LED TUTORIAL

STEPS TO PROGRAM A LED

1. To program LED blinking we require to set of registers RCC and GPIO defined in their respective structures.
2. Both the structures should also be defined in a sequence which is in accordance to the availability of those registers.
3. Structures should be defined , base addresses should also be defined and pointer to that structure should also be done as given in the following format.

Example is given below

Firstly define the GPIO structure (Note this structure member elements are with same name as sequence as that define in the GPIO Register (check RM0390 manual section 7.4) so that it registers can be reached.

**typedef** **struct**

{

\_\_IO uint32\_t MODER;

\_\_IO uint32\_t OTYPER;

\_\_IO uint32\_t OSPEEDR;

\_\_IO uint32\_t PUPDR;

\_\_IO uint32\_t IDR;

\_\_IO uint32\_t ODR;

\_\_IO uint32\_t BSRR;

\_\_IO uint32\_t LCKR;

\_\_IO uint32\_t AFR[2];

} GPIO\_TypeDef;

Also \_\_IO is defined as

#define \_\_IO volatile

Secondly define the Base address of GPIOA from the STM32F446re Data sheet , the address we get is

GPIOA base address is

**#define** GPIOA\_BASE (0x40020000)

Thirdly point the GPIO structure to the base address so that the registers are accessible as follows

**#define** GPIOA ((GPIO\_TypeDef \*) GPIOA\_BASE)

Fourthly define the RCC structure (Note this structure member elements are with same name as sequence as that define in the RCC Register (check RM0390 manual section 6.3) so that it registers can be reached.

Which can be done as according to the following example

typedef struct

{

\_\_IO uint32\_t CR;

\_\_IO uint32\_t PLLCFGR;

\_\_IO uint32\_t CFGR;

\_\_IO uint32\_t CIR;

\_\_IO uint32\_t AHB1RSTR;

\_\_IO uint32\_t AHB2RSTR;

\_\_IO uint32\_t AHB3RSTR;

uint32\_t RESERVED0;

\_\_IO uint32\_t APB1RSTR;

\_\_IO uint32\_t APB2RSTR;

uint32\_t RESERVED1[2];

\_\_IO uint32\_t AHB1ENR;

\_\_IO uint32\_t AHB2ENR;

\_\_IO uint32\_t AHB3ENR;

uint32\_t RESERVED2;

\_\_IO uint32\_t APB1ENR;

\_\_IO uint32\_t APB2ENR;

uint32\_t RESERVED3[2];

\_\_IO uint32\_t AHB1LPENR;

\_\_IO uint32\_t AHB2LPENR;

\_\_IO uint32\_t AHB3LPENR;

uint32\_t RESERVED4;

\_\_IO uint32\_t APB1LPENR;

\_\_IO uint32\_t APB2LPENR;

uint32\_t RESERVED5[2];

\_\_IO uint32\_t BDCR;

\_\_IO uint32\_t CSR;

uint32\_t RESERVED6[2];

\_\_IO uint32\_t SSCGR;

\_\_IO uint32\_t PLLI2SCFGR;

\_\_IO uint32\_t PLLSAICFGR;

\_\_IO uint32\_t DCKCFGR;

\_\_IO uint32\_t CKGATENR;

\_\_IO uint32\_t DCKCFGR2;

} RCC\_TypeDef;

**#define** RCC\_BASE (0x40023800)

#define RCC ((RCC\_TypeDef \*) RCC\_BASE)

1. Refer to Schematic pack for user LED connection which is to PA5.
2. Port A is connected to AHB1 Bus. So clock to port A must be enabled on AHB1 for PORTA thru RCC\_AHB1ENR register.(check for the bit on which it should be set to enable clock for PORTA.
3. Now coming to registers related to PORTA, firstly the PORTA should be configured as either input or output, using (GPIOx\_MODER register on relevant bit);
4. If it is configured as output then one of the two types in output( like output push pull reset or output open drain ) by selecting the bits on GPIOx\_OTYPER(we go with output push pull reset so reset those bits on bit 11,10 to 0,0).
5. Next step is to select the speed of the ports at which it operates(means to say the clock pulses to be given, available options are to make it work on **low speed, medium speed , fast speed and high speed by selecting options 00,01,10,11 respectively**).
6. Next step is to select either a **no pull up pull down , pull up, pull down, reserved)** if the port is selected as output first option needs to be taken if the port is to be read like in case of switch programming then it should be configured as pull up. Reserved option tells that it is not available for the programmer to program.
7. After configuring it as output if we want to send any data to the portA either 1 or 0, we do it by sending the data on ODR register, as GPIOA->ODR which will affect on the relevant port in the form of ON or OFF if it is connected to LED.
8. Now firstly enable the clock for PORTA on AHB1ENR as follows

RCC->AHB1ENR|=(1<<0);

Now program the LED blinking with alternately sending 1’s and 0’s on GPIOA->ODR as follows.

**for**(;;)

{

GPIOA->ODR |= (1<<5);

delay();

GPIOA->ODR &= ~(1<<5);

delay();

}

Then complete the compile(build steps ) and debug steps to see the behavior on the board LED blinking on PA5.