MICROCONTROLLERS

Chapter 7

STM32 Peripherials - GPIO

Dr. Saeed Ebadollahi

References:

- ARM® Cortex® M4 Cookbook – Mark Fischer – Packt publishing – 2016
- The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors – Joseph Yiu – Newnes – 2014
- Discovering the STM32 Microcontroller - Geoffrey Brown - 2012

Clock distribution

In the world of embedded processors, power consumption is critical; hence, most sophisticated embedded processors provide mechanisms to power down any resources that are not required for a particular application. The STM32 has a complex clock distribution network which ensures that only those peripherals that are actually needed are powered. This system, called Reset and Clock Control (RCC) is supported by the firmware module `stm32f10x_rcc.[ch]`. While this module can be used to control the main system clocks and PLLs, any required configuration of those is handled by the startup code provided with the examples in this book. Our concern here is simply with enabling the peripheral clocks.

Clock distribution (Cont.)

The STM32 peripherals are organized into three distinct groups called APB1, APB2, and AHB. APB1 peripherals include the I2C devices, USARTs 2-5, and SPI devices; APB2 devices include the GPIO ports, ADC controllers and USART 1. AHB devices are primarily memory oriented including the DMA controllers and external memory interfaces (for some devices)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM17	TIM16	TIM15
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	URT1	Res.	SPI1	TIM1	Res.	ADC1	IOPG	IOPF	IOPE	IOPD	IOPC	IOPB	IOPA	Res.	AFIO

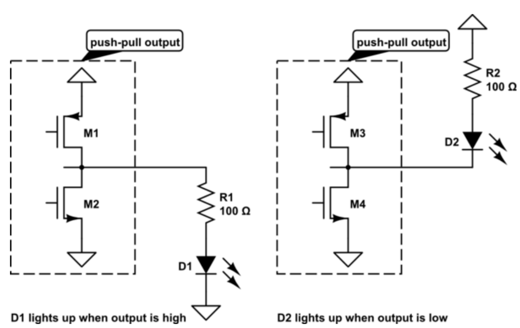
Figure 4.2: APB2 Peripheral Clock Enable Register

What is GPIO ?

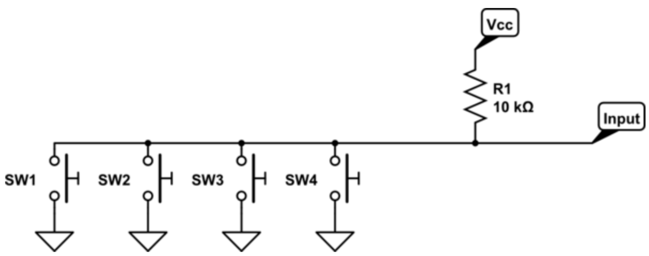
General-purpose input/output (**GPIO**) is a generic pin on an integrated circuit or computer board whose behavior—including whether it is an input or output pin—is controllable by the user at run time.

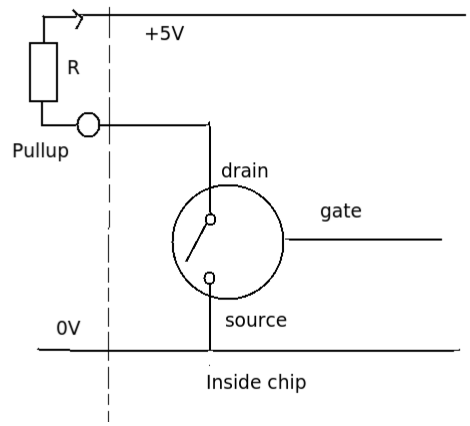
GPIO pins have no predefined purpose, and go unused by default.[1][2] The idea is that sometimes a system integrator who is building a full system might need a handful of additional digital control lines—and having these available from a chip avoids having to arrange additional circuitry to provide them. For example, the Realtek ALC260 chips (audio codec) have 8 GPIO pins, which go unused by default. Some system integrators (Acer Inc. laptops) use the first GPIO (GPIO_0) on the ALC260 to turn on the amplifier for the laptop's internal speakers and external headphone jack.

Output Push-Pull Circuit

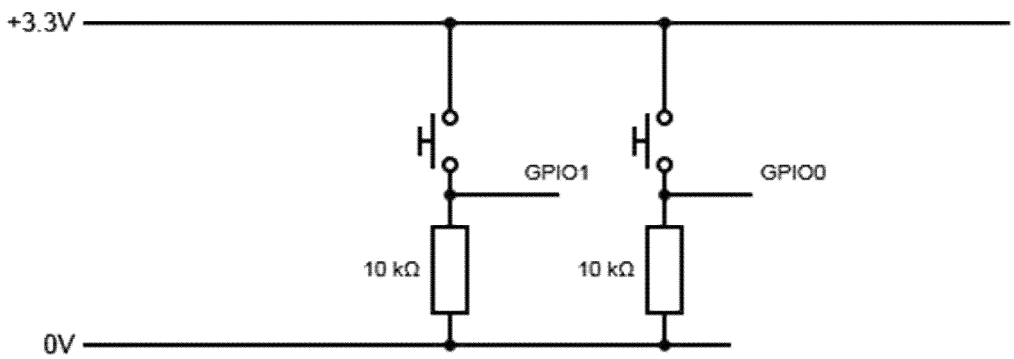


Output Open-Drain Circuit





Input Circuit



GPIO Features

Function	Library Constant
Alternate function open-drain	GPIO_Mode_AF_OD
Alternate function push-pull	GPIO_Mode_AF_PP
Analog	GPIO_Mode_AIN
Input floating	GPIO_Mode_IN_FLOATING
Input pull-down	GPIO_Mode_IPD
Input pull-up	GPIO_Mode_IPU
Output open-drain	GPIO_Mode_Out_OD
Output push-pull	GPIO_Mode_Out_PP

HAL Driver

In computers, a hardware abstraction layer (HAL) is a layer of programming that allows a computer operating system to interact with a hardware device at a general or abstract level rather than at a detailed hardware level. Windows 2000 is one of several operating systems that include a hardware abstraction layer. The hardware abstraction layer can be called from either the operating system's kernel or from a device driver. In either case, the calling program can interact with the device in a more general way than it would otherwise.

How to use GPIO HAL driver?

Example 1:

- Write a code to turn on a LED on your discovery board:
  1. Find board pins that have a LED on it
  2. Configure that pin in STM32Cube
  3. Use uvision keil with HAL to turn LED on

Delay

Whenever we want MCU to wait we can use this function

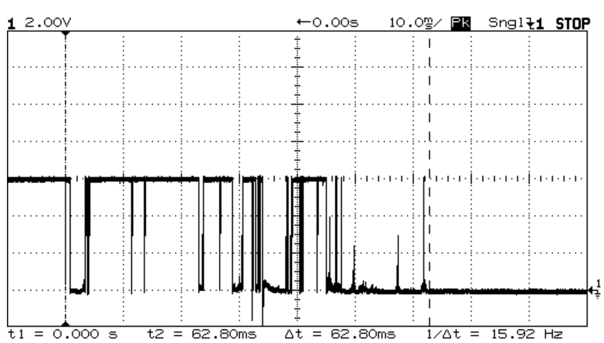
Example 2:

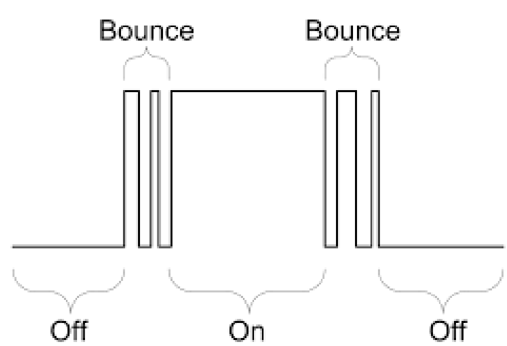
- Write a code to blink a LED on your discovery board:
  1. Find board pins that have a LED on it
  2. Configure that pin in STM32Cube
  3. Use uvision keil with HAL blink LED

Example 3:

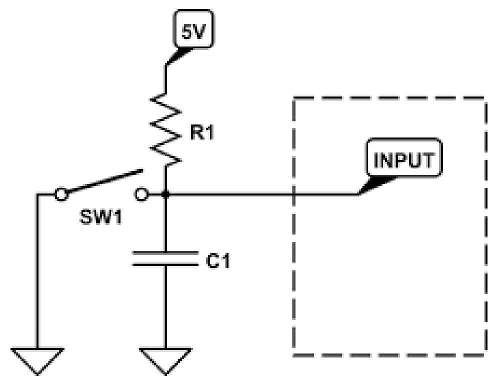
- Write a code to change LED condition by pushing a button on your discovery board:
  1. Find board pins that have a push button on it
  2. Configure that pin in STM32Cube

Bounce





Debouncing circuit



Example 4:

- Find a way to debounce push button without that circuit and only with code.