OSES ASSIGNMENT #1

Description: The purpose of this assignment is to implement a proxity alert system using:

- The HC-SR04 Ultrasonic Ranging Module;
- The FRDM-K64F board;
- The Micrium μC/OS.

Changing the led's lights and frequency, according to documentation of assignment.

Solution

1) HC-SR04 Ultrasonic Ranging Module

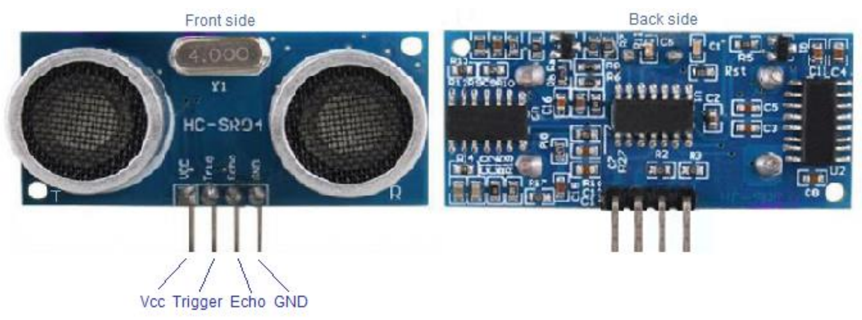


Figure 1. HC-SR04 Sensor

- **PIN Assignment**

The first thing to do is link the sensor to the board, according to these assignments:

VCC	P5V_USB
Trigger	PTB23
Echo	PTC1
GND	Ground

2) *FlexTimer Configuration*

The FTM0, in general, has four pairs or eight channels: CH0/CH1, CH2/CH3, CH4/CH5, and CH6/CH7. All the channel pins can output PWM signals and can be input pins to get capture signals. The FTM includes different features as PWM, quadrature decoder with input filters, relative position counting and interrupt on position count or capture of position count on external event and so on. In our case, we need dual edge capture for pulse measurement. For this purpose, we have to set the right configuration of FTM. The first thing to do is set FTMEN=1, DECAPEN=1, the ELSnB:ELSnA bits select the first edge of the captured signal. The ELSn+1B:ELSn+1A bits select the second edge of the captured signal. MSnA bit selects either the one-shot mode or the continuous mode. We set these parameters whit this function ***FTM_DRV_Init(HW_FTM0, &ftm0Info)*** that sets values in: channel (n) status and control register (FTM*x*_C*n*SC), counter initial value (FTM*x*_CNTIN), Features Mode Selection (FTM*x*_MODE) as so on. In few words, we need to disable writing protection of registers, set an initial value of the counter, the initial frequency. Then, select the dual edge capture mode ***FTM_HAL_SetDualEdgeCaptureCmd(),*** select the edge detection for each channel, one channel on rising edge and the other on falling edge, select the timer in one-shot mode (in one-shot mode, the timer will count up from zero to the latched value , generate an interrupt, reload the zero value and then stop) with these functions: ***FTM_HAL_SetChnMSnBAMode(),FTM_HAL_SetChnEdgeLevel().*** After that we have to enable the interrupts on these channel with ***FTM_HAL_EnableChnInt()***, set the clock source and the FTM starts ***FTM_HAL_SetDualChnDecapCmd().***

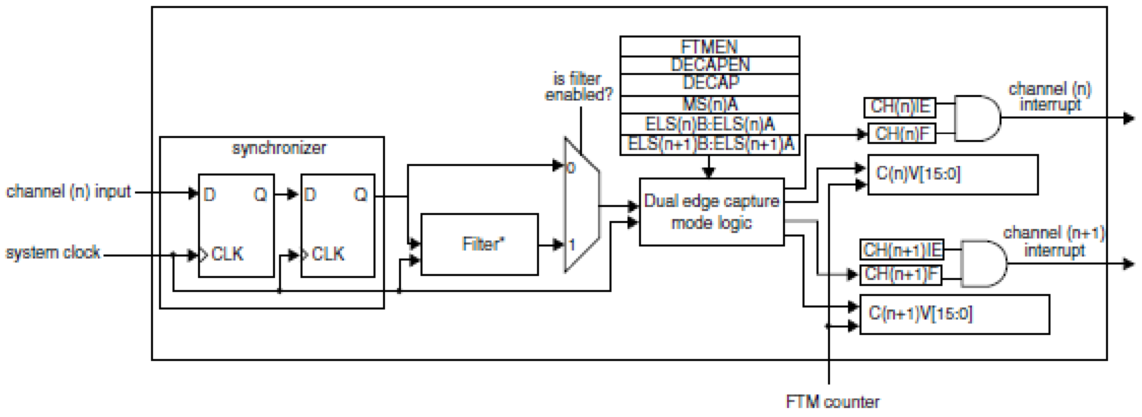
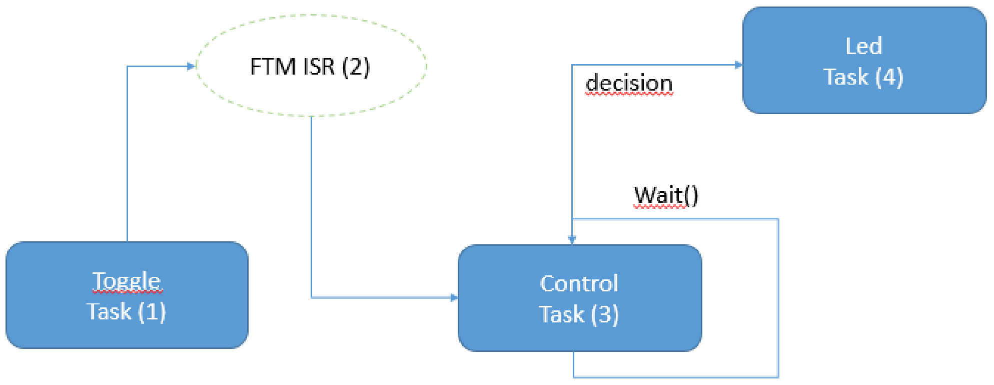


Figure 2. Dual-edge capture mode

After this configuration, we have to write an ISR, after enabling the interrupt for FTM0/1 channels. We have to test the status register until an event occurs, ***FTM_HAL_GetChnEventStatus(),*** if an event occurs we have to wait for the rising edge from the first channel and save the value of counter. Do the same for the second channel on rising edge, save the counter again, and make the difference between these two values. Use the square wave duration in order to compute the distance measured from the sensor. To have the right wave's duration we have to: use prescaler to divide the clock (divided by 64), 60 MHz is BUS Clock, because FTM module use this kind of clock
, a time interval is (t_after-T_before)/frequency in this case bus frequency (60 MH) but I use the prescaler to divide the clock. In our case, divide by 64, so (60/64) => frequency*64/60 and then divide by 58 to have the distance in cm. If I use the system clock (120MHz) and if I measure a long distance, the counter could overflow and it could fake the result. After this consideration, we have to clear some flags in the status register, set to zero the counter, and re-enable the FTM in order to do another measurement.

3) Solution's schematic



(1) In the toggle task, there is the toggling of Trigger signal with 1 ms in order to activate the Echo output of FlexTimer.

(2) In the FlexTimer ISR was caught the interrupt on the rising edge for the first channel of FTM and stored the value of the counter, after that the values of the counter were taken in the second channel of FTM, active on the falling edge and, in order to evaluate the Echo wave of sensor, was made the difference between those values. When the values was saved, we signal the control task that are blocked.

(3) This task is blocked until the signaling when the measured value is ready. After the measurement of distance, these values are stored in a buffer and it's called a procedure that sorts the values in the buffer and takes the middle buffer value.

(4) In the led task, there are all the operations necessary to turn on the correct light.

Brief user manual

In this part, all the instructions to use this system are described.

- Open the project in the Micrium folder. First of all, you have to put the files, in which there are the API in order to use Flex Timer module, in a Micrium folder and then add these files to the project *(Project -> Add Files..)* (File List: ***fsl_ftm_common.c, fsl_ftm_common.h, fsl_ftm_driver.c, fsl_ftm_driver.h, fsl_ftm_features.h, fsl_ftm_hal.c, fsl_ftm_hal.h***).

- ***(Optional)*** After that, in order to test this code, use the I/O to visualize the distances, it needs a terminal emulator like Tera Term or similar, to use serial port to communicate (as UART), and select the right baud rate (in this case 115200).

- The final step is build the project with this new files and the main program (in the attachments) and download the program on the board. Open the terminal e launch the program. The user can see the results on the terminal and see the leds blinking, according to the distance.