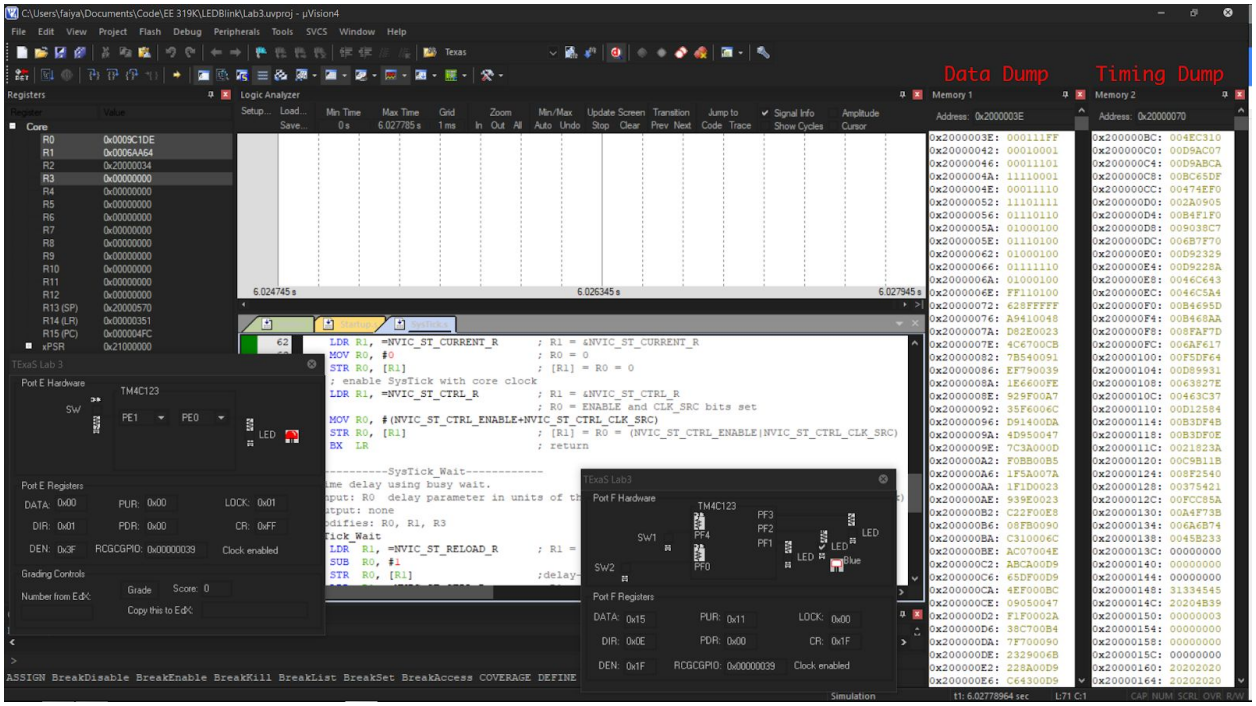
Faiyaz Mostofa
Zaine Ahmed



```
;****************** main.s ***************
; Program written by: Faiyaz Mostofa & Zain Ahmed
; Date Created: 2/4/2017
; Last Modified: 2/28/2018
; Brief description of the program:
;   A collection of operations are done in this program (at once):
;          - An internal Blue LED blinks repeatedly indicating that the program is running
;          - A Red LED blinks on and off based on different patterns which can be changed by a
switch (8Hz).
;            The LED will cycle through different patterns as specified below:
;              :on for (1/40s) -> on for (1/20s) -> on for (3/40s) -> on for (1/10s) -> always on ->
always off -> loop:
;            The delays for the blinking is handled by SysTick.s
;          - The Red LED also has a breathing function can be activated by holding the internal
button @ PF4 on the micro-controller.
;            The delays for the breathing is handled by the delay function @ the end of this file.
;   - A debugging capture tool that runs 3rd time the loop finishes.
; Hardware connections (External: One button and one LED)
;  PE1 is Button input  (1 means pressed, 0 means not pressed)
;  PE0 is a Red LED output (1 activates external LED on protoboard)
;  PF2 is a Green LED output (1 activates external LED on protoboard)
;  PF4 is builtin button SW1 on Launchpad (Internal)
;        Negative Logic (0 means pressed, 1 means not pressed)
```

Faiyaz Mostofa
Zaine Ahmed

```
;  Main Loop takes 925 cycles before debug_capture takes 22 cycles, for mere 2.3%
intrusiveness

; PortE device registers
GPIO_PORTE_DATA_R  EQU 0x400243FC
GPIO_PORTE_DIR_R   EQU 0x40024400
GPIO_PORTE_AFSEL_R EQU 0x40024420
GPIO_PORTE_DEN_R   EQU 0x4002451C
; PortF device registers
GPIO_PORTF_DATA_R  EQU 0x400253FC
GPIO_PORTF_DIR_R   EQU 0x40025400
GPIO_PORTF_AFSEL_R EQU 0x40025420
GPIO_PORTF_PUR_R   EQU 0x40025510
GPIO_PORTF_DEN_R   EQU 0x4002551C
GPIO_PORTF_LOCK_R  EQU 0x40025520
GPIO_PORTF_CR_R    EQU 0x40025524
GPIO_LOCK_KEY      EQU 0x4C4F434B     ; Unlocks the GPIO_CR register
SYSCTL_RCGCGPIO_R  EQU 0x400FE608
; System Clock reigsters
NVIC_ST_CURRENT_R     EQU 0xE000E018

;Variables that hold the maximum values
MAX_DELAY               EQU 0x1864A8        ; The interval size of the delays (in cycles)
                                                        ;       0x0c (in 10ms)
BREATHE_DELAY_MAX  EQU 0x5E00        ; The delay required

    IMPORT TExaS_Init
          IMPORT SysTick_Init

    THUMB
;------------Global Variables----------------------------------------------------------
    AREA    DATA, ALIGN=2

          ;Blinking variables
delay_inc                SPACE 4      ; how to increment the delays when we need to change
them (1/5 of MAX_DELAY)
delay_off                SPACE 4      ; how long the LED will stay off (in cycles)
delay_on                 SPACE 4      ; how long the LED will stay on (in cycles)
prev_button_state SPACE     1        ; captures whether a button has been released or pushed
green_counter  SPACE 1      ; it counts everytime the main loop is run and toggles the blue LED
after a certain time is met.

          ;Debugging arrays
```

Faiyaz Mostofa
Zaine Ahmed

```
data_capture   SPACE 50      ; Array of 50 8-byte numbers
                                                    ; Start: 0x2000003f
                                                    ; End:   0x20000090
time_capture   SPACE 200     ; Array of 50 32-byte numbers
                                                    ; Start: 0x20000074
                                                    ; End:   0x20000138
debug_capture_counter        SPACE          1       ; it counts everytime the main loop is run
and captures debugging data after a certain amount of loops

NEntries                SPACE 1                 ; Number of entries in either array

    AREA    |.text|, CODE, READONLY, ALIGN=2
    THUMB
    EXPORT  Start

;---------Main Code-------------------------------------------------------------------------
Start
 ; TExaS_Init sets bus clock at 80 MHz
   BL  TExaS_Init ; voltmeter, scope on PD3
   BL  Debug_Init ;     Initializes the Debugging Tools
        BL      SysTick_Init;   Initializes the SysTick (method in SysTick.s)
        BL      Ports_Init;             Initializes Ports E,F
                                        ; PE0 = Red LED output
                                        ; PE1 = positive logic Input (Switch)
                                        ; PF2 = Blue LED output
                                        ; PF4 = Hold switch for breathing functionality

; Setting up variables
Configure
        LDR    R1, =MAX_DELAY;
        MOV    R2, #5;
        UDIV R2, R1, R2;                split the max delay into 5 equal sections
        LDR    R1, =delay_inc;
        STR    R2, [R1];                        delay_inc = (MAX_DELAY / 5)

        LDR    R1, =delay_inc;
        LDR    R2, [R1];
        MOV    R3, #4;
        MUL    R2, R2, R3;
        LDR    R1, =delay_off;
        STR    R2, [R1];                        Default: the delay_off starts @ 4/5 of the
MAX_DELAY
```

Faiyaz Mostofa
Zaine Ahmed

```
        LDR    R1, =delay_inc;
        LDR    R2, [R1];
        LDR R1, =delay_on;
        STR    R2, [R1];                    Default: the delay_on on starts @ 1/5 of the
MAX_DELAY

        LDR    R1, =green_counter;
        MOV    R2, #0;
        STRB R2, [R1];                      Initially set the green_counter to 0
        LDR    R1, =debug_capture_counter;
        MOV    R2, #0;
        STRB R2, [R1];                      Initially set the debug_counter to 0

        MOV R12, #0;                R12 will be used to quickly collect the seven states after
release

   CPSIE  I    ; TExaS voltmeter, scope runs on interrupts

; The main loop engine
main_loop

        BL      Check_Green         ; Check whether to toggle the green LED on or not
        BL      Check_Breathe       ; Check if whether we need to make the LED Breathe

;If a button @ PE1 is pushed, increment the blinking pattern
Blink_ifPushed
        LDR    R1, =GPIO_PORTE_DATA_R;
        LDR R2, =prev_button_state;
        LDRB R2, [R2];
        LDR    R3, [R1];                        <- R3 holds the data from the PortE data
register
        AND    R3, R3, #0x02;                   Check whether the button has been pushed
or not
        CMP R3, R2;                             <- Check if the button is in the same state
as before
        BEQ Blink;
        LDR    R2, =prev_button_state;
        STRB R3, [R2];
; If the button is pushed, set PE4 to 1
        CMP    R3, #0x00;                   If the button is released
        BNE    Collect_skip;
        CMP    R12, #0;
        BNE    Collect_skip;
```

Faiyaz Mostofa
Zaine Ahmed

```
        MOV   R12, #7;
Collect_skip
        CMP   R3, #0x02;                    If the button is pushed
        BNE   Blink_incrementDuty;
        BL    Debug_Capture;
        B Blink;
Blink_incrementDuty
; Incrementing the duty time
        LDR   R2, =delay_inc;
        LDR   R2, [R2];
        LDR   R1, =delay_off;
        LDR   R3, [R1];
        SUB   R3, R3, R2;                       Decrement the off time
        STR   R3, [R1];
        LDR   R1, =delay_on;
        LDR   R3, [R1];
        ADD   R3, R3, R2;                       Increment the on time
        STR R3, [R1];
;Check if the duty time needs to be reset (always on -> always off)
        LDR   R1, =delay_off;
        LDR   R2, [R1];
        CMP   R2, #0;
        BPL   Blink;                        If the the off time is < 0 (off < 0%, on > 100%),
reset the values to off = 100%, on = 0%
        LDR   R2, =MAX_DELAY;
        LDR R1, =delay_off;
        STR R2, [R1];
        LDR   R1, =delay_on;
        MOV   R2, #0;                           Reset the on time to 0 (light is always off)
        STR   R2, [R1];
Blink
; Turn off the light and wait
        CMP   R12, #0;
        BEQ   Blink_Off_Cap_Skip
        BL Debug_Capture;
        SUB   R12, R12, #1;
Blink_Off_Cap_Skip
        LDR   R1, =GPIO_PORTE_DATA_R;
        LDR   R2, [R1];
        BIC   R2, #0x01;
        STR   R2, [R1];
        LDR R2, =delay_off;
        LDR R0, [R2];
```

Faiyaz Mostofa
Zaine Ahmed

```
        BL      delay;  ;BL     delay;  Delay the program for a amount of time specified in R0
; Turn on the light and wait
        CMP    R12, #0;
        BEQ    Blink_On_Cap_Skip
        BL Debug_Capture;
        SUB    R12, R12, #1;
Blink_On_Cap_Skip
        LDR    R1, =GPIO_PORTE_DATA_R;
        LDR    R2, [R1];
        ORR    R2, #0x01;
        STR    R2, [R1];
        LDR R2, =delay_on;
        LDR R0, [R2];
        BL      delay   ;BL     delay

   B   main_loop
;-------------------------------------------------------------------------------------------
Breathe_Start
; a subroutine that handles all the breathing functionality by completly reworking everything
        PUSH {R0-R7};
        PUSH {R8, LR};

        ; Setting up variables
        LDR R0, =GPIO_PORTE_DATA_R;
        LDR    R9, =GPIO_PORTF_DATA_R;
        LDR    R2, =BREATHE_DELAY_MAX;
        MOV    R3, #500;
        UDIV R4, R2, R3;                    The increments of the delay
        ADD    R5, R2, #0;                      Default: off for 4/5 of 80Hz
        MOV    R6, #0;                          Default: on for 1/5 of 80Hz
        ADD    R7, R2, #0;

Breathe_loop
        LDR    R1, [R9];                        <- R1 holds the data from the data register
Breathe_ifPushed
        AND    R3, R1, #0x10;                   Check whether the button has been pushed
or not
        CMP    R3, #0x10;                       Keep Breathing until the button is released.
        BNE    Breathe_incrementDuty;
        B Breathe_Stop;

Breathe_incrementDuty
; Incrementing the duty time
```

Faiyaz Mostofa
Zaine Ahmed

```
        SUB    R5, R5, R4;                  Decrement the off time
        ADD    R6, R6, R4;                  Increment the on time
        CMP    R5, #0;
        BMI Breathe_Verse;
        BEQ Breathe_Verse;                  Check if we've stopped or froze the delay of the
light (either R5 or R6 reach zero)
        CMP    R6, #0;
        BPL Breathe;
Breathe_Verse
        MOV    R3, #-1;
        MUL    R4, R4, R3;                  Once we reach a maximum, down/up or up
depending on the scenario
        SUB    R5, R5, R4;                  Decrement the off time
        ADD    R6, R6, R4;                  Increment the on time
Breathe
; Turn off the light and wait
        BIC     R1, #0x01;
        STR     R1, [R0];
        PUSH {R0, R1};
        ADD     R0, R5, #0;
        BL      delay;                       Delay the program for a amount of time
specified in R7
        POP {R0, R1};
; Turn on the light and wait
        ORR     R1, #0x01;
        STR     R1, [R0];
        PUSH {R0, R1};
        ADD     R0, R6, #0;
        BL      delay;
        POP {R0, R1};

   B   Breathe_loop

Breathe_Stop
        POP {R8,LR};
        POP {R0-R7};

        BX LR;

;-------CHECK_debug-----------------------------------------------------------------------
; Wait 5 duty cycles, then save the points in the Dubugging arrays
Check_Debug
        PUSH {R0, R1};
```

Faiyaz Mostofa
Zaine Ahmed

```
        PUSH {R2, LR};
        LDR    R1, =debug_capture_counter;
        LDRB R2, [R1];
        ADD    R2, R2, #1;                    debug_capture_counter++;
        STRB R2, [R1];
        CMP    R2, #6;
        BNE    Check_Debug_Leave;
        BL     Debug_Capture;                 if(debug_capture_counter == 3) capture data
        MOV    R2, #0;
        STRB R2, [R1];
Check_Debug_Leave
        POP    {R2, LR};
        POP {R0, R1};
        BX LR;

;-------CHECK_Green-----------------------------------------------------------------------
; Wait 5 duty cycles, then save the points in the Dubugging arrays
Check_Green
        PUSH {R0, R1};
        PUSH {R2, LR};
        LDR    R1, =green_counter;
        LDRB R2, [R1];
        ADD    R2, R2, #1;                    green_counter++
        STRB R2, [R1];
        CMP    R2, #3;
        BNE    Check_Green_Leave;
        BL     Toggle_Green;                  if(green_counter == 3) toggle Green LED
        MOV    R2, #0;
        STRB R2, [R1];
Check_Green_Leave
        POP    {R2, LR};
        POP {R0, R1};
        BX LR;

;-------CHECK_Breathe---------------------------------------------------------------------
; If the button @ PF4 is pushed, Start breathing
Check_Breathe
        PUSH {R0, R1};
        PUSH {R2, LR};
        LDR    R1, =GPIO_PORTF_DATA_R;
        LDR    R2, [R1];
        AND    R2, R2, #0x10;                       Check whether the button has been pushed
or not
```

Faiyaz Mostofa
Zaine Ahmed

```
        CMP    R2, #0x00;
        BNE    Check_Breathe_Leave;        If SW1 is pushed, start the breathing
        BL Breathe_Start;
Check_Breathe_Leave
        POP    {R2, LR};
        POP {R0, R1};
        BX      LR;

;-------DEBUG_Init------------------------------------------------------------------------
   ;Initiliazing Debug Dump
Debug_Init
        PUSH {R0, R1}
        PUSH {R2, R3}
        LDR R2, =data_capture;
        LDR R3, =time_capture;                  Created pointers
; Fill the data array with 0xFF (signifying empty)
        MOV R0, #50;
setting_data_capture
        SUB R0,R0, #0x01
        MOV R1, #0xFF;
        STRB R1, [R2]
        ADD R2, R2, #1;
        CMP R0, #0x0;
        BNE setting_data_capture
 ; Fill the time array with 0xFFFFFFFF (signifying empty)
        MOV R0, #50;
setting_time_capture
        SUB R0,R0, #1;
        MOV R1, #0xFFFFFFFF;
        STR R1, [R3]
        ADD R3, R3, #4;
        CMP R0, #0x0;
        BNE setting_time_capture
RestNEntries
        LDR    R0, =NEntries;
        MOV    R2, #0;
        STRB R2, [R0];

        POP {R2, R3}
        POP {R0, R1}
        BX LR

;-------DEBUG_CAPTURE---------------------------------------------------------------------
```

Faiyaz Mostofa
Zaine Ahmed

```
; saves one data point
Debug_Capture
        PUSH {R0,R1}
        PUSH {R2,LR}
        LDR R0 , =NEntries
        LDRB R1, [R0]
        CMP R1 , #50
        BHS DONE_C;                     if (the array is not full)
        ADD    R1, R1, #1;              Add a new entry
        STRB R1, [R0];                  NEntries++;
; Record the current data entries
        LDR R0, =GPIO_PORTE_DATA_R;
        LDR    R0, [R0];
        ADD    R1, R0, #0;
        AND R0, R0, #0x01;          R0 holds the data for PE0
        AND    R1, R1, #0x02;           R1 holds the data for PE1
        LSL R1, R1, #3;                 Move PE1 to PE4
        ORR    R1, R1, R0;              Merge the two bits (PE0 | PE4)
        LDR R0, =data_capture;
        LDR    R2, =NEntries;
        LDRB R2, [R2];
        ADD    R0, R0, R2;
        STRB R1, [R0];                  Store the value in the correct spot on the data array
; Record the current time
        MOV    R0, #4;
        MUL R2, R2, R0;                 Increment in the time array by 4 bytes
        LDR    R1, =time_capture;
        ADD    R1, R1, R2;
        LDR R0, =NVIC_ST_CURRENT_R
        LDR R0, [R0];
        STR    R0, [R1];                Store the current time in the correct spot on the
time array
; Restore the registers and leave
DONE_C
        POP {R2, LR};
        POP {R0,R1}
        BX LR;

;-------Toggle Green LED (PF2)--------------------------------------------------------------
;Toggles the Green LED on and off (PF2)
Toggle_Green
        PUSH {R0, R1};
        LDR    R0, =GPIO_PORTF_DATA_R;
```

Faiyaz Mostofa
Zaine Ahmed

```
        LDR    R1, [R0];
        EOR    R1, #0x04;
        STR    R1, [R0];
        POP {R0, R1};
        BX LR;

;-------------------------------------------------------------------------------------------
delay
; a subroutine that loops using the value at R0
        PUSH {R0, R1};
        MOV    R1, #0;
delayLoop
        CMP    R0, R1;                          Loop until temporary value, R1, reaches R0
        BEQ    delayDone;
        ADD    R1, R1, #1;
        B      delayLoop;
delayDone
        POP {R0, R1};
        BX LR;

;-------------------------------------------------------------------------------------------
; Port Initialization
Ports_Init
        PUSH {R0, R1};
        PUSH {R2, LR};
        LDR    R0, =SYSCTL_RCGCGPIO_R;
        LDR    R1, [R0];
        ORR    R1, R1, #0x30;                   Start up Port F and Port E
        STR    R1, [R0];
        NOP;
        NOP;
 ; Configure Port E
        LDR    R0, =GPIO_PORTE_DIR_R;
        LDR    R1, [R0];
        ORR    R1, R1, #0x01;                       PE0 is set to output (LED)
        BIC    R1, R1, #0x12;                       PE1,4 are set to input (buttons)
        STR    R1, [R0];
        LDR    R0, =GPIO_PORTE_AFSEL_R;
        LDR    R1, [R0];
        MOV    R1, #0;                              Disables the "alternate functions" in
the port
        STR    R1,     [R0];
        LDR    R0, =GPIO_PORTE_DEN_R;
```

Faiyaz Mostofa
Zaine Ahmed

```
        LDR    R1, [R0];
        MOV    R1, #0xFF;                                  1 means enable digital I/O
        STR    R1, [R0];
; Configure Port F
        LDR R1, =GPIO_PORTF_LOCK_R; 2) unlock the lock register
        LDR R0, =GPIO_LOCK_KEY;                            unlock GPIO Port F Commit
Register
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_CR_R;      enable commit for Port F
        MOV R0, #0xFF;                 1 means allow access
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_DIR_R;     5) set direction register
        MOV R0,#0x0E;
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_AFSEL_R;           6) regular port function
        MOV R0, #0;                    0 means disable alternate function
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_PUR_R;             pull-up resistors for PF4,PF0
        MOV R0, #0x11;                 1)enable for negative logic
        STR R0, [R1];
        LDR R1, =GPIO_PORTF_DEN_R;             7) enable Port F digital port
        MOV R0, #0xFF;                 1 means enable digital I/O
        STR R0, [R1];
        POP  {R2, LR};
        POP     {R0, R1};
        BX LR;

;------------------------------------------------------------------------------------------
   ALIGN      ; make sure the end of this section is aligned
   END        ; end of file
```

Frequency Calculations:

Time array captures after 6 main loops:

Time_array[0] = 0x02430A4

Time_array[1] = 0x0B604F4

Subtracting the times: 0x0B604F4 - 0x02430A4

= 0x091D450 ← Change in time over 6 main loops (in cycles)

= 9,557,072 (in decimal form)

Faiyaz Mostofa
Zaine Ahmed

Divide the value by 6: 0x091D450 / 6

= 1,592,845.33 ← Change in time over 1 main loop (in decimal)

8,000,000 cycles = 1 sec

secs = 1,592,845 / 8* 10^9

Secs = .19910562

Period = 1/ secs

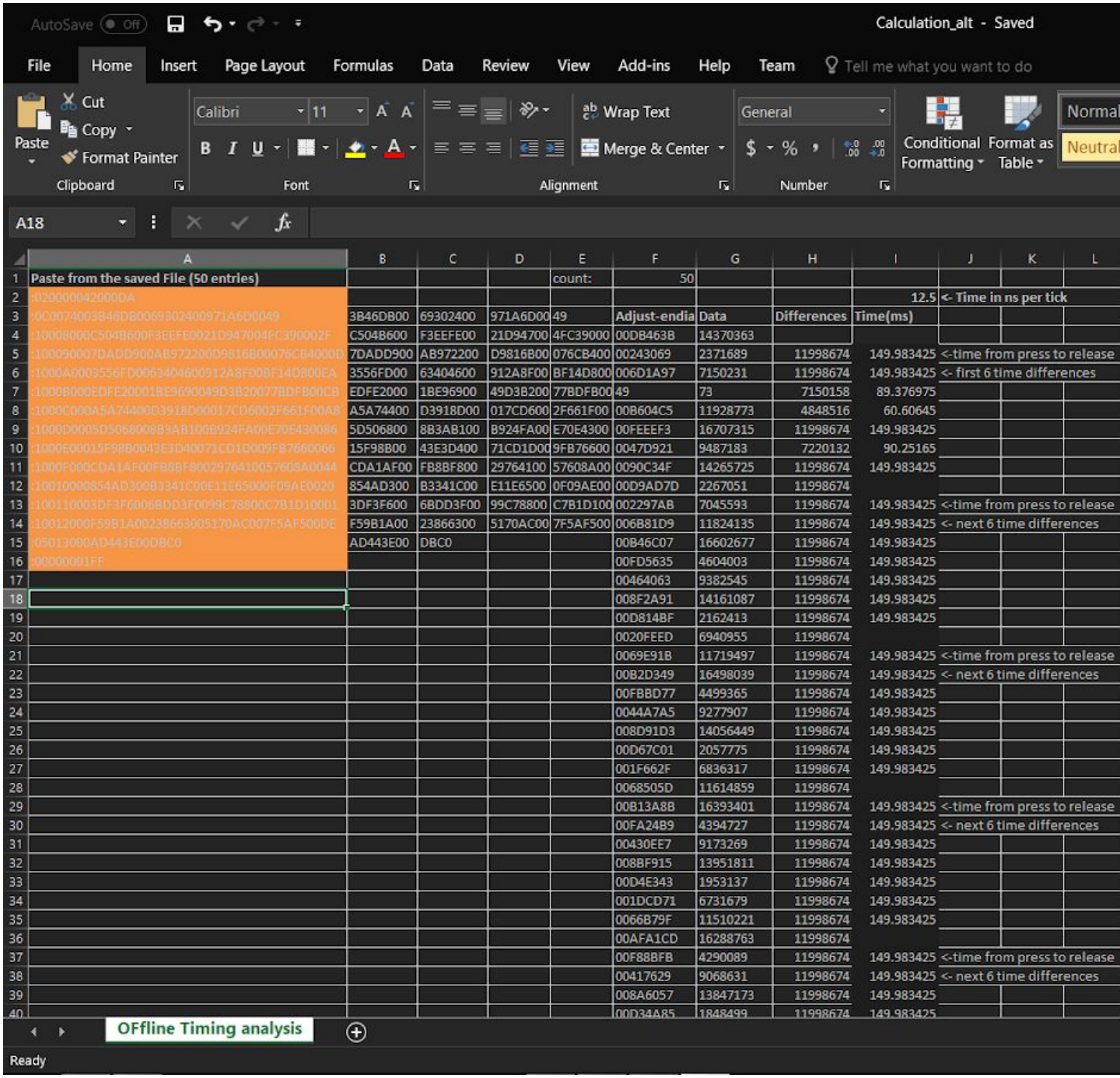
Period = 5 hz

Intrusiveness calculations

((2 cycles / instruction) * (22 instructions / debug_capture) / (1 / 8 period time for LED)) * (12.5 ns / clock cycle) = 4.4e-6 s = 0.0000044 secs

---

Faiyaz Mostofa
Zaine Ahmed



AutoSave Off — Calculation_alt - Saved

File | Home | Insert | Page Layout | Formulas | Data | Review | View | Add-ins | Help | Team | Tell me what you want to do

Paste | Cut | Copy | Format Painter | Clipboard | Calibri | 11 | Font | Wrap Text | Merge & Center | Alignment | General | Number | Conditional Formatting | Format as Table | Normal | Neutral

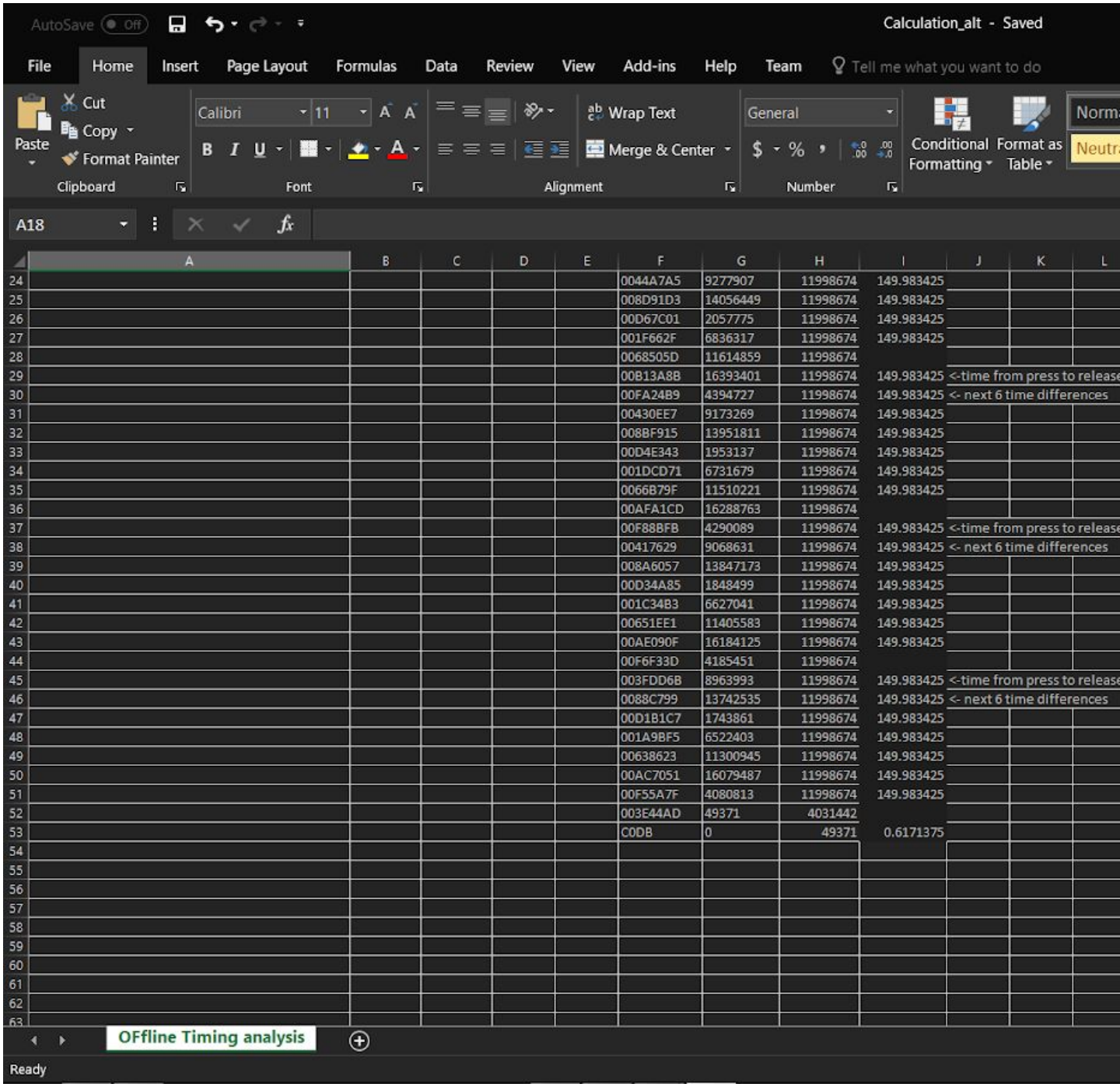
A18

	A	B	C	D	E	F	G	H	I	J	K	L
1	Paste from the saved File (50 entries)				count:	50						
2	:020000042000DA								12.5	<- Time in ns per tick		
3	:0C0074003B46DB0069302400971A6D0049	3B46DB00	69302400	971A6D00	49	Adjust-endia	Data	Differences	Time(ms)			
4	:10008000C504B600F3EEFE0021D947004FC390002F	C504B600	F3EEFE00	21D94700	4FC39000	00DB463B	14370363					
5	:100090007DADD900AB972200D9816B00076CB4000D	7DADD900	AB972200	D9816B00	076CB400	00243069	2371689	11998674	149.983425	<-time from press to release		
6	:1000A0003556FD00634046009 12A8F00BF14D800EA	3556FD00	63404600	912A8F00	BF14D800	006D1A97	7150231	11998674	149.983425	<- first 6 time differences		
7	:1000B000EDFE20001BE9690049D3B20077BDFB00CB	EDFE2000	1BE96900	49D3B200	77BDFB00	49	73	7150158	89.376975			
8	:1000C000A5A74400D3918D00017CD6002F661F00A8	A5A74400	D3918D00	017CD600	2F661F00	00B604C5	11928773	4848516	60.60645			
9	:1000D0005D5068008B3AB100B924FA00E70E430086	5D506800	8B3AB100	B924FA00	E70E4300	00FEEEF3	16707315	11998674	149.983425			
10	:1000E00015F98B0043E3D40071CD1D009FB7660066	15F98B00	43E3D400	71CD1D00	9FB76600	0047D921	9487183	7220132	90.25165			
11	:1000F000CDA1AF00FB8BF8002976410057608A0044	CDA1AF00	FB8BF800	29764100	57608A00	0090C34F	14265725	11998674	149.983425			
12	:10010000854AD300B3341C00E11E65000F09AE0020	854AD300	B3341C00	E11E6500	0F09AE00	00D9AD7D	2267051	11998674				
13	:100110003DF3F6006BDD3F0099C78800C7B1D10001	3DF3F600	6BDD3F00	99C78800	C7B1D100	002297AB	7045593	11998674	149.983425	<-time from press to release		
14	:10012000F59B1A00238663005170AC007F5AF500DE	F59B1A00	23866300	5170AC00	7F5AF500	006B81D9	11824135	11998674	149.983425	<- next 6 time differences		
15	:05013000AD443E00DBC0	AD443E00	DBC0			00B46C07	16602677	11998674	149.983425			
16	:00000001FF					00FD5635	4604003	11998674	149.983425			
17						00464063	9382545	11998674	149.983425			
18						008F2A91	14161087	11998674	149.983425			
19						00D814BF	2162413	11998674	149.983425			
20						0020FEED	6940955	11998674				
21						0069E91B	11719497	11998674	149.983425	<-time from press to release		
22						00B2D349	16498039	11998674	149.983425	<- next 6 time differences		
23						00FBBD77	4499365	11998674	149.983425			
24						0044A7A5	9277907	11998674	149.983425			
25						008D91D3	14056449	11998674	149.983425			
26						00D67C01	2057775	11998674	149.983425			
27						001F662F	6836317	11998674	149.983425			
28						0068505D	11614859	11998674				
29						00B13A8B	16393401	11998674	149.983425	<-time from press to release		
30						00FA24B9	4394727	11998674	149.983425	<- next 6 time differences		
31						00430EE7	9173269	11998674	149.983425			
32						008BF915	13951811	11998674	149.983425			
33						00D4E343	1953137	11998674	149.983425			
34						001DCD71	6731679	11998674	149.983425			
35						0066B79F	11510221	11998674	149.983425			
36						00AFA1CD	16288763	11998674				
37						00F88BFB	4290089	11998674	149.983425	<-time from press to release		
38						00417629	9068631	11998674	149.983425	<- next 6 time differences		
39						008A6057	13847173	11998674	149.983425			
40						00D34A85	1848499	11998674	149.983425			

OFfline Timing analysis

Ready

Faiyaz Mostofa
Zaine Ahmed



	A	B	C	D	E	F	G	H	I	J	K	L
24						0044A7A5	9277907	11998674	149.983425			
25						008D91D3	14056449	11998674	149.983425			
26						00D67C01	2057775	11998674	149.983425			
27						001F662F	6836317	11998674	149.983425			
28						0068505D	11614859	11998674				
29						00B13A8B	16393401	11998674	149.983425	<-time from press to release		
30						00FA24B9	4394727	11998674	149.983425	<- next 6 time differences		
31						00430EE7	9173269	11998674	149.983425			
32						008BF915	13951811	11998674	149.983425			
33						00D4E343	1953137	11998674	149.983425			
34						001DCD71	6731679	11998674	149.983425			
35						0066B79F	11510221	11998674	149.983425			
36						00AFA1CD	16288763	11998674				
37						00F88BFB	4290089	11998674	149.983425	<-time from press to release		
38						00417629	9068631	11998674	149.983425	<- next 6 time differences		
39						008A6057	13847173	11998674	149.983425			
40						00D34A85	1848499	11998674	149.983425			
41						001C34B3	6627041	11998674	149.983425			
42						00651EE1	11405583	11998674	149.983425			
43						00AE090F	16184125	11998674	149.983425			
44						00F6F33D	4185451	11998674				
45						003FDD6B	8963993	11998674	149.983425	<-time from press to release		
46						0088C799	13742535	11998674	149.983425	<- next 6 time differences		
47						00D1B1C7	1743861	11998674	149.983425			
48						001A9BF5	6522403	11998674	149.983425			
49						00638623	11300945	11998674	149.983425			
50						00AC7051	16079487	11998674	149.983425			
51						00F55A7F	4080813	11998674	149.983425			
52						003E44AD	49371	4031442				
53						C0DB	0	49371	0.6171375			
54												
55												
56												
57												
58												
59												
60												
61												
62												
63												