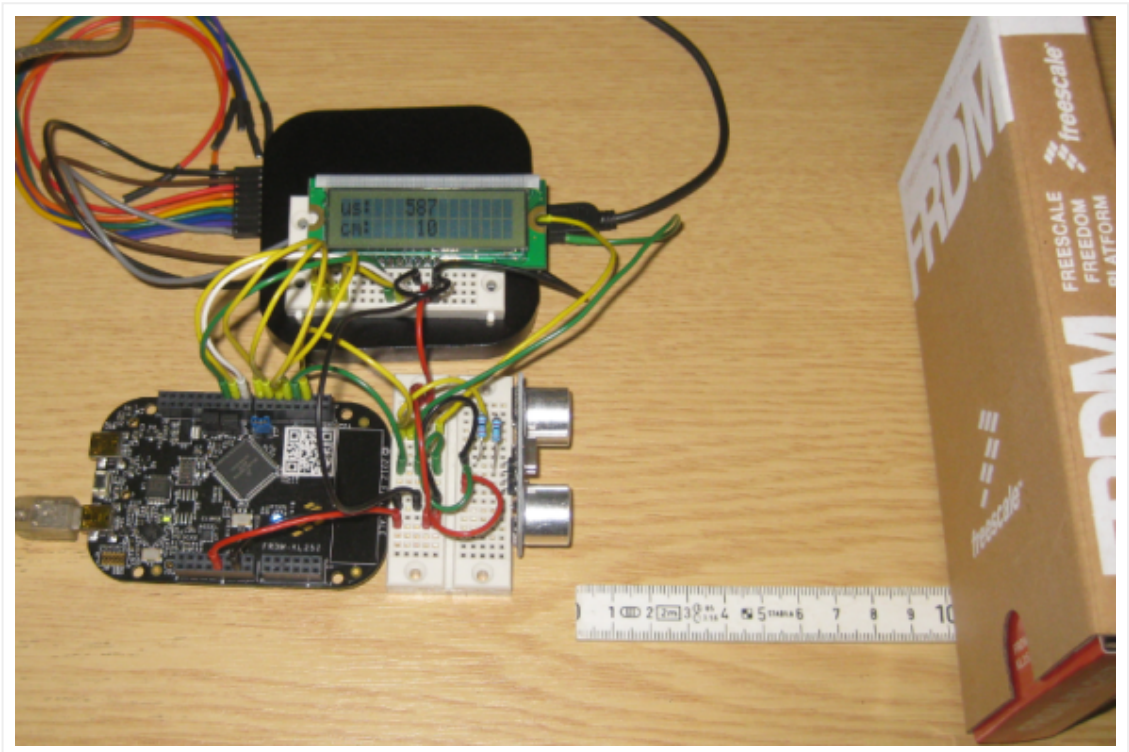
Tutorial: Ultrasonic Ranging with the Freedom Board

Posted on **January 1, 2013** by **Erich Styger**

13 Votes

Question: What makes 8 times 'beep', but I cannot hear it?

Answer: My ultrasonic range finder🙂

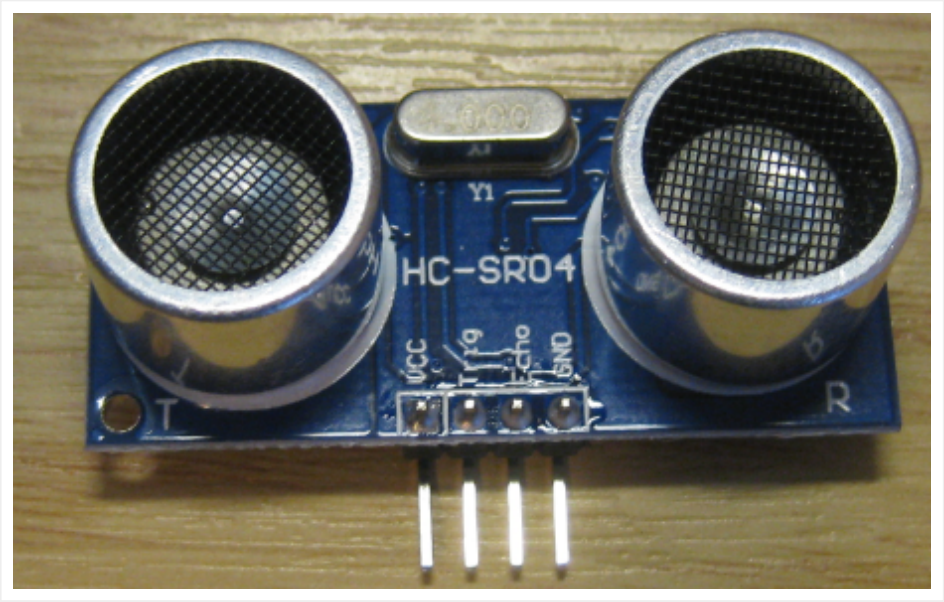


— FRDM-KL25Z with HC-SR04

What I have added to my FRDM-KL25Z board is an ultrasonic distance sensor, measuring distances up to 4 meters.

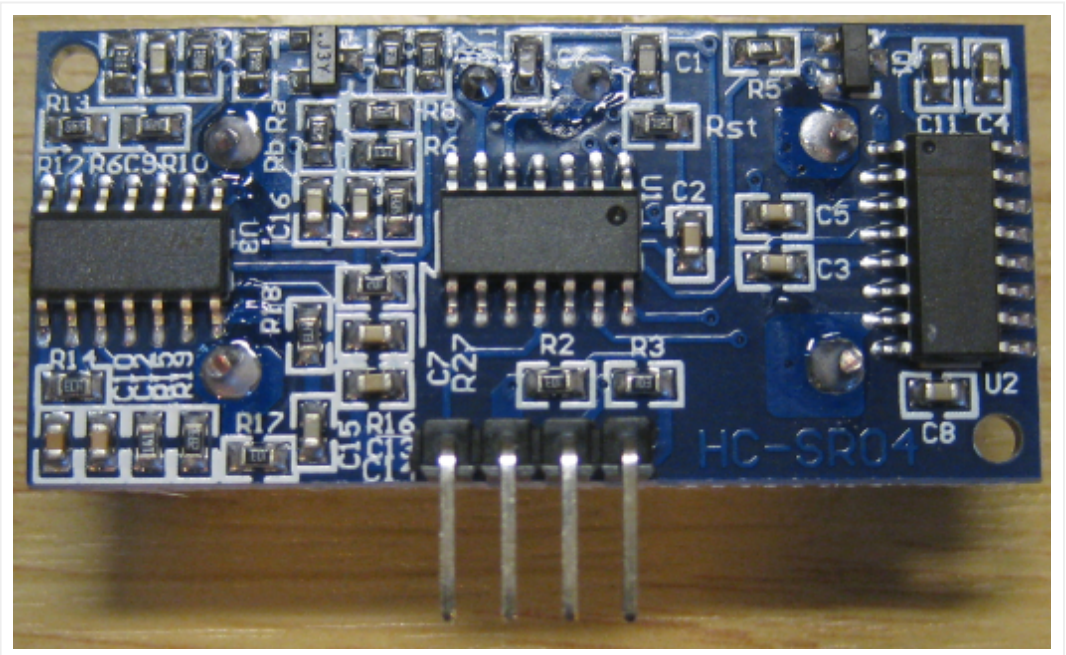
HC-SR04 Ultrasonic Sensor

The HC-SR04 sensor is a 4 pin sensor, at an incredible price point. I have mine from Play-Zone, but I have seen offers in the internet for around $6 too.



— HC-SR04 Front Side

The backside features all the electronics to which simply usage of the sensor:



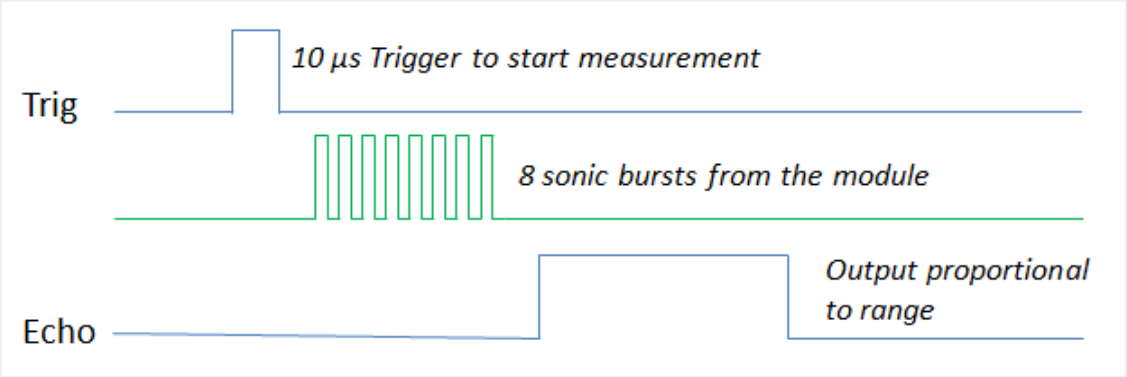
— HC-SR04 Back Side

With simple I mean, it only needs 4 pins:

1. Vcc (+5V DC supply)
2. Trig (TTL input, to trigger a measurement)
3. Echo (TTL output, pulse proportional to distance)
4. GND (ground)

Note: There is a similar module on the market: the Parallax Ping))) or Seedstudio one which only use 3 pins (Vcc, Trig+Echo, GND). While this saves a pin, it makes usage a big more complicated as I would have to switch between output and input on a single pin. So I prefer the 4 pin variant.

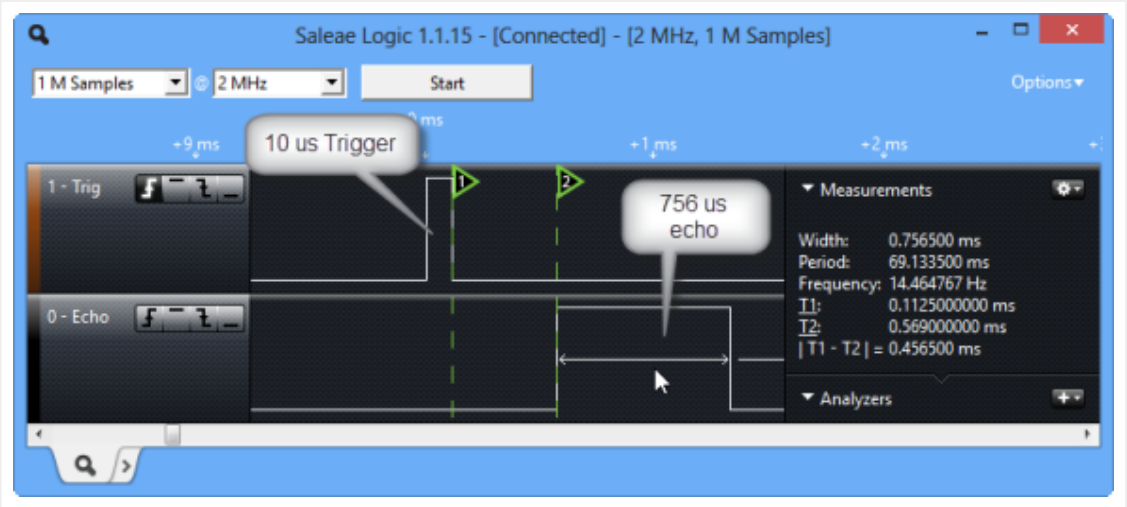
Using the sensor is simple:



— HC-SR04 Timing Diagram

1. Send a 10 us pulse on Trig
2. In reaction to this, the sensor will send 8 sonic bursts to measure the distance
3. After sending the burst, the sensor will raise the Echo signal until it receives the echo back. That means the length of the Echo signal corresponds to time the burst was travelling forward and echoed back.

Below is a snapshot with a logic analyzer:



— HC-SR04 Timing

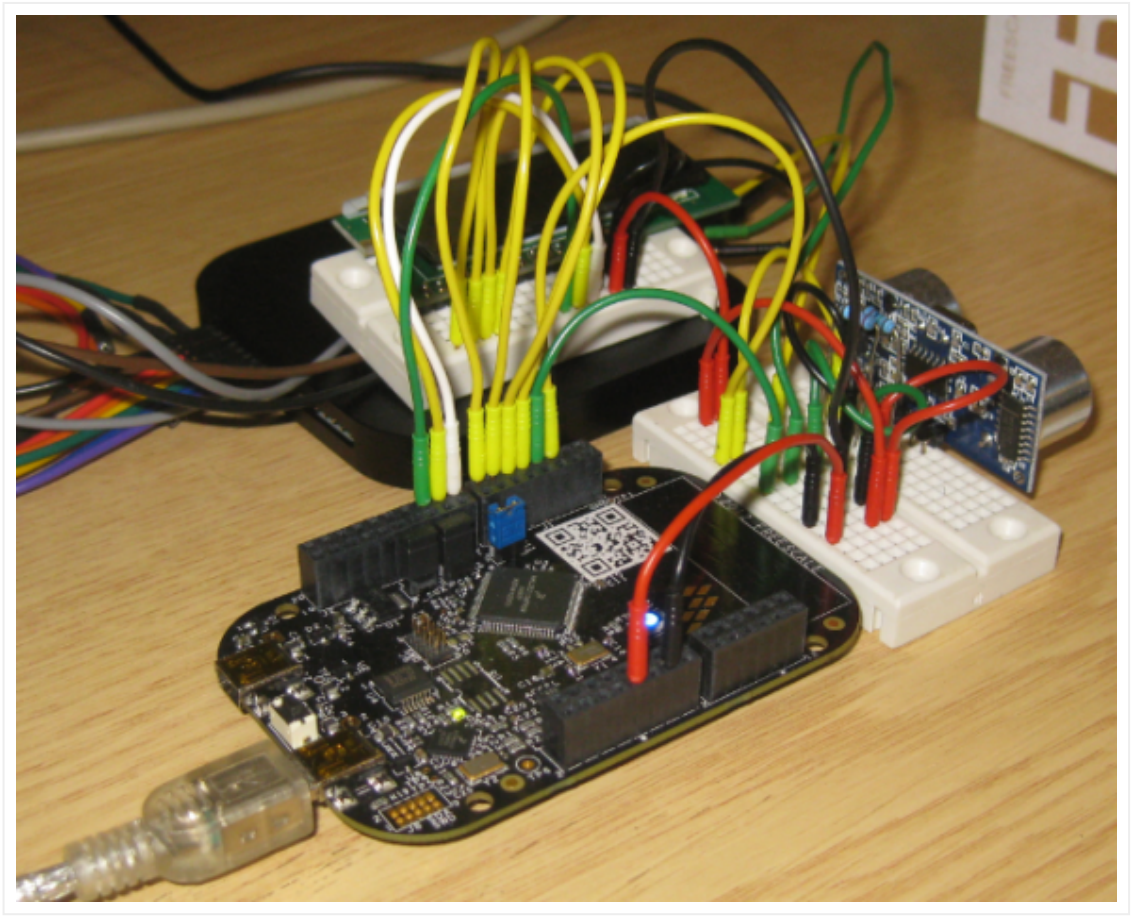
After the 10 us Trigger signal, the sensor sends the burst during '1' and '2' (around 456 us), followed by the Echo signal duration of 756 us. The 756 us corresponds to the time it took for the burst to come back.

The speed of sound is about 340 m per second (depends on temperature and air relative humidity). A good approximation is 29 us per cm (or 58 us per centimeter as it is for back and forward). With the above example, the 756 us would result in 756/29/2 = 13 cm distance.

💡 *For decimal system challenged engineers: instead of a divider of 58, use a divider of 148 to get the result in inch instead of centimeter.*

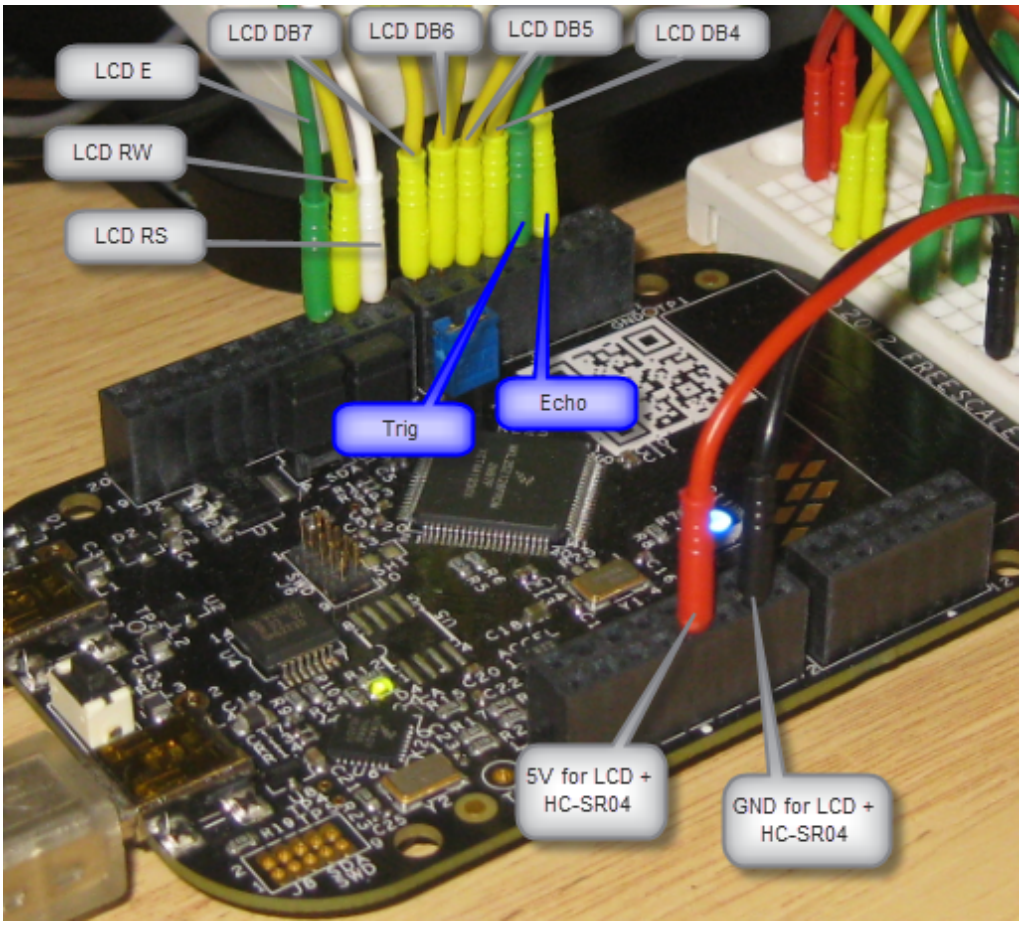
Hardware Connection

I'm using a breadboard for easy wiring and connection.



— Breadboard Wiring with Freedom Board, Sensor, LCD and Logic analyzer

To visualize the measurement, I'm using the 2×16 Character display I have used in my earlier post. Of course a LCD is optional. I used the same wiring as before:

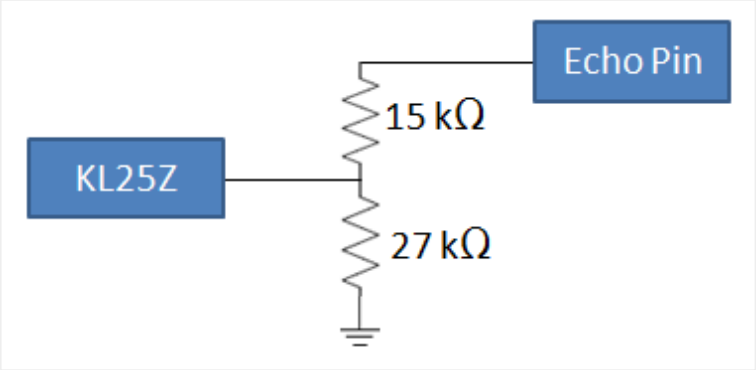


— Signals with Labels

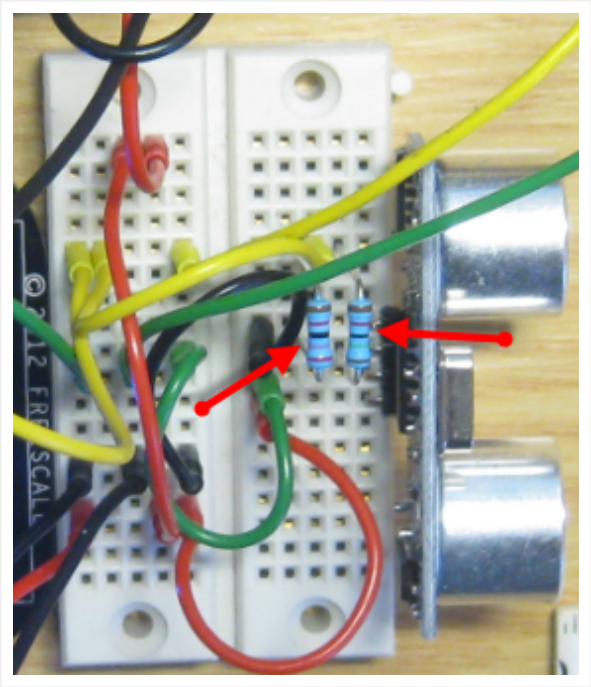
```
================================================================
 SIGNAL LIST
----------------------------------------------------------------
SIGNAL-NAME [DIR]         => PIN-NAME [PIN-NUMBER]
----------------------------------------------------------------
DB4_D4 [I/O]              => TSI0_CH5/PTA4/I2C1_SDA/TPM0_CH1/NMI_b [30]
DB5_D5 [I/O]              => PTA5/USB_CLKIN/TPM0_CH2 [31]
DB6_D6 [I/O]              => CMP0_IN2/PTC8/I2C0_SCL/TPM0_CH4 [65]
DB7_D7 [I/O]              => CMP0_IN3/PTC9/I2C0_SDA/TPM0_CH5 [66]
E_D10 [Output]            => PTD0/SPI0_PCS0/TPM0_CH0 [73]
LED_BLUE [Output]         => ADC0_SE5b/PTD1/SPI0_SCK/TPM0_CH1 [74]
LED_GREEN [Output]        => TSI0_CH12/PTB19/TPM2_CH1 [54]
LED_RED [Output]          => TSI0_CH11/PTB18/TPM2_CH0 [53]
RS_D8 [Output]            => PTA13/TPM1_CH1 [33]
RW_D9 [Output]            => ADC0_SE6b/PTD5/SPI1_SCK/UART2_TX/TPM0_CH5
US_Echo_D2 [Input]        => PTD4/LLWU_P14/SPI1_PCS0/UART2_RX/TPM0_CH4
US_Trig_D3 [Output]       => PTA12/TPM1_CH0 [32]
================================================================
```

What I need is the supply voltage, ground, an output pin (Trig) and an input pin to measure the signal (Echo).

The HC-SR04 uses TTL (5V) supply voltage and logic levels. The FRDM-KL25Z processor is a 3.3V one, but the board provides 5V on the header. The HC-SR04 can use 3.3V levels on the Trig signal, but provides a 5V signal on the Echo pin. To get the signal to the 3.3V level, I use a simple voltage divider with a 27k Ohm and 15k Ohm resistor:



— Voltage Divider with two resistors



— Voltage Divider Detail

CodeWarrior Project

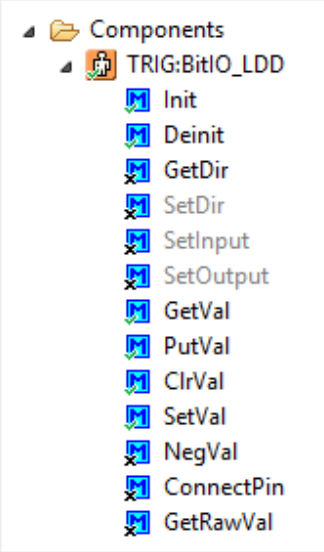
With the wizard (File > New Bareboard Project) I create a new project for the KL25Z and Processor Expert. I need two things:

1. Create a 10 us pulse on the Trig (Trigger) pin

2. Measure the echo on the Echo pin

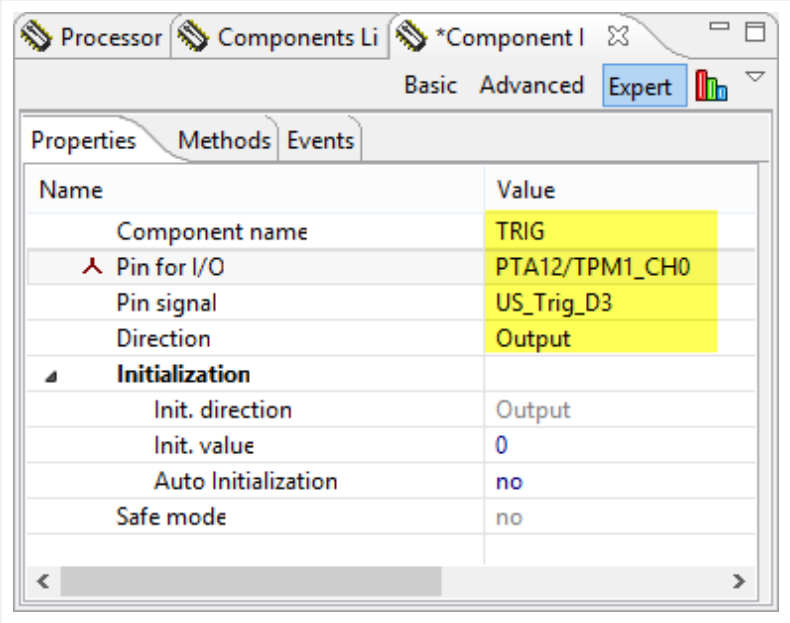
Trigger

For the Trigger pin I add a **BitIO_LDD** component to my project:



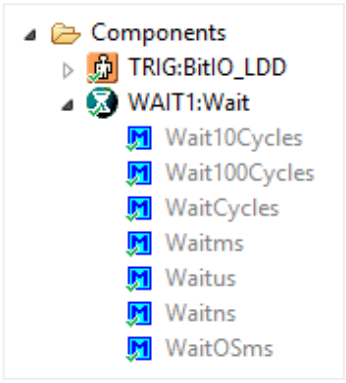
— BitIO_LDD Component

I configure it as Output pin on PTA12:



— Trigger Properties

Additionally I add the Wait component to the project:



— Wait Component

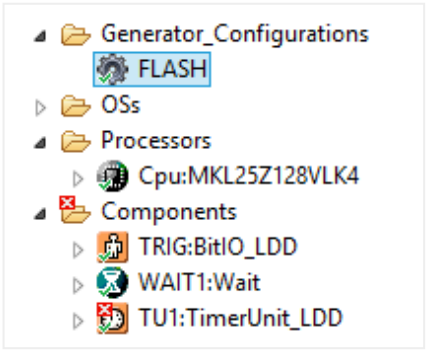
Generating the 10 us pulse now is trivial:

```
1 TRIG_SetVal(trigDevice);
2 WAIT1_Waitus(10);
3 TRIG_ClrVal(trigDevice);
```

Wait! There is some more infrastructure needed, such as this **trigDevice** handle. More on this below.

Echo Pulse Measurement

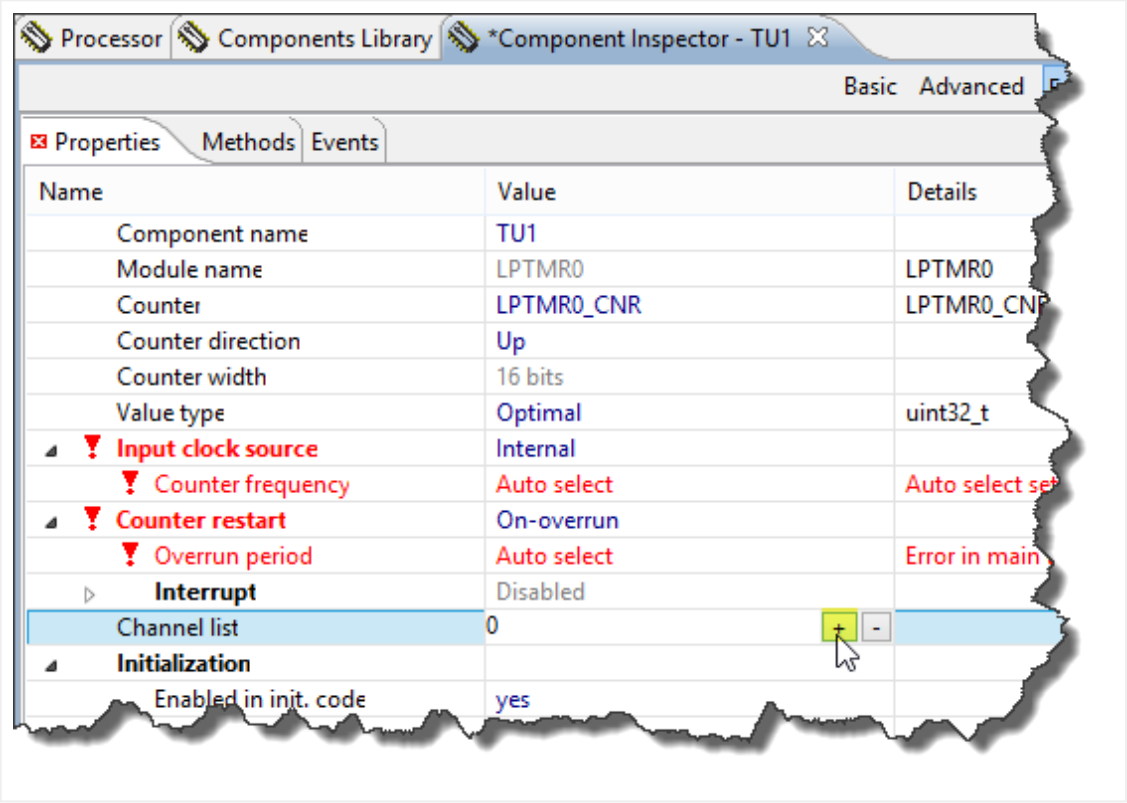
To measure the echo, I add a **TimerUnit_LDD** component:



— TimerUnit_LDD added

That component shows errors for now, because I need to configure it for my needs. What I want is a timer measuring the echo signal. For this I need to set up a timer which starts counting on the echo raising edge, and stops at the falling edge. For this I need a timer channel.

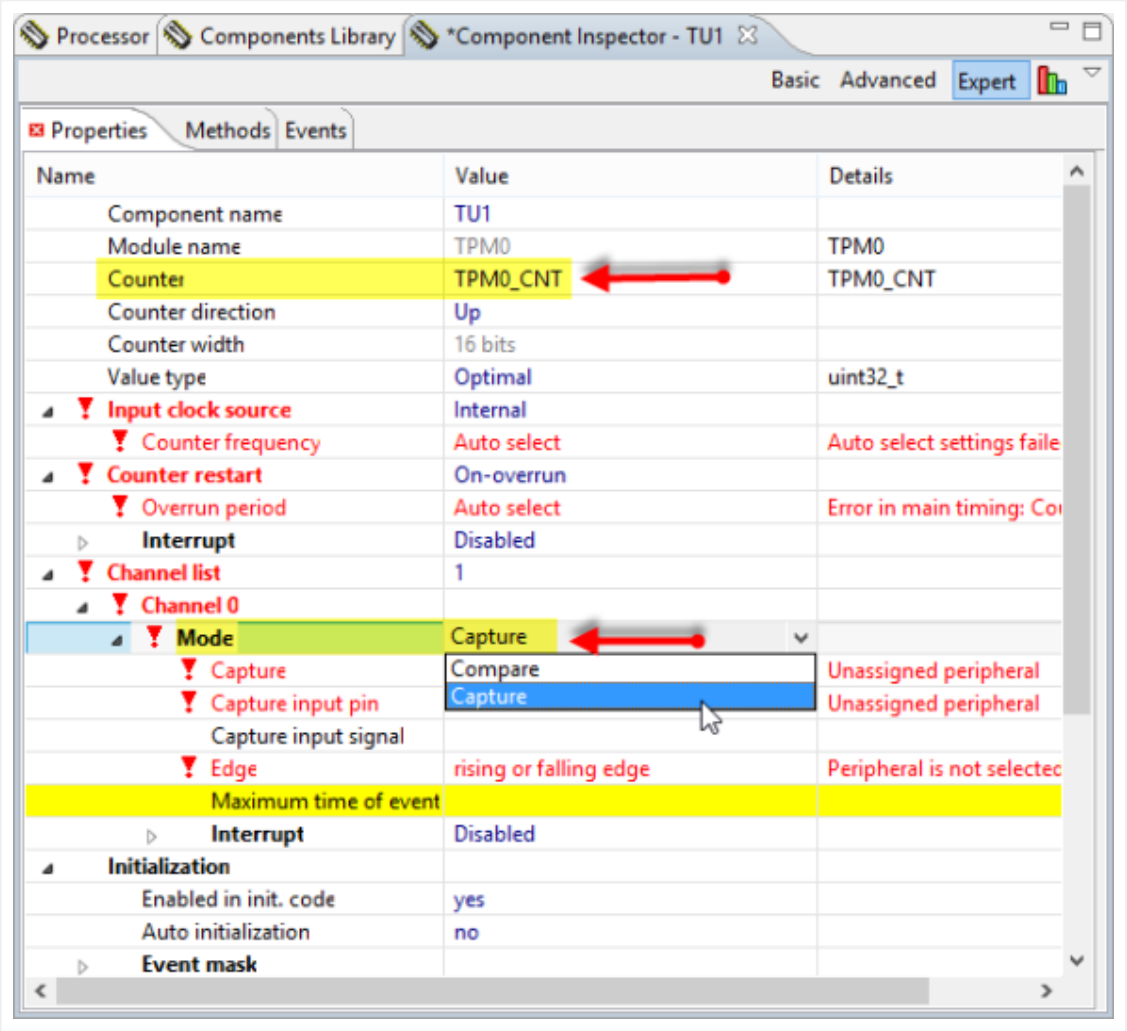
I add a new channel using the '+' icon:



— Adding Timer Unit Channel

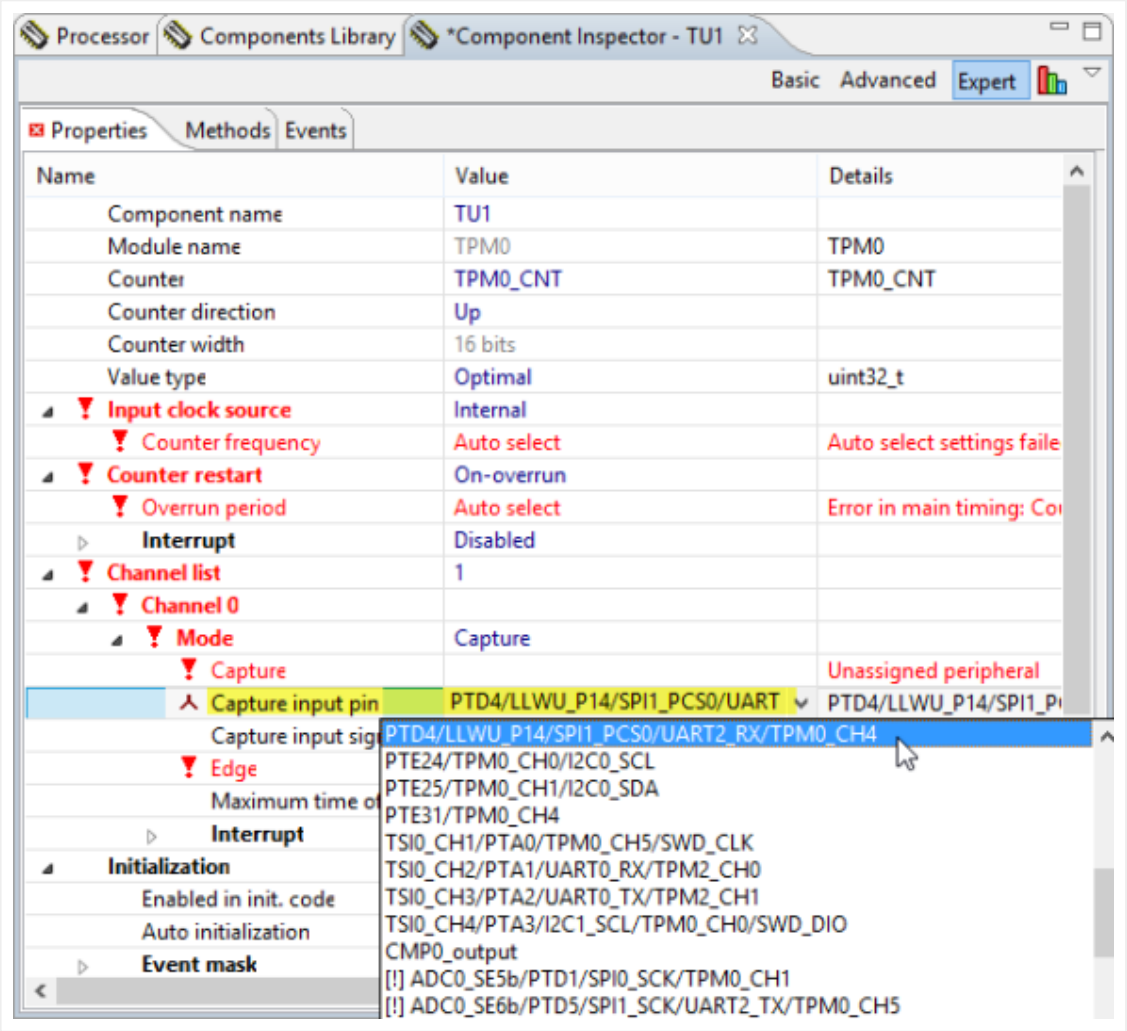
For this I'm selecting the **TPM0_CNT** as counter and configure the channel for 'capture':

You will understand later on why I have selected this particular counter.



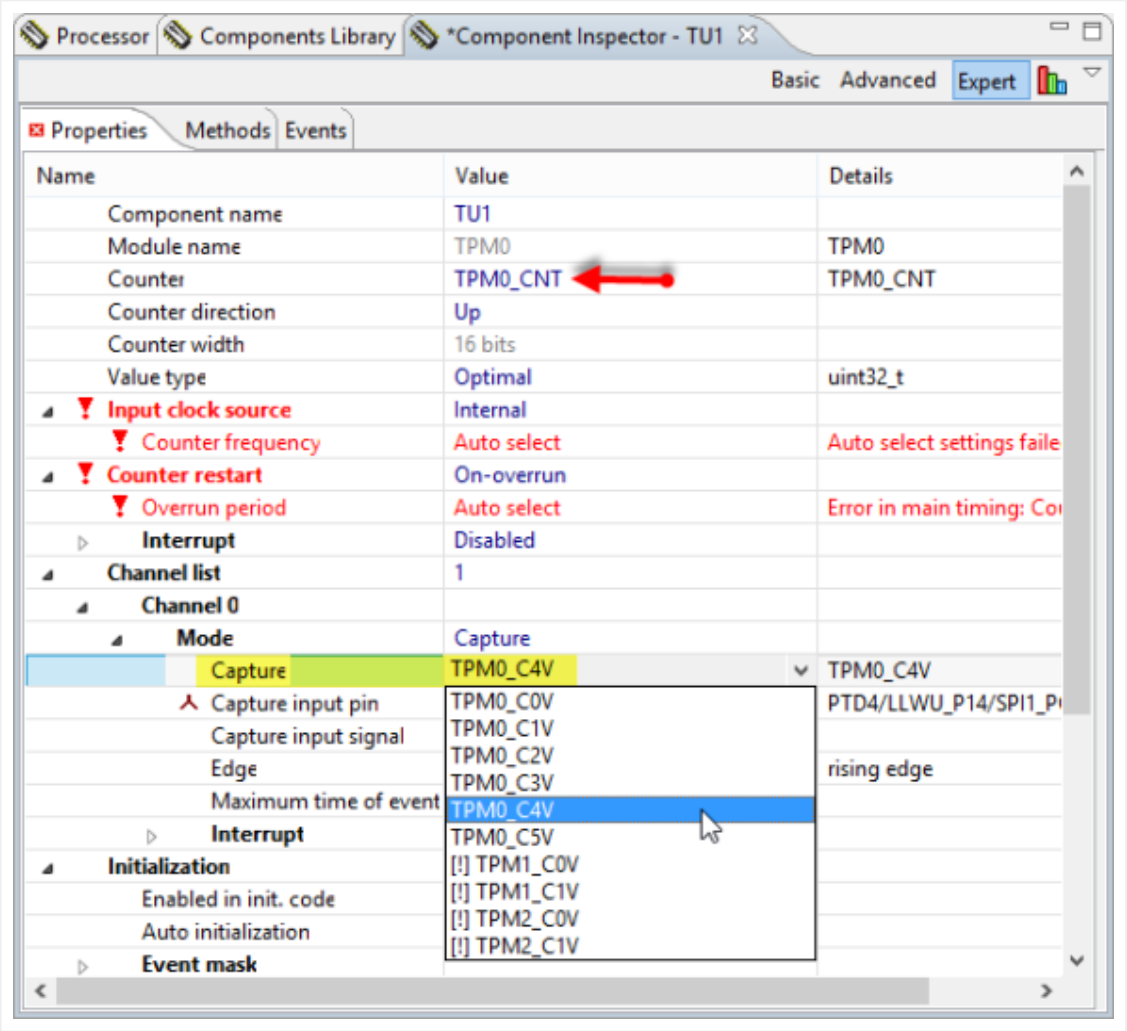
— Using TPM0_CNT and Capture Mode

As I have connected the Echo signal on **PTD4**, I need to use that pin as 'Capture input pin':



— PTD4 as input capture pin

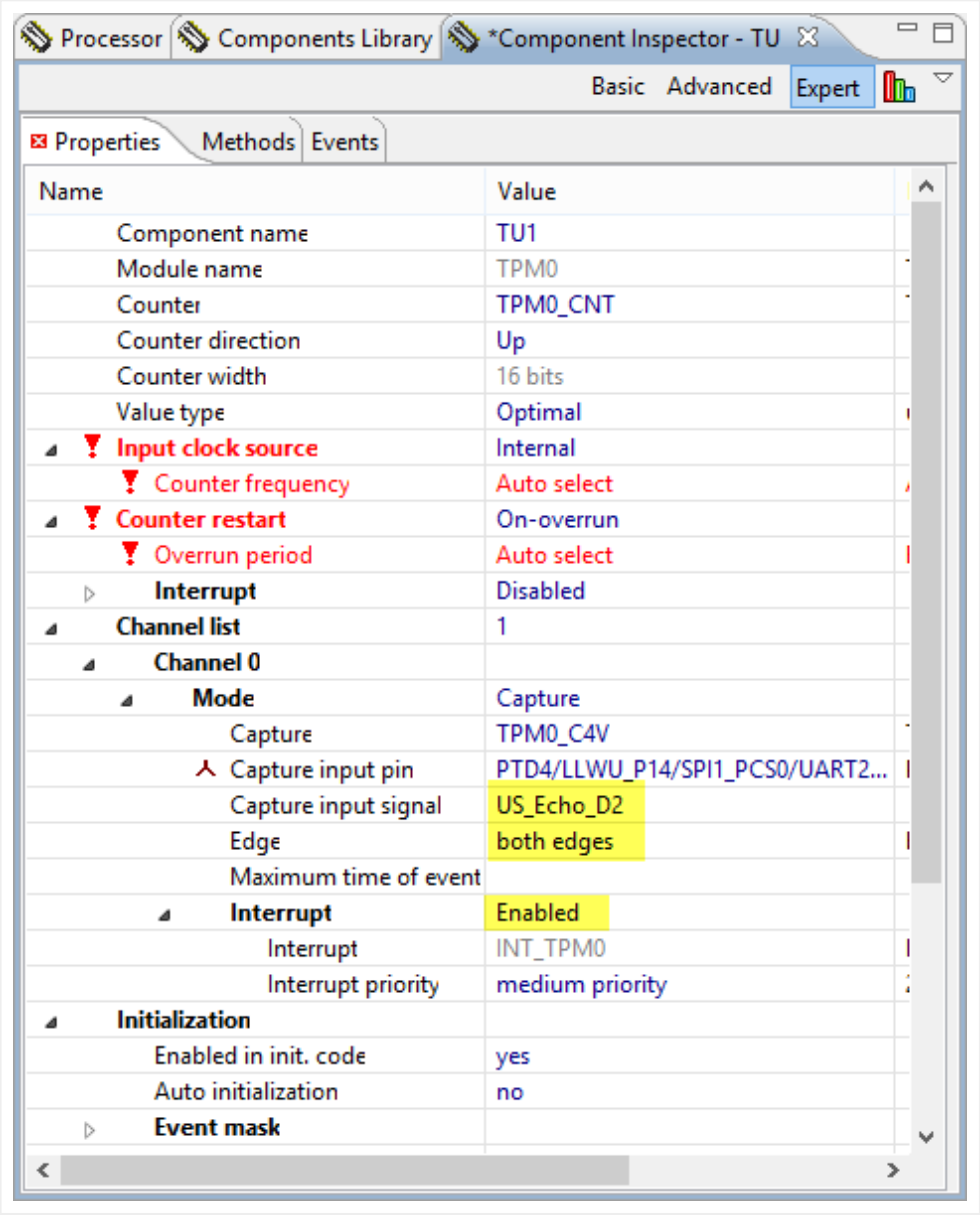
As hinted by the pin name (PTD4/LLWU_P14/SPI1_PCS0/UART2_RX/**TPM0_CH4**), this pin is available on **TPM0_CH4**, so I need to configure this as capture device:



— selected capture device

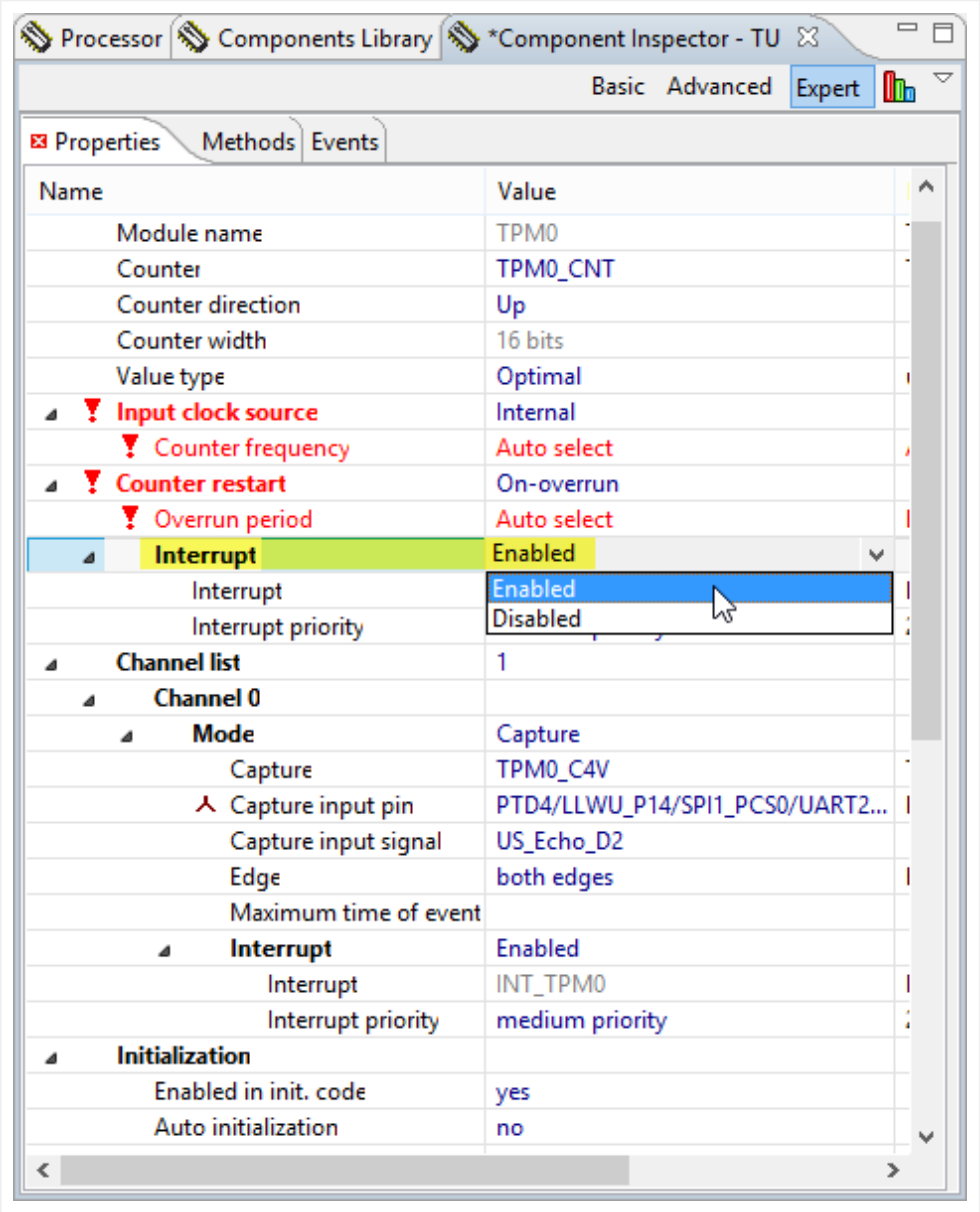
💡 *Now you see why I have selected TPM0_CNT above as Counter device. I admit: that's not something easy to know about, because this KL25Z has mirades of pin and mapping configuration (called Pin Muxing)😟. One way is to guess what is behind from the pin names (this is what I try first). Otherwise you need to read through the device reference manual (which is very time-consuming). I wish these devices would be much less complicated and easier to use....*

I want to get an interrupt for both the raising edge and falling edge of the Echo signal: I enable interrupts and configure it for both edges. And I give the signal a nice name:



— Interrupt Settings

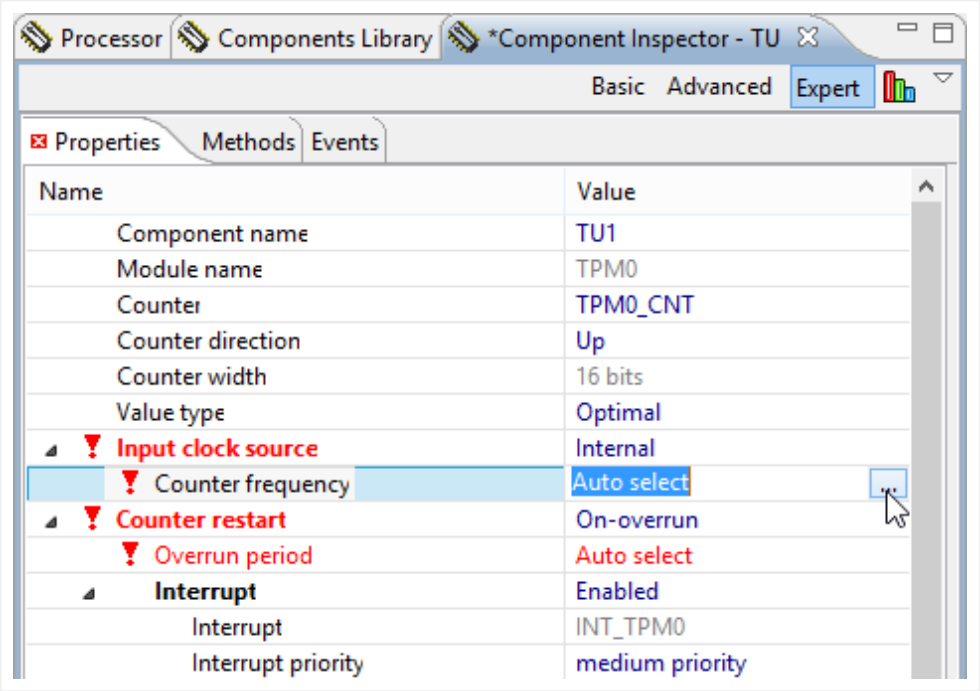
That's not enough: I need to enable interrupts for the counter too:



— Enabled Interrupts for Counter

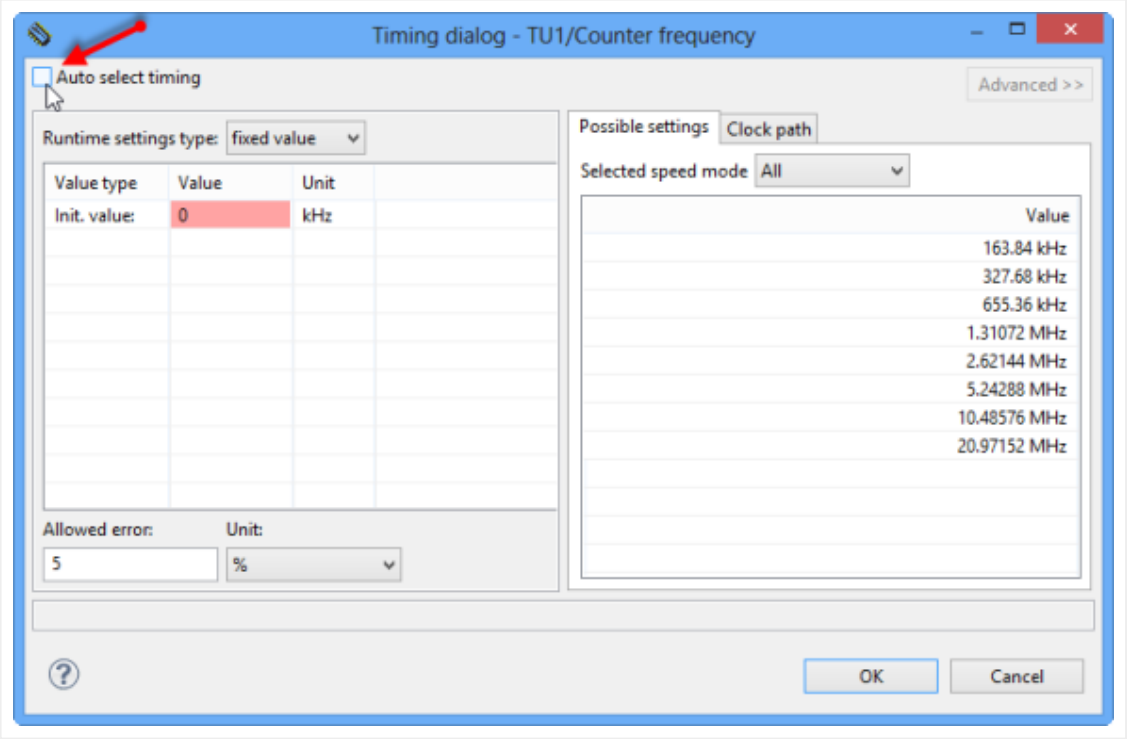
Counter Frequency

I still have errors showing up for my Timer/Counter: I need to configure the counter frequency: This is the clock which is used for my counter. In principle: the higher the frequency, the more precise the measurement will be. As always: a good balance is required. First, I configure the counter frequency:



— Configuring the counter frequency

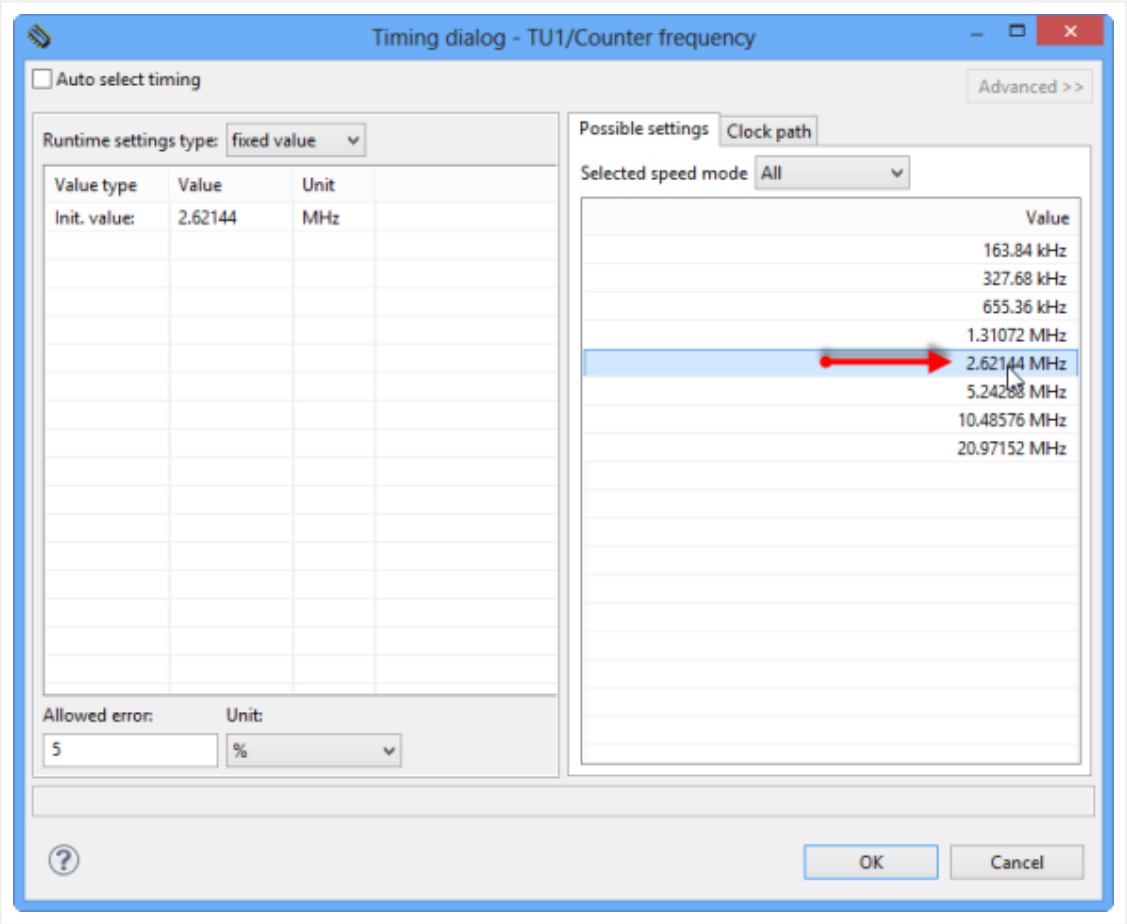
The first thing to do is to disable 'Auto select timing':



— Disabling Auto Select timing

💡 *Processor Expert usually chooses good default values, but fails (for whatever reason?) to do this properly for timers on the ARM platform. So as solution I always disable auto-select-timing and explicitly make my choice. Maybe better that way anyway*🙂

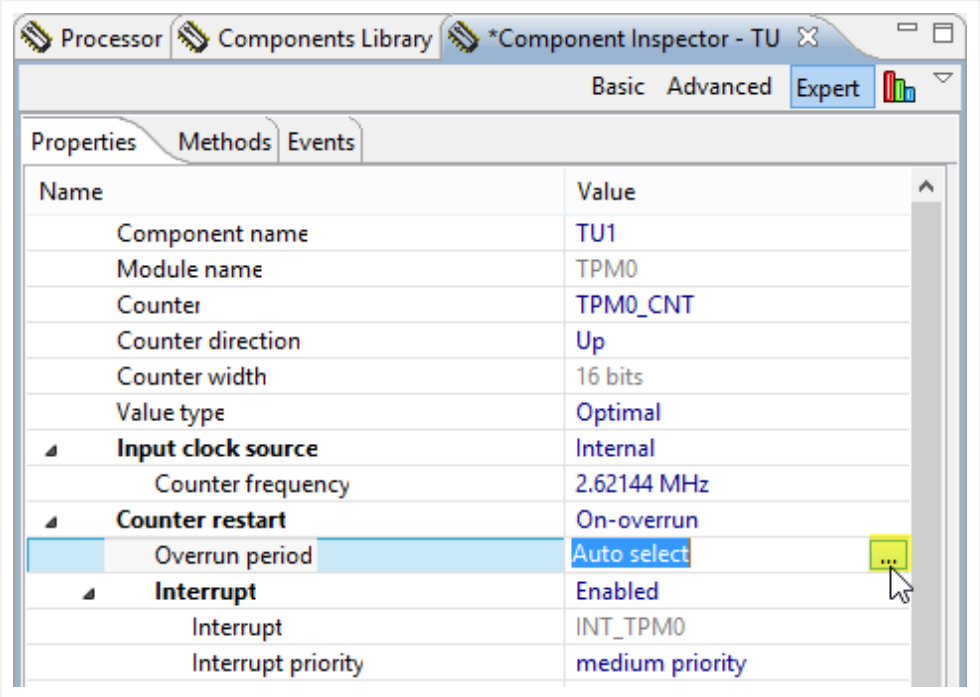
I make a gut guess and select a 2.x MHz clock from the list (double-click to assign it):



— Selected clock frequency

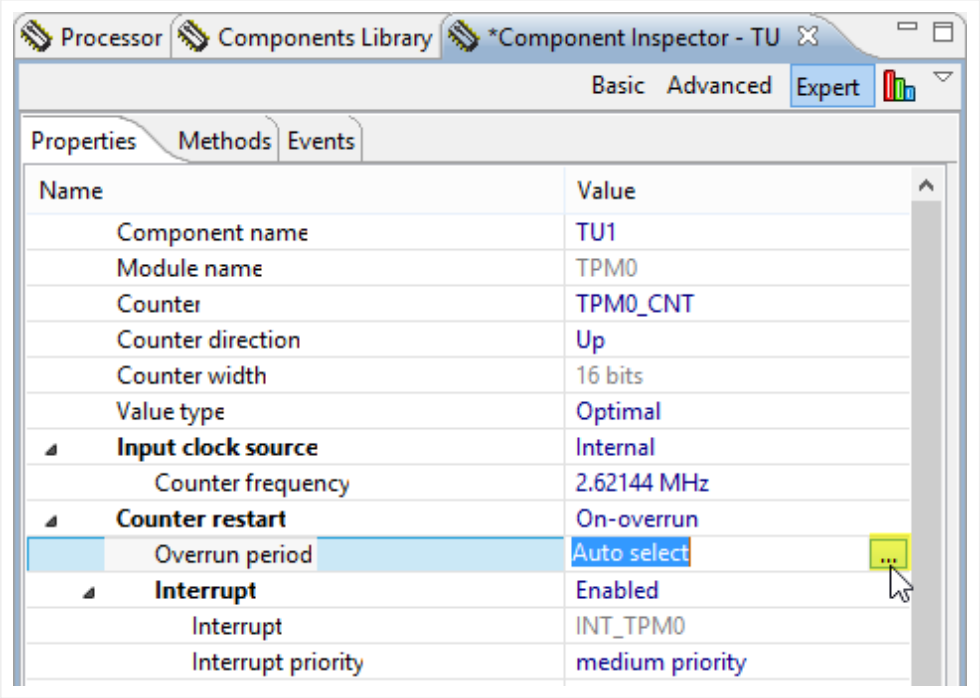
Counter Overrun

There one thing to consider: when will the counter overflow? I keep in mind that with the speed of sound, around 4 meters distance means 400*58us makes 23.5 ms Echo signal pulse. So I need to make sure that for this distance, my counter does *not* overflow. I can check this in the 'Overrun period' settings:



— Overrun period

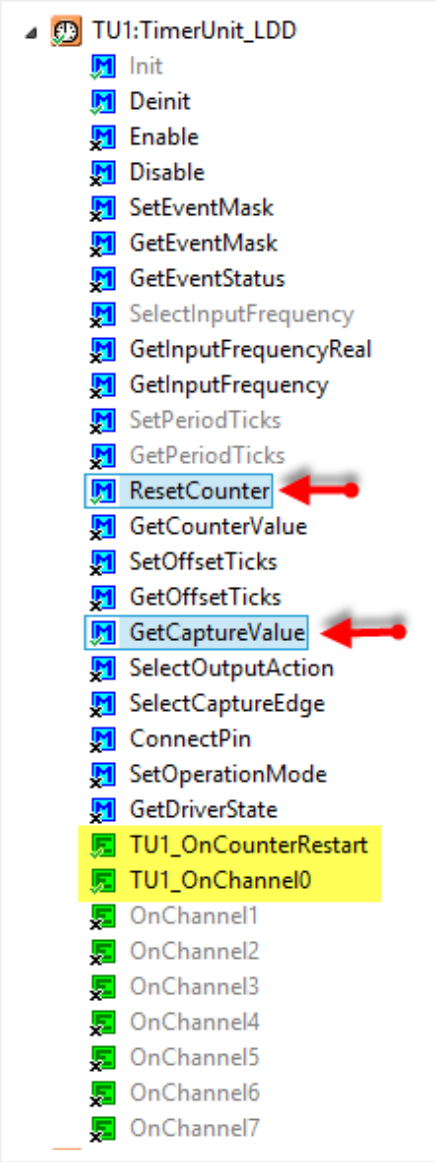
Deselecting ‘auto select timing’ again will show the overrun period time:



— Overrun period

So my timer will overrun after 25 ms, so I’m fine🙂

Next, I need to enable two methods I want to use. I need to read the counter value, and I need to reset it. As such, I enable the two methods (right-click on the method name and select 'Toggle Enable/Disable'):



— Enabled TimerUnit_LDD Methods

I notice the two events called:

- **OnCounterRestart()** is called when there is a restart/overflow of the counter
- **OnChannel()** is called for every interrupt of my channel. Remember that I have it configured for creating an interrupt/event for both raising and falling edge.

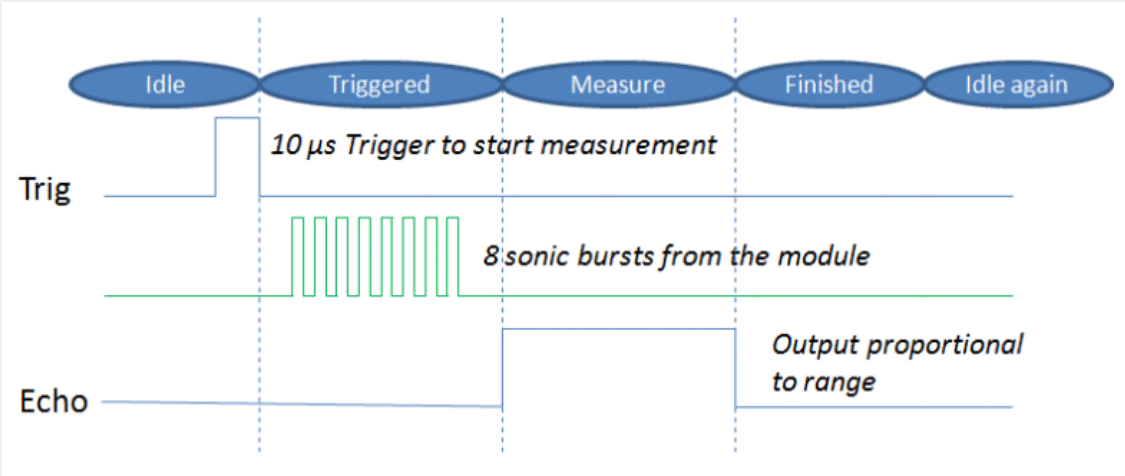
State Machine

I'm using a state machine to go through the different measurement phases:

— State Diagram

- **Idle**: Sensor is idle
- **Triggered**: we sent the trigger signal to the sensor
- **Measure**: we received raising edge of echo signal and are measuring
- **Overflow**: Counter overflow happened (see above)
- **Finished**: received falling echo signal edge and we have captured the echo signal time

Another way is to show the different states on the timing diagram (without the Overflow case):



— Timing and State Diagram

With this in mind, the implementation is pretty straight forward. First, an enumeration for the state machine states:

```
1 typedef enum {
```

```
  ECHO_IDLE, /* device not used */
  ECHO_TRIGGERED, /* started trigger pulse */
  ECHO_MEASURE, /* measuring echo pulse */
  ECHO_OVERFLOW, /* measurement took too long */
  ECHO_FINISHED /* measurement finished */
} US_EchoState;
```

Next the data structure to keep all my data:

```
typedef struct {
  LDD_TDeviceData *trigDevice; /* device handle for the Trigger pin */
  LDD_TDeviceData *echoDevice; /* input capture device handle (echo pin) *
  volatile US_EchoState state; /* state */
  TU1_TValueType capture; /* input capture value */
} US_DeviceType;

static US_DeviceType usDevice; /* device handle for the ultrasonic device
```

And here is how it gets initialized:

```
void US_Init(void) {
  usDevice.state = ECHO_IDLE;
  usDevice.capture = 0;
  usDevice.trigDevice = TRIG_Init(NULL);
  usDevice.echoDevice = TU1_Init(&usDevice);
}
```

Event Handlers

Notice the call to `TU1_Init()` where I pass a pointer to my data structure: That way I get a nice handle passed from the Processor Expert event routines (in events.c) I can use for my own routines.
Remember that I get interrupts for raising and falling edge, and for overflow.
I handle that in Events.c:

```
/*
** ===================================================================
**     Event       :  TU1_OnChannel0 (module Events)
**
**     Component   :  TU1 [TimerUnit_LDD]
**     Description :
**         Called if compare register match the counter registers or
**         capture register has a new content. OnChannel0 event and
**         Timer unit must be enabled. See  and
**          methods. This event is available only if a
**          is enabled.
**     Parameters  :
**         NAME            - DESCRIPTION
**       * UserDataPtr     - Pointer to the user or
**                           RTOS specific data. The pointer passed as
**                           the parameter of Init method.
**     Returns     : Nothing
** ===================================================================
*/
void TU1_OnChannel0(LDD_TUserData *UserDataPtr)
{
  US_EventEchoCapture(UserDataPtr);
}

/*
** ===================================================================
**     Event       :  TU1_OnCounterRestart (module Events)
**
```

```
**     Component   :  TU1 [TimerUnit_LDD]
**     Description :
**         Called if counter overflow/underflow or counter is
**         reinitialized by modulo or compare register matching.
**         OnCounterRestart event and Timer unit must be enabled. See
**          and  methods. This event is
**         available only if a  is enabled.
**     Parameters  :
**         NAME            - DESCRIPTION
**       * UserDataPtr     - Pointer to the user or
**                           RTOS specific data. The pointer passed as
**                           the parameter of Init method.
**     Returns     : Nothing
** ===================================================================
*/
void TU1_OnCounterRestart(LDD_TUserData *UserDataPtr)
{
  US_EventEchoOverflow(UserDataPtr);
}
```

In case of Overflow I simply set the state machine to the overflow state:

```
void US_EventEchoOverflow(LDD_TUserData *UserDataPtr) {
  US_DeviceType *ptr = (US_DeviceType*)UserDataPtr;

  ptr->state = ECHO_OVERFLOW;
}
```

While in the interrupt/event for raising or falling edge, I reset the counter value at raising edge, or read out the counter value at falling edge:

```
US_DeviceType *ptr = (US_DeviceType*)UserDataPtr;

if (ptr->state==ECHO_TRIGGERED) { /* 1st edge, this is the raising edge, s
  TU1_ResetCounter(ptr->echoDevice);
  ptr->state = ECHO_MEASURE;
} else if (ptr->state==ECHO_MEASURE) { /* 2nd edge, this is the falling ec
  (void)TU1_GetCaptureValue(ptr->echoDevice, 0, &ptr->capture);
  ptr->state = ECHO_FINISHED;
}
```

With this I reset the counter on the raising edge, and get the counter value at the falling edge. Then I set the state to 'finished': this will be the state I wait for (polling) we will see later.

Measure!

I have implemented a function which sends the trigger, and then waits until the measurement has been finished:

```
uint16_t US_Measure_us(void) {
  uint16_t us;

  /* send 10us pulse on TRIG line. */
  TRIG_SetVal(usDevice.trigDevice);
  WAIT1_Waitus(10);
  usDevice.state = ECHO_TRIGGERED;
  TRIG_ClrVal(usDevice.trigDevice);
  while(usDevice.state!=ECHO_FINISHED) {
    /* measure echo pulse */
    if (usDevice.state==ECHO_OVERFLOW) { /* measurement took too long? */
      usDevice.state = ECHO_IDLE;
      return 0; /* no echo, error case */
```

```
    }
  }
  us = (usDevice.capture*1000UL)/(TU1_CNT_INP_FREQ_U_0/1000);
  return us;
}
```

It sends the 10 us trigger and then waits for the finished state. In case of overflow (no echo received), I simply return a value of zero.

The function returns the measured echo time in microseconds. To deal with different timer frequencies, the macro `TU1_CNT_INP_FREQ_U_0` is used which is generated by Processor Expert.

Calculating to Distance

Usually I'm not interested in the echo time, but rather in the distance itself. For this I have created a utility function to transform the echo time (microsecond) into centimeter distance:

```
static uint16_t calcAirspeed_dms(uint8_t temperatureCelsius) {
  /* Return the airspeed depending on the temperature, in deci-meter per
  unsigned int airspeed; /* decimeters per second */

  airspeed = 3313 + (6 * temperatureCelsius); /* dry air, 0% humidity, se
  airspeed -= (airspeed/100)*15; /* factor in ~15% for a relative humidit
  return airspeed;
}

uint16_t US_usToCentimeters(uint16_t microseconds, uint8_t temperatureCe
  return (microseconds*100UL)/calcAirspeed_dms(temperatureCelsius)/2; /*
}
```

Speed of sound depends on factors like temperature, relative humidity and above sea level (media density). The above function is not counting in everything, but is accurate enough for me.

Example Application

Now using it is pretty simple: from my main routine I intialize my drivers, and then periodically call the function to measure the distance. To make things visual, I show values on a LCD plus show with the RGB LED the distance:

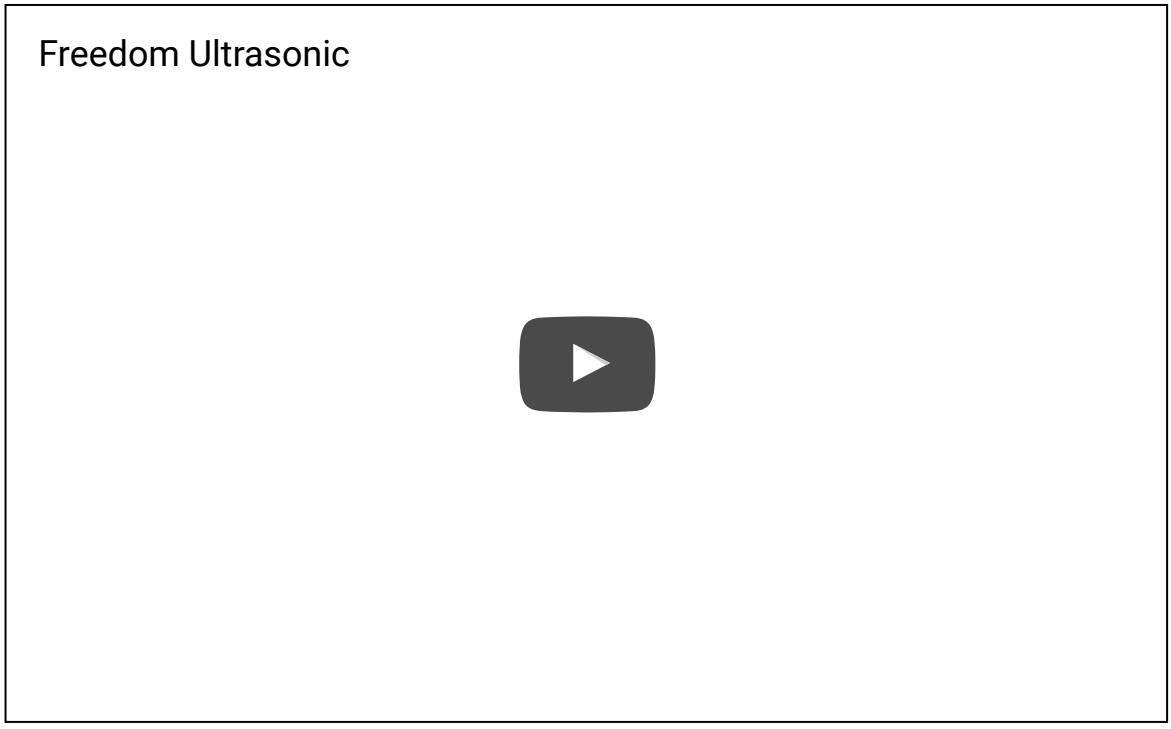
```
static void Measure(void) {
  uint16_t us, cm;
  uint8_t buf[8];

  us = US_Measure_us();
  UTIL1_Num16uToStrFormatted(buf, sizeof(buf), us, ' ', 5);
  LCD1_GotoXY(1,5);
  LCD1_WriteString((char*)buf);

  cm = US_usToCentimeters(us, 22);
  UTIL1_Num16uToStrFormatted(buf, sizeof(buf), cm, ' ', 5);
  LCD1_GotoXY(2,5);
  LCD1_WriteString((char*)buf);

  LEDR_Put(cm<10); /* red LED if object closer than 10 cm */
  LEDB_Put(cm>=10&&cm<=100); /* blue LED if object is in 10..100 cm range
  LEDG_Put(cm>100); /* blue LED if object is in 10..100 cm range */
}

```

```
/*lint -save  -e970 Disable MISRA rule (6.3) checking. */
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.

  /* Write your code here */
  US_Init();
  LCD1_Clear();
  LCD1_WriteLineStr(1, "us: ");
  LCD1_WriteLineStr(2, "cm: ");
  for(;;) {
    Measure();
    WAIT1_Waitms(50); /* wait at least for 50 ms until the next measureme
  }

  /*** Don't write any code pass this line, or it will be deleted during
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS cor
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. M
  #endif
  /*** End of RTOS startup code.  ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! **
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! **
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

If you do not have a LCD, simply strip off the LCD code.

The other thing to consider: inside the `main()` loop I use a delay of 50 ms: In case of an echo it will take about 25 ms for the echo to arrive: with 50 ms I will be calm for some time until I send another ‘ping’.

And here is a short video to see everything in action:



Summary

Once getting through the concept and principles, it is rather easy to use an ultrasonic range sensor like the HC-SR04. Setting up the input capture/counter is the most tricky part. But I think with the concepts and steps outlined above, this should work out pretty good for any other microcontroller, especially if used with CodeWarrior and Processor Expert.

The project shown above is available here.

If you are looking for improvements, here are some ideas:

- properly calculate in the humidity and air density.
- Using a temperature sensor to factor in the current air temperature.
- Averaging the samples.
- Dynamic measurement time, adaptive to the distance found to save battery energy.
- Using an RTOS like FreeRTOS. Especially to reduce the 'busy waiting' time and to do something useful during that time.
- Shell support (e.g. FSShell) to show values on a console, including setting of calibration values.
- Create a Processor Expert Component for it.
- As the ultrasonic beam is rather focused: add a pan a tilt servo to it to 'explore' more.
- Adding more sensors and actuators to use it for a robot application🙂

Happy Ultrasonic-Sounding🙂

About these ads

SHARE THIS:

Print Email Reddit Twitter Facebook 6 LinkedIn 2
Tumblr Pinterest Google Pocket

★ Like
Be the first to like this.

RELATED

Tracked Robot Update: Ultrasonic, Line Sensor and Shell with SMAC 802.15.4
In "Embedded"

Using the DHT11/DHT22 Temperature/Humidity Sensor with a FRDM Board
In "ARM"

Zumo Line Following with FRDM-KL25Z
In "Boards"

This entry was posted in **Boards**, **CodeWarrior**, **Embedded**, **Embedded Components**, **Kinetis**, **KL25Z Freedom Board**, **Processor Expert**, **Tips & Tricks** and tagged **CodeWarrior**, **Eclipse**, **Embedded Component**, **freedom board**, **KL25Z Freedom Board**, **Processor Expert**, **Tips&Tricks** by **Erich Styger**. Bookmark the **permalink [https://mcuoneclipse.com/2013/01/01/tutorial-ultrasonic-ranging-with-the-freedom-board/]** .

About Erich Styger

Embedded is my passion....
View all posts by Erich Styger →

87 THOUGHTS ON "TUTORIAL: ULTRASONIC RANGING WITH THE FREEDOM BOARD"

Alex
on **January 2, 2013 at 16:06** said:

Hi, Erich!
Great job!, I really enjoy your posts!

I'm having some troubles adapting an example of an USB HOST MSD example to work with PE, I posted the problem in the Freescale Forums:
https://community.freescale.com/thread/303017
Can you give some orientation?
Thanks in advance🙂

★ Like

Erich Styger
on **January 4, 2013 at 20:08** said:

Hi Alex,
sorry, I do not have that K20, so I'm not of any help here I think.

Alex
on **January 4, 2013 at 21:00** said:

I tried the same in my Freedom board and failed😢 , It would be great for you to do a tutorial on this, when you can, no pressure🙂

★ Like

Erich Styger
on **January 4, 2013 at 21:33** said:

Hi Alex,
well, I thought that this *is* already a tutorial? Maybe one of your settings is wrong? Could you try the project (link at the end of the tutorial). You might simply remove the LCD code, and if you use the same pins, then it should work for you as well.
Hope this helps,
Erich

★ Like

Alex
on **January 5, 2013 at 03:17** said:

Upps!! Sorry for the confusion, I meant to read a USB stick with the Freedom board.

★ Like

Injun Ear
on **January 4, 2013 at 07:54** said:

Hi Erich,

Thanks for the above. It may save someone's life some day (remote chance). I have been pondering how to impliment an emergency stop circuit on a device I have been designing with the FRDM board.

I think using the above idea to detect if someone (or part of someone) is in a dangerous place is a good idea.

Thank you.

★ Like

Erich Styger
on **January 4, 2013 at 08:06** said:

You are welcome. Just keep in mind that the ultrasonic beam is rather small and focused. To cover a larger area it is necessary to move the beam around, or to use multiple sensors. Depending on your application: in robotics (I mean big industrial robots) they are using in many cases a 'laser curtain': multiple laser beams to form a curtain to detect if something/someone is moving into the danger zone. That's simpler and more reliable than ultrasonic in my view.

★ Like

Pingback: The Freedom Robot | MCU on Eclipse

Cristian Iacob
on **January 26, 2013 at 17:19** said:

Hi,
After reading your posts I ordered one FRDM-KL25Z and started to have fun:)
Can you please help me with this application:
First of all the compiled code provided is not loading properly. (No LCD attached but the RGB indication should have worked)
Downloaded the project.
Removed the LCD component.
Recompiled the code.

After recompiling and loading the RED led stays always on.
Added ConsoleIO component and printf("time is:%u\n", us); after us = US_Measure_us(); so I can read the values on my PC.
Now the I can see that us has some values as I move an object in front of the sensor but they are like:27, 24, 21,18,15,12, 9 and only at very close range(5 cm). If the object is farther away then the us value displayed is always 12.
Some representation problem? Missing some bits from the uint16_t?

Thank you.
Cristian

★ Like

Erich Styger
on **January 26, 2013 at 17:32** said:

Hi Cristian,
Could you verify that your sensor is returning the correct pulse on the echo line? Best if you could use a logic analyzer or oszilloscope.

Erich

★ Like

Cristian Iacob
on **January 26, 2013 at 17:55** said:

Hi,
thank you for the quick answer,
-No logic analyzer, maybe will borrow a scope.
-Will check the setup again tomorrow
-When trying to load the precompiled code provided by you (Freedom_HC-SR04.hex) the green led near the pemicro chip acts strangely (series of 7 or 8 rapid blinks, pause, then blinks again). Something is wrong with that code?
-I am not mistreating that bird, that baby entered my house and just before relieving it I took a snapshot🙂 They are little owls.The whole family lives right under my roof. Something like this:
http://www.beleefdelente.nl/vogel/steenuil

★ Like

Erich Styger
on **January 26, 2013 at 18:04** said:

Nice birds, indeed!
I suggest that you check as well the wiring:Make sure you have that voltage divider in place, plus everything connected to the right pins on the Freedom board.
And it could be possible as well that your ultrasonic module is not working properly. I assume you are using the HC-SR04? Because there might be different types of modules around.

★ Like

Cristian Iacob
on **January 31, 2013 at 20:23** said:

Hi!
Problem Solved!
The HC-SR04 module was defective! Replaced it and now the application works nicely🙂
Maybe some people are used to it, but I discovered today: The PicKit2 programmer has a Logic Tool function which is very useful.
Still 2 problems remain:
1-The code provided by you Erich has the Flash image format set as “ihex” and I cannot load it into the FRDM-KL25Z. I have to set it as “srec” and rebuild the project. Also the LCD component has to be removed if LCD module not available.
2-Measuring the Trig signal I noticed it is 120us long, not 10us as expected. Trying to decrease the argument of the WAIT1_Waitus(10); function I can only obtain 3 us for WAIT1_Waitus(5) and no other values in between. Why could this happen?

Keep up the good work!

Cristian

★ Like

Erich Styger
on **January 31, 2013 at 22:03** said:

Hi Cristian,
good to hear you are making progress.
About 1): Thanks, I have fixed this now on https://github.com/ErichStyger/mcuoneclipse/tree/master/Examples/FRDM-KL25Z/Freedom_HC-SR04
About 2): Are you sampling with a high enough frequency in your logic analycer. I mean it should be at least in the 1 us range if not faster.
And are you using the latest version of 'Wait' component? I had fixed some timing back in December (http://steinerberg.com/EmbeddedComponents/Wait/home.htm). Wait is using 'busy waiting' and not very precise, and it can be interrupted by other interrupts. So you might disable interrupts around the Wait() call.

★ Like

lecsdexter
on **February 3, 2013 at 19:19** said:

Hi, do you have any project in which you read analogic signal (outside of the board) and display it at LCD?

★ Like

Erich Styger
on **February 3, 2013 at 19:26** said:

Hello, no, not at this moment.

★ Like

John
on **April 12, 2013 at 00:51** said:

I can use this sensor in PIC18F4550?

★ Like

> 
>
> **Erich Styger**
> on **April 12, 2013 at 05:47** said:
>
> Hi John, yes, I do not see why this should be a problem. All what you need is an output pin plus an input pin. And of course adopt the driver/software for the PIC.
>
> ★ Like

John
on **April 12, 2013 at 12:52** said:

I'm now learning PIC am newbie. I know very little about PIC
I have a project, and I have to measure the distance from this sensor (HC-SR04) using PIC18F4550.
Do you know any tutorial to learn how to do?

★ Like

> 
>
> **Erich Styger**
> on **April 12, 2013 at 12:57** said:
>
> Hi John,
> I did an internet search for "HC-SR04 PIC" and received plenty of hits/pages. I suggest to start there?
>
> ★ Like

John
on **April 18, 2013 at 10:51** said:

I found this: http://www.youtube.com/watch?v=dRq0e9rwqHo
but does not use the program that work, work with MPLAB.
Do you know any tutorial for this program?

★ Like

Erich Styger
on **April 18, 2013 at 12:04** said:

Hi John,
no, I do not have a tutorial for MPLAB, sorry.

★ Like

Pingback: Tracked Robot Update: Ultrasonic, Line Sensor and Shell with SMAC 802.15.4 | MCU on Eclipse

Pingback: Mini Sumo Robot with Proximity Sensors | MCU on Eclipse

Fernando
on **September 18, 2013 at 21:32** said:

Hello,
First, sorry my english.
My name is Fernando, I from Brazil. Congratulation foi job.
I wonder if it is possible to add an additional ultrasonic sensor using the same component TimerUnit_LDD. And how do I?

Thank you

★ Like

Erich Styger
on **September 18, 2013 at 21:48** said:

Hi Fernando,
thanks🙂
no worries about English: mine is not perfect too😉
I think it should be possible to add another sensor, but I have not done this. Would need to try first myself. Sure the same time base could be used to measure the signal.

★ Like

Fernando
on **September 18, 2013 at 21:53** said:

Thank you for the reply!
This, put two sensors, however, he only reads a sensor, the other gives error in this part:

```
while (usDevice.state! = ECHO_FINISHED) {
if (usDevice.state == ECHO_OVERFLOW) {
usDevice.state = ECHO_IDLE;
return 0; / * ERROR * /
}
}
```

The second sensor enters if and returns 0. I had to declare another variable of type “static US_DeviceType” you know if you need to set something before you start using?

★ Like

Fernando
on **September 20, 2013 at 21:09** said:

Thank you for the reply!
This, put two sensors, however, he only reads a sensor, the other gives error in this part:

```
while (usDevice.state! = ECHO_FINISHED) {
if (usDevice.state == ECHO_OVERFLOW) {
usDevice.state = ECHO_IDLE;
return 0; / * ERROR * /
}
}
```

The second sensor enters if and returns 0. I had to declare another variable of type “static US_DeviceType” you know if you need to set something before you start using?

★ Like

Fernando
on **September 20, 2013 at 21:10** said:

Or a sensor picks up the same distance from each other!

★ Like

Calin Dragos George
on **October 3, 2013 at 20:04** said:

It could be much better if you expand the tutorial in order to build more complex robotics applications. A series of tutorial can be found in this article http://www.intorobotics.com/interfacing-programming-ultrasonic-sensors-tutorials-resources/

★ Like

Erich Styger
on **October 3, 2013 at 20:09** said:

Yes, agreed. This tutorial is just a starting point. On GitHub I have placed some more advanced usage examples, see https://mcuoneclipse.com/2013/09/08/mini-sumo-robot-with-proximity-sensors/

BTW: Nice summary you wrote about different ultrasonic modules, thanks for this!

★ Like

Jonny
on **November 26, 2013 at 16:13** said:

Hi Erich,

Great tutorial- been really helpful so far! I am new to programming and have a couple of questions.

With the code, what header file should be used?

Should the code you have given on this page be in one .c source file or a different file for each section?

Thank you for your help

Jonny

★ Like

Erich Styger
on **November 26, 2013 at 16:31** said:

Hi Jonny,
best if you check out the sources here:
https://github.com/ErichStyger/mcuoneclipse/tree/master/Examples/FRDM-KL25Z/Freedom_HC-SR04/Sources

★ Like

Jonny
on **November 26, 2013 at 18:47** said:

Spot on, thank you!

The only problem I'm having is that some of the headers have errors (unresolved inclusions?). These are the following:

#include "LEDR.h"
#include "LEDpin1.h"
#include "BitIoLdd1.h"
#include "LEDG.h"
#include "LEDpin2.h"
#include "BitIoLdd2.h"
#include "LEDB.h"
#include "LEDpin3.h"
#include "BitIoLdd3.h"
#include "TU1.h"
#include "LCD1.h"
#include "RW1.h"
#include "BitIoLdd15.h"
#include "EN1.h"
#include "BitIoLdd4.h"
#include "RS1.h"
#include "BitIoLdd5.h"
#include "DB41.h"
#include "BitIoLdd10.h"
#include "DB51.h"
#include "BitIoLdd11.h"
#include "DB61.h"
#include "BitIoLdd12.h"
#include "DB71.h"
#include "BitIoLdd13.h"
#include "UTIL1.h"

Do you know what the problem might be with this? Do I need to install a library somewhere?

Thanks

Jonny

★ Like

Erich Styger
on **November 26, 2013 at 20:19** said:

Yes, you need to load additional Processor Expert libraries/components (see https://mcuoneclipse.com/2013/05/09/processor-expert-component-peupd-files-on-github/) and generate code.

★ Like

Jonny
on **December 1, 2013 at 01:10** said:

Once loaded do I need to add each component into the project? Many of them have errors- will these need setting for the specific pins etc on the board? Thanks

★ Like

Erich Styger
on **December 1, 2013 at 13:33** said:

Hi Jonny,
The components are already added to the project. If you have all components installed, you can simply generate code and everything should work out of the box. Of xourse if you have the sensor on different pins, then you need to change the pin settings. Maybe you are not (yet?) familiar with Processor Expert and how this works? I suggest to have a look at this tutorial (https://mcuoneclipse.com/2012/09/07/tutorial-enlighting-the-freedom-kl25z-board/) as it goes through the individual steps.

★ Like

Seumas

on **March 13, 2014 at 18:09** said:

Erich,
First off, thank you so much for all your tutorials. I've been playing with the Kinetis boards for some hobby projects over the last year and your tutorials and insights have been invaluable.

I've modified your code to use 4 sensors. I added three additional channels to the timer unit. I also added function pointers for the trigger pin functions to your US_DeviceType struct and then created another struct (US_Devices) that holds an array of 4 US_DeviceType structs and a counter. In the init function I set the function pointers for each sensor to the appropriate pin function. Then I modified your code to loop through the 4 sensors updating the counter in the struct using the counter and pin function pointers in the struct pointer passed to the events. Theoretically it should be usable for any number of sensors.

I've posted it to GitHub so others can use it if they like.

https://github.com/madseumas/Kinetis/tree/master/Ultrasonic

It has some other code from FreeRTOS; I keep the US sensors in their own task since US_Measure blocks and I send off the results to another task via a queue.

At some point I'd like to get it so I can fire the sensors in pairs (front/back and left/right) at the same time, but since the timer unit's overflow event fires for all 4 channels without indicating which channel overflowed I'm not sure yet how to go about that.

I'm a relative C newb so I wouldn't be surprised if there are any bugs I haven't caught yet.

Once again thank you so much for all of your tutorials!

★ Like

Erich Styger
on **March 13, 2014 at 20:44** said:

Hi,
many thanks for sharing your work. I plan to build a dual-sensor system in the near future, so this for sure would be helpful. Many thanks again!

★ Like

Romik
on **March 27, 2014 at 11:35** said:

Thank you for your blog, it's very helpful.
I made robot on FRDM-KL05Z with HC-SR04 too.
http://www.robotmaker.ru/2014/03/22/1366/

★ Like

Erich Styger
on **March 27, 2014 at 16:39** said:

thanks for sharing, that one really looks very nice!

★ Like

Bartek
on **May 15, 2014 at 10:13** said:

Hi Erich,
Thank for your great work. I have some troubles with load your ready code to CodeWarrior and programm to board.
Can you tell how can I do it properly ?
(sorry for that simple issue but I'm a newbie ;P)
Best regards

★ Like

Erich Styger
on **May 15, 2014 at 10:16** said:

Hi Bartek,
are you using the FRDM-KL25Z board? Keep in mind that the board comes with a bootloader shipped (and not with the debug application). So you need to load the right firmware. I recommend that you use the latest P&E one (see https://mcuoneclipse.com/2013/12/14/new-pe-opensda-firmware-v114/).
See https://mcuoneclipse.com/2012/09/20/opensda-on-the-freedom-kl25z-board/ how to program the new firmware.

★ Like

Bartek
on **May 15, 2014 at 14:32** said:

Hi,
yes I use FRDM-KL25Z. I did like this: enter the bootloader mode, than I droped this file “MSD-DEBUG-FRDM-KL25Z_Pemicro_v114.SDA” to root and replug the board.

★ Like

Erich Styger
on **May 15, 2014 at 14:57** said:

Yes, that sounds ok. What operating system are you using? Windows 8.1? If so, then there is a known problem: https://mcuoneclipse.com/2013/10/12/frdm-board-bootloader-fails-with-windows-8-1-preview/

That problem is solved with the V1.14 firmware (but you need a Windows 7 machine (or 8.0)) to update the bootloader firmware.

★ Like

Bartek
on **May 15, 2014 at 15:04** said:

I use Windows 7 (home premium version x64).

★ Like

Erich Styger
on **May 15, 2014 at 15:23** said:

Then this should not be an issue. Can you copy the file with the DOS shell/command prompt?
And if you have a virus scanner, to disable it?

★ Like

Bartek
on **May 15, 2014 at 15:39** said:

All this stuff was done earlier (open sda update). Thing is that my board is working fine (I can run the .srec precompiled program and it works). The point is I want to load ultra sonic project on board properly. I use “flash programmer” –> “flash file to target”, I choose .hex file (“file to flash”) from your project and press “erase and program”. It shows that transfer is succesful but after replug nothing is happend (rgb diode is dead). All devices are connect properly.

★ Like

Bartek
on **May 15, 2014 at 16:41** said:

Ok I did it ! I load .elf file instead .hex and it’s work😀
Thank for your help
Best regards !

★ Like

Erich Styger
on **May 15, 2014 at 18:20** said:

Try with an S19 file (Motorola S19 file). .Hex are usually intel hex files, and I think this is not supported by the bootloader.

★ Like

Pingback: Programmable Ultrasonic Sensor Shield for FRDM Board | MCU on Eclipse

Pingback: Sensor and Communication Shield for Sumo Robot | MCU on Eclipse

augusto
on **September 29, 2015 at 02:38** said:

Hi Erich,
Thank for your great work. But I am new at this and I would like know how can I create a simple timer using external interrupt.
Thank you.

★ Like

Erich Styger
on **October 1, 2015 at 20:54** said:

Welcome to a new world🙂 Not sure what you mean with 'using an external interrupt'? I have a tutorial related to timers for example here: https://mcuoneclipse.com/2012/08/13/tutorial-timer-led-with-processor-expert-for-kinetis/

★ Like

Michael
on **November 12, 2015 at 18:14** said:

Hi Erick, thank you for your blog, it's very helpful.
I have a question regarding the timer channels. My application needs 2 timers. I tried to add one more channel in the processor expert. However, I don't know how to build the code with two or more channels. Could you help me?
Thank you so much.

★ Like

Erich Styger
on **November 12, 2015 at 18:19** said:

Hi Michael,
You can have multiple timers with each having a number of channels. You can use the TimerUnit_LDD for example for this.

Erich

★ Like

akshay
on **December 3, 2015 at 08:07** said:

can i used this sensor for scouring depth measurement ??

★ Like

Erich Styger
on **December 3, 2015 at 10:11** said:

That will depend on the sonic reflection of your material, and of course depends on the sensor itself.

★ Like

akshay
on **December 3, 2015 at 08:14** said:

what is %error in distance measurement by ultrasonic sensor instead of gauge scale? & how to compare accuracy of in measured distance by help of both [ultrasonic sensor & gauge scale]

★ Like

Erich Styger
on **December 3, 2015 at 10:12** said:

Not sure what the 'gauge scale' is for you. But yes, you can use ultrasonic sensors to meausure the height/gauge of liquids e.g. in tanks.

★ Like

Pingback: Interfacing Arduino with HC-SR04 Ultrasonic Distance Sensor – Electronics Hobby Club, J.I.S.T.

magalie.grd@gmail.com
on **April 21, 2016 at 17:04** said:

Hi, can you explain how can we implement the temperature sensor (mine is MCP9801) ? Is it the component I2C_LDD ? I've found a sample code on http://cache.nxp.com/files/sensors/doc/app_note/AN4481.pdf and I think the "Single-Byte Read" is adequate but I am not sure.. Can you help me ?

★ Like

Erich Styger
on **April 21, 2016 at 20:46** said:

Have a look at this tutorial:
https://mcuoneclipse.com/2012/09/21/tutorial-accelerating-the-kl25z-freedom-board/

★ Like

Magalie
on **April 21, 2016 at 22:24** said:

Ok thanks I'll try this !

★ Like

Petric
on **July 4, 2016 at 15:41** said:

Hi Erich,

Thank you for the tutorial. I will use your example to solve my task. I have a signal that gives me 'good time' and 'bad time' and I will only measure the 'good time'. 'Good time' starts with a rising edge and ends with e falling edge. The signal is constantly coming I can't use a trigger. Therefore my question is:
– How can I detect if it is a rising edge (–> reset timer) or a falling edge (–>capture value)?

Thanks Petric

★ Like

Erich Styger
on **July 4, 2016 at 15:47** said:

Hi Petric,
configure the pin to generate an interrupt both for the rising and falling edge. Inside that interrupt routine you can check/read the state of the pin (high or low) and then reset the timer or capture the value.

Erich

★ Like

Antonio Carlucci
on **July 4, 2016 at 17:30** said:

Hi Erich, we use the TimerUnit_LDD for this task. Ho we can check the state of the pin? Is there a function in the Timer Unit that we can use?
With the TimerUnit we detect both edges and starts the corresponding Timer. The problem is, we don't know the state of the pin…

Thanks antonio

★ Like

Petric
on **July 5, 2016 at 09:02** said:

Hi Erich,
thanks for the quick answer!
I guess you mean the Pin PTD4 and therefore the Interrupt TU1_OnChannel0, right? I cant' see how to read the state of the pin.
Or do you mean to configure a new External interrupt an configure that one?

Thanks
Petric

★ Like

Petric
on **July 5, 2016 at 10:31** said:

UPDATE: It's running now with an ExternalInterrupt. And I switcht the TU1_OnChannel0 off. Is not so elegant but working:-)
But I'm still woundering, if there is a way to to read the state of the pin in the TU1_OnChannel0 interrupt.

★ Like

Erich Styger
on **July 5, 2016 at 15:18** said:

I think you might be able to simply read the data port register. In the worst case you would need to mux the port as GPIO inside the interrupt.

★ Like

OMSIM ONURES SEROMA
on **September 26, 2016 at 06:56** said:

Hello! where should I add sentences to use another pin as a signal output? I want to implement a motor CD and change their velocity with differents distances using pwm but where can I write the code? I had tried to use a pin as output and used "if" to activated HIGH with differents distances but it doesn't works.. i had tried to add if sentences with pins but always is wrong.. how can I add more conditions and to have another pins as output ? could you write an example? how to control other output pins with distance? help plis!

★ Like

Erich Styger


on **September 26, 2016 at 11:09** said:

This is really up to you: you can add your motor driver code anywhere. But I recommend that you create a dedicated .c and .h file for such a functionality.

★ Like


GhelGAMISH
on **October 12, 2016 at 22:24** said:

Hi sir Styger,

Thanks for this code, I already put an FreeRTOS in the project and I combined it with the accelerometer project. Unfortunately the accelerometer doesn't work.

★ Like


Erich Styger
on **October 13, 2016 at 07:50** said:

Have you set the interrupt priorities correctly for the I2C part and FreeRTOS? Keep in mind that some interrupts are masked out by the RTOS. Have a read at https://mcuoneclipse.com/2016/08/14/arm-cortex-m-interrupts-and-freertos-part-1/ and especially part 3. I suggest that you give the I2C interrupt the highest priority as I think you are running into a timeout.

★ Like


aluminumiron
on **October 14, 2016 at 03:47** said:

Okay sir, I'll try it. I;ll update you if it works.

★ Like

Carlos Esteban Miguens
on **October 17, 2016 at 23:33** said:

Hello Erich! How are you?

I run on a FRDM KL25Z the firmware of this post with a US-100 ultrasonic sensor, is a similar sensor like HC-SR04 and run very well, but I have a cuestion. What you recommends do to use more than 1 ultrasonic sensor on a single frdm board?

Thanks!

★ Like

Carlos Esteban Miguens
on **October 17, 2016 at 23:48** said:

I need put a new Trigger and a new Timer Unit more and setup the pins of the new ultrasonic sensor?

★ Like

Erich Styger
on **October 18, 2016 at 09:57** said:

See my other reply: this would be ideal. Otherwise you need to build up hardware to deal with multiple input signals e.g. with a multiplexer.

★ Like

Erich Styger
on **October 18, 2016 at 09:56** said:

Best of course is if you have a dedicated timer input channel for each sensor, so you can read in values in parallel. Ultrasonic signals are rather slow and you cannot ping too frequently, so having independent channels is better.

★ Like

Carlos Esteban Miguens
on **October 19, 2016 at 22:36** said:

Hi Erich! Thanks for your reply. I was able to implement succesfully two ultrasonic sensors using the code of this post and following your instructions. On this project:

https://github.com/devtodev/Kinetis-ARM/tree/master/medidorDistancia

I used the code written by you on that post, so I will mention you in the sources, Is it ok?

Thanks!

★ Like

Erich Styger
on **October 20, 2016 at 05:57** said:

Hi Carlos,
great, congratulations!
And yes, thanks for referencing the original source/article.
Erich

★ Like

SaiPriya
on **October 19, 2016 at 20:57** said:

Hello Erich,
I have a quick question.
1. I am trying to make use of a tachometer and continuously measure the distance between two consecutive tachometer edges and maybe blink an LED if the measured distance between edges is beyond a certain limit which means the motor is stalled. Do you think basing my initial steps off of this tutorial is a good option?

2. In microcontroller terms, I am trying to capture the register values between the rising and falling edge of pulses in a single pulse train and do my task based off of a value .

3. My state machine would be measure->overflow->finish->measure. From my understanding there is no need for an idle state.

I would like to know about your thoughts as to how to go about implementing this. And if you feel if my approach has any flaws. Thanks!

P.S. I would appreciate it if you could redirect me to the proper tutorials too, if available. I am using a KE02Z FRDM board.

★ Like

Erich Styger
on **October 20, 2016 at 21:00** said:

Hi SaiPriya,
about 1) this all depends on the speed of your signal/edges. Keep in mind that for the Ultrasonic I'm using interrupts. For a quadrature encoder you might get too many interrupts that way.
So I would not use that approach if the signal frequency is say mor than 1 kHz.
Instead, rather use a sampling of the quadrature signals. I don't have a specific tutorial for this (I'm covering this in my university classes, but never had the time to write an article about this).
You might have a look at my QuadCounter Processor Expert component which implements this.

★ Like

SaiPriya
on **October 21, 2016 at 15:22** said:

I have to control two motors whose speeds are 11320 RPM and 7000RPM . So I am guessing the timer interrupt structure would be the best way to go about it, since my tachometer signal frequencies would be less than 250Hz. Thoughts?

★ Like

Erich Styger
on **October 21, 2016 at 18:47** said:

So you only get one signal per RPM? That's very low and basically not usable for any real speed estimation or positioning. Typical quadrature signals are in the range of 6000-10000 signals per revolution.
So if your signal is so slow, yes, you could use a timer.

★ Like

SaiPriya
on **October 24, 2016 at 23:22** said:

Hi Erich,
1. A quick update My Tach gives out 6 pulses per rotation of the motor. I would like to know if my thought process is right. There are in total 7220 rotations per minute. so per second there are 120.3 rotations. Which means one rotation would take 8.3ms. So I was thinking maybe if my motor skips 3 rotations i.e approximately 25ms of no tach pulse, I can safely assume that the motor has stalled. So my timer overun period could be 25ms??

2. I looked up "help on the component" for Event Capture on the TU1 module and I have trouble initialising the pointer in TU1_Init(). I have not used the structure and the pointer to structure that you have used to send as event handles. (New C programmer, pointer woes!). Do I necessarily have to use a pointer like you have done here?
My tach pulses need to be read on pin PTC4 on the microcontroller. and on both the rising and falling edges. If

my understnadin is right, I need the address of pin PTC4.
How do I go about finding that out?
Thanks for the help. Really appreciate it.

★ Like

Erich Styger
on **October 25, 2016 at 10:24** said:

Yes, you certainly could use the approach in 1 (measuring the pulse width). But I would not do this: instead I would sample the quadrature signal and estimate the speed. Based on the speed (zero for stalled) I would make that decision. That way no interrupts are needed and it is more safe way of implementation with no reentrancy problems. But it is your decision and design: I simply would not do it that way you describe.

About 2) I'm using a user data pointer so I get the data passed to my event modules. That pointer is a good thing to deal with reentrancy. It has nothing to do with the address of a pin.

★ Like