Video Link: https://youtu.be/I7QGx03Almo

Keypad.h

```
/*
 * keypad.h
 *
 *  Created on: Sep 29, 2021
 *      Author: Sereen
 */

#ifndef SRC_KEYPAD_H_
#define SRC_KEYPAD_H_

void keypad_init(void);
uint8_t read_keypad(void);

#define STAR   10
#define POUND  11
#define NO_KEY 0xFF

// Ports PB4 - PB7
#define ROW1 GPIO_IDR_ID4
#define ROW2 GPIO_IDR_ID5
#define ROW3 GPIO_IDR_ID6
#define ROW4 GPIO_IDR_ID7

#define ROW_PORT (GPIOA->IDR)
#define COL_PORT (GPIOC->ODR)

#define COL1 GPIO_ODR_OD5
#define COL2 GPIO_ODR_OD6
#define COL3 GPIO_ODR_OD7

#define COL_MASK (COL1|COL2|COL3)
#define ROW_MASK (ROW1|ROW2|ROW3|ROW4)


#endif /* SRC_KEYPAD_H_ */
```

Keypad.c

```
#include "main.h"
#include "keypad.h"


 void keypad_init(void)
 {
       RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOAEN); // enable GPIOB clock on bus
       RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOCEN); // enable GPIOC clock on bus

    // clear GPIOA PA4-PA7 (also sets them for input
       GPIOA->MODER &= ~(    GPIO_MODER_MODE4
                           | GPIO_MODER_MODE5
                           | GPIO_MODER_MODE6
                           | GPIO_MODER_MODE7);
       // clear MODE PC5 - PC7  bits for keypad and use as columns
       GPIOC->MODER &= ~(GPIO_MODER_MODE5
                           | GPIO_MODER_MODE6
                           | GPIO_MODER_MODE7);

       // set PC5-PC7 as outputs for columns
       GPIOC->MODER |= ( (1 << GPIO_MODER_MODE5_Pos)
                         | (1 << GPIO_MODER_MODE6_Pos)
                         | (1 << GPIO_MODER_MODE7_Pos) );

       // enable pulldown resistor for rows on PA4-7

       // clear pupdr
       GPIOA->PUPDR &=  ~(    GPIO_PUPDR_PUPD4_1
                           | GPIO_PUPDR_PUPD5_1
                           | GPIO_PUPDR_PUPD6_1
                           | GPIO_PUPDR_PUPD7_1);


       GPIOA->PUPDR |=  (    GPIO_PUPDR_PUPD4_1
                           | GPIO_PUPDR_PUPD5_1
                           | GPIO_PUPDR_PUPD6_1
                           | GPIO_PUPDR_PUPD7_1);

       // enable push-pull for columns on PC5-PC7
       // (check later if there is problems)
```

```
	GPIOC->OTYPER &= ~(	GPIO_OTYPER_OT5
					| GPIO_OTYPER_OT6
					| GPIO_OTYPER_OT7);


	// set slow speed for columns (PC5-PC7)

	GPIOC->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED5
						| GPIO_OSPEEDR_OSPEED6
						| GPIO_OSPEEDR_OSPEED7); // PA0
slow speed

	// set columns to high
	COL_PORT |= COL_MASK;

}


uint8_t calculate_key(uint16_t col, uint8_t row)
{
	uint8_t key;
	uint8_t rows;

	rows = row >> 4; // right shift rows 4 places for easiers calc

	if(rows == 4)
		rows = 3;
	if(rows == 8)
		rows = 4;
	// calculate key based on col
	switch(col)
	{
		case 0:
			key = (3 * rows) - 2;
			break;
		case 1:
			key = (3 * rows) - 1;
			break;
		case 2:
			key = (3 * rows);
			break;
	}
```

```
//    if(key == 10)   // leave as 10 so it can output to LED (change with
LCD)
      //    key  = '*'; // leave it in ASCII for later use
      if(key == 11)
            key = 0;
//    if(key == 12)  // leave as 12 so so it can output to LED (change with
LCD)
      //    key = '#';

      return key;
}

void check_columns(uint8_t cur_col)
{
      COL_PORT  &= ~(COL_MASK); // turn all colums off
      switch(cur_col)  // set a col high depending on the row
      {
            case 0:
                  COL_PORT |= COL1;
                  break;
            case 1:
                  COL_PORT |= COL2;
                  break;
            case 2:
                  COL_PORT |= COL3;
                  break;

      }
}


uint8_t read_keypad(void)
{
      uint8_t rows;
      uint8_t cur_col ;
      uint8_t key;
      // Read the rows PB4-PB7 only
      rows = ROW_PORT & ROW_MASK ;

      if(rows == 0) // check to see if all the rows are low (is so return
NO KEY)
            return NO_KEY;
```

```
        for(cur_col = 0; cur_col < 3; cur_col++)
        {

              // set current columns high others low
              check_columns(cur_col);
              // read the rows
            rows = ROW_PORT & ROW_MASK;

              if(rows != 0)
              {
                    // calculate button from row and col
                    key = calculate_key(cur_col, rows);
                    // set columns to high
                    COL_PORT |= COL_MASK;
                    return key;
              }

        }

        // set columns to high
        COL_PORT |= COL_MASK;

        return NO_KEY;


}
```

LED.h

```
 /*
  * LED.h
  *
  *  Created on: Oct 1, 2021
  *      Author: Sereen
  */

 #ifndef SRC_LED_H_
 #define SRC_LED_H_
```

```
void count(uint8_t num);
void set_4bitLED(void);
void set_bit0LED(void);
void set_bit1LED(void);
void set_bit2LED(void);
void set_bit3LED(void);

#endif /* SRC_LED_H_ */
```

LED.c

```
/*
 * LED.c
 *
 *  Created on: Oct 1, 2021
 *      Author: Sereen
 */
#include "main.h"
#include "LED.h"

void count(uint8_t num)
{
      GPIOC->ODR &= ~(GPIO_ODR_OD0 | GPIO_ODR_OD1 | GPIO_ODR_OD2 |
GPIO_ODR_OD3); //resets registers in ODR
      GPIOC->ODR |= num; // sets the num in the ODR to displays to 4 LEDs

}

// Initializes the PINS to output as a 4 bit LED
void set_4bitLED(void)
{
   RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOCEN); // enable GPIOA clock on bus
   set_bit0LED();
   set_bit1LED();
   set_bit2LED();
   set_bit3LED();
```

```
}
// initialzes the first LED as a 1st bit representation
void set_bit0LED(void)
{
  GPIOC->MODER &= ~(GPIO_MODER_MODE0);   // clear MODE0 bits
  GPIOC->MODER |= (1 << GPIO_MODER_MODE0_Pos); // set PA0 as GPIO output
  GPIOC->OTYPER &= ~(GPIO_OTYPER_OT0);        // PA0 set to push-pull
  GPIOC->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED0); // PA0 slow speed
  GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD0);  // PA0 no pullup/pulldown resistor
}

// initializes the second LED as a 2nd bit representation
void set_bit1LED(void)
{
  GPIOC->MODER &= ~(GPIO_MODER_MODE1); // clear MODE1 bits
  GPIOC->MODER |= (1 << GPIO_MODER_MODE1_Pos); // set PA1 as GPIO output
  GPIOC->OTYPER &= ~(GPIO_OTYPER_OT1); // PA1 set to push-pull
  GPIOC->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED1); // PA1 slow speed
  GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD1);  // PA1 no pullup/pulldown resistor
}

// initializes the third LED as a third bit representation
void set_bit2LED(void)
{
  GPIOC->MODER &= ~(GPIO_MODER_MODE2);  // clear MODE2 bits
  GPIOC->MODER |= (1 << GPIO_MODER_MODE2_Pos); // set PA2 as GPIO output
  GPIOC->OTYPER &= ~(GPIO_OTYPER_OT2);  // PA2 set to push-pull
  GPIOC->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED2);  // PA2 slow speed
  GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD2);  //PA2 no pullup/pulldown resistor
}

// initializes the fourth  LED as a 4th bit representation
void set_bit3LED(void)
{
  GPIOC->MODER &= ~(GPIO_MODER_MODE3);  // clear MODE3 bits
  GPIOC->MODER |= (1 << GPIO_MODER_MODE3_Pos); // set PA3 as GPIO output
  GPIOC->OTYPER &= ~(GPIO_OTYPER_OT3); // PA3 set to push-pull
  GPIOC->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEED3); // PA3 slow speed
  GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD3); // PA3 no pullup/pulldown resistor
}
```

Main.c

```c
#include "main.h"
#include "keypad.h"
#include "LED.h"
#include <stdio.h>


void SystemClock_Config(void);


int main(void)
{

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();
  uint8_t key;

  /* Configure the system clock */
  SystemClock_Config();

  keypad_init();
  set_4bitLED(); // initializes Ports PC0 - PC3 for LED output in binary

  while (1)
  {
        key = read_keypad(); //retrieve key

        if(key != NO_KEY)
              count(key);  // display key in binary on LED display

  }

}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
```

```
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) !=
HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
```

```
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

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