INPUT CAPTURE WITH ST62 16-BIT AUTO-RELOAD TIMER

by 8-bit Micro Application Team

1 INTRODUCTION

This note presents how to use the ST62 16-bit Auto-Reload Timer (ARTimer) to measure durations or frequencies of an input signal. An example shows how to capture an input signal to make an output signal with the same frequency as input signal but with a duty cycle equal to 50%.

1.1 16 BIT AUTO-RELOAD TIMER DESCRIPTION

This timer is a 16-bit downcounter timer with prescaler (see Figure 1.). It includes auto-reload PWM, capture and compare capability with two input(CP1,CP2) and two output pins(OVF,PWM). It is controlled by the following registers (8 bit):

– Status control registers (SCR1, SCR2, SCR3, SCR4)

– Capture register high (CPH) and low (CPL). For the total 16-bit the register is CP.

– Mask register high (MASKH) and low (MASKL). For the total 16-bit the register is MASK.

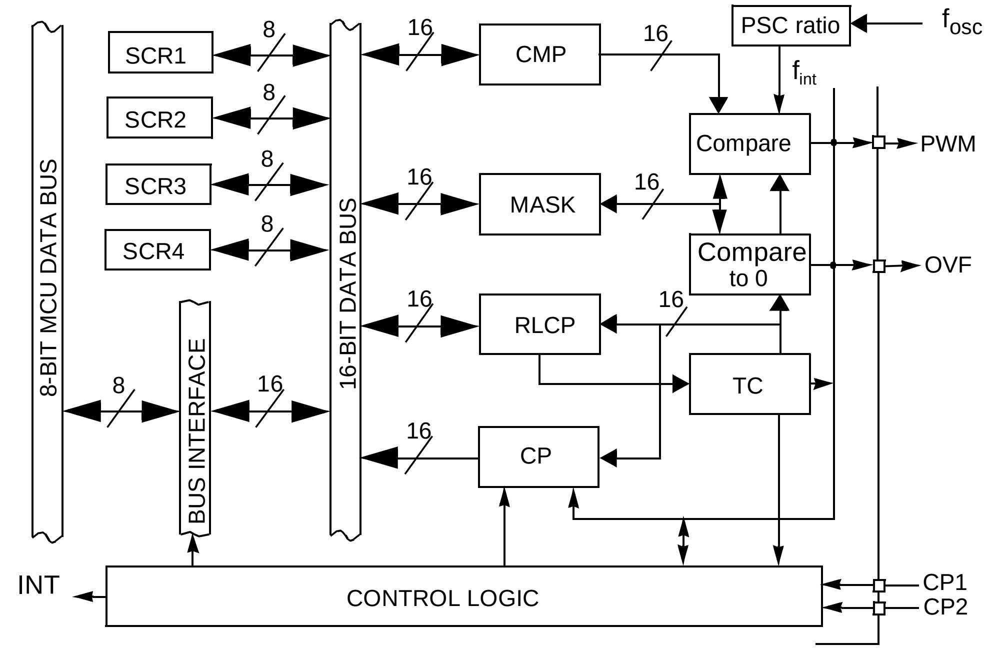
– Decremental counter register (TC with 16-bit)

– Compare register high (CMPH) and low (CMPL). For the total 16-bit the register is CMP.

– Reload/Capture register high (RLCPH) and low (RLCPL). For the total 16-bit the register is RLCP.

The prescaler ratio can be programmed to choose the timer input frequency f_{int}(see Table 1).

Figure 1. 16-bit Auto-Reload Timer Block Diagram



1.2 CAPTURE MODE

This can be used to measure time duration or frequencies (see Figure 2.).This mode is used to measure the time elapsed between two edges of one or two external signal. Each edge could be rising or falling depend on initialisation.

With the 16-bit TC downcounter and with f_{osc} to 8Mhz, a signal of 4ms duration can be measured with a resolution of 1/32768.

Example:

Let's measure the time elapsed between two rising edges on CP2:

The 16-bit CP value contains the time between the two CP2 rising edges and will be divided by two to be loaded in the 16-bit CMP register.

The capture mode uses the CP2 triggered restart mode with CP2 event detection (RDSEL2=1, RDSEL1=0 of SCR2 register). It's mean that each CP2 edge sets off the capture of the TC value in the CP register and then reloads TC register with the RLCP value.

The CP2 interrupt is enabled (CP2IEN=1 of SCR3 register) and CMP interrupt is enabled (CMPIEN=1 of SCR3 register) to manage the output bit PA2.

In the CP2 interrupt sub-program the output bit PA2 is set to 1. In the CMP interrupt sub-program the output bit PA2 is set to 0.

The main program calculates the division by 2 of the captured 16-bit value and saves it in NewCMPh and NewCMPl.

The prescaler ratio must be programmed according to the expected duration to measure. In this example it is programmed to: prescaler ratio = 16, clock source = f_{osc} = 8Mhz.

The period to measure must be in the range of 250µs to 133ms.

The sharing of a 16-bit data between the main program and the interrupt sub-program obliges to disable the interruption for each handling of this data in the main program. This causes a jitters of up to 30µs.

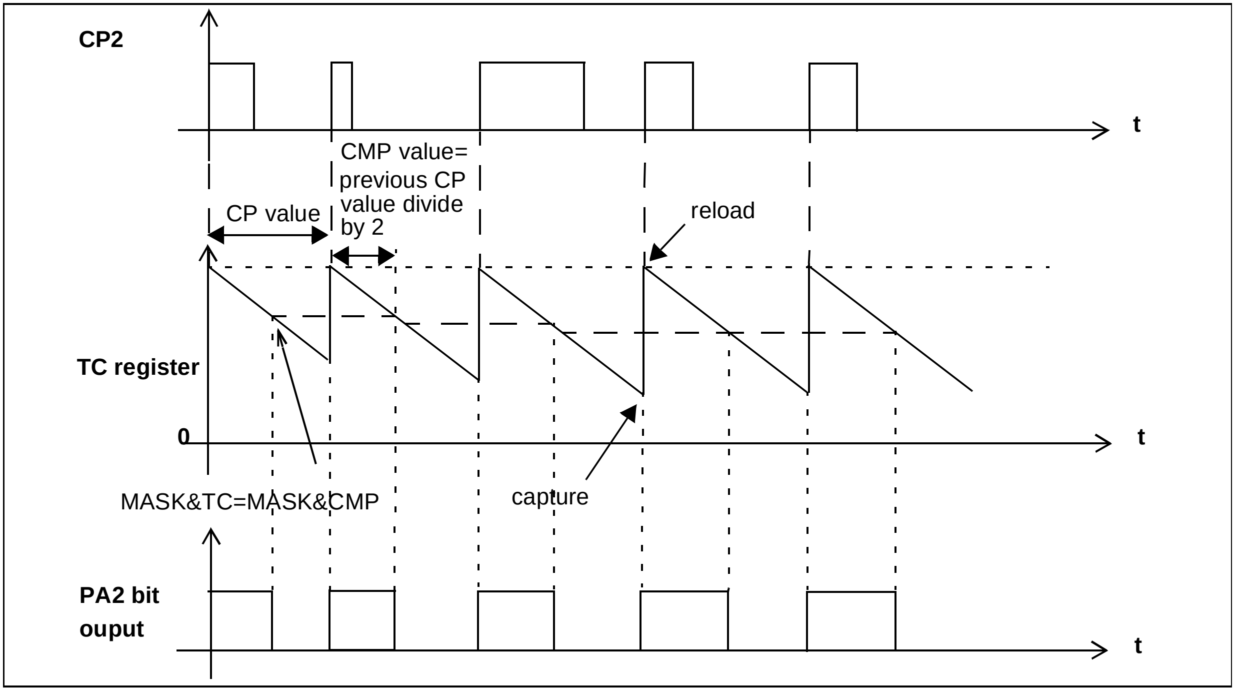
The delay between the input signal active edge and the output signal is of 36µs.

The RLCP register is load with FFFFh to avoid subtraction to calculate the delay between the CP2 edge and the compare value reached by the TC value.

Table 1. Prescaler Programming Ratio

PSC2	PSC1	PRESCALER Ratio
0	0	Clock Disabled
0	1	1
1	0	4
1	1	16

Figure 2. TC, CP and CMP value evolution synchronized with the input CP2.



Program example

```
;****************************************************************************
;*****************ST6230 Auto-Reload 16-bit Capture mode ******************
;***
;*** object: Give an output TTL square siqnal at the same frequency
;***    of the no symmetrical TTL input signal
;***
;*** input : TTL signal in the range of 7.5Hz to 4000Hz on CP2
;***
;*** output: TTL signal with the same frequency of CP2 but with
;***    a duty cycle of 50%. The signal has a delay of 36µs
;***    and a jitters of 30µs with a clock frequency of 8Mhz.
;***
;*** author: Jean-Luc CREBOUW
;***
;****************************************************************************
       .vers "st6230"
       .romsize 8
;*** data registers ***
  .input "623x.asm"
;*** data RAM ***
templ  .def        084h         ; low byte of the divider by two
temph  .def        085h         ; high byte of the divider by two
NewCMPl.def        086h         ; low byte of the result divider
NewCMPh.def        087h         ; high byte of the result divider
data   .def        088h         ; data copy of the A port
save_cpl.def       089h         ; save the CP hight
save_cph.def       08ah         ; save the CP low

;************************** INITIALIZATION *********************************
      .org         800h
reset
      reti

;*** ART16 Initialisation ***
      ldi          SCR1,0F0h    ; prescal by 16 to have f$_{int}$=.5 Mhz
                                ; Reload mode
                                ; Runres
                                ; No interrupt with overflow
                                ; Reset mode for OVFMD
      ldi          SCR2,02h     ; CP1 input interrupt disable
                                ; CP2 triggered restart mode with CP2 event
                                ; detection
      ldi          SCR3,0D0h    ; CP2 polarity with rising edge
```

```
                                ; CP2 interrupt enable
                                ; Compare interrupt enable
                                ; Compare to zero interrupt disable
      ldi         SCR4,0h       ; Overflow output disable
                                ;PWM output disable
      ldi         RLCPH,0FFh    ; RLCP register to FFFFh
      ldi         RLCPL,0FFh    ;
      ldi         CMPH,0FFh     ; CMP register to FF00h
      ldi         CMPL,000h     ;
      ldi         MASKH,0FFh    ; MASK = 0FFFFh
      ldi         MASKL,0FFh
;*** PortA initialisation for output bit 2 and CP2 input
      ldi         ddra,04h
      ldi         ora,04h
      clr         a
      ld          data,a        ; data = 0
;*** GENERAL INTERRUPT ***
      ldi         ior,10h       ;Enables all interrupts.

;**************************************************************************
;****************************Main program*******************************
;***********divide the CP value by two to load CMP register with **********
PULSE:
;*** read the previous capture out of interrupt to avoid save_cpl
;*** and save_cph from a different CP value
      ldi         ior,00h       ; disables all interrupts.
      ld          a,save_cpl
      ld          templ,a
      ld          a,save_cph
      ldi         ior,10h       ; Enables all interrupts.
      ld          temph,a
;*** divide by two temp (16-bit)
      clr         a
      ld          a,templ
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      ld          templ,a
      clr         a
      ld          a,temph
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      rlc         a
      ld          temph,a
      jrnc        no_1
      ld          a,templ
      addi        a,080h
      ld          templ,a
no_1:
      ld          a,temph
      addi        a,080h
      ld          temph,a
      ld          a,templ
;*** store the next CMP value out of interrupt
      ldi         ior,00h       ; disables all interrupts.
      ld          NewCMPl,a
      ld          a,temph
      ld          NewCMPh,a
      ldi         ior,10h       ; Enables all interrupts.
      jp          PULSE

;*************************End of Main program***************************
;**********************************************************************
;*************************UART IT management****************************
it_uart:
      ld          x,a           ; save a
;*** if compare interrupt
      jrs         5,SCR3,it_cp2
      ld          a,data ; output port PA2 = 0
      ld          dra,a
      set         2,data        ; data bit 2 = 1
      res         3,SCR3        ; reset CMPFLG
      ld          a,x           ; restore a
      reti
it_cp2:
;*** else CP2 interrupt
      ld          a,data        ; output port PA2 = 1
      ld          dra,a
      res         2,data        ; data bit 2 = 0
      res         5,SCR3        ; reset CP2FLG
      res         5,SCR2        ; reset CP2ERR
      ld          a,CPH         ; read CP register and save the 16-bit value
      ld          save_cph,a    ; in save_cph and save_cpl
      ld          a,CPL
      ld          save_cpl,a
      ld          a,NewCMPh
      ld          CMPH,a        ; store NewCMPh and NewCMPl in CMP to have
      ld          a,NewCMPl     ; CMP = previous CP / 2
      ld          CMPL,a
      ld          a,x           ; restore a
      reti

;***********************End of UART IT management*************************
;***************************************************************************
;******************** Restart and interrupt Vectors ************************
      .org        0ff0h
      reti                      ; FF0h
      reti
      jp          it_uart       ; FF2h
      reti                      ; FF4h
      reti
      reti                      ; FF6h
      reti
      .org        0ffch
nmi   nop
      reti
res   jp          reset
```

"THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNEXION WITH THEIR PRODUCTS."

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without the express written approval of STMicroelectronics.

STMicroelectronics Group of Companies
Australia - Brazil - Canada - China - France - Germany - Italy - Japan - Korea - Malaysia - Malta - Mexico - Morocco - The Netherlands - Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A.

http://www.st.com