MOSRAC® Mosrac motor

PT Series Magnetic Encoder Data Sheet

PT Series Magnetic Encoder

PT magnetic encoder is an ultra-compact product that uses Mosrac motor pattern magnetic technology to achieve high-precision measurement in an extremely limited volume, achieving photoelectric-like resolution and accuracy, and has strong resistance to environmental interference.

The encoder is driven by magneto-electric technology and has unique interference shielding technology. The encoder has multiple high-precision Hall sensors inside to measure the magnetic field changes of the rotor magnetic ring, and is formed with precision calibration technology provided by Mosrac Motor. Each product has unique magnetic field calibration data at the factory, providing the best measurement accuracy.

The unique tolerances fit installation method with rotor and stator simplifies the installation for users, but also guarantees the measurement accuracy.

The separated magnetoelectric solution can ensure that the encoder can avoid the interference of external environmental factors such asvibration, dust, oil pollution, even when it is running at ultra-high speed, without affecting the accuracy and service life.

Ultra-thin body with hollow shaft could easily suit with any application.

- 24-bit absolute output
- ±0.01° accuracy
- Ultra-compact (radial single side 5mm)
- Models for every 5mm on ID
- Stator and rotor tolerance fit installation
- Magnetic Interference Shielding Technology
- Hollow structure doesn't limit installation position
- Max speed 8,000rpm
- Unique data calibration
- Multi-lap battery mode
- Various output interfaces
- Resistance to various environmental disturbances

Models

PT–20–30–A–24–S–N–A–M

Rotor ID
Refer to Size
XX - Thread type
XXB - Bonding type

Stator OD
Refer to Size
XX - Thread type
XXB - Bonding type

Output type
A- absolute

Output parameter
24 – 24-bit single turn

24M - 24-bit single turn + normal multi turn	Multi-turn clear after power failure Multi-turn only valid when power on
24BM - 24-bit single turn + battery multi turn	After power failure, it can still maintain multi-turn measurement by using battery power
24FM - 24-bit single turn + flash multi turn	Only ±90° rotation is allowed after power failure No battery required

Output interface
S – SSI (1)
B–BISS-C
R–RS485
A – RS422
T – T485 (compatible with 17-bit Tamagawa) (2)
T – T485 (compatible with 23-bit Tamagawa) (2)
D – BUS (high-speed bus)
P – PERIOD (periodic sending)

Input voltage
N – 5V

Operating temperature
A – -40 ~ 85 °C
B – -40 ~ 105 °C
C – -40 ~ 125 °C
（compact models supports only ）

Addition
M- no addition Baud rate - when the interface is R , A

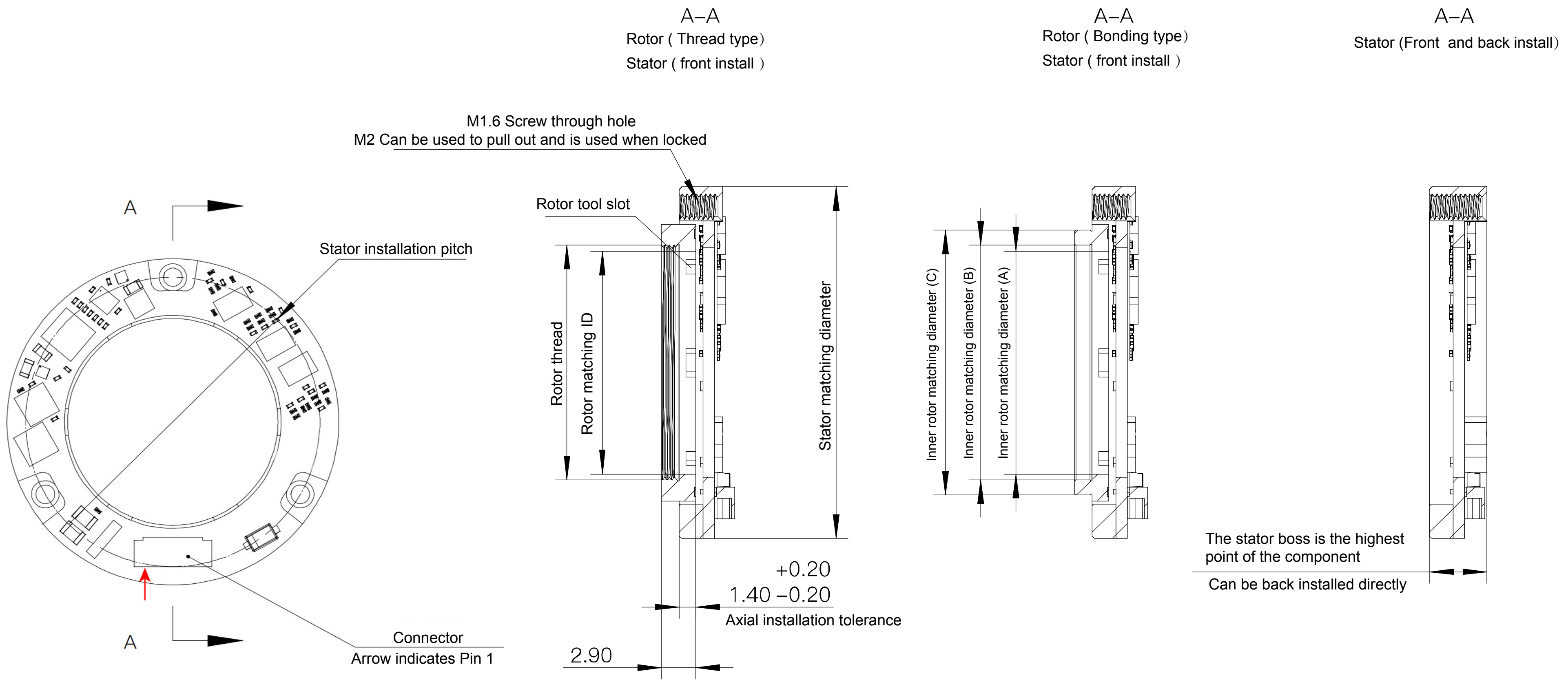
(1) The SSI protocol does not have CRC check. It is recommended to use BISS-C with the same hardware to improve reliability.

(2) When the Tamagawa protocol selects 17 bits, the output parameters are 17M-T, 17BM-T, and 17FM-T. When 23 bits are selected, the output parameters are 23M-T, 23BM-T, and 23FM-T.

Size

Series	Types	Rotor style	Stator style	Inner rotor Thread	Rotor Match OD (H7)			Stator Match OD (h7)	Stator installation pitch circle	The stator can be back installed directly(2)	Connector Models	Overall thickness
					A	B	C					
10	PT-10-20	Thread type	front install	M10x0.4mm	9			20	16.6	–	SM06B-XSRS-ETB	6.7
	PT-10B-20	bonding type	front install	–	9	10	12.6	20	16.6	–	SM06B-XSRS-ETB	
	PT-10-20H	Thread type	Front and back install	M10x0.4mm	9			20	16.6	✓	SM06B-XSRS-ETB	
	PT-10B-20H	bonding type	Front and back install	–	9	10	12.6	20	16.6	✓	SM06B-XSRS-ETB	
15	PT-13-25	Thread type	front install	M13x0.4mm	12			25	21.6	–	SM06B-XSRS-ETB	
	PT-13-25H	Thread type	Front and back install	M13x0.4mm	12			25	21.6	✓	SM06B-XSRS-ETB	
	PT-14-25	Thread type	front install	M14x0.4mm	13			25	21.6	–	SM06B-XSRS-ETB	
	PT-14-25H	Thread type	Front and back install	M14x0.4mm	13			25	21.6	✓	SM06B-XSRS-ETB	
	PT-15-25	Thread type	front install	M15x0.4mm	14			25	21.6	–	SM06B-XSRS-ETB	
	PT-15B-25	bonding type	front install	–	14	15	17.6	25	21.6	–	SM06B-XSRS-ETB	
	PT-15-25H	Thread type	Front and back install	M15x0.4mm	14			25	21.6	✓	SM06B-XSRS-ETB	
	PT-15B-25H	bonding type	Front and back install	–	14	15	17.6	25	21.6	✓	SM06B-XSRS-ETB	
20	PT-20-30	Thread type	front install	M20x0.4mm	19			30	26.6	–	SM08-SURS-TF	
	PT-20B-30	bonding type	front install	–	19	20	22.6	30	26.6	–	SM08-SURS-TF	
	PT-20-30H	Thread type	Front and back install	M20x0.4mm	19			30	26.6	✓	SM08-SURS-TF	
	PT-20B-30H	bonding type	Front and back install	–	19	20	22.6	30	26.6	✓	SM08-SURS-TF	
25	PT-25-35	Thread type	front install	M25x0.4mm	24			35	31.6	–	SM08-SURS-TF	
	PT-25B-35	bonding type	front install	–	24	25	27.6	35	31.6	–	SM08-SURS-TF	
	PT-25-35H	Thread type	Front and back install	M25x0.4mm	24			35	31.6	✓	SM08-SURS-TF	
	PT-25B-35H	bonding type	Front and back install	–	24	25	27.6	35	31.6	✓	SM08-SURS-TF	
30	PT-30-40	Thread type	front install	M30x0.4mm	29			40	36.6	–	SM08-SURS-TF	
	PT-30B-40	bonding type	front install	–	29	30	32.6	40	36.6	–	SM08-SURS-TF	
	PT-30-40H	Thread type	Front and back install	M30x0.4mm	29			40	36.6	✓	SM08-SURS-TF	
	PT-30B-40H	bonding type	Front and back install	–	29	30	32.6	40	36.6	✓	SM08-SURS-TF	
35	PT-35-45	Thread type	front install	M35x0.4mm	34			45	41.6	–	SM08-SURS-TF	
	PT-35B-45	bonding type	front install	–	34	35	37.6	45	41.6	–	SM08-SURS-TF	
	PT-35-45H	Thread type	Front and back install	M35x0.4mm	34			45	41.6	✓	SM08-SURS-TF	
	PT-35B-45H	bonding type	Front and back install	–	34	35	37.6	45	41.6	✓	SM08-SURS-TF	

Drawing



When installing the stator, can use an M1.6 screw to pass through the M2 threaded hole or can use an M2 thread with an M2 screw to lock it.

When installing the rotor use a special tool to tighten it.

Electrical connection

Connector

	Wire-to-board connector
Model	Refer to Size
Type	Wire-to-board
Wire	4PxAWG32 Teflon twisted pair

Pin

Pin	Color	S	B	A	R	T	D	P
		SSI	BISS–C	RS422	RS485	T485	BUS	PERIOD
1	Red	+5V						
2	Black	0V（GND）						
3	Blue	Clock +	MA +	RX +	A	A	A	–
4	Green	Clock -	MA -	RX -	B	B	B	–
5	Yellow	Data +	SLO +	TX +	–	–	–	TX +
6	Orange	Data -	SLO -	TX -	–	–	–	TX -
7	White	Battery +	Battery +	Battery +	Battery +	Battery +	Battery +	Battery +
8	Gray	Battery -	Battery -	Battery -	Battery -	Battery -	Battery -	Battery -

* Battery +/- only appears in the Battery Multiturn (BM) version
* Battery- is connected to GND inside the encoder
* Shield layer is recommended to be grounded at the drive side

Parameters

System

Installation method	Axial hollow
Accuracy	±0.01°

Electrical

Power supply	4.5 ~ 5.5 V
Battery	2.7 ~ 3.6 V
Start-up time	15 ms
Connection method	Wire-to-board connector
Current	≈ 100 mA
Low power consumption current	≈6 μA (battery voltage 3.6V, in low power static detection mode)
ESD resistance	HBM, max. ±2 kV CDM, max. ±1 kV

Mechanical

rotor bracket	Stainless steel
Stator bracket	Aluminum alloy

Environmental

Operating temperature	-40 ~ 85 ℃ / -40 ~ 105 ℃ / -40 ~ 125 ℃

Parameter Detailed Explanation

Max Speed

With a non-contact structure, there is no friction between the rotor and the stator, and the max speed can be extremely high, with a higher speed application environment.

Environmental interference

The measurement principle of the magnetic encoder can make it have a certain resistance to vibration and will not cause fatal damage to the encoder body. It has strong anti-interference ability for non-magnetic objects such as oil, dust, etc., and will not affect the measurement.

External magnetic field interference

PT encoder has unique anti-electromagnetic interference technology. Electromagnetic shielding technology can shield electromagnetic interference transmitted from the environment (such as motors), and the magnetic field strength of the magnetic ring is generally much higher than the external magnetic field interference. The combination of the two can stably resist external interference.

Accuracy

During the processing, the magnetization process of the magnetic ring has the problem of low consistency. However, each encoder of Mosrac Motor is calibrated at the factory, and it has the unique data information of the corresponding magnetic ring in the combination.

Absolute value series

Output format

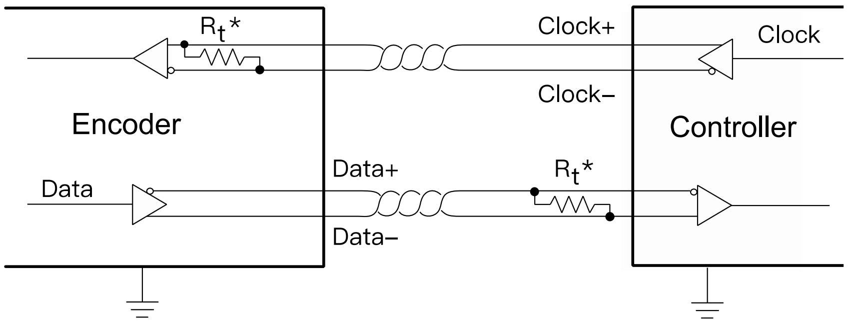
Single turn	
Output angle data	
Multi-turn	
Output turns and angle data	The number of tums cannot be kept after power off, and it will reset to zero after repower on
Battery multi-turn	
Output turns and angle data	After the power supply is off, the battery is take over, and encoder enters to the low power mode. The rotation of the motor can still be measured, and the number of turns could be read after the power is recovered

Absolute value parameter

Single turn bit number	24 bit
Max speed	8,000 rpm
Update frequency	50 kHz
Repeatability	0~±1 bit (varies according to the number of bit)
Output interface	SSI、BISS-C、RS485、RS422、T485、BUS、PERIOD

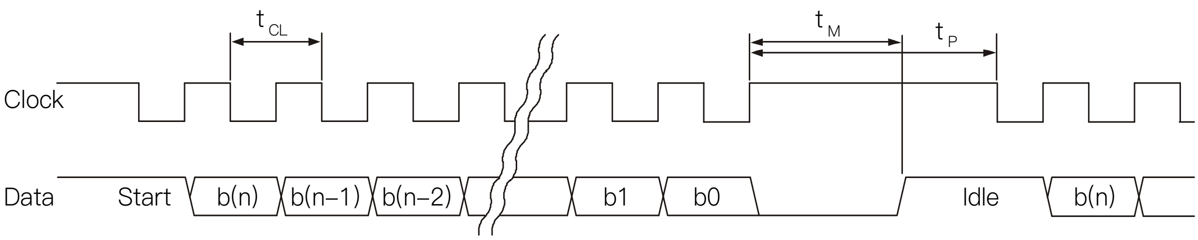
SSI protocol interface

Electrical connection diagram:



It is a four-wire interface, include the differential lines of Clock and Data. And the terminal resistors of the Clock lines have been integrated into the encoder. The users only need to configure terminal resistors or choose other impedance matching schemes for Data lines.

Sequence diagram:



In this protocol, when the first falling edge of the Clock arrives, the system latches current data. The data is written to Data line on the rising edge of each Clock from the MSB, and on the controller side, the data on Data line is read from the falling edge of Clock, and so on until the LSB is read by the controller.

After the transfer is complete, when the tm transfer time ends , the Data line goes high and the Clock signal must remain high until the next read is allowed, i.e. after tp time. tcl, must be less than tm, and the read can be terminated by making the time exceed tm while any read operation is in progress.

Timing Parameters:

Parameters	Symbol	Min value	Typical value	Max value
Clock period	t_{CL}	400 ns		14 μs
Clock frequency	t_{CL}	110 kHz		1.5 MHz(1)
Transmit timeout	t_{M}		10 μs	
Pause duration	t_{P}	20 μs		

(1) If the Clock can be held at the first low level for 500ns, the subsequent clock frequency can be up to 10MHz

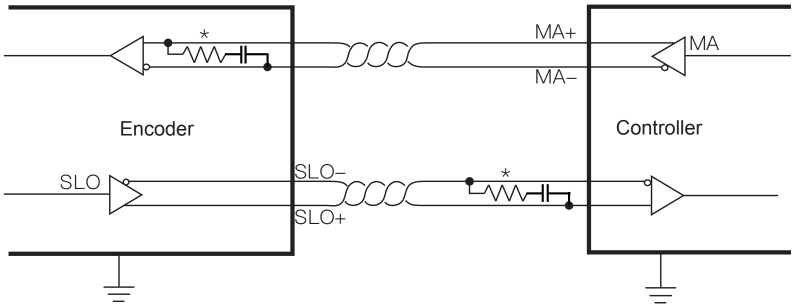
Data format:

Bits	b(23 + X) : b(8 + X)	b(7 + X) : b8	b7	b6	b5 : b0
Length	16 bits	X bits	1bit	1bit	6 bits
Data	Multiturn count (1)	Singleturn angle	Error bit	Warning bit	Status bit

(1) Multiturn count is only available on multiturn and battery multiturn versions

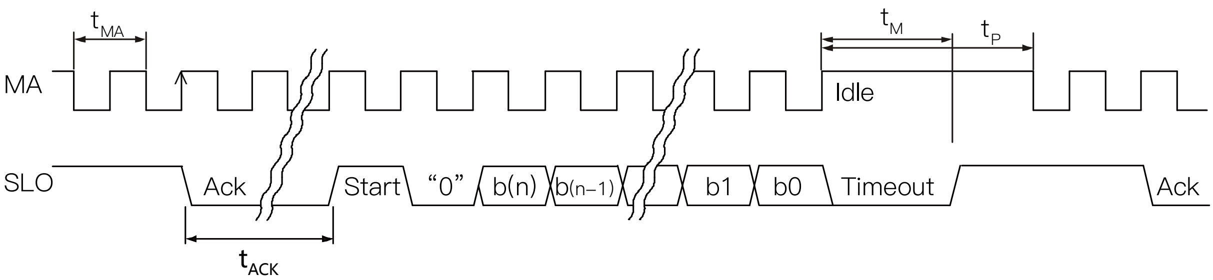
BISS-C protocol interface

BiSS-C Electrical connection diagram:



It is a four-wire interface, include the differential lines of MA and SLO. And the terminal resistors of the MA lines have been integrated into the encoder. The users only need to configure terminal resistors or choose other impedance matching schemes for SLO lines.

Timing diagram :



The protocol uses MA as the synchronization clock, and the MA line is high when idle. When the first falling edge of the synchronous clock arrives, the system latches the current data.

The communication will start on the first faling edge, encoder will configure SLO low on the second MA rising edge. After the "0", the MSB will be written to the SLO line on each rising edge of MA, and on controller side, the data on Data line is read on the falling edge of Clock, and so on until the LSB is read by the controller.

Timing parameters:

Parameters	Symbol	Min value	Type value	Max value
Clock period	tMA	400 ns		14 μs
Clock frequency	f	120 kHz		2.5 MHz(1)
ACK length	tACK		5 bits	
Transmit timeout	tM		10 μs	
Pause duration	tP	20 μs		

(1) Up to 10 MHz if the user can compensate for the delay between differential conversions with phase compensation techniques

After the transfer is complete, when the tm transfer time is over, the SLO line goes high and the MA signal must remain high until the next read is allowed, i.e. after tp time. tcl, must be less than tm, and the read can be terminated by making the time exceed tm while any read operation is in progress.

Data format:

Bits	b(24 + X) : b(9 + X)	b(8 + X) : b8	b7	b6	b5 : b0
Length	16 bits	X bits	1bit	1bit	6 bits
Data	Multiturn count(1)	Signleturn angle	Error bit(2)	Warning bit(3)	CRC(4)

(1) Multiturn count is only available on multiturn and battery multiturn versions

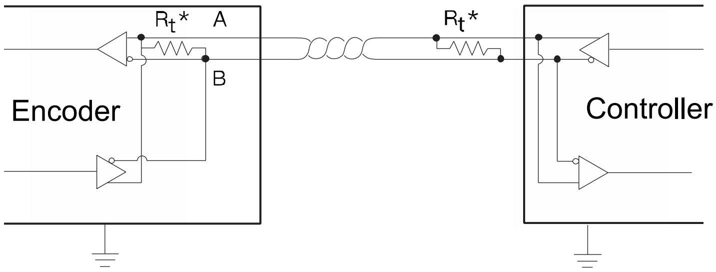
(2) The error bit is valid at low level. This bit may be low level only in the battery multi-turn version. When it is 1, it means that the battery-related status bits are normal and the multi-turn data is reliable; when it is 0, it means that the battery low voltage and battery interruption errors in the status bits are triggered. For solutions, please refer to the status bit section.

(3) The warning bit is low level effective. When it is 1, it means no error or warning is triggered; when it is 0, it means at least one error or warning is triggered.

(4) The CRC polynomial is x6+x1+1 (i.e. 0x43). According to the BISS-c protocol requirements, the calculated CRC will be inverted before being sent.

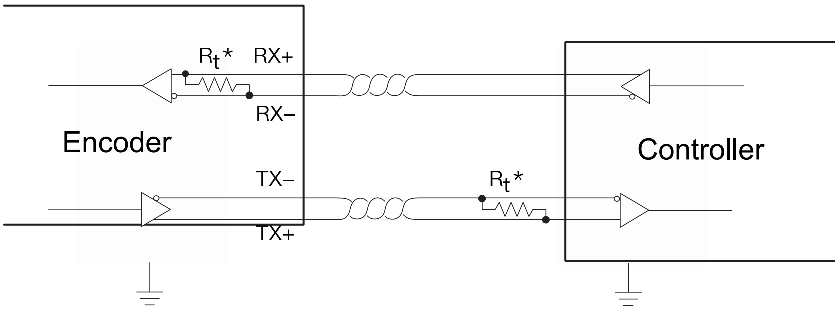
RS485/RS422 protocol interface

RS485 Electrical connection diagram:



The interface is a two-wire system, which is mainly differential A and B phases. The terminals of the two lines need to be connected to terminal resistors in parallel. The terminal resistors at the encoder end have been integrated into the encoder. Users need to connect terminal resistors or other impedance matching solutions on the controller side A and B.

RS422 Electrical connection diagram:



It is a four-wire interface, include the differential lines of TX and RX. And the terminal resistors of the encoder RX lines have been integrated into the encoder. The users only need to configure terminal resistors or choose other impedance matching schemes for TX lines.

The underlying protocol of both RS485 and RS422 is UART. Since this protocol has no clock line, the encoder and controller must have the same transmission freguency and data format in order to ensure proper signal transmission.

Protocol Configuration:

Length	Parity check	Stop bit	Stream control	Byte order
8 bit	-	1	-	LSB first

Baud rate supported (if not marked in additional, the default and recommended B)

Code	A	B
Baud rate (Mbps)	0.1152	2.5

Command and data

Command	Introduction		Output parameter	N	Return data (N bytes)							
					B0	B1	B2	B3	B4	B5	B6	B7
0x30	Set	Zero position[1]	–	2	C	CRC						
0x31	Get	Position	24	4	A2	A1	A0	CRC				
			24M 24BM 24FM	6	M1	M0	A2	A1	A0	CRC		
0x64	Get	Position+ status	24	5	A2	A1	A0	S	CRC			
			24M 24BM 24FM	7	M1	M0	A2	A1	A0	S	CRC	
0x74	Get	Position+ temperature[2]	24	6	A2	A1	A0	T1	T0	CRC		
			24M 24BM 24FM	8	M1	M0	A2	A1	A0	T1	T0	CRC

(1) To set the zero position, need to send the command "0x31" at intervals and send the command "0x30" 10 times in total to successfully set it. When the command "0x30" returns a count value of 10, the zero position is set.The temperature information is the junction temperature of the chip

(2) The temperature information is the junction temperature of the chip

In the above table, the letters correspond to the data:

M	A	C	S	T	CRC
Multiturn data	Singleturn angle	Clear reading value	Status	Temperature[2]	CRC check[1]

For a detailed description of the status bits, see the Status Bits section below.

(1) CRC Byte (CRC polynomial is $x^8+x^7+x^4+x^2+x^1+1$, Calculation method see Appendix CRC-8 table($x^8+x^7+x^4+x^2+x^1+1$))

(2) Temperature calculation method is: : int16_t temp = T0 | T1 << 8; float tempFloat = temp / 10.f; Unit °C

Example: (24BM) :

If you send 0x32, you get uint8_t Buffer[7]

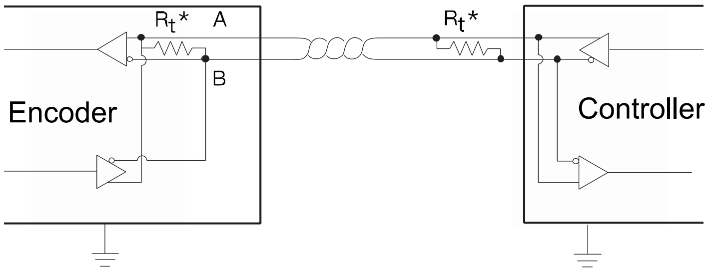
Then when using:

```
uint16_t multi = Buffer[0] << 8 | Buffer[1];
uint32_t angle = Buffer[2] << 16 | Buffer[3] << 8 | Buffer[4];
float angleFloat = angle / (float)(1 << 24) * 360;
```

T485 protocol interface

This interfaceis compatible with the tamagawa protocol

RS485 Electrical connection diagram:



It is a diferential two-wire interface, both wires need to be terminated with parallel termination resistors.The terminal resistors have been integrated into the encoder. The users only need to configure terminal resistors or choose other impedance matching schemes for the differential wires of the controller side.

T485 is based on RS485 and has a certain communication protocol. The interface receives a 1Bvte operation requests, returns the corresponding encoder data based on the requested data, and adds CRC-8 to the end of the data for check.

Data length	Parity check	Stop bit	Stream control	Bytes order
8 bit	-	1	-	LSB frist

Protocol configuration

Command :

Bits	b7 ~ b3	b2	b1	b0
Data content	Operation type	0	1	0

Return :

Bytes	B0	B1	B(2 ~ n)	B(n + 1)
Data content	Command (1)	Status	Return data	CRC (2)

(1) The operation request returned is the same as the one sent.

(2) CRC byte(CRC polynomial is x8 + 1, see CRC-8 calculate of Appendix for calculation method).

B1 format

Bits	b7	b6	b5	b4	b3	b2	b1	b0
Data content	0	Transmit error	Encoder error	0	0	0	0	0

B(2 ~ n), operation type and the corresponding data returned（A for angle; M for multiturn; E for error）

Operation type					Introduction	n	Data returned							
b7	b6	b5	b4	b3			B2	B3	B4	B5	B6	B7	B8	B9
0	0	0	0	0	Get angle	4	A0	A1	A2					
1	0	0	0	1	Get multiturn	4	M0	M1	M2					
0	0	0	1	1	Get all data	9	A0	A1	A2	17	M0	M1	M2	E
1	1	0	0	0	Reset angle(3)	4	A0	A1	A2					
0	1	1	0	0	Reset multiturn(4)	4	A0	A1	A2					

(1) An and Mn are left-aligned, i.e., if A is 17 bits of data, the higher 7 bits of A2 are 0

(2) If the operation type does not appear in the table of B (2~n), a communication error is triggered and the return data is the same as the return data of the get angle request.

(3) The reset of angle or multiturn requires 10 consecutive requests for the corresponding operation to take effect

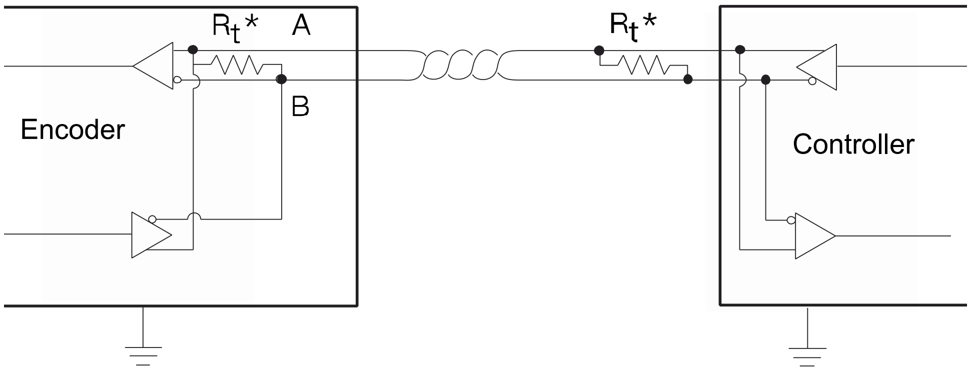
E, error byte(see the Status section later, the message in this byte only indicates an error exception):

b7	b6	b5	b4	b3	b2	b1	b0
Battery lost	Battery low voltage	0	0	0	0	0	0

For LED-related indications, please refer to the Status chapter.

BUS (High speed bus) protocol interface

BUS Electrical connection diagram：



It is a differential two-wire interface, both wires need to be terminated with parallel termination resistors. The terminaresistors have been integrated into the encoder. The users only need to configure terminal resistors or choose otherimpedance matching schemes for the differential wires of the controller side.

The protocol is the same as the 485 protocol level logic the data is based on UART format and the operating frequency is 2.5 Mbps.the interface receives 1Byte operation request returns the correspondi ng encoder data and adds CRC-8 (x8+1) to the end of the data.

Version for select :

Singleturn bits	Multiturn bits	Model no.
XX	-	XX–D
	16	XXM–D

Protocol configuration：

Byte length	Parity check	Stop bit	Stream control	Byte order
8 bit	-	1	-	LSB first

Command:

Data bits	b7	b6 ~ b5	b4 ~ b0
Data content	Odd parity check	Operation type	Address

Return data：

Bytes	B0	B(1 ~ n)	B(n + 1)
Data content	Operation request	Return data：	CRC

B(1 ~ n), operation type and the corresponding data returned（A is angle; M is multiturn; C is set count; S is status）

Operation type		Introduction	Model Versior		n	Return data：					
b6	b5		Singleturn bits	Multiturn bits		B1	B2	B3	B4	B5	B6
0	0	Get data	≤16	-	3	S	A0	A1			
			≤16	16	5	S	A0	A1	M0	M1	
			>16	-	4	S	A0	A1	A2		
			>16	16	6	S	A0	A1	A2	M0	M1
0	1	Set zero position			2	S	C				
1	0	Set address			2	S	C				

Operation type		Introductioı	N	Data returned		
b6	b5			B1	B2	B3
0	0	Get data	3	S	A0	A1
0	1	Set zero position	2	S	C	
1	0	Set address	2	S	C	

For a detailed description of the status bit, see the Status section

later. Note :

1. When the operation type does not appear in the table of B(1~n), such as 0b11, there will be no return response.

2. C is the count data returned when the encoder is set, when it returns 10, it means that the setting will be executed immediately (the process takes about 50ms maximum, during which the encoder will not respond to any command)

3. The default address is 0x1F, i.e. 0b11111

Set zero position：

To ensure that the setting of the zero position is not mistakenly operated, it is necessary to send the command of operation type 00, 01 in succession alternately, a total of ten groups (after each command is sent, the encoder finishing replying must be waited, before sending the next command), in order to set it successfully. The number of times still need to be sent based on the C value returned by the 01 command.

1. When the previous command of 01 is not 00, the sequence start requirement is not met and the C value is 0
2. When the command before previous command of 01 is not 01, the sequence is not satisfied, the C value is 1, and the count starts from there

Set ID :

To ensure that the setting ID is not mistakenly operated, it is necessary to send a certain sequence of address values to make sure that the encoder has already enter the state of ID configuration, and then configure the ID. For example, the address value is sent continuously as shown in the table below, the next address value sent is twice the return value.

Address	X	2	4	6	8	10	12	14	16	Y
C	-	-	-	4	5	6	7	8	9	10

Note :

1. X could be any value, Y is the actual address value want to be set
2. The first three sets of data sent are not returned with any corresponding response, to prevent the bus from being mistakenly triggered in the working state
3. When another command is inserted, the returned C value of the next set command is 1, and the count starts from there
4. When the sent address value does not match the sequence, the returned C value is 1, and the count restarts from 1

Bus devices:

The devices on the bus need got into "sleep" for a certain period of time when the received ID is not their own, no responding to any commands on the bus, so as to prevent trigger the response ripples.

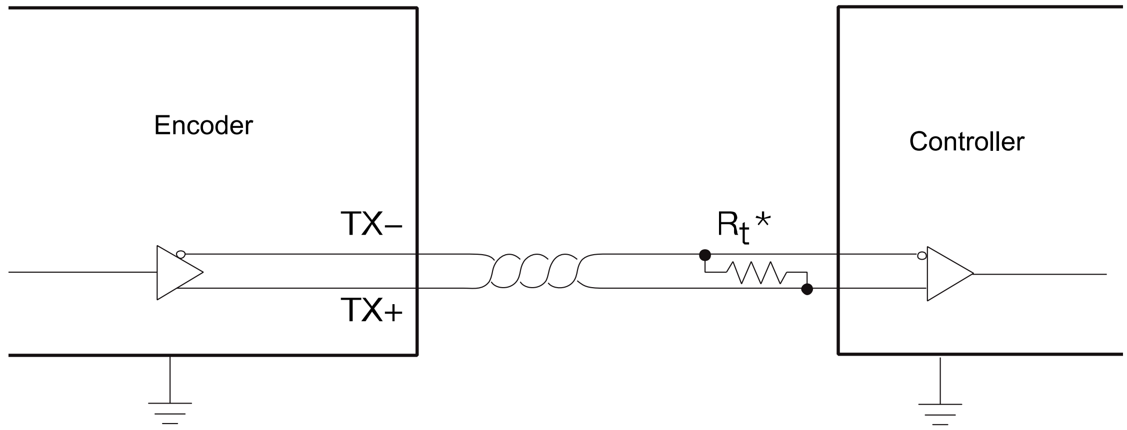
The sleep time is TSUSPEND, and the following table shows the calculation

Singleturn bit	Multiturn bits	Number of bus sleep bytes $B_{SUSPEND}$	Time of bus sleep bytes $T_{SUSPEND}$ (us)
XX	YY	ceil(XX/8) + YY/8 + 4	$B_{SUSPEND}$ * (1 + 8 + 1) / 2.5

Take the 16M1-D model as an example: $T_{suspend}$=(ceil(16 / 8) + 8 / 8 + 4) * (1 + 8 + 1) / 2.5 = 28us

PERIOD cycle sending protocol

PERIOD Electrical connection diagram：



It is a differential two-wire interface both wires need to be terminated with parallel termination resistors. The terminal resistors have been integrated into the encoder. The users only need to configure terminal resistors or choose other impedance matching schemes for the differential wires of the controller side.

This protocol is based on the RS422 protocol the only difference is that it actively sends data out through the TX at 1khz and does not respond to any message the encoder internally triggers the command “d” (0x64) periodically to send the corresponding data, refer to RS485/RS422 protocol interface.

Status:

In SSI/BISS-C/RS485/RS422 protocol, the usage of status bit is consistent, when a warning or error occurs, the warning bit or error bit will be set, and the user can specify the cause of the warning or error by viewing the status bit information.

The location of error/warning in protocols:

	Error bit	Warning bit
SSI	b7	b6
BISS–C	b13	b12
RS485/RS422	b7	b6
BUS	b7	b6
PERIOD	b7	b6

Status bit:

Location	For battery multiturn version only		b3	b2	b1	b0
	b5	b4				
Description	Battery lost	Battery low voltage	Large magnetic field	Weak magnetic field	Temperature out of range	Over speed
LED flashing	√	√	-	-	-	-

In normal condition, the LED status light is green. When the warning bit is 1, the data is still valid, and the LED turns yellow, but some parameters of status bit are close to their limit values, which can be viewed through the status bit. When the error bit is 1, the data is no longer valid, and the LED turns red. And the status bit shows specific error message.

The LED flashes in 1s intervals to alert the user to the occurrence of the corresponding error/warning problem.

Battery-related status bits:

	b5	b4
	Battery lost	Battery low voltage
Warning is 1 1	–	Battery voltage<2.9V
Error bit is 1	The encoder is disconnected during a power failure or the battery is too low, resulting in an interruption of the multiturn count and thus untrustworthy multiturn data, and the multiturn is cleared if this error occur	Battery voltage<2.7V
Solution	Check the battery voltage supply, repower the encoder then this bit will reset	Replace the battery

Appendix

CRC-8 Table($x^8+x^7+x^4+x^2+x^1+1$)

```
//poly = x8+x7+x4+x2+x1+1
uint8_t crcTable [256] = {
        0x00, 0x97, 0xB9, 0x2E, 0xE5, 0x72, 0x5C, 0xCB, 0x5D, 0xCA, 0xE4, 0x73, 0xB8, 0x2F, 0x01, 0x96,0xBA, 0x2D, 0x03, 0x94,
0x5F, 0xC8, 0xE6, 0x71, 0xE7, 0x70, 0x5E, 0xC9, 0x02, 0x95, 0xBB, 0x2C,0xE3, 0x74, 0x5A, 0xCD, 0x06, 0x91, 0xBF, 0x28, 0xBE,
0x29, 0x07, 0x90, 0x5B, 0xCC, 0xE2, 0x75,0x59, 0xCE, 0xE0, 0x77, 0xBC, 0x2B, 0x05, 0x92, 0x04, 0x93, 0xBD, 0x2A, 0xE1, 0x76,
0x58, 0xCF,0x51, 0xC6, 0xE8, 0x7F, 0xB4, 0x23, 0x0D, 0x9A, 0x0C, 0x9B, 0xB5, 0x22, 0xE9, 0x7E, 0x50, 0xC7,0xEB, 0x7C, 0x52,
0xC5, 0x0E, 0x99, 0xB7, 0x20, 0xB6, 0x21, 0x0F, 0x98, 0x53, 0xC4, 0xEA, 0x7D,0xB2, 0x25, 0x0B, 0x9C, 0x57, 0xC0, 0xEE, 0x79,
0xEF, 0x78, 0x56, 0xC1, 0x0A, 0x9D, 0xB3, 0x24,0x08, 0x9F, 0xB1, 0x26, 0xED, 0x7A, 0x54, 0xC3, 0x55, 0xC2, 0xEC, 0x7B, 0xB0,
0x27, 0x09, 0x9E,0xA2, 0x35, 0x1B, 0x8C, 0x47, 0xD0, 0xFE, 0x69, 0xFF, 0x68, 0x46, 0xD1, 0x1A, 0x8D, 0xA3, 0x34,0x18, 0x8F,
0xA1, 0x36, 0xFD, 0x6A, 0x44, 0xD3, 0x45, 0xD2, 0xFC, 0x6B, 0xA0, 0x37, 0x19, 0x8E,0x41, 0xD6, 0xF8, 0x6F, 0xA4, 0x33, 0x1D,
0x8A, 0x1C, 0x8B, 0xA5, 0x32, 0xF9, 0x6E, 0x40, 0xD7,0xFB, 0x6C, 0x42, 0xD5, 0x1E, 0x89, 0xA7, 0x30, 0xA6, 0x31, 0x1F, 0x88,
0x43, 0xD4, 0xFA, 0x6D,0xF3, 0x64, 0x4A, 0xDD, 0x16, 0x81, 0xAF, 0x38, 0xAE, 0x39, 0x17, 0x80, 0x4B, 0xDC, 0xF2, 0x65,0x49,
0xDE, 0xF0, 0x67, 0xAC, 0x3B, 0x15, 0x82, 0x14, 0x83, 0xAD, 0x3A, 0xF1, 0x66, 0x48, 0xDF,0x10, 0x87, 0xA9, 0x3E, 0xF5, 0x62,
0x4C, 0xDB, 0x4D, 0xDA, 0xF4, 0x63, 0xA8, 0x3F, 0x11, 0x86, 0xAA, 0x3D, 0x13, 0x84, 0x4F, 0xD8, 0xF6, 0x61, 0xF7, 0x60, 0x4E,
0xD9, 0x12, 0x85, 0xAB, 0x3C
};

uint8_t calcCRC(uint8_t * buffer, uint8_t length){
        uint8_t temp = *buffer++;
        while(--length){
                temp = *buffer++ ^ crcTable[temp];
        }

        return crcTable[temp];
}
```

CRC-8 Calculate(x^8+1)

```
//poly = x8+1
//The value of table check is the same as the result of polynomial calculation

uint8_t calcCRC(uint8_t * buffer, uint8_t length){
        uint8_t temp = *buffer++;
        while(--length){
                temp = *buffer++ ^ temp;
        }

        return temp;
}
```

CRC-6 Calculate

```c
#define DATA_TOTAL_BIT_LENGTH 47


//poly = x^6+x^1+1
uint8_t tableCRC6[64] = {
0x00, 0x03, 0x06, 0x05, 0x0C, 0x0F, 0x0A, 0x09, 0x18, 0x1B, 0x1E, 0x1D, 0x14, 0x17, 0x12, 0x11, 0x30, 0x33, 0x36, 0x35, 0x3C, 0x3F, 0x3A, 0x39, 0x28, 0x2B, 0x2E, 0x2D, 0x24, 0x27, 0x22, 0x21, 0x23,
0x20, 0x25, 0x26, 0x2F, 0x2C, 0x29, 0x2A, 0x3B, 0x38, 0x3D, 0x3E, 0x37, 0x34, 0x31, 0x32, 0x13, 0x10, 0x15, 0x16, 0x1F, 0x1C, 0x19, 0x1A, 0x0B, 0x08, 0x0D, 0x0E, 0x07, 0x04, 0x01, 0x02
};

uint8_t calcBissCCRC(uint8_t buffer[]){

	#define CRC_BIT_LENGTH 6
	#define DATA_CRC_MASK ((1 << CRC_BIT_LENGTH) - 1)
	#define DATA_WITHOUT_CRC_BIT_LENGTH (DATA_TOTAL_BIT_LENGTH - CRC_BIT_LENGTH)
	#define TOP_BYTE_BITLENGTH (DATA_WITHOUT_CRC_BIT_LENGTH % CRC_BIT_LENGTH)
	#if TOP_BYTE_BITLENGTH == 0
		#undef TOP_BYTE_BITLENGTH
		#define TOP_BYTE_BITLENGTH CRC_BIT_LENGTH
	#endif

	uint32_t firstWord = __REV(*(uint32_t *) buffer);
	#if DATA_WITHOUT_CRC_BIT_LENGTH > 32
	uint32_t secondWord = __REV(*(uint32_t *) (buffer + 4));
	#endif

	uint8_t crc = tableCRC6[firstWord >> (32 - TOP_BYTE_BITLENGTH)];

	#undef CURRENT_CRC_BIT_LENGTH
	#define CURRENT_CRC_BIT_LENGTH (TOP_BYTE_BITLENGTH + CRC_BIT_LENGTH * 1)
	#if DATA_WITHOUT_CRC_BIT_LENGTH - CURRENT_CRC_BIT_LENGTH >= 0
		crc = tableCRC6[crc ^ (firstWord >> (32 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
	#endif

	#undef CURRENT_CRC_BIT_LENGTH
	#define CURRENT_CRC_BIT_LENGTH (TOP_BYTE_BITLENGTH + CRC_BIT_LENGTH * 2)
	#if DATA_WITHOUT_CRC_BIT_LENGTH - CURRENT_CRC_BIT_LENGTH >= 0
		crc = tableCRC6[crc ^ (firstWord >> (32 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
	#endif

	#undef CURRENT_CRC_BIT_LENGTH
	#define CURRENT_CRC_BIT_LENGTH (TOP_BYTE_BITLENGTH + CRC_BIT_LENGTH * 3)
	#if DATA_WITHOUT_CRC_BIT_LENGTH - CURRENT_CRC_BIT_LENGTH >= 0
		crc = tableCRC6[crc ^ (firstWord >> (32 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
	#endif

	#undef CURRENT_CRC_BIT_LENGTH
	#define CURRENT_CRC_BIT_LENGTH (TOP_BYTE_BITLENGTH + CRC_BIT_LENGTH * 4)
	#if DATA_WITHOUT_CRC_BIT_LENGTH - CURRENT_CRC_BIT_LENGTH >= 0
		crc = tableCRC6[crc ^ (firstWord >> (32 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
	#endif

	#undef CURRENT_CRC_BIT_LENGTH
	#define CURRENT_CRC_BIT_LENGTH (TOP_BYTE_BITLENGTH + CRC_BIT_LENGTH * 5)
	#if DATA_WITHOUT_CRC_BIT_LENGTH - CURRENT_CRC_BIT_LENGTH >= 0
		#if 32 - CURRENT_CRC_BIT_LENGTH >= 0
			crc = tableCRC6[crc ^ (firstWord >> (32 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
		#elif 32 - CURRENT_CRC_BIT_LENGTH > -CRC_BIT_LENGTH
			crc = tableCRC6[crc ^ (((firstWord << -(32 - CURRENT_CRC_BIT_LENGTH)) & DATA_CRC_MASK) | (secondWord >> (64 - CURRENT_CRC_BIT_LENGTH)))];
		#else
			crc = tableCRC6[crc ^ (secondWord >> (64 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
		#endif
	#endif

	#undef CURRENT_CRC_BIT_LENGTH
	#define CURRENT_CRC_BIT_LENGTH (TOP_BYTE_BITLENGTH + CRC_BIT_LENGTH * 6)
	#if DATA_WITHOUT_CRC_BIT_LENGTH - CURRENT_CRC_BIT_LENGTH >= 0
		crc = tableCRC6[crc ^ (secondWord >> (64 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
	#endif

	#undef CURRENT_CRC_BIT_LENGTH
	#define CURRENT_CRC_BIT_LENGTH (TOP_BYTE_BITLENGTH + CRC_BIT_LENGTH * 7)
	#if DATA_WITHOUT_CRC_BIT_LENGTH - CURRENT_CRC_BIT_LENGTH >= 0
		crc = tableCRC6[crc ^ (secondWord >> (64 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
	#endif

	#undef CURRENT_CRC_BIT_LENGTH
	#define CURRENT_CRC_BIT_LENGTH (TOP_BYTE_BITLENGTH + CRC_BIT_LENGTH * 8)
	#if DATA_WITHOUT_CRC_BIT_LENGTH - CURRENT_CRC_BIT_LENGTH >= 0
		crc = tableCRC6[crc ^ (secondWord >> (64 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
	#endif

	#undef CURRENT_CRC_BIT_LENGTH
	#define CURRENT_CRC_BIT_LENGTH (TOP_BYTE_BITLENGTH + CRC_BIT_LENGTH * 9)
	#if DATA_WITHOUT_CRC_BIT_LENGTH - CURRENT_CRC_BIT_LENGTH >= 0
		crc = tableCRC6[crc ^ (secondWord >> (64 - CURRENT_CRC_BIT_LENGTH) & DATA_CRC_MASK)];
	#endif

	#if 32 - DATA_TOTAL_BIT_LENGTH >= 0
		crc = tableCRC6[crc ^ DATA_CRC_MASK ^ (firstWord >> (32 - DATA_TOTAL_BIT_LENGTH) & DATA_CRC_MASK)];
	#elif 32 - CURRENT_CRC_BIT_LENGTH > -CRC_BIT_LENGTH
		crc = tableCRC6[crc ^ DATA_CRC_MASK ^ (((firstWord << -(32 - DATA_TOTAL_BIT_LENGTH)) & DATA_CRC_MASK) | (secondWord >> (64 - DATA_TOTAL_BIT_LENGTH)))];
	#else
		crc = tableCRC6[crc ^ DATA_CRC_MASK ^ (secondWord >> (64 - DATA_TOTAL_BIT_LENGTH) & DATA_CRC_MASK)];
	#endif

	return crc;
}
```

Instruction:

This program can be applied to ARM series MCU, and can generate the fastest CRC-6 check through the compiler, just modify DATA_TOTAL_BIT_LENGTH to the value of the corresponding model.

For example: 17M model to 47，16 model to 30

Caution:

When called this function, a 32-bit read command is used for the buffer, requiring the buffer to be 4-byte aligned (some core versions don’t support unaligned data read; Even with support for unaligned reads, the kernel consumes more concatenation time).

For the reception of BISS-C, the first byte received will be a placeholder byte with ACK, and the position data starts from the second byte, to read data and calculate CRC quickly, it is necessary to calculate CRC from the address where the position data starts, and require the address to be 4-byte alignment data.

For example:

```
struct{uint8_t notUsedForAlignment[3];          //Address alignment only
       uint8_t placeholder;                     //BISS-C The first placeholder byte of
uint8_t buffer[8] __attribute__ ((aligned(4))); } receiveBuffer;   BISS-C is 0x82 //Buffer of 4 bytes data
                                                align, for fast CRC

//Configure SPI and DMA
///Use &receiveBuffer.placeholder as the receiving address

//……

//CRC calculate
//Calculated using the receiveBuffer.buffer which is already 4-byte aligned uint8_t
crc = calcBissCCRC(receiveBuffer.buffer);

//The CRC result is equal to 0, indicating that the verification is passed
if ( crc != 0 ){
       //crc  validation failed
}
```

Version history

Time	Version	Revised content
2023/09/28	V0.1	Initialized version
2024/01/05	V0.2	Added more dimensions
2023/05/01	V0.3	Added standard drawings Added wire colors

ShenZhen Mosrac Motor Co., Ltd

E-mail: sales12@mosrac.com
Tel: +8618100274370
Fax: 0755-23091465
Company address: Building 1, COFCO (Fu'an) Robot Technology Park, No. 90 Dayang Road, Bao'an District, Shenzhen,518103, China
Factory Address: No. 36, Xingda Road, Yanluo Street, Bao'an District, Shenzhen
Website: http://www.mosrac.com