NC STATE UNIVERSITY

Dept. of Electrical and Computer Engineering

ECE 560 : Fall 2019

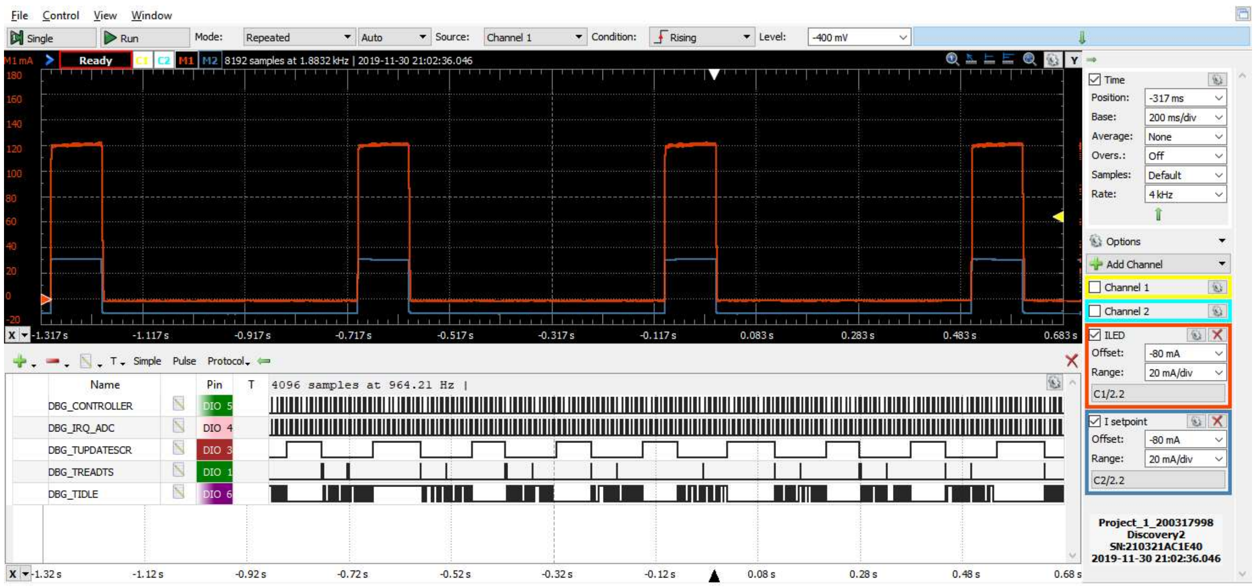
Project#2 : DISPLAYING CURRENTS & SHARING THE ADC

By

VISHNU SURESH MENON

NCSU ID : 200317998

Screenshot#1


Mode: Repeated | Auto | Source: Channel 1 | Condition: Rising | Level: -400 mV

8192 samples at 1.8832 kHz | 2019-11-30 21:02:36.046

Time — Position: -317 ms; Base: 200 ms/div; Average: None; Overs.: Off; Samples: Default; Rate: 4 kHz

ILED — Offset: -80 mA; Range: 20 mA/div; C1/2.2

I setpoint — Offset: -80 mA; Range: 20 mA/div; C2/2.2

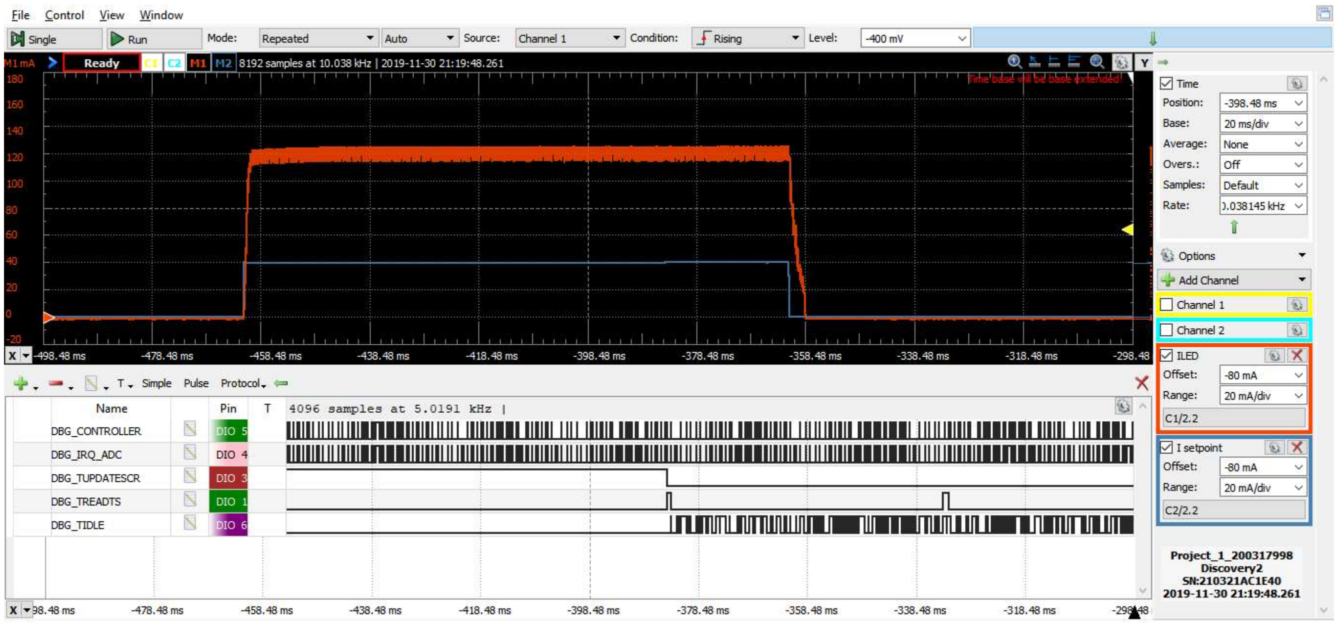
4096 samples at 964.21 Hz |

Name	Pin
DBG_CONTROLLER	DIO 5
DBG_IRQ_ADC	DIO 4
DBG_TUPDATESCR	DIO 3
DBG_TREADTS	DIO 1
DBG_TIDLE	DIO 6

Project_1_200317998
Discovery2
SN:210321AC1E40
2019-11-30 21:02:36.046

The above mixed signal screenshot illustrates the oscilloscope reading of ILED current flowing through the buck converter along with the debug signals of different threads and interrupt handlers used in this project. This screenshot depicts only the priority conversion of the ADC ie; when the touchscreen is not touched. Here, the threads “Update Screen”, “Read TS”, “Idle Thread” and the ADC interrupt request handler are shown in the logic analyzer window.

Screenshot#2


Mode: Repeated | Auto | Source: Channel 1 | Condition: Rising | Level: -400 mV

8192 samples at 10.038 kHz | 2019-11-30 21:19:48.261

Time base will be base extended!

Time — Position: -398.48 ms; Base: 20 ms/div; Average: None; Overs.: Off; Samples: Default; Rate:).038145 kHz

ILED — Offset: -80 mA; Range: 20 mA/div; C1/2.2

I setpoint — Offset: -80 mA; Range: 20 mA/div; C2/2.2

4096 samples at 5.0191 kHz |

Name	Pin
DBG_CONTROLLER	DIO 5
DBG_IRQ_ADC	DIO 4
DBG_TUPDATESCR	DIO 3
DBG_TREADTS	DIO 1
DBG_TIDLE	DIO 6

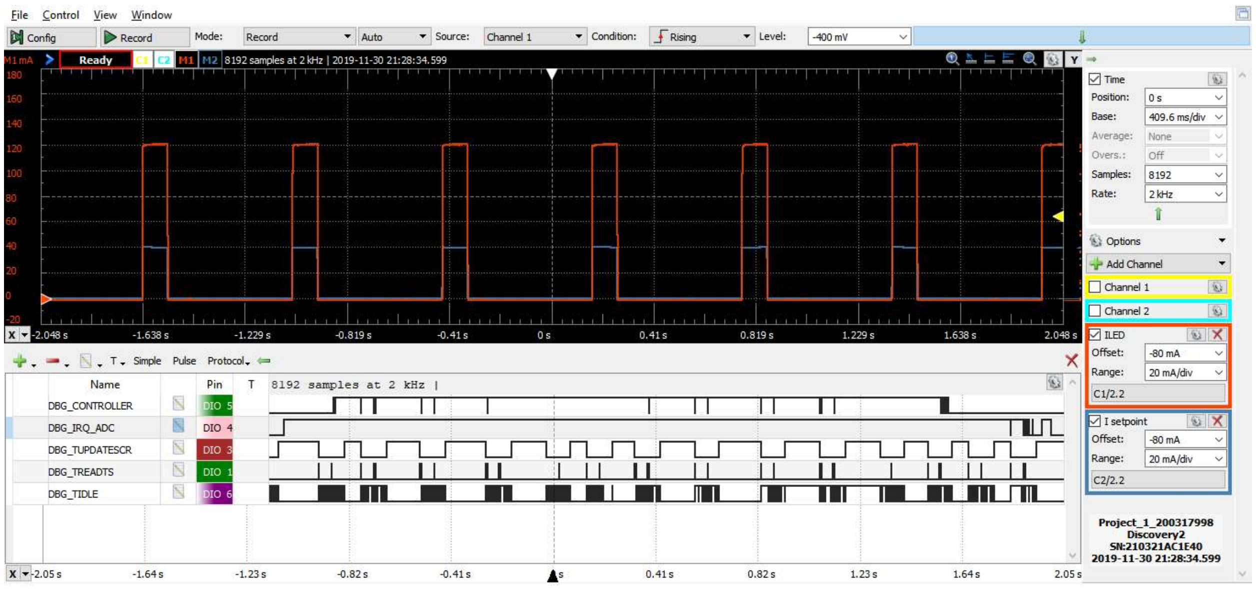
Project_1_200317998
Discovery2
SN:210321AC1E40
2019-11-30 21:19:48.261

The screenshot#2 is the same as above mentioned (screenshot#1) with slightly zoomed in version of it.

Observations and Explanations

1. Here, only the priority conversion is handled where the ADC is utilized by the buck controller only and the conversion is hardware triggered (overflow of TPM0)
2. The orange and blue signals of the oscilloscope are the ILED measured current and ILED set current respectively.
3. The average ILED set current is constant stabilized value of 40 mA and the measured current keeps on charging and discharging until it reaches the set current value of buck converter.
4. The ADC IRQ which calls the Control HBLED function gets invoked each time an ADC interrupt is issued post the ADC conversion which in turn is controlled by the timer overflow of TPM0.ie, the ADC IRQ gets called within period of counting (2*MOD*period of timer module counter clock).
5. The synchronization between the oscillation of buck current(ILED) and the invocation of the HBLED controller is evident from screenshot#2, ie; at instances when controller gets called the ILED current starts rising to the set current average value and the function Control HBLED controls the measured current to reach the value of set current by adjusting the duty cycle based on different controller modes.
6. Thread Read Update Screen comes into the blocked state since there exist no queued conversion request from the thread Read TS as the screen is not pressed and moreover this thread gets blocked by the ADC IRQ on a periodic basis and also by the higher priority thread "Buck Update Set Point". Hence , there exist no sufficient time for the thread to run.
7. Thread Read TS runs when there is no higher priority thread is being run. Each time the thread runs it checks if there exist any input/request from the screen to process. In this case as the screen is not pressed the thread doesn't encounter any event to get blocked.

Screenshot#3



Mode: Record | Auto | Source: Channel 1 | Condition: Rising | Level: -400 mV

8192 samples at 2 kHz | 2019-11-30 21:28:34.599

Time	
Position:	0 s
Base:	409.6 ms/div
Average:	None
Overs.:	Off
Samples:	8192
Rate:	2 kHz

Channel 1
Channel 2

ILED	
Offset:	-80 mA
Range:	20 mA/div
C1/2.2	

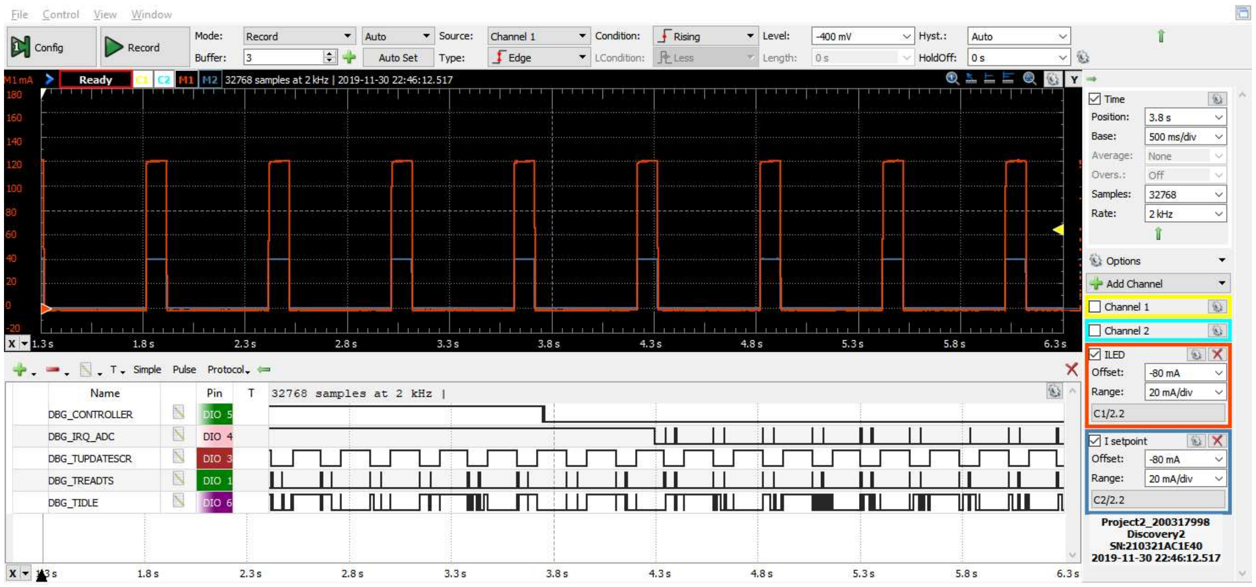
I setpoint	
Offset:	-80 mA
Range:	20 mA/div
C2/2.2	

Name	Pin
DBG_CONTROLLER	DIO 5
DBG_IRQ_ADC	DIO 4
DBG_TUPDATESCR	DIO 3
DBG_TREADTS	DIO 1
DBG_TIDLE	DIO 6

Project_1_200317998
Discovery2
SN:210321AC1E40
2019-11-30 21:28:34.599

The above mixed signal screenshot illustrates the oscilloscope reading of ILED current flowing through the buck converter along with the debug signals of different threads and interrupt handlers used in this project. This screenshot depicts the queued conversion of the ADC ie; when the touchscreen is pressed. Here, the threads “Update Screen”, “Read TS” and the ADC interrupt request handler are shown in the logic analyzer window.

Screenshot#4



Mode: Record | Auto | Source: Channel 1 | Condition: Rising | Level: -400 mV | Hyst.: Auto
Buffer: 3 | Type: Edge | LCondition: Less | Length: 0 s | HoldOff: 0 s

32768 samples at 2 kHz | 2019-11-30 22:46:12.517

Time	
Position:	3.8 s
Base:	500 ms/div
Average:	None
Overs.:	Off
Samples:	32768
Rate:	2 kHz

Channel 1
Channel 2

ILED	
Offset:	-80 mA
Range:	20 mA/div
C1/2.2	

I setpoint	
Offset:	-80 mA
Range:	20 mA/div
C2/2.2	

Name	Pin
DBG_CONTROLLER	DIO 5
DBG_IRQ_ADC	DIO 4
DBG_TUPDATESCR	DIO 3
DBG_TREADTS	DIO 1
DBG_TIDLE	DIO 6

Project2_200317998
Discovery2
SN:210321AC1E40
2019-11-30 22:46:12.517

Observations and Explanations

1. Here, only the queued conversion is handled when the ADC is utilized by the touchscreen controller only and the conversion is software triggered.
2. The orange and blue signals of the oscilloscope are the ILED measured current and ILED set current respectively.
3. The average ILED set current is constant stabilized value of 40 mA and the measured current keeps on charging and discharging until it reaches the set current value of buck converter.
4. The ADC IRQ which calls the Control HBLED function gets invoked each time an ADC interrupt is issued post the ADC conversion which in turn is initiated by the software trigger ie., when the analog data on the touchscreen channel is fed on the SC1 register of the ADC peripheral.
5. The smaller pulses on the debug signal of ADC IRQ are the request handlers in response to the ADC interrupts issued by the queued conversion requests.
6. Read Update Screen switches between thread blocking and thread running state unlike in the priority conversion as it encounters an event or waits for the field (global values like "set current" and "measured current") to update and display on the screen. This is evident from the screenshot#4 that as soon the thread Read TS gets unblocked once the result for the request message is retrieved, this thread Update Screen gets blocked waiting for the result to update the LCD display.
7. Read TS thread gets blocked whenever the screen gets pressed as its waiting for the digital converted data(X and Y coordinates of the touch) to be available in the queue.

Synchronization and Data Exchange among "Control_HBLED" & "Thread_Update_Screen"(Plotting Thread)

There exist factors like sharing of global data, ie; measured current and set current among threads that plot values and produces these global values on periodic instances. Here, the function "UI_Draw_Screen" invoked from Thread_Update_Screen is responsible enough to plot the global current values that are obtained from the producer which is Control_HBLED that gets to run within the period of counting based on timer overflow.

The approach used to share the global data is making use of two arrays for the parameters "measured current" and "set current" respectively. The two arrays are used to store the averaged value of four samples of these parameters taken each time the buck controller function runs every 21us post an ADC interrupt.

Also, the timing synchronization for the data exchange between the threads is also a necessary factor to be considered as there exist a constraint that the values needs to get plotted at every 20ms and every new measurement of "measured current" values gets updated on every 21us (approx.)

The flash parameters in this project are as specified below

1. Flash duration : 100ms
2. Flash period : 600ms

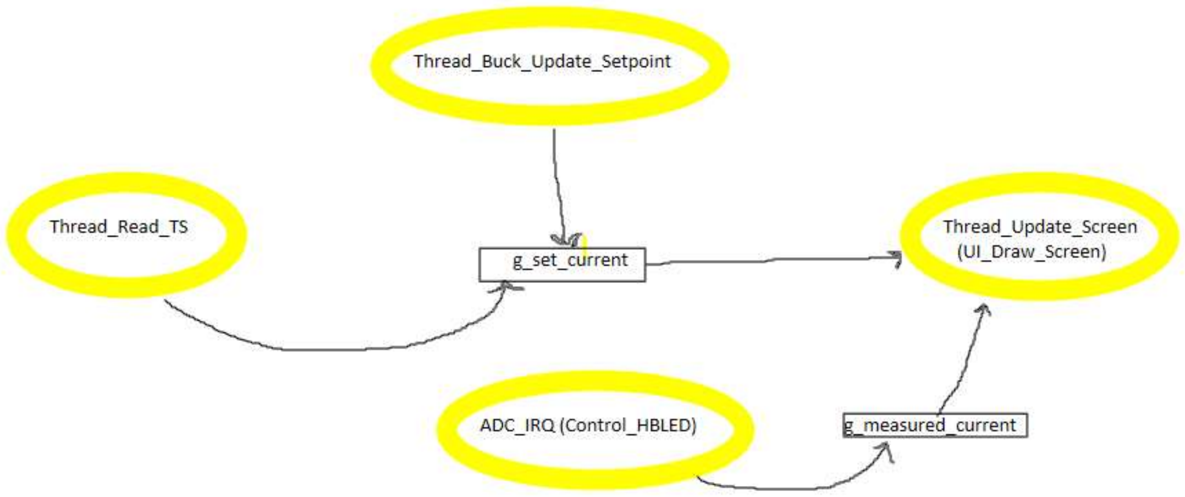
“Thread_Buck_Update_Setpoint” manages the HBLED’s current setpoint based on the above flash parameters to manage flashing. The “Update_Set_Current” manages the LED flashing by controlling the set current to vary among the peak current and zero value based on the flash delay parameter.

The flash delay down counts from 600 to 0 and the set current equals the peak current when the count reaches 100. Ie; the LED stays ON between delay count 100 to 0 and stays OFF for the remaining count.

Ton = 100ms, Toff = 500ms

Assumptions: When the enable flash is high, the flash delay count decrements within every 1ms period as mentioned in the starter code.

Therefore, in order to synchronize the plotting with the start of each flash/rising edge of flash it is necessary that plot values need to be gathered and ready for the plotting function. Therefore, a control flag global variable that sets a few milliseconds before the flash starts ON is utilized in managing to store the average sampled values of measured current to an array. Thus, once all the required 240-pixel data is gathered in the global array, the plotting thread can start plotting the array pixel values based on the X and Y coordinates as described in the function.



ADC_IRQ into Finite State Machine

When the PWM period is defined as 500 the period of counting, ie; the time duration between timer overflow is ~20us and the altogether time taken to run the ADC_IRQ is ~24 us. Hence, there is no sufficient time for the queued conversion and also there do exist high possibility to miss out the next priority conversion. Therefore, the ADC_IRQ handler is broken into finite state machine with two user defined states, one for priority and another for the queued conversion.

Priority case :

1. Invokes the Control_HBLED function.
2. Checks if there is any additional request message in the queue.
3. Get the request message and configure the ADC for software trigger.
4. Start the ADC conversion.

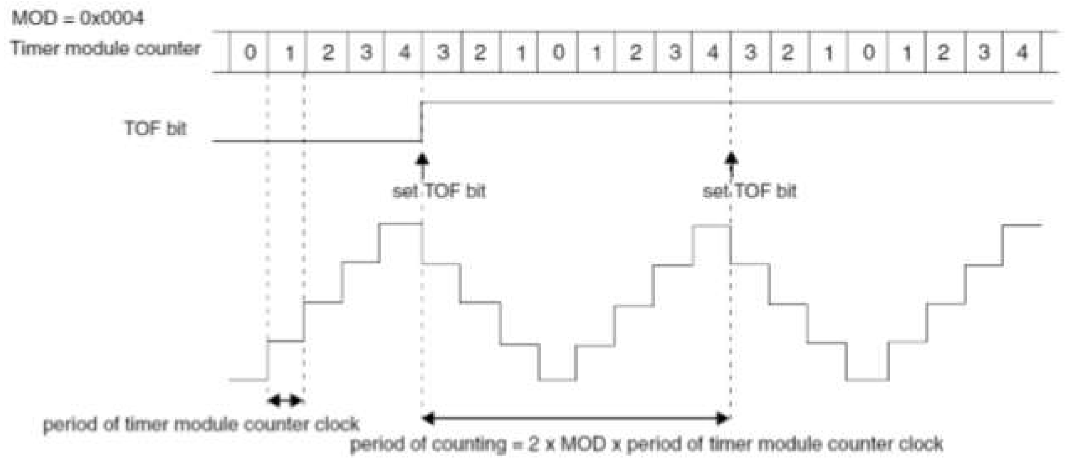
Queued case :

1. Enqueue the result message obtained after the previous conversion back into the result queue.
2. Configure the ADC for hardware triggering to enable the priority conversion.

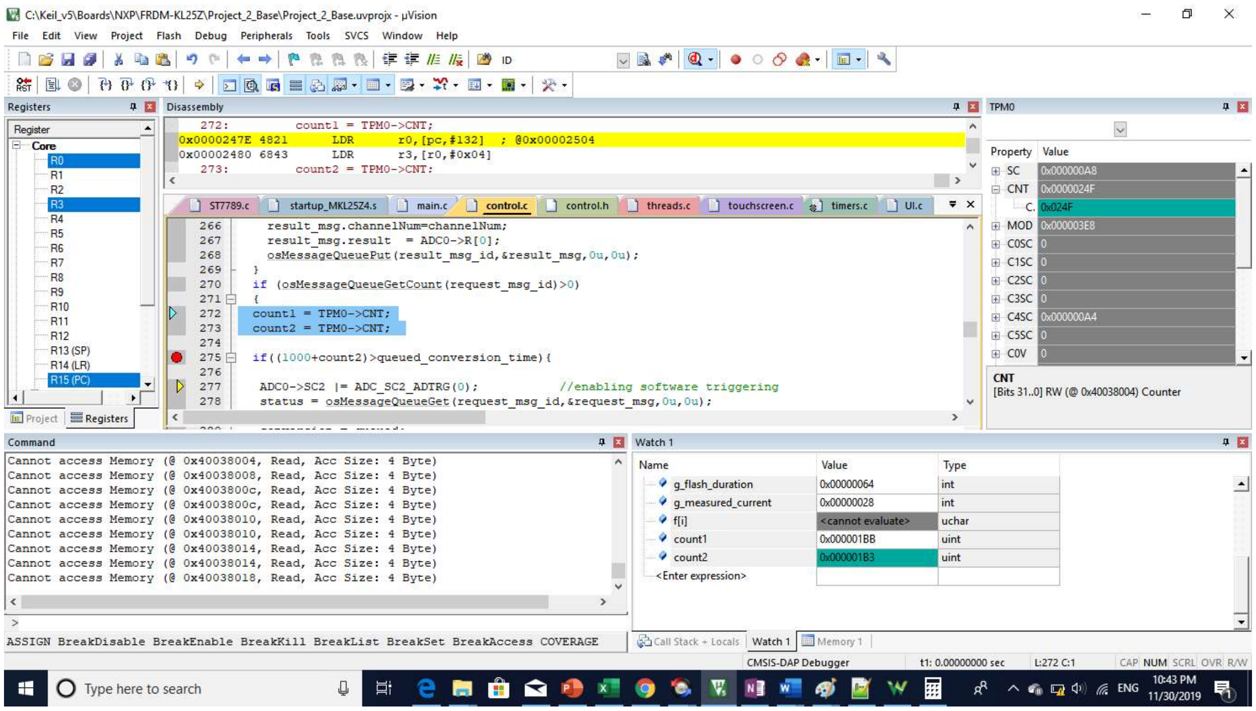
Extra Credit

1. The required time for the queued conversion is defined as the time duration between the start of the ADC conversion and the next timer overflow. To measure the remaining time for the queued conversion, CNT register of the TPM0 peripheral is monitored twice inside the ADC_IRQ handler while checking for any additional requests message in the queue. The reason for monitoring the values twice is to identify if the current count pattern is in incrementing/decrementing mode.

   Note : In this case, the PWM_PERIOD is changed to 1000.


MOD = 0x0004
Timer module counter
0 1 2 3 4 3 2 1 0 1 2 3 4 3 2 1 0 1 2 3 4
TOF bit
set TOF bit
set TOF bit
period of timer module counter clock
period of counting = 2 x MOD x period of timer module counter clock

From the below screenshot it is evident that the CNT is decreasing by monitoring the count1 and count2 global variables from the watch window. So, the remaining count for the next timer overflow is calculated as “1000 + count2”.


C:\Keil_v5\Boards\NXP\FRDM-KL25Z\Project_2_Base\Project_2_Base.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
TPM0
ST7789.c startup_MKL25Z4.s main.c control.c control.h threads.c touchscreen.c timers.c UI.c
Command
Watch 1
Project Registers
Call Stack + Locals Watch 1 Memory 1
CMSIS-DAP Debugger t1: 0.00000000 sec L:272 C:1
Type here to search
10:43 PM 11/30/2019

```
   272:          count1 = TPM0->CNT;
0x0000247E 4821      LDR      r0,[pc,#132]  ; @0x00002504
0x00002480 6843      LDR      r3,[r0,#0x04]
   273:          count2 = TPM0->CNT;
```

```
    result_msg.channelNum=channelNum;
    result_msg.result  = ADC0->R[0];
    osMessageQueuePut(result_msg_id,&result_msg,0u,0u);
  }
  if (osMessageQueueGetCount(request_msg_id)>0)
  {
  count1 = TPM0->CNT;
  count2 = TPM0->CNT;

  if((1000+count2)>queued_conversion_time){

   ADC0->SC2  |= ADC_SC2_ADTRG(0);          //enabling software triggering
   status = osMessageQueueGet(request_msg_id,&request_msg,0u,0u);
```

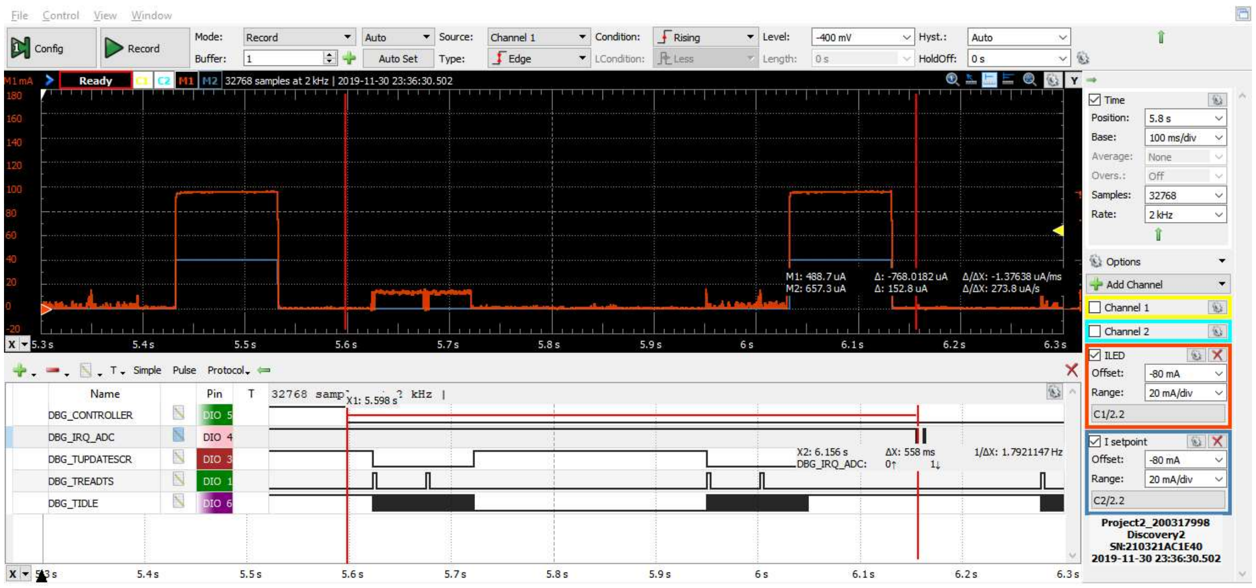
Property	Value
SC	0x000000A8
CNT	0x0000024F
C.	0x024F
MOD	0x000003E8
C0SC	0
C1SC	0
C2SC	0
C3SC	0
C4SC	0x000000A4
C5SC	0
C0V	0

CNT
[Bits 31..0] RW (@ 0x40038004) Counter

```
Cannot access Memory (@ 0x40038004, Read, Acc Size: 4 Byte)
Cannot access Memory (@ 0x40038008, Read, Acc Size: 4 Byte)
Cannot access Memory (@ 0x4003800c, Read, Acc Size: 4 Byte)
Cannot access Memory (@ 0x4003800c, Read, Acc Size: 4 Byte)
Cannot access Memory (@ 0x40038010, Read, Acc Size: 4 Byte)
Cannot access Memory (@ 0x40038010, Read, Acc Size: 4 Byte)
Cannot access Memory (@ 0x40038014, Read, Acc Size: 4 Byte)
Cannot access Memory (@ 0x40038014, Read, Acc Size: 4 Byte)
Cannot access Memory (@ 0x40038018, Read, Acc Size: 4 Byte)
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
```

Name	Value	Type
g_flash_duration	0x00000064	int
g_measured_current	0x00000028	int
f[i]	<cannot evaluate>	uchar
count1	0x000001BB	uint
count2	0x000001B3	uint
<Enter expression>		

Now the queued conversion time is calculated using debug signals, ie; by setting the debug bit when the software triggering is initiated and clearing the debug bit when the next ADC IRQ handler gets called. This analysis of the debug signal gives the time for the queued conversion. The remaining time duration for the queued conversion is found to be around 560 from the below screenshot which gives the measurement of the left out time after the Control HBLED completed its execution and the interrupt handler gets invoked.


Mode: Record
Buffer: 1
Auto
Source: Channel 1
Type: Edge
Condition: Rising
LCondition: Less
Level: -400 mV
Length: 0 s
Hyst.: Auto
HoldOff: 0 s

Ready | 32768 samples at 2 kHz | 2019-11-30 23:36:30.502

M1: 488.7 uA
M2: 657.3 uA
Δ: -768.0182 uA
Δ: 152.8 uA
Δ/ΔX: -1.37638 uA/ms
Δ/ΔX: 273.8 uA/s

Time
Position: 5.8 s
Base: 100 ms/div
Average: None
Overs.: Off
Samples: 32768
Rate: 2 kHz

ILED
Offset: -80 mA
Range: 20 mA/div
C1/2.2

I setpoint
Offset: -80 mA
Range: 20 mA/div
C2/2.2

X1: 5.598 s
X2: 6.156 s
DBG_IRQ_ADC:
ΔX: 558 ms
1/ΔX: 1.7921147 Hz

Name	Pin
DBG_CONTROLLER	DIO 5
DBG_IRQ_ADC	DIO 4
DBG_TUPDATESCR	DIO 3
DBG_TREADTS	DIO 1
DBG_TIDLE	DIO 6

Project2_200317998
Discovery2
SN:210321AC1E40
2019-11-30 23:36:30.502

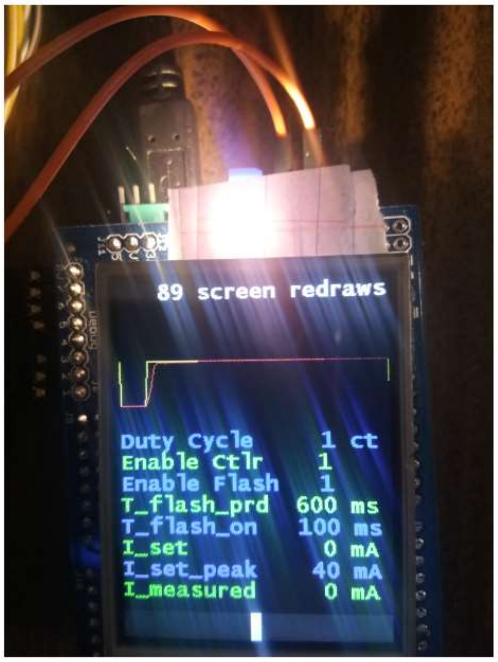
Therefore, if the above calculated value (1000 +count2) is greater than the queued conversion time then the request message in the queue is processed for ADC conversion while not missing the next timer overflow to switch back to the priority conversion.

Note : The code supports this version of code if USE_ADC_TPM_CNT is non-zero and PWM_PERIOD is changed to 1000, then the code checks if there exists sufficient time to service the queued message by monitoring the CNT register of TPM0. Also, the default version is if USE_ADC_FSM is non-zero where the ADC ISR is broken in FSM.

2. Please subscribe to the below video link that shows the system operation. The link shows the LED flashing even when the UI parameters are changed. The synchronization among the graphical current plot and LED flashing can also be observed in this link.

   Video link : https://youtu.be/0jLRtDvHa7g

3. The color scheme of the default UI parameters on the LCD screen are changed to accommodate different colors by updating the parameters of the structure “UI_FIELD_T “ which is passed as an argument to the function “UI_Update_Field_Values” invoked when “Thread_Update_Screen” is in running model.



```
    89 screen redraws

Duty Cycle       1 ct
Enable Ctlr      1
Enable Flash     1
T_flash_prd    600 ms
T_flash_on     100 ms
I_set            0 mA
I_set_peak      40 mA
I_measured       0 mA
```

THE END***********************************