**راه اندازی محیط توسعه AS600-mini**

جهت راه اندازی AS600-mini ابتدا بایستی SDK آن را بر روی ویندوز راه اندازی نمود.

ساختار SDK به شرح ذیل می باشد:

C:\AS600mini\_SDK

|---- APP

|--- OBJ

|---- SYSOBJ

|---- DOC

|---- INC

|---- LIB

|---- UTIL

|---- ZA9Sign

setenv.bat

AS600 mini-Quick Reference (v2.0).pdf

APP: فایل های سورس برنامه نمونه به همراه Makefile آن.

OBJ: فایل های Object فایل های Source که توسط کامپایلر تولید می شود.

SYSOBJ: فایل های Object سیستمی که به همراه SDK عرضه می شود (نبایستی حذف شود).

DOC: مستندات کامل SDK در قالب PDF.

INC: فایل های Header مربوط به SDK

LIB: کتابخانه های عرضه شده به همراه SDK

UTIL: Toolchain و سایر ابزارهای عرضه شده به همراه SDK

ZA9Sign: ابزار مربوط برای امضای باینری نهائی

setenv.bat: Batch fileی که جهت تنظیم نمودن متغیرهای محیطی مورد نیاز برای کار با SDK به کار می رود.

AS600 mini-Quick Reference (v2.0).pdf: فایل مستندات راه اندازی اولیه دستگاه و SDK که شامل چند اشکال جزئی می باشد که در ادامه به تشریح آن خواهیم پرداخت.

**مراحل نصب SDK (راهنمای اصلی SDK دارای تصویر می باشد)**

۱. ابتدا با مراجعه به C:\AS600mini\_SDK\UTIL\GNUARM اقدام به نصب Tool Chain جهت کامپایل برنامه ها می نمائیم.

۲. چنانچه از ویندوز x64 استفاده می نمائید پوشه GNUARM را از مسیر “Program Files (x86)” به پوشه “Program Files” منقل نمائید.

۳. اقدام به تغییر نام پوشه GNUARM واقع در “Program Files” به GNUARM.410 نمائید.

۴. چنانچه از ویندوز Vista و یا بالاتر استفاده می نمائید اقدام به رونویسی فایل “C:\Program Files\GNUARM.410\bin\ cygwin1.dll” با فایل “C:\AS600mini\_SDK\UTIL\GNUARM\cygwin1.dll” نمائید.

۵. با مراجعه به cmd.exe و وارد شدن به پوشه C:\AS600mini\_SDK دستور ذیل را اجرا نمائید:

C:\AS600mini\_SDK>setenv gcc

**بردن دستگاه به System Operation Mode**

تمامی مراحل مطابق با راهنمای اصلی دستگاه می باشد. تنها نکته نگه داشتن کلید Cancel قبل از روشن نمودن دستگاه تا نمایان شدن پیغام می باشد که به اشتباه در راهنما special three keys ذکر شده است.

**فرستادن فایل باینری به دستگاه با استفاده از COM**

۱. بر خلاف اظهارات راهنمای دستگاه مبنی بر اتصال کابل RS232 به پورت COM2 جهت نصب فایل باینری برنامه بایستی کابل را به پورت مشخص شده در راهنما به شماره COM0-1 نمائید.

۲. این بخش در راهنما ذکر نشده است: با مراجعه به برنامه “C:\AS600mini\_SDK\UTIL\DSS\DSS Command Line (UART)\DSSDLL20.exe” و اجرای دستور ذیل اقدام به آماده سازی فایل کامپایل شده برنامه جهت ارسال می نمائیم:

DSSDLL20 APP\_sig.bin -Cn

که n شماره پورت COM متصل به دستگاه در کامپیوتر شما می باشد. مثلا ۱ برای COM1، ۲ برای COM2 و ....

۳. مطابق راهنما در صفحه ۱۵ دستگاه را به مد System Operation ببرید.

۴. در این مرحله دستگاه اقدام به دریافت و Flash فایل باینری نموده و بصورت خودکار Reboot می نماید.‍

**فرستادن فایل باینری به دستگاه با استفاده از Ethernet**

تمامی مراحل مطابق راهنما تصویری SDK می باشد.

**تست ماژول های دستگاه**

تمامی سورس کدهای برنامه بایستی در پوشه C:\Users\Administrator\Desktop\SDK.Release\APP قرار گیرد. خوشبختانه از قبل یک نمونه برنامه در داخل این پوشه قرار داده شده است که از اکثر توابع SDK استفاده نموده است. همچنین به ازای هر فایل سورس کدی که به پروژه اضافه یا از آن حذف می نمائید بایستی فایل makefile واقع در این پوشه را ویرایش نمائید. برای مثال می خواهیم فایلی با نام NewFile.c را به پروژه اضافه نمائیم، با پیدا نمودن بخش های مربوطه در makefile تغییرات ذیل را اعمال می نمائیم:

# --- program objects ---

OBJECTS = OBJ\GlobalVar.o

OBJECTS += OBJ\EMVCB.o

OBJECTS += OBJ\UTILS.o

OBJECTS += OBJ\TEST.o

OBJECTS += OBJ\MAIN.o

OBJECTS += OBJ\NewFile.o

# --- application program source codes ---

OBJ\GlobalVar.o: GlobalVar.c

$(CC) $(CFLAGS) $< -o $@

OBJ\EMVCB.o: EMVCB.c

$(CC) $(CFLAGS) $< -o $@

OBJ\UTILS.o: UTILS.c

$(CC) $(CFLAGS) $< -o $@

OBJ\TEST.o: TEST.c

$(CC) $(CFLAGS) $< -o $@

OBJ\MAIN.o: MAIN.c

$(CC) $(CFLAGS) $< -o $@

OBJ\NewFile.o: NewFile.c

$(CC) $(CFLAGS) $< -o $@

یکی از فایل های پروژه با نام UTILS.C با فراهم نمودن یک Wrapper بر روی API اصلی میزان Abstraction قابل قبولی را برای برنامه های شما فراهم می نماید. توصیه می شود به جای استفاده مستقیم از API تا حد ممکن از توابع موجود در این فایل که با پیشوند UT\_ شروع می شوند، استفاده نمائید.

برای مثال برای چاپ رشته ای بر روی LCD به جای API api\_lcd\_putstring می توان از تابع UT\_PutStr به شکل زیر استفاده نمود:

UT\_PutStr( 0, 0, FONT0, strlen(msg\_1\_LCD), msg\_1\_LCD );

کد تابع UT\_PutStr جهت مقایسه:

void UT\_PutStr(UCHAR row, UCHAR col, UCHAR font, UCHAR len, UCHAR \*msg)

{

UCHAR sbuf[3];

UCHAR dbuf[80];

sbuf[0]=row;

sbuf[1]=col;

sbuf[2]=font;

dbuf[0]=len;

memmove(&dbuf[1], msg, len);

api\_lcd\_putstring(ut\_dhn\_lcd, sbuf, dbuf);

}

لطفا برای مشاهده لیستی از کارهای معمولی که می توان با استفاده از SDK انجام داد به این فایل مراجعه نمائید.

نمونه برنامه ای که که با استفاده از توابع موجود در UTILS.C اقدام به چاپ متنی با سه قلم مختلف بر روی LCD می نمایند:

void TEST\_LCD( void )

{

UCHAR row; // row #

UCHAR col; // column #

UCHAR pattern[] = {"012345678901234567890123456789"};

UT\_BlackenScreen();

UT\_WaitTime(100); // wait 1 sec

UT\_ClearScreen();

UT\_WaitTime(100); // wait 1 sec

// test FONT0

col = 0;

for( row=0; row<8; row++ )

UT\_PutStr( row, col, FONT0, 30, pattern ); // 240x128 (8x16)

UT\_WaitTime(100); // wait 1 sec

UT\_ClearScreen();

// test FONT1

col = 0;

for( row=0; row<8; row++ )

UT\_PutStr( row, col, FONT1+attrISO, 30, pattern ); // 240x128 (8x16)

UT\_WaitTime(100); // wait 1 sec

UT\_ClearScreen();

// test FONT2

col = 0;

for( row=0; row<4; row++ )

UT\_PutStr( row, col, FONT2+attrISO, 15, pattern ); // 240x128 (16x32)

UT\_WaitTime(100); // wait 1 sec

}

جهت مشاهده نمونه کدهای بیشتر به فایل TEST.C عرضه شده به همراه SDK مراجعه نمائید.

**مشکلات راه اندازی و راه حل آن ها**

در ابتدای راه اندازی دستگاه با مشکلاتی روبرو شدیم که تقریبا اکثریت ان ها حل شده یا راه حل آن ها را پیدا نمودیم. برخی از مشکلات با ارائه آپدیت برای SDK از سوی سازنده حل و برخی دیگر مشکلات سخت افزاری بود که مرتفع گردیده است. برای مثال آپدیت 20140421 مربوط به حل مشکلات MSR, SCR and Contactless Reader می باشد. البته Contactless Reader یک مشکل سخت افزاری دارد که مربوط به کابل اتصال آن می باشد و بایستی ترتیب کابل های اتصال مشابه تصویری که در ادامه ارائه خواهیم نمود باشد. آپدیت های 20140512 و 20140526 به درخواست ما جهت حل مشکلات فارسی سازی ارائه شده است که ما را قادر به ارائه راه حلی جهت فارسی سازی دستگاه می نماید.

**حل مشکل تداخل با کامپایلرهای تعیین شده در PATH:**

در حال پیش فرض پس از اجرا نمودن فایل setenv.bat چنانچه کامپایلر MinGW دیگری (مثلا همراه Qt) در PATH وجود داشته باشد با فایل های Toolchain همراه SDK تداخل ایجاد می شود. جهت رفع این مشکل، کافیست تمامی خطوط واقع در فایل setenv.bat را که متغییر PATH را تغییر می دهند ویرایش بنمائیم. برای مثال:

set PATH=%PATH%;C:\Program Files\GNUARM.410\bin

را به

set PATH=C:\Program Files\GNUARM.410\bin;%PATH%

تغییر می دهیم. در واقع با جابجا نمودن متغییر %PATH% به انتهای دستور اولویت را به کامپایلر همراه SDK برنامه می دهیم. توجه فرمائید که این تغییر بایستی در تمامی خطوط مشابه اعمال شود.

**حل مشکل ماژول MSR**

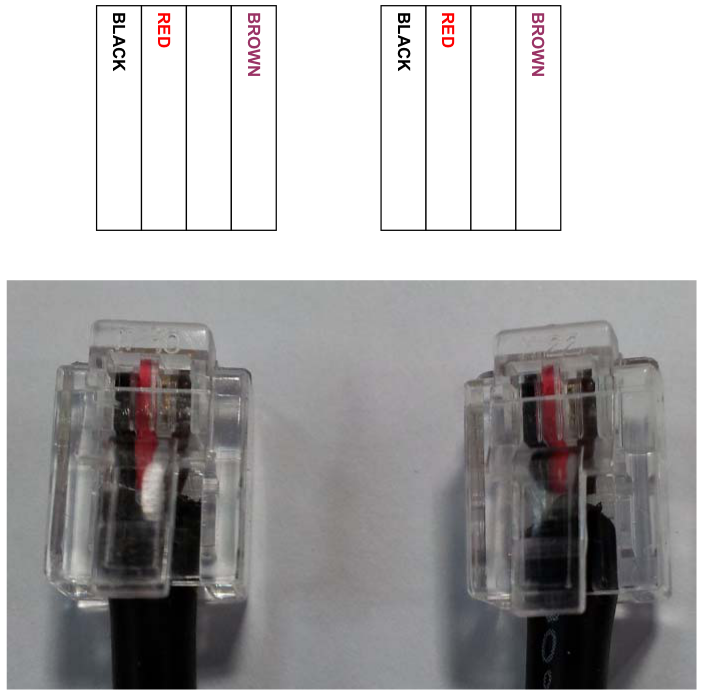
آپدیت SDK با Patch نسخه 20140421

**حل مشکل ماژول SCR**

آپدیت SDK با Patch نسخه 20140421

**حل مشکل ماژول Contactless Reader**

1. آپدیت SDK با Patch نسخه 20140421
2. تغییر کابل اتصال دستگاه به ترتیب ذیل:



**ماژول Modem**

متاسفانه به دلیل عدم دسترسی به سرویس دهنده HDLC امکان تست کامل مودم و ارسال و دریافت اطلاعات برای ما فراهم نشد. اما امکان شماره گیری وجود داشت.

**حل مشکل فارسی سازی**

جهت رفع این مشکل با سازنده دستگاه تماس حاصل نمودیم. با توجه به عدم ارائه راه حل مناسبی توسط سازنده، خودمان دست به کار شده و ابزاری جهت تبدیل فونت های Mono-spaced TrueType به C-Bitmap طراحی نمودیم (یک نمونه از این فونت با پشتیبانی از فارسی در پوشه resources/fonts واقع در پوشه سورس برنامه گنجانده شده است). البته این ابزار عدم نیاز به دست کاری دستی فونت ها را صد در صد حل نمی کند و قلم ها پس از تولید بایستی مقداری به اصطلاح Fine Tune شوند. شما همچنین می توانید از فونت های معمول TTF هم استفاده نمائید، اما با توجه به عدم هم فاصله بودن کاراکترها در این نوع فونت ها بدلیل نیاز به Fine Tune دستی بیش از حد اصلا توصیه نمی شوند. سپس از سازنده دستگاه درخواست نمودیم تا راه حلی جهت ذخیره کاراکترهای بیش از ۱۲۸ عدد (ASCII) ارائه نماید. این تغییرات به خوبی در فایل های به اصطلاح Release Note هر آپدیت مستند شده است. نکته قابل توجه استفاده از یک فونت مشترک برای Printer و LCD پس از ارائه این آپدیت ها می باشد (بر خلاف گذشته که هر ماژول روش ایجاد و استفاده فونت خود را داشت). سپس ابزار مذکور را توسعه داده و ابزار دیگری جهت تولید متن فارسی یونیکد نمایشی تولید نمودیم (توجه! این ابزار ممکن است هنوز مشکلاتی داشته باشد) که به خوبی توانائی تولید متن bi-directional فارسی و انگلیسی را دارا می باشد. می توان با استفاده از الگوریتم استفاده شده در این ابزار کد C مشابه را جهت استفاده مستقیم در دستگاه تولید نمود.

جهت Build نمودن این ابزار که با استفاده از C++/Qt تولید شده است به نیازمندی های ذیل نیازمندیم:

Boost

fftw3

freetype

Magick++ (GraphicsMagick or ImageMagick)

lcms

lcms2

libiconv

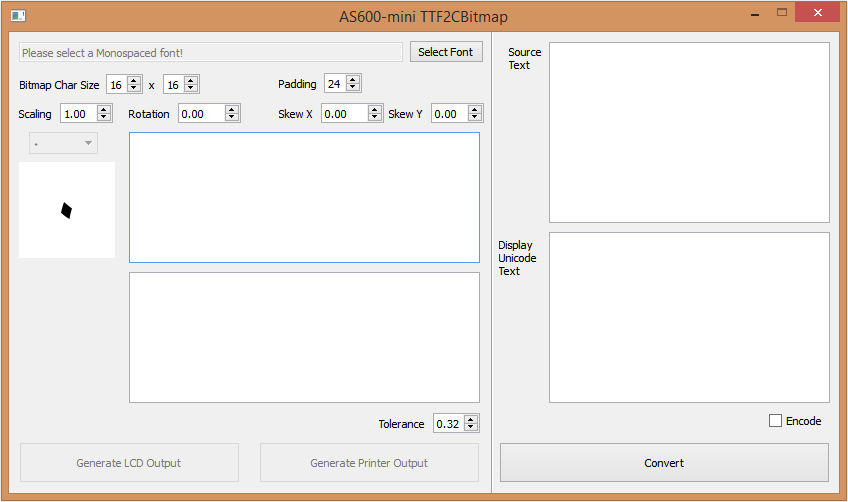
libpng

zlib

مراحل کامپایل و نصب این کتابخانه ها برای ویندوز با استفاده از MinGW در پوشه \_deps-win32-build-instructions واقع در پوشه نصب برنامه به تفصیل توضیح داده شده است. چنانچه از ‌‌BSD یا GNU/Linux استفاده می نمائید، بایستی ابتدا از Package Manager خود اقدام به نصب آنها نمائید. در سیستم عامل Mac OS X می توانید از Fink, MacPorts, و یا Homebrew استفاده نمائید.

در پوشه سورس برنامه و در \_deps-win32 نسخه ای کامپایل شده از تمامی این پیش نیازها توسط MinGW-Builds 4.8.2 که با Qt 5.3.0 عرضه می شود به صورت Debug و Release موجود می باشد.

نمائی از نرم افزار تولید فونت و متن فارسی:



**نحوه تبدیل فونت TTF به C-Bitmap**

ابتدا تمامی کاراکترهای فارسی (شامل اعداد و حروف) موردنیاز را در تمامی حالات در آرایه ای با نام s\_glyphs تعریف نموده ایم. بعدا در زمان صدا شدن Constructor کلاس MainWindow تمامی این کاراکترها به همراه کاراکترهای انگلیسی به Vectorی با نام m\_glyphs جهت تولید فونت C اضافه خواهند شد. پس از اعمال تغییرات توسط کاربر (از قبیل سایز فونت مثلا 8x16 یا 16x16، بزرگنمائی، چرخش، اعوجاج، فاصله، میزان خطا در تبدیل رنگ پیکسل ها (Tolernace) و .....) با کلیک بر دکمه های Generate LCD Output و Generate Printer Output به ترتیب کد توابع on\_lcdOutputPushButton\_clicked و on\_printerOutputPushButton\_clicked اجرا شده و کد کاراکترها در کادر بالا و خود کاراکترها در کادر پائین چاپ خواهند شد. توجه نمائید که LCD Output فقط برای کاراکترهای انگلیسی مناسب است و در آپدیت جدید می توان از Printer Output برای هر دو ماژول LCD و Printer استفاده نمود.

کد موجود در هر دو تابع ذکر شده بسیار شبیه به یکدیگر می باشد. ابتدا پوشه ای در مسیر Temp ایجاد شده و سپس برای هر کاراکتر موجود در وکتور m\_glyphs یک تصویر PNG ایجاد خواهد شد. پس از تغییر سایز فایل PNG به سایز مقصد فونت مورد نظر، اقدام به خواندن تک تک نقاط تصویر نموده، سپس با توجه به مستندات SDK بیت های مورد نیاز را تولید می نمائیم. پس از تولید بیت ها مطابق راهنمای SDK کد Hex را برای هر کاراکتر تولید نموده و در خروجی چاپ می نمائیم.

**نحوه تبدیل متن فارسی به یونیکد نمایشی**

ابتدا پنج آرایه به نام های s\_alphanumGlyphs (کاراکترهای فارسی معمولی) و s\_detachedGlyphs (شکل غیرچسبان کاراکتر های فارسی) و s\_initialGlyphs (شکل کاراکترهای چسبان آغازین) و s\_medialGlyphs (شکل کاراکترهای چسبان از هر دو طرف یا به اصطلاح میانی) و s\_finalGlyphs (شکل کاراکترهای چسبان انتهائی) تعریف می نمائیم. چنانچه کاراکتری شکل چسبان نداشته باشد به جای آن مقدار ماکروی GLYPH\_IS\_NOT\_DEFINED قرار خواهد گرفت. در زمان صدا شدن Constructor کلاس MainWindow یک نقشه از کاراکترها در یک C++ Map که کلید آن مقادیر آرایه s\_alphanumGlyphs بوده و مقدار آن ۴ شکل کاراکتر های فارسی در قالب یک C++ Vector با ۴ خانه می باشد تولید خواهد شد. از این به بعد m\_glyphsMap مرجع ما برای قضاوت در مورد شکل حروف تولید شده خواهد بود.

تمامی کد تولید متن فارسی یونیکد نمایشی در تابع ConvertToDisplayUnicode می باشد که با فشردن دکمه Convert نسخه ای از این تابع با پارامترهای مورد نظر فراخوانی می شود. ورودی های این تابع شامل متن اصلی، متغیری از نوع std::wstring جهت نگهداری متن تولید شده و یک پارامتر Boolean جهت تعیین اینکه متن تولید شده به شکل خام باشد یا به شکل C Hex می باشد.

جهت تعیین نوع کاراکتر های خروجی یک Enum Class تعریف نموده ایم با مقادیر ذیل :

enum class Glyph : unsigned char {

None = 255, // none

Detached = 0, // e.g. ﺏ

Initial = 1, // e.g. ﺑ

Medial = 2, // e.g. ﺒ

Final = 3, // e.g. ﺐ

Combined = 4 // e.g. ﻼ

};

سپس در ادامه تابع با یک حلقه حرف به حرف تا انتهای متن پیش میرویم و با توجه به موقیعت حرف که آخر، ابتدا و یا وسط می باشد و نوع آن (مثلا آیا نوع چسبان دارد؟ اگر دارد از چه نوعی است؟) اقدام به تولید متن خروجی می نمائیم. اگر Check Box با نام Encode چک خورده بود هم متن خروجی را به شکل C-Hex تبدیل می نمائیم.