s./

GENESIS Learning Report -

Embedded C: Hardware + Programming + Testing

(99003161)



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1.0 | 24-12-20 | DEEPAK KUMAR SAHOO |  | Dr. Vivek K and Bhargav N |  |
| 1.1 | 28-12-20 | DEEPAK KUMAR SAHOO |  | Dr. Vivek K and Bhargav N |  |
| 1.2 | 29-12-20 | DEEPAK KUMAR SAHOO |  | Dr. Vivek K and Bhargav N |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

Contents

[1. Activity1 5](#_Toc60585566)

[**1.1 Linker Script** 5](#_Toc60585567)

[**1.2 Make file** 6](#_Toc60585568)

[**1.2.1 Main code** 6](#_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)

[**1.5 GitHub Link of code files** 8](#_Toc60585576)

[**1.6 Debugging Techniques** 8](#_Toc60585577)

[**1.6.1 Step in, step over and step out** 8](#_Toc60585578)

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

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

[2. Activity 2 (Driver code development) 10](#_Toc60585581)

[**2.1 MCU Specific Header File** 10](#_Toc60585582)

[**2.2 GPIO Driver File** 13](#_Toc60585583)

[**2.3 Source File** 15](#_Toc60585584)

[**2.4 Github link to these files:** 20](#_Toc60585585)

[3. Activity 3 (Mini Project) 21](#_Toc60585586)

[**3.1 Main Logic** 21](#_Toc60585587)

[**3.2 Arduino Code** 21](#_Toc60585588)

[**3.3 Github link to the project** 22](#_Toc60585589)

[4. References 23](#_Toc60585590)

Table of Figures

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

[Figure 2. Main code 6](file:///C:\Users\Deepak%20Sahoo\Desktop\learning_report_embedded.docx#_Toc60584398)

[Figure 3. Make file execution 6](#_Toc60584399)

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

[Figure 5. Startup 7](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 21](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

Figure .Linker Script

## **1.2 Make file**

### **1.2.1 Main code**

### Graphical user interface, text, application Description automatically generated

Figure . Main code

### **1.2.2 Make file execution**

### 

Figure . Make file execution

### **1.2.3 Make file code**

Figure . Make file code

## **1.3 Startup**

A picture containing text, black, screenshot

Description automatically generated

Figure . Startup

## **1.4 Output Files**

Graphical user interface, text, application

Description automatically generated

Figure . Output files

## **1.5 GitHub Link of code files**

[embeddedC/embedded at master · 99003161/embeddedC (github.com)](https://github.com/99003161/embeddedC/tree/master/embedded)

## **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**

**#include** <stdint.h>

**#ifndef** INC\_STM32F407XX\_H\_

**#define** INC\_STM32F407XX\_H\_

//To find base addresses of the memories

**#define** FLASH\_BASEADDR 0x08000000U

**#define** SRAM1\_BASEADDR 0x20000000U

**#define** SRAM2\_BASEADDR 0x2001C000U

**#define** SRAM 0x20000000U

**#define** ROM\_BASEADDR 0x1FFF0000U

// To find bus address

**#define** APB1\_BASEADDR 0x40000000U

**#define** APB2\_BASEADDR 0x40010000U

**#define** AHB1\_BASEADDR 0x40020000U

**#define** AHB2\_BASEADDR 0x50000000U

//To find the base address of the peripheral hanging on AHB1

**#define** GPIOA\_BASEADDR (AHB1\_BASEADDR + 0x0000U)

**#define** GPIOB\_BASEADDR (AHB1\_BASEADDR + 0x0400U)

**#define** GPIOC\_BASEADDR (AHB1\_BASEADDR + 0x0800U)

**#define** GPIOD\_BASEADDR (AHB1\_BASEADDR + 0x0C00U)

**#define** GPIOE\_BASEADDR (AHB1\_BASEADDR + 0x1000U)

**#define** GPIOF\_BASEADDR (AHB1\_BASEADDR + 0x1400U)

**#define** GPIOG\_BASEADDR (AHB1\_BASEADDR + 0x1800U)

**#define** GPIOH\_BASEADDR (AHB1\_BASEADDR + 0x1C00U)

**#define** GPIOI\_BASEADDR (AHB1\_BASEADDR + 0x2000U)

**#define** GPIOJ\_BASEADDR (AHB1\_BASEADDR + 0x2400U)

**#define** GPIOK\_BASEADDR (AHB1\_BASEADDR + 0x2800U)

**#define** RCC\_BASEADDR (AHB1\_BASEADDR + 0x3800U)

//To find the base address of the peripheral hanging on APB1

**#define** I2C1 (APB1\_BASEADDR + 0x5400U)

**#define** I2C2 (APB1\_BASEADDR + 0x5800U)

**#define** I2C3 (APB1\_BASEADDR + 0x5C00U)

**#define** SPI2 (APB1\_BASEADDR + 0x3800U)

**#define** SPI3 (APB1\_BASEADDR + 0x3C00U)

**#define** USART2 (APB1\_BASEADDR + 0x4400U)

**#define** USART3 (APB1\_BASEADDR + 0x4800U)

//To find the base address of the peripheral hanging on APB2

**#define** SPI1 (APB2\_BASEADDR + 0x3000U)

**#define** USART1 (APB2\_BASEADDR + 0x1000U)

**#define** USART6 (APB2\_BASEADDR + 0x1400U)

**#define** EXTI (APB2\_BASEADDR + 0x3C00U)

**#define** SYSCFG (APB2\_BASEADDR + 0x3800U)

//Registers or Reg structures of GPIOA peripherals

**typedef** **struct**

{

**volatile** uint32\_t MODER;

**volatile** uint32\_t OTYPER;

**volatile** uint32\_t OSPEEDR;

**volatile** uint32\_t PUPDR;

**volatile** uint32\_t IDR;

**volatile** uint32\_t ODR;

**volatile** uint32\_t BSRR;

**volatile** uint32\_t LCKR;

**volatile** uint32\_t AFRL;

**volatile** uint32\_t AFRH;

}GPIO\_REGDEF\_t;

//Registers or Reg structure of RCC

**typedef** **struct**

{

**volatile** uint32\_t CR;

**volatile** uint32\_t PLLCFGR;

**volatile** uint32\_t CFGR;

**volatile** uint32\_t CIR;

**volatile** uint32\_t AHB1RSTR;

**volatile** uint32\_t AHB2RSTR;

**volatile** uint32\_t AHB3RSTR;

**volatile** uint32\_t RESERVED0;

**volatile** uint32\_t APB1RSTR;

**volatile** uint32\_t APB2RSTR;

**volatile** uint32\_t RESERVED1[0];

**volatile** uint32\_t RESERVED1[1];

**volatile** uint32\_t AHB1ENR;

**volatile** uint32\_t AHB2ENR;

**volatile** uint32\_t AHB3ENR;

**volatile** uint32\_t RESERVED2;

**volatile** uint32\_t APB1ENR;

**volatile** uint32\_t APB2ENR;

**volatile** uint32\_t RESERVED3[0];

**volatile** uint32\_t RESERVED3[1];

**volatile** uint32\_t AHB1LPENR;

**volatile** uint32\_t AHB2LPENR;

**volatile** uint32\_t AHB3LPENR;

**volatile** uint32\_t RESERVED4;

**volatile** uint32\_t APB1LPENR;

**volatile** uint32\_t APB2LPENR;

**volatile** uint32\_t RESERVED5[0];

**volatile** uint32\_t RESERVED5[1];

**volatile** uint32\_t BDCR;

**volatile** uint32\_t CSR;

**volatile** uint32\_t RESERVED6[0];

**volatile** uint32\_t RESERVED6[1];

**volatile** uint32\_t SSCGR;

**volatile** uint32\_t PLLI2SCFGR;

**volatile** uint32\_t PLLSAICFGR;

**volatile** uint32\_t DCKCFGR;

}RCC\_REGDEF\_t;

//Peripheral definition for GPIO

**#define** GPIOA ((GPIO\_REGDEF\_t\*) GPIOA\_BASEADDR)

**#define** GPIOB ((GPIO\_REGDEF\_t\*) GPIOB\_BASEADDR)

**#define** GPIOC ((GPIO\_REGDEF\_t\*) GPIOC\_BASEADDR)

**#define** GPIOD ((GPIO\_REGDEF\_t\*) GPIOD\_BASEADDR)

**#define** GPIOE ((GPIO\_REGDEF\_t\*) GPIOE\_BASEADDR)

**#define** GPIOF ((GPIO\_REGDEF\_t\*) GPIOF\_BASEADDR)

**#define** GPIOG ((GPIO\_REGDEF\_t\*) GPIOG\_BASEADDR)

**#define** GPIOH ((GPIO\_REGDEF\_t\*) GPIOH\_BASEADDR)

**#define** GPIOI ((GPIO\_REGDEF\_t\*) GPIOI\_BASEADDR)

// Peripheral definition for RCC

**#define** RCC ((RCC\_REGDEF\_t\*) RCC\_BASEADDR)

//Clock enable macros for GPIOx

**#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 for GPIOx

**#define** GPIOA\_PCLK\_DI() (RCC->AHB1ENR &= ~(1<<0))

**#define** GPIOB\_PCLK\_DI() (RCC->AHB1ENR &= ~(1<<1))

**#define** GPIOC\_PCLK\_DI() (RCC->AHB1ENR &= ~(1<<2))

**#define** GPIOD\_PCLK\_DI() (RCC->AHB1ENR &= ~(1<<3))

**#define** GPIOE\_PCLK\_DI() (RCC->AHB1ENR &= ~(1<<4))

**#define** GPIOF\_PCLK\_DI() (RCC->AHB1ENR &= ~(1<<5))

**#define** GPIOG\_PCLK\_DI() (RCC->AHB1ENR &= ~(1<<6))

**#define** GPIOH\_PCLK\_DI() (RCC->AHB1ENR &= ~(1<<7))

**#define** GPIOI\_PCLK\_DI() (RCC->AHB1ENR &= ~(1<<8))

// some important macros

**#define** ENABLE 1

**#define** DISABLE 0

**#define** SET ENABLE

**#define** RESET DISABLE

**#define** GPIO\_PIN\_SET SET

**#define** GPIO\_PIN\_RESET RESET

// GPIO Reset

**#define** GPIOA\_REG\_RESET() **do**{(RCC->AHB1RSTR |=1<<0); (RCC->AHB1RSTR &=~(1<<0));}**while**(0)

**#define** GPIOB\_REG\_RESET() **do**{(RCC->AHB1RSTR |=1<<1); (RCC->AHB1RSTR &=~(1<<1));}**while**(0)

**#define** GPIOC\_REG\_RESET() **do**{(RCC->AHB1RSTR |=1<<2); (RCC->AHB1RSTR &=~(1<<2));}**while**(0)

**#define** GPIOD\_REG\_RESET() **do**{(RCC->AHB1RSTR |=1<<3); (RCC->AHB1RSTR &=~(1<<3));}**while**(0)

**#define** GPIOE\_REG\_RESET() **do**{(RCC->AHB1RSTR |=1<<4); (RCC->AHB1RSTR &=~(1<<4));}**while**(0)

**#define** GPIOF\_REG\_RESET() **do**{(RCC->AHB1RSTR |=1<<5); (RCC->AHB1RSTR &=~(1<<5));}**while**(0)

**#define** GPIOG\_REG\_RESET() **do**{(RCC->AHB1RSTR |=1<<6); (RCC->AHB1RSTR &=~(1<<6));}**while**(0)

**#define** GPIOH\_REG\_RESET() **do**{(RCC->AHB1RSTR |=1<<7); (RCC->AHB1RSTR &=~(1<<7));}**while**(0)

**#define** GPIOI\_REG\_RESET() **do**{(RCC->AHB1RSTR |=1<<8); (RCC->AHB1RSTR &=~(1<<8));}**while**(0)

**#endif** /\* INC\_STM32F407XX\_H\_ \*/

## **2.2 GPIO Driver File**

/\*

\* stm32f407xx\_gpio\_driver.h

\*/

**#ifndef** INC\_STM32F407XX\_GPIO\_DRIVER\_H\_

**#define** INC\_STM32F407XX\_GPIO\_DRIVER\_H\_

**#include** "STM32F407xx.h"

**#include** <stdint.h>

**typedef** **struct**

{

uint8\_t GPIO\_PINNUMBER;

uint8\_t GPIO\_PINNAME;

uint8\_t GPIO\_PINMODE;

uint8\_t GPIO\_PINSPEED;

uint8\_t GPIO\_PINOPTYPE;

uint8\_t GPIO\_PINPUPDCONTROL;

uint8\_t GPIO\_PINALTFUNMODE;

}GPIO\_PINCONFIG\_t;

**typedef** **struct**

{

GPIO\_REGDEF\_t \*pGPIOx;

GPIO\_PINCONFIG\_t GPIO\_PINCONFIG;

}GPIO\_HANDLE\_t;

//GPIO\_PinNumber of GPIOx

**#define** GPIO\_PINNUMBER\_0 0

**#define** GPIO\_PINNUMBER\_1 1

**#define** GPIO\_PINNUMBER\_2 2

**#define** GPIO\_PINNUMBER\_3 3

**#define** GPIO\_PINNUMBER\_4 4

**#define** GPIO\_PINNUMBER\_5 5

**#define** GPIO\_PINNUMBER\_6 6

**#define** GPIO\_PINNUMBER\_7 7

**#define** GPIO\_PINNUMBER\_8 8

**#define** GPIO\_PINNUMBER\_9 9

**#define** GPIO\_PINNUMBER\_10 10

**#define** GPIO\_PINNUMBER\_11 11

**#define** GPIO\_PINNUMBER\_12 12

**#define** GPIO\_PINNUMBER\_13 13

**#define** GPIO\_PINNUMBER\_14 14

**#define** GPIO\_PINNUMBER\_15 15

//Possible Modes

**#define** GPIO\_MODE\_IN 0

**#define** GPIO\_MODE\_OUT 1

**#define** GPIO\_MODE\_AFN 2

**#define** GPIO\_MODE\_ANALOG 3

//GPIO\_PinSpeed

**#define** GPIO\_SPEED\_LOW 0

**#define** GPIO\_SPEED\_MEDIUM 1

**#define** GPIO\_SPEED\_HIGH 2

**#define** GPIO\_SPEED\_VERY\_HIGH 3

//GPIO\_PinOPType

**#define** GPIO\_OP\_TYPE\_PP 0

**#define** GPIO\_OP\_TYPE\_OD 1

//GPIO\_PinPuPdControl

**#define** GPIO\_NO\_PU\_PD 0

**#define** GPIO\_PU 1

**#define** GPIO\_PD 2

**#define** GPIO\_RESERVED 3

**void** **GPIO\_INIT**(GPIO\_HANDLE\_t \*GPIOHANDLE);

**void** **GPIO\_DINIT**(GPIO\_REGDEF\_t \*pGPIOx);

**void** **GPIO\_PERICLKCTRL**(GPIO\_REGDEF\_t \*pGPIOx, uint8\_t EorDi);

uint8\_t **GPIO\_READFROMIPPIN**(GPIO\_REGDEF\_t \*pGPIOx, uint8\_t PINNO);

uint16\_t **GPIO\_READFROMIPPORT**(GPIO\_REGDEF\_t \*pGPIOx);

**void** **GPIO\_WRITETOOUPUTPIN**(GPIO\_REGDEF\_t \*pGPIOx, uint8\_t PINNO, uint8\_t VALUE);

**void** **GPIO\_WRITETOOUPUTPORT**(GPIO\_REGDEF\_t \*pGPIOx, uint16\_t VALUE);

**void** **GPIO\_TOGGLEOUTPUTPIN**(GPIO\_REGDEF\_t \*pGPIOx, uint8\_t PINNO);

**#endif** /\* INC\_STM32F407XX\_GPIO\_DRIVER\_H\_ \*/

## **2.3 Source File**

/\*

\* stm32f407xx\_gpio\_driver.c

\*/

**#include** "stm32f407xx\_gpio\_driver.h"

**#include** <stdint.h>

/\*

\*

\* @Brief description: function to eanble RCC clock

\*

\* @Function-GPIOX\_PCLK\_EN(),where x=a..i

\* @Param1-GPIO\_REGDEF\_t \*pGPIOx

\* @Param2-uint8\_t EnorD

\* @Definition-GPIO Clock enable and disable-

\*/

**void** **GPIO\_PERICLKCTRL**(GPIO\_REGDEF\_t \*pGPIOx, uint8\_t EorDi)

{

**if**(EorDi==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** **if**(pGPIOx == GPIOI)

{

GPIOI\_PCLK\_EN();

}

}

**else**

{

**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** **if**(pGPIOx == GPIOI)

{

GPIOI\_PCLK\_DI();

}

}

}

/\*

\*

\* @Brief description:GPIO Clock enable and disable

\* @Function-gPIO Port intzatization

\* @Param1-GPIO\_Handle\_t \*

\*/

**void** **GPIO\_INIT**(GPIO\_HANDLE\_t \*GPIOHANDLE)

{

//Init Mode

uint32\_t temp=0;

temp = (GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINMODE<<(2 \*(GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER)));

(GPIOHANDLE->pGPIOx->MODER) &= ~(0x03<<2\*GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER); //First Reset the same pins and then set the values

GPIOHANDLE->pGPIOx->MODER |=temp;

//Config Speed

temp=0;

temp = (GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINSPEED <<(2 \* (GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER)));

(GPIOHANDLE->pGPIOx->OSPEEDR) &= ~(0x03<<2\*GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER);

GPIOHANDLE->pGPIOx->OSPEEDR |=temp;

//Config Pull up and pull down

temp=0;

temp = (GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINPUPDCONTROL << (2\*(GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER)));

(GPIOHANDLE->pGPIOx->PUPDR) &= ~(0x03<<2\*GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER);

GPIOHANDLE->pGPIOx->PUPDR |=temp;

//Config Output type

temp=0;

temp = (GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINOPTYPE << (GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER));

(GPIOHANDLE->pGPIOx->OTYPER) &= ~(0x01<<GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER);

GPIOHANDLE->pGPIOx->OTYPER |=temp;

//Config Alternating Function

**if**(GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINMODE == GPIO\_MODE\_AFN)

{

uint8\_t temp1=0, temp2=0;

temp1 = GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER/8;

temp2 = GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINNUMBER%8;

GPIOHANDLE->pGPIOx->AFR[temp1] &= ~(0x0F<<4\*temp2);

GPIOHANDLE->pGPIOx->AFR[temp1]= GPIOHANDLE->GPIO\_PINCONFIG.GPIO\_PINALTFUNMODE<<(4\*temp2);

}

}

/\*

\*

\*

\* @Brief description:reset of GPIO port

\* @Function-GPIO\_DInit

\* @Param1-GPIO\_REGDEF\_t\*

\*/

**void** **GPIO\_DINIT**(GPIO\_REGDEF\_t \*pGPIOx)

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_REG\_RESET();

}

**else** **if**(pGPIOx == GPIOB)

{

GPIOB\_REG\_RESET();

}

**else** **if**(pGPIOx == GPIOC)

{

GPIOC\_REG\_RESET();

}

**else** **if**(pGPIOx == GPIOD)

{

GPIOD\_REG\_RESET();

}

**else** **if**(pGPIOx == GPIOE)

{

GPIOE\_REG\_RESET();

}

**else** **if**(pGPIOx == GPIOF)

{

GPIOF\_REG\_RESET();

}

**else** **if**(pGPIOx == GPIOG)

{

GPIOG\_REG\_RESET();

}

**else** **if**(pGPIOx == GPIOH)

{

GPIOH\_REG\_RESET();

}

**else** **if**(pGPIOx == GPIOI)

{

GPIOI\_REG\_RESET();

}

}

/\*

\*

\*

\* @Brief description:GPIO Input from pin

\* @Function-GPIO\_ReadFromInptPin

\* @Param1-GPIO\_REGDEF\_t \*

\*

\*/

uint8\_t **GPIO\_READFROMIPPIN**(GPIO\_REGDEF\_t \*pGPIOx, uint8\_t PINNO)

{

uint8\_t value=0;

value = (uint8\_t)(pGPIOx->IDR>>PINNO)&(0x00000001);

**return** value;

}

/\*

\*

\*

\* \* @Brief description:Input from port input

\* @Function-GPIO\_ReadFromInptPort

\* @Param1-GPIO\_REGDEF\_t \*

\*

\*/

uint16\_t **GPIO\_READFROMIPPORT**(GPIO\_REGDEF\_t \*pGPIOx)

{

uint16\_t value=0;

value = (uint16\_t)(pGPIOx->IDR);

**return** value;

}

/\*

\*

\*

\* @Brief description:output port pin set or reset

\* @Function-GPIO\_WriteToOutputPin

\* @Param1-GPIO\_REGDEF\_t \*

\*

\*

\*/

**void** **GPIO\_WRITETOOUPUTPIN**(GPIO\_REGDEF\_t \*pGPIOx, uint8\_t PINNO, uint8\_t VALUE)

{

**if**(VALUE == GPIO\_PIN\_SET)

{

pGPIOx->ODR|=(1<<PINNO);

}

**else**

{

pGPIOx->ODR&=~(1<<PINNO);

}

}

/\*

\*

\* @Brief description:to write to output port

\* @Function-GPIO\_WriteToOutputPort

\* @Param1-GPIO\_REGDEF\_t \*

\*

\*/

**void** **GPIO\_WRITETOOUPUTPORT**(GPIO\_REGDEF\_t \*pGPIOx, uint16\_t VALUE)

{

pGPIOx->ODR = VALUE;

}

/\*

\*

\* @Function-to call toggle function

\* @Param1-GPIO\_REGDEF\_t \*

\*

\*/

**void** **GPIO\_TOGGLEOUTPUTPIN**(GPIO\_REGDEF\_t \*pGPIOx, uint8\_t PINNO)

{

pGPIOx->ODR^=(1<<PINNO);

}

## **2.4 Github link to these files:**

[embeddedC/driver\_development\_99003161 at master · 99003161/embeddedC (github.com)](https://github.com/99003161/embeddedC/tree/master/driver_development_99003161)

# 3. Activity 3 (Mini Project)

## **3.1 Main Logic**

Graphical user interface, text, application

Description automatically generated

Figure . 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");

}

}

}

## **3.3 Github link to the project**

[embeddedC/Embedded\_C\_Mini\_Project at master · 99003161/embeddedC (github.com)](https://github.com/99003161/embeddedC/tree/master/Embedded_C_Mini_Project)

# 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>