MFC243

Programmer's Manual

Document #: PM0118
Revision 1
2020-06-12

NOTICE

The issuer of this document has made every effort to provide accurate information. The issuer will not be held liable for any technical and editorial omission or errors made herein; nor for incidental consequential damages resulting from the furnishing, performance or use of this material. The information provided in this manual is subject to change without notice.

AGENCY APPROVED

- Specification for FCC Class B
- Specification for CE Class B

FEDERAL COMMUNICATIONS COMMISSION INTERFERENCE STATEMENT

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

-Reorient or relocate the receiving antenna.
-Increase the separation between the equipment and receiver.
-Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
-Consult the dealer or an experienced radio/ TV technician for help.

CAUTION:

Any changes or modifications not expressly approved by the grantee of this device could void the user's authority to operate the equipment.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

RF exposure warning

This equipment must be installed and operated in accordance with provided instructions and the antenna(s) used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter. End-users and installers must be provide with antenna installation instructions and transmitter operating conditions for satisfying RF exposure compliance.

This transmitter module is authorized only for use in device where the antenna may be installed such that 20cm may be maintained between the antenna and users. The final end product must be labeled in a visible area with the following: "Contains FCC ID: A8TBM70ABCDEFGH”

Document History

Document Version	Author	Summary of Change(s)	Date
Rev 0.92	UIC engineers		2019/09/23
Rev 1	Moby Chen	First release	2020/06/12

Table of Contents

General Description

This Half-insert and RFID Mifare Card Reader is designed to read high or low coercivity magnetic card, Mifare cards and Bluetooth Low Energy (BLE) Advertising for gaming applications. The illuminated bezel provides full color display, variable intensity and flashing rates. This product communicates with a host computer or other terminal via a USB interface.

Features

The MFC243 provides the following features:

1	Light Weight
2	High Performance
3	USB interface in HID 1.12 specification
4	Compatible with GDS Card Reader Communication Protocol Standard v1.4
5	Programmable illuminated bezel
6	In System Program (ISP) allows reprogramming of device

Hardware Specification

Magnetic Card Specifications

Card Type

ISO standard card (ISO 7810 and 7811)

Thickness

0.76mm ± 0.08mm

Card Format

Track 1 & 3: 210 bpi

Track 2: 75 bpi

Magnetic Head Life

Min. 500K passes

Error Rate

Read < 0.5%

MTBF

165,000 hours

Magnetic Stripe reading

300 ~ 4000 Oe

Card Data Information

Channel	Track 1	Track 2	Track 3
Recording method	F2F (FM)	F2F (FM)	F2F (FM)
Recording density	210 bpi	75 bpi	210 bpi
Max. Character	38	19	53

Card Operation Speed

Test Card	Speed (IPS)
ISO standard card	4 ~ 40
*Jitter	5 ~ 35
**Low Amplitude	5 ~ 35

Notes:
* Jitter card: Reliable reading of magnetic stripes encoded with bit cell length variations within ±15% of normal as defined by ISO 7811.
** Low amplitude: Reliable reading of magnetic stripes encoded at 60% or more of the encoding amplitude as defined by ISO 7811.

Mifare Card Specifications

Compliant with the ISO/IEC 14443 standard (Type A 13.56MHz).
Compatible with and configured to handle multiple types of Mifare cards

Card Types Support
Guest cards: 1K Mifare Ultralight C

Reference document:
RFID SeaPass & Crew Card – Technical Specifications.doc

MTBF
165,000 hours

Mechanical Specifications

Body Material

SABIC PC 945A

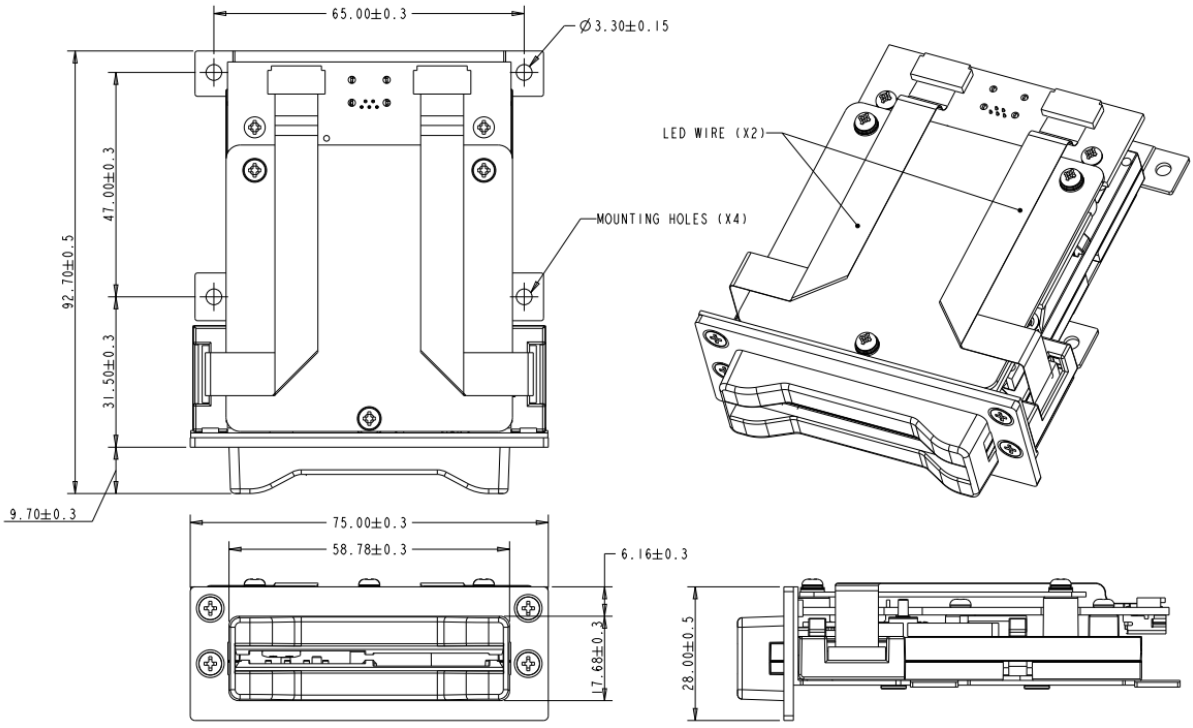
Weight

Approx. 110g

Dimension

Length: 92.7mm
Width: 75 mm
Height: 28 mm

Dimensions of MFC243



Electrical Specifications

Power Required

5VDC, +/-5%

Power Consumption

Standby: 35mA Max.

Operating with three LED turned on: 315mA Max.

Ripple Voltage

50 mVp-p or less

Dielectric

500VDC for 1 minute

Communication

USB Interface: Compatible with USB specification Revision 2.0

Temperature

Operating: -10 ~ 50°C

Storage: -30 ~ 70°C

Humidity

Operating: 10% to 90% non condensing

Storage: 10% to 90% non condensing

ESD

- Install with FCC/CE certificated Lab PC: +/- 15KV air, +/- 8KV contact - The reader can be at least recovered back automatically after ESD discharge is applied.
- Install with customer machine with good proper ground system: +/- 27KV air, +/- 10KV contact and the reader can be at least recovered back automatically after ESD discharge is applied.

EMC

FCC/CE Class B Certificate

Memory/Firmware

Number of writing light sequences to SRAM: unlimited, per microprocessor design.

Number of times erasing and flashing firmware: up to 1000 times Max

Pin Assignment

Mini-USB Type B Pin assignment

PIN NO.	DESCRIPTION
1	Power
2	Data -
3	Data +
4	NC
5	Ground

Software Specification

Related Documents

GDS® Card Reader: Communication Protocol v1.4
Universal Serial Bus (USB) Specification, v2.0
Device Class Definition for USB HID, v1.12
USB Engineering Change Notice – UNICODE UTF-16LE for String Descriptors
RFC 2781 (Unicode standard version 3.0) new REV 5.2
Bluetooth Specification Version 4.0

Memory Organization

System Memory is divided into two parts:

Base addresses	Block	Size
0x0801 0000 – 0x0807 FFFF	Application Program Block	448K Bytes
0x0800 0000 – 0x0800 7FFF	Boot Loader Block	32K Bytes

Boot Loader Block

The first 32K byte of memory is factory programmed with a boot loader. The boot loader is designed to update application program through USB communication per HID V1.12 with a predefined communication protocol. Please refer to the appendix section for more information.

Command and Response Format

Command Format:

<Header><LEN_1><Command_1>
<LEN_2><Command_2><DATA><ADDLRC><XORLRC>

Response Format:

<Header><Length><DATA>

Note:

The <Header> of Command/Response must be 'C2' (Hex).

The <LEN 1> field indicates the length from <Command_1> to <XORLRC>, it is two bytes.

The <LEN 2> field indicates the length from <Command_2> to <ADDLRC>, it is two bytes.

The <Data> field are command or response data. See following section "Command and Response Code" for details.

Command and Response Code

Command_1	Command_2	Description
0x09	0x42 0x4C	Enter Boot Loader Block
0x21	0x7E	Check UNIFORM Produce
0x21	0x57 0x46	Program Flash Address Data Length (max size 1024)
0x21	0x42 0x4E	Get Previous Command
0x21	0x45 0x42 0x4C	Get Boot Loader Version
0x21	0x45 0x4D	Erase Flash
0x39	N/A	Get Application Program Version
0x7F	N/A	Warm Reset

0x09 –

0x42 0x4C - Enter Boot Loader Block

Command Length: 3 bytes

Response Data: = 06h, if success. Length is 1 byte

= 15h, if failure. Length is 1 byte

0x21 –

0x7E - Check UNIFORM Produce

Command Length: 2 bytes

Response Data: = UNIFORM XOR some data, if success. Length is 8 bytes.

= 15h, if failure. Length is 1 byte.

0x57 0x46 - Program Flash Address Data Length (max size 1024)

Command Length: 3 bytes

Response Data: = 06h, if success. Length is 1 byte

= 15h, if failure. Length is 1 byte

0x42 0x4E - Get Previous Command

Command Length: 3 bytes

Response Data: = ”OK”, if success. Length is 2 bytes.

= 15h, if failure. Length is 1 byte.

0x45 0x42 0x4C - Get Boot Loader Version

Command Length: 4 bytes

Response Data: = [Version], if success. Length is 8 bytes.

= 15h, if failure. Length is 1 byte.

0x45 0x4D - Erase Flash

Command Length: 3 bytes

Response Data: = ”OK”, if success. Length is 2 bytes.

= 15h, if failure. Length is 1 byte.

0x39 - Get Application Program Version

Command Length: 1 byte

Response Data: = [Version], if success. Length is 8 bytes.

= 15h, if failure. Length is 1 byte.

0x7F - Warm Reset

Command Length: 1 byte

Response Data: = 06h, if success. Length is 1 byte. Warm Reset will perform.

= 15h, if failure. Length is 1 byte. Warm Reset not allowed

Application Program Block

The 448K byte from 0x0801 0000 to 0x0807 FFFF is Application Program Block. It is the main application code held in the microprocessor. The microprocessor will execute it to perform related operation per USB HID V1.12. Please refer to the appendix section for more information.

HID Command and Response Format

Byte 0	Page report ID
Byte 1	Data or length or states
Byte 2	Data or length

Command

Report ID	Usage ID	Name	Operation When Device Enabled	Operation When Device Disabled
0x02	0x41	Enable	Yes	Yes
0x03	0x42	Disable	Yes	Yes
0x04	0x43	Self Test	No	Yes
0x05	0x44	Request GAT Report	No	Yes
0x08	0x47	Calculate CRC	No	Yes
0x5A	0x0610	Get CRD Configuration	No	Yes
0x5B	0x0611	Read Card Data	Yes	No
0x5C	0x0612	Get ATR **(Not implemented)**	Yes	No
0x5D	0x0613	Transfer to ICC **(Not implemented)**	Yes	No
0x5E	0x0614	Release Latch **(Not implemented)**	Yes	Yes
0x5F	0x0615	Light Control	Yes	Yes
0x60	0x0616	Clear Buffer	Yes	No
0x61	0x0617	Get Count Status **(Not implemented)**	Yes	Yes
0x68	0x061E	Latch Mode **(Not implemented)**	Yes	Yes
0x7D	User set feature	BLE Advertising	Yes	No
0x7E	User set feature	UIC Function	No	Yes
0x7F	User set feature	Dump Memory	Yes	Yes

0x02 0x41 - Enable

bit	7	6	5	4	3	2	1	0
Byte 0	0x02							

0x03 0x42 - Disable

bit	7	6	5	4	3	2	1	0
Byte 0	0x03							

0x04 0x43 - Self Test

bit	7	6	5	4	3	2	1	0
Byte 0	0x04							
	-	-	-	-	-	-	-	NVM

Name	Value	Description
NVM	0	Perform self-test

0x05 0x44 - Request GAT Report

bit	7	6	5	4	3	2	1	0
Byte 0	0x05							

0x08 0x47 - Calculate CRC

bit	7	6	5	4	3	2	1	0
Byte 0	0x08							
Byte 1	Seed 0							
Byte 2	Seed 1							
Byte 3	Seed 2							
Byte 4	Seed 3							

0x5A 0x0610 - Get CRD Configuration

bit	7	6	5	4	3	2	1	0
Byte 0	0x5A							

0x5B 0x0611 - Read Card Data

bit	7	6	5	4	3	2	1	0
Byte 0	0x5B							
Byte 1	-	-	-	-	Mifare Ultralight C	Track3	Track2	Track1

Name	Value	Description
Track1	0	Do not read from Track1
	1	Read data from Track1
Track2	0	Do not read from Track2
	1	Read data from Track2
Track3	0	Do not read from Track3
	1	Read data from Track3
Mifare Ultralight C	0	Do not read from Mifare Ultralight C
	1	Read data from Mifare Ultralight C

0x5F 0x0615 - Light Control

bit	7	6	5	4	3	2	1	0
Byte 0	0x5F							
Byte 1	Pattern 1Red LSB							
Byte 2	Pattern 1Red MSB							
Byte 3	Pattern 1Green LSB							
Byte 4	Pattern 1Green MSB							
Byte 5	Pattern 1Blue LSB							
Byte 6	Pattern 1Blue MSB							
Byte 7	Pattern 2 Red LSB							
Byte 8	Pattern 2 Red MSB							
Byte 9	Pattern 2 Green LSB							
Byte 10	Pattern 2 Green MSB							
Byte 11	Pattern 2 Blue LSB							
Byte 12	Pattern 2 Blue MSB							
Byte 13	Flashing Frequency							

0x60 0x0616 - Clear Buffer

This command is used to clear Read data in buffer.

bit	7	6	5	4	3	2	1	0
Byte 0	0x60							

0x7D - BLE Advertising

This command is used to control the advertising of Bluetooth module.

bit	7	6	5	4	3	2	1	0
Byte 0	0x7D							
Byte 1	-	-	-	-	-	-	-	Start

Byte 2	Interval (LSB)
Byte 3	Interval (MSB)
Byte 4	Length
Byte 5-n	Advertising Data

Note:
1. The field of Interval, Length and Advertising Data is only available when the field of Start is equal 1.
2. The field of Interval is an interval of advertising. The unit is millisecond.
3. The field of Length is the length of advertising data. The maximum length is 31 bytes and does not include itself.

Name	Value	Description
Start	0	Stop the advertising
	1	Start the advertising

0x7E – UIC Function

This command is used to do others function motion, now and only has is switch to bootloader.

bit	7	6	5	4	3	2	1	0
Byte 0	0x7E							
Byte 1~10	0xc2, 0x00, 0x07, 0x09, 0x00, 0x03, 0x42, 0x4c, 0x00, 0x04							

0x7F - Dump Memory

This command is used to read the binary code of Boot Loader or Application Program. The parameter [EEPROM Address] must be in the range 0x0800 0000 to 0x0807 FFFF.

bit	7	6	5	4	3	2	1	0
Byte 0	0x7F							
Byte 1	EEPROM Address (MSB)							
Byte 2	EEPROM Address (LSB)							
Byte 3	Dump Length (MSB)							
Byte 4	Dump Length (LSB)							

Note:
1. The maximum length of Dump Length is 63.

EVENT SUPPORT

Report ID	Usage ID	Event	Data
N/A	USB Defined	Connection	No
N/A	USB Defined	Disconnection	No
0x06	0x45	Power Status **(Not implemented)**	Yes
0x07	0x46	GAT Data	Yes
0x09	0x48	CRC Data	Yes
0x0A	0x49	Device State **(Not implemented)**	Yes
0x62	0x0618	Failure Status **(Not implemented)**	Yes
0x63	0x0619	CRD Configuration Data	Yes
0x64	0x061A	Card Status	Yes
0x65	0x061B	Card Data	Yes
0x66	0x061C	Error Data **(Not implemented)**	Yes
0x67	0x061D	Count Status **(Not implemented)**	Yes

0x07 0x46 - GAT Data

bit	7	6	5	4	3	2	1	0
Byte 0	0x07							
Byte 1	Index							
Byte 2	Size							
Byte 3	Data 1							
...	...							
Byte 63	Data 61							

0x09 0x48 - CRC Data

bit	7	6	5	4	3	2	1	0
Byte 0	0x09							
Byte 1	Result 0							
Byte 2	Result 1							
Byte 3	Result 2							
Byte 4	Result 3							

0x0A 0x49 - Device State

bit	7	6	5	4	3	2	1	0
Byte 0	0x0A							
Byte 1	-	-	-	-	-	-	Disable	Enable

0x62 0x0618 - Failure Status

bit	7	6	5	4	3	2	1	0
Byte 0	0x062							
Byte 1	-	-	-	-	-	-	ICC Power Fail	Firmware Fail
Byte 2	Diagnostics							

0x63 0x0619 - CRD Configuration

bit	7	6	5	4	3	2	1	0
Byte 0	0x63							
Byte 1	-	-	-	Mifare Ultralight C	Track3	Track2	Track1	-

0x64 0x061A - Card Status

bit	7	6	5	4	3	2	1	0
Byte 0	0x64							
Byte 1	-	-	-	-	Partially Inserted	Card Present	Removed	Inserted
Byte 2	-	-	-	Mifare Ultralight C	Track3	Track2	Track1	-

Note:
1. The field of Byte 2 bit 4 indicates that Mifare Ultralight C is present and supported.
2. The fields of Byte 2 bit 3-1 indicate that MSR card is present and supported.

0x65 0x061B - Card Data

bit	7	6	5	4	3	2	1	0
Byte 0	0x65							
Byte 1	Index (LSB)							
Byte 2	Index (MSB)							
Byte 3	Size							
Byte 4	Type							
Byte 5	Data 1							
…	…							
Byte 63	Data 59							

Name	Value	Description
Type	1-3 or 5	Data type:

		1 = Track 1 2 = Track 2 3 = Track 3 5 = Mifare Ultralight C
Data	0 to 255	Data is ASCII

0x66 0x061C - Error Data

bit	7	6	5	4	3	2	1	0
Byte 0	0x66							
Byte 1	Error Code							

0x67 0x061D - Count Status

bit	7	6	5	4	3	2	1	0
Byte 0	0x67							
Byte 1	Index (LSB)							
Byte 2	Index (MSB)							
Byte 3	Size							
Byte 4	Type							
Byte 5	Data 1							
…	…							
Byte 63	Data 59							

0x7F - Dump content

bit	7	6	5	4	3	2	1	0
Byte 0	7F							
Byte 1	Dump content							
...	...							
Byte 63	Dump content (Max Length :63)							

Appendix A. Application Program Block Related Information

Device Descriptor

Field	Value
	GAME CONTROLS PAGE
Length	12
DescriptorType	01
USB	0200
DeviceClass	00
DeviceSubClass	00
DeviceProtocol	00
MaxPacketSize	08
Vendor	6352
Product	243A
Device	0103
Manufacturer	01
Product	02
SerialNumber	00
NumConfigurations	01

USB Configurations Descriptor

Field	Value	Description
Length	9	Configurations Descriptor size
Descriptor Type	2	Configuration
Total Length	34	Bytes returned
Num Interfaces	1	1 interface
Value	1	Configuration value
Configuration	03	Index of string descriptor describing
Attributes	80	BUS powered
Max Power		500 mA
Length	9	Interface Descriptor size
Descriptor Type	3	Interface
Interface Number	0	
Alternate Setting	0	Alternate setting
Num Endpoints	1	Usage (Card Encode Type)
Interface Class	3	HID
Interface Sub Class	0	no boot
Interface Protocol	0	0=none
Interface	4	string descriptor
Length: HID Descriptor size	9	HID Descriptor size
Descriptor Type	21	HID
BCD HID	1.11	HID Class Spec release number
Country Code	0	Taiwan
Num Descriptors	1	HID class descriptors to follow
Descriptor Type	22	HID
Item Length		Report descriptor
Length	7	Endpoint Descriptor size
Descriptor Type	05	Endpoint
Endpoint Address	81	
Attributes	3	Interrupt endpoint
Max Packet Size	0x40	bytes
Interval	0x2E	ms

Report STRING Descriptor

Manufacturer index of string descriptor
Interface index of string descriptor
SerialNumber index of string descriptor

Manufacturer

- May be used to give a Unicode representation of the idVendor. This is assigned by each manufacturer and kept consistent with regards to case and spelling.
  For example:
  GSA Member Company Name

Interface

- A string must be returned in this format:
- <Protocol Level>,<Product Name>,<Firmware Issue>,<Build Version>, <Manufacturing Date>
- Unicode string made up of several comma-delimited sub-strings. Redundant, trailing commas may be omitted. Leading and trailing spaces within sub-strings will be ignored.
  For example:
  1.1.1,ProductName,1.01,A,2004-01-01
  Only the first three items are compulsory. At a minimum we could have...
  1.1.1,ProductName,1A2B3C, 1.01

SerialNumber

- Serial numbers must be returned as a Unicode string (126 character limit), such as 12345678.
  For example:
  Leading zeros are acceptable, 00000123.

HID Report Descriptor:

There were basically follow the "**GDS® CARD READER: COMMUNICATION PROTOCOL V1.4**", which is at appendix B, **B.1 Card Reader Report Descriptors**, the different part is add for extend feature command.

```
    0x05, 0x92,         // USAGE_PAGE (GSA Gaming Device)
    0x09, 0x16,         // USAGE (Card Reader)
    0xa1, 0x01,         // COLLECTION (Application)
// Enable
    0x09, 0x41,         // USAGE (Enable)
    0x85, 0x02,         // REPORT_ID (2)
    0x15, 0x00,         // LOGICAL_MINIMUM (0)
    0x25, 0x01,         // LOGICAL_MAXIMUM (1)
    0x75, 0x08,         // REPORT_SIZE (8)
    0x95, 0x01,         // REPORT_COUNT (1)
    0xb1, 0x03,         // FEATURE (Cnst, Var, Abs)
// Disable
    0x09, 0x42,         // USAGE (Disable)
    0x85, 0x03,         // REPORT_ID (3)
    0x75, 0x08,         // REPORT_SIZE (8)
    0x95, 0x01,         // REPORT_COUNT (1)
    0xb1, 0x03,         // FEATURE (Cnst,Var,Abs)
// Self Test
    0x09, 0x43,         // USAGE (Self Test)
    0x85, 0x04,         // REPORT_ID (4)
    0xa1, 0x02,         // COLLECTION (Logical)
    0x15, 0x00,         // LOGICAL_MINIMUM (0)
    0x25, 0x01,         // LOGICAL_MAXIMUM (1)
    0x75, 0x01,         // REPORT_SIZE (1)
    0x95, 0x01,         // REPORT_COUNT (1)
    0x09, 0x93,         // USAGE (NVM)
    0xb1, 0x02,         // FEATURE (Data,Var,Abs)
    0x95, 0x07,         // REPORT_COUNT (7)
    0xb1, 0x03,         // FEATURE (Cnst,Var,Abs)
    0xc0,               // END_COLLECTION
// Request GAT Report
    0x09, 0x44,         // USAGE (Request GAT Report)
    0x85, 0x05,         // REPORT_ID (5)
    0x75, 0x08,         // REPORT_SIZE (8)
    0x95, 0x01,         // REPORT_COUNT (1)
    0xb1, 0x03,         // FEATURE (Cnst,Var,Abs)
```

```
// Power Status
        0x09, 0x45,             // USAGE (Power Status)
        0x85, 0x06,             // REPORT_ID (6)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x25, 0x01,             // LOGICAL_MAXIMUM (1)
        0x75, 0x01,             // REPORT_SIZE (1)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x09, 0x91,             // USAGE (Ext. Power)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x09, 0x92,             // USAGE (Need Reset)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x95, 0x06,             // REPORT_COUNT (6)
        0x81, 0x03,             // INPUT (Cnst,Var,Abs)
        0xc0,                   // END_COLLECTION
// GAT Data
        0x09, 0x46,             // USAGE (GAT Data)
        0x85, 0x07,             // REPORT_ID (7)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x09, 0x61,             // USAGE (Index)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x25, 0x3d,             // LOGICAL_MAXIMUM (61)
        0x09, 0x62,             // USAGE (Length)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x95, 0x3d,             // REPORT_COUNT (61)
        0x09, 0xb0,             // USAGE (Miscellaeous data)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0xc0,                   // END_COLLECTION
// Calculate CRC
        0x09, 0x47,             // USAGE (Calculate CRC)
        0x85, 0x08,             // REPORT_ID (8)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
```



```
        0x95, 0x04,             // REPORT_COUNT (4)
        0x09, 0x63,             // USAGE (Seed)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0xc0,                   // END_COLLECTION
// CRC Data
        0x09, 0x48,             // USAGE (CRC Data)
        0x85, 0x09,             // REPORT_ID (9)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x04,             // REPORT_COUNT (4)
        0x09, 0x64,             // USAGE (Result)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0xc0,                   // END_COLLECTION
// Device State
        0x09, 0x49,             // USAGE (Device State)
        0x85, 0x0a,             // REPORT_ID (10)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x25, 0x01,             // LOGICAL_MAXIMUM (1)
        0x75, 0x01,             // REPORT_SIZE (1)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x09, 0x94,             // USAGE (Enable)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x09, 0x95,             // USAGE (Disable)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x95, 0x06,             // REPORT_COUNT (6)
        0x81, 0x03,             // INPUT (Cnst,Var,Abs)
        0xc0,                   // END_COLLECTION
// Get CRD Config
        0x0a, 0x10, 0x06,       // USAGE (Get CRD Config)
        0x85, 0x5a,             // REPORT_ID (90)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x01,             // REPORT_COUNT (1)
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
// Read Card Data
        0x0a, 0x11, 0x06,       // USAGE (Read Card Data)
        0x85, 0x5b,             // REPORT_ID (91)
        0xa1, 0x02,             // COLLECTION (Logical)
```

```
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x25, 0x01,             // LOGICAL_MAXIMUM (1)
        0x75, 0x01,             // REPORT_SIZE (1)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x60, 0x06,       // USAGE (Track1)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0x0a, 0x61, 0x06,       // USAGE (Track2)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0x0a, 0x62, 0x06,       // USAGE (Track3)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0x95, 0x05,             // REPORT_COUNT (5)
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
        0xc0,                   // END_COLLECTION
// Get ATR   ;not implemented
        0x0a, 0x12, 0x06,       // USAGE (Get ATR)
        0x85, 0x5c,             // REPORT_ID (92)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x01,             // REPORT_COUNT (1)
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
// Transfer to ICC ;not implemented
        0x0a, 0x13, 0x06,       // USAGE (Transfer to ICC)
        0x85, 0x5d,             // REPORT_ID (93)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        GDSR Card Reader:
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x02,             // REPORT_COUNT (2)
        0x0a, 0x30, 0x06,       // USAGE (Index)
        0xb1, 0x02,             // Feature (Data,Var,Abs)
        0x25, 0x3d,             // LOGICAL_MAXIMUM (61)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x31, 0x06,       // USAGE (Size)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x95, 0x3c,             // REPORT_COUNT (60)
        0x0a, 0x80, 0x06,       // USAGE (Data)
        0xb2, 0x02, 0x01,       // FEATURE (Data,Var,Abs,Buf)
        0xc0,                   // END_COLLECTION
// Release Latch   ;not implemented
```

```
        0x0a, 0x14, 0x06,       // USAGE (Release Latch)
        0x85, 0x5e,             // REPORT_ID (94)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x01,             // REPORT_COUNT (1)
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
// Light Control    ;not implemented
        0x0a, 0x15, 0x06,       // USAGE (Light Control)
        0x85, 0x5f,             // REPORT_ID (95)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x25, 0x01,             // LOGICAL_MAXIMUM (1)
        0x75, 0x01,             // REPORT_SIZE (1)
        0x95, 0x01,             // REPORT_COUNT (1)
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
        0x0a, 0x63, 0x06,       // USAGE (Red)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0x0a, 0x64, 0x06,       // USAGE (Green)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0x0a, 0x65, 0x06,       // USAGE (Yellow)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0x95, 0x04,             // REPORT_COUNT (4)
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x32, 0x06,       // USAGE (LED Timer)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0xc0,                   // END_COLLECTION
// Clear Buffer
        0x0a, 0x16, 0x06,       // USAGE (Clear Buffer)
        0x85, 0x60,             // REPORT_ID (96)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x01,             // REPORT_COUNT (1)
        0xb1, 0x03,             // FEATURE (Cnst, Var, Abs)
// Get Count Status
        0x0a, 0x17, 0x06,       // USAGE (Get Count Status)
        0x85, 0x61,             // REPORT_ID (97)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x01,             // REPORT_COUNT (1)
        0xb1, 0x03,             // FEATURE (Cnst, Var, Abs)
```



```
// Failure Status
        0x0a, 0x18, 0x06,       // USAGE (Failure Status)
        0x85, 0x62,             // REPORT_ID (98)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x25, 0x01,             // LOGICAL_MAXIMUM (1)
        0x75, 0x01,             // REPORT_SIZE (1)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x66, 0x06,       // USAGE (Firmware)
        0x81, 0x02,             // INPUT (Data, Var, Abs)
        0x0a, 0x67, 0x06,       // USAGE (ICC Power Fail)
        0x81, 0x02,             // INPUT (Data, Var, Abs)
        0x95, 0x05,             // REPORT_COUNT (5)
        0x81, 0x03,             // INPUT (Cnst,Var,Abs)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x68, 0x06,       // USAGE (Other)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x33, 0x06,       // USAGE (Diagnostics)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0xc0,                   // END_COLLECTION
// CRD Configuration Data
        0x0a, 0x19, 0x06,       // USAGE (CRD Config Data)
        0x85, 0x63,             // REPORT_ID (99)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x25, 0x01,             // LOGICAL_MAXIMUM (1)
        0x75, 0x01,             // REPORT_SIZE (1)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x69, 0x06,       // USAGE (ICC)
        0x81, 0x02,             // INPUT (Data, Var, Abs)
        0x0a, 0x60, 0x06,       // USAGE (Track1)
        0x81, 0x02,             // INPUT (Data, Var, Abs)
        0x0a, 0x61, 0x06,       // USAGE (Track2)
        0x81, 0x02,             // INPUT (Data, Var, Abs)
        0x0a, 0x62, 0x06,       // USAGE (Track3)
        0x81, 0x02,             // INPUT (Data, Var, Abs)
        0x95, 0x04,             // REPORT_COUNT (4)
```

```
        0x81, 0x03,             // INPUT (Cnst, Var, Abs)
        0xc0,                   // END_COLLECTION
// Card Status
        0x0a, 0x1a, 0x06,       // USAGE (Card Status)
        0x85, 0x64,             // REPORT_ID (100)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x25, 0x01,             // LOGICAL_MAXIMUM (1)
        0x75, 0x01,             // REPORT_SIZE (1)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x6a, 0x06,       // USAGE (Inserted)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x6b, 0x06,       // USAGE (Removed)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x6c, 0x06,       // USAGE (Card Present)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x6d, 0x06,       // USAGE (Partially Inserted)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x95, 0x04,             // REPORT_COUNT (4)
        0x81, 0x03,             // INPUT (Cnst,Var,Abs)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x69, 0x06,       // USAGE (ICC)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x60, 0x06,       // USAGE (Track1)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x61, 0x06,       // USAGE (Track2)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x62, 0x06,       // USAGE (Track3)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x95, 0x04,             // REPORT_COUNT (4)
        0x81, 0x03,             // INPUT (Cnst,Var,Abs)
        0xc0,                   // END_COLLECTION
// Card Data
        0x0a, 0x1b, 0x06,       // USAGE (Card Data)
        0x85, 0x65,             // REPORT_ID (101)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x02,             // REORT COUNT (2)
```

```
        0x0a, 0x30, 0x06,       // USAGE (Index)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x25, 0x3b,             // LOGICAL_MAXIMUM (59)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x31, 0x06,       // USAGE (Size)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x25, 0x04,             // LOGICAL_MAXIMUM (4)
        0x0a, 0x34, 0x06,       // USAGE (Type)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x95, 0x3b,             // REPORT_COUNT (59)
        0x0a, 0x81, 0x06,       // USAGE (Data)
        0x82, 0x02, 0x01,       // INPUT (Data,Var,Abs,Buf)
        0xc0,                   // END_COLLECTION
// Error Data
        0x0a, 0x1c, 0x06,       // USAGE (Error Data)
        0x85, 0x66,             // REPORT_ID (102)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x35, 0x06,       // USAGE (Error Code)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0xc0,                   // END_COLLECTION
// Count Status
        0x0a, 0x1d, 0x06,       // USAGE (Count Status)
        0x85, 0x67,             // REPORT_ID (103)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x03,             // REPORT_COUNT (3)
        0x0a, 0x36, 0x06,       // USAGE (Mag. Pass Count)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x37, 0x06,       // USAGE (Mag. Error Count Track1)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x38, 0x06,       // USAGE (Mag. Error Count Track2)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x39, 0x06,       // USAGE (Mag. Error Count Track3)
```

```
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x3a, 0x06,       // USAGE (IC Try Count)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0x0a, 0x3b, 0x06,       // USAGE (IC Error Count)
        0x81, 0x02,             // INPUT (Data,Var,Abs)
        0xc0,                   // END_COLLECTION
// Latch Mode     ;not implemented
        0x0a, 0x1e, 0x06,       // USAGE (Latch Mode)
        0x85, 0x68,             // REPORT_ID (104)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x25, 0x01,             // LOGICAL_MAXIMUM (1)
        0x75, 0x01,             // REPORT_SIZE (1)
        0x95, 0x01,             // REPORT_COUNT (1)
        0x0a, 0x6e, 0x06,       // USAGE (Lock)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0x0a, 0x6f, 0x06,       // USAGE (Release)
        0xb1, 0x02,             // FEATURE (Data,Var,Abs)
        0x95, 0x06,             // REPORT_COUNT (6)
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
        0xc0,                   // END_COLLECTION
//Report 0x7f
        0x0a, 0x1F, 0x06,       // USAGE (Dump Memory)
        0x85, 0x7F,             // REPORT_ID(0x7f)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x0a, 0x98, 0x06,       // USAGE (Release)
        0x95, 0x3f,             // REPORT_COUNT (63)
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
        0xc0,                   // END_COLLECTION
//Report 0x7E
        0x0a, 0x20, 0x06,       // USAGE (UIC Function)
        0x85, 0x7E,             // REPORT_ID(0x7e)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x3F,             // REPORT_COUNT (63)
```

```
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
        0xc0,                   // END_COLLECTION
//Report 0x7D
        0x0a, 0x21, 0x06,       // USAGE (BLE Advertising)
        0x85, 0x7D,             // REPORT_ID(0x7d)
        0xa1, 0x02,             // COLLECTION (Logical)
        0x15, 0x00,             // LOGICAL_MINIMUM (0)
        0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
        0x75, 0x08,             // REPORT_SIZE (8)
        0x95, 0x3F,              // REPORT_COUNT (63)
        0xb1, 0x03,             // FEATURE (Cnst,Var,Abs)
        0xc0,                   // END_COLLECTION
        0xc0                    // END_COLLECTION
```

Appendix B. Boot Loader Block Related Information

- Go-Into-Bootloader mode command: **<7E><C2><00><05><09><00><02><42><4C>**

Device Descriptor

Field	Value
Length	12
DescriptorType	01
USB	0200
DeviceClass	00
DeviceSubClass	00
DeviceProtocol	00
MaxPacketSize	40
Vendor	6352
Product	2130
Device	0103
Manufacturer	01
Product	02
SerialNumber	03
NumConfigurations	01

USB Configurations Descriptor

Field	Value	Description
Length	9	Configurations Descriptor size
Descriptor Type	2	Configuration
Total Length	34	Bytes returned
Num Interfaces	1	1 interface
Value	1	Configuration value
Configuration	03	Index of string descriptor describing
Attributes	80	BUS powered
Max Power		500 mA
Length	9	Interface Descriptor size
Descriptor Type	3	Interface
Interface Number	0	
Alternate Setting	0	Alternate setting
Num Endpoints	1	Usage (Card Encode Type)
Interface Class	3	HID
Interface Sub Class	0	no boot
Interface Protocol	0	0=none
Interface	0	string descriptor
Length: HID Descriptor size	9	HID Descriptor size
Descriptor Type	21	HID
BCD HID	1.11	HID Class Spec release number
Country Code	0	Taiwan
Num Descriptors	1	HID class descriptors to follow
Descriptor Type	22	HID
Item Length		Report descriptor
Length	7	Endpoint Descriptor size
Descriptor Type	05	Endpoint
Endpoint Address	81	
Attributes	3	Interrupt endpoint
Max Packet Size	0x08	8 bytes
Interval	0x0A	10ms

Report STRING Descriptor

Manufacturer index of string descriptor
Interface index of string descriptor
SerialNumber index of string descriptor

Manufacturer

- May be used to give a Unicode representation of the idVendor. This is assigned by each manufacturer and kept consistent with regards to case and spelling.
  For example:
  GSA Member Company Name

Interface

- A string must be returned in this format:
- <Protocol Level>,<Product Name>,<Firmware Issue>,<Build Version>, <Manufacturing Date>
- Unicode string made up of several comma-delimited sub-strings. Redundant, trailing commas may be omitted. Leading and trailing spaces within sub-strings will be ignored.
  For example:
  1.1.1,ProductName,1.01,A,2004-01-01
  Only the first three items are compulsory. At a minimum we could have...
  1.1.1,ProductName,1A2B3C, 1.01

SerialNumber

- Serial numbers must be returned as a Unicode string (126 character limit), such as 12345678.
  For example:
  Leading zeros are acceptable, 00000123.

HID Report Descriptor:

```
0x06, 0x00, 0Xff,          //  Usage Page (MSR)

0x09, 0x01,                //  Usage (Decoding Reader)
0xA1, 0x01,                //  Collection (application)

0x15, 0x00,                //  Logical Minimum
0x26, 0xff, 0x00,          //  Logical Maximum
/*12*/
0x75, 0x08,                //  Report Size
0x09, 0x20,                //  Usage (Tk1 Decode Status)
0x09, 0x21,                //  Usage (Tk2 Decode Status)
0x09, 0x22,                //  Usage (Tk3 Decode Status)
0x09, 0x28,                //  Usage (Tk1 Data Length)
0x09, 0x29,                //  Usage (Tk2 Data Length)
0x09, 0x2A,                //  Usage (Tk3 Data Length)

0x09, 0x38,                //  Usage (Card Encode Type)
/*28*/
0x95, 0x07,                //  Report count (7)
0x81, 0x02,                //  Input (Data, Var, Abs, Bit Field)

0x09, 0x30,                //  Usage (Total Sending Length)
0x95, 0x02,                //  Input (Data, Var, Abs, Bit Field
/*34*/
0x82, 0x02, 0x01,          //  Usage (Output Data)
0x09, 0x31,                //  Report Count (328*)
0x96, 0x10, 0x02,          //  Input (Data, Var, Abs, Bit Field)
0x82, 0x02, 0x01,
0x09, 0x20,                //  Usage (Command Message)
0x96, 0x50, 0x03,          //  Report count (520 bytes)
0xb2, 0x02, 0x01,          //  Feature (Data, Var, Abs, Buffered Bytes)
0xa4, 0xb4,
0xc0,                      //  End collection
```

