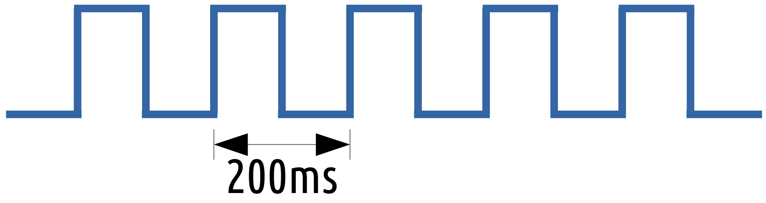
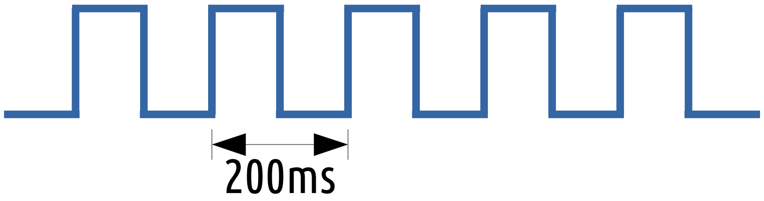
Timers – Example 1

- Problem:
  Blink a LED at a rate of 5Hz without using loop delays



Timers – Example 1

- Problem:
  Blink a LED at a rate of 5Hz without using loop delays



- Solution:
  A periodic interrupt service routine will toggle the LED every 100ms
- Questions: which timer? which mode? which prescaler? which count value?

(Answers here)

Timers – Example 1

- Facts:
  - $T_{INTR} = 100ms$
  - $F_{CLK} = 16MHz$
- Strategy:
  - $T_{CLK} = 62{,}5ns$
  - $T_{INTR} = 1.600.000 \cdot T_{CLK}$

- Goal: find CP, TP and CNT that verify:

  CP x TP x CNT = 1.600.000

- Restrictions:
  - CNT < 65535 (TC1)
  - CNT < 255 (TC0, TC2)
  - CP: 1,2,4,8,16,32,64,128
  - TP: 1,8,32,64,128,256,1024

Timers – Example 1

- CP=1 to avoid changing other peripherals, so:

$$TP \times CNT = 1600000$$

- Some values, by trial and error:

	1600000 = 1x1600000 (a)	Too large count, even for 16-bit
	= 8x200000 (b)	Too large count, even for 16-bit
	= 32x50000 (c)	Illegal (needs a 16 bit number in TC2)
	= 64x25000 (d)	Ok: CP=1, TP=64, CNT=25000 (16-bit)
	= 128x12500 (e)	Illegal (needs a 16 bit number in TC2)
Best choice (why?):	= 256x6250 (f)	Ok: CP=1, TP=256, CNT=6250 (16-bit)
	= 1024x1562,5 (g)	Rounding error

Timers – Example 1

- Solution (timer 1): CP=1, TP=256, CNT=6250
- Mode NORMAL
  - Counts from X to 65535, interrupt on overflow
  - X = 65536 - 6250 = 59286
  - Needs to reinitiate X
- Mode CTC
  - Counts from X to OCR1A, interrupt when TCNT1=OCR1A
  - X = 0, OCR1A= 6250
  - Don't need to reinitiate X

Timer/Counter – Example 1

```
/**********************************************
 * example1.c
 *  A simple demo using periodic interrupts.
 *   Purpose: Blink an LED at 5Hz (T=200ms)
 *            without using delays.
 *  Solution: The LED has to toggle every 100ms.
 *            This will be accomplished by the
 *            ISR of a periodic interrupt
 *            request.
 *********************************************
 * If Fosc=16MHz, Tosc=62,5ns. To generate an
 * interrupt request every 100ms we need to
 * count 1600000 clock periods.
 * We need to find combinations of CP, TP and
 * COUNT verifying CPxTPxCOUNT=1600000.
 *
 * Let's start with CP=1 to avoid disturbing
 * other peripherals:
 *
 * 1600000 = 1x1024x1562,5 = 1x256x6250
 *         = 1x128x12500   = 1x64x25000
 *         = 1x32x50000
 *
```

```
* The first combination will introduce a
* timing error because the count value is not
* an integer. All the others have zero error.
* We need a 16-bit counter/timer (TC1) which
* immediately eliminates TP=32 and TP=128
* that are only valid for the 8-bit timer TC2.
*
* We are then limited to 2 combinations:
*  1600000 = 1x256x6250 = 1x64x25000
*
* Let's choose CP=1, TP=256, COUNT=6250 since
* the timer input frequency will be lower
*
*********************************************
* For the mode:
* Mode 0: TC1 starts at a given BOTTOM value,
* counts up to 65535, and overflows to zero
* without stopping.
*
* Mode 2: TC1 starts at a given BOTTOM value,
* counts up to the value in OCR1A and returns
* to zero without stopping
*
```

Timer/Counter – Example 1

```c
 *
 * The other modes are for PWM generation
 *
 * Let’s choose mode NORMAL. Mode CTC is
 * left as an exercise for the student
 *
 *************************************************
 * Created: Oct 12, 2014
 *  Author: jpsousa@fe.up.pt (eclipse+gcc)
 *************************************************/

#include <avr/io.h>
#include <avr/interrupt.h>

#define LED PB5

/* 100ms = 6250 clock cycles @ 16MHz/(1*256)  */
#define T1BOTTOM 65536-6250
```

```c
/*************************************************
 * The main loop is empty since everything
 * is handled by the ISR of Timer 1 which
 * is executed every 100ms
 *************************************************/

void main(void) {
  DDRB |= (1 << LED);  // LED as output
  tc1_init();      // Init Timer 1
  sei();           // Enable global int

  while(1);        // Main loop is empty!
}
```

Timer/Counter – Example 1

```c
/*********************************************
 * Timer 1 initialization in NORMAL mode
 *********************************************
 * - Stop TC1 and clear pending interrupts
 * - Define mode of operation & BOTTOM value
 * - Set the required interrupt mask
 * - Start timer with the proper prescaler
 *********************************************/
void tc1_init(void) {
  TCCR1B = 0;           // Stop TC1
  TIFR1 = (7<<TOV1)     // Clear all
        | (1<<ICF1);    //   pending interrupts
  TCCR1A = 0;           // NORMAL mode
  TCNT1 = T1BOTTOM;     // Load BOTTOM value
  TIMSK1 = (1<<TOIE1); // Enable Ovf intrpt
  TCCR1B = 4;           // Start TC1 (TP=256)
}
```

```c
/*********************************************
 * Timer 1 ISR is executed each 100ms
 *********************************************
 *   - Reload BOTTOM value
 *   - Toggle LED
 *********************************************/
ISR(TIMER1_OVF_vect) {
  TCNT1 = T1BOTTOM;             // reload TC1
  PORTB = PORTB ^ (1<<LED);     // toggle LED
 }
```

Example 1 – Design space exploration

Goal: CP x TP x COUNT = 1.600.000

	Fosc (MHz)	Timer Prescaler (TP)	Values in red only applicable to TC2					
	16							
	TINTR (ms)	001/001	010/010	---/011	011/100	---/101	100/110	101/111
	100	1	8	32	64	128	256	1024
CLK Prescaler (CP)	(2^{0}) 1				25000		6250	1562,5
	(2^{1}) 2				12500		3125	781,25
	(2^{2}) 4		50000		6250		1562,5	390,63
	(2^{3}) 8		25000		3125		781,25	195,31
	(2^{4}) 16		12500		1562,5		390,63	97,66
	(2^{5}) 32	50000	6250		781,25		195,31	48,83
	(2^{6}) 64	25000	3125		390,63	195,31	97,66	24,41
	(2^{7}) 128	12500	1562,5		195,31	97,66	48,83	12,21
	(2^{8}) 256	6250	781,25	195,31	97,66	48,83	24,41	6,1

Error (%)						
1	8	32	64	128	256	1024
0	0	0	0	0	0	0,03
0	0	0	0	0	0	0,03
0	0	0	0	0	0,03	0,16
0	0	0	0	0,03	0,03	0,16
0	0	0	0,03	0,03	0,16	0,68
0	0	0,03	0,03	0,16	0,16	1,7
0	0	0,03	0,16	0,16	0,68	1,68
0	0,03	0,16	0,16	0,68	1,7	1,72
0	0,03	0,16	0,68	1,7	1,68	1,64