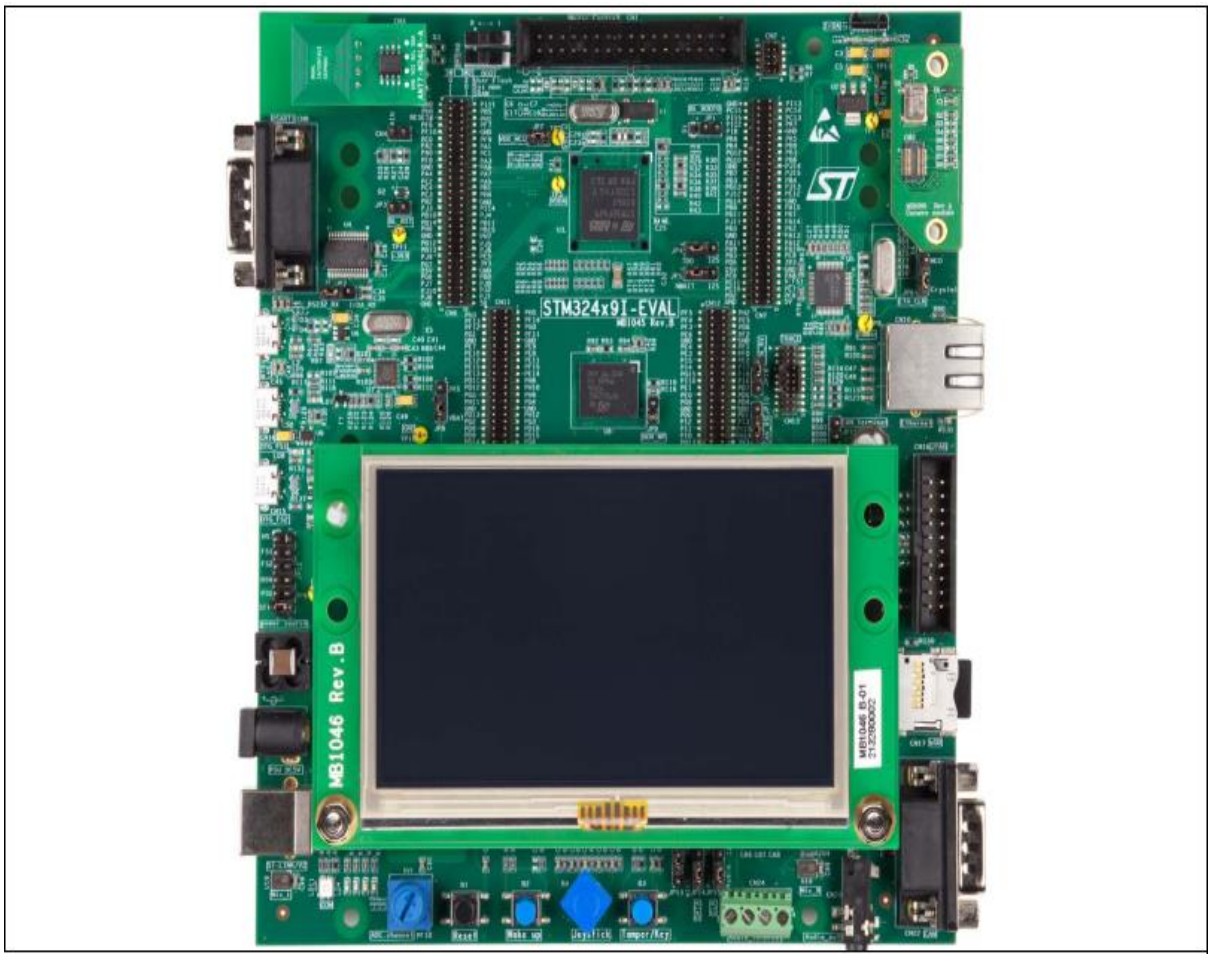
LABORATORY MANUAL

SUBJECT: EMBEDDED SYSTEMS LAB

ERS MMT

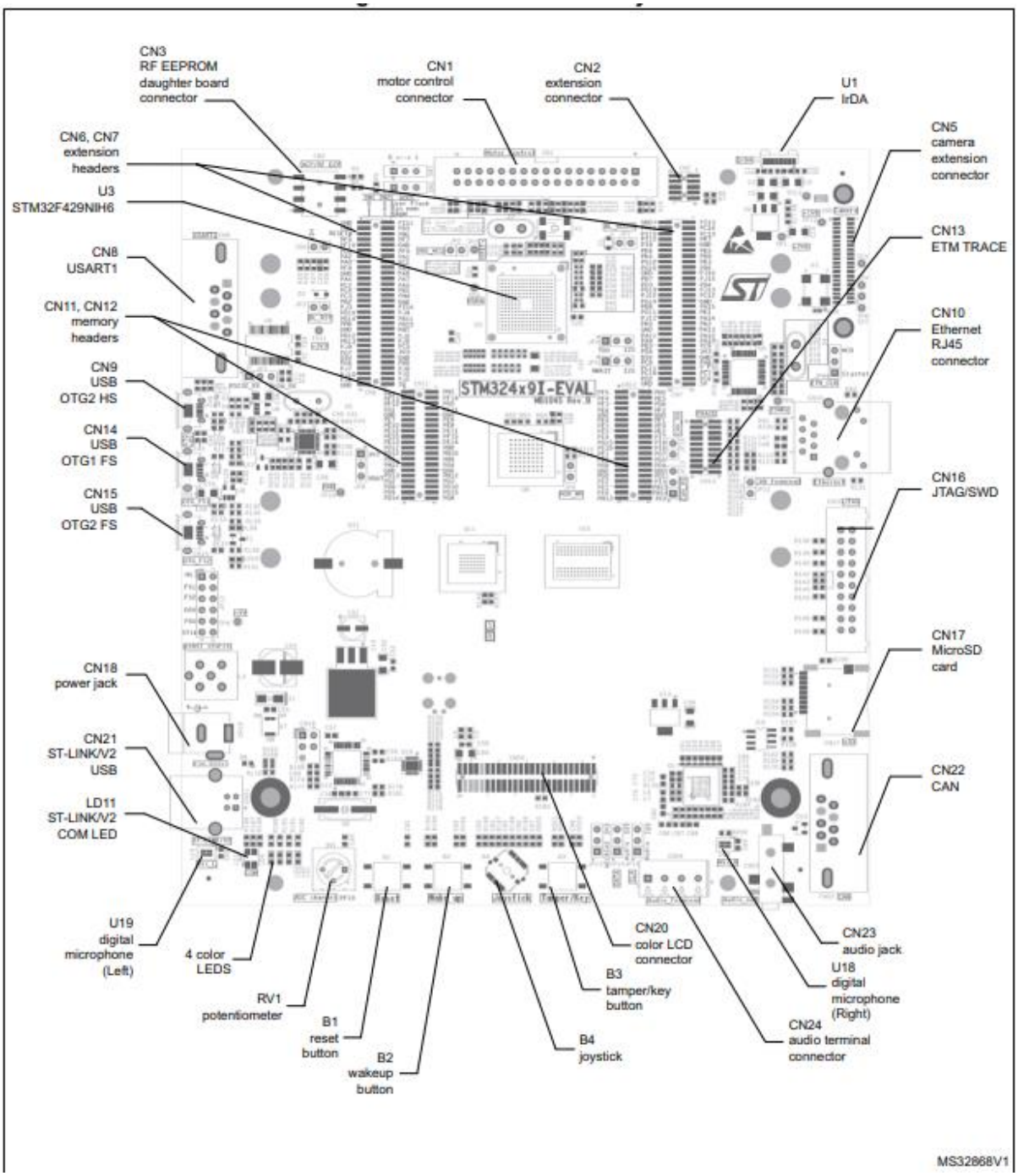
HCL Technologies Ltd

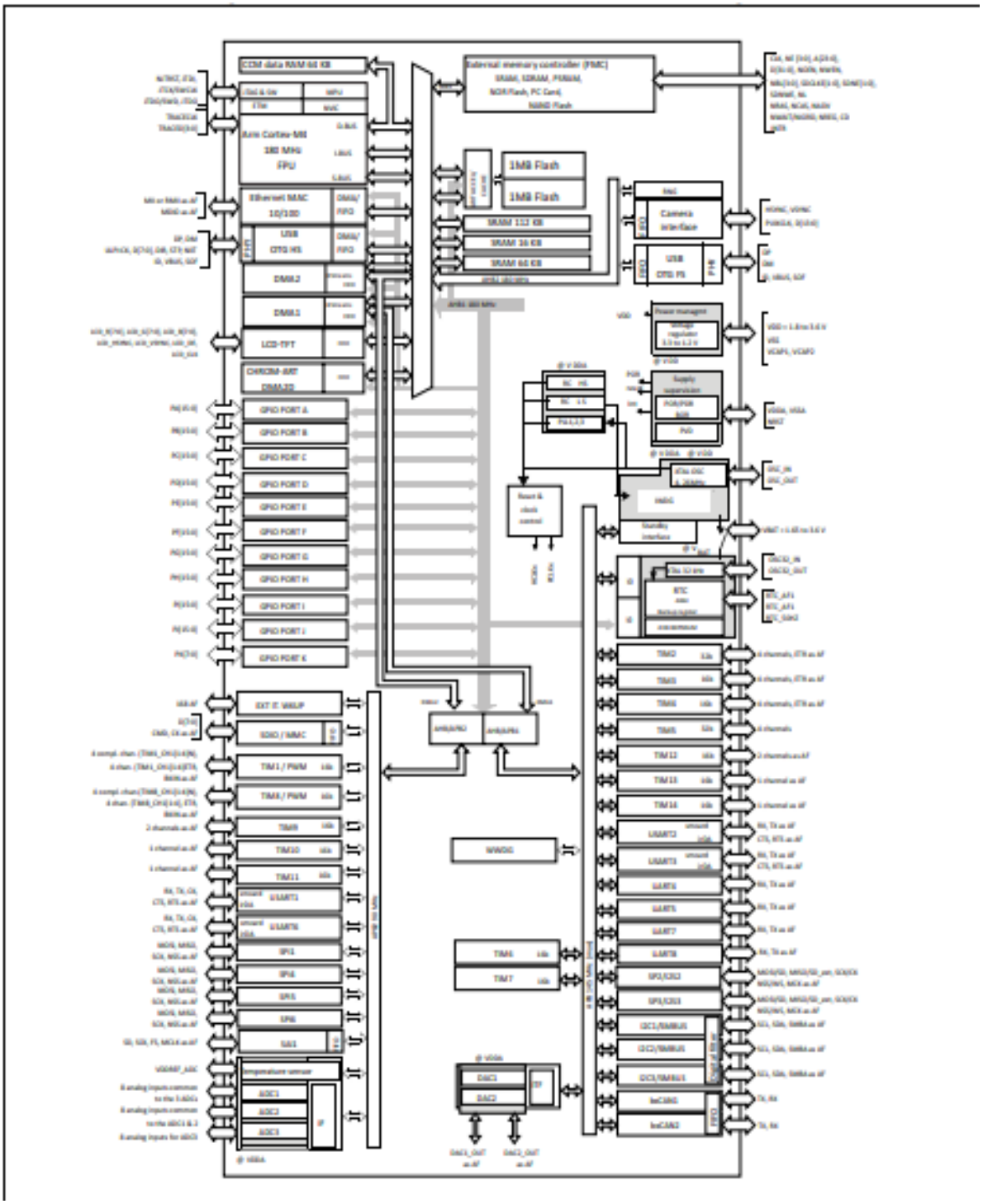
STM32F429I Discovery Board Details



STM32F429I Discovery Board

STM32F429I


CN3
RF EEPROM
daughter board
connector
CN1
motor control
connector
CN2
extension
connector
U1
IrDA
CN6, CN7
extension
headers
U3
STM32F429NIH6
CN5
camera
extension
connector
CN13
ETM TRACE
CN8
USART1
CN11, CN12
memory
headers
CN10
Ethernet
RJ45
connector
CN9
USB
OTG2 HS
STM324x9I-EVAL
CN14
USB
OTG1 FS
CN15
USB
OTG2 FS
CN16
JTAG/SWD
CN17
MicroSD
card
CN18
power jack
CN21
ST-LINK/V2
USB
CN22
CAN
LD11
ST-LINK/V2
COM LED
U19
digital
microphone
(Left)
4 color
LEDS
RV1
potentiometer
B1
reset
button
B2
wakeup
button
B4
joystick
B3
tamper/key
button
CN20
color LCD
connector
CN23
audio jack
U18
digital
microphone
(Right)
CN24
audio terminal
connector
MS32868V1

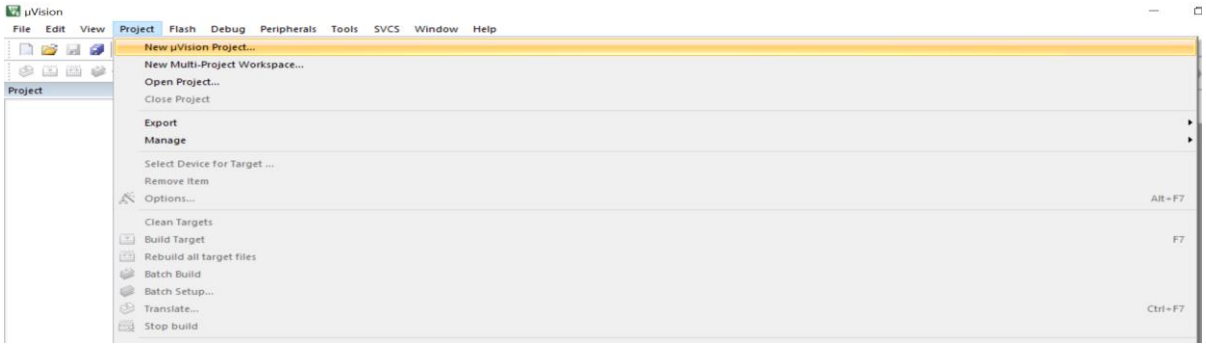

CCM data RAM 64 KB
External memory controller (FMC)
SRAM, SDRAM, PSRAM,
NOR Flash, PC Card,
NAND Flash
Arm Cortex-M4
180 MHz
FPU
1MB Flash
1MB Flash
Ethernet MAC
10/100
DMA/
FIFO
USB
OTG HS
DMA2
DMA1
LCD-TFT
CHROM-ART
DMA2D
SRAM 112 KB
SRAM 16 KB
SRAM 64 KB
DCMI
Camera
interface
USB
OTG FS
GPIO PORT A
GPIO PORT B
GPIO PORT C
GPIO PORT D
GPIO PORT E
GPIO PORT F
GPIO PORT G
GPIO PORT H
GPIO PORT I
GPIO PORT J
GPIO PORT K
EXT IT WKUP
SDIO / MMC
TIM1 / PWM
TIM8 / PWM
TIM9
TIM10
TIM11
USART1
USART6
SPI1
SPI4
SPI5
SPI6
SAI1
temperature sensor
ADC1
ADC2
ADC3
IWDG
TIM6
TIM7
DAC1
DAC2
TIM2
TIM3
TIM4
TIM5
TIM12
TIM13
TIM14
USART2
USART3
UART4
UART5
UART7
UART8
SPI2/I2S2
SPI3/I2S3
I2C1/SMBUS
I2C2/SMBUS
I2C3/SMBUS
bxCAN1
bxCAN2
Power managmt
Supply
supervision
Reset &
clock
control
RTC
Backup register
Standby
interface

1 Features

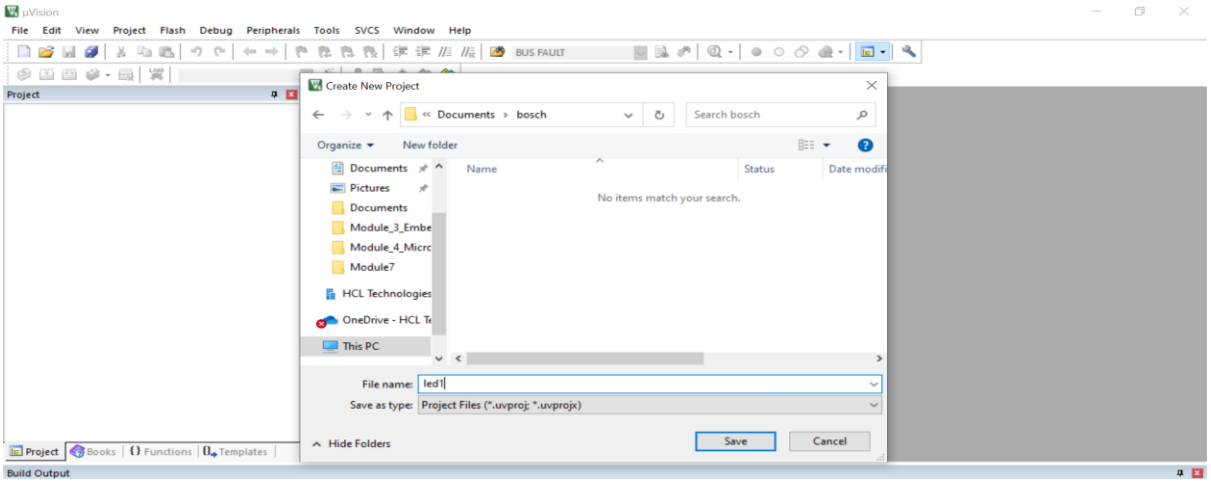
- STM32F429NIH6 microcontroller based on the Arm® Cortex®-M4 core[a]
- 4.3-inch 480 × 272 pixel TFT LCD with resistive touch panel
- Camera module
- SAI Audio DAC
- Stereo MEMS microphone
- IEEE-802.3-2002 compliant Ethernet
- USB OTG HS and FS
- 2-Gbyte (or more) SDIO interface microSD™ card
- RF-EEPROM on I^2C compatible serial interface
- RS-232 communication
- CAN2.0A/B compliant connection
- IrDA transceiver
- 8 M × 32-bit SDRAM, 1 M × 16-bit SRAM, 8 M × 16-bit NOR Flash memory
- Joystick with 4-direction control and selector
- Reset, WakeUp, Tamper or key button
- 4 user color LEDs
- RTC with backup battery
- Potentiometer
- Board connectors:
  - microSD™ card
  - 3× USB Micro-AB and 1× USB Type-B
  - Audio terminal connector for external speakers
  - 3.5 mm stereo audio jack
  - Motor control
  - 2× RS-232
  - RJ45
  - Extension connectors and memory connectors for daughterboard or wrapping board
- Six 5 V power supply options: power jack, ST-LINK/V2, User USB HS, User USB FS1, User USB FS2, daughterboard
- Debug with JTAG/SWD, ETM trace, and on-board ST-LINK/V2 debugger/programmer
- Comprehensive free software libraries and examples available with the STM32CubeF4 MCU Package
- Support of a wide choice of Integrated Development Environments (IDEs) including IAR Embedded Workbench®, MDK-ARM, and STM32CubeIDE

Procedure to work on Keil New µvision

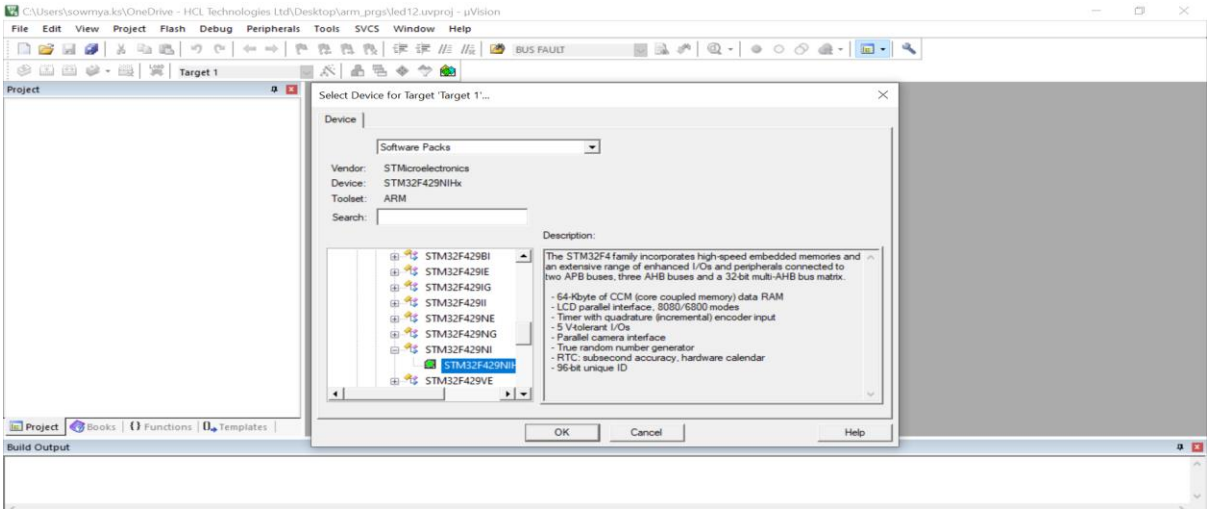
Step 1: Project → New µvision Project



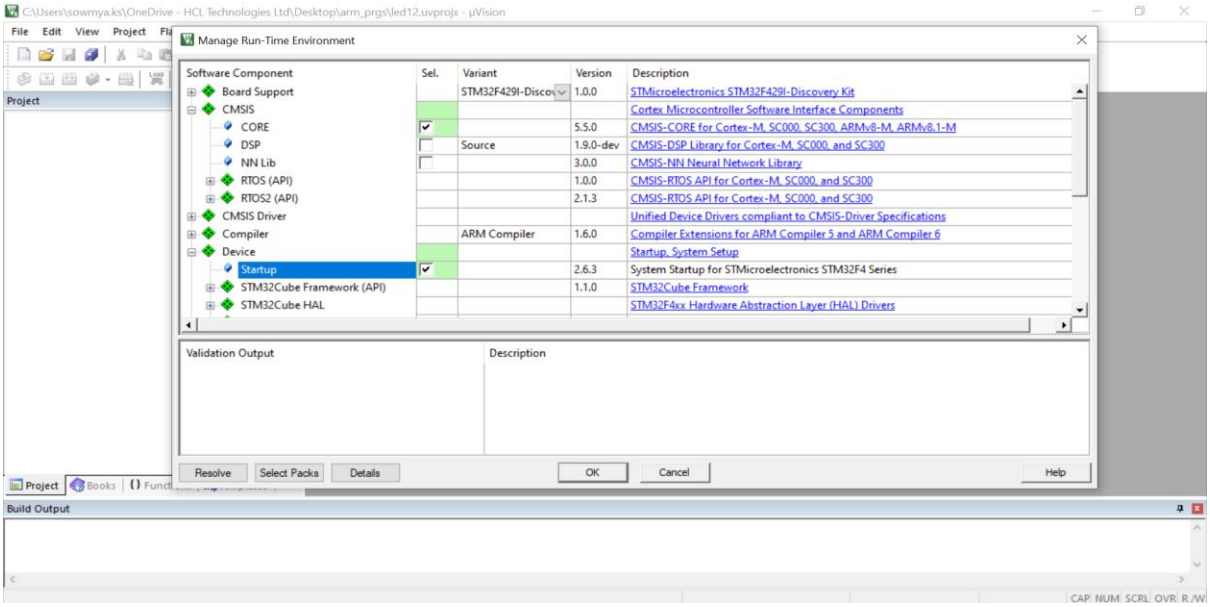
Step 2: Choose the Location and Give the Project name and save



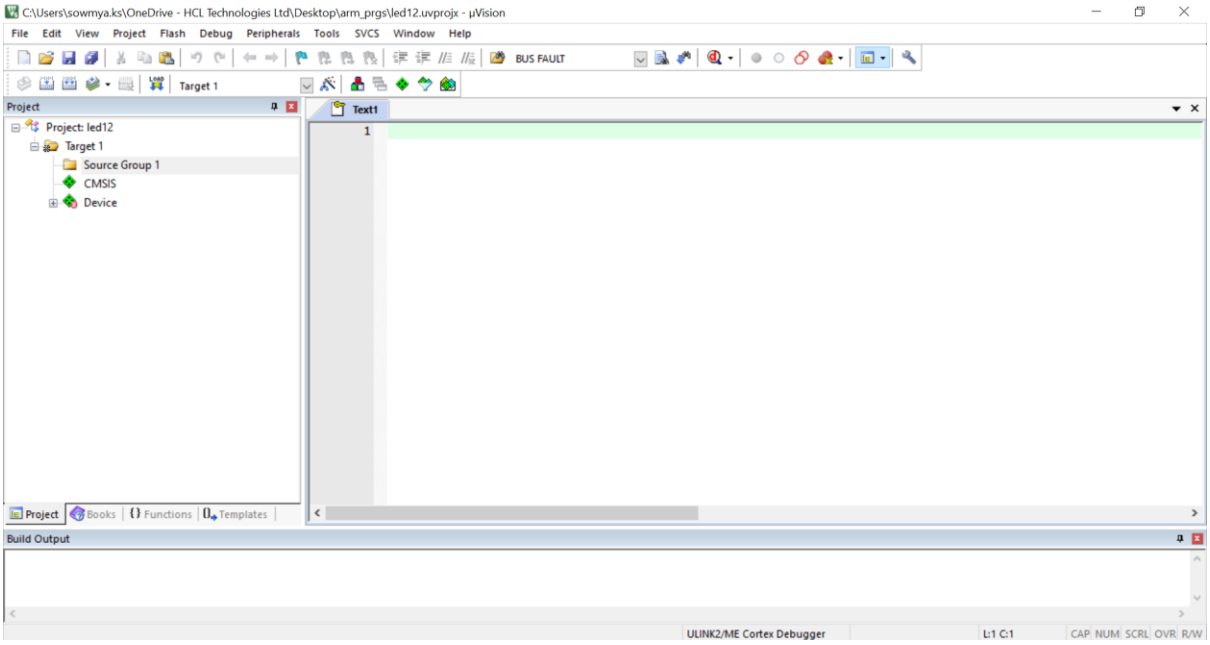
Step 3: Select the Device STM32F429NI for the Target and press ok



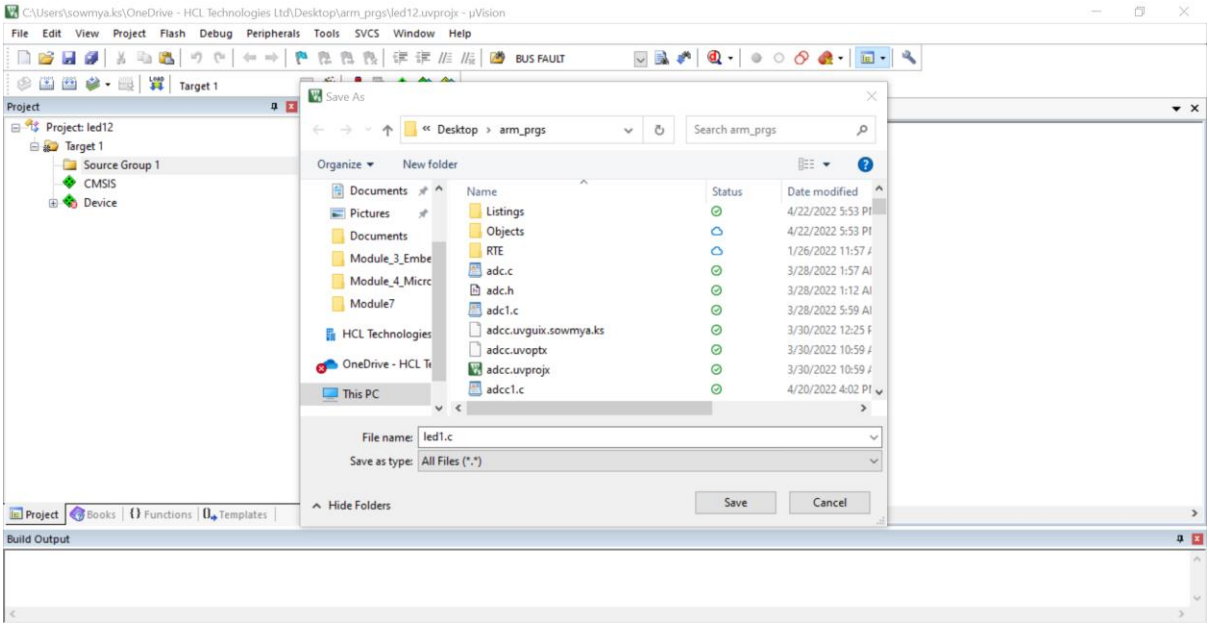
Step 4: In Manage RunTime Enviornment ---→ Select STM32F429I Discovery under Variant , CORE under CMSIS, Startup under Device and press ok



Step 5: Create a new source File --→ File → New

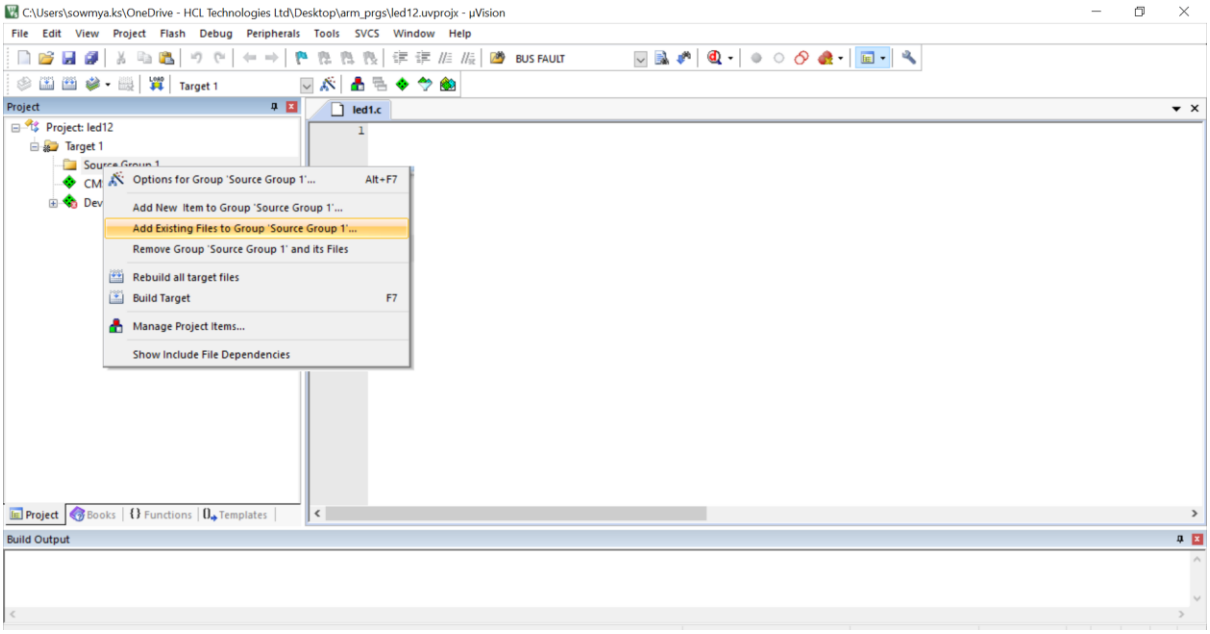


Step 6: File → Save as -→ filename.c

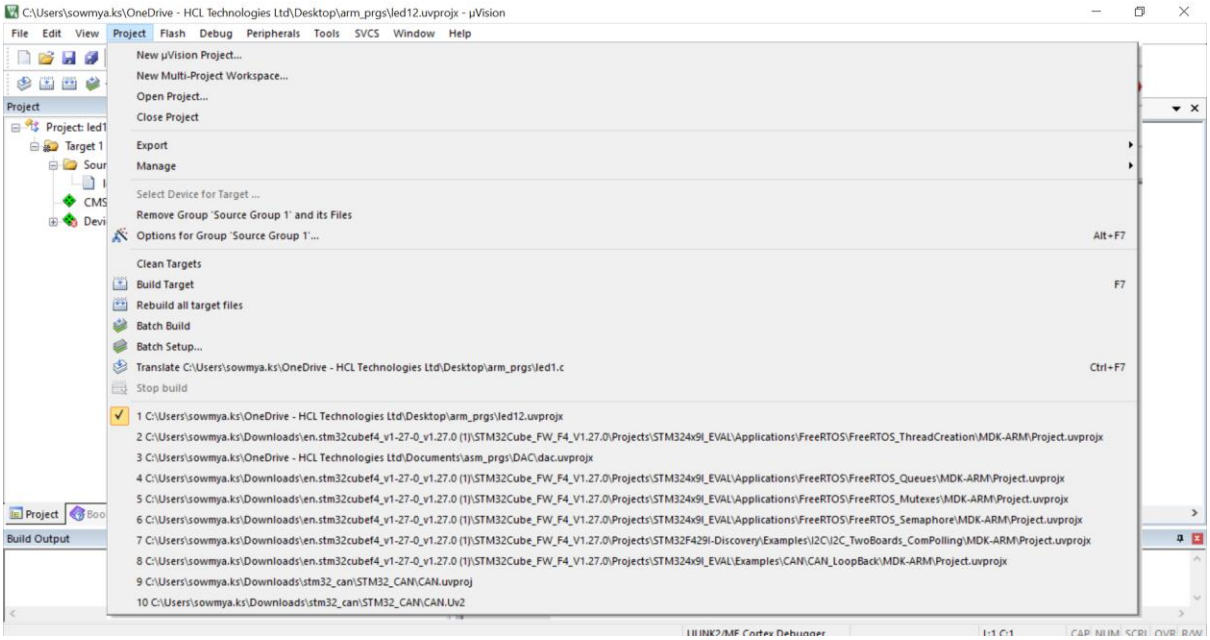


Step 7: Enter the program and save

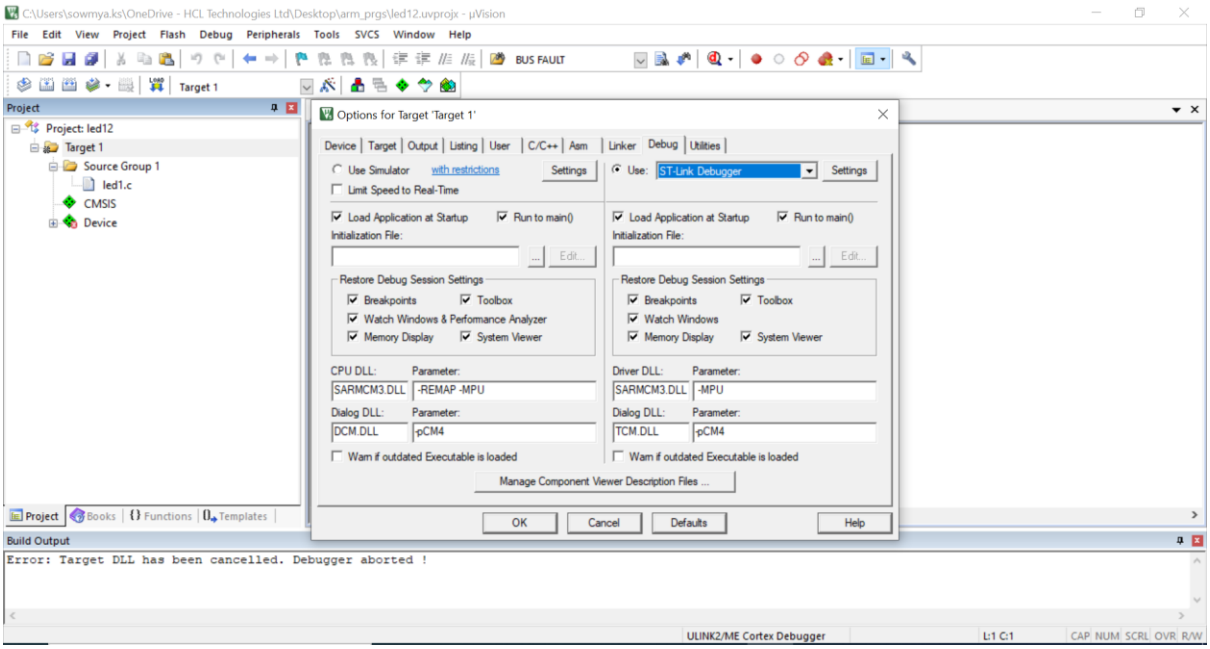
Step 8: Add the source file to Target. Right click on source group1→ Add Existing file to source group1→ choose the file →add-→close



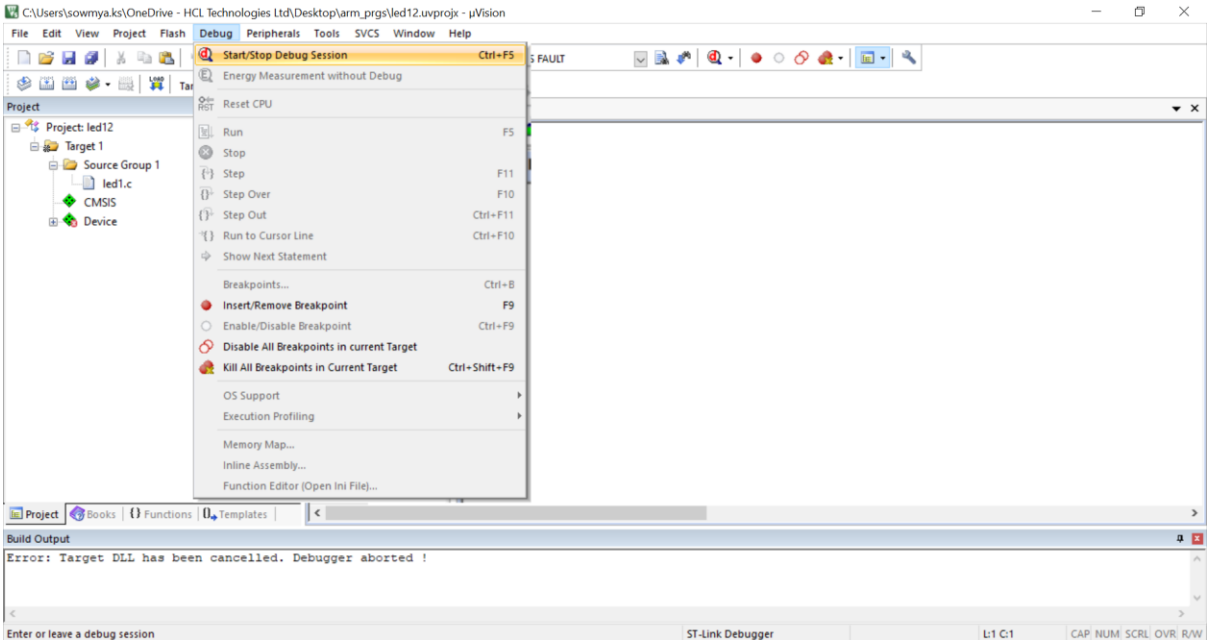
Step 9: Project -→ Translate , Build and Rebuild all



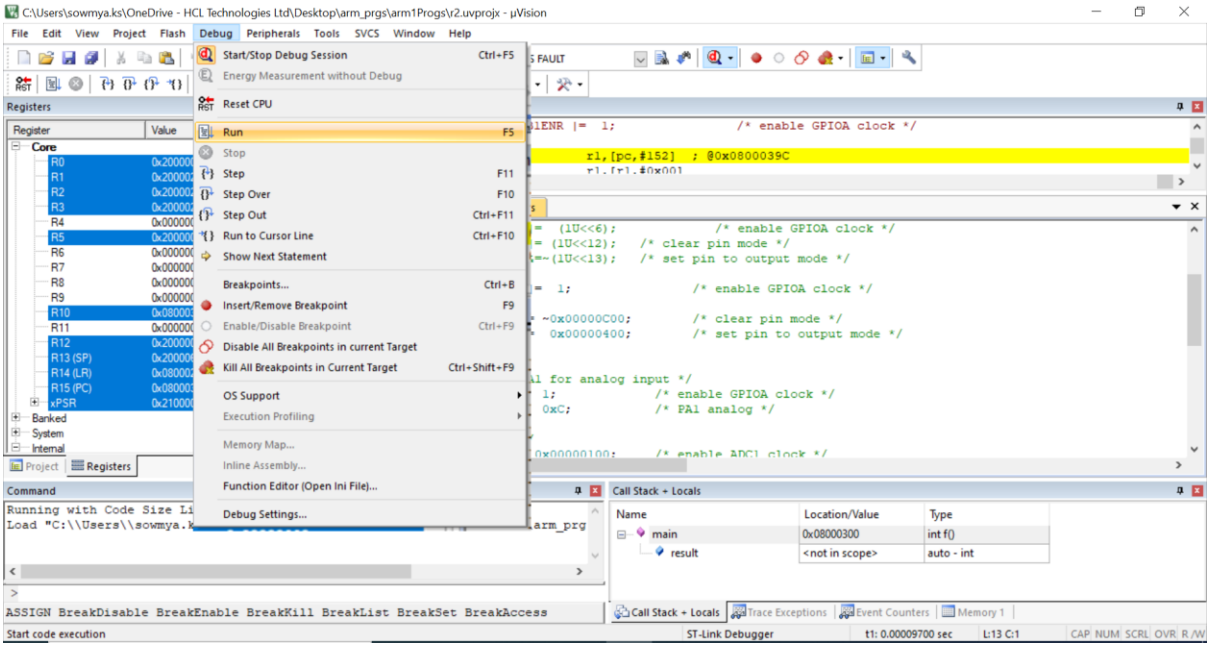
Step 10: Right click on Target1 → Options for Target 1 → Debug → select Use STlink Debugger and Ok



Step 11: Debug --→ Start/Stop debug Session



Step 12: Debug → Run



Programs

LED and Switch Interfacing

Experiment 1.a: Programming on board LED

Pin configuration

PG6 PIN IS CONNECTED TO inbuilt LED1

```
#include "stm32f4xx.h"

void delayMs(int n);

int main(void) {
RCC->AHB1ENR |=  1<<6;            /* enable GPIOG clock */

   GPIOG->MODER &= ~0x00003000;    /* mode 6 your clearing ,clear pin mode */
   GPIOG->MODER |=  0x00001000;    /* set pin to output mode */

   while(1) {
    //   GPIOG->ODR |=  0x00000040;  /* turn on LED             0100 0000 */
     //  delayMs(500);
     //  GPIOG->ODR &= ~0x00000040;  /* turn off LED */
     //  delayMs(500);
                                GPIOG->ODR ^=  0x00000040;  /* turn on LED */
                                for(int i=0;i<100000;i++){}

   }
}

/* 16 MHz SYSCLK */
void delayMs(int n) {
   int i;
   for (; n > 0; n--)
       for (i = 0; i < 3195; i++) ;
}
```

EXPERIMENT:1.B PROGRAMMING ON BOARD LED_ALTERNATE_2_4

PROGRAM:

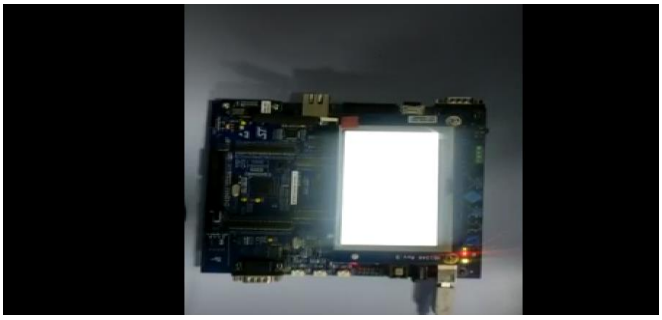
Pin configuaration

PG6 PIN IS CONNECTED TO inbuilt LED1

```
#include "stm32f4xx.h"

void delayMs(int n);

int main(void) {
RCC->AHB1ENR |=  1<<6;            /* enable GPIOG clock */

   GPIOG->MODER &= ~0x00003000;    /* clear pin mode */
         //GPIOG->MODER &= ~0x0000c000;    //* clear pin mode fpr pg7 */
         GPIOG->MODER &= ~0x00300000;   /* clear pin mode pg10 */
         //GPIOG->MODER &= ~0x03000000;    /* clear pin mode  pg12*/



   GPIOG->MODER |=  0x00001000;    /* set pin to output mode */
         //GPIOG->MODER |=  0x00004000;     /* set pin to output mode pg7 */
         GPIOG->MODER |=  0x00100000;      /* set pin to output mode  pg10*/
         //GPIOG->MODER |=  0x01000000;      /* set pin to output mode pg 12*/

   while(1) {
    //   GPIOG->ODR |=  0x00000040;  /* turn on LED            01000000 */
     //  delayMs(500);
```

```
    //  GPIOG->ODR &= ~0x00000040;  /* turn off LED */
    //  delayMs(500);
                        GPIOG->ODR ^=  0x00000040;     /* turn on LED */
                        //GPIOG->ODR ^=  0x00000080;     /* turn on LED */
                        GPIOG->ODR ^=  0x00000400;                    /* turn on LED */
                        //GPIOG->ODR ^=  0x00001000;      /* turn on LED */
                        for(int i=0;i<100000;i++){}
                         }
}
/* 16 MHz SYSCLK */
void delayMs(int n) {
   int i;
   for (; n > 0; n--)
      for (i = 0; i < 3195; i++) ;
}
```

OUTPUT DIAGRAM:



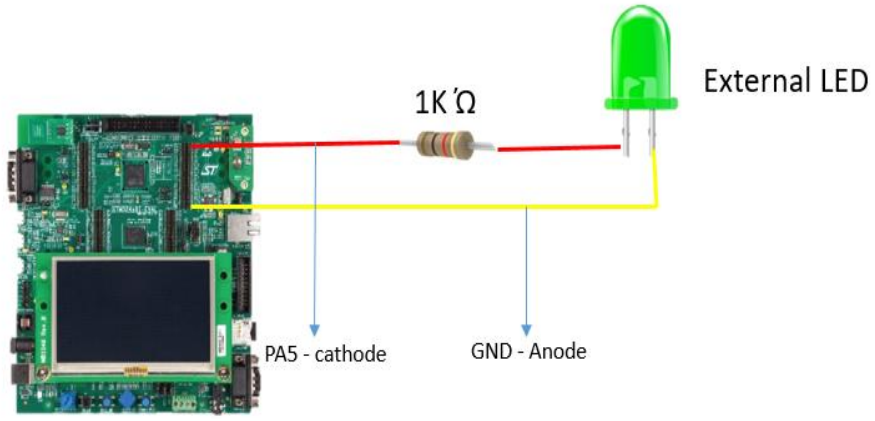
Experiment 1.c : Interfacing External LED to STM32 Board

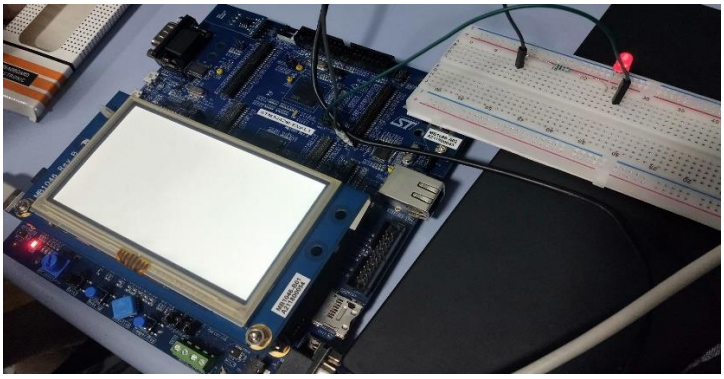
Pin configuarationtgn

PA5 PIN IS CONNECTED TO LED

```
#include "stm32f4xx.h"
int main(void) {
```

```
    RCC->AHB1ENR |=  4;              /* enable GPIOC clock */
    RCC->AHB1ENR |=  1;              /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;       /* clear pin mode */
    GPIOA->MODER |=  0x00000400;       /* set pin to output mode */
    GPIOC->MODER &= ~0x0C000000;       /* clear pin mode to input mode                    0*/
    while(1) {
      if (GPIOC->IDR & 0x2000)       /* if PC13 is high */
        GPIOA->BSRR = 0x00200000;   /* turn off green LED */
      else
        GPIOA->BSRR = 0x00000020;   /* turn on green LED */
    }
}
```





Experiment 1.d : Programming LED and external Switch

Pin configuaration

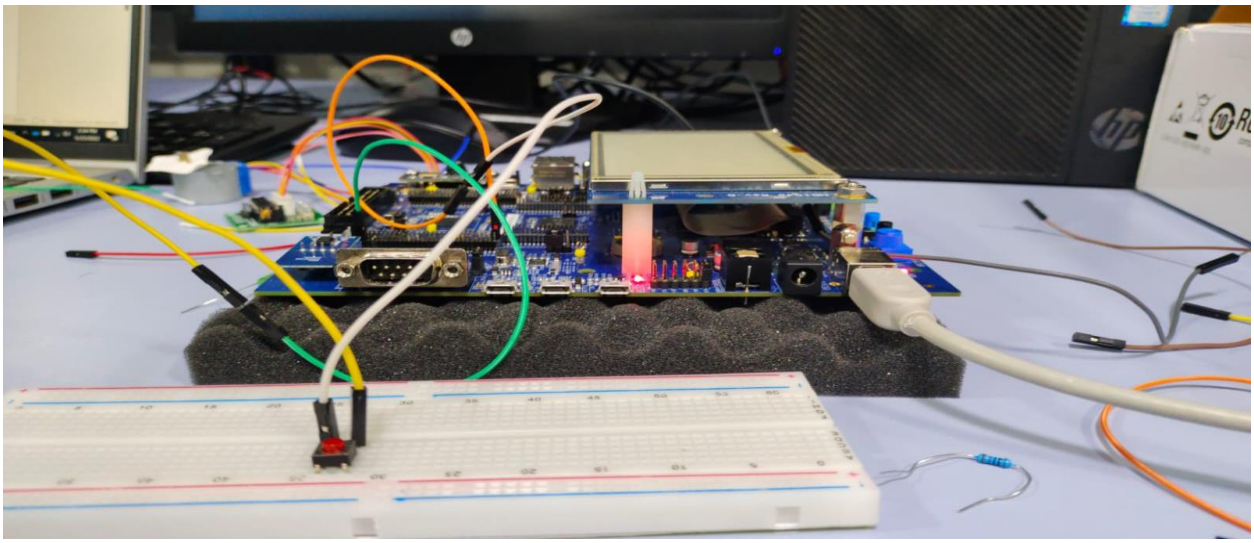
PG6 PIN IS CONNECTED TO LED1

PC13 is connected to Switch

Program

```
#include "stm32f4xx.h"

int main(void) {

   RCC->AHB1ENR |=  1<<2;              /* enable GPIOC clock  led*/

   RCC->AHB1ENR |=  1<<6;              /* enable GPIOA clock */

   GPIOG->MODER &= ~0x0000 3000;       /* clear pin mode 6th mode */

   GPIOG->MODER |=  0x00001000;        /* set pin to output mode 6th mode as output */

   GPIOC->MODER &= ~0x0C00 0000;       /* clear pin mode to input mode,,,13th mode clear
and set as input mode  */

   while(1) {

      if (GPIOC->IDR & 0x2000)        /* if PC13 is high */0010 0000 0000 0000

         GPIOG->BSRR = 0x00400000;   /* turn off green LED  6th reset*/

         else

         GPIOG->BSRR = 0x00000040;   /* turn on green LED  6th bit set*/ 0000
0100 0000
```

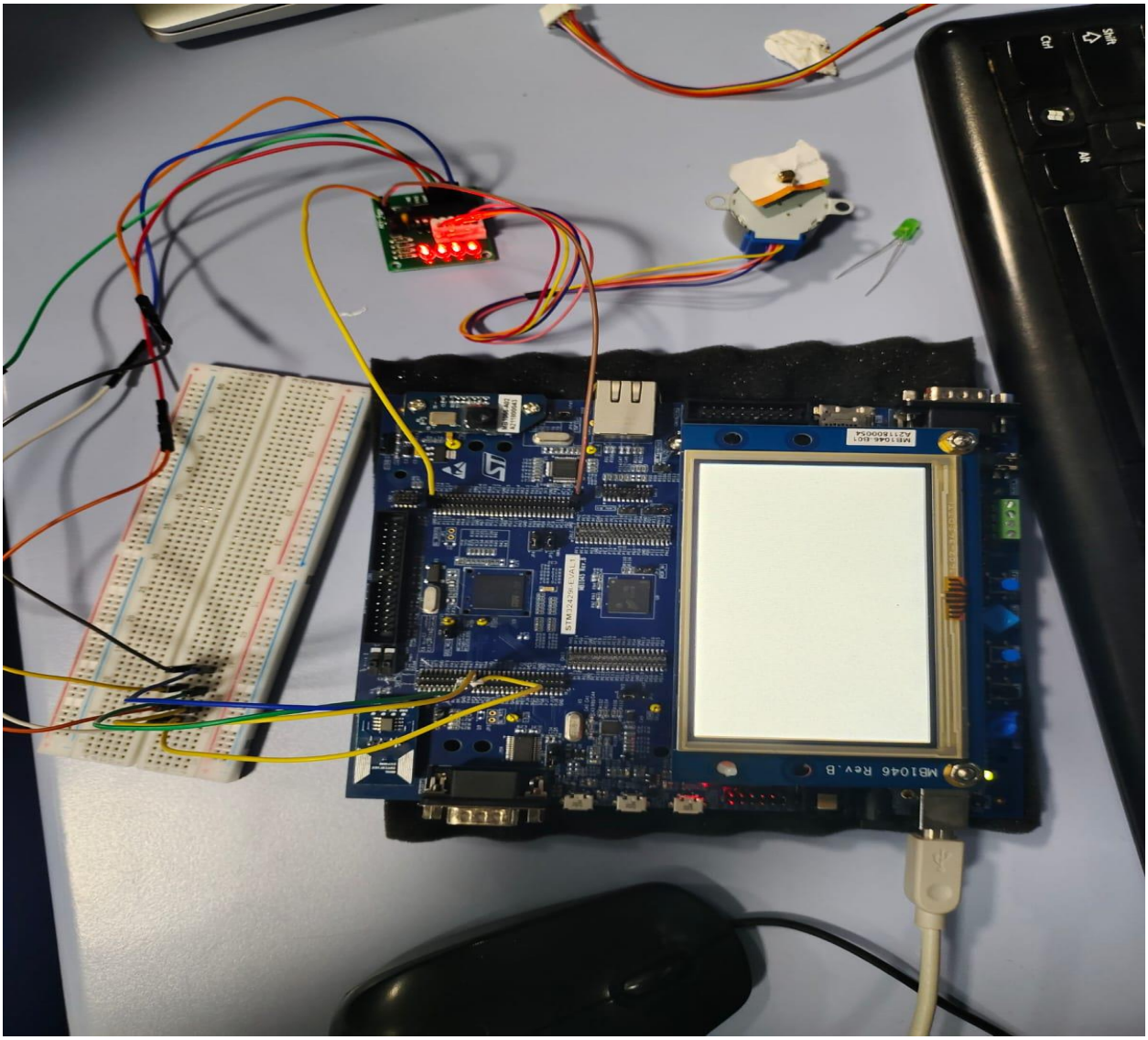
```
    }
}
```



STEPPER MOTOR INTERFACING

Experiment 2: Interfacing Stepper Motor to STM32 Board

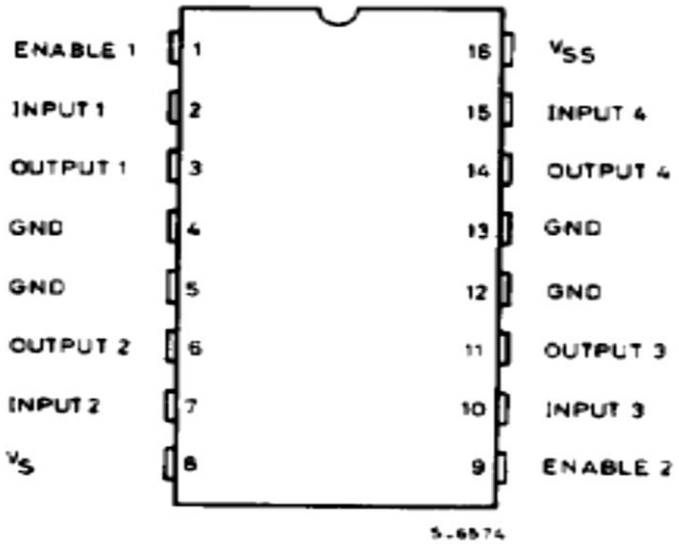
```
#include "stm32f4xx.h"
void delayMs(int n);
int main(void) {
   const char steps[ ] = {0x90, 0x30, 0x60, 0xC0};
   int i;
   /* PA 7, 6, 5, 4 for motor control */
   RCC->AHB1ENR |=  1;                /* enable GPIOA clock */
   GPIOA->MODER &= ~0x0000 FF00;    /* clear pin mode */
   GPIOA->MODER |=  0x00005500;    /* set pins to output mode */ 01 01 01 01
   while (1)
                    {
   for (i=0;i<=3;i++) {
      GPIOA->ODR = steps[i ];   /* set the output pattern */
      delayMs(100);
   }
         }
}
/* 16 MHz SYSCLK */
void delayMs(int n) {
   int i;
   for (; n > 0; n--)
      for (i = 0; i < 3195; i++) ;
}
```


MB1046 Rev.B

DC MOTOR INTERFACING

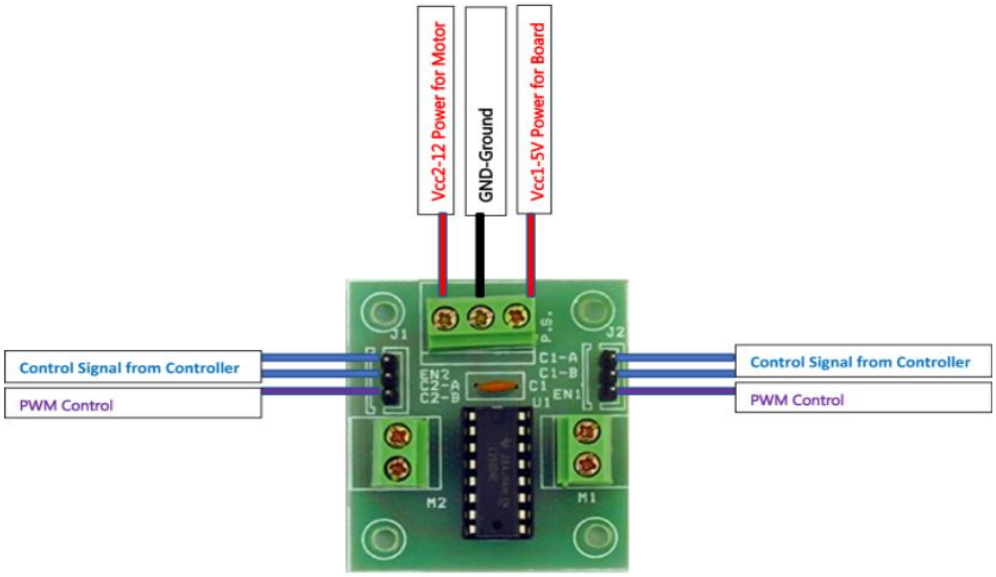
INTERFACING DIAGRAM

Interfacing Diagram



Powerdip(12+2+2)

Block Diagram of L293D



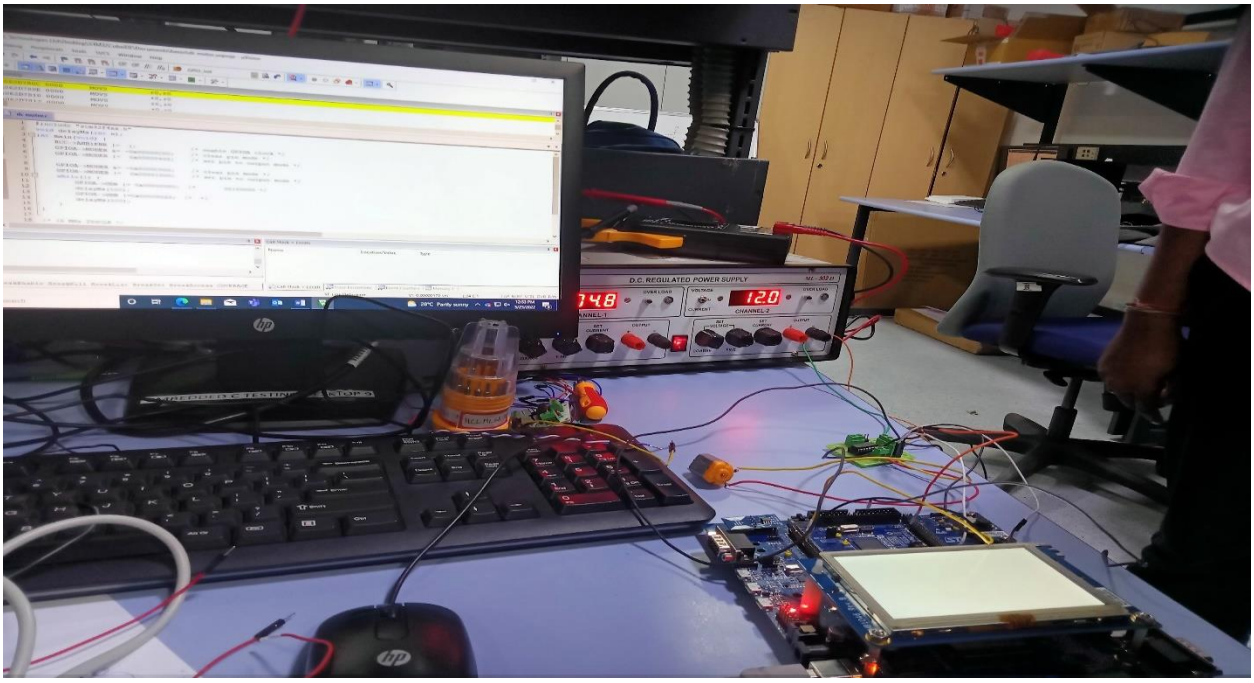
Block Diagram of L293D Motor Driver Board

- A and B are Control Signal from Controller used to control the motor.
- If logical input A=1 & B=0 given to Motor then It Rotates in One direction suppose clockwise. And if A=1 & B=0 the it will rotate in Anti clock wise.
- EN1 and EN2 are Enable pins. Connect 5V to EN1 and EN2 for normal operation. i.e. motor rotates at rated speed. (if 100 rpm motor is used, it rotates at 100 rpm)
- If Speed control is Needed the give PWM signals to EN1 and EN2 from Controller.
- Vcc2 (12V) is the Power Supply for the motor. If 12V dc geared motor is used, then connect 12V to Vcc2.
- Make sure to Connect to connect ground to both Controller GND and 12 V supply GND.

Program

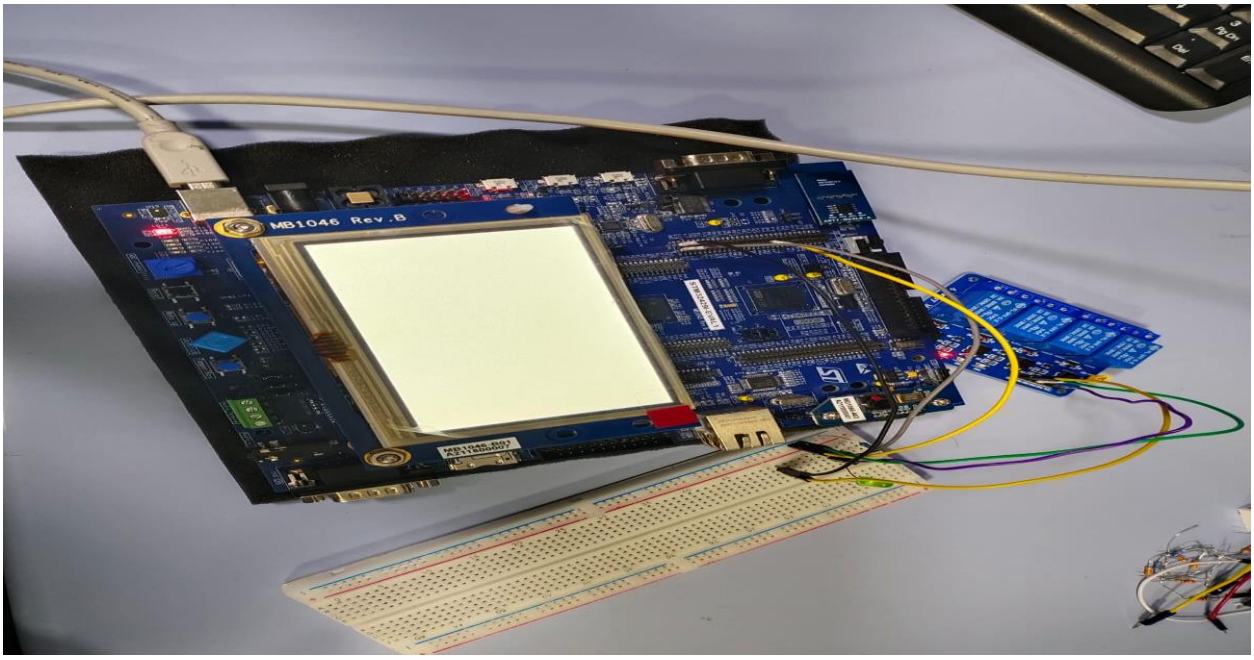
```
#include "stm32f4xx.h"

void delayMs(int n);

int main(void) {
```

```
    RCC->AHB1ENR |=  1;            /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000400;    /* set pin to output mode */
    GPIOA->MODER &= ~0x0000C000;    /* clear pin mode */
    GPIOA->MODER |=  0x00004000;    /* set pin to output mode */
    while(1) {
        GPIOA->ODR |=  0x00000080;  /*       1000 0000 */
        delayMs(500);
        GPIOA->ODR |=0x00000040;  /*  */0000 0100
        delayMs(500);
    }
}

/* 16 MHz SYSCLK */
void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 3195; i++) ;
}
```


D.C. REGULATED POWER SUPPLY
12.0
CHANNEL-2

RELAY INTERFACING

```
/* This program turns the relay connected to PA5 on and off every second.
#include "stm32f4xx.h"
void delayMs(int n);
int main(void) {
    RCC->AHB1ENR |=  1;              /* enable GPIOA clock */
   GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
   GPIOA->MODER |=  0x00000400;    /* set pin to output mode */
   while(1) {
      GPIOA->BSRR = 0x00000020;   /* turn on output */
      delayMs(500);
      GPIOA->BSRR = 0x00200000;   /* turn off output */
      delayMs(500);
   }
}
/* 16 MHz SYSCLK */
void delayMs(int n) {
   int i;
   for (; n > 0; n--)
      for (i = 0; i < 3195; i++) ;
}
```


MB1046 Rev.B

7 SEGMENT LED INTERFACING

```
#include "stm32f4xx.h"

void delayMs(int n);

int main(void) {

    RCC->AHB1ENR |=  4;             /* enable GPIOC
clock */

    GPIOC->MODER &= ~0x0000FFFF;    /* clear pin mode
*/
    GPIOC->MODER |=  0x00005555;    /* set pins to
output mode */
    for(;;)
    {
        GPIOC->ODR = 0x0007;            /* display ten
digit */

        delayMs(8);
    }
}

/* 16 MHz SYSCLK */
void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 3195; i++) ;
}
```

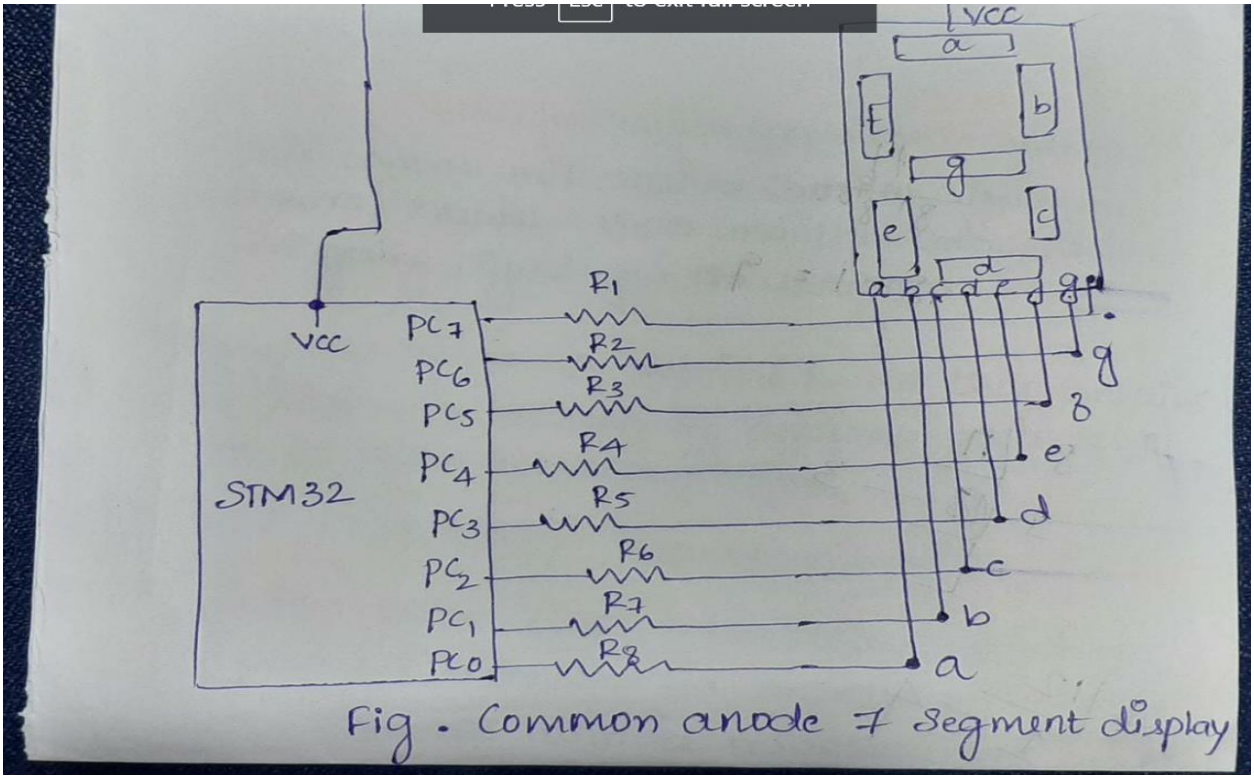
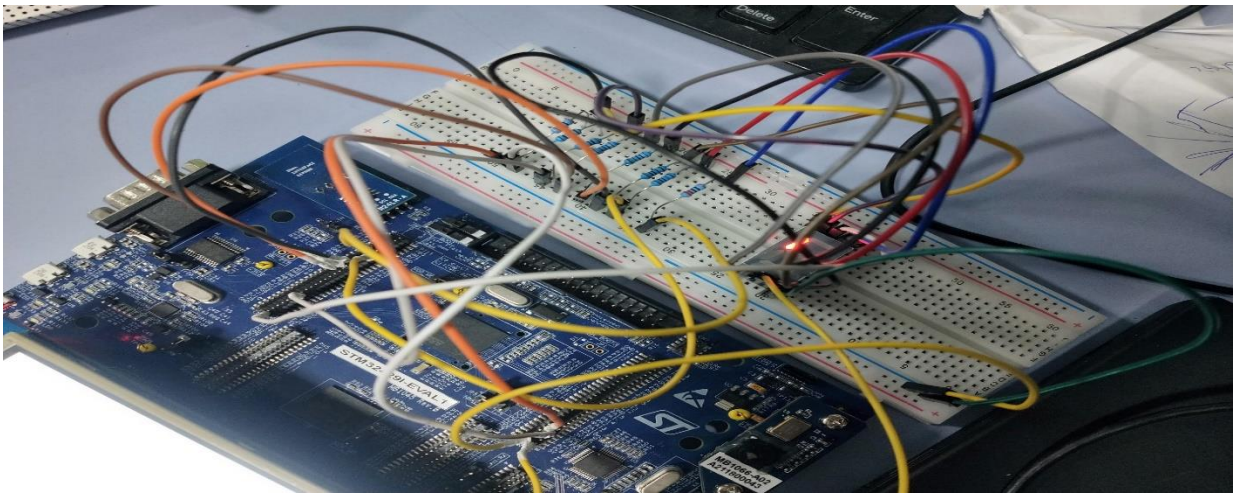


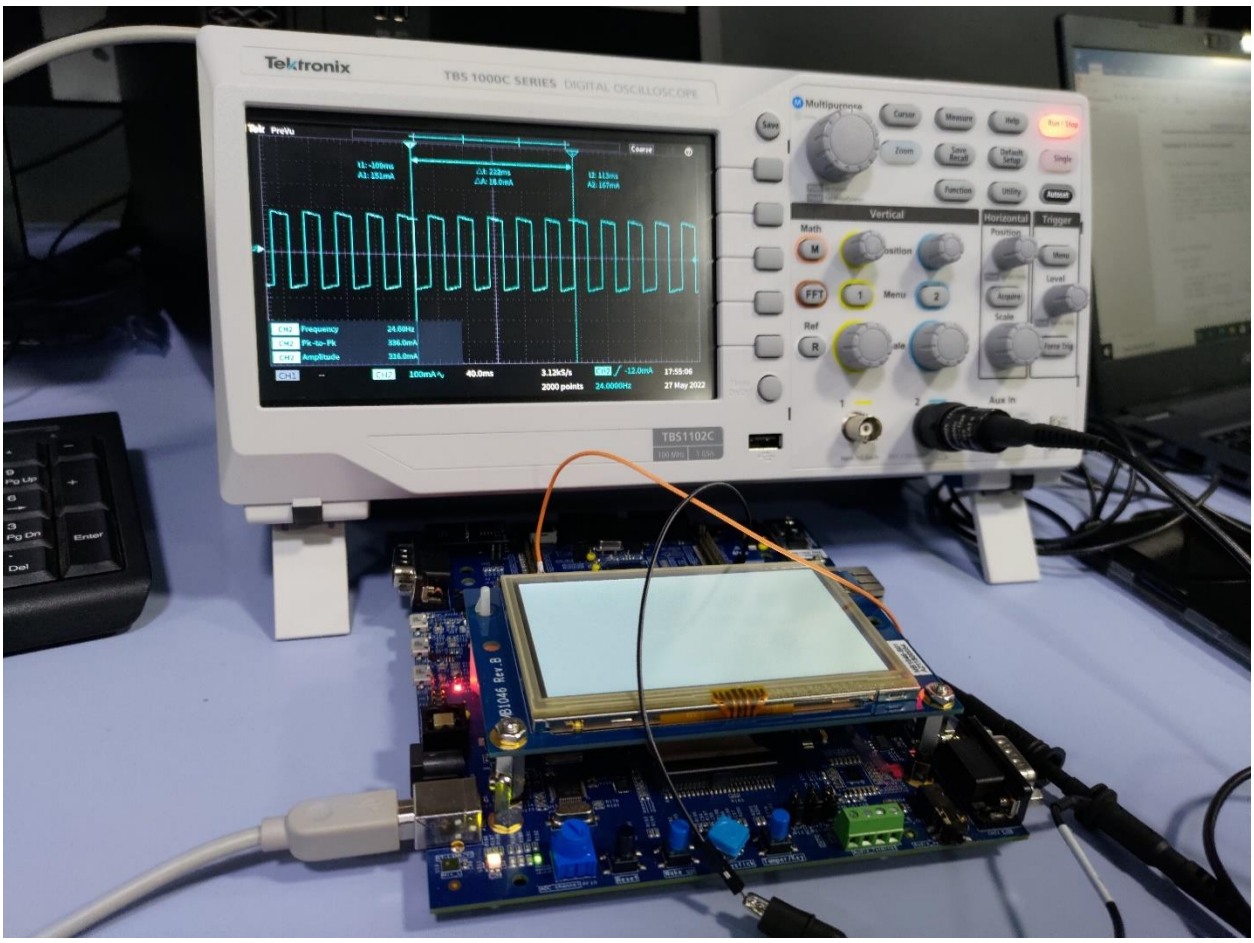
Fig. Common anode 7 segment display



TOGGLE GREEN LED (LD1) USING TIMERS (TIM2)

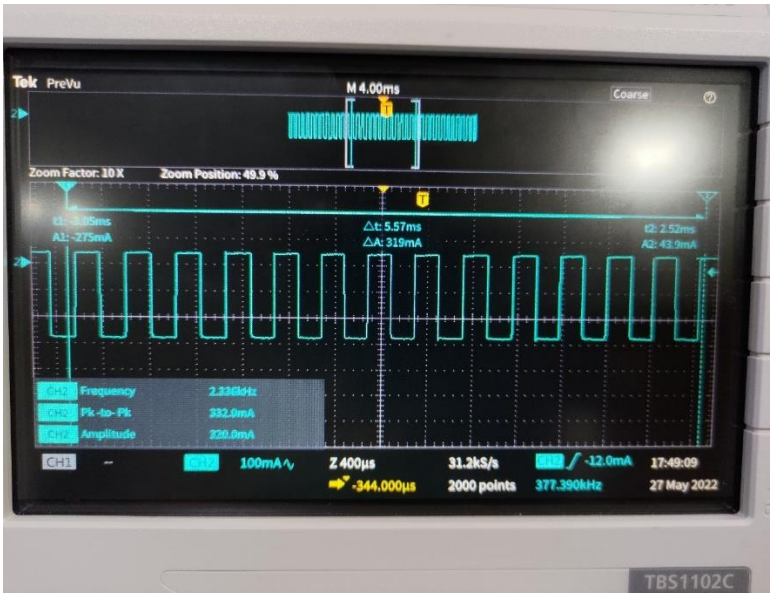
```
* This program configures TIM2 with prescaler divides
by 1600
 * and counter wraps around at 10000. So the 16 MHz
system clock
 * is divided by 1600 then divided by 10000 to become 1
Hz.
 * Every time the counter wraps around, it sets UIF
flag
 * (bit 0 of TIM2_SR register). The program waits for
the
 * UIF to set then toggles the LED (PA5).f=f/p*p F=1/t
 *
 *
 */
#include "stm32f4xx.h"

int main(void) {
    // configure PA5 as output to drive the LED
    RCC->AHB1ENR |=  1<<6;    /* enable GPIOA clock */

    GPIOG->MODER &= ~0x00003000;   /* clear pin mode */
    GPIOG->MODER |=  0x00001000;    /* set pin to
output mode */

    // configure TIM2 to wrap around at 1 Hz
    RCC->APB1ENR |= 1;              /* enable TIM2
clock */
    TIM2->PSC = 64 - 1;           /* divided by 1600 */
    TIM2->ARR = 5000 - 1;        /* divided by 10000 */

    TIM2->CNT = 0;            /* clear timer counter */

    TIM2->CR1 = 1;                  /* enable TIM2 */

    while (1) {
        while(!(TIM2->SR & 1)) {}   /* wait until UIF
set */
```

```
        TIM2->SR &= ~1;                /* clear UIF */

        GPIOG->ODR ^= 0x00000040;/* toggle green LED */
    }
}
```



Toggle Green LED (LD2) using SysTick

```
#include "stm32f4xx.h"

void delayMs(int n);

int main(void) {
 RCC->AHB1ENR |=  1<<6;/* enable GPIOA clock */

GPIOG->MODER &= ~0x00003000;/* clear pin mode*/

GPIOG->MODER |=  0x00001000;/* set pin to output

mode */

   /* Configure SysTick */


SysTick->LOAD = 0xFFFFFF;     /* reload with max

value */

   SysTick->CTRL = 5;     /* enable it, no
interrupt, use system clock */

   while (1) {
      /* take bit 23 of SysTick current value and shift it to bit 5
         then write it to PortA */
      GPIOG->ODR ^= (SysTick->VAL >> (23 - 6)) & 0x00000040;
   }
}
```

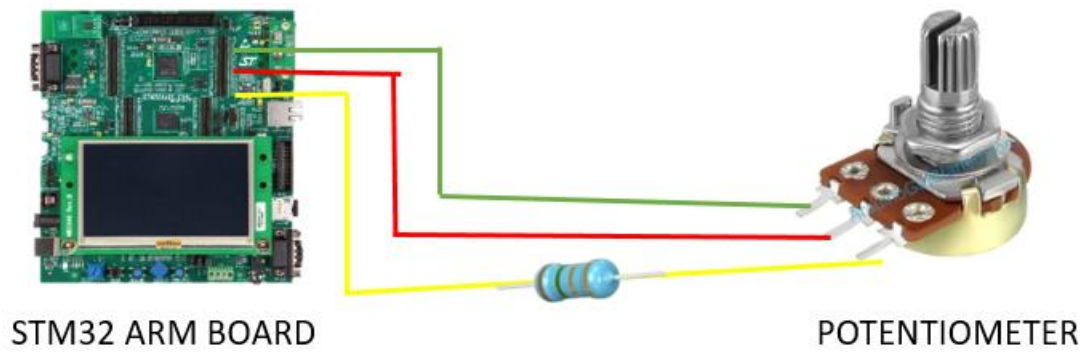


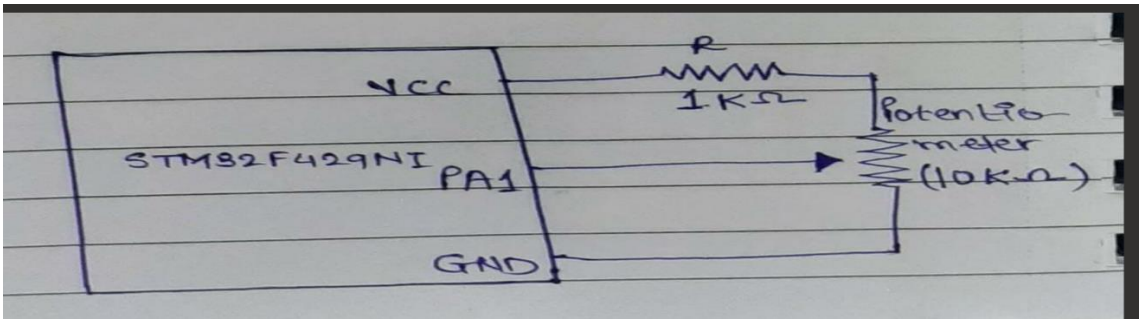
A to D conversion of channel 1

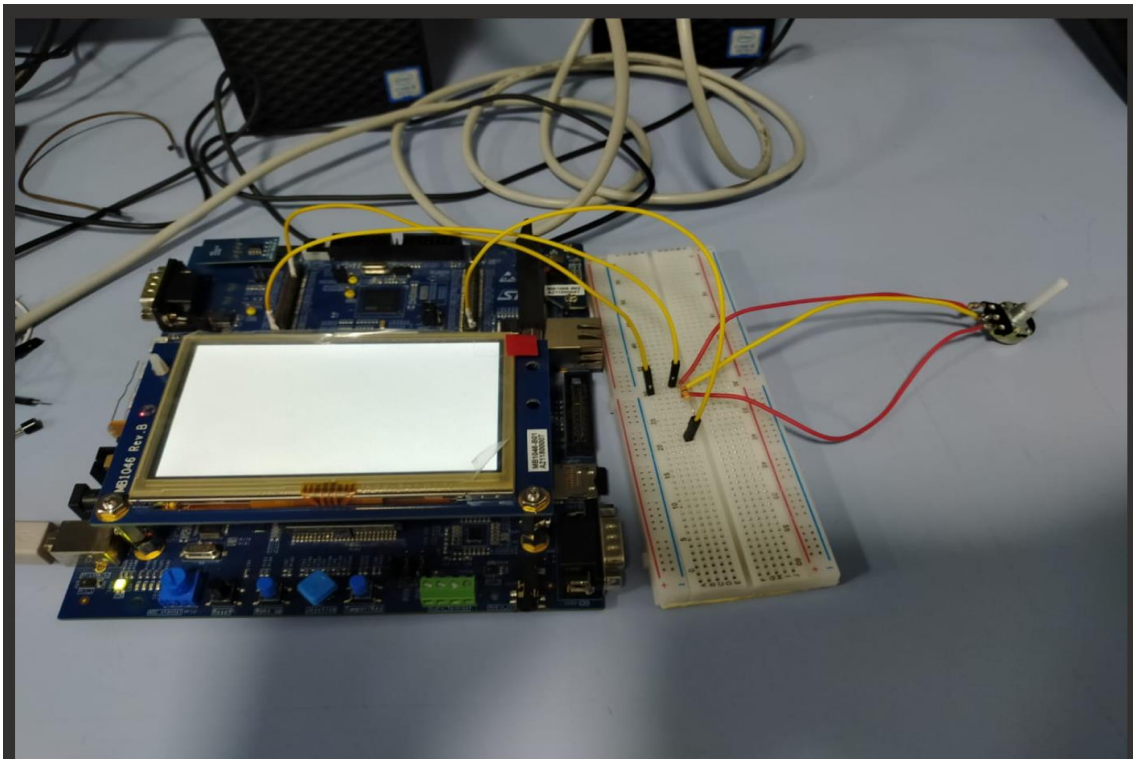
A) Potentiometer

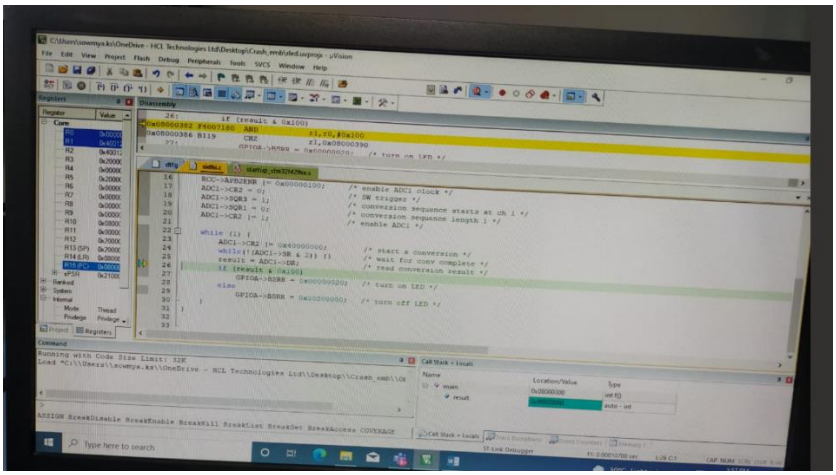
```
/* A to D conversion of channel 1

 * This program converts the analog input from channel 1.
 * Channel 1 is connected to input from PA1.
 * Clock prescaler is left at 0 (divided by 2) and Sampling
 * time is also left at default of 3 cycles.
 * Software trigger is used. The bit 8 of the conversion
 * result is used to turn on/off the LED0. For the full
 * scale of the reference voltage, the LED0 should be on
 * for 8 times.
 *
 *
 */

#include "stm32f4xx.h"

int main (void) {
    int result;

    /* set up pin PA5 for LED */
    RCC->AHB1ENR |=  1;                    /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000400;    /* set pin to output mode */

    /* set up pin PA1 for analog input */
    RCC->AHB1ENR |=  1;                    /* enable GPIOA clock */
    GPIOA->MODER |=  0xC;           /* PA1 analog */

    /* setup ADC1 */
    RCC->APB2ENR |= 0x00000100;     /* enable ADC1 clock */
    ADC1->CR2 = 0;                  /* SW trigger */
```

```
    ADC1->SQR3 = 1;                 /* conversion sequence starts at ch 1 */
    ADC1->SQR1 = 0;                 /* conversion sequence length 1 */
    ADC1->CR2 |= 1;                 /* enable ADC1 */

    while (1) {
        ADC1->CR2 |= 0x40000000;        /* start a conversion */
        while(!(ADC1->SR & 2)) {}       /* wait for conv complete */
        result = ADC1->DR;              /* read conversion result */
        if (result & 0x100)
            GPIOA->BSRR = 0x00000020;   /* turn on LED */
        else
            GPIOA->BSRR = 0x00200000;   /* turn off LED */
    }
}
```





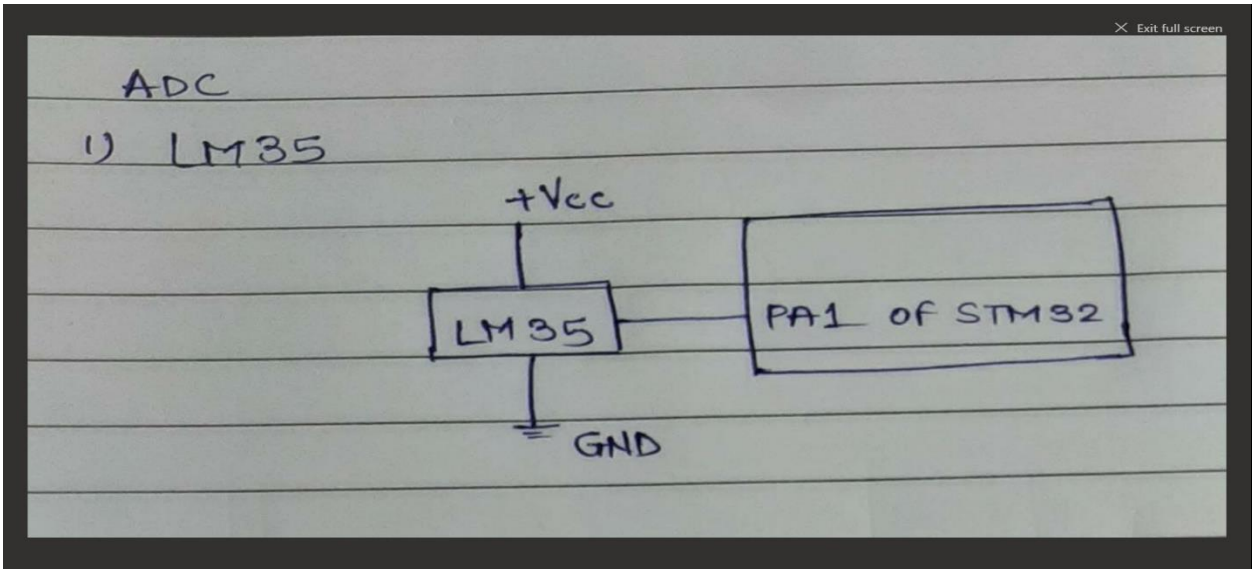


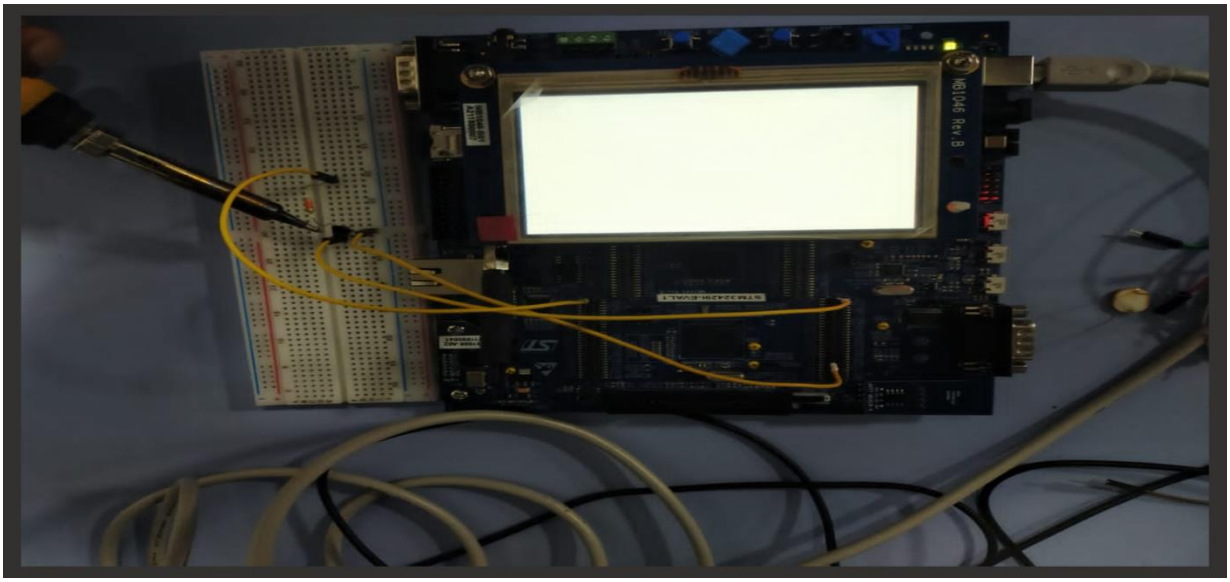


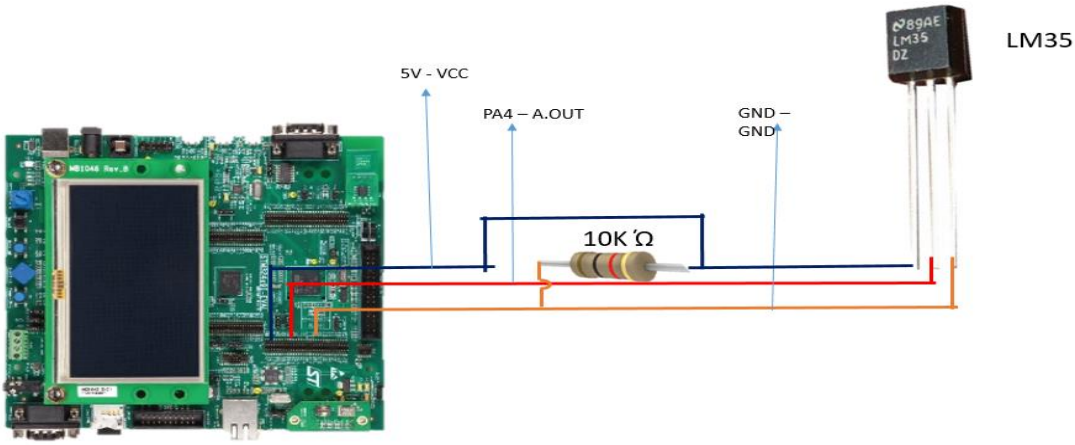
B) Temperature Sensor - Interfacing

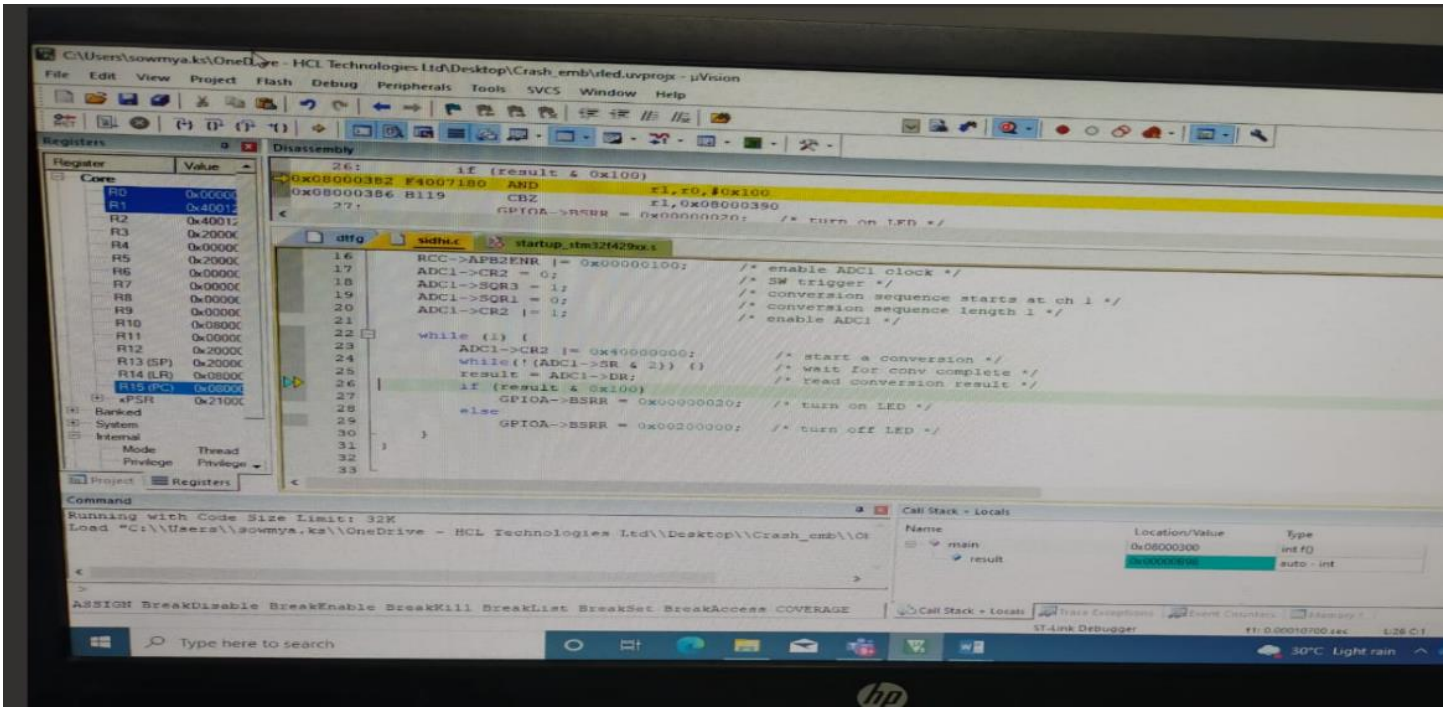
```
#include "stm32f4xx.h"

int main (void) {
    int result;
```

```
    /* set up pin PA5 for LED */
    RCC->AHB1ENR |=  1;                    /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000400;    /* set pin to output mode */

    /* set up pin PA1 for analog input */
    RCC->AHB1ENR |=  1;                    /* enable GPIOA clock */
    GPIOA->MODER |=  0xC;           /* PA1 analog */

    /* setup ADC1 */
    RCC->APB2ENR |= 0x00000100;     /* enable ADC1 clock */
    ADC1->CR2 = 0;                  /* SW trigger */
    ADC1->SQR3 = 1;                 /* conversion sequence starts at ch 1 */
    ADC1->SQR1 = 0;                 /* conversion sequence length 1 */
    ADC1->CR2 |= 1;                 /* enable ADC1 */

    while (1) {
        ADC1->CR2 |= 0x40000000;        /* start a conversion */
        while(!(ADC1->SR & 2)) {}       /* wait for conv complete */
        result = ADC1->DR;              /* read conversion result */
        if (result & 0x100)
            GPIOA->BSRR = 0x00000020;   /* turn on LED */
        else
            GPIOA->BSRR = 0x00200000;   /* turn off LED */
    }
}
```





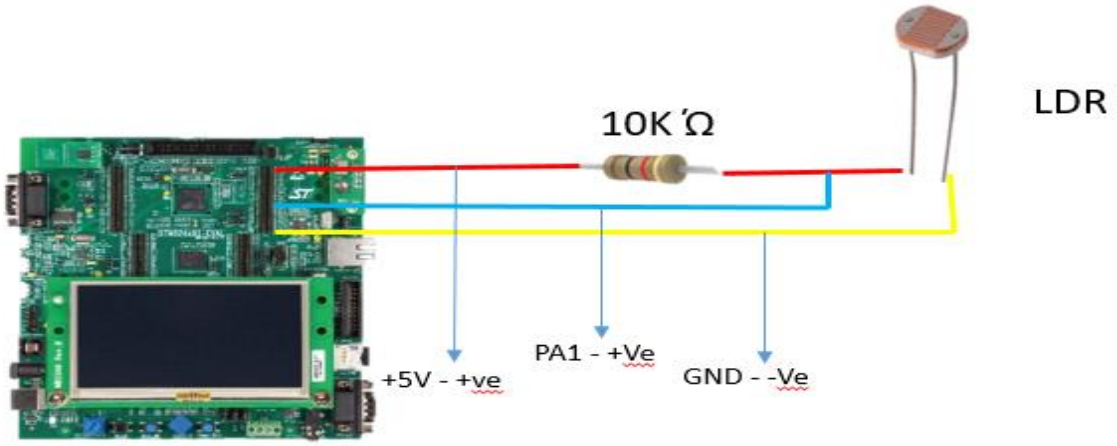

LM35
5V - VCC
PA4 – A.OUT
GND –
GND
10K Ω

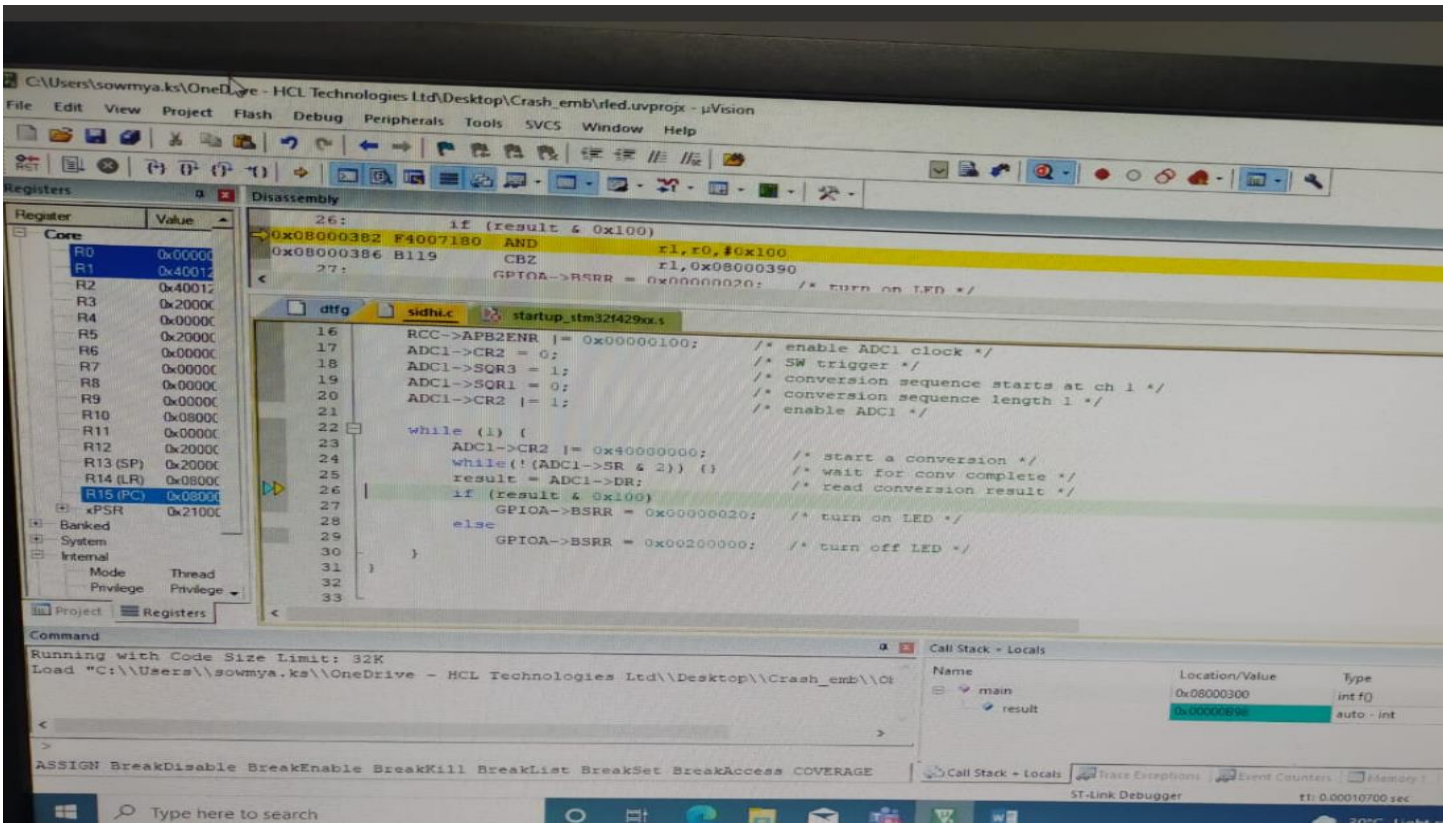


c) LDR Interfacing

```
#include "stm32f4xx.h"

int main (void) {
    int result;

    /* set up pin PA5 for LED */
    RCC->AHB1ENR |=  1;                     /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;     /* clear pin mode */
    GPIOA->MODER |=  0x00000400;     /* set pin to output mode */

    /* set up pin PA1 for analog input */
    RCC->AHB1ENR |=  1;                     /* enable GPIOA clock */
    GPIOA->MODER |=  0xC;            /* PA1 analog */

    /* setup ADC1 */
    RCC->APB2ENR |= 0x00000100;      /* enable ADC1 clock */
    ADC1->CR2 = 0;                   /* SW trigger */
    ADC1->SQR3 = 1;                  /* conversion sequence starts at ch 1 */
    ADC1->SQR1 = 0;                  /* conversion sequence length 1 */
    ADC1->CR2 |= 1;                  /* enable ADC1 */

    while (1) {
        ADC1->CR2 |= 0x40000000;         /* start a conversion */
        while(!(ADC1->SR & 2)) {}        /* wait for conv complete */
        result = ADC1->DR;               /* read conversion result */
        if (result & 0x100)
```

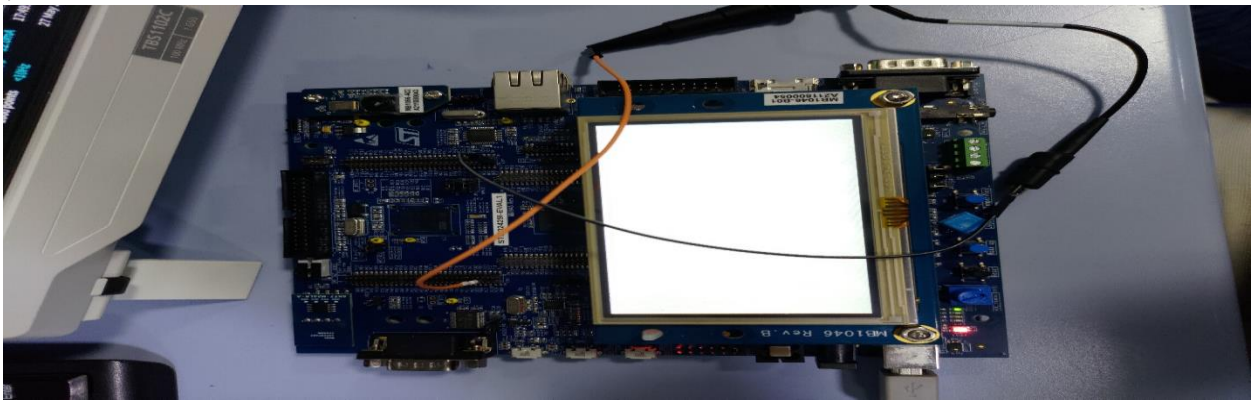
```
            GPIOA->BSRR = 0x00000020;    /* turn on LED */
        else
            GPIOA->BSRR = 0x00200000;    /* turn off LED */
    }
}
```

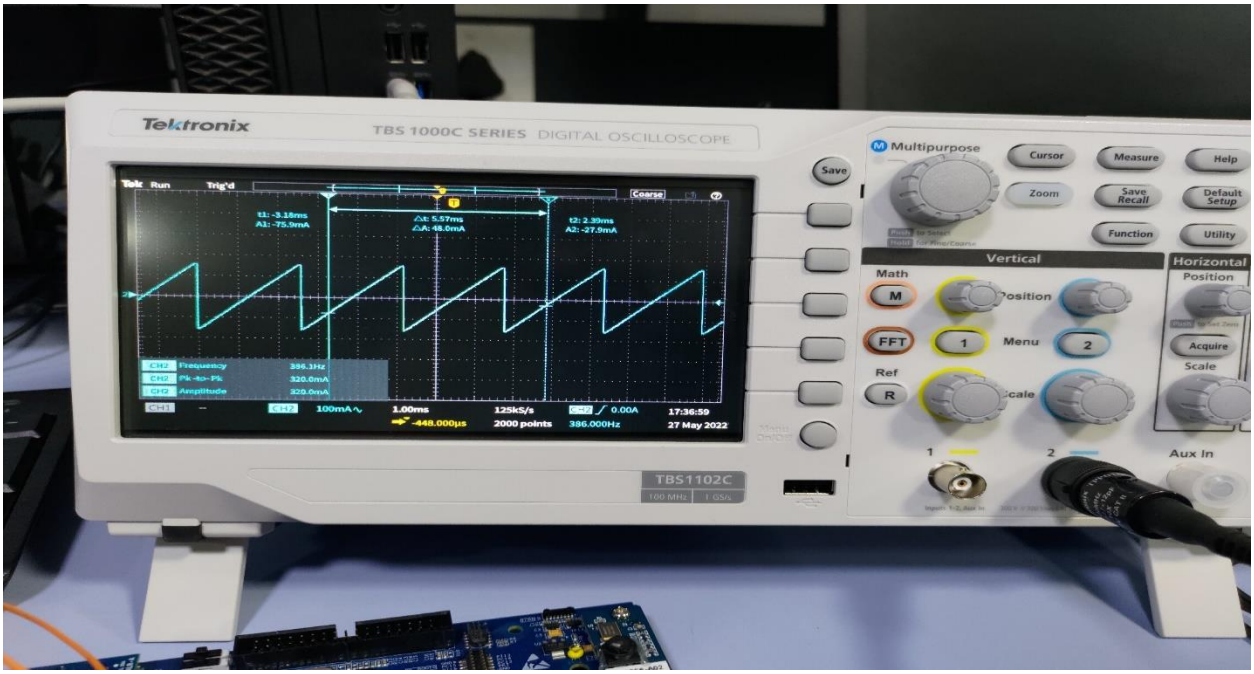




DAC to generate sawtooth waveform

```
/* Use DAC to generate sawtooth waveform
 * The DAC is initialized with no buffer or trigger, so every
 * write to the DAC data register will change the analog output.
 * The data is incremented in the loop and written to the DAC.
 * The output of DAC is on pin PA4.
 *
 *
 */
#include "stm32f4xx.h"

int main(void) {
    int data =0;
    RCC->AHB1ENR |=  1;                    /* enable GPIOA clock */
    GPIOA->MODER |=  0x00000300;    /* PA4 analog */
    /* setup DAC */
    RCC->APB1ENR |= 1 << 29;        /* enable DAC clock */
    DAC->CR |= 1;                   /* enable DAC */

    while(1) {
        DAC->DHR12R1 = data++ & 0x0FFF;
    }
}
```





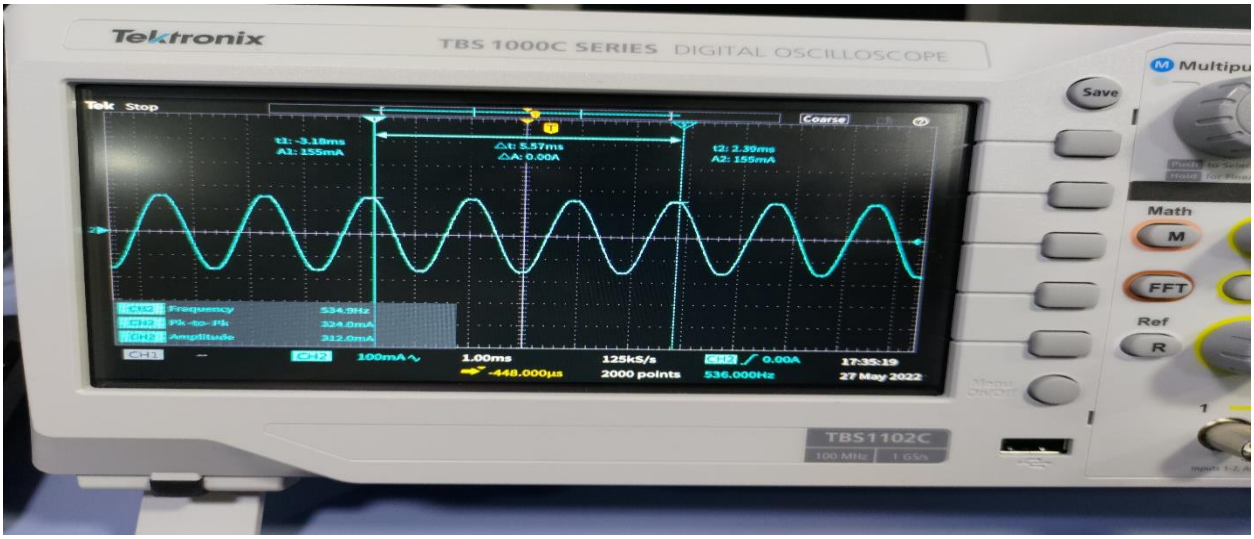
DAC to generate sine waveform

```
/* Use DAC to generate sine waveform
 * This program uses a pre-calculated lookup table to generate a
 * sine wave output through a DAC.
 * The DAC is initialized with no buffer or trigger, so every
 * write to the DAC data register will change the analog output.
 * The index of the lookup table is incremented in the loop and
 * the data is written to the DAC.
 * The output of DAC is on pin PA4.
```

```
#include "stm32f4xx.h"

void delayUs(int n);

int main(void) {
    int i;
    const static int sineWave[] =
        { 2048, 2149, 2250, 2350, 2450, 2549, 2646, 2742, 2837, 2929, 3020, 3108, 3193,
3275, 3355, 3431, 3504, 3574, 3639, 3701, 3759, 3812, 3861, 3906, 3946, 3982, 4013, 4039,
4060, 4076, 4087, 4094, 4095, 4091, 4082, 4069, 4050, 4026, 3998, 3965, 3927, 3884, 3837,
3786, 3730, 3671, 3607, 3539, 3468, 3394, 3316, 3235, 3151, 3064, 2975, 2883, 2790, 2695,
2598, 2500, 2400, 2300, 2199, 2098, 1997, 1896, 1795, 1695, 1595, 1497, 1400, 1305, 1212,
1120, 1031, 234, 283, 336, 394, 456, 521, 591, 664, 740, 820, 902, 987, 1075, 1166, 1258,944,
```

860, 779, 701, 627, 556, 488, 424, 365, 309, 258, 211, 168, 130, 97, 69, 45, 26, 13, 4, 0, 1, 8, 19, 35, 56, 82, 113, 149, 189,1353, 1449, 1546, 1645, 1745, 1845, 1946, 2047};

```
    RCC->AHB1ENR |=  1;         /* enable GPIOA clock */

    GPIOA->MODER |=  0x00000300;    /* PA4 analog */
    s
    /* setup DAC */
    RCC->APB1ENR |= 1 << 29;     /* enable DAC clock */

    DAC->CR |= 1;                   /* enable DAC */

    while(1) {
        for (i = 0; i < sizeof(sineWave)/sizeof(int);
i++)
        {
            DAC->DHR12R1 = sineWave[i]; /* write value
of sinewave to DAC */
            delayUs(10);
        }
    }
}

/* 16 MHz SYSCLK */
void delayUs(int n) {
    int i;
    for (; n > 0; n--)
      for (i = 0; i < 3; i++) ;
}
```


Tektronix
TBS 1000C SERIES DIGITAL OSCILLOSCOPE
Tek Stop
Coarse
t1: -3.18ms
A1: 155mA
Δt: 5.57ms
ΔA: 0.00A
t2: 2.39ms
A2: 155mA
Frequency 534.9Hz
Pk-to-Pk 324.0mA
Amplitude 312.0mA
CH1
CH2 100mA
1.00ms
-448.000µs
125kS/s
2000 points
0.00A
536.000Hz
17:35:19
27 May 2022
TBS1102C
100 MHz
1 GS/s
Save
Math
M
FFT
Ref
R

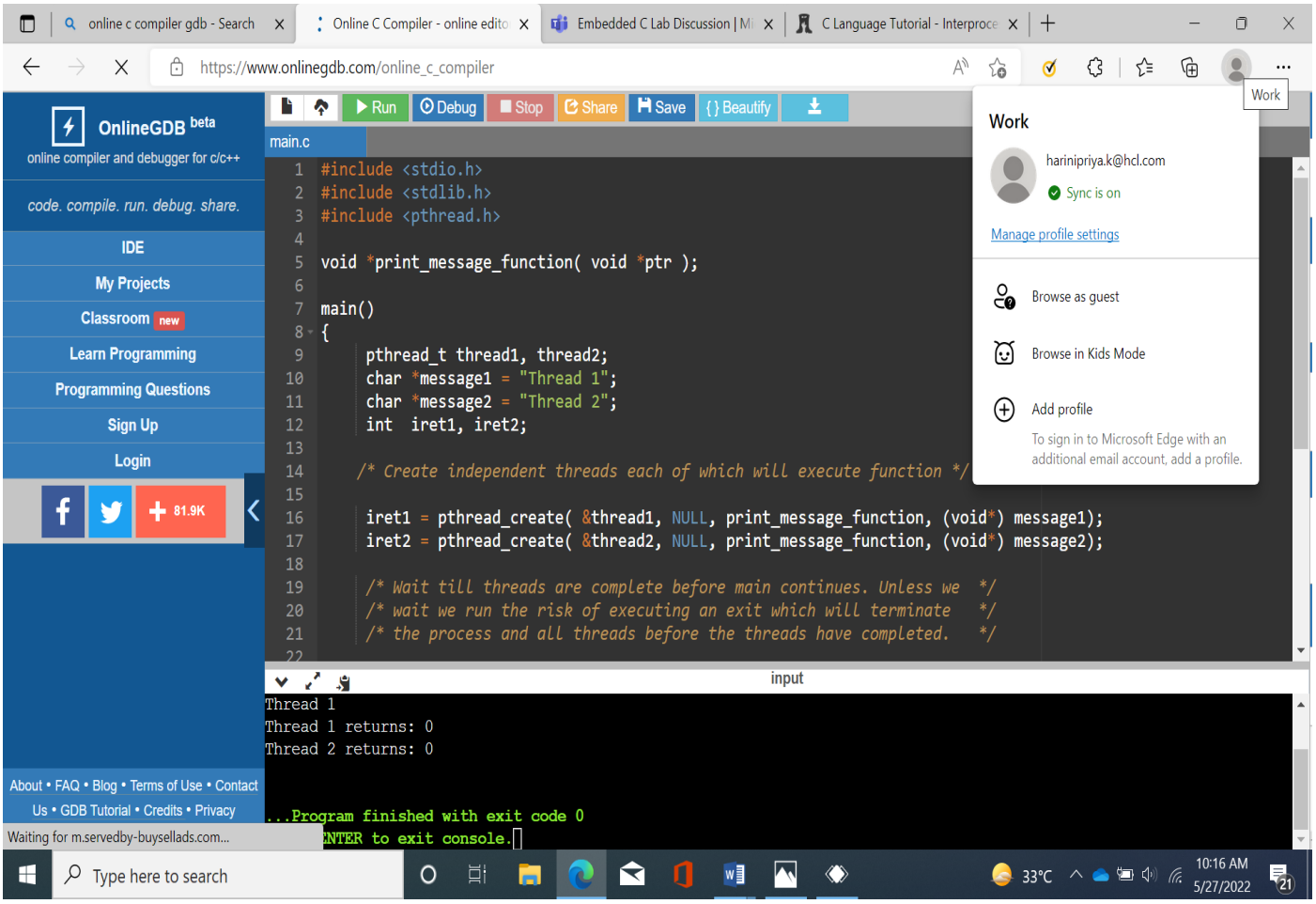
On-chip temperature sensor

```
/* On-chip temperature sensor
 * This program converts the on-chip temperature sensor output
 * and display the temperture in Celcius on the console.
 * Timer TIM2 is configured to generate 1 Hz output, which is
 * used to trigger the A/D conversion. The internal temperature
 * sensor is connected to channel 18 of ADC1 Master. The sensor
 * is turned on at bit 23 of ADC_CCR register.
 *
 * The conversion result is displayed as Celcius using the
 * formula and parameters in the datasheet on the console.
 * The console UART driver is from Program 4-5.
 */

#include "stm32f4xx.h"
#include <stdio.h>

void USART2_init(void);
int USART2_write(int c);

int main(void) {
    int data;
    double volt, temp;

    RCC->AHB1ENR |=  1;                     /* enable GPIOA clock */

    /* setup TIM2 */
    RCC->APB1ENR |= 1;              /* enable TIM2 clock */
    TIM2->PSC = 1600 - 1;           /* divided by 1600 */
    TIM2->ARR = 10000 - 1;          /* divided by 10000, sample at 1 Hz */
    TIM2->CNT = 0;
    TIM2->CCMR1 = 0x00006800;       /* pwm1 mode,  preload enable */
    TIM2->CCER = 0x0010;            /* ch2 enable */
    TIM2->CCR2 = 50 - 1;
    TIM2->CR1 = 1;                  /* enable timer2 */

    /* setup ADC1 */
    RCC->APB2ENR |= 0x00000100;     /* enable ADC1 clock */
    /* turn on the temp sensor */
    ADC->CCR |= 0x800000;
    ADC->CCR &= ~0x400000;          /* VBATE must be disabled for temp sensor
*/
    /* configure A to D converter */
    ADC1->SMPR1 = 0x4000000;        /* sampling time minimum 10 us */
    ADC1->SQR3 = 18;                /* ch18 - internal temp sensor, single
channel */
    ADC1->CR2 = 0x13000000;         /* trigger: EXTEN rising edge, EXTSEL 3 =
tim2.2 */
    ADC1->CR2 |= 1;                 /* enable ADC1 */

    /* initialize USART2 for output */
    USART2_init();
```

```
    printf("ADC internal temperature sensor \r\n");

    while(1) {
        while(!(ADC1->SR & 2)) {}
        data = ADC1->DR;
        /* Temperature (in °C) = {(VSENSE – V25) / Avg_Slope} + 25 */
        /* V25 = 0.76V, slope = 2.5 mV/C */
        volt = (double)data / 4095 * 3.3;   /* convert ADC output to voltage
*/
        temp = (volt - 0.76) / 0.0025 + 25; /* convert voltage to temperature
C */
        printf("%d, %.2f\370C\r\n", data, temp);    /* octal 370 is degree
sign in extended ASCII */
    }
}

/* initialize USART2 to transmit at 9600 Baud */
void USART2_init (void) {
    RCC->AHB1ENR |= 1;          /* Enable GPIOA clock */
    RCC->APB1ENR |= 0x20000;    /* Enable USART2 clock */

    /* Configure PA2 for USART2_TX */
    GPIOA->AFR[0] &= ~0x0F00;
    GPIOA->AFR[0] |=  0x0700;   /* alt7 for USART2 */
    GPIOA->MODER  &= ~0x0030;
    GPIOA->MODER  |=  0x0020;   /* enable alternate function for PA2 */

    USART2->BRR = 0x0683;       /* 9600 baud @ 16 MHz */
    USART2->CR1 = 0x000C;       /* enable Tx, Rx, 8-bit data */
    USART2->CR2 = 0x0000;       /* 1 stop bit */
    USART2->CR3 = 0x0000;       /* no flow control */
    USART2->CR1 |= 0x2000;      /* enable USART2 */
}

/* Write a character to USART2 */
int USART2_write (int ch) {
    while (!(USART2->SR & 0x0080)) {}   /* wait until Tx buffer empty */
    USART2->DR = ch;
    return ch;
}
```

Thread Creation

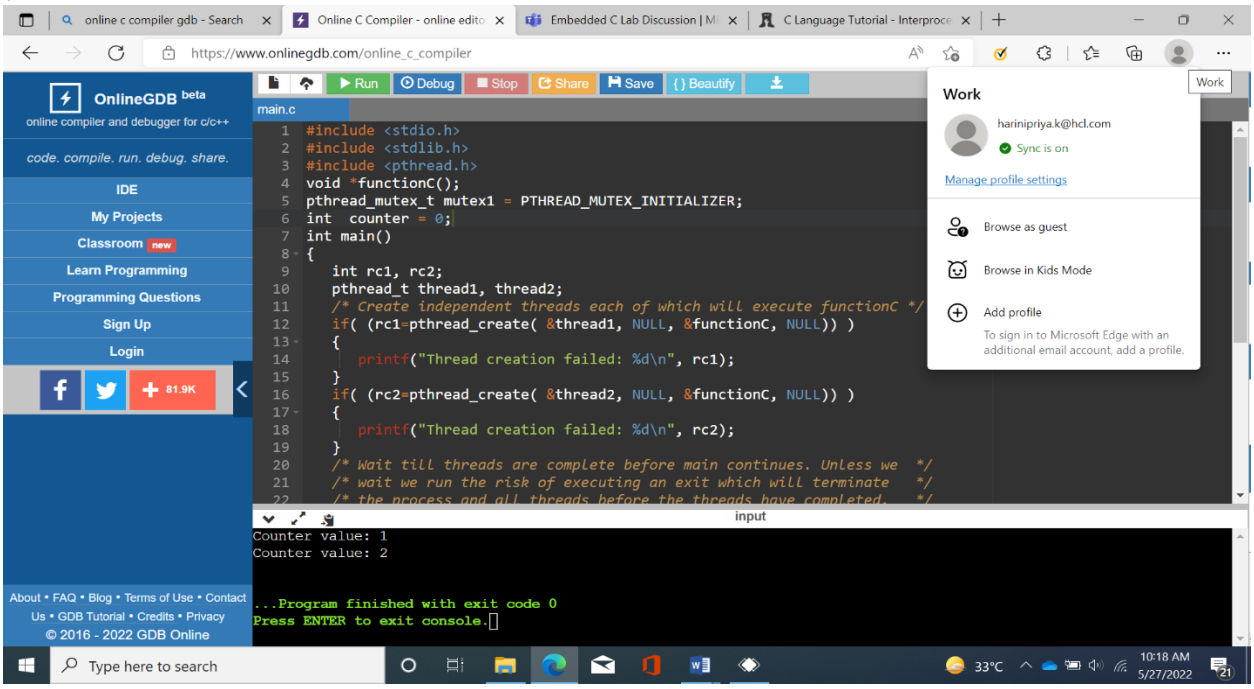
```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *print_message_function( void *ptr );
main()
{
     pthread_t thread1, thread2;
     char *message1 = "Thread 1";
     char *message2 = "Thread 2";
     int  iret1, iret2;

    /* Create independent threads each of which will execute function */

     iret1 = pthread_create( &thread1, NULL, print_message_function, (void*)
message1);
     iret2 = pthread_create( &thread2, NULL, print_message_function, (void*)
message2);

     /* Wait till threads are complete before main continues. Unless we  */
     /* wait we run the risk of executing an exit which will terminate   */
     /* the process and all threads before the threads have completed.   */

     pthread_join( thread1, NULL);
     pthread_join( thread2, NULL);

     printf("Thread 1 returns: %d\n",iret1);
     printf("Thread 2 returns: %d\n",iret2);
     exit(0);
}

void *print_message_function( void *ptr )
{
     char *message;
     message = (char *) ptr;
     printf("%s \n", message);
}
```



Mutex Semaphores

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *functionC();
pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;
int  counter = 0;
main()
{
   int rc1, rc2;
   pthread_t thread1, thread2;
   /* Create independent threads each of which will execute functionC */
   if( (rc1=pthread_create( &thread1, NULL, &functionC, NULL)) )
   {
      printf("Thread creation failed: %d\n", rc1);
   }
   if( (rc2=pthread_create( &thread2, NULL, &functionC, NULL)) )
   {
      printf("Thread creation failed: %d\n", rc2);
   }
   /* Wait till threads are complete before main continues. Unless we  */
```

```
   /* wait we run the risk of executing an exit which will terminate   */
   /* the process and all threads before the threads have completed.   */
   pthread_join( thread1, NULL);
   pthread_join( thread2, NULL);
   exit(0);
}
void *functionC()
{
   pthread_mutex_lock( &mutex1 );
   counter++;
   printf("Counter value: %d\n",counter);
   pthread_mutex_unlock( &mutex1 );
}
```

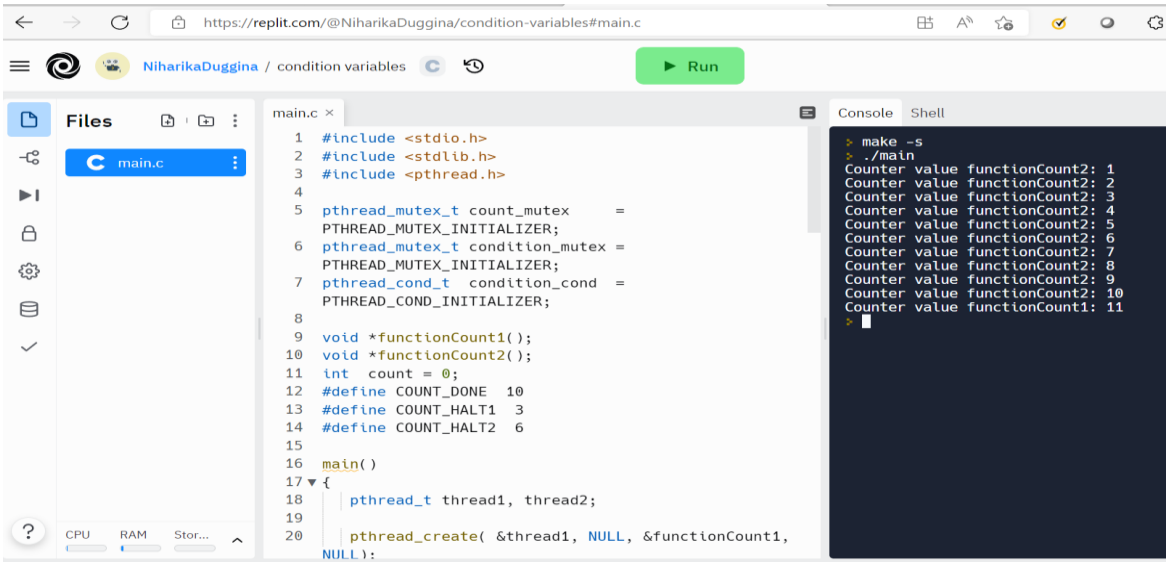


Experiment 11: Condition Variables

```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>


pthread_mutex_t count_mutex     = PTHREAD_MUTEX_INITIALIZER;

pthread_mutex_t condition_mutex = PTHREAD_MUTEX_INITIALIZER;

pthread_cond_t  condition_cond  = PTHREAD_COND_INITIALIZER;


void *functionCount1();
```

```
void *functionCount2();
int  count = 0;
#define COUNT_DONE  10
#define COUNT_HALT1  3
#define COUNT_HALT2  6

main()
{
   pthread_t thread1, thread2;

   pthread_create( &thread1, NULL, &functionCount1, NULL);
   pthread_create( &thread2, NULL, &functionCount2, NULL);
   pthread_join( thread1, NULL);
   pthread_join( thread2, NULL);

   exit(0);
}

void *functionCount1()
{
   for(;;)
   {
      pthread_mutex_lock( &condition_mutex );
      while( count >= COUNT_HALT1 && count <= COUNT_HALT2 )
      {
         pthread_cond_wait( &condition_cond, &condition_mutex );
      }
      pthread_mutex_unlock( &condition_mutex );

      pthread_mutex_lock( &count_mutex );
      count++;
```

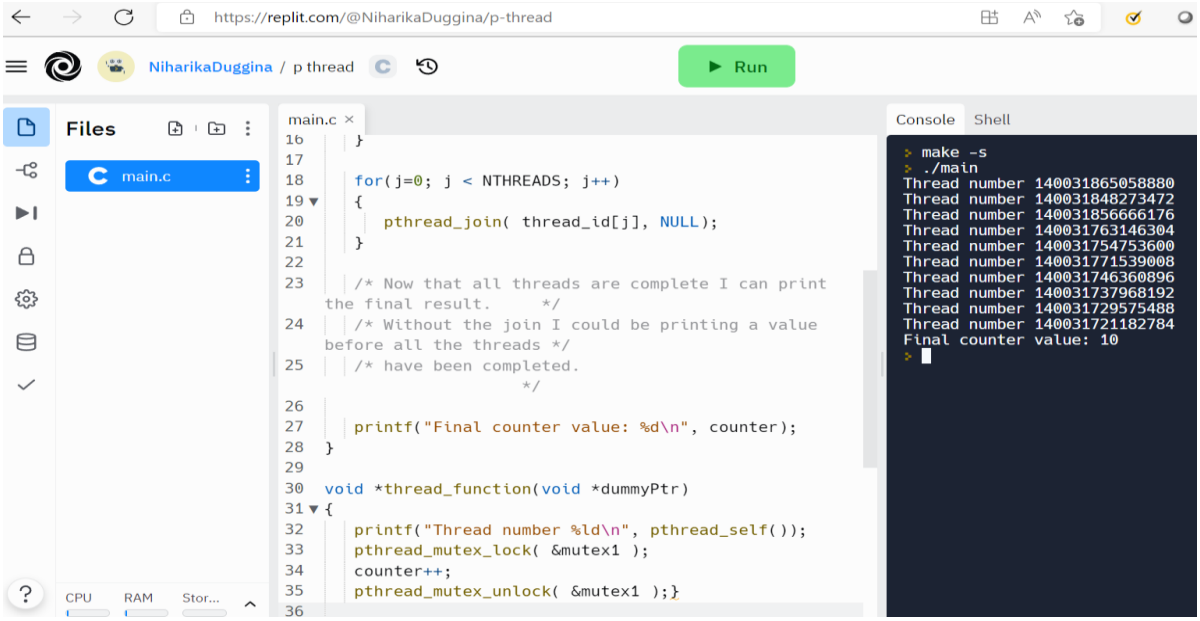
```
      printf("Counter value functionCount1: %d\n",count);
      pthread_mutex_unlock( &count_mutex );

      if(count >= COUNT_DONE) return(NULL);
    }
}

void *functionCount2()
{
    for(;;)
    {
       pthread_mutex_lock( &condition_mutex );
       if( count < COUNT_HALT1 || count > COUNT_HALT2 )
       {
          pthread_cond_signal( &condition_cond );
       }
       pthread_mutex_unlock( &condition_mutex );

       pthread_mutex_lock( &count_mutex );
       count++;
       printf("Counter value functionCount2: %d\n",count);
       pthread_mutex_unlock( &count_mutex );

       if(count >= COUNT_DONE) return(NULL);
    }

}
```



Experiment 12: P-thread

```
#include <stdio.h>

#include <pthread.h>

#define NTHREADS 10

void *thread_function(void *);

pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;

int  counter = 0;


main()

{

   pthread_t thread_id[NTHREADS];

   int i, j;


   for(i=0; i < NTHREADS; i++)

   {

      pthread_create( &thread_id[i], NULL, thread_function, NULL );

   }


   for(j=0; j < NTHREADS; j++)

   {
```

```
        pthread_join( thread_id[j], NULL);
    }

    /* Now that all threads are complete I can print the final result.     */
    /* Without the join I could be printing a value before all the threads */
    /* have been completed.                                                */

    printf("Final counter value: %d\n", counter);
}

void *thread_function(void *dummyPtr)
{
    printf("Thread number %ld\n", pthread_self());
    pthread_mutex_lock( &mutex1 );
    counter++;
    pthread_mutex_unlock( &mutex1 );}
```



Send a string "YES" to USART2

```
/* Send a string "YES" to USART2
 *
 * USART2 is connected to the ST-Link virtual COM port.
 * Use Tera Term to see the message "YES" on a PC.
 *
 * By default, the clock is running at 16 MHz.
 * The UART2 is configured for 9600 Baud.
 * PA2 - USART2 TX (AF7)
 *

#include "stm32F4xx.h"

void USART2_init(void);
void USART2_write(int c);
void delayMs(int);

/*----------------------------------------------------------------------------
-
  MAIN function
 *----------------------------------------------------------------------------
-*/
int main (void) {
    USART2_init();          /* initialize USART2 */

    while(1) {              /* Loop forever */
        USART2_write('Y');
        USART2_write('e');
        USART2_write('s');
        delayMs(10);        /* leave a gap between messages */
    }
}

/*----------------------------------------------------------------------------
-
  Initialize UART pins, Baudrate
 *----------------------------------------------------------------------------
-*/
void USART2_init (void) {
    RCC->AHB1ENR |= 1;          /* Enable GPIOA clock */
    RCC->APB1ENR |= 0x20000;    /* Enable USART2 clock */

    /* Configure PA2 for USART2_TX */
    GPIOA->AFR[0] &= ~0x0F00;
    GPIOA->AFR[0] |=  0x0700;   /* alt7 for USART2 */
    GPIOA->MODER  &= ~0x0030;
    GPIOA->MODER  |=  0x0020;   /* enable alternate function for PA2 */

    USART2->BRR = 0x0683;       /* 9600 baud @ 16 MHz */
    USART2->CR1 = 0x0008;       /* enable Tx, 8-bit data */
    USART2->CR2 = 0x0000;       /* 1 stop bit */
```

```
    USART2->CR3 = 0x0000;       /* no flow control */
    USART2->CR1 |= 0x2000;      /* enable USART2 */
}


/* Write a character to USART2 */
void USART2_write (int ch) {
    while (!(USART2->SR & 0x0080)) {}   // wait until Tx buffer empty
    USART2->DR = (ch & 0xFF);
}

void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 2000; i++) ;
}
```

USART2 Receive at 115200 Baud

```
/* p4_2.c UART on USART2 Receive at 115200 Baud
 *
 * Receive key strokes from terminal emulator (TeraTerm) of the
 * host PC to the USART2 of the Nucleo-F446RE board.
 * USART2 is connected to the ST-Link virtual COM port.
 * Launch Tera Term on a PC and hit any key.
 * The LED program from P3_5 of Chapter 3 is used to blink the green LED
 * according to the key received.
 * You need to wait till the blinking stops before hitting another key.
 * Received key is not echoed back to the terminal, so you will not see
 * the character displayed.
 *
 * By default, the clock is running at 16 MHz.
 * The UART2 is configured for 115200 Baud.
 * PA3 - USART2 RX (AF7)
 *

 */
#include "stm32F4xx.h"

void USART2_init(void);
char USART2_read(void);
void LED_blink(int value);
void delayMs(int);

/*-----------------------------------------------------------------------------
-
  MAIN function
 *-----------------------------------------------------------------------------
-*/
int main (void) {
    char c;

    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000400;    /* set pin to output mode */
```

```
    USART2_init();                  /* initialize USART2 */

    while(1) {                      /* Loop forever */
        c = USART2_read();          /* wait for a character received */
        LED_blink(c);               /* blink the LED */
    }
}

/*-----------------------------------------------------------------------------
-
  Initialize UART pins, Baudrate
 *-----------------------------------------------------------------------------
-*/
void USART2_init (void) {
    RCC->AHB1ENR |= 1;          /* Enable GPIOA clock */
    RCC->APB1ENR |= 0x20000;    /* Enable USART2 clock */

    /* Configure PA3 for USART2 RX */
    GPIOA->AFR[0] &= ~0xF000;
    GPIOA->AFR[0] |=  0x7000;   /* alt7 for USART2 */
    GPIOA->MODER  &= ~0x00C0;
    GPIOA->MODER  |=  0x0080;   /* enable alternate function for PA3 */

    USART2->BRR = 0x008B;       /* 115200 baud @ 16 MHz */
    USART2->CR1 = 0x0004;       /* enable Rx, 8-bit data */
    USART2->CR2 = 0x0000;       /* 1 stop bit */
    USART2->CR3 = 0x0000;       /* no flow control */
    USART2->CR1 |= 0x2000;      /* enable USART2 */
}

/* Read a character from USART2 */
char USART2_read(void) {
    while (!(USART2->SR & 0x0020)) {}   // wait until char arrives
    return USART2->DR;
}

/* turn on or off the LEDs according to the value */
void LED_blink(int value) {
    value %= 16;                    /* cap the max count at 15 */

    for (; value > 0; value--) {
        GPIOA->BSRR = 0x00000020;   /* turn on LED */
        delayMs(200);
        GPIOA->BSRR = 0x00200000;   /* turn off LED */
        delayMs(200);
    }
    delayMs(800);
}

void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 2000; i++) ;
```

USART:

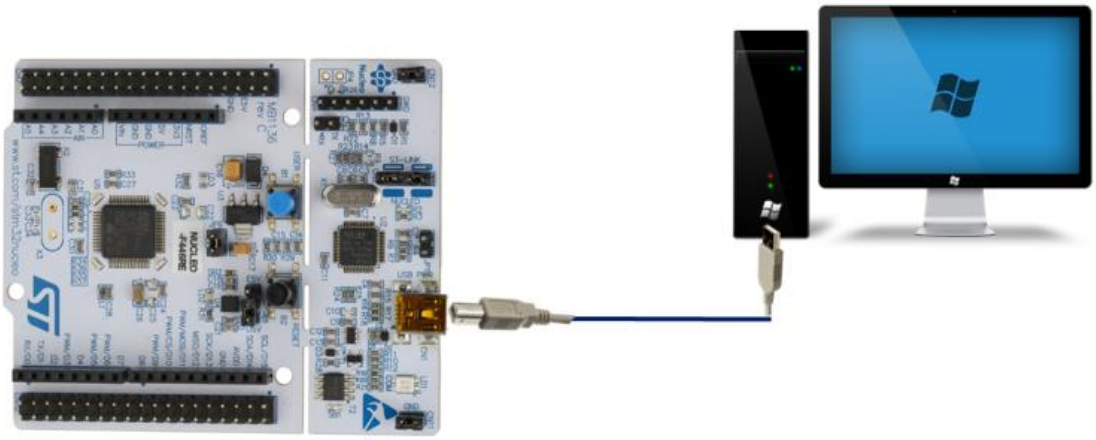
Algorithm:

- Start to write the program for USART in STM32 in Keil µvision 5
- Open the Arduino IDE. Go to the tools.
- Select the Serial Port -> **COM1 (or) COM4**
- Next Go to the Serial Monitor and Select the band width'**115200**'
- Step by Step running method has to be followed.
- By applying 2 bits, it will stop after receiving the all bits at receiver end.

Send a string "YES" to UART4

```
/* Send a string "YES" to UART4
 *
 * UART4 Tx signal is connected to pin PA0. To see the output of UART4
 * on a PC, you need to use a USB-serial module. Connect the Rx pin of
 * the module to the PA0 pin of the STM32F446 Nucleo board. Make sure
 * the USB-serial module you use has a 3.3V interface.
 *
 * By default, the clock is running at 16 MHz.
 * The UART is configured for 9600 Baud.
 * PA0 - UART4 TX (AF8)
 *

 */
#include "stm32F4xx.h"

void UART4_init(void);
void UART4_write(int c);
void delayMs(int);

/*----------------------------------------------------------------------------
-
  MAIN function
 *----------------------------------------------------------------------------
-*/
int main (void) {
    char message[] = "Hello\r\n";
    int i;

    UART4_init();

    while (1) {
        for (i = 0; i < 7; i++) {
            UART4_write(message[i]); /* send a char */
        }
        delayMs(10);                /* leave a gap between messages */
    }
}

/*----------------------------------------------------------------------------
-
```

```
  Initialize UART pins, Baudrate
 *-----------------------------------------------------------------------------
-*/
void UART4_init (void) {
    RCC->AHB1ENR |= 1;          /* Enable GPIOA clock */
    RCC->APB1ENR |= 0x80000;    /* Enable UART4 clock */

    /* Configure PA0 for UART4 TX */
    GPIOA->AFR[0] &= ~0x000F;
    GPIOA->AFR[0] |=  0x0008;   /* alt8 for UART4 */
    GPIOA->MODER  &= ~0x0003;
    GPIOA->MODER  |=  0x0002;   /* enable alternate function for PA0 */

    UART4->BRR = 0x0683;       /* 9600 baud @ 16 MHz */
    UART4->CR1 = 0x0008;       /* enable Tx, 8-bit data */
    UART4->CR2 = 0x0000;       /* 1 stop bit */
    UART4->CR3 = 0x0000;       /* no flow control */
    UART4->CR1 |= 0x2000;      /* enable UART4 */
}


/* Write a character to UART4 */
void UART4_write (int ch) {
    while (!(UART4->SR & 0x0080)) {}   // wait until Tx buffer empty
    UART4->DR = (ch & 0xFF);
}

void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 2000; i++) ;
}
```



UART:

Algorithm:

- Start to write the program for UART in STM32 in Keil µvision 5
- Open the Arduino IDE. Go to the tools.
- Select the Serial Port -> **COM1 (or) COM4**
- Next Go to the Serial Monitor and Select the band width**'9600**'
- Step by Step running method has to be followed.
- By applying 2 bits, it will stop after receiving the all bits at receiver end.

Initialize and display "hello" on the LCD

```
#include "stm32f4xx.h"

#define RS 0x20     /* PB5 mask for reg select */
#define RW 0x40     /* PB6 mask for read/write */
#define EN 0x80     /* PB7 mask for enable */

void delayMs(int n);
void LCD_command(unsigned char command);
void LCD_data(char data);
void LCD_init(void);
void PORTS_init(void);

int main(void) {
    /* initialize LCD controller */
    LCD_init();

    while(1) {
        /* Write "hello" on LCD */
        LCD_data('h');
        LCD_data('e');
        LCD_data('l');
        LCD_data('l');
        LCD_data('o');
        delayMs(1000);

        /* clear LCD display */
        LCD_command(1);
        delayMs(500);
    }
}

/* initialize port pins then initialize LCD controller */
void LCD_init(void) {
    PORTS_init();

    delayMs(30);            /* initialization sequence */
    LCD_command(0x30);
    delayMs(10);
    LCD_command(0x30);
    delayMs(1);
    LCD_command(0x30);

    LCD_command(0x38);      /* set 8-bit data, 2-line, 5x7 font */
    LCD_command(0x06);      /* move cursor right after each char */
    LCD_command(0x01);      /* clear screen, move cursor to home */
    LCD_command(0x0F);      /* turn on display, cursor blinking */
}

void PORTS_init(void) {
    RCC->AHB1ENR |=  0x06;          /* enable GPIOB/C clock */

    /* PB5 for LCD R/S */
```

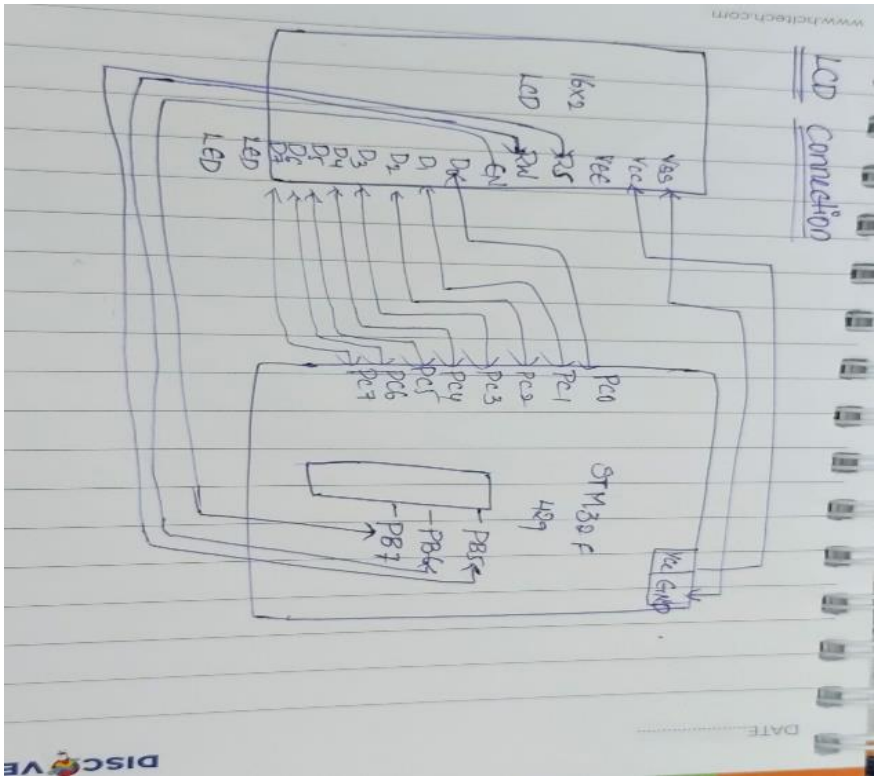
```
    /* PB6 for LCD R/W */
    /* PB7 for LCD EN */
    GPIOB->MODER &= ~0x0000FC00;    /* clear pin mode */
    GPIOB->MODER |=  0x00005400;    /* set pin output mode */
    GPIOB->BSRR = 0x00C00000;       /* turn off EN and R/W */

    /* PC0-PC7 for LCD D0-D7, respectively. */
    GPIOC->MODER &= ~0x0000FFFF;    /* clear pin mode */
    GPIOC->MODER |=  0x00005555;    /* set pin output mode */
}

void LCD_command(unsigned char command) {
    GPIOB->BSRR = (RS | RW) << 16;  /* RS = 0, R/W = 0 */
    GPIOC->ODR = command;           /* put command on data bus */
    GPIOB->BSRR = EN;               /* pulse E high */
    delayMs(0);
    GPIOB->BSRR = EN << 16;         /* clear E */

    if (command < 4)
        delayMs(2);         /* command 1 and 2 needs up to 1.64ms */
    else
        delayMs(1);         /* all others 40 us */
}

void LCD_data(char data) {
    GPIOB->BSRR = RS;               /* RS = 1 */
    GPIOB->BSRR = RW << 16;         /* R/W = 0 */
    GPIOC->ODR = data;              /* put data on data bus */
    GPIOB->BSRR = EN;               /* pulse E high */
    delayMs(0);
    GPIOB->BSRR = EN << 16;         /* clear E */

    delayMs(1);
}

/* delay n milliseconds (16 MHz CPU clock) */
void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 3195; i++) ;
}
```

LCD:

Algorithm:

- Start to write the program for LCD in STM32 in Keil µvision 5
- Connect the D0 pin in LCD to PC0 pin in STM32
- Connect the D1 pin in LCD to PC1 pin in STM32
- Connect the D2 pin in LCD to PC2 pin in STM32
- Connect the D3 pin in LCD to PC3 pin in STM32
- Connect the D4 pin in LCD to PC4 pin in STM32
- Connect the D5 pin in LCD to PC5 pin in STM32
- Connect the D6 pin in LCD to PC6 pin in STM32

- Connect the D7 pin in LCD to PC7 pin in STM32
- Connect the RS pin in LCD to PB5 pin in STM32
- Connect the RW pin in LCD to PB6 pin in STM32
- Connect the EN pin in LCD to PB7 pin in STM32
- Connect the Vcc pin in LCD to Vcc pin in STM32
- Connect the GND pin in LCD to GND pin in STM32
- By running, we can observe the Output.



Toggle Green LED (LD2) on STM32F446RE Nucleo64 using TIM2

```
/* Toggle Green LED (LD2) on STM32F446RE Nucleo64 using TIM3
 *
 * This program configures TIM3 with prescaler divides by 1600
 * and counter wraps around at 10000. So the 16 MHz system clock
 * is divided by 1600 then divided by 10000 to become 1 Hz.
 * Every time the counter wraps around, it sets UIF flag
 * (bit 0 of TIM3_SR register). The program waits for the
 * UIF to set then toggles the LED (PA5).
 *
 *
 */
#include "stm32f4xx.h"

int main(void) {
    // configure PA5 as output to drive the LED
    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000400;    /* set pin to output mode */

    // configure TIM3 to wrap around at 1 Hz
    RCC->APB1ENR |= 2;              /* enable TIM3 clock */
    TIM3->PSC = 1600 - 1;           /* divided by 1600 */
    TIM3->ARR = 10000 - 1;          /* divided by 10000 */
    TIM3->CNT = 0;                  /* clear timer counter */
    TIM3->CR1 = 1;                  /* enable TIM3 */

    while (1) {
        while(!(TIM3->SR & 1)) {}   /* wait until UIF set */
        TIM3->SR &= ~1;             /* clear UIF */
        GPIOA->ODR ^= 0x00000020;   /* toggle green LED */
    }
}
```

Toggle Green LED using TIM2 Compare mode

```
/*  Toggle Green LED using TIM2 Compare mode
 *
 * This program configures TIM2 with prescaler divides by 1600
 * and counter wraps around at 10000. So the 16 MHz system clock
 * is divided by 1600 then divided by 10000 to become 1 Hz.
 * The channel 1 is configured for compare mode to toggle the output
 * pin every time the timer counter matches the CCR1 register.
 * The output pin of TIM2 CH1 is PA5 and the alternate function
 * of PA5 should be set to AF1.
 *

 */
#include "stm32f4xx.h"

int main(void) {
    // configure PA5 as output to drive the LED
    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000800;    /* set pin to alternate function */
    GPIOA->AFR[0] &= 0x00F00000;    /* clear pin AF bits */
    GPIOA->AFR[0] |= 0x00100000;    /* set pin to AF1 for TIM2 CH1 */

    // configure TIM2 to wrap around at 1 Hz
        // and toggle CH1 output when the counter value is 0
    RCC->APB1ENR |= 1;              /* enable TIM2 clock */
    TIM2->PSC = 1600 - 1;           /* divided by 1600 */
    TIM2->ARR = 10000 - 1;          /* divided by 10000 */
    TIM2->CCMR1 = 0x30;             /* set output to toggle on match */
    TIM2->CCR1 = 0;                 /* set match value */
    TIM2->CCER |= 1;                /* enable CH1 compare mode */
    TIM2->CNT = 0;                  /* clear counter */
    TIM2->CR1 = 1;                  /* enable TIM2 */

    while (1) {
    }
}
```

TIM3 Input Capture

```
/* TIM3 Input Capture

 * This program configures TIM2 CH1 to toggle PA5 as was done in p5_6.
 * The TIM3 CH1 is set to do Input Capture from PA6. The program waits
 * for capture flag (CC1IF) to set then stores the captured value in a
 * global variable to be watched by the debugger. The flag is cleared
 * by reading CCR1.
 * A jumper should be used to connect PA5 to PA6.
 * /

#include "stm32f4xx.h"

int timestamp = 0;
int hold = 1;

int main(void) {
    // configure PA5 as output of TIM2 CH1
    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000800;    /* set pin to alternate function */
    GPIOA->AFR[0] &= ~0x00F00000;    /* clear pin AF bits */
    GPIOA->AFR[0] |= 0x00100000;    /* set pin to AF1 for TIM2 CH1 */

    // configure TIM2 to wrap around at 1 Hz
        // and toggle CH1 output when the counter value is 0
    RCC->APB1ENR |= 1;              /* enable TIM2 clock */
    TIM2->PSC = 1600 - 1;           /* divided by 1600 */
    TIM2->ARR = 10000 - 1;          /* divided by 10000 */
    TIM2->CCMR1 = 0x30;             /* set output to toggle on match */
    TIM2->CCR1 = 0;                 /* set match value */
    TIM2->CCER |= 1;                /* enable ch 1 compare mode */
    TIM2->CNT = 0;                  /* clear counter */
    TIM2->CR1 = 1;                  /* enable TIM2 */

    // configure PA6 as input of TIM3 CH1
    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00003000;    /* clear pin mode */
    GPIOA->MODER |=  0x00002000;    /* set pin to alternate function */
    GPIOA->AFR[0] &= ~0x0F000000;   /* clear pin AF bits */
    GPIOA->AFR[0] |= 0x02000000;    /* set pin to AF2 for TIM3 CH1 */

    // configure TIM3 to do input capture with prescaler ...
    RCC->APB1ENR |= 2;              /* enable TIM3 clock */
    TIM3->PSC = 16000 - 1;          /* divided by 16000 */
    TIM3->CCMR1 = 0x41;             /* set CH1 to capture at every edge */
    TIM3->CCER = 1;                 /* enable CH 1 capture rising edge */
    TIM3->CR1 = 1;                  /* enable TIM3 */

    while (1) {
        while (!(TIM3->SR & 2)) {}  /* wait until input edge is captured */
        timestamp = TIM3->CCR1;     /* read captured counter value */
    }
```

```
}
```

TIM3 Measuring Period and Frequency

```
/* TIM3 Measuring Period and Frequency

 * This program configures TIM2 CH1 to toggle PA5 as was done in p5_6.
 * The TIM3 CH1 is set to do Input Capture from PA6. The program waits
 * for capture flag (CC1IF) to set then calculates the period and
 * frequency of the input signal.
 * A jumper should be used to connect PA5 to PA6.
 *

#include "stm32f4xx.h"

int period;
float frequency;

int main(void) {
    int last = 0;
    int current;

    // configure PA5 as output of TIM2 CH1
    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000800;    /* set pin to alternate function */
    GPIOA->AFR[0] &= ~0x00F00000;   /* clear pin AF bits */
    GPIOA->AFR[0] |= 0x00100000;    /* set pin to AF1 for TIM2 CH1 */

    // configure TIM2 to wrap around at 1 Hz
        // and toggle CH1 output when the counter value is 0
    RCC->APB1ENR |= 1;              /* enable TIM2 clock */
    TIM2->PSC = 1600 - 1;           /* divided by 1600 */
    TIM2->ARR = 3000 - 1;           /* divided by 3000 */
    TIM2->CCMR1 = 0x30;             /* set output to toggle on match */
    TIM2->CCR1 = 0;                 /* set match value */
    TIM2->CCER |= 1;                /* enable ch 1 compare mode */
    TIM2->CNT = 0;                  /* clear counter */
    TIM2->CR1 = 1;                  /* enable TIM2 */

    // configure PA6 as input of TIM3 CH1
    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00003000;    /* clear pin mode */
    GPIOA->MODER |=  0x00002000;    /* set pin to alternate function */
    GPIOA->AFR[0] &= ~0x0F000000;   /* clear pin AF bits */
    GPIOA->AFR[0] |= 0x02000000;    /* set pin to AF2 for TIM3 CH1 */

    // configure TIM3 to do input capture with prescaler ...
    RCC->APB1ENR |= 2;              /* enable TIM3 clock */
    TIM3->PSC = 16000 - 1;          /* divided by 16000 */
    TIM3->CCMR1 = 0x41;             /* set CH1 to capture at every edge */
    TIM3->CCER = 0x0B;              /* enable CH 1 capture both edges */
```

```
    TIM3->CR1 = 1;                  /* enable TIM3 */

    while (1) {
        while (!(TIM3->SR & 2)) {}  /* wait until input edge is captured */
        current = TIM3->CCR1;       /* read captured counter value */
        period = current - last;    /* calculate the period */
        last = current;
        frequency = 1000.0f / period;
        last = current;
    }
}
```

TIM2 External Event Counting

```
/* TIM2 External Event Counting

 * This program configures TIM2 to use external input PB8 as the
 * counter clock source. Each rising edge of the input signal
 * increments the TIM2 counter by 1.
 * Use an external signal to drive PB8 for testing or use a jumper
 * between PC13 and PB8. PC13 is connected to the user switch.
 *

#include "stm32f4xx.h"
int main(void) {
    // configure PA5 output for the LED
    RCC->AHB1ENR |=  1;                    /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000400;    /* set pin to output mode */

    // configure PB8 as input of TIM2 ETR
    RCC->AHB1ENR |=  2;             /* enable GPIOB clock */
    GPIOB->MODER &= ~0x00030000;    /* clear pin mode */
    GPIOB->MODER |=  0x00020000;    /* set pin to alternate function */
    GPIOB->AFR[1] &= ~0x0000000F;   /* clear pin AF bits */
    GPIOB->AFR[1] |= 0x00000001;    /* set pin to AF1 for TIM2 ETR */

    // configure TIM2 to use external input as counter clock source
    RCC->APB1ENR |= 1;              /* enable TIM2 clock */
    TIM2->SMCR = 0x4377;            /* use ETR input as clock, no prescaler
*/
    TIM2->CNT = 0;                  /* clear counter */
    TIM2->CR1 = 1;                  /* enable TIM2 */

    while (1) {
        // use bit 0 to turn LED on/off
        if (TIM2->CNT & 1)
            GPIOA->ODR |= 0x20;     /* turn on LED */
        else
            GPIOA->ODR &= ~0x20;    /* turn off LED */
    }
}
```

using SW0 interrupt

```
/* using SW0 interrupt
 *
 * User Switch B1 is used to generate interrupt through PC13.
 * The user button is connected to PC13. It has a pull-up resistor
 * so PC13 stays high when the button is not pressed.
 * When the button is pressed, PC13 becomes low.
 * The falling-edge of PC13 (when switch is pressed) triggers an
 * interrupt from External Interrupt Controller (EXTI).
 * In the interrupt handler, the user LD2 is blinked twice.
 * It serves as a crude way to debounce the switch.
 * The green LED (LD2) is connected to PA5.
 *

#include "stm32f4xx.h"

void delayMs(int n);

int main(void) {
    __disable_irq();                    /* global disable IRQs */

    RCC->AHB1ENR |= 4;                 /* enable GPIOC clock */
    RCC->AHB1ENR |= 1;                  /* enable GPIOA clock */
    RCC->APB2ENR |= 0x4000;             /* enable SYSCFG clock */

    /* configure PA5 for LED */
    GPIOA->MODER &= ~0x00000C00;        /* clear pin mode */
    GPIOA->MODER |=  0x00000400;        /* set pin to output mode */

    /* configure PC13 for push button interrupt */
    GPIOC->MODER &= ~0x0C000000;        /* clear pin mode to input mode */

    SYSCFG->EXTICR[3] &= ~0x00F0;       /* clear port selection for EXTI13 */
    SYSCFG->EXTICR[3] |= 0x0020;        /* select port C for EXTI13 */

    EXTI->IMR |= 0x2000;                /* unmask EXTI13 */
    EXTI->FTSR |= 0x2000;               /* select falling edge trigger */

//    NVIC->ISER[1] = 0x00000100;         /* enable IRQ40 (bit 8 of ISER[1])
*/
    NVIC_EnableIRQ(EXTI15_10_IRQn);

    __enable_irq();                     /* global enable IRQs */

    while(1) {
    }
}

void EXTI15_10_IRQHandler(void) {
        GPIOA->BSRR = 0x00000020;   /* turn on green LED */
```

```
        delayMs(250);
        GPIOA->BSRR = 0x00200000;   /* turn off green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00000020;   /* turn on green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00200000;   /* turn off green LED */
        delayMs(250);

        EXTI->PR = 0x2000;          /* clear interrupt pending flag */
}

/* 16 MHz SYSCLK */
void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 3195; i++) ;
}
```

Interrupts from two input pins

```
/* Interrupts from two input pins
 *
 * User Switch PB10 is used to generate interrupt through PC13.
 * PB10 is configured to generate interrupt. It is has the
 * pull-up resistor enabled. Use a jumper wire to ground the
 * pin to generate interrupts.
 * In the interrupt handler, the interrupt pending bits are
 * checked. If the interrupt is from PC13, the LED blinks twice.
 * If the interrupt is from PB10, the LED blinks three times.
 * The green LED (LD2) is connected to PA5.
 *
 *
 */

#include "stm32f4xx.h"

void delayMs(int n);

int main(void) {
    __disable_irq();                    /* global disable IRQs */

    RCC->AHB1ENR |= 4;                 /* enable GPIOC clock */
    RCC->AHB1ENR |= 2;                  /* enable GPIOB clock */
    RCC->AHB1ENR |= 1;                  /* enable GPIOA clock */
    RCC->APB2ENR |= 0x4000;             /* enable SYSCFG clock */

    /* configure PA5 for LED */
    GPIOA->MODER &= ~0x00000C00;        /* clear pin mode */
    GPIOA->MODER |=  0x00000400;        /* set pin to output mode */

    /* configure PC13 for push button interrupt */
    GPIOC->MODER &= ~0x0C000000;        /* clear pin mode to input mode */

    SYSCFG->EXTICR[3] &= ~0x00F0;       /* clear port selection for EXTI13 */
    SYSCFG->EXTICR[3] |= 0x0020;        /* select port C for EXTI13 */

    EXTI->IMR |= 0x2000;                /* unmask EXTI13 */
    EXTI->FTSR |= 0x2000;               /* select falling edge trigger */

    /* configure PB10 for interrupt */
    GPIOB->MODER &= ~0x00030000;        /* clear pin mode to input mode */
    GPIOB->PUPDR &= ~0x00300000;
    GPIOB->PUPDR |=  0x00100000;        /* enable pull up resistor */

    SYSCFG->EXTICR[2] &= ~0x0F00;       /* clear port selection for EXTI10 */
    SYSCFG->EXTICR[2] |= 0x0100;        /* select port B for EXTI10 */

    EXTI->IMR |= 0x0400;                /* unmask EXTI10 */
    EXTI->FTSR |= 0x0400;               /* select falling edge trigger */
```

```
    NVIC_EnableIRQ(EXTI15_10_IRQn);     /* enable interrupt in NVIC */
    __enable_irq();                     /* global enable IRQs */

    while(1) {
    }
}

void EXTI15_10_IRQHandler(void) {
    if (EXTI->PR == 0x2000) {       /* interrupt from PC13 */
        GPIOA->BSRR = 0x00000020;   /* turn on green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00200000;   /* turn off green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00000020;   /* turn on green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00200000;   /* turn off green LED */
        delayMs(250);

        EXTI->PR = 0x2000;          /* clear interrupt pending flag */
    }
    else if (EXTI->PR == 0x0400) {  /* interrupt from PB10 */
        GPIOA->BSRR = 0x00000020;   /* turn on green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00200000;   /* turn off green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00000020;   /* turn on green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00200000;   /* turn off green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00000020;   /* turn on green LED */
        delayMs(250);
        GPIOA->BSRR = 0x00200000;   /* turn off green LED */
        delayMs(250);

        EXTI->PR = 0x0400;          /* clear interrupt pending flag */
    }
}

/* 16 MHz SYSCLK */
void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 3195; i++) ;
}
```

UART on USART2 Receive at 115200 Baud using Interrupts

```
/* p6_3.c UART on USART2 Receive at 115200 Baud
 *
 * Receive key strokes from terminal emulator (Tera Term) of the
 * host PC to the UART on USART2 of the Nucleo-F446RE board and
 * use the key value to blink the LED.
 *
 * By default, the clock is running at 16 MHz.
 * The UART is configured for 115200 Baud.
 * PA3 - USART2 RX (AF7)
 *

#include "stm32F4xx.h"

void USART2_init(void);
void LED_blink(int value);
void delayMs(int);

/*----------------------------------------------------------------------------
-
  MAIN function
 *----------------------------------------------------------------------------
-*/
int main (void) {
    __disable_irq();                /* global disable IRQs */

    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000400;    /* set pin to output mode */

    USART2_init();                  /* initialize USART2 */

    USART2->CR1 |= 0x0020;          /* enable Rx interrupt */
    NVIC_EnableIRQ(USART2_IRQn);    /* enable interrupt in NVIC */
    __enable_irq();                 /* global enable IRQs */

    while(1) {                      /* Loop forever */
    }
}

/*----------------------------------------------------------------------------
-
  Initialize UART pins, Baudrate
 *----------------------------------------------------------------------------
-*/
void USART2_init (void) {
    RCC->AHB1ENR |= 1;          /* Enable GPIOA clock */
    RCC->APB1ENR |= 0x20000;    /* Enable USART2 clock */

    /* Configure PA2 for USART2 TX */
```

```
    GPIOA->AFR[0] &= ~0xF000;
    GPIOA->AFR[0] |=  0x7000;   /* alt7 for USART2 */
    GPIOA->MODER  &= ~0x00C0;
    GPIOA->MODER  |=  0x0080;   /* enable alternate function for PA3 */

    USART2->BRR = 0x008B;       /* 115200 baud @ 16 MHz */
    USART2->CR1 = 0x0004;       /* enable Rx, 8-bit data */
    USART2->CR2 = 0x0000;       /* 1 stop bit */
    USART2->CR3 = 0x0000;       /* no flow control */
    USART2->CR1 |= 0x2000;      /* enable USART2 */
}

void USART2_IRQHandler(void) {
    char c;

    if (USART2->SR & 0x0020) {
        c = USART2->DR;             /* Read a character from USART2 */
        LED_blink(c);               /* blink the LED */
    }
}

/* turn on or off the LEDs according to the value */
void LED_blink(int value) {
    value %= 16;                    /* cap the max count at 15 */

    for (; value > 0; value--) {
        GPIOA->BSRR = 0x00000020;   /* turn on LED */
        delayMs(200);
        GPIOA->BSRR = 0x00200000;   /* turn off LED */
        delayMs(200);
    }
    delayMs(800);
}

void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 2000; i++) ;
}
```

Toggle the LED using the SysTick interrupt

```
/* Toggle the LED using the SysTick interrupt
 *
 * This program sets up the SysTick to interrupt at 1 Hz.
 * The system clock is running at 16 MHz.
 * 1 sec/1 us = 16,000,000-1 for RELOAD register.
 * In the interrupt handler, the LED is toggled.
 *

#include "stm32F4xx.h"

int main(void) {
    __disable_irq();                /* global disable IRQs */

    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000400;    /* set pin to output mode */

    /* Configure SysTick */
    SysTick->LOAD = 16000000-1;     /* reload with number of clocks per
second */
    SysTick->VAL = 0;
    SysTick->CTRL = 7;              /* enable SysTick interrupt, use system
clock */
    __enable_irq();

    while (1) {
    }
}

void SysTick_Handler(void) {
    GPIOA->ODR ^= 0x00000020;   /* turn on LED */
}
```

Timer TIM2 interrupt

```
/*  Timer TIM2 interrupt
 *
 * Timer TIM2 is configured as an up-counter. By default, the system clock is
 * running at 16 MHz. The prescaler is set to divide by 16,000 that gives a
 * 1 kHz clock to the counter. The counter auto-reload is set to 999. When
 * the counter counts to 999, it updates the counter to zero and sets the
 * update interrupt flag (UIF). The UIE bit of TIM2->DIER is set so that
 * the UIF triggers a timer interrupt. The interrupt frequency is 1 Hz.
 * In the timer interrupt handler, the green LED (PA5) is toggled and
 * the UIF is cleared.
 *
 *
 */
#include "stm32f4xx.h"

void delayMs(int n);

int main(void) {
    __disable_irq();                /* global disable IRQs */
    RCC->AHB1ENR |=  1;                   /* enable GPIOA clock */

    GPIOA->MODER &= ~0x00000C00;
    GPIOA->MODER |=  0x00000400;

    /* setup TIM2 */
    RCC->APB1ENR |= 1;              /* enable TIM2 clock */
    TIM2->PSC = 16000 - 1;          /* divided by 16000 */
    TIM2->ARR = 1000 - 1;           /* divided by 1000 */
    TIM2->CR1 = 1;                  /* enable counter */

    TIM2->DIER |= 1;                /* enable UIE */
    NVIC_EnableIRQ(TIM2_IRQn);      /* enable interrupt in NVIC */

    __enable_irq();                 /* global enable IRQs */

    while(1) {
    }
}

void TIM2_IRQHandler(void) {
    TIM2->SR = 0;                   /* clear UIF */
    GPIOA->ODR ^= 0x20;                             /* toggle LED */
}
```

Demonstrate Nested Interrupts

```
/*  Demonstrate Nested Interrupts
 *
 * TIM3 is setup to interrupt at about 1 Hz. In the interrupt handler,
 * PB3 is turned on and a delay function of 500 ms is called.
 * PB3 is turned off at the end of the delay.
 *
 * TIM4 is setup to interrupt at about 10 Hz. In the interrupt handler,
 * PB4 and the LED0 is turned on and a delay function of 25 ms is called.
 * PB4 and the LED0 is turned off at the end of the delay.
 *
 * When TIM3 has the higher priority (the way this code is), the TIM4
 * interrupts are blocked by TIM3 interrupt handler and the LED stops
 * blinking intermittently. You can see their interactions better
 * with an oscilloscope on PB3 and PB4.
 *
 * When TIM4 has the higher priority (you need to switch the priority of
 * the two timers at the #define statements), the TIM3 interrupt handler is
 * preempted by TIM4 interrupts and the LED is blinking all the time.
 *
 *
 */

#include "stm32f4xx.h"

void delayMs(int n);

/* priority of TIM3 and TTIM4 should be between 0 and 15 */
#define PRIO_TIM3 3
#define PRIO_TIM4 4

int main (void) {
    __disable_irq();                /* global disable IRQs */
    RCC->AHB1ENR |=  1;                   /* enable GPIOA clock */
    RCC->AHB1ENR |=  2;                   /* enable GPIOB clock */

    GPIOA->MODER &= ~0x00000C00;    /* pin PA5 (LED) output */
    GPIOA->MODER |=  0x00000400;
    GPIOB->MODER &= ~0x000003C0;    /* pins PB3, PB4 output */
    GPIOB->MODER |=  0x00000140;

    /* setup TIM3 */
    RCC->APB1ENR |= 2;              /* enable TIM3 clock */
    TIM3->PSC = 1000 - 1;           /* divided by 1000 */
    TIM3->ARR = 15500 - 1;          /* divided by 15500 */
    TIM3->CR1 = 1;                  /* enable counter */

    TIM3->DIER |= 1;                /* enable UIE */
    NVIC_SetPriority(TIM3_IRQn, PRIO_TIM3);
    NVIC_EnableIRQ(TIM3_IRQn);      /* enable interrupt in NVIC */
```

```
    /* setup TIM4 */
    RCC->APB1ENR |= 4;              /* enable TIM4 clock */
    TIM4->PSC = 1000 - 1;           /* divided by 1000 */
    TIM4->ARR = 1562 - 1;           /* divided by 1562 */
    TIM4->CR1 = 1;                  /* enable counter */

    TIM4->DIER |= 1;                /* enable UIE */
    NVIC_SetPriority(TIM4_IRQn, PRIO_TIM4);
    NVIC_EnableIRQ(TIM4_IRQn);      /* enable interrupt in NVIC */

    __enable_irq();                 /* enable interrupt globally */

    while(1) {
    }
}

void TIM3_IRQHandler(void) {
    GPIOB->BSRR = 0x00000008;       /* turn on PB3 */
    delayMs(500);
    GPIOB->BSRR = 0x00080000;       /* turn off PB3 */
    TIM3->SR = 0;                   /* clear UIF */
}

void TIM4_IRQHandler(void) {
    GPIOA->BSRR = 0x00000020;       /* turn on LED */
    GPIOB->BSRR = 0x00000010;       /* turn on PB4 */
    delayMs(25);
    GPIOA->BSRR = 0x00200000;       /* turn off LED */
    GPIOB->BSRR = 0x00100000;       /* turn off PB4 */
    TIM4->SR = 0;                   /* clear UIF */
}

/* 16 MHz SYSCLK */
void delayMs(int n) {
    int i;
    for (; n > 0; n--)
        for (i = 0; i < 3195; i++) ;
}
```

PIN CONFIGUARATION

Table 10. STM32F427xx and STM32F429xx pin and ball definitions

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
1	1	B2	A2	1	D8	1	A3	PE2	I/O	FT	-	TRACECLK, SPI4_SCK, SAI1_MCLK_A, ETH_MII_TXD3, FMC_A23, EVENTOUT	-
2	2	C1	A1	2	C10	2	A2	PE3	I/O	FT	-	TRACED0, SAI1_SD_B, FMC_A19, EVENTOUT	-
3	3	C2	B1	3	B11	3	A1	PE4	I/O	FT	-	TRACED1, SPI4_NSS, SAI1_FS_A, FMC_A20, DCMI_D4, LCD_B0, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
4	4	D1	B2	4	D9	4	B1	PE5	I/O	FT	-	TRACED2, TIM9_CH1, SPI4_MISO, SAI1_SCK_A, FMC_A21, DCMI_D6, LCD_G0, EVENTOUT	-
5	5	D2	B3	5	E8	5	B2	PE6	I/O	FT	-	TRACED3, TIM9_CH2, SPI4_MOSI, SAI1_SD_A, FMC_A22, DCMI_D7, LCD_G1, EVENTOUT	-
-	-	-	-	-	-	-	G6	V_{SS}	S	-	-	-	-
-	-	-	-	-	-	-	F5	V_{DD}	S	-	-	-	-
6	6	E5	C1	6	C11	6	C1	V_{BAT}	S	-	-	-	-
-	-	NC (2)	D2	7	-	7	C2	PI8	I/O	FT	(3) (4)	EVENTOUT	TAMP_2
7	7	E4	D1	8	D10	8	D1	PC13	I/O	FT	(3) (4)	EVENTOUT	TAMP_1
8	8	E1	E1	9	D11	9	E1	PC14-OSC32_IN (PC14)	I/O	FT	(3) (4)	EVENTOUT	OSC32_IN (5)
9	9	F1	F1	10	E11	10	F1	PC15-OSC32_OUT (PC15)	I/O	FT	(3) (4)	EVENTOUT	OSC32_OUT(5)
-	-	-	-	-	-	-	G5	V_{DD}	S	-	-	-	-
-	-	E2	D3	11	-	11	E4	PI9	I/O	FT	-	CAN1_RX, FMC_D30, LCD_VSYNC, EVENTOUT	-
-	-	E3	E3	12	-	12	D5	PI10	I/O	FT	-	ETH_MII_RX_ER, FMC_D31, LCD_HSYNC, EVENTOUT	-
-	-	NC (2)	E4	13	-	13	F3	PI11	I/O	FT	-	OTG_HS_ULPI_DIR, EVENTOUT	-
-	-	F6	F2	14	E7	14	F2	V_{SS}	S	-	-	-	-
-	-	F4	F3	15	E10	15	F4	V_{DD}	S	-	-	-	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
-	10	F2	E2	16	F11	16	D2	PF0	I/O	FT	-	I2C2_SDA, FMC_A0, EVENTOUT	-
-	11	F3	H3	17	E9	17	E2	PF1	I/O	FT	-	I2C2_SCL, FMC_A1, EVENTOUT	-
-	12	G5	H2	18	F10	18	G2	PF2	I/O	FT	-	I2C2_SMBA, FMC_A2, EVENTOUT	-
-	-	-	-	-	-	19	E3	PI12	I/O	FT	-	LCD_HSYNC, EVENTOUT	-
-	-	-	-	-	-	20	G3	PI13	I/O	FT	-	LCD_VSYNC, EVENTOUT	-
-	-	-	-	-	-	21	H3	PI14	I/O	FT		LCD_CLK, EVENTOUT	-
-	13	G4	J2	19	G11	22	H2	PF3	I/O	FT	(5)	FMC_A3, EVENTOUT	ADC3_IN9
-	14	G3	J3	20	F9	23	J2	PF4	I/O	FT	(5)	FMC_A4, EVENTOUT	ADC3_ IN14
-	15	H3	K3	21	F8	24	K3	PF5	I/O	FT	(5)	FMC_A5, EVENTOUT	ADC3_ IN15
10	16	G7	G2	22	H7	25	H6	V_{SS}	S	-	-	-	-
11	17	G8	G3	23	-	26	H5	V_{DD}	S	-	-	-	-
-	18	NC (2)	K2	24	G10	27	K2	PF6	I/O	FT	(5)	TIM10_CH1, SPI5_NSS, SAI1_SD_B, UART7_Rx, FMC_NIORD, EVENTOUT	ADC3_IN4
-	19	NC (2)	K1	25	F7	28	K1	PF7	I/O	FT	(5)	TIM11_CH1, SPI5_SCK, SAI1_MCLK_B, UART7_Tx, FMC_NREG, EVENTOUT	ADC3_IN5
-	20	NC (2)	L3	26	H11	29	L3	PF8	I/O	FT	(5)	SPI5_MISO, SAI1_SCK_B, TIM13_CH1, FMC_NIOWR, EVENTOUT	ADC3_IN6

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)[1]	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
-	21	NC (2)	L2	27	G8	30	L2	PF9	I/O	FT	(5)	SPI5_MOSI, SAI1_FS_B, TIM14_CH1, FMC_CD, EVENTOUT	ADC3_IN7
-	22	H1	L1	28	G9	31	L1	PF10	I/O	FT	(5)	FMC_INTR, DCMI_D11, LCD_DE, EVENTOUT	ADC3_IN8
12	23	G2	G1	29	J11	32	G1	PH0-OSC_IN (PH0)	I/O	FT	-	EVENTOUT	OSC_IN[5]
13	24	G1	H1	30	H10	33	H1	PH1-OSC_OUT (PH1)	I/O	FT	-	EVENTOUT	OSC_OUT (5)
14	25	H2	J1	31	H9	34	J1	NRST	I/O	RST	-	-	-
15	26	G6	M2	32	H8	35	M2	PC0	I/O	FT	(5)	OTG_HS_ULPI_STP, FMC_SDNWE, EVENTOUT	ADC123_IN10
16	27	H5	M3	33	K11	36	M3	PC1	I/O	FT	(5)	ETH_MDC, EVENTOUT	ADC123_IN11
17	28	H6	M4	34	J10	37	M4	PC2	I/O	FT	(5)	SPI2_MISO, I2S2ext_SD, OTG_HS_ULPI_DIR, ETH_MII_TXD2, FMC_SDNE0, EVENTOUT	ADC123_IN12
18	29	H7	M5	35	J9	38	L4	PC3	I/O	FT	(5)	SPI2_MOSI/I2S2_SD, OTG_HS_ULPI_NXT, ETH_MII_TX_CLK, FMC_SDCKE0, EVENTOUT	ADC123_IN13
19	30	-	-	36	G7	39	J5	V_{DD}	S	-	-	-	-
-	-	-	-	-	-	-	J6	V_{SS}	S	-	-	-	-
20	31	J1	M1	37	K10	40	M1	V_{SSA}	S	-	-	-	-
-	-	J2	N1	-	-	-	N1	V_{REF-}	S	-	-	-	-
21	32	J3	P1	38	L11	41	P1	V_{REF+}	S	-	-	-	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
22	33	J4	R1	39	L10	42	R1	V_{DDA}	S	-	-	-	-
23	34	J5	N3	40	K9	43	N3	PA0-WKUP (PA0)	I/O	FT	(6)	TIM2_CH1/TIM2_ETR, TIM5_CH1, TIM8_ETR, USART2_CTS, UART4_TX, ETH_MII_CRS, EVENTOUT	ADC123_ IN0/WKUP (5)
24	35	K1	N2	41	K8	44	N2	PA1	I/O	FT	(5)	TIM2_CH2, TIM5_CH2, USART2_RTS, UART4_RX, ETH_MII_RX_CLK/ETH_RMII_REF_CLK, EVENTOUT	ADC123_ IN1
25	36	K2	P2	42	L9	45	P2	PA2	I/O	FT	(5)	TIM2_CH3, TIM5_CH3, TIM9_CH1, USART2_TX, ETH_MDIO, EVENTOUT	ADC123_ IN2
-	-	L2	F4	43	-	46	K4	PH2	I/O	FT	-	ETH_MII_CRS, FMC_SDCKE0, LCD_R0, EVENTOUT	-
-	-	L1	G4	44	-	47	J4	PH3	I/O	FT	-	ETH_MII_COL, FMC_SDNE0, LCD_R1, EVENTOUT	-
-	-	M2	H4	45	-	48	H4	PH4	I/O	FT	-	I2C2_SCL, OTG_HS_ULPI_NXT, EVENTOUT	-
-	-	L3	J4	46	-	49	J3	PH5	I/O	FT	-	I2C2_SDA, SPI5_NSS, FMC_SDNWE, EVENTOUT	-
26	37	K3	R2	47	M11	50	R2	PA3	I/O	FT	(5)	TIM2_CH4, TIM5_CH4, TIM9_CH2, USART2_RX, OTG_HS_ULPI_D0, ETH_MII_COL, LCD_B5, EVENTOUT	ADC123_ IN3
27	38	-	-		-	51	K6	V_{SS}	S	-	-	-	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)[1]	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
-	-	M1	L4	48	N11	-	L5	BYPASS_ REG	I	FT	-	-	-
28	39	J11	K4	49	J8	52	K5	V_{DD}	S	-	-	-	-
29	40	N2	N4	50	M10	53	N4	PA4	I/O	TTa	(5)	SPI1_NSS, SPI3_NSS/I2S3_WS, USART2_CK, OTG_HS_SOF, DCMI_HSYNC, LCD_VSYNC, EVENTOUT	ADC12_ IN4 /DAC_ OUT1
30	41	M3	P4	51	M9	54	P4	PA5	I/O	TTa	(5)	TIM2_CH1/TIM2_ETR, TIM8_CH1N, SPI1_SCK, OTG_HS_ULPI_CK, EVENTOUT	ADC12_ IN5/DAC_ OUT2
31	42	N3	P3	52	N10	55	P3	PA6	I/O	FT	(5)	TIM1_BKIN, TIM3_CH1, TIM8_BKIN, SPI1_MISO, TIM13_CH1, DCMI_PIXCLK, LCD_G2, EVENTOUT	ADC12_ IN6
32	43	K4	R3	53	L8	56	R3	PA7	I/O	FT	(5)	TIM1_CH1N, TIM3_CH2, TIM8_CH1N, SPI1_MOSI, TIM14_CH1, ETH_MII_RX_DV/ETH_ RMII_CRS_DV, EVENTOUT	ADC12_ IN7
33	44	L4	N5	54	M8	57	N5	PC4	I/O	FT	(5)	ETH_MII_RXD0/ETH_ RMII_RXD0, EVENTOUT	ADC12_ IN14
34	45	M4	P5	55	N9	58	P5	PC5	I/O	FT	(5)	ETH_MII_RXD1/ETH_ RMII_RXD1, EVENTOUT	ADC12_ IN15
-	-	-	-	-	J7	59	L7	V_{DD}	S	-	-	-	-
-	-	-	-	-	-	60	L6	VSS	S	-	-	-	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)[1]	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
35	46	N4	R5	56	N8	61	R5	PB0	I/O	FT	(5)	TIM1_CH2N, TIM3_CH3, TIM8_CH2N, LCD_R3, OTG_HS_ULPI_D1, ETH_MII_RXD2, EVENTOUT	ADC12_IN8
36	47	K5	R4	57	K7	62	R4	PB1	I/O	FT	(5)	TIM1_CH3N, TIM3_CH4, TIM8_CH3N, LCD_R6, OTG_HS_ULPI_D2, ETH_MII_RXD3, EVENTOUT	ADC12_IN9
37	48	L5	M6	58	L7	63	M5	PB2-BOOT1 (PB2)	I/O	FT	-	EVENTOUT	-
-	-	-	-	-	-	64	G4	PI15	I/O	FT	-	LCD_R0, EVENTOUT	-
-	-	-	-	-	-	65	R6	PJ0	I/O	FT	-	LCD_R1, EVENTOUT	-
-	-	-	-	-	-	66	R7	PJ1	I/O	FT	-	LCD_R2, EVENTOUT	-
-	-	-	-	-	-	67	P7	PJ2	I/O	FT	-	LCD_R3, EVENTOUT	-
-	-	-	-	-	-	68	N8	PJ3	I/O	FT	-	LCD_R4, EVENTOUT	-
-	-	-	-	-	-	69	M9	PJ4	I/O	FT	-	LCD_R5, EVENTOUT	-
-	49	M5	R6	59	M7	70	P8	PF11	I/O	FT	-	SPI5_MOSI, FMC_SDNRAS, DCMI_D12. EVENTOUT	-
-	50	N5	P6	60	N7	71	M6	PF12	I/O	FT	-	FMC_A6, EVENTOUT	-
-	51	G9	M8	61	-	72	K7	V_{SS}	S		-	-	-
-	52	D10	N8	62	-	73	L8	V_{DD}	S		-	-	-
-	53	M6	N6	63	K6	74	N6	PF13	I/O	FT	-	FMC_A7, EVENTOUT	-
-	54	K7	R7	64	L6	75	P6	PF14	I/O	FT	-	FMC_A8, EVENTOUT	-
-	55	L7	P7	65	M6	76	M8	PF15	I/O	FT	-	FMC_A9, EVENTOUT	-
-	56	N6	N7	66	N6	77	N7	PG0	I/O	FT	-	FMC_A10, EVENTOUT	-
-	57	M7	M7	67	K5	78	M7	PG1	I/O	FT	-	FMC_A11, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)[1]	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
38	58	N7	R8	68	L5	79	R8	PE7	I/O	FT	-	TIM1_ETR, UART7_Rx, FMC_D4, EVENTOUT	-
39	59	J8	P8	69	M5	80	N9	PE8	I/O	FT	-	TIM1_CH1N, UART7_Tx, FMC_D5, EVENTOUT	-
40	60	K8	P9	70	N5	81	P9	PE9	I/O	FT	-	TIM1_CH1, FMC_D6, EVENTOUT	-
-	61	J6	M9	71	H3	82	K8	V_{SS}	S		-		-
-	62	G10	N9	72	J5	83	L9	V_{DD}	S		-		-
41	63	L8	R9	73	J4	84	R9	PE10	I/O	FT	-	TIM1_CH2N, FMC_D7, EVENTOUT	-
42	64	M8	P10	74	K4	85	P10	PE11	I/O	FT	-	TIM1_CH2, SPI4_NSS, FMC_D8, LCD_G3, EVENTOUT	-
43	65	N8	R10	75	L4	86	R10	PE12	I/O	FT	-	TIM1_CH3N, SPI4_SCK, FMC_D9, LCD_B4, EVENTOUT	-
44	66	H9	N11	76	N4	87	R12	PE13	I/O	FT	-	TIM1_CH3, SPI4_MISO, FMC_D10, LCD_DE, EVENTOUT	-
45	67	J9	P11	77	M4	88	P11	PE14	I/O	FT	-	TIM1_CH4, SPI4_MOSI, FMC_D11, LCD_CLK, EVENTOUT	-
46	68	K9	R11	78	L3	89	R11	PE15	I/O	FT	-	TIM1_BKIN, FMC_D12, LCD_R7, EVENTOUT	-
47	69	L9	R12	79	M3	90	P12	PB10	I/O	FT	-	TIM2_CH3, I2C2_SCL, SPI2_SCK/I2S2_CK, USART3_TX, OTG_HS_ULPI_D3, ETH_MII_RX_ER, LCD_G4, EVENTOUT	-
48	70	M9	R13	80	N3	91	R13	PB11	I/O	FT	-	TIM2_CH4, I2C2_SDA, USART3_RX, OTG_HS_ULPI_D4, ETH_MII_TX_EN/ETH_RMII_TX_EN, LCD_G5, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
49	71	N9	M10	81	N2	92	L11	V_{CAP_1}	S	-	-	-	-
-	-	-	-	-	H2	93	K9	V_{SS}	S	-	-	-	-
50	72	F8	N10	82	J6	94	L10	V_{DD}	S	-	-	-	-
-	-	-	-	-	-	95	M14	PJ5	I/O	-	-	LCD_R6, EVENTOUT	-
-	-	N10	M11	83	-	96	P13	PH6	I/O	FT	-	I2C2_SMBA, SPI5_SCK, TIM12_CH1, ETH_MII_RXD2, FMC_SDNE1, DCMI_D8, EVENTOUT	-
-	-	M10	N12	84	-	97	N13	PH7	I/O	FT	-	I2C3_SCL, SPI5_MISO, ETH_MII_RXD3, FMC_SDCKE1, DCMI_D9, EVENTOUT	-
-	-	L10	M12	85	-	98	P14	PH8	I/O	FT	-	I2C3_SDA, FMC_D16, DCMI_HSYNC, LCD_R2, EVENTOUT	-
-	-	K10	M13	86	-	99	N14	PH9	I/O	FT	-	I2C3_SMBA, TIM12_CH2, FMC_D17, DCMI_D0, LCD_R3, EVENTOUT	-
-	-	N11	L13	87	-	100	P15	PH10	I/O	FT	-	TIM5_CH1, FMC_D18, DCMI_D1, LCD_R4, EVENTOUT	-
-	-	M11	L12	88	-	101	N15	PH11	I/O	FT	-	TIM5_CH2, FMC_D19, DCMI_D2, LCD_R5, EVENTOUT	-
-	-	L11	K12	89	-	102	M15	PH12	I/O	FT	-	TIM5_CH3, FMC_D20, DCMI_D3, LCD_R6, EVENTOUT	-
-	-	E7	H12	90	-	-	K10	V_{SS}	S	-	-	-	-
-	-	H8	J12	91	-	103	K11	V_{DD}	S	-	-	-	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)[1]	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
51	73	N12	P12	92	M2	104	L13	PB12	I/O	FT	-	TIM1_BKIN, I2C2_SMBA, SPI2_NSS/I2S2_WS, USART3_CK, CAN2_RX, OTG_HS_ULPI_D5, ETH_MII_TXD0/ETH_RMII_TXD0, OTG_HS_ID, EVENTOUT	-
52	74	M12	P13	93	N1	105	K14	PB13	I/O	FT	-	TIM1_CH1N, SPI2_SCK/I2S2_CK, USART3_CTS, CAN2_TX, OTG_HS_ULPI_D6, ETH_MII_TXD1/ETH_RMII_TXD1, EVENTOUT	OTG_HS_VBUS
53	75	M13	R14	94	K3	106	R14	PB14	I/O	FT	-	TIM1_CH2N, TIM8_CH2N, SPI2_MISO, I2S2ext_SD, USART3_RTS, TIM12_CH1, OTG_HS_DM, EVENTOUT	-
54	76	L13	R15	95	J3	107	R15	PB15	I/O	FT	-	RTC_REFIN, TIM1_CH3N, TIM8_CH3N, SPI2_MOSI/I2S2_SD, TIM12_CH2, OTG_HS_DP, EVENTOUT	-
55	77	L12	P15	96	L2	108	L15	PD8	I/O	FT	-	USART3_TX, FMC_D13, EVENTOUT	-
56	78	K13	P14	97	M1	109	L14	PD9	I/O	FT	-	USART3_RX, FMC_D14, EVENTOUT	-
57	79	K11	N15	98	H4	110	K15	PD10	I/O	FT	-	USART3_CK, FMC_D15, LCD_B3, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
58	80	H10	N14	99	K2	111	N10	PD11	I/O	FT	-	USART3_CTS, FMC_A16, EVENTOUT	-
59	81	J13	N13	100	H6	112	M10	PD12	I/O	FT	-	TIM4_CH1, USART3_RTS, FMC_A17, EVENTOUT	-
60	82	K12	M15	101	H5	113	M11	PD13	I/O	FT	-	TIM4_CH2, FMC_A18, EVENTOUT	-
-	83	-	-	102	-	114	J10	V_{SS}	S		-	-	-
-	84	F7	J13	103	L1	115	J11	V_{DD}	S		-	-	-
61	85	H11	M14	104	J2	116	L12	PD14	I/O	FT	-	TIM4_CH3, FMC_D0, EVENTOUT	-
62	86	J12	L14	105	K1	117	K13	PD15	I/O	FT	-	TIM4_CH4, FMC_D1, EVENTOUT	-
-	-	-	-	-	-	118	K12	PJ6	I/O	FT	-	LCD_R7, EVENTOUT	-
-	-	-	-	-	-	119	J12	PJ7	I/O	FT	-	LCD_G0, EVENTOUT	-
-	-	-	-	-	-	120	H12	PJ8	I/O	FT	-	LCD_G1, EVENTOUT	-
-	-	-	-	-	-	121	J13	PJ9	I/O	FT	-	LCD_G2, EVENTOUT	-
-	-	-	-	-	-	122	H13	PJ10	I/O	FT	-	LCD_G3, EVENTOUT	-
-	-	-	-	-	-	123	G12	PJ11	I/O	FT	-	LCD_G4, EVENTOUT	-
-	-	-	-	-	-	124	H11	VDD	I/O	FT	-	-	-
-	-	-	-	-	-	125	H10	VSS	I/O	FT	-	-	-
-	-	-	-	-	-	126	G13	PK0	I/O	FT	-	LCD_G5, EVENTOUT	-
-	-	-	-	-	-	127	F12	PK1	I/O	FT	-	LCD_G6, EVENTOUT	-
-	-	-	-	-	-	128	F13	PK2	I/O	FT	-	LCD_G7, EVENTOUT	-
-	87	H13	L15	106	J1	129	M13	PG2	I/O	FT	-	FMC_A12, EVENTOUT	-
-	88	NC (2)	K15	107	G3	130	M12	PG3	I/O	FT	-	FMC_A13, EVENTOUT	-
-	89	H12	K14	108	G5	131	N12	PG4	I/O	FT	-	FMC_A14/FMC_BA0, EVENTOUT	-
-	90	G13	K13	109	G6	132	N11	PG5	I/O	FT	-	FMC_A15/FMC_BA1, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
-	91	G11	J15	110	G4	133	J15	PG6	I/O	FT	-	FMC_INT2, DCMI_D12, LCD_R7, EVENTOUT	-
-	92	G12	J14	111	H1	134	J14	PG7	I/O	FT	-	USART6_CK, FMC_INT3, DCMI_D13, LCD_CLK, EVENTOUT	-
-	93	F13	H14	112	G2	135	H14	PG8	I/O	FT	-	SPI6_NSS, USART6_RTS, ETH_PPS_OUT, FMC_SDCLK, EVENTOUT	-
-	94	J7	G12	113	D2	136	G10	V_{SS}	S		-	-	-
-	95	E6	H13	114	G1	137	G11	V_{DD}	S		-	-	-
63	96	F9	H15	115	F2	138	H15	PC6	I/O	FT	-	TIM3_CH1, TIM8_CH1, I2S2_MCK, USART6_TX, SDIO_D6, DCMI_D0, LCD_HSYNC, EVENTOUT	-
64	97	F10	G15	116	F3	139	G15	PC7	I/O	FT	-	TIM3_CH2, TIM8_CH2, I2S3_MCK, USART6_RX, SDIO_D7, DCMI_D1, LCD_G6, EVENTOUT	-
65	98	F11	G14	117	E4	140	G14	PC8	I/O	FT	-	TIM3_CH3, TIM8_CH3, USART6_CK, SDIO_D0, DCMI_D2, EVENTOUT	-
66	99	F12	F14	118	E3	141	F14	PC9	I/O	FT	-	MCO2, TIM3_CH4, TIM8_CH4, I2C3_SDA, I2S_CKIN, SDIO_D1, DCMI_D3, EVENTOUT	-
67	100	E13	F15	119	F1	142	F15	PA8	I/O	FT	-	MCO1, TIM1_CH1, I2C3_SCL, USART1_CK, OTG_FS_SOF, LCD_R6, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)[1]	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
68	101	E8	E15	120	E2	143	E15	PA9	I/O	FT	-	TIM1_CH2, I2C3_SMBA, USART1_TX, DCMI_D0, EVENTOUT	OTG_FS_VBUS
69	102	E9	D15	121	D5	144	D15	PA10	I/O	FT	-	TIM1_CH3, USART1_RX, OTG_FS_ID, DCMI_D1, EVENTOUT	-
70	103	E10	C15	122	D4	145	C15	PA11	I/O	FT	-	TIM1_CH4, USART1_CTS, CAN1_RX, LCD_R4, OTG_FS_DM, EVENTOUT	-
71	104	E11	B15	123	E1	146	B15	PA12	I/O	FT	-	TIM1_ETR, USART1_RTS, CAN1_TX, LCD_R5, OTG_FS_DP, EVENTOUT	-
72	105	E12	A15	124	D3	147	A15	PA13 (JTMS-SWDIO)	I/O	FT	-	JTMS-SWDIO, EVENTOUT	-
73	106	D12	F13	125	D1	148	E11	V_{CAP_2}	S		-	-	-
74	107	J10	F12	126	D2	149	F10	V_{SS}	S		-	-	-
75	108	H4	G13	127	C1	150	F11	V_{DD}	S		-	-	-
-	-	D13	E12	128	-	151	E12	PH13	I/O	FT	-	TIM8_CH1N, CAN1_TX, FMC_D21, LCD_G2, EVENTOUT	-
-	-	C13	E13	129	-	152	E13	PH14	I/O	FT	-	TIM8_CH2N, FMC_D22, DCMI_D4, LCD_G3, EVENTOUT	-
-	-	C12	D13	130	-	153	D13	PH15	I/O	FT	-	TIM8_CH3N, FMC_D23, DCMI_D11, LCD_G4, EVENTOUT	-
-	-	B13	E14	131	-	154	E14	PI0	I/O	FT	-	TIM5_CH4, SPI2_NSS/I2S2_WS[7], FMC_D24, DCMI_D13, LCD_G5, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)[1]	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
-	-	C11	D14	132	-	155	D14	PI1	I/O	FT	-	SPI2_SCK/I2S2_CK[7], FMC_D25, DCMI_D8, LCD_G6, EVENTOUT	-
-	-	B12	C14	133	-	156	C14	PI2	I/O	FT	-	TIM8_CH4, SPI2_MISO, I2S2ext_SD, FMC_D26, DCMI_D9, LCD_G7, EVENTOUT	-
-	-	A12	C13	134	-	157	C13	PI3	I/O	FT	-	TIM8_ETR, SPI2_MOSI/I2S2_SD, FMC_D27, DCMI_D10, EVENTOUT	-
-	-	D11	D9	135	F5	-	F9	V_{SS}	S		-	-	-
-	-	D3	C9	136	A1	158	E10	V_{DD}	S		-	-	-
76	109	A11	A14	137	B1	159	A14	PA14 (JTCK-SWCLK)	I/O	FT	-	JTCK-SWCLK/ EVENTOUT	-
77	110	B11	A13	138	C2	160	A13	PA15 (JTDI)	I/O	FT	-	JTDI, TIM2_CH1/TIM2_ETR, SPI1_NSS, SPI3_NSS/I2S3_WS, EVENTOUT	-
78	111	C10	B14	139	A2	161	B14	PC10	I/O	FT	-	SPI3_SCK/I2S3_CK, USART3_TX, UART4_TX, SDIO_D2, DCMI_D8, LCD_R2, EVENTOUT	-
79	112	B10	B13	140	B2	162	B13	PC11	I/O	FT	-	I2S3ext_SD, SPI3_MISO, USART3_RX, UART4_RX, SDIO_D3, DCMI_D4, EVENTOUT	-
80	113	A10	A12	141	C3	163	A12	PC12	I/O	FT	-	SPI3_MOSI/I2S3_SD, USART3_CK, UART5_TX, SDIO_CK, DCMI_D9, EVENTOUT	-
81	114	D9	B12	142	B3	164	B12	PD0	I/O	FT	-	CAN1_RX, FMC_D2, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
82	115	C9	C12	143	C4	165	C12	PD1	I/O	FT	-	CAN1_TX, FMC_D3, EVENTOUT	-
83	116	B9	D12	144	A3	166	D12	PD2	I/O	FT	-	TIM3_ETR. UART5_RX, SDIO_CMD, DCMI_D11, EVENTOUT	-
84	117	A9	D11	145	B4	167	C11	PD3	I/O	FT	-	SPI2_SCK/I2S2_CK, USART2_CTS, FMC_CLK, DCMI_D5, LCD_G7, EVENTOUT	-
85	118	D8	D10	146	B5	168	D11	PD4	I/O	FT	-	USART2_RTS, FMC_NOE, EVENTOUT	-
86	119	C8	C11	147	A4	169	C10	PD5	I/O	FT	-	USART2_TX, FMC_NWE, EVENTOUT	-
-	120	-	D8	148	-	170	F8	V_{SS}	S		-	-	-
-	121	D6	C8	149	C5	171	E9	V_{DD}	S		-	-	-
87	122	B8	B11	150	F4	172	B11	PD6	I/O	FT	-	SPI3_MOSI/I2S3_SD, SAI1_SD_A, USART2_RX, FMC_NWAIT, DCMI_D10, LCD_B2, EVENTOUT	-
88	123	A8	A11	151	A5	173	A11	PD7	I/O	FT	-	USART2_CK, FMC_NE1/FMC_NCE2, EVENTOUT	-
-	-	-	-	-	-	174	B10	PJ12	I/O	FT	-	LCD_B0, EVENTOUT	-
-	-	-	-	-	-	175	B9	PJ13	I/O	FT	-	LCD_B1, EVENTOUT	-
-	-	-	-	-	-	176	C9	PJ14	I/O	FT	-	LCD_B2, EVENTOUT	-
-	-	-	-	-	-	177	D10	PJ15	I/O	FT	-	LCD_B3, EVENTOUT	-
-	124	NC (2)	C10	152	E5	178	D9	PG9	I/O	FT	-	USART6_RX, FMC_NE2/FMC_NCE3, DCMI_VSYNC(8), EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
-	125	C7	B10	153	C6	179	C8	PG10	I/O	FT	-	LCD_G3, FMC_NCE4_1/FMC_NE3, DCMI_D2, LCD_B2, EVENTOUT	-
-	126	B7	B9	154	B6	180	B8	PG11	I/O	FT	-	ETH_MII_TX_EN/ETH_RMII_TX_EN, FMC_NCE4_2, DCMI_D3, LCD_B3, EVENTOUT	-
-	127	A7	B8	155	A6	181	C7	PG12	I/O	FT	-	SPI6_MISO, USART6_RTS, LCD_B4, FMC_NE4, LCD_B1, EVENTOUT	-
-	128	NC (2)	A8	156	D6	182	B3	PG13	I/O	FT	-	SPI6_SCK, USART6_CTS, ETH_MII_TXD0/ETH_RMII_TXD0, FMC_A24, EVENTOUT	-
-	129	NC (2)	A7	157	F6	183	A4	PG14	I/O	FT	-	SPI6_MOSI, USART6_TX, ETH_MII_TXD1/ETH_RMII_TXD1, FMC_A25, EVENTOUT	-
-	130	D7	D7	158	-	184	F7	V_{SS}	S		-	-	-
-	131	L6	C7	159	E6	185	E8	V_{DD}	S		-	-	-
-	-	-	-	-	-	186	D8	PK3	I/O	FT	-	LCD_B4, EVENTOUT	-
-	-	-	-	-	-	187	D7	PK4	I/O	FT	-	LCD_B5, EVENTOUT	-
-	-	-	-	-	-	188	C6	PK5	I/O	FT	-	LCD_B6, EVENTOUT	-
-	-	-	-	-	-	189	C5	PK6	I/O	FT	-	LCD_B7, EVENTOUT	-
-	-	-	-	-	-	190	C4	PK7	I/O	FT	-	LCD_DE, EVENTOUT	-
-	132	C6	B7	160	A7	191	B7	PG15	I/O	FT	-	USART6_CTS, FMC_SDNCAS, DCMI_D13, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
89	133	B6	A10	161	B7	192	A10	PB3 (JTDO/TRACESWO)	I/O	FT	-	JTDO/TRACESWO, TIM2_CH2, SPI1_SCK, SPI3_SCK/I2S3_CK, EVENTOUT	-
90	134	A6	A9	162	C7	193	A9	PB4 (NJTRST)	I/O	FT	-	NJTRST, TIM3_CH1, SPI1_MISO, SPI3_MISO, I2S3ext_SD, EVENTOUT	-
91	135	D5	A6	163	C8	194	A8	PB5	I/O	FT	-	TIM3_CH2, I2C1_SMBA, SPI1_MOSI, SPI3_MOSI/I2S3_SD, CAN2_RX, OTG_HS_ULPI_D7, ETH_PPS_OUT, FMC_SDCKE1, DCMI_D10, EVENTOUT	-
92	136	C5	B6	164	A8	195	B6	PB6	I/O	FT	-	TIM4_CH1, I2C1_SCL, USART1_TX, CAN2_TX, FMC_SDNE1, DCMI_D5, EVENTOUT	-
93	137	B5	B5	165	B8	196	B5	PB7	I/O	FT	-	TIM4_CH2, I2C1_SDA, USART1_RX, FMC_NL, DCMI_VSYNC, EVENTOUT	-
94	138	A5	D6	166	C9	197	E6	BOOT0	I	B	-		V_{PP}
95	139	D4	A5	167	A9	198	A7	PB8	I/O	FT	-	TIM4_CH3, TIM10_CH1, I2C1_SCL, CAN1_RX, ETH_MII_TXD3, SDIO_D4, DCMI_D6, LCD_B6, EVENTOUT	-

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number								Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA169	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
96	140	C4	B4	168	B9	199	B4	PB9	I/O	FT	-	TIM4_CH4, TIM11_CH1, I2C1_SDA, SPI2_NSS/I2S2_WS, CAN1_TX, SDIO_D5, DCMI_D7, LCD_B7, EVENTOUT	-
97	141	B4	A4	169	B10	200	A6	PE0	I/O	FT	-	TIM4_ETR, UART8_RX, FMC_NBL0, DCMI_D2, EVENTOUT	-
98	142	A4	A3	170	A10	201	A5	PE1	I/O	FT	-	UART8_Tx, FMC_NBL1, DCMI_D3, EVENTOUT	-
99	-	F5	D5	-	-	202	F6	V_{SS}	S		-		-
-	143	C3	C6	171	A11	203	E5	PDR_ON	S		-		-
100	144	K6	C5	172	D7	204	E7	V_{DD}	S		-		-
-	-	B3	D4	173	-	205	C3	PI4	I/O	FT	-	TIM8_BKIN, FMC_NBL2, DCMI_D5, LCD_B4, EVENTOUT	-
-	-	A3	C4	174	-	206	D3	PI5	I/O	FT	-	TIM8_CH1, FMC_NBL3. DCMI_VSYNC, LCD_B5, EVENTOUT	-
-	-	A2	C3	175	-	207	D6	PI6	I/O	FT	-	TIM8_CH2, FMC_D28, DCMI_D6, LCD_B6, EVENTOUT	-
-	-	B1	C2	176	-	208	D4	PI7	I/O	FT	-	TIM8_CH3, FMC_D29, DCMI_D7, LCD_B7, EVENTOUT	-

Appendix A STM32429I-EVAL I/O assignment

Table 36. STM32429I-EVAL I/O assignment

Pin number	Pin name	STM32429I-EVAL standard	Motor control variant	Camera variant
A1	PE4	A20/ TRACED1	-	-
A2	PE3	A19/ TRACED0	-	-
A3	PE2	TRACECLK	-	-
A4	PG14	MII_TXD1	-	-
A5	PE1	NBL1	-	-
A6	PE0	NBL0	-	-
A7	PB8	MII_TXD3	DISSIPATIVE_BRAKE	D6
A8	PB5	ULPI_D7	-	-
A9	PB4	JTRST	-	-
A10	PB3	JTDO-SWO/ I2S3_CK	-	-
A11	PD7	NE1	-	-
A12	PC12	SDCARD_CK/ SPI3_MOSI	-	D9
A13	PA15	JTDI	-	-
A14	PA14	JTCK-SWCLK	-	-
A15	PA13	JTMS-SWDIO	-	-
B1	PE5	A21/ TRACED2	-	-
B2	PE6	A22/ SAI1_SD_A/ TRACED3	-	D7
B3	PG13	MII_TXD0	-	-
B4	PB9	I2C1_SDA	-	-
B5	PB7		-	VSYNC
B6	PB6	I2C1_SCL	-	-
B7	PG15	SDNCAS	-	-
B8	PG11	MII_TX_EN	NTC_BYPASS_IO	-
B9	PJ13	LCD_B1	-	-
B10	PJ12	LCD_B0	-	-
B11	PD6	NWAIT/ I2S3_SD	-	D10

Table 36. STM32429I-EVAL I/O assignment (continued)

Pin number	Pin name	STM32429I-EVAL standard	Motor control variant	Camera variant
B12	PD0	D2	-	-
B13	PC11	SDCARD_D3/ SPI3_MISO	-	D4
B14	PC10	SDCARD_D2/ SPI3_SCK	-	D8
B15	PA12	CAN1_TX/ USB_FS1_DP	PFC_SYNC2	-
C1	VBAT	VBAT	-	-
C2	PI8- ANTI TAMP2	EXPANDER_INT	-	-
C3	PI4	NBL2	-	-
C4	PK7	LCD_DE	-	-
C5	PK6	LCD_B7	-	-
C6	PK5	LCD_B6	-	-
C7	PG12	LED4	-	-
C8	PG10	LED3	-	-
C9	PJ14	LCD_B2	-	-
C10	PD5NWE	-	-	-
C11	PD3	-	-	D5
C12	PD1	D3	-	-
C13	PI3	D27	-	-
C14	PI2	D26	-	-
C15	PA11	CAN1_RX/ USB_FS1_DM	PFC_PWM	
D1	PC13	KEY_TAMP_1	-	-
D2	PF0	A0	-	-
D3	PI5	NBL3	-	-
D4	PI7	D29	-	-
D5	PI10	D31/ MII_RX_ER	-	-
D6	PI6	D28	-	-
D7	PK4	LCD_B5	-	-
D8	PK3	LCD_B4	-	-
D9	PG9	NE2	-	-
D10	PJ15	LCD_B3	-	-
D11	PD4	NOE	-	-

Table 36. STM32429I-EVAL I/O assignment (continued)

Pin number	Pin name	STM32429I-EVAL standard	Motor control variant	Camera variant
D12	PD2	SDCARD_CMD	-	D11
D13	PH15	D23	-	-
D14	PI1	D25	-	-
D15	PA10	RS232_IRDA_RX/ USB_FS1_VBUS/ USB_FS1_ID	-	-
E1	PC14	OSC32_IN	-	-
E2	PF1	A1	-	-
E3	PI12	LCD_HSYNC	-	-
E4	PI9	D30	-	-
E5	PDR_ON	-	-	-
E6	BOOT0	BOOT0	-	-
E7	VDD_3	-	-	-
E8	VDD_11	-	-	-
E9	VDD_10	-	-	-
E10	VDD_15	-	-	-
E11	VCAP2	-	-	-
E12	PH13	D21	-	-
E13	PH14	D22	-	-
E14	PI0	D24	-	-
E15	PA9	RS232/ IRDA_USART1_TX/ USB_FS1_VBUS	-	-
F1	PC15	OSC32_OUT	-	-
F2	VSS_13_18	-	-	-
F3	PI11	ULPI_DIR		
F4	VDD_13	-	-	-
F5	VDD_17	-	-	-
F6	VSS_3	-	-	-
F7	VSS_11	-	-	-
F8	VSS_10	-	-	-
F9	VSS_15	-	-	-
F10	VSS 2	-	-	-
F11	VDD_2	-	-	-
F12	PK1	LCD_G6	-	-

Table 36. STM32429I-EVAL I/O assignment (continued)

Pin number	Pin name	STM32429I-EVAL standard	Motor control variant	Camera variant
F13	PK2	LCD_G7	-	-
F14	PC9	SDCARD_D1	-	D3
F15	PA8	LCD_BL_CTRL	PFC_SYNC1	-
G1	PH0	OSC_IN	-	-
G2	PF2	A2	-	-
G3	PI13	LCD_VSYNC	-	-
G4	PI15	LCD_R0	-	-
G5	VDD_18	-	-	-
G6	VSS_17	-	-	-
G10	VSS_9	-	-	-
G11	VDD_9	-	-	-
G12	PJ11	LCD_G4	-	-
G13	PK0	LCD_G5	-	-
G14	PC8	SDCARD_D0	WH	D2
G15	PC7	-	VH	D1
H1	PH1	OSC_OUT	-	-
H2	PF3	A3	-	-
H3	PI14	LCD_CLK	-	-
H4	PH4	ULPI_NXT	-	-
H5	VDD_5	-	-	-
H6	VSS_5	-	-	-
H10	VSS_20	-	-	-
H11	VDD_20	-	-	-
H12	PJ8	LCD_G1	-	-
H13	PJ10	LCD_G3	-	-
H14	PG8	SDCLK/ MII_PPS_OUT	-	-
H15	PC6	LCD_ON_OFF	UH	D0
J1	NRST	NRST	-	-
J2	PF4	A4	-	-
J3	PH5	SDNWE	-	-
J4	PH3	SDNE0/ MII_COL	-	-
J5	VDD_12	-	-	-
J6	VSS_12	-	-	-

Table 36. STM32429I-EVAL I/O assignment (continued)

Pin number	Pin name	STM32429I-EVAL standard	Motor control variant	Camera variant
J10	VSS_8	-	-	-
J11	VDD_8	-	-	-
J12	PJ7	LCD_G0	-	-
J13	PJ9	LCD_G2	-	-
J14	PG7	LED2	-	-
J15	PG6	LED1	-	-
K1	PF7	SAI1_MCLK_B	-	-
K2	PF6	SAI1_SD_B	-	-
K3	PF5	A5	-	-
K4	PH2	SDCKE0	-	-
K5	VDD_4	-	-	-
K6	VSS_4	-	-	-
K7	VSS_6	-	-	-
K8	VSS_7	-	-	-
K9	VSS_1	-	-	-
K10	VSS_14	-	-	-
K11	VDD_14	-	-	-
K12	PJ6	LCD_R7		
K13	PD15	D1	-	-
K14	PB13	ULPI_D6/ USB_FS2_VBUS	-	-
K15	PD10	D15	-	-
L1	PF10	Potentiometer	-	-
L2	PF9	SAI1_FS_B	-	-
L3	PF8	SAI1_SCK_B	-	-
L4	PC3	MII_TX_CLK	CURRENT_C	-
L5	BYPASS_REG	-	-	-
L6	VSS_19	-	-	-
L7	VDD_19	-	-	-
L8	VDD_6	-	-	-
L9	VDD_7	-	-	-
L10	VDD_1	-	-	-
L11	VCAP1	-	-	-
L12	PD14	D0	-	-

Table 36. STM32429I-EVAL I/O assignment (continued)

Pin number	Pin name	STM32429I-EVAL standard	Motor control variant	Camera variant
L13	PB12	ULPI_D5/ USB_FS2_ID	-	-
L14	PD9	D14	-	-
L15	PD8	D13	-	-
M1	VSSA	-	-	-
M2	PC0	ULPI_STP	-	-
M3	PC1	MII_MDC	-	-
M4	PC2	MII_TXD2	CURRENT_B	-
M5	PB2	BOOT1	-	-
M6	PF12	A6	-	-
M7	PG1	A11	-	-
M8	PF15	A9	-	-
M9	PJ4	LCD_R5	-	-
M10	PD12	A17	-	-
M11	PD13	A18	-	-
M12	PG3	A13	-	-
M13	PG2	A12	-	-
M14	PJ5	LCD_R6	-	-
M15	PH12	D20	-	-
N1	VREF-	-	-	-
N2	PA1	MII_RX_CLK	ENCODER_B	-
N3	PA0	KEY_WKUP/ MII_CRS	ENCODER_A	-
N4	PA4	-	DAC_OUT1	HSYNC
N5	PC4	MII_RXD0	BUSVOLTAGE	-
N6	PF13	A7	-	-
N7	PG0	A10	-	-
N8	PJ3	LCD_R4	-	-
N9	PE8	D5	-	-
N10	PD11	A16	-	-
N11	PG5	A15/ BA1	-	-
N12	PG4	A14/ BA0	-	-
N13	PH7	MII_RXD3	-	-

Table 36. STM32429I-EVAL I/O assignment (continued)

Pin number	Pin name	STM32429I-EVAL standard	Motor control variant	Camera variant
N14	PH9	D17	-	-
N15	PH11	D19	-	-
P1	VREF+	-	-	-
P2	PA2	MII_MDIO	INDEX	-
P3	PA6	-	STOP	PIXCK
P4	PA5	ULPI_CK	DAC_OUT2	-
P5	PC5	MII_RXD1	HEATSINK	-
P6	PF14	A8	-	-
P7	PJ2	LCD_R3	-	-
P8	PF11	SDNRAS	-	-
P9	PE9	D6	-	-
P10	PE11	D8	-	-
P11	PE14	D11	-	-
P12	PB10	ULPI_D3	-	-
P13	PH6	MII_RXD2	-	-
P14	PH8	D16	-	-
P15	PH10	D18	-	-
R1	VDDA	-	-	-
R2	PA3	ULPI_D0	-	-
R3	PA7	MII_RX_DV	UL	-
R4	PB1	ULPI_D2	WL	-
R5	PB0	ULPI_D1	VL	-
R6	PJ0	LCD_R1	-	-
R7	PJ1	LCD_R2	-	-
R8	PE7	D4	-	-
R9	PE10	D7	-	-
R10	PE12	D9	-	-
R11	PE15	D12	-	-
R12	PE13	D10	-	-
R13	PB11	ULPI_D4	-	-
R14	PB14	USB_FS2_DM	-	-
R15	PB15	USB_FS2_DP	-	-