GENESIS Learning Report - Embedded C: Hardware + Programming + Testing (99003157)

LTTS
GLOBAL
ENGINEERING
ACADEMY

L&T Technology Services

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
1.0	24-12-20	DIMANTH G S		Dr. Vivek K and Bhargav N	
1.1	28-12-20	DIMANTH G S		Dr. Vivek K and Bhargav N	

Contents

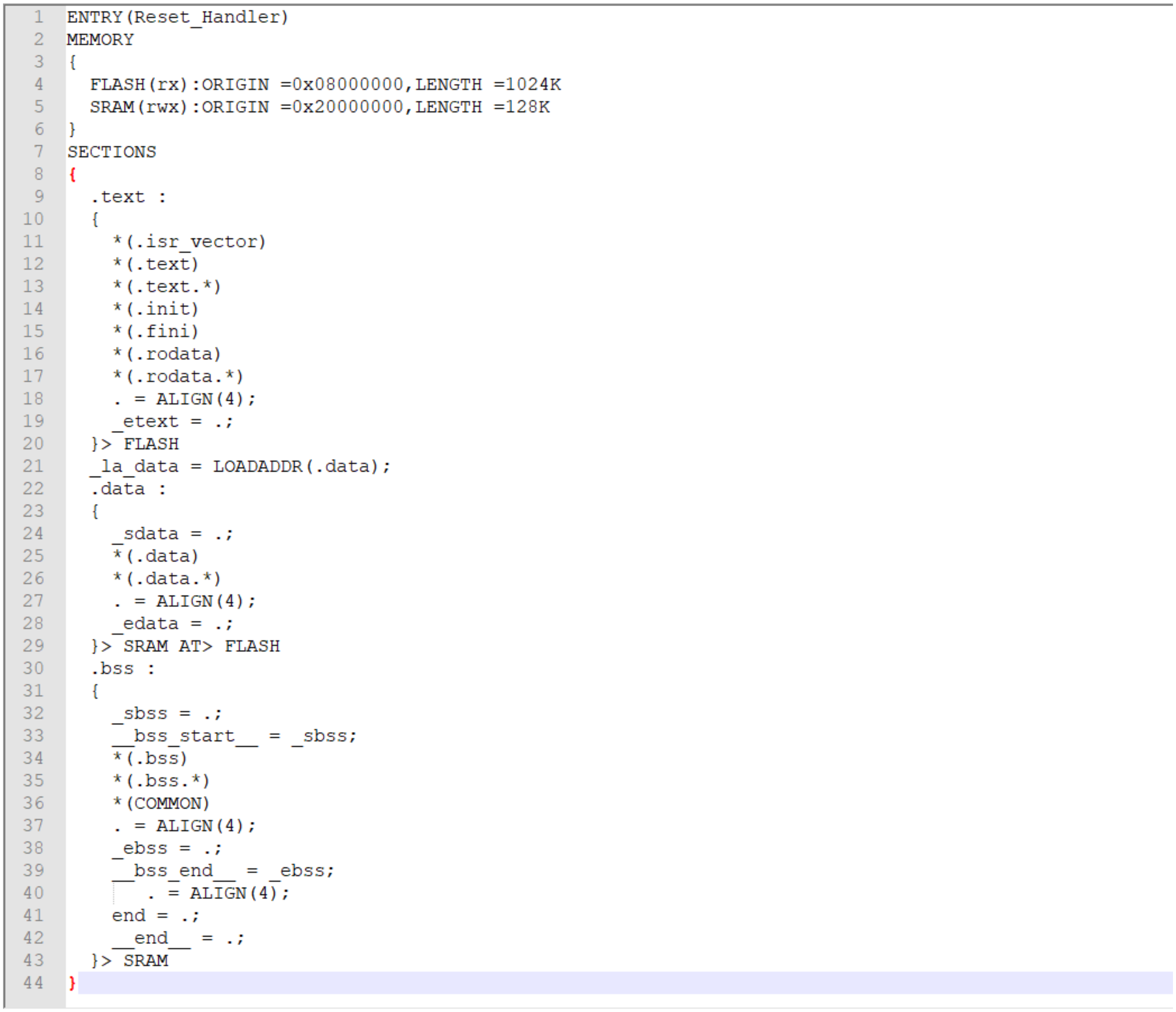
Table of Figures

List of Tables

No table of figures entries found.

1. Activity1

1.1 Linker Script



```
ENTRY(Reset_Handler)
MEMORY
{
  FLASH(rx):ORIGIN =0x08000000,LENGTH =1024K
  SRAM(rwx):ORIGIN =0x20000000,LENGTH =128K
}
SECTIONS
{
  .text :
  {
    *(.isr_vector)
    *(.text)
    *(.text.*)
    *(.init)
    *(.fini)
    *(.rodata)
    *(.rodata.*)
    . = ALIGN(4);
    _etext = .;
  }> FLASH
  _la_data = LOADADDR(.data);
  .data :
  {
    _sdata = .;
    *(.data)
    *(.data.*)
    . = ALIGN(4);
    _edata = .;
  }> SRAM AT> FLASH
  .bss :
  {
    _sbss = .;
    __bss_start__ = _sbss;
    *(.bss)
    *(.bss.*)
    *(COMMON)
    . = ALIGN(4);
    _ebss = .;
    __bss_end__ = _ebss;
       . = ALIGN(4);
    end = .;
    __end__ = .;
  }> SRAM
}
```

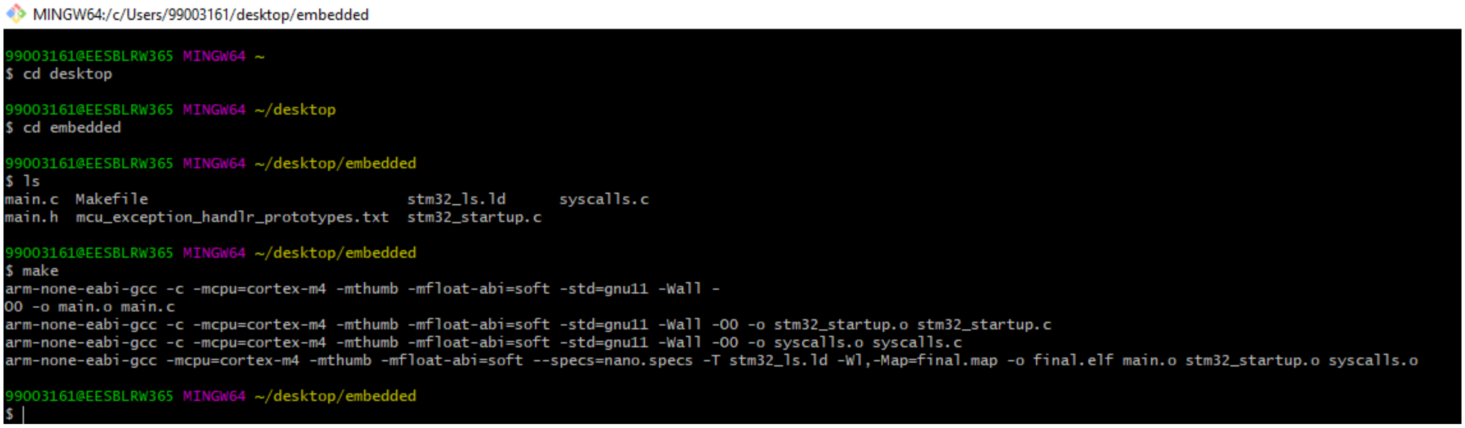
Fig 1.Linker Script

1.2 Make file

1.2.1 Main code

```
#include<stdio.h>
int main()
{
 int i;
 int fact=1,num;
 printf("Enter a number: ");

  scanf("%d",&num);
    for(i=1;i<=num;i++)
            {
      fact=fact*i;
    }
  printf("Factorial of %d is: %d",num,fact);
return 0;
}
```

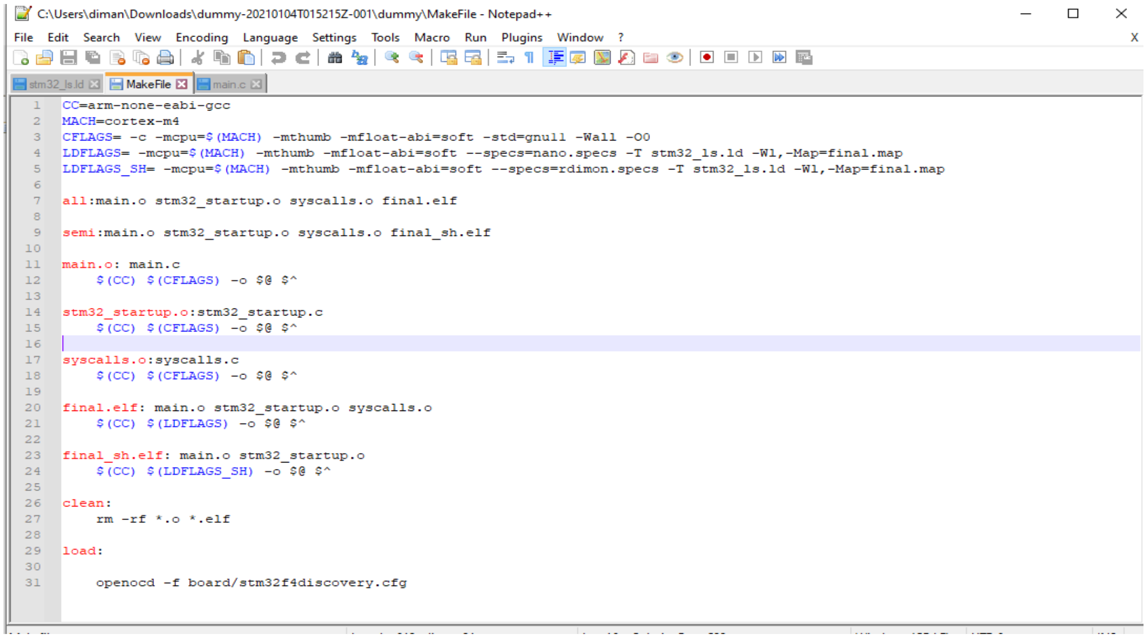
1.2.2 Make file execution



```
MINGW64:/c/Users/99003161/desktop/embedded
99003161@EESBLRW365 MINGW64 ~
$ cd desktop

99003161@EESBLRW365 MINGW64 ~/desktop
$ cd embedded

99003161@EESBLRW365 MINGW64 ~/desktop/embedded
$ ls
main.c  Makefile                          stm32_ls.ld      syscalls.c
main.h  mcu_exception_handlr_prototypes.txt  stm32_startup.c

99003161@EESBLRW365 MINGW64 ~/desktop/embedded
$ make
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -mfloat-abi=soft -std=gnu11 -Wall -
O0 -o main.o main.c
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -mfloat-abi=soft -std=gnu11 -Wall -O0 -o stm32_startup.o stm32_startup.c
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -mfloat-abi=soft -std=gnu11 -Wall -O0 -o syscalls.o syscalls.c
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=soft --specs=nano.specs -T stm32_ls.ld -Wl,-Map=final.map -o final.elf main.o stm32_startup.o syscalls.o

99003161@EESBLRW365 MINGW64 ~/desktop/embedded
$ |
```

Figure 2. Make file execution

1.2.3 Make file code

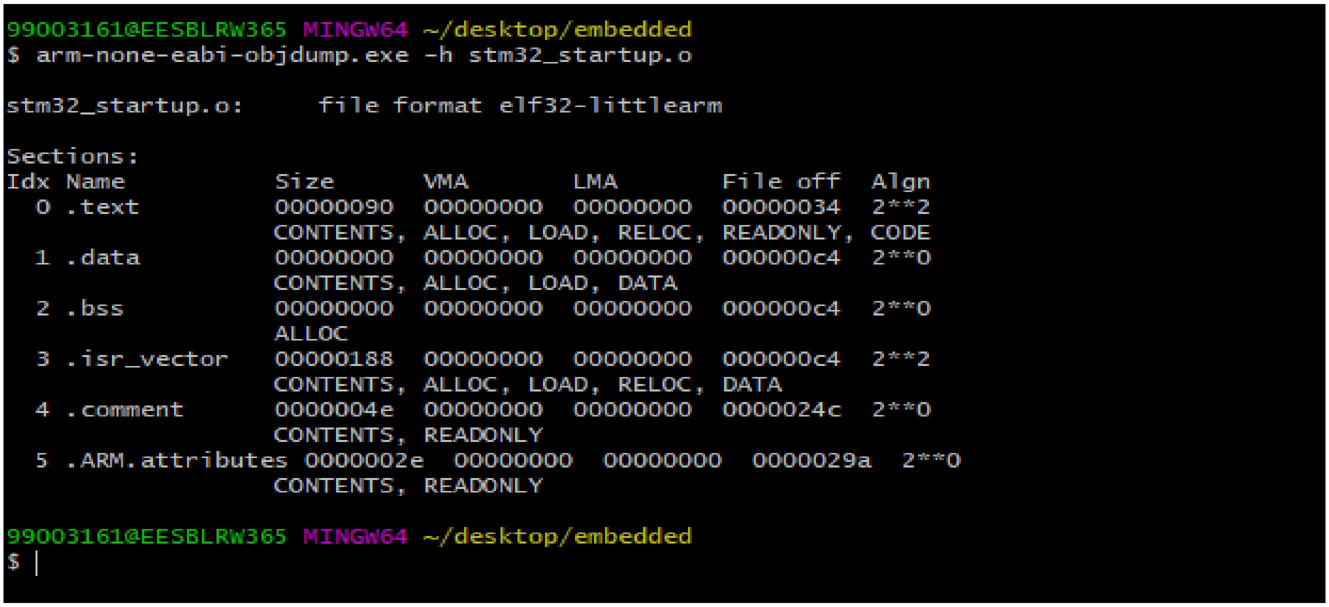


C:\Users\diman\Downloads\dummy-20210104T015215Z-001\dummy\MakeFile - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
stm32_ls.ld
MakeFile
main.c

```
CC=arm-none-eabi-gcc
MACH=cortex-m4
CFLAGS= -c -mcpu=$(MACH) -mthumb -mfloat-abi=soft -std=gnu11 -Wall -O0
LDFLAGS= -mcpu=$(MACH) -mthumb -mfloat-abi=soft --specs=nano.specs -T stm32_ls.ld -Wl,-Map=final.map
LDFLAGS_SH= -mcpu=$(MACH) -mthumb -mfloat-abi=soft --specs=rdimon.specs -T stm32_ls.ld -Wl,-Map=final.map

all:main.o stm32_startup.o syscalls.o final.elf

semi:main.o stm32_startup.o syscalls.o final_sh.elf

main.o: main.c
	$(CC) $(CFLAGS) -o $@ $^

stm32_startup.o:stm32_startup.c
	$(CC) $(CFLAGS) -o $@ $^

syscalls.o:syscalls.c
	$(CC) $(CFLAGS) -o $@ $^

final.elf: main.o stm32_startup.o syscalls.o
	$(CC) $(LDFLAGS) -o $@ $^

final_sh.elf: main.o stm32_startup.o
	$(CC) $(LDFLAGS_SH) -o $@ $^

clean:
	rm -rf *.o *.elf

load:

	openocd -f board/stm32f4discovery.cfg
```

Fig 3. Make file code

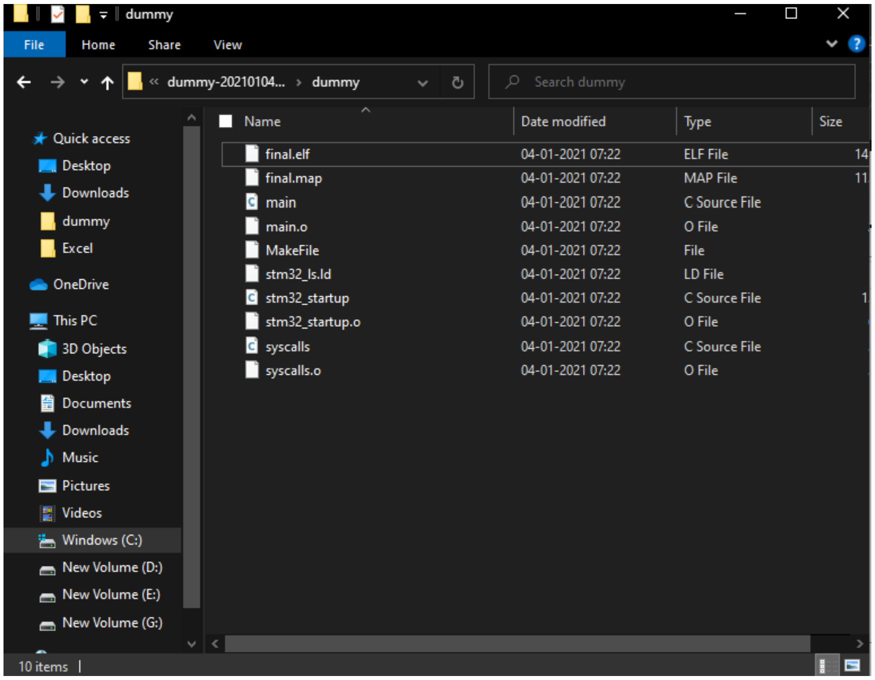
1.3 Startup



```
99003161@EESBLRW365 MINGW64 ~/desktop/embedded
$ arm-none-eabi-objdump.exe -h stm32_startup.o

stm32_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          00000000  00000000  00000000  000000c4  2**0
                  ALLOC
  3 .isr_vector   00000188  00000000  00000000  000000c4  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  4 .comment      0000004e  00000000  00000000  0000024c  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 0000002e  00000000  00000000  0000029a  2**0
                  CONTENTS, READONLY

99003161@EESBLRW365 MINGW64 ~/desktop/embedded
$ |
```

Figure 4. Startup

1.4 Output Files



Name	Date modified	Type	Size
final.elf	04-01-2021 07:22	ELF File	14
final.map	04-01-2021 07:22	MAP File	11
main	04-01-2021 07:22	C Source File	
main.o	04-01-2021 07:22	O File	
MakeFile	04-01-2021 07:22	File	
stm32_ls.ld	04-01-2021 07:22	LD File	
stm32_startup	04-01-2021 07:22	C Source File	1
stm32_startup.o	04-01-2021 07:22	O File	
syscalls	04-01-2021 07:22	C Source File	
syscalls.o	04-01-2021 07:22	O File	

Figure 5. Output files

1.5 GitHub Link of code files

1.6 Debugging Techniques

1.6.1 Step in, step over and step out

Step over – An action to take in the debugger that will step over a given line. If the line contains a function, the function will be executed and the result returned without debugging each line. If we have a break point in the program and if we press step over button, then the line where the program was paused, that line will be executed. Then the program would pause at the next line.

Step into – An action to take in the debugger. If the current program line contains a function or a method, we can shift the debugging control into the function by pressing ‘step in’ button.

Step out - Once ‘step in’ action is performed, step return will be enabled. The debugging control will return from the method when step return is pressed. When execution is paused inside a function, you can click the Step Out button on the Debug toolbar or select Debug: Step Out to step out of the function. The debugger executes the rest of the function without pausing, and then returns to the line after the function call and pauses.

1.6.2 Disassembly

The Debug Disassembly Window gives the user access to debugging in assembly language for project written in C. The Debug Disassembly Window allows the user to perform all the normal debug operations including single stepping and setting breakpoints on the individual assembly instructions generated from C code.

1.6.3 Break points

Setting breakpoints while debugging code for an embedded system is a common and familiar task. Each Cortex M-series device supports some number of hardware breakpoints. These are comparators in the CPU core that pauses the core when a designated match condition occurs (e.g. the program counter matches the value that corresponds to the address of a specific instruction).

2. Activity 2 (Driver code development)

2.1 MCU Specific Header File

```
/*
 * STM32F4XX.h
 *
 *  Created on: Dec 28, 2020
 *      Author: 99003157
 */

#ifndef INC_STM32F4XX_H_
#define INC_STM32F4XX_H_

#include<stdint.h>

/*defining base addresses*/
#define __vo volatile

#define ENABLE                    1
#define DISABLE            0
#define GPIO_PIN_SET   ENABLE
#define GPIO_PIN_RESET DISABLE
//macros for memories
#define FLASHMEM    0x80000000U
#define SRAM2             0x2001C000U
#define SRAM1             0x20000000U
#define ROM         0x1FFF0000U
```

```
#define SRAM        SRAM1


//macros for buses

#define AHB1        0x40020000U

#define AHB2        0x50000000U

#define APB1        0x40000000U

#define APB2        0x40010000U

#define PERIPHERAL  APB1


//macros for peripherals in AHB1 bus

#define GPIOA_BASE        0x40020000U

#define GPIOB_BASE        (GPIOA + (0x0400U))

#define GPIOC_BASE        (GPIOB + (0x0400U))

#define GPIOD_BASE        (GPIOC + (0x0400U))

#define GPIOE_BASE        (GPIOD + (0x0400U))

#define GPIOF_BASE        (GPIOE + (0x0400U))

#define GPIOG_BASE        (GPIOF + (0x0400U))

#define GPIOH_BASE        (GPIOG + (0x0400U))

#define GPIOI_BASE        (GPIOH + (0x0400U))

#define RCC_BASE          (AHB1+(0x3800U))


#define RCC                     ((RCC_Reg_def_t*)RCC_BASE)

//macros for peripherals in APB bus

#define I2C3        0x40005C00U

#define I2C2        0x40005800U

#define I2C1        0x40005400U


#define UART5       0x40055000U
```

```
#define UART4       0x40004C00U


#define USART6      0x40011400U
#define USART3            0x40004800U
#define USART2            0x40004400U
#define USART1      0x40011000U


#define SPI3_I2S3   0x40003C00U
#define SPI2_I2S2   0x40003800U
#define SPI1        0x40013000U


//GPIO peripheral registers
typedef struct
{
        __vo uint32_t MODER;
        __vo uint32_t OTYPER;
        __vo uint32_t OSPEEDR;
        __vo uint32_t PUPDR;
        __vo uint32_t IDR;
        __vo uint32_t ODR;
        __vo uint32_t BSRR;
        __vo uint32_t LCKR;
        __vo uint32_t AFR [2]; //AFR[1]-(AFRL)Low register, AFR[0]-(AFRH)High Registers

}GPIO_Regdef_t;


typedef struct
{
        __vo uint32_t CR;
        __vo uint32_t PLLCFGR;
        __vo uint32_t CFGR;
        __vo uint32_t CIR;
        __vo uint32_t AHB1RSTR;
        __vo uint32_t AHB2RSTR;
        __vo uint32_t AHB3RSTR;
        uint32_t RESERVED0;
        __vo uint32_t APB1RSTR;
        __vo uint32_t APB2RSTR;
        uint32_t RESERVED1[2];
        __vo uint32_t AHB1ENR;
        __vo uint32_t AHB2ENR;
        __vo uint32_t AHB3ENR;
        uint32_t RESERVED2;
        __vo uint32_t APB1ENR;
        __vo uint32_t APB2ENR;
        uint32_t RESERVED3[2];
        __vo uint32_t AHB1LPENR;
        __vo uint32_t AHB2LPENR;
        __vo uint32_t AHB3LPENR;
        uint32_t RESERVED4;
        __vo uint32_t APB1LPENR;
        __vo uint32_t APB2LPENR;
        uint32_t RESERVED5[2];
        __vo uint32_t BDCR;
        __vo uint32_t CSR;
        uint32_t RESERVED6[2];
```

```
	__vo uint32_t SSCGR;

	__vo uint32_t PLLI2SCFGR;

	__vo uint32_t PLLSAICFGR;

	__vo uint32_t DCKCFGR;

}RCC_Reg_def_t;


#define GPIOA (GPIO_Regdef_t*)GPIOA_BASE

#define GPIOB (GPIO_Regdef_t*)GPIOB_BASE

#define GPIOC (GPIO_Regdef_t*)GPIOC_BASE

#define GPIOD (GPIO_Regdef_t*)GPIOD_BASE

#define GPIOE (GPIO_Regdef_t*)GPIOE_BASE

#define GPIOF (GPIO_Regdef_t*)GPIOF_BASE

#define GPIOG (GPIO_Regdef_t*)GPIOG_BASE

#define GPIOH (GPIO_Regdef_t*)GPIOH_BASE

#define GPIOI (GPIO_Regdef_t*)GPIOI_BASE


//clock enable macros

#define GPIOA_PCLK_EN() (RCC -> AHB1ENR |= 1<<0) ;

#define GPIOB_PCLK_EN() (RCC -> AHB1ENR |= 1<<1) ;

#define GPIOC_PCLK_EN() (RCC -> AHB1ENR |= 1<<2) ;

#define GPIOD_PCLK_EN() (RCC -> AHB1ENR |= 1<<3) ;

#define GPIOE_PCLK_EN() (RCC -> AHB1ENR |= 1<<4) ;

#define GPIOF_PCLK_EN() (RCC -> AHB1ENR |= 1<<5) ;

#define GPIOG_PCLK_EN() (RCC -> AHB1ENR |= 1<<6) ;

#define GPIOH_PCLK_EN() (RCC -> AHB1ENR |= 1<<7) ;

#define GPIOI_PCLK_EN() (RCC -> AHB1ENR |= 1<<8) ;


//clock disable macros
```

```
#define GPIOA_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0));
}while(0) ;

#define GPIOB_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 1)); (RCC->AHB1RSTR &= ~(1 << 1));
}while(0) ;

#define GPIOC_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 2)); (RCC->AHB1RSTR &= ~(1 << 2));
}while(0) ;

#define GPIOD_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 3)); (RCC->AHB1RSTR &= ~(1 << 3));
}while(0) ;

#define GPIOE_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 4)); (RCC->AHB1RSTR &= ~(1 << 4));
}while(0) ;

#define GPIOF_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 5)); (RCC->AHB1RSTR &= ~(1 << 5));
}while(0) ;

#define GPIOG_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 6)); (RCC->AHB1RSTR &= ~(1 << 6));
}while(0) ;

#define GPIOH_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 7)); (RCC->AHB1RSTR &= ~(1 << 7));
}while(0) ;

#define GPIOI_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 8)); (RCC->AHB1RSTR &= ~(1 << 8));
}while(0) ;


#endif /* INC_STM32F4XX_H_ */
```

2.2 GPIO Driver File

```
/*

 * STM32F4XX.h

 *

 *  Created on: Dec 28, 2020

 *      Author: 99003157

 */


#ifndef INC_STM32F4XX_H_

#define INC_STM32F4XX_H_
```

```
#include<stdint.h>


/*defining base addresses*/

#define __vo volatile


#define ENABLE                   1

#define DISABLE             0

#define GPIO_PIN_SET   ENABLE

#define GPIO_PIN_RESET DISABLE

//macros for memories

#define FLASHMEM    0x80000000U

#define SRAM2             0x2001C000U

#define SRAM1             0x20000000U

#define ROM         0x1FFF0000U

#define SRAM        SRAM1


//macros for buses

#define AHB1        0x40020000U

#define AHB2        0x50000000U

#define APB1        0x40000000U

#define APB2        0x40010000U

#define PERIPHERAL  APB1


//macros for peripherals in AHB1 bus

#define GPIOA_BASE        0x40020000U

#define GPIOB_BASE        (GPIOA + (0x0400U))

#define GPIOC_BASE        (GPIOB + (0x0400U))

#define GPIOD_BASE        (GPIOC + (0x0400U))
```

```
#define GPIOE_BASE        (GPIOD + (0x0400U))
#define GPIOF_BASE        (GPIOE + (0x0400U))
#define GPIOG_BASE        (GPIOF + (0x0400U))
#define GPIOH_BASE        (GPIOG + (0x0400U))
#define GPIOI_BASE        (GPIOH + (0x0400U))
#define RCC_BASE          (AHB1+(0x3800U))

#define RCC                      ((RCC_Reg_def_t*)RCC_BASE)
//macros for peripherals in APB bus
#define I2C3        0x40005C00U
#define I2C2        0x40005800U
#define I2C1        0x40005400U

#define UART5       0x40055000U
#define UART4       0x40004C00U

#define USART6      0x40011400U
#define USART3            0x40004800U
#define USART2            0x40004400U
#define USART1      0x40011000U

#define SPI3_I2S3   0x40003C00U
#define SPI2_I2S2   0x40003800U
#define SPI1        0x40013000U

//GPIO peripheral registers
typedef struct
{
        __vo uint32_t MODER;
        __vo uint32_t OTYPER;
        __vo uint32_t OSPEEDR;
        __vo uint32_t PUPDR;
        __vo uint32_t IDR;
        __vo uint32_t ODR;
        __vo uint32_t BSRR;
        __vo uint32_t LCKR;
        __vo uint32_t AFR [2]; //AFR[1]-(AFRL)Low register, AFR[0]-(AFRH)High Registers

}GPIO_Regdef_t;

typedef struct
{
        __vo uint32_t CR;
        __vo uint32_t PLLCFGR;
        __vo uint32_t CFGR;
        __vo uint32_t CIR;
        __vo uint32_t AHB1RSTR;
        __vo uint32_t AHB2RSTR;
        __vo uint32_t AHB3RSTR;
        uint32_t RESERVED0;
        __vo uint32_t APB1RSTR;
        __vo uint32_t APB2RSTR;
        uint32_t RESERVED1[2];
        __vo uint32_t AHB1ENR;
        __vo uint32_t AHB2ENR;
        __vo uint32_t AHB3ENR;
        uint32_t RESERVED2;
        __vo uint32_t APB1ENR;
        __vo uint32_t APB2ENR;
        uint32_t RESERVED3[2];
        __vo uint32_t AHB1LPENR;
        __vo uint32_t AHB2LPENR;
        __vo uint32_t AHB3LPENR;
        uint32_t RESERVED4;
        __vo uint32_t APB1LPENR;
        __vo uint32_t APB2LPENR;
        uint32_t RESERVED5[2];
        __vo uint32_t BDCR;
        __vo uint32_t CSR;
        uint32_t RESERVED6[2];
        __vo uint32_t SSCGR;
        __vo uint32_t PLLI2SCFGR;
        __vo uint32_t PLLSAICFGR;
        __vo uint32_t DCKCFGR;
}RCC_Reg_def_t;


#define GPIOA (GPIO_Regdef_t*)GPIOA_BASE
#define GPIOB (GPIO_Regdef_t*)GPIOB_BASE
#define GPIOC (GPIO_Regdef_t*)GPIOC_BASE
#define GPIOD (GPIO_Regdef_t*)GPIOD_BASE
#define GPIOE (GPIO_Regdef_t*)GPIOE_BASE
#define GPIOF (GPIO_Regdef_t*)GPIOF_BASE
#define GPIOG (GPIO_Regdef_t*)GPIOG_BASE
#define GPIOH (GPIO_Regdef_t*)GPIOH_BASE
```

```
#define GPIOI (GPIO_Regdef_t*)GPIOI_BASE


//clock enable macros
#define GPIOA_PCLK_EN() (RCC -> AHB1ENR |= 1<<0) ;
#define GPIOB_PCLK_EN() (RCC -> AHB1ENR |= 1<<1) ;
#define GPIOC_PCLK_EN() (RCC -> AHB1ENR |= 1<<2) ;
#define GPIOD_PCLK_EN() (RCC -> AHB1ENR |= 1<<3) ;
#define GPIOE_PCLK_EN() (RCC -> AHB1ENR |= 1<<4) ;
#define GPIOF_PCLK_EN() (RCC -> AHB1ENR |= 1<<5) ;
#define GPIOG_PCLK_EN() (RCC -> AHB1ENR |= 1<<6) ;
#define GPIOH_PCLK_EN() (RCC -> AHB1ENR |= 1<<7) ;
#define GPIOI_PCLK_EN() (RCC -> AHB1ENR |= 1<<8) ;


//clock disable macros


#define GPIOA_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0));
}while(0) ;

#define GPIOB_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 1)); (RCC->AHB1RSTR &= ~(1 << 1));
}while(0) ;

#define GPIOC_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 2)); (RCC->AHB1RSTR &= ~(1 << 2));
}while(0) ;

#define GPIOD_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 3)); (RCC->AHB1RSTR &= ~(1 << 3));
}while(0) ;

#define GPIOE_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 4)); (RCC->AHB1RSTR &= ~(1 << 4));
}while(0) ;

#define GPIOF_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 5)); (RCC->AHB1RSTR &= ~(1 << 5));
}while(0) ;

#define GPIOG_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 6)); (RCC->AHB1RSTR &= ~(1 << 6));
}while(0) ;

#define GPIOH_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 7)); (RCC->AHB1RSTR &= ~(1 << 7));
}while(0) ;
```

```
#define GPIOI_PCLK_DI() do{ (RCC->AHB1RSTR |= (1 << 8)); (RCC->AHB1RSTR &= ~(1 << 8));
}while(0) ;


#endif /* INC_STM32F4XX_H_ */
```

2.3 Source File

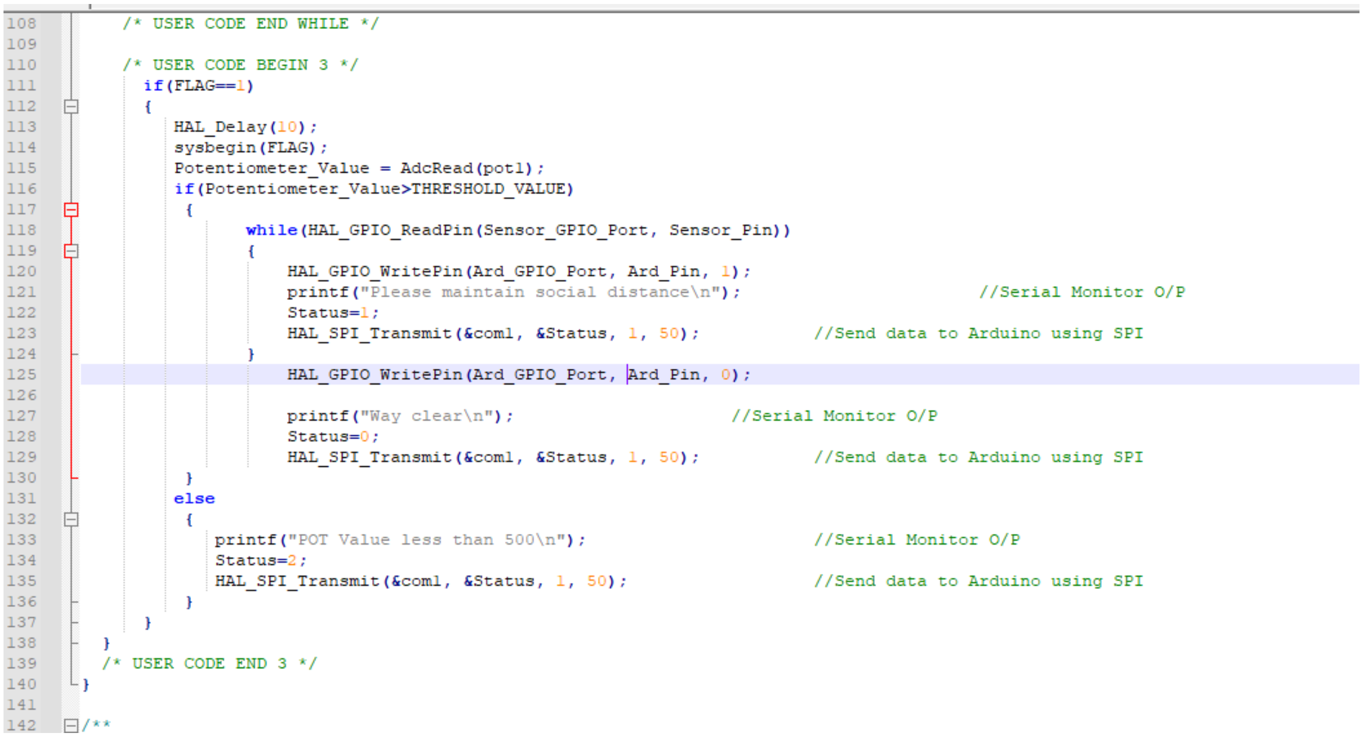
```
/*
 * STM32FXX_GPIO_Driver.c
 *
 *  Created on: Dec 28, 2020
 *      Author: 99003157
 */

#include "STM32FXX_GPIO_Driver.h"
#include<stdio.h>
void GPIO_PeriClockControl(GPIO_Regdef_t *pGPIOx, uint8_t EnorDi)
{
       if(EnorDi == ENABLE)
       {
             if(pGPIOx == GPIOA)
             {GPIOA_PCLK_EN();}

             else if(pGPIOx == GPIOB)
             {GPIOB_PCLK_EN();}

             else if(pGPIOx == GPIOC)
             {GPIOC_PCLK_EN();}

             else if(pGPIOx == GPIOD)
             {GPIOD_PCLK_EN();}

             else if(pGPIOx == GPIOE)
             {GPIOE_PCLK_EN();}

             else if(pGPIOx == GPIOF)
             {GPIOF_PCLK_EN();}

             else if(pGPIOx == GPIOG)
             {GPIOG_PCLK_EN();}

             else if(pGPIOx == GPIOH)
             {GPIOH_PCLK_EN();}

             else
             {GPIOI_PCLK_EN();}
       }
       else
       {
        if(EnorDi == DISABLE)
            {
                if(pGPIOx == GPIOA)
                {GPIOA_PCLK_DI();}

                else if(pGPIOx == GPIOB)
                {GPIOB_PCLK_DI();}

                else if(pGPIOx == GPIOC)
                {GPIOC_PCLK_DI();}

                else if(pGPIOx == GPIOD)
                {GPIOD_PCLK_DI();}

                else if(pGPIOx == GPIOE)
                {GPIOE_PCLK_DI();}

                else if(pGPIOx == GPIOF)
                {GPIOF_PCLK_DI();}

                else if(pGPIOx == GPIOG)
                {GPIOG_PCLK_DI();}

                else if(pGPIOx == GPIOH)
                {GPIOH_PCLK_DI();}

                else
                {GPIOI_PCLK_DI();}
            }
    }
}

void GPIO_Init(GPIO_HANDLE_t *pGPIOHandle)
{
    GPIO_PIN_CONFIG_t *a;
    if(a->GPIO_PinMode <= GPIO_PIN_MODE_Analog)//configure mode
    {uint32_t temp=0;

    temp = pGPIOHandle->GPIO_PIN_CONFIG.GPIO_PinMode << (2*pGPIOHandle-
>GPIO_PIN_CONFIG.GPIO_PinNumber);
    pGPIOHandle -> pGPIOx -> MODER |= temp;
    }
    //configure speed
    uint32_t temp1=0;

        temp1 = pGPIOHandle -> GPIO_PIN_CONFIG.GPIO_PinSpeed << (2*pGPIOHandle-
>GPIO_PIN_CONFIG.GPIO_PinNumber);
        pGPIOHandle->pGPIOx->OSPEEDR|=temp1;

    //configure PU PD
    uint32_t temp2=0;
```

```
                temp2 = pGPIOHandle ->GPIO_PIN_CONFIG.GPIO_PinPuPdControl << (2*pGPIOHandle-
>GPIO_PIN_CONFIG.GPIO_PinNumber);
                pGPIOHandle->pGPIOx->PUPDR|=temp2;

        //configure output type
        uint32_t temp3=0;

                temp3 = pGPIOHandle ->GPIO_PIN_CONFIG.GPIO_PinOPType << (2*pGPIOHandle-
>GPIO_PIN_CONFIG.GPIO_PinNumber);
                pGPIOHandle->pGPIOx->OTYPER|=temp3;

        if(pGPIOHandle->GPIO_PIN_CONFIG.GPIO_PinMode==GPIO_PIN_MODE_AFM)
        {uint32_t temp4,temp5;

                temp4=pGPIOHandle->GPIO_PIN_CONFIG.GPIO_PinNumber/8;
                temp5=pGPIOHandle->GPIO_PIN_CONFIG.GPIO_PinNumber%8;
                pGPIOHandle->pGPIOx->AFR[temp4]|=pGPIOHandle-
>GPIO_PIN_CONFIG.GPIO_PinAltFunMode<<(4*temp5);
        }
}
void GPIO_DeInit(GPIO_Regdef_t *pGPIOx)
{


                                 if(pGPIOx == GPIOA)
                                 {GPIOA_PCLK_DI();}

                                 else if(pGPIOx == GPIOB)
                                 {GPIOB_PCLK_DI();}

                                 else if(pGPIOx == GPIOC)
                                 {GPIOC_PCLK_DI();}

                                 else if(pGPIOx == GPIOD)
                                 {GPIOD_PCLK_DI();}

                                 else if(pGPIOx == GPIOE)
                                 {GPIOE_PCLK_DI();}

                                 else if(pGPIOx == GPIOF)
                                 {GPIOF_PCLK_DI();}

                                 else if(pGPIOx == GPIOG)
                                 {GPIOG_PCLK_DI();}

                                 else if(pGPIOx == GPIOH)
                                 {GPIOH_PCLK_DI();}

                                 else
                                 {GPIOI_PCLK_DI();}

}
```

```
uint8_t GPIO_ReadFromInputPin(GPIO_Regdef_t *pGPIOx,uint8_t PinNumber)
{
        uint8_t Value;
        Value=(pGPIOx->IDR>>PinNumber)*(0x00000001);
        return Value;
}
uint16_t GPIO_ReadFromInputPort(GPIO_Regdef_t *pGPIOx)
{
        uint16_t value1;
        value1=(uint16_t)(pGPIOx->IDR);
        return value1;
}
void GPIO_WriteToOutputPin(GPIO_Regdef_t *pGPIOx,uint8_t PinNumber,uint8_t Value)
{
        if(Value==GPIO_PIN_SET)
        {
                pGPIOx->ODR|=(1<<PinNumber);
        }
        else
        {
                pGPIOx->ODR&=~(1<<PinNumber);
        }
}
void GPIO_WriteToOutputPort(GPIO_Regdef_t *pGPIOx,uint16_t Value)
{
        pGPIOx->ODR=Value;
}
void GPIO_ToggleOutputPin(GPIO_Regdef_t *pGPIOx,uint8_t PinNumber)
{
        pGPIOx->ODR=pGPIOx->ODR^(1<<PinNumber);
}

//void GPIO_IRQConfig(uint8_t IRQNumber,uint8_t IRQPriority,uint8_t EnorDi);
//void GPIO_IRQHandling(uint8_t PinNumber);
```

3. Activity 3 (Mini Project)

3.1 Main Logic



```
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
      if(FLAG==1)
      {
         HAL_Delay(10);
         sysbegin(FLAG);
         Potentiometer_Value = AdcRead(pot1);
         if(Potentiometer_Value>THRESHOLD_VALUE)
          {
                while(HAL_GPIO_ReadPin(Sensor_GPIO_Port, Sensor_Pin))
                {
                    HAL_GPIO_WritePin(Ard_GPIO_Port, Ard_Pin, 1);
                    printf("Please maintain social distance\n");                       //Serial Monitor O/P
                    Status=1;
                    HAL_SPI_Transmit(&coml, &Status, 1, 50);           //Send data to Arduino using SPI
                }
                    HAL_GPIO_WritePin(Ard_GPIO_Port, Ard_Pin, 0);

                    printf("Way clear\n");                     //Serial Monitor O/P
                    Status=0;
                    HAL_SPI_Transmit(&coml, &Status, 1, 50);           //Send data to Arduino using SPI
          }
         else
          {
             printf("POT Value less than 500\n");                      //Serial Monitor O/P
             Status=2;
             HAL_SPI_Transmit(&coml, &Status, 1, 50);                  //Send data to Arduino using SPI
          }
      }
  }
  /* USER CODE END 3 */
}

/**
```

Figure 6. Main Logic function

3.2 Arduino Code

```
#include<SPI.h>

volatile boolean info;
volatile int Slave_data;


void setup()

{
 Serial.begin(9600);

pinMode(MISO, OUTPUT);
```

```
  SPCR |= _BV(SPE);                   //Turn on SPI in Slave Mode
  info = false;

  SPI.attachInterrupt();               //Interuupt ON is set for SPI commnucation
  }

ISR (SPI_STC_vect)                    //Inerrrput routine function
{
  Slave_data = SPDR;        // Value received from master
  info = true;              //Sets received as True

}

void loop()
{ if(info)
  {
 delay(500);
 Serial.println(Slave_data);
     if(Slave_data==0)
     {

            Serial.println("Please maintain social distancing \n");
     }
     else if(Slave_data==1)
       {
            Serial.println("Way clear\n");
       }
     else
     {
            Serial.println("Sensor value is less than 500\n");

     }
}
}
```

4. References

[1]https://www.youtube.com/watch?v=5aafG5mjZ_Y&list=PLERTijJOmYrDiiWd10iRHY0VRHdJwUH4g&index=5

[2] https://youtu.be/B7oKdUvRhQQ

[3] https://youtu.be/5aafG5mjZ_Y

[4] https://youtu.be/Bsq6P1B8JqI

[5] https://youtu.be/2Hm8eEHsgls