC/AC transformers and AC/DC converters, it is undesirable for wind turbines because of the resulting free spinning when unloaded. Consequently, series charging regulators, including the PS200/CH200, should not be used to regulate the output of wind turbines without the inclusion of a way to load the turbine when the batteries require little or no charging current.

The PS200/CH200 chargers have several safety features intended to protect the charging source, battery, charger, and load devices. Both the SOLAR – G and CHARGE – CHARGE input terminals incorporate hardware current limits and polarity reversal protection. There is a 1.85 Amp failsafe self-resettable thermal fuse in series with the CHARGE – CHARGE inputs in the event of a catastrophic AC/AC or AC/DC charging source failure. There is a 4 Amp self-resettable thermal fuse in series with the 12 V output terminals of the charger in the event of an output load fault. The PS200/CH200 incorporate battery reversal protection, which is catastrophic for most chargers. ESD and surge protection are incorporated on all inputs and outputs of the PS200/CH200.

3.1 Specifications

(CHARGE - CHARGE terminals) AC or DC Source:	AC – (18 to 24) VRMS with 1.2 ARMS maximum DC – (16 to 40) VDC with 1.1 A DC maximum
(SOLAR - ⏚ terminals) Solar Panel or Other DC Source[1]:	15 to 40 VDC Maximum Charging Current: 3.6 ADC typical 2.8 ADC to 4.3 ADC depending on individual charger
Battery Charging[2]	
Two-step temperature compensated constant- voltage charging for valve-regulated lead-acid batteries:	CYCLE Charging: Vbatt(T) = 14.70 V – (24 mV)·(T – 25°C) FLOAT Charging: Vbatt(T) = 13.65 V – (18 mV)·(T – 25°C) ±1% Accuracy on charging voltage over –40 to +60°C range
Operational Temperature Range[3]:	–40 to +60°C
Measurements	
Average Battery Voltage:	± (1% of reading +15mV) over -40 to +60°C range
Average Battery/Load Current[4] Regulator Input Voltage:	±(2% of reading +2mA) over -40 to +60°C range
Solar[5]:	±(1% of reading -0.25V)/- (1% of reading +1V) over -40 to +60°C range
Continuous[6]:	±(1% of reading -0.5V)/- (1% of reading +2V) over -40 to +60°C range
Charger Temperature:	± 2°C
Power Out (+12 terminals)	
Voltage:	Unregulated 12 V from Battery

4 A Self-Resettable Thermal Fuse Hold Current Limits:	> 4 A @ < 20°C 4.0 A @ 20°C 3.1 A @ 50°C 2.7 A @ 60°C

Quiescent Current

No Charge Source Present:	300 μA Maximum
No Battery Connected:	2 mA Maximum

Physical Specifications

PS200:	4.2 in (10.6 cm) tall, 7.5 in (19.3 cm) long, 3 in (7.6 cm) wide
CH200:	3.9 in (10.0 cm) tall, 3 in (7.5 cm) long, 1.5 in (3.7 cm) wide

1. Battery voltages below 8.7 V may result in < 3.0 A current limit because of foldback current limit.

2. Cycle and Float charging voltage parameters are programmable with the default values listed.

3. VRLA battery manufacturers state that "heat kills batteries" and recommend operating batteries ≤ 50°C.

4. Impulse type changes in current may have an average current error of ±(10% of reading + 2 mA).

5. 1.0 V negative offset is worst-case due to reversal protection diode on input. Typical diode drop is 0.35 V.

6. 2.0 V negative offset is worst-case due to two series diodes in AC full-bridge. Typical diode drops are 0.35 each for 0.7 V total.

3.2 Battery Packs

Operating Temperature Range: -40 to +50ºC

Capacity

PS200:	7 Amp hours
BP12:	12 Amp hours
BP24:	24 Amp hours

WARNING **Battery life is shortened if the battery is allowed to discharge below 11.5 V.**

3.3 Charging Sources

Campbell Scientific Solar Panels			
	SP10	**SP20**	**SP70**
Peak Power	10 W	20 W	70 W
Voltage @ Peak Power	16.8 V	16.8 V	17.1 V
Current @ Peak Power	0.59 A	1.19 A	4.1 A

Notes:

1. Specifications assume a 1 kilowatt per square metre illumination and a solar panel temperature of 25°C (77°F).

2. Individual panels may vary up to 10%.

3. The output panel voltage increases as the panel temperature decreases, which is in the same direction as the recommended VRLA battery charging voltage change with temperature.

A number of different AC adaptors are available depending on the market and application.

3.4 Communication Interface Cables

The PS200/CH200 does not by default come with interface cables. Cables must be ordered separately. Communication cables are needed if changes to the default settings of the PS200/CH200 are required or if you wish to take advantage of the data coming from the PS200/CH200 using advanced programming techniques. See the cable listings below for the different cable types and how they are used.

RS-232 Interface cable – 6 foot length: Campbell Scientific Item No. #20770. This cable terminates in a 9-pin RS-232 serial connector for connecting to a computer and used to configure the PS200/CH200 using Device Configuration Utility (see Section 5.1 Configuring the PS200/CH200 Using Campbell Scientific Device Configuration Utility) or terminal emulation (see Section 5.1.3 Terminal Tab). This cable could be used to connect to the datalogger's RS-232 port but would also require a null modem cable such as Campbell Scientific Item No. #18663.

SDI-12 Interface Cable - 2 foot length: Campbell Scientific Item No. #20769. Terminates in wires. See Section 5.2 SDI-12 for programming and wiring instructions.

Tx/Rx Datalogger RS-232 Interface Cable - 2 foot length: Campbell Scientific Item No. #25356. Terminates in wires. See Section 5.3 RS-232 Interface for programming and wiring instructions.

3.5 Communication Cable Interface Connector Pin-Out

The communication connector has four pins. The following drawing shows the connector pin-out as viewed looking at the connector.

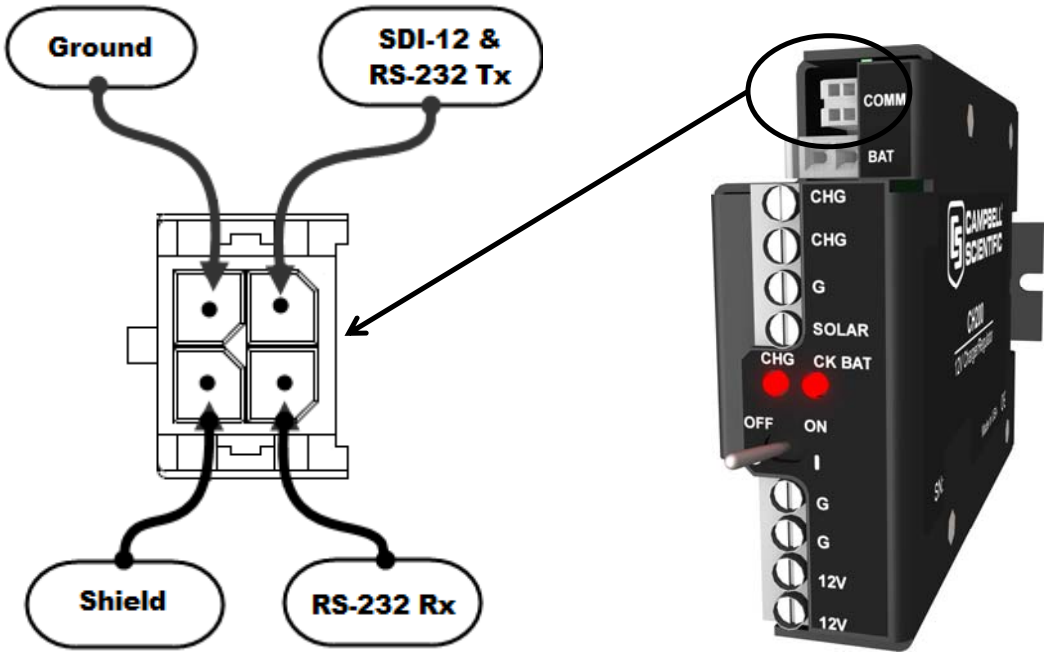


Figure 3-1. Communication Connector Pin-Out

4. Operational Overview

A simplified schematic of the CH/PS200 charging regulator is illustrated in Figure 4-1. A 12-V 7Amp-hr rechargeable battery is included with the PS200, whereas the user provides the rechargeable battery, such as the BP12 or BP24 offered by Campbell Scientific, Inc., for the CH200. See Section 3.2 Battery Packs for rechargeable batteries offered for the CH200 by Campbell Scientific.

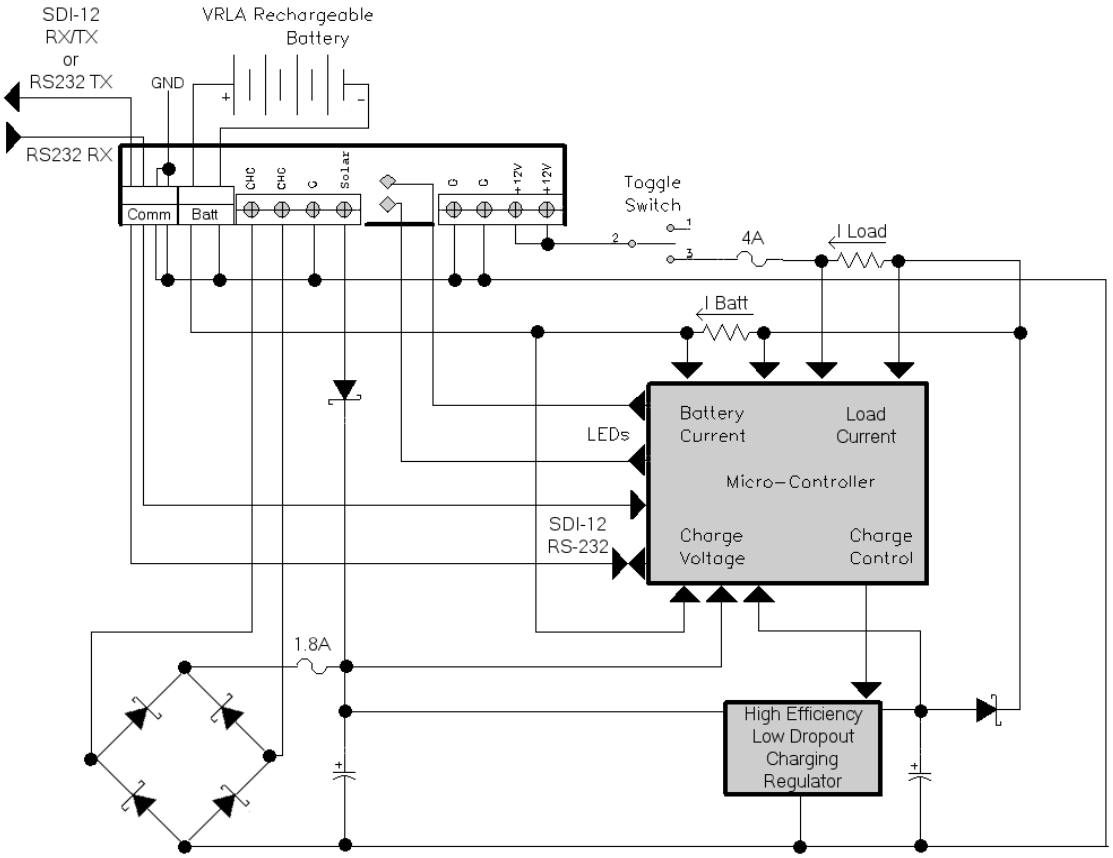


Figure 4-1. PS200/CH200 Schematic

Charging power for the PS200/CH200 is typically supplied by an unregulated solar panel, AC/AC transformer, or AC/DC converter. As illustrated in Figure 3-1, the CHARGE – CHARGE terminals are connected to a diode bridge, accommodating either AC or DC voltages from a charge source. Because of the diode bridge, polarity does not matter when connecting sources to the CHARGE – CHARGE input terminals. In order to protect AC/AC or AC/DC sources when charging discharged batteries, the CHARGE – CHARGE input terminals offer an approximately 1.1 amps DC (1.2 A RMS) current limit. The SOLAR – G input terminals are intended for connection to solar panels, or other high-current DC sources. Polarity definitely matters on the DC only SOLAR – G input terminals, with positive connected to SOLAR and the return or negative connected to G, with reversal protection included. The SOLAR – G input terminals have an input current limit of approximately 3.6 amps, making the PS200/CH200 well suited for 70 watt or smaller solar panels. The PS200/CH200 can be simultaneously powered from both the CHARGE – CHARGE and SOLAR – G input terminals, as the internal diodes will route power from the source with the highest input voltage. This allows for an AC mains powered application with a solar panel for back-up. If the reverse is needed – solar power as the primary supply and AC as the secondary supply – then a solar panel should be used with a 24V DC output and an AC, or AC to DC, source with a voltage output less than the solar panel voltage.

An AC/DC converter charge source could be connected to either the CHARGE – CHARGE input terminals or the SOLAR - G input terminals. The best input terminals to use with a given AC/DC converter should be based on the converters output current capability. For example, the CHARGE – CHARGE input terminals provide a user programmable current limit, with a maximum limit of 2.0 A DC for

the charging source. Whereas the SOLAR - G input terminals have a fixed 3.6 A DC typical current limit, providing faster battery charging for a charge source that can deliver up to 3.6 A DC current without damage.

The SOLAR – G terminals are optimal for solar panels because of the high-current charging capability when solar resources are available. A Maximum Power Point Tracking algorithm is also utilized when the PS200/CH200 detects the charging source is connected to the SOLAR input. Whereas, powering from the CHARGE – CHARGE terminals is preferable if a continuous charge source is available because of better overload protection of the charge source and less aggressive battery charging which is preferable for a continuous charge source.

The +12 V output terminals are intended to power a datalogger and any peripherals. Power to these output terminals is controlled by a toggle switch, with the total output current limited by a 4 amp self-resettable thermal fuse (see Section 3.1 Specifications for hold current limits at various temperatures). The A105 Additional 12 V Terminals Adapter may be used to provide extra 12 V and ground terminals where the power supply is used to power several devices, noting that the hold current limit on the 4 amp self-resettable thermal fuse still applies.

The charging algorithm takes into account battery temperature to calculate the correct charging voltage required to maintain the battery. The source of the temperature used by default is the temperature of the charger itself with the assumption that the charger resides in the same enclosure with the battery and the two temperatures will be similar. If the battery used by this system is located in separate enclosure it is advised to put a temperature sensor on the battery, or close to it, that is read by a datalogger connected to the PS200/CH200 and send this temperature information to the module using the appropriate SDI-12 (see Section 5.2.3.1 Write Remote Battery Temperature to PS200/CH200 and Section 5.2.3.2 Restore Internal Battery Temperature Measurement) or RS-232 (see Section 5.3.3.2 Write Remote Battery Temperature to PS200/CH200 and Section 5.3.3.3 Restore Internal Battery Temperature Measurement).

The PS200/CH200 have two flashing LED indicators, the CHARGE LED and the CHECK BATTERY LED. See Tables 3-1 and 3-2 for details on the various conditions and associated colours for these two indicator LEDs.

Communication with the CH/PS200 is accomplished through the COMM port to either a datalogger or computer. Datalogger communication can be done via a SDI-12 interface cable (item# 20769) as indicated in Figure 3-9, or an RS-232 TX/RX interface cable (item #25356) as indicated in Figure 3-10. Communication to a computer requires a RS-232 interface cable (item# 20770) that terminates in a standard 9-pin connector. See Section 5.2 SDI-12 and Section 5.3 RS-232 Interface for details of using these cables and programming examples where required.

5. User Interface

Unlike any other power supply that Campbell Scientific has ever built the PS200/CH200 has the unique ability to send a wealth of information back to the user to observe and manage power requirements and possible problems. It can also be configured to work with a wide range of batteries and input power supplies and test the existing battery system for possible shorting and sulphation problems.

NOTE If viewing this manual as a PDF file all the datalogger programming examples can be highlighted, copied, and pasted directly into the CRBasic editor.

The PS200/CH200 can be used directly from the factory without any configuration or communication interface. Programming examples are provided in this manual but can also be downloaded directly from Campbell Scientific's web site (http://www.campbellsci.com/downloads). For those who wish to take advantage of some of the additional features of the PS200/CH200, however, the PS200/CH200 supports four kinds of serial communication:

1. Campbell Scientific software, Device Configuration Utility (DevConfig), via RS-232 by which PS200/CH200 settings can be viewed and changed. Or, an upgraded operating system can be sent to a PS200/CH200. Also, battery voltage, battery current, load current, input voltage, input current, temperature, charge state, charge source (solar or continuous), and a check battery flag can be viewed. DevConfig may be downloaded free of charge from Campbell Scientific's web site (http://www.campbellsci.com/downloads). See Section 5.1 Configuring the PS200/CH200 Using Campbell Scientific Device Configuration Utility.

2. A simple text based interface via RS-232 that can be used with Microsoft Hyper Terminal or Campbell Scientific DevConfig terminal screen. Battery voltage, battery current, load current, input voltage, input current, temperature, charge state, charge source (solar or continuous), and a check battery flag can be viewed. Also, the SDI-12 address of a PS200/CH200 can be changed via the text based interface. See Section 5.1.3 Terminal Tab.

3. SDI-12 by which the PS200/CH200 performs as an "SDI-12 sensor" according to the SDI-12 standard (http://www.sdi-12.org). Battery voltage, battery current, load current, input voltage, input current, temperature, charge state, charge source (solar or continuous), and a check battery flag can be read from a PS200/CH200 via SDI-12 using the Campbell Scientific CRBasic instruction "SDI12Recorder" (Campbell Scientific CR1000 datalogger program examples in Section 5.2.2 SDI-12 Measurements). See Section 5.2 SDI-12 for an in-depth explanation.

4. RS-232 communications specifically designed to work with a Campbell Scientific datalogger and a Campbell Scientific datalogger program using CRBasic commands “SerialOut” and “SerialIn” (Campbell Scientific CR1000 datalogger program example in Section 5.3.4 Datalogger Programming for RS-232 Communication to the PS200/CH200). Battery voltage, battery current, load current, input voltage, input current, temperature, charge state, charge source (solar or continuous), and some other charge control parameters normally not of interest to the user can be viewed. See Section 5.3 RS-232 Interface for an in-depth explanation.

The advantage of SDI-12 over RS-232 is that many sensors can share a single datalogger SDI-12 port, whereas, RS-232 is limited to one device per port. The advantage of RS-232 over SDI-12 is that its speed can be faster than SDI-12. Datalogger programming for SDI-12 is usually simpler than programming for a RS-232 device.

All PS200/CH200 serial communications are via its four-pin COMM connector. The PS200/CH200 will detect which of the four kinds of serial communications is being used and will reconfigure itself accordingly.

Some configurations of the PS200/CH200 can also be done via a datalogger using RS-232 or SDI-12 commands but require knowledge of CRBasic programming. See Section 5.2 SDI-12 for SDI-12 programming examples and Section 5.3 RS-232 Interface for datalogger RS-232 programming.

5.1 Configuring the PS200/CH200 Using Campbell Scientific Device Configuration Utility (DevConfig)

The easiest way to configure the PS200/CH200 is using Device Configuration Utility and a computer. The only way to update the operating system in a PS200/CH200 is via Device Configuration Utility.

To use Device Configuration Utility the PS200/CH200 must have power and a Campbell Scientific RS-232 interface cable (item #20770) (see Figure 5-1). Some key features of DevConfig include:

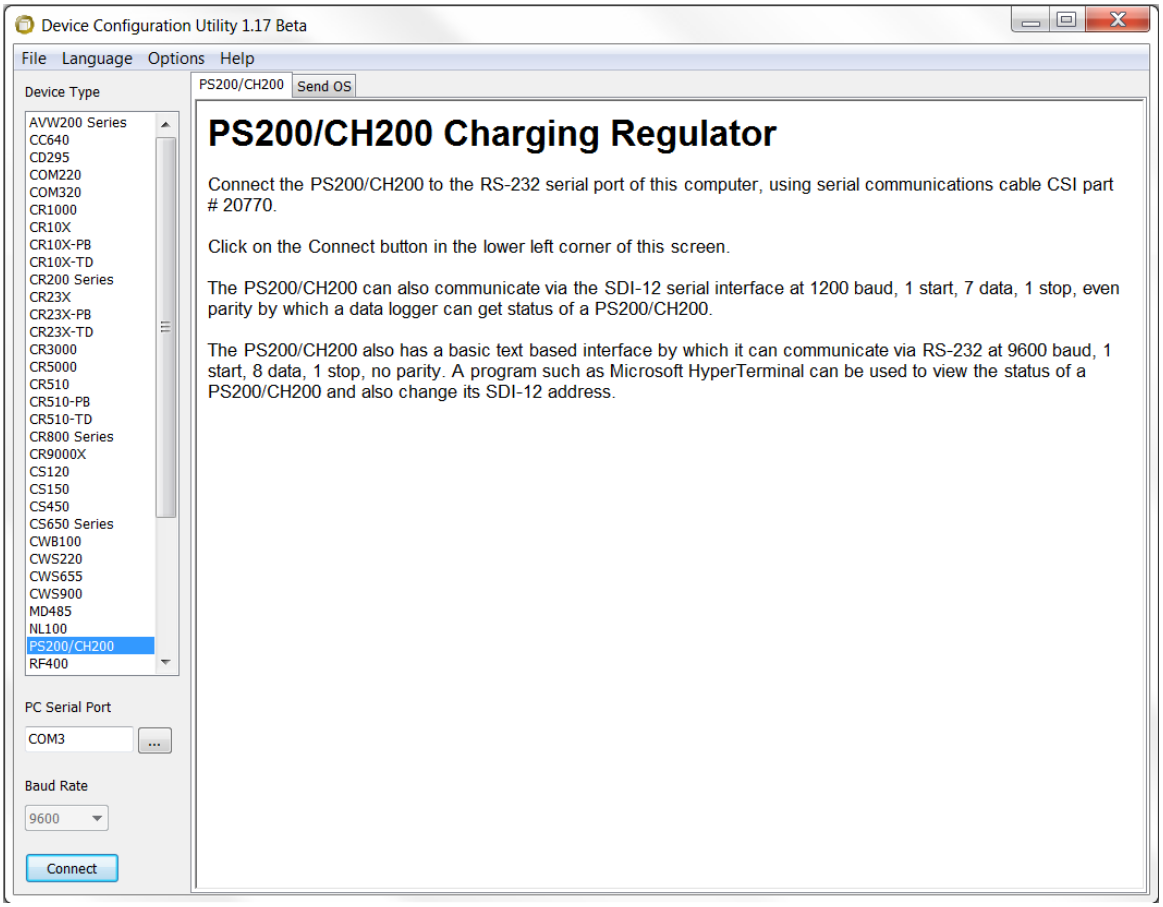
- Supports direct serial connections between the PC and devices.
- Sends operating systems to supported device types.
- Provides a reporting facility where a summary of the current configuration of a device can be shown on the screen and printed. This configuration can also be saved to a file and used to restore the settings in the same or a replacement device.
- Help for DevConfig is shown as prompts and explanations on its main screen.
- Updates to DevConfig are available from Campbell Scientific's web site, http://www.campbellsci.com/downloads. These may be installed over older versions.

NOTE Before opening DevConfig, close other software on the computer that uses the computer's serial (COM) ports (e.g., LoggerNet, PC400, or PC200W).

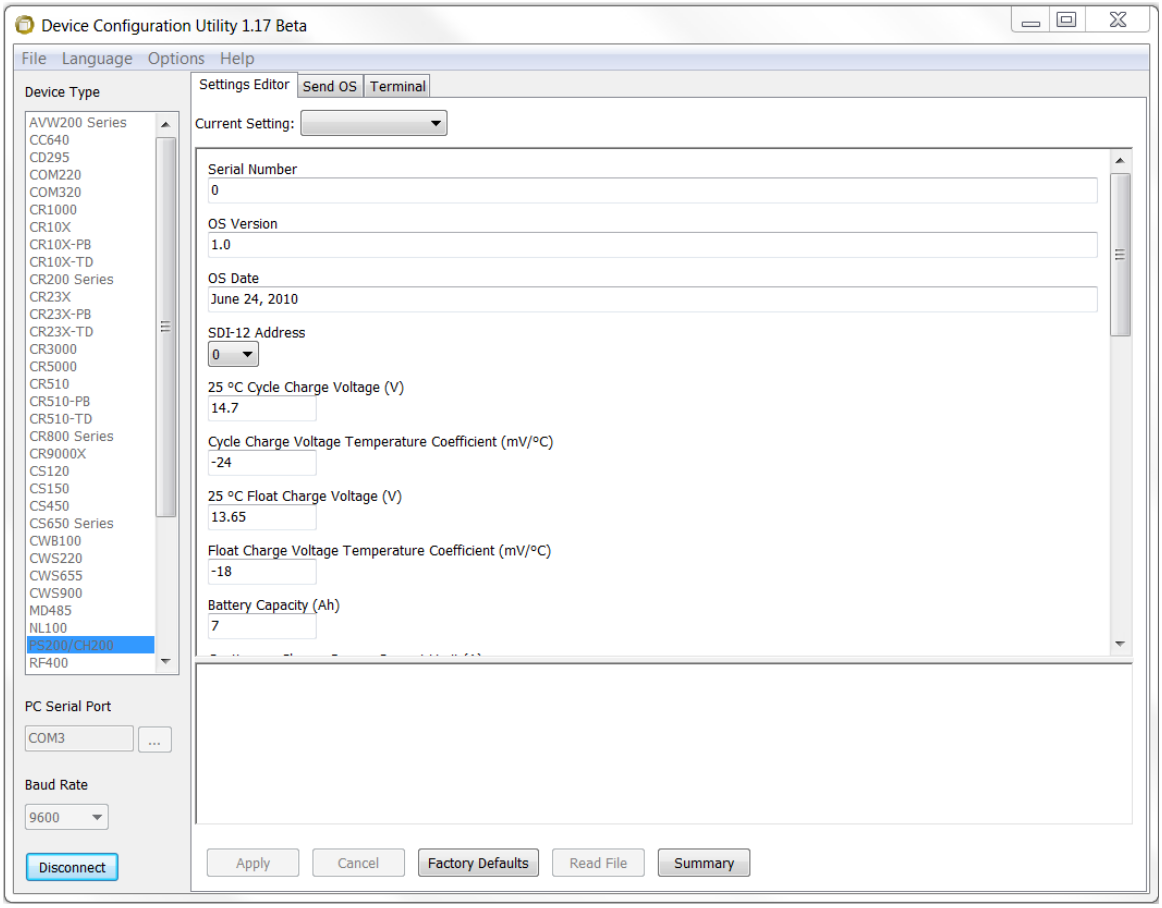
Figure 5-1. PS200 20770 Serial Cable

5.1.1 Main DevConfig Screen

The DevConfig window is divided into two main sections: the device selection panel on the left side and device-specific tabs on the right side. Click on the PS200/CH200 device name from the list on the left, for which a list of installed RS-232 serial ports (COM1, COM2, etc.) will appear. The PS200/CH200 has a fixed baud rate of 9600. The page for each device presents instructions about how to set up the device to communicate with DevConfig.



When the user clicks on the Connect button, the device type, serial port, and baud rate selector controls become disabled and, if DevConfig is able to connect to the PS200/CH200, the button will change from "Connect" to "Disconnect". The Display will change to:



5.1.2 Settings Editor Tab

Serial Number displays the PS200/CH200serial number. This setting is set at the factory and cannot be edited.

OS Version displays the PS200/CH200 operating system version. Visit the web site, http://www.campbellsci.com/downloads , to determine whether a newer operating system is available.

OS Date displays the last revision date of the operating system presently in the PS200/CH200.

SDI-12 Address allows the user to set the SDI-12 address of the PS200/CH200. Factory default for this address is 0. Each SDI-12 sensor connected to a port of an SDI-12 recorder (datalogger) should have a unique SDI-12 address. See Section 5.2 SDI-12.

25°C Cycle Charge Voltage (V) is the ideal first-stage charging voltage for the battery connected to the PS200/CH200 at 25 °C. The allowable range is within 12 V and 15 V. **This is factory set and should not be altered unless the user has specific information from a battery manufacturer that recommends a different setting.**

Cycle Charge Voltage Temperature Coefficient (mV/°C) is the millivolts of change in cycle charge voltage per degree Celsius of increase in ambient temperature. The allowable range is within -50 and 50 mV/°C. **This is factory set and should not be altered unless the user has specific information from a battery manufacturer that recommends a different setting.**

25 °C Float Charge Voltage (V) is the ideal second-stage charging voltage for the battery connected to the PS200/CH200 at 25°C. The allowable range is within 12 Volts and 15 Volts. **This is factory set and should not be altered unless the user has specific information from a battery manufacturer that recommends a different setting.**

Float Charge Voltage Temperature Coefficient (mV/°C) is the millivolts of change in float charge voltage per degree Celsius of increase in ambient temperature. The allowable range is from -50 to 50 mV/°C. **This is factory set and should not be altered unless the user has specific information from a battery manufacturer that recommends a different setting.**

Battery Capacity (Ah) is the user entered battery capacity. Cycle charging is disabled if set to 0 Ah. PS200 default = 7 Ah. CH200 default = 0 Ah. Set to 24 Ah for BP24. Set to 12 Ah for BP12.

Continuous Charge Source Current Limit (A) is a current limit setting for the continuous charging source terminals, marked "CHG" on the PS200/CH200, to which a user might connect a small AC transformer or DC power supply. Set the current limit to less than the maximum continuous output current rating of the transformer or power supply. This will prevent the transformer or power supply from being overloaded even if the battery is deeply discharged or has a shorted cell. Default = 1.1 A.

Real-Time Measurements (not user adjustable except where indicated) include:

Battery Voltage (V) indicates the present battery charge voltage.

Battery Current (A) indicates current being supplied to, or drawn from, the battery. A positive reading means the battery is charging. A negative reading means the battery is discharging.

Load Current (A) indicates current being supplied to the load connected to the PS/CH200.

Charge Input Voltage (V) indicates the DC voltage going into the regulation circuitry. If an AC source is being used to supply power to the PS200/CH200, then this voltage is the rectified DC voltage.

Charge Input Current (A) indicates current at the PS200/CH200 current input, which can be coming from the SOLAR terminals or from the CHG terminals.

Temperature (°C) indicates the PS200/CH200 on board temperature measurement. Not user adjustable via DevConfig, but the serial interface can be used to override the PS200/CH200 on board temperature measurement. See Section 5.2.3.1 Write Remote Battery Temperature to PS200/CH200 and Section 5.2.3.2 Restore Internal Battery Temperature Measurement for SDI-12 programming examples and Section 5.3.3.2 Write Remote Battery Temperature to PS200/CH200 and Section 5.3.3.3 Restore Internal Battery Temperature Measurement for RS-232 programming examples. The PS200/CH200 uses this temperature in calculating optimal charging voltage.

PS200/CH200 Charge Status indicates the PS200/CH200 present state of charging. One of these text strings will be displayed:

"Not Charging." (No charging source detected.)
"Current Limited Charging."
"Cycle Charging."
"Float Charging."
"Battery Test Mode." (See Section 5.2.3.3 Change Battery Capacity Value in PS200/CH200 for SDI-12 battery test mode and Section 5.3.3.5 Enter Battery Test Stat for RS-232 battery test mode.)
"Internal Fault Detected. SERVICE REQUIRED."

PS200/CH200 Charge Source indicates the PS200/CH200 active charging source. One of these text strings will be displayed:
"No Detectable Charge Source."
"Solar Input (Solar and G terminals)."
"Continuous Source Input (CHG CHG terminals)."

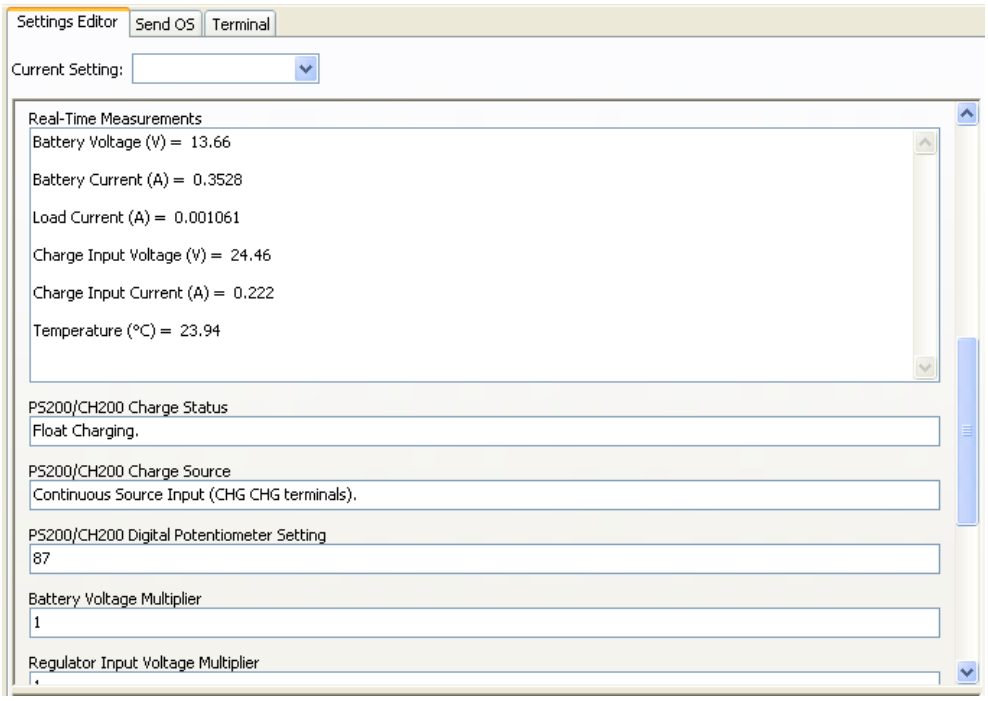
PS/CH200 Digital Potentiometer Setting indicates the present value being sent to the PS200/CH200 battery voltage control digital potentiometer. This value is not in volts but sets the battery voltage control according to the formula: battery voltage target = K / (DPot + L).

The following values are set at the factory and cannot be edited. Values should never change. If they do change, send the PS200/CH200 in for repair.

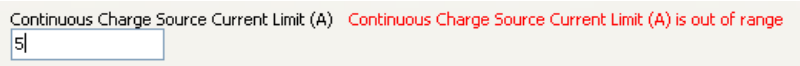
Battery Voltage Multiplier	0.9 to 1.1
Regulator Input Voltage Multiplier	0.9 to 1.1
Battery Current Offset	± 0.010
Load Current Offset	± 0.010
Battery Current Multiplier	0.9 to 1.1
Load Current Multiplier	0.9 to 1.1
Digital Potentiometer K Value	5013 to 6127
Digital Potentiometer L Value	287 to 351

Press F5 to refresh the Settings Editor.

The top of the Settings Editor is a grid that allows the user to view and edit the settings for the device. The grid is divided into two columns with the setting name appearing in the left hand column and the setting value appearing in the right hand column. To change a setting, select the field to be changed, type a new value and press Enter. If the value entered is not within an acceptable range, a message in red lettering will appear to the right of the entered value. Real time values can be viewed while in this screen. These values are updated every second.



An example of an incorrect value entered in for the continuous charging source current is shown below.

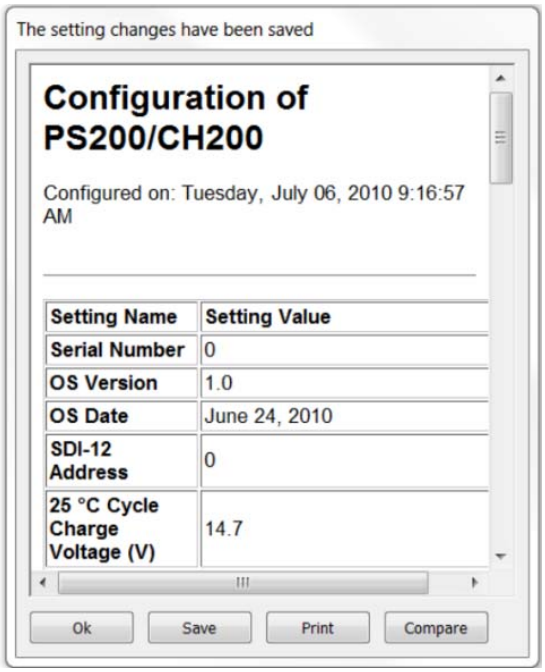


The bottom of the Settings Editor displays help for the selected setting. The **Apply** and **Cancel** button will become active when a setting is changed. An asterisk (*) will also appear next to values that have been changed.

Clicking the **Factory Defaults** button on the Settings Editor will send a command to the PS200/CH200 to revert to its factory default settings (PS200/CH200 factory made calibration values and other "read only" values are not affected).

The **Read File** can be used for restoring settings from a file that was created earlier. The settings from the saved file are immediately sent to the PS200/CH200.

Edit settings as needed, click on the **Apply** button; or, if no changes were made, click on the **Summary** button; and, if the PS200/CH200 accepts the settings, a configuration summary dialogue will appear. The user will have a chance to save the settings so that **Read File** can be used later and/or print the settings for the device:

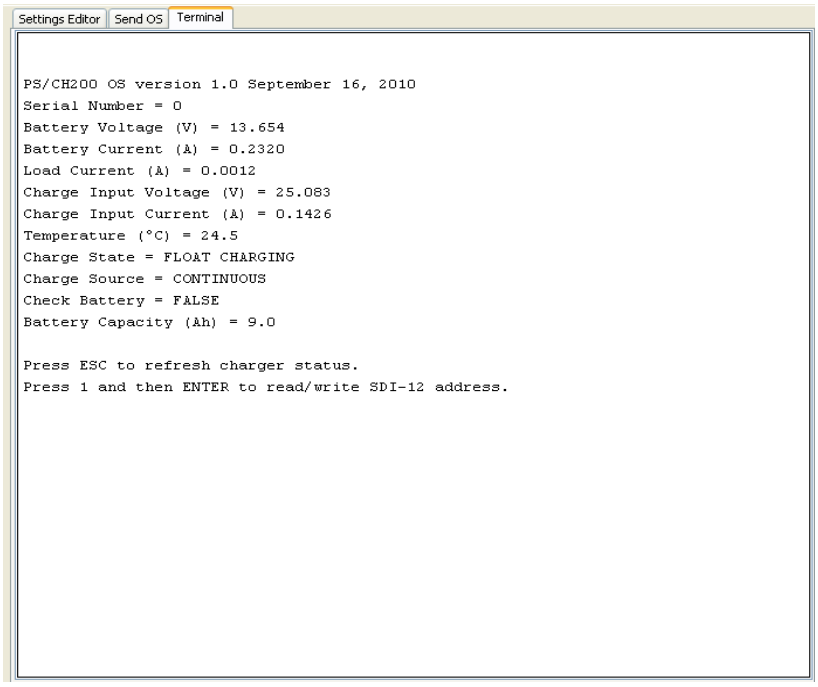


Click on the **Ok** button to return to the settings editor if no changes were made or to the main screen if changes were made.

5.1.3 Terminal Tab

Click on the "Terminal" tab to invoke the PS200/CH200 terminal interface.

Only the SDI12 address can be changed from this window. (Or, Windows Hyper Terminal could be used. See Section 5.1.3 Terminal Tab.) A screen similar to the following should appear:



Press ESC (may have to press ESC twice) to invoke the PS200/CH200 RS-232 text based interface. For more on the text based interface, see Section 5.1.3 Terminal Tab.

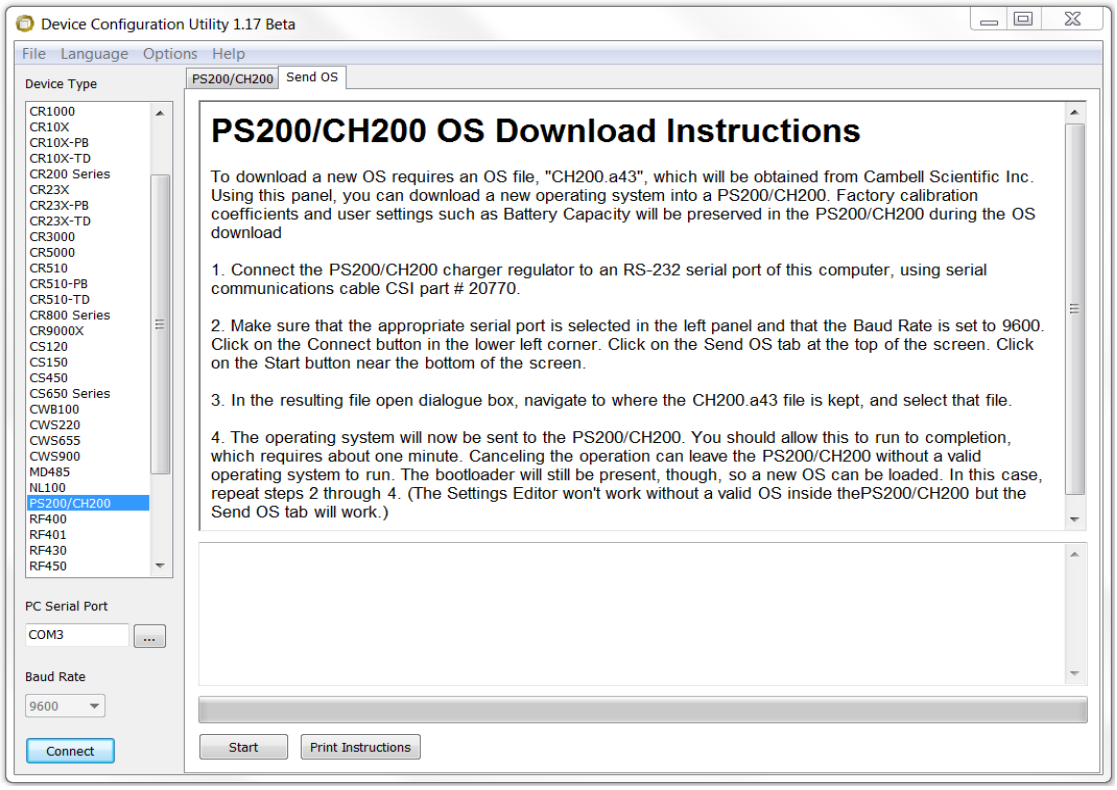
5.1.4 Send OS Tab - Downloading an Operating System

Having to send a new operating system is rare but can be done easily using Device Configuration Utility and the RS-232 interface cable Item No. #20770.

Get The Latest Operating PS200/CH200 Operating System
The latest operating system for the PS200/CH200 is available from the Campbell Scientific website at www.campbellsci.com. From the home page select "Support". From the pop-up menu select "Downloads". Search for "PS200". This file will be an executable file that will automatically install itself into the correct folders. Run the file on the computer that will be used to connect to the PS200/CH200.

Load The Operating System Into the PS200/CH200
Make sure the PS200/CH200 has power and connect it to the computer used to send the new operating system. Open Device Configuration Utility after the operating system has been downloaded on the computer. Select a PS200/CH200 from the list on the left hand side of the Device Configuration Utility. Select the tab at the top marked "Send OS". Click on the "Start" button at the bottom of the screen. A dialog box marked "Select the operating system to send" should appear. If the operating system was stored on the computer as shown above then navigate to the "C:\Campbellsci\LIB\OperatingSystems" folder. Select the file "CH200.a43" and click on the "Open" button. If you stored the file somewhere else on this computer then go find it and open it. The LED indicator on the

PS200/CH200 marked “CHG” should change from red to green indicating the module is receiving a new operating system. This will take a few minutes. DO NOT disconnect from the PS200/CH200 during this process. When the operating system has been successfully sent the green LED will go out. Device Configuration Utility will also bring up a window showing the new operating system was successfully sent.



5.2 SDI-12

SDI-12 is a serial communications standard specifically designed for reading measurements from multiple measurement sensors to a "data recorder", which in this context is synonymous with datalogger. For a complete technical definition of the SDI-12 protocol, go to the web site, http://www.sdi-12.org.

The PS200/CH200 is an SDI-12 sensor compliant with version 1.3 of the standard, and Campbell Scientific loggers are capable of acting as SDI-12 recorders. A Campbell Scientific datalogger or other SDI-12 recorder can be connected to a PS200/CH200 via the cable Campbell Scientific part # 20769. SDI-12 uses only one wire and a ground for serial communications. The Campbell Scientific CR1000 datalogger, for example, has four SDI-12 ports on control ports 1, 3, 5, and 7.

Campbell Scientific Item #20769 SDI-12 Cable Wiring (see Figure 5-2)

Green: Control port used for SDI-12 communication. (CR1000 odd numbered control ports)

Black: G

Clear: G

NOTE

All SDI-12 sensors take time to reply back to a request for data. It is recommended to put any SDI-12 instructions in a "slow sequence" to minimize interrupting the main program.

Green: C1
Black: G
Shield: G

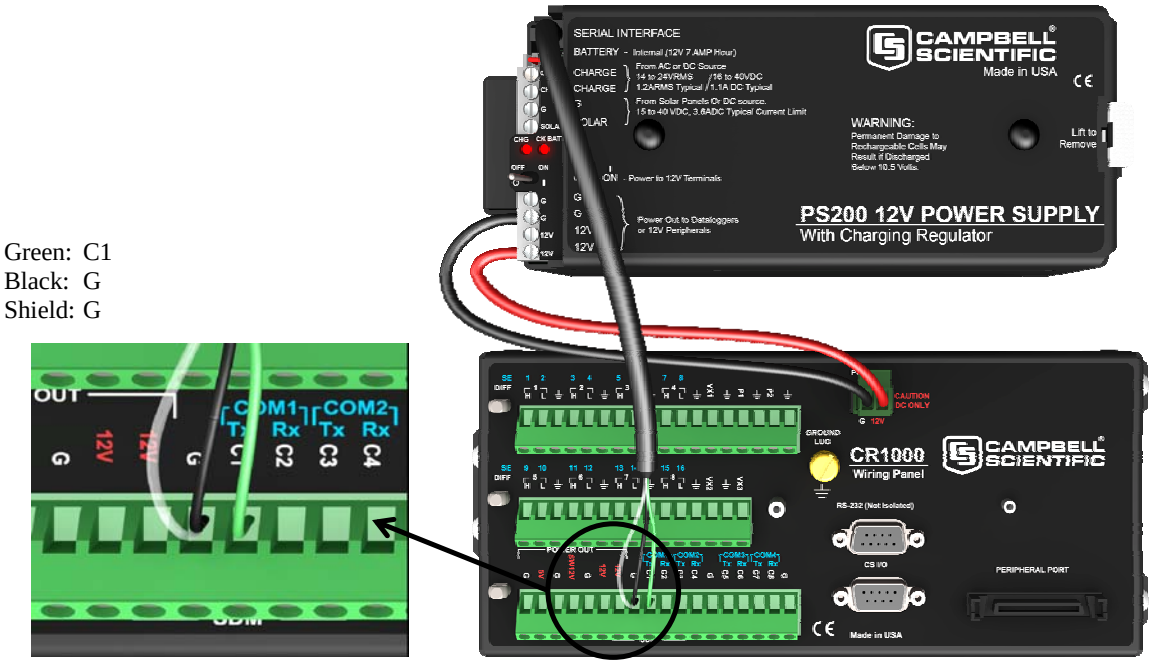


Figure 5-2. PS200 Connected to a CR1000 via SDI-12

5.2.1 SDI-12 Address

The SDI-12 standard is designed so that multiple SDI-12 sensors can be connected in parallel to a common SDI-12 port of an SDI-12 recorder (datalogger). Each of the SDI-12 sensors is assigned a unique SDI-12 address. Valid addresses are 0 through 9, A through Z, and a through z. From the factory a PS200/CH200 has an SDI-12 address of 0. There are three different means by which a PS200/CH200 SDI-12 address can be changed:

1. Campbell Scientific DevConFig. (See Section 5.1.2 Settings Editor Tab.)

2. The RS-232 text based interface. (See Section 5.1.3 Terminal Tab.)

3. Via SDI-12 according to the SDI-12 standard.

Normally Device Configuration Utility (see Section 5.1.2 Settings Editor Tab) or the RS-232 text based interface (see Section 5.1.3 Terminal Tab) is used to change the SDI-12 address of a PS200/CH200. The SDI-12 address is usually set once and then seldom if ever changed again. The SDI-12 address stored in the PS200/CH200 is saved in "non-volatile" memory and will not be lost by power cycling the device. If required a CR1000 or newer table based datalogger can be used to send a new SDI-12 address to the PS200/CH200. For a programming example see Appendix A.6 Changing The SDI-12 Address Programming Example.

If a datalogger program is created to set the SDI-12 address of a PS200/CH200 or other SDI-12 sensor via SDI-12, the program can only be used for one SDI-12 sensor at a time. All other sensors must be disconnected from that SDI-12 port of the datalogger.

5.2.2 SDI-12 Measurements

The PS200/CH200 reports 9 measurements via the SDI-12 M! command, as described below. Because SDI-12 is a low speed standard, this M! command should only be used in a slow sequence with a scan rate of 2 seconds or longer. Individual measurements using M1! through M6! (described below) can be used in scan rates of 1 second. Note: If you are using a CR2XX datalogger, the OutString parameter is entered as aMx! without quotes, where a is the associated SDI-12 Recorder address and x is 1…6.

NOTE Older CR10(X) and CR23X dataloggers will only work with aMx! commands.

Recorder: MC! Return All Main Values.

PS200/CH200 measurements:
1 - Battery Voltage (V).
2 - Battery Current (A).
3 - Load Current (A).
4 - Charge Input Voltage (V).
5 - Charge Input Current (A).
6 - Temperature (°C).
7 - Charge state (-1 = regulator fault, 0 = no charge, 1 = current limited charging, 2 = cycle charging, 3 = float charging, 4 = battery test).
8 - Charge Source (0 = none, 1 = solar, 2 = continuous).
9 – Check Battery (0 = normal, 1 = Check Battery - indicates battery voltage not increasing after significant charging).

The SDI-12 commands “Mx!” with x = 1...6 can be used to select different subsets of the above measurements, as described below:

Recorder: M1! Return Battery Voltage, Battery and Load Current.

PS200/CH200 measurements:
1 - Battery Voltage (V).
2 – Battery Current (A).
3 – Load Current (A).

Recorder: M2! Return Battery Voltage, Battery and Load Current and Temperature.

PS200/CH200 measurements:
1 - Battery Voltage (V).
2 – Battery Current (A).
3 – Load Current (A).
4 – Temperature (°C)

Recorder: M3! Regulator Input and Output Parameters and Temperature.

PS200/CH200 measurements:
1 - Battery Voltage (V).
2 – Battery Current (A).
3 – Load Current (A).
4 - Charge Input Voltage (V).
5 - Charge Input Current (A).
6 - Temperature (°C).

Recorder: M4! Return Charge Input Voltage and Current and Temperature.

PS200/CH200 measurements:
1 – Charge Input Voltage (V).
2 - Charge Input Current (A).
3 - Temperature (°C).

Recorder: M5! Return Charge State, Charge Source and Check Battery.

PS200/CH200 measurements:
1 - Charge State (-1 = regulator fault, 0 = no charge, 1 = current limited charging, 2 = cycle charging, 3 = float charging, 4 = battery test).
2 - Charge Source (0 = none, 1 = solar, 2 = continuous).
3 – Check Battery (0 = normal, 1 = Check Battery – indicates battery voltage not increasing after significant charging).

Recorder: M6! Return Target Battery Voltage, Battery Capacity, Qloss and Digital Pot Setting.

1 – Target Battery Voltage (V)
2 – Battery Capacity (Ah)
3 – Qloss (Ah)
4 – Digital Pot Setting

See Appendix A.4.1 for an example program.

Recorder: M7! Available for future use.

Recorder: M8! Available for future use.

Recorder: M9! Available for future use.

Below is a CR1000 programming example that acquires all 9 measurements once per minute from a PS200/CH200. Note that the PS200/CH200 does internal averaging of the voltage, current, and temperature measurements, with an averaging period that is set by the scan rate, so that in this example, the measurements are 30 second averages:

```
'CR1000 Series Datalogger
'Program: SDI12_Status_Info.CR1
'Date: 1.September.2010
'Ver: A
'
'Notes: Program returns all information from PS200/CH200.
'PS200/CH200 configured with SDI-12 address 0 (zero).
'Connect the PS200/CH200 to SDI-12 port C3 of the CR1000.
'Use Campbell Scientific SDI-12 cable part # 20769.
'
'SDI-12 CABLE TO CR1000 WIRING
'------------------------------
'WHITE: C3
'BLACK: G
'CLEAR: G
'
'Public variables.
  'Datalogger panel temperature: Celsius
Public PanelTempC
  'Array to hold all the data coming from the PS200/CH200
Public CH200_M0(9)
  'Alias names for array elements.
  'Battery voltage: VDC
Alias CH200_M0(1)=VBatt
  'Current going into, or out of, the battery: Amps
Alias CH200_M0(2)=IBatt
  'Current going to the load: Amps
Alias CH200_M0(3)=ILoad
  'Voltage coming into the charger: VDC

Alias CH200_M0(4)=V_in_chg
  'Current coming into the charger: Amps
Alias CH200_M0(5)=I_in_chg
  'Charger temperature: Celsius
Alias CH200_M0(6)=Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Alias CH200_M0(7)=Chg_State
  'Charging source: None, AC, or Solar
```

```
Alias CH200_M0(8)=Chg_Source
  'Check battery error: 0=normal, 1=check battery
Alias CH200_M0(9)=Ck_Batt

  'Arrays to hold the associated words for the charge state, charge source,
  'and check battery values.
Dim ChargeStateArr(6) As String
Dim ChargeSourceArr(3) As String
Dim CheckBatteryArr(2) As String

  'Variables to hold the words for charge state, charge source, and check
  'battery.
Public ChargeState As String
Public ChargeSource As String
Public CheckBattery As String

  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (6,CH200_M0(),FP2,0,False)
  Maximum (6,CH200_M0(),FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Main Program
BeginProg
    'Load arrays with words to associate with the charge state, charge source
    'and check battery values from the PS/CH200.
  ChargeStateArr(1) = "Regulator Fault"
  ChargeStateArr(2) = "No Charge"
  ChargeStateArr(3) = "Current Limited"
  ChargeStateArr(4) = "Cycle Charging"
  ChargeStateArr(5) = "Float Charging"
  ChargeStateArr(6) = "Battery Test"

  ChargeSourceArr(1) = "None"
  ChargeSourceArr(2) = "Solar"
  ChargeSourceArr(3) = "Continuous"

  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  Scan (5,Sec,0,0)
    PanelTemp(PanelTempC,_60Hz)       'Measure datalogger panel temperature.
    CallTable Hour
  NextScan

    'SlowSequence with SDI-12 measurements
  SlowSequence
  Scan(30,sec,0,0)
    SDI12Recorder (CH200_M0(),3,0,"MC!",1.0,0)
      'Array values start with one. Values for charge state start with -1.
      'Have to shift the value by two to line it up with the correct words
      'in the array.
    ChargeState = ChargeStateArr(Chg_State + 2)
      'Values for charge source start with zero. Have to shift the value
      'by one to line it up with the correct words in the array.
    ChargeSource = ChargeSourceArr(Chg_Source + 1)
      'Values for check battery start with zero. Have to shift the value
```

```
    'by one to line it up with the correct words in the array.
   CheckBattery = CheckBatteryArr(Ck_Batt + 1)
  NextScan
EndProg
```

5.2.3 SDI-12 Extended Commands

The response to all of the commands in this section is **aOK**, where '**a**' is the SDI-12 address of the PS200/CH200.

See Appendix A for example programs using these extended commands.

5.2.3.1 Write Remote Battery Temperature to PS200/CH200

aXTmm.dd! command overrides the internal temperature measurement, where `a' is the PS200/CH200's SDI-12 address and mm.dd refers to an ASCII string of numerals with a decimal point of temperature in degrees Centigrade within -40 and 100°C. This command overrides the PS200/CH200 internal temperature measurement for 15 minutes or until cancelled by a temperature restore command. The temperature value is used in calculating the optimal battery charge voltage.

See Appendix A.1.1.

5.2.3.2 Restore Internal Battery Temperature Measurement

aXTR! Restore to internal PS200/CH200 battery temperature, where `a' is the SDI-12 address of the PS200/CH200.

Undo what was done with the **XTmm.dd! c**ommand.

See Appendix A.1.1.

5.2.3.3 Change Battery Capacity Value in PS200/CH200

aXCmm.d! Enter battery capacity, where `a' is the PS200/CH200's SDI-12 address and mm.d is ASCII representation of battery capacity in Ah.

To use the quicker dual rate charging scheme in the PS200/CH200 a value other than zero must be entered into the battery capacity field. BUT! This value needs to be the actual amp-hour capacity of the battery being used or the charger will not charge correctly. This instruction allows the user to change the battery capacity in the field if they find themselves swapping out batteries with a different amp-hour capacity model.

See Appendix A.2.1 for a programming example using this instruction.

5.2.3.4 Enter Battery Test State

aXB! Enter battery test state with 90 second time out, where `a' is the SDI-12 address of the PS200/CH200.

Use this instruction to cause the system to run strictly on battery voltage and test the battery for sulphation or other issues.

This command can be used in a datalogger program to test a battery connected to a PS200/CH200. This command instructs a PS200/CH200 to set its charging voltage to 11.5 VDC for 90 seconds. Generally, the battery voltage will be greater than 11.5 VDC, so this command momentarily causes a PS200/CH200 to stop charging the battery. Then, the datalogger program can analyze the battery charging capacity (see example program). Because the BATT_TEST command stays in effect for only 90 seconds, the datalogger program must issue the

BATT_TEST command more frequently than once per 90 seconds for as long as the test is to last.

See Appendix A.3.1 for a programming example using this instruction.

5.2.3.5 Zero Out Qloss (Battery Charge Deficit)

aXRQ! Set Qloss to zero, where `a' is the SDI-12 address of the PS200/CH200.

Qloss will not automatically set itself back to zero unless it has been trying to cycle charge a battery for eight hours or more. Qloss is a handy way to monitor the charge/discharge characteristics of a system but occasionally it might need to be set back to zero.

See Appendix A.4.1 for a programming example using this instruction.

5.2.3.6 Power Usage

Although this is not an instruction in and of itself the ability to calculate power usage by the battery and the system is a very powerful capability of the PS200/CH200. Measuring and storing power usage is useful to determine system power requirements (power budget) and help to identify devices using too much power.

See Appendix A.5.1 for a programming example.

5.3 RS-232 Interface

5.3.1 Text Based Interface

The PS200/CH200 has a simple text based interface via RS-232 that can be used with HyperTerminal or Campbell Scientific DevConfig terminal screen. Battery Voltage (V), Battery Current (A), Load Current (A), Charge Input Voltage (V), Charge Input Current (A), Temperature (°C), Charge State, Charge Source (solar or continuous), Check Battery flag (True or False) and Capacity (Ah) can be viewed. Also, the SDI-12 address of a PS200/CH200 can be changed via the text based interface.

The cable Campbell Scientific part # 20770 connects a PS200/CH200 to an RS-232 COM port of a computer for use with the text based interface or Campbell Scientific DevConfig.

The RS-232 settings are:

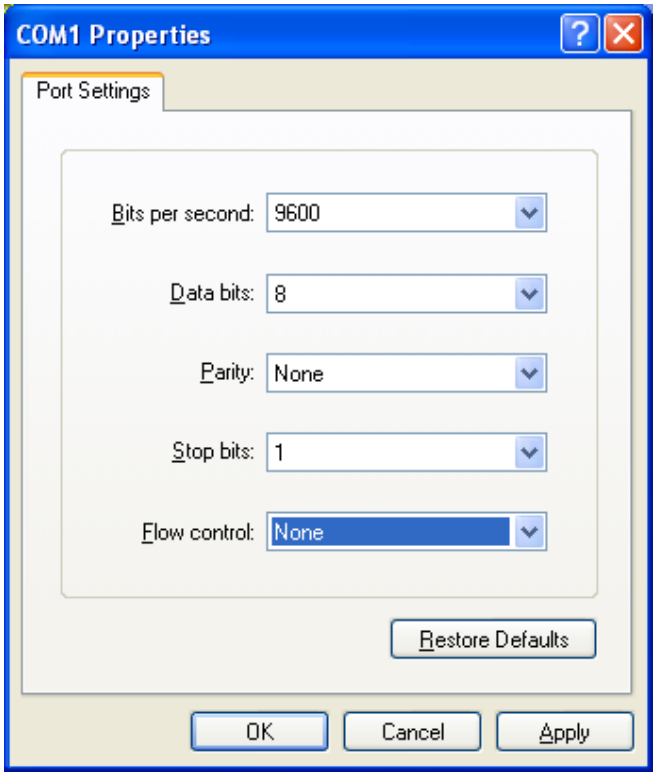
Bits per second:	9600
Data bits:	8
Parity:	None
Stop bits:	1
Flow Control:	None

To use Microsoft Hyper Terminal on a computer running Microsoft Windows XP, for example, click on Start, All Programs, Accessories, Communications, Hyper Terminal. A dialog box will open:

Pick a name for the connection, such as "CH200".

Click on Ok, and then select the COM port to use with the PS200/CH200:

Click on OK, and then set the COM port settings:

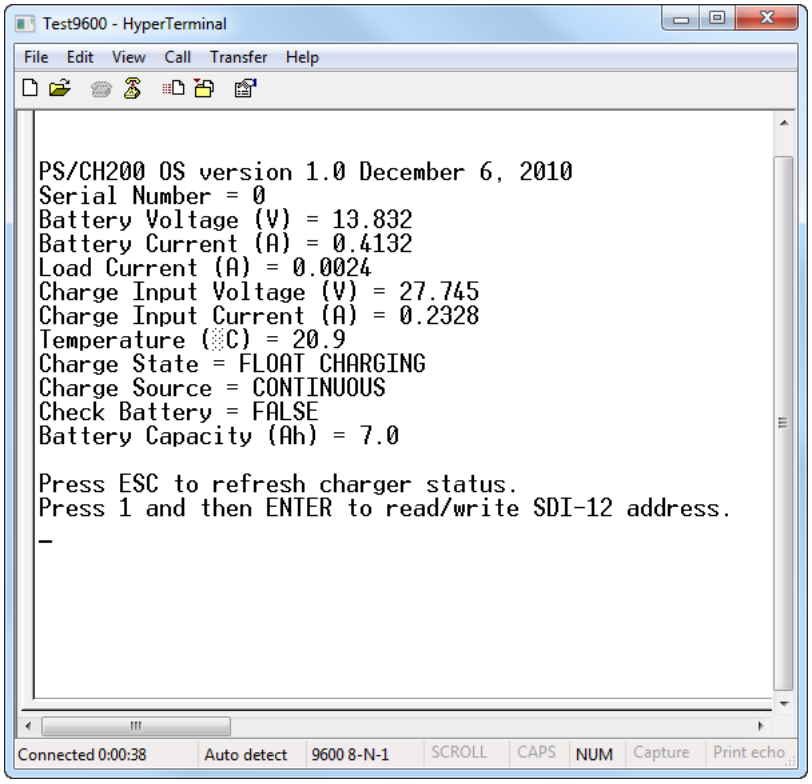


Click on Apply and then OK, and the terminal screen should open. Click inside the terminal screen and press ESC (may have to press ESC twice).

To refresh the charger status screen, press ESC.

To change the SDI-12 address of the PS200/CH200, press 1 and then Enter.

Also, as shown in Section 5.2.2 SDI-12 Measurements, the PS200/CH200 RS-232 text based interface can be viewed from within Campbell Scientific DevConfig.



5.3.2 RS-232 Communications with a Campbell Scientific Datalogger

The Campbell Scientific CRBasic commands “SerialOut” and “SerialIn” are designed for RS-232 communications in general and can be used with a PS200/CH200 to read battery voltage (V), battery current (A), load current (A), charge input voltage (V), charge input current (A), temperature (°C), charge state, charge source (solar or continuous), and a check battery flag (true or false).

All PS200/CH200 serial communications are via its four-pin connector. There are two RS-232 interface cables that can be used to connect a datalogger to the PS200/CH200.

To use the RS-232 9-pin connector on the face of the datalogger use RS-232 serial cable item# 20770 with an additional null modem cable such as Campbell Scientific item# 18663.

To use control ports configured as RS-232 serial ports (COM1 – COM4) use Campbell Scientific item # 25356.

Campbell Scientific Item #25356 RS-232 Datalogger Cable Wiring (see Figure 5-3)

Green: Tx COM Port
White: Rx COM Port
Black: G
Clear: G

The RS-232 settings are:

Bits per second:	9600
Data bits:	8
Parity:	None
Stop bits:	1
Flow Control:	None

Green: C1 (Tx)
White: C2 (Rx)
Black: G
Shield: G

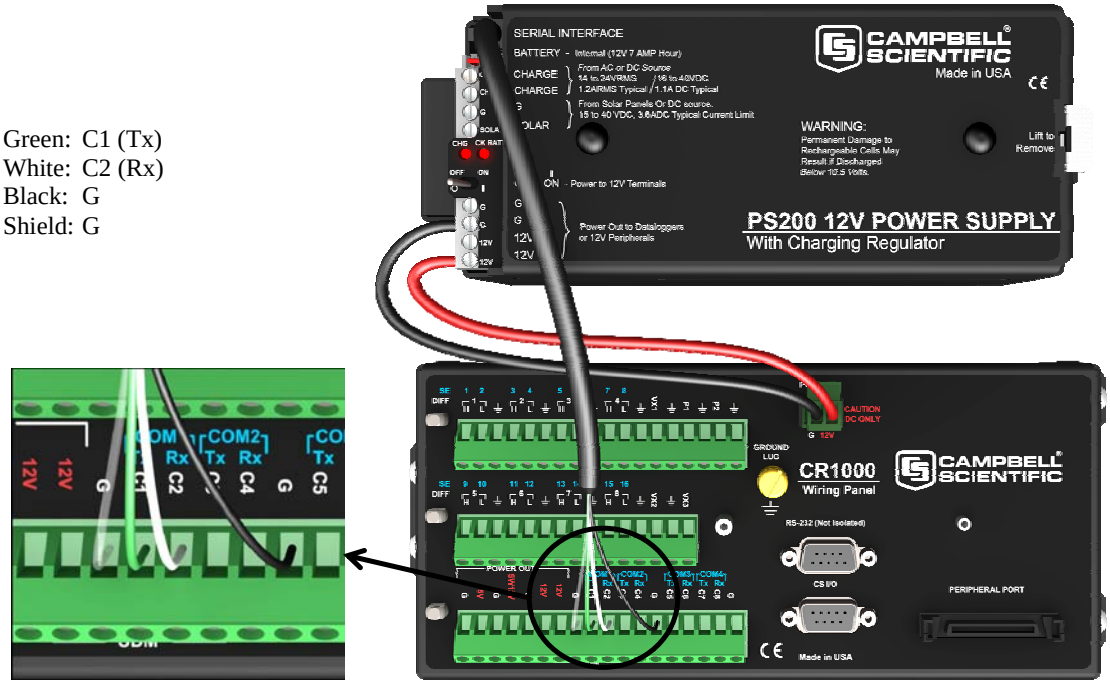


Figure 5-3. PS200 Connected to a CR1000 via RS-232

The PS200/CH200 will answer via RS-232 when commanded by an RS-232 host. The host initiates communication by sending an ASCII command string (ASCII '>'). Communication must be terminated with a carriage return, denoted below by <CR> which is the ASCII hexadecimal character 0D. The PS200/CH200 response string will begin with the ASCII '*' character followed by the requested data strings separated by commas and then two check sum characters and finally the string termination character, hexadecimal 00. The check sum characters are computed within the PS200/CH200 and deciphered by the datalogger using the CRBasic instruction “CheckSum (ch200string,5,0)”. A correct check sum will produce a result of zero. A non-zero result would indicate that transmission error(s) occurred. See more about how to use the check sum instruction below.

5.3.3 RS-232 Host (Datalogger) Command Strings and PS200/CH200 Response Strings

5.3.3.1 Read Status

Host command string: **RD_STAT><CR>**

PS200/CH200 response string: ***,nnnn,nnnn,….,cc0**

where "nnnn' refers to ASCII character strings that are separated by ASCII commas, "cc" is a two-character CRC nullifier, and 0 is the string terminator. The PS200/CH200 character strings occur in the following order (the sequence order numbers shown below do not occur in the character strings):

1 – Battery Voltage, 2 – Battery Current, 3 – Load Current,
4 – Charge Input Voltage, 5 – Charge Input Current, 6 – Temperature,
7 – Charge State, 8 – Charge Source, 9 – Check Battery.

The character strings are numerical values expressed in ASCII numerals and decimal points except for Charge State and Charge Source, which appear as ASCII letter strings.

The Charge State string will be one of these character strings:

REGULATOR FAULT
NO CHARGE
CURRENT LIMITED
CYCLE CHARGING
FLOAT CHARGING
BATTERY TEST

The Charge Source string will be one of these character strings:

NONE
SOLAR
CONTINUOUS

5.3.3.2 Write Remote Battery Temperature to PS200/CH200

Host command string: **WR_BT(mm.dd)> <CR>**

where **mm.dd** refers to an ASCII string of numerals with a decimal point of temperature in degrees Centigrade within -40 and 100 that overrides the PS200/CH200 internal temperature measurement for 15 minutes or until cancelled by a temperature restore command. The temperature value is used in calculating the optimal battery charge voltage.

PS200/CH200 response string

***Battery Temperature = nnnn,cc0**

where **nnnn** refers to the ASCII numerical value string just sent from the host, and **"cc"** is a two-character CRC nullifier, and 0 is the string terminator.

See Appendix A.1.2 for a programming example using this instruction.

5.3.3.3 Restore Internal Battery Temperature Measurement

Host command string: **RS_INTBT> <CR>**

PS200/CH200 response string: ***OKcc0**

where **"cc"** is a two-character CRC nullifier, and 0 is the string terminator.

See Appendix A.1.2 for a programming example using this instruction.

5.3.3.4 Change Battery Capacity Value in PS200/CH200

Host command string: **WR_BC(xx.xx)> <CR>**

This command sets the battery capacity in Ah, which the PS200/CH200 uses in its charging algorithm.

PS200/CH200 response string: ***Battery capacity = xx.xx <crc> <CR> <LF>**.

See Appendix A.2.2 for a programming example using this instruction.

5.3.3.5 Enter Battery Test Stat

Host command string: **BATT_TEST> <CR>**

This command can be used in a datalogger program to test a battery connected to a PS200/CH200. This command instructs a PS200/CH200 to set its charging voltage to 11.5 VDC for 90 seconds. Generally, the battery voltage will be greater than 11.5 VDC, so this command momentarily causes a PS200/CH200 to stop charging the battery. Then, the datalogger program can analyze the battery charging capacity (see example program). Because the BATT_TEST command stays in effect for only 90 seconds, the datalogger program must issue the BATT_TEST command more frequently than once per 90 seconds for as long as the test is to last.

PS200/CH200 response string: ***OKcc0**

where **"cc"** is a two-character CRC nullifier, and 0 is the string terminator.

See Appendix A.3.2 for a programming example using this instruction.

5.3.3.6 Zero Out Qloss

Host command string: **RESET_QLOSS> <CR>**

PS200/CH200 response string: ***OKcc0**

where **"cc"** is a two-character CRC nullifier, and 0 is the string terminator.

See Appendix A.4.2 for a programming example using this instruction.

5.3.3.7 Read Special PS200/CH200 Settings and Variables

Host command string: **RD_SPECIAL> <CR>**

PS200/CH200 response string: ***,nnnn,nnnn,….,cc0**

where "nnnn' refers to ASCII character strings that are separated by ASCII commas, "cc" is a two-character CRC nullifier, and 0 is the string terminator. The PS200/CH200 character strings occur in the following order (the sequence order numbers shown below do not occur in the character strings):

1 - Target Battery Voltage, 2 - Digital Pot Setting, 3 - Capacity, 4 – Qloss, 5 - Continuous Input Volts.

The character strings are numerical values expressed in ASCII numerals and decimal points except for Charge State and Charge Source, which appear as ASCII letter strings.

See Appendix A.4.2 for a programming example using this instruction.

5.3.3.8 Power Usage

Although this is not an instruction in and of itself the ability to calculate power usage by the battery and the system is a very powerful capability of the PS200/CH200. Measuring and storing power usage is useful to determine system power requirements (power budget) and help to identify devices using too much power.

See Appendix A.5.2 for a programming example.

5.3.4 Datalogger Programming for RS-232 Communication to the PS200/CH200

The normal condition for the PS200/CH200 is that it expects SDI-12 communication, so when the host first sends RS-232 characters, the first character might be missed. Therefore, the host should initiate RS-232 communication by sending two backspace characters (hexadecimal 08). See the example below.

```
'Subroutine sends two back space characters to the PS/CH200 to
 'wake it up and switch over to RS-232 mode.
Sub WAKEUP
  SerialOut (COMPRT,CHR(&H08),"",1,3)
  SerialOut (COMPRT,CHR(&H08),"",1,3)
EndSub
```

In processing the strings from a PS200/CH200, use the CRBasic instruction "SerialIn" and then use the instruction "SplitStr" to separate the numerical values into individual values. The CRBasic instruction "Checksum" can optionally be used to validate data from a PS200/CH200. If "CheckSum" is to be used, two successive "SerialIn" instructions should be used—one to discard everything up to and including the "*" character which starts a PS200/CH200 response string, and another to obtain all the characters after the "*" up to and including the string terminator character, zero.

Below is an example CRBasic program that reads the status information from a PS200/CH200 via RS-232.

```
'CR1000 Series Datalogger
'Program: RS-232_Status_Info.CR1
'Date: 1.November.2010
'Ver: A
'
'Notes: Program returns all information from PS200/CH200.
'Connect the PS200/CH200 to RS-232 port COM2 of the CR1000.
'Use Campbell Scientific SDI-12 cable part # 25356.
'
'RS-232 CABLE TO CR1000 WIRING
'-----------------------------
'GREEN: C3 (TX)
'WHITE: C4 (RX)
'BLACK: G
'CLEAR: G
'
  'Public Variables
Public PanelTempC  'Datalogger panel temperature: Celsius
  'Change this constant if using a different com port on the CR1000.
Const COMPRT = COM2
  'Hexadecimal equivalent of a carriage return.
Const CR = CHR(&H0D)
  'It's a comma. Used to sort PS/CH200 strings.
Const COMMA = CHR(&H2C)
  'Command sent to the PS/CH200 to get status information.
Const CH200STATS_CMD = "RD_STAT>"
  'Holds the check sum value of the received PS/CH200 string.
Public Check_sum
  'Used to catch clutter from the PS/CH200
Dim xmit_str As String
  'Holds the entire status string coming from the PS/CH200.
Dim CH200string As String * 140
  'Holds parsed string values.
Dim CH200_M0(9) As String * 20
  'Battery voltage: VDC
Public VBatt
  'Current going into, or out of, the battery: Amps
Public IBatt
  'Current going to the load: Amps
Public ILoad
  'Voltage coming into the charger: VDC
Public V_in_chg
  'Current coming into the charger: Amps
Public I_in_chg
  'Charger temperature: Celsius
Public Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Public ChargeState As String
  'Charging source: None, AC, or Solar
Public ChargeSource As String
  'Check battery error: 0=normal, 1=check battery
Public Ck_Batt
  'Word or phrase equivalent of check battery error.
Public CheckBattery As String
  'Array to hold the names of the battery error information.
Dim CheckBatteryArr(2) As String

  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
```

```
  Minimum (1,VBatt,FP2,0,False)
  Minimum (1,IBatt,FP2,0,False)
  Minimum (1,ILoad,FP2,0,False)
  Minimum (1,V_in_chg,FP2,0,False)
  Minimum (1,I_in_chg,FP2,0,False)
  Minimum (1,Chg_TmpC,FP2,0,False)
  Maximum (1,VBatt,FP2,False,False)
  Maximum (1,IBatt,FP2,False,False)
  Maximum (1,ILoad,FP2,False,False)
  Maximum (1,V_in_chg,FP2,False,False)
  Maximum (1,I_in_chg,FP2,False,False)
  Maximum (1,Chg_TmpC,FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Subroutine sends two back space characters to the PS/CH200 to
  'wake it up and switch over to RS-232 mode.
Sub WAKEUP
  SerialOut (COMPRT,CHR(&H08),"",1,3)
  SerialOut (COMPRT,CHR(&H08),"",1,3)
EndSub

  'Main Program
BeginProg

    'Load arrays with words to associated with the check battery values
    'from the PS/CH200.
  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  Scan (5,Sec,0,0)
    PanelTemp(PanelTempC,_60Hz)       'Measure datalogger panel temperature.
    CallTable Hour
  NextScan

    'SlowSequence with RS-232 measurements
  SlowSequence
  SerialOpen (COMPRT,9600,3,0,150)
  Scan(30,sec,0,0)
      'Get the PS/CH200 status information.
      'Call WAKEUP to wake up the PS/CH200.
    Call WAKEUP
      'Send "read status" command to PS200 followed by a carriage return.
    SerialOut (COMPRT,CH200STATS_CMD,"",1,3)
    SerialOut (COMPRT,CR,"",1,3)
      'Discard characters in the receiver buffer up to and including asterisks "*"
    SerialIn (xmit_str,COMPRT,50,"*",40)
      'Capture the PS/CH200 response string after the asterisks and including the
      'string terminator hexadecimal zero (&H00).
    SerialIn (CH200string,COMPRT,50,&H00,140)
    Check_sum = CheckSum (CH200string,5,0 )
      'For the string to be correct the check sum must be zero.
    If Check_sum = 0 Then
        'Split up the main string from PS/CH200 into separate components.
      SplitStr (CH200_M0(),CH200string,COMMA,9,4)
        'Sort out the components to specific variables.
      VBatt = CH200_M0(1)
      IBatt = CH200_M0(2)
      ILoad = CH200_M0(3)
```

```
    V_in_chg = CH200_M0(4)
    I_in_chg = CH200_M0(5)
    Chg_TmpC = CH200_M0(6)
    ChargeState = CH200_M0(7)
    ChargeSource = CH200_M0(8)
    Ck_Batt = CH200_M0(9)
      'Values for check battery start with zero. Have to shift the value
      'by one to line it up with the correct words in the array.
    CheckBattery = CheckBatteryArr(Ck_Batt + 1)
  EndIf
 NextScan
EndProg
```

6. Charging Details

6.1 Charging Algorithm

The PS200/CH200 offers both Continuous and Solar charging inputs. The Continuous charging input has a user adjustable input current limit with a maximum (default) value of 1.1 A DC to help protect AC/AC transformers and AC/DC converters. The 3.6 ADC typical current limit of the PS200/CH200 Solar charging input is well suited for 70 W solar panels. Typical Continuous charging inputs would be AC/AC transformers or AC/DC converters in which a charge voltage is continuously applied except for line power outages. When powered from a Continuous charging input, the PS200/CH200 tries to maintain the battery at a float charge voltage, which can continue indefinitely without overcharging the battery.

When powered from a Solar charging input, the PS200/CH200 utilizes a two-step constant voltage charging algorithm, which is the preferred method for rapidly charging VRLA batteries[1]. When powered from the Solar charging input, the battery charge deficit (Qloss) is compared to the specified battery capacity in order to determine if aggressive cycle charging is necessary. Cycle charging is then utilized if Qloss is determined to be greater than 20% of the battery capacity. Upon detection that the battery is near full charge, the constant voltage charging level is reduced from the aggressive cycle charging voltage to the non-aggressive float charge voltage. To prevent overcharging and unwanted gassing of the battery.

In the PS200, the battery capacity is set to 7 Ah hours at the factory, whereas the battery capacity is left at the default value of 0 Ah in the CH200 where the user provides the battery. If the battery capacity is left at the default value of zero, then Qloss always equals 0, which disables cycle charging. Consequently, CH200 users must enter in the battery capacity in order to enable the aggressive cycle charging capability of the two-step constant voltage charging algorithm.

Discharged VRLA batteries can initially accept large charging currents, often resulting in the charger being unable to initially maintain a constant voltage because of current limiting by the charge source or the charger itself. As a result, a typical two-step constant voltage charging cycle usually consists of three distinct stages; a current limited charge stage, a constant cycle voltage charge stage, and a constant float voltage charge stage, as illustrated in Figure 6-1. The current limited stage and/or the constant cycle charge voltage stage may not occur if the battery size is small relative to the current capability of the charge source, or if the battery is near full charge at the beginning of a charge cycle.

Normally cycle charging terminates and float charging begins when Qloss has been reduced to zero. Two exceptions exist, both of which help prevent overcharging of the battery. One exception to this normal cycle to float transition

occurs if cycle charging has been on-going for 8 hours without interruption, indicating some sort of battery problem. In this case cycle charge will be terminated and Qloss will be zeroed. The other exception to the normal cycle to float transition when Qloss has been reduced to zero occurs if during cycle charging the battery charge current is reduced to below C/100, where C is the user entered battery capacity. This condition indicates the charger is trying to aggressively cycle charge a fully charged battery, perhaps because a new fully charged battery was swapped in replacing an old battery which suffered from a significant Qloss. During cycle charging if the battery charge current falls below C/100, then cycle charge will be terminated and Qloss will be zeroed.

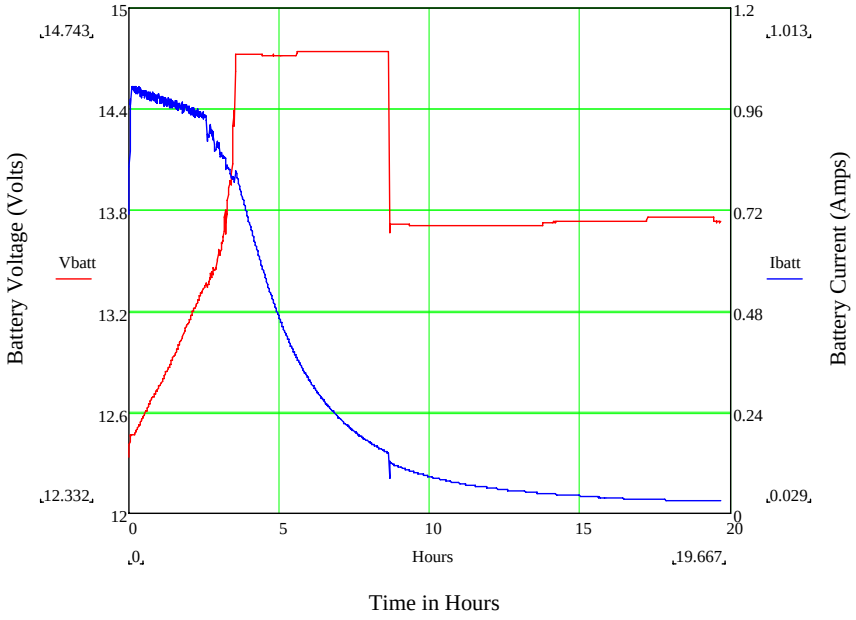


Figure 6-1. 2 Step Constant Voltage Battery Charging by PS200/CH200

6.2 Maximum Power Point Tracking

The current and power versus voltage for a 70 W solar panel are illustrated in Figure 6-2. As can be seen from the figure, a Maximum Power Point of operation exists for solar panels. Adjusting the load on the solar panel so it operates at this Maximum Power Point is referred to as Maximum Power Point Tracking (MPPT). MPPT is beneficial when insufficient power is available from the charge source, which is the case during current limited charging. The somewhat noisy charging current and voltage observed in Figure 6-1 during the initial current limited charging stage is due to the MPPT algorithm of the PS200/CH200 searching for the maximum power point of the associated solar panel.

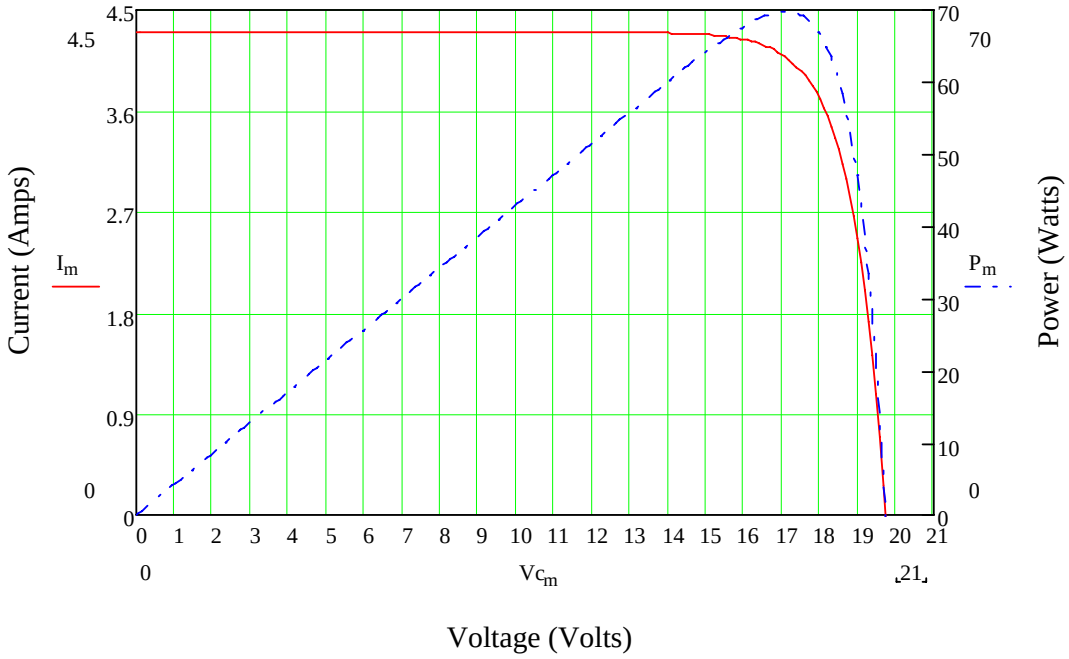


Figure 6-2. 70 W Solar Panel I – V and Power Characteristics

7. A100 Null Modem Adapter

The A100 adapter has two 9-pin CS I/O ports with a null modem between them. The ports are used to connect two 9-pin devices (e.g. modems or RF radios) that would normally be connected to the CS I/O port of a Campbell datalogger. The charger supplies 12 volts and 5 volts to the appropriate pins on the connector for powering the connected devices. This functionality is required in a station where a datalogger will not be present, such as a phone-to-RF base station. The A100 cannot be used as a null modem between two RS-232 devices.

Figure 7-1. Null Modem Connections

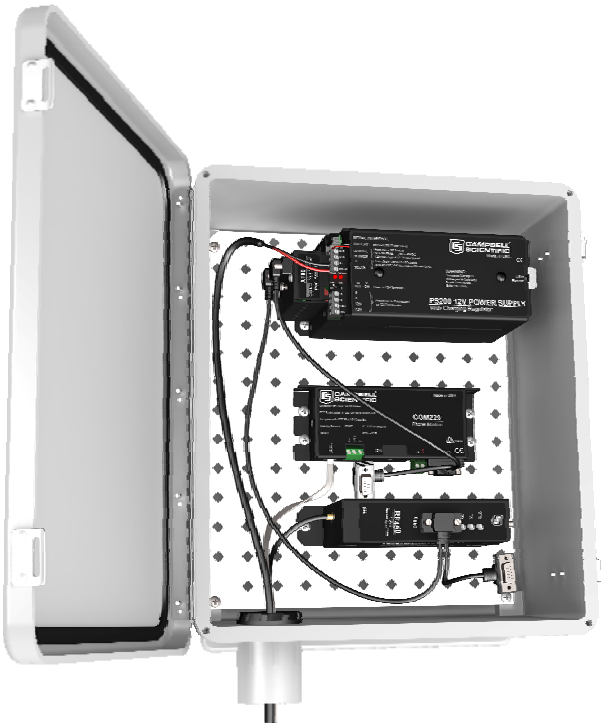


Figure 7-2. PS200 with A100 Module using a COM220 and RF450

8. A105 Additional 12 V Terminals Adapter

The A105 adapter adds four 12 V terminals and four ground terminals to a PS200/CH200 charging regulator. The extra terminals make it easier to wire multiple continuously powered 12V DC devices to the power supply.

Figure 8-1. A105 Adapter

9. References

1 – Genesis Application Manual – Genesis NP and NPX Series US-NP-AM-002, June 2006.

Appendix A. Advanced Programming Techniques

The following CR1000 programming examples show how to use both SDI-12 and RS-232 advanced instruction programming techniques. All of these example programs can be pulled down from Campbell Scientific's website at www.campbellsci.com. After connecting to the home page select "Support" and then "Downloads" from the pull-down menu. Search for PS200.

Wiring for the communication interface cable for all programs is as follows.

SDI12 CABLE (ITEM# 20769) WIRING
WHITE: C3
BLACK: G
CLEAR: G

RS-232 CABLE (ITEM# 25356) WIRING
GREEN: C3 (Tx)
WHITE: C4 (Rx)
BLACK: G
CLEAR: G

A.1 Write/Reset Remote Battery Temperature

The purpose of these instructions is to cause the CH200 to change its charging algorithm based on an external temperature of an external battery vs. the normal internal temperature of the CH200. To prevent the battery from being over, or under, charged the CH200 must know the temperature of the battery itself and make adjustments accordingly.

A.1.1 Write/Reset Remote Battery Temperature SDI12 Programming Example

```
'CR1000 Series Datalogger
'Program: SDI12_Set_Batt_Temp.CR1
'Date: 7.October.2010
'Ver: A
'
'Notes: This program sets the battery temperature value via SDI-12.
'PS200/CH200 configured with SDI-12 address 0 (zero).
'
'Use Campbell Scientific SDI-12 cable part # 20769.
'
'SDI-12 CABLE TO CR1000 WIRING
'------------------------------
'WHITE: C3
'BLACK: G
'CLEAR: G
'
'This program uses the SDI-12 command XT! to overwrite the
'internal CH200 temperature if the user sets the variable
'SetBattTemp = True.
'the internal CH200 temperature is used if SetBattTemp = False.

  'Public Variables
Public SetBattTemp As Boolean

Public CH200_M0(6)                    'Array to hold data coming from the PS200/CH200
```

```
  'Alias names for array elements.
Alias CH200_M0(1)=VBatt          'Battery voltage: VDC
Alias CH200_M0(2)=IBatt          'Current going into, or out of, the battery: Amps
Alias CH200_M0(3)=ILoad          'Current going to the load: Amps
Alias CH200_M0(4)=V_in_chg       'Voltage coming into the charger: VDC
Alias CH200_M0(5)=I_in_chg       'Current coming into the charger: Amps
Alias CH200_M0(6)=Chg_TmpC       'Charger temperature: Celsius

Public PanelTempC                'Datalogger panel temperature: Celsius
Public ExtBatTempC               'Battery temperature from an external source.

  'SDI-12 formatted external battery temperature value.
Public SDI12Command As String
  'Response from PS/CH200. Returns the address of the unit and "OK" if
  'all went well.
Public SDI12Result As String
  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (6,CH200_M0(),FP2,0,False)
  Maximum (6,CH200_M0(),FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

BeginProg
    'Set SetBattTemp to TRUE to have ExtBatTempC overwrite internal
    'CH200 temperature.
  SetBattTemp = True
  Scan (5,Sec,0,0)
    PanelTemp (PanelTempC,250)
      'Measure battery temp with TC attached to external battery
    TCDiff (ExtBatTempC,1,mV2_5C,1,TypeT,PanelTempC,True ,0,_60Hz,1.0,0)
    CallTable Hour
  NextScan

    'Put SDI-12 measurements in a SlowSequence to avoid any missed main
    'scans if there are any sensor problems
  SlowSequence
  Scan (30,Sec,0,0)
    SDI12Recorder(CH200_M0(),3,0,"MC!",1.0,0)
    If SetBattTemp = True Then           'Overwrite CH200 internal temperature
      SDI12Command = "XT" & FormatFloat (ExtBatTempC,"%4.2f") & "!"
    Else
       'Set the CH200 back to using its internal temperature.
      SDI12Command = "XTR!"
    EndIf
    SDI12Recorder (SDI12Result,3,0,SDI12Command,1.0,0)
  NextScan
EndProg
```

A.1.2 Write/Reset Remote Battery Temperature RS-232 Programming Example

```
'CR1000 Series Datalogger
'Program: RS-232_Set_Batt_Temp.CR1
'Date: 1.November.2010
'Ver: A
'
'Notes:  This program sets the battery temperature value.
'Connect the PS200/CH200 to RS-232 port COM2 of the CR1000.
'Use Campbell Scientific RS-232 cable part # 25356.
'
'RS-232 CABLE TO CR1000 WIRING
'-----------------------------
'GREEN: C3 (TX)
'WHITE: C4 (RX)
'BLACK: G
'CLEAR: G
'
'TYPE 'T' THERMOCOUPLE USED TO MEASURE
'EXTERNAL BATTERY TEMPERATURE WIRING
'--------------------------------------
'BLUE: 1H
'RED: 1L
'
' This program uses the  RS-232 command WR_BT to overwrite the internal CH200 temperature if
' the user sets the variable SetBattTemp = True.
' The external CH200 temperature is used if SetBattTemp = True.

'Constants & Public Variables
  'Change this constant if using a different com port on the CR1000.
Const COMPRT = COM2
  'Hexadecimal equivalent of a carriage return.
Const CR = CHR(&H0D)
  'It is a comma. Used to sort PS/CH200 strings.
Const COMMA = CHR(&H2C)

Public PanelTempC                    'Datalogger panel temperature: Celsius
  'Battery temperature from an external thermocouple.
Public ExtBatTempC

'Variables and constants used to get status information from the PS/CH200
  'Command sent to the PS/CH200 to get status information.
Const CH200STATS_CMD = "RD_STAT>"
  'Holds the check sum value of the received PS/CH200 string.
Public Check_sum
  'Used to catch clutter from the PS/CH200
Dim xmit_str As String
  'Holds the entire status string coming from the PS/CH200.
Dim CH200string As String * 140
  'Holds parsed string values.
Dim CH200_M0(9) As String * 20
  'Battery voltage: VDC
Public VBatt
  'Current going into, or out of, the battery: Amps
Public IBatt
  'Current going to the load: Amps
Public ILoad
  'Voltage coming into the charger: VDC
Public V_in_chg
```

```
  'Current coming into the charger: Amps
Public I_in_chg
  'Charger temperature: Celsius
Public Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Public ChargeState As String
  'Charging source: None, AC, or Solar
Public ChargeSource As String
  'Check battery error: 0=normal, 1=check battery
Public Ck_Batt
  'Word or phrase equivalent of check battery error.
Public CheckBattery As String
  'Array to hold the names of the battery error information.
Dim CheckBatteryArr(2) As String

'Variables & constants used to write an external battery temperature to the
  'PS/CH200 and reset it.
  'True - use external battery temp. False - use PS/CH200 temp.
Public SendBattTemp As Boolean
  'Command sent to the PS/CH200 to write external battery temperature.
Const WRITE_BATT_TEMP_CMD = "WR_BT("
  'Command sent to the PS/CH200 to restore.
Const RESTR_BATT_TEMP_CMD = "RS_INTBT>"
  'String storing the formatted ASCII equivalent of the ext battery temp.
Public ExtBatString As String * 5
  'String coming back from either the "WR_BT" or "RS_INTBT" command
Public ReturnStr As String * 40
  'Number signifying how far into the return string for "RS_INTBT"
  'command the "OK" is located.
Dim OK_Loc

  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (1,VBatt,FP2,0,False)
  Minimum (1,IBatt,FP2,0,False)
  Minimum (1,ILoad,FP2,0,False)
  Minimum (1,V_in_chg,FP2,0,False)
  Minimum (1,I_in_chg,FP2,0,False)
  Minimum (1,Chg_TmpC,FP2,0,False)
  Maximum (1,VBatt,FP2,False,False)
  Maximum (1,IBatt,FP2,False,False)
  Maximum (1,ILoad,FP2,False,False)
  Maximum (1,V_in_chg,FP2,False,False)
  Maximum (1,I_in_chg,FP2,False,False)
  Maximum (1,Chg_TmpC,FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable
```

```
 'Subroutine sends two back space characters to the PS/CH200 to
 'wake it up and switch over to RS-232 mode.
Sub WAKEUP
 SerialOut (COMPRT,CHR(&H08),"",1,3)
 SerialOut (COMPRT,CHR(&H08),"",1,3)
EndSub

 'Main Program
BeginProg

  'Load arrays with words to associated with the check battery values
  'from the PS/CH200.
 CheckBatteryArr(1) = "Normal"
 CheckBatteryArr(2) = "Check Battery"

 Scan (5,Sec,0,0)
  PanelTemp(PanelTempC,_60Hz) 'Measure datalogger panel temperature.
  CallTable Hour
 NextScan

  'SlowSequence with RS-232 measurements
 SlowSequence
 SendBattTemp = True
  'Configure the COM port used with the PS/CH200
 SerialOpen (COMPRT,9600,3,0,150)
 Scan(30,sec,0,0)
   'Measure battery temp with TC attached to external battery
  TCDiff (ExtBatTempC,1,mV2_5C,1,TypeT,PanelTempC,True ,0,_60Hz,1.0,0)
   'Send the external battery voltage temperature if SendBattTemp is true.
  If SendBattTemp Then
     'Send out the external battery temperature and format it.
     'Convert the digital value for the external battery temperature to an
     'ASCII string. Specify four digits with two decimals after the decimal point.
    ExtBatString = FormatFloat (ExtBatTempC,"%04.2f")
     'Completed string sent to the PS/CH200 must be "WR_BT(nn.nn)>" where
     'nn.nn are the ASCII numeric external battery temperature in Celsius.
    xmit_str = WRITE_BATT_TEMP_CMD + ExtBatString + ")>"
  Else                                  'SendBattTemp is false so send the reset string.
   xmit_str = RESTR_BATT_TEMP_CMD
  EndIf
   'Send WAKEUP to wake up the PS/CH200.
  Call WAKEUP
   'Send out the PS/CH200 command followed by a carriage return.
  SerialOut (COMPRT,xmit_str,"",1,3)
  SerialOut (COMPRT,CR,"",1,3)
   'Bring in the result from either command and process it.
   'Discard characters in the receiver buffer up to and including asterisks "*"
  SerialIn (xmit_str,COMPRT,50,"*",40)
   'Capture the PS/CH200 response string after the asterisks and including the
   'string terminator hexadecimal zero (&H00).
  SerialIn (CH200string,COMPRT,50,&H00,140)
   'Return strings are different for both instructions. Have to process them
   'different as well.
```

```
  If SendBattTemp Then
    'PS/CH200 should return the battery temperature sent to it.
   SplitStr (ReturnStr,CH200string,"String",1,0)
  Else                                  'Reset command sent to the PS/CH200. Look for "OK".
   OK_Loc = InStr(1,xmit_str,"O",2)
   ReturnStr = Mid (xmit_str,OK_Loc,2)
  EndIf

   'Get the PS/CH200 status information.
   'Send WAKEUP to wake up the PS/CH200.
  Call WAKEUP
   'Send "read status" command to PS200 followed by a carriage return.
  SerialOut (COMPRT,CH200STATS_CMD,"",1,3)
  SerialOut (COMPRT,CR,"",1,3)
   'Discard characters in the receiver buffer up to and including asterisks "*"
  SerialIn (xmit_str,COMPRT,50,"*",40)
   'Capture the PS/CH200 response string after the asterisks and including the
   'string terminator hexadecimal zero (&H00).
  SerialIn (CH200string,COMPRT,50,&H00,140)
  Check_sum = CheckSum (CH200string,5,0 )
   'For the string to be correct the check sum must be zero.
  If Check_sum = 0 Then
    'Split up the main string from PS/CH200 into separate components.
   SplitStr (CH200_M0(),CH200string,COMMA,9,4)
    'Sort out the components to specific variables.
   VBatt = CH200_M0(1)
   IBatt = CH200_M0(2)
   ILoad = CH200_M0(3)
   V_in_chg = CH200_M0(4)
   I_in_chg = CH200_M0(5)
   Chg_TmpC = CH200_M0(6)
   ChargeState = CH200_M0(7)
   ChargeSource = CH200_M0(8)
   Ck_Batt = CH200_M0(9)
    'Values for check battery start with zero. Have to shift the value
    'by one to line it up with the correct words in the array.
   CheckBattery = CheckBatteryArr(Ck_Batt + 1)
  EndIf
 NextScan
EndProg
```

A.2 Change Battery Capacity

Battery capacity is used by the PS/CH200 to determine when to switch from a higher current charge state, such as cycle charging or current limited, to a lower current charging state – float charging. The reason for the switching back and forth is to charge the battery in the quickest possible time without damaging the battery. By default the PS200 is set to a battery capacity of 7 Amp-hours and the CH200 is set to a battery capacity of zero. Setting the battery capacity to zero forces the CH200 to only charge at the lower float charging rate. When the CH200 is built it is not known what Amp-hour capacity of battery will be used with it so it is purposely set to a lower charging rate to protect the user.

Setting the battery capacity can be done via Device Configuration Utility or in the field using the following programs. Both of the following programs look at a variable that contains the battery capacity to be used with the charger. When this value is changed the datalogger will automatically send the new value to the charger.

A.2.1 Change Battery Capacity SDI12 Programming Example

```
'CR1000 Series Datalogger
'Program: SDI12_Set_Batt_Cap.CR1
'Date: 7.October.2010
'Ver: A
'
'Notes: This program sets the battery capacity in the PS/CH200
'via SDI-12.
'PS200/CH200 configured with SDI-12 address 0 (zero).
'
'Use Campbell Scientific SDI-12 cable part # 20769.
'
'SDI-12 CABLE TO CR1000 WIRING
'-----------------------------
'WHITE: C3
'BLACK: G
'CLEAR: G
'
  'Public Variables
Public PanelTempC                 'Datalogger panel temperature: Celsius

Public CH200_M0(6)                'Array to hold data coming from the PS200/CH200
  'Alias names for array elements.
Alias CH200_M0(1)=VBatt           'Battery voltage: VDC
Alias CH200_M0(2)=IBatt           'Current going into, or out of, the battery: Amps
Alias CH200_M0(3)=ILoad           'Current going to the load: Amps
Alias CH200_M0(4)=V_in_chg        'Voltage coming into the charger: VDC
Alias CH200_M0(5)=I_in_chg        'Current coming into the charger: Amps
Alias CH200_M0(6)=Chg_TmpC        'Charger temperature: Celsius

Public CH200_MX(4)                'Array to hold extended data from the PS200/CH200
  'Alias names for array elements.
Alias CH200_MX(1) = BattTargV     'Battery charging target voltage.
Alias CH200_MX(2) = DgtlPotSet    'Digital potentiometer setting.
Alias CH200_MX(3) = BattCap       'Present battery capacity.
Alias CH200_MX(4) = Qloss         'Battery charge deficit.

  'SDI-12 formatted battery capacity value.
Public SDI12command As String
  'Response from PS/CH200. Returns the address of the unit and "OK" if
  'all went well.
Public SDI12result As String
Public NewBattCap                 'New battery capacity.
```

```
'Define Data Tables
  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (6,CH200_M0(),FP2,0,False)
  Maximum (6,CH200_M0(),FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Save the new and old battery capacity values and a time
  'stamp indicating when it was changed.
DataTable (BattCapChng,True,48)
  Sample (1,NewBattCap,FP2)
  Sample (1,BattCap,FP2)
EndTable

  'Main Program
BeginProg
    'For this exercise change the battery capacity to 12 Amp-hours.
    'Value can be changed on the fly by changing the variable.
  NewBattCap = 12
  Scan (5,Sec,0,0)
    PanelTemp (PanelTempC,250)
    CallTable Hour
  NextScan

  SlowSequence
  Scan (30,Sec,3,0)
     'Get PS200/CH200 values.
    SDI12Recorder (CH200_M0(),3,0,"M3!",1.0,0)
     'Get present battery capacity settings.
    SDI12Recorder (CH200_MX(),3,0,"M6!",1.0,0)
     'If the present battery capacity is not the same as the new
     'battery capacity then send the new one to the charger.
    If BattCap <> NewBattCap Then
      CallTable BattCapChng
      SDI12command = "XC" & FormatFloat (NewBattCap,"%4.1f") & "!"
      SDI12Recorder (SDI12result,3,0,SDI12command,1.0,0)
    EndIf
  NextScan
EndProg
```

A.2.2 Change Battery Capacity RS-232 Programming Example

```
'CR1000 Series Datalogger
'Program: RS-232_Set_Batt_Cap.CR1
'Date: 1.November.2010
'Ver: A
'
'Notes:  This program sets the battery capacity value in the PS/CH200.
'Connect the PS200/CH200 to RS-232 port COM2 of the CR1000.
'Use Campbell Scientific RS-232 cable part # 25356.
'
'RS-232 CABLE TO CR1000 WIRING
'------------------------------
'GREEN: C3 (TX)
'WHITE: C4 (RX)
'BLACK: G
'CLEAR: G
'
'This program uses the RS-232 command "WR_BC(nnnn)>" to write battery capacity to the PS/CH200.
'It also collects extended PS/CH200 information using the "RD_SPECIAL>" command.

'Constants & Public Variables
  'Change this constant if using a different com port on the CR1000.
Const COMPRT = COM2
  'Hexadecimal equivalent of a carriage return.
Const CR = CHR(&H0D)
  'It is a comma. Used to sort PS/CH200 strings.
Const COMMA = CHR(&H2C)

Public PanelTempC                    'Datalogger panel temperature: Celsius
  'Battery temperature from an external thermocouple.
Public ExtBatTempC

'Variables and constants used to get status information from the PS/CH200
  'Command sent to the PS/CH200 to get status information.
Const CH200STATS_CMD = "RD_STAT>"
  'Holds the check sum value of the received PS/CH200 string.
Public Check_sum
  'Used to catch clutter from the PS/CH200
Dim xmit_str As String
  'Holds the entire status string coming from the PS/CH200.
Dim CH200string As String * 140
  'Holds parsed string values.
Dim CH200_M0(9) As String * 20
  'Battery voltage: VDC
Public VBatt
  'Current going into, or out of, the battery: Amps
Public IBatt
  'Current going to the load: Amps
Public ILoad
  'Voltage coming into the charger: VDC
Public V_in_chg
  'Current coming into the charger: Amps
Public I_in_chg
  'Charger temperature: Celsius
```

```
Public Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Public ChargeState As String
  'Charging source: None, AC, or Solar
Public ChargeSource As String
  'Check battery error: 0=normal, 1=check battery
Public Ck_Batt
  'Word or phrase equivalent of check battery error.
Public CheckBattery As String
  'Array to hold the names of the battery error information.
Dim CheckBatteryArr(2) As String

'Variables & constants used to write the new battery capacity
'to the PS/CH200.
  'Command sent to the PS/CH200 to write external battery temperature.
  'Instruction responds with "OK" if all went well.
Const WRITE_BATT_CAP_CMD = "WR_BC("
Public NewBattCap                    'New battery capacity.
  'String storing the formatted ASCII equivalent of the ext battery
  'temp.
Public NewBattString As String * 5
  'String coming back from either the "WR_BT" or "RS_INTBT" command.
Public ReturnStr As String * 40

  'Command to get extended status information.
Const EXTSTATS_CMD = "RD_SPECIAL>"
  'Array to hold extended data from the PS200/CH200
Public CH200_MX(4) As String * 20
Public BattTargV                     'Battery charging target voltage.
Public DgtlPotSet                    'Digital potentiometer setting.
Public BattCap                       'Present battery capacity.
Public Qloss                         'Battery charge deficit.

 'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (1,VBatt,FP2,0,False)
  Minimum (1,IBatt,FP2,0,False)
  Minimum (1,ILoad,FP2,0,False)
  Minimum (1,V_in_chg,FP2,0,False)
  Minimum (1,I_in_chg,FP2,0,False)
  Minimum (1,Chg_TmpC,FP2,0,False)
  Maximum (1,VBatt,FP2,False,False)
  Maximum (1,IBatt,FP2,False,False)
  Maximum (1,ILoad,FP2,False,False)
  Maximum (1,V_in_chg,FP2,False,False)
  Maximum (1,I_in_chg,FP2,False,False)
  Maximum (1,Chg_TmpC,FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable
```

```
 'Save time stamp and changes to battery capacity for record
 'keeping.
DataTable (BattCap,True,100)
  Sample (1,BattCap,FP2)
  Sample (1,NewBattCap,FP2)
EndTable

  'Subroutine sends two back space characters to the PS/CH200 to
  'wake it up and switch over to RS-232 mode.
Sub WAKEUP
  SerialOut (COMPRT,CHR(&H08),"",1,3)
  SerialOut (COMPRT,CHR(&H08),"",1,3)
EndSub

  'Main Program
BeginProg

    'Load arrays with words to associated with the check battery values
    'from the PS/CH200.
  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  Scan (5,Sec,0,0)
    PanelTemp(PanelTempC,_60Hz)       'Measure datalogger panel temperature.
    CallTable Hour
  NextScan

    'SlowSequence with RS-232 measurements
  SlowSequence
    'For this exercise a battery capacity of 12 will be written to the CH200.
  NewBattCap = 12
    'Configure the COM port used with the PS/CH200
  SerialOpen (COMPRT,9600,3,0,150)
  Scan(30,sec,0,0)

    'NOTE: THE ORDER IN WHICH COMMANDS ARE SENT, AND READ FROM, THE CH200
    'IS VERY IMPORTANT! KEEP THE STRUCTURE AS SHOWN IN THIS EXAMPLE. IT
    'WILL TAKE TWO COMPLETE SCAN CYCLES TO UPDATE ALL VARIABLE
    'INFORMATION.

      'Get the PS/CH200 extended status information.
      'Send WAKEUP to wake up the PS/CH200.
    Call WAKEUP
      'Send "read extended status" command to PS/CH200 followed by a carriage
      'return.
    SerialOut (COMPRT,EXTSTATS_CMD,"",1,3)
    SerialOut (COMPRT,CR,"",1,3)
      'Discard characters in the receiver buffer up to and including asterisks "*"
    SerialIn (xmit_str,COMPRT,50,"*",40)
      'Capture the PS/CH200 response string after the asterisks and including the
      'string terminator hexadecimal zero (&H00).
    SerialIn (CH200string,COMPRT,50,&H00,140)
    Check_sum = CheckSum (CH200string,5,0 )
      'For the string to be correct the check sum must be zero.
```

```
If Check_sum = 0 Then
   'Split up the main string from PS/CH200 into separate components.
  SplitStr (CH200_MX(),CH200string,COMMA,4,4)
   'Sort out the components to specific variables.
  BattTargV = CH200_MX(1)
  DgtlPotSet = CH200_MX(2)
  BattCap = CH200_MX(3)
  Qloss = CH200_MX(4)
EndIf

  'Get the PS/CH200 status information.
  'Send WAKEUP to wake up the PS/CH200.
Call WAKEUP
  'Send "read status" command to PS200 followed by a carriage return.
SerialOut (COMPRT,CH200STATS_CMD,"",1,3)
SerialOut (COMPRT,CR,"",1,3)
  'Discard characters in the receiver buffer up to and including asterisks "*"
SerialIn (xmit_str,COMPRT,50,"*",40)
  'Capture the PS/CH200 response string after the asterisks and including the
  'string terminator hexadecimal zero (&H00).
SerialIn (CH200string,COMPRT,50,&H00,140)
Check_sum = CheckSum (CH200string,5,0 )
  'For the string to be correct the check sum must be zero.
If Check_sum = 0 Then
   'Split up the main string from PS/CH200 into separate components.
  SplitStr (CH200_M0(),CH200string,COMMA,9,4)
   'Sort out the components to specific variables.
  VBatt = CH200_M0(1)
  IBatt = CH200_M0(2)
  ILoad = CH200_M0(3)
  V_in_chg = CH200_M0(4)
  I_in_chg = CH200_M0(5)
  Chg_TmpC = CH200_M0(6)
  ChargeState = CH200_M0(7)
  ChargeSource = CH200_M0(8)
  Ck_Batt = CH200_M0(9)
   'Values for check battery start with zero. Have to shift the value
   'by one to line it up with the correct words in the array.
  CheckBattery = CheckBatteryArr(Ck_Batt + 1)
EndIf

  'Send new battery capacity to the CH200 if different from the presently
  'stored value.
If NewBattCap <> BattCap Then
   'Save the old and new values before it changes. Store a time stamp.
  CallTable BattCap
   'Convert the new battery capacity value to ASCII text.
   'Specify four digits with one decimal after the decimal point.
   'No leading zeroes.
  NewBattString = FormatFloat (NewBattCap,"%4.1f")
   'Completed string sent to the PS/CH200 must be "WR_BC(nn.n)>" where
   'nn.n are the ASCII numeric external battery temperature in Celsius.
  xmit_str = WRITE_BATT_CAP_CMD + NewBattString + ")>"
   'Send WAKEUP to wake up the PS/CH200.
```

```
    Call WAKEUP
      'Send out the PS/CH200 command followed by a carriage return.
    SerialOut (COMPRT,xmit_str,"",1,3)
    SerialOut (COMPRT,CR,"",1,3)
      'Bring in the result from the command and process it.
      'Discard characters in the receiver buffer up to and including asterisks "*"
    SerialIn (xmit_str,COMPRT,50,"*",40)
      'Capture the PS/CH200 response string after the asterisks and including the
      'string terminator hexadecimal zero (&H00).
    SerialIn (CH200string,COMPRT,50,&H00,140)
      'PS/CH200 should return the new battery capacity sent to it.
    SplitStr (ReturnStr,CH200string,"String",1,0)
  EndIf
 NextScan
EndProg
```

A.3 Enter Battery Test State

Over time batteries die in two possible ways.

1. Shorted cells due to sulphur crystal growth within the cell. Battery simply dies and will not hold a charge, or its voltage, but it will pull in all the current it can get possibly damaging the charging source. The PS/CH200 will detect this problem, set the check battery (CK BATT) light to indicate a problem, and divert current from going into the battery to protect the charging supply. Shorted cells are a frequent problem if the battery is severely discharged and must be replaced.

2. Sulphated cells due to sulphation on the lead plates over time will block the battery's ability to fully recharge. This happens very slowly. When a charging voltage is present the battery will appear to charge up to the target voltage but as soon as any load is put on the battery it will die. The key here is that the battery must be put under a load to detect the problem. Detecting a sulphated cell is easy if the system is running on a solar panel – at night the battery voltage drops very quickly and if not replaced will eventually die. Unfortunately systems running on AC power, and float charging the battery all the time, will not have a problem until it is really needed – when the AC supply is lost. The PS/CH200 charger cannot detect this condition. In order to detect this problem a test must be done on the battery where the charging voltage is removed from it and the battery is monitored to see how it holds up on it's own.

Entering a battery test state causes the PS/CH200 target charging voltage to drop down to 11.5V DC – effectively forcing the system to rely exclusively on the battery to run the system. If the battery cannot keep the system running then the charging supply will kick back in stopping the system from dying. Not sending the battery test instruction will cause the charger to go back to normal within 90 seconds. Depending on the state of the battery and the load on the system it might take only a few minutes or several hours to detect a bad battery. The examples provided for this instruction leave the system running on battery voltage for three hours.

Both the SDI12 and RS-232 programming examples show two different methods of using the battery test instruction.

The first method runs the test based on a calendar date – a battery test is run on the same day of each month.

The second method runs the test based on the number of days between the test. Instead of running the test every month it might be better to run it every 90 or 120 days depending on conditions and preference.

A.3.1 Enter Battery Test State SDI12 Programming Example

A.3.1.1 Calendar Date Battery Test – SDI12

```
'CR1000 Series Datalogger
'Program: SDI12_Battery_Load_Test.CR1
'Date: 7.October.2010
'Ver: A
'
'Notes: This program performs a battery test on a set day into
'each month and at a particular time.
'
'PS200/CH200 configured with SDI-12 address 0 (zero).
'Use Campbell Scientific SDI-12 cable part # 20769.
'
'SDI-12 CABLE TO CR1000 WIRING
'-----------------------------
'WHITE: C3
'BLACK: G
'CLEAR: G
'
PipeLineMode
'Public Variables
  'I like to use the words "on" and "off" vs. "true" and "false" when
  'working with Boolean variables.
Const On = True
Const Off = False

Public PanelTempC  'Datalogger panel temperature: Celsius

  'Store the present time information.
Public rTime(9)                    'Array to store present time
Alias rTime(1) = Year              'assign the alias Year to rTime(1)
Alias rTime(2) = Month             'assign the alias Month to rTime(2)
Alias rTime(3) = DOM               'assign the alias Day to rTime(3)
Alias rTime(4) = Hour              'assign the alias Hour to rTime(4)
Alias rTime(5) = Minute            'assign the alias Minute to rTime(5)
Alias rTime(6) = Second            'assign the alias Second to rTime(6)
Alias rTime(7) = uSecond           'assign the alias uSecond to rTime(7)
Alias rTime(8) = WeekDay           'assign the alias WeekDay to rTime(8)
Alias rTime(9) = Day_of_Year       'assign the alias Day_of_Year to rTime(9)

  'Array to hold all the data coming from the PS200/CH200
Public CH200_M0(9)
  'Alias names for array elements.
  'Battery voltage: VDC
Alias CH200_M0(1)=VBatt
  'Current going into, or out of, the battery: Amps
```

```
Alias CH200_M0(2)=IBatt
  'Current going to the load: Amps
Alias CH200_M0(3)=ILoad
  'Voltage coming into the charger: VDC
Alias CH200_M0(4)=V_in_chg
  'Current coming into the charger: Amps
Alias CH200_M0(5)=I_in_chg
  'Charger temperature: Celsius
Alias CH200_M0(6)=Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Alias CH200_M0(7)=Chg_State
  'Charging source: None, AC, or Solar
Alias CH200_M0(8)=Chg_Source
  'Check battery error: 0=normal, 1=check battery
Alias CH200_M0(9)=Ck_Batt

  'Arrays to hold the associated words for the charge state, charge source,
  'and check battery values.
Dim ChargeStateArr(6) As String
Dim ChargeSourceArr(3) As String
Dim CheckBatteryArr(2) As String

  'Variables to hold the words for charge state, charge source, and check
  'battery.
Public ChargeState As String
Public ChargeSource As String
Public CheckBattery As String

Public CH200_MX(4)              'Array to hold extended data from the PS200/CH200
  'Alias names for array elements.
Alias CH200_MX(1) = BattTargV   'Battery charging target voltage.
Alias CH200_MX(2) = DgtlPotSet  'Digital potentiometer setting.
Alias CH200_MX(3) = BattCap     'Present battery capacity.
Alias CH200_MX(4) = Qloss       'Battery charge deficit.

  'Setting this value to true, or on, causes a manual battery test.
Public ManualTest As Boolean
  'When value is true the battery is being tested.
Public TestBatt As Boolean
  'Stores the results from the PC/CH200 query. In this case nothing is returned.
Public SDI12Result As String
  'Elapsed time of test in hours
Public ElapsedHrs As Long
  'Do not leave "TestLengthHrs" at zero, or test will stop after sending command
  'once.
Public TestLengthHrs As Long

'Define Data Tables
  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (6,CH200_M0(),FP2,0,False)
  Maximum (6,CH200_M0(),FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
```

```
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Only save this information when the TestBatt flag
  'is on.
DataTable (DischargeTest,TestBatt,144)
  DataInterval (0,10,Min,10)
  Sample (1,ElapsedHrs,FP2)
  Average (1,Chg_TmpC,FP2,False)
  Average (1,VBatt,FP2,False)
  Average (1,IBatt,FP2,False)
  Minimum (1,VBatt,FP2,False,False)
  Sample (1,Qloss,FP2)
  Sample (1,ChargeState,String)
  Sample (1,ChargeSource,String)
  Sample (1,CheckBattery,String)
EndTable

  'Main Program
BeginProg
    'Put length of battery test in hours here. Can also be modified on
    'the fly.
  TestLengthHrs = 3

    'Load arrays with words to associate with the charge state, charge
    'source and check battery values from the PS/CH200.
  ChargeStateArr(1) = "Regulator Fault"
  ChargeStateArr(2) = "No Charge"
  ChargeStateArr(3) = "Current Limited"
  ChargeStateArr(4) = "Cycle Charging"
  ChargeStateArr(5) = "Float Charging"
  ChargeStateArr(6) = "Battery Test"

  ChargeSourceArr(1) = "None"
  ChargeSourceArr(2) = "Solar"
  ChargeSourceArr(3) = "Continuous"

  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  Scan (5,Sec,0,0)
    PanelTemp (PanelTempC,250)
    CallTable Hour
  NextScan

  SlowSequence
  Scan (30,Sec,3,0)
    RealTime (rTime())
    ElapsedHrs = Timer (1,Hr,4)          'timer for elapsed time of test.
      'Run this section of code if elapsed time is less than the
      'set amount of time for the test and the battery voltage has not
      'dropped below 11.7V DC. The PS/CH200 will not let the battery drop
      'below 11.5V DC.
```

```
  If ElapsedHrs <= TestLengthHrs AND VBatt >= 11.7 Then
    'For this exercise the datalogger will automatically
    'start a test on the 17th of each month @ 8 AM OR when
    'the ManualTest flag is set to on. Have to turn on the
    'timer when the test begins.
   If ManualTest = On OR (DOM = 17 AND Hour = 8) Then
    ManualTest=Off
    TestBatt = On
    Timer (1,Hr,0)
   EndIf
    'Battery test command will be sent every 30 seconds. Must send it at
    'least every 90 seconds or the PS/CH200 will automatically reset!
   If TestBatt Then
    SDI12Recorder (SDI12Result,3,0,"XB!",1.0,0)
   EndIf
  Else
    'When the test is over reset the timer and turn of the TestBatt flag.
   Timer (1,Hr,3)
   TestBatt = Off
  EndIf
   'Get PS200/CH200 values.
  SDI12Recorder (CH200_M0(),3,0,"MC!",1.0,0)
   'Array values start with one. Values for charge state start with -1.
   'Have to shift the value by two to line it up with the correct words
   'in the array.
  ChargeState = ChargeStateArr(Chg_State + 2)
   'Values for charge source start with zero. Have to shift the value
   'by one to line it up with the correct words in the array.
  ChargeSource = ChargeSourceArr(Chg_Source + 1)
   'Values for check battery start with zero. Have to shift the value
   'by one to line it up with the correct words in the array.
  CheckBattery = CheckBatteryArr(Ck_Batt + 1)
   'Get extended battery capacity settings.
  SDI12Recorder (CH200_MX(),3,0,"M6!",1.0,0)
  CallTable (DischargeTest) 'write data before zeroing ElapsedHrs
 NextScan
EndProg
```

A.3.1.2 Set Number Of Days Battery Test – SDI12

```
'CR1000 Series Datalogger
'Program: SDI12_XDays_Battery_Load_Test.CR1
'Date: 20.October.2010
'Ver: A
'
'Notes: This program sets the battery capacity via SDI-12.
'PS200/CH200 configured with SDI-12 address 0 (zero).
'
'Use Campbell Scientific SDI-12 cable part # 20769.
'
'SDI-12 CABLE TO CR1000 WIRING
'-----------------------------
'WHITE: C3
'BLACK: G
'CLEAR: G
'
PipeLineMode
'Public Variables
  'I like to use the words "on" and "off" vs. "true" and "false".
Const On = True
Const Off = False

Public PanelTempC                    'Datalogger panel temperature: Celsius

  'Array to hold all the data coming from the PS200/CH200
Public CH200_M0(9)
  'Alias names for array elements.
  'Battery voltage: VDC
Alias CH200_M0(1)=VBatt
  'Current going into, or out of, the battery: Amps
Alias CH200_M0(2)=IBatt
  'Current going to the load: Amps
Alias CH200_M0(3)=ILoad
  'Voltage coming into the charger: VDC
Alias CH200_M0(4)=V_in_chg
  'Current coming into the charger: Amps
Alias CH200_M0(5)=I_in_chg
  'Charger temperature: Celsius
Alias CH200_M0(6)=Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Alias CH200_M0(7)=Chg_State
  'Charging source: None, AC, or Solar
Alias CH200_M0(8)=Chg_Source
  'Check battery error: 0=normal, 1=check battery
Alias CH200_M0(9)=Ck_Batt

  'Arrays to hold the associated words for the charge state, charge source,
  'and check battery values.
Dim ChargeStateArr(6) As String
Dim ChargeSourceArr(3) As String
Dim CheckBatteryArr(2) As String
```

```
  'Variables to hold the words for charge state, charge source, and check
  'battery.
Public ChargeState As String
Public ChargeSource As String
Public CheckBattery As String

Public CH200_MX(4)  'Array to hold extended data from the PS200/CH200
  'Alias names for array elements.
Alias CH200_MX(1) = BattTargV       'Battery charging target voltage.
Alias CH200_MX(2) = DgtlPotSet      'Digital potentiometer setting.
Alias CH200_MX(3) = BattCap         'Present battery capacity.
Alias CH200_MX(4) = Qloss           'Battery charge deficit.

  'Setting this value to true, or on, causes a manual battery test.
Public ManualTest As Boolean
  'When value is true the battery is being tested.
Public TestBatt As Boolean
  'Stores the results from the PC/CH200 query. In this case nothing is returned.
Public SDI12Result As String
  'Elapsed time of test in hours
Public ElapsedHrs As Long
  'Do not leave "TestLengthHrs" at zero, or test will stop after sending command
  'once.
Public TestLengthHrs As Long
  'How many days to spread out the test.
Public XDays As Long                'How many days to spread out the test.
  'Counter to hold the days the program has run since the last test. Gets set
  'to one when a test is initiated.
Dim DayCntr

  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (6,CH200_M0(),FP2,0,False)
  Maximum (6,CH200_M0(),FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Only save this information when the TestBatt flag
  'is on. Set lapses to zero so it stores a time stamp with every
  'array of data.
DataTable (DischargeTest,TestBatt,144)
  DataInterval (0,10,Min,0)
  Sample (1,ElapsedHrs,FP2)
  Average (1,Chg_TmpC,FP2,False)
  Average (1,VBatt,FP2,False)
  Average (1,IBatt,FP2,False)
  Minimum (1,VBatt,FP2,False,False)
  Sample (1,Qloss,FP2)
  Sample (1,ChargeState,String)
  Sample (1,ChargeSource,String)
  Sample (1,CheckBattery,String)
EndTable
```

```
  'Main Program
BeginProg
   'Put length of battery test in hours here. Can also be modified on
   'the fly.
  TestLengthHrs = 3
   'Run test every 90 days from the time the program is loaded.
  XDays = 90

   'Load arrays with words to associate with the charge state,
   'charge source and check battery values from the PS/CH200.
  ChargeStateArr(1) = "Regulator Fault"
  ChargeStateArr(2) = "No Charge"
  ChargeStateArr(3) = "Current Limited"
  ChargeStateArr(4) = "Cycle Charging"
  ChargeStateArr(5) = "Float Charging"
  ChargeStateArr(6) = "Battery Test"

  ChargeSourceArr(1) = "None"
  ChargeSourceArr(2) = "Solar"
  ChargeSourceArr(3) = "Continuous"

  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

   'Set the day counter to one the first time the program is run.
  DayCntr = 1
  Scan (5,Sec,0,0)
    PanelTemp (PanelTempC,250)
    CallTable Hour
     'Add one to DayCntr at midnight but only if not in a test mode.
    If TestBatt = Off AND IfTime (0,1,Day) Then DayCntr = DayCntr + 1
  NextScan

  SlowSequence
  Scan (30,Sec,3,0)
    ElapsedHrs = Timer (1,Hr,4)          'timer for elapsed time of test.
     'Run this section of code if elapsed time is less than the
     'set amount of time for the test and the battery voltage has not
     'dropped below 11.7V DC. The PS/CH200 will not let the battery drop
     'below 11.5V DC.
     'For this exercise the datalogger will automatically
     'start a test on the 90th day from the time the program was started
     'or if the ManualTest flag is set to on. Have to turn on the
     'timer when the test begins.
    If TestBatt = Off AND (ManualTest = On OR XDays >= DayCntr)  Then
      ManualTest=Off
      TestBatt = On
      DayCntr = 1
      Timer (1,Hr,0)
    EndIf
     'Battery test command will be sent every 30 seconds. Must send it at
     'least every 90 seconds or the PS/CH200 will automatically reset!
    If TestBatt= On AND ElapsedHrs <= TestLengthHrs AND VBatt >= 11.7 Then
      SDI12Recorder (SDI12Result,3,0,"XB!",1.0,0)
```

```
  Else
    'When the test is over reset the timer and turn of the TestBatt flag.
   Timer (1,Hr,3)
   TestBatt = Off
  EndIf
   'Get PS200/CH200 values.
  SDI12Recorder (CH200_M0(),3,0,"MC!",1.0,0)
   'Array values start with one. Values for charge state start with -1.
   'Have to shift the value by two to line it up with the correct words
   'in the array.
  ChargeState = ChargeStateArr(Chg_State + 2)
   'Values for charge source start with zero. Have to shift the value
   'by one to line it up with the correct words in the array.
  ChargeSource = ChargeSourceArr(Chg_Source + 1)
   'Values for check battery start with zero. Have to shift the value
   'by one to line it up with the correct words in the array.
  CheckBattery = CheckBatteryArr(Ck_Batt + 1)
   'Get extended battery capacity settings.
  SDI12Recorder (CH200_MX(),3,0,"M6!",1.0,0)
  CallTable (DischargeTest)              'write data before zeroing ElapsedHrs
 NextScan
EndProg
```

A.3.2 Enter Battery Test State RS-232 Programming Example

A.3.2.1 Calendar Date Battery Test – RS-232

```
'CR1000 Series Datalogger
'Program: RS-232_Calendar_Battery_Load_Test.CR1
'Date: 1.November.2010
'Ver: A
'
'Notes:  This program tests the existing battery for problems by
'setting the target voltage down to 11.5VDC forcing the system to run
'strictly on battery power for user settable amount of time.
'Test is based on a time interval in days and test duration in hours.
'
'Use Campbell Scientific RS-232 cable part # 25356.
'
'RS-232 CABLE TO CR1000 WIRING
'-----------------------------
'GREEN: C3 (TX)
'WHITE: C4 (RX)
'BLACK: G
'CLEAR: G
'
'This program uses the RS-232 command "BATT_TEST>" to force the battery target voltage
'down to 11.5vdc.

PipeLineMode
'Constants & Public Variables
  'Change this constant if using a different com port on the CR1000.
Const COMPRT = COM2
```

```
  'Hexadecimal equivalent of a carriage return.
Const CR = CHR(&H0D)
  'It is a comma. Used to sort PS/CH200 strings.
Const COMMA = CHR(&H2C)
  'I like to use the words "on" and "off" vs. "true" and "false".
Const On = True
Const Off = False

Public PanelTempC  'Datalogger panel temperature: Celsius

'Store the present time information.
Public rTime(9)                 'declare as public and dimension rTime to 9
Alias rTime(1) = Year           'assign the alias Year to rTime(1)
Alias rTime(2) = Month          'assign the alias Month to rTime(2)
Alias rTime(3) = DOM            'assign the alias Day to rTime(3)
Alias rTime(4) = Hour           'assign the alias Hour to rTime(4)
Alias rTime(5) = Minute         'assign the alias Minute to rTime(5)
Alias rTime(6) = Second         'assign the alias Second to rTime(6)
Alias rTime(7) = uSecond        'assign the alias uSecond to rTime(7)
Alias rTime(8) = WeekDay        'assign the alias WeekDay to rTime(8)
Alias rTime(9) = Day_of_Year    'assign the alias Day_of_Year to rTime(9)

  'Command sent to the PS/CH200 to get status information.
Const CH200STATS_CMD = "RD_STAT>"
  'Holds the check sum value of the received PS/CH200 string.
Public Check_sum
  'Used to catch clutter from the PS/CH200
Dim xmit_str As String
  'Holds the entire status string coming from the PS/CH200.
Dim CH200string As String * 140
  'Holds parsed string values.
Dim CH200_M0(9) As String * 20
  'Battery voltage: VDC
Public VBatt
  'Current going into, or out of, the battery: Amps
Public IBatt
  'Current going to the load: Amps
Public ILoad
  'Voltage coming into the charger: VDC
Public V_in_chg
  'Current coming into the charger: Amps
Public I_in_chg
  'Charger temperature: Celsius
Public Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Public ChargeState As String
  'Charging source: None, AC, or Solar
Public ChargeSource As String
  'Check battery error: 0=normal, 1=check battery
Public Ck_Batt
  'Word or phrase equivalent of check battery error.
Public CheckBattery As String
  'Array to hold the names of the battery error information.
Dim CheckBatteryArr(2) As String
```

```
  'Command to get extended status information.
Const EXTSTATS_CMD = "RD_SPECIAL>"
  'Array to hold extended data from the PS200/CH200
Public CH200_MX(4) As String * 20
  'Battery charging target voltage.
Public BattTargV
  'Digital potentiometer setting.
Public DgtlPotSet
  'Present battery capacity.
Public BattCap
  'Battery charge deficit.
Public Qloss

'Variables & constants used to write an external battery temperature to
  'the PS/CH200 and reset it.
  'Command sent to the PS/CH200 to write external battery temperature.
Const BATTEST_CMD = "BATT_TEST>"
  'String coming back from either the "WR_BT" or "RS_INTBT" command
Public ReturnStr As String * 40
  'Number signifying how far into the return string for "RS_INTBT"
  'command the "OK" is located.
Dim OK_Loc
  'Setting this value to true, or on, causes a manual battery test.
Public ManualTest As Boolean
  'When value is true the battery is being tested.
Public TestBatt As Boolean
  'Elapsed time of test in hours
Public ElapsedHrs As Long
  'Do not leave at zero, or test will stop after sending command once
  'test is initiated.
Public TestLengthHrs As Long

  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (1,VBatt,FP2,0,False)
  Minimum (1,IBatt,FP2,0,False)
  Minimum (1,ILoad,FP2,0,False)
  Minimum (1,V_in_chg,FP2,0,False)
  Minimum (1,I_in_chg,FP2,0,False)
  Minimum (1,Chg_TmpC,FP2,0,False)
  Maximum (1,VBatt,FP2,False,False)
  Maximum (1,IBatt,FP2,False,False)
  Maximum (1,ILoad,FP2,False,False)
  Maximum (1,V_in_chg,FP2,False,False)
  Maximum (1,I_in_chg,FP2,False,False)
  Maximum (1,Chg_TmpC,FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable
```

```
  'Only save this information when the TestBatt flag
  'is on.
DataTable (DischargeTest,TestBatt,144)
  DataInterval (0,1,Min,0)
  Sample (1,ElapsedHrs,FP2)
  Average (1,Chg_TmpC,FP2,False)
  Average (1,VBatt,FP2,False)
  Average (1,IBatt,FP2,False)
  Minimum (1,VBatt,FP2,False,False)
  Sample (1,BattTargV,FP2)
  Sample (1,Qloss,FP2)
  Sample (1,ChargeState,String)
  Sample (1,ChargeSource,String)
  Sample (1,CheckBattery,String)
EndTable

  'Subroutine sends two back space characters to the PS/CH200 to
  'wake it up and switch over to RS-232 mode.
Sub WAKEUP
  SerialOut (COMPRT,CHR(&H08),"",1,3)
  SerialOut (COMPRT,CHR(&H08),"",1,3)
EndSub

  'Main Program
BeginProg
    'Put length of battery test in hours here. Can also be modified on
    'the fly.
  TestLengthHrs = 5

    'Load arrays with words to associated with the check battery values
    'from the PS/CH200.
  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  Scan (5,Sec,0,0)
    PanelTemp(PanelTempC,_60Hz)       'Measure datalogger panel temperature.
    CallTable Hour
  NextScan

    'SlowSequence with RS-232 measurements
  SlowSequence
    'Configure the COM port used with the PS/CH200
  SerialOpen (COMPRT,9600,3,0,150)
  Scan(30,sec,0,0)
    RealTime (rTime())
    ElapsedHrs = Timer (1,Min,4)          'timer for elapsed time of test.

      'Run this section of code if elapsed time is less than the
      'set amount of time for the test and the battery voltage has not
      'dropped below 11.7V DC. The PS/CH200 will not let the battery drop
      'below 11.5V DC.
      'For this exercise the datalogger will automatically
      'start a test on the 17th of each month @ 8 AM OR when
      'the ManualTest flag is set to on. Have to turn on the
      'timer when the test begins.
    If TestBatt = Off Then
      If ManualTest = On OR (DOM = 17 AND Hour = 8) Then
        ManualTest=Off
        TestBatt = On
        Timer (1,Min,0)
```

```
  EndIf
EndIf

  'Battery test command will be sent every 30 seconds. Must send it
  'at least every 90 seconds or the PS/CH200 will automatically reset!
If TestBatt=On AND ElapsedHrs <= TestLengthHrs AND VBatt >= 11.7 Then
    'Call WAKEUP to wake up the PS/CH200.
  Call WAKEUP
    'Send out the PS/CH200 command followed by a carriage return.
  SerialOut (COMPRT,BATTEST_CMD,"",1,3)
  SerialOut (COMPRT,CR,"",1,3)
    'Bring in the result from either command and process it.
    'Discard characters in the receiver buffer up to and including
    'asterisks "*"
  SerialIn (xmit_str,COMPRT,50,"*",40)
    'Capture the PS/CH200 response string after the asterisks and
    'including the string terminator hexadecimal zero (&H00).
  SerialIn (CH200string,COMPRT,50,&H00,140)
    'Return strings are different for both instructions. Have to process
    'them different as well.
  OK_Loc = InStr(1,xmit_str,"O",2)
  ReturnStr = Mid (xmit_str,OK_Loc,2)
Else
    'When the test is over reset the timer and turn of the TestBatt flag.
  Timer (1,Min,3)
  TestBatt = Off
EndIf

  'Get the PS/CH200 status information.
  'Call WAKEUP to wake up the PS/CH200.
Call WAKEUP
  'Send "read status" command to PS200 followed by a carriage return.
SerialOut (COMPRT,CH200STATS_CMD,"",1,3)
SerialOut (COMPRT,CR,"",1,3)
    'Discard characters in the receiver buffer up to and including
    'asterisks "*"
SerialIn (xmit_str,COMPRT,50,"*",40)
    'Capture the PS/CH200 response string after the asterisks and
    'including the string terminator hexadecimal zero (&H00).
SerialIn (CH200string,COMPRT,50,&H00,140)
Check_sum = CheckSum (CH200string,5,0 )
  'For the string to be correct the check sum must be zero.
If Check_sum = 0 Then
    'Split up the main string from PS/CH200 into separate components.
  SplitStr (CH200_M0(),CH200string,COMMA,9,4)
    'Sort out the components to specific variables.
  VBatt = CH200_M0(1)
  IBatt = CH200_M0(2)
  ILoad = CH200_M0(3)
  V_in_chg = CH200_M0(4)
  I_in_chg = CH200_M0(5)
  Chg_TmpC = CH200_M0(6)
  ChargeState = CH200_M0(7)
  ChargeSource = CH200_M0(8)
  Ck_Batt = CH200_M0(9)
    'Values for check battery start with zero. Have to shift the value
    'by one to line it up with the correct words in the array.
  CheckBattery = CheckBatteryArr(Ck_Batt + 1)
EndIf
```

```
    'Get the PS/CH200 extended status information.
    'Call WAKEUP to wake up the PS/CH200.
   Call WAKEUP
    'Send "read extended status" command to PS/CH200 followed by a
    'carriage return.
   SerialOut (COMPRT,EXTSTATS_CMD,"",1,3)
   SerialOut (COMPRT,CR,"",1,3)
     'Discard characters in the receiver buffer up to and including
     'asterisks "*"
   SerialIn (xmit_str,COMPRT,50,"*",40)
     'Capture the PS/CH200 response string after the asterisks and
     'including the string terminator hexadecimal zero (&H00).
   SerialIn (CH200string,COMPRT,50,&H00,140)
   Check_sum = CheckSum (CH200string,5,0 )
    'For the string to be correct the check sum must be zero.
   If Check_sum = 0 Then
      'Split up the main string from PS/CH200 into separate components.
    SplitStr (CH200_MX(),CH200string,COMMA,4,4)
      'Sort out the components to specific variables.
    BattTargV = CH200_MX(1)
    DgtlPotSet = CH200_MX(2)
    BattCap = CH200_MX(3)
    Qloss = CH200_MX(4)
   EndIf
   CallTable (DischargeTest)              'write data before zeroing ElapsedHrs
  NextScan
EndProg
```

A.3.2.2 Set Number Of Days Battery Test – SDI12

```
'CR1000 Series Datalogger
'Program: RS-232_XDays_Battery_Load_Test.CR1
'Date: 1.November.2010
'Ver: A
'
'Notes:  This program tests the existing battery for problems by
'setting the target voltage down to 11.5VDC forcing the system to
'run strictly on battery power.
'Test is based on a time interval in days and test duration is in
'hours.
'
'Use Campbell Scientific RS-232 cable part # 25356.
'
'RS-232 CABLE TO CR1000 WIRING
'-----------------------------
'GREEN: C3 (TX)
'WHITE: C4 (RX)
'BLACK: G
'CLEAR: G
'
'This program uses the RS-232 command "BATT_TEST>" to force the
'battery target voltage down to 11.5vdc.

PipeLineMode
  'Constants & Public Variables
  'Change this constant if using a different com port on the CR1000.
Const COMPRT = COM2
  'Hexadecimal equivalent of a carriage return.
Const CR = CHR(&H0D)
  'It is a comma. Sort the string from the PS/CH200 using commas.
Const COMMA = CHR(&H2C)
  'I like to use the words "on" and "off" vs. "true" and "false".
Const On = True
Const Off = False

Public PanelTempC  'Datalogger panel temperature: Celsius

  'Variables and constants used to get status information from the PS/CH200
  'Command sent to the PS/CH200 to get status information.
Const CH200STATS_CMD = "RD_STAT>"
  'Holds the check sum value of the received PS/CH200 string.
Public Check_sum
  'Used to catch clutter from the PS/CH200
Dim xmit_str As String
  'Holds the entire status string coming from the PS/CH200.
Dim CH200string As String * 140
  'Holds parsed string values.
Dim CH200_M0(9) As String * 20
  'Battery voltage: VDC
Public VBatt
  'Current going into, or out of, the battery: Amps
Public IBatt
  'Current going to the load: Amps
```

```
Public ILoad
  'Voltage coming into the charger: VDC
Public V_in_chg
  'Current coming into the charger: Amps
Public I_in_chg
  'Charger temperature: Celsius
Public Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Public ChargeState As String
  'Charging source: None, AC, or Solar
Public ChargeSource As String
  'Check battery error: 0=normal, 1=check battery
Public Ck_Batt
  'Word or phrase equivalent of check battery error.
Public CheckBattery As String

  'Array to hold the names of the battery error information.
Dim CheckBatteryArr(2) As String

  'Command to get extended status information.
Const EXTSTATS_CMD = "RD_SPECIAL>"
  'Array to hold extended data from the PS200/CH200
Public CH200_MX(4) As String * 20
  'Battery charging target voltage.
Public BattTargV
  'Digital potentiometer setting.
Public DgtlPotSet
  'Present battery capacity.
Public BattCap
  'Battery charge deficit.
Public Qloss

'Variables & constants used to write an external battery temperature
  'to the PS/CH200 and reset it.
  'Command sent to the PS/CH200 to write external battery temperature.
Const BATTEST_CMD = "BATT_TEST>"
  'String coming back from either the "WR_BT" or "RS_INTBT" command
Public ReturnStr As String * 40
  'Number signifying how far into the return string for "RS_INTBT"
  'command the "OK" is located.
Dim OK_Loc
  'Setting this value to true, or on, causes a manual battery test.
Public ManualTest As Boolean
  'When value is true the battery is being tested.
Public TestBatt As Boolean
  'Elapsed time of test in hours
Public ElapsedHrs As Long
  'Do not leave "TestLengthHrs" at zero, or test will stop after
  'sending command once.
Public TestLengthHrs As Long
  'How many days to spread out the test.
Public XDays As Long
  'Counter to hold the days the program has run since the last
  'test. Gets zeroed out when a test is initiated.
Public DayCntr

  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (1,VBatt,FP2,0,False)
```

```
  Minimum (1,IBatt,FP2,0,False)
  Minimum (1,ILoad,FP2,0,False)
  Minimum (1,V_in_chg,FP2,0,False)
  Minimum (1,I_in_chg,FP2,0,False)
  Minimum (1,Chg_TmpC,FP2,0,False)
  Maximum (1,VBatt,FP2,False,False)
  Maximum (1,IBatt,FP2,False,False)
  Maximum (1,ILoad,FP2,False,False)
  Maximum (1,V_in_chg,FP2,False,False)
  Maximum (1,I_in_chg,FP2,False,False)
  Maximum (1,Chg_TmpC,FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Only save this information when the TestBatt flag
  'is on.
DataTable (DischargeTest,TestBatt,800)
  DataInterval (0,1,Min,0)
  Sample (1,ElapsedHrs,FP2)
  Average (1,Chg_TmpC,FP2,False)
  Average (1,VBatt,FP2,False)
  Average (1,IBatt,FP2,False)
  Minimum (1,VBatt,FP2,False,False)
  Sample (1,BattTargV,FP2)
  Sample (1,Qloss,FP2)
  Sample (1,ChargeState,String)
  Sample (1,ChargeSource,String)
  Sample (1,CheckBattery,String)
EndTable

  'Subroutine sends two back space characters to the PS/CH200 to
  'wake it up and switch over to RS-232 mode.
Sub WAKEUP
  SerialOut (COMPRT,CHR(&H08),"",1,3)
  SerialOut (COMPRT,CHR(&H08),"",1,3)
EndSub

  'Main Program
BeginProg
    'Put length of battery test in hours here. Can also be modified on
    'the fly.
  TestLengthHrs = 5
  XDays = 90                          'Run test every 90 days from the time the program is loaded.
  DayCntr = 1                         'set the day counter to one the first time the program is run.

    'Load arrays with words to associated with the check battery values
    'from the PS/CH200.
  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  Scan (5,Sec,0,0)
    PanelTemp(PanelTempC,_60Hz)       'Measure datalogger panel temperature.
      'Add one to DayCntr every midnight.
    If TestBatt = Off AND IfTime (0,1,Day) Then DayCntr = DayCntr + 1
    CallTable Hour
  NextScan

    'SlowSequence with RS-232 measurements
  SlowSequence
```

```
  'Configure the COM port used with the PS/CH200
SerialOpen (COMPRT,9600,3,0,150)
Scan(30,sec,0,0)
  ElapsedHrs = Timer (1,Hr,4)            'timer for elapsed time of test.

    'Run this section of code if elapsed time is less than the
    'set amount of time for the test and the battery voltage has not
    'dropped below 11.7V DC. The PS/CH200 will not let the battery drop
    'below 11.5V DC.

  If TestBatt = Off AND (ManualTest = On OR XDays <= DayCntr)  Then
    ManualTest=Off
    TestBatt = On
    DayCntr = 0
    Timer (1,Hr,0)
  EndIf

    'Battery test command will be sent every 30 seconds. Must send it
    'at least every 90 seconds or the PS/CH200 will automatically reset!
  If TestBatt=On AND ElapsedHrs <= TestLengthHrs AND VBatt >= 11.7 Then
     'Call WAKEUP to wake up the PS/CH200.
    Call WAKEUP
     'Send out the PS/CH200 command followed by a carriage return.
    SerialOut (COMPRT,BATTEST_CMD,"",1,3)
    SerialOut (COMPRT,CR,"",1,3)
     'Bring in the result from either command and process it.
     'Discard characters in the receiver buffer up to and including asterisks "*"
    SerialIn (xmit_str,COMPRT,50,"*",40)
     'Capture the PS/CH200 response string after the asterisks and including the
     'string terminator hexadecimal zero (&H00).
    SerialIn (CH200string,COMPRT,50,&H00,140)
     'Return strings are different for both instructions. Have to process them
     'different as well.
    OK_Loc = InStr(1,xmit_str,"O",2)
    ReturnStr = Mid (xmit_str,OK_Loc,2)
  Else
     'When the test is over reset the timer and turn of the TestBatt flag.
    Timer (1,Hr,3)
    TestBatt = Off
  EndIf
```

```
    'Get the PS/CH200 status information.
    'Call WAKEUP to wake up the PS/CH200.
   Call WAKEUP
    'Send "read status" command to PS200 followed by a carriage return.
   SerialOut (COMPRT,CH200STATS_CMD,"",1,3)
   SerialOut (COMPRT,CR,"",1,3)
    'Discard characters in the receiver buffer up to and including asterisks "*"
   SerialIn (xmit_str,COMPRT,50,"*",40)
    'Capture the PS/CH200 response string after the asterisks and including the
    'string terminator hexadecimal zero (&H00).
   SerialIn (CH200string,COMPRT,50,&H00,140)
   Check_sum = CheckSum (CH200string,5,0 )
    'For the string to be correct the check sum must be zero.
   If Check_sum = 0 Then
      'Split up the main string from PS/CH200 into separate components.
    SplitStr (CH200_M0(),CH200string,COMMA,9,4)
      'Sort out the components to specific variables.
    VBatt = CH200_M0(1)
    IBatt = CH200_M0(2)
    ILoad = CH200_M0(3)
    V_in_chg = CH200_M0(4)
    I_in_chg = CH200_M0(5)
    Chg_TmpC = CH200_M0(6)
    ChargeState = CH200_M0(7)
    ChargeSource = CH200_M0(8)
    Ck_Batt = CH200_M0(9)
      'Values for check battery start with zero. Have to shift the value
      'by one to line it up with the correct words in the array.
    CheckBattery = CheckBatteryArr(Ck_Batt + 1)
   EndIf

    'Get the PS/CH200 extended status information.
    'Call WAKEUP to wake up the PS/CH200.
   Call WAKEUP
    'Send "read extended status" command to PS/CH200 followed by a carriage return.
   SerialOut (COMPRT,EXTSTATS_CMD,"",1,3)
   SerialOut (COMPRT,CR,"",1,3)
    'Discard characters in the receiver buffer up to and including asterisks "*"
   SerialIn (xmit_str,COMPRT,50,"*",40)
    'Capture the PS/CH200 response string after the asterisks and including the
    'string terminator hexadecimal zero (&H00).
   SerialIn (CH200string,COMPRT,50,&H00,140)
   Check_sum = CheckSum (CH200string,5,0 )
    'For the string to be correct the check sum must be zero.
   If Check_sum = 0 Then
      'Split up the main string from PS/CH200 into separate components.
    SplitStr (CH200_MX(),CH200string,COMMA,4,4)
      'Sort out the components to specific variables.
    BattTargV = CH200_MX(1)
    DgtlPotSet = CH200_MX(2)
    BattCap = CH200_MX(3)
    Qloss = CH200_MX(4)
   EndIf
   CallTable (DischargeTest) 'write data before zeroing ElapsedHrs
  NextScan
EndProg
```

A.4 Zero QLoss

Qloss is used to track current in amp-hours going in and out of the battery. It is a handy way of monitoring the conditions of the entire system if using a solar panel. It is not the best way to calculate a power budget – see Appendix A.5 for an example of measuring current used by the entire system across a specified time.

The value for Qloss increases as the battery discharges and decreases as the battery is charged. The value will never drop below zero. Depending on load and charging characteristics this value may never approach zero. Also, if a battery is replaced it is handy to zero out Qloss if it is being used.

NOTE An added bonus with this section is it uses the instructions to get extended status information from the PS/CH200.

A.4.1 Zero Out QLoss SDI12 Programming Example

```
'CR1000 Series Datalogger
'Program: SDI12_Clear_QLoss.CR1
'Date: 7.October.2010
'Ver: A
'
'Notes: This program sets QLoss back to zero using the SDI-12
'command XRQ! to set QLoss to zero.
'PS200/CH200 configured with SDI-12 address 0 (zero).
'
'Use Campbell Scientific SDI-12 cable part # 20769.
'
'SDI-12 CABLE TO CR1000 WIRING
'-----------------------------
'WHITE: C3
'BLACK: G
'CLEAR: G
'
  'Setting the variable Clear_QLoss = True will clear QLoss.
Public Clear_QLoss As Boolean

  'Array to hold data coming from the PS200/CH200
Public CH200_M0(6)
  'Alias names for array elements.
Alias CH200_M0(1)=VBatt          'Battery voltage: VDC
Alias CH200_M0(2)=IBatt          'Current going into, or out of, the battery: Amps
Alias CH200_M0(3)=ILoad          'Current going to the load: Amps
Alias CH200_M0(4)=V_in_chg       'Voltage coming into the charger: VDC
Alias CH200_M0(5)=I_in_chg       'Current coming into the charger: Amps
Alias CH200_M0(6)=Chg_TmpC       'Charger temperature: Celsius

Public CH200_MX(4)               'Array to hold extended data from the PS200/CH200
  'Alias names for array elements.
Alias CH200_MX(1) = BattTargV    'Battery charging target voltage.
Alias CH200_MX(2) = DgtlPotSet   'Digital potentiometer setting.
Alias CH200_MX(3) = BattCap      'Present battery capacity.
Alias CH200_MX(4) = Qloss        'Battery charge deficit.

Public PanelTempC                'Datalogger panel temperature: Celsius
Dim SDI12Result As String        'Have to have a variable for the SDI12
                                 'instruction but nothing will be written to it, so hide it.

'Define Data Tables
```

```
  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (6,CH200_M0(),FP2,0,False)
  Maximum (6,CH200_M0(),FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Save the old QLoss value and a time when the value was cleared.
DataTable (ClearQLoss,True,48)
  Sample (1,Qloss,FP2)
EndTable

BeginProg
  Clear_QLoss = False
  Scan (5,Sec,0,0)
    PanelTemp (PanelTempC,250)
    CallTable Hour
  NextScan

    'Put SDI-12 measurements in a SlowSequence to avoid any missed main
    'scans if there are any sensor problems
  SlowSequence
  Scan (30,Sec,0,0)
     'Get PS200/CH200 values.
    SDI12Recorder(CH200_M0(),3,0,"M3!",1.0,0)
     'Get present battery capacity settings.
    SDI12Recorder (CH200_MX(),3,0,"M6!",1.0,0)
     'If Clear_QLoss is set to true then set QLoss in the CH200 to zero.
    If Clear_QLoss = True Then
      CallTable ClearQLoss
      Clear_QLoss = False
      SDI12Recorder (SDI12Result,3,0,"XRQ!",1.0,0)
    EndIf
  NextScan
EndProg
```

A.4.2 Zero Out QLoss RS-232 Programming Example

```
'CR1000 Series Datalogger
'Program: RS-232_Clear_QLoss.CR1
'Date: 8.Nov.2010
'Ver: A
'
'Notes: This program sets QLoss back to zero using the RS-232
'command "RESET_QLOSS>" to set QLoss to zero.
'
'Use Campbell Scientific RS-232 cable part # 25356.
'
'RS-232 CABLE TO CR1000 WIRING
'-----------------------------
'GREEN: C3 (TX)
'WHITE: C4 (RX)
'BLACK: G
'CLEAR: G
'
PipeLineMode
  'Constants & Public Variables
  'Change this constant if using a different com port on the CR1000.
Const COMPRT = COM2
  'Hexadecimal equivalent of a carriage return.
Const CR = CHR(&H0D)
  'It is a comma. Sort the string from the PS/CH200 using commas.
Const COMMA = CHR(&H2C)
  'I like to use the words "on" and "off" vs. "true" and "false".
Const On = True
Const Off = False

Public PanelTempC  'Datalogger panel temperature: Celsius

'Variables and constants used to get status information from the PS/CH200
  'Command sent to the PS/CH200 to get status information.
Const CH200STATS_CMD = "RD_STAT>"
  'Holds the check sum value of the received PS/CH200 string.
Public Check_sum
  'Used to catch clutter from the PS/CH200
Dim xmit_str As String
  'Holds the entire status string coming from the PS/CH200.
Dim CH200string As String * 140
  'Holds parsed string values.
Dim CH200_M0(9) As String * 20
  'Battery voltage: VDC
Public VBatt
  'Current going into, or out of, the battery: Amps
Public IBatt
  'Current going to the load: Amps
Public ILoad
  'Voltage coming into the charger: VDC
Public V_in_chg
  'Current coming into the charger: Amps
Public I_in_chg
  'Charger temperature: Celsius
```

```
Public Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Public ChargeState As String
  'Charging source: None, AC, or Solar
Public ChargeSource As String
  'Check battery error: 0=normal, 1=check battery
Public Ck_Batt
  'Word or phrase equivalent of check battery error.
Public CheckBattery As String

  'Array to hold the names of the battery error information.
Dim CheckBatteryArr(2) As String

  'Command to get extended status information.
Const EXTSTATS_CMD = "RD_SPECIAL>"
  'Array to hold extended data from the PS200/CH200
Public CH200_MX(4) As String * 20
  'Battery charging target voltage.
Public BattTargV
  'Digital potentiometer setting.
Public DgtlPotSet
  'Present battery capacity.
Public BattCap
  'Battery charge deficit.
Public Qloss
'
  'Command to clear out QLoss.
Const QLOSS_RES_CMD = "RESET_QLOSS>"
  'Setting the variable Clear_QLoss to true, or on, clears QLoss.
Public Clear_QLoss As Boolean
  'Number signifying how far into the return string for "RESET_QLOSS"
  'command the "OK" is located.
Public QL_OK_Loc
  'Return string containing the "OK".
Public QL_ReturnStr As String * 40

  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (1,VBatt,FP2,0,False)
  Minimum (1,IBatt,FP2,0,False)
  Minimum (1,ILoad,FP2,0,False)
  Minimum (1,V_in_chg,FP2,0,False)
  Minimum (1,I_in_chg,FP2,0,False)
  Minimum (1,Chg_TmpC,FP2,0,False)
  Maximum (1,VBatt,FP2,False,False)
  Maximum (1,IBatt,FP2,False,False)
  Maximum (1,ILoad,FP2,False,False)
  Maximum (1,V_in_chg,FP2,False,False)
  Maximum (1,I_in_chg,FP2,False,False)
  Maximum (1,Chg_TmpC,FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Save the old QLoss value and a time when the value was cleared.
DataTable (ClearQLoss,True,48)
  Sample (1,Qloss,FP2)
EndTable
```

```
'Subroutine sends two back space characters to the PS/CH200 to
  'wake it up and switch over to RS-232 mode.
Sub WAKEUP
  SerialOut (COMPRT,CHR(&H08),"",1,3)
  SerialOut (COMPRT,CHR(&H08),"",1,3)
EndSub

  'Main Program
BeginProg
  Clear_QLoss = False

  'Load arrays with words to associated with the check battery values
    'from the PS/CH200.
  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  Scan (5,Sec,0,0)
    PanelTemp(PanelTempC,_60Hz)       'Measure datalogger panel temperature.
    CallTable Hour
  NextScan

    'SlowSequence with RS-232 measurements
  SlowSequence
    'Configure the COM port used with the PS/CH200
  SerialOpen (COMPRT,9600,3,0,150)
  Scan(30,sec,0,0)

    'Get the PS/CH200 status information.
      'Call WAKEUP to wake up the PS/CH200.
    Call WAKEUP
      'Send "read status" command to PS200 followed by a carriage return.
    SerialOut (COMPRT,CH200STATS_CMD,"",1,3)
    SerialOut (COMPRT,CR,"",1,3)
      'Discard characters in the receiver buffer up to and including asterisks "*"
    SerialIn (xmit_str,COMPRT,50,"*",40)
      'Capture the PS/CH200 response string after the asterisks and including the
      'string terminator hexadecimal zero (&H00).
    SerialIn (CH200string,COMPRT,50,&H00,140)
    Check_sum = CheckSum (CH200string,5,0 )
      'For the string to be correct the check sum must be zero.
    If Check_sum = 0 Then
       'Split up the main string from PS/CH200 into separate components.
      SplitStr (CH200_M0(),CH200string,COMMA,9,4)
       'Sort out the components to specific variables.
      VBatt = CH200_M0(1)
      IBatt = CH200_M0(2)
      ILoad = CH200_M0(3)
      V_in_chg = CH200_M0(4)
      I_in_chg = CH200_M0(5)
      Chg_TmpC = CH200_M0(6)
      ChargeState = CH200_M0(7)
      ChargeSource = CH200_M0(8)
      Ck_Batt = CH200_M0(9)
        'Values for check battery start with zero. Have to shift the value
        'by one to line it up with the correct words in the array.
      CheckBattery = CheckBatteryArr(Ck_Batt + 1)
    EndIf

    'Clear out QLoss
    'Call WAKEUP to wake up the PS/CH200.
```

```
    'If Clear_QLoss is set to true then set QLoss in the CH200 to zero.
  If Clear_QLoss = True Then
    CallTable ClearQLoss
    Clear_QLoss = False
    Call WAKEUP
      'Send "read status" command to PS200 followed by a carriage return.
    SerialOut (COMPRT,QLOSS_RES_CMD,"",1,3)
    SerialOut (COMPRT,CR,"",1,3)
      'Discard characters in the receiver buffer up to and including asterisks "*"
    SerialIn (xmit_str,COMPRT,50,"*",40)
      'Capture the PS/CH200 response string after the asterisks and including the
      'string terminator hexadecimal zero (&H00).
    SerialIn (CH200string,COMPRT,50,&H00,140)
      'Return strings are different for both instructions. Have to process them
      'different as well.
    QL_OK_Loc = InStr(1,xmit_str,"K",2)
    QL_ReturnStr = Mid (xmit_str,QL_OK_Loc-1,2)
  EndIf

  'Get the PS/CH200 extended status information.
    'Call WAKEUP to wake up the PS/CH200.
  Call WAKEUP
    'Send "read extended status" command to PS/CH200 followed by a carriage return.
  SerialOut (COMPRT,EXTSTATS_CMD,"",1,3)
  SerialOut (COMPRT,CR,"",1,3)
    'Discard characters in the receiver buffer up to and including asterisks "*"
  SerialIn (xmit_str,COMPRT,50,"*",40)
    'Capture the PS/CH200 response string after the asterisks and including the
    'string terminator hexadecimal zero (&H00).
  SerialIn (CH200string,COMPRT,50,&H00,140)
  Check_sum = CheckSum (CH200string,5,0 )
    'For the string to be correct the check sum must be zero.
  If Check_sum = 0 Then
      'Split up the main string from PS/CH200 into separate components.
    SplitStr (CH200_MX(),CH200string,COMMA,4,4)
      'Sort out the components to specific variables.
    BattTargV = CH200_MX(1)
    DgtlPotSet = CH200_MX(2)
    BattCap = CH200_MX(3)
    Qloss = CH200_MX(4)
  EndIf
 NextScan
EndProg
```

A.5 Station Power Usage

Station power usage can be a very serious question if batteries at a site are frequently dying or if the system is known to use a lot of power and the mix of battery and solar panel size are still being ascertained.

Having the station calculating a power budget on an hourly basis will help to identify when higher than normal load conditions occur and separate out whether the load is due to the battery or the datalogger and sensors/communication options connected to it.

A.5.1 Station Power Usage SDI12 Programming Example

```
'CR1000 Series Datalogger
'Program: SDI12_Amp-Hr_Test.CR1
'Programmer: Bart Nef - CSI
'Date: 12.November.2010
'Ver: A
'
'Notes: This program measures and records current usage by the station
'in amp-hours. Data is stored for hourly and daily values as well as
'PS/CH200 status variables.
'In order to calculate amp-hour capacity the measured currents coming
'from the PS/CH200 must be modified to include the scan rate and then
'totalled over the data table interval.
'PS200/CH100 configured with SDI-12 address 0 (zero).
'Use CSI cable part # 20769.  The black and clear wires connect to the CR1000
'ground, and the white wire connects to the CR1000 control port C3.
'
 'Public Variables
PipeLineMode
 'Constants & Public Variables
Const SlowScan = 30                    'scan rate used with the slow scan section.
 'I like to use the words "On" and "Off" vs. True and False.
Const On = True
Const Off = False
Public PanelTempC                      'Datalogger panel temperature: Celsius
Public Batt_Volt
  'Array holding all the three Amp-hour calculations
Public Amp_Hour(3)
  'Amp-hours used by the entire system.
Alias Amp_Hour(1) = Sys_Amp_hr
  'Amp-hours used by the datalogger, sensors and any communication options.
Alias Amp_Hour(2) = Load_Amp_hr
  'Amp-hours going into the battery. Current coming out of the battery is
  'part of the Load_Amp_hr value.
Alias Amp_Hour(3) = Batt_Amp_hr
Dim n                                  'Used as a counter.
```

```
Public CH200_M0(9)                  'Array to hold all the data coming from the PS200/CH200
 'Alias names for array elements.
Alias CH200_M0(1)=VBatt             'Battery voltage: VDC
Alias CH200_M0(2)=IBatt             'Current going into, or out of, the battery: Amps
Alias CH200_M0(3)=ILoad             'Current going to the load: Amps
Alias CH200_M0(4)=V_in_chg          'Voltage coming into the charger: VDC
Alias CH200_M0(5)=I_in_chg          'Current coming into the charger: Amps
Alias CH200_M0(6)=Temp_C            'Charger temperature: Celsius
Alias CH200_M0(7)=Chg_State         'Charging state: Cycle, Float, Current Limited, or None
Alias CH200_M0(8)=Chg_Source        'Charging source: None, AC, or Solar
Alias CH200_M0(9)=Ck_Batt           'Check battery error: 0=normal, 1=check battery

Dim ChargeStateArr(6) As String
Dim ChargeSourceArr(3) As String
Dim CheckBatteryArr(2) As String

Public ChargeState As String
Public ChargeSource As String
Public CheckBattery As String

Public CH200_MX(4)                  'Array to hold extended data from the PS200/CH200
 'Alias names for array elements.
Alias CH200_MX(1) = BattTargV       'Battery charging target voltage.
Alias CH200_MX(2) = DgtlPotSet      'Digital potentiometer setting.
Alias CH200_MX(3) = BattCap         'Present battery capacity.
Alias CH200_MX(4) = Qloss           'Battery charge deficit.

'Define Data Tables
 'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (6,CH200_M0(),FP2,0,False)
  Maximum (6,CH200_M0(),FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Stored hourly Amp-hour values and present Qloss value.
DataTable (Amp_hr,True,-1)
  DataInterval (0,1,Hr,10)
  Totalize (3,Amp_Hour(),FP2,False)
  Sample (1,Qloss,FP2)
EndTable

  'Stored daily Amp-hour values and present Qloss value.
DataTable (Amp_Day,True,-1)
  DataInterval (0,1,Day,10)
  Totalize (3,Amp_Hour(),FP2,False)
  Sample (1,Qloss,FP2)
EndTable
```

```
 'Main Program
BeginProg
  ChargeStateArr(1) = "Regulator Fault"
  ChargeStateArr(2) = "No Charge"
  ChargeStateArr(3) = "Current Limited"
  ChargeStateArr(4) = "Cycle Charging"
  ChargeStateArr(5) = "Float Charging"
  ChargeStateArr(6) = "Battery Test"

  ChargeSourceArr(1) = "None"
  ChargeSourceArr(2) = "Solar"
  ChargeSourceArr(3) = "Continuous"

  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  Scan (5,Sec,0,0)
    PanelTemp (PanelTempC,250)
    Battery (Batt_Volt)
    CallTable Hour
  NextScan

  SlowSequence
  Scan (SlowScan,Sec,3,0)
     'Get extended status values.
    SDI12Recorder (CH200_MX(),3,0,"M6!",1.0,0)
     'Get PS200/CH200 values.
    SDI12Recorder (CH200_M0(),3,0,"MC!",1.0,0)
     'Values for the following parameters start at -1 for the charge state and 0 for
     'the charge source and check battery. Array locations start with one. Have to
     'shift the returned values so they start with one.
    ChargeState = ChargeStateArr(Chg_State + 2)
    ChargeSource = ChargeSourceArr(Chg_Source + 1)
    CheckBattery = CheckBatteryArr(Ck_Batt + 1)
     'Add up the current being consumed by the system. Include battery current in
     'the calculation ONLY when it is positive - going into the battery.
     'Current values coming from the PS/CH200 are in amps. Have to convert that to
     'amp-hours based on the slow scan rate.
    Sys_Amp_hr = Sys_Amp_hr + ILoad
    If IBatt > 0 Then
      Sys_Amp_hr = Sys_Amp_hr + IBatt
      Batt_Amp_hr = IBatt
    EndIf
    Load_Amp_hr = ILoad

    For n = 1 To 3
      Amp_Hour(n) = Amp_Hour(n) * SlowScan/3600
    Next n
     'Must call these tables from the slow scan or the values will be wrong!
    CallTable Amp_hr
    CallTable Amp_Day
  NextScan
EndProg
```

A.5.2 Station Power Usage RS-232 Programming Example

```
'CR1000 Series Datalogger
'Program: RS-232__Amp-Hr_Test.CR1
'Date: 8.Nov.2010
'Ver: A
'
'Notes: This program measures and records current usage by the station
'in amp-hours. Data is stored for hourly and daily values as well as
'PS/CH200 status variables.
'In order to calculate amp-hour capacity the measured currents coming
'from the PS/CH200 must be modified to include the scan rate and then
'totalled over the data table interval.
'Use Campbell Scientific RS-232 cable part # 25356.
'
'RS-232 CABLE TO CR1000 WIRING
'----------------------------
'GREEN: C3 (TX)
'WHITE: C4 (RX)
'BLACK: G
'CLEAR: G
'
  'Public Variables
PipeLineMode
  'Constants & Public Variables
Const SlowScan = 30  'scan rate used with the slow scan section.
  'Change this constant if using a different com port on the CR1000.
Const COMPRT = COM2
  'Hexadecimal equivalent of a carriage return.
Const CR = CHR(&H0D)
  'It is a comma. Sort the string from the PS/CH200 using commas.
Const COMMA = CHR(&H2C)
  'I like to use the words "on" and "off" vs. "true" and "false".
Const On = True
Const Off = False

Public PanelTempC  'Datalogger panel temperature: Celsius
  'Array holding all the three Amp-hour calculations
Public Amp_Hour(3)
  'Amp-hours used by the entire system.
Alias Amp_Hour(1) = Sys_Amp_hr
  'Amp-hours used by the datalogger, sensors and any communication options.
Alias Amp_Hour(2) = Load_Amp_hr
  'Amp-hours going into the battery. Current coming out of the battery is
  'part of the Load_Amp_hr value.
Alias Amp_Hour(3) = Batt_Amp_hr
Dim n  'Used as a counter.

'Variables and constants used to get status information from the PS/CH200
  'Command sent to the PS/CH200 to get status information.
Const CH200STATS_CMD = "RD_STAT>"
  'Holds the check sum value of the received PS/CH200 string.
Public Check_sum
  'Used to catch clutter from the PS/CH200
Dim xmit_str As String
  'Holds the entire status string coming from the PS/CH200.
Dim CH200string As String * 140
  'Holds parsed string values.
Dim CH200_M0(9) As String * 20
  'Battery voltage: VDC
```

```
Public VBatt
  'Current going into, or out of, the battery: Amps
Public IBatt
  'Current going to the load: Amps
Public ILoad
  'Voltage coming into the charger: VDC
Public V_in_chg
  'Current coming into the charger: Amps
Public I_in_chg
  'Charger temperature: Celsius
Public Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Public ChargeState As String
  'Charging source: None, AC, or Solar
Public ChargeSource As String
  'Check battery error: 0=normal, 1=check battery
Public Ck_Batt
  'Word or phrase equivalent of check battery error.
Public CheckBattery As String

  'Array to hold the names of the battery error information.
Dim CheckBatteryArr(2) As String

  'Command to get extended status information.
Const EXTSTATS_CMD = "RD_SPECIAL>"
  'Array to hold extended data from the PS200/CH200
Public CH200_MX(4) As String * 20
  'Battery charging target voltage.
Public BattTargV
  'Digital potentiometer setting.
Public DgtlPotSet
  'Present battery capacity.
Public BattCap
  'Battery charge deficit.
Public Qloss

  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (1,VBatt,FP2,0,False)
  Minimum (1,IBatt,FP2,0,False)
  Minimum (1,ILoad,FP2,0,False)
  Minimum (1,V_in_chg,FP2,0,False)
  Minimum (1,I_in_chg,FP2,0,False)
  Minimum (1,Chg_TmpC,FP2,0,False)
  Maximum (1,VBatt,FP2,False,False)
  Maximum (1,IBatt,FP2,False,False)
  Maximum (1,ILoad,FP2,False,False)
  Maximum (1,V_in_chg,FP2,False,False)
  Maximum (1,I_in_chg,FP2,False,False)
  Maximum (1,Chg_TmpC,FP2,False,False)
```

```
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

DataTable (Amp_hr,True,-1)
  DataInterval (0,1,Hr,10)
  Totalize (3,Amp_Hour(),FP2,False)
  Sample (1,Qloss,FP2)
EndTable

DataTable (Amp_Day,True,-1)
  DataInterval (0,1,Day,10)
  Totalize (3,Amp_Hour(),FP2,False)
  Sample (1,Qloss,FP2)
EndTable

'Subroutine sends two back space characters to the PS/CH200 to
 'wake it up and switch over to RS-232 mode.
Sub WAKEUP
  SerialOut (COMPRT,CHR(&H08),"",1,3)
  SerialOut (COMPRT,CHR(&H08),"",1,3)
EndSub

  'Main Program
BeginProg
  'Load arrays with words to associated with the check battery values
    'from the PS/CH200.
  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  Scan (5,Sec,0,0)
    PanelTemp(PanelTempC,_60Hz)       'Measure datalogger panel temperature.
    CallTable Hour
  NextScan

    'SlowSequence with RS-232 measurements
  SlowSequence
    'Configure the COM port used with the PS/CH200
  SerialOpen (COMPRT,9600,3,0,150)
  Scan(SlowScan,sec,0,0)
    'Get the PS/CH200 status information.
    'Call WAKEUP to wake up the PS/CH200.
    Call WAKEUP
    'Send "read status" command to PS200 followed by a carriage return.
    SerialOut (COMPRT,CH200STATS_CMD,"",1,3)
    SerialOut (COMPRT,CR,"",1,3)
    'Discard characters in the receiver buffer up to and including asterisks "*"
    SerialIn (xmit_str,COMPRT,50,"*",40)
    'Capture the PS/CH200 response string after the asterisks and including the
    'string terminator hexadecimal zero (&H00).
    SerialIn (CH200string,COMPRT,50,&H00,140)
    Check_sum = CheckSum (CH200string,5,0 )
    'For the string to be correct the check sum must be zero.
```

```
If Check_sum = 0 Then
  'Split up the main string from PS/CH200 into separate components.
  SplitStr (CH200_M0(),CH200string,COMMA,9,4)
  'Sort out the components to specific variables.
  VBatt = CH200_M0(1)
  IBatt = CH200_M0(2)
  ILoad = CH200_M0(3)
  V_in_chg = CH200_M0(4)
  I_in_chg = CH200_M0(5)
  Chg_TmpC = CH200_M0(6)
  ChargeState = CH200_M0(7)
  ChargeSource = CH200_M0(8)
  Ck_Batt = CH200_M0(9)
  'Values for check battery start with zero. Have to shift the value
  'by one to line it up with the correct words in the array.
  CheckBattery = CheckBatteryArr(Ck_Batt + 1)
EndIf

'Get the PS/CH200 extended status information.
'Call WAKEUP to wake up the PS/CH200.
Call WAKEUP
'Send "read extended status" command to PS/CH200 followed by a carriage return.
SerialOut (COMPRT,EXTSTATS_CMD,"",1,3)
SerialOut (COMPRT,CR,"",1,3)
'Discard characters in the receiver buffer up to and including asterisks "*"
SerialIn (xmit_str,COMPRT,50,"*",40)
'Capture the PS/CH200 response string after the asterisks and including the
'string terminator hexadecimal zero (&H00).
SerialIn (CH200string,COMPRT,50,&H00,140)
Check_sum = CheckSum (CH200string,5,0 )
'For the string to be correct the check sum must be zero.
If Check_sum = 0 Then
  'Split up the main string from PS/CH200 into separate components.
  SplitStr (CH200_MX(),CH200string,COMMA,4,4)
  'Sort out the components to specific variables.
  BattTargV = CH200_MX(1)
  DgtlPotSet = CH200_MX(2)
  BattCap = CH200_MX(3)
  Qloss = CH200_MX(4)
EndIf

 'Add up the current being consumed by the system. Include battery current in
 'the calculation ONLY when it is positive - going into the battery.
 'Current values coming from the PS/CH200 are in amps. Have to convert that to
 'amp-hours based on the slow scan rate.
Sys_Amp_hr = Sys_Amp_hr + ILoad
If IBatt > 0 Then
  Sys_Amp_hr = Sys_Amp_hr + IBatt
  Batt_Amp_hr = IBatt
EndIf
Load_Amp_hr = ILoad
```

```
  For n = 1 To 3
    Amp_Hour(n) = Amp_Hour(n) * SlowScan/3600
  Next n
    'Must call these tables from the slow scan or the values will be wrong!
  CallTable Amp_hr
  CallTable Amp_Day
 NextScan
EndProg
```

A.6 Changing the SDI-12 Address Programming Example

The CR1000 program example below shows how to change the SDI-12 address in PS200/CH200 using SDI-12 port 3 (C3). The Campbell Scientific programming software, CRBasic, would be used to set the variable "new_sdi12_address" initially to a desired value but the value can be changed “on the fly” while monitoring the data by changing the variable to a new value. The CR1000 communicates with the PS200/CH200 via SDI-12 to change the PS200/CH200's SDI-12 address and then reads all the measurements from the PS200/CH200 as a test.

```
'CR1000 Series Datalogger
'Program: Change_SDI12_Address.CR1
'Date: 1.September.2010
'Ver: A
'
'Notes: Program sets the SDI-12 address of a PS200/CH200 and tests
'the SDI-12 interface. Changes to the SDI-12 address can be made
'"on the fly" by changing the variable "new_sdi12_address"!
'
'SDI-12 addresses must be numbered 0 through 9, a through z, or A
'through Z.
'Connect the PS200/CH200 to SDI-12 port C3 of the CR1000.
'Use Campbell Scientific SDI-12 cable part # 20769.
'
'SDI-12 CABLE TO CR1000 WIRING
'-----------------------------
'WHITE: C3
'BLACK: G
'CLEAR: G
'
'CAUTION:  ONLY ONE SDI-12 DEVICE MUST BE CONNECTED TO THE
'SDI-12 CONTROL PORT 3 OR THIS WILL NOT WORK:
'
'Public Variables
  'Datalogger panel temperature: Celsius
Public PanelTempC
  'Existing SDI-12 address in the PS/CH200
Public sdi12_address As String
  'New SDI-12 address.
Public new_sdi12_address As String
Dim SDI_Addr_Chng As String

  'Array to hold all the data coming from the PS200/CH200
Public CH200_M0(9)
  'Alias names for array elements.
  'Battery voltage: VDC
Alias CH200_M0(1)=VBatt
```

```
  'Current going into, or out of, the battery: Amps
Alias CH200_M0(2)=IBatt
  'Current going to the load: Amps
Alias CH200_M0(3)=ILoad
  'Voltage coming into the charger: VDC
Alias CH200_M0(4)=V_in_chg
  'Current coming into the charger: Amps
Alias CH200_M0(5)=I_in_chg
  'Charger temperature: Celsius

Alias CH200_M0(6)=Chg_TmpC
  'Charging state: Cycle, Float, Current Limited, or None
Alias CH200_M0(7)=Chg_State
  'Charging source: None, AC, or Solar
Alias CH200_M0(8)=Chg_Source
  'Check battery error: 0=normal, 1=check battery
Alias CH200_M0(9)=Ck_Batt

  'Arrays to hold the associated words for the charge state, charge source,
  'and check battery values.
Dim ChargeStateArr(6) As String
Dim ChargeSourceArr(3) As String
Dim CheckBatteryArr(2) As String

  'Variables to hold the words for charge state, charge source, and check
  'battery.
Public ChargeState As String
Public ChargeSource As String
Public CheckBattery As String

  'Stored hourly data.
DataTable (Hour,1,-1)
  DataInterval (0,1,Hr,10)
  Minimum (6,CH200_M0(),FP2,0,False)
  Maximum (6,CH200_M0(),FP2,False,False)
  Minimum (1,PanelTempC(),FP2,0,False)
  Maximum (1,PanelTempC(),FP2,False,False)
EndTable

  'Main Program
BeginProg
    'Load arrays with words to associate with the charge state, charge source
    'and check battery values from the PS/CH200.
  ChargeStateArr(1) = "Regulator Fault"
  ChargeStateArr(2) = "No Charge"
  ChargeStateArr(3) = "Current Limited"
  ChargeStateArr(4) = "Cycle Charging"
  ChargeStateArr(5) = "Float Charging"
  ChargeStateArr(6) = "Battery Test"

  ChargeSourceArr(1) = "None"
  ChargeSourceArr(2) = "Solar"
  ChargeSourceArr(3) = "Continuous"

  CheckBatteryArr(1) = "Normal"
  CheckBatteryArr(2) = "Check Battery"

  new_sdi12_address = "3"                  'Set this to your desired SDI-12 address. For this
    'example use a 3.
```

```
  'Get current SDI-12 address.
SDI12Recorder (sdi12_address,3,sdi12_address,"?!",1.0,0)

Scan (5,Sec,0,0)
  PanelTemp(PanelTempC,_60Hz)       'Measure datalogger panel temperature.
  CallTable Hour
NextScan

  'Test the PS200/CH200 at the new SDI-12 address.
SlowSequence
Scan (10,Sec,0,0)
  If new_sdi12_address <> sdi12_address Then
    SDI_Addr_Chng = "A" + new_sdi12_address + "!"
    SDI12Recorder (sdi12_address,3,sdi12_address,SDI_Addr_Chng,1.0,0)
  EndIf

  SDI12Recorder (CH200_M0(),3,new_sdi12_address,"MC!",1.0,0)
    'Array values start with one. Values for charge state start with -1.
    'Have to shift the value by two to line it up with the correct words
    'in the array.
  ChargeState = ChargeStateArr(Chg_State + 2)
    'Values for charge source start with zero. Have to shift the value
    'by one to line it up with the correct words in the array.
  ChargeSource = ChargeSourceArr(Chg_Source + 1)
    'Values for check battery start with zero. Have to shift the value
    'by one to line it up with the correct words in the array.
  CheckBattery = CheckBatteryArr(Ck_Batt + 1)
NextScan
EndProg
```

CAMPBELL SCIENTIFIC COMPANIES

Campbell Scientific, Inc. (CSI)
815 West 1800 North
Logan, Utah 84321
UNITED STATES
www.campbellsci.com • info@campbellsci.com

Campbell Scientific Africa Pty. Ltd. (CSAf)
PO Box 2450
Somerset West 7129
SOUTH AFRICA
www.csafrica.co.za • sales@csafrica.co.za

Campbell Scientific Australia Pty. Ltd. (CSA)
PO Box 8108
Garbutt Post Shop
QLD 4814 AUSTRALIA
www.campbellsci.com.au • info@campbellsci.com.au

Campbell Scientific do Brazil Ltda. (CSB)
Rua Luisa Crapsi Orsi, 15 Butantã
CEP: 005543-000 São Paulo SP BRAZIL
www.campbellsci.com.br • suporte@campbellsci.com.br

Campbell Scientific Canada Corp. (CSC)
11564 - 149th Street NW
Edmonton, Alberta T5M 1W7
CANADA
www.campbellsci.ca • dataloggers@campbellsci.ca

Campbell Scientific Centro Caribe S.A. (CSCC)
300N Cementerio, Edificio Breller
Santo Domingo, Heredia 40305
COSTA RICA
www.campbellsci.cc • info@campbellsci.cc

Campbell Scientific Ltd. (CSL)
Campbell Park
80 Hathern Road
Shepshed, Loughborough LE12 9GX
UNITED KINGDOM
www.campbellsci.co.uk • sales@campbellsci.co.uk

Campbell Scientific Ltd. (France)
3 Avenue de la Division Leclerc
92160 ANTONY
FRANCE
www.campbellsci.fr • info@campbellsci.fr

Campbell Scientific Spain, S. L.
Avda. Pompeu Fabra 7-9
Local 1 - 08024 BARCELONA
SPAIN
www.campbellsci.es • info@campbellsci.es

Campbell Scientific Ltd. (Germany)
Fahrenheitstrasse13, D-28359 Bremen
GERMANY
www.campbellsci.de • info@campbellsci.de

Please visit www.campbellsci.com to obtain contact information for your local US or International representative.