

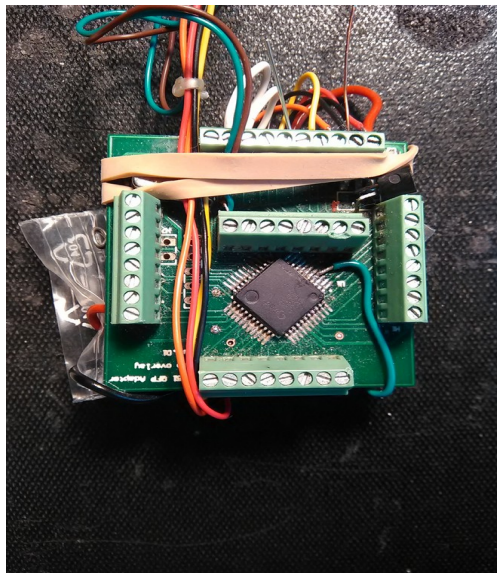
CRC8 & CRC16 Investigation with Delphi and 8051

Introduction

Integrity of long clusters of data cannot be safeguarded by the use of a simple checksum. If an even number of errors were to occur within the bit stream, they could potentially cancel each other, invalidating the checksum. CRC provides a method to detect multiple errors over long bitstreams. Based on AN27 from Dallas Semiconductor, two versions of the CRC polynomials used by Dallas Semiconductor were implemented, one for Delphi/Object Pascal and one for 8051 assembly. Their calculations were then compared to verify that they produce the same results, given the same parameters.

Materials and methods

The standard experimenter's board by DRG Electronics was linked to a PC via a serial TTL to USB cable. The board was populated with a CRD89L51AD1T. The cable was based on an FT232RQ.



The 8051 assembler programme was written with MC Tools 8051 developer IDE version 6.0.6. The Delphi programme was written with the Delphi 7 compiler. SerialTest version 1.01C was used as an intermediary emulating the expected functionality of the two programmes. Information was exchanged between the two programmes based on a simple protocol:

From the PC to the MCU:

'!C' + New data byte in hex + ',' + Previous CRC8 in hex + ',' + Previous CRC16 in hex + CR (character code 13)

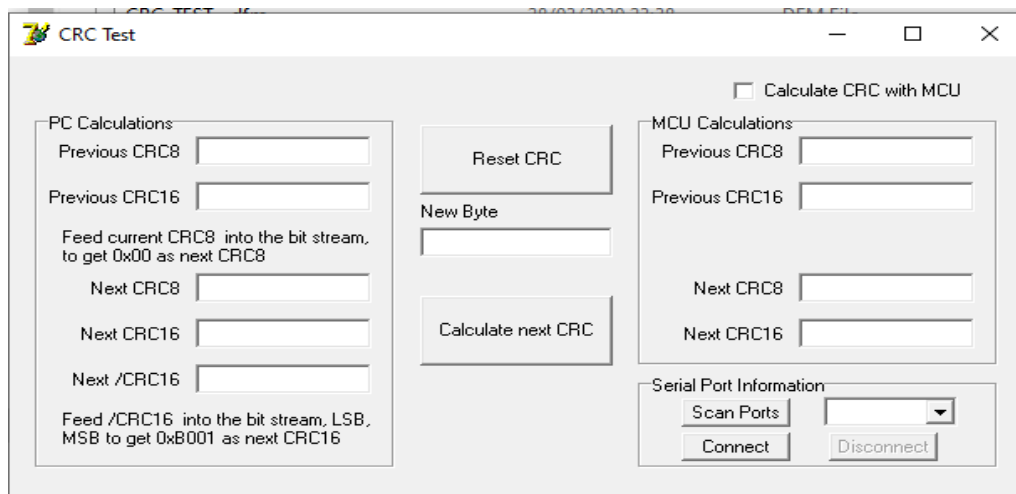
After calculating the new CRC values, the MCU responds to the PC:

'!C' + Previous CRC8 in hex + ',' + Previous CRC16 in hex + ',' + Next CRC8 in hex + ',' + Next CRC16 in hex + CR (character code 13)

The CRC engine used at the 8051 programme can be found in the file: CRC_Utility.a51. The CRC engine used in the Delphi programme can be found in the file: CRC.PAS.

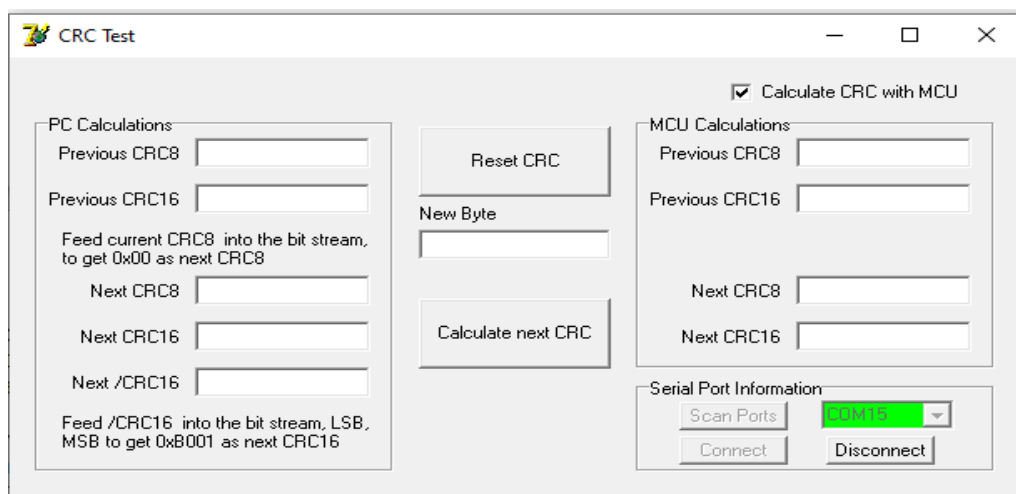
Results

After calling the PC s/w, the user sees:



The screenshot shows the 'CRC Test' application window. At the top right, the checkbox 'Calculate CRC with MCU' is unchecked. The interface is divided into three main sections: 'PC Calculations' on the left, a central control area, and 'MCU Calculations' on the right. The 'PC Calculations' section includes input fields for 'Previous CRC8', 'Previous CRC16', 'Next CRC8', 'Next CRC16', and 'Next /CRC16', along with instructions on how to feed current CRC values into the bit stream. The central area contains buttons for 'Reset CRC', 'New Byte' (with an input field), and 'Calculate next CRC'. The 'MCU Calculations' section has similar input fields for 'Previous', 'Next', and 'Previous /CRC16'. At the bottom right, the 'Serial Port Information' section includes a 'Scan Ports' button, a dropdown menu, and 'Connect' and 'Disconnect' buttons.

Clicking on the **Calculate CRC with MCU** checkbox enables/disables the groupboxes involved with the communication with the MCU. When active, the user may click on the **Scan Ports** button to locate the serial port of the MCU. Available serial ports appear in the selection box at the right hand side of the **Scan Ports** button. Select the required port and then click on the **Connect** button. If the action succeeds to connect to the serial port, the background of the selection box will turn green and the **Disconnect** button will be activated. The **Scan Ports** and **Connect** buttons will be deactivated too.



This screenshot shows the same 'CRC Test' application window, but now the 'Calculate CRC with MCU' checkbox is checked. The layout remains the same, but the 'Serial Port Information' section at the bottom right has changed. The 'Scan Ports' button is now disabled (greyed out), and the dropdown menu next to it has a green background, indicating a successful connection. The 'Connect' button is also disabled, and the 'Disconnect' button is now active (not greyed out).

Clicking on the **Reset CRC** button, zeroes all CRC and New Byte values.

The screenshot shows the 'CRC Test' application window. It has a title bar with a logo and the text 'CRC Test'. The window contains several sections:

- PC Calculations:** Includes input fields for 'Previous CRC8' (00), 'Previous CRC16' (0000), 'Next CRC8' (00), 'Next CRC16' (0000), and 'Next /CRC16' (FFFF). Below these are instructions: 'Feed current CRC8 into the bit stream, to get 0x00 as next CRC8' and 'Feed /CRC16 into the bit stream, LSB, MSB to get 0xB001 as next CRC16'.
- Buttons:** A 'Reset CRC' button is located between the PC and MCU sections. A 'Calculate next CRC' button is below the 'New Byte' field.
- New Byte:** An input field containing the value '00'.
- MCU Calculations:** Includes input fields for 'Previous CRC8' (00), 'Previous CRC16' (0000), 'Next CRC8' (00), and 'Next CRC16' (0000).
- Serial Port Information:** Includes a 'Scan Ports' button, a dropdown menu showing 'COM1', and 'Connect' and 'Disconnect' buttons.

A checkbox labeled 'Calculate CRC with MCU' is checked in the top right corner.

After entering the required **New Byte** to be applied to the bit stream and pressing the **Calculate new CRC** button, the PC side moves the previous CRC values to the “Previous” group of values and calculates the new CRC values which are displayed at the “Next” group of values within the **PC Calculations** groupbox. It also transmits the new byte and the previous CRC values to the MCU. The MCU calculates the next CRC values and transmits them back to the PC, where they are parsed and displayed at the **MCU Calculations** groupbox. For example, applying the byte 0x12 to the zeroed CRC variables, yields:

This screenshot shows the same 'CRC Test' application window after the 'Calculate next CRC' button has been pressed. The values have updated:

- PC Calculations:** 'Next CRC8' is now 21, 'Next CRC16' is 0D80, and 'Next /CRC16' is F27F.
- New Byte:** The input field now contains the value '12'.
- MCU Calculations:** 'Next CRC8' is now 21 and 'Next CRC16' is 0D80, matching the PC side results.

The 'Reset CRC' button is now disabled (greyed out). The 'Calculate next CRC' button remains enabled.

The process can be repeated indefinitely. Reapplying the CRC8, will lead the Next CRC8 value to be zeroed. Applying the inverse of CRC16 to the bit stream, LSB first, MSB second, yields a CRC16 result of 0xB001. Applying the CRC8 to the bit stream, yields a CRC8 result of 0x00.

During the tests the results shown at the MCU Calculations groupbox matched those shown at the PC Calculations groupbox, indicating that both the Pascal and the 8051 engines produced the same results, assuming the same input parameters. The example bitstreams provided at AN27 of Dallas Semiconductor were also applied to both engines and resulted to the same CRC values.

Conclusion

Two versions of the Dallas Semiconductor CRC routines were implemented, one for Delphi/Object Pascal and one for the 8051 assembly language. The two CRC engines were compared by running one on the PC and the other on an 8051 variant. They were provided with the same parameters and they were verified to produce the same results.