ACR80 Extended PCSC Functions

Version 1.00
(14-April-2004)

Introduction

ACR80 is a smart card reader with built-in keypad and LCD display. In order to use these functions of ACR80 through PC/SC, some vendor-specific functions have to be defined. These functions are an extension to the ordinary PC/SC API's. This document describes the method for a PC/SC application program to use these functions.

The following functions are covered in this document.
- KeyPad
- LCD Display

Programming with the Extended Functions

PC/SC allows transmission of vendor-specific commands to the IFD through the SCardControl API. This API communicates with the IFD driver directly by sending the driver specific ControlCode to perform the required function.

The Function Prototype:

```
LONG SCardControl(
  SCARDHANDLE hCard,          // Card handle obtained by a call to SCardConnect
  DWORD dwControlCode,        // Code to sepcify which function to perform
  LPCVOID lpInBuffer,         // User supplied input buffer
  DWORD nInBufferSize,        // Size of user supplied input buffer
  LPVOID lpOutBuffer,         // User supplied output buffer
  DWORD nOutBufferSize,       // Size of user supplied output buffer
  LPDWORD lpBytesReturned     // Number of bytes actually returned
);
```

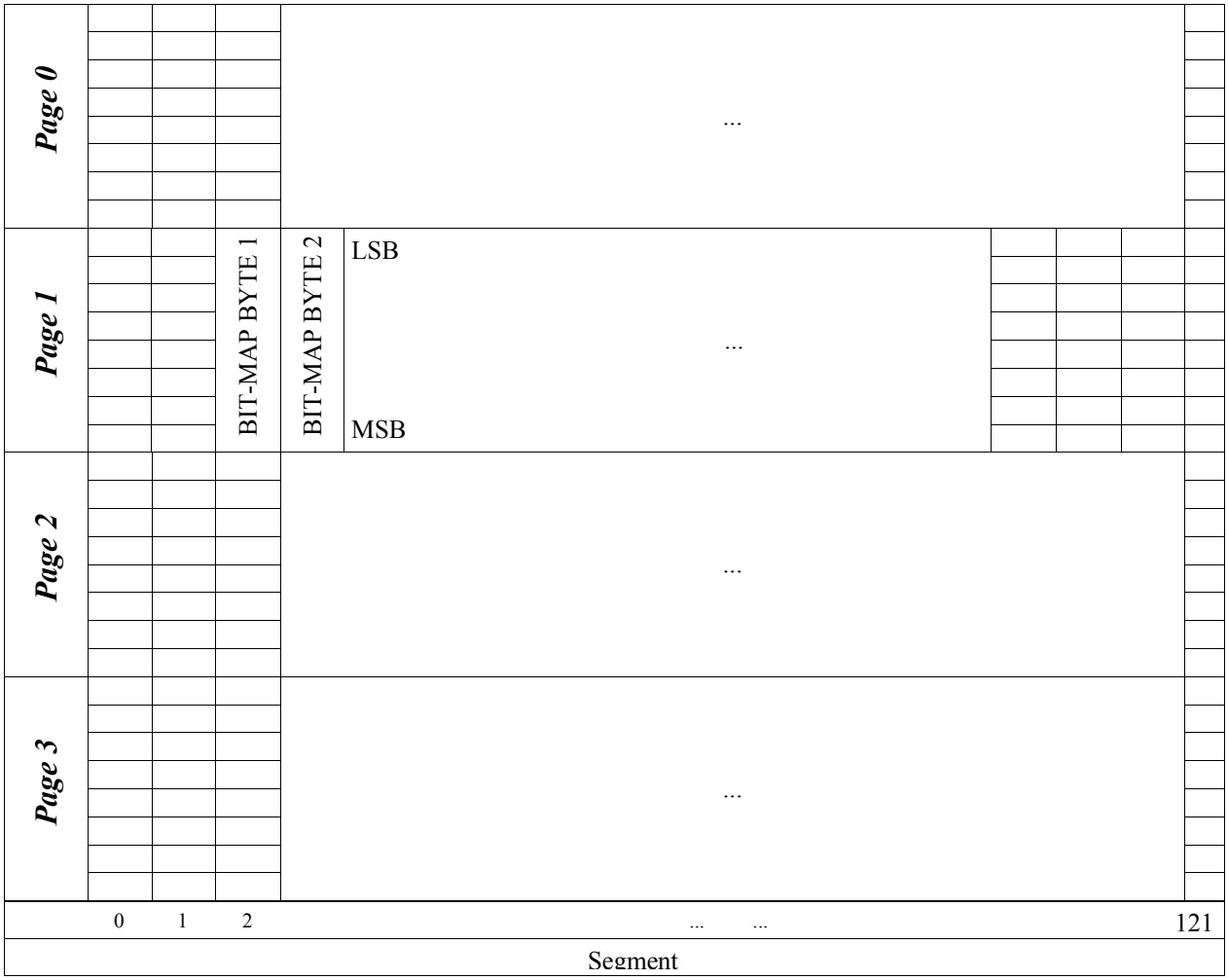
List of defined ControlCodes
- IOCTL_SMARTCARD_READ_KEYPAD
- IOCTL_SMARTCARD_DRAW_LCDBMP
- IOCTL_SMARTCARD_DISPLAY_LCD
- IOCTL_SMARTCARD_CLR_LCD

Programming with the LCD

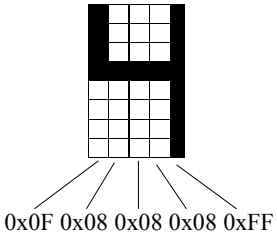
The LCD Module:

The LCD modules of ACR80 is divided into four pages. Each page is a 122 (width) x 8 (height) dot matrix display.

Following is the page and segment layout of the LCD modules.



Graphics and text are displayed on the LCD modules as bitmap. A bit value of "1" turns an LCD dot on while a "0" turns it off. Therefore, bytes sequence of [0x0F, 0x08, 0x08, 0x08, 0xFF] would produce a bit-map pattern (digital "4") as below.



The Display Mode:

LCD modules can be programmed in one of the three modes, Alphanumeric Mode, Map Mode and Full Map mode.

Alphanumeric Mode:
- This mode utilizes ACR80 driver built-in ASCII character map to display a character on a page of the display. It is the responsibility of application to calculate the correct offset to display the character.

Map Mode:
- This mode tells ACR80 to draw a 16 dots x 16 dots icon-sized graphic on LCD display. The icon is not restricted to the page boundary and can be located on any page at any offset value. It is the responsibility of application to prepare the icon graphic buffer and calculate the correct offset to display the icon.

Full Map Mode:
- This mode tells ACR80 to draw a 122 dots x 32 dots (i.e. The whole LCD display panel) full-sized graphic on LCD display. The graphic can be located on any page at any offset value. It is the responsibility of application to prepare the icon graphic buffer and calculate the correct offset to display the icon.

The Programming Function:

Control code applicable to LCD display function:
- IOCTL_SMARTCARD_DRAW_LCDBMP
- IOCTL_SMARTCARD_DISPLAY_LCD
- IOCTL_SMARTCARD_CLR_LCD

IOCTL_SMARTCARD_DRAW_LCDBMP:
Function:
- Instructs the driver to format the bitmap pattern and store the formatted pattern internally for LCD module to display.

Input Buffer:
- Pointer to the data structure below.

```
typedef struct _AC_GRAPHICS {
      BYTE        StartXPos;  // starting position of x-coordinate
      BYTE        StartYPos;  // starting position of y-coordinate
      BYTE        Type;       // mode of graphic
      BYTE        Map[488];   // ASCII string or bitmap of graphic
}AC_GRAPHICS;
```

Input Buffer Size:
- Size, in bytes, of the structure, AC_GRAPHICS.

Output Buffer:
- Pointer to a driver-filled status word of 2 bytes (*preliminary and subjects to change in the next release*).

Output Buffer Size:
- Size, in bytes, of the Output Buffer (*preliminary and subjects to change in the next release*).

Return Value:
- 4-byte smart card return code of type, long.

IOCTL_SMARTCARD_DISPLAY_LCD:
Function:
- Instruct the driver to display the formatted bitmap pattern to LCD module.

Input Buffer:
- Not used, but cannot be NULL

Input Buffer Size:
- 0

- Pointer to a driver-filled status word of 2 bytes (*preliminary and subjects to change in the next release*).

Output Buffer Size:
- Size, in bytes, of the Output Buffer (*preliminary and subjects to change in the next release*).

Return Value:
- 4-byte smart card return code of type, long.

IOCTL_SMARTCARD_CLR_LCD:
Function:
- Instruct the driver to clear the content of specific LCD pages.

Input Buffer:
- Byte bit-mask indicating the page(s) to be cleared.
    - Bit 0 = Page 0 to be cleared (1) or left unchanged (0)
    - Bit 1 = Page 1 to be cleared (1) or left unchanged (0)
    - Bit 2 = Page 2 to be cleared (1) or left unchanged (0)
    - Bit 3 = Page 3 to be cleared (1) or left unchanged (0)

Input Buffer Size:
- size, in bytes, of the bit-mask byte (i.e. 1 byte).

- Pointer to a driver-filled status word of 2 bytes (*preliminary and subjects to change in the next release*).

Output Buffer Size:
- Size, in bytes, of the Output Buffer (*preliminary and subjects to change in the next release*).

Return Value:
- 4-byte smart card return code of type, long.

Alphanumeric Example:

```
AC_GRAPHICS       LcdMap;
BYTE              lpInBuffer[256], lpOutBuffer[256]
DWORD             dwOutBufferSize=256, dwByteReturned=0;
INT               i,j;
long              ret;

LcdMap.StartXPos = 0;   // x-coordinate of alphanumeric (top-left corner)
LcdMap.StartYPos = 0;   // y-coordinate of alphanumeric (top-left corner)
LcdMap.Type = AC_ALPHANUMERIC;
LcdMap.Map[0] = 'A';    // display an 'A' on the LCD panel

// format the alphanumeric in the internal buffer of the driver
ret = SCardControl(
      hCard,
      IOCTL_SMARTCARD_DRAW_LCDBMP,
      (AC_GRAPHICS*) &LcdMap,
      sizeof(AC_GRAPHICS),
      lpOutBuffer,
      dwOutBufferSize,
      &dwByteReturned);

if(ret != SCARD_S_SUCCESS) {
      // error handling
      return ret;
}

// display the formatted alphanumeric to LCD panel
dwOutBufferSize = 256;
ret = SCardControl(
      hCard,
      IOCTL_SMARTCARD_DISPLAY_LCD,
      NULL,
      0,
      lpOutBuffer,
      dwOutBufferSize,
      &dwByteReturned);

if(ret != SCARD_S_SUCCESS) {
      // error handling
      return ret;
}

// clear only page 1
lpInBuffer[0] = 0x02;   // bit-mask: 00000010b
lpInBufferSize = 1;
lpOutBufferSize = 256;
ret = SCardControl(
      hCard,
      IOCTL_SMARTCARD_CLR_LCD,
      lpInBuffer,
      dwInBufferSize,
      lpOutBuffer,
      dwOutBufferSize,
      &dwByteReturned);
```

Map Example:

```
AC_GRAPHICS       LcdMap;
BYTE              lpInBuffer[256], lpOutBuffer[256]
DWORD             dwOutBufferSize=256, dwByteReturned=0;
INT               i,j;
long              ret;

// display the map graphic on Page 3 & 4
LcdMap.StartXPos = 0;   // x-coordinate of map graphic (top-left corner)
LcdMap.StartYPos = 16;  // y-coordinate of map graphic (top-left corner)
LcdMap.Type = AC_MAP;   // draw a 16 x 16 icon graphic (i.e. 32 bytes graphic)
// checkerboard pattern
memcpy(
      LcdMap.Map,
      "\x55\xAA\x55\xAA\x55\xAA\x55\xAA\x55\xAA\x55\xAA\x55\xAA\x55\xAA\x55\xAA
      \x55\xAA\x55\xAA\x55\xAA\x55\xAA\x55\xAA\x55\xAA\x55\xAA\x55\xAA", 32*sizeof(char));

// format the MAP in the internal buffer of the driver
ret = SCardControl(
      hCard,
      IOCTL_SMARTCARD_DRAW_LCDBMP,
      (AC_GRAPHICS*) &LcdMap,
      sizeof(AC_GRAPHICS),
      lpOutBuffer,
      dwOutBufferSize,
      &dwByteReturned);

if(ret != SCARD_S_SUCCESS) {
      // error handling
      return ret;
}

// display the formatted checkerboard pattern to LCD panel
dwOutBufferSize = 256;
ret = SCardControl(
      hCard,
      IOCTL_SMARTCARD_DISPLAY_LCD,
      NULL,
      0,
      lpOutBuffer,
      dwOutBufferSize,
      &dwByteReturned);

if(ret != SCARD_S_SUCCESS) {
      // error handling
      return ret;
}

// clear only page 1 & 3
lpInBuffer[0] = 0x0A;   // bit-mask: 00001010b
lpInBufferSize = 1;
lpOutBufferSize = 256;
ret = SCardControl(
      hCard,
      IOCTL_SMARTCARD_CLR_LCD,
      lpInBuffer,
      dwInBufferSize,
      lpOutBuffer,
      dwOutBufferSize,
      &dwByteReturned);
```

Full Map Example:

```
AC_GRAPHICS       LcdMap;
BYTE              lpInBuffer[256], lpOutBuffer[256]
DWORD             dwOutBufferSize=256, dwByteReturned=0;
INT               i,j;
long              ret;

LcdMap.StartXPos = 0;         // x-coordinate of map graphic (top-left corner)
LcdMap.StartYPos = 0;         // y-coordinate of map graphic (top-left corner)
LcdMap.Type = AC_FULL_MAP;    // draw a 122 x 32 graphic (i.e. 488 bytes graphic)

// turn all dots on the LCD panel on
memset(LcdMap.Map,0xFF,488*sizeof(char));

// format the MAP in the internal buffer of the driver
ret = SCardControl(
      hCard,
      IOCTL_SMARTCARD_DRAW_LCDBMP,
      (AC_GRAPHICS*) &LcdMap,
      sizeof(AC_GRAPHICS),
      lpOutBuffer,
      dwOutBufferSize,
      &dwByteReturned);

if(ret != SCARD_S_SUCCESS) {
      // error handling
      return ret;
}

// display the formatted checkerboard pattern to LCD panel
dwOutBufferSize = 256;
ret = SCardControl(
      hCard,
      IOCTL_SMARTCARD_DISPLAY_LCD,
      NULL,
      0,
      lpOutBuffer,
      dwOutBufferSize,
      &dwByteReturned);

if(ret != SCARD_S_SUCCESS) {
      // error handling
      return ret;
}

// clear only all pages
lpInBuffer[0] = 0x0F;   // bit-mask: 00001111b
lpInBufferSize = 1;
lpOutBufferSize = 256;
ret = SCardControl(
      hCard,
      IOCTL_SMARTCARD_CLR_LCD,
      lpInBuffer,
      dwInBufferSize,
      lpOutBuffer,
      dwOutBufferSize,
      &dwByteReturned);
```

Programming with the KeyPad

The Programming Function

Control code applicable to KeyPad function:
- IOCTL_SMARTCARD_READ_KEYPAD

IOCTL_SMARTCARD_READ_KEYPAD:
Function:
- Instruct the reader to scan its keypad for key input from the user.

Input Buffer:
- Pointer to 1-byte buffer storing the timeout constant, in second, to wait for each key press.

Input Buffer Size:
- Size, in bytes, of the Input Buffer

- Pointer to 3-byte buffer storing the keypad scan code and a driver-filled status word. The 1st byte of the buffer stores the keypad scan code. The 2nd and 3rd bytes stores the status word (*the status word is preliminary and subjects to change in the next release*)

Scan Code to Key Matching Table	
Returned Scan Code	**Key(s) on Keypad**
00_H to 09_H	**0 – 9**
$0A_H$	**Enter**
$0B_H$	**Clear**
$0C_H$	**F1**
$0D_H$	**F2**
$0E_H$	**F3**
$0F_H$	**F4**

Output Buffer Size:
- Size, in bytes, of the Output Buffer.

Return Value:
- 4-byte smart card return code of type, long.

KeyPad Example:

```
SCARDCONTEXT hContext = 0;
SCARDHANDLE  hCard = 0;
CHAR         szReadersName[100];
BYTE         lpInBuffer[256], lpOutBuffer[256];
DWORD        dwActiveProtocol = 0;
DWORD        dwInBufferSize = 256, dwOutBufferSize = 256, dwReadersNameSize = 100;
DWORD        dwByteReturned = 0;
long         ret;

ret = SCardEstablishContext(SCARD_SCOPE_USER,NULL,NULL,&hContext);
if(ret != SCARD_S_SUCCESS) {
      // error handling
      return ret;
}

ret = SCardListReaders(hContext,NULL,szReadersName,&dwReadersNameSize);
if(ret != SCARD_S_SUCCESS) {
      // error handling
      return ret;
}

ret = ScardConnect(
      hContext,
      szReadersName,
      SCARD_SHARE_DIRECT,
      0,
      &hCard,
      &dwActiveProtocol);

lpOutBuffer[0] = 0x00;
// continue to scan for keypad until “Enter” key is pressed
while(lpOutBuffer[0] != 0x0A) {
      // Issue command to read keypad
      dwInBufferSize = 1;
      dwOutBufferSize = 3;
      lpInBuffer[0] = 10;     // 10 seconds timeout
      cout << "Please press a key (ENTER to quit)\n";
      ret = SCardControl(
            hCard,
            IOCTL_SMARTCARD_READ_KEYPAD,
            lpInBuffer,
            dwInBufferSize,
            lpOutBuffer,
            dwOutBufferSize,
            &dwByteReturned);
      if(ret == SCARD_S_SUCCESS) {
            cout << "Pressed Key = " << (int) lpOutBuffer[0] << endl;
      }
}
```