ECE3411 – Fall 2016

Lecture 4a.

External Interrupts & Task Based Programming

Marten van Dijk
Department of Electrical & Computer Engineering
University of Connecticut
Email: marten.van_dijk@uconn.edu

Copied from Lecture 4a, ECE3411 – Fall 2015, by
Marten van Dijk and Syed Kamran Haider

Based on the Atmega328P datasheet

UCONN

Example Problem 1

Consider the following code:

```
ISR(TIMER0_COMPA_vect)
{
   if (flag_timer > 0) {flag_timer--;}
   if (flag_timer == 0) {flag = ??; }
}

ISR(INT1_vect)
{
   flag = ??;
   flag_timer = ??;
}
```

Assume ISR(TIMER0_COMPA_vect) is triggered every 1ms. How should the question marks be filled in such that

- as soon as ISR(INT1_vect) is triggered, then flag is set to 1, and
- as soon as 1 second (with approx 1ms precision) has passed since the last time ISR(INT1_vect) was triggered, flag is set to 0.

Example Problem 1

Solution:

```
ISR(TIMER0_COMPA_vect)
{
  if (flag_timer > 0) {flag_timer--;}
  if (flag_timer == 0) {flag = 0; }
}

ISR(INT1_vect)
{
  flag = 1;
  flag_timer = 1000;
}
```

Assume ISR(TIMER0_COMPA_vect) is triggered every 1ms. How should the question marks be filled in such that

- as soon as ISR(INT1_vect) is triggered, then flag is set to 1, and
- as soon as 1 second (with approx 1ms precision) has passed since the last time ISR(INT1_vect) was triggered, flag is set to 0.

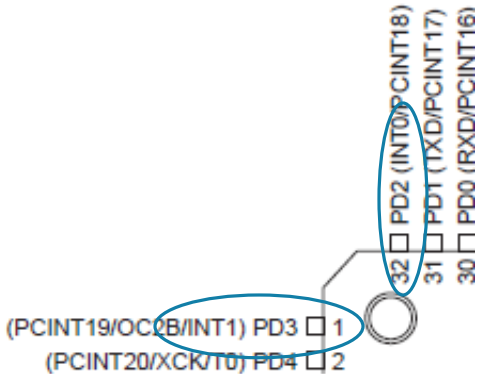
Example Problem 2

- One of your colleagues has already written the code for a task() (which takes about 100 micro seconds) and asks you to write:
- The main code int main(void), which
  - starts by setting registers, enabling interrupts, and executing task() for a first time, and
  - concludes with a while loop which starts executing task() as soon as within a 1 second (with approx 1ms precision) time frame each of the two pins 1 and 32 have signaled a falling edge after the last time task() finished executing

- **Example**: Consider the moment when task() finished executing at for example t=10.0001 seconds.
- Suppose that pin 1 signals falling edges at times t1=13, t1=13.7, and t1=14.3 seconds, pin 32 signals falling edges at times t0 = 12, t0= 13.5, and t_0=14.2 seconds. Assume for the purpose of this example that no other falling edges happen.
- After time t=10.0001 when task() finished executing, the first moment each of the two pins signal a falling edge within a 1 second time frame happens at t=13.5. So, task() should start executing at time t=13.5.
- Suppose it finishes at t=13.5001. After time t=13.5001, the first moment each of the two pins all signal a falling edge within a 1 second time frame happens at t=14.2. So, task() should again start executing at time t=14.2.

Example Problem 2

- Besides the main code you are also required to write the appropriate ISRs and declare variables. Assume the MCU runs at 20MHz. You can use the next two pages to write your code.
- **Hint**: As in problem 1, program a flag0 and a flag1 for each of the two pins: As soon as they sum up to 2, each pin triggered an ISR within the last 1 second timeframe.



Example Problem 2

```
// Put the declaration of your global variables here:
#define t_flag 1000 // 1000ms = 1 second

volatile int flag0, flag1;
volatile int flag0_timer, flag1_timer;

ISR(TIMER0_COMPA_vect)
{
  // Put your code here:
  if (flag0_timer > 0) {flag0_timer--;}
  if (flag0_timer == 0) {flag0 = 0; }

  if (flag1_timer > 0) {flag1_timer--;}
  if (flag1_timer == 0) {flag1 = 0; }
}
```

```
ISR(INT0_vect)
{
  // Put the code of your second ISR here:
  flag0 = 1;
  flag0_timer = t_flag;
}

ISR(INT1_vect)
{
  // Put the code of your third ISR here:
  flag1 = 1;
  flag1_timer = t_flag;
}
```

Example Problem 2

```
int main(void)
{
  // Put your code of the main body (including initializations) here:

  // An accurate 1ms timer (as explained in class):
  TIMSK0 = 2; // enable interrupt
  TCCR0A = 0x02; // return on clear-on-match
  TCCR0B = 0x02; // prescalar @ 8
  OCR0A = 249; // each time tick is 8(OCR0A+1)/20MHz = 1ms exactly
  // An accurate enough timer is needed otherwise the flag_timers drift with respect
  // to real time and we may not meet the specification of ~1ms precision.

  // Initialize external interrupts INT0 (= pin 32) and INT1 (= pin 1) on falling edges
  DDRD = 0x00; // D.2 = pin 32 and D.3 = pin 1 are inputs
  EICRA = (1<<ISC01) | (1<<ISC11);
  EIMSK = (1<<INT0) | (1<<INT1);

  task_timer = t_task;
  flag0 = 0;
  flag1 = 0;

  // Globally enable interrupts
  sei();
```

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	

Table 12-1. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	

Example Problem 2

```
// Execute task() before entering the while loop
// This allows us to formally meet the specifications of the program
task();

while (1)
{
  if (flag0 + flag1 == 2) // See hint
  {
    task();
    // All flags should be reset, since we just finished executing task()
    flag0 = 0;
    flag1 = 0;
  }
}

return 0;
}
```

Example Problem 3

- One of your colleagues has already written the code for two tasks task1() and task2() (each taking only about 100 micro seconds) and asks you to write:
  The main code int main(void), which
  - starts by setting registers and enabling interrupts, and
  - concludes with a while loop which
    - starts executing task1() every 1 millisecond (as accurate as possible), and
    - starts executing task2() as soon as
      1. a rising edge is received over pin 1 since the last time task2() finished executing and
      2. at least 1 second has passed since the last time task2() finished executing.

    Example: task2() finished executing at time t=0.

    (a) If the next rising edge after t=0 is detected at e.g. time t=300 milliseconds, then the while loop waits another 700 milliseconds (such that 1 full second has passed) before it starts executing task2().

    (b) If the next rising edge after t=0 is detected at e.g. time t=1100 milliseconds, then the while loop immediately starts executing task2() (since 1 full second has already passed).

- Besides the main code you are also required to write the appropriate ISRs. Assume the MCU runs at 20MHz.

Example Problem 3

```
// Put the declaration of your global variables here:
// define t1 as 1ms and t2 as 1000ms = 1 second
#define t1 1
#define t2 1000

volatile int task1_timer;
volatile int task2_timer;
volatile int flag;

ISR(TIMER0_COMPA_vect)
{
  // Put the code of your first ISR here:
  // Set up virtual timers
  if (task1_timer > 0) {task1_timer--;}
  if (task2_timer > 0) {task2_timer--;}
}

ISR(INT1_vect)
{
  // Put the code of your second ISR here:
  // Set flag
  flag = 1;
}
```

Example Problem 3

```
int main(void)
{
  // Put the code the main body here:

  // An accurate 1ms timer (as explained in class):
  TIMSK0 = 2; // enable interrupt
  TCCR0A = 0x02; // return on clear-on-match
  TCCR0B = 0x02; // prescalar @ 8
  OCR0A = 249; // each time tick is 8(OCR0A+1)/20MHz = 1ms exactly

  // Initialize external interrupt INT1 (= pin 1) on rising edge
  DDRD = 0x00; // D.3 = pin 1 is an input
  EICRA = (1<<ISC11) | (1<<ISC10);
  EIMSK = (1<<INT1);

  task1_timer = t1;
  task2_timer = t2;
  flag = 0;

  // Globally enable interrupts
  sei();
```

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	

Table 12-1. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	

Example Problem 3

```
while (1)
  {
    if (task1_timer == 0)
    {
      task1_timer = t1; // Before calling task1(), otherwise task1() is called
                        // every 1.1 ms where the 100 micro second delay comes
                        // from the execution of task1()
      task1();
    }

    if (task2_timer == 0) && (flag == 1)
    {
      task2();
      flag = 0;   // As soon as task2() is finished we want to be able
                  // to detect external interrupts
      task2_timer == t2; // After calling task2() since we measure the time that
                         // passes since task2() has finished executing for the
                         // last time
    }
  }

  return 0;
}
```