RT568 Porting Guide

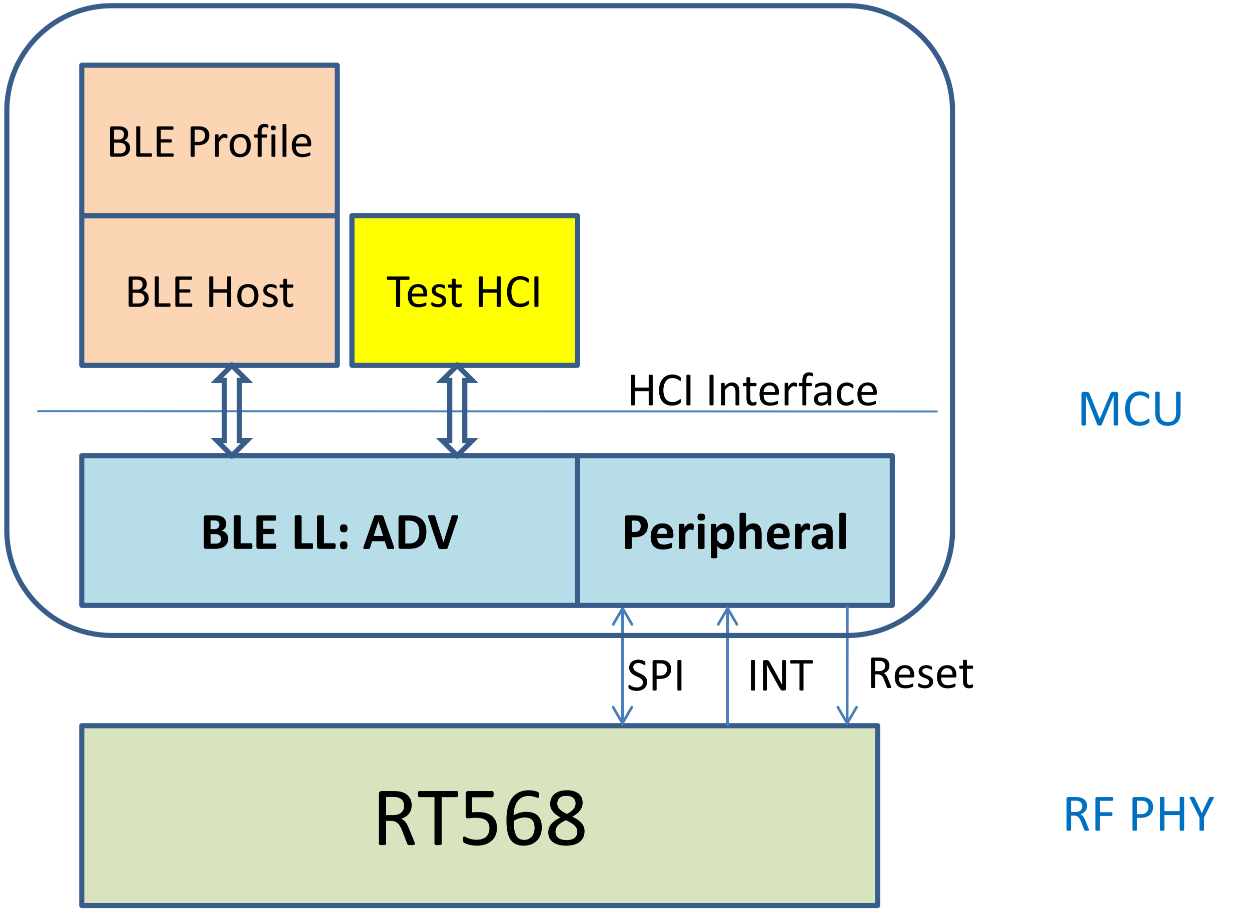
Rafael Micro

Agenda

- RT568 ADV example code architecture
- MCU Requirement
- RF Initialize Flow
  - RF external reset
  - SIP & IO mapping flow
- MCU Porting – step by step
- Check Point of each Porting Stage

(1) RT568 SDK Architecture

Rafael sample codes includes: (a) MCU peripheral functions (b) ADV example



(2) MCU Requirement

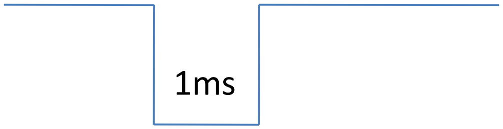
- MCU clock >= 48MHz
- SPI > 8MHz
  - Recommend 16MHz, or 12MHz
  - Mode0, MSB first, 8-bit width, master mode
- SPI with DMA
- GPIO interrupt supports Level HIGH trigger
- GPIO interrupt is highest priority
- When BLE connection established, application task should not occupy CPU >30ms

(3) RF Initialize Flow – main()

- SYS_Init()
- RF_Open()
  - Enable MCU Timer, delay 25ms for RF power on
  - Set MCU GPIO pin to reset RF, delay 50ms for reset complete
  - Do SPI I/O re-mapping: RF_SpiIoMapping()
    - This is **MUST**. After IO re-mapping, MCU is able to control RF
  - Initial SPI PDMA
  - Call RF_Init() to initialize RF
- Call BleApp_Init() to activate BLE ADV

(3-1) RF External Reset

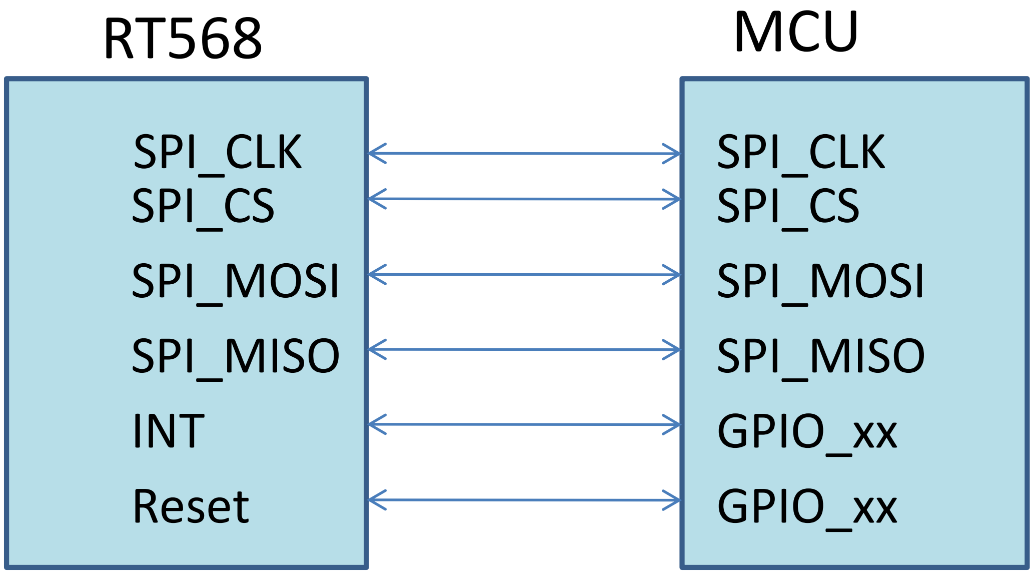
- RF provide an external reset pin for MCU to fully reset RF
- MCU assign one GPO pin, connect to RF external reset pin. Initial value set High
- It is active LOW reset:
  - High->Low(1ms)->High
  - Wait 50ms for reset complete



- The behavior is same as Power-On Reset (POR)
  - MUST do I/O re-mapping after POR or external reset

(3-2) SIP & I/O re-mapping

- For easily SIP with MCU, user can dynamic configure RF SPI pins. Please reference RF_SpiIoMapping() code
- Details please refer to “RT568_IO-remapping_vx.x.pdf”
- If you have not decided SIP pin allocation, you can use SPI_IO_ORDER=1 (normal pin connection).



(4) MCU porting – step by step

- Replace functions in these files
  - main.c
  - mcu_definition.h
  - porting_spi.c
  - porting_misc.c

(4-1) main.c

- Replace SYS_Init()
  - Set system clock
  - Set UART port
- Replace UART related code

(4-2) mcu_definition.h

- #include MCU header file to refer to peripheral and core
  - Example: #include “NuMicro.h”
- Define Tiny_Delay as microsecond delay function(or macro)
  - Example: #define Tiny_Delay(x) delay_us(x)
- Define interrupt enable/disable function
  - Example: #define InterruptDisable __disable_irq
    #define InterruptEnable __enable_irq

(4-3) porting_spi.c

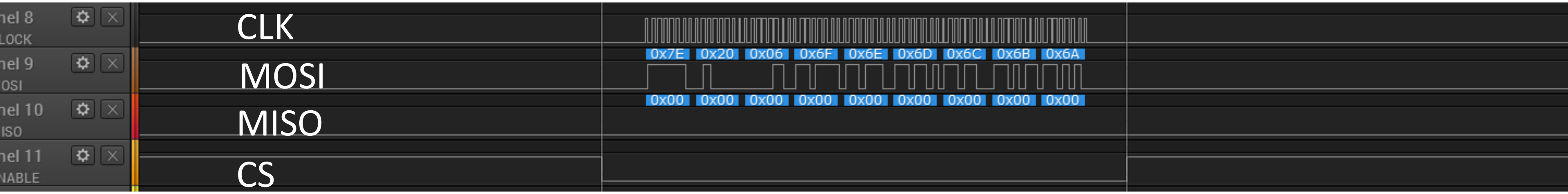
- SPI command format to write/read RT568, please see "(SW) RT568 API User Guide V1.0 Preliminary.pdf" – chapter 3 and 5
- MCU_SpiInit()
  - Initialize SPI and enable SPI module
  - Master mode, mode0, use 16M or 12M clock, 8-bit width, MSB first
- SPI_1BYT_SetRx(uint8_t regAddr)
  - Read RF one byte register
- SPI_1BYT_SetRx_Isr(uint8_t regAddr)
  - Read RF one byte register inside ISR
- SPI_1BYT_SetTx(uint8_t regAddr, uint8_t u8SrcData)
  - Set RF one byte register

- SPI_1BYT_SetTx_Isr(uint8_t regAddr, uint8_t u8SrcData)
  - Set RF one byte register inside ISR
- SPI_2BYT_SetTx_Isr(uint8_t regAddr, uint8_t u8SrcData)
  - Set RF two bytes register inside ISR
- SPI_PDMA_Init()
  - Initial SPI DMA and enable module
- SPI_PDMA_waitFinish()
  - Wait for SPI DMA complete
- SPI_PDMA_SetRx_Isr(uint8_t regAddr, uint32_t u32DstAddr, uint32_t u32TransCount)
  - Use SPI DMA read register or data from RF
- SPI_PDMA_SetTx(uint8_t regAddr, uint32_t u32SrcAddr, uint32_t u32TransCount)
  - Use SPI DMA write register or data to RF

SPI interval

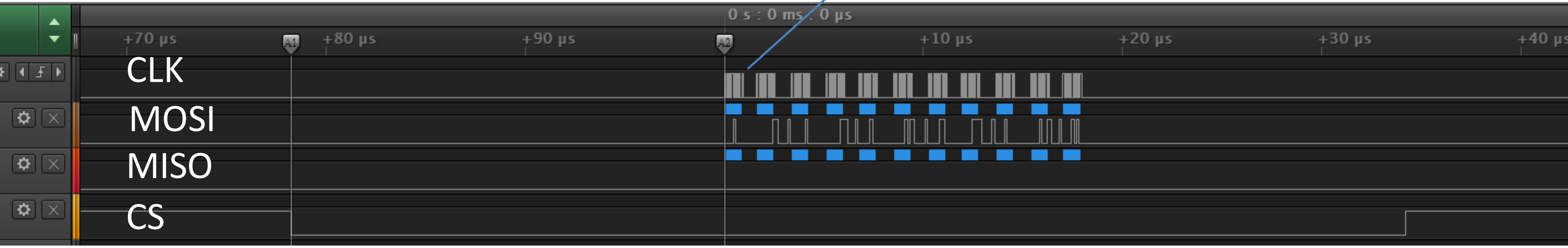
- Byte-to-byte interval as short as possible

Good SPI waveform



Bad!

byte-to-byte interval shorter is better



too long

too long

Rafael Micro

Proprietary and Confidential

(4-4) porting_misc.c

- MCU_GpioResetInit()
  - Initial one GPIO as Reset pin
  - Output mode, initial = HIGH
- MCU_GpioReset()
  - Reset RF: high->low(1ms)->high
- MCU_GpioPinInit()
  - Initial one GPIO as EXT_INT pin
  - Input mode, initial disable INT
- MCU_GpioIntEnable()
  - EXT_INT interrupt enable
- MCU_GpioIntDisable()
  - EXT_INT interrupt disable
- GPxx_IRQHandler()
  - GPIO external interrupt service routine
  - Clear GPIO interrupt status & Run LL_GPIO_Isr()

(4-4) porting_misc.c (cont.)

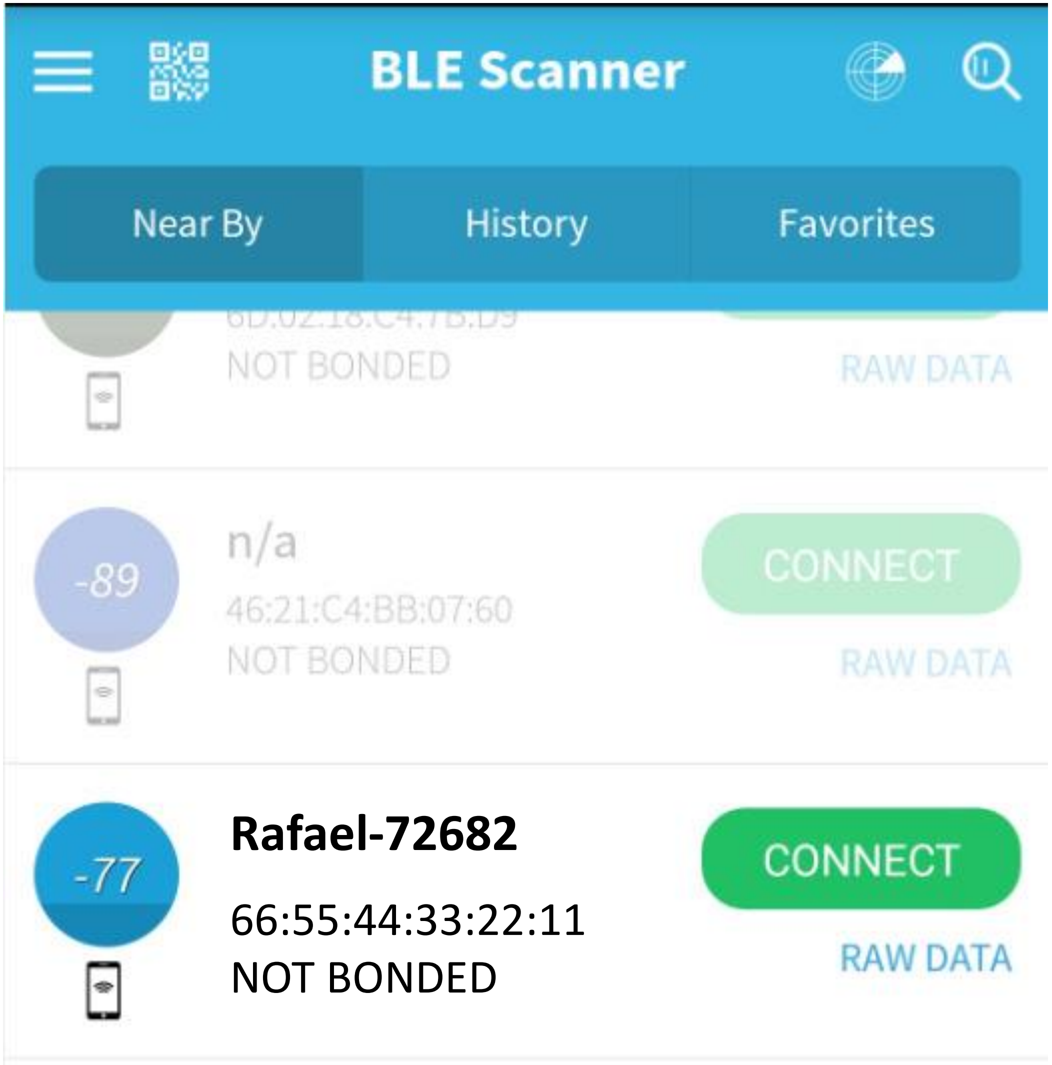
- spiGpioDelay()
  - Delay 1us
- spiGpioWriteReg(const unsigned char regAddr, const unsigned char regData)
  - Use 4 GPIO pins to emulate SPI master write operation
  - Write one byte to RF register
- SPI_GPIO_Init()
  - Set SPI_CLK, SPI_CS, SPI_MOSI, SPI_MISO, INT, as GPIO output pins
- RF_SpiIoMapping()
  - Implement IO mapping flow

(5) Check Point of each Stage

- (a) In main(), after IO mapping: read Register R0, it should get 0x66 or 0x67
- (b) Check SPI DMA write and read results
  - SPI DMA write 30 bytes to register R8~R37

```
28, 128, 223, 164, 224, 243, 127,   5,  33,  16,     //R8~R17
240, 160,  63, 187, 239, 194, 143, 122,   5, 128,     //R18~R27
255, 255, 255,  39,  98,   0, 112, 210, 136, 172,     //R28~R37
```

  - SPI DMA read 2/10/20/30 bytes from register R8~ Check the results.
- (c) Normal run. Open cell phone APP "BLE_Scanner", it should scan one BLE device, which name is "Rafael_72682"



6D:02:18:C4:7B:D9
NOT BONDED

n/a
46:21:C4:BB:07:60
NOT BONDED

Rafael-72682
66:55:44:33:22:11
NOT BONDED