Software CRC

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Use lookup table or perform software algorithm to calculate the CRC-32/16/8 result
BSP Version	NUC230/240 Series BSP CMSIS v3.01.002
Hardware	NuTiny–EVB–NUC240–LQFP100 V1.0

1 Function Description

1.1 Introduction

This example code demonstrates how to perform software algorithm to calculate the CRC-16/8 result and use the specify CRC-32/16/8 lookup table to find CRC result more efficiently.

1.2 Principle

A **cyclic redundancy check (CRC)** is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. (Refer to https://en.wikipedia.org/wiki/Cyclic_redundancy_check)

This example achieves how to use lookup table or perform software algorithm to get the CRC result on NUC240 series, but will not mention to the detail principle of CRC algorithm. Open and start Timer0 counting is used to measure the performance while getting the CRC result by lookup table or perform specify algorithm.

Moreover, the calculated CRC result on NUC240 series can be verified with others CRC tools, e.g. Online CRC Calculator on PC (Refer to https://crccalc.com).
Figure 1-1 shows “123456789” CRC-32 result on Online CRC Calculator.

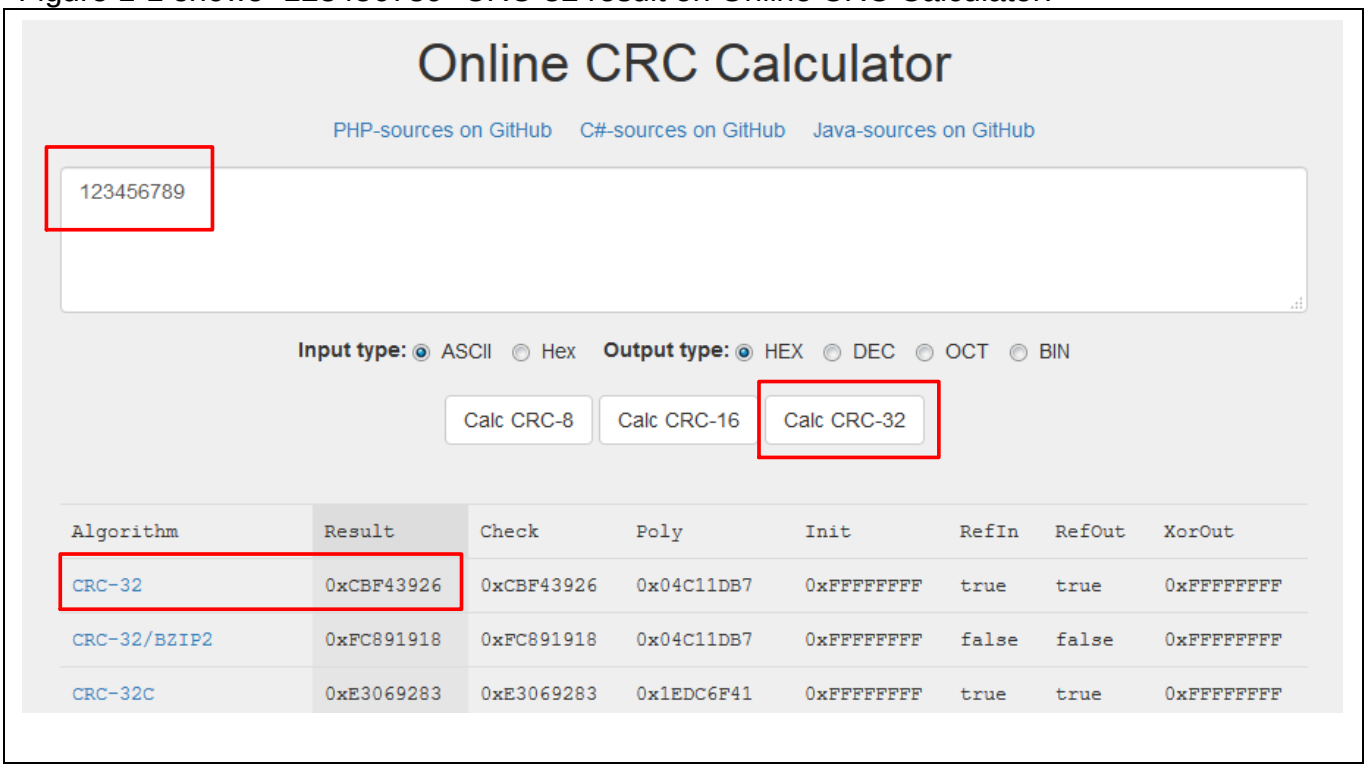

Figure 1-1 CRC-32 Result on Online CRC Calculator

Figure 1-2 shows “123456789” CRC-16/MODBUS result on Online CRC Calculator.

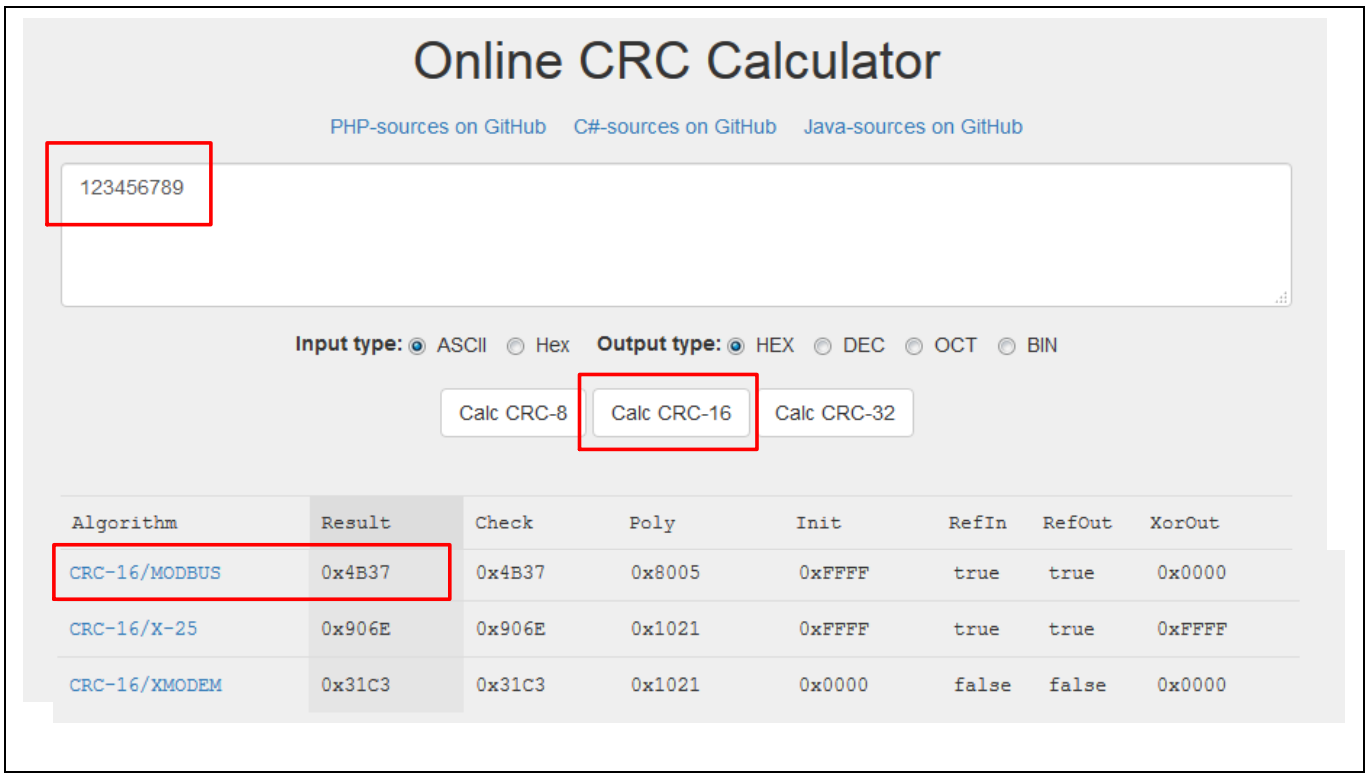

Figure 1-2 CRC-16/MODBUS Result on Online CRC Calculator

Figure 1-3 shows “123456789” CRC-8 result on Online CRC Calculator.

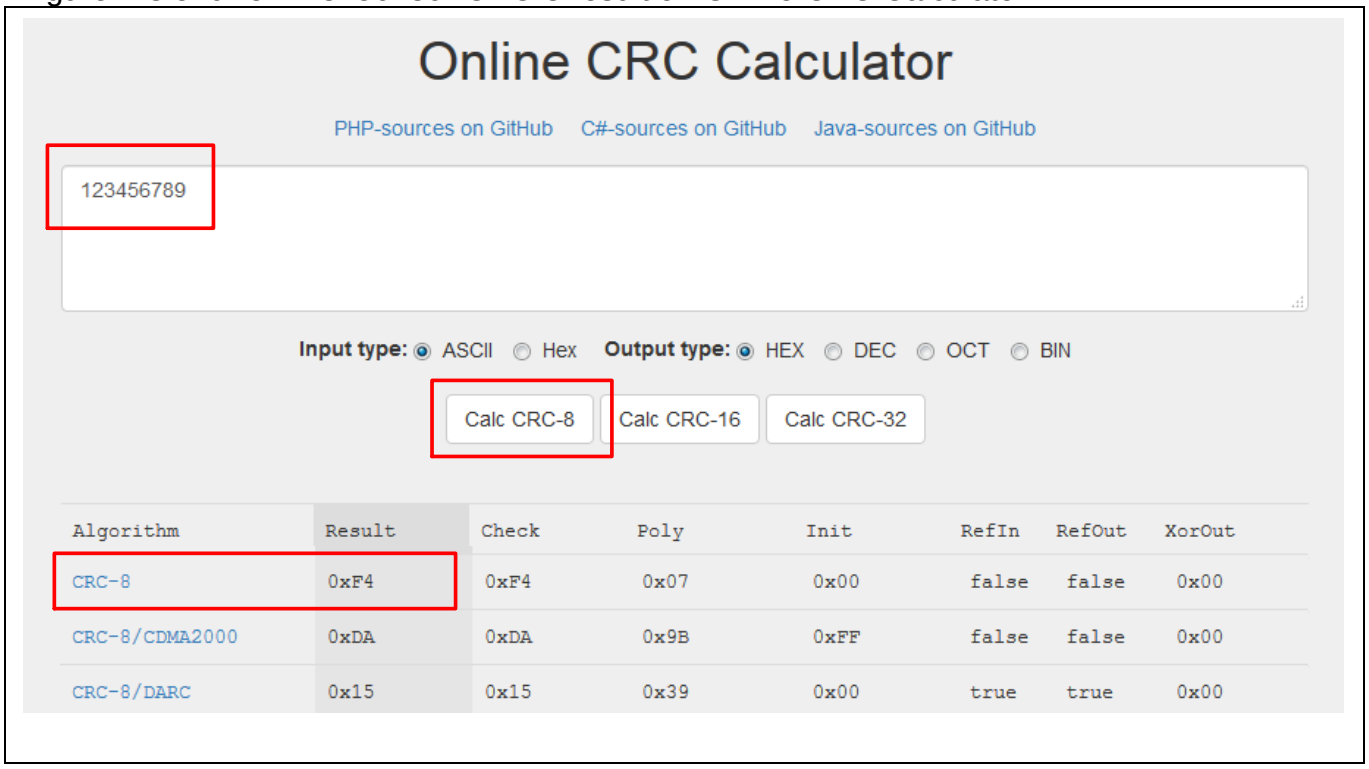

Figure 1-3 CRC-8 Result on Online CRC Calculator

1.3 Demo Result

The execution result can be output on UART, baud rate 115200, as shown in Figure 1-4 on PC.

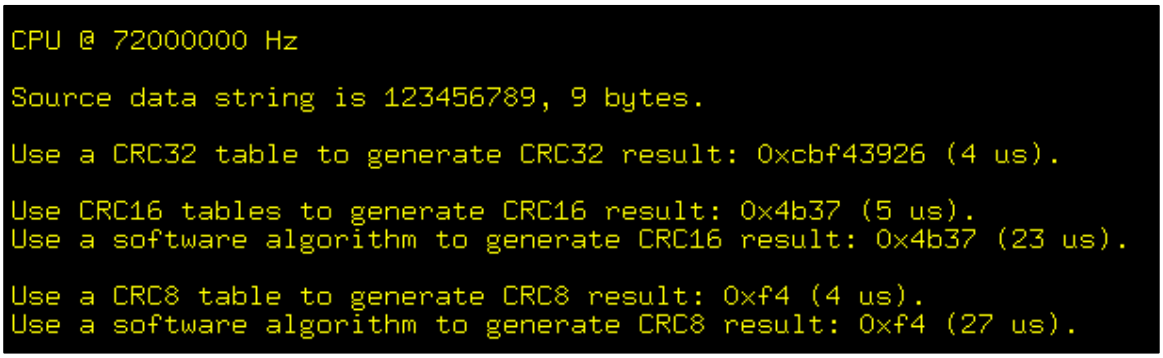

Figure 1-4 Execution Result on UART

2 Code Description

Declare the CRC-32/16/8 lookup table.

```
/*---------------------------------------------------------------------------------------------------
/* The following lookup table is used for finding the CRC-32 result
/*---------------------------------------------------------------------------------------------------
const uint32_t g_au32CRC32tab[] = {……};

/*---------------------------------------------------------------------------------------------------
/* The following lookup table is the Low byte table for finding the CRC-16/MODBUS result
/*---------------------------------------------------------------------------------------------------
const uint8_t g_au32CRC16Ltab[] = {……};

/*---------------------------------------------------------------------------------------------------
/* The following lookup table is used for finding the CRC-8 result
/*---------------------------------------------------------------------------------------------------
const uint8_t g_au8CRC8tab[] = {……};
```

Below functions are used to find the CRC-32/16/8 result.

```
/*---------------------------------------------------------------------------------------------------
/* Use CRC-32 lookup table to find the CRC-32 result
/*---------------------------------------------------------------------------------------------------
uint32_t crc32_by_table(char *data, uint32_t size)
{
    ……
}

/*---------------------------------------------------------------------------------------------------
/* Use CRC-16/MODBUS lookup table to find the CRC-16/MODBUS result
/*---------------------------------------------------------------------------------------------------
uint16_t crc16_by_table(char *data, uint32_t size)
{
    ……
}

/*---------------------------------------------------------------------------------------------------
/* Use CRC-8 lookup table to find the CRC-8 result
/*---------------------------------------------------------------------------------------------------
uint8_t crc8_by_table(char *data, uint32_t size)
```

```
{
    ……
}
```

Below functions are used to generate the CRC-16/8 result by specify software algorithm.

```
/*---------------------------------------------------------------------------------------------
/* Get the CRC-16 result by a specify software algorithm
/*---------------------------------------------------------------------------------------------
uint16_t crc16(char *data, uint32_t size)
{
    ……
}

/*---------------------------------------------------------------------------------------------
/* Get the CRC-8 result by a specify software algorithm
/*----------------------------------------------------------------------------------------------
uint8_t crc8(char *data, uint32_t size)
{
    ……
}
```

Open and start Timer to measure the processing time while getting the CRC result.

```
    /* Enable Timer 0 module clock */
    CLK_EnableModuleClock(TMR0_MODULE);

    /* Select Timer 0 module clock source */
    CLK_SetModuleClock(TMR0_MODULE, CLK_CLKSEL1_TMR0_S_HXT, 0);

    ……

    /* Open and start Timer0 counting in Periodic mode and one tick is 1 us. */
    TIMER0->TCSR = TIMER_PERIODIC_MODE | (12-1);
    TIMER0->TCMPR = 0xFFFFFF;
    TIMER0->TCSR |= TIMER_TCSR_CEN_Msk;
```

The following code shows how to get CRC-16 result by lookup table or software algorithm, and use Timer to measure the efficiency between these two methods.

```
    /* These sections are for software CRC-16 */
```

```
    TIMER0->TCMPR = 0xFFFFFF; // Reload TCMPR to restart Timer0 counting from TDR = 0
    u32CRCResult = crc16_by_table(str, u32StrLen);
    u32CRC16Time = TIMER0->TDR;
    printf("Use CRC-16 tables to generate CRC-16 result: 0x%x (%d us).\n", u32CRCResult,
u32CRC16Time);
    TIMER0->TCMPR = 0xFFFFFF; // Reload TCMPR to restart Timer0 counting from TDR = 0
    u32CRCResult = crc16(str, u32StrLen);
    u32CRC16Time = TIMER0->TDR;
    printf("Use a software algorithm to generate CRC-16 result: 0x%x (%d us).\n",
u32CRCResult, u32CRC16Time);
    printf("\n");
```

3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230/240 Series BSP CMSIS v3.01.002
- IDE version
  - Keil uVersion 5.28

Hardware Environment

- Circuit component
  - NuTiny–EVB–NUC240–LQFP100 V1.0
- Diagram
  - Connect UART TX (PB.1) pin to PC UART RX for display the execution result of example code on PC.

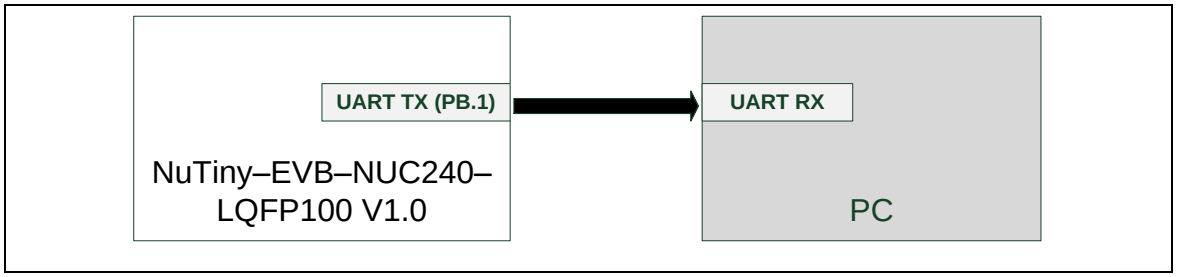

Figure 3-1 NUC240 UART TX and PC UART RX Connection

4 Directory Information

- EC_NUC240_Software_CRC_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click Software_CRC.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Jul. 23, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.