Sensoray Models 618/619
PCI Smart A/D Cards

Revised May 12, 1999

Table of Contents

1 Basics

1.1 Limited Warranty

Sensoray Company, Incorporated (Sensoray) warrants the Model 618/619 hardware to be free from defects in material and workmanship and perform to applicable published Sensoray specifications for two years from the date of shipment to purchaser. Sensoray will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The warranty provided herein does not cover equipment subjected to abuse, misuse, accident, alteration, neglect, or unauthorized repair or installation. Sensoray shall have the right of final determination as to the existence and cause of defect.

As for items repaired or replaced under warranty, the warranty shall continue in effect for the remainder of the original warranty period, or for ninety days following date of shipment by Sensoray of the repaired or replaced part, whichever period is longer.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. Sensoray will pay the shipping costs of returning to the owner parts that are covered by warranty.

Sensoray believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, Sensoray reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult Sensoray if errors are suspected. In no event shall Sensoray be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, SENSORAY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF SENSORAY SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. SENSORAY WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEROF.

1.2 Special Handling Instructions

The model 618 and 619 Smart A/D circuit boards contain CMOS circuitry that is sensitive to Electrostatic Discharge (ESD).

Special care should be taken in handling, transporting, and installing the Smart A/D board to prevent ESD damage. In particular:

- Do not remove the circuit board from its protective anti-static bag until you are ready to configure the board for installation.
- Handle the circuit board only at grounded, ESD protected stations.
- Remove power from the target system before installing or removing the circuit board.

2 Introduction

2.1 Functional Description

Sensoray models 618 and 619 Smart A/D boards provide eight (model 618) or sixteen (model 619) independent sensor channels that interface process sensors directly to the PCI bus.

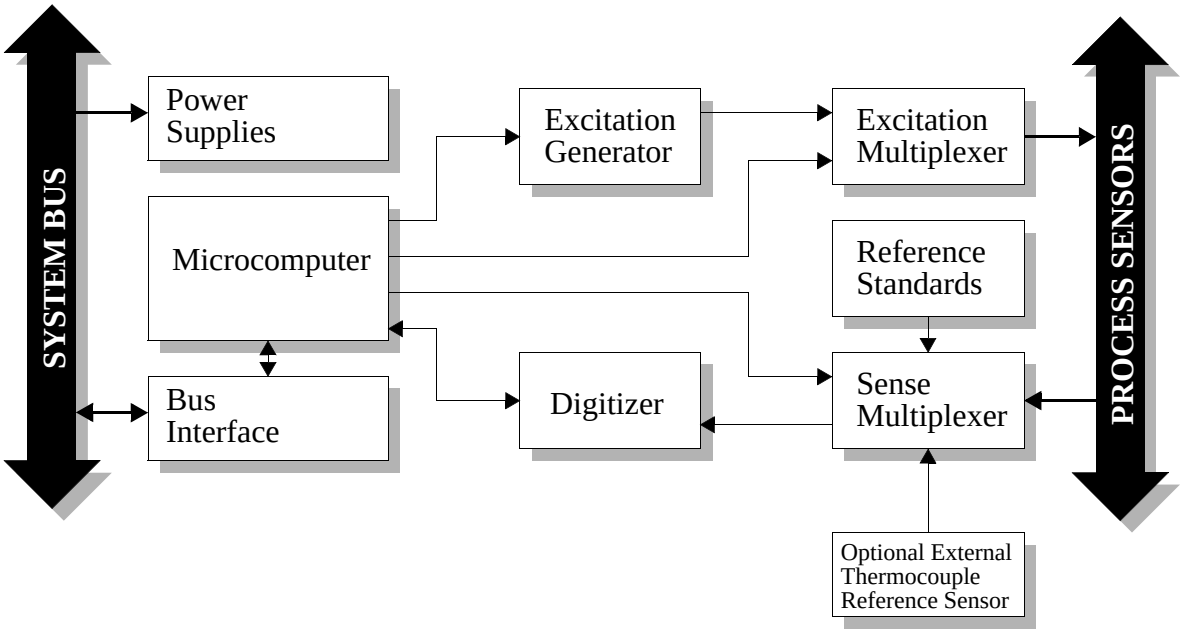
An onboard microcomputer continuously scans the sensor channels. As each channel is scanned, the sensor is excited, digitized, linearized, converted to engineering units appropriate for the sensor type, and stored in onboard memory. The freshest sensor data from all channels are always available for instant access.

Standard 100 mil, 40-pin headers are provided for connecting the sensor channels to external circuitry. Optional mating termination boards are available to facilitate connections to application sensors.

Smart A/D features include:

- Embedded processor off-loads the most demanding data acquisition functions from the host.
- Each sensor channel may be independently configured, via software, to accept thermocouples, RTD's, strain gages, thermistors, resistors, 4-to-20 mA current loops, or DC voltage inputs.
- Sensor channels employ fully differential sense inputs to help eliminate ground loops.
- All sense inputs are protected from high differential and common mode voltages.
- Pulsed excitation is provided for passive sensors, such as thermistors, resistors, RTD's, and strain gages. Pulsed excitation minimizes sensor self-heating effects and helps to reduce system power consumption.
- Sensor excitation signals are electrically separate from the sense inputs so that high accuracy three-wire or four-wire circuits may be implemented.
- Automatic cold junction compensation is provided for thermocouples. The remote compensation sensor—present on all Smart A/D termination boards—employs a dedicated measurement channel so that all process sensor channels remain available for application use.
- Programmable alarm limits for each channel. A hardware Alarm status flag provides instant access to limit violation detection without cumbersome polling and evaluation of sensor data for limit violation detection.
- Fully electronic calibration—no trimpots to adjust. Highly stable internal standards provide excellent measurement accuracy without the need for frequent calibrations.

2.2 Block Diagram



3 Configuration

The Smart A/D board requires the configuration of programming shunts to select hardware options. These options must be configured before installing the Smart A/D board into the target system. Shunts are configured by installing or removing shorting shunt plugs, at various locations on the Smart A/D circuit board, as appropriate for your application.

A supply of programming shunts, of sufficient number to program all board options, is included with the Smart A/D board.

3.1 Signal Conditioning Circuit (SCC)

Each channel is provided with a signal conditioning circuit (SCC) which may be inserted into the sense signal path. This conditioning circuit performs three functions:

- **Common-mode tie-down.** This function helps to prevent the common mode voltage (CMV) from straying beyond the input CMV range of the Smart A/D measurement section. This function is required when a sensor channel is driven from an isolated source, such as a battery, isolated power supply, or thermocouple.
- **Open-sensor detect.** If either of the two sense signals should become disconnected from the source, this function forces a differential voltage of 700mV, minimum, to appear across the sense inputs. This function is useful when a sensor is used in a control loop.
- **RF shunt.** This function, which shunts RF noise to ground, is essential when connecting to isolated sources, such as thermocouples.

3.1.1 Side Effects

Sense input impedance is reduced when a channel's SCC is enabled. See "General Specifications" on page 19 for details.

3.1.2 Recommended Settings

The decision of whether or not to enable a channel SCC is influenced primarily by the type of sensor to be interfaced, and, in the case of active sources, whether or not the source is ground-referenced.

Listed below are recommendations for enabling vs. disabling the SCC:

Table 1: Recommended signal conditioner settings

Sensor Type	SCC
Ungrounded Thermocouple	Enabled
Isolated DC Voltage Source	
Grounded Thermocouple	Disabled
Ground-referenced DC Voltage Source	
4-to-20 mA loop	
Resistor	
RTD	
Strain/Pressure Gage	
Thermistor	

3.2 Shunt Mapping

Shunts E1 through E32 are used to control the enabling of SCCs for all sensor channels. Each channel is affiliated with two shunts. To enable a channel's SCC, you must install both of the affiliated shunts. Similarly, to disable the SCC, you must remove both shunts.

The following tables show the mapping between channels and affiliated shunt pairs for SCC enables.

Table 2: Signal Conditioner shunt mapping, channels 0-7

Channel	0	1	2	3	4	5	6	7
Shunt Numbers	E1	E2	E3	E4	E5	E6	E7	E8
	E17	E18	E19	E20	E21	E22	E23	E24

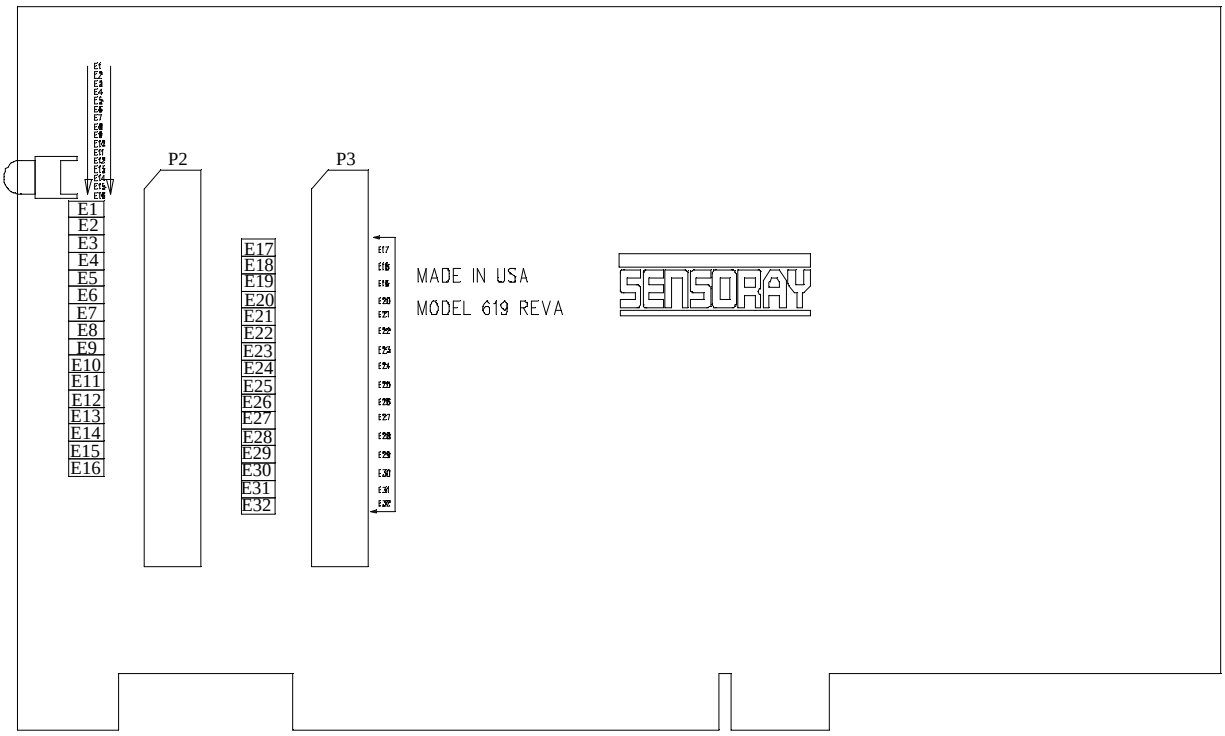
Table 3: Signal Conditioner shunt mapping, channels 8-15

Channel	8	9	10	11	12	13	14	15
Shunt Numbers	E9	E10	E11	E12	E13	E14	E15	E16
	E25	E26	E27	E28	E29	E30	E31	E32

For example, install shunts E3 and E19 to enable the SCC for sensor channel 2.

3.3 Shunt Locations

The diagram below shows the locations of hardware programming shunts on the Smart A/D board.



3.4 Reset State

At power-up, or after a soft reset (see Section 5.2.1) or a system hard reset, the Smart A/D board is in its default state as follows:

- Standard speed mode (22ms/channel) is selected.
- 60Hz rejection mode is active.
- All channels are enabled for scanning.
- Sensor data values are reset to zero. Sensor data is not available immediately after a board reset. The maximum elapsed time that must pass before new sensor data becomes available is equal to 35 channel slot times. Details of channel slot timing can be found in Section 8.1
- Channels are configured for the 5-volt input range, with 500µV resolution.
- Software filter constants are set to zero (disabled).
- Alarm threshold limit values are indeterminate.
- All channel alarms are disabled.
- All channels are programmed to fail high upon open-sensor detection.

3.5 Fault Indicator

A red light-emitting diode (LED) on the Smart A/D board is used as a status indicator. This LED is energized under either of the following circumstances:

- A board reset is in progress.
- A board fault has been detected.

When the Smart A/D board is functioning normally, the LED is energized for approximately one-half second in response to a board reset, either hard or soft. In the case of power-up and hard resets, this time period may be longer as the result of a stretched hardware reset signal on the system bus.

If the LED is persistently energized, there may be a hardware fault on the A/D board. In this case, be sure to check all sensor connections, board orientation and seating, etc. before concluding that the board has a fault.

4 Sensor Connections

4.1 Overview

Sensors are connected to the Smart A/D by means of 40-pin headers P2 and P3.

4.1.1 Termination Boards

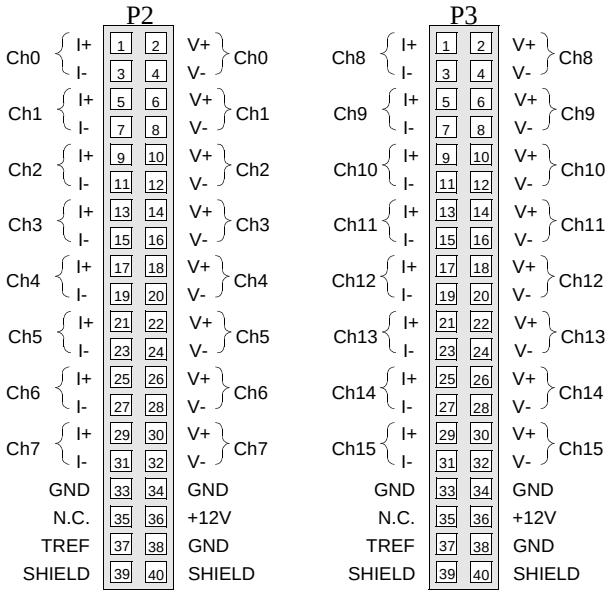
It is recommended that sensors be connected to the Smart A/D board by means of optional screw termination boards (TB), such as Sensoray models 7409TB or 7409TC. These termination boards integrate reference-junction sensors for thermocouples and are designed to mate directly to Smart A/D board headers P2 and P3.

4.1.2 Sensor Hot Insertion

Sensors may be electrically connected and disconnected from the Smart A/D or termination board without removing the Smart A/D board or power from the backplane.

4.2 Connector Pinouts

Show here are the pin assignments for headers P2 and P3:



Sensor channels 0 through 7 connect to header P2.

On model 619 only, sensor channels 8 through 15 connect to header P3. Although it may be present on the model 618, header P3 is not used

4.2.1 Reference Junction Sensor

Three of the signals shown near the bottom of the above header diagrams—+12V, GND and TREF—are used for interfacing the Smart A/D to a remote thermocouple reference-junction temperature sensor. One such sensor is provided on all models of Sensoray Smart A/D termination boards.

The +12V and GND signals provide excitation to the reference sensor. The TREF connection provides a path from the sensor output to a dedicated input channel on the Smart A/D board.

If you are using thermocouples in your application and you will be implementing your own termination system, you must provide your own remote temperature sensor. The sensor, which produces an output of 10mV/°K, must have its output signal routed to the TREF pin.

4.3 Sensor Channels

Each sensor channel provides five signals for interfacing to a sensor. A sensor can have as many as five connections to a channel channel, or as few as two.

4.3.1 Sense Terminals

Two of the channel signals, designated V+ and V-, are universally used for all sensor types. These two signals are, respectively, the positive and negative differential voltage sense inputs.

Since the sense inputs accept differential signals, it is not required for either of the two input signals to be at ground potential. The Smart A/D digitizer measures only the difference voltage across the two sense inputs.

Due to common-mode voltage (CMV) input constraints, however, you must ensure that neither sense input, with respect to system ground, exceeds the maximum input CMV permitted by the Smart A/D.

Small excursions beyond the CMV limit may degrade measurement accuracy on the offending channel, while large excursions may cause measurement errors on other sensor channels, or worse yet, damage the board.

See “General Specifications” on page 19 for input CMV limit values.

4.3.2 Excitation Terminals

Passive sensors require connection to the I+ and I- signals. The I+ and I- signals supply positive and negative excitation, respectively, to passive sensors.

Different classes of excitation are applied to sensors; the class of excitation that is applied depends on the type of sensor to be measured.

Constant current excitation is used for high-accuracy, low resistance measurements. A current limited, constant voltage excitation is applied when measuring higher resistance values. Finally, a 10 Volt, high current excitation is applied to strain and pressure gages.

4.3.3 Shield Terminal

The fifth lead—named *S* for shield—connects to the cable shield, if present. The S signal is internally connected to the backplane ground. To avoid ground loops, the cable shield should never be connected to anything at the sensor end of the cable, nor should it be connected to another ground point.

A shield conductor is never absolutely required, but is sometimes an unavoidable noise-abatement measure.

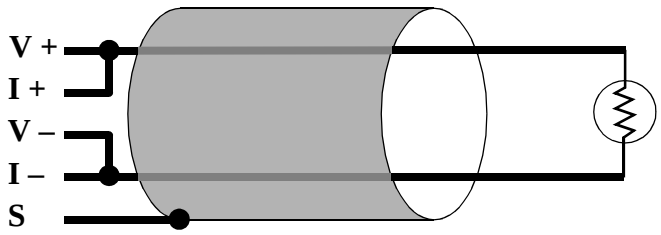
4.4 RTDs, Resistors and Thermistors

RTDs, resistors and thermistors can be connected to the TB in any of three possible configurations: two-wire, three-wire, and four-wire. In the following discussion, *sensor* is used interchangeably with *RTD*, *resistor* and *thermistor*.

4.4.1 Two-wire Circuit

The simplest configuration is the two-wire circuit. As the name implies, this configuration requires only two wires. The V+ and I+ terminals are shorted together at the TB, as are the V- and I- terminals.

Two-wire connection.

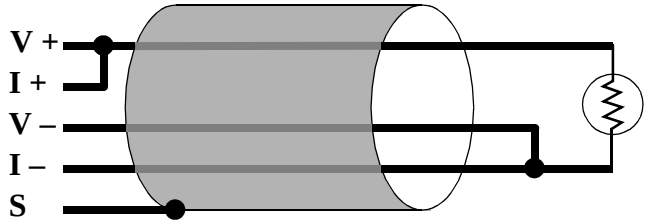


This circuit conserves wire, but it causes measurement error due to the voltage drop between the TB and sensor. There are two potential solutions for this problem: use short wires, or use more than two wires.

4.4.2 Three-wire Circuit

By using three wires, it is possible to reduce cable loss errors by 50 percent. Instead of shorting V- and I- together at the TB, discrete conductors are run from each of these terminals to the sensor. The wires are then shorted together at the sensor.

Three-wire connection

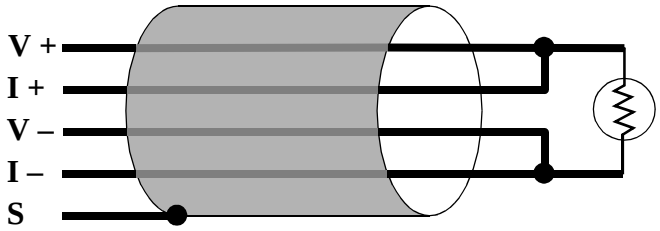


In this situation, the high impedance V- terminal detects the voltage at the sensor before any voltage drop can occur.

4.4.3 Four-wire circuit

Unlike the three-wire circuit, which exhibits half the cable losses of an equivalent two-wire circuit, a four-wire circuit completely eliminates cable losses.

Four-wire connection



Like the three-wire circuit, separate conductors are run from the V- and I- terminals to the sensor. In addition, separate conductors are run from V+ and I+ to the sensor, where they are tied together. This circuit eliminates cable loss effects from both the V+ and V- lines.

4.4.4 Recommended Practice

If sensor field wiring is exposed to electrical noise (i.e., the cable run is long or is close to noisy conductors) you should consider using shielded cable. The cable shield must be connected only to the S terminal on the TB and left unconnected at the sensor end of the cable.

The four-wire circuit should be used when accuracy is critical. This is especially important when making precision low-resistance value measurements, such as when using RTDs.

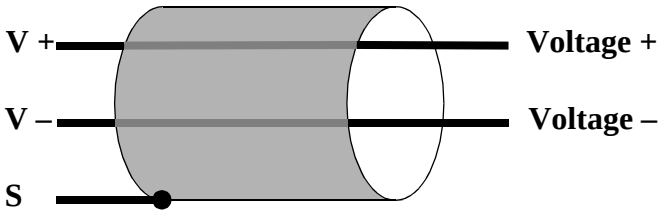
Thermistors have much higher resistance values than RTDs over most of their operating range. Consequently, the two-wire circuit may be used if your sensor will be operating exclusively at high resistance values. However, if your thermistor will be operating at lower resistance values, you should consider implementing a three-wire or four-wire circuit to reduce or eliminate cable-loss errors.

4.5 DC Voltage Sources

DC voltage sources are connected directly to the V+ and V- terminals. DC voltage sources should never be connected to the I+ or I- terminals.

Since all voltage input ranges are bipolar, DC voltages may be connected in either signal polarity. Although the diagram below shows Voltage+ connected to V+ and Voltage- connected to V-, there is no reason that Voltage+ must be positive with respect to Voltage-.

DC voltage connection



4.5.1 Recommended Practice

In order to assure accurate DC voltage measurements, high common-mode voltages (CMV) must not be permitted to appear on V+ or V-.

If your signal source is isolated (i.e., sourced from a battery or isolated power supply), you can connect V+ or V- to either S or the backplane power supply return. This connection can be implemented as a direct short or can be made through, for example, a 10KΩ resistor. There should be no significant DC current flowing through this connection; its purpose is to limit the common-mode voltage at the sense inputs.

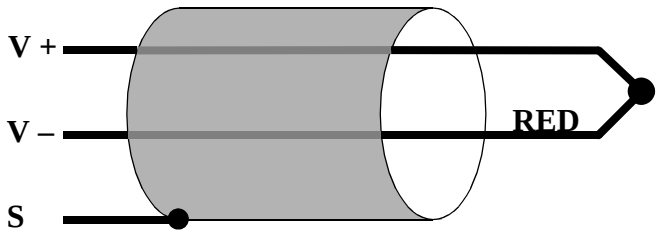
In the case of low source impedance voltage sources, another way to limit CMV is to install the channel signal conditioning circuit as described in Section 3.

4.6 Thermocouples

Thermocouple (TC) signals are connected directly to the V+ and V- terminals. TCs should never be connected to the I+ or I- terminals.

TC wires are color coded to indicate polarity. The red thermocouple wire is always negative, by convention. The positive TC wire should always be connected to the V+ terminal, and the negative wire should always be connected to the V- terminal.

Thermocouple connection



4.6.1 Recommended Practice

A TC reference-junction compensation sensor is required in order to obtain accurate TC measurements. This sensor is present on all Sensoray Smart A/D termination boards.

If you will be using a TB of your own design, you should physically arrange the compensation sensor so that it is thermally coupled to the TC reference junctions.

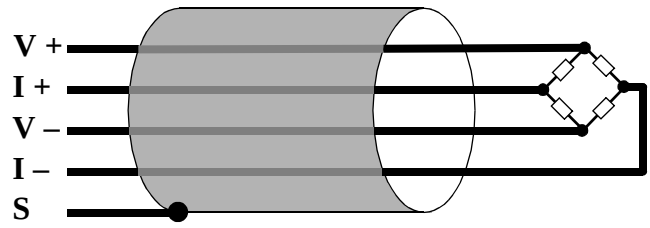
Significant measurement errors can occur if you allow the TB to be exposed to thermal transients, such as air flows from cooling fans or ambient breezes. For best results, the TB should be located within a protective enclosure after sensor terminations have been made.

If your TC is ungrounded, always enable the channel signal conditioning circuit. See “Configuration” on page 3 for details.

4.7 Strain and Pressure Gages

In a typical strain/pressure gage sensor, four wires are used to connect the gage to the sensor channel. Two of the wires supply gage excitation, while the other two wires transport the gage output signal to the Smart A/D measurement circuitry.

Strain/pressure gage connection



4.7.1 Recommended Practice

Due to the high measurement gain used, strain and pressure gages should use shielded cable. The shield should be connected only to the channel S terminal.

If you are using a six-wire gage, you should connect the gage's excitation source and excitation sense leads together at the I+ and I- terminals. Note that remote sensing of the excitation signals is not supported.

The gage impedance seen by the Smart A/D excitation source must be at least 120Ω to prevent automatic current limiting. If the input impedance is less than 120Ω, a resistor must be inserted in series with the gage excitation terminals to prevent excessive excitation circuit loading. Note that this will alter the mV/V rating of the sensor.

Gage output voltage must fall between -500mV and +500mV under all load conditions, including any offset caused by bridge imbalance. If the gage output voltage, at 10V excitation, might exceed these input limits, it may be necessary to configure the sensor channel for voltage input and supply external excitation to the gage.

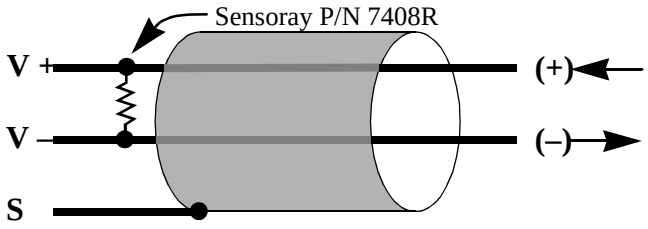
4.8 4-to-20 mA Current Loops

The Smart A/D supports 4-to-20 mA current loop inputs on any channel, subject to two constraints:

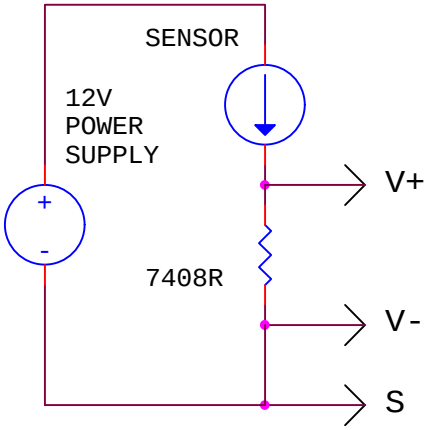
1. A 250Ω, 0.01% resistor must be installed as shown in the following diagram. This resistor, which converts the loop current to a voltage, is available as an option for the Smart A/D; order Sensoray part number 7408R.

2. Only the *grounded* end of the current loop may be connected to a sensor channel. This ensures that the common-mode voltage will not exceed Smart A/D input limits.

4-20 mA current loop connection



It is important to note the Smart A/D does not provide excitation for current loops. Current loops must be energized from an external power source, such as the isolated 12V power supply in the following schematic:



If the external power source is not isolated (i.e., its negative output terminal is connected to the backplane power supply return), you should not connect to the S terminal as shown in the above diagram, as a potential ground loop would be created. In such a case, leave the S terminal disconnected and connect to the V+ and V- terminals only.

5 Programming

The Smart A/D board is programmed by means of a small, yet powerful built-in command set. Commands are issued to the board by a client, which can be any PCI master. Some commands cause the board to respond by returning data to the client.

This chapter details the embedded Smart A/D command set and the programming interface which encapsulates the command access methods.

5.1 Standardized Interface

Because of the complexity of both the PCI interface and the native Smart A/D command set, a standardized application programming interface (API) has been developed. This API enables access to Smart A/D commands while maintaining consistency across various operating system and hardware platforms.

5.1.1 Distribution Diskette

The API consists of several software components that are distributed on floppy diskette with the Smart A/D board. The files on the distribution diskette include:

- `SmartAD.dll`—Windows 95/98/NT dynamic link library for Sensoray Smart A/D boards. Use this if you are developing a Windows application.
- `ClsSmartAD.hpp`—platform-independent C++ class for Sensoray Smart A/D boards. Use this if you are developing a non-Windows application.
- Application programming examples and demonstration programs.
- `ReadMe.txt`—contains information about the latest release and explains how to install and use the software components contained on the diskette.

5.1.2 API Functions

The remainder of this chapter focuses on the functional characteristics of each API function by discussing what happens on the Smart A/D board when an API function is executed, and the meaning of any values that are passed to or received from a function.

In some cases, arguments that are passed to or from an API function are specified as `void`. These arguments employ nomenclature borrowed from the C/C++ languages to indicate that no parameter value is passed.

Similarly, an argument that is prefixed with an asterisk (“*”) is indicative of a pointer.

You should refer to the documentation on the distribution diskette for detailed, up-to-date information about the current API release. This is recommended because:

- API arguments and return values shown in this document are only representative of the actual parameter values and ordering used.
- API functions usually include additional arguments that are not discussed here.
- The current version of the API may not include all functions or sensor types listed in this document.
- The API may contain additional functions, not mentioned here, that are essential to the creation of working application programs.

5.1.3 Channel Numbering Convention

Many of the API functions require a sensor channel number as a parameter. By convention, channel numbers on Smart A/D boards always begin at zero.

Channel numbers range from 0 to 7 on the model 618, and from 0 to 15 on the model 619.

5.2 Board Configuration Functions

5.2.1 ResetBoard

> Arguments: `void`
> Returns: `void`

This function, which invokes a “soft reset,” provides a means for forcing the Smart A/D to its reset state without altering the states of any other boards in the system. A soft reset has the same effect on the Smart A/D board as a hard system reset.

After executing this function, the Smart A/D is in its default state. See Section 3.4 for detailed information about the default state of the Smart A/D board.

5.2.2 GetFaultFlags

> Arguments: `void`
> Returns: `FaultFlags`

The Smart A/D board employs a hardware status register to provide rapid access to the states of various hardware fault conditions. In addition, the API maintains a log of fault events that have occurred in the API software functions. The status of all of these fault conditions, both

hardware and software, are available through the GetFaultFlags function.

For details on this function, refer to the documentation supplied on the distribution diskette.

5.2.3 GetFWVersion

Arguments: `void`
Returns: `VersionTimes100`

Returns the Smart A/D firmware version number, times 100. The version number is printed on the EPROM device that is plugged into the Smart A/D board.

5.2.4 SetHiSpeedMode

Arguments: `void`
Returns: `void`

Decreases the channel slot time (Section 8.1) to enable higher throughput.

Assuming the board is operating in the default 60Hz rejection mode, all channel slot times are reduced from 22 milliseconds (default) to 9 milliseconds, and the board is said to be operating in the "high-speed mode."

Decreasing the slot time causes measurement accuracy to be reduced by approximately 75 percent and may cause increased measurement noise. Also, line frequency rejection is no longer possible as the digitization period is shorter than one line cycle.

A board reset will restore slot times to their default values. This is the only way to resume the default update rate after invoking the SetHiSpeedMode function.

When the high-speed mode is active, any attempt to invoke this command again will be ignored.

5.2.5 SetReject50Hz

Arguments: `void`
Returns: `void`

Increases the digitizer's integration period from 16.7 milliseconds (default) to 20.0 milliseconds, enabling the Smart A/D to better reject 50Hz differential noise.

This function should be used in systems targeted for 50Hz power environments.

A board reset will restore the board to the default 60Hz rejection mode. This is the only way to return to the 60Hz rejection mode after invoking the SetReject50Hz function.

When the 50Hz rejection mode is active, any attempt to invoke this command again will be ignored.

5.3 Channel Configuration Functions

5.3.1 SetSensorType

Arguments: `ChannelNumber`
`SensorDefinitionCode`
Returns: `void`

Declares the type of sensor connected to a channel. By declaring the sensor type, you are implicitly selecting the excitation mode, measurement gain setting and linearization curve.

Upon successful execution of this function, the onboard microcomputer will perform all subsequent scans of the specified channel as appropriate for the declared sensor type.

This function has several side-effects that should be noted:

- Sensor data is not available immediately after declaring a channel's sensor type. After executing this function, sensor data is reset to zero. The maximum elapsed time that must pass before new sensor data becomes available is equal to ten (model 618) or nineteen (model 619) channel slot times. Details of channel slot timing can be found in Section 8.1.
- The channel filter constant (Section 5.3.2) is reset to zero, effectively disabling the software filter function.
- Limit alarms are disabled and limit values are indeterminate, as the new sensor type may have a new type of engineering units.

Typically, the client will call this function once for each sensor channel. Each execution of this function declares the sensor type for a single channel, therefore eight (model 618) or sixteen (model 619) invocations must occur to declare all channel sensor types.

Each sensor type supported by the Smart A/D board is assigned a unique value, called the Sensor Definition Code. This value, along with the target sensor channel number, is passed to the Smart A/D board as a function argument.

Upon reset of the Smart A/D board, all channel sensor types default to the ±5-volt input range, with 500µV resolution, and all sensor data values are reset to zero.

Channel sensor types remain at the default setting until changed by this function. If the default sensor type is suitable for your application, you need not execute this function at all.

Any attempt to declare an unsupported sensor type will result in selection of the default sensor type.

Declaring a sensor type as "Disabled" inhibits scanning of the corresponding channel, resulting in increased throughput for all remaining active channels.

See "Sensor Specifications" on page 20 for a complete list of supported sensor types. Refer to the API documentation on the distribution diskette for a list of sensor definition codes.

5.3.2 SetFilter

Arguments: `ChannelNumber`
`FilterConst`
Returns: `void`

Establishes a digital single-pole low-pass filter for the specified channel. Filter weighting is specified by the `FilterConst` parameter, which may have any value from 0 to 255. The relationship between `FilterConst` and filter percentage is:

$$FilterPercentage = \frac{FilterConst}{256}$$

For example, a `FilterConst` value of 0x40 (64 decimal) results in a 25% filter. This means that the sensor data value made available to the client is the sum of 75% of the newest unfiltered value plus 25% of the previous filtered value.

All filter constants default to zero upon board reset. This value disables the filters, thereby maximizing the frequency response of all channels.

A non-zero filter constant will roll off the channel frequency response. Since the filter is implemented in the digital domain, noise frequencies at or above nyquist must be pre-filtered before arriving at the Smart A/D.

Some applications require known filter time constants. In such systems, you may use the following expression to calculate the time constant (in seconds) as a function of `FilterConst`:

$$\tau \approx \frac{(-1.06)(NumActiveChans)(ChanSlotTime)}{\ln(FilterConst/256)}$$

In the above expression `NumActiveChans` is the number of channels in the scan loop (all channels that have not been disabled), and `ChanSlotTime` is the channel slot time, as described in Section 8.1.

It may not be practical to compute exact filter constants in cases where the application doesn't require a known time constant. In such cases, you will probably find that it is more effective to find the best filter weighting by experimenting with different filter constants.

5.3.3 SetFailMode

Arguments: `ChannelNumber`
`FailHigh`
Returns: `void`

Specifies whether, in the event an open sensor is detected, the sensor data value should default to a large positive value (fail high) or a large negative value (fail low).

The `ChannelNumber` argument designates the sensor channel to be affected, while the boolean `FailHigh` specifies whether the channel should fail high or low.

This function is useful for triggering alarms when open sensors are detected, and for forcing the desired system response to sensor fault conditions in closed-loop control applications.

After a board reset, all channels are, by default, programmed to fail high.

5.4 Sensor Acquisition Functions

5.4.1 GetAmbientTemp

Arguments: `void`
Returns: `Temperature`

Returns the reference temperature, in degrees Centigrade, from an optional Sensoray termination board.

This is useful for monitoring thermocouple reference junctions and can also serve as a debugging aid during Smart A/D installation and application development.

This function returns an indeterminate value if no Sensoray termination board is connected to the Smart A/D board.

5.4.2 GetSensorData

Arguments: `ChannelNumber`
Returns: `SensorData`

Returns the most recently acquired sensor data from one sensor channel. The returned value, `SensorData`, is expressed in engineering units that are appropriate for the channel's sensor type.

For example, channels configured for any type of temperature sensor typically return values in Degrees Centigrade, while channels configured for DC Voltage measurement return values in Volts.

Channels that have been programmed as sensor type "Disabled" return indeterminate values.

Refer to "Sensor Specifications" on page 20 for information regarding the resolution, accuracy and engineering units that apply to specific sensor types.

5.4.3 GetAllSensors

Arguments: `*SensorData`
Returns: `void`

Fills an array with sensor data from all channels on a Smart A/D board. The `SensorData` argument is a pointer to an application array which is to receive the sensor data.

Data values are returned from all sensor channels, even those that have been declared as type "Disabled." Disabled channels return indeterminate values.

As in the GetSensorData function, sensor data are scaled to engineering units suited to the sensor type. Refer to "Sensor Specifications" on page 20 for information regarding the resolution, accuracy and engineering units that apply to specific sensor types.

5.5 Alarm Functions

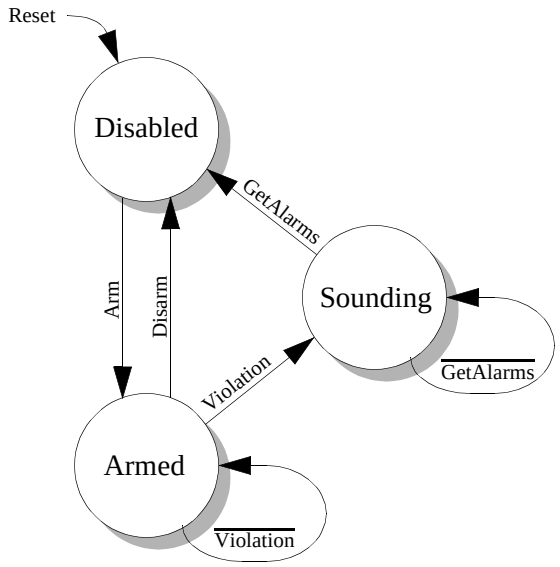
Each sensor channel has the ability to detect limit violations. A limit violation occurs when a channel's sensor data value exceeds a programmed threshold. When such violations are detected, a channel is said to be "sounding" an alarm.

Alarm limit thresholds are independently programmed for each channel by means of the SetAlarmLimits function. After setting these thresholds, alarm sounding may be enabled under software control.

Although each channel has two alarm limit values, each channel technically has only one alarm. When either limit is violated (exceeded), the channel alarm is asserted.

5.5.1 Alarm States

The following state diagram illustrates the operation of alarms for an individual sensor channel:



Initially, all channel alarms are disabled and each channel is in the Disabled state. The SetAlarmEnable function is used to switch between the Disabled and Armed states.

If a limit violation is detected while the alarm is Armed, the channel switches to the Sounding state. This causes the Smart A/D Alarm status flag to be asserted. The Alarm status flag may be accessed indirectly through the GetAlarms function.

A channel remains in the Sounding state until a GetAlarms function is executed. When the GetAlarms function executes, the channel switches to the disabled state, where it remains until the client again arms the channel alarm.

5.5.2 SetAlarmLimits

Arguments: `ChannelNumber`
`HighLimit`
`LowLimit`
Returns: `void`

Declares the upper (`HighLimit`) and lower (`LowLimit`) alarm threshold values for a channel. An alarm will sound if the target channel's sensor data value becomes positive with respect to `HighLimit`, or

negative with respect to `LowLimit`, while the channel alarm is armed.

The `HighLimit` and `LowLimit` arguments are expressed in the same engineering units used by the sensor that is connected to the target channel. For this reason, the sensor type must be declared (see Section 5.3.1) before invoking this function.

Suppose, for example, the target channel has been previously configured for a type K thermocouple. By specifying `LowLimit` and `HighLimit` values of 110.0 and 126.0, respectively, and then arming the channel alarms (Section 5.5.3), an alarm will sound if the channel's thermocouple temperature wanders outside the range +110 to +126°C.

The `HighLimit` value should always be positive with respect to the `LowLimit` value, otherwise an alarm will sound immediately when alarms are armed.

Alarm limit values are indeterminate after a board reset. Channel alarm limit values should be set by this function before arming the channel alarms.

5.5.3 SetAlarmEnable

Arguments: `ChannelNumber`
`LimitEnable`
Returns: `void`

Enables or disables limit violation detection for the specified channel. The `ChannelNumber` parameter designates the target channel, and the boolean `LimitEnable` argument specifies whether alarm sounding is to be permitted (i.e., the channel is to be armed) on the target channel.

Channel alarm limit values should be set (Section 5.5.2) before invoking this function.

By default, alarms are disabled on all channels after a board reset. In addition, individual channel alarms are disabled in response to any of the following events:

- The client explicitly disables an alarm by calling this function with `LimitEnable` set to false.
- An alarm sounds on the channel. This prevents the alarm from sounding again after the client has acknowledged the alarm with the GetAlarms function.
- A new sensor type is declared for a channel. This not only disables the channel alarm, but also renders the channel's limit values indeterminate.

5.5.4 GetAlarms

Arguments: `*AlarmStatus`
Returns: `AlarmSounding`

Fills an array with the alarm status of all channels on a Smart A/D board and restores any sounding channels to the Disabled alarm state.

The `AlarmStatus` argument is a pointer to an application array which is to receive the alarm status.

The returned boolean value, `AlarmSounding`, indicates whether any alarms were sounding when this function was called. It is obviously not necessary to evaluate the contents of the `AlarmStatus` array if `AlarmSounding` returns false.

5.6 Strain/Pressure Gage Functions

The strain/pressure gage functions discussed in this section are applicable only to sensor channels that have been configured for gages. API gage functions that are directed to non-gage channels (channels which have not been declared to be the strain/pressure gage sensor type) will be ignored.

After a board reset, you must configure any channels that will be connected to strain or pressure gages. Follow this procedure to configure a sensor channel for operation with a gage:

- Call the SetSensorType function to declare that the sensor channel will be connected to a gage.
- Calibrate the gage. The first time this is done, you must use the SetGageZero and SetGageSpan functions. If this is not the first gage calibration for the target channel, you may optionally use the SetGageCal function, or you may continue to use the SetGageZero and SetGageSpan functions.

5.6.1 SetGageZero

Arguments: `ChannelNumber`
Returns: `void`

Declares a zero-load condition for a strain/pressure gage channel.

This function tells the Smart A/D that there is presently no load applied to the target channel's gage. The Smart A/D registers this condition and uses it as a normalization reference for all subsequent measurements on the target channel.

Typically, this function is invoked just before a call to SetGageSpan. Together, the SetGageZero and SetGageSpan functions serve to completely calibrate a gage channel.

5.6.2 SetGageSpan

Arguments: `ChannelNumber`
`LoadValue`
Returns: `void`

Declares a known load condition for a strain/pressure gage channel, and the numerical load value expected from the Smart A/D board when measuring the current load.

This function is used in conjunction with the SetGageZero function to prepare a sensor channel for gage measurements. Invoke this function after declaring the sensor type, but before acquiring sensor data from a channel.

The SetGageSpan function requires that a precisely known load, called the "calibration load," be applied to the target sensor. In some cases the calibration load can be simulated by using an internal, precision calibration resistor—present in many gages—to unbalance the sensor bridge. In the case of gages that do not provide a calibration resistor, a mechanical load must be applied to the sensor.

Ideally, the calibration load will be close to the actual load values to be measured. If a wide range of load values are to be measured, it is best to employ a calibration load that is equal to or larger than the largest application load to be measured.

Before invoking this command, you must apply either a simulated or true mechanical zero load condition to the sensor and invoke the SetGageZero command, as described in Section 5.6.1. This must be done before (preferably immediately before) executing the SetGageSpan function.

5.6.3 SetGageTare

Arguments: `ChannelNumber`
Returns: `void`

Tares the strain/pressure gage connected to the specified channel. This function is used to compensate application load offsets.

For example, an empty container is often used to hold a material that is to be weighed. Since the container itself is not to be measured, the SetGageTare function can be invoked with a load consisting of only the empty container. After executing this function, the Smart A/D board will subtract off the container load and return only the load value of the container contents.

Before invoking this command, the gage channel should be calibrated. See Section 5.6.1, Section 5.6.2 and Section 5.6.5 for details.

5.6.4 GetGageCal

Arguments: `ChannelNumber`
`*GageCal`
Returns: `void`

Returns the gage calibration parameters from the sensor channel specified by `ChannelNumber`. The `*GageCal` argument is a pointer to an application buffer which is to receive the calibration parameters.

This function makes it possible to restore a channel's gage calibration without using the SetGageZero or SetGageSpan functions. The returned parameters may be saved by the application and used to later restore a gage calibration by means of the SetGageCal function.

The returned parameters are useful only if the gage has already been calibrated. This prior calibration may have been accomplished in either of two ways:

- Using SetGageZero and SetGageSpan functions. This method must be used if the sensor channel has never been calibrated for the gage to which it is connected.
- Using the SetGageCal function.

5.6.5 SetGageCal

Arguments: `ChannelNumber`
`*GageCal`
Returns: `void`

This command restores a channel's strain/pressure gage calibration settings by using calibration parameters that were obtained from the GetGageCal function.

The `*GageCal` argument is a pointer to an application buffer which contains the gage calibration parameters.

6 Calibration

Smart A/D boards are factory calibrated to meet or exceed published specifications. Because the board employs high-stability standards, calibration should not be necessary unless the board has been subjected to environmental extremes or has been in service for an extended time period.

WARNING: RECALIBRATION AFFECTS THE ACCURACY OF ALL SENSOR MEASUREMENTS. After calibrating a Smart A/D board, you can't "undo" the new calibration. The only way to correct a calibration error is to perform another calibration.

6.0.1 User-supplied References

To perform a board calibration, you must supply the following external reference standards:

- 5.0000V signal.
- 500.00mV signal
- 380.00Ω resistor.

The actual values of your standards may deviate from these target values by up to ±1%, but the values must be stable and precisely known.

6.0.2 Calibrate

Arguments: `ChannelNumber`
`StdNum`
`RefValue`

Returns: `Ignore`

This API function calibrates one of the Smart A/D internal reference standards and stores the calibration value in non-volatile memory on the Smart A/D board.

Although this function returns a value, the value is only of use for factory test purposes and should therefore be ignored by your application.

The `RefValue` argument is used to specify the exact value of the user-supplied reference. As mentioned earlier, the absolute value of the reference is not critical, but knowledge of the exact reference value is required.

`ChannelNumber` specifies the sensor channel number that is connected to the user-supplied reference. You may use any sensor channel of your choosing.

The `StdNum` argument specifies which of the external references is supplied to the target channel and the measurement range to be used for the calibration.

Table 4: StdNum Values for the Calibrate Function

StdNum	Reference	Measurement Range
0	5.0000 V	±5V, 200uV resolution
1	500.00 mV	±500mV, 20uV resolution
2	380.00Ω	400Ω, 0.02Ω resolution

6.1 Calibration Procedure

6.1.1 Order of Calibration

Three invocations of the Calibrate function are necessary to completely calibrate the Smart A/D board. These invocations must occur in the order shown in Table 4, from top to bottom.

Specifically, the 5V standard is calibrated first, followed by the 500mV standard and then the resistance standard.

6.1.2 Calibration Process

Each measurement range is calibrated by following a five-step procedure:

1. Connect the appropriate external reference standard, as specified in Table 4, to the target channel. In the case of the reference resistor, you must use a four-wire connection to the resistance standard.
1. Configure the target channel's sensor type as specified under Measurement Range in Table 4. This is done by calling the SetSensorType function with the appropriate sensor definition code.
2. Allow sufficient time for sensor data to stabilize on the target channel. A simple way to do this is to poll the target channel's sensor data, using the GetSensorData function, until non-zero data is returned.
3. Invoke the Calibrate command, using the StdNum value from Table 4, then wait for the return value.
4. Reset the Smart A/D board. This is normally accomplished by invoking the ResetBoard function.

7 Theory of Operation

7.1 Firmware

The Smart A/D's internal microcomputer executes a firmware-resident control program. At the heart of this program is a high-performance, event-driven kernel.

The following paragraphs describe the functions of some of the tasks that are managed by the kernel.

7.1.1 Digitizer

This task performs the fundamental data acquisition function central to the Smart A/D's purpose.

The Digitizer begins a conversion by selecting the next channel to be digitized. The sense and excitation multiplexers are switched to the desired channel, and, if the channel is connected to a passive sensor, the pulsed excitation source is configured and activated. The scanner then sleeps while the sensor signal stabilizes.

When the sensor signal has stabilized, the Digitizer initiates an A/D conversion. Other tasks are allowed to run while the conversion is taking place.

Upon completion of the conversion, the digitized value is forwarded to the Cruncher task.

7.1.2 Cruncher

The Cruncher task is responsible for polishing raw A/D data into finished form. Typically, the Cruncher task runs while the scanner is digitizing the next channel in the scan loop.

This task begins when it receives a digitized value from the Digitizer task. The data is normalized to the internal standards, then is linearized and converted to engineering units appropriate for the declared sensor type. The resulting value is passed through a single-pole low-pass filter.

Next, the filtered data is checked for alarm limit violations. If a limit has been exceeded, the channel alarm is disabled and the Alarm status flag is set.

Finally, the processed sensor data is posted to internal dual-ported RAM for rapid client access.

7.1.3 Command Processor

The Command Processor receives and executes commands from the client. Top priority is given to this task in order to minimize communication latency.

This task is run when a command is received from the client.

7.2 Analog Circuits

The Smart A/D analog circuitry is functionally partitioned into Measurement and Excitation sections.

7.2.1 Measurement Section

The Measurement section selects and conditions a sensor input signal for digitizing.

The Measurement section begins sensor signal processing by switching a pair of channel sense signals through a differential analog multiplexer. The selected channel is then conditioned by a programmable gain amplifier. Finally, the amplified signal is applied to the input of an integrating A/D converter.

Channel selection, amplifier gain, and A/D functions are all under control of the onboard microcomputer. None of these circuits are directly accessible by the client.

7.2.2 Excitation Section

The Excitation section sources a pulsed DC current or voltage that is used to excite passive sensors.

The Excitation section supplies one of three pulsed signals to passive sensors, depending on the declared sensor type. The length of the excitation pulse is equal to the channel slot time (see Section 8.1 for an explanation of slot time).

In the case of strain and pressure gages, 10VDC excitation is applied. This voltage is current limited so that shorting the excitation signals together or to ground will not damage the excitation source.

In the case of RTD's and the 400Ω and 3KΩ resistance ranges, a constant current of approximately 1.2mADC is forced through the sensor.

5VDC is applied, in series with a reference resistor, to sensors measured on the 600KΩ resistance range.

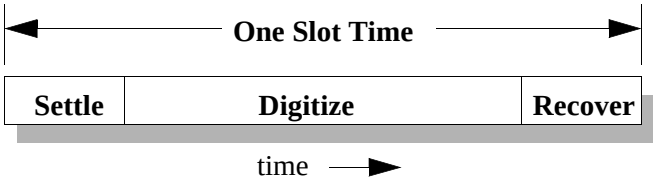
8 Timing

Three timing parameters are important from the application developer's viewpoint: Channel Update Rate, Data Age and Communication Latency. This section discusses these parameters along with some important related issues.

8.1 Slot Time

Channel Slot Time is the length of time required for the Smart A/D to completely process one sensor channel.

Each channel slot time encompasses three functional phases as shown in the following diagram:



In the first phase, a sensor channel is excited and its sense signals are routed through to the Smart A/D measurement section. The embedded microcomputer configures the signal path as appropriate for the sensor type, then the analog front end is allotted time to stabilize.

The digitizer acquires the sensor data value in the second phase. The Update Rate, which is described in the next section, can be increased by reducing the duration of the time slot Digitize phase. See Section 5.2.4 for details.

A Recover phase occurs at the end of the time slot. The function of this phase is to reset the analog front end in preparation for the next conversion.

Computationally intensive processes, such as linearization, alarm processing and software filtering, don't affect the slot time because they execute concurrently during the subsequent channel slot time.

8.2 Update Rate

Update Rate is defined as the number of times each second that a sensor channel acquires new sensor data.

8.2.1 Primary Influences

The two primary influences on update rate are the number of active channels and the channel slot time.

Due to timing uncertainties, the update rate is expressed as a range of values bounded by minimum and maximum times. The minimum and maximum update rate for any active channel is given by these functions:

$$UpdateRate_{min} = \frac{1}{(NumActiveChans + 1) \times SlotTime}$$

$$UpdateRate_{max} = \frac{1}{NumActiveChans \times SlotTime}$$

The time difference between the minimum and maximum update rate is due to the automatic, interleaved measurement of internal Smart A/D reference standards. Other than their timing impact, these measurements—which are scheduled and executed autonomously by the embedded processor—are transparent to the application program. At most, one internal standard will be measured per every sixteen sensor channel measurements.

Clearly, the update rate increases as the number of active channels decrease. Channels can be removed from the scan loop by disabling them with the SetSensorType command with the Disabled sensor definition code.

For example, a Smart A/D that has ten active channels and is running at the default channel slot time (22 milliseconds) would have a worst-case channel update rate ranging from 4.1 to 4.5 samples per second:

$$UpdateRate_{min} = \frac{1}{(10 + 1) \times 0.022} \approx 4.1Hz$$

$$UpdateRate_{max} = \frac{1}{10 \times 0.022} \approx 4.5Hz$$

8.2.2 Secondary Influences

Another influence on the update rate is the frequency of communication between client and Smart A/D.

The Smart A/D firmware is designed to minimize the impact of communications on the update rate. Even so, communication activity will sometimes interrupt scanning, which in turn will stretch the current channel's slot time in a non-deterministic way.

Because of the unpredictable effect on update rate, excessively frequent communication traffic between client and Smart A/D should be avoided.

8.3 Data Age

Data Age is the measure of time that has elapsed since a channel's data was acquired by the Smart A/D board. The age of any channel's data is directly related to the update rate.

The minimum possible data age is, of course, zero. This would be the case if the client retrieves sensor data immediately after new data is posted by the Smart A/D.

The maximum possible data age would apply if the client retrieves sensor data immediately before new data is posted by the Smart A/D. In this case, the data age is the reciprocal of the minimum update rate.

In practice, data age almost always lies between the minimum and maximum possible values.

8.3.1 Example

As an example, take the case shown at the end of Section 8.2.1. The minimum update rate is approximately 4.1 Hertz, so the maximum data age would be:

$$DataAge_{max} = \frac{1}{4.1} \approx 244ms$$

In this example, the age of sensor data from any given channel may range from zero to 244 milliseconds.

8.4 Communication Latency

Communication Latency is the time elapsed from a client request for sensor data to the acquisition of that data. This latency can be viewed as having two components: client overhead and Smart A/D response time.

Client overhead is the time spent executing those parts of the Smart A/D API functions that are in the computational domain of the client (i.e., not pending on a Smart A/D response). Since this overhead—which is purely a function of client clock rate and architecture—varies from one system to another, it must be independently determined for each system.

Smart A/D response time is the time spent in an API function while waiting for a Smart A/D response. For obvious reasons, short communication latency is a high-priority design objective in many applications.

Communication functions are event-driven in the Smart A/D's local environment and given highes priority in order to minimize the communication latency. As a result, Smart A/D response time is predictable with a good degree of accuracy:

Table 5: Command response times

Command	Response Time (microseconds)
GetSensorData	100
GetAllSensors	130

9 Specifications

9.1 General Specifications

Table 6: General Specifications

Category	Parameter	Condition	Specification
Power	Quiescent	+5VDC, ±5%	100mA, typical.
		+12VDC, ±5%	35mA, typical.
		-12VDC, ±5%	35mA, typical.
Temperature	Operating range	Guaranteed accuracy	-25°C to +85°C
Sense inputs	CMRR	CMV < 5V, f < 1KHz	80dB, minimum
	CMV	Guaranteed accuracy	±5.0V, maximum
	Input impedance	SCC disabled	1000 MΩ, minimum
		SCC enabled	1 MΩ, typical
	Input protection	Continuous	±20V CMV, maximum
		2 seconds, max.	±70V CMV, maximum
A/D converter	Type		Integrating, 16-bit resolution.
	Measurement period, per channel	60Hz reject mode (default mode)	16.7 milliseconds, typical (default mode). 4.0 milliseconds, typical (high-speed mode).
		50Hz reject mode	20 milliseconds, typical (default mode). 4.8 milliseconds, typical (high-speed mode).
	Total time slot, per channel	60Hz reject mode (default mode)	22 milliseconds, maximum (default mode). 9 milliseconds, maximum (high-speed mode).
		50Hz reject mode	25.3 milliseconds, maximum (default mode). 11 milliseconds, maximum (high-speed mode).
Excitation source	Pulse width		Applied for duration of channel time slot.
	Type	Strain/pressure gage	10VDC, 40mA maximum.
		400Ω/3KΩ ranges	1.2mADC.
		600KΩ range	5VDC in series with 4KΩ.
Bus interface	Type		PCI Slave, 32-bit, 33MHz
	Address requirements		Three I/O address blocks: sizes 32, 128 and 256 bytes. PCI configuration registers require an additional amount.

9.2 Sensor Specifications

Table 7: Sensor Specifications

Sensor Type		Range	Resolution [1]	Accuracy [1]
Thermocouple	B	0 to +1820°C	0.1°C	3.3°C
	C	0 to +1820°C	0.1°C	2.1°C
	E	-270 to +990°C	0.1°C	0.8°C
	J	-210 to +760°C	0.1°C	0.6°C
	K	-270 to +1360°C	0.1°C	1.0°C
	N	-270 to +1347°C	0.1°C	0.9°C
	T	-270 to +400°C	0.1°C	0.6°C
	S	0 to +1760°C	0.1°C	3.0°C
	R	0 to +1760°C	0.1°C	2.8°C
Thermistor	Omega 44006 or 44031	-55°C to +145°C	0.01°C	0.05°C
RTD	Cu 10Ω, 0.0367Ω/°C	0 to +119°C	0.1°C	0.6°C
	Pt 100Ω, 0.385Ω/°C	-200 to +800°C	0.05°C	0.2°C
		-200 to +409°C	0.0125°C	0.2°C
	Pt 100Ω, 0.392Ω/°C	-200 to +800°C	0.05°C	0.2°C
		-200 to +409°C	0.0125°C	0.2°C
	Ni 120Ω, 0.380Ω/°F [5]	-100 to +482°F	0.1°F	0.3°F
	Ni 200Ω, 1.10Ω/°C	-58.9 to +151.6°C	0.02°C	0.07°C
	Ni 1KΩ, 5.60Ω/°C	-50.0 to +129°C	0.025°C	0.08°C
Gage [6]		-500 to +500 mV	5µV [7]	30µV
DC Voltage		-5 to +5V [3]	500µV	600µV
		-5 to +5V	200µV	600µV
		-500 to +500mV	20µV	40µV
		-100 to +100mV	5µV	30µV
Current Loop [2]		4 to 20mA	0.01%	0.08%
Resistance		0 to 400Ω	0.02Ω	0.04Ω
		0 to 3KΩ	0.125Ω	0.25Ω
		0 to 600KΩ	31Ω	130Ω
Disabled [4]		Not Applicable — sensor removed from scan loop		

Notes

1. Measurement resolution and accuracy is specified for the default (versus high-speed) measurement mode. Derate resolution and accuracy by 75% when using the high-speed measurement mode.
2. Channels configured for 4-20mA current loops return data values representing a percentage of full-scale current (4mA= 0%, 20mA=100%). For example, a 12mA current would produce the data value 50.0, indicating 50% of full-scale.
3. This is the default sensor type after a board reset. It is provided for back-compatibility with earlier Sensoray Smart A/D products. Use of this type is not recommended for new designs, as higher resolution is available on the 5V, 200uV resolution range.
4. Declaring a channel's sensor type as Disabled removes the channel from the scan loop. Data values returned from disabled channels are indeterminate and should be ignored.
5. This is the only temperature sensor type that returns data in degrees Fahrenheit. All other temperature sensors return data in degrees Centigrade.
6. Gage specifications assume 4-wire bridge circuit, 120Ω minimum impedance, 10V excitation applied by sensor channel.
7. Use the following function to express gage measurement resolution in load units:

$$Resolution = \frac{RatedLoad}{(2 \times 10^6) \times RatedVoltageOut}$$

For example, take the case of a pressure gage that is rated at 3mV/V at 100 PSI full scale. The Smart A/D would be able to resolve a load to:

$$Resolution = \frac{100PSI}{(2 \times 10^6) \times 0.003V} = 0.017PSI$$