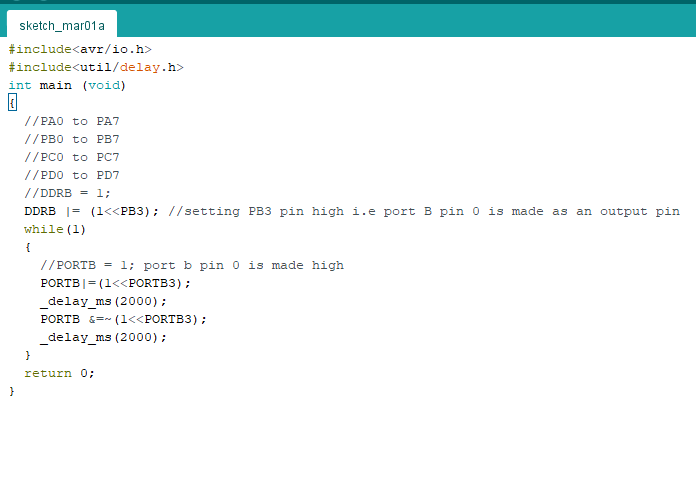
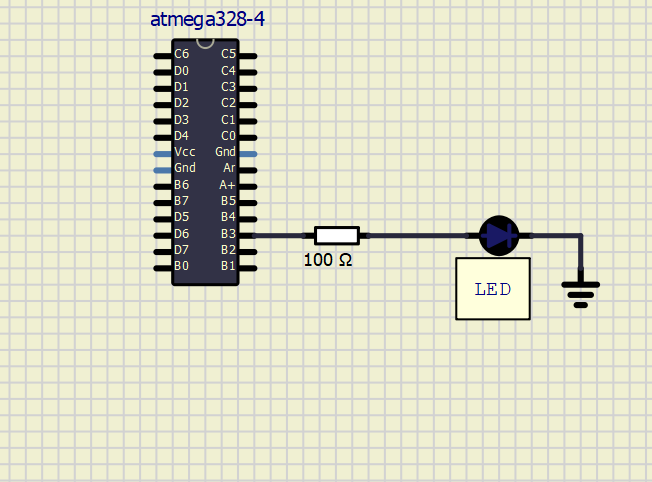
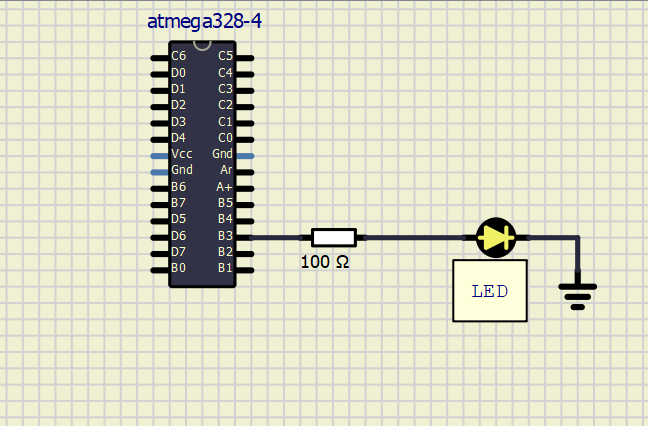
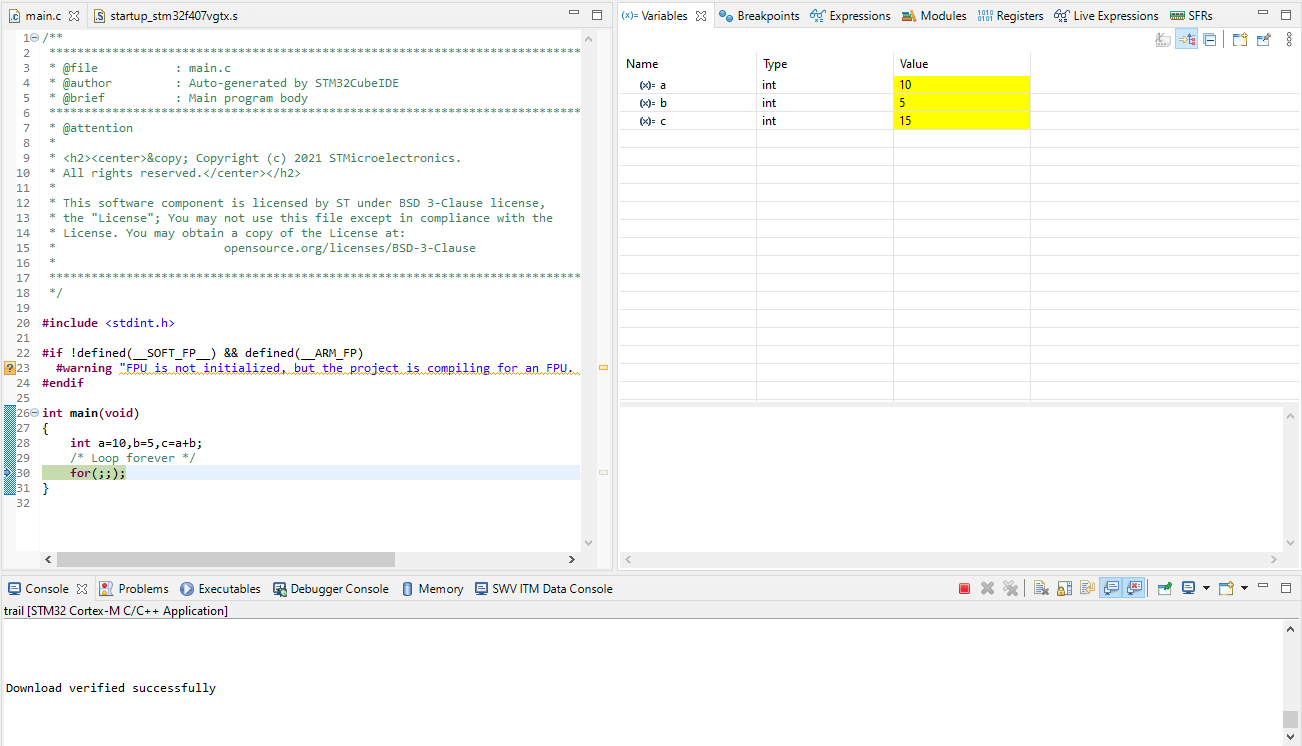
**Arduino beginning:**







**Trial :- to check USB accessibility**



**#include** <stdint.h>

**#if** !defined(\_\_SOFT\_FP\_\_) && defined(\_\_ARM\_FP)

**#warning** "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."

**#endif**

**int** **main**(**void**)

{

**int** a=10,b=5,c=a+b;

/\* Loop forever \*/

for(;;);

}

**Hello world program:**

**#include** <stdint.h>

**#if** !defined(\_\_SOFT\_FP\_\_) && defined(\_\_ARM\_FP)

**#warning** "FPU is not initialized, but the project is compiling for an FPU. Please initialize the FPU before use."

**#endif**

**#include** <stdio.h>

**int** **main**(**void**)

{

**printf**("Hello World\n");

**for**(;;);

}

//////for building and debugging library, basically to display data of microcontroller to the console using SWV library

**#include** <sys/stat.h>

**#include** <stdlib.h>

**#include** <errno.h>

**#include** <stdio.h>

**#include** <signal.h>

**#include** <time.h>

**#include** <sys/time.h>

**#include** <sys/times.h>

/////////////////////////////////////////////////////////////////////////////////////////////////////////

// Implementation of printf like feature using ARM Cortex M3/M4/ ITM functionality

// This function will not work for ARM Cortex M0/M0+

// If you are using Cortex M0, then you can use semihosting feature of openOCD

/////////////////////////////////////////////////////////////////////////////////////////////////////////

//Debug Exception and Monitor Control Register base address

**#define** DEMCR \*((**volatile** uint32\_t\*) 0xE000EDFCU )

/\* ITM register addresses \*/

**#define** ITM\_STIMULUS\_PORT0 \*((**volatile** uint32\_t\*) 0xE0000000 )

**#define** ITM\_TRACE\_EN \*((**volatile** uint32\_t\*) 0xE0000E00 )

**void** **ITM\_SendChar**(uint8\_t ch)

{

//Enable TRCENA

DEMCR |= ( 1 << 24);

//enable stimulus port 0

ITM\_TRACE\_EN |= ( 1 << 0);

// read FIFO status in bit [0]:

**while**(!(ITM\_STIMULUS\_PORT0 & 1));

//Write to ITM stimulus port0

ITM\_STIMULUS\_PORT0 = ch;

}

/\* Variables \*/

//#undef errno

**extern** **int** errno;

**extern** **int** **\_\_io\_putchar**(**int** ch) **\_\_attribute\_\_**((weak));

**extern** **int** **\_\_io\_getchar**(**void**) **\_\_attribute\_\_**((weak));

**register** **char** \* stack\_ptr **asm**("sp");

**char** \*\_\_env[1] = { 0 };

**char** \*\*environ = \_\_env;

/\* Functions \*/

**void** **initialise\_monitor\_handles**()

{

}

**int** **\_getpid**(**void**)

{

**return** 1;

}

**int** **\_kill**(**int** pid, **int** sig)

{

errno = EINVAL;

**return** -1;

}

**void** **\_exit** (**int** status)

{

\_kill(status, -1);

**while** (1) {} /\* Make sure we hang here \*/

}

**\_\_attribute\_\_**((weak)) **int** **\_read**(**int** file, **char** \*ptr, **int** len)

{

**int** DataIdx;

**for** (DataIdx = 0; DataIdx < len; DataIdx++)

{

\*ptr++ = \_\_io\_getchar();

}

**return** len;

}

**\_\_attribute\_\_**((weak)) **int** **\_write**(**int** file, **char** \*ptr, **int** len)

{

**int** DataIdx;

**for** (DataIdx = 0; DataIdx < len; DataIdx++)

{

//\_\_io\_putchar(\*ptr++);

ITM\_SendChar(\*ptr++);

}

**return** len;

}

**int** **\_close**(**int** file)

{

**return** -1;

}

**int** **\_fstat**(**int** file, **struct** stat \*st)

{

st->st\_mode = S\_IFCHR;

**return** 0;

}

**int** **\_isatty**(**int** file)

{

**return** 1;

}

**int** **\_lseek**(**int** file, **int** ptr, **int** dir)

{

**return** 0;

}

**int** **\_open**(**char** \*path, **int** flags, ...)

{

/\* Pretend like we always fail \*/

**return** -1;

}

**int** **\_wait**(**int** \*status)

{

errno = ECHILD;

**return** -1;

}

**int** **\_unlink**(**char** \*name)

{

errno = ENOENT;

**return** -1;

}

**int** **\_times**(**struct** tms \*buf)

{

**return** -1;

}

**int** **\_stat**(**char** \*file, **struct** stat \*st)

{

st->st\_mode = S\_IFCHR;

**return** 0;

}

**int** **\_link**(**char** \*old, **char** \*new)

{

errno = EMLINK;

**return** -1;

}

**int** **\_fork**(**void**)

{

errno = EAGAIN;

**return** -1;

}

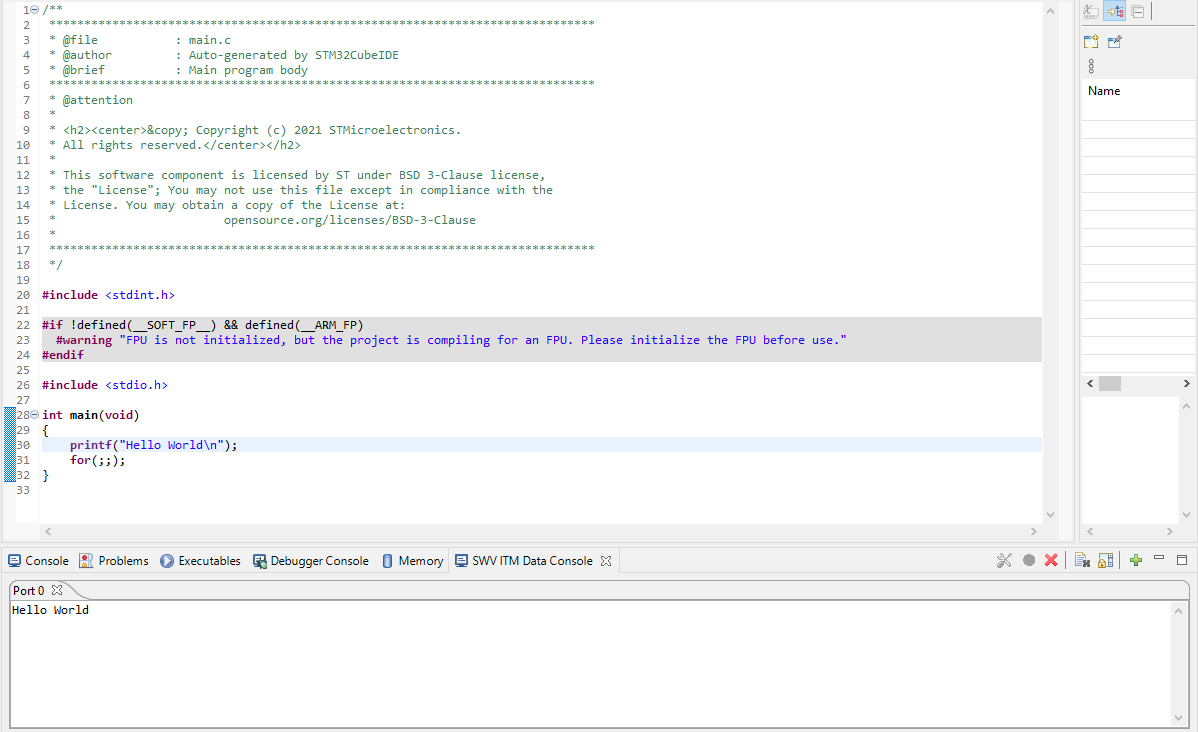
**int** **\_execve**(**char** \*name, **char** \*\*argv, **char** \*\*env)

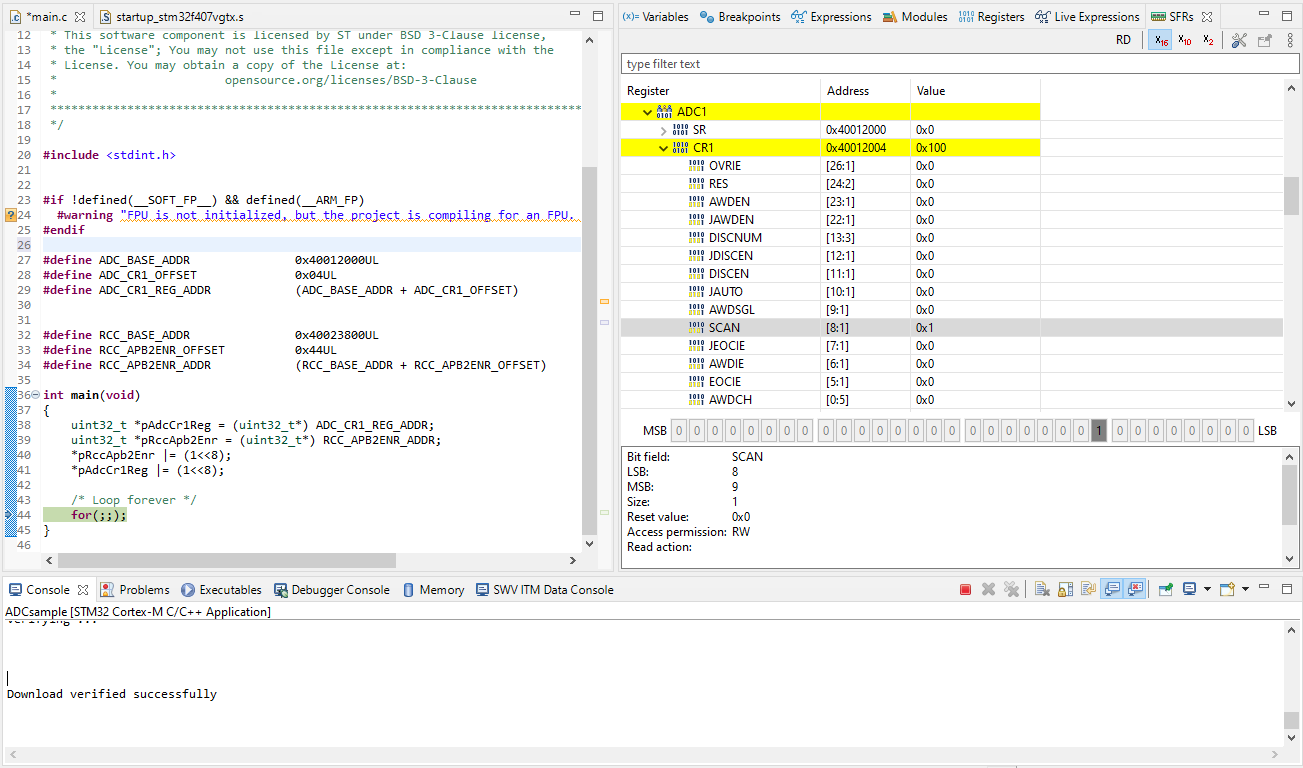
{

errno = ENOMEM;

**return** -1;

}





**Driver file:**

stm32f4xx.h

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

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

**#include**<stdint.h>

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

/\* Defining macros for the various memories \*/

**#define** FLASH\_ADDR 0X08000000U

**#define** SRAM1\_ADDR 0X20000000U

**#define** SRAM2\_ADDR 0X2001C000U

**#define** ROM\_ADDR 0X1FFF0000

**#define** SRAM\_ADDR SRAM1\_ADDR

/\* Defining macros for buses \*/

**#define** AHB1\_BASE\_ADDR 0X40020000U

**#define** AHB2\_BASE\_ADDR 0X50000000U

**#define** APB1\_BASE\_ADDR 0X40000000U

**#define** APB2\_BASE\_ADDR 0X40010000U

**#define** PER1\_BASE\_ADDR APB1\_BASE\_ADDR

/\* Defining macros for peripherals hanging onto AHB1 Bus\*/

**#define** GPIOA\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X0000U))

**#define** GPIOB\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X0400U))

**#define** GPIOC\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X0800U))

**#define** GPIOD\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X0C00U))

**#define** GPIOE\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X1000U))

**#define** GPIOF\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X1400U))

**#define** GPIOG\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X1800U))

**#define** GPIOH\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X1C00U))

**#define** GPIOI\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X2000U))

**#define** RCC\_BASE\_ADDR (AHB1\_BASE\_ADDR + (0X3800U))

/\* Defining macros for peripherals hanging onto AHB2 Bus - None\*/

/\* Defining macros for peripherals hanging onto APB1 Bus\*/

**#define** SPI2\_BASE\_ADDR (APB1\_BASE\_ADDR + (0X3800U))

**#define** SPI3\_BASE\_ADDR (APB1\_BASE\_ADDR + (0X3C00U))

**#define** UART4\_BASE\_ADDR (APB1\_BASE\_ADDR + (0X4C00U))

**#define** UART5\_BASE\_ADDR (APB1\_BASE\_ADDR + (0X5000U))

**#define** I2C1\_BASE\_ADDR (APB1\_BASE\_ADDR + (0X5400U))

**#define** I2C2\_BASE\_ADDR (APB1\_BASE\_ADDR + (0X5800U))

**#define** I2C3\_BASE\_ADDR (APB1\_BASE\_ADDR + (0X5C00U))

**#define** USART2\_BASE\_ADDR (APB1\_BASE\_ADDR + (0X4400U))

**#define** USART3\_BASE\_ADDR (APB1\_BASE\_ADDR + (0X4800U))

/\* Defining macros for peripherals hanging onto APB2 Bus\*/

**#define** SPI1\_BASE\_ADDR (APB2\_BASE\_ADDR + (0X3000U))

**#define** USART1\_BASE\_ADDR (APB2\_BASE\_ADDR + (0X1000U))

**#define** USART6\_BASE\_ADDR (APB2\_BASE\_ADDR + (0X1400U))

/\* 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[0] - AFRL, AFR[1] - AFRH

}GPIO\_Reg\_def\_t;

/\* RCC Peripheral Registers \*/

**typedef** **struct**

{

\_\_vo uint32\_t RCC\_CR;

\_\_vo uint32\_t RCC\_PLLCFGR;

\_\_vo uint32\_t RCC\_CFGR;

\_\_vo uint32\_t RCC\_CIR;

\_\_vo uint32\_t RCC\_AHB1RSTR;

\_\_vo uint32\_t RCC\_AHB2RSTR;

\_\_vo uint32\_t RCC\_AHB3RSTR;

uint32\_t RESERVED0;

\_\_vo uint32\_t RCC\_APB1RSTR;

\_\_vo uint32\_t RCC\_APB2RSTR;

uint32\_t RESERVED1[2];

\_\_vo uint32\_t RCC\_AHB1ENR;

\_\_vo uint32\_t RCC\_AHB2ENR;

\_\_vo uint32\_t RCC\_AHB3ENR;

uint32\_t RESERVED2;

\_\_vo uint32\_t RCC\_APB1ENR;

\_\_vo uint32\_t RCC\_APB2ENR;

uint32\_t RESERVED3[2];

\_\_vo uint32\_t RCC\_AHB1LPENR;

\_\_vo uint32\_t RCC\_AHB2LPENR;

\_\_vo uint32\_t RCC\_AHB3LPENR;

uint32\_t RESERVED4;

\_\_vo uint32\_t RCC\_APB1LPENR;

\_\_vo uint32\_t RCC\_APB2LPENR;

uint32\_t RESERVED5[2];

\_\_vo uint32\_t RCC\_BDCR;

\_\_vo uint32\_t RCC\_CSR;

uint32\_t RESERVED6[2];

\_\_vo uint32\_t RCC\_SSCGR;

\_\_vo uint32\_t RCC\_PLLI2SCFGR;

\_\_vo uint32\_t RCC\_PLLSAICFGR;

\_\_vo uint32\_t RCC\_DCKCFGR;

}RCC\_Reg\_def\_t;

/\* GPIO Peripheral Sections \*/

**#define** GPIOA ((GPIO\_Reg\_def\_t\*)GPIOA\_BASE\_ADDR)

**#define** GPIOB ((GPIO\_Reg\_def\_t\*)GPIOB\_BASE\_ADDR)

**#define** GPIOC ((GPIO\_Reg\_def\_t\*)GPIOC\_BASE\_ADDR)

**#define** GPIOD ((GPIO\_Reg\_def\_t\*)GPIOD\_BASE\_ADDR)

**#define** GPIOE ((GPIO\_Reg\_def\_t\*)GPIOE\_BASE\_ADDR)

**#define** GPIOF ((GPIO\_Reg\_def\_t\*)GPIOF\_BASE\_ADDR)

**#define** GPIOG ((GPIO\_Reg\_def\_t\*)GPIOG\_BASE\_ADDR)

**#define** GPIOH ((GPIO\_Reg\_def\_t\*)GPIOH\_BASE\_ADDR)

**#define** GPIOI ((GPIO\_Reg\_def\_t\*)GPIOI\_BASE\_ADDR)

**#define** RCC ((RCC\_Reg\_def\_t\*)RCC\_BASE\_ADDR)

/\* GPIO Clock Enable \*/

**#define** GPIOA\_PCLOCK\_EN() (RCC->RCC\_AHB1ENR |= (1<<0))

**#define** GPIOB\_PCLOCK\_EN() (RCC->RCC\_AHB1ENR |= (1<<1))

**#define** GPIOC\_PCLOCK\_EN() (RCC->RCC\_AHB1ENR |= (1<<2))

**#define** GPIOD\_PCLOCK\_EN() (RCC->RCC\_AHB1ENR |= (1<<3))

**#define** GPIOE\_PCLOCK\_EN() (RCC->RCC\_AHB1ENR |= (1<<4))

**#define** GPIOF\_PCLOCK\_EN() (RCC->RCC\_AHB1ENR |= (1<<5))

**#define** GPIOG\_PCLOCK\_EN() (RCC->RCC\_AHB1ENR |= (1<<6))

**#define** GPIOH\_PCLOCK\_EN() (RCC->RCC\_AHB1ENR |= (1<<7))

**#define** GPIOI\_PCLOCK\_EN() (RCC->RCC\_AHB1ENR |= (1<<8))

/\* GPIO Peripheral Clock Disable Macros \*/

**#define** GPIOA\_PCLOCK\_DI() {(RCC->RCC\_AHB1RSTR&=~(1<<0));}

**#define** GPIOB\_PCLOCK\_DI() {(RCC->RCC\_AHB1RSTR&=~(1<<1));}

**#define** GPIOC\_PCLOCK\_DI() {(RCC->RCC\_AHB1RSTR&=~(1<<2));}

**#define** GPIOD\_PCLOCK\_DI() {(RCC->RCC\_AHB1RSTR&=~(1<<3));}

**#define** GPIOE\_PCLOCK\_DI() {(RCC->RCC\_AHB1RSTR&=~(1<<4));}

**#define** GPIOF\_PCLOCK\_DI() {(RCC->RCC\_AHB1RSTR&=~(1<<5));}

**#define** GPIOG\_PCLOCK\_DI() {(RCC->RCC\_AHB1RSTR&=~(1<<6));}

**#define** GPIOH\_PCLOCK\_DI() {(RCC->RCC\_AHB1RSTR&=~(1<<7));}

**#define** GPIOI\_PCLOCK\_DI() {(RCC->RCC\_AHB1RSTR&=~(1<<8));}

/\*Other Macro Definitions \*/

**#define** ENABLE 1

**#define** DISABLE 0

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

**#define** GPIO\_Pin\_Reset DISABLE

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

stm32f4xx\_gpio\_driver.h

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

**#define** INC\_STM32F4XX\_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\_PinOPType;

uint8\_t GPIO\_PinAltFunMode;

}GPIO\_PIN\_CONFIG\_t;

/\* INC\_STM32407XX\_GPIO\_DRIVER\*/

/\* GPIO Handle Structure \*/

**typedef** **struct**

{

GPIO\_Reg\_def\_t \*pGPIOx;

GPIO\_PIN\_CONFIG\_t PIN\_CONGFIG;

}GPIO\_HANDLE\_t;

/\* GPIO 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

/\* GPIO Pin Modes \*/

**#define** GPIO\_Pin\_Mode\_In 0

**#define** GPIO\_Pin\_Mode\_Out 1

**#define** GPIO\_Pin\_Mode\_Altfn 2

**#define** GPIO\_Pin\_Mode\_Analog 3

/\* @GPIO\_Pin\_Speed

\* GPIO pin possible output speeds

\*/

**#define** GPIO\_Pin\_Speed\_Low 0

**#define** GPIO\_Pin\_Speed\_Medium 1

**#define** GPIO\_Pin\_Speed\_Fast 2

**#define** GPIO\_Pin\_Speed\_High 3

/\*

\* GPIO pin pull up and pull down configuration macros

\*/

**#define** GPIO\_Pin\_No\_PUPD 0

**#define** GPIO\_Pin\_PU 1

**#define** GPIO\_Pin\_PD 2

**#define** GPIO\_Pin\_PP 0

**#define** GPIO\_Pin\_OD 1

/\*

#define GPIO\_Pin\_AltFunMode\_0 0

#define GPIO\_Pin\_AltFunMode\_1 1

#define GPIO\_Pin\_AltFunMode\_2 2

#define GPIO\_Pin\_AltFunMode\_3 3

#define GPIO\_Pin\_AltFunMode\_4 4

#define GPIO\_Pin\_AltFunMode\_5 5

#define GPIO\_Pin\_AltFunMode\_6 6

#define GPIO\_Pin\_AltFunMode\_7 7

#define GPIO\_Pin\_AltFunMode\_8 8

#define GPIO\_Pin\_AltFunMode\_9 9

#define GPIO\_Pin\_AltFunMode\_10 10

#define GPIO\_Pin\_AltFunMode\_11 11

#define GPIO\_Pin\_AltFunMode\_12 12

#define GPIO\_Pin\_AltFunMode\_13 13

#define GPIO\_Pin\_AltFunMode\_14 14

#define GPIO\_Pin\_AltFunMode\_15 15

\*/

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

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

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

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

/\*void GPIO\_IRQConfig(uint8\_t IRQNumber,uint8\_t IRQPriority,uint8\_t EnorDi);

void GPIO\_IRQHandling(uint8\_t PinNumber);

\*/

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

**stm32f4xx\_gpio\_driver.c**

**#include**"STM32F4XX\_GPIO\_DRIVER.h"

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

{

**if**(EnorDi == ENABLE)

{

**if**(pGPIOx==GPIOA)

{

GPIOA\_PCLOCK\_EN();

}

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

{

GPIOB\_PCLOCK\_EN();

}

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

{

GPIOC\_PCLOCK\_EN();

}

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

{

GPIOD\_PCLOCK\_EN();

}

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

{

GPIOE\_PCLOCK\_EN();

}

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

{

GPIOF\_PCLOCK\_EN();

}

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

{

GPIOG\_PCLOCK\_EN();

}

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

{

GPIOH\_PCLOCK\_EN();

}

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

{

GPIOI\_PCLOCK\_EN();

}

}

**else**

{

**if**(pGPIOx==GPIOA)

{

GPIOA\_PCLOCK\_DI();

}

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

{

GPIOB\_PCLOCK\_DI();

}

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

{

GPIOC\_PCLOCK\_DI();

}

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

{

GPIOD\_PCLOCK\_DI();

}

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

{

GPIOE\_PCLOCK\_DI();

}

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

{

GPIOF\_PCLOCK\_DI();

}

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

{

GPIOG\_PCLOCK\_DI();

}

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

{

GPIOH\_PCLOCK\_DI();

}

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

{

GPIOI\_PCLOCK\_DI();

}

}

}

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

{

//1.Configuring the mode

uint32\_t temp=0;

**if**((pGPIOHandle->PIN\_CONGFIG.GPIO\_PinMode)<=GPIO\_Pin\_Mode\_Analog)

{

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

pGPIOHandle->pGPIOx->MODER |= temp;

}

//2.Configuring the speed

temp=0;

temp = pGPIOHandle->PIN\_CONGFIG.GPIO\_PinSpeed << (2\*pGPIOHandle->PIN\_CONGFIG.GPIO\_PinNumber);

pGPIOHandle->pGPIOx->OSPEEDR |= temp;

//3.Configuring output type

temp=0;

temp = pGPIOHandle->PIN\_CONGFIG.GPIO\_PinOPType << (pGPIOHandle->PIN\_CONGFIG.GPIO\_PinNumber);

pGPIOHandle->pGPIOx->OTYPER |= temp;

//4.Configuring pin PuPd Control

temp=0;

temp = pGPIOHandle->PIN\_CONGFIG.GPIO\_PinPuPdControl<< (2\*pGPIOHandle->PIN\_CONGFIG.GPIO\_PinNumber);

pGPIOHandle->pGPIOx->PUPDR |= temp;

//5. Alternate Functions

**if**(pGPIOHandle->PIN\_CONGFIG.GPIO\_PinMode==GPIO\_Pin\_Mode\_Altfn)

{

uint32\_t temp1=0;

uint32\_t temp2=0;

temp1 = pGPIOHandle->PIN\_CONGFIG.GPIO\_PinNumber/8;

temp2 = pGPIOHandle->PIN\_CONGFIG.GPIO\_PinNumber%8;

pGPIOHandle->pGPIOx->AFR[temp1]|=pGPIOHandle->PIN\_CONGFIG.GPIO\_PinAltFunMode<<(4\*temp2);

}

}

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

{

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

}

led\_blink.c

**#include** "STM32F4XX.h"

**#include** "stm32f4xx\_gpio\_driver.h"

**#include** "STM32F4XX.h"

**void** **delay**(**void**)

{

**for**(uint32\_t i=0;i<500000;i++)

{

;

}

}

**int** **main**()

{

GPIO\_HANDLE\_t GPIO\_LED;

GPIO\_LED.pGPIOx = GPIOD;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinNumber = GPIO\_Pin\_Number\_15;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinMode = GPIO\_Pin\_Mode\_Out;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinSpeed = GPIO\_Pin\_Speed\_Fast;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinOPType = GPIO\_Pin\_PP;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinPuPdControl=GPIO\_Pin\_PU;

GPIO\_PeriClockControl(GPIOD, ENABLE);

GPIO\_Init(&GPIO\_LED);

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinNumber = GPIO\_Pin\_Number\_12;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinMode = GPIO\_Pin\_Mode\_Out;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinSpeed = GPIO\_Pin\_Speed\_Fast;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinOPType = GPIO\_Pin\_PP;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinPuPdControl=GPIO\_Pin\_PU;

GPIO\_PeriClockControl(GPIOD, ENABLE);

GPIO\_Init(&GPIO\_LED);

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinNumber = GPIO\_Pin\_Number\_13;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinMode = GPIO\_Pin\_Mode\_Out;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinSpeed = GPIO\_Pin\_Speed\_Fast;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinOPType = GPIO\_Pin\_PP;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinPuPdControl=GPIO\_Pin\_PU;

GPIO\_PeriClockControl(GPIOD, ENABLE);

GPIO\_Init(&GPIO\_LED);

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinNumber = GPIO\_Pin\_Number\_14;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinMode = GPIO\_Pin\_Mode\_Out;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinSpeed = GPIO\_Pin\_Speed\_Fast;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinOPType = GPIO\_Pin\_PP;

GPIO\_LED.PIN\_CONGFIG.GPIO\_PinPuPdControl=GPIO\_Pin\_PU;

GPIO\_PeriClockControl(GPIOD, ENABLE);

GPIO\_Init(&GPIO\_LED);

**while**(1)

{

GPIO\_ToggleOutputPin(GPIOD, 13);

delay();

GPIO\_ToggleOutputPin(GPIOD, 12);

delay();

GPIO\_ToggleOutputPin(GPIOD, 14);

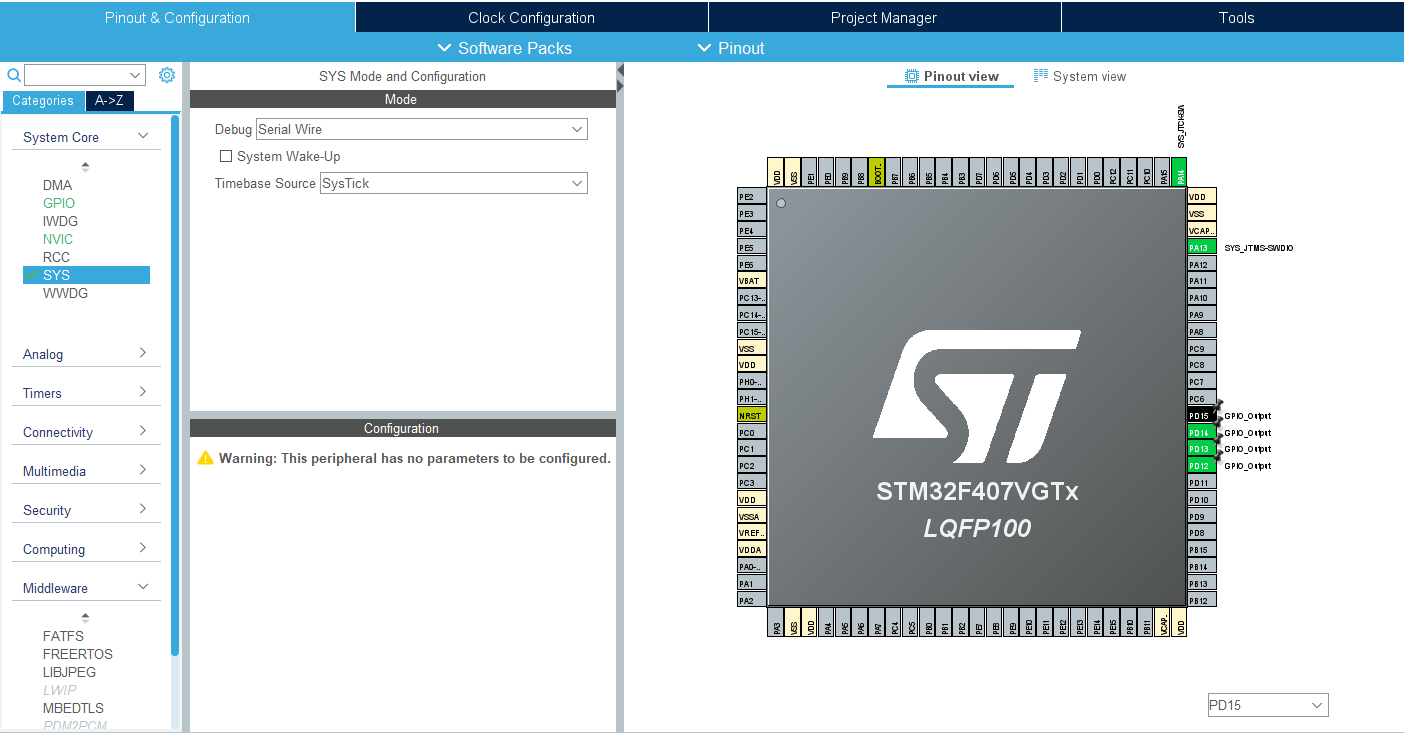
delay();

GPIO\_ToggleOutputPin(GPIOD, 15);

delay();

}

}



Main.c :- Automated generated code

//added following under while loop

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_12 , GPIO\_PIN\_SET);  
HAL\_Delay(2000);  
HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_12 , GPIO\_PIN\_RESET);  
HAL\_Delay(2000);

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under Ultimate Liberty license

\* SLA0044, the "License"; You may not use this file except in compliance with

\* the License. You may obtain a copy of the License at:

\* www.st.com/SLA0044

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

**int** **main**(**void**)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

/\* USER CODE BEGIN 2 \*/

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_12 , *GPIO\_PIN\_SET*);

HAL\_Delay(2000);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_12 , *GPIO\_PIN\_RESET*);

HAL\_Delay(2000);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13 , *GPIO\_PIN\_SET*);

HAL\_Delay(2000);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_13 , *GPIO\_PIN\_RESET*);

HAL\_Delay(2000);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14 , *GPIO\_PIN\_SET*);

HAL\_Delay(2000);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_14 , *GPIO\_PIN\_RESET*);

HAL\_Delay(2000);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_15 , *GPIO\_PIN\_SET*);

HAL\_Delay(2000);

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_15 , *GPIO\_PIN\_RESET*);

HAL\_Delay(2000);

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

**void** **SystemClock\_Config**(**void**)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PLLM = 8;

RCC\_OscInitStruct.PLL.PLLN = 50;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 7;

**if** (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != *HAL\_OK*)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

**if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_1) != *HAL\_OK*)

{

Error\_Handler();

}

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

**static** **void** **MX\_GPIO\_Init**(**void**)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_12|GPIO\_PIN\_13|GPIO\_PIN\_14|GPIO\_PIN\_15, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pins : PD12 PD13 PD14 PD15 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_12|GPIO\_PIN\_13|GPIO\_PIN\_14|GPIO\_PIN\_15;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

**void** **Error\_Handler**(**void**)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

**while** (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

**#ifdef** USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

**void** assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

**#endif** /\* USE\_FULL\_ASSERT \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/