32-bit microcontroller

HC32L110 / HC32F003 / HC32F005

CRC of series

Suitable

series	Product number
HC32L110	HC32L110C6UA HC32L110C6PA HC32L110C4UA HC32L110C4PA HC32L110B6PA HC32L110B4PA
HC32F003	HC32F003C4UA HC32F003C4PA
HC32F005	HC32F005C6UA HC32F005C6PA HC32F005D6UA

content

1 Summary

This application note mainly introduces how to use the CRC function of the HC32L110 / HC32F003 / HC32F005 series for encoding and verification.

2 Introduction to CRC

What is **CRC?**

CRC stands for Cyclic Redundancy Check (Cyclic Redundancy Check): It is the most commonly used error check in the field of data communication. It is characterized in that the length of the information field and the check field can be arbitrarily selected. Cyclic Redundancy Check (CRC) is a data The transmission error detection function performs polynomial calculation on the data and attaches the obtained result to the back of the frame. The receiving device also performs a similar algorithm to ensure the correctness and integrity of data transmission.

(Quoted from 'Baidu Encyclopedia', 'Interactive Encyclopedia', 'Wikipedia')

CRC Fundamentals?

The basic principle of the Cyclic Redundancy Check Code (CRC) is: after the K-bit information code is spliced with the R-bit check code, the entire code length is The degree is N bits, so this encoding is also called (N, K) code. For a given (N, K) code, it can be shown that there is a A polynomial G(x) whose highest power is NK=R. A check code of K bits of information can be generated according to G(x), and G(x) is called the generator polynomial of this CRC code. The specific generation process of the check code is as follows: Assuming that the information to be sent is represented by a polynomial C(X), the C(x) is shifted to the left by R bits (which can be expressed as C(x)*2R), so that the right side of C(x) will leave R bits, which is the position of the check code. The remainder obtained by dividing C(x)*2R by the generator polynomial G(x) is the check code.

Any code consisting of a string of binary bits can correspond one-to-one with a polynomial whose coefficients are only '0' and '1'. example For example: the polynomial corresponding to the code 1010111 is x6+x4+x2+x+1, and the polynomial is the code corresponding to x5+x3+x2+x+1 101111ÿ

Application of **CRC ?**

CRC verification utility library In the field of data storage and data communication, in order to ensure the correctness of the data, error detection has to be used. s method. Among many error detection methods, CRC is the most famous one. The full name of CRC is Cyclic Redundancy Check, which is characterized by: The error detection ability is very strong, the overhead is small, and it is easy to be realized by the encoder and the detection circuit. From its error detection ability, it cannot detect The chance of error is just under 0.0047%. In terms of performance and overhead, it is far superior to parity check and arithmetic sum check, etc. Way. Therefore, in the field of data storage and data communication, CRC is ubiquitous: FCS (Frame Check) of the well-known communication protocol X.25

error sequence) using CRC-CCITT, WinRAR, NERO, ARJ, LHA and other compression tool software using CRC32, the read and write of the disk drive adopts CRC16, and the common image storage formats GIF, TIFF, etc. also use CRC as the for error detection.

3 CRC

3.1 Introduction

The CRC module algorithm of BGI HC32L110 / HC32F003 / HC32F005 series MCU complies with ISO/IEC13239

Definition, using CRC of 16-bit length, the calculation polynomial is $x^{16} + x^{12} + x^{5} + 1$, 16-bit initial value is "FFFF".

The functions of this module include:

- Provide CRC code generation and CRC code check

- Provides 8/16/32 bit wide little-endian access

Take the calculation of CRC for the data string 00112233445566778aabbcddeeff as an example:

- 8-bit mode input sequence: 00, 11, 22 ....

- 16-bit mode input sequence: 0x1100 0x3322ÿ0x5544ÿ0x7766

- 32-bit mode input sequence: 0x33221100, 0x77665544

3.2 Description

This subsection introduces the CRC module of the HC32L110 / HC32F003 / HC32F005 family, including registers and workflow.

3.2.1 Register introduction

The operation of the CRC module is mainly carried out through the following registers:

abbreviation	Register Name
CRC_RESULT	CRC result register. After the calculation is completed, read this register to get the result.
CRC_DATA	CRC data register, used to input the data to be calculated

3.2.2 Workflow Introduction

CRC encoding

CRC code generation is to operate on a string of data to generate a 16-bit CRC code result. The operation process is as follows:

1) Set the CRC_RESULT register to the initial value of 16'hFFFF. This can be done by directly writing 16'hFFFF to
CRC_RESULT[15:0] register to achieve, can also be achieved by reset.

2) Write the data to be calculated into the CRC_DATA register in turn, and each write operation corresponds to 1 data input (8 bits, 16 bits)
bit or 32 bits). For example, if there are 10 data, write 10 times to the CRC data register, one at a time.
The address of the CRC_DATA register is a range (0x80-0xFF). If 32-bit data transmission mode is selected, the
It is recommended to use STM instructions to speed up data input by selecting an address of this register (such as 8'h80), and then
Send several pieces of data to the CRC_DATA data register at a time through the STM instruction (note the change of the address in the STM instruction
change, do not exceed the address range specified by the CRC_DATA data register).

3) After writing all the data to be operated into the CRC_DATA result register, read the CRC_RESULT result register
[15:0] of 16-bit CRC code can be obtained.

CRC check

CRC check is to judge a string of data and 16-bit CRC code to check whether it is correct. CRC check operation
The operation is very similar to that generated by the previous CRC encoding. Just add the following steps after the third step:

1-3: Same as steps 1-3 in CRC encoding.

4: After writing all the data to be operated into the CRC_DATA result register, then write the 16-bit CRC code.

5: Read the CRC_RESULT result register [16], if it is 1, it means the check is successful; if it is 0, it means that the check fails.
defeat.

4 Sample code

4.1 Code introduction

Users can write their own code to learn and verify the module according to the above workflow, or directly through Huada Semiconductor

The CRC sample code downloaded from the website directly uses the API functions provided by the CRC driver library for encoding and verification applications.

The following sections briefly describe the functionality of the various parts of the code:

1) CRC data declaration and initialization:

```
//CRC TEST DATA INIT
    uint8_t au8CrcTestData[8] = {0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0}; uint32_t
    u32TestDataLen = 8; uint16_t u16RefCrc16 = 0x5234;
```

2) CRC encoding:

```
en_result_t CrcCodingTest(void)
{ uint16_t u16CrcResult = 0;
    en_result_t enResult = Error;


    u16CrcResult = CRC16_Get8(au8CrcTestData, u32TestDataLen);
    if(u16RefCrc16 == u16CrcResult) { enResult = Ok;


    }

    return enResult; }
```


3) CRC check:

```
en_result_t CrcCheckTest(void)
{ return
    CRC16_Check8(au8CrcTestData, u32TestDataLen, u16RefCrc16); }
```

4) CRC encoding and verification:

```
if(Ok != CrcCodingTest())
{ u8TestFlag |= 0x01; } if(Ok !=
    CrcCheckTest()) { u8TestFlag |=
0x02 ;
```

4.2 Code running

Users can download the CRC sample code through the website of Huada Semiconductor, and run the relevant code with the evaluation board to learn how to use it.

Use the CRC module.

The following sections describe how to run the CRC sample code on the evaluation board and observe the results:

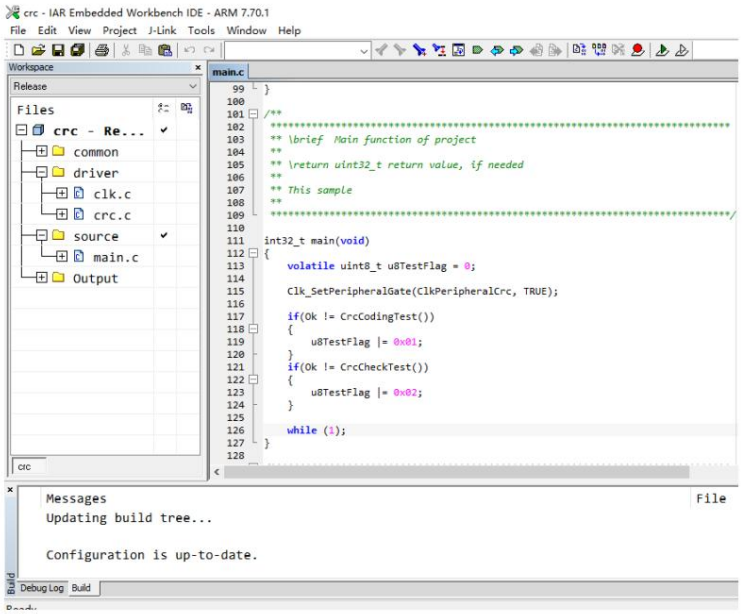
- Make sure to install the correct IAR (or Keil, here IAR is used as a sample description, the operation method is similar in Keil) tool (please

  Completely download the corresponding installation package from Huada Semiconductor, and refer to the user manual for installation).

- Download the CRC sample code from the Huada Semiconductor website.

- Download and run the sample code:

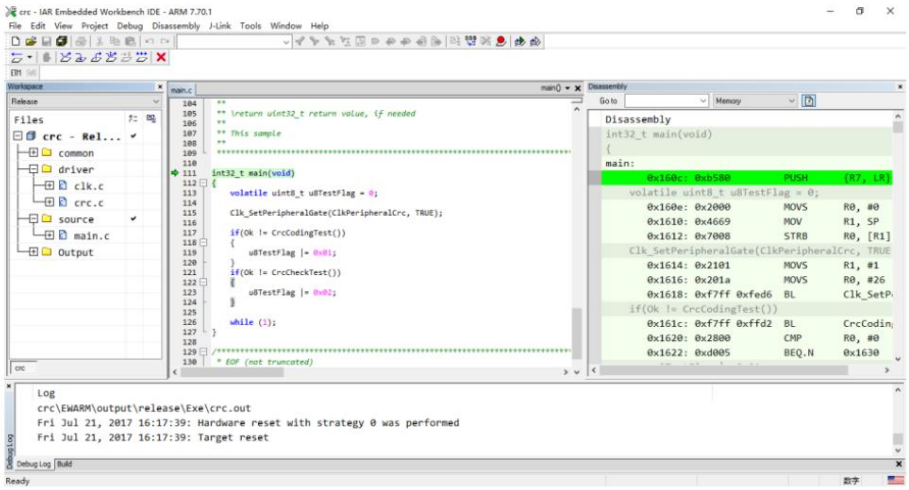
  1) Open the CRC project, and open the 'main.c' view as follows:



  2) Click  Recompile to link the entire project.

  3) Click  Download the code to the evaluation board.

4) You can see a view similar to the following:



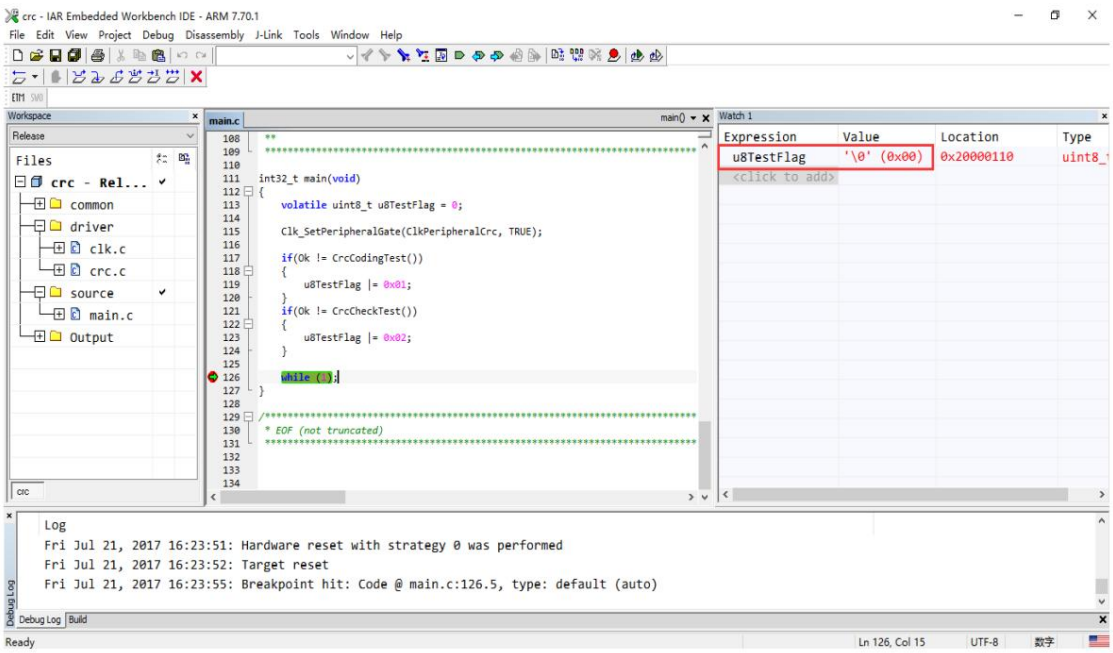
5) Set a breakpoint on the line after 'main(void)':

6) Click "View -> Watch -> Watch1" to open a 'watch1' window, and add the 'u8TestFlag' variable to observe its value.

7) Click Run: 

8) The code runs and stops at the breakpoint of 'main(void)', if 'u8TestFlag = 0', it means encoding and verification functions

Correct execution, as shown in the figure below: (The test data in this example are all from the calculation results of standard CRC software encoding tools).



9) You can close the project file after running.

10) The user can also further learn the function of the CRC module by modifying the CRC test data (data and length) in the code.

can.

5 Summary

The above section briefly introduces the CRC of the HC32L110 / HC32F003 / HC32F005 series, and details the CRC module

The register and operation flow of the block demonstrate how to use the relevant sample code for coding and verification. In actual development, the user

You can configure and use the CRC encoding and verification functions according to your own needs.

6 Version Information & Contact Information

date	Version revision record	
2018/6/4	Rev1.0 initial release	

If you have any comments or suggestions in the process of purchasing and using, please feel free to contact us.

Email:mcu@hdsc.com.cn

Website: www.hdsc.com.cn

Mailing address: No. 39, Lane 572, Bibo Road, Zhangjiang Hi-Tech Park, Shanghai

Postcode: 201203