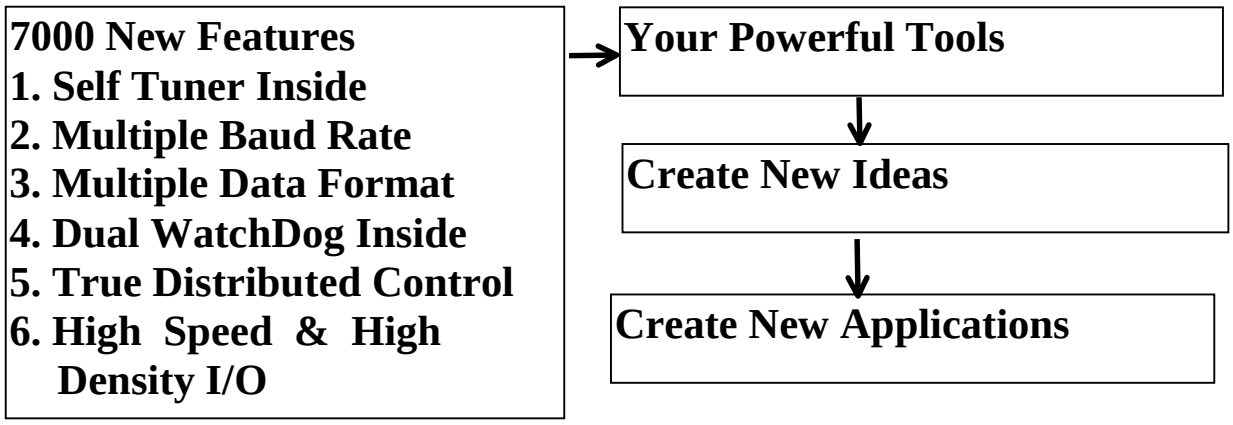
7011/D, 7012/D, 7014D

User Manual



Warranty

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

Warning

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

Copyright

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

Table of Contents

1. Introduction

7000 is a family of network data acquisition and control modules. They provide A/D, D/A, DI/O, Timer/Counter and other functions. These modules can be remote controlled by a set of commands. The common features of 7011 & 7012 are given as following:

- 24 bits sigama-delta A/D converter to provide 16 bits precision.
- with 2 channels of D/O(O.C.)
- with 1 channels of event counter
- Input range is programmable.
- Software Calibration

The 7011 is a single channel thermocouple input module. The 7011D is the 7011 with a 4½ digit LED display. The 7012 is a single channel analog input module. The 7012D is the 7012 with a 4½ digit LED display. The 7014D is very similar to 7012D except the following features:

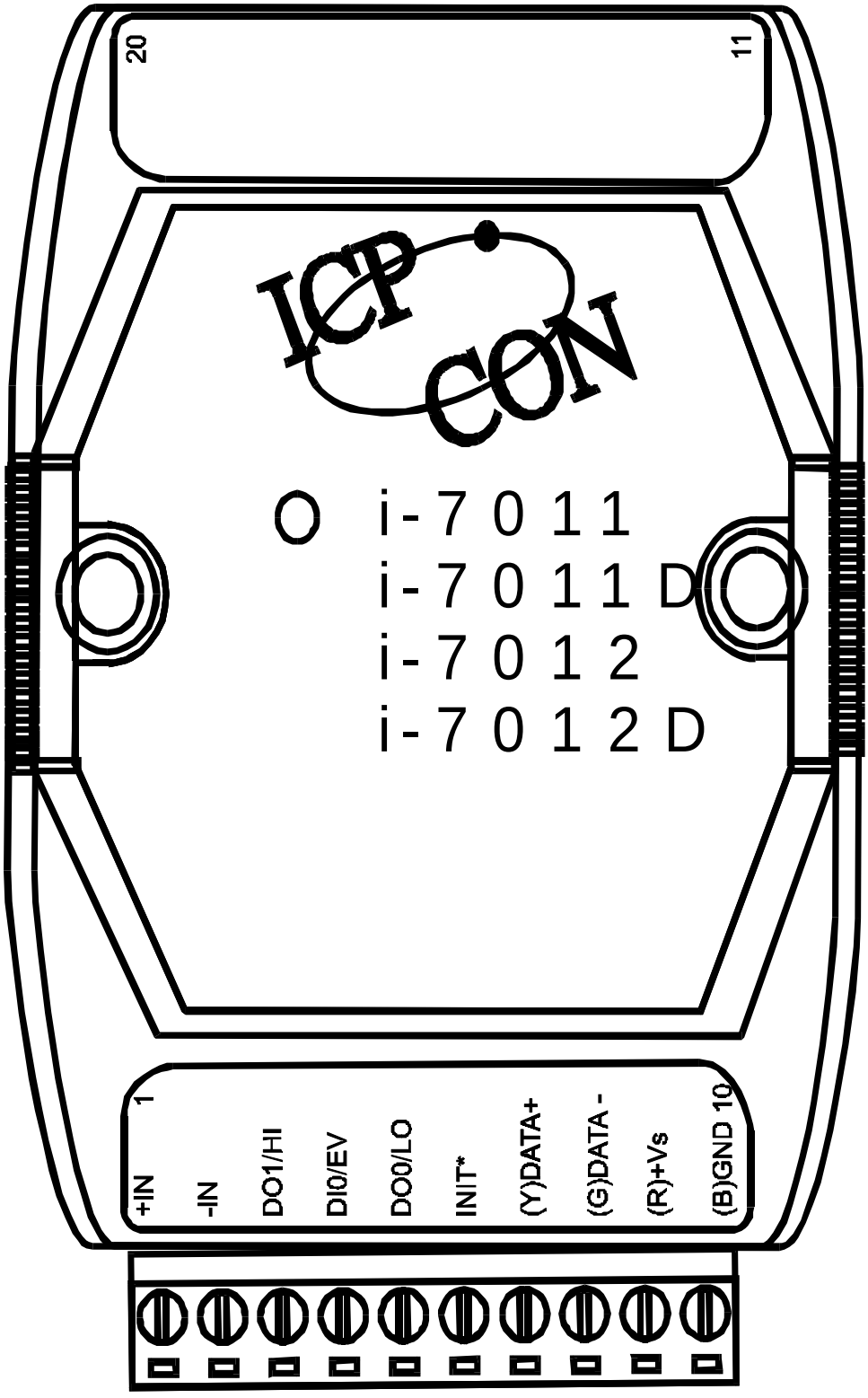
- isolated loop power, +15V, 20 mA max.
- linear mapping function
- built-in 125• 0.1% resistor for current measurement

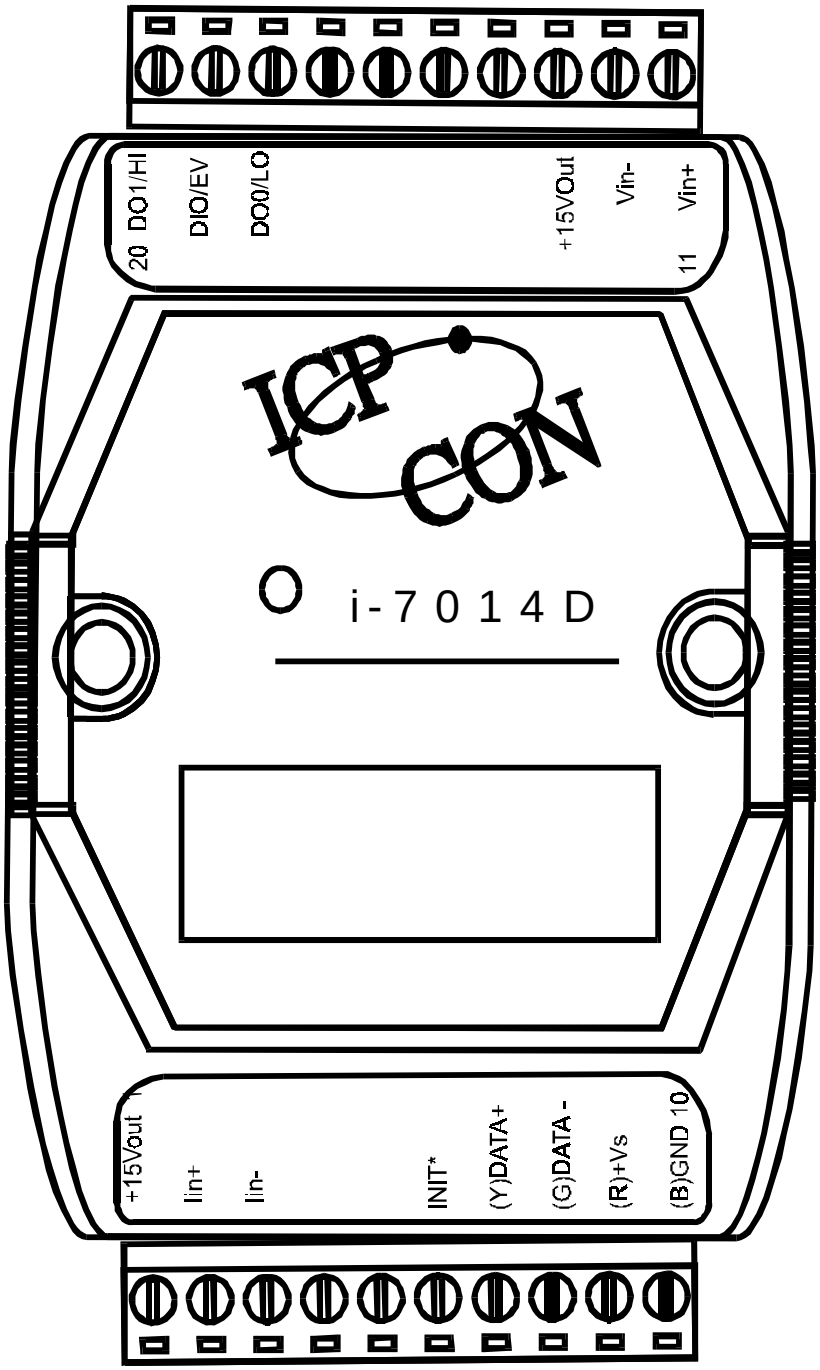
1.1 More Information

Refer to . 7000 Bus Converter User Manual. chapter 1 for more information as following:

1.1 7000 Overview
1.2 7000 Related Documentation
1.3 7000 Common Features
1.4 7000 System Network Configuration
1.5 7000 Dimension

1.2 Pin Assignment




20 DO1/HI
DIO/EV
DO0/LO
+15VOut
Vin-
Vin+
11
ICP CON
i-7014D
+15Vout
Iin+
Iin-
INIT*
(Y)DATA+
(G)DATA -
(R)+Vs
(B)GND 10

1.3Specifications

7011: thermocouple input module
7011D: 7011 with display
Analog Input

- Type: thermocouple, mV, V , or mA
- Channels: 1
- Thermocouple type:

J=0°C~760°C	S=500°C~~1750°C
K=0°C~1370°C	B=500°C~~1800°
T=-100°C~+400°C	N=-270°C~1300°C
E=0°C~1000°C	C=0°C~2320°C
R=500°C~1750°C	

- Voltage range: ±15mV, ±50mV, ±100mV, ±500mV, ±1V, ±2.5V
- Current range: ±20mA
- Sampling rate: 10 samples/sec
- Bandwidth: 4 Hz
- Accuracy: ±0.05% or better
- Zero drift: ±0.033 ppm/°C
- Span drift: 25ppm/°C
- CMR @ 50/60 Hz: 150 dB
- NMR @50/60 Hz: 100 dB
- Over voltage protection: ±10V

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(7011D)

Power consumption: 1.2W(7011), 2.2W(7012D)

7012: Analog Input Module
7012D: 7012 with display
7014D: Analog Input with Display

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 (7014D)

Digital Input: same as 7011

Digital Output: same as 7011

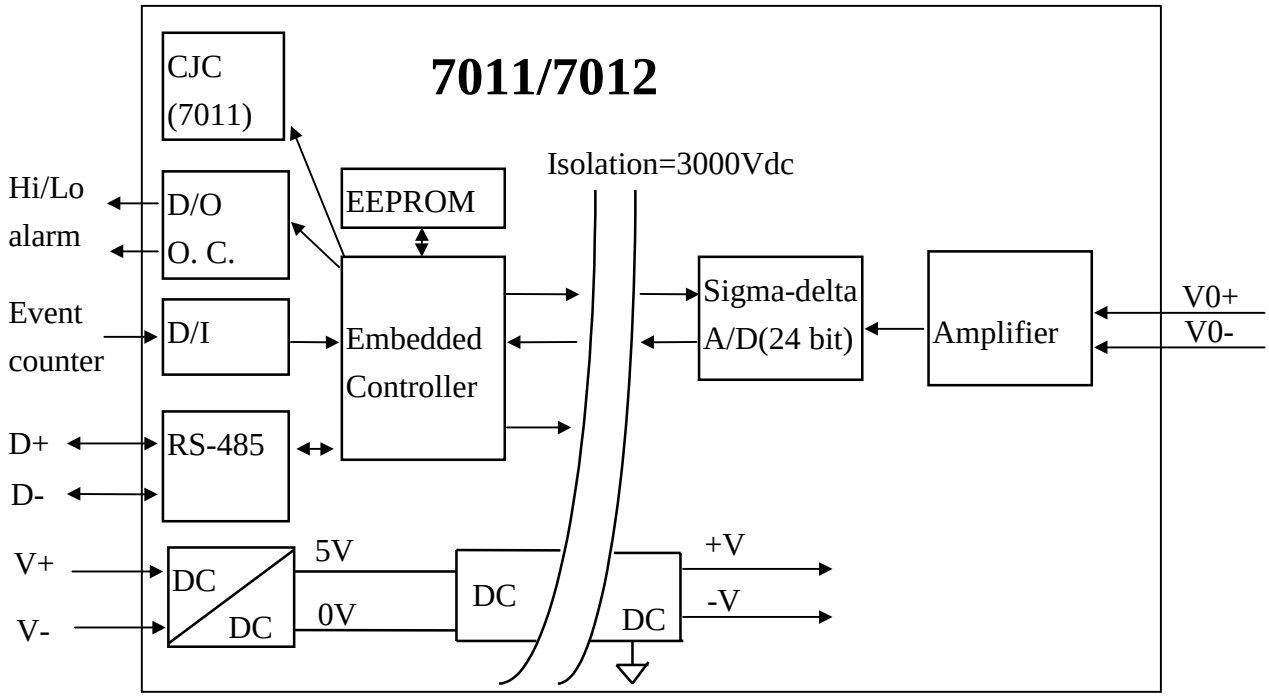
Display:

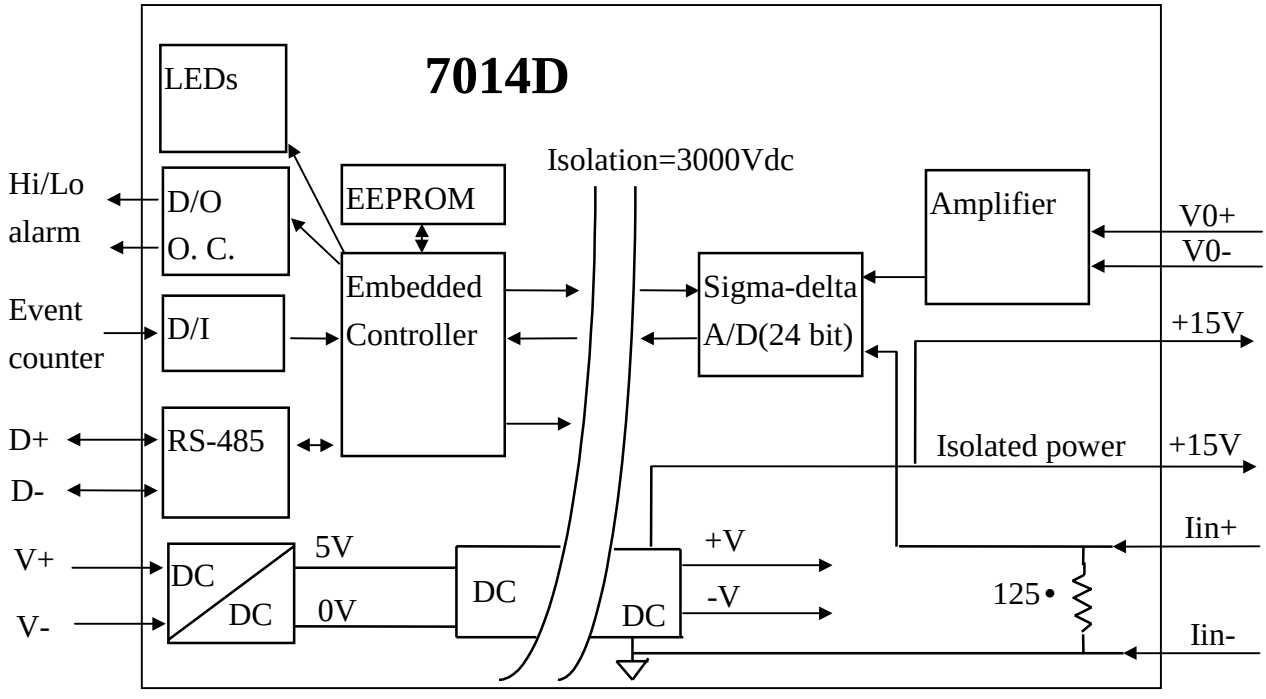
- LED: 4½ digit(7012D, 7014D)

Power consumption:

- 1.2W for 7012
- 2.2W for 7012D
- 2.2W for 7014D

1.4 Block Diagram

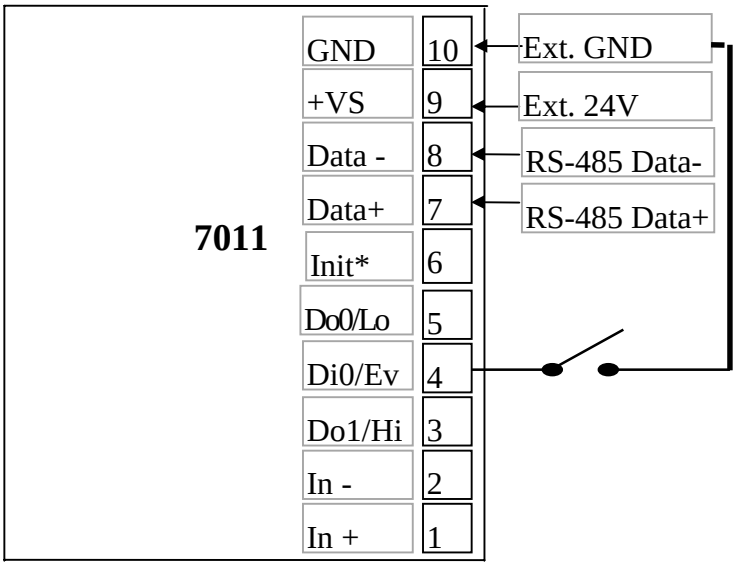




1.5 Application Wiring

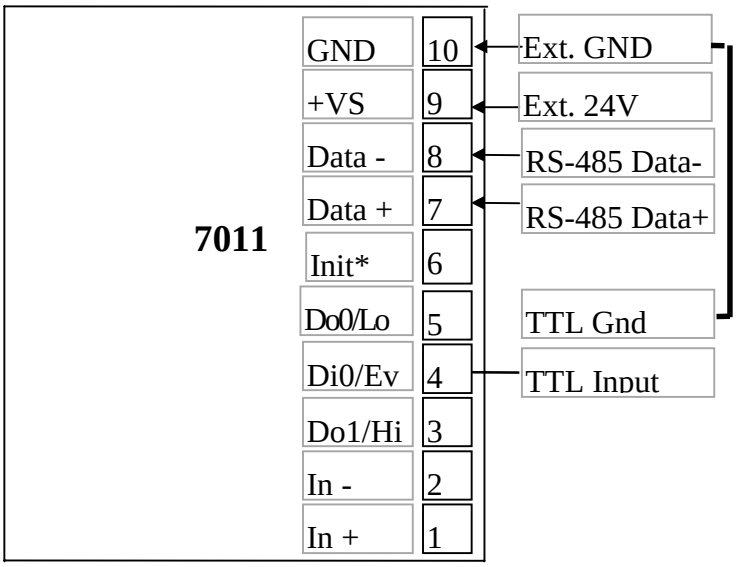
1.5.1 Dry Contact Input

- **For 7011/7011D/7012/7012D/7014D**



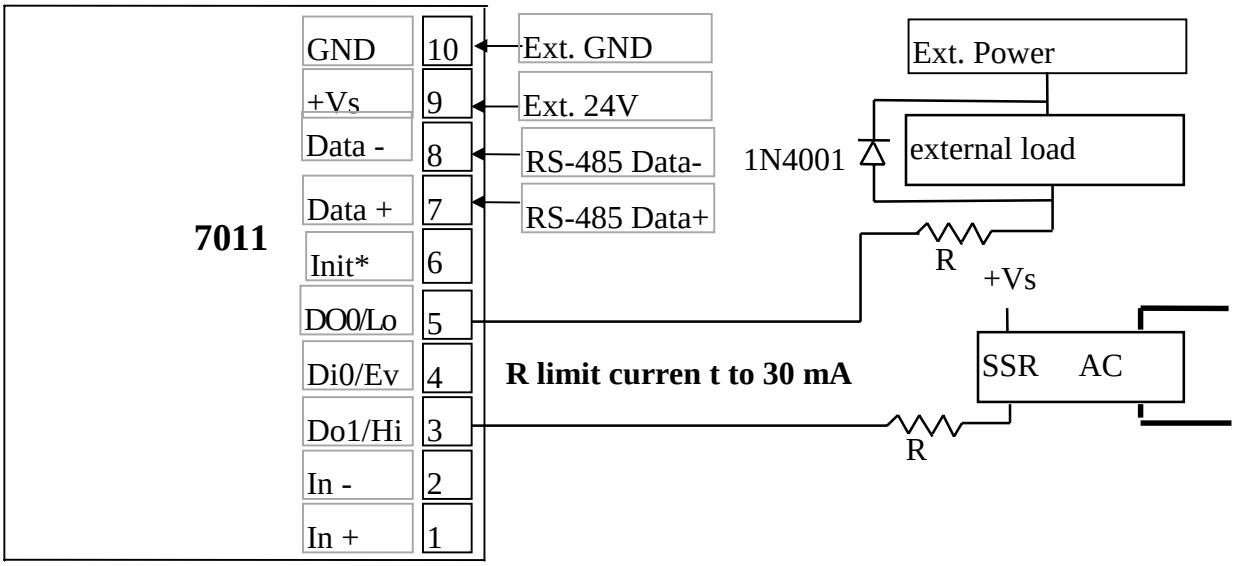
1.5.2 TTL Input

- **For 7011/7011D/7012/7012D/7014D**



1.5.3 Output Drive SSR & Load

- **For 7011/7011D/7012/7012D/7014D**

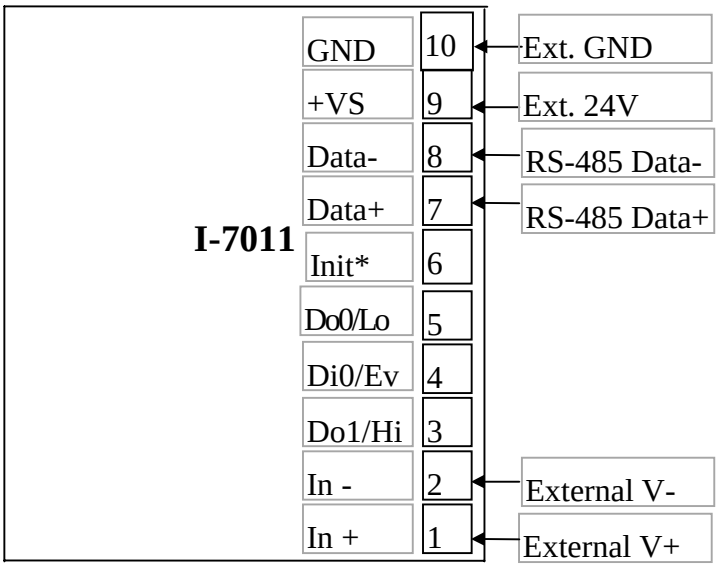


Note:

- If the external load is resistive load, the 1N4001 can be omitted. (transistor, lamp, resistor, …)
- If the external load is inductive load, the 1N4001 can.t be omitted. (relay, coil, …)

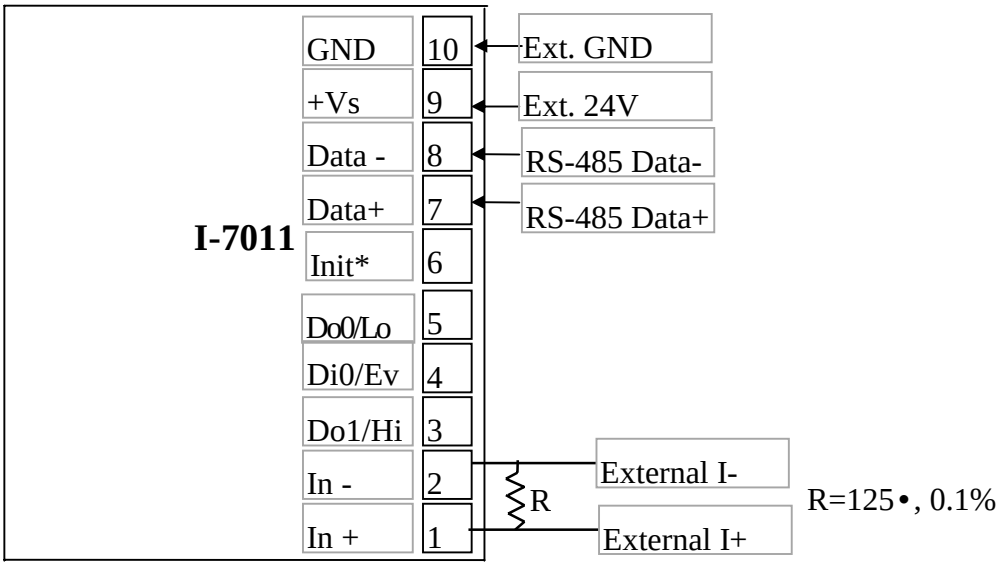
1.5.4 Voltage Input

- **For I-7011/7011D/7012/7012D**

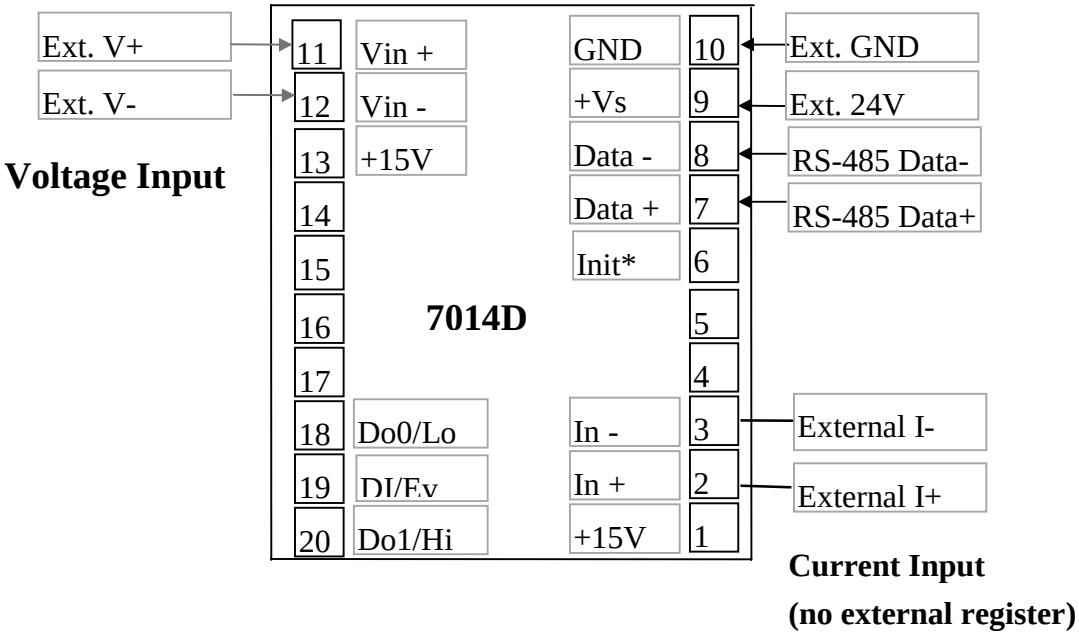


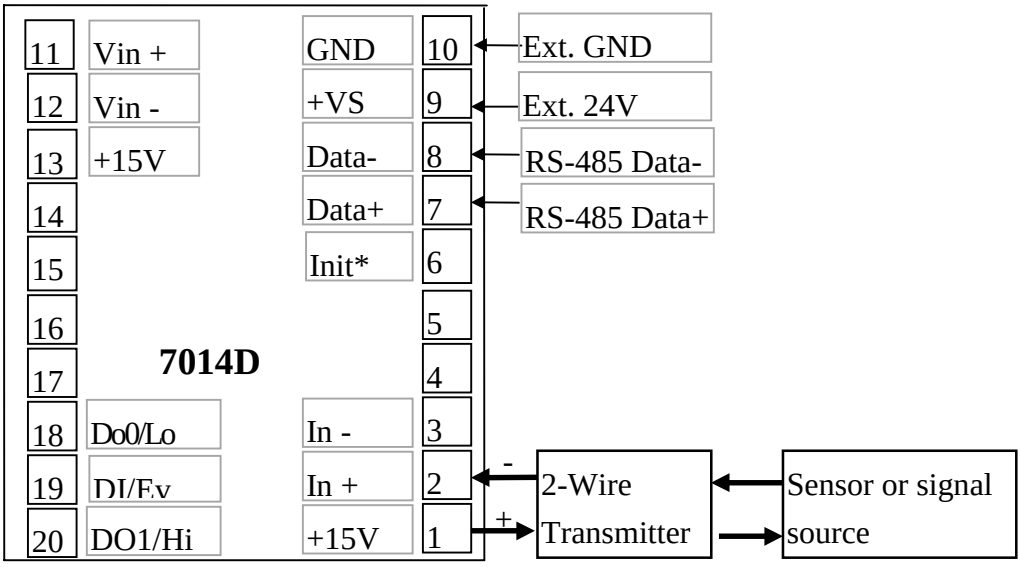
1.5.5 Current Input

- **For I-7011/7011D/7012/7012D**

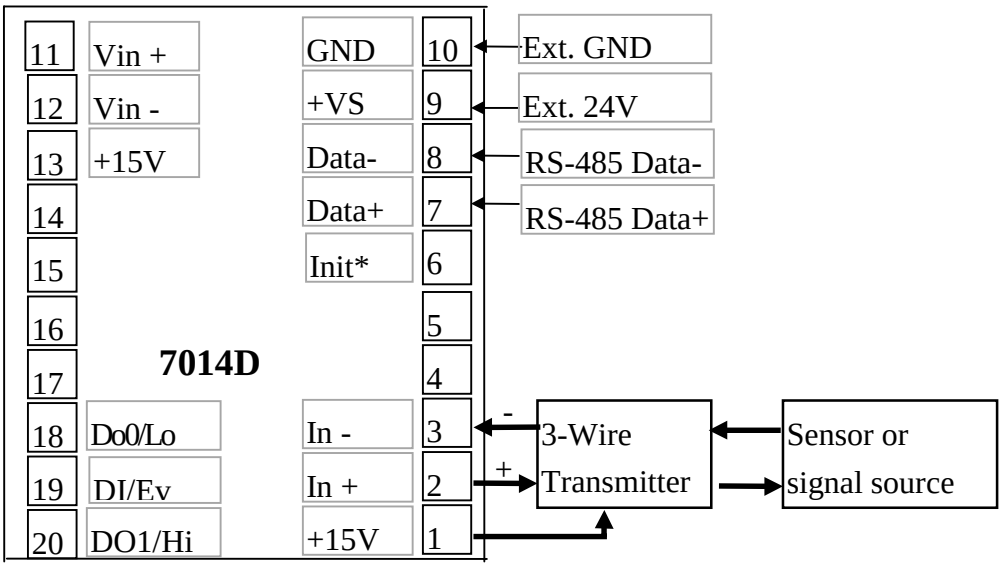


1.5.6 7014D Analog Input

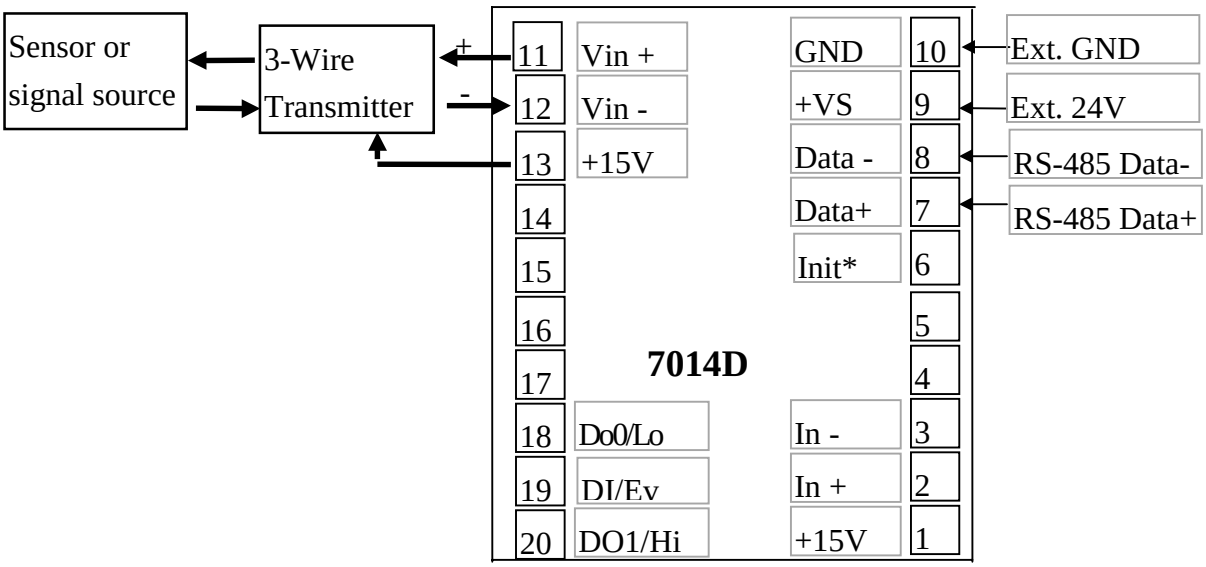




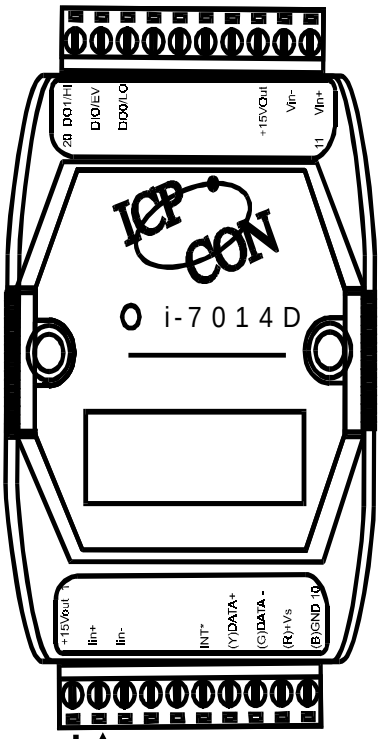
2-wire transmitter current input



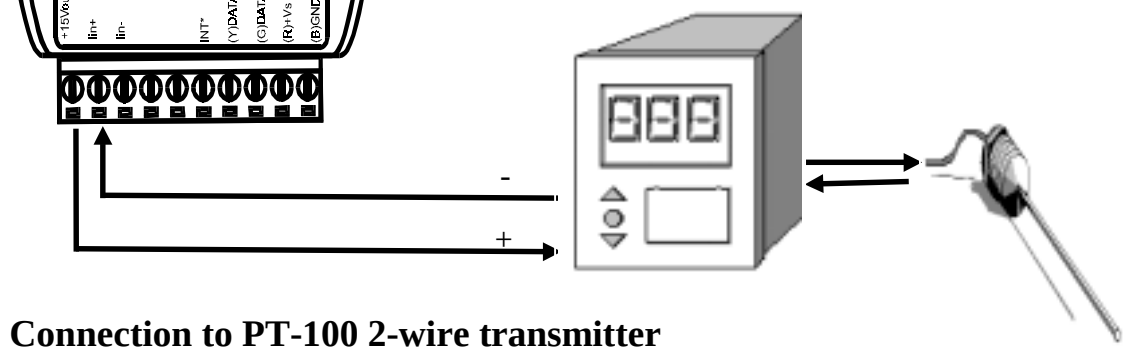
3-wire transmitter current input



3-wire transmitter voltage input



- **Enable linear mapping**
- **Temperature will be displayed in LED directly**
- **Refer to Sec. 3.9 for more information**



Connection to PT-100 2-wire transmitter

1.6 Quick Start

Refer to ．7000 Bus Converter User Manual． chapter-5 for the following functions:

- **module status unknown**(Sec. 5.1), **change address**(Sec. 5.2)
- **change baud rate**(Sec. 5.3), **checksum enable/disable**(Sec. 5.4)
- Wire connection(Sec 2.4)
- Test program **TEST.EXE**(Refer to ．NAP7000S User Manual． for details.

1.6.1 Voltage Input Measurememt

1. Refer to Sec. 1.5.4 for wire connection. Power on and run **test.exe**
2. press **2**
3. press **$012[Enter]** → Receive=!01080600
4. press **2**
5. press **#01[Enter]** → Receive=>+??.???
6. press **2**
7. press **%0101080602[Enter]**→ Receive=!01
8. press **2**
9. press **#01[Enter]** → Receive=>????
10. press **2**
11. press **%0101090600[Enter]**→ Receive=!01
12. press **2**
13. press **#01[Enter]** → Receive=+?.????

- step 3: the status of 7012 is +/- 10V, engineering unit format
- step 5: show the input voltage in engineering unit format
- step 7: change from engineering unit format to hex format
- step 9: show the input voltage in hex format
- step 11: change status to +/- 5V, engineering unit
- step 13: show the input voltage (+/- 5V range)

1.6.2 Current Input Measurement

1. Refer to Sec. 1.5.5 for wire connection, power on and run **test.exe**
2. press **2**
3. press **$012[Enter]** → Receive=!01080600
4. press **2**
5. press **%01010D0600[Enter]**→ Receive=!01
6. press **2**
7. press **#01[Enter]** → Receive=>+??.???

- step 3 : read the status of 7012
- step 5 : change status to +/- 20mA, engineering unit format
- step 7 : show the input current in engineering unit format

1.6.3 Thermocouple Measurement

1. Refer to Sec. 1.5.4 for wire connection, power on and run **test.exe**
2. press **2**
3. press **$012[Enter]** → Receive=!01050600
4. press **2**
5. press **%01010E0600[Enter]**→ Receive=!01
6. press **2**
7. press **#01[Enter]** → Receive=>+???.??

- step 3 : read the status of 7011
- step 5 : change to J-type thermocouple, engineering unit format
- step 7 : show the input temperature value in engineering unit format

1.6.4 7014D Linear Mapping Enable

1. Refer to Sec. 3.9 for more information about linear mapping.
2. Refer to Sec. 1.5.6 for 2-wire transmitter wire connection. Assume the 2-wire transmitter will connect to a PT-100 sensor. The measure range of PT-100 is from 0°C to 100°C. The output range of transmitter is from 4mA to 20mA. Therefore the linear mapping is from [4, 20] to [0, 100].
3. Power on and run **test.exe**
4. press **2**
5. press **%01010D0600[Enter]** → Receive=!01
6. press **2**
7. press **$016+04.000+20.000**→ Receive=!01
8. press **2**
9. press **$017+000.00+100.00**→ Receive=!01
10. press **2**
11. press **$013** → Receive=!01+04.000+20.000
12. press **2**
13. press **$015** → Receive=!01+000.00+100.00
14. press **2**
15. press **$01A1** → Receive=!01
16. press **2**
17. press **$01B** → Receive=!011
18. press **2**
19. press **#01[Enter]** → Receive=>+???.??

- step 5: change this 7014D to type 0x0D.
- step 6-9: set linear mapping [4, 20] to [0, 100]
- step 10-13: linear mapping read back
- step 15: enable linear mapping
- step 17: linear mapping status read back, now it is enable
- step 19: read the physical temperature(after linear mapping).

1.7 Default Setting

The default setting is given as following:

- address=01
- baud rate=9600
- checksum disable
- data=1 start+8 data+1 stop(no parity)
- type 08=±10V input range (for 7012, 7012D, 7014D)
  05=±2.5V input range (for 7011, 7011D)

1.8 Calibration

Zero/Span Table for 7012, 7012D 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%

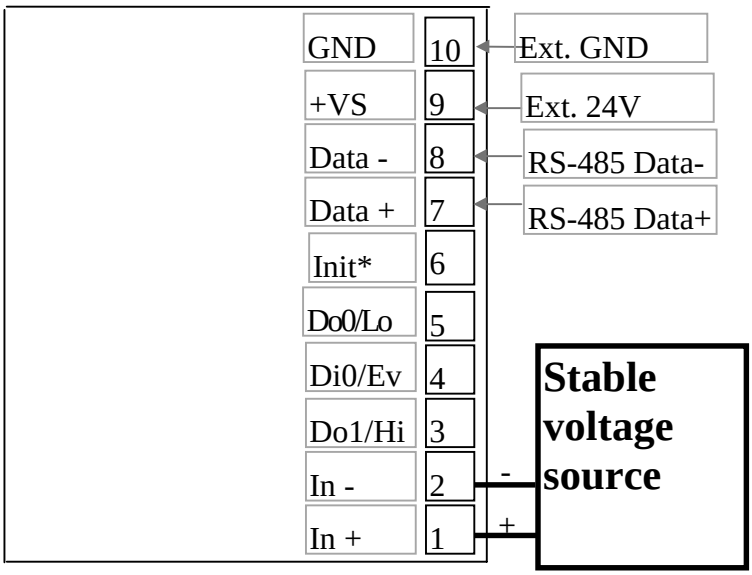
Zero/Span Table for 7014D Calibration.

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

Zero/Span Table for 7011, 7011D Calibration.

Input Range Code	Input Range	Zero Voltage	Span Voltage
00	±15 mV	0V	15 mV
01	±50 mV	0V	50 mV
02	±100 mV	0V	100 mV
03	±500 mV	0V	500 mV
04	±1 V	0V	1 V
05	±2.5 V	0V	2.5 V
06	±20mA	0V or 0mA with 125• 0.1%	2.5V or 20mA with 125• 0.1%
0E	J-type(0~760°C)	0mV	42.922mV
0F	K-type(0~1320°C)	0mV	53.093mV
10	T-type(-100~-400°C)	0mV	20.869mV
11	E-type(0~1000°C)	0mV	76.358mV
12	R-type(500~1750°C)	0mV	21.108mV
13	S-type(500~1750°C)	0mV	18.698mV
14	B-type(500~1800°C)	0mV	13.814mV
15	N-type(-270~1300°C)	0mV	47.502mV
16	C-type(0~2320°C)	0mV	37.107mV

1.8.1 7012, 7012D 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

The calibration steps of type-08 are given as following:

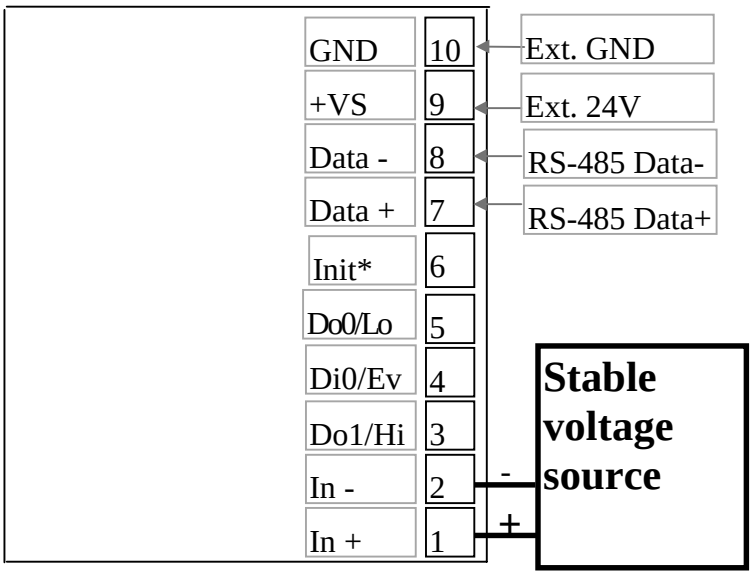
1. Run TEST.EXE
2. press **2**
3. press **%0101080600[Enter]**→ Receive=!01
4. Apply ZERO-Voltage to Channel_0
5. press **2**
6. press **$011[Enter]** → Receive=!01
7. Apply SPAN-Voltage to Channel_0
8. press **2**
9. press **$010[Enter]** → Receive=!01
10. repeat step-4 to step-9 three times.

4→ 5→ 6
Zero Calibration

7→ 8→ 9
Span Calibration

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

1.8.2 7011, 7011D 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-00 calibration
Step 4: Perform type-01 calibration
..
Step 18: Perform type-16 calibration

The calibration steps of type-00 are given as following:

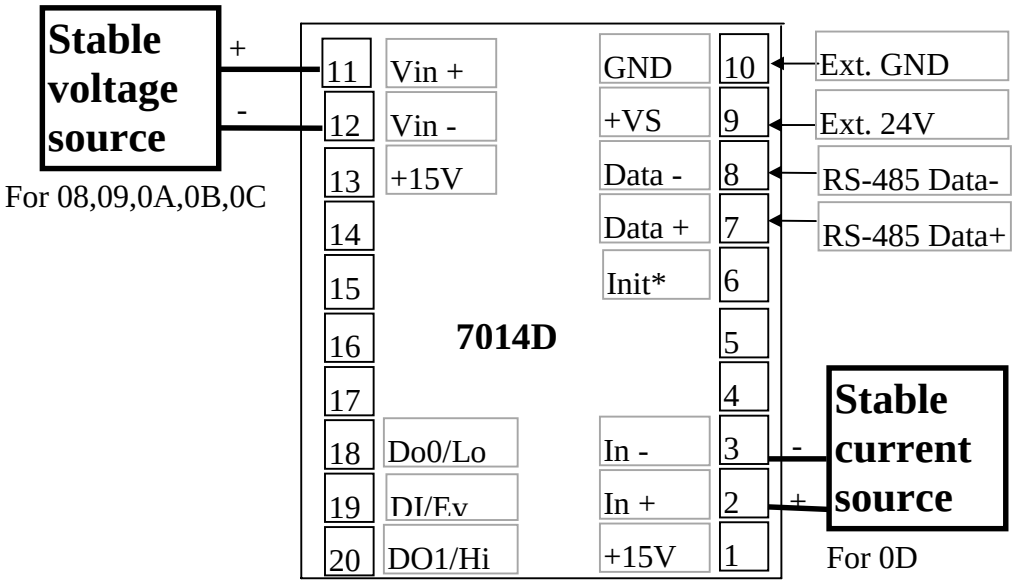
1. Run TEST.EXE
2. press **2**
3. press **%0101000600[Enter]**→ Receive=!01
4. Apply ZERO-Voltage to Channel_0
5. press **2**
6. press **$011[Enter]** → Receive=!01
7. Apply SPAN-Voltage to Channel_0
8. press **2**
9. press **$010[Enter]** → Receive=!01
10. repeat step-4 to step-9 three times.

4→ 5→ 6
Zero Calibration

7→ 8→ 9
Span Calibration

NOTE: calibration steps are all the same for type-00 to type-16. Only the SPAN-Voltage is different.

1.8.3 7014D 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

The calibration steps of type-08 are given as following:

1. Run TEST.EXE
2. press **2**
3. press **%0101080600[Enter]→** Receive=!01
4. Apply ZERO-Voltage to Channel_0
5. press **2**
6. press **$011[Enter]** → Receive=!01
7. Apply SPAN-Voltage to Channel_0
8. press **2**
9. press **$010[Enter]** → Receive=!01
10. repeat step-4 to step-9 three times.

4→ 5→ 6
Zero Calibration

7→ 8→ 9
Span Calibration

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

1.9 Tables

Configuration Code Table : CC(for all)

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 all)

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 (for 7012, 7012D, 7014D)

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

Configuration Code Table : TT (for 7011, 7011D)

TT	Input Range
00	±15 mV
01	±50 mV
02	±100 mV
03	±500 mV
04	±1 V
05	±2.5 V
06	±20mA
0E	J-type(0°C ~ 760°C)
0F	K-type(0°C ~ 1320°C)
10	T-type(-100°C ~ 400°C)
11	E-type(0°C ~ 1000°C)
12	R-type(500°C ~ 1750°C)
13	S-type(500°C ~ 1750°C)
14	B-type(500°C ~ 1800°C)
15	N-type(-270°C ~ 1300°C)
16	C-type(0°C ~ 2320°C)

Data Format Table (data): (for 7012, 7012D, 7014D)

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

Data Format Table (data): (for 7011, 7011D)

TT	Range	Format	+FSR	Zero	-FSR
00	±15mV	Engineering Unit	+15.000	±00.000	-15.000
		% of FSR	+100.00	±000.00	-100.00
		2.s complement	7FFF	0000	8000
01	±50mV	Engineering Unit	+50.000	±00.000	-50.000
		% of FSR	+100.00	±000.00	-100.00
		2.s complement	7FFF	0000	8000
02	±100mV	Engineering Unit	+100.00	±000.00	-100.00
		% of FSR	+100.00	±000.00	-100.00
		2.s complement	7FFF	0000	8000
03	±500mV	Engineering Unit	+500.00	±000.00	-500.00
		% of FSR	+100.00	±000.00	-100.00
		2.s complement	7FFF	0000	8000
04	±1V	Engineering Unit	+1.0000	±0.0000	-1.0000
		% of FSR	+100.00	±000.00	-100.00
		2.s complement	7FFF	0000	8000
05	±2.5V	Engineering Unit	+2.5000	±0.0000	-2.5000
		% of FSR	+100.00	±000.00	-100.00
		2.s complement	7FFF	0000	8000
06	± 20 mA	Engineering Unit	+20.000	±00.000	-20.000
		% of FSR	+100.00	±000.00	-100.00
		2.s complement	7FFF	0000	8000

0E	J-type thermocouple	Engineering Unit	+760.00	+000.00	
		% of FSR	+100.00	±000.00	
		2.s complement	7FFF	0000	
0F	K-type thermocouple	Engineering Unit	+1000.0	+0000.0	
		% of FSR	+100.00	±000.00	
		2.s complement	7FFF	0000	
10	T-type thermocouple	Engineering Unit	+400.00	±000.00	-100.00
		% of FSR	+100.00	±000.00	-100.00
		2.s complement	7FFF	0000	8000
11	E-type thermocouple	Engineering Unit	+1000.0	+0000.0	
		% of FSR	+100.00	±000.00	
		2.s complement	7FFF	0000	
12	R-type thermocouple	Engineering Unit	+1750.0	+0000.0	
		% of FSR	+100.00	±000.00	
		2.s complement	7FFF	0000	
13	S-type thermocouple	Engineering Unit	+1750.0	+0500.0	
		% of FSR	+100.00	+028.57	
		2.s complement	7FFF	2492	
14	B-type thermocouple	Engineering Unit	+1800.0	+0500.0	
		% of FSR	+100.00	+027.77	
		2.s complement	7FFF	2381	
15	N-type thermocouple	Engineering Unit	+1300.0	+0000.0	-0270.0
		% of FSR	+100.00	±000.00	-100.00
		2.s complement	7FFF	0000	8000
16	C-type thermocouple	Engineering Unit	+2320.0	+0000.0	
		% of FSR	+100.00	±000.00	
		2.s complement	7FFF	0000	

2. Command Set

Command Set Table

Command	Response	Description	Reference
%AANNTTCCFF	!AA	Set module configuration	Sec. 2.1
#**	No Response	Synchronized Sampling	Sec. 2.2
#AA	>(data)	Read analog input	Sec. 2.3
$AA0	!AA	Perform span calibration	Sec. 2.4
$AA1	!AA	Perform zero calibration	Sec. 2.5
$AA2	!AATTCCFF	Read configuration	Sec. 2.6
$AA3(7011/11D)	>S(data)	Read CJC value	Sec. 2.7
$AA3(7014D)	!AA(LO)(HI)	Read source linear mapping	Sec. 2.8
$AA4	!AA(data)	Read Synchronized Data	Sec. 2.9
$AA5(7014D)	!AA(LO)(HI)	Read target linear mapping	Sec. 2.10
$AA6(LO)(HI) (7014D)	!AA	Write source linear mapping	Sec. 2.11
$AA7(LO)(HI) (7014D)	!AA	Write target linear mapping	Sec. 2.12
$AA8V (7011D/12D/14D)	!AA	Select LED Configuration	Sec. 2.13
$AA9SCCCC (7011/11D)	!AA	Set CJC Offset Value	Sec. 2.14
$AA9S(data) (7012D/14D)	!AA	Send LED Display	Sec. 2.15
$AAAV (7014D)	!AA	Enable/Disable Linear Mapping	Sec. 2.16
$AAB(7011/11D)	!AAS	Thermocouple open detection	Sec. 2.17
$AAB(7014D)	!AAS	Linear Mapping Status	Sec. 2.18
$AAF	!AA(data)	Read firmware number	Sec. 2.19
$AAM	!AA(data)	Read the module name	Sec. 2.20
$AAZS(data) (7011D)	!AA	Send LED display	Sec. 2.21

Command Set Table

@AADI	!AAS0D0I	Read DIO & alarm status	Sec. 2.22
@AADO0D	!AA	Set D/O	Sec. 2.23
@AAEAT	!AA	Enable alarm	Sec. 2.24
@AAHI(data)	!AA	Set high alarm	Sec. 2.25
@AALO(data)	!AA	Set low alarm	Sec. 2.26
@AADA	!AA	Disable alarm	Sec. 2.27
@AACA	!AA	Clear latch alarm	Sec. 2.28
@AARH	!AA(data)	Read high alarm	Sec. 2.29
@AARL	!AA(data)	Read low alarm	Sec. 2.30
@AARE	!AA(data)	Read event counter	Sec. 2.31
@AACE	!AA	Clear event counter	Sec. 2.32
~**	No Response	Host OK	Sec. 2.33
~AA0	!AASS	Read Module Status	Sec. 2.34
~AA1	!AA	Reset Module Status	Sec. 2.35
~AA2	!AATT	Read Host Watchdog Timer	Sec. 2.36
~AA3ETT	!AA	Enable Host Watchdog Timer	Sec. 2.37
~AA4	!AAVV00	Read safe value	Sec. 2.38
~AA5	!AA	Set /safe value	Sec. 2.39
~AAO(name)	!AA	Set module name	Sec. 2.40

2.1 %AANNTTCCFF

7011/7011D/7012/7012D/7014D

- **Description**: Set the configuration of module.

- **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 Sec. 1.9
  CC=baud rate code, refer to Sec. 1.9
  FF=status code, refer to Sec. 1.9
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:
  command: %0102080600(cr)
  response : !02(cr)
  address 01 is configured to a new address 02, ±10V input

  command: %0202090600(cr)
  response : !02(cr)
  change to ±5V analog input

Refer to "7000 Bus Converter User Manual" chapter-5 for the following functions:
- **module status unknown**(Sec. 5.1), **change address**(Sec. 5.2)
- **change baud rate**(Sec. 5.3), **checksum enable/disable**(Sec. 5.4)

2.2 #**

- **Description**: Order all input module, digital and analog, to sample all their input data immediately and store these data in the internal register of module. Later the host computer can read these data one by one by the command **$AA4, read synchronized data.**
- **Syntax**: #**[chk](cr)
  # is a delimiter character
  * is a command character
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D
- **Response**: no response
- **Example**:

command: #**(cr)	Order all modules perform synchronized sampling
response: no response	
command: $014(cr)	Read these synchronized sampling data one-by one. In this example, read module-01, 02, 03. © is a character depended on module.s wiring and command
response: !1©©©©©©©©(cr)	
command: $024(cr)	
response: !1©©©©©©©©(cr)	
command: $034(cr)	
response: !1©©©©©©©©(cr)	

NOTE : What.s . synchronize sampling. ?

The host computer can send only one command string once a time. If there are two modules, the host computer must send and receive the module-1 command then the module-2 command. **So there is a time delay between these two commands.** The . synchronize sampling. command is designed for all input modules. When receiving **#**[0x0D]**, synchronized sampling command, **all the input modules in the RS-485 network will perform the input function at the same time and store these values into the module.s memory.** Then the host computer can send out the . $AA4, read synchronize data. command to read these data separately.

2.3 #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 disable → 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 Sec. 1.9
  [chk]=2-character checksum, if checksum disable → 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

Note: If the user use 7014D & enable the linear mapping function, the #AA will read the value after linear mapping.

2.4 $AA0

- **Description**: Perform the SPAN calibration. Refer to Sec. 1.8.1, Sec. 1.8.2 and Sec. 1.8.3 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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example:**
  command: $010(cr)
  response : !01(cr)

  address 01 perform SPAN calibration

  command: $020(cr)
  response : !02(cr)

  address 02 perform SPAN calibration

2.5 $AA1

- **Description**: Perform the ZERO calibration. Refer to Sec. 1.8.1, Sec. 1.8.2 and Sec. 1.8.3 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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → 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

2.6 $AA2

7011/7011D/7012/7012D/7014D

- **Description**: Read the configuration of module.
- **Syntax**: $AA2[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  TT, CC, FF: refer to Sec. 1.9
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D
- **Example**:
  command: $012(cr)
  response : !01080600(cr)

  address 01, ±10V, 9600 BPS, checksum disable, engineering unit

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

  address 02, ±2.5V, 19200 BPS, checksum disable, engineering unit

NOTE: If the user use %AANNTTCCFF command 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 EEPROM data of 7000 can be read infinite times and can be written about 100,000 times max.** Therefore the user should not change configuration code often for testing.
The $AA2 command is used to read EEPROM data only, therefore the user can send this command to 7000 module infinitely.

2.7 $AA3

7011/7011D

- **Description**: Read current CJC value. Refer to Sec. 3.6 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 disable → 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 a invalid command
  AA=2-character HEX module address
  S=+ or -
  (data)=CJC value
  [chk]=2-character checksum, if checksum disable → 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

2.8 $AA3

7014D

- **Description**: Read the [low, high] value of source linear mapping value. Refer to Sec. 3.9 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 disable → 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 a 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 disable → no [chk]
  (cr)=0x0D

- **Example**:

command: $013(cr)
response : !01+04.000+20.000(cr)

Source linear mapping read back = [4.0, 20.0]

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

Source linear mapping read back = [0.0, 100.0]

Note: the data format of (HI) & (LO) is the same as current configuration. Refer to . Data Format Table (data). in Sec. 1.9 for details.

2.9 $AA4

7011/7011D/7012/7012D/7014D

- **Description**: Read the 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 disable → 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 a invalid command
  AA=2-character HEX module address
  S=1=first reading, S=0=not first reading
  (data) = refer to Sec. 1.9
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D
- **Example** :

command: $01M(cr)
response : !017011(cr)
command: $02M(cr)
response : !027011D(cr)
command: #**
response : No Responde
command: $014(cr)
response : !1+123.45(cr)
command: $014(cr)
response : !00+123.45(cr)
command: $024(cr)
response : !1-123.45(cr)
command: $024(cr)
response : !0-123.45(cr)

(1) address-01 is 7011.
(2) address-02 is 7011D
(3) perform synchronized sampling

(1) synchronized data = +123.45, first time
(2) synchronized data = +123.45, not first time

(1) synchronized data = -123.45, first time
(2) synchronized data = -123.45, not first time

2.10 $AA5

7014D

- **Description**: Read the [low, high] value of target linear mapping value. Refer to Sec. 3.9 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 disable → 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 a 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 disable → no [chk]
  (cr)=0x0D

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

  Targe linear mapping read back = [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 giving as following:

- **first char is + or -**
- **the next 6 char will includes one decimal point**
- **min. value → -19999.**
- **max. value → +19999.**

2.11 $AA6(LO)(HI)

7014D

- **Description**: Write the [low, high] value of source linear mapping value. Refer to Sec. 3.9 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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → 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 Sec. 1.9 for details.

2.12 $AA7(LO)(HI)

7014D

- **Description**: Write the [low, high] value of target linear mapping value. Refer to Sec. 3.9 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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → 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 targe linear mapping = [0.0, 100.0]

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

- **first char is + or -**
- **the next 6 char will includes one decimal point**
- **min. value → -19999.**
- **max. value → +19999.**

2.13 $AA8V

7011D/7012D/7014D

- **Description**: Select LED Configuration.

- **Syntax**: $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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

command: $0181(cr)
response : !01(cr)

Module will control LED
(the analog input value will be automatically shown on LED.)

command: $0282(cr)
response : !02(cr)
command: $029+040.00(cr)
response : !02(cr)

Host will control LED

Host send +040.00 to LED

2.14 $AA9SCCCC

7011/7011D

- **Description**: Set CJC offset value. Refer to Sec. 3.6 for more information.

- **Syntax**: $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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

  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

2.15 $AA9S(data)

7012D/7014D

- **Description**: Send LED display.

- **Syntax**: $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.
  Min. → -19999.
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

  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

2.16 $AAAV

7014D

- **Description**: Enable/disable linear mapping. Refer to Sec. 3.9 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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

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

  Disable linear mapping.

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

  Enable linear mapping.

2.17 $AAB

7011/7011D

- **Description**: Open detection of thermocouple input.

- **Syntax**: $AAB[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  S=0 → thermocouple is close, S=1 → thermocouple is open
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

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

  Thermocouple is close.

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

  Thermocouple is open.

NOTE: this command will be available in version 2.0

2.18 $AAB

I-7014D

- **Description**: Linear mapping status read back. Refer to Sec. 3.9 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 disable → 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 a invalid command
  AA=2-character HEX module address
  S=0 → linear mapping is disable
  1 → linear mapping is enable
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

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

  Linear mapping is disable.

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

  Linear mapping is disable.

2.19 $AAF

7011/701D/7012/7012D/7014D

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

- **Syntax**: $AAF[chk](cr)
  $ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  data=5-character for version number
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

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

  module 01 version A2.0

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

  module 02 version A3.0

2.20 $AAM

7011/701D/7012/7012D/7014D

- **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 disable → 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 a invalid command
  AA=2-character HEX module address
  data=4-character for module name
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

command: $01M(cr)
response : !017011(cr)

Module name of 01 is 7011

command: $02M(cr)
response : !027012(cr)

Module name of 02 is 7012

command: $03M(cr)
response : !037011D(cr)

Module name of 03 is 7011D

command: $04M(cr)
response : !047012D(cr)

Module name of 04 is 7012D

2.21 $AAZS(data)

7011D

- **Description**: Send LED display.

- **Syntax**: $AAZS(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.
  Min. → -19999.
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

  command: $01Z+19999.(cr)
  response : !01(cr)

  Show max = +19999.

  command: $02Z-19999.(cr)
  response : !02(cr)

  Show min = -19999.

  command: $03Z+12.345(cr)
  response : !03(cr)

  Show display = +12.345

2.22 @AADI

- **Description**: Read the status of DIO & alarm.

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

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

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

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

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

2.23 @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 disable → no [chk]
  (cr)=0x0D
- **Response**: valid command → !AA[chk](cr)
  invalid command → ?AA[chk](cr)
  alarm is enable → ?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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → 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 enable, the DO1 & DO2 will be always controlled by module. Therefore the following D/O commands will be ignored.

- **power-on value is changed to hi/lo condition immediately**
- **safe value is ignored**
- **the @AADO0D command is ignored.**

2.24 @AAEAT

7011/7011D/7012/7012D/7014D

- **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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → 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 enable, the DO1 & DO2 will be always controlled by module. Therefore the following D/O commands will be ignored.

- **power-on value is changed to hi/lo condition immediately**
- **safe value is ignored**
- **the @AADO0D command is ignored.**

2.25 @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 Sec. 1.9.
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → 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

2.26 @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 Sec. 1.9.
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → 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

2.27 @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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D
- **Example**:

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

Alarm disable.

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

Alarm disable.

2.28 @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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D
- **Example**:

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

Clear latch alarm.

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

Clear latch alarm.

2.29 @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 disable → 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 a invalid command
  AA=2-character HEX module address
  data = engineering unit format. Refer to Sec. 1.9.
  [chk]=2-character checksum, if checksum disable → 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

2.30 @AARL

7011/7011D/7012/7012D/7014D

- **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 disable → 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 a invalid command
  AA=2-character HEX module address
  data= engineering unit format. Refer to Sec. 1.9.
  [chk]=2-character checksum, if checksum disable → 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

2.31 @AARE

7011/7011D/7012/7012D/7014D

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

- **Syntax**: @AARE[chk](cr)
  @ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  data=5-character HEX value, from 00000 to 65535
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D
- **Example**:

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

Event counter=1.

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

Event counter=12345.

2.32 @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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → 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

2.33 ~**

- **Description**: Host send this command to tell all modules . Host is OK. . Refer to Sec. 3.2, Sec. 3.3 and Sec. 3.8 for more information.

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

- **Response**: no response

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

2.34 ~AA0

7011/7011D/7012/7012D/7014D

- **Description**: Read the module status. The module status will be latched until ~AA1 command is sent**. If the host watchdog is enable and the host is down, the module status will be set to 4. If the module status=4, all output command will be ignored.** Refer to Sec. 3.2, Sec. 3.3 and Sec. 3.8 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 disable → 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 a invalid command
  AA=2-character HEX module address
  SS=2-character HEX status value as following:
  Bit_0, Bit_1 = reserved
  Bit_2 = 0 → OK,
  1 → host watchdog failure
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

command: ~010(cr)
response : !0100(cr)

Status of module 01 is OK

command: ~020(cr)
response : !0204(cr)

Module status = 4 → host watchdog failure → HOST is down now

2.35 ~AA1

7011/7011D/7012/7012D/7014D

- **Description**: Reset the module status. The module status will be latched until ~AA1 command is sent. **If the module status=4, all output command will be ignored.** Therefore the user should read the module status first to make sure that the module status is 0. If the module status is not 0, only ~AA1 command can clear the module status. Refer to Sec., 3.2, 3.3 and 3.8 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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

command: ~010(cr)	
response : !0104(cr)	module status=0x04 → host is
command: @01DO00(cr)	
response : !(cr)	Output command is ignored
command: ~011(cr)	
response : !01(cr)	clear module status
command: ~010(cr)	
response : !0100(cr)	module status=0x00
command: @01DO00(cr)	
response : >(cr)	Output command is OK

2.36 ~AA2

7011/7011D/7012/7012D/7014D

- **Description**: Read the status and timer value of host watchdog. The host watchdog timer is designed for host watchdog. When the host watchdog is enable, 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 restart. Use ~AA3ETT to enable/disable/setting the host watchdog timer. Refer to Sec. 3.8 for more information.

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

- **Example**:

  command: ~012(cr)
  response : !01000(cr)

  Host watchdog timer of module 01 is disable

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

  host watchdog timer of module 02 is enable and =0.1*10 =1 second.

2.37 ~AA3ETT

7011/7011D/7012/7012D/7014D

- **Description**: Enable/disable the timer value of host watchdog. The host watchdog timer is designed for software host watchdog. When the software host watchdog is enable, 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 restart. Use ~AA2 to read the host watchdog status & value. Refer to Sec. 3.8 for more information.

- **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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:
  command: ~013000(cr)
  response : !01(cr)

  disable host watchdog timer of module 01

  command: ~02310A(cr)
  response : !02(cr)

  host watchdog timer of module 02 is enable and equal to 0.1*10 =1 second.

2.38 ~AA4

7011/7011D/7012/7012D/7014D

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

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

- **Example**:

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=on

2.39 ~AA5

7011/7011D/7012/7012D/7014D

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

- **Syntax** : ~AA5PPSS[chk](cr)
  ~ is a delimiter character
  AA=2-character HEX module address, from 00 to FF
  [chk]=2-character checksum, if checksum disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → 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

2.40 ~AAO(name)

7011/7011D/7012/7012D/7014D

- **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 disable → 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 a invalid command
  AA=2-character HEX module address
  [chk]=2-character checksum, if checksum disable → no [chk]
  (cr)=0x0D

- **Example**:

  command: $01M(cr)
  response : !017050(cr)
  command: ~01O8050(cr)
  response : !01(cr)

  Change module name from 7050 to 8050

  command: $01M(cr)
  response : !017080D(cr)
  command: ~01O8080D(cr)
  response : !01(cr)

  Change module name from 7080D to 8080D

Note: This command is designed for OEM/ODM user. The user can use it to change the module name for other purpose.

3. Operations Principle & Application Notes

3.1 INIT*_pin Operation Principle

All 7000 modules contain an EEPROM to store configuration information. Therefore the user is difficult to find out the status of the 7000 modules. The user can connect the INIT*_pin to GND_pin and power on the module. The 7000 modules will **go to the factory default setting without change the EEPROM data.** The factory default setting is given as following:

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

If the user disconnect the INIT*_pin and GND_pin, the 7000 module will be auto configured according to the EEPROM data. The user is easy to find the EEPROM configuration data in the default setting. The steps are shown as following:

Step 1: power off and connect INIT*_pin to GND_pin
Step 2: power on
Step 3: send command string **$002[0x0D]** to the module, the module will return back the EEPROM data.
Step 4: record the EEPROM data of this 7000 module
Step 5: power off and disconnect INIT*_pin and GND_pin
Step 6: power on

Refer to .7000 Bus Converter User Manual. Sec. 5.1 for more information.

3.2 Dual WatchDog Operation Principle

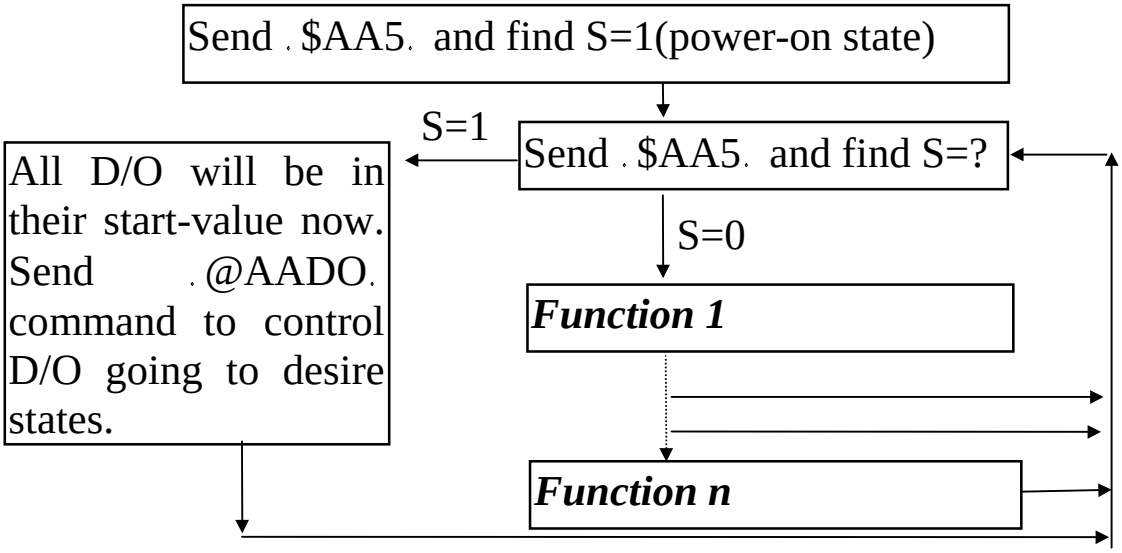
Dual watchdog = host watchdog + module watchdog
The host watchdog is a software watchdog.
The module watchdog is a hardware watchdog.

The 7000 series are designed for industry applications, therefore they can work in the harsh envioronment. There are many couple noise or energy transient in such environment. The modules may be down if these noise is really too large. The module watchdog can reset the module if it is down for too large signal. Sometimes even the host-PC may be down for hardware or software reasons. The host watchdog can monitor the status of host-PC. If the host-PC is down, all the output of 7000 modules will go to their predefined safe states for safety protection.

If the RS-485 network is open**,** all the host command can not send to remote modules. This is very dangerous in real world application. The 7000 output module will force their output going to their predefined safe state for safety consideration if the host watchdog is enable. This dual watchdog feature will increase the system reliability very much.

The .$AA5. command is designed to detect the condition of module watchdog failure. **If the module is down, the module watchdog circuit will reset this module. After this reset the output state of module will go to their start-value. The start-value may be different from those output-value before module reset. Therefore the user need to send output command to module for keeping the same output state before and after the reset of the module watchdog.**

The flow chart of module watchdog failure detection is given as following.



3.3 D/O Operation Principle

(1) Refer to Sec. 2.23 if alarm is enable.

(2) The D/O output of 7011/D, 7012/D and 7014D modules will go to their start-value after first power on.

(3) The D/O output will be changed to the desire state if the . @AADO. command is received. Then all these D/O will keep in the same states until next . @AADO. command.

(4) If the module is reset by the hardware watchdog, all D/O will go to their power-on start-value immediately. The host computer can use . $AA5. command to detect this condition. If the host computer send out . @AADO. to those modules now, those modules will change their D/O without any warning information. Therefore it is recommended to use . $AA5. command to detect the 7011/D, 7012/D and 7014D status.

(5) If the host watchdog is active, all the D/O will go to their safe-value immediately and the module status is set to 04. If the host computer send out . @AADO. to those modules now, those modules will ignore this command and return . !. as warning information. The host can use . ~AA1. command to clear the module status to 0, then the 7011/D, 7012/D and 7014D will accept the . @AADO. again.

3.4 Analog Data Format

The 7000 can be configured to one of the following data formats: (1) Engineering unit (2) Percent of FSR (3)Two.s complement hexadecimal, Assume the range is ±5V, the data format are given as following:

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

The above table is validate for 7011/D, 7012/D and 7014D. If the 7011 is configured to type-00, 01, 02, 03, 04, 05, 06 and 07, the data format is the same as mention above. The data format of thermocouple is given as following:

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 to select . engineering unit. if the 7011 is configured as thermocouple type.

3.5 7000 Module Status

(1) module watchdog reset
- all D/O go to their start-value
- **module status no change**
- accept host D/O command to change D/O state without warning

(2) host watchdog failure
- all D/O go to their save-value
- **module status=04 → host watchdog fails**
- **ignore** all host D/O command until module status is clear to 0 by ~AA1 command.

3.6 Temperature Measurement

The 7011/Dand 7018 can be configured as thermocouple input. The steps to measure thermocouple is given as following:

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

The steps to measure CJC is given as following:

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 $AA9 command.
- The CJC-value can be read back by $AA3 command.

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

1. Thermocouple error → small
2. A/D converter error → small
3. Table lookup error → small
4. CJC-error → **may be big**

Temperature error = (1)+(2)+(3)+(4)

The operation steps of CJC offset calibration are given as following:

1. Place a silver temperature sensor just beside the 7011/D, 7018 CJC sensor. Power on and warm-up about 30 minutes in real place. This step is used to find the circumstance temperature. The silver sensor is used to calibrate the CJC sensor.
2. Use $AA9+0000 to set CJC offset=0
3. Use $AA3 to read out CJC value, T1
4. Read out silver sensor temperature, T2
5. CJC offset=T2-T1
6. Use $AA9+???? to set CJC offset
7. Use $AA3 to read out CJC value, T1
8. Repeat step 2 to step 7 until T1=T2

3.7 Command Response Time

The command response time of 7000 modules is given as following:

1. Host send command
2. 7000 module receive this command
3. **7000 wait a character time**
4. 7000 get the current value and send this value back to host.

Assume the communication condition is given as following:

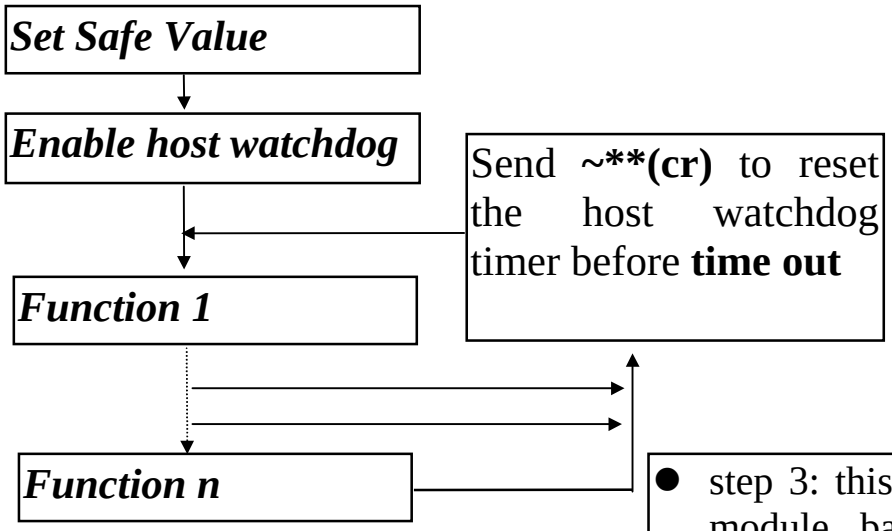
- baud rate=115.2K
- command= **#01(cr)** → 4 character
- wait 1 character
- response= >**HHHH(cr)** → 6 character
- total character= 4+1+6=11 character
- 1 character= 10 bits → 115.2K/10=11.52K
- 11 character → 11.52K/11=1.0K max.→ 1000 command/ response per second max.

The length of the command/response is different for different command. There will be some extra computation & control time in real world application. Therefore the user can not reach this max. peformance. The 7000 A/D modulues have very heavy computation load. Therefore they can not reach this max. performance. **There are many demo program given in NAP7000S and NAP7000P to test the real performance.** The real performance of 7000 analog modules are given as following: **(Pentium-120, Win95, VC++ and 115.2K)**

Module	**Speed**	**Test Condition**
7011/11D/12/12D/13/ 13D/14D	**256 samples per 0.6 second**	**7000_AnalogIn (demo20)**
7017	**1700 samples per second**	**7000_7017In8 (demo27)**
7018	**1000 samples per second**	**7000_AnalogIn8 (demo22)**

3.8 Host WatchDog Applications Notes

The host watchdog is designed to monitor the host computer. If the host computer fails, the output of the 7000 modules will automatically go to their safe states to avoid unpredictable damage. The flow chart for the host computer is given as following.

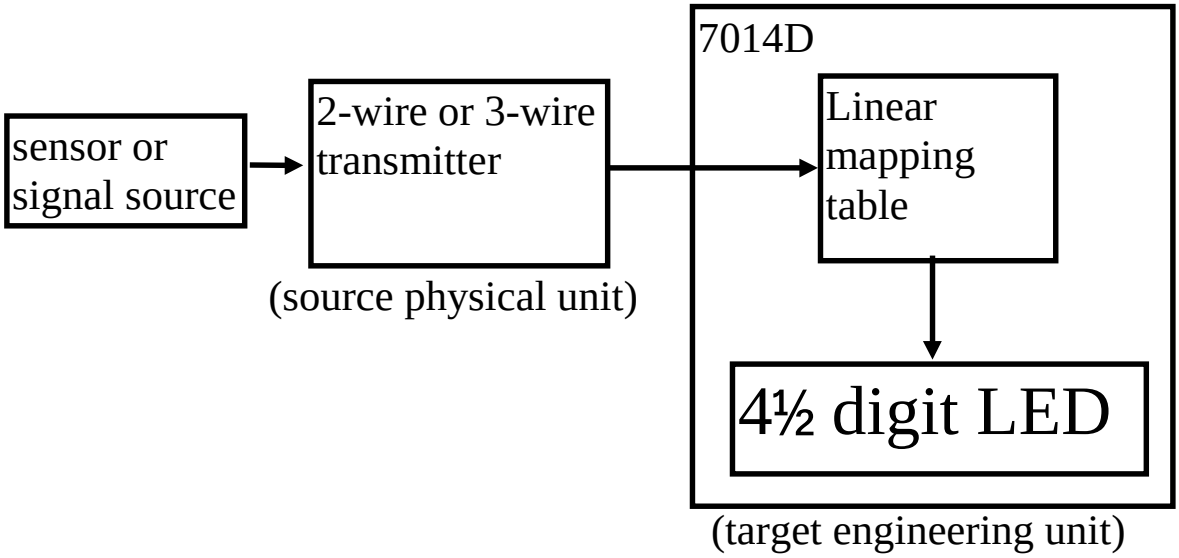


1. Refer to Sec. 1.5 for wire connection. Power on and run **test.exe**
2. press **2**
3. press **$012Enter]→** Receive=!01080600
4. press **2**
5. press **~0150000[Enter]→** Receive=!01
6. press **2**
7. press **@01DO03[Enter]→** Receive=!01
8. press **2**
9. press **~01311E[Enter]→** Receive=!01
10. wait 3 second, you will find that the led of I-7012 is flashed and all D/O go to their safe value → all D/O are OFF.
11. press **2**
12. press **~011[Enter]** → Receive=!01

- step 3: this is a 7012 module, baud rate = 9600.
- step 5: set power-on value and safe value
- step 7: set all D/O on
- step 9: enable the host watchdog and timer=1EH*0.1s=3 sec.
- step 10: this action simulates that the host computer is fail and the host watchdog is active. All D/O output go to their safe states now.
- step 12: clear the module status to 00

3.9 Linear Mapping of 7014D

The 7014D can convert any linear input signal to engineering unit by linear mapping function. The block diagram of linear mapping is given as following:



Example1:	sensor=PT100 source physical unit = 4mA to 20mA target engineering unit = 0°C to 100°C

In this example, the sensor is a PT-100 which can measure temperature from 0°C to 100°C. The transmitter will convert the PT-100 analog signal to [4mA, 20mA] output. If the output is 4mA, the temperature is 0°C. If the output is 20mA, the temperature is 100°C.

Because the transmitter output is 4mA to 20 mA, the user should select type-0D to measure this analog signal. If the linear mapping function is disable, the user will get the measurement result from 4mA to 20mA. Therefore the user has to write another program to convert from [4mA, 20mA] to [0°C, 100°C]. If the linear mapping function is enable, the user can get the measurement result from 0°C to 100°C directly.

Refer to Sec. 1.6.4 for detail operation steps.

Example2:	sensor=PT100 source physical unit = 4mA to 20mA target engineering unit = 0°C to 35°C

In this example, the sensor is a PT-100 which can measure temperature from 0°C to 35°C. The transmitter will convert the PT-100 analog signal to [4mA, 20mA] output. If the output is 4mA, the temperature is 0°C. If the output is 20mA, the temperature is 35°C. This example is very similar to example1 except that the PT-100 is designed to measure temperature from 0°C to 35°C. The 2-wire transmitter is the same as example1.

The detail operation steps are given as following:

1. Refer to Sec. 1.5.6 for 2-wire transmitter wire connection.
2. Power on and run **test.exe**
3. press **2**
4. press **%01010D0600[Enter]** → Receive=!01
5. press **2**
6. press **$016+04.000+20.000**→ Receive=!01
7. press **2**
8. press **$017+00.000+35.000**→ Receive=!01
9. press **2**
10. press **$013** → Receive=!01+04.000+20.000
11. press **2**
12. press **$015** → Receive=!01+00.000+35.000
13. press **2**
14. press **$01A1** → Receive=!01
15. press **2**
16. press **$01B** → Receive=!011
17. press **2**
18. press **#01[Enter]** → Receive=>+???.??

- refer to Sec. 1.6.4 for more information

3.10 7000 Analog Module Selection