Joy of innovation
nuvoTon

NuMicro® GPIO

A Leading MCU Platform Provider

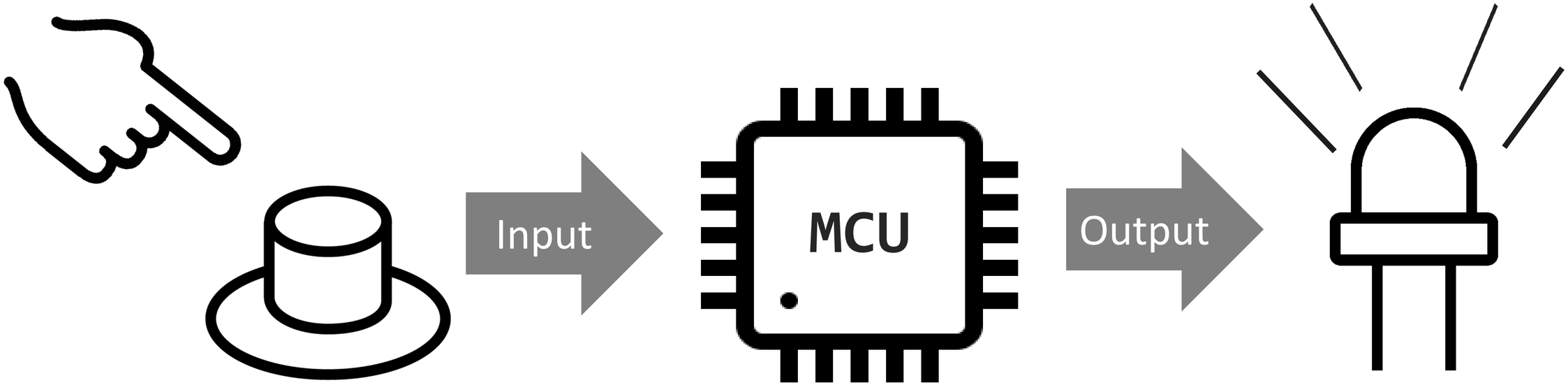
www.nuvoton.com

Agenda

- What is GPIO
- Feature
- Block Diagram
- GPIO Mode
- GPIO Functions
- Example code

What is GPIO

- **General Purpose Input Output**

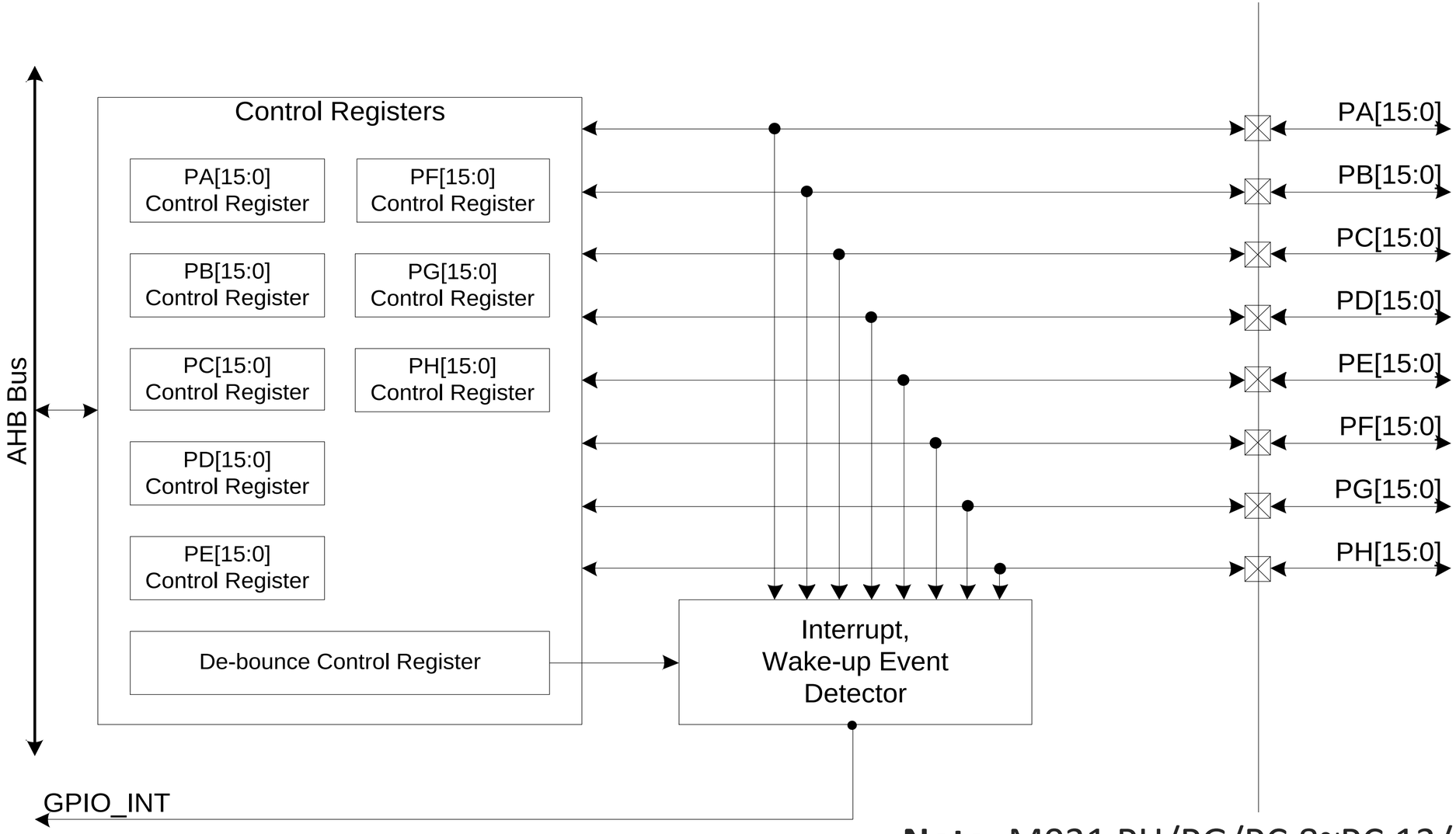


Feature

- **4 I/O modes:**
  - Input only with high impendence, Push-Pull Output, Open-Drain Output, Quasi-bidirectional
- **Configurable default I/O mode of all pins**
  - tri-state or Quasi-bidirectional
- **All pins support interrupt and wake-up function**
  - Level trigger or Edge trigger
- **TTL/Schmitt* trigger input selectable**

* : M031 only Schmitt trigger input

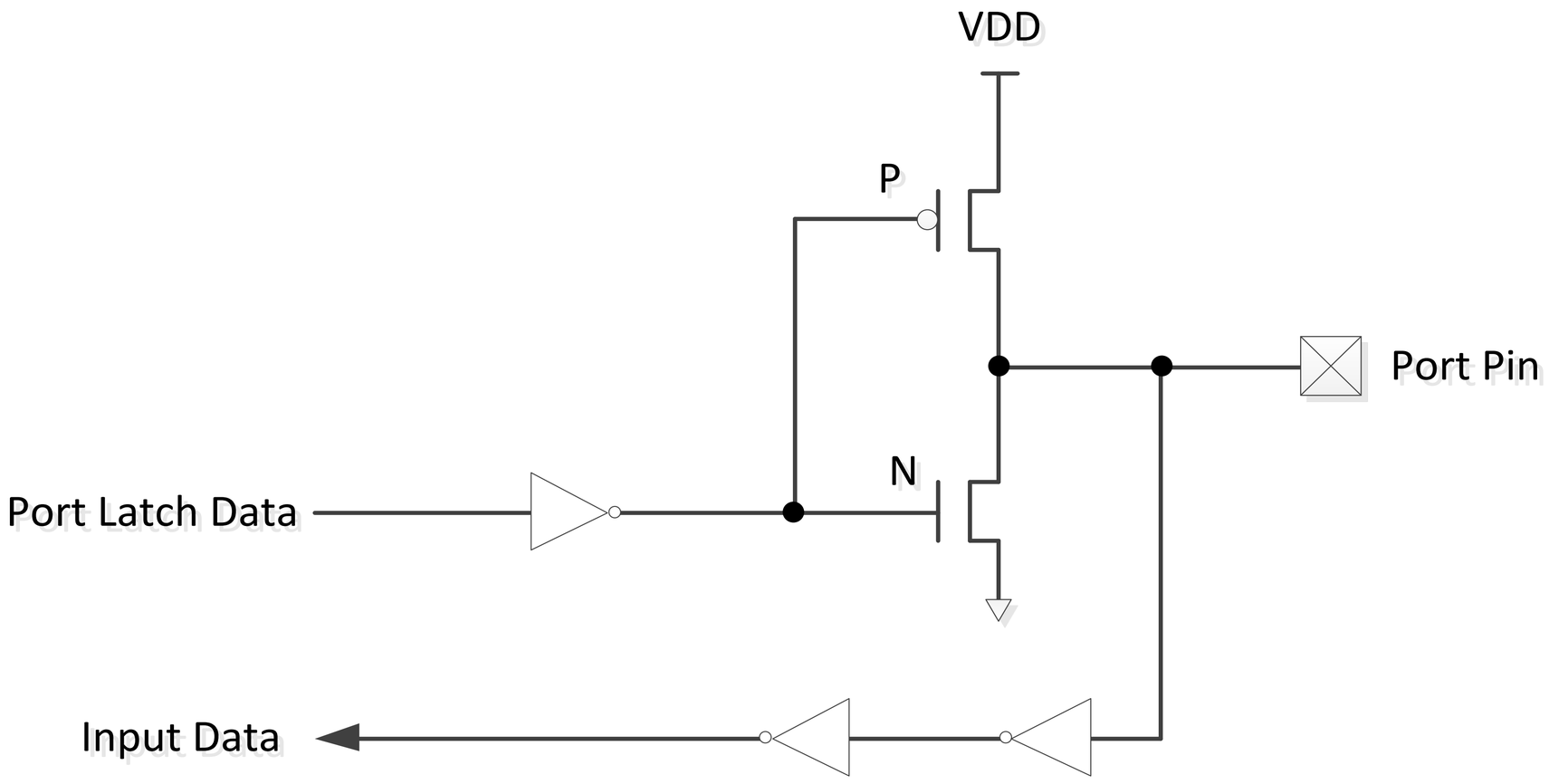
Block Diagram



Note: M031 PH/PG/PC.8~PC.13/PC.15/PD.4~PD.14/ PE.0~PE.15/PF.7~PF13 pins are ignored.

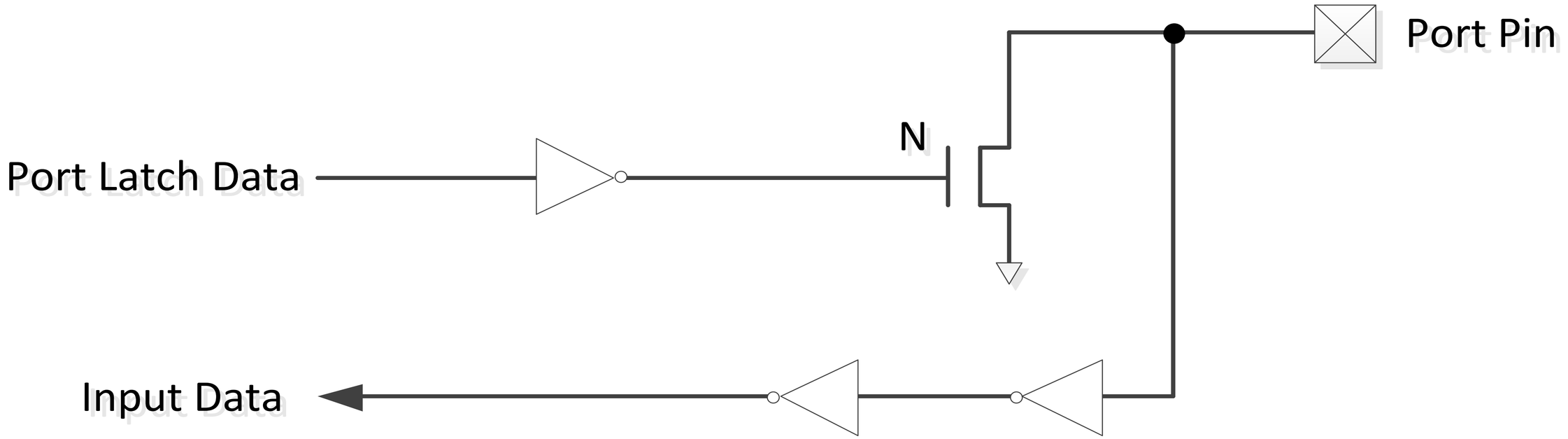
Push-pull Output Mode

- **The I/O pin supports digital output function with source/sink current capability.**



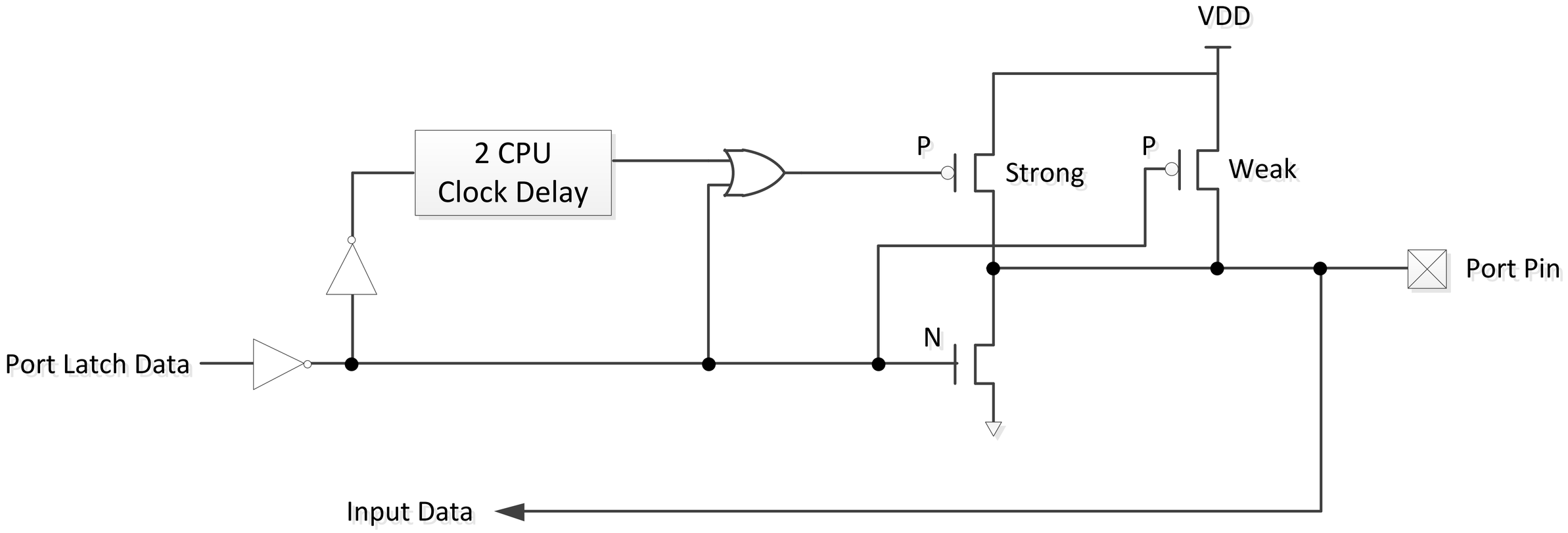
Open-drain Mode

- **Only NMOS sink current capability**



Quasi-bidirectional Mode

- **Supports digital output and input function at the same time**



GPIO mode

§ GPIO_SetMode()

```
void GPIO_SetMode ( GPIO_T *  port,
                    uint32_t  u32PinMask,
                    uint32_t  u32Mode
                  )
```

Set GPIO operation mode.

Parameters

[in]	**port**	GPIO port. It could be It could be PA, PB, PC, PD, PE, PF, PG or PH.
[in]	**u32PinMask**	The single or multiple pins of specified GPIO port. It could be BIT0 ~ BIT15 for PA, PB, PC, PD, PF and PH GPIO port. It could be BIT0 ~ BIT13 for PE GPIO port. It could be BIT0 ~ BIT11 for PG GPIO port.
[in]	**u32Mode**	Operation mode. It could be GPIO_MODE_INPUT, GPIO_MODE_OUTPUT, GPIO_MODE_OPEN_DRAIN, GPIO_MODE_QUASI.

Returns

None

```
/* Set PC.3 ~ PC.5 to GPIO output */
GPIO_SetMode(PC, (BIT3 | BIT4 | BIT5), GPIO_MODE_OUTPUT);
```

GPIO interrupt

◆ GPIO_EnableInt()

```
void GPIO_EnableInt ( GPIO_T *  port,
                      uint32_t  u32Pin,
                      uint32_t  u32IntAttribs
                    )
```

Enable GPIO interrupt.

Parameters

[in]	**port**	GPIO port. It could be PA, PB, PC, PD, or PF.
[in]	**u32Pin**	The pin of specified GPIO port. It could be 0 ~ 15 for PA and PB. It could be 0 ~ 7, and 14 for PC. It could be 0 ~ 3, and 15 for PD. It could be 0 ~ 6, 14, and 15 for PF.
[in]	**u32IntAttribs**	The interrupt attribute of specified GPIO pin. It could be

- **GPIO_INT_RISING**
- **GPIO_INT_FALLING**
- **GPIO_INT_BOTH_EDGE**
- **GPIO_INT_HIGH**
- **GPIO_INT_LOW**

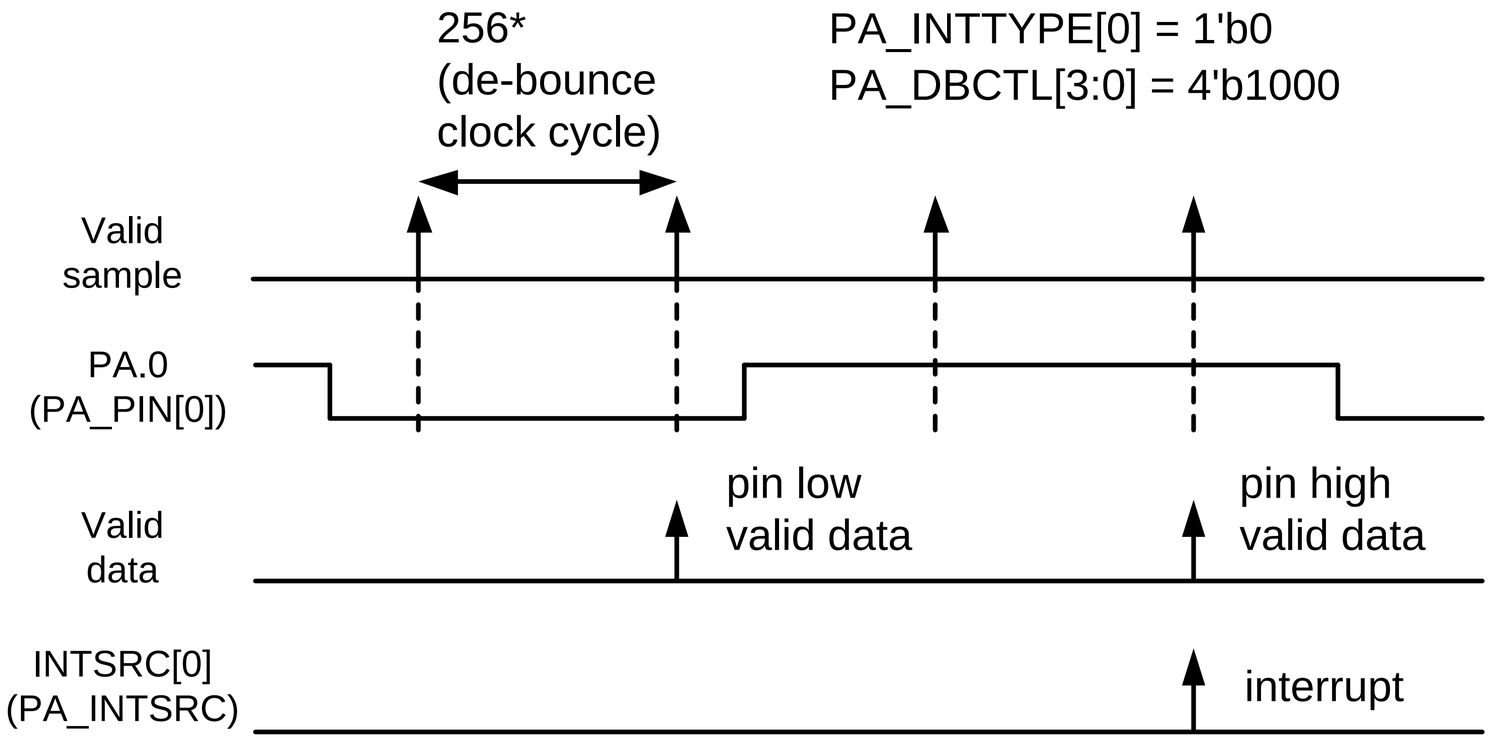
Returns

None

```
/* Enable PB.4 interrupt with falling edge trigger */
GPIO_EnableInt(PB, 4, GPIO_INT_FALLING);
```

GPIO De-bounce Function

- **The GPIO de-bounce function prevent unexpected interrupt happened which caused by noise.**



GPIO De-bounce Function

◆ GPIO_ENABLE_DEBOUNCE

```
#define GPIO_ENABLE_DEBOUNCE (  port,
                                u32PinMask
                             )
```

Enable Pin De-bounce Function.

Parameters

[in] **port** GPIO port. It could be PA, PB, PC, PD, or PF.

[in] **u32PinMask** The single or multiple pins of specified GPIO port. It could be BIT0 ~ BIT15 for PA and PB. It could be BIT0 ~ BIT7, and BIT14 for PC. It could be BIT0 ~ BIT3, and BIT15 for PD. It could be BIT0 ~ BIT6, BIT14, and BIT15 for PF.

Returns

None

§ GPIO_SET_DEBOUNCE_TIME

```
#define GPIO_SET_DEBOUNCE_TIME (  u32ClkSrc,
                                  u32ClkSel
                               )
```

Set De-bounce Sampling Cycle Time.

Parameters

[in] **u32ClkSrc** The de-bounce counter clock source. It could be GPIO_DBCTL_DBCLKSRC_HCLK or GPIO_DBCTL_DBCLKSRC_LIRC.

[in] **u32ClkSel** The de-bounce sampling cycle selection. It could be

- GPIO_DBCTL_DBCLKSEL_1
- GPIO_DBCTL_DBCLKSEL_2
- GPIO_DBCTL_DBCLKSEL_4
- GPIO_DBCTL_DBCLKSEL_8
- GPIO_DBCTL_DBCLKSEL_16
- GPIO_DBCTL_DBCLKSEL_32
- GPIO_DBCTL_DBCLKSEL_64
- GPIO_DBCTL_DBCLKSEL_128
- GPIO_DBCTL_DBCLKSEL_256
- GPIO_DBCTL_DBCLKSEL_512
- GPIO_DBCTL_DBCLKSEL_1024
- GPIO_DBCTL_DBCLKSEL_2048
- GPIO_DBCTL_DBCLKSEL_4096
- GPIO_DBCTL_DBCLKSEL_8192
- GPIO_DBCTL_DBCLKSEL_16384
- GPIO_DBCTL_DBCLKSEL_32768

Returns

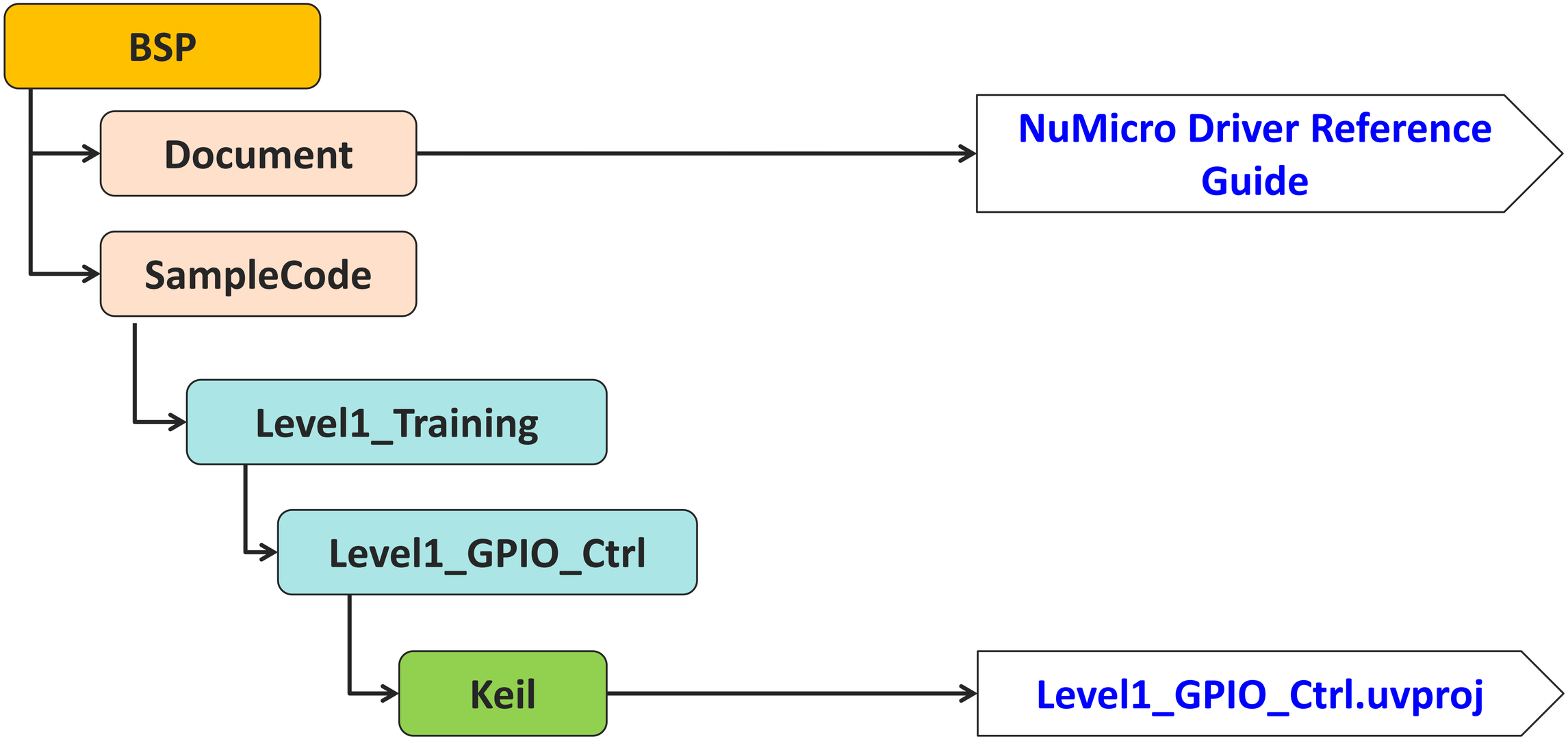
None

```
/* Set de-bounce function */
GPIO_SET_DEBOUNCE_TIME(GPIO_DBCTL_DBCLKSRC_LIRC, GPIO_DBCTL_DBCLKSEL_512);
GPIO_ENABLE_DEBOUNCE(PB, BIT4);
```

Example code

Blink LED with/without debounce

Example - Path



Example – Function

- **Press SW1 to control LED_R On / Off**
  - With De-bounce
- **Press SW2 to control LED_G On / Off**
  - Without De-bounce

Function	NuMaker-M031SD	NuMaker-M480-ETM
SW1	PB.4	PB.9
SW2	PB.0	PB.0
LED_R	PC.4	PC.9
LED_G	PC.5	PC.10
LED_B	PC.3	PC.11

Example - LED_Init()

Use mask to set multi-function pin to avoid affecting same group other pins

```
void LED_Init(void)
{
    /* Set PC.3 ~ PC.5 to GPIO */
    SYS->GPC_MFPL = (SYS->GPC_MFPL & ~(SYS_GPC_MFPL_PC3MFP_Msk |
                     SYS_GPC_MFPL_PC4MFP_Msk | SYS_GPC_MFPL_PC5MFP_Msk)) |
                    (SYS_GPC_MFPL_PC3MFP_GPIO | SYS_GPC_MFPL_PC4MFP_GPIO |
                     SYS_GPC_MFPL_PC5MFP_GPIO);

    /* Set PC.3 ~ PC.5 to GPIO output */
    GPIO_SetMode(PC, (BIT3 | BIT4 | BIT5), GPIO_MODE_OUTPUT);

    /* Let LED off after initialize */
    LED_R = LED_OFF;
    LED_G = LED_OFF;
    LED_B = LED_OFF;
}
```

Example - BTN_Init()

```
/**************  SW1 ***************/
/* Set PB.4 to GPIO */
SYS->GPB_MFPL = (SYS->GPB_MFPL & ~(SYS_GPB_MFPL_PB4MFP_Msk)) |
                (SYS_GPB_MFPL_PB4MFP_GPIO);
/* Set PB.4 to GPIO intput */
GPIO_SetMode(PB, BIT4, GPIO_MODE_INPUT);
GPIO_EnableInt(PB, 4, GPIO_INT_FALLING);
NVIC_EnableIRQ(GPIO_PAPB_IRQn);

/**************  SW2 ***************/
/* Set PB.0 to GPIO */
SYS->GPB_MFPL = (SYS->GPB_MFPL & ~(SYS_GPB_MFPL_PB0MFP_Msk)) |
                (SYS_GPB_MFPL_PB0MFP_GPIO);
/* Set PB.0 to GPIO intput */
GPIO_SetMode(PB, BIT0, GPIO_MODE_INPUT);
GPIO_EnableInt(PB, 0, GPIO_INT_FALLING);

/* Set de-bounce function */
GPIO_SET_DEBOUNCE_TIME(GPIO_DBCTL_DBCLKSRC_LIRC, GPIO_DBCTL_DBCLKSEL_512);
GPIO_ENABLE_DEBOUNCE(PB, BIT4);
```

The de-bounce clock source now is set to “LIRC”, remember to enable LIRC first.

Example - main()

```
/* Init LED */
LED_Init();

/* Init BTN */
BTN_Init();

while(1) {
    /* Check if the SW1 is pressed */
    if (sw1_int_cnt != sw1_cnt) {
        sw1_cnt = sw1_int_cnt;
        printf("SW1 interrupt count: %d\n", sw1_cnt);
    }
    /* Check if the SW2 is pressed */
    if (sw2_int_cnt != sw2_cnt) {
        sw2_cnt = sw2_int_cnt;
        printf("SW2 interrupt count: %d\n", sw2_cnt);
    }
}
```

Example - ISR

```
void GPAB_IRQHandler(void)
{
    /* Check if PB.4 the interrupt occurred */
    if(GPIO_GET_INT_FLAG(PB, BIT4)) {
        LED_R ^= 1;
        sw1_int_cnt++;
        /* Clear PB.4 interrupt flag */
        GPIO_CLR_INT_FLAG(PB, BIT4);
      /* Check if PB.0 the interrupt occurred */
    } else if(GPIO_GET_INT_FLAG(PB, BIT0)) {
        LED_G ^= 1;
        sw2_int_cnt++;
        /* Clear PB.0 interrupt flag */
        GPIO_CLR_INT_FLAG(PB, BIT0);
    } else {
        /* Un-expected interrupt. Just clear all PB interrupts */
        PB->INTSRC = PB->INTSRC;
        printf("Un-expected interrupts.\n");
    }
}
```

Example – Exercise

- Press SW1 to control LED_B On / Off

Additionally – Change Printf output port

- **Initialize UART port you used**
- **retarget.c**
  - Change the definition of DEBUG_PORT to UART port you used

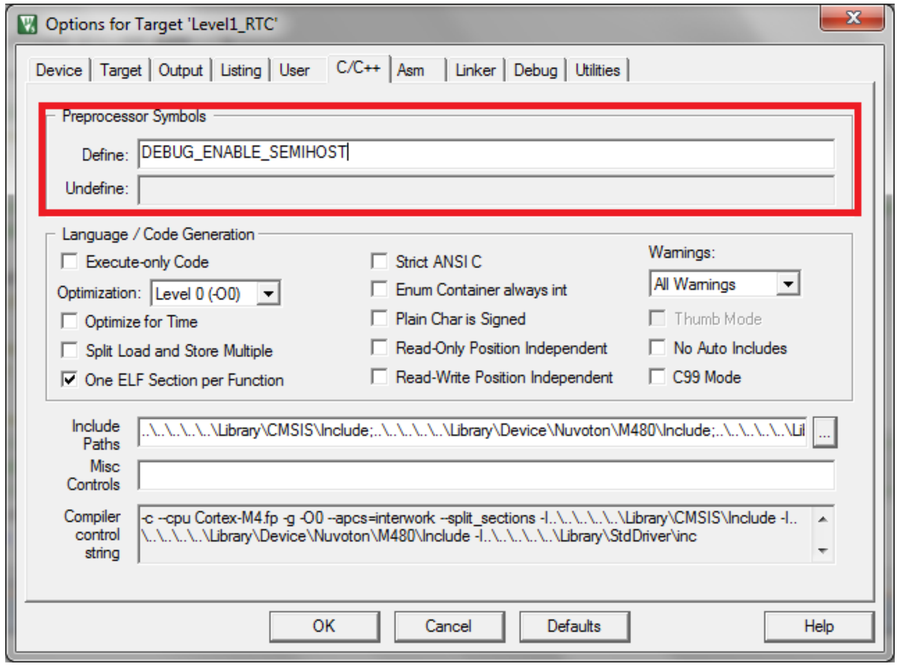
```
#if defined(DEBUG_ENABLE_SEMIHOST)
    #ifndef DISABLE_UART
        #define DISABLE_UART
    #endif
#endif

#define DEBUG_PORT    UART0
/*----------------------------------------------------------------------
/* Global variables
/*----------------------------------------------------------------------
#if !(defined(__ICCARM__) && (__VER__ >= 6010000))
struct   FILE
```

- **Rebulid**
- **Run**

Additionally – Enable Semihost (1/2)

- **Options -> C/C++**

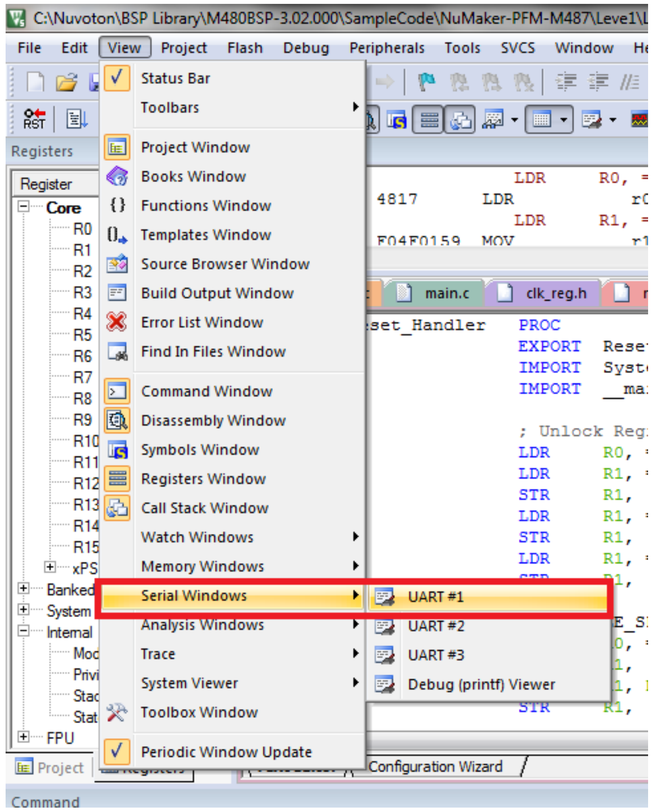


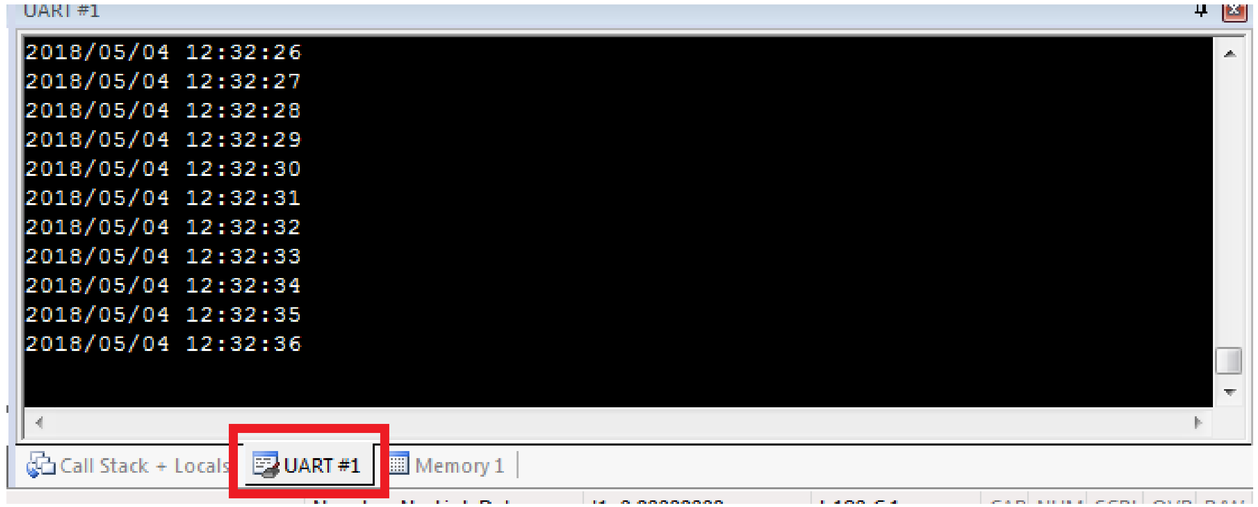
- **Rebuild**
- **Enter debug mode**

Note: It consumes CPU resources

Additionally – Enable Semihost (2/2)

- View -> Serial Windows -> UART #1





- Run

Joy of innovation
nuvoTon

Q & A

Thank you!

www.nuvoton.com