NT96680 Pure Linux Application Note

Table of Content

1. GPIO 4
2. I²C 10
3. SDIO 17
4. USB-HOST 24
5. SPI NAND/NOR MTD 31
6. SPI 35
7. ETHERNET 41
8. PWM 45
9. UART 74

Revision History

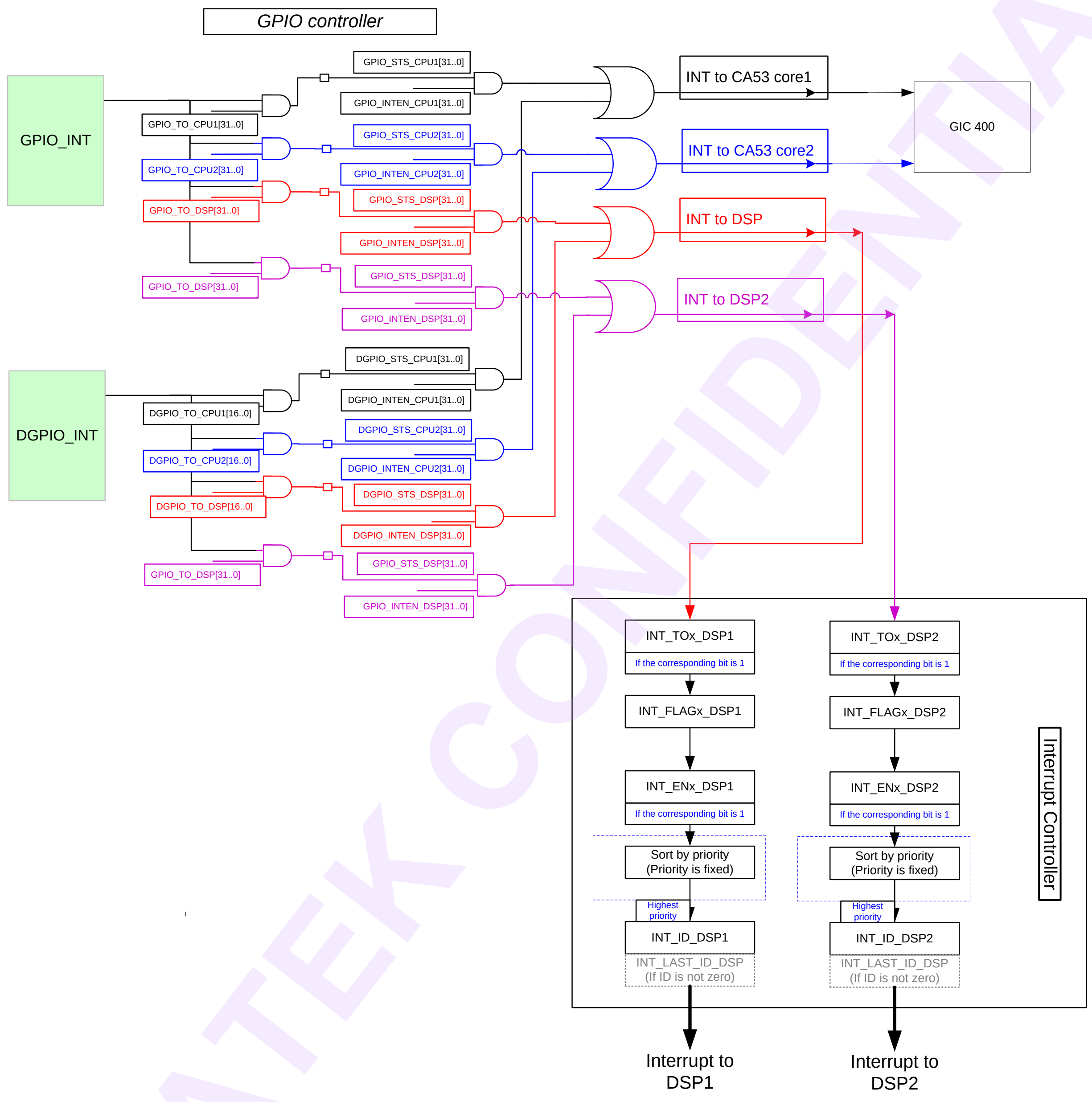
Date	Contents
2017/08/17	Initial version

1. GPIO

1.1. Feature

- Set/get pin to Input or output
- Set pin to HIGH/LOW (in output mode)
- Get PIN status (in input mode)
- Get interrupt status for specific PINs
- Set specific PINs to trigger interrupt
- Wake up system from sleep mode through GPIO interrupt PIN
- Edge (rising/falling/both) or level (low/high) trigger interrupt type

1.2. Block diagram



1.3. GPIO interrupt mapping

Interrupt No	GPIO PIN	Interrupt No	GPIO PIN
0	C_GPIO[3]	16	P_GPIO[36]
1	C_GPIO[9]	17	P_GPIO[37]
2	C_GPIO[15]	18	P_GPIO[41]
3	C_GPIO[21]	19	P_GPIO[20]
4	C_GPIO[26]	20	P_GPIO[21]
5	C_GPIO[27]	21	P_GPIO[43]
6	C_GPIO[28]	22	C_GPIO[25]
7	C_GPIO[33]	23	C_GPIO[31]
8	P_GPIO[3]	24	L_GPIO[14]
9	P_GPIO[6]	25	L_GPIO[15]
10	P_GPIO[11]	26	L_GPIO[20]
11	P_GPIO[15]	27	L_GPIO[21]
12	P_GPIO[19]	28	L_GPIO[22]
13	P_GPIO[29]	29	L_GPIO[26]
14	P_GPIO[31]	30	P_GPIO[44]
15	P_GPIO[35]	31	P_GPIO[47]

1.4. Usage in Linux

- General description

  Linux gpio driver provides APIs to get/set the gpio values, get/set the gpio directions, and achieve the irq number of specified gpio pins. Please use the provided GPIO mapping APIs (C_GPIO/P_GPIO/S_GPIO/L_GPIO/H_GPIO/D_GPIO) to indicate the gpio.

- Flow

  Following is an example to perform usage in gpio APIs.

```
#include <mach/nvt-gpio.h>
/* GPIO mapping APIs */
/* C_GPIO(pins) */
/* P_GPIO(pins) */
/* S_GPIO(pins) */
/* L_GPIO(pins) */
/* H_GPIO(pins) */
/* D_GPIO(pins) */

void foo(void)
{
    // Set C_GPIO 3 as example
    Uint32_t value, direction;
    gpio_direction_input(C_GPIO(3)); // Set C_CPIO 3 as an input pin
    Value = gpio_get_value(C_GPIO(3)); // Get the value of C_GPIO 3
    gpio_direction_output(C_GPIO(3), value);  // Set the output direction of C_CPIO 3 and target value
                                              // set output high or output low
    gpio_set_value(C_GPIO(3), value);  // Set the value of C_CPIO 3 if pin was set as output
    value = gpio_to_irq(C_GPIO(3)); // Get the irq number corresponding to C_GPIO 3
}
```

1.5. Usage in U-boot

- General description

  U-boot gpio driver provides APIs to get/set the gpio values, get/set the gpio directions. Please use the provided GPIO mapping APIs (C_GPIO/P_GPIO/S_GPIO/L_GPIO/H_GPIO/D_GPIO) to indicate the gpio.

- Enable gpio function in configuration file

BSP\u-boot\include\configs\nvt-na51000.h

```
/* Enable GPIO Usage*/
#define CONFIG_NA51000_GPIO
```

- An example for using general gpio APIs

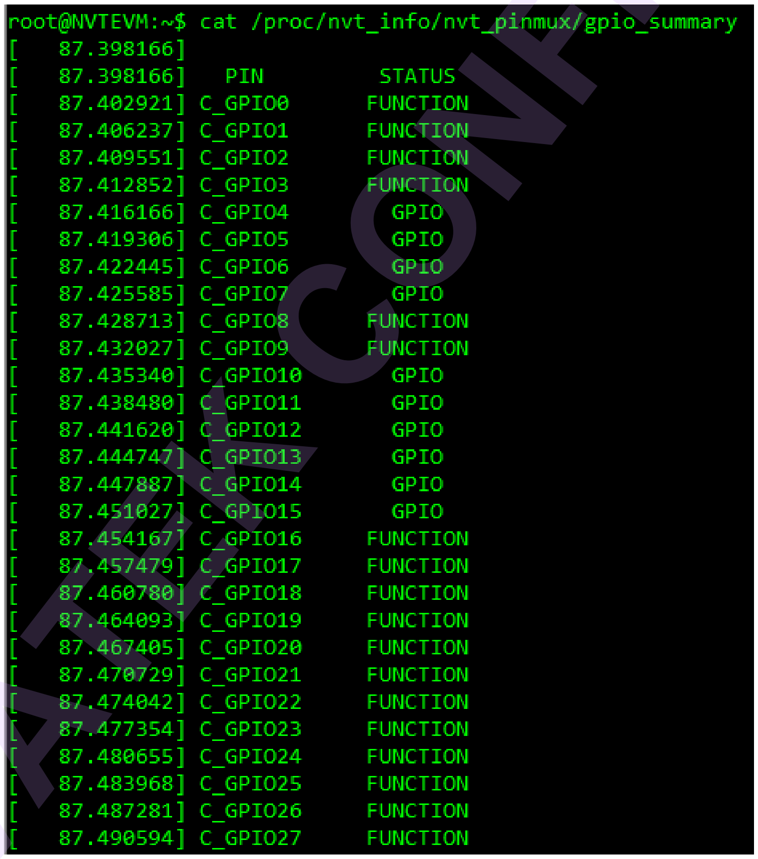
```
#include <asm/gpio.h>
/* GPIO mapping APIs */
/* C_GPIO(pins) */
/* P_GPIO(pins) */
/* S_GPIO(pins) */
/* L_GPIO(pins) */
/* H_GPIO(pins) */
/* D_GPIO(pins) */

void foo(void)
{
    // Set C_GPIO 3 as example
    Uint32_t value, direction;
    gpio_direction_input(C_GPIO(3)); // Set C_CPIO 3 as an input pin
    Value = gpio_get_value(C_GPIO(3)); // Get the value of C_GPIO 3
    gpio_direction_output(C_GPIO(3), value);  // Set the output direction of C_CPIO 3
    gpio_set_value(C_GPIO(3), value);  // Set the value of C_CPIO 3
}
```

1.6. Usage in pinctrl tool

- ☐ Each pin default status could be configured in pinctrl tool, including pad driving, gpio direction, and pull-up/down.
  - ◆ Location: build/nvt-tools/nvt_pinctrl_tool/top_generator.xlsm
    - ● Please refer more details in NT96680_TOOL_UI_pinctrl_tool

1.7. Troubleshooting

- ☐ GPIO pins are not functional even if controlling APIs were set.
  - ◆ Check gpio info to make sure pins are switched to GPIO mode
    - ● cat /proc/nvt_info/nvt_pinmux/gpio_summary



```
root@NVTEVM:~$ cat /proc/nvt_info/nvt_pinmux/gpio_summary
[   87.398166]
[   87.398166]   PIN         STATUS
[   87.402921] C_GPIO0      FUNCTION
[   87.406237] C_GPIO1      FUNCTION
[   87.409551] C_GPIO2      FUNCTION
[   87.412852] C_GPIO3      FUNCTION
[   87.416166] C_GPIO4        GPIO
[   87.419306] C_GPIO5        GPIO
[   87.422445] C_GPIO6        GPIO
[   87.425585] C_GPIO7        GPIO
[   87.428713] C_GPIO8      FUNCTION
[   87.432027] C_GPIO9      FUNCTION
[   87.435340] C_GPIO10       GPIO
[   87.438480] C_GPIO11       GPIO
[   87.441620] C_GPIO12       GPIO
[   87.444747] C_GPIO13       GPIO
[   87.447887] C_GPIO14       GPIO
[   87.451027] C_GPIO15       GPIO
[   87.454167] C_GPIO16     FUNCTION
[   87.457479] C_GPIO17     FUNCTION
[   87.460780] C_GPIO18     FUNCTION
[   87.464093] C_GPIO19     FUNCTION
[   87.467405] C_GPIO20     FUNCTION
[   87.470729] C_GPIO21     FUNCTION
[   87.474042] C_GPIO22     FUNCTION
[   87.477354] C_GPIO23     FUNCTION
[   87.480655] C_GPIO24     FUNCTION
[   87.483968] C_GPIO25     FUNCTION
[   87.487281] C_GPIO26     FUNCTION
[   87.490594] C_GPIO27     FUNCTION
```

2. I²C

2.1. Feature

- Support standard (100 kbit/s) and fast (400 kbit/s) and fast plus(1Mbit/s) mode through configuring the clock divided register.
- Provide Master-Transmit, Master-Receive.
- Multiple masters can reside on the I²C bus at the same time.

Note:

1. Source clock is a fixed 48 MHz clock from CG.

2.2. Block diagram

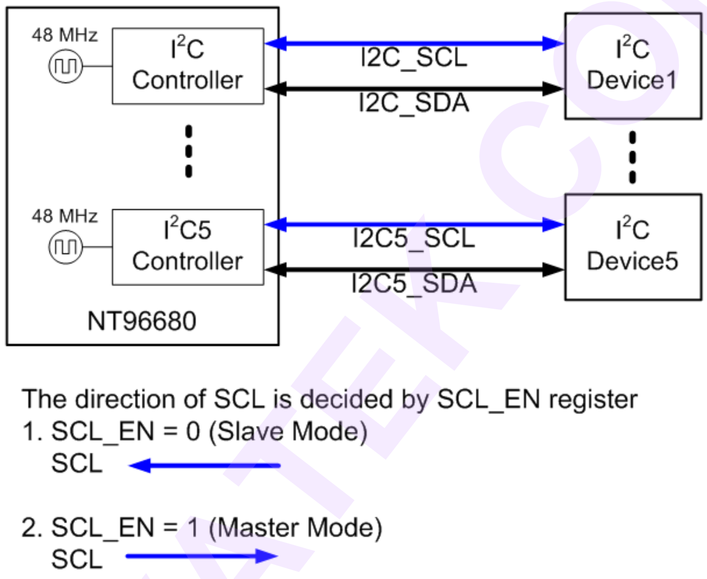


Figure 2-1 I²C diagram

2.3. Header file

The header file is located in /include/linux/i2c.h.

Place #include <linux/i2c.h> in your i2c device driver.

2.4. Function description

In Linux, there are two types of I²C driver: I²C controller driver and I²C device driver. I²C controller driver is provided by Novatek to abstract details of I²C h/w controller. I²C device driver should be written by NT9668x users/customers to communicate with their I²C slave devices.

This document is intended to address on how to link NT9668x I²C controller driver to your own I²C device driver.

2.4.1. Write your I²C device driver

- General description

NT9668x I²C controller driver is encapsulated as a platform driver. You should create a platform device in the device tree to probe I²C controller driver.

- Limitations
  - ☐ .name (of platform_device) should be "nvt_i2c"

- Flow

Following is a sample to create a platform device which can probe I²C controller driver:

```
i2c@f0220000 {                                      //i2c register base address
     compatible = "nvt,nvt_i2c";
     reg = <0xf0220000 0x100>;                      //i2c register base address + offset
     interrupts = <GIC_SPI 41 IRQ_TYPE_LEVEL_HIGH>;  //i2c interrupt
     clock-frequency = <100000>;                    //i2c clock rate
     #address-cells = <1>;
     #size-cells = <0>;

     myI2C@50 {                                     // child node for i2c slave device
        compatible = "nvt,my_i2c_name";
        reg = <0x50>;                               // i2c slave device address
     };
}
```

2.4.2. Write your I^{2}C device driver

- General description

  This document provides some hints on writing kernel space I^{2}C device driver:

  1. Declare a i2c_driver object
  2. Register your i2c_driver object by MACRO module_i2c_driver()

- Limitations (for kernel space driver)

  ☐ . of_match_table of (i2c_driver) should be the same with dts

- Flow (for kernel space driver)

  Here is a sample to register an I^{2}C device driver:

```
static const struct i2c_device_id my_i2c_id[] = {
    { "my_i2c_name", MY_I2C_ADDR },
    { },
};
static struct of_device_id my_match_table[] = {
    { .compatible = "nvt,my_i2c_name",},
    { },
};
static struct i2c_driver my_i2c_driver = {
    .driver = {
        .name = "my_i2c_name",
        .owner = THIS_MODULE,
        .of_match_table = of_match_ptr(my_match_table),
    },
    .probe = my_i2c_probe,
    .remove = my_i2c_remove,
    .id_table = my_i2c_id,
};
module_i2c_driver(my_i2c_driver);
```

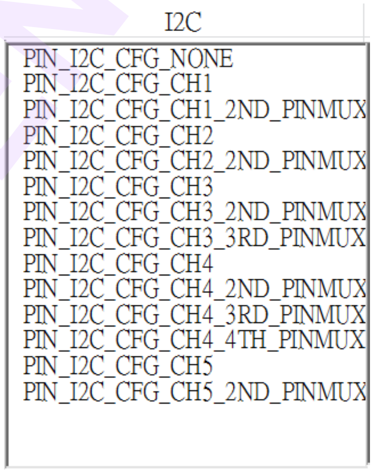
Here is a sample to send and read data by Linux I^{2}C APIs:

```
#include <linux/i2c.h>

/*read data*/
static int32_t nvt_i2c_read(struct i2c_client *client, uint8_t *buf, uint16_t len) {
struct i2c_msg msgs[2];
int32_t ret = -1;

        msgs[0].flags = !I2C_M_RD;
        msgs[0].addr  = client->addr;
        msgs[0].len   = 2;
        msgs[0].buf   = &buf[0];

        msgs[1].flags = I2C_M_RD;
        msgs[1].addr  = client->addr;
        msgs[1].len   = len-2;
        msgs[1].buf   = &buf[2];

        ret = i2c_transfer(client->adapter, msgs, 2);
        return ret;
}

/*write data*/
static int32_t nvt_i2c_write(struct i2c_client *client, uint8_t *buf, uint16_t len)
{
        struct i2c_msg msg;
        int32_t ret = -1;

        msg.flags = !I2C_M_RD;
        msg.addr  = client->addr;
        msg.len   = len;
        msg.buf   = buf;

        ret = i2c_transfer(client->adapter, &msg, 1);
return ret;
}
```

2.4.3. Introduce other I^{2}C buses in Linux

Basically, I^{2}C is designated as Linux dedicated interface while other I^{2}C buses might be used in uitron platform. However, people could be able to use all of I^{2}C buses in Linux platform with the following steps

1. All the related pinmux configuration shall be setup with pinctrl tool
   - Location: build/nvt-tools/nvt_pinctrl_tool/top_generator.xlsm
   - Please refer more details in NT96680_TOOL_UI_pinctrl_tool


I2C
PIN_I2C_CFG_NONE
PIN_I2C_CFG_CH1
PIN_I2C_CFG_CH1_2ND_PINMUX
PIN_I2C_CFG_CH2
PIN_I2C_CFG_CH2_2ND_PINMUX
PIN_I2C_CFG_CH3
PIN_I2C_CFG_CH3_2ND_PINMUX
PIN_I2C_CFG_CH3_3RD_PINMUX
PIN_I2C_CFG_CH4
PIN_I2C_CFG_CH4_2ND_PINMUX
PIN_I2C_CFG_CH4_3RD_PINMUX
PIN_I2C_CFG_CH4_4TH_PINMUX
PIN_I2C_CFG_CH5
PIN_I2C_CFG_CH5_2ND_PINMUX

2. Register bus in device tree

```
i2c@f0220000 {                                        //i2c register base address
    compatible = "nvt,nvt_i2c";
    reg = <0xf0220000 0x100>;                       //i2c register base address + offset
    interrupts = <GIC_SPI 41 IRQ_TYPE_LEVEL_HIGH>;   //i2c interrupt
    clock-frequency = <100000>;                      //i2c clock rate
};
i2c@f0350000 {                                        //i2c register base address
    compatible = "nvt,nvt_i2c";
    reg = <0xf0350000 0x100>;                       //i2c register base address + offset
    interrupts = <GIC_SPI 42 IRQ_TYPE_LEVEL_HIGH>;   //i2c interrupt
    clock-frequency = <100000>;                      //i2c clock rate
};

i2c@f03a0000 {                                        //i2c register base address
    compatible = "nvt,nvt_i2c";
    reg = <0xf03a0000 0x100>;                       //i2c register base address + offset
    interrupts = <GIC_SPI 60 IRQ_TYPE_LEVEL_HIGH>;   //i2c interrupt
    clock-frequency = <100000>;                      //i2c clock rate
};

i2c@f03b0000 {                                        //i2c register base address
    compatible = "nvt,nvt_i2c";
    reg = <0xf03b0000 0x100>;                        //i2c register base address + offset
    interrupts = <GIC_SPI 82 IRQ_TYPE_LEVEL_HIGH>;   //i2c interrupt
    clock-frequency = <100000>;                      //i2c clock rate
};

i2c@f03c0000 {                                        //i2c register base address
    compatible = "nvt,nvt_i2c";
    reg = <0xf03c0000 0x100>;                        //i2c register base address + offset
    interrupts = <GIC_SPI 83 IRQ_TYPE_LEVEL_HIGH>;   //i2c interrupt
    clock-frequency = <100000>;                      //i2c clock rate
};
```

3. Register clock in clock driver

BSP\linux-kernel\drivers\clk\novatek\na51000-clk.c

```
static struct nvt_composite_gate_clk na51000_cgate_clk[] __initdata = {
#ifdef CONFIG_USE_OF
…
…
COMP_GATE_CONF("f0220000.i2c", "fix48m", 48000000, 0, 0, 0,
CG_CLK_EN_REG1_OFFSET, BIT4, NOT_ENABLE,
CG_SYS_RESET_REG1_OFFSET, BIT4, NOT_RESET,
0, 0, 0),
COMP_GATE_CONF("f0350000.i2c", "fix48m", 48000000, 0, 0, 0,
CG_CLK_EN_REG1_OFFSET, BIT5, NOT_ENABLE,
CG_SYS_RESET_REG1_OFFSET, BIT5, NOT_RESET,
0, 0, 0),
COMP_GATE_CONF("f03a0000.i2c", "fix48m", 48000000, 0, 0, 0,
CG_CLK_EN_REG1_OFFSET, BIT31, NOT_ENABLE,
CG_SYS_RESET_REG1_OFFSET, BIT31, NOT_RESET,
0, 0, 0),
COMP_GATE_CONF("f03b0000.i2c", "fix48m", 48000000, 0, 0, 0,
CG_CLK_EN_REG2_OFFSET, BIT26, NOT_ENABLE,
CG_SYS_RESET_REG2_OFFSET, BIT26, NOT_RESET,
0, 0, 0),
COMP_GATE_CONF("f03c0000.i2c", "fix48m", 48000000, 0, 0, 0,
CG_CLK_EN_REG2_OFFSET, BIT27, NOT_ENABLE,
CG_SYS_RESET_REG2_OFFSET, BIT27, NOT_RESET,
0, 0, 0),
..
…
```

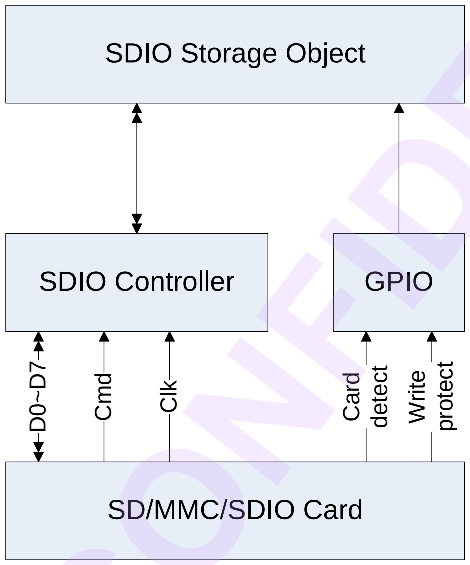
2.5. Troubleshooting

3. SDIO

3.1. Feature

- Support SD/MMC/IO interface driver for Linux SD/MMC/IO Core framework.

3.2. Block Diagram



3.3. Header File

The header file is located in \include\linux\mmc\

Place #include <linux/sdio.h>, #include <linux/mmc.h> … in \include\linux\mmc\ directory in your source code for SDIO, SDIO2 and SDIO3 modules.

3.4. Function Description

The SD/MMC driver module provides utility to access SD/MMC card for file system or IO access.

- PIN Description

Pin Name	Descriptions
SDIO_CLK	SDIO clock pin
SDIO_CMD	SDIO command pin
SDIO_D0	SDIO data 0 pin
SDIO_D1	SDIO data 1 pin

SDIO_D2	SDIO data 2 pin
SDIO_D3	SDIO data 3 pin

Pin Name	Descriptions
SDIO2_CLK	SDIO2 clock pin
SDIO2_CMD	SDIO2 command pin
SDIO2_D0	SDIO2 data 0 pin
SDIO2_D1	SDIO2 data 1 pin
SDIO2_D2	SDIO2 data 2 pin
SDIO2_D3	SDIO2 data 3 pin

Pin Name	Descriptions
SDIO3_CLK	SDIO3 clock pin
SDIO3_CMD	SDIO3 command pin
SDIO3_D0	SDIO3 data 0 pin
SDIO3_D1	SDIO3 data 1 pin
SDIO3_D2	SDIO3 data 2 pin
SDIO3_D3	SDIO3 data 3 pin
SDIO3_D4	SDIO3 data 4 pin
SDIO3_D5	SDIO3 data 5 pin
SDIO3_D6	SDIO3 data 6 pin
SDIO3_D7	SDIO3 data 7 pin

3.4.1. Linux configurations

- General Description

  Please check the configuration is correct in Linux.

CONFIG_MMC=y

CONFIG_MMC_BLOCK=y

CONFIG_MMC_BLOCK_MINORS=8

CONFIG_MMC_BLOCK_BOUNCE=y

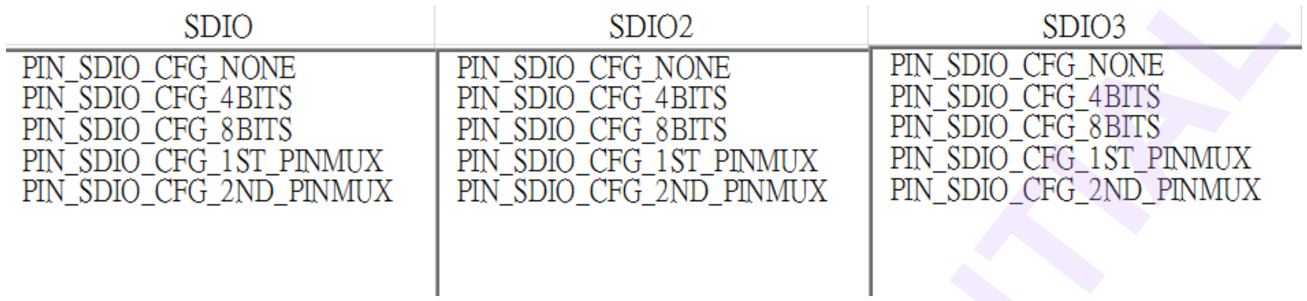
CONFIG_MMC_NA51000=y

- Configuration such as:

```
{
 CONFIG_MMC=y
# CONFIG_MMC_DEBUG is not set
# CONFIG_MMC_UNSAFE_RESUME is not set
# CONFIG_MMC_CLKGATE is not set
# CONFIG_MMC_EMBEDDED_SDIO is not set
# CONFIG_MMC_PARANOID_SD_INIT is not set

#
# MMC/SD/SDIO Card Drivers
#
CONFIG_MMC_BLOCK=y
CONFIG_MMC_BLOCK_MINORS=8
CONFIG_MMC_BLOCK_BOUNCE=y
# CONFIG_MMC_BLOCK_DEFERRED_RESUME is not set
# CONFIG_SDIO_UART is not set
# CONFIG_MMC_TEST is not set

#
# MMC/SD/SDIO Host Controller Drivers
#
# CONFIG_MMC_SDHCI is not set
# CONFIG_MMC_SDHCI_PXAV3 is not set
# CONFIG_MMC_SDHCI_PXAV2 is not set
CONFIG_MMC_NA51000=y
# CONFIG_MMC_SPI is not set
# CONFIG_MMC_VUB300 is not set
# CONFIG_MMC_USHC is not set
}
```

3.4.2.

1. All the related pinmux configuration shall be setup with pinctrl tool
   - ☐ Location: build/nvt-tools/nvt_pinctrl_tool/top_generator.xlsm
   - ☐ Please refer more details in NT96680_TOOL_UI_pinctrl_tool



SDIO	SDIO2	SDIO3
PIN_SDIO_CFG_NONE PIN_SDIO_CFG_4BITS PIN_SDIO_CFG_8BITS PIN_SDIO_CFG_1ST_PINMUX PIN_SDIO_CFG_2ND_PINMUX	PIN_SDIO_CFG_NONE PIN_SDIO_CFG_4BITS PIN_SDIO_CFG_8BITS PIN_SDIO_CFG_1ST_PINMUX PIN_SDIO_CFG_2ND_PINMUX	PIN_SDIO_CFG_NONE PIN_SDIO_CFG_4BITS PIN_SDIO_CFG_8BITS PIN_SDIO_CFG_1ST_PINMUX PIN_SDIO_CFG_2ND_PINMUX

3.4.3. Enable and select card detect pins

- General Description

Enable card detect function by using interrupt pin, and select target gpio for card detect pin. These configurations are set in nvt-na51000-peri.dts

- Enable card detect configuration such as:

```
mmc@f0420000 {
     compatible = "nvt,nvt_mmc";
     reg = <0xf0420000 0x1000>;
     interrupts = <GIC_SPI 30 IRQ_TYPE_LEVEL_HIGH>;
     max-frequency = <48000000>; //Set maximum clock frequency
     voltage-switch = <0>; //Set 1 if SDIO supports SDR50/SDR104 voltage switch (3.3V->1.8V)
     max-voltage = <3300>; //Set maximum voltage of your SDIO module required
     bus-width = <4>;      // bus-width, only SDIO3-2 supports 8 bits for emmc solution
     driving = <20 15 15 20 15 15 30 25 25 40 30 30>; /*Drving value of each mode*/
     cd_gpio = <D_GPIO(0) GPIO_FALLING_EDGE GPIO_POLLING>;
                              // Set card-detect pin and target value
                              // Set detection method as polling mode
                              // Comment out if don't need
                              // set pin as 0 with GPIO_INTERRUPT if device is not removable
     Card_power_gpio = <P_GPIO(1) GPIO_LOW>;
                              // Set GPIO-controlled card power pin
                              // Set Enable state
                              // Comment out if don't need

     ro_gpio = <D_GPIO(1) GPIO_RISING_EDGE GPIO_INTERRUPT>;
```

```
                                         // Set wirte-protect pin and target value
                                          // Set detection method as interrupt mode
                                          // Comment out if don't need
};
```

- Adding more sdio channels

```
mmc@f0500000 {
     compatible = "nvt,nvt_mmc";
     reg = <0xf0500000 0x1000>;
     interrupts = <GIC_SPI 31 IRQ_TYPE_LEVEL_HIGH>;
     max-frequency = <48000000>; //Set maximum clock frequency
     voltage-switch = <0>; //Set 1 if SDIO supports SDR50/SDR104 voltage switch (3.3V->1.8V)
     max-voltage = <3300>; //Set maximum voltage of your SDIO module required
     bus-width = <4>;       // bus-width, only SDIO3-2 supports 8 bits for emmc solution
     driving = <20 15 15 20 15 15 30 25 25 40 30 30>; /*Drving value of each mode*/
     cd_gpio = <0 GPIO_FALLING_EDGE GPIO_INTERRUPT >;
                                        // Set card-detect pin and target value
                                        // Set detection method as polling mode
                                        // Comment out if don't need
                                        // set pin as 0 with GPIO_INTERRUPT if device is not removable
     Card_power_gpio = <P_GPIO(1) GPIO_LOW>;
                                        // Set GPIO-controlled card power pin
                                        // Set Enable state
                                        // Comment out if don't need

     ro_gpio = <D_GPIO(1) GPIO_RISING_EDGE GPIO_INTERRUPT>;
                                        // Set wirte-protect pin and target value
                                        // Set detection method as interrupt mode
                                        // Comment out if don't need
};

mmc@f0510000 {
     compatible = "nvt,nvt_mmc";
```

```
        reg = <0xf0510000 0x1000>;
        interrupts = <GIC_SPI 32 IRQ_TYPE_LEVEL_HIGH>;
        max-frequency = <48000000>; //Set maximum clock frequency
        voltage-switch = <0>; //Set 1 if SDIO supports SDR50/SDR104 voltage switch (3.3V->1.8V)
        max-voltage = <3300>; //Set maximum voltage of your SDIO module required
        bus-width = <4>;       // bus-width, only SDIO3-2 supports 8 bits for emmc solution
        cd_gpio = <0 GPIO_FALLING_EDGE GPIO_INTERRUPT>;
                                     // Set card-detect pin and target value
                                     // Set detection method as polling mode
                                     // Comment out if don't need
                                     // set pin as 0 with GPIO_INTERRUPT if device is not removable
        Card_power_gpio = <P_GPIO(1) GPIO_LOW>;
                                     // Set GPIO-controlled card power pin
                                     // Set Enable state
                                     // Comment out if don't need

        ro_gpio = <D_GPIO(1) GPIO_RISING_EDGE GPIO_INTERRUPT>;
                                     // Set wirte-protect pin and target value
                                     // Set detection method as interrupt mode
                                     // Comment out if don't need
    };
```

3.5. Configurate SDIO clock/command/data pin drivings

- General Description

As different PCB requires different driving configuration, there are options for adjusting pin driving with corresponding SDIO modes. Please verify and adjust value with HW suggestions The configuration of driving values are located at *peri.dts file.

```
        driving = <20 15 15 20 15 15 30 25 25 40 30 30>; /*Drving value of each mode*/
```

The driving values of corresponding definition are:

<DS_Mode_Data	DS_Mode_Cmd	DS_Mode_Clock
HS_Mode_Data	HS_Mode_Cmd	HS_Mode_Clock
SDR50_Mode_Data	SDR50_Mode_Cmd	SDR50_Mode_Clock
SDR104_Mode_Data	SDR104_Mode_Cmd	SDR104_Mode_Clock>

The configurable options: 5/10/15/20/25/30/35/40 mA

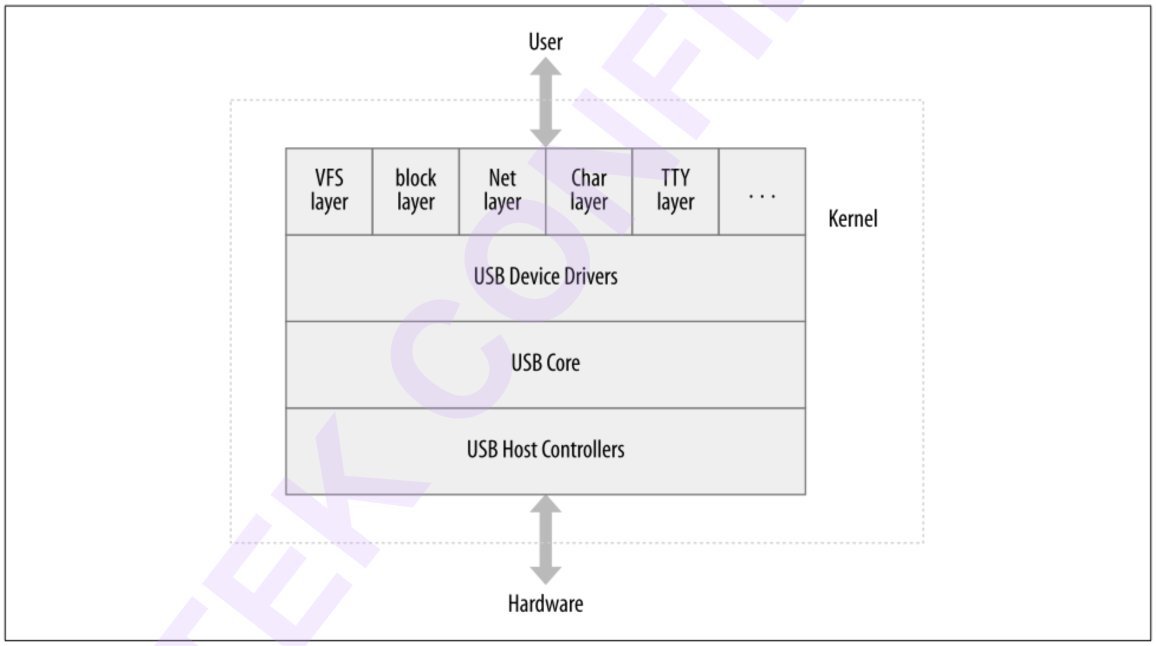
3.6. Troubleshooting

4. USB-Host

4.1. Feature

- Standard XHCI USB Host interface driver for Linux USB Core framework.(USB 3.0 Port)
- Standard EHCI USB Host interface driver for Linux USB Core framework.(USB 2.0 Port)
- USB 3.0 Port can also act as Device Mode.
- USB 3.0 Port is Host/Device function switchable.
- Support Control / Bulk / Interrupt / Isochronous transfer types.

4.2. Block Diagram



4.3. Header File

The header file is located in \Include\linux\usb.h

Place #include <linux/usb.h> in your source code for USB Host module.

4.4. Function Description

The usb host/device driver is common framework in the linux. The usage of the usb driver can reference to the textbook “Linux Device Driver 3e“, Chapter 13 for reference.

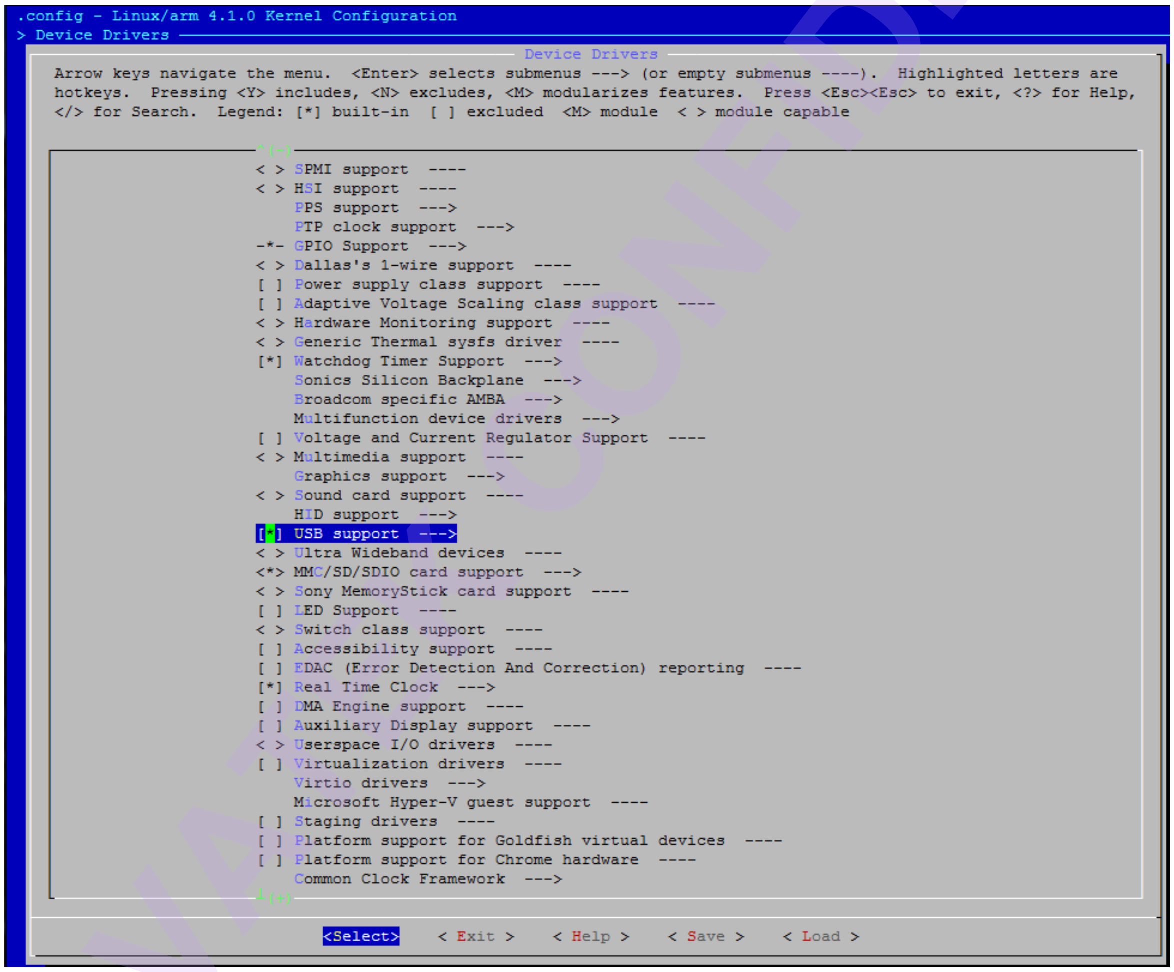
The NT9668x has two USB controllers, One is USB 3.0 Port which can act as Host/Device Dual Role switchable usb port. And the other is USB 2.0 Port which can act as Host function in the linux in current driver design. The USB 2.0 Host Port is not support for HUB function.

In the following sections, we will show the steps for using NT9668x Linux USB Host Controller for Mass-Storage application in Linux by using NT9668x.

4.4.1. Step-2: Make sure linux configurations

Please make sure the following configurations are correct:

Device Drivers -> USB support = y



```
.config - Linux/arm 4.1.0 Kernel Configuration
> Device Drivers ─────────────────────────────────────────────────────────────
                              Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help,
  </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                      < > SPMI support  ----
                      < > HSI support  ----
                          PPS support  --->
                          PTP clock support  --->
                      -*- GPIO Support  --->
                      < > Dallas's 1-wire support  ----
                      [ ] Power supply class support  ----
                      [ ] Adaptive Voltage Scaling class support  ----
                      < > Hardware Monitoring support  ----
                      < > Generic Thermal sysfs driver  ----
                      [*] Watchdog Timer Support  --->
                          Sonics Silicon Backplane  --->
                          Broadcom specific AMBA  --->
                          Multifunction device drivers  --->
                      [ ] Voltage and Current Regulator Support  ----
                      < > Multimedia support  ----
                          Graphics support  --->
                      < > Sound card support  ----
                          HID support  --->
                      [*] USB support  --->
                      < > Ultra Wideband devices  ----
                      <*> MMC/SD/SDIO card support  --->
                      < > Sony MemoryStick card support  ----
                      [ ] LED Support  ----
                      < > Switch class support  ----
                      [ ] Accessibility support  ----
                      [ ] EDAC (Error Detection And Correction) reporting  ----
                      [*] Real Time Clock  --->
                      [ ] DMA Engine support  ----
                      [ ] Auxiliary Display support  ----
                      < > Userspace I/O drivers  ----
                      [ ] Virtualization drivers  ----
                          Virtio drivers  --->
                          Microsoft Hyper-V guest support  ----
                      [ ] Staging drivers  ----
                      [ ] Platform support for Goldfish virtual devices  ----
                      [ ] Platform support for Chrome hardware  ----
                          Common Clock Framework  --->

                 <Select>    < Exit >    < Help >    < Save >    < Load >
```

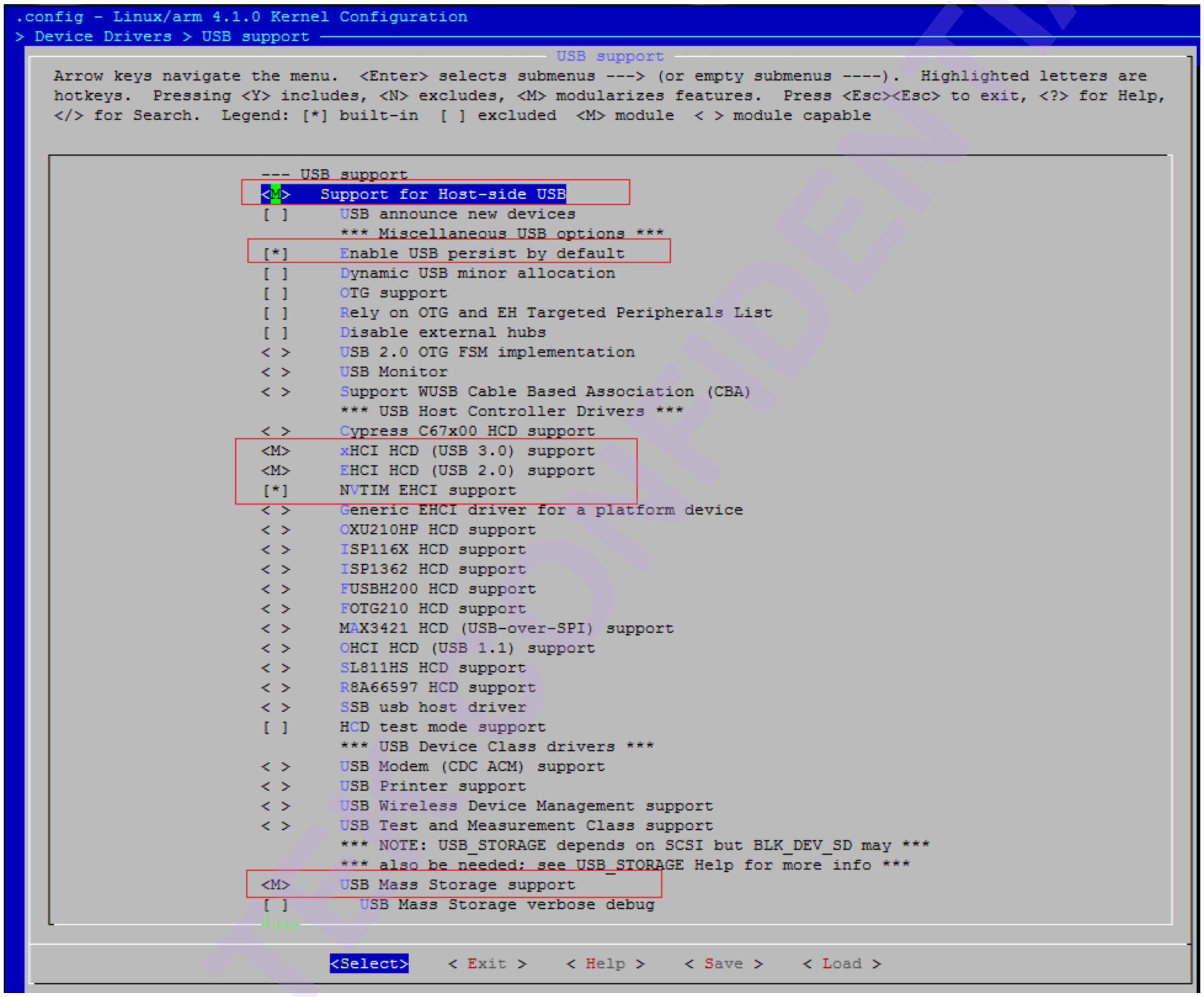
Device Drivers -> USB support -> Support for Host-side USB = m

Device Drivers -> USB support -> xHCI HCD (USB 3.0) support = m

Device Drivers -> USB support -> EHCI HCD (USB 2.0) support = m

Device Drivers -> USB support -> NVTIM EHCI support = y

Device Drivers -> USB support -> USB Mass Storage support = m



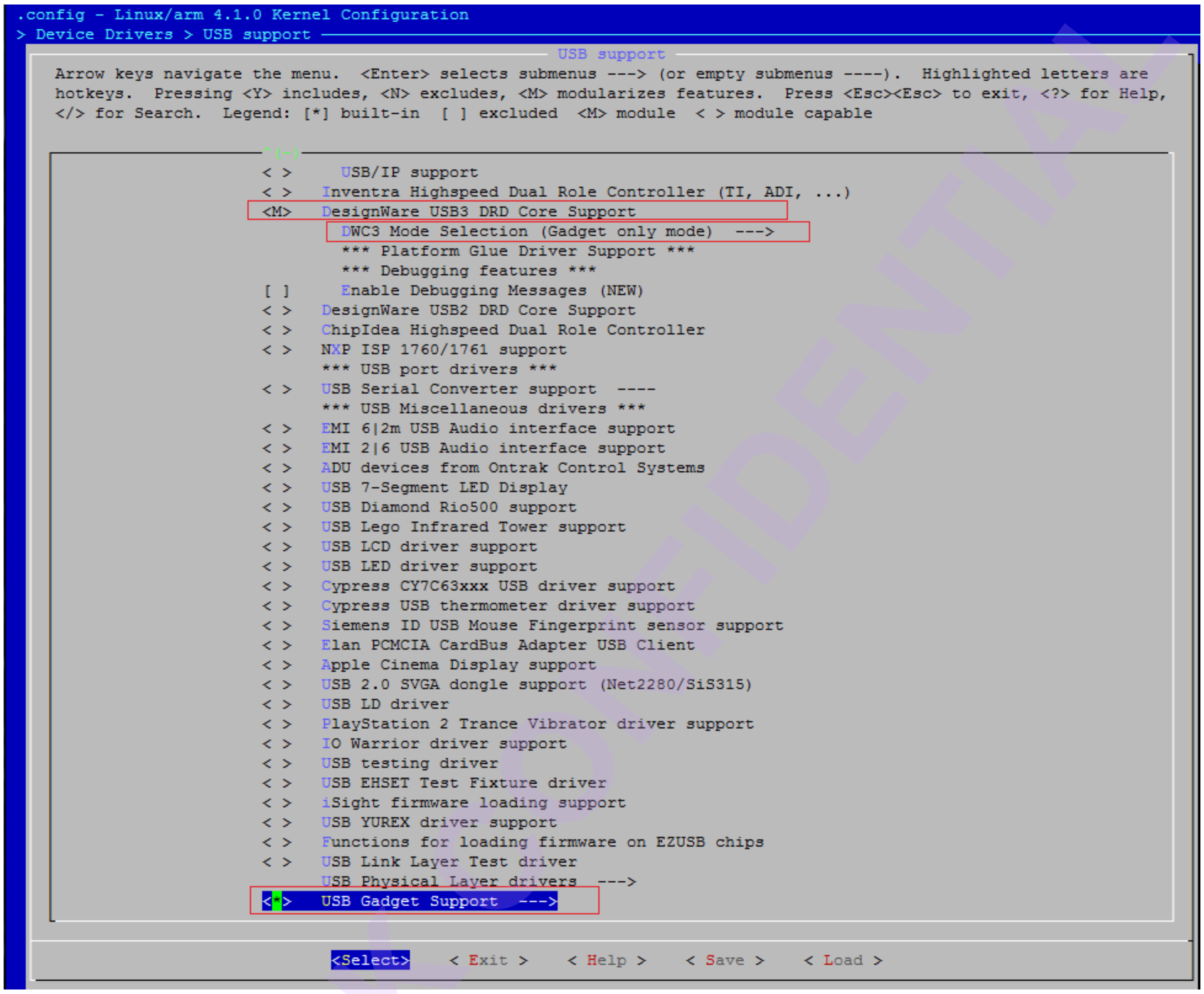
```
.config - Linux/arm 4.1.0 Kernel Configuration
> Device Drivers > USB support ─────────────────────────────────────
                              USB support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help,
  </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                    --- USB support
                    <M>   Support for Host-side USB
                    [ ]     USB announce new devices
                            *** Miscellaneous USB options ***
                    [*]     Enable USB persist by default
                    [ ]     Dynamic USB minor allocation
                    [ ]     OTG support
                    [ ]     Rely on OTG and EH Targeted Peripherals List
                    [ ]     Disable external hubs
                    < >     USB 2.0 OTG FSM implementation
                    < >     USB Monitor
                    < >     Support WUSB Cable Based Association (CBA)
                            *** USB Host Controller Drivers ***
                    < >     Cypress C67x00 HCD support
                    <M>     xHCI HCD (USB 3.0) support
                    <M>     EHCI HCD (USB 2.0) support
                    [*]     NVTIM EHCI support
                    < >     Generic EHCI driver for a platform device
                    < >     OXU210HP HCD support
                    < >     ISP116X HCD support
                    < >     ISP1362 HCD support
                    < >     FUSBH200 HCD support
                    < >     FOTG210 HCD support
                    < >     MAX3421 HCD (USB-over-SPI) support
                    < >     OHCI HCD (USB 1.1) support
                    < >     SL811HS HCD support
                    < >     R8A66597 HCD support
                    < >     SSB usb host driver
                    [ ]     HCD test mode support
                            *** USB Device Class drivers ***
                    < >     USB Modem (CDC ACM) support
                    < >     USB Printer support
                    < >     USB Wireless Device Management support
                    < >     USB Test and Measurement Class support
                            *** NOTE: USB_STORAGE depends on SCSI but BLK_DEV_SD may ***
                            *** also be needed; see USB_STORAGE Help for more info ***
                    <M>     USB Mass Storage support
                    [ ]       USB Mass Storage verbose debug
                    ↓(+)

             <Select>    < Exit >    < Help >    < Save >    < Load >
```

Device Drivers -> USB support -> USB Gadget Support = y

Device Drivers -> USB support -> DesignWare USB3 DRD Core Support = m

Device Drivers -> USB support -> **DWC3 Mode Selection =Gadget only mode**

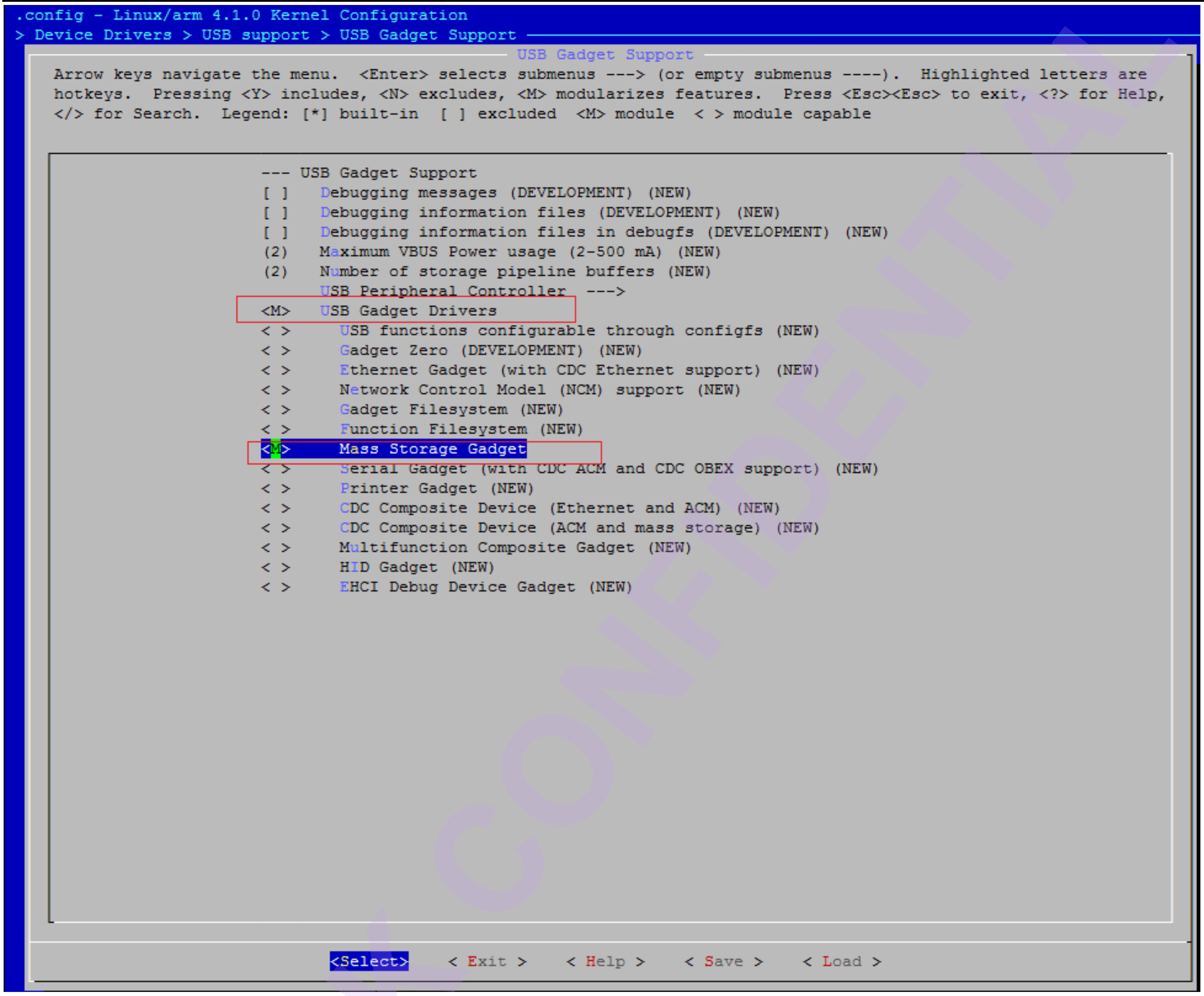
Device Drivers -> USB support -> USB Attached SCSI = m (Optional)



```
.config - Linux/arm 4.1.0 Kernel Configuration
> Device Drivers > USB support
                                  USB support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help,
  </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                    < >    USB/IP support
                    < >   Inventra Highspeed Dual Role Controller (TI, ADI, ...)
                    <M>   DesignWare USB3 DRD Core Support
                            DWC3 Mode Selection (Gadget only mode)  --->
                            *** Platform Glue Driver Support ***
                            *** Debugging features ***
                    [ ]     Enable Debugging Messages (NEW)
                    < >   DesignWare USB2 DRD Core Support
                    < >   ChipIdea Highspeed Dual Role Controller
                    < >   NXP ISP 1760/1761 support
                          *** USB port drivers ***
                    < >   USB Serial Converter support  ----
                          *** USB Miscellaneous drivers ***
                    < >   EMI 6|2m USB Audio interface support
                    < >   EMI 2|6 USB Audio interface support
                    < >   ADU devices from Ontrak Control Systems
                    < >   USB 7-Segment LED Display
                    < >   USB Diamond Rio500 support
                    < >   USB Lego Infrared Tower support
                    < >   USB LCD driver support
                    < >   USB LED driver support
                    < >   Cypress CY7C63xxx USB driver support
                    < >   Cypress USB thermometer driver support
                    < >   Siemens ID USB Mouse Fingerprint sensor support
                    < >   Elan PCMCIA CardBus Adapter USB Client
                    < >   Apple Cinema Display support
                    < >   USB 2.0 SVGA dongle support (Net2280/SiS315)
                    < >   USB LD driver
                    < >   PlayStation 2 Trance Vibrator driver support
                    < >   IO Warrior driver support
                    < >   USB testing driver
                    < >   USB EHSET Test Fixture driver
                    < >   iSight firmware loading support
                    < >   USB YUREX driver support
                    < >   Functions for loading firmware on EZUSB chips
                    < >   USB Link Layer Test driver
                          USB Physical Layer drivers  --->
                    <*>   USB Gadget Support  --->

                 <Select>    < Exit >    < Help >    < Save >    < Load >
```

Device Drivers -> USB support -> USB Gadget Support -> USB Gadget Drivers = m

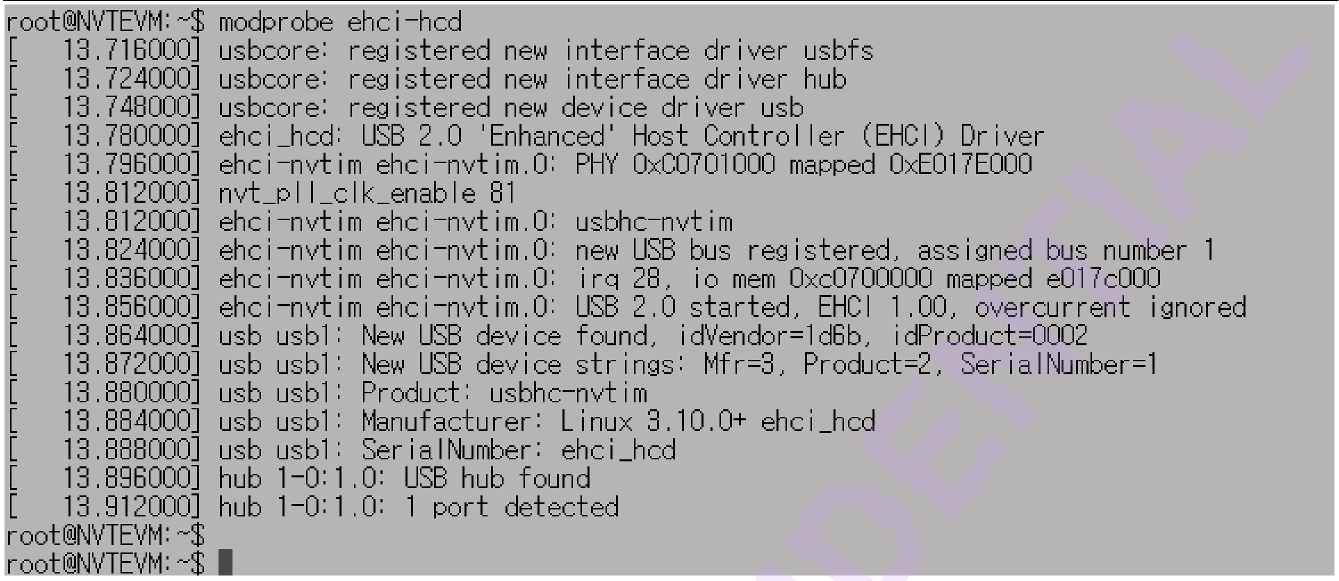
Device Drivers -> USB support -> USB Gadget Support -> Mass Storage Gadget = m



```
.config - Linux/arm 4.1.0 Kernel Configuration
> Device Drivers > USB support > USB Gadget Support ──────────────────────────
                                 USB Gadget Support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help,
  </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                    --- USB Gadget Support
                    [ ]   Debugging messages (DEVELOPMENT) (NEW)
                    [ ]   Debugging information files (DEVELOPMENT) (NEW)
                    [ ]   Debugging information files in debugfs (DEVELOPMENT) (NEW)
                    (2)   Maximum VBUS Power usage (2-500 mA) (NEW)
                    (2)   Number of storage pipeline buffers (NEW)
                          USB Peripheral Controller  --->
                    <M>   USB Gadget Drivers
                    < >     USB functions configurable through configfs (NEW)
                    < >     Gadget Zero (DEVELOPMENT) (NEW)
                    < >     Ethernet Gadget (with CDC Ethernet support) (NEW)
                    < >     Network Control Model (NCM) support (NEW)
                    < >     Gadget Filesystem (NEW)
                    < >     Function Filesystem (NEW)
                    <M>     Mass Storage Gadget
                    < >     Serial Gadget (with CDC ACM and CDC OBEX support) (NEW)
                    < >     Printer Gadget (NEW)
                    < >     CDC Composite Device (Ethernet and ACM) (NEW)
                    < >     CDC Composite Device (ACM and mass storage) (NEW)
                    < >     Multifunction Composite Gadget (NEW)
                    < >     HID Gadget (NEW)
                    < >     EHCI Debug Device Gadget (NEW)

                  <Select>    < Exit >    < Help >    < Save >    < Load >
```

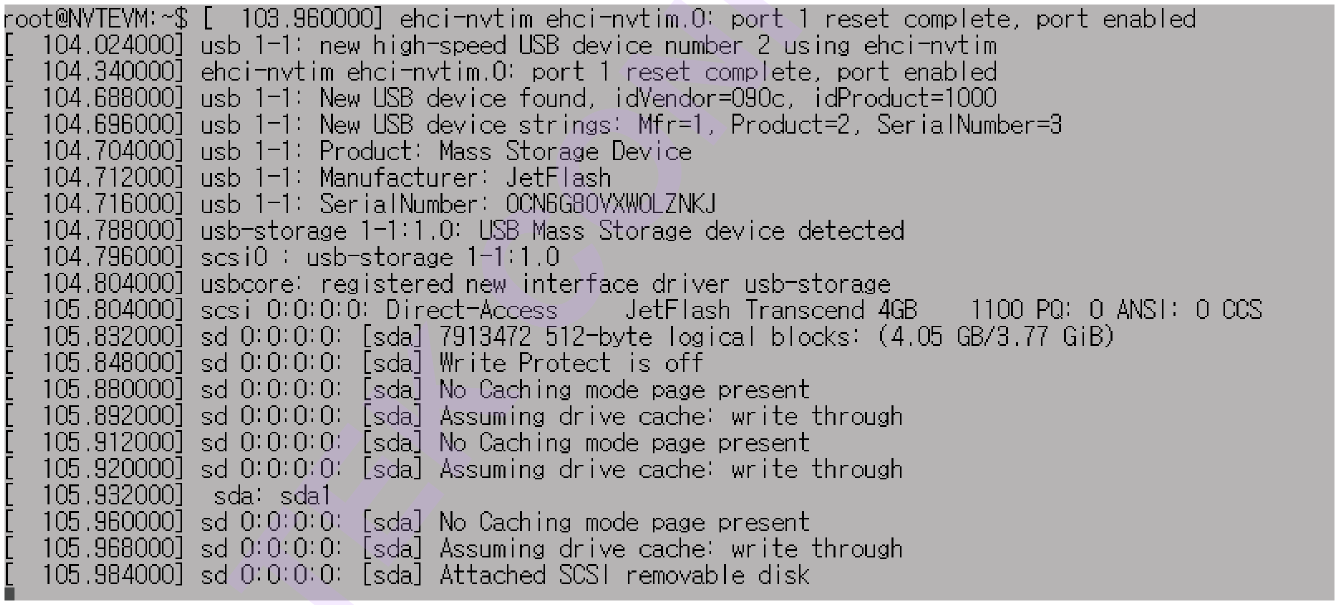
4.4.2. Step-5: Load USB driver after entering linux OS

After entering linux OS, enter "modprobe ehci-hcd" to load usb 2.0 port host driver or "modprobe xhci_plat_hcd" to load usb3.0 port host driver.



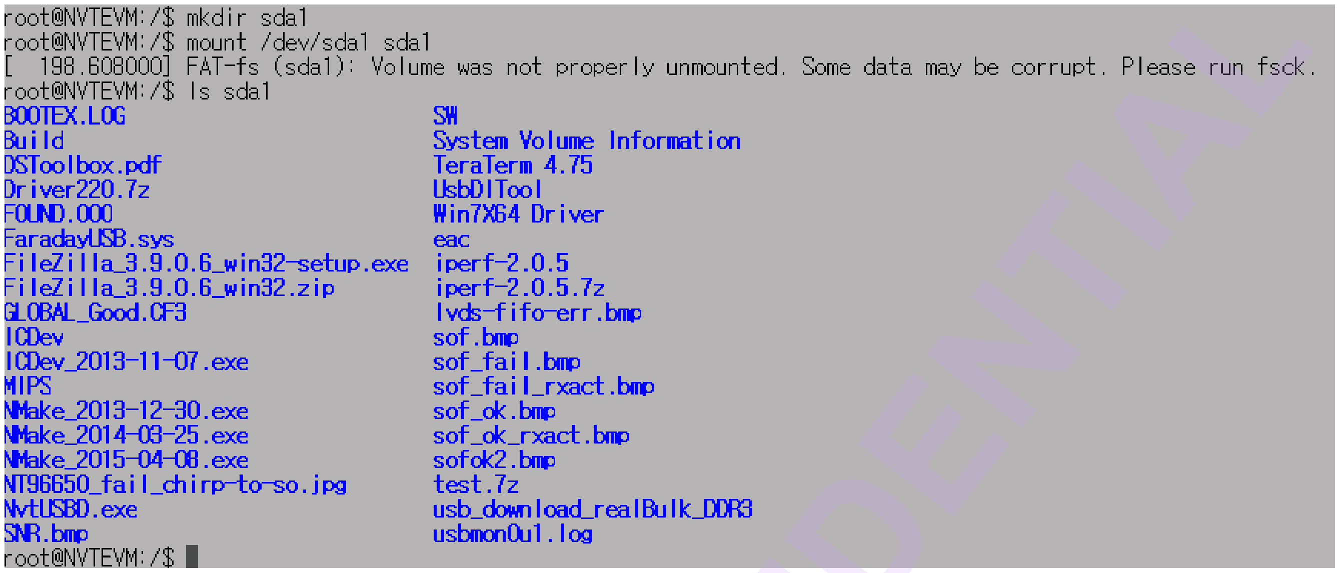
```
root@NVTEVM:~$ modprobe ehci-hcd
[   13.716000] usbcore: registered new interface driver usbfs
[   13.724000] usbcore: registered new interface driver hub
[   13.748000] usbcore: registered new device driver usb
[   13.780000] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[   13.796000] ehci-nvtim ehci-nvtim.0: PHY 0xC0701000 mapped 0xE017E000
[   13.812000] nvt_pll_clk_enable 81
[   13.812000] ehci-nvtim ehci-nvtim.0: usbhc-nvtim
[   13.824000] ehci-nvtim ehci-nvtim.0: new USB bus registered, assigned bus number 1
[   13.836000] ehci-nvtim ehci-nvtim.0: irq 28, io mem 0xc0700000 mapped e017c000
[   13.856000] ehci-nvtim ehci-nvtim.0: USB 2.0 started, EHCI 1.00, overcurrent ignored
[   13.864000] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
[   13.872000] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[   13.880000] usb usb1: Product: usbhc-nvtim
[   13.884000] usb usb1: Manufacturer: Linux 3.10.0+ ehci_hcd
[   13.888000] usb usb1: SerialNumber: ehci_hcd
[   13.896000] hub 1-0:1.0: USB hub found
[   13.912000] hub 1-0:1.0: 1 port detected
root@NVTEVM:~$
root@NVTEVM:~$
```

And then plug the usb disk to run the mass storage application.



```
root@NVTEVM:~$ [  103.960000] ehci-nvtim ehci-nvtim.0: port 1 reset complete, port enabled
[  104.024000] usb 1-1: new high-speed USB device number 2 using ehci-nvtim
[  104.340000] ehci-nvtim ehci-nvtim.0: port 1 reset complete, port enabled
[  104.688000] usb 1-1: New USB device found, idVendor=090c, idProduct=1000
[  104.696000] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  104.704000] usb 1-1: Product: Mass Storage Device
[  104.712000] usb 1-1: Manufacturer: JetFlash
[  104.716000] usb 1-1: SerialNumber: OCN6G8OVXWOLZNKJ
[  104.788000] usb-storage 1-1:1.0: USB Mass Storage device detected
[  104.796000] scsi0 : usb-storage 1-1:1.0
[  104.804000] usbcore: registered new interface driver usb-storage
[  105.804000] scsi 0:0:0:0: Direct-Access     JetFlash Transcend 4GB    1100 PQ: 0 ANSI: 0 CCS
[  105.832000] sd 0:0:0:0: [sda] 7913472 512-byte logical blocks: (4.05 GB/3.77 GiB)
[  105.848000] sd 0:0:0:0: [sda] Write Protect is off
[  105.880000] sd 0:0:0:0: [sda] No Caching mode page present
[  105.892000] sd 0:0:0:0: [sda] Assuming drive cache: write through
[  105.912000] sd 0:0:0:0: [sda] No Caching mode page present
[  105.920000] sd 0:0:0:0: [sda] Assuming drive cache: write through
[  105.932000]  sda: sda1
[  105.960000] sd 0:0:0:0: [sda] No Caching mode page present
[  105.968000] sd 0:0:0:0: [sda] Assuming drive cache: write through
[  105.984000] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

Test for the storage RW.



```
root@NVTEVM:/$ mkdir sda1
root@NVTEVM:/$ mount /dev/sda1 sda1
[  198.608000] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
root@NVTEVM:/$ ls sda1
BOOTEX.LOG                           SW
Build                                System Volume Information
DSToolbox.pdf                        TeraTerm 4.75
Driver220.7z                         UsbDITool
FOUND.000                            Win7X64 Driver
FaradayUSB.sys                       eac
FileZilla_3.9.0.6_win32-setup.exe    iperf-2.0.5
FileZilla_3.9.0.6_win32.zip          iperf-2.0.5.7z
GLOBAL_Good.CF3                      lvds-fifo-err.bmp
ICDev                                sof.bmp
ICDev_2013-11-07.exe                 sof_fail.bmp
MIPS                                 sof_fail_rxact.bmp
NMake_2013-12-30.exe                 sof_ok.bmp
NMake_2014-03-25.exe                 sof_ok_rxact.bmp
NMake_2015-04-08.exe                 sofok2.bmp
NT96650_fail_chirp-to-so.jpg         test.7z
NvtUSBD.exe                          usb_download_realBulk_DDR3
SNR.bmp                              usbmon0u1.log
root@NVTEVM:/$ 
```

If user want to plug USB3.0 Device Mode MSDC to Host PC, just need to enter:

```
modprobe dwc3
modprobe configfs
modprobe libcomposite
modprobe usb_f_mass_storage
modprobe g_mass_storage iSerialNumber=123456 file=/dev/sda1 stall=0 removable=1
```

The file /dev/sda1 shall be prepared your own. This can be SDCARD storage or any.

If the user want to act as OTG device to switch Host/Device, you shall remove the Host/Device module driver first and then probe the Device/Host module driver.

4.5. Troubleshooting

5. SPI NAND/NOR MTD

5.1. Feature

- Encapsulated as a MTD device
- Operating frequency can be 500 KHz ~ 120 MHz
- Support SPI-NAND flash(ETRON/MXIC/Winbond/GD…)
- Support SPI-NOR flash

5.2. Block diagram

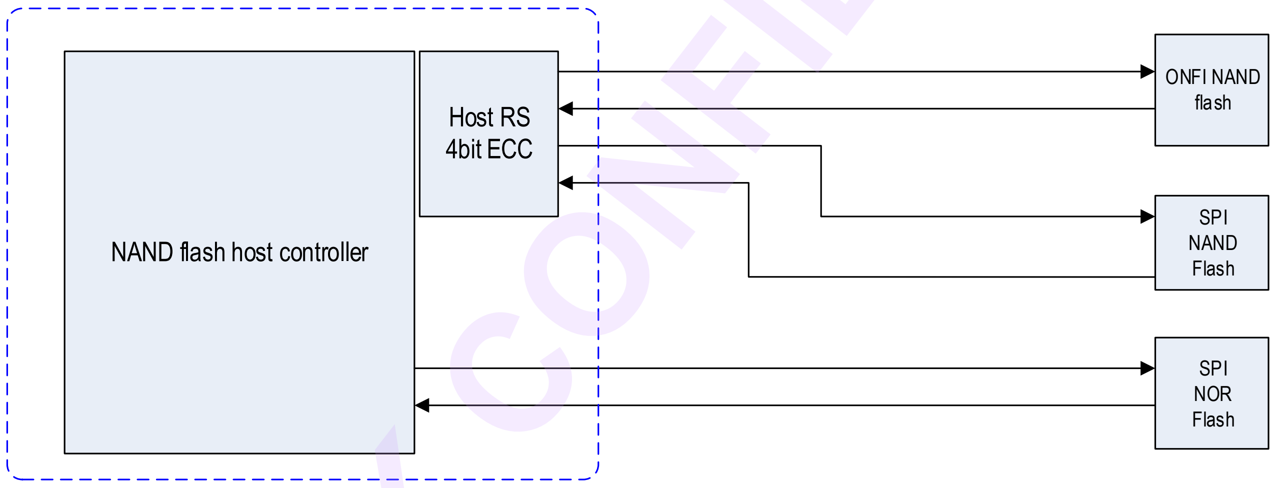


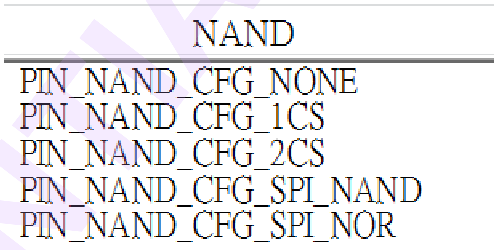
Figure 5-1 NAND (Flash) controller diagram

5.3. Function description

NT9668x SPI NAND/NOR is encapsulated as a MTD device driver. It can be partitioned to store u-Boot, Linux kernel, UBI FS, etc…

5.3.1. All the related pinmux configuration shall be setup with pinctrl tool

- ☐ Location: build/nvt-tools/nvt_pinctrl_tool/top_generator.xlsm
- ☐ Please refer more details in NT96680_TOOL_UI_pinctrl_tool



NAND
PIN_NAND_CFG_NONE
PIN_NAND_CFG_1CS
PIN_NAND_CFG_2CS
PIN_NAND_CFG_SPI_NAND
PIN_NAND_CFG_SPI_NOR

5.3.2. Probing SPI NAND/NOR platform driver

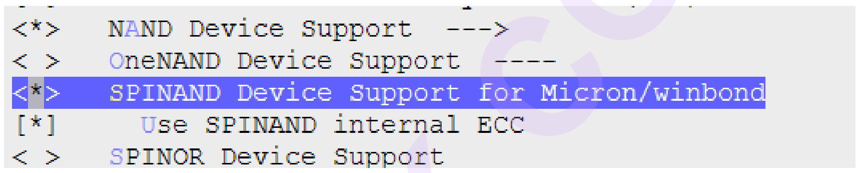
● General description

NT9668x SPI NAND/NOR MTD driver is encapsulated as a platform driver. You should enable module by selecting it in Linux menuconfig to probe SPI NAND/NOR MTD.

SPI NAND

CONFIG_MTD_SPINAND=y

CONFIG_MTD_SPINAND_ONDIEECC=y



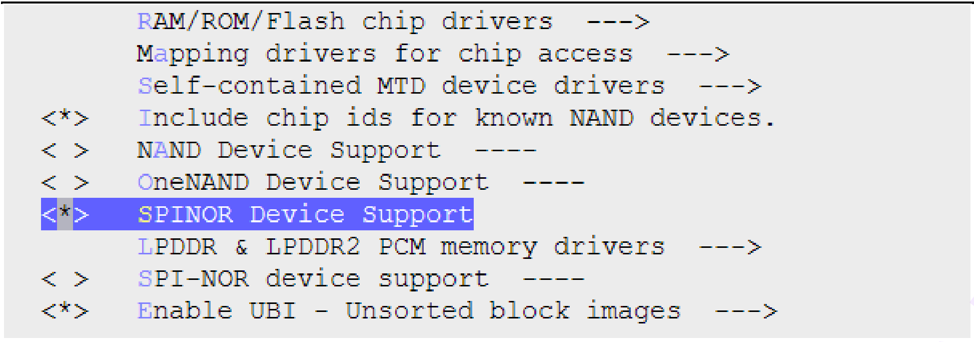
```
<*>   NAND Device Support  --->
< >   OneNAND Device Support  ----
<*>   SPINAND Device Support for Micron/winbond
[*]     Use SPINAND internal ECC
< >   SPINOR Device Support
```

SPI NOR:

# CONFIG_MTD_NAND is not set

# CONFIG_MTD_ONENAND is not set

CONFIG_MTD_SPINOR=y



```
        RAM/ROM/Flash chip drivers  --->
        Mapping drivers for chip access  --->
        Self-contained MTD device drivers  --->
  <*>   Include chip ids for known NAND devices.
  < >   NAND Device Support  ----
  < >   OneNAND Device Support  ----
  <*>   SPINOR Device Support
        LPDDR & LPDDR2 PCM memory drivers  --->
  < >   SPI-NOR device support  ----
  <*>   Enable UBI - Unsorted block images  --->
```

5.3.3. Partition SPI NAND/NOR MTD

- General description

You can use Linux kernel command line “mtdparts” to configure MTD partitions. Format of this command is:

```
mtdparts=<mtddef>[;<mtddef]

<mtddef>  := <mtd-id>:<partdef>[,<partdef>]

<partdef> := <size>[@offset][<name>][ro]

<mtd-id>  := unique id used in mapping driver/device (number of flash bank)

<size>    := standard linux memsize OR "-" to denote all remaining space

<name>    := '(' NAME ')'
```

- Limitations
  - ☐ <mtd-id> should be “spi_nand.0” or “spi_nor.0”

- Flow

In NT9668x Linux platform, the kernel command is coded in u-Boot and not recommends users to modify it. If you have requirements to define your own partition table, please find Novatek contact window to discuss it.

5.4. Using New Spi-NAND flash

- Fill up table in u-boot

BSP\u-boot\drivers\mtd\nand\nand_ids.c

```
 const struct nand_flash_dev nvt_nand_ids[] = {
   // name             id/pagesize(unit byte)/chip size(unit MB)/erasesize(unit byte)/option
   {"NAND 256MiB 3,3V 8-bit",    0xDA, 2048, 256, 0x20000,     NAND_NO_SUBPAGE_WRITE},
   {"SPI-NAND 1GiB 3,3V 8-bit",  0xD1, 2048, 128, 0x20000,     NAND_NO_SUBPAGE_WRITE},
   {"SPI-NAND 128MiB 3V",        0xAA, 2048, 128, 0x20000,     NAND_NO_SUBPAGE_WRITE},
   {"SPI-NAND 128MiB 3V",        0x12, 2048, 128, 0x20000,     NAND_NO_SUBPAGE_WRITE},
   {"SPI-NAND 128MiB 3V ",       0xC2, 2048, 128, 0x20000,     NAND_NO_SUBPAGE_WRITE},
   {"SPI-NAND 128MiB 3V ",       0x01, 2048, 128, 0x20000,     NAND_NO_SUBPAGE_WRITE},
};
```

- Fill up table in linux

BSP\linux-kernel\drivers\mtd\spiflash\spinand.c

```
 static struct nand_flash_dev spinand_flash_ids[] = {
/*
* Some incompatible NAND chips share device ID's and so must be
* listed by full ID. We list them first so that we can easily identify
* the most specific match.
*/
   // GigaDevice
   {"GD5F1GQ4UBYIG 1GiB 3.3V",
        { .id = {MFG_ID_GD, 0xD1} },
        SZ_2K, SZ_128, SZ_128K, 0, 2, 128 },
   // MXIC
   {"MX35LF1GE4AB 1GiB 3.3V",
        { .id = {MFG_ID_MXIC, 0x12} },
        SZ_2K, SZ_128, SZ_128K, 0, 2, 64 },
};
```

5.5. Troubleshooting

6. SPI

6.1. Feature

- Linux SPI Master controller interface
- SPI frequency can be 100 KHz ~ 48 MHz
- Only SPI1 supports DMA mode

6.2. Block diagram

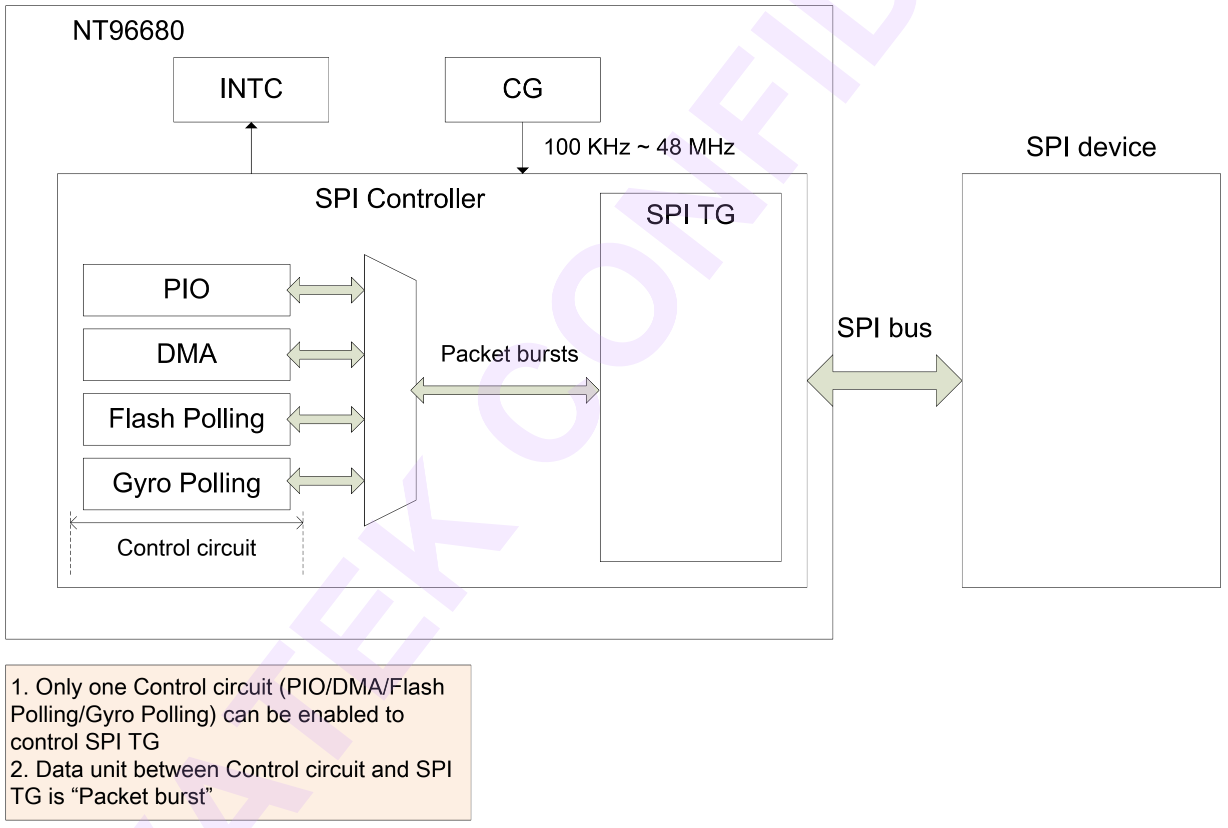

NT96680
INTC
CG
100 KHz ~ 48 MHz
SPI device
SPI Controller
SPI TG
PIO
DMA
Flash Polling
Gyro Polling
Packet bursts
SPI bus
Control circuit
1. Only one Control circuit (PIO/DMA/Flash Polling/Gyro Polling) can be enabled to control SPI TG
2. Data unit between Control circuit and SPI TG is "Packet burst"

Figure 6-1 SPI diagram

6.3. Header file

The header file is located in /include/linux/spi.h.

Place #include <linux/spi/spi.h> in your SPI protocol driver.

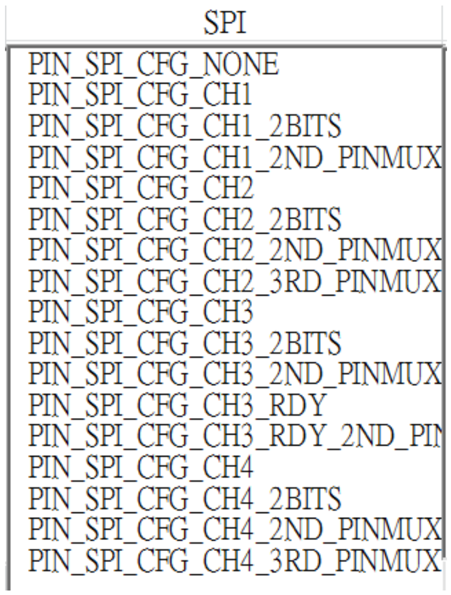
6.4. Function description

In Linux, there are two types of SPI driver: SPI controller driver and SPI protocol driver. SPI controller driver is provided by Novatek to abstract details of SPI h/w controller. SPI protocol driver should be written by NT9668x users/customers to communicate with their SPI slave devices.

This document is intended to address on how to link NT9668x SPI controller driver to your own SPI protocol driver. As to details of SPI protocol driver, it's the freedom of customer and can refer to Linux kernel SPI document.

6.4.1. All the related pinmux configuration shall be setup with pinctrl tool

- ☐ Location: build/nvt-tools/nvt_pinctrl_tool/top_generator.xlsm
- ☐ Please refer more details in NT96680_TOOL_UI_pinctrl_tool

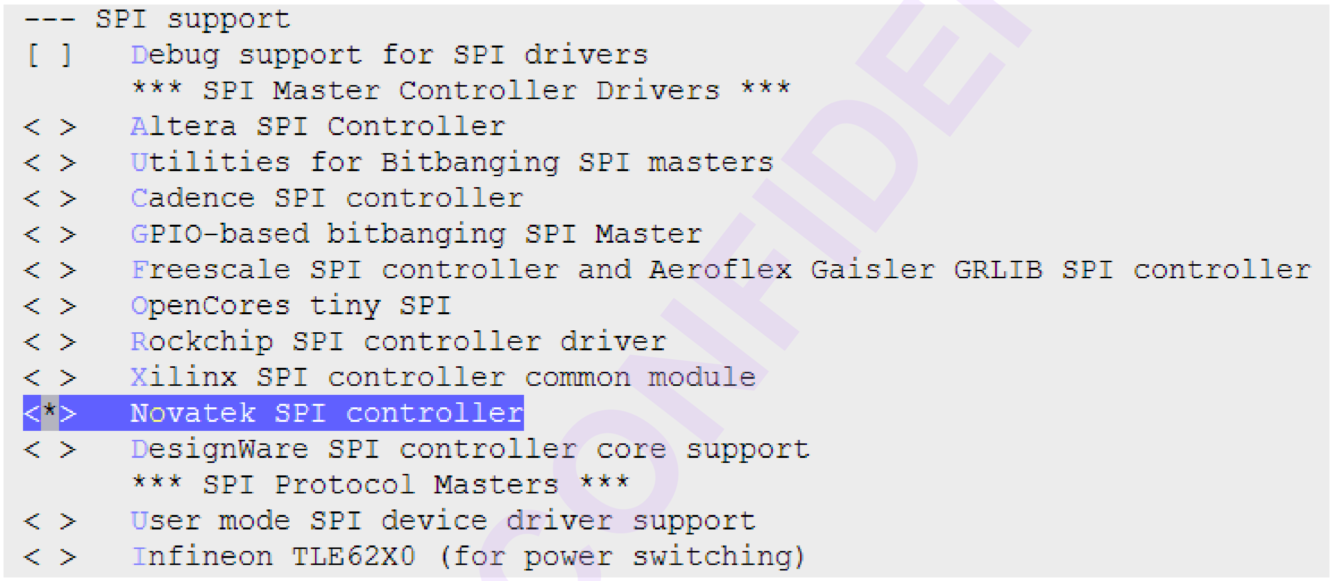

SPI
PIN_SPI_CFG_NONE
PIN_SPI_CFG_CH1
PIN_SPI_CFG_CH1_2BITS
PIN_SPI_CFG_CH1_2ND_PINMUX
PIN_SPI_CFG_CH2
PIN_SPI_CFG_CH2_2BITS
PIN_SPI_CFG_CH2_2ND_PINMUX
PIN_SPI_CFG_CH2_3RD_PINMUX
PIN_SPI_CFG_CH3
PIN_SPI_CFG_CH3_2BITS
PIN_SPI_CFG_CH3_2ND_PINMUX
PIN_SPI_CFG_CH3_RDY
PIN_SPI_CFG_CH3_RDY_2ND_PIN
PIN_SPI_CFG_CH4
PIN_SPI_CFG_CH4_2BITS
PIN_SPI_CFG_CH4_2ND_PINMUX
PIN_SPI_CFG_CH4_3RD_PINMUX

6.4.2. Probing SPI controller driver

- General description

NT9668x SPI controller driver is encapsulated as a platform driver. Please enable module by selecting Novatek SPI controller in linux menuconfig.

```
#
# SPI Master Controller Drivers
#
# CONFIG_SPI_ALTERA is not set
# CONFIG_SPI_BITBANG  is not set
# CONFIG_SPI_CADENCE  is not set
# CONFIG_SPI_GPIO  is not set
# CONFIG_SPI_FSL_SPI is not set
# CONFIG_SPI_OC_TINY  is not set
# CONFIG_SPI_PXA2XX_PCI  is not set
# CONFIG_SPI_ROCKCHIP  is not set
# CONFIG_SPI_XILINX  is not set
CONFIG_SPI_NA51000=y
# CONFIG_SPI_DESIGNWARE  is not set
```



```
--- SPI support
[ ]   Debug support for SPI drivers
      *** SPI Master Controller Drivers ***
< >   Altera SPI Controller
< >   Utilities for Bitbanging SPI masters
< >   Cadence SPI controller
< >   GPIO-based bitbanging SPI Master
< >   Freescale SPI controller and Aeroflex Gaisler GRLIB SPI controller
< >   OpenCores tiny SPI
< >   Rockchip SPI controller driver
< >   Xilinx SPI controller common module
<*>   Novatek SPI controller
< >   DesignWare SPI controller core support
      *** SPI Protocol Masters ***
< >   User mode SPI device driver support
< >   Infineon TLE62X0 (for power switching)
```

6.4.3. Write your SPI protocol driver

- General description

Details of writing SPI protocol driver is out of scope. For kernel space SPI protocol driver, please refer to spi-summary. For userspace drivers, please refer to spidev.

This document still provides some hints on writing kernel space SPI protocol driver:

3. Declare a spi_driver object
4. Register your spi_driver object under spi node in device tree

- Flow (for kernel space driver)

  An example to add a spi-nor device driver in nvt-na51000-peri.dts

```
spi@f0320000 {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "nvt,nvt_spi";
        reg = <0xf0320000 0x10000>;
        interrupts = <GIC_SPI 36 IRQ_TYPE_LEVEL_HIGH>;
        dma-support = <0>;
        flash: m25p80@0 {
             #address-cells = <1>;
             #size-cells = <1>;
             compatible = "winbond,w25q32", "jedec,spi-nor";
             reg = <0>;
             spi-max-frequency = <40000000>;
             m25p,fast-read;
        };
 };
```

6.4.4. Introduce other SPI buses in Linux

Example to configure all of SPI buses in Linux platform with the following steps

Register bus in device tree

```
spi@f0230000 {                                      //spi register base address
    compatible = "nvt,nvt_spi";
    reg = <0xf0230000 0x100>;                      //spi register base address + offset
    interrupts = <GIC_SPI 35 IRQ_TYPE_LEVEL_HIGH>;  //spi interrupt
    dma-support = <0>;                              //spi dma mode, set to 1 to enable DMA transfer
    nvt-devname = <0>;
};
spi@f0320000 {                                      //spi register base address
    compatible = "nvt,nvt_spi";
    reg = <0xf0320000 0x100>;                      //spi register base address + offset
    interrupts = <GIC_SPI 36 IRQ_TYPE_LEVEL_HIGH>;   //spi interrupt
    dma-support = <0>;                               //spi dma mode, not support on SPI2.
    nvt-devname = <1>;
};

spi@f0340000 {                                        //spi register base address
    compatible = "nvt,nvt_spi";
    reg = <0xf0340000 0x100>;                        //spi register base address + offset
    interrupts = <GIC_SPI 37 IRQ_TYPE_LEVEL_HIGH>;   //spi interrupt
    dma-support = <0>;                               //spi dma mode, not support on SPI3.
    nvt-devname = <2>;
};

spi@f0360000 {                                        //spi register base address
    compatible = "nvt,nvt_spi";
    reg = <0xf0360000 0x100>;                        //spi register base address + offset
    interrupts = <GIC_SPI 38 IRQ_TYPE_LEVEL_HIGH>;   //spi interrupt
    dma-support = <0>;                               //spi dma mode, not support on SPI4.
    nvt-devname = <3>;
};
```

1. Register clock in clock driver

BSP\linux-kernel\drivers\clk\novatek\na51000-clk.c

```
static struct nvt_composite_gate_clk na51000_cgate_clk[] __initdata = {
#ifdef CONFIG_USE_OF
…
…
COMP_GATE_CONF("f0230000.spi", "fix192m", 24000000,
CG_SPI_CLK_DIV_REG0_OFFSET, BIT0, WID11,
CG_CLK_EN_REG1_OFFSET, BIT6, NOT_ENABLE,
CG_SYS_RESET_REG1_OFFSET, BIT6, NOT_RESET,
0, 0, 0),
COMP_GATE_CONF("f0320000.spi", "fix192m", 24000000,
CG_SPI_CLK_DIV_REG0_OFFSET, BIT16, WID11,
CG_CLK_EN_REG1_OFFSET, BIT7, NOT_ENABLE,
CG_SYS_RESET_REG1_OFFSET, BIT7, NOT_RESET,
0, 0, 0),
COMP_GATE_CONF("f0340000.spi", "fix192m", 24000000,
CG_SPI_CLK_DIV_REG1_OFFSET, BIT0, WID11,
CG_CLK_EN_REG1_OFFSET, BIT8, NOT_ENABLE,
CG_SYS_RESET_REG1_OFFSET, BIT8, NOT_RESET,
0, 0, 0),
COMP_GATE_CONF("f0360000.spi", "fix192m", 24000000,
CG_SPI_CLK_DIV_REG1_OFFSET, BIT16, WID11,
CG_CLK_EN_REG1_OFFSET, BIT15, NOT_ENABLE,
CG_SYS_RESET_REG1_OFFSET, BIT15, NOT_RESET,
0, 0, 0),..
…
```

6.5. Troubleshooting

7. Ethernet

7.1. Feature

- Support 10/100 Mbps data transfer
- Connect external PHY with MII/RMII
- Support IP/TCP/UDP/ICMP checksum offload
- Support Linux PHY Abstraction Layer

7.2. Block diagram

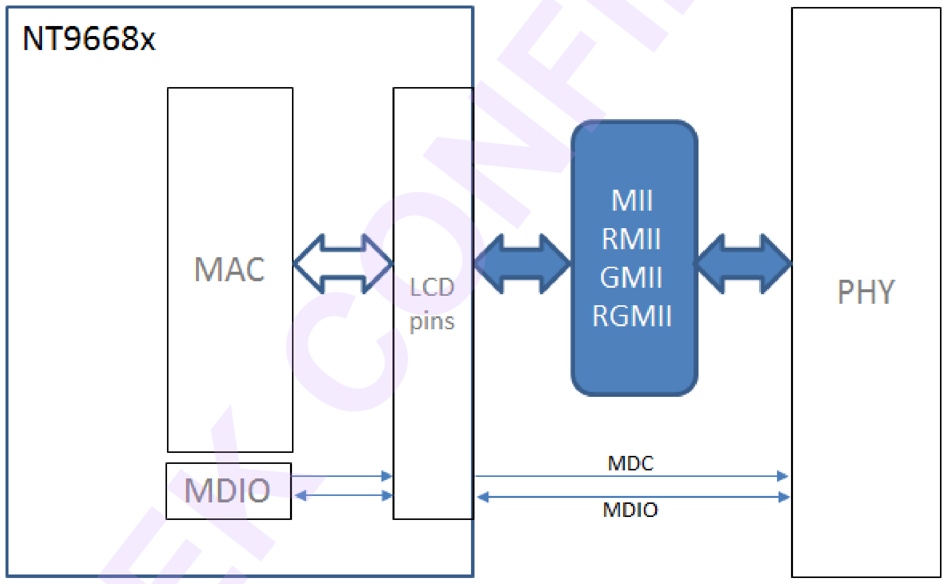
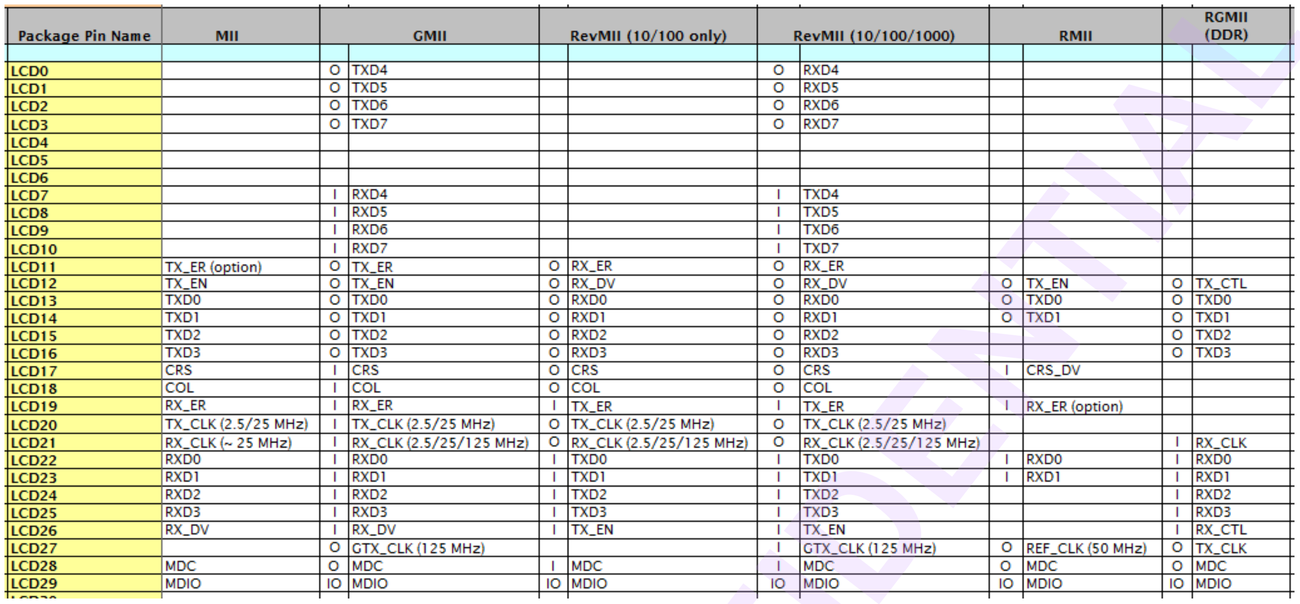


Figure 7-1 PHY interface



Package Pin Name	MII	GMII		RevMII (10/100 only)		RevMII (10/100/1000)		RMII		RGMII (DDR)	
LCD0		O	TXD4			O	RXD4				
LCD1		O	TXD5			O	RXD5				
LCD2		O	TXD6			O	RXD6				
LCD3		O	TXD7			O	RXD7				
LCD4											
LCD5											
LCD6											
LCD7		I	RXD4			I	TXD4				
LCD8		I	RXD5			I	TXD5				
LCD9		I	RXD6			I	TXD6				
LCD10		I	RXD7			I	TXD7				
LCD11	TX_ER (option)	O	TX_ER	O	RX_ER	O	RX_ER				
LCD12	TX_EN	O	TX_EN	O	RX_DV	O	RX_DV	O	TX_EN	O	TX_CTL
LCD13	TXD0	O	TXD0	O	RXD0	O	RXD0	O	TXD0	O	TXD0
LCD14	TXD1	O	TXD1	O	RXD1	O	RXD1	O	TXD1	O	TXD1
LCD15	TXD2	O	TXD2	O	RXD2	O	RXD2			O	TXD2
LCD16	TXD3	O	TXD3	O	RXD3	O	RXD3			O	TXD3
LCD17	CRS	I	CRS	O	CRS	O	CRS	I	CRS_DV		
LCD18	COL	I	COL	O	COL	O	COL				
LCD19	RX_ER	I	RX_ER	I	TX_ER	I	TX_ER	I	RX_ER (option)		
LCD20	TX_CLK (2.5/25 MHz)	I	TX_CLK (2.5/25 MHz)	O	TX_CLK (2.5/25 MHz)	O	TX_CLK (2.5/25 MHz)				
LCD21	RX_CLK (~ 25 MHz)	I	RX_CLK (2.5/25/125 MHz)	O	RX_CLK (2.5/25/125 MHz)	O	RX_CLK (2.5/25/125 MHz)			I	RX_CLK
LCD22	RXD0	I	RXD0	I	TXD0	I	TXD0	I	RXD0	I	RXD0
LCD23	RXD1	I	RXD1	I	TXD1	I	TXD1	I	RXD1	I	RXD1
LCD24	RXD2	I	RXD2	I	TXD2	I	TXD2			I	RXD2
LCD25	RXD3	I	RXD3	I	TXD3	I	TXD3			I	RXD3
LCD26	RX_DV	I	RX_DV	I	TX_EN	I	TX_EN			I	RX_CTL
LCD27		O	GTX_CLK (125 MHz)			I	GTX_CLK (125 MHz)	O	REF_CLK (50 MHz)	O	TX_CLK
LCD28	MDC	O	MDC	I	MDC	I	MDC	O	MDC	O	MDC
LCD29	MDIO	IO	MDIO	IO	MDIO	IO	MDIO	IO	MDIO	IO	MDIO

Figure 7-2 ethernet pinmux

7.3. Function description

NT9668x Ethernet driver is compatible to PHY Abstraction Layer. Normally it can work with all Ethernet PHY listed in /drivers/net/phy.

7.3.1. Outputting crystal clock to external PHY

- General description

Most Ethernet PHYs require additional crystal/oscillator to generate PHY's clock source. NT9668x provides SP_CLK (special clock) which can output clock source to PHY for replacing crystal/oscillator. If your board is designed to use SP_CLK as PHY's clock source, you should setup both pinmux in ultron (CPU1) side and CKGEN (for clock frequency) in uboot (CPU2) side.

- Limitations
  - ☐ If PHY interface is not RMII, SP_CLK CAN NOT be configured on LCD26 (3rd SP_CLK pinmux position)
  - ☐ The default output crystal clock setting (If need to change, please contact with novatek)
    - ◆ P_GPIO22 (1st SP_CLK)

- ◆ The pll source is PLL4 (shared with IDE/SDIO spread frequency)
- ◆ Output clock: 25MHz

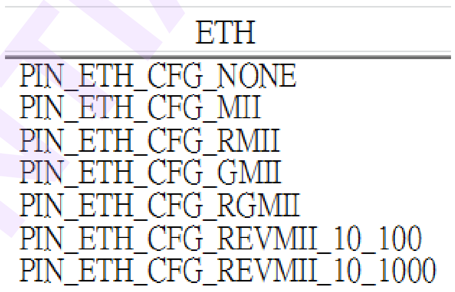
☐ The feature is disable by default

- Pinmux

(1) All the related pinmux configuration shall be setup with pinctrl tool

☐ Location: build/nvt-tools/nvt_pinctrl_tool/top_generator.xlsm

☐ Please refer more details in NT96680_TOOL_UI_pinctrl_tool



ETH
PIN_ETH_CFG_NONE
PIN_ETH_CFG_MII
PIN_ETH_CFG_RMII
PIN_ETH_CFG_GMII
PIN_ETH_CFG_RGMII
PIN_ETH_CFG_REVMII_10_100
PIN_ETH_CFG_REVMII_10_1000

7.3.2. Probing Ethernet platform driver

- General description

NT9668x Ethernet driver is encapsulated as a platform driver. You should create a platform device in the device tree to probe Ethernet driver.

- Limitations

☐ .name (of platform_device) should be “synopsys_eth”

- Flow

Following is a sample to register a platform device which can probe Ethernet driver:

```
eth@f02b0000 {
     compatible = "nvt,synopsys_eth";
     reg = <0xf02b0000 0x10000>;                    //eth register base address + offset
     interrupts = <GIC_SPI 34 IRQ_TYPE_LEVEL_HIGH>;  //eth interrupt
};…
```

7.3.3. Load and start Ethernet driver

- General description

Ethernet driver is supplied as a kernel module and located in *ּ/lib/modules/4.1.0/extra/net/synopsys/ntkimethmac.ko*. You can use shell command “insmod” or “modprobe” to load it.

- Flow

  Following is a script sample to load and start eth0:

```
modprobe ntkimethmac


ifconfig eth0 down

ifconfig eth0 hw ether 00:80:48:BA:d1:30

ifconfig eth0 up


ifconfig eth0 192.168.0.3
```

7.4. Troubleshooting

7.4.1. Dose Novatek has Ethernet PHY compatibility list?

[**RGMII**]

RTL8211F

RTL8211E

Atheros 8035

ICPlus1001

[**RMII**]

RTL8201F

8. PWM

8.1. Feature

- PWM

☐ Input clock frequency:	7324Hz ~ 30 MHz
☐ Output PWM frequency:	
❖ PWM8~PWM19	28 Hz ~ 15 MHz (①)
❖ PWM0~PWM7	1 Hz ~ 15MHz (①)
☐ Channel number:	20 channels
☐ Parameter:	Base period, Rising time, Falling time
☐ Cycle number:	Free run or a specific number
☐ Output polarity:	Invert or not invert
☐ Interrupt event:	Cycle finished, PWM stop

- Micro-stepping

☐ PPS (Pulse per Second):	2000
☐ Channel:	4 sets(4 channels per set)
☐ Graduations	512
☐ PWM basis period:	fixed value of 100
☐ Clock frequency:	Maximum operating clock is 30MHz
☐ Change clock frequency:	Set 0 ~ 2 allow change clock rate during Micro-stepping Operation (Set 3 is not available) (②) (③)

- Cycle count calculator

☐ Clock frequency:	3 MHz, independent of PWM clock
☐ Channel:	4
☐ Polarity:	Invert or not invert
☐ Count condition:	Rising edge or both rising and falling edge
☐ Interrupt event:	Each detection, Trigger detection, Timeout
☐ Filter:	Ignore glitch pulse

Output frequency reference = Input clock / base period

Input clock = 120MHz / (4+n), where n = 0~16380

Input clock range = 7324 Hz ~ 30MHz

Basis period : 2~255 (Unit : input clock)

So, output frequency = input clock / Basis period

Max = input clock 30MHz and Basis period 2 = 15MHz

Min = input clock 7324Hz and Basis period 255 = 28 Hz

Out put 50KHz frequency waveform@PWM channel 0

```
pwm_open(PWMID_0);
PWMInfo.uiPrd          = 2;
PWMInfo.uiRise         = 0;
PWMInfo.uiFall         = 1;
PWMInfo.uiOnCycle      = 0;    // Free run
PWMInfo.uiInv          = 0;
//PWM0~3 use same clock source
pwm_pwmConfigClockDiv(PWM0_3_CLKDIV, 1199);
pwm_pwmConfig(PWMID_0, &PWMInfo);
pwm_pwmEnable(PWMID_0);
```

Clock source organization please reference

As source clock = 120MHz

Output frequency = [Input clock / (Div+1)] / base period

= (120000KHz / (1199+1))/2 = 50KHz

Duty:

Rising time : 0~255

Falling time : 0~255

Basis Period : 2~255

(Unit input clock)

Basis period ≧ Falling time ≧Rising time

Duty can configure as 0%~100% by setting Rising / Falling / Basis period.

Resolution = 1/Basis period

0% : if rising time = falling time

100% : If rising time = 0 and falling time = basis period, the output will always keep high .

Example:

Assume Input clock X, base period Y

➔PWM output freq = X / Y

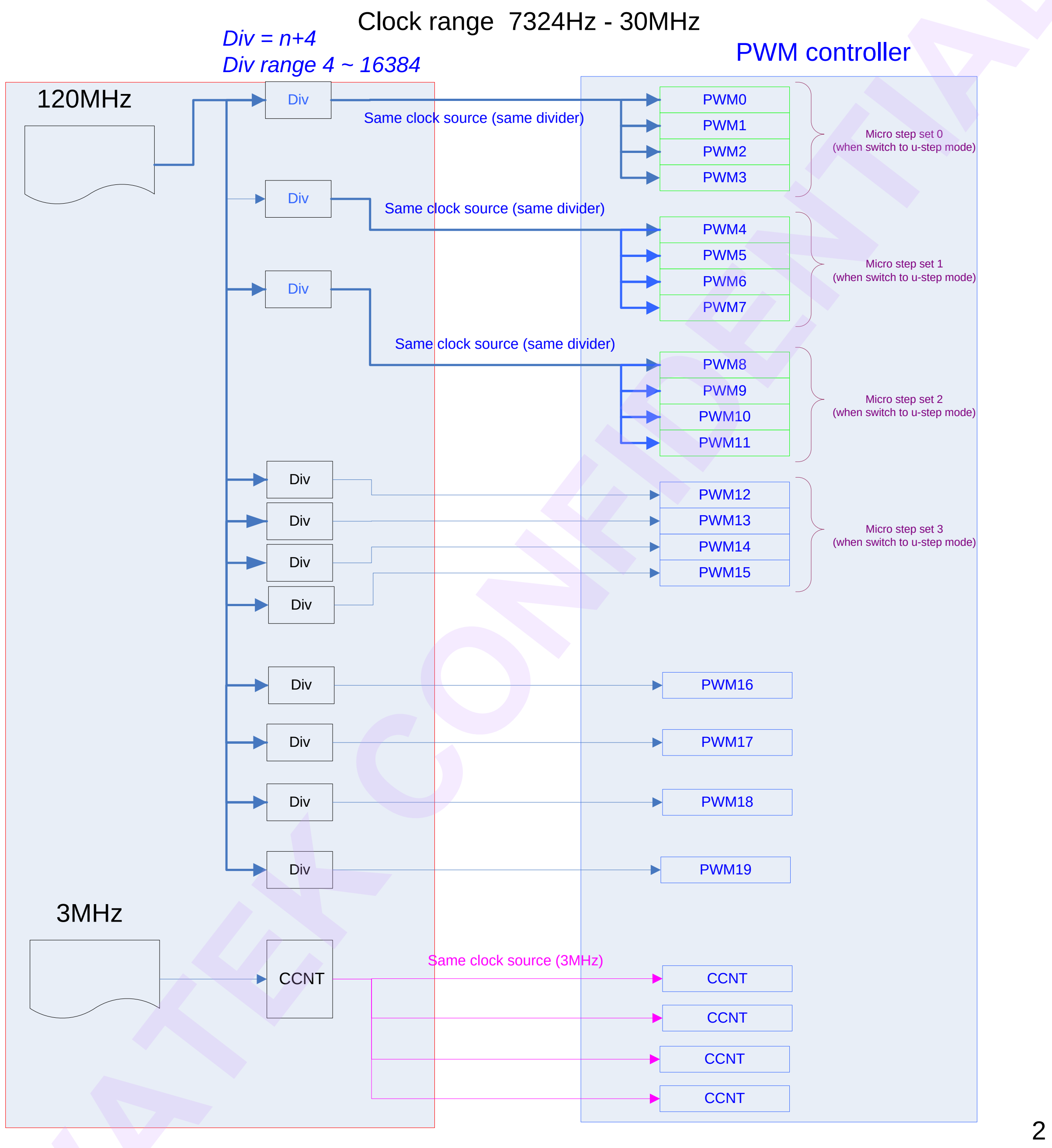
➔Resolution of duty depend on value of Y, For example, Y = 4,

➔ Available Duty will be 0%, 25%, 50%, 75%, 100%

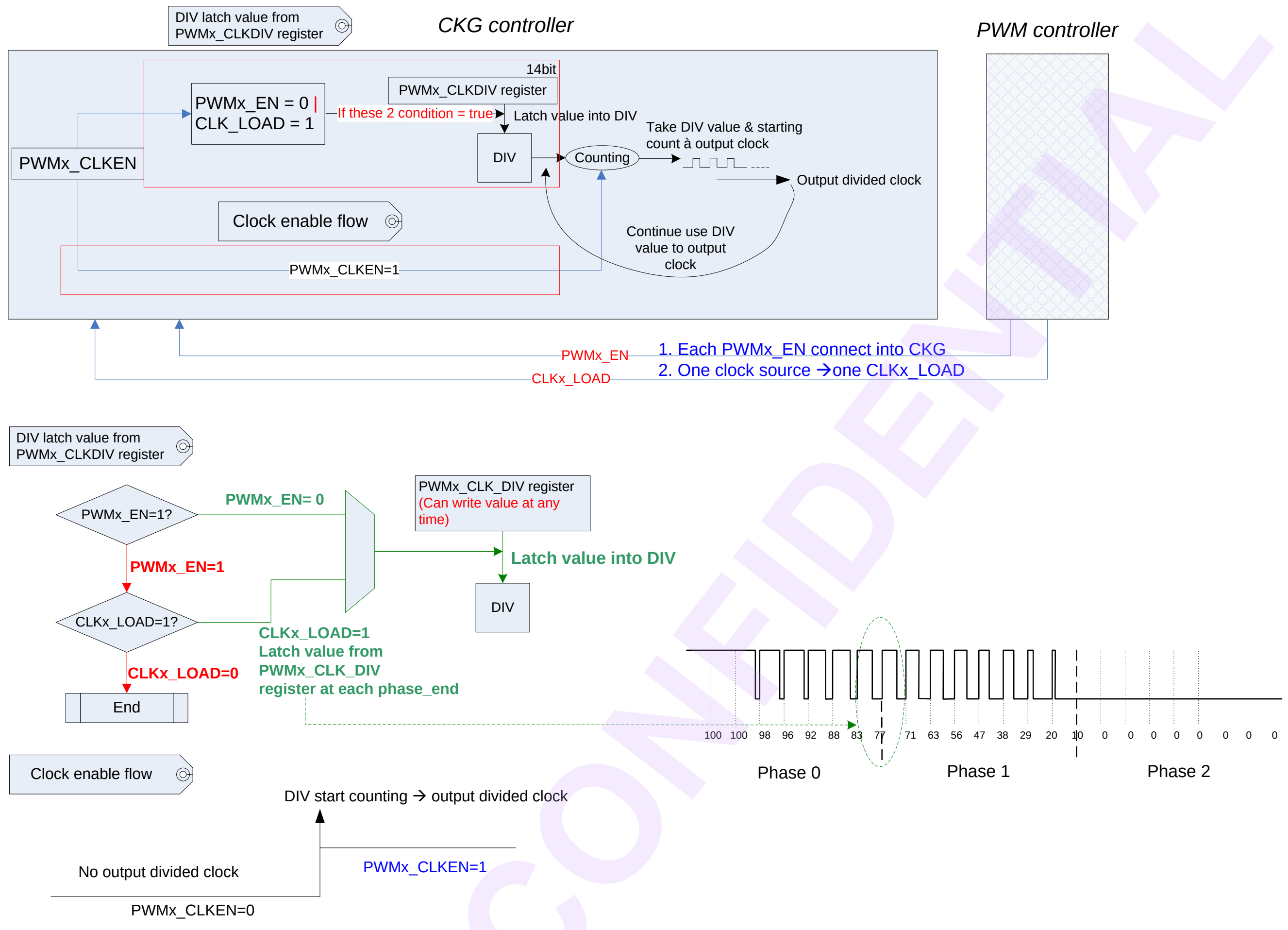
➔ Duty graduation = 100 / Y

➔ Y↑, duty graduation (resolution)↑, PWM max output clock↓

8.2. Block Diagram



Clock generator latch clock DIV flow


DIV latch value from PWMx_CLKDIV register
CKG controller
PWM controller
14bit
PWMx_CLKDIV register
PWMx_EN = 0 | CLK_LOAD = 1
If these 2 condition = true
Latch value into DIV
Take DIV value & starting count à output clock
PWMx_CLKEN
DIV
Counting
Output divided clock
Clock enable flow
Continue use DIV value to output clock
PWMx_CLKEN=1
PWMx_EN
CLKx_LOAD
1. Each PWMx_EN connect into CKG
2. One clock source →one CLKx_LOAD
DIV latch value from PWMx_CLKDIV register
PWMx_EN=1?
PWMx_EN= 0
PWMx_CLK_DIV register (Can write value at any time)
Latch value into DIV
PWMx_EN=1
CLKx_LOAD=1?
DIV
CLKx_LOAD=1 Latch value from PWMx_CLK_DIV register at each phase_end
CLKx_LOAD=0
End
100 100 98 96 92 88 83 77 71 63 56 47 38 29 20 10 0 0 0 0 0 0 0 0
Phase 0
Phase 1
Phase 2
Clock enable flow
DIV start counting → output divided clock
No output divided clock
PWMx_CLKEN=1
PWMx_CLKEN=0

Update policy

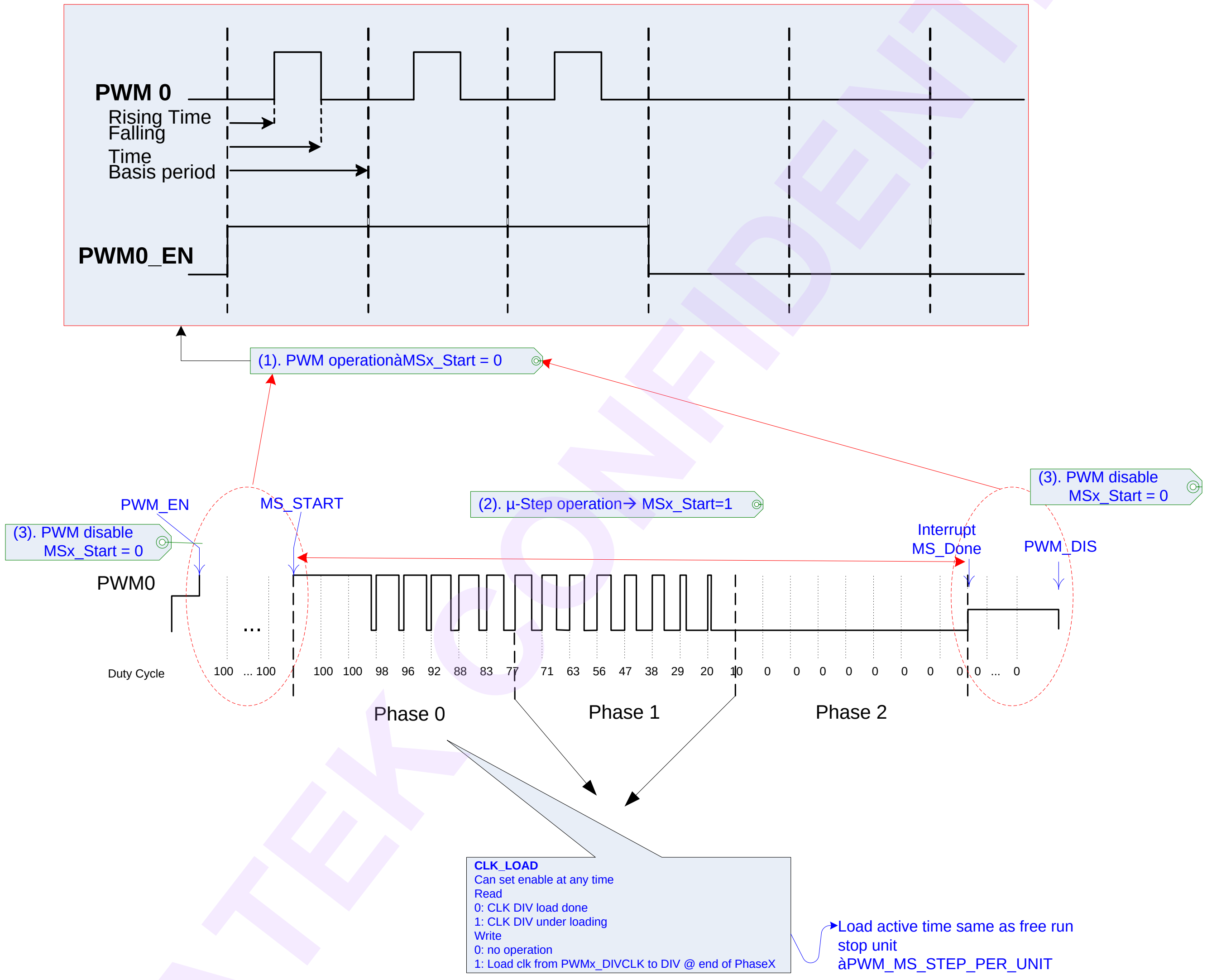
PWMx_CLKEN	0	0	1	1
PWM_EN	0	1	0	1
Update policy	V	X	V	X

Clock generate controller architecture

PWM operation flow (including Micro step operation)

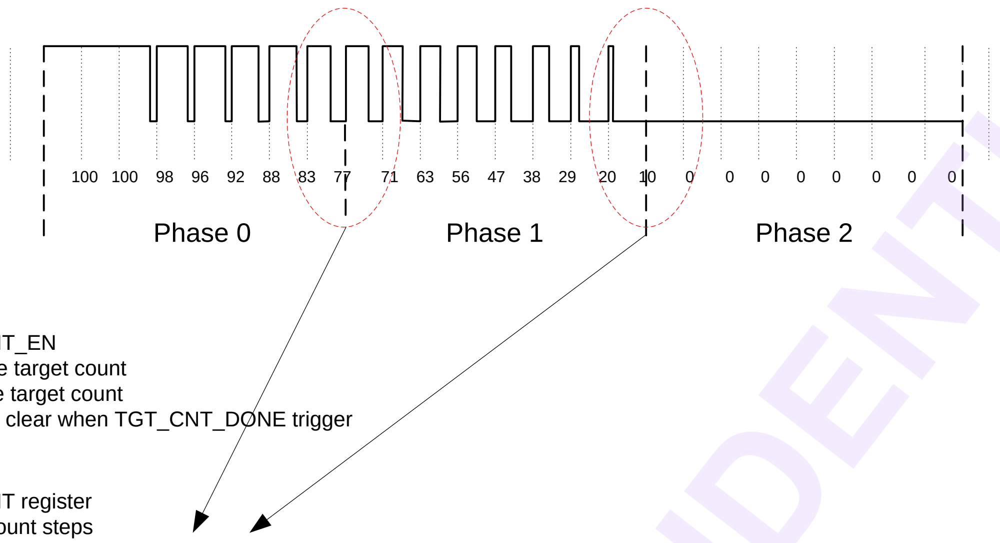
There are three stage of PWM operation flow.

1. PWM enable stage.(Action of normal PWM operation can treat as stage (1)).
2. Micro step operation stage(Clock can be changed dynamically, reference here)
3. PWM disable stage



Clock load timing diagram

Step count operation flow (helpful to handle the timing of set new clock)



TGT_CNT_EN
0: disable target count
1: enable target count
HW auto clear when TGT_CNT_DONE trigger

TGT_CNT register
Target count steps
If TGT_CNT_EN = 1→ Decrease 1 at each phase end →Issue TGT_CNT_DONE interrupt if arrive 0
If TGT_CNT_EN = 0→ Do nothing

Flow chat please reference here

8.3. Header File

The header file is located in \k_driver\include\comm\pwm.h

Place #include "comm\pwm.h" in your source code.

8.4. Function Description

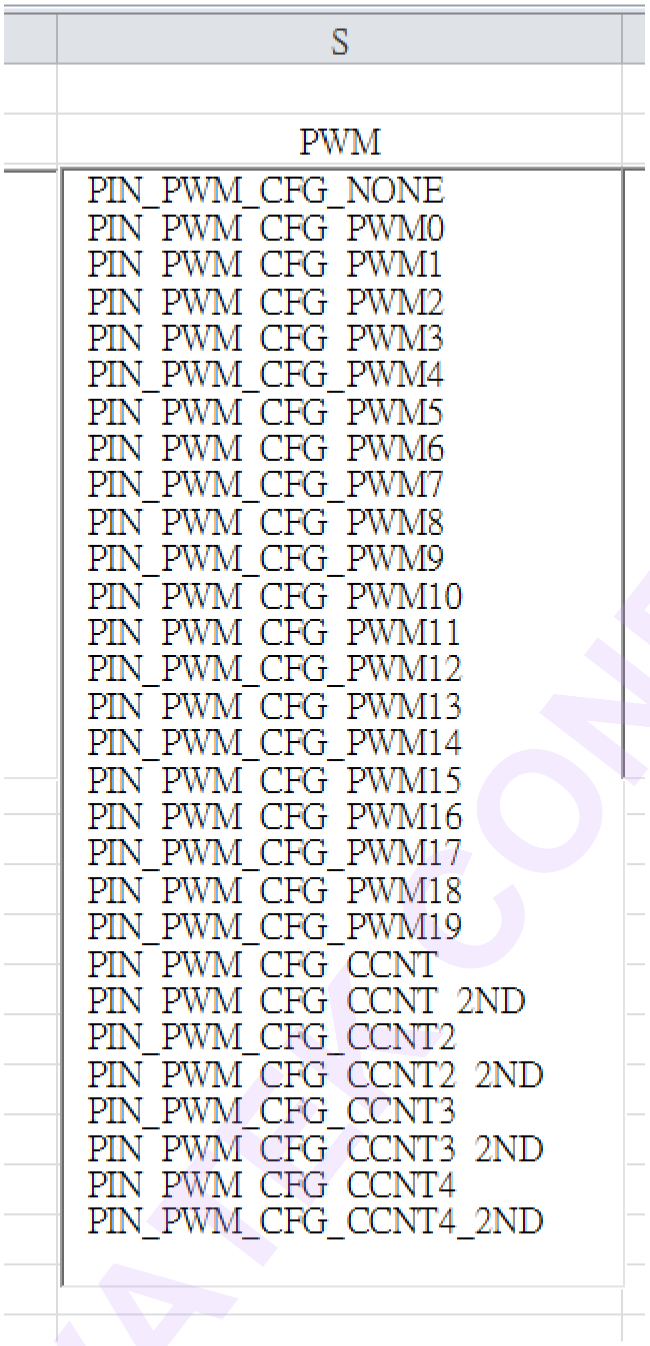
8.4.1. General PWM function

The Pulse Width Modulation (PWM) is supported to control external devices. Totally 20 PWM channels and 4 cycle counter calculator are provided. 16 of these 20 PWM channels can configure as micro stepping set (4 channels per set).

The working frequency range of PWM0~PWM19 is from 30MHz down to 7324Hz. Among these 20 PWM channels, PWM0-3, PWM4-7, PWM8-11 use same clock source, others with dedicate clock source.

The pulse number of each PWM channel can be configured as free run or a specified number of cycles. The output of PWM channels can be inverted. If an interrupt is enabled, it will assert when all cycles finished and stop.

The pinmux configuration file is the "na51000_linux_sdk\build\nvt-tools\nvt_pinctrl_tool\top_generator". The user must well define the Group "PIN_FUNC_PWM" (Configure which PWM or CCNT you would like to open).



S

PWM

```
PIN_PWM_CFG_NONE
PIN_PWM_CFG_PWM0
PIN_PWM_CFG_PWM1
PIN_PWM_CFG_PWM2
PIN_PWM_CFG_PWM3
PIN_PWM_CFG_PWM4
PIN_PWM_CFG_PWM5
PIN_PWM_CFG_PWM6
PIN_PWM_CFG_PWM7
PIN_PWM_CFG_PWM8
PIN_PWM_CFG_PWM9
PIN_PWM_CFG_PWM10
PIN_PWM_CFG_PWM11
PIN_PWM_CFG_PWM12
PIN_PWM_CFG_PWM13
PIN_PWM_CFG_PWM14
PIN_PWM_CFG_PWM15
PIN_PWM_CFG_PWM16
PIN_PWM_CFG_PWM17
PIN_PWM_CFG_PWM18
PIN_PWM_CFG_PWM19
PIN_PWM_CFG_CCNT
PIN_PWM_CFG_CCNT_2ND
PIN_PWM_CFG_CCNT2
PIN_PWM_CFG_CCNT2_2ND
PIN_PWM_CFG_CCNT3
PIN_PWM_CFG_CCNT3_2ND
PIN_PWM_CFG_CCNT4
PIN_PWM_CFG_CCNT4_2ND
```

Caution:

PWMID_0~3(use same clock divider to generate target source clock)

PWMID_4~7(use same clock divider to generate target source clock)

PWMID_8~11(use same clock divider to generate target source clock)

PWMID_12~19(use dedicated clock divider to generate target source clock)

(Please refer to here)

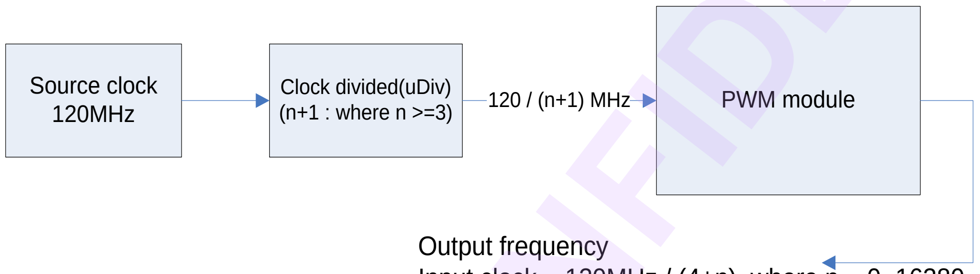
- PWM clock decision : Configured by uiDiv belong to structure PWM_CFG

  Source clock rate = 120MHz

  uiDiv valid value = 0x3 – 0x3FFF

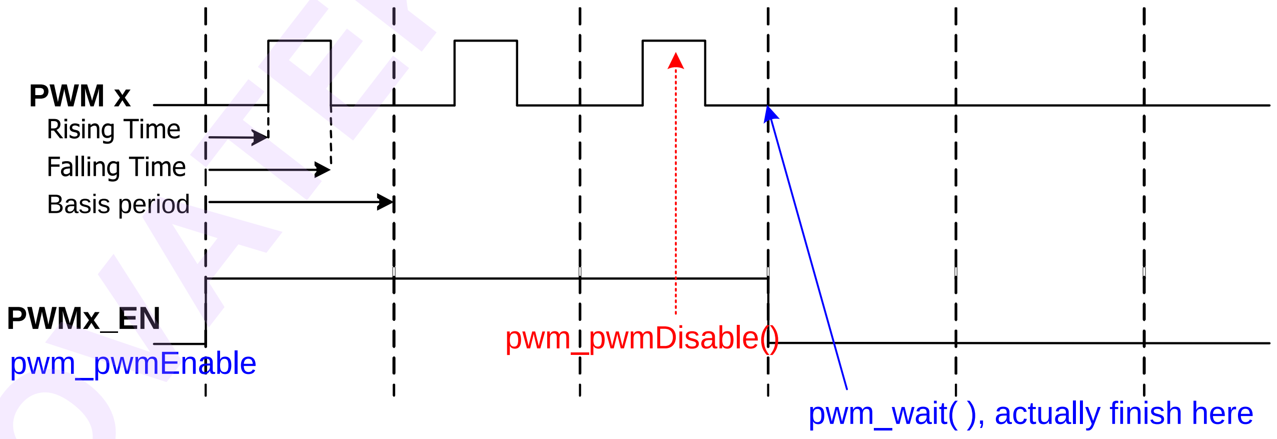
  PWM input clock = source clock / (uiDiv + 1)

  PWM input clock = 120 / Div + 1, range is 30MHz – 7324Hz

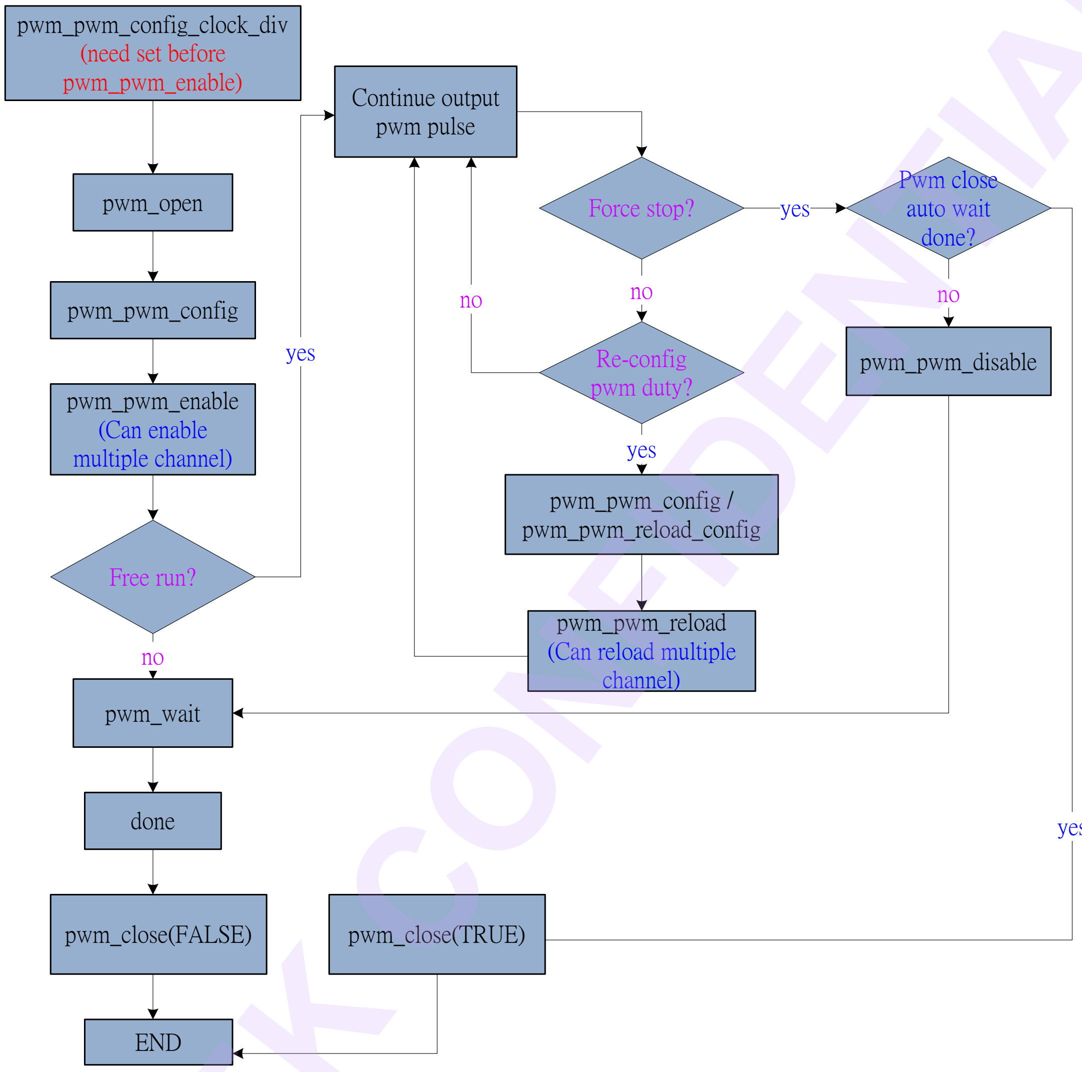


Output frequency
Input clock = 120MHz / (4+n), where n = 0~16380
Input clock range = 7324 Hz ~ 30MHz
Basis period : 2~255 (Unit : input clock)
So, output frequency = input clock / Basis period
Max = input clock 30MHz and Basis period 2 = 15MHz
Min = input clock 7324Hz and Basis period 255 = 28 Hz

- The PWM control flow :



Flow chart:



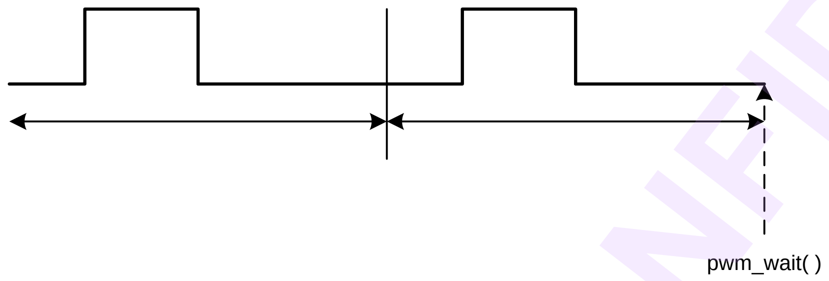
Note: where pwm_pwm_enable and pwm_pwm_reload can enable / reload multi channels at the same time.

Usage example:

```
PWM_CFG PWMInfo;
pwm_open(PWMID_2);// Open pwm channel 1 ( 0 – 19 )+
pwm_pwm_config_clock_div(PWM0_3_CLKDIV, 3);
PWMInfo.ui_prd = 10;
PWMInfo.ui_rise = 2;
PWMInfo.ui_fall = 5;
PWMInfo.ui_on_cycle = 2;
PWMInfo.ui_inv = 0
pwm_pwm_config(PWMID_2, &PWMInfo);
pwm_pwm_enable(PWMID_2, &PWMInfo);
pwm_wait(PWMID_2, PWM_TYPE_PWM);
pwm_close(PWMID_2, FALSE);
or
pwm_close(PWMID_2, TRUE);
```

Input clock frequence = 120 / (3+1) = 30 MHz

Output pwm frequence = 30 / 10 = 3 MHz



PWM0 ~ PWM7 rising, falling and base period can be set from 0 to 65535.

PWM8 ~ PWM19 rsing, falling and base period can be set from 0 to 255.

Duty calculate:

Rising time : 0~255

Falling time : 0~255

Basis Period : 2~255

(Unit input clock)

Basis period ≧ Falling time ≧Rising time

Duty can configure as 0%~100% by setting Rising / Falling / Basis period.

Resolution = 1/Basis period

0% : if rising time = falling time

100% : If rising time = 0 and falling time = basis period, the output will always keep high .

Example:

Assume Input clock X, base period Y

➔PWM output freq = X / Y

➔Resolution of duty depend on value of Y, For example, Y = 4,

➔Available Duty will be 0%, 25%, 50%, 75%, 100%

➔Duty graduation = 100 / Y

➔Y↑, duty graduation (resolution)↑, PWM max output clock↓

If user is going to output PWM pulse as 1 or 0, example code is as bellow

Please reference output_0 and output_1

Usage example:

```
PWM_CFG PWMInfo_0;

PWM_CFG PWMInfo_100;


pwm_open(PWMID_2);          // Open pwm channel 1 ( 0 – 19 )

pwm_open(PWMID_1);

pwm_pwm_config_clock_div(PWM0_3_CLKDIV, 3);              //Any divided

PWMInfo_0.ui_prd = 2;

PWMInfo_0.ui_rise = 1;

PWMInfo_0.ui_fall = 1;          //rising = faling time

PWMInfo_0.ui_on_cycle = 0;  //Free run

PWMInfo_0.ui_inv = 0



PWMInfo_100.ui_prd = 10;

PWMInfo_100.ui_rise = 0;        //Rising time = 0

PWMInfo_100.ui_fall = 10;       //Falling = base period ➔ output always high

PWMInfo_100.ui_on_cycle = 0;          //Free run

PWMInfo_100.ui_inv = 0


pwm_pwm_config(PWMID_2, & PWMInfo_0);

pwm_pwm_config(PWMID_1, & PWMInfo_100);


pwm_pwm_enable(PWMID_2);

pwm_pwm_enable(PWMID_1);

PWM2 will output 0 and PWM1 will output 1
```

Usage example:

```
//Output device maximum frequence is 500Hz
```

```
//So, input maximum frequence is 500 x 255(max base period) = 127500(Hz)
// 120000000Hz / (X+1) = 127500, X = 939 ➔ Output period ~ 500Hz
PWM_CFG PWMInfo1
PWM_CFG PWMInfo2;

pwm_open(PWMID_2);                // Open pwm channel 2 ( 0 – 19 )
pwm_open(PWMID_1);                // Open pwm channel 1 ( 0 – 19 )
pwm_pwm_config_clock_div(PWM0_3_CLKDIV, 939);                // Any divided
PWMInfo1.ui_prd = 2;
PWMInfo1.ui_rise = 1;
PWMInfo1.ui_fall = 1;             //rising = faling time
PWMInfo1.ui_on_cycle = 0;         //Free run
PWMInfo1.ui_inv = 0               //Input clock


PWMInfo2.ui_prd = 10;
PWMInfo2.ui_rise = 0;             //Rising time = 0
PWMInfo2.ui_fall = 10;            //Falling = base period ➔ output always high
PWMInfo2.ui_on_cycle = 0;         //Free run
PWMInfo2.ui_inv = 0

pwm_pwm_config(PWMID_2, & PWMInfo2);
pwm_pwm_config(PWMID_1, & PWMInfo1);
pwm_pwm_enable(PWMID_1 | PWMID_2); // Enable pwm1 &pwm2 at the same time


//Also, can reload at the same time
PWMInfo1.ui_prd = 20;
PWMInfo1.ui_rise = 5;
PWMInfo1.ui_fall = 10;
pwm_pwm_config(PWMID_1, & PWMInfo1);
PWMInfo2.ui_rise = 0;             //Rising time = 0
PWMInfo2.ui_fall = 10;
```

```
PWMInfo2.ui_prd = 100;
pwm_pwm_config(PWMID_2, & PWMInfo2);
or
pwm_pwm_reload_config(PWMID_2, PWMInfo2.ui_rise, PWMInfo2.ui_fall, PWMInfo2.ui_prd);
//Note : Only Base period / rising / falling time can be reload while pwm running
pwm_pwm_reload(PWMID_1 | PWMID_2); // Reload pwm1 &pwm2 at the same time
```

Note: When multi-channels enable / reload at the same time, the maximum latency is 1T PWM clock cycle time

In this example, 1T pwm clock cycle = 1 / (120MHz / 940) ~= 7.8us latency

Once if need output 500Hz, base case is input clock = 120MHz / 4 = 30MHz

1T = 0.33us

But Base period need be minimum 30 000 000 / X = 500 ➔ X = 60000

For this case, better increase base period / rising / falling up to 16 bits.pwm

8.4.2. Micro step function

We separate 512 graduations from 360 degree. So a duty cycle of each PWM cycle is calculated by sin operation. The duty cycle value table is list as follow. The angle value gap between two continuous rows of following table is 0.703125 degree.

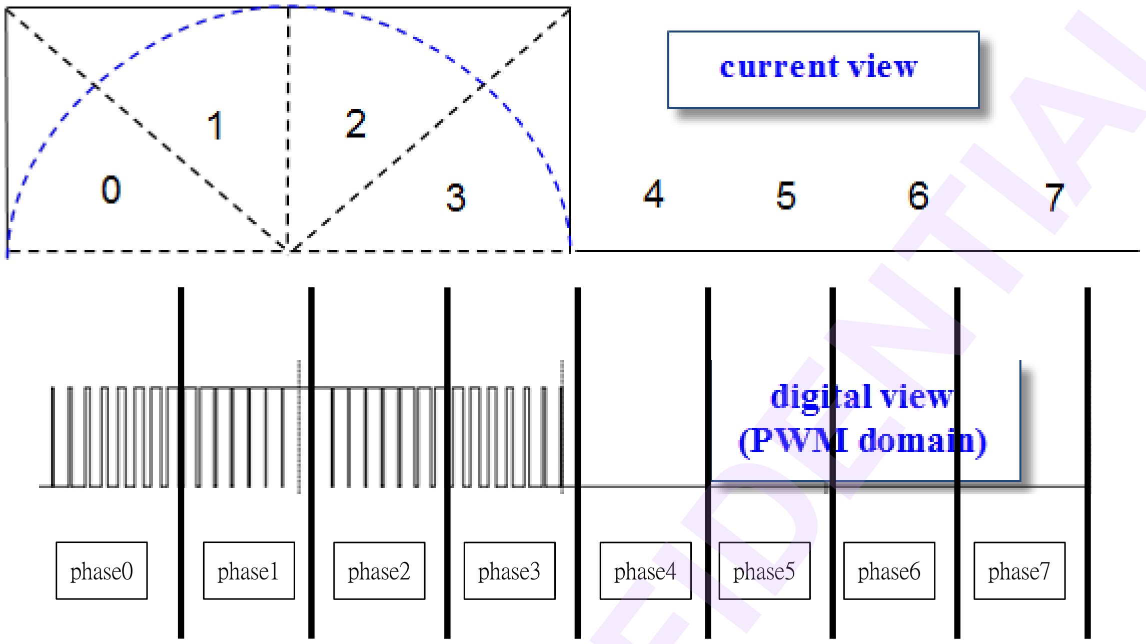
Each element of the following table is the number of duty-cycle (%).

(Each angle = 360 / 512 = 0.703125, duty cycle = sin(angle) x 100, example sin(2.8125) = 5 (round it to the nearest whole number))

In micro step control block, there are 4 channels action at the same time.(0-3/4-7/8-11/12-15), therefore, there are 4 set of micro-step. Related APIs are list as follow

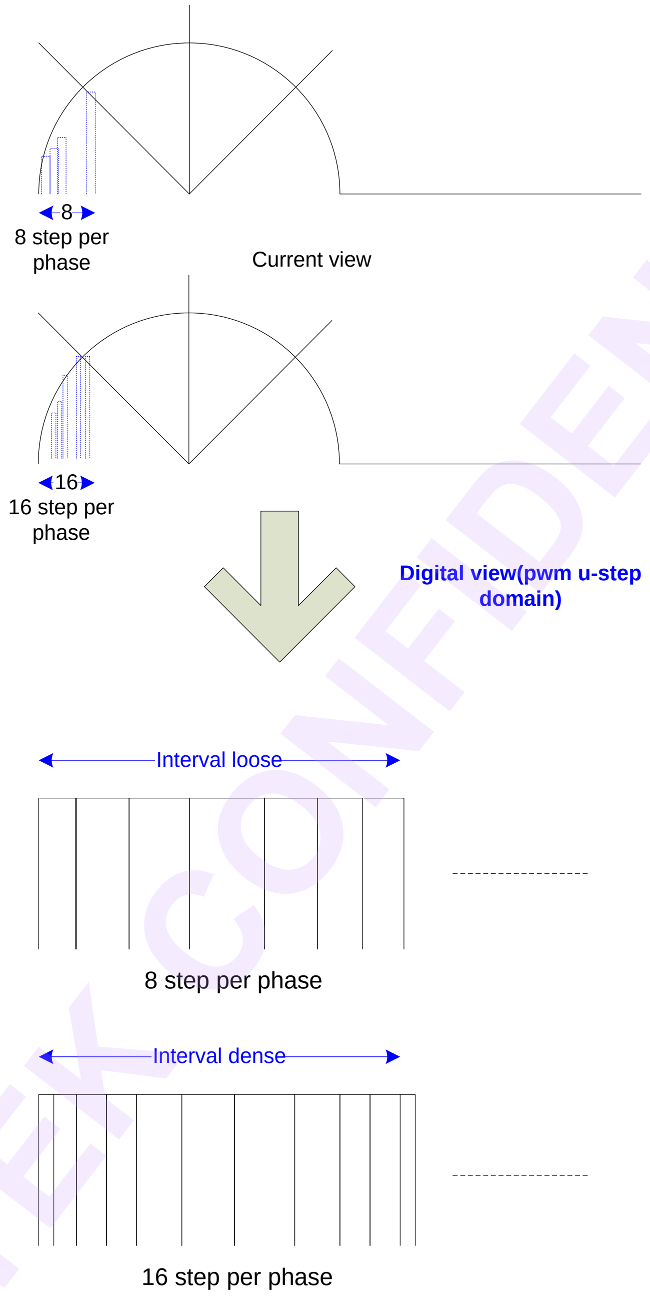
Before use µ-step, need configured some type of µ-step such as

- ☐ Configured micro step motor specific configuration
- ☐ Configured phase of each channel(there are 4 channels of each µ-step set)
- ☐ Example configured phase = 8 such as,



- Configured other configuration remain µ-step configuration, illustrate as follow
  - Direction : Increase or decrease, for example phase 0→2→4→6→ … or 6→4→2→0
  - Cycle : Number of phase for µ-step
  - Step per phase : 8 step/phase or 16 step/phase or 32 step/phase or 64 step/phase

Step per phase example:



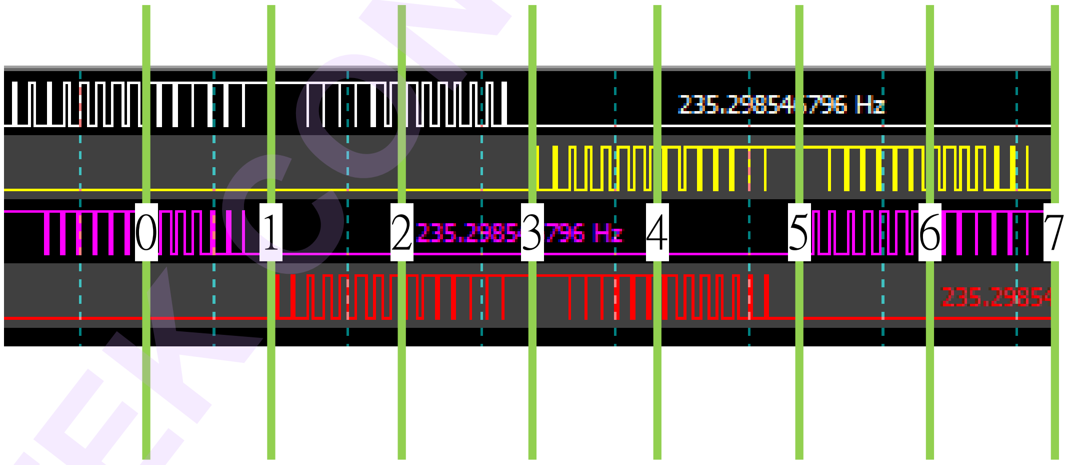
Note: The more step per phase, the more PWM pulse will output at one cycle, so the motor will more smooth.

❖ Phase type : 1 unit each operation and 2 unit each operation
 - 1 unit : will output 0→1→2→ …→7
 - 2 unit : will output 0→2→4→ .. →6

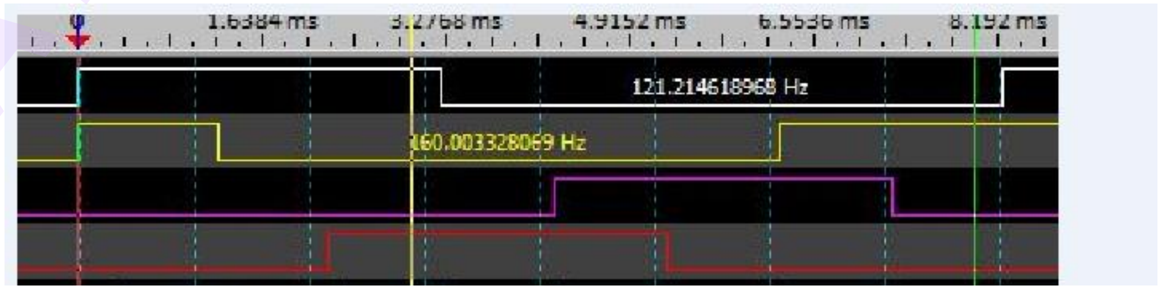
- How to calculate clock divider ?
  - Assume one unit is β ms.
  - Clk_div = (120MHz *β)/(step_per_phase*100*unit) – 1
- How to map truth table to micro step signal ?
  - + point in truth table is mapping to the end of the step (not like the way you mapping to square wave, because micro step is imitation of anolog signal.)
  - 1-2 phase
    - Truth table

	0	1	2	3	4	5	6	7
Pwm0(white)	+	+	+					
Pwm1(yellow)					+	+	+	
Pwm2(pink)	+						+	+
Pwm3(red)			+	+	+			

    - Micro step waveform



    - square waveform

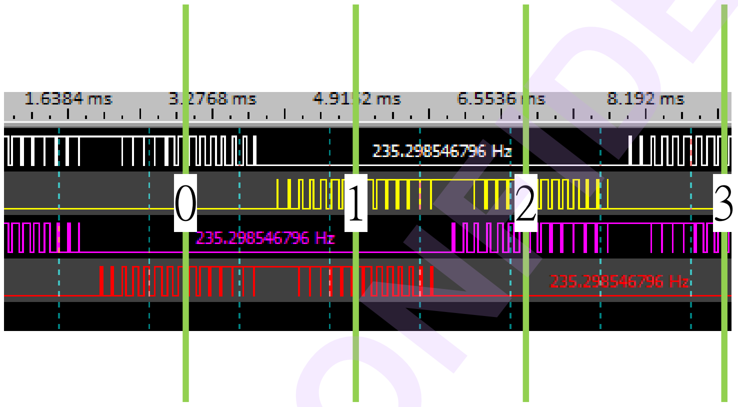


❖ 2-2 phase

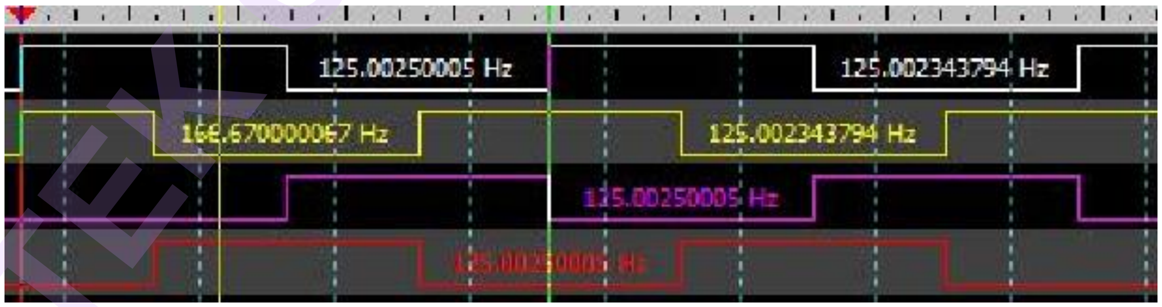
➢ Truth table

	0	1	2	3
Pwm0(white)	+			+
Pwm1(yellow)		+	+	
Pwm2(pink)			+	+
Pwm3(red)	+	+		

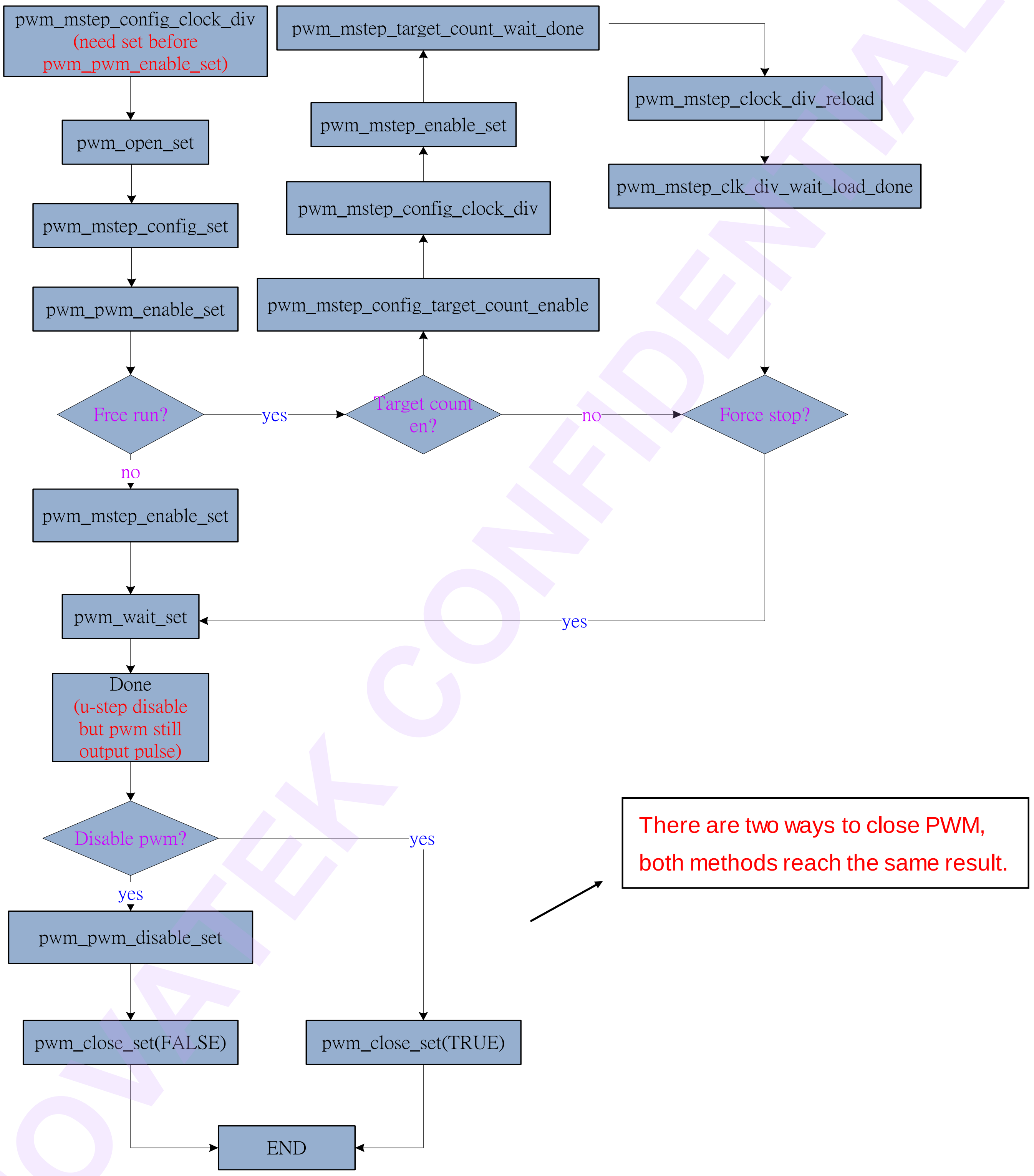
➢ Micro step waveform



➢ square waveform



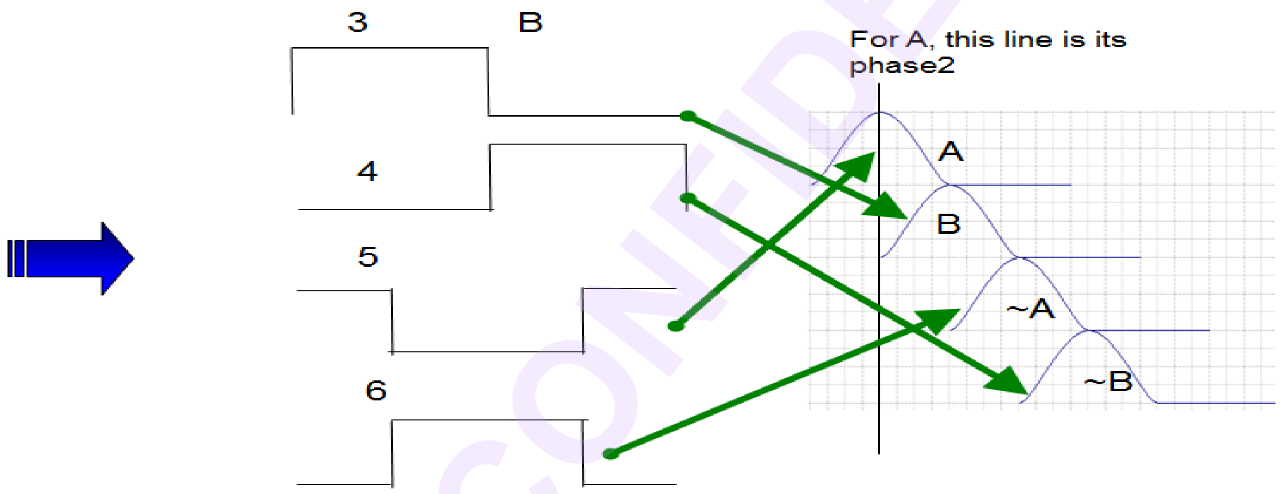
Flow chart:


pwm_mstep_config_clock_div (need set before pwm_pwm_enable_set)
pwm_open_set
pwm_mstep_config_set
pwm_pwm_enable_set
Free run?
yes
no
pwm_mstep_enable_set
pwm_wait_set
Done (u-step disable but pwm still output pulse)
Disable pwm?
yes
yes
pwm_pwm_disable_set
pwm_close_set(FALSE)
pwm_close_set(TRUE)
END
Target count en?
pwm_mstep_config_target_count_enable
pwm_mstep_config_clock_div
pwm_mstep_enable_set
pwm_mstep_target_count_wait_done
no
pwm_mstep_clock_div_reload
pwm_mstep_clk_div_wait_load_done
Force stop?
yes
There are two ways to close PWM, both methods reach the same result.

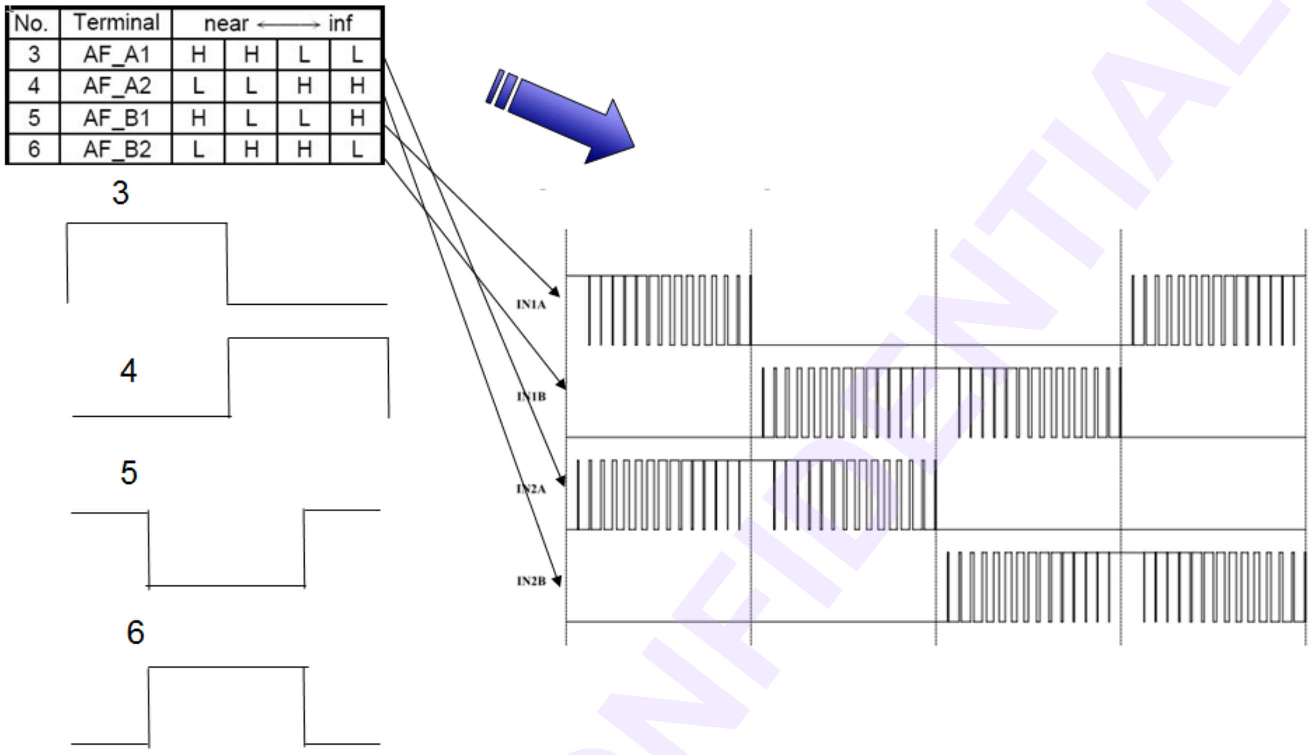
Example:

0 → 3 step

No.	Terminal	near ←→ inf			
3	AF_A1	H	H	L	L
4	AF_A2	L	L	H	H
5	AF_B1	H	L	L	H
6	AF_B2	L	H	H	L



Target output wave will be as bellow :

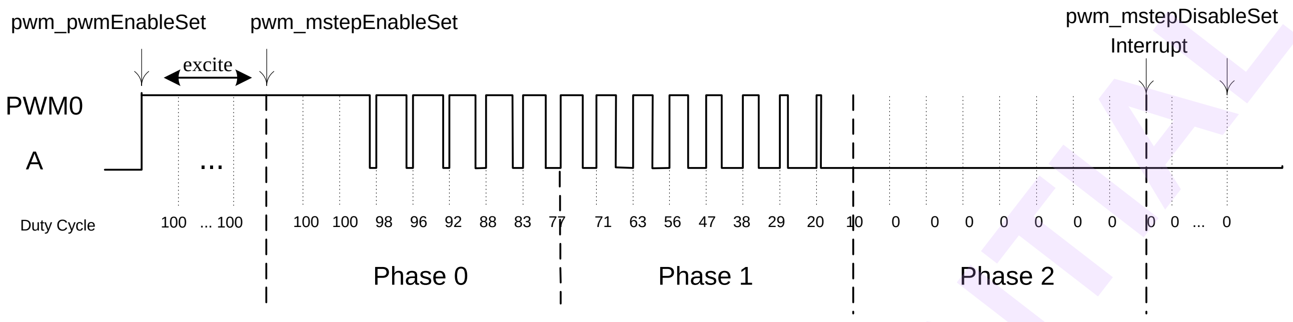


No.	Terminal	near ←→ inf			
3	AF_A1	H	H	L	L
4	AF_A2	L	L	H	H
5	AF_B1	H	L	L	H
6	AF_B2	L	H	H	L

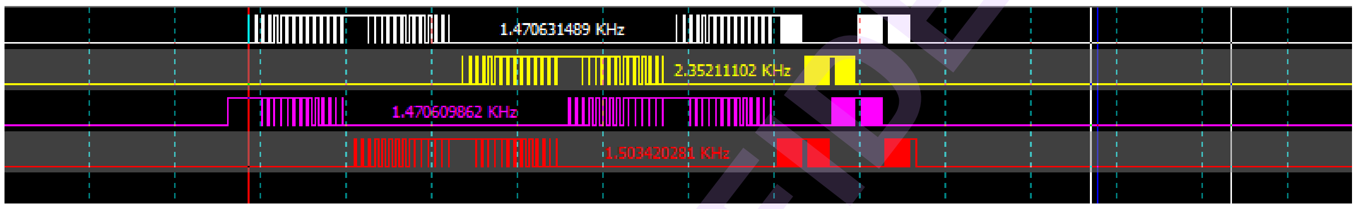
Sample code:

```
PWM_4CH pwmChannel[4] =
{
    //AF_A1 AF_A2    AF_B1   AF_B2
    {    0,     4,     2,     6}, //Step0
    {    2,     6,     4,     0}, //Step1
    {    4,     0,     6,     2}, //Step2
    {    6,     2,     0,     4}, //Step3
}
MSCOMMON_CFG      msCommonCfg;
MS_CH_PHASE_CFG msSetChPhCfg;
msCommonCfg.ui_dir = uiDir;
msCommonCfg.ui_on_cycle = 10;                          //Assume total 10 operation units and change clock     //after 4
                                                       operation units
msCommonCfg.ui_step_per_phase = ((TOTAL_08_STEP_PER_PHASE)); // 8 step per phase
msCommonCfg.ui_phase_type = PWM_MS_2_2_PHASE_TYPE;
msCommonCfg.is_square_wave = FALSE; → TRUE for using square wave, FALSE for micro step
msCommonCfg.ui_threshold_en = FALSE;
msCommonCfg.ui_threshold = 0;
// Note:programmer MUST take care the correct step of this flow
msSetChPhCfg.ui_ch0_phase = pwmChannel[uiStartStep].uiCh_A1;
msSetChPhCfg.ui_ch1_phase = pwmChannel[uiStartStep].uiCh_A2;
msSetChPhCfg.ui_ch2_phase = pwmChannel[uiStartStep].uiCh_B1;
msSetChPhCfg.ui_ch3_phase = pwmChannel[uiStartStep].uiCh_B2;
pwm_config_clock_div_set(PWM_MS_SET_0, 23);
if(pwm_open_set(PWM_MS_SET_0) != E_OK)
{
    debug_msg("MS set open error\r\n");
    return;
}
if(pwm_mstep_config_set(PWM_MS_SET_0, &msSetChPhCfg, &msCommonCfg) != E_OK)
{
    debug_msg("MS set config error\r\n");
    return;
}
pwm_pwm_enable_set(PWM_MS_SET_0);                           // Excitation
pwm_mstep_config_clock_div(PWM_MS_SET_0, newclockDiv);//Re-config new target clock div but not active
pwm_mstep_config_target_count_enable(PWM_MS_SET_0, 4, TRUE);    //Config assert interrupt after 4 step
Delay(4ms);                                                 // Excitation 4 ms
pwm_mstep_enable_set(PWM_MS_SET_0);                         // Enable u-step function
pwm_mstep_target_count_wait_done(PWM_MS_SET_0);             //Wait 4 step count arrived
pwm_mstep_clock_div_reload(PWM_MS_SET_0, TRUE);             //Wait reload done
//From here, new clock frequency was apply on
pwm_wait_set(PWM_MS_SET_0)                      // wait done
pwm_pwm_disable_set(PWM_MS_SET_0);
pwm_close_set(PWM_MS_SET_0, FALSE);
```

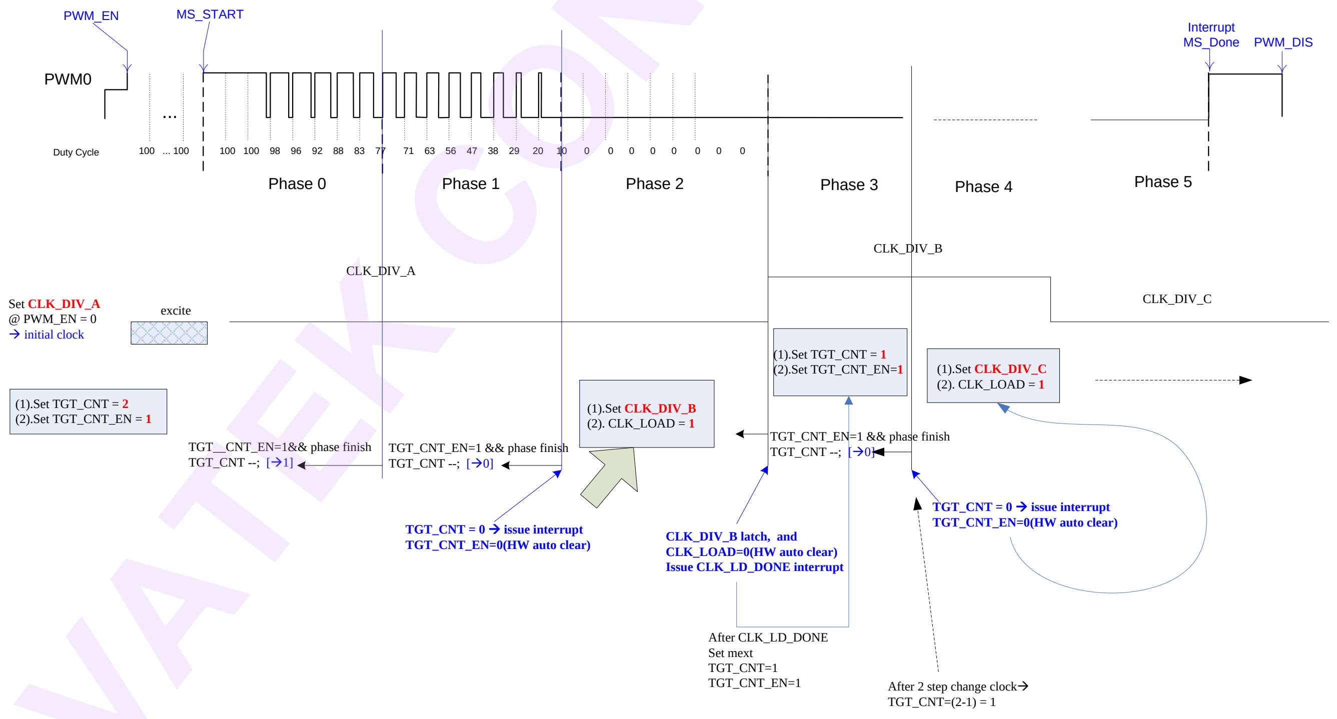
Flow will be:



Output waveform will be



- Example 3



Limitation

Micro-step set0 and set1 cannot use target count at the same time.

If you need to use 2 micro-steps and both target count functions are also needed, you should use micro-step set0 and set2.

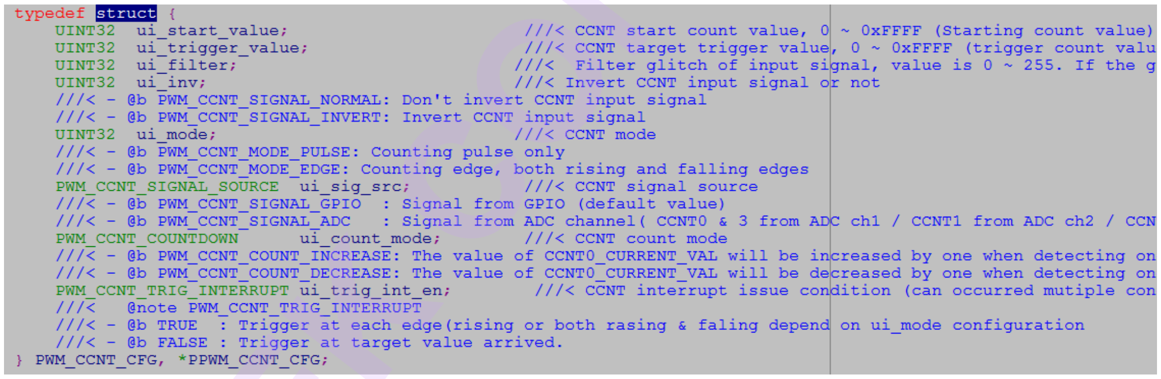
If you need to use 3 micro-steps and all target count functions are also needed, you should wait set0/set1 finish target count then set1/set0 can be used. Or you can use normal open close flow like the flow chart above.

8.4.3. Cycle count function

The cycle count calculator is connected to the GPIO (usually apply for output of the photo interrupter (PI)). It counts the number of the pulses from GPIO(PI). A count number must be set before counting. The number decreases one on each incoming pulse. An interrupt occurs when the number equals to 0.

- There are three conditions of CCNT trigger event
  - ☐ Edge trigger (Can configured as rising edge count only or both rising and falling edge count)
  - ☐ Target value arrived trigger event
  - ☐ Timeout event
- Application layer can wait these three events up to user's configuration.

PWM_CCNT_CFG → ui_trigger_en



```
typedef struct {
    UINT32  ui_start_value;                         ///< CCNT start count value, 0 ~ 0xFFFF (Starting count value)
    UINT32  ui_trigger_value;                       ///< CCNT target trigger value, 0 ~ 0xFFFF (trigger count valu
    UINT32  ui_filter;                             ///<  Filter glitch of input signal, value is 0 ~ 255. If the g
    UINT32  ui_inv;                                ///< Invert CCNT input signal or not
    ///< - @b PWM_CCNT_SIGNAL_NORMAL: Don't invert CCNT input signal
    ///< - @b PWM_CCNT_SIGNAL_INVERT: Invert CCNT input signal
    UINT32  ui_mode;                               ///< CCNT mode
    ///< - @b PWM_CCNT_MODE_PULSE: Counting pulse only
    ///< - @b PWM_CCNT_MODE_EDGE: Counting edge, both rising and falling edges
    PWM_CCNT_SIGNAL_SOURCE  ui_sig_src;             ///< CCNT signal source
    ///< - @b PWM_CCNT_SIGNAL_GPIO   : Signal from GPIO (default value)
    ///< - @b PWM_CCNT_SIGNAL_ADC    : Signal from ADC channel( CCNT0 & 3 from ADC ch1 / CCNT1 from ADC ch2 / CCN
    PWM_CCNT_COUNTDOWN      ui_count_mode;          ///< CCNT count mode
    ///< - @b PWM_CCNT_COUNT_INCREASE: The value of CCNT0_CURRENT_VAL will be increased by one when detecting on
    ///< - @b PWM_CCNT_COUNT_DECREASE: The value of CCNT0_CURRENT_VAL will be decreased by one when detecting on
    PWM_CCNT_TRIG_INTERRUPT ui_trig_int_en;          ///< CCNT interrupt issue condition (can occurred mutiple con
    ///<   @note PWM_CCNT_TRIG_INTERRUPT
    ///< - @b TRUE  : Trigger at each edge(rising or both rasing & faling depend on ui_mode configuration
    ///< - @b FALSE : Trigger at target value arrived.
} PWM_CCNT_CFG, *PPWM_CCNT_CFG;
```

Where PWM_CCNT_TRIG_INTERRUPT can set as

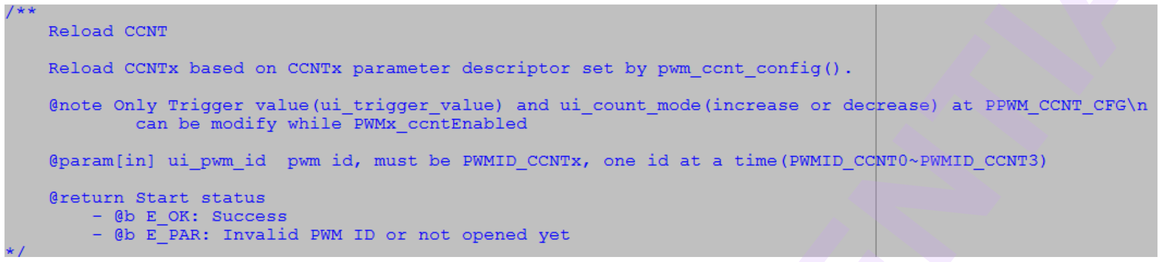
PWM_CCNT_EDGE_TRIG_INTEN or PWM_CCNT_TAGT_TRIG_INTEN or both of these two event.

Note: Once PWM_CCNT_EDGE_TRIG_INTEN is configured, interrupt will issue after each edge. Programmer shell consider about if interrupt each edge is necessary.

- Wait event occurred by pwm_wait(CCNTx_id, PWM_TYPE_CCNT)

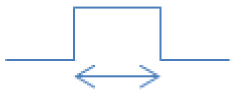
Note : These three events of CCNT might occurred at the same time

Two parameters can be modified while PWMx_CCNT is enabled (not necessary to disable CCNT). Use pwm_ccnt_reload to arrive.



```
/**
    Reload CCNT

    Reload CCNTx based on CCNTx parameter descriptor set by pwm_ccnt_config().

    @note Only Trigger value(ui_trigger_value) and ui_count_mode(increase or decrease) at PPWM_CCNT_CFG\n
            can be modify while PWMx_ccntEnabled

    @param[in] ui_pwm_id  pwm id, must be PWMID_CCNTx, one id at a time(PWMID_CCNT0~PWMID_CCNT3)

    @return Start status
        - @b E_OK: Success
        - @b E_PAR: Invalid PWM ID or not opened yet
*/
```

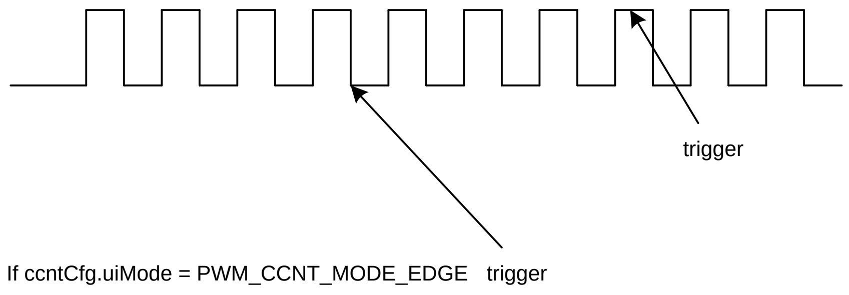
The CCNT clock source is fixed in 3MHz so the clock cycle is 0.3us. such as



Need gather than 0.3 us, > 0.4 us is better

Example 1:

```
PWM_CCNT_CFG ccntCfg;
ccntCfg.ui_mode = PWM_CCNT_MODE_PULSE;                 // Count pulse only
ccntCfg.ui_sig_src = PWM_CCNT_SIGNAL_GPIO;                  // From GPIO Pin
ccntCfg.ui_filter = 0;                                 // Not enable filter function
ccntCfg.ui_inv = 0;                                    // Not invert signal
ccntCfg.ui_start_value =   0;                          // Count from 0
ccntCfg.ui_trigger_value    = 1000;                    // Trigger if detect 8 pulse
ccntCfg.ui_trig_int_en   =   FALSE;                         // FALSE: target arrvied, TRUE: every count
ccntCfg.ui_count_mode   =   PWM_CCNT_COUNT_INCREASE;
pwm_open(PWMID_CCNT1);                                      // Open ccnt channel 1
pwm_ccnt_config(PWMID_CCNT1, &ccntCfg);
pwm_ccnt_enable(PWMID_CCNT1);
pwm_wait(PWMID_CCNT1, PWM_TYPE_CCNT);
```


trigger
If ccntCfg.uiMode = PWM_CCNT_MODE_EDGE trigger

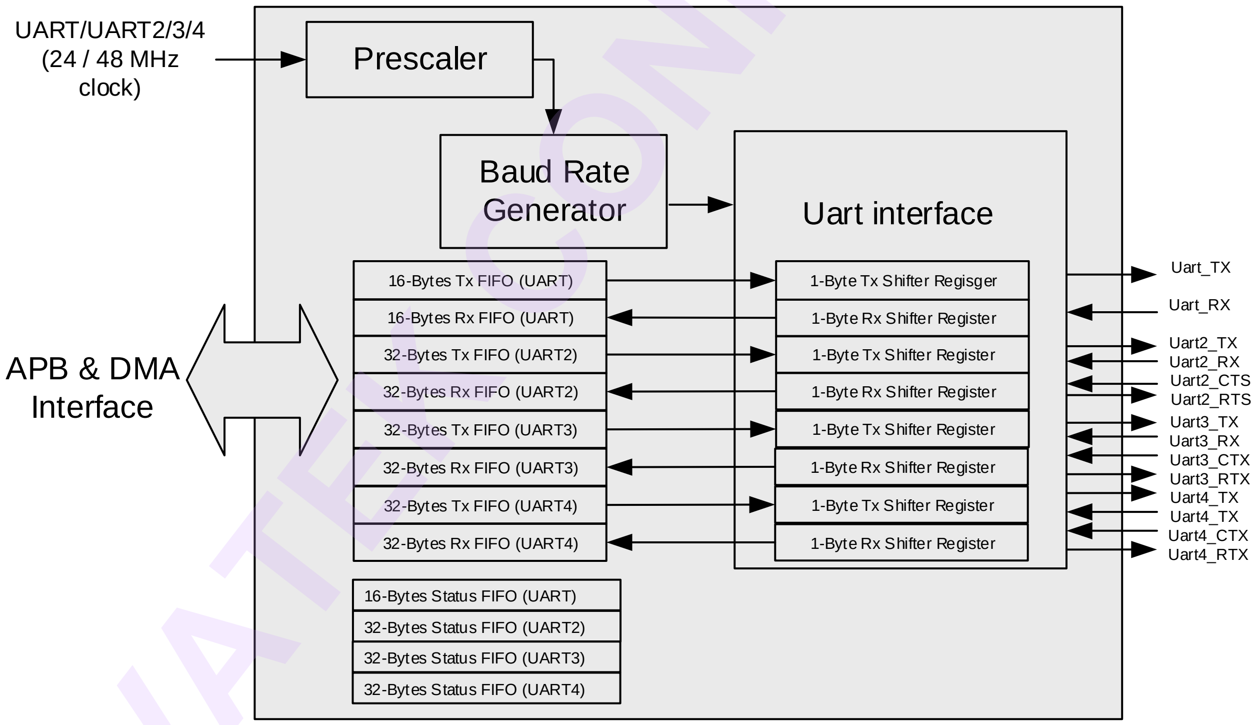
8.5. Troubleshooting

9. UART

9.1. Feature

- High-Speed NS 16C550A-Compatible UART.
- 16-bytes transmit/receive FIFOs for UART and 32-bytes transmit/receive FIFOs for UART2/3/4.
- Maximum baud rate speed 1.5 Mbps for UART and 3.0Mbps for UART2/3/4.
- Internal diagnostic capabilities: Break, parity, overrun, framing error detection.
- Hardware flow control to prevent error caused by long CPU interrupt latency and improve data throughput. .(UART2/3/4 Only)
- Support Full Duplex transmission.

9.2. Block diagram

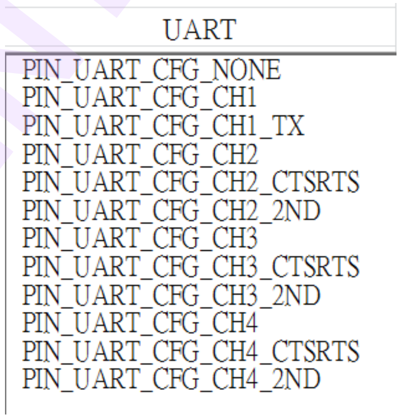


9.3. Configuration

At default state, the UART is dedicated to CPU console output. However, the UART ports could be configured dynamically for dedicated CPU console port. The following description shows the setup steps to changes console port for U-boot and Linux.

9.3.1. All the related pinmux configuration shall be setup with pinctrl tool

- ☐ Location: build/nvt-tools/nvt_pinctrl_tool/top_generator.xlsm
- ☐ Please refer more details in NT96680_TOOL_UI_pinctrl_tool



UART

PIN_UART_CFG_NONE
PIN_UART_CFG_CH1
PIN_UART_CFG_CH1_TX
PIN_UART_CFG_CH2
PIN_UART_CFG_CH2_CTSRTS
PIN_UART_CFG_CH2_2ND
PIN_UART_CFG_CH3
PIN_UART_CFG_CH3_CTSRTS
PIN_UART_CFG_CH3_2ND
PIN_UART_CFG_CH4
PIN_UART_CFG_CH4_CTSRTS
PIN_UART_CFG_CH4_2ND

9.3.2. Modify dedicated UART port in U-boot

Following is a sample to setup UART1 to CPU4 in BSP\u-boot\include\configs\nvt-na51000.h

```
/*-----------------------------------------------------------------------
 * Serial console configuration
 */
#define CONSOLE_UART1  0
#define CONSOLE_UART2  1
#define CONSOLE_UART3  2
#define CONSOLE_UART4  3
#ifdef _NVT_LINUX_SMP_ON_
#define CONFIG_SYS_UART                         CONSOLE_UART1
#else
#define CONFIG_SYS_UART                         CONSOLE_UART4
#endif
#define CONFIG_SYS_BAUDRATE_TABLE               { 9600, 19200, 38400, 57600, 115200 }
#define CONFIG_BAUDRATE                         115200
```

```
/*
 * UART2_1 UART2_TX/RX
 * UART2_2 C_GPIO29/C_GPIO30
 * UART3_1 P_GPIO30/P_GPIO31
 * UART3_2 P_GPIO8/P_GPIO9
 * UART4_1 P_GPIO4/P_GPIO5
 * UART4_2 C_GPIO12/C_GPIO13
 */
#define PINMUX_CHANNEL_1    0
#define PINMUX_CHANNEL_2    1

#define UART_PINMUX_SEL     PINMUX_CHANNEL_1

….
#define CONFIG_BOOTARGS_COMMON                 "earlyprintk console=ttyS3,115200 rootwait "
```

9.4. Flow control

UART 2/3/4 support hardware flow-control feature. As requiring to enable hardware flow-control feature, please following the steps below to configure target channel and functions.

- ☐ Modify options in device tree

nvt-na51000-peri.dts

```
    uart@f0380000 {
         compatible = "ns16550a";
         reg = <0xf0380000 0x1000>;
         interrupts = <GIC_SPI 46 IRQ_TYPE_LEVEL_HIGH>;
         baud = <115200>;
         reg-shift = <2>;
         reg-io-width = <4>;
         no-loopback-test = <1>;
         clock-frequency = <48000000>;
         hw_flowctrl = <1>;
    };
```

9.5. Configure RTS GPIO to UART

UART can configure GPIO as an RTS pin to match the RS485 requirement.

☐ Modify options in device tree

nvt-na51000-peri.dts

Example: Configure L_GPIO(0) as RTS GPIO to uart2

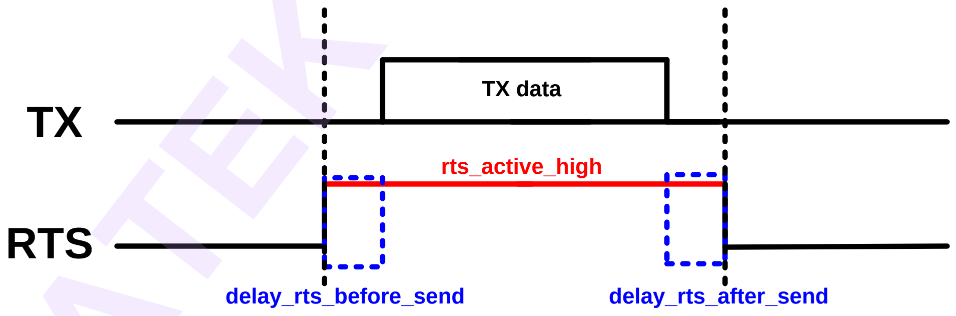
```
uart@f0300000 {
…
     rts-gpio-enable = <1>;
     rts-gpio = <L_GPIO(0) GPIO_LOW>;
     rts-active-high = <1>;
     rts-delay = <5 5>;
};
```

-rts-gpio-enable: Assign 1 to enable the feature, 0 to disable

-rts-gpio: The specified GPIO number and its default value

-rts-active-high: RTS is active high on sending, 0 is the reverse

-rts-delay: The delay before sending and the delay after sending, unit is millisecond(ms)


TX data
TX
rts_active_high
RTS
delay_rts_before_send
delay_rts_after_send