

Issue: HwiHdr is called twice

Reporoduce:

[http://software-dl.ti.com/processor-sdk-rtos/esd/docs/latest/rtos/index\\_how\\_to\\_guides.html#rtos-customization-using-an-external-input-to-trigger-an-interrupt-on-am57x](http://software-dl.ti.com/processor-sdk-rtos/esd/docs/latest/rtos/index_how_to_guides.html#rtos-customization-using-an-external-input-to-trigger-an-interrupt-on-am57x)

We reproduced with AM574IDK.

I uploaded project that can be reproduced.

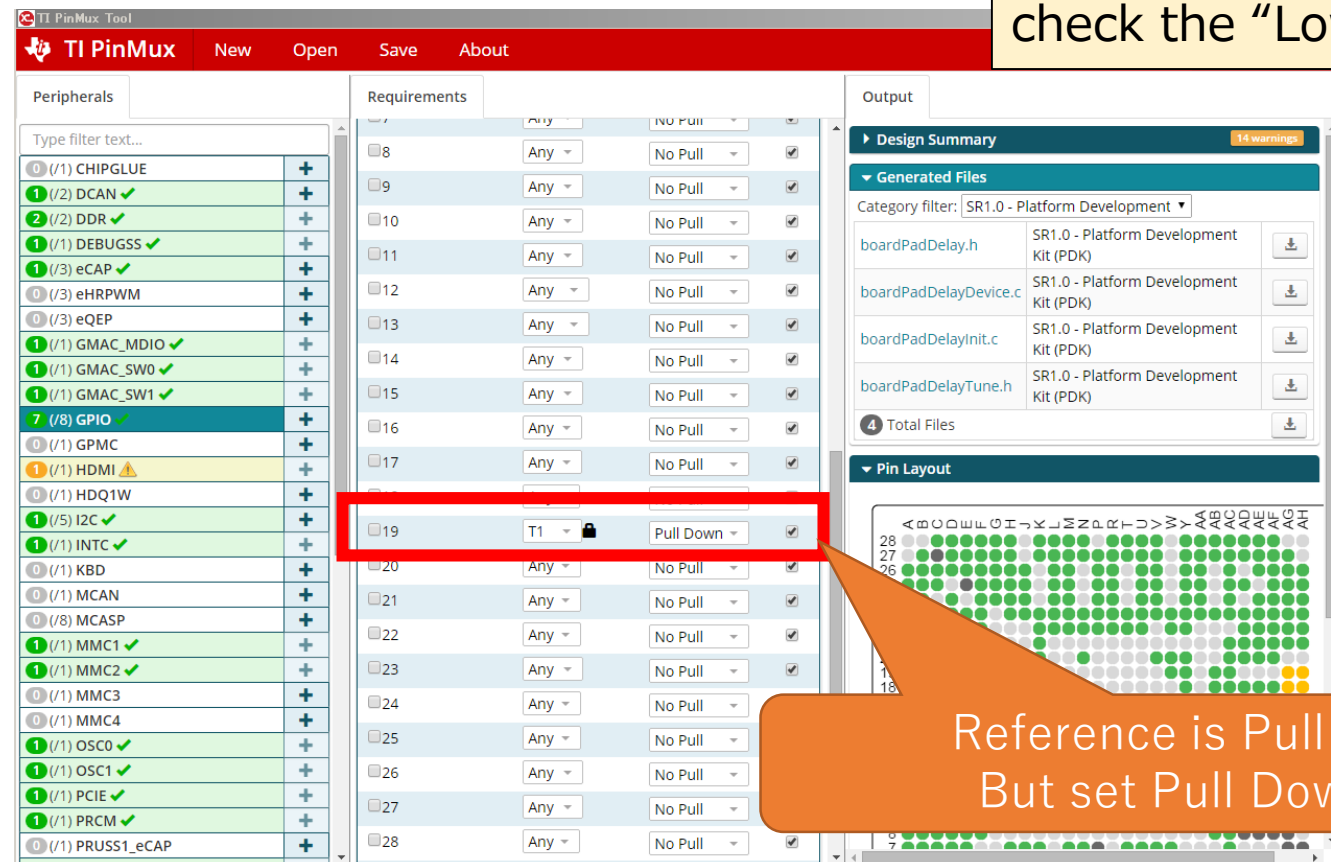
Difference from original test project is "Interrupt trigger".

Original test	My test
Edge trigger	→ High Level(Low Level)

# \*Supplement

When checking "High Level", it is necessary to set Pull Down in PinMux Tool and rebuild.

If you want to check with Pull Up, check the "Low Level".



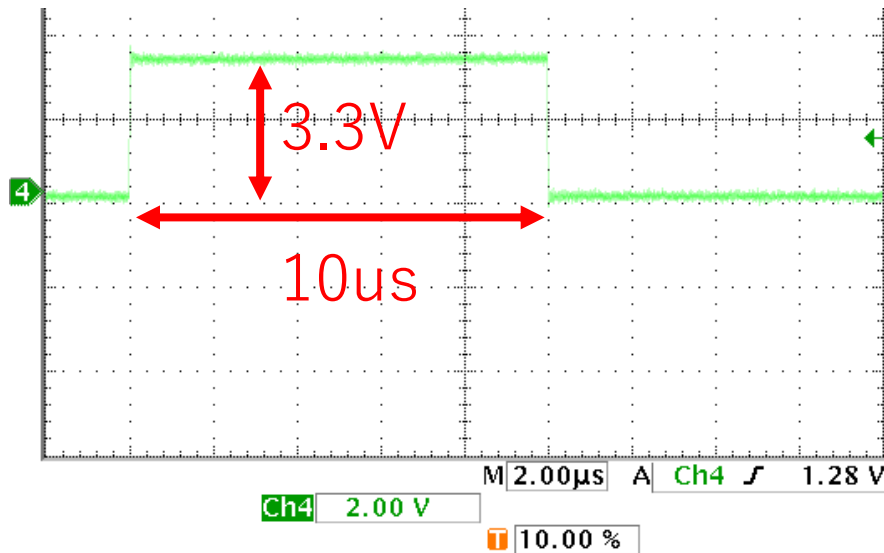
Reference is Pull Up.  
But set Pull Down.

C:\ti\pdk\_am57xx\_1\_0\_11\packages\ti\board\src\idkAM574x\idkAM574x\_pinmux.c

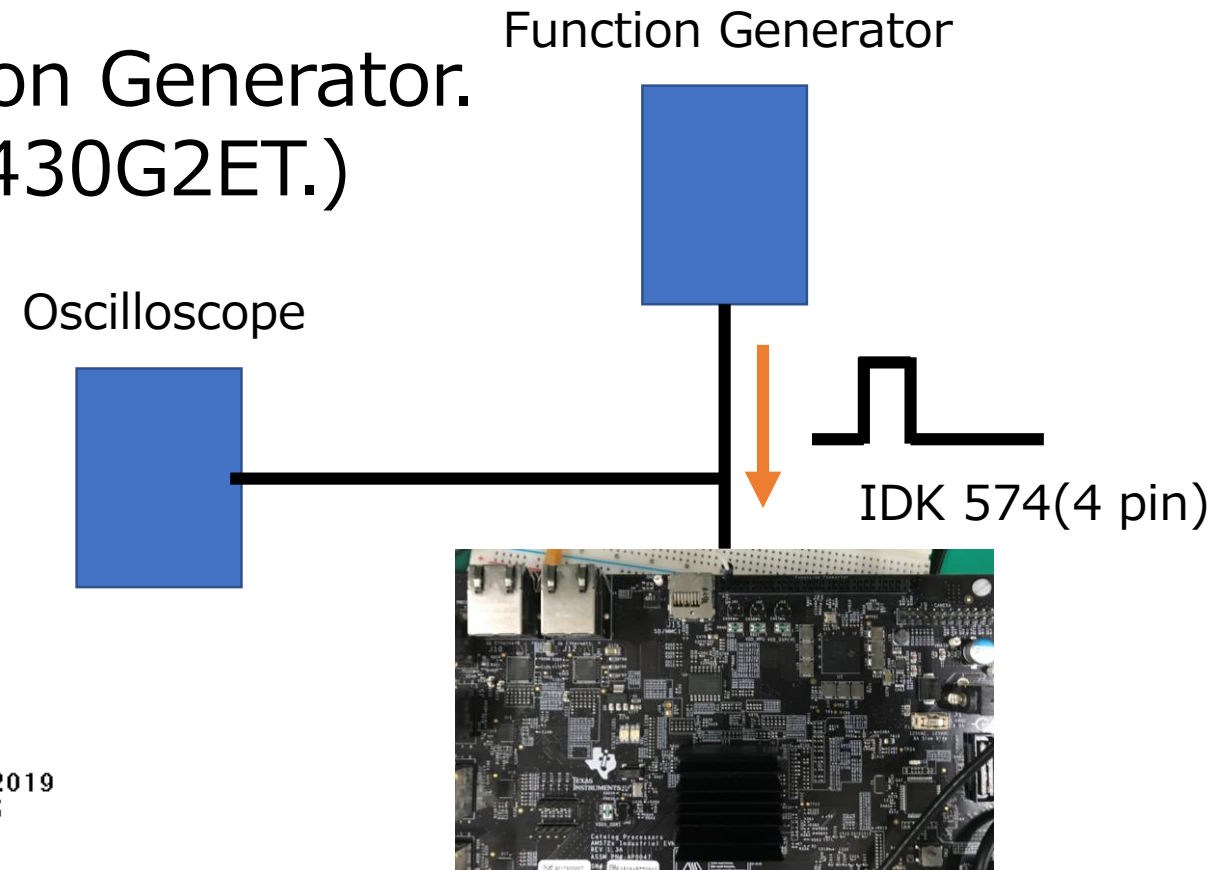
Set up :

Input a pulse to pin4 of the AM574IDK Expansion Connector.  
You can check GPIO interrupt.

I input a 10us pulse from Function Generator.  
(And make it easier, I used MSP430G2ET.)

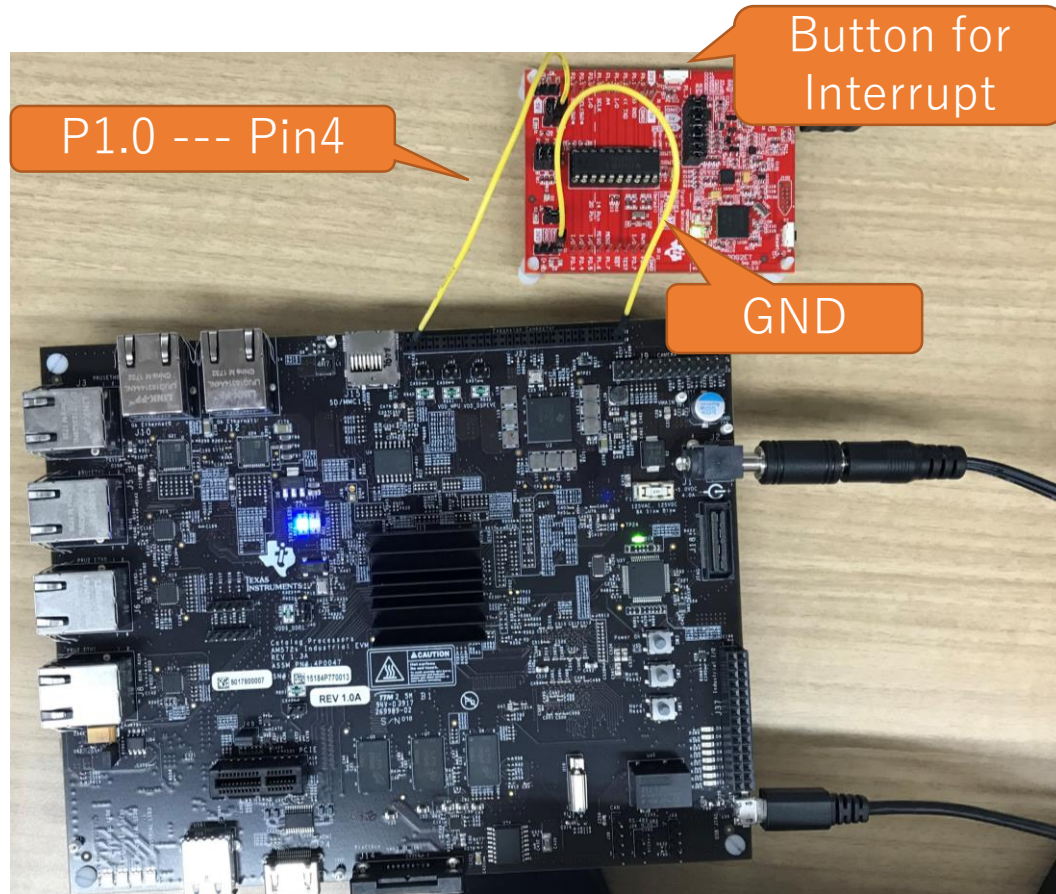


6 Aug 2019  
09:22:25



# \*Supplement

## Example of MSP430G2ET



```
#include <msp430.h>
```

```
int main(void)
```

```
{
```

```
    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer  
    P1DIR |= 0x01;             // Set P1.0 to output direction
```

```
    while (1)                  // Test P1.3
```

```
{
```

```
    if (0x08 & ~P1IN){
```

```
        P1OUT |= 0x01;        // if P1.3 set, set P1.0
```

```
        __delay_cycles(7);    // about 10.2us
```

```
        P1OUT &= ~0x01;
```

```
        __delay_cycles(1000000); //Chattering prevention
```

```
    }
```

```
    else
```

```
        P1OUT &= ~0x01;      // else reset
```

```
}
```

```
}
```

# Confirmation :

GPIO CallbackFxn worked twice during one pulse input.  
574IDK Blue LED also remains lit by toggling twice.

```
257 /*
258 * ===== Callback function =
259 */
260 void AppGpioCallbackFxn(void)
261 {
262     /* Toggle LED1 */
263     count++;
264     GPIO_toggle(USER_LED1);
265     AppDelay(DELAY_VALUE);
266     gpio_intr_triggered = 1;
267 }
268
```

xpression	Type	Value
(x)= count	int	2
+ Add new expression		



Count + 2

xpression	Type	Value
(x)= count	int	4

\* GPIO\_clearInt () is not added.  
I added debouncing code, but the situation got worse.

# Experiments :

So we did some experiments.

- Add `GPIO_clearInt()` in `CallbackFxn`
- Change the trigger    `HighLevel(LowLevel) -> Rising edge`

## Results :

- Add GPIO\_clearInt() in CallbackFxn

CallbackFxn is called only once.  
(count is once )

```
0
1  /* Toggle LED1 */
2  count++;
3  check=1;
4  GPIO_toggle(USER_LED1);
5  AppDelay(DELAY_VALUE);
6  gpio intr triggered = 1;
7  GPIO_clearInt(USER_LED0);
8 }
9
```

- Change the trigger in GPIO\_idkAM574x\_board.c

CallbackFxn is called only once.  
(count is once )

```
/* GPIO driver board specific pin configuration structure */
GPIO_PinConfig gpioPinConfigs_1p3[] = {
    /* Input pin with interrupt enabled */
    GPIO_DEVICE_CONFIG(GPIO_GRN_LED_PORT_NUM_1P3, GPIO_GRN_LED_PIN_NUM_1P3) |
    // GPIO_CFG_IN_INT_RISING | GPIO_CFG_INPUT,
    // GPIO_CFG_IN_INT_HIGH | GPIO_CFG_IN_PD | GPIO_CFG_INPUT,
    GPIO_CFG_IN_INT_HIGH | GPIO_CFG_INPUT,
```

Root issue :

We thought this issue was solved. However we confirmed about Hwi. So we noticed a serious anomaly of GPIO Driver (or sysbios).

We used Hwi.addHookSet function to check that Hwi is working properly. (Hwi timing, timestamp, what interrupt)

GpioCallbackFxn was executed once, but Hwi was happening twice.



## How to check :

When Hwi Begin/End or CallbackFxn is executed, execution time and contents are stored in "gLogData".

Stop Logging when a continuous interrupt is detected (when Hwi occurs twice in a short time).

Therefore, after generating an interrupt once, suspend IDK and please check "gLogData".

\*"gLogData" is ring buffer, it needs to be sorted.

# How to check : (Number format is Hex)

The screenshot shows a debugger's Variables window with a list of variables. The 'gLogData' variable is expanded, showing an array of log entries. Each entry is a struct containing fields like pStend, nEvtId, hHwi, and nTime. An orange callout box points to these fields and explains their values.

**nEvtId =**  
0x10000000 : HwiBegin  
0x10000001 : HwiEnd  
0x20000001 : CallbackFxn

**What interrupt from hHwi =**  
0x80025B1C : xxx (Timer?)  
0x810298B4 : GPIO  
0x00000000 : (CallbackFxn)

**Timestamp value**

Expression	Type	Value	Address
(x)= count	int	1	0x80026AC8
gLogData	struct <unnamed>[32]	[pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5FB1F58],... (Hex)	0x8102A57C
[0]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5FB1F58} (Hex)	0x8102A57C
[1]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5FB21E2} (Hex)	0x8102A58C
[2]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF60A6260} (Hex)	0x8102A59C
[3]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF60A64EA} (Hex)	0x8102A5AC
[4]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x810298B4,nTime=0xF613B652} (Hex)	0x8102A5BC
[5]	struct <unnamed>	{pStend=0x80022DAC {0x48 'C'},nEvtId=0x20000000,hHwi=0x00000000,nTime=0xF613BA97} (Hex)	0x8102A5CC
[6]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x810298B4,nTime=0xF683C257} (Hex)	0x8102A5DC
[7]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x810298B4,nTime=0xF683C56A} (Hex)	0x8102A5EC
[8]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x810298B4,nTime=0xF683C939} (Hex)	0x8102A5FC
[9]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF683CC10} (Hex)	0x8102A60C
[10]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF553744C} (Hex)	0x8102A61C
[11]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF55376D6} (Hex)	0x8102A62C
[12]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF562C244} (Hex)	0x8102A63C
[13]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF562C4CE} (Hex)	0x8102A64C
[14]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF571E990} (Hex)	0x8102A65C
[15]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF571EC1A} (Hex)	0x8102A66C
[16]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5811B34} (Hex)	0x8102A67C
[17]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5811DBE} (Hex)	0x8102A68C
[18]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5906E74} (Hex)	0x8102A69C
[19]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF59070FE} (Hex)	0x8102A6AC
[20]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF59FC1B4} (Hex)	0x8102A6BC
[21]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF59FC43E} (Hex)	0x8102A6CC
[22]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5AEF164} (Hex)	0x8102A6DC
[23]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5AEF3EE} (Hex)	0x8102A6EC
[24]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5BE1784} (Hex)	0x8102A6FC
[25]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5BE1A0E} (Hex)	0x8102A70C
[26]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5CD6868} (Hex)	0x8102A71C
[27]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5CD6AF2} (Hex)	0x8102A72C
[28]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5DCBB78} (Hex)	0x8102A73C
[29]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5DCBE02} (Hex)	0x8102A74C
[30]	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5EBF938} (Hex)	0x8102A75C
[31]	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5EBFBC2} (Hex)	0x8102A76C

# How to check : (Number format is Hex)

The screenshot shows a debugger's Variables window with the following columns: Expression, Type, Value, and Address. The variable `gLogData` is expanded, showing an array of 32 struct elements. The first few elements are highlighted with a red box, and a text box explains the sequence of events.

Expression	Type	Value	Address
<code>(x)= count</code>	int	1	0x80026AC8
<code>gLogData</code>	struct <unnamed>[32]	[pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5FB1F58],... (Hex)	0x8102A57C
<code>[0]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5FB1F58} (Hex)	0x8102A57C
<code>[1]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5FB21E2} (Hex)	0x8102A58C
<code>[2]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF60A6260} (Hex)	0x8102A59C
<code>[3]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF60A64EA} (Hex)	0x8102A5AC
<code>[4]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x810298B4,nTime=0xF613B652} (Hex)	0x8102A5BC
<code>[5]</code>	struct <unnamed>	{pStend=0x80022DAC {0x48 'C'},nEvtId=0x20000001,hHwi=0x00000000,nTime=0xF613BA97} (Hex)	0x8102A5CC
<code>[6]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x810298B4,nTime=0xF683C257} (Hex)	0x8102A5DC
<code>[7]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x810298B4,nTime=0xF683C56A} (Hex)	0x8102A5EC
<code>[8]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x810298B4,nTime=0xF683C939} (Hex)	0x8102A5FC
<code>[9]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF683CC10} (Hex)	0x8102A60C
<code>[10]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF553744C} (Hex)	0x8102A61C
<code>[11]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF55376D6} (Hex)	0x8102A62C
<code>[12]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x810298B4,nTime=0xF613B652} (Hex)	0x8102A63C
<code>[13]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x20000001,hHwi=0x00000000,nTime=0xF613BA97} (Hex)	0x8102A64C
<code>[14]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x810298B4,nTime=0xF683C257} (Hex)	0x8102A65C
<code>[15]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x810298B4,nTime=0xF683C56A} (Hex)	0x8102A66C
<code>[16]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF683CC10} (Hex)	0x8102A67C
<code>[17]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF553744C} (Hex)	0x8102A68C
<code>[18]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF55376D6} (Hex)	0x8102A69C
<code>[19]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x810298B4,nTime=0xF613B652} (Hex)	0x8102A6AC
<code>[20]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x20000001,hHwi=0x00000000,nTime=0xF613BA97} (Hex)	0x8102A6BC
<code>[21]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x810298B4,nTime=0xF683C257} (Hex)	0x8102A6CC
<code>[22]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x810298B4,nTime=0xF683C56A} (Hex)	0x8102A6DC
<code>[23]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5AEF164} (Hex)	0x8102A6EC
<code>[24]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5AEF3EE} (Hex)	0x8102A6FC
<code>[25]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5AEF600} (Hex)	0x8102A70C
<code>[26]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5CD6AF2} (Hex)	0x8102A71C
<code>[27]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5DCBB78} (Hex)	0x8102A72C
<code>[28]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5DCBE02} (Hex)	0x8102A73C
<code>[29]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5EBF938} (Hex)	0x8102A74C
<code>[30]</code>	struct <unnamed>	{pStend=0x80022D24 {0x65 'e'},nEvtId=0x10000001,hHwi=0x80025B1C,nTime=0xF5EBFBC2} (Hex)	0x8102A75C
<code>[31]</code>	struct <unnamed>	{pStend=0x80022D1C {0x62 'b'},nEvtId=0x10000000,hHwi=0x80025B1C,nTime=0xF5EBFBC2} (Hex)	0x8102A76C

GPIO Begin  
CallbackFxn  
GPIO End  
GPIO Begin  
GPIO End

CallbackFxn is once, GPIO Hwi is twice.

Root issue :

CallBackFxn is once, GPIO Hwi is twice.

When calculating the value of Timestamp (nTime), the wasteful Hwi occupies about 50us. It's a serious loss.

We checked in what case this occurred.

Level or Rising	GPIO_clearInt()	CallBackFxn	Hwi begin/end	Total Result
High/Low Level	-	2	2	NG
High/Low Level	Add	1	2	NG
Rising edge	-	1	2	NG
Rising edge	add	1	1	*OK

\* Result is "OK", but my customers want to use High Level.

# Requests:

We want to fix the root issue.

Level or Rising	GPIO_clearInt()	CallBackFxn	Hwi begin/end	Total Result
High/Low Level	*	1	1	OK
High/Low Level	Add	1	1	OK
Rising edge	*	1	1	OK
Rising edge	Add	1	1	OK

\* If GPIO\_clearInt ( ) needs to be added by user, please tell us? (At least not a test project .)

Reference: [C:/ti/pdk\\_am57xx\\_1\\_0\\_11/packages/ti/drv/gpio/docs/doxygen/html/\\_g\\_p\\_i\\_o\\_8h.html#a3653b24f2fa808ad1f35d12f7cbbbaecf](C:/ti/pdk_am57xx_1_0_11/packages/ti/drv/gpio/docs/doxygen/html/_g_p_i_o_8h.html#a3653b24f2fa808ad1f35d12f7cbbbaecf)

void GPIO\_clearInt()

Note :It is not necessary to call this API within a callback assigned to a pin.