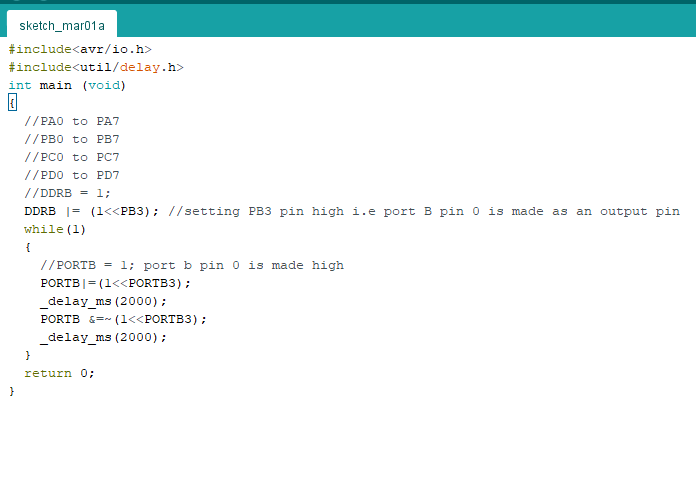
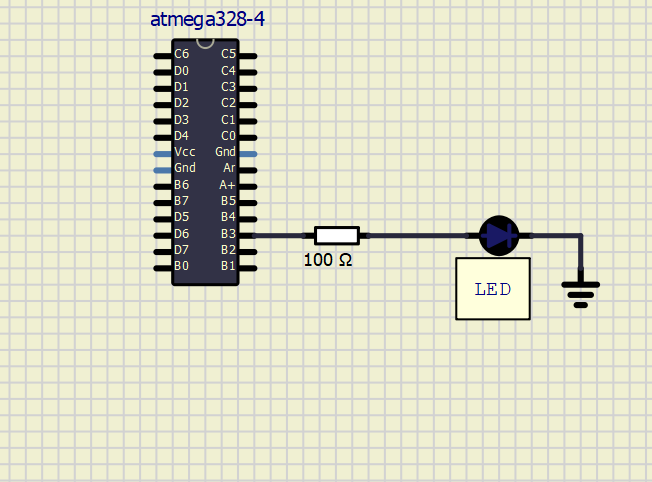
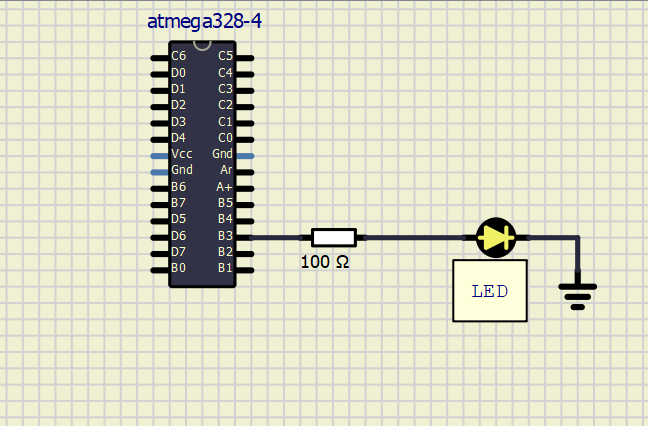
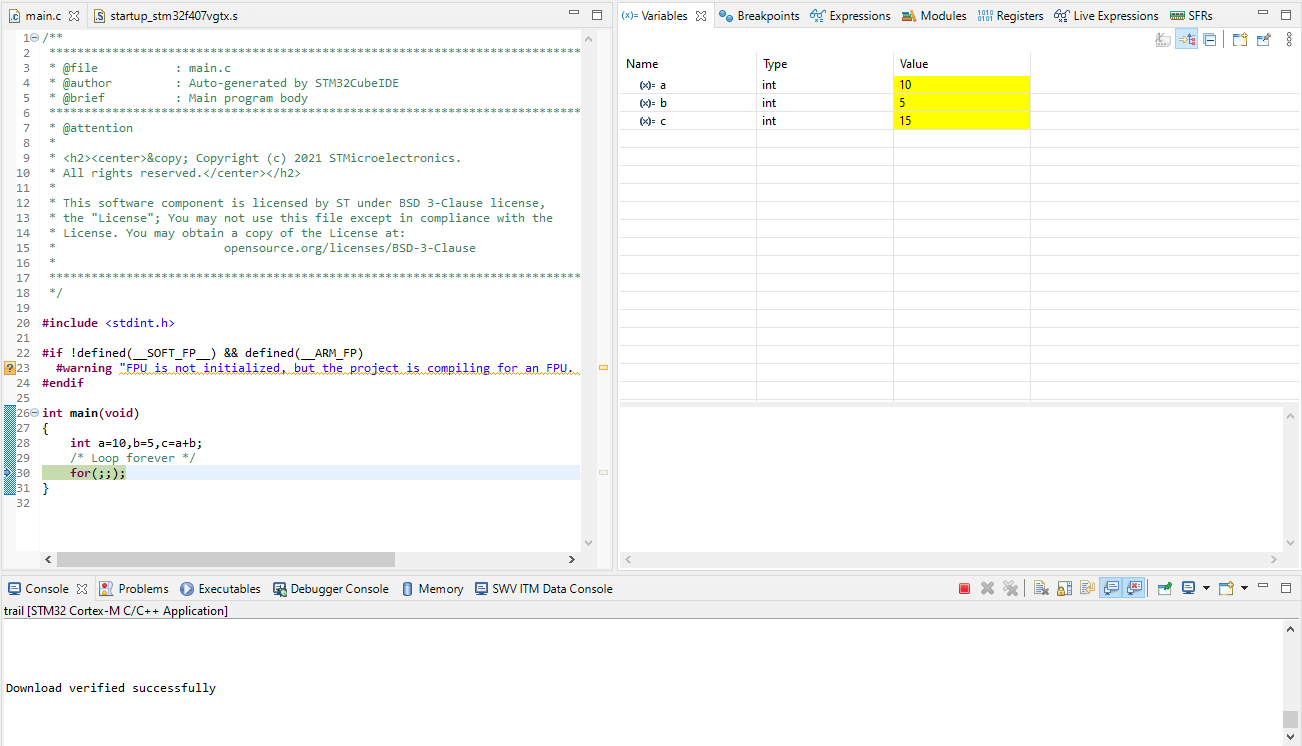
**Arduino beginning:**







**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;

}

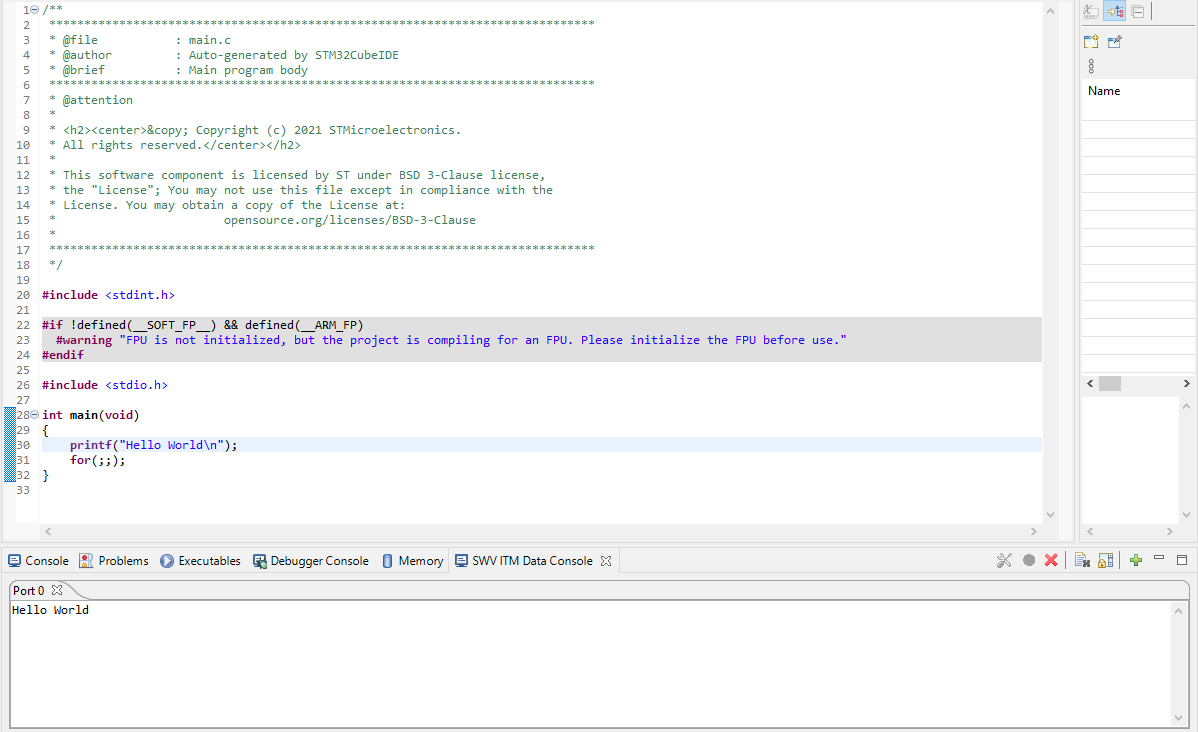
**int** **\_execve**(**char** \*name, **char** \*\*argv, **char** \*\*env)

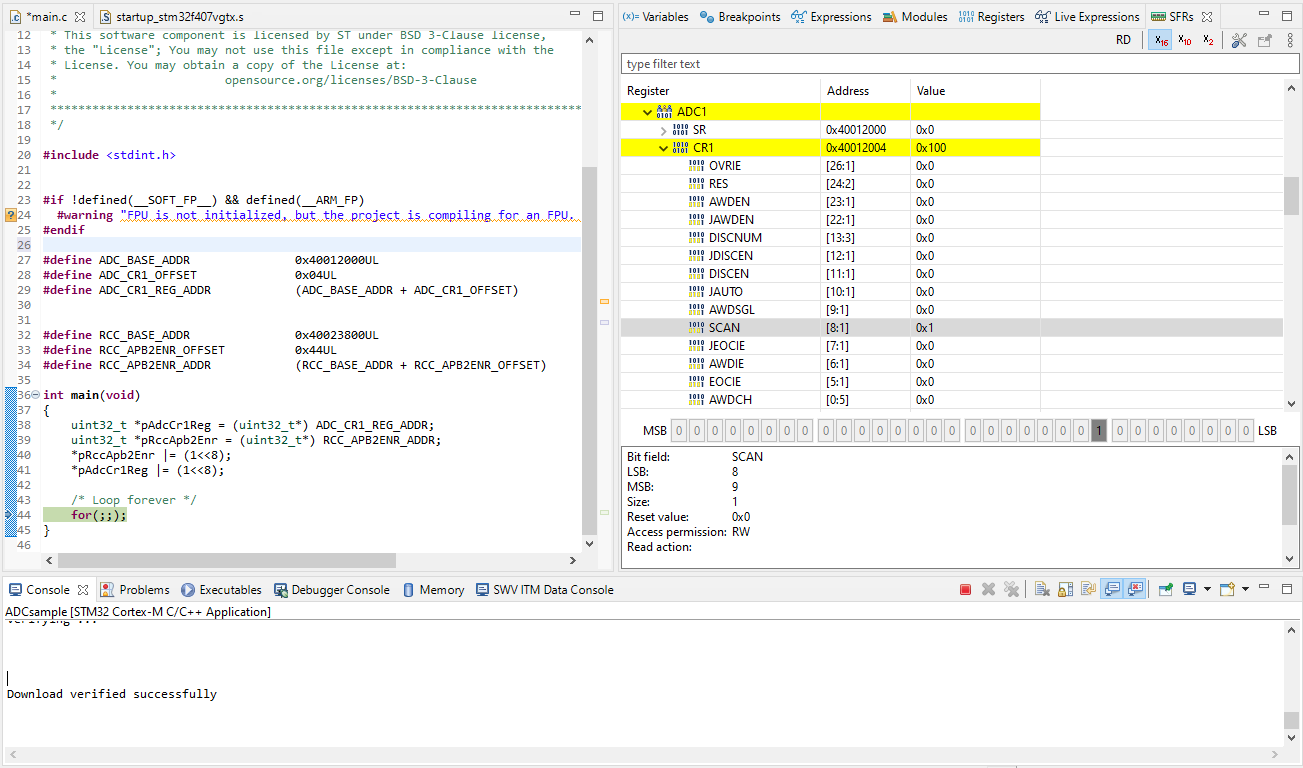
{

errno = ENOMEM;

**return** -1;

}





**#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**

**#define** ADC\_BASE\_ADDR 0x40012000UL

**#define** ADC\_CR1\_OFFSET 0x04UL

**#define** ADC\_CR1\_REG\_ADDR (ADC\_BASE\_ADDR + ADC\_CR1\_OFFSET)

**#define** RCC\_BASE\_ADDR 0x40023800UL

**#define** RCC\_APB2ENR\_OFFSET 0x44UL

**#define** RCC\_APB2ENR\_ADDR (RCC\_BASE\_ADDR + RCC\_APB2ENR\_OFFSET)

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

{

uint32\_t \*pAdcCr1Reg = (uint32\_t\*) ADC\_CR1\_REG\_ADDR;

uint32\_t \*pRccApb2Enr = (uint32\_t\*) RCC\_APB2ENR\_ADDR;

\*pRccApb2Enr |= (1<<8);

\*pAdcCr1Reg |= (1<<8);

/\* Loop forever \*/

for(;;);

}

**#include** <stdint.h>

// this is MCU specific header file

//general macros

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

**#define** RCC (RCC\_RegDef\_t\*) ///0x40023800

//macros for different memories

**#define** SRAM1 0x20000000U

**#define** SRAM2 0x2001C000U

**#define** FLASH 0x08000000U //(Uint32\_t) 0x08000000

**#define** ROM 0x1FFF0000U

//macros for bus system

**#define** BUS\_BASE\_ADDR 0x40000000U ///APB1

**#define** APB1\_BASE\_ADDR 0x40000000U

**#define** APB2\_BASE\_ADDR 0x40010000U

**#define** AHB1\_BASE\_ADDR 0x40020000U

**#define** AHB2\_BASE\_ADDR 0x50000000U

**#define** AHB3\_BASE\_ADDR 0xA0000000U

//Macros for GPIO

**#define** GPIOA\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x0000) //0x40020000

**#define** GPIOB\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x0400) //0400 is offset address

**#define** GPIOC\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x0800)

**#define** GPIOD\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x0C00)

**#define** GPIOE\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x1000)

**#define** GPIOF\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x1400)

**#define** GPIOG\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x1800)

**#define** GPIOH\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x1C00)

**#define** GPIOI\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x2000)

**#define** GPIOJ\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x2400)

**#define** GPIOK\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x2800)

**#define** CRC\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x3000)

**#define** RCC\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x3800)

**#define** FLASH\_INT\_REG\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x3C00)

**#define** BKPSRAM\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x4000)

**#define** DMA1\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x6000)

**#define** DMA2\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x6400)

**#define** ETHERNET\_MAC\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x8000)

**#define** DMA2D\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0xB000)

**#define** USB\_OTG\_HS\_BASE\_ADDR (AHB1\_BASE\_ADDR + 0x0000)

//MACROS FOR PHERIPHERALS HANGING ONTO APB1 BUS

//APB1\_BASE\_ADDR 0x40000000U

**#define** TIM2\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x0000)

**#define** TIM3\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x0400)

**#define** TIM4\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x0800)

**#define** TIM5\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x0C00)

**#define** TIM6\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x1000)

**#define** TIM7\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x1400)

**#define** TIM12\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x1800)

**#define** TIM13\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x1C00)

**#define** TIM14\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x2000)

**#define** DAC\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x7400)

**#define** PWR\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x7000)

**#define** CAN2\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x6800)

**#define** CAN1\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x6400)

**#define** I2C3\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x5C00)

**#define** I2C2\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x5800)

**#define** I2C1\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x5400)

**#define** UART5\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x5000)

**#define** UART4\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x4C00)

**#define** USART3\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x4800)

**#define** USART2\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x4400)

**#define** I2S3EXT\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x4000)

**#define** RTC\_BKP\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x2800)

**#define** WWDG\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x2C00)

**#define** IWDG\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x3000)

**#define** I2S2EXT\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x3400)

**#define** SPI2\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x3800)

**#define** SPI3\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x3C00)

**#define** UART8\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x7C00)

**#define** UART7\_BASE\_ADDR (APB1\_BASE\_ADDR + 0x7800)

//MACROS FOR PHERIPHERALS HANGING ONTO APB2

//APB2\_BASE\_ADDR 0x40010000U

**#define** TIM1\_BASE\_ADDR (APB2\_BASE\_ADDR+0x0000)

**#define** TIM8\_BASE\_ADDR (APB2\_BASE\_ADDR+0x0400)

**#define** USART1\_BASE\_ADDR (APB2\_BASE\_ADDR+1000)

**#define** USART6\_BASE\_ADDR (APB2\_BASE\_ADDR+0x1400)

**#define** ADC1\_ADC2\_ADC3\_BASE\_ADDR (APB2\_BASE\_ADDR+0x2000)

**#define** SDIO\_BASE\_ADDR (APB2\_BASE\_ADDR+0x2C00)

**#define** SPI1\_BASE\_ADDR (APB2\_BASE\_ADDR+0x3000)

**#define** SPI4\_BASE\_ADDR (APB2\_BASE\_ADDR+0x 3400)

**#define** SYSCFG\_BASE\_ADDR (APB2\_BASE\_ADDR+0x3800)

**#define** EXT1\_BASE\_ADDR (APB2\_BASE\_ADDR + 0x3C00)

**#define** TIM9\_BASE\_ADDR (APB2\_BASE\_ADDR + 0x4000)

**#define** TIM10\_BASE\_ADDR (APB2\_BASE\_ADDR + 0x4400)

**#define** TIM11\_BASE\_ADDR (APB2\_BASE\_ADDR + 0x4800)

**#define** SPI5\_BASE\_ADDR (APB2\_BASE\_ADDR + 0x5000)

**#define** SPI6\_BASE\_ADDR (APB2\_BASE\_ADDR + 0x5400)

**#define** SAI1\_BASE\_ADDR (APB2\_BASE\_ADDR + 0x5800)

**#define** LCD\_TFT\_BASE\_ADDR (APB2\_BASE\_ADDR + 0x6800)

//GPIO REGISTER DEFINITIONS

**typedef** **struct**

{

\_\_vo uint32\_t MODER; //+00 offset

\_\_vo uint32\_t OTYPER; //+04 offset [because hexadecimal (increment by 4)]

\_\_vo uint32\_t OSPEEDR; //+08 offset

\_\_vo uint32\_t PUPDR; //+0C offset

\_\_vo uint32\_t IDR; //+10 offset

\_\_vo uint32\_t ODR; //+14 offset

\_\_vo uint32\_t BSRR; //+18 offset

\_\_vo uint32\_t LCKR; //+1C offset

\_\_vo uint32\_t AFR[2]; //+20 offset [AFRH and AFRL]

} GPIO\_RegDef\_t;

//define pointer for structure

//then bring pointer and structure together

GPIO\_RegDef\_t \*pGPIOA = (GPIO\_RegDef\_t\*)0x40020000; //type casting

//RCC REGISTER DEFINITIONS

**typedef** **struct**

{

\_\_vo uint32\_t CR; //+00 offset

\_\_vo uint32\_t PLLCFGR; //+04 offset [because hexadecimal (increment by 4)]

\_\_vo uint32\_t CFGR; //+08 offset

\_\_vo uint32\_t CIR; //+0C offset

\_\_vo uint32\_t AHB1RSTR[3]; //+10 offset

//\_\_vo uint32\_t AHB2RSTR; //+14 offset

//\_\_vo uint32\_t AHB3RSTR; //+18 offset

uint32\_t RESERVED0;

\_\_vo uint32\_t APB1RSTR[2]; //+20 offset

//\_\_vo uint32\_t APB2RSTR; //+24 offset

uint32\_t RESERVED1[2];

\_\_vo uint32\_t AHB1ENR[3]; //+30 offset

//\_\_vo uint32\_t AHB2ENR; //+34 offset

//\_\_vo uint32\_t AHB3ENR; //+38 offset

uint32\_t RESERVED2;

\_\_vo uint32\_t APB1ENR[2]; //+40 offset

//\_\_vo uint32\_t APB2ENR; //+44 offset

uint32\_t RESERVED3[2];

\_\_vo uint32\_t AHB1LPENR[3]; //+50 offset

//\_\_vo uint32\_t AHB2LPENR; //+54 offset

//\_\_vo uint32\_t AHB3LPENR; //+58 offset

uint32\_t RESERVED4;

\_\_vo uint32\_t APB1LPENR[2]; //+60 offset

//\_\_vo uint32\_t APB2LPENR; //+64 offset

uint32\_t RESERVED5[2];

\_\_vo uint32\_t BDCR; //+70 offset

\_\_vo uint32\_t CSR; //+74 offset

uint32\_t RESERVED6[2];

\_\_vo uint32\_t SSCGR; //+80 offset

\_\_vo uint32\_t PLLI2SCFGR; //+84 offset

\_\_vo uint32\_t PLLSAICFGR; //+88 offset

\_\_vo uint32\_t DCKCFGR; //+8C offset

} RCC\_RegDef\_t;

//define pointer for structure

//then bring pointer and structure together

RCC\_RegDef\_t \*pRCC = (RCC\_RegDef\_t\*)0x40023800; //type casting

//MACROS FOR GPIO POINTER STRUCTURE

**#define** GPIOA (GPIO\_RegDef\_t\*) GPIOA\_BASE\_ADDR

**#define** GPIOB ()

//to do

//MACROS FOR ENABLING THE CLOCK FOR GPIOS

**#define** GPIOA\_PCLOCK\_ENABLE (RCC->AHB1ENR = (1<<0))

**#define** GPIOA

//to do

//MACROS FOR DISABLING THE CLOCK FOR GPIOS

**#define** GPIOA\_PCLOCK\_DISABLE (RCC->AHB1ENR = ~(1<<0))

//TO DO