CPE 316

Microcontrollers and Embedded Applications

keypad program revisit, UART position, TIM2 use

Prof Peter Han

Objectives

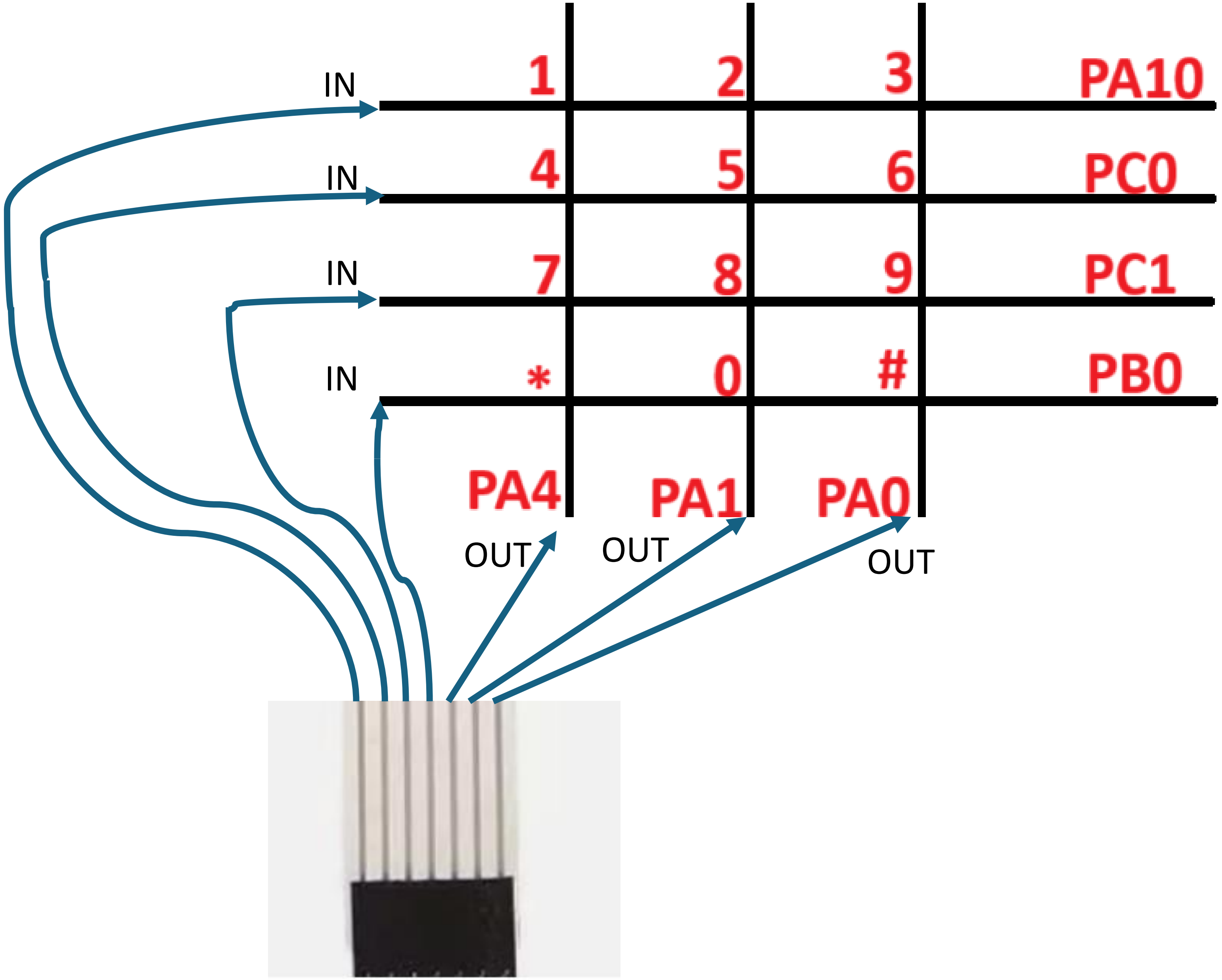
1. Review the Keypad program with a flow chart.

2. Learn how to move the cursor to a location on a Terminal window.
   VT100 command

1. Learn how to use Timer interrupt and its callback function. How to setup prescaler. How to set up ARR.

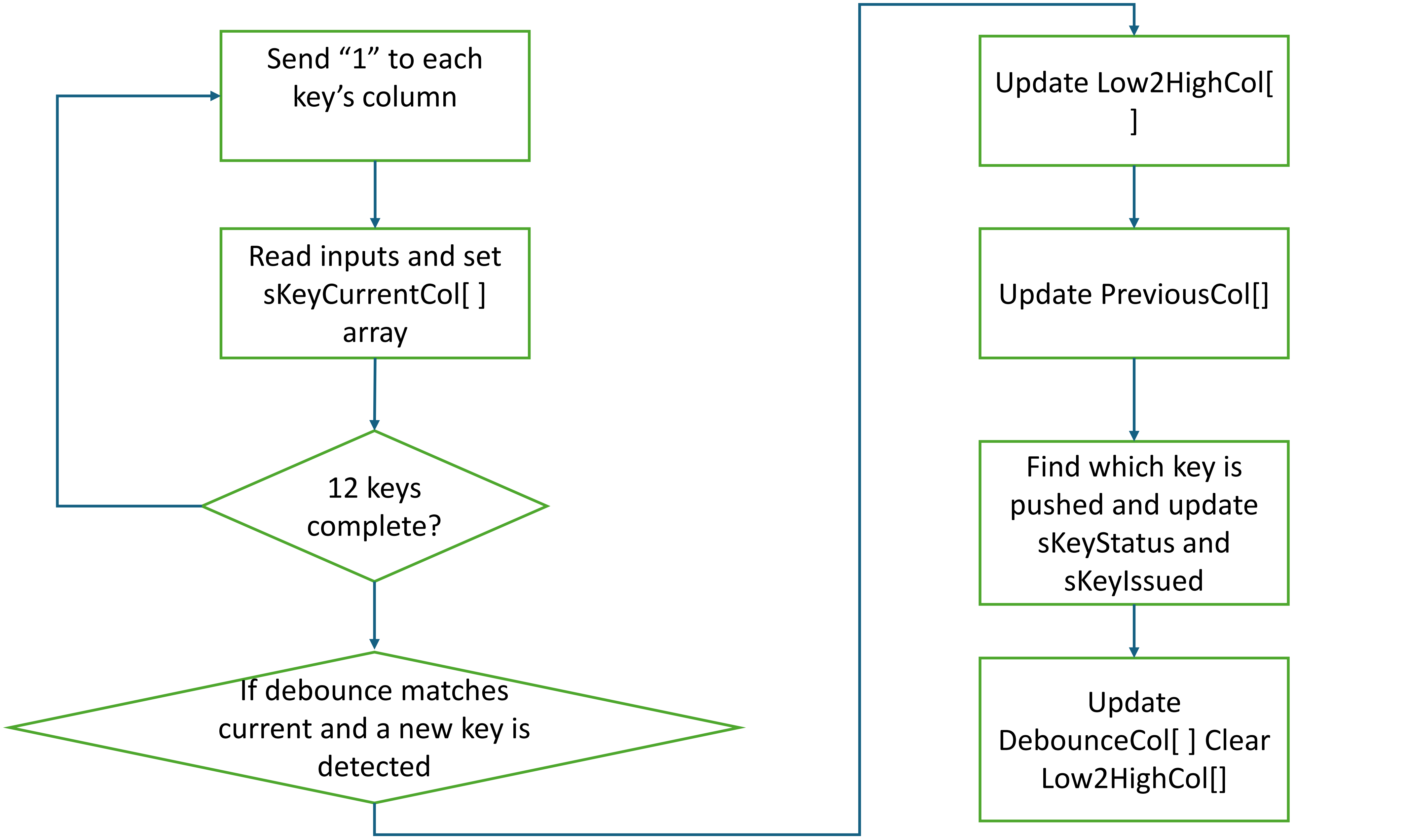
2. Learn how to use Timer to generate delay

Keypad connection diagram



IN	1	2	3	PA10
IN	4	5	6	PC0
IN	7	8	9	PC1
IN	*	0	#	PB0
	PA4	PA1	PA0	
	OUT	OUT	OUT	

Keypad scanning review



Send "1" to each key's column

Read inputs and set sKeyCurrentCol[ ] array

12 keys complete?

If debounce matches current and a new key is detected

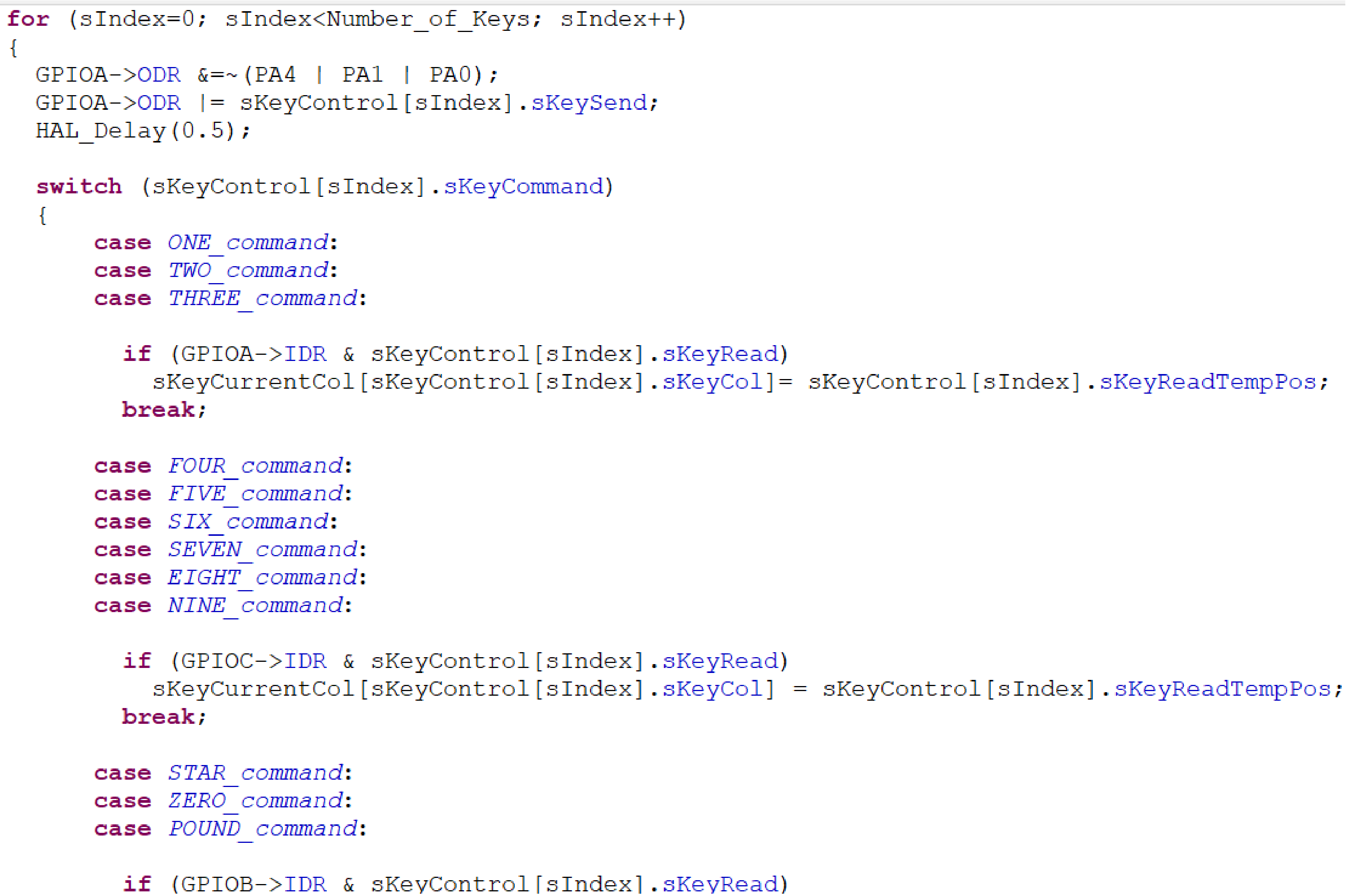
Update Low2HighCol[ ]

Update PreviousCol[]

Find which key is pushed and update sKeyStatus and sKeyIssued

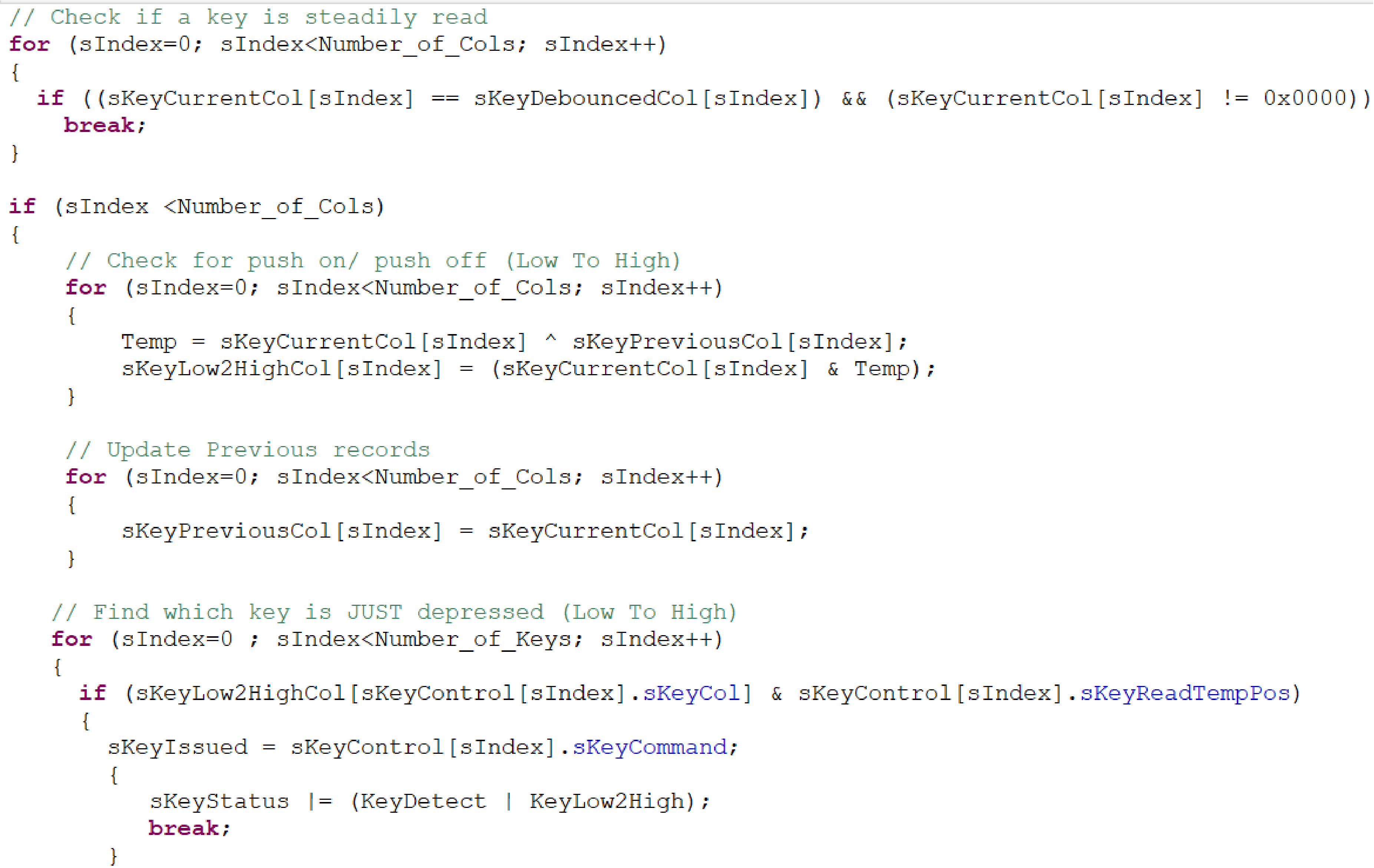
Update DebounceCol[ ] Clear Low2HighCol[]

Loop through each key and update their associated sKeyCurrentCol[ ] variables.



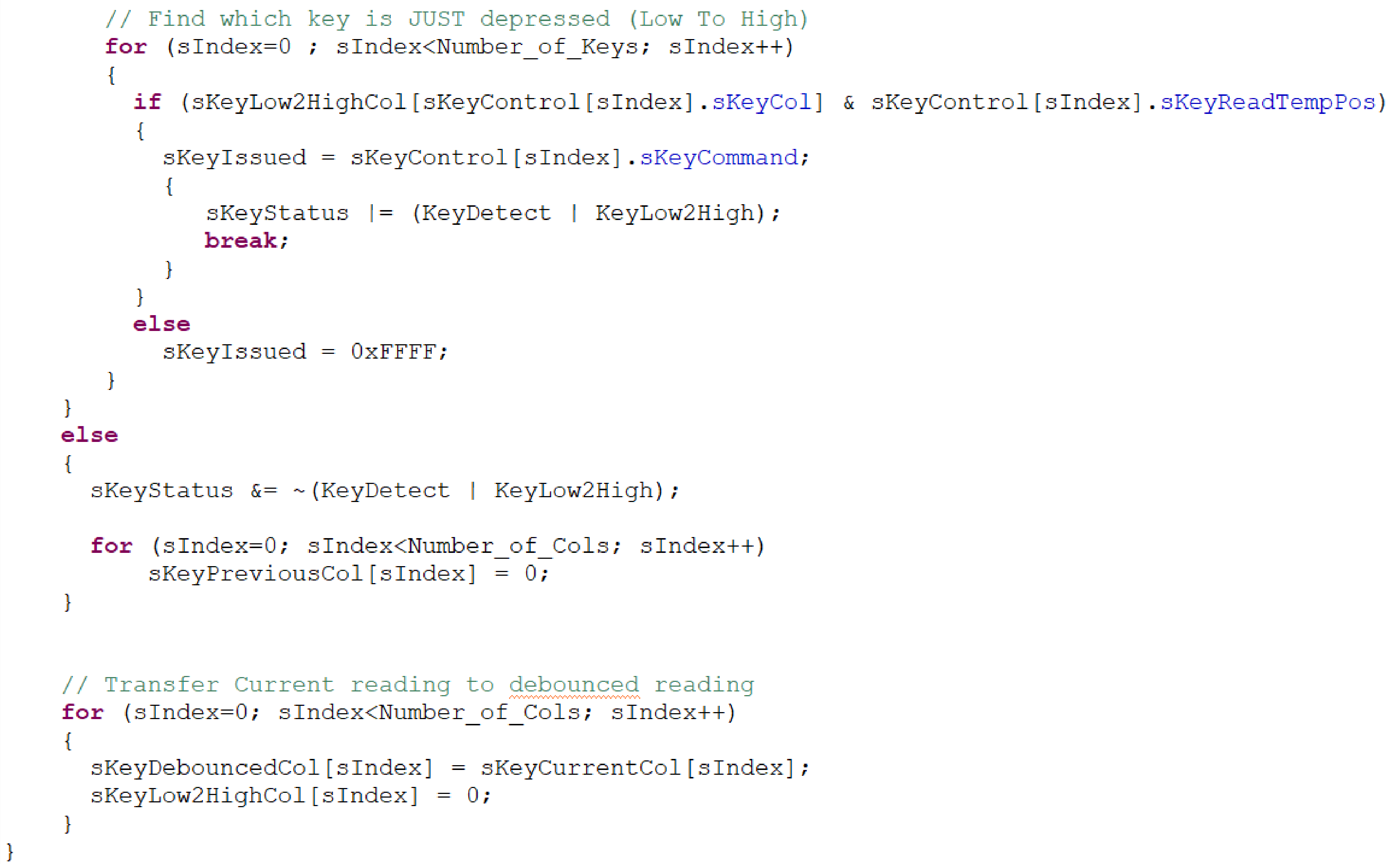
```
for (sIndex=0; sIndex<Number_of_Keys; sIndex++)
{
 GPIOA->ODR &=~(PA4 | PA1 | PA0);
 GPIOA->ODR |= sKeyControl[sIndex].sKeySend;
 HAL_Delay(0.5);

 switch (sKeyControl[sIndex].sKeyCommand)
 {
   case ONE_command:
   case TWO_command:
   case THREE_command:

    if (GPIOA->IDR & sKeyControl[sIndex].sKeyRead)
     sKeyCurrentCol[sKeyControl[sIndex].sKeyCol]= sKeyControl[sIndex].sKeyReadTempPos;
    break;

   case FOUR_command:
   case FIVE_command:
   case SIX_command:
   case SEVEN_command:
   case EIGHT_command:
   case NINE_command:

    if (GPIOC->IDR & sKeyControl[sIndex].sKeyRead)
     sKeyCurrentCol[sKeyControl[sIndex].sKeyCol] = sKeyControl[sIndex].sKeyReadTempPos;
    break;

   case STAR_command:
   case ZERO_command:
   case POUND_command:

    if (GPIOB->IDR & sKeyControl[sIndex].sKeyRead)
```

If a key is debounced and non-trivial, update their sKeyLow2HighCol[ ] variables and status flags



```
// Check if a key is steadily read
for (sIndex=0; sIndex<Number_of_Cols; sIndex++)
{
  if ((sKeyCurrentCol[sIndex] == sKeyDebouncedCol[sIndex]) && (sKeyCurrentCol[sIndex] != 0x0000))
    break;
}

if (sIndex <Number_of_Cols)
{
    // Check for push on/ push off (Low To High)
    for (sIndex=0; sIndex<Number_of_Cols; sIndex++)
    {
        Temp = sKeyCurrentCol[sIndex] ^ sKeyPreviousCol[sIndex];
        sKeyLow2HighCol[sIndex] = (sKeyCurrentCol[sIndex] & Temp);
    }

    // Update Previous records
    for (sIndex=0; sIndex<Number_of_Cols; sIndex++)
    {
        sKeyPreviousCol[sIndex] = sKeyCurrentCol[sIndex];
    }

   // Find which key is JUST depressed (Low To High)
   for (sIndex=0 ; sIndex<Number_of_Keys; sIndex++)
   {
     if (sKeyLow2HighCol[sKeyControl[sIndex].sKeyCol] & sKeyControl[sIndex].sKeyReadTempPos)
     {
       sKeyIssued = sKeyControl[sIndex].sKeyCommand;
       {
          sKeyStatus |= (KeyDetect | KeyLow2High);
          break;
       }
```

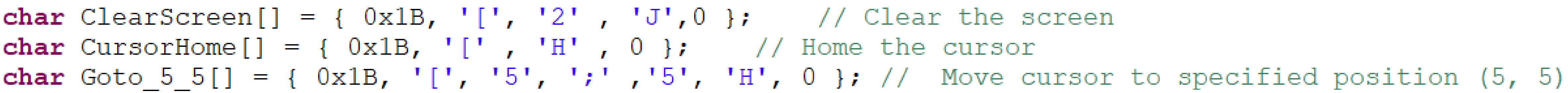
Update sKeyIssued with accepted keycommand. Otherwise, reset Previous and update DebouncedCol.



```
       // Find which key is JUST depressed (Low To High)
       for (sIndex=0 ; sIndex<Number_of_Keys; sIndex++)
       {
         if (sKeyLow2HighCol[sKeyControl[sIndex].sKeyCol] & sKeyControl[sIndex].sKeyReadTempPos)
         {
           sKeyIssued = sKeyControl[sIndex].sKeyCommand;
           {
              sKeyStatus |= (KeyDetect | KeyLow2High);
              break;
           }
         }
         else
           sKeyIssued = 0xFFFF;
       }
    }
    else
    {
      sKeyStatus &= ~(KeyDetect | KeyLow2High);

      for (sIndex=0; sIndex<Number_of_Cols; sIndex++)
          sKeyPreviousCol[sIndex] = 0;
    }


    // Transfer Current reading to debounced reading
    for (sIndex=0; sIndex<Number_of_Cols; sIndex++)
    {
      sKeyDebouncedCol[sIndex] = sKeyCurrentCol[sIndex];
      sKeyLow2HighCol[sIndex] = 0;
    }
}
```

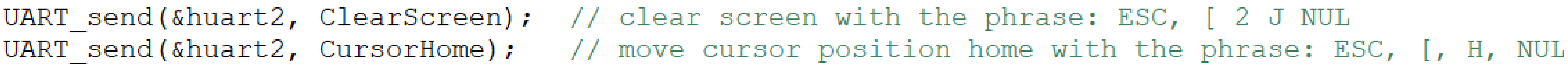
Use VT100 command to move cursor position and clear screen

Define the string for Terminal control (VT100).



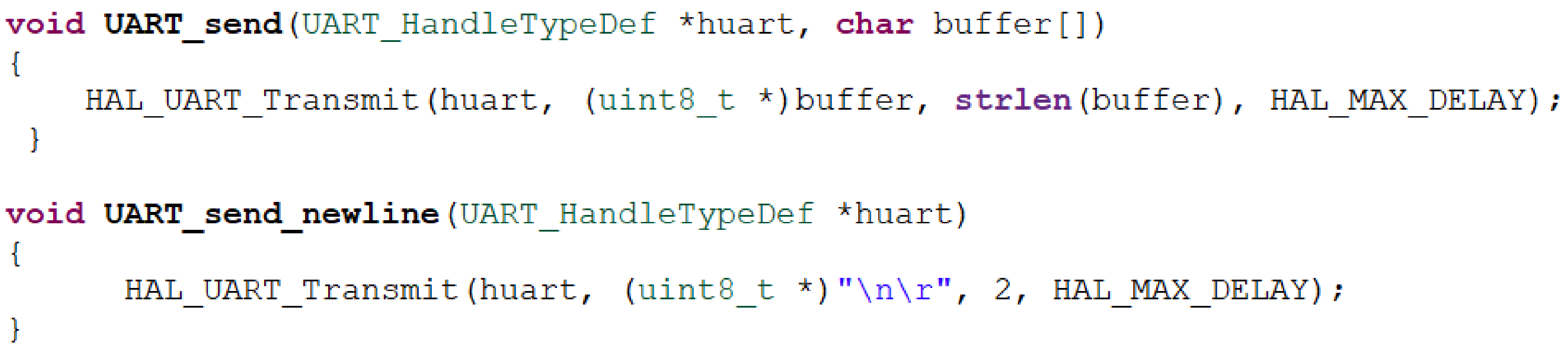
```
char ClearScreen[] = { 0x1B, '[', '2' , 'J',0 };    // Clear the screen
char CursorHome[] = { 0x1B, '[' , 'H' , 0 };    // Home the cursor
char Goto_5_5[] = { 0x1B, '[', '5', ';' ,'5', 'H', 0 }; //  Move cursor to specified position (5, 5)
```

Use the function call to clear the terminal screen or get the cursor to its home position.



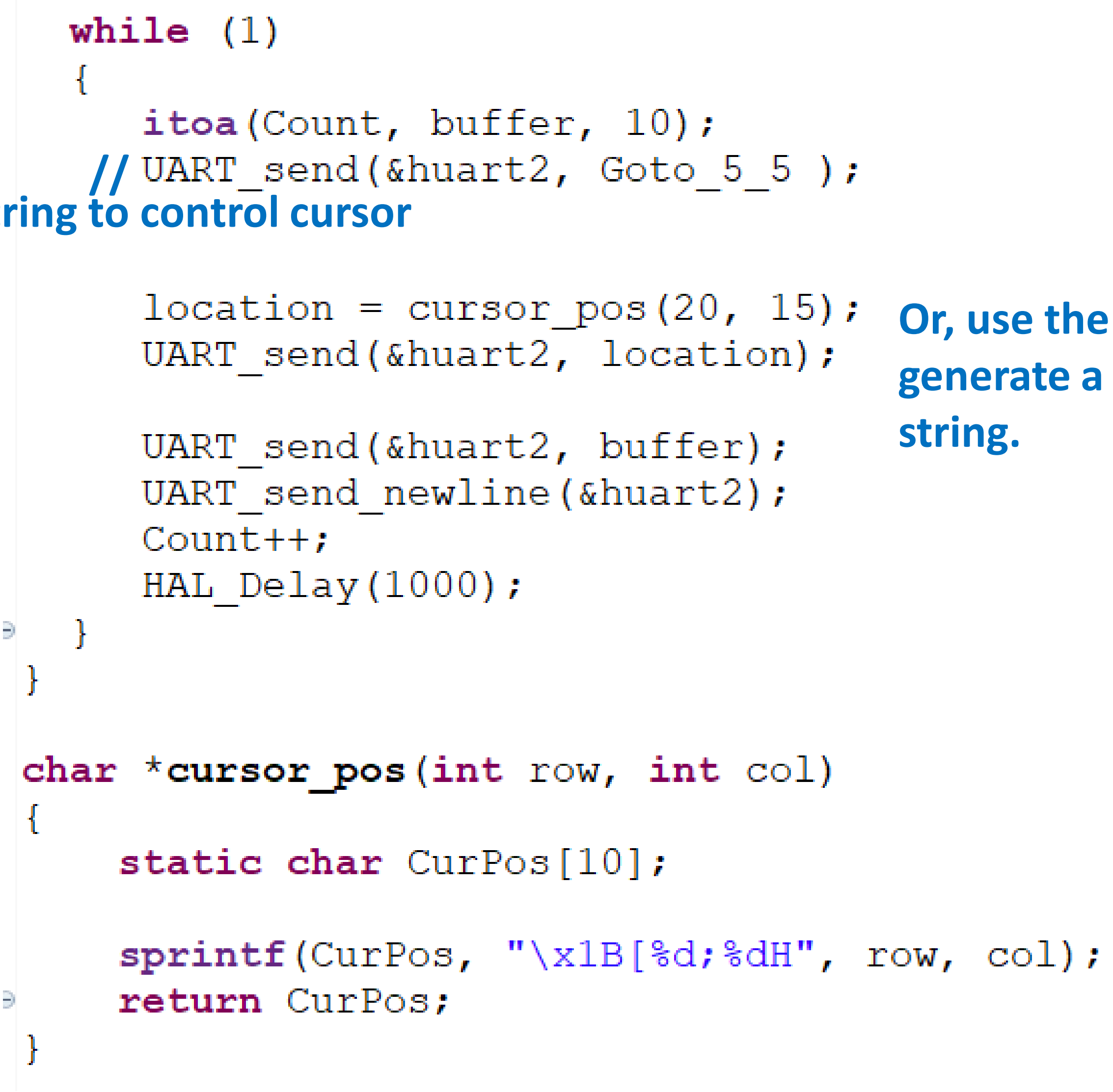
```
UART_send(&huart2, ClearScreen);  // clear screen with the phrase: ESC, [ 2 J NUL
UART_send(&huart2, CursorHome);   // move cursor position home with the phrase: ESC, [, H, NUL
```

Implement the helper functions below.



```
void UART_send(UART_HandleTypeDef *huart, char buffer[])
{
    HAL_UART_Transmit(huart, (uint8_t *)buffer, strlen(buffer), HAL_MAX_DELAY);
 }

void UART_send_newline(UART_HandleTypeDef *huart)
{
      HAL_UART_Transmit(huart, (uint8_t *)"\n\r", 2, HAL_MAX_DELAY);
}
```

The VT100 command is char based. Create a helper function for cursor pos.

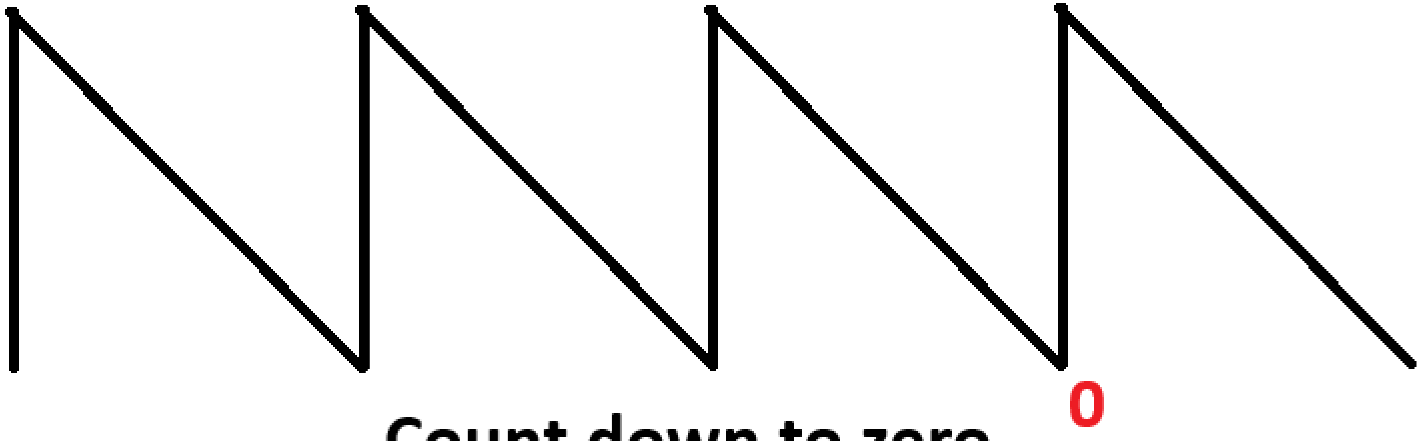


```
  while (1)
  {
     itoa(Count, buffer, 10);
  // UART_send(&huart2, Goto_5_5 );

     location = cursor_pos(20, 15);
     UART_send(&huart2, location);

     UART_send(&huart2, buffer);
     UART_send_newline(&huart2);
     Count++;
     HAL_Delay(1000);
  }
}

char *cursor_pos(int row, int col)
{
    static char CurPos[10];

    sprintf(CurPos, "\x1B[%d;%dH", row, col);
    return CurPos;
}
```

Use a predefined string to control cursor location.

Or, use the function below to generate a cursor position string.

Interrupt and use Timer 2 to generate 1ms periodically



Re-load the same value

0

Count down to zero

When reaching zero, interrupt will be generated.

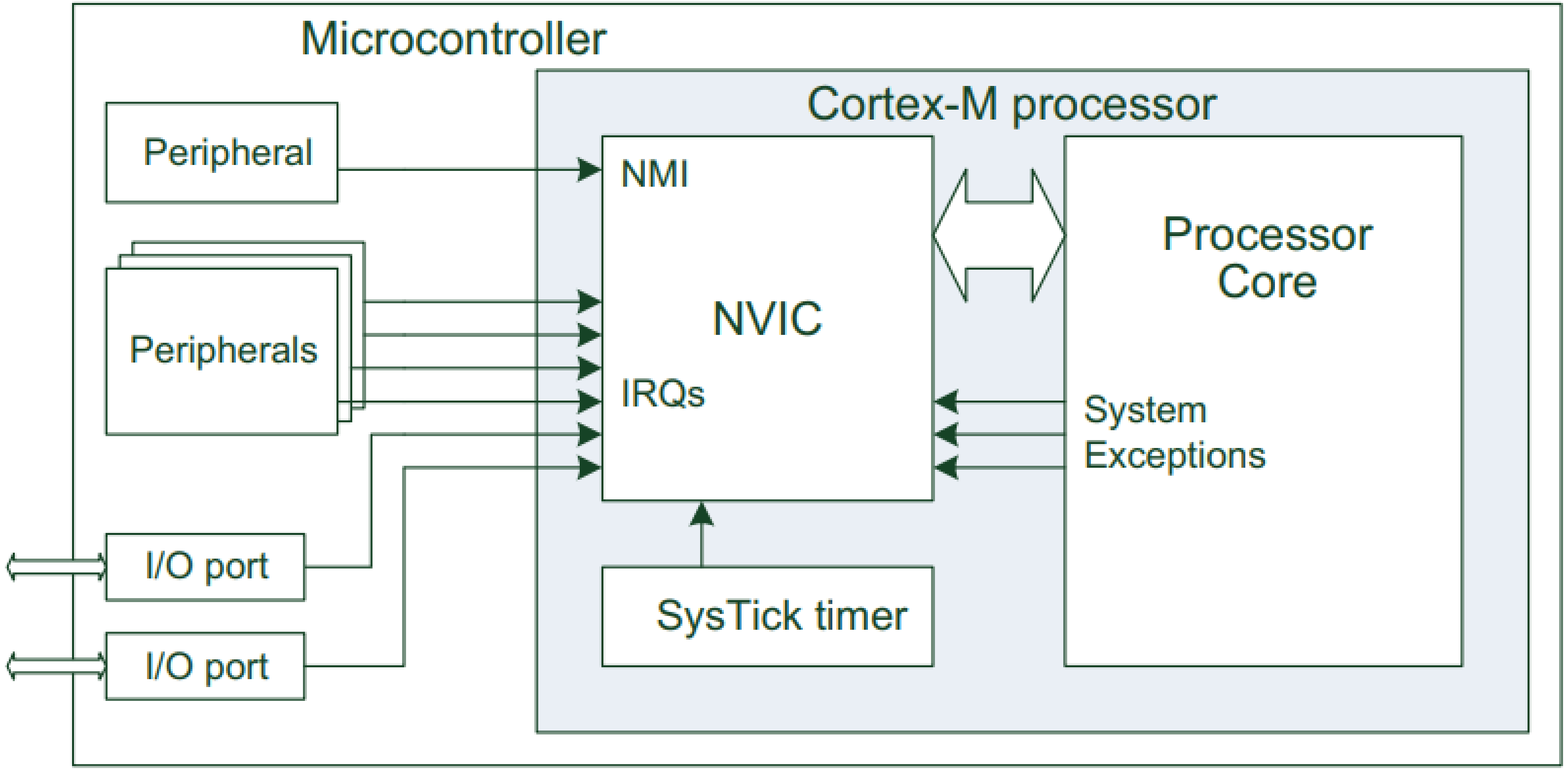
There are five different types of timers:
Basic (16 bits)
General-Purpose (16 or 32 bits)
Advanced
High resolution
Low power

What timers are in your microcontroller?

- 2×16-bit advanced control timers (TIM1 and TIM8);
- 2× 32-bit general-purpose timers (**TIM2** and TIM5);
- 5× 16-bit general-purpose timers (TIM3, TIM4, TIM15, TIM16, TIM17);
- 2× 16-bit basic timer (TIM6 and TIM7);
- 2× low-power 16-bit timers (LPTIM1 and LPTIM2);
- 2× watchdog timers;
- 1× SysTick timer.

What is Interrupt?

NVIC (Nested Vector Interrupt Controller) supports up to 240 IRQs (Interrupt Requests), including:
* a Non-Maskable Interrupt (NMI)
* a SysTick (System Tick) timer interrupt
* a number of system exceptions.
* IRQs by peripherals such as timers, I/O ports, and communication interfaces (e.g., UART).
* NMI is usually generated from peripherals like a watchdog timer or Brown-Out Detector.

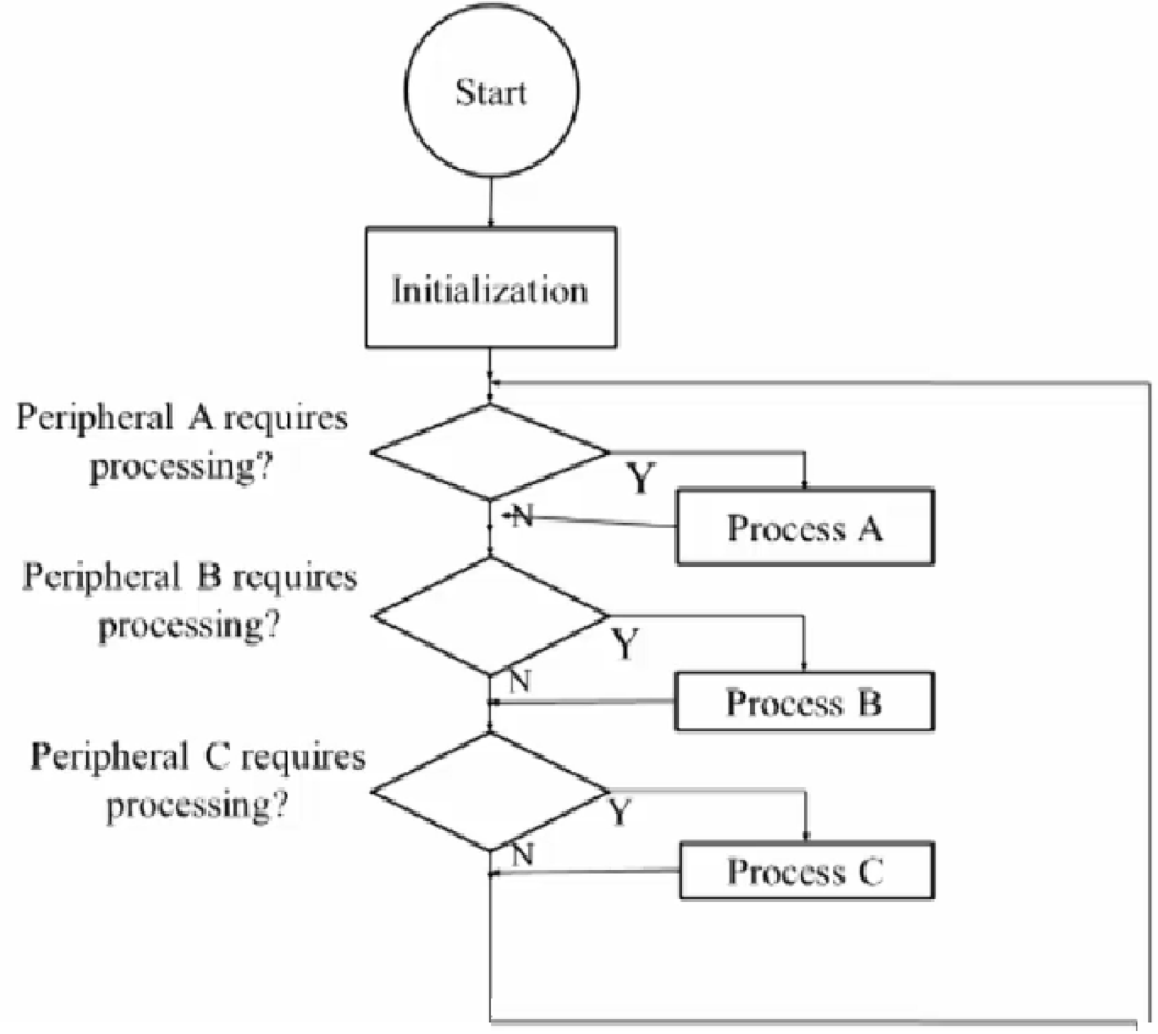


What is Polling? (not interrupt)

As shown in the diagram, the CPU needs to poll each peripheral to see if it needs service.

If a service takes too much time, the next service will be delayed.

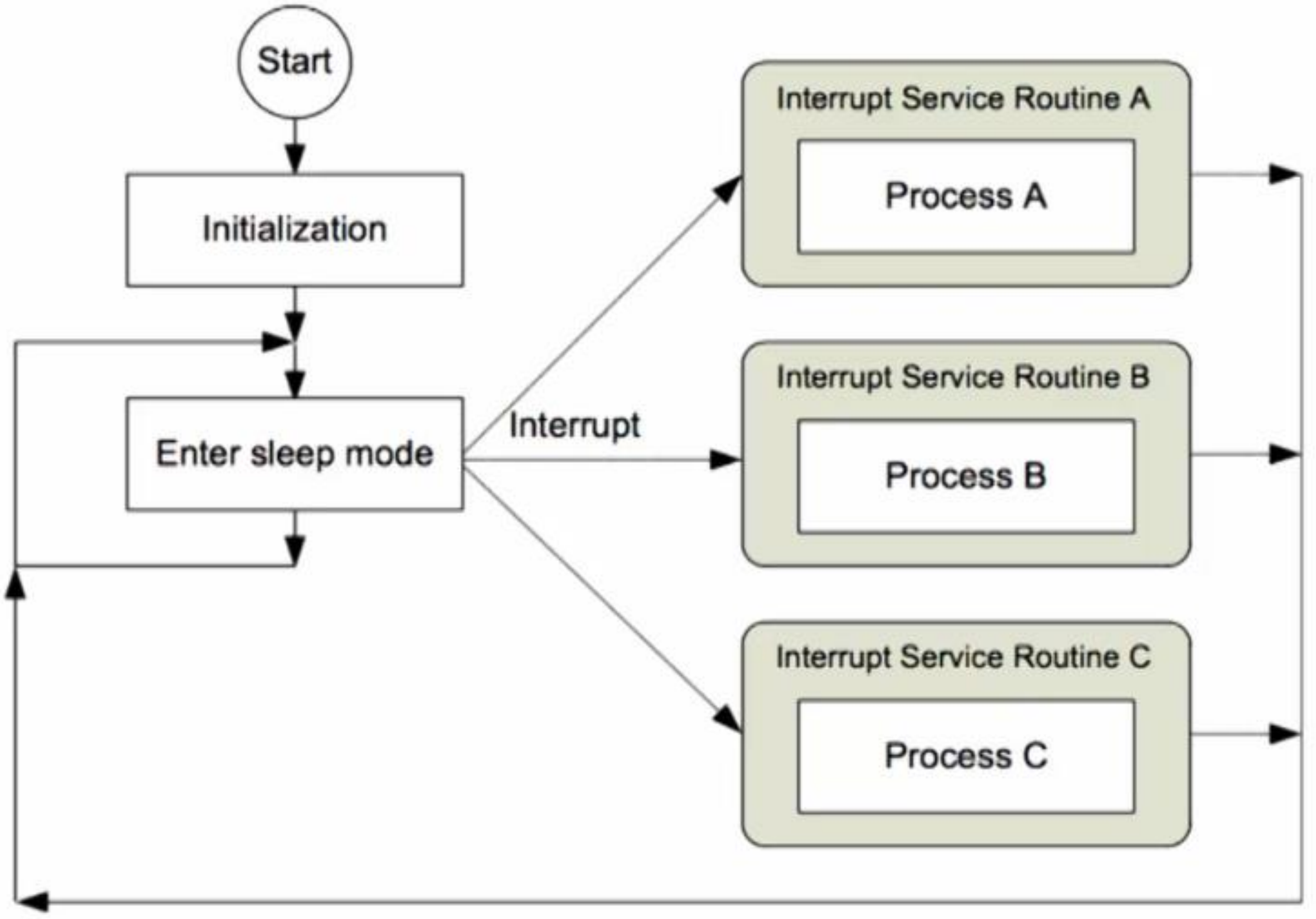
Interrupt can solve problems like this.


Start
Initialization
Peripheral A requires processing?
Y
N
Process A
Peripheral B requires processing?
Y
N
Process B
Peripheral C requires processing?
Y
N
Process C

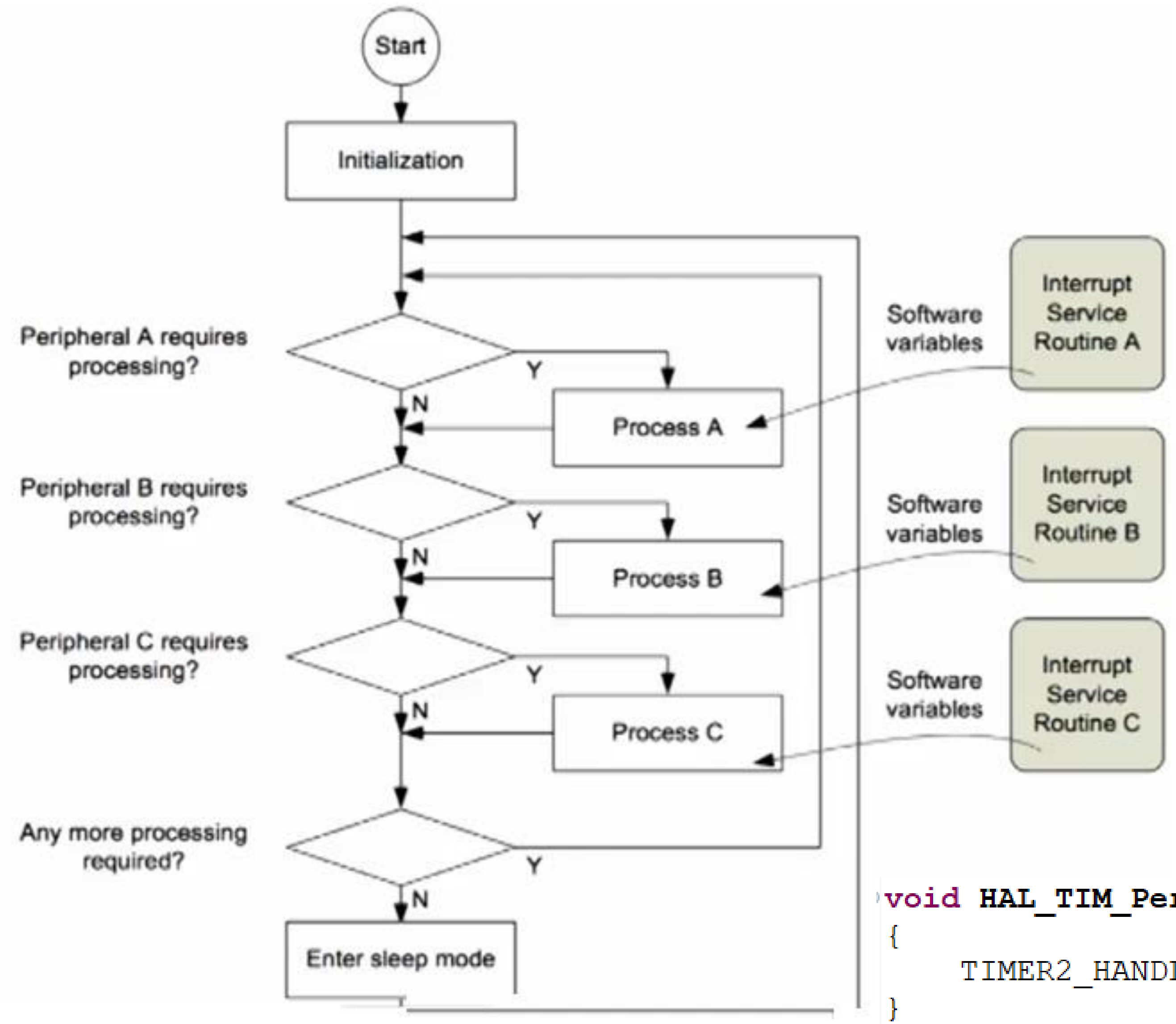
Should we let each interrupt to handle a complete process?

Let the CPU goes to sleep mode.

Should each INT complete its request inside each ISR callback function?


Start
Initialization
Enter sleep mode
Interrupt
Interrupt Service Routine A
Process A
Interrupt Service Routine B
Process B
Interrupt Service Routine C
Process C

How does interrupt work?


Start
Initialization
Peripheral A requires processing?
Y
N
Process A
Peripheral B requires processing?
Y
N
Process B
Peripheral C requires processing?
Y
N
Process C
Any more processing required?
Y
N
Enter sleep mode
Software variables
Interrupt Service Routine A
Software variables
Interrupt Service Routine B
Software variables
Interrupt Service Routine C

We can use interrupt to set flags and updates in software variables then go back to process. (Keep short in **ISR.** (Interrupt Service Routine.)

This is the TIM callback (ISR) function.

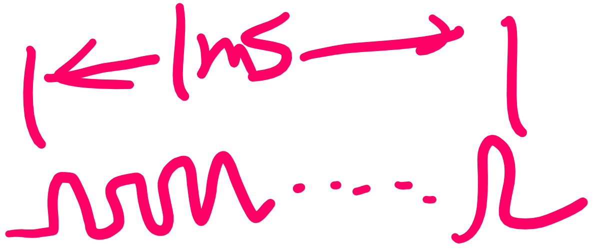
```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    TIMER2_HANDLE();
}
```

How to set up TIM2 INT

$$Reload\ Value = \frac{Clock_{freq} \times Time\_to\_INT}{(Prescaler + 1)} - 1$$

$Reload\ Value: It\ is\ the\ value\ to\ be\ loaded\ into\ the\ auto\text{-}reload\ register$
$Prescaler$: (1 ~ 65536) 16-bit value.
$Time\_to\_INT$: required interrupt time in seconds.

PSC



If we want to make a scheduler in 1ms, the reload value is below.
(Use Prescaler 3999.,)

$$Reload\ Value = \frac{80000000 \times 0.001}{(3999 + 1)} - 1 = 19$$

How to set up TIM2 INT

Step 1: Click Timers, then click TIM2, and select the Clock Source as Internal clock.

Step 2: In configuration, click on Prescaler (PSC – 16-bit value) and make sure it is set to Decimal and change its value to 3999.

Step 3: Change the Counter Mode to Down.

Step 4: Click on Counter Period (Auto reload register) and make sure it is set to Decimal. Change its value to 19.

Step 5: Click Internal Clock Division and set it to 2. (Note: this is not clock division. It is to relate the filter for the input sampler).

Step 6: Click auto-reload preload and click to Enable it.

Step 7: Click System Core followed by NVIC. Click to enable TIM2 global interrupt.

How to set up TIM2 INT (cont’d)

Step 8: Click File, followed by Save and click YES to generate code.

This handler below should have been added.

```
TIM_HandleTypeDef htim2;
```

Add the following code just before the while loop to enable timer interrupts:

```
HAL_TIM_Base_Start_IT(&htim2); // Enable the timer
```

Add Timer interrupt service routine (ISR):

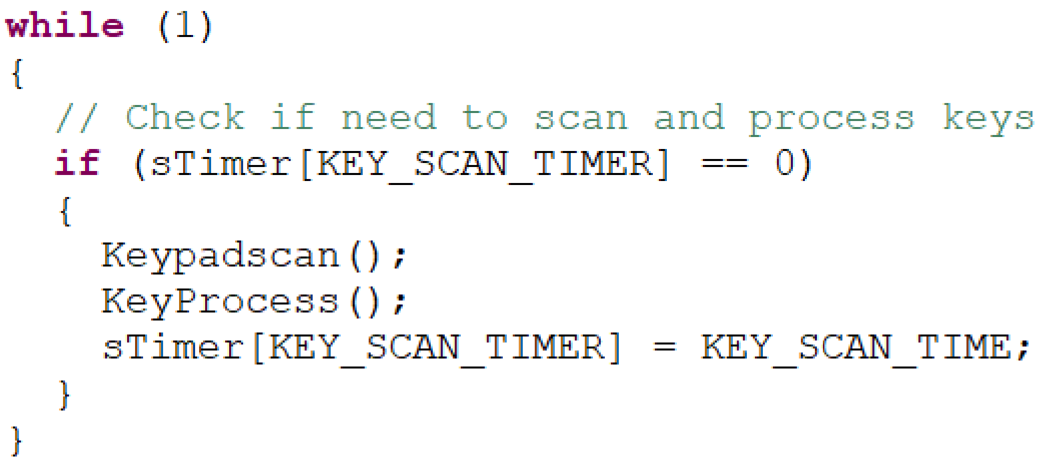
```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{

}
```

In the file stm32l4xx_it.c , the timer interrupt handler is added automatically.

```
void TIM2_IRQHandler(void)
{
 HAL_TIM_IRQHandler(&htim2);
}
```

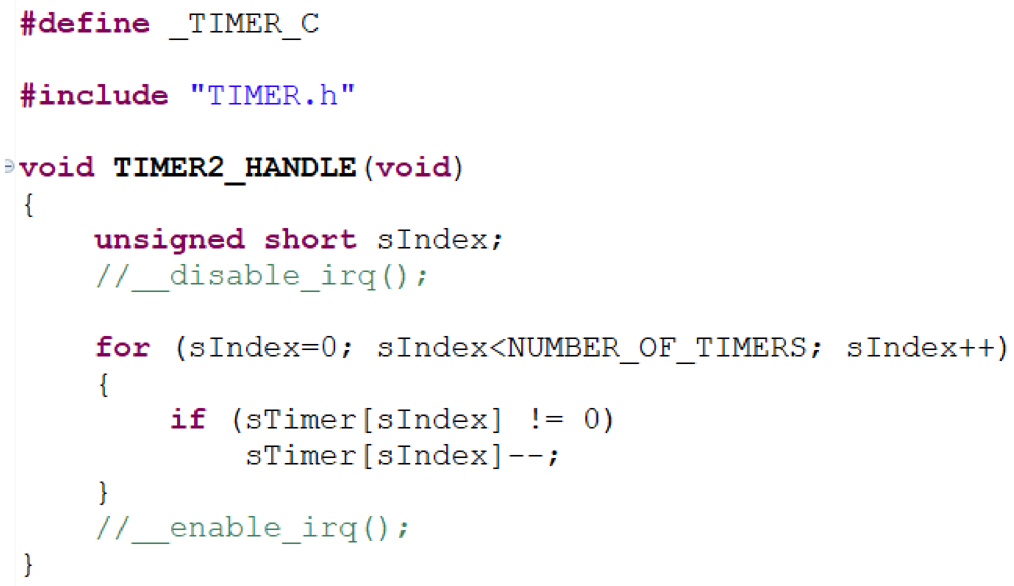
How to set up TIM2 INT (cont'd)

in main.c



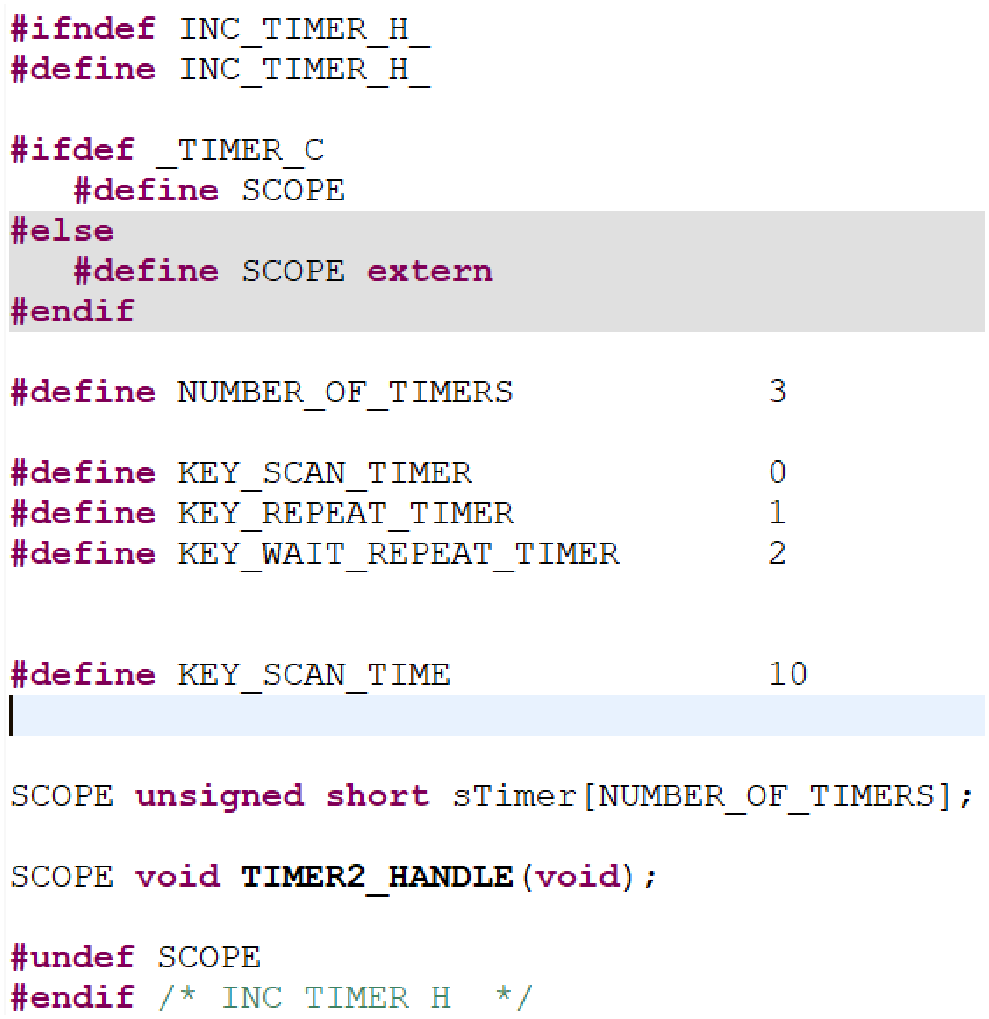
```
while (1)
{
  // Check if need to scan and process keys
  if (sTimer[KEY_SCAN_TIMER] == 0)
  {
    Keypadscan();
    KeyProcess();
    sTimer[KEY_SCAN_TIMER] = KEY_SCAN_TIME;
  }
}
```

in Timer.c



```
#define _TIMER_C

#include "TIMER.h"

void TIMER2_HANDLE(void)
{
    unsigned short sIndex;
    //__disable_irq();

    for (sIndex=0; sIndex<NUMBER_OF_TIMERS; sIndex++)
    {
        if (sTimer[sIndex] != 0)
            sTimer[sIndex]--;
    }
    //__enable_irq();
}
```

in Timer.h



```
#ifndef INC_TIMER_H_
#define INC_TIMER_H_

#ifdef _TIMER_C
   #define SCOPE
#else
   #define SCOPE extern
#endif

#define NUMBER_OF_TIMERS            3

#define KEY_SCAN_TIMER              0
#define KEY_REPEAT_TIMER            1
#define KEY_WAIT_REPEAT_TIMER       2


#define KEY_SCAN_TIME               10


SCOPE unsigned short sTimer[NUMBER_OF_TIMERS];

SCOPE void TIMER2_HANDLE(void);

#undef SCOPE
#endif /* INC TIMER H  */
```

Q & A

Please follow my demonstration and see if you can get it to work!