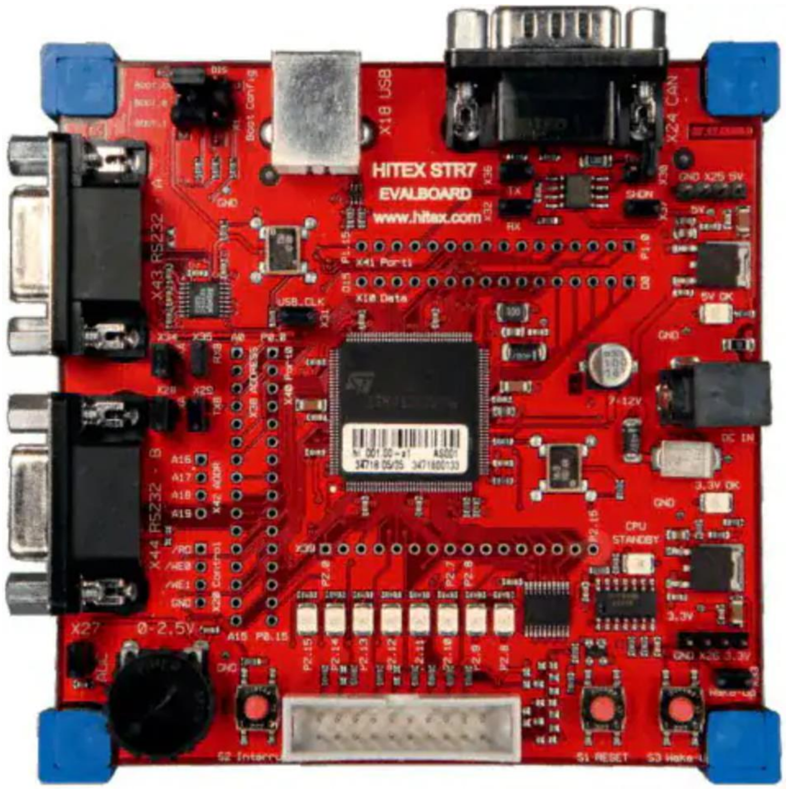
LAB HW6: “By your command”

We will give out HW7 prior to the July 22nd lab – please don’t get behind in the course.

This lab will guide you through running a simple assembly program on a simulated HITEX STR730 EVALBOARD with a ST730FZ2T7 microController. The software being used with the STR730 is Keil uVision.

STR730F family of microControllers combines the high performance ARM7TDMI™ 32-bit RISC CPU with an extensive range of peripheral functions and enhanced I/O capabilities. All devices have on-chip high-speed single voltage Flash memory and high-speed RAM. The STR730F family has an embedded ARM core and is therefore compatible with ARM tools and software, which makes it a powerful platform.

The STR730 Eval Board has several peripherals similar to those that can be seen in the image below, including LEDs, UARTs and a 10-bit ADC (shown below is a similar board, the STR7). The code provided in this lab will acquaint you with the simulated board. Due to the board being simulated, there are no actual LEDs to observe for the labs but, instead, one of the GPIO interfaces will be set to allow outputs and its data pins will substitute for LEDs. I have slowed the Crystal Oscillator (Xtal) for the simulated board down to 1.0 MHz from 8.0 MHz.



There is plenty of documentation on this technology in the ST Reference Files area on Canvas.

1. Running and Debugging your Program on the STR730 Eval Board

The file asm_include.h in the project is useful because it contains the address map of peripherals as seen from the CPU core. The peripherals we are using are memory-mapped into the CPU address space and as such we can use normal LOAD and STORE operations to access them. For reference, the base addresses of some GPIO (General Purpose Input/Output) blocks are:

```
#define APB            0xFFFF8000

#define APB_BASE       APB

#define GPIO0_BASE     (APB + 0x5400)

#define GPIO1_BASE     (APB + 0x5410)

#define GPIO2_BASE     (APB + 0x5420)

#define GPIO3_BASE     (APB + 0x5430)

#define GPIO4_BASE     (APB + 0x5440)
```

If you comment out the call to the subroutine that initializes GPIO ports (i.e. the subroutine *InitHwAssembly)*, the program doesn't do much when we run it. It turns on Bit 0 in the PD (Port Data) register for GPIO0, but none of the GPIO0 pins actually change, because we have not initialized GPIO0 to allow any output. The UART that we started using in the last assignment for *printf()* output is also not working, as the UART output steals a single pin from GPIO6 (Alternate Function, AF, output).

If you allow the *InitHwAssembly()* function to be called and run (by "uncommenting" the line and rebuilding and rerunning), then you will see a message appear on the UART output pane (UART Transmit signal now in the mode with the AF/Alternate-Function configuration and PP/Push-Pull for output – *GPIO_Mode_AF_PP*) , and you will see Pin 0 of GPIO0 turn on. We have initialized Bit 0 of GPIO0 to the mode with the OUT/Output configuration and PP/Push-Pull for output as seen in Table 16 on page 68 of manual RM0001 (cd00164537-str73x-[…].pdf in the ST reference files area on Canvas). That mode is sometimes known as *GPIO_Mode_OUT_PP*. So that means for Port GPIO0 that Bit 0 of each of Port Configuration Registers PC0 and PC2 is set to 1 and Bit 0 of PC1 is set to 0. Or stated another way, PC0(0) and PC2(0) are 1 and PC1(0) is 0.

Please modify the code in file *assembly.s* so that we achieve the following results. We want to make the pins of I/O port 0 to strobe back and forth between all the Bits in the range Bit 0 to Bit 15. We are assuming that we have 16 LEDs in a row connected to these I/O signals, similar to the STR7 picture above. The STR7 only has 8 LEDS in a row, but we are assuming that we have 16 LEDs in a row, one of each PIN of GPIO0.

We want the strobing to be slow, about the speed of a Cylon from my childhood:

Animated version at:

https://i.giphy.com/media/mM2XQp96a0zZe/giphy.webp

or another animated image at

https://media.tenor.com/images/c074023be0788f908f215e43e16dd740/tenor.gif

To achieve this, for this lab we are asking you to include some instructions to slow the strobing down. Any instructions are fine – it is up to you. Try to use as few instructions as possible to get a desirable strobing speed.

Let us know if you have any questions. “By your command.”