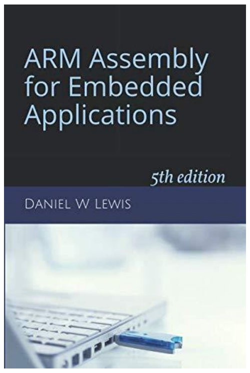


Click to download Lab7H-Main.c

Programming Lab 7H

Contolling LEDs

Topics: Bit manipulation, shift instructions, bitfields, bit-banding.

Prerequisite Reading: Chapters 1-7
Revised: May 2, 2022

Background*[1]*: There are two user-programmable LEDs on our board –red and green. They are connected to two of the 16 pins of an I/O device that is controlled by a set of 32-bit I/O ports memory-mapped to fixed locations in the address space:

GPIOC_MODER (Address 40021800_{16}; read/write): Used to configure each pin to one of four modes.

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
MODER15	MODER14	MODER13	MODER12	MODER11	MODER10	MODER9	MODER8	MODER7	MODER6	MODER5	MODER4	MODER3	MODER2	MODER1	MODER0

GPIOC_ODR (Address 40021814_{16}; read/write): Used to turn an LED on or off by writing 1 or 0 to the corresponding bit.

31 … 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0

GPIOC_BSRR (Address 40021818_{16}; write-only): Writing 1 to a bit in `BS0-BS15` turns the corresponding pin on; writing a 1 to a bit in `BR0-BR15` turns it off. `BS0-BS15` take precedence if 1's are simultaneously written to both.

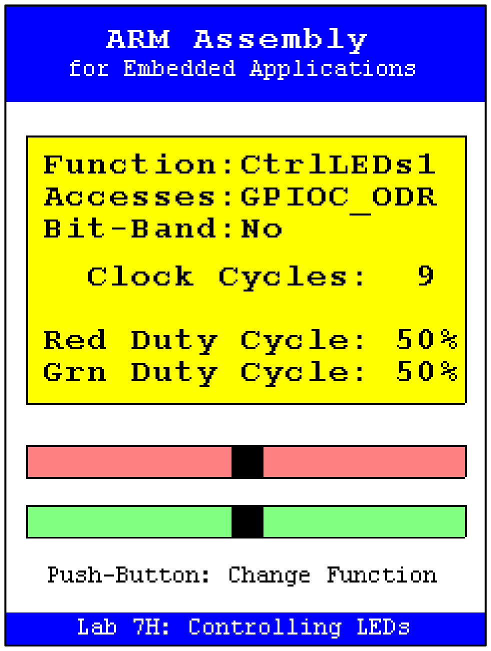
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0

Assignment: The main program will compile and run without writing any assembly. However, your task is to create equivalent replacements in assembly language for the following four functions found in the C main program. The original C versions have been defined as "weak" so that the linker will automatically replace them in the executable image by those you create in assembly; you do not need to remove the C versions. This allows you to create and test your assembly language functions one at a time.

```
void InitLEDs(void) ;
void CtrlLEDs1(BOOL red, BOOL grn) ;
void CtrlLEDs2(BOOL red, BOOL grn) ;
void CtrlLEDs3(BOOL red, BOOL grn) ;
```

`InitLEDs` configures the LED pins as outputs by setting `MODER13` and `MODER14` each to 01_2. The other three functions are used to turn the LEDs on and off. `CtrlLEDs1` does this by accessing `GPIOC_ODR` directly at address 40021814_{16}; `CtrlLEDs2` does this by accessing `GPIOC_ODR` using a hand-calculated bit-banding address. `CtrlLEDs3` turns the LEDs on and off using `GPIOC_BSRR`.

Test your code using the C main program. It turns the LEDs on and off at a very fast rate, controlling the apparent brightness by varying the duty cycle (the percentage of time the LED is on). Use the blue push button to cycle through the three functions that control the LEDs. Holding the push button down will modulate the brightness using a sine function.



```
ARM Assembly
for Embedded Applications

Function:CtrlLEDs1
Accesses:GPIOC_ODR
Bit-Band:No
   Clock Cycles:   9

Red Duty Cycle:  50%
Grn Duty Cycle:  50%

Push-Button: Change Function
Lab 7H: Controlling LEDs
```

[1] https://en.wikipedia.org/wiki/Memory-mapped_I/O