Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 002-04934

Spec Title: AN204934 - Note on Calculating Reload Values FFMC8L

Replaced by: None

Note on Calculating Reload Values FFMC8L

This application note describes how to generate circular interrupts with a fixed reload time and also shows how to calculate the correct reload value taking in account the produced extra code.

1 Introduction

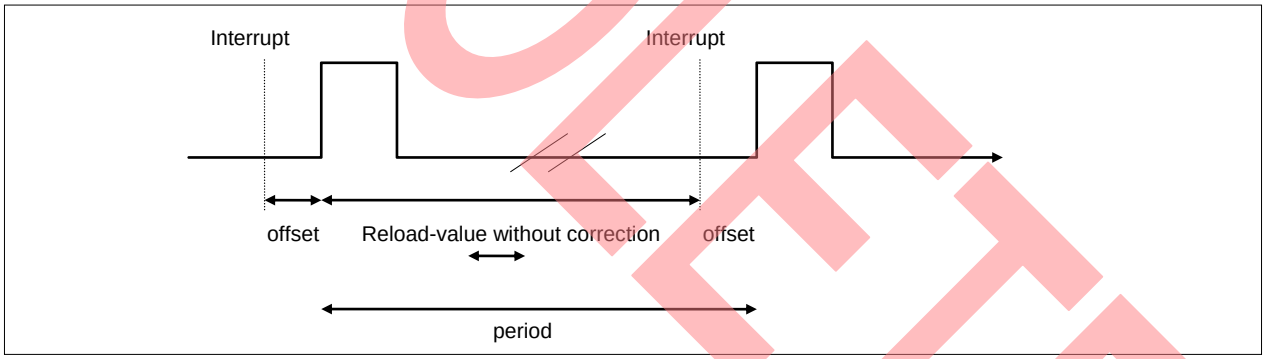
When using timers in reload mode, the main intention is to generate circular interrupts with a fixed reload time. But when interrupt service routines are called, a certain overhead will be produced by the compiler (context save!). This note shows how to calculate the correct reload value taking in account the produced extra code.

2 Calculating Reload Values FFMC8L

In this example, a rectangular signal with a frequency of 1,5 kHz should be produced on pin 40. The reload value has to be calculated to obtain this period. The positive width of the signal is fixed and much smaller than the period itself.

The diagram below shows the signal and the ISR procedure: When the interrupt is signaled and processed immediately, which would assume a certain priority and no other interrupt currently active, the context will be saved on the stack after the ISR-subroutine is called. This context save is produced by the compiler. After that the intended taks can be processed – in this case a signal on pin 40 with a definite width.

Figure 1. Producing a signal on pin 40 using a 16-bit(reload)-timer interrupt



In the produced code, the offset (due to calling ISR and context save) takes 64 cycles (40_{hex}) which must be included in the calculation :

$Reload = FFFF_{hex} - (f_c / (4\ f_{int})) + 40_{hex}$ mit : f_c : *Quartzfreq.*
f_{int} : *Interruptfreq.*

In this example: With f_c = 10 MHz and f_{int}=1500Hz: *Reload*=$F9BD_{hex}$

2.1 Interrupt-Service Routione 16-bit-Timer:

(C-Source-code printed bold)

before calling ISR : finish actual command and save PS and address on stack approx. 10-14 cycles

```
220:  void TC16INT6()
221: {
        C2BA: 40  PUSHW    A                4
        C2BB: 43  XCHW     A,T              2
        C2BC: 40  PUSHW    A                4
        C2BD: F3 MOVW      A,EP             2
        C2BE: 40 PUSHW     A                4
        C2BF: 41 PUSHW     IX               4
        C2C0: F1 MOVW      A,SP        2
        C2C1: E2 MOVW      IX,A        2
        C2C2: 08 MOV       A,R0             3
        C2C3: 10 SWAP                       2
        C2C4: 09 MOV       A,R1             3
        C2C5: 40 PUSHW     A                4
222: TMCR = 0x22;   /* Int enable egain */
        C2C6: 851822 MOV       18,#22       4
223:  TCHR = 0xF6;   /* Reload value */
        C2C9: 8519F6 MOV       19,#F6       4
224:  TCLR = 0x3B;
        C2CC: 851A3B MOV       1A,#3B       4
225:  TCS = 1;       /* start counter again */
        C2CF: A818 SETB     18:00           4
                                            64 cycles
226:  PDR4_0 = 1;    /* show pulse on PIN 40 */
        C2D1: A80F SETB     0F:00
227:  wait(20);
        C2D3: E40014 MOVW      A,#0014
        C2D6: 40 PUSHW     A
        C2D7: 31C04B CALL      \wait
        C2DA: 50 POPW      A
228:  PDR4_0 = 0;
        C2DB: A00F CLRB     0F:00
```

```
229:}
        C2DD: 50 POPW     A
        2DE: 49 MOV       R1,A
        2DF: 10 SWAP
        2E0: 48 MOV       R0,A
        2E1: F2 MOVW      A,IX
        2E2: E1 MOVW      SP,A
        2E3: 51 POPW      IX
        2E4: 50 POPW      A
        2E5: E3 MOVW      EP,A
        2E6: 50 POPW      A
        2E7: 43 XCHW      A,T
        2E8: 50 POPW      A
        2E9: 30 RETI
```

Document History

Document Title: AN204934 - Note on Calculating Reload Values FFMC8L

Document Number: 002-04934

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	—	WOFR	02/20/2014	Initial release.
*A	5235611	WOFR	04/22/2016	Migrated Spansion Application Note from MCU-AN-389012-E-V10to Cypress format. This AN to be Obsolete, since this is the old 8-bit MCU

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com