POSITAL
FRABA

IMPLEMENTATION GUIDE
FOR KIT ENCODER

BiSS C over SPI
Type KCD-BC

AMERICA
FRABA Inc.
1800 East State Street, Suite 148
Hamilton, NJ 08609-2020, USA
T +1-609-750-8705, F +1-609-750-8703
www.posital.com, info@posital.com

EUROPE
FRABA GmbH
Zeppelinstraße 2
50667 Cologne, Germany
T +49 221 96213-0, F +49 221 96213-20
www.posital.de, info@posital.de

ASIA
FRABA Pte. Ltd.
114 Lavender Street
#08-60 CT Hub 2, Singapore 338729
T +65 65148880, F +65 62711792
www.posital.com, info@posital.sg

IMPLEMENTATION GUIDE
FOR KIT ENCODER

Introduction

Within this document we explain how easy a realization of reading BiSS C telegrams via a SPI interface can be handled. BiSS C is a powerful interface offering synchronization of reading position values with a high timing accuracy and fast transmission rates. Capture of position value is triggered by reading out the sensor and additional information like diagnostic data or CRC can also be used to ensure a safe transmission.

BiSS C and SSI are very similar in principle, because the kind of data transmission is handled synchronously to a clock signal provided by the PLC or drive controller. At the end a microcontroller or FPGA must read in the information which is quite easy to realize in hardware. All microcontroller on the market offer peripherial units for serial communication and SPI is a de facto standard and free of charge.

Figure 1

In the figure 1 above you see a SPI master configuration providing clock signals to the kit encoder and receiving data from the kit on the MISO pin on master side. Data are received synchronously to the sent clock signals.

IMPLEMENTATION GUIDE
FOR KIT ENCODER

Short Introduction to BiSS C

In the signal timing diagram figure 2 you can see in the beginning of the transmission a start phase containing 2 bits with always high level or logical 1. The position value capture is triggered implicitly with the falling edge of the clock signal. After this start phase we have a time frame called busy time within the kit encoder is preparing the transmission of the position value in its serial interface. When this process is done a signature phase containing 2 bits with logical “1” and “0” is transmitted. This signature is important for the SPI master, because it indicates that the following bits are relevant now like position value, diagnostic bits and CRC. CRC ensures a detection of wrong transmission caused by EMC or other effects.

For simple applications it’s might be sufficient to just read in the position value and abort immediately afterwords the transmission. As you can see the overhead of the BiSS protocol is quite small and very efficient regarding bandwidth.

Be aware that the busy time is NOT constant and varies slightly within 6 µs up to 7 µs. Because of this issue it is strongly recommended to search in the received data stream for the “signature” to detect the start position of position value. Don’t assume a constant number of bits for the transmission. Otherwise you might get sporadic wrong interpretation of data values!

Figure 2

Latch:	First rising edge of clock signal latches encoder position
Busy Time:	Indicated as ACK, With second rising edge of clock signal the encoder sends “0” until data processed internal and ready to send. This takes approximately 5.5 µs. This time is not constant and might vary. It is recommended to add one 1µs for safety reasons. Depending of the used clock frequency (CLK) the number of cycles must be adapted. Example: CLK frequency of 2.6 MHz. Busy time ~ 5.5µs + 1 µs ~ 6.5µs. Number of needed CLK pulses: 14 to 16 CKL pulses.
Signature	consists of 2 bits: Start: Encoder is ready to transmit data indicated by “1” CDS: Is always “0” in unidirectional mode, that means for SSI operation
Position Value	Consists of Multiurn (MT) 16 bit Singletrun (ST) 17 bit
Diagnostics	Error: e.g. HW-failure detected like memory. 0 = failure, 1 = data valid Warning: currently not supported. Constant =“1”. 0 = failure, 1 = good
CRC	Checksum. Inverted. See app note. http://www.biss-interface.com/files/BiSS_AN3_appnote_E1en.pdf
Time Out	Corresponds to tm at SSI, tm=12µs. It is shorter than standard SSI tm = 20 µs, but allows to have interface cycle of 50µs starting with 1MHz CLK

IMPLEMENTATION GUIDE
FOR KIT ENCODER

In the following source example we assumed a read in value via the SPI interface of a microcontroller as a left aligned 64 bit value, which means the first read in bit is on the left side. You can use a 8-bit or any other SPI register length by reading in the bit stream for several times without interruptions longer than the time out value of 12 µs. An exact handling of the SPI register itself is not given here, because each microcontroller handels the register for controlling differently and also the naming.

Following steps should be done:

1. Define for your application the clock frequency. In most cases below 2MHz you don't need to consider latency times caused by line drivers or cables. Allowed clock frequency range is from 100 kHz up to 10 MHz.
2. Decide which informations you really need beside the position value like diagnostics or CRC.
   The whole bit stream can be read in and afterwards relevant data frames can be masked out or you can abort immediately the transmission after reading in position value.

Abstract Code Example

The function **`u64_getBissPositionValue`** gets a raw value as bit stream with maximum 64 bit length and the return value is the right aligned position value consisting of 16 bit for multi-turn and 17 bit for single-turn. If the header of the protocol or signature was not detected correctly, then the return value is `0xDEADBEEFDEADBEEF` to clearly indicate something went wrong and to differentiate easily the error number from a position value.

```
uint64_t u64_getBissPositionValue(uint64_t l_u64RawSPIValue)
{

#define BISS_PROTOCOL_CHECK_MASK 0xE000000000000000LLU //this is the 3 MSB position sampled by the spi interface
#define BISS_LEADING_ONE_VALUE 0xC000000000000000LLU  //this are the 2 ones from start phase and the 0 from busy time at the start of the biss protocol header
#define BISS_START_BIT_VALUE 0x4000000000000000LLU     // "010", last 0 from busy time and 10 from signature
#define MAX_BIT 61 //constant for 64 bit variable
#define BISS_FRABA_POSITAL_ENCODER_NUMBER_OF_BITS 33

uint64_t l_u64TempValue;
uint8_t i=0;
uint8_t l_u8ProtocolIsValid = 0;
uint32_t l_u32StartBitPosition = 0xB16B00B5;

l_u64TempValue = l_u64RawSPIValue;
```

```
//do a protocol check and search for a "110"
do
{
            if( (l_u64TempValue & BISS_PROTOCOL_CHECK_MASK) == BISS_LEADING_ONE_VALUE)
            {
                        l_u8ProtocolIsValid = 1;
                        break;
            }
            l_u64TempValue <<= 1;
            i++;
}while( i != 61 );

//check if protocol was valid
if(l_u8ProtocolIsValid == 0)
{
            return 0xDEADBEEFDEADBEEFLLU;
}

//search for start Bit
do
{
            if( (l_u64TempValue & BISS_PROTOCOL_CHECK_MASK) == BISS_START_BIT_VALUE)
            {
                        l_u32StartBitPosition = i;
                        break;
            }
            l_u64TempValue <<= 1;
            i++;
}while( i != 61 );

if(l_u32StartBitPosition == 0xB16B00B5)
{
            return 0xDEADBEEFDEADBEEFLLU;
}

//now right align the bit stream and mask out everything that is not position value
l_u64RawSPIValue >>= MAX_BIT - i - BISS_FRABA_POSITAL_ENCODER_NUMBER_OF_BITS;
//and mask out the header
l_u64RawSPIValue &= 0x1FFFFFFFFLLU;

return l_u64RawSPIValue;
}
```

POSITAL
FRABA

IMPLEMENTATION GUIDE
FOR KIT ENCODER

Document History

Version	Date	Remark
1	20171218	Initial version.