A hand-held restaurant credit card scanner includes a scanner and an LCD display, shown below.

When a credit card is scanned, a sequence of 8-bit codes appears on the scanner’s data output lines (SCAN_DATA8-1). Each byte is held on these lines for about 1 microsecond; when that byte is stable on these lines, a short pulse (0-1-0) is generated on the scanner’s DATA_VALID output line. The data byte should be read while that pulse is active.

To display a character, its ASCII code should be placed on the display’s data input lines (DATA_IN8-1) and then the display’s STROBE input line raised from low to high (it can be returned low at any time.) When the STROBE signal is thus activated, the display’s READY* output line goes to its inactive state (high) until the display has processed that character. Then READY* returns to its active state (low). The DATA_IN8-1 lines must not be disturbed while the display is in a “not ready” state.

1. The scanner is to be interfaced to our microcontroller using GPIOB pins, and the display using GPIOC pins. List appropriate GPIO pins to connect to each of the signal lines on these two devices.
2. Write a subroutine that initializes the port pins connected to these two devices.
3. Write a subroutine that returns, in R0, the 8-bit code of the next byte produced by the scanner. This subroutine should use a “busy/wait” structure.
4. Write a subroutine that will display a string of ASCII characters, stored in array "String" in data memory. The end of the character string is the ASCII "null" character (0x00). Again, busy/wait operation should be used.

These programs should be entered into MDK-ARM, assembled, and printed.

**PART 2**

It is desired to change the Scanner operation from busy-wait to interrupt driven, with the DATA_VALID signal connected to pin PA2 to trigger interrupts as “External Interrupt Signal 2” (EXTI2) on our microcontroller. In preparation for doing this, look up the following information from the documents on the course web site.
1. In the Microcontroller Technical Reference Manual (Chapter 12 – “Interrupts and Events”)
   a. What must be done to select pin PA2 as the source of the EXTI2 interrupt signal?
      (what register must be programmed?)
   b. What has to be done to enable EXTI2 to trigger an interrupt? (What register must be
      programmed?) Note that there are 23 different sources of “external interrupts” that
      can be individually enabled or masked.
   c. What has to be done to select the rising edge of EXTI2 to trigger the interrupt? (What
      register must be programmed?) This would allow the interrupt to be triggered when
      the scanner’s DATA_VALID signal goes from low to high.
   d. Referring to Table 61 in this chapter, which external interrupt number (position in the
      interrupt vector table) corresponds to EXTI2?
   e. In the startup code file, what is the name of the interrupt handler for the EXTI2
      interrupt (look in the interrupt vector table.)

2. Chapter 11 of the textbook, the STM32F4xx Cortex-M4 Programming Manual (Chapter 4.3),
   and the ARM Cortex-M4 User Guide (Chapter 4.2) discuss the Nested Vectored Interrupt
   Controller (NVIC). (These two documents contain basically the same information.)
   a. What NVIC register must be programmed to enable the EXTI2 signal to interrupt the
      CPU, what is the address of that register, and what bit of that register must be set?
   b. What NVIC register is used to set the priority of the EXTI2 interrupt, what is the
      address of that register, and what bits of that register must be programmed? How
      many priority levels does our microcontroller support?
   c. Write a short instruction sequence that would program the NVIC to enable the EXTI2
      interrupt and set its priority to 5.