s./

GENESIS Learning Report -

Embedded C: Hardware + Programming + Testing

(99003157)



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

Contents

[1. Activity1 4](#_Toc60585566)

[**1.1 Linker Script** 4](#_Toc60585567)

[**1.2 Make file** 5](#_Toc60585568)

[**1.2.1 Main code** 5](#_Toc60585569)

[**1.2.2 Make file execution** 6](#_Toc60585571)

[**1.2.3 Make file code** 7](#_Toc60585573)

[**1.3 Startup** 7](#_Toc60585574)

[**1.4 Output Files** 8](#_Toc60585575)

[**1.6 Debugging Techniques** 9](#_Toc60585577)

[**1.6.1 Step in, step over and step out** 9](#_Toc60585578)

[**1.6.2 Disassembly** 9](#_Toc60585579)

[**1.6.3 Break points** 9](#_Toc60585580)

[2. Activity 2 (Driver code development) 11](#_Toc60585581)

[**2.1 MCU Specific Header File** 11](#_Toc60585582)

[**2.2 GPIO Driver File** 11](#_Toc60585583)

[**2.3 Source File** 16](#_Toc60585584)

[3. Activity 3 (Mini Project) 26](#_Toc60585586)

[**3.1 Main Logic** 26](#_Toc60585587)

[**3.2 Arduino Code** 26](#_Toc60585588)

[4. References 28](#_Toc60585590)

Table of Figures

[Figure 1.Linker Script 5](#_Toc60584397)

[Figure 2. Main code **Error! Bookmark not defined.**](file:///C:\Users\Deepak%20Sahoo\Desktop\learning_report_embedded.docx#_Toc60584398)

[Figure 3. Make file execution 7](#_Toc60584399)

[Figure 4. Make file code 7](file:///C:\Users\Deepak%20Sahoo\Desktop\learning_report_embedded.docx#_Toc60584400)

[Figure 5. Startup 8](file:///C:\Users\Deepak%20Sahoo\Desktop\learning_report_embedded.docx#_Toc60584401)

[Figure 6. Output files 8](file:///C:\Users\Deepak%20Sahoo\Desktop\learning_report_embedded.docx#_Toc60584402)

[Figure 7. Main Logic function 26](file:///C:\Users\Deepak%20Sahoo\Desktop\learning_report_embedded.docx#_Toc60584403)

List of Tables

No table of figures entries found.

# 1. Activity1

## **1.1 Linker Script**

Graphical user interface, text, application

Description automatically generated

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;**

### **}**

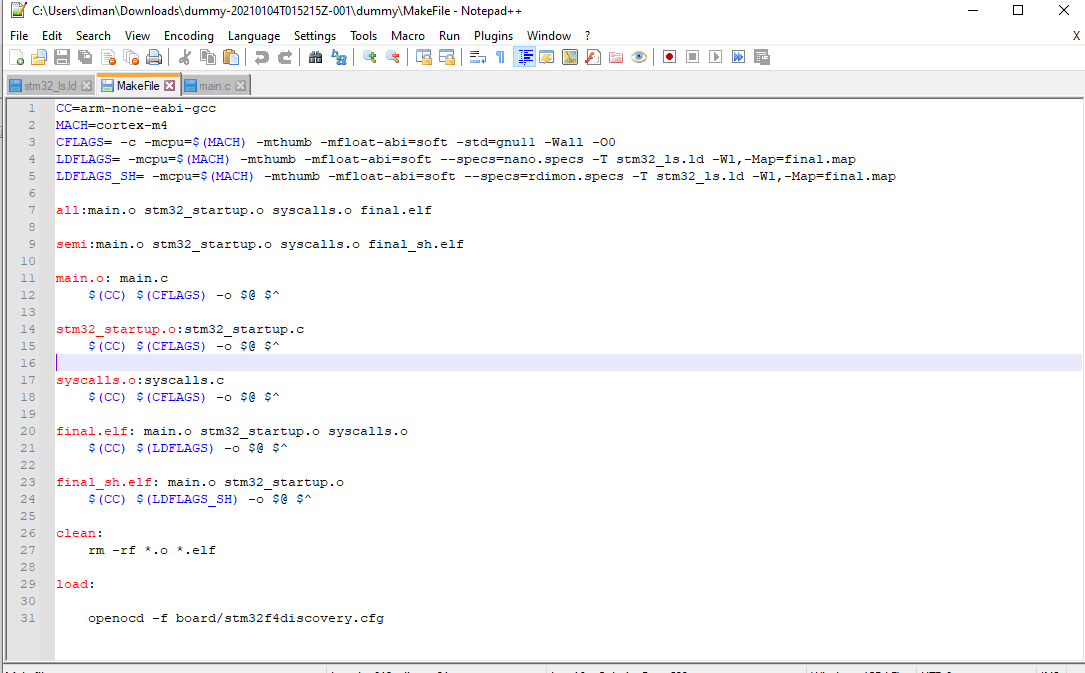
### **1.2.2 Make file execution**

### 

Figure 2. Make file execution

### **1.2.3 Make file code**

Fig 3. Make file code



## **1.3 Startup**

A picture containing text, black, screenshot

Description automatically generated

Figure 4. Startup

## **1.4 Output Files**

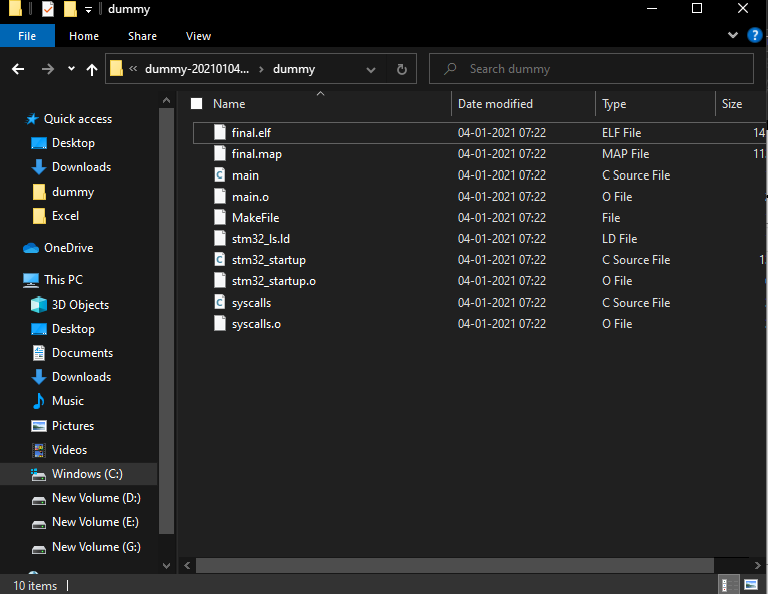


Figure 5. Output files

## **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**

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>