SuperLogics 8000 Series Analog Input Modules

SuperLogics' 8000 Series of compact remote data acquisition modules provides intelligent signal conditioning, analog I/O, and digital I/O. Through a cost-effective two-wire RS-485 communication network, remote data acquisition and control systems can be easily configured.

8000 Series New Features

1. Internal Self Tuner
2. Multiple Baud Rates Support
3. Multiple Data Formats Support
4. Internal Dual WatchDog
5. True Distributed Control
6. High Speed & High Density I/O

Warranty

All products manufactured by SuperLogics are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Disclaimer

SuperLogics assumes no liability for damages consequent to the use of this product. SuperLogics reserves the right to change this manual at any time without notice. The information furnished by SuperLogics is believed to be accurate and reliable. However, no responsibility is assumed by SuperLogics for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright 1999 by SuperLogics. All rights reserved.

Trademarks

The trademarks and tradenames used herein are registered to their respective owners.

Table of Contents

SuperLogics 8000 Series Analog Input Modules....1
8013....4
Introduction....4
More Information....4
8013 Pin Assignment....5
8013 Specifications....6
8013 Block Diagrams....7
8013 Application Wiring....8
8013 Default Settings....9
8013 Calibration....9
8013 Calibration....10
8013 Tables....11
Command Set Table....14
8013D Command Set Table....15
8014D....16
Introduction....16
More Information....16
8014D Pin Assignment....17
8014D Specifications....18
8014D Block Diagrams....19
8014D Application Wiring....20
Dry Contact Input....20
TTL Input....20
Output Drive SSR & Load....21
8014D Analog Input....21
8014D Default Settings....24
8014D Calibration....24
8014D Tables....26
8014D Command Set Table....28
8017....29
Introduction....29
1.1 More Information....29
8017 Pin Assignment....30
8017 Specifications....31
8017 Block Diagrams....32
8017 Application Wiring....33
8017 Default Settings....35
8017 Calibration....35
8017 Calibration....36
8017 Tables....37
Command Set Table....39
8017 Command Set Table....40
8018....41
Introduction....41
More Information....41
8018 Pin Assignment....42
8018 Specifications....43
8018 Block Diagrams....44
8018 Application Wiring....45
8018 Default Settings....47

8018 Calibration....48
8018 Calibration....49
8018 Tables....50
Command Set Table....54
8018 Command Set Table....55
APPENDIX A: COMMAND SETS....56
A1.1 %AANNTTCCFF....56
A1.2 #**....57
A1.3 #AA....58
A1.4 #AAN....59
A1.5 $AA0....60
A1.6 $AA1....61
A1.7 $AA2....62
A1.8 $AA3....63
A1.9 $AA4....64
A1.10 $AA5VV....65
A1.11 $AA6....66
A1.12 $AA8V....67
A1.13 $AA9SCCCC....68
A1.14 $AA9S(data)....69
A1.15 $AAA....70
A1.16 $AAF....71
A1.17 $AAM....72
A1.18 ~**....73
A1.19 ~AA0....74
A1.20 ~AA1....75
A1.21 ~AA2....76
A1.22 ~AA3ETT....77
A1.23 ~AA4....78
A1.24 ~AA5....79
A1.25 ~AAO(name)....80
A1.26 $AA3....82
A1.28 $AA5....83
A1.28 $AA6(LO)(HI)....84
A1.29 $AA7(LO)(HI)....85
A1.30 $AAAV....86
A1.31 $AAB....87
A1.32 @AADI....88
A1.33 @AADO0D....89
A1.34 @AAEAT....90
A1.35 @AAHI(data)....91
A1.36 @AALO(data)....92
A1.37 @AADA....93
A1.38 @AACA....94
A1.39 @AARH....95
A1.40 @AARL....96
A1.41 @AARE....97
A1.42 @AACE....98
Operation Principles & Application Notes....99
INIT*_pin Operation Principle....99
Dual WatchDog Operation Principle....100
Temperature Measurement....102
CJC Offset Calibration....103
Command Response Time....104

8013

Introduction

SuperLogics' 8000 Series analog I/O modules measure voltage, current, temperature, pressure and various types of digital inputs. The modules themselves perform all conditioning and conversion functions, so that data can be transmitted as various types of data representation in ASCII format, directly to the PC via a serial port. All the modules are software programmable and require no DIP switch settings. Parameters such as address, baud rate, etc. are assigned via simple commands transmitted through the computer's serial port.

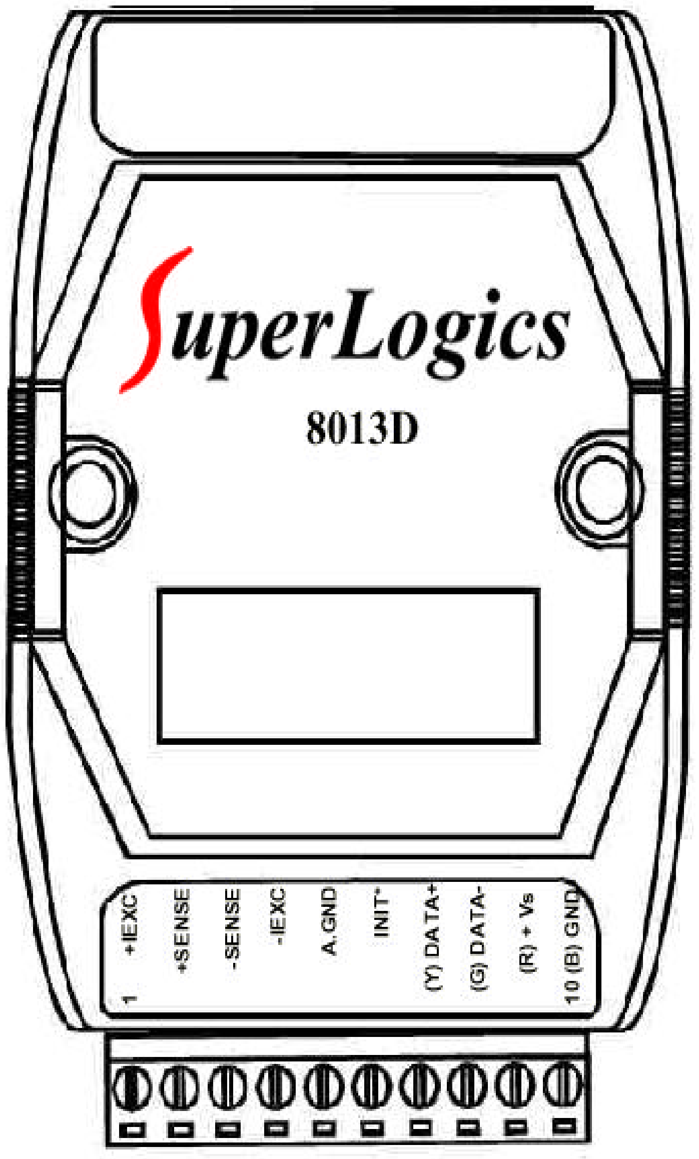
All 8000 series analog input modules use a microprocessor to control a 16-bit Sigma-Delta A/D to acquire analog signals. The 8017 and 8018 each have eight analog input channels, making these modules extremely cost-effective for industrial applications. The 8013D has one analog RTD input channel and is equipped with a 4 ½ digit LED window which can display single channel readings in real time.

More Information

Refer to chapter one of the 8520 manual for information on the following:

1.1 8000 Series Overview
1.2 8000 Common Features
1.3 8000 System Network Configuration
1.4 8000 Dimension

8013 Pin Assignment

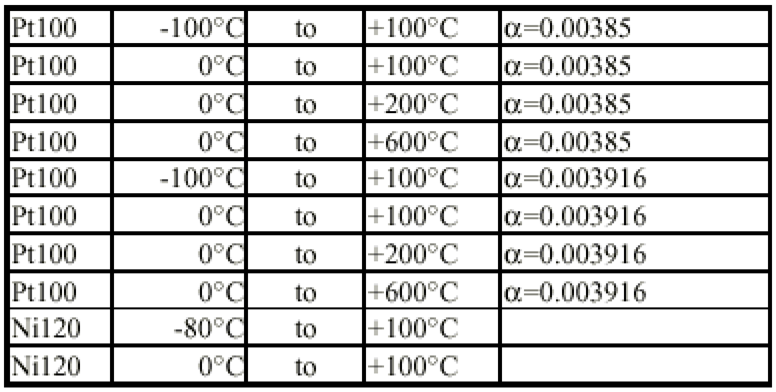


8013 Specifications

8013D: Single Channel RTD Input Module with LED display

Analog Input

- Channel: 1
- Input Type: Pt , Ni
- RTD temperature range



Pt100	-100°C	to	+100°C	α=0.00385
Pt100	0°C	to	+100°C	α=0.00385
Pt100	0°C	to	+200°C	α=0.00385
Pt100	0°C	to	+600°C	α=0.00385
Pt100	-100°C	to	+100°C	α=0.003916
Pt100	0°C	to	+100°C	α=0.003916
Pt100	0°C	to	+200°C	α=0.003916
Pt100	0°C	to	+600°C	α=0.003916
Ni120	-80°C	to	+100°C	
Ni120	0°C	to	+100°C	

- Sampling rate : 10 samples/sec
- Bandwidth : 4 Hz
- Wire connection: 2/3/4 wire
- Accuracy: ±0.05% or better
- Zero drift ：±0.3uV/°C
- CMR @ 50/60 Hz: 92 dB min
- NMR @50/60 Hz: 100 dB
- Span drift ：±25ppm/°C

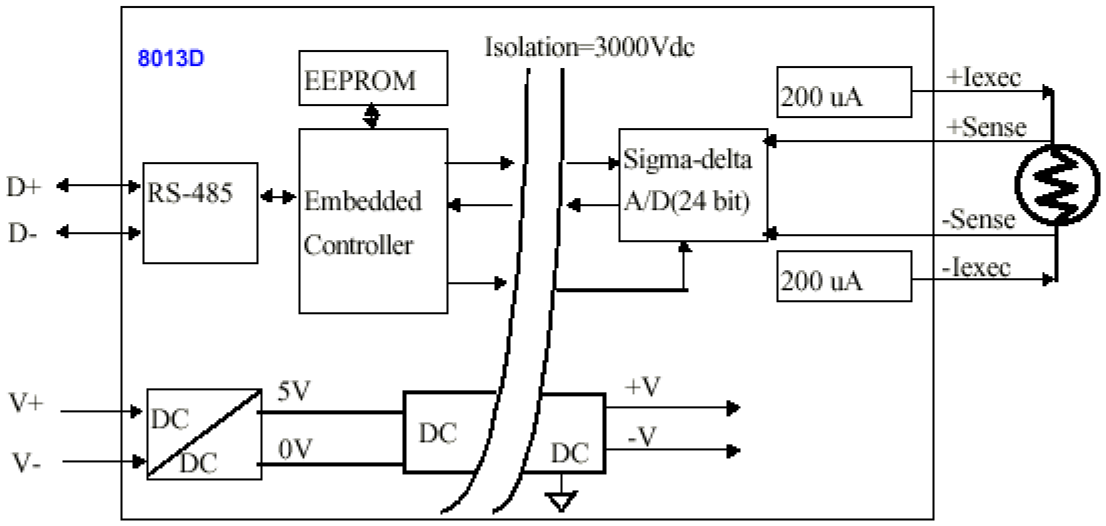
Display:

- LED: 4½ digit

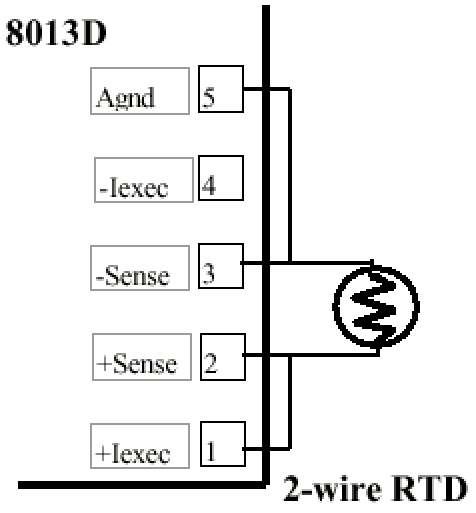
Power Consumption:

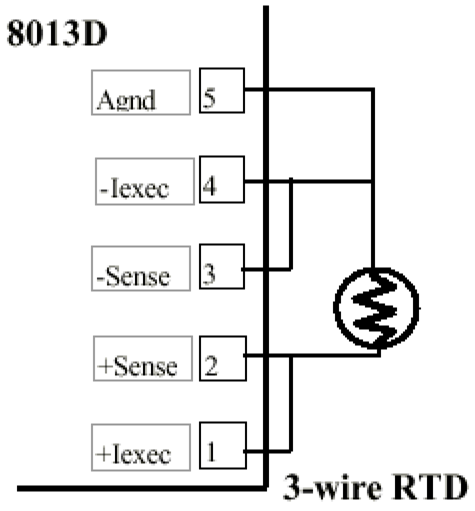
- 2.2W

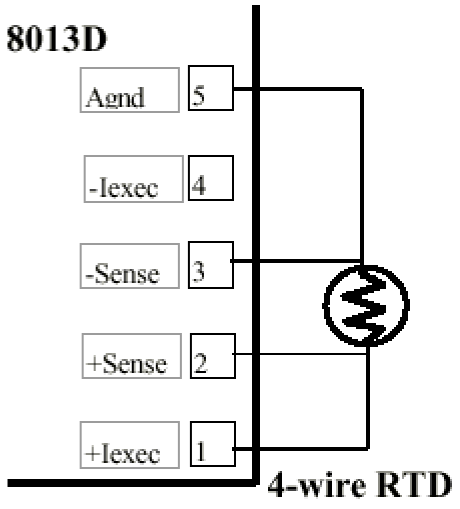
8013 Block Diagrams



8013 Application Wiring







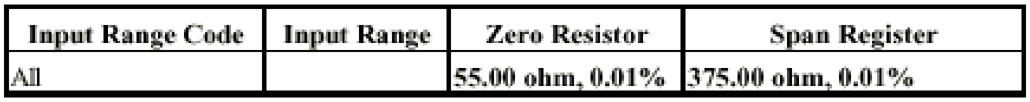
8013 Default Settings

The default settings for 8000 analog modules are:

- address=01, baud rate=9600, checksum disabled
- type=08=±10V input range (for 8017)
- type=05=±2.5V input range (for 8018)
- type=20=platinum, ±100°C(for 8013)
- data=1 start+8 data+1 stop(no parity)

8013 Calibration

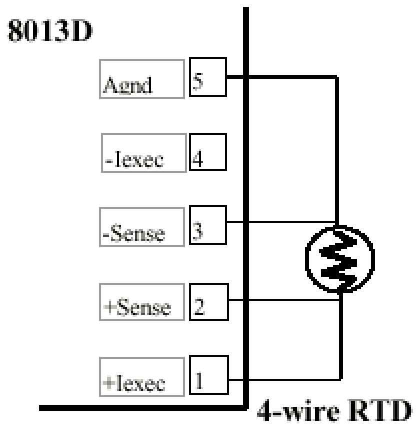
Zero/Span Table for 8013D Calibration.



Input Range Code	Input Range	Zero Resistor	Span Register
All		55.00 ohm, 0.01%	375.00 ohm, 0.01%

NOTE: One type calibrating is enough.

8013 Calibration



Step 1: Wire connection, install a stable voltage source to channel_0.
Step 2: Power-on, warm-up about 30 minutes
Step 3: Perform type-20 calibration

8013 Tables

Configuration Code Table : CC (for 8013D)

CC	Baud Rate
03	1200 BPS
04	2400 BPS
05	4800 BPS
06	9600 BPS
07	19200 BPS
08	38400 BPS
09	57600 BPS
0A	115200 BPS

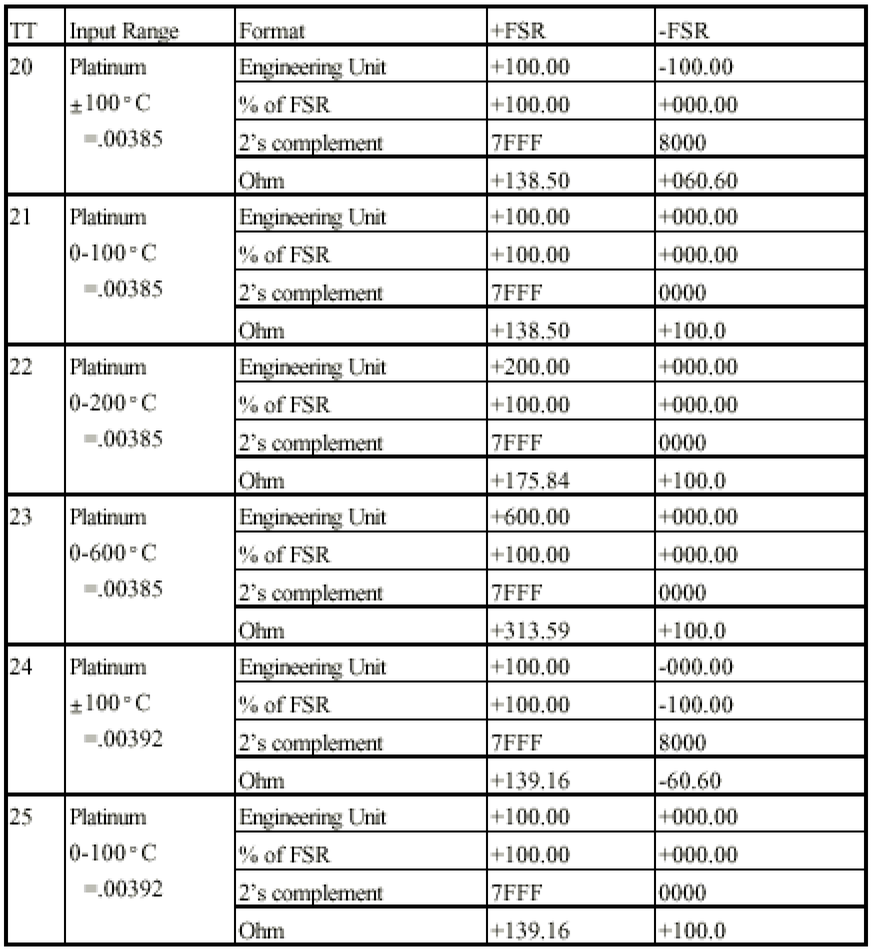
Configuration Code : FF, 2-char (for 8013D)

7	6	5	4	3	2	1	0
0	checksum 0=disable 1=enable	0				00: engineering unit 01: % of FSR 10: 2's complement of hexadecimal 11: Ohms (for 8013)	

Configuration Code Table: TT (for 8013D)

TT	Input Range
20	P.,±100 ° C, =.00385
21	P.,0-100 ° C, =.00385
22	P.,0-200 ° C, =.00385
23	P.,0-600 ° C, =.00385
24	P.,±100 ° C, =.003916
25	P.,0-100 ° C, =.003916
26	P.,0-200 ° C, =.003916
27	P.,0-600 ° C, =.003916
28	N.,-80 ° C to 100 ° C
29	N.,0 ° C to 100 ° C

Data Format Table : TT (for 8013D)



TT	Input Range	Format	+FSR	-FSR
20	Platinum ±100 °C =.00385	Engineering Unit	+100.00	-100.00
		% of FSR	+100.00	+000.00
		2's complement	7FFF	8000
		Ohm	+138.50	+060.60
21	Platinum 0-100 °C =.00385	Engineering Unit	+100.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement	7FFF	0000
		Ohm	+138.50	+100.0
22	Platinum 0-200 °C =.00385	Engineering Unit	+200.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement	7FFF	0000
		Ohm	+175.84	+100.0
23	Platinum 0-600 °C =.00385	Engineering Unit	+600.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement	7FFF	0000
		Ohm	+313.59	+100.0
24	Platinum ±100 °C =.00392	Engineering Unit	+100.00	-000.00
		% of FSR	+100.00	-100.00
		2's complement	7FFF	8000
		Ohm	+139.16	-60.60
25	Platinum 0-100 °C =.00392	Engineering Unit	+100.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement	7FFF	0000
		Ohm	+139.16	+100.0

Data Format Table : TT (for 8013D, continued)

26	Platinum 0-200°C =.00392	Engineering Unit	+200.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement	7FFF	0000
		Ohm	+177.13	+100.0
27	Platinum 0-600°C =.00392	Engineering Unit	+600.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement	7FFF	0000
		Ohm	+317.28	+100.0
28	120 ohm Nickel -80-100°C	Engineering Unit	+100.00	-080.00
		% of FSR	+100.00	+080.00
		2's complement	7FFF	999A
		Ohm	+200.64	+066.60
29	120 ohm Nickel 0-100°C	Engineering Unit	+100.00	+000.00
		% of FSR	+100.00	+000.00
		2's complement	7FFF	0000
		Ohm	+200.64	+120.00

Command Set Table

Command	Response	Description	Reference
%AANNTTCCFF	!AA	Set module configuration	A1.1
#**	No Response	Synchronized Sampling	A1.2
#AA	>(data)	Read analog input	A1.3
#AAN	>(data)	Read analog input from channel_N	A1.4
$AA0	!AA	Perform span calibration	A1.5
$AA1	!AA	Perform zero calibration	A1.6
$AA2	!AATTCCFF	Read configuration	A1.7
$AA3	!AA(data)	Read CJC value	A1.8
$AA4	!AA(data)	Read Synchronized Data	A1.9
$AA5VV	!AA	Enable/disable channel multiplexing	A1.10
$AA6	!AAVV	Read channel multiplexing status	A1.11
$AA8V	!AA	Select Led Configuration	A1.12
$AA9SCCCC	!AA	Set CJC Offset Value	A1.13
$AA9S(data)	!AA	Send Led Display	A1.14
$AAA	>(data)*8	Read all 8 channel data	A1.15
$AAF	!AA(data)	Read the firmware version number	A1.16
$AAM	!AA(data)	Read the module name	A1.17
~**	No Response	Host OK Sec. 2.18	A1.18
~AA0	!AASS	Read Module Status	A1.19
~AA1	!AA	Reset Module Status	A1.20
~AA2	!AATT	Read Host Watchdog Timer Value	A1.21
~AA3ETT	!AA	Enable Host Watchdog Timer	A1.22
~AAO(name)	!AA	Set module name	A1.25

8013D Command Set Table

Command	Response	Description	Reference
%AANNTTCCFF	!AA	Set module configuration	A1.1
#**	No Response	Synchronized Sampling	A1.2
#AA	>(data)	Read analog input	A1.3
$AA0	!AA	Perform span calibration	A1.5
$AA1	!AA	Perform zero calibration	A1.6
$AA2	!AATTCCFF	Read configuration	A1.7
$AA4	!AA(data)	Read Synchronized Data	A1.8
$AA8V	!AA	Select Led Configuration	A1.12
$AA9S(data)	!AA	Send Led Display	A1.14
$AAF	!AA(data)	Read the firmware version number	A1.16
$AAM	!AA(data)	Read the module name	A1.17
~**	No Response	Host OK	A1.18
~AA0	!AASS	Read Module Status	A1.19
~AA1	!AA	Reset Module Status	A1.20
~AA2	!AASTT	Read Host Watchdog Timer Value	A1.21
~AA3ETT	!AA	Enable Host Watchdog Timer	A1.22
~AAO(name)	!AA	Set module name	A1.25

8014D

Introduction

SuperLogics's 8014D is a single channel analog input module with two digital output channels and one digital input channel. The module has a 4½ digit LED window which can display single channel readings in real-time. The 8014D also has an event counter that can be used to count up to 65,356 transitions occurring on the digital input channel. The event counter may be read and cleared by the host computer. It can also be used in production lines to keep a record of repetitious operations. The digital outputs are open-collector transistor switches that can be controlled by the host computer. These switches may be used to control relays which may control pumps, heaters, monitors, etc. To protect the module, 3000V isolation is provided.

FEATURES

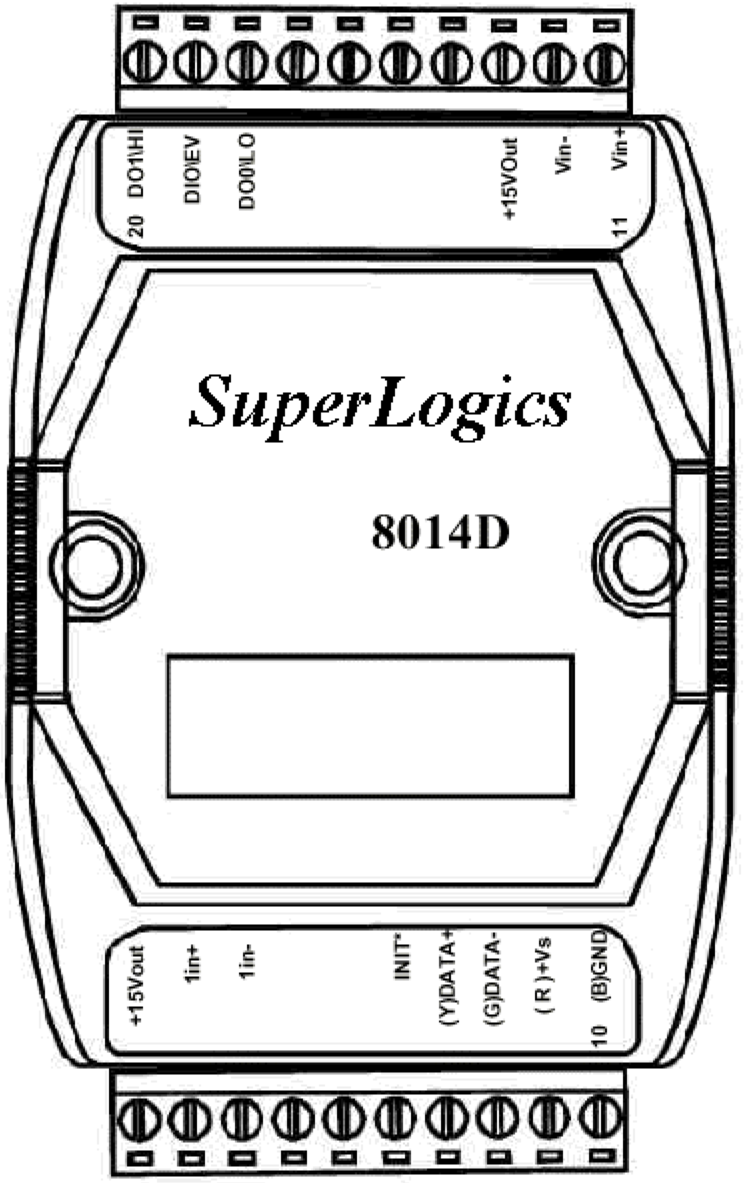
- 24 bits sigma-delta A/D converter to provide 16 bit precision.
- One Analog Input Channel
- One Digital Input Channel (can be used as an event counter)
- Two Digital Output Channels
- Input range is programmable.
- Software Calibration
- 4½ digit LED display
- Isolated loop power, +15V, 20 mA max.
- Linear mapping function
- Built-in 125˜ 0.1% resistor for current measurement

More Information

Refer to chapter one of the 8520 manual for information on the following:

1.1 8000 Series Overview
1.2 8000 Common Features
1.3 8000 System Network Configuration
1.4 8000 Dimension

8014D Pin Assignment



8014D Specifications

Analog Input

- Channels: 1
- Type: mV, V , mA
- Voltage range: ±150mA, ±500mV, ±1V, ±5V, ±10V
- Current range: ± 20mA
- Sampling rate: 10 samples/sec
- Bandwidth: 4 Hz
- Accuracy: ±0.05% or better
- Zero drift: ±6uV/°C
- Span drift：±25PPm/°C
- CMR @ 50/60 Hz: 150 dB
- NMR @50/60 Hz: 100 dB
- Over voltage protection: ±10V
- Isolated loop power: 15VDC @ 30mA (QTM-8014D)

Digital Input

- Channel: 1
- Logic 0: 0 to 1V, Logic 1: 3.5V to 30V
- Input frequency: 50Hz max.
- Input pulse width: 1ms min.

Digital Output

- Channels: 2
- Open collector to 30V, 30mA load max.
- Power dissipation: 300mW

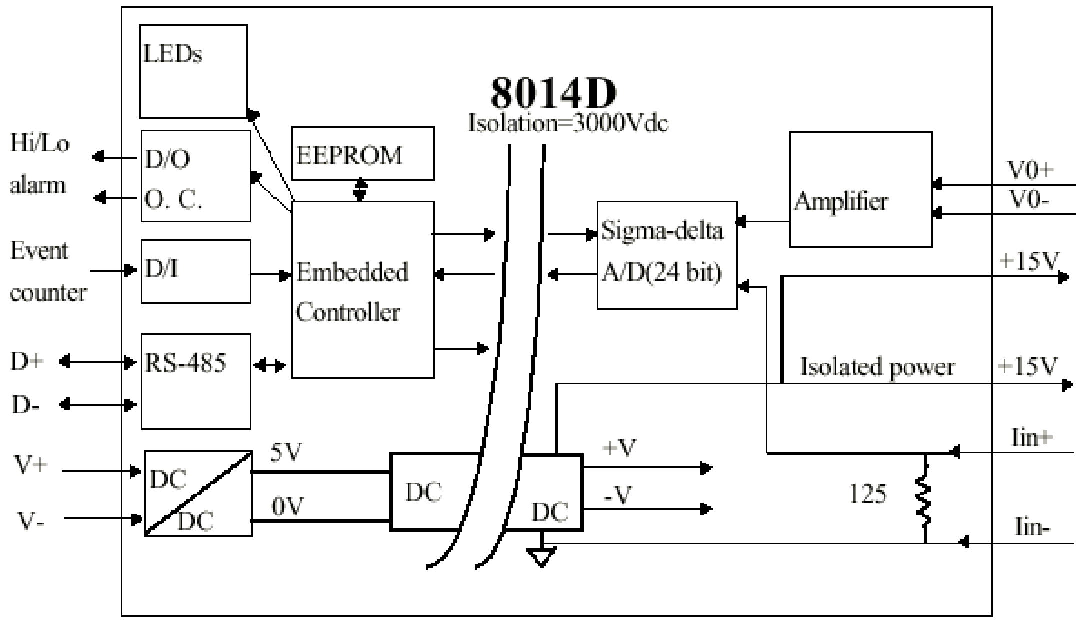
Display:

- LED: 4½ digit

Power consumption:

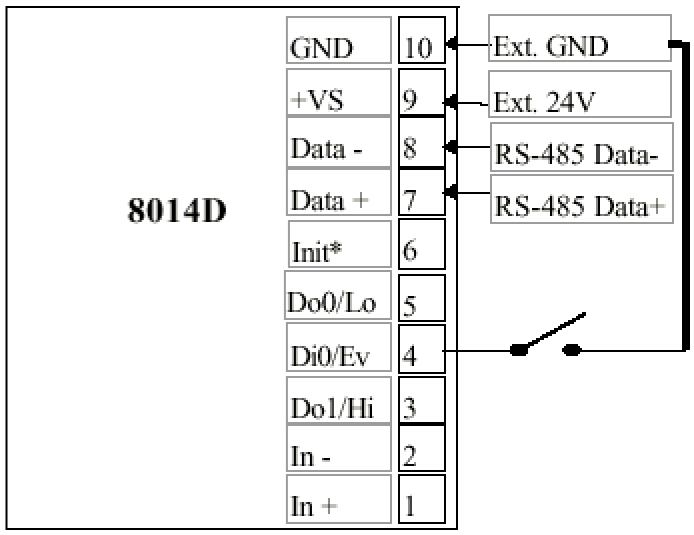
- 2.2W

8014D Block Diagrams

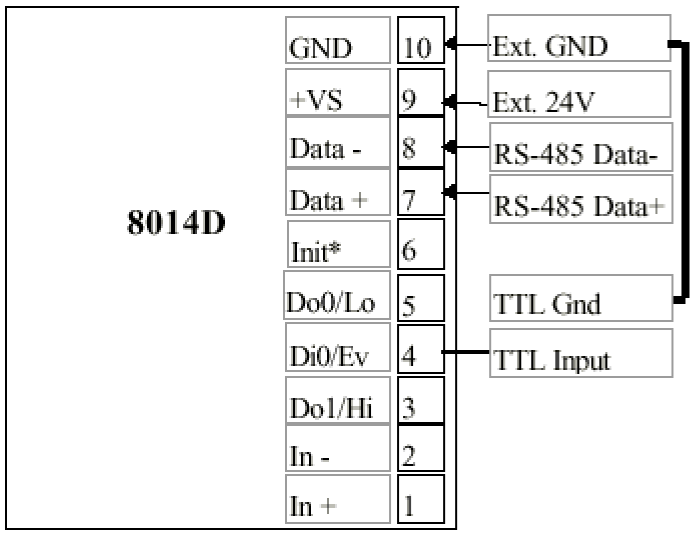


8014D Application Wiring

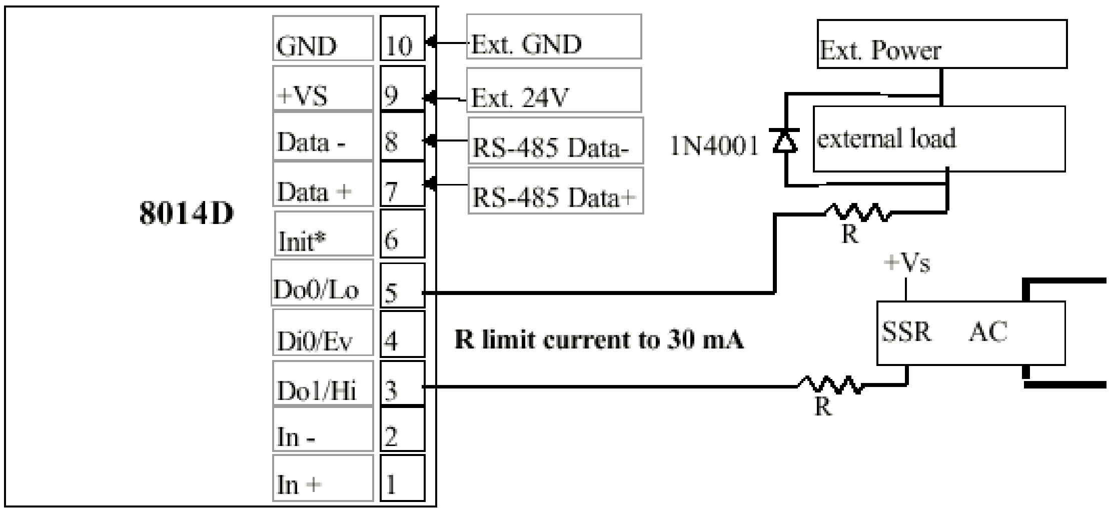
Dry Contact Input



TTL Input

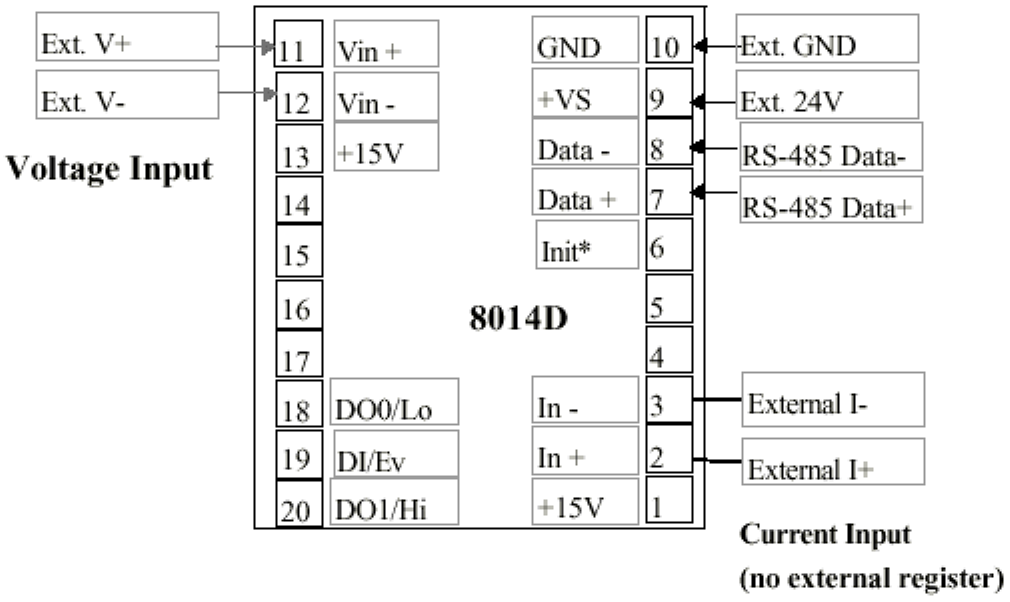


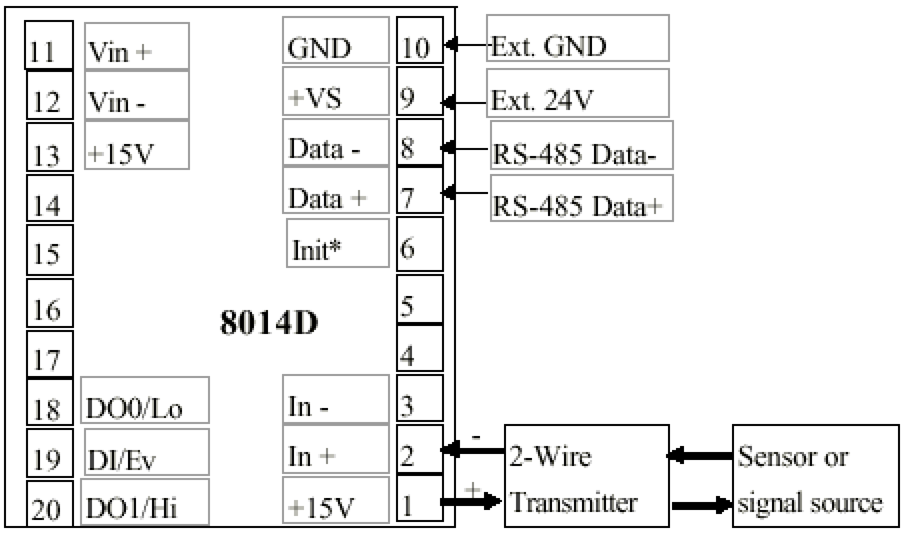
Output Drive SSR & Load



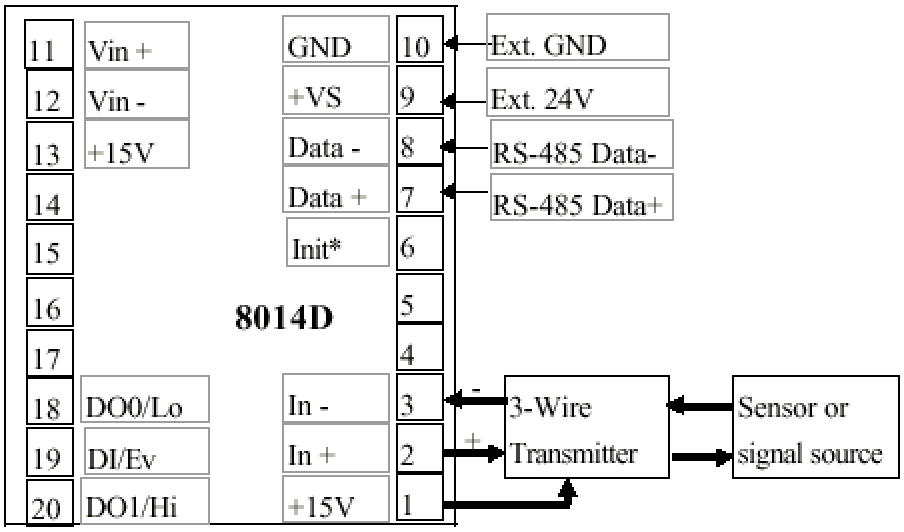
Note:
- If external load is resistive, the IN4001 can be omitted. (transistor, lamp, resistor,…)
- If external load is inductive, the IN4001 can't be omitted. (relay, coil,…)

8014D Analog Input

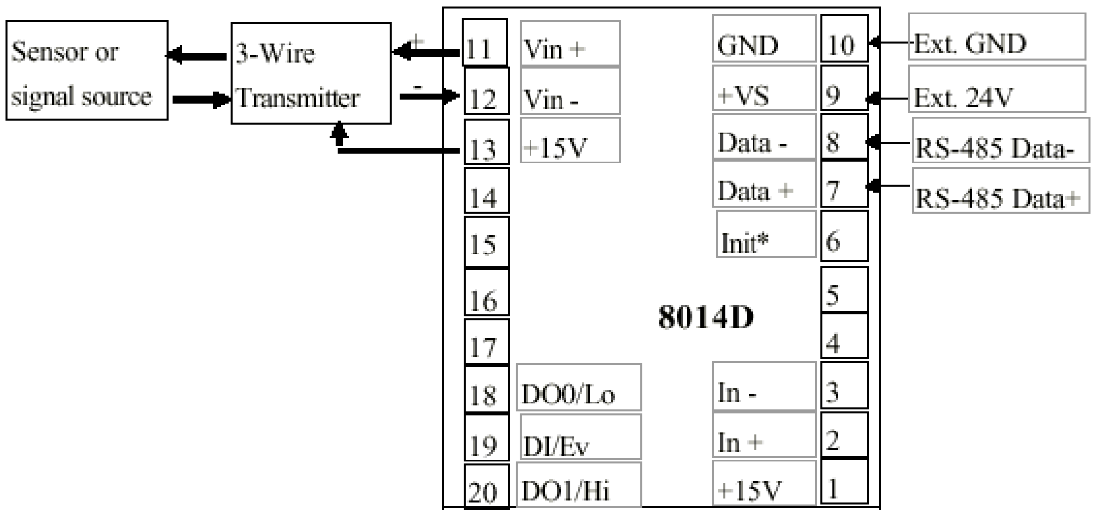




2-wire transmitter current input

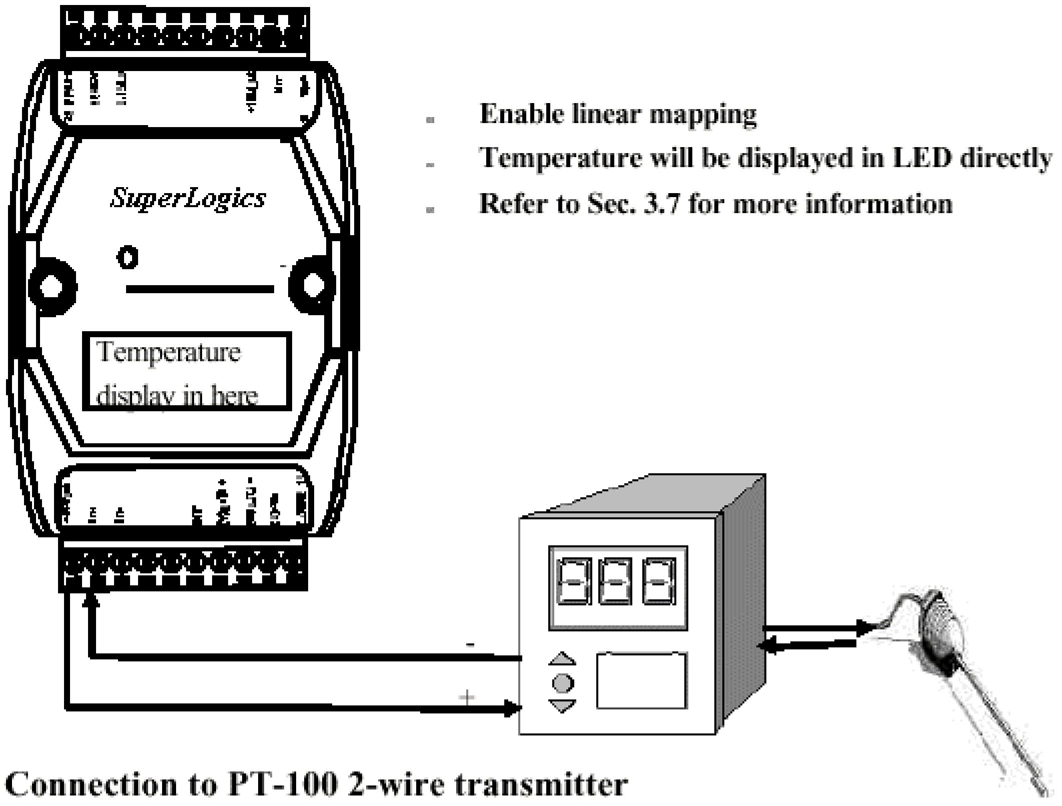


3-wire transmitter current input



3-wire transmitter voltage input

- Enable linear mapping
- Temperature will be displayed in LED directly
- Refer to Sec. 3.7 for more information



Connection to PT-100 2-wire transmitter

8014D Default Settings

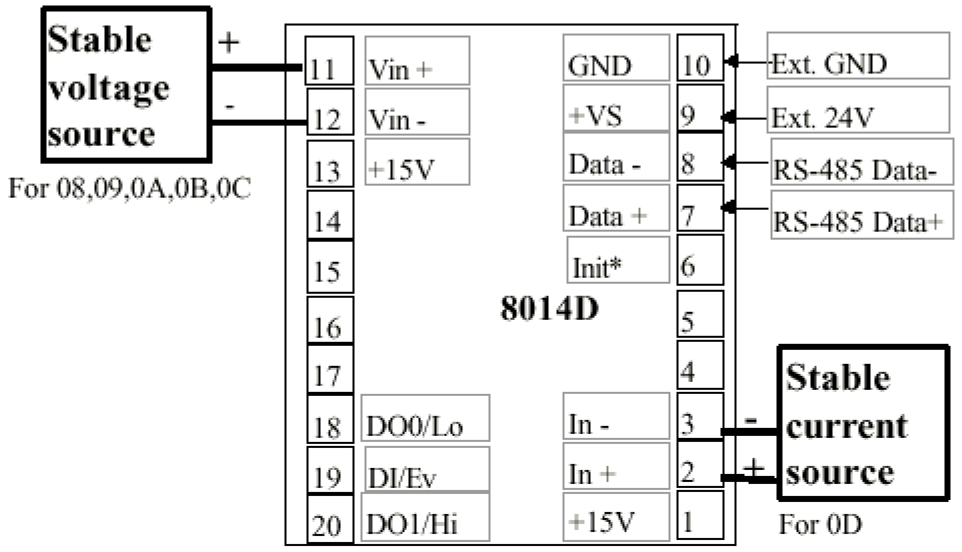
Default settings for the 8014D are as follows:

- address=01
- baud rate=9600
- checksum: disable
- data=1 start+8 data+1 stop(no parity)
- type 08=±10V input range

8014D Calibration

Zero/Span Table for QTM-8014D Calibration.

Input Range Code	Input Range	Zero Voltage	Span Voltage
08	±10V	0V	10V
09	±5V	0V	5V
0A	±1V	0V	1V
0B	±500mV	0V	500mV
0C	±150mV	0V	150mV
0D	±20mA	0mA	20mA



Step 1: Wire connection, install a stable voltage source to channel_0. Step 2: Power-on, warm-up about 30 minutes Step 3: Perform type-08 calibration Step 4: Perform type-09 calibration .. Step 8: Perform type-0D calibration

NOTE: calibration steps are all the same for type-08 to type-0D. Only the SPAN-Voltage is different.

8014D Tables

Configuration Code Table: CC

CC	Baud Rate
03	1200 BPS
04	2400 BPS
05	4800 BPS
06	9600 BPS
07	19200 BPS
08	38400 BPS
09	57600 BPS
0A	115200 BPS

Configuration Code: FF, 2-char

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	checksum 0=disable 1=enable	0				00: engineering unit 01: % of FSR 10: 2's complement of hexadecimal 11: reserved	

Configuration Code Table: TT

TT	Input Range
08	+/- 10V
09	+/- 5V
0A	+/- 1V
0B	+/- 500mV
0C	+/- 150mV
0D	+/- 20mA

Data Format Table (data):

TT	Range	Format	+FSR	Zero	-FSR
08	±10V	Engineering Unit	+10.000	±00.000	-10.000
		% of FSR	+100.00	±000.00	-100.00
		2's complement	7FFF	0000	8000
09	±5V	Engineering Unit	+5.0000	±0.0000	-5.0000
		% of FSR	+100.00	±000.00	-100.00
		2's complement	7FFF	0000	8000
0A	±1V	Engineering Unit	+1.0000	±0.0000	-1.0000
		% of FSR	+100.00	±000.00	-100.00
		2's complement	7FFF	0000	8000
0B	±500mV	Engineering Unit	+500.00	±000.00	-500.00
		% of FSR	+100.00	±000.00	-100.00
		2's complement	7FFF	0000	8000
0C	±150mV	Engineering Unit	+150.00	±000.00	-150.00
		% of FSR	+100.00	±000.00	-100.00
		2's complement	7FFF	0000	8000
0D	±20mA	Engineering Unit	+20.000	±00.000	-20.000
		% of FSR	+100.00	±000.00	-100.00
		2's complement	7FFF	0000	8000

8014D Command Set Table

Command	Response	Description	Reference
%AANNTTCCFF	!AA	Set module configuration	A1.1
#**	No Response	Synchronized Sampling	A1.2
#AA	>(data)	Read analog input	A1.3
$AA0	!AA	Perform span calibration	A1.5
$AA1	!AA	Perform zero calibration	A1.6
$AA2	!AATTCCFF	Read configuration	A1.7
$AA3	!AA(LO)(HI)	Read source linear mapping	A1.8
$AA4	!AA(data)	Read Synchronized Data	A1.9
$AA5	!AA(LO)(HI)	Read target linear mapping	A1.27
$AA6(LO)(HI)	!AA	Write source linear mapping	A1.28
$AA7(LO)(HI)	!AA	Write target linear mapping	A1.29
$AA8V	!AA	Select LED Configuration	A1.12
$AA9S(data)	!AA	Send LED Display	A1.14
$AAAV	!AA	Enable/Disable Linear Mapping	A1.30
$AAB	!AAS	Linear Mapping Status	A1.31
$AAF	!AA(data)	Read firmware number	A1.16
$AAM	!AA(data)	Read the module name	A1.17
@AADI	!AAS0D0I	Read DIO & alarm status	A1.32
@AADO0D	!AA	Set D/O	A1.33
@AAEAT	!AA	Enable alarm	A1.34
@AAHI(data)	!AA	Set high alarm	A1.35
@AALO(data)	!AA	Set low alarm	A1.36
@AADA	!AA	Disable alarm	A1.37
@AACA	!AA	Clear latch alarm	A1.38
@AARH	!AA(data)	Read high alarm	A1.39
@AARL	!AA(data)	Read low alarm	A1.40
@AARE	!AA(data)	Read event counter	A1.41
@AACE	!AA	Clear event counter	A1.42
~**	No Response	Host OK	A1.18
~AA0	!AASS	Read Module Status	A1.19
~AA1	!AA	Reset Module Status	A1.20
~AA2	!AATT	Read Host Watchdog Timer	A1.21
~AA3ETT	!AA	Enable Host Watchdog Timer	A1.22
~AA4	!AAVV00	Read power-on/safe value	A1.23
~AA5	!AA	Set power-on/safe value	A1.24
~AAO(name)	!AA	Set module name	A1.25

8017

Introduction

SuperLogics' 8000 Series analog I/O modules measure voltage, current, temperature, pressure and various types of digital inputs. The modules themselves perform all conditioning and conversion functions, so that data can be transmitted as various types of data representation in ASCII format, directly to the PC via a serial port. All the modules are software programmable and require no DIP switch settings. Parameters such as address, baud rate, etc. are assigned via simple commands transmitted through the computer's serial port.

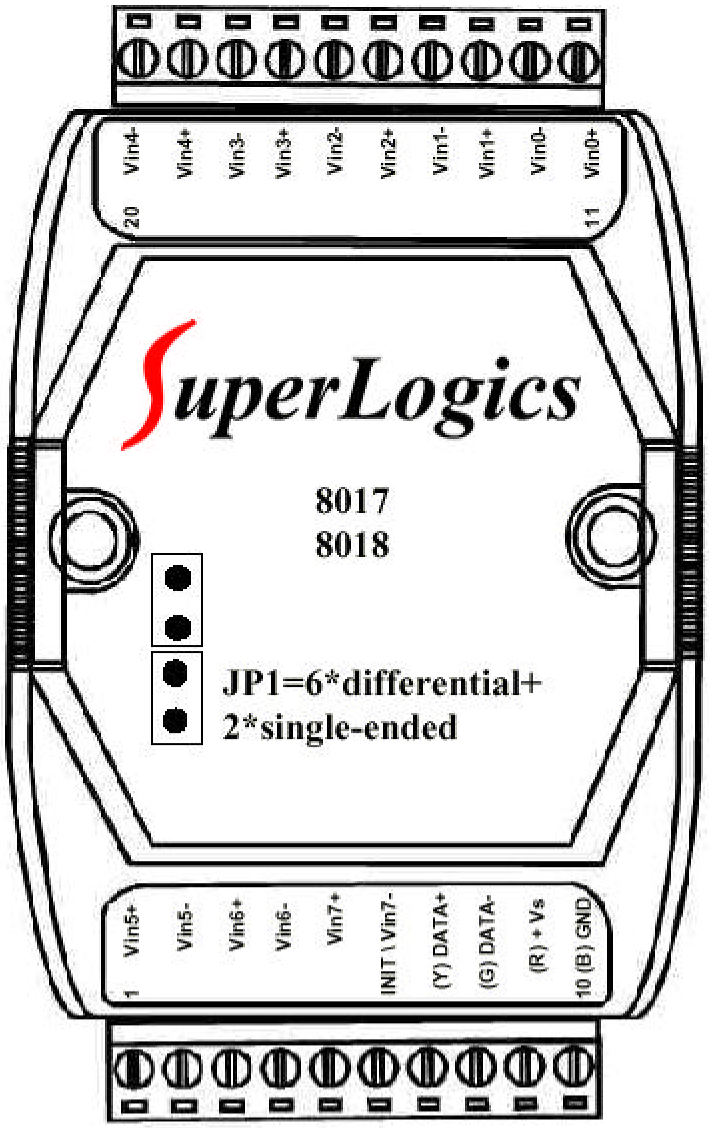
All 8000 series analog input modules use a microprocessor to control a 16-bit Sigma-Delta A/D to acquire analog signals. The 8017 and 8018 each have eight analog input channels, making these modules extremely cost-effective for industrial applications. The 8013D has one analog RTD input channel and is equipped with a 4 ½ digit LED window which can display single channel readings in real time.

1.1 More Information

Refer to chapter one of the 8520 manual for information on the following:

1.1 8000 Series Overview
1.2 8000 Common Features
1.3 8000 System Network Configuration
1.4 8000 Dimension

8017 Pin Assignment



8017 Specifications

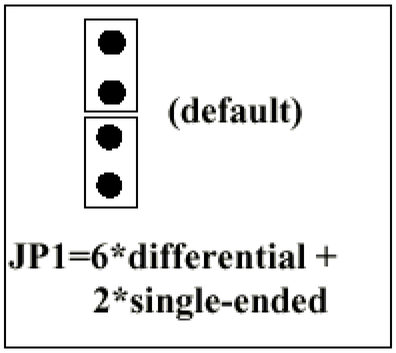
8017: 8 Channel Analog Input Module

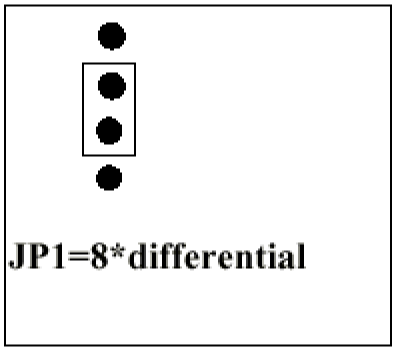
Analog Input

- Channels: 6 differential + 2 single-ended or 8 differential (selected by JP1)
- Input type: mV, V , mA
- Input range: ±150mV, ±500mV, ±1V, ±5V, ± 10V and ±20mA
- Sample rate: 10 sample/sec(total)
- Bandwidth: 13.1 Hz
- Accuracy: ±0.1% or better
- Zero drift: ±0.03uV/°C
- Span drift: ±25ppm/°C
- CMR @ 50/60 Hz : 92 dB min
- Over voltage protection: ±35V

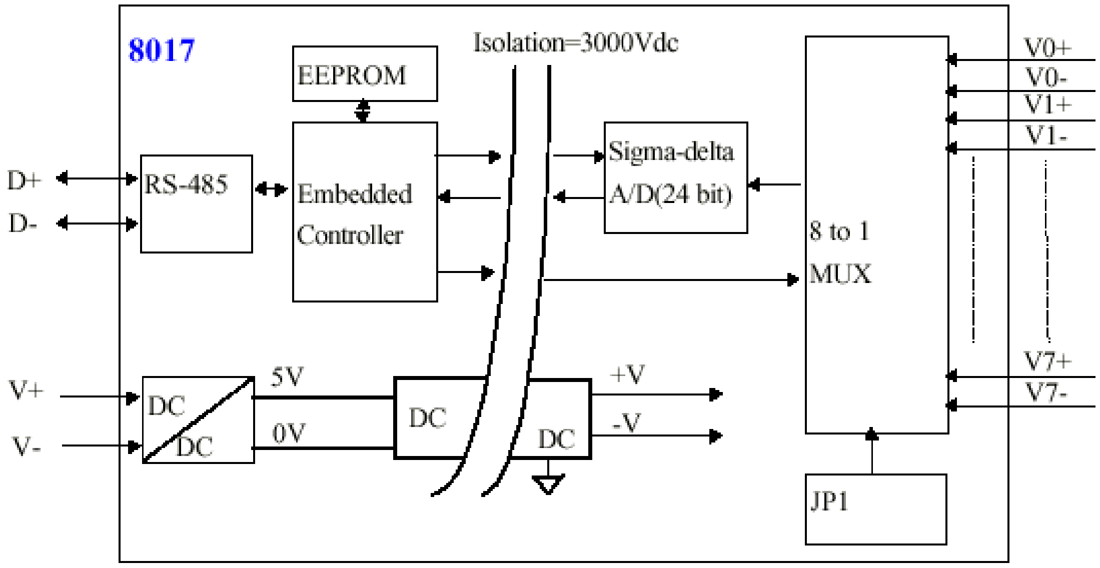
Power:

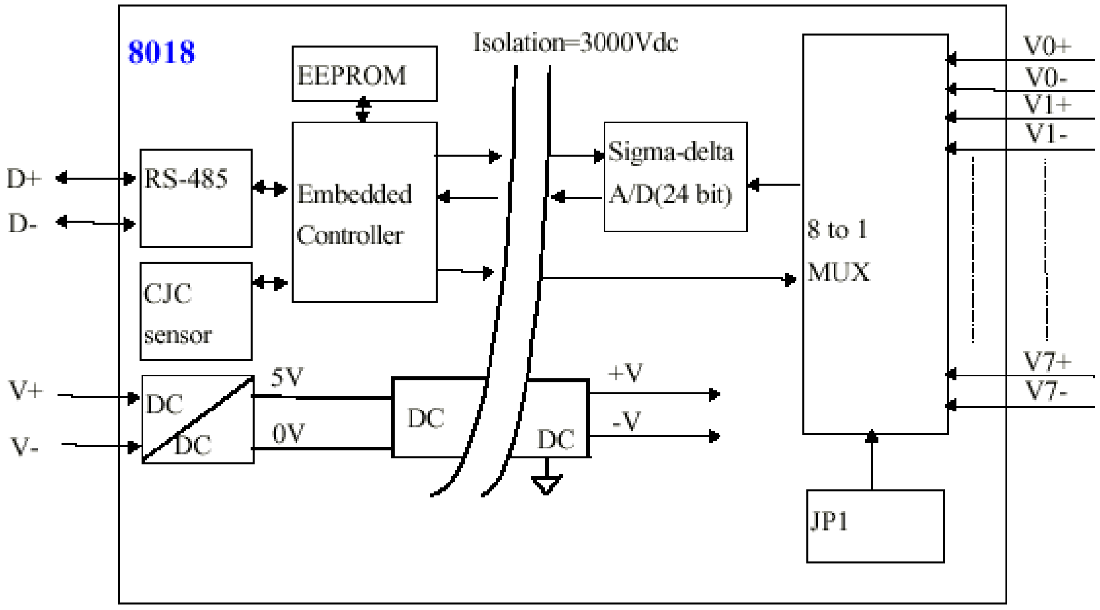
- Power consumption: 2W





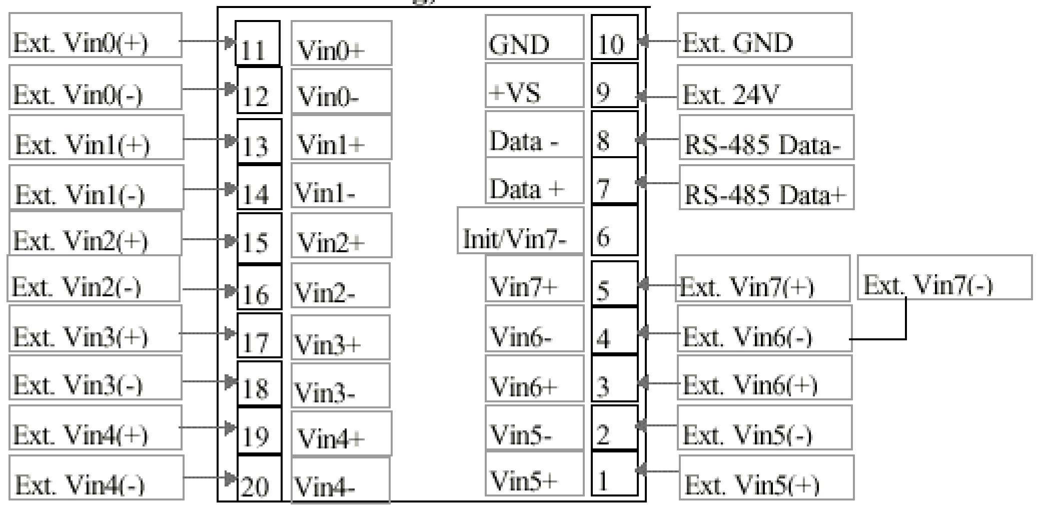
8017 Block Diagrams





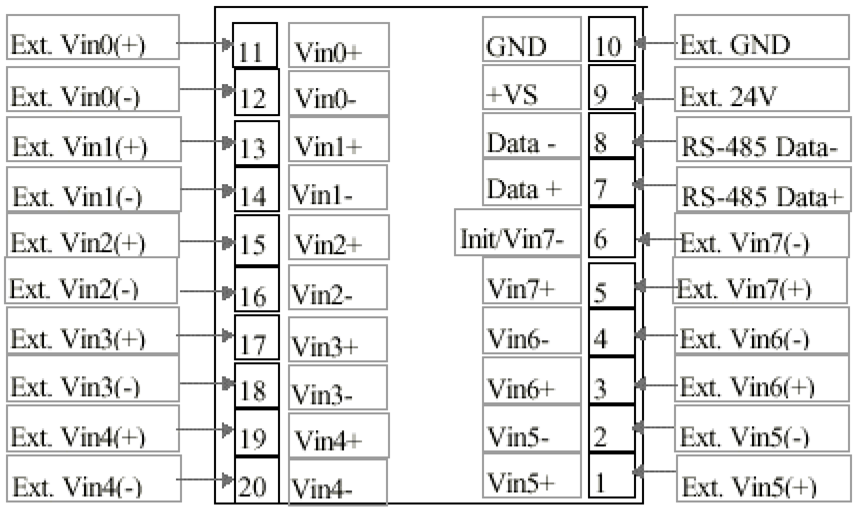
8017 Application Wiring

Where JP1 is set for 6*differential + 2*single-ended inputs (refer to Sec. 1.3 for JP1 setting):



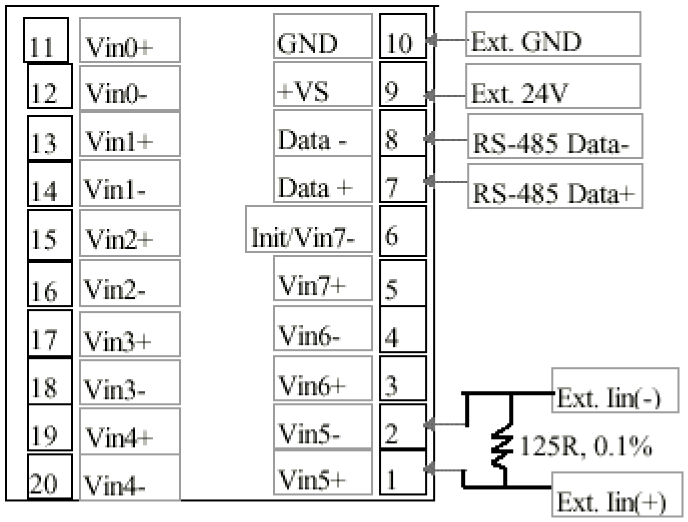
8017 & 8018

Where JP1 is used to select 8*differential inputs (refer to Sec. 1.3 for JP1 setting):



8017 & 8018

Current Measurement



8017 & 8018

8017 Default Settings

The default settings for 8000 analog modules are:

- address=01, baud rate=9600, checksum disabled
- type=08=±10V input range (for 8017)
- type=05=±2.5V input range (for 8018)
- type=20=platinum, ±100°C(for 8013)
- data=1 start+8 data+1 stop(no parity)

NOTE:

On the 8017 and 8018, JP1 is used to select analog input channels as either 6 differential and 2 single-ended or 8 differential. The default setting is for the 6/2 combination. (See section 1.3 for more information).

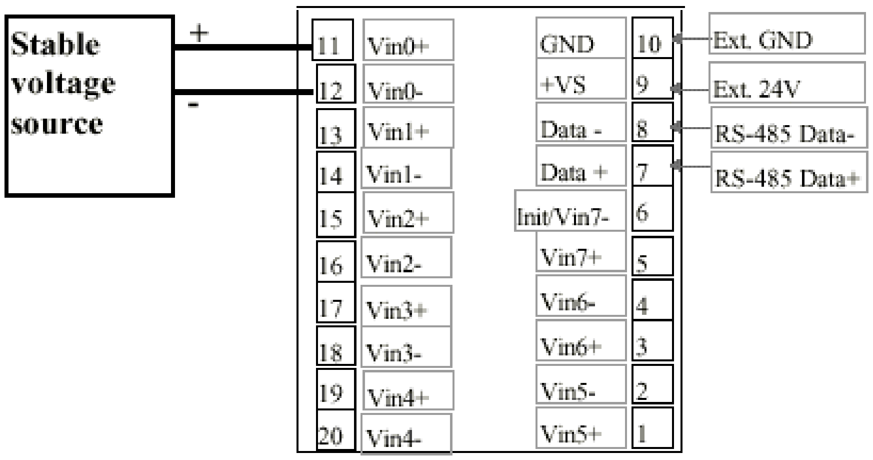
8017 Calibration

Zero/Span Table for 8017 Calibration.

Input Range Code	Input Range	Zero Voltage	Span Voltage
08	±10V	0V	10V
09	±5V	0V	5V
0A	±1V	0V	1V
0B	±500mV	0V	500mV
0C	±150mV	0V	150mV
0D	±20mA	0V or 0mA with 125 0.1%	2.5V or 20mA with 125 0.1%

NOTE: One type calibrating is enough.

8017 Calibration



Step 1: Wire connection, install a stable voltage source to channel_0.
Step 2: Power-on, warm-up about 30 minutes
Step 3: Perform type-08 calibration
Step 4: Perform type-09 calibration
..
Step 8: Perform type-0D calibration

8017 Tables

Configuration Code Table : CC (for 8017)

CC	Baud Rate
03	1200 BPS
04	2400 BPS
05	4800 BPS
06	9600 BPS
07	19200 BPS
08	38400 BPS
09	57600 BPS
0A	115200 BPS

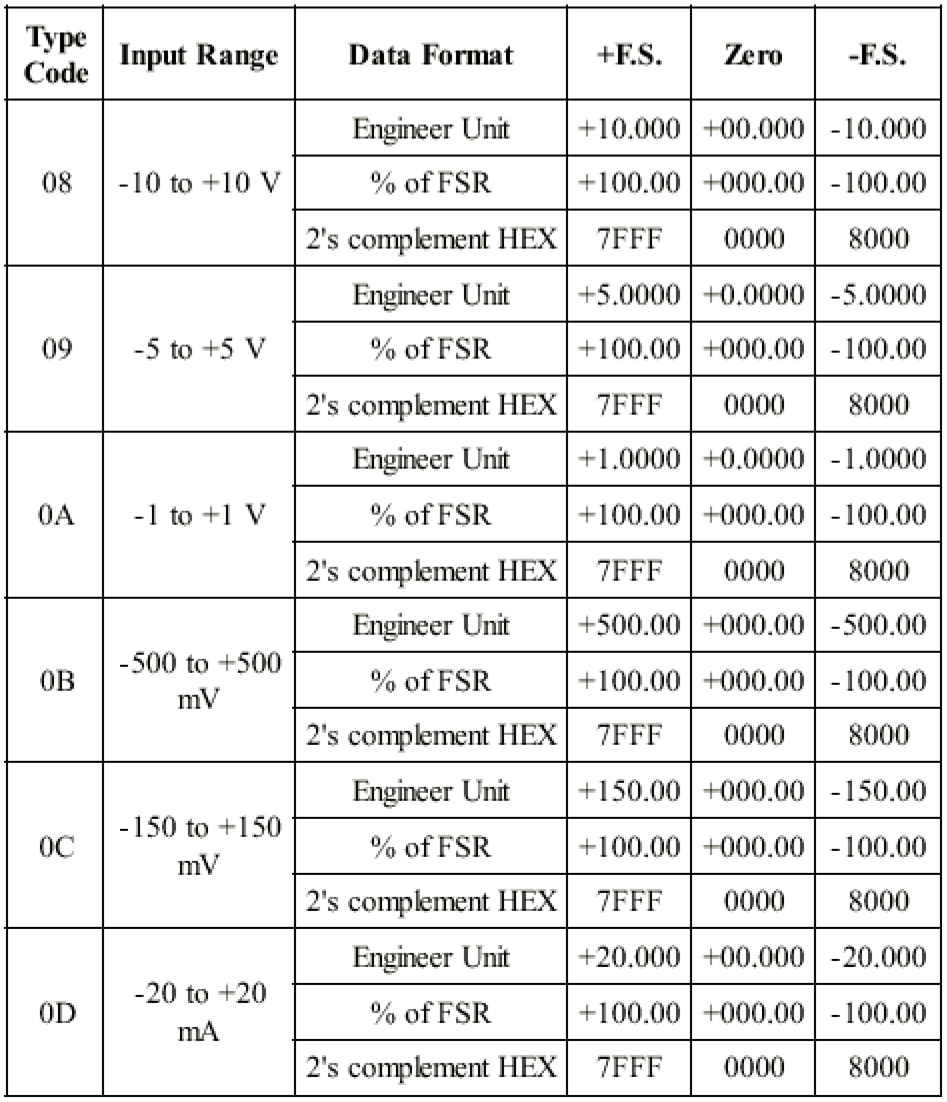
Configuration Code : FF, 2-char (for 8017)

7	6	5	4	3	2	1	0
0	checksum 0=disable 1=enable	0				00: engineering unit 01: % of FSR 10: 2's complement of hexadecimal 11: Ohms (for 8013)	

Configuration Code Table : TT (for 8017)

TT	Input Range
08	+/- 10V
09	+/- 5V
0A	+/- 1V
0B	+/- 500mV
0C	+/- 150mV
0D	+/- 20mA

Data Format Table (data): (for 8017)



Type Code	Input Range	Data Format	+F.S.	Zero	-F.S.
08	-10 to +10 V	Engineer Unit	+10.000	+00.000	-10.000
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
09	-5 to +5 V	Engineer Unit	+5.0000	+0.0000	-5.0000
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
0A	-1 to +1 V	Engineer Unit	+1.0000	+0.0000	-1.0000
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
0B	-500 to +500 mV	Engineer Unit	+500.00	+000.00	-500.00
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
0C	-150 to +150 mV	Engineer Unit	+150.00	+000.00	-150.00
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
0D	-20 to +20 mA	Engineer Unit	+20.000	+00.000	-20.000
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000

Command Set Table

Command	Response	Description	Reference
%AANNTTCCFF	!AA	Set module configuration	A1.1
#**	No Response	Synchronized Sampling	A1.2
#AA	>(data)	Read analog input	A1.3
#AAN	>(data)	Read analog input from channel_N	A1.4
$AA0	!AA	Perform span calibration	A1.5
$AA1	!AA	Perform zero calibration	A1.6
$AA2	!AATTCCFF	Read configuration	A1.7
$AA3	!AA(data)	Read CJC value	A1.8
$AA4	!AA(data)	Read Synchronized Data	A1.9
$AA5VV	!AA	Enable/disable channel multiplexing	A1.10
$AA6	!AAVV	Read channel multiplexing status	A1.11
$AA8V	!AA	Select Led Configuration	A1.12
$AA9SCCCC	!AA	Set CJC Offset Value	A1.13
$AA9S(data)	!AA	Send Led Display	A1.14
$AAA	>(data)*8	Read all 8 channel data	A1.15
$AAF	!AA(data)	Read the firmware version number	A1.16
$AAM	!AA(data)	Read the module name	A1.17
~**	No Response	Host OK Sec. 2.18	A1.18
~AA0	!AASS	Read Module Status	A1.19
~AA1	!AA	Reset Module Status	A1.20
~AA2	!AATT	Read Host Watchdog Timer Value	A1.21
~AA3ETT	!AA	Enable Host Watchdog Timer	A1.22
~AAO(name)	!AA	Set module name	A1.25

8017 Command Set Table

Command	Response	Description	Reference
%AANNTTCCFF	!AA	Set module configuration	A1.1
#AAN	>(data)	Read analog input from channel_N	A1.4
$AA0	!AA	Perform span calibration	A1.5
$AA1	!AA	Perform zero calibration	A1.6
$AA2	!AATTCCFF	Read configuration	A1.7
$AA5VV	!AA	Enable/disable channel multiplexing	A1.10
$AA6	!AAVV	Read channel multiplexing status	A1.11
$AAA	>(data)*8	Read all 8 channel data	A1.15
$AAF	!AA(data)	Read the firmware version number	A1.16
$AAM	!AA(data)	Read the module name	A1.17
~**	No Response	Host OK	A1.18
~AA0	!AASS	Read Module Status	A1.19
~AA1	!AA	Reset Module Status	A1.20
~AA2	!AATT	Read Host Watchdog Timer Value	A1.21
~AA3ETT	!AA	Enable Host Watchdog Timer	A1.22
~AAO(name)	!AA	Set module name	A1.25

8018

Introduction

SuperLogics' 8000 Series analog I/O modules measure voltage, current, temperature, pressure and various types of digital inputs. The modules themselves perform all conditioning and conversion functions, so that data can be transmitted as various types of data representation in ASCII format, directly to the PC via a serial port. All the modules are software programmable and require no DIP switch settings. Parameters such as address, baud rate, etc. are assigned via simple commands transmitted through the computer's serial port.

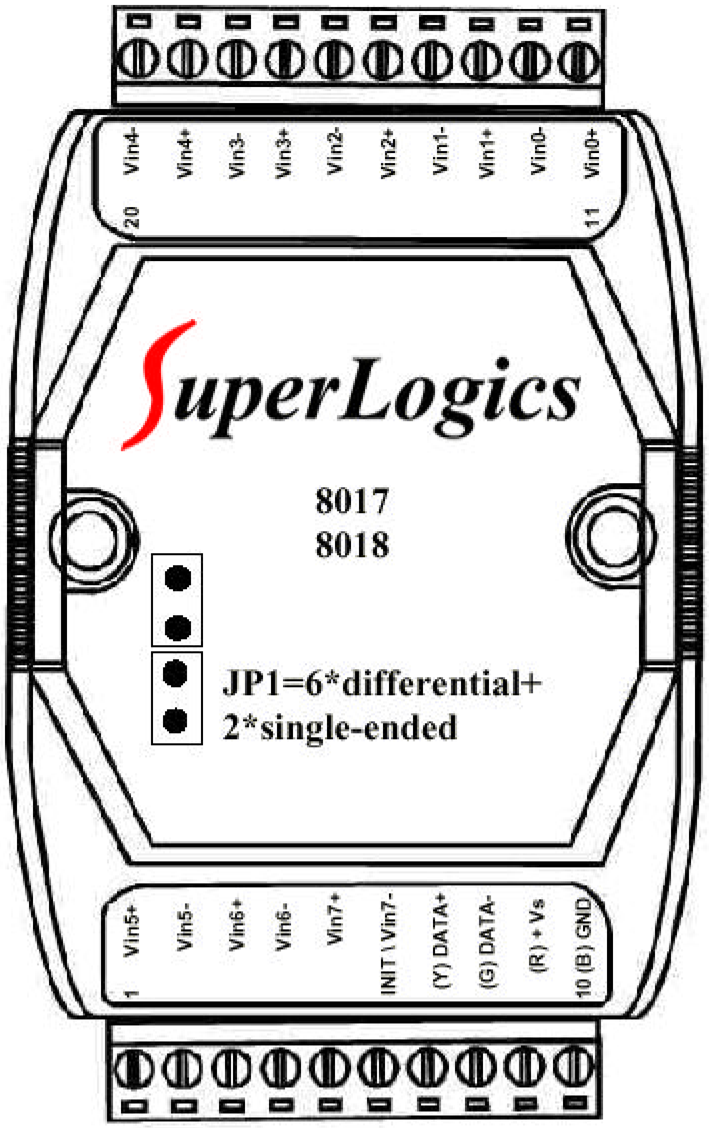
All 8000 series analog input modules use a microprocessor to control a 16-bit Sigma-Delta A/D to acquire analog signals. The 8017 and 8018 each have eight analog input channels, making these modules extremely cost-effective for industrial applications. The 8013D has one analog RTD input channel and is equipped with a 4 ½ digit LED window which can display single channel readings in real time.

More Information

Refer to chapter one of the 8520 manual for information on the following:

1.1 8000 Series Overview
1.2 8000 Common Features
1.3 8000 System Network Configuration
1.4 8000 Dimension

8018 Pin Assignment



8018 Specifications

8018: 8-Channel Thermocouple Input Module

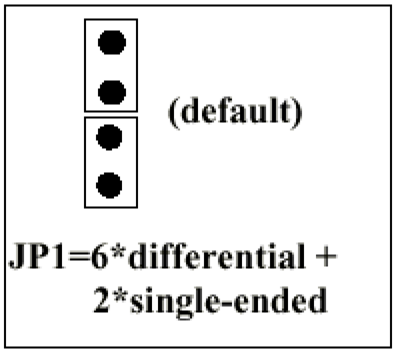
Analog Input

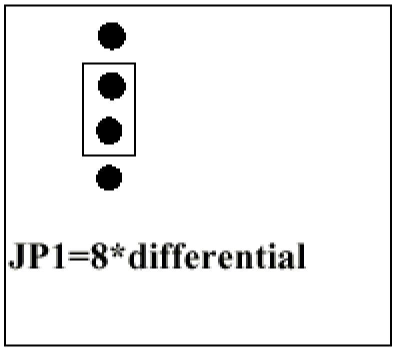
- Type: thermocouple, mV, V , or mA
- Channels: 6 differential + 2 single-ended or
  8 differential(jumper select)
- Thermocouple type:

Type	Range	Type	Range
J	-210^{0}C ~ 760^{0}C	S	0^{0}C ~ 1768^{0}C
K	-270^{0}C ~ 1372^{0}C	B	0^{0}C ~ 1820^{0}C
T	-270^{0}C ~ 400^{0}C	N	-270^{0}C ~ 1300^{0}C
E	-270^{0}C ~ 1000^{0}C	C	0^{0}C ~ 2320^{0}C
R	0^{0}C ~ 1768^{0}C		

- Voltage range: ±15mV, ±50mV, ±100mV, ±500mV, ±1V, ±2.5V
- Current range: ±20mA
- Sampling rate: 10 samples/sec(total)
- Bandwidth: 13.1 Hz
- Accuracy: ±0.05% or better
- Zero drift: ±0.033ppm/°C
- CMR @ 50/60 Hz: 150 dB
- NMR @50/60 Hz: 100 dB
- Span drift: 25ppm/°C
- Over voltage protection: ±35V

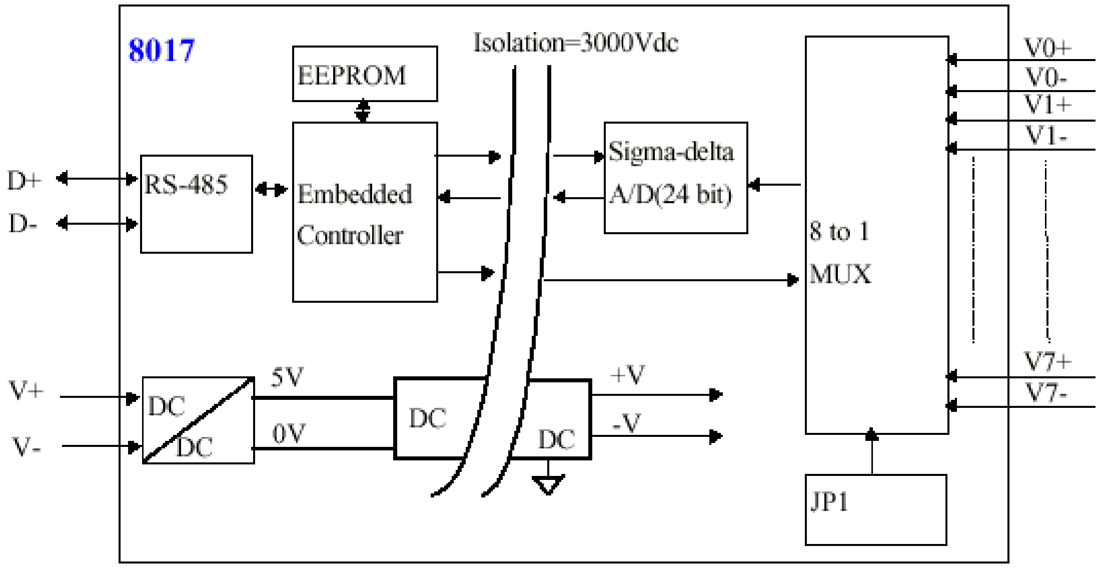
Power

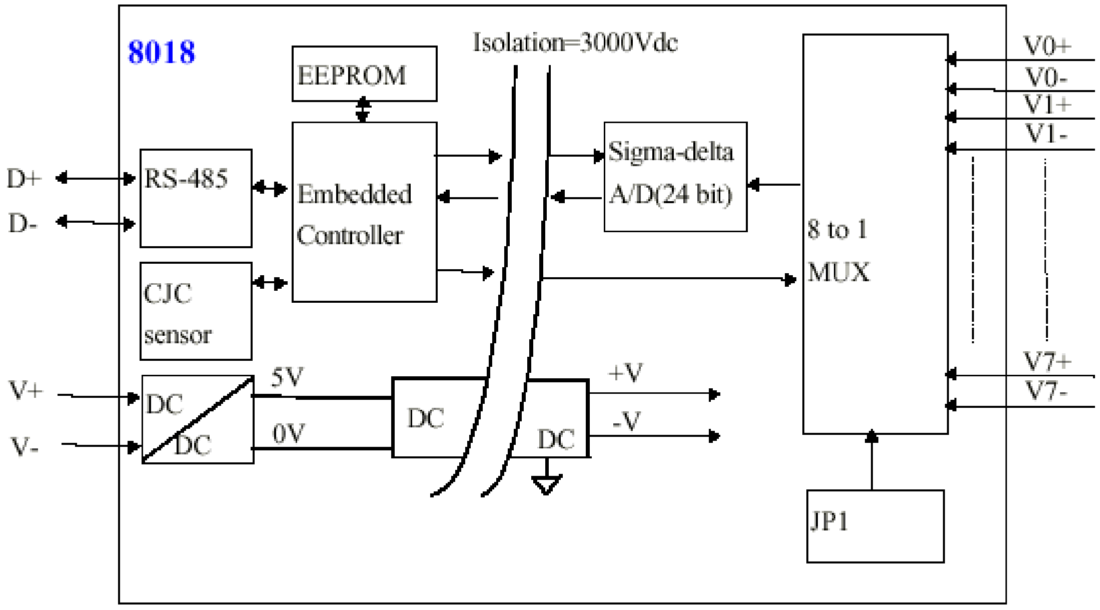




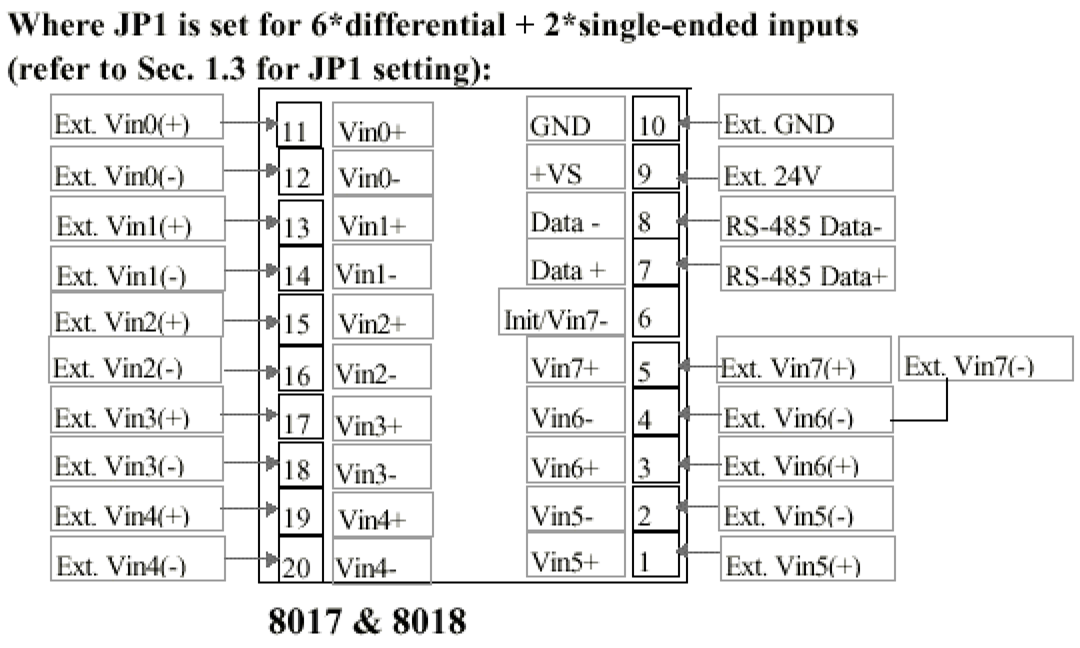
- Power consumption: 2W

8018 Block Diagrams

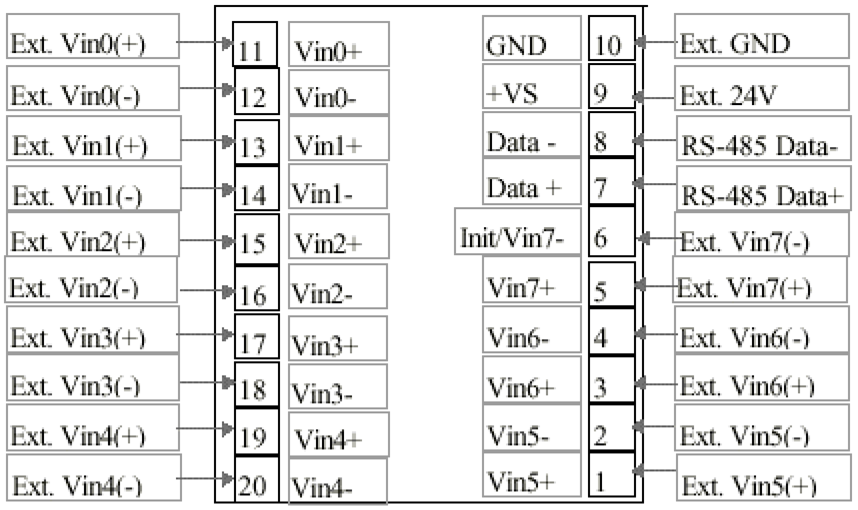




8018 Application Wiring

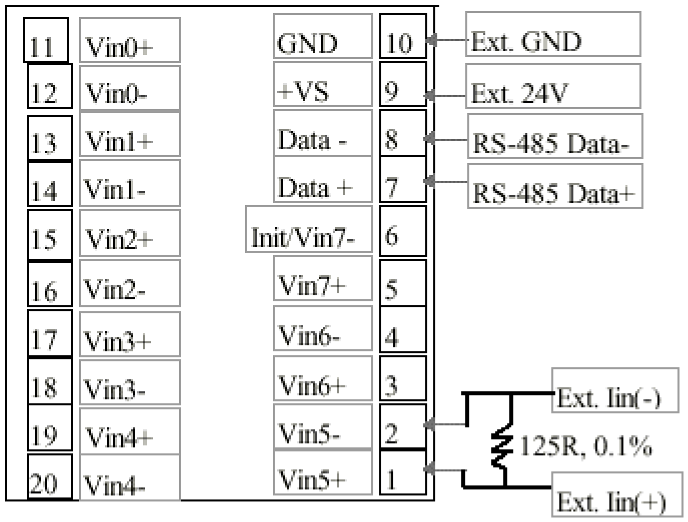


Where JP1 is used to select 8*differential inputs (refer to Sec. 1.3 for JP1 setting):



8017 & 8018

Current Measurement



8017 & 8018

8018 Default Settings

The default settings for 8000 analog modules are:

. address=01, baud rate=9600, checksum disabled
. type=08=±10V input range (for 8017)
. type=05=±2.5V input range (for 8018)
. type=20=platinum, ±100°C(for 8013)
. data=1 start+8 data+1 stop(no parity)

NOTE:
On the 8017 and 8018, JP1 is used to select analog input channels as either 6 differential and 2 single-ended or 8 differential. The default setting is for the 6/2 combination. (See section 1.3 for more information).

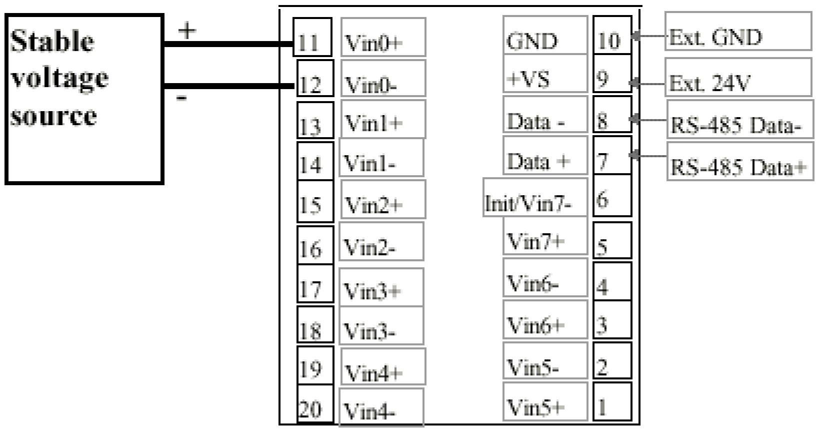
8018 Calibration

Zero/Span Table for 8018 Calibration.

Input Range Code	Input Range	Zero Voltage	Span Voltage
00	± 15mV	0V	15mV
01	± 50mV	0V	50mV
02	± 100mV	0V	100mV
03	± 500mV	0V	500mV
04	± 1V	0V	1V
05	± 2.5V	0V	2.5V
06	± 20mA	0V or 0mA with 125 0.1%	2.5V or 20mA with 125 0.1%
0E	J-type	0mV	42.922mV
0F	K-type	0mV	54.875mV
10	T-type	0mV	20.9mV
11	E-type	0mV	76.358mV
12	R-type	0mV	21.108mV
13	S-type	0mV	18.698mV
14	B-type	0mV	13.814mV
15	N-type	0mV	47.502mV
16	C-type	0mV	37.107mV

NOTE: One type calibrating is enough.

8018 Calibration



Notes:

1. Before calibration, warm-up the module for about 30 minutes for better accuracy.
2. Connect a stable calibration voltage (or current) signal to the module's input channel 0.
3. When calibrating type 06 an external shunt resistor will need to be connected (125 ohms, 0.1%).

Example Calibration Sequence for Type 00:

1. Setting Type to 00	-> Refer to A1.1
2. Enable Calibration	-> Refer to
3. Apply Zero Calibration Voltage (0mV)	
4. Perform Zero Calibration Command	-> Refer to A1.6
5. Apply Span Calibration Voltage (15mV)	
6. Perform Span Calibration Command	-> Refer to A1.5
7. Repeat Step 1 to Step 6 three times.	

Calibration for other types is similar, changing the type in step 1 being the only difference.

8018 Tables

Configuration Code Table: CC (for 8018)

CC	Baud Rate
03	1200 BPS
04	2400 BPS
05	4800 BPS
06	9600 BPS
07	19200 BPS
08	38400 BPS
09	57600 BPS
0A	115200 BPS

Configuration Code: FF, 2-char (for 8018)

7	6	5	4	3	2	1	0
0	checksum 0=disable 1=enable	0				00: engineering unit 01: % of FSR 10: 2's complement of hexadecimal 11: Ohms (for 8013)	

Configuration Code Table: TT (for 8018)

Type Code	00	01	02	03	04	05	06
Min. Input	-15 mV	-50 mV	-100 mV	-500 mV	-1 V	-2.5 V	-20 mA
Max Input	+15 mV	+50 mV	+100 mV	+500 mV	+1 V	+2.5 V	+20 mA

Type Code	0E	0F	10	11	12	13	14	15	16		
T.C. Type	J	K	T	E	R	S	B	N	C		
Min Temp.	-210	-270	-270	-270	0	0	0	-270	0		
Max Temp.	760	1372	400	1000	1768	1768	1820	1300	2320		
The temperature is shown in degree Celsius											

Data Format Table (data): (for 8018)

Type Code	Input Range	Data Format	+F.S.	Zero	-F.S.
00	-15 to +15 mV	Engineer Unit	+15.000	+00.000	-15.000
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
01	-50 to +50 mV	Engineer Unit	+50.000	+00.000	-50.000
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
02	-100 to +100 mV	Engineer Unit	+100.00	+000.00	-100.00
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
03	-500 to +500 mV	Engineer Unit	+500.00	+000.00	-500.00
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
04	-1 to +1 V	Engineer Unit	+1.0000	+0.0000	-1.0000
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
05	-2.5 to +2.5 V	Engineer Unit	+2.5000	+0.0000	-2.5000
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000
06	-20 to +20 mA	Engineer Unit	+20.000	+00.000	-20.000
		% of FSR	+100.00	+000.00	-100.00
		2's complement HEX	7FFF	0000	8000

Type Code	Input Range	Data Format	+F.S.	Zero	-F.S.
0E	J Type -210 to 760 degree Celsius	Engineer Unit	+760.00	+00.000	-210.00
		% of FSR	+100.00	+000.00	-027.63
		2's complement HEX	7FFF	0000	DCA2
0F	K Type -270 to 1372 degree Celsius	Engineer Unit	+1372.0	+00.000	-0270.0
		% of FSR	+100.00	+000.00	-019.68
		2's complement HEX	7FFF	0000	E6D0
10	T Type -270 to 400 degree Celsius	Engineer Unit	+400.00	+000.00	-270.00
		% of FSR	+100.00	+000.00	-067.50
		2's complement HEX	7FFF	0000	A99A
11	E Type -270 to 1000 degree Celsius	Engineer Unit	+1000.0	+000.00	-0270.0
		% of FSR	+100.00	+000.00	-027.00
		2's complement HEX	7FFF	0000	DD71
12	R Type 0 to 1768 degree Celsius	Engineer Unit	+1768.0	+0000.0	+0000.0
		% of FSR	+100.00	+0000.0	+0000.0
		2's complement HEX	7FFF	0000	0000
13	S Type 0 to 1768 degree Celsius	Engineer Unit	+1786.0	+0.0000	+0000.0
		% of FSR	+100.00	+000.00	+0000.0
		2's complement HEX	7FFF	0000	0000
14	B Type 0 to 1820 degree Celsius	Engineer Unit	+1820.0	+00.000	+0000.0
		% of FSR	+100.00	+000.00	+0000.0
		2's complement HEX	7FFF	0000	0000
15	N Type -270 to 1300 degree Celsius	Engineer Unit	+1300.0	+00.000	-0270.0
		% of FSR	+100.00	+000.00	-20.77
		2's complement HEX	7FFF	0000	E56B

Type Code	Input Range	Data Format	+F.S.	Zero	-F.S.
16	C Type 0 to 2320 degree Celsius	Engineer Unit	+2320.0	+00.000	+00.000
		% of FSR	+100.00	+000.00	+000.00
		2's complement HEX	7FFF	0000	0000

Command Set Table

Command	Response	Description	Reference
%AANNTTCCFF	!AA	Set module configuration	A1.1
#**	No Response	Synchronized Sampling	A1.2
#AA	>(data)	Read analog input	A1.3
#AAN	>(data)	Read analog input from channel_N	A1.4
$AA0	!AA	Perform span calibration	A1.5
$AA1	!AA	Perform zero calibration	A1.6
$AA2	!AATTCCFF	Read configuration	A1.7
$AA3	!AA(data)	Read CJC value	A1.8
$AA4	!AA(data)	Read Synchronized Data	A1.9
$AA5VV	!AA	Enable/disable channel multiplexing	A1.10
$AA6	!AAVV	Read channel multiplexing status	A1.11
$AA8V	!AA	Select Led Configuration	A1.12
$AA9SCCCC	!AA	Set CJC Offset Value	A1.13
$AA9S(data)	!AA	Send Led Display	A1.14
$AAA	>(data)*8	Read all 8 channel data	A1.15
$AAF	!AA(data)	Read the firmware version number	A1.16
$AAM	!AA(data)	Read the module name	A1.17
~**	No Response	Host OK Sec. 2.18	A1.18
~AA0	!AASS	Read Module Status	A1.19
~AA1	!AA	Reset Module Status	A1.20
~AA2	!AATT	Read Host Watchdog Timer Value	A1.21
~AA3ETT	!AA	Enable Host Watchdog Timer	A1.22
~AAO(name)	!AA	Set module name	A1.25

8018 Command Set Table

Command	Response	Description	Reference
%AANNTTCCFF	!AA	Set module configuration	A1.1
#AAN	>(data)	Read analog input from channel_N	A1.4
$AA0	!AA	Perform span calibration	A1.5
$AA1	!AA	Perform zero calibration	A1.6
$AA2	!AATTCCFF	Read configuration	A1.7
$AA3	!AA(data)	Read CJC value	A1.8
$AA5VV	!AA	Enable/disable channel multiplexing	A1.9
$AA6	!AAVV	Read channel multiplexing status	A1.10
$AA9SCCCC	!AA	Set CJC Offset Value	A1.13
$AAF	!AA(data)	Read the firmware version number	A1.16
$AAM	!AA(data)	Read the module name	A1.17
~**	No Response	Host OK	A1.18
~AA0	!AASS	Read Module Status	A1.19
~AA1	!AA	Reset Module Status	A1.20
~AA2	!AATT	Read Host Watchdog Timer Value	A1.21
~AA3ETT	!AA	Enable Host Watchdog Timer	A1.22
~AAO(name)	!AA	Set module name	A1.25
~AAEV	!AA	Enable/Disable Calibration	A1.26

APPENDIX A: COMMAND SETS

A1.1 %AANNTTCCFF

- **Description**: Set module configuration.

- **Syntax**: %AANNTTCCFF[chk](cr)
  % is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  NN=new AA
  TT=Input range code, refer to section 1.8
  CC=baud rate code, refer to section 1.8
  FF=status code, refer to section 1.8
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

  command: %0102090600(cr)
  response : !02(cr)

  Address 01 is configured to a new address 02, ±5V input

  command: %0202080600(cr)
  response : !02(cr)

  Change to ±10V input

A1.2 #**

- **Descriptio**n: Order all digital and analog input modules to sample their input data immediately and store that data in their internal registers for later retrieval by the host via the **$AA4, read synchronized data** command.

- **Synta**x: #**[chk](cr)
  # is a delimiter character
  * is a command character
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Respons**e: no response

- **Exampl**e:
  command: #**(cr)

  response: no response

  Order all modules to perform synchronized sampling.

  command: $014(cr)

  response: !1©©©©©©©(cr)

  command: $024(cr)

  response: !1©©©©©©©(cr)

  command: $034(cr)

  response: !1©©©©©©©(cr)

  Read the data stored by each module, in turn. In this example, read module-01, 02, 03. © is a character dependent upon module wiring and previously entered commands.

NOTE: “synchronized sampling” Explained

The host computer can send only one command string at a time. If there are two modules, the host computer must send a command and receive a reply from module-1 and then send a command and receive a reply from module-2. Obviously, there is a time delay between these two commands. The “synchronize sampling” command is designed to address all modules in the network at the same time to achieve simultaneous sampling. When receiving the #**[0x0D] synchronized sampling command, all the input modules in the RS-485 network perform the input function at the same time and store the data values in memory. Then the host computer sends out the $AA4, “read synchronize data” command to each of
the modules in turn to retrieve the stored data.

A1.3 #AA

- **Description**: Read the analog input value.

- **Syntax** : #AA[chk](cr)
  # is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response** : valid command → >(data)[chk](cr)
  invalid command → No Response
  no response → syntax error or communication error or address error
  > is a delimiter character indicating a valid command
  (data) = refer to section 1.8
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:
  command: #01(cr)
  response : >+100.00(cr)

  Temperature=100.0 °C

  command: #02(cr)
  response : >-100.00(cr)

  Temperature=-100.0 °C

A1.4 #AAN

- **Description**: Read the analog value from channel N.

- **Syntax**: #AAN[chk](cr)
  # is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  N=channel number, from 0 to 7
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → >(data)[chk](cr)
  invalid command → No Response
  no response → syntax error or communication error or address error
  > is a delimiter character indicating a valid command
  (data) = refer to section 1.8
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command:	#010(cr)	channel_0=1.2345V
response :	>+1.2345(cr)	
command:	#012(cr)	channel_2=444.44mV
response :	>+444.44(cr)	

A1.5 $AA0

- **Description**: Perform SPAN calibration. Refer to the module calibration sections for more information.

- **Syntax**: $AA0[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error

  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example:**

command: $010(cr) response : !01(cr)	Address 01 performs SPAN calibration
command: $020(cr) response : !02(cr)	address 02 performs SPAN calibration

A1.6 $AA1

- **Description**: Perform ZERO calibration. . Refer to the module calibration sections for more information.

- **Syntax**: $AA1[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: $011(cr) response : !01(cr)	address 01 perform ZERO calibration
command: $021(cr) response : !02(cr)	address 02 perform ZERO calibration

A1.7 $AA2

- **Description**: Read module configuration.

- **Syntax**: $AA2[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D
- **Response**: valid command → !AATTCCFF[chk](cr),
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  TT, CC, FF: refer to section 1.8
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:
  command: $012(cr)
  response : !01080600(cr)

  Address 01, ±10V, 9600 BPS, checksum disabled, engineering unit

  command: $022(cr)
  response : !02050700(cr)

  Address 02, ±2.5V, 19200 BPS, checksum disabled, engineering unit

NOTE: If the %AANNTTCCFF command is used to change module configuration, the new configuration code will be stored into EEPROM immediately. The configuration code includes module address, module type, baud rate code, checksum enable/disable code, calibration code, power-on value and safe value. The 8000 Series EEPROM data can be read infinitely many times, but can be written to about 100,000 times max. Therefore careful consideration should be given to changing the configuration code. The $AA2 command is used to solely read EEPROM data, therefore this command can be sent to 8000 Series modules as often as necessary.

A1.8 $AA3

- **Description**: Read current CJC value. Refer to Sec. 3.5 for more information.

- **Syntax**: $AA3[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !S(data)[chk](cr),
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  S=+ or –
  (data)=CJC value
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: $013(cr) response : !+0030.0(cr)	CJC=30 °C
command: $023(cr) response : !+0032.1(cr)	CJC=32.1 °C

A1.9 $AA4

- **Description**: Read synchronized data.

- **Syntax**: $AA4[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !S(data)[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  S=1=first reading, S=0=not first reading
  (data) = refer to section 1.8
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example** :
  command: $01M(cr)
  response : !018013D(cr)
  command: $02M(cr)
  response : !028013D(cr)
  command: #**
  response : No Response

  Address-01 is 8013D.
  Address-02 is 8013D
  Perform synchronized sampling

  command: $014(cr)
  response : !1+123.45(cr)
  command: $014(cr)
  response : !00+123.45(cr)

  Synchronized data = +123.45, first time
  Synchronized data = +123.45, not first time

  command: $024(cr)
  response : !1-123.45(cr)
  command: $024(cr)
  response : !0-123.45(cr)

  Synchronized data = -123.45, first time
  Synchronized data = -123.45, not first time

A1.10 $AA5VV

- **Descriptio**n: Enable or disable channel multiplexing.

- **Synta**x: $AA5VV[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  VV=2-character HEX value, from 00 to FF, 8 bits refer to 8 channels,
  1=enable, 0=disable
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Respons**e: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Exampl**e:

command: $015F0(cr) response : !01(cr)	enable channel_7 to channel_4 disable channel_0 to channel_3
command: $025AA(cr) response : !02(cr)	enable channel_7/5/3/1 disable channel_6/4/2/0

A1.11 $AA6

- **Description**: Read channel multiplexing status.

- **Syntax**: $AA6[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AAVV[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  VV=2-character HEX value, from 00 to FF, 8 bits refer to 8 channels,
  1=enable, 0=disable
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command:	$016(cr)	channel_7 to channel_4 are enabled channel_0 to channel_3 are disabled
response :	!01F0(cr)	

command:	$026(cr)	channel_7/5/3/1 are enabled channel_6/4/2/0 are disabled
response :	!02AA(cr)	

A1.12 $AA8V

- **Descriptio**n: Select LED Configuration.

- **Synta**x: $AA8V[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  V=1 → module control LED, V=2 → host control LED
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Respons**e: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Exampl**e:

command: $0181(cr) response : !01(cr)	8013D to control LED
command: $0282(cr) response : !02(cr)	Host to control LED

A1.13 $AA9SCCCC

- **Descriptio**n: Set CJC offset value. Refer to Sec. 3.5 for more information.

- **Synta**x: $AA9SCCCC[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  S= + or -CCCC=
  4-char HEX value, 1 count=0.01 °C
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Respons**e: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Exampl**e:
  command: $019+000A(cr)
  response : !01(cr)

  CJC offset=10*0.01=0.1 °C

  command: $029-0014(cr)
  response : !02(cr)

  CJC offset=-20*0.01=-0.2 °C

A1.14 $AA9S(data)

- **Descriptio**n: Send LED display.

- **Synta**x: $AA9S(data)[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  S=+ or -(
  data)=5 decimal digit + 1 decimal point, max=19999.
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Respons**e: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Exampl**e:

command: $019+19999.(cr) response : !01(cr)	Show max = +19999.
command: $029-19999.(cr) response : !02(cr)	Show min = -19999.
command: $039+12.345(cr) response : !03(cr)	Show display = +12.345

A1.15 $AAA

- **Description**: Read data from all 8 analog input channels. Refer to section 3.6 for more information.

- **Syntax**: $AAA[chk](cr)

$ is a delimiter character
AA=2-character HEX module address, from 00 to FF
[chk]=2-character checksum, if checksum disabled → no [chk]
(cr)=0x0D

- **Response**: valid command → !(data)*8[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  data=4-character HEX value, from 0000 to FFFF, 2's complement data format
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:
  command: $01A(cr)
  response : !00001111222233334444555566667777(cr)

channel_0=0000
channel_1=1111
channel_2=2222
channel_3=3333
channel_4=4444
channel_5=5555
channel_6=6666
channel_7=7777

- All data are in HEX format
- 8000 → min
- 7FFF → max
- 0000 → 0
- assume type=08,
1. 8000 → -10V
2. 7FFF → +10V
3. 0000 → 0V
4. 1000 → +1.25V
5. F000 → -1.25V

A1.16 $AAF

- **Description**: Read the firmware version number.

- **Syntax**: $AAF[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA(data)[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  data=5-character for version number
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:
  command: $01F(cr)
  response : !01A2.0(cr)

  module 01 uses version A2.0

  command: $02F(cr)
  response : !02A3.0(cr)

  module 02 uses version A3.0

A1.17 $AAM

- **Description**: Read the module name.

- **Syntax**: $AAM[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA(data)[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  data=4-character for module name
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:
  command: $01M(cr)
  response : !017017(cr) or !018017(cr)

  Module 01 is 8017

  command: $02M(cr)
  response : !027018(cr) or !028018(cr)

  Module 02 is 8018

  command: $03M(cr)
  response : !037013D(cr) or !038013D(cr)

  Module 03 is 8013D

A1.18 ~**

- **Description**: The host uses this command to tell all modules in the network that it is functioning properly. See section 3.5 for more information.

- **Syntax**: ~**[chk](cr)
  ~ is a delimiter character
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: no response

- **Example**:
  command: ~**(cr)
  response : No Response

A1.19 ~AA0

- **Description**: Read the module status. The module status will be latched until the ~AA1 command is sent. If the host watchdog is enabled and the host is down, the module status will be set to 4. If the module status=4, all output commands will be ignored.
- **Syntax**: ~AA0[chk](cr)
  ~ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D
- **Response**: valid command → !AASS[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  SS=2-character HEX status value
  Bit_0, Bit_1 = reserved
  Bit_2 = 0 → OK,
  1 → host watchdog failure
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D
- **Example**:

command: ~010(cr) response : !0100(cr)	Status of module 01 is OK
command: ~020(cr) response : !0204(cr)	Status of module 02 is "host watchdog failure" → HOST is down

A1.20 ~AA1

- **Description**: Reset module status. The module status will be latched until the ~AA1 command is sent. **If the module statue=0x04, all output commands will be ignored.** Therefore the user should read the module status before sending other commands to make sure that the module status is 0. If the module status is not 0, only the ~AA1 command can clear the module status.

- **Syntax**: ~AA1[chk](cr)
  ~ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: ~010(cr) response : !0104(cr)	module status=0x04 → host is down
command: #0105.000(cr) response : !(cr)	Output commands are ignored
command: ~011(cr) response : !01(cr)	clear module status
command: ~010(cr) response : !0100(cr)	module status=0x00
command: #0105.000(cr) response : >(cr)	Output commands can now be processed

A1.21 ~AA2

- **Description**: Read the host watchdog status and the host watchdog timer value. The host watchdog timer is designed for the software host watchdog. When the software host watchdog is enabled, the host must send the ~**, "HOST is OK" command, to all modules before the timer is up. When the ~** command is received, the host watchdog timer is reset and restarts. Use the ~AA3ETT command to enable/disable/set the host watchdog timer.

- **Syntax**: ~AA2[chk](cr)
  ~ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AASTT[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  S=0: host watchdog is disable
  S=1: host watchdog is enable
  TT=2-character HEX value, from 00 to FF, unit=0.1 second
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:
  command: ~012(cr)
  response : !01000(cr)

  Module 01 host watchdog timer is disabled

  command: ~022(cr)
  response : !0210A(cr)

  Module 02 host watchdog timer is enabled and =0.1*10 =1 second.

A1.22 ~AA3ETT

- **Description**: Enable/disable the host watchdog timer. The host watchdog timer is designed for the software host watchdog. When the software host watchdog is enabled, the host must send ~** command to all modules before the timer is up. When the ~** command is received, the host watchdog timer is reset and restarted. Use the ~AA2 to read the host watchdog status & value.

- **Syntax**: ~AA3ETT[chk](cr)
  ~ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  E=0 is disable and 1 is enable
  TT=2-character HEX value, from 00 to FF, unit=0.1 second
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: ~013000(cr) response : !01(cr)	**disable module 01 host watchdog timer**
command: ~02310A(cr) response : !02(cr)	**host watchdog timer of module 02 is enabled and equal to 0.1*10 =1 second.**

A1.23 ~AA4

- **Description** : Read power-on value and safe value.

- **Syntax** : ~AA4[chk](cr) → read safe value
  ~ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Respons**e: valid command → !AAPPSS[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  PP= power-on value, SS=safe value
  00 → DO1=DO2=OFF
  01 → DO1=ON, DO2=OFF
  02 → DO1=OFF, DO2=ON
  03 → DO1=DO2=ON
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Exampl**e:

command: ~014(cr)
response : !010003(cr)

Power-on value→ Do1=off, Do2=off
Safe value→ Do1=on, Do2=on

command: ~024(cr)
response : !020201(cr)

Power-on value→ Do1=off, Do2=on
Safe value→ Do1=on, Do2=off

A1.24 ~AA5

- **Description**: Set power-on value and safe value.

- **Syntax** : ~AA5PPSS[chk](cr) → set safe value
  ~ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  PP= power-on value, SS=safe value
  00 → DO1=DO2=OFF
  01 → DO1=ON, DO2=OFF
  02 → DO1=OFF, DO2=ON
  03 → DO1=DO2=ON
  (cr)=0x0D

- **Response** : valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: ~0150003(cr)
response : !01(cr)

Power-on value→ Do1=off, Do2=off
Safe value→ Do1=on, Do2=on

command: ~0250201(cr)
response : !02(cr)

Power-on value→ Do1=off, Do2=on
Safe value→ Do1=on, Do2=off

A1.25 ~AAO(name)

- **Description**: Set module name.

- **Syntax**: ~AAO(name)[chk](cr)
  ~ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  (name)=4-character/5-character module name
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:
  command: $01M(cr)
  response : !018017(cr)
  command: ~01O1234(cr)
  response : !01(cr)

  Change module name from 8017 to 1234

  command: $01M(cr)
  response : !018013D(cr)
  command: ~01O5678D(cr)
  response : !01(cr)

  Change module name from 8013D to 5678D

Note: This command is designed for OEM/ODM users. However if for some reason a general user needs to rename the modules it can be useful.

A1.26 ~AAEV

- **Description**: Enable/Disable Calibration.

- **Syntax**: ~AAEV[chk](cr)
  ~ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  E = command for enable/disable calibration
  V 1= enable calibration, 0= disable calibration
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: $010	Perform address 01 span calibration.
response : ?01	It is not ready for calibration.
command: ~01E1	Set address 01 to enable calibration.
response : !01	Return success.
command: $010	Perform address 01 calibration.
response : !01	Return success.

A1.27 $AA3

- **Descriptio**n: Read the source linear mapping value [low, high]. Refer to Sec. 3.7 for more information.

- **Synta**x: $AA3[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Respons**e: valid command → !AA(LO)(HI)[chk](cr),
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  (LO)=low value of source linear mapping
  (HI) = high value of source linear mapping
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Exampl**e:
  command: $013(cr)
  response : !01+04.000+20.000(cr)

Source linear mapping readback = [4.0, 20.0]

  command: $023(cr)
  response : !02+000.00+100.00(cr)

Source linear mapping readback = [0.0, 100.0]

Note: the data format of (HI) & (LO) is the same as the current configuration. Refer to “Data Format Table (data)” in Section 1.8 for details.

A1.28 $AA5

- **Description**: Read the target linear mapping value [low, high]. Refer to Sec. 3.7 for more information.

- **Syntax**: $AA5[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA(LO)(HI)[chk](cr),
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  (LO)=low value of target linear mapping
  (HI) = high value of target linear mapping
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

  command: $015(cr)
  response : !01+04.000+20.000(cr)

  Target linear mapping readback = [4.0, 20.0]

  command: $025(cr)
  response : !02+000.00+100.00(cr)

  Target linear mapping read back = [0.0, 100.0]

Note: the data format of (HI) & (LO) is given as follows:

- **first char is + or -**
- **the next 6 characters must include one decimal point**
- **min. value → 19999.**
- **max. value → +19999.**

A1.28 $AA6(LO)(HI)

- **Description**: Write the source linear mapping value [low, high]. Refer to Sec. 3.7 for more information.

- **Syntax**: $AA6(LO)(HI)[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  (LO)=low value of source linear mapping
  (HI) = high value of source linear mapping
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr),
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: $016+04.000+20.000(cr) response : !01(cr)	Set source linear mapping = [4.0, 20.0]
command: $026+000.00+100.00(cr) response : !02(cr)	Set source linear mapping = [0.0, 100.0]

Note: the data format of (HI) & (LO) is the same as current configuration. Refer to "Data Format Table (data)" in Section 1.8 for details.

A1.29 $AA7(LO)(HI)

- **Description**: Write the target linear mapping value [low, high]. Refer to Sec. 3.7 for more information.

- **Syntax**: $AA7(LO)(HI)[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  (LO)=low value of target linear mapping
  (HI) = high value of target linear mapping
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr),
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: $017+04.000+20.000(cr)
response : !01(cr)

Set target linear mapping = [4.0, 20.0]

command: $027+000.00+100.00(cr)
response : !02(cr)

Set target linear mapping = [0.0, 100.0]

Note: the data format of (HI) & (LO) is given as following:

- **first char is + or -**
- **the next 6 character must include one decimal point**
- **min. value → -19999.**
- **max. value → +19999.**

A1.30 $AAAV

- **Description**: Enable/disable linear mapping. Refer to Sec. 3.7 for more information.

- **Syntax**: $AAAV[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  V= 0: disable linear mapping
  1: enable linear mapping
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: $01A0(cr)
response : !01(cr)

Disable linear mapping.

command: $02A1(cr)
response : !02(cr)

Enable linear mapping.

A1.31 $AAB

- **Description**: Linear mapping status readback. Refer to Sec. 3.7 for more information.

- **Syntax**: $AAB[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AAS[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  S= 0 → linear mapping is disabled
  1 → linear mapping is enabled
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: $01B(cr)
response : !010(cr)

Linear mapping is disabled.

command: $02B(cr)
response : !021(cr)

Linear mapping is enabled.

A1.32 @AADI

- **Description**: Read the digital I/O and alarm status.

- **Syntax**: @AADI[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AAS0D0I[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  S =0 → disable, 1=momentary alarm, 2=latch alarm
  D =0 → DO1=DO2=OFF
  =1 → DO1=ON, DO2=OFF
  =2 → DO1=OFF, DO2=ON
  =3 → DO1=DO2=ON
  I =0 → D/I is low, I=1 → D/I is high
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01DI(cr)
response : !0100001(cr)

Alarm disabled. DO1=DO2=OFF. D/I is high.

command: @02DI(cr)
response : !0210100(cr)

Alarm enabled. DO1=ON. DO2=OFF. D/I is low.

A1.33 @AADO0D

- **Description**: Set digital output.

- **Syntax**: @AADO0D[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  D =0 → DO1=DO2=OFF
  =1 → DO1=ON, DO2=OFF
  =2 → DO1=OFF, DO2=ON
  =3 → DO1=DO2=ON
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  alarm is enabled → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01DO00(cr)
response : !01(cr)

Turn all D/O OFF.

command: @02DO01(cr)
response : !02(cr)

Turn DO1 ON, DO2 OFF.

NOTE: If the Hi/Lo alarm is enabled, the module controls the digital output channels. Therefore in the case of a system failure, the power-on value is changed to Hi/Lo condition immediately, and the safe value as well as the @AADO0D commands are ignored.

A1.34 @AAEAT

- **Description**: Enable alarm.

- **Syntax**: @AAEAT[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  T=M → momentary alarm, T=L → latch alarm
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01EAL(cr) response : !01(cr)	Latch alarm.
command: @02EAM(cr) response : !02(cr)	Momentary alarm.

NOTE: If the Hi/Lo alarm is enabled, the module will control the digital output channels. Therefore in the case of a system failure, the power-on value is changed to Hi/Lo condition immediately, and the safe value as well as the @AADO0D commands are ignored.

A1.35 @AAHI(data)

- **Description**: Set high alarm value.

- **Syntax**: @AAHI(data)[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  data: engineering unit format. Refer to Section 1.8.
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01HI+050.00(cr) response : !01(cr)	High alarm=50 °C
command: @02HI+100.00(cr) response : !02(cr)	High alarm=100 °C

A1.36 @AALO(data)

- **Description**: Set low alarm value.

- **Syntax**: @AALO(data)[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  data: engineering unit format. Refer to Section 1.8.
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01LO+000.00(cr)
response : !01(cr)

Low alarm=0 °C

command: @02LO-010.00(cr)
response : !02(cr)

Low alarm=-10 °C

A1.37 @AADA

- **Description**: Disable alarm.

- **Syntax**: @AADA[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01DA(cr) response : !01(cr)	Alarm disabled.
command: @02DA(cr) response : !02(cr)	Alarm disabled.

A1.38 @AACA

- **Description**: Clear latch alarm.

- **Syntax**: @AACA[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01CA(cr) response : !01(cr)	Clear latch alarm.
command: @02CA(cr) response : !02(cr)	Clear latch alarm.

A1.39 @AARH

- **Description**: Read high alarm value.

- **Syntax**: @AARH[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA(data)[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  data = engineering unit format. Refer to Section 1.8.
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

  command: @01RH(cr)
  response : !01+100.00(cr)

  High alarm=100 °C

  command: @02RH(cr)
  response : !02+050.00(cr)

  High alarm=50 °C

A1.40 @AARL

- **Description**: Read low alarm value.

- **Syntax**: @AARL[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA(data)[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  data= engineering unit format. Refer to Section 1.8.
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01RL(cr)
response : !01+000.00(cr)

Low alarm=0 °C

command: @02RL(cr)
response : !02-010.00(cr)

Low alarm=-10 °C

A1.41 @AARE

- **Description**: Read the event counter value.

- **Syntax**: @AARE[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA(data)[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  data=5-character HEX value, from 00000 to 65535
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01RE(cr)
response : !0100001(cr)

Event counter=1.

command: @02RE(cr)
response : !0212345(cr)

Event counter=12345.

A1.42 @AACE

- **Description**: Clear the event counter

- **Syntax**: @AACE[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  no response → syntax error or communication error or address error
  ! is a delimiter character indicating a valid command
  ? is a delimiter character indicating an invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disabled → no [chk]
  (cr)=0x0D

- **Example**:

command: @01CE(cr) response : !01(cr)	Clear the event counter to 0
command: @02CE(cr) response : !02(cr)	Clear the event counter to 0

Operation Principles & Application Notes

INIT*_pin Operation Principle

All 8000 Series modules contain an EEPROM to store configuration information. It is difficult to find out the status of the 8000 Series modules. When jumpered to INIT, and the INIT*_pin is connected to the GND_pin and the module is powered on, all 8000 Series modules will revert to factory default setting without changing the EEPROM data . The factory default setting for analog input modules is as follows:

Address	= 00
baud rate	= 9600
checksum	= DISABLE
data format	= 1 start + 8 data bits + 1 stop bit

If the user disconnects the INIT*_pin from the GND_pin, the 8000 module will be auto configured according to the EEPROM data.

Follow these steps to find EEPROM configuration data in the default setting:

Step 1 : power off and connect the INIT*_pin to the GND_pin
Step 2 : power on
Step 3 : send command string **$002[0x0D]**
Step 4 : record the status of this 8000 module
Step 5 : power off and disconnect INIT*_pin and GND_pin
Step 6 : power on

Dual WatchDog Operation Principle

Dual watchdog = host watchdog + module watchdog
The host watchdog is a software watchdog.
The module watchdog is a hardware watchdog.

The 8000 series is designed for harsh environments and industrial applications. In such environments, there is bound to be a problem with noise and transient energy. If there is a very large amount of interference, it may cause problems in the 8000 modules. The watchdog timers are constantly monitoring the system to make sure that everything is functioning within acceptable parameters. The module (hardware) watchdog concerns itself with the individual modules, and if a problem is detected it can reset a single module without altering the entire network. The host (software) watchdog is responsible for the whole system, and will reset the entire network if problems are encountered.

When a problem is encountered in a single module it will revert to its predefined start value. If there is a network problem, all modules will revert to safe states. If the host-PC is down, all modules revert to their predefined safe states for safety protection. **This dual watchdog system greatly increases system reliability, and greatly reduces the potential damage which could result from a system failure.**

Since the The 8017, 8018, and 8013D modules are input only modules, they can not cause any damage to the system if they malfunction, or in the case of a host failure. Consequently, they will not be reset if a failure is detected, and application programs need not take steps to detect watchdog status before sending commands.

Analog Data Format

8000 Series analog input modules can be configured to one of the following data formats:

- Engineering units
- Percent of FSR
- Two's complement hexadecimal

Assuming a ±5V range, data format are as follows:

Engineering Units	Percent of FSR	Two's complement
-5V	-100.00	8000
0V	+000.00	0000
+5V	+100.00	7FFF

The above table is valid for 8017 and 8013D. It is also valid for the 8018 module when it is configured for -00, 01,02, 03, 04,05, 06 and 07.

The following table provides thermocouple data format for the 8018:

Volt Engineering Unit	Percent of FSR	Two's complement
-max	Table(-max)+CJC	-100.00 8000
0V	Table(0)+CJC	+000.00 0000
+max	Table(+max)+CJC	+100.00 7FFF

It is recommended that Engineering Units be used when the 8018 functions in thermocouple mode.

Temperature Measurement

The 8018 can be configured for thermocouple inputs.

Use the following steps when processing thermocouple inputs:

1. A/D conversion → measure thermocouple voltage
2. Table lookup → T1=Table(thermocouple voltage)
3. Get Temperature → Temperature=T1+CJC-value

Use the following steps when measuring CJC:

1. A/D conversion → measure CJC voltage
2. Table lookup → T1=Table(CJC voltage)
3. CJC compensation → CJC-value=T1+CJC-offset

- The CJC-offset is defined by the $AA9 command.
- The CJC-value can be read back by the $AA3 command.

Therefore the temperature error is composed of four errors as follows:

1. Thermocouple error → small
2. A/D converter error → small
3. Table lookup error → small
4. CJC-error → may be big
5. Temperature error = (1)+(2)+(3)+(4)

Refer to Sec. 3.5 for more information on CJC offset calibration.

CJC Offset Calibration

Use the following steps to perform CJC offset calibration:

1. Place a silver temperature sensor just beside the 8018 CJC sensor. Power on and warm-up the module for about 30 minutes. This step is used to find the circumstance temperature. The silver sensor is used to calibrate the CJC sensor.
2. Use the $AA9+0000 command to set CJC offset=0
3. Use the $AA3 command to read out the CJC value, T1
4. Read out the silver sensor temperature, T2
5. CJC offset=T2-T1
6. Use the $AA9±CCCC command to set CJC offset
7. Use the $AA3 command to read out CJC value, T1
8. Repeat step 2 through 7 until T1=T2

Alternative method for calibration (calibrated temperature source required):

1. Power on and warm-up the module for about 30 minutes.
2. Input a designated temperature from a calibrated temperature source into channel 0 of the 8018 module.
3. Use the $AA9+0000 command to set CJC offset=0
4. Use the #AAN command to determine the value of channel 0, T1.
5. CJC offset=Calibrated temp. source – T1
6. Use the $AA9±CCCC command to set CJC offset (CCCC= 4-char HEX value, 1 count=0.01 °C, ex. 2.7°C = 010E in 4 char HEX format)
7. Use the $AAN command to read out CJC value, T1
8. Repeat step 4 through 7 until T1= calibrated temp. source

Command Response Time

The command response sequence for the 8000 Series analog input modules is outlined below:

1. Host sends command
2. 80XX module receives command
3. 80XX waits a character time
4. 80XX takes a data reading and transmits it back to the host.

Assuming a 115.2K baud rate, a typical 8000 transaction would work as follows:

- baud rate=115.2K
- command = **#01(cr)** → 4 character
- wait 1 character
- response = >**HHHH(cr)** → 6 character
- total characters = 4+1+6=11 character
- 1 character = 10 bits → 115.2K/10=11.52K
- 11 characters → 11.52K/11=1.0K max.→ 1000 command/ response per second max.

The length of the command/response time differs with the command sent. The above example provides timing for an ideal system and provides the best possible performance. Clearly, in real world applications, extra computation and control time will be necessary. Typical performance for 8000 Series modules used in a system running a Pentium-120 is about 820 command/response per second. Up to 256 modules can be installed in a single RS-485 network, and the time needed to process 256 command/response sets is about 256/820=0.3 sec. However, 8000 Series analog input modules have a very heavy computation load, so they can not reach the same speeds as digital modules. The real processing speeds achievable when using the 8017 with a Pentium-120 system is about 63 command/response sets per second. The 8017 has 8 analog input channels, thus the module can process about 500 channels per second (63*8=500).