Unit 3 Embedded C
Lecture 4
Lab 3 Report

Creating a BareMetal SW on

TM4C123 ARM CortexM4

By: Yara Ashraf

Step 1] main.c

- Below code to toggle LED connected to GPIOF Pin3

```c
/*
 Learn-in-depth
 Eng. Yara Ashraf
 Baremetal SW on TM4C123 ARM CortexM4
*/

#define SYSCTL_RCGC2_R  (*((volatile unsigned long *)0x400FE108))
#define GPIO_PORTF_DATA_R   (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R    (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_DEN_R    (*((volatile unsigned long *)0x4002551C))

int main (void)
{
    SYSCTL_RCGC2_R = 0x00000020;
    //delay to make sure GPIOF is up and runnning

    volatile unsigned long delay_count=0; //volatile to prevent optimization
    for(delay_count; delay_count < 200 ; delay_count++);
    GPIO_PORTF_DIR_R |= 1<<3; //to set direction of pin3 as output
    GPIO_PORTF_DEN_R |= 1<<3; //to enable pin3

    while(1)
    {
        GPIO_PORTF_DATA_R |= 1<<3;
        for(delay_count; delay_count < 200000 ; delay_count++);
        GPIO_PORTF_DATA_R &= ~(1<<3);
        for(delay_count; delay_count < 200000 ; delay_count++);

    }

    return 0;
}
```

Step 2] Editing startup.c to define the Stack_top by making use of the .bss section

- Method 1] Using a global uninitialized array
- Method 2] Using a global array of constant pointers to function

```c
#include <stdint.h>

void Reset_Handler ();
extern int main (void);

void Default_Handler ()
{
    Reset_Handler ();
}

void NMI_Handler () __attribute__ ((weak, alias ("Default_Handler")));
void H_fault_Handler () __attribute__ ((weak, alias ("Default_Handler")));


//reserving 1024 bytes located by .bss using unintialized global array of 256 elements (256*4=1024)
//Static to limit scope to this file only
static unsigned long stack_top[256];

//Method 1] using global unint array
/*uint32_t vectors[] __attribute__((section(".vectors"))) = {

    (uint32_t) &stack_top[0]+ sizeof(stack_top),
    (uint32_t) &Reset_Handler,
    (uint32_t) &NMI_Handler,
    (uint32_t) &H_fault_Handler,

};*/

//Method 2] using global array of constant pointers to function takes anything and returns void
//Per acrchitecture pointer size is 32bit so equivalent to method 1 in size

void (* const g_p_fn_Vectors[]) () __attribute__((section(".vectors"))) = {

    (void (*)()) ((unsigned long)stack_top+ sizeof(stack_top)),
    //below  are not casted as they are already def as func takes anything and returns void

    &Reset_Handler,
    &NMI_Handler,
    &H_fault_Handler,

};

extern unsigned int _E_text ;
extern unsigned int _S_Data ;
extern unsigned int _E_Data ;
extern unsigned int _S_bss ;
extern unsigned int _E_bss ;

void Reset_Handler ()
{
    //copy .data section from flash to sram
    unsigned int DATA_size = (unsigned char*)&_E_Data - (unsigned char*)&_S_Data ;
    unsigned char* P_src = (unsigned char*)&_E_text ;
    unsigned char* P_dst = (unsigned char*)&_S_Data ;
    int i;
    for (i=0; i< DATA_size ; i++)
    {
        *((unsigned char*)P_dst++) = *((unsigned char*)P_src++) ;
    }
    //init .bss section in SRAM = 0
    unsigned int bss_size = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss ;
    P_dst = (unsigned char*)&_S_bss;
    for (i=0; i< bss_size ; i++)
    {
        *((unsigned char*)P_dst++) = (unsigned char)0 ;
    }
    //jump to main()

    main();
}
```

Step 3] Linker_script

```
/* linker script for Cortex M4
Eng. Yara
*/

MEMORY
{
    flash(rx) : ORIGIN = 0x00000000, LENGTH = 512M
    sram(rwx) : ORIGIN = 0x20000000, LENGTH = 512M
}

SECTIONS
{
    .text : {
            *(.vectors*)
            *(.text*)
            *(.rodata)
            _E_text = . ;
    }> flash
    .data : {
            _S_Data = . ;
            *(.data)
            . = ALIGN(4) ;
            _E_Data = . ;
    }> sram AT> flash

    .bss : {
            _S_bss = . ;
            *(.bss*)
            _E_bss = . ;
    }> sram
}
```

Step 3] Makefile

- Copying. elf content to .axf file so it can be passed to Keil uVision
- Passing debug option to gcc to be able to debug

```
#@copyright : Yara
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m4 -gdwarf-2 -g
INCS=-I .
LIBS=
SRC= $(wildcard *.c)
OBJ= $(SRC:.c=.o)
As= $(wildcard *.s)
AsOBJ= $(As:.s=.o)
Project_name=Unit3_Lab4_CortexM4

all: $(Project_name).bin

%.o: %.c
    $(CC)gcc.exe -c $(CFLAGS) $(INCS) $< -o $@

$(Project_name).elf: $(OBJ) $(AsOBJ)
    $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@ -Map=Map_file.map
    cp $(Project_name).elf $(Project_name).axf

$(Project_name).bin: $(Project_name).elf
    $(CC)objcopy.exe -O binary $< $@

clean all:
    rm *.o *.elf *.axf *.bin *.map

clean:
    rm *.elf *.axf *.bin
```

Step 3] Analyzing object files Section Headers and Symbols

1] main.o

- no global uninitialized data so .bss size = 0

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedd
$ arm-none-eabi-objdump.exe -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000008c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000c0  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000c0  2**0
                  ALLOC
  3 .debug_info   00000066  00000000  00000000  000000c0  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 0000005c  00000000  00000000  00000126  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000038  00000000  00000000  00000182  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  000001ba  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000064  00000000  00000000  000001da  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000000f5  00000000  00000000  0000023e  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007c  00000000  00000000  00000333  2**0
                  CONTENTS, READONLY
 10 .debug_frame  0000002c  00000000  00000000  000003b0  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  000003dc  2**0
                  CONTENTS, READONLY

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedd
$ arm-none-eabi-nm.exe main.o
00000000 T main

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedd
```

2] startup.o

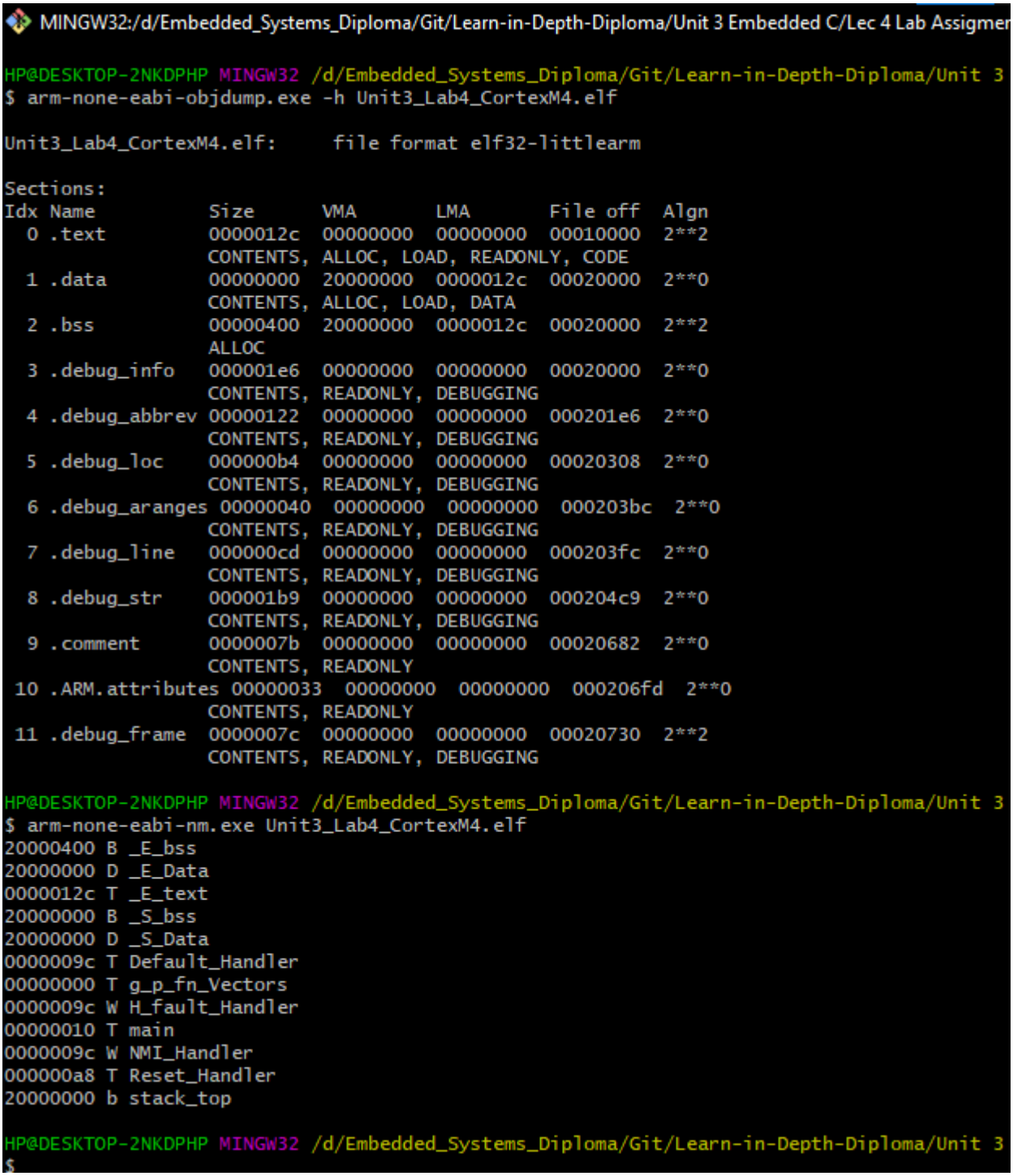
- as seen below symbols are unresolved yet

```
MINGW32:/d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedded C/Lec 4 La
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000090  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000c4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000400  00000000  00000000  000000c4  2**2
                  ALLOC
  3 .vectors      00000010  00000000  00000000  000000c4  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, DATA
  4 .debug_info   00000180  00000000  00000000  000000d4  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000c6  00000000  00000000  00000254  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    0000007c  00000000  00000000  0000031a  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  00000396  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   00000069  00000000  00000000  000003b6  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    000001d5  00000000  00000000  0000041f  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      0000007c  00000000  00000000  000005f4  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000050  00000000  00000000  00000670  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 12 .ARM.attributes 00000033  00000000  00000000  000006c0  2**0
                  CONTENTS, READONLY

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma
$ arm-none-eabi-nm.exe startup.o
         U _E_bss
         U _E_Data
         U _E_text
         U _S_bss
         U _S_Data
00000000 T Default_Handler
00000000 R g_p_fn_Vectors
00000000 W H_fault_Handler
         U main
00000000 W NMI_Handler
0000000c T Reset_Handler
00000000 b stack_top

HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma
```

Step 4] .elf file

- sections of main.o and startup.o are merged and output to the below sections

- all symbols are resolved



- Readelf shows entry point is at 0x0

```
HP@DESKTOP-2NKDPHP MINGW32 /d/Embedded_Systems_Diploma/Git/Learn-in-Depth-Diploma/Unit 3 Embedded C/Lec 4
$ arm-none-eabi-readelf.exe -a Unit3_Lab4_CortexM4.elf
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          52 (bytes into file)
  Start of section headers:          133872 (bytes into file)
  Flags:                             0x5000200, Version5 EABI, soft-float ABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         2
  Size of section headers:           40 (bytes)
  Number of section headers:         16
  Section header string table index: 15

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        00000000 010000 00012c 00  AX  0   0  4
  [ 2] .data             PROGBITS        20000000 020000 000000 00  WA  0   0  1
  [ 3] .bss              NOBITS          20000000 020000 000400 00  WA  0   0  4
  [ 4] .debug_info       PROGBITS        00000000 020000 0001e6 00      0   0  1
  [ 5] .debug_abbrev     PROGBITS        00000000 0201e6 000122 00      0   0  1
  [ 6] .debug_loc        PROGBITS        00000000 020308 0000b4 00      0   0  1
  [ 7] .debug_aranges    PROGBITS        00000000 0203bc 000040 00      0   0  1
  [ 8] .debug_line       PROGBITS        00000000 0203fc 0000cd 00      0   0  1
  [ 9] .debug_str        PROGBITS        00000000 0204c9 0001b9 01  MS  0   0  1
  [10] .comment          PROGBITS        00000000 020682 00007b 01  MS  0   0  1
  [11] .ARM.attributes   ARM_ATTRIBUTES  00000000 0206fd 000033 00      0   0  1
  [12] .debug_frame      PROGBITS        00000000 020730 00007c 00      0   0  4
  [13] .symtab           SYMTAB          00000000 0207ac 000210 10     14  22  4
  [14] .strtab           STRTAB          00000000 0209bc 000096 00      0   0  1
  [15] .shstrtab         STRTAB          00000000 020a52 00009d 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  y (purecode), p (processor specific)
```

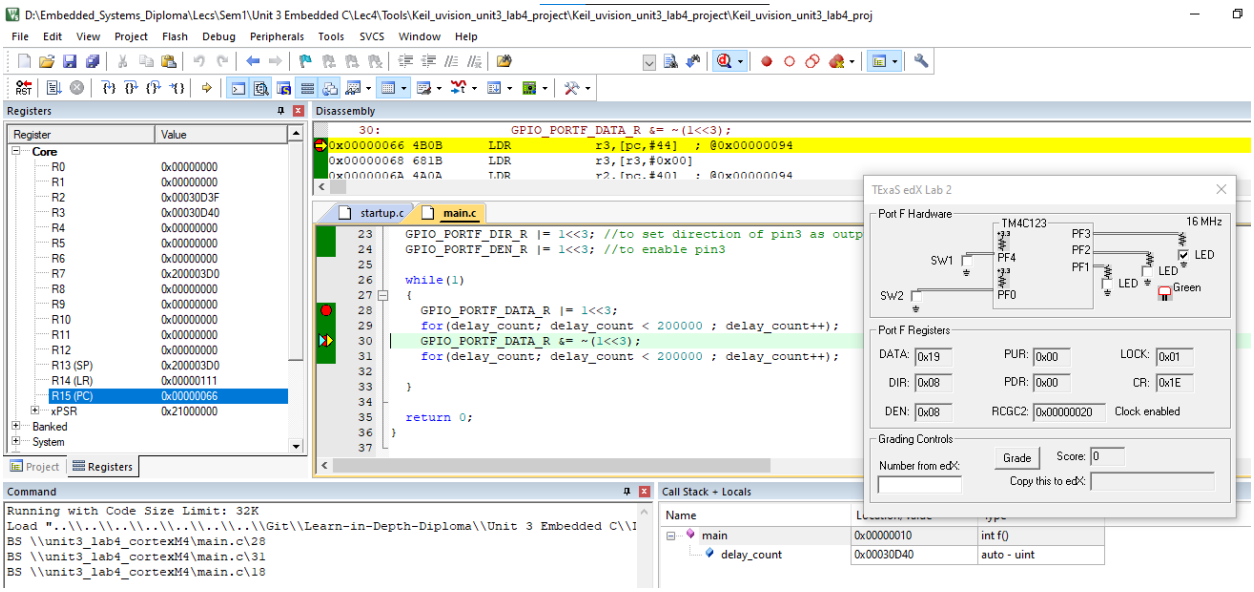
Step 5] Mapfile

- .text in Flash includes .vectors/ .text/ .ro data sections (starts at 0x00000000)
- .data in FLASH is copied to SRAM via startup (starting at 0x20000000)
- .bss is reserved in SRAM via linker_script and initialized to 0 via startup

```
Memory Configuration

Name             Origin             Length             Attributes
flash            0x0000000000000000 0x0000000020000000 xr
sram             0x0000000020000000 0x0000000020000000 xrw
*default*        0x0000000000000000 0xffffffffffffffff

Linker script and memory map


.text           0x0000000000000000      0x12c
 *(.vectors*)
 .vectors       0x0000000000000000       0x10 startup.o
                0x0000000000000000                g_p_fn_Vectors
 *(.text*)
 .text          0x0000000000000010       0x8c main.o
                0x0000000000000010                main
 .text          0x000000000000009c       0x90 startup.o
                0x000000000000009c                H_fault_Handler
                0x000000000000009c                Default_Handler
                0x000000000000009c                NMI_Handler
                0x00000000000000a8                Reset_Handler
 *(.rodata)
                0x000000000000012c                _E_text = .

.glue_7         0x000000000000012c        0x0
 .glue_7        0x000000000000012c        0x0 linker stubs

.glue_7t        0x000000000000012c        0x0
 .glue_7t       0x000000000000012c        0x0 linker stubs

.vfp11_veneer   0x000000000000012c        0x0
 .vfp11_veneer  0x000000000000012c        0x0 linker stubs

.v4_bx          0x000000000000012c        0x0
 .v4_bx         0x000000000000012c        0x0 linker stubs

.iplt           0x000000000000012c        0x0
 .iplt          0x000000000000012c        0x0 main.o

.rel.dyn        0x000000000000012c        0x0
 .rel.iplt      0x000000000000012c        0x0 main.o

.data           0x0000000020000000        0x0 load address 0x000000000000012c
                0x0000000020000000                _S_Data = .
 *(.data)
 .data          0x0000000020000000        0x0 main.o
 .data          0x0000000020000000        0x0 startup.o
                0x0000000020000000                . = ALIGN (0x4)
                0x0000000020000000                _E_Data = .

.igot.plt       0x0000000020000000        0x0 load address 0x000000000000012c
 .igot.plt      0x0000000020000000        0x0 main.o

.bss            0x0000000020000000      0x400 load address 0x000000000000012c
                0x0000000020000000                _S_bss = .
 *(.bss*)
 .bss           0x0000000020000000        0x0 main.o
 .bss           0x0000000020000000      0x400 startup.o
                0x0000000020000400                _E_bss = .
LOAD main.o
LOAD startup.o
OUTPUT(Unit3_Lab4_CortexM4.elf elf32-littlearm)

.debug_info     0x0000000000000000      0x1e6
 .debug_info    0x0000000000000000       0x66 main.o
 .debug_info    0x0000000000000066      0x180 startup.o

```

Step 6] Simulation and debug via Keil uVision

- Board simulation using TExas edX Lab



- Using logic analyzer for PORTF pin3