# HANMEGA Device Interface Specifications for Universal Kiosk

Linux SDK Ver 0.7.0

15 September 2022

R&D Center S/W Team

REVISION HISTORY

| Rev. | Date | Page | Description of Change |
|---|---|---|---|
| Pre 0.1 | 03 February 2016 | | Draft Edition |
| 2.5 | 11 November 2016 | | First Release |
| 2.6 | 08 August 2017 | | Add Get Serial Number |
| 2.7 | 15 March 2018 | | Modify SIU (UCONC Board-Flicker and Status) |
| 2.8 | 08 November 2018 | | Added Get / Set Parameter Function in RPU |
| 2.9 | 12 November 2018 | | Added params of emvkrnl_parameter_init function in EMV<br>Added CIS(Card Image Scanner) |
| 5.30 | 22 November 2018 | | delete libusb.h in package file<br>Change version format |
| 5.31 | 27 November 2018 | | Added support for present type CDU<br>Updated CDU Structure |
| 5.32 | 25 January 2019 | | Added ForceEject function for present type CDU<br>Support 128K BMP Image Printing |
| 5.33 | 31 January 2019 | | Added Set MCR Latch control value in MCR<br>Added MS Data buffer Clear in MCR<br>Opened IC chip command function in MCR/CRS<br>Support the over 128K BMP Image Printing and image format jpg, png, tif<br>Added Set/Get the end line value after printing the image |
| 5.34 | 08 February 2019 | | Fixed an infinite loop(CPU 100%) when an error occurred in the PrintImage cmd in RPU<br>Fixed, the response waiting time was increased to wait up to 180 seconds in CDU |
| 5.35 | 13 February 2019 | | Added function to cancel Encrypt PIN in EPP<br>Added  function to end Encrypt PIN in EPP<br>Updated the EPP_SetActiveKey and EPP_SetKeyMode function in EPP<br>Added feed action for check exit module for the check scanner in SIU.<br>Added support ASIC board(Added SIU_SetFlickerColor/SIU_SetLED Function) |
| 5.36 | 20 February 2019 | | Supports Ubuntu 64 bit.<br>Updated CxImage library for RPU image processing to version 7.01. |
| 5.37 | 21 February 2019 | | Fixed a deadlock issue of CancelEncryptPin / EndEncryptPin |
| 5.38 | 25 February 2019 | | Set forcePIN, regardless of version(Remove the 30 second rule) |
| 5.39 | 25 April 2019 | | Changed command send and receive method in the communication thread in EPP.<br>Changed command send and receive method in the communication thread in SIU. |
| 5.40 | 13 June 2019 | | Return the cancel code of EPP when 30-second timeout from EncryptPIN |
| 5.41 | 17 June 2019 | | Added the HM_DEV_BUSY return value to EPP for multi-threaded support.<br>Returns HM_DEV_BUSY when another command is sent during command execution.<br>To exit before EncryptPIN returns, use Reset, CancelEncryptPIN, EndEncryptPIN |
| 5.42 | 09 August 2019 | | Fixed timeout error in SIU(ASIC) (200ms interval between command)<br>Added  function to RPU_GetSRAMType() in RPU<br>SRAM 512K board support, Max image height 4914 pixels at a time (F/W RPU94 or higher) |
| 5.43 | 14 July 2020 | | Update CDU_SetCassetteNum Function Number 0x0n=>0x3n (Support F/W VOG2A08) |
| 5.44 | 05 August 2020 | | Update EMV 4.3 (Update emvkrnl_parameter_init, Add CAPublicKeys.ini, EMVParam.ini)<br>Added EMV kernel in SDK for 64bit<br>Add SDK log, Change device log to always be saved.(Default storage day value is 30 days) |
| 5.45 | 27 August 2020 | | Fixed SDK to include libCxImage, libusbcis library in the static library |
| 5.46 | 15 September 2020 | | Supports EPP PCIv5.0(included OpenSSL's(v1.1.1g) crypto library(libcrypto.a) |
| 5.47 | 25 September 2020 | | Supports RPU USB communication type<br>Added  function to RPU_UsbOpen(), RPU_UsbClose(), RPU_FWDownload() in RPU |
| 5.48 | 28 September 2020 | | Changed the RPU_CutPaper() and RPU_Reset() timeouts to 15 seconds<br>(reflects RPU USB type escrow timeout) |
| 5.49 | 26 November 2020 | | Fixed missing file close in the log part<br>Fixed to log save that Standard output part of ComPortOpen. |
| 6.0 | 30 November 2020 | | Added CSK Device. Supports CSK(check scanner) wisecube<br>Added poll using status in RPU |
| 6.1 | 21 December 2020 | | Added function to EPP_InstallDefaultKey (Excluding AuthorizedFixing from EPP_InstallKey)<br>Added EPP KeyManagement (Executable: /usr/local/bin/LxKeyManagement)<br>Added Terminal information files (path: /usr/local/share/genmegadevice/terminfo) |
| 6.2 | 04 January 2021 | | Added FirmwareDownload(Executable: /usr/local/bin/LxHmFwDn) |
| 6.3 | 02 February 2021 | | Support Multi-Currency in BillAcceptor(BAU,BA2) |
| 6.4 | 22 March 2021 | | Update EMV 4.3 missing part |
| 6.5 | 27 April 2021 | | Fixed the Expanded Note Inhibits Function in BAU/BA2 |
| 6.6 | 29 September 2021 | | Add Standard C++(libstdc++) library link |
| 6.7 | 09 December 2021 | | Supports Newland Barcode Scanner(BCS) |
| 6.8 | 05 January 2022 | | Bug fix - [BAU, BA2, CIS, SIU, COMMON] memory buffer control (sprintf, etc) |
| 6.9 | 16 March 2022 | | Add CDU_DispenseGlobal and CDU_TestDispenseGlobal Function in CDU<br>Fixed BarcodeData(szCode) from char to unsigned char in BCS (BCSScanData structure)<br>Support New F/W Version Format (New Chipset) SDK(CDU) and LxHmFwDn(CDU) |
| 6.10 | 13 April 2022 | | Change barcode data suffix from default CR(0x0D) (Resolve PDF417 barcode) in BCS<br>Added device string to BCS_GetInfor() in BCS<br>Added ErrorCode HM_DEV_NOT_CHANGEDPWD (0x43) in EPP |

| Rev. | Date | Page | Description of Change |
|---|---|---|---|
| 6.11 | 11 May 2022 | | Add BCS_OpenEx() for Version(Hwv1.0, HL[H/W]-[F/W])<br>Add 300ms Sleep after call FACDEF Command in BCS(NewLand)<br>Add Serial Port File Lock (return HM_DEV_PORTLOCKFAIL(-17) File Lock Fail) |
| 7.0 | 15 September 2022 | | Added HOP Device. Supports HOP(Coin Hopper)<br>Added MCD Device. Supports MCD(Card Dispenser)<br>PCI5.0 Change Moving/Fixing ClearAllData timeout time 3 -> 6 seconds in EPP<br>Fixed MSRead command track 3 data length 0 in MCR<br>Changed LxKeyManagement and LxDevTP to GUI APPs.<br>Added LxDeviceDiagnostic and LxFWDownloader as GUI APPs. |

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

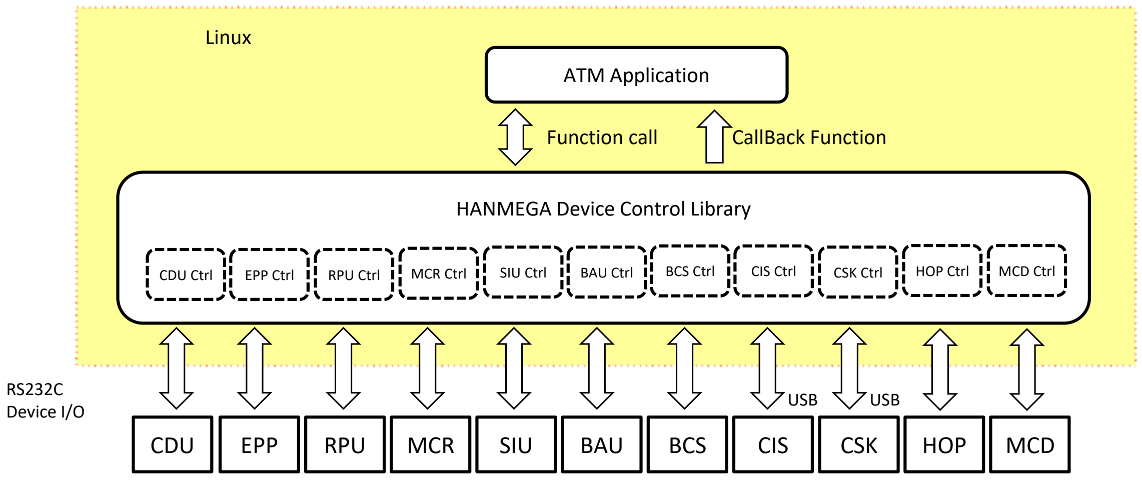
## 1. Introduction

This document describes Device Interface Specifications for Universal Kiosk(Below ATM) application development under Linux Ubuntu environment.
under Linux Ubuntu environment.
The Devices applying this Interface specifications shall be followings :

- CDU : Cash Dispense Unit
- RPU : Receipt Print Unit
- SIU : Sensor and Indicators Unit
- BAU : Bill Acceptor Unit
- CRS(CIS) : Card Reader & Scanner
- EPP : Encrypted PIN Pad
- MCR : Magnetic Card Reader
- BCS : Barcode Scanner
- BA2 : Bill Acceptor Unit
- CSK : Check Scanner

### 1.1 Architecture

Following structure describes the connectivity between or among ATM Application, Device Control DLL and Device.



### 1.2 Environment

Development Environment are as follows :

| | |
|---|---|
| - OS | : Linux (i386, x86_64) |
| - Development Tool | : gcc/g++ 5.4.0 |
| - SDK | : linux SDK (for Universal Kiosk) |

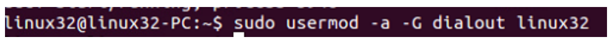
Device Control Library SDK are as follows :

- libgenmegadevice.a(i386), libgenmegadevice64.a(x86_64)
- libgenmegadevice.so(i386), libgenmegadevice64.so(x86_64)
- libgenmegadevice.so.0(i386), libgenmegadevice64.so.0(x86_64)

Device Control DLL is made under standard Linux API and application program can be called by DLL through LIB link.

Since v0.5.45 version, Openssl's(v1.1.1g) crypto library(libcrypto.a) has been included in the SDK with EPP PCIv5.0 support.
Using static library: You need to link libm and libusb and libdl to compile (-lm -lusb-1.0 -ldl)

Enable serial port(COM) device user account.
- Include the dialout group in user account. -> sudo usermod -a -G dialout user



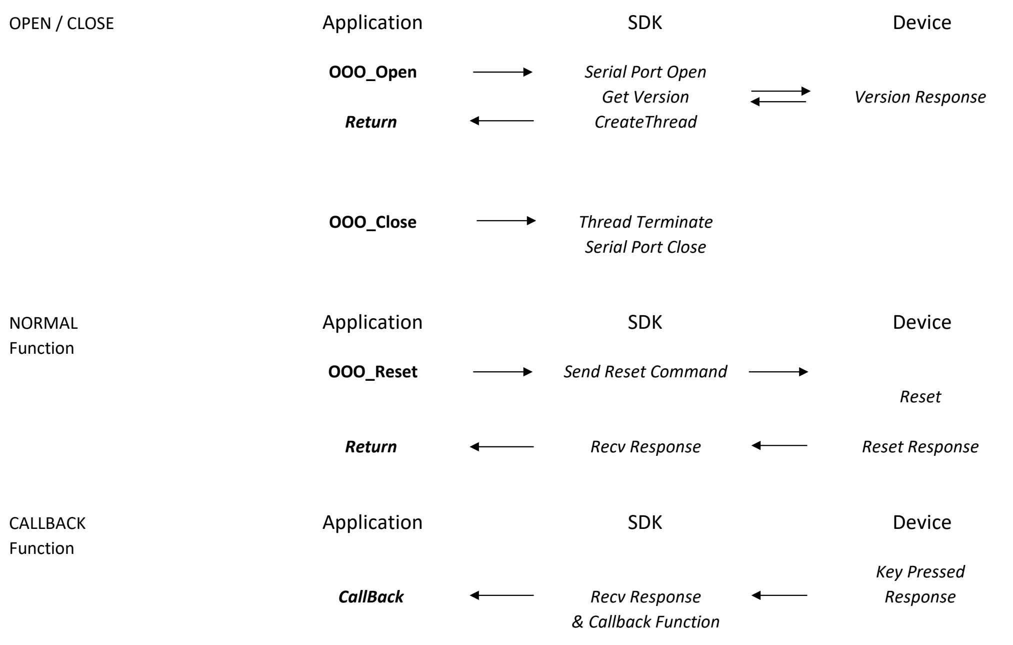
```
linux32@linux32-PC:~$ sudo usermod -a -G dialout linux32
```

- Reboot or log in again

For instructions on how to enable RPU USB device user accounts, see RPU sessions. (page 53)
For instructions on how to enable CIS USB device user accounts, see CIS sessions. (page 149)

## 2. Common SDK Interface

### 2.1 Basic Function Flow



### 2.2 Common Return Value

| Value | Description |
|---|---|
| HM_DEV_OK (0) | Function Success |
| HM_DEV_HW_ERR (-1) | Hardware Error or Function Fail |
| HM_DEV_BUSY (-2) | Function of the previous operation is not completed |
| HM_DEV_INVALID_DATA (-3) | Invalid Parameter or Invalid Data |
| HM_DEV_SECURE_MODE_ERR (-4) | Secure Mode Error (EPP Device Only) |
| HM_DEV_CANCEL (-5) | Key canceled at the input (EPP Device Only) |
| HM_DEV_NOT_READY (-6) | Device is not ready |
| HM_DEV_ALREADY_OPEN (-7) | Device is already open |
| HM_DEV_INTERNAL_ERR (-8) | SDK or device internal error |
| HM_DEV_TIMEOUT (-9) | Timeout |
| HM_DEV_RXOVERFLOW (-10) | Receive data buffer overflow |
| HM_DEV_OPENPORTFAIL (-11) | Communication port open error |
| HM_DEV_INPUT_ERR (-12) | Image Error (RPU Device Only) |
| HM_DEV_REJECTED_BILL (-13) | Return the accepted bill (BAU, BA2 Device Only) |
| HM_DEV_NOPRESENT_BILL (-14) | Not present bill in the escrow (BAU, BA2 Device Only) |
| HM_DEV_PRESENT_BILL (-15) | Present bill in the escrow (BAU, BA2 Device Only) |
| HM_DEV_NOT_CHANGEDPWD (-16) | Return if the password is default when call secure mode command (EPP Device Only) |
| HM_DEV_PORTLOCKFAIL (-17) | Returns if the serial port is already open when opening the device. |
| HM_DEV_NOT_AUTHORIZED (-18) | License key authentication failed (CDU Device Only) |
| HM_DEV_USB_COMM_FAILED (-19) | Failed to send USB data (CIS, CSK Device Only) |
| HM_DEV_USB_INVALID_BLOCK_SIZE (-20) | The image size is different from (width x hight) values. (CIS, CSK Device Only) |
| HM_DEV_NOTSUPPORT (-21) | Returned when calling an not supported method |
| HM_DEV_IMAGE_ERROR (-22) | Image processing error (RPU Device Only) |
| HM_DEV_LONG_DATA (-23) | Data length is too long |
| HM_DEV_TOO_BIG_IMAGE (-24) | Image size cannot be processed (RPU Device Only) |
| HM_DEV_FILENOTOPENED (-25) | The file not exist or not opened in the file path |
| HM_DEV_ERR_UNKNOWN (-26) | Unknown error occurs |
| HM_DEV_SEMISUCCESS (-27) | Need to check the status of the device to determine whether to proceed. |
| HM_DEV_DOING (3) | Waiting or doing some action (BAU, BA2 Device Only) |

3. CDU

(1) It describes following interfaces in order to control Cash Dispense Unit.

| Function | | Description |
|---|---|---|
| 1 | CDU_Open | Open Serial Port |
| 2 | CDU_Close | Close Serial Port |
| 3 | CDU_Reset | Reset CDU in H/W |
| 4 | CDU_Status | Get the Status of CDU |
| 5 | CDU_SetCassetteNum | Define the Number of Cassette of CDU |
| 6 | CDU_Dispense | Dispense the notes from defined Cassette of CDU |
| 7 | CDU_DispenseGlobal | Dispense the notes of different lengths from defined Cassette of CDU |
| 8 | CDU_Present | Present a cash to customer with opening the shutter (Present Type Only) |
| 9 | CDU_ShutterAction | Controls the shutter(Open/Close) (Present Type Only) |
| 10 | CDU_Retract | Retract a cash to Reject bin with closing the shutter (Present Type Only) |
| 11 | CDU_ForceEject | Force a cash-bundle at stacker to move into eject-ready position(Present Type Only) |
| 12 | CDU_TestDispense | Reject the notes after picking-up from defined Cassette of CDU |
| 13 | CDU_TestDispenseGlobal | Reject after picking-up the notes of different lengths from defined Cassette of CDU |
| 14 | CDU_GetLastError | Get the final H/W Error Code of CDU |
| 15 | CDU_Verify_LicenseKey | Verify the CDU license key. |
| 16 | CDU Structure | |
| 17 | Error Code | |
| | | |

## 3.1 CDU_Open

**CDU_Open**

(1) Prototype

**int CDU_Open(IN const char* szPortName, OUT unsigned char szVerInfo[15])**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to CDU ( Ex) "/dev/ttyS2" )

(3) Output Parameter

**unsigned char** szVerInfo[15]
Array Pointer to obtain the F/W version of CDU

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_NOT_AUTHORIZED
HM_DEV_OPENPORTFAIL
HM_DEV_TIMEOUT
HM_DEV_RXOVERFLOW

(5) Message

**void**

(6) Description

Open the Serial Port of CDU
Obtain the Firmware Version of CDU

## 3.2 CDU_Close

**CDU_Close**

1. Introduction

**void CDU_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close the Serial Port of CDU

3.3 CDU_Reset

**CDU_Reset**

(1) Prototype

**int CDU_Reset(IN int iInitializedMode)**

(2) Input Parameter

**int** iInitializedMode : NORMAL_INIT / FORCED_INIT
H/W Initialize Mode

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Reset CDU Device in H/W
Reject the notes to Reject Bin if the notes are on the feeding path
Not checking whether Cassette loaded or not

* Max Processing Time : 30 sec

## 3.4 CDU_Status

**CDU_Status**

(1) Prototype

**void CDU_Status(OUT CDU_STATUS *CduStatus)**

(2) Input Parameter

**void**

(3) Output Parameter

**CDU_STATUS** *CduStatus
Pointer of CDU_STATUS Structure Buffer obtaining CDU Status information

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Obtain Status information of CDU

## 3.5 CDU_SetCassetteNum

**CDU_SetCassetteNum**

(1) Prototype

**int CDU_SetCassetteNum(IN int iCassetteNum)**

(2) Input Parameter

| | |
|---|---|
| **int** iCassetteNum | : 1 ~ 6 |
| Number of Cassette | |

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Designate the number of cassette for using CDU
Renew the status information in accordance with the designated number of cassette
The number of cassette is valid until the function is called again (including Power OFF/ON)

3.6 CDU_Dispense

**CDU_Dispense**

(1) Prototype

**int CDU_Dispense(IN int iDispenseCnt[6], OUT DISPENSED_RESULT lpDispensedResult[6])**

(2) Input Parameter

**int** iDispenseCnt[6]
Interger Array Pointer which is designated number of notes to dispense from each cassette

(3) Output Parameter

**DISPENSED_RESULT** DispensedResult[6]
DISPENSED_RESULT Structure Array Pointer to obtain the result of dispense operation
Array[0] ~ [5] will display the result of Cassette 1 ~ 6

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Dispense the designated number of notes from the each cassette of CDU Device.
HM_DEV_INVALID_DATA will be <Return> if dispense command to un-specified cassette on CDU_SetCassetteNum

* Max Processing Time : (180 + Dispense count) sec

## 3.7 CDU_DispenseGlobal

**CDU_DispenseGlobal**

(1) Prototype

**int CDU_DispenseGlobal(IN int iDispenseCnt[6], IN short sNoteLength[6], OUT DISPENSED_RESULT lpDispensedResult[6])**

(2) Input Parameter

**int** iDispenseCnt[6]
Interger Array Pointer which is designated number of notes to dispense from each cassette
**short** sNoteLength[6]
Short interger Array Pointer which is designated length of notes to dispense from each cassette

(3) Output Parameter

**DISPENSED_RESULT** DispensedResult[6]
DISPENSED_RESULT Structure Array Pointer to obtain the result of dispense operation
Array[0] ~ [5] will display the result of Cassette 1 ~ 6

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

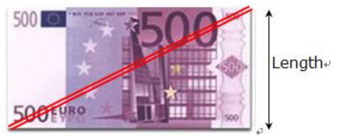
**void**

(6) Description

Dispense the designated number of notes of different lengths from the each cassette of CDU Device.
HM_DEV_INVALID_DATA will be <Return>
if dispense command to un-specified cassette on CDU_SetCassetteNum or having an unsupported length.

* Max Processing Time : (180 + Dispense count) sec

Dispense notes of different lengths for each cassette.
[NoteLength] 60mm ~ 79mm



ex)

```
              //Cassette :  1,  2,  3,  4,  5,  6
    int dispensCount[6] = {  1,  1,  1,  1,  0,  0 };
    short noteLength[6] = {66, 67, 76, 77, 0, 0}
    DISPENSED_RESULT result[6];

    iRet = CDU_DispenseGlobal( dispensCount, noteLength, result );
    if(iRet == HM_DEV_OK ) {
       printf("\n [RESULT]: %d notes dispensed with %d notes rejected", result[0].iDispensedCount,result[0].iRejectedCount);
    }
```

## 3.8 CDU_Present

**CDU_Present**

(1) Prototype

**int CDU_Present()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Present the cash to the customer with opening the shutter the dispensed cash.
If CDU_Present return value is HM_DEV_OK, check Status to check if there is cash remaining in the shutter

**ex) Wait 30 seconds to taken the cash**

```
iRet=CDU_Dispense( dispcnt, result );   // Dispense cash
if(iRet != HM_DEV_OK )  {
    printf("CDU DISPENSE FAIL\n");
    return -1;
} else {
    printf("CDU DISPENSE SUCCESS\n");
    if(CduStatus.iDispenseType == PRESENT_TYPE) {  // If CDU is present type
        iRet=CDU_Present();    // Present cash
        if(iRet == HM_DEV_OK ) {
            printf("CDU PRESENT SUCCESS\n");
            printf("CHECK STATUS FOR 30 SECONDS\n");
            iStartT = GetTickCount();
            while(1) {
                iCurT = GetTickCount();
                if((iStartT + 30*1000) < iCurT) {   // Wait 30 seconds
                    printf("DID NOT TAKE CASH\n");
                    printf("RETRACT CASH\n");
                    iRet = CDU_Retract();    // Retract cash
                    if(iRet == HM_DEV_OK ) {
                        printf("CDU RETRACT SUCCESS\n");
                        break;
                    } else {
                        printf("CDU RETRACT FAIL\n");
                        return -1;
                    }
                }
                iRet = CDU_Status(&CduStatus);
                if(iRet == HM_DEV_OK ) {
                    if(CduStatus.iShutterRemain == 0) {    // If Empty Shutter
                        printf("TOOK CASH / CLOSE SHUTTER\n");
                        iRet = CDU_ShutterAction(0);    // Shutter close
                        break;
                    }
                }
            }
        }
    }
}
```

## 3.9 CDU_ShutterAction

**CDU_ShutterAction**

(1) Prototype

**int CDU_ShutterAction(IN int iOpen)**

(2) Input Parameter

| | |
|---|---|
| **int** iOpen | : 0 : Close, 1 : Open |

This parameter controls the shutter

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Processes a command to open or close the shutter

## 3.10 CDU_Retract

**CDU_Retract**

(1) Prototype

**int CDU_Retract()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Retract a cash to reject bin with closing the shutter

### 3.11 CDU_ForceEject

**CDU_ForceEject**

(1) Prototype

**int CDU_ForceEject()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Force a cash-bundle at stacker to move into eject-ready position

Command Flow
1. CDU_Dispens() => Error occurred (Notes was detected on the stacker)
2. CDU_ForceEject() => Move the detected notes in stacker into eject-ready position
3. CDU_Present() or CDU_Reset(Forced) => present notes to the customer(present command) or retract notes(Forced reset command).

3.12 CDU_TestDispense

## CDU_TestDispense

(1) Prototype

**int CDU_TestDispense( IN int iDispenseCnt[6], OUT DISPENSED_RESULT DispensedResult[6])**

(2) Input Parameter

**int** iDispenseCnt[4]
Integer Array Pointer which designates number of notes to be recovered from each cassette after picking-up

(3) Output Parameter

**DISPENSED_RESULT** *DispensedResult
DISPENSED_RESULT Structure Array Pointer which obtains the result of recovery operation after dispensing
Array[0] ~ [5] will display the rusult of Cassette 1 ~ 6

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**Void**

(6) Description

Reject the designated number of notes from the each cassette of CDU Device after picking up
Typically it can be used to confirm whether the notes are picked up normally or not, after loading the notes to the cassette
HM_DEV_INVALID_DATA will be <Return> if dispense command to un-specified cassette on CDU_SetCassetteNum

* Max Processing Time : (180 + Dispense count) sec

## 3.13 CDU_TestDispenseGlobal

**CDU_TestDispenseGlobal**

(1) Prototype

**int CDU_TestDispenseGlobal( IN int iDispenseCnt[6], IN short sNoteLength[6],**
**OUT DISPENSED_RESULT DispensedResult[6])**

(2) Input Parameter

**int** iDispenseCnt[6]
Integer Array Pointer which designates number of notes to be recovered from each cassette after picking-up
**short** sNoteLength[6]
Short Interger Array Pointer which designates length of notes to be recovered from each cassette after picking-up

(3) Output Parameter

**DISPENSED_RESULT** *DispensedResult
DISPENSED_RESULT Structure Array Pointer which obtains the result of recovery operation after dispensing
Array[0] ~ [5] will display the rusult of Cassette 1 ~ 6

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

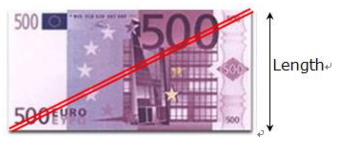
**Void**

(6) Description

Reject the designated number of the notes of different lengths from the each cassette of CDU Device after picking up
Typically it can be used to confirm whether the notes are picked up normally or not, after loading the notes to the cassette
HM_DEV_INVALID_DATA will be <Return>
if dispense the command to un-specified cassette on CDU_SetCassetteNum or having an unsupported length.

* Max Processing Time : (180 + Dispense count) sec

Dispense notes of different lengths for each cassette.
[NoteLength] 60mm ~ 79mm



ex)
```
            //Cassette :  1,  2,  3,  4,  5,  6
    int dispensCount[6] = { 1,  1,  1,  1,  0,  0 };
    short noteLength[6] = {66, 67, 76, 77, 0, 0}
    DISPENSED_RESULT result[6];

    iRet = CDU_TestDispenseGlobal( dispensCount, noteLength, result );
    if(iRet == HM_DEV_OK ) {
       printf("\n [RESULT]: %d notes dispensed with %d notes rejected", result[0].iDispensedCount,result[0].iRejectedCount);
    }
```

## 3.14 CDU_GetLastError

**CDU_GetLastError**

(1) Prototype

**void CDU_GetLastError(OUT unsigned char szErrorCode[5])**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** szErrorCode[5]
Array Pointer which obtains final ErrorCode of CDU

(4) Return Value

**void**

(5) Message

**void**

(6) Description

it obtains final H/W ErrorCode of CDU Device

## 3.15 CDU_Verify_LicenseKey

**CDU_Verify_LicenseKey**

(1) Prototype

**int CDU_Verify_LicenseKey(IN char szKey[16])**

(2) Input Parameter

**char szKey[16]**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_AUTHORIZED

(5) Message

**void**

(6) Description

Verify the CDU license key.  If it is failed, CDU do not open.

## 3.16 CDU Structure

### (1) CDU Status

CDU_STATUS Structure

| | | |
|---|---|---|
| int iLineStatus | : | HM_DEV_CONNECT / HM_DEV_DISCONNECT<br>Displays the connection status with CDU Device |
| int iCstNum | : | 1 ~ 6<br>Displays the designated number of cassette |
| int iDispenseType | : | 0 : SPRAY_TYPE, 1 : PRESENT_TYPE<br>Cash dispense type definition of CDU. |
| int iJamStatus | : | JAM_NO / JAM_CST1 / JAM_CST2 / JAM_CST3 / JAM_CST4 / JAM_CST5 / JAM_CST6 / JAM_TRANSFER<br>Displays the place where the JAM occurred |
| int iCst1Status | : | CST_NORMAL / CST_NEAREND / CST_MISSING<br>Displays the status of Cassette Number 1 |
| int iCst2Status | : | CST_NORMAL / CST_NEAREND / CST_MISSING<br>Displays the status of Cassette Number 2 |
| int iCst3Status | : | CST_NORMAL / CST_NEAREND / CST_MISSING<br>Displays the status of Cassette Number 3 |
| int iCst4Status | : | CST_NORMAL / CST_NEAREND / CST_MISSING<br>Displays the status of Cassette Number 4 |
| int iCst5Status | : | CST_NORMAL / CST_NEAREND / CST_MISSING<br>Displays the status of Cassette Number 5 |
| int iCst6Status | : | CST_NORMAL / CST_NEAREND / CST_MISSING<br>Displays the status of Cassette Number 6 |
| int iShutterStatus | : | 0 : Shutter Close, 1 : Shutter Open<br>Displays the status of Shutter (Present type CDU only) |
| int iShutterRemain | : | 0 : Empty, 1 : Remain<br>Displays whether cash is present in shutter (Present type CDU only) |
| int iStackerRemain | : | 0 : Empty, 1 : Remain<br>Displays whether cash is present in stacker (Present type CDU only) |
| int iTransporterRemain | : | 0 : Empty, 1 : Remain<br>Displays whether cash is present in Transporter (Present type CDU only) |

* The status information of un-specified cassette number will always be displayed <0> and include <Empty> status in case of CST_NEAREND

### (2) CDU Dispense

DISPENSED_RESULT Structure

| | | |
|---|---|---|
| int iDispensedCount | : | Displays the number of note completed to dispense |
| int iRejectedCount | : | Displays total number of notes which has been rejected |
| int iPassedCount | : | Displays the number of notes picked up from the cassette |
| int iSkewCount | : | Displays the number of notes rejected due to Skew feeding |
| int iAbnormalSpaceCount | : | Displays the number of notes rejected due to abnormal dispensing gap |
| int iLongCount | : | Displays the number of notes rejected due to long note |
| int iShortCount | : | Displays the number of notes rejected due to short note |
| int iDoubleNoteCount | : | Displays the number of notes rejected due to abnormal thickness |
| int iHalfSizeCount | : | Displays the number of notes rejected due to half length of note |

* DISPENSED_RESULT Structure Array[0] ~ [5]은 1 ~ 6 Cassette의 정보를 나타낸다.

### (3) CDU TestDispense

DISPENSED_RESULT Structure

| | | |
|---|---|---|
| int iDispensedCount | : | Displays the number of note completed to dispense |
| int iRejectedCount | : | Displays total number of notes which has been rejected |
| int iPassedCount | : | Displays the number of notes picked up from the cassette |
| int iSkewCount | : | Displays the number of notes rejected due to Skew feeding |
| int iAbnormalSpaceCount | : | Displays the number of notes rejected due to abnormal dispensing gap |
| int iLongCount | : | Displays the number of notes rejected due to long note |
| int iShortCount | : | Displays the number of notes rejected due to short note |
| int iDoubleNoteCount | : | Displays the number of notes rejected due to abnormal thickness |
| int iHalfSizeCount | : | Displays the number of notes rejected due to half length of note |

* DISPENSED_RESULT Structure Array[0] ~ [5] will display the information of cassette number 1 ~ 6

## 3.17 CDU Error Code

| Error Codes Table | |
|---|---|
| CODE | Description |
| C0000 | NORMAL |
| C0010 | No notes detect at Stacker before presenting action. |
| C0012 | GTR/GTL Sensor detects a note before dispensing or after initializing. |
| C0014 | Not normal position when Stacker Base closing |
| C0015 | Not normal position when Stacker Base opening |
| C0016 | Notes detect at Stacker after presenting action. |
| C0019 | Notes free away at withdrawal area after presenting action. |
| C0020 | Time out during withdrawal monitoring |
| C0021 | DBL detects a note remains |
| C0022 | SKRA/SKLA sensor detects a note remains when initializing or dispensing |
| C0024 | Not normal position of Presenting Unit |
| C0025 | No notes detect at withdrawal area before retracting action or initializing(Init.Flag0x02) |
| C0026 | Notes detect at withdrawal area after retracting action. |
| C0028 | EJR/L Sensor detects a note after Initializing or before dispensing |
| C0029 | Notes detect at stacker path after retracting action or initializing |
| C0030 | Main motor failure |
| C0034 | Notes detect at stacker after retracting action |
| C0036 | EXIT , EJR/L, PCAM sensor detects notes before initializing |
| C0037 | Double feed detection module operates abnormally |
| C0039 | Gate Solenoid failure |
| C003A | Request more than 5 notes during Test Dispensing |
| C003B | SKRB/SKLB sensor detects a note remains when initializing or dispensing |
| C0043 | Reject more than 20 notes |
| C0044 | More than 10 times consecutive reject occurs |
| C0045 | The requested number < the exit number |
| C0046 | Exit sensor detects a note remains when initializing or dispensing |
| C0047 | Miss Pick Up Error at 1st cassette |
| C0049 | Request to dispense 0 note |
| C004A | Jam is detected at 1st cassette exit path(SKLA/SKLB) during dispensing |
| C004B | Shutter open failure during presenting |
| C004D | First cassette is not detected |
| C004E | 2nd cassette is not detected |
| C004F | More than 85 seconds passed during driving motor |
| C0051 | The requested notes are more than 150 |
| C0052 | Note-jam is detected at the exit area of 1st cassette after dispensing. |
| C0054 | Long length is detected on the EJR sensor during dispensing |
| C0058 | The logical number of notes < the exit number |
| C005B | Miss Pick Up Error at the 2nd cassette |
| C0060 | Note-jam is detected at the exit area of 3rd cassette after dispensing. |
| C006A | Jam is detected at 2nd cassette exit path during dispensing |
| C006B | SKRC/SKLC sensor detects a note remains when initializing or dispensing |
| C006F | Abnormal Note Size |
| C0070 | Note-jam is detected at the exit area of 4th cassette after dispensing. |
| C007A | Jam is detected at the fourth cassette exit path during dispensing |
| C007B | SKRD/SKLD sensor detects a note remains when initializing or dispensing |
| C007C | Miss Pick Up Error at the fourth cassette |
| C007D | The fourth cassette is not detected |
| C0080 | Note-jam is detected at the exit area of the 2nd cassette after dispensing. |
| C0081 | Jam is detected at the DBL sensor area during dispensing |
| C0083 | Jam is detected at the GTL sensor area during dispensing |
| C0086 | Jam is detected at the path from GTR/L to EJR sensor during dispensing |
| C009A | Jam is detected at the 3rd cassette exit path during dispensing |
| C009D | The 3rd cassette is not detected |
| C009F | Miss Pick Up Error at the 3rd cassette |
| C00A0/C00A1 | No ACK/NACK Response after transmission command. |
| C00A2/C00A3 | No response within 15 seconds after shutter operation |
| C00B3 | Can't close the shutter |
| C00B4 | Can't open the shutter |
| C00B5 | Can't open the shutter(shutter is between open sensor and close sensor) |
| C00E0 | Note-jam is detected at the exit area of the 5-cassette after dispensing. |
| C00EA | Jam is detected at the 5-cassette exit path during dispensing |
| C00EB | SKRE/SKLE sensor detects a note remains when initializing or dispensing |
| C00EC | Miss Pick Up Error at the 5-cassette |
| C00ED | The 5-cassette is not detected |
| C00F0 | Note-jam is detected at the exit area of the 6-cassette after dispensing. |
| C00FA | Jam is detected at the 6-cassette exit path during dispensing |
| C00FB | SKRF/SKLF sensor detects a note remains when initializing or dispensing |
| C00FC | Miss Pick Up Error at the 6-cassette |
| C00FD | The 6-cassette is not detected |

## 4. EPP

(1) It describes following interfaces to control Encrypted PIN pad.

| | Function | Description |
|---|---|---|
| 1 | EPP_Open | Open Serial Port and start communication thread |
| 2 | EPP_Close | Close Serial Port and end communication thered |
| 3 | EPP_Reset | Change EPP into Normal Mode |
| 4 | EPP_ClearAllData | Initialize EPP |
| 5 | EPP_SetSecureMode | Enter into Secure Mode |
| 6 | EPP_ChangeSecurePassword | Change the Password of Secure Mode |
| 7 | EPP_SetKeyMode | Specify the type of EPP encryption |
| 8 | EPP_DownloadKey | Download the Working Key to EPP |
| 9 | EPP_DownloadPCI3Key | Download the Working Key to EPP (PCI 3.0 or higher) |
| 10 | EPP_ConfirmKeyValue | Confirm Checksum Value |
| 11 | EPP_EncryptPIN | Encrypt the PIN |
| 12 | EPP_CancelEncryptPIN | Cancel encrypt PIN |
| 13 | EPP_EndEncryptPIN | End encrypt PIN |
| 14 | EPP_EncryptByMAC | Encrypt Data with MAC |
| 15 | EPP_SetActiveKey | Specify the Key to encrypt among EPP keys |
| 16 | EPP_GetActiveKey | Get the information of Key saved currently |
| 17 | EPP_GetKeyStatus | Get the information of all Keys saved currently |
| 18 | EPP_InputKey | Put the Key to EPP |
| 19 | EPP_GetStatus | Get the information of status |
| 20 | EPP_GetKeyMode | Get the information of the type of EPP encryption |
| 21 | EPP_InputControl | Control to input NON-PIN(Plain Text)Key |
| 22 | EPP_DownloadTRKey | Load Transmit Key(TR) |
| 23 | EPP_InstallKey | Install the Removal Protection |
| 24 | EPP_InstallDefaultKey | Excluding AuthorizedFixing from EPP_InstallKey |
| 25 | EPP_AuthorizedMoving | Remove the Removal Protection off |
| 26 | EPP_AuthorizedFixing | Set the Removal Protection |
| 27 | EPP_GetPCIType | Get the PCI version of the EPP device. |
| 28 | EPP_GetLastError | Get the final H/W Error Code of EPP |

| | CallBack Function | Description |
|---|---|---|
| 29 | EPP_CallBackRegister | Send a Message to the registered Function whenever key is pressed on PIN Pad |

**[Differences between EPP v3.0 and EPP v5.0]**

1. Secure password : EPP v3.0: 6 digit -> EPP v5.0: 8 digit
2. EPP v5.0 secure password: Must be at least two different digits, and PartA and PartB must be different.
3. Support Encrypt Mode

| | |
|---|---|
| EPP v3.0 : | SINGLE_DES / DUAL_DES / UNIQUE_SINGLE_DES / TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_SINGLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES |
| EPP v5.0 : | TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES |

## 4.1 EPP_Open

**EPP_Open**

(1) Prototype

**int EPP_Open(IN const char* szPortName, OUT unsigned char szVerInfo[10])**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to EPP ( Ex) "/dev/ttyS1" )

(3) Output Parameter

**unsigned char** szVerInfo[10]
Array Pointer to obtain the F/W version of EPP

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Events

**void**

(6) Description

Open Serial Port with EPP
Get Firmware Version of EPP
Start the Thread for communication with EPP

4.2 EPP_Close

**EPP_Close**

1. Introduction

**void EPP_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Events

**void**

(6) Description

End the communication thread of EPP
Close Serial Port of EPP

## 3.3 EPP_Reset

**EPP_Reset**

(1) Prototype

**int EPP_Reset()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Change EPP into Normal Mode

## 4.4 EPP_ClearAllData

**EPP_ClearAllData**

(1) Prototype

**int EPP_ClearAllData()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Clear all internal information of EPP
After then, EPP will be initialized and the password for entering into Secure Mode also will be initialized automatically
Therefore, Master Key and MAC Key, etc must be specified from the beginning.

* Contact separately about Secure Mode Default Password

## 4.5 EPP_SetSecureMode

**EPP_SetSecureMode**

(1) Prototype

**int EPP_SetSecureMode(IN int iPart)**

(2) Input Parameter

**int** iPart : PART_A / PART_B
Specify the Part of Password for Secure Mode

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message
Fuction will be called when key is pressed on PIN Pad

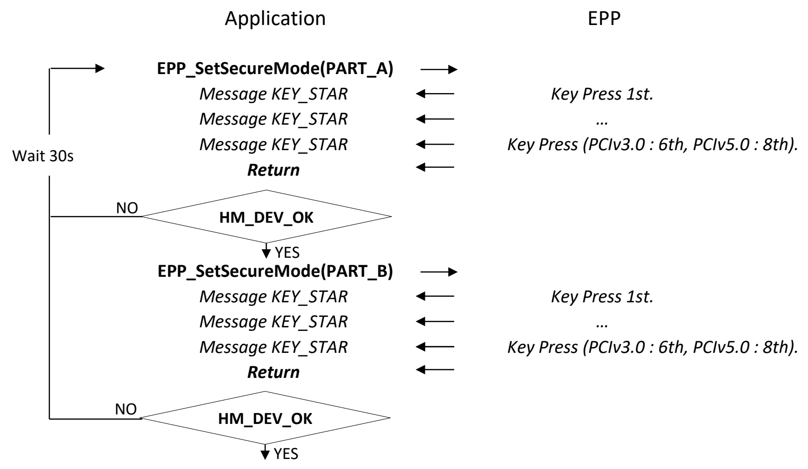
| | | |
|---|---|---|
| int iId | : | HM_DEV_EPP_MSG |
| int iKind | : | EPP_KEY_PRESSED |
| unsigned char cValue | : | KEY_STAR |

(6) Description

Put the each (PCIv3.0 : 6 digit, PCIv5.0 : 8 digit) password with PART_A, PART_B for entering Secure Mode
If password does not match, HM_DEV_HW_ERR will return
Whether HM_DEV_HW_ERR returns or this function is called again in 30 seconds if succefully enter into Secure Mode
HM_DEV_SECURE_MODE_ERR will return.
Secure Mode function can be performed as follows;

| | |
|---|---|
| EPP_ChangeSecurePassword | EPP_InstallKey |
| EPP_SetKeyMode | EPP_InstallDefaultKey |
| EPP_SetActiveKey | EPP_AuthorizedMoving |
| EPP_InputKey | EPP_AuthorizedFixing |
| EPP_ClearAllData (PCI v5.x only. Not PCI v3.0.) | |

Flow



## 4.6 EPP_ChangeSecurePassword

**EPP_ChangeSecurePassword**

(1) Prototype

**int EPP_ChangeSecurePassword(IN int iPart, IN int iInputType)**

(2) Input Parameter

**int** iPart : PART_A / PART_B
Specify the Part of Password for Secure Mode to change
**int** iInputType : KEY_INPUT / KEY_VERIFY
Specify Input Type of password to change

(3) Output Parameter

**void**

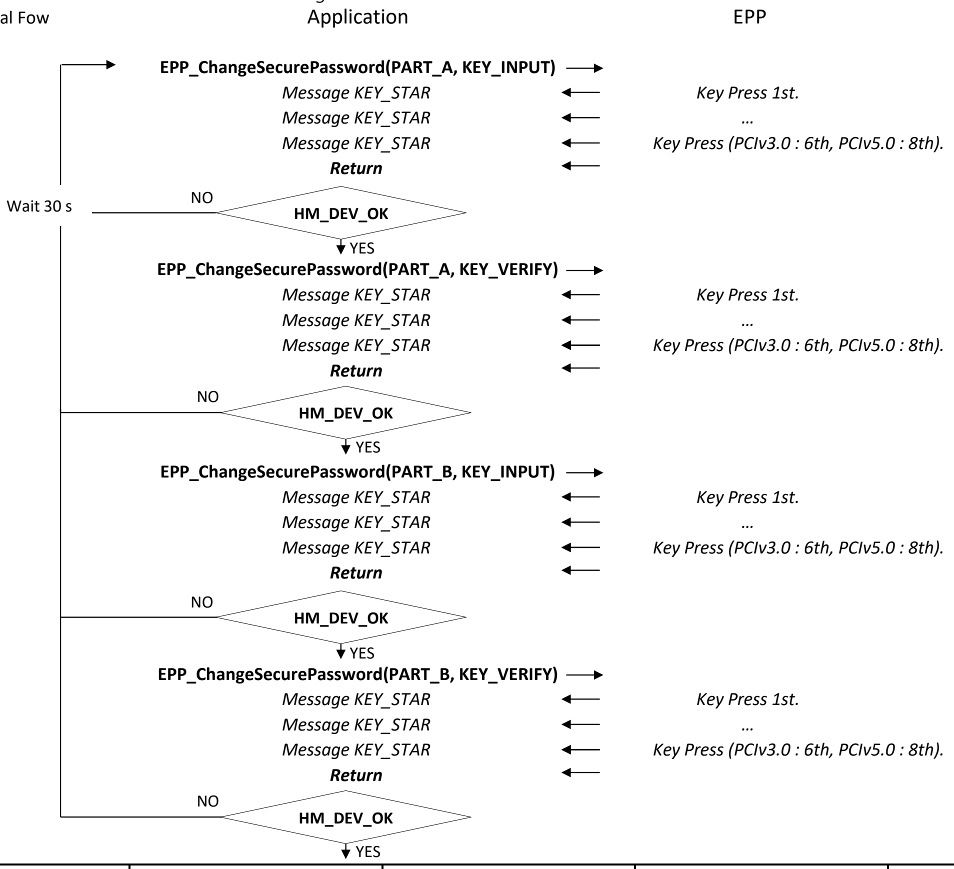
(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message
Fuction will be called when key is pressed on PIN Pad

| | | |
|---|---|---|
| int iId | : | HM_DEV_EPP_MSG |
| int iKind | : | EPP_KEY_PRESSED |
| unsigned char cValue | : | KEY_STAR |

(6) Description
**EPP v5.0 secure password: Must be at least two different digits, and PartA and PartB must be different.**
Change password to enter into Secure Mode
Password must be changed from default setting at first operation
You must enter Secure Mode before executing this command.
Normal Fow



4.7 EPP_SetKeyMode

**EPP_SetKeyMode**

(1) Prototype

**int EPP_SetKeyMode(IN int iKeyMode)**

(2) Input Parameter

**int** iKeyMode : Specify Key Mode
EPP v3.0 : SINGLE_DES / DUAL_DES / UNIQUE_SINGLE_DES / TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_SINGLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES
EPP v5.0 : TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Specify Key Mode to encrypt in EPP

| Key Mode | Remarks |
|---|---|
| SINGLE_DES | Single-DES encryption with Master Key |
| DUAL_DES | Dual-DES encryption with Master Key |
| UNIQUE_SINGLE_DES | Unique Single-DES encryption with Master Key |
| TRIPLE_DES | Triple-DES encryption with Master Key |
| UNIQUE_TRIPLE_DES | Unique Triple-DES encryption with Master Key |
| MAC_SINGLE_DES | Single-DES encryption with Master Key, Applied MAC on Processor message |
| MAC_TRIPLE_DES | Triple-DES encryption with Master Key, Applied MAC on Processor message |
| TRIPLE_MAC_TRIPLE_DES | Triple-DES encryption with Master Key, Applied MAC on Processor message |

4.8 EPP_DownloadKey

## EPP_DownloadKey

(1) Prototype

**int EPP_DownloadKey(IN int iKeyMode, IN unsigned char byKey1[8], IN unsigned char byKey2[8], IN unsigned char byKey3[8], OUT unsigned short *wCheck1, OUT unsigned short *wCheck2, OUT unsigned short *wCheck3)**

(2) Input Parameter

**int** iKeyMode : Specify Key Mode
EPP v3.0 : SINGLE_DES / DUAL_DES / UNIQUE_SINGLE_DES / TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_SINGLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES
EPP v5.0 : TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES

**unsigned char** byKey1[8]
Array Pointer that save 8 Byte Key Value

**unsigned char** byKey2[8]
Array Pointer that save 8 Byte Key Value

**unsigned char** byKey3[8]
Array Pointer that save 8 Byte Key Value

(3) Output Parameter

**unsigned short ***wCheck1
Checksum Value of the byKey1

**unsigned short ***wCheck2
Checksum Value of the byKey2

**unsigned short ***wCheck3
Checksum Value of the byKey3

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

This command is executed only in the Sensitive mode.
Specify Key Mode and Key Value using encryption type applied to EPP, then download them to EPP
Key Value in accordance with Key Mode is as follows;
You must enter Secure Mode before executing this command.

- PIN_KEY

| Key Mode | byKey1 | byKey2 | byKey3 |
|---|---|---|---|
| SINGLE_DES | Use | - | - |
| DUAL_DES | Use | Use | - |
| TRIPLE_DES | Use | Use | Use |
| UNIQUE_TRIPLE_DES | Use | Use | Use |
| MAC_SINGLE_DES | Use | - | - |
| MAC_TRIPLE_DES | Use | Use | Use |

- MAC_KEY

| Key Mode | byKey1 | byKey2 | byKey3 |
|---|---|---|---|
| MAC_SINGLE_DES | Use | - | - |
| MAC_TRIPLE_DES | Use | - | - |

- NPIN_KEY

| Key Mode | byKey1 | byKey2 | byKey3 |
|---|---|---|---|
| -1 | Use | Use | Use |

4.9 EPP_DownloadPCI3Key

## EPP_DownloadPCI3Key

(1) Prototype

**int EPP_DownloadPCI3Key(IN int iKeyKind, IN int iKeyMode, IN unsigned char byKey1[8], IN unsigned char byKey2[8], IN unsigned char byKey3[8], OUT unsigned short *wCheck1, OUT unsigned short *wCheck2, OUT unsigned short *wCheck3)**

(2) Input Parameter

**int** iKeyKind : PIN_KEY / NPIN_KEY / MAC_KEY
Specify Key Kind
**int** iKeyMode : Specify Key Mode
EPP v3.0 : SINGLE_DES / DUAL_DES / UNIQUE_SINGLE_DES / TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_SINGLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES
EPP v5.0 : TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES
**unsigned char** byKey1[8]
Array Pointer that save 8 Byte Key Value
**unsigned char** byKey2[8]
Array Pointer that save 8 Byte Key Value
**unsigned char** byKey3[8]
Array Pointer that save 8 Byte Key Value

(3) Output Parameter

**unsigned short ***wCheck1
Checksum Value of the byKey1
**unsigned short ***wCheck2
Checksum Value of the byKey2
**unsigned short ***wCheck3
Checksum Value of the byKey3

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

This command is executed only in the Sensitive mode.
Specify Key Mode and Key Value using encryption type applied to EPP, then download them to EPP
Key Value in accordance with Key Mode is as follows;
You must enter Secure Mode before executing this command.

- PIN_KEY

| Key Mode | byKey1 | byKey2 | byKey3 |
|---|---|---|---|
| SINGLE_DES | Use | - | - |
| DUAL_DES | Use | Use | - |
| TRIPLE_DES | Use | Use | Use |
| UNIQUE_TRIPLE_DES | Use | Use | Use |
| MAC_SINGLE_DES | Use | - | - |
| MAC_TRIPLE_DES | Use | Use | Use |

- MAC_KEY

| Key Mode | byKey1 | byKey2 | byKey3 |
|---|---|---|---|
| MAC_SINGLE_DES | Use | - | - |
| MAC_TRIPLE_DES | Use | - | - |

- NPIN_KEY

| Key Mode | byKey1 | byKey2 | byKey3 |
|---|---|---|---|
| Not used inside | Use | Use | Use |

4.10 EPP_ConfirmKeyValue

## EPP_ConfirmKeyValue

(1) Prototype

**int EPP_ConfirmKeyValue(IN int iKeyMode, OUT unsigned short *wCheckKey1, OUT unsigned short *wCheckKey2, OUT unsigned short *wCheckKey3, OUT unsigned short *wCheckMac)**

(2) Input Parameter

**int** iKeyMode : Specify Key Mode
TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES

(3) Output Parameter

**unsigned short ***wCheckKey1
Checksum value of the Working Key or Triple DES Working Key1
**unsigned short ***wCheckKey2
Checksum value of the Common Key or Triple DES Working Key2
**unsigned short ***wCheckKey3
Checksum value of the Triple DES Working Key3
**unsigned short ***wCheckMac
Checksum value of the MAC Key, If Key mode is 0Ah, it is 9 bytes of Mac key1, Mac key2, Mac key3 checksum

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

This command confirms whether the values of downloaded keys are normal or abnormal.
Checksum Value in accordance with Key Mode is as follows:

| Key Mode | wCheckKey1 | wCheckKey2 | wCheckKey3 | wCheckMac |
|---|---|---|---|---|
| TRIPLE_DES | Wkey1 | Wkey2 | Wkey3 | - |
| UNIQUE_TRIPLE_DES | Wkey1 | Wkey2 | Wkey3 | - |
| MAC_TRIPLE_DES | Wkey1 | Wkey2 | Wkey3 | MAC Key |
| TRIPLE_MAC_TRIPLE_DES | Wkey1 | Wkey2 | Wkey3 | MAC Key1, MAC Key2, MAC Key3 |
| | | | | |
| | | | | |

## 4.11 EPP_EncryptPIN

**EPP_EncryptPIN**

(1) Prototype

**int EPP_EncryptPIN(IN unsigned char *szAccountNo, IN int iLeastLen, IN unsigned char bEnterKey,
OUT int *iPINLen, OUT unsigned char byEncryptedPIN[8])**

(2) Input Parameter

**unsigned char** *szAccountNo
Buffer's Pointer that saved Account number reading from the Card
**int** iLeastLen : 4 ~ 12
Minimum input length of PIN
**unsigned char** bEnterKey
Specify whether to complete PIN input with Enter Key

(3) Output Parameter

**int** *iPINLen
Length of input PIN
**unsigned char** byEncryptedPIN[8]
Array Pointer to get encrypted PIN Data

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA
HM_DEV_CANCEL

(5) Message

Fuction will be called when key is pressed on PIN Pad

| | | |
|---|---|---|
| int iId | : | HM_DEV_EPP_MSG |
| int iKind | : | EPP_KEY_PRESSED |
| unsigned char cValue | : | KEY_STAR / KEY_ENTER / KEY_CLEAR / KEY_CANCEL |

(6) Description

Input PIN from EPP
If bEnterKey is False after specifying iLeastLen, then it shall be <Return> after input as much as iLeastlen of PIN
In case of pushing ENTER_KEY, it shall be <Return> to HM_DEV_HW_ERR

In case that bEnterKey is <True> after specifying iLeastLen, it shall be as follows;

| | |
|---|---|
| Minimum Input Length of PIN <= Input PIN Length <= 12 | HM_DEV_OK |
| Minimum Input Length of PIN > Input PIN Length | HM_DEV_HW_ERR |

If CANCEL_KEY is pushed at PIN input, then it shall be <Return> to HM_DEV_CANCEL and PIN input shall be cancelled .

## 4.12 EPP_CancelEncryptPIN

**EPP_CancelEncryptPIN**

(1) Prototype

**int EPP_CancelEncryptPIN()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_CANCEL

(6) Description

Cancel the input of the EPP_EncryptPin function. The EPP_EncryptPin function returns HM_DEV_CANCEL(-5)

*** The use of the EPP_CancelEncryptPin function requires EPP_EncryptPin function to be called in a different thread because it is in blocking mode.

ex :)

```
int gbThreadEnd;
void *EppCancelEncryptPinThread() {
    int iRet;  char ch = EOF;
    struct termios preSettings, newSettings;

    printf(" --- Press any key to cancel EncryptPIN. ---\n");

    tcgetattr(0, &preSettings);
    newSettings = preSettings;
    newSettings.c_lflag &= ~ICANON;  newSettings.c_lflag &= ~ECHO;  newSettings.c_lflag &= ~ISIG;
    newSettings.c_cc[VMIN] = 0;  newSettings.c_cc[VTIME] = 0;
    tcsetattr(0, TCSANOW, &newSettings);

    while(gbThreadEnd) {
        ch = getchar();
        if(ch != EOF)  break;
        usleep(1000);
    }
    tcsetattr(0, TCSANOW, &preSettings);
    if(ch != EOF) {
        iRet = EPP_CancelEncryptPIN();
        if(iRet == HM_DEV_OK )    printf("\n --- EPP CANCEL ENC PIN SUCCESS ---\n");
        else                      printf("\n --- EPP CANCEL ENC PIN FAIL ---\n");
    }
    pthread_exit(0);
}

int main (int argc, char *argv[]) {
    int iThreadRet, iRet;
    pthread_t pThreadID;
    char buf[15] = {0}, cEncryptedPIN[8] = {0};
    strcpy(buf, "123456789012);
    gbThreadEnd = 1;
    iThreadRet = pthread_create(&pThreadID, NULL, EppCancelEncryptPinThread, NULL);
    if(iThreadRet != 0) {
        printf("\n --- CancelEncryptPin Thread FAIL ---\n");
        gbThreadEnd = 0;   return -1;
    }
    iRet = EPP_EncryptPIN(&buf[0], 4, 1, &iLen, &cEncryptedPIN[0]);
    if(iRet == HM_DEV_CANCEL)  printf("\n --- EPP EncryptPin Result : CANCEL ---\n");
    else if(iRet == HM_DEV_OK )  printf("\n --- EPP EncryptPin Result : SUCCESS ---\n");
    else                         printf("\n --- EPP EncryptPin Result : FAIL ---\n");
    gbThreadEnd = 0;
    pthread_join(pThreadID, NULL);
}
```

## 4.13 EPP_EndEncryptPIN

**EPP_EndEncryptPIN**

(1) Prototype

**int EPP_EndEncryptPIN()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(6) Description

End the input of the EPP_EncryptPin function. The EPP_EncryptPin function returns the encrypted value.
*** The use of the EPP_EndEncryptPin function requires EPP_EncryptPin function to be called in a different thread because it is in blocking mode.
ex :)

```
int gbThreadEnd;
void *EppEndEncryptPinThread() {
    int iRet;  char ch = EOF;
    struct termios preSettings, newSettings;

    printf(" --- Press any key to end EncryptPIN. ---\n");

    tcgetattr(0, &preSettings);
    newSettings = preSettings;
    newSettings.c_lflag &= ~ICANON;  newSettings.c_lflag &= ~ECHO;  newSettings.c_lflag &= ~ISIG;
    newSettings.c_cc[VMIN] = 0;  newSettings.c_cc[VTIME] = 0;
    tcsetattr(0, TCSANOW, &newSettings);

    while(gbThreadEnd) {
        ch = getchar();
        if(ch != EOF)  break;
        usleep(1000);
    }
    tcsetattr(0, TCSANOW, &preSettings);
    if(ch != EOF) {
        iRet = EPP_EndEncryptPIN();
        if(iRet == HM_DEV_OK )    printf("\n --- EPP END ENC PIN SUCCESS ---\n");
        else                      printf("\n --- EPP END ENC PIN FAIL ---\n");
    }
    pthread_exit(0);
}

int main (int argc, char *argv[]) {
    int iThreadRet, iRet;
    pthread_t pThreadID;
    char buf[15] = {0}, cEncryptedPIN[8] = {0};
    strcpy(buf, "123456789012);
    gbThreadEnd = 1;
    iThreadRet = pthread_create(&pThreadID, NULL, EppEndEncryptPinThread, NULL);
    if(iThreadRet != 0) {
        printf("\n --- EndEncryptPin Thread FAIL ---\n");
        gbThreadEnd = 0;   return -1;
    }
    iRet = EPP_EncryptPIN(&buf[0], 4, 1, &iLen, &cEncryptedPIN[0]);
    if(iRet == HM_DEV_OK )        printf("\n --- EPP EncryptPin Result : SUCCESS ---\n");
    else                          printf("\n --- EPP EncryptPin Result : FAIL ---\n");
    gbThreadEnd = 0;
    pthread_join(pThreadID, NULL);
}
```

## 4.14 EPP_EncryptByMAC

**EPP_EncryptByMAC**

(1) Prototype

**int EPP_EncryptByMAC(IN unsigned char *byData, IN int iLen, OUT unsigned char byEncryptedData[8])**

(2) Input Parameter

**unsigned char ***byData
Byte type Pointer which saved the data to encrypt
**int** iLen
Data Length to encrypt

(3) Output Parameter

**unsigned char** byEncryptedData[8]
Array Pointer which saved the encrypted data by MAC Algorithm

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Encrypt input data by MAC algorithm

## 4.15 EPP_SetActiveKey

**EPP_SetActiveKey**

(1) Prototype

**int EPP_SetActiveKey(IN int iKeyMode, IN int iKeyIndex, OUT int *iCurIndex, OUT unsigned short *wCheck)**

(2) Input Parameter

**int** iKeyMode : Specify Key Mode

| | |
|---|---|
| EPP v3.0 : | SINGLE_DES / DUAL_DES / UNIQUE_SINGLE_DES / TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_SINGLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES |
| EPP v5.0 : | TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES |

**int** iKeyIndex : 0 ~ 11
Specify Key Index

(3) Output Parameter

**int** ***iCurIndex
Pointer to get currently activated Key Index
**unsigned short** ***wCheck
Checksum Value of the Current Active Key

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Specify Key Index to use among Keys input into EPP

4.16 EPP_GetActiveKey

## EPP_GetActiveKey

(1) Prototype

**int EPP_GetActiveKey(IN int iKeyMode, OUT int *iKeyIndex, OUT unsigned short *wCheck, OUT int *iMacIndex, OUT unsigned short *wMacCheck)**

(2) Input Parameter

**int** iKeyMode : Specify Key Mode

| | |
|---|---|
| EPP v3.0 : | SINGLE_DES / DUAL_DES / UNIQUE_SINGLE_DES / TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_SINGLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES |
| EPP v5.0 : | TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES |

(3) Output Parameter

**int ***iKeyIndex
Pointer to get currently activated Key Index
**unsigned short ***wCheck
Checksum Value of the Current Active Key
**int ***iMacIndex
Pointer to get currently activated MacKey Index
**unsigned short ***wMacCheck
Checksum Value of the Current Active MacKey

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Get Index and Check Value of currently activated Key into EPP
Get Index and Check Value of currently activated MacKey into EPP

4.17 EPP_GetKeyStatus

## EPP_GetKeyStatus

(1) Prototype

**int EPP_GetKeyStatus(IN int iKeyMode, OUT EPP_KEY_STATUS EppKeyStatus[16])**

(2) Input Parameter

**int** iKeyMode : Specify Key Mode

| | |
|---|---|
| EPP v3.0 : | SINGLE_DES / DUAL_DES / UNIQUE_SINGLE_DES / TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_SINGLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES |
| EPP v5.0 : | TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES |

(3) Output Parameter

**EPP_KEY_STATUS** EppKeyStatus[16]
EPP_KEY_STATUS Structure Array Pointer to get whole status of Key input into EPP

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Get 16 Keys input into EPP

```
typedef struct tag_EPP_KEY_STATUS
{
        int iKeyIndex;        : 0 ~ 15
        BYTE byStatus;        :  0x00 - non Input / 0x01 - PART A Input only / 0x02 - PART B only Input / 0x03 - Input completion
        WORD wCheck;          : Checksum Value of the Key

} EPP_KEY_STATUS
```

## 4.18 EPP_InputKey

**EPP_InputKey**

(1) Prototype

**int EPP_InputKey(IN int iKeyMode, IN int iKeyIndex, IN int iKeyPart, IN int iInputType, OUT unsigned short *wCheck)**

(2) Input Parameter

**int** iKeyMode : Specify Key Mode

EPP v3.0 : SINGLE_DES / DUAL_DES / UNIQUE_SINGLE_DES / TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_SINGLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES

EPP v5.0 : TRIPLE_DES / UNIQUE_TRIPLE_DES / MAC_TRIPLE_DES / TRIPLE_MAC_TRIPLE_DES

**int** iKeyIndex : 0 ~ 11, 15

Specify Index to be saved Key 0 ~ 11 for specifying Master Key, 15 for MAC Key

**int** iKeyPart : PART_A / PART_B

Specify in case of SINGLE_DES, DUAL_DES, MAC_SINGLE_DES

: PART_LEFT_A / PART_RIGHT_A / PART_LEFT_B / PART_RIGHT_B

Specify incase of TRIPLE_DES, MAC_TRIPLE_DES

**int** iInputType : KEY_INPUT / KEY_VERIFY

Specify Input Type of Password to change

(3) Output Parameter

**unsigned short** *wCheck

Effective only if Check Value, InputType which input Key is KEY_VERIFY

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

Fuction will be called when key is pressed on PIN Pad

| | | |
|---|---|---|
| int iId | : | HM_DEV_EPP_MSG |
| int iKind | : | EPP_KEY_PRESSED |
| unsigned char cValue | : | KEY_STAR |

(6) Description

This command is executed only in the Sensitive mode.
Input Key into EPP. You must enter Secure Mode before executing this command.

| Key Mode | PART_A Key | PART_B Key | | EPP |
|---|---|---|---|---|
| SINGLE_DES<br>DUAL_DES<br>MAC_SINGLE_DES<br>MAC Key (index=15) | 8 Byte<br>(16th Key Press) | 8 Byte<br>(16th Key Press) | → | PART_A Key<br>XOR<br>PART_B Key<br>= 8 Byte Master Key |
| | PART_A CheckValue | PART_B CheckValue | | Master Key CheckValue |

| Key Mode | PART_LEFT_A Key | PART_RIGHT_A Key | PART_LEFT_B Key | PART_RIGHT_B Key | | EPP |
|---|---|---|---|---|---|---|
| TRIPLE_DES<br>MAC_TRIPLE_DES | 8 Byte<br>(16th Key Press) | 8 Byte<br>(16th Key Press) | 8 Byte<br>(16th Key Press) | 8 Byte<br>(16th Key Press) | → | PART_A Key<br>XOR<br>PART_B Key<br>= 16 Byte Master Key |
| | | PART_A CheckValue | | PART_B CheckValue | | Master Key CheckValue |

EPP Key Value

| | | | |
|---|---|---|---|
| **1** | **2** | **3** | CANCEL<br>(**D**) |
| **4** | **5** | **6** | CLEAR<br>(**E**) |
| **7** | **8** | **9** | ENTER<br>(**F**) |
| ◁<br>(**A**) | **0** | ▷<br>(**B**) | (None)<br>(**C**) |

| Flow | Application (PART A Key Input) | | |
|---|---|---|---|
| SINGLE_DES | **EPP_InputKey(SINGLE_DES, iIndex, PART_X, KEY_INPUT, &wCheck)** | → | |
| DUAL_DES | *Message KEY_STAR* | ← | *Key Press 1st.* |
| MAC_SINGLE_DES | *Message KEY_STAR* | ← | ... |
| MAC Key (index=15) | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | **EPP_InputKey(SINGLE_DES, iIndex, PART_X, KEY_VERIFY, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | Application (PART B Key Input) | | |
| | **EPP_InputKey(SINGLE_DES, iIndex, PART_B, KEY_INPUT, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | **EPP_InputKey(SINGLE_DES, iIndex, PART_B, KEY_VERIFY, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |

| Flow | Application (PART A Key Input) | | EPP |
|---|---|---|---|
| TRIPLE_DES | **EPP_InputKey(TRIPLE_DES, iIndex, PART_LEFT_A, KEY_INPUT, &wCheck)** | → | |
| MAC_TRIPLE_DES | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | **EPP_InputKey(TRIPLE_DES, iIndex, PART_LEFT_A, KEY_VERIFY, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | **EPP_InputKey(TRIPLE_DES, iIndex, PART_RIGHT_A, KEY_INPUT, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | **EPP_InputKey(TRIPLE_DES, iIndex, PART_RIGHT_A, KEY_VERIFY, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | Application (PART B Key Input) | | EPP |
| | **EPP_InputKey(TRIPLE_DES, iIndex, PART_LEFT_B, KEY_INPUT, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | **EPP_InputKey(TRIPLE_DES, iIndex, PART_LEFT_B, KEY_VERIFY, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | **EPP_InputKey(TRIPLE_DES, iIndex, PART_RIGHT_B, KEY_INPUT, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |
| | **EPP_InputKey(TRIPLE_DES, iIndex, PART_RIGHT_B, KEY_VERIFY, &wCheck)** | → | |
| | *Message KEY_STAR* | ← | *Key Press 1st.* |
| | *Message KEY_STAR* | ← | ... |
| | *Message KEY_STAR* | ← | *Key Press 16th.* |
| | ***Return*** | | |

## 4.19 EPP_GetStatus

**EPP_GetStatus**

(1) Prototype

**int EPP_GetStatus(OUT EPP_STATUS *cStatus)**

(2) Input Parameter

**void**

(3) Output Parameter

**EPP_STATUS** cStatus
Pointer of EPP_STATUS Structure Buffer obtaining EPP Status information

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Get Status information of EPP

```
typedef struct tag_EPP_STATUS
{
        char cTKKeyStatus;           : Status of the loaded TK Key to encrypt NPIN_WKKey
                                       0x00 - NO SET / 0x01 - SET
        char cNPIN_WKKeyStatus;      : Status of the loaded Working Key to set the Removal Protection and Non-PIN Key encryption
                                       0x00 - NO SET / 0x01 - SET
        char cFixMovingStatus;       : Status of the Removal Protection
                                       0x00 - MOVING(Clear) / 0x01 - FIXING(Set)
        char cNPINDataStatus;        : Input status of the Non-PIN Key
                                       0x00 - Enable / 0x01 - Disable
        char cPasswordStatus;        : Status of the Secure Mode Password
                                       0x00 - Default Password(No-Changed) / 0x01 - Changed Password
} EPP_STATUS
```

## 4.20 EPP_GetKeyMode

**EPP_GetKeyMode**

(1) Prototype

**int EPP_GetKeyMode(OUT int *iCurKeyMode)**

(2) Input Parameter

**void**

(3) Output Parameter

**int** *iCurKeyMode
Pointer to get currently activated Key mode

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Get Key mode of currently activated Key into EPP

## 4.21 EPP_InputControl

**EPP_InputControl**

(1) Prototype

**int EPP_InputControl(IN char cEnable)**

(2) Input Parameter

**char** cEnable : NONPIN_ENABLE / NONPIN_DISABLE
Specify the Input of NON-PIN Key

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Control the Input of NON-PIN(Plain text) KEY

4.22 EPP_DownloadTRKey

**EPP_DownloadTRKey**

(1) Prototype

**int EPP_DownloadTRKey()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_NOT_CHANGEDPWD
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

This command is executed only in the Sensitive mode.
Load Transmit Key(TK)
Load the key for data encryption on the EPP device

**Note : You must enter Secure Mode before executing this command.

## 4.23 EPP_InstallKey

**EPP_InstallKey**

(1) Prototype

**int EPP_InstallKey()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA
HM_DEV_NOT_CHANGEDPWD

(5) Message

**void**

(6) Description

This command is executed only in the Sensitive mode.
Load TK Key to encrypt NPIN_WKKey
Loaded Working Key to set the Removal Protection and Non-PIN Key encryption
Set the Removal Protection
Can be Received the NON-PIN Key after set this command

**Note : After complete this command, It is cleared All data when the EPP is arbitrairily removed from ATM.
Please Check the status of the Removal Protection, when it uninstall the EPP
You must enter Secure Mode before executing this command.

4.24 EPP_InstallDefaultKey

## EPP_InstallDefaultKey

(1) Prototype

**int EPP_InstallDefaultKey()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA
HM_DEV_NOT_CHANGEDPWD

(5) Message

**void**

(6) Description

This command is executed only in the Sensitive mode.
Load TK Key to encrypt NPIN_WKKey
Loaded Working Key to set the Removal Protection and Non-PIN Key encryption
After setting this command, you can't get a NON-PIN key yet.
EPP must be changed to Operation mode with the EPP_AuthorizedFixing command.

**Note : After complete this command, It is cleared All data when the EPP is arbitrairily removed from ATM.
Please Check the status of the Removal Protection, when it uninstall the EPP
You must enter Secure Mode before executing this command.

## 4.25 EPP_AuthorizedMoving

**EPP_AuthorizedMoving**

(1) Prototype

**int EPP_AuthorizedMoving()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

This command is executed only in the Sensitive mode.
Uninstall the Removal Protection
Even if the EPP detach from ATM at the status of Moving, All of key data will be kept

**Note : You must enter Secure Mode before executing this command.

## 4.26 EPP_AuthorizedFixing

**EPP_AuthorizedFixing**

(1) Prototype

**int EPP_AuthorizedFixing()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

This command is executed only in the Sensitive mode.
Set the Removal Protection
All of key data will be cleared when the EPP is detached from ATM at the status of Fixing

**Note : You must enter Secure Mode before executing this command.

## 4.27 EPP_GetPCIType

(1) Prototype

**int EPP_GetPCIType()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

PCI Version value
PCI3.0 : 3, PCI5.0 : 5, other version value

(5) Message

**void**

(6) Description

Get the PCI version of the EPP device.
PCI3.0 : 3, PCI5.0 : 5, other : 0 or other version value

## 4.28 EPP_GetLastError

**EPP_GetLastError**

(1) Prototype

**void EPP_GetLastError(OUT unsigned char szErrorCode[5])**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** szErrorCode[5]
Array Pointer which obtains final ErrorCode of EPP

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_SECURE_MODE_ERR
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

it obtains final H/W ErrorCode of EPP Device

4.29 EPP_CallBackRegister

**EPP_CallBackRegister**

(1) Prototype

**void EPP_CallBackRegister(callback_key handler)**

(2) Input Parameter

**callback_key** handler
CallBacked Function

(3) Output Parameter

Fuction will be called when key is pressed on PIN Pad

typedef void (*callback_key)(int iId, int iKind, unsigned char cValue);

| | | |
|---|---|---|
| int iId | : | HM_DEV_EPP_MSG |
| int iKind | : | EPP_KEY_PRESSED |
| unsigned char cValue | : | KEY_0 ~ KEY_9 / KEY_LEFT / KEY_RIGHT / KEY_CANCEL / KEY_CLEAR / KEY_ENTER / KEY_NONE / KEY_STAR |

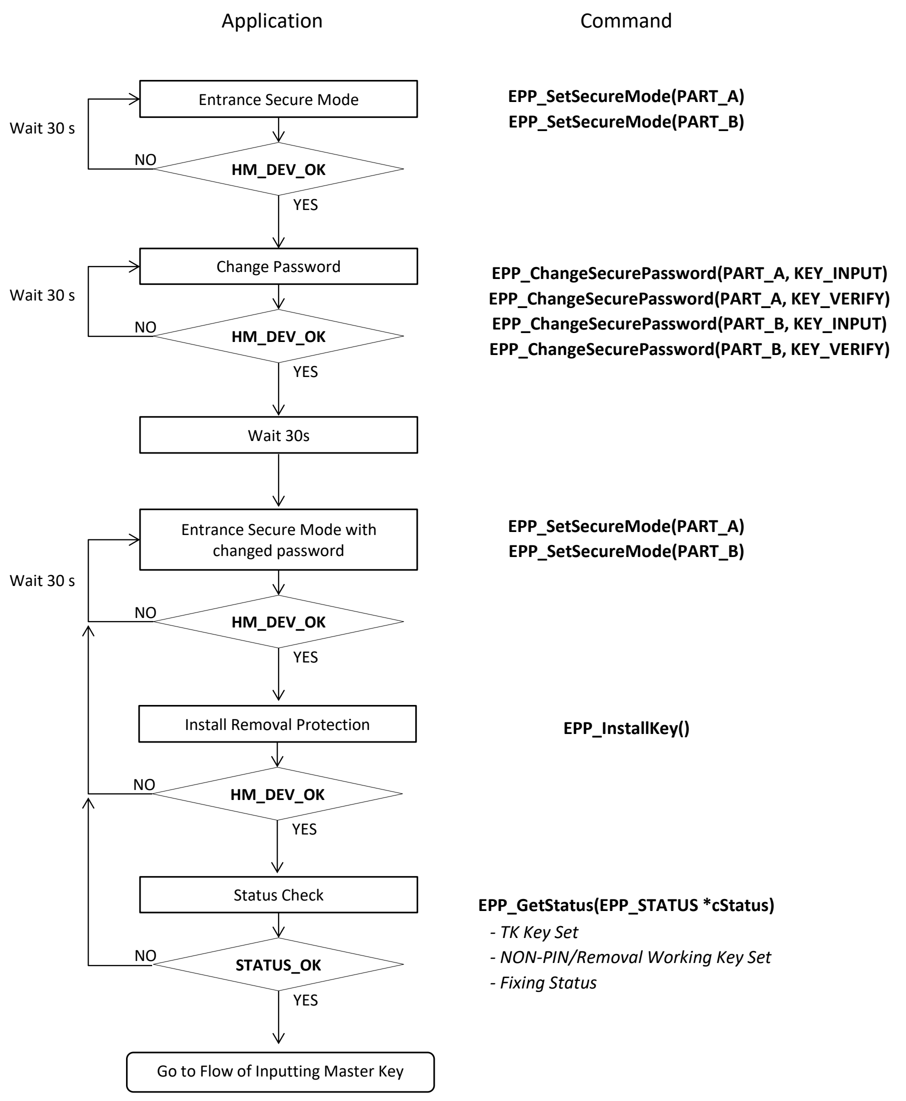
| | | | |
|---|---|---|---|
| 1 (**KEY_1**) | 2 (**KEY_2**) | 3 (**KEY_3**) | CANCEL (**KEY_CANCE** |
| 4 (**KEY_4**) | 5 (**KEY_5**) | 6 (**KEY_6**) | CLEAR (**KEY_CLEAR**) |
| 7 (**KEY_7**) | 8 (**KEY_8**) | 9 (**KEY_9**) | ENTER (**KEY_ENTER**) |
| ◁ (**KEY_LEFT**) | 0 (**KEY_0**) | ▷ (**KEY_RIGHT**) | (None) (**KEY_NONE**) |

* When inputting Password, Key Pressed will be displayed as KEY STAR

## EPP INTALL

4.30 EPP INSTALL FLOW

(1) INSTALL FLOW FROM Status of ClEAR KEY
It is the Flow of install  from the status of cleared key

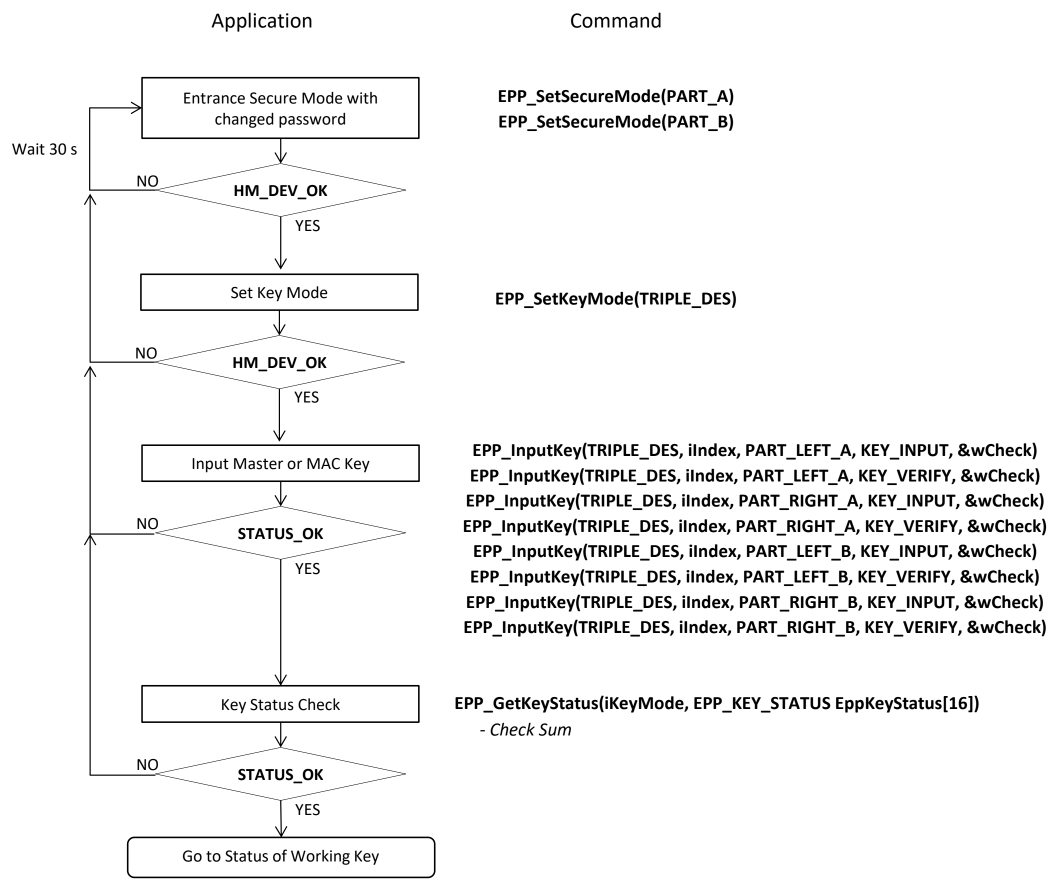

Application
Command
Entrance Secure Mode
Wait 30 s
NO
HM_DEV_OK
YES
EPP_SetSecureMode(PART_A)
EPP_SetSecureMode(PART_B)
Change Password
Wait 30 s
NO
HM_DEV_OK
YES
EPP_ChangeSecurePassword(PART_A, KEY_INPUT)
EPP_ChangeSecurePassword(PART_A, KEY_VERIFY)
EPP_ChangeSecurePassword(PART_B, KEY_INPUT)
EPP_ChangeSecurePassword(PART_B, KEY_VERIFY)
Wait 30s
Entrance Secure Mode with changed password
Wait 30 s
NO
HM_DEV_OK
YES
EPP_SetSecureMode(PART_A)
EPP_SetSecureMode(PART_B)
Install Removal Protection
EPP_InstallKey()
NO
HM_DEV_OK
YES
Status Check
EPP_GetStatus(EPP_STATUS *cStatus)
- TK Key Set
- NON-PIN/Removal Working Key Set
- Fixing Status
NO
STATUS_OK
YES
Go to Flow of Inputting Master Key

* Please refer to the Flow description of each command about detail command processing
Also, The EPP is a state that you can enter the MASTER / MAC key after this flow is completed normally

## EPP INTALL

(2) INPUT MASTER/MAC FLOW

This Flow is a Triple DES sample. Other Key is the same way.



**EPP_SetSecureMode(PART_A)**
**EPP_SetSecureMode(PART_B)**

**EPP_SetKeyMode(TRIPLE_DES)**

**EPP_InputKey(TRIPLE_DES, iIndex, PART_LEFT_A, KEY_INPUT, &wCheck)**
**EPP_InputKey(TRIPLE_DES, iIndex, PART_LEFT_A, KEY_VERIFY, &wCheck)**
**EPP_InputKey(TRIPLE_DES, iIndex, PART_RIGHT_A, KEY_INPUT, &wCheck)**
**EPP_InputKey(TRIPLE_DES, iIndex, PART_RIGHT_A, KEY_VERIFY, &wCheck)**
**EPP_InputKey(TRIPLE_DES, iIndex, PART_LEFT_B, KEY_INPUT, &wCheck)**
**EPP_InputKey(TRIPLE_DES, iIndex, PART_LEFT_B, KEY_VERIFY, &wCheck)**
**EPP_InputKey(TRIPLE_DES, iIndex, PART_RIGHT_B, KEY_INPUT, &wCheck)**
**EPP_InputKey(TRIPLE_DES, iIndex, PART_RIGHT_B, KEY_VERIFY, &wCheck)**

**EPP_GetKeyStatus(iKeyMode, EPP_KEY_STATUS EppKeyStatus[16])**
*- Check Sum*

* You have to process from Set KeyMode if you are in SecureMode.

5. RPU

(1) It describes following interfaces to control Receipt Print Unit

|  | Function | Description |
|---|---|---|
| 1 | RPU USB Type Environment Setting | The environment setting for using RPU USB communication. |
| 2 | RPU_Open | Open Serial Port |
| 3 | RPU_Close | Close Serial Port |
| 4 | RPU_Reset | Reset RPU |
| 5 | RPU_Status | Get RPU's Status |
| 6 | RPU_PrintText | Print the text on receipt |
| 7 | RPU_PrintImage | Print the image on receipt |
| 8 | RPU_PrintImageEx | An extension of the RPU_PrintImage function(Unlimited height) |
| 9 | RPU_DownloadImage | Download the image data to internal memory of RPU |
| 10 | RPU_PrintDownloadImage | Print the image data of internal memory |
| 11 | RPU_CutPaper | Eject the receipt after cutting |
| 12 | RPU_GetParam | Get Parameter(Head,Tail,Pitch) of RPU |
| 13 | RPU_SetParam | Set Parameter(Head,Tail,Pitch) of RPU |
| 14 | RPU_GetImageEndLine |  |
| 15 | RPU_SetImageEndLine |  |
| 16 | RPU_GetSRAMType | Get SRAM capacity of RPU |
| 17 | RPU_GetLastError | Get final H/W Error Code of RPU |
| 18 | RPU_UsbOpen | Open Usb |
| 19 | RPU_UsbClose | Close Usb |
| 20 | RPU_FWDownload | Firmware file download (main firmware, boot firmware) Only USB type |
| 21 | ESC Command | String(Esc Command) Special Function |
|  |  |  |

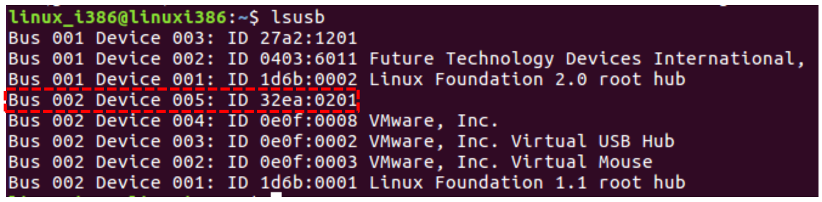
|  | CallBack Function | Description |
|---|---|---|
| 22 | RPU_CallBackDLProgress | Send a message to the registered function whenever firmware download progress. |

5.1 RPU USB Type Environment Setting

## CIS Environment Setting

(1) Check USB Port Recognition

First check with the lsusb command to see if the receipt printer(RPU) is connected (vid 0x32ea, pid 0x0201 or 0x0204)

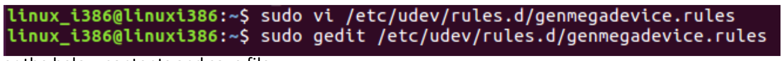


```
linux_i386@linuxi386:~$ lsusb
Bus 001 Device 003: ID 27a2:1201
Bus 001 Device 002: ID 0403:6011 Future Technology Devices International,
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 005: ID 32ea:0201
Bus 002 Device 004: ID 0e0f:0008 VMware, Inc.
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

(2) Enable RPU USB device user account.

1) Create a rules file in /etc/udev/rules.d. ex) sudo touch /etc/udev/rules.d/genmegadevice.rules

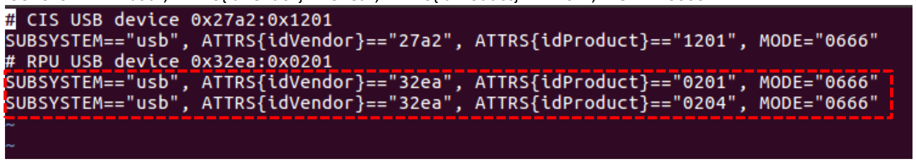
ex)



```
linux_i386@linuxi386:~$ sudo vi /etc/udev/rules.d/genmegadevice.rules
linux_i386@linuxi386:~$ sudo gedit /etc/udev/rules.d/genmegadevice.rules
```

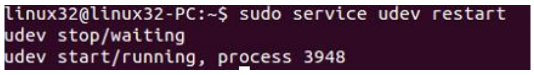
Enter the below contents and save file.

SUBSYSTEM=="usb", ATTRS{idVendor}=="32ea", ATTRS{idProduct}=="1201", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="32ea", ATTRS{idProduct}=="1204", MODE="0666"



```
# CIS USB device 0x27a2:0x1201
SUBSYSTEM=="usb", ATTRS{idVendor}=="27a2", ATTRS{idProduct}=="1201", MODE="0666"
# RPU USB device 0x32ea:0x0201
SUBSYSTEM=="usb", ATTRS{idVendor}=="32ea", ATTRS{idProduct}=="0201", MODE="0666"
SUBSYSTEM=="usb", ATTRS{idVendor}=="32ea", ATTRS{idProduct}=="0204", MODE="0666"
~
~
```

2) Restart Service -> sudo service udev restart



```
linux32@linux32-PC:~$ sudo service udev restart
udev stop/waiting
udev start/running, process 3948
```

3) Disconnect the USB cable of the connected RPU and reconnect it.

## 5.2 RPU_Open

**RPU_Open**

(1) Prototype

**int RPU_Open(IN const char* szPortName,  OUT char *ver)**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to RPU  ( Ex) "/dev/ttyS4" )

(3) Output Parameter

**char** *ver
Array Pointer to obtain the F/W version of RPU

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open Serial Port of RPU
Get Firmware Version of RPU.

If serial port connection fails, try USB connection (PID: 0x0201, 0x0204)

5.3 RPU_Close

**RPU_Close**

1. Introduction

**void RPU_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close Serial Port of RPU

5.4 RPU_Reset

**RPU_Reset**

(1) Prototype

**int RPU_Reset()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Check the status of RPU Device, and
If RPU is normal, then do test-printing on the receipt and cutting

## 5.5 RPU_Status

(1) Prototype

**int RPU_Status(OUT RPU_STATUS *RpuStatus)**

(2) Input Parameter

**void**

(3) Output Parameter

**RPU_STATUS** *RpuStatus
RPU_STATUS Structure Buffer's Pointer to get the status information of RPU

(4) Return Value

HM_DEV_OK
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Get the status information of RPU

```
typedef struct tag_RPU_STATUS
{
        char LineStatus;          :   HM_DEV_CONNECT/ HM_DEV_DISCONNECT
                                      Displays the connection status with RPU Device
        char PaperLoad;           :   RPU_NORMAL / RPU_NO_SET
                                      Display whether the receipt paper is loaded or not
        char PaperTphLoad;        :   RPU_NORMAL / RPU_NO_SET
                                      Display whether the receipt paper is loaded or not
        char PaperNearEnd;        :   Option
                                      Display whether the receipt is near end or not
        char TphLever;            :   RPU_NORMAL / RPU_NO_SET
                                      Display whether the reed lever is mounted or not
        char PaperJam;            :   RPU_NORMAL / RPU_REMAIN
                                      Display whether the paper remains or not betweeen Printing Part and Dispensing Part
        char CutterHome;          :   RPU_NORMAL / RPU_NO_SET
                                      Display whether the cutter is loaded or not
        char PaperNormal;         :   Reserve

} RPU_STATUS
```

5.6 RPU_PrintText

**RPU_PrintText**

(1) Prototype

**int RPU_PrintText(IN char *PrintText)**

(2) Input Parameter

**char ***szPrintText
Array Pointer to print Text

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Print specified text strings on ther receipt
ASCII code shall be used for printing
Line change shall be based upon CR(0x0D) + LF(0x0A) format
Multi line printing shall be available by adding CR+LF between Text Strings
Can use the ESC command. Ref 5.20
Max number of printing per Line shall be as follows :
- 2 Inch Receipt : 35 Character
- 3 Inch Receipt : 40 Character

Ex) Normal Text Print
```
    RPU_PrintText("Test Print Line 1 \r\nTest Print Line 2 \r\nTest Print Line 3 \r\n");
```
Ex) Horizontal 2 Big (1B 30 32), Vertical Normal (1B 31 31) Text Print (With ESC command)
```
    RPU_PrintText("[0x1B 0x30 0x32 0x1B 0x31 0x31]Test Print Line 1 \r\n");
```
Ex) Horizontal 2 Big (1B 30 32), Vertical 2 Big (1B 31 32) Text Print (With ESC command)
```
    RPU_PrintText("[0x1B 0x30 0x32 0x1B 0x31 0x32]Test Print Line 1 \r\n");
```

## 5.7 RPU_PrintImage

**RPU_PrintImage**

(1) Prototype

**int RPU_PrintImage(IN int nLeftMargin, IN char *fname)**

(2) Input Parameter

**int** nLeftMargin
Specify Left margin
**char** *fname
Image File Name's Pointer to print

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Print specified image file on the receipt
Available Image File to print shall be as follows:
- Support Image Format : BMP, JPG, PNG, TIF
    Convert to BMP format in 2 colors (black / white) and print.
- Max Width Pixel
    2 Inch : 488 Pixel
    3 Inch : 576 Pixel

- Convert BMP maximum Image Size = (Width Pixel / 8 * Hight Pixel)Byte + 62 Byte <= **131,056** Byte (128KB - 16Byte)
    The maximum image width and height of the converted BMP ( 576 X 1638)
    If maximum size, it takes max 12 second to print out

** 3 inch Model RPU only be available

5.8 RPU_PrintImageEx

## RPU_PrintImageEx

(1) Prototype

**int RPU_PrintImageEx(IN int nLeftMargin, IN char *fname)**

(2) Input Parameter

**int** nLeftMargin
Specify Left margin
**char** *fname
Image File Name's Pointer to print

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Print specified image file on the receipt
Available Image File to print shall be as follows:
- Support Image Format : BMP, JPG, PNG, TIF
  Convert to BMP format in 2 colors (black / white) and print.
- Max Width Pixel
  2 Inch : 488 Pixel
  3 Inch : 576 Pixel
- File Size = File size and height are unlimited
  (but if the sum of the left margin and the image width exceeds the maximum, HM_DEV_IMAGE_ERROR is returned).

- The maximum image width and height of the converted BMP that can be printed once. ( 576 X 1638)

- Convert BMP maximum Image Size = (Width Pixel / 8 * Hight Pixel)Byte + 62 Byte <= **131,056** Byte (128KB - 16Byte)
- In addition, if the maximum height that can be printed once is exceeded, the image is divided by the maximum height and printed several times.
- It takes a maximum of 12 seconds for a maximum size of printed once, so it takes longer time to print several times.

** 3 inch Model RPU only be available

## 5.9 RPU_PrintDownloadImage

**RPU_DownloadImage**

(1) Prototype

**int RPU_DownloadImage(IN int index, IN char *fname)**

(2) Input Parameter

**int** iIndex : 0 / 1
Index to save Image
**char ***szFileName
Image File Name's Pointer to download to RPU

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Download specified image file to RPU's Flash ROM
Available image files to download shall be as follows:
- 2 Color(Black/White) BMP Format File
- Max Width Pixel
    2 Inch : 488 Pixel
    3 Inch : 576 Pixel
- File Size = (Width Pixel / 8 * Hight Pixel)Byte + 62 Byte <= **32,704** Byte

Total 2 image files shall be available to download to Index( 0 and 1)
If maximum size, it takes max 4 seconds to download

** 3 inch Model RPU only be available

## 5.10 RPU_PrintDownloadImage

**RPU_PrintDownloadImage**

(1) Prototype

**int RPU_PrintDownloadImage(IN int index, IN int LeftMargin)**

(2) Input Parameter

**int** iIndex : 0 / 1
Downloaded image index to Print
**int** LeftMargin
Specify Left margin

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Print the image date saved in RPU Flash ROM

** 3 inch Model RPU only be available

5.11 RPU_CutPaper

**RPU_CutPaper**

(1) Prototype

**int RPU_CutPaper()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Eject the receipt after cutting

5.12 RPU_GetParam

## RPU_GetParam

(1) Prototype

**int RPU_GetParam(OUT int *iHead, OUT int *iTail, OUT int *iPitch)**

(2) Input Parameter

**void**

(3) Output Parameter

**int ***iHead : 0 / 254 (A value of 1 indicates 0.125 mm)
Get the head(top) margin value of the paper
**int ***iTail : 0 / 254 (A value of 1 indicates 0.125 mm)
Get the tail(buttom) margin value of the paper
**int ***iPitch : 0 / 1
Gets the characters pitch value

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_BUSY
HM_DEV_INVALID_DATA
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_NOTSUPPORT

(5) Message

**void**

(6) Description

Get the values for the print position (Head margin, Tail margin) and the character pitch.

Firmware version is supported only for RPU81 or later.
If it is older version, HM_DEV_NOTSUPPORT is returned.
Character pitch applies only to RPU90 and later versions, and older versions are ignored.

## 5.13 RPU_SetParam

**RPU_SetParam**

(1) Prototype

**int RPU_SetParam(IN int iHead, IN int iTail, IN int iPitch)**

(2) Input Parameter

**int** iHead : 0 / 254 (A value of 1 indicates 0.125 mm)
Set the head(top) margin value of the paper
**int** iTail : 0 / 254 (A value of 1 indicates 0.125 mm)
Set the tail(buttom) margin value of the paper
**int** iPitch : 0 / 1
Set the character pitch value
0 : The character pitch is fixed to the default setting
1 : The character pitch changes according to the scale

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_BUSY
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA
HM_DEV_NOTSUPPORT

(5) Message

**void**

(6) Description

Set the values for the print position (Head margin, Tail margin) and the character pitch.

Firmware version is supported only for RPU81 or later.
If it is older version, HM_DEV_NOTSUPPORT is returned.
Character pitch applies only to RPU90 and later versions, and older versions are ignored.

5.14 RPU_GetImageEndLine

## RPU_GetImageEndLine

(1) Prototype

**int RPU_GetImageEndLine(int *nEndLine)**

(2) Input Parameter

**void**

(3) Output Parameter

**int ***nEndLine
End line margin size after printing the image.

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_BUSY
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA
HM_DEV_NOTSUPPORT

(5) Message

**void**

(6) Description

Get the size of the end line margin after printing the image
The set value remains unchanged unless the firmware changes or the set value is changed.

Firmware version is supported only for RPU92 or later.
If it is older version, HM_DEV_NOTSUPPORT is returned.

## 5.15 RPU_SetImageEndLine

**RPU_SetImageEndLine**

(1) Prototype

**int RPU_SetImageEndLine(int nEndLine)**

(2) Input Parameter

**int** nEndLine : 0 / 254 (A value of 1 indicates 0.125 mm)
End line margin size after printing the image.

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_BUSY
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_INVALID_DATA
HM_DEV_NOTSUPPORT

(5) Message

**void**

(6) Description

Set the size of the end line margin after printing the image.
The set value remains unchanged unless the firmware changes or the set value is changed.

Firmware version is supported only for RPU92 or later.
If it is older version, HM_DEV_NOTSUPPORT is returned.

## 5.16 RPU_GetSRAMType

(1) Prototype

**int RPU_GetSRAMType(void)**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

RPU_256SRAM (0) or RPU_512SRAM (1)

(5) Message

**void**

(6) Description

Returns the SRAM capacity of the RPU Device. (F/W Version RPU94 or higher)

If the SRAM is 256KB, it returns RPU_256SRAM (0). (Default)
If the SRAM is 512 KB, it returns RPU_512SRAM (1).

## 5.17 RPU_GetLastError

**RPU_GetLastError**

(1) Prototype

**void RPU_GetLastError(OUT char *errmsg)**

(2) Input Parameter

**void**

(3) Output Parameter

**char *** errmsg
Array Pointer to get final ErrorCode of RPU

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Get final H/W ErrorCode of RPU Device

## 5.18 RPU_UsbOpen

**RPU_UsbOpen**

(1) Prototype

**int RPU_UsbOpen(IN const unsinged short sProductID,  OUT char *ver)**

(2) Input Parameter

**const unsinged short** *sProductID
Usb of connecting to RPU Product ID  ( Ex) 0x0201, 0x0204 : 3Inch UsbType RPU)

(3) Output Parameter

**char** *ver
Array Pointer to obtain the F/W version of RPU

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_USB_COMM_FAILED
HM_DEV_OPENPORTFAIL
HM_DEV_NOT_READY

(5) Message

**void**

(6) Description

Open Usb of RPU
Get Firmware Version of RPU.

5.19 RPU_UsbClose

**RPU_UsbClose**

1. Introduction

**void RPU_UsbClose()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close Usb of RPU

## 5.20 RPU_FWDownload

**RPU_FWDownload**

(1) Prototype

**int RPU_FWDownload(int iFwType, char *szFwFile);**

(2) Input Parameter

**int iFwType** : (0/1 : 0 - main firmware, 1 - boot firmware)
char *szFwFile : Full path of firmware file (.bin)

(3) Output Parameter

**Void**

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_USB_COMM_FAILED
HM_DEV_NOTSUPPORT
HM_DEV_NOT_READY

(5) Message

**void**

(6) Description

Firmware file download (main firmware, boot firmware)

If the firmware type is boot firmware, it is applied when the RPU power is turned on after downloading is complete.
If the firmware type is main firmware, the RPU USB connection is disconnected after the download is complete.
Try to connect again after about 5 seconds.

5.21 ESC Command

## String(Esc Command) Special Function

(1) BAR code function support

(Add as below to PRINT DATA when ASCII DATA PRINT (including PAGE MODE) command)

A) ESC, h, n (0x1B, 0x68): Specify the barcode height.

n is a value of n and specifies the length n * 0.125mm

B) ESC, d, n (0x1B, 0x64): Specify the direction

n: '0' (0x30) landscape printing

C) ESC, m, n (0x1B, 0x6D): Specify the starting position.

n is the starting position plus 0x20 and is actually (n-0x20) * 1mm apart

If n is 'z' (0x7A), the center is aligned

D) ESC, w, n (0x1B, 0x77): Set the barcode width

n: ‘1’ (0x31) narrow element (0.125 mm) wide element (0.375 mm)

‘2’ (0x32) narrow element (0.25 mm) wide element (0.75 mm)

‘3’ (0x33) narrow element (0.375 mm) wide element (1.125 mm)

E) ESC, p, n (0x1B, 0x70): Specify whether to print barcode HRI characters.

n: no argument of ‘0’ (0x30)

'1' (0x31) barcode printing box

F) ESC, i, n1, n2, d0 to dn (0x1B, 0x69): Specify the barcode type and length.

n1: Barcode type '(default:' 1 ')

| | |
|---|---|
| '0' (0x30): code93 | ‘1’ (0x31): code128 A |
| ‘2’ (0x32): code39 | ‘3’ (0x33): code bar |
| ‘4’ (0x34): Interleaved 2 of 5 | ‘5’ (0x35): code128 B |
| ‘6’ (0x36): code128 C | |

n2: Barcode length + 0x20 is expressed as (n2-0x20) is d0 ~ dn number.

d0 ~ dn: barcode string

G) ESC, q, n1, n2, n3, n4, d0 to dn (0x1B, 0x71): Set QR code printing.

n1: Specifies the size. The range is 1 to 8 (0x31 to 0x38).

n2: Designate starting position. The range is 0 to 60 (0x20 to 0x5C).

Specify the setting value * 0.125mm position from the left

n3: barcode length 1. The range is 0 to 95 (0x20 to 0x7F).

n4: barcode length 2. The range is 0 to 95 (0x20 to 0x7F).

※. Barcode length is up to 190 with n3 + n4, see error code if mismatch

Ex) If the barcode length is 100 n3 n4: 0x7F 0x25

If 20, n3 n4: 0x34 0x20

d0 ~ dn: barcode string

2) Implement font directly

ESC, M d0 ~ d48 (0x1B, 0x4D): Implement font. d0 ~ d48: font string (48 byte)

3) Horizontal Zoom

ESC, 0, n (0x1B, 0x30, n): n is 1 to 3 (0x31 to 0x33), n times enlarged.

4) Vertical enlargement

ESC, 1, n (0x1B, 0x31, n): n is 1 to 3 (0x31 to 0x33), n times enlarged.

5) Thick

ESC, B (0x1B, 0x42): Makes the text darker.

ESC, b (0x1B, 0x62): Deselects the text in bold.

6) Reversed phase

ESC, R (0x1B, 0x52): reverses the text.

ESC, r (0x1B, 0x72): Reverse the text.

7) Underline

ESC, U (0x1B, 0x55): Underlines the text.

ESC, u (0x1B, 0x75): Underline the text.

8) Align Text

ESC, C (0x1B, 0x43): Set the alignment of text

ESC, c (0x1B, 0x63): Un align the text

## 5.22 RPU_CallBackDLProgress

**RPU_CallBackDLProgress**

(1) Prototype

**void RPU_CallBackDLProgress(callback_dlprogress handler)**

(2) Input Parameter

**callback_dlprogress** handler
CallBacked Function

(3) Output Parameter

Downloading the firmware with RPU_FWDownload() invokes the function.

typedef void (*callback_dlprogress)(int iMode, int iSent, int iTotal);

| | | |
|---|---|---|
| int iMode | : | Download status.  0 : Download Start, 1 : Downloading, 2: Download Finish. |
| int iSent | : | The size of the data downloaded from the firmware file. |
| int iTotal | : | The total size of the firmware file |

## 6. MCR

(1) It describes following interfaces to control Magnetic/Chip Card Reader

| Function | | Description |
|---|---|---|
| 1 | MCR_Open | Open Serial Port |
| 2 | MCR_Close | Close Serial Port |
| 3 | MCR_Status | Get MCR's Status |
| 4 | MCR_MSRead | Get MS Data of Card |
| 5 | MCR_ICReset | Connect IC Chip |
| 6 | MCR_ICDirect | Communicate with IC Chip |
| 7 | MCR_Eject | Eject Card |
| 8 | MCR_SetLatchControl | Control the card latch |
| 9 | MCR_MSClear | Clear the MS data stored |
| 10 | MCR_GetLastError | Get final H/W Error Code of MCR |
| | | |

6.1 MCR_Open

## MCR_Open

(1) Prototype

**int MCR_Open(IN const char* szPortName, OUT unsigned char szVerInfo[10])**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to MCR ( Ex) "/dev/ttyS0" )

(3) Output Parameter

**unsigned char** szVerInfo[10]
Array Pointer to obtain the F/W version of MCR

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open Serial Port with MCR

6.2 MCR_Close

**MCR_Close**

1. Introduction

**void MCR_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close Serial Port of MCR

## 6.3 MCR_Status

**MCR_Status**

(1) Prototype

**int MCR_Status(OUT MCR_STATUS *sts)**

(2) Input Parameter

**void**

(3) Output Parameter

**MCR_STATUS** *sts
MCR_STATUS Structure Buffer's Pointer to get the status information of MCR

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Get the status information of MCR

```
typedef struct tag_MCR_STATUS
{
        unsigned char iLineStatus;    :   DEV_CONNECT / DEV_DISCONNECT
                                          Displays the connection status with MCR Device
        unsigned char iStatus;        :   CARD PRESENT / NOPRESENT / LATCHED
                                          Displays the input status of Card
        unsigned char  iMsStatus;     :   MS_NOPRESENT / MS_PRESENT
                                          Displays the MS Data  status
} MCR_STATUS
```

6.4 MCR_MSRead

**MCR_MSRead**

1. Introduction

**int MCR_MSRead(OUT MCR_MS_DATA *McrMsData)**

(2) Input Parameter

**void**

(3) Output Parameter

**MCR_MS_DATA ***McrMsData
MCR_MS_DATA Structure Buffer's Pointer to get the MS data

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Get the MS Data from MCR

```
typedef struct tag_MCR_MS_DATA
{
    int iTrack1Len;                  :    Data Length for Track 1
    unsigned char szTrack1[200];     :    MS Data for Track 1
    int iTrack2Len;                  :    Data Length for Track 2
    unsigned char szTrack2[200];     :    MS Data for Track 2
    int iTrack3Len;                  :    Data Length for Track3
    unsigned char szTrack3[200];     :    MS Data for Track 3

}MCR_MS_DATA;
```

* Return Value will be treated as a HM_DEV_HW_ERR when read error occurs at all Tracks at reading MS

6.5 MCR_ICReset

**MCR_ICReset**

1. Introduction

**int MCR_ICReset(OUT int *iAtrLen, OUT unsigned char *byAtr)**

(2) Input Parameter

**void**

(3) Output Parameter

**int** *iAtrLen
Length of ATR
**unsigned char** *byAtr
Array Pointer to get ATR Data

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Supply the power to IC Chip.
Get the ATR data from IC Chip

6.6 MCR_ICDirect

## MCR_ICDirect

1. Introduction

**int MCR_ICDirect(int iIcSendLen, unsigned char *szIcSend, int *iIcRecvLen, unsigned char *szIcRecv)**

(2) Input Parameter

**int** iIcSendLen
Length of IC send data
**unsigned char** *szIcSend
Array pointer of data to send to IC

(3) Output Parameter

**int** *iIcRecvLen
Length of IC receive data
**unsigned char** *szIcRecv
Array Pointer to get IC Data

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

- This Command is controled at Emv Kernel

This is a command for operation under ISO7816.
User can handle all IC Cards Conforming to ISO 7816-4 and T=0, T=1

* Note : Send and Receive Data Packet refer to Data block of Command Packet specified in ISO 7816-4 APDU

6.7 MCR_Eject

## MCR_Eject

1. Introduction

**int MCR_Eject()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Eject the Card

## 6.8 MCR_SetLatchControl

**MCR_SetLatchControl**

(1) Prototype

**int MCR_SetLatchControl(IN int iUnLock)**

(2) Input Parameter

**int** iUnLock : 0 (Latch Card) / 1 (Unlatch Card)
Set whether or not to latch when inserting the card

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_TIMEOUT
HM_DEV_NOT_READY
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

If it is 0, it is latched when the card is inserted. If it is 1, it is not latched when the card is inserted.
If it is 1, it is not latched even if it is set to 0 while the card is inserted.

6.9 MCR_MSClear

**MCR_MSClear**

1. Introduction

**int MCR_MSClear()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Clear the MS data stored in MCR

## 6.10 MCR_GetLastError

**MCR_GetLastError**

(1) Prototype

**void MCR_GetLastError(OUT unsigned char szErrorCode[5])**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** szErrorCode[5]
Array Pointer which obtains final ErrorCode of MCR

(4) Return Value

**void**

(5) Message

**void**

(6) Description

it obtains final H/W ErrorCode of MCR Device

7. SIU

(1) It describes following interfaces to control Sensor and Indicators Unit

| Function | | Description |
|---|---|---|
| 1 | SIU_Open | Open Driver and, start status check theread |
| 2 | SIU_Close | Close Driverand, end status check theread |
| 3 | SIU_Status | Get status of SIU |
| 4 | SIU_Flicker | On and Off Flicker |
| 5 | SIU_Reset | All off Flicker |
| 6 | SIU_FeedAction | Feed action for check exit module for the check scanner. |
| 7 | SIU_SetFlickerColor | Set the color of each flicker. |
| 8 | SIU_SetLED | Set the color of each LED. |

7.1 SIU_Open

**SIU_Open**

(1) Prototype

**int SIU_Open(IN const char* szPortName, OUT unsigned char szVerInfo[10])**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to SIU ( Ex) "/dev/ttyS5" )

(3) Output Parameter

**unsigned char** szVerInfo[10]
Array Pointer to obtain the F/W version of SIU

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open each Control Driver for SIU
Start status check thread for SIU

7.2 SIU_Close

**SIU_Close**

1. Introduction

**void SIU_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close status check thered for SIU
Close each control device for SIU

## 7.3 SIU_Status

**SIU_Status**

(1) Prototype

**void SIU_Status(OUT SIU_STATUS *SiuStatus)**

(2) Input Parameter

**void**

(3) Output Parameter

**SIU_STATUS** *SiuStatus
SIU_STATUS Structure Buffer's Pointer to get SIU's status information

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Get SIU's status information

```
typedef struct tag_SIU_STATUS
{
        unsigned char chDoor1;       :  SIU_CLOSE / SIU_OPEN
                                        Open/Close status of Door
        unsigned char chDoor2;       :  SIU_CLOSE / SIU_OPEN
                                        Open/Close status of Door
        unsigned char chDoor3;       :  SIU_CLOSE / SIU_OPEN
                                        Open/Close status of Door
        unsigned char chDoor4;       :  SIU_CLOSE / SIU_OPEN
                                        Open/Close status of Door
        unsigned char chDoor5;       :  SIU_CLOSE / SIU_OPEN
                                        Open/Close status of Door
        unsigned char chDoor6;       :  SIU_CLOSE / SIU_OPEN
                                        Open/Close status of Door
        unsigned char chDoor7;       :  SIU_CLOSE / SIU_OPEN
                                        Open/Close status of Door
        unsigned char chAudioJack;   :  SIU_NOT_PRESENT / SIU_PRESENT
                                        Connection Status of Earphone to Audio Output Socket
} SIU_STATUS
```

7.4 SIU_Flicker

## SIU_Flicker

(1) Prototype

**int SIU_Flicker(IN int iDev, IN char bOnOff)**

(2) Input Parameter

**int** iDevice : FLICKER_ALL / FLICKER_RPU / FLICKER_MCR / FLICER_CDU / FLICKER_EPP
Specify the device to control flicker
**char** bOnOff : FLICKER_ON / FLICKER_OFF / FLICKER_ON_CONTINUE
Specify flicker ON and OFF

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_TIMEOUT
HM_DEV_NOT_READY
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

On and Off the specified flicker

## 7.5 SIU_Reset

**SIU_Reset**

(1) Prototype

**int SIU_Reset()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Flicker All OFF

## 7.6 SIU_FeedAction

**SIU_FeedAction**

(1) Prototype

**int SIU_FeedAction()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Feed action for check exit module for the check scanner.
If there is no Exit Module, HM_DEV_TIMEOUT(-9) is returned.

## 7.7 SIU_SetFlickerColor

**SIU_SetFlickerColor**

(1) Prototype

**int SIU_SetFlickerColor(RGB_LIST RgbList[10])**

(2) Input Parameter

**RGB_LIST** RgbList[10]
Store the RGB color of each flicker in an array (10 fixed)

```
typedef struct tag_RGB_LIST {
    int iRed;
    int iGrn;
    int iBlu;
} RGB_LIST;
```

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_HW_ERR
HM_DEV_TIMEOUT
HM_DEV_NOTSUPPORT

(5) Message

**void**

(6) Description

Set the color of each flicker.
Only works on the ASIC board. If it is not the ASIC board, HM_DEV_NOTSUPPORT is returned.

7.8 SIU_SetLED

## SIU_SetLED

(1) Prototype

**int SIU_SetLED(int iActMode, RGB_LIST RgbList[6])**

(2) Input Parameter

**int iActMode** : LED_ALL_OFF(0) / LED_ALL_ON(1) / LED_ALL_RANDOM(2) / LED_ALL_DEFAULT(3)
Specifies the action of the LED.

**RGB_LIST** RgbList[6]
Store the RGB color of each LED in an array (6 fixed)

```
typedef struct tag_RGB_LIST {
    int iRed;
    int iGrn;
    int iBlu;
} RGB_LIST;
```

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_HW_ERR
HM_DEV_TIMEOUT
HM_DEV_NOTSUPPORT

(5) Message

**void**

(6) Description

Set the color of each LED.
Only works on the ASIC board. If it is not the ASIC board, HM_DEV_NOTSUPPORT is returned.

[Note]
LED_ALL_RANDOM(2) and LED_ALL_DEFAULT(3) commands operate when the LED is ON.
To use the above two commands, first execute LED_ALL_ON(1) command and then execute the command.

[Default]

| | |
|---|---|
| LED1 : Left – Top | LED2 : Left – Middle |
| LED3 : Left – Bottom | LED4 : Right – Top |
| LED5 : Right – Middle | LED6 : Right – Bottom |

[UK2 Model]

| | |
|---|---|
| LED1 : Left – Top | LED2 : Left – Bottom |
| LED3 : Not use | LED4 : Right - Top |
| LED5 : Right – Bottom | LED6 : Not use |

8. EMV Kernel

(1) It is a EMV kernel which contains the functionality required to perform an EMV transaction.

| | Function | Description |
|---|---|---|
| 1 | emvkrnl_parameter_init | EMV Initialization and Parameter File Load(CAPublicKeys.ini, EMVParam.ini) |
| 2 | emvkrnl_set_term | EMV Parameters Setting for Transaction Initialization |
| 3 | emvkrnl_application_selection | Application(VISA/MASTER/AMEX...) Selection supported by IC card and Terminal |
| 4 | emvkrnl_get_candidateList | Read Candidate List supported by IC card and Terminal |
| 5 | emvkrnl_read_application | Read data from Selected Application |
| 6 | emvkrnl_processing_restrictions | Verify Processing Restrictions (version, usage control, valid date, expiration date) |
| 7 | emvkrnl_offline_data_authentication | Authenticate IC card certificate with public key (This process may pass in |
| 8 | emvkrnl_cardolder_verification | Cardholder Verification |
| 9 | emvkrnl_transaction_type_select | Select Transaction Type |
| 10 | emvkrnl_account_type_select | Select Account Type |
| 11 | emvkrnl_set_purchase_amount | Set Amount |
| 12 | emvkrnl_terminal_risk_management | Terminal Risk Management: Process that offline transaction switches to |
| 13 | emvkrnl_terminal_action_analysis | Terminal Action Analysis: Based on previous transaction, determine |
| 14 | emvkrnl_online_process | Online Received Message Process |
| 15 | emvkrnl_card_action_analysis | Online Approval Process/Card Action Analysis |
| 16 | emvkrnl_unable_online | Unable Online Process |
| 17 | emvkrnl_online_referral | Online REFERRAL Process |
| 18 | emvkrnl_online_reject | Online Reject Process |
| 19 | emvkrnl_online_advice | Online Advice Process |
| 20 | emvkrnl_online_confirm | Online CONFIRM Process |
| 21 | emvkrnl_online_reversal | Online REVERSAL Process |
| 22 | emvkrnl_completion | EMV Transaction Completion Process |
| 23 | emvkrnl_read_dataEl | Read EMV Data Element Value |
| 24 | emvkrnl_set_config | Set configrations to manage EMV kernel |
| 25 | emvKrnl_datacapture_clear | Clear data captured |
| 26 | emvKrnl_DataCapture | Capture or cancel data for the offline transaction |
| | | |

(2) Kernel Library
static library : (32bit) /usr/local/lib/libgenemv_api.a, (64bit) /usr/local/lib/libgenemv64_api.a
shard library : (32bit) /usr/local/lib/libgenemv.so, (64bit) /usr/local/lib/libgenemv64.so

(3) Header file
/usr/local/include/genmegadevice/genemv_api.h

(4) EMV Kernel Parameter File
1. Parameter file location setting file: /etc/genmegadevice/genmegadevice.cfg
default location : /etc/genmegadevice/emvparam
2. Parameter Files
CAPublicKeys.ini, EMVParam.ini

## Return Code

8.1 Return Code

```
#define    GEN_CANCEL               -2     //CANCEL, 거래종료
#define    GEN_ERROR                -1     //Error
#define    GEN_SUCCESS              0x00   //TC(success)
#define    GEN_DECLINED             0x02   //Declined, 거래거절
#define    GEN_ARQC                 0x03   //ARQC, 온라인 승인요청
#define    GEN_REJECT               0x10   //Online Reject
#define    GEN_REFERRAL             0x11   //Online referral
#define    GEN_UNONLINE             0x12   //Unable Online
#define    GEN_REVERSAL             0x20   //Reversal
#define    GEN_CONFIRM              0x21   //Confirm
#define    GEN_ADVICE               0x22   //Advice 요청
#define    GEN_RDECLINE             0x23   //referral decline
#define    GEN_INIT_VALUE           0x40   //Initial Value
#define    GEN_NOINIT_VALUE         0x41   //No Initial Value
#define    GEN_CONTINUE             0x42   //Continue
#define    GEN_INVALID_PARAM        -3     //Invalid parameter
#define    GEN_INVALID_PASSWORD     -4     //Invalid password
#define    GEN_REQ_RETRY            0x61   //Need to retry the method
#define    GEN_INVALID_PASSWORD     0x62   //Need to select AID
#define    GEN_REQ_AIDSELECT        0x63   //Need to password
#define    GEN_REQ_CONFIRM          0x64   //Need to confirm
```

8.2 EMV Method

**emvkrnl_parameter_init**

### 8.2.1 emvkrnl_parameter_init

(1) Prototype

**int emvkrnl_parameter_init ( int terminalType, const char* termID, int iDeviceID)**

(2) Input Parameter

**int terminalType**
20 : unattended financial Online
36 : unattended merchant Online
37 : unattended merchant Online / Offline

**const char* termID**
Terminal ID of the machine

**int iDeviceID**
Device ID to use EMV.
GEN_DEVID_MCR(0) : Use EMV as an MCR device.
GEN_DEVID_CIS(1) : Use EMV as an CIS device.

(3) Output Parameter

**void**

(4) Return Value

GEN_SUCCESS
GEN_ERROR

(5) Message

**void**

(6) Description

EMV kernel Parameter Initialization

** If parameters files(CAPublicKeys.ini, EMVParam.ini) exist, the EMV kernel parameter is initialized by reading files
Parameter files default location : /etc/genmegadevice/emvparam
The parameter file location setting value is saved in the SDK configuration file.
SDK configuration file location : /etc/genmegadevice/genmegadevice.cfg

8.2.2 emvkrnl_set_term

(1) Prototype

**int emvkrnl_set_term ( int seq_cnt )**

(2) Input Parameter

**int seq_cnt**
Terminal Sequence Number ( 4 digits )

(3) Output Parameter

**void**

(4) Return Value

GEN_SUCCESS
GEN_ERROR

(5) Message

**void**

(6) Description

EMV Parameters Setting for Transaction Initialization

## 8.2.3 emvkrnl_application_selection

(1) Prototype

**int emvkrnl_application_selection ( int keyValue )**

(2) Input Parameter

**int keyValue**
0 : Application Selection Key (default)
0 < key : Application Number selected by customer

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR :
GEN_SUCCESS :
GEN_REQ_RETRY : Retry Application Select Step
GEN_REQ_AIDSELECT : Need User Input for Application Selection

(5) Message

**void**

(6) Description

Application(VISA/MASTER/AMEX...) Selection supported by IC card and Terminal

## 8.2.4 emvkrnl_get_candidateList

(1) Prototype

**int emvkrnl_get_candidateList ( char* rdata )**

(2) Input Parameter

**void**

(3) Output Parameter

**char* rdata**
Response Format
[Count=2]\n
[AID:X]US DEBIT\n
[AID:X]VISA DEBIT

(4) Return Value

GEN_SUCCESS :

(5) Message

**void**

(6) Description

Read Candidate List supported by IC card and Terminal

## emvkrnl_read_application

8.2.5 emvkrnl_read_application

(1) Prototype

**int emvkrnl_read_application ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR :
GEN_SUCCESS :

(5) Message

**void**

(6) Description

Read data from Selected Application

### 8.2.6 emvkrnl_processing_restrictions

(1) Prototype

**int emvkrnl_processing_restrictions ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR :
GEN_SUCCESS :

(5) Message

**void**

(6) Description

Verify Processing Restrictions (version, usage control, valid date, expiration date)

### 8.2.7 emvkrnl_offline_data_authentication

(1) Prototype

**int emvkrnl_offline_data_authentication ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR :
GEN_SUCCESS :

(5) Message

**void**

(6) Description

Authenticate IC card certificate with public key (This process may pass in case of ONLINE-ONLY Terminal)

## emvkrnl_cardholder_verification

### 8.2.8 emvkrnl_cardholder_verification

(1) Prototype

**int emvkrnl_cardholder_verification ( int nPinLen, const char* pinValue )**

(2) Input Parameter

**int nPinLen**
PIN string length
**const char* pinValue**
PIN Plain Text
Ex) “1234”

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR :
GEN_SUCCESS :
GEN_REQ_PASSWORD : When PASSWORD is required, request PIN Transfer
GEN_INVALID_PASSWORD : Password is not valid

(5) Message

**void**

(6) Description

Authenticate IC card certificate with public key (This process may pass in case of ONLINE-ONLY Terminal)

## emvkrnl_transaction_type_select

8.2.9 emvkrnl_transaction_type_select

(1) Prototype

**int emvkrnl_transaction_type_select ( int tranType )**

(2) Input Parameter

**int tranType**
1 : Cash
2 : Inquiry
3 : Transfer

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR :
GEN_SUCCESS :
GEN_INVALID_PARAM

(5) Message

**void**

(6) Description

Select Transaction Type

## 8.2.10 emvkrnl_account_type_select

(1) Prototype

**int emvkrnl_account_type_select ( int accType )**

(2) Input Parameter

**int accType**
1 : Default
2 : Saving
3 : Check/Debit
4 : Credit

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR :
GEN_SUCCESS :
GEN_INVALID_PARAM

(5) Message

**void**

(6) Description

Select Account Type

## emvkrnl_set_purchase_amount

8.2.11 emvkrnl_set_purchase_amount

(1) Prototype

**int emvkrnl_set_purchase_amount ( int nType, int nAmount )**

(2) Input Parameter

**int nType**
0 : Purchase Amount
1 : Cashback Amount
**int nAmount**
$1 -> 100
$100 -> 10000

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR :
GEN_SUCCESS :
GEN_INVALID_PARAM

(5) Message

**void**

(6) Description

Set purchase amount or cash back amount

### 8.2.12 emvkrnl_terminal_risk_management

(1) Prototype

**int emvkrnl_terminal_risk_management ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR :
GEN_SUCCESS :
GEN_INVALID_PARAM

(5) Message

**void**

(6) Description

Terminal Risk Management: Process that offline transaction switches to online to lower risk when predefined conditions are met

8.2.13 emvkrnl_terminal_action_analysis

(1) Prototype

**int emvkrnl_terminal_action_analysis ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Error / Decline
GEN_SUCCESS : Offline Approval
GEN_ARQC : Online Approval
GEN_ADVICE : Advice

(5) Message

**void**

(6) Description

Terminal Action Analysis: Based on previous transaction, determine
Online/Offline Approval, Reject

### 8.2.14 emvkrnl_online_process

(1) Prototype

**int emvkrnl_online_process ( GENEMV_HOST_DATA *host_data )**

(2) Input Parameter

**GENEMV_HOST_DATA *host_data**

```
typedef struct
{
    unsigned char        full_chip_data_option;      //Y/N
    unsigned char        ARC[3];                     //length(1)+data
    unsigned char        add_resp_data[32];          //length(1)+data
    unsigned char        IAD[32];                    //length(1)+data
    int                  isr_len;                    //Issuer Length
    unsigned char        issuer_script[256];         //length(1)+data
} GENEMV_HOST_DATA;
```

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Error / Decline
GEN_SUCCESS : Offline Approval
GEN_REJECT : Online Reject
GEN_REFERRAL : Online Referral

(5) Message

**void**

(6) Description

Online Received Message Process

### 8.2.15 emvkrnl_card_action_analysis

(1) Prototype

**int emvkrnl_card_action_analysis ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Fail
GEN_SUCCESS : Success
GEN_REVERSAL : Reversal
GEN_CONFIRM : Confirm
GEN_ADVISE : Advise

(5) Message

**void**

(6) Description

Online Approval Process/Card Action Analysis

### 8.2.16 emvkrnl_unable_online

(1) Prototype

**int emvkrnl_unable_online ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Fail
GEN_SUCCESS : Success
GEN_REVERSAL : Reversal
GEN_CONFIRM : Confirm
GEN_ADVISE : Advise

(5) Message

**void**

(6) Description

Unable Online Process

### 8.2.17 emvkrnl_online_referral

(1) Prototype

**int emvkrnl_online_referral ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Fail
GEN_REVERSAL : Reversal
GEN_CONFIRM : Confirm
GEN_ADVISE : Advise
GEN_RDECLINE : Referral Decline

(5) Message

**void**

(6) Description

Online REFERRAL Process

### 8.2.18 emvkrnl_online_reject

(1) Prototype

**int emvkrnl_online_reject ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Fail
GEN_SUCCESS : Success
GEN_REVERSAL : Reversal
GEN_ADVISE : Advise

(5) Message

**void**

(6) Description

Online Reject Process

## 8.2.19 emvkrnl_online_advice

(1) Prototype

**int emvkrnl_online_advice ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Fail
GEN_SUCCESS : Success
GEN_REVERSAL : Reversal
GEN_CONFIRM : Confirm

(5) Message

**void**

(6) Description

Online Advice Process

### 8.2.20 emvkrnl_online_confirm

(1) Prototype

**int emvkrnl_online_confirm ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Fail
GEN_SUCCESS : Success

(5) Message

**void**

(6) Description

Online CONFIRM Process

### 8.2.21 emvkrnl_online_reversal

(1) Prototype

**int emvkrnl_online_reversal ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Fail
GEN_SUCCESS : Success

(5) Message

**void**

(6) Description

Online REVERSAL Process

8.2.22 emvkrnl_completion

(1) Prototype

**int emvkrnl_completion ( void )**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

GEN_ERROR : Fail
GEN_SUCCESS : Success

(5) Message

**void**

(6) Description

EMV Transaction Completion Process

8.2.23 emvkrnl_read_dataEl

(1) Prototype

**int emvkrnl_read_dataEl ( const char* tagname, GENEMV_DATA_ELEMENT *data_el )**

(2) Input Parameter

**const char* tagname**
Data Elementary Tag name:
Ex] EMV Transaction TAG
DE_9F06 / DE_82 / DE_9F36 / DE_9F26 / DE_9F27 / DE_9F10 / DE_9F18 / DE_95 /
DE_9F37 / DE_5F34 / DE_9F1A / DE_5F2A / DE_9A / DE_9C / DE_9F41 / DE_9F1E /
DE_9F33 / DE_9F35 / DE_8E / DE_9F34 / DE_9F39 / DE_9B / DE_5F24

(3) Output Parameter

**GENEMV_DATA_ELEMENT *data_el**

```
typedef struct
{
    int                     rLen;
    unsigned char           rdata[128];
} GENEMV_DATA_ELEMENT;
```

(4) Return Value

GEN_ERROR : Fail
GEN_SUCCESS : Success

(5) Message

**void**

(6) Description

Read EMV Data Element Value

### 8.2.24 emvkrnl_set_config

(1) Prototype

**int emvkrnl_set_config(const char* tagName, const char *tagValue )**

(2) Input Parameter

**const char* tagName**
Configration name:
"ConfPath" - The writable path to save a configration and data files.

**const char *tagValue**
Example for 'ConfPath'
"/genmega/data/"

(3) Output Parameter

**void**

(4) Return Value
int
GEN_ERROR : Fail
GEN_SUCCESS : Success

(5) Message

**void**

(6) Description

Set configrations to manage EMV kernel.

## 8.2.25 emvKrnl_datacapture_clear

(1) Prototype

**int emvKrnl_datacapture_clear(void)**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

int
GEN_ERROR : Fail
GEN_SUCCESS : Success

(5) Message

**void**

(6) Description

Clear data captured.

## 8.2.26 emvKrnl_DataCapture

(1) Prototype

**int emvKrnl_DataCapture(byte type, byte cnt)**

(2) Input Parameter

**byte type**
0x00 : Approve
0x01 : Cancel
0x02 : Advice

**byte type**
The count to be canceld when the type is 'Cancel'.

(3) Output Parameter

**void**

(4) Return Value
int
GEN_UNABLE_CAPTURE : Unable to capture a data
GEN_SUCCESS : Success

(5) Message

**void**

(6) Description

Capture or cancel data for the offline transaction.

## 8.3 Terminal AID List

**Terminal AID List**

| | |
|---|---|
| 01. A0000006200620 | [COMMON DEBIT - DNA] |
| 02. A0000001524010 | [COMMON DEBIT - DISCOVER] |
| 03. A0000000980840 | [COMMON DEBIT - VISA] |
| 04. A0000000042203 | [COMMON DEBIT - MASTER] |
| 05. A000000002501 | [AMERICAN EXPRESS] |
| 06. A0000000651010 | [JAPAN CREDIT BUREAU] |
| 07. A0000001523010 | [DISCOVER] |
| 08. A0000000038010 | [PLUS] |
| 09. A0000000032010 | [VISA ELECTRON] |
| 10. A0000000031010 | [VISA CREDIT / DEBIT] |
| 11. A0000000046000 | [CIRRUS] |
| 12. A0000000043060 | [MAESTRO(DEBIT)] |
| 13. A0000000041010 | [MASTER CREDIT / DEBIT] |

8.4 EMV Kernel TAG Description

## EMV Kernel TAG Description

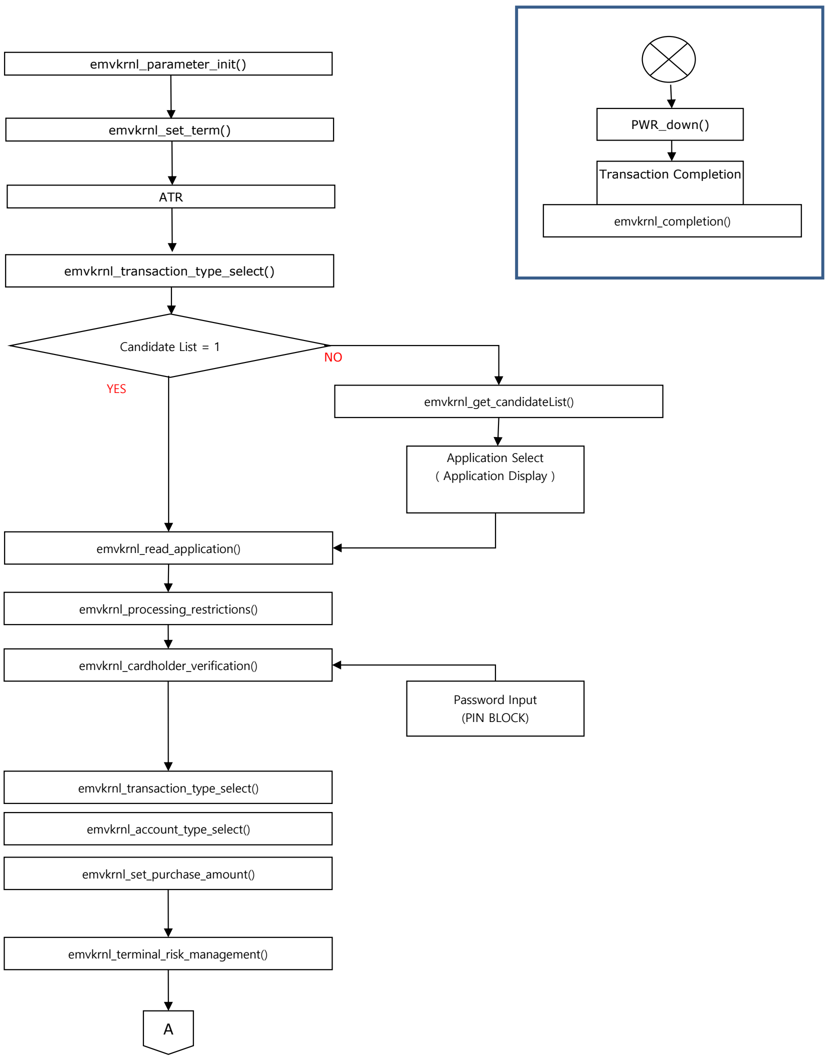
| TID | TAG | DESCRIPTION |
|---|---|---|
| 0 | 9F01 | Acquirer Identifier |
| 1 | 9F40 | Additional Terminal Capability |
| 2 | 8100 | Amount Authorized (Binary) |
| 3 | 9F02 | Amount Authorized (Numeric) |
| 4 | 9F04 | Amount Other (Binary) |
| 5 | 9F03 | Amount Other (Numeric) |
| 6 | 9F3A | Amount Reference Currency |
| 7 | DF01 | Amount Transaction |
| 8 | 9F26 | Application Cryptogram |
| 9 | 9F42 | Application Currency Code |
| 10 | 9F44 | Application Currency Exponent |
| 11 | 9F05 | Application DiscretionaryData |
| 12 | 5F25 | Application Effective Date |
| 13 | 5F24 | Application Expired Date |
| 14 | 9400 | Application File Locator |
| 15 | 4F00 | Application Identifier |
| 16 | 8200 | Application Interchange Profile |
| 17 | 5000 | Application Label |
| 18 | 9F12 | Application Preferred Name |
| 19 | 5A00 | Primary Account Number |
| 20 | 5F34 | rimary Account Sequence Number |
| 21 | 8700 | Application Priority Indicator |
| 22 | 9F3B | Application Reference Currency |
| 23 | 9F43 | Application Reference Currency Exponent |
| 24 | 9F36 | Application Transaction Counter |
| 25 | 9F07 | Application Usage Control |
| 26 | 9F08 | ICC Application Version Number |
| 27 | 9F09 | Terminal Application Version Number |
| 28 | 8A00 | Authorization Response Code |
| 29 | 8C00 | CDOL1 |
| 30 | 8D00 | CDOL2 |
| 31 | 5F20 | Cardholder Name |
| 32 | 9F0B | Cardholder Name |
| 33 | 8 | Cardholder Verification Method List |
| 34 | 9F34 | Cardholder Verification Method Result |
| 35 | 8F00 | CA Public Key Index |
| 36 | 9F27 | Cryptogram Information Data |
| 37 | 9F45 | Data Authorization Code |
| 38 | 8400 | DF Name |
| 39 | D600 | Default DDOL |
| 40 | D700 | Default TDOL |
| 41 | DF02 | Enciphered PIN Data |
| 42 | 9F49 | DDOL |
| 43 | BF0C | FCI Issuer Discretionary Data |
| 44 | 9F4C | ICC Dynamic Data |
| 45 | 9F2D | ICC PIN Public Key Certificate |
| 46 | 9F2E | ICC PIN Public Key Exponent |
| 47 | 9F2F | ICC PIN Public Key Remainder |
| 48 | 9F46 | ICC Public Key Certificate |
| 49 | 9F47 | ICC Public Key Exponent |
| 50 | 9F48 | ICC Public Key Remainder |
| 51 | 9F1E | IFD Serial Number |
| 52 | 9F0D | Issuer Action Code Default |
| 53 | 9F0E | Issuer Action Code Denial |
| 54 | 9F0F | Issuer Action Code Online |
| 55 | 9F10 | Issuer Application Date |
| 56 | 9100 | Issuer Authorization Data |
| 57 | 9F11 | Issuer Code Table Index |
| 58 | 5F28 | Issuer Country Code |
| 59 | 9000 | Issuer Public Key Certificate |
| 60 | 9F32 | Issuer Public Key Exponent |
| 61 | 9200 | Issuer Public Key Remainder |
| 62 | 9F18 | Issuer Script Identifier |
| 63 | DF03 | Issuer Script Result |
| 64 | 7100 | Issuer Script Template 1 |
| 65 | 7200 | Issuer Script Template 2 |

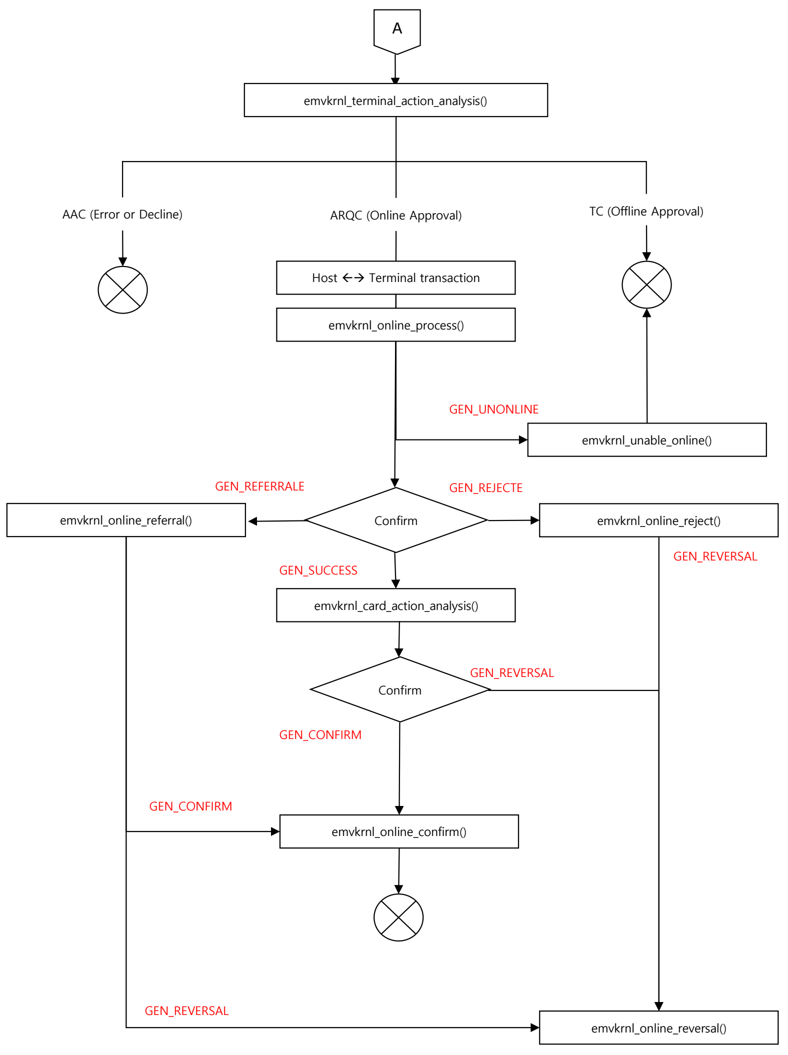
## EMV Kernel TAG Description

| TID | TAG | DESCRIPTION |
|---|---|---|
| 66 | 5F2D | Language Preference |
| 67 | 9F13 | Last Online Application Transaction Counter |
| 68 | 9F14 | Lower Offline Limit |
| 69 | 9F15 | Merchant Category Code |
| 70 | 9F16 | Merchant Identifier |
| 71 | DF04 | Merchant Name Location |
| 72 | DF05 | Message Type |
| 73 | 9F39 | POS Entry Code |
| 74 | 9F38 | PDOL |
| 75 | 5F30 | Service Code |
| 76 | 9F4B | Signed Dynamic Application Data |
| 77 | 9300 | Signed Static Application Data |
| 78 | 9F4A | Static Data Authentication Tag List |
| 79 | D800 | Terminal Action Code Default |
| 80 | D900 | Terminal Action Code Denial |
| 81 | DA00 | Terminal Action Code Online |
| 82 | 9F33 | Terminal Capability |
| 83 | 9F1A | Terminal Country Code |
| 84 | 9F1B | Terminal Floor Limit |
| 85 | 9F1C | Terminal Identifier |
| 86 | 9F1D | Terminal Risk Management Data |
| 87 | 9F35 | Terminal Type |
| 88 | 9500 | Terminal Verification Result |
| 89 | 9F1F | Track1 Discretionary Data |
| 90 | 9F20 | Track2 Discretionary Data |
| 91 | 5700 | Track2 Equivalent Data |
| 92 | 9700 | TDOL |
| 93 | 9800 | Transaction Certificate Hash Value |
| 94 | 5F2A | Transaction Currency Code |
| 95 | 5F36 | Transaction Currency Exponent |
| 96 | 9A00 | Transaction Date |
| 97 | 9F3C | Transaction Reference Currency Code |
| 98 | DB00 | Transaction Reference Currency Conversion |
| 99 | 9F3D | Transaction Reference Currency Exponent |
| 100 | 9F41 | Transaction Sequence Counter |
| 101 | 9B00 | Transaction Status Information |
| 102 | 9F21 | Transaction Time |
| 103 | 9C00 | Transaction Type |
| 104 | 9F37 | Unpredictable Number |
| 105 | 9F23 | Upper Offline Limit |
| 106 | D100 | Installment Number |
| 107 | D400 | Approval Number |
| 108 | DF06 | Issuer Public Key |
| 109 | DF07 | Static Application Data |
| 110 | D200 | Tax Amount |
| 111 | D300 | Service Fee |
| 112 | D500 | Original Transaction Date |
| 113 | DC00 | Merchant No |
| 114 | DE00 | Original Approval No |
| 115 | DF08 | Acquirer Name |
| 116 | DF09 | Host Message |
| 117 | DF0A | Terminal Counter |
| 118 | DF0B | Recovered ICC Public Key |
| 119 | DF0C | Dynamic Application Data |
| 120 | 9F06 | Application Identifier (AID) - terminal |
| 121 | 9F22 | Certification Authority Public Key Index |
| 122 | DF0D | Target Percentage |
| 123 | DF0E | Threshold Value |
| 124 | DF0F | Maximum Target Percentage |
| 125 | 9F53 | Transaction Category Code |
| 126 | DF1D | Host Notice |
| 127 | DF1E | Display Control |
| 128 | 9F4D | Log Entry |
| 129 | 5F57 | Account Type |

8.5 EMV TRANSACTION FLOW

## EMV TRANSACTION FLOW


emvkrnl_parameter_init()
emvkrnl_set_term()
ATR
emvkrnl_transaction_type_select()
Candidate List = 1
NO
YES
emvkrnl_get_candidateList()
Application Select
( Application Display )
emvkrnl_read_application()
emvkrnl_processing_restrictions()
emvkrnl_cardholder_verification()
Password Input
(PIN BLOCK)
emvkrnl_transaction_type_select()
emvkrnl_account_type_select()
emvkrnl_set_purchase_amount()
emvkrnl_terminal_risk_management()
A
PWR_down()
Transaction Completion
emvkrnl_completion()


A
emvkrnl_terminal_action_analysis()
AAC (Error or Decline)
ARQC (Online Approval)
TC (Offline Approval)
Host ←→ Terminal transaction
emvkrnl_online_process()
GEN_UNONLINE
emvkrnl_unable_online()
GEN_REFERRALE
GEN_REJECTE
emvkrnl_online_referral()
Confirm
emvkrnl_online_reject()
GEN_REVERSAL
GEN_SUCCESS
emvkrnl_card_action_analysis()
GEN_REVERSAL
Confirm
GEN_CONFIRM
GEN_CONFIRM
emvkrnl_online_confirm()
GEN_REVERSAL
emvkrnl_online_reversal()

## 9. BAU

(1) It describes following interfaces in order to control Bill Acceptor Unit(BAU/BA2).

| | Function | Description |
|---|---|---|
| 1 | BAU_Open | Open Serial Port |
| | BA2_Open | |
| 2 | BAU_Close | Close Serial Port |
| | BA2_Close | |
| 3 | BAU_Reset | Reset BAU |
| | BA2_Reset | |
| 4 | BAU_Status | Get the Status of BAU |
| | BA2_Status | |
| 5 | BAU_SetCapabilities | Set Capabilities of BAU |
| | BA2_SetCapabilities | |
| 6 | BAU_GetCapabilities | Get Capabilities of BAU |
| | BA2_GetCapabilities | |
| 7 | BAU_AcceptBill | Accept the notes (For multi-currency use AcceptBillEx Function) |
| | BA2_AcceptBill | |
| 8 | BAU_AcceptBillEx | Extended command to accept the notes |
| | BA2_AcceptBillEx | |
| 9 | BAU_Cancel | Cancel accept bill (For multi-currency use CancelEx Function) |
| | BA2_Cancel | |
| 10 | BAU_CancelEx | Extended command to cancel accept bill |
| | BA2_CancelEx | |
| 11 | BAU_StackBill | Stack the note at escrow into the cash box |
| | BA2_StackBill | |
| 12 | BAU_ReturnBill | Return the note at escrow to the customer |
| | BA2_ReturnBill | |
| 13 | BAU_SetEnableDenom | Set the string to enable denomination by currency. |
| | BA2_SetEnableDenom | |
| 14 | BAU_GetEnableDenom | Get the string to enable denomination by currency. |
| | BA2_GetEnableDenom | |
| 15 | BAU_GetSupportCurrency | Get the string of the denomination list by currency |
| | BA2_GetSupportCurrency | |
| 16 | BAU_GetAcceptorIDs | Gets the information of the BillAcceptor device |
| | BA2_GetAcceptorIDs | |
| 17 | BAU_GetLastError | Get the final Error Code of BAU |
| | BA2_GetLastError | |
| | | |
| | | |

*** BAU - Bill Acceptor Unit1, BA2 - Bill Acceptor Unit2

9.1 BAU_Open / BA2_Open

## BAU_Open/BA2_Open

(1) Prototype

**int BAU_Open(IN const char* szPortName, OUT unsigned char szVerInfo[10])**
**int BA2_Open(IN const char* szPortName, OUT unsigned char szVerInfo[10])**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to BAU ( Ex) "/dev/ttyS2" )

(3) Output Parameter

**unsigned char** szVerInfo[10]
Array Pointer to obtain the F/W version of BAU

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open the Serial Port of BAU
Obtain the Firmware Version of BAU
Set the default capabilities of BAU

| | |
|---|---|
| - Denomination | : All Enable |
| - Orientation Control | : 4-Way ( Accept bills fed any way ) |
| - Escrow Mode | : Enable |

9.2 BAU_Close / BA2_Close

## BAU_Close / BA2_Close

1. Introduction

**void BAU_Close()**
**void BA2_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close the Serial Port of BAU
End the thread of BAU

9.3 BAU_Reset / BA2_Reset

## BAU_Reset / BA2_Reset

(1) Prototype

**int BAU_Reset()**
**int BA2_Reset()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description
Initialize the Bill Acceptor(BAU) without stacking

## 9.4 BAU_Status / BA2_Status

**BAU_Status / BA2_Status**

(1) Prototype

**void BAU_Status(OUT BAU_STATUS *BauStatus)**
**void BA2_Status(OUT BAU_STATUS *BauStatus)**

(2) Input Parameter

**void**

(3) Output Parameter

**BAU_STATUS** *BauStatus
Pointer of BAU_STATUS Structure Buffer obtaining BAU Status information

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Obtain Status information of BAU

```
typedef struct tag_BAU_STATUS{

      unsigned char   bLineStatus          : TF : Displays the connection status with BAU Device
                                                  HM_DEV_CONNECT / HM_DEV_DISCONNECT
      unsigned char   bIdling;                  : The bill acceptor is idling between bill transactions.
      unsigned char   bAccepting;               : The bill acceptor is drawing in a bill.
      unsigned char   bEscrow;                  : There is a valid bill in escrow.
      unsigned char   bStacking;                : The bill acceptor is stacking a bill.
      unsigned char   bReturning;               : The bill acceptor is returning a bill to the customer.
      unsigned char   bJammed;                  : The bill path is blocked and the bill acceptor has been unable to resolve the issue.
      unsigned char   bStackerFull;             : The cash box is full of bank notes and no more may be accepted
      unsigned char   bCassetteAttached;        : The cash box has been removed. No bills may be accepted. #1
      unsigned char   bPaused;                  : The customer is attempting to feed another note while the previous note is still being processed.
      unsigned char   bCalibration;             : The unit is in calibration mode. (not used)
      unsigned char   bFailure;                 : The bill acceptor has encountered a problem and is out of service
      unsigned char   bPushNoPush;              : Set according to device's type. (not used)
      unsigned char   bFlashDownload;           : A flash download is ready to commence
                                                  TRUE / FALSE   ( SET - TRUE )
                                                  *** #1 : SET - TRUE ( The Status of Removed Cassette )
}BAU_STATUS;
```

9.5 BAU_SetCapabilities / BA2_SetCapabilities

## BAU_SetCapabilities / BA2_SetCapabilities

(1) Prototype

**int BAU_SetCapabilities(IN unsigned char bDenomination, IN int iOrientation, IN unsigned char bEscrowEnable)**
**int BA2_SetCapabilities(IN unsigned char bDenomination, IN int iOrientation, IN unsigned char bEscrowEnable)**

(2) Input Parameter

**unsigned char** bDenomination
Specify Denomination enable
ex) BAU_NOTE1 | BAU_NOTE2 | BAU_NOTE3 | BAU_NOTE4 | BAU_NOTE5 | BAU_NOTE6 | BAU_NOTE7

**int** iOrientation
This field controls the acceptance of bank notes based on the orientation of those notes as they enter the bill acceptor
ex) BAU_ONEWAY or BAU_TWOWAY or BAU_FOURWAY

**unsigned char** bEscrowEnable
This mode determines how bills are handled after the bills have been validated
ex) BAU_OKESCROW (Enable) / BAU_NOESCROW (Disable)

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Set the capabilities of BAU
If you want to change the capabilities, you have to set the capabilities after BAU_OPEN Function.

9.6 BAU_GetCapabilities / BA2_GetCapabilities

## BAU_GetCapabilities / BA2_GetCapabilities

(1) Prototype

**void BAU_GetCapabilities(OUT unsigned char *pDenom, OUT int *pOrientation, OUT unsigned char *pEscrowEnable)**
**void BA2_GetCapabilities(OUT unsigned char *pDenom, OUT int *pOrientation, OUT unsigned char *pEscrowEnable)**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** *pDenomination
Specify Denomination enable
ex) BAU_NOTE1 | BAU_NOTE2 | BAU_NOTE3 | BAU_NOTE4 | BAU_NOTE5 | BAU_NOTE6 | BAU_NOTE7

**int** *pOrientation
This field controls the acceptance of bank notes based on the orientation of those notes as they enter the bill acceptor
ex) BAU_ONEWAY or BAU_TWOWAY or BAU_FOURWAY

**unsigned char** *pEscrowEnable
This mode determines how bills are handled after the bills have been validated
ex) BAU_OKESCROW (Enable) / BAU_NOESCROW (Disable)

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Get the capabilities of BAU
you have to get the capabilities after BAU_OPEN Function.

## 9.7 BAU_AcceptBill / BA2_AcceptBill

**BAU_AcceptBill / BA2_AcceptBill**

(1) Prototype

**int BAU_AcceptBill(IN char bMode, OUT int *iBillDenom)**
**int BA2_AcceptBill(IN char bMode, OUT int *iBillDenom)**

(2) Input Parameter

**int** bMode : SENDONLY / RECVONLY
Communication mode with BAU

(3) Output Parameter

**int** *iBillDenom :
Denomination validated in BAU

(4) Return Value

HM_DEV_OK
HM_DEV_REJECTED_BILL
HM_DEV_NOTSUPPORT
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_DOING

(5) Message

**void**

(6) Description

Accepts a bill and validates. If the result of validate is OK, the bill should be escrowed or returned to customer.
If you want to cancel to accept bill, you have to execute BAU_Cancel command as below.
**For BillAcceptor using multi-currency, HM_DEV_NOTSUPPORT is returned. (Use the BAU_AcceptBillEx() function)**
**The BAU_AcceptBill() function does not know the currency.**

ex) Sample code
```
time_t StartTime, CurTime;
int iBillDenom = 0;

iRet = BAU_AcceptBill(SENDONLY, &iBillResult);
time(&StartTime);
while(1){
    time(&CurTime);
    if((StartTime+30) < CurTime) {          // and can add routine When the customer select cancel to accept a bill
        iRet = BAU_Cancel(&iBillResult);
        break;
    }
    iRet = BAU_AcceptBill(RECVONLY, &iBillResult);
    if( iRet != HM_DEV_DOING) break;
    usleep(300*1000);
}
if(( iRet != HM_DEV_REJECTED_BILL) &&( iRet != HM_DEV_OK)) {
    // Error Process
} else if(iRet == HM_DEV_OK && iBillResult != 0) {
    iBillDenom = 0;
    switch(iBillResult) {
        case 1: iBillDenom = 1; break;
        case 2: iBillDenom = 2; break;
        case 3: iBillDenom = 5; break;
        case 4: iBillDenom = 10; break;
        case 5: iBillDenom = 20; break;
        case 6: iBillDenom = 50; break;
        case 7: iBillDenom = 100; break;
        default : break;
    }
    printf("\n [RESULT]:[%02X] - %d Dollar \n", iBillResult, iBillDenom);
}
```

9.8 BAU_AcceptBillEx / BA2_AcceptBillEx

## BAU_AcceptBillEx / BA2_AcceptBillEx

(1) Prototype

**int BAU_AcceptBillEx(IN char bMode, OUT EXPVALUE_INFO *pExpValue)**
**int BA2_AcceptBillEx(IN char bMode, OUT EXPVALUE_INFO *pExpValue)**

(2) Input Parameter

**int** bMode : SENDONLY / RECVONLY
Communication mode with BAU

(3) Output Parameter

**EXPVALUE_INFO *** pExpValue :
Denomination value struct in BAU

(4) Return Value

HM_DEV_OK
HM_DEV_REJECTED_BILL
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_DOING

(5) Message

**void**

(6) Description

Accepts a bill and validates. If the result of validate is OK, the bill should be escrowed or returned to customer.
If you want to cancel to accept bill, you have to execute BAU_CancelEx command as below.

```
EXPVALUE_INFO Structure
typedef struct tag_EXPVALUE_INFO {
    char szISOCode[5];          // ISO Code of Currency  ex) USD, CAD
    int nDenom;                 // Denomination
    int nOrientation;           // Orientation of acceptance of bills.
    char szTSCV[5];             // szTSCV[0]:Type, szTSCV[1]:Series, szTSCV[2]:Compatibility, szTSCV[3]:Version
}EXPVALUE_INFO;
```

szTSCV[0]:Type => An ASCII letter that documents the note type
szTSCV[1]:Series => An ASCII letter that documents the note series
szTSCV[2]:Compatibility => An ASCII letter that documents the revision of the recognition core used.
szTSCV[3]:Version => An ASCII letter that documents the version of the note's recognition criteria

ex) Sample code

```
        time_t StartTime, CurTime;
        EXPVALUE_INFO stExpValue = {0};

        iRet = BAU_AcceptBillEx(SENDONLY, &stExpValue);
        time(&StartTime);
        while(1){
                time(&CurTime);
                if((StartTime+30) < CurTime){          // and can add routine When the customer select cancel to accept a bill
                    iRet = BAU_CancelEx(&stExpValue);
                    break;
                }
                iRet = BAU_AcceptBillEx(RECVONLY, &stExpValue);
                if( iRet != HM_DEV_DOING) break;
                usleep(300*1000);
        }
        if(( iRet != HM_DEV_REJECTED_BILL) &&( iRet != HM_DEV_OK)) {
            // Error Process
        } else if(iRet == HM_DEV_OK && stExpValue.nDenom != 0) {
            printf("\n [RESULT] %s - %d (Orientation:%d)\n", stExpValue.szISOCode, stExpValue.nDenom, stExpValue.nOrientation);
        }
```

9.9 BAU_Cancel / BA2_Cancel

## BAU_Cancel / BA2_Cancel

(1) Prototype

**int BAU_Cancel(OUT int *iBillDenom)**
**int BA2_Cancel(OUT int *iBillDenom)**

(2) Input Parameter

**void**

(3) Output Parameter

**int** *iBillDenom :
Denomination validated in BAU

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOTSUPPORT
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**Void**

(6) Description

Cancel to accept a bill.
If a bill is escrowed at the same time with executing cancel command, output the result of denomination of bill.

**For BillAcceptor using multi-currency, HM_DEV_NOTSUPPORT is returned. (Use the BAU_CancelEx() function)**
**The BAU_Cancel() function does not know the currency.**

9.10 BAU_CancelEx / BA2_CancelEx

## BAU_CancelEx / BA2_CancelEx

(1) Prototype

**int BAU_CancelEx(OUT EXPVALUE_INFO *pExpValue)**
**int BA2_CancelEx(OUT EXPVALUE_INFO *pExpValue)**

(2) Input Parameter

**void**

(3) Output Parameter

**EXPVALUE_INFO ***pExpValue :
Denomination value struct in BAU

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**Void**

(6) Description

Cancel to accept a bill.
If a bill is escrowed at the same time with executing cancel command, output the result of denomination of bill.

EXPVALUE_INFO Structure

```
typedef struct tag_EXPVALUE_INFO {
    char szISOCode[5];              // ISO Code of Currency  ex) USD, CAD
    int nDenom;                     // Denomination
    int nOrientation;               // Orientation of acceptance of bills.
    char szTSCV[5];                 // szTSCV[0]:Type, szTSCV[1]:Series, szTSCV[2]:Compatibility, szTSCV[3]:Version
}EXPVALUE_INFO;
```

szTSCV[0]:Type => An ASCII letter that documents the note type
szTSCV[1]:Series => An ASCII letter that documents the note series
szTSCV[2]:Compatibility => An ASCII letter that documents the revision of the recognition core used.
szTSCV[3]:Version => An ASCII letter that documents the version of the note's recognition criteria

9.11 BAU_StackBill / BA2_StackBill

## BAU_StackBill / BA2_StackBill

(1) Prototype

**int BAU_StackBill(void)**
**int BA2_StackBill(void)**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**Void**

(6) Description

If a bill is in escrow, stack it in the cash box. Note that this command is only valid if Escrow mode is enabled and a bill is in escrow.
This command and the Bill Return command are mutually exclusive.

## 9.12 BAU_ReturnBill / BA2_ReturnBill

**BAU_ReturnBill / BA2_ReturnBill**

(1) Prototype

**int BAU_ReturnBill(void)**
**int BA2_ReturnBill(void)**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**Void**

(6) Description

If a bill is in escrow, return it to the consumer. Note that this command is only valid if Escrow mode is enabled and a bill is in escrow.
This command and the Bill Return command are mutually exclusive.

## 9.13 BAU_SetEnableDenom / BA2_SetEnableDenom

**BAU_SetEnableDenom / BA2_SetEnableDenom**

(1) Prototype

**int BAU_SetEnableDenom(IN char szEnableDenom[512])**
**int BA2_SetEnableDenom(IN char szEnableDenom[512])**

(2) Input Parameter

**char** szEnableDenom[512] :
String of enable denomination list by currency

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**Void**

(6) Description

Set the string to enable denomination by currency.
If you want to change the enable denomination, you have to set the enable denomination after BAU_OPEN Function.

ex)
If only USD is supported : USD,1,2,5,10,20,50,100
All Enable : **USD,1111111**
$1, $2 and $5 is Disalbe : **USD,0001111**
If USD and CAD are supported : USD,1,2,5,10,20,50,100;CAD,5,10,20,50,100
All Enable : **USD,1111111;CAD,11111**
$1, $2 and $5 of USD is Disalbe and $20 and $50 of CAD is disable : **USD,0001111;CAD,11001**

9.14 BAU_GetEnableDenom / BA2_GetEnableDenom

## BAU_GetEnableDenom / BA2_GetEnableDenom

(1) Prototype

**void BAU_GetEnableDenom(OUT char szEnableDenom[512])**
**void BA2_GetEnableDenom(OUT char szEnableDenom[512])**

(2) Input Parameter

**void**

(3) Output Parameter

**char** szEnableDenom[512] :
String of enable denomination list by currency

(4) Return Value

**void**

(5) Message

**Void**

(6) Description

Get the string to enable denomination by currency.

ex)
If only USD is supported : USD,1,2,5,10,20,50,100
All Enable : **USD,1111111**
$1, $2 and $5 is Disable : **USD,0001111**
If USD and CAD are supported : USD,1,2,5,10,20,50,100;CAD,5,10,20,50,100
All Enable : **USD,1111111;CAD,11111**
$1, $2 and $5 of USD is Disable and $20 and $50 of CAD is disable : **USD,0001111;CAD,11001**

9.15 BAU_GetSupportCurrency / BA2_GetSupportCurrency

## BAU_GetSupportCurrency / BA2_GetSupportCurrency

(1) Prototype

**int BAU_Cancel(OUT char szDenomData[512])**
**int BA2_Cancel(OUT char szDenomData[512])**

(2) Input Parameter

**void**

(3) Output Parameter

char szDenomData[512] :
String of Denomination List by Currency

(4) Return Value

Count of currencies supported

(5) Message

**void**

(6) Description

Get the string of the denomination list by currency.

ex)
If only USD is supported : **USD,1,2,5,10,20,50,100**
If USD and CAD are supported : **USD,1,2,5,10,20,50,100;CAD,5,10,20,50,100**

## 9.16 BAU_GetAcceptorIDs / BA2_GetAcceptorIDs

**BAU_GetAcceptorIDs / BA2_GetAcceptorIDs**

(1) Prototype

**void BAU_GetAcceptorIDs(OUT BAU_IDS *pBauIDs)**
**void BA2_GetAcceptorIDs(OUT BAU_IDS *pBauIDs)**

(2) Input Parameter

**void**

(3) Output Parameter

**BAU_IDS ***pBauIDs :
Structure contain BillAcceptor information.

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Gets the information of the BillAcceptor device.

BAU_IDS Structure

```
typedef struct tag_BAU_IDs {
    char szAcceptorType[21];          // ex) SCL6627R
    char szSerialNumber[21];          // ex) 42194352199
    char szAppPartNumber[10];         // ex) 28276131
    char szApplicationID[10];         // ex) 28276131
    char szVariantPartNumber[10];     // ex) 49197422
    char szVariantID[10];             // ex) 49197422
}BAU_IDS;
```

## 9.17 BAU_GetLastError / BA2_GetLastError

(1) Prototype

**void BAU_GetLastError(OUT unsigned char szErrorCode[5])**
**void BA2_GetLastError(OUT unsigned char szErrorCode[5])**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** szErrorCode[5]
Array Pointer which obtains final ErrorCode of BAU

(4) Return Value

**void**

(5) Message

**void**

(6) Description

it obtains final ErrorCode of BAU Device

| Error Code | | Description |
|---|---|---|
| 1 | 0013 | The bill not recognized is rejected to the customer |
| 2 | 0014 | The bill is not detected in stack or return command |
| 3 | 0015 | The bill acceptor detect a bill at escrow in accept command |
| 4 | 0016 | The bill acceptor received an invalid command |
| 5 | 0019 | The cash box is full |
| 6 | 0020 | Cassette is not attached |
| 7 | 0030 | The bill path is blocked in accept command |
| 8 | 0031 | The bill path is blocked in stack command |
| 9 | 0032 | The bill path is blocked in return command |
| 10 | 0040 | Command execution failure |
| 11 | 0041 | BillAcceptor detects cheated status |
| 12 | DN01 | Can't open port |
| 13 | DN02 | No response from device after send command |
| | | |

10. BCS

(1) It describes following interfaces to control Barcode Scanner

| | Function | Description |
|---|---|---|
| 1 | BCS_Open | Open Serial Port |
| 2 | BCS_OpenEx | Open Serial Port and Version |
| 3 | BCS_Close | Close Serial Port |
| 4 | BCS_Reset | Initialize the Device |
| 5 | BCS_AcceptScanCode | Start to wait for a scancode to be accepted in the Barcode Scanner |
| 6 | BCS_CancelScanCode | Cancel to wait a scancode accepting |
| 7 | BCS_GetInfor | Get information of device |
| 8 | BCS_SetCommand | Send the command to the device |
| 9 | BCS_GetLastError | Get the final Error Code |

| | CallBack Function | Description |
|---|---|---|
| 1 | BCS_CallBackRegister | Send a Message to the registered Function whenever barcode data is scanned |

## 10.1 BCS_Open

**BCS_Open**

(1) Prototype

**int BCS_Open(IN const char* szPortName, IN char bMobilePhoneMode)**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to HOP ( Ex) "/dev/ttyS7" )
**char** bMobilePhoneMode
When this mode is selected, your scanner is optimized to read bar codes from mobile phone or other LED displays
ex) BCS_MOBILEPHONE_ENABLE / BCS_MOBILEPHONE_DISABLE

(3) Output Parameter

**void**

4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open the Serial Port of BCS
Start the thread for BCS

## 10.2 BCS_OpenEx

**BCS_OpenEx**

(1) Prototype

**int BCS_OpenEx(IN const char* szPortName, IN char bMobilePhoneMode, OUT unsigned char szVerInfo[35])**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to MCR ( Ex) "/dev/ttyS7" )
**char** bMobilePhoneMode
When this mode is selected, your scanner is optimized to read bar codes from mobile phone or other LED displays
ex) BCS_MOBILEPHONE_ENABLE / BCS_MOBILEPHONE_DISABLE

(3) Output Parameter

**unsigned char** szVerInfo[35]
Array Pointer to obtain the version of BCS

4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open the Serial Port of BCS
Start the thread for BCS

10.3 BCS_Close

**BCS_Close**

1. Introduction

**void BCS_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close Serial Port of BCS
End the  thread of BCS

10.4 BCS_Reset

**BCS_Reset**

(1) Prototype

**int BCS_Reset()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Initialize the device

## 10.5 BCS_AcceptScanCode

**BCS_AcceptScanCode**

1. Introduction

**int BCS_AcceptScanCode(IN char bPresentationMode)**

(2) Input Parameter

**char** bPresentationMode
Use Presentation Mode if application need to get barcode data continuously.
ex) BCS_PRESENTATION_ENABLE / BCS_PRESENTATION_DISABLE

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

Callback Fuction will be called whenever barcode data is scanned from device untill application call BCS_CancelScanCode function.

| | | |
|---|---|---|
| int iId | : | HM_DEV_BCS_MSG |
| int iKind | : | BCS_DATA_SCANED |
| BCSScanData *szScanData | : | Array Pointer to get Scaned Data |

(6) Description

Make the barcode scanner wait for scancode.
whenever barcode is scanned, callback function is called with scanned data.
Presentation Mode uses ambient light to detect bar codes. The LEDs remain off until a bar code is presented to the scanner, then the LEDs turn on automatically to read the code.
If you choose to be disable the presentation mode, The barcode scanner will scan only once, then turn off the Led.

10.6 BCS_CancelScanCode

## BCS_CancelScanCode

1. Introduction

**int BCS_CancelScanCode()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Cancel to wait a scancode accepted

10.7 BCS_GetInfor

**BCS_GetInfor**

1. Introduction

**int BCS_GetInfor(OUT unsigned char *szDeviceInfor)**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** *szDeviceInfor
Array Pointer to obtain the information of BCS

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Get the information from barcode scanner as the current software revision, unit serial number, and other product information for both the scanner and base.

10.8 BCS_SetCommand

## BCS_SetCommand

1. Introduction

**BCS_SetCommand(IN unsigned char *szCommand)**

(2) Input Parameter

**unsigned char** *szCommand
Array Pointer of command to send to the device
ex) "REVINF." / "PAPHHF."/etc

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Send the serial programming command via a RS232C

## 10.9 BCS_GetLastError

**BCS_GetLastError**

(1) Prototype

**void BCS_GetLastError(OUT unsigned char szErrorCode[5])**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** szErrorCode[5]
Array Pointer which obtains final ErrorCode

(4) Return Value

**void**

(5) Message

**void**

(6) Description

it obtains final  ErrorCode of Device
This Device is Only Communication Error (No Hardware Error)

10.10 BCS_CallBackRegister

**BCS_CallBackRegister**

(1) Prototype

**void BCS_CallBackRegister(callback_key handler)**

(2) Input Parameter

**callback_key** handler
Callbacked Function

(3) Output Parameter

Callback Fuction will be called whenever barcode data is scanned from device untill application called BCS_CancelScanCode function.

typedef void (*callback_key)(int iId, int iKind, BCSScanData *szScanData);

```
int iId                :   HM_DEV_BCS_MSG
int iKind              :   BCS_DATA_SCANED
BCSScanData *szScanData    :   Array Pointer to get Scaned Data

                               typedef struct tag_BCSScanData
                               {
                                   unsigned char szCode[8192];   : scanned barcode data
                                   short         wSize;          : length of data
                               }BCSScanData;
```

(4) Example Code

```
void BCS_ScanedBarcodeData(int iId, int iKind, BCSScanData *BcsScanData) {
      // Input Code in here to process something
      ---
}

main() {
 ---
      // Register
    BCS_CallBackRegister(BCS_ScanedBarcodeData);
---
    while() {
      ---
    }
---
}
```

## 11. CIS

(1) It describes following interfaces to control Card Image Scanner

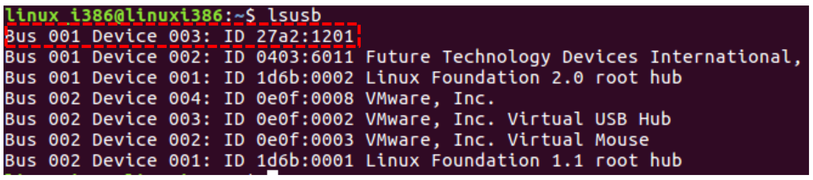
| | Function | Description |
|---|---|---|
| 1 | CIS Environment Setting | The environment setting for using CIS communication. |
| 2 | CRS_Open | Open Serial Port / USB Port |
| 3 | CRS_Close | Close Serial Port / USB Port |
| 4 | CRS_Reset | Initialize the Device |
| 5 | CRS_Status | Get CIS's Status |
| 6 | CRS_Entry | Entry Card |
| 7 | CRS_MSRead | Get MS Data of Card |
| 8 | CRS_ICReset | Connect IC Chip (IC power ON) |
| 9 | CRS_ICDeactivation | Disconnect IC Chip (IC power OFF) |
| 10 | CRS_ICDirect | Communicate with IC Chip |
| 11 | CRS_Eject | Eject Card |
| 12 | CRS_Retract | Retract Card |
| 13 | CRS_CISScan | Scan Card Image. (Front / Rear) |
| 14 | CRS_GetLastError | Get final H/W Error Code of CIS |
| | | |

11.1 CIS Environment Setting

## CIS Environment Setting

(1) Check USB Port Recognition

First check with the lsusb command to see if the card reader is connected (vid 0x27a2, pid 0x1201)



```
linux_i386@linuxi386:~$ lsusb
Bus 001 Device 003: ID 27a2:1201
Bus 001 Device 002: ID 0403:6011 Future Technology Devices International,
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 004: ID 0e0f:0008 VMware, Inc.
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

(2) install g++

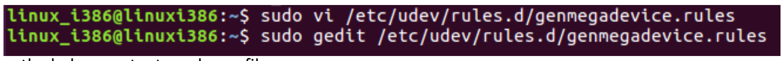
You need to install g++ to compile the test program.

For details, see 1.2 Set environment in TP_Guide_for_Linux.pdf.

(3) Enable CIS USB device user account.

1) Create a rules file in /etc/udev/rules.d.  ex) sudo touch /etc/udev/rules.d/genmegadevice.rules

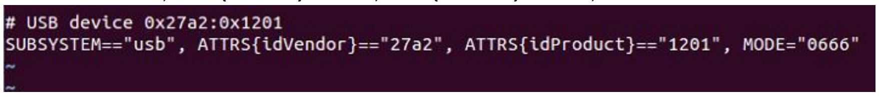
ex)



```
linux_i386@linuxi386:~$ sudo vi /etc/udev/rules.d/genmegadevice.rules
linux_i386@linuxi386:~$ sudo gedit /etc/udev/rules.d/genmegadevice.rules
```

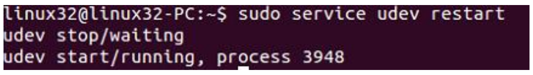
Enter the below contents and save file.

SUBSYSTEM=="usb", ATTRS{idVendor}=="27a2", ATTRS{idProduct}=="1201", MODE="0666"



```
# USB device 0x27a2:0x1201
SUBSYSTEM=="usb", ATTRS{idVendor}=="27a2", ATTRS{idProduct}=="1201", MODE="0666"
~
~
```

2) Restart Service -> sudo service udev restart



```
linux32@linux32-PC:~$ sudo service udev restart
udev stop/waiting
udev start/running, process 3948
```

3) Disconnect the USB cable of the connected CIS and reconnect it.

11.2 CRS_Open

**CRS_Open**

(1) Prototype

**int CRS_Open(IN const char* szPortName, OUT unsigned char *szVerInfo)**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to CIS ( Ex) "/dev/ttyS2" )

(3) Output Parameter

**unsigned char** *szVerInfo
Array Pointer to obtain the F/W version of CIS

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_ALREADY_OPEN
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open Serial Port and USB Port of CIS
Get Firmware Version of CIS.

## 11.3 CRS_Close

**CRS_Close**

1. Introduction

**void CRS_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close Serial Port and USB Port of CIS

## 11.4 CRS_Reset

(1) Prototype

**int CRS_Reset()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_BUSY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Check the status of CIS Device, and
If card is inside CIS, card is ejected.

11.5 CRS_Status

**CRS_Status**

(1) Prototype

**int CRS_Status(OUT CRS_STATUS *sts)**

(2) Input Parameter

**void**

(3) Output Parameter

**CRS_STATUS** *sts
CRS_STATUS Structure Buffer's Pointer to get the status information of CIS

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Get the status information of CIS

```
typedef struct tag_CRS_STATUS
{
        unsigned char bLineStatus;    :   DEV_CONNECT / DEV_DISCONNECT
                                          Displays the connection status with CIS Device
        unsigned char bStatus;        :   CARD UNKOWN / PRESENT / NOPRESENT / LATCHED / ENTRIED / MSREAD / SCANED / INSIDE
                                          Displays the input status of Card
        unsigned char  iMsStatus;     :   Not Used

} CRS_STATUS
```

## 11.6 CRS_Entry

**CRS_Entry**

(1) Prototype

**int CRS_Entry()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

If the card is detected by the front sensor of the CIS, card is inserted.
This command must be used in conjunction with the CRS_Status() command.
This command should be executed when bStatus of CRS_STATUS is CARD_PRESENT.

ex)

```
int sec = 10;   // Max Wait Time
int iRet;
CRS_STATUS CrsStatus;
unsigned int iStartTick = GetTickCount();
unsigned int iCurTick = iStartTick;

while (1) {
    sleep(10);
    iRet = CRS_Status(&CrsStatus);
    if(iRet == HM_DEV_OK ) {
        if (CrsStatus.bStatus == CARD_PRESENT) {    //Detected card in front.
            iRet = CRS_Entry();
            if(iRet != HM_DEV_OK ) {
                printf("\n card entry fail.\n");
                return -1;
            }
        }
        else if(CrsStatus.bStatus >= CARD_ENTRIED) {   //The card is inside the device.
            printf("\n Chip card has been entried.\n");
            break;
        }
        iCurTick = GetTickCount();
        if ( (iCurTick-iStartTick) > (sec * 1000)) {
            printf("\n Time out.\n");
            return -2;
        }
    }
    else {
        printf("\n card status fail.\n");
        return -1;
    }
}
if(CrsStatus.bStatus >= CARD_ENTRIED) {
    CRS_MSRead(...);
}
```

11.7 CRS_MSRead

## CRS_MSRead

1. Introduction

**int CRS_MSRead(IN int iMode, , OUT CRS_MS_DATA *MsTrackData)**

(2) Input Parameter

**int iMode**
Value defined for track selection to get from the magnetic card.
CRS_MS_TRACK1(1) : Get the MS Track 1 of the card
CRS_MS_TRACK2(2) : Get the MS Track 2 of the card
CRS_MS_TRACK3(3) : Get the MS Track 3 of the card
CRS_MS_TRACK_ALL(5) : Get the MS Track 1/2/3 of the card
CRS_DL_TRACK3(8) : Get MS Track 3 of the driver's license

(3) Output Parameter

**MCR_MS_DATA ***McrMsData
MCR_MS_DATA Structure Buffer's Pointer to get the MS data

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Get the MS Data from CIS

```
typedef struct tag_CRS_MS_DATA
{
    int iTrack1Len;
    unsigned char szTrack1[200];
    int iTrack2Len;
    unsigned char szTrack2[200];
    int iTrack3Len;
    unsigned char szTrack3[200];

}CRS_MS_DATA;
```

* Return Value will be treated as a HM_DEV_HW_ERR when read error occurs at all Tracks at reading MS

11.8 CRS_ICReset

## CRS_ICReset

1. Introduction

**int  CRS_ICReset(OUT int *iAtrLen, OUT unsigned char *byAtr)**

(2) Input Parameter

**void**

(3) Output Parameter

**int** iAtrLen
Length of ATR
**unsigned char** *byAtr
Array Pointer to get ATR Data

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Supply the power to IC Chip.
Get the ATR data from IC Chip

## 11.9 CRS_ICDeactivation

1. Introduction

**int CRS_ICDeactivation()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Power off the IC chip.

11.10 CRS_ICDirect

**CRS_ICDirect**

1. Introduction

**int CRS_ICDirect(int iIcSendLen, unsigned char *szIcSend, int *iIcRecvLen, unsigned char *szIcRecv)**

(2) Input Parameter

**int** iIcSendLen
Length of IC send data
**unsigned char** *szIcSend
Array pointer of data to send to IC

(3) Output Parameter

**int** *iIcRecvLen
Length of IC receive data
**unsigned char** *szIcRecv
Array Pointer to get IC Data

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

- This Command is controled at Emv Kernel

This is a command for operation under ISO7816.
User can handle all IC Cards Conforming to ISO 7816-4 and T=0, T=1

* Note : Send and Receive Data Packet refer to  Data block of Command Packet specified in ISO 7816-4 APDU

11.11 CRS_Eject

## CRS_Eject

1. Introduction

**int CRS_Eject()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Eject the Card

## 11.12 CRS_Retract

**CRS_Retract**

1. Introduction

**int CRS_Retract()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

retract the card

11.13 CRS_CISScan

**CRS_CISScan**

1. Introduction

**int CRS_CISScan(IN char *szFileName)**

(2) Input Parameter

**char *szFileName**
Name of the file to save the scanned image (path can be included)
ex) szFileName : "./test/Card"
    result files : "./test/Card_top.bmp", "./test/Card_bottom.bmp"

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_USB_COMM_FAILED
HM_DEV_USB_INVALID_BLOCK_SIZE

(5) Message

**void**

(6) Description

Scan the front(top) / rear(bottom) of the inserted card to get two bmp files.

11.14 CRS_GetLastError

**CRS_GetLastError**

(1) Prototype

**void CRS_GetLastError(OUT unsigned char szErrorCode[5])**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** szErrorCode[5]
Array Pointer which obtains final ErrorCode of CIS

(4) Return Value

**void**

(5) Message

**void**

(6) Description

it obtains final H/W ErrorCode of CIS Device

## 12. CSK

(1) It describes following interfaces to control Check/Card Scanner Kit

| | Function | Description |
|---|---|---|
| 1 | CSK Environment Setting | The environment setting for using CSK communication. |
| 2 | CSK_Open | Device USB Open |
| 3 | CSK_Close | Device USB Close |
| 4 | CSK_Status | Get CSK's Status |
| 5 | CSK_Accept | Accept Check/Card |
| 6 | CSK_CancelAccept | Cancel Accept Command  in progress |
| 7 | CSK_Eject | Eject Check/Card to inlet |
| 8 | CSK_Capture | Pull Check/Card to back |
| 9 | CSK_GetMICRData | Get MICR data of Check |
| 10 | CSK_SaveImage | Save images of the front or back of the scanned check/card |
| 11 | CSK_SaveBothImage | Save images of the front and back of the scanned check/card |
| 12 | CSK_GetMICRwithOCR | Check the OCR data on the image along with the MICR data |
| 13 | CSK_CheckFeed | Move check (forward/backward) |
| 14 | CSK_Stamp | Actuate stamp |
| 15 | CSK_FWDownload | Firmware update |
| 16 | CSK_EnableWatchdog | Enable the device's watchdog. |
| 17 | CSK_DisableWatchdog | Disable the device's watchdog. |
| 18 | CSK_GetLastError | Get final H/W Error Code of CSK |
| | | |

| | CallBack Function | Description |
|---|---|---|
| 1 | CSK_CallBackRegister | Send a Message to the registered Function whenever accept on Check / Card |
| 2 | CSK_CallBackUnregister | Unregister the registered callback function. |

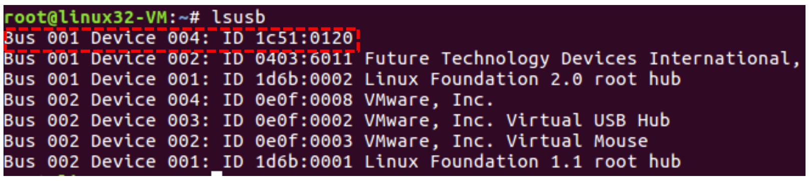
| | ETC | Description |
|---|---|---|
| 1 | CSK ERROR CODE | |

## CSK Environment Setting

12.1 CSK Environment Setting

(1) Check USB Port Recognition

First check with the lsusb command to see if the card reader is connected (vid 0x1c51, pid 0x0120)



```
root@linux32-VM:~# lsusb
Bus 001 Device 004: ID 1c51:0120
Bus 001 Device 002: ID 0403:6011 Future Technology Devices International,
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 004: ID 0e0f:0008 VMware, Inc.
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

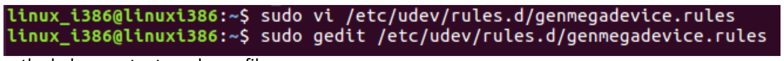
(2) install g++

You need to install g++ to compile the test program.
For details, see 1.2 Set environment in TP_Guide_for_Linux.pdf.

(3) Enable CSK USB device user account.

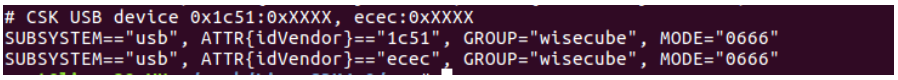
1) Create a rules file in /etc/udev/rules.d. ex) sudo touch /etc/udev/rules.d/genmegadevice.rules

ex)



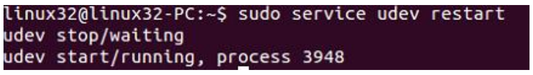
```
linux_i386@linuxi386:~$ sudo vi /etc/udev/rules.d/genmegadevice.rules
linux_i386@linuxi386:~$ sudo gedit /etc/udev/rules.d/genmegadevice.rules
```

Enter the below contents and save file.
SUBSYSTEM=="usb", ATTRS{idVendor}=="1c51", GROUP="wisecube", MODE="0666"



```
# CSK USB device 0x1c51:0xXXXX, ecec:0xXXXX
SUBSYSTEM=="usb", ATTR{idVendor}=="1c51", GROUP="wisecube", MODE="0666"
SUBSYSTEM=="usb", ATTR{idVendor}=="ecec", GROUP="wisecube", MODE="0666"
```

2) Restart Service -> sudo service udev restart



```
linux32@linux32-PC:~$ sudo service udev restart
udev stop/waiting
udev start/running, process 3948
```

3) Disconnect the USB cable of the connected CSK and reconnect it.

## 12.2 CSK_Open

**CSK_Open**

(1) Prototype

**int CSK_Open(OUT unsigned char *szVerInfo)**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** *szVerInfo
Array Pointer to obtain the F/W version of CSK

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_ALREADY_OPEN
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open USB Port of CSK
Get Firmware Version of CSK.

## 12.3 CSK_Close

**CSK_Close**

1. Introduction

**void CSK_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close USB Port of CSK

## 12.4 CSK_Status

**CSK_Status**

(1) Prototype

**int CSK_Status(OUT CSK_STATUS *sts)**

(2) Input Parameter

**void**

(3) Output Parameter

**CSK_STATUS** *sts
CSK_STATUS Structure Buffer's Pointer to get the status information of CSK

(4) Return Value

HM_DEV_OK
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Get the status information of CSK

```
typedef struct tag_CSK_STATUS
{
        unsigned char bLineStatus          :   DEV_CONNECT / DEV_DISCONNECT
        unsigned char bStsAccept           :   0 - IDLE, 1 - CSK_STS_ACCEPTING, 2 - CSK_STS_ACCEPTED
        unsigned char  bStsKind            :   0 - EMPTY, 1 - CHECK, 2 - CARD
        unsigned char  bInletRightFront    :   0 - NOT DETECTED, 1 - DETECTED
        unsigned char  bInletRightRear     :   0 - NOT DETECTED, 1 - DETECTED
        unsigned char  bFront              :   0 - NOT DETECTED, 1 - DETECTED
        unsigned char  bInletLeftFront     :   0 - NOT DETECTED, 1 - DETECTED
        unsigned char  bInletLeftRear      :   0 - NOT DETECTED, 1 - DETECTED
        unsigned char  bCenterFront        :   0 - NOT DETECTED, 1 - DETECTED
        unsigned char  bCoverStatus        :   0 - OPEN, 1 - CLOSE
        unsigned char  bCenterRear         :   0 - NOT DETECTED, 1 - DETECTED
        unsigned char  bExitSensor         :   0 - NOT DETECTED, 1 - DETECTED
} CSK_STATUS
```

12.5 CSK_Accept

**CSK_Accept**

(1) Prototype

**int CSK_Accept(int iMediaType)**

(2) Input Parameter

**int iMediaType**
Value defined for type of media to accept
CSK_KIND_CHECK(1) : Accept Check
CSK_KIND_CARD(2) : Accept Card

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Wait until you put a check or card in the entrance.
If the insertion is detected, the scan will proceed.

12.6 CSK_CancelAccept

**CSK_CancelAccept**

(1) Prototype

**int CSK_CancelAccept()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Accept cancel

12.7 CSK_Eject

## CSK_Eject

1. Introduction

**int CSK_Eject()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Eject Check / Card

**CSK_Capture**

12.8 CSK_Capture

1. Introduction

**int CSK_Capture()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Pull check / card back

12.9 CSK_GetMICRData

**CSK_GetMICRData**

1. Introduction

**int CSK_GetMICRData(char* szMICRData)**

(2) Input Parameter

**void**

(3) Output Parameter

**char** *szMICRData
Array Pointer to obtain MICR data of Check

(4) Return Value

HM_DEV_OK
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Get MICR data of Check

## 12.10 CSK_SaveImage

**CSK_SaveImage**

(1) Prototype

**int CSK_SaveImage(int nSide, const char* szFile)**

(2) Input Parameter

**int iSide**
CSK_IMG_FRONT(0), CSK_IMG_BACK(1)
**const char* szFile**
File name of the image of the scanned Check/Card

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Save images of the front or back of the scanned check/card

12.11 CSK_SaveBothImage

## CSK_SaveBothImage

1. Introduction

**int CSK_SaveBothImage(const char* szFrontFile, const char* szBackFile)**

(2) Input Parameter

**const char*** szFrontFile
File name of the front image of the scanned Check/Card
**const char*** szBackFile
File name of the back image of the scanned Check/Card

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Save images of the front and back of the scanned check/card.

12.12 CSK_GetMICRwithOCR

## CSK_GetMICRwithOCR

1. Introduction

**int CSK_GetMICRwithOCR(int nDpi, const char* szFrontImg, const char* szBackImg, char* szOCRData, int* pOrient)**

(2) Input Parameter

**int** nDPI
Dots per inch of Image
**const char*** szFrontImg
Saved front image file
**const char*** szBackImg
Saved back image file

(3) Output Parameter

**char*** szOCRData
Scanner MICR data or image OCR data
**int*** pOrient
Location of MICR data in the image

| | |
|---|---|
| 0 : CSK_MICR_UNKNOWN | 1: CSK_MICR_FRONTRIGHT |
| 2: CSK_MICR_FRONTLEFT | 3: CSK_MICR_BACKRIGHT |
| 4: CSK_MICR_BACKLEFT | |

(4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Check the OCR data on the image along with the MICR data
The scanner's MICR data is given priority.

You should have an e13b_check file in the /usr/local/bin directory.
After installing the SDK Debian package, run e13b_check in the /usr/local/bin directory to get OCR.

12.13 CSK_CheckFeed

**CSK_CheckFeed**

1. Introduction

**int CSK_CheckFeed(int iDirection, int mm)**

(2) Input Parameter

**int** iDirection
CSK_FEED_FORWORD(0), CSK_FEED_BACKWORD(1)
**int mm**
Distance, unit is millimeters

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_NOT_READY
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Move the check
iDirection
CSK_FEED_FORWORD(0) : Move check forward(to rear side)
CSK_FEED_BACKWORD(1) : Move check backward (to inlet side)

12.14 CSK_Stamp

**CSK_Stamp**

1. Introduction

**int CSK_Stamp()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Actuate stamp the check
Push stamp pad to upward by pull solenoid, then release it

12.15 CSK_FWDownload

## CSK_FWDownload

(1) Prototype

**int CSK_FWDownload(const char* szFwFile)**

(2) Input Parameter

**const char*** szFwFile
Firmware file name including path

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Firmware update
When the firmware update is complete, the USB connection is reconnected.

## 12.16 CSK_EnableWatchdog

**CSK_EnableWatchdog**

(1) Prototype

**int CSK_EnableWatchdog(int seconds)**

(2) Input Parameter

**int** seconds
Time to check at Watchdog. unit is seconds

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Enable the device's watchdog.
If there be no USB command during timer value, device execute reset itself.
When reset, watchdog timer is set to 0.

## 12.17 CSK_DisableWatchdog

**CSK_DisableWatchdog**

(1) Prototype

**int CSK_DisableWatchdog()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_HW_ERR
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Disable the device's watchdog

12.18 CSK_GetLastError

**CSK_GetLastError**

(1) Prototype

**void CSK_GetLastError(OUT unsigned char szErrorCode[5])**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** szErrorCode[5]
Array Pointer which obtains final ErrorCode of CSK

(4) Return Value

**void**

(5) Message

**void**

(6) Description

it obtains final H/W ErrorCode of CSK Device

## 12.19 CSK_CallBackRegister

(1) Prototype

**void CSK_CallBackRegister(callback_accept handler)**

(2) Input Parameter

**callback_accept** handler
CallBacked Function

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

When the medium is inserted after the Accept command, the function is called.

## 12.20 CSK_CallBackUnregister

(1) Prototype

**void CSK_CallBackUnregister()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Unregister the registered callback function.

12.21 CSK ERROR CODE

## CSK ERROR CODE

Error Codes Table

| CODE | Description |
|---|---|
| K0000 | NORMAL |
| K0001 | error occurs during pick up |
| K0002 | Jamming occurs during feed |
| K0003 | Jamming occurs during micr |
| K0004 | Jamming occurs during scan |
| K0005 | Jamming occurs during escrow |
| K0006 | Jamming occurs during reject |
| K0007 | Jamming occurs during print |
| K0008 | Jamming occurs during stamp |
| K0009 | Jamming occurs during stack |
| K0010 | Invalid image setup argument |
| K0011 | Error over escrow |
| K0012 | no check/card at inlet. |
| K0013 | no check/card at escrow. |
| K0014 | error alignment |
| K0015 | error pick up |
| K0016 | Error max escrow |
| K0017 | Error no feed |
| K0018 | Error no check |
| K0019 | Detected inside sensor |
| K0020 | Detected exit sensor |
| K0021 | jamming double |
| K0022 | Jamming occurs during feedback |
| K0023 | Error slip |
| K0030 | Timeout Watchdog |
| K0032 | Error INS_TAKEN |
| K0033 | Error JAM_IN |
| K0044 | long card inserted |
| K0045 | Short card inserted |
| K0050 | Error INIT |
| K0051 | Error EEP_WR |
| K0052 | Error EEP_RD |
| K0053 | Failure during firmware download due to address or size mismatch |
| K0054 | Undefined command |
| K0055 | Illegal frame format for command |
| K0057 | Illegal argument in command |
| K0057 | Illegal argument in command |
| K0070 | Error USB |
| K0071 | Error USB IO |
| K0072 | Invalid parameter |
| K0073 | Access denied (insufficient permissions) |
| K0074 | No such device (it may have been disconnected) |
| K0075 | Entity not found |
| K0076 | Resource busy |
| K0077 | Operation timed out |
| K0078 | Error Overflow |
| K0079 | Pipe error |
| K0080 | System call interrupted (perhaps due to signal) |
| K0081 | Insufficient memory |
| K0082 | Operation not supported or unimplemented on this platform |
| K0083 | No Initialize data |
| K0084 | memory alloc error |
| K0091 | Error Syntax |
| K0092 | not supported command |
| K0093 | Scanner is not connected |
| K0094 | no ocr encoding |
| K0095 | no ocr engine |
| K0096 | No response from scanner |
| K0097 | Error during file handling |
| K0098 | No scanned image data |
| K0099 | Other error |
| | |

13. HOP

(1) It describes following interfaces to control Coin Hopper

| | Function | Description |
|---|---|---|
| 1 | HOP_Open | Open Serial Port |
| 2 | HOP_Close | Close Serial Port |
| 3 | HOP_Reset | Initialize the Device |
| 4 | HOP_Status | Get status of HOP |
| 5 | HOP_Dispense | Dispense the coins from defined Hopper of HOP |
| 6 | HOP_Purge | All coins are purged until the designated Hopper is empty |
| 7 | HOP_GetLastError | Get the final Error Code of HOP |

## 13.1 HOP_Open

**HOP_Open**

(1) Prototype

**int HOP_Open(IN const char* szPortName, IN int iExtensionCount, OUT unsigned char szVersion[15])**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to HOP ( Ex) "/dev/ttyS1" )
**int** iExtensionCount
The number of additionally installed hoppers (0~3). (Default: 3, Extended: 3)
ex) If 0, only the default 3 Hoppers are used.

(3) Output Parameter

**unsigned char** szVersion[15]
Array Pointer to obtain the F/W version of HOP

4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open the Serial Port of HOP
Start the thread for HOP status.

13.2 HOP_Close

## HOP_Close

1. Introduction

**void HOP_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close Serial Port of HOP
End the  thread of HOP status

13.3 HOP_Reset

**HOP_Reset**

(1) Prototype

**int HOP_Reset(OUT int *iLastErrHopper)**

(2) Input Parameter

**void**

(3) Output Parameter

**int** *iLastErrHopper
Pointer to the index of the hopper in error. (If all Hoppers are normal, 0 is returned)

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Initialize the device

13.4 HOP_Status

## HOP_Status

(1) Prototype

**int HOP_Status(OUT HOP_STATUS *sts)**

(2) Input Parameter

**void**

(3) Output Parameter

**HOP_STATUS** *sts
HOP_STATUS Structure Buffer's Pointer to get the status information of HOP

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Get the status information of HOP

```
typedef struct tag_HOP_STATUS
{
        char bLineStatus[6];        :   DEV_CONNECT / DEV_DISCONNECT
                                        Displays the connection status with each hopper
        char bNearend[6];           :   HOP_NORMAL/ HOP_NEAREND
                                        Display whether coins is near end or not
        char  bPathRemain[6];       :   HOP_EMPTY / HOP_JAMMED
                                        Display whether coins exist in each hopper path
} HOP_STATUS
```

## HOP_Dispense

13.5 HOP_Dispense

1. Introduction

**int HOP_Dispense(IN int iReqDispense[6], OUT int iDispensedCnt[6], OUT int* iLastErrHopper)**

(2) Input Parameter

**int** iReqDispense[6]
Interger Array Pointer which is designated number of coins to dispense from each hopper

(3) Output Parameter

**int** iDispensedCnt[6]
Interger Array Pointer to obtain the result of dispense operation from each hopper
**int** *iLastErrHopper
Pointer to the index of the hopper in error. (If all Hoppers are normal, 0 is returned)

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Dispense the designated number of coins from the each hopper of HOP Device
if an error occurs in the hopper, the index of the error hopper is set

13.6 HOP_Purge

## HOP_Purge

1. Introduction

**int HOP_Purge(IN int iPurgeHopper, OUT int iDisensedCnt[6], OUT int* iLastErrHopper)**

(2) Input Parameter

**int** iPurgeHopper
Hopper index to purge all coins (0 ~ 6)
0 : All Hopper, 1: Hopper#1, 2: Hopper#2, 3: Hopper#3, 4: Extended Hopper#1, 5: Extended Hopper#5, 6: Extended Hopper#3

(3) Output Parameter

**int** iDispensedCnt[6]
Interger Array Pointer to obtain the result of dispense operation from each hopper
**int** *iLastErrHopper
Pointer to the index of the hopper in error. (If all Hoppers are normal, 0 is returned)

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

All coins are purged until the designated Hopper is empty

13.7 HOP_GetLastError

## HOP_GetLastError

(1) Prototype

**void HOP_GetLastError(OUT unsigned char szErrorCode[5])**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** szErrorCode[5]
Array Pointer which obtains final ErrorCode

(4) Return Value

**void**

(5) Message

**void**

(6) Description

it obtains final ErrorCode of Coin Hopper Device

1.Common Error Code

| Code | Description |
|---|---|
| HDN01 | The communication line is down |
| HDN02 | Send/Receive Timeout |
| HDN03 | Communication buffer overflow |
| HDN06 | Another process or thread is occupying the serial port |
| | |

2. Hopper Error Code

| Code | Description |
|---|---|
| H0100 | Absolute maximum current exceeded |
| H0200 | Payout timeout occurred. Hopper is empty |
| H0800 | Opto fraud attempt, path blocked during idle |
| H1000 | Opto fraud attempt, short-circuit during idle |
| H2000 | Opto blocked permanently during payout |
| H4000 | Checksum A error |
| H4100 | Checksum B error |
| H4200 | Checksum C error |
| H4300 | Checksum D error |
| | |

14. MCD

(1) It describes following interfaces to control Card Dispenser

| | Function | Description |
|---|---|---|
| 1 | MCD_Open | Open Serial Port |
| 2 | MCD_Close | Close Serial Port |
| 3 | MCD_Reset | Initialize the Device |
| 4 | MCD_Status | Get status of MCD |
| 5 | MCD_Eject | Eject a card |
| 6 | MCD_Exit | Non-retrievable eject |
| 7 | MCD_Retrieve | Retrieve a card |
| 8 | MCD_DispenseWait | Dispense a card and wait |
| 9 | MCD_DispenseEject | Dispense a card and eject |
| 10 | MCD_DispenseExit | Dispense a card and non-retrievable eject |
| 11 | MCD_Diagnosis | Diagnosis the sensors |
| 12 | MCD_MotorTest | Test the DC Motor |
| 13 | MCD_ShutterTest | Test the shutter solenoid |
| 14 | MCD_WriteTrackData | Write a Track1 and Track2 and Track3 data in the MS Card |
| 14 | MCD_GetLastError | Get the final Error Code of MCD |

* From the firmware version of V01.00.00, the card dispenser device has device id which can be set with dip switches.
For the old versions of firmware and hardware of device, the DeviceID property is ignored,
that is, because the control does not check the value of the property, any value is available to use the cash dispenser device.
However, if the device is updated to new firmware and hardware, the value of DeviceID must be identical with the dip switch setting.

* Dipswitch Specification

| Dipswitch #2 | Dipswitch #3 | Dipswitch #4 | DeviceID |
|---|---|---|---|
| OFF | X | X | 0 |
| ON | OFF | OFF | 1 |
| ON | ON | OFF | 2 |
| ON | OFF | ON | 3 |
| ON | ON | ON | 4 |

X: Don't care

## 14.1 MCD_Open

**MCD_Open**

(1) Prototype

**int MCD_Open(IN const char* szPortName, OUT unsigned char szVersion[10])**

(2) Input Parameter

**const char** *szPortName
Serial Port of connecting to MCD ( Ex) "/dev/ttyS1" )

(3) Output Parameter

**unsigned char** szVersion[10]
Array Pointer to obtain the F/W version of MCD

4) Return Value

HM_DEV_OK
HM_DEV_ALREADY_OPEN
HM_DEV_INTERNAL_ERR
HM_DEV_OPENPORTFAIL
HM_DEV_RXOVERFLOW
HM_DEV_TIMEOUT

(5) Message

**void**

(6) Description

Open the Serial Port of MCD
Start the thread for MCD status.

## 14.2 MCD_Close

**MCD_Close**

1. Introduction

**void HOP_Close()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Close Serial Port of MCD
End the  thread of MCD status

## 14.3 MCD_Reset

**MCD_Reset**

(1) Prototype

**int MCD_Reset()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Initialize the Card Dispenser.
The device retrieves all cards in the passage to the reject bin during initializing.

14.4 HOP_Status

**MCD_Status**

(1) Prototype

**int MCD_Status(OUT MCD_STATUS *sts)**

(2) Input Parameter

**void**

(3) Output Parameter

**MCD_STATUS** *sts
MCD_STATUS Structure Buffer's Pointer to get the status information of MCD

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Get the status information of MCD

```
typedef struct tag_MCD_STATUS
{
        char bLineStatus;          :   DEV_CONNECT / DEV_DISCONNECT
                                       Displays the connection status with each hopper
        char bTransporter;         :   MCD_STS_NORMAL / MCD_STS_S1_DETECTED / MCD_STS_S2_DETECTED
                                       Display whether card exist in transporter path
        char  bReadporter;         :   MCD_STS_NORMAL / MCD_STS_DETECTED
                                       Display whether card exist in readporter path
        char bOutlet;              :   MCD_STS_NORMAL / MCD_STS_PRESENT
                                       Displays whether card is present in outlet
        char  bHPSolenoid;         :   MCD_STS_STANDBY / MCD_STS_HPDOWN
                                       Display whether solenoid is pulled down or Stand-by
        char bNearEnd;             :   MCD_STS_NORMAL / MCD_STS_LOW / MCD_STS_END
                                       Display Whether there are card to be dispensed
        char  bRejectFull;         :   MCD_STS_NORMAL / MCD_STS_REJFULL
                                       Display whether reject bin is full
} MCD_STATUS
```

13.5 MCD_Eject

## MCD_Eject

1. Introduction

**int MCD_Eject()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Eject a card. Retrieve method can be called after this method
This method is usually used after DispenseAndWait succeeds.

14.6 MCD_Exit

**MCD_Exit**

1. Introduction

**int MCD_Exit()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Draw off a card in the draw route of the device to the position where a card cannot be retrieved.

The Card Dispenser ejects the dispensed card to the end of outlet where the card cannot be retrieved by Retrieve method.
This method can be used when a card is located at the waiting position(after DispenseAndWait method is called),
the eject position(after Eject related method is called), and the exit position(after Exit related method is called)

14.7 MCD_Retrieve

## MCD_Retrieve

1. Introduction

**int MCD_Retrieve()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Retrieve cards in the passage to the reject bin.

14.8 MCD_DispenseWait

## MCD_DispenseWait

1. Introduction

**int MCD_DispenseWait(IN int bMsRead, OUT MCD_MS_DATA *McrMsData)**

(2) Input Parameter

**int** bMsRead
Set whether to read MS data from dispensed card (0: without read MS data, 1: read MS data)

(3) Output Parameter

**MCD_MS_DATA*** McrMsData
MCD_MS_DATA Structure Buffer's Pointer to get the MS data

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

The Card Dispenser picks up a card and reads magnetic data of the card according to the parameter
and then waits the next command

```
typedef struct tag_MCD_MS_DATA
{
    int iTrack1Len;                  :   Data Length for Track 1
    unsigned char szTrack1[200];     :   MS Data for Track 1
    int iTrack2Len;                  :   Data Length for Track 2
    unsigned char szTrack2[200];     :   MS Data for Track 2
    int iTrack3Len;                  :   Data Length for Track3
    unsigned char szTrack3[200];     :   MS Data for Track 3

}MCD_MS_DATA;
```

* In Firmware version v00.00.01, dispensing a card without reading MS data (that is, CardRead is FALSE) is not supported

14.9 MCD_DispenseEject

## MCD_DispenseEject

1. Introduction

**int MCD_DispenseEject(IN int bMsRead, OUT MCD_MS_DATA *McrMsData)**

(2) Input Parameter

**int** bMsRead
Set whether to read MS data from dispensed card (0: without read MS data, 1: read MS data)

(3) Output Parameter

**MCD_MS_DATA*** McrMsData
MCD_MS_DATA Structure Buffer's Pointer to get the MS data

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

The Card Dispenser picks up a card and reads magnetic data of the card according to the parameter,
and then ejects the card.

```
typedef struct tag_MCD_MS_DATA
{
    int iTrack1Len;                   :   Data Length for Track 1
    unsigned char szTrack1[200];      :   MS Data for Track 1
    int iTrack2Len;                   :   Data Length for Track 2
    unsigned char szTrack2[200];      :   MS Data for Track 2
    int iTrack3Len;                   :   Data Length for Track3
    unsigned char szTrack3[200];      :   MS Data for Track 3

}MCD_MS_DATA;
```

* In Firmware version v00.00.01, dispensing a card without reading MS data (that is, CardRead is FALSE) is not supported

14.10 MCD_DispenseExit

## MCD_DispenseExit

1. Introduction

**int MCD_DispenseExit(IN int bMsRead, OUT MCD_MS_DATA *McrMsData)**

(2) Input Parameter

**int** bMsRead
Set whether to read MS data from dispensed card (0: without read MS data, 1: read MS data)

(3) Output Parameter

**MCD_MS_DATA*** McrMsData
MCD_MS_DATA Structure Buffer's Pointer to get the MS data

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

The Card Dispenser picks up a card and reads magnetic data from the card according to the CardRead parameter and then ejects the card to the end of outlet. The dispensed card cannot be retrieved by Retrieve method.

```
typedef struct tag_MCD_MS_DATA
{
    int iTrack1Len;                    :   Data Length for Track 1
    unsigned char szTrack1[200];       :   MS Data for Track 1
    int iTrack2Len;                    :   Data Length for Track 2
    unsigned char szTrack2[200];       :   MS Data for Track 2
    int iTrack3Len;                    :   Data Length for Track3
    unsigned char szTrack3[200];       :   MS Data for Track 3

}MCD_MS_DATA;
```

14.11 MCD_Diagnosis

## MCD_Diagnosis

1. Introduction

**int MCD_Diagnosis()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

The Card Dispenser diagnoses the sensors, picks up a card, dispense the card and to S4 sensor

* In Firmware version v00.00.01, this method is not supported

14.12 MCD_MotorTest

## MCD_MotorTest

1. Introduction

**int MCD_MotorTest(int iMotorAction)**

(2) Input Parameter

**int** iMotorAction
the direction to test (0 ~ 6)
 0 – M1 & M2 DC MOTOR OFF
 1 – M1 DC MOTOR FORWARD
 2 – M1 DC MOTOR BACKWARD
 3 – M2 DC MOTOR FORWARD
 4 – M2 DC MOTOR BACKWARD
 5 – M1 & M2 DC MOTOR FORWARD
 6 - M1 & M2 DC MOTOR BACKWARD

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Test the DC MOTOR of the device to turn in various directions.
The test must be done after all the cards in the passage are retrieved. Do not make the motor move for long time.

14.13 MCD_ShutterTest

## MCD_ShutterTest

1. Introduction

**int MCD_ShutterTest(int iShutterAction)**

(2) Input Parameter

**int** iShutterAction
Shutter on/off test (0 : Shutter OFF, 1: Shutter ON)

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

Test if the shutter solenoid is working well or not.

14.14 MCD_DispenseExit

## MCD_DispenseExit

1. Introduction

**int MCD_WriteTrackData(IN unsigned char *szTrack1, IN unsigned char *szTrack2, IN unsigned char *szTrack3,**
**OUT MCD_MS_DATA *McrMsData)**

(2) Input Parameter

**unsigned char*** szTrack1
magnetic track1 data to write in card (The maximum buffer size is 200 bytes)
**unsigned char*** szTrack2
magnetic track2 data to write in card (The maximum buffer size is 200 bytes)
**unsigned char*** szTrack3
magnetic track2 data to write in card (The maximum buffer size is 200 bytes)

(3) Output Parameter

**MCD_MS_DATA*** McrMsData
MCD_MS_DATA Structure Buffer's Pointer to get the MS data

(4) Return Value

HM_DEV_OK
HM_DEV_RXOVERFLOW
HM_DEV_INTERNAL_ERR
HM_DEV_TIMEOUT
HM_DEV_HW_ERR

(5) Message

**void**

(6) Description

The Card Dispenser write a track1 and track2 and track3 data in the MS Card

```
typedef struct tag_MCD_MS_DATA
{
    int iTrack1Len;                  :   Data Length for Track 1
    unsigned char szTrack1[200];     :   MS Data for Track 1
    int iTrack2Len;                  :   Data Length for Track 2
    unsigned char szTrack2[200];     :   MS Data for Track 2
    int iTrack3Len;                  :   Data Length for Track3
    unsigned char szTrack3[200];     :   MS Data for Track 3

}MCD_MS_DATA;
```

14.15 MCD_GetLastError

## MCD_GetLastError

(1) Prototype

**void MCD_GetLastError(OUT unsigned char szErrorCode[10], OUT char szErrorDesc[256])**

(2) Input Parameter

**void**

(3) Output Parameter

**unsigned char** szErrorCode[10]
Array Pointer which obtains final ErrorCode
**unsigned char** szErrorDesc[256]
Array Pointer which obtains final Error Description

(4) Return Value

**void**

(5) Message

**void**

(6) Description

The error code is a 8-length string (looks like “2382E280”),
which is consisted of 4 parts (“23”, “82”, “E2” and “80”) having 2-length string per each.
The first 2-length string indicates the error status, the other three strings indicate error codes described in the below section

The following codes are the first two digits of the error code when an error occurs.
If the value is MCDU_NORMAL, an application can go on its process but if the value is MCD_SEMI_NORMAL,
the application must check device status with its sensors and then decide whether the application can continue or not.
When MCDU_ERROR is the value, the device cannot work until it is initialized by 'MCD_Reset' method.
(After firmware version v00.00.04, the application can continue even after an error occurs.)
If the error cannot be recovered, a person should settle up the problem directly

| Code Name | Value | Description |
|---|---|---|
| MCD_NORMAL | 20 | No Error |
| MCD_SEMINORMAL | 21 | Semi-normal |
| MCD_ERROR | 23 | Error |

The following error codes are the last 2-length string of the error code
when the first part is MCD_ERROR (“23”) and the other two parts are “80”.
The description of the code can be retrieved with ErrorDesc property

| Code Name | Value | Description |
|---|---|---|
| MCD_ERR_CARD_POSITION | 81 | The position of a card is invalid |
| MCD_ERR_SOLENOID_CLOSE | 82 | Solenoid H.P Close Error |
| MCD_ERR_SOLENOID_OPEN | 83 | Solenoid H.P Open Error |
| MCD_ERR_READBUFF_OVERFLOW | 84 | Decorder Read Buffer Overflow |
| MCD_ERR_RETREIVE_ERROR | 85 | Card Retrieve Error |
| MCD_ERR_UNKNOWN_COMMAND | 86 | Unknown Commands |
| MCD_ERR_REJECTBIN_FULL | 87 | The reject bin is full |
| MCD_ERR_ERROR_NOT_CLEARED | 88 | Error not clear |
| MCD_ERR_NOT_DETECTED | 91 | No card is detected |
| MCD_ERR_NO_CARD | 92 | There is no card to be dispensed |
| MCD_ERR_LONG_LENGTH | 93 | The length of the card is too long |
| MCD_ERR_SHORT_LENGTH | 94 | The length of the card is too short |
| MCD_ERR_NOT_AVAILABLE | 95 | The card cannot be processed |

1.Common Error Code

| Code | Description |
|---|---|
| MCDXDN01 | The communication line is down |
| MCDXDN02 | Send/Receive Timeout |
| MCDXDN03 | Communication buffer overflow |
| MCDXDN06 | Another process or thread is occupying the serial port |

14.16 Analyzing Error Code

## Analyzing Error Code

If the second, third and the last part are not “80”(NOERROR) and the first byte is not ERROR_NORMAL (“20”),
an application can analyze the error codes after converting the part to one BYTE data
because a one byte error code has meaning for each bit or just use the ErrorDesc property, which contains the analyzed description

Ex>
ErrorCode : “2382E283”
ErrorDesc : “Error Detected on Sensor2 while Waiting for Sensor Off, Detected Error(s) is Jammed,Abnormal Direction(Sensor1, Sensor2)”

When an application analyzes the error code by itself, refer to the following description

*the second byte of the error code: detected sensor

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | HP | S4 | S3 | S2 | S1 |

Ex> if the second = “83”(0x83) S1 sensor and S2 sensor are ON
“Error Detected on Sensor2 while Waiting for Sensor Off, Detected Error(s) is Jammed,Abnormal Direction(Sensor1, Sensor2)”

*the third byte of the error code: action

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 1 | Direction | Jam | Abnormal OFF | Abnormal ON | Stand ready to Move | Stand ready for OFF | Stand ready for ON |

Ex> if the third = “E2”(0xE2) Detect Abnormal Direction and Jammed while waiting for sensor off
“Error Detected on Sensor2 while Waiting for Sensor Off, Detected Error(s) is Jammed,Abnormal Direction(Sensor1, Sensor2)”

*the last byte of the error code: error detected sensor

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | HP | S4 | S3 | S2 | S1 |

Ex> if the last part = “83”(0x83) Error is detected on S2 sensor and S1 sensor
“Error Detected on Sensor2 while Waiting for Sensor Off, Detected Error(s) is Jammed,Abnormal Direction(Sensor1, Sensor2)”

15. ETC

(1) It describes for use of HMDev Control DLL.

| Function | | Description |
|---|---|---|
| 1 | HMDEV_SetLogConf | Enable/Disable Log of Device (Not used since SDK v5.44 version) |
| 2 | GetDevLogConfig | Get log setting value |
| 3 | GetTickCount | Retrieves the number of milliseconds of current time |
| 4 | CheckTimeOut | Check the time has elapsed |
| | | |
| | | |

** The log file directory was fixed at the '/var/log/genmegadevice' directory from 0.5.45 version.

## 15.1 HMDEV_SetLogConf **(Not used since SDK v5.44 version)**

**HMDEV_SetLogConf**

(1) Prototype

**int HMDEV_SetLogConf(IN char* szUnitName,IN int iSetValue, IN unsigned char* szFilePath)**

(2) Input Parameter

**char ***szUnitName
String Pointer of Unit Name
Ex) "CDU" or "RPU" or "EPP" ---
**int** iSetValue
Value of Log Enable/Disable
Ex) LOG_DISALBE or LOG_ENABLE
**unsigned char ***szFilePath
String Pointer of Path
Ex) "./genmega/data"

(3) Output Parameter

**void**

(4) Return Value

HM_DEV_OK
HM_DEV_INTERNAL_ERR
HM_DEV_INVALID_DATA

(5) Message

**void**

(6) Description

Select whether to save the Communication Log for each device. (Enable / Disable)
Configuration file is created in the specified Path. (~/HmDevice.Conf)
Also, If you select Enable, Communication Log is saved by date in the specified Path. ( ~/HmDevTrace/0517000.txt)
This function must be called before device open( ex) MCR_Open()).

**The log file directory was fixed at the '/var/log/genmegadevice' directory from 0.5.45 version.**

15.2 GetDevLogConfig

## GetDevLogConfig

(1) Prototype

**void GetDevLogConfig(OUT int *pLogLevel, OUT int *pPreserveDays, OUT int *pMaxLogSize, OUT char *szLogPath)**

(2) Input Parameter

**void**

(3) Output Parameter

**int *** pLogLevel
Level value to be saved in log file. (0~99, Not Save Value : -1, default : 1)
**int*** pPreserveDays
Log file storage days (1~365, default : 30)
**int*** pMaxLogSize
Maximum storage capacity for one log file. (default : 100000000)
**char*** szLogPath
Path where log files will be saved (default path: /var/log/genmegadevice/HmDevTrace/)

(4) Return Value

**void**

(5) Message

**void**

(6) Description

Gets the configuration values of SDK log storage (SDK configuration file location is: /etc/genmegadevice/genmegadevice.cfg)
Log Configuration value is LogLevel, Preserve Days, MaxLogFileSize, Log File Path
Default value : LogLevel = 1, PreserveDays = 30, MaxLogFileSize = 100000000
LogFilePath = /var/log/genmegadevice/HmDevTrace/

## 15.3 GetTickCount

**GetTickCount**

(1) Prototype

**unsigned int GetTickCount()**

(2) Input Parameter

**void**

(3) Output Parameter

**void**

(4) Return Value

**unsigned int**
Returns the number of milliseconds as an unsigned int.

(5) Message

**void**

(6) Description

Return the number of milliseconds of current time

### 15.4 CheckTimeOut

**CheckTimeOut**

(1) Prototype

**int CheckTimeOut(unsigned int dwStart, int dwTimeOut)**

(2) Input Parameter

**unsigned int** dwStart
The value returned by the GetTickCount function call
Ex) unsigned int dwStart = GetTickCount();

**int** dwTimeOut
Maximum wait time (milliseconds)
Ex) int iTimeOut = CheckTimeOut(dwStart, 30*1000);  // Wait for 30 seconds

(3) Output Parameter

**void**

(4) Return Value

**int**
0 : Inputted time of the parameter has not elapsed
1 : Inputted time of the parameter has elapsed

(5) Message

**void**

(6) Description

Check whether the inputted time of the parameter has elapsed

ex)
```
    unsigned int dwStart = 0;
    int iTimeOut = 0;
    dwStart = GetTickCount();
    while(1)
    {
        iTimeOut = CheckTimeOut(dwStart, 30*1000);  // Wait for 30 seconds
        if(iTimeOut == 1) {
            printf("30 seconds elapsed");
            break;
        }
        usleep(1000);  // 1ms
    }
```