Application Report
SLOA214–February 2015

Using Special Direct Mode With the TRF7970A

John Crutchfield — *S2 MCU NFC/RFID Applications Engineer*

ABSTRACT

This document is intended as an introduction and reference for using Special Direct Mode and Direct Mode 1 with the TRF79xxA devices. Previously, Direct Mode 0 was required when using the TRF79xxA with popular non-ISO standard RFID protocols, such as MIFARE® Classic. Special Direct Mode and Direct Mode 1 reduce the overall system complexity and cost in these types of application use cases. In summary, these modes can be used to simplify the implementation of proprietary RFID protocols with low cost microcontrollers which might have intentional resource limitations, such as the MSP430G2xx series devices.

Source code discussed in this application report can be downloaded from http://www.ti.com/lit/zip/sloa214.

Contents

List of Figures

List of Tables

1 Theory of Operation

There are possible use cases where supporting a proprietary protocol may be required. The most common use case is supporting MIFARE Classic tags. These tags follow ISO14443A anticollision and selection procedures, but then divert into an non ISO standard based protocol. MIFARE Classic tags, although not fully ISO standard compliant, are still very popular and have long been used in common applications such as access control and micro-payment or pre-payment solutions.

The TRF79xxA NFC/RFID transceiver family has built in automated support for handling NFC/RFID compliant protocols. This is used for anticollision and selection procedure of MIFARE classic tags. The TRF79xxA also has direct mode support, allowing the user more flexibility and control to support custom protocols. The TRF7970A device is the superset device of the family, the TRF7970AEVM was used in this application note for hardware platform, and the Code Composer Studio™ (CCS) IDE was used for code development.

2 Overview

2.1 TRF7970A Direct Modes

The TRF7970A supports several different control modes, and each is discussed below. Each mode supports different functionality as well as its own type of control interface. The direct modes are as follows:

- Direct Mode 2 (Normal Operation)
- Special Direct Mode (SDM)
- Direct Mode 1 (DM1)
- Direct Mode 0 (DM0)

The different levels of control that each modes enables can be modeled, as seen in Figure 1. Each direct mode has access to a different layer of the TRF79xx. SDM and DM1 access the same layer.

Table 1. TRF7970A Pin Function

TRF7970A Pin	Function
IRQ	
DATA_CLK	
I/O_7	MOSI
I/O_6	MISO
I/O_5	SDM Bit Clock
I/O_4	Slave Select
I/O_3	SDM TX Data
I/O_2	SDM TX Enable
I/O_1	
I/O_0	
MOD	

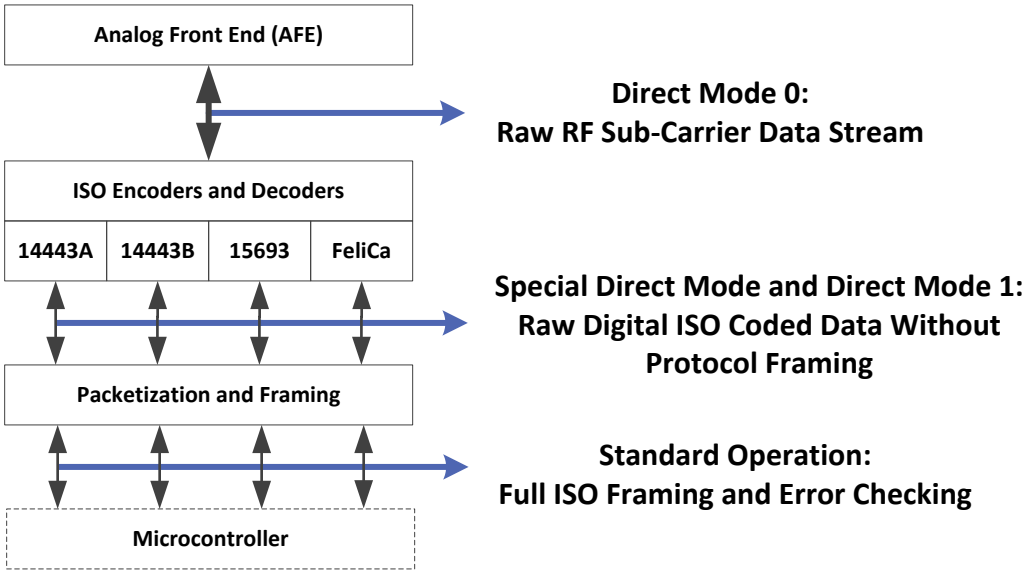

Figure 1. Direct Modes

2.1.1 Direct Mode 0 (DM0)

Direct mode 0 uses the TRF79xx as an AFE only, giving access to the raw RF data stream. In this mode, the host controller has access to the demodulated RF subcarrier data stream. This means the host is directly in control over the modulation for both transmitting and receiving. It can control the modulation for transmitting through the MOD pin. For receiving, the host must decode the modulation on I/O_6, the MISO line. This functionality allows DM0 to support practically any protocol, even proprietary modulation schemes.

Direct mode 0 allows for the most control, but is also the most demanding on the host resources. The host must generate and check any required CRC bytes or parity bits. Communication is asynchronous in DM0, so the host must also generate and keep the required timing for modulation and demodulation for each bit/byte.

Typically, these timing demands require the host must be running of a <1% tolerance clock, and either be running at a multiple of 13.56 MHz, or >40 MHz. The clock source requirements are usually accomplished by driving the host from the SYS_CLK output on the TRF79xx. These timing constraints usually eliminate many low end microcontrollers, that don't support HF clock sources, from this use case. In fact, the purpose of this application note is to exploit the power of using Special Direct Mode versus Direct Mode 0 for supporting these types of transponders as it allows the use low end, resource limited MCUs, thus bringing about an overall lower system cost solution from what was previously unavailable to the market.

Direct mode 0 implementation is covered in detail in the *Direct Mode* section of the TRF7970A data sheet (SLOS743).

2.1.2 Special Direct Mode and Direct Mode 1

While both SDM and DM1 access the TRF7970A at the same layer, SDM is used for transmitting while DM1 is used for receiving. Because both modes use the built in ISO encoders and decoders, these modes only support ISO15693, ISO14443A, ISO14443B, and FeliCa™ modulation schemes.

Both modes support synchronous communication for sending and receiving each bit. The synchronous communication relaxes the host timing demands, when compared to Direct Mode 0, allowing for even TI Value Line MCU hosts. Like DM0 though, the host still has direct responsibility for the raw data packaging. This requires the host to generate and check any CRC bytes or parity bits.

2.1.2.1 Direct Mode 1

Direct Mode 1, in this application example, is only used for receiving. The TRF79xxA handles modulation decoding internally, so as the TRF79xx decodes each bit, it has to pass them to the host. Each bit is written on I/O_6, the MISO pin. The TRF79xx also generates a data clock signal on I/O_5, Bit Block (SDM), for synchronous data communication. The TRF79xx's IRQ signal indicates the end of receive.

2.1.2.2 Special Direct Mode

SDM is only used for transmitting. As seen in Figure 1, SDM utilizes the TRF79xx's built in ISO encoders instead of generating the raw RF data, like in DM0, during transmit. The host MCU has to pass each bit to the TRF79xx at the correct time for it to be encoded properly.

To begin a transmission, the host MCU places the first data bit on I/O_3 (SDM TX Data) and brings I/O_2 (SDM TX Enable) high. The TRF79xxA then generate a clock signal on I/O_5 (SDM Bit Clock) to indicate when it needs the next bit. To end a transmission, the host MCU pulls I/O_2 low. These firmware steps are covered in greater detail in Section 4.

2.1.3 Direct Mode 2 / ISO Mode

ISO Mode is standard operation for the TRF79xxA and is the easiest to use mode. In this mode, the TRF79xxA family fully supports NFC/RFID compliant protocols. The host can take full advantage of all the TRF79xxA features, especially the built-in data FIFO. Also, all communication is over SPI.

This functionality is covered in detail in the TRF7970A data sheet (SLOS743).

2.2 MIFARE Classic

This application note is focused on the authentication of MIFARE Classic tags by using SDM and DM1. This section is a brief introduction to MIFARE Classic.

MIFARE Classic tags are a unique tag type that was developed originally by Mikron, acquired by Philips (now NXP), and also available in various configurations from Infineon, Hitachi Renesas, Gemalto, and Oberthur Technologies. Figure 2 shows a simple tag interaction flow.

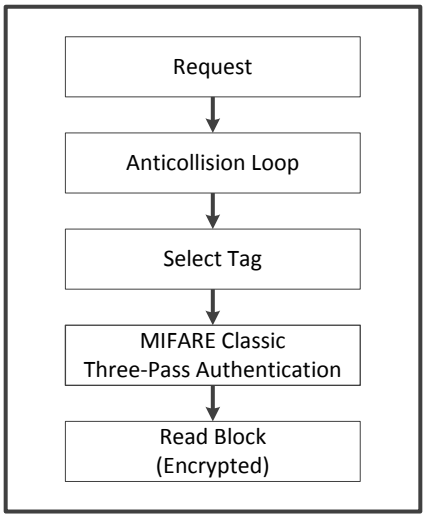

Figure 2. MIFARE Card Interaction Flow

MIFARE tags begin by following ISO14443A but branch into non-ISO standard commands immediately after being selected. This means normal operation can be used by the TRF79xx for the request, anticollision loop, and selection stages of the tag. After this stage, the TRF79xx normal operation can no longer be used because MIFARE encryption happens after standard packaging. Either DM0 or SDM/DM1 must be used to bypass the packetization stage in the TRF79xx, but SDM/DM1 is the recommended choice.

After selection, the reader must go through a three-pass authentication. This is where MIFARE classic deviates from the ISO standard, and the TRF79xx must be placed in SDM. All over the air communication from this point is encrypted in MIFARE classic's proprietary encryption.

3 Hardware Description

3.1 TRF7970A – NFC/RFID Transceiver IC

TRF7970A is a high performance 13.56-MHz HF RFID/NFC Transceiver IC composed of an integrated analog front end (AFE) and a built-in data framing engine for ISO15693, ISO14443A, ISO14443B, and FeliCa. This includes data rates up to 848 kbps for ISO14443 with all framing and synchronization tasks on board (in default mode). The TRF7970A also supports NFC Tag Type 2, 3, 4, and 5 operations. This architecture enables the customer to build a complete cost-effective yet high-performance multi-protocol 13.56-MHz RFID/NFC system together with a low-cost microcontroller (for example, an MSP430™ MCU).

3.2 MSP430F2370 – 16-Bit RISC Mixed-Signal Microcontroller

The Texas Instruments MSP430 family of ultra-low-power microcontrollers consists of several devices featuring different sets of peripherals targeted for various applications. The architecture, combined with five low-power modes, is optimized to achieve extended battery life in portable measurement applications. The device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency. The digitally controlled oscillator (DCO) allows wake-up from low-power modes to active mode in less than 1 µs.

The MSP430F23x0 series is an ultra-low-power microcontroller with two built-in 16-bit timers, one universal serial communication interface (USCI), a versatile analog comparator, and 32 I/O pins.

3.3 TRF7970A Evaluation Module

The TRF7970A EVM is a self-contained development platform which can be used to independently evaluate/test the performance of the TRF7970A RFID/Near Field Communications (NFC) Transceiver IC, custom firmware, customer-designed antennas, and potential transponders for a customer-defined RFID/NFC application.

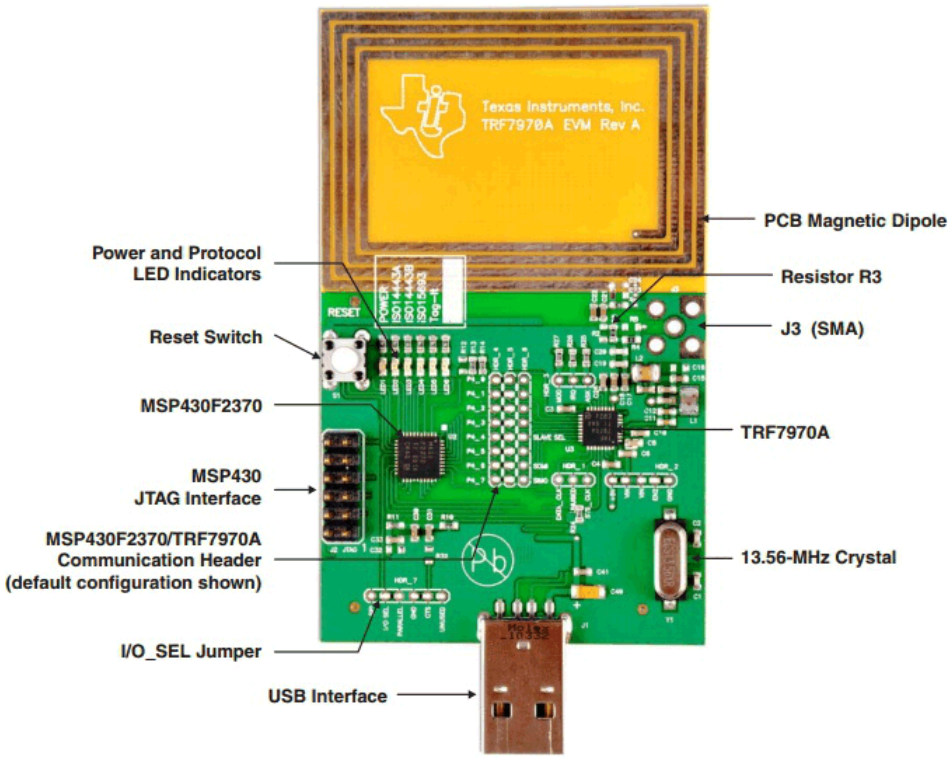

Figure 1. TRF7970A EVM (Top Side)

Figure 3. TRF7970A Evaluation Module

3.3.1 Hardware Setup for SDM

The TRF7970A EVM hardware must be configured to support SDM and DM1. There are 0-Ω resistors on the front of the EVM that must be moved to easily configure the hardware. Figure 4 shows the required positions for the 0-Ω resistors on the top of the EVM.

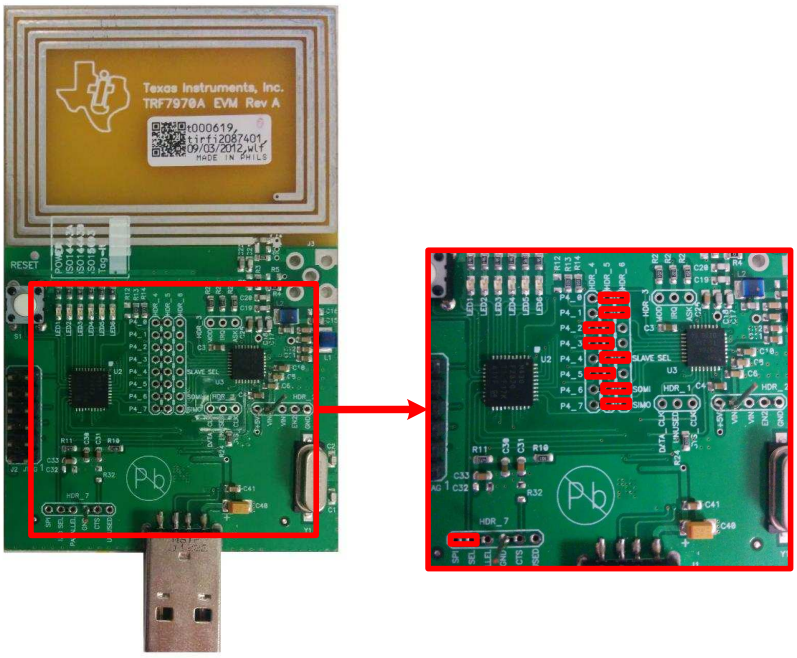

Figure 4. TRF7970A EVM Hardware Setup

Table 2 shows the functions of each TRF7970A pin in Special Direct Mode. I/O_2 is also used at startup to place the TRF7970A in SPI mode, which means it must be brought high before EN.

Table 2. SDM Connections

TRF79xx Pin	Function in SDM
I/O_0	I/O_0
I/O_1	I/O_1
I/O_2	SDM TX Enable
I/O_3	SDM TX Data
I/O_4	Slave Select
I/O_5	SDM Bit Clock
I/O_6	MISO
I/O_7	MOSI

4 Firmware Description

The example firmware (http://www.ti.com/lit/zip/sloa214) is discussed in three sections:

1. General Overview: a high level description of the functionality (Section 4.1)
2. Transmitting with Special Direct Mode (Section 4.2)
3. Receiving with Direct Mode 1 (Section 4.3)

These sections contains step-by-step instructions for implementing SDM and DM1. These steps are used, and commented in the example firmware.

4.1 General Overview

4.1.1 Code Description

The provided code example demonstrates the TRF7970A as a NFC/RFID reader. The TRF7970A EVM polls for ISO14443A tags every 100 ms. After a tag is selected, a three-pass MIFARE Classic authentication with the default keys is attempted. If authentication is successful, an encrypted read of block 0 is attempted.

SDM and DM1 are using in the Auth1, Auth2, Auth3, and Read Block states. The onboard LEDs indicate each successful layer according to Table 3.

Table 3. LED Key on TRF7970A EVM

LED	Indication
LED1	Power
LED2	ISO14443A Tag Found
LED3	Authentication 1 Success
LED4	Authentication 2 Success
LED5	Authentication 3 Success
LED6	Encrypted Read Block Success

4.1.2 Flow Diagram

Figure 5 shows the flow diagram of the code example.

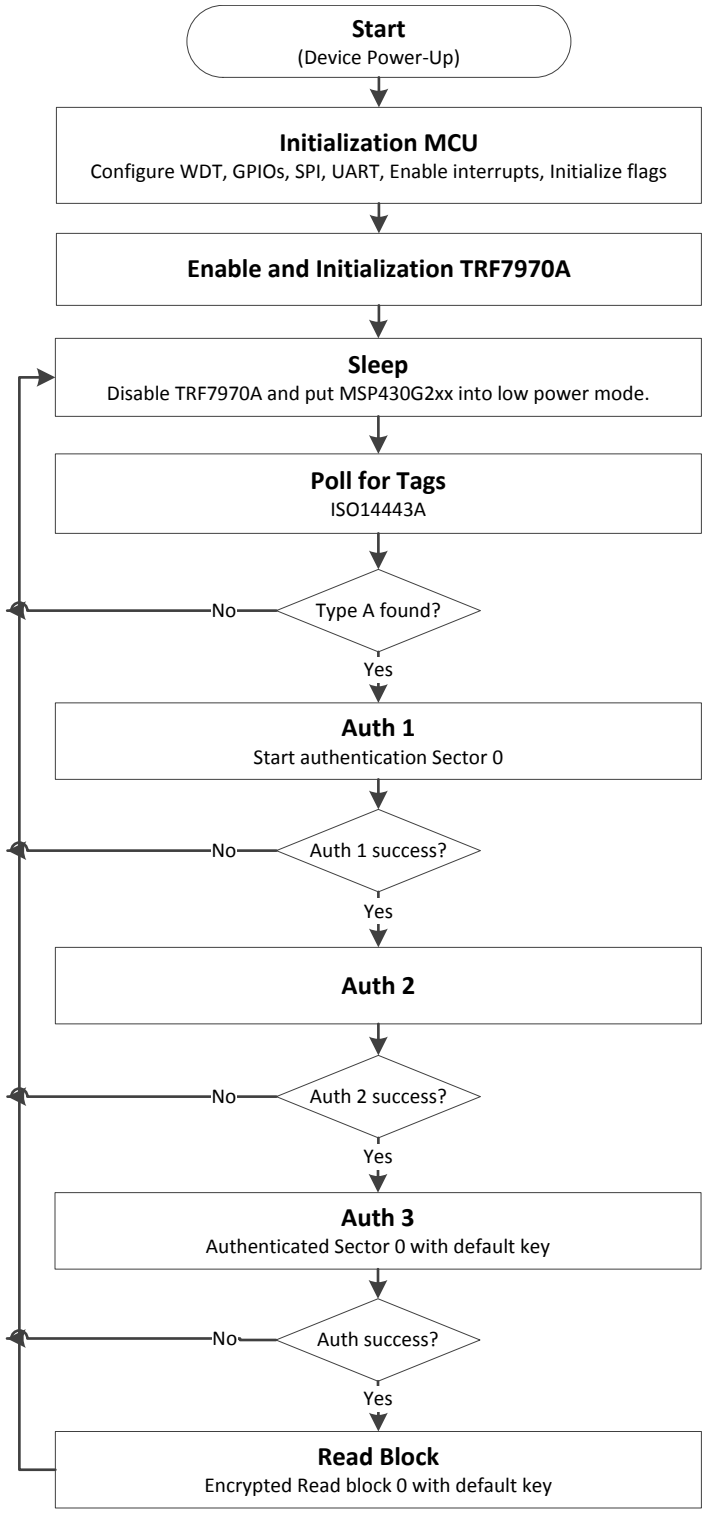

Figure 5. General Overview Flow Diagram

4.2 Transmitting Using Special Direct Mode

This section describes how to with SDM. All code that is shown is part of http://www.ti.com/lit/zip/sloa214. The following code snippet shows the provided SDM functions for transmitting.

```
    Mifare_SDM_Config();
    Mifare_SDM_Enter();       //Enter Special Direct Mode
    Mifare_SDM_Transmit(g_pui16TxData, g_ui8TxLength, g_bTxParity);
    Mifare_SDM_Exit();
```

The TRF7970A must be configured properly first. Next, SDM can be entered, data transmitted, and exited. There is a response expected after each transmission, so it is critical to quickly exit SDM and enter DM1.

4.2.1 SDM Configuration

These are the steps to configure the TRF79xx for SDM:

1. Set TX_Data (I/O_3) Low (required for sending correct SOF when entering SDM)
2. Write register 0x0D with 0x3E (Disable No Response IRQ)
3. Setup register 0x01 (Write with 0x88 for ISO14443A/Mifare Classic at 106 kbps)
4. Write register 0x00 with 0x21 (enable RF)
5. Send direct command 0x16 to disable decoders (send 0x96)
6. Clear IRQ by reading register 0x0C

```
void Mifare_SDM_Config(void){
    uint8_t dummy_read;

    SDM_TXENABLE_OFF;
    SDM_DATA_OFF;                                          // Step 1
    SDM_PORT_SET;

    TRF79x0_writeSingle(0x3E, TRF79X0_IRQ_MASK_REG);       // Step 2

    TRF79x0_writeSingle(0x88, TRF79X0_ISO_CONTROL_REG);    // Step 3

    TRF79x0_writeSingle(0x21, TRF79X0_CHIP_STATUS_CTRL_REG);   // Step 4

    TRF79x0_directCommand(TRF79X0_STOP_DECODERS_CMD);      // Step 5

    TRF79x0_readSingle(&dummy_read,TRF79X0_IRQ_STATUS_REG);    // Step 6
    g_ui8IrqFlag = 0;
}
```

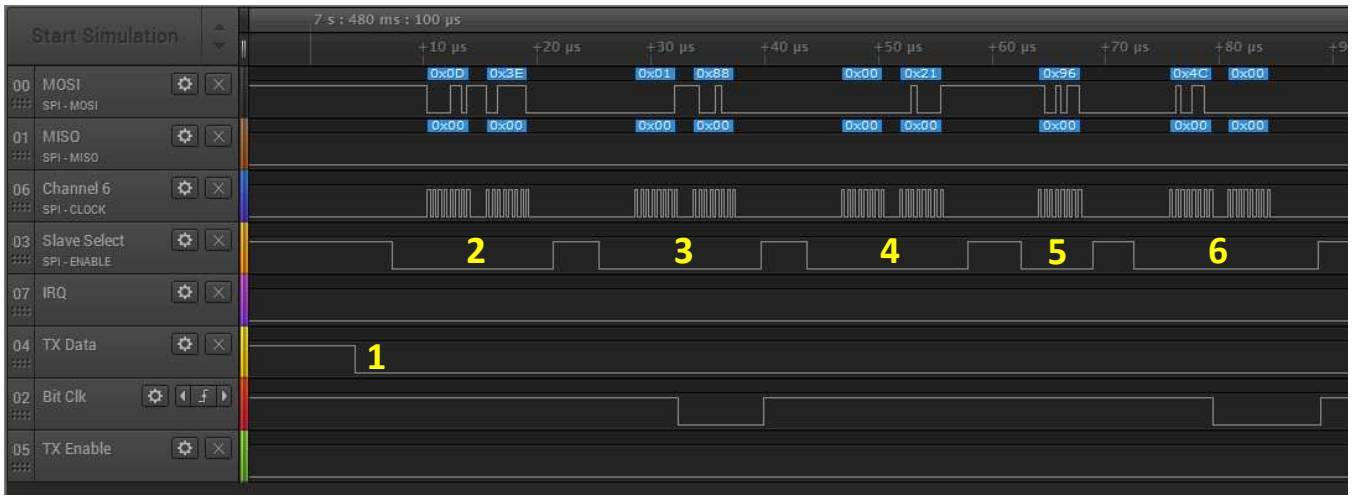

Figure 6. Special Direct Mode Configuration

4.2.2 SDM Enter

These are the steps to enter SDM:

7. Write register 0x10 with 0x08 (Enable Special Direct Mode)
8. Write register 0x00 with 0x61 with *no stop condition, SS pin stays low* (enter Special Direct Mode).
9. Send eight clock cycles to enter Special Direct Mode. *SS pin stays low*.

```
void Mifare_SDM_Enter(void)
{
    uint8_t dummy_write = 0;

    TRF79x0_writeSingle(0x08, TRF79X0_SPECIAL_FUNC_1_REG);     // Step 7

    g_ui8DirectMode = 1;  // Keeps Slave Select Low
    TRF79x0_writeSingle(0x61, TRF79X0_CHIP_STATUS_CTRL_REG);   // Step 8

    TRF79x0_rawWrite(&dummy_write, 1);                         // Step 9

    g_ui8TimeOutFlag = 0;
    MCU_timerInit(40, (uint8_t*) &g_ui8TimeOutFlag);
}
```

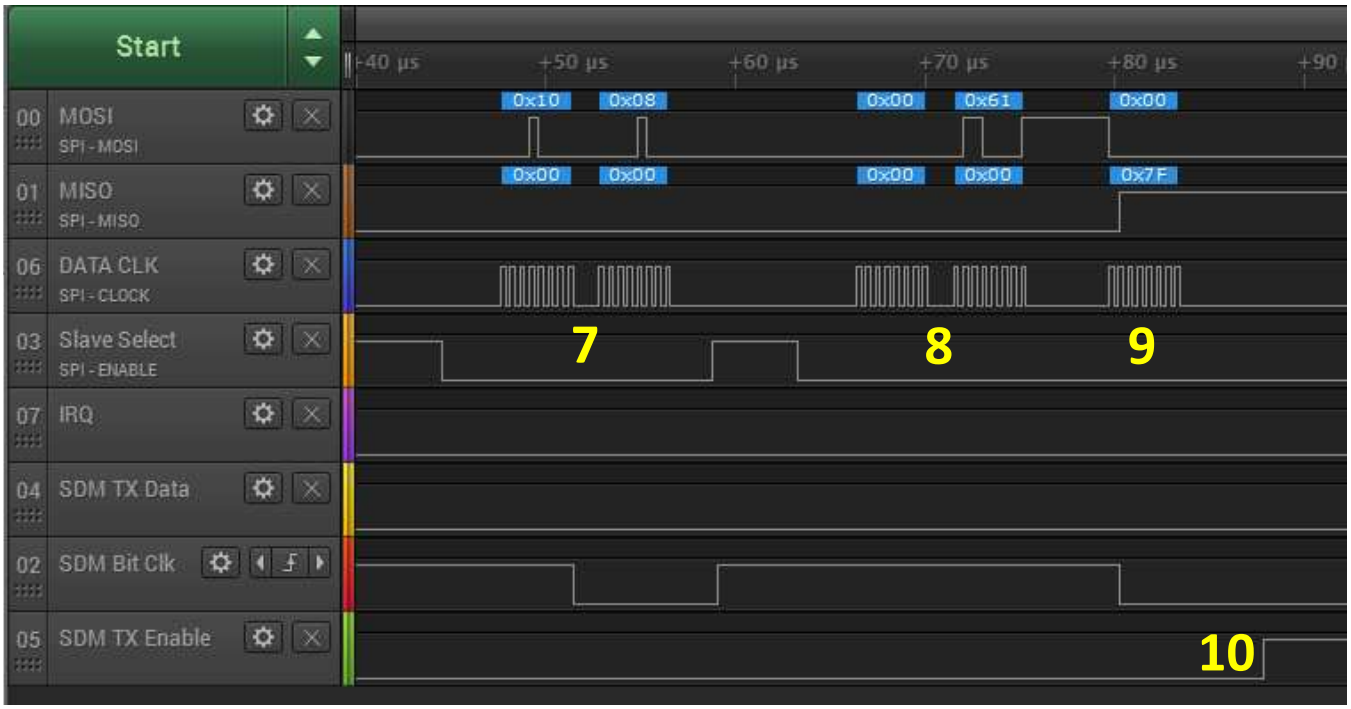

Figure 7. Entering Special Direct Mode

4.2.3 SDM Transmit

The following is an SDM transmit function implementation. This example was done running on a MSP430F2370 running at 8 MHz. An interrupt based implementation was tested, but this polling version was quicker.

Steps to transmit in SDM (see Figure 8 and Figure 9):

10. I/O_2 goes high and enables TX
11. Set TX Data (I/O_3) on bit clock (I/O_5) rising edge
    a. Data is latched on bit clock (I/O_5) falling edge
    b. Data is transmitted out LSB for MIFARE Classic communication
12. I/O_2 goes low and disables TX
    a. Must wait until after Bit Clk rising edge on last data bit

```
void Mifare_SDM_Transmit(uint16_t *pui16TxData, uint8_t ui8ByteCount, uint8_t bSDMTxParity){
    uint8_t i, j, ui8ByteWidth;

    SDM_TXENABLE_ON;                                                    // Step 10

    if(bSDMTxParity){  // Parity bits?
        ui8ByteWidth = 9;
    }
    else{
        ui8ByteWidth = 8;
    }

    for(i=0; i<ui8ByteCount; i++){        // Byte Count
        for(j=0; j<ui8ByteWidth; j++){    // Bit Count
            // wait for Falling edge
            while(SDM_BITCLK && (!g_ui8IrqFlag) && (!g_ui8TimeOutFlag));
```

```
            if(pui16TxData[i] & (0x01 << j)){  // Next bit a '0' or '1'?
                // wait for Rising edge
                while((!SDM_BITCLK) && (!g_ui8IrqFlag) && (!g_ui8TimeOutFlag))
                SDM_DATA_ON;    // '1'                                    // Step 11
            }
            else{
                // wait for Rising edge
                while((!SDM_BITCLK)  && (!g_ui8IrqFlag) && (!g_ui8TimeOutFlag));
                SDM_DATA_OFF;    // '0'                                   // Step 11
            }
        }
    }
    /************** EOF Sequence ************/
    if((!g_ui8IrqFlag) ){
        while(SDM_BITCLK && (!g_ui8TimeOutFlag));
        while(!SDM_BITCLK && (!g_ui8TimeOutFlag));

        SDM_TXENABLE_OFF;                                                 // Step 12
        while(SDM_BITCLK && (!g_ui8TimeOutFlag));
        while(!SDM_BITCLK && (!g_ui8TimeOutFlag));
    }
}
```

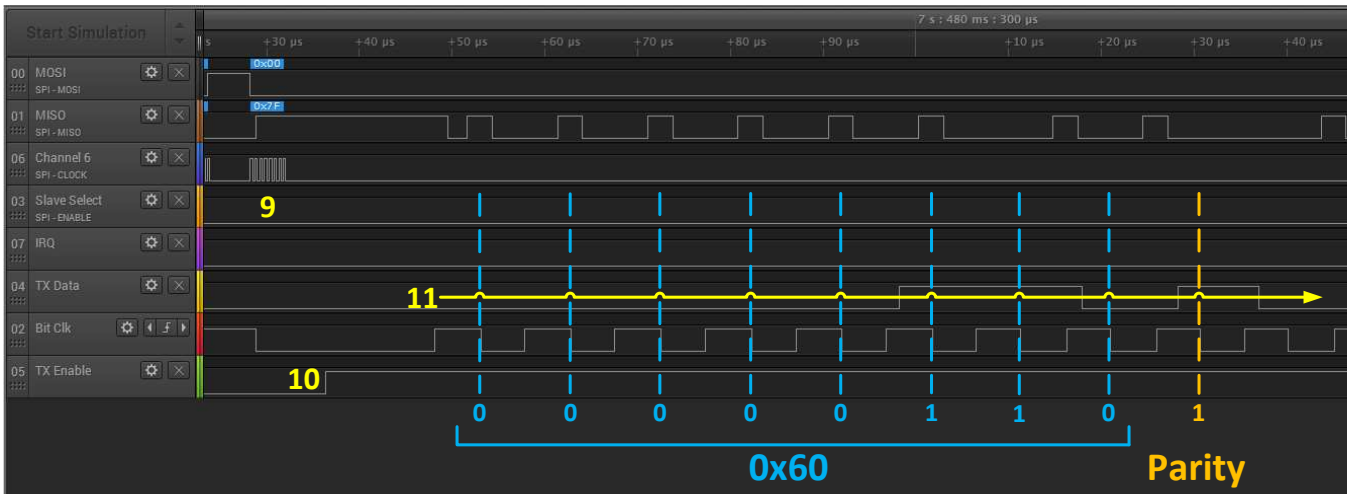

Figure 8. SDM Transmit

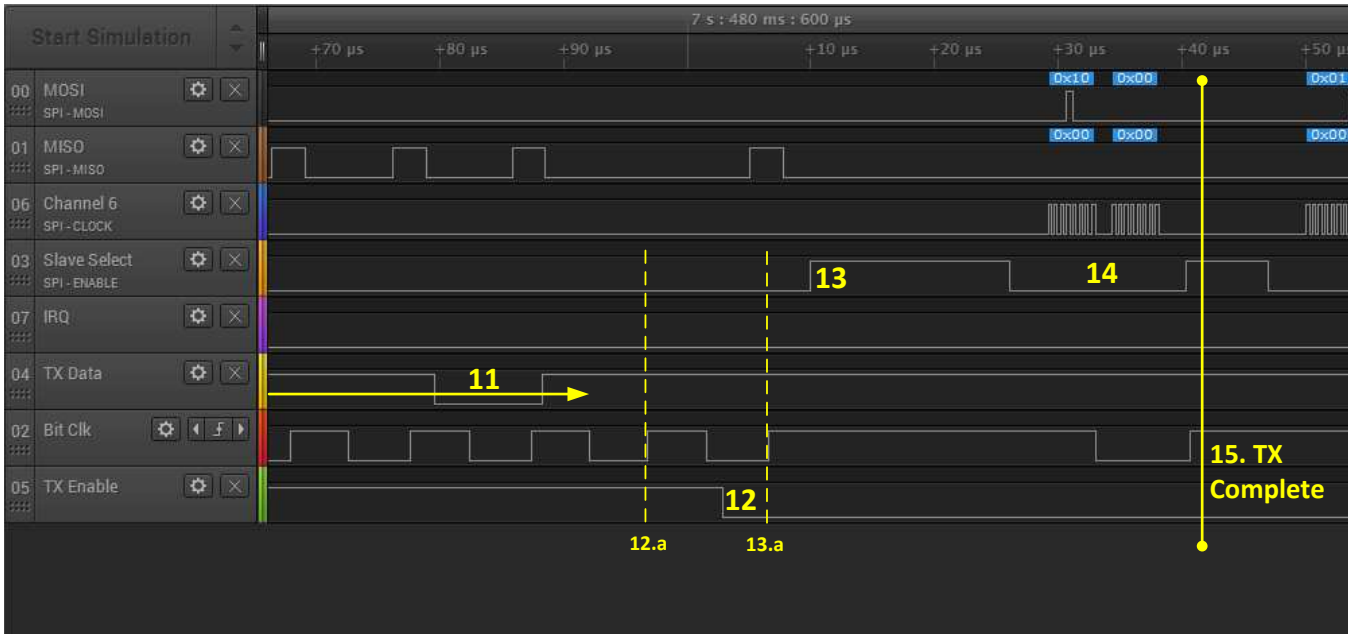

Figure 9. SDM Transmit Part 2 and SDM Exit

4.2.4 SDM Exit

Steps to exit SDM (see Figure 9):

13. Stop condition issued on SPI (SS line brought high)
    a. Must wait until after the next SDM Bit Clk rising edge.
14. Write Register 0x10 with 0x00 (disabling Special Direct Mode)
15. TX Complete

```
void Mifare_SDM_Exit(void)
{
    SS_HIGH;                        // Step 13
    g_ui8DirectMode = 0;

    buf[0] = TRF797x_SPECIAL_FUNCTION;
    buf[1] = 0x00;
    SpiWriteSingle(buf, 2);         // Step 14 and Step 15
}
```

4.3 Receiving With Direct Mode 1 (DM1)

This section describes receiving with DM1. DM1 must be entered quickly after transmissions to make sure that all response data is received.

```
    Mifare_DM1_Enter();      //Enter Direct Mode 1
    Mifare_DM1_Receive(g_pui16RxData, &g_ui8RxLengthBytes, &g_ui8RxLengthBits, g_bRxParity);
    Mifare_DM1_Exit();
```

4.3.1 DM1 Enter

Steps to enter Direct Mode 1:

16. Write register 0x01 with 0x48 (enable Direct Mode 1).
17. Write register 0x00 with 0x61 with *no stop condition, SS pin stays low* (to enter Direct Mode 1).
18. Send eight clock cycles to enter Direct Mode 1.

```
void Mifare_DM1_Enter(void)
{
    uint8_t dummy_buf[2];
    uint8_t dummy_write = 0;

    TRF79x0_writeSingle(0x48, TRF79X0_ISO_CONTROL_REG);            // Step 16

    g_ui8DirectMode = 1;  // Enter Direct by keeping Slave Select Low
    TRF79x0_writeSingle(0x61, TRF79X0_CHIP_STATUS_CTRL_REG);       // Step 17
    TRF79x0_rawWrite(&dummy_write, 1);     // 8 dummy clocks       // Step 18
}
```

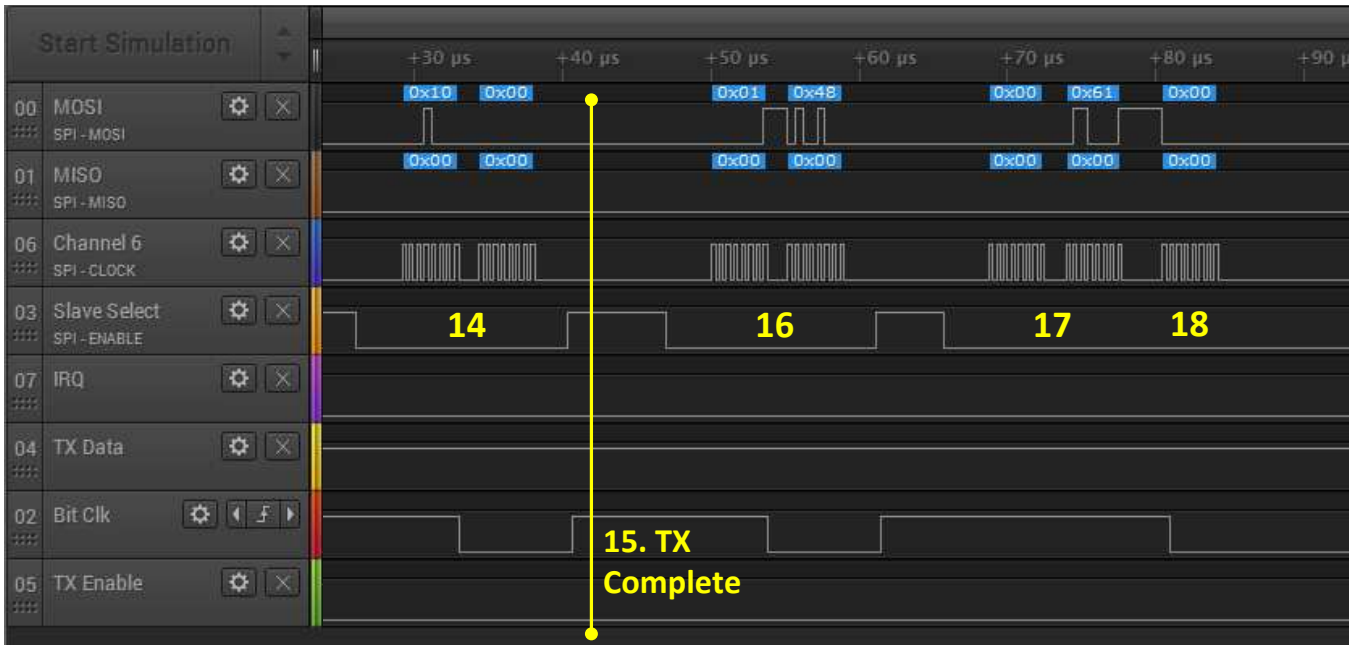

Figure 10. Entering Direct Mode 1

4.3.2 DM1 Receive

This receiving routine is probably the most sensitive to changes in clock speed. For slower MCUs, a polling routine is going to be most the most efficient but is timing sensitive. Because the polling looks for edges, if the clock speed slows down, an entire clock pulse may be missed. If the clock is sped up, the routine may think that it is already on the next bit and latch again. Interrupt driven routines were also tested, but the extra overhead of executing jump commands ended up requiring a faster clock source than the polling method.

In order to easily assist in debugging the DM1 receive routine, the TX Data line was used as a trigger for when each bit was latched. This allows the timing of the DM1 receive routine to easily tested.

To further reduce timing demands, the routine searches for the falling edges instead of the rising edges. The RX data is updated on the falling SDM Bit Clock edge so it can be latched on the rising edge. Each bit is precisely 9.44 µs long, so this only gives the host 4.72 µs to detect the rising edge and latch the data. By detecting the falling edge instead, the host has approximately 9.4 µs to latch the data.

Steps to receive data in Direct Mode 1:

19. Receive data in Direct Mode 1
    a. Data is updated on the bit clock falling edge.
    b. Data is received LSB first.
    c. IRQ goes high to indicate Receive Complete.

```
void MIFARE_DM1_Receive(uint16_t *pui16RxData, uint8_t *pui8ByteCount, uint8_t *pui8BitCount,
uint8_t bParityFlag){

   uint8_t ui8TempByteCount = 0;
   uint8_t ui8TempBitCount = 0;
   uint8_t ui8ByteWidth = 0;

#ifdef DM1_RX_DEBUGGING_SIGNAL
   SDM_DATA_OFF;
#endif  /* DM1_RX_DEBUGGING_SIGNAL */

   MISO_DATA_SETUP;      // DATA signal

   ui8TempByteCount = 0;
   ui8TempBitCount = 0;
   pui16RxData[ui8TempByteCount] = 0; // initialize buffer as used

   if(bParityFlag){
      ui8ByteWidth = 9;
   }
   else{
      ui8ByteWidth = 8;
   }

/*
 * In Direct Mode 1 Receiving, RX_Data (MISO) is updated on the falling Bit_Clk edge
 * (I/O_5), and latched on the rising Bit_Clk edge.  To make this decode functionality
 * work @ 8MHz DCO, I only check for the Bit_Clk falling edge, then sample the Data.
 * This allows us to latch proper data, and gives us the best time flexibility.
 *
 * Currently, interrupts are disabled in Direct mode.
 */

   while( !(SDM_BITCLK || g_ui8TimeOutFlag) );  // wait for first rising edge: Start of RX

   do{

#ifdef DM1_RX_DEBUGGING_SIGNAL
   SDM_DATA_ON;
#endif
```

```
      if(DM1_DATA_BIT){      // If RX_data == "1" (Buffer is pre-cleared to all 0x00.)
         pui16RxData[ui8TempByteCount] |= (0x01 << ui8TempBitCount);            // STEP 19
      }

      if(g_ui8IrqFlag){      // need to break free before incrementing bit_count
         break;
      }

      ui8TempBitCount++;

      if(ui8TempBitCount == ui8ByteWidth){            // Store Data in buffer
         ui8TempByteCount++;
         ui8TempBitCount = 0;
         pui16RxData[ui8TempByteCount] = 0;           // Pre-clear next Buffer byte
      }

#ifdef DM1_RX_DEBUGGING_SIGNAL
   SDM_DATA_OFF;
#endif

      while(SDM_BITCLK){                              //wait for next falling edge
         if((g_ui8IrqFlag) || (g_ui8TimeOutFlag)){    // breakouts for IRQ (EOF) and timeout
            break;
         }
      }

   }while( (!g_ui8IrqFlag) && (!g_ui8TimeOutFlag) );  // While Receive not complete.

   *pui8ByteCount = ui8TempByteCount;
   *pui8BitCount = ui8TempBitCount;

   MISO_SPI_SETUP;      // Switch MISO back to SPI pin
}
```

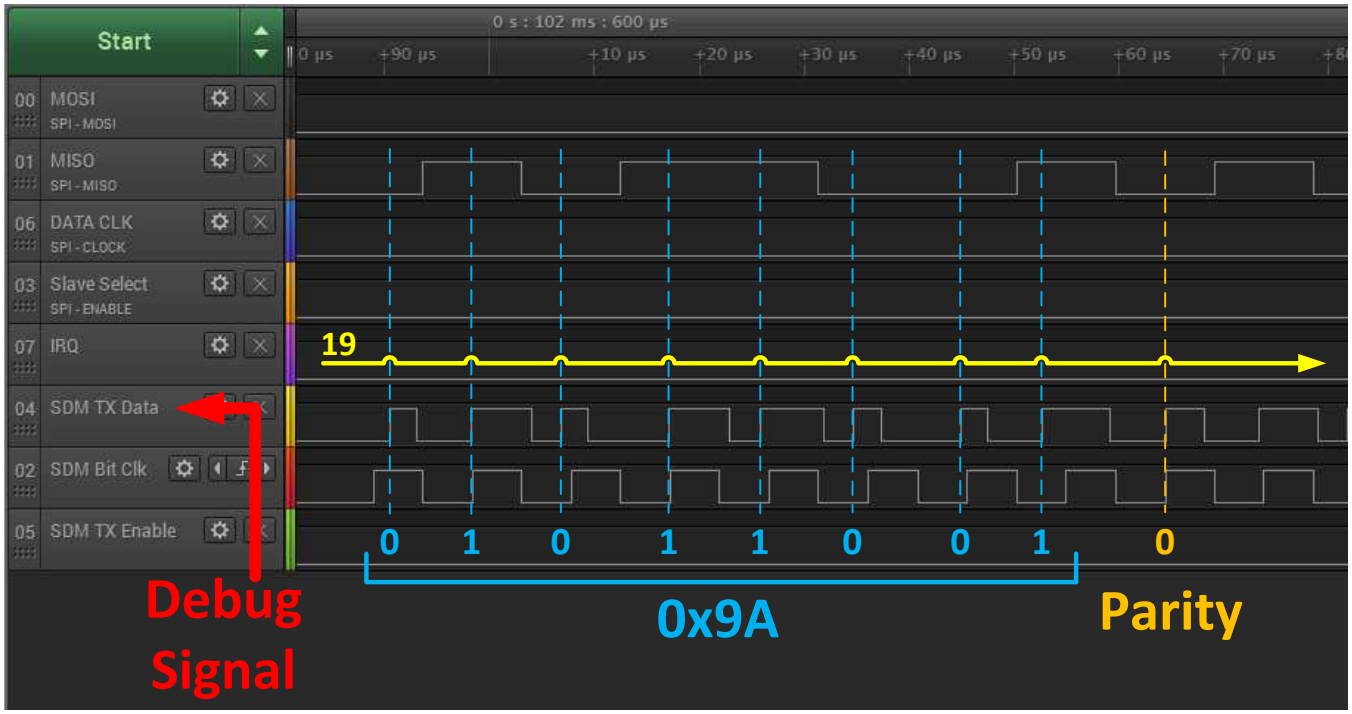

Figure 11. Receiving in Direct Mode 1

4.3.3 DM1 Exit

Steps to exit Direct Mode 1:

20. IRQ Goes high to signal RX Complete
21. Stop condition issued on SPI (SS line brought high)
22. Write Register 0x10 with 0x00 (disabling Special Direct Mode)
23. Clear the IRQ by reading register 0x0C

```
void MIFARE_dm1Exit(void)
{
    uint8_t temp[1];

    SLAVE_SELECT_HIGH;                                              // Step 21
    g_ui8DirectMode = 0;

    MCU_timerDisable();

    TRF79x0_writeSingle(0x00, TRF79X0_SPECIAL_FUNC_1_REG);          // Step 22

    TRF79x0_writeSingle(0x88, TRF79X0_ISO_CONTROL_REG);

    g_ui8IrqFlag = 0;
    TRF79x0_readSingle(temp, TRF79X0_IRQ_STATUS_REG);  // Clear IRQ  // Step 23
}
```

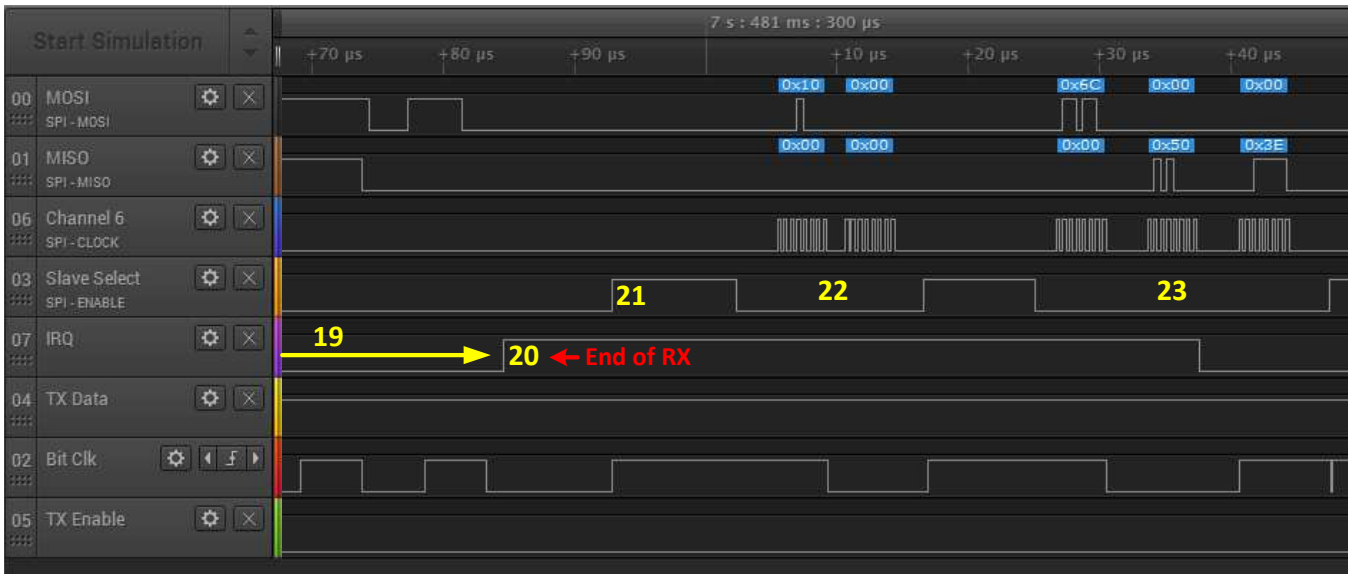

Figure 12. Exiting Direct Mode 1

5 Conclusion

This application note presents everything required to implement Special Direct Mode and Direct Mode 1. These modes can be used to support proprietary or custom protocols. In this examples case, MIFARE Classic authentication was used to demonstrate the flexibility that these modes enable.

There are still some timing and clock demands on the host. MIFARE Classic tags respond quickly to most commands, approximately 100 µs, and each bit is only 9.44 µs wide. Because of these requirements, the slowest a host could probably run would be approximately 6 MHz, but at least 8 MHz is recommended. This example was developed and demonstrated at 8 MHz.

Source code discussed in this application report can be downloaded from http://www.ti.com/lit/zip/sloa214.

For follow up questions concerning this reference design, visit the E2E™ online community for NFC at http://e2e.ti.com/support/wireless_connectivity/f/667.

6 References

1. *TRF7970A Multi-Protocol Fully Integrated 13.56-MHz RFID and NFC Transceiver IC* (SLOS743)

2. *TRF7970A Firmware Design Hints* (SLOA159)

3. ISO14443-2 (http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50941)

4. ISO14443-3 (http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50942)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265