Senstickファーム実装概要書

作成者: 上原 昭宏

Revision 1.00

作成日: 2016年5月31日

目次

1. はじめに

これは、SenstickのNordic Semiconductor社のnRF51822に内蔵されたCortex-M0プロセッサで動作するファームウェアの実装概要書です。

ファームウェア開発者が読むことを想定して、ソースコードから読み取れない情報を補い、また開発の手助けになる雑多な情報をまとめます。ファームウェア仕様書を読んでいることを想定します。またnRF51822の開発環境など一般的な開発知識を前提取ります。

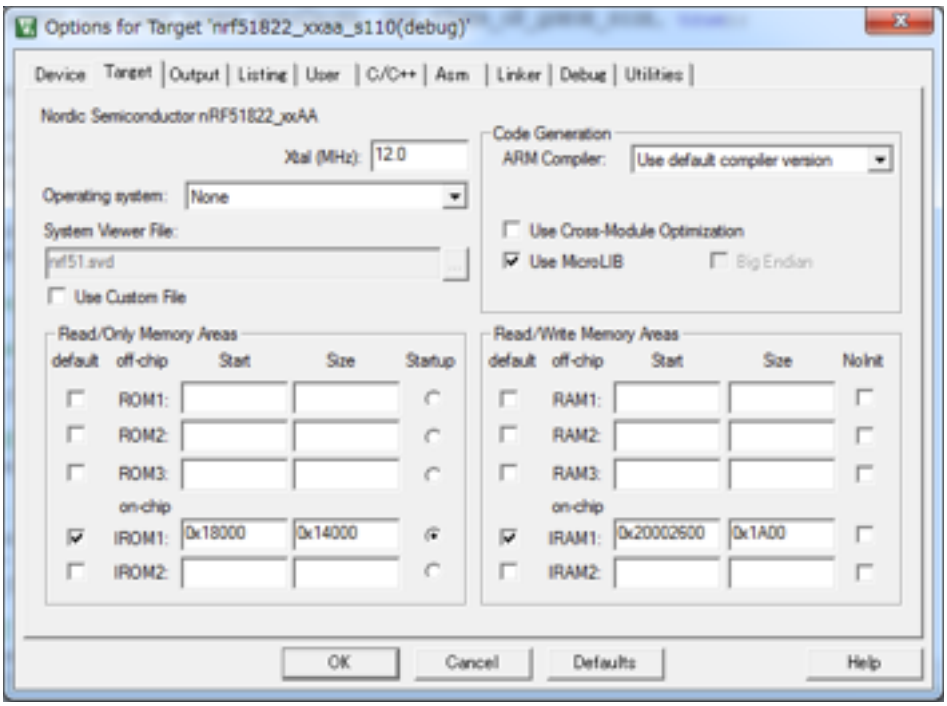
この仕様書のバージョン番号は、この仕様書に対応するファームウェアのバージョン番号と同じです。

1.1. ソースコード

1.1.1. 開発環境

開発環境はKeil MDK-ARM, uVision V5.17.0.0。

ターゲット設定は、nRF51822_xxAA。ターゲットの設定は:



ソフトウェアデバイスのリビジョンは 8.0、nRF51 SDKのバージョンはSDK10を使用している。

RAMのスタートアドレスが通常は0x20002000だが、この設定は0x20002600と0x600大きい。BLEのサービスの数が多く、そのために必要なメモリ領域をソフトウェアデバイス側に与えている。このメモリ領域の設定は、ソースファイル ble_stack.c でソフトウェアデバイス側に使用するメモリ領域として設定している。

ソフトウェアデバイスの消去や書き込みは、nRFgo Studioで行う。

このファームウェアは、J-Link liteでprintfデバッグをしている。シリアル端子がないので、J-Link Real Time Viewerというのを使って、SWI経由で吸い上げている。これはuVisionのデバッグビルドを選択すると、ファームに自動的に入る。ビューアのプログラム名は、J-Link RTT Viewer。これはJ-Link

Liteと接続するので、nRFgo Studioとは排他利用。どっちかを使うときは、どっちかを閉じておく。もしも両方開くと、JTAGがなんとかというダイアログが出るが、キャンセルして閉じて、どっちかのアプリを閉じればいい。

1.1.2. リビジョン管理

ソースコードは、Gihtub上でリビジョン管理下にある。URL https://github.com/ubi-naist/SenStick-firmware

ブランチ名は次の2つで運用。

- master
- release

ブランチのルールは適当。とりあえず、バグがあるとローカルの適当なブランチで作業して、動作確認をすればmasterにマージしてGithubにpushする。

リリースになったならば、releaseブランチにマージする。製造側など他社にファイルを渡す場合はreleaseブランチを使う想定で。

1.1.3. ファイル構成

トップのフォルダ構成:

- binary
  - リビジョン名のフォルダのしてに、ブートローダおよびファームウェアのイメージファイルがあります。
- bootloader
  - ブートローダのソースファイルです。
- docs
  - ドキュメントおよびデータシートがあります。
- firmware
  - Senstick本体のファームウェアです。
- ios
  - iOSのSDKおよび動作確認用ビューアアプリがあります。

。

axf形式のファイルをhex形式に変換するには、MDK-ARMにあるツールで、 fromelf --i32 --output o.hex axf_file_name とします。パスは適当に。

bootloaderはnRF51のSDK10のデフォルトのブートローダに、1.ボタンを押していてもOTAに入らない、2.水晶のクロックをデフォルトの16MHzではなくブレイブリッジのモジュールが使っている32MHzに変更、3. LEDのピン番号をSenstickの回路図にあわせて設定、の変更を指定ます。

firmwareフォルダにある、senstick.xcodeproj は、ファームウェアのファイル群をまとめるxcodeのプロジェクトです。勝手な開発環境として、ファームウェアのソースコード編集はMacのXcodeで行い、仮想

環境のWindowsのuVisionでビルドしています。そのためソースコードのタブや改行コードは、Xcodeのデフォルト値となっています。エディタとしての好みでしかないので、不便があるときはlintツールなどで、適当に変更してください。

docsはこのドキュメントを含む一連のドキュメントと資料を置いています。参考のためにnRF51822およびネットから適当にダウンロードした周辺ICの資料を置いています。

1.1.4. ビルドおよびリリース手順

firmware以下のプロジェクトから、ビルドしてできるaxfファイル、senstick_xxaa_s110_debug.axfを、binaryフォルダにコピーします。ソフトウェアデバイス8系では、hexファイル単体ではDFUできず、ファイルのチェックサムなどを収めたファイルを作りそれをアーカイブしたzipファイルにしなければなりません。

binaryフォルダの下にある、create_dfu_zip_batを使って、create_dfu_bat.zip axfファイル名、とすればdfu.zipができます。このファイルを適当なファイル名に変更します。batファイルが使うコマンドにパスが通るように、PATH設定します。

DFUではファームウェアのバージョン情報も入れることができます。アプリケーションは使わないので任意値でよいのですが、設定するならば、batファイルの —aplication-versionの引数を変更します。詳細はSDKのドキュメントを参照してください。

1.1.4.1. OTAイメージの作成手順

Nordic Semi.社のサイトからダウンロードしたMaster Control Panelをインストールします。ここでは3.10系をインストールしています。

Windowsの環境変数の設定で、変数”PATH"を以下のように設定します。

C:\Program Files (x86)\Nordic Semiconductor\Master Control Panel\3.10.0.14\nrf;C:\Keil_v5\ARM\ARMCC\bin

Firmware/Binaryに、OTAファイルイメージにしたいファームウェアのaxfファイルを作成します。Windowsからみてファームウェアがネットワークドライブにあるならば、そのフォルダに移動します。例えば、Z:\フォルダにマウントされているならば、”cd /d z:¥”としてドライブを移動します。

.batファイルを開いてて、ファームウェアのバージョン番号を設定します。いまファームウェアのバージョンが1.4.0であれば、”--application-version 0x0100 ”、とします。

OTAファイルを作成します。”create_dfu_zip.bat senstick_xxaa_s110_debug.axf”

1.1.5. ファームウェア内部の解説

デバッグビルドは、”DEBUG”を定義します。Real time viewerを使う、使わないの定義は、debug_rtt.hで行っています。メッセージ用に内部で128バイトの固定バッファを確保しています。

1.1.6. サンプリング周期の設定

センサーのデータ取得と、フラッシュへの書き込みは、別のコンテキストで実行される。センサデータの取得は、TIMER2割り込み、フラッシュへの書き込みは、スケジューラで実行されるタスクで処理される。この処理の間のデータやりとりは、mailboxを使う。

フラッシュの書き込みは0.5ミリ秒typ. 1.5ミリ秒 max 。また、4kセクタ消去は30ミリ秒 typ. 120ミリ秒 max。消去の最大値は、2.7V 85℃での値。

フラッシュ消去処理中のセンサデータを保持できるだけのメールボックスが必要である。フラッシュのワーストケースを模擬するために、spi_slave_mx25_flash_memory.c 428行目に90ミリ秒のウエイト文を入れている。

ワーストケース (セクタ消去の本来の処理時間30ミリ秒+ウエイト時間90ミリ秒)

- 加速度センサーのみ、周期10ミリ秒 ー> 最大のキューの深さ 14
- 加速度、角速度、磁場の3センサー、周期10ミリ秒 ー> 3秒程度でファームが終了(キューの食いつぶし)
- センサ3つ、周期20ミリ秒 ー> 数秒でファームが落ちる
- センサ3つ、周期40ミリ秒 ー> OK

通常動作

- センサ3つ、10ミリ秒 ー>数秒で落ちる
- センサ3つ、20ミリ秒 ー> キューの深さ18
- センサ3つ、30ミリ秒 ー>キューの深さ17

2.改訂履歴

- 2016年5月31日 初版。