Cyclic Redundancy Check
Software Modules

ABSTRACT

This routine calculates a cyclic redundancy check with a fast table-driven algorithm.

Contents

1   Introduction

This routine calculates a cyclic redundant check, which is used to detect accidental changes to raw computer data. Once the cyclic redundancy check was calculated the program does a bitwise inversion before sending back the result.

2   Software Interface

The crc_buf method calculates the cyclic redundant check with a series of values versus crc_8 method which calculates the cyclic redundant check for an 8-bit value.

2.1   Source Code

The archive for this software contains all the necessary header files to enable the code and run the functions: crc.c and crc.h

2.2   Software Flow

The crcbuf method loops through the input data and uses a lookup table and the cyclic redundant formula to calculate the 16-bit CRC value.

2.3   Header File – 'crc.h'

```
/*
** Simma Software, Inc.
** https://www.simmasoftware.com
*/
#ifndef __CRC_H__
#define __CRC_H__
#include <stdint.h>

extern uint16_t
crc_8( uint16_t crcvalue, uint8_t val );

extern uint16_t
crc_buf( uint16_t crcvalue, uint8_t *buf, uint16_t size );

#endif
```

2.4   Source Code – 'crc.c'

```c
#include <stdint.h>
#include "crc.h"


/* CRC lookup table. */
uint16_t const static crctable[256] =
{
  0x0000, 0x1189, 0x2312, 0x329b, 0x4624, 0x57ad, 0x6536, 0x74bf,
  0x8c48, 0x9dc1, 0xaf5a, 0xbed3, 0xca6c, 0xdbe5, 0xe97e, 0xf8f7,
  0x1081, 0x0108, 0x3393, 0x221a, 0x56a5, 0x472c, 0x75b7, 0x643e,
  0x9cc9, 0x8d40, 0xbfdb, 0xae52, 0xdaed, 0xcb64, 0xf9ff, 0xe876,
  0x2102, 0x308b, 0x0210, 0x1399, 0x6726, 0x76af, 0x4434, 0x55bd,
  0xad4a, 0xbcc3, 0x8e58, 0x9fd1, 0xeb6e, 0xfae7, 0xc87c, 0xd9f5,
  0x3183, 0x200a, 0x1291, 0x0318, 0x77a7, 0x662e, 0x54b5, 0x453c,
  0xbdcb, 0xac42, 0x9ed9, 0x8f50, 0xfbef, 0xea66, 0xd8fd, 0xc974,
  0x4204, 0x538d, 0x6116, 0x709f, 0x0420, 0x15a9, 0x2732, 0x36bb,
  0xce4c, 0xdfc5, 0xed5e, 0xfcd7, 0x8868, 0x99e1, 0xab7a, 0xbaf3,
  0x5285, 0x430c, 0x7197, 0x601e, 0x14a1, 0x0528, 0x37b3, 0x263a,
  0xdecd, 0xcf44, 0xfddf, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,
  0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,
  0xef4e, 0xfec7, 0xcc5c, 0xddd5, 0xa96a, 0xb8e3, 0x8a78, 0x9bf1,
  0x7387, 0x620e, 0x5095, 0x411c, 0x35a3, 0x242a, 0x16b1, 0x0738,
  0xffcf, 0xee46, 0xdcdd, 0xcd54, 0xb9eb, 0xa862, 0x9af9, 0x8b70,
  0x8408, 0x9581, 0xa71a, 0xb693, 0xc22c, 0xd3a5, 0xe13e, 0xf0b7,
  0x0840, 0x19c9, 0x2b52, 0x3adb, 0x4e64, 0x5fed, 0x6d76, 0x7cff,
  0x9489, 0x8500, 0xb79b, 0xa612, 0xd2ad, 0xc324, 0xf1bf, 0xe036,
  0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,
  0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,
  0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,
  0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,
  0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,
  0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
  0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
  0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
  0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
  0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,
  0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,
  0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,
  0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78
};


/*
** Calculates CRC for a buffer. Fast table-driven algorithm.
*/
uint16_t crc_buf( uint16_t crcvalue, uint8_t *buf, uint16_t size )
{
  while (--size )
    crcvalue = (crcvalue >> 8) ^ crctable[*buf++ ^ (crcvalue & 0xff)];


  return crcvalue;
}
```

```
/*
** Calculates CRC for a byte. Fast table-driven algolithm.
*/
uint16_t crc_8( uint16_t crcvalue, uint8_t buf )
{
    crcvalue = (crcvalue >> 8) ^ crctable[buf ^ (crcvalue & 0xff)];

  return crcvalue;
}
```

3   About Simma Software, the SAE J1939 and UDS Experts

Simma Software, Inc. specializes in real-time embedded software for the automotive industry. Products and services include protocol stacks, bootloaders, device drivers, training, and consultation on the following technologies: J1939, CAN, CAN FD, J1587, J1708, J2497, J1922, J1979, ISO 15765, OBD-II, CANopen, UDS, XCP, NMEA2000, and Secure Boot.