IV. STM32 – Exception and Interrupts

BE2M37MAM – Microprocessors

Stanislav Vítek

Czech Technical University in Prague

Part I

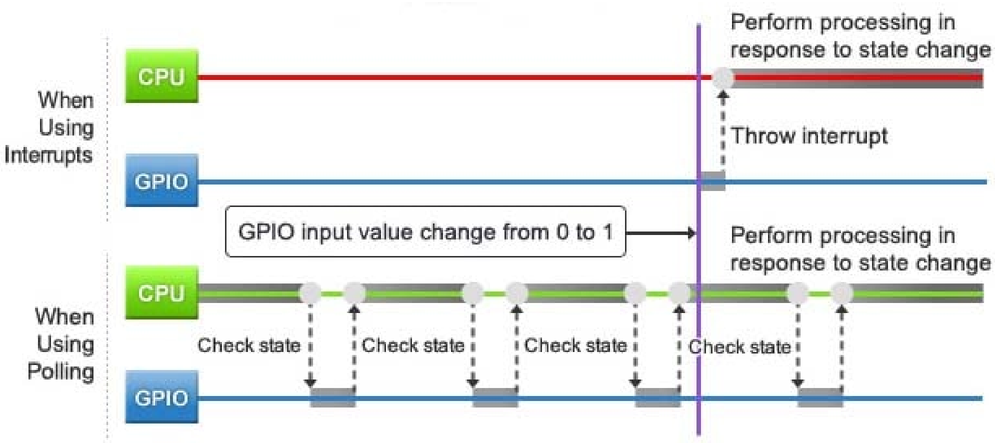
Exceptions

What is an exception?

- A special event that requires the CPU to stop normal program execution and perform some service related to the event.
- Examples of exceptions include
  - I/O completion, timer time-out, end of conversion,
  - illegal opcodes, arithmetic overflow, divide-by-0, etc.
- Functions of exceptions
  - Respond to infrequent but important events
    - Alarm conditions like low battery power
    - Error conditions
  - I/O synchronization
    - Trigger interrupt when signal on a port changes
  - Periodic interrupts
    - Generated by the timer at a regular rate
    - Systick timer can generate interrupt when it hits zero

Reload value + frequency determine interrupt rate

Polling vs. Interrupt



Interrupt Properties

- **Interrupt maskability**
  - Interrupts that can be ignored by the CPU are called maskable interrupts.
  - A maskable interrupt must be enabled before it can interrupt the CPU.
  - An interrupt is enabled by setting an enable bit.
  - Interrupts that can't be ignored by the CPU are called nonmaskable interrupts.
- **Exception priority**
  - Allow multiple pending interrupt requests
  - Resolve the order of service for multiple pending interrupts
- **Interrupt service routine (ISR)**
  - An interrupt handler is a callback subroutine in microcontroller firmware whose execution is triggered by the reception of an interrupt.

Interrupt handlers have a multitude of functions, which vary based on the reason the interrupt was generated.

Interrupt vector

- Starting address of the interrupt handler
- Interrupt vector table
  - table of interrupt vectors that associates an interrupt handler with an interrupt request
- Methods of determining interrupt vectors
  - Predefined locations (Microchip PIC18, 8051 variants)
  - Fetching the vector from a predefined memory location (HCS12, STM32)
  - Executing an interrupt acknowledge cycle to fetch a vector number in order to locate the interrupt vector (68000 and x86 families)

Interrupt Service Cycle

- Saving the program counter value in the stack
- Saving the CPU status (including the CPU status register and some other registers) in the stack
- Identifying the cause of interrupt
- Resolving the starting address of the corresponding interrupt service routine
- Executing the interrupt service routine
- Restoring the CPU status and the program counter from the stack
- Restarting the interrupted program

Part II

Cortex M4 Core Peripherals

Cortex M4 Core Peripherals

System Control Block It provides system implementation information and control. In particular it supports exception configuration, control, and processing.

Nested Vectored Interrupt Controller It supports low latency interrupt configuration, control, and processing.

System timer (SysTick) Use this 24-bit count-down timer as a Real Time Operating System (RTOS) tick timer or as a simple counter.

Memory Protection Unit It improves system reliability by defining the memory attributes for different memory regions.

Floating-point Unit It provides IEEE754-compliant operations on single- precision, 32-bit, floating-point values.

II. Cortex M4 Core Peripherals

System Control Block

SysTick Timer

Nested Vector Interrupt Controller

System Control Block

- Exception enables.
- Setting or clearing exceptions to/from the pending state.
- Exception status (Inactive, Pending, or Active). Inactive is when an exception is neither Pending nor Active.
- Priority setting (for configurable system exceptions)
- The exception number of the currently executing code and highest pending exception.

System Control Block – Registers 1/2

Name	Description	Operation
ACTLR	Auxiliary Control Register	disables certain aspects of functionality within the processor
CPUID	CPUID Base Register	specifies the ID and version numbers, and the implementation details of the processor core
ICSR	Interrupt Control State Register	Used to: * set a pending Non-Maskable Interrupt(NMI) * set or clear a pending PendSV * set or clear a pending SysTick * check for pending exceptions * check the vector number of the highest priority pended exception * check the vector number of the active exception
VTOR	Vector Table Offset Register	indicates the offset of the vector table base address from memory address `0x0000 0000`

System Control Block – Registers 2/2

Name	Description	Operation
AIRCR	Application Interrupt and Reset Control Register	provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system
SCR	System Control Register	speccontrols features of entry to and exit from low power state
CCR	Configuration and Control Register	permanently enables stack alignment and causes unaligned accesses to result in a Hard Fault
SHPRx	System handler priority registers	set the priority level of the exception handlers that have configurable priority

II. Cortex M4 Core Peripherals

System Control Block

SysTick Timer

Nested Vector Interrupt Controller

SysTick Timer

- 24-bit system timer, that counts down from the reload value to zero, reloads the value in the STK_LOAD register on the next clock edge, then counts down on subsequent clocks.
- When the processor is halted for debugging the counter does not decrement.
- Systick can be used to generate an exception (#15).
- It can be used as the basic timer for an operating system, as an alarm timer, for timing measurements, and more.

Address	Name	Type	Description
0xE000E01	STK_CTRL	RW	SysTick control and status register
0xE000E01	STK_LOAD	RW	SysTick reload value register
0xE000E01	STK_VAL	RW	SysTick current value register
0xE000E01	STK_CALIB	RO	SysTick calibration value register

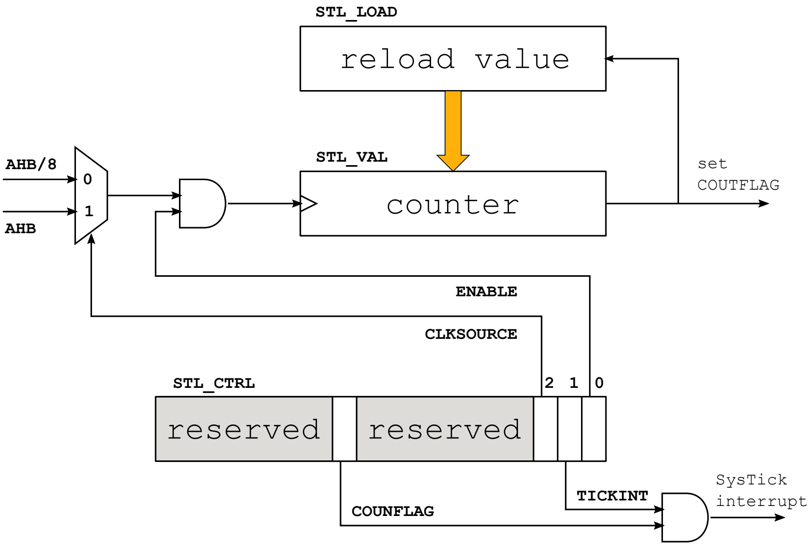
STM32 Cortex-M4 MCUs and MPUs programming manual, page 246

SysTick Control and Status Register

Bits	Name	Type	Reset value	Description
16	COUTFLAG	RO	0	Returns 1 if timer counted to 0 since last time this register was read
2	CLKSOURCE	RW	0	Clock source selection 0: AHB/8 (reset value) 1: Processor clock (AHB)
1	TICKINT	RW	0	SysTick exception request enable
0	ENABLE	RW	0	SysTick timer enable

STM32 Cortex-M4 MCUs and MPUs programming manual, page 247

How SysTick works?



SysTick Exception Configuration

- The SysTick interrupt is an internal Cortex exception and is handled in the system registers.
- Some of the internal exceptions are permanently enabled; these include the reset and NMI interrupts, but also the SysTick timer, so there is no explicit action required to enable the SysTick interrupt within the NVIC.
- To configure the SysTick interrupt we need to set the timer going and enable the interrupt within the peripheral itself:

```
STK_CTRL |= 0;          // reset register, clock source in AHB/8
STK_VAL  |= 0;          // initial value
STL_LOAD |= 2000000;    // reload value
STL_CTRL |= 0x02;       // enable exception
STL_CRTL |= 0x01;       // enable Systick
```

SysTick Interrupt Handler

- To handle SysTick interrupt one can create interrupt service routine (ISR) by declaring function with the same address as an address on declared on 15th position of Interupt Vector Table

```
unsigned long *vtable[] __attribute__((section(".isr_vector"))) = {
    (unsigned long *)SRAM_END,        // 0 initial stack pointer
    (unsigned long *)main,            // 1 main as Reset_Handler
    ...
    (unsigned long *)systick_handler // 15 Systick
};

void systick_handler(void) {
  // code to be executed
}

```

II. Cortex M4 Core Peripherals

System Control Block

SysTick Timer

Nested Vector Interrupt Controller

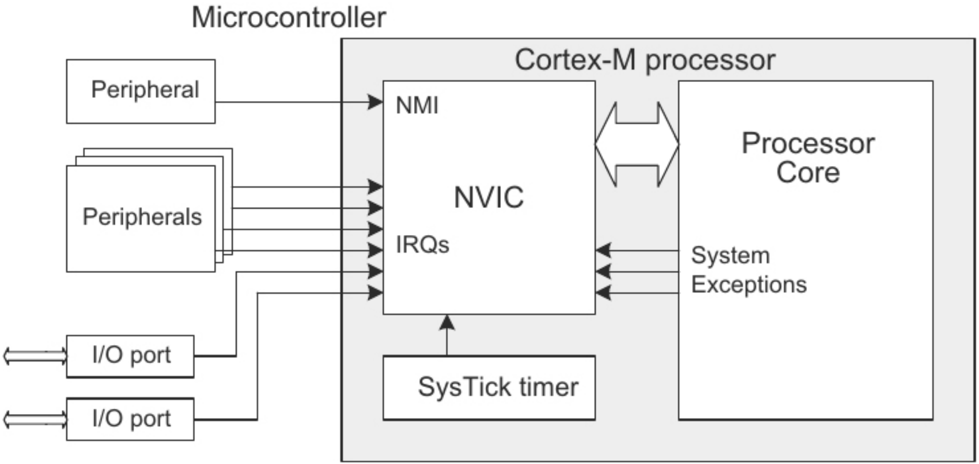
NVIC features

- Up to 240 interrupts (STM32F4 – 52 maskable interrupt channels)

  Not including the 16 interrupt lines of Cortex-M4 with FPU

- 16 programmable priority levels (0-15, 4 bits)
  - A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority
  - Dynamic reprioritization of interrupts
  - Grouping of priority values into group priority and subpriority fields
- Level and pulse detection of interrupt signals
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers
- An external Non-maskable interrupt (NMI)

NVIC in the Cortex-M4 core



Interrupt and exception vectors

no	pos	pr	decription	associated periph.
16	0	7	Window Watchdog interrupt	WWDG
17	1	8	PVD through EXTI line detection interrupt	EXTI16/PVD
18	2	9	Tamper and TimeStamp interrupts through the EXTI line	EXTI21/TAMP_STAMP
19	3	10	RTC Wakeup interrupt through the EXTI line	EXTI22/RTC_WKUP
20	4	11	Flash global interrupt	FLASH
21	5	12	RCC global interrupt	RCC
22	6	13	EXTI Line0 interrupt	EXTI0
23	7	14	EXTI Line1 interrupt	EXTI1

STM32F401xD/E advanced ARM-based 32-bit MCUs, reference manual, page 202

Interupt Latency

- The NVIC is designed for fast and efficient interrupt handling
  - on a Cortex-M4 you will reach the first line of C code in your interrupt routine after 12 cycles for a zero wait state memory system.
- This interrupt latency is fully deterministic
  - from any point in the background (non-interrupt) code you will enter the interrupt with the same latency.
- Multi-cycle instructions can be halted with no overhead and then resumed once the interrupt has finished.

Exception States

- Inactive:
  - The exception is not active and not pending.
- Pending:
  - The exception is waiting to be serviced by the processor.
  - An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
- Active:
  - An exception that is being serviced by the processor but has not completed.
  - An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state.
- Active and pending
  - The exception is being serviced by the processor and there is a pending exception from the same source.

NVIC registers

address	name	decription
0xE000E100 – 0xE000E11F	NVIC_ISER0 – NVIC_ISER7	Int. set-enable registers, p. 210
0XE000E180 – 0xE000E19F	NVIC_ICER0 – NVIC_ICER7	Int. clear-enable registers, p.211
0XE000E200 – 0xE000E21F	NVIC_ISPR0- NVIC_ISPR7	Int. set pending registers, p. 212
0XE000E280 – 0xE000E29F	NVIC_ICPR0- NVIC_ICPR7	Int. clear-pending registers, p. 213
0xE000E300 – 0xE000E31F	NVIC_IABR0-NVIC_IABR7	Int. active bit register, p. 214
0xE000E400 – 0xE000E4EF	NVIC_IPR0- NVIC_IPR59	Int. priority registers, p. 215
0xE000EF0	STIR	Software trigger int. register, p. 216

Write to the STIR to generate a Software Generated Interrupt (SGI). The value to be written is the Interrupt ID of the required SGI, in the range 0-239. For example, a value of 3 specifies interrupt IRQ3.

STM32 Cortex-M4 MCUs and MPUs programming manual, page 208