```c
/* spi_polling_25AA02UID_EEPROM_Rev2.c
 *
 *
 * Copyright (c) 2016, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */


#include "board.h"
#include "fsl_debug_console.h"
#include "mx25r_flash.h"
#include "fsl_spi.h"
#include "pin_mux.h"
#include <stdbool.h>

/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define EXAMPLE_SPI_MASTER SPI3
#define EXAMPLE_SPI_MASTER_IRQ FLEXCOMM3_IRQn
#define EXAMPLE_SPI_MASTER_CLK_SRC kCLOCK_Flexcomm3
#define EXAMPLE_SPI_MASTER_CLK_FREQ CLOCK_GetFreq(kCLOCK_Flexcomm3)
#define EXAMPLE_SPI_SSEL 2
#define EXAMPLE_SPI_SPOL kSPI_SpolActiveAllLow

/*******************************************************************************
 * Prototypes
 ******************************************************************************/

/*******************************************************************************
 * Variables
 ******************************************************************************/

uint32_t destBuff_Size = 4;
uint8_t destBuff[];

struct mx25r_instance mx25r;

/*******************************************************************************
 * Code
 ******************************************************************************/

int flash_transfer_cb(void *transfer_prv, uint8_t *tx_data, uint8_t *rx_data, size_t
dataSize, bool eof)
{

    spi_transfer_t xfer = {0};
    xfer.txData = tx_data;
    xfer.rxData = rx_data;
    xfer.dataSize = dataSize;
    /* terminate frame */
    if (eof)
    {
      xfer.configFlags |= kSPI_FrameAssert;
    }

    SPI_MasterTransferBlocking((SPI_Type *)transfer_prv, &xfer);

    PRINTF("MADE IT TO FLASH_TRANSFER: The receive data is: \r \n");
    return 0;
}

//==================================================================================

int main(void)
{
    spi_master_config_t userConfig = {0};
    uint32_t srcFreq = 0;
    uint32_t i = 0;


    /* attach 12 MHz clock to FLEXCOMM0 (debug console) */
    CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);

    /* attach 12 MHz clock to SPI3 */
    CLOCK_AttachClk(kFRO12M_to_FLEXCOMM3);

    /* reset FLEXCOMM for SPI */
    RESET_PeripheralReset(kFC3_RST_SHIFT_RSTn);

    BOARD_InitPins();
    BOARD_BootClockFROHF48M();
    BOARD_InitDebugConsole();
    PRINTF("\n\rMaster Start...\n\r");
    /*
     * userConfig.enableLoopback = false;
     * userConfig.enableMaster = true;
     * userConfig.polarity = kSPI_ClockPolarityActiveHigh;
     * userConfig.phase = kSPI_ClockPhaseFirstEdge;
     * userConfig.direction = kSPI_MsbFirst;
     * userConfig.baudRate_Bps = 500000U;
     */
    SPI_MasterGetDefaultConfig(&userConfig);
    srcFreq = EXAMPLE_SPI_MASTER_CLK_FREQ;
    userConfig.sselNum = (spi_ssel_t)EXAMPLE_SPI_SSEL;
    userConfig.sselPol = (spi_spol_t)EXAMPLE_SPI_SPOL;
    userConfig.baudRate_Bps = 100000U;
    SPI_MasterInit(EXAMPLE_SPI_MASTER, &userConfig, srcFreq); //original

    mx25r_init(&mx25r, flash_transfer_cb, EXAMPLE_SPI_MASTER);

//====== read n bytes starting at 'address'

 //   mx25r_cmd_read(struct mx25r_instance *instance, uint32_t address, uint8_t
*buffer, uint32_t size)
//      mx25r_cmd_read(&mx25r, 0xFC, destBuff, size); // Works, but still returns '0'
    mx25r_cmd_read(&mx25r, 0xFC, destBuff, destBuff_Size);
```

```
        /*Check if the data is right*/
          for (i = 0; i < destBuff_Size; i++)
          {
            PRINTF("The destBuff[%d] =  %d\n\r", i, destBuff[i]);
          }


    while (1)
    {
    }
}
```

```c
/*    mx25r_flash.c
 * Copyright (c) 2016, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include "mx25r_flash.h"

// #define MX25R_BYTE_ADDR(address) (address & 0xf)

/* initialize 'mx25r_instance' */
mx25r_err_t mx25r_init(struct mx25r_instance *instance, transfer_cb_t callback, void
*callback_prv)
{
    instance->callback = callback;
    instance->prv = callback_prv;
    return mx25r_err_ok;
}

/* read n bytes starting at 'address' */
mx25r_err_t mx25r_cmd_read(struct mx25r_instance *instance, uint32_t address, uint8_t
*buffer, uint32_t size)
{

//    if (address & 0xFF000000U)
//    {
//        return mx25r_err_out_of_range;
//    }
    instance->cmd[0] = 0x03;
 //   instance->cmd[1] = MX25R_BYTE_ADDR(address);
    instance->cmd[1] = 0xFC;
    instance->callback(instance->prv, instance->cmd, NULL, 2, false);
    instance->callback(instance->prv, NULL, (uint8_t *)buffer, size, true);

    return mx25r_err_ok;
}
```

```c
/*   mx25r_flash.h
 * Copyright (c) 2016, Freescale Semiconductor, Inc.
 * Copyright 2016-2017 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#ifndef _MX25R_FLASH_H_
#define _MX25R_FLASH_H_

#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>

typedef enum mx25r_err
{
    mx25r_err_ok = 0,
    mx25r_err_out_of_range,
    mx25r_err_alignement,
} mx25r_err_t;

typedef int (*transfer_cb_t)(void *transfer_prv, uint8_t *tx_data, uint8_t *rx_data,
size_t dataSize, bool eof);

struct mx25r_instance
{
    void *prv;
    transfer_cb_t callback;
    uint8_t cmd[5];
};

#if defined(__GNUC__)
#else
__packed
#endif
struct mx25r_rdid_result
{
    char manufacturer;
    char device[2];
#if defined(__GNUC__)
} __attribute__((packed));
#else
};
#endif

#if defined(__GNUC__)
#else
__packed
#endif
struct mx25r_rdsr_result
{
    char sr0;
    char sr1;
#if defined(__GNUC__)
```

```
} __attribute__((packed));
#else
};
#endif

#if defined(__GNUC__)
#else
__packed
#endif
struct mx25r_read_result
{
    uint32_t word[4];
#if defined(__GNUC__)
} __attribute__((packed));
#else
};
#endif

union mx25r_result
{
    struct mx25r_rdid_result rdid;
    struct mx25r_rdsr_result rdsr;
    struct mx25r_read_result read;
};

mx25r_err_t mx25r_init(struct mx25r_instance *instance, transfer_cb_t callback, void
*callback_prv);
mx25r_err_t mx25r_cmd_rdid(struct mx25r_instance *instance, struct mx25r_rdid_result
*result);
mx25r_err_t mx25r_cmd_read(struct mx25r_instance *instance, uint32_t address, uint8_t
*buffer, uint32_t size);
mx25r_err_t mx25r_cmd_nop(struct mx25r_instance *instance);
mx25r_err_t mx25r_cmd_rdsr(struct mx25r_instance *instance, struct mx25r_rdsr_result
*result);
mx25r_err_t mx25r_cmd_wrdi(struct mx25r_instance *instance);
mx25r_err_t mx25r_cmd_wren(struct mx25r_instance *instance);
mx25r_err_t mx25r_cmd_write(struct mx25r_instance *instance,
                            uint32_t address_256_align,
                            uint8_t *buffer,
                            uint32_t size_256_max);
mx25r_err_t mx25r_cmd_sector_erase(struct mx25r_instance *instance, uint32_t
address);

#endif
```