EMBEDDED SYSTEM DESIGN

LAB ASSIGNMENT -1

SUBMITTED BY- SANCHIT AGRAWAL (2016H140106)
ABHINAV GARG (2016H140103)

OBJECTIVE

To design a LED pattern repeating infinitely using STM8S105C6 microcontroller, 4 LEDs and a tactile switch such that the first LED glows when the tactile switch is pressed for the first time, the second one glows on the second press of the switch and so on till the last LED glows.

PROJECT WORKSPACE

- Used IAR Embedded Workbench with embedded C coding and used ST Visual Programmer for programming the STM8S micro controller.

- In this project we have to create a LED pattern that run infinite times, so that an LED glows every time we press the switch and after 4 LEDs are ON, on the fifth press of the switch all LEDs should be off and this pattern must be repeated infinitely.

CRITICAL ISSUES

Due to the transients in the tactile switch many interrupts were generated one press of switch. To settle down the Transients we have to close the all the interrupts in ISR and provide a proper delay.

METHODOLOGY

Connected a tactile push down switch on port D (pin4) of microcontroller. 4 LED'S are connected to port D (PD.0 to PD.3) as outputs. The LED blinking code of required pattern was written in IAR Embedded Workbench compiler and then burn into STM8S105C6 Microcontroller using STVP.

HARDWARE DESIGN

Used 4 LED's and connected them to port B (PD.0 to PD.3). Output resistors of value 10k were also used to limit the current.

The LED's are driven by the STM8S105C6 microcontroller and the input is received by tactile switch as shown in figure above.

SCHEMATIC

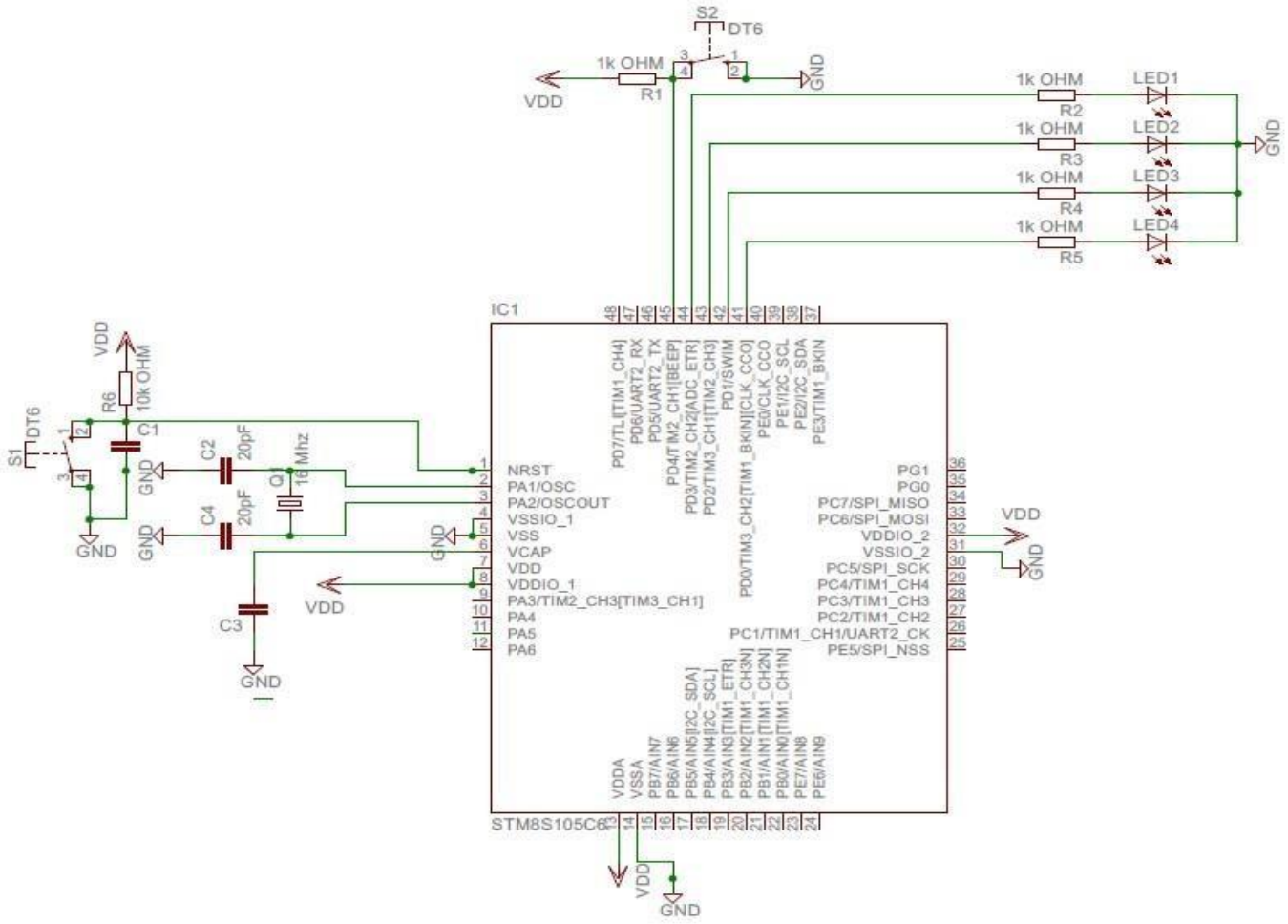


Figure 1: Schematic Diagram for LED Blink

SOFTWARE DESIGN

Used General Purpose Input Output (GPIO) function to control the input (tactile switch) and output i.e. LED's by making use of Data Direction Register (DDR), Input Data Register (IDR), Output Data Register (ODR), and Control Registers 1 (CR1 & CR2)

EXPLANATION

1. The Header file (iostm8S105C6.h) includes all the input-output functions.
2. Header file "intrinsics.h" includes intrinsic function definition such as __disable_interrupt(), __enable_interrupt (here "__" represents an intrinsic function).
3. **The interrupt that we have used on PD4 pin is an external interrupt of repeat type.**
4. Global integer "int" is defined and initialized with count=0. "int" will be used in both main function and interrupt handler function.
5. The "**pragma**" directives control the behaviour of the compiler, for example how it allocates memory for variables and functions, whether it allows extended keyword, and whether it outputs warning messeges. The pragma directives are always enabled in the compiler.
6. **"__interrupt"** is a **function attribute**, it adds an attribute to the function that it neither receive nor pass any argument.
7. In the main program loop, **"__disable_interrupt()"** is used. The significance of "__" is **intrinsic function** calling (from intrinsic.h file) for fast execution.
8. The various function performed on setting the DDR, IDR, CR registers at various pin positions is shown in table below-

Table 21. I/O port configuration summary

Mode	DDR bit	CR1 bit	CR2 bit	Function	Pull-up	P-buffer	Diodes	
							to V_{DD}	to V_{SS}
Input	0	0	0	Floating without interrupt	Off	Off	On	On
	0	1	0	Pull-up without interrupt	On			
	0	0	1	Floating with interrupt	Off			
	0	1	1	Pull-up with interrupt	On			
Output	1	0	0	Open drain output	Off	Off		
	1	1	0	Push-pull output		On		
	1	0	1	Open drain output, fast mode		Off		
	1	1	1	Push-pull, fast mode	Off	On		
	1	x	x	True open drain (on specific pins)	Not implemented		Not implemented (1)	

1. The diode connected to V_{DD} is not implemented in true open drain pads. A local protection between the pad and V_{OL} is implemented to protect the device against positive stress.

Figure 2: I/O Registers Mapping

9. We kept CR1 register of port D (C14 bit) as logic 0 so that PD4 can act as floating input, (as described in above table).
10. In "CR1_PDIS=2;", 2 is used to define the sensitivity of port D external interrupts towards falling edge only. (Details shown in figure below)

6.9.3 External interrupt control register 1 (EXTI_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
PDIS[1:0]		PCIS[1:0]		PBIS[1:0]		PAIS[1:0]	
rw		rw		rw		rw	

Bits 7:6 **PDIS[1:0]:** Port D external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of Port D external interrupts.

00: Falling edge and low level

01: Rising edge only

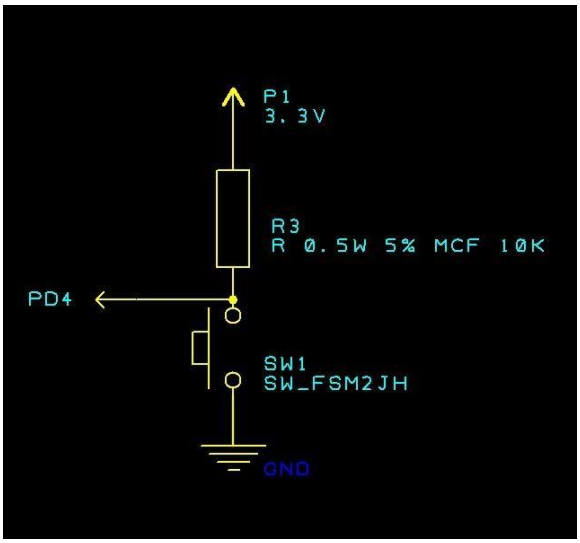
10: Falling edge only

11: Rising and falling edge

Figure 3: Interrupt Registers

11. **PD_ODR_bit structure** is used. In general structure must be defined in header file to manipulate each bit of ODR register.

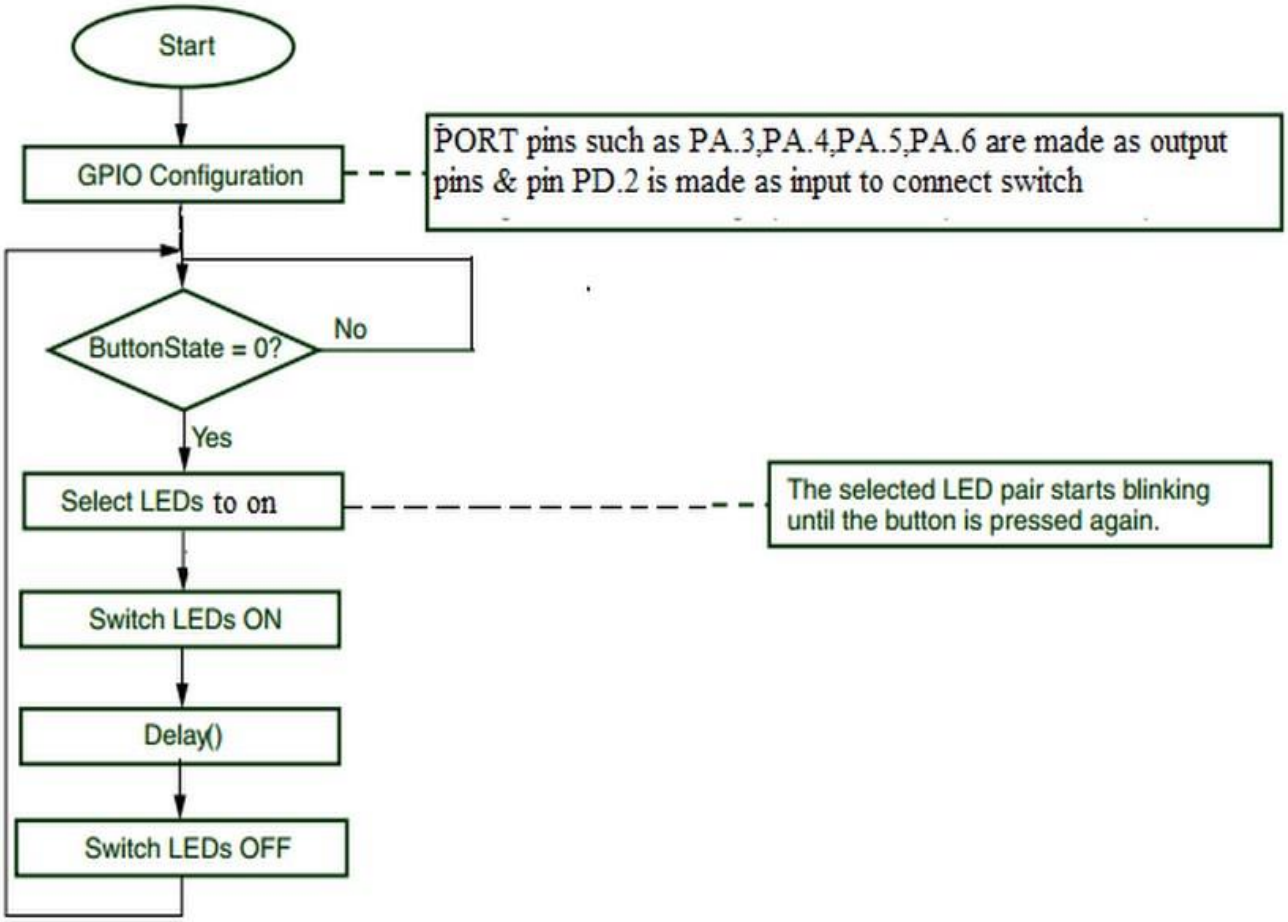
12. While(1) loop is used since we need an infinite loop to be executed.
13. Then, a tactile switch is connected to pin 4 of port D for triggering operation. (tactile switch connection)



14. After connecting the switch, count increment is performed that increases the switch count.
15. Then, we performed switch case operation to turn on the desired LED.
16. In case 0, all LED's are off.
17. In case 1, we made pin 0 of port D as logic 1(High) to turn on the LED connected to it.
18. Similarly, as above case 1,2,3,4,5 are performed to obtain the desired LED pattern.

Note- #prgma vector=8, this statement tells the compiler which interrupt vector we are going to be writing. In our case, it is vector 8 which is EXTI_PORTD interrupt vector.

FLOW CHART-



References

RM0016 – STM8S105C6 user manual

STM8S Datasheet for connections.

OBSERVATION

We are able to able to observe the LED pattern and learnt to control the switching of LED's with the use of interrupts.

RESULT-

The LED's are made to glow in the desired pattern by pressing the tactile switch.