HWR Library User Manual

V1.0 – August 26, 2005

19, Innovation Road 1 • Science-Based Industrial Park • Hsin-Chu • Taiwan • R.O.C.

886-3-578-6005 886-3-578-4418 www.sunplus.com.tw

Table of Content

Revision History

Revision	Date	By	Remark
V1.0	2005/08/26	pengtj	first revision

1 Function List

Index	name
1	GetHWRLibVersion
2	HRWInit
3	HWRSetMode
4	HWRClose
5	HandWriteRecognition
6	HWRBlockDisplayON
7	HWRBlockDisplayOFF

Attachment 1：

HWR DEMO

2 Attention

2.1.Hand Write Recognize user guide

- write in one of blocks. After strokes finish, you can write another block or wait 1 second then HWR will run recognize program itself.
- After HWR complete, it will show recognition result at below. User can stylus tap choose correct word.
- User can stylus tap small block which at right-below to clear all words of input.

2.2 HWR Hardware Requirement

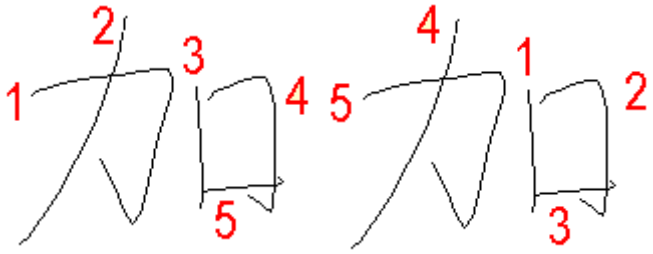
- **HWR Machine System Requirement**:
  System Requirement for HWR machine depends on applications.
- **SRAM**:
  Chinese/Japanese/Korean/Thailand… require 10KB SRAM
  English/Europe/Arabic/Hebrew… require 4KB SRAM
- **ROM (FONT Display Data is not included)**
  ROM size requirement depends on different character set of languages.
  The larger the character set of language is, the larger the size of ROM is.
  Complete Traditional Chinese (13060 characters) require 768KB ROM
  Simplified English + Digital (totally 90 characters) require 100KB ROM
- **CPU Speed**
  Hand Writing Recognition Speed dominantly depends on CPU speed
  Take "European + English + Digits + Symbols " as an example,
  CPU runs on 3MHz average takes 0.75sec
  CPU runs on 6MHz average takes 0.37sec
  Take "Simplified + Traditional Chinese " as an example,
  CPU runs on 3MHz average takes 2sec
  CPU runs on 6MHz average takes 1sec

2.3 FINEART HWR Feature

- **Supporting the multiplex recognition**. The kernel does the recognition process when the user is writing another character.
- **Offer an intelligent learning mode** to satisfy different user's need. Through this function, we give a solution to recognize rare characters and very cursive handwriting characters. The recognition program can adapt to user's handwritings over time.

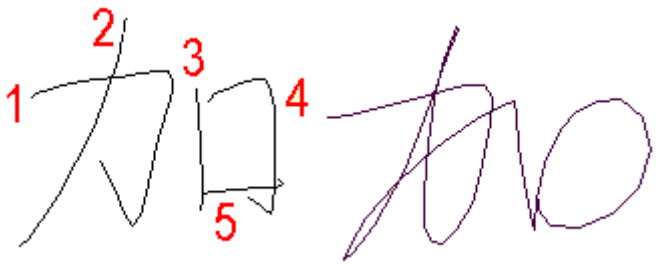
- **Directly accept most of the shorthand characters** have the flexibility in handwritings with connect strokes.
- **The kernel can automatically adjust the recognition speed** for connected-stroke characters and cursive handwritings. If you want to speed up the recognition process, write formally. If you want to have a cursive handwriting, the recognition speed will be slower but the recognition rate will try to keep high.
- **Mixed mode recognition**. You can combine different character set and recognize all of them at the same time. For example, combine Japanese, English and digits.
- **Support multiplex output code**. We support different output code formats Just like as follow: Unicode / GB / Big5 / CNS / JIS / KSC / ANSI / CCCII / Xerox:
- **Free stroke sequence input:** You can inverse any stroke sequence during input a character.

  - Original : - Free Stroke :



- **Cursive stroke support:** You can input a character with cursive. That means you can write down the character without pen up and that will help you to input character quickly.

  - Original : - Cursive Stroke :



Note that **Free stroke sequence input** and **Cursive stroke support** features can be disabled for some application (for example, Educational Product)

- **Customized Character Set or Customized Pattern Design.** Customers can define their own pattern or special character set for other applications (for example, Education Toy)

- **Scoring for each recognition.** Scoring for hand writing recognition, this feature is good for Hand Writing Auxiliary Learning Product.
- **Supporting multiplex language.**

Traditional Chinese (BIG5)	13,060 characters
Full Simplified Chinese (GB2312)	6763 characters
Full Korean symbol 2350, traditional font 4888 (KSC5601)	7238 characters
Japanese S-JIS (JIS, EUC)	6353 characters
Japanese Hiragana/Katakana	169 characters
HONGKONG extra fonts 1	727 characters
HONGKONG extra fonts 2	3049 characters
HONGKONG extra fonts 3	4335 characters
English, digit, symbol	94 characters
Europe (Deutsch, Italian, French, Spanish, Portugal, German)	158 characters
Arabic	58 characters
Hebrew	50 characters
Thailand	2002 characters
Gesture	10 characters

2.4 Application

PDA Like Hand Held Device

Data Bank Like Hand Held Device

Educational Module

Educational Toy

Learning Product

Hand Writing Learning Product

Security Identification (Hand Writing Signature)

Others

3 TouchPanel Interface

3.1 GetHWRLibVersion

API Name		GetHWRLibVersion
Function Description		Get HWR LibVersion
Header File	C	HWR.h
	ASM	
Syntax	C	UINT GetHWRLibVersion (void)
	ASM	
Parameters		N/A
Return Values		Ver

3.2 DrvTPClose

API Name		HRWInit
Function Description		Initial for HWR
Header File	C	HWR.h
	ASM	
Syntax	C	void HRWInit (void)
	ASM	
Parameters		Open the HWR Module
Return Values		N/A
Remarks		N/A

3.3 HWRSetMode

API Name		HWRSetMode
Function Description		select recognition mode
Header File	C	HWR.h
	ASM	
Syntax	C	Void HWRSetMode (int Mode)
	ASM	
Parameters		ENGLISH（英文）,OFFENUSEDCHI（汉字）,DIGIT（数字）
Return Values		N/A
Remarks		N/A

3.4 HWRClose

API Name		HWRClose
Function Description		close HWR Module

Header File	**C**	HWR.h
	ASM	
Syntax	**C**	void HWRClose ()
	ASM	
Parameters		N/A
Return Values		N/A
Remarks		

3.5 HandWriteRecognition

API Name		**HandWriteRecognition**
Function Description		Hand Write Recognition Process
Header File	**C**	HWR.h
	ASM	
Syntax	**C**	void HandWriteRecognition(unsigned *str)
	ASM	
Parameters		N/A
Return Values		the recognition result = str[ ]; the end of str[] is 0
Remarks		

3.6 HWRBlockDisplayON

API Name		**HWRBlockDisplayON**
Function		Allow Handwriting Display
Description		avoid the second entrance the lib of graphic.
Header File	**C**	HWR.h
	ASM	
Syntax	**C**	void HWRBlockDisplayON (void)
	ASM	
Parameters		N/A
Return Values		N/A
Remarks		

3.7 HWRBlockDisplayOFF

API Name		**HWRBlockDisplayOFF**
Function		Forbid Handwriting Display
Description		avoid the second entrance the lib of graphic.
Header File	**C**	HWR.h
	ASM	
Syntax	**C**	void HWRBlockDisplayOFF (void)
	ASM	

Parameters N/A

Return Values N/A

Remarks

Attachment 1:

HWR Demo Process

```
void HWRApp (void *para)
{
    int LCD_X,LCD_Y;
    int   key;
    STMsg* pmsg;
    int str[22];
    for(;;)
    {
        pmsg = SysGetMSG(hMainQ);
        switch (GetMsgType(pmsg))
        {
            case MSG_AP_RESTART:
                DrawInterface();
                HRWInit();
                SysFreeMSG(pmsg);
                break;
            case MSG_INT_TP:
            if(pmsg->Reserved==TP_UP)
            {
                key=0;
                DrvTPGetValue(pmsg,(int*)&LCD_X,(int*)&LCD_Y);
              if((C_EngButtonX0 < LCD_X)&&(LCD_X < C_EngButtonX1)&&(C_EngButtonY0 <
    LCD_Y)&&(LCD_Y < C_EngButtonY1))   //EnglishButton
                {
                  DrvSetWriteMode(C_NotPut);
                  DrvBar(C_EngButtonX0,C_EngButtonY0,C_EngButtonX1,C_EngButtonY1);
                  SysTimeDly(TPDispDelayTime);
                  DrvBar(C_EngButtonX0,C_EngButtonY0,C_EngButtonX1,C_EngButtonY1);

                  HWRSetMode(ENGLISH);
                }
                else if((C_ChiButtonX0 < LCD_X)&&(LCD_X < C_ChiButtonX1)&&(C_ChiButtonY0
< LCD_Y)&&(LCD_Y < C_ChiButtonY1)) //Chinese button
                {
                  DrvSetWriteMode(C_NotPut);
                  DrvBar(C_ChiButtonX0,C_ChiButtonY0,C_ChiButtonX1,C_ChiButtonY1);
```

```
                SysTimeDly(TPDispDelayTime);
                DrvBar(C_ChiButtonX0,C_ChiButtonY0,C_ChiButtonX1,C_ChiButtonY1);
                HWRSetMode(OFFENUSEDCHI);
            }
            else if((C_NumButtonX0 < LCD_X)&&(LCD_X < C_NumButtonX1)&&(C_NumButtonY0 < LCD_Y)&&(LCD_Y < C_NumButtonY1))  //Number Button
            {
                DrvSetWriteMode(C_NotPut);
                DrvBar(C_NumButtonX0,C_NumButtonY0,C_NumButtonX1,C_NumButtonY1);
                SysTimeDly(TPDispDelayTime);
                DrvBar(C_NumButtonX0,C_NumButtonY0,C_NumButtonX1,C_NumButtonY1);
                HWRSetMode(DIGIT);
            }
            else if((C_EscButtonX0 < LCD_X)&&(LCD_X < C_EscButtonX1)&&(C_EscButtonY0 < LCD_Y)&&(LCD_Y < C_EscButtonY1)) //Number Button
            {
                DrvSetWriteMode(C_NotPut);
                DrvBar(C_EscButtonX0,C_EscButtonY0,C_EscButtonX1,C_EscButtonY1);
                SysTimeDly(TPDispDelayTime);
                DrvBar(C_EscButtonX0,C_EscButtonY0,C_EscButtonX1,C_EscButtonY1);
                HWRClose();       //if no use HWR, must close it
                SysSetNextAP(TSK_CALENDAR);
                SysSendMSG( hDispatchQ,MSG_AP_SUICIDE,0);
            }
        }
        else if (pmsg->Reserved==HWRREADY)
        {
            HandWriteRecognition(str);
            HWRBlockDisplayOFF();          //set word display flag and text data move finish flag
           DrvClearArea(C_WordBlockX0,C_WordBlockY0,C_WordBlockX1,C_WordBlockY1);
            DrvSetFont(C_GB12Font);
            DrvOutTextXy_Expand(C_WordBlockX0,C_WordBlockY0,str);      //show her result text
            HWRBlockDisplayON();           //clear word display flag
        }
            SysFreeMSG(pmsg);
            break;
```

```
                case MSG_AP_RQEXIT:
                    SysSendMSG( hDispatchQ,MSG_AP_ACKEXIT,0);
                    SysFreeMSG(pmsg);
                    break;
                default:
                    SysFreeMSG(pmsg);
                    break;
            }
        }
}

void DrawInterface()
{
    DrvClearDevice();
    DrvSetFont(C_GB12Font);
    DrvRectangle(HWR_Block1X0,HWR_Block1Y0,HWR_Block1X1,HWR_Block1Y1);
    DrvRectangle(HWR_Block2X0,HWR_Block2Y0,HWR_Block2X1,HWR_Block2Y1);
    DrvRectangle(C_TextWinX0-1,C_TextWinY0-1,C_TextWinX1+1,C_TextWinY1+1);
    DrawAllButtons();
    DrvSetLineStyle(C_SolidLine,0x0,C_LineNormWidth);
}
```