



# **CS108 and CS463 Bluetooth and USB Byte Stream**

## **API Specifications**

**CS108: Bluetooth and USB**

**CS463: Bluetooth only**

**Version 1.45**

**2021 09 08**

# Chapter 1: Release Notes

Dates	Release	Description
2021 09 08	1.45	Add EM4325 GetSensorData command 0x003B
2021 08 11	1.44	Add MAC Register 0x0201 and Command Active packet
2021 08 10	1.43	Add 0xA008 and 0xA009 commands
2020 11 26	1.42	Add FM13DT160 MAC registers and HST commands
2020 11 18	1.41	Add R2000 low power mode MAC register 0x200 and HST command 0x14 (must set MAC register and then send MAC command)
2020 09 24	1.40	Add Inventory Cycle Begin command packet that contains MAC Error 0x0336
2020 09 16	1.39	Add MAC Register 0x0005 MAC Error Codes
2020 09 10	1.38	Update pkt-ver of OEM register read response and R2000 register read response
2020 09 09	1.37	Add R2000 registers descriptions section, R2000 write response packet
2020 09 08	1.36	Add MAC temperature registers: ambient and PA temperature, add MAC bypass read and write host commands, add R2000 0x0450 register for RF LNA Gain, IF LNA Gain, RF High Compression Mode, AGC

		Gain Range configuration.
2020 05 20	1.35	Add MAC commands and clean up definitions wordings.
2019 12 16	1.34	Add more wordings to clarify the RFID reader tag uplink
2019 12 05	1.33	Add UCode 8 Brand ID capability in INV_CFG with explanations of how it works.
2019 09 16	1.32	Update phase will use only bit 0 to bit 5, and also define the equation to get phase in radians and phase in degrees
2019 09 11	1.31	Update Normal Mode Inventory Data Byte 1 Flag definitions. One of them is phase availability
2019 08 30	1.30	Add Antenna Port # to Byte 6 of Compact Mode Inventory Uplink Packet
2019 08 28	1.29	Add CS463 Bluetooth compatibility
2019 08 22	1.28	Add Silicon Lab Reset API 0xB00C
2019 08 08	1.27	Add CW time after Select command configuration in 0x0801 MAC register. Improve some descriptions
2019 05 03	1.26	Add error codes for Tag Access Packets
2019 04 15	1.25	Add descriptions of Block Write in Section A.2
2019 01 14	1.24	Add select targets
2018 12 17	1.23	Update TAGWRDAT_SEL and TAGWRDAT_ register
2018 09 28	1.22	Add HST_CMD of read OEM and related registers
2018 08 23	1.21	Add "Fast Trigger Button

		Barcode Scanning” mode using special notification commands 0xA006 and 0xA007
2018 08 20	1.20	Add HST_CMNDIAGS, and remind user to disable Command Active packet
2018 08 16	1.19	Add Trigger Release button simultaneous send RFID Abort setting (Default will send RFID Abort)
2018 08 03	1.18	Add barcode uplink packet format in Chapter 9.2.1; add RFID reader command packet version explanation in Appendix A
2018 05 28	1.17	Modify barcode reader commands
2018 04 17	1.16	Add recommendation for Tag Delay parameter in INV_CFG register and INV_CYCLE_DELAY register
2018 04 17	1.15	Add reminders of the “Reversely Populated” packetizing arrangement convention of firmware registers
2018 04 16	1.14	Add INV_CYCLE_DELAY
2018 04 10	1.13	Add BT disconnection command
2018 03 23	1.12	Add barcode reader command to turn on and off vibrator
2018 02 27	1.11	Add Appendix D on barcode pre-setup and normal operation
2018 02 13	1.10	Add lock tag example in C.6
2018 02 12	1.9	Add MAC Error 0x0336 error definition
2018 01 30	1.8	Add search tag example in C.5
2018 01 26	1.7	Add CRC lookup table and

		codes in Appendix K
2017 12 11	1.6	Add descriptions on Bluetooth and USB connection methods
2017 11 20	1.5	Add sequence number in packet header byte 4 “reserve byte” when doing RFID inventory uplink
2017 11 15	1.4	Update INV_CFG to add inv_mode: adding a compact inventory mode to reduce overhead in inventory response packet – hence much faster transfer rate
2017 11 10	1.3	Update factory default settings
2017 08 21	1.2	Update TAGACC_DESC_CFG to allow 5 bits retry count
2017 05 18	1.1	Add commands in chapter 11
2017 05 02	1.0	Initial Release

# Chapter 2: Content

Chapter 1: Release Notes .....	2
Chapter 2: Content.....	6
Chapter 3: Introduction .....	9
Chapter 4: Interfaces to CS108 and CS463 .....	10
4.1 Bluetooth Connection Details .....	10
4.1 USB Connection Details.....	11
Chapter 5: LED definitions .....	12
Chapter 6: Push buttons Definition .....	13
Chapter 7: API Set Definition .....	14
7.1 API Format.....	14
7.2: Header of RFID Reader Commands .....	16
7.3: Header of Barcode Reader Commands.....	17
7.4: Header of Special Notifications.....	18
7.5: Header of Silicon Lab IC Commands .....	19
7.6: Header of Bluetooth IC Commands .....	20
Chapter 8: Payload – RFID Reader Commands .....	21
8.1 Downlink Payload.....	21
8.2 Uplink Payload.....	22
Chapter 9: Payload – Barcode Reader Commands .....	23
9.1 Downlink Payload.....	24
9.2 Uplink Payload.....	25
9.2.1 Uplink Barcode Payload Format.....	25
Chapter 10: Payload – Special Notifications .....	26
10.1 Downlink Payload.....	27
10.2 Uplink Payload.....	28
Chapter 11: Payload – Silicon Lab IC Commands.....	29
11.1 Downlink Payload.....	29
11.2 Uplink Payload.....	31
Chapter 12: Payload – Bluetooth IC Commands.....	32
12.1 Downlink Payload.....	32
12.2 Uplink Payload.....	33
Chapter 13: Factory Default Settings .....	34
Appendix A - CSL RFID Firmware Command Specification .....	35

A.1 Introduction .....	35
A.2 High Level API vs Low Level API .....	36
A.3 MAC Register Access Packet (Downlink).....	39
A.4 MAC Registers Description .....	41
A.5 R2000 Registers Description .....	74
A.6 OEM Registers Description .....	75
A.7 Command State Packet (Uplink) .....	76
A.8 Control Command (Downlink & Uplink) .....	90
Appendix B – RFID Firmware Error Codes.....	91
Appendix C – RFID Reader Firmware Command Sequence Examples.....	97
C.1 RFID Reader Initialization Example .....	97
Radio Open.....	97
Set RF Power.....	98
Set Channel .....	98
Set Profile.....	99
C.2 Tag Operations: Inventory Example .....	100
Set Inventory Parameters .....	100
Set Inventory Algorithm .....	100
Start Inventory.....	100
Stop Inventory .....	100
C.3 Tag Operations: Read Example .....	102
Set Inventory Parameters .....	102
Set Inventory Algorithm (Fixed Q = 0).....	102
Select the desired tag (e.g.111122223333444455556666).....	102
Start reading tag data of the selected tag .....	103
C.4 Tag Operations: Write Example .....	104
Set Inventory Parameters .....	104
Set Inventory Algorithm (Fixed Q = 0).....	104
Select the desired tag (e.g.111122223333444455556666).....	104
Start writing tag EPC data (e.g.000022223333444455556666) .....	105
C.5 Tag Operations: Search Tag Example .....	106
Set Inventory Parameters .....	106
Set Inventory Algorithm (Fixed Q = 0).....	106
Select the desired tag (e.g.111122223333444455556666).....	106
Start Searching .....	107
Stop Searching .....	107
C.6 Tag Operations: Lock Example .....	108
Set Inventory Parameters .....	108

Set Inventory Algorithm (Fixed Q = 0).....	108
Select the desired tag (e.g.111122223333444455556666).....	108
Start locking access password memory (assume access password is 0x11223344).....	109
Appendix D: Barcode Reader Command Sequence Examples.....	110
D.1 Pre-setup of Barcode Reader.....	110
D.2 Normal Operation of Barcode Reader .....	111
Appendix E – Reader Modes/Link Profiles .....	112
Appendix F – Session.....	113
Appendix G – Tag Population and Q .....	114
Appendix H – Query Algorithm .....	115
Appendix I – Target .....	116
Appendix J – Security .....	117
Appendix K – Tag Focus.....	119
Appendix L – Receive Path Parameters.....	120
Appendix M – Models & Regulatory Regions .....	121
Appendix N – CRC Table/Compute Codes.....	122



# Chapter 3: Introduction

CS108 is a Bluetooth or USB controlled RFID handheld reader sled. The sled allows the mounting of popular smart phones such as Android phone and iPhone, or iPod, onto the top; or connecting via USB from a PC or embedded system. One can also connect via Bluetooth from a Windows 10 laptop to it. An application inside the smart phones, downloadable free of charge from corresponding stores (Android Market or Apple App Store) will send commands via Bluetooth to the reader sled and operate the RFID reader module, the barcode module, the LEDs inside. Likewise, an application in the PC or embedded system can send commands down via USB to the CS108. The App in the smart phone or the PC will also handle the display of data for interactive use by the operator, plus transferring of data out to the Internet Cloud.

The data collected by the smart phone App will contain the following:

1. RDID tag data
2. Barcode data

The user can further add functionalities to the application by developing their own applications based on the source codes of the smart phone App or the PC App, which is freely available from Convergence Systems Website.

This document describes the Bluetooth and USB Byte Stream API set. The CS108 can be controlled via a USB cable by a PC or any embedded system, sending byte stream down to CS108, and with CS108 responding back via the USB cable in a byte stream manner. The CS108 can also be controlled via Bluetooth by iPhone, iPad, Android phone, Windows 10 PC with Bluetooth BLE 4.0 connectivity, sending byte stream down to CS108, and with CS108 responding back via Bluetooth in a byte stream manner. The following chapters define this byte stream interface. On the PC or embedded system platform, the user can employ any language: C#, Objective C, Swift, Java, Python, machine language, etc.

# Chapter 4: Interfaces to CS108 and CS463

For CS108, the commands can be fed in via 2 physical interfaces:

- 1) Bluetooth BLE 4.0/4.1/4.2 (BLE stands for Bluetooth Low Energy)
- 2) USB

For CS463, this document only describes how to connect to it via BLE 4.0/4.1/4.2.

(CS463 has many other connectivity options, such as Ethernet, Wi Fi, which are NOT described in this document)

These are all byte stream commands.

There are 5 different groups of commands and replies/notifications:

- 1) RFID reader commands
- 2) Barcode reader commands
- 3) Special Notifications
- 4) Silicon Lab IC commands
- 5) Bluetooth IC commands

## 4.1 Bluetooth Connection Details

For Bluetooth 4.0/4.1/4.2 connection, i.e. Bluetooth Low Energy, GATT protocol is used for search and connection.

UUID is discoverable by the smart phone whenever the CS108 is flashing the Bluetooth LED. (CS463 does not have a Bluetooth LED)

When the application “scan” for Bluetooth devices, the iOS OS will provide the application with the Reader’s Name, Device UUID (9800), Service UUID, etc.

CS108’s specific Service UUID is 9900 and 9901. 9900 is for downlink service from smart phone to

CS108 (write), 9901 is for uplink service from CS108 to smart phone (notification)

iOS will use these 2 numbers (9900 and 9901) to generate “characteristic UUIDs” for the application to use.

Write	=	00009900-0000-1000-8000-00805f9b34fb	(for downlink command sending)
notification	=	00009901-0000-1000-8000-00805f9b34fb	(for uplink notification)

In other words, CS108/CS463 reader uses Bluetooth notification to send data back to smart phone. CS108/CS463 reader allows smart phone to use Bluetooth write to send data to the reader.

## 4.1 USB Connection Details

USB HID library on the host platform is used. The protocol using USB connection as described in this document is only for CS108.

# Chapter 5: LED definitions

There are 7 LEDs on the CS108 device:

- 1) Power on LED: Turn on whenever power switch turns on power
- 2) Charging LED: Turn on whenever battery is being charged. Brighter when battery is more drained and charging current is higher.
- 3) External Power LED: Turn on whenever external power is available and used.
- 4) RFID LED: turned on continuous when RFID power is on. Flashes when tags are being read – (inventory started).
- 5) Barcode LED: turned on continuous when BARCODE power is on. Flashes when barcodes are being read – (barcode reading started).
- 6) Status LED: status of Silicon Lab IC
- 7) Bluetooth LED: status of Bluetooth – on (waiting for pairing), off, pairing, paired
  1. When Bluetooth push button is pressed for 3 seconds, Bluetooth LED starts slow flashing, 1 second on, 1 second off. Unit is now ready for discovery and pairing by smart phone
  2. During pairing, fast flash, 0.5 second on, 0.5 second off
  3. When successful paired, Bluetooth LED changes to continuous on
  4. Anytime when Bluetooth is on (Bluetooth LED either slow flashing or fast flashing or continuous), by long pressing Bluetooth push button for 3 seconds Bluetooth LED will be turned off. Bluetooth will be turned off.

# Chapter 6: Push buttons Definition

There are 3 buttons on the CS108 device:

- 1) Reset button: press reset button continuously for 20 seconds, reset to factory default
- 2) Power button: press power button continuously for 3 seconds, release button, after that power will turn on with power LED light up. Press power button continuously for 3 seconds, release button, then it will power off with power LED turned off.
- 3) Bluetooth button: this button controls Bluetooth on, off and pairing.
  1. When Bluetooth push button is pressed for 3 seconds, Bluetooth LED starts slow flashing, 1 second on, 1 second off. Unit is now ready for pairing by smart phone
  2. Anytime when Bluetooth is on (Bluetooth LED either slow flashing or fast flashing or continuous), by long pressing Bluetooth push button for 3 seconds Bluetooth LED will be turned off. Bluetooth will be turned off.

# Chapter 7: API Set Definition

There are 5 types of commands/responses/notifications API. All of them are byte streams:

- 1) RFID Reader
- 2) Barcode Reader
- 3) Special Notifications
- 4) Silicon Lab IC
- 5) Bluetooth IC

## 7.1 API Format

API consists of an 8-byte header and a payload.

Header (8 bytes)	Payload (maximum 120 bytes)
------------------	-----------------------------

The header contains information of whether it is a command/reply/notification, downlink or uplink, the length of the payload inside, CRC of packet.

### Header Format:

Prefix (1 byte)	Connection (1 byte)	Payload Length (1 byte)	Destination/ Source (1 byte)	Reserve (1 byte)	Direction (1 byte)	CRC of packet (2 bytes)
--------------------	------------------------	-------------------------------	------------------------------------	---------------------	-----------------------	----------------------------

### Header Field:

Name	Length (byte)	Value – Description.
Prefix	1	0xA7
Connection	1	0xE6 – USB 0xB3 – Bluetooth
Payload Length	1	1 to 120 – Payload length
Destination/Source	1	0xC2 – RFID 0x6A – Barcode 0xD9 – Notification 0xE8 – Silicon Lab IC

		0x5F – Bluetooth IC
Reserve	1	0x82 in most cases, <b>but for RFID Uplink (byte 3=C2, byte 5=9E), byte , this is a sequence number incrementing from 0 to 255 and repeat.</b> Application needs to store this sequence number. It should take in the first number and then track it as it increments.
Direction	1	0x37 – Downlink 0x9E – Uplink
CRC	2	CRC of packet

**Notes:**

1. CRC is not used when value is zero. For downlink, no need to use.
2. Payload length cannot be greater than 120.
3. Downlink means from smart phone/PC toward CS108. Uplink means CS108 toward smart phone/PC

In summary, there are 5 types of commands:

1	RFID Reader Command Header	RFID Reader Command Payload
2	Barcode Reader Command Header	RFID Reader Command Payload
3	Special Notification Command Header	Special Notification Command Payload
4	Silicon Lab IC Command Header	Silicon Lab IC Command Payload
5	Bluetooth IC Command Header	Bluetooth IC Command Payload

## 7.2: Header of RFID Reader Commands

### Downlink commands

Header is:

A7	B3/E6	Payload Length (1 byte)	C2	82	37	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------------	----	----	----	----------------------------	----------------------------

Payload is RFID reader commands. Please refer to Chapter 8 and Appendix A for definitions of RFID reader commands payload.

### Uplink replies/notifications

(Notifications include tag returns)

Header is:

A7	B3/E6	Payload Length (1 byte)	C2	82	9E	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------------	----	----	----	----------------------------	----------------------------

Payload is RFID reader command replies and tag data. Please refer to Chapter 8 and Appendix A for definitions of payload.



## 7.3: Header of Barcode Reader Commands

### Downlink commands

Header is:

A7	B3/E6	Payload Length (1 byte)	6A	82	37	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------	----	----	----	----------------------	----------------------

Payload is barcode reader commands. Please refer to Chapter 9 for definitions of payload.

### Uplink replies/notifications

(Notifications include barcode returns)

Header is:

A7	B3/E6	Payload Length (1 byte)	6A	82	9E	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------	----	----	----	----------------------	----------------------

Payload is barcode command replies and barcodes. Please refer to Chapter 9 for definitions of payload.

## 7.4: Header of Special Notifications

### Downlink commands

Header is:

A7	B3/E6	Payload Length (1 byte)	D9	82	37	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------------	----	----	----	----------------------------	----------------------------

Payload is special notification. Please refer to Chapter 10 for definitions of payload.

### Uplink replies/notifications

(Notifications include special notifications)

Header is:

A7	B3/E6	Payload Length (1 byte)	D9	82	9E	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------------	----	----	----	----------------------------	----------------------------

Payload is special notifications. Please refer to Chapter 10 for definitions of payload.

## 7.5: Header of Silicon Lab IC Commands

### Downlink commands

Header is:

A7	B3/E6	Payload Length (1 byte)	E8	82	37	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------------	----	----	----	----------------------------	----------------------------

Payload is Silicon Lab IC commands. Please refer to Chapter 11 for definitions of payload.

### Uplink replies/notifications

Header is:

A7	B3/E6	Payload Length (1 byte)	E8	82	9E	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------------	----	----	----	----------------------------	----------------------------

Payload is Silicon Lab ICs notifications. Please refer to Chapter 11 for definitions of payload.

## 7.6: Header of Bluetooth IC Commands

### Downlink commands

Header is:

A7	B3/E6	Payload Length (1 byte)	5F	82	37	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------------	----	----	----	----------------------------	----------------------------

Payload is Bluetooth IC commands. Please refer to Chapter 12 for definitions of payload.

### Uplink replies/notifications

Header is:

A7	B3/E6	Payload Length (1 byte)	5F	82	9E	CRC of packet byte 1	CRC of packet byte 2
----	-------	-------------------------------	----	----	----	----------------------------	----------------------------

Payload is Bluetooth IC replies and notifications. Please refer to Chapter 12 for definitions of payload.

# Chapter 8: Payload – RFID Reader

## Commands

This chapter describes the payload for RFID reader operations. There are various downlink commands and uplink replies, plus uplink tag data and reader status notifications.

Payload Format:

Event code (2 bytes)	Data
-------------------------	------

Reply Format:

Event code (2 bytes)	Status
-------------------------	--------

## 8.1 Downlink Payload

Payload Field

Event code	Data length in byte(s)	Event/Data Description.
0x8000	0	RFID reader power on.
0x8001	0	RFID reader power off.
0x8002	Depends on payload length	RFID firmware command data. See Appendix A

Reply Payload Field

Event code	Status length in byte(s)	Status Description.
0x8000	1	Status: 0x00 – Power on success 0xFF – Failure with unknown reason
0x8001	1	Status: 0x00 – Power off success 0xFF – Failure with unknown reason
0x8002	1	Status: 0x00 – Success 0xFF – Failure with unknown reason

## 8.2 Uplink Payload

Event code	Data length in byte(s)	Event/Data Description.
0x8100	Depends on payload length	RFID firmware response data. See Appendix A

Note that uplink payload can be:

- 1) Command reply
- 2) Tag data - when it is tag data, the header reserve byte contains the sequence number

# Chapter 9: Payload – Barcode Reader

## Commands

This chapter describes the payload for Barcode reader operation. The only downlink Barcode commands are barcode power on and power off. Once the barcode reader is power on, any press of the push button at the gun handle will start barcode reading. Any successful barcode capture will then be sent back to the originating source of the commands – either Bluetooth or USB.

Payload Format:

Event code (2 bytes)	Data
-------------------------	------

Reply Format:

Event code (2 bytes)	Status
-------------------------	--------

## 9.1 Downlink Payload

### Payload Field

Event code	Data length in byte(s)	Event/Data Description.
0x9000	0	Barcode reader power on.
0x9001	0	Barcode reader power off.
0x9002	0	Trigger to start a scan
0x9003	1-50	Command data* sent to barcode device
0x9004	3	Vibrator on. Byte 0: Mode: 0 = Normal; 1 = Inventory on; 2 = Barcode good read on. Byte 1-2: On time in ms (all 0's mean forever)
0x9005	0	Vibrator off. Reset all modes in 0x9004

\*for command data of barcode reader, please reference "Serial\_Programming\_Command\_Manual\_V1.3.1" from Newland (barcode module manufacturer)

### Reply Payload Field

Event code	Status length in byte(s)	Status Description.
0x9000	1	Status: 0x00 – Power on success 0xFF – Failure with unknown reason
0x9001	1	Status: 0x00 – Power off success 0xFF – Failure with unknown reason
0x9002	1	Status: 0x00 – Trigger success 0x01 – Failure with barcode not powered on 0x02 – Failure with previous scan not ended 0xFF – Failure with unknown reason
0x9004	1	Status: 0x00 – Vibrator on success 0xFF – Failure with unknown reason
0x9005	1	Status: 0x00 –Vibrator off success 0xFF – Failure with unknown reason



## 9.2 Uplink Payload

Event code	Data length in byte(s)	Event/Data Description.
0x9100	Depends on payload length	Barcode read/reply data.
0x9101	0	Good read

### 9.2.1 Uplink Barcode Payload Format

In the CS108 barcode scanner, the format of uplink “barcode successfully scanned” payload is as follows:

Prefix, Barcode, Suffix

Prefix = Self-prefix + Code ID + AIM ID

Suffix = Self-suffix

Self-prefix = STX, Null, BEL, BS, SI, VT (encoded by ascii table as 020007101713)

Self-suffix = ENQ, SOH, TAB, SO, ETX, EOT (encoded by ascii table as 050111160304)

(you can find the corresponding encoding from ascii table)

Code ID, 1 byte, is a commonly used code that defines the type of barcode.

AIM ID, 3 bytes, is another commonly used code that defines the type of barcode.

Some users like to use Code ID, some users like to use AIM ID, so both of them are included here in the barcode scanner return.

# Chapter 10: Payload – Special

## Notifications

This chapter describes the payload for special notifications.

Downlink Payload Format:

Event code (2 bytes)	Data (variable bytes)
-------------------------	--------------------------

Reply Payload Format:

Event code (2 bytes)	Data (variable bytes)
-------------------------	--------------------------

## 10.1 Downlink Payload

### Payload Field

Event code	Data length in byte(s)	Event/Data Description.
0xA000	0	Get current battery voltage.
0xA001	0	Get current trigger state
0xA002	0	Start battery 5 seconds auto reporting (for BT connection only)
0xA003	0	Stop battery auto reporting
0xA004	1	1=invoke RFID Abort (default); 0= Not invoke Set Trigger Release Not to invoke Abort RFID (default is invoke RFID)
0xA005	0	Get Trigger Release Setting (1 = invoke RFID Abort, 0 = Not invoke RFID Abort)
0xA006	1	Set “Fast Trigger Button Barcode Scanning” mode, once in this mode, just press trigger button the reader will itself send 0x1b, 0x33 command to barcode engine, and release trigger will send 0x1b, 0x30 command to barcode engine 0 = off (default); 1=on
0xA007	0	Get “Fast Trigger Button Barcode Scanning” mode setting
0xA008	1	Start trigger state auto reporting (for BT connection only) 1 byte value = interval in second
0xA009	0	Stop trigger state auto reporting

### Reply Payload Field

Event code	Data length in byte(s)	Data Description.
0xA000	2	Current battery voltage in mV Value 0xFFFF = battery fault
0xA001	1	0 = Released; 1 = Pushed
0xA002	1	0=success; 1=failure
0xA003	1	0=success; 1=failure
0xA004	1	0=success; 1=failure
0xA005	1	1 = invoke RFID Abort, 0 = Not invoke RFID Abort

0xA006	1	0=success; 1=failure
0xA007	1	0 = off; 1=on
0xA008	1	0=success; 1=failure
0xA009	1	0=success; 1=failure

## 10.2 Uplink Payload

Payload Field:

Event code	Data length in byte(s)	Event/Data Description.
0xA100	0	Reserve
0xA101	2	Error code 0x0000 – Wrong header prefix 0x0001 – Payload length too large 0x0002 – Unknown target 0x0003 – Unknown event
0xA102	0	Trigger is pushed
0xA103	0	Trigger is released

# Chapter 11: Payload – Silicon Lab IC

## Commands

This chapter describes the payload for silicon lab IC command.

Payload Format:

Event code (2 bytes)	Data
-------------------------	------

Reply Payload Format:

Event code (2 bytes)	Data
-------------------------	------

## 11.1 Downlink Payload

Payload Field

Event code	Data length in byte(s)	Event/Data Description.
0xB000	0	Get Silicon Lab IC firmware version.
0xB001	118	Image raw data. Auto update when subpart = total number of subparts. Byte 0-1: total number of subparts Byte 2-3: subpart index starting from 1 Byte 4-117: Image raw data
0xB002	118	Bootloader raw data. Auto update when subpart = total number of subparts. Byte 0-1: total number of subparts Byte 2-3: subpart index starting from 1 Byte 4-117: Bootloader raw data
0xB003	0	Set EEPROM write protect (only available for serial number, model and permanent keys, only via USB)
0xB004	1	Get 16-byte serial number. (Null terminated) Byte 0: 0 = Permanent; 1 = Custom
0xB005	17	Set 16-byte serial number. (Null terminated)

		Byte 0: 0 = Permanent; 1 = Custom Byte 1-17: Serial number
0xB006	0	Get 16-byte model. (Null terminated)
0xB007	16	Set 16-byte model. (Null terminated)
0xB008	2	Get key 1. Byte 0: 0 = Permanent; 1 = Custom Byte 1: Length in bytes (16 or 32)
0xB009	2-18 / 34	Set key 1 Byte 0: 0 = Permanent; 1 = Custom Byte 1: Length in bytes (16 or 32) Byte 2-18/34: key value
0xB00A	2	Get key 2. Byte 0: 0 = Permanent; 1 = Custom Byte 1: Length in bytes (16 or 32)
0xB00B	2-18 / 34	Set key 2 Byte 0: 0 = Permanent; 1 = Custom Byte 1: Length in bytes (16 or 32) Byte 2-18/34: key value
0xB00C	0	Reset Silicon Lab

### Reply Payload Field

Event code	Data length in byte(s)	Data Description.
0xB000	3	Byte 0: Major version Byte 1: Minor version Byte 2: Build version
0xB001	1	0x00 – success 0x01 – failure 0x02 – full image received successfully
0xB002	1	0x00 – success 0x01 – failure 0x02 – full bootloader received successfully
0xB003	1	0x00 – success 0x01 – failure
0xB004	16	16-byte serial number (Null terminated)
0xB005	1	0x00 – success 0x01 – failure

0xB006	16	16-byte model (Null terminated)
0xB007	1	0x00 – success 0x01 – failure
0xB008	16 / 32	128 or 256-bit key1
0xB009	1	0x00 – success 0x01 – failure
0xB00A	16 / 32	128 or 256-bit key2
0xB00B	1	0x00 – success 0x01 – failure
0xB00C	1	0x00 – success 0x01 – failure

## 11.2 Uplink Payload

There is no uplink payload yet.

# Chapter 12: Payload – Bluetooth IC

## Commands

This chapter describes the payload for special notifications.

Payload Format:

Event code (2 bytes)	Data
-------------------------	------

Reply Format:

Event code (2 bytes)	Data
-------------------------	------

## 12.1 Downlink Payload

Payload Field

Event code	Data length in byte(s)	Event/Data Description.
0xC000	0	Get Bluetooth IC firmware version.
0xC001	68	Image raw data. Auto update when subpart = total number of subparts. Byte 0-1: total number of subparts Byte 2-3: subpart index starting from 1 Byte 4-67: Image raw data
0xC002	68	Bootloader raw data. Auto update when subpart = total number of subparts. Byte 0-1: total number of subparts Byte 2-3: subpart index starting from 1 Byte 4-67: Bootloader raw data
0xC003	21	Set device name. Fixed length 21 including null-terminate char.
0xC004	0	Get device name
0xC005	0	Force BT disconnection



## Reply Payload Field

Event code	Data length in byte(s)	Data Description.
0xC000	3	Byte 0: Major version (for CS463, 3) Byte 1: Minor version (for CS463, 0) Byte 2: Build version (for CS463, 0)
0xC001	1	0x00 – success (for CS463, 0x00) 0x01 – failure 0x02 – full image received successfully
0xC002	1	0x00 – success (for CS463, 0x00) 0x01 – failure 0x02 – full bootloader received successfully
0xC003	1	0 = Success; 1= Fail (for CS463, 0x00)
0xC004	21	Device name
0xC005	1	0 = Success; 1= Fail (for CS463, 0x00)

## 12.2 Uplink Payload

There is no uplink payload yet.

# Chapter 13: Factory Default Settings

Factory default settings of CS108 includes:

- 1) Device Name (CS108ReaderXXXXXX) where XXXXXX are the last 6 hex numbers of the Bluetooth MAC address

# Appendix A - CSL RFID Firmware

## Command Specification

### A.1 Introduction

To configure and control the RFID module, one has to:

- 1) Understand the differences between low level API and high level API ([Section A.2](#))
- 2) How to read and write MAC firmware registers ([Section A.3](#))
- 3) Know the definition of each MAC firmware registers ([Section A.4](#))
- 4) Use HST\_CMD commands to tell the RFID module to start various operations ([end of Section A.4](#))  
– sending a command to MAC is achieved by writing the command to a special MAC firmware register 0xF000, hence it is described in Section A.2.
- 5) Know the definition of R2000 registers. Need to use MAC Bypass Read and Write HST\_CMD 0x05 and 0x06 to access them. ([Section A.5](#))
- 6) Know the definition of OEM registers. Need to use OEM Write and Read HST\_CMD 0x02 and 0x03 to access them. ([Section A.6](#))
- 7) Receive Command State Packets to monitor RFID module operation status and receive RFID tag info ([Section A.7](#))
- 8) Use Control Command to stop operations. ([Section A.8](#))

There are 3 types of registers that the application needs to access (read or write):

- 1) MAC registers: these are the most common one to be accessed. Use [REG\\_REQ](#) to access.
- 2) OEM register: only 1 register, 0xA2, is of concern to the application. Use [HST\\_CMD 0x02 and 0x03](#) to access
- 3) R2000 register: only 1 register, 0x0450, is of concern to the application. Use MAC Bypass [HST\\_CMD 0x05 and 0x06](#) to access.

## A.2 High Level API vs Low Level API

Due to historical reasons, the CSL Unified API byte protocol has a high level API version and a low level API version. The differences are only in Byte 0: Packet Version (**pkt\_ver**) byte, Byte 1: Access Flag (**access\_flg**) byte and Byte 2 & 3: the Packet Type (**pkt\_type**) word of each packet. These are specially remarked in the following pages so that the differences can be easily understood. The reader itself actually accepts both high level and low level API seamlessly. However, the responses going back to the PC host can be either high level or low level API depending on customer's own choice (which is usually affected by what they historically used).

All CS108 and CS463 reader are set default to high level API ex-factory. Therefore for new developer it is simpler just to assume High Level API. Although the examples in Appendix C are shown in Low Level API format, it is quite easy to convert them to High Level API. Just refer to the remaining Appendices A.3-8 to know how to convert between them.

Whether the reader behaves as a High Level API reader or a Low Level API reader is defined in an OEM location 0x000000A2: 0 means high level; 1 means low level. Default is high level. One can change that by writing to OEM address using HST\_CMD 0x00000002 (need to first write the desired OEM address to MAC register OEM\_ADDR; write the value you want to set to MAC register OEM\_DATA, and then after that write the value 0x00000002 to MAC register HST\_CMD (0xF000))

As mentioned above, whether a packet is high level or low level is mainly determined by **pkt\_ver**, which is the first byte of the byte stream communication packet (so called byte 0):

<b>Downlink/Uplink</b>	<b>pkt_ver</b>	<b>Packet Category/Types</b>
<b>Downlink</b>	<b>70(low level)</b> <b>00/01(high level – read/write)</b>	MAC Register Read or Write Request, For high level, 00 means Read, 01 means Write For low level, Read and Write are determined by access_flg, the second byte: 00 in second byte means read, 01 in second byte means write
<b>Uplink</b>	<b>70(low level)</b> <b>00(high level)</b>	MAC Register Read Response
<b>Uplink</b>	<b>02(low level)</b> <b>01(high level)</b>	R2000 IC command state packets such as command begin, command end, etc.
<b>Uplink</b>	<b>03 (same for high level or low level)</b>	inventory packet
<b>Uplink</b>	<b>04 (same for high level or low level)</b>	compact mode inventory packet
<b>Uplink</b>	<b>01 (same for high level or low level)</b>	Tag Access packet (actually tag access packet is also a command state packet – we need to describe this in a separate row because tag access packets for low level API and high level API BOTH use pkt_ver = 01)
<b>Downlink and Uplink</b>	<b>40 (same for high level or low level)</b>	Abort command
<b>Uplink</b>	<b>01 (same for high level or low level)</b>	OEM Register Read Response and R2000 Register Read Response packet

Note that pkt\_ver for the last 5 rows are the same whether the reader is set to high level or low level.

Note that although **pkt\_ver = 01** appears in 3 uplink rows above, but their **pkt\_type** are different. So they can be differentiated.

So, if you are tracking the packets in Wireshark, and just looking at the byte 0, packet version, and the logic is

**Downlink (from PC to reader):**

Whenever you see 70/00/01:                      MAC register read or write request and response

**Uplink (from reader to PC):**

Whenever you see 70/00:                      MAC register read response

Whenever you see 02/01/03/04:              Command state packets (e.g., inventory packet, compact mode inventory packet, tag access packets)

**Uplink or Downlink:**

Whenever you see 40:                      Abort command and Abort Response command

To further differentiate them, you need to also look at byte 1: flag; and byte 2 &3: packet type.

In summary, the process of decoding basic category of packets requires knowledge of Byte 0, 1, 2 and 3. Byte 0 is packet version, Byte 1 is flag, Byte 2 and 3 are packet type.

## A.3 MAC Register Access Packet (Downlink)

### REG\_REQ packet format

These packets are sent from Host to firmware to access (read or write) a firmware register. This is a Downlink packet.

Byte Offset(s)	Name	Description
0	Header (pkt_ver)	0x70 for low level (for high level API, Byte 0 is used to define write or read: write = 1, read = 0)
1	access_flg	For low level: 0 = read 1 = write (for high level API, Byte 1 is always 0)
3:2	reg_addr	16bit address of the register to be written. <b>Note that Byte 3 is the high byte, Byte 2 is the low byte !!!!! Example, an address of 0xF000 will become 00F0 in the packet.*</b>
7:4	reg_data	For <i>write register</i> this is the 32 bits of data to be written. For <i>read register</i> set to zeros. <b>Again, note the reverse order of byte placement. Byte 7 is the first byte, byte 6 is second byte, and so on.*</b>

**\*Note that for RFID firmware commands and replies, when a field is more than 1 byte, it is always REVERSELY POPULATED. For example, if a field contains 2 bytes, then the first byte is the LOW BYTE, and second byte is the HIGH BYTE. If a field contains 3 bytes, then the first byte is the LSB, second byte is the mid byte, third byte is the MSB. Likewise for fields containing 4 bytes, 5 bytes, etc. This is called byte-swapped.**

## REG\_RESP packet format

These are response packets returned to the host while the firmware is not in command state. **This response packet only comes back when the operation is Read register. There is no response if the operation is write register.** This is an uplink packet.

Byte Offset(s)	Name	Description
0	header (pkt_ver)	0x70 for Low Level (0x00 for High Level)
1	reserved	Read as zero
3:2	reg_addr	16 bit address of the register to be read (echo back for confirmation and easy tracking)
7:4	reg_data	These are the 4 bytes of register data you want to read



## A.4 MAC Registers Description

The following are the relevant MAC registers that can be read or written to according to A.1.

**Reminder: please note that for RFID firmware commands and replies, when a field is more than 1 byte, it is always REVERSELY POPULATED (byte-swapped). For example, if a field contains 2 bytes, then the first byte is the LOW BYTE, and second byte is the HIGH BYTE. If a field contains 3 bytes, then the first byte is the LSB, second byte is the mid byte, third byte is the MSB. Likewise for fields containing 4 bytes, 5 bytes, etc.**

### FIRMWARE\_VER

Holds the firmware version information

reg\_addr = 0x0000

Bits	Name	Description
11:0	Patch Version	12 bit Patch version number of firmware image
23:12	Minor Version	12 bit Minor version number of firmware image
31:24	Major Version	8 bit Major version number of firmware image

### MAC\_Error

Holds the firmware version information

reg\_addr = 0x0005

Bits	Name	Description
15:0	Code	MAC Error Code, See Appendix B for MAC error codes definitions
31:16	Reserved	Reserved

## HST\_CMNDIAGS

This register configures common diagnostics

reg\_addr = 0x0201

power up defaults = 0x00000210

Bits	Name	Description
0:0	en_diags	Enable diagnostics packets
1:1	en_status	Enable status packets
2:2	en_sycle	Enable protocol module end of cycle statistics packets only
3:3	en_round	Enable protocol module end of round statistics packets only
4:4	en_invresp	Enable sending of the inventory response packets during access commands
5:5	en_cwdiags	Enable sending of carrier diagnostics packets only
6:6	en_accessdiags	Enable sending of access diagnostics packets
7:7	en_randomcwdiags	Enable sending of Random CW diagnostics packets only
8:8	en_csmdiags	Enable sending of Core State Machine diagnostics packets only
9:9	en_commandactive	Enable periodic output of Command Active packet for long running commands, in the absence of any other host interface output
31:10	Reserved_31_10	Reserved

Note that Bit 9, en\_commandactive, is a nice feature so that during inventory, if there is no RFID tag in front of reader, the reader will still send out, every 3 seconds, a Command Active packet, so that the user will know the reader is still in inventory mode.

## IMPINJ\_EXTENSIONS (TagFocus and FastID)

This register provides support for Impinj specific extensions

reg\_addr = 0x0203

Bits	Name	Description
3:0	blockwrite_mode	<p>Determines the maximum number of words to write per BlockWrite transaction with the tag.</p> <p>0 = Auto-detect (Default). One or two word BlockWrite will be determined automatically.</p> <p>1 = Force one word BlockWrite. Unconditionally use one word BlockWrites in all cases.</p> <p>2 = Force two word BlockWrite. Unconditionally use two word BlockWrites in all cases. A protocol error will occur if the tags in the field do not support this feature.</p> <p>3-15 = Reserved for future use</p>
4:4	tag_focus	<p>If this feature is enabled, once a tag has been singulated it will remain out of the tag population (the tag's session 1 inventoried flag remains in B state) until the inventory operation is complete.</p> <p>This feature is only effective in conjunction with other inventory controls Session=S1, target=A only (toggle disabled)</p> <p>0=disabled</p> <p>1=enabled</p>
5:5	fast_id	<p>If this feature is enabled and an Impinj tag is in the field, then the 6-word TID will be returned along with the EPC when the tag is singulated. The first 5 bits of PC will be automatically adjusted to account for the length increase.</p> <p>0=disabled</p> <p>1=enabled</p>
31:6	reserved	reserved

## PROTSCH\_SMIDX

Indicate the FCC or ETSI country set. (**read only**)

reg\_addr = 0x0300

Bits	Name	Description
31:0	country	0 = FCC 1 = ETSI

## PROTSCH\_SMCFG

Specify the configuration bits for the current protocol state machine.

reg\_addr = 0x0301

FCC Country:

Bits	Name	Description
31:0	reserved	read / write as zero

ETSI Country:

Bits	Name	Description
0	en_lbt	enable LBT. 0 = off 1 = on
1	en_scan	enable Scan Mode - if an interferer is detected by the LBT logic the next enabled frequency channel is selected. LBT will also be performed on the new channel before use. The logic is repeated until a channel is found with no interferer. <b>Recommended to set 1 when LBT is on.</b>
2	reserved	read / write as zero
3	dis_eoff	"disable early cw off" when no tags found. In other words, don't turn off CW when tags aren't immediately found. This is useful for applications with fast moving tag populations (ex. conveyor applications). Where the device may need to continuously look for tags without lowering the carrier before the TX ON time limit is reached.
7:4	reserved	read / write as zero
15:8	lbt_rssi_thr	8 bit rssi threshold value. During LBT operation, RSSI readings >= to this threshold indicate to the LBT logic that an interferer is present. <b>Recommended value is 0x2F.</b>
31:16	reserved	read / write as zero

## PROTSCH\_SMCFG\_SEL

Used to select the desired bank for the PROTSCH\_SMCFG register

reg\_addr = 0x0304

Bits	Name	Description
31:0	index	For ETSI, set to 1 before changing PROTSCH_SMCFG register.

## ANT\_CYCLES

Specify the number of times the enabled logical antenna port should be cycled through in order to complete protocol command execution. In a single antenna cycle, the firmware scans all available logical antennas starting at #0.

reg\_addr = 0x0700

Bits	Name	Description
15:0	cycles	The number of antenna cycles to be completed for command execution 0x0001 = once cycle through enabled antennas (Non-continuous mode) 0xFFFF = cycle forever until a ABORT is received (Continuous mode)
17:16	mode	Antenna sequence mode <b>(for CS468 only)</b> 0x00 = normal mode (antenna switch from port 0 to port 15 sequentially) 0xX1 = sequence mode (antenna switch based on the sequence stored in OEM table address 0xA7-0xAC) 0x1X = smart check mode (quick switch to next enabled antenna if no tag is detected.)
23:18	sequence size	Sequence size of sequence mode. Max = 48. <b>(for CS468 only)</b>
24	freq_agile	Frequency Agile mode. 0 = disable 1 = enable
31:25	reserved	reserved - read / write as 0

## ANT\_PORT\_SEL

Select the antenna port.

reg\_addr = 0x0701

Bits	Name	Description
31:0	port_idx	zero based index of descriptor to select for port.  Valid values are 0x0 through 0xF.

## ANT\_PORT\_CFG

Sets bits in this register to indicate the configuration of the logical antenna specified by the ANT\_PORT\_SEL register.

reg\_addr = 0x0702

Bit	Name	Description
0	Enable	1=logical antenna enabled 0=logical antenna disabled
1	Inv_mode	0 = Global mode (use global parameters). CS108 must set as 0. 1 = Local mode (use port dedicated parameters)
3:2	Inv_algo	Inventory algorithm to use.
7:4	StartQ	Starting Q value. 0 - 15
8	Profile_mode	0 = Global mode (use last CURRENT_PROFILE parameters). CS108 must set as 0. 1 = Local mode (use port dedicated parameters)
12:9	Profile	0-3
13	Freq_mode	0 = Global mode (use first enabled frequency). CS108 must set as 0. 1 = Local mode (use port dedicated frequency)
19:14	Freq_chan	Frequency channel
20	EAS_enable	1=EAS detection enabled 0=EAS detection disabled
31:21	RFU	reserved - read / write as 0

## ANT\_PORT\_DWELL

Set the dwell time for the logical antenna selected by ANT\_PORT\_SEL.

reg\_addr = 0x0705

Bits	Name	Description
31:0	time_ms	number of milliseconds to communicate on this antenna during a given Antenna Cycle 0x00000000 indicates that dwell time should not be used.

## ANT\_PORT\_POWER

Set the output power for the logical antenna selected by ANT\_PORT\_SEL.

reg\_addr = 0x0706

Bits	Name	Description
31:0	power	power in steps of 0.1dBm (Maximum value is 300)

## ANT\_PORT\_INV\_CNT

Set how many inventory rounds should be executed for the logical antenna selected by ANT\_PORT\_SEL.

reg\_addr = 0x0707

Bits	Name	Description
31:0	count	number of inventory rounds for current port 0x00000000 indicates that inventory round count should not be used.

**Power up defaults for each bank (bank selected using ANT\_PORT\_SEL register):**

ANT_PORT_SEL	ANT_PORT_CFG	ANT_PORT_DWELL	ANT_PORT_POWER	ANT_PORT_INV_CNT
0x0	0x00000001	0x000007D0	0x0000012C	0x00002000
0x1	0x00000000	0x000007D0	0x0000012C	0x00002000
0x2	0x00000000	0x000007D0	0x0000012C	0x00002000
0x3	0x00000000	0x000007D0	0x0000012C	0x00002000
0x4	0x00000000	0x000007D0	0x0000012C	0x00002000
0x5	0x00000000	0x000007D0	0x0000012C	0x00002000
0x6	0x00000000	0x000007D0	0x0000012C	0x00002000
0x7	0x00000000	0x000007D0	0x0000012C	0x00002000
0x8	0x00000000	0x000007D0	0x0000012C	0x00002000
0x9	0x00000000	0x000007D0	0x0000012C	0x00002000
0xA	0x00000000	0x000007D0	0x0000012C	0x00002000
0xB	0x00000000	0x000007D0	0x0000012C	0x00002000
0xC	0x00000000	0x000007D0	0x0000012C	0x00002000
0xD	0x00000000	0x000007D0	0x0000012C	0x00002000
0xE	0x00000000	0x000007D0	0x0000012C	0x00002000
0xF	0x00000000	0x000007D0	0x0000012C	0x00002000

## TAGMSK\_DESC\_SEL

Write this register to select which *Select* descriptor and corresponding mask register set to access.

Total of 8 selects can be **AND** together.

reg\_addr = 0x0800

Power up default = 0x00000000

Bit	Name	Description / Values
2:0	desc_idx	Index of the Select descriptor to be used.
31:3	reserved	reserved - read / write as zero



## TAGMSK\_DESC\_CFG

Specify the parameters for a *Select* operation. When an inventory operation is performed and the tag select feature is enabled, these descriptors will be scanned. Each descriptor that has its enable bit set will cause a select command to be issued prior to the inventory.

reg\_addr = 0x0801

Power up default = 0x00000000

Bit	Name	Description / Values
0	Enable	1 descriptor enabled, 0 descriptor disabled
3:1	sel_target	<i>Target</i> value for <i>Select</i> operations. 0 = Inventoried(S0); 1 = Inventoried(S1); 2 = Inventoried(S2); 3 = Inventoried(S3); 4 = SL; ( <b>Recommend to set this value</b> )
6:4	sel_action	what action to perform on <i>inventories</i> or <i>SL</i> flags as indicated to tags during <i>Select</i> operation. <b>Recommend to set zero.</b>
7:7	trunk_en	truncate enable bit value - passed to underlying <i>Select</i> operation.
15:8	delay	CW hold time in ms after <i>Select</i> command.
31:16	reserved	reserved - write as zero

### sel\_action:

sel_action	Tag Matching	Tag Not-Matching
000	assert <b>SL</b> or <b>inventoried</b> → A	deassert <b>SL</b> or <b>inventoried</b> → B
001	assert <b>SL</b> or <b>inventoried</b> → A	do nothing
010	do nothing	deassert <b>SL</b> or <b>inventoried</b> → B
011	negate <b>SL</b> or (A → B, B → A)	do nothing
100	deassert <b>SL</b> or <b>inventoried</b> → B	assert <b>SL</b> or <b>inventoried</b> → A
101	deassert <b>SL</b> or <b>inventoried</b> → B	do nothing
110	do nothing	assert <b>SL</b> or <b>inventoried</b> → A
111	do nothing	negate <b>SL</b> or (A → B, B → A)

## TAGMSK\_BANK

Specify which memory bank is applied to during *Select*.

18000-6C memory banks list as follows:

0x0 - Reserved

0x1 - EPC

0x2 - TID

0x3 - USER

reg\_addr = 0x0802

Power up default = 0x00000000

Bits	Name	Description
1:0	mask_bank	Mask bank selection
31:2	reserved	reserved - read / write as 0

## TAGMSK\_PTR

Specify the bit offset in tag memory at which the configured mask will be applied during *Select*.

Used in conjunction with TAGMSK\_BANK, TAGMSKPTR and TAGMSK[0,31] registers.

reg\_addr = 0x0803

Power up default = 0x00000000

Bits	Name	Description
31:0	ptr	offset to be used with the mask. 0x00000000 - 0xFFFFFFFF supported

## TAGMSK\_LEN

Specify the number of valid bits in TAGMSK[0,31] registers that will be sent with the underlying *Select* command.

reg\_addr = 0x0804

Power up default = 0x00000000

Bits	Name	Description
7:0	length	number of valid bits the combined TAGMSK[0,31] registers.
31:8	reserved	reserved - read / write as 0

## TAGMSK\_0\_3

Write the tag mask data to the TAGMSK\_xx registers for use with subsequent *Select* commands.

reg\_addr = 0x0805

Power up default = 0x00000000

When the R1000 firmware executes a *Select* command it scans the tag mask registers as a byte array, starting with the low order byte of the TAGMSK\_0\_3 and continues until it scans the number of bits specified in the TAGMSK\_LEN register. For a non-8 bit aligned tag mask the final bits are left justified (high order bits) in the final byte. Consider the following example:

Tag mask = {0xFF, 0x07}

Length = 0xB

1) Host writes 0x000070FF to register TAGMSK\_0\_3

2) Host writes 0x0000000B to register TAGMSK\_LEN

3) When firmware executes a *Select* operation it will build the mask by scanning 0xB bits from TAGMSK\_0\_3 starting at the low byte. The final, non-8 bit aligned mask bits, are taken from the second byte of TAGMSK\_0\_3 using the logic : final\_bits = (8-( TAGMSK\_LEN % 8)).

Bits	Name	Description
31:8	reserved	reserved - read / write as 0

## TAGMSK\_4\_7

See TAGMSK\_0\_3 for details.

reg\_addr = 0x0806

Power up default = 0x00000000

## TAGMSK\_8\_11

See TAGMSK\_0\_3 for details.

reg\_addr = 0x0807

Power up default = 0x00000000

## TAGMSK\_12\_15

See TAGMSK\_0\_3 for details.

reg\_addr = 0x0808

Power up default = 0x00000000

## TAGMSK\_16\_19

See TAGMSK\_0\_3 for details.

reg\_addr = 0x0809

Power up default = 0x00000000

## TAGMSK\_20\_23

See TAGMSK\_0\_3 for details.

reg\_addr = 0x080A

Power up default = 0x00000000

## TAGMSK\_24\_27

See TAGMSK\_0\_3 for details.

reg\_addr = 0x080B

Power up default = 0x00000000

## TAGMSK\_28\_31

See TAGMSK\_0\_3 for details.

reg\_addr = 0x080C

Power up default = 0x00000000

## QUERY\_CFG

Configure parameters used in underlying Query and inventory operations.

reg\_addr = 0x0900

Bit	Name	Description / Values
3:0	reserved	Reserved - read / write as zero
4	query_target	Starting <i>Target</i> argument (A or B) to underlying <i>Query</i> 0 = A; 1 = B.
6:5	query_session	<i>Session</i> argument to underlying <i>Query</i> 0 = S0; 1 = S1; 2 = S2; 3 = S3.
8:7	query_sel	<i>Select</i> argument to underlying <i>Query</i> 0 = All; 1 = All; 2 = ~SL; 3 = SL; <b>Recommend to Set 0 for inventory operation.</b> <b>Recommend to Set 3 for tag select operation. Reference to INV_CFG and TAGMSK_DESC_CFG.</b>
31:9	reserved	reserved - read / write as zero

## INV\_CFG

Inventory configuration. Configure parameters used in underlying inventory operations

reg\_addr = 0x0901

Power up default = 0x00000001

Bit	Name	Description / Values
5:0	Inv_algo	Inventory algorithm to use.
13:6	match_rep	Stop after "N" tags inventoried (zero indicates no stop)
14	tag_sel	1 = enable tag select prior to inventory, read, write, lock or kill. 0 = no select issued.
15	disable_inventory	Do not run inventory.
17:16	tag_read	0 = no tag read issued. 1 = enable read 1 bank after inventory. 2 = enable read 2 banks after inventory
18	crc_err_read	0 = disable crc error read 1 = enable crc error read
19	QT_mode	0 = disable QT temporary read private EPC 1 = enable QT temporary read private EPC
25:20	tag_delay	Time delay for each tag in ms (use to reduce tag rate), 6 bits, binary. For Bluetooth normal mode inventory (see byte 26 below), this should be set to 30. For Bluetooth compact mode inventory (see byte 26 below), this should be set to 0 or some small values, most desirably less than 7. For USB, this should be set to 0.
26	inv_mode	0 = normal mode 1 = compact mode. See Inventory-Response Packet (Compact mode)
27	brand_ID	0 = disable brand ID for Ucode8 1 = enable brand ID for Ucode8*
31:28	reserved	reserved - read / write as zero

\*NXP Ucode 8 has a new feature where a Brand ID can be saved ex-factory for brand owners of the world (e.g. LV). This Brand ID is 16 bits long (2 bytes, 4 hex numbers). When you select EPC bank with offset=204h, length=1, mask=1b, then the inventory results in a LONGER number (PC becomes bigger!!) with the last 4 hex numbers being the Brand ID. The actual data is scrambled and will only be descrambled if you set bit 27 of INV\_CFG register to 1.

## INV\_SEL

Select which set of algorithm parameter registers to access.

reg\_addr = 0x0902

Bits	Name	Description
31:0	desc_idx	zero based index of descriptor to access 0 through 3.

## INV\_ALG\_PARM\_0

The algorithm that will be used for the next Inventory command is specified by the register INV\_CFG. The definition of each register varies depending on the algorithm chosen. For instance, if you wish to set the parameters for algorithm 1, then set INV\_SEL to 1 and load PARM as specified.

reg\_addr = 0x0903

Power up defaults for each bank (bank selected using HST\_INV\_SEL register):

Algorithm #	Power up Value
0x0	0x00000004
0x3	0x000040F4

The following tables indicate how each algorithm interprets the value for its bank:

### Algorithm 0 (Fixed Q)

Bits	Name	Description
3:0	Q value	Q value that algorithm 0 will use when running an inventory

### Algorithm 3 (Dynamic Q Algorithm 3)

Bits	Name	Description
3:0	StartQ	Starting Q value
7:4	MaxQ	Maximum Q value
11:8	MinQ	Minimum Q Value
19:12	Tmult	Threshold multiplier. This is a fixed point fraction with the decimal point between bit 2 and 3. The field looks like bbbbbb.bb which allows fractional values of 1/2, 1/4 and 3/4.

## INV\_ALG\_PARM\_1

The algorithm that will be used for the next Inventory command is specified by the register INV\_CFG. The definition of each register varies depending on the algorithm chosen. For instance, if you wish to set the

parameters for algorithm 1, then set INV\_SEL to 1 and load PARM as specified.

reg\_addr = 0x0904

Power up defaults for each bank (bank selected using HST\_INV\_SEL register):

Algorithm #	Power up Value
0x0	0x00000000
0x1	0x00000000
0x2	0x00000000
0x3	0x00000000

The following tables indicate how each algorithm interprets the value for its bank:

### All Algorithms

Bits	Name	Description
7:0	Retry	Number of times to retry a query / query rep sequence for the session/target before flipping the target or exiting. For example, if q is 2 then there will be one query and 4 query reps for each retry.

## INV\_ALG\_PARM\_2

The algorithm that will be used for the next Inventory command is specified by the register INV\_CFG. The definition of each register varies depending on the algorithm chosen. For instance, if you wish to set the parameters for algorithm 1, then set INV\_SEL to 1 and load PARM as specified.

reg\_addr = 0x0905

Power up defaults for each bank (bank selected using HST\_INV\_SEL register):

Algorithm #	Power up Value
0x0	0x00000003
0x1	0x00000001
0x2	0x00000001
0x3	0x00000001

The following tables indicate how each algorithm interprets the value for its bank:

### Algorithm 0

Bits	Name	Description
------	------	-------------

0	Toggle	If set to one, the target will flip from A to B or B to A after all rounds have been run on the current target. This is done after no tags have been read if continuous mode is selected and all retry's have been run.
1	RunTillZero	Continue running inventory rounds until a round is completed without reading any tags.

**Algorithm 1-3**

Bits	Name	Description
0	Toggle	If set to one, the target will flip from A to B or B to A after all rounds have been run on the current target. This is done after no tags have been read if continuous mode is selected and all retry's have been run.

**TAGACC\_DESC\_CFG**

Tag access configuration register

reg\_addr = 0x0A01

Power up default = 0x00000000

Bits	Name	Description
0	Verify	Verify after write. <b>Must set as 1 if writing tag.</b>
5:1	Retry	Number of time to retry the operation if failed
31:6	Reserved	reserved - read / write as zero

**TAGACC\_BANK**

Specify which memory bank is to be accessed during the various *access* operations.

18000-6C memory banks list follows:

0x0 - Reserved

0x1 - EPC

0x2 - TID

0x3 - USER

reg\_addr = 0x0A02

Power up default = 0x00000001

Bits	Name	Description
1:0	acc_bank	Tag memory bank selection for Tag Accesses
3:2	acc_bank2	Tag memory bank selection for second bank read. <b>Used for INV_CFG tag_read = 2 only. Otherwise must be set to 0.</b>
31:4	reserved	reserved - read / write as 0



## TAGACC\_PTR

Specify the offset (16 bit words) in tag memory for tag accesses (read and write).

reg\_addr = 0x0A03

Power up default = 0x00000002

For INV\_CFG tag\_read = 0.

Bits	Name	Description
31:0	ptr	offset to be used in tag accesses. 0x00000000 - 0xFFFFFFFF supported

For INV\_CFG tag\_read = 1 or 2

Bits	Name	Description
15:0	ptr	offset to be used in first bank tag accesses. 0x0000 – 0xFFFF supported
31:16	ptr2	offset to be used in second bank tag accesses. 0x0000 – 0xFFFF supported

## TAGACC\_CNT

Write this register to specify the number of 16 bit words that should be accessed when issuing read or write commands. Note that the value zero which is used to specify the maximum number of words contained in a bank is not supported at this time.

Reg\_addr = 0x0A04

Power up default = 0x00000001

Bits	Name	Description
7:0	length	number of 16 bit words to read/write – maximum value: Read: 255, Write:32
15:8	length2	number of 16 bit words to read second bank. <b>Used for INV_CFG tag_read = 2 only. Otherwise must be set to 0.</b>
31:16	reserved	reserved - read / write as 0

## TAGACC\_LOCKCFG

Write this register prior to issuing the “Lock” command (0x12).

reg\_addr = 0x0A05

Power up default = 0x00000000

Bits	Name	Description
9:0	action	lock command action bits
19:10	mask	lock command mask bits

31:20	reserved	reserved - write as 0
-------	----------	-----------------------

Mask defines which bank to execute the locking, action defines what type of lock or unlock commands to carry out. For details please reference EPC Air Interface document.

## TAGACC\_ACCPWD

Set this register to the access password that is specified in the reserved tag memory prior to issuing access commands.

reg\_addr = 0x0A06

Power up default = 0x00000000

Bits	Name	Description
15:0	AWORD0	Access password value - word 0
31:16	AWORD1	Access password value - word 1

## TAGACC\_KILLPWD

Set this register to the desired password value prior to issuing the "Kill" command (0x13).

reg\_addr = 0x0A07

Power up default = 0x00000000

Bits	Name	Description
15:0	KWORD0	Kill password value - word 0
31:16	KWORD1	Kill password value - word 1

## TAGWRDAT\_SEL

Used to access the tag write buffer.

The buffer is set up as a 16 register array.

reg\_addr = 0x0A08

Power up default = 0x00000000

Bits	Name	Description
2:0	bank	Bank select value of TAGWRDAT. Can be 0 or 1 only. 0 for TAGWRDAT_0 - TAGWRDAT_15; 1 for TAGWRDAT_16 - TAGWRDAT_31; (same reg_addr as TAGWRDAT_0 - TAGWRDAT_15)
31:3	reserved	Reserved, write as zero

For Block Write,

Bits	Name	Description
2:0	bank	Bank select value of TAGWRDAT. Can be 0 to 7 0 TAGWRDAT_0 to TAGWRDAT_15; 1 TAGWRDAT_16 to TAGWRDAT_31; 2 TAGWRDAT_32 to TAGWRDAT_47 3 TAGWRDAT_48 to TAGWRDAT_63 4 TAGWRDAT_64 to TAGWRDAT_79 5 TAGWRDAT_80 to TAGWRDAT_95 6 TAGWRDAT_96 to TAGWRDAT_111 7 TAGWRDAT_112 to TAGWRDAT_127
31:3	reserved	Reserved, write as zero

## TAGWRDAT\_0 - TAGWRDAT\_15

Set these registers to valid data prior to issuing the "Write" command (0x11).

reg\_addr = 0x0A09 - 0x0A18

Power up default = 0x00000000

Bits	Name	Description
15:0	data_word	data to be written to during underlying <i>write</i> access.
31:16	offset	This value plus the value of TAGACC_PTR gives the word offset from start of the bank specified by TAGACC_BANK to write the data.

For Block Write,

Bits	Name	Description
15:0	Word 0	Tag Block Write Command - Low order data to be written to during underlying write access
31:16	Word 1	Tag Block Write Command - High order data to be written to during underlying write access.

## CURRENT\_PROFILE

Specify the selector of the active profile. The active profile may be changed by writing the selector of the desired link profile, and executing the "update the link profile" command (command 0x19).

reg\_addr = 0x0B60

Power up default = 0x00000002

Bits	Name	Description
7:0	Profile selector	valid values are 0 through 3* (see Appendix C for Profile descriptions)
31:8	RFU	Reserved for future use

## FREQCH\_SEL

Select the frequency channel.

reg\_addr = 0x0C01

Bits	Name	Description
31:0	Freq Channel	Frequency Channel selector - valid values are 0 through 49. Note that FCC scans all 50 channels; ETSI only scans channels 0 to 14.

## FREQCH\_CFG

Indicate which frequency channels are enabled. The frequency channel selected by writing the desired channel number to the FREQCH\_SEL register, and then writing the desired bits to this register.

When a channel is disabled, the firmware will skip over it when selecting the next channel to use. Note that, for the ETSI, that the first enabled channel is the only channel used until *Scan Mode* is enabled in the PROTSCH\_SMCFG register.

reg\_addr = 0x0C02

Bits	Name	Description
0	status	1=channel enabled, 0=channel disabled
31:1	reserved	Reserved, write as zero

## FREQCH\_DESC\_PLLDIVMULT

Configure the frequency for the corresponding frequency channel. The host must write HST\_RFTC\_FRQCH\_SEL to the desired channel number and then configure the bits in this register. As the protocol scheduler iterates through enabled frequency channels, it passes the value in this register to the RF Control module so that when carrier is turned on it will be at the desired frequency.

Reg\_addr = 0x0C03

Bits	Name	Description
15:0	MULTRAT	PLL multiply ratio
24:16	DIVRAT	PLL divide ratio
31:25	RFU	Reserved for future use

The relevant equations are:

$$\text{CWFrequency} = 24\text{MHz} * \text{MULTRAT} / (\text{DIVRAT} * 4)$$

## AUTHENTICATE\_CFG

Authenticate command configuration .

reg\_addr = 0x0F00

Bits	Name	Description
0	SenRep	0: store 1: send
1	IncRepLen	0: Omit length from reply 1: Include length in reply
9:2	CSI	CSI value write as zero
21:10	Length	length of message in bits (must be multiple of 8)
31:22	reserved	reserved - write as 0

## AUTHENTICATE\_MSG0

Address: 0x0F01

Bit	Name	Description / Values
31:0	Message	Upper 32 bits of message

## AUTHENTICATE\_MSG1

Address: 0x0F02

Bit	Name	Description / Values
31:0	Message	2 <sup>nd</sup> 32 bits of message

## AUTHENTICATE\_MSG2

Address: 0x0F03

Bit	Name	Description / Values
31:0	Message	3 <sup>rd</sup> 32 bits of message

## AUTHENTICATE\_MSG3

Address: 0x0F04

Bit	Name	Description / Values
31:0	Message	Lower 32 bits of message

## READBUFFER\_PTR

ReadBuffer command starting address .

reg\_addr = 0x0A03

Bits	Name	Description
31:0	prt	16-bit word starting address

## READBUFFER\_LEN

ReadBuffer command length .

reg\_addr = 0x0A04

Bits	Name	Description
11:0	length	Number of bits to read (max 256 bits)
31:12	reserved	reserved - write as 0

## UNTRACEABLE\_CFG

Untraceable command configuration .

reg\_addr = 0x0F05

Bits	Name	Description
1:0	Range	00: normal 01: toggle temporarily 10: reduced 11: RFU
2	User	0: view 1: hide
4:3	TID	00: hide none 01: hide some 10: hide all 11: RFU
10:5	EPC	MSB (show/hide): 0: show memory above EPC 1: hide memory above EPC 5 LSBs (length): New EPC length field (new <b>L</b> bits)
11	U	0: Deassert <b>U</b> in XPC_W1 1: Assert <b>U</b> in XPC_W1

31:12	reserved	reserved - write as 0
-------	----------	-----------------------

## INV\_CYCLE\_DELAY

Delay time between inventory cycle.

reg\_addr = 0x0F0F

Bits	Name	Description
31:0	cycle_delay	Time delay in-between each inventory cycle in ms (use to reduce tag rate) The values should be between 0 to 2000. 0 means fastest tag rate.

**Reminder:** When writing this register, please be careful in the packetizing process that the Least Significant Byte (LSB) should be placed first, Most Significant Byte (MSB) should be placed last. This is the so called “Reversely Populated” principle as mentioned at the beginning of this chapter.

## HST\_CMNDIAGS

Common diagnostics configuration register

reg\_addr = 0x0201

Power Up default = 0x00000210

For Vibrator Operation, MUST CHANGE to 0x00000010 (disable Command Active Packet !!!)

Bits	Name	Description
0:0	en_diags	Enable diagnostics packets
1:1	en_status	Enable status packets
2:2	en_cycle	Enable protocol module end of cycle statistics packets only
3:3	en_round	Enable protocol module end of round statistics packets only
4:4	en_invresp	Enable sending of the inventory response packets during access commands
5:5	en_cwdiags	Enable sending of carrier diagnostics packets only
6:6	en_accessdiags	Enable sending of access diagnostic packets
7:7	en_randomcwdiags	Enable sending of random CW diagnostics packets only
8:8	en_csmdiags	Enable sending of core state machine diagnostics packets only



9:9	en_commandactive	Enable periodic output of Command Active packet for long running commands, in the absence of any other host interface output
31:10	Reserved_31_10	Reserved

## AMBIENTTEMP

Ambient temperature of the reader.

reg\_addr = 0x0B06

Bits	Name	Description
31:0	temperature	Ambient temperature, in units of degrees Celsius. This is a two's complement representation of a signed 32-bit integer. If the temperature is positive, i.e., MSB is 0, then the value is directly the temperature.

**Reminder:** Right after power up, this MAC register has value 0 if you read it. You must at least do 1 round of inventory before this register has meaningful value.

## PATEMP

RF power amplifier temperature.

reg\_addr = 0x0B0A

Bits	Name	Description
31:0	temperature	RF power amplifier temperature, in units of degrees Celsius. This is a two's complement representation of a signed 32-bit integer. If the temperature is positive, i.e., MSB is 0, then the value is directly the temperature.

**Reminder:** Right after power up, this MAC register has value 0 if you read it. You must at least do 1 round of inventory before this register has meaningful value.

## HST\_PWRMGMT

Power Management Control Register

reg\_addr = 0x0200

Bits	Name	Description
2:0	Mode_set	Power management mode set bits 0x0 = normal mode (power up default) 0x1 = low power stand-by mode 0x2 – 0x7 = reserved, read/write as 0
31:3	Reserved_31_3	Reserved

**After setting this MAC register, you must send an HST command: 0x14 to the MAC firmware to activate this power management mode change!!!**

**If you do not send the HST command 0x14, this register change has no effect and the power management mode will not change!!!!**

## FM13DT160\_CMDCFGPAR

Cmd Cfg parameter

reg\_addr = 0x0117

Bit	Name	Description / Values
31:0	cmd_cfg	Store the cmd cfg parameter for the FM13DT160 command

## FM13DT160\_REGADDRPAR

Reg addr parameter

reg\_addr = 0x0118

Bit	Name	Description / Values
15:0	Reg addr	Store the reg addr parameter for the FM13DT160 command

**FM13DT160\_WRITEPAR**

Write para parameter

reg\_addr = 0x0119

Bit	Name	Description / Values
15:0	Write para	Store the write para parameter for the FM13DT160 command

**FM13DT160\_PWDPAR**

Password parameter

reg\_addr = 0x011A

Bit	Name	Description / Values
31:0	Password	Store the Password parameter for the FM13DT160 command

**FM13DT160\_STOBLOCKADDPAR**

Store Block Address parameter

reg\_addr = 0x011B

Bit	Name	Description / Values
7:0	Store Block Address	Store Block Address parameter for the FM13DT160 Get Temperature command

**FM13DT160\_STARTADDRPAR**

Start addr parameter

reg\_addr = 0x011C

Bit	Name	Description / Values
15:0	Start addr	Start addr parameter for the FM13DT160 Read/Write Memory command

**FM13DT160\_READWRITELENPAR**

Read/Write len parameter

reg\_addr = 0x011D

Bit	Name	Description / Values
15:0	Len	Read/Write length parameter for the FM13DT160 Read/Write Memory command (Read: Max 512; Write: Max 4)

**FM13DT160\_DATAPAR**

Data parameter

reg\_addr = 0x011E

Bit	Name	Description / Values
31:0	Data	Data parameter for the FM13DT160 Write Memory command

## EM4325\_CFG

Data parameter

reg\_addr = 0x011F

Bits	Name	Description
0	SendUID	Send UID bit 0: Do not send UID 1: Do send UID
1	NewSample	New Sample bit 0: Get last sample 1: Get new sample
31:2	Reserved	Reserved

## HST\_CMD (Downlink)

HST\_CMD is host command register. Host software writes one of the supported command values to this register and causes the reader firmware to execute the new command. Once you write the values into this register, the reader will execute the command. This is a “register write to enable action” methodology to send a command to another IC.

**reg\_addr = 0xF000**

Bits	Name	Description
31:0	Command	Command value

## Command Set Summary

When the host wishes to issue a command to the reader firmware, it writes a *command value* to the HST\_CMD register. The host is then required to successively read *Response Packets*. First it should receive a *Command begin Packet*. Then it should receive *Response Packets*. At the end it should receive a *Command End Packet*. The *Command End Packet* signifies completion of the command and, if the reader firmware is not in an error state, it is then ready to execute more commands. If there is an error, the command end packet will contain that and the host can analyze it and check against the MAC error table in Appendix B.

Command Value (for HST_CMD register 0xF000)	Command Mnemonic	Description
0x00000002	CMD_WROEM	Write OEM register (set OEM_ADDR and OEM_DATA before you do this)
0x00000003	CMD_RDOEM	Read OEM register (set OEM_ADDR before you do this)
0x00000005	CMD_MBPRDREG	MAC Bypass Register Read - directly read R2000 register (set MAC_BYPASS_ADDR before you do this)
0x00000006	CMD_MBPWRREG	MAC Bypass Register Write - directly write R2000 register (set MAC_BYPASS_ADDR and MAC_BYPASS_DATA before you do this)
0x0000000F	CMD_18K6CINV	Start Inventory
0x00000010	CMD_18K6CREAD	Read
0x00000011	CMD_18K6CWRITE	Write

0x00000012	CMD_18K6CLOCK	Lock
0x00000013	CMD_18K6CKILL	Kill
0x00000014	CMD_SETPWRMGMTCFG	Set Power Management Configuration
0x00000019	CMD_UPDATELINKPROFILE	Change the link profile
0x0000001F	CMD_18K6CBLOCKWRITE	BlockWrite
0x00000026	CUSTOMNXPCHANGE EAS	ChangeEAS for G2XM and G2XL IC of NXP
0x0000003B	CUSTOMMEMGETSENSOR DATA	Get sensor data from EM4325
0x00000050	CMD_18K6AUTHENTICATE	Authenticate
0x00000051	CMD_18K6READBUFFER	ReadBuffer
0x00000052	CMD_18K6UNTRACEABLE	Untraceable
0x00000053	CUSTOMMMFM13DTREADMEMORY	FM13DT160 Read Memory
0x00000054	CUSTOMMMFM13DTWRITEMEMORY	FM13DT160 Write Memory
0x00000055	CUSTOMMMFM13DTAUTH	FM13DT160 Auth
0x00000056	CUSTOMMMFM13DTGETTEMP	FM13DT160 Get Temperature
0x00000057	CUSTOMMMFM13DTSTARTLOG	FM13DT160 Start Logging
0x00000058	CUSTOMMMFM13DTSTOPLOG	FM13DT160 Stop Logging
0x00000059	CUSTOMMMFM13DTWRITEREG	FM13DT160 Write Reg
0x0000005A	CUSTOMMMFM13DTREADREG	FM13DT160 Read Reg
0x0000005B	CUSTOMMMFM13DTDEEPSLEEP	FM13DT160 Deep Sleep
0x0000005C	CUSTOMMMFM13DTOPMODECHK	FM13DT160 Op Mode Check
0x0000005D	CUSTOMMMFM13DTINITIALREGFILE	FM13DT160 Initial Regfile
0x0000005E	CUSTOMMMFM13DTLEDCTRL	FM13DT160 Led Ctrl

Note that whenever an HST\_CMD is sent using MAC Register 0xF0000, then the reader will execute the requested actions, and send back first a Command Begin packet, then certain number of operation-response packets, and then ending with a Command End packet.

## OEM\_ADDR

Sets this address register to indicate which address in the OEM table should be accessed with a subsequent read OEM command. You must set this value BEFORE you execute HST\_CMD 0x02 or 0x03

Address: 0x0500

Bits	Name	Description
15:0	OEM_ADDR	The address (or index) of the OEM data to be accessed
31:16	Reserved	Reserved

## OEM\_DATA

Use this register to save the actual data you want to write to a certain OEM address as defined by OEM\_ADDR above. You must set this value BEFORE you execute HST\_CMD 0x02

Address: 0x0501

Bits	Name	Description
31:0	OEM_DATA	The OEM data to be written

## MAC\_BYPASS\_ADDR

Sets this address register to indicate which address in the OEM table should be accessed with a subsequent read OEM command. You must set this value BEFORE you execute HST\_CMD 0x02 or 0x03

Address: 0x0400

Bits	Name	Description
11:0	R2000_ADDR	The address (or index) of the R2000 Register to be accessed
31:12	Reserved	Reserved

## MAC\_BYPASS\_DATA

Use this register to save the actual data you want to write to a certain OEM address as defined by OEM\_ADDR above. You must set this value BEFORE you execute HST\_CMD 0x02

Address: 0x0401

Bits	Name	Description
------	------	-------------



31:0	OEM_DATA	The OEM data to be written
------	----------	----------------------------

## A.5 R2000 Registers Description

This R2000 register contains RF High Compression Mode setting, RF LNA Gain, IF LNA Gain, AGC Gain Range for application configure. Use **MAC Bypass Write and Read HST\_CMD (i.e., 0x05 and 0x06)** to access this R2000 register.

### ANA\_RX\_GAIN\_NORM

Configures RF LNA Gain, RF LNA High Compression Mode ON/OFF, IF LNA Gain

IF AGC Gain

Address: 0x0450

450	ANA_RX_GAIN_NORM	R/W	8	rflna_high_comp_norm	h0000	<p>The rflna_gain setting generates the following RF-LNA gains:</p> <ul style="list-style-type: none"> <li>• 0 = 1 dB</li> <li>• 2 = 7 dB</li> <li>• 3 = 13 dB</li> </ul> <p>The high compression mode of the RFLNA can only be combined with the 1 and 7 dB gain settings.</p> <p>The iflna_gain setting generates the following IF-LNA gains:</p> <ul style="list-style-type: none"> <li>• 0 = 24 dB</li> <li>• 1 = 18 dB</li> <li>• 3 = 12 dB</li> <li>• 7 = 6 dB</li> </ul> <p>The ifagc_gain setting generates the following AGC gain values:</p> <ul style="list-style-type: none"> <li>• 0 = -12 dB</li> <li>• 4 = -6 dB</li> <li>• 6 = 0 dB</li> <li>• 7 = 6 dB</li> </ul>
			7:6	rflna_gain_norm		
			5:3	iflna_gain_norm		
			2:0	ifagc_gain_norm		

## A.6 OEM Registers Description

### HIGHLOWLEVEL

Holds the high low level configuration of the reader

reg\_addr = 0x000000A2

Bits	Name	Description
31:0	High Low Level	0 is high level; 1 is low level

### OEMCFGADDR\_AUTO\_LOW\_POWER\_CONFIG

The configuration register for Auto Low Power mechanism when MAC firmware boots.

reg\_addr = 0x000003DA

Bits	Name	Description
0:0	boot	Enable Low Power standby when firmware boots.
31:1	Reserved	Reserved

## A.7 Command State Packet (Uplink)

Command state packets come back from RFID reader module to the host processor to report important information.

### Command-Begin Packet

The command-begin packet indicates the start of a sequence of packets for an ISO 18000-6C tag-protocol operation (i.e. inventory etc.). The type of command executed by the RFID radio module determines which data packets appear, and in what order they appear, between the command-begin/end packet pair.

**Table: Command-Begin Packet Fields**

Byte Offset(s)	Name	Description
0	pkt_ver	0x02 for low level (or 0x01 for high level)
1	flags	0 = Non-continuous mode 1 = Continuous mode
3:2	pkt_type	0x8000 for low level (or 0x0000 for high level)
5:4	pkt_len	Length = 0x0002
7:6	reserved	Reserved. Read as zero.
11:8	command	The command that initiated the packet sequence: 0x0000000F - Start Inventory
15:12	ms_ctr	Firmware millisecond counter when the operation started.

## Command-End Packet

The command-end packet indicates the end of sequence of packets for an ISO 18000-6C tag-protocol operation. A command-end packet is always used to terminate a packet sequence regardless of the fact that a tag-access operation is completed successfully or not. If not successful, a possible reason is a MAC error has happened. The MAC error type is stored in Byte 12 and 13 (in reverse order) of the Command End Packet Fields.

**Table: Command-End Packet Fields**

Byte Offset(s)	Name	Description
0	<b>pkt_ver</b>	<b>0x02 for low level</b> <b>(0x01 for high level)</b>
1	flags	Reserved. Read as zero.
3:2	pkt_type	0x8001 for low level (or 0x0001 for high level)
5:4	pkt_len	Length = 0x0002
7:6	reserved	Reserved. Read as zero.
11:8	ms_ctr	Firmware millisecond counter when the operation ended.
13:12	status	The completion status of the operation. Values are: 0x0000 – Success. See Appendix B for RFID Firmware Error Codes table.
14	error_port	If status is non-zero, this reports the error port. Otherwise read as zero.
15	reserved	Reserved. Read as zero.

## Inventory-Response Packet

The ISO 18000-6C inventory-response packet contains the data a tag backscatters during the tag-singulation phase. This data is generated for tag inventories as well as ISO 18000-6C tag-access operations (i.e. read, write, etc.). Assuming a valid CRC, the data contains the PC+EPC+CRC16 received during the singulation of a tag.

**Table: Inventory-Response Packet Fields**

Byte Offset(s)	Name	Description																					
0	<b>pkt_ver</b>	<b>Currently = 0x03 (for both high level and low level)</b>																					
1	flags	<div>Flags</div> <table> <tr> <th>Bits</th><th>Name</th><th>Description</th></tr> <tr> <td>0:0</td><td>CRC</td><td>0 = Valid CRC 1 = Invalid CRC</td></tr> <tr> <td>1:1</td><td>ANA_CTL Format</td><td>0 = R1000 Format 1 = R2000 Format</td></tr> <tr> <td>3:2</td><td>FastID Data</td><td>0 = No FastID data in packet 1 = Monza 4 FastID data included (12 bytes) 2-3 = Reserved</td></tr> <tr> <td>4:4</td><td>Phase Data</td><td>0 = No Phase Data (0) 1 = Phase Data available</td></tr> <tr> <td>5:5</td><td>Reserved</td><td>Reserved</td></tr> <tr> <td>7:6</td><td>Pad Bytes</td><td>Number of padding bytes to Inv_Data to force 32-bit packet boundary. Padding bytes are always at the end of the packet.</td></tr> </table>	Bits	Name	Description	0:0	CRC	0 = Valid CRC 1 = Invalid CRC	1:1	ANA_CTL Format	0 = R1000 Format 1 = R2000 Format	3:2	FastID Data	0 = No FastID data in packet 1 = Monza 4 FastID data included (12 bytes) 2-3 = Reserved	4:4	Phase Data	0 = No Phase Data (0) 1 = Phase Data available	5:5	Reserved	Reserved	7:6	Pad Bytes	Number of padding bytes to Inv_Data to force 32-bit packet boundary. Padding bytes are always at the end of the packet.
Bits	Name	Description																					
0:0	CRC	0 = Valid CRC 1 = Invalid CRC																					
1:1	ANA_CTL Format	0 = R1000 Format 1 = R2000 Format																					
3:2	FastID Data	0 = No FastID data in packet 1 = Monza 4 FastID data included (12 bytes) 2-3 = Reserved																					
4:4	Phase Data	0 = No Phase Data (0) 1 = Phase Data available																					
5:5	Reserved	Reserved																					
7:6	Pad Bytes	Number of padding bytes to Inv_Data to force 32-bit packet boundary. Padding bytes are always at the end of the packet.																					
3:2	pkt_type	0x8005 for low level (or 0x0005 for high level)																					
5:4	pkt_len	Length = variable (greater than or equal to 3)																					
7:6	reserved	Reserved. Read as zero.																					
11:8	ms_ctr	Firmware millisecond counter tag was inventoried.																					

12	wb_rssi	<p>Wideband Receive Signal Strength Indicator.</p> <p>Value Conversion to dB:</p> <p>Mantissa = 3:0</p> <p>Exponent = 7:4</p> <p>Mantissa_Size = 4</p> $20 * \log_{10} (2^{\text{Exponent}} * (1 + \text{Mantissa} / 2^{\text{Mantissa\_Size}}))$ <p>Example: Value 0x48</p> <p>Mantissa = 8</p> <p>Exponent = 4</p> $20 * \log ( 2^4 * ( 1 + 8 / 2^4 ) ) = 27.60$
13	nb_rssi	<p>Narrowband Receive Signal Strength Indicator.</p> <p>Value Conversion to dB:</p> <p>Mantissa = 2:0</p> <p>Exponent = 7:3</p> <p>Mantissa_Size = 3</p> $20 * \log_{10} (2^{\text{Exponent}} * (1 + \text{Mantissa} / 2^{\text{Mantissa\_Size}}))$ <p>Example: Value 0x48</p> <p>Mantissa = 0</p> <p>Exponent = 9</p> $20 * \log ( 2^9 * ( 1 + 0 / 2^3 ) ) = 54.19$
14	phase	<p>The Phase Data Bits 6:0 represents signed phase value at the time the EPC is received. Bit 7 is reserved and always 0. Phase is not available on R1000. For R2000, only bit 0 to bit 5 are used. Bit 6 is discarded. Phase is only available for profiles where the following is true: <math>8 * 2^{\text{Miller-Number}} / \text{BLF} \geq 20.0 \text{ us}</math>. (BLF is backscatter Link frequency)</p> <p>Phase in Radian = &lt;bit 0 to bit 5&gt; x <math>2 \times \text{Pie} / 128</math></p> <p>Phase in Degree = Phase in Radian x <math>180 / \text{Pie}</math></p>
15	chidx	Current channel index.
16	data1_count	DATA1 16 bit word length
17	data2_count	DATA2 16 bit word length
19:18	port	Current antenna port.
n:20	inv_data	The data that was backscattered by the tag (i.e. PC + EPC + CRC16) during tag singulation. The data is presented in the same format as it is transmitted over the air from the tag to the RFID radio module – i.e. the

		<p>data has not been changed to match the endianness of the host processor. The total length of this field (in bytes) can be determined by the following formula:</p> $((\text{pkt\_len} - 3) * 4) - ((\text{flags} \gg 6) \& 3)$ <p>If tag_read in INV_CFG is 1, the backscattered data is in format PC+EPC+DATA1+CRC16. EPC length is <math>((\text{PC} \gg 11) * 2)</math></p> <p>If tag_read in INV_CFG is 2, the backscattered data is in format PC+EPC+DATA1+DATA2+CRC16. EPC length is <math>((\text{PC} \gg 11) * 2)</math></p>
--	--	---



## Inventory-Response Packet (Compact mode)

This is Inventory-Response packet when compact mode is set in INV\_CFG. Note the important difference marked in bold red.

**Table: Inventory-Response Packet (Compact mode) Fields**

Byte Offset(s)	Name	Description						
0	pkt_ver	Currently = 0x04 (for both high level and low level)						
1	flags	<div>Flags<table><tr><th>Bit(s)</th><th>Description</th></tr><tr><td>0</td><td>0 = no CRC error 1 = CRC error</td></tr><tr><td>7:1</td><td>Reserved. Read as zero.</td></tr></table></div>	Bit(s)	Description	0	0 = no CRC error 1 = CRC error	7:1	Reserved. Read as zero.
Bit(s)	Description							
0	0 = no CRC error 1 = CRC error							
7:1	Reserved. Read as zero.							
3:2	pkt_type	0x8005 for low level (or 0x0005 for high level)						
5:4	pkt_len	Payload length in byte (!!!!!!!)						
6	Antenna Port #	Port, indexing from 0 to 15. For CS463, 0 is Port 1, 1 is Port 2, 2 is Port 3, 3 is Port 4. All the payloads in n:8 below are for this antenna port only.						
7	Reserved	Reserved. Read as zero						
n:8	payload	<div>The payload data includes one or multiple tag data. Each tag data is in format: PC(2 bytes)+EPC(EPC length)+NB_RSSI(1 byte). EPC length is ((PC &gt;&gt; 11) * 2).  Narrowband Receive Signal Strength Indicator (NB_RSSI). Value Conversion to dB: Mantissa = 2:0 Exponent = 7:3 Mantissa_Size = 3 20 * log10 (2^Exponent * (1 + Mantissa / 2^Mantissa_Size)) Example: Value 0x48 Mantissa = 0 Exponent = 9 20 * log ( 2^9 * ( 1 + 0 / 2^3 ) ) = 54.19</div>						

## Tag-Access Packet

The ISO 18000-6C tag-access packet indicates the result of the tag-access command upon the ISO 18000-6C tag. Valid tag access commands are as follows:

- Read
- Write
- Kill
- Lock

If a tag operation is simply limited to an inventory operation, ISO 18000-6C tag operation packets will not appear in the packet sequence.

**Table: Tag-Access Packet Fields**

Byte Offset(s)	Name	Description														
0	pkt_ver	Currently = 0x01 (for both high level and low level)														
1	flags	<div>Flags</div> <table><tr><th>Bit(s)</th><th>Description</th></tr><tr><td>0</td><td>Error flag: 0 = Access operation succeeded 1 = An error occurred. If one of the following error-specific bit fields does not indicate an error, the error code appears in the data field.</td></tr><tr><td>1</td><td>Tag backscatter error flag: 0 = Tag did not backscatter an error. 1 = Tag backscattered an error. See error_code field.</td></tr><tr><td>2</td><td>ACK timeout flag: 0 = Tag responded within timeout. 1 = Tag failed to respond within timeout.</td></tr><tr><td>3</td><td>CRC invalid flag: 0 = CRC was valid 1 = CRC was invalid</td></tr><tr><td>5:4</td><td>Reserved. Read as zero.</td></tr><tr><td>7:6</td><td>Number of padding bytes added to end of tag access data to force packet to end on 32-bit boundary.</td></tr></table>	Bit(s)	Description	0	Error flag: 0 = Access operation succeeded 1 = An error occurred. If one of the following error-specific bit fields does not indicate an error, the error code appears in the data field.	1	Tag backscatter error flag: 0 = Tag did not backscatter an error. 1 = Tag backscattered an error. See error_code field.	2	ACK timeout flag: 0 = Tag responded within timeout. 1 = Tag failed to respond within timeout.	3	CRC invalid flag: 0 = CRC was valid 1 = CRC was invalid	5:4	Reserved. Read as zero.	7:6	Number of padding bytes added to end of tag access data to force packet to end on 32-bit boundary.
Bit(s)	Description															
0	Error flag: 0 = Access operation succeeded 1 = An error occurred. If one of the following error-specific bit fields does not indicate an error, the error code appears in the data field.															
1	Tag backscatter error flag: 0 = Tag did not backscatter an error. 1 = Tag backscattered an error. See error_code field.															
2	ACK timeout flag: 0 = Tag responded within timeout. 1 = Tag failed to respond within timeout.															
3	CRC invalid flag: 0 = CRC was valid 1 = CRC was invalid															
5:4	Reserved. Read as zero.															
7:6	Number of padding bytes added to end of tag access data to force packet to end on 32-bit boundary.															
3:2	pkt_type	Packet type value = 0x0006 (for both high level and low level)														
5:4	pkt_len	Length = variable (greater than or equal to 3)														
7:6	reserved	Reserved. Read as zero.														
11:8	ms_ctr	Firmware millisecond counter tag was inventoried.														

12	command	<p>ISO 18000-6C access command:</p> <p>0xC2 – Read</p> <p>0xC3 – Write</p> <p>0xC4 – Kill</p> <p>0xC5 – Lock</p> <p>0xC7 – Block Write</p> <p>0x04 – EAS</p>
13	error_code	<p>If the tag backscattered an error (i.e. the tag backscatter error flag is set), this value is the error code that the tag backscattered. Values are:</p> <p>0x00 – general error (catch-all for errors not covered by codes)</p> <p>0x03 – specified memory location does not exist or the PC value is not supported by the tag</p> <p>0x04 – specified memory location is locked and/or permalocked and is not writeable</p> <p>0x0B – tag has insufficient power to perform the memory write</p> <p>0x0F – tag does not support error-specific codes</p>
15:14	port	Current antenna port.
19:16	reserved	Reserved. Read as zero.
n:20	data	<p>If there were no errors, and the ISO 18000-6C tag-access operation was a read (i.e. the command field contains 0xC2), this field contains the data that was read from the specified tag memory bank. The data should be treated as a sequence of bytes and is presented in the same format as it is transmitted over the air from the tag to the radio module – i.e. the data has not been changed to match the endianness of the host Firmware processor. The length of this field can be determined by the following formula:</p> $((pkt\_len - 3) * 4) - ((flags \gg 6) \& 3)$ <p>If there were no errors, and the ISO 18000-6C tag-access operation was anything besides a read, then this field will not be present.</p> <p>If the error flag is set, and none of the error-specific bits are set in the flags field, this field contains a 32-bit error code. The error code can be one of the following:</p> <p>0x0000 = No error</p> <p>0x0001 = Handle mismatch</p> <p>0x0002 = CRC error on tag response</p> <p>0x0003 = No tag reply</p>

		<p>0x0004 = Invalid password</p> <p>0x0005 = Zero kill password</p> <p>0x0006 = Tag lost</p> <p>0x0007 = Command format error</p> <p>0x0008 = Read count invalid</p> <p>0x0009 = Out of retries</p> <p>0xFFFF = Operation failed</p>
--	--	--

## Antenna-Cycle-End Packet

The antenna-cycle-end packet indicates the end of one iteration through all enabled antennas.

**Table: Antenna-Cycle-End Packet Fields**

Byte Offset(s)	Name	Description
0	<b>pkt_ver</b>	<b>0x02 for low level (or 0x01 for high level)</b>
1	flags	Reserved. Read as zero.
3:2	pkt_type	0x8007 for low level (or 0x0007 for high level)
5:4	pkt_len	Length = 0x0000
7:6	reserved	Reserved. Read as zero.

## Command Active Packet

The Command Active packet provides an indication that a command is still running, in the absence of any other host interface output from the radio.

For long running commands with minimal packets enabled, this packet lets the host processor/application know that the radio is still operating.

This packet will only be sent during command operations, and only if there are no other host interface packets output for more than 3 seconds, in which case the Command Active packet will be sent on a 3 second interval.

**Table: Command Active Packet Fields**

Byte Offset(s)	Name	Description
0:0	<b>pkt_ver</b>	<b>0x01 for high or low level</b>
1:1	flags	Reserved. Read as zero.
3:2	pkt_type	<b>0x000E for high level or low level</b>
5:4	pkt_len	Length = 0x0001
7:6	Reserved	Reserved
11:8	MS_Ctr	Firmware millisecond counter when antenna usage stopped

## Inventory-Cycle-Begin Packet

The inventory-cycle-begin packet **is normally not sent back**, but only if MAC Error 0x0336 happens. MAC Error 0x0336 means the requested forward power level is not achieved during power ramp. The MAC Error 0x0336 code is in Byte 7:6

**Table: Inventory-Cycle-Begin Packet Fields**

Byte Offset(s)	Name	Description
<b>0</b>	<b>pkt_ver</b>	<b>0x01 for high or low level</b>
1	flags	Reserved. Read as zero.
3:2	pkt_type	<b>0x000A for high level or low level</b>
5:4	pkt_len	Length = 0x0000
7:6	Error Code	MAC Error 0x0336
11:8	MS_Ctr	Firmware millisecond counter when antenna usage stopped

## OEM Registers Read Response Packet

The OEM Register Read Response packet indicates the OEM values in an OEM address, which comes back after an HST\_CMD OEM Read operation.

**Table: OEM Register Read Response Packet Fields**

Byte Offset(s)	Name	Description
0	<b>pkt_ver</b>	<b>0x01 for both low level and high level</b>
1	flags	Reserved. Read as zero.
3:2	pkt_type	Packet type value = 0x3007
5:4	pkt_len	Length = 0x0002
7:6	reserved	Currently = 0xFF
11:8	address	Address of requested OEM configuration register
15:12	Data	Data in requested OEM configuration register

## OEM Register Write Response Packet

No response after OEM register write command other than the command begin and command end response.

Therefore application must execute an OEM Register Read command to confirm the write.

Since OEM registers are flash memory, a good practice is to wait 100 msec after sending the write OEM register command, and then do a read OEM register.



## R2000 Register Read Response Packet

The R2000 Register Read Response packet indicates the values in an R2000 register address, which comes back after an HST\_CMD MAC Bypass Read operation.

**Table: R2000 Register Read Response Packet Fields**

Byte Offset(s)	Name	Description
0	<b>pkt_ver</b>	<b>0x01 for both low level and high level</b>
1	flags	Reserved. Read as zero.
3:2	pkt_type	Packet type value = 0x3005
5:4	pkt_len	Length = 0x0001
7:6	reserved	Currently = 0xFF
9:8	address	Address of requested R2000 configuration register
11:10	Data	Data in requested R2000 register

## R2000 Registers Write Response Packet

No response after R2000 register write command other than the command begin and command end response.

Therefore application must execute an R2000 Register Read command to confirm the write.

## A.8 Control Command (Downlink & Uplink)

There are 2 control commands that are the same for high level or low level API:

Abort (Downlink)

Abort Response (Uplink)

**Byte 0 = pkt\_ver = 40 (for both low level and high level)**

**Byte 1 = 03**

Packet Name	Direction	Command Data	Description
ABORT control command	<b>Downlink</b>	40:03:00:00:00:00:00:00	Control command to abort operation. Explanation: 4003000000000000 means abort operation.
ABORT control command response	<b>Uplink</b>	40:03:bf:fc:bf:fc:bf:fc	Command response of abort operation. 4003bffcbbfcbbfc means abort success

# Appendix B – RFID Firmware Error Codes

The “status” field of the Command End Packet comes back with error code field. If this field is non-zero then there are errors. **Again, this is reversed, so byte 13 is the high byte, byte 12 is the low byte.**

Code (hex number)	Description
0x0102	Set by the USB interface module when an unsupported descriptor <i>TYPE</i> has been requested by the host (i.e. <u>not</u> a device, string, configuration descriptor type. This may be due to compatibility problems with the USB host.
0x0103	Set by the USB interface module when an unsupported device descriptor index has been requested by the Host.
0x0104	Set by the USB interface module when it is unable to transmit the response to a request on USB endpoint 0 (aka control endpoint). This may be due to compatibility or synchronization problems with the USB host.
0x0105	RESERVED
0x0106	Set by the USB interface module when higher level firmware requests an unsupported buffer length. This may be due to a firmware build error or corrupted firmware in flash.
0x0107	This is set by the Host interface module when the underlying physical interface module returns an unknown error code on receive from the host. This may be due to a firmware build issue, corrupted firmware image or corrupted SRAM due to errant Intel® R1000 Firmware code.
0x0108	This is set by the Host interface module when the underlying physical interface module returns an unknown error code on transmit to the Host. This may be due to a firmware build issue, corrupted firmware image or corrupted SRAM due to errant code.
0x0109	This is set when the Host interface code detects that its internal state machine out of sync. This could be due to a corrupted firmware image or corrupted SRAM due to errant Intel® R1000 Firmware code.
0x010A	RESERVED
0x010B	Set by the host interface module when an invalid Intel® R1000 Firmware register read or

	write is attempted (either by the host or internally by the Intel® R1000 Firmware).
0x010C	RESERVED
0x010D	This is set by the host interface module during initialization if it is unable to retrieve USB string descriptors from non-volatile memory (i.e. flash) OEM configuration area. This may be due to a corrupt or unformatted OEM Configuration area. It may also be due to a firmware build issue if the OEM configuration definition is out of sync with the Intel® R1000 Firmware code.
0x010E	This is set when the host attempts to *write* a value to a selector type register that is out of range for that selector.
0x010F	Some firmware tried to send a packet to the host that was too long for the underlying host interface code.
0x0110	Not currently set by Intel® R1000 Firmware.
0x0111	Set by the low level host interface logic if an upper level requests an unsupported raw mode. This may occur if the system is corrupted.
0x0112	Set by the low level host interface logic if a system corrupt occurs and the link manager can not determine the current link state.
0x0113	Set by the low level host interface logic if an unknown / unsupported control command is received from the host. This may occur if the host logic and the Intel® R1000 Firmware logic are out of sync. in terms of the lowest level host interface (UART, USB)
0x0114	This is set if the upper layer host logic attempts to receive data and the lower layer cannot support the buffer size requested. This will happen if the system is corrupted.
0x0115	Set by the low level host interface logic if a control command is received from the host while in raw mode - which is not allowed. This would happen if the host caused the the Intel® R1000 Firmware to enter non-volatile memory update mode, which uses raw mode, and then the host proceeded to issue control commands.
0x0116	Set by the host interface module at boot time if the OEM configuration area is specifying an unsupported host interface.
0x0300	This is set during the PLL lock logic when a bounds check fails while checking the frequency channel configuration registers.
0x0301	This is set if an unsupported frequency hopping mode is detected - during the PLL lock logic.
0x0302	This is set if the PLL fails to lock.
0x0303	This is set when the RFTC module's AUX ADC function times out waiting for an ADC conversion.
0x0304	This is set when the RFTC module times out waiting for Intel® UHF RFID Transceiver

	R1000 to indicate RX or TX filter tuning is complete.
0x0305	This is set when the RFTC module detects ambient temperature sensor indicates too hot.
0x0306	This is set when the RFTC module detects transceiver temperature sensor indicates too hot.
0x0307	This is set when the RFTC module detects PA temperature sensor indicates too hot.
0x0308	This is set when the RFTC module detects that the delta between the PA temperature and the ambient temperature is too great.
0x0309	This is set when the reverse power level is too high as measured by the configured reverse power level threshold in the register set.
0x030A	This is set when an incorrect current gain setting is passed into the IFLNA gain adjustment logic. May indicate corrupted code.
0x030B	Returned by RFTC code when errors occur in transmitting a bit over the RF interface
0x030C	Returned by RFTC code when errors occur in transmitting a buffer of bytes over the RF interface
0x030D	Returned by RFTC code when errors occur in transmitting an “end of transfer” command over the RF interface
0x030E	Returned by RFTC code when errors occur in transmitting a “preamble” command over the RF interface
0x030F	Returned by RFTC code when errors occur in transmitting a “frame-sync” command over the RF interface
0x0310	Indicates that the RF Transceiver (Intel® UHF RFID Transceiver R1000) failed to set expected ISR bits in a timely fashion. Indicates a failure in either the RFTC state machine logic or in the RF Transceiver state machine logic.
0x0311	This is set when invalid link parameters are detected when the filter tuning logic is run.
0x0312	This indicates a failure in either the RFTC state machine logic or in the RF Transceiver state machine logic. This error can only occur if the RF Transceiver (Intel® UHF RFID Transceiver R1000) starts filling its RX FIFO with received data, but fails return the requested number of bits in a timely fashion.
0x0313	Not currently in use. May occur in the future when switching between link profiles if some of the required information is not properly coded in the Intel® R1000 Firmware.
0x0314	May occur when switching between link profiles if the set of Intel® UHF RFID Transceiver R1000 registers that correspond to the requested link profile’s ID_High/ID_Low cannot be found. For the time being, since profiles are compiled into the code, this would be an indication of a corrupted data segment or a build-time fault.
0x0315	Internal error. The error is the direct result of the Intel® R1000 Firmware having to do a “dBm to linear” conversion on a dBm measurement that is outside the range of -99dBm through +45dBm. It the unlikely event that this error is encountered, it is probably the result

	of a faulty RF Peak Detector, a bug in the code that computes the dBm value from the RF Peak Detector ADC reading, or a faulty external PA circuit.
0x0316	If, during RF power-ramping, it is determined that the RF power at the antenna port has momentarily exceeded 35dBm, or has exceeded 33dBm steady-state, this error will be thrown. Encountering this error is often the result of an incorrect calibration of the “gross gains”. See Intel® R1000 Firmware command 0x1B for more information on how to calibrate the system.
0x0317	Internal error that may occur if memory is corrupted.
0x0318	Indicates that the target power (in Intel® R1000 Firmware Virtual Register 0x706) is higher than the maximum allowed output power, which is +33dBm.
0x0319	Indicates that the specified ADC reference voltage is outside the allowed range.
0x031A	Indicates that the measured value of the antenna-sense resistor (reported in the Intel® R1000 Firmware Virtual Register 0x703) exceeds the threshold specified (specified in the Intel® R1000 Firmware Virtual register 0xB12). To determine which antenna was disconnected, the list of enabled antennas will need to be scanned for the one exceeding the threshold (this is done by iterating through all valid selectors in register 0x701 and examining the MAC_ANT_DESC_STAT register at address 0x703).
0x031B	Indicates that the OEMCFG’s HW_OPTIONS_FORMAT value is not recognized by the RFTC subsystem
0x031C	Indicates that the forward power detection option found in OEMCFG’s HW_OPTIONS0 field is not recognized by the RFTC subsystem
0x031D	Indicates that the reverse power detection option found in OEMCFG’s HW_OPTIONS0 field is not recognized by the RFTC subsystem
0x031E	Indicates that the DRM Filter option found in OEMCFG’s HW_OPTIONS0 field is not recognized by the RFTC subsystem
0x031F	Indicates that ambient temperature sensor option found in OEMCFG’s HW_OPTIONS0 field is not recognized by the RFTC subsystem
0x0320	Indicates that PA temperature sensor option found in OEMCFG’s HW_OPTIONS0 field is not recognized by the RFTC subsystem
0x0321	Indicates that transceiver temperature sensor option found in OEMCFG’s HW_OPTIONS0 field is not recognized by the RFTC subsystem
0x0322	Indicates that antenna-sense resistor sensor option found in OEMCFG’s HW_OPTIONS0 field is not recognized by the RFTC subsystem
0x0323	The range specified for the IF LNA AGC gain limits is bad. Either the “min” is higher than the “max”, or the min or max setting is incorrect.
0x0324	When invoking the CMD_LPROF_RDXCVRREG or CMD_LPROF_WRXCVRREG commands, one of the arguments is the selector of a valid link profile (in the Intel® R1000

	Firmware release 1.0 and 1.1, profile selectors 0 through 5 are valid). New link profile selectors cannot be created through these commands, so if a selector outside this range is passed, the RFTC_ERR_LPROFBADSELECTOR error will be generated.
0x0325	The R1000 transceiver has about 421 register addresses between 0x0000 and 0x0500. One of the arguments to the CMD_LPROF_RDXCVRREG or CMD_LPROF_WRXCVRREG commands is the transceiver register address to configure. If the address passed is not a valid R1000 transceiver address, this error will be thrown.
0x0326	Not all valid R1000 transceiver addresses may be configured through the link profiles. The excluded addresses include those registers which are read-only (refer to the R1000 register map), and the indirect address for the R2T command register: 0x0105.
0x0327	Set by the RFTC module if an unsupported RFLNA gain level is requested. This will happen if the protocol scheduler's cycle granular configuration is enabled and the user has specified an unsupported gain level in the HST_PROTSCH_CYCCFG_DESC_ADJ1 banked register
0x0328	Set by the RFTC module if an unsupported IFLNA gain level is requested. This will happen if the protocol scheduler's cycle granular configuration is enabled and the user has specified an unsupported gain level in the HST_PROTSCH_CYCCFG_DESC_ADJ1 banked register
0x0329	Set by the RFTC module if an unsupported AGC/MIXER gain level is requested. This will happen if the protocol scheduler's cycle granular configuration is enabled and the user has specified an unsupported gain level in the HST_PROTSCH_CYCCFG_DESC_ADJ1 banked register
0x032A	Set by the RFTC module if an unsupported compensation option is detected at OEMCFG address 0xA1.
0x0336	This error is not in Command End packet, but in Inventory-Cycle-Begin packet. This error will not cause inventory to stop. This error is generated when the requested forward power level is not achieved during power ramp.





# Appendix C – RFID Reader Firmware

## Command Sequence Examples

### C.1 RFID Reader Initialization Example

**There are 4 steps within Reader Initialization that MUST be done:**

- Step 1: Open Radio
- Step 2: Set RF Power
- Step 3: Set Channel
- Step 4: Set Profile

#### Radio Open

This command ensures the Radio is alive.

Packet Name	Direction	Command Data	Description
ABORT control command	Downlink	40:03:00:00:00:00:00:00	Control command to abort operation. Explanation: 4003
ABORT control command return	Uplink	40:03:bf:fc:bf:fc:bf:fc	Command response of abort operation.

## Set RF Power

Packet Name	Direction	Command Data	Description
REG_REQ*	Downlink	70:01:01:07:00:00:00:00	ANT_PORT_SEL (See appendix A*)
REG_REQ	Downlink	70:01:06:07:2c:01:00:00	ANT_PORT_POWER (See appendix A)

\*REG\_REQ means register requests, which can either be read or write.

\*\*Appendix A describes how to read and write registers and the defined content within each register. For example, ANT\_PORT\_SEL is a register. You can read from it to see what the current setting is. You can write to it to change it to whatever setting you want. The registers contain the configuration for the RFID reader to follow in operation.

## Set Channel

Set channel means setting the frequency channels. Note that this is only needed for readers where the frequency set is NOT locked. **If the version has the frequency channels locked, then there is no need to do this Set Channel operation. For example, if your reader is CS108-2, that means it is FCC and the frequency channels are locked, complying with USA government requirements. Then you do not need to set channel.** Please refer to Appendix K for reader model versions versus frequency channel set.

The following steps are followed:

Step 1: Disable all available channels

Step 2: Enable only the desired channel.

Example of disabling channel 0.

Packet Name	Direction	Command Data	Description
REG_REQ	Downlink	70:01:01:0c:00:00:00:00	FREQCH_SEL (see appendix A)
REG_REQ	Downlink	70:01:02:0c:00:00:00:00	FREQCH_CFG (see appendix A)

Example of enable channel 1.

Packet Name	Direction	Command Data	Description
REG_REQ	Downlink	70:01:01:0c:01:00:00:00	FREQCH_SEL (see appendix A)
REG_REQ	Downlink	70:01:02:0c:01:00:00:00	FREQCH_CFG (see appendix A)

## Set Profile

Packet Name	Direction	Command Data	Description
REG_REQ	Downlink	70:01:60:0b:01:00:00:00	CURRENT_PROFILE (see appendix A) Explanation: 7001 means write to register. Address is 0B60 which is the address that stores the current profile. Profile is to be set to 00000001, which is Profile 1 for CS108.
REG_REQ	Downlink	70:01:00:f0:19:00:00:00	HST_CMD (see appendix A) Explanation: 7001 means write to register. F000 is the HST_CMD address (note the byte reversal). Command is 00000019 (always reversing bytes) which means change profile. So the action is change profile.
Command-Begin Packet	Uplink	02:00:00:80:02:00:00:00:1 9:00:00:00:13:41:00:00	See appendix A. This is the packet that would come back from RFID module. It signals RFID module is starting to do the command above (i.e. change profile).
Command-End Packet	Uplink	02:00:01:80:02:00:00:00:1 6:41:00:00:00:00:00:00	See appendix A. This is the packet that would come back from RFID module. It signals it has completed the command of changing profile.

## C.2 Tag Operations: Inventory Example

**There are 3 steps within Inventory that **MUST** be done:**

Step 1: Set Inventory Parameters

Step 2: Set Inventory Algorithm

Step 3: Start/Stop Inventory

### Set Inventory Parameters

Packet Name	Directions	Command Data	Description
REG_REQ	Downlink	70:01:00:07:ff:ff:00:00	ANT_CYCLES (See appendix A)
REG_REQ	Downlink	70:01:00:09:00:00:00:00	QUERY_CFG (See appendix A)

### Set Inventory Algorithm

REG_REQ	Downlink	70:01:02:09:01:00:00:00	INV_SEL (See appendix A)
REG_REQ	Downlink	70:01:03:09:f7:00:50:03	INV_ALG_PARM_0 (See appendix A)
REG_REQ	Downlink	70:01:01:09:01:00:00:00	INV_CFG (See appendix A)

### Start Inventory

REG_REQ	Downlink	70:01:00:f0:0f:00:00:00	HST_CMD (See appendix A)
Command-Begin Packet	Downlink	02:01:00:80:02:00:00:00:0f:00:00:00:00:61:44:00:00	See appendix A.
Inventory-Response Packet <b>S</b>	Uplink	02:00:05:80:07:00:00:00:73:44:00:00:81:5f:83:06:00:00:00:00:30:00:10:00:00:00:00:00:00:00:00:00:06:87:71:34	See appendix A. (This is the tag info during inventory. The reader will keep on sending tag data inventoried back to the host processor until you stop inventory)

### Stop Inventory

ABORT control command	Downlink	40:03:00:00:00:00:00	This is the command you use to stop inventory.
ABORT control command return	Uplink	40:03:bf:fc:bf:fc:bf:fc	This is the command response you will receive for stopping inventory.

**Note:** After the “ABORT” command to stop inventory, a 2 seconds delay is required for the reader to clear buffer before it can execute another command

## C.3 Tag Operations: Read Example

**There are 4 steps within Read that MUST be done:**

Step 1: Set Inventory Parameters

Step 2: Set Inventory Algorithm. (Assume only 1 tag in front of reader)

Step 3: Select the desired tag.

Step 4: Start reading the tag data.

The followings example illustrates how to read the TID bank.

### Set Inventory Parameters

Packet Name	Command Data	Description
REG_REQ	70:01:00:07:01:00:00:00	ANT_CYCLES (See appendix A)
REG_REQ	70:01:00:09:80:01:00:00	QUERY_CFG (See appendix A)

### Set Inventory Algorithm (Fixed Q = 0)

REG_REQ	70:01:02:09:00:00:00:00	INV_SEL (See appendix A)
REG_REQ	70:01:03:09:00:00:00:00	INV_ALG_PARM_0 (See appendix A)
REG_REQ	70:01:05:09:00:00:00:00	INV_ALG_PARM_2 (See appendix A)

### Select the desired tag (e.g.111122223333444455556666)

REG_REQ	70:01:01:08:09:00:00:00	TAGMSK_DESC_CFG (See appendix A 1)
REG_REQ	70:01:02:08:01:00:00:00	TAGMSK_BANK (See appendix A)
REG_REQ	70:01:03:08:20:00:00:00	TAGMSK_PTR (See appendix A)
REG_REQ	70:01:04:08:60:00:00:00	TAGMSK_LEN (See appendix A)
REG_REQ	70:01:05:08:11:11:22:22	TAGMSK_0_3 (See appendix A)
REG_REQ	70:01:06:08:33:33:44:44	TAGMSK_4_7 (See appendix A)
REG_REQ	70:01:07:08:55:55:66:66	TAGMSK_8_11 (See appendix A)
REG_REQ	70:01:01:09:40:40:00:00	INV_CFG (See appendix A)

## Start reading tag data of the selected tag

REG_REQ	70:01:02:0a:02:00:00:00	TAGACC_BANK (See appendix A)
REG_REQ	70:01:03:0a:00:00:00:00	TAGACC_PTR (See appendix A)
REG_REQ	70:01:04:0a:02:00:00:00	TAGACC_CNT (See appendix A)
REG_REQ	70:01:06:0a:00:00:00:00	TAGACC_ACCPWD (See appendix A)
REG_REQ	70:01:00:f0:10:00:00:00	HST_CMD (See appendix A)
Command-Begin Packet	02:00:00:80:02:00:00:00: 10:00:00:00:d6:8b:00:00	See appendix A.
Inventory-Response Packet	02:00:05:80:07:00:00:00: ec:8b:00:00:00:00:00:00: 00:00:00:00:30:00:11:11: 22:22:33:33:44:44:55:55: 66:66:18:35:	See appendix A. (This is the selected tag)
Tag-Access Packet	01:00:06:00:04:00:00:00: f0:8b:00:00:c2:00:00:00: 00:00:00:00:e2:00:10:50	See appendix A.
Command-End Packet	02:00:01:80:02:00:00:00: f5:8b:00:00:00:00:00:00	See appendix A.

## C.4 Tag Operations: Write Example

**There are 4 steps within Write that MUST be done:**

Step 1: Set Inventory Parameters

Step 2: Set Inventory Algorithm. (Assume only 1 tag in front of reader)

Step 3: Select the desired tag.

Step 4: Start writing the tag data.

The followings example assumes reading TID bank.

### Set Inventory Parameters

Packet Name	Command Data	Description
REG_REQ	70:01:00:07:01:00:00:00	ANT_CYCLES (See appendix A)
REG_REQ	70:01:00:09:80:01:00:00	QUERY_CFG (See appendix A)

### Set Inventory Algorithm (Fixed Q = 0)

REG_REQ	70:01:02:09:00:00:00:00	INV_SEL (See appendix A)
REG_REQ	70:01:03:09:00:00:00:00	INV_ALG_PARM_0 (See appendix A)
REG_REQ	70:01:05:09:00:00:00:00	INV_ALG_PARM_2 (See appendix A)

### Select the desired tag (e.g.111122223333444455556666)

REG_REQ	70:01:01:08:09:00:00:00	TAGMSK_DESC_CFG (See appendix A 1)
REG_REQ	70:01:02:08:01:00:00:00	TAGMSK_BANK (See appendix A)
REG_REQ	70:01:03:08:20:00:00:00	TAGMSK_PTR (See appendix A)
REG_REQ	70:01:04:08:60:00:00:00	TAGMSK_LEN (See appendix A)
REG_REQ	70:01:05:08:11:11:22:22	TAGMSK_0_3 (See appendix A)
REG_REQ	70:01:06:08:33:33:44:44	TAGMSK_4_7 (See appendix A)
REG_REQ	70:01:07:08:55:55:66:66	TAGMSK_8_11 (See appendix A)
REG_REQ	70:01:01:09:40:40:00:00	INV_CFG (See appendix A)



## Start writing tag EPC data

(e.g.000022223333444455556666)

REG_REQ	70:01:01:0a:0f:00:00:00	TAGACC_DESC_CFG (See appendix A)
REG_REQ	70:01:02:0a:01:00:00:00	TAGACC_BANK (See appendix A)
REG_REQ	70:01:03:0a:00:00:00:00	TAGACC_PTR (See appendix A)
REG_REQ	70:01:04:0a:06:00:00:00	TAGACC_CNT (See appendix A)
REG_REQ	70:01:06:0a:00:00:00:00	TAGACC_ACCPWD (See appendix A)
REG_REQ	70:01:09:0a:00:00:02:00	TAGWRDAT_0 (See appendix A)
REG_REQ	70:01:0a:0a:22:22:03:00	TAGWRDAT_1 (See appendix A)
REG_REQ	70:01:0b:0a:33:33:04:00	TAGWRDAT_2 (See appendix A)
REG_REQ	70:01:0c:0a:44:44:05:00	TAGWRDAT_3 (See appendix A)
REG_REQ	70:01:0d:0a:55:55:06:00	TAGWRDAT_4 (See appendix A)
REG_REQ	70:01:0e:0a:66:66:07:00	TAGWRDAT_5 (See appendix A)
REG_REQ	70:01:00:f0:11:00:00:00	HST_CMD (See appendix A)
Command-Begin Packet	02:00:00:80:02:00:00:00: 11:00:00:00:d6:8b:00:00	See appendix A.
Tag-Access Packet	01:00:06:00:03:00:00:00: 10:76:01:00:c3:00:00:00: 00:00:00:00	See appendix A.
Command-End Packet	02:00:01:80:02:00:00:00: f5:8b:00:00:00:00:00:00	See appendix A.

## C.5 Tag Operations: Search Tag Example

**There are 4 steps within Search that MUST be done:**

Step 1: Set Inventory Parameters

Step 2: Set Inventory Algorithm.

Step 3: Select the desired tag.

Step 4: Start searching the tag.

The followings example illustrates how to read the TID bank.

### Set Inventory Parameters

Packet Name	Command Data	Description
REG_REQ	70:01:00:07:01:00:00:00	ANT_CYCLES (See appendix A)
REG_REQ	70:01:00:09:80:01:00:00	QUERY_CFG (See appendix A)

### Set Inventory Algorithm (Fixed Q = 0)

REG_REQ	70:01:02:09:00:00:00:00	INV_SEL (See appendix A)
REG_REQ	70:01:03:09:00:00:00:00	INV_ALG_PARM_0 (See appendix A)
REG_REQ	70:01:05:09:00:00:00:00	INV_ALG_PARM_2 (See appendix A)

### Select the desired tag (e.g.111122223333444455556666)

REG_REQ	70:01:01:08:09:00:00:00	TAGMSK_DESC_CFG (See appendix A 1)
REG_REQ	70:01:02:08:01:00:00:00	TAGMSK_BANK (See appendix A)
REG_REQ	70:01:03:08:20:00:00:00	TAGMSK_PTR (See appendix A)
REG_REQ	70:01:04:08:60:00:00:00	TAGMSK_LEN (See appendix A)
REG_REQ	70:01:05:08:11:11:22:22	TAGMSK_0_3 (See appendix A)
REG_REQ	70:01:06:08:33:33:44:44	TAGMSK_4_7 (See appendix A)
REG_REQ	70:01:07:08:55:55:66:66	TAGMSK_8_11 (See appendix A)
REG_REQ	70:01:01:09:40:40:00:00	INV_CFG (See appendix A)

## Start Searching

REG_REQ	Downlink	70:01:00:f0:0f:00:00:00	HST_CMD (See appendix A)
Command-Begin Packet	Downlink	02:01:00:80:02:00:00:00:0f:00:00:00:06:14:44:00:00	See appendix A.
Inventory-Response Packet	Uplink	02:00:05:80:07:00:00:00:73:44:00:00:81:5f:83:06:00:00:00:00:30:00:11:11:22:22:33:33:44:44:55:55:66:66:71:34	See appendix A. (This is the tag info if tag is found. The reader will keep on sending tag data inventoried back to the host processor until you stop inventory)

## Stop Searching

ABORT control command	Downlink	40:03:00:00:00:00:00:00	This is the command you use to stop inventory.
ABORT control command return	Uplink	40:03:bf:fc:bf:fc:bf:fc	This is the command response you will receive for stopping inventory.

**Note:** After the “ABORT” command to stop inventory, a 2 seconds delay is required for the reader to clear buffer before it can execute another command

## C.6 Tag Operations: Lock Example

**There are 4 steps within Lock that MUST be done:**

Step 1: Set Inventory Parameters

Step 2: Set Inventory Algorithm. (Assume only 1 tag in front of reader)

Step 3: Select the desired tag.

Step 4: Start locking the access password memory.

The followings example assumes locking access password memory.

### Set Inventory Parameters

Packet Name	Command Data	Description
REG_REQ	70:01:00:07:01:00:00:00	ANT_CYCLES (See appendix A)
REG_REQ	70:01:00:09:80:01:00:00	QUERY_CFG (See appendix A)

### Set Inventory Algorithm (Fixed Q = 0)

REG_REQ	70:01:02:09:00:00:00:00	INV_SEL (See appendix A)
REG_REQ	70:01:03:09:00:00:00:00	INV_ALG_PARM_0 (See appendix A)
REG_REQ	70:01:05:09:00:00:00:00	INV_ALG_PARM_2 (See appendix A)

### Select the desired tag (e.g.111122223333444455556666)

REG_REQ	70:01:01:08:09:00:00:00	TAGMSK_DESC_CFG (See appendix A 1)
REG_REQ	70:01:02:08:01:00:00:00	TAGMSK_BANK (See appendix A)
REG_REQ	70:01:03:08:20:00:00:00	TAGMSK_PTR (See appendix A)
REG_REQ	70:01:04:08:60:00:00:00	TAGMSK_LEN (See appendix A)
REG_REQ	70:01:05:08:11:11:22:22	TAGMSK_0_3 (See appendix A)
REG_REQ	70:01:06:08:33:33:44:44	TAGMSK_4_7 (See appendix A)
REG_REQ	70:01:07:08:55:55:66:66	TAGMSK_8_11 (See appendix A)
REG_REQ	70:01:01:09:40:40:00:00	INV_CFG (See appendix A)

Start locking access password memory (assume access  
password is 0x11223344)

REG_REQ	70:01:01:0a:0f:00:00:00	TAGACC_DESC_CFG (See appendix A)
REG_REQ	70:01:05:0a:80:00:03:00	TAGACC_LOCKCFG (See appendix A)
REG_REQ	70:01:06:0a:44:33:22:11	TAGACC_ACCPWD (See appendix A)
REG_REQ	70:01:00:f0:12:00:00:00	HST_CMD, 12 for locking (See appendix A)
Command-Begin Packet	02:00:00:80:02:00:00:00: 11:00:00:00:d6:8b:00:00	See appendix A.
Tag-Access Packet	01:00:06:00:03:00:00:00: 10:76:01:00:c5:00:00:00: 00:00:00:00	See appendix A.
Command-End Packet	02:00:01:80:02:00:00:00: f5:8b:00:00:00:00:00:00	See appendix A.

# Appendix D: Barcode Reader Command Sequence Examples

## D.1 Pre-setup of Barcode Reader

Barcode reader requires presetting it to Trigger Mode once (saving settings in non-volatile memory inside the barcode reader). This is actually done ex-factory and normally not needed to do, unless the barcode reader has been set to other modes by using early version of Apps.

This step is normally NOT required because CS108 has its barcode reader preset to Trigger mode ex-factory.

Step 1: Power on barcode reader using 0x9000 command

Step 2: Use 0x9003 barcode downlink command data API to send the following to CS108:

```
nls0006010;  
nls0313000=30000;  
nls0302000;  
nls0006000;
```

Since 0x9003 can handle 50 bytes of command payload, so the above 4 commands, 50 characters and hence 50 bytes of ascii, can be concatenated and sent down in one shot!!!!

This command will set the barcode reader properly and save that to non-volatile memory. Next time on power up the barcode reader will behave properly - no need to send this again.

Step 3: Power off barcode reader using 0x9001 command

## D.2 Normal Operation of Barcode Reader

This is the normal operation step

- Step 1: On initial successful connection to CS108, power on barcode reader using **0x9000 command** and **leave it on**.
- Step 2: In barcode reading mode, when the “Start” button or the Trigger button is pressed, use **0x9003 command** to send **0x1b, 0x33** down to barcode (only need to send 1 time BT downlink, 2 bytes of payload)
- Step 3: In barcode reading mode, when the “Stop” button or the Trigger button is released, use **0x9003 command** to send **0x1b, 0x30** down to barcode
- Step 4: On final closing of App, then power off the barcode reader using **0x9001 command**.

## Appendix E – Reader Modes/Link Profiles

There are 4 link profiles in CS108: 0, 1, 2, 3. Only 1 profile is active at any time in CS108. The purpose of each link profile is explained below. These purposes correspond to different business and physical scenarios. The user should try out each profile to see which one gives best performance.

Link Profile	0	1	2	3
Purpose	Best Multipath Fading Resistance	Longest Read Range, Dense Reader Mode	Read Range and Throughput, Dense Reader Mode	Maximum Throughput
R-T Modulation	DSB-ASK	PR-ASK	PR-ASK	DSB-ASK
Tari (μs)	25.00	25.00	25.00	6.25
X	1.00	0.50	0.50	0.50
PW (Pulse Width in usec)	12.50	12.50	12.50	3.13
RTcal (usec)	75.00	62.50	62.50	15.63
TRcal (usec)	200.00	85.33	71.11	20.00
DR (Divide Ratio)	8	64/3	64/3	8
T-R Modulation	FM0	Miller-4	Miller-4	FM0
TRExt	1	1	1	1
Link Frequency(LF) (KHz)	40	250	300	400
Data Rate (Kbps)	40	62.5	75	400



## Appendix F – Session

Session is a concept of EPC to allow a tag to respond to multiple readers inventorying it at the same time, each using a different session number.

There are 4 possible sessions: S0, S1, S2, S3.

The user however has to be careful because these 4 sessions have different behavior, notably how the tag flag “persist” in time. A tag, before inventory or when just after power on, has a flag of State A. When it is inventoried, the flag will go to State B. The tag flag will stay in State B until the tag powers off or the persistence time is up.

A reader can declare it only wants to inventory flag A, so that after a tag is inventoried and its flag gone to State B, it will no longer respond to further inventory rounds – until the end of the persistence time.

Now for S0, S1, S2 and S3, the persistence times are DIFFERENT! Because of that, one has to be very careful in choosing which session to use.

Session	Tag Flags Persistence Time
S0	Tag Energized: indefinite Tag Not Energized: none
S1	Tag Energized: 0.5 second < Persistence Time < 5 seconds Tag Not Energized: 0.5 second < Persistence Time < 5 seconds
S2	Tag Energized: indefinite Tag Not Energized: 2 seconds < Persistence Time
S3	Tag Energized: indefinite Tag Not Energized: 2 seconds < Persistence Time

# Appendix G – Tag Population and Q

Tag Population is the RFID tag population that is to be inventoried. To be more precise, it is the population of tags that can be “seen” by the RFID reader.

Q is an EPC concept related to the way a group of tags is inventoried. When a reader broadcast its desire to inventory tags, it sends out a Q value. The tag will, based on that Q, calculate a certain number and define that as the number of repeated inventories the reader will do. Basically, the relationship of Inventory Repeats and Q is:

$$\text{Inventory Repeats} = 2^Q$$

The tag will then choose by random a certain number less than this Inventory Repeats. When the reader starts doing inventory, the tag will then respond at that repeat number.

In other words, the Inventory Repeats should correspond to Tag Population:

$$\text{Tag Population} = \text{Inventory Repeats} = 2^Q$$

For example, if there are 8 tags, then in theory the Q can be 3, and if each tag chooses a number different from that of the other 7 (miraculously, of course), then the 8 tags will be inventoried in an orderly manner in turn.

**Of course this will never happen**, as the tags will easily choose a number the same as that of another one, and a collision will happen.

Therefore, it is a normal practice to have a bigger Q, such as 4 in this case, so that the 8 tags would have a lower chance of choosing the same number.

Therefore, reversing the equation, ideally, we can have:

$$Q = \text{INTEGER}(\text{LOG}_2(\text{Tag Population}))$$

But in reality, we need some headroom, so that:

$$Q = \text{INTEGER}(\text{LOG}_2(\text{Tag Population} \times 2) + 1)$$

# Appendix H – Query Algorithm

There are 2 types of Query Algorithm: Fixed Q and Dynamic Q.

For Fixed Q, the Q value does not change. In other words, the expected Tag Population does not change.

For Dynamic Q, the Q value changes adaptively: when there are a lot of inventory repeats where no tags respond, the reader will interpret that there are not that many RFID tags in the front, and hence it is more efficient to change the Q to a smaller value. When there are a lot of inventory repeats where the reader receive data but they do not satisfy checksum, meaning there is heavy collision, then the reader will interpret that there are too many RFID tags in the front of the reader, and hence it is better to increase the value of Q. Dynamic Q algorithm is a way to allow the RFID reader to adapt to different amount of RFID tags being seen by the reader. The idea is that if there are not so many tags, then the Q can be reduced and the reader can collect all the tag data faster.

# Appendix I – Target

Target here actually refers to the target flag that the reader wants to inventory. There are 2 possible flags of an RFID tag: State A and State B.

When an RFID tag is first powered up, it has a flag of State A. After it is inventoried, the state of the flag becomes State B.

The tag will only go back to State A if either it is powered off and powered on again, or if its persistence time has run up.

For each round of inventory, the reader sends out notification to the world which tag flag state it wants to inventory. It can keep on inventory State A, or it can inventory State A and State B alternatively from one round of inventory to the next round of inventory.

In theory, it is a good thing to inventory only State A. The reason being that those tags that have been inventoried should not respond again, and will hence quickly reduce the amount of collision between tags. So in general if you set inventory to State A only, the inventory of large amount of tags can be very fast.

The only catch is that when a tag responds to the reader, it does not know another tag is colliding with it. It sends out the response and thinks it has done the job, hence transitioning to flag State B. So in such case, the tag will not respond to further inventory, even though its response has been lost due to collision. Because of that, sometimes the user will set the inventory to target State A in one inventory round, and then State B in the next round, and vice versa, and so on. This is called A/B Toggle or A & B Dual Target or simply Dual Target.

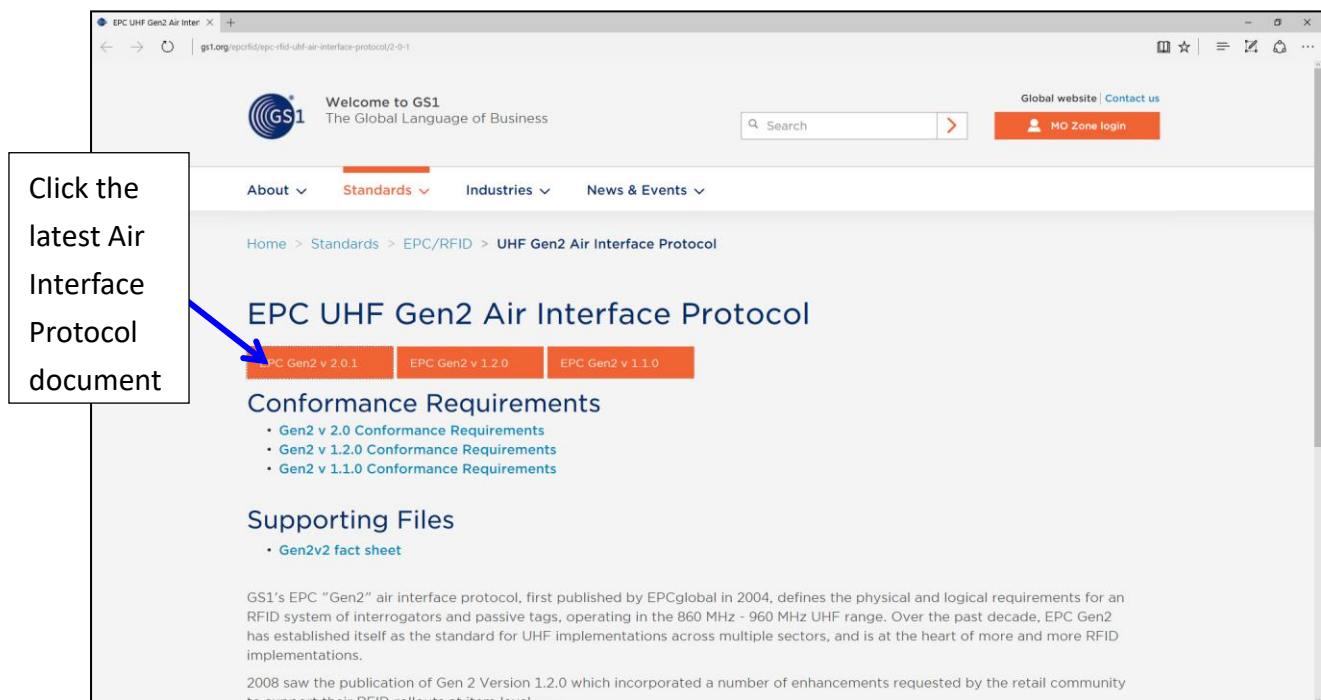
# Appendix J – Security

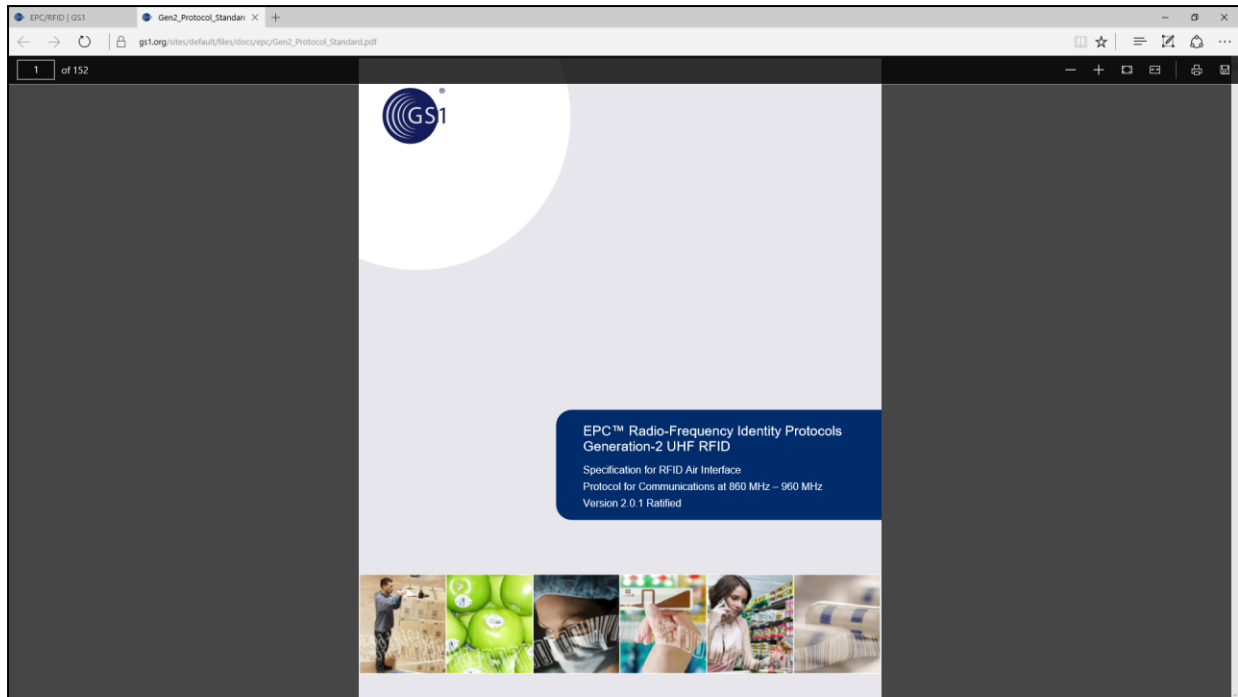
There are 4 actions you can apply on the memory inside an RFID tag:

- 1) Lock
- 2) Unlock
- 3) Permanent Lock
- 4) Permanent Unlock

You can obtain an EPC Global document which can be downloaded from the EPC Global website that explains this:

Once there, press the button showing the latest air interface protocol document and click on it to get the pdf file.





For the Access Password and Kill Password the security locking affects both reading and writing.

For the EPC memory bank and the User memory bank, the security locking affects only writing.

For the TID memory bank, since we are the user and not the manufacturing vendor, there is no security action that can be applied. It has been permanently unlocked in the factory and it cannot be changed.

# Appendix K – Tag Focus

Tag Focus is a special feature of Impinj tag IC. When enabled, and when the reader is using Session S1 and Target A to query the tag, the tag will, once inventoried, remain in Flag B until the inventory is completed.

This is in contrast to the normal EPC query using S1 and Target A, where the tag will only remain in Flag B for 2 to 5 seconds – this time being defined as persistence time.

The original purpose of EPC S1 and Target A is so that those tags that have been inventoried before would not come back and be inventoried again so quickly, so that more time slots are available for other tags that have not been inventoried yet. However, the time of 2 to 5 seconds are simply too short if there are many tags in the environment being illuminated by the reader. This is particularly true if the reader is “seeing” a whole bunch of tagged items in a warehouse, where there may be more than 1000 tags the reader can “see” at any moment. The end result is before all the tags have been inventoried, the early inventoried tags, having passed 5 seconds after it was first inventoried, join back to be inventoried!!

In the early days of EPC standard, a few hundred tags being inventoried is already a rather unthinkable matter, and so 2 to 5 seconds of persistence time is enough. Nowadays, with the ever improving sensitivity of tags, 5 seconds is not enough.

With Impinj Tag Focus enabled, the tag simply would not respond again until the inventory is completely over.

# Appendix L – Receive Path Parameters

The receive path of CSL Reader consists of RF and IF Low Noise Amplifier and Automatic Gain Control Amplifier. The gains of these can be controlled.

Here are the 4 parameters:

- 1) RF LNA High Compression Mode: 0 or 1; if RF LNA Gain is 13, this must be 0. Default is 1
- 2) RF LNA Gain: 1, 7, or 13 dB. If this is set to 13 dB, then RF High Compression Mode must be 0. Default is 1
- 3) IF LNA Gain: 24, 18, 12, 6 dB. Default is 24
- 4) AGC Gain Range: -12, -6, 0, 6 dB. Default is -6



# Appendix M – Models & Regulatory Regions

There are various models, denoted by the alphanumeric key to the right of the dash after the “CS108-“, here denoted by “**N**”. The applicable regulatory regions for each model are described below:

<b>N=1:</b>	865-868 MHz for Europe ETSI, Russia, Middle East countries, 865-867 MHz for India
<b>N=2:</b>	902-928 MHz, FCC, for USA, Canada and Mexico. Hopping frequencies locked
<b>N=2 AS:</b>	920-926 MHz, Australia. Hopping frequencies locked
<b>N=2 NZ:</b>	921.5-928 MHz, New Zealand. Hopping frequencies locked
<b>N=2 OFCA:</b>	920-925 MHz, Hong Kong. Hopping frequencies locked
<b>N=2 RW:</b>	920-928 MHz, Rest of the World, e.g. Philippines, Brazil, Peru, Uruguay, etc.
<b>N=4:</b>	922-928 MHz, Taiwan
<b>N=6:</b>	917-920.8 South Korea
<b>N=7:</b>	920-925 MHz, China
<b>N=8:</b>	916.7-920.9 MHz, Japan
<b>N=9:</b>	915-921 MHz, Europe Upper Band

# Appendix N – CRC Table/Compute Codes

The following is the CRC lookup table and compute code.

```
const unsigned int xdata crc_lookup_table[256] =
{
    0x0000,0x1189,0x2312,0x329b,0x4624,0x57ad,0x6536,0x74bf,
    0x8c48,0x9dc1,0xaf5a,0xbed3,0xca6c,0xdbe5,0xe97e,0xf8f7,
    0x1081,0x0108,0x3393,0x221a,0x56a5,0x472c,0x75b7,0x643e,
    0x9cc9,0x8d40,0xbfdb,0xae52,0xdaed,0xcb64,0xf9ff,0xe876,
    0x2102,0x308b,0x0210,0x1399,0x6726,0x76af,0x4434,0x55bd,
    0xad4a,0xbcc3,0x8e58,0x9fd1,0xeb6e,0xfae7,0xc87c,0xd9f5,
    0x3183,0x200a,0x1291,0x0318,0x77a7,0x662e,0x54b5,0x453c,
    0xbdc b,0xac42,0x9ed9,0x8f50,0xfbef,0xea66,0xd8fd,0xc974,
    0x4204,0x538d,0x6116,0x709f,0x0420,0x15a9,0x2732,0x36bb,
    0xce4c,0xdfc5,0xed5e,0xfcd7,0x8868,0x99e1,0xab7a,0xbaf3,
    0x5285,0x430c,0x7197,0x601e,0x14a1,0x0528,0x37b3,0x263a,
    0xdccd,0xcf44,0xfddf,0xec56,0x98e9,0x8960,0xbbfb,0xaa72,
    0x6306,0x728f,0x4014,0x519d,0x2522,0x34ab,0x0630,0x17b9,
    0xef4e,0xfec7,0xcc5c,0xdd5,0xa96a,0xb8e3,0x8a78,0x9bf1,
    0x7387,0x620e,0x5095,0x411c,0x35a3,0x242a,0x16b1,0x0738,
    0xffcf,0xee46,0xdcdd,0xcd54,0xb9eb,0xa862,0x9af9,0x8b70,
    0x8408,0x9581,0xa71a,0xb693,0xc22c,0xd3a5,0xe13e,0xf0b7,
    0x0840,0x19c9,0x2b52,0x3adb,0x4e64,0x5fed,0x6d76,0x7cff,
    0x9489,0x8500,0xb79b,0xa612,0xd2ad,0xc324,0xf1bf,0xe036,
    0x18c1,0x0948,0x3bd3,0x2a5a,0x5ee5,0x4f6c,0x7df7,0x6c7e,
    0xa50a,0xb483,0x8618,0x9791,0xe32e,0xf2a7,0xc03c,0xd1b5,
    0x2942,0x38cb,0x0a50,0x1bd9,0x6f66,0x7eef,0x4c74,0x5dfd,
    0xb58b,0xa402,0x9699,0x8710,0xf3af,0xe226,0xd0bd,0xc134,
    0x39c3,0x284a,0x1ad1,0x0b58,0x7fe7,0x6e6e,0x5cf5,0x4d7c,
    0xc60c,0xd785,0xe51e,0xf497,0x8028,0x91a1,0xa33a,0xb2b3,
    0x4a44,0x5bcd,0x6956,0x78df,0x0c60,0x1de9,0x2f72,0x3efb,
    0xd68d,0xc704,0xf59f,0xe416,0x90a9,0x8120,0xb3bb,0xa232,
    0x5ac5,0x4b4c,0x79d7,0x685e,0x1ce1,0x0d68,0x3ff3,0x2e7a,
    0xe70e,0xf687,0xc41c,0xd595,0xa12a,0xb0a3,0x8238,0x93b1,
    0x6b46,0x7acf,0x4854,0x59dd,0x2d62,0x3ceb,0x0e70,0x1ff9,
```

```

0xf78f,0xe606,0xd49d,0xc514,0xb1ab,0xa022,0x92b9,0x8330,
0x7bc7,0x6a4e,0x58d5,0x495c,0x3de3,0x2c6a,0x1ef1,0x0f78
};

// -----
// ComputeCRC
//
// This function computes the CRC returns the 16-bit CRC
// -----
unsigned int ComputeCRC(unsigned char* input, unsigned int length, unsigned int init)
{
    unsigned int CRC;
    unsigned int i;

    CRC = init;
    for (i = 0; i < length; i++)
    {
        CRC = UpdateCRC (CRC, input[i]);
    }
    return CRC;
}

// -----
// UpdateCRC
//
// This function accepts a CRC argument and a <newbyte> and returns an
// updated CRC value; Uses the CRC Lookup Table
// -----
unsigned int UpdateCRC (unsigned int crc, unsigned char newbyte)
{
    unsigned short retval;
    unsigned short index;
    unsigned short table_value;
    index = (crc ^ newbyte) & 0xff ;
    table_value = crc_lookup_table[index];
    retval = (crc >> 8) ^ table_value;
    return retval;
}

```