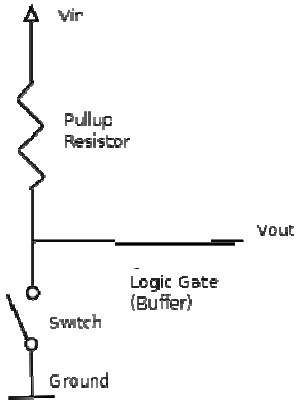
EA079 - Laboratório de Mico- e Mini-computadores: *Hardware*
Segunda Avaliação Individual - A

RA: _____ Nome:_____Turma: C

1. Na montagem de um circuito contendo diodos e capacitores eletrolíticos,
   a) (0.5 pt) como você identifica **visualmente** um diodo?
   b) (0,5 pt) como você diferencia **visualmente** a polaridade de um capacitor eletrolítico?

2. (1.0 pt) Determine a resistência do resistor *pull-up* no circuito para um pino Vout do microcontrolador MCF51CN128, considerando Vin=5V e a corrente máxima permitida I=0.5mA.



2. Em um projeto foram utilizados 3 componentes do microcontrolador MCF51CN128 para gerar/tratar interrupções: IRQ (PTC4), de prioridade 7; *input capture* (PTE3), de nível de prioridade 6; e *external interrupt* (PTG0), de nível de prioridade 1.
   c) (1.0 pt) Qual(is) destes componentes geram interrupções por eventos externos? Justifique.
   d) (0.5 pt) Qual das interrupções é não-mascarável? Justifique.
   e) (0.5 pt) A qual dos três tipos de interrupção enquadra a seguinte descrição: "*When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) are selected as the active edge that triggers an event handler.*"
   f) (0,5 pt) Uma *external interrupt* **sempre suspende** o fluxo corrente de execução? Justifique.
   g) (0.5 pt) IRQ pode suspender o fluxo de tratamento da exceção *input capture*. Se quisermos que um fragmento de instruções dependentes do tratamento da exceção *input capture* seja executado sem interrupção, o que podemos fazer?

3. No projeto de um circuito de interface entre o microcontrolador MCF51CN128 e um *keypad*, foram utilizados 2 componentes do *Processor Expert* para tratar *bouncing* das teclas. A seguir são mostrados os trechos de código dos arquivos ProcessorExpert.c e Events.c, considerando que a montagem do *hardware* seja tal que o sinal no pino PTG0 passe para nível lógico 0 quando uma tecla for pressionada.

ProcessorExpert.c

```
#include "TOC.h"
#include "Keyboard.h"
/* Include shared modules, which are
used for whole project */
unsigned char get_char();
byte key_status;
unsigned char aux = 0x00;

void main(void)
{
  unsigned char c;
  PE_low_level_init();
  key_status = 0x00U;
  Keyboard_SetEdge(0x00U);
  Keyboard_Enable();
  TOC_DisableEvent();
  for(;;){c = get_char();}
}
unsigned char get_char () {
      unsigned char buffer;
      while (!aux) {};
      buffer = aux;
      aux = 0;
      return buffer;
}
```

Events.c

```
extern byte key_status;

void TOC_OnInterrupt(void)
{
      if (key_status) {
            /* Scan the keys */
            Keyboard_SetEdge(1);
      } else {
            Keyboard_SetEdge(0);
      }
      TOC_DisableEvent();
      Keyboard_Enable();
}

void Keyboard_OnInterrupt(void)
{
      if(key_status)key_status=0;
      else key_status = 0x01U;
      Keyboard_Disable();
      TOC_SetPeriodMode(1);
      TOC_EnableEvent();
}
```

Os dois componentes foram configurados conforme os parâmetros apresentados nas seguintes telas capturadas:

Properties | Methods | Events

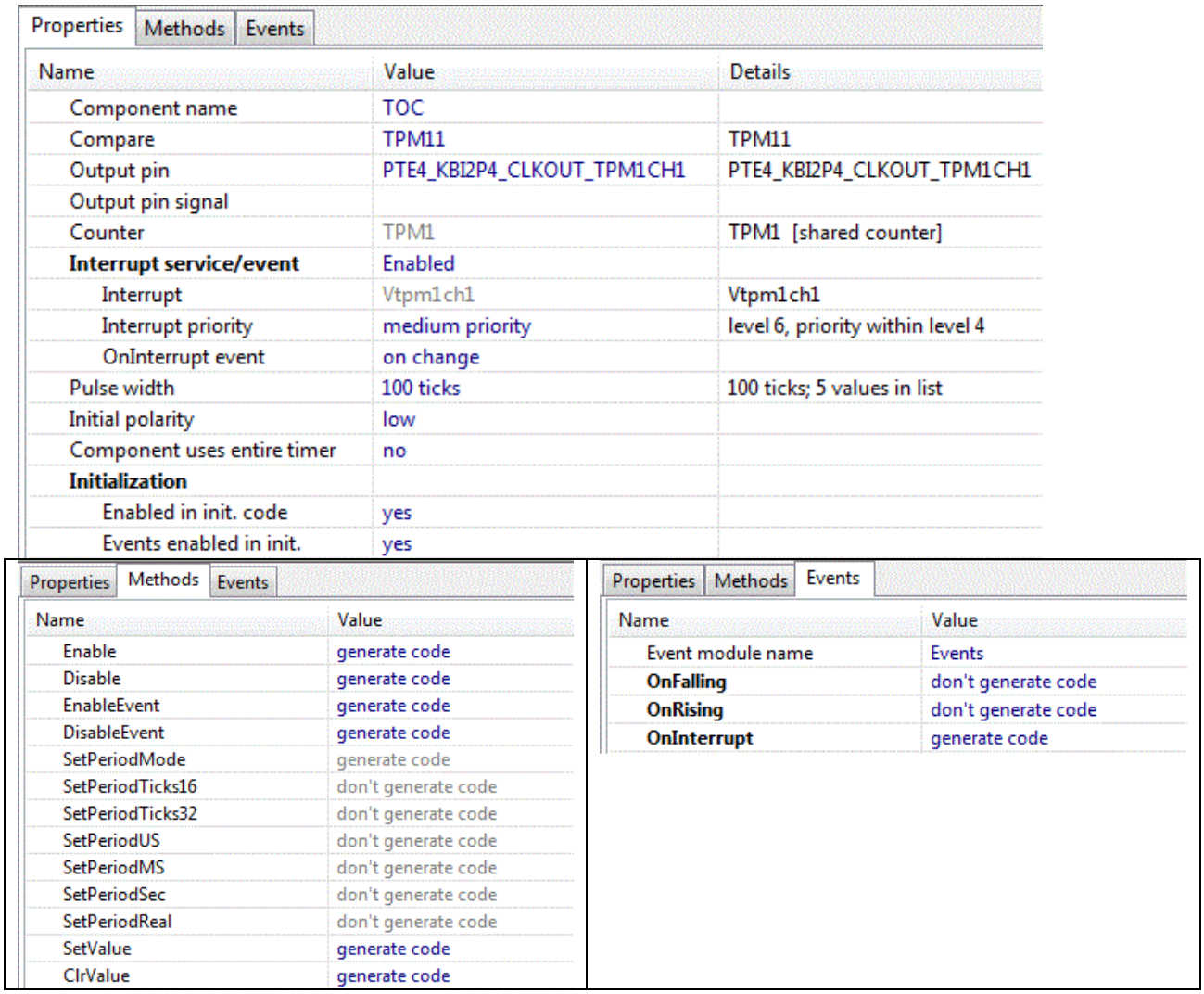
Name	Value	Details
Component name	Keyboard	
Pin	PTG0_KBI1P0_FB_A8_FB_AD8_SCL2	PTG0_KBI1P0_FB_A8_FB_AD8_SCL2
Pin signal		
Pull resistor	autoselected pull	no pull resistor
Generate interrupt on	falling edge	falling edge
Invert interrupt trigger condition	no	
Interrupt	Vkeyboard	Vkeyboard
Interrupt priority	medium priority	level 1, priority within level 2
Initialization		
Enabled in init. code	yes	

Properties | Methods | Events

Name	Value
Enable	generate code
Disable	generate code
GetVal	generate code
SetEdge	generate code

Properties | Methods | Events

Name	Value	Details
Event module name	Events	
OnInterrupt	generate code	
Event procedure name	Keyboard_OnInterrupt	
Priority	same as interrupt	1



Properties | Methods | Events

Name	Value	Details
Component name	TOC	
Compare	TPM11	TPM11
Output pin	PTE4_KBI2P4_CLKOUT_TPM1CH1	PTE4_KBI2P4_CLKOUT_TPM1CH1
Output pin signal		
Counter	TPM1	TPM1 [shared counter]
Interrupt service/event	Enabled	
Interrupt	Vtpm1ch1	Vtpm1ch1
Interrupt priority	medium priority	level 6, priority within level 4
OnInterrupt event	on change	
Pulse width	100 ticks	100 ticks; 5 values in list
Initial polarity	low	
Component uses entire timer	no	
Initialization		
Enabled in init. code	yes	
Events enabled in init.	yes	

a) (2.0 pt) Em qual modo de operação foi configurado o canal 1 do módulo TPM1, se analisarmos a função gerada pelo *Processor Expert*? Justifique.

```
static void HWEnDi(void)
{
  word TmpCmpVal;          /* Temporary variable for compare value */

  if (EnUser) {
    (void)TPM1C1SC;       /* Reset request flag and */
 /* TPM1C1SC: CH1F=0,CH1IE=0,MS1B=0,MS1A=1,ELS1B=0,ELS1A=1,??=0,??=0 */
    setReg8(TPM1C1SC, 0x14U);
    TmpCmpVal = (word)(TPM1CNT + CmpVal);
    TPM1C1V = TmpCmpVal;
    while (TPM1C1V != TmpCmpVal) {} /* Wait for register update */
    /* TPM1C1SC: CH1IE=1 */
    setReg8Bits(TPM1C1SC, 0x40U);       /* Enable interrupt */
 /* PTEPF1: E4=3 */
  setReg8Bits(PTEPF1, 0x03U);
  } else {
 /* TPM1C1SC: CH1F=0,CH1IE=0,MS1B=0,MS1A=0,ELS1B=0,ELS1A=0,??=0,??=0 */
    setReg8(TPM1C1SC, 0x00U);           /* Disable interrupt */
```

```
  /* PTEPF1: E4=0 */
  clrReg8Bits(PTEPF1, 0x03U);
}
```

b) (2.0 pt) Explique a função da rotina **TOC_SetPeriodMode** conforme o código gerado pelo *Processor Expert*.

```
 byte TOC_SetPeriodMode(byte Mode)
 {
  bool TmrRun;         /* Temporary flag enable/disable */
  if (Mode > 0x04U) {/* Is a values of mode identifier out of range? */
      return ERR_VALUE;  /* If yes then error */
  }
  TmrRun = EnUser;  /* Store actual device state */
  if (EnUser) {     /* Is the device enabled by user? */
    EnUser = FALSE;  /* If yes then set the flag "device disabled" */
    HWEnDi();     /* Enable/disable device according to status flags */
  }
  TMode = Mode; /* Store given mode identifier to the variable TMode */
  TOC_SetCV(HighComp[Mode]); /* Set compare value for new period */
  if (TmrRun) {                        /* Was the device disabled? */
    EnUser=TRUE;           /* If yes set flag "device enabled" */
    HWEnDi();   /* Enable/disable device according to status flags */
  }
  return ERR_OK;                       /* OK */
}

static void TOC_SetCV(word Val)
{
  TPM1C1V = (word)(TPM1CNT + Val); /* Count and save new value */
  CmpVal = Val;               /* Store result to the variable CmpVal */
}
```

c) (1.0 pt) Explique o procedimento de tratamento de *bouncing* com uso dos dois componentes.