

Fintek

Linux API Guide

v1.33

Dec 13, 2022



Datasheet Revision History

Date	Version	Revision History
2014/9/22	V1.00	1. Initial version.
2014/9/24	V1.01	1. SuperIO detection function modified. 2. Function SetWdtConfiguration parameter modified. Sample code & Library version: 854091fe7
2015/6/9	V1.02	1. Add support for F81768 SuperIO. 2. Static Library released with x86/x86_64. Sample code & Library version: 38a40dea5e
2015/8/10	V1.03	1. Add support GPIO for F81504/508/512 PCIE to UARTs. 2. Fix F81866 GPIO 0x/1x/2x/8x setting loss. 3. Add demo app compile & operate section. Sample code & Library version: 1088a3cc9d
2015/8/19	V1.04	1. Add support EEPROM modification for F81504/508/512 PCIE to UARTs. Sample code & Library version: 7c55f05304
2015/8/25	V1.05	1. Add support for F71808A Watchdog. Sample code & Library version: f28939d0f6
2015/8/26	V1.06	1. Fix detecting multi-SuperIO Issue. Sample code & Library version: f1a336a001
2015/9/10	V1.07	1. Add support for F71869A SuperIO with WDT function. Sample code & Library version: 9dffe67749
2015/10/13	V1.08	1. Add support for F81866 SuperIO with I2C function. Sample code & Library version: eb4ee4f830
2015/12/25	V1.09	1. Add support for F71869A SuperIO with GPIO function. Sample code & Library version: 8005078325
2015/12/28	V1.10	1. Fix for init_fintek_sio() error when multiple SuperIO exist. 2. Add more debug message option. Sample code & Library version: f216964384
2016/9/22	V1.11	1. Fix GPIO mapping for F81803 & F81768. Sample code & Library version: 97527cc592

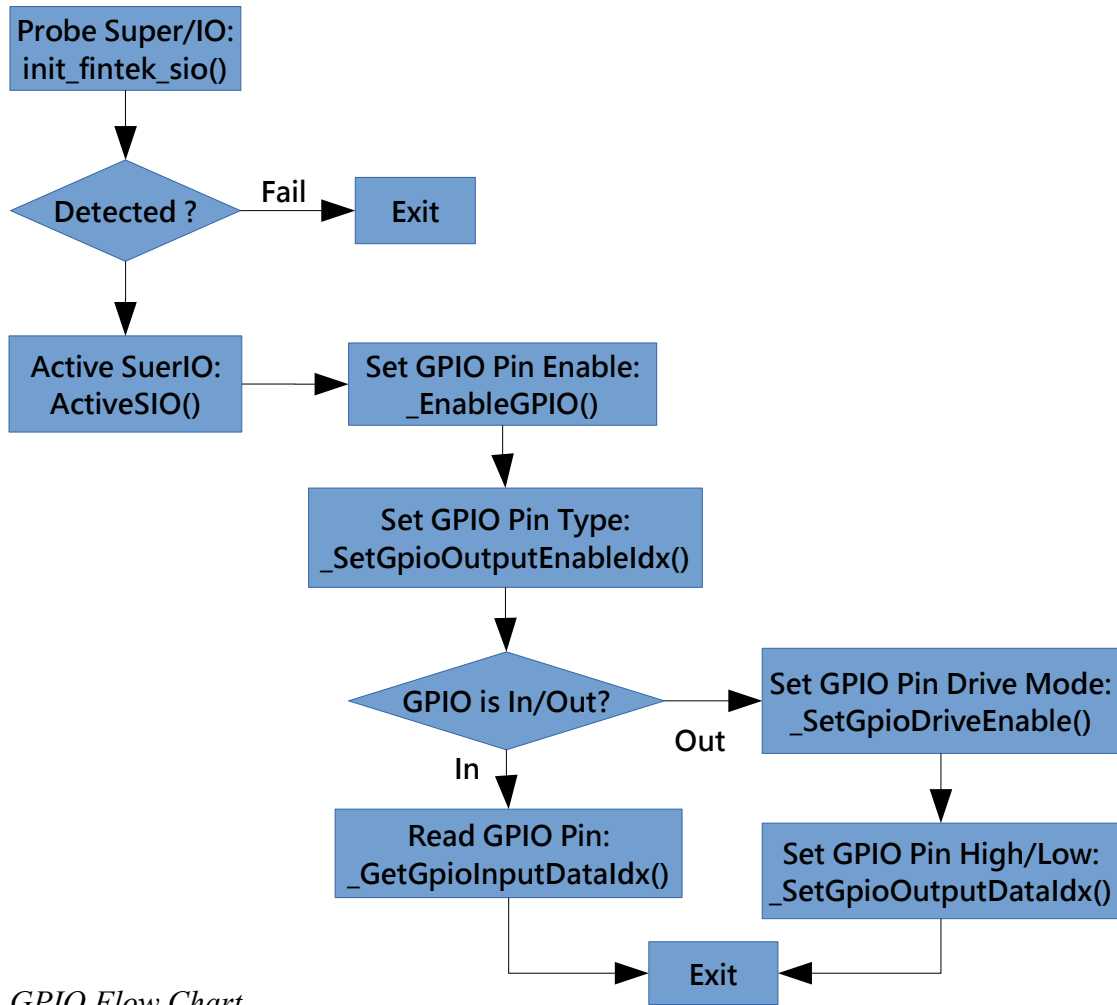


2016/10/11	V1.12	Increase F81886 GPIO read/write performance with 3.7us pulse width. Sample code & Library version: 6627a95cb9
2017/3/16	V1.13	1. Add support for F75113 GPIO & WatchDog function. Sample code & Library version: 7cce21887d
2017/5/16	V1.14	1. Fix for F75113 GPIO4x. Sample code & Library version: fa0b58cd29
2017/5/24	V1.15	1. Add multi-WDT support. (F75113). Sample code & Library version: 5f26f0e4fe
2017/6/6	V1.16	1. Fix for newer GCC build failed issue. Sample code & Library version: ef7eba4668
2017/10/2	V1.17	1. Add support for F75114 GPIO function. Sample code & Library version: d609ca08ff
2017/12/5	V1.18	1. Add support for F81801 GPIO/WDT function. Sample code & Library version: 9f90d44fa9
2017/12/5	V1.19	1. Fix support for F81801 GPIO/WDT function. Sample code & Library version: 1d4dd6b7be
2017/12/28	V1.20	1. Add support for F81804/F81966 GPIO/WDT function. 2. Fix API for demo code "demo_list_device.c" crash on non-HID device. Sample code & Library version: e2476b4759
2018/4/17	V1.21	1. Add demo code "demo_id.c" for separate from F75114 with ID (GPIO03/04) Sample code & Library version: 6684982276
2018/8/20	V1.22	1. Add demo code "demo_spi.c" to demonstrate SPI API usage. 2. Add support for F81532A/F81534A/F81535/F81536 (driver needed) 3. Using older GCC to make library for older system compatibility Sample code & Library version: 6fe4d90695
2018/10/24	V1.23	1. Add support for F75113 (I2C/SMBUS protocol) Sample code & Library version: 3c609b4a57
2018/10/26	V1.24	1. Fix F75113 (I2C/SMBUS protocol) for only 2 sets WDT. Sample code & Library version: c85c4b9308
2018/12/24	V1.25	1. Add armv7l arch for RK3388 Android Sample code & Library version: cd690cbcf2

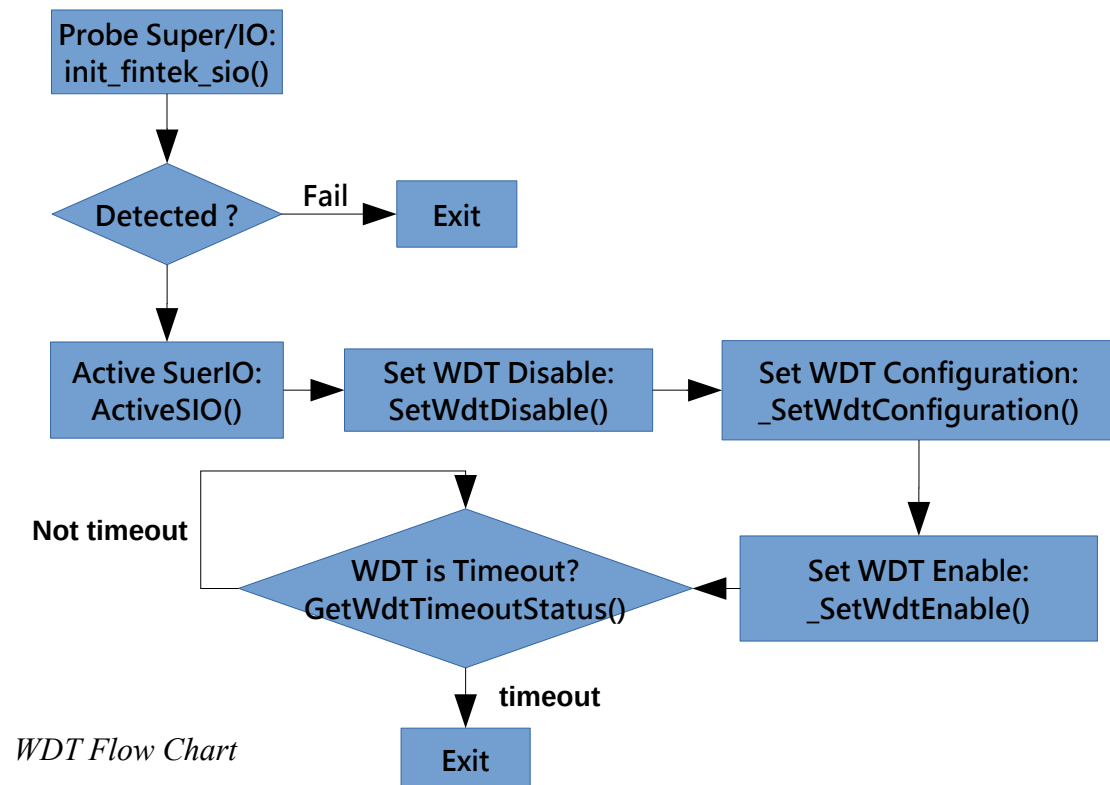


2019/1/17	V1.26	1. Add GPIO interrupt mode for F81866 2. Fix GPIO support and check for F81532A/534A/535/536 3. Improve performance for F75114 GPIO operation Sample code & Library version: a3f9c2db8a
2019/3/13	V1.27	1. Support F81216/AD/H watchdog 2. Support F75115 GPIO 3. Fix some IC lack GetoutputEnable/GetDriveEnable() Sample code & Library version: 28b1ea5bff
2019/9/17	V1.28	1. Add demo_gpio_int.c to handle GPIO interrupt (need driver) 2. Add case open support for F81866 3. Support F81866/966 GPIO interrupt mode Sample code & Library version: c285953950
2020/2/10	V1.29	1. Improve F75115 GPIO set/get speed & group control 2. Add GPIO performance measure function "gpio_performance_demo()" in demo_gpio.c 3. Implement F81216E WDT/GPIO API. 4. Fix demo_gpio.c for interrupt handling issue. 5. Implement F75113 GPIO group read/write mode. 6. Force enable GPIO config by "_EnableGPIO()" on F81504/508/512 Sample code & Library version: 5aa89c2774
2020/6/24	V1.30	1. Add F75115 I2C 2. Add F81966 GPIO interrupt trigger type (Rising/Falling Edge selector) 3. Add arm64 AARCH64 library Sample code & Library version: 4ee1af2d10
2021/2/5	V1.31	1. Add F81968 GPIO/WDT 2. Add F81866 PWM Sample code & Library version: 3d4c1d3e40
2021/10/29	V1.32	1. Add F75115 UART/PWM/Multi-functions switch. Sample code & Library version: 4a8bc87e0d
2022/12/13	V1.33	1. Add F81504/508/512 GPIO group set/get mode. Sample code & Library version: 4a8bc87e0d

Flow Control



GPIO Flow Chart



Demo API Function

This documents contains GPIO / WDT / EEPROM / I2C/SPI configuration API for Fintek IC. The support list is below:

- GPIO
 - F81866/F81803/F81768/F71869A/F81504/F81508/F81512/F75113(LPC,I2C)*/
F75114/F75115/F81801/F81804/F81966/F81532A/F81534A/F81535/F81536/
F81216E/F75111/F81968
- Watch Dog
 - F81866/F81803/F81768/F71808A/F71869A/F75113/F81801/F81804/F81966/
F81216/F81216AD/F81216H/F81216E/F75111/F81968
- EEPROM
 - F81504/F81508/F81512
- I2C
 - F81866 / F75115
- SPI
 - F75115
- UART
 - F75115
- PWM
 - F75115 / F81866

Some chip likes F75113 should refer ch4.2.4 section "I2C/SMBUS bus driver" for more help.

2.1 GPIO Function:

2.1.1 Function List

- int _EnableGPIO(unsigned int uldx, eGPIO_Mode eMode)
- int _SetGpioOutputDataIdx(unsigned int uldx, unsigned int uValue)
- int _GetGpioOutputDataIdx(unsigned int uldx, unsigned int *uValue)
- int _GetGpioInputDataIdx(unsigned int uldx, unsigned int *uValue)
- int _SetGpioDriveEnable(unsigned int uldx, eGPIO_Drive_Mode eMode)
- int _SetGpioOutputEnableIdx(unsigned long uldx, eGPIO_Direction eMode)
- int _SetGpioPullMode(unsigned int uldx, eGPIO_Pull_Mode eMode)

- int _GetGpioPullMode(unsigned int uldx, eGPIO_Pull_Mode *eMode)
- int _SetGpioGroupOutputDataIdx(unsigned int uldx, unsigned int uValue)
- int _GetGpioGroupInputDataIdx(unsigned int uldx, unsigned int *uValue)
- int _EnableGpioInt(unsigned int set, unsigned int en_bit, unsigned int irq)
- int _DisableGpioInt(unsigned int set, unsigned int dis_bit)
- int _GetGpioIntStatus(unsigned int set, unsigned int *bit_status)
- int _ClearGpioIntStatus(unsigned int set, unsigned int clear_bit_status)

2.1.2 Value Description

Define	Value	Description
_EnableGPIO	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)
	eMode	Enum eGPIO_Mode {eGPIO_Mode_Disable, eGPIO_Mode_Enable}
_SetGpioOutputDataIdx	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)
	uValue	Set GPIO Output value: 0: Low 1:High
_SetGpioGroupOutputDataIdx	set	GPIO set (ex. GPIO3x => set=0x3)
	uValue	Set max to 8 pin data
_GetGpioOutputDataIdx	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)
	*uValue	Read GPIO Current Output value: 0: Low 1:High
_GetGpioInputDataIdx	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)
	*uValue	Read GPIO Current Input value: 0: Low 1:High
_GetGpioGroupInputDataIdx	set	GPIO set (ex. GPIO3x => set=0x3)
	*uValue	Get max to 8 pin data
_SetGpioDriveEnable	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)
	eMode	Enum eGPIO_Drive_Mode {eGPIO_Drive_Mode_OpenDrain, eGPIO_Drive_Mode_Pushpull}
_GetGpioDriveEnable	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)
	*eMode	EEnum eGPIO_Drive_Mode {eGPIO_Drive_Mode_OpenDrain, eGPIO_Drive_Mode_Pushpull}
_SetGpioOutputEnableIdx	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)
	eMode	Enum eGPIO_Direction {eGPIO_Direction_In, eGPIO_Direction_Out}
_GetGpioOutputEnableIdx	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)
	*eMode	Enum eGPIO_Direction {eGPIO_Direction_In, eGPIO_Direction_Out}
_SetGpioPullMode	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)

	eMode	Enum eGPIO_Pull_Mode {eGPIO_Pull_Low, eGPIO_Pull_High, eGPIO_Pull_Disable}
_GetGpioPullMode	uldx	GPIO Index (ex. GPIO16 => uldx=0x16)
	*eMode	Enum eGPIO_Pull_Mode {eGPIO_Pull_Low, eGPIO_Pull_High, eGPIO_Pull_Disable}
_EnableGpioInt	set	Enable GPIO interrupt with special set
	en_bit	Enable GPIO interrupt with special set & bit
	irq	IRQ channel
_DisableGpioInt	set	Disable GPIO interrupt with special set
	dis_bit	Disable GPIO interrupt with special set & bit
_GetGpioIntStatus	set	Get GPIO interrupt status with special set
	*status	Get pins status
_ClearGpioIntStatus	set	Clear GPIO interrupt status with special set
	status	Clear pins status

2.2 WDT Function:

2.2.1 Function List

- 1st WDT control
 - int SetWdtDisable()
 - int SetWdtEnable()
 - int SetWdtConfiguration(int iTimerCnt, int iClkSel, int iPulseMode, int iUnit, int iActive, int iPulseWidth)
 - int GetWdtTimeoutStatus(int *Status, int* RemainTime)
- Other WDT control (by index, 1st WDT = 0)
 - int SetWdtIdxConfiguration(int idx, int iTimerCnt, int iClkSel, int iPulseMode, int iUnit, int iActive, int iPulseWidth)
 - int SetWdtIdxEnable(int idx)
 - int SetWdtIdxDisable(int idx)
 - int GetWdtIdxTimeoutStatus(int idx, int *Status, int *RemainTime)

2.2.2 Value Description

Define	Value	Description
SetWdtConfiguration	iTimerCnt	0-255 (second or minute program by "iUnit")
	iClkSel	Watchdog timer clock. -1: Remain default value setting by BIOS (recommend) 0: 10Hz clock divided by CLKIN. 1: Internal 10Hz clock.
	iPulseMode	Watchdog output mode. -1: Remain default value setting by BIOS (recommend) 0: Select level output mode. 1: Select pulse output mode.
	iUnit	Watchdog unit select. -1: Remain default value setting by BIOS (recommend) 0: Select second. 1: Select minute.
	iActive	Watchdog output polarity of WDTRST -1: Remain default value setting by BIOS (recommend) 0: low active. 1: high active.
	iPulseWidth	Watchdog output pulse width -1: Remain default value setting by BIOS (recommend) 0: Select pulse width of 1 ms. 1: Select pulse width of 25 ms. 2: Select pulse width of 125 ms. 3: Select pulse width of 5 sec.
SetWdtDisable		Disable WDT & clear WDTMOUT_STS
SetWdtEnable		Enable WDT & WDTMOUT_STS
GetWdtTimeoutStatus	*Status	0: Timeout event is not occurred 1: Timeout event is occurred
	* RemainTime	Remain time of WDT

2.3 I2C function:

2.3.1 Function List

- int SelectI2CChannel(unsigned int ch)
- int WriteI2CData(unsigned int addr, unsigned int data)
- int ReadI2CData(unsigned int addr, unsigned int *data)
- int I2C_Start(int idx)
- int I2C_Write_Data(int idx, uint8_t data)
- int I2C_Read_Data(int idx, uint8_t *data, int nack)
- int I2C_Stop(int idx)

2.3.2 Value Description

Define	Value	Description
SelectI2CChannel	ch	Target channel
WriteI2CData	dev	Set slave address
	addr	Set slave register/command
	data	Set write data to slave
ReadI2CData	dev	Set slave address
	addr	Set slave register/command
	*data	Read data from slave
I2C_Start	idx	Target channel
I2C_Write_Data	idx	Target channel
	data	Write data
I2C_Read_Data	idx	Target channel
	*data	Read data
	nack	Reply ack / nack to device
I2C_Stop	idx	Target channel

2.4 PWM function:

2.4.1 Function List

- int GetPWMCount()
- int SetPWMSioRawConfig(int idx, uint32_t val)
- int GetPWMSioRawConfig(int idx, uint32_t *val)
- int GetPWMFreqSupport_Size(int idx)
- int GetPWMFreqSupport_List(int idx, uint32_t * data, int in_size)
- int SetPWM_Freq_Div_Percentage(int idx, int freq, int div, int percentage)
- int GetPWM_Freq_Div_Percentage(int idx, int *freq, int *div, int *percentage)
- int GetPWMMaxDivider(int idx, int *max_div)

2.4.2 Value Description

Define	Value	Description
GetPWMCount		Get max PWM sets for target IC
SetPWMSioRawConfig		
	idx	Target PWM index
	val	Set raw value
GetPWMSioRawConfig		
	idx	Target PWM index
	*val	Get raw value
GetPWMFreqSupport_Size		Get PWM support base frequency list count
	idx	Target PWM index
GetPWMFreqSupport_List		Get PWM support base frequency list table
	idx	Target PWM index
	*data	Frequency table
	in_size	Pass size to SDK (size divided by int)
SetPWM_Freq_Div_Percentage		Set PWM Freq/Div/Duty
	idx	Target PWM index
	freq	Set PWM Frequency
	div	Set Frequency divider
	percentage	Set PWM Duty percentage
GetPWM_Freq_Div_Percentage		Get PWM Freq/Div/Duty
	idx	Target PWM index
	*freq	Get PWM Frequency

	*div	Get Frequency divider
	*percentage	Get PWM Duty percentage
GetPWMMaxDivider		Get PWM max divider
	idx	Target PWM index
	*max_div	Get PWM max divider

2.5 SPI function:

2.5.1 Function List

- int SetSpiCsEn(int idx, int en)
- int ReadSpiData(int idx, unsigned char *data)
- int WriteSpiData(int idx, unsigned char data)

2.5.2 Value Description

Define	Value	Description
SetSpiCsEn	idx	SPI index
	en	0: Chip select low, 1: Chip select high
ReadSpiData	idx	SPI index
	*data	Read 1 byte data from device (MISO)
WriteSpiData	idx	SPI index
	data	Write 1 byte data to device (MOSI)

2.6 UART function:

2.6.1 Function List

- int UART_GetMaxChannel()
- int UART_SetBaudRate(uint32_t idx, uint32_t baudrate)
- int UART_SetDTR(uint32_t idx, uint32_t en)
- int UART_SetRTS(uint32_t idx, uint32_t en)
- int UART_GetDCD(uint32_t idx, uint32_t * status)
- int UART_GetDSR(uint32_t idx, uint32_t * status)
- int UART_GetCTS(uint32_t idx, uint32_t * status)
- int UART_GetRI(uint32_t idx, uint32_t * status)
- int UART_TX(uint32_t idx, uint8_t * data, uint8_t len)
- int UART_RX(uint32_t idx, uint8_t * data, uint8_t data_len, uint8_t * read_len, uint32_t timeout)

2.6.2 Value Description

Define	Value	Description
UART_GetMaxChannel		Get max UART count of target IC
UART_SetBaudRate	idx	UART index
	baudrate	Set desired baudrate
UART_SetDTR	idx	UART index
	en	Set DTR enable (Active Low)
UART_SetRTS	idx	UART index
	en	Set DTR enable (Active Low)
UART_GetDCD	idx	UART index
	*status	Get DCD status (Active Low)
UART_GetDSR	idx	UART index
	*status	Get DSR status (Active Low)
UART_GetCTS	idx	UART index
	*status	Get CTS status (Active Low)
UART_GetRI	idx	UART index
	*status	Get Ring-In status (Active Low)
UART_TX	idx	UART index
	*data	UART TX data array
	len	UART TX data array size

UART_RX	idx	UART index
	*data	UART RX data array
	data_len	UART RX data array size
	*read_len	UART current read data size
	timeout	RX max timeout (ms)

Linux Demo code Guide :

3.1 Files Description

We provide x86, x86_64 & Android SDK. It's named with:

- fintek_demo_release_i686-<ver>.tar.gz
- fintek_demo_release_x86_64-<ver>.tar.gz
- fintek_demo_release-<ver>-armv7l.tar.gz

It contained:

- libfintek_api.a
- demo_spi.c
- demo_wdt.c
- demo_gpio.c
- demo_raspberry.c
- demo_id.c
- demo_eeprom.c
- demo_i2c.c
- demo_i2c_protocol.c
- demo_pwm.c
- demo_uart.c
- fintek_api.h
- Makefile

Example :

4.1 Decompress & compile demo app

Decompress `fintek_demo_release_<arch>-<version>.tar.gz` to your working directory.

For examples, We'll decompress to “/home/code/demo” with arch:x86 version:4a8bc87e0d

1. `mkdir /home/code/demo`
2. `cd /home/code/demo`
3. `cp <somewhere>/fintek_demo_release-4a8bc87e0d-i686.tar.gz .`
4. `tar xf fintek_demo_release-4a8bc87e0d-i686.tar.gz`
5. `make`
6. use demo code to test on your platform.
 - 4.2.3 for simple command to control GPIO
 - 4.3.2 for simple command to control WDT
 - 4.4.2 for simple command to control PWM
 - 4.5.2 for simple command to Read/Write I2C
 - 4.6.2 for simple command to Read/Write SPI Flash
 - 4.7.2 for simple command to Case-open demo
 - 4.8.2 for simple command to UART demo

If the demo application show the message “Cant Found any Fintek SIO Product”. We should change the first parameter in demo source code with `init_fintek_sio()`. It can be referenced from file “fintek_api.h”.

```
typedef enum {
    eSIO_TYPE_SIO = 0,
    eSIO_TYPE_F71808A = eSIO_TYPE_SIO,
    eSIO_TYPE_F81866,
    eSIO_TYPE_F81803,
    eSIO_TYPE_F81768,

    eSIO_TYPE_PCI,
    eSIO_TYPE_PCI_F81504 = eSIO_TYPE_PCI,
    eSIO_TYPE_PCI_F81508,
    eSIO_TYPE_PCI_F81512,

    eSIO_TYPE_UNKNOWN,
    eSIO_TYPE_INVALID,
} eSIO_TYPE;
```

4.2 GPIO Read/Write/Interrupt

- Please reference the function gpio_demo() within demo code “demo_gpio.c” for the detail.
- The following demo code should run with privileged user (root).

4.2.1. GPIO Read Example with API (Read from GPIO06):

```
init_fintek_sio(eSIO_TYPE_F81866, 0, &sio_data)
ActiveSIO(sio_data.ic_port, sio_data.key);
CHECK_RET(_EnableGPIO(0x06, eGPIO_Mode_Enable));
CHECK_RET(_SetGpioOutputEnableIdx( 0x06, eGPIO_Direction_In));
CHECK_RET(_GetGpioInputDataIdx( 0x06, &data));
DeactiveSIO(sio_data.ic_port);
```

4.2.2. GPIO Write Example with API (Write to GPIO06 with High):

```
init_fintek_sio(eSIO_TYPE_F81866, 0, &sio_data)
ActiveSIO(sio_data.ic_port, sio_data.key);
CHECK_RET(_EnableGPIO(0x06, eGPIO_Mode_Enable));
CHECK_RET(_SetGpioOutputEnableIdx( 0x06, eGPIO_Direction_Out));
```

```
CHECK_RET(_SetGpioDriveEnable( 0x06 , eGPIO_Drive_Mode_OpenDrain));  
CHECK_RET(_SetGpioOutputDataIdx( 0x06 , 1));  
DeactiveSIO(sio_data.ic_port);
```

4.2.3. GPIO Write with demo_gpio (Write to GPIO80 with High / Read GPIO81):

The command line format of demo_gpio:

```
./demo_gpio <idx> <dir> <mode> <value>
```

<idx>: index of gpio, e.g., 0x80.

<dir>: 0 for input. 1 for output

<mode>: 0 for open-drain. 1 for Push-pull

<value>: 0 for output low. 1 for output high

We'll use following command to control GPIO80 with Push-Pull & High Level

```
./demo_gpio 0x80 1 1 1
```

And read GPIO81 with

```
./demo_gpio 0x81 0 0 0
```

4.2.4. GPIO Interrupt handle with demo_gpio (GPIO86):

1. Install driver in gpio_interrupt_driver folder.
2. We can refer gpio_int_demo() in demo_gpio.c or demo_gpio_int.c

4.2.5. GPIO with demo_raspberry (PIN40 → GPIO27):

The command line format of demo_raspberry, it's most same with demo_gpio:

```
./demo_raspberry <pin> <dir> <mode> <value>
```

<pin>: pin of F75115 (using raspberry index)

```

root@code-desktop: /home/code [114x23]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)
^C
root@code-desktop: /home/code/ddd/old/hpeter/fintek/fintek_api# ./demo_gpio 0x86 5
FINTEK Lib Version: 557ec5be85
idx: 0x86 => set:8, count:6, irq: 5
Dir: 0x0 => IN
cnt: 0, data: 0
cnt: 1, data: 1
cnt: 2, data: 0
cnt: 3, data: 0
cnt: 4, data: 0
cnt: 5, data: 1
cnt: 6, data: 0
cnt: 7, data: 0
cnt: 8, data: 0

-----
 5:      1      0      66207      0      IO-APIC      5-edge      fintek-gpio-int
root@code-desktop: /home/code# cat /proc/interrupts | grep fintek
 5:      1      0      66208      0      IO-APIC      5-edge      fintek-gpio-int
root@code-desktop: /home/code# cat /proc/interrupts | grep fintek
 5:      1      0      66216      0      IO-APIC      5-edge      fintek-gpio-int
root@code-desktop: /home/code#
[0] 0:bash- 1:bash 2:bash* 3:bash "code-desktop" 17:18 17- -- -19

```

4.2.4. Notice

I2C/SMBUS bus driver:

All I2C/SMBUS device must dependent on i2c bus driver. For example, we need modprobe “i2c_i801” for bus driver in Intel platform before using any demo code.

The F75113 support LPC/I2C protocol. We can choose protocol type when calling “init_fintek_sio”, passing “eSIO_TYPE_F75113” with LPC mode and “eSIO_TYPE_F75113_I2C” with I2C mode.

4.3 WDT Set and Monitor

- Please reference the function watchdog_demo() within demo code "demo_wdt.c" for the detail.
- The following demo code should run with privileged user (root).

4.3.1. WDT Set and Monitor Example (Count down for 10 second)

```
init_fintek_sio(eSIO_TYPE_F81866, 0 ,&sio_data)
ActiveSIO(sio_data.ic_port, sio_data.key);
SetWdtDisable();
// countdown 10s, other parameter using original value
SetWdtConfiguration(10, -1, -1, 0, -1, -1);
DeactiveSIO(sio_data.ic_port);

while( !GetWdtTimeoutStatus(&status, &timer) ) {
    static int old_timer = 0;

    if(old_timer != timer) {
        fprintf(stderr, "status:%d, timer:%d\n", status, timer);
        old_timer = timer;
    }

    if(status) {
        fprintf(stderr, "status:%d, timer:%d\n", status, timer);
        break;
    }

    usleep(10000);
}

DeactiveSIO(sio_data.ic_port);
```

4.3.2. WDT Set and Monitor Example with demo_wdt

./demo_wdt

4.3.3. Multi-WDT control

If can be referenced by function “watchdog2_demo()” in demo_wdt.c

4.4 PWM control

- Please reference the function `pwm_demo()` within demo code “`demo_pwm.c`” for the detail.
- The following demo code should run with privileged user (root).

4.4.1. PWM API example

See demo code “`demo_pwm.c`”

4.4.2. PWM example with `demo_pwm` on F75115

Example1. Set PWM1 with frequency 23.437KHz(approx.), duty 50%

$(24\text{MHz} / (1023+1) = 23.437\text{KHz})$

`./demo_pwm 1 1 24000000 1023 50`

Example2. Set PWM1 with frequency 1.2MHz(approx.), duty 80%

$(12\text{MHz} / (9+1) = 1.2\text{MHz})$

`./demo_pwm 1 1 12000000 9 80`

4.5 I2C

- Please reference the function `i2c_demo()` within demo code “demo_i2c.c” for the detail.
- We provide customize I2C command in `i2c_demo()` with demo code “demo_i2c_protocol_.c”.
- The following demo code should run with privileged user (root).

4.5.1. I2C Read & Write Example (Read from slave 0x20 & Write 0xaa to slave from ch 0)

```
unsigned int tmp;
```

```
init_fintek_sio(eSIO_TYPE_F75115_HID, 0, &sio_data)
```

```
ActiveSIO(sio_data.ic_port, sio_data.key);
```

```
status = SelectI2CChannel(0); // success with status=0, others are failed.
```

```
status = ReadI2CData(0x20, 0x10, &tmp); // read from slave 0x20 with reg 0x10
```

```
status = WriteI2CData(0x20, 0x10, 0xaa); // write 0xaa to slave 0x20 with reg 0x10
```

```
DeactiveSIO(sio_data.ic_port);
```

4.5.2. I2C Read & Write Example with demo_i2c

Example1. Read from slave 0xa0 by channel 0 with reg 0x10

```
./demo_i2c r 0x00 0xa0 0x10
```

Example2. Write to slave 0xa0 with data 0xaa by channel 0 with reg 0x10

```
./demo_i2c w 0x00 0xa0 0x10 0xaa
```

Example3. Dump EEPROM by channel 0

```
./demo_i2c d 0x00 0xa0
```

4.6 SPI

- Please reference the function spi_flash_demo() within demo code “demo_spi.c” for the detail.
- The following demo code should run with privileged user (root).

4.6.1. Read / Write SPI with SPI channel 0

```
unsigned char tmp;
sFintek_sio_data sio_data;

init_fintek_sio(eSIO_TYPE_F75115_HID, 0, &sio_data)
ActiveSIO(sio_data.ic_port, sio_data.key);

SetSpiCsEn(0, 1); // force CS high
SetSpiCsEn(0, 0); // set CS low

WriteSpiData(0, 0x03); // write 0x03 to MOSI
ReadSpiData(0, &tmp); // read data from MISO

SetSpiCsEn(0, 1); // set CS high

DeactiveSIO(sio_data.ic_port);
```

4.6.2. SPI Flash Read & Write Example with demo_spi

Example1. Read SPI Flash addr = 0x1000
./demo_spi r 0 0x1000

Example2. Write String “abcde” to SPI Flash addr = 0x1000
./demo_spi w 0 0x1000 abcde

4.7 Case open detect

- Please reference the function `demo_caseopen()` within demo code “demo_caseopen.c” for the detail.
- The following demo code should run with privileged user (root).

4.7.1. Read case open status

```
./demo_caseopen
```

4.7.2. Read and clear case open status

```
./demo_caseopen 1
```

4.8 UART control

- Please reference the function `uart_demo()` within demo code “demo_uart.c” for the detail.
- The following demo code should run with privileged user (root).

4.4.1. UART API example

See demo code “demo_uart.c”

4.4.2. UART example with demo_uart on F75115

Example. Use UART channel 0 to do self test. Please connect TX/RX, RTS/CTS, DTR/DSR.
./demo_uart 0

```
root@code-ms7c82:/home/code/ddd/old/hpeter/fintek/docker/fintek_api/fintek_api# ./demo_uart 0
FINTEK Lib Version: b1f9034262
Please connect TX/RX, DTR/DSR, RTS/CTS for loopback test
set baudrate 115200 ok
TX ok
RX ok, read_len: 32
TX:
19 BC 6A 8C 86 93 FD 86 4B 99 25 15 70 0B 2E 51 BC 5F 71 98 E3 7F 82 BC 07 CA EF BF BE DB 73 D7
RX:
19 BC 6A 8C 86 93 FD 86 4B 99 25 15 70 0B 2E 51 BC 5F 71 98 E3 7F 82 BC 07 CA EF BF BE DB 73 D7
Set RTS disable, check CTS: 0 ok
Set RTS enable, check CTS: 1 ok
Set DTR disable, check DSR: 0 ok
Set DTR enable, check DSR: 1 ok
read DCD: 0, ok
read DSR: 1, ok
read CTS: 1, ok
read RI: 0, ok
root@code-ms7c82:/home/code/ddd/old/hpeter/fintek/docker/fintek_api/fintek_api#
```