NFC Forum Type 3 Tag Platform Operations with the TRF7970A

NFC/RFID Training Module (2014)

S2 MCU NFC/RFID Applications Team

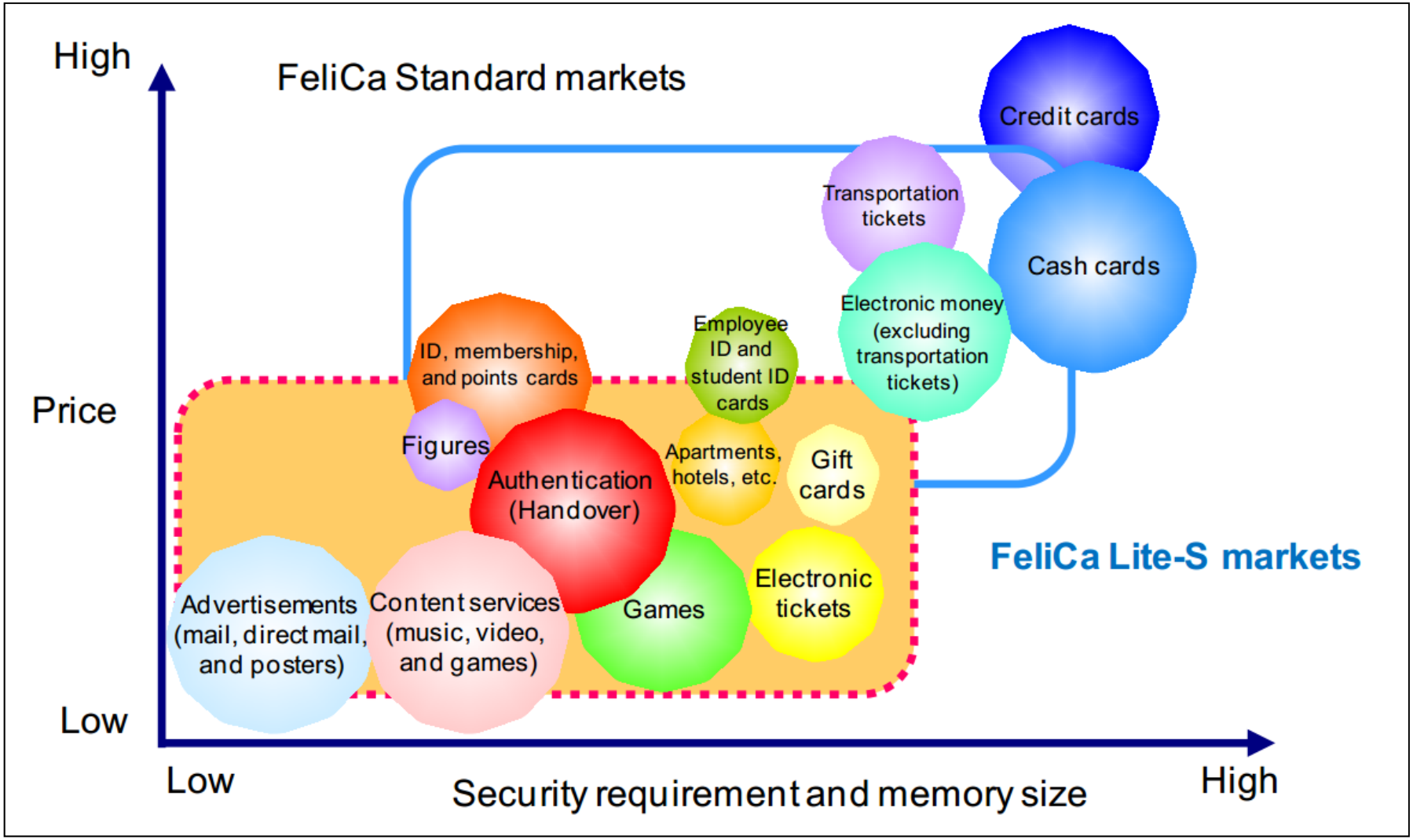
AGENDA

- **Brief overview of FeliCa**
  - What is FeliCa?
  - What markets does it serve?
- **FeliCa Operational Details**
  - Command Set
  - Anti-Collision
  - Expected Responses
- **NFC Forum Type 3 Tag Operation Specification Overview**
  - Introduction to T3T Platform
  - State Diagram
  - NDEF Access
  - NDEF Detection
  - Attribute Block
  - NDEF message retrieval
  - NDEF Formatting and Writing NDEF Message
- **Using TRF7964A / -70A with FeliCa / NFC T3T Platform**
  - Configuring the TRF7964A / TRF7970A for FeliCa / T3T Platform Operations
  - Command Issuance / Tag Response examples
  - Overview of MSP430G2553 LaunchPad / TRF7970A BoosterPack Code
  - Lab
    - MSP430G2553 LaunchPad + TRF7970A BoosterPack + T3T Platform

What is FeliCa / NFC T3T Platform?

- FeliCa is from Sony Corporation and uses worldwide accepted 13.56MHz carrier frequency, ASK modulation method with a 10% depth at either 212kbps or 424kbps data rates and bit coding is Manchester, MSB first.
- It is listed as NFC Forum Type 3 Tag Platform, and there are four main products that are offered from Sony for NFC applications.
  1. FeliCa Standard is used throughout contactless smartcard and mobile FeliCa products that can support various types of applications, such as transportation, e-money, and employee IDs.
  2. FeliCa Lite and Lite-S are minimized contactless smartcard products with an optimized file system and streamlined security functions, they can be used for cards and various other form factors, such as stickers.
  3. FeliCa Link refers to the series of products with both wireless and wired interfaces that combine the functions of FeliCa Lite-S and FeliCa Plug.
  4. FeliCa Plug (T3T NFC Dynamic Tag) is a wireless-interface product, which can be embedded into electronic devices, enabling the device to communicate with any NFC reader/writer or with any NFC smartphone.

FeliCa target markets (where you might find these being used)



FeliCa / NFC T3T Platform
Basic Operational Details

FeliCa Command Set

- **Polling** (**0x00**)
  - This command is used to acquire and identify a card
  - Returns response code 0x01, IDm, PMm of card and two more bytes of specific data, if requested. (i.e. System Code or Communication Performance / Data Rate Capability)

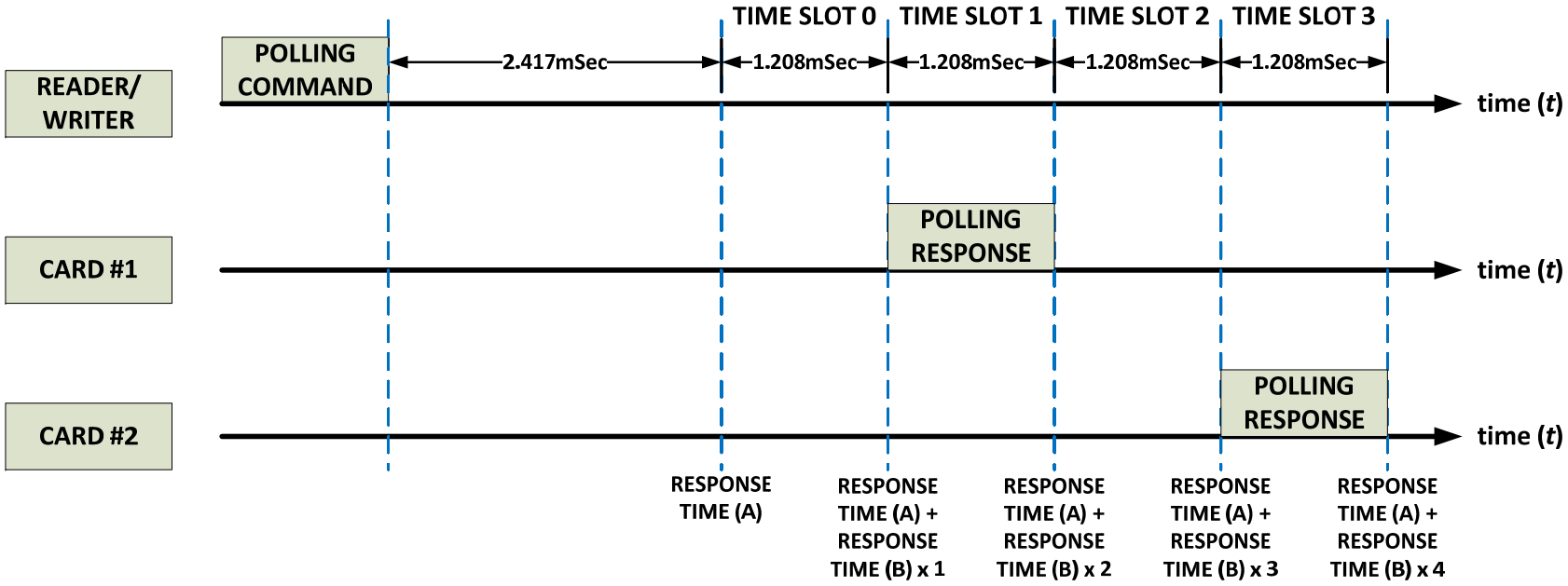
This command is formatted in the following manner:

# of bytes card will be receiving, including this one	Command Code	System Code (SC) (two bytes, wildcard values shown)		Request Code (RC)	Time Slot (one slot shown)
0x06	0x00	0xFF	0XFF	0x00	0x00

- **System Code**
  - System Code can be wildcard for either of these bytes, using in both makes any FeliCa card respond, other common ones to use are 0x12FC and 0x88B4.
- **Request Code**
  - 0x00 (shown) is a no request to the card. If a 0x01 or 0x02 is used here, System Code and Communication performance values are returned, respectively.
- **Time Slot**
  - Time slots available are: 0x00, 0x01, 0x03, 0x07 and 0x0F, for respectively allowing responses in 1,2, 4, 8 or 16 time slots. When using the timeslot method, the card will select a time slot randomly and transmit its response back. This approach is intended to reduce the probability of collisions between cards, not eliminate them as one finds in other card protocols anti-collision approaches.

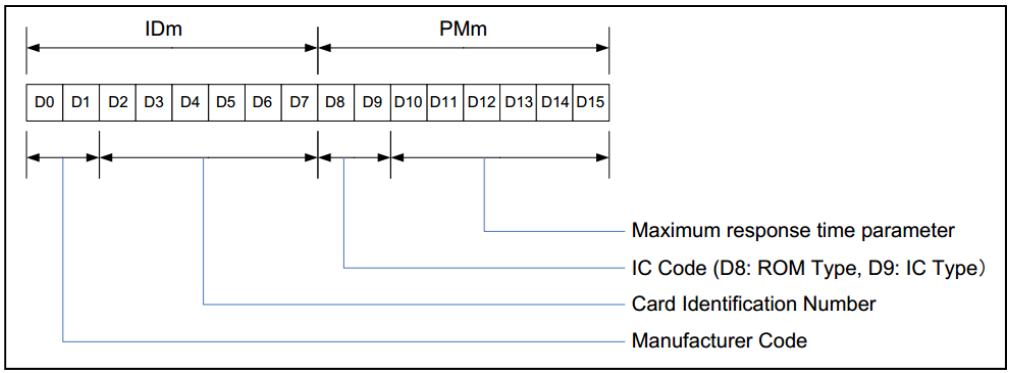
Anti-Collision with FeliCa

- FeliCa technology uses Time Slot method to reduce the probability of collisions between responses returned from multiple cards in the field of the reader.
- The start of the first time slot is called “Response Time (A)” and the width of the time slot is called ‘Response Time (B)”
  - Response Time (A) = 512 x 64/fc, where fc = 13.56MHz = 2.41652mSec
  - Response Time (B) = 256 x 64/fc, where fc = 13.56MHz = 1.20826mSec
- The number of time slots to be shared between reader and cards is sent in the polling command string, as previously mentioned, and can be either 1, 2, 4, 8 or 16 slots.
- In the diagram below is 4 slot example with two cards that selected slots 1 and 3 to respond in.

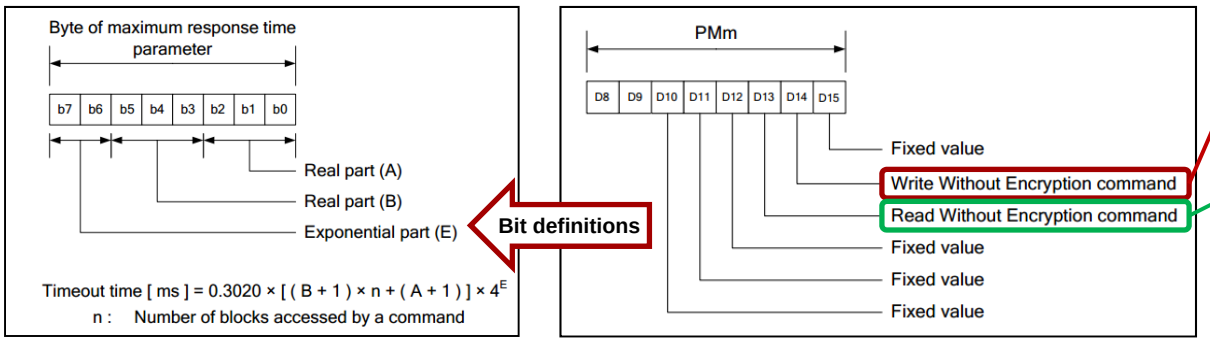


FeliCa Polling Response Details

- **IDm**: Manufacturing ID, this is an 8 byte field the card will return in response to a Polling command, includes Manufacturer Code (MC) and Card Identification Number (CIN)
- **PMm**: made up of the IC Code and the Manufacturing Parameters



- Inside the PMm are the maximum response time parameter bytes, which indicate the maximum time the card could take to respond to either a Read Without Encryption Command or a Write Without Encryption Command. (shown below as bytes D13 and D14)



D14 - **MAX WRITE TIME RESPONSE EXAMPLE**:

Card returns 0x43 in this byte, then timeout time would equal, based on the formula in the far left box, 4.8416mSec x ***n*** blocks

D13 - **MAX READ TIME RESPONSE EXAMPLE**:

Card returns 0x01 in this byte, then timeout time would equal, based on the formula in the far left box, 0.6052mSec x ***n*** blocks

FeliCa Polling Response Details (cont.)

- **Request Data Bytes**
  - **System Code**
    - **Common System Codes for FeliCa are: 0x12FC, 0x88B4**
  - **Communication Performance Bit Definitions**
    - **These bytes are indicating the data rate(s) the tag is capable of operating at:**

D0	D1								
0x00	B7	B6	B5	B4	B3	B2	B1	B0	
	---	---	---	---	---	---	---	X	0b: 212kbps not possible
									1b: 212kbps possible
	---	---	---	---	---	---	X	---	0b: 424kbps not possible
									1b: 424kbps possible
	---	---	---	---	---	0	---	---	0b: 848kbps not possible
									1b: 848kbps possible (reserved)
	---	---	---	---	0	---	---	---	0b: 1.6Mbps not possible
									1b: 1.6Mbps possible (reserved)
	---	0	0	0	---	---	---	---	Fixed value (all others RFU)
	X	---	---	---	---	---	---	---	0b: communication rate auto detect non-compliant
									1b: communication rate auto detect compliant

FeliCa Polling Response Example #1 (FeliCa Lite)

- **Response Packet Data Format**
  - **In the case when request data is not requested by setting RC = 0x00, in the Polling Command**

# of bytes reader will be receiving, including this one	Response Code	IDm								PMm							
0x12	0x01	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15

OR

  - **In the case when request data is requested by using RC = 0x01 (for System Code) in the Polling Command**

# of bytes reader will be receiving, including this one	Response Code	Idm								PMm								Request Data	
0x14	0x01	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D0	D1
FOR EXAMPLE ➔		0x01	0x27	0x00	0x62	0x99	0xE4	0x69	0xC6	0x00	0xF0	0x00	0x00	0x02	0x06	0x03	0x00	0x88	0xB4

Response Timings — System Code

OR

  - **In the case when request data is requested by using RC = 0x02 (for Data Rate), in the Polling Command**

# of bytes reader will be receiving, including this one	Response Code	Idm								PMm								Request Data	
0x14	0x01	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D0	D1
FOR EXAMPLE ➔		0x01	0x27	0x00	0x62	0x99	0xE4	0x69	0xC6	0x00	0xF0	0x00	0x00	0x02	0x06	0x03	0x00	0x00	0x01

Supported Data Rate = 212kbps

FeliCa Polling Response Example #2 (FeliCa Lite-S)

- **Response Packet Data Format**
  - **In the case when request data is not requested by setting RC = 0x00, in the Polling Command**

# of bytes reader will be receiving, including this one	Response Code	IDm								PMm							
0x12	0x01	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15

OR

  - **In the case when request data is requested by using RC = 0x01 (for System Code) in the Polling Command**

# of bytes reader will be receiving, including this one	Response Code	Idm								PMm								Request Data	
0x14	0x01	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D0	D1
FOR EXAMPLE ➔		0x01	0x2E	0x30	0xC8	0x51	0x59	0x41	0x82	0x00	0xF1	0x00	0x00	0x00	0x01	0x43	0x00	0x88	0xB4

Response Timings — System Code

OR

  - **In the case when request data is requested by using RC = 0x02 (for Data Rate), in the Polling Command**

# of bytes reader will be receiving, including this one	Response Code	Idm								PMm								Request Data	
0x14	0x01	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D0	D1
FOR EXAMPLE ➔		0x01	0x2E	0x30	0xC8	0x51	0x59	0x41	0x82	0x00	0xF1	0x00	0x00	0x00	0x01	0x43	0x00	0x00	0x83

Supported Data Rates = 212kbps and 424kbps

FeliCa Polling Response Example #3 (FeliCa)

- **Response Packet Data Format**
  - **In the case when request data is not requested by setting RC = 0x00, in the Polling Command**

# of bytes reader will be receiving, including this one	Response Code	IDm								PMm							
0x12	0x01	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15

OR

  - **In the case when request data is requested by using RC = 0x01 (for System Code) in the Polling Command**

# of bytes reader will be receiving, including this one	Response Code	Idm								PMm								Request Data	
0x14	0x01	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D0	D1
FOR EXAMPLE ➔		0x01	0x01	0x07	0x01	0x7E	0x0F	0x80	0x00	0x0F	0x0D	0x23	0x04	0x2F	0x77	0x83	0xFF	0x12	0xFC

Response Timings **System Code**

OR

  - **In the case when request data is requested by using RC = 0x02 (for Data Rate), in the Polling Command**

# of bytes reader will be receiving, including this one	Response Code	Idm								PMm								Request Data	
0x14	0x01	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D0	D1
FOR EXAMPLE ➔		0x01	0x01	0x07	0x01	0x7E	0x0F	0x80	0x00	0x0F	0x0D	0x23	0x04	0x2F	0x77	0x83	0xFF	0x00	0x83

Supported Data Rates = 212kbps and 424kbps

FeliCa Command Set (cont.)

- **Read Without Encryption** (**0x06**)
  - This command is used to read block data from a service that does not require encryption (≤ 4 at one time)
  - Returns response code 0x07, IDm, Status Flag bytes, # of Blocks and 16 byte wide data from each block

  This command is formatted in the following manner:

# of bytes card will be receiving, including this one	Command Code	IDm (retrieved from Polling Command Response)								# of Services	Service Code List (Little Endian)		# of Blocks to be read (1:4)	Block List Element	Block # to be read
0x10	0x06	D0	D1	D2	D3	D4	D5	D6	D7	0x01	0x09 0x0B	0x00	0x01	0x80	0x00

- # **of services**: Equal to 0x01
- **Service Code List**: Little Endian oriented two byte field in which the lower byte can be 0x09 or 0x0B, where: 0x09 = Random Service, R/W permission and 0x0B = Random Service, R/O and R/W permission
- # **of Blocks to be read**: Can be between 1 and 4. Each block to be read needs its own Block List Element and Block # specified, here above is showing reading one block. For up to four blocks, the # of blocks to be read should be incremented and followed by appropriate pairs of Block List Element + Block # to read. Also, if more bytes are sent out, the # of bytes the card will be receiving value should be incremented to correct value, too.
- **Block List Element**: To specify a Service and Block Number to be targeted for access, use Block List. In the Block List, the elements are enumerated.
- **Block # to be read**: Block # to read data from according to memory map

FeliCa Command Set (cont.)

- **Write Without Encryption** (**0x08**)
  - This command is used to write block data to a service that does not require encryption (1 block allowed at a time)
  - Returns response code 0x09, IDm and two status flag bytes

  This command is formatted in the following manner:

Length of Packet	CC	IDm (retrieved from Polling Command Response)								# of svcs	Service Code List (Little Endian)		# of Blocks to Write	Block List Element	Blk # to write	Block Data (16 bytes)													
0x20	0x08	D 0	D 1	D 2	D 3	D 4	D 5	D 6	D 7	0x01	0x09	0x00	0x01	0x80	0x00	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

- **# of services**: Equal to 0x01
- **Service Code List**: Little Endian oriented two byte field in which the lower byte shall be 0x09, where: 0x09 = Random Service, R/W permission (check examples that show 0xC9, 0x0B also)
- **# of Blocks to Write**: shall be equal to 0x01
- **Block List Element**: To specify a Service and Block Number to be targeted for access, use Block List. In the Block List, the elements are enumerated.
- **Block # to write**: Block # to write data to, according to memory map and access conditions
- **Block Data**: 16 bytes that are desired to be stored on the card at the block location

NFC Forum Type 3 Tag Operation Specification Overview

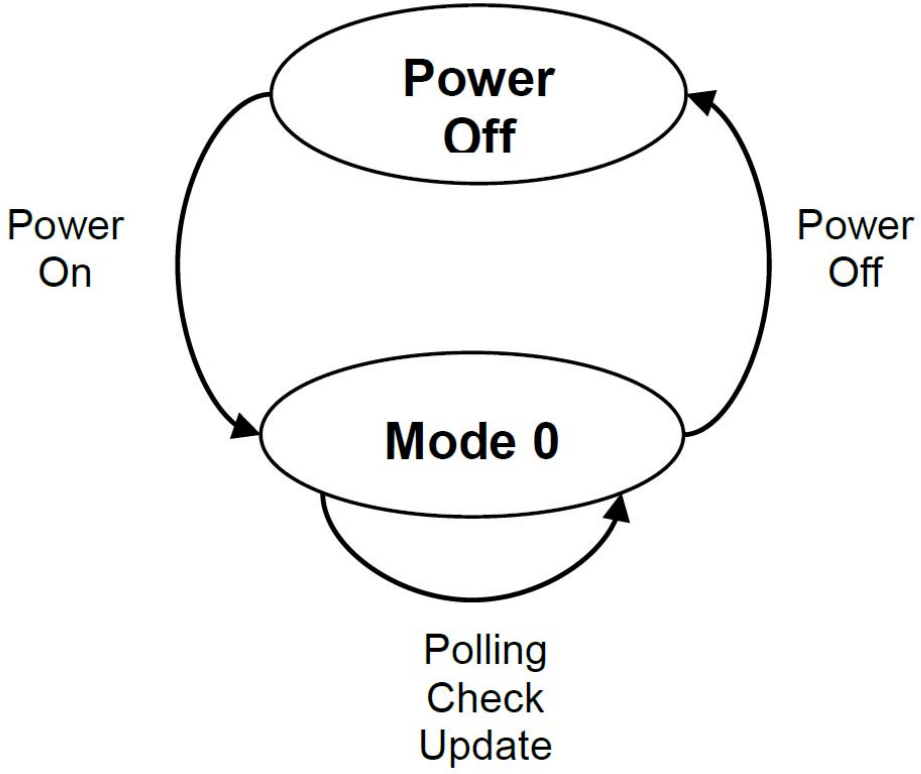
Introduction to T3T Platform

- NFC Forum Specification for FeliCa tags is called NFC Forum Type 3 Tag Operation Specification. This is a technical spec which outlines how these tags are to be used in an NFC application.
- The NFC Forum also has Analog and Digital specifications for these tags, and the content of those is same as what you will find in Sony specs regarding air interface, framing, and transmission handling.
- The command set is same as what we have just reviewed in the preceding slides, except NFC Forum changed two terms, so for clarification, a translation is appropriate:

  - FeliCa "**Polling**" command = NFC Forum "**Polling**" command or "**SENSF_REQ**"
  - FeliCa "**Read Without Encryption**" command = NFC Forum "**Check**" command
  - FeliCa "**Write Without Encryption**" = NFC Forum "**Update**" command

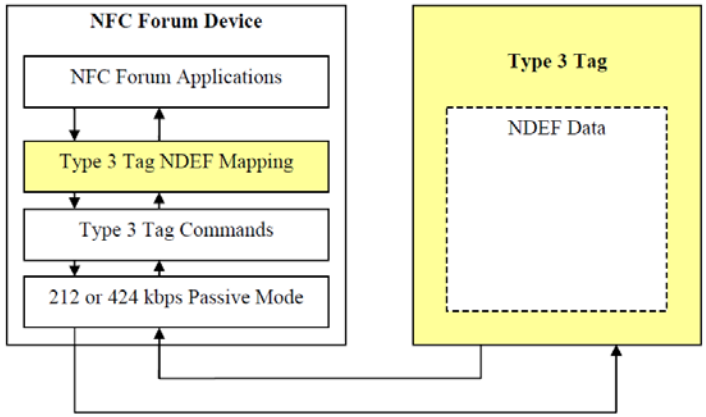
NFC Forum T3T State Diagram

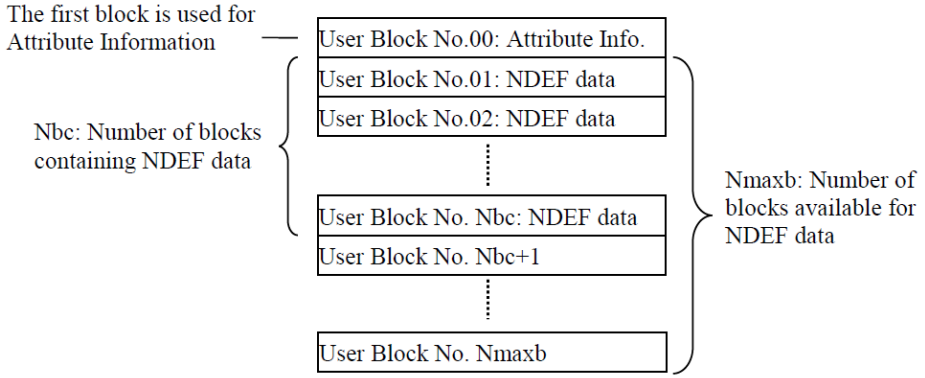
- As defined by the NFC Forum, the T3T Platform has only one state, called “Mode 0”. In this state the Polling, Check and Update commands can be received. None of these commands change the state of the Type 3 tag.



NDEF Access

- After detection, Check commands are used to determine the Attributes and retrieve the NFC Data Exchange Format (NDEF) data from the tag, if it already formatted and with content.
- Update commands would be used to NDEF format the tag or change the content in the case it was not previously configured for NFC applications.

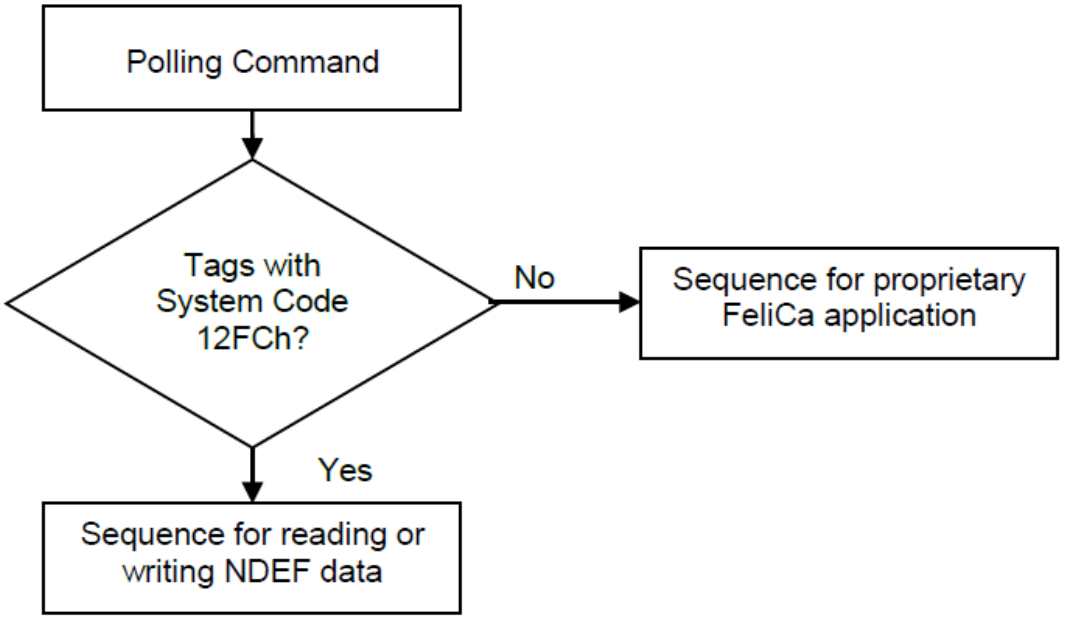




User Block No.00															
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15
Ver	Nbr	Nbw	Nmaxb		unused	unused	unused	unused	WriteF	RW Flag	Ln			Checksum	

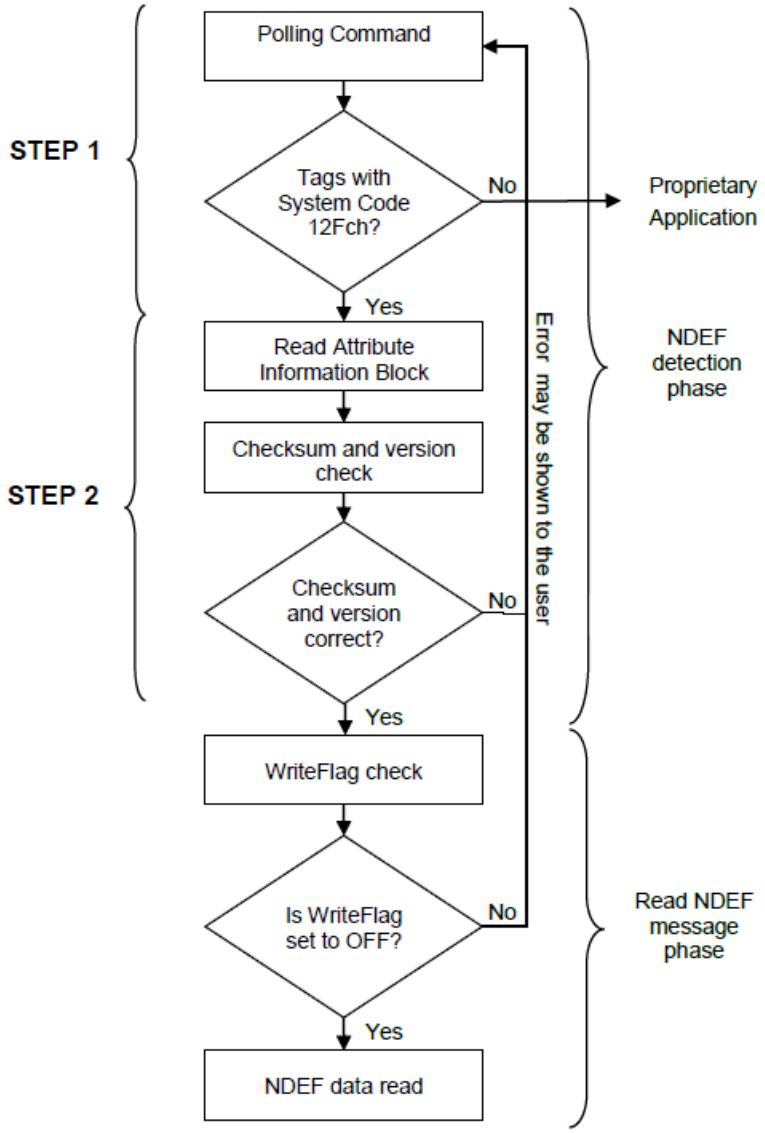
NDEF Detection

- The first step in detecting NDEF enabled T3 tags is to find the tags in the RF field that have the System Code of 0x12FC. FeliCa Lite and Lite S naturally have system code of 0x88B4, but they will respond to a polling command which has 0x12FC specified in the command string.
- Here below is the simple flow:



NDEF Detection (cont.)

- The polling command would be executed and the System Code checked in the response.
- Then Block 0 would be read out to determine the attributes of the card.



Reference: Section 6.1 in the NFC Forum T3T Platform specification, entitled NDEF Management Data

Deciphering the Attribute Block

- According the flow, after the Polling Command response, the Attribute Block must be read out.
- Below is Block 0 of an NDEF Formatted T3T Platform with contents of:

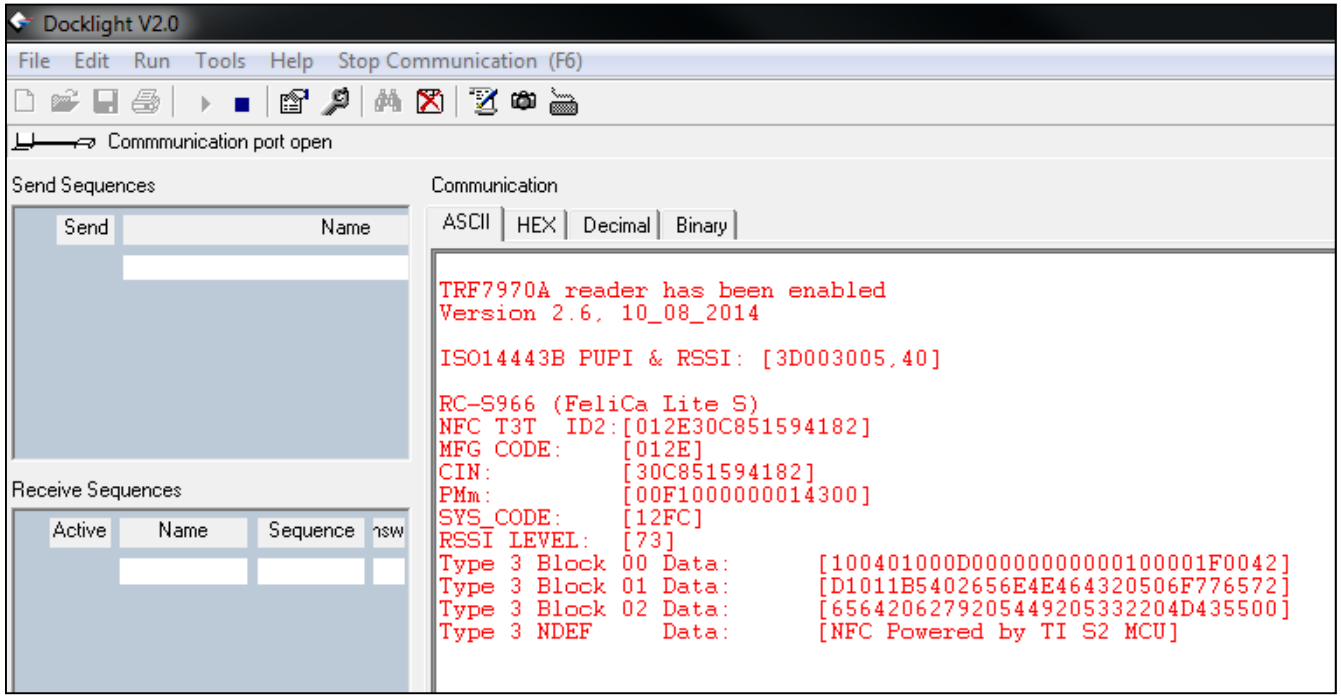
Mapping Version	Max Blocks to Read	Max Blocks to Write	Blocks for NDEF Storage		Unused				Write Flag	NDEF Access Read and Write Flag	Current NDEF Message Length			Checksum	
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
10	04	01	00	0D	00	00	00	00	00	01	00	00	1F	00	42

- To satisfy step 2 of the flow chart on the previous slide, this shows tag has Mapping Version 1.0, Write Flag is set to 0x00, NDEF Length is 31 bytes (on this particular card example), with a checksum of 0x42.
- With the Checksum in B14 and B15 being calculated using the formula:
  - Checksum = B0+B1+…B13

NOTE: B14 and B15 need to be updated anytime any values in B0 to B13 are changed.

Reading NDEF Data

- Now that polling and checks are complete, the reading of NDEF data can take place.
- In our example, from reading out the Attribute Block, Bytes B11 – B13, we know the NDEF message length in 31 bytes long.
- Since the T3T platform blocks are 16 bytes wide, this means we only need to read two blocks to retrieve the NDEF message content.
- Here below is showing the complete sequence of polling, reading out Block 0, then reading out two blocks of hex data from blocks 0x01 to 0x02, and converting to ASCII for human readability.

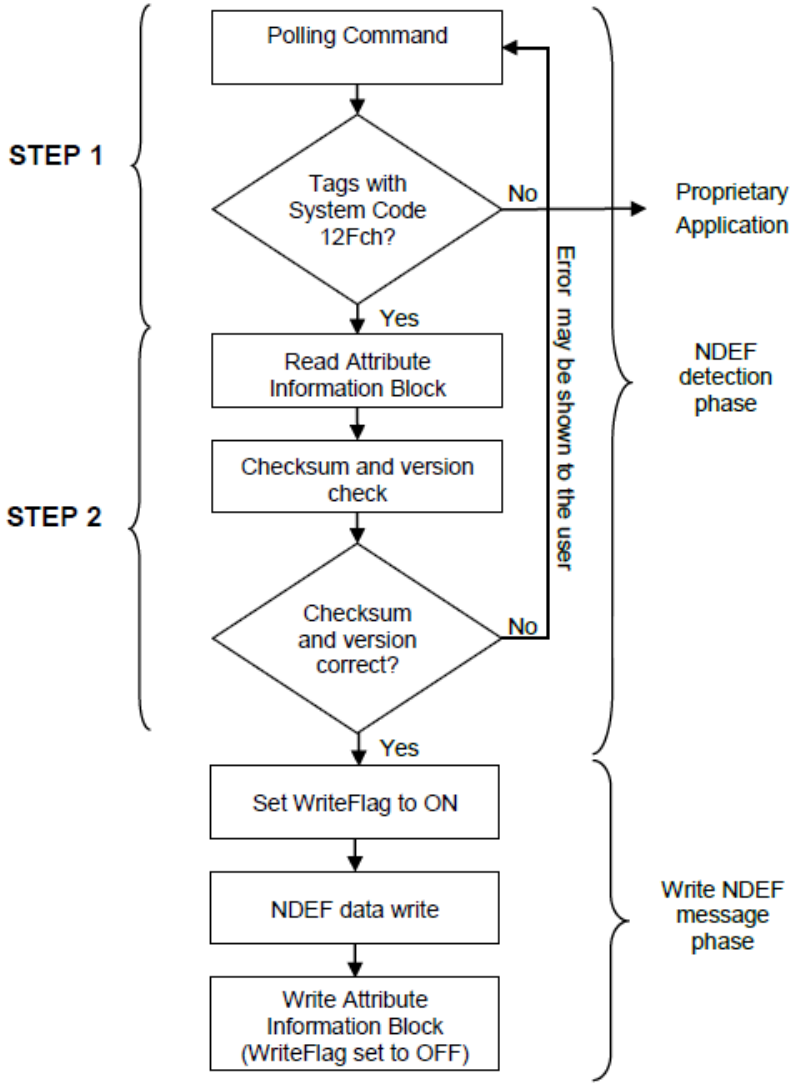


Meaning of the Bytes in NDEF Message Area (in this example)

- In the block 01 and 02 that were retrieved are bytes for:
  - **NDEF Record Header**:
    - SR=1, TNF=0x01(NFC Forum Well Known Type), ME=1, MB=1
  - **Length of Record Name**: **0x01**
    - 1 byte
  - **Length of the Payload Data**: **0x1B**
    - 27 bytes
  - **Record Name**: **0x54**
    - "T" = RTD Type Text
  - **Status Byte**: **0x02**
    - Length of language code in bytes = 2
  - **Language Code**: **0x65**, **0x6E**
    - "en" (hex to ASCII)
  - **NDEF message content**: **0x4E**, **0x46**, **0x43**, **0x20**, **0x50**, **0x6F, 0x77**, **0x65**, **0x72**, **0x65**, **0x64**, **0x20**, **0x62**, **0x79**, **0x20**, **0x54**, **0x49**, **0x20**, **0x53**, **0x32**, **0x20**, **0x4D**, **0x43**, **0x55**
    - "NFC Powered by TI S2 MCU" (hex to ASCII)

Writing NDEF Data

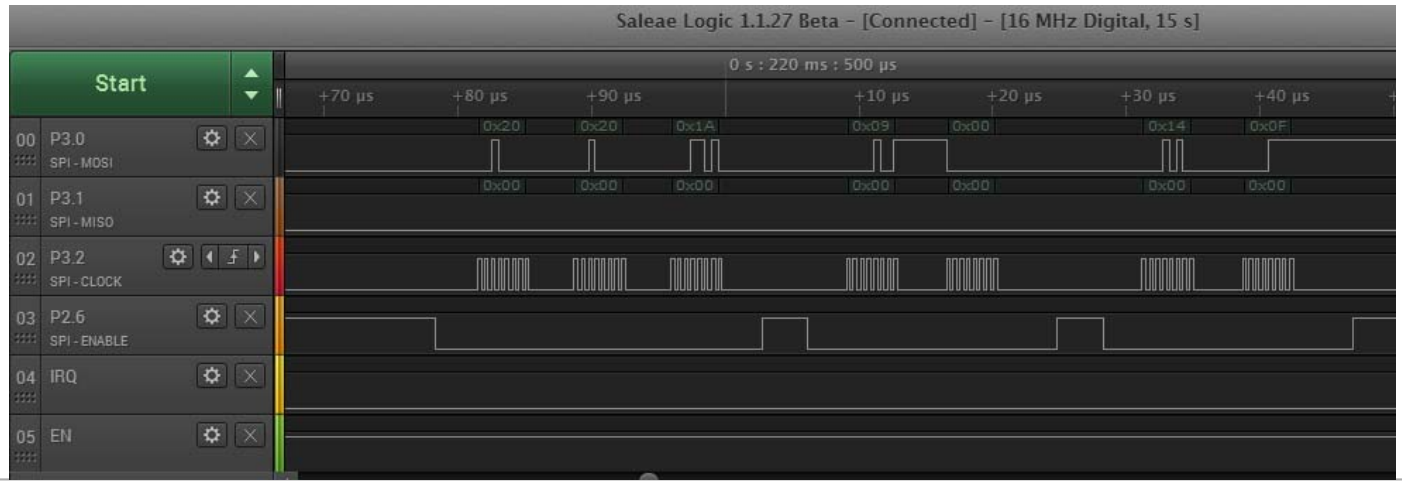
- Writing NDEF data has a similar flow as reading with the T3T Platform.
- See Slide 14 for command execution details.



Using the TRF7964A / TRF7970A with FeliCa / NFC T3T Platforms

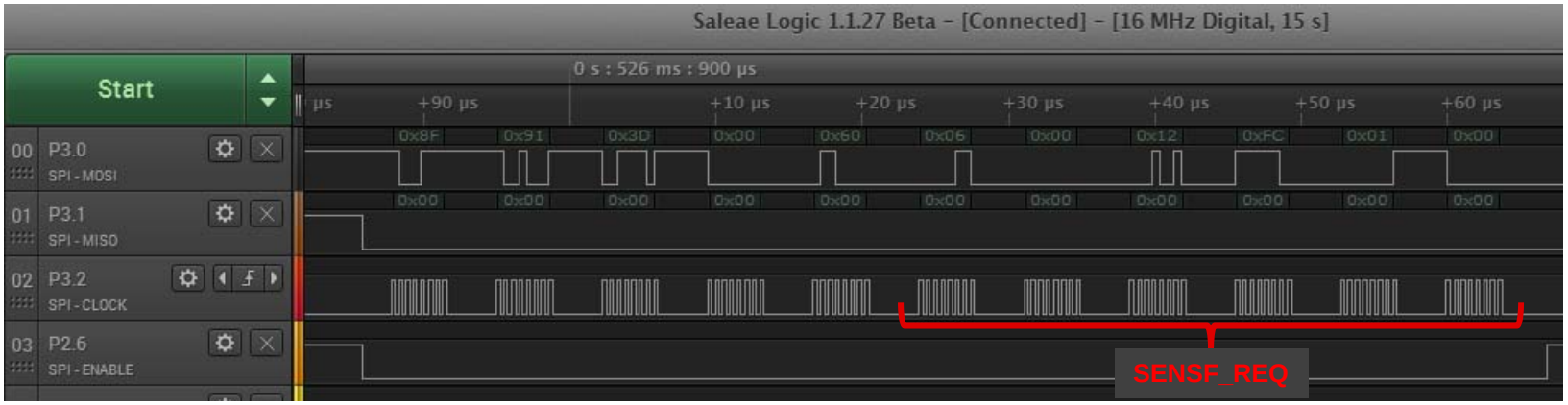
Configuring the TRF7964A / TRF7970A for FeliCa / T3T Platform Operations

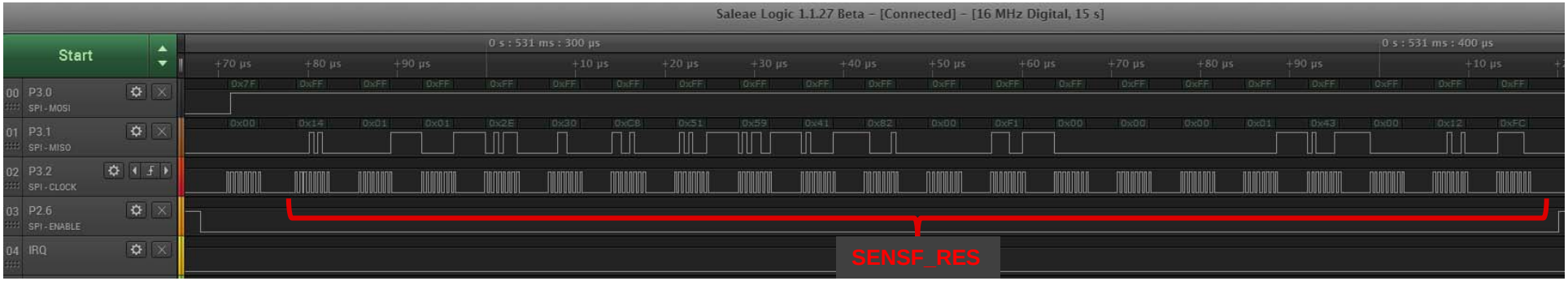
- Now that we know some relevant details about what the card is expecting from a protocol perspective, we can configure the TRF79xxA device accordingly.
- **Chip Status Control Register** (0x00)
  - Setting based on voltage in (VIN) to TRF79xxA
    - +5VDC ➔ 0x21 (full power out), 0x31 (half power out)
    - +3.3VDC ➔ 0x20 (full power out), 0x30 (half power out)
- **ISO Control Register** (0x01)
  - Must be set to 0x1A for 212kpbs, 0x1B for 424kbps (recommend to start with 0x1A)
- **SYS_CLK & Modulation Depth Register** (0x09)
  - Must be set to 0xY0, where Y= input clock frequency and desired SYS_CLK out
    - In MSP430G2553 project that will be discussed in next section, this register is set for 0x00, since MSP430G2553 is not using TRF7970A SYS_CLK Output
- **Adjustable FIFO IRQ Levels** Register (0x14)
  - Should be set for 0x0F



SENSF_REQ / SENSF_RES

- Then we can issue SENSF_REQ Command and get back SENSF_RES.

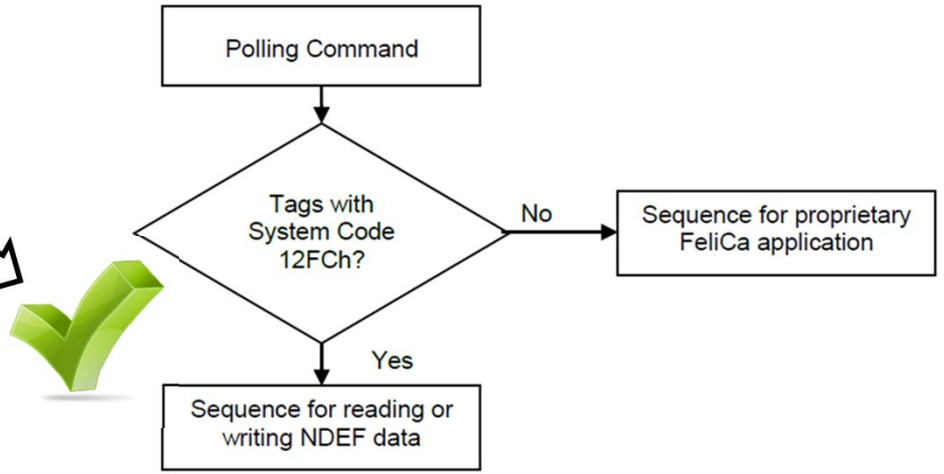




Following the NFC T3T Spec for NDEF Detection

- From the SENSF_REQ (issued with RC = 0x01), we get SENSF_RES.
- Because we set the System Code to 0x12FC and the RC = 0x01 in the SENSF_REQ, and the card we are using is responding to that System Code, at the end of the SENSF_RES response packet is the System Code 0x12FC.

From SENSF_RES
(see previous slide for entire string)



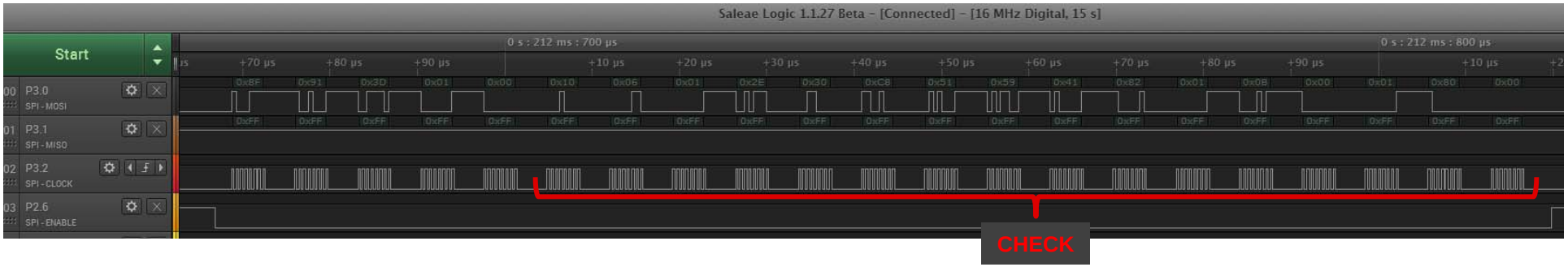
- Passing this check allows us to move on to reading out Block 0 for the Attributes of the card, which is next step in implementing the NFC Spec Flow.
- **NOTE**: FeliCa Lite and Lite S cards will respond to 0x12FC with 0x12FC and respond with 0x88B4 to the wildcards. Therefore, if wildcards are used for System Code in SENSF_REQ, the decision block above could be modified to accept cards that respond with 0x12FC **or** 0x88B4. Suica (Japan Railpass) cards will respond with an SC of 0x0003 if wildcards are used, indicating a proprietary FeliCa application, so there would be no disturbance or disruption to the flow shown above, if wildcards in were used instead of hardcoding 0x12FC into the SENSF_REQ.

Following the NFC T3T Spec for NDEF Detection (cont.)

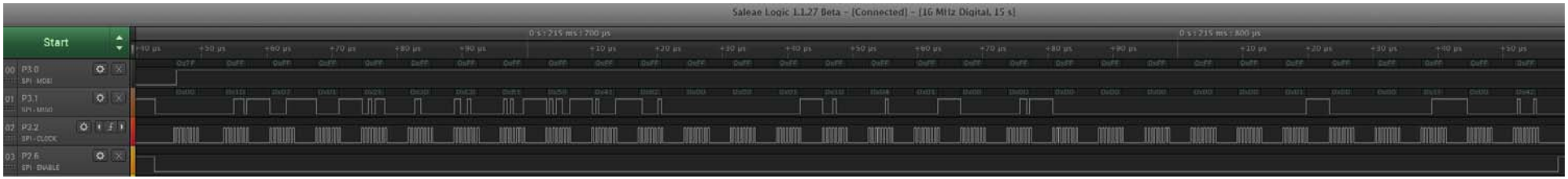
- Now we can issue a “Check” command on Block 0, to determine the attributes of the card.
- To review, here is the format of the command, which includes the IDm retrieved from SENSF_RES

# of bytes card will be receiving, including this one	Command Code	IDm (retrieved from Polling Command Response)								# of Services	Service Code List (Little Endian)		# of Blocks to be read (1:4)	Block List Element	Block # to be read
0x10	0x06	D0	D1	D2	D3	D4	D5	D6	D7	0x01	0x09	0x00	0x01	0x80	0x00
											0x0B				

- Here is the command actually being issued



- And the Check Response, for the Attributes Block of this card, to satisfy step 2 of the flow chart on Slide 20 , this shows tag has Mapping Version 1.0, Write Flag is set to 0x00, NDEF Length is 31 bytes (on this particular card example), with a checksum of 0x42.

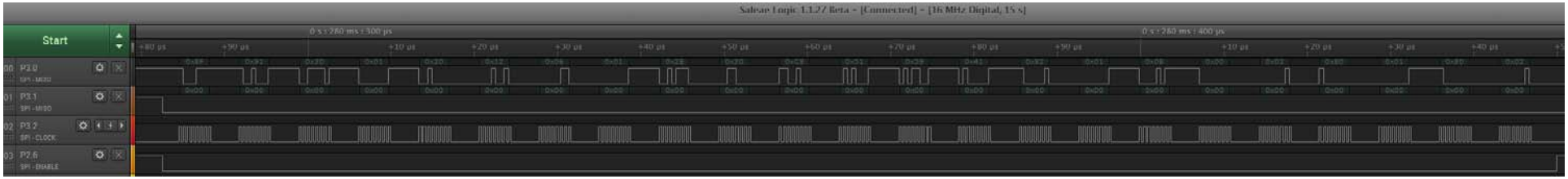


Following the NFC T3T Spec for NDEF Detection (cont.)

- From what we have derived from the card by reading the attributes block, now we can proceed forth with doing a two block read, since we know that the NDEF content is wholly contained inside of 31 bytes.
- To review, here is the format of the command, and in this case we will read two blocks, so we increment the length byte to 0x12, the # of blocks to be read out byte to 0x02 and add the extra two bytes at the end to read out block #2 (in comparison to the previous example on Slide 13)

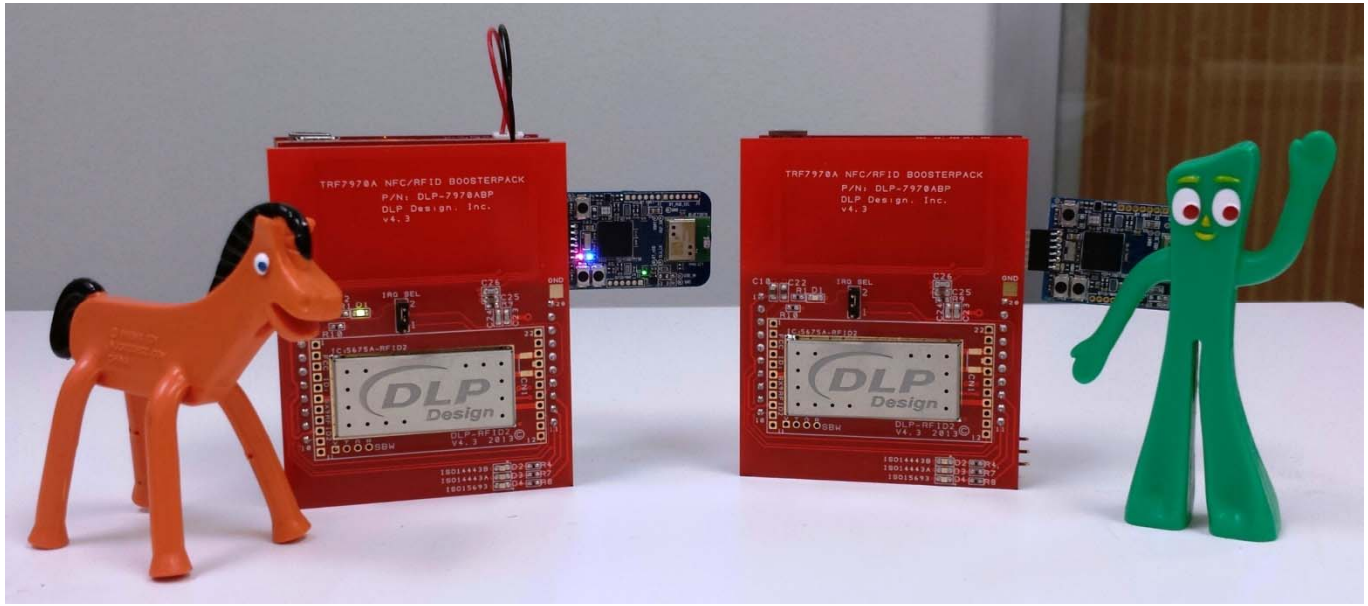
# of bytes card will be receiving, including this one	Command Code	IDm (retrieved from Polling Command Response)								# of Services	Service Code List (Little Endian)		# of Blocks to be read (1:4)	Block List Element	Block # to be read	Block List Element	Block # to be read
0x12	0x06	D0	D1	D2	D3	D4	D5	D6	D7	0x01	0x0B	0x00	0x02	0x80	0x01	0x80	0x02

- Here is the command actually being issued.



- The response is too long to show well in this presentation with the logic analyzer, but to review, please go to slide 23
- Presenter to show/capture actual LSA shot here.
- The data bytes captured are parsed out accordingly and converted to ASCII for human readability or just passed directly to a host for that processing to take place.
- The next section discusses / reviews the standalone MSP430G2553 LaunchPad + TRF7970A BoosterPack code project. (which was used to create this collateral).

MSP430G2553 LaunchPad + TRF7970A BoosterPack Code Example for T3T Platform



Background on the code project

- Compiled in Code Composer Studio Version: 6.0.1.00040
- MSP430G2553 + TRF7970A specific functions use about 1.9kB of Flash
- FeliCa / NFC T3T specific functions (reviewed earlier), with UART output strings use about 2.4kB of Flash.
- Combined total = ~4.3kB of flash needed to run the standalone example which does complete NDEF detection loop and returns data back to terminal program via UART.

NOTE: these values are from compiling with no optimizations still yet another improvement that can be made ☺

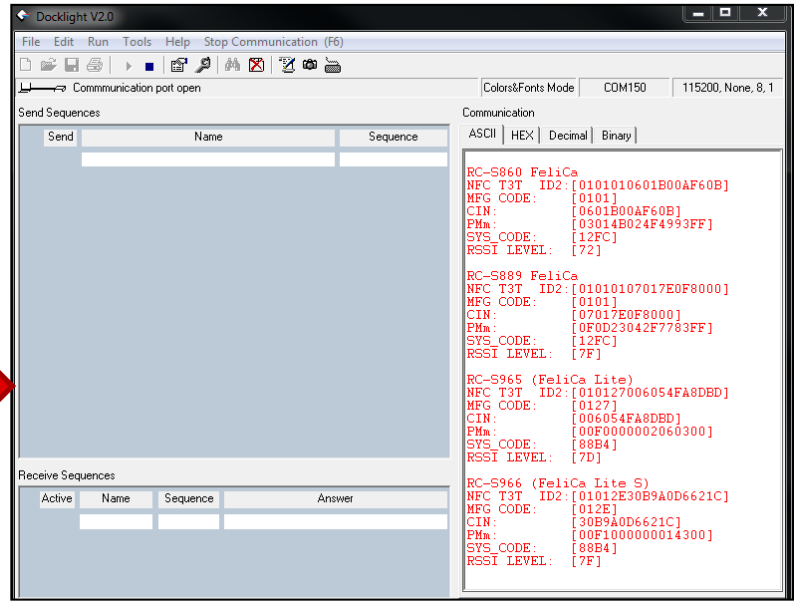
- The code project currently reads out Type 2, Type 3, Type 4A, Type 4B, Type 5 and HID PicoPass cards. The basic idea/concept/motivation of the project is to demonstrate a cost effective reader/writer solution which can be realized with the MSP430G series MCUs and the TRF79xxA NFC/RFID transceivers.
- INSERT NOTE HERE: to set expectation correctly about firmware limitations of this example.
- It (the code project) was implemented using the MSP-EXP430G2 LaunchPad and the DLP Design TRF7970A BoosterPack and can also be loaded onto the TI Design TIDM-NFC-EZ430-MODULE ➔ http://www.ti.com/tool/TIDM-NFC-EZ430-MODULE, with modification only needed to the trf7970BoosterPack.h file, for GPIO reassignments for the LEDs. (project has this modification already done, labeled and commented out)

Hardware & Development Environment Requirements

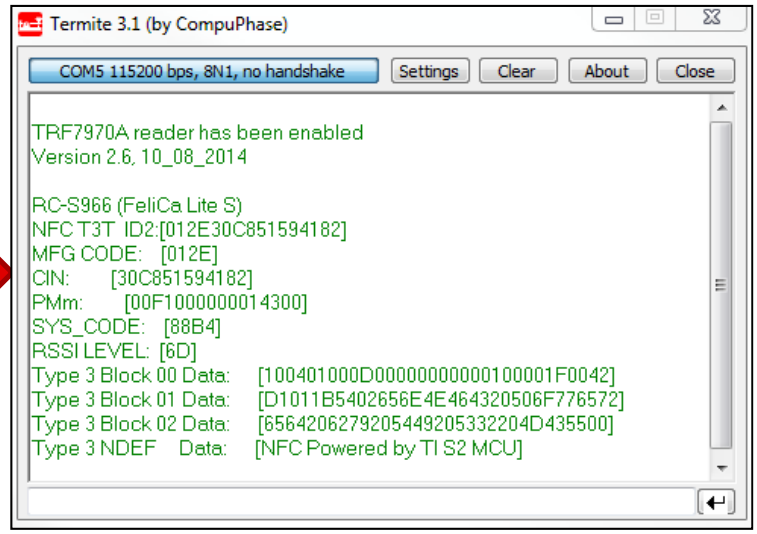
- MSP-EXP430G2 LaunchPad with MSP430G2553 installed
  - http://www.ti.com/tool/msp-exp430g2
- DLP-7970ABP (TRF7970A BoosterPack)
  - http://www.ti.com/tool/dlp-7970abp
- Code Composer Studio IDE for MSP430
  - http://processors.wiki.ti.com/index.php/Download_CCS
  - Latest recommended, code size limited free version will also work
- Terminal Program (for displaying UART output)
  - Use one integrated into CCS (when not debugging)
    - Or if debugging and you want to see UART output
  - Docklight ➔ http://www.docklight.de/
  - Termite ➔ http://www.compuphase.com/software/termite-3.1.zip
- Logic Analyzer (for debugging, if you make changes)
  - 8 or 16 channel ➔ https://www.saleae.com/cart

Importance of the Terminal Display

- Here is example of using TRF7970A BoosterPack with MSP-EXPG2 Launchpad to read out basic information from 4 different variations of Type 3 Tag platforms using Docklight



- Here is example of using TRF7970A BoosterPack with MSP-EXPG2 Launchpad to read out block data and also display NDEF information from FeliCa Lite-S card using Termite



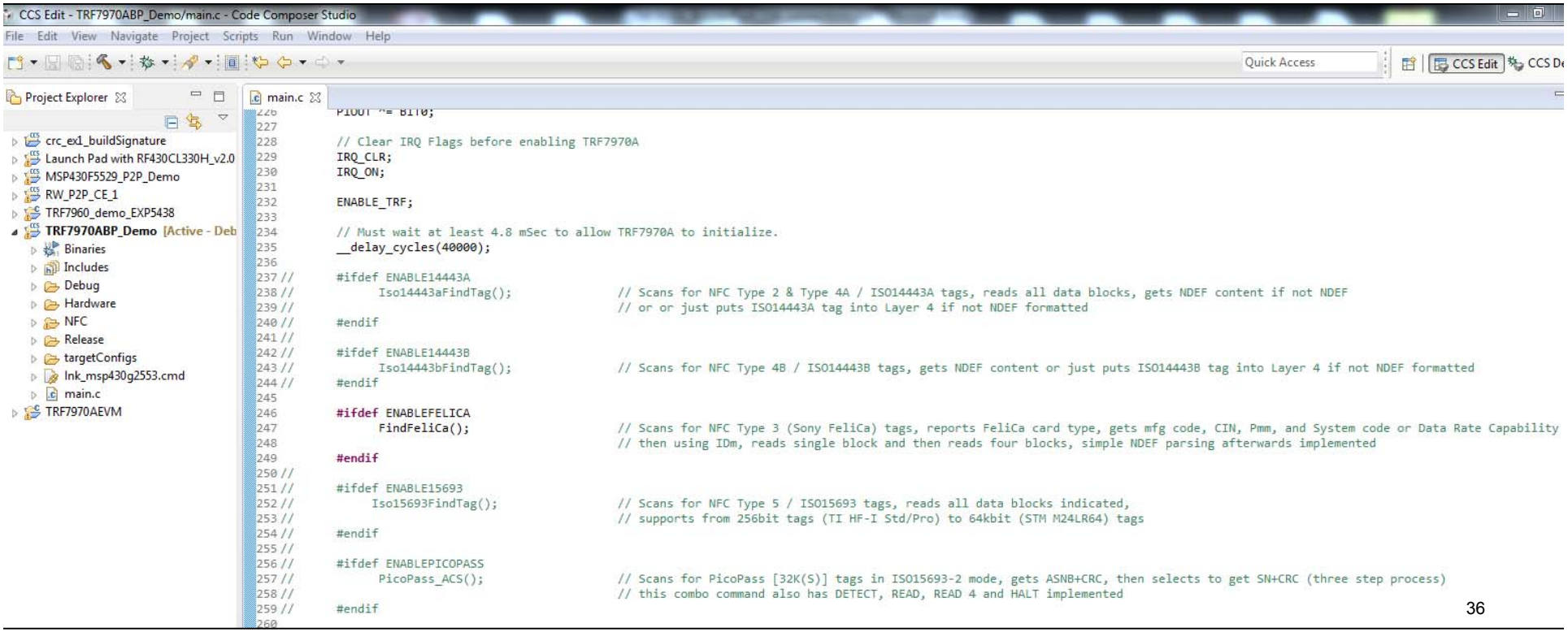
Logic Analyzer Connections to TRF7970A BoosterPack

- Connecting a Logic Analyzer to the TRF7970A BoosterPack is accomplished following the table below.
- This is handy tool to use when trying to quickly debug a new feature you are trying to implement, without having to figure out where to set breakpoints or watch windows.
- The connections in color (red, yellow, orange, green, brown, black and grey) correspond to channels on the Saleae LSA.

value	Outer Header 1	Inner Headers	TRF7970A BoosterPack LSA Pinout	Inner Headers	Outer Header 2	value
3VDC	1	9		20,11	20	GND
ANALOG IN	2				19	GPIO / PWM
UART RX	3				18	GPIO / CS
UART RX	4				17	
GPIO	5				16	RST
ANALOG IN	6			13	15	MOSI
SPI_CLK	7	12		16	14	MISO
IRQ	8	17			13	ISO15693 LED
SLAVE_SELECT	9	14			12	ISO14443A LED
ENABLE	10	10			11	ISO14443B LED

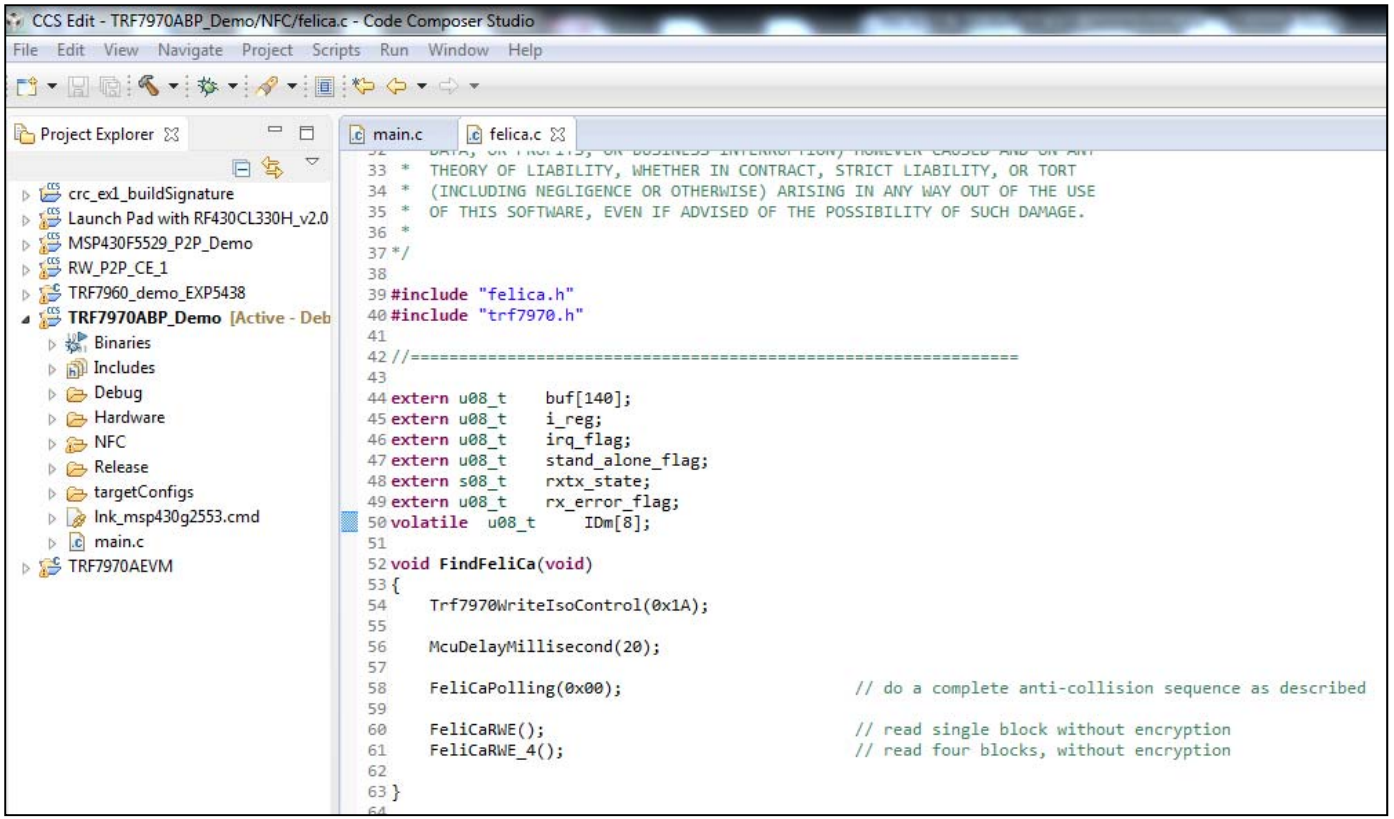
main.c

- Key feature of this project is to ability to enable/disable protocols easily.
- #if def ENABLE statements are utilized and can be commented in/out to realize this.
- Below is screen capture from main.c file which illustrates all protocols except FeliCa being commented out.
- A compile and download to the target at this point would yield system only looking for T3T tag types.



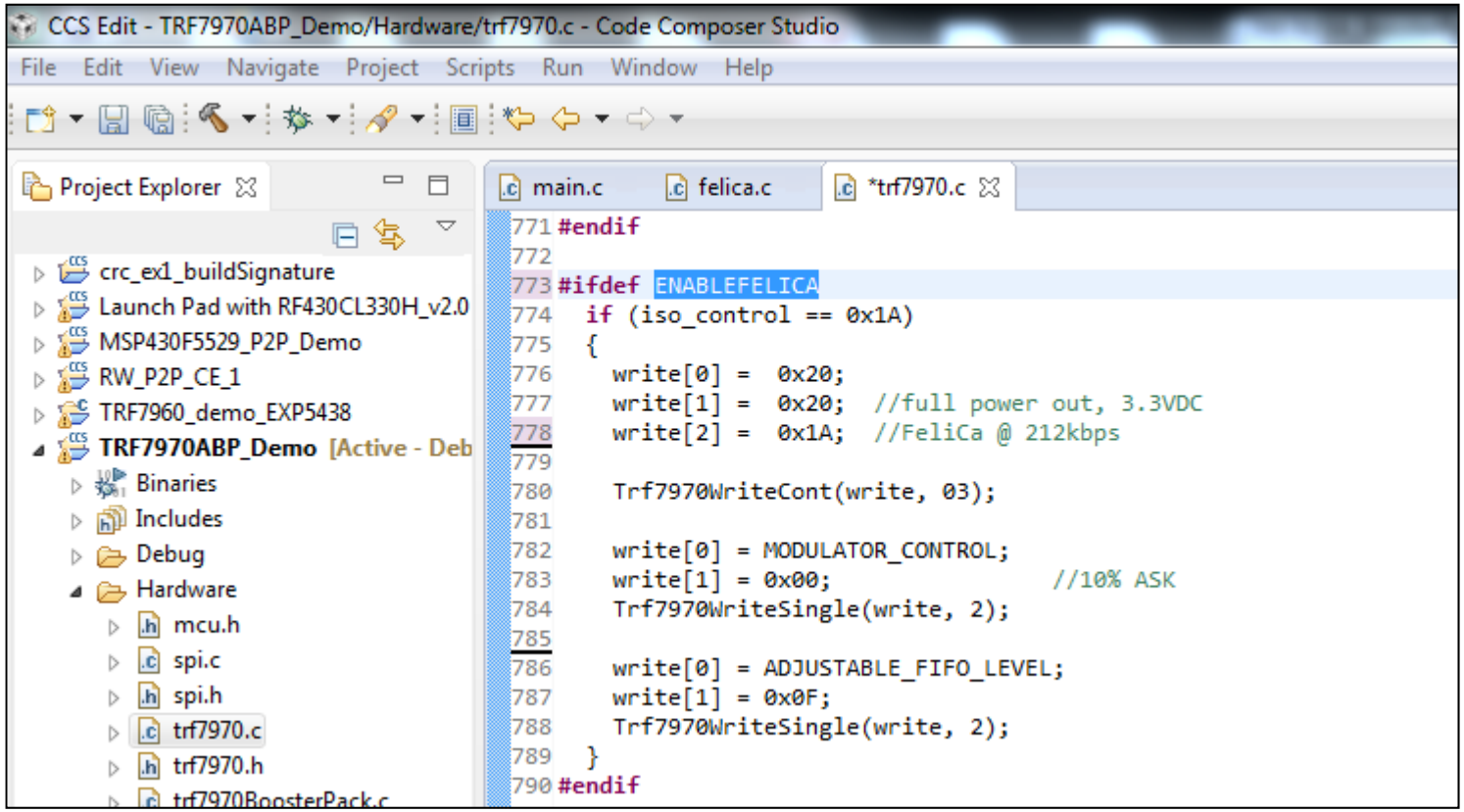
Declaration: FindFeliCa

- If we follow the declaration FindFeliCa from main.c, we can see that a simple flow was created which writes the ISO Control Register (which triggers another function), the polling function, followed by a single block read and a four block read.
- Next steps on this are to make the multiple block reads more dynamic, based on the content of Block 0, as previously discussed is required by NFC Forum.



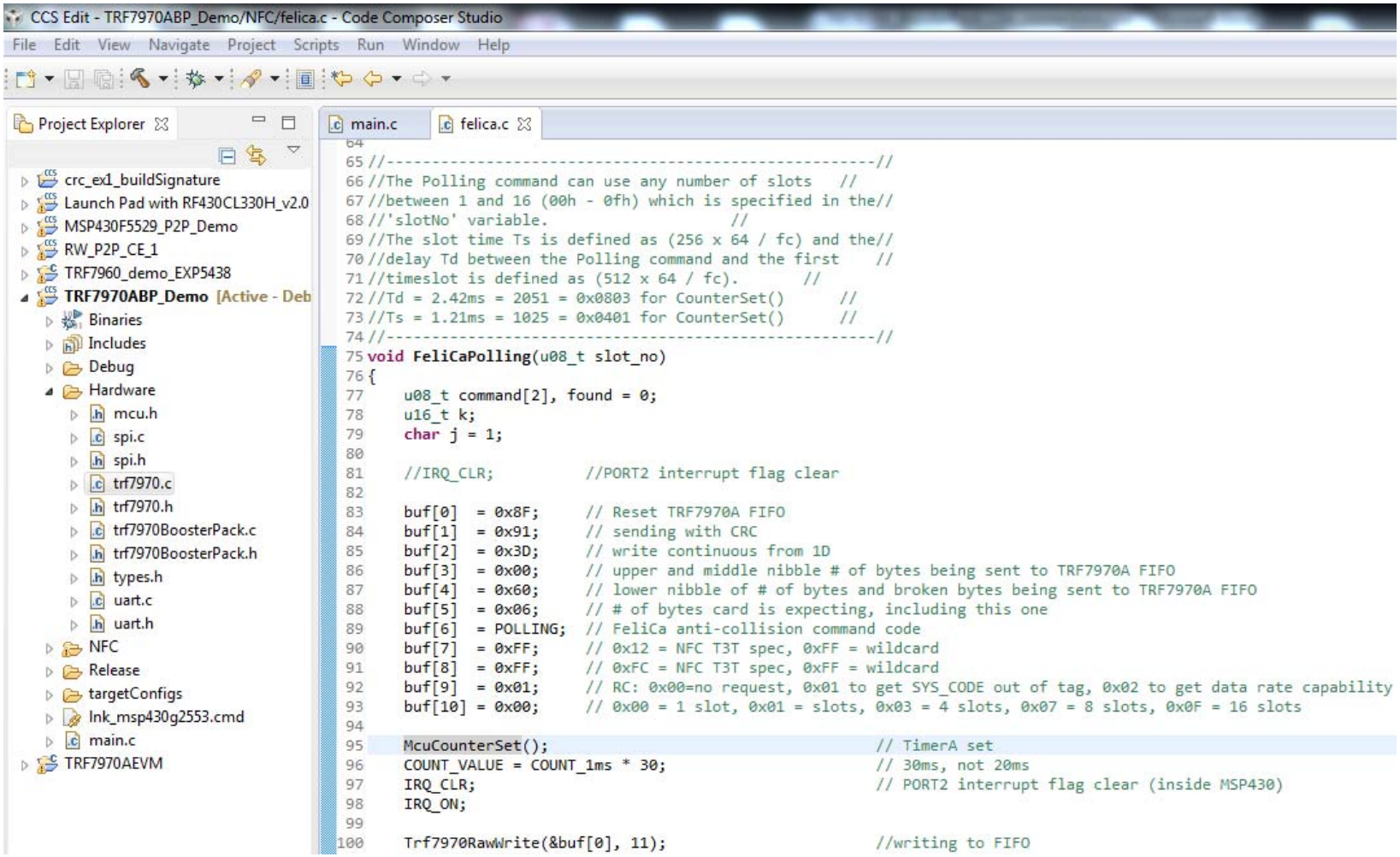
#ifdef ENABLEFELICA

- As mentioned in previous slide, the writing of the ISO Control Register in the FindFeliCa function triggers another function that checks ISO Control register and then configures the TRF79xxA correctly for the given protocol, as shown below.



FeliCaPolling

- FeliCaPolling Command Function:



FeliCaRWE Function

- This function performs a read of Block 0 from a FeliCa / T3T platform