NXP Semiconductors
Application Note

Document Number: AN12078
Rev. 1, 04/2018

Measuring Interrupt Latency

Contents

1. Introduction

The term interrupt latency refers to the delay between the start of an Interrupt Request (IRQ) and the start of the respective Interrupt Service Routine (ISR). The interrupt latency is expressed in core clock cycles. There is another exact definition-the number of clock cycles from the assertion of the interrupt request to the first ISR instruction executed, as shown in Figure 1.

When the clock frequency is known, the interrupt latency is also expressed in terms of time (us or ns).

In some cases, there are broad sense definitions of the term to include more delay components. For example:

- The start point definition may be from the generation of the interrupt signal, rather than the generation of the interrupt request on the core. It may take several cycles for the interrupt signal source to trigger the interrupt request.
- The response point definition may be another event triggered by the ISR rather than the first instruction in the ISR, such as the response of an RTOS task woken up by the interrupt.

The interrupt latency is expected to be a known value and as short as possible, especially for applications with real-time requirements. However, it is always affected by many factors.

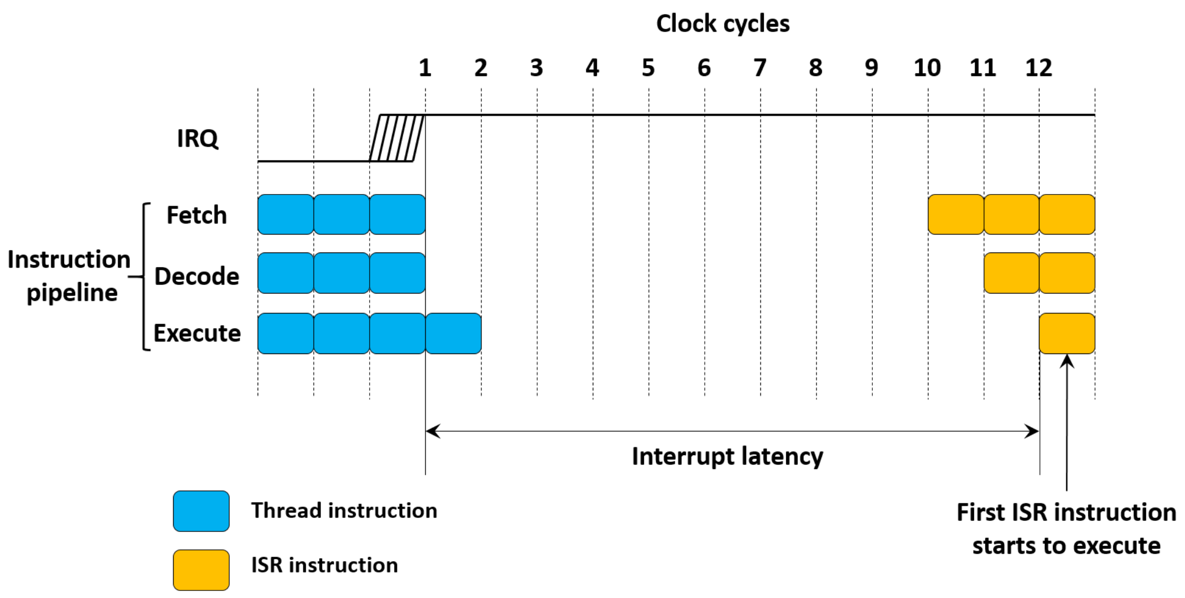


Figure 1. Definition of interrupt latency

Normally, we can get the interrupt latency of a processor (core/CPU) from the specification which is defined by the designer, based on the processor architecture. For a broad-sense definition of the interrupt latency of an actual MCU or MPU SoC platform, more additional influence factors make it difficult to provide an exact common value.

On one hand, you can take all the influence factors into the account and calculate a theoretical value. On the other hand, you can observe the relevant signals on the pads of the platform with an oscilloscope, and ascertain the interrupt latency based on the measured data.

This application note describes how to measure the interrupt latency of the i.MX RT1050 device.

2. Cause of interrupt latency

The interrupt latency is usually affected by a lot of factors.

For a narrow sense of the interrupt latency, there are these typical influence factors:

- For most processor architectures, the processor usually completes the current instruction, which may be a multi-cycle instruction.
- To save the current scene to restore the states when returning from the ISR, the processor pushes various necessary core registers (usually the program counter, flag registers, linker register, and so on) to the stack.
- Some processor architectures need additional software statements to select the right ISR.
- Time to fetch and decode the ISR instructions to fill the pipeline.
- Most memory systems that store the code (such as flash) usually have wait states because the memory system clock frequency is usually much slower than the CPU clock.
- The interrupt may be preempted by other higher-priority interrupts anytime, including before the first ISR instruction is executed.

For a broad-sense definition of the interrupt latency, there are other additional factors:

- The interrupt request signal must be synchronized to the CPU clock timing, which may take several cycles for the interrupt signal source to trigger the interrupt request.
- If the interrupt request signal comes from outside of the processor device, the signal must be firstly synchronized to the bus/peripheral clock.
- The RTOS may temporarily disable the interrupts when accessing critical resources. The latency is longer if the interrupt request asserts during the interrupt is disabled.
- The response point may be defined as another event triggered by the ISR rather than the first instruction in the ISR, such as the response of an RTOS task which is blocked before and woken up by the interrupt. It may take many cycles for the software to complete the process before the defined response point.

3. i.MX RT1050 introduction

i.MX RT1050 is a processor family that features NXP’s advanced implementation of the Arm® Cortex®-M7 core, which operates at speeds of up to 600 MHz to provide high CPU performance and the best real-time response.

i.MX RT1050 has 512-KB on-chip RAM, which can be flexibly configured as Tightly-Coupled Memory (TCM) or general-purpose On-Chip RAM (OCRAM). It also provides interfaces to connect various external memories and a wide range of other communication interfaces, such as USB OTG, Ethernet, UART, I2C, SPI, and CAN. It has rich audio and video features including the LCD display, basic 2D graphics, camera interface, SPDIF, and I2S audio interface. Other notable features include various modules for security, motor control, analog signal processing, and power management.

Reading the code/data from the RAM is optional to having no wait state (TCM only), or having an additional clock cycle that can avoid a potential timing problem caused by a relatively long memory access time at a high frequency.

4. Arm Cortex-M interrupt latency

The Cortex-M processor closely integrates a configurable Nested Vectored Interrupt Controller (NVIC), providing a fast execution of ISRs. The interrupt handlers do not require wrapping in code that removes any code overheads from the ISRs. The tail-chain, late-arrival, and pop-preemption mechanisms also significantly reduce the overhead when switching from one ISR to another. The Cortex-M processor latencies are provided in Table 1.

Table 1. Cortex-M interrupt latency

Processors	Cycles with zero wait states
Cortex-M0	16
Cortex-M0+	15
Cortex-M3/M4	12
Cortex-M7	10~12

The interrupt latency listed in Table 1 belongs to the narrow-sense definition and has the assumption that the memory system has zero wait states.

5. Measuring interrupt latency

This section describes how to measure the interrupt latency of the i.MX RT1050 MPU. There are three types of latency that are measured:

- The latency from the timer interrupt to the ISR execution.
- The latency from the timer interrupt to the RTOS highest-priority task, which is woken up by the interrupt and starts running.
- The latency of the external GPIO interrupt response.

During the measurements, the Cortex-M7 core operates at 600 MHz, and the IPG clock runs at 150 MHz. The code runs from the TCM with no wait states.

5.1. Timer interrupt latency

This type of latency conforms to the narrow-sense definition (from the start of the IRQ to the start of the ISR), which is affected little by various hardware and software factors.

The GPT1 (General Purpose Timer 1) on-chip timer module is configured to generate compare events on the compare channel1. A compare event simultaneously generates an output signal on the respective pad (GPT1_COMPARE1) and an interrupt request on the NVIC.

At the beginning of the GPT1 ISR, a GPIO pin (GPIO2-23) is toggled twice (firstly outputting high level, then outputting low level) to indicate the interrupt event. The GPT1 ISR is shown in Example 1, and the assembly code to toggle the GPIO pin is shown in Example 2.

Example 1. GPT1 ISR for timer interrupt latency measurement

```
void GPT_IRQHandler(void)
{
    GPIO2->DR = 1u << 23;
    GPIO2->DR = 0u;
    GPT_ClearStatusFlags(GPT_BASE, GPT_CHANNEL_FLAG);
}
```

Example 2. Instructions to toggle GPIO pin

```
    LDR.N    R0, [PC, #0x78]        ; GPIO2_DR
    MOV.W    R1, #8388608           ; 0x800000
    STR      R1, [R0]
    MOVS     R1, #0
    STR      R1, [R0]
```

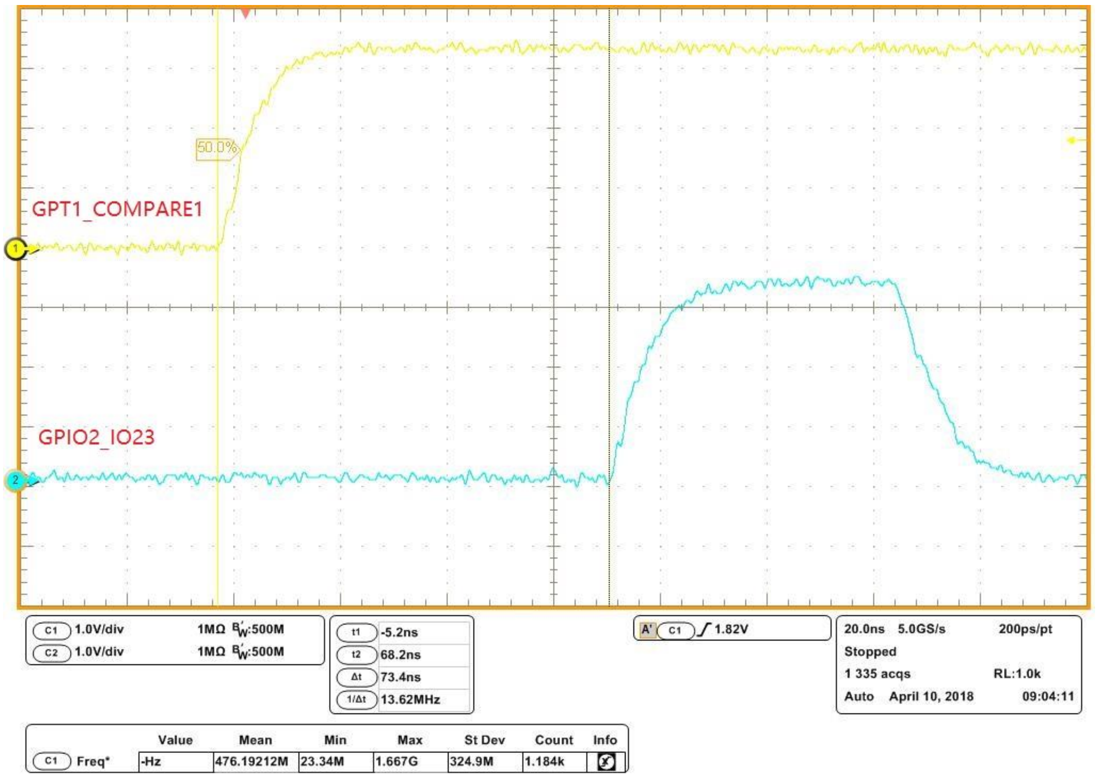
As per Example 2, there are five instructions in total to toggle the GPIO twice:

- The first instruction (LDR) loads the GPIO data register’s address to R0, which takes two cycles.
- The second instruction (MOV) fills R1 with the immediate data 8388608(1<<23), which takes one cycle.
- The third instruction (STR) updates the GPIO data register to generate the rising edge.
- The fourth instruction (MOVS) clears R1, which takes one cycle.

- The fifth instruction (STR) updates the GPIO data register to generate the falling edge.

Due to the low speed of the peripheral clock when compared to the core, there are wait states when the core accesses the GPIO data register. Therefore, it is not known how much time it takes to complete the STR instruction in a glance like other instructions. You may observe the signals and determine the amount of clock cycles cost by the STR with a little trick. And that’s why the GPIO pin is toggled twice.

An oscilloscope is used to observe the GPT1_COMPARE1 and GPIO2_IO23 pads, as shown in Figure 2



```
C1 1.0V/div   1MΩ BW:500M      t1    -5.2ns       A' C1 ∫ 1.82V      20.0ns  5.0GS/s   200ps/pt
C2 1.0V/div   1MΩ BW:500M      t2    68.2ns                          Stopped
                               Δt    73.4ns                          1 335 acqs        RL:1.0k
                               1/Δt  13.62MHz                        Auto  April 10, 2018  09:04:11
```

	Value	Mean	Min	Max	St Dev	Count	Info
C1 Freq*	-Hz	476.19212M	23.34M	1.667G	324.9M	1.184k	

Figure 2. Timer interrupt waveforms

There is a time delay between the rising edge of the timer compare output and the rising edge of the indicator GPIO output. However, this is not the definitive interrupt latency. It also includes some clock cycles taken by other instructions, as shown in Example 2. The signals’ edges are quite gentle, which results from the capacitive components connected to the pads on the PCB.

Based on the instructions shown in Example 2 and the wave form shown in Figure 2, you can draw the timing sequence diagram of the interrupt process, as shown in Figure 3.

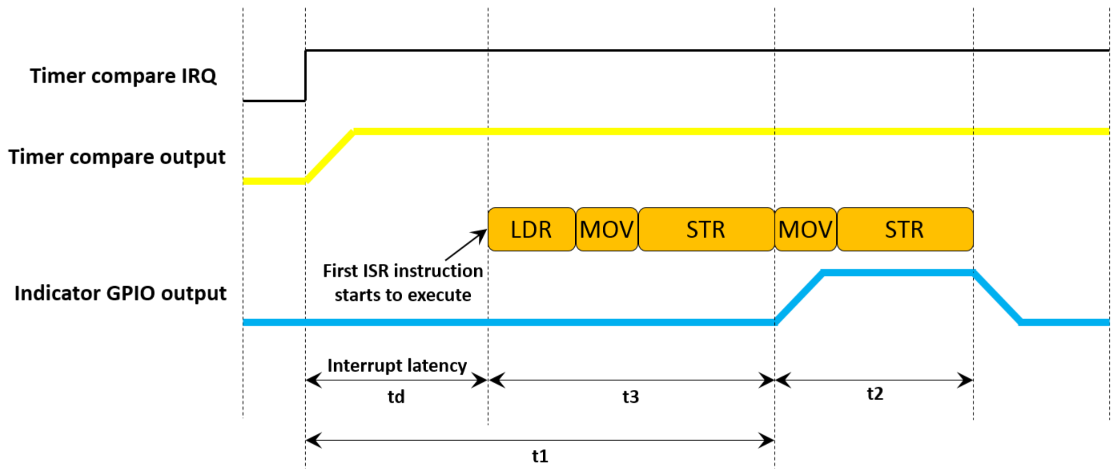


Figure 3. Timer interrupt timing sequence

LDR is the first instruction in the GPT1 ISR. The interrupt latency is called “td” in Figure 3. The time interval between the starts of the two rising edges is called “t1”. The time interval between the start of the indicator GPIO rising edge and the start of the falling edge is called “t2”. The time interval from the start of the first instruction (LDR) to the indicator GPIO rising edge is called “t3”.

In the oscilloscope waveform, t1 = 73.4 ns and t2 = 53.4ns. The core clock frequency is 600 MHz, so it completes 0.6 cycles per 1 ns. Therefore:

- t1 = 73.4 ns * 0.6 cycles/ns = 44 cycles
- t2 = 53.4 ns * 0.6 cycles/ns = 32 cycles

The LDR instruction takes two cycles, therefore:

- t3 = 2 cycles + t2 = 2 cycles + 32 cycles = 34 cycles

The interrupt latency is:

- td = t1 – t3 = 44 cycles – 34 cycles = 10 cycles

which is identical to the Cortex-M7 theoretical value.

5.2. Timer interrupt waking an RTOS task latency

The measurement of this type of latency is used to ascertain the response time from a timer interrupt request to a blocked RTOS task woken up by the interrupt. The result can be taken as a reference for similar applications.

The FreeRTOS version 9.0.0 real-time operating system is used in the project. In the highest-priority task, a binary semaphore is checked and taken (if available). A GPIO (GPIO2-23) pin is toggled twice. If the semaphore is not available, it waits until the semaphore is available. The task function is shown in Example 3.

Example 3. GPT ISR and RTOS task function

```
static void DemoTask(void *pvParameters)
{
    for(;;)
    {
        xSemaphoreTake(xGpTSemaphore, portMAX_DELAY);
        GPIO2->DR = 1u << 23
        GPIO2->DR = 0u;
    }
}
```

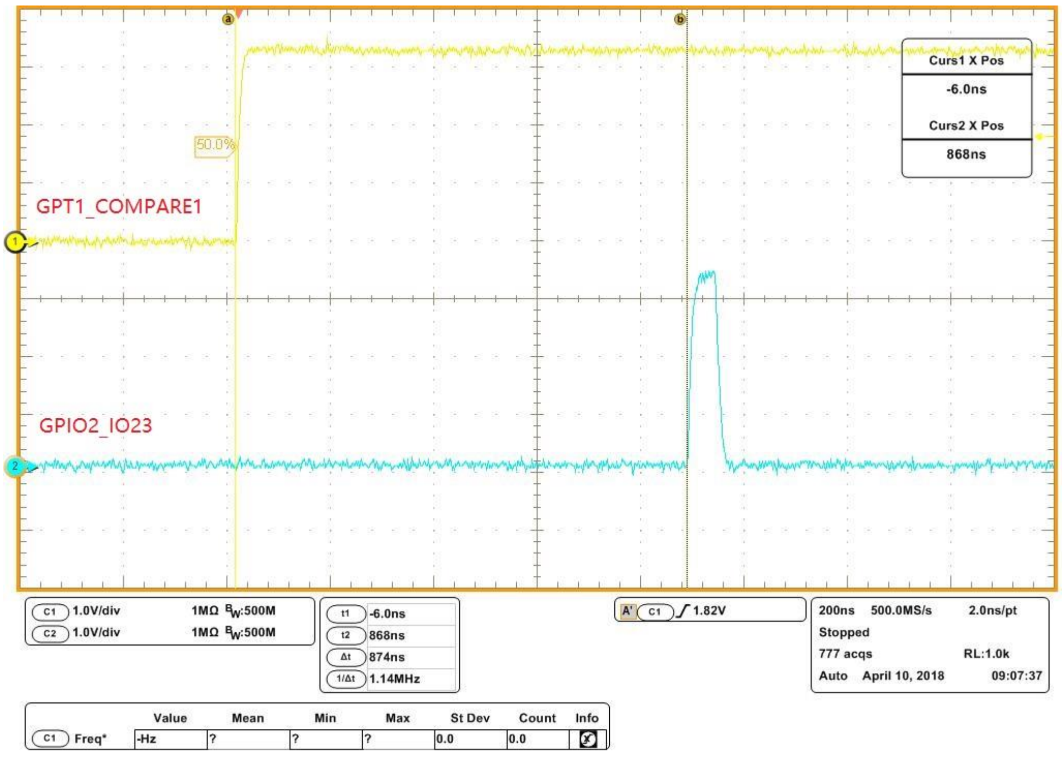
The assembly instructions that toggle the indicator GPIO are the same as those in Example 2.

The GPT1 timer module is configured to generate compare events on the compare channel1. The compare event simultaneously generates the output signal on the GPT1_COMPARE1 pad and the interrupt request on the NVIC. In the GPT1 ISR, the semaphore that the “DemoTask” requires is given. The GPT1 ISR is shown in Example 4.

Example 4. GPT1 ISR for timer interrupt waking an RTOS task latency measurement

```
void GPT_IRQHandler(void)
{
    GPT_ClearStatusFlags(GPT_BASE, GPT_CHANNEL_FLAG);
    xSemaphoreGiveFromISR(xGpTSemaphore, &xHigherPriorityTaskWoken);
    portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
}
```

The waveforms of the GPT1 compare output and the indicator GPIO output are captured by observing the GPT1_COMPARE1 and GPIO2_IO23 pads, as shown in Figure 4.



	Value	Mean	Min	Max	St Dev	Count	Info
C1 Freq*	-Hz	?	?	?	0.0	0.0	

Figure 4. Timer interrupt waking RTOS task waveforms

The timing sequence diagram of the interrupt process is shown in Figure 5.

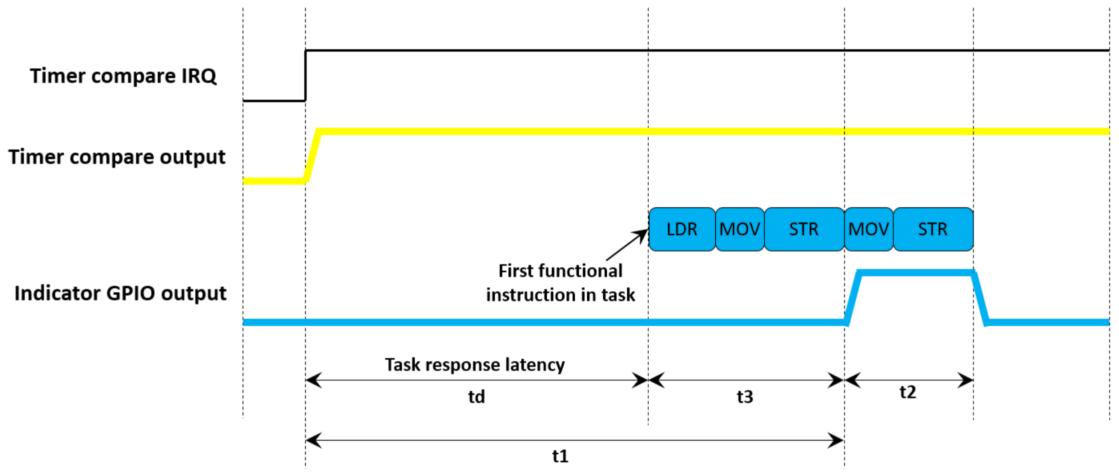


Figure 5. Timer interrupt waking RTOS task timing sequence

In the waveform, the measured values are t1 = 874 ns and t2 = 53.4 ns. Therefore:

- t1 = 874 ns * 0.6 cycles/ns = 524 cycles
- t2 = 53.4 ns * 0.6 cycles/ns = 32 cycles
- t3 = 2 cycles + t2 = 2 cycles + 32 cycles = 34 cycles

The latency is:

- td = t1 – t3 = 524 cycles – 34 cycles = 490 cycles

The RTOS takes quite a lot of cycles to re-schedule the tasks.

5.3. GPIO interrupt latency

The measurement of this type of latency is used to determine the response time of an input GPIO interrupt.

A GPIO button (GPIO5-00) is configured to generate an GPIO interrupt request on the falling edge of the input signal. Another GPIO pin (GPIO2-23) is toggled twice in the GPIO5 ISR, as shown in Example 5.

Example 5. GPIO5 ISR for GPIO interrupt latency measurement

```
void GPIO5_Combined_0_15_IRQHandler (void)
{
    GPIO2->DR = 1u << 23
    GPIO2->DR = 0u;
    GPIO_ClearPinsInterruptFlags(SW_GPIO_BASE, 1u << SW_GPIO_PIN);
}
```

The assembly instructions that toggle the indicator GPIO are the same as those in Example 2.

The waveforms of the button GPIO input and the indicator GPIO output signals are captured by observing the GPIO5_IO00 and GPIO2_IO23 pads, as shown in Figure 6.

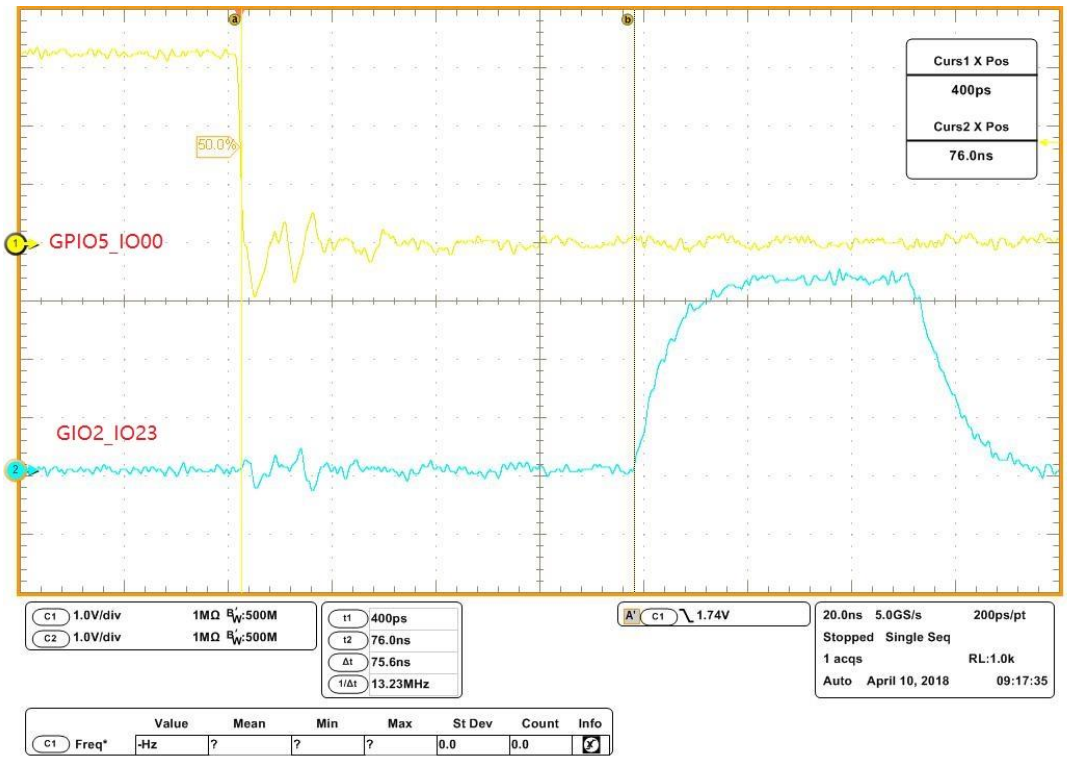


Figure 6. GPIO interrupt waveforms

The timing sequence of the interrupt process is shown in Figure 7.

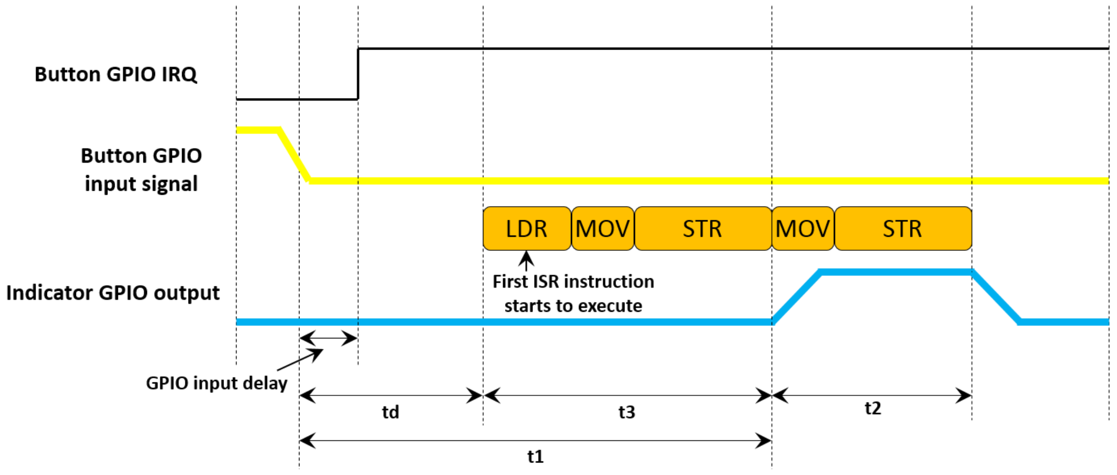


Figure 7. GPIO interrupt timing sequence diagram

The time interval from the falling edges of the button GPIO input signal to the start of the rising edge indicator GPIO rising edge is called “t1”. The time interval from the indicator GPIO rising edge to the falling edge is called “t2”.

NOTE

There is a delay from the falling edge of the input GPIO pin signal to the IRQ.

Typically, the input signal trigger point is marked at the point of 0.3 VDD. It is 0.99 V when VDD is 3.3 V.

In the waveform, t1 = 75.6 ns and t2 = 53.4 ns. Therefore:

- t1 = 75.6 ns * 0.6 cycles/ns = 45 cycles
- t2 = 53.4 ns * 0.6 cycles/ns = 32 cycles
- t3 = 2 cycles + t2 = 2 cycles + 32 cycles = 34 cycles

The latency is:

- td = t1 – t3 = 45 cycles – 34 cycles = 11 cycles

When compared to the timer interrupt latency cycles, it takes three cycles for the input signal to synchronize.

6. Conclusion

The interrupt latency is one of the key characteristics of a processor. It is usually affected by many factors. To be aware of the influence factors and the specific latency cycles of a processor is crucial for many applications with real-time requirements. This application note describes the method to measure three types of interrupt latency on the i.MX RT1050 device. The measurement method can be taken as a reference to determine the interrupt latency of your application's processor.

The results indicate that i.MX RT1050 has ideal short interrupt latency, as shown in Table 2.

Table 2. RT1050 interrupt latency summary

Latency type	Latency in clock cycles	Latency in ns
Timer interrupt latency	10	16.67
Timer interrupt waking an RTOS task latency	490	816.67
GPIO interrupt latency	11	18.33

The interrupt latency of i.MX RT1020 device operating at 500 MHz has also been measured using the same method. The result is shown in Table 3.

Table 3. RT1020 interrupt latency summary

Latency type	Latency in clock cycles	Latency in ns
Timer interrupt latency	10	20
Timer interrupt waking an RTOS task latency	462	924
GPIO interrupt latency	11	22

7. Revision history

Table 4. Revision history

Revision number	Date	Substantive changes
0	10/2017	• Initial release
1	04/2018	• Set i.MX RT1050 IPG clock frequency to 150 MHz, and updated the results. It was set to 300 MHz before which is not allowed. • Added i.MX RT1020 results.

How to Reach Us:

Home Page:
www.nxp.com

Web Support:
nxp.com/support

.

Document Number: AN12078
Rev. 1
04/2018