BiSS Interface

AN3: CYCLIC REDUNDANCY CODES

RECOMMENDED CRC POLYNOMALS

Polynome			**max. Data Length**	**Hamming Distance**	**Application**
(hexadecimal)	(binary)	(polynomal)			
0x13	0b1.0011	$X^4 + X^1 + X^0$	up to 11 Bit	3	Register Communication
0x25	0b10.0101	$X^5 + X^2 + X^0$	up to 26 Bit	3	Sensor data (SCD)
0x43	0b100.0011	$X^6 + X^1 + X^0$	up to 57 Bit	3	Sensor data (SCD)
0x190D9	0b1.1001.0000 .1101.1001	$X^{16} + X^{15} + X^{12} + X^7 +$ $+ X^6 + X^4 + X^3 + X^0$	up to 64 Bit	6	Sensor data (SCD) (extended safety)

Table 1: *BiSS* Polynomes

CRC IN SCD COMMUNICATION

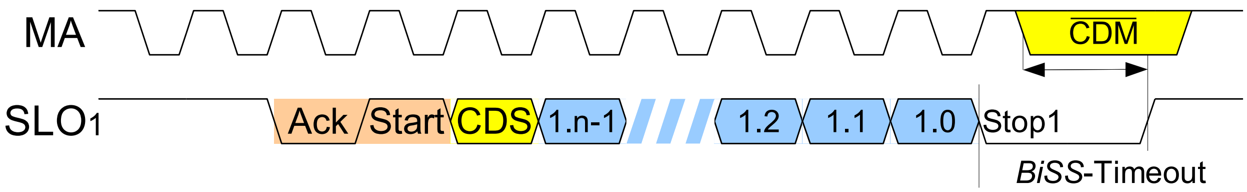

Figure 1: *BiSS* Frame (SCD point-to-point configuration)

The SCD communication is CRC secured. Beyond the CDS bit the transmitted data is secured by CRC:

- CRC start value is typically 0x00
- CRC polynome is typically 0x43
- CRC result length is typically 6 bit
- CRC result is transmitted inverted on the SL line

Copyright © 2012 iC-Haus

CRC IN REGISTER COMMUNICATION

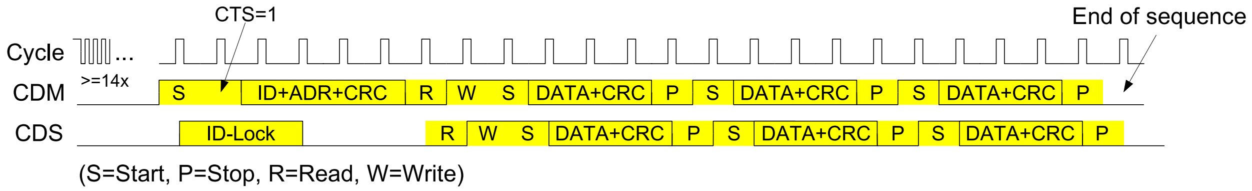

Figure 2: *BiSS* register write access)

The register communication is CRC secured. Beyond the ”S” START bits all the transmitted data is secured by CRC:

- CRC start value is 0x00
- CRC polynome is 0x13
- CRC result length is 4 bit
- CRC result is transmitted inverted in the CDM bit stream
- CDM bit stream is transmitted inverted on the MA line
- CDS bit stream reponse is transmitted non-inverted on the SL line

The master uses for the register addressing CRC calculation:

- CTS
- ID[2:0]
- ADR[6:0]

The master uses for the register data CRC calculation:

- DATA[7:0]

The BiSS slave uses for the register data CRC calculation:

- DATA[7:0]

CRC CODE EXAMPLE: n BIT CRC

This example C file illustrates a possible n bit crc calculation.

```
#include <conio.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*******************************************************
* Function : generates the CRC Code from a given
*            data set + CRC Polynome
*            the length of the code by the size of 1 shorter than the
*            the length of the poynome
*            input strings from LSB to MSB, therefor turn around
*            turn around
*
*******************************************************/
char *pstrCRCGEN (char *pstrDataSetIn, char *pstrCRCPolynomeIn)
{
  int i,j;
  char           *pstrDataSet = strdup(pstrDataSetIn),
                   *pstrCRCPolynome = strdup(pstrCRCPolynomeIn);
  unsigned short usDatLen = strlen(pstrDataSet),
                   usCRCPolyLen = strlen(pstrCRCPolynome),
                   usCRCLen = usCRCPolyLen-1;
  char           *pstrCRC = malloc(usCRCLen);
  char           cExOr = '0';

  pstrDataSet  = strrev(pstrDataSet);
  pstrCRCPolynome = strrev(pstrCRCPolynome);
  for (i = usCRCLen-1; i >= 0; i--)
    pstrCRC[i] = '0';                              // pstrCRC initialize with '0'

  for (i = usDatLen-1; i >= 0; i--)
  {
    switch (pstrDataSet[i])
    {
      case '0':
        if (pstrCRC[usCRCLen-1] == '0')
          cExOr = '0';
        else
          cExOr = '1';
      break;
      case '1':
        if (pstrCRC[usCRCLen-1] == '1')
          cExOr = '0';
        else
          cExOr = '1';
      break;
      default:
      exit(1);
    }                   // end 'switch (pstrDataSet[i])'
    for (j = usCRCLen-1; j > 0; j--)                       // Bits <max>..1
    {
          if (pstrCRCPolynome[j] == '1')
          {
        if (pstrCRC[j-1] == cExOr)
          pstrCRC[j] = '0';
        else
          pstrCRC[j] = '1';
          }
          else
        pstrCRC[j] = pstrCRC[j-1];
        }               // end 'for (j = usCRCLen; j >= 0; j--)'
        pstrCRC[0] = cExOr;   // Bit 0
  }                   // end 'for (i = usDatLen; i >= 0; i--)'
  return (pstrCRC);
}


char *main(int argc, char *argv[])
{
  int  i = 0;
  char *pstrCRCWert = NULL;

  if (argc < 2)
  {
        printf("Call:␣CRCGen␣<data>␣<CRC-Polynome>\n");
        printf("␣␣␣␣␣␣␣␣<data>␣binary␣data,␣the␣crc-code␣should␣be␣created␣for\n");
    printf("␣␣␣␣␣␣␣␣<CRC-Polynome>␣binary␣Polynome,␣used␣to␣create␣the␣CRC\n");
    return (0);
  } /* endif */
  else
  {
    pstrCRCWert = pstrCRCGEN (argv[1], argv[2]);
    printf ("\n\n");
    printf ("data␣␣␣%s\n", argv[1]);
    printf ("CRCPolynome␣␣%s\n", argv[2]);
    printf ("\nergibt␣CRC␣␣");
    for (i = strlen(argv[2])-2; i >= 0; i--)
      printf ("%c", pstrCRCWert[i]);
    printf ("\n\n\n");
  }
  return (pstrCRCWert);
}
```