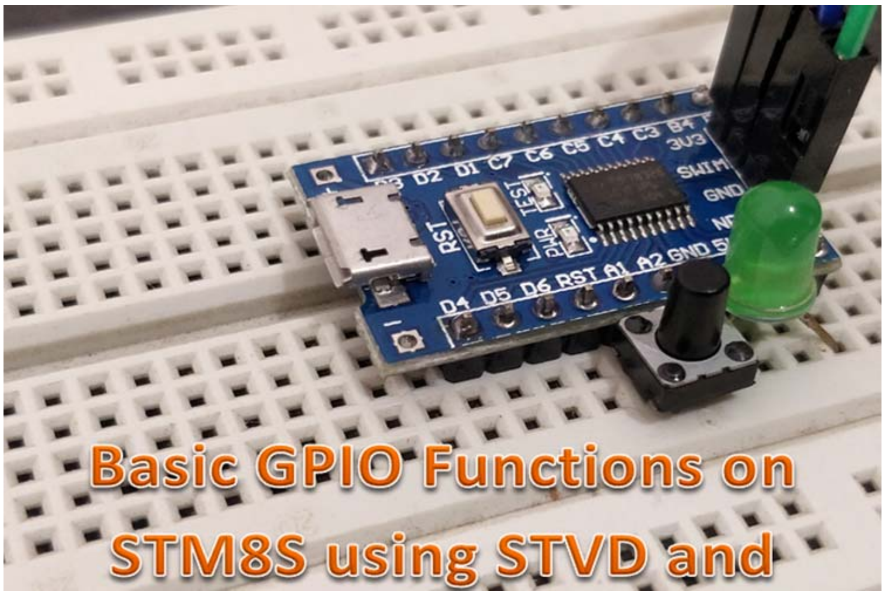
ELECTRONICS (HTTPS://CIRCUITDIGEST.COM/ELECTRONICS-PROJECTS)

GPIO Functions on STM8S using Cosmic C and SPL – Blinking and Controlling LED with Push Button (/microcontroller-projects/gpio-functions-on-stm8s-using-cosmic-c-and-spl-blinking-led-with-push-button)

By (page_author.html)Aswinth Raj (/users/aswinth-raj) Jun 19, 2020 3

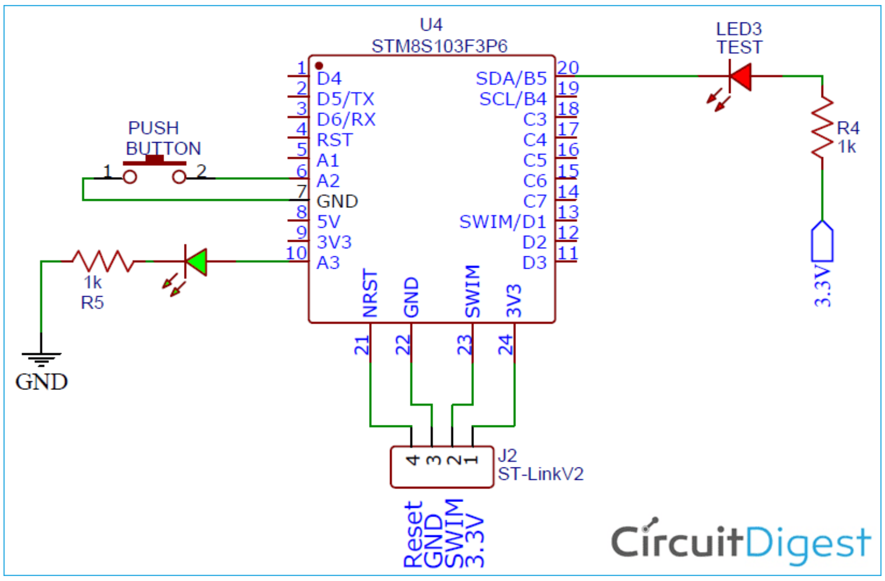


STM8S103F GPIO Functions

For microcontrollers, an LED blinking program is equivalent to the “hello world” program. In our previous tutorial, (https://circuitdigest.com/microcontroller-projects/getting-started-with-stm8s-using-stvd-and-cosmic-c-compiler) we learned how to get started with STM8S103F3 Development Board and how to set up the IDE and compiler to program our STM8S controllers. We have also learned how to use the standard peripheral libraries, and how to compile and upload the code into our microcontroller. With all the basics covered, lets actually start writing code. In this tutorial, we will learn how to perform general GPIO functions on STM8S controllers. The board already has an onboard LED connected to pin 5 of port B, we will learn how to blink this LED and also add an external LED and control it with a push-button. If you are completely new, it is highly recommended to read the previous tutorial before you proceed any further.

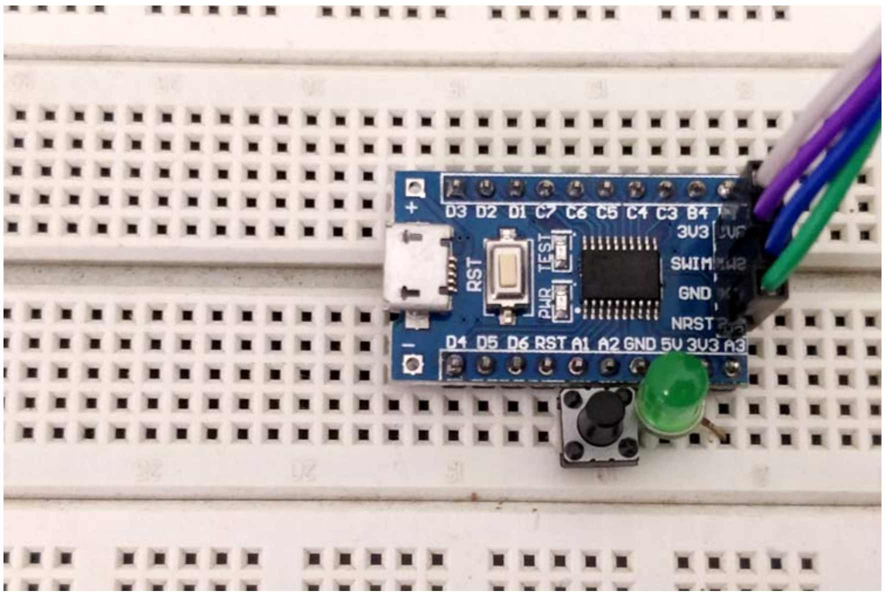
Getting the Hardware Ready

Before we dive into the program, let get the hardware connections ready. As mentioned early, we will be using two LEDs here, one is an onboard LED which will blink continuous and the other is an external LED which will be toggled with a push button. The idea is to learn all the GPIO functionality in a simple set up. The on-board Led is already connected to PB5 (pin5 of PORTB), so I have just connected an LED to PA3 and a push-button to PA2, as you can see in the diagram below.



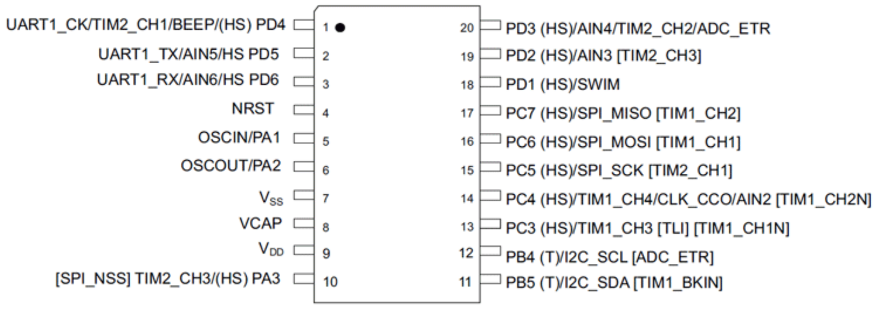
(/fullimage?i=circuitdiagram_mic/STM8S103F3-Internal-Circuit.png)

But, of all the output pins available on our controlled why did I select PA3 for output and PA2 for input? The questions are valid and I will explain that later in this article. My hardware set-up for this tutorial is shown below. As you can see, I have also connected my ST-link programmer to programming pins which will not only program our board but will also act as a power source.



Understanding GPIO Pinouts on STM8S103F

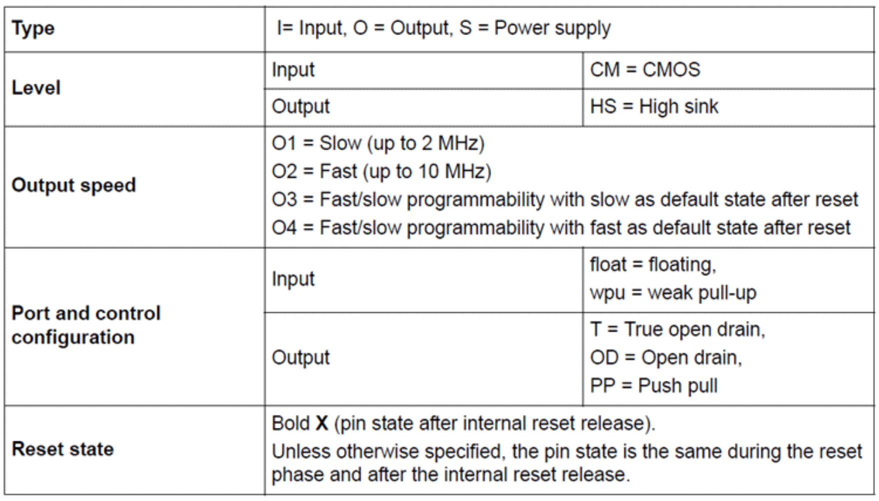
Now coming back to the question, why PA2 for input and why PA3 for output? To understand that, let's take a closer look at the pinout of the microcontroller which is shown below.


UART1_CK/TIM2_CH1/BEEP/(HS) PD4
UART1_TX/AIN5/HS PD5
UART1_RX/AIN6/HS PD6
NRST
OSCIN/PA1
OSCOUT/PA2
V_{SS}
VCAP
V_{DD}
[SPI_NSS] TIM2_CH3/(HS) PA3
1
2
3
4
5
6
7
8
9
10
20
19
18
17
16
15
14
13
12
11
PD3 (HS)/AIN4/TIM2_CH2/ADC_ETR
PD2 (HS)/AIN3 [TIM2_CH3]
PD1 (HS)/SWIM
PC7 (HS)/SPI_MISO [TIM1_CH2]
PC6 (HS)/SPI_MOSI [TIM1_CH1]
PC5 (HS)/SPI_SCK [TIM2_CH1]
PC4 (HS)/TIM1_CH4/CLK_CCO/AIN2 [TIM1_CH2N]
PC3 (HS)/TIM1_CH3 [TLI] [TIM1_CH1N]
PB4 (T)/I2C_SCL [ADC_ETR]
PB5 (T)/I2C_SDA [TIM1_BKIN]

As per the pinout diagram, we have four ports on our microcontroller, namely, PORT A, B, C, and D denoted by PA, PB, PC, and PD respectively. Each GPIO pin is also clubbed with some other special functionality. For example, the PB5 (pin 5 of PORT B) can not only work as a GPIO pin but also as an SDA pin for I2C communication and as a Timer 1 output pin. So, if we use this pin for simple GPIO purposes like connecting an LED, then we won't be able to use I2C and the LED at the same time. Sadly, the on-board LED is connected to this pin, so we don't have much of a choice here, and in this program, we are not going to use I2C, so it's not much of a problem.

Pinout Description and Tips for STM8S103F GPIO Selection

Truly speaking, it would not hurt to use PA1 an input pin and it would just work pin. But I deliberately brought this up to provide me an opportunity to show you some common traps that you might fall into when selecting GPIO pins on a new microcontroller. The best to avoid the traps is to read the pin details and pin description provided in the STM8S103F3P6 datasheet (https://www.st.com/resource/en/datasheet/stm8s103f2.pdf). For the STM8S103F3P6 microcontroller pin description details that are mentioned in the datasheet are shown below images.



Type	I= Input, O = Output, S = Power supply	
Level	Input	CM = CMOS
	Output	HS = High sink

Output speed	O1 = Slow (up to 2 MHz)
	O2 = Fast (up to 10 MHz)
	O3 = Fast/slow programmability with slow as default state after reset
	O4 = Fast/slow programmability with fast as default state after reset	

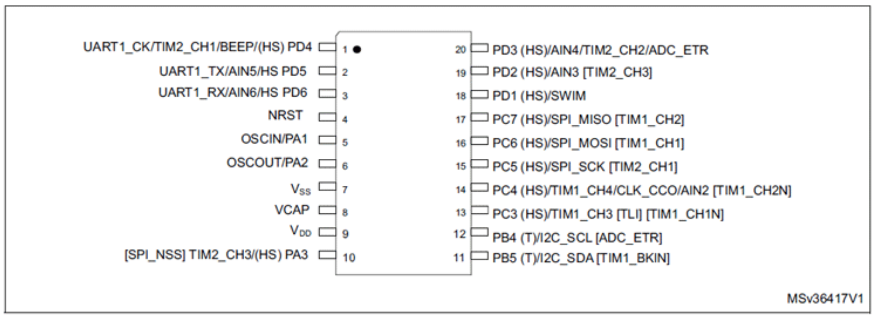
Port and control configuration	Input	float = floating,
		wpu = weak pull-up

	Output	T = True open drain,
		OD = Open drain,
		PP = Push pull

Reset state	Bold **X** (pin state after internal reset release). Unless otherwise specified, the pin state is the same during the reset phase and after the internal reset release.	

The input pins on our microcontroller can either be floating or weak pull-up and the output pins can either be Open Drain or Push-pull. The difference between Open Drain and Push-Pull Output pins (https://circuitdigest.com/forums/embedded/difference-between-open-drain-and-push-pull)is already discussed, hence we won't get into details of that. To put it simple, an Open Drain output pin can make the output only as low not as high, while a push-pull output pin can make the output both as high as well as high.

Apart from that from the above table, you can also notice that an output pin can either be Fast output (10 Mhz) or Slow Output (2 MHz). This determines the GPIO Speed, if you want to switch your GPIO pins between high and low very fast, then we can choose Fast output.


UART1_CK/TIM2_CH1/BEEP/(HS) PD4 1
UART1_TX/AIN5/HS PD5 2
UART1_RX/AIN6/HS PD6 3
NRST 4
OSCIN/PA1 5
OSCOUT/PA2 6
V_{SS} 7
VCAP 8
V_{DD} 9
[SPI_NSS] TIM2_CH3/(HS) PA3 10
20 PD3 (HS)/AIN4/TIM2_CH2/ADC_ETR
19 PD2 (HS)/AIN3 [TIM2_CH3]
18 PD1 (HS)/SWIM
17 PC7 (HS)/SPI_MISO [TIM1_CH2]
16 PC6 (HS)/SPI_MOSI [TIM1_CH1]
15 PC5 (HS)/SPI_SCK [TIM2_CH1]
14 PC4 (HS)/TIM1_CH4/CLK_CCO/AIN2 [TIM1_CH2N]
13 PC3 (HS)/TIM1_CH3 [TLI] [TIM1_CH1N]
12 PB4 (T)/I2C_SCL [ADC_ETR]
11 PB5 (T)/I2C_SDA [TIM1_BKIN]
MSv36417V1

1. HS high sink capability.
2. (T) True open drain (P-buffer and protection diode to VDD not implemented).
3. [ ] alternate function remapping option (If the same alternate function is shown twice, it indicates an exclusive choice not a duplication of the function)

Some GPIO pins on our controller support **True Open Drain (T)** and **High Sink Current (HS)** as mentioned in the above image. A considerable difference between Open Drain and True Open Drain (https://circuitdigest.com/forums/embedded/true-open-drain-and-high-sink-capability-microcontroller-gpio-pins-meaning) is that the output connected to open drain cannot be pulled high more than the operating voltage of microcontroller (Vdd) while a true open-drain output pin can be pulled higher than Vdd. Pins with High Sink Capability means that it can sink more current. The source and sink current of any GPIO HS pin is 20mA, while the power line can consume up to 100 mA.

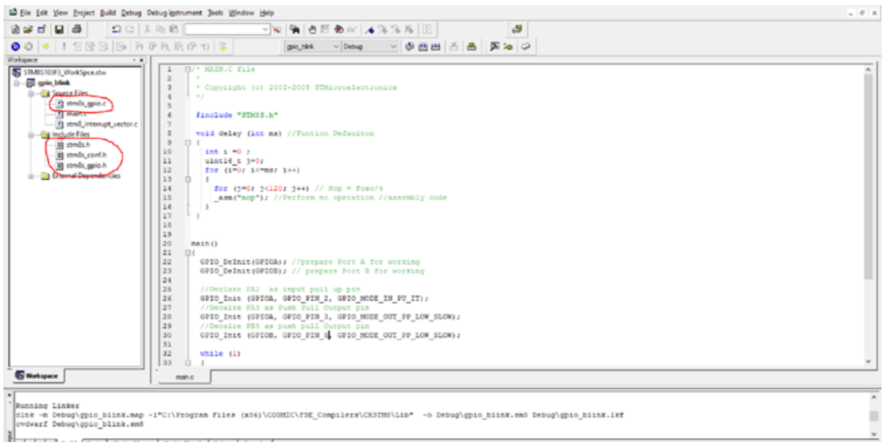
Taking a closer look on the above image, you will notice that almost all GPIO pins are High Sink Current (HS) type except for PB4 and PB5 which are True Open Drain Type(T). This means that these pins cannot be made high, they will not be able to provide 3.3V even when the pin is made high. This is why the onboard led is connected to a 3.3V and grounded through PB5 instead of powering it directly from the GPIO pin.

1. I/O pins used simultaneously for high current source/sink must be uniformly spaced around the package. In addition, the total driven current must respect the absolute maximum ratings.
2. When the MCU is in halt/active-halt mode, PA1 is automatically configured in input weak pull-up and cannot be used for waking up the device. In this mode, the output state of PA1 is not driven. It is recommended to use PA1 only in input mode if halt/active-halt is used in the application.
3. In the open-drain output column, "T" defines a true open-drain I/O (P-buffer, weak pull-up, and protection diode to VDD are not implemented).1

Refer to page 28 on the datasheet for the detailed pin description. As mentioned in the above image, PA1 is automatically configured as a weak pull-up and is not recommended to be used as an output pin. Anyways it can be used as an input pin along with a push-button, but I decided to use PA2 just to try enabling pull up from the program. These are just a few basic things which will be useful when we write much more complicated programs. For now, it's okay if many things bounced off your head, we will get into it layer in other tutorials.

Programming STM8S for GPIO Input and Output using SPL

Create a workspace and new project as we discussed in our first tutorial (https://circuitdigest.com/microcontroller-projects/getting-started-with-stm8s-using-stvd-and-cosmic-c-compiler). You can either add all the header and source files or only add the gpio, config, and stm8s files. Open the *main.c* file and start writing your program.



```
/* MAIN.C file
 *
 * Copyright (c) 2002-2005 STMicroelectronics
 */

#include "STM8S.h"

void delay (int ms) //Funtion Defeniton
{
  int i =0 ;
  uint16_t j=0;
  for (i=0; i<=ms; i++)
  {
    for (j=0; j<120; j++) // Nop = Fosc/4
    _asm("nop"); //Perform no operation //assembly code
  }
}


main()
{
  GPIO_DeInit(GPIOA); //prepare Port A for working
  GPIO_DeInit(GPIOB); // prepare Port B for working

  //Declare PA2  as input pull up pin
  GPIO_Init (GPIOA, GPIO_PIN_2, GPIO_MODE_IN_PU_IT);
  //Decalre PA3 as Push Pull Output pin
  GPIO_Init (GPIOA, GPIO_PIN_3, GPIO_MODE_OUT_PP_LOW_SLOW);
  //Decalre PB5 as push pull Output pin
  GPIO_Init (GPIOB, GPIO_PIN_5, GPIO_MODE_OUT_PP_LOW_SLOW);

  while (1)
  {
```

```
Running Linker
clnk -m Debug\gpio_blink.map -l"C:\Program Files (x86)\COSMIC\FSE_Compilers\CXSTM8\Lib"  -o Debug\gpio_blink.sm8 Debug\gpio_blink.lkf
cvdwarf Debug\gpio_blink.sm8
```

Make sure you have included the header files as shown in the image above. Open the *main.c* file and starting the code. The complete main.c code can be found at the bottom of this page and you will also be able to download the project file from there. The explanation of the code is as follows, you can also refer to the SPL User manual (https://github.com/CircuitDigest/STM8S103F3_SPL/blob/master/SPL_User_Manual.chm) or the video linked at the bottom of this page if you are confused about the coding part.

De-Initializing the Required Port

We begin our program by De-Initializing the required ports. As we discussed earlier, each GPIO pin will have many other functions associated with it other than just working like a normal Input and Output. If these pins have been previously used for some other applications, then it should be De-Initialized before we use them. It is not mandatory, however, it is a good practice. The following two lines of code are used to De-Initialize Port A and Port B. Just use syntax ***GPIO_DeInit (GPIOx);*** and mention the port name in place of x.

```
GPIO_DeInit(GPIOA); //prepare Port A for working
GPIO_DeInit(GPIOB); // prepare Port B for working
```

Input and Output GPIO Declaration

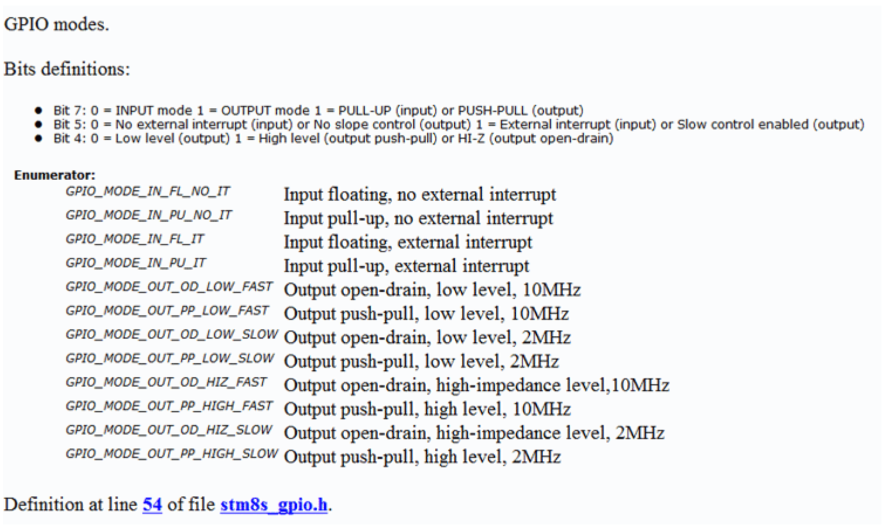
Next, we have to declare which pins will be used as input and which as output. In our case, pin PA2 will be used as input, we will also declare this pin with internal Pull-up so that we don't have to use one externally. The syntax is ***GPIO_Init (GPIOx, GPIO_PIN_y, GPIO_PIN_MODE_z);***. Where x is the port name, y is the pin number, and z is the GPIO Pin mode.

```
//Declare PA2  as input pull up pin
                GPIO_Init (GPIOA, GPIO_PIN_2, GPIO_MODE_IN_PU_IT);
```

Next, we have to declare the pins PA3 and PB5 as output. Again many types of output declaration are possible but we will be using *"GPIO_MODE_OUT_PP_LOW_SLOW"* which means that we will declare it as an output pin of push-pull type with slow speed. And by default, the value will be low. The syntax will be the same.

```
GPIO_Init (GPIOA, GPIO_PIN_3, GPIO_MODE_OUT_PP_LOW_SLOW);
                //Declare PB5 as push pull Output pin
                GPIO_Init (GPIOB, GPIO_PIN_5, GPIO_MODE_OUT_PP_LOW_SLOW);
```

The below snapshot from the SPL user manual mentions all possible GPIO modes (z).



GPIO modes.

Bits definitions:

- Bit 7: 0 = INPUT mode 1 = OUTPUT mode 1 = PULL-UP (input) or PUSH-PULL (output)
- Bit 5: 0 = No external interrupt (input) or No slope control (output) 1 = External interrupt (input) or Slow control enabled (output)
- Bit 4: 0 = Low level (output) 1 = High level (output push-pull) or HI-Z (output open-drain)

Enumerator:

GPIO_MODE_IN_FL_NO_IT	Input floating, no external interrupt
GPIO_MODE_IN_PU_NO_IT	Input pull-up, no external interrupt
GPIO_MODE_IN_FL_IT	Input floating, external interrupt
GPIO_MODE_IN_PU_IT	Input pull-up, external interrupt
GPIO_MODE_OUT_OD_LOW_FAST	Output open-drain, low level, 10MHz
GPIO_MODE_OUT_PP_LOW_FAST	Output push-pull, low level, 10MHz
GPIO_MODE_OUT_OD_LOW_SLOW	Output open-drain, low level, 2MHz
GPIO_MODE_OUT_PP_LOW_SLOW	Output push-pull, low level, 2MHz
GPIO_MODE_OUT_OD_HIZ_FAST	Output open-drain, high-impedance level,10MHz
GPIO_MODE_OUT_PP_HIGH_FAST	Output push-pull, high level, 10MHz
GPIO_MODE_OUT_OD_HIZ_SLOW	Output open-drain, high-impedance level, 2MHz
GPIO_MODE_OUT_PP_HIGH_SLOW	Output push-pull, high level, 2MHz

Definition at line 54 of file stm8s_gpio.h.

Infinite while loop

After the pin declaration, we need to create an infinite loop inside which we will keep blinking the LED forever and monitor the status of the push button to toggle the LED. The infinite loop can either create with a *while(1)* or with a for *(;;)*. Here I have used *while (1).*

```
while (1)
                {
                }
```

Checking the status of the Input pin

We have to check the status of the input pin, the syntax to do that is ***GPIO_ReadInputPin(GPIOx, GPIO_PIN_y);*** where x is the port name and y is the pin number. If the pin is high, we will get '1' and if the pin is low, we will get a '0'. We have used to inside an if loop to check if the pin is high or low.

```
if (GPIO_ReadInputPin(GPIOA, GPIO_PIN_2)) //if button pressed
```

Making a GPIO Pin High or Low

To make a GPIO pin High or Low, we can use ***GPIO_WriteHigh(GPIOx,GPIO_PIN_y);*** and ***GPIO_WriteLow(GPIOx,GPIO_PIN_y);*** respectively. Here we have made the LED to turn on if the button is pressed and turn off if the button is not pressed.

```
if (GPIO_ReadInputPin(GPIOA, GPIO_PIN_2)) //if button pressed
GPIO_WriteLow(GPIOA,GPIO_PIN_3); //LED ON
else
GPIO_WriteHigh(GPIOA,GPIO_PIN_3); //LED OFF
```

Toggling a GPIO Pin

To toggle a GPIO pin, we have ***GPIO_WriteReverse(GPIOx,GPIO_PIN_y);*** calling this function will change the status of the output pin. If the pin is high, it will be changed to low, and if it is low, it will be changed to high. We are using this function to blink the onboard LED on PB5.

```
GPIO_WriteReverse(GPIOB,GPIO_PIN_5);
```

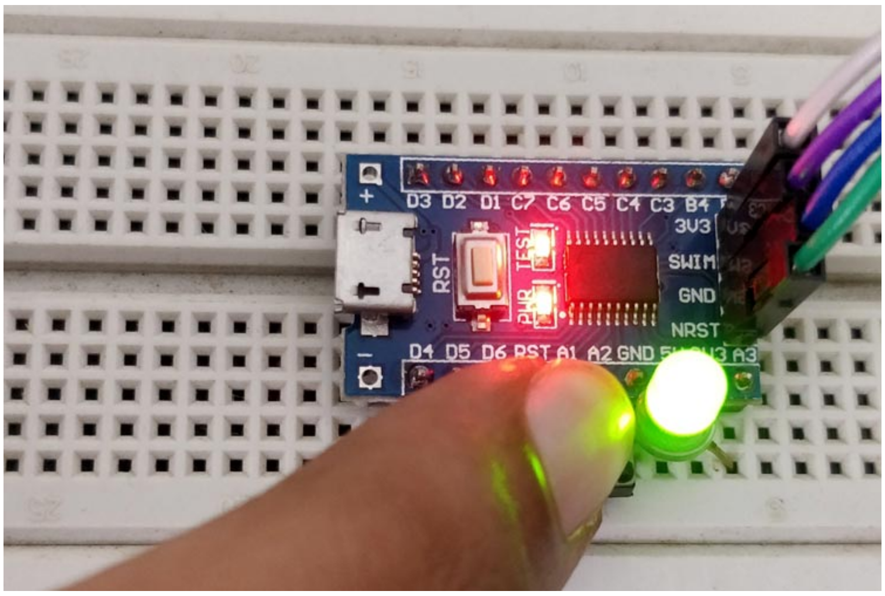
Delay Function

Unlike Arduino, the cosmic compiler does not have a pre-defined delay function. So we have to create one on our own. My delay function is given below. The value doe the delay will be received in the variable ms and we will use two for loop to hold or program execution. Like ***_asm("nop")*** is an assembly instruction which stands for no operation. This means that the controller will be looping into the for loop without performing any operation, thus creating a delay.

```
void delay (int ms) //Function Definition
 {
                 int i =0 ;
                 int j=0;
                 for (i=0; i<=ms; i++)
                 {
                                 for (j=0; j<120; j++) // Nop = Fosc/4
                                 _asm("nop"); //Perform no operation //assembly code
                 }
 }
```

Uploading and Testing the Program

Now that our program is ready, we can upload it and test it. Once uploaded, my hardware was working as expected. The on-board red LED was blinking for every 500 milliseconds and the external green LED turned on every time I pressed the switch.



The complete working can be found in the video linked below. Once you have reached this point, you can try to connect the switch and LED to different pins and re-write the code to understand the concept. You can also play with the delay timing to check if you have understood the concepts clearly.

If you have any questions, please leave them in the comment section below and for other technical questions, you can use our forums (https://circuitdigest.com/forums/embedded/true-open-drain-and-high-sink-capability-microcontroller-gpio-pins-meaning). Thanks for following by, see you in the next tutorial.

Code

```
/* MAIN.C file
 *
 * Copyright (c) 2002-2005 STMicroelectronics
 */

 #define Green_LED GPIOA, GPIO_PIN_3
 #include "STM8S.h"

void delay (int ms) //Function Definition
{
int i =0 ;
int j=0;
for (i=0; i<=ms; i++)
{
for (j=0; j<120; j++) // Nop = Fosc/4
_asm("nop"); //Perform no operation //assembly code
}
}

main()
{
GPIO_DeInit(GPIOA); //prepare Port A for working
GPIO_DeInit(GPIOB); // prepare Port B for working

//Declare PA2  as input pull up pin
GPIO_Init (GPIOA, GPIO_PIN_2, GPIO_MODE_IN_PU_IT);

//Declare PA3 as Push Pull Output pin
GPIO_Init (Green_LED, GPIO_MODE_OUT_PP_LOW_SLOW);

//Declare PB5 as push pull Output pin
GPIO_Init (GPIOB, GPIO_PIN_5, GPIO_MODE_OUT_PP_LOW_SLOW);

while (1)
{
if (GPIO_ReadInputPin(GPIOA, GPIO_PIN_2)) //if button pressed
GPIO_WriteLow(Green_LED); //LED ON
else
GPIO_WriteHigh(Green_LED); //LED OFF
GPIO_WriteReverse(GPIOB,GPIO_PIN_5);
delay (100);
}
}
```

Video

Basic GPIO Functions on STM8S Microcontroller - LED Blink and Toggle wit...

TAGS STM8 (/TAGS/STM8) STMICROELECTRONICS (/TAGS/STMICROELECTRONICS) GPIO PIN (/TAGS/GPIO-PIN) PROGRAMMING (/TAGS/PROGRAMMING) LED BLINKING (/TAGS/LED-BLINKING)

RECOMMENDED POSTS

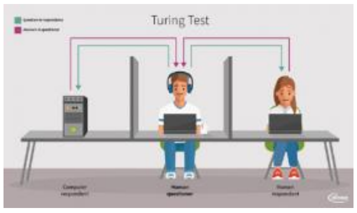
A.I. DAY
Assemble all the pieces needed for AI at the Edge of the IoT
September 9

(https://bit.ly/3serpko)
AI Day – September 9th 2021 (https://bit.ly/3serpko)

(https://bit.ly/2XmYEqp)
Register to Win an 11th Generation Intel® Core i7-based NUC Topaz (https://bit.ly/2XmYEqp)

(https://bit.ly/3g2qYFa)
Hands-On Experience: Build Your First AI Solution in an Hour with the OpenVINO Toolkit (https://bit.ly/3g2qYFa)

(https://bit.ly/2VTK7SM)
Embedded Computing Design's Annual AI Survey (https://bit.ly/2VTK7SM)



(https://bit.ly/2VWV71N)
Know Thy AI Processor & Programming Options (https://bit.ly/2VWV71N)

(https://bit.ly/3g2zGDa)
How to Accelerate AI at the IoT Edge with Siemens (https://bit.ly/3g2zGDa)

(https://bit.ly/3xQmtmV)
Embedded Toolbox: Let's Build a Robot with VxWorks (https://bit.ly/3xQmtmV)

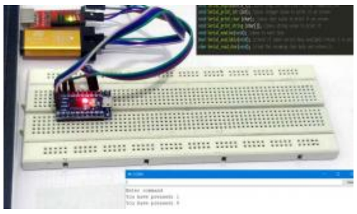
(https://bit.ly/2VT7njP)
Dev Kit Weekly: Xilinx Kria KV260 Vision AI Starter Kit (https://bit.ly/2VT7njP)

RELATED CONTENT

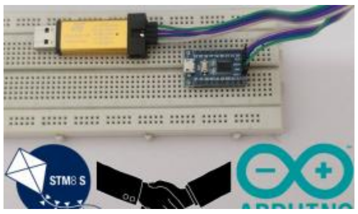
(/microcontroller-projects/pulse-width-modulation-pwm-with-stm8-using-cosmic-c-and-stvd)
Pulse width Modulation (PWM) with STM8 using Cosmic C and STVD: Controlling Brightness of LED (/microcontroller-projects/pulse-width-modulation-pwm-with-stm8-using-cosmic-c-and-stvd)

(/microcontroller-projects/adc-on-stm8s-using-c-compiler-reading-multiple-adc-values-and-displaying-on-lcd)
ADC on STM8S using Cosmic C Compiler – Reading Multiple ADC Values and Displaying on LCD (/microcontroller-projects/adc-on-stm8s-using-c-compiler-reading-multiple-adc-values-and-displaying-on-lcd)

(/microcontroller-projects/interfacing-16x2-lcd-display-with-stm8-microcontroller)
Interfacing 16x2 Alphanumeric LCD display with STM8 Microcontroller (/microcontroller-projects/interfacing-16x2-lcd-display-with-stm8-microcontroller)



(/microcontroller-projects/serial-monitor-on-stm8s-using-cosmic-and-stvd)
Serial UART Communication on STM8 using Cosmic C and STVD - Print / Read Characters and Strings (/microcontroller-projects/serial-monitor-on-stm8s-using-cosmic-and-stvd)



(/microcontroller-projects/programming-stm8s-microcontrollers-using-arduino-ide)

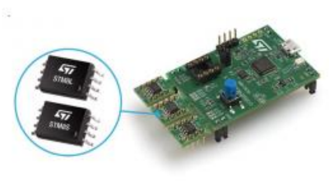
(/microcontroller-projects/getting-started-with-stm8s-using-stvd-and-cosmic-c-compiler)

Programming STM8S Microcontrollers using Arduino IDE (/microcontroller-projects/programming-stm8s-microcontrollers-using-arduino-ide)

Getting Started with STM8S using STVD and Cosmic C Compiler (/microcontroller-projects/getting-started-with-stm8s-using-stvd-and-cosmic-c-compiler)

(/news/stm32-and-stm8-controllers-get-functional-safety-packages-for-safety-critical-applications)

STM32 and STM8 Controllers get Functional Safety Packages for Safety Critical Applications (/news/stm32-and-stm8-controllers-get-functional-safety-packages-for-safety-critical-applications)



(/news/one-board-discovery-kit-contains-three-8-pin-stm8-microcontrollers)

One-Board Discovery Kit Contains Three 8-Pin STM8 Microcontrollers for Best Convenience and Value (/news/one-board-discovery-kit-contains-three-8-pin-stm8-microcontrollers)

< PREVIOUS POST

Digital Audio Volume Control Circuit using PT2258 IC and Arduino (https://circuitdigest.com/microcontroller-projects/digital-audio-volume-control-circuit-using-pt2258-ic-and-arduino)

NEXT POST >

MPU6050 Gyro Sensor Interfacing with ESP32 Board (https://circuitdigest.com/microcontroller-projects/mpu6050-gyro-sensor-interfacing-with-esp32-nodemcu-board)

COMMENTS

(/users/joseph-tannenbaum)

Joseph Tannenbaum (/users/joseph-tannenbaum)
Jun 26, 2020

Hi, Thought I would mention, I got a batch of these from

Log in (/user/login?destination=node/6456%23comment-form) or register (/user/register?destination=node/6456%23comment-form) to post comments

hiletgo and found the ground pin on the programming pins was not really ground on the board, so the ST-Link's ground had to go to the board ground instead. Also out of the box, either the chip or the ST-Visual programmer said the chip was protected. found this: https://electronics.stackexchange.com/questions/128931/how-to-flash-protected-stm8-mcu (https://electronics.stackexchange.com/questions/128931/how-to-flash-protected-stm8-mcu) which tells how to undo this.

Do you have a list of all your projects with this board like you did with the PIC16F877A?

Joe

(/users/joseph-tannenbaum)

Joseph Tannenbaum (/users/joseph-tannenbaum)
Jun 26, 2020

Hi,

Log in (/user/login?destination=node/6456%23comment-form) or register (/user/register?destination=node/6456%23comment-form) to post comments

Bought a batch of these from Hiletgo and had this issue:

1. The ground programming pin was not ground.

2. The Programmer said "Device is protected"

For issue #1, connect the ST-Link to the board's ground pin

For issue #2 I found this:

https://electronics.stackexchange.com/questions/128931/how-to-flash-protected-stm8-mcu (https://electronics.stackexchange.com/questions/128931/how-to-flash-protected-stm8-mcu)

Hope this helps someone.

Are you going to do a series on this board as you did with the PIC16F877A? Hope you do.

Joe

(/users/violet)

Violet (/users/violet)
Sep 18, 2020

Hey, when will you publish the full tutorial? Please share the list of all

Log in (/user/login?destination=node/6456%23comment-form) or register (/user/register?destination=node/6456%23comment-form) to post comments

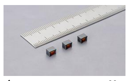
when ready

Thanks

LOG IN (/USER/LOGIN?DESTINATION=NODE/6456%23COMMENT-FORM) OR REGISTER (/USER/REGISTER?DESTINATION=NODE/6456%23COMMENT-FORM) TO POST COMMENT

(https://bit.ly/3qzgJfD
)

NEWS ARTICLES PROJECTS


(/news/small-sized-broadband-inductors-for-high-impedance-performance-in-vehicle-mounted-poc)

Small-Sized Broadband Inductors for High-Impedance Performance in Vehicle-Mounted PoC (/news/small-sized-broadband-inductors-for-high-impedance-performance-in-vehicle-mounted-poc)

(/news/oceanmedallion-wearable-devices-integrated-with-dialog-semiconductor%E2%80%99s-wira-technology-for-enhanced-user-experience)

OceanMedallion Wearable Devices Integrated with Dialog Semiconductor's WiRa Technology for Enhanced User Experience (/news/oceanmedallion-wearable-devices-integrated-with-dialog-semiconductor%E2%80%99s-wira-technology-for-enhanced-user-experience)

(/news/new-prototype-robocar-with-second-generation-ai-chip-designed-for-autonomous-driving)

New Prototype "Robocar" with Second-Generation AI Chip designed for Autonomous Driving (/news/new-prototype-robocar-with-second-generation-ai-chip-designed-for-autonomous-driving)

(/news/iit-jodhpur-researchers-develops-new-blockchain-technology-to-secure-iot-networks)

IIT Jodhpur Researchers develops new blockchain technology to secure IoT networks (/news/iit-jodhpur-researchers-develops-new-blockchain-technology-to-secure-iot-networks)

(/news/asia%E2%80%99s-most-valuable-firm-is-now-tsmc-overtakes-tencent-and-alibaba)

Asia's most valuable firm is now TSMC; overtakes Tencent and Alibaba (/news/asia%E2%80%99s-most-valuable-firm-is-now-tsmc-overtakes-tencent-and-alibaba)

Subscribe