Mifare Classic Operations with TRF79xxA NFC/RFID Transceivers

S2 Microcontroller Division

NFC/RFID Applications Team

03/2014

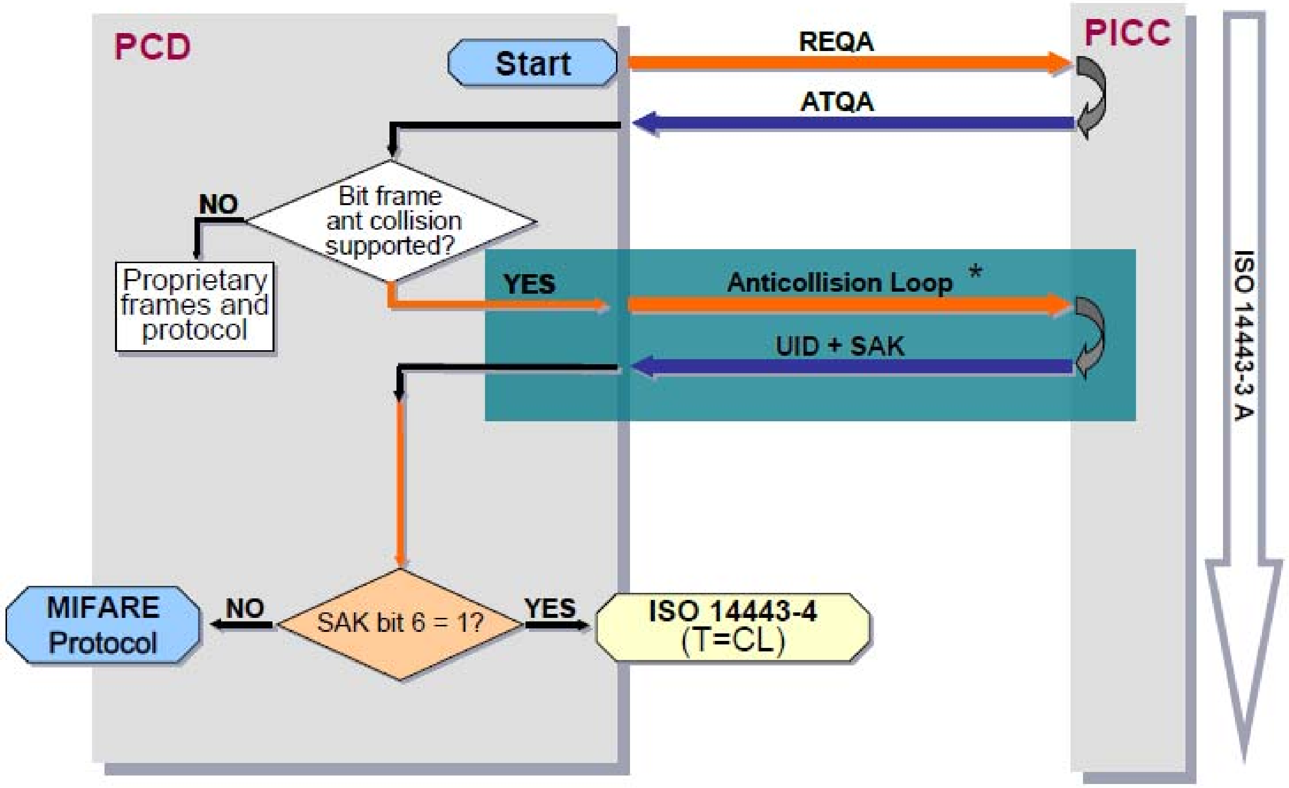
Agenda

- Mifare Classic Operations Overview & Standards based timings
- Technical Information about using the TRF79xxA devices with the Mifare Classic transponders
  - Activation & Selection
  - Authentication
  - Read Blocks & Sectors
  - Writing Blocks & Sectors
- Handling 7 byte UID Mifare Classic Tags
- Timing Observations
  - Activation and Selection
  - Authenticaiton
  - Read Block
  - Read Sector
  - Conclusion
- Backup Slides

Mifare Classic Operations Overview

- Mifare Classic uses ISO14443A air interface protocol, so TRF79xxA is setup for ISO14443A, and Mifare Classic card UID is read and then selected.
- After this point, a three round authentication must take place. This is where Mifare deviates from the ISO standard so the TRF79xxA must be placed in Direct Mode 0 (Analog Front End Mode).
- For TX, this is the relationship of MOD pin to carrier, as the MCU must modulate the RF according to the ISO14443A air interface and do so in accordance with Mifare Classic protocol.
- For RX, this is the digitized data bit stream from I/O_6, which the MCU must decode, according to the ISO14443A air interface standard.

Mifare Classic Activation Sequence

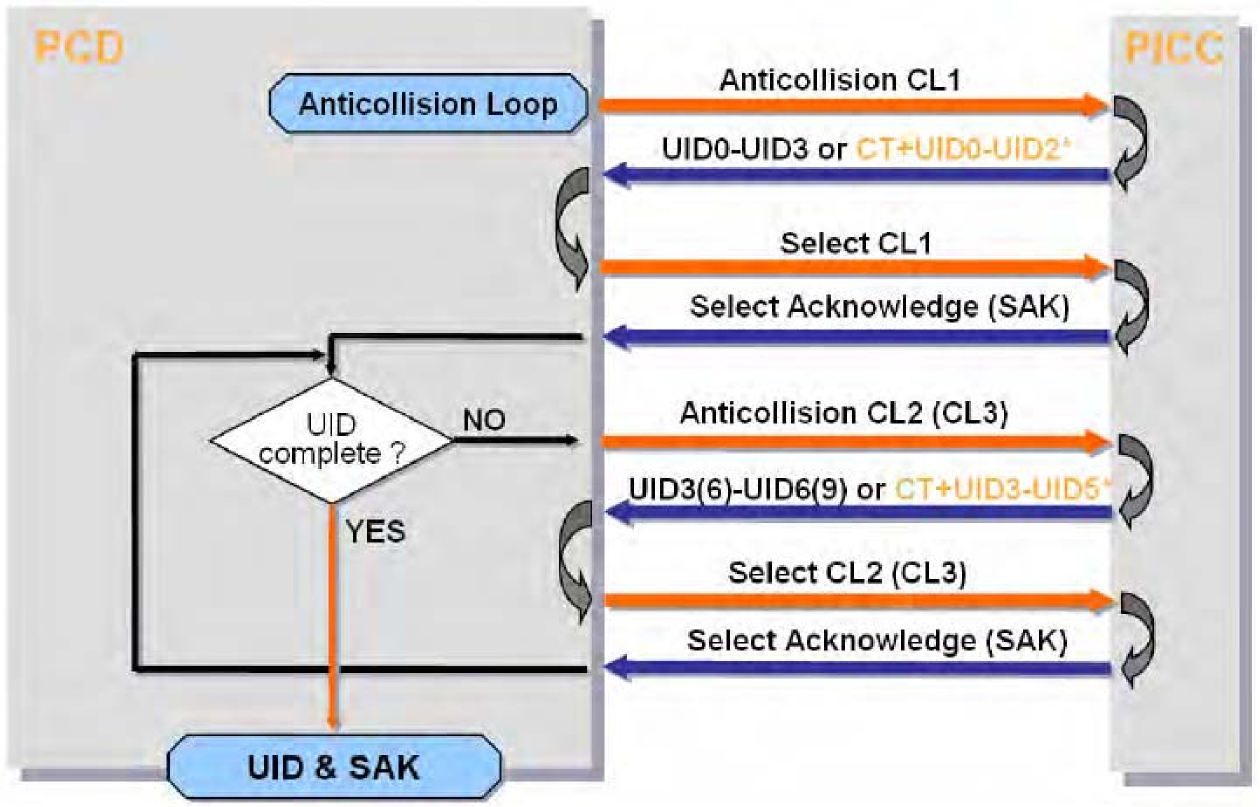


(1) Details of the Anticollision Loop see Fig 2.

Fig 1. Principle of the Card Activation Sequence

- The bit 6 in the SAK indicates, whether the PICC is compliant to the ISO/IEC14443-4 or not. However, it does not necessarily indicate, whether the PICC supports the MIFARE Protocol or not.

Mifare Classic Anti-collision Loop



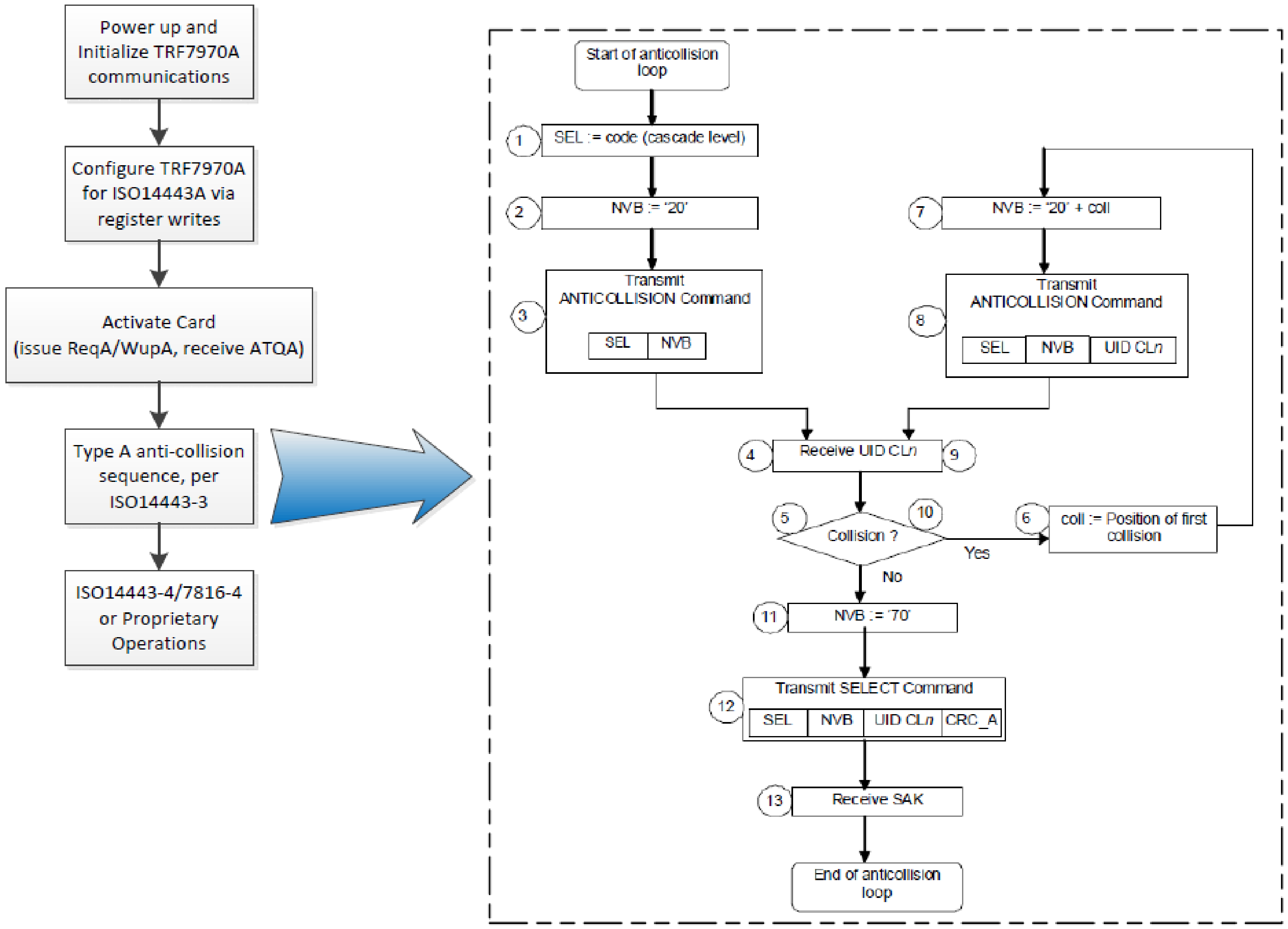
(3) CT = Cascade Tag

(4) CL = Cascade Level

Fig 2. Anticollision sequence

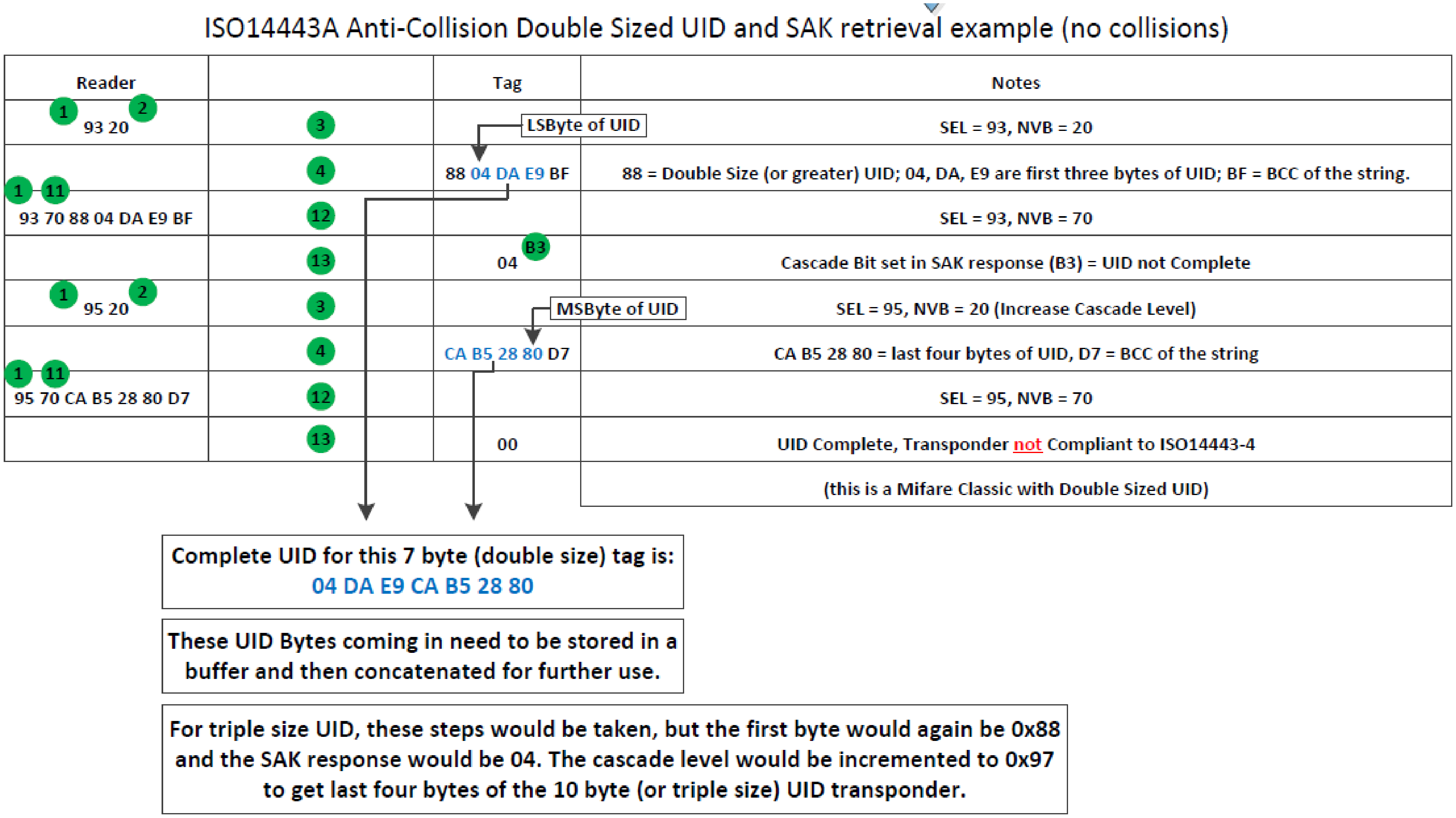
- Mifare tags recently started being released with double sized (7 byte) UID's. Mifare 4 byte UID's are now labeled "non-unique". Currently there are no triple sized UID Mifare tags.

ISO14443A Anti-collision Loop



Power up and Initialize TRF7970A communications
Configure TRF7970A for ISO14443A via register writes
Activate Card (issue ReqA/WupA, receive ATQA)
Type A anti-collision sequence, per ISO14443-3
ISO14443-4/7816-4 or Proprietary Operations
Start of anticollision loop
1 SEL := code (cascade level)
2 NVB := '20'
3 Transmit ANTICOLLISION Command
SEL
NVB
4 Receive UID CLn
9
5 Collision ?
10
Yes
No
6 coll := Position of first collision
7 NVB := '20' + coll
8 Transmit ANTICOLLISION Command
SEL
NVB
UID CLn
11 NVB := '70'
12 Transmit SELECT Command
SEL
NVB
UID CLn
CRC_A
13 Receive SAK
End of anticollision loop

ISO14443A Selection Process



ISO14443A Anti-Collision Double Sized UID and SAK retrieval example (no collisions)

Reader		Tag	Notes
1 93 20 2	3	LSByte of UID	SEL = 93, NVB = 20
	4	88 04 DA E9 BF	88 = Double Size (or greater) UID; 04, DA, E9 are first three bytes of UID; BF = BCC of the string.
1 11 93 70 88 04 DA E9 BF	12		SEL = 93, NVB = 70
	13	04 B3	Cascade Bit set in SAK response (B3) = UID not Complete
1 95 20 2	3	MSByte of UID	SEL = 95, NVB = 20 (Increase Cascade Level)
	4	CA B5 28 80 D7	CA B5 28 80 = last four bytes of UID, D7 = BCC of the string
1 11 95 70 CA B5 28 80 D7	12		SEL = 95, NVB = 70
	13	00	UID Complete, Transponder not Compliant to ISO14443-4
			(this is a Mifare Classic with Double Sized UID)

Complete UID for this 7 byte (double size) tag is:
04 DA E9 CA B5 28 80

These UID Bytes coming in need to be stored in a buffer and then concatenated for further use.

For triple size UID, these steps would be taken, but the first byte would again be 0x88 and the SAK response would be 04. The cascade level would be incremented to 0x97 to get last four bytes of the 10 byte (or triple size) UID transponder.

NXP’s Mifare Tag Types

- You can see in the table below the different functionality available in NXP’s MIFARE tags.

- These tags are quite common in the field.

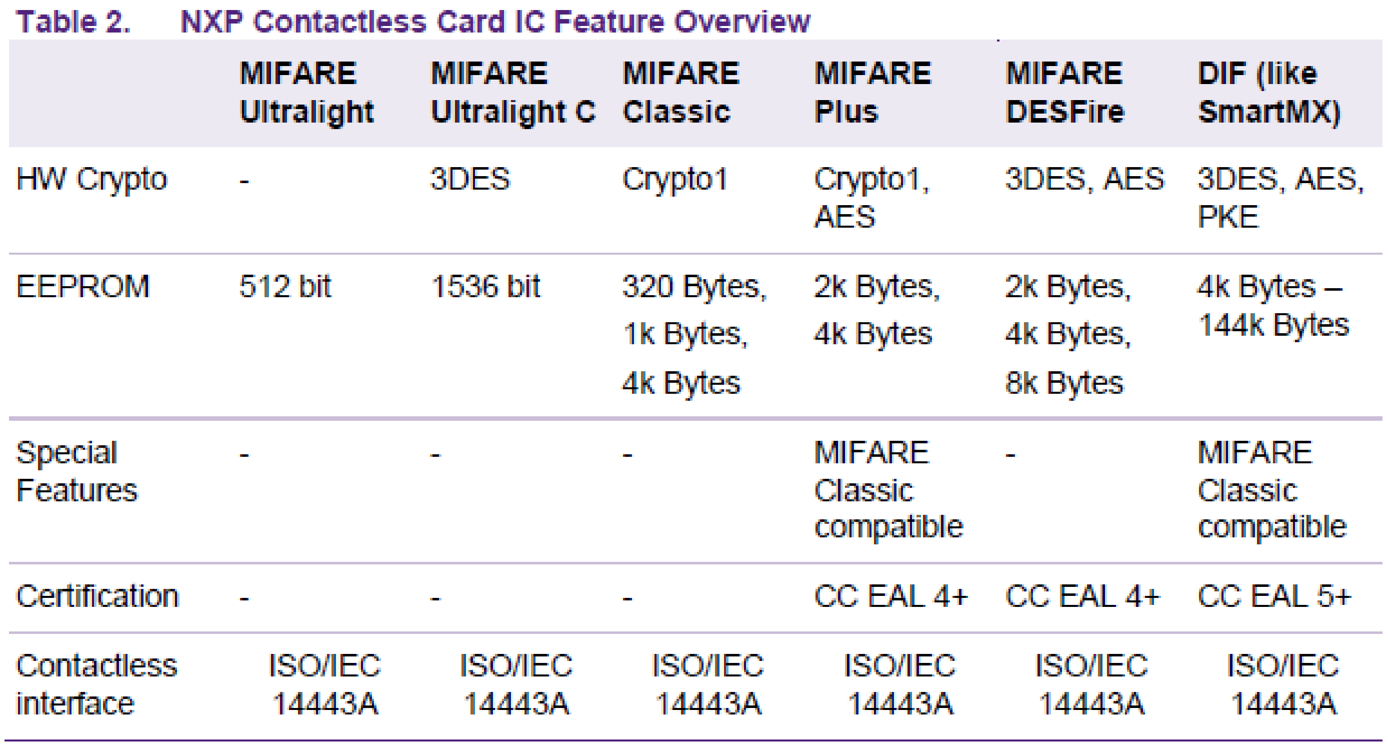


Table 2. NXP Contactless Card IC Feature Overview

	MIFARE Ultralight	MIFARE Ultralight C	MIFARE Classic	MIFARE Plus	MIFARE DESFire	DIF (like SmartMX)
HW Crypto	-	3DES	Crypto1	Crypto1, AES	3DES, AES	3DES, AES, PKE
EEPROM	512 bit	1536 bit	320 Bytes, 1k Bytes, 4k Bytes	2k Bytes, 4k Bytes	2k Bytes, 4k Bytes, 8k Bytes	4k Bytes – 144k Bytes
Special Features	-	-	-	MIFARE Classic compatible	-	MIFARE Classic compatible
Certification	-	-	-	CC EAL 4+	CC EAL 4+	CC EAL 5+
Contactless interface	ISO/IEC 14443A	ISO/IEC 14443A	ISO/IEC 14443A	ISO/IEC 14443A	ISO/IEC 14443A	ISO/IEC 14443A

ISO14443A Standard Important Timings

- 128/fc = 9.435uSec = t_b (106kbps data rate)
- 64/fc = 4.719uSec = t_x time
- 32/fc = 2.359uSec = t_1 time

Figure 10 together with the timing parameters in Table 7 illustrate sequences X, Y and Z.

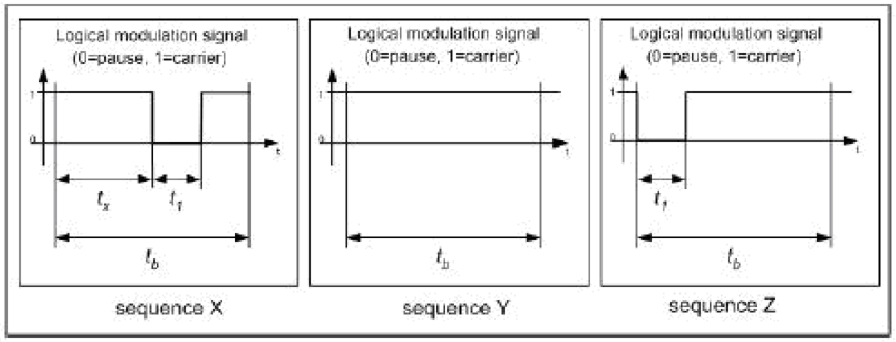
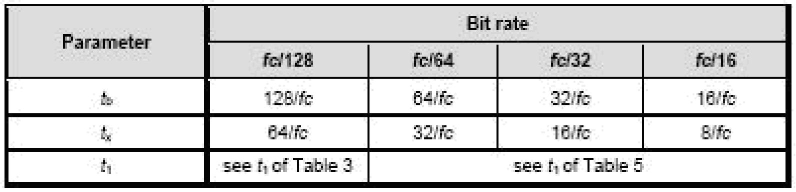
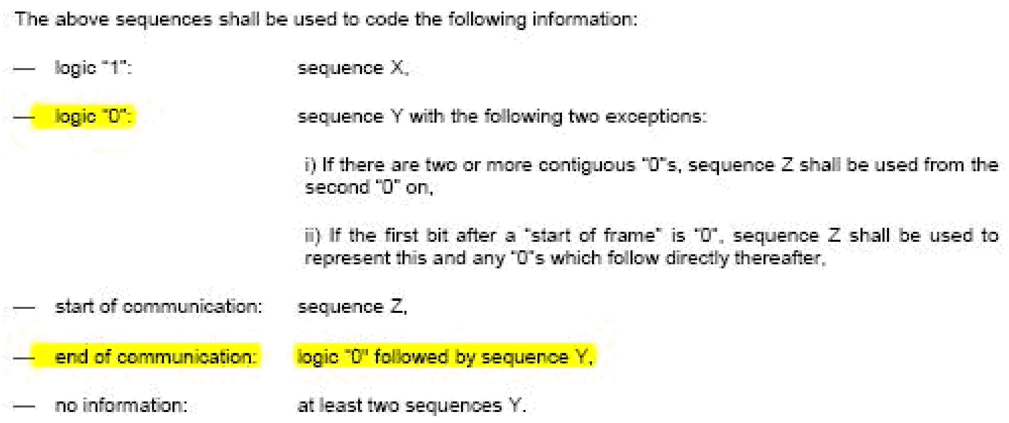


Figure 10 — Sequences for Type A communication PCD to PICC

Table 7 — Parameters for sequences



Parameter	Bit rate			
	fc/128	fc/64	fc/32	fc/16
t_b	128/fc	64/fc	32/fc	16/fc
t_x	64/fc	32/fc	16/fc	8/fc
t_1	see t_1 of Table 3	see t_1 of Table 5		



The above sequences shall be used to code the following information:

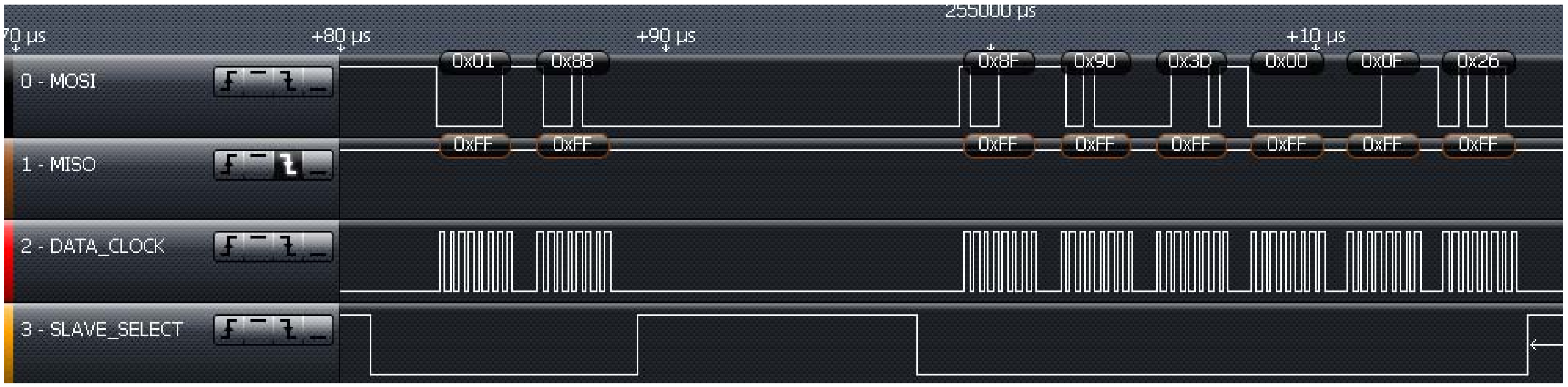
— logic "1":	sequence X,
— logic "0":	sequence Y with the following two exceptions:
	i) If there are two or more contiguous "0"s, sequence Z shall be used from the second "0" on,
	ii) If the first bit after a "start of frame" is "0", sequence Z shall be used to represent this and any "0"s which follow directly thereafter,
— start of communication:	sequence Z,
— end of communication:	logic "0" followed by sequence Y,
— no information:	at least two sequences Y.

Technical Information

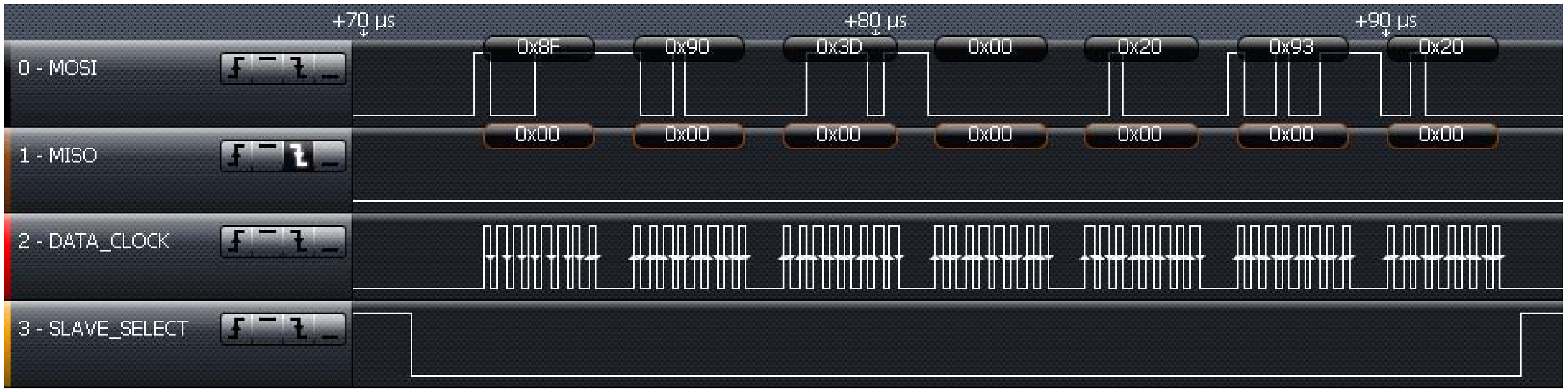
- The following several slides go into the details of using the TRF79xx devices first in Direct Mode 2 to activate the card and select it, then in Direct Mode 0 during Mifare Card specific operations.
  - Note: subsequent slides do not show all register configurations, but this has been done beforehand, per the TRF79xxA datasheets.

Reading and Selecting the Mifare Card in Direct Mode 2

- TRF79xx is configured for ISO14443A operations, ISO Control Register is set for RX w/no CRC present in response and REQA is issued.

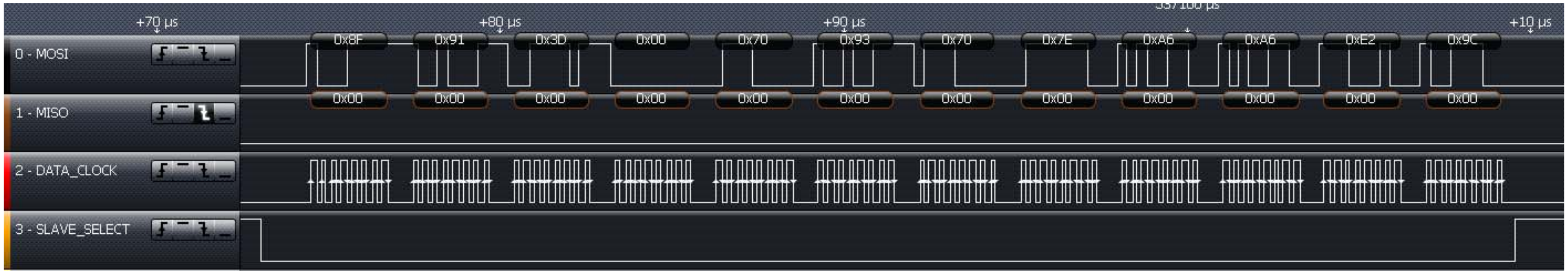


- ATQA is received, and then anti-collision sequence is started.

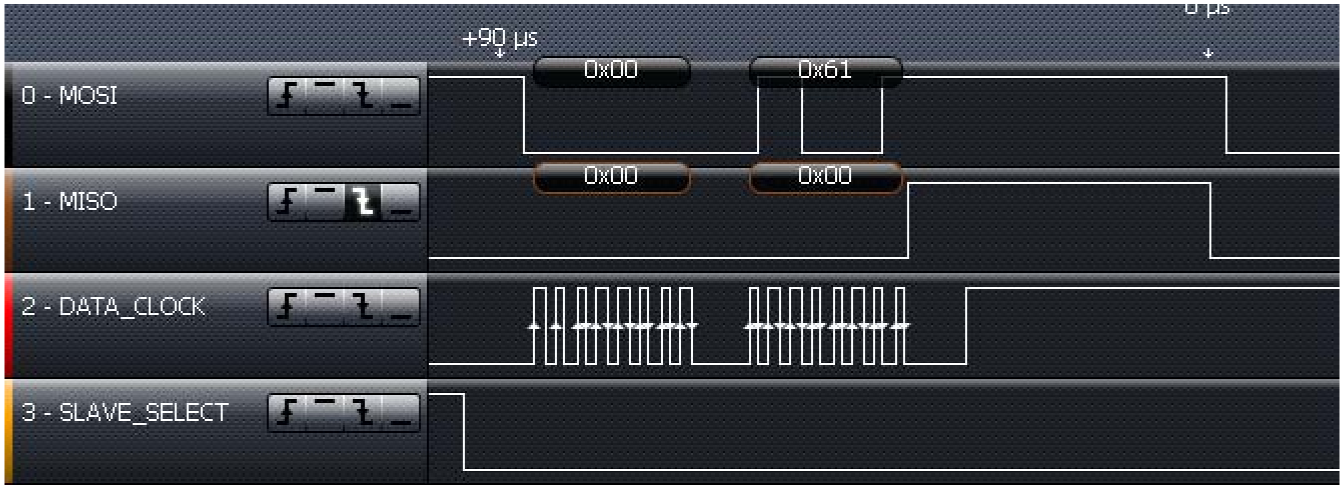


Reading and Selecting the Mifare Card in Direct Mode 2

- We receive the UID CLn, and then transmit the SELECT Command.

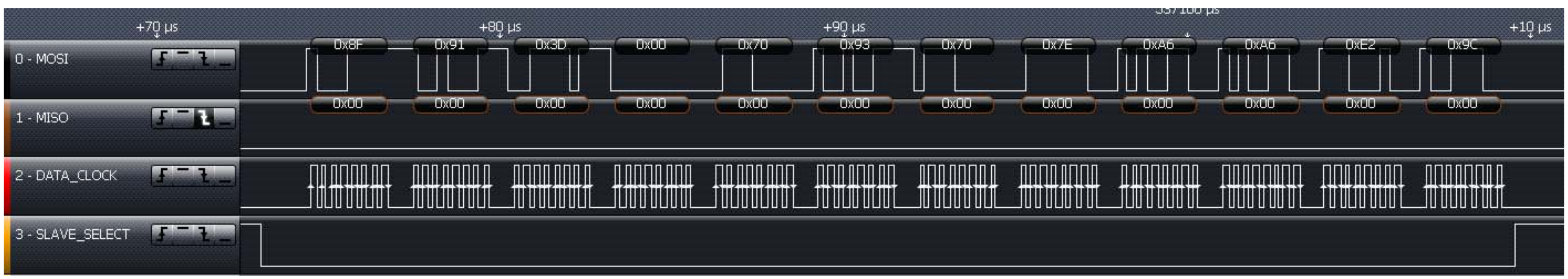


- We receive the SAK, and then the TRF7960/-60A is placed into Direct Mode 0.



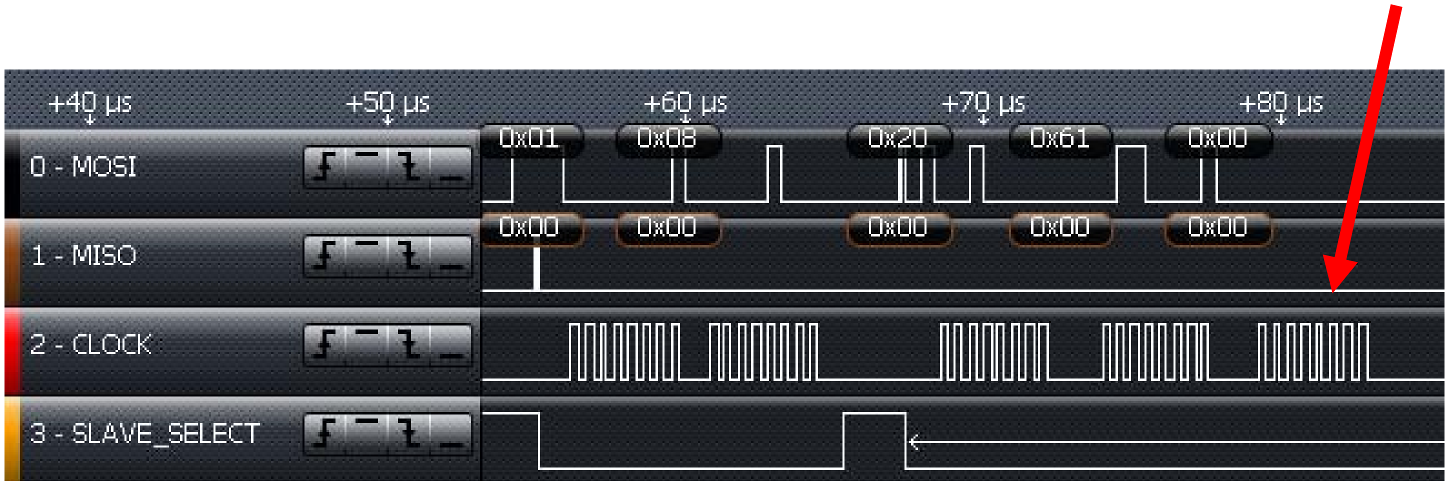
Reading and Selecting the Mifare Card in Direct Mode 2
(cont., TRF7970A DM0 specific)

- We receive the UID CLn, and then transmit the SELECT Command.



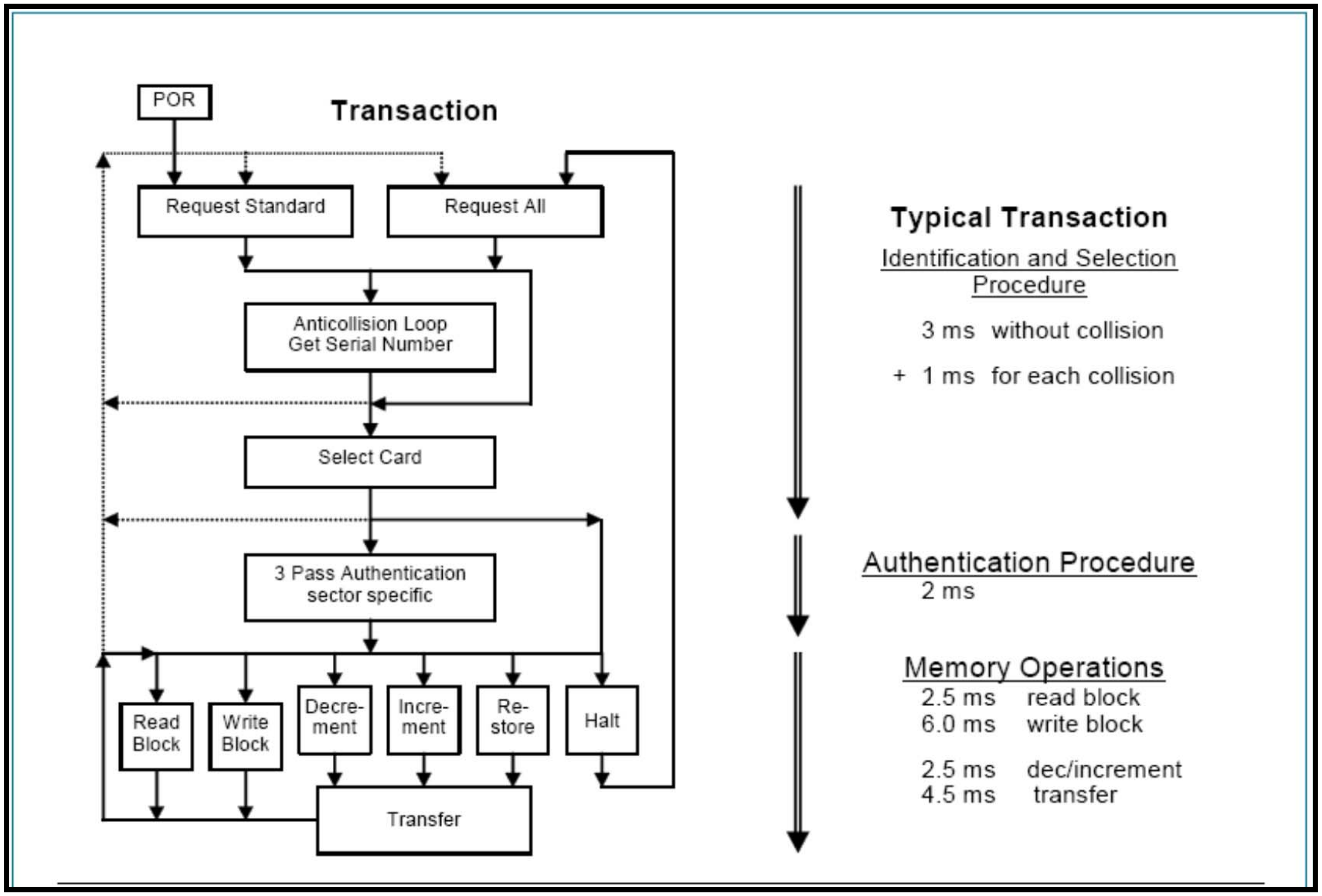
- We receive the SAK, and then the TRF7970A is placed into Direct Mode 0.

NOTE : see extra 8 clocks after sending 0x61 to Chip Status Control Register (Direct Mode 0) setting



Mifare Card Interaction Flow

(from NXP MF1ICS50 Data Sheet, marked public document)



Typical Transaction

Identification and Selection Procedure

3 ms without collision

+ 1 ms for each collision

Authentication Procedure
2 ms

Memory Operations

2.5 ms	read block
6.0 ms	write block
2.5 ms	dec/increment
4.5 ms	transfer

Three pass authentication sequence

1. The reader specifies the sector to be accessed and chooses key A or B.
2. The card reads the secret key and the access conditions from the sector trailer. Then the card sends a random number as the challenge to the reader (pass one).
3. The reader calculates the response using the secret key and additional input. The response, together with a random challenge from the reader, is then transmitted to the card (pass two).
4. The card verifies the response of the reader by comparing it with its own challenge and then it calculates the response to the challenge and transmits it (pass three).
5. The reader verifies the response of the card by comparing it to its own challenge.

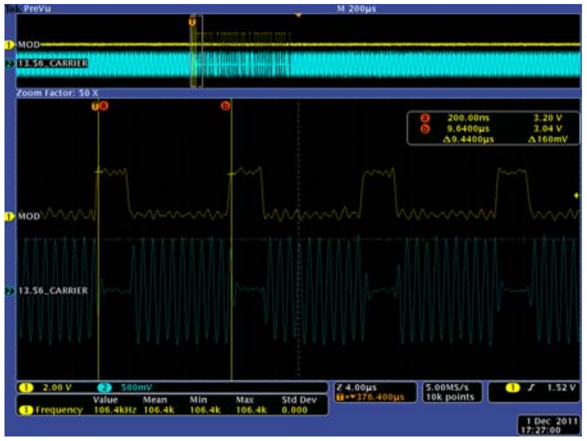
NOTE: After transmission of the first random challenge the communication between card and reader is encrypted.

Three pass authentication sequence (cont.)

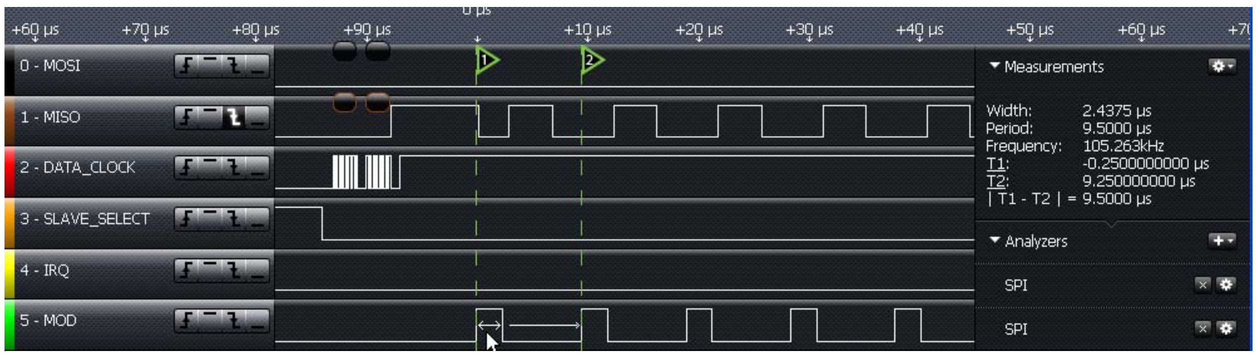
- The AUTH protocol does:
  - Reader sends 32 bit random number (+ error correction)
  - Card responds with 32 bit key stream XOR a derivative of the reader random number XOR its own random number
  - Reader responds with 32 bit key stream XOR the card random number
- **NOTE: None of this information can really be checked for plausibility without the secret keys, as it's either random or encrypted (or both mixed). This is why you will see later in the slides that 38 bits are sent out instead of the 80 bits that one might expect from reading publicly available NXP documents.**

Using the TRF79xx in Direct Mode 0 for Authentication

- At this point, the MOD pin is being driven by the MCU, which in turn is driving the TRF79xx transmitter, in accordance with the ISO14443A air interface specifications. For example, below is a logic analyzer capture showing the TRF79xx being put into Direct Mode 0, and also shown below is exact same thing, but on oscilloscope, so that the RF output can be correlated to how MOD pin is being driven. (next slide shows full size of the RF screen shot)



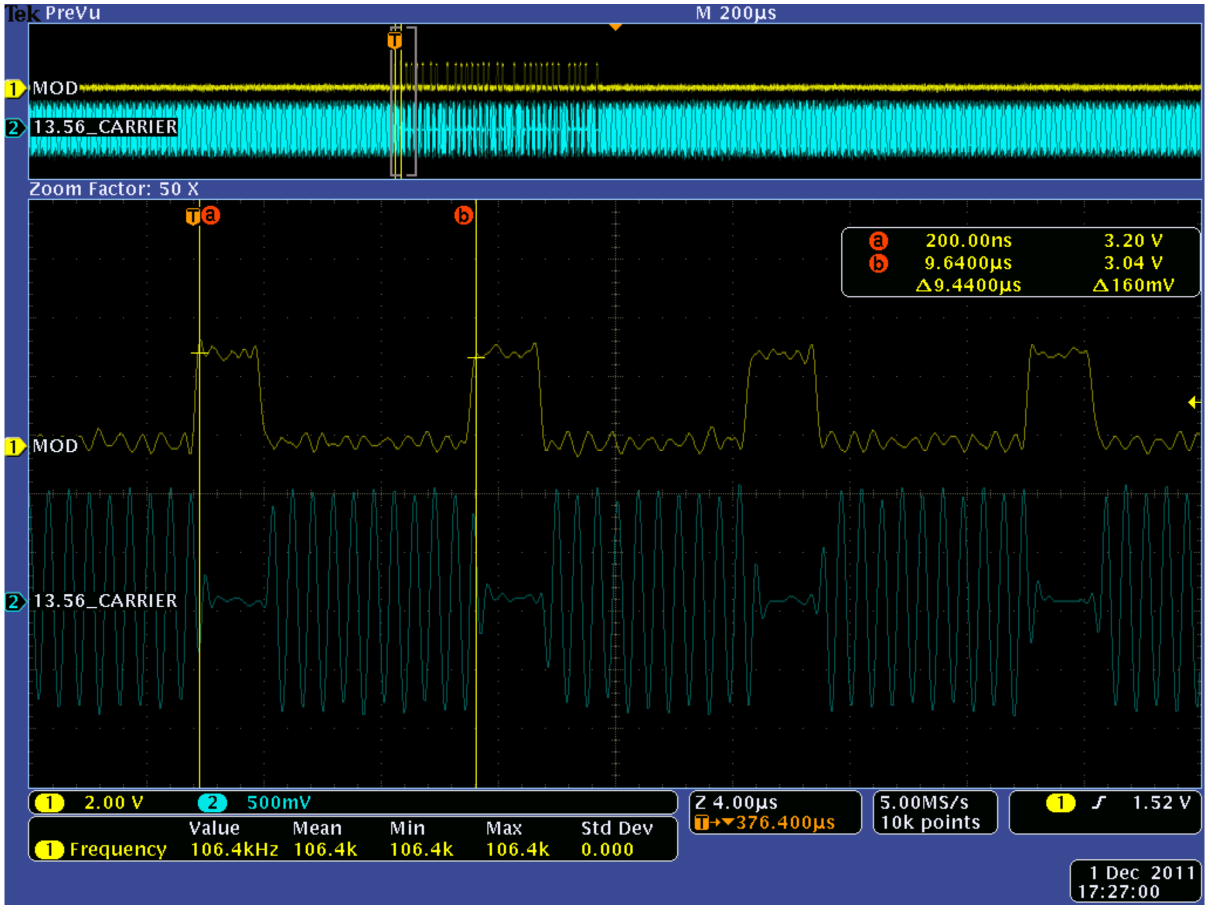
	Value	Mean	Min	Max	Std Dev
Frequency	106.4kHz	106.4k	106.4k	106.4k	0.000



Width: 2.4375 µs
Period: 9.5000 µs
Frequency: 105.263kHz
T1: -0.2500000000 µs
T2: 9.250000000 µs
| T1 - T2 | = 9.5000 µs

ISO14443A SOF (in Direct Mode 0)

- Per the ISO14443A Standard, this is sequence Z, which is used for the PCD to PICC Start of Communication (SOF)



a 200.00ns 3.20 V
b 9.6400µs 3.04 V
Δ9.4400µs Δ160mV

1 2.00 V 2 500mV

	Value	Mean	Min	Max	Std Dev
1 Frequency	106.4kHz	106.4k	106.4k	106.4k	0.000

Z 4.00µs
T→▾376.400µs

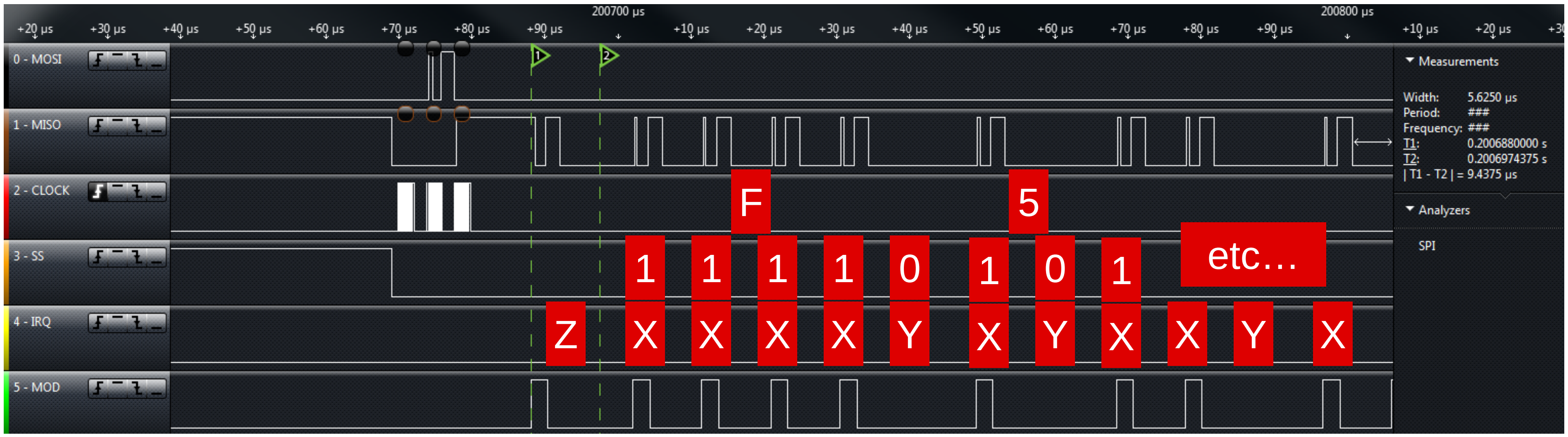
5.00MS/s
10k points

1 ʃ 1.52 V

1 Dec 2011
17:27:00

Using the TRF79xx in Direct Mode 0 for Authentication

- Here is another example, captured with TRF7970A this time, showing SOF after going into DM0.
  - Note extra clock cycle after register 0x00 is written with 0x61.
- Cursors are on MOD (ignore MISO line here) starting with the ISO14443A SOF (Sequence Z) - as when MOD is high, TX is off
- This is followed by bit stream starting with: XXXXYXYX…(11110101)…0xF5…and goes on.

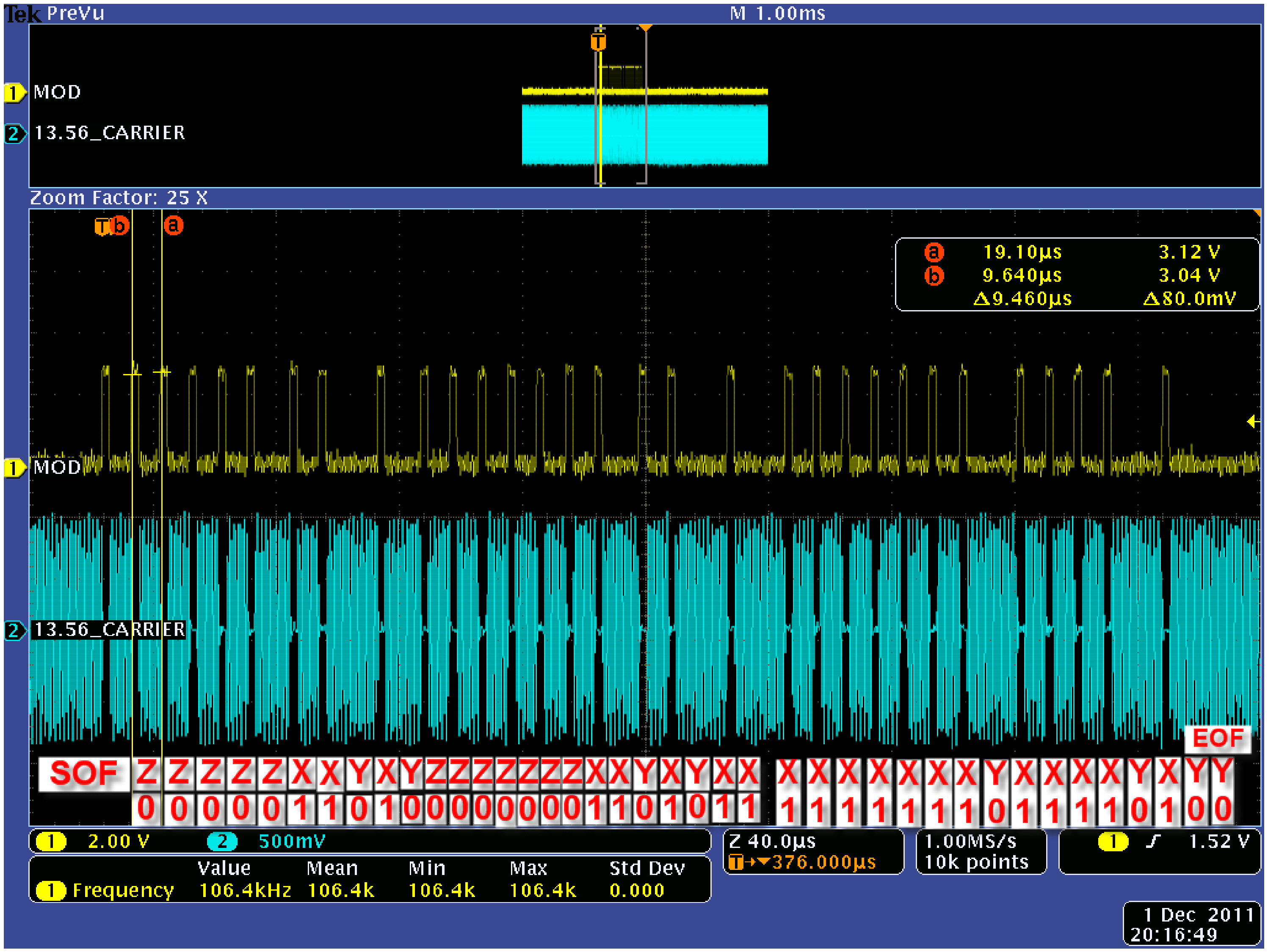


Width: 5.6250 µs
Period: ###
Frequency: ###
T1: 0.2006880000 s
T2: 0.2006974375 s
| T1 - T2 | = 9.4375 µs

MSP430 Code Snippet

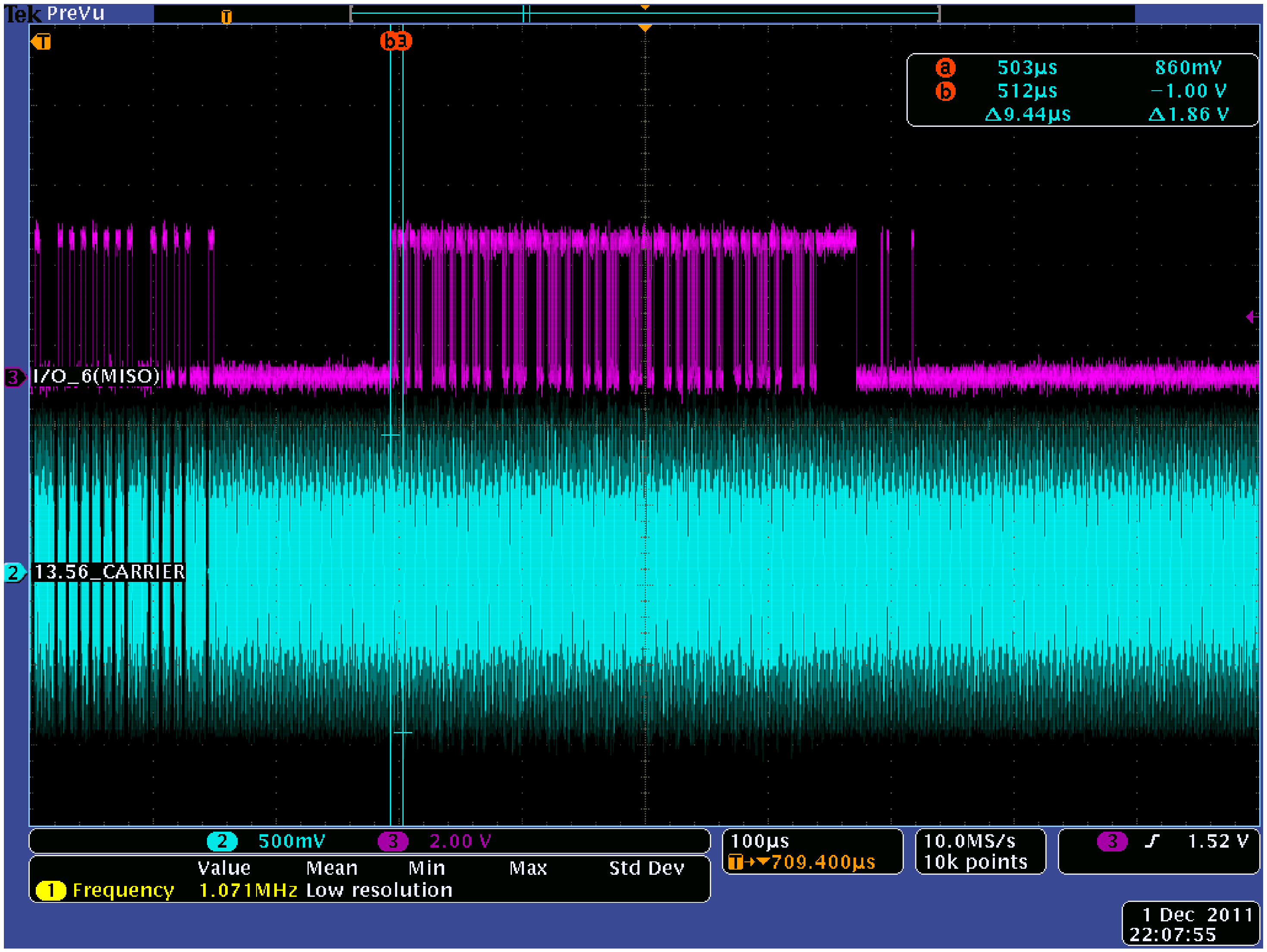
- * First stage of mutual authentication given a card's UID.
- * card_challenge is the card nonce as an integer
- */
- void crypto1_mutual_1(crypto1_state *state, uint32_t uid, uint32_t card_challenge)
- {
-         state->ops->mutual_1(state, uid, card_challenge);
- }

First TRF79xx TX Sequence Out in DM0 for Mifare Authentication



	Value	Mean	Min	Max	Std Dev
1 Frequency	106.4kHz	106.4k	106.4k	106.4k	0.000

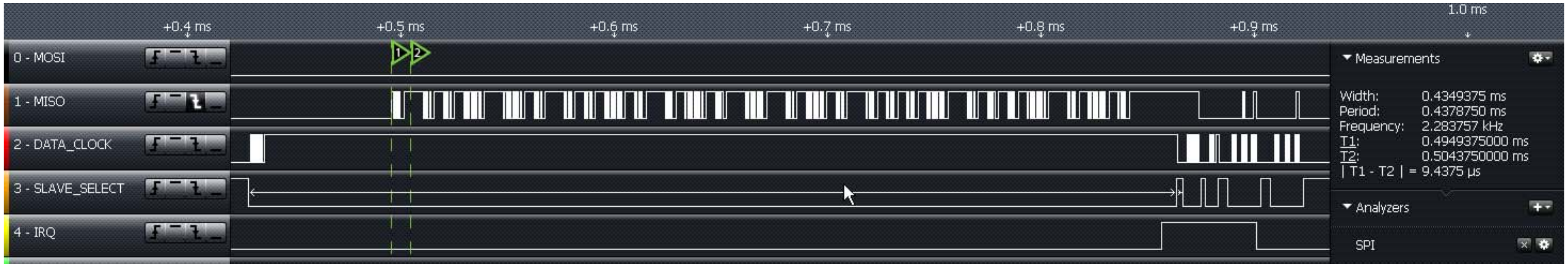
First TRF79xx RX Sequence on I/O_6 in DM0 for Mifare Authentication (Analog Capture)



```
a   503µs    860mV
b   512µs   −1.00 V
   Δ9.44µs  Δ1.86 V

2  500mV   3  2.00 V
        Value      Mean  Min  Max  Std Dev
1 Frequency  1.071MHz  Low resolution

100µs
T→▼709.400µs

10.0MS/s
10k points

3  ∫  1.52 V

1 Dec 2011
22:07:55
```

First TRF79xx RX Sequence on I/O_6 in DM0 for Mifare Authentication (cont., digital capture of previous slide)

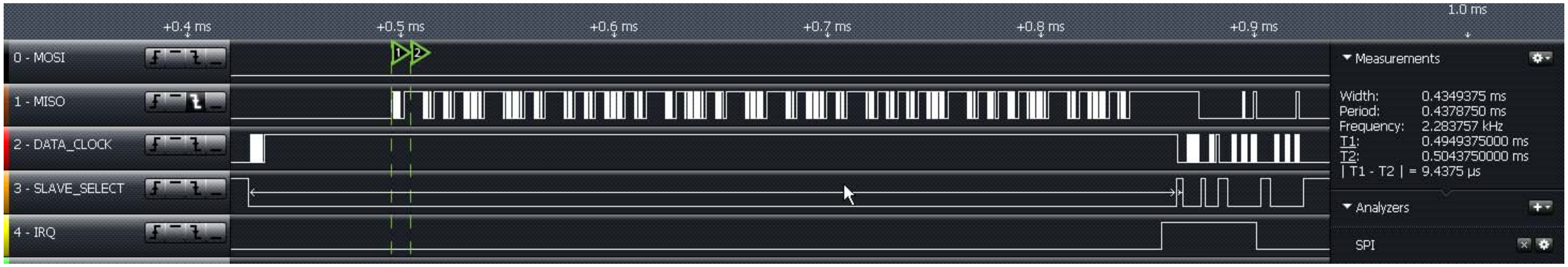
- PICC is communicating back @ 106kbps (fc/128), so:
  - The following sequences are defined:
    - Sequence D: the carrier shall be modulated with the subcarrier for the first half (50 %) of the bit duration,
    - Sequence E: the carrier shall be modulated with the subcarrier for the second half (50 %) of the bit duration,
    - Sequence F: the carrier is not modulated with the subcarrier for one bit duration.
  - Bit coding shall be Manchester with the following definitions:
    - logic "1": Sequence D
    - logic "0": Sequence E
    - Start of Communication: Sequence D
    - End of Communication: Sequence F
    - No Information: No Subcarrier.
- T1:T2 cursors are ISO14443A PICC to PCD SOF (Sequence D)



```
Width:       0.4349375 ms
Period:      0.4378750 ms
Frequency:   2.283757 kHz
T1:          0.4949375000 ms
T2:          0.5043750000 ms
| T1 - T2 | = 9.4375 µs
```

First TRF79xx RX Sequence on I/O_6 in DM0 for Mifare Authentication (cont., digital capture of previous slide)

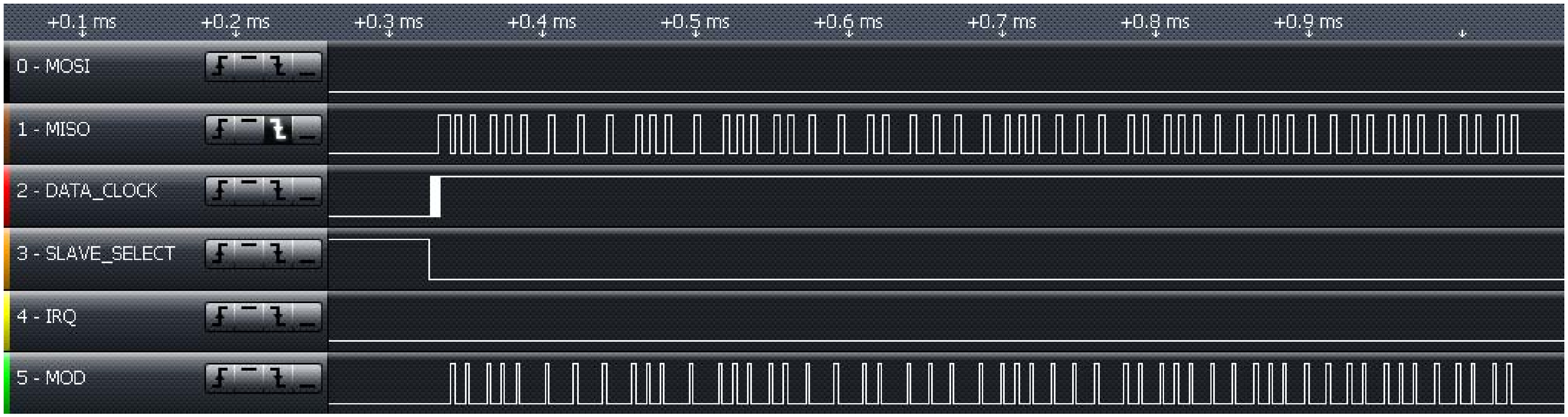
- Data is encrypted:



```
Width:      0.4349375 ms
Period:     0.4378750 ms
Frequency:  2.283757 kHz
T1:         0.4949375000 ms
T2:         0.5043750000 ms
| T1 - T2 | = 9.4375 µs
```

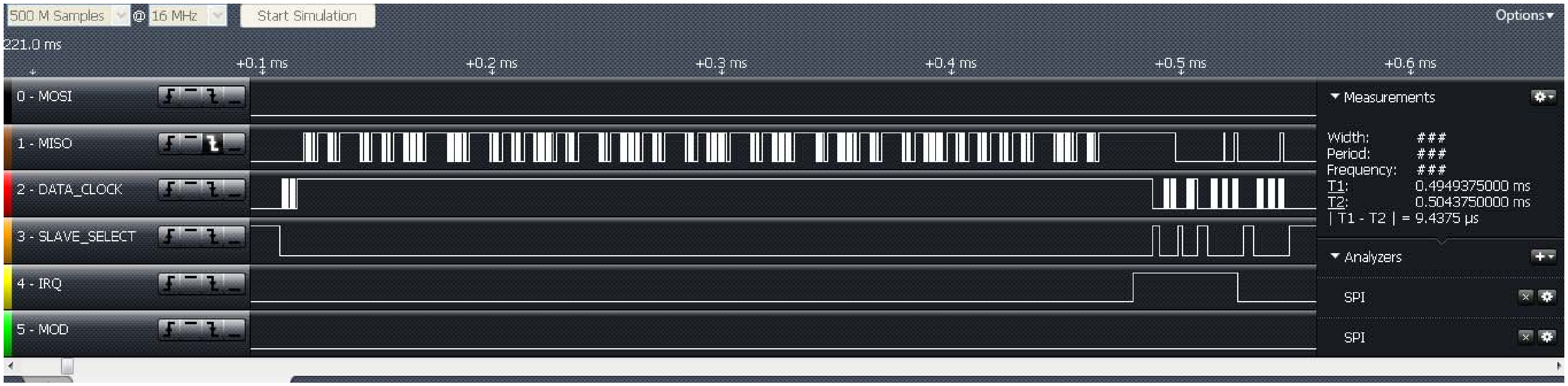
Second TRF79xx TX Sequence Out in DM0 for Mifare Authentication (analog)

- This is encrypted TX out.



Second TRF79xx RX Sequence on I/O_6 in DM0 for Mifare Authentication (cont., digital dapture of previous slide)

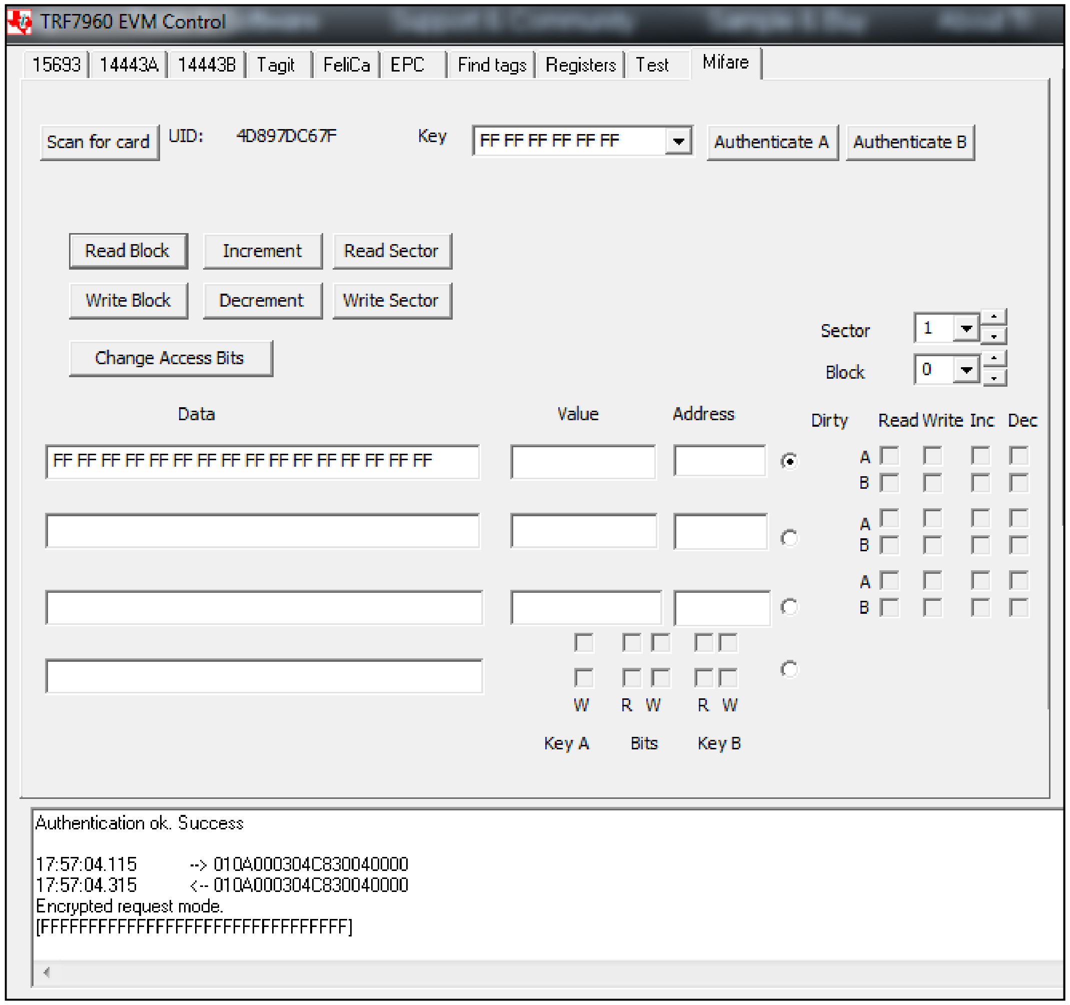
- This is encrypted RX in



```
Width:        ###
Period:       ###
Frequency:    ###
T1:           0.4949375000 ms
T2:           0.5043750000 ms
| T1 - T2 | = 9.4375 µs
```

Using the GUI to do Encrypted Read Operations

- Here we are using the GUI to do encrypted read of Block 0 in Sector 1 (it was previously programmed with all 0xFF data.
- Using Host Command Code 0xC830



UID: 4D897DC67F

Key FF FF FF FF FF FF

Sector 1

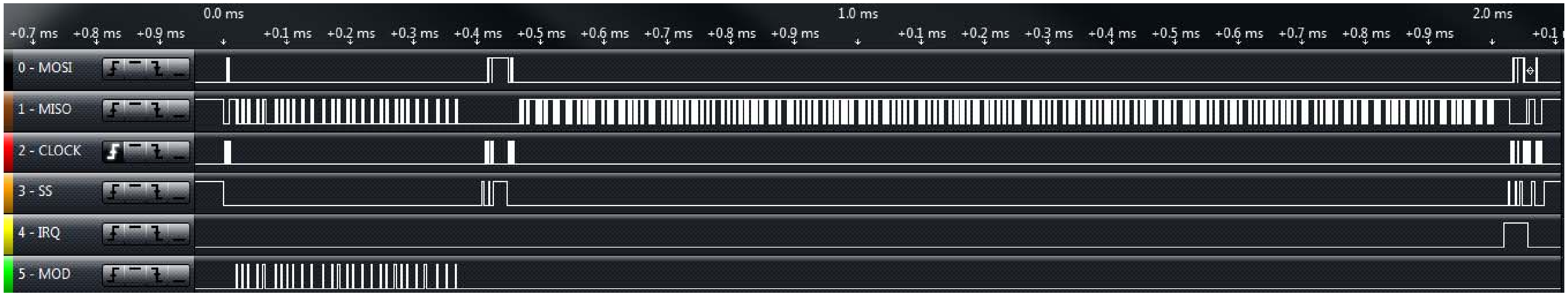
Block 0

Data: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

```
Authentication ok. Success

17:57:04.115        --> 010A000304C830040000
17:57:04.315        <-- 010A000304C830040000
Encrypted request mode.
[FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF]
```

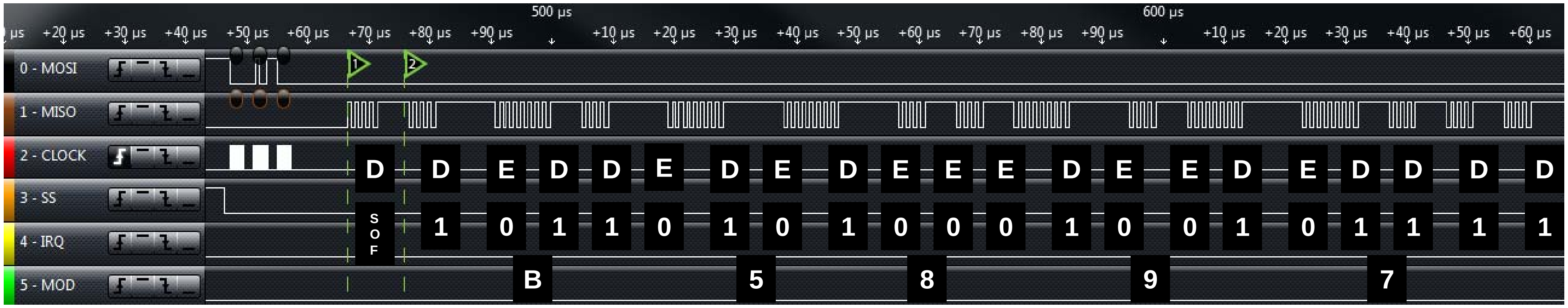
Logic Shots of Encrypted Read Command

- Here we have the Encrypted Read TX out and Encrypted RX in for one Block.
- This is done by TX out in DM0, modulating the carrier and RX in using DM0, recovering the demodulated and digitized subcarrier.



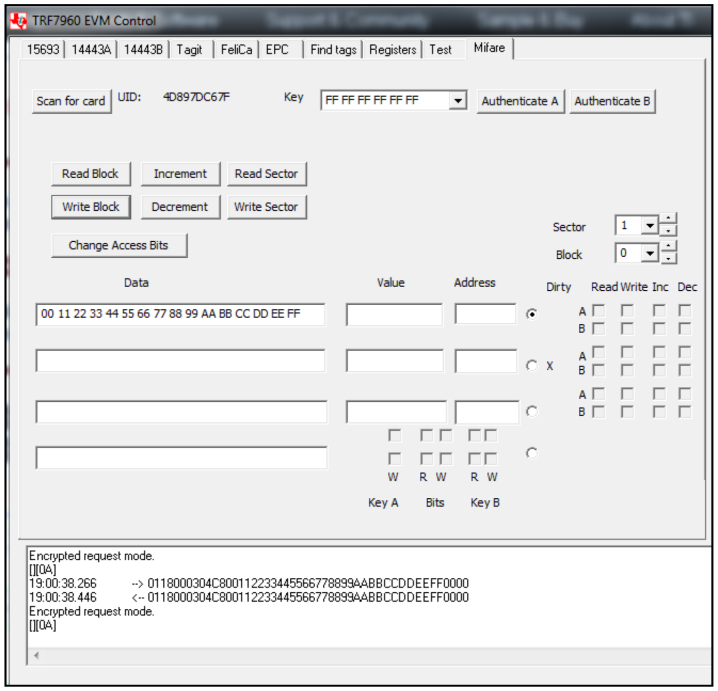
Logic Shots of Encrypted Read Response

- We have seen the TX out – here is the RX in response, zoomed – this is from reading a block of data.
- The encrypted bit stream response coming in from I/O_6 (normally MISO line during SPI operations) follows the ISO14443A standard.
- As the bits come in, they are timed (in 9.44uSec increments) and decoded, as Sequences D, E or F, then decrypted, all by the MCU.
- At the start we are measuring a Sequence D (logic 1) – but in this case it is also the tag response SOF.



Using the GUI to do Encrypted Write Operations

- Here we are using the GUI to do encrypted read of Block 0 in Sector 1.
- Using Host Command Code 0xC800 + Data
- Below is entire sequence, zoomed out
- To the right is the GUI interface.



UID: 4D897DC67F

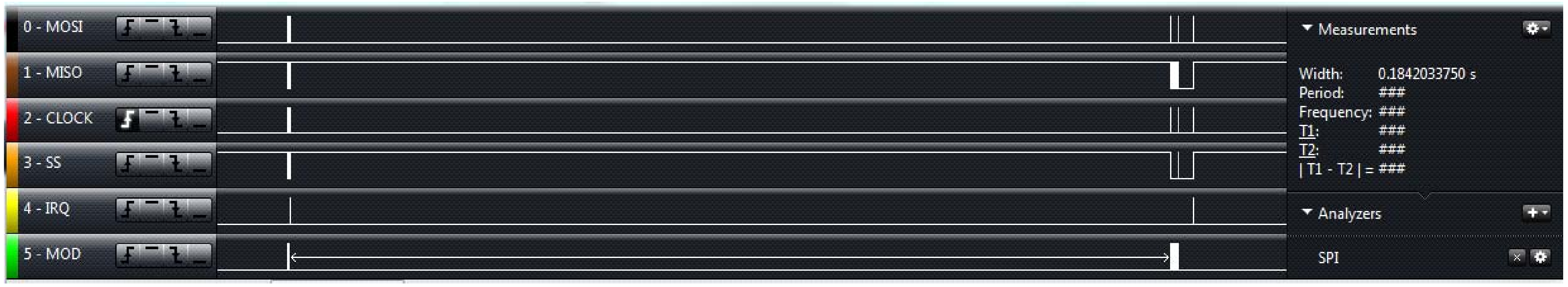
Key: FF FF FF FF FF FF

Sector: 1

Block: 0

Data: 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF

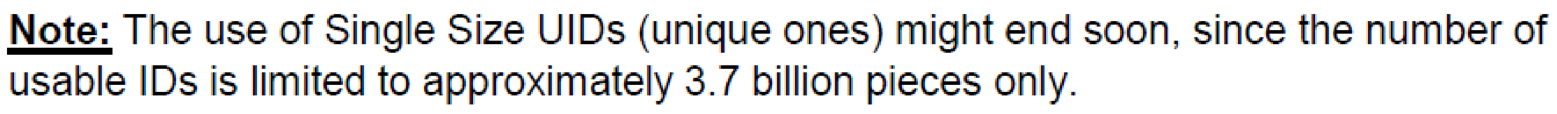
```
Encrypted request mode.
[][0A]
19:00:38.266        --> 0118000304C800112233445566778899AABBCCDDEEFF0000
19:00:38.446        <-- 0118000304C800112233445566778899AABBCCDDEEFF0000
Encrypted request mode.
[][0A]
```



```
Width:      0.1842033750 s
Period:     ###
Frequency:  ###
T1:         ###
T2:         ###
| T1 - T2 | = ###
```

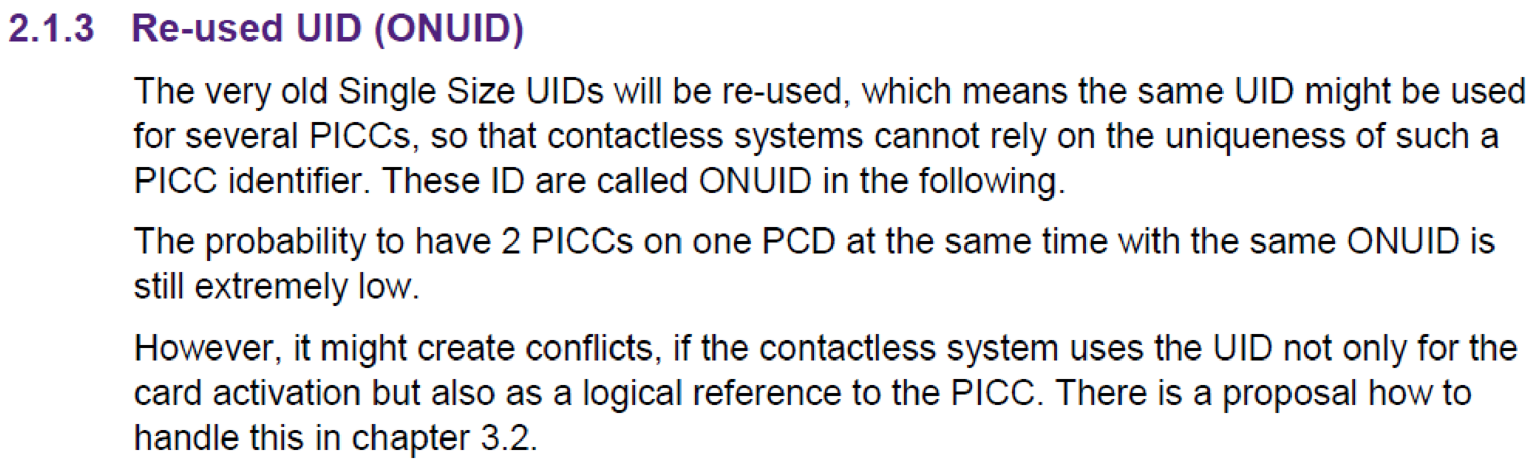
MIFARE Classic cards with 7 Byte UID’s

- NXP has nearly run out of unique 4 byte UID’s for their MIFARE Classic tags. This has caused them to recently release MIFARE Classic tags with Double size (7 Byte) UID’s.
  - Taken out of one of NXP’s App notes:



Note: The use of Single Size UIDs (unique ones) might end soon, since the number of usable IDs is limited to approximately 3.7 billion pieces only.

- It is very rare that you would ever encounter 4 byte tags with the same UID, but it is possible. NXP renamed the UID for this cards.



2.1.3 Re-used UID (ONUID)

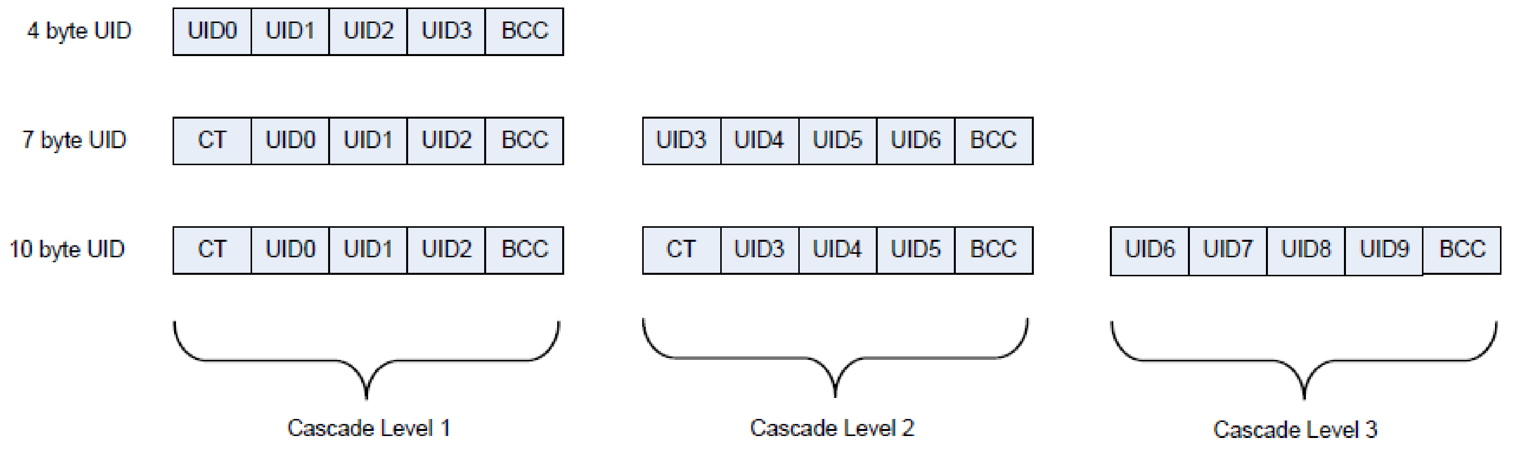
The very old Single Size UIDs will be re-used, which means the same UID might be used for several PICCs, so that contactless systems cannot rely on the uniqueness of such a PICC identifier. These ID are called ONUID in the following.

The probability to have 2 PICCs on one PCD at the same time with the same ONUID is still extremely low.

However, it might create conflicts, if the contactless system uses the UID not only for the card activation but also as a logical reference to the PICC. There is a proposal how to handle this in chapter 3.2.

Handling Double Size UID

- Here is the break down of the PICC's response during anti-collision.
- We'll need to understand this, to understand how to handle this in code.

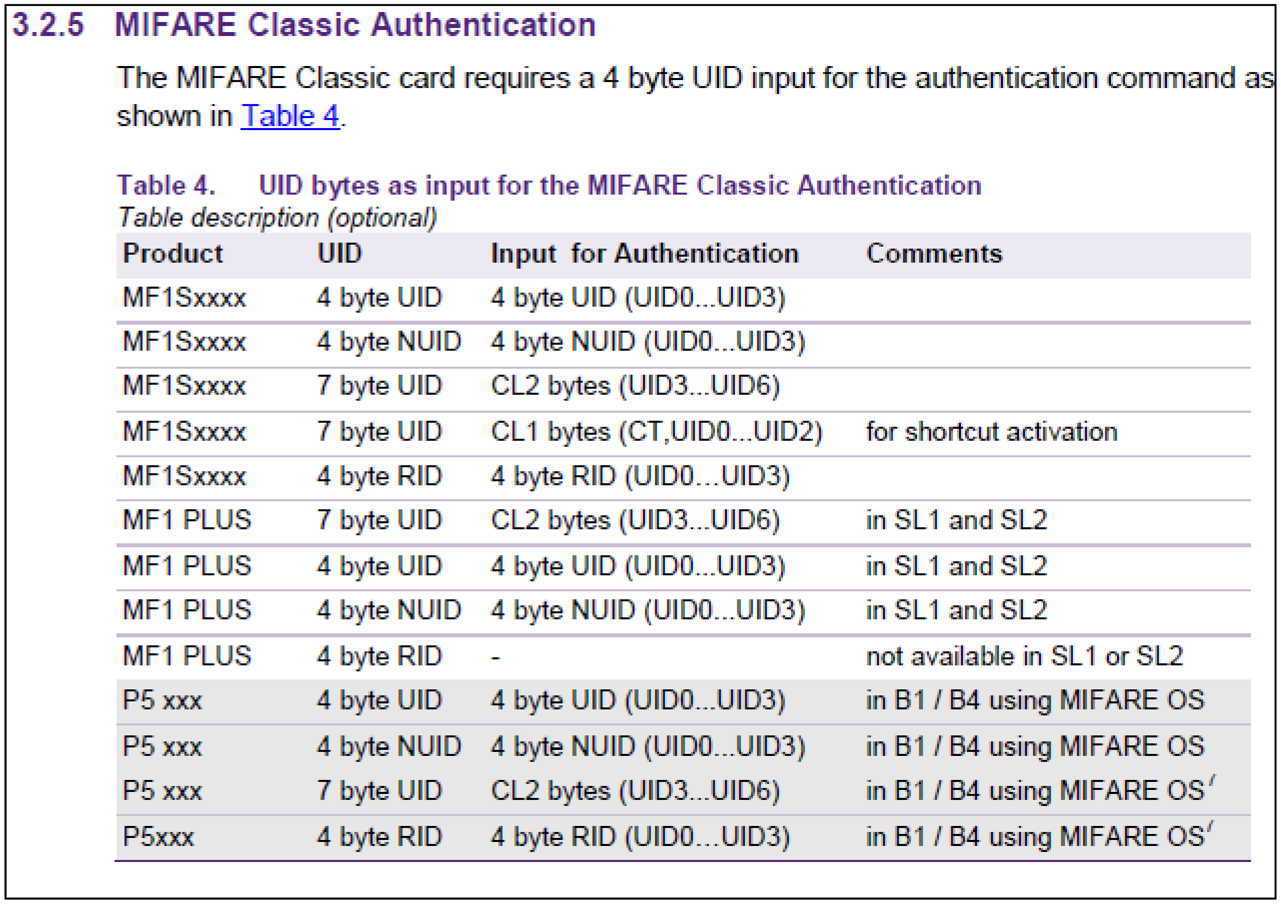


(1) BCC = Block Check Character, it is calculated as exclusive-or over the 4 previous bytes.
(2) CT = Cascade Tag, to indicate a following cascade level.

Fig 1. UIDs according to ISO/IEC 14443

Handling Double Size UID

- The Authentication procedure requires a 4 byte UID input.
- Table 4 defines which 4 bytes are used for 7 byte UID cards.



3.2.5 MIFARE Classic Authentication

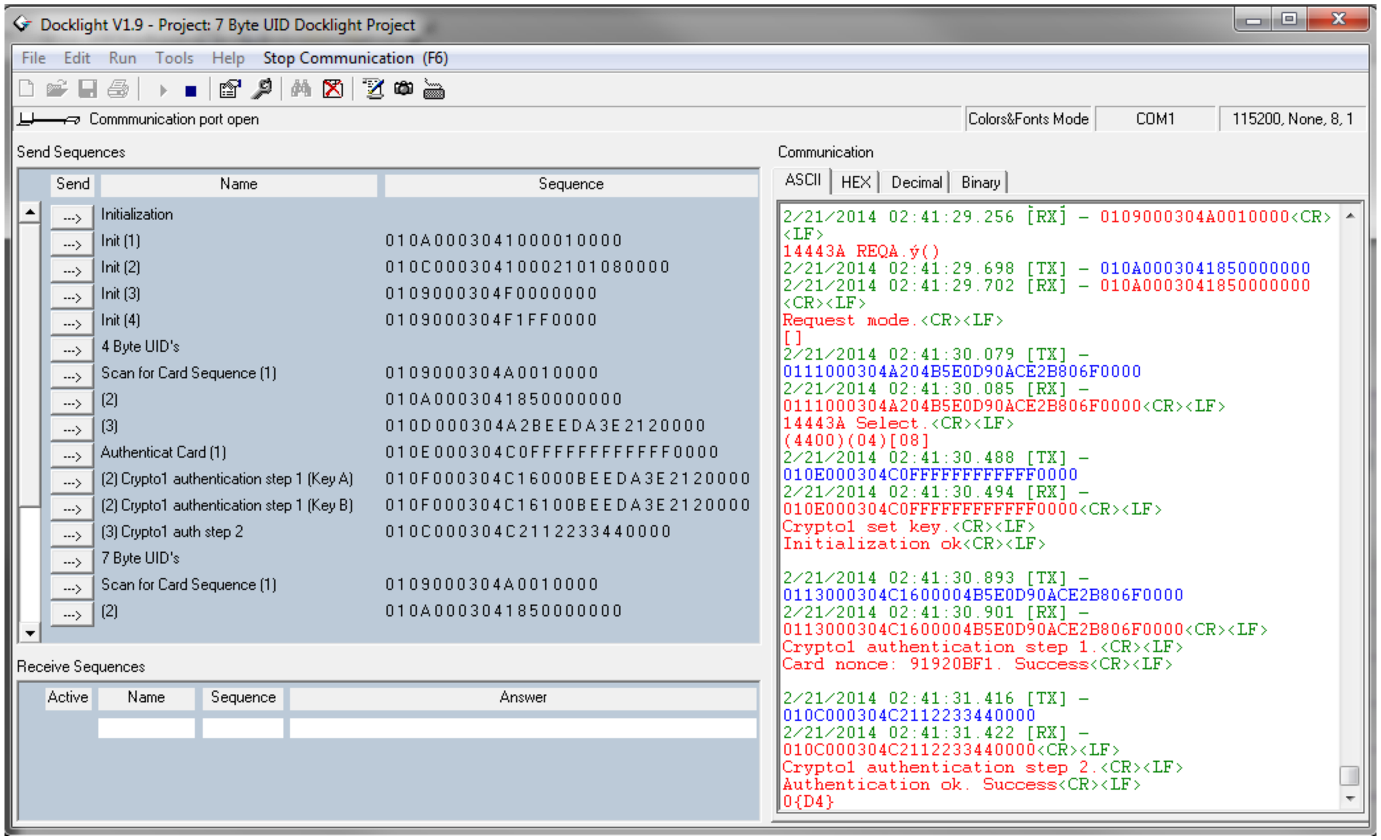
The MIFARE Classic card requires a 4 byte UID input for the authentication command as shown in Table 4.

Table 4. UID bytes as input for the MIFARE Classic Authentication
Table description (optional)

Product	UID	Input for Authentication	Comments
MF1Sxxxx	4 byte UID	4 byte UID (UID0...UID3)	
MF1Sxxxx	4 byte NUID	4 byte NUID (UID0...UID3)	
MF1Sxxxx	7 byte UID	CL2 bytes (UID3...UID6)	
MF1Sxxxx	7 byte UID	CL1 bytes (CT,UID0...UID2)	for shortcut activation
MF1Sxxxx	4 byte RID	4 byte RID (UID0...UID3)	
MF1 PLUS	7 byte UID	CL2 bytes (UID3...UID6)	in SL1 and SL2
MF1 PLUS	4 byte UID	4 byte UID (UID0...UID3)	in SL1 and SL2
MF1 PLUS	4 byte NUID	4 byte NUID (UID0...UID3)	in SL1 and SL2
MF1 PLUS	4 byte RID	-	not available in SL1 or SL2
P5 xxx	4 byte UID	4 byte UID (UID0...UID3)	in B1 / B4 using MIFARE OS
P5 xxx	4 byte NUID	4 byte NUID (UID0...UID3)	in B1 / B4 using MIFARE OS
P5 xxx	7 byte UID	CL2 bytes (UID3...UID6)	in B1 / B4 using MIFARE OS′
P5xxx	4 byte RID	4 byte RID (UID0...UID3)	in B1 / B4 using MIFARE OS′

MIFARE Classic 7 byte UID Demo

- 7 byte UID's are not supporting in TI's Mifare demo GUI.
- We currently have a Docklight project used for evaluation purposes.



Send Sequences

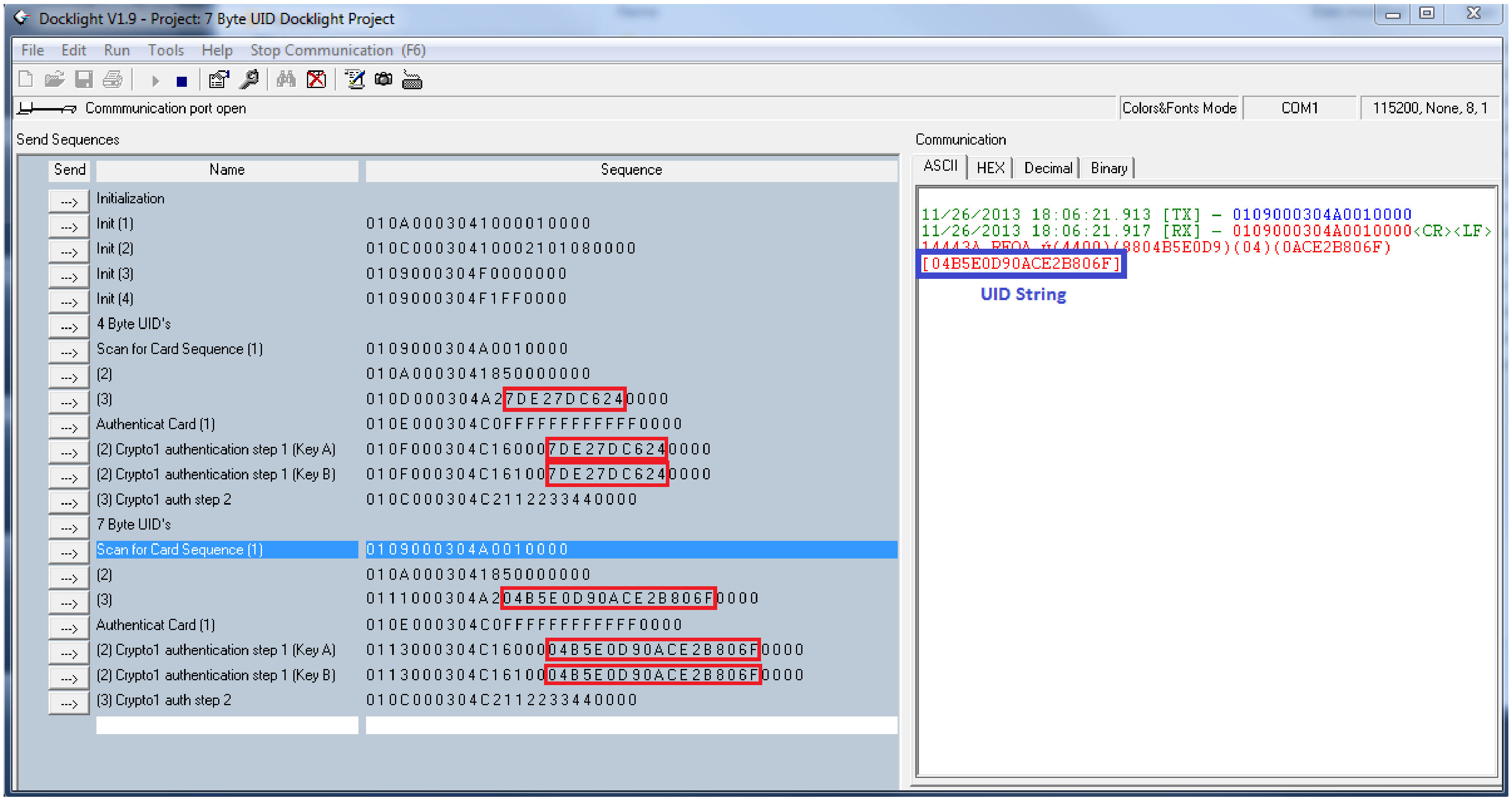
Send	Name	Sequence
--->	Initialization	
--->	Init (1)	010A0003041000010000
--->	Init (2)	010C00030410002101080000
--->	Init (3)	0109000304F0000000
--->	Init (4)	0109000304F1FF0000
--->	4 Byte UID's	
--->	Scan for Card Sequence (1)	0109000304A0010000
--->	(2)	010A0003041850000000
--->	(3)	010D000304A2BEEDA3E2120000
--->	Authenticat Card (1)	010E000304C0FFFFFFFFFFFF0000
--->	(2) Crypto1 authentication step 1 (Key A)	010F000304C16000BEEDA3E2120000
--->	(2) Crypto1 authentication step 1 (Key B)	010F000304C16100BEEDA3E2120000
--->	(3) Crypto1 auth step 2	010C000304C2112233440000
--->	7 Byte UID's	
--->	Scan for Card Sequence (1)	0109000304A0010000
--->	(2)	010A0003041850000000

Receive Sequences

Active	Name	Sequence	Answer

```
2/21/2014 02:41:29.256 [RX] - 0109000304A0010000<CR>
<LF>
14443A REQA.ý()
2/21/2014 02:41:29.698 [TX] - 010A0003041850000000
2/21/2014 02:41:29.702 [RX] - 010A0003041850000000
<CR><LF>
Request mode.<CR><LF>
[]
2/21/2014 02:41:30.079 [TX] -
0111000304A204B5E0D90ACE2B806F0000
2/21/2014 02:41:30.085 [RX] -
0111000304A204B5E0D90ACE2B806F0000<CR><LF>
14443A Select.<CR><LF>
(4400)(04)[08]
2/21/2014 02:41:30.488 [TX] -
010E000304C0FFFFFFFFFFFF0000
2/21/2014 02:41:30.494 [RX] -
010E000304C0FFFFFFFFFFFF0000<CR><LF>
Crypto1 set key.<CR><LF>
Initialization ok<CR><LF>

2/21/2014 02:41:30.893 [TX] -
0113000304C1600004B5E0D90ACE2B806F0000
2/21/2014 02:41:30.901 [RX] -
0113000304C1600004B5E0D90ACE2B806F0000<CR><LF>
Crypto1 authentication step 1.<CR><LF>
Card nonce: 91920BF1. Success<CR><LF>

2/21/2014 02:41:31.416 [TX] -
010C000304C2112233440000
2/21/2014 02:41:31.422 [RX] -
010C000304C2112233440000<CR><LF>
Crypto1 authentication step 2.<CR><LF>
Authentication ok. Success<CR><LF>
0{D4}
```

Copy over UID from Scan for Card Response



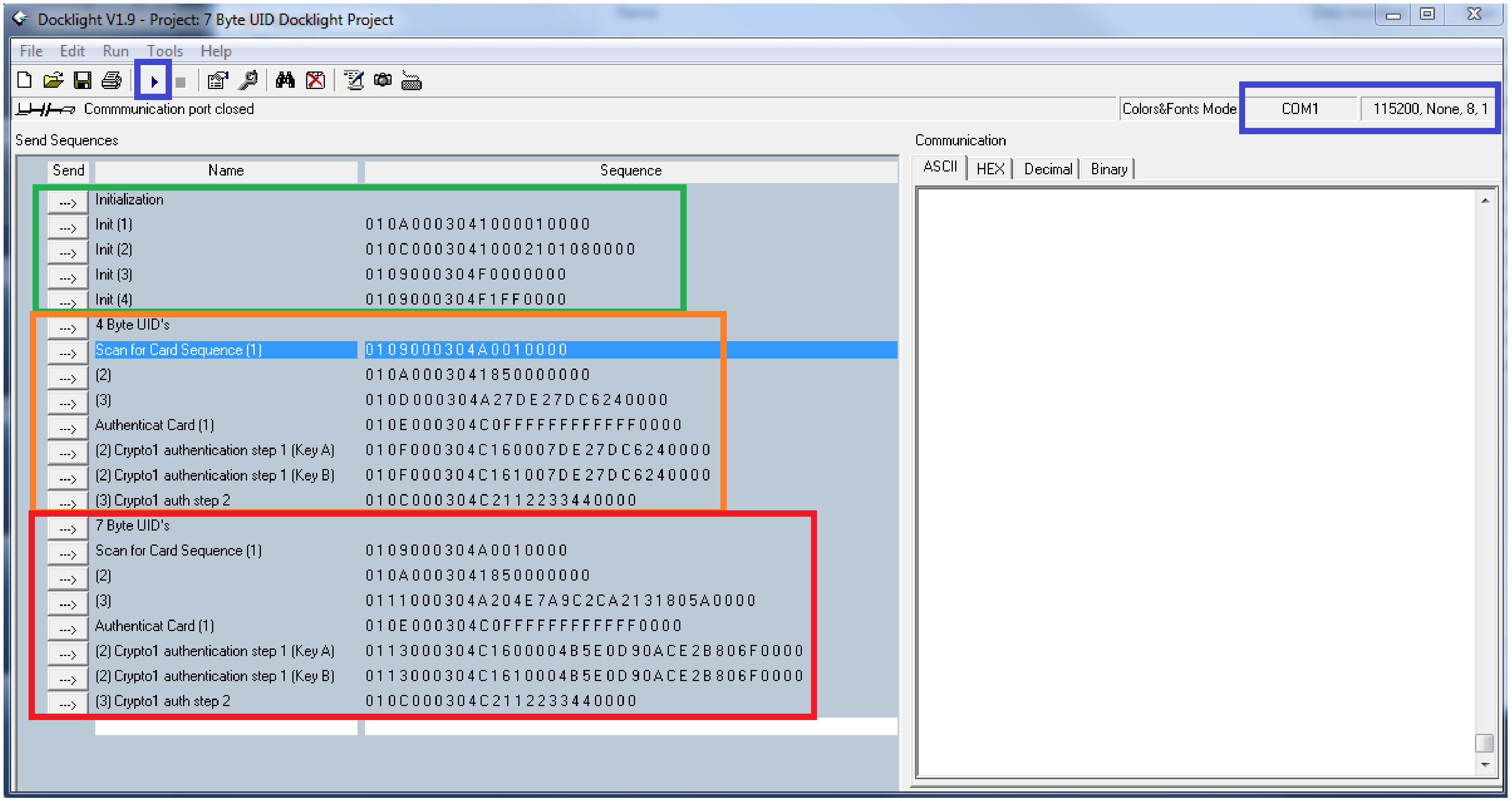
Docklight V1.9 - Project: 7 Byte UID Docklight Project

Send	Name	Sequence
--->	Initialization	
--->	Init (1)	010A0003041000010000
--->	Init (2)	010C00030410002101080000
--->	Init (3)	0109000304F0000000
--->	Init (4)	0109000304F1FF0000
--->	4 Byte UID's	
--->	Scan for Card Sequence (1)	0109000304A0010000
--->	(2)	010A00030418500000000
--->	(3)	010D000304A27DE27DC6240000
--->	Authenticat Card (1)	010E000304C0FFFFFFFFFFFF0000
--->	(2) Crypto1 authentication step 1 (Key A)	010F000304C16000 7DE27DC6240000
--->	(2) Crypto1 authentication step 1 (Key B)	010F000304C16100 7DE27DC6240000
--->	(3) Crypto1 auth step 2	010C000304C2112233440000
--->	7 Byte UID's	
--->	Scan for Card Sequence (1)	0109000304A0010000
--->	(2)	010A00030418500000000
--->	(3)	0111000304A204B5E0D90ACE2B806F0000
--->	Authenticat Card (1)	010E000304C0FFFFFFFFFFFF0000
--->	(2) Crypto1 authentication step 1 (Key A)	0113000304C1600004B5E0D90ACE2B806F0000
--->	(2) Crypto1 authentication step 1 (Key B)	0113000304C1610004B5E0D90ACE2B806F0000
--->	(3) Crypto1 auth step 2	010C000304C2112233440000

```
11/26/2013 18:06:21.913 [TX] - 0109000304A0010000
11/26/2013 18:06:21.917 [RX] - 0109000304A0010000<CR><LF>
14443A REQA ú(4400)(8804B5E0D9)(04)(0ACE2B806F)
[04B5E0D90ACE2B806F]
```

UID String

Connect to COM port and step through commands.



Commmunication port closed

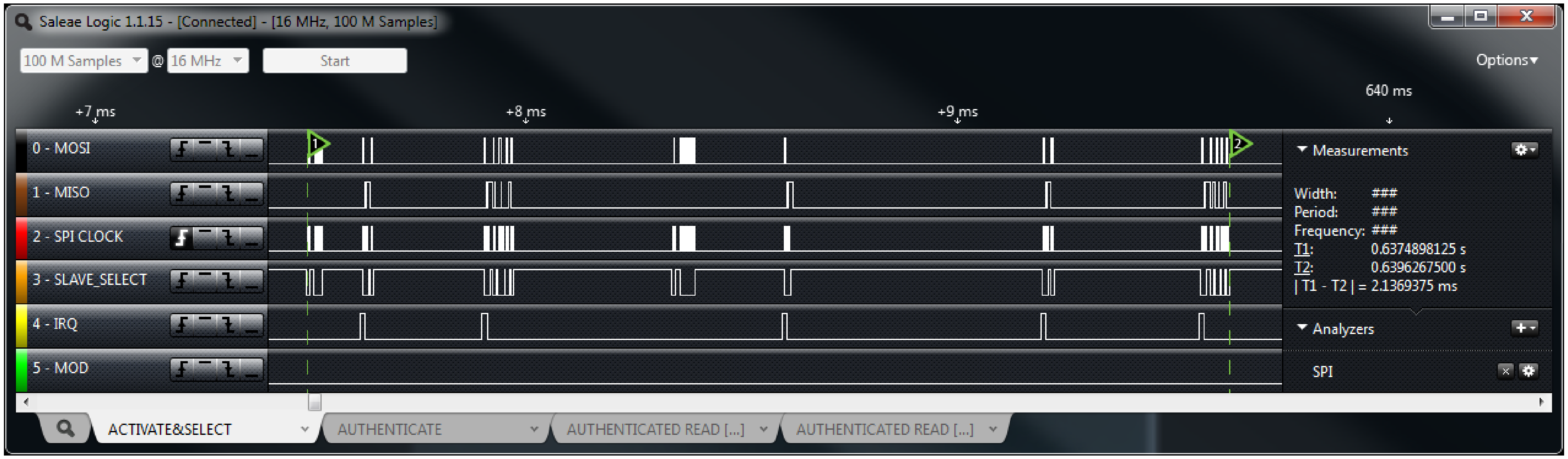
COM1 | 115200, None, 8, 1

Send Sequences

Send	Name	Sequence
--->	Initialization	
--->	Init (1)	0 1 0 A 0 0 0 3 0 4 1 0 0 0 0 1 0 0 0 0
--->	Init (2)	0 1 0 C 0 0 0 3 0 4 1 0 0 0 2 1 0 1 0 8 0 0 0 0
--->	Init (3)	0 1 0 9 0 0 0 3 0 4 F 0 0 0 0 0 0 0
--->	Init (4)	0 1 0 9 0 0 0 3 0 4 F 1 F F 0 0 0 0
--->	4 Byte UID's	
--->	Scan for Card Sequence (1)	0 1 0 9 0 0 0 3 0 4 A 0 0 1 0 0 0 0
--->	(2)	0 1 0 A 0 0 0 3 0 4 1 8 5 0 0 0 0 0 0 0
--->	(3)	0 1 0 D 0 0 0 3 0 4 A 2 7 D E 2 7 D C 6 2 4 0 0 0 0
--->	Authenticat Card (1)	0 1 0 E 0 0 0 3 0 4 C 0 F F F F F F F F F F F F 0 0 0 0
--->	(2) Crypto1 authentication step 1 (Key A)	0 1 0 F 0 0 0 3 0 4 C 1 6 0 0 0 7 D E 2 7 D C 6 2 4 0 0 0 0
--->	(2) Crypto1 authentication step 1 (Key B)	0 1 0 F 0 0 0 3 0 4 C 1 6 1 0 0 7 D E 2 7 D C 6 2 4 0 0 0 0
--->	(3) Crypto1 auth step 2	0 1 0 C 0 0 0 3 0 4 C 2 1 1 2 2 3 3 4 4 0 0 0 0
--->	7 Byte UID's	
--->	Scan for Card Sequence (1)	0 1 0 9 0 0 0 3 0 4 A 0 0 1 0 0 0 0
--->	(2)	0 1 0 A 0 0 0 3 0 4 1 8 5 0 0 0 0 0 0 0
--->	(3)	0 1 1 1 0 0 0 3 0 4 A 2 0 4 E 7 A 9 C 2 C A 2 1 3 1 8 0 5 A 0 0 0 0
--->	Authenticat Card (1)	0 1 0 E 0 0 0 3 0 4 C 0 F F F F F F F F F F F F 0 0 0 0
--->	(2) Crypto1 authentication step 1 (Key A)	0 1 1 3 0 0 0 3 0 4 C 1 6 0 0 0 0 4 B 5 E 0 D 9 0 A C E 2 B 8 0 6 F 0 0 0 0
--->	(2) Crypto1 authentication step 1 (Key B)	0 1 1 3 0 0 0 3 0 4 C 1 6 1 0 0 0 4 B 5 E 0 D 9 0 A C E 2 B 8 0 6 F 0 0 0 0
--->	(3) Crypto1 auth step 2	0 1 0 C 0 0 0 3 0 4 C 2 1 1 2 2 3 3 4 4 0 0 0 0

Timing Observations with two different MCU platforms

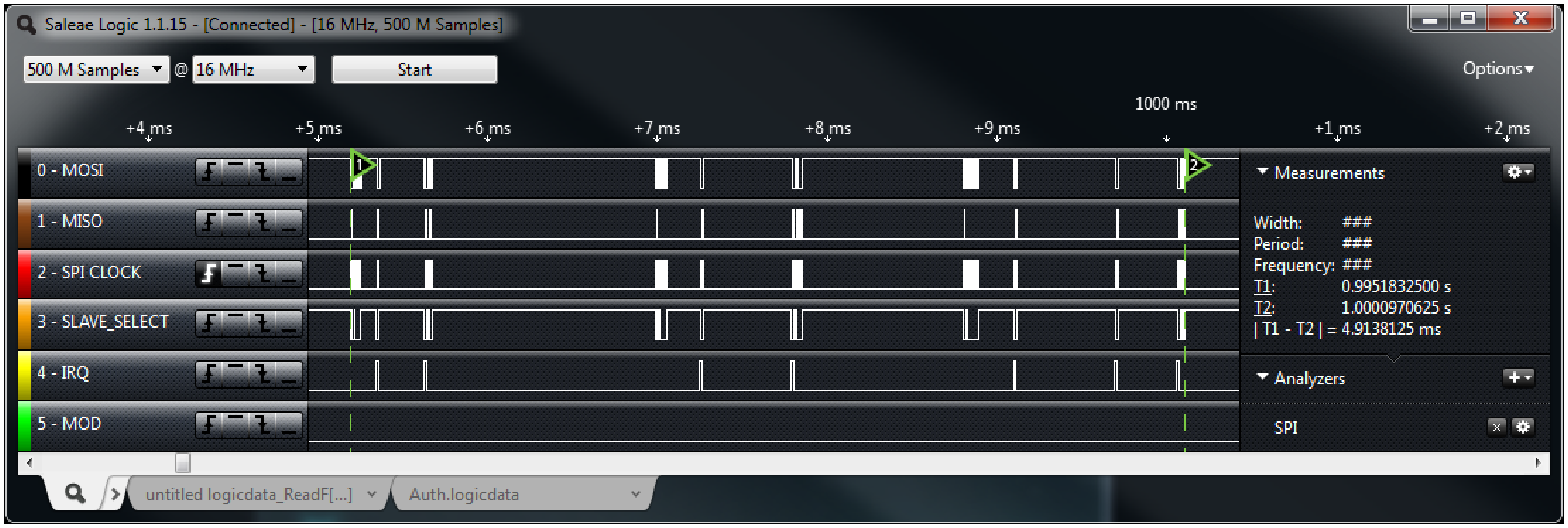
- There are four main card operations to study for timings
  - Activation & Selection of the card (this is according to ISO14443A)
  - Authentication of the Card
  - Authenticated Read of a Block
  - Authenticated Read of a Sector
- Below here we can see a logic shot of activation and selection of the card, according to ISO144443-2, 3.
- Marker 1 is set at when we write the ISO Control Register for 0x88, marker 2 is set for end of the reception of the SAK and the RSSI value of the card.
- Total time using TRF79xxA + MSP430 is measured here as 2.1mSec (slides 45 & 46 from NXP public data, shows they are specifying taking up to 3mSec for this operation == we are on par / slightly better here.



Width: ###
Period: ###
Frequency: ###
T1: 0.6374898125 s
T2: 0.6396267500 s
| T1 - T2 | = 2.1369375 ms

Timing Observations with two different MCU platforms (cont.)

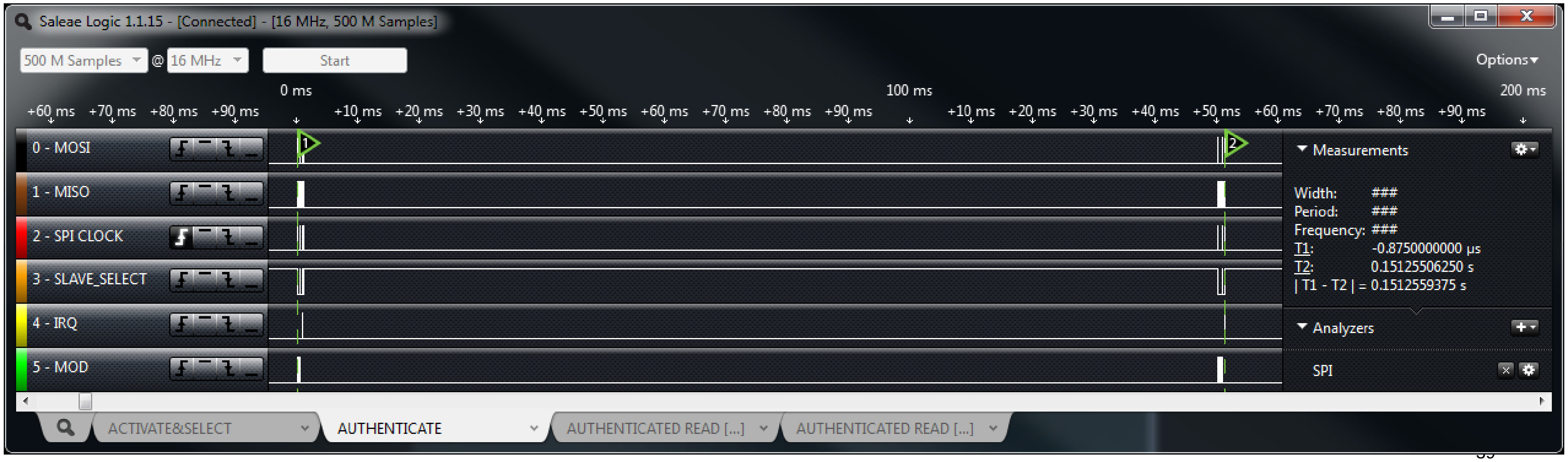
- Below here we can see a logic shot of activation and selection of the card, according to ISO144443-2, 3.
- Marker 1 is set at when we write the ISO Control Register for 0x88, marker 2 is set for end of the reception of the SAK and the RSSI value of the card.
- Total time using TRF79xxA + Cortex M3 is measured here as 4.91mSec (slides 45 & 46) from NXP public data, shows they are specifying taking up to 3mSec for this operation == we are on slower here and there is a 1.31mSec delay in between getting the ATQA and issuing the first anti-collision command which could be optimized.



Width: ###
Period: ###
Frequency: ###
T1: 0.9951832500 s
T2: 1.0000970625 s
| T1 - T2 | = 4.9138125 ms

Authentication of the Card Timing

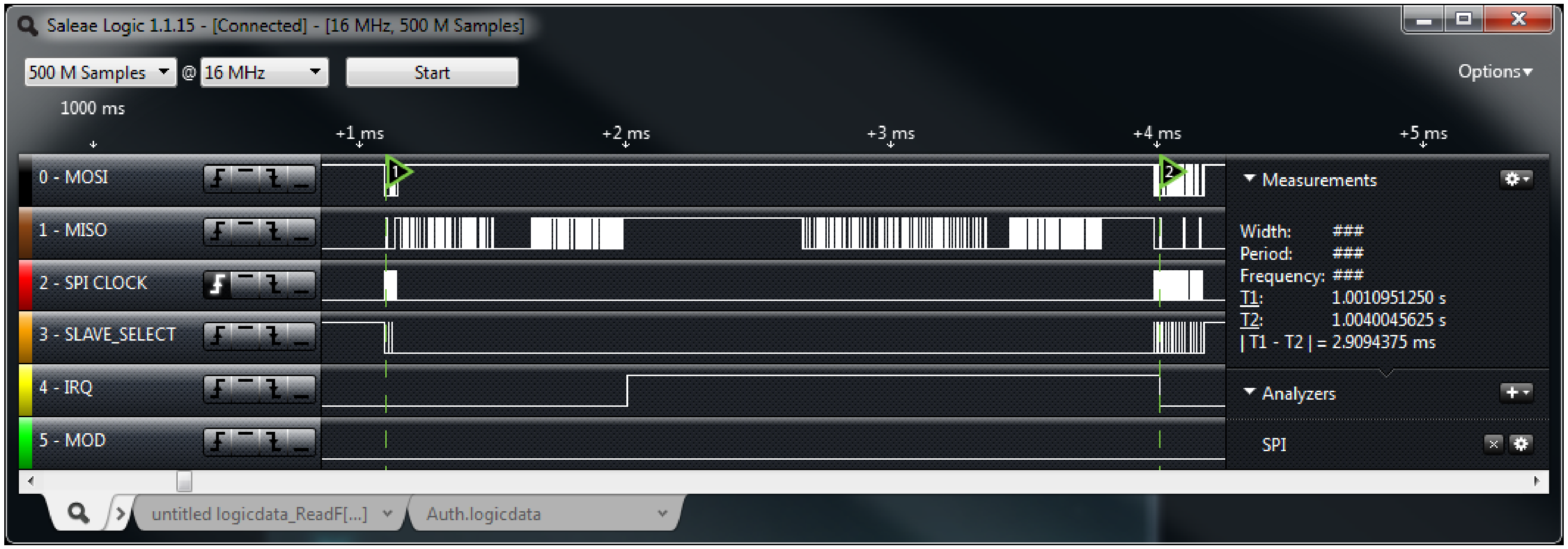
- Authenticating the card is the next step after activation and selection.
- Here below one can see a logic shot which is showing performing the authentication process using the MSP430.
- Marker 1 is set for when the TRF79xxA is placed into Direct Mode 0 to begin the non-ISO standard mode of operation that Mifare Classic requires and Marker 2 is set for the end of that process.
- Total time measured is 151.26mSec NXP specs 2mSec for this.
- This could possibly be optimized (see next slide).



Width: ###
Period: ###
Frequency: ###
T1: -0.8750000000 µs
T2: 0.15125506250 s
| T1 - T2 | = 0.1512559375 s

Authentication of the Card Timing

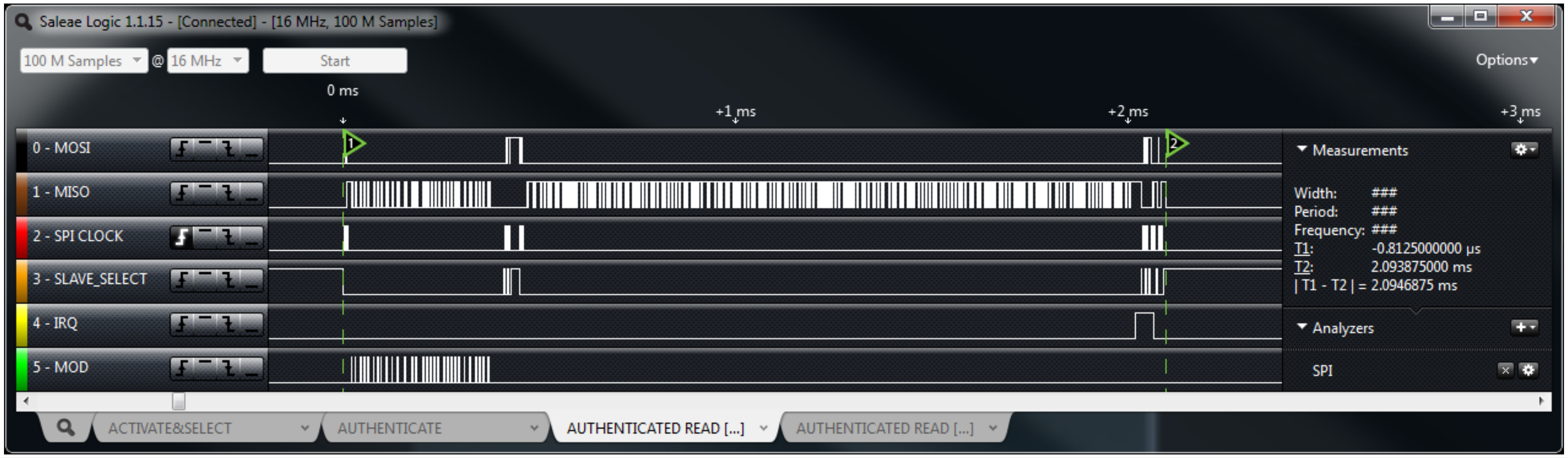
- Authenticating the card is the next step after activation and selection.
- Here below one can see a logic shot which is showing performing the authentication process using the Cortex M.
- Marker 1 is set for when the TRF79xxA is placed into Direct Mode 0 to begin the non-ISO standard mode of operation that Mifare Classic requires and Marker 2 is set for the end of that process.
- Total time measured is 2.9mSec NXP specs 2mSec for this.
- This is doing in firmware close to what NXP is doing in hardware.



```
Width:      ###
Period:     ###
Frequency:  ###
T1:         1.0010951250 s
T2:         1.0040045625 s
| T1 - T2 | = 2.9094375 ms
```

Authenticated Read Block Timing

- Authenticated read of a block is a common next step in the process, after activation/selection & authentication.
- Here below one can see a logic shot which is showing performing the authenticated read of a block, using current MSP430 code example.
- Marker 1 is set for when the TRF79xxA is placed into Direct Mode 0 to begin the non-ISO standard mode of operation that Mifare Classic requires and Marker 2 is set for the end of that process.
- Total time measured is 2.09mSec, NXP specs 2.5mSec for this.



Saleae Logic 1.1.15 - [Connected] - [16 MHz, 100 M Samples]

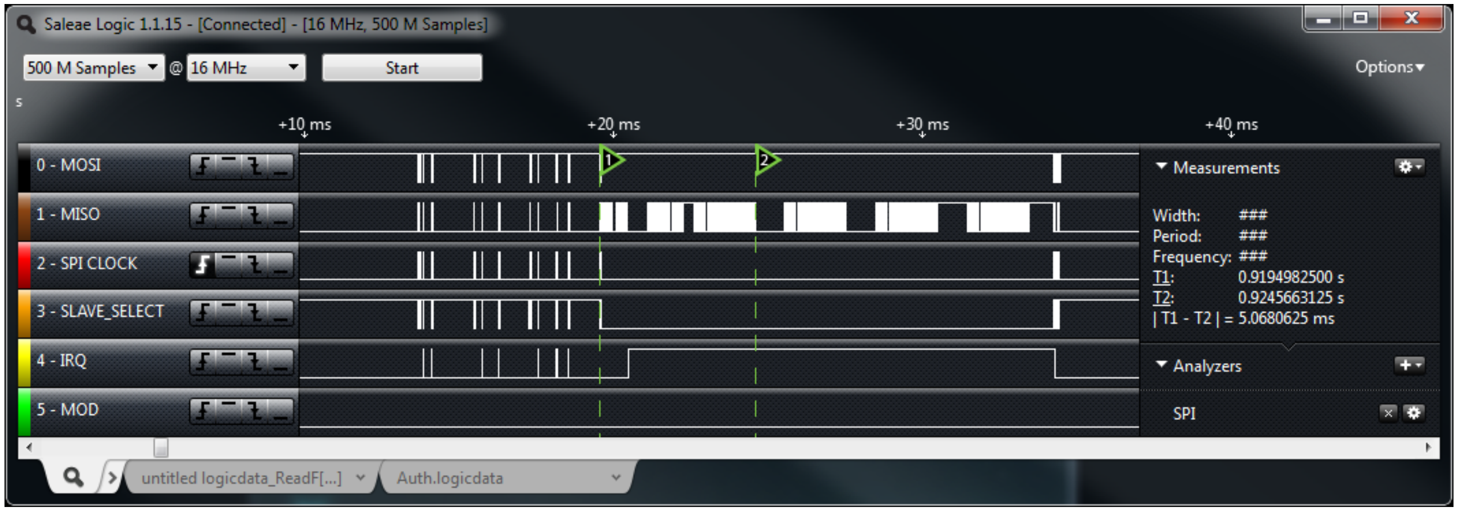
100 M Samples @ 16 MHz

```
Width:       ###
Period:      ###
Frequency:   ###
T1:          -0.8125000000 µs
T2:          2.093875000 ms
| T1 - T2 | = 2.0946875 ms
```

Texas Instruments

Authenticated Read Block Timing

- Authenticated read of a block is a common next step in the process, after activation/selection & authentication.
- Here below one can see a logic shot which is showing performing the authenticated read of a block using current Cortex M code example. .
- Marker 1 is set for when the TRF79xxA is placed into Direct Mode 0 to begin the non-ISO standard mode of operation that Mifare Classic requires and Marker 2 is set for the end of that process.
- Total time measured is 5.07mSec, NXP specs 2.5mSec for this.

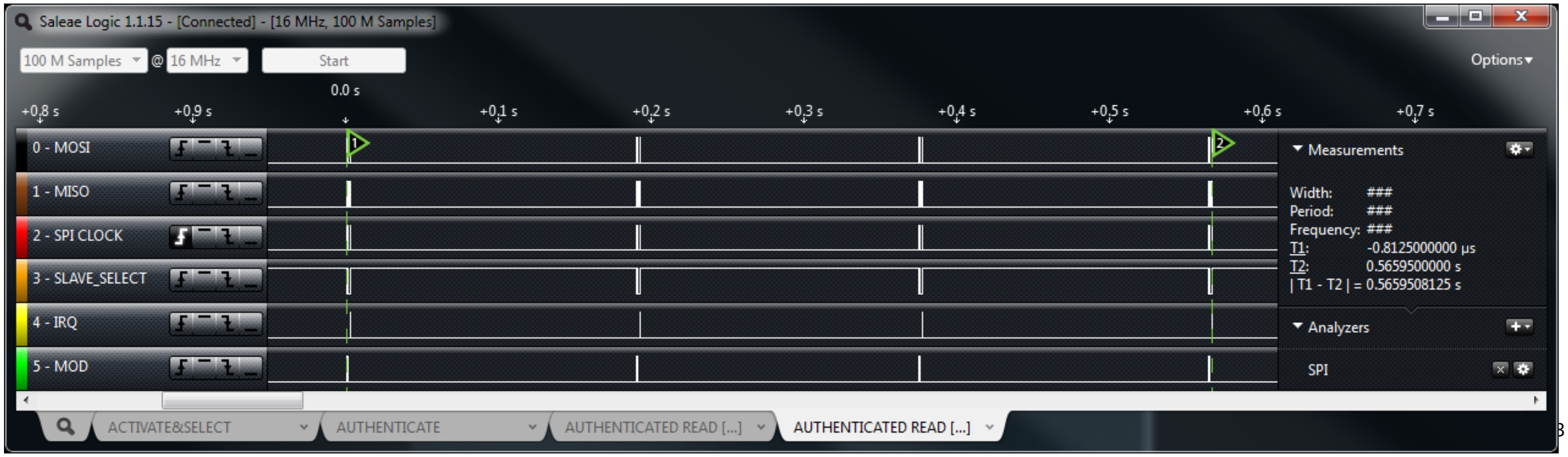


Width: ###
Period: ###
Frequency: ###
T1: 0.9194982500 s
T2: 0.9245663125 s
| T1 - T2 | = 5.0680625 ms

Texas Instruments

Authenticated Read Sector Timing

- Authenticated read of a sector (multiple blocks) is another common next step in the process, after activation/selection & authentication.
- Here below one can see a logic shot which is showing performing the authenticated read of a sector (four blocks), using current MSP430 code example.
- Marker 1 is set for when the TRF79xxA is placed into Direct Mode 0 to begin the non-ISO standard mode of operation that Mifare Classic requires and Marker 2 is set for the end of that process.
- Total time measured is 569mSec, with 187mSec between what are essentially four authenticated read blocks – this interval time could be shortened. NXP has not any public benchmark on this process.


Saleae Logic 1.1.15 - [Connected] - [16 MHz, 100 M Samples]

100 M Samples @ 16 MHz

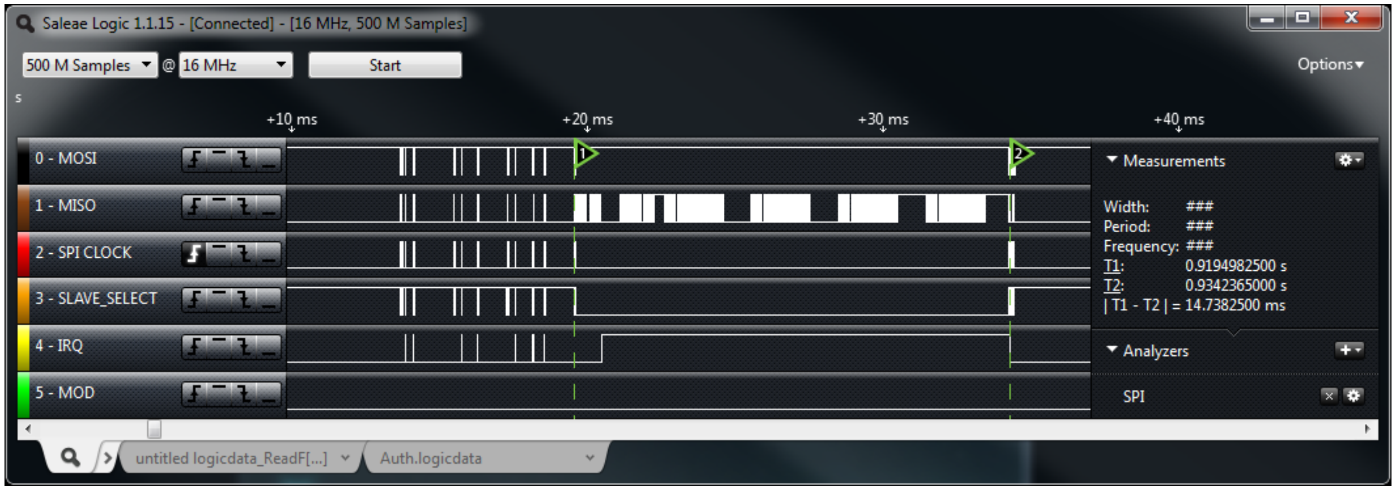
Measurements

```
Width:     ###
Period:    ###
Frequency: ###
T1:        -0.8125000000 µs
T2:        0.5659500000 s
| T1 - T2 | = 0.5659508125 s
```

TEXAS INSTRUMENTS

Authenticated Read Sector Timing

- Authenticated read of a sector (multiple blocks) is another common next step in the process, after activation/selection & authentication.
- Here below one can see a logic shot which is showing performing the authenticated read of a sector (four blocks), using current Cortex M code example. .
- Marker 1 is set for when the TRF79xxA is placed into Direct Mode 0 to begin the non-ISO standard mode of operation that Mifare Classic requires and Marker 2 is set for the end of that process.
- Total time measured is 14mSec, with ~1mSec between what are essentially four authenticated read blocks – NXP has not any public benchmark on this process.


Saleae Logic 1.1.15 - [Connected] - [16 MHz, 500 M Samples]

500 M Samples @ 16 MHz

```
Width:      ###
Period:     ###
Frequency:  ###
T1:         0.9194982500 s
T2:         0.9342365000 s
| T1 - T2 | = 14.7382500 ms
```

TEXAS INSTRUMENTS

Conclusion/Comparison on the timing with MSP430

- TRF79xxA Operations Timing

Process	Time
Activation and Selection	2.1mSec
Authentication	151mSec
Authenticated Read Block	2.09mSec
Total Time	155.19mSec

- NXP Public Spec Timing

Process	Time
Activation and Selection	3mSec
Authentication	2mSec
Authenticated Read Block	2.5mSec
Total Time	7.5mSec

- Conclusion = Current Texas Instruments Mifare Classic with MSP430 solution is slower, but still should not impact any user experience, and could most likely be optimized further.

Conclusion/Comparison on the timing with Cortex M

- TRF79xxA Operations Timing

Process	Time
Activation and Selection	4.91mSec
Authentication	2.9mSec
Authenticated Read Block	5.07mSec
Total Time	12.88mSec

- NXP Public Spec Timing

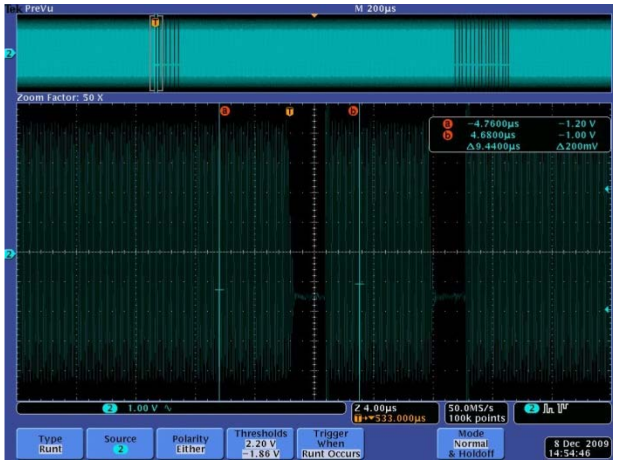
Process	Time
Activation and Selection	3mSec
Authentication	2mSec
Authenticated Read Block	2.5mSec
Total Time	7.5mSec

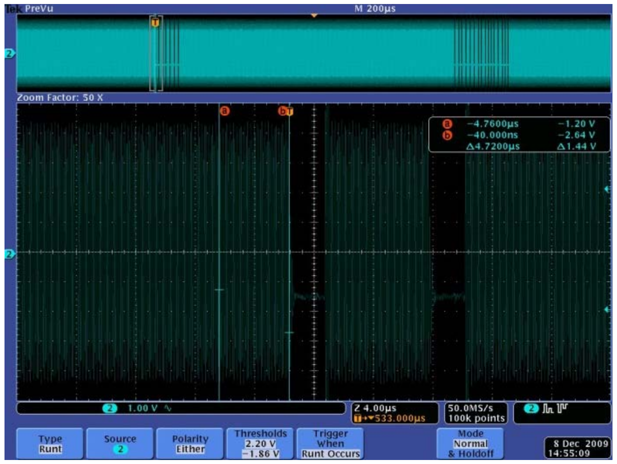
- Conclusion = Current Texas Instruments Mifare Classic with Cortex M3 solution is **slighty slower than what NXP is recommending**, but should not impact any user experience!

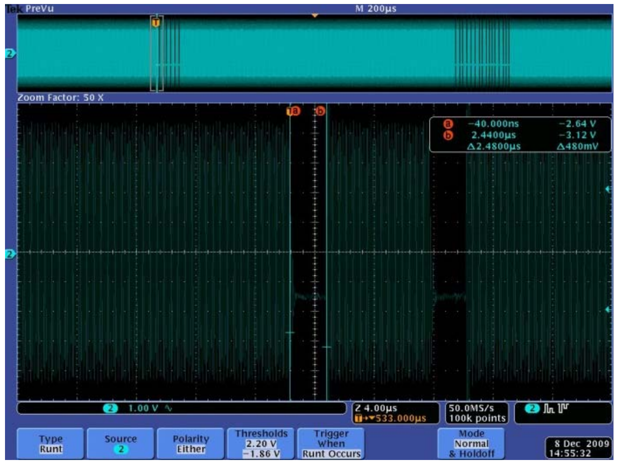
Backup

EVM Screen Captures

- These captures illustrate sequence X, as taken from the TRF7960EVM





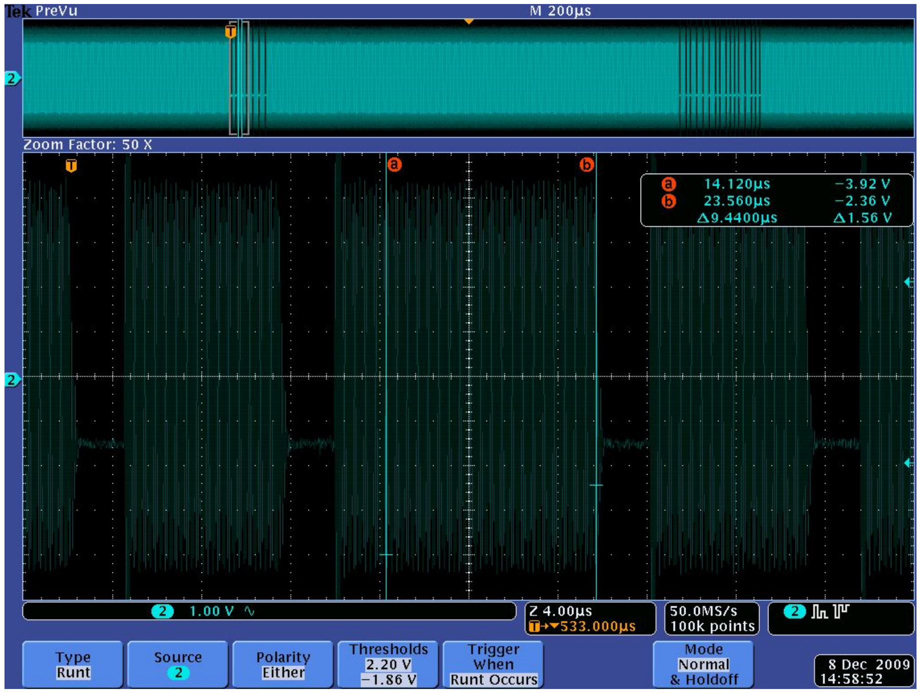


t_b = 9.44uSec t_x = 4.72uSec t_1 = 2.48uSec

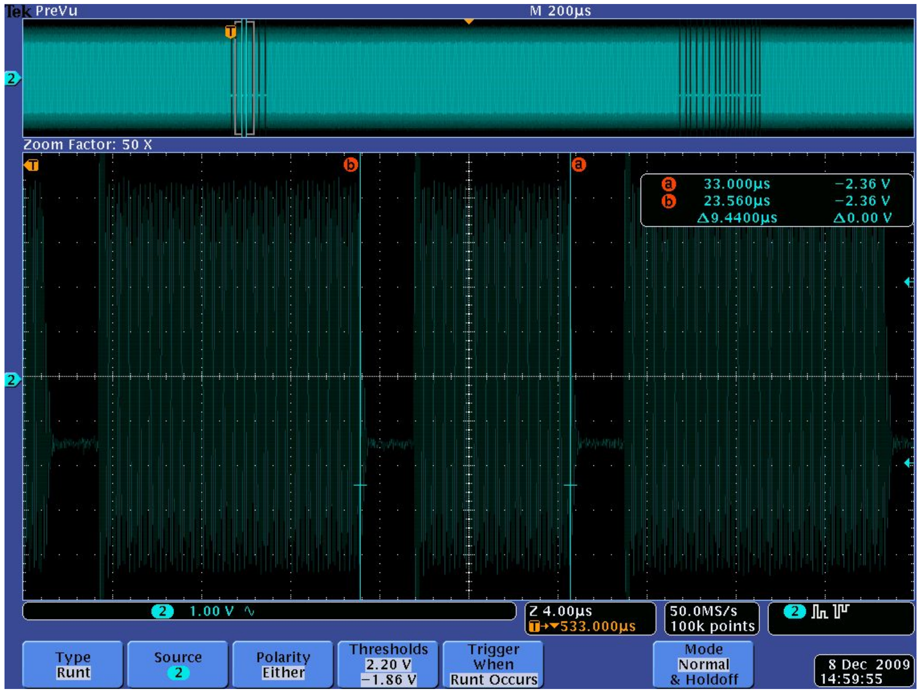
TEXAS INSTRUMENTS

EVM Screen Captures

- These captures illustrate Sequences Y and Z, as taken from the TRF7960EVM

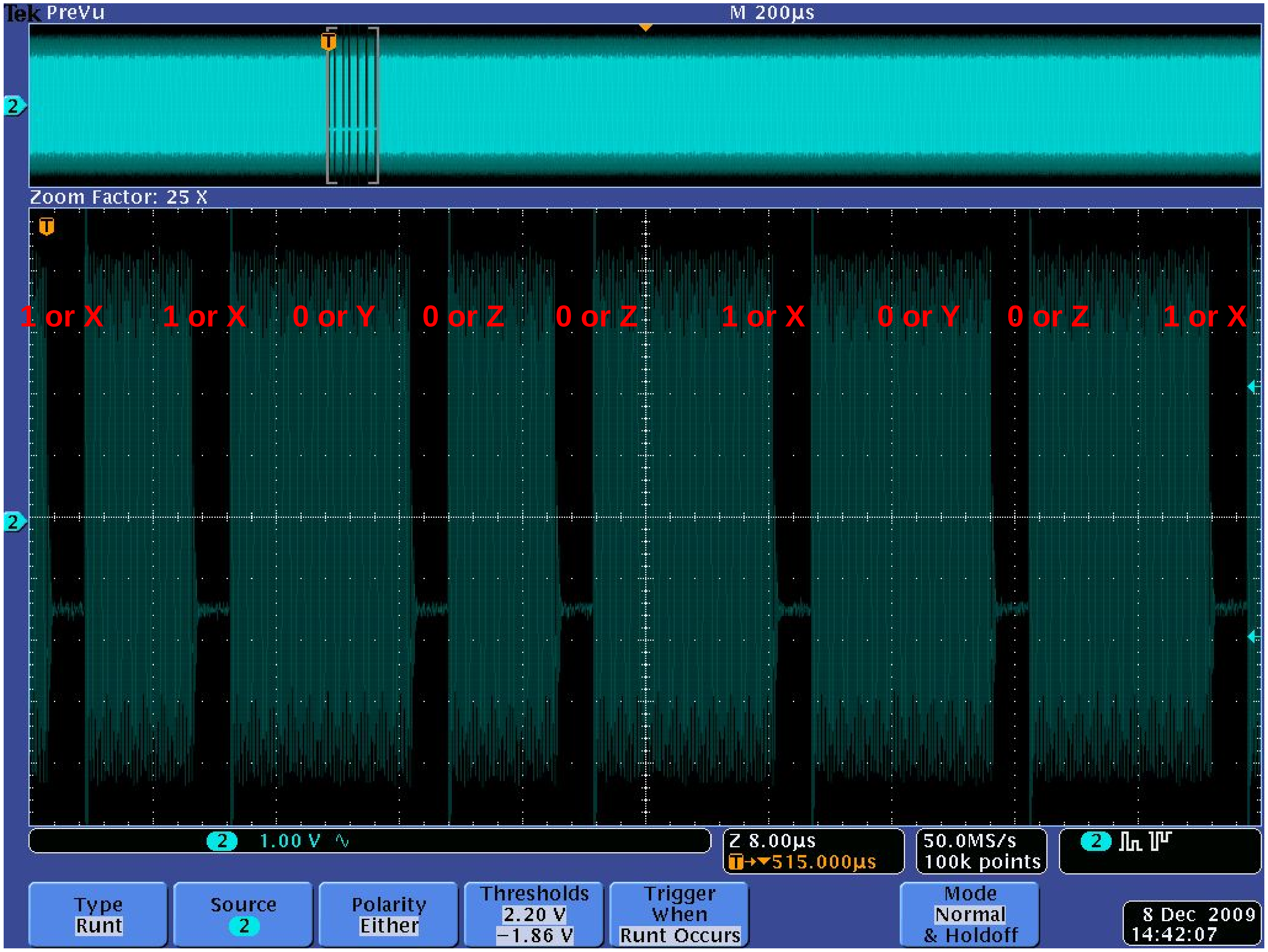


Sequence Y = Carrier for 9.44uSec



Sequence Z = Pause for 2uSec-3uSec, Carrier for Remainder of 9.44uSec

EVM Screen Capture Decoded



Data from GUI for Mifare Authentication

- 19:45:04.452 --> 010E000304C0FFFFFFFFFFFF0000
- 19:45:04.749 <-- 010E000304C0FFFFFFFFFFFF0000
- Crypto1 set key.
- Initialization ok

- 19:45:04.812 --> 010F000304C160007EA6A6E29C0000
- 19:45:05.015 <-- 010F000304C160007EA6A6E29C0000
- Crypto1 authentication step 1.
- Card nonce: D06530F3. Success

- 19:45:05.077 --> 010C000304C2112233440000
- 19:45:05.312 <-- 010C000304C2112233440000
- Crypto1 authentication step 2.
- Authentication ok. Success