DESIGN NOTE #038

AUTHOR: ANTHONY BARRETT

KEYWORDS: CRC, SRAM, ERROR DETECTION

Fast Look-up Table Driven 16-bit CRC Routine for Atmel AVR Microcontrollers

There are a few application notes around for 16-bit CRC (Cyclic Redundancy Check) calculating routines, but none are available in AVR assembler or are look-up table driven. The routine provided here is not only optimised in assembler but also uses a pre-calculated look-up table for absolute CRC calculation speed! This 16-bit CRC is the same as used in XMODEM, YMODEM, and ZMODEM File Transfer Protocols.

> "CRC-16 guarantees detection of all single and double bit errors, all errors with an odd number of error bits, all burst errors of length 16 or less, 99.9969% of all 17-bit error bursts, and 99.9984% of all possible longer error bursts. By contrast, a double bit error, or a burst error of nine bits or more can sneak past the arithmetic checksum".

This routine was primarily written for error checking of serially transmitted data. The application uses this routine to calculate a CRC on the data before and after it is transmitted over an error prone medium e.g., a lengthy RS-232 serial connection. The pre-calculated CRC is sent along with the data and is then compared to another calculated CRC of the received data. If the two do not agree then an error in the data transmission has been detected. The data can then be retransmitted or ignored depending on the application's needs and protocol. The code could also be modified to calculate and check Program FLASH ROM.

To use this routine, simply load Index Register Y (r28 & r29) to point to the data in SRAM. Then load r18 with the number of bytes of data and call the subroutine "Calc16CRC". The 16-bit result is returned in r16 (low Byte) and r17 (High byte). Here is an example:

```
DoIt: ldi   YL,low(Data)
      ldi   YH,high(Data)
      ldi   r18,100
      rcall Calc16CRC
      sts   CRCLow,r16
      sts   CRCHi,r17
      ...
```

Obviously the number of bytes is limited to 255 (eight bits), but again the routine can be modified to support a 16 bit count if needed.

The code has been provided to “www.AVRFreaks.com” in Zip-format (FastCRC16.zip) for you to download. I do not recommend typing in the look-up table manually as any typo errors will compromise the results.

Interestingly, I used this routine in my home alarm system to provide secure serial communications between the Main Controller (Mega163) and the Control Panels (8515). I decided for simplicity to use serial TTL levels and a good quality shielded cable for the hardware level. Sure, TTL levels are not meant for long cable runs . So to make the system reliable I provided a robust protocol level using the 16-bit CRC. After all, program code costs me nothing but extra hardware does (RS-232 driver/receiver chips). Anyway to get to the point, I was surprised to find that the TTL level serial communications over a shielded cable (approx. 16 meters in length) has produce no detectable errors for over a year now. So I guess that’s a complement for AVR’s I/O Port and UART Hardware.

Reference

XMODEM/YMODEM PROTOCOL REFERENCE by Chuck Forsberg, 18-June-1988

Code

```
;     **********************************
;     *                                *
;     *    Fast lookup table driven    *
;     *     16 bit CRC Routine for     *
;     *   ATMEL AVR microcontrollers   *
;     *                                *
;     *          17-Oct-2002           *
;     *                                *
;     *       By Anthony Barrett       *
;     *                                *
;     **********************************
;
; Y (r28, R29) points to data in SRAM, r18 = data size in bytes.
; CRC result is returned in r16 (low byte) & r17 (high byte).
; Uses regisisters: r0, r1, r2, r16, r17, r18, r19, r28, r29, r30, r31

Calc16CRC:clr    r16
          clr    r17
          clr    r1

CRCLoop:  ld     r2,Y+
          mov    r19,r17
          eor    r19,r2
          ldi    ZL,low(2 * CRC16Tab)
          ldi    ZH,high(2 * CRC16Tab)
          add    ZL,r19
          adc    ZH,r1
          add    ZL,r19
          adc    ZH,r1
          lpm
          mov    r17,r16
```

```
                mov     r16,r0
                adiw    ZL,1
                lpm
                eor     r17,r0
                dec     r18
                brne    CRCLoop
                ret


        ; High speed 16 bit CRC lookup table:
        ; Polynomial: X^12 + X^5 + 1

        CRC16Tab: .dw   $0000,$1021,$2042,$3063,$4084,$50a5,$60c6,$70e7
                  .dw   $8108,$9129,$a14a,$b16b,$c18c,$d1ad,$e1ce,$f1ef
                  .dw   $1231,$0210,$3273,$2252,$52b5,$4294,$72f7,$62d6
                  .dw   $9339,$8318,$b37b,$a35a,$d3bd,$c39c,$f3ff,$e3de
                  .dw   $2462,$3443,$0420,$1401,$64e6,$74c7,$44a4,$5485
                  .dw   $a56a,$b54b,$8528,$9509,$e5ee,$f5cf,$c5ac,$d58d
                  .dw   $3653,$2672,$1611,$0630,$76d7,$66f6,$5695,$46b4
                  .dw   $b75b,$a77a,$9719,$8738,$f7df,$e7fe,$d79d,$c7bc
                  .dw   $48c4,$58e5,$6886,$78a7,$0840,$1861,$2802,$3823
                  .dw   $c9cc,$d9ed,$e98e,$f9af,$8948,$9969,$a90a,$b92b
                  .dw   $5af5,$4ad4,$7ab7,$6a96,$1a71,$0a50,$3a33,$2a12
                  .dw   $dbfd,$cbdc,$fbbf,$eb9e,$9b79,$8b58,$bb3b,$ab1a
                  .dw   $6ca6,$7c87,$4ce4,$5cc5,$2c22,$3c03,$0c60,$1c41
                  .dw   $edae,$fd8f,$cdec,$ddcd,$ad2a,$bd0b,$8d68,$9d49
                  .dw   $7e97,$6eb6,$5ed5,$4ef4,$3e13,$2e32,$1e51,$0e70
                  .dw   $ff9f,$efbe,$dfdd,$cffc,$bf1b,$af3a,$9f59,$8f78
                  .dw   $9188,$81a9,$b1ca,$a1eb,$d10c,$c12d,$f14e,$e16f
                  .dw   $1080,$00a1,$30c2,$20e3,$5004,$4025,$7046,$6067
                  .dw   $83b9,$9398,$a3fb,$b3da,$c33d,$d31c,$e37f,$f35e
                  .dw   $02b1,$1290,$22f3,$32d2,$4235,$5214,$6277,$7256
                  .dw   $b5ea,$a5cb,$95a8,$8589,$f56e,$e54f,$d52c,$c50d
                  .dw   $34e2,$24c3,$14a0,$0481,$7466,$6447,$5424,$4405
                  .dw   $a7db,$b7fa,$8799,$97b8,$e75f,$f77e,$c71d,$d73c
                  .dw   $26d3,$36f2,$0691,$16b0,$6657,$7676,$4615,$5634
                  .dw   $d94c,$c96d,$f90e,$e92f,$99c8,$89e9,$b98a,$a9ab
                  .dw   $5844,$4865,$7806,$6827,$18c0,$08e1,$3882,$28a3
                  .dw   $cb7d,$db5c,$eb3f,$fb1e,$8bf9,$9bd8,$abbb,$bb9a
                  .dw   $4a75,$5a54,$6a37,$7a16,$0af1,$1ad0,$2ab3,$3a92
                  .dw   $fd2e,$ed0f,$dd6c,$cd4d,$bdaa,$ad8b,$9de8,$8dc9
                  .dw   $7c26,$6c07,$5c64,$4c45,$3ca2,$2c83,$1ce0,$0cc1
                  .dw   $ef1f,$ff3e,$cf5d,$df7c,$af9b,$bfba,$8fd9,$9ff8
                  .dw   $6e17,$7e36,$4e55,$5e74,$2e93,$3eb2,$0ed1,$1ef0
```