Fintek

Linux API Guide

v1.22

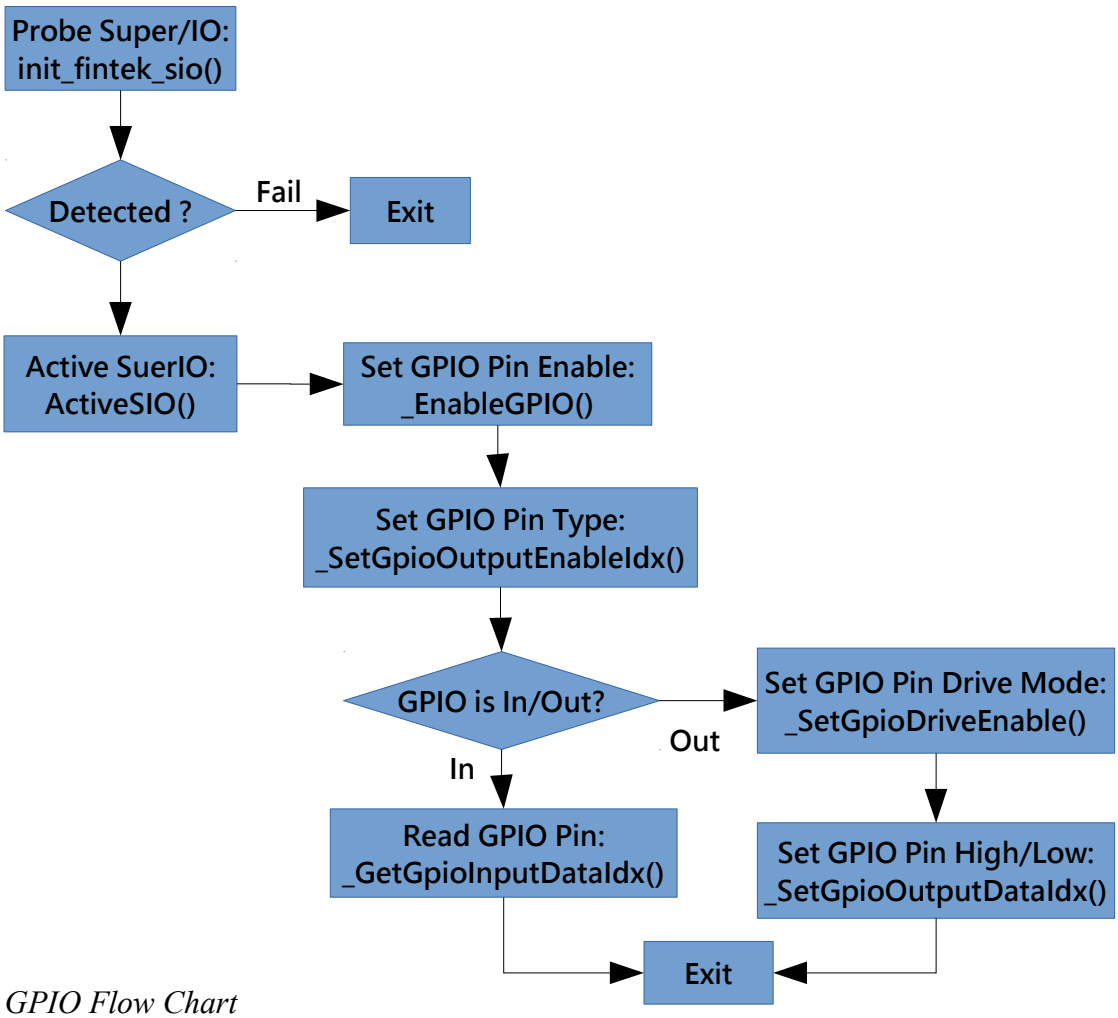
Aug 20, 2018

Datasheet Revision History

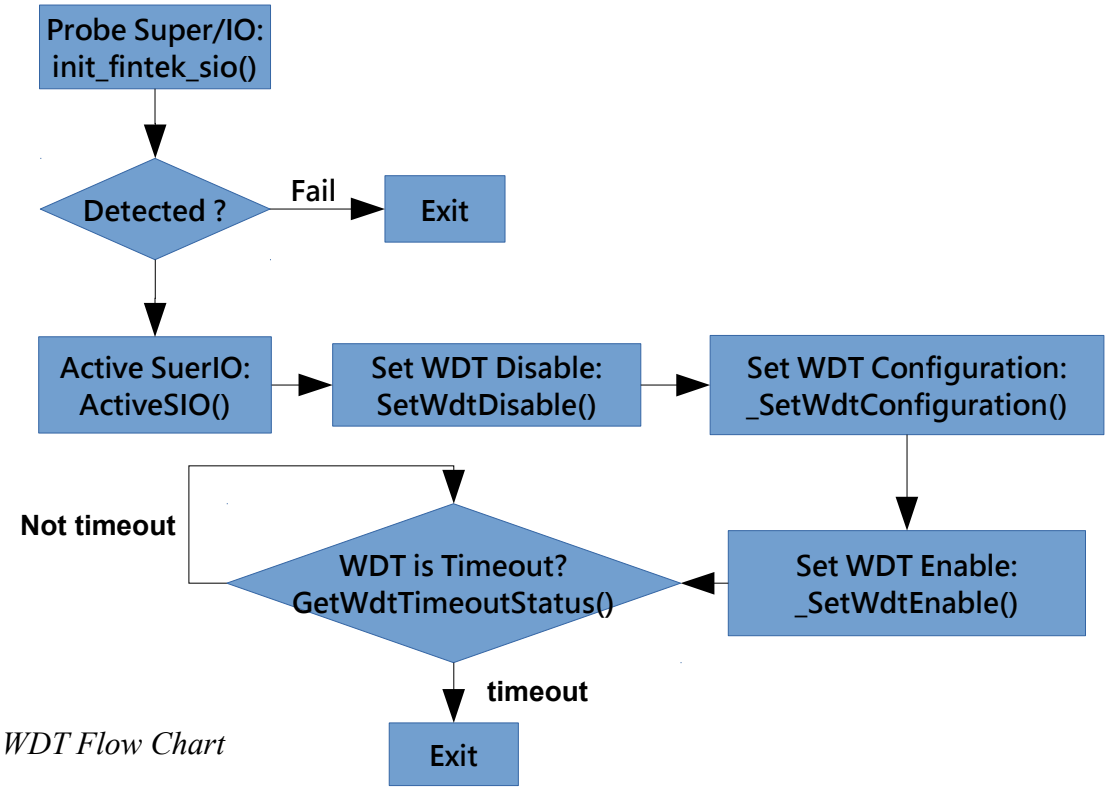
Date	Version	Revision History
2014/9/22	V1.00	1. Initial version.
2014/9/24	V1.01	1. SuperIO detection function modified. 2. Function SetWdtConfiguration parameter modified. Sample code & Library version: 854091fe7
2015/6/9	V1.02	1. Add support for F81768 SuperIO. 2. Static Library released with x86/x86_64. 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. Add support for F75115 4. Using older GCC to make library for older system compatibility 5. Some IC (like F75115) support “eGPIO_Direction_BiDirectional” for GPIO direction control – SetGpioOutputEnableIdx()/GetGpioOutputEnableIdx(). Please read ch4.2.4 for future information. Sample code & Library version: 6fe4d90695

Flow Control



GPIO Flow Chart



WDT 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/F75114/F81801/F81804/F81966/F81532A/F81534A/F81535/F81536
- Watch Dog
  - F81866/F81803/F81768/F71808A/F71869A/F75113/F81801/F81804/F81966
- EEPROM
  - F81504/F81508/F81512
- I2C
  - F81866
- SPI
  - F75115

2.1 GPIO Function:

2.1.1 Function List

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

2.1.2 Value Description

Define	Value	Description
_EnableGPIO	uIdx	GPIO Index (ex. GPIO16 => uIdx=0x16)
	eMode	Enum eGPIO_Mode {eGPIO_Mode_Disable, eGPIO_Mode_Enable}
_SetGpioOutputDataIdx	uIdx	GPIO Index (ex. GPIO16 => uIdx=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	uIdx	GPIO Index (ex. GPIO16 => uIdx=0x16)
	*uValue	Read GPIO Current Output value: 0: Low 1:High
_GetGpioInputDataIdx	uIdx	GPIO Index (ex. GPIO16 => uIdx=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	uIdx	GPIO Index (ex. GPIO16 => uIdx=0x16)
	eMode	Enum eGPIO_Drive_Mode {eGPIO_Drive_Mode_OpenDrain, eGPIO_Drive_Mode_Pushpull}
_GetGpioDriveEnable	uIdx	GPIO Index (ex. GPIO16 => uIdx=0x16)
	*eMode	EEnum eGPIO_Drive_Mode {eGPIO_Drive_Mode_OpenDrain, eGPIO_Drive_Mode_Pushpull}
_SetGpioOutputEnableIdx	uIdx	GPIO Index (ex. GPIO16 => uIdx=0x16)
	eMode	Enum eGPIO_Direction {eGPIO_Direction_In, eGPIO_Direction_Out, eGPIO_Direction_BiDirectional}
_GetGpioOutputEnableIdx	uIdx	GPIO Index (ex. GPIO16 => uIdx=0x16)
	*eMode	Enum eGPIO_Direction {eGPIO_Direction_In, eGPIO_Direction_Out, eGPIO_Direction_BiDirectional}
_SetGpioPullMode	uIdx	GPIO Index (ex. GPIO16 => uIdx=0x16)
	eMode	Enum eGPIO_Pull_Mode {eGPIO_Pull_Low, eGPIO_Pull_High, eGPIO_Pull_Disable}
_GetGpioPullMode	uIdx	GPIO Index (ex. GPIO16 => uIdx=0x16)
	*eMode	Enum eGPIO_Pull_Mode {eGPIO_Pull_Low, eGPIO_Pull_High, eGPIO_Pull_Disable}

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 EEPROM function:

2.3.1 Function List

- int InitEEPROM(void)
- int SetEEPROMValue(unsigned int addr, unsigned int data)
- int GetEEPROMValue(unsigned int addr, unsigned int *data)

2.3.2 Value Description

Define	Value	Description
InitEEPROM		Initialize EEPROM Environment
SetEEPROMValue	addr	EEPROM target register address.
	data	EEPROM data you want to write.
GetEEPROMValue	addr	EEPROM target register address.
	*data	EEPROM data you want to read.

2.4 I2C function:

2.4.1 Function List

- int GetHWMONAddr(unsigned int *addr)
- int SelectI2CChannel(unsigned int ch)
- int WriteI2CData(unsigned int addr, unsigned int data)
- int ReadI2CData(unsigned int addr, unsigned int *data)

2.4.2 Value Description

Define	Value	Description
GetHWMONAddr	*addr	Get the H/W Monitor io address to manual operation
SelectI2CChannel	ch	Select the I2C channel with ch
WriteI2CData	addr	Set slave address
	data	Set write data to slave
ReadI2CData	addr	Set slave address
	*data	Read data from slave

2.4.3 Notice

The SelectI2CChannel() may had different operation with different Fintek chips.

For F81866

- ch 0: PIN 67/68
- ch 1: PIN 71/76
- ch 2: PIN 61/62

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)

Linux Demo code Guide：

3.1 Files Description

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

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

It contained:

- libfintek_api.a
- demo_spi.c
- demo_wdt.c
- demo_gpio.c
- demo_id.c
- demo_eeprom.c
- demo_i2c.c
- fintek_api.h
- Makefile

Support IC List:

- GPIO
  - F81866/F81803/F81768/F71869A/F81504/F81508/F81512/F75113/F75114/F81801/F81804/F81966/F81532A/F81534A/F81535/F81536
- Watch Dog
  - F81866/F81803/F81768/F71808A/F71869A/F75113/F81801/F81804/F81966
- EEPROM
  - F81504/F81508/F81512
- I2C
  - F81866
- SPI
  - F75115

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_32 version:6fe4d90695

1. mkdir /home/code/demo
2. cd /home/code/demo
3. cp <somewhere>/fintek_demo_release-6fe4d90695-i686.tar.gz .
4. tar xf fintek_demo_release-6fe4d90695-i686.tar.gz
5. make
6. use demo_gpio & demo_wdt & demo_eeprom & demo_i2c 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 Read/Write EEPROM
   - 4.5.2 for simple command to Read/Write I2C
   - 4.6.2 for simple command to Read/Write SPI Flash

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

- 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. Notice

Bidirectional Mode:

Some GPIO in/output only support bidirectional mode, like F75115 GPIO2x~6x. It's in/output data is same register. So we need to set it output high & open-drain when input mode. If we had do this operation, the input value will be wrong.

Group Output Mode:

Some GPIO output only support group mode, like F75115 GPIO2x~6x. It's in/output data is same register. We can't read it with original output data, it's input data only when read. So user need to save the GPIO set data in program and re-apply it when output value change.

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 EEPROM Read & Write

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

4.4.1. EEPROM Read & Write Example (Read from 0x30 & Write 0x01 to reg 0x21)

```
unsigned int data;

init_fintek_sio(eSIO_TYPE_PCI_F81504, 0 ,&sio_data)
ActiveSIO(sio_data.ic_port, sio_data.key);

status = InitEEPROM(); // success with status=0, others are failed.
status = GetEEPROMValue(0x30, &data); // read from 0x30
status = SetEEPROMValue(0x21, 0x01); // write reg:0x21 with value:0x01

DeactiveSIO(sio_data.ic_port);
```

4.4.2. EEPROM Read & Write Example with demo_gpio

Example1. Read from reg 0x30:

./demo_eeprom r 0x30

Example2. Write reg 0x02 with value 0xff:

./demo_eeprom w 0x02 0xff

Example3. Dump all EEPROM Data:

./demo_eeprom d

4.5 I2C

- Please reference the function i2c_demo() within demo code “demo_i2c.c” for the detail.
- 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_F81866, 0 ,&sio_data)
ActiveSIO(sio_data.ic_port, sio_data.key);

status = SelectI2CChannel(0); // success with status=0, others are failed.
status = ReadI2CData(0x20, &tmp); // read from slave 0x20
status = WriteI2CData(0x20 0xaa); // write 0xaa to slave 0x20

DeactiveSIO(sio_data.ic_port);
```

4.5.2. I2C Read & Write Example with demo_i2c

Example1. Read from slave 0x20 by channel 0

```
./demo_i2c r 0x00 0x20
```

Example2. Write to slave 0x20 with data 0xaa by channel 0

```
./demo_i2c w 0x00 0x20 0xaa
```

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, 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 0 0x1000 0
```

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

```
./demo_spi 1 0x1000  abcde
```