KingKong.tech

MPT Magnetic Encoder Datasheet v0.6



MPT Magnetic Encoder



Compact MPT magnetic encoder featuring an ultra-small form factor, based on KingKong Technology's proprietary pattern magnetic technology. It delivers high-precision measurement within extremely limited space, achieving optical-level resolution and accuracy, while maintaining strong resistance to environmental interference.

The MPT is based on magnetoelectric sensing technology and features proprietary interference shielding. Multiple high-precision magnetic field sensors are integrated inside the encoder to detect variations in the magnetic field of the rotor magnet ring. Combined with KingKong Technology's precision calibration process, each unit is individually calibrated at the factory with unique magnetic field data to ensure optimal measurement accuracy.

The unique rotor-stator tolerance-matching mounting method simplifies installation while ensuring measurement precision.

The separated magnetoelectric design allows the encoder to effectively resist external disturbances such as vibration, dust, and oil contamination. Even under ultra-high-speed operation, accuracy and service life remain unaffected.

The ultra-thin design with a hollow shaft makes it easy to integrate into a wide range of applications.

- 24-bit absolute output
- ±0.01° accuracy
- Ultra-compact (Ring width 5mm)
- Models for every 5mm on diameter
- Stator and rotor tolerance fit mounting
- Magnetic interference shielding
- Hollow structure
- Permissible speed 8,000 rpm
- Battery mode speed: 3,000 rpm
- Multi-turn with battery mode or flash
- Various output interfaces
- Resistance to various environmental disturbances

Models

MPT-20-30-A-24-S-N-A-M

Rotor inner diameter

Refer to Size

xx — Thread type

xxB — Bonding type

Stator outer diameter

Refer to Size

xx — Thread type

xxB — Bonding type

Output type

A — absolute

Output parameter

24 — 24 bits single-turn

Multi-turn count resets after power-off and is only retained during power-on. — 24M — 24 bits + Normal multi-turn

After power-off, battery backup maintains the multi-turn data — 24BM — 24 bits + Battery multi-turn

Multi-turn data is stored in flash memory, allowing up to ±90° rotation after power-off. — 24FM — 24 bits + Flash multi-turn

Input voltage

N — 5V

Output interface

S — SSI(1)

B — BISS-C

R — R485

A — R422

T — T485 (17-bit / 23-bit Tamagawae)(2)

D — BUS (high speed bus)

P — PERIOD (periodic send)

Baud rate

M — Default: 2.5 M

V – 5M

A –115200

Operating temperature

A — -40 ~ 85 °C

B — -40 ~ 105 °C

C — -40 ~ 125 °C

(1) The SSI protocol does not support CRC verification. For improved reliability, the BiSS-C protocol using the same hardware platform is recommended

(2) For the Tamagawa protocol, 17-bit corresponds to 17M-T / 17BM-T / 17FM-T, while 23-bit corresponds to 23M-T / 23BM-T / 23FM-T

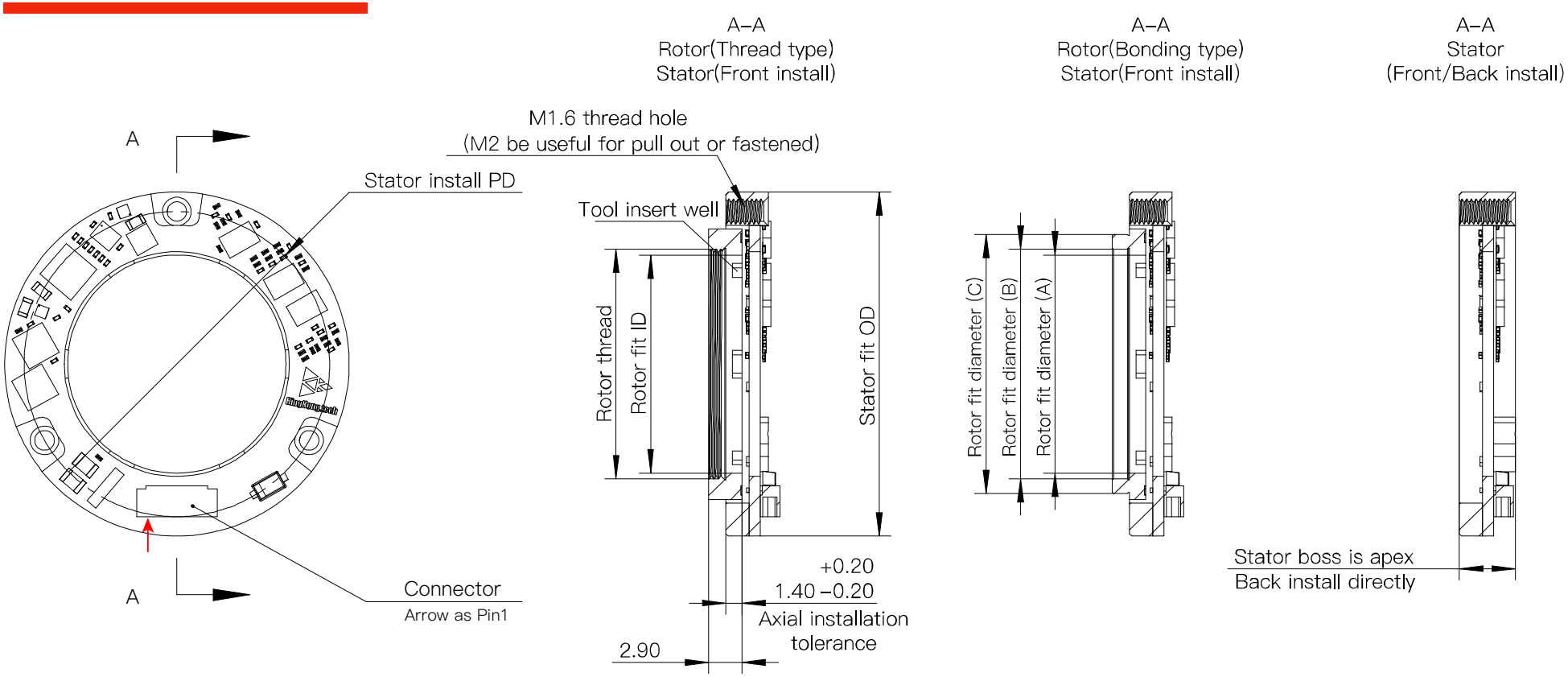
(3) Includes RS485, RS422, BUS, and PERIOD. See following pages for other protocols

(4) Fixed periodic transmission interval of 1 ms

Size

Series	Size	Rotor type	Stator type	Inner rotor thread	Rotor ID(H7)			Stator fit OD (h7)	Stator mounting PD	Stator back install directly[2]	Connector model
					A	B	C				
10	MPT-10-20	Thread	Front	M10x0.4mm		9		20	16.6	–	X0600WRS-08-LPV01HF
	MPT-10B-20	Bonding	Front	–	9	10	12.6			–	
	MPT-10-20H	Thread	Front/Back	M10x0.4mm		9				✓	
	MPT-10B-20H	Bonding	Front/Back	–	9	10	12.6			✓	
15	MPT-13-25	Thread	Front	M13x0.4mm		12		25	21.6	–	
	MPT-13-25H	Thread	Front/Back	M13x0.4mm		12				✓	
	MPT-14-25	Thread	Front	M14x0.4mm		13				–	
	MPT-14-25H	Thread	Front/Back	M14x0.4mm		13				✓	
	MPT-15-25	Thread	Front	M15x0.4mm		14				–	
	MPT-15B-25	Bonding	Front	–	14	15	17.6			–	
	MPT-15-25H	Thread	Front/Back	M15x0.4mm		14				✓	
	MPT-15B-25H	Bonding	Front/Back	–	14	15	17.6			✓	
20	MPT-20-30	Thread	Front	M20x0.4mm		19		30	26.6	–	X0800WRS-08HF-LPV01
	MPT-20B-30	Bonding	Front	–	19	20	22.6			–	
	MPT-20-30H	Thread	Front/Back	M20x0.4mm		19				✓	
	MPT-20B-30H	Bonding	Front/Back	–	19	20	22.6			✓	
25	MPT-25-35	Thread	Front	M25x0.4mm		24		35	31.6	–	
	MPT-25B-35	Bonding	Front	–	24	25	27.6			–	
	MPT-25-35H	Thread	Front/Back	M25x0.4mm		24				✓	
	MPT-25B-35H	Bonding	Front/Back	–	24	25	27.6			✓	
30	MPT-30-40	Thread	Front	M30x0.4mm		29		40	36.6	–	
	MPT-30B-40	Bonding	Front	–	29	30	32.6			–	
	MPT-30-40H	Thread	Front/Back	M30x0.4mm		29				✓	
	MPT-30B-40H	Bonding	Front/Back	–	29	30	32.6			✓	
35	MPT-35-45	Thread	Front	M35x0.4mm		34		45	41.6	–	
	MPT-35B-45	Bonding	Front	–	34	35	37.6			–	
	MPT-35-45H	Thread	Front/Back	M35x0.4mm		34				✓	
	MPT-35B-45H	Bonding	Front/Back	–	34	35	37.6			✓	

Drawings



The “red arrow” in the figure indicates Pin 1. For details, please refer to Page 5.

For stator mounting, either M1.6 screws can be passed through the M2 threaded holes, or M2 screws can be used with the M2 threads for fastening.

For rotor mounting, a dedicated tool should be used for tightening. For details, please refer to the DPT Design and Mounting Guidelines.

Calibration must be performed after mounting. For calibration software, procedures, and drawings, please contact KingKong Technology.

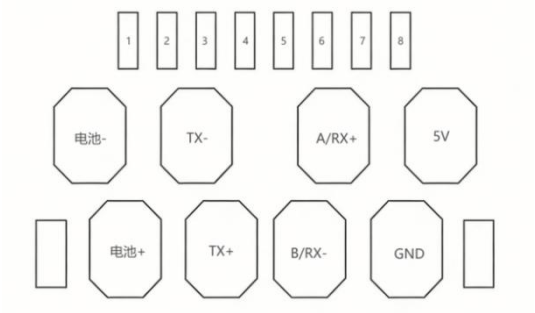
Electrical connections

Connectors

	Wire-to-Board Connector
Model	See the Size
Type	Wire to board
Wire	MPT-10 and MPT-15 use 4P × AWG34 Teflon twisted-pair cable, while all other sizes use 4P × AWG32 Teflon twisted-pair cable.

Pad Connector

Pads under the connector (model suffix: PC) require manual soldering to the corresponding pins. Glue fixation is recommended.



Pin descriptions

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

* Battery +/– only applies to battery-backed multi-turn (BM) versions
* Battery – is internally connected to GND.
* Shield layer is recommended to be grounded at the driver side.

Specifications

System Parameters

Mounting	Axial hollow
Accuracy	±0.01° (assembly dependent)

Electrical Parameters

Supply Voltage	4.5 ~ 5.5 V
Battery Voltage	2.7 ~ 3.6 V
Connection	Wire to board connector
Current Consumption	≈ 100 mA
Low-Power Current	≈ 12 μA (static power consumption)
ESD resistance	HBM, max. ±2 kV CDM, max. ±1 kV

Mechanical Parameters

Rotor Bracket	Stainless steel
Stator Bracket	Aluminum alloy

Environmental Parameters

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

Specification

Output Format

Single-Turn

Outputs angular position data

Standard Multi-turn

Outputs turn count and angular position data	Turn count is not retained after power-off and resets from zero after power-on

Battery Multi-turn

Outputs turn count and angular position data	After power-off, the encoder enters low-power mode powered by the battery, allowing continued rotation tracking and valid turn count retention after power-on

Flash multi-turn

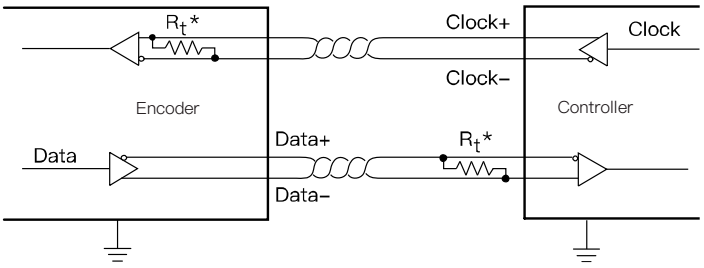
Output the data of angle and multi-turn	After power-off, multi-turn data is stored in flash memory. Only ±90° mechanical rotation is allowed; multi-turn data becomes unreliable beyond this range

Absolute Parameters

Resolution	24 bit
Maximum Speed	8,000 rpm
Update Rate	50 kHz
Repeatability	0 ~ ±1 bit (depending on resolution selection)
Output interface	SSI、BISS-C、RS485、RS422、T485、BUS、PERIOD

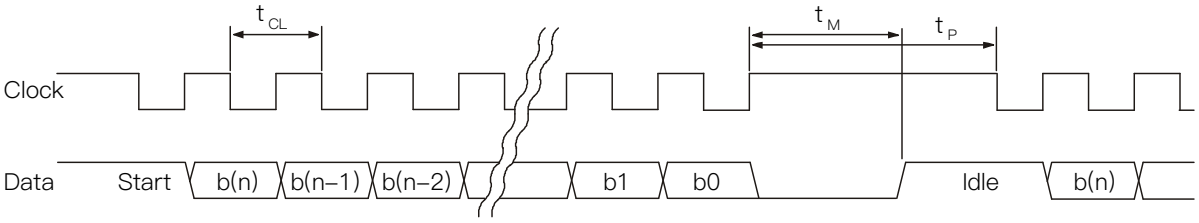
SSI protocol interface

Electrical connection diagram:



The interface uses four differential signal lines: Clock+/Clock— and Data+/Data—. The Clock termination resistor is integrated inside the encoder. A termination resistor or other impedance matching method is recommended on the controller-side Data line.

Timing Diagram:



This protocol uses the Clock signal to synchronize data transmission. When the first falling edge of Clock arrives, the current data is latched. Starting from the MSB, data is output to the Data line on each rising edge of Clock, while the controller reads the Data line on each falling edge of Clock until the LSB has been received.

After transmission is completed and the timeout periodt_M expires, the Data line returns to high level. The Clock signal must remain high until the next read operation is allowed, i.e., after the pause time tP. tCL must be shorter than tM. During any read operation, if the interval exceeds tM, the transmission will be terminated.

Timing Parameters:

Parameters	Symbol	Min.	Typ.	Max.
Clock period	t_{CL}	400 ns	–	14 µs
Clock frequency	$1/t_{CL}$	110 kHz	–	1.5 MHz(1)
Transmit timeout	t_M	–	10 µs	–
Pause duration	t_P	20 µs	–	–

Data format:

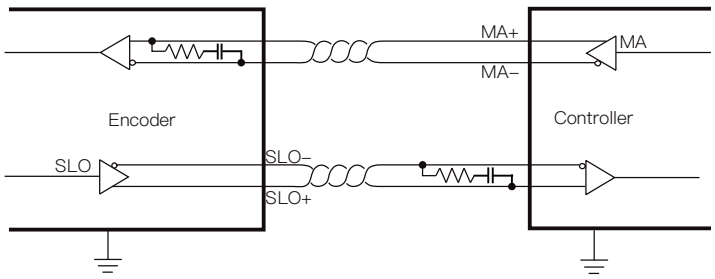
Bits	b(23 + X) : b(8 + X)	b(7 + X) : b8	b7	b6	b5 : b0
Length	16 bits	X bits	1bit	1bit	6 bits
Data	Multi-turn count(1)	Angle	Error bit	Warning bit	Status bit

(1) Multi-turn count is supported only in standard multi-turn, battery multi-turn, and flash multi-turn versions.

For details of the status bits, see the following Status Bits section.

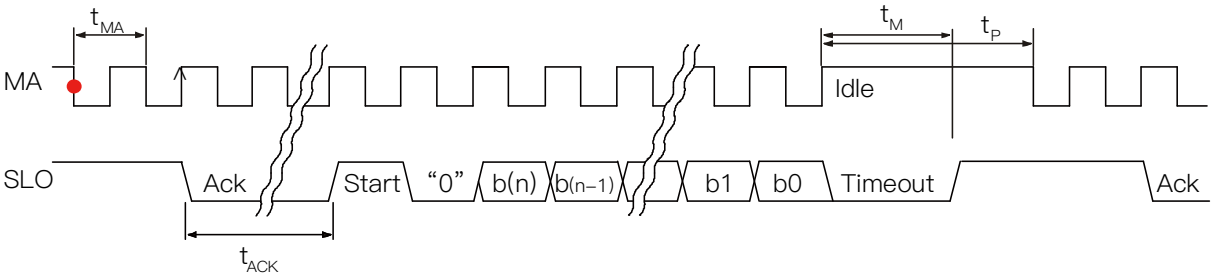
BISS-C protocol interface

Electrical connection diagram:



The interface uses four differential signal lines: MA+/MA− and SLO+/SLO−. The MA termination resistor is integrated inside the encoder. A termination resistor or other impedance matching scheme should be added on the controller-side SLO line.

Timing diagram:



(1) The encoder latches the current angular position data at the red point.

The protocol uses MA for clock synchronization. The MA line remains high when idle. On the first falling edge of MA, the current data is latched and communication starts. The encoder drives SLO low on the second rising edge of MA. Following the start bit "0", data is transmitted MSB first on each rising edge of MA, while the controller samples SLO on each falling edge of MA until the LSB is received.

Timing parameters:

Parameters	Symbol	Min.	Typ.	Max..
Clock period	t_{CL}	400 ns	–	14 µs
Clock frequency	f	120 kHz	–	2.5 MHz(1)
ACK length	t_{ACK}	–	5 bits	–
Transmit timeout	t_M	–	10 µs	–
Pause duration	t_P	20 µs	–	–

After the transfer is complete, when the t_M 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 t_P time. t_{CL} must be less than t_m, and the read can be terminated by making the time exceed t_m while any read operation is in progress.

After transmission is completed and the timeout period t_M expires, the SLO line returns high. The MA signal must remain high until the next read operation is allowed, i.e., after the pause time t_P. tCL must be shorter than t_m. During any read operation, if the interval exceeds t_m, the transmission will be terminated.

Data format:

Bit	b(23 + X) : b(8 + X)	b(7 + X) : b8	b7	b6	b5 : b0
Length	16 bits	X bits	1bit	1bit	6 bits
Data	Multi-turn Count(1)	Angular position data	Error bit(2)	Warning bit(3)	CRC(4)

(1) Multi-turn count is only available in standard multi-turn, battery multi-turn, and flash multi-turn versions.

(2) The error bit is active low and may only become low in battery-backed multi-turn versions. When the bit is high, all battery-related status bits are normal and the multi-turn data is valid. When the bit is low, the battery low-voltage or battery disconnection status bit has been triggered. Refer to the Status Bits section for corrective actions.

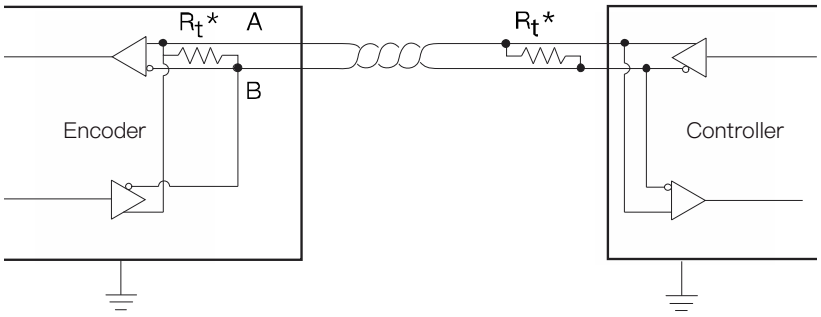
(3) The warning bit is active low. When the bit is high, no error or warning condition is present. When the bit is low, at least one error or warning condition has been triggered.

(4) The CRC polynomial is x6 + x1 + 1 (i.e. 0x43), According to the BISS-C protocol, the calculated CRC is inverted before transmission. The appendix "CRC-6 Calculation" provides directly portable reference code.

For detailed descriptions of the error and warning bits, refer to the Status Bits section.

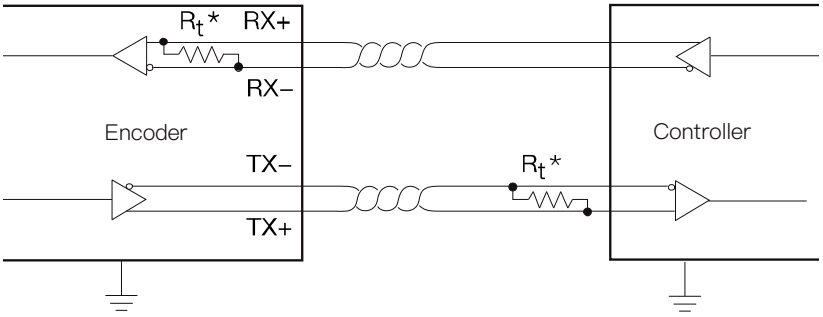
RS485/RS422 protocol interface

RS485 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. The interface uses a two-wire differential A/B signal configuration. Termination resistors are required on both lines. The encoder-side termination resistor is integrated inside the encoder, and users should add termination resistors or other impedance matching schemes on the controller-side A and B lines.

RS422 Electrical connection diagram:



The interface uses four differential signal lines, including differential TX output and differential RX input. The RX termination resistor is integrated inside the encoder. Users should add a termination resistor or other impedance matching scheme on the controller-side RX line.

The underlying protocol of both RS485 and RS422 is UART. Since no clock line is provided, the encoder and controller must operate with the same predefined baud rate and data format to ensure proper data transmission.

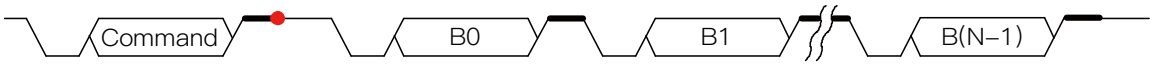
Protocol configuration:

Length	Parity	Stop bit	Flow Control	Byte order
8 bit	–	1	–	LSB first

Supported Baud Rate:

Code	A	M	V
Baud rate (Mbps)	0.1152	2.5	5

Sequence chart:



(1) The current angular position data is latched at the red point.

Command and Data:

Com-mand	Function		Output parameter	N	Return Data (N Bytes)							
					B0	B1	B2	B3	B4	B5	B6	B7
0x30	set	Zero(1)	–	2	C	CRC						
0x31	get	Position	24	4	A2	A1	A0	CRC				
			24M	6	M1	M0	A2	A1	A0	CRC		
			24BM									
			24FM									
0x64	get	Position + Status	24	5	A2	A1	A0	S	CRC			
			24M	7	M1	M0	A2	A1	A0	S	CRC	
			24BM									
			24FM									
			24BM									
			24FM									

(1) Zero setting requires sending commands “0x31” and “0x30” alternately for 10 consecutive cycles. Zero setting is triggered when command “0x30” returns a count value of 10.

In the table above, the letter definitions are as follows:

M	A	C	S	CRC
Multi-turn	Angular Position	Zero-Setting Counter Value	Status	CRC check(1)

For a detailed description of the status bits, refer to Status section.

(1) CRC byte (CRC polynomial is $x^8+x^7+x^4+x^2+x^1+1$, See Appendix for calculation method_CRC-8 table x8+x7+x4+x2+x1+1)

Example (24BM):

If command 0x31 is sent and uint8_t Buffer[7] is received:

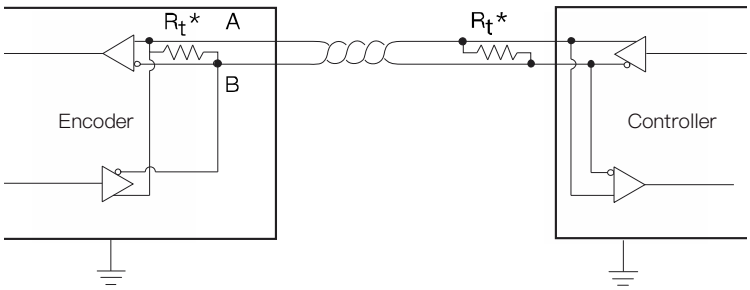
In use:

```
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

Compatible with the Tamagawa protocol.

Electrical connection diagram:



The interface uses a two-wire differential A/B signal configuration. Termination resistors are required on both lines. The encoder-side termination resistor is integrated inside the encoder, and users should add termination resistors or other impedance matching schemes on the controller side.

T485 is based on the RS485 interface and implements a dedicated communication protocol. The interface receives a 1-byte command request, returns the corresponding encoder data according to the request, and appends a CRC-8 checksum at the end of the data for verification.

Protocol configuration:

Character Length	Parity	Stop Bit	Flow Control	Byte order
8 bit	–	1	–	LSB first

Timing Diagram:



(1) The current angular position data is latched at the red point.

Command Request Data:

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

Return Data:

Byte	B0	B1	B(2 ~ n)	B(n + 1)
Data content	Operation Request(1)	Status	Return Data	CRC(2)

(1) The returned operation request matches the transmitted operation request.

(2) CRC byte (CRC polynomial is $x^8 + 1$, see Appendix CRC-8 calculate for calculation method)

B1 Status Format:

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

B(2 ~ n), operation types and the corresponding return data (A for angle; M for Multi-turn; E for error) :

Operation types					Request	n	Return Data							
b7	b6	b5	b4	b3			B2	B3	B4	B5	B6	B7	B8	B9
0	0	0	0	0	Get Position	4	A0	A1	A2					
1	0	0	0	1	Get multi-turn	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 multi-turn(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 exist in the B (2~n) table, a communication error will be triggered and the return data will be the same as that of the "Get Position" operation

(3) Reset Position becomes effective only after the command is sent continuously 10 times

(4) Reset Multi-turn Data becomes effective only after the command is sent continuously 10 times

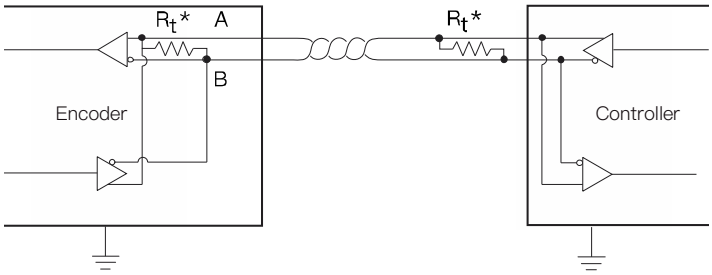
E, Error and Warning Bits:

b7	b6	b5	b4	b3	b2	b1	b0
Error Bit	Warning Bit	–	–	–	–	–	–
Battery Disconnect	Battery Low Voltage	–	–	–	–	–	–

For the T485 interface, the error and warning bits only indicate battery-related status in battery-backed encoder versions. Battery and LED indications are described in the Status Bits section.

BUS (High speed bus) protocol interface

Electrical connection diagram:



The interface uses a two-wire differential A/B signal configuration. Termination resistors are required on both lines. The encoder-side termination resistor is integrated inside the encoder, and a termination resistor should be added on the controller side.

The protocol uses RS485 electrical signaling and UART data format, with a default baud rate of 2.5 Mbps. The interface receives a 1-byte command request, returns the corresponding encoder data according to the request, and appends a CRC-8 ($x^8 + 1$) checksum at the end of the data for verification.

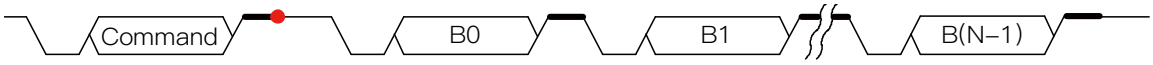
Version Selection:

Single-turn Resolution	Multi-turn Resolution	Model
XX	—	XX-D
	16	XXM-D

Protocol Configuration:

Character Length	Parity	Stop bit	Flow control	Bit Order
8 bit	—	1	—	LSB first

Timing Diagram:



(1) The current angular position data is latched at the red point.

Command Request Data:

Data Bits	b7	b6 ~ b5	b4 ~ b0
Data content	Odd Parity	Operation Type	Device Address

Return data:

Byte	B0	B(1 ~ n)	B(n + 1)
Data content	Operation request	Return Data	CRC

B(1 ~ n), Operation Types and Corresponding Return data (M: Multi-turn Data, A: Single-turn Position, C: Setting Counter, S: Status) :

Operation type		Description	Model version		n	Return Data					
b6	b5		Single-turn Resolution	Multi-turn Resolution		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		Request	n	Data		
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 detailed descriptions of the status bit, refer to the Status section.

Note :

(1) If the operation type does not exist in the B(1 ~ n) table, e.g., 0b11, no response will be returned..

(2) C represents the consecutive setting count returned during encoder configuration. When the returned value reaches 10, the setting is executed immediately. This process may take up to approximately 50 ms, during which the encoder will not respond to any commands.

(3) The factory default address is 0x1F (0b11111).

Set zero position:

To prevent accidental zero setting, operation types 00 and 01 must be sent alternately for a total of 10 consecutive cycles (00, 01, 00, 01, …, 00, 01). Each command must be sent only after the encoder response to the previous command has been

completely received. Zero setting is completed only after all 10 cycles are executed successfully. The remaining number of required cycles can be determined from the C value returned by operation type 01.

Note:

(1) If the command preceding 01 is not 00, the sequence start condition is not satisfied, and the C value is 0.

(2) If the command two cycles before 01 is not 01, the sequence condition is not satisfied, and the C value is 1. Counting starts from this point.

Set ID Operation:

To prevent accidental ID configuration, a specific sequence of address values must be used to enter the Set ID mode before configuring the ID. During the Set ID process, the transmitted address value shall be twice the previously returned C value, as shown below:

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

Note:

(1) X represents an arbitrary value, while Y represents the actual address value to be configured.

(2) No response is returned for the first three sequence groups. This mechanism is intended to prevent accidental triggering during bus operation. Since this command bypasses the ID limitation, unintended transmission could otherwise cause all devices on the bus to respond simultaneously

(3) If any other command is inserted into the sequence, the C value returned by the next Set ID command will be 1, and counting restarts from this point.

(4) If the transmitted address value does not match the required sequence, the returned C value will be 1, and counting restarts from this point.

Bus devices:

All devices on the bus shall enter a temporary “sleep” state when the received ID does not match their own device ID. During this period, the device shall not respond to any commands on the bus, preventing response ripple effects on the bus.

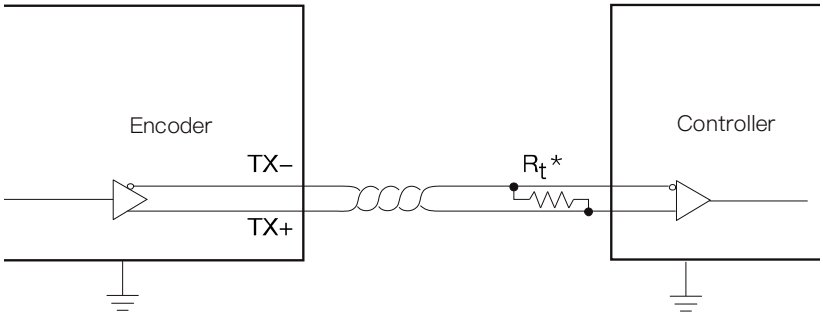
For scalability, the sleep duration is defined as $T_{SUSPEND}$, The calculation method is shown in the table below:

Single-turn Resolution	Multi-turn Resolution	Bus Sleep Bytes BSUSPEND $B_{SUSPEND}$	Bus Sleep Time $T_{SUSPEND}$ (us)
XX	YY	ceil(XX/8) + YY/8 + 4	$B_{SUSPEND}$ * (1 + 8 + 1) / 2.5

Example: For the 16M1-D model, $T_{SUSPEND} = \left(ceil\left(\frac{16}{8}\right) + \frac{8}{8} + 4\right) * \frac{1+8+1}{2.5} = 28us$

Period Periodic Output protocol Interface

Electrical connection diagram:



The interface uses a two-wire differential A/B signal configuration. Termination resistors are required on both lines, and a termination resistor should be added on the controller side.

This protocol is based on the RS422 protocol. The only difference is that the encoder actively transmits data through TX at a fixed interval of 1 ms and does not respond to data on the RX line. Internally, the encoder periodically triggers command “d” (0x64) to transmit the corresponding command data, refer to RS485/RS422 protocol interface section for details.

Status Bits

The error bit, warning bit, and status bits are used to indicate the current operating status of the encoder. The functions of the error bit and warning bit are consistent across all interfaces. However, for different functional versions of the MPT product series, the error bit and warning bit may represent different information.

The error bit and warning bit mainly indicate alarms and error conditions related to the single-turn absolute position and the battery-backed or flash multi-turn functions. Detailed alarm and error information is shown in the table below:

	LED On	LED Flashing	
	General Indication	Battery Multi-turn Version	Flash Multi-turn Version
Warning Bit	The encoder detects a possible mounting deviation, but the encoder is still operating normally.	Battery voltage is below 2.9 V.	–
Error Bit	The encoder detects an abnormal error condition.	Battery voltage is below 2.7 V or the battery is disconnected.	Mechanical rotation exceeded ±90° during power-off, resulting in abnormal multi-turn data. The multi-turn value is automatically cleared.
Solution	Check for mechanical structure changes or perform remounting and recalibration.	Check the encoder battery voltage and wiring, or replace the encoder battery.	Check the mechanical structure and power on the encoder again.

The LED color changes according to the status of the error bit and warning bit. When the warning bit is asserted, the data remains valid and the LED indicator turns yellow. In this state, some parameters indicated by the status bits are approaching their limit values and can be checked through the status bits. When the error bit is asserted, the data becomes invalid and the LED indicator turns red. Detailed error information can be checked through the status bits. During normal operation, the LED indicator remains green.

In the SSI, RS485, RS422, BUS, and PERIOD interfaces, one byte is used to represent status information.

Error Bit, Warning Bit, and Status Bit:

Bit Position	b7	b6	b5	b4	b3	b2	b1	b0
Function	Error Bit	Warning Bit	Status Bits					

(1) The BiSS-C protocol does not provide detailed status bit information.

When an error or warning occurs, the corresponding error bit or warning bit in the output is asserted. The specific cause can then be identified through the status bits.

LED and Status Bit Mapping:

	Only available in battery multi-turn or flash multi-turn versions		b3	b2	b1	b0
	b5	b4				
Battery Multi-turn	Battery Disconnect	Battery Low Voltage	Reserved			
Flash Multi-turn	Single-turn Rotation Exceeds ±90°	Reserved				
LED Flashing	Yes	Yes	No	No	No	No

Appendix

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

```
//poly = x^8+x^7+x^4+x^2+x^1+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 = x^8+1
//The value of table check is the same as the result of polynomial calculation
// The calculation value of the polynomial is the same as itself

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

    return temp;
}
```

CRC-6 Calculate

```
#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];         //Only align data
        uint8_t placeholder;                    //The first placeholder byte of BISS-C is 0x82
        uint8_t buffer[8] __attribute__ ((aligned(4))); //Buffer of 4 bytes data align, for fast CRC
} receiveBuffer;

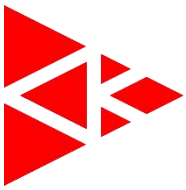
//Configure SPI and DMA
//Use  &receiveBuffer.placeholder  as the receiving address
//......

//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
}
```

Revision history

Time	Edition	Description
2023/09/28	V0.1	Initial release
2024/01/05	V0.2	Add more sizes
2024/05/01	V0.3	Add standard drawing Add colors of wires
2024/12/09	V0.4	Add rotor type with bonding Add stator front/back install type Change pin1 to left of connector
2024/12/22	V0.5	Add latch point of angle
2026/06/01	V0.6	Added speed description Added baud rate type options Added pad connection description Improved protocol descriptions



KingKong.tech

Beijing KingKong Technology Co., Ltd.

Address: 3F, B2, Yard 82, Shuangying West Rd, Science and Technology Park, Changping District, Beijing, China

Website: https://kingkong.tech

Email: contact@kingkong.tech

Tel: +8610-8011-1669