MOSRAC® Mosrac motor

T Magnetic Dual Encoder Data Sheet

T Magnetic dual encoder

T Magnetic dual encoder is specially designed for robot integrated joints. It uses Mosrac motor pattern magnetic technology to measure two axes within a limited volume, achieving photoelectric-like resolution and accuracy, and has high 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.

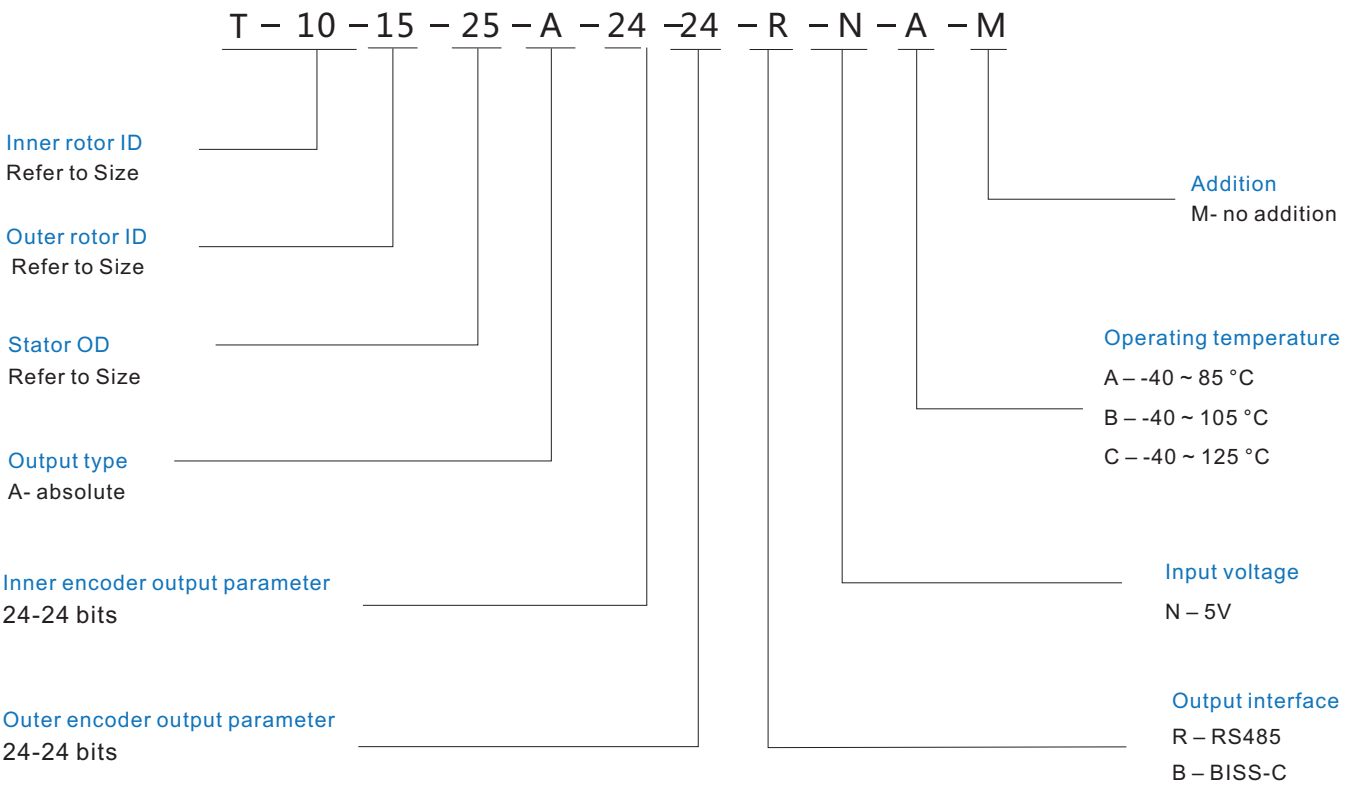
The separated magnetoelectric solution can ensure that the encoder can avoid the interference of external environmental factors such asvibration, dust, oil pollution, even when it is running at ultra-high speed, without affecting the accuracy and service life.

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.

- 24-bit absolute output
- +0.01° accuracy
- Ultra-compact (radial single side 7mm)
- Models for every 5mm on diameter
- Stator and rotor tolerance fit installation
- Hollow structure doesn't limit installation position
- Max speed 8,000rpm
- Unique data calibration
- Various output interfaces
- Resistance to various environmental disturbances

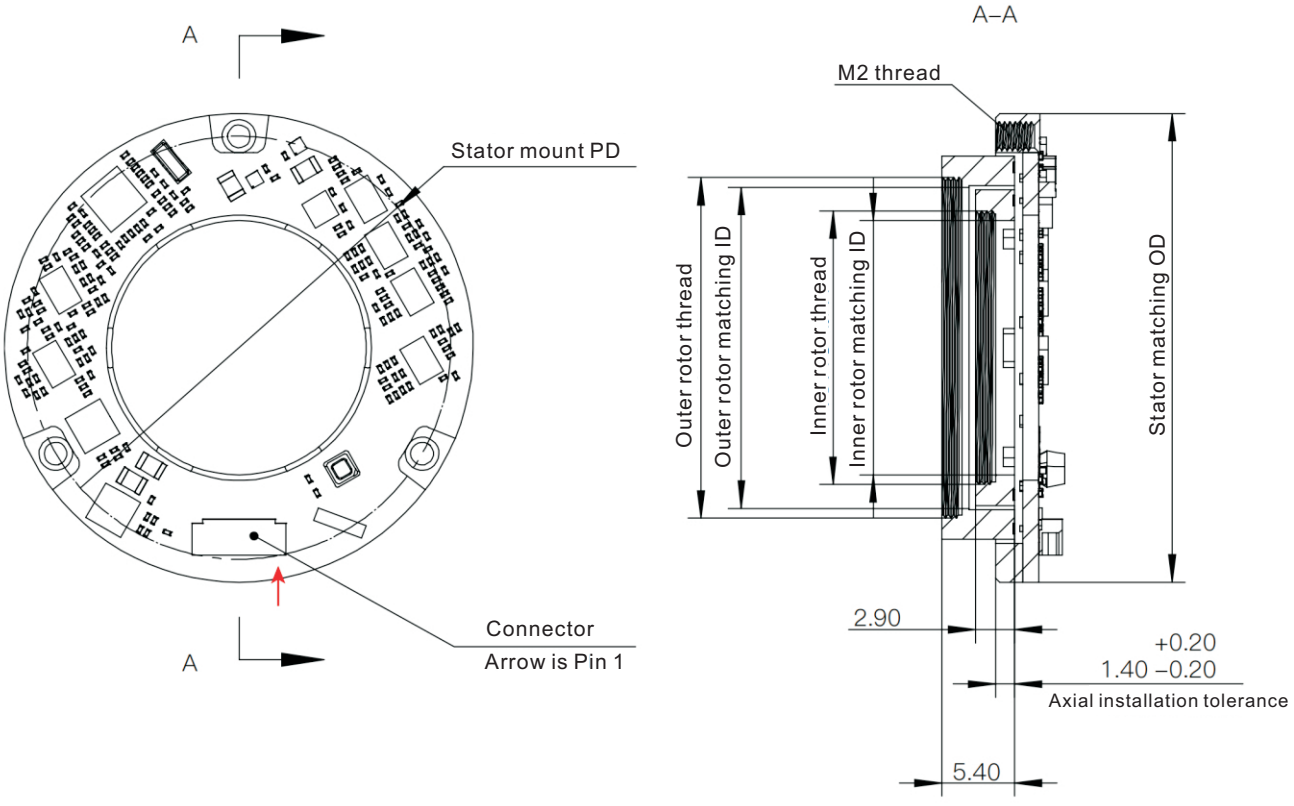
Models



Size

Series	Model	Inner rotor Thread	Inner rotor Match ID (H7)	Outer rotor Thread	Outer rotor Match ID (H7)	Stator Match OD (H7)
10-15-25	T-10-15-25	M10x0.4mm	9	M15x0.4mm	14	25
15-20-30	T-15-20-30	M15x0.4mm	14	M20x0.4mm	19	30
20-25-35	T-20-25-35	M20x0.4mm	19	M25x0.4mm	24	35
25-30-40	T-25-30-40	M25x0.4mm	24	M30x0.4mm	29	40
30-35-45	T-30-35-45	M30x0.4mm	29	M35x0.4mm	34	45

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	SM06B-SURS-TF
Type	Wire-to-board
Wire	AWG32 wiring

Pin

Pin	Color	R	B
		RS485	BISS-C
1	Red	+5V	
2	Black	OV (GND)	
3	Blue	A	MA +
4	Green	B	MA-
5	Yellow	-	SLO +
6	Orange	-	SLO-

Parameters

System

Installation method	Axial hollow
Resolution	24bit
Accuracy	±0.01°

Electrical

Power supply	4.5 ~ 5.5 V
Start-up time	15 ms
Connection method	Wire-to-board connector
Current	≈ 130 mA
ESD resistance	HBM，max. ±2 kV CDM，max. ±1 kV

Mechanical

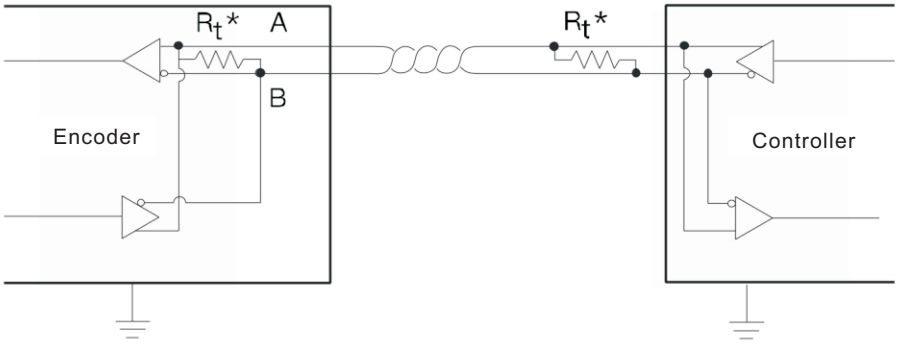
Magnetic ring bracket	Stainless steel

Environmental

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

RS485 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.

The underlying protocol of RS485 is UART. Since this protocol has no clock line, the encoder and controller must work at the same agreed frequency and data format to complete data 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	B	C	D
Baud rate (Mbps)	2.5	5	7.75

Command and data:

Command	Introduction		N	Return data (N bytes)							
				B0	B1	B2	B3	B4	B5	B6	B7
0x29	Set	Inner zero position (1)	2	C	CRC						
0x30	Set	Outer zero position (2)	2	C	CRC						
0x31	Get	Inner angle	4	A0	A1	A2	CRC				
0x32	Get	Outer angle	4	B0	B1	B2	CRC				
0x33	Get	Inner angle Outer angle	7	A0	A1	A2	B0	B1	B2	CRC	
0x41	Get	Inner angle Status information	5	A0	A1	A2	S	CRC			
0x42	Get	Outer angle Status information	5	A0	A1	A2	S	CRC			
0x43	Get	Inner angle Outer angle Status information	8	A0	A1	A2	B0	B1	B2	S	CRC
0x74	Get	Temperature information	3	T0	T1						

The letters in the above table correspond to the following data:

A	B	C	S	T	CRC
Inner encoder angle	Outer encoder angle	Clear reading value	Status	Temperature	CRC check

(1) To set the zero position, need to send command 0x31 and command 0x29 10 times continuous intervals to successfully set it; When the return count value is 10, the trigger sets the zero position.

(2) To set the zero position, need to send command 0x32 and command 0x30 10 times continuous intervals to successfully set it; When the return count value is 10, the trigger sets the zero position.

(4) Digital size, byte order.

(5) CRC byte (CRC polynomial is x8+x7+x4+x2+x1+1,the calculation method is shown in Appendix CRC-8 Table ($X^8+x^7+x^4+x^2+x^1+1$)

Eg:

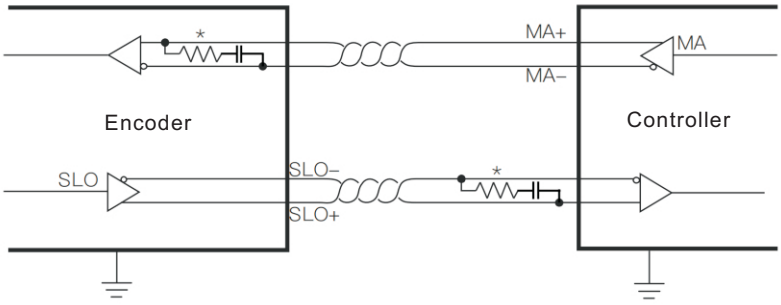
If send 0x33,get uint8_t Buffer[7];

Then when using:

```
uint32_t angleInner = Buffer[2] << 16 | buffer[1] << 8 | buffer[0];
uint32_t anngleOutter = Buffer[5] << 16 | buffer[4] << 8 | buffer[3];
float angleInnerFloat = angleInner / (float)(1 << 24) * 360;
float angleOutterFloat = angleOutter / (float)(1 << 24) *360;
```

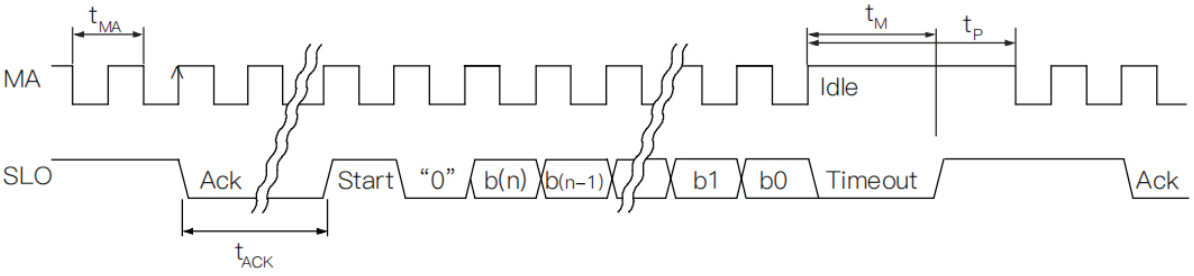
BISS-C protocol interface

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:

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

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:

Bit	B48:b33	B32:b8	B7	B6	b5 : b0
Length	24 bits	X bits	1 bit	1bit	6 bits
Data	Inner circle angle	Outer circle angle	Error bit (1)	Warning bit (2)	CRC (3)

(1) The error bit is valid at low level.

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

(3) The CRC polynomial is x6+x1+1 (i.e. 0x43). According to the BISS-℃ protocol requirements,the calculated CRC will be inverted before being sent. Appendix CRC-6 Calculation provides a directly portable calculation code for easy reference.

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.

Error/warning location in each interface:

	Error bit	Warning bit
BISS-C	b13	b12
RS485	b7	b6

Status bit

Location	b3	b2	b1	b0
Description	Outer rotor too far	Outer rotor too close	Inner rotor too far	Inner rotor too close

In normal condition, the LED status light is green. When the warning bit is 1, the data is stil 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.

Appendix

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

/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-6 calculation

```
#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 ARIM 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 extra splicing 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];
uint8_t placeholder;
uint8_t buffer[8] __attribute__ (aligned(4));
} receiveBuffer;
```

```
//Only align address
//The first placeholder byte of 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

}
```

MOSRAC® Mosrac motor

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