فصل سوم : General Purpose Input and Outputs (GPIOs)

وضعیت‌های پایه‌های ورودی و خروجی

- **Input floating**
- **Input pull-up**
- **Input pull-down**
- **Analog**
- **Output open-drain**
- **Output push-pull**
- **Alternate function push-pull**
- **Alternate function open-drain**

ثبات‌های تعریف وضعیت ورودی و خروجی

- دو رجیستر به منظور اعمال تنظیمات (GPIOx_CRL, GPIOx_CRH)
- دو رجیستر 32 بیتی به منظور خواندن یا نوشتن دیتا (GPIOx_IDR, GPIOx_ODR)
- یک رجیستر 32 بیتی به منظور مقداردهی (Set/Reset) (GPIOx_BSRR)
- یک رجیستر 16 بیتی به منظور ریست کردن (GPIOx_BRR)
- یک رجیستر 32 بیتی به منظور قفل گذاری بر روی پایه‌ها (GPIOx_LCKR)

رجیسترهای تنظیمات (CRL, CRH)

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01 10 11 see *Table 21*		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			Don't care
	Open-drain		1			Don't care
Input	Analog	0	0	00		Don't care
	Input floating		1			Don't care
	Input pull-down	1	0			0
	Input pull-up					1

MODE[1:0]	Meaning
00	Reserved
01	Maximum output speed 10 MHz
10	Maximum output speed 2 MHz
11	Maximum output speed 50 MHz

- برای هر یک پایه، دو بیت **Mode** و دو بیت **CNF** وجود دارد.
- با در نظر گرفتن 32 بیتی بودن رجیسترها و اختصاص 4 بیت به هر پایه، یک رجیستر می‌تواند وضعیت 8 پایه را مشخص نماید.
- بنابراین دو رجیستر **CRL** و **CRH** وجود دارد تا بتواند هر 16 پایه از یک خانواده ورودی خروجی را پوشش دهد.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

رجیسترهای دیتا (IDR, ODR)

- دسترسی به این رجیسترها به صورت 32 بیتی می‌باشد و امکان دسترسی‌در ابعاد Half-word (16 بیت)، Byte (8 بیت) و بیت وجود ندارد.
- رجیستر IDR بیانگر وضعیت هر یک از پایه‌های ورودی می‌باشد. بنابراین با خواندن این رجیستر، می‌توانیم وضعیت پایه ورودی را دریابیم (0 یا 1).
- رجیستر ODR قابلیت نوشتن و خواندن دارد. هنگام نوشتن، وضعیت پایه به مقدار نوشته شده تغییر می‌یابد و هنگام خواندن، وضعیت پایه خروجی مشخص می‌گردد.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

رجیستر مقداردهی (BSRR)

- **این رجیستر قابلیت مقداردهی در به هر پایه را ارائه می‌دهد**
- **مثال : با نوشتن 1 در بیت صفر این رجیستر (معادل با BS0) مقدار خروجی پایه متناظر با آن 1 می‌شود.**
  همچنین با نوشتن در 1 در بیت 16 (معادل با BR0) مقدار خروجی هم همان پایه، 0 می‌شود.
- **مقداردهی با این رجیستر به صورت Atomic اعمال می‌گردد.!؟**
- **در صورت نوشتن مقدار صفر!؟**

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

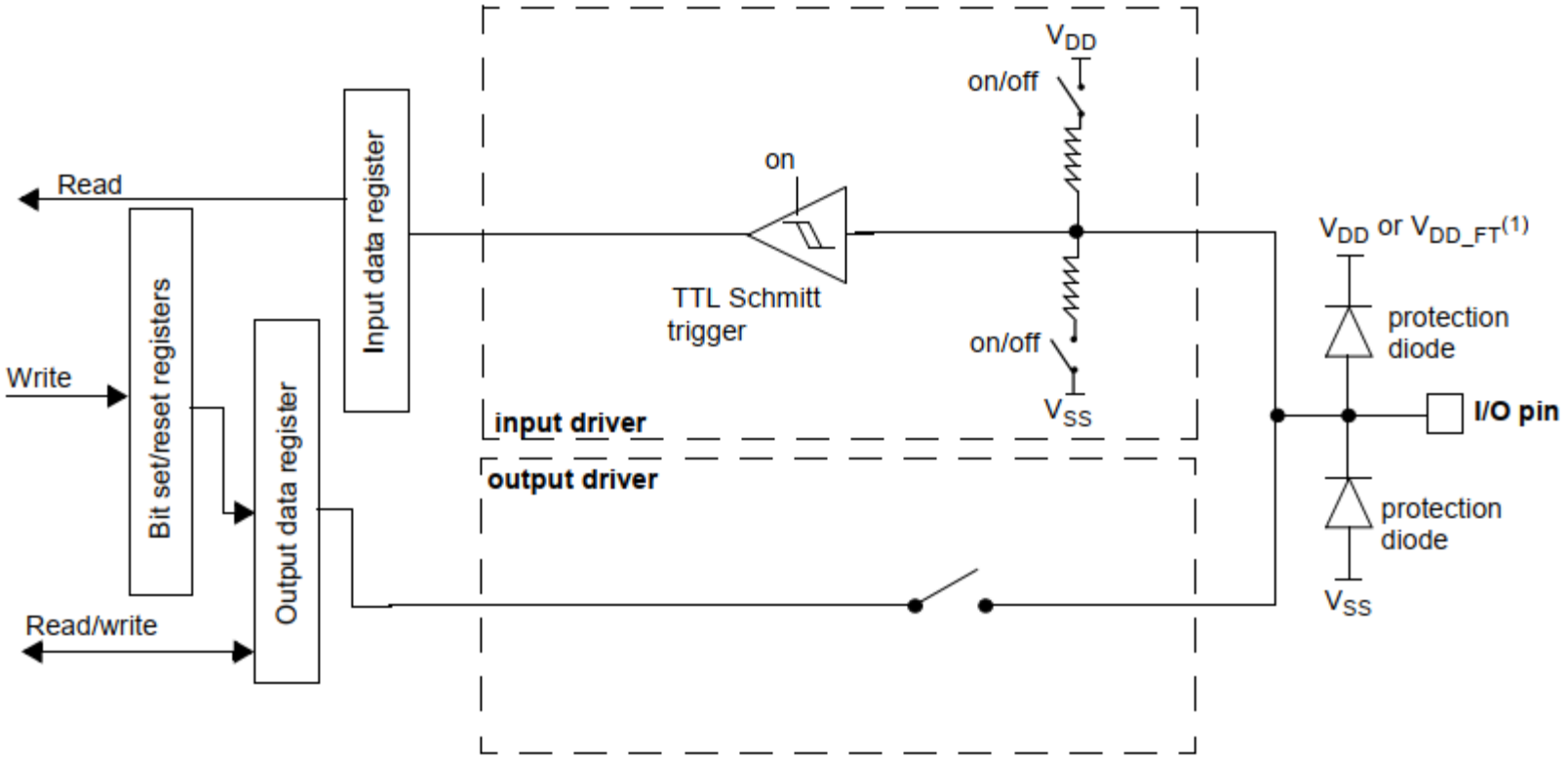
رجیستر ریست (BRR)

- **این رجیستر برای ریست یا 0 کردن خروجی استفاده می‌گردد و معادل با بیت‌های ۱۶ تا ۳۱ از رجیستر BSRR می‌باشد.**
- **مثال : با نوشتن مقدار 1 در بیت صفر (معادل با BR0)، خروجی متناظر با پایه آن ریست می‌گردد.**
- **چرا یک رجیستر مجزا ارائه شده است؟!**
- **نوشتن مقدار 0 در بیت BR0!؟**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

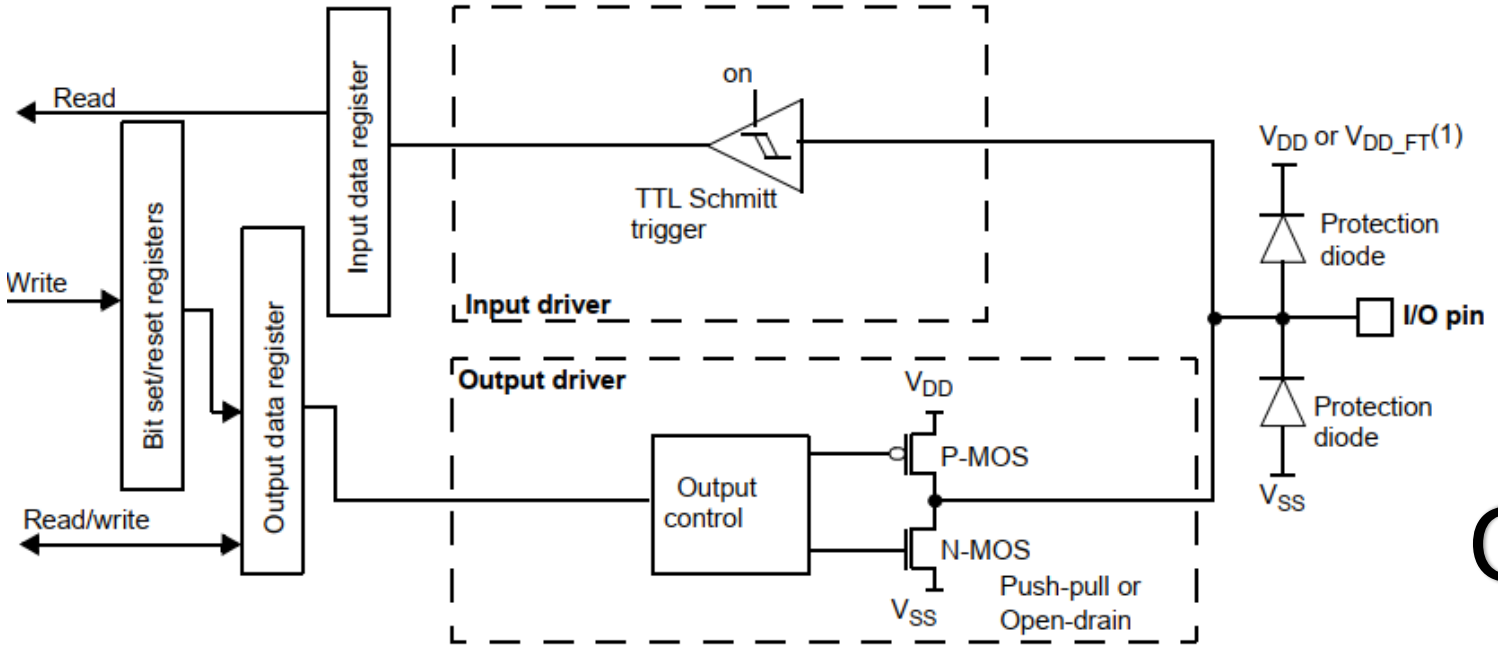
معماری پایه‌های Input

- غیرفعال‌سازی بخش خروجی
- فعال‌سازی Schmitt Trigger
- فعال‌سازی وضعیت Pull-up، Pull-down یا Floating
- در هر APB2 clock، مقادیر پایه‌های ورودی در رجیستر دیتای ورودی (IDR) نوشته می‌شود.



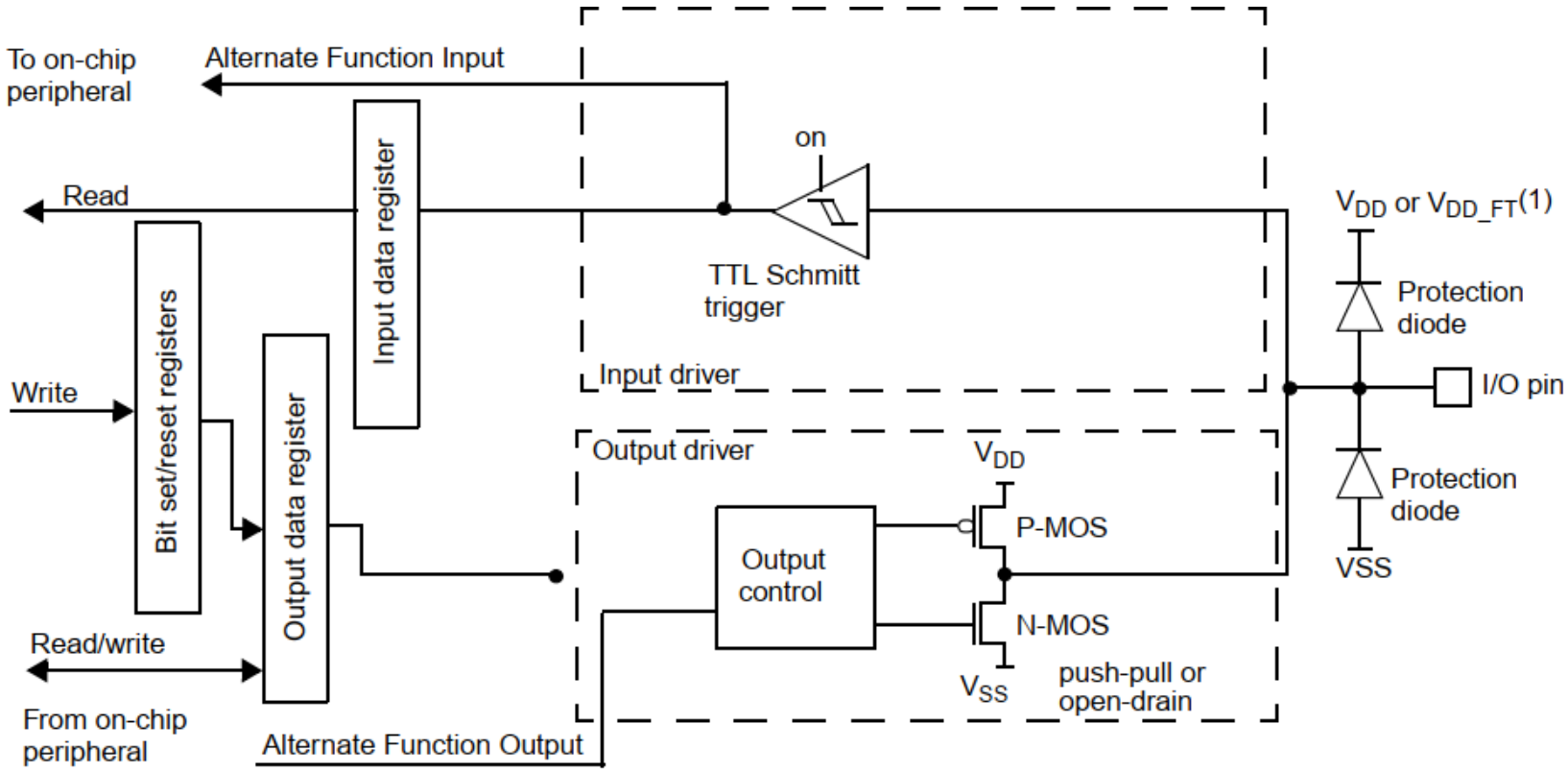
معماری پایه‌های Output

- فعال‌سازی بخش خروجی
  - خروجی از نوع push-pull : نوشتن 0 در رجیستر خروجی ترانزیستور N-MOS را روشن نموده، بنابراین خروجی صفر می‌گردد. نوشتن مقدار 1 در رجیستر خروجی، ترانزیستور P-MOS را روشن می‌کندکه سبب می‌شود تا خروجی 1 گردد.
  - خروجی از نوع open-drain : در این حالت ترانزیستور P-MOS وجود ندارد، بنابراین با نوشتن مقدار 0 در رجیستر خروجی، ترانزیستور N-MOS روشن می‌شود و خروجی صفر می‌گردد. نوشتن مقدار 1 در رجیستر خروجی سبب می‌گردد تا ترانزیستور N-MOS خاموش گردد، باتوجه به عدم وجود ترانزیستور P-MOS، مقدار در وضعیت اتصال‌باز یا نامعلوم (HiZ) قرار می‌گیرد.
- فعال‌سازی Schmitt Trigger
- غیرفعال‌سازی وضعیت Pull-up، Pull-down
- در هر APB2 clock، مقادیر پایه‌های ورودی در رجیستر دیتای ورودی (IDR) نوشته می‌شود.



چرا Open-drain!؟

معماری پایه‌های Alternate Function

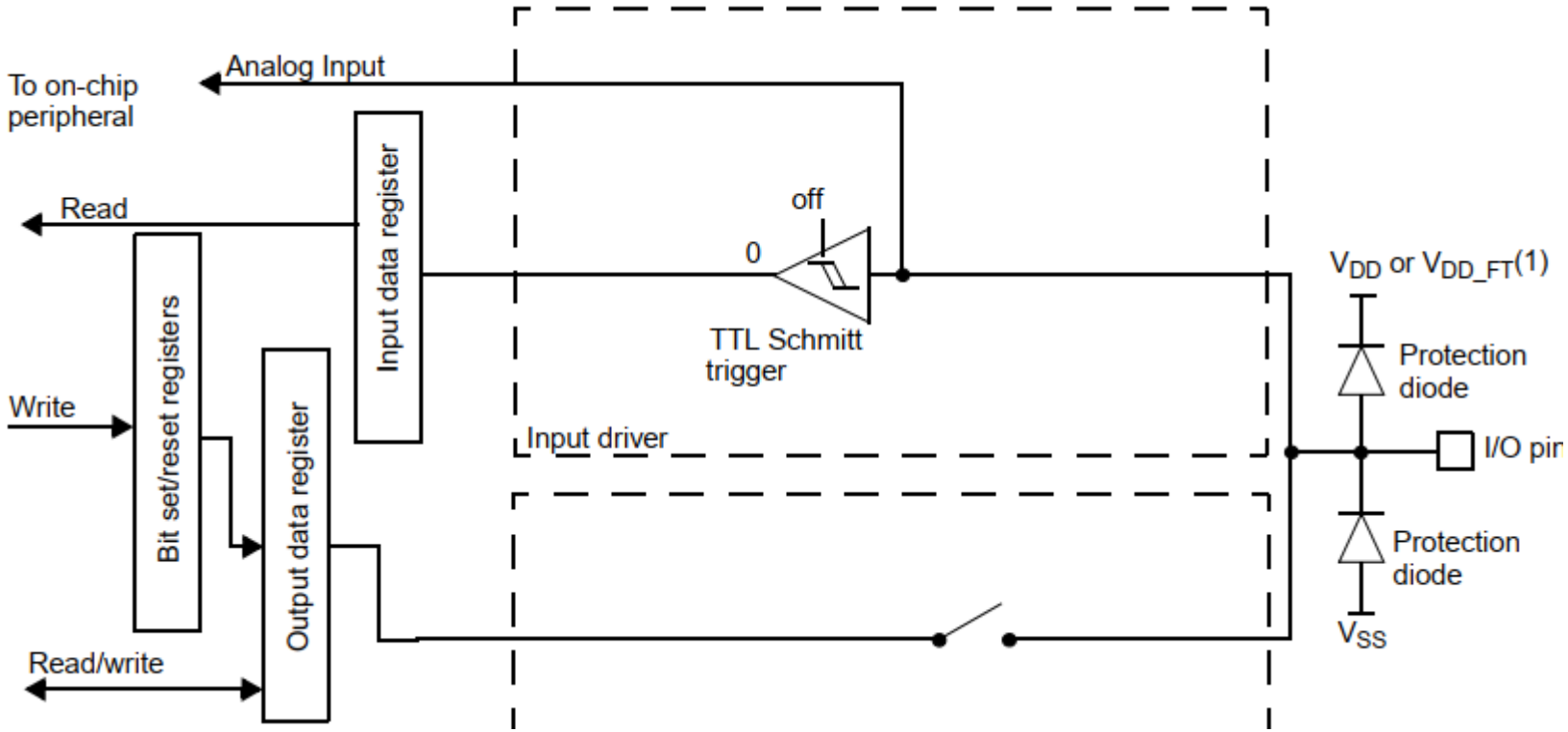


- فعال‌سازی مدار خروجی در یکی از دو معماری Open-drain یا Push-pull با این تفاوت که مدار خروجی توسط واحد مطلوب درایو می‌شود.
- فعال‌سازی Schmitt Trigger
- غیرفعال‌سازی وضعیت Pull-up، Pull-down
- در هر APB2 clock، مقادیر پایه‌های ورودی

در رجیستر دیتای ورودی (IDR) نوشته می‌شود.

معماری پایه‌های Analog

- غیرفعال‌سازی مدار خروجی
- غیر فعال‌سازی Schmitt Trigger
- غیرفعال‌سازی وضعیت Pull-up، Pull-down
- باتوجه به صفر بودن خروجی Schmitt Trigger، هنگامی که پایه در وضعیت Analog قرار داده شود، خواندن رجیستر دیتای ورودی (IDR) همواره مقدار 0 را برمی‌گرداند.



External Interrupt/Wakeup Lines قابلیت

- تمامی پایه‌های میروکنترولر قابلیت اتصال به واحد وقفه را دارند. در این حالت بایستی پایه در ساختار ورودی قرار گیرد. بنابراین اگر تغییراتی در محیط پیرامون ایجاد گردد که منجر به تغییر در مقدار پایه میکرو گردد، می‌تواند واحد وقفه یا بیدار کردن پردازنده را اجرا نمایید.
- ادامه مطلب در فصل وقفه‌ها...

قابلیت Remapping

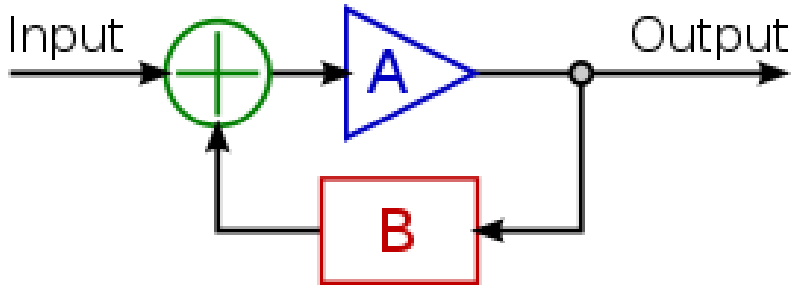
- با توجه به تراکم واحدهای مختلف، این قابلیت وجود دارد که پورت‌های ورودی و خروجی یک واحد را از حالت پیش‌فرض به حالت ثانویه تغییر داد تا آن واحد از پایه‌های دیگری برای ورودی و خروجی خود استفاده نماید.
- مثال : فرض کنید در یک پروژه به هر دو واحد Timer1 (channel2) و ارتباط سریال UART نیازمندید. هر دو واحد از پایه PA9 در میکروکنترولر stm32f103c8 استفاده می‌کنند! بنابراین به ناچار بایستی یکی از آن‌ها را استفاده نمود. اما با قابلیت Remapping، این امکان وجود دارد تا پایه‌های یکی از دو واحد فوق را تغییر داد تا بتوان به صورت همزمان از هر دو واحد استفاده نمود.
- کاربردهای دیگر :
  - قابلیت ساده‌سازی سیم‌کشی در PCB
  - قابلیت جداسازی بخش‌های آنالوگ و دیجیتال
  - ...

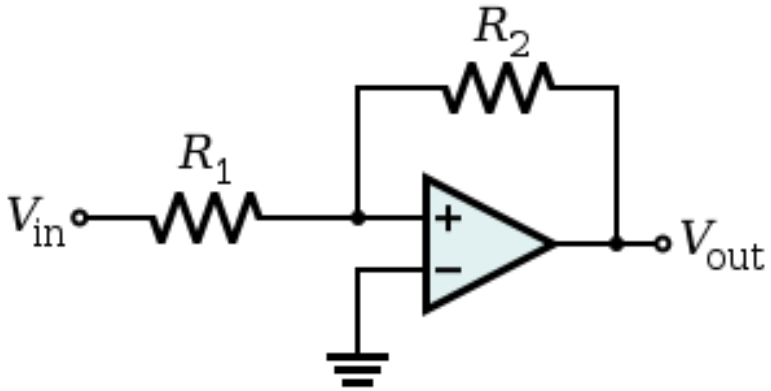
تمرین

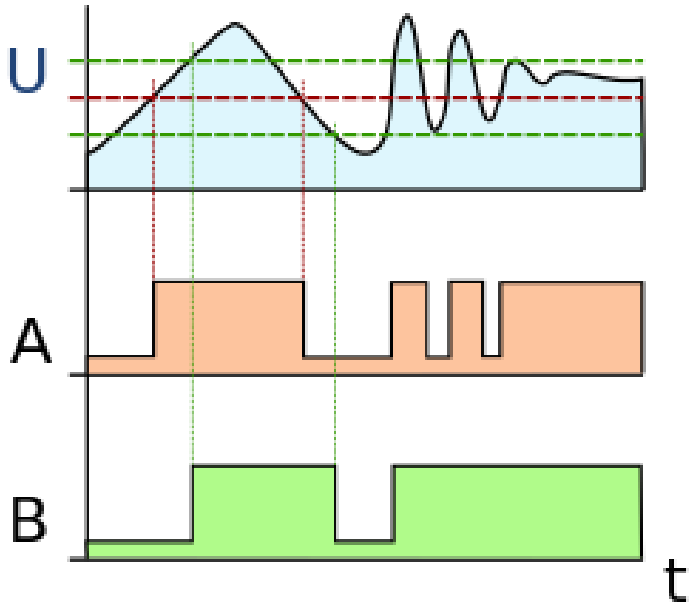
- **برنامه‌ای بنویسید که بیت A0 را بخواند!**
- **برنامه‌ای بنویسید که بیت A1 را در حالت خروجی قرار دهد. اگر بیت A0 برابر 0 بود، خروجی A1 نیز 0 شود، اگر بیت A1 برابر با 1 بود، خروجی A1 نیز 1 گردد.**

Schmitt Trigger

- **چرا باید از Schmitt Trigger استفاده کرد؟**

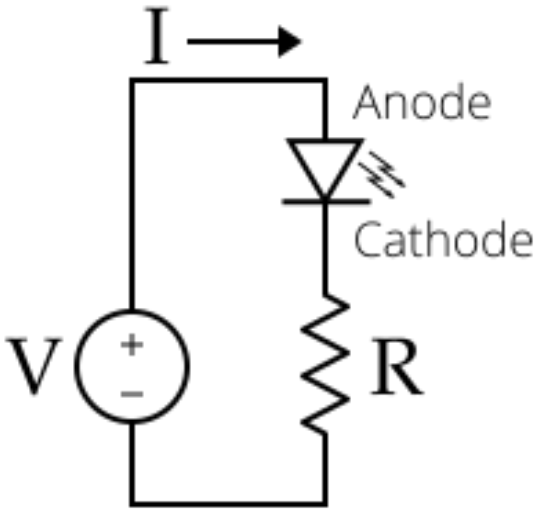




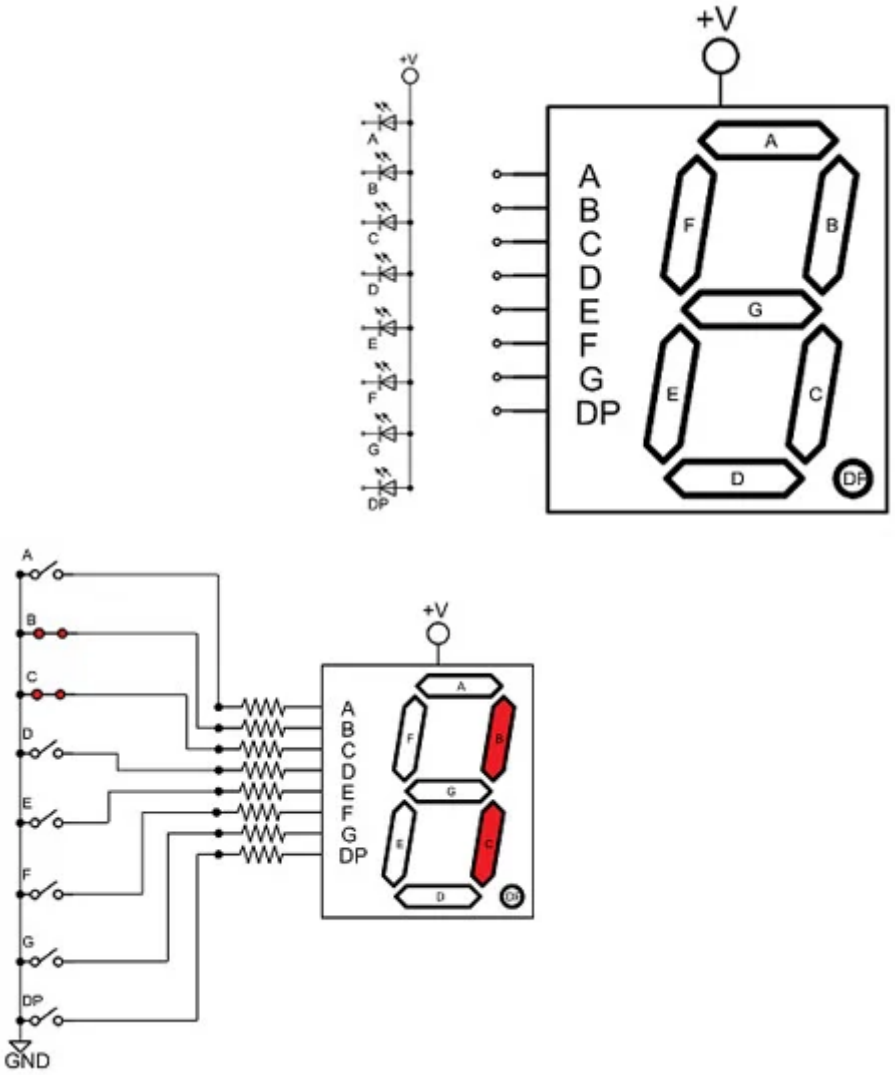


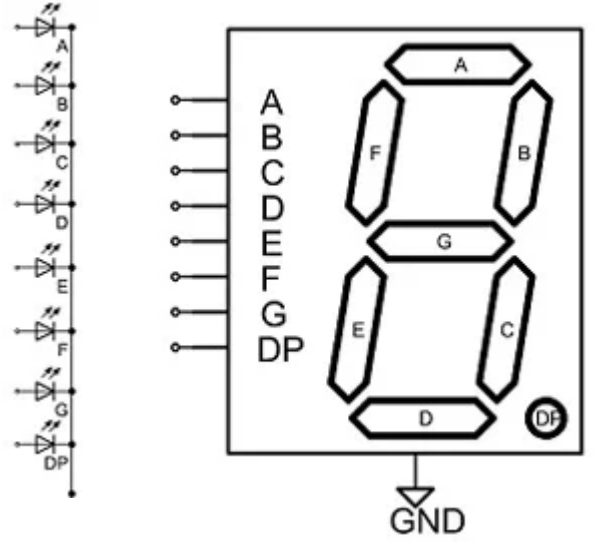
Seven Segment

- یکی از انواع نمایشگرها، نمایشگر Seven Segment می‌باشد.
- این نمایشگرها که از 8 LED نورانی تشکیل شده‌اند.
- دو نوع Seven Segment موجود است :
  - آند مشترک (Common Anode)
  - کاتد مشترک (Common Cathode)
- نمایشگرهای برپایه Seven Segment دارای ابعاد مختلفی هستند.
- در Seven Segmentهای بزرگ، باتوجه به افزایش جریان مصرفی، به منظور عدم آسیب به پردازنده بایستی از مدارات جانبی استفاده نماییم. چه مداری؟!

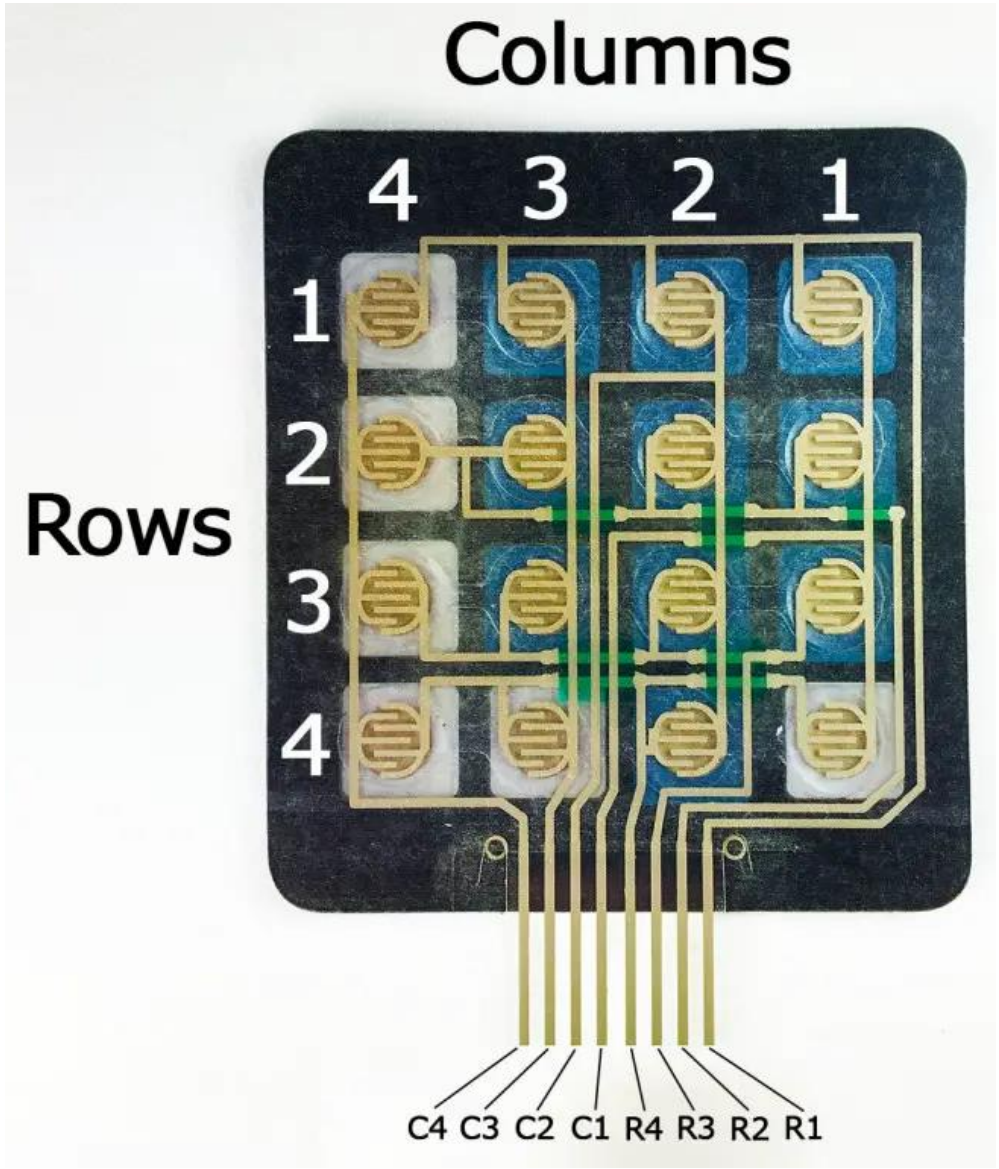


Common Anode VS Common Cathode

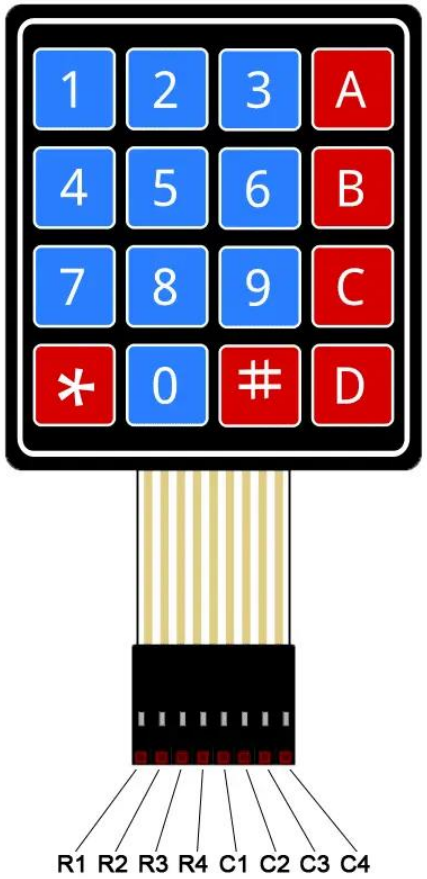




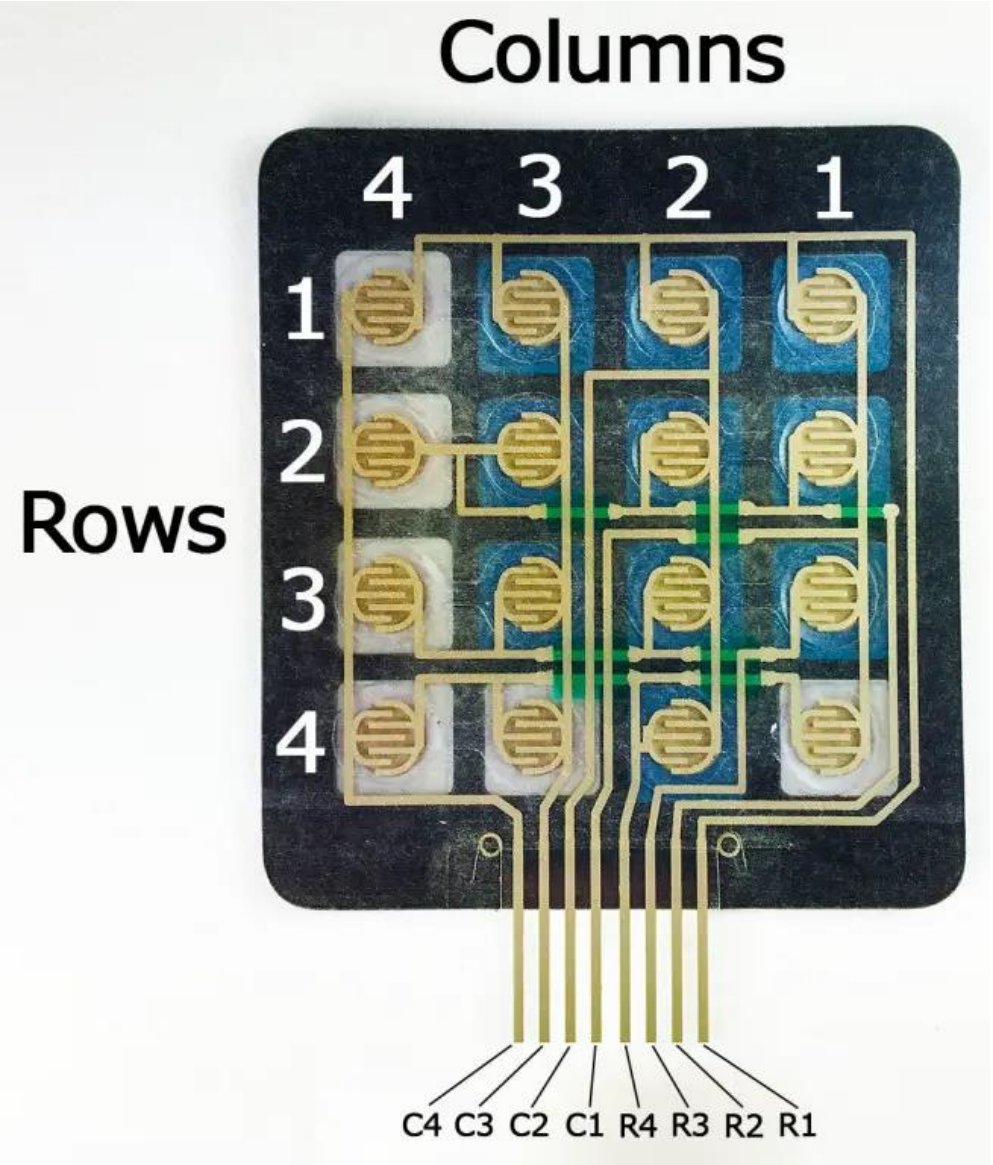
Keypad

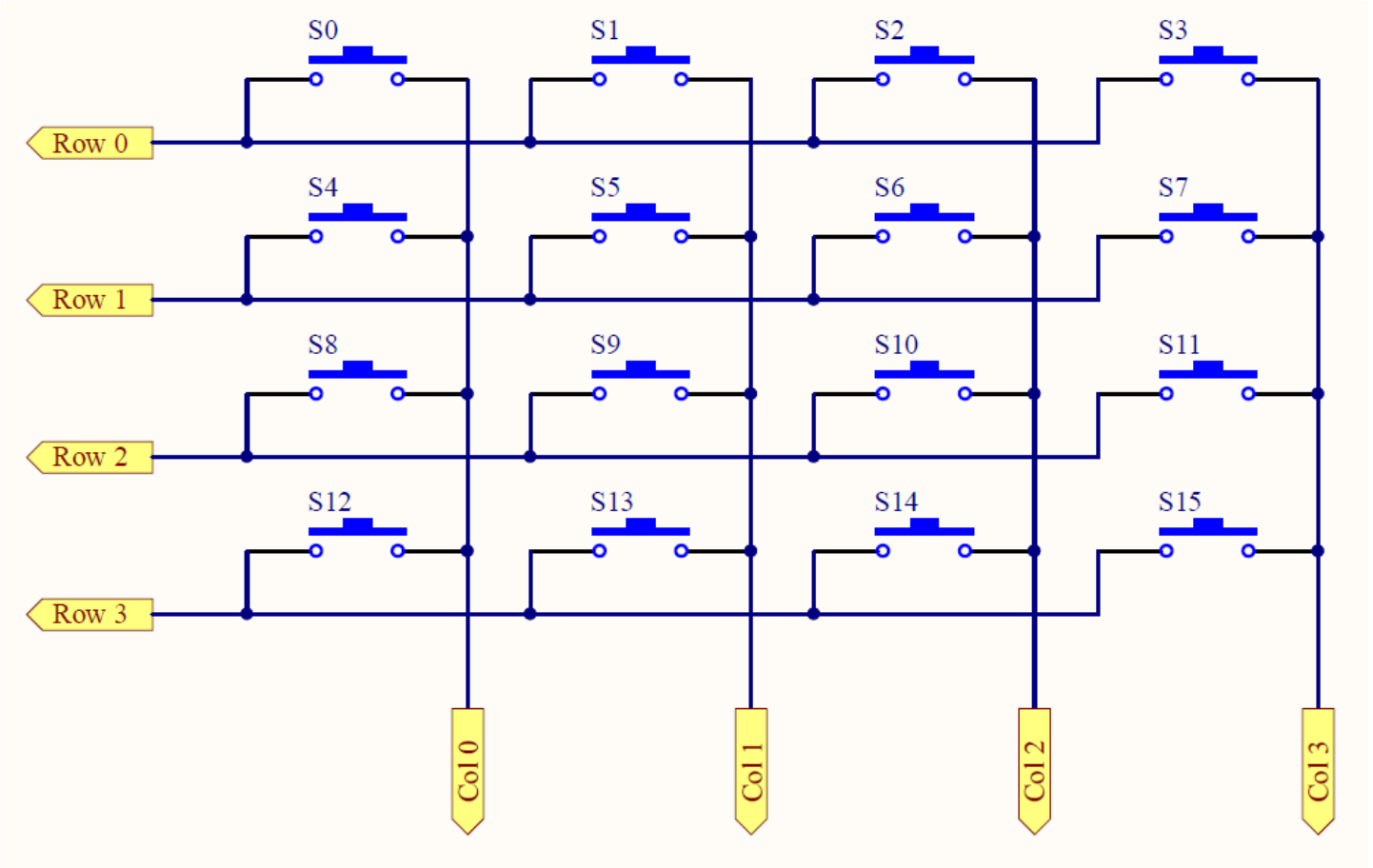

Columns
Rows
4 3 2 1
1
2
3
4
C4 C3 C2 C1 R4 R3 R2 R1


1 2 3
4 5 6
7 8 9
* 0 #
R1 R2 R3 R4 C1 C2 C3


1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D
R1 R2 R3 R4 C1 C2 C3 C4

Keypad


Columns
4 3 2 1
Rows
1
2
3
4
C4 C3 C2 C1 R4 R3 R2 R1


S0
S1
S2
S3
Row 0
S4
S5
S6
S7
Row 1
S8
S9
S10
S11
Row 2
S12
S13
S14
S15
Row 3
Col 0
Col 1
Col 2
Col 3

جدول درستی منطق‌های AND و OR

OR

A	B	X
0	0	0
1	0	1
0	1	1
1	1	1

AND

A	B	X
0	0	0
1	0	0
0	1	0
1	1	1

- در جدول OR حتی اگر یکی از ورودی‌ها 1 باشد، ورودی دیگر مهم نیست و نتیجه 1 می‌شود. بنابراین برای 1 کردن هر بیت، می‌توانیم آن را با بیت 1، OR نماییم.
- در جدول AND حتی اگر یکی از ورودی‌ها 0 باشد، ورودی دیگر مهم نیست و نتیجه 0 می‌شود. بنابراین برای 0 کردن هر بیت، می‌توانیم از آن را با بیت 0، AND نماییم.

- ویژگی دیگر گیت AND از مقایسه، سطر دوم و چهارم جدول آن بدست می‌آید. جاییکه مقدار A=1 بوده، در این حالت وضعیت بیت B تعیین کننده وضعیت بیت خروجی می‌باشد. بنابراین هنگامی که می‌خواهیم تعدادی بیت را از یک مجموعه بیت تفکیک کنیم، کافی است بیت‌های مطلوب را با 1، AND نماییم. در اینصورت اگر بیت AND شده، صفر بوده باشد، مقدار خروجی نیز 0 می‌گردد و اگر مقدار آن 1 بوده باشد، مقدار خروجی نیز 1 می‌گردد. **(مهم، حتما با مثال بیان شود)**

برنامه‌های مورد نیاز برای کار با GPIOs به صورت رجیستری

- گام اول : تعیین تنظیمات نوع پایه و مدار آن به کمک رجیسترهای GPIOx_CRL و GPIOx_CRH (اگر از نرم‌افزار Stm32Cube استفاده می‌کنید، این مرحله به صورت خودکار توسط نرم‌افزار انجام می‌شود.)
- گام دوم : اگر پایه از نوع ورودی باشد، تنها قابلیت خواندن وجود دارد. بنابراین با رجیستر GPIOx_IDR می‌توانیم مقدار آن را بخوانیم.
- گام سوم : اگر پایه از نوع خروجی باشید، بایستی وضعیت آن (صفر یا یک) را تعیین نماییم که برای آن سه رجیستر وجود دارد.
  - رجیستر ODR : قابلیت صفر و یک کردن دارد.
  - رجیستر BSRR : قابلیت صفر و یک کردن دارد. (atomic)
  - رجیستر BRR : قابلیت صفر کردن دارد. (atomic)

بهترین کار استفاده از رجیستر BSRR برای یک کردن و استفاده از رجیستر BRR برای ریست کردن پایه‌ها است.

مثال : !؟

برنامه‌های مورد نیاز برای کار با GPIOs با استفاده از توابع کتابخانه HAL

- گام اول : تعیین تنظیمات نوع پایه که به صورت خودکار توسط نرم افزار Stm32Cube انجام می‌شود.
- گام دوم : اگر پایه از نوع ورودی باشد، تنها قابلیت خواندن وجود دارد. بنابراین با استفاده از تابع HAL_GPIO_ReadPin(GPIOx, GPIO_Pin) مقدار آن را می‌خوانیم.
- گام سوم : اگر پایه از نوع خروجی باشید، بایستی وضعیت آن (صفر یا یک) را تعیین نماییم که برای آن از تابع HAL_GPIO_WritePin(GPIOx, GPIO_Pin, GPIO_State) استفاده می‌کنیم. تابع WritePin از رجیستر BSRR استفاده می‌کند.

مثال : !؟

توابع HAL کاربردی در مبحث GPIO

انتخاب شماره پایه بین 0 تا 15
مثال : GPIO_PIN_5

HAL_GPIO_ReadPin(GPIOx, GPIO_Pin)

انتخاب خانواده نظیر A، B، C و ...
مثال : GPIOB

HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_5)

توابع HAL کاربردی در مبحث GPIO

انتخاب شماره پایه بین 0 تا 15
مثال : GPIO_PIN_5

HAL_GPIO_WritePin(GPIOx, GPIO_Pin, PinState)

انتخاب خانواده نظیر A، B، C و ...
مثال : GPIOB

تعیین وضعیت خروجی بین 0 یا 1 با استفاده از عبارت GPIO_PIN_SET برای مقدار 1 یا استفاده از مقدار GPIO_PIN_RESET برای مقدار 0 البته این قابلیت وجود دارد که مستقیما از 0 یا 1 نیز استفاده شود.
مثال : GPIO_PIN_SET یا 1

HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, 1)

HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET)

توابع HAL کاربردی در مبحث GPIO

انتخاب شماره پایه بین 0 تا 15
مثال : GPIO_PIN_5

HAL_GPIO_TogglePin(GPIOx, GPIO_Pin)

انتخاب خانواده نظیر A، B، C و ...
مثال : GPIOB

HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5)