Forum: 32-bit SAM(ARM core)
Topic: ATSAMD20J18 I2C slave問題
Subject: Re: ATSAMD20J18 I2C slave問題
作者: alan_hung
2018年10月01日 16:03:46

我發現到用start產生的程式碼有個地方錯誤，如果有microchip人員看到此篇，請幫忙回報

hpl_sercom.c
static void _sercom_i2c_s_irq_handler(struct _i2c_s_async_device *device)使用了hri_sercomi2cm_read_INTFLAG_reg(hw);這邊去讀master register。正確的因該要用hri_sercomi2cs_read_INTFLAG_reg(hw);

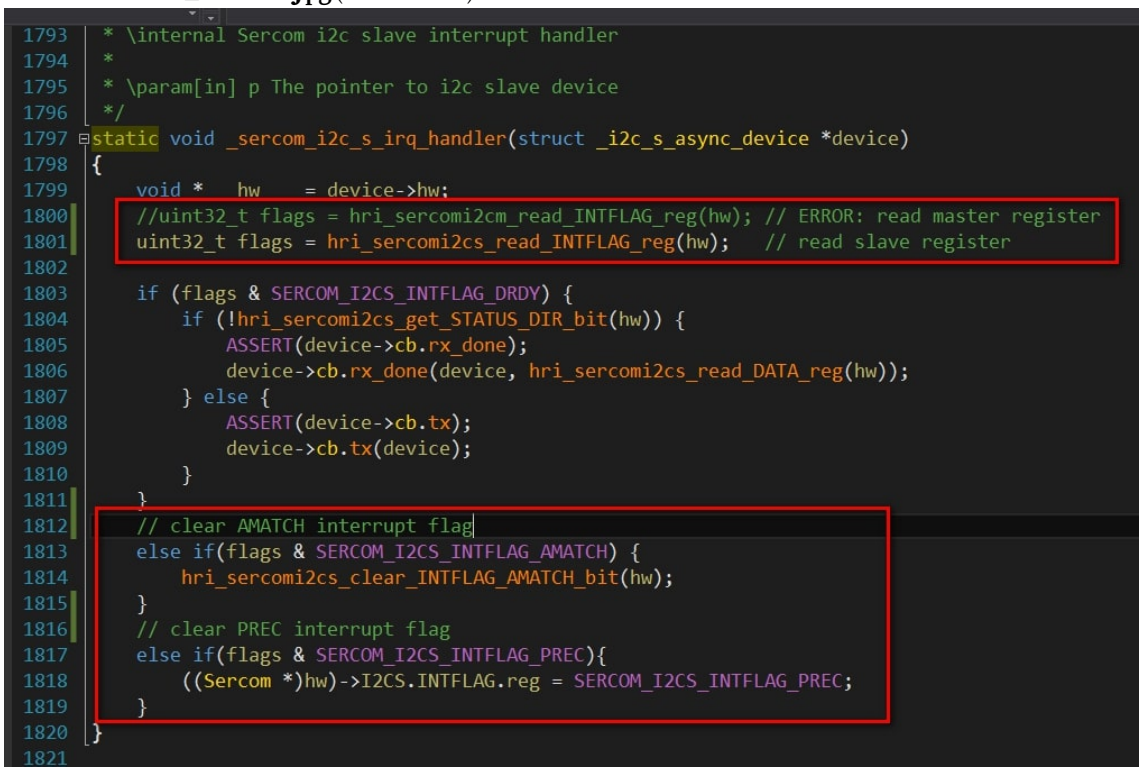
我修改過並增加部分程式，已經可以成卉all back讀資料
修改過的程式碼(紅色框起部分)貼在下方圖片，供大家參考

附加檔案:

2018-10-01_154408.jpg(75.57 KB)

```
15 static void I2C_0_rx_complete(const struct i2c_s_async_descriptor *const descr)
16 {
17     uint8_t c;
18
19     io_read(io, &c, 1);
20     // output to UART
21     printf("%c", c);
22 }
23
24 void I2C_0_example(void)
25 {
26     i2c_s_async_get_io_descriptor(&I2C_0, &io);
27     i2c_s_async_register_callback(&I2C_0, I2C_S_RX_COMPLETE, I2C_0_rx_complete);
28     i2c_s_async_enable(&I2C_0);
29     // enable AMATCH interrupt
30     ((Sercom *)I2C_0.device.hw)->I2CS.INTENSET.reg |= SERCOM_I2CS_INTENSET_AMATCH;
31 }
32
33 /**
34  * Example of using TARGET_IO to write "Hello World" using the IO abstraction.
35  */
36 void TARGET_IO_example(void)
37 {
38     struct io_descriptor *io;
39     usart_sync_get_io_descriptor(&TARGET_IO, &io);
40     usart_sync_enable(&TARGET_IO);
```

2018-10-01_154516.jpg(105.09 KB)



```
1793  * \internal Sercom i2c slave interrupt handler
1794  *
1795  * \param[in] p The pointer to i2c slave device
1796  */
1797 static void _sercom_i2c_s_irq_handler(struct _i2c_s_async_device *device)
1798 {
1799     void *   hw    = device->hw;
1800     //uint32_t flags = hri_sercomi2cm_read_INTFLAG_reg(hw); // ERROR: read master register
1801     uint32_t flags = hri_sercomi2cs_read_INTFLAG_reg(hw);   // read slave register
1802
1803     if (flags & SERCOM_I2CS_INTFLAG_DRDY) {
1804         if (!hri_sercomi2cs_get_STATUS_DIR_bit(hw)) {
1805             ASSERT(device->cb.rx_done);
1806             device->cb.rx_done(device, hri_sercomi2cs_read_DATA_reg(hw));
1807         } else {
1808             ASSERT(device->cb.tx);
1809             device->cb.tx(device);
1810         }
1811     }
1812     // clear AMATCH interrupt flag
1813     else if(flags & SERCOM_I2CS_INTFLAG_AMATCH) {
1814         hri_sercomi2cs_clear_INTFLAG_AMATCH_bit(hw);
1815     }
1816     // clear PREC interrupt flag
1817     else if(flags & SERCOM_I2CS_INTFLAG_PREC){
1818         ((Sercom *)hw)->I2CS.INTFLAG.reg = SERCOM_I2CS_INTFLAG_PREC;
1819     }
1820 }
1821
```