Lab5 7-Seg LED

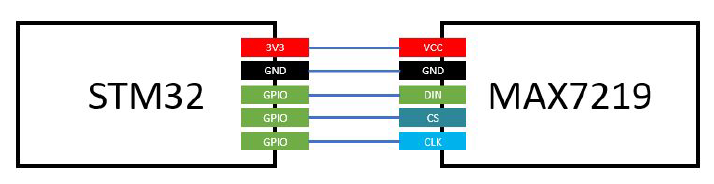
0510532 楊上萱

# Lab objectives 實驗目的

* 了解MAX7219使用原理
* 設計7-Seg LED程式

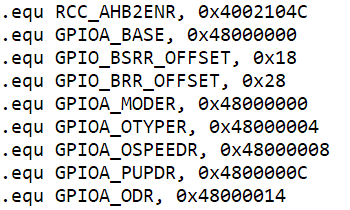
# Steps 實驗步驟

(三題的接線都一樣，如下圖。)



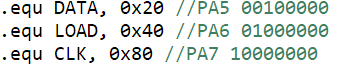
# Lab5.1: Max7219與7-Seg LED練習(Practice of Max7219 and 7-Seg)

Initialize GPIO setting：( use GPIOA)



GPIO\_init：

PA5 for DIN, PA6 for CS, PA7 for CLK



Max7219\_init：

DECODE\_MODE: 因為Decode\_mode不能顯示英文字母，所以disable。

DESPLAY\_TEST: normal operation -> 設為0

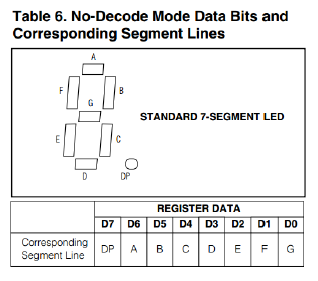
SCAN\_LIMIT: 只要看一個BIT，所以設為0

INTENSITY: 看得順眼的亮度->0xA

SHUTDOWN: normal operation -> 設為1

LED pattern：

依照下圖來自訂LED pattern。



Max7219\_send：

Use this function to send a message to max7219。

參考上課講義。

Display0toF：

概念類似用一個for loop去display，從0開始數，當數到16，再回到0繼續數。

Delay：

參考老師上課說的方式，並把X、Y改成1000。

* 1. Lab5.2: Max7219與7-Seg LED練習(Practice of Max7219 and 7-Seg LED)—use code B decode mode

Initialize GPIO setting & GPIO\_init：( use GPIOA)

跟上一題一樣都開GPIOA的port，PIN角也相同。

Max7219\_init：(改DECODE\_MODE、SCAN\_LIMIT)

DECODE\_MODE: enable decode mode，因為D0~D7都要，所以設為0xFF

DESPLAY\_TEST: normal operation -> 設為0

SCAN\_LIMIT: 學號有七個數字，所以設為6

INTENSITY: 看得順眼的亮度->0xA

SHUTDOWN: normal operation -> 設為1

Display 學號：

將學號存進名叫data區塊，並用load把它一個一個byte load進register。

2.3 Lab5.3 Max7219與7-SEG LED練習(Practice of Max7219 and 7-Seg LED)—顯示Fibonacci數(Show the Fibonacci number)

設定幾乎跟上一題一樣，加一個PC13的input pin。

高位數是零的話就不顯示：

如果N/1000\_0000==0: 不顯示，看下一位

->如果N/100\_0000==0: 不顯示，看下一位

->…直到N/10^m!=0，顯示N%10^m

Debouncing：

紀錄current\_state和last\_state，當input跟current\_state不一樣的時候，delay一段時間，然後再檢查一次，如果input跟last\_state不一樣就update (代表有按)，如果是一樣，那就等待下一次。

長按歸零：

用counter去數，當一秒鐘內都是按下的狀態，則讓數值歸零。

Fibonacci：

int fib(int N){

int fib = f0 = 0;

int f1 = 1;

if (N == 0) return 0;

else if (N == 1) return 1;

else {

for(int i = 1; i < N; i++){

fib = f1 + f0;

f0 = f1;

f1 = fib;

}

return fib;

}

}

# Results and analysis 實驗結果與分析

Lab5.1: Max7219與7-Seg LED練習(Practice of Max7219 and 7-Seg**)**

Max7219會在Bit-0的位置每隔一秒顯示0, 1, 2, …, 9, A, b, …, F, 0, 1…

Lab5.2: Max7219與7-Seg LED練習(Practice of Max7219 and 7-Seg LED)—use code B decode mode

Max7219會顯示學號。

Lab5.3 Max7219與7-SEG LED練習(Practice of Max7219 and 7-Seg LED)—顯示Fibonacci數(Show the Fibonacci number)

依據user按的次數顯示Fib(N)的值，超過8個Bit顯示-1，長按歸零。

1. **Conclusions and ideas 心得討論與應用聯想**

這次第三個實驗花非常多時間，不僅要理解max7219的原理，還要實作button的應用，雖然複雜，但是在做的過程中學到許多像是如何從個位、十位、百位…依序顯示出來、如何偵測按鈕按下、如何debounce等等。

1. **Code**

5.1

|  |
| --- |
| .syntax unified  .cpu cortex**-**m4  .thumb  .data  **//** 0**:** 01111110**,** 0x7E  **//** 1**:** 00110000**,** 0x30  **//** 2**:** 01101101**,** 0x6D  **//** 3**:** 01111001**,** 0x79  **//** 4**:** 00110011**,** 0x33  **//** 5**:** 01011011**,** 0x5B  **//** 6**:** 01011111**,** 0x5F  **//** 7**:** 01110010**,** 0x72  **//** 8**:** 01111111**,** 0x7F  **//** 9**:** 01111011**,** 0x7B  **//** A**:** 01110111**,** 0x77  **//** b**:** 00011111**,** 0x1F  **//** **C:** 01001110**,** 0x4E  **//** d**:** 00111101**,** 0x3D  **//** E**:** 01001111**,** 0x4F  **//** F**:** 01000111**,** 0x47  **//**TODO**:** put 0 to F 7**-**Seg LED pattern here  arr**:** .byte 0x7E**,** 0x30**,** 0x6D**,** 0x79**,** 0x33**,** 0x5B**,** 0x5F**,** 0x72**,** 0x7F**,** 0x7B**,** 0x77**,** 0x1F**,** 0x4E**,** 0x3D**,** 0x4F**,** 0x47  **.text**  .global main  .equ RCC\_AHB2ENR**,** 0x4002104C  .equ GPIOA\_BASE**,** 0x48000000  .equ GPIO\_BSRR\_OFFSET**,** 0x18  .equ GPIO\_BRR\_OFFSET**,** 0x28  .equ GPIOA\_MODER**,** 0x48000000  .equ GPIOA\_OTYPER**,** 0x48000004  .equ GPIOA\_OSPEEDR**,** 0x48000008  .equ GPIOA\_PUPDR**,** 0x4800000C  .equ GPIOA\_ODR**,** 0x48000014  .equ DECODE\_MODE**,** 0x09  .equ SHUTDOWN**,** 0x0C  .equ INTENSITY**,** 0x0A  .equ SCAN\_LIMIT**,** 0x0B  .equ DISPLAY\_TEST**,** 0x0F  .equ DATA**,** 0x20 **//**PA5 00100000  .equ LOAD**,** 0x40 **//**PA6 01000000  .equ CLK**,** 0x80 **//**PA7 10000000  .equ X**,** 0x3E8 **//** 1000**(DEC)**  .equ Y**,** 0x3E8 **//** 1000**(DEC)**  main**:**  **BL** GPIO\_init  **BL** max7219\_init  ldr **r9,** **=**arr  ldr r2**,** **=**#0  B Display0toF  GPIO\_init**:**  **//**TODO**:** Initialize three GPIO pins as output for max7219 DIN**,** **CS**  **//** enable AHB2 clock  **movs** r0**,** #0x1  ldr r1**,** **=**RCC\_AHB2ENR  **str** r0**,** **[**r1**]**  **//** set PA5**,** PA6**,** PA7 as output mode  **movs** r0**,** #0x5400  ldr r1**,** **=**GPIOA\_MODER  ldr r2**,** **[**r1**]**  **and** r2**,** #0xFFFF03FF  orrs r2**,**r2**,**r0  **str** r2**,[**r1**]**  **//** set PA5 as high speed mode  **movs** r0**,** #0xA800  ldr r1**,** **=**GPIOA\_OSPEEDR  **str** r0**,[**r1**]**  ldr r1**,** **=**GPIOA\_ODR  **BX** LR  Display0toF**:**  **//**TODO**:** Display 0 to F at first digit on 7**-**SEG LED. Display one per second.  **mov** r0**,** #1  ldrb r1**,** **[r9,** r2**]**  **BL** MAX7219Send  **BL** Delay  **add** r2**,** r2**,** #1  **cmp** r2**,** #16  **//**bne .for\_loop  bne Display0toF  **mov** r2**,** #0  b Display0toF  MAX7219Send**:**  **//**input parameter**:** r0 is ADDRESS **,** r1 is DATA  **//**TODO**:** Use this function to send a message to max7219  **push** {r0**,** r1**,** r2**,** r3**,** r4**,** r5**,** r6**,** r7**,** **r8,** **r9,** lr}  **lsl** r0**,** r0**,** #8  **add** r0**,** r0**,** r1  ldr r1**,** **=**#GPIOA\_BASE  ldr r2**,** **=**#LOAD  ldr r3**,** **=**#DATA  ldr r4**,** **=**#CLK  ldr r5**,** **=**#GPIO\_BSRR\_OFFSET  ldr r6**,** **=**#GPIO\_BRR\_OFFSET  **mov** r7**,** #16 **//**r7 **=** i  **//BX** LR  .max7219send\_loop**:**  **mov** **r8,** #1  **sub** **r9,** r7**,** #1  **lsl** **r8,** **r8,** **r9** **//** **r8** **=** mask  **str** r4**,** **[**r1**,**r6**]//**HAL\_GPIO\_WritePin**(**GPIOA**,** CLOCK**,** 0**)**;  tst r0**,** **r8**  beq .bit\_not\_set**//**bit **not** set  **str** r3**,** **[**r1**,**r5**]**  b .if\_done  .bit\_not\_set**:**  **str** r3**,** **[**r1**,**r6**]**  .if\_done**:**  **str** r4**,** **[**r1**,**r5**]**  subs r7**,** r7**,** #1  bgt .max7219send\_loop  **str** r2**,** **[**r1**,**r6**]**  **str** r2**,** **[**r1**,**r5**]**  **pop** {r0**,** r1**,** r2**,** r3**,** r4**,** r5**,** r6**,** r7**,** **r8,** **r9,** pc}  **BX** LR  max7219\_init**:**  **//**TODO**:** Initialize max7219 registers  **push** {r0**,** r1**,** r2**,** lr}  ldr r0**,** **=**#DECODE\_MODE  ldr r1**,** **=**#0x0  **BL** MAX7219Send  ldr r0**,** **=**#DISPLAY\_TEST  ldr r1**,** **=**#0x0  **BL** MAX7219Send  ldr r0**,** **=**#SCAN\_LIMIT  ldr r1**,** **=**0x0  **BL** MAX7219Send  ldr r0**,** **=**#INTENSITY  ldr r1**,** **=**#0xA  **BL** MAX7219Send  ldr r0**,** **=**#SHUTDOWN  ldr r1**,** **=**#0x1  **BL** MAX7219Send  **pop** {r0**,** r1**,** r2**,** pc}  **BX** LR  Delay**:**  **//**TODO**:** Write a delay 1sec function  LDR R3**,** **=**X  L1**:** LDR R4**,** **=**Y  L2**:** SUBS R4**,** #1  BNE L2  SUBS R3**,** #1  BNE L1  **BX** LR |

5.2

|  |
| --- |
| .syntax unified  .cpu cortex**-**m4  .thumb  .data  student\_id**:** .byte 0**,** 5**,** 1**,** 0**,** 5**,** 3**,** 2 **//**TODO**:** put your student id here  **.text**  .global main  .equ RCC\_AHB2ENR**,** 0x4002104C  .equ GPIOA\_BASE**,** 0x48000000  .equ GPIO\_BSRR\_OFFSET**,** 0x18  .equ GPIO\_BRR\_OFFSET**,** 0x28  .equ GPIOA\_MODER**,** 0x48000000  .equ GPIOA\_OTYPER**,** 0x48000004  .equ GPIOA\_OSPEEDR**,** 0x48000008  .equ GPIOA\_PUPDR**,** 0x4800000C  .equ GPIOA\_ODR**,** 0x48000014  .equ DECODE\_MODE**,** 0x09  .equ SHUTDOWN**,** 0x0C  .equ INTENSITY**,** 0x0A  .equ SCAN\_LIMIT**,** 0x0B  .equ DISPLAY\_TEST**,** 0x0F  .equ DATA**,** 0x20 **//**PA5 00100000  .equ LOAD**,** 0x40 **//**PA6 01000000  .equ CLK**,** 0x80 **//**PA7 10000000  main**:**  **BL** GPIO\_init  **BL** max7219\_init  **//** **r9:** student\_id pointer  **//** r2**:** pointer iterator  **//** r3**:** 8 **->** 7 **->** 6 ...  ldr **r9,** **=**student\_id  ldr r2**,** **=**#0  ldr r3**,** **=**#8  ldr r4**,** **=**#9  **//**TODO**:** display your student id on 7**-**Seg LED  **loop:**  ldrb r1**,** **[r9,**r2**]**  **add** r0**,** r2**,** #1  **sub** r0**,** r3**,** r0  **BL** MAX7219Send  **add** r2**,** r2**,** #1  **cmp** r2**,** r3  bne **loop**  Program\_end**:**  B Program\_end  GPIO\_init**:**  **//**TODO**:** Initialize three GPIO pins as output for max7219 DIN**,** **CS**  **//** enable AHB2 clock  **movs** r0**,** #0x1  ldr r1**,** **=**RCC\_AHB2ENR  **str** r0**,** **[**r1**]**  **//** set PA5**,** PA6**,** PA7 as output mode  **movs** r0**,** #0x5400  ldr r1**,** **=**GPIOA\_MODER  ldr r2**,** **[**r1**]**  **and** r2**,** #0xFFFF03FF  orrs r2**,**r2**,**r0  **str** r2**,[**r1**]**  **//** set PA5 as high speed mode  **movs** r0**,** #0xA800  ldr r1**,** **=**GPIOA\_OSPEEDR  **str** r0**,[**r1**]**  ldr r1**,** **=**GPIOA\_ODR  **BX** LR  MAX7219Send**:**  **//**input parameter**:** r0 is ADDRESS **,** r1 is DATA  **//**TODO**:** Use this function to send a message to max7219  **push** {r0**,** r1**,** r2**,** r3**,** r4**,** r5**,** r6**,** r7**,** **r8,** **r9,** lr}  **lsl** r0**,** r0**,** #8  **add** r0**,** r0**,** r1  ldr r1**,** **=**#GPIOA\_BASE  ldr r2**,** **=**#LOAD  ldr r3**,** **=**#DATA  ldr r4**,** **=**#CLK  ldr r5**,** **=**#GPIO\_BSRR\_OFFSET  ldr r6**,** **=**#GPIO\_BRR\_OFFSET  **mov** r7**,** #16 **//**r7 **=** i  **//BX** LR  .max7219send\_loop**:**  **mov** **r8,** #1  **sub** **r9,** r7**,** #1  **lsl** **r8,** **r8,** **r9** **//** **r8** **=** mask  **str** r4**,** **[**r1**,**r6**]//**HAL\_GPIO\_WritePin**(**GPIOA**,** CLOCK**,** 0**)**;  tst r0**,** **r8**  beq .bit\_not\_set**//**bit **not** set  **str** r3**,** **[**r1**,**r5**]**  b .if\_done  .bit\_not\_set**:**  **str** r3**,** **[**r1**,**r6**]**  .if\_done**:**  **str** r4**,** **[**r1**,**r5**]**  subs r7**,** r7**,** #1  bgt .max7219send\_loop  **str** r2**,** **[**r1**,**r6**]**  **str** r2**,** **[**r1**,**r5**]**  **pop** {r0**,** r1**,** r2**,** r3**,** r4**,** r5**,** r6**,** r7**,** **r8,** **r9,** pc}  **BX** LR  max7219\_init**:**  **//**TODO**:** Initialize max7219 registers  **push** {r0**,** r1**,** r2**,** lr}  ldr r0**,** **=**#DECODE\_MODE  ldr r1**,** **=**#0xFF **//** change to 11111111  **BL** MAX7219Send  ldr r0**,** **=**#DISPLAY\_TEST  ldr r1**,** **=**#0x0  **BL** MAX7219Send  ldr r0**,** **=**#SCAN\_LIMIT  ldr r1**,** **=**0x6 **//** change LIMIT to 6  **BL** MAX7219Send  ldr r0**,** **=**#INTENSITY  ldr r1**,** **=**#0xA  **BL** MAX7219Send  ldr r0**,** **=**#SHUTDOWN  ldr r1**,** **=**#0x1  **BL** MAX7219Send  **pop** {r0**,** r1**,** r2**,** pc}  **BX** LR |

5.3

|  |
| --- |
| .syntax unified  .cpu cortex**-**m4  .thumb  .data  answer**:** .byte 0  **.text**  **//**.global main  .equ RCC\_AHB2ENR**,** 0x4002104C  .equ GPIOA\_BASE**,** 0x48000000  .equ GPIO\_BSRR\_OFFSET**,** 0x18  .equ GPIO\_BRR\_OFFSET**,** 0x28  .equ GPIOA\_MODER**,** 0x48000000  .equ GPIOC\_MODER**,** 0x48000800  .equ GPIOC\_IDR**,** 0x48000810  .equ GPIOA\_OTYPER**,** 0x48000004  .equ GPIOA\_OSPEEDR**,** 0x48000008  .equ GPIOA\_PUPDR**,** 0x4800000C  .equ GPIOA\_ODR**,** 0x48000014  .equ DECODE\_MODE**,** 0x09  .equ SHUTDOWN**,** 0x0C  .equ INTENSITY**,** 0x0A  .equ SCAN\_LIMIT**,** 0x0B  .equ DISPLAY\_TEST**,** 0x0F  .equ DATA**,** 0x20 **//**PA5  .equ LOAD**,** 0x40 **//**PA6  .equ CLOCK**,** 0x80 **//**PA7  .equ X**,** 0x3E8 **//** 1000**(DEC)**  .equ Y**,** 0x3E8 **//** 1000**(DEC)**  **//** r0**:** N  **//** r4**:** fib**(**N**)**  **//** **r9:** display on max7219  main**:**  **BL** GPIO\_init  **BL** max7219\_init  **//**TODO**:**  **movs** **r11,** #0  K**:**  **movs** r0**,** **r11**  **bl** fib  **movs** r5**,**#0  **cmp** r4**,** **-**1 **//** if**(**fib**(**N**)==-**1**)**  beq minus1  ldr **r9,** **=** answer  ldr **r10,** **=**#10000000  udiv r1**,** r4**,** **r10** **//** r1 **=** fib**(**N**)** **/** 1000\_0000  **cmp** r5**,** #0  bne C1  **cmp** r1**,** #0  beq **C**  bne C1  **C:** **movs** r1**,** #0xf  strb r1**,** **[r9,**0**]**  **movs** r1**,** #0  b CC  C1**:** **movs** r5**,** #1  strb r1**,** **[r9,**0**]**  CC**:** **mul** r1**,** r1**,** **r10**  subs r4**,** r4**,** r1  ldr **r10,** **=**#1000000  udiv r1**,** r4**,** **r10**  **cmp** r5**,** #0  bne D1  **cmp** r1**,** #0  beq D  bne D1  D**:** **movs** r1**,** #0xf  strb r1**,** **[r9,**1**]**  **movs** r1**,** #0  b DD  D1**:** **movs** r5**,** #1  strb r1**,** **[r9,**1**]**  DD**:** **mul** r1**,** r1**,** **r10**  subs r4**,** r4**,** r1  ldr **r10,** **=**#100000  udiv r1**,** r4**,** **r10**  **cmp** r5**,** #0  bne E1  **cmp** r1**,** #0  beq E  bne E1  E**:** **movs** r1**,** #0xf  strb r1**,** **[r9,**2**]**  **movs** r1**,** #0  b EE  E1**:** **movs** r5**,** #1  strb r1**,** **[r9,**2**]**  EE**:** **mul** r1**,** r1**,** **r10**  subs r4**,** r4**,** r1  ldr **r10,** **=**#10000  udiv r1**,** r4**,** **r10**  **cmp** r5**,** #0  bne F1  **cmp** r1**,** #0  beq F  bne F1  F**:** **movs** r1**,** #0xf  strb r1**,** **[r9,**3**]**  **movs** r1**,** #0  b FF  F1**:** **movs** r5**,** #1  strb r1**,** **[r9,**3**]**  FF**:** **mul** r1**,** r1**,** **r10**  subs r4**,** r4**,** r1  ldr **r10,** **=**#1000  udiv r1**,** r4**,** **r10**  **cmp** r5**,** #0  bne G1  **cmp** r1**,** #0  beq G  bne G1  G**:** **movs** r1**,** #0xf  strb r1**,** **[r9,**4**]**  **movs** r1**,** #0  b GG  G1**:** **movs** r5**,** #1  strb r1**,** **[r9,**4**]**  GG**:** **mul** r1**,** r1**,** **r10**  subs r4**,** r4**,** r1  ldr **r10,** **=**#100  udiv r1**,** r4**,** **r10**  **cmp** r5**,** #0  bne H1  **cmp** r1**,** #0  beq H  bne H1  H**:** **movs** r1**,** #0xf  strb r1**,** **[r9,**5**]**  **movs** r1**,** #0  b HH  H1**:** **movs** r5**,** #1  strb r1**,** **[r9,**5**]**  HH**:** **mul** r1**,** r1**,** **r10**  subs r4**,** r4**,** r1  ldr **r10,** **=**#10  udiv r1**,** r4**,** **r10**  **cmp** r5**,** #0  bne I1  **cmp** r1**,** #0  beq I  bne I1  I**:** **movs** r1**,** #0xf  strb r1**,** **[r9,**6**]**  **movs** r1**,** #0  b II  I1**:** **movs** r5**,** #1  strb r1**,** **[r9,**6**]**  II**:** **mul** r1**,** r1**,** **r10**  subs r4**,** r4**,** r1  strb r4**,** **[r9,**7**]**  back**:**  ldr r2**,** **=**#0  ldr r4**,** **=**#9  .for\_loop**:**  ldrb r1**,** **[r9,**r2**]**  **add** r0**,** r2**,** #1  **sub** r0**,** r4**,** r0  **BL** MAX7219Send  **add** r2**,** r2**,** #1  **cmp** r2**,** r4  bne .for\_loop  **movs** r4**,** #0  b Debounce  Program\_end**:**  B Program\_end  Debounce**:**  **//**buttonState r0  **//**lastButtonState r1  ldr r2**,** **=**GPIOC\_IDR  ldr r3**,** **[**r2**]**  lsr r3**,** #13  ldr r6 **,=**#0x10000  **cmp** r4**,** r6  bgt reset  **cmp** r3**,** #0  beq **add**  bne noadd  **add:**  adds r4**,**#1  noadd**:**  **cmp** r3**,** r1  bne **wait**  beq Debounce  **wait:** **Bl** Delay  ldr r2**,** **=**GPIOC\_IDR  ldr r3**,** **[**r2**]**  lsr r3**,** #13  **cmp** r3**,**r0  bne change  beq Debounce  change**:movs** r0**,** r3  **movs** r1**,** r3  **cmp** r0**,** #0  beq do\_pushed  bne Debounce  do\_pushed**:**  adds **r11,**#1  B K  reset**:**  **movs** **r11,**#0  B K  minus1**:**  ldr r0**,** **=**#0x0f0f0f0f  **str** r0**,** **[r9]**  ldr r0**,** **=**#0x010a0f0f  **str** r0**,** **[r9,**4**]**  b back  GPIO\_init**:**  **//**TODO**:** Initialize three GPIO pins as output for max7219 DIN**,** **CS** **and** CLK  **movs** r0**,** #0x5  ldr r1**,** **=**RCC\_AHB2ENR  **str** r0**,[**r1**]**  **movs** r0**,** #0x5400  ldr r1**,** **=**GPIOA\_MODER  ldr r2**,** **[**r1**]**  **and** r2**,** #0xFFFF03FF  orrs r2**,**r2**,**r0  **str** r2**,[**r1**]**  **movs** r0**,** #0xA800  ldr r1**,** **=**GPIOA\_OSPEEDR  strh r0**,[**r1**]**  ldr r1**,** **=**GPIOC\_MODER  ldr r0**,** **[**r1**]**  ldr r2**,** **=**#0xF3FFFFFF  **and** r0**,** r2  **str** r0**,** **[**r1**]**  **BX** LR  MAX7219Send**:**  **//**input parameter**:** r0 is ADDRESS **,** r1 is DATA  **//**TODO**:** Use this function to send a message to max7219  **push** {r0**,** r1**,** r2**,** r3**,** r4**,** r5**,** r6**,** r7**,** **r8,** **r9,** lr}  **lsl** r0**,** r0**,** #8  **add** r0**,** r0**,** r1  ldr r1**,** **=**#GPIOA\_BASE  ldr r2**,** **=**#LOAD  ldr r3**,** **=**#DATA  ldr r4**,** **=**#CLOCK  ldr r5**,** **=**#GPIO\_BSRR\_OFFSET  ldr r6**,** **=**#GPIO\_BRR\_OFFSET  **mov** r7**,** #16**//**r7 **=** i  .max7219send\_loop**:**  **mov** **r8,** #1  **sub** **r9,** r7**,** #1  **lsl** **r8,** **r8,** **r9** **//** **r8** **=** mask  **str** r4**,** **[**r1**,**r6**]** **//**HAL\_GPIO\_WritePin**(**GPIOA**,** CLOCK**,** 0**)**;  tst r0**,** **r8**  beq .bit\_not\_set **//**bit **not** set  **str** r3**,** **[**r1**,**r5**]**  b .if\_done  .bit\_not\_set**:**  **str** r3**,** **[**r1**,**r6**]**  .if\_done**:**  **str** r4**,** **[**r1**,**r5**]**  subs r7**,** r7**,** #1  bgt .max7219send\_loop  **str** r2**,** **[**r1**,**r6**]**  **str** r2**,** **[**r1**,**r5**]**  **pop** {r0**,** r1**,** r2**,** r3**,** r4**,** r5**,** r6**,** r7**,** **r8,** **r9,** pc}  **BX** LR  max7219\_init**:**  **//**TODO**:** Initial max7219 registers.  **push** {r0**,** r1**,** r2**,** lr}  ldr r0**,** **=**#DECODE\_MODE  ldr r1**,** **=**#0xFF  **BL** MAX7219Send  ldr r0**,** **=**#DISPLAY\_TEST  ldr r1**,** **=**#0x0  **BL** MAX7219Send  ldr r0**,** **=**#SCAN\_LIMIT  ldr r1**,** **=**0x7  **BL** MAX7219Send  ldr r0**,** **=**#INTENSITY  ldr r1**,** **=**0xA  **BL** MAX7219Send  ldr r0**,** **=**#SHUTDOWN  ldr r1**,** **=**#0x1  **BL** MAX7219Send  **pop** {r0**,** r1**,** r2**,** pc}  **BX** LR  fib**:**  **//**TODO  **//** r2**:** f0  **//** r3**:** f1  **movs** r2**,** #0  **movs** r3**,** #1  **cmp** r0**,** #0 **//** N**==**0  beq L1  **cmp** r0**,** #39 **//** N**==**39**(**end**)**  bgt L2  **cmp** r0**,** #1 **//** N**==**1  beq L3  b L4  L1**:** **movs** r4**,** #0 **//** return 0  **bx** lr  L2**:** **movs** r4**,** #0 **//** return **-**1  subs r4**,** #1  **bx** lr  L3**:** **movs** r4**,** #1 **//**return 1  **bx** lr **//**return to caller  L4**:** **movs** r5**,** #1  **Loop:**  **//** for**(**r5**=**1; r5<N; r5++)  **//** r4**:** fib  **cmp** r5**,** r0  bge exit  adds r4**,** r2**,** r3 **//** fib **=** f0**+**f1  **movs** r2**,**r3 **//** f0**=**f1  **movs** r3**,**r4 **//** f1**=**f2  adds r5**,** #1  b **Loop**  exit**:**  **bx** lr **//**return to the caller  Delay**:**  **//**TODO**:** Write a delay 1sec function  LDR R3**,** **=**X  LL1**:** LDR R4**,** **=**Y  LL2**:** SUBS R4**,** #1  BNE L2  SUBS R3**,** #1  BNE L1  **BX** LR |