Freescale Semiconductor
Application Note

Document Number:AN4657
Rev. 0, 01/2013

Cyclic Redundant Checker Calculation on Power Architecture® Technology and Comparison of Big-Endian Versus Little-Endian

by: Tomas Kulig

Contents

1 Introduction

This application note includes two main parts. The first part is documentation that gives basic information about cyclic redundant checker (CRC), differences between calculation of CRC on personal computer (PC) and Freescale's Qorivva devices based on Power Architecture technology, and differences between big-endian and little-endian. The second part contains example code for calculating the CRC with the same sequence as Power Architecture, which was developed in DevC++ environment. This application note focuses on CRC on the MPC5744P.

2 Big-endian versus little-endian: what does it mean?

The difference is in how data is stored in the memory. The formats of storing the same data in little-endian and big-endian are shown in Figure 1. Little-endian stores low significant bytes in the lower addresses and big-endian stores more significant bytes in the lower addresses.

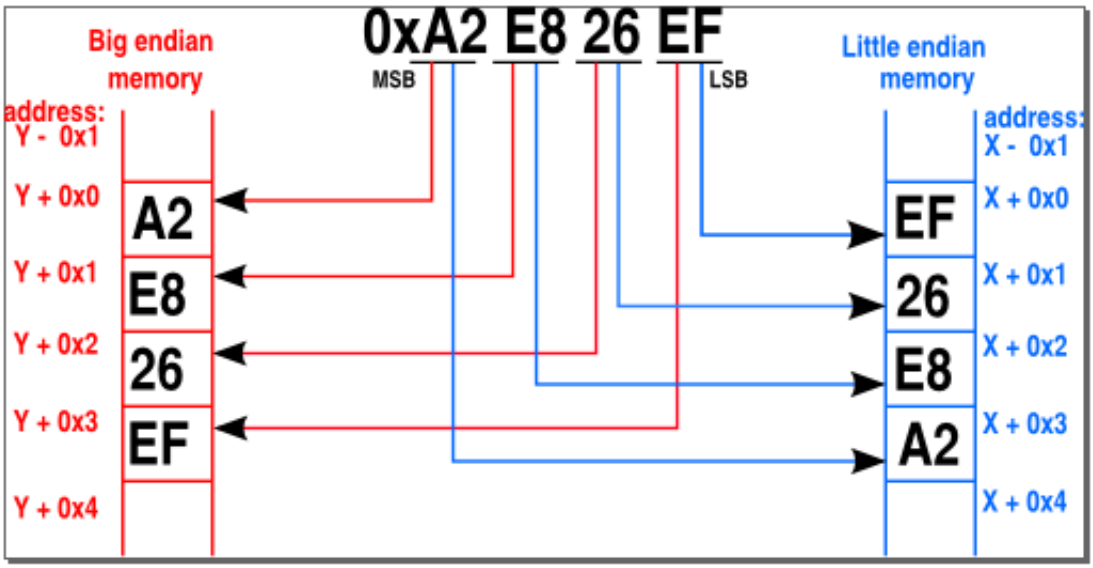

Figure 1. The formats of storing data in big-endian and little-endian

The collision can appear when data is stored in big-endian but the device works with them in the little-endian or on the contrary. Formats in which data is stored and the device works are very important. Figure 2 shows data stored in big-endian but read in little-endian. It is visible that the number 0xA2E826EF is not equal to the 0xEF26E8A2. The same collision will appear when data would be stored in little-endian and the device works in big-endian.

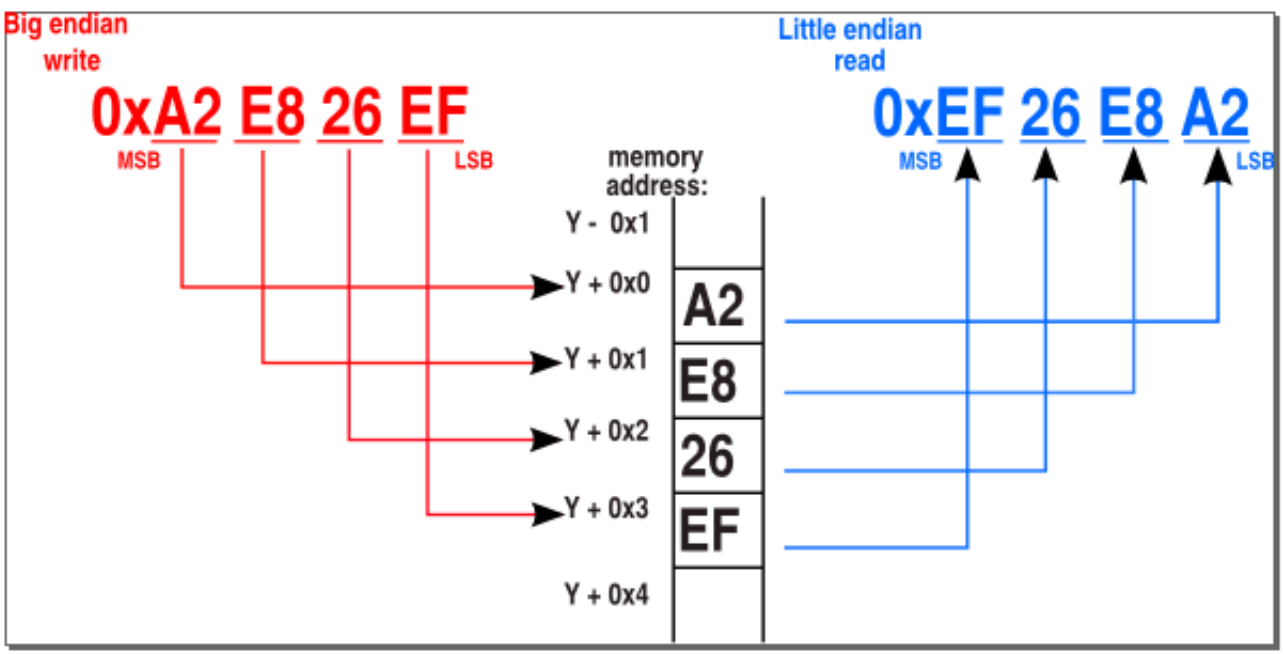

Figure 2. Collision between data stored in the big-endian and read in the little-endian

NOTE

Power Architecture works in big-endian and PC works in little-endian but the different results of CRC for the same polynomial on Power Architecture and PC are given by the sequence of calculation, not by data storage format (little- or big-endian). Power Architecture starts calculation of CRC from low significant bit of data against PC which starts calculation of the CRC from the most significant bit of data. It means the value of CRC does not depend on the data storage format.

3 Cyclic redundant checker (CRC)

CRC is the special type of hash function that detects errors in data during transmission or storage. CRC is transferred or stored with data where there are possibilities of data errors. New CRC is calculated immediately after receiving or loading data. If the new CRC is not equal to received or stored CRC, it is clear that an error has occurred during the storage or transmission. If both the CRCs are equal, it is clear that an error has not occurred during the storage or transmission. The probabilities of detection error are in Table 1 for all generating polynomials. Also there is a possibility to repair error through CRC in special cases.

Table 1. Probability of detection error

Bit of CRC	S[-]
8	0.9960937500
16	0.9999847412
32	0.9999999998

S means the probability of error detection.

3.1 Types of CRC code

CRC codes are sorted by number of bits of generating polynomial which is longer than length of calculated CRC by about one bit. Power Architecture mostly includes up to three kinds of generating polynomials. First is 8 bits CRC, which is used for CAN. Second is 16 bits CRC, which is used for Bluetooth, XMODEM, High-Level Data Link Control (HDLC), and so on. The last one is 32 bits CRC, which is used for Ethernet, MPEG-2, PNG, and so on. Generating polynomials that are used in MPC5744P are shown in Table 2.

Table 2. Types of CRC polynomial

Bit of CRC	Name of CRC	Generic polynomial
8	CRC-8 VDA CAN	X8+X4+X3+X2+1
16	CRC-16-CCITT	X16+X12+X5+1
32	CRC-32 Ethernet	X32+X26+X23+X22+X16+X12+X11+ X10+X8+X7+X5+X4+X2+X+1

3.2 Principle and example of CRC calculation

A simple example of transmitter and receiver is used to show the main force of CRC. Figure 3 shows calculation of CRC on transmitter side. During the transmission there was a change in data in the most significant bit. Figure 4 shows calculation and CRC check on receiver side, which is compared with CRC numbered by transmitter. As soon as the receiver compared both the CRCs, it found out that data change occurred during the transfer. The power of CRC is shown in this example where only 1 bit, which was changed during transfer, absolutely changes the CRC code. CRC is 0xC442 for correct or original data and 0xD463 for changed or invalid data.

Both Figure 3 and Figure 4 show the example of generating 16 bit CRC with polynomial x16+x12+x5+1. There are used modulo 2 (XOR functionality) for calculating CRC. Input sequence is 32 bit long 0xF2A9C0D1 (MSB first).

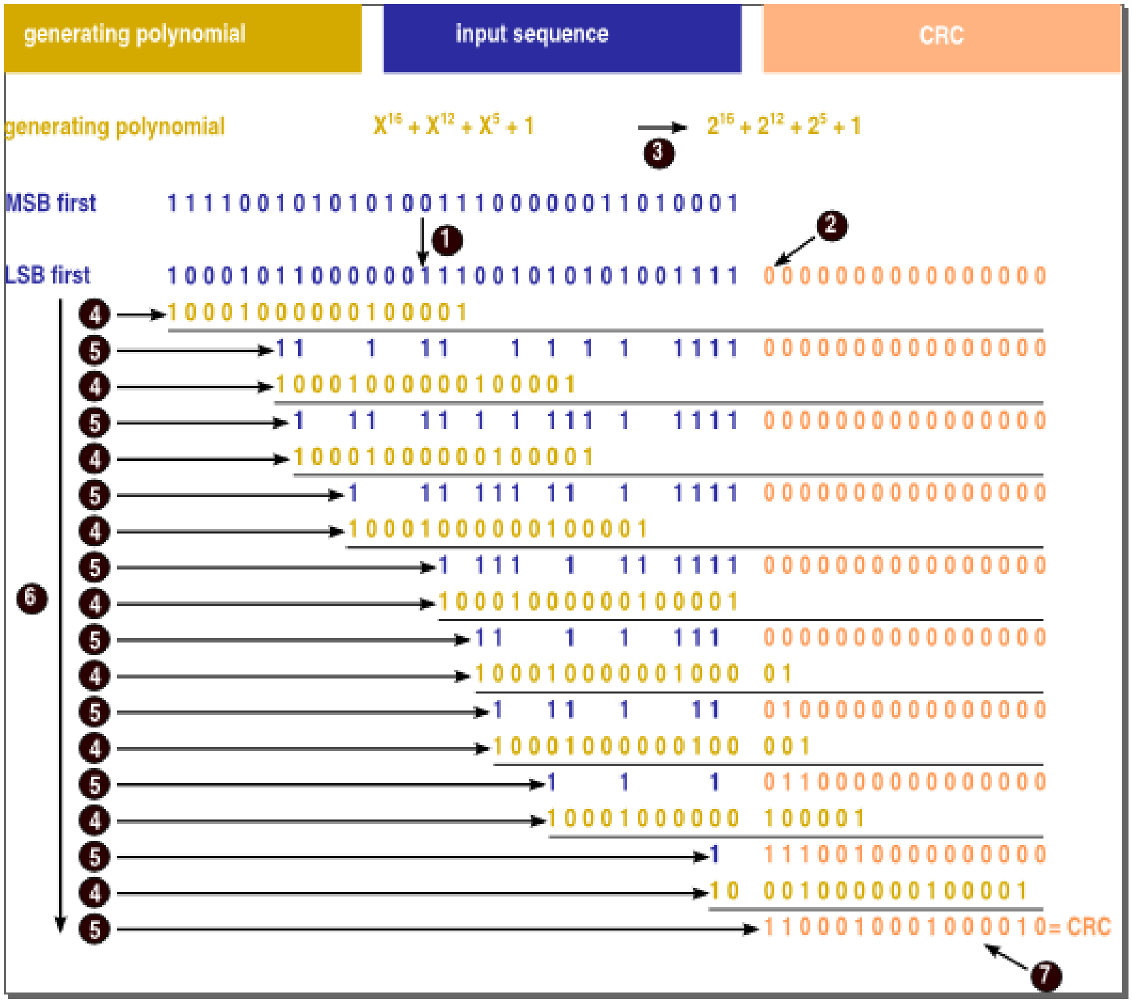

Figure 3. Calculation of CRC on the transmitter side

Sequence of calculating CRC:

1. Input sequence should have low significant bit on the left side.
2. Add zeros on the right side of input sequence, the number of zeros is given by the length of CRC. In this example, it is 16 zeros – 16 bits CRC.
3. Get generating polynomial in binary form so that X base will be replaced by 2 base .
4. Write generating polynomial in binary form under input sequence – align under first ‘1’ in input sequence.
5. Write new row as (input sequence | CRC) XOR (generating polynomial).
6. Do steps 4 and 5 until the input sequence is not equal to zero.
7. Read CRC.

The first step is very important because Power Architecture calculates CRC code from low significant bit. The new rows do not contain zeros for better readability. The polynomial that is used in this example is implemented in MPC5744P device and attached program.

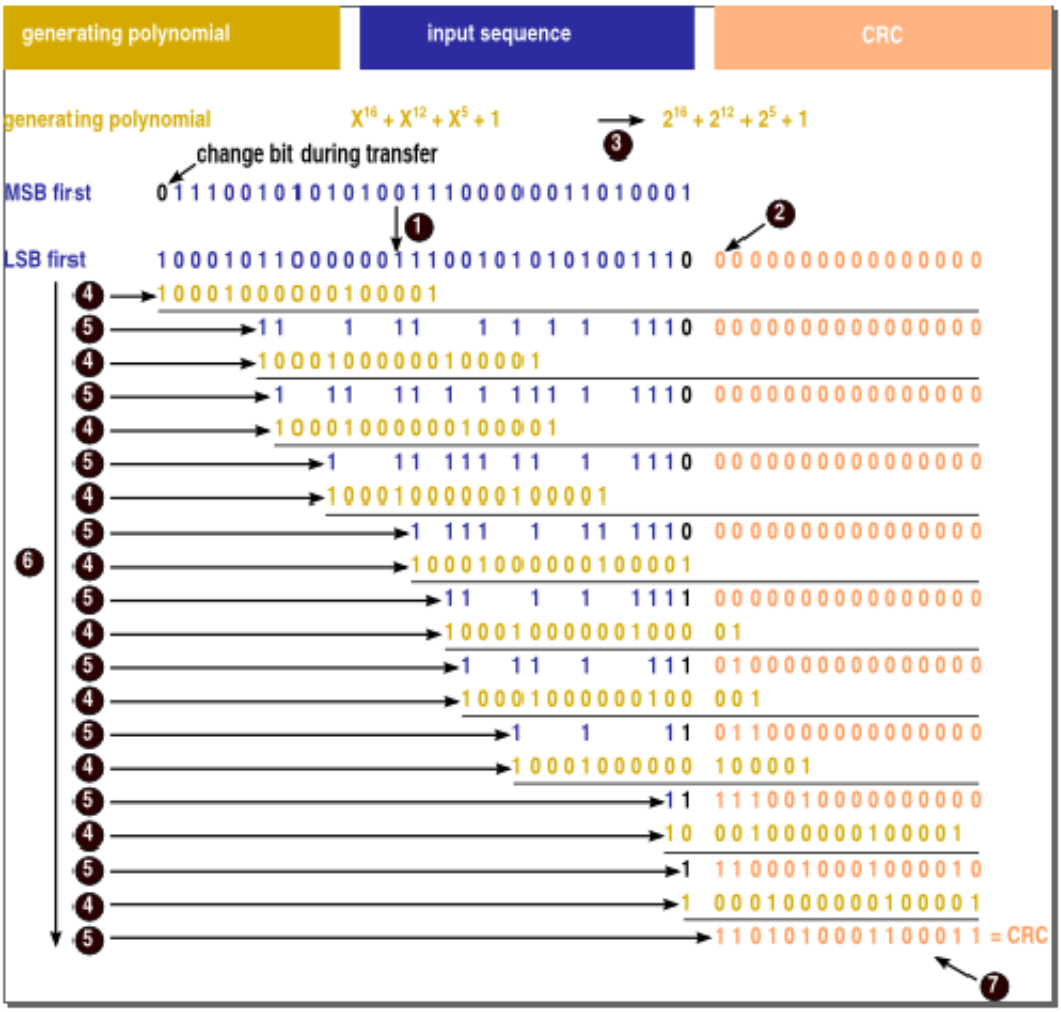

Figure 4. Calculation of CRC on the receiver side

NOTE

There are zero-wait states during the CRC computation on Power Architecture because the module uses pipeline scheme.

4 Description of program

Program can calculate CRC from three polynomials that are in Table 2. Data is possible to include via keyboard or from file. The flowchart of program is in Appendix A.

Data can be inserted in three numeral systems: decimal, hexadecimal, and binary for both source of data. Each of them has different prefix and different valid characters, which are shown in Table 3. Invalid character is replaced by zero. For example, if the program read number is “b12103”, replace it by “0b10100”. All the three formats of data can be used together in a file. For example, the first data will be in binary; second, in hexadecimal; and the third, in decimal.

Table 3. Valid characters

Base	Prefix	Valid character
Binary	b	‘0’ and ‘1’

Table continues on the next page...

Table 3. Valid characters (continued)

Decimal	no prefix	‘0’–‘9’
hexadecimal	x	‘0’–‘9’, ‘a’–‘f’, ‘A’–‘F’

4.1 Data source is file

The file format is shown in Figure 5. Initialization value of generating algorithm is on the first line in the file. Next lines are input data. All lines are separated by symbol ‘\0’. Symbol ‘end of file’ is inserted after the last data.

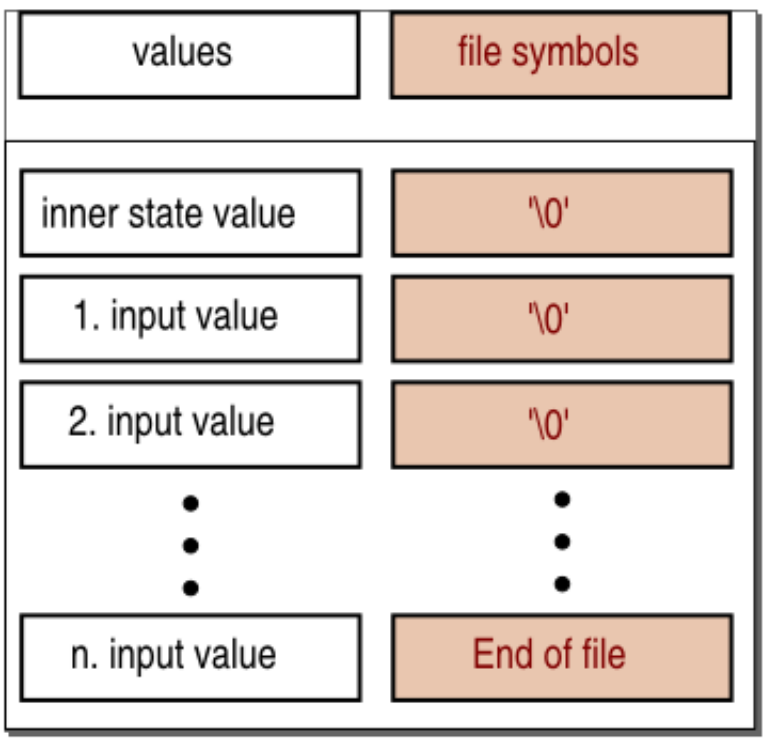

Figure 5. Structure of data file

4.2 Data source is keyboard

When the file is used, the first input from keyboard is initialization value of generating algorithm. When the file is used, the first input from keyboard is initialization value of generating algorithm and the next inputs are input data. The numeral system is not limited for inputs. After entering input data, the program enquires whether data entry would be continued.

5 Appendix A

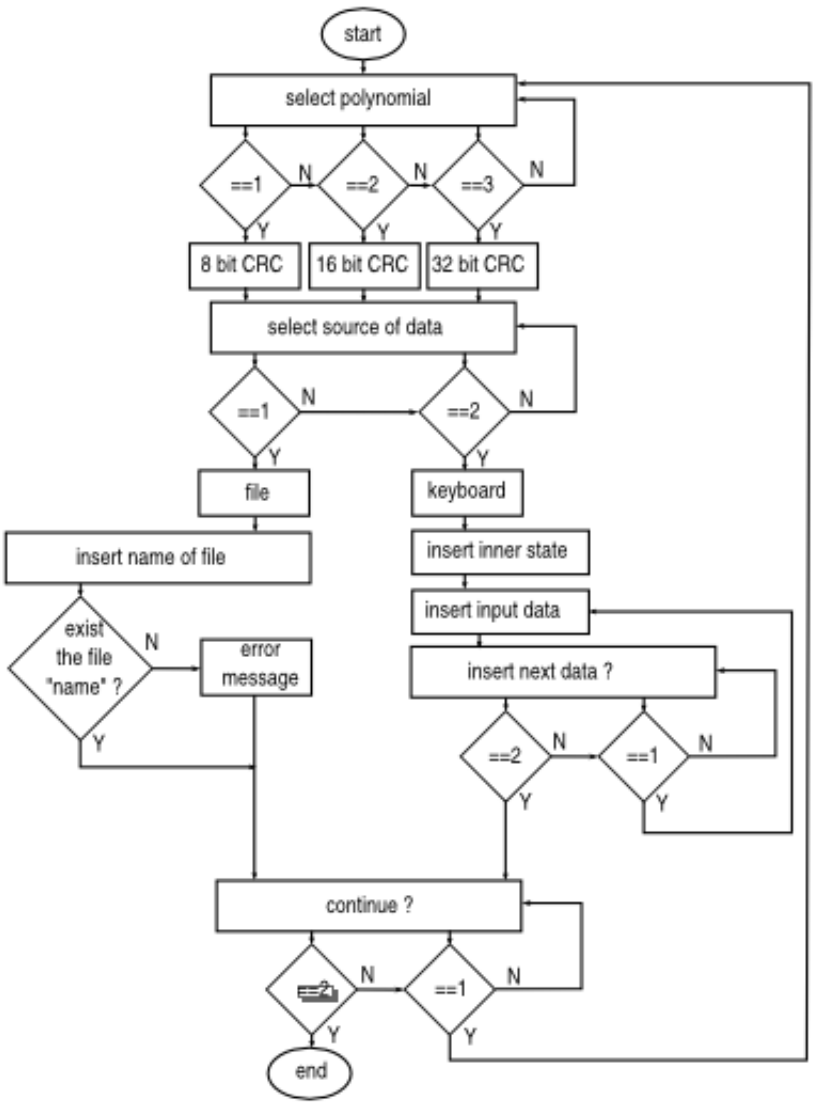

Figure 6. Flowchart of CRC program

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Document Number: AN4657
Rev. 0, 01/2013