Notes for STM32 MCUs

Tips, hints, and tricks when working on STM32 ARM Cortex-M MCUs

#notes #pi

Last update: 2021-07-08 22:05:59

Terminal application

Links have review about terminal applications:

- https://oliverbetz.de/pages/PIM/TerminalPrograms
- https://learn.sparkfun.com/tutorials/terminal-basics/all

The good ones are:

- CoolTerm
- YAT
- MobaXterm

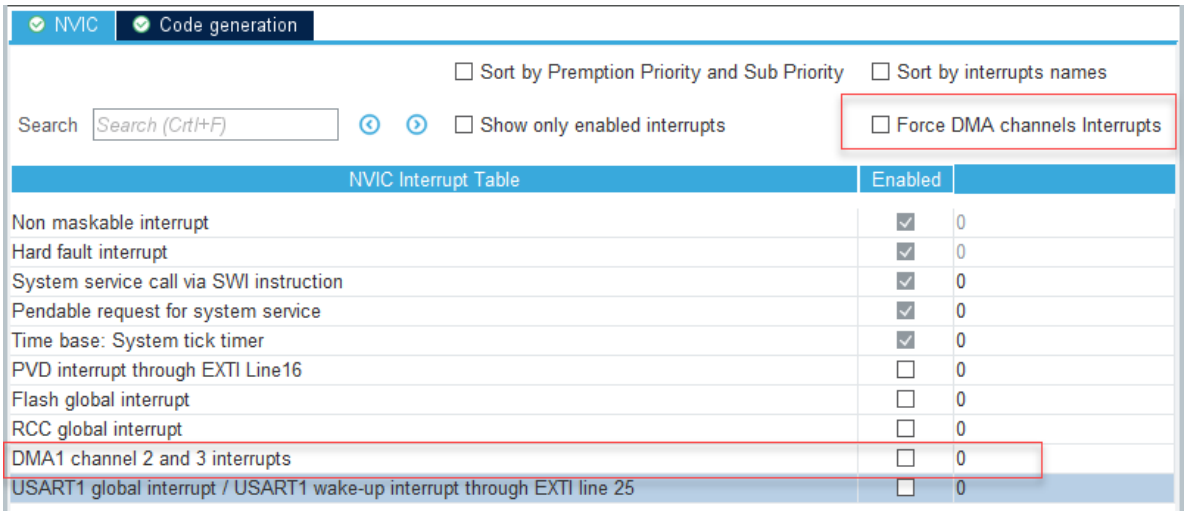
Use float with `printf` and `scanf`

The `newlib-nano` library does not enable `float` support by default. When `float` is used in `printf()`, `scanf()` or in `sprintf()`, it must be explicitly enabled in linker.

To enable, go to **Project** » **Properties** menu, then go to **C/C++ Build** » **Settings** » **Cross ARM C++ Linker** » **Miscellaneous** and check **Use float with nano printf/scanf**.

DMA Polling in a infinite loop

The function `HAL_DMA_PollForTransfer()` works properly only when the DMA interrupts are disabled. It needs to turn off the option *Force DMA Channels Interrupts* to be able to disable DMA interrupts.



Force turning off DMA interrupt

If DMA interrupts are enabled, the interrupt handler `HAL_DMA_IRQHandler()` may clear the interrupt flag and set DMA State to *HAL_DMA_STATE_READY*, which causes the

function `HAL_DMA_PollForTransfer()` runs in a infinite loop if it is called with `Timeout = HAL_MAX_DELAY`:

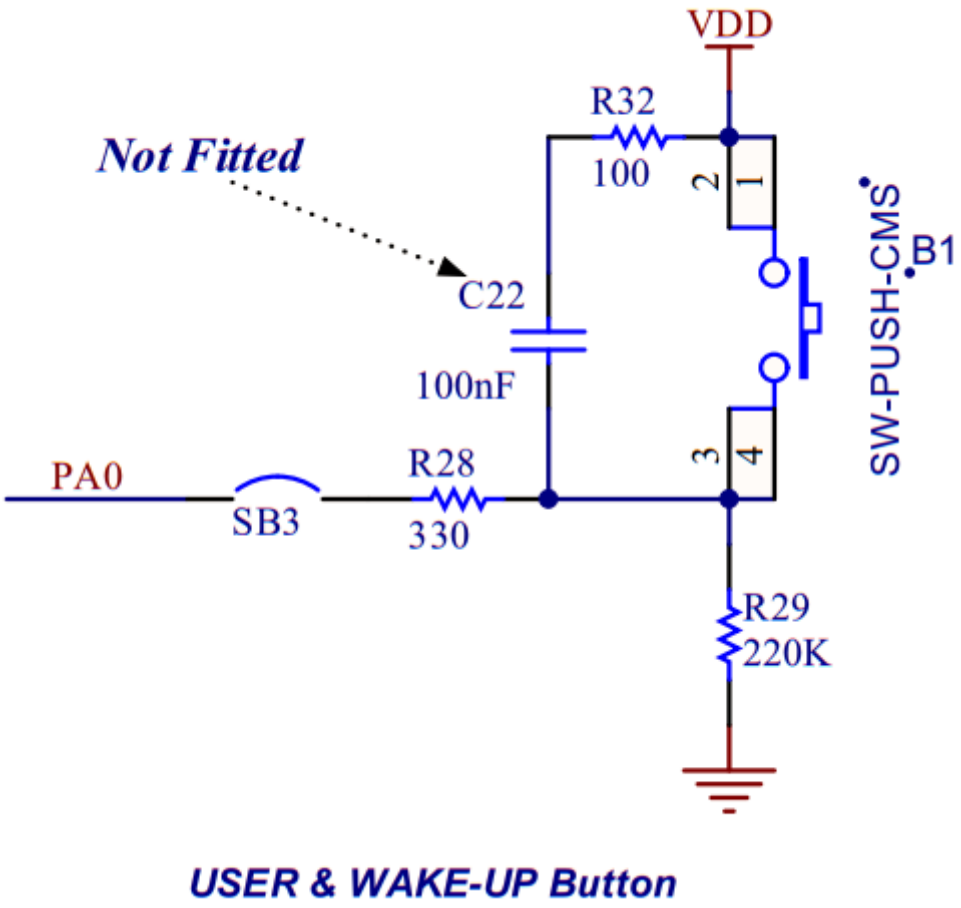
```
HAL_StatusTypeDef HAL_DMA_PollForTransfer(DMA_HandleTypeDef *hdma,
                                          uint32_t CompleteLevel,
                                          uint32_t Timeout)
{
    ...
    /* Get tick */
    tickstart = HAL_GetTick();

    // hdma->DmaBaseAddress->ISR = 0 due to modified hdma->DmaBaseAddress-
>IFCR
    while(RESET == (hdma->DmaBaseAddress->ISR & temp))
    {
        // always enter here
        if(RESET != (hdma->DmaBaseAddress->ISR & (DMA_FLAG_TE1 << hdma-
>ChannelIndex))) {
            // never enter here
        }

        if(Timeout != HAL_MAX_DELAY) {
            // never enter here
        }
    }
}
```

To fix this, just need to check the DMA state inside the while loop. If the state is already *HAL_DMA_STATE_READY*, exit the loop and return *HAL_OK*.

Pulling-up or Pulling-down an input

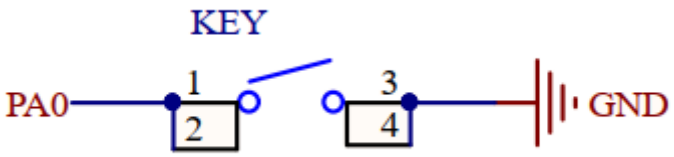
When a logic analyser cannot catch the level of an input pin, it's mostly because of no pull-up or pull-down is applied for that pin.

At the time I wanted to see input bouncing on the push button soldered on my STM32F0 Discovery board, I plugged my Logic Analyzer to the pin **PA0**. The weird thing was that the logic level of **PA0** is always High, even when I pressed on the button. Without logic analyser plugged in, the button works well.



A button using pull-down resistor

It turns out that the pin **PA0** did not have pull-up or pull-down setting in MCU side, while the logic analyser sets its input probe pin with pull-up resistor. Therefore, when probe pin is connected, the logic level is always HIGH. Setting pull-down resistor in in MCU side will solve this problem.



A floating input button

One other case is working with the USER button on Black Pill F411 or Blue Pill F103 boards. The User Key is left floating without any resistor. If in MCU side, no pull-up or pull-down is set, the input is always floating. If I connect a probe pin from logic analyser, the button will work normally. Setting pull-up resistor in in MCU side will solve this problem.

Computer architecture

The most of STM32 MCUs share the same computer architecture except for STM32F0 and STM32L0 that are based on the Cortex-M0/0+ cores. They, in fact, are the only Cortex-M cores based on the *von Neumann architecture*, compared to the other Cortex-M cores that are based on the (modified) *Harvard architecture*[1].

The fundamental distinction between the two architectures is that:

- Cortex-M0/0+ cores access to Flash, SRAM and peripherals using one common bus
- The other Cortex-M cores have:
  - two separated bus lines for the access to the flash (one for the fetch of instructions called instruction bus, or simply I-Bus or even I-Code, and one for the access to const data called data bus, or simply D-Bus or even D-Code)
  - one dedicated line for the access to SRAM and peripherals (also called system bus, or simply S-Bus).

Windows 10 USB to Serial driver

Windows 10 does not support PL2303 USB to Serial, but here is the fix for this problem: https://github.com/johnstevenson/pl2303-win10. This will install an old but compatible driver for EOL PL2303 chips.

---

1. https://en.wikipedia.org/wiki/Modified_Harvard_architecture ↩