s./



Learning Report – Embedded C – Hardware + Programming + Testing

Course Code: <CODE>

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

**Document History**

Contents

[1. Activity 4](#_Toc60519624)

[**1.1 Linker Script** 4](#_Toc60519625)

[**1.2 Make file** 5](#_Toc60519626)

[**1.3 Startup** 6](#_Toc60519627)

[**1.4 Output Files** 7](#_Toc60519628)

[**1.5 GitHub Link of code files** 7](#_Toc60519629)

[2. Activity 7](#_Toc60519630)

[**2.1 MCU Specific Header File** 8](#_Toc60519631)

[**2.2 GPIO Driver File** 12](#_Toc60519632)

[**2.3 Source File** 14](#_Toc60519633)

[3. Activity 18](#_Toc60519634)

[**3.1 Screenshot of main logic function** 18](#_Toc60519635)

[**3.2 Arduino Code** 18](#_Toc60519636)

[**3.3 GitHub link** 19](#_Toc60519637)

[References 19](#_Toc60519638)

**List of Figures**

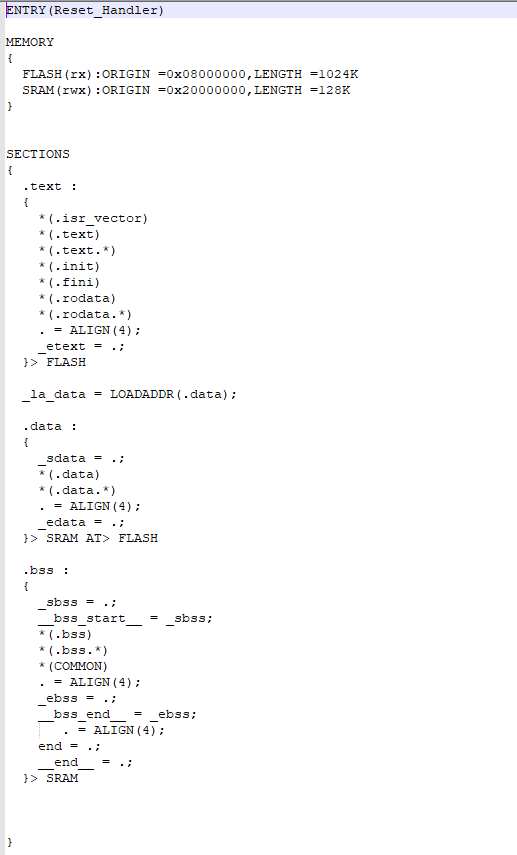
[Figure 1 Makefile 6](#_Toc60495667)

[Figure 2 Startup 7](#_Toc60495668)

[Figure 3 Output files 7](#_Toc60495669)

# 1. Activity

## **1.1 Linker Script**



## **1.2 Make file**

**Main code:**

#include <stdio.h>

int main() {

int i, n, t1 = 0, t2 = 1, nextTerm;

printf("Enter the number of terms: ");

scanf("%d", &n);

printf("Fibonacci Series: ");

for (i = 1; i <= n; ++i) {

printf("%d, ", t1);

nextTerm = t1 + t2;

t1 = t2;

t2 = nextTerm;

}

return 0;

}

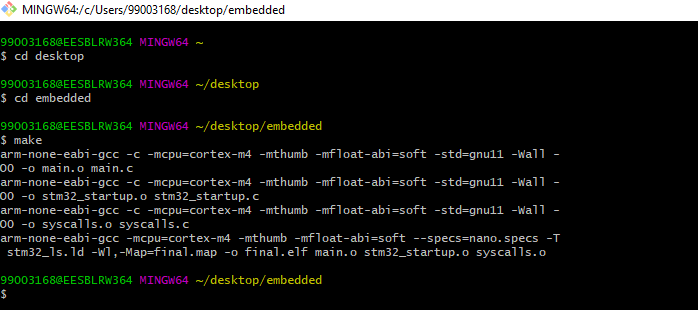


Figure 1 Makefile

## **1.3 Startup**

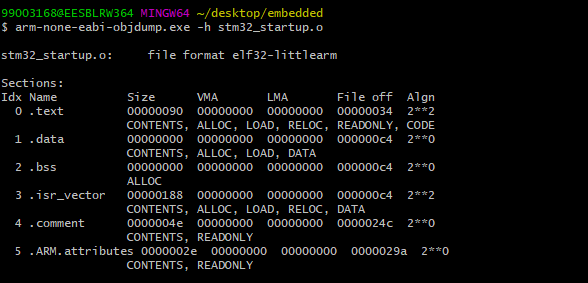


Figure 2 Startup

## **1.4 Output Files**

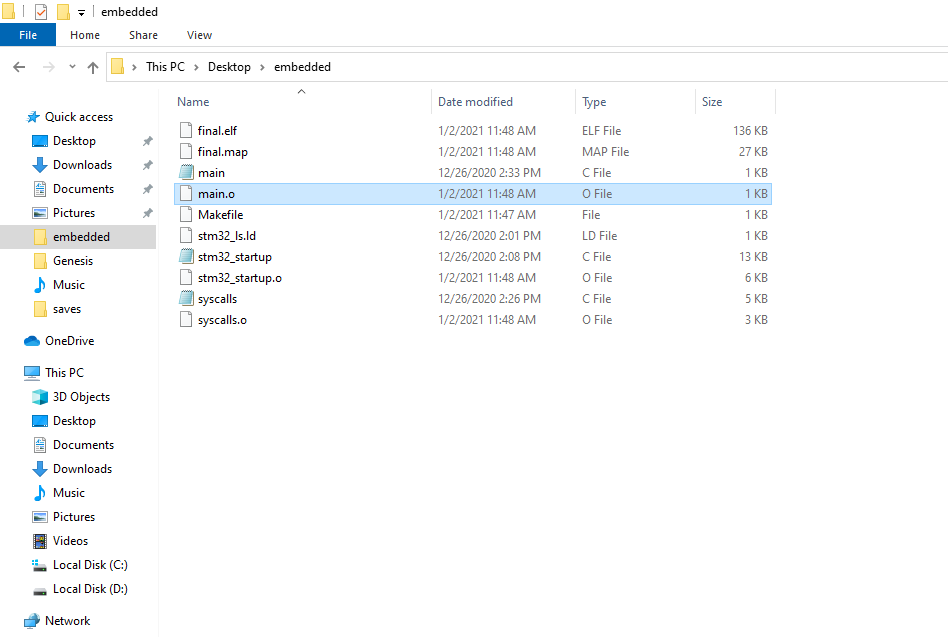


Figure 3 Output files

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

[99003168/Embedded-C (github.com)](https://github.com/99003168/Embedded-C)

# 2. Activity

## **2.1 MCU Specific Header File**

/\*

\* STM32F4xx.h => MCU specific header file =>driver level

\*

\* Created on: Dec 28, 2020

\* Author: 99003171

\*/

**#ifndef** INC\_STM32F4XX\_H\_

**#define** INC\_STM32F4XX\_H\_

**#include**<stdint.h>

**#define** \_\_vo **volatile**

//defining macros for the various memories

**#define** FLASHADDR 0x08000000U

**#define** SRAM1ADDR 0x20000000U

**#define** SRAM2ADDR 0x2001C000U

**#define** SRAMADDR SRAM1ADDR //same as SRAM1

**#define** ROM\_BASEADDR 0x1FFF0000U

//defining macros for the various buses

**#define** APB1\_BASEADDR 0x40000000U

**#define** APB2\_BASEADDR 0x40010000U

**#define** AHB1\_BASEADDR 0x40020000U

**#define** AHB2\_BASEADDR 0x50000000U

**#define** PERI\_BASEADDR APB1\_BASEADDR //peripheral base address

//#define AHB3ADDR 0x60000000U

//defining macros for peripherals hanging on to AHB1 bus

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

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

//defining macros for peripherals hanging on to AHB2 bus

//NONE

//defining macros for peripherals hanging on to APB1 bus

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

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

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

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

**#define** UART4\_BASEADDR (APB1\_BASEADDR + (0x4C00U)) //UART peripheral

**#define** UART5\_BASEADDR (APB1\_BASEADDR + (0x5000U))

**#define** I2C1\_BASEADDR (APB1\_BASEADDR + (0x5400U)) //I2C peripheral

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

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

**#define** TIM2\_BASEADDR (APB1\_BASEADDR + (0x0000U)) //Timer peripheral

**#define** TIM3\_BASEADDR (APB1\_BASEADDR + (0x0400U))

**#define** TIM4\_BASEADDR (APB1\_BASEADDR + (0x0800U))

**#define** TIM5\_BASEADDR (APB1\_BASEADDR + (0x0C00U))

**#define** TIM6\_BASEADDR (APB1\_BASEADDR + (0x1000U))

**#define** TIM7\_BASEADDR (APB1\_BASEADDR + (0x1400U))

**#define** TIM12\_BASEADDR (APB1\_BASEADDR + (0x1800U))

**#define** TIM13\_BASEADDR (APB1\_BASEADDR + (0x1C00U))

**#define** TIM14\_BASEADDR (APB1\_BASEADDR + (0x2000U))

//defining macros for peripherals hanging on to APB2 bus

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

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

**#define** USART6\_BASEADDR (APB2\_BASEADDR + (0x0000U))

//#define ADC1-ADC2-ADC3\_BASEADDR (APB2\_BASEADDR + (0x2000U)) //ADC peripheral

**#define** TIM1\_BASEADDR (APB2\_BASEADDR + (0x2000U))// Timer peripheral

**#define** TIM8\_BASEADDR (APB2\_BASEADDR + (0x0400U))

**#define** TIM9\_BASEADDR (APB2\_BASEADDR + (0x4000U))

**#define** TIM10\_BASEADDR (APB2\_BASEADDR + (0x4400U))

**#define** TIM11\_BASEADDR (APB2\_BASEADDR + (0x4800U))

//defining macros for GPIO Peripheral registers

//#define GPIOA\_ODR\_BASEADDR (AHB1\_BASEADDR + (0x0000U) + (0x0014U))

**typedef** **struct**

{ // at port level definitions

\_\_vo uint32\_t MODER; // Address offset: 0x00

\_\_vo uint32\_t OTYPER; // Address offset: 0x04

\_\_vo uint32\_t OSPEEDR; // Address offset: 0x08

\_\_vo uint32\_t PUPDR; // Address offset: 0x0C

\_\_vo uint32\_t IDR; // Address offset: 0x10

\_\_vo uint32\_t ODR; // Address offset: 0x14

\_\_vo uint32\_t BSRRL; // Address offset: 0x18

\_\_vo uint32\_t BSRRH; // Address offset: 0x1A

\_\_vo uint32\_t LCKR; // Address offset: 0x1C

\_\_vo uint32\_t AFR [2]; // AFR[0] - AFR Low registers, AFR[1] - AFR high registers // Address offset: 0x20-0x24

} GPIO\_Reg\_def\_t;

//Reg\_def\_t \*pGPIOA = (Reg\_def\_t\*)GPIOA\_BASEADDR; //== #define GPIOA ((Reg\_def\_t\*)GPIOA\_BASEADDR)

**#define** GPIOA ((GPIO\_Reg\_def\_t\*)GPIOA\_BASEADDR)

**#define** GPIOB ((GPIO\_Reg\_def\_t\*)GPIOB\_BASEADDR)

**#define** GPIOC ((GPIO\_Reg\_def\_t\*)GPIOC\_BASEADDR)

**#define** GPIOD ((GPIO\_Reg\_def\_t\*)GPIOD\_BASEADDR)

**#define** GPIOE ((GPIO\_Reg\_def\_t\*)GPIOE\_BASEADDR)

**#define** GPIOF ((GPIO\_Reg\_def\_t\*)GPIOF\_BASEADDR)

**#define** GPIOG ((GPIO\_Reg\_def\_t\*)GPIOG\_BASEADDR)

**#define** GPIOH ((GPIO\_Reg\_def\_t\*)GPIOH\_BASEADDR)

**#define** GPIOI ((GPIO\_Reg\_def\_t\*)GPIOI\_BASEADDR)

// defining macros for RCC peripheral registers

**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 RCC1 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC2 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC3 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC4 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC5 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC6 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC7 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC8 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC9 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC10 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC11 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC12 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC13 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC14 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC15 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC16 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC17 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC18 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC19 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC20 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC21 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC22 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC23 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC24 ((RCC\_Reg\_def\_t\*)0x4002 3800)

#define RCC25 ((RCC\_Reg\_def\_t\*)0x4002 3800)

\*/

**#define** RCC ((RCC\_Reg\_def\_t\*)RCC\_BASEADDR)

//GPIO clock enable

**#define** GPIOA\_pclock\_enable() (RCC->AHB1ENR |=(1<<0))

**#define** GPIOB\_pclock\_enable() (RCC->AHB1ENR |=(1<<1))

**#define** GPIOC\_pclock\_enable() (RCC->AHB1ENR |=(1<<2))

**#define** GPIOD\_pclock\_enable() (RCC->AHB1ENR |=(1<<3))

**#define** GPIOE\_pclock\_enable() (RCC->AHB1ENR |=(1<<4))

**#define** GPIOF\_pclock\_enable() (RCC->AHB1ENR |=(1<<5))

**#define** GPIOG\_pclock\_enable() (RCC->AHB1ENR |=(1<<6))

**#define** GPIOH\_pclock\_enable() (RCC->AHB1ENR |=(1<<7))

**#define** GPIOI\_pclock\_enable() (RCC->AHB1ENR |=(1<<8))

//GPIO peripheral clock disable macros

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

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

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

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

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

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

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

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

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

//important macro definitions

**#define** ENABLE 1

**#define** DISABLE 0

**#define** GPIO\_Pin\_Set ENABLE

**#define** GPIO\_Pin\_Reset DIABLE

**#include** "STM32FXX\_GPIO\_DRIVER.h"

**#endif** /\* INC\_STM32F4XX\_H\_ \*/

## **2.2 GPIO Driver File**

/\*

\* STM32Fxx\_GPIO\_DRIVER.h

\*

\* Created on: Dec 28, 2020

\* Author: 99003171

\*/

**#ifndef** INC\_STM32FXX\_GPIO\_DRIVER\_H\_

**#define** INC\_STM32FXX\_GPIO\_DRIVER\_H\_

**#include** "STM32F4XX.h"

//GPIO Pin configuration

**typedef** **struct**

{

uint8\_t GPIO\_PinNumber;

uint8\_t GPIO\_PinMode;

uint8\_t GPIO\_PinSpeed;

uint8\_t GPIO\_PinPuPdControl;

uint8\_t GPIO\_PinOType;

uint8\_t GPIO\_PinAltFunMode;

}GPIO\_Pin\_Config\_t;

// GPIO Handle Structure

**typedef** **struct**

{ // pin definitions

GPIO\_Reg\_def\_t \*pGPIOx; //this holds base address of GPIO port to which port belongs

GPIO\_Pin\_Config\_t pin\_config;//GPIO pin config setting

}GPIO\_Handle\_t;

// macros for pin numbers

**#define** GPIO\_Pin\_Number\_0 0

**#define** GPIO\_Pin\_Number\_1 1

**#define** GPIO\_Pin\_Number\_2 2

**#define** GPIO\_Pin\_Number\_3 3

**#define** GPIO\_Pin\_Number\_4 4

**#define** GPIO\_Pin\_Number\_5 5

**#define** GPIO\_Pin\_Number\_6 6

**#define** GPIO\_Pin\_Number\_7 7

**#define** GPIO\_Pin\_Number\_8 8

**#define** GPIO\_Pin\_Number\_9 9

**#define** GPIO\_Pin\_Number\_10 10

**#define** GPIO\_Pin\_Number\_11 11

**#define** GPIO\_Pin\_Number\_12 12

**#define** GPIO\_Pin\_Number\_13 13

**#define** GPIO\_Pin\_Number\_14 14

**#define** GPIO\_Pin\_Number\_15 15

// macros for pin modes

**#define** GPIO\_PinMode\_IN 0 // non interrupt modes

**#define** GPIO\_PinMode\_OUT 1

**#define** GPIO\_PinMode\_ALTFN 2

**#define** GPIO\_PinMode\_ANALOG 3

**#define** GPIO\_PinMode\_IT\_FT 4 // falling edge triggered

**#define** GPIO\_PinMode\_IT\_RT 5 // raising edge triggered

**#define** GPIO\_PinMode\_IT\_RFT 6 // falling & raising edge triggered

// macros for pin speed

**#define** GPIO\_Speed\_LOW 0 //low speed

**#define** GPIO\_Speed\_MEDIUM 1 //Medium speed

**#define** GPIO\_Speed\_FAST 2 //High Speed

**#define** GPIO\_Speed\_HIGH 3 //Very High speed

// macros for pin PUPD control

**#define** GPIO\_PinPuPdControl\_PUPD 0 // no pull up pull down

**#define** GPIO\_PinPuPdControl\_PU 1 // pull up

**#define** GPIO\_PinPuPdControl\_PD 2 // pull down

**#define** GPIO\_PinPuPdControl\_Reserved 3 // reserved

// macros for pin OType

**#define** GPIO\_PinOType\_PP 0 //push pull

**#define** GPIO\_PinOType\_OD 1 //open drain

// GPIO driver API'S

//Peripheral clock setup

**void** **GPIO\_PeriClockControl**(GPIO\_Reg\_def\_t \*pGPIOx, uint8\_t EnorDi);

//Init and Deinit

**void** **GPIO\_Init**(GPIO\_Handle\_t \*pGPIOHandle);

**void** **GPIO\_DeInit**(GPIO\_Reg\_def\_t \*pGPIOx);

//Data Read and Write

uint8\_t **GPIO\_ReadFromInputPin**(GPIO\_Reg\_def\_t \*pGPIOx, uint8\_t PinNumber);

uint16\_t **GPIO\_ReadFromInputPort**(GPIO\_Reg\_def\_t \*pGPIOx);

**void** **GPIO\_WriteToOutputPin**(GPIO\_Reg\_def\_t \*pGPIOx, uint8\_t PinNumber, uint8\_t Value);

**void** **GPIO\_WriteToOutputPort**(GPIO\_Reg\_def\_t \*pGPIOx, uint16\_t Value);

**void** **GPIOToggleOutputPin**(GPIO\_Reg\_def\_t \*pGPIOx, uint8\_t PinNumber);

**#endif** /\* INC\_STM32FXX\_GPIO\_DRIVER\_H\_ \*/

## **2.3 Source File**

/\*

\* STM32Fxx\_GPIO\_DRIVER.c

\*

\* Created on: Dec 28, 2020

\* Author: 99003171

\*/

**#include** "STM32FXX\_GPIO\_DRIVER.h"

// GPIO driver API'S

//Peripheral clock setup

**void** **GPIO\_PeriClockControl**(GPIO\_Reg\_def\_t \*pGPIOx, uint8\_t EnorDi)

{

**if**( EnorDi == ENABLE )

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_pclock\_enable();

}

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

{

GPIOB\_pclock\_enable();

}

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

{

GPIOC\_pclock\_enable();

}

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

{

GPIOD\_pclock\_enable();

}

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

{

GPIOE\_pclock\_enable();

}

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

{

GPIOF\_pclock\_enable();

}

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

{

GPIOG\_pclock\_enable();

}

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

{

GPIOH\_pclock\_enable();

}

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

{

GPIOI\_pclock\_enable();

}

}

**else**

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_pclock\_disable();

}

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

{

GPIOB\_pclock\_disable();

}

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

{

GPIOC\_pclock\_disable();

}

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

{

GPIOD\_pclock\_disable();

}

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

{

GPIOE\_pclock\_disable();

}

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

{

GPIOF\_pclock\_disable();

}

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

{

GPIOG\_pclock\_disable();

}

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

{

GPIOH\_pclock\_disable();

}

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

{

GPIOI\_pclock\_disable();

}

}

}

//Initialization and Deinitialization

**void** **GPIO\_Init**(GPIO\_Handle\_t \*pGPIOHandle)

{

//1. configuring the mode

uint32\_t temp=0;

**if**(pGPIOHandle->pin\_config.GPIO\_PinMode <= GPIO\_PinMode\_ANALOG )//non interrupt modes

{

temp= pGPIOHandle->pin\_config.GPIO\_PinMode<<(2\*pGPIOHandle->pin\_config.GPIO\_PinNumber);

pGPIOHandle->pGPIOx->MODER |= temp;

}

//2. configuring the speed

uint32\_t temp1=0;

temp1= pGPIOHandle->pin\_config.GPIO\_PinSpeed<<(2\*pGPIOHandle->pin\_config.GPIO\_PinNumber);

pGPIOHandle->pGPIOx->OSPEEDR |= temp1;

//3. configuring the pu pd control

uint32\_t temp2=0;

temp2= pGPIOHandle->pin\_config.GPIO\_PinPuPdControl<<(2\*pGPIOHandle->pin\_config.GPIO\_PinNumber);

pGPIOHandle->pGPIOx->PUPDR |= temp2;

//4. configuring the output type

uint32\_t temp3=0;

temp3= pGPIOHandle->pin\_config.GPIO\_PinOType<<(pGPIOHandle->pin\_config.GPIO\_PinNumber);

pGPIOHandle->pGPIOx->OTYPER |= temp3;

//Alternate function

**if**(pGPIOHandle->pin\_config.GPIO\_PinMode==GPIO\_PinMode\_ALTFN)

{

uint32\_t temp4,temp5;

temp4=pGPIOHandle->pin\_config.GPIO\_PinNumber/8;

temp5=pGPIOHandle->pin\_config.GPIO\_PinNumber%8;

pGPIOHandle->pGPIOx->AFR[temp4] |= pGPIOHandle->pin\_config.GPIO\_PinAltFunMode<<(4\*temp5);

}

}

**void** **GPIO\_DeInit**(GPIO\_Reg\_def\_t \*pGPIOx)

{

**if**(pGPIOx == GPIOA)

{

GPIOA\_pclock\_disable();

}

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

{

GPIOB\_pclock\_disable();

}

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

{

GPIOC\_pclock\_disable();

}

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

{

GPIOD\_pclock\_disable();

}

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

{

GPIOE\_pclock\_disable();

}

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

{

GPIOF\_pclock\_disable();

}

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

{

GPIOG\_pclock\_disable();

}

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

{

GPIOH\_pclock\_disable();

}

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

{

GPIOI\_pclock\_disable();

}

}

//Data Read and Write

uint8\_t **GPIO\_ReadFromInputPin**(GPIO\_Reg\_def\_t \*pGPIOx, uint8\_t PinNumber)

{

uint8\_t value;

value=(uint8\_t)((pGPIOx->IDR>>PinNumber)\*(0x00000001));

**return** value;

}

uint16\_t **GPIO\_ReadFromInputPort**(GPIO\_Reg\_def\_t \*pGPIOx)

{

uint16\_t value1;

value1=(uint16\_t)(pGPIOx->IDR);

**return** value1;

}

**void** **GPIO\_WriteToOutputPin**(GPIO\_Reg\_def\_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\_Reg\_def\_t \*pGPIOx, uint16\_t Value)

{

pGPIOx->ODR = Value;

}

**void** **GPIOToggleOutputPin**(GPIO\_Reg\_def\_t \*pGPIOx, uint8\_t PinNumber)

{

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

}

# 3. Activity

## **3.1 Screenshot of main logic function**

Graphical user interface, text

Description automatically generated

## **3.2 Arduino Code**

#include<SPI.h>  
volatile boolean received;  
volatile int Slave;  
  
  
void setup()  
{  
  Serial.begin(9600);  
pinMode(MISO, OUTPUT);  
  SPCR |= \_BV(SPE);                       //Turn on SPI in Slave Mode  
  received = false;  
  SPI.attachInterrupt();                  //Interuupt ON is set for SPI commnucation  
  }  
ISR (SPI\_STC\_vect)                        //Inerrrput routine function  
{  
  Slave = SPDR;         // Value received from master  
  received = true;                        //Sets received as True  
}  
void loop()  
{ if(received)                              
  {  
 delay(500);  
 Serial.println(Slave);  
      switch (Slave)  
      {  
        case 0:  
               Serial.println("Human is absent\n");  
               break;  
        case 1:  
               Serial.println("Human is present\n");  
               break;  
        case 2:  
               Serial.println("Sensor value is less than 512\n");  
               break;                
      }  
}  
}

## **3.3 GitHub link**

<https://github.com/99003168/Embedded-C/tree/main/Embedded-C-Project>

# References

[1] <https://youtu.be/B7oKdUvRhQQ>

[2] <https://youtu.be/5aafG5mjZ_Y>

[3] <https://youtu.be/Bsq6P1B8JqI>

[4] <https://youtu.be/2Hm8eEHsgls>