Advanced Card Systems Ltd.
Card & Reader Technologies

eH880 and ACR880

Application Note: Software Programming

Subject to change without prior notice

info@acs.com.hk
www.acs.com.hk

Table of Contents

1.0. Introduction

This document includes other software programming topics that are not included in the API manual.

2.0. Serial Programming

The device node of eH880/ACR880 RS232/Serial port is ***/dev/tts/2***. User is able to use standard Linux serial programming to access the device node to send/receive data via the RS232/Serial port.

For more detail of Linux serial programming, you can refer to “Serial Programming HOWTO” (http://www.tldp.org/HOWTO/Serial-Programming-HOWTO/index.html)

3.0. Network Programming

For eH880/ACR880 connected to the Internet, user is able to use standard Linux socket programming to send/receive data via Internet.

For more detail about Linux socket programming, you can refer to “Beej’s Guide to Network Programming Using Internet Sockets” (http://beej.us/guide/bgnet/).

4.0. GPRS (for ACR880 with GPRS Option Only)

User is able to start GPRS Internet connection by calling the script ***"/usr/local/ro/GPRS/GPRS-pon <APN>***", where <APN> is the Access Point Name (APN) of your GPRS vendor.

For example, the APN for Peoples (HK) is "peoples.net" and the APN for China Mobile is "CMNET".

Ask the GPRS vendor to get the exact APN. The script in turn will start "***pppd***" with the dialer script "***/usr/local/ro/GPRS/GPRS-dialer***".

When the scripts are run successfully, the user will be able to connect to the Internet and send/receive data via Internet by standard Linux socket programming.

User may stop the GPRS Internet connection by calling the script " ***/usr/local/ro/GPRS/GPRS-poff*** ".

If your GPRS vendor requires you to set additional parameters other than APN to configure the GPRS connection, you can refer and modify the above scripts to fit your GPRS setting.

5.0. Contact Card Readers

By default, the Linux implementation of PC/SC API (***pcscd***) is enabled. User can access the embedded contact card readers by the PC/SC API. The reader name of the embedded contact reader is as following:

	Reader Name
First Card Slot	`A880 SCard Rdr 1 xx yy`
Second Card Slot	`A880 SCard Rdr 2 xx yy`
First SAM Slot	`A880 SAM Rdr 1 xx yy`
Second SAM Slot	`A880 SAM Rdr 2 xx yy`

Please note that the "***xx yy***" in above reader names are assigned by "***pcscd***" at startup. They are not necessarily assigned to the same number every time.

6.0. PC/SC API for intrinsic contactless reader

By default, the PC/SC API for the intrinsic contactless reader of eH880/ACR880 is not enabled.

Alternately, user could access the reader by the native/proprietary API defined in the API Manual.

User can enable the PC/SC API for the intrinsic contactless reader by loading a ***pcscd*** driver (***A880_PCSC_PCD.so***) during the boot time.

To do this, the user should modify the configuration file "***/usr/local/rw/etc/.sysconf***" in the device as seen below. The changes will be effective after the system reboot.

Configuration file location	/usr/local/rw/etc/.sysconf
Default (Disable)	`CONTACTLESS_PCSC=`
Enable	`CONTACTLESS_PCSC=ON`

After enabling the contactless PC/SC API, the intrinsic contactless reader could be accessed by using PC/SC API via the reader name "***A880 PCSC PCD xx yy***", where "**xx yy**" is the number assigned by the ***pcscd***. It is "***00 00***" in the default implementation.

Note: The use of the native/proprietary API and ***A880_PCSC_PCD.so*** (pcscd driver) for the intrinsic contactless reader at the same time will lead to conflict in resources.

Most of the current demos are written using the native API. Try not to run these demos when you enable the PC/SC API for the intrinsic contactless reader to avoid conflict.

The current version of ***A880_PCSC_PCD.so*** supports the following cards for PC/SC API access:

PC/SC Listed Contactless Smart Card: ISO14443 Type A, Part 4 compliant (e.g. Mifare DESFire) ISO14443 Type B, Part 4 compliant (e.g. GTML2)
PC/SC Listed Contactless Storage Card: Mifare Standard 1K (SS-NN = 03 00 01) Mifare Standard 4K (SS-NN = 03 00 02) Mifare Ultra Light (SS-NN = 03 00 03) SRI4K/SRT512B (SS-NN = 06 00 07)
Non-PC/SC Listed Contactless Card: CTM512B/CTS512B (SS-NN = 06 FF 01) GTML/CD97/CD21 (Innovatron) (SS-NN = 06 FF 02)

The current version of ***A880_PCSC_PCD.so*** implements the following PC/SC ***SCardControl*** command code:

Command code	Argument	Functionality
SCARD_CTL_CODE(1)	0x23,0x01,0x9E	Disable the card polling (insertion and removal detection) Return: 0x9000 [successful]
SCARD_CTL_CODE(1)	0x23,0x01,0x9F	Enable the card polling (insertion and removal detection) Return: 0x9000 [successful]
SCARD_CTL_CODE(1)	0x18,0x00	Get firmware version Return: e1 00 00 00 01 +String+90 00 [successful]

SCARD_CTL_CODE(1)	0x22,0x01,0x0A	Manual PICC Polling Return: e1 00 00 00 01 + Status+90 00 Status: 0x00 -- Card detected. 0xff -- No detected.
SCARD_CTL_CODE(1)	0x25,0x01,0x00	Turn off Antenna (Note: Please Disable Card Polling first)
SCARD_CTL_CODE(1)	0x25,0x01,0x01	Turn on Antenna (Note: Please Enable Card Polling after that)

The implementation of the ***A880_PCSC_PCD.so*** has the following differences compared to ACR128.

	ACR128	A880_PCSC_PCD.so
ATR for ISO14443, Type A, Part 4 Compliant card	Whole ATS	Only the Historical Bytes in ATS (see PCSC Specification 2.01.09, Part 3, Table 3-5)
ATR for SRI4K/SRT512B	SS = 07	SS = 06 as it is only ISO14443B part 2 compliant

Pseudo APDU for Mifare 1k/4k

Command	**Response**
Get Data (FF CA 00 00 00)	UID0 … UID3 SW1 SW2 UID0 … UID3: The returned UID
Load Keys (FF 82 00 P2 06 Key0 … Key5) P2: Key number to load to Key0 … Key5: The 6 bytes key data.	SW1 SW2
General Authenticate (FF 86 00 00 05 01 00 BlkNo KeyType KeyNo) BlkNo The Block number to login KeyType Mifare Key A = 0x60 Mifare Key B = 0x61 KeyNo: The Key to be used	SW1 SW2
Read Binary (FF B0 00 P2 Le) P2: The Block number to be read Le: Number of byte to be read	Data0 … DataN SW1 SW2 Data0 … DataN The returned data bytes

Update Binary (FF D6 00 P2 10 Data0 … Data15) P2: The Block number to be written Data0 … Data15: The 16 bytes data to be written.	SW1 SW2

Pseudo APDU for Mifare Ultra Light

Command	Response
Get Data (FF CA 00 00 00)	UID0 … UID3 SW1 SW2 UID0 … UID3: The returned UID
Read Binary (FF B0 00 P2 Le) P2: The Block number to be read Le: Number of byte to be read	Data0 … DataN SW1 SW2 Data0 … DataN The returned data bytes
Update Binary (FF D6 00 P2 Lc Data0 … DataN) P2: The Block number to be written Lc: Length of to be written. 4 Bytes Per Block Data0 … DataN: The N bytes data to be written.	SW1 SW2

Pseudo APDU for SRI4K/SRT512B

Command	Response
Get Data (FF CA 00 00 00)	UID0 … UID7 SW1 SW2 UID0 … UID7: The returned UID
Read Binary (FF B0 00 P2 Le) P2: The Block number to be read Le: Number of byte to be read	Data0 … DataN SW1 SW2 Data0 … DataN The returned data bytes
Update Binary (FF D6 00 P2 Lc Data0 … DataN) P2: The Block number to be written Lc: Length of to be written. 4 Bytes Per Block Data0 … DataN: The N bytes data to be written.	SW1 SW2

Pseudo APDU for CTM512B/CTS512B

Command	Response
Get Data (FF CA 00 00 00)	UID0 … UID4 SW1 SW2 UID0 … UID4: The returned UID
Read Binary (FF B0 00 P2 Le) P2: The Block number to be read Le: Number of byte to be read	Data0 … DataN SW1 SW2 Data0 … DataN The returned data bytes
Update Binary (FF D6 00 P2 Lc Data0 … DataN) P2: The Block number to be written Lc: Length of to be written. 2 Bytes Per Block Data0 … DataN: The N bytes data to be written.	SW1 SW2

Pseudo APDU for GTML/CD97/CD21 (Innovatron)

Command	Response
Get Data (FF CA 00 00 00)	UID0 … UID3 SW1 SW2 UID0 … UID3: The returned UID