## 1. Background:

Cortex-M4 for i.MX6SoloX that is new to i.MX6SX customers. They concerns GPIO ISRs response time are not real time or delayed some times while Android/Linux is running on Cortex-A9 in i.MX6SoloX.

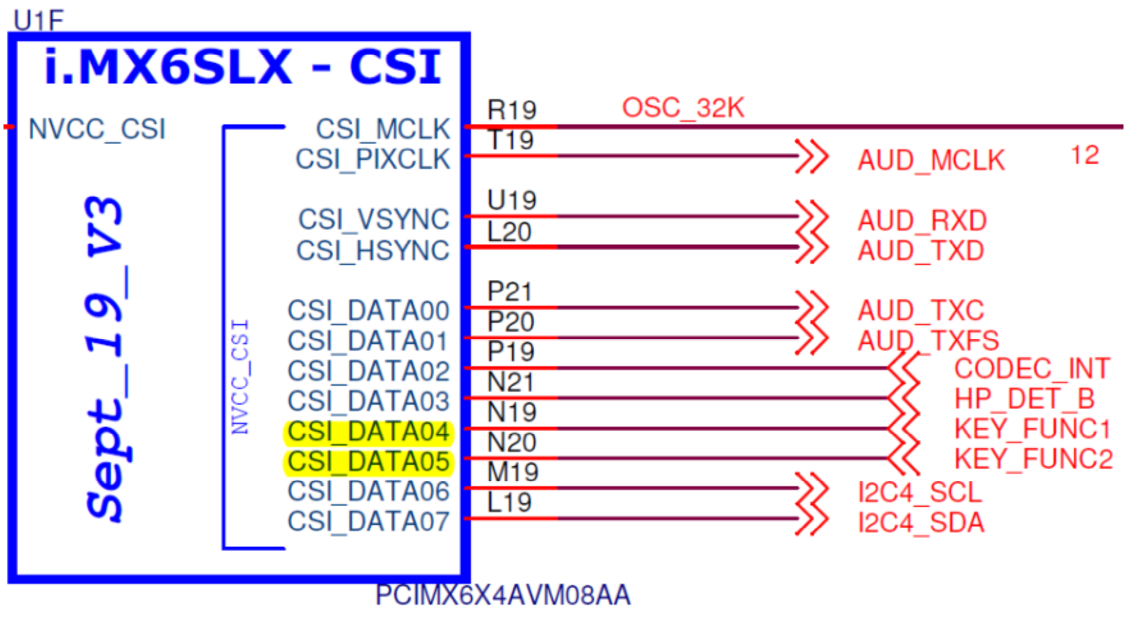
The Interrupt Response Time (IRT) is defined as the time it takes from the assertion of the interrupt request signal by the interrupting device, until the start of the execution of the first useful instruction in the Interrupt Service Routine (ISR). The first useful instruction is the first instruction that is directly related to the actions requested by the interrupting device and not related to interrupt processing.

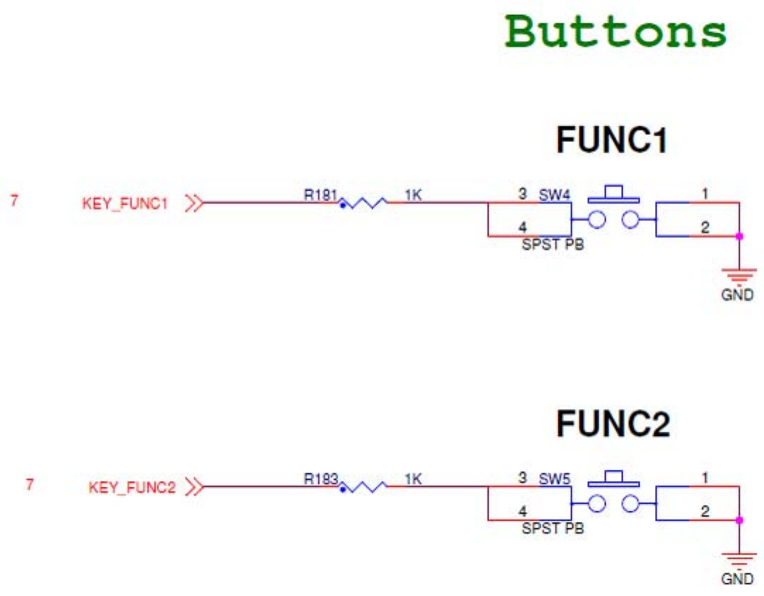
Here are my test steps to get the actual value of Cortex-M4 IRT on i.MX6SX SabreSD board.

## 2. Installation Instructions:

a. H/W preparation

Target H/W: i.MX6SoloX SabreSD
CSI_DATA04: ISR input source (map to SW4 in H/W board)
CSI_DATA05: GPIO output as trigger pin for an oscilloscope

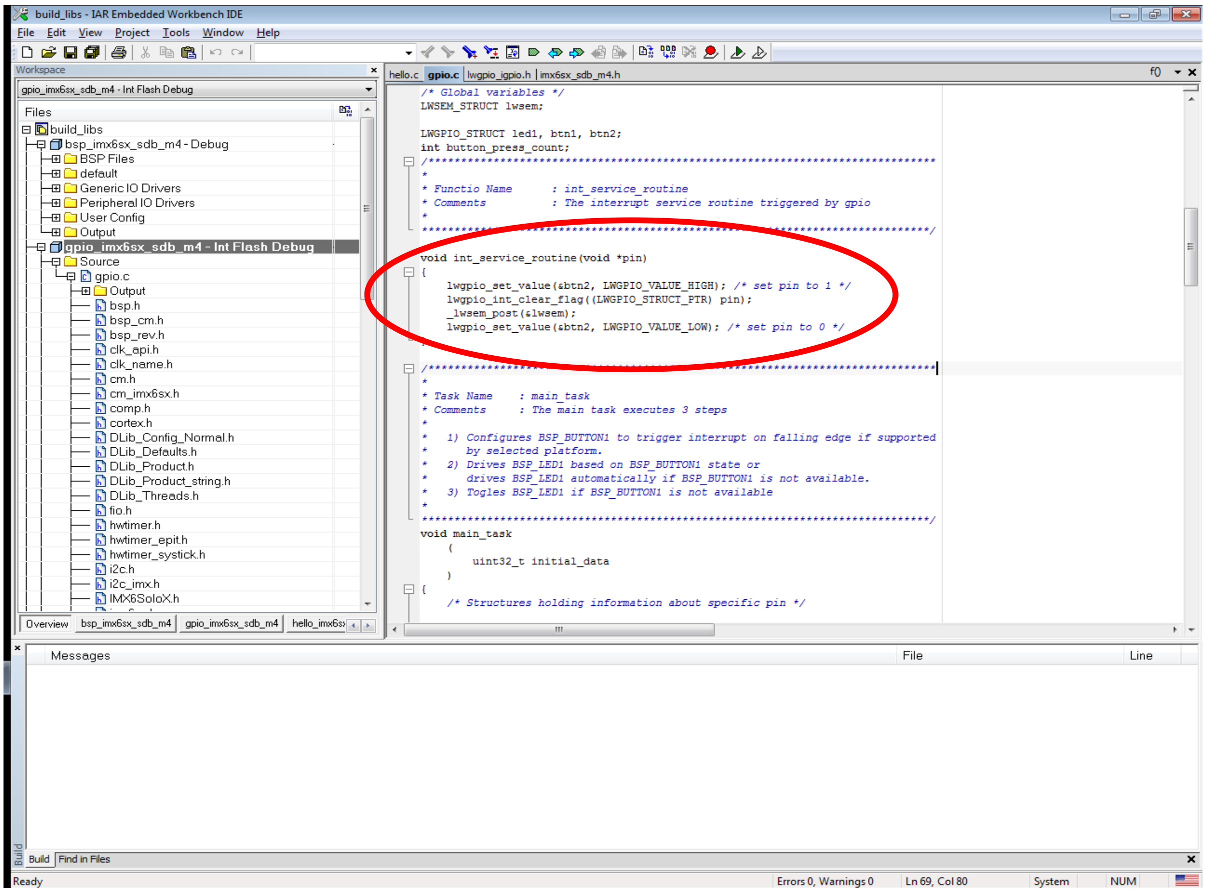




b. S/W preparation

Target S/W: gpio example code on MQX 4.1.0 i.MX 6SoloX Beta BSP, the example code consist of just one task (main_task) and the interrupt service routine triggered by the gpio pin (int_service_routine), under ..\Freescale_MQX_4_1_IMX6SX\mqx\examples\gpio. Modify int_service_routine() in gpio.c to output a high-low pulse. Please refer to the attached source and binary.

```
void int_service_routine(void *pin)
{
   lwgpio_set_value(&btn2, LWGPIO_VALUE_HIGH); /* set pin to 1 */
   lwgpio_int_clear_flag((LWGPIO_STRUCT_PTR) pin);
   _lwsem_post(&lwsem);
   lwgpio_set_value(&btn2, LWGPIO_VALUE_LOW); /* set pin to 0 */
}
```

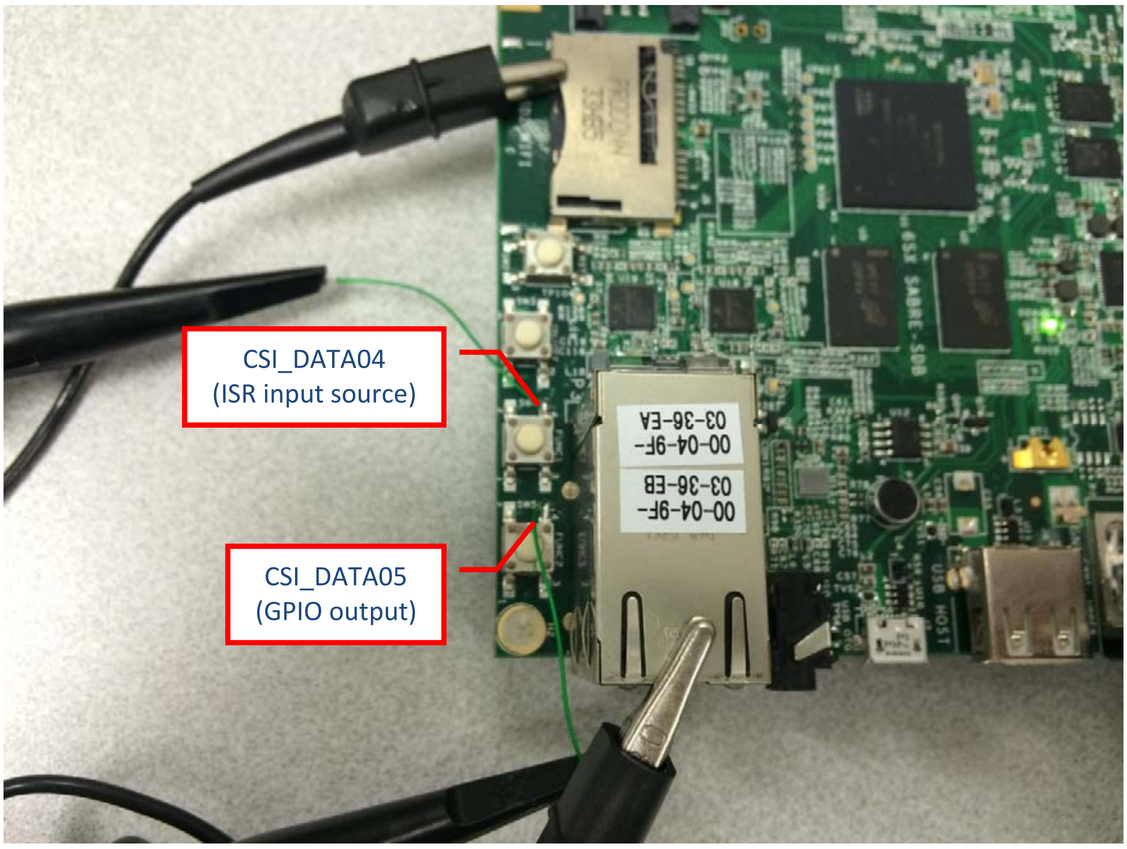


```
/* Global variables */
LWSEM_STRUCT lwsem;

LWGPIO_STRUCT led1, btn1, btn2;
int button_press_count;
/*****************************************************************************
*
* Functio Name      : int_service_routine
* Comments          : The interrupt service routine triggered by gpio
*
*****************************************************************************/

void int_service_routine(void *pin)
{
    lwgpio_set_value(&btn2, LWGPIO_VALUE_HIGH); /* set pin to 1 */
    lwgpio_int_clear_flag((LWGPIO_STRUCT_PTR) pin);
    _lwsem_post(&lwsem);
    lwgpio_set_value(&btn2, LWGPIO_VALUE_LOW); /* set pin to 0 */
}

/*****************************************************************************
*
* Task Name     : main_task
* Comments      : The main task executes 3 steps
*
*   1) Configures BSP_BUTTON1 to trigger interrupt on falling edge if supported
*      by selected platform.
*   2) Drives BSP_LED1 based on BSP_BUTTON1 state or
*      drives BSP_LED1 automatically if BSP_BUTTON1 is not available.
*   3) Togles BSP_LED1 if BSP_BUTTON1 is not available
*
*****************************************************************************/
void main_task
    (
        uint32_t initial_data
    )
{
    /* Structures holding information about specific pin */
```



### 3. Test results:

According to my test results below, I got 15.4us average interrupt latency is the time gap between external H/W GPIO interrupt pin and the start of the execution of the first useful instruction in the Interrupt Service Routine. 19.2us average interrupt latency is the time gap between external H/W GPIO interrupt pin and the end of useful instruction in the Interrupt Service Routine. The response time is no difference if Cortex-A9 with Android/Linux BSP running or not.

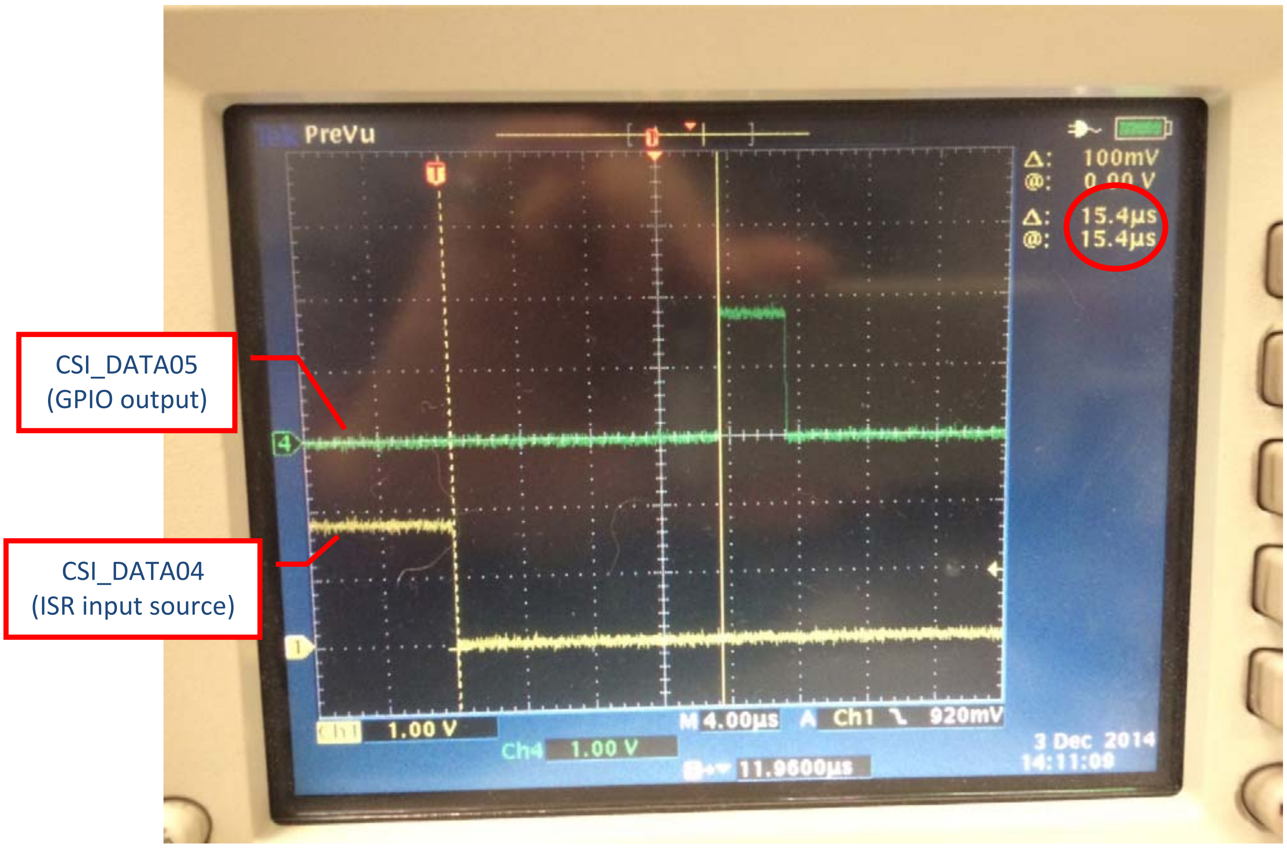


Figure 1. 15.4us is the time gap between external H/W GPIO interrupt pin and the start of the execution of the first useful instruction in ISR.

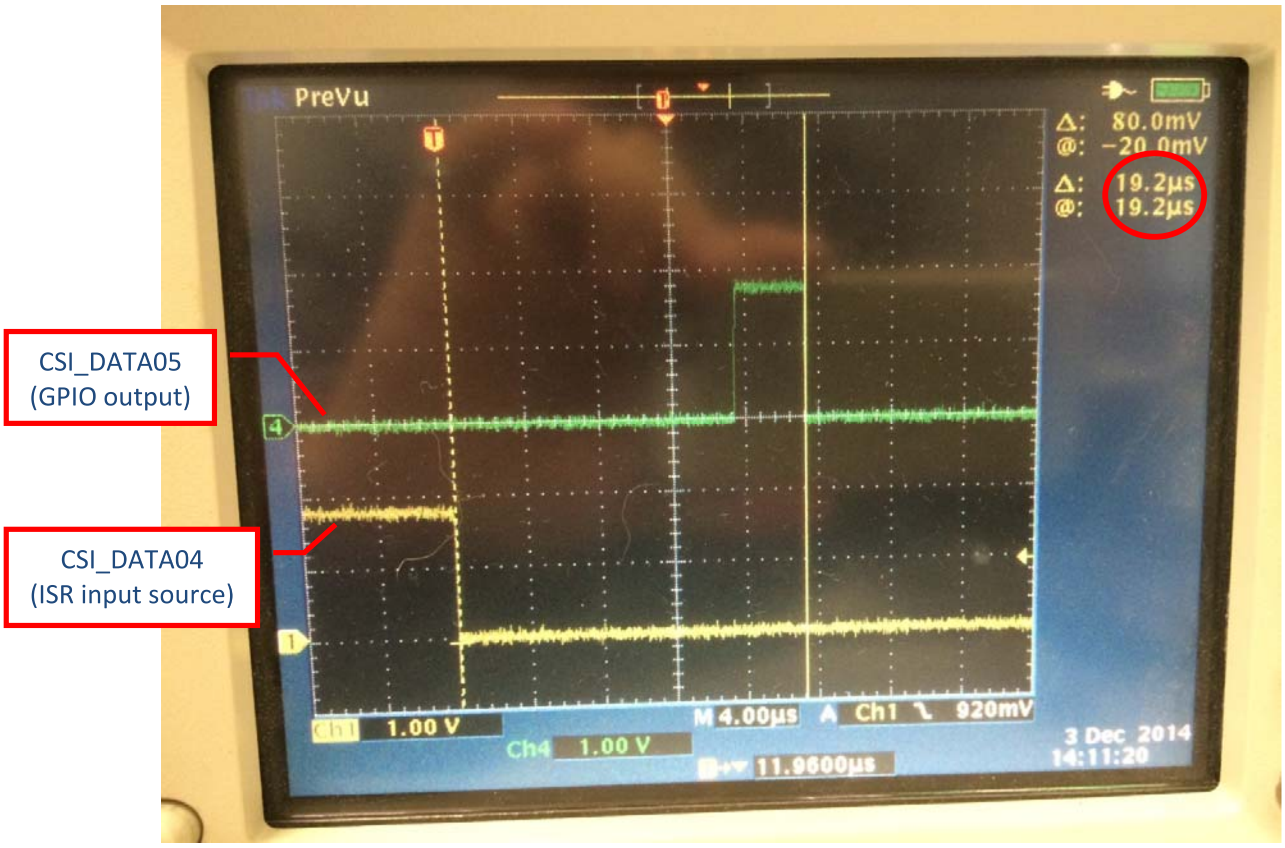


Figure 2. 19.2us is the time gap between external H/W GPIO interrupt pin and the end of useful instruction in ISR.