Power Manage in FSP

Hardware and Software Requirement

1. Renesas FSP V4.0.0 or later
2. Renesas RA Series Development Board
3. Nor Flash

Introduction

Usually, Nor Flash has more than one power mode, such as Normal mode and DeepPowerDown, but in FSP, it is not very easy to change Flash power mode, In this application, we introduce a middleware to manage Flash Power mode, and take MX25R as an example for interpretation. The Power Manage Middleware just offers some interfaces and the Flash vender should realize the interfaces according to related rules.

power_init ()

setNormalMode ()

setHighPerformanceMode ()

setLowPowerMode ()

setSuperLowPowerMode ()

setDeepPowerDownMode ()

Each vender should realize the interfaces above. power_init() is used to do some initialization work, and then vender should select some power mode according to actual Flash. Finally, the other power should return NOT_SUPPORT.

Sample Code

```
void hal_entry (void)
{
    mx25r_ID id             = {0xff,0xff,0xff};
    srand(RANDOM_SEED);

    SEGGER_RTT_printf(0, "Demo Start\r\n");
    g_qspi0.p_api->open(g_qspi0.p_ctrl, g_qspi0.p_cfg);
    g_mx25r0.p_api->rdid(&id);
    SEGGER_RTT_printf(0, "ID = %x%x%x\r\n", id.manufacturer_id,
id.memory_capacity, id.memory_type);
    g_qspi0.p_api->close(g_qspi0.p_ctrl);
    g_powermanage0.power_init();

    flash_lowpower_mode_test();
    flash_highperformance_mode_test();
    flash_deep_power_down_test();

    SEGGER_RTT_printf(0, "Demo End\r\n");
    while (1)
```

```
    {
    }
}
```

```
static fsp_err_t get_flash_status()
{
    spi_flash_status_t status = {.write_in_progress = true};
    int32_t time_out          = (INT32_MAX);
    fsp_err_t err             = FSP_SUCCESS;

    do
    {
        /* Get status from QSPI flash device */
        err = R_QSPI_StatusGet(&g_qspi0_ctrl, &status);
        if (FSP_SUCCESS!= err)
        {
            SEGGER_RTT_printf(0, "R_QSPI_StatusGet Failed\r\n");
            return err;
        }

        /* Decrement time out to avoid infinite loop in case of consistent
failure */
        --time_out;

        if ( 0 >= time_out)
        {
            SEGGER_RTT_printf(0, "\r\n ** Timeout : No result from QSPI flash
status register ** \r\n");
            return FSP_ERR_TIMEOUT;
        }
    }while (false != status.write_in_progress);
    return err;
}
```

```
static void read_flash_register_data()
{
    mx25r_status_reg sr     = {{0xff}};
    mx25r_cfg_reg cr        = {{0xff},{0xff}};
    mx25R_security_reg scur = {{0xff}};

    g_mx25r0.p_api->rdsr(&sr);
    SEGGER_RTT_printf(0, "Status Register = %x\r\n", sr.status);

    g_mx25r0.p_api->rdscur(&scur);
    SEGGER_RTT_printf(0, "Security Register = %x\r\n", scur.security);

    g_mx25r0.p_api->rdcr(&cr);
    SEGGER_RTT_printf(0, "Configration Register1 = %x\r\n", cr.configuration1);
    SEGGER_RTT_printf(0, "Configration Register2 = %x\r\n", cr.configuration2);
}
```

```
static void flash_lowpower_mode_test()
{
    SEGGER_RTT_printf(0, "***************LowPower
Test*******************\r\n");
```

```
    g_qspi0.p_api->open(g_qspi0.p_ctrl, g_qspi0.p_cfg);
    ///Exchange Frequence///
    R_QSPI->SFMCMD = 1U;
    R_QSPI->SFMSKC = 48;
    R_QSPI->SFMCMD = 0U;

    /* Fill write buffer with test data. Clear read buffers */
    for (int i = 0; i < BUFFER_LENGTH; i++)
    {
      writeBuffer[i] = (uint8_t) ( rand() % 256 ) ;
      readBuffer[i]  = (uint8_t) 0x00;
    }

    g_powermanage0.setLowPowerMode();
    read_flash_register_data();
    g_qspi0.p_api->erase(g_qspi0.p_ctrl, (uint8_t *)QSPI_DEVICE_START_ADDRESS,
4096);
    get_flash_status();

    g_qspi0.p_api->write(g_qspi0.p_ctrl, writeBuffer, (uint8_t
*)QSPI_DEVICE_START_ADDRESS, BUFFER_LENGTH);
    get_flash_status();

    memcpy(readBuffer, (uint8_t *)QSPI_DEVICE_START_ADDRESS, BUFFER_LENGTH);
    for(int i = 0; i<BUFFER_LENGTH; i++)
    {
        SEGGER_RTT_printf(0, "\t\tAddress: %02X,    ReadData:%02X,
ExpectedData:%02X\n", i,readBuffer[i],writeBuffer[i]);
        if( writeBuffer[i] == readBuffer[i] )
        {
            success = true;
        }
        else
        {
            success = false;
            break;
        }
    }
    if(success)
    {
        SEGGER_RTT_printf(0, (const char *)"\t\tData match\n");
    }
    else
    {
        SEGGER_RTT_printf(0, (const char *)"\t\tData error\n");
    }
    g_qspi0.p_api->close(g_qspi0.p_ctrl);
}
```

```
static void flash_highperformance_mode_test()
{
    SEGGER_RTT_printf(0, "***************High Performance
Test*******************\r\n");
    g_qspi0.p_api->open(g_qspi0.p_ctrl, g_qspi0.p_cfg);

    /* Fill write buffer with test data. Clear read buffers */
    for (int i = 0; i < BUFFER_LENGTH; i++)
```

```
    {
      writeBuffer[i] = (uint8_t) ( rand() % 256 ) ;
      readBuffer[i]  = (uint8_t) 0x00;
    }

    g_powermanage0.setHighPerformanceMode();
    read_flash_register_data();
    g_qspi0.p_api->erase(g_qspi0.p_ctrl, (uint8_t *)QSPI_DEVICE_START_ADDRESS,
4096);
    get_flash_status();

    g_qspi0.p_api->write(g_qspi0.p_ctrl, writeBuffer, (uint8_t
*)QSPI_DEVICE_START_ADDRESS, BUFFER_LENGTH);
    get_flash_status();

    memcpy(readBuffer, (uint8_t *)QSPI_DEVICE_START_ADDRESS, BUFFER_LENGTH);
    for(int i = 0; i<BUFFER_LENGTH; i++)
    {
        SEGGER_RTT_printf(0, "\t\tAddress: %02X,    ReadData:%02X,
ExpectedData:%02X\n", i,readBuffer[i],writeBuffer[i]);
        if( writeBuffer[i] == readBuffer[i] )
        {
            success = true;
        }
        else
        {
            success = false;
            break;
        }
    }
    if(success)
    {
        SEGGER_RTT_printf(0, (const char *)"\t\tData match\n");
    }
    else
    {
        SEGGER_RTT_printf(0, (const char *)"\t\tData error\n");
    }
    g_qspi0.p_api->close(g_qspi0.p_ctrl);
}
```

```
static void flash_deep_power_down_test()
{
    SEGGER_RTT_printf(0, "***************DeepPowerDown
Test*******************\r\n");
    g_qspi0.p_api->open(g_qspi0.p_ctrl, g_qspi0.p_cfg);
    /* Fill write buffer with test data. Clear read buffers */
    for (int i = 0; i < BUFFER_LENGTH; i++)
    {
      writeBuffer[i] = (uint8_t) ( rand() % 256 ) ;
      readBuffer[i]  = (uint8_t) 0x00;
    }

    g_qspi0.p_api->erase(g_qspi0.p_ctrl, (uint8_t *)QSPI_DEVICE_START_ADDRESS,
4096);
    get_flash_status();
```

```
    g_qspi0.p_api->write(g_qspi0.p_ctrl, writeBuffer, (uint8_t
*)QSPI_DEVICE_START_ADDRESS, BUFFER_LENGTH);
    get_flash_status();

    if(!(memcmp(writeBuffer, (uint8_t *)QSPI_DEVICE_START_ADDRESS,
BUFFER_LENGTH)))
    {
        SEGGER_RTT_printf(0, "Data has been writed to Flash\r\n");
    }
    else
    {
        SEGGER_RTT_printf(0, "Write data error\r\n");
    }

    SEGGER_RTT_printf(0, "Change Macronix MX25xxx power mode to
ENTER_DEEP_POWER_DOWN\r\n");
    g_powermanage0.setDeepPowerDownMode();

    memcpy(readBuffer, (uint8_t *)QSPI_DEVICE_START_ADDRESS, BUFFER_LENGTH);
    for(int i = 0; i<BUFFER_LENGTH; i++)
    {
      SEGGER_RTT_printf(0, "\t\tAddress: %02X,    ReadData:%02X,
ExpectedData:%02X\n", i,readBuffer[i],writeBuffer[i]);
      if( 0xff == readBuffer[i] )
      {
          success = true;
      }
      else
      {
          success = false;
          break;
      }
    }
    if(success)
    {
        SEGGER_RTT_printf(0, (const char *)"\t\tData match\n");
    }
    else
    {
        SEGGER_RTT_printf(0, (const char *)"\t\tData error\n");
    }

    SEGGER_RTT_printf(0, "Change Macronix MX25xxx power mode to
RELEASE_DEEP_POWER_DOWN\r\n");
    g_powermanage0.setHighPerformanceMode();

    memcpy(readBuffer, (uint8_t *)QSPI_DEVICE_START_ADDRESS, BUFFER_LENGTH);
    for(int i = 0; i<BUFFER_LENGTH; i++)
    {
        SEGGER_RTT_printf(0, "\t\tAddress: %02X,    ReadData:%02X,
ExpectedData:%02X\n", i,readBuffer[i],writeBuffer[i]);
        if( writeBuffer[i] == readBuffer[i] )
        {
         success = true;
        }
        else
        {
         success = false;
```

```
            break;
           }
    }
    if(success)
    {
        SEGGER_RTT_printf(0, (const char *)"\t\tData match\n");
    }
    else
    {
        SEGGER_RTT_printf(0, (const char *)"\t\tData error\n");
    }
    g_qspi0.p_api->close(g_qspi0.p_ctrl);
}
```