Application Note

Application Note

Document No.: AN1101

APM32F4xx_CRC Application Note

Version: V1.0

1 Introduction

This application note provides a guide on how to configure and apply CRC peripheral on APM32F4xx series and introduces the CRC parameter model and CRC algorithm.

Contents

2 CRC Introduction

The full name of CRC is Cyclic Redundancy Check. This unit can get 32-bit CRC computing result by calculating the input data through a fixed polynomial calculation, and is mainly used to detect or verify the correctness and integrity of the data after transmission or saving.

2.1 CRC parameter model

Usually, a parameter model needs to be known in order to calculate the accurate CRC value. A complete CRC parameter model should include the following information: POLY, WIDTH, INIT, REFIN, REFOUT and XOROUT. Usually, if only one polynomial is provided without specifying other parameters, the initialization value INIT is 0x00, the input inversion REFIN is false, the output inversion REFOUT is false, and the XOR output XOROUT is 0x00. After inputting the number that needs to be checked, calculate it using the CRC algorithm according to the parameter model and the check value can be obtained. Where:

1) POLY: Hexadecimal polynomial, but the most significant bit 1 is omitted. For example, x8+x4+x3+x+1, the binary is 1 0001 1011, the most significant bit 1 is omitted, and it is converted to the hexadecimal 0x1B.

2) WIDTH: The generated CRC data bit width; for example, the generated CRC of CRC-16 is 16 bits.

3) Name: The name of parameter model.

4) INIT: represents the initial value of CRC, which remains the same as the WIDTH bit width.

5) REFIN: Its value is either false or true. False means that the original data does not need to be flipped before calculation; while true means that the original data needs to be flipped. Take the original data: 0x5D=0101 1101 as an example. If REFIN is true, it will be 1011 1010=0xBA after flipping.

6) REFOUT: true or false, which is a parameter that specifies whether to flip the CRC value obtained after the operation is completed.

7) XOROUT: Perform XOR operation on the calculation result to obtain the final CRC value, ensuring that its bit width is the same as WIDTH.

2.1.1 Common CRC parameter models

The commonly used parameter models of CRC and the application scenarios are shown in Table 1.

Table 1 Introduction to Commonly Used CRC

Name of CRC algorithm	Polynomial formula/hexadecimal polynomial	Hexadecimal polynomial	Input value inversion	Output value inversion
CRC-5/ITU	X^5+X^3+X+1	15	true	true

Name of CRC algorithm	Polynomial formula/hexadecimal polynomial	Hexadecimal polynomial	Input value inversion	Output value inversion
CRC-5/ITU	X^5+X^3+X+1	15	true	true
CRC-5/USB	X^5+X^2+1	05	true	true
CRC-7/MMC	X^7+X^3+1	09	false	false
CRC-8/ITU	$X^8+X^7+X^3+X^2+1$	07	false	false
CRC-16/USB	$X^{16}+X^{12}+X^5+1$	8005	true	True
CRC-16/MODBUS	$X^{16}+X^{12}+X^5+1$	8005	true	true
CRC-32/ MPEG-2	$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$	04C11DB7	false	false
CRC-32	$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$	04C11DB7	true	true

2.2 CRC algorithm

The concept of CRC is to add a check code after the data to be transmitted to generate a new transmission frame and transmit it to the receiving end. Assuming that the data to be transmitted is K bits and the check code added at the end is R bits, the generated new frame is K+R bits. After receiving the data at the receiving end, verify whether the data transmission is correct by comparing with the check code. This comparison method requires that the receiving end and the transmitting end determine a common divisor. Before transmitting the data frame, perform remainder processing by adding a number. When the data transmission is correct, the CRC calculation result should have no remainder; otherwise, the calculation result has a remainder, which indicates that an error occurs in the transmission process.

The specific procedure is as follows:

1) Before communication, the transmitting end and the receiving end shall first agree on the value of the polynomial, the divider P. The number of digits of the divisor P should be the number of digits of check code plus 1.

2) The transmitting end shifts the original data left for R bits by adding R zeros after the original K-bit data.

3) Perform module-2 division operation. Divide the data with a length of K+R by P to make loop calculation, and stop the calculation when the order of the remainder is less than R. The obtained remainder is an additional check code. When the length is less than R bit, zeros shall be supplemented in front.

4) The transmitting end appends the check code of R bit to the original data, and transmits the entire data with check code to the receiver.

5) After receiving the data, the receiver uses the divisor P to perform division operation on the data using modulo-2 division. When there is a remainder, it proves that an error occurs in the transmission process; on the contrary, the transmission is normal.

Below is an application example of CRC-8 polynomial 0x07.

1) For CRC-8 polynomial 0x07, the source data is 10101110; according to the agreed polynomial, we initialize the CRC check value to 0x00.

2) Add 8 zeros to the end of the data, namely, shift the data left by 8 bits, and obtain: 1010111000000000

3) Take the front 8-bit data (10101110), perform XOR operation with polynomial 0x07 and obtain 11001010.

4) Shift the result left by 1 bit, discard the most significant bit, and obtain 10010100. Repeat the above steps, take 8-bit data from the results in sequence to perform XOR operation with 0x07, and then shift left until the remaining data bits are less than 8. The final remainder is 00101011. This 00101011 is the calculated CRC check bit.

5) Append it to the end of the original data to get the transmitted data: 1010111000101011. After receiving the data, the receiving end also performs CRC check.

6) Perform module-2 division operation on the received data according to the same process until the final obtained remainder is 0, indicating that no error occurred. If the remainder is not 0, it indicates an error.

3 Introduction to APM32F4XX CRC

3.1 APM32F4XX CRC parameter model

The built-in hardware CRC calculation module of APM32 used is CRC-32

model, and its polynomial is 0x4C11DB7——X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X4 + X2 + X +1.
According to the model parameter analysis *mentioned in 错误!未找到引用源。 above*, the data bit width of APM32 CRC is 32 bits, the hexadecimal polynomial is 0x4C11DB7, INIT=0xFFFFFFFF, REFIN=false, REFOUT=false, XOROUT=0x00000000.

3.2 Hardware implementation advantages

The built-in hardware CRC calculation module of APM32 brings the following advantages when implementing CRC using hardware:

1) High-speed computing: For the hardware implementation of CRC, the special circuits and parallel computing can be used to achieve fast CRC computing. Compared with software implementation, hardware can complete CRC computing within a few clock cycles, to provide higher processing speed.

2) Low power consumption: Due to the use of special circuits to perform CRC computing, hardware implementation can reduce power consumption through circuit design optimization. Compared with using loops for CRC computing in software, hardware implementation can save a lot of energy.

3) Occupation of few resources: Software implementation of CRC may require large memory and processor resources to perform complex calculation. However, hardware implementation usually only requires some logic gates and registers, occupying few resources, and is particularly suitable for embedded systems with limited resources.

4) Real-time performance: The CRC implemented by hardware can perform continuous computation in real-time data stream without delay or interrupt. This is very important for applications that require fast processing of data streams, such as real-time transmission and communication protocols.

5) Strong anti-interference capability of hardware: CRC implemented by hardware usually can better resist noise and interference, which is because it can use differential signals and physical layer technologies to improve the anti-interference capability. This is very important for the reliability and error detection of data transmission.

In summary, the hardware implementation of CRC has such advantages as high-speed computing, low power consumption, occupation of less resources, good real-time performance, and strong anti-interference capability. So the hardware implementation has more advantages in applications that require high efficiency, reliability, and real-time performance.

4 APM32F4XX CRC Application Routine

4.1 Software design process

The standard check value has been calculated based on data in the APM32F4XX routine. The webpage calculation process is shown in Figure 3, and the calculated standard check value is 0x379E9F06. The data string to be checked is stored in an array for CRC check. When the output check value is equal to the standard check value, it proves that the transmission is normal; on the contrary, if they are different, it proves that the data transmission is wrong.

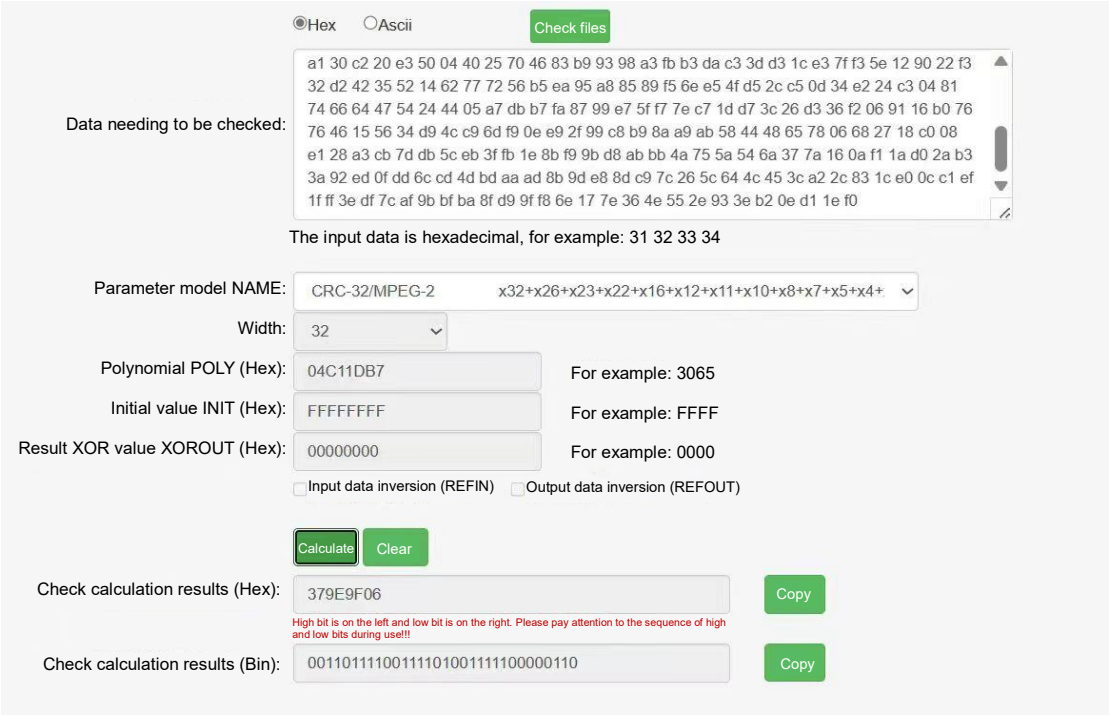

Figure 3 Calculation of Standard Check Value

The CRC computing unit contains a 32-bit data register (CRC_DATA) for data input and storing CRC calculation results. The calculation time is four AHB clock cycles. Every time a new data is written, the result will be a combination of the last calculation result and the new calculation result (operation is performed on the entire word). Write operation of CPU will be suspended during calculation, so that "back-to-back" write or continuous "read-write" operation can be performed on the register CRC_DATA. To calculate the CRC of supporting data, operate according to the following steps:

1) Enable CRC peripheral clock through RCC peripheral.

2) Configure the initial CRC value register (CRC_DATA) to set the CRC data register to the initial

CRC value.

3) Reset the CRC peripheral through the Reset bit in the CRC control register (CRC_CTRL).

4) Set the data to the CRC data register.

5) Read the content of the CRC data register and it is the calculated CRC value.

The software programming flow chart is as follows:

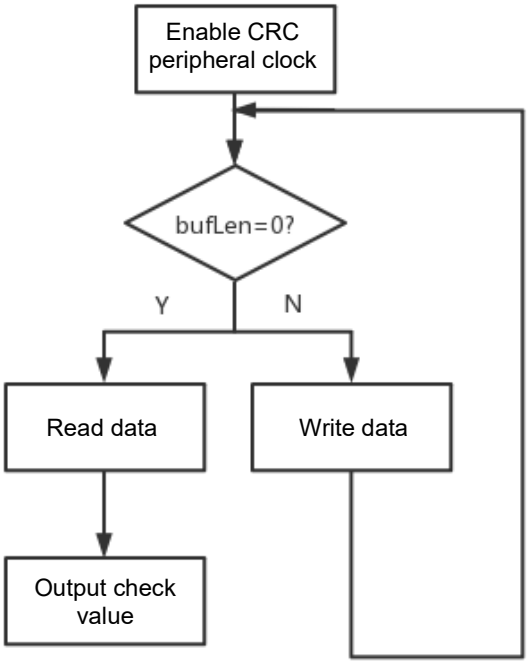

Figure 4 CRC Software Programming Flow Chart

4.2 Software implementation

4.2.1 Definitions of CRC parameters

Define the array to store the data to be checked, and define the 32-bit integer variable to store the CRC output check value; complete the serial port initialization and related parameter setting: set the baud rate to 115200, the check bit to 0, and the stop bit to 1. The related codes are as follows:

```
static const uint32_t aDataBuffer[BUFFER_SIZE] =
  {
      0x00001021, 0x20423063, 0x408450a5, 0x60c670e7, 0x9129a14a,
      0xb16bc18c, 0xd1ade1ce, 0xf1ef1231, 0x32732252, 0x52b54294,
      0x72f762d6, 0x93398318, 0xa35ad3bd, 0xc39cf3ff, 0xe3de2462, 0x34430420,
      0x64e674c7, 0x44a45485, 0xa56ab54b, 0x85289509, 0xf5cfc5ac,
      0xd58d3653, 0x26721611, 0x063076d7, 0x569546b4, 0xb75ba77a,
      0x97198738, 0xf7dfe7fe, 0xc7bc48c4, 0x58e56886, 0x78a70840,
      0x18612802, 0xc9ccd9ed, 0xe98ef9af, 0x89489969, 0xa90ab92b,
      0x4ad47ab7, 0x6a961a71, 0x0a503a33, 0x2a12dbfd, 0xfbbfeb9e,
      0x9b798b58, 0xbb3bab1a, 0x6ca67c87, 0x5cc52c22, 0x3c030c60,
      0x1c41edae, 0xfd8fcdec, 0xad2abd0b, 0x8d689d49, 0x7e976eb6,
      0x5ed54ef4, 0x2e321e51, 0x0e70ff9f, 0xefbedfdd, 0xcffcbf1b, 0x9f598f78,
      0x918881a9, 0xb1caa1eb, 0xd10cc12d, 0xe16f1080, 0x00a130c2,
      0x20e35004, 0x40257046, 0x83b99398, 0xa3fbb3da, 0xc33dd31c,
      0xe37ff35e, 0x129022f3, 0x32d24235, 0x52146277, 0x7256b5ea,
      0x95a88589, 0xf56ee54f, 0xd52cc50d, 0x34e224c3, 0x04817466,
      0x64475424, 0x4405a7db, 0xb7fa8799, 0xe75ff77e, 0xc71dd73c,
      0x26d336f2, 0x069116b0, 0x76764615, 0x5634d94c, 0xc96df90e,
      0xe92f99c8, 0xb98aa9ab, 0x58444865, 0x78066827, 0x18c008e1,
      0x28a3cb7d, 0xdb5ceb3f, 0xfb1e8bf9, 0x9bd8abbb, 0x4a755a54,
      0x6a377a16, 0x0af11ad0, 0x2ab33a92, 0xed0fdd6c, 0xcd4dbdaa,
      0xad8b9de8, 0x8dc97c26, 0x5c644c45, 0x3ca22c83, 0x1ce00cc1, 0xef1fff3e,
      0xdf7caf9b, 0xbfba8fd9, 0x9ff86e17, 0x7e364e55, 0x2e933eb2, 0x0ed11ef0
  };
uint32_t uCRCValue = 0;

USART_Config_T usartConfigStruct; /* USART configuration */
USART_ConfigStructInit(&usartConfigStruct); usartConfigStruct.baudRate =
115200;
usartConfigStruct.mode = USART_MODE_TX_RX; usartConfigStruct.parity =
USART_PARITY_NONE; usartConfigStruct.stopBits = USART_STOP_BIT_1;
usartConfigStruct.wordLength = USART_WORD_LEN_8B;
usartConfigStruct.hardwareFlow = USART_HARDWARE_FLOW_NONE;
```

4.2.2 Enable CRC peripheral clock

Before starting to use CRC, the CRC peripheral clock shall be enabled through

```
RCM_EnableAHB1PeriphClock(RCM_AHB1_PERIPH_CRC);
```

the RCM peripheral.

4.2.3 Configure initial CRC value

Set the RST bit of CRC_CTRL register to 1, configure the initial CRC value register (CRC_DATA), and set the CRC data register to the initial CRC value

```
CRC_ResetDATA();
```

0xFFFFFFF.

4.2.4 CRC data check

After the clock is enabled and the initial value is configured, call the CRC calculation function, and use the while loop to write data to the data register in 32 bits. When BUFFER_SIZE is equal to zero, it means that the data has been written; read the value of the data register and obtain the output check value. The standard check value and output check value can be observed through the serial assistant. When the output check value is equal to the standard check value, it proves that there is no error in data transmission.

The 32-bit data, as an input register, is new data stored in the CRC calculator during writing. As an output register, it returns the results of CRC computing when reading.

```
CRC_ResetDATA();
uCRCValue=CRC_CalculateBlockCRC((uint32_t*)aDataBuffer,BUFFER_SIZE);
printf("BlockCRC = 0x379E9F06 \r\n");
printf("CalculateBlockCRC = 0x%08X \r\n", uCRCValue);
```

The function code of CRC_CalcBlockCRC is as follows:

```
uint32_t CRC_CalculateBlockCRC(uint32_t *buf, uint32_t bufLen)
{
  while(bufLen--)
  {
    CRC->DATA = *buf++;
  }
  return (CRC->DATA);
}
```

5 Revision history

Table 2 Document Version History

Date	Version	Revision History
July 5, 2023	1.0	First draft

Statement

This manual is formulated and published by Zhuhai Geehy Semiconductor Co., Ltd. (hereinafter referred to as "Geehy"). The contents in this manual are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to correct and modify this manual at any time. Please read this manual carefully before using the product. Once you use the product, it means that you (hereinafter referred to as the "users") have known and accepted all the contents of this manual. Users shall use the product in accordance with relevant laws and regulations and the requirements of this manual.

1. Ownership of rights

This manual can only be used in combination with chip products and software products of corresponding models provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this manual for any reason or in any form.

The "Geehy" or "Geehy" words or graphics with "®" or "TM" in this manual are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No intellectual property license

Geehy owns all rights, ownership and intellectual property rights involved in this manual.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale and distribution of Geehy products and this manual.

If any third party's products, services or intellectual property are involved in this manual, it shall not be deemed that Geehy authorizes users to use the aforesaid third party's products, services or intellectual property, unless otherwise agreed in sales order or sales contract of Geehy.

3. Version update

Users can obtain the latest manual of the corresponding products when ordering Geehy products.

If the contents in this manual are inconsistent with Geehy products, the agreement in Geehy sales order or sales contract shall prevail.

4. Information reliability

The relevant data in this manual are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this manual. The relevant data in this manual are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance. Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If loses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Compliance requirements

Users shall abide by all applicable local laws and regulations when using this manual and the matching Geehy products. Users shall understand that the products may be restricted by the export, re-export or other laws of the countries of the product suppliers, Geehy, Geehy distributors and users. Users (on behalf of itself, subsidiaries and affiliated enterprises) shall agree and promise to abide by all applicable laws and regulations on the export and re-export of Geehy products and/or technologies and direct products.

6. Disclaimer

This manual is provided by Geehy "as is". To the extent permitted by applicable laws, Geehy does not provide any form of express or implied warranty, including without limitation the warranty of product merchantability and applicability of specific purposes. Geehy will bear no responsibility for any disputes arising from the subsequent design and use of Geehy products by users.

7. Limitation of liability

In any case, unless required by applicable laws or agreed in writing, Geehy and/or any third party providing this manual "as is" shall not be liable for damages, including any general damages, special direct, indirect or collateral damages arising from the use or no use of the information in this manual (including without limitation data loss or inaccuracy, or losses suffered by users or third parties).

8. Scope of application

The information in this manual replaces the information provided in all previous versions of the manual.