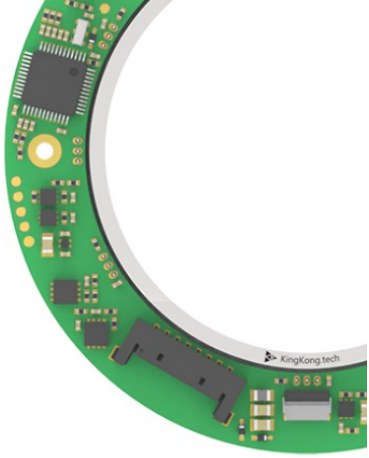
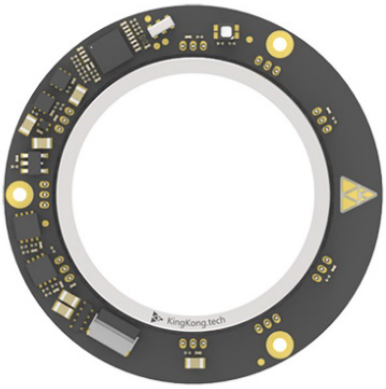
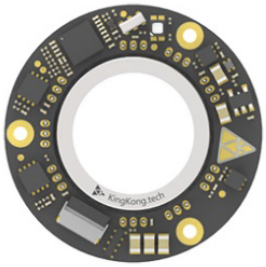
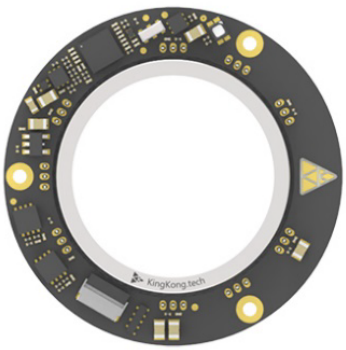
KingKong.tech

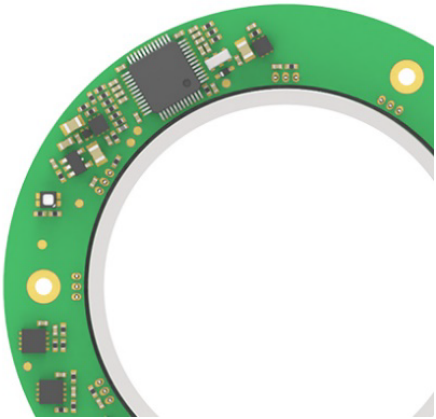


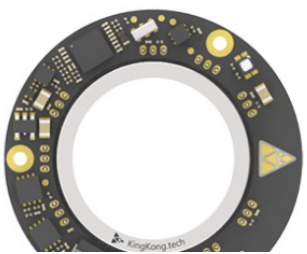
# MBS Magnetic Encoder Datasheet v2.4



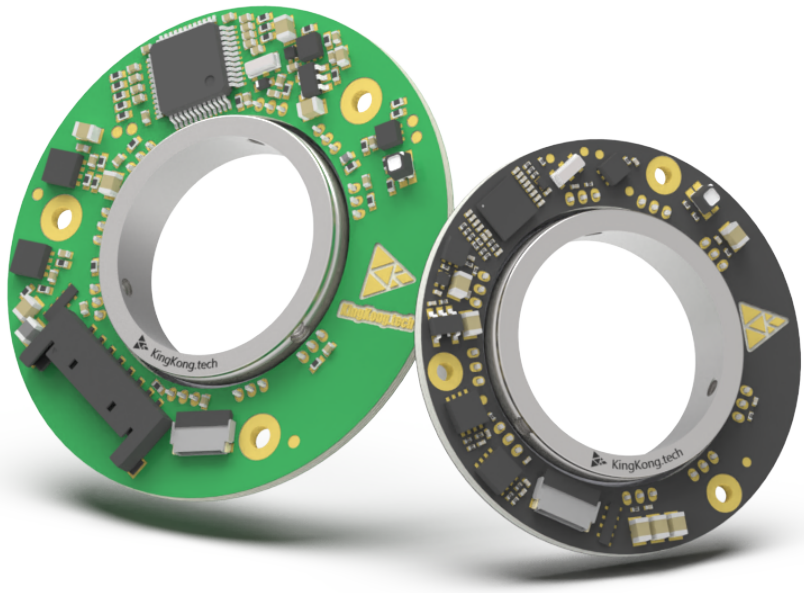








# MBS Magnetic Encoder




MBS magnetic encoder is an ultra-thin absolute encoder with multiple output formats and strong anti-electromagnetic interference capability. It is available in multiple sizes and suitable for scenarios that require high output accuracy and tight spaces.

MBS is driven by magnetoelectric technology and has a unique interference shielding technology. There are multiple high-precision magnetic field sensors inside the encoder to measure the filed strength changes of the rotor magnetic ring, and it is formed by the precision calibration technology provided by KingKong Technology. 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 as vibration, dust, oil pollution, even when it is running at ultra-high speed, without affecting the accuracy and service life.

With unique magnetic shielding technology, the encoder can maintain the normal operation of high-precision output, in most electromagnetic interference environment.

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

- 17-bit absolute output
- ±0.05° accuracy
- Ultra-thin (7 mm)
- Models for every 2~3mm on diameter
- Ring width 8.5mm
- Stator and rotor tolerance fit installation
- Magnetic Interference Shielding Technology
- Hollow structure
- Top speed > 20,000 rpm
- Unique data calibration
- Multi-turn with battery mode
- Various output interfaces
- Resistance to various environmental disturbances

# Models

## MBS-20-45-A-16-S-N-A-M

**Rotor inner diameter**

Refer to Size

**Stator outer diameter**

Refer to Size

**Output type**

A — absolute

**Output parameter**

16 — 16 bits singleturn

17 — 17 bits singleturn

16M — 16 bits singleturn & multiturn

17M — 17 bits singleturn & multiturn

16BM — 16 bits singleturn & battery multiturn

17BM — 17 bits singleturn & battery multiturn

**Output interface**

S — SSI(1)

B — BISS-C

R — R485

A — R422

T — T485 (Tamagawa compatible)(2)

D — BUS (high speed bus)

P — PERIOD (periodic send)

**Input voltage**

N — 5V

**Operating temperature**

A — -40 ~ 85 °C

B — -40 ~ 105 °C

C — -40 ~ 125 °C

**Addition**

M — no addition

Baud rate: interface in R/A

PC — Pad Connector

(1) SSI interface has no CRC check, it is recommended to use BISS-C instead on the same hardware to get more reliability

(2) The only optional output parameters of T485 protocol are 17M or 17BM

# Size

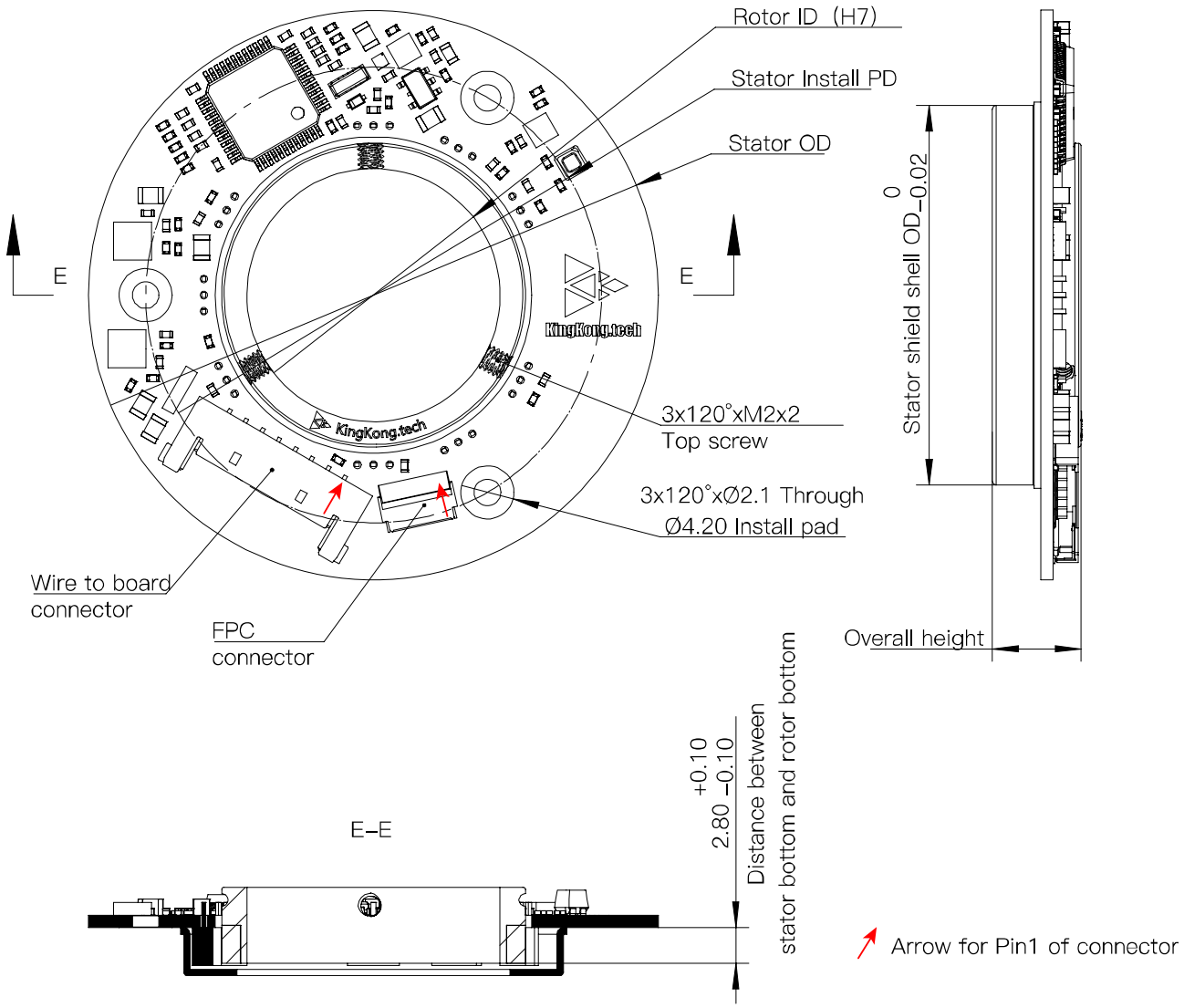
| Series | Type | Size | Rotor ID | Stator OD | Stator mount PD | Stator shield OD | Connector | | Overall thickness |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Wire to board | FPC | |
| 10 | Standard | MBS-8-36 | 8 | 36 | 25 | 20 | ✓ | ✓ | 7 |
| | | MBS-10-36 | 10 | | | | | | |
| | Tiny | MBS-8-27 | 8 | 27 | 23 | | – | ✓ | |
| | | MBS-10-27 | 10 | | | | | | |
| 15 | Standard | MBS-12-40 | 12 | 40 | 30 | 25 | ✓ | ✓ | |
| | | MBS-13-40 | 13 | | | | | | |
| | | MBS-15-40 | 15 | | | | | | |
| | Tiny | MBS-12-32 | 12 | 32 | 28 | | – | ✓ | |
| | | MBS-13-32 | 13 | | | | | | |
| | | MBS-15-32 | 15 | | | | | | |
| 20 | Standard | MBS-6-45 | 6 | 45 | 36 | 30 | ✓ | ✓ | |
| | | MBS-8-45 | 8 | | | | | | |
| | | MBS-10-45 | 10 | | | | | | |
| | | MBS-15-45 | 15 | | | | | | |
| | | MBS-18-45 | 18 | | | | | | |
| | | MBS-20-45 | 20 | | | | | | |
| | Tiny | MBS-18-37 | 18 | 37 | 33 | | – | ✓ | |
| | | MBS-20-37 | 20 | | | | | | |
| 25 | Standard | MBS-23-50 | 23 | 50 | 40 | 35 | ✓ | ✓ | |
| | | MBS-25-50 | 25 | | | | | | |
| | Tiny | MBS-23-42 | 23 | 42 | 38 | | – | ✓ | |
| | | MBS-25-42 | 25 | | | | | | |
| 30 | Standard | MBS-28-55 | 28 | 55 | 45 | 40 | ✓ | ✓ | |
| | | MBS-30-55 | 30 | | | | | | |
| | Tiny | MBS-28-47 | 28 | 47 | 43 | | – | ✓ | |
| | | MBS-30-47 | 30 | | | | | | |
| 35 | Standard | MBS-33-60 | 33 | 60 | 50 | 45 | ✓ | ✓ | |
| | | MBS-35-60 | 35 | | | | | | |
| | Tiny | MBS-33-52 | 33 | 52 | 48 | | – | ✓ | |
| | | MBS-35-52 | 35 | | | | | | |
| 40 | Standard | MBS-38-65 | 38 | 65 | 55 | 50 | ✓ | ✓ | |
| | | MBS-40-65 | 40 | | | | | | |
| | Tiny | MBS-38-57 | 38 | 57 | 53 | | – | ✓ | |
| | | MBS-40-57 | 40 | | | | | | |
| 45 | Standard | MBS-43-70 | 43 | 70 | 60 | 55 | ✓ | ✓ | |
| | | MBS-45-70 | 45 | | | | | | |
| | Tiny | MBS-43-62 | 43 | 62 | 58 | | – | ✓ | |
| | | MBS-45-62 | 45 | | | | | | |
| 50 | Standard | MBS-48-75 | 48 | 75 | 65 | 60 | ✓ | ✓ | |
| | | MBS-50-75 | 50 | | | | | | |
| | Tiny | MBS-48-67 | 48 | 67 | 63 | | – | ✓ | |
| | | MBS-50-67 | 50 | | | | | | |
| 55 | Standard | MBS-53-80 | 53 | 80 | 70 | 65 | ✓ | ✓ | |
| | | MBS-55-80 | 55 | | | | | | |
| | Tiny | MBS-53-72 | 53 | 72 | 68 | | – | ✓ | |
| | | MBS-55-72 | 55 | | | | | | |

Download 3D models: https://kingkong.tech/encoder/MBS

# Drawings

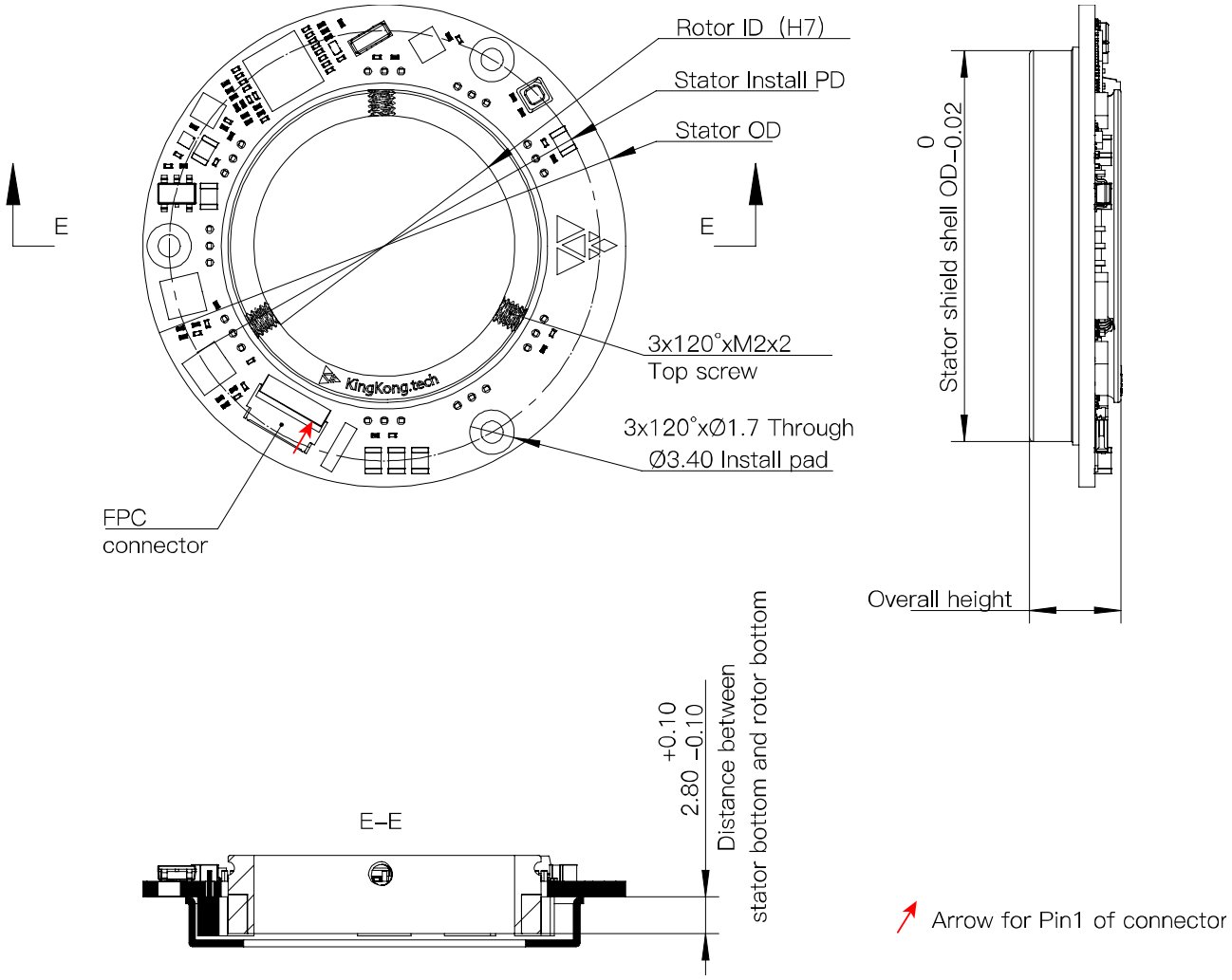
## Standard

Standard stator and rotor drawings



## Tiny

Tiny stator and rotor drawings



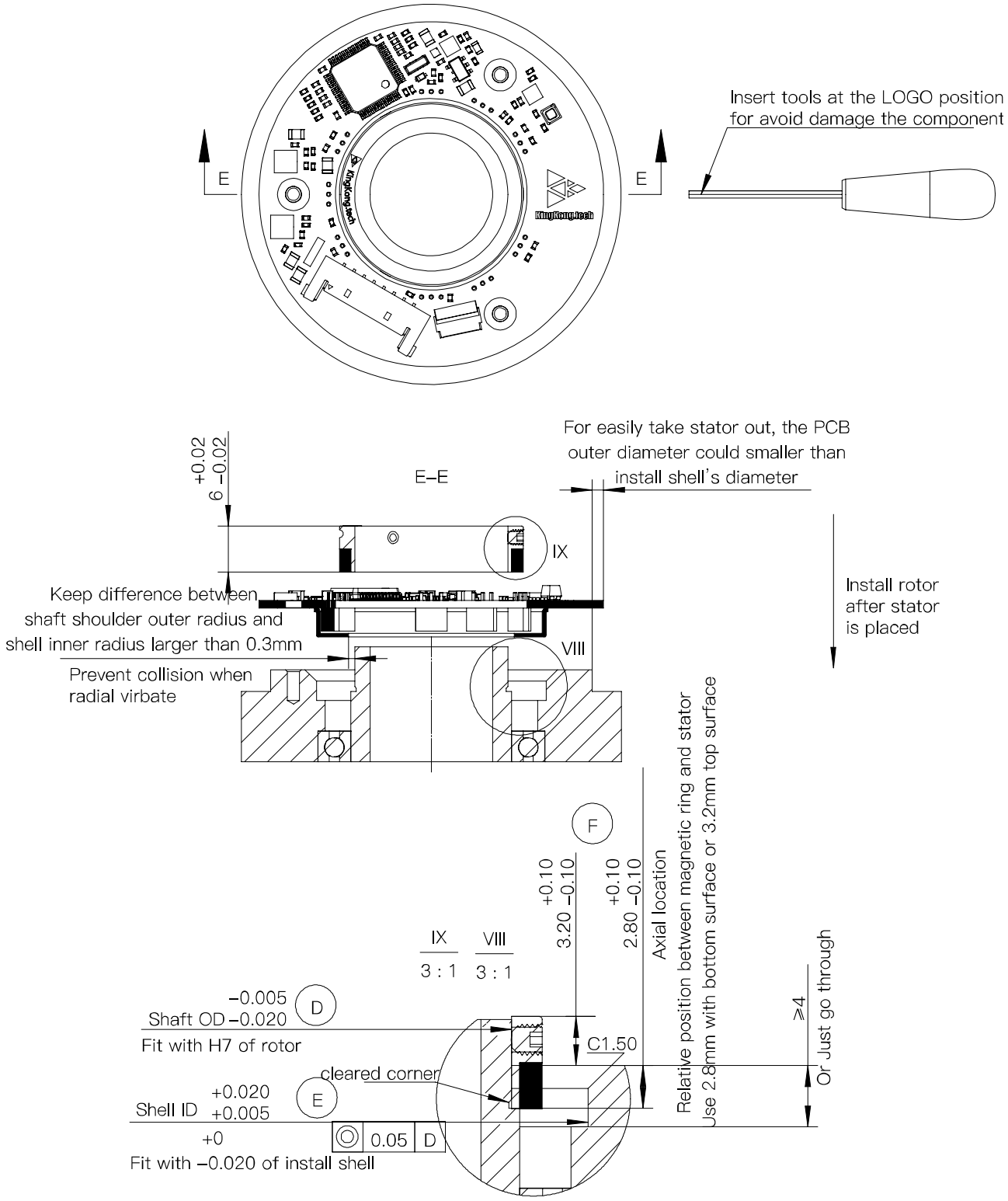

## Stator and rotor tolerance fit installation

Use the tolerance of the stator shield outer diameter to locate the radial of the encoder body, and install the encoder rotor with the target shaft by the inner diameter tolerance. These two operations ensure the concentricity of the stator and rotor of the encoder, thereby basically eliminating installation error.

The rotor should be located inside the shield of the stator, and it should be inserted in the axial direction during installation, and should be tightened with the shaft by the radial top screw. To ensure the encoder work normally, the distance between the bottom surface of the rotor and the stator shield should be less than 1mm. In order to achieve the highest measurement accuracy, the rotor should be axially positioned using the shaft shoulder, and the distance between the shaft shoulder and the mounting surface of the stator is C. During installation and use, when C is 2.8mm, the encoder can restore the measurement accuracy of factory calibration to the greatest extent. For details, please refer to the installation suggestions in the next chapter.

## Design and Installation Recommendations

Stator shield tolerance fit with shell inner circle; rotor bracket tolerance fit with shaft outer circle



There are three tolerance fit, which respectively locate the:

1. Radial position of the encoder rotor (D)

2. Radial position of the encoder stator (E)

3. Relative axial position between encoder rotor and stator (F)

The three dimensions and positions can clearly determine the relative position between the encoder stator and the rotor. When the relative position is determined, the repeatable accuracy and absolute accuracy of the angle can be locked.

The installation method shown in the figure can restore the relative position of the factory calibration between the rotor and the stator as much as possible. Installed with the same tolerance, the best data accuracy can be directly obtained without post-calibration.

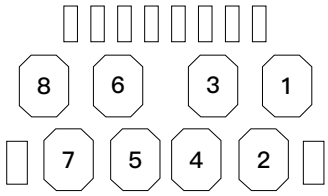
# Electrical connections

## Connectors

| Model | MOLEX 504050-0891 | FH34SRJ-8S-0.5SH | Soldering Pad |
|---|---|---|---|
| Type | Wire to board | FPC double-sided | Manual solder |
| Wire | 4PxAWG28 twisted pair (OD≈4.0mm) | 8P 0.5mm FPC | Recommended wire diameter: less than AWG28 |
| Scenario | Long wire, such as: servo motor | Small space or the driving circuit is head of encoder | Compact space where the it is so small that FPC cannot be used |

## Soldering pad

Soldering pad under FPC connector (Add PC addition in model), for manual solder corresponding pin(glue after are suggested)




## Pin descriptions

| 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 | Green | Clock + | MA + | RX + | A | A | A | – |
| 4 | Green/Black | Clock – | MA – | RX – | B | B | B | – |
| 5 | Yellow | Data + | SLO + | TX + | – | – | – | TX + |
| 6 | Yellow/Black | Data – | SLO – | TX – | – | – | – | TX – |
| 7 | White | Battery + | Battery + | Battery + | Battery + | Battery + | Battery + | Battery + |
| 8 | White/black | Battery + | Battery – | Battery – | Battery – | Battery – | Battery – | Battery – |

* Battery +/– only in battery multiturn (BM) version
* The battery– is connected to the internal GND of the encoder
* The shield is recommended to be grounded at the driver side

# Parameters

## System

| | |
|---|---|
| **Installation method** | Axial hollow |
| **Accuracy** | ±0.05° |
| **Temperature drift** | ±0.01°/3 °C |

## Electrical

| | |
|---|---|
| **Power supply** | 4.5 ~ 5.5 V |
| **Battery** | 2.7 ~ 3.6 V |
| **Startup time** | 15ms |
| **Connection** | Wire to board connector MOLEX 504050/FPC connector FH34SRJ-8S-0.5SH/Soldering Pad |
| **Current** | ≈ 60mA |
| **Battery mode current** | ≈ 35μA (battery voltage 3.6V, in 6000rpm low power detection mode, see Battery features for details) |
| **ESD resistance** | HBM, max. ±2 kV<br>CDM, max. ±1 kV |

## Mechanical

| | |
|---|---|
| **Rotor material** | Stainless steel<br>Aluminum alloy (MBS-6-45/MBS-8-45/MBS-10-45/MBS-12-45) |

## Environmental

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

# Features

## Maximum speed

With non-contact structure, there is no friction between the rotor and the stator, and the maximum speed can be extremely high, so it can be applied to high speed applications.

## Environmental interference

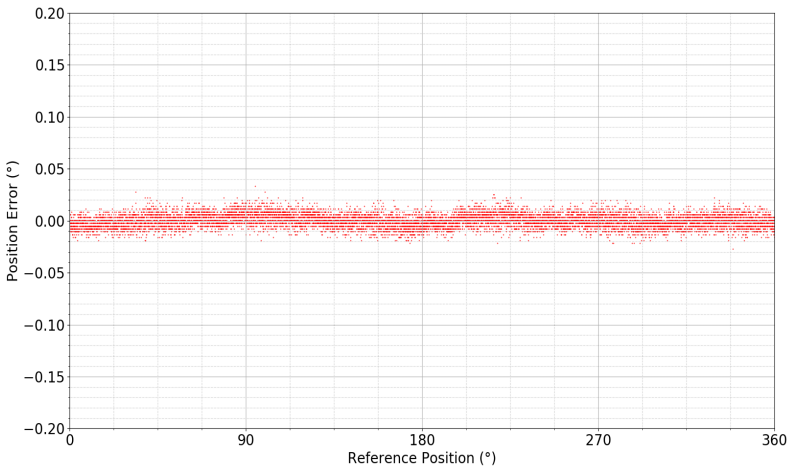
The measurement method of the MBS can bring a certain resistance to vibration, and will not damage the encoder. It has strong anti-interference ability against non-magnetic objects such as oil and dust.

## External magnetic field interference

The MBS encoder has a unique anti-electromagnetic interference technology. The electromagnetic shielding shell can shield the electromagnetic interference transmitted from the environment (such as motor coils), and the magnetic field strength of the magnetic ring is generally much greater than the external interference magnetic field. The combination of the shield and internal magnetic field can stable against external interferences.

## Accuracy

During the magnetization process of the magnetic ring, the problem of low consistency of the magnetic field may occur, however each encoder of KingKong Technology has the unique data information of its corresponding magnetic ring after being calibrated at the factory. Therefore, the error caused by the magnetic field consistency of the magnetic ring can be significantly reduced.




The picture shows the typical encoder accuracy after calibration.

# Specification

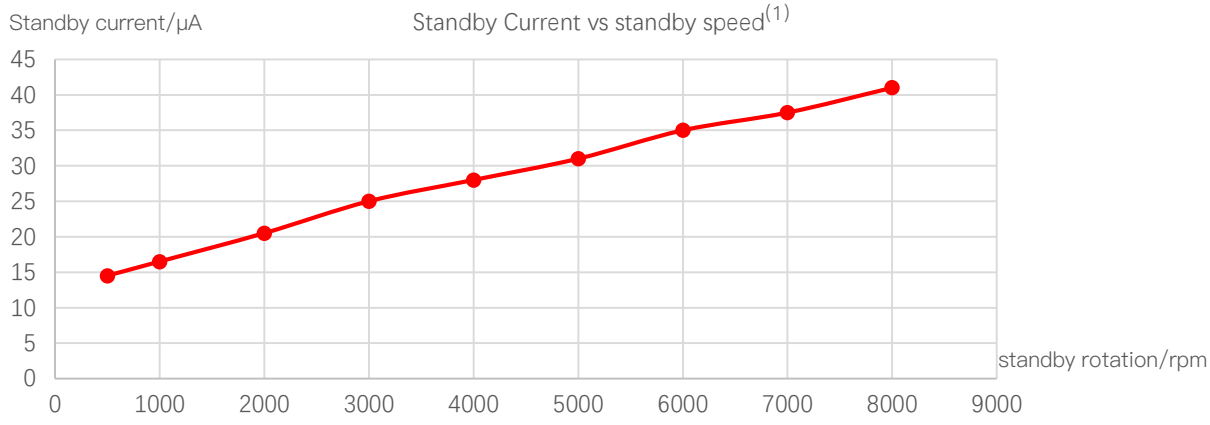
## Format selection

| | |
|---|---|
| **Resolution** | Output angle data |
| **Multiturn** | Output the data of angle and turns |
| The number of turns cannot be kept after power off, and it will reset to zero after repower on | |
| **Battery multiturn** | Output the data of angle and turns |
| 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 | |

## Calculation parameters

| | |
|---|---|
| **Angle resolution** | 17、16 bit |
| **Rotation speed** | > 20,000 rpm |
| **Update frequency** | 27 kHz |
| **repeatable accuracy** | 0 ~ ±1 bit (change by singleturn angle bits selection) |
| **Output interface** | SSI/BISS-C/RS485/RS422/T485/BUS/PERIOD |

## Battery features



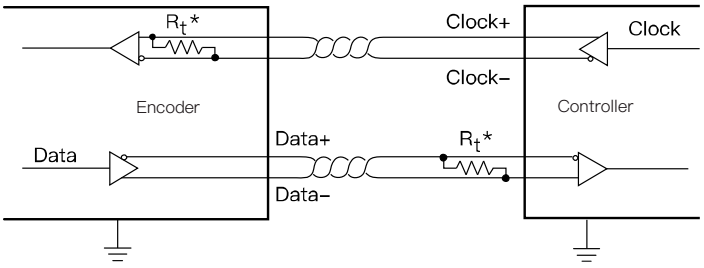

Calculate of battery standby time: $\frac{\text{battery capacity(mAh)}\times 1000}{\text{standby current(μA)}\times 24\times 365}$=standby time(year)

Please select the standby speed according to application to achieve longer battery life and standby time. The default shipment is 6000rpm. Please contact supplier for this option.

(1) Measured at a battery voltage of 3.6v and a temperature of 25℃

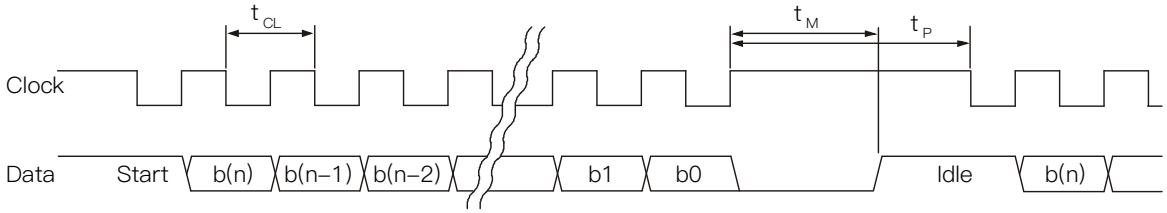
# SSI protocol interface

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 $t_M$ transfer time ends , the Data line goes high and the Clock 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.

Timing Parameters:

| Parameters | Symbol | Min value | Typical value | Max value |
|---|---|---|---|---|
| **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 | | |

(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:

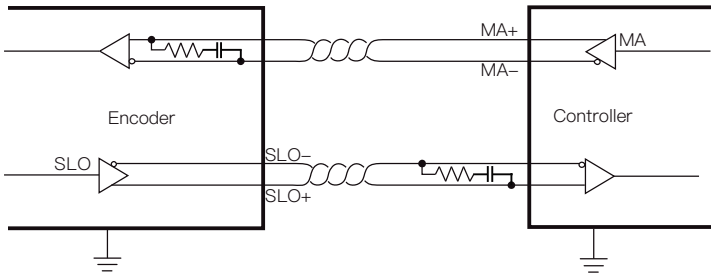
| **Bits** | b(23 + X) : b(8 + X) | b(7 + X) : b8 | b7 | b6 | b5 : b0 |
|---|---|---|---|---|---|
| **Length** | 16 bits | X bits | 1bit | 1bit | 6 bits |
| **Data** | Multiturn count(1) | Singleturn angle | Error bit | Warning bit | Status bit |

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

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

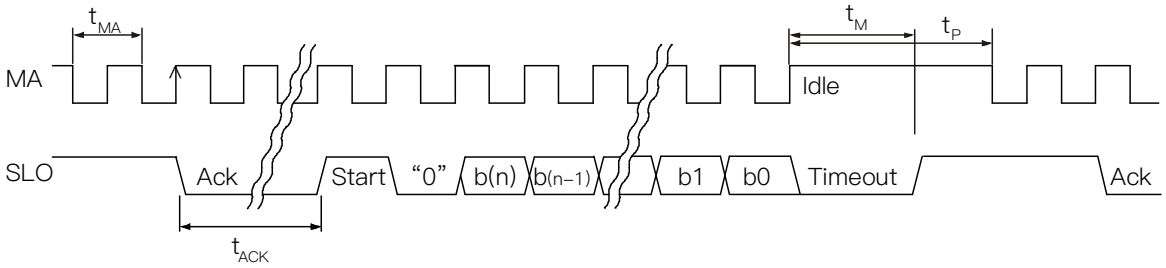
# BISS-C protocol interface

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 falling 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** | $t_{CL}$ | 400 ns | | 14 µs |
| **Clock frequency** | $1/t_{CL}$ | 120 kHz | | 2.5 MHz(1) |
| **ACK length** | $t_{ACK}$ | | 5 bits | |
| **Transmit timeout** | $t_M$ | | 10 µs | |
| **Pause duration** | $t_P$ | 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 $t_M$ 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.

Data format:

| Bits | b(29 + X) : b(14 + X) | b(13 + X) : b14 | b13 | b12 | b11 : b6 | b5 : b0 |
|---|---|---|---|---|---|---|
| Length | 16 bits | X bits | 1bit | 1bit | 6 bits | 6 bits |
| Data | Multiturn count(1) | Signleturn | Error bit | Warning bit | Status bit | CRC(2) |

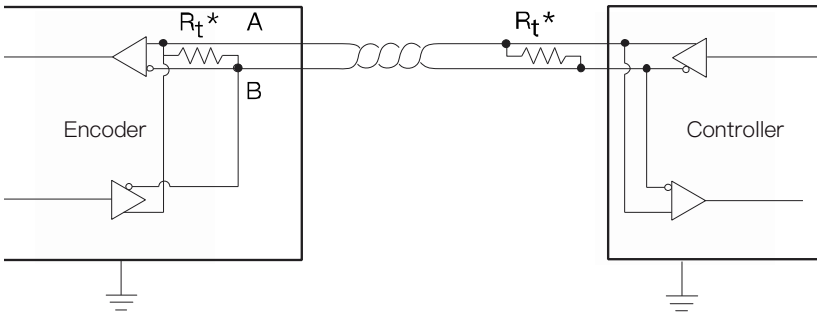
(1) Multiturn count is only available on multiturn and battery multiturn versions

(2) The CRC polynomial is x6 + x1 + 1 (i.e. 0x43), According to the BISS-C protocol requirements, the calculated CRC value will be inverted before send. Appendix CRC-6 calculations gives directly portable calculation codes for easy reference

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

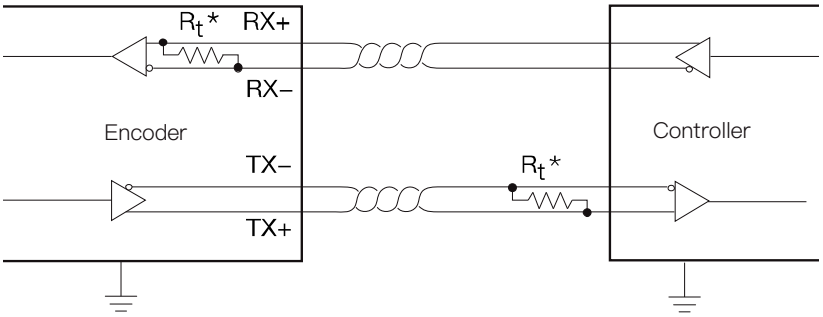
# RS485/RS422 protocol interface

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.

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 frequency 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 (default A if not marked in additional and recommended B):

| Code | A | B |
|---|---|---|
| Baud rate (Mbps) | 0.1152 | 2.5 |

Interactive configuration command:

## Command "0" (0x30) set encoder zero[3]

Return
1 byte count value
1 byte CRC-8 check

## Command "1" (0x31) get position

Return
2 bytes multiturn count [1], MSB
n bytes signleturn [2], MSB
1 byte CRC-8 check

## Command "d" (0x64) get position + status

Return
2 bytes multiturn count [1], MSB
n bytes signleturn [2], MSB
1 byte status
1 byte CRC-8 check

## Command "s" (0x73) get position + rpm

Return
2 bytes multiturn count [1], MSB
n bytes signleturn [2], MSB
2 bytes rpm (the value = round/second * 10), signed, MSB
1 byte CRC-8 check

## Command "t" (0x74) get position + temperature[4]

Return
2 bytes multiturn count [1], MSB
n bytes signleturn [2], MSB
2 bytes temperature (the value = ℃ * 10), signed, MSB
1 byte CRC-8 check [5]

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

(2) If singleturn bits <= 16, n = 2; else if singleturn 16 < bits < 24, n = 3.

(3) Send the command "1" and then command "0" one after another, and cycle it ten times to set the zero position. When the last command "0" is sent and the return count value is 10, the encoder will set the current position to 0. Afterwards, the encoder will not respond to any command for 40ms.

(4) The temperature is the junction temperature of the chip

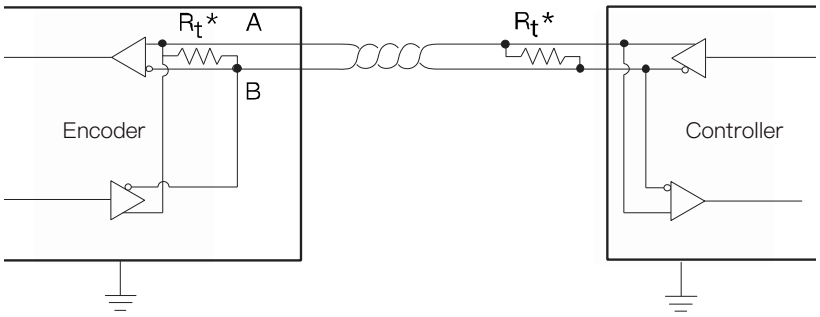
(5) CRC byte (CRC polynomial is $x^8+x^7+x^4+x^2+x^1+1$, See Appendix for calculation method CRC-8 table x8+x7+x4+x2+x1+1)

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

# T485 protocol interface

*This interface is compatible with the tamagawa protocol.

T485 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.

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

Protocol configuration :

| Data length | Parity check | Stop bit | Stream control | Bytes order |
|---|---|---|---|---|
| 8 bit | — | 1 | — | LSB first |

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 | Data | CRC(2) |

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

(2) CRC byte (CRC polynomial is $x^8 + 1$, 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 | | | | | Request | 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(3) | 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):

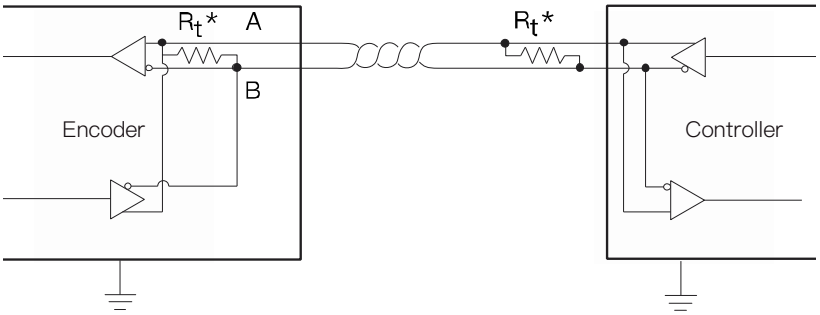
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| Battery lost | Battery low voltage | 0 | Temperature error | 0 | Excessive magnetic field | Weak magnetic field | High speed |

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

Note: Support read/write EEPROM operation in the Tamagawa protocol, and 50ms waiting time is required for data writing.

# 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 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 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 corresponding encoder data and adds CRC-8 (x8+1) to the end of the data.

Version for select:

| Singleturn bits | Multiturn bits | Model number |
|---|---|---|
| XX | — | XX-D |
| | 8 | XXM1-D |
| | 16 | XXM2-D |

Protocol configuration:

| Byte length | Parity check | Stop bit | Stream control | Bytes length |
|---|---|---|---|---|
| 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 | 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 | | Request | Model version | | n | Data | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| b6 | b5 | | Singleturn bits | Multiturn bits | | B1 | B2 | B3 | B4 | B5 | B6 |
| 0 | 0 | Get data | ≤16 | — | 3 | S | A0 | A1 | | | |
| | | | ≤16 | 8 | 4 | S | A0 | A1 | M0 | | |
| | | | ≤16 | 16 | 5 | S | A0 | A1 | M0 | M1 | |
| | | | >16 | — | 4 | S | A0 | A1 | A2 | | |
| | | | >16 | 8 | 5 | S | A0 | A1 | A2 | M0 | |
| | | | >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 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.

Note:

(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.

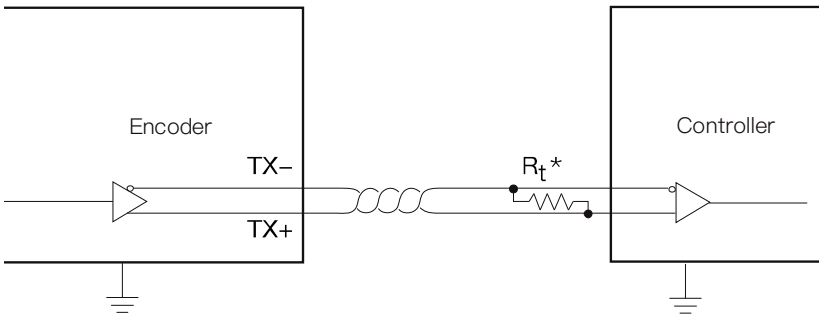
The sleep time is $T_{SUSPEND}$, and the following table shows the calculation:

| Singleturn bits | 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: $T_{SUSPEND} = \left(ceil\left(\frac{16}{8}\right) + \frac{8}{8} + 4\right) * \frac{1+8+1}{2.5} = 28us$

# 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 is1** | – | 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
// 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 = x6+x1+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
}
```

**Beijing KingKong Technology Co., Ltd.**

Address: No. 26, Rd. YongAn, Science and Technology Park, Changping District, Beijing, China

Website: https://kingkong.tech

Email: contact@kingkong.tech

Tel: +86-010-8011-1669