Dhruv Sandesara- djs3967

3/10/17

Homework 6. Loops and Arrays in C

Due: Wednesday 3/8 / Thursday 3/9 in Class **Problem6.2**: Write a function (`most_freq`), that takes an array of numbers and finds out the

most frequently occurring value in the array.

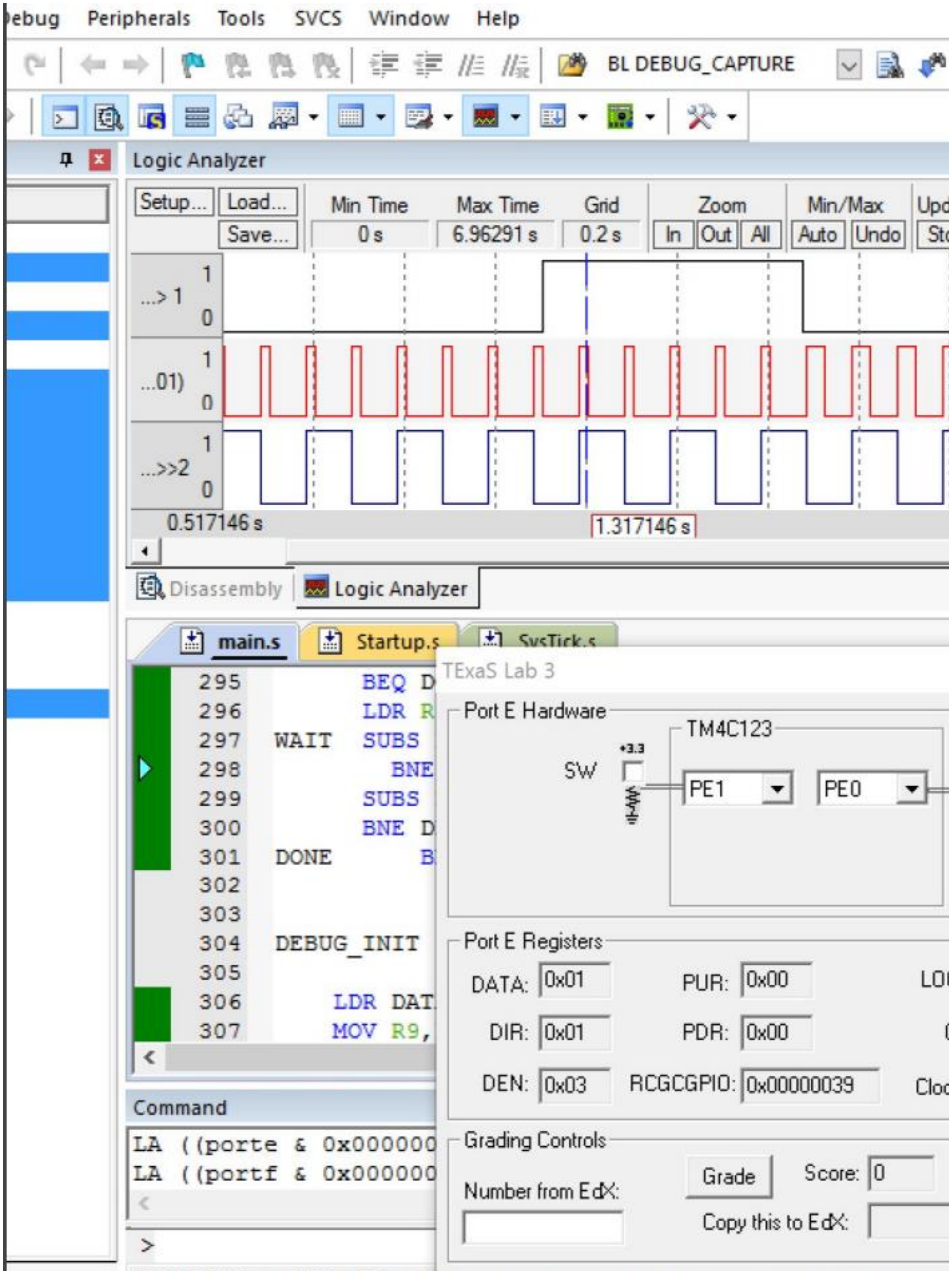
```
int  most_freq (int* input, int size);
void main() { int array[10] = {1,2,2,3,3,3,7,8,9,3}; int
freq = most_freq(array, 10);

print(freq); // assume this prints to an LED screen or an
output port }

int  most_freq (int* input, int size)
{
int count=1, tempCount;
int popular = input[0];
int temp = 0;
for (int i=0; i<size; i++)
{
if(temp== input[j])
{tempCount++;
}

if(tempCount > count)
{
popular= temp;
count=tempCount;
}
}
return popular;
} }
```

______

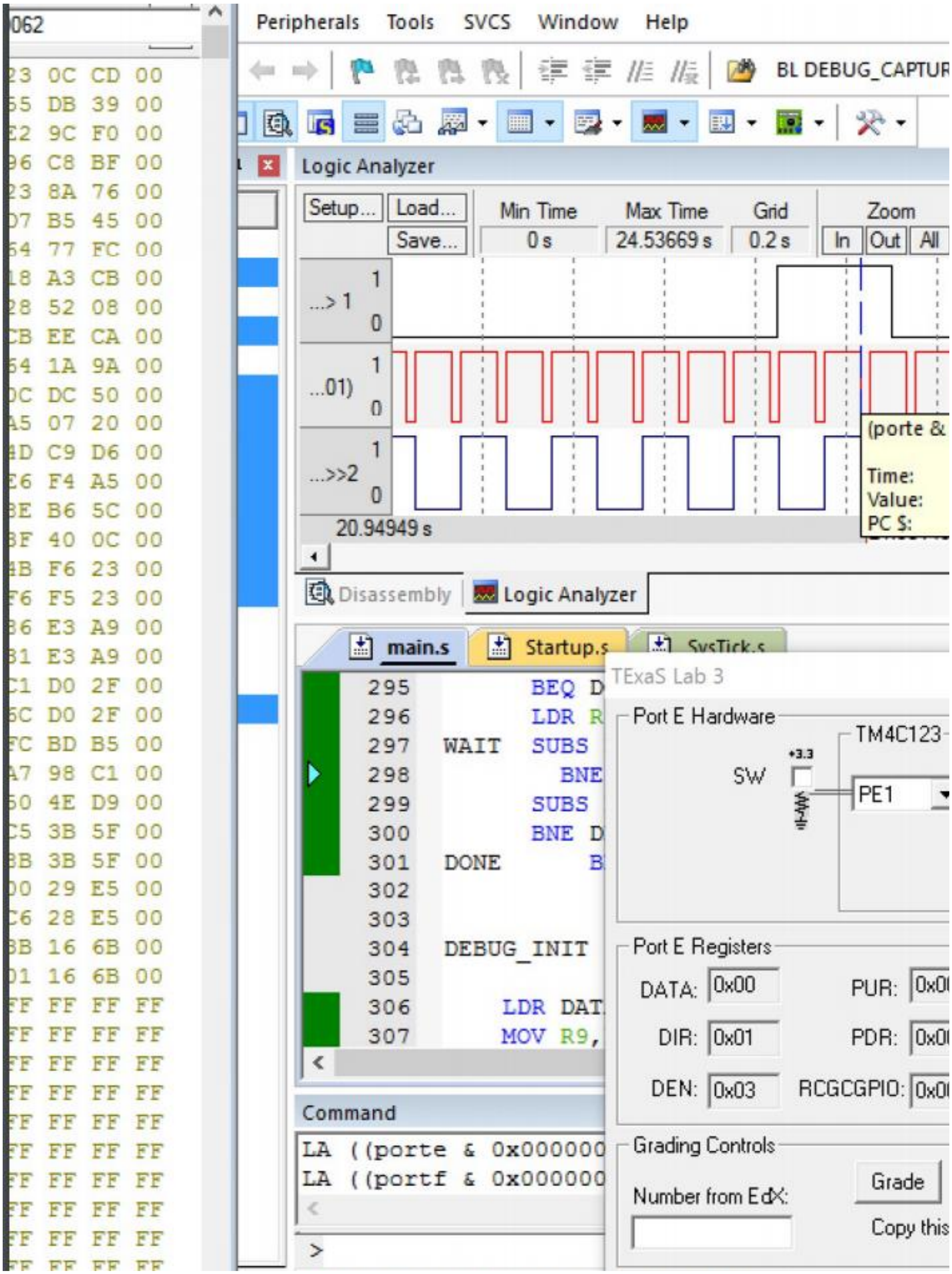


```
295         BEQ D
296         LDR R
297 WAIT    SUBS
298           BNE
299         SUBS
300         BNE D
301 DONE        B
302
303
304 DEBUG_INIT
305
306       LDR DAT
307       MOV R9,
```

DATA:	0x01	PUR:	0x00
DIR:	0x01	PDR:	0x00
DEN:	0x03	RCGCGPIO:	0x00000039

Score: 0

```
LA ((porte & 0x000000
LA ((portf & 0x000000
>
```



```
062
23 0C CD 00
55 DB 39 00
E2 9C F0 00
96 C8 BF 00
23 8A 76 00
07 B5 45 00
54 77 FC 00
18 A3 CB 00
28 52 08 00
CB EE CA 00
54 1A 9A 00
0C DC 50 00
A5 07 20 00
4D C9 D6 00
E6 F4 A5 00
BE B6 5C 00
BF 40 0C 00
4B F6 23 00
F6 F5 23 00
B6 E3 A9 00
B1 E3 A9 00
C1 D0 2F 00
5C D0 2F 00
FC BD B5 00
A7 98 C1 00
50 4E D9 00
C5 3B 5F 00
BB 3B 5F 00
D0 29 E5 00
C6 28 E5 00
BB 16 6B 00
01 16 6B 00
FF FF FF FF
FF FF FF FF
FF FF FF FF
FF FF FF FF
FF FF FF FF
FF FF FF FF
FF FF FF FF
FF FF FF FF
FF FF FF FF
FF FF FF FF
```

Min Time: 0 s Max Time: 24.53669 s Grid: 0.2 s Zoom

```
...> 1     1 / 0
...01)     1 / 0
...>>2     1 / 0
20.94949 s
(porte &
Time:
Value:
PC $:
```

```
                BEQ D
                LDR R
        WAIT    SUBS
                  BNE
                SUBS
                BNE D
        DONE        B


        DEBUG_INIT

             LDR DAT
             MOV R9,
```

```
LA ((porte & 0x000000
LA ((portf & 0x000000
>
```

PE1

DATA:	0x00	PUR:	0x0
DIR:	0x01	PDR:	0x0
DEN:	0x03	RCGCGPIO:	0x0

Number from EdX:

Copy this

```
/* main.c for Homework 6
Dhruv Sandesara djs3967
;
;
;
;
;
;
;
;
;
;
;
;
;
;
; 32-bit entry in the Time Buffer
Brief description of the program The LED toggles at 8
Hz and a varying duty-cycle Repeat the functionality
from Lab2-3 but now we want you to insert debugging
instruments which gather data (state and timing) to
verify that the system is functioning as expected.

Hardware connections (External: One button and one
LED) PE1 is Button input (1 means pressed, 0 means
not pressed) PE0 is LED output (1 activates external
LED on protoboard) PF2 is Blue LED on Launchpad used
as a heartbeat

Instrumentation data to be gathered is as
follows: After Button(PE1) press collect one state
and time entry. After Buttin(PE1) release, collect 7
state and time entries on each change in state of the
LED(PE0): An entry is one 8-bit entry in the Data
Buffer and one

The Data Buffer entry (byte) content has: Lower
nibble is state of LED (PE0) Higher nibble is state
```

```
of Button (PE1)

;
;
;
;  The Time Buffer entry (32-bit) has:
; ; ; ; ; ; */

24-bit value of the SysTick's Current register
(NVIC_ST_CURRENT_R) Note: The size of both buffers is
50 entries. Once you fill these

entries you should stop collecting data The heartbeat
is an indicator of the running of the program. On
each iteration of the main loop of your program
toggle the LED to indicate that your code(system) is
live (not stuck or dead).

// ***** 1. Pre-processor Directives Section *****
#include <stdint.h> #include "tm4c123gh6pm.h"

// ***** 2. Global Declarations Section *****

// FUNCTION PROTOTYPES: Each subroutine defined void
DisableInterrupts(void); // Disable interrupts void
EnableInterrupts(void); // Enable interrupts void
TExaS_Init(void); // ***** 3. Subroutines Section
***** void Ports_Init(void) {

volatile unsigned long delay; SYSCTL_RCGC2_R =
0x30; delay = SYSCTL_RCGC2_R;

//Init PortE GPIO_PORTE_DIR_R &= 0xFD;
GPIO_PORTE_DIR_R |=0x1; GPIO_PORTE_AFSEL_R &= 0xFC;
GPIO_PORTE_DEN_R |= 0x3;

//Init PortF GPIO_PORTF_DIR_R &= 0xFD;
GPIO_PORTF_DIR_R |=0x4; GPIO_PORTF_AFSEL_R &= 0xFB;
GPIO_PORTF_DEN_R |= 0x4; GPIO_PORTF_PUR_R &= 0xFB;
}

// Initialize SysTick with busy wait running at bus
clock. void SysTick_Init(void)

{
  NVIC_ST_CTRL_R = 0;
NVIC_ST_RELOAD_R = 0x00FFFFFF; NVIC_ST_CURRENT_R = 0;
NVIC_ST_CTRL_R = 0x00000005;

}
uint32_t Delaytime = 116548;
// Delay for the count units (ms)
void Delay(uint8_t count)
{
    uint8_t i;
    for(i = count; i > 0; i--)
    {
        Delaytime = 116548;
        while (Delaytime > 0)
        {
            Delaytime--;
        }
} }

// -UUU- Declare your debug dump arrays and indexes
into the // arrays here uint8_t
DataBuffer[50]; uint32_t TimeBuffer[50];

void DebugInit(void)
{ uint32_t p;

for (p = 0; p < 50; p++)

{ TimeBuffer[p] = 0xFFFFFFFF;

        DataBuffer[p] = 0xFF;
    }
}
```

```
void ToggleLEDOn(void)
{
    GPIO_PORTF_DATA_R |= 0x4;
    GPIO_PORTE_DATA_R |= 0x1;
}
void ToggleLEDOff(void)
{ GPIO_PORTF_DATA_R &= ~(0x4);

GPIO_PORTE_DATA_R &= ~(0x1); }

uint32_t intree = 0;
void Debug_Capture(void)
{
    if(intree == 50)
{ return;

} else {

DataBuffer[intree] = ((GPIO_PORTE_DATA_R & 0x1)) |
((GPIO_PORTE_DATA_R & 0x2)<<3);

intree++; }

}

uint8_t onTrack = 1;
uint8_t Pushtrack = 0;
uint8_t offTrack = 4;
uint8_t onCapture = 0;
uint8_t offCapture = 0;
uint8_t onAllow = 1;
int main(void)
{ TExaS_Init(); Ports_Init(); // initialize ports
SysTick_Init(); // initialize SysTick
EnableInterrupts(); //Enable interrupts

    DebugInit();
while(1)

{ if ((GPIO_PORTE_DATA_R & 0x2)==2)
```

```
        {
            Pushtrack = 1;
            if (onAllow == 1)
            {
            onCapture = 1;
            onAllow--;
} }

        else if(Pushtrack == 1)
{ onAllow++;

                onTrack++;
                offTrack--;
                offCapture = 7;
                Pushtrack = 0;
                    if (onTrack == 6)
TimeBuffer[intree] = NVIC_ST_CURRENT_R;

}

{ onTrack = 0;

    offTrack = 5;
}
        if(onTrack > 0)
        {
            ToggleLEDOn();
        }
        if(onCapture == 1)
        {
            onCapture = 0;
            Debug_Capture();
        }
        if (offCapture > 0)
        {
            Debug_Capture();
            offCapture--;
        }
        Delay(onTrack);
        if (offTrack > 0)
        {
            ToggleLEDOff();
        }
        if (offCapture > 0)
        {
            Debug_Capture();
            offCapture--;
}

        Delay(offTrack);
    }
}
```