TELIUM SDK

UCM COMPONENT Reference Manual

Reference: SMO/DFO-0069
Revision: R
Enter Date: 18/04/2013

Revision Approval: R

	Name	Function
Written by:	C.PLESSIS	RTL
Checked or approved by:	J.DOBLADO	Project Manager
Authorized by:	A.SOUBIRANE	UCM TELIUM Product Manager

Revision Record

Issue No.	Issue Date	Nature of amendment
A	12/15/2006	First issue
B	01/25/2007	Document translated in English
C	10/11/2007	Updates and corrections. No description of UCM parameters
D	01/02/2008	Adding external swipe (vending) Adding external device payment
E	11/22/2008	Adding external swipe ISO123
F	08/31/2009	Adding new Led and Buzzer Channel
G	07/10/2009	Adding new API
H	14/01/2010	Minor Correction
I	26/04/2010	Adding new API
J	25/10/2010	Adding new API
K	03/03/2011	Update Error list
L	03/05/2011	Update Error list
M	20/12/2011	Adding new API for IUN
O	14/12/2012	Minor corrections
P	17/07/2012	Adding new API.
Q	28/09/2012	Correction maintenance mode
R	18/09/2013	Reorganization of chapters

TABLE OF CONTENTS

1. PURPOSE OF THE DOCUMENT

The purpose of this document is to describe the UCM TELIUM software architecture and the interfaces of the embedded application software.

2. DOCUMENT HISTORY

– Revision A

Creation of the "DOCUMENT HISTORY" section.

– Revision B:

"PARAMETER SET-UP FILE" section: modifications made to take account of MPC10S peripherals. "ILIBUCM_Pinpad_Input" section: additional information on figure keys, function keys and pin code inputs. "ILIBUCM_Pinpad_Status" section: additional Information.

– Revision C

Corrections of input data, parameters set-up file chapter, breakdown and IAC chapters.

Simplification of kinematic description.

– Revision D

Adding chapter "DLL PROTOCOL INTERFACE”
Adding external swipe in vending
Adding external device payment.

– Revision E

Adding external swipe ISO 1 2 3
Adding UCMC parameters download.
Adding chapter “Payment with automatic vending machine and ICC device managed by application.

– Revision F

Adding LED and Buzzer functions with channel

– Revision G

Adding function to initialize Banking Host message Application simulation
Adding Events(USB, Key and LLT press buton) functions
Adding reading device configuration functions

– Revision J

Adding function to initialize DLL Security
Error list update

– Revision K

Description command to initialize DLL Security
Error list update

- Revision L

Error list update

- Revision M

Adding function for IUN backlight
Adding function for French Domain (ITP)

- Revision O

Correction iLIBUCM_Icc_Status
Correction iLIBUCM_Pinpad_Getchar()
Warning using iLIBUCM_Pinpad_Input() and iLIBUCM_Icc_PCode()

- Revision P

Modification API iLIBUCM_Pinpad_Cmd
New command for iLIBUCM_Icc_Cmd
Warning using iLIBUCM_Pinpad_Input() and iLIBUCM_Pinpad_Getchar()

- Revision Q

Maintenance chapter modification.
Adding chapter Sleep Power Management
Read swipe description.
Icc_Power returns description

- Revision R

Reorganization of chapters. New API iLIBUCM_Device_Ttestall_STOP. Buzzer, power management, CAM power on synchro, buzzer, backlight descriptions.

3. INPUT DATA

OPE1275 Manual " TELIUM Manager user’s guide "

OPE1286 System Reference Manuel"

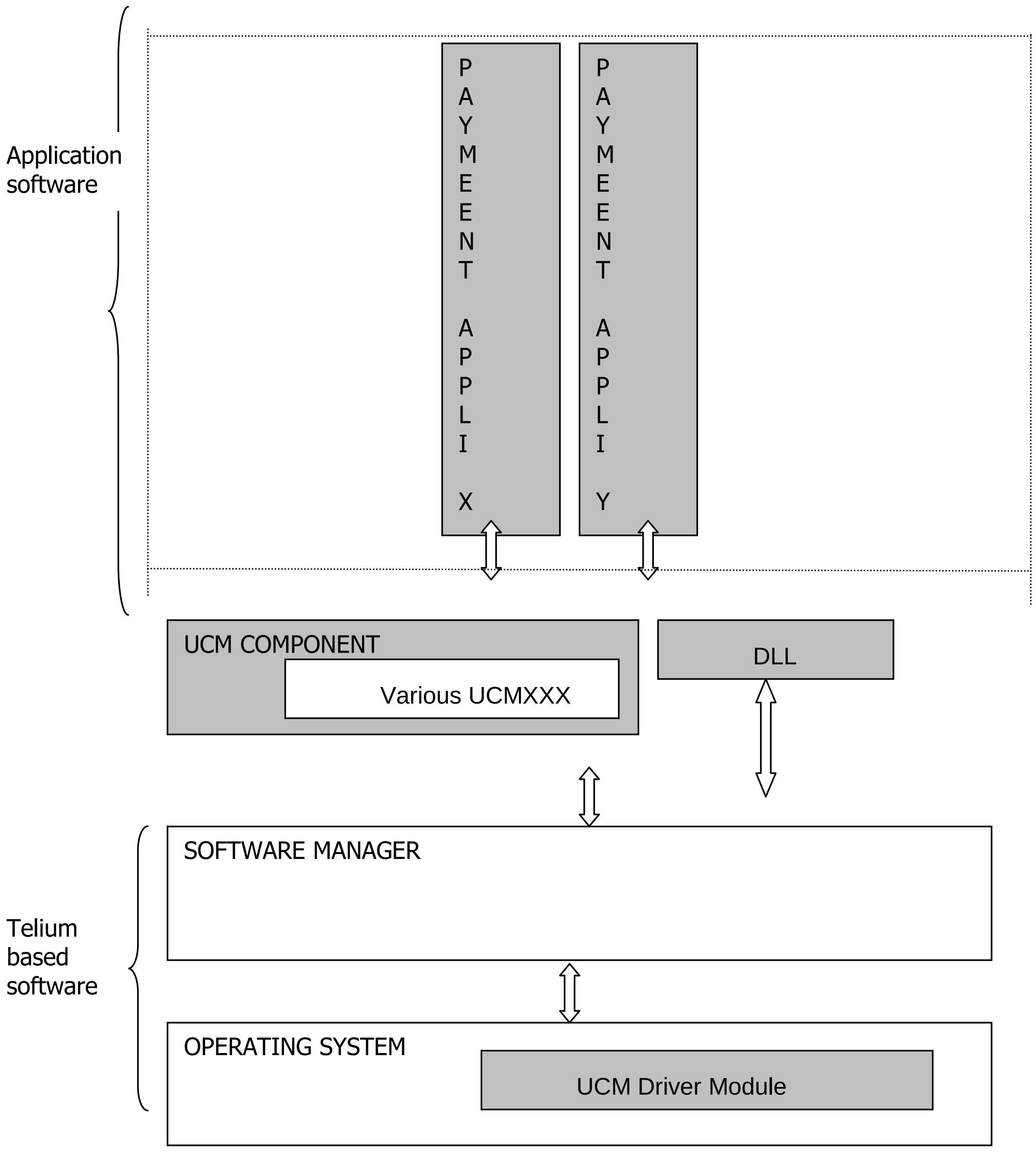
4. TERMINOLOGY

UCM	= Universal Communication Module
LC	= Card reader
DA	= Automatic vending machine
PSC	= Standard Communication Protocol
IAC	= Interface Application Code

ICC = Integrated Card Circuit: Card reader.

5. INTRODUCTION TO THE SOFTWARE

5.1 SOFTWARE ARCHITECTURE



Shaded area: UCM platform-specific software.

5.2 OPERATING SYSTEM

The operating system consists of the TELIUM generic shell and the UCM system module.

5.3 MANAGER

The manager is the same as in the other terminals of the TELIUM range. The IAM library is essential.

If manager is not initialized, UCM displayed message “INITIALIZE TELIUM MANAGER”.

But after UCMC starting, if a param.ini file is present in the HOST directory and manager manage this file, “TELIUM MANAGER INIT DONE” is displayed during 2 seconds and UCM restart with exit 0x9471.

5.4 UCM COMPONENT

This is the application the UCM uses for setting up the parameters, managing the peripherals (keypad, display, printer, swipe, leds, buzzer, cam, sam), communicating with the host and handling the payment kinematics. The purpose is to keep the applications and the manager from handling the connected equipment and the dialogs with the payment equipment. It is mandatory to use the existing UCM API to access devices. Lan, Bluetooth, GPRS, com, USBKey, memory card … are managed by standard functions.

The UCM component defines all the external (managed by the UCM component) and internal (managed by the system) peripherals (display, keypad, printer)... using the parameter set-up file of the UCM component.

According to the parameter set-up, it runs the host protocol DLL (1042, MDB, EXE, or other) and adapts to the payment kinematics. It provides the applications with all the services needed for using the peripherals, e.g. UCM_Display().

It uses the system services (if any) to control the peripherals (e.g. display PPR30); otherwise, it runs the DLL of the peripherals that have been configured but are not managed by the system.

It will be possible to use a protocol DLL developed by a VAR taking account of the entry points defined in the other sections (initialization, launching / stopping of dialog task, management of messages, entry points).

The UCM component continuously sends status changes to the Protocol DLL (card present, unavailable, remote parameter set-up/remote collection, keypad/reader out-of-order).

According to the parameter set-up, it's possible to use UCM Component only to access peripherals. In this case the manager manages the payment flow calling standard entry function. The payment transaction flow described in this document is not used.

5.5 UCMXXX DLL

These DLLs are installed by the UCM component according to the parameter set-up. The purpose of these DLLs is to control peripherals such as a host, a DA (automatic vending machine), a coin meter, etc.

The DLLs to be loaded are defined in the parameter set-up file.

Since Pack UCM v0205, it possible for a application to initialize banking Host DLL and to send and receive 1042 banking message.

5.6 UCMSTART DLL

This DLL is loaded by the software manager before the applications are run.

It is intended to load the UCM parameter setting drivers.

It contains the UCM error file management services.

It contains the UCM parameter control services.

It can restart the UCM if required by the new parameters.

It manages maintenance mode.

5.7 PAYMENT APPLICATIONS

The payment applications automatically adapt to the peripherals configuration by calling the UCM component services for display, printing, data input, CAM, SAM, etc.

They can identify the list of available peripherals and generate an error at initialization if a major peripheral in the list is missing (printer, display unit, keypad, SAM, etc.).

5.8 DRIVERS

The UCM-specific drivers are launched by the UCMSTART DLL. These drivers are defined in the UCM parameter settings.

6. SPECIAL FEATURES

6.1 CARD DETECTION

When a payment application is at an entry point, it will use the iLIBUCM_Icc_Status () function to inform the UCM component which will send the information to the host protocol.

The UCM periodically calls the manager which indicates that the card is present.

6.2 LED MANAGEMENT

All 3 LEDs are managed by the system. The system will provide a driver running the " on", "off" and " flash " modes on each LED.

LED can be exist on 2 devices.

6.3 LLT DETECTION

LLT mode selected on startup using the pushbutton.

Managed by the system.

6.4 MAINTENANCE MODE-EXPLOITATION MODE

There is a concept of maintenance / exploitation mode for unattended.

Exploitation mode is for customer using. It's for payment. Use only UCMC API during this mode. In this mode it's impossible to access more_function(). DLL Security is configured for cipher and pin according to device.

Maintenance mode is for more_function() access. For CAD30 you need CAD30 Tool. For IUN you need using push button. In this mode display and keyboard driver are directly accessible. UCMC APIs are compatibles. DLL Security is configured for cipher only to IUN according to device.

6.5 MODIFICATION OF UCM PARAMETERS

The parameters are modified by file. This file are signed and provided. It is possible to download this file in LTT mode (“HOST” disk). It is possible to download this file in USB key with manager menu. It is possible to choice from several parameters with UCMC menu (parameters files must be in embedded = download this file in “SWAP” disk).

6.6 MODIFICATION OF SOFTWARE MANAGER PARAMETERS

The "Pinpad" manager parameter must be set to "no".

The "cash register" manager parameter must be set to "no" to avoid using a serial port resource.

6.7 MODIFICATION OF APPLICATION SOFTWARE SETTINGS

Use the same method as for the software manager and UCM component (TMS Toolkit).

6.8 HEADER

There is no header on a UCM platform except with maintenance mode. It is disabled by the software manager.

6.9 IDLE SCREEN

Parameters for the idle screen are set in the UCM component. This idle screen can be modified by the applications through the iLIBUCM_Display_New_Idle_Msg() service.

6.10 PROTOCOLS

Each protocol (1042, EXE, MDB) has to be developed in a DLL. The DLLs use the same memory addresses because they cannot be used at the same time.

The name of the DLL to be loaded is in the parameter set-up. The DLL shall follow the rules for the entry points with the Host; See document “vending kinematics conception” or “banking kinematics conception”

6.11 PERIPHERALS MANAGEMENT

The peripherals list is made up by setting the UCM component parameters.

The UCM component relies on the system services to control the peripherals using a driver.

6.12 MODEM

These parameters are handled in the manager. The modem parameter in the host is defined in the UCM component parameter set-up.

The manager will interrogate the UCM component to find out if the modem is on the host.

When the modem is on the host, the manager will call UCM component entry points to request connections, transmissions, receptions, modem status requests, disconnections, etc.

6.13 RESERVED ENTRY POINT

IS_CARD_SPECIFIC, IDLE_MESSAGE, KEYBOARD_EVENT, FALL_BACK, CARD_OUTSIDE and CARD_REMOVAL manager services are prohibited in applications

6.14 ERROR AND WARNING MANAGEMENT

The services for managing the errors and warnings detected by the UCM component are loaded in the UCMSTART DLL.

The error file may be consulted in maintenance mode on the display unit (if available) or on the printer (if available).

If necessary, CAD30 or IUN are is restarted up to 3 times to find an operational state expect in maintenance mode.

6.15 SLEEP POWER MANAGEMENT

Light and deep sleep is possible with CAD30 UCR.

Deep sleep is possible with iUP250 or IUC180 or IUC180B (with / without IUR250 and IUC150). The awakening is performed by green key with IUP250, card insertion with IUR250 (need COM specialized pin cable), external key with IUC180 or IUC180B or IUC150. See API iLIBUCM_Device_Cmd(UCMDEVICE_CMD_GET_POWER_MNG, …)

7. TRANSACTION FLOW

7.1 INITIALIZATION

STATUS, EVENT	Manager	UCMSTART DLL	UCM Component	UCMPROT DLL	Controller or Host
Power on	Loading Initialization	Reading new parameter set-up file. Initialization. Driver loading.			
	Starting		Reading of parameter set-up file Loading of various DLLs If device error, reset up to 3 times.	iUcmHostDll_Init	Dialog in process
	Host File Management		Delete new parameter set-up file		
	Idle State		Idle_message() Idle screen of parameter settings		
Keyboard key pressed			Keyboard_event() Lockout the key to avoid more_function()		

7.2 PARAMETER SETTING BY LLT

STATUS, EVENT	Manager	UCM Component	Controller or Host
Pushbutton LLT tool sends a file LLT output			
Reset	Initialization $\longrightarrow$	Parameter updating Finding out whether the file comes from system disk (not taken into account) or from HOST disk (erased afterwards).	

The parameter set-up file has to be stored in the HOST disk.

7.3 PAYMENT WITH HOST (BANKING CONFIGURATION)

STATUS	Manager	Payment application	UCM Component	Host
1> Idle			Response = Available + no card	← Status request
2> Card insertion	→	→	Is_Card_specifique()	
3>Idle			Response = Available + card present	← Status request
4>Solvency request	Standard process of application selection	←	Payment application selection	Solvency request ←
5>Application selection	→ → →	Is_Card_For_You_after AID_Emv() Is_Card_for_you() Acceptation		
6>Debit	→	Debit_Emv() → Card processing ←	iLIBUCM_Pay_Ready_For_Debit ()	
			Response = Available + card present	← Status request
		End of card processing → → DEBIT_EMV() output	iLIBUCM_Pay_Result_Debit (solvency) → ILIBUCM_Pay_End()	Cr solvency

7>Idle	Time_function (1 min)			
8> Record		←	Call application who made solvency ←	Recording request ←
	→	Debit_Emv() → Processing ←	iLIBUCM_Pay_Ready_For_Debit ()	
			Response = Available + card present	Status request ←
		End of card processing → → DEBIT_EMV() output	iLIBUCM_Pay_Result_Debit(recording) → ILIBUCM_Pay_End()	Cr recording
9> Idle			Available + card present	Status request ←
10> Card removal	→	→	card_outside () card present = 0	
11> Idle			Response = Available + no card	Status request ←

Note: In idle status, the manager may call the Time_function() entry point and make the UCM unavailable in PSC protocol. This is normal in MPC10S.

In idle status, after a timeout, the manager may request a card removal.

card_outside () is a manager service that can detect card removal in idle status.

7.4 PAYMENT WITH AUTOMATIC VENDING MACHINE

STATUS	Manager	Payment application	UCM Component	Protocol DLL	DA
1> Idle					Polling
2> Card insertion			Is_Card_specifique()		Polling
5>Application selection		Is_Card_For_You_After() Acceptation			Polling
4>Debit		Debit_Non_Emv() Card processing Send e-purse balance Displays balance	iLIBUCM_Pay_Ready_For_Debit () (Read e-purse) iLIBUCM_Pay_Host_Cmd (epurse balance) iLIBUCM_Display_Message (Credit)	DLL_Result_Debit	Polling Polling Credit
5> Waiting for selection			iLIBUCM_Device_Ttestall (SELECTION + CARD_RETRIEVED + timeout) Waiting	DLL_Received()	Polling
6> Selection			Exit from waiting		Selection
7> Debit		Processing continued	iLIBUCM_Pay_ Ready_For_Debit (debit e-purse)		Polling

		Debit processing	iLIBUCM_Pay_Result_Debit (Vending in process)	DLL_Result_Debit →	Ok for vending
8> Waiting for end of distribution		→	iLIBUCM_Device_Ttestall (END OF VENDING + timeout) Waiting ↔	→ DLL_Received()	← Polling
9> End of distribution			← Exit from waiting()		← Vending OK
10> End of transaction		Processing continued → Record transaction → Ask end of transaction →	iLIBUCM_Pay_Ready_for_Debit (vending result) iLIBUCM_Pay_Result_Debit (Record) iLIBUCM_ Pay_Host_Cmd (End_Debit)	→	← Polling
11> Card removal request		If card present waiting for card removal →	iLIBUCM_Display_No_Wait (Remove the card) iLIBUCM_Device_Ttestall(ICC + timeout)	→	← Polling
10> Card removal		Next processing →	iLIBUCM_ Pay_Host_Cmd (End_Debit)	→	← Polling
13> Idle		Exit Debit_Non_Emv()		→	← Polling

Note: Upon reception of the selection number, depending on the application number and on the information on the card, it will be necessary to find out the item price in the price list. This price list may be managed in a specific application (such as SELECTA).

For setting the UCM component parameters in multivending, the payment application, even if it receives multivending data, may decide to interrupt the transaction by informing the UCM component via the iLIBUCM_Pay_Result_Debit () function.

7.5 PAYMENT WITH AUTOMATIC VENDING MACHINE AND ICC MANAGED BY APPLICATION

STATUS	Manager	Payment application	UCM Component	Protocol DLL	DA
1> Idle				→	← Polling
2> Card insertion	→	Card detected by application iLIBUCM_Pay_Host_Cmd (UCMHOSTLIB_MSG_DEM_PAY)		→	Polling ←
3>Application selection		←	Application selection request	→	Polling ←
Same steps (4 to 13) as chapter "Payment with automatic vending machine)					

The application is in charge to detect card insertion and managed card input / output.

7.6 STATUS REQUEST

Manager	Payment application	UCM component	HOST DLL	Host
GetGeneralStatus		Status request		
			iUCMHOST_Set_Status() Store new state	
			Standalone response	Status request

7.7 MODEM ON HOST TRANSMISSION

Manager	Payment application	UCM Component	HOST DLL	Host
	Call			
Connection request		Returns " modem on host "		
Connection to Host		Blocking command		
			iUcmHostDll_Modem_Cnx()	

			Connection →	
			← Cr Connection	Connection performed
	←	Cr Connection		
Transmission/Reception handle "modem"				

			Driver Modem calls
			iUcmHostDll_Modem_Write()
			iUcmHostDll_Modem_Read() →
			iUcmHostDll_Modem_Status()	

Disconnection →	→	Blocking command		
		→	iUcmHostDll_Modem_Dcnx()	
			Disconnection →	
			Cr Disconnection ←	Disconnected
	←	Cr Disconnection		
Disconnection				

8. UCM APPLICATION SETTINGS

8.1 INTRODUCTION

Access to some parameters can only be achieved by changing the parameter set-up file.

This file can be loaded by LLT in the "host" disk using the copy process.

To update the UCM component, it is necessary to copy the file in the UCM "host" disk.

NAME = ucmxxxxx.par in ASCII with defined wordings to identify the fields for CAD30

NAME = ucmxxxxx.pas in ASCII and signed with defined wordings to identify the fields for IUN

Xxxxx is a file version.

The file copied in the "host" disk may be incomplete (it must contain at least one tag). This way, it is possible to load just one part of the parameters.

8.2 PARAMETER SET-UP FILE

The parameter set-up file contains tags each defining a topic.

Files are delivered with UCM device configuration.

9. UCM COMPONENT BUZZER IAC

9.1 INTRODUCTION

The "user" buzzers can either be located on a reader, a display, a keyboard or the host.

The commands are: ON, OFF.

It is possible to use three tones and delay in units of seconds or 10ms

9.2 ILIBUCM_BUZZER_CMD ( )

Description: Used to control the buzzers.

Prototype: extern int iLIBUCM_BUZZER_Cmd(T_UCMC_IAC_BUZ *pdata_p) ;

pdata_p = see T_UCMC_IAC_BUZ in UcmcLib.h and samples.

Return: FCT_OK or a negative value in the event of an error.

9.3 ILIBUCM_BUZZER_EXIST ( )

Description: Sends data on the type defined in parameter set-up.

Prototype: iLIBUCM_BUZZER_Exist (void)

Return: Value of the type defined in parameter set-up, in decimals.

9.4 ILIBUCM_BUZZER_INIT ( )

Description: Initializes the buzzer. Only used by the UCM.

9.5 ILIBUCM_BUZZER_STATUS ( )

Description: Sends data on the buzzer status, even without opening the peripheral. Reserved for future use.

9.6 ILIBUCM_BUZER_CMDCH ( )

Description: Used to control the buzzers.

Prototype: extern int iLIBUCM_BUZZER_CmdCh(unsigned char ucChannel_p , T_UCMC_IAC_BUZ *pdata_p) ;

ucChannel_p: Buzzer channel. Use UCMC_BUZZER by default

pdata_p = see see T_UCMC_IAC_BUZ in UcmcLib.h and samples.

Return: FCT_OK or a negative value in the event of an error.

9.7 ILIBUCM_BUZZER_EXISTCH ( )

Description: Sends data on the type defined in parameter set-up.

Prototype: iLIBUCM_BUZZER_ExistCh (unsigned char ucChannel_p)

ucChannel_p: Buzzer channel. Use UCMC_BUZZER by default

Return: Value of the type defined in parameter set-up, in decimals.

9.8 ILIBUCM_BUZZER_EXIST ( )

Description: Sends data on the buzzer layout

Prototype: iLIBUCM_BUZZER_Exist (unsigned char ucChannel_p)

ucChannel_p: Buzzer channel. Use UCMC_BUZZER by default

Return: Value of the type defined in parameter set-up, in decimals.

9.9 ILIBUCM_BUZZER_INITCH ( )

Description: Initializes the buzzer. Only used by the UCM.

9.10 ILIBUCM_BUZZER_STATUSCH ( )

Description: Sends data on the buzzer status, even without opening the peripheral. Reserved for future use.

10. MULTIDEVICE

10.1 INTRODUCTION

10.2 ILIBUCM_DEVICE_CMD ( )

Description: Device command.

Used to initialized Security DLL if necessary: UCMDEVICE_CMD_SETDLLSECU command or UCMDEVICE_CMD_SETDLLSECU or UCMDEVICE_CMD_GETDLLSECU structure.

Used for deep management. UCMDEVICE_CMD_SET_POWER_MNG. See samples and “Energy Management on CAD30UCR” sdk_telium_xxxx.chm. If power managed is set by manager menu this command returns error.

Prototype: iLIBUCM_Device_Cmd(unsigned short usCmd_p, void *pData_p, int *piLgData_p, int *piRet_p)

usCmd_p: See TE_UCM_DEVICE in ucmtelium.h

pvData_p : See structure T_UCM_DEVICE

piLgData_p : See structure T_UCM_DEVICE

piRet : See structure T_UCM_DEVICE

Return: FCT_OK or a negative value in the event of an error.

10.3 ILIBUCM_DEVICE_CONFIG ( )

Description: Get configuration maintenance or operational of selected peripheral.

Prototype: iLIBUCM_Device_Config(unsigned char ucDevice_p, T_UCM_DEVICE *psDevice_p)

ucDevice_p: See TE_UCM_DEVICE in ucmtelium.h

psDevice_p : See structure T_UCM_DEVICE

Return: FCT_OK or a negative value in the event of an error.

10.4 ILIBUCM_DEVICE_CONFIGALL ( )

Description: Get maintenance or operational complete configuration.

Prototype: iLIBUCM_Device_ConfigAll(T_UCM_PARAM *psParam_p)

psParam _p : T_UCM_PARAM.

Return: FCT_OK or a negative value in the event of an error.

10.5 ILIBUCM_DEVICE_CONFIGALL_OPE( )

Description: Get exploitation mode complete configuration.

Prototype: iLIBUCM_Device_ConfigAll(T_UCM_PARAM *psParam_p)

psParam _p : T_UCM_PARAM.

Return: FCT_OK or a negative value in the event of an error.

10.6 ILIBUCM_DEVICE_CONFIG_OPE ( )

Description: Get exploitation mode configuration of selected peripheral.

Prototype: iLIBUCM_Device_Config_OPE(unsigned char ucDevice_p, T_UCM_DEVICE *psDevice_p)

ucDevice_p: See TE_UCM_DEVICE in ucmtelium.h

psDevice_p : See structure T_UCM_DEVICE

Return: FCT_OK or a negative value in the event of an error.

10.7 ILIBUCM_DEVICE_EVENT_SERVICE_SET ( )

Description: Provide event service of selected event.

Need application implementation of function UCMHOST_FCTAPP_EVEN, UCMAPPLIMODULE_FCTAPP module of service UCMAPPLIMODULE_FCTAPP.

Prototype: iLIBUCM_Device_Event_Service_Set(T_UCMC_IAC_EVENT *psEvent_p)

psEvent _p : See structure T_UCMC_IAC_EVENT

Return: FCT_OK or a negative value in the event of an error.

10.8 ILIBUCM_DEVICE_GETINFO ( )

Description: Give system information about UCMC or UCMSTART.

Prototype: int iLIBUCM_Device_GetInfo(unsigned char ucDevice_p , object_info *pinfos_p) ;

ucDevice_p : UCM_DEVICE_UCMC ou UCM_DEVICE_UCMSTART.

pinfos_p : See Telium object_info.

Return: FCT_OK or a negative value in the event of an error.

10.9 ILIBUCM_DEVICE_GETSTATUS ( )

Description: Give information about parameters of device.

Prototype: int iLIBUCM_Device_GetStatus(unsigned char ucDevice_p , T_UCM_DEVICE_STATUS *psDevice_p) ;

ucDevice_p : TE_UCM_DEVICE.

PsDevice_p : Device structure.

Return: FCT_OK or a negative value in the event of an error.

10.10 ILIBUCM_DEVICE_TTESTALL ( )

Description: Waits for an event on the requested peripherals.

Prototype: int iLIBUCM_Device_Ttestall(unsigned short *pusWhat, unsigned short usDelay_p)

pusWhat: Composition of devices like UCMMULTI_TTESTALL_ICC | UCMMULTI_TTESTALL_HOST.

usDelay_p: Timeout within 10ms

Return: pusWhat takes the value of the peripheral that triggered the event.

UCMTTESTALL_CR_OK_EVENT or UCMTTESTALL_CR_NO_DEVICE or

UCMTTESTALL_CR_NO_DECLARED or UCMTTESTALL_CR_NO_EVENT or UCMTTESTALL_CR_TIME_OUT.

10.11 ILIBUCM_DEVICE_TTESTALL_STOP ( )

Description: Stop waiting.

Prototype: int iLIBUCM_Device_Ttestall_Stop(void)

Return: 0 no waiting or 1 if waiting be stop as soon as possible.

10.12 ILIBUCM_DEVICE_UCM_STATUS ( )

Description: Sends data on the status of all peripherals.

Prototype: int iLIBUCM_Device_Ucm_Status(T_LIBUCM_DEVICE_STATUS *pUCMState_p)
PUCMState_p : See T_LIBUCM_DEVICE_STATUS in ucmclib.h

Return: FCT_OK or a negative value in the event of an error.

10.13 ILIBUCM_SYSTEMFIOCTL ( )

Description: Same system function. Informations are update with UCMC parameters.

Prototype: int iLIBUCM_SystemFioctl(int iFioCmd_p , void *vpData_p, int iLgData_p) ;
iFioCmd_p: See SystemFioctl ().
vpData_p: See SystemFioctl ().
iLgData_p: Length of parameter input by vpData.

Return: FCT_OK or a negative value in the event of an error.

11. UCM COMPONENT DISPLAY IAC

11.1 INTRODUCTION

The following entry points are IACs that can be used by the payment applications.
They are described in the ucmclib.h file and require the UCMC.LIB library.
Two display units can be managed (see ucChannel_p).

The keywords allowed in the messages are:
"\1B" to clear the display
"\n" to go to the next line
\yyyy to display year
\yy to display year in 2 digits.
\mm to display month
\dd to display day
\hh to display hour
\ii to display minutes
\ss to display seconds

11.2 ILIBUCM_DISPLAY_BACKLIGHT_COLOR ( )

Description: Command display backlight color.
Use only on backlight color display.

Prototype: int iLIBUCM_Display_Backlight_Color (unsigned char ucChannel_p, T_UCMC_DISPLAY_BACKLIGHT_COLOR *pBck_color_p) ;
ucChannel_p: Display channel. Use UCMC_DISPLAY by default.
pBck_color_p: structure T_UCMC_DISPLAY_BACKLIGHT_COLOR

Example:

```
  sBlkColor.ucColorDefined = UCMCDISPLAY_BKL_COLOR_BLUE;
  sBlkColor.ucOption = 0; /* 0 =Fix       1 to 5 =Flash speed to slow */
  sBlkColor.usBlue = 0;
  sBlkColor.usGreen = 0;
  sBlkColor.usRed = 0;
```

if sBlkColor.ucColorDefined=UCMCDISPLAY_BKL_COLOR_NO_DEFINED, it'is possible to refined color usBlue, usRed with same value as RETRO_ECLAIRAGE_C30_RGB_FIOCTL_T Struct.

Return: 0=OK or negative error.

11.3 ILIBUCM_DISPLAY_BACKLIGHT_COLOR_EXIST ( )

Description: Returns display backlight color existence.

Use only on backlight color display.

Prototype: int iLIBUCM_Display_Backlight_Color_Exist (unsigned char ucChannel_p) ;

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

Return: Value of the type defined in parameter set-up (example PARAM_TYPE_IUP250) or negative error.

11.4 ILIBUCM_DISPLAY_CLEAR( )

Description: Clears all lines

Prototype: int iLIBUCM_Display_Clear(unsigned char ucChannel_p)

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

Return: FCT_OK or negative error.

11.5 ILIBUCM_DISPLAY_CLEAR_LINE( )

Description: Clears a line. No effect for all type of display.

Prototype: int iLIBUCM_Display_Clear_Line(unsigned char ucChannel_p)

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

Return: FCT_OK or negative error.

11.6 ILIBUCM_DISPLAY_CLOSE ( )

Description: Closes display peripheral

No effect if the display is on host.

Prototype: int iLIBUCM_Display_Close(unsigned char ucChannel_p)

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

Return: FCT_OK or negative error.

11.7 ILIBUCM_DISPLAY_CMD ( )

Description: Display peripheral command. Used by the UCM. Reserved for future use.

11.8 ILIBUCM_DISPLAY_EXIST ( )

Description: Returns device existence..

Prototype: int iLIBUCM_Display_Exist(unsigned char ucChannel_p) ;

Return: Value of the type defined in parameter set-up (example PARAM_TYPE_UPP) or negative error.

11.9 ILIBUCM_DISPLAY_GET_MSG ( )

Description: Gives the message used by the UCM component.

Prototype: int iLIBUCM_Display_Get_Msg(unsigned char ucChannel_p, unsigned short usMsgNumber_p, char *cMessage_p) ;

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

usMsgNumber_p: message number to be recovered (0 = idle message).

cMessage_p: message corresponding to the number above.

Return: 0=None 1=Connected 3=on Host. See parameter set-up.

11.10 ILIBUCM_DISPLAY_GRAPHIC_START( )

Description: Start the graphic display mode.

Use only on graphic display device. No effect in maintenance mode.

Function allows the use of standard graphics. Use iLIBUCM_Display_Graphic_Stop after.

Prototype: int iLIBUCM_Display_Graphic_Start (unsigned char ucChannel_p) ;

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

Return: FCT_OK or a negative value in the event of an error.

The vending font is defined in file fontlib.h. Tuo use Vending font, add macros:

#define *dVENDING*NORMAL_   “dVENDNORMAL”

Example:

DisplayMSG(0,0,iNoMessage_p, *OFF*, dVENDINGNORMAL_, _FIXED_WIDTH_) ;

11.11 ILIBUCM_DISPLAY_GRAPHIC_STOP( )

Description: Stop the graphic display mode.

Use only on graphic display device. No effect in maintenance mode.

Prototype: int iLIBUCM_Display_Graphic_Stop (unsigned char ucChannel_p) ;

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

Return: FCT_OK or a negative value in the event of an error.

11.12 ILIBUCM_DISPLAY_IDLE_EVENT ( )

Description: Provide event service to display idle message in application.

Prototype: int iLIBUCM_Display_Idle_Event (unsigned char ucChannel_p , unsigned char ucOnOff_p) ;

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

ucOnOff _p: 1=Application is called 0=Stop call application .

Need application implementation of function UCMHOST_FCTAPP_IDLE_EVENT, UCMAPPLIMODULE_FCTAPP module of service UCMAPPLIMODULE_FCTAPP.

Par.sDisplay.cBuf of structure T_UCMC_IAC_SERVICE gives idle message normally displayed.

Return: FCT_OK or negative error.

11.13 ILIBUCM_DISPLAY_INIT( )

Description: Initializes display peripheral. Only used by the UCM module.

11.14 ILIBUCM_DISPLAY_IS_CONNECT ( )

Description: Informs if the peripheral is connect.

Prototype: int iLIBUCM_Display_Is_Connect(unsigned char ucChannel_p) ;

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

Return: 0=Connect. Negative=Error

11.15 ILIBUCM_DISPLAY_IS_OPEN ( )

Description: Informs that the peripheral is open.

If the display is on host, sends back "open".

Prototype: int iLIBUCM_Display_Is_Open(unsigned char ucChannel_p) ;

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

Return: 0=Closed. 1=Open. Negative=Error

11.16 ILIBUCM_DISPLAY_MESSAGE ( )

Description: Manages the display of a character string.

If the peripheral is not open, it opens and then closes it.

Uses the options: UCMDISPLAY_OPEN_IF_NOT | UCMDISPLAY_CLOSE_IF_OPEN

Prototype: int iLIBUCM_Display_Message (unsigned char ucChannel_p, char *cMessage_p , unsigned short usTimeout_p)

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

*cMessage_p: Message to be displayed.

usTimeout_p : Timeout within 10 ms.

Return: FCT_OK or a negative value in the event of an error.

11.17 ILIBUCM_DISPLAY_NEWLINE ( )

Description: goes to a next line (carriage return + line feed).

Prototype: int iLIBUCM_Display_NewLine(unsigned char ucChannel_p) ;

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

Return: FCT_OK or a negative value in the event of an error.

11.18 ILIBUCM_DISPLAY_NEW_IDLE_MSG ( )

Description: Enables to display a new idle message different from the parameter set-up message.

Prototype: int iLIBUCM_Display_New_Idle_Msg(unsigned char ucChannel_p, char *cMessage_p)

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

*cMessage_p: Message to be displayed.

Return: FCT_OK or a negative value in the event of an error.

11.19 ILIBUCM_DISPLAY_NO_WAIT ( )

Description: Manages the display of a character string during a timeout and allows the next display after the timeout. If the timeout has not elapsed, only the display functions iLIBUCM_Display_No_Wait() or iLIBUCM_Display_No_Wait2MSG () are enabled.

The idle massage is not displayed before the timeout has elapsed.

If the peripheral is not open, it opens and then closes it.

Uses the options: UCMDISPLAY_OPEN_IF_NOT | UCMDISPLAY_CLOSE_IF_OPEN

Prototype: int iLIBUCM_Display_No_Wait(unsigned char ucChannel_p, char *cMessage_p , unsigned short usTimeout_p)

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

*cMessage_p: Message to be displayed.

usTimeout_p : Timeout within 10 ms.

Return: FCT_OK or a negative value in the event of an error.

11.20 ILIBUCM_DISPLAY_NO_WAIT2MSG ( )

Description: Manages the display of 2 messages during timeouts and enables the next display after the timeouts if the non-stop option is not selected. If the timeout has not elapsed, only the display functions iLIBUCM_Display_No_Wait() or iLIBUCM_Display_No_Wait() are enabled.

The Idle message is not displayed before the timeouts have elapsed.

If the peripheral is not open, it opens and then closes it.

Uses the options: UCMDISPLAY_OPEN_IF_NOT | UCMDISPLAY_CLOSE_IF_OPEN

Prototype: int iLIBUCM_Display_No_Wait2Msg(unsigned char ucChannel_p,
unsigned short usContinus_p,
char *cMessage1_p, unsigned short usTimeout1_p,
char *cMessage2_p, unsigned short usTimeout2_p)

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

usContinus_p: 0= No effect. 1= Continuous switching from message 1 to message 2.

*cMessage1_p: Message to be displayed.

UsTimeout1_p : Timeout within 10 ms.

*cMessage2_p: Message to be displayed.

UsTimeout2_p : Timeout within 10 ms.

Return: FCT_OK or a negative value in the event of an error.

11.21 ILIBUCM_DISPLAY_OPEN ( )

Description: Opens the peripheral in the required mode.

Uses the options: “W” “W+” “A”

No effect if the display is on host.

Prototype: int iLIBUCM_Display_Open(unsigned char ucChannel_p, char *cOption_p)

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

*cOption_p: Options.

Return: FCT_OK or negative error.

11.22 ILIBUCM_DISPLAY_OPTION ( )

Description: Manages the display of a character string with option.

Prototype: int iLIBUCM_Display_Option(unsigned char ucChannel_p, char *cMessage_p , unsigned short usOption, unsigned short usTimeout_p) ;

ucChannel_p: Display channel. Use UCMC_DISPLAY by default.

usOption: use option as UCMDISPLAY_OPEN_IF_NOT | UCMDISPLAY_CLOSE_IF_OPEN

*cMessage_p: Message to be displayed.

usTimeout_p : Timeout within 10 ms.

Return: FCT_OK or a negative value in the event of an error.

12. CAM & SAM IAC OF UCM COMPONENT

12.1 INTRODUCTION

These entry points manage SAMs, CAM and magnetic stripes, depending on the requested channel:

UCMC_ICC chip channel, by default. There are 2 possible channels.

UCMC_SAM SAM channel, by default. There are 5 possible channels.

UCMC_ISO2 magnetic stripe channel located on an ICC channel if the reader is a mixed reader.

12.2 ILIBUCM_ICC_BACKLIGHT_COLOR ( )

Description: Command reader backlight color.

Prototype: int iLIBUCM_Icc_Backlight_Color (unsigned char ucChannel_p, T_UCMC_DISPLAY_BACKLIGHT_COLOR *pBck_color_p) ;

ucChannel_p: Display channel. Use UCMC_ICC by default.

pBck_color_p: structure T_UCMC_DISPLAY_BACKLIGHT_COLOR

Example: See iLIBUCM_Display_Backlight_Color() API.

Return: 0=OK or negative error.

12.3 ILIBUCM_ICC_BACKLIGHT_COLOR_EXIST ( )

Description: Returns reader backlight color existence.

Prototype: int iLIBUCM_Icc_Backlight_Color_Exist (unsigned char ucChannel_p) ;

ucChannel_p: Display channel. Use UCMC_ICC by default.

Return: Value of the type defined in parameter set-up (example PARAM_TYPE_IUR250) or negative error.

12.4 ILIBUCM_ICC_CLOSE ( )

Description: Closes the peripheral.

Prototype: int iLIBUCM_Icc_Close (unsigned char ucChannel_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM or UCMC_ISO2

Return: FCT_OK or a negative value in the event of an error.

12.5 ILIBUCM_ICC_CMD ( )

Description: Icc peripheral command.

Description: Sends reader command.

Prototype: iLIBUCM_Icc_Cmd(T_UCMC_IAC_ICC_CMD *pdata_p) ;

pdata_p : pointer of the structure described in the ucmclib.h file.

See sample UCMCICC_CMD_LOCK (lever lock of IUR250 only).

Return: : FCT_OK or a negative value in the event of an error.

12.6 ILIBUCM_ICC_EMVAPDU ( )

Description: Used to send a command in EMV format to the card.

Depending on the options, if the peripheral is not open, the command opens and then closes it.

Prototype: int iLIBUCM_Icc_EmvApdu(unsigned char ucChannel_p, unsigned short usOption_p, T_APDU *pC_apdu_p, T_APDU *pR_apdu_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM or UCMC_ISO2

usOption_p: Option defined in UCMCLIB.H file such as UCMICC_OPEN_ALWAYS.

pC_apdu_p : APDU command

pR_apdu_p : Response to the command

Return: FCT_OK or a negative value in the event of an error. Same returns as EMV_apdu() if not negative.

12.7 ILIBUCM_ICC_EXIST ( )

Description: Sends data on the type defined in parameter set-up.

Prototype: int iLIBUCM_Icc_Exist (unsigned char ucChannel_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM or UCM_ISO1 or UCMC_ISO2 or UCMC_ISO3

Return: Value of the type defined in parameter set-up, in decimals or negative value if reader error.

12.8 ILIBUCM_ICC_F_SYNC_FCT ( )

Description: Command for synchronous card. See f_sync_fct() of SDK.

Prototype: int iLIBUCM_Icc_F_Sync_Fct (unsigned char ucChannel_p, unsigned short usOption_p, unsigned char ucFunction_p, unsigned char ucParam_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM or UCMC_ISO2

usOption_p: Option defined in UCMCLIB.H file as UCMICC_OPEN_ALWAYS.

ucFunction_p: see SDK

ucParam_p : see SDK

Return: FCT_OK or a negative value in the event of an error.

12.9 ILIBUCM_ICC_INIT()

Description: Initializes the peripheral.

Reserved.

12.10 ILIBUCM_ICC_INPUT ( )

Description: Input command for T0 type chip card.

Depending on the options, if the peripheral is not open, the command opens and then closes it.

Prototype: int iLIBUCM_Icc_Input(unsigned char ucChannel, unsigned short usOption_p, COMMAND_CAM *pCmd_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM or UCMC_ISO2

usOption_p: Option defined in UCMCLIB.H file as UCMICC_OPEN_ALWAYS.

pCmd_p: Command defined in SDK

Return: FCT_OK or a negative value in the event of an error. Same value as standard function input_command() if positif.

12.11 ILIBUCM_ICC_IS_OPEN ( )

Description: Informs that the peripheral is open.

Sends "open" if the printer is on host.

Prototype: int iLIBUCM_Icc_Is_Open(unsigned char ucChannel_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM or UCMC_ISO2

Return: 0=Closed. 1=Open. Negative=Error.

12.12 ILIBUCM_ICC_IS_CONNECT ( )

Description: Informs if the peripheral is connect.

Prototype: int iLIBUCM_Icc_Is_Connect(unsigned char ucChannel_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM or UCMC_ISO2

Return: 0=Connect. Negative=Error.

12.13 ILIBUCM_ICC_OPEN ( )

Description: Opens the peripheral

Prototype: iLIBUCM_Icc_Open(unsigned char ucChannel, char *cOption_p) ;

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM or UCMC_ISO2

cOption_p : "W" or "RW"…

Return: FCT_OK or a negative value in the event of an error.

12.14 ILIBUCM_ICC_OUTPUT ( )

Description: Output command for T0 type chip card

Depending on the options, if the peripheral is not open, the command opens and then closes it.

Depending on the set-up, calls on output_command(). See SDK

Prototype: iLIBUCM_Icc_Output(unsigned char ucChannel, unsigned short usOption_p, COMMAND_CAM *pCmd_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM or UCMC_ISO2

usOption_p: Option defined in UCMCLIB.H file as UCMICC_OPEN_ALWAYS.

pCmd_p = Card command. See SDK

Return: FCT_OK or a negative value in the event of an error. Same value as standard function output_command().

12.15 ILIBUCM_ICC_PCODE

Warning: This function doesn't work with IUN device and is deprecated with CAD30 device. Used DLL Security API.

Description: previously entered code presentation request.

This command is used on secured peripherals.

Prototype: int iLIBUCM_Icc_PCode(unsigned char ucChannel_p, unsigned short usOption_p,

T_APDU *pC_apdu_p, T_APDU *pR_apdu_p, T_UCMC_ICC_CODE *pCode)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM

usOption_p: 0. RUF.

pC_apdu_p: Pointer on APDU command: Not currently used. Set NULL

pR_apdu_p: Pointer on response to APDU command

pCode : Enables to format the code entered in APDU command. Not used for current configurations.

Return: FCT_OK or a negative value in the event of an error.

12.16 ILIBUCM_ICC_POWERDOWN ( )

Description: Used to power down the card

According to parameter set-up, calls on power_down ()

Prototype: int iLIBUCM_Icc_PowerDown(unsigned char ucChannel, unsigned short usPowerDownType_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM

usPowerDownType_p: UCMCICC_POWER_DOWN ou UCMCICC_POWER_DOWN_SYNC

Return: FCT_OK or a negative value in the event of an error. Same value as standard function power_down()

12.17 ILIBUCM_ICC_POWERON ( )

Description: Used to power on the card and to send ATR.

Calls on the setup peripheral.

If the peripheral is not open, it opens and then closes it.

Prototype: iLIBUCM_Icc_PowerOn(unsigned char ucChannel_p, unsigned short usPowerOnType_p, HISTORIC *pHisto_p)

ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM

pHisto_p=Historical bytes or ATR.

usPowerOnType_p: UCMCICC_POWER_ON or UCMCICC_EMV_POWER_ON or UCMCICC_POWER_ON_SYNC or UCMCICC_POWER_ATR.

For ATR do power on before then use UCMCICC_POWER_ATR and UCMC_ATR structure instead of pHisto.

For UCMCICC_POWER_ON_SYNC use pHisto_p[0] = protocol number T1G,T2G ….

Return: FCT_OK or a negative value in the event of an error. Same negative value as standard function power_down()

0	= ok
-2	= invalid card 2 Answer To Reset is not EMV compliant
- 3	= card is mute
-4/5	= VCC or VPP problem
-6	= communication problem
-7	= card removed

12.18 ILIBUCM_ICC_READ_SWIPE ( )

Description: Swipe reader access, read magnetic card ISO2 if peripheral exist.

Exit if “Cancel” key is pressed or “cancel” MSG from protocol SES1042 received).

Prototype: int iLIBUCM_Icc_Read_Swipe(unsigned char ucChannel_p, unsigned short usOption_p, T_UCMC_ICC_TRACK *pTrack_p)

ucChannel_p: Channel. Use UCMC_ISO2.

usOption_p: Not used

pTrack_p: Icc structure command and result.

UcTypeTrack=2 for ISO2 track reading

UcResultLg_ref = length of PAN. Track could be size of buffer.

IResultStatus=Result of track reading: Example UCMHOST_TRACK_OK or UCMHOST_TRACK_CANCEL.

UsTimeout = Maximun waiting value in second of card reading. 0= No wait.

u.sTrack1= track value.

Return: 0= FCT_OK or other value (example UCMTTESTALL_CR_NO_EVENT) if error or cancel.

12.19 ILIBUCM_ICC_READ_SWIPEMULTI ( )

Description: Swipe reader access, read magnetic card ISO1 or 2 or 3 if peripheral exist.

Exit if “Cancel” key is pressed or “cancel” MSG from protocol SES1042 received).

Same functionality as Is_iso1(), Is_iso2() and is_iso3() using by other ingenico terminals.

Prototype: int iLIBUCM_Icc_Read_SwipeMulti(unsigned char ucChannel_p, unsigned short usOption_p, T_UCMC_ICC_TRACKMULTI *psTrack_p) ;

ucChannel_p: Channel. Use UCMC_ICC.

usOption_p: Not used.

pTrack_p: Icc structure command and result.

UcTypeTrack= 1=Track1 and/or 2=Track2 and/or 4=Track3. Check before if canal exist.

Example: 1+2+4=7 for all tracks.

UcResultLg_ref1 = length of Pan1. Track could be size of buffer

IresultStatus1=Result of track1 reading: Example UCMHOST_TRACK_OK or UCMHOST_TRACK_CANCEL.

UsTimeout = Maximun waiting value in second of card reading. 0= No wait.

sTrack1= track1 value.

Return: 0= FCT_OK or other value (example UCMTTESTALL_CR_NO_EVENT) if error or cancel.

12.20 ILIBUCM_ICC_STATUS ( )

Description: Sends reader status, even without opening the peripheral. Same as for OEMC status (CAM, &status).

Prototype: int iLIBUCM_Icc_Status(unsigned char ucChannel_p, unsigned char *puc_Card_p)
ucChannel_p: Channel. By default, use UCMC_ICC or UCMC_SAM

puc_Card_p: ICC status.

Return: UCMC_STATUS_CARD_OUTSIDE UCMC_STATUS_CARD_INSIDE or other value (example UCMTTESTALL_CR_NO_EVENT) if error.

12.21 ILIBUCM_ICC_TTESTALL ( )

Description: Waits for an event on channel peripheral.

Prototype: int iLIBUCM_ Icc _Ttestall(unsigned char ucChannel , unsigned short usDelay_p)

ucChannel_p: Pinpad channel. Use UCMC_ICC by default.

usDelay_p: Timeout within 10ms

Return: UCMTTESTALL_CR_OK_EVENT or UCMTTESTALL_CR_TIMEOUT or a negative value of an error.

13. UCM COMPONENT LED IAC

13.1 INTRODUCTION

The LEDs "'visible to the user" can either be located on a reader, a display or the host. LEDs located on UCM are managed by the system. Not covered by the section, see specifications SES0107. 3 LEDs (YELLOW, RED, GREEN) are provided per channel (location).

The UCM component functions are used to manage the LEDs.

13.2 ILIBUCM_LED_CMD ( )

Description: Used to command the LEDs.

Prototype: int iLIBUCM_LED_Cmd(T_UCMC_IAC_LED *pdata_p) ;

pdata_p = see ucmclib.h file for description of T_UCMC_IAC_LED.

Return: FCT_OK or a negative value in the event of an error.

13.3 ILIBUCM_LED_EXIST ( )

Description: Sends data on the type defined in parameter set-up..

Prototype: int iLIBUCM_LED_Exist(void)

Return: Value of the type defined in parameter set-up, in decimals.

13.4 ILIBUCM_LED_INIT ( )

Description: Initializes LED management. Only used by the UCM.

13.5 ILIBUCM_LED_STATUS ( )

Description: Sends the LED status;

Prototype: int iLIBUCM_LED_Status (unsigned long *pulLed_p) ;

pulLed_p : LED status. See UcmcLib.h file for definitions of masks for each LED.

Return: FCT_OK or a negative value in the event of an error.

13.6 ILIBUCM_LED_CMDCH ( )

Description: Used to command the LEDs.

Prototype: int iLIBUCM_LED_CmdCh(unsigned char ucChannel_p, T_UCMC_IAC_LED *pdata_p) ;

ucChannel_p: Led channel. Use UCMC_LED by default.

pdata_p = see ucmclib.h file for description of T_UCMC_IAC_LED.

Return: FCT_OK or a negative value in the event of an error.

13.7 ILIBUCM_LED_EXISTCH ( )

Description: Sends data on the type defined in parameter set-up.

Prototype: int iLIBUCM_LED_ExistCh(unsigned char ucChannel_p)

ucChannel_p: Led channel. Use UCMC_LED by default.

Return: Value of the type defined in parameter set-up, in decimals.

13.8 ILIBUCM_LED_INITCH ( )

Description: Initializes LED management. Only used by the UCM.

13.9 ILIBUCM_LED_STATUSCH ( )

Description: Sends the LED status;

Prototype: int iLIBUCM_LED_StatusCh (unsigned char ucChannel_p , unsigned long *pulLed_p) ;

ucChannel_p: Led channel. Use UCMC_LED by default.

pulLed_p : LED status. See UcmcLib.h file for definitions of masks for each LED.

Return: FCT_OK or a negative value in the event of an error.

14. UCM COMPONENT MODEM IAC

14.1 INTRODUCTION

The modem can either be internal or external, connected directly or to the host.

14.2 ILIBUCM_MODEM_EXIST ( )

Description: Sends data on the type defined in parameter set-up.

Prototype: extern int iLIBUCM_Modem_Exist(void) ;

Return: Value of the type defined in parameter set-up, in decimals. 2= Managed by Host.

14.3 ILIBUCM_MODEM_CNX ( )

Description: Sends the connection frame to the modem. Called on by the manager.

Prototype: int iLIBUCM_Modem_Cnx (T_UCMC_IAC_MODEM_CNX *pCnx_p);

pCnx_p : See T_UCMC_IAC_MODEM_CNX

Return: FCT_OK or a negative value in the event of an error.

14.4 ILIBUCM_MODEM_CONFIG ( )

Description: Gives the complete modem configuration as defined in the parameter set-up.
Reserved for future use.

14.5 ILIBUCM_MODEM_DCNX ( )

Description: Disconnects the modem. Called on by the manager.

Prototype: int iLIBUCM_Modem_Dcnx(unsigned char ucOption_p);

ucOption_p: Reason for disconnection

Return: FCT_OK or a negative value in the event of an error.

14.6 ILIBUCM_MODEM_INIT ( )

Description: Initializes the modem. Only used by the UCM.

14.7 ILIBUCM_MODEM_READ ( )

Description: Peripheral read. Not used; RUF.

14.8 ILIBUCM_MODEM_WRITE ( )

Description: Peripheral write. Not used; RUF.

15. UCM COMPONENT PAYMENT IAC

15.1 INTRODUCTION

These entry points are used for payment.

They enable to realise a payment transaction in compliance with payment kinematics on the controller.

15.2 ILIBUCM_PAY_INIT ()

Description: Reserved for future use.

15.3 ILIBUCM_PAY_READY_FOR_DEBIT ()

Description: Called on by payment applications. Used to fill in the host protocol or DA when an application is in Debit_xxx() and is ready for a debit. The payment application gives the debit type:

Prototype: int iLIBUCM_Pay_Ready_For_Debit(int iSize_p, void * pData_p) ;

iSize_p: size of data passed in pData_p. i.e. sizeof(T_UCMHOST_DEBIT)

pData_p: Data passed: T_UCMHOST_DEBIT

Return: FCT_OK or a negative value in the event of an error.

15.4 ILIBUCM_PAY_RESULT_DEBIT ()

Description: Used to fill in the UCM component (protocol) of the debit request result (solvency, debit or recording).

Prototype: int iLIBUCM_Pay_Result_Debit(int iSize_p, void * pData_p) ;

iSize_p: size of data transmitted in pData_p. i.e. sizeof(T_UCMHOST_R_DEBIT)

pData_p: Data transmitted: T_UCMHOST_R_DEBIT

Return: FCT_OK or a negative value in the event of an error.

15.5 ILIBUCM_PAY_END ()

Description: Fills in the UCM component (protocol) when an application exits from Debit_xxx().

Prototype: int iLIBUCM_Pay_End(int iSize_p, void * pData_p)

iSize_p: size of data passed in pData_p, i.e. 0.

pData_p: Data passed: NULL

Return: FCT_OK or a negative value in the event of an error.

15.6 ILIBUCM_PAY_HOST_CMD ()

Description: Sends a command to the host task (protocol).

Prototype: int iLIBUCM_Pay_Host_Cmd(int iSize_p, void * pData1_p, void * pData2_p)

iSize_p: size of data transmitted in pData1_p, i.e. sizeof(T_UCMC_IAC_HOST.)

pData1_p: Data transmitted, see T_UCMC_IAC_HOST

pData1_p: Data received, see T_UCMC_IAC_HOST

Return: FCT_OK or a negative value in the event of an error.

15.7 ILIBUCM_PAY_HOST_GET_LAST_CMD ()

Description: Identifies the last message received from the host.

Prototype: int iLIBUCM_Pay_Host_Get_Last_Cmd(int iSize_p, void pData1_p *, void * pData2_p)

iSize_p: size of data transmitted in pData1_p. i.e. sizeof(T_UCMC_IAC_HOST.)

pData1_p: Data transmitted, see T_UCMC_IAC_HOST

pData1_p: Data received, see T_UCMC_IAC_HOST

Return: FCT_OK or a negative value in the event of an error.

15.8 ILIBUCM_HOST_INIT ()

Description: Initialize Host protocol.

This function need specific UCMC parameters. Before read HOST Device configuration. Use iLIBUCM_Device_Config_OPE(UCM_DEVICE_HOST, &sDevice,). if sDevice. UcType not equal 0 and sDevice ucPilote equal 0 and sDevice. UctNomDriver not equal “0xFFFF” it’s possible to initialize Host DLL. Il is forbidden in other case.

To simulate Host it’s necessary to implement

Prototype: int iLIBUCM_Host_ Read(int iSize_p, void *pData_p, int *piSizeData_p) ;

iSize_p: size of data transmitted in pData1_p, i.e. sizeof(T_UCMC_IAC_HOST.)

pData_p: Data transmitted

piSizeData _p: Size of data transmitted.

Return: FCT_OK or a negative value in the event of an error.

Exemple:

After_reset function :

```
  if( ( sDevice.ucType == 1 ) && ( sDevice.ucPilote == 0 ) &&
     ( memcmp(&sDevice.uctNomDriver[0] ,"0xFFFF", 6) == 0) )
        {
             sDevice.ucType = 1 ;
             sDevice.ucMode = 0 ;
             sDevice.ucPilote = 2 ;
             sprintf( &sDevice.uctNomDriver[0], "%s", "0x006E" ) ;  /*  type of application see ADF
file */
             sprintf( &sDevice.uctNomDll[0], "%s", "HOTE10S" ) ;   /* Dll name */
             sDevice.ucCom = 0;  /* Port com number: COM is managed by application in this case
*/

             memclr ( &sDevice.u.uctData[ 0 ], UCM_MAX_SIZE_DATA_DEVICE );

             iRet = iLIBUCM_Host_Init( &sDevice ) ;
        if( iret >= 0 ) {  /* Ok */ }
        }
```

To receive and send messages see banking Sample. Service IAC UCMAPPLIMODULE_PROTOCOL.

15.9 ILIBUCM_HOST_READ ()

Description: Reads a message received by the host (protocol).
This message is unknown by UCM component. The message must be defined in Host protocol.
Not implemented. RUF.

Prototype: int iLIBUCM_Host_ Read(int iSize_p, void *pData_p, int *piSizeData_p) ;
iSize_p: size of data transmitted in pData1_p, i.e. sizeof(T_UCMC_IAC_HOST.)
pData_p: Data transmitted
piSizeData _p: Size of data transmitted.

Return: FCT_OK or a negative value in the event of an error.

15.10 ILIBUCM_HOST_SEND ()

Description: Sends a message to the host (protocol).
This message is unknown by UCM component. The message must be defined in Host protocol.
Not implemented. RUF.

Prototype: int iLIBUCM_Host_Send(int iSize_p, void *pData_p, void *psResult_p)
iSize_p: size of data transmitted in pData_p.
pData_p: Data transmitted
psResult_p: Data result

Return: FCT_OK or a negative value in the event of an error.

16. UCM COMPONENT DATA INPUT IAC

16.1 INTRODUCTION

The following entry points are IACs that can be used by the payment applications.

Two keypads are managed.

16.2 ILIBUCM_PINPAD_CLOSE ( )

Description: Closes pinpad peripheral

Prototype: int iLIBUCM_Pinpad_Close(unsigned char ucChannel)

ucChannel_p: keypad channel. Use UCMC_PPAD by default

Return: FCT_OK or a negative value in the event of an error.

16.3 ILIBUCM_PINPAD_CMD ( )

Description: Keypad peripheral command.

Description: Sends pinpad command.

Prototype: iLIBUCM_Pinpad_Cmd(unsigned char ucChannel_p, T_UCM_IAC_PINPAD_CMD *sCmd_p) ;

ucChannel_p: Keyboard channel. Use UCMC_PPAD by default

sCmd_p : pointer of the structure described in the ucmclib.h file.

See sample UCM_PPAD_CMD_KEY_CTRL.

Return: : FCT_OK or other value if error.

16.4 ILIBUCM_PINPAD_EXIST ( )

Description: Sends data on the type defined in parameter set-up.

Prototype: int iLIBUCM_Pinpad_Exist (unsigned char ucChannel)

ucChannel_p: Keyboard channel. Use UCMC_PPAD by default

Return: Value of the type defined in parameter set-up, in decimals.

16.5 ILIBUCM_PINPAD_GETCHAR ( )

Description: Sends the code of the last key pressed.
Nonblocking. Time out equal 260s.

Prototype: int iLIBUCM_Pinpad_GetChar (unsigned char ucChannel)
ucChannel_p: Pinpad channel. Use UCMC_PPAD by default.

Return: Function Key code or numerical key code (only with IUP250 and CAD30 Tool).
Returns UCMTTESTALL_CR_TIME_OUT if time out.

16.6 ILIBUCM_PINPAD_INIT ( )

Description: Initializes the keyboard peripheral. Only used by the UCM module.

16.7 ILIBUCM_PINPAD_INPUT ( )

Warning: This function doesn't work with IUN device and is deprecated with CAD30 device. Used DLL Security API. ECHO_NORMAL doesn't work with UPP. If ucType equal UCM_PPAD_CODE, card must be inserted.

ucType value	UCM_PPAD_NUMERIQUE	UCM_PPAD_CODE
CAD30 TOOL	OK	OK
CA30 PPS-D	OK	OK
CAD30 UPP	KO	OK but deprecated. Use DLL Security.
IUP250	KO	KO. Use DLL Security.
IUP250 (maintenance)	OK	KO. Use DLL Security.
IUC180B	KO	KO

Description: Used to enter a pincode (UCM_PPAD_CODE) and to activate function keys and figure keys (UCM_PPAD_NUMERIQUE), according to the 'ucType' field of T_UCM_ENTRY_PPAD structure.

Prototype: int iLIBUCM_Pinpad_Input(unsigned char ucChannel, unsigned short usOption_p, T_UCM_ENTRY_PPAD *pCmd_p)

ucChannel_p: Pinpad channel. Use UCMC_PPAD by default.

usOption_p : Options like UCMPPAD_OPEN_IF_NOT. See ucmclib.h

pCmd_p : pointer on the structure described in the ucmclib.h file. Defines the authorized keys, the number of characters to be entered, the timeout, etc.

Return: FCT_OK or other value if error (-13976 if not supported).

ucResultCr =UCM_PPAD_RESULTCR_TIME_OUT if time out.

16.8 ILIBUCM_PINPAD_IS_OPEN ( )

Description: Informs that the peripheral is open.

Sends "Open" if the pinpad is set up on the host.

Prototype: int iLIBUCM_ Pinpad _Is_Open(unsigned char ucChannel_p)

ucChannel_p: Pinpad channel. Use UCMC_PPAD by default.

Return: 0=Closed. 1=Open. Negative=Error.

16.9 ILIBUCM_PINPAD_IS_CONNECT ( )

Description: Informs if the peripheral is connect.

Prototype: int iLIBUCM_ Pinpad _Is_Connect(unsigned char ucChannel_p)

ucChannel_p: Pinpad channel. Use UCMC_PPAD by default.

Return: 0=Connect. Negative=Error.

16.10 ILIBUCM_PINPAD_OPEN ( )

Description: Opens pinpad peripheral.

No effect if the pinpad is on host.

Prototype: int iLIBUCM_Pinpad_Open(unsigned char ucChannel_p, char *cOption_p) ;

ucChannel_p: Pinpad channel. Use UCMC_PPAD by default.

cOption_p : Options like "R", "R"

Return: FCT_OK or a negative value in the event of an error.

16.11 ILIBUCM_PINPAD_OPTION ( )

Description: Pinpad peripheral command. Used by the UCM. Reserved for future use.

16.12 ILIBUCM_ PINPAD_STATUS ( )

Description: Sends the status of the pinpad peripheral, even without opening the peripheral.

No effect if the pinpad is on host.

Prototype: int iLIBUCM_Pinpad_Status(unsigned char ucChannel, unsigned char *puc_Status_p)

ucChannel_p: Pinpad channel. Use UCMC_PPAD by default.

* puc_Status_p:	0 = peripheral closed
	1 = peripheral open

Return: FCT_OK or a negative value in the event of an error.

16.13 ILIBUCM_ PINPAD_TTESTALL ( )

Description: Waits for an event on the pinpad.

Prototype: int iLIBUCM_Pinpad_Ttestall(unsigned char ucChannel , unsigned short usDelay_p)

ucChannel_p: Pinpad channel. Use UCMC_PPAD by default.

usDelay_p: Timeout within 10ms

Return: FCT_OK or a other value in the event of an error. UCMTTESTALL_CR_TIME_OUT if time out.

17. UCM COMPONENT PRINTING IAC

17.1 INTRODUCTION

The following entry points are IACs that can be used by the payment applications.
Two printers can be used.
The options allowed in the messages are:
« \1E » to switch to bold if the printer allows for it.

17.2 ILIBUCM_PRINT_CLOSE ( )

Description: Closes printing peripheral.
No effect if the printer is on host.

Prototype: int iLIBUCM_Print_Close(unsigned char ucChannel_p) ;
ucChannel_p: Printer channel. Use UCMC_PRINT by default
Return: FCT_OK or a negative value in the event of an error.

17.3 ILIBUCM_PRINT_CMD ( )

Description: Printing peripheral command. Used by the UCM component.

17.4 ILIBUCM_PRINT_CUTPAPER ( )

Description: Cut-paper command. Calls on the setup peripheral.
Does not work on all printers.
Prototype: int iLIBUCM_Print_CutPaper (unsigned char ucChannel_p)
ucChannel_p: Printer channel. Use UCMC_PRINT by default
Return: FCT_OK or a negative value in the event of an error.

17.5 ILIBUCM_PRINT_DEFPRINTERPATTERN ( )

Description: Manages pattern on printer.

Same as oemc defprinterpattern') function.

Prototype: int iLIBUCM_Print_Defprinterpattern(unsigned char ucChannel_p, char cKey_p, *pcMessage_p) ;

ucChannel_p: Display channel. Use UCMC_PRINT by default.

ckey_p: Same parameter as oemc defprinterpattern') function.

PcMessage_p: Same parameter as oemc defprinterpattern') function.

Return: FCT_OK or a negative value in the event of an error.

17.6 ILIBUCM_PRINT_EXIST ( )

Description: Sends the value of the type defined in parameter set-up.

Prototype: int iLIBUCM_Print_Exist(unsigned char ucChannel_p)

ucChannel_p: Printer channel. Use UCMC_PRINT by default

Return: Value of the type defined in parameter set-up, in decimals.

17.7 ILIBUCM_PRINT_INIT ( )

Description: Initializes the printing peripheral. Only used by the UCM module.

17.8 ILIBUCM_PRINT_IS_OPEN ( )

Description: Informs that the peripheral is open.

Sends "open" if the printer is on host.

Prototype: int iLIBUCM_Print_Is_Open(unsigned char ucChannel_p)

ucChannel_p: Printer channel. Use UCMC_PRINT by default

Return: 0=Closed. 1=Open. Negative=Error.

17.9 ILIBUCM_PRINT_NEWLINE ( )

Description: Paper feed command.

Prototype: int iLIBUCM_Print_NewLine(unsigned char ucChannel_p, unsigned char ucNbLine)
ucChannel_p: Printer channel. Use UCMC_PRINT by default.
ucNbLine = Number of the following line.

Return: FCT_OK or a negative value in the event of an error.

17.10 ILIBUCM_PRINT_MESSAGE ( )

Description: Manages the printing of a character string.
Calls on the configured peripheral.
If the peripheral is not open, it opens and then closes it.
Uses the options: UCMPRINT_OPEN_IF_NOT | UCMPRINT_CLOSE_IF_OPEN
| UCMPRINT_WAIT_END

Prototype: int iLIBUCM_Print_Message(unsigned char ucChannel_p, char *cMessage_p) ;
ucChannel_p: Display channel. Use UCMC_PRINT by default.
cMessage_p: Message to be printed.

Return: FCT_OK or a negative value in the event of an error.

17.11 ILIBUCM_PRINT_OPEN ( )

Description: Opens printing peripheral.
No effect if the printer is on host.

Prototype: int iLIBUCM_Print_Open(unsigned char ucChannel_p, char *cOption_p) ;
ucChannel_p: Display channel. Use UCMC_PRINT by default.
cOption_p : Option like "W", "W"

Return: FCT_OK or a negative value in the event of an error.

17.12 ILIBUCM_PRINT_OPTION ( )

Description: Manages the printing of a character string.

Uses the options defined by the initiator

Prototype: int iLIBUCM_Print_Option(unsigned char ucChannel_p, char *cMessage_p , unsigned short usOption_p)

ucChannel_p: Display channel. Use UCMC_PRINT by default.

cMessage_p: Message to be printed.

usOption_p: Options like UCMPRINT_OPEN_IF_NOT. See ucmclib.h

Return: FCT_OK or a negative value in the event of an error.

17.13 ILIBUCM_PRINT_STATUS ( )

Description: Sends the printer status; even without opening the peripheral.

Prototype: int iLIBUCM_Print_Status(unsigned char ucChannel_p, unsigned char *pucState_p)

ucChannel_p: Display channel. Use UCMC_PRINT by default.

pucState_p: Pointer on 1 byte. Printer status.

Return: 0=Ok. Negative=Error.

17.14 ILIBUCM_PRINT_TTESTALL ( )

Description: Waits for a printer event;

Prototype: int iLIBUCM_Print_Ttestall(unsigned char ucChannel_p, unsigned short usDelay_p)

ucChannel_p: Display channel. Use UCMC_PRINT by default.

usDelay_p : Timeout within 10 ms

Return: 0=Ok. 1 =Event. Negative=Error.

18. UCM APPLICATION ERROR MANAGEMENT IAC

18.1 INTRODUCTION

The applications cannot access this file, but it can be used by the UCMC.

The file is cyclic and cannot contain more than 200 lines of 80 characters (16 kbytes max.). The fields are separated by semi-colons for processing in a spreadsheet program.

The file can be consulted (display or printing) via a maintenance terminal.

18.2 CONTENTS

The error file contents for each line are as follows:

Date; time; degree; error; error wording (max=50 characters).

Degree= E for an error. The terminal enters into a loop for 10 seconds. The 3 LEDs are on. Error display, if possible.

S: Same as E. Fatal error with reset after 10 seconds.

F: Same as E. Fatal error but no restart. The UCM remains blocked.

W: Warning without display.

Error = xxxxzzyy on 4 bytes

Xxxx=application name   Zz=function number   Yy=error number

Example:

060619 13:40:36; (E) UCMC 1203 -14996; iUCMBUZZER_Init() return:-14996 ;

Error (E) happened on 19 June 2006, at 01:40 pm

It happened while the UCMC component was in phase 1203 (UCM_PARAM_FCT, see ucmtelium.h file).

The error is –14996 (14decimal=buzzer module see LSB of UCM_BUZZER_FCT=0x120E ; 996=error number of this function).

The function is "iUCMBUZZER_Init()".

The return is the same error because it is a direct call.

18.3 LED ERROR MANAGMENT

A flashing red LED indicates a hardware fault.

A flashing green LED indicates a software fault.

LED flashes red and green: Telium Manager not initialized.

The number of flashes indicates the number of the device involved.

1 flashing:	HOST2
2 flashings:	Buzzer, Led of second device
3 flashings:	Reader0
4 flashings:	Reader1
5 flashings:	SAM0
6 flashings:	SAM1
7 flashings:	SAM2
8 flashings:	Display0
9 flashings:	Display2
10 flashings:	Pinpad0
11 flashings:	Pinpad2
12 flashings:	Printer0
13 flashings:	Printer1
14 flashings:	Modem
15 flashings:	External Led
16 flashings:	Buzzer
17 flashings:	HOST1

18.4 ERROR LIST

Error number	Description
-17999	HOST2: no com
-17998	HOST2: initialization
-17997	HOST2: no message
-17996	HOST2: not available
-17995	HOST2: ISO2 request send message
-17994	HOST2: Date change request send message
-17993	HOST2: Maintenance mode request message
-17992	HOST2: Application request send message
-17991	HOST2: Device command size
-16999	DEVICE: initialization
-16998	DEVICE: Size Message
-16997	DEVICE: Open
-16996	DEVICE: No device
-16995	DEVICE: library service
-16994	DEVICE: no command
-16993	DEVICE: library service get
-16992	DEVICE: library service call
-16991	DEVICE: library memory allocation

-16990	DEVICE: library null pointer
-16989	DEVICE: no DLL
-16988	DEVICE: ICC initialization
-16987	DEVICE: DLL initialization
-16986	DEVICE: event service no
-16985	DEVICE: event service application full
-16984	DEVICE: event service out of order
-16983	DEVICE: event service no event
-16982	DEVICE: max tabulation
-16981	DEVICE : reset
-16980	DEVICE : system initialization. Bad exchange Reader and Pinpad.
-16979	DEVICE: com initialization
-16978	DEVICE: get version
-16977	DEVICE: get diagnostic
-16976	DEVICE: command platform
-16975	DEVICE: command parameters
-16974	DEVICE: command execution
-16973	DEVICE: not exist
-16972	DEVICE: command enable by manager
-15999	LED: initialization
-15998	LED: Size Message
-15997	LED: Open
-15996	LED: No device
-15995	LED: Status
-15994	LED: library service get
-15993	LED: library service call
-15992	LED: no command
-15991	LED: no channel
-15990	LED: no default device
-15989	LED: not opened
-15988	LED: library memory allocation
-14999	BUZZER: initialization
-14998	BUZZER: Size Message
-14997	BUZZER: No device
-14996	BUZZER: no command
-14995	BUZZER: Status
-14994	BUZZER: library service get
-14993	BUZZER: library service call
-14992	BUZZER: no channel
-14991	BUZZER: no default device
-14990	BUZZER: not opened
-14989	BUZZER: library memory allocation
-13999	PINPAD: initialization
-13998	PINPAD: Size Message
-13997	PINPAD: Open
-13996	PINPAD: no channel
-13995	PINPAD: no device

-13994	PINPAD: no default device
-13993	PINPAD: not opened
-13992	PINPAD: library service get
-13991	PINPAD: library service call
-13990	PINPAD: unknown entry type
-13989	PINPAD: unknown response entry
-13988	PINPAD: IAC processing
-13987	PINPAD: IAC not available
-13986	PINPAD: IAC not authorized
-13985	PINPAD: unauthorized state
-13984	PINPAD: out of service state
-13983	PINPAD: unknown state
-13982	PINPAD: function state
-13981	PINPAD: library memory allocation
-13980	PINPAD: device driver
-13979	PINPAD: device detected
-13978	PINPAD: transfer to booster
-13977	PINPAD: device disconnected
-13976	PINPAD: not supported
-13975	PINPAD: no command
-13974	PINPAD: command platform
-13973	PINPAD: command parameters
-13972	PINPAD: command execution
-12999	ICC: initialization
-12998	ICC: Size Message
-12997	ICC: Open
-12996	ICC: no channel
-12995	ICC: no device
-12994	ICC: no default device
-12993	ICC: no SAM
-12992	ICC: no default SAM
-12991	ICC: not opened
-12990	ICC: power on
-12989	ICC: power down
-12988	ICC: not inside
-12987	ICC: no ISO 1
-12986	ICC: no ISO2
-12985	ICC: no ISO3
-12984	ICC: library service get
-12983	ICC: library service call
-12982	ICC: library memory allocation
-12981	ICC: submit pin
-12980	ICC: unknown command
-12979	ICC: size length
-12978	ICC: unauthorized state
-12977	ICC: out of service state
-12976	ICC: device detected
-12975	ICC: device driver
-12974	ICC: no DLL

-12973	ICC: no configurate
-12972	ICC: initialization driver
-12971	ICC: submit pin
-12970	ICC: unknown state
-12969	ICC: not supported
-12968	ICC: no command
-12967	ICC: command platform
-12966	ICC: command parameters
-12965	ICC: command execution
-11999	MODEM: initialization
-11998	MODEM: Size Data
-11997	MODEM: connection
-11996	MODEM: disconnection
-11995	MODEM: library service get
-11994	MODEM: library service call
-11993	MODEM: Open
-11992	MODEM: Close
-11991	MODEM: Read
-11990	MODEM: Read no data
-11989	MODEM: Read no response
-11988	MODEM: write
-11987	MODEM: write no data
-11986	MODEM: connection no response
-11985	MODEM: connection response ko
-11984	MODEM: connection key
-11983	MODEM: connection wait
-11982	MODEM: library memory allocation
-08999	PAY: initialization
-08998	PAY: Solvability request
-08997	PAY: Solvability response
-08996	PAY: Record request
-08995	PAY: Record response
-08994	PAY: Solvability request send message
-08993	PAY: Record request send message
-08992	PAY: ISO2 request send message
-08991	PAY: library service get
-08990	PAY: library service call
-08989	PAY: library null pointer
-08988	PAY: HOST size message
-08987	PAY: HOST send message
-08986	PAY: library memory allocation
-07999	HOST: no com
-07998	HOST: initialization
-07997	HOST: no message
-07996	HOST: not available
-07995	HOST: ISO2 request send message
-07994	HOST: Date change request send message

-07993	HOST: Maintenance mode request message
-07992	HOST: Application request send message
-07991	HOST: Pay request send message
-07990	HOST: no command
-07989	HOST: Application data request send message
-07988	HOST: Device initialization send message
-07987	HOST: Device initialization not supported
-07986	HOST: Device initialization size
-07985	HOST: Device send message
-07984	HOST: Device application
-07983	HOST: Device command
-07982	HOST: Device command Data
-07981	HOST: Device command no to
-07980	HOST: Device command size
-07979	HOST: Device application size
-06999	FILE: initialization
-06998	FILE: Message size
-06997	FILE: open
-06996	FILE: no device
-06995	FILE: library service
-06994	FILE: no command
-06993	FILE: library service get
-06992	FILE: library service call
-06991	FILE: library memory allocation
-06990	FILE: library null pointer
-06989	FILE: No DLL
-06988	FILE: Null file
-05999	PRINTER: initialization
-05998	PRINTER: Size Message
-05997	PRINTER: Open
-05996	PRINTER: No device
-05995	PRINTER: no channel
-05994	PRINTER: no default device
-05993	PRINTER: not opened
-05992	PRINTER: library service get
-05991	PRINTER: library service call
-05990	PRINTER: library memory allocation
-05989	PRINTER: device out of order
-05988	PRINTER: not available
-04999	DISPLAY: initialization
-04998	DISPLAY: Size Message
-04997	DISPLAY: Open
-04996	DISPLAY: No device
-04995	DISPLAY: no channel
-04994	DISPLAY: no default device
-04993	DISPLAY: not opened
-04992	DISPLAY: library service

-04991	DISPLAY: library memory allocation
-04990	DISPLAY: library service get
-04989	DISPLAY: library service call
-04988	DISPLAY: library processing
-04987	DISPLAY: library not available
-04986	DISPLAY: Get no message
-04985	DISPLAY: not supported
-04984	DISPLAY: device disconnected
-04983	DISPLAY: Function Parameters
-04982	DISPLAY: Application function no service
-04981	DISPLAY: Application function no answer
-50	UCMHOSTLIB: link
-51	UCMHOSTLIB: object load
-52	UCMHOSTLIB: command unknown
-53	UCMHOSTLIB: dll name
-54	UCMHOSTLIB: write buffer
-55	UCMHOSTLIB: write buffer 2
-56	UCMHOSTLIB: write buffer 3
-57	UCMHOSTLIB: read buffer
-58	UCMHOSTLIB: dll com
-59	UCMHOSTLIB: dll bad pilote
-60	UCMHOSTLIB: dll no message
-61	UCMHOSTLIB: dll com handle
-62	UCMHOSTLIB: dll com number
-63	UCMHOSTLIB: dll com closed
-64	UCMHOSTLIB: dll data lg
-65	UCMHOSTLIB: dll data lg0
-66	UCMHOSTLIB: command not authorized
-67	UCMHOSTLIB: dll message unknown
-68	UCMHOSTLIB: dll message display unknown
-69	UCMHOSTLIB: dll message printer unknown
-70	UCMHOSTLIB: dll message modem unknown
-71	UCMHOSTLIB: dll message overrun
-72	UCMHOSTLIB: dll message creation
-73	UCMHOSTLIB: dll message icc unknown
-74	UCMHOSTLIB: dll message pinpad unknown
-75	UCMHOSTLIB: dll message buzzer unknown
-76	UCMHOSTLIB: dll message led unknown
-150	UCMHOSTLIB2: link
-151	UCMHOSTLIB2: object load
-152	UCMHOSTLIB2: command unknown
-153	UCMHOSTLIB2: dll name
-154	UCMHOSTLIB2: write buffer
-155	UCMHOSTLIB2: write buffer 2
-156	UCMHOSTLIB2: write buffer 3
-157	UCMHOSTLIB2: read buffer
-158	UCMHOSTLIB2: dll com
-159	UCMHOSTLIB2: dll bad pilote

-160	UCMHOSTLIB2: dll no message
-161	UCMHOSTLIB2: dll com handle
-162	UCMHOSTLIB2: dll com number
-163	UCMHOSTLIB2: dll com closed
-164	UCMHOSTLIB2: dll data lg
-165	UCMHOSTLIB2: dll data lg0
-166	UCMHOSTLIB2: command not authorized
-167	UCMHOSTLIB2: dll message unknown
-168	UCMHOSTLIB2: dll message display unknown
-169	UCMHOSTLIB2: dll message printer unknown
-170	UCMHOSTLIB2: dll message modem unknown
-171	UCMHOSTLIB2: dll message overrun
-172	UCMHOSTLIB2: dll message creation
-173	UCMHOSTLIB2: dll message icc unknown
-174	UCMHOSTLIB2: dll message pinpad unknown
-175	UCMHOSTLIB2: dll message buzzer unknown
-176	UCMHOSTLIB2: dll message led unknown

UCMC reset reason

0x9470	UCMEXIT_NEWPARAM
0x9471	UCMEXIT_PARAMFTC
0x9472	UCMEXIT_HOST_RESTART
0x9473	UCMEXIT_NEWPARAM_DRIVER
0x9474	UCMEXIT_DETECT_CADTOOL
0x9475	UCMEXIT_LOSS_CADTOOL
0x9476	UCMEXIT_DEVICE_RECONNECTION
0x9480	UCMEXIT_UCMCENTRY
0x9490	UCMEXIT_ERR_FTC
0x94A0	UCMEXIT_DISPLAYFCT
0x94B0	UCMEXIT_PRINT_FCT
0x94C0	UCMEXIT_FILE_FCT
0x94D0	UCMEXIT_HOST_FCT
0x94E0	UCMEXIT_PAY_FCT
0x94F0	UCMEXIT_DLLSTART_FCT
0x9500	UCMEXIT_DLLHOST_FCT
0x9501	UCMEXIT_DLLHOST_INIT
0x9510	UCMEXIT_MODEM_FCT
0x9520	UCMEXIT_ICC_FCT
0x9521	UCMEXIT_NO_TIME_FCT
0x9550	UCMEXIT_DLLHOST2_FCT
0x95F1	UCMEXIT_DLLHOST2_INIT
0x9610	UCMEXIT_PINPAD_FCT