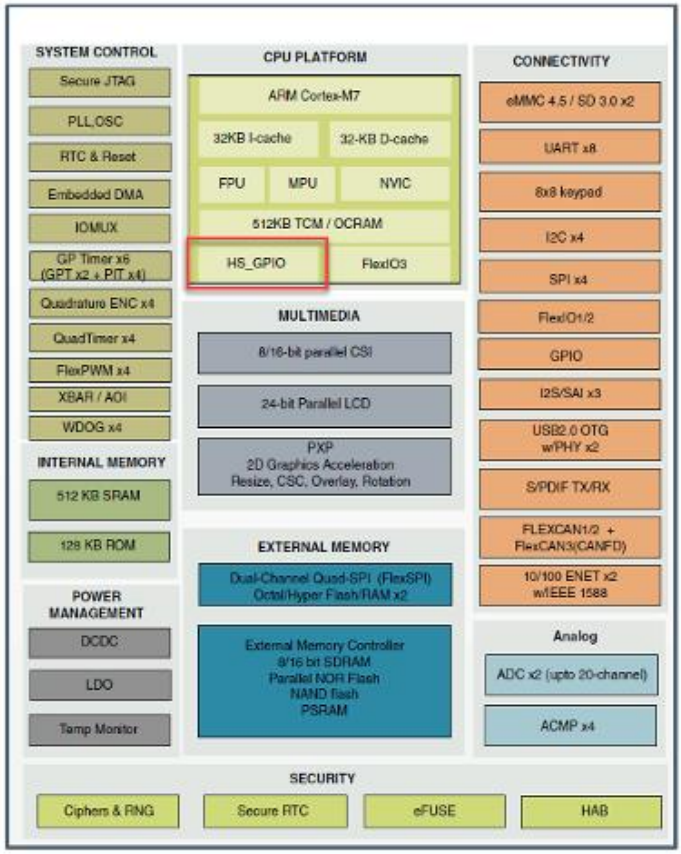
RT1060 的普通 GPIO 和 快速 GPIO 的对比

原文：https://community.nxp.com/docs/DOC-342954

i.MXRT1060 提供了紧耦合的通用 GPIO，可以进行高频率的操作。



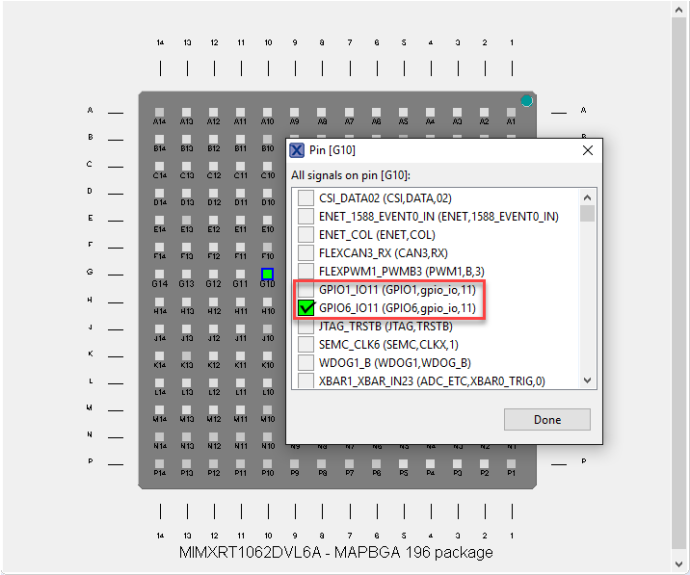
RT1060 提供了两组 GPIO 寄存器来控制端口输出。GPIO 1 到 GPIO 3 是通用 GPIO，而 GPIO 6 到 GPIO 8 是紧耦合 GPIO，但是它们共享同一个端口，这意味着 GPIO 引脚可以用 GPIO 1/2/3 或 GPIO 6/7/8 进行选择。

寄存器 IOMUXC_GPR_GPR26、IOMUXC_GPR_GPR27 和 IOMUXC_GPR_GPR28 也用于 GPIO 的选择。

IOMUXC_GPR_GPR26 field descriptions

Field	Description
GPIO_MUX1_GPIO_SEL	GPIO1 and GPIO6 share same IO MUX function, GPIO_MUX1 selects one GPIO function. This register controls GPIO_MUX1 to select GPIO1 or GPIO6. For bit *n*, • 0: GPIO1[*n*] is selected; • 1: GPIO6[*n*] is selected.

要用 GPIO 1/2/3 或 GPIO 6/7/8 选择 GPIO 引脚，你可以使用 MCUXpresso Config 工具。例如，如果你选择 G10 引脚，你既可以选择 GPIOI_IO11 作为普通 GPIO，也可以选择 GPIO 6_IO11 作为快速 GPIO。



我基于 SDKV2.5 做了一个例子，比较普通 GPIO 和快速 GPIO 的速度。为此，我使用了引脚 G10 (GPIOI_IO11 和 GPIO 6_IO11)。

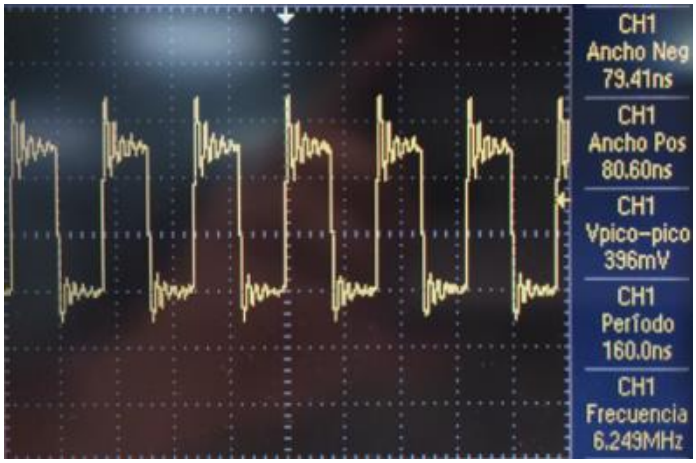
注意：正如下图所示，GPIO 1、2、3、4 和 5 的时钟来自 IPG_CLK_ROOT，而 GPIO 6、7、8 和 9 来自 AHB_CLK_ROOT。这意味着对于 GPIO 6，7，8 和 9，你不需要激活任何时钟。SDK 的当前版本(v2.5)没有考虑到这一点，因此如果你想使用这些 GPIO 中的任何一个，你需要在 GPIO_PinInit 函数中添加以下 if 语句。这个错误已经被报告，并且将在 SDK 的新版本中被修复。

GPIO*n*	gpio1_ipg_clk_s	ipg_clk_root	CCGR1[CG13] (gpio1_clk_enable)
	gpio2_ipg_clk_s	ipg_clk_root	CCGR0[CG15] (gpio2_clk_enable)
	gpio3_ipg_clk_s	ipg_clk_root	CCGR2[CG13] (gpio3_clk_enable)
	gpio4_ipg_clk_s	ipg_clk_root	CCGR3[CG6] (gpio4_clk_enable)
	gpio5_ipg_clk_s	ipg_clk_root	CCGR1[CG15] (gpio5_clk_enable)
	gpio6_ipg_clk_s	ahb_clk_root	
	gpio7_ipg_clk_s	ahb_clk_root	
	gpio8_ipg_clk_s	ahb_clk_root	
	gpio9_ipg_clk_s	ahb_clk_root	

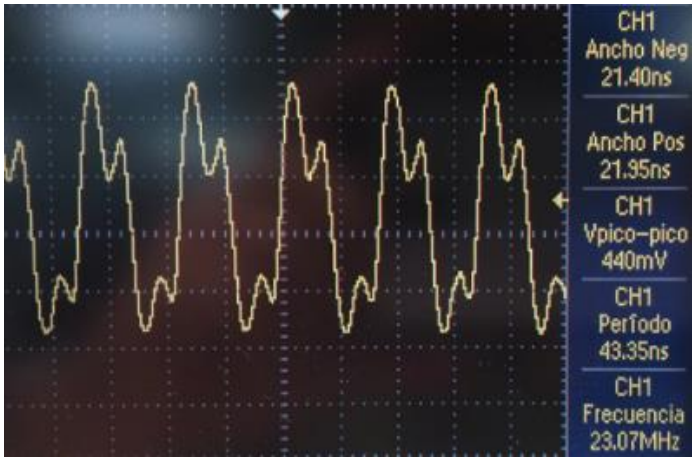
```
71  void GPIO_PinInit(GPIO_Type *base, uint32_t pin, const gpio_pin_config_t *Config)
72  {
73  #if !(defined(FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL) && FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL)
74      /* Enable GPIO clock. */
75      if(base < GPIO6)
76      {
77          CLOCK_EnableClock(s_gpioClock[GPIO_GetInstance(base)]);
78      }
79  #endif /* FSL_SDK_DISABLE_DRIVER_CLOCK_CONTROL */
80
81      /* Register reset to default value */
82      base->IMR &= ~(1U << pin);
83
84      /* Configure GPIO pin direction */
85      if (Config->direction == kGPIO_DigitalInput)
86      {
87          base->GDIR &= ~(1U << pin);
88      }
89      else
90      {
91          GPIO_PinWrite(base, pin, Config->outputLogic);
92          base->GDIR |= (1U << pin);
93      }
94
95      /* Configure GPIO pin interrupt mode */
96      GPIO_SetPinInterruptConfig(base, pin, Config->interruptMode);
97  }
```

首先，我使用了普通的 GPIO 引脚(GPIOI_IO 11)。我将通过直接写入 GPIO_DR 寄存器来翻转引脚电平。请注意，你可以通过评估板上的 J22 的 PIN 3 测量此引脚，从而观测该引脚的性

能。结果如下：

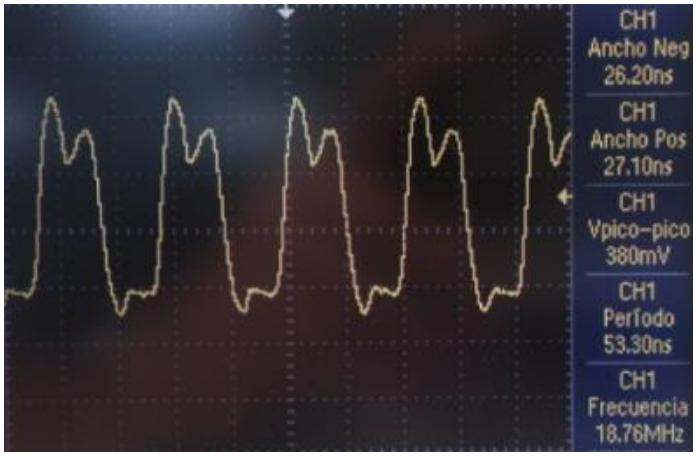


对于普通的 GPIO 引脚，直接写入 GPIO_DR 寄存器的周期达到 160 ns。现在，如果我们切换到快速 GPIO 并使用相同的指令，我们将得到以下结果。



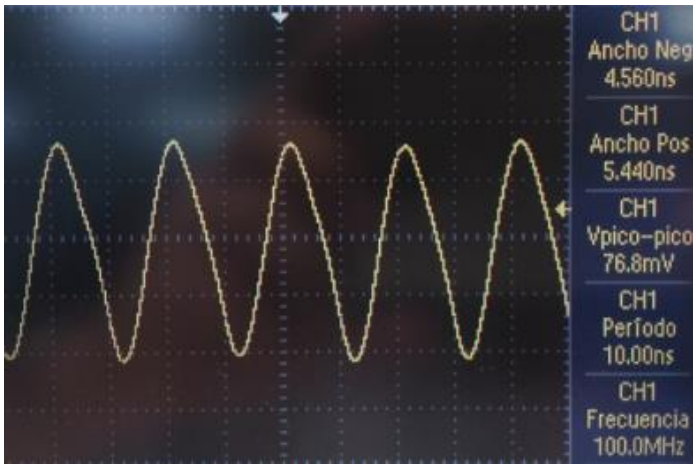
正如你看到的，在使用快速 GPIO 引脚时，信号的周期几乎是使用普通 GPIO 时的三分之一。

现在，RT1060 的 A1 版芯片有一个新的 GPIO 翻转功能。如果我们用新的寄存器 DR_TOGGLE 来翻转引脚而不是用寄存器 GPIO_DR，我们将获得更好的性能，无论是普通 GPIO 还是快速 GPIO。以下是带有 DR_TOGGLE 寄存器的普通 GPIO 的结果。



如你所见，当使用带 DR_TOGGLE 寄存器的普通 GPIO 引脚时，我们的周期大约为 53 ns，而当写入 GPIO_DR 寄存器时，我们得到的是 160 ns。当使用寄存器 DR_TOGGLE 和快速 GPIO

时，我们将获得最佳的引脚性能，结果如下。



感谢 Jorge Antonio Alcala Vazquez 对这篇文档提供的宝贵帮助。

希望这能给你带来帮助！

最诚挚的问候，
Victor。