Notes for ARM and STM32 programming

#arm #stm32 #notes

Last update: May 3, 2021

Table of Content

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

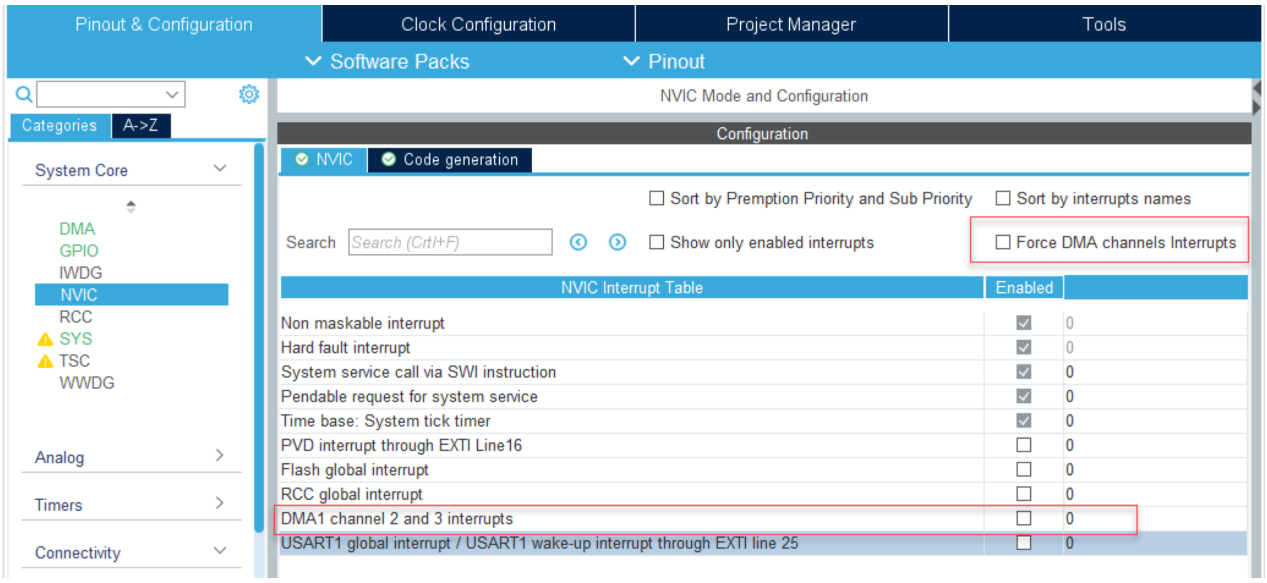
Use float with printf and scanf

`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

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.



☐ Sort by Premption Priority and Sub Priority ☐ Sort by interrupts names

☐ Show only enabled interrupts ☐ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	
Non maskable interrupt	✓	0
Hard fault interrupt	✓	0
System service call via SWI instruction	✓	0
Pendable request for system service	✓	0
Time base: System tick timer	✓	0
PVD interrupt through EXTI Line16	☐	0
Flash global interrupt	☐	0
RCC global interrupt	☐	0
DMA1 channel 2 and 3 interrupts	☐	0
USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25	☐	0

If DMA interrupts are enabled, the interrupt handler `HAL_DMA_IRQHandler()` may clean 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*.

DMA callbacks

UART

> `HAL_UART_TxCpltCallback()` is not called when DMA is in Normal mode, and UART Interrupt is disabled.
>
> > In `UART_DMATransmitCplt()`, if DMA mode is Normal mode, it will enable UART Transmit Complete Interrupt, and transfer that interrupt handling right to the UART interrupt routines

Debug interrupt routine

Even the CPU is stopped by a breakpoint in a Interrupt service routine, the other hardwares on MCU are still running, such as DMA, ADC continuous mode;

This makes debugging ISR more difficult because of changed registers affected by the running peripherals.

Delay in Interrupt handler

By default, peripheral interrupts have the same priority with System Tick interrupt (0), therefore, if there is any `HAL_Delay()` function used in a peripheral interrupt, it will block the System tick from being called, causing tick counter won't be increased, then the delay will be an infinite loop.

Fix git-revision-date-localized

util.py/

```
def get_git_commit_timestamp:
    ...
    try:
        ret = int(commit_timestamp)
    except:
        ts = commit_timestamp.split('\n')
        try:
            ret = ts[len(ts)-1]
        except:
            ret = int(time.time())

    return ret
```