CRC calculation application note

8-bit CRC calculation with 0x97 polynome

Some of the communication interfaces offer a CRC value to check the correctness of the data read from the encoder. This chapter gives an example of the CRC calculation on the receiver side. The CRC calculation must always be done over the complete set of data. The polynomial for the CRC calculation is P(x) = x8 + x7 + x4 + x2 + x1 + 1, also represented as 0x97.

Code example:

```
//Lookup table for polynome = 0x97
static const u8 ab_CRC8_LUT[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};

/* CRC 0x97 Polynomial, 64-bit input data, right alignment, calculation over 64 bits */

u8 CRC_SPI_97_64bit(u64 dw_InputData)
{
  u8 b_Index = 0;
  u8 b_CRC = 0;
  b_Index = (u8)((dw_InputData >> 56u) & (u64)0x000000FFu);
  b_CRC = (u8)((dw_InputData >> 48u) & (u64)0x000000FFu);
  b_Index = b_CRC ^ ab_CRC8_LUT[b_Index];
  b_CRC = (u8)((dw_InputData >> 40u) & (u64)0x000000FFu);
  b_Index = b_CRC ^ ab_CRC8_LUT[b_Index];
  b_CRC = (u8)((dw_InputData >> 32u) & (u64)0x000000FFu);
  b_Index = b_CRC ^ ab_CRC8_LUT[b_Index];
  b_CRC = (u8)((dw_InputData >> 24u) & (u64)0x000000FFu);
  b_Index = b_CRC ^ ab_CRC8_LUT[b_Index];
  b_CRC = (u8)((dw_InputData >> 16u) & (u64)0x000000FFu);
  b_Index = b_CRC ^ ab_CRC8_LUT[b_Index];
  b_CRC = (u8)((dw_InputData >> 8u) & (u64)0x000000FFu);
  b_Index = b_CRC ^ ab_CRC8_LUT[b_Index];
  b_CRC = (u8)(dw_InputData & (u64)0x000000FFu);
  b_Index = b_CRC ^ ab_CRC8_LUT[b_Index];
  b_CRC = ab_CRC8_LUT[b_Index];

  return b_CRC;
}
```

Example:

```
uint8_t rx_buffer[numOfBytes];   // contains received bytes
// TODO: load rx_buffer array with received data from the encoder
uint64_t dw_CRCinputData = 0;
uint8_t calculated_crc=0;
dw_CRCinputData = ((uint64_t)rx_buffer[0] << 32) + ((uint64_t)rx_buffer[1] << 24) +
                  ((uint64_t)rx_buffer[2] << 16) + ((uint64_t)rx_buffer[3] << 8) +
                  ((uint64_t)rx_buffer[4] << 0);
calculated_crc = ~(CRC_SPI_97_64bit(dw_CRCinputData))& 0xFF;      //inverted CRC
```

6-bit CRC calculation with 0x43 polynome for BiSS

BiSS communication offers a CRC value to check the correctness of the data read from the encoder. This chapter gives an example of the CRC calculation on the receiver side. The CRC calculation must always be done over the complete set of data. The polynomial for the CRC calculation is P(x) = x6 + x1 + 1, also represented as 0x43.
Following code example must be modified to fit actual data length. Position data, error and warning bits must be included into calculation in the same order as in the BiSS data packet. ACK, Start and CDS bits are not included in the CRC calculation.

Code example:

```
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};

/*32-bit input data, right alignment, Calculation over 24 bits (mult. of 6) */
uint8_t CRC_BiSS_43_24bit (uint32_t w_InputData)
{
  uint8_t b_Index = 0;
  uint8_t b_CRC = 0;

  b_Index = (uint8_t )(((uint32_t)w_InputData >> 18u) & 0x0000003Fu);

  b_CRC = (uint8_t )(((uint32_t)w_InputData >> 12u) & 0x0000003Fu);
  b_Index = b_CRC ^ ab_CRC6_LUT[b_Index];

  b_CRC = (uint8_t )(((uint32_t)w_InputData >> 6u) & 0x0000003Fu);
  b_Index = b_CRC ^ ab_CRC6_LUT[b_Index];

  b_CRC = (uint8_t )((uint32_t)w_InputData & 0x0000003Fu);
  b_Index = b_CRC ^ ab_CRC6_LUT[b_Index];

  b_CRC = ab_CRC6_LUT[b_Index];

  return b_CRC;
}

/*64-bit input data, right alignment, Calculation over 42 bits (mult. of 6) */
uint8_t CRC_BiSS_43_42bit(uint64_t dw_InputData)
{
  uint8_t b_Index = 0;
  uint8_t b_CRC = 0;

  b_Index = (uint8_t)((dw_InputData >> 36u) & (uint64_t)0x00000003Fu);

  b_CRC = (uint8_t)((dw_InputData >> 30u) & (uint64_t)0x0000003Fu);
  b_Index = b_CRC ^ ab_CRC6_LUT[b_Index];

  b_CRC = (uint8_t)((dw_InputData >> 24u) & (uint64_t)0x0000003Fu);
  b_Index = b_CRC ^ ab_CRC6_LUT[b_Index];

  b_CRC = (uint8_t)((dw_InputData >> 18u) & (uint64_t)0x0000003Fu);
  b_Index = b_CRC ^ ab_CRC6_LUT[b_Index];

  b_CRC = (uint8_t)((dw_InputData >> 12u) & (uint64_t)0x0000003Fu);
  b_Index = b_CRC ^ ab_CRC6_LUT[b_Index];

  b_CRC = (uint8_t)((dw_InputData >> 6u) & (uint64_t)0x0000003Fu);
  b_Index = b_CRC ^ ab_CRC6_LUT[b_Index];

  b_CRC = (uint8_t)(dw_InputData & (uint64_t)0x0000003Fu);
  b_Index = b_CRC ^ ab_CRC6_LUT[b_Index];

  b_CRC = ab_CRC6_LUT[b_Index];

  return b_CRC;
}
```

Recommended literature:

Painless guide to CRC error detection algorithm; Ross N. Williams.
- Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks; P. Koopman, T. Chakravarty

Head office

RLS merilna tehnika d.o.o.

Poslovna cona Žeje pri Komendi
Pod vrbami 2
SI-1218 Komenda
Slovenia

T +386 1 5272100
F +386 1 5272129
E mail@rls.si

www.rls.si

Global support

Visit our **website** to contact your nearest sales representative.

This product is not designed or intended for use outside the environmental limitations and operating parameters expressly stated on the product's datasheet. Products are not designed or intended for use in medical, military, aerospace, automotive or oil & gas applications or any safety-critical applications where a failure of the product could cause severe environmental or property damage, personal injury or death. Any use in such applications must be specifically agreed to by seller in writing, and is subject to such additional terms as the seller may impose in its sole discretion. Use of products in such applications is at buyer's own risk, and buyer will indemnify and hold harmless seller and its affiliates against any liability, loss, damage or expense arising from such use. Information contained in this datasheet was derived from product testing under controlled laboratory conditions and data reported thereon is subject to the stated tolerances and variations, or if none are stated, then to tolerances and variations consistent with usual trade practices and testing methods. The product's performance outside of laboratory conditions, including when one or more operating parameters is at its maximum range, may not conform to the product's datasheet. Further, information in the product's datasheet does not reflect the performance of the product in any application, end-use or operating environment buyer or its customer may put the product to. Seller and its affiliates make no recommendation, warranty or representation as to the suitability of the product for buyer's application, use, end-product, process or combination with any other product or as to any results buyer or its customer might obtain in their use of the product. Buyer should use its own knowledge, judgment, expertise and testing in selecting the product for buyer's application, end-use and/or operating environment, and should not rely on any oral or written statement, representation, or samples made by seller or its affiliates for any purpose. EXCEPT FOR THE WARRANTIES EXPRESSLY SET FORTH IN THE SELLER'S TERMS AND CONDITIONS OF SALE, SELLER MAKES NO WARRANTY EXPRESS OR IMPLIED WITH RESPECT TO THE PRODUCT, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE, WHICH ARE DISCLAIMED AND EXCLUDED. All sales are subject to seller's exclusive terms and conditions of sale which, where the seller is (a) RLS Merilna tehnika d.o.o., are available at https://www.rls.si/eng/salesterms, (b) Renishaw, Inc., are available at https://www.renishaw.com/legal/en/--42186, or (c) another person, are available on request, and in each case, are incorporated herein by reference, and are the exclusive terms of sale. No other terms and conditions apply. Buyer is not authorized to make any statements or representations that expand upon or extend the environmental limitations and operating parameters of the products, or which imply permitted usage outside of that expressly stated on the datasheet or agreed to in writing by seller.

RLS Merilna tehnika d.o.o. has made considerable effort to ensure the content of this document is correct at the date of publication but makes no warranties or representations regarding the content. RLS Merilna tehnika d.o.o. excludes liability, howsoever arising, for any inaccuracies in this document.

A RENISHAW associate company