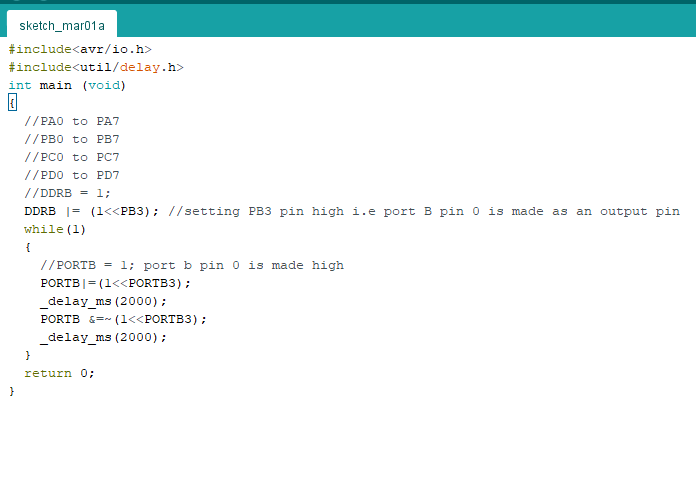
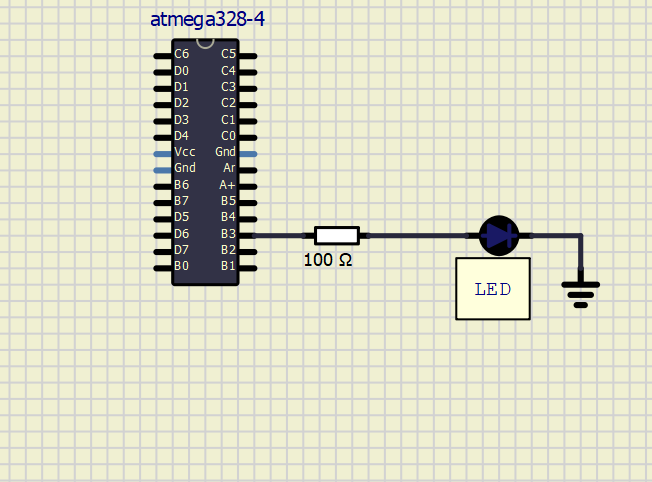
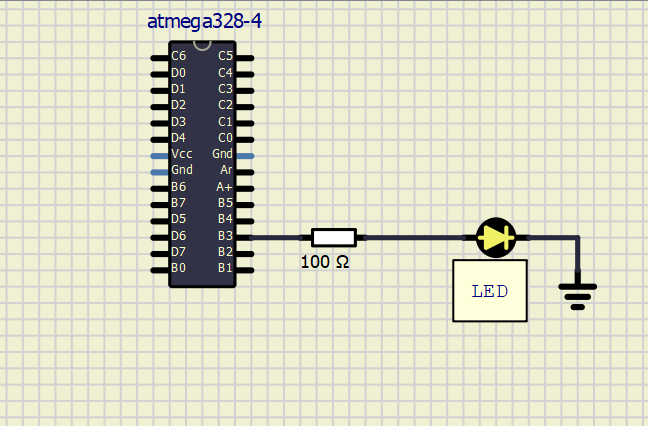
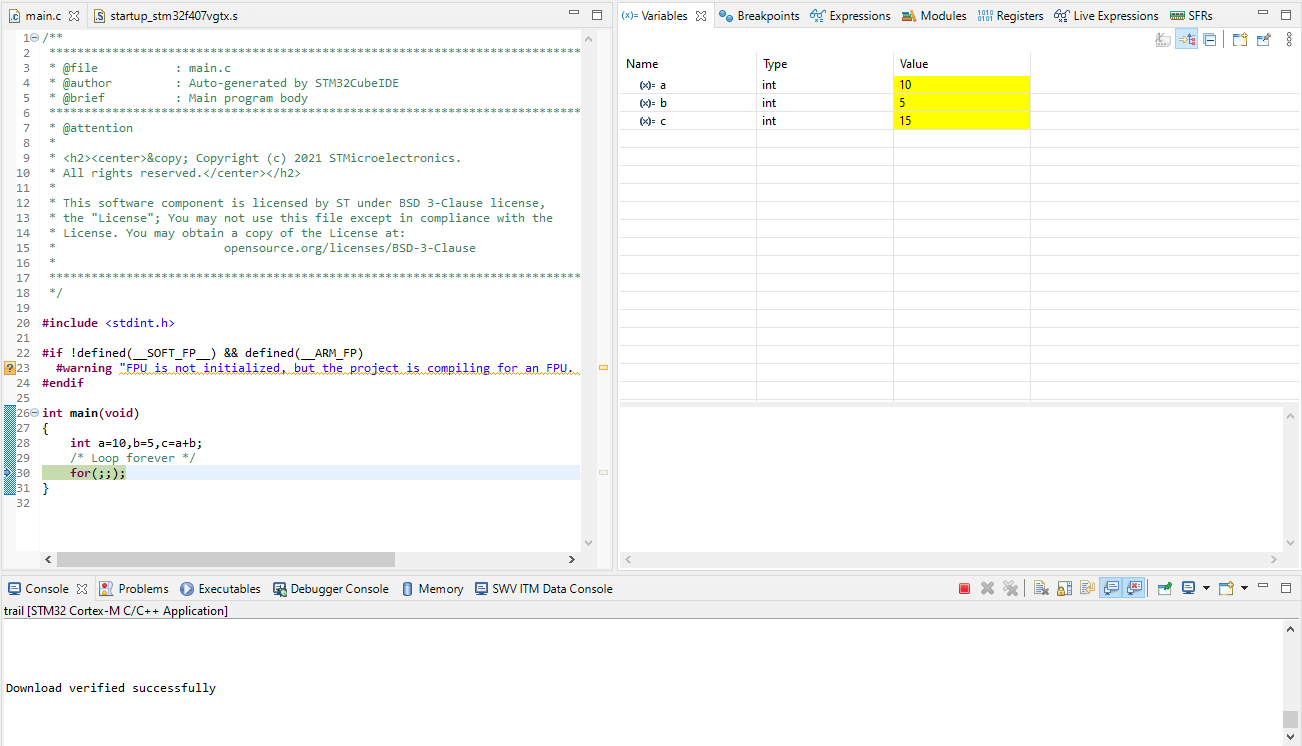
**Arduino beginning:**







**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;

}

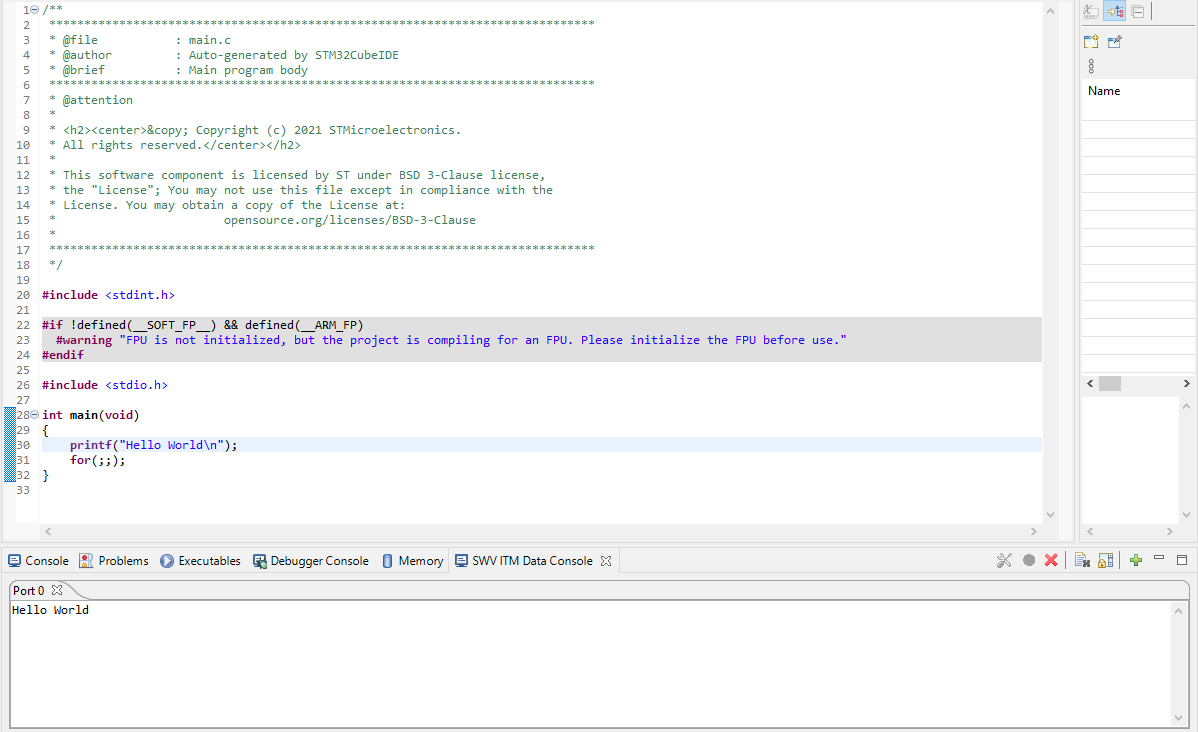
**int** **\_execve**(**char** \*name, **char** \*\*argv, **char** \*\*env)

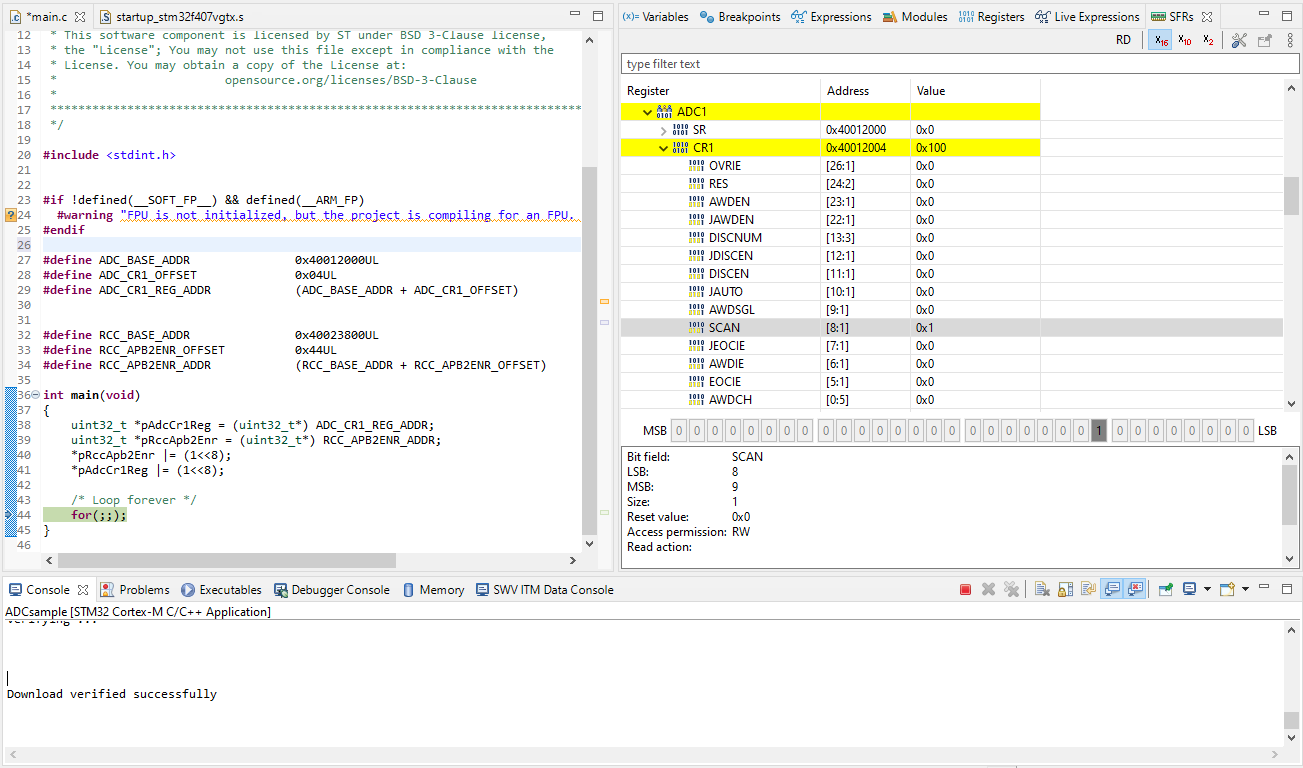
{

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(;;);

}