3軸加速度センサーの組み込み

前々回、作ったパラメータ通信機能と、パラメータの書出し、読込みプログラムに、3軸加速度センサーMMA8452Qの ソースを組み込みます。 ただ、ソースを 移すだけなら、すぐできますが、パラメータに設定しているサンプルレイトで、加速度データを 取り込み出来るようにします。

それと、パソコン側からコマンドで トリガON状態に設定して パラメータに設定している収録秒数の間、SDカードに 加速度データを データファイルとして書き出す事を 目標とします。
SDカードに書き込む事が 出来れば 手動のロガーとして、動作の確認は 出来ます。
データを SDカードに収録後、SDカードを パソコンに 持って行き データを確認するための グラフ表示の ビューアソフトを 作る必要があります。(作る事は出来ますが、これがちょっと面倒) で、パソコン側のビューアまで含めて 1週間以内に作るのは、ちょっと厳しいと思いますので、PC側ビューアは 間に合わなかったら、ごめんなさい。 という事で 作業を進めます。

あとデータのフォーマットを どうするのか検討する必要があります。 収録データを 小さく出来るのは、センサから入ってくる生データを使う方が データファイルを 小さく出来ます。

MMA8452Qの場合、12bitのバイナリデータが ＸＹＺ の 3チャネル分と なります。 データ型としては 2byte 整数×3で 記録する事になります。
単位変換の演算を行うと 単精度 浮動小数点データ 4byte×3 と なります。

加速度センサーのようなデータの場合、トリガ判定機能で イベントが 発生した際、記録する事になりますが、以前お話しましたが、プリトリガ機能といって、イベントが発生する、5秒か10秒前のデータから 収録したいという機能を要求されるので、最大 10秒の遅延を作るデータ用のリングバッファを用意する必要があります。 最大 100Hzサンプリングで、4チャネル、10秒の遅延を作るのは 100×4×2×10で、8kbyte以上の リングバッファが 必要です。(2byte整数の場合)、単精度浮動小数点であれば、リングバッファが 倍のサイズに なります。 ESP32であれば 倍のサイズでも、RAMメモリに入るとは思いますが、今回は、2byte整数のリングバッファを構築します。
トリガの判定も 整数を想定しているので この方が いいと思います。

単位変換の処理は、パソコンで 後処理的にも出来ますが、ESP32マイコンの演算処理能力が早いので、マイコン側で ファイルに書き出す直前で、単位変換の演算を行い 4byte浮動小数点データとして SDカードに 書き出す事に します。 今の SDカードは SDHC で 最大 32 Gbyteで 容量が 大きいので 全く問題ないと思います。

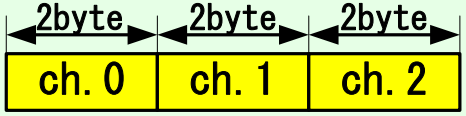
後は、パラメータ構造体に、若干 追加項目を用意する必要があります。 パラメータ構造体は、事前に予備エリアを 多数用意しているので
構造体の宣言に 多少追加が生じますが サイズは 変りません。

以上の事柄を 次のページに整理してみます。

バッファサイズ等の検討結果

MMA8452Qから取り込んだ生データ

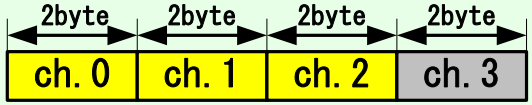
3チャネル共通で 12bitなので、1サンプル
3チャネルの　16bit（2byte整数）として扱う。



1サンプルのデータ:
計 6byte

生データ遅延用リングバッファの データ

リングバッファは 1つのデータを 2byte整数で扱い
1サンプル／4チャネルで　扱う事にする。

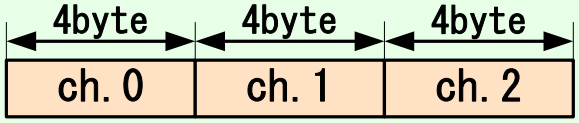


1サンプルのデータ:
計 8byte　　ch.3は　未使用

リングバッファサイズは　最大　100Hz　10秒以上
の容量が必要。
1000サンプル以上で（ 8 Kbyte以上 ） という事で
8192byteとする。

SDカード書込み時のデータ

データは　SDカード書き込み直前で　単位変換を
行い　4byte 浮動小数点データに変換する。
今回の　3軸加速度データ収録の場合は　殆どの場合
3軸を組みに 扱うと思われるので、3チャネル固定と
考える。



1サンプルのデータ:
計 12byte

全体のファイルサイズは、100Hzサンプリング 3分 で
256（ヘッダ）＋ 12×100×180 ＝　216 Kbyte に なり
ます。　やや大きい感じもしますが、100Hzサンプリン
グで　収録すれば　こんなもんです。

リングバッファとは　バッファが　輪になったイメージで、同じメモリ範囲内を　ぐるぐる回って書込み、読出しを行うバッファです。 これをメモリ上で構築するには、横にチャネル数　縦にサンプル数の２次元配列を　イメージして下さい。

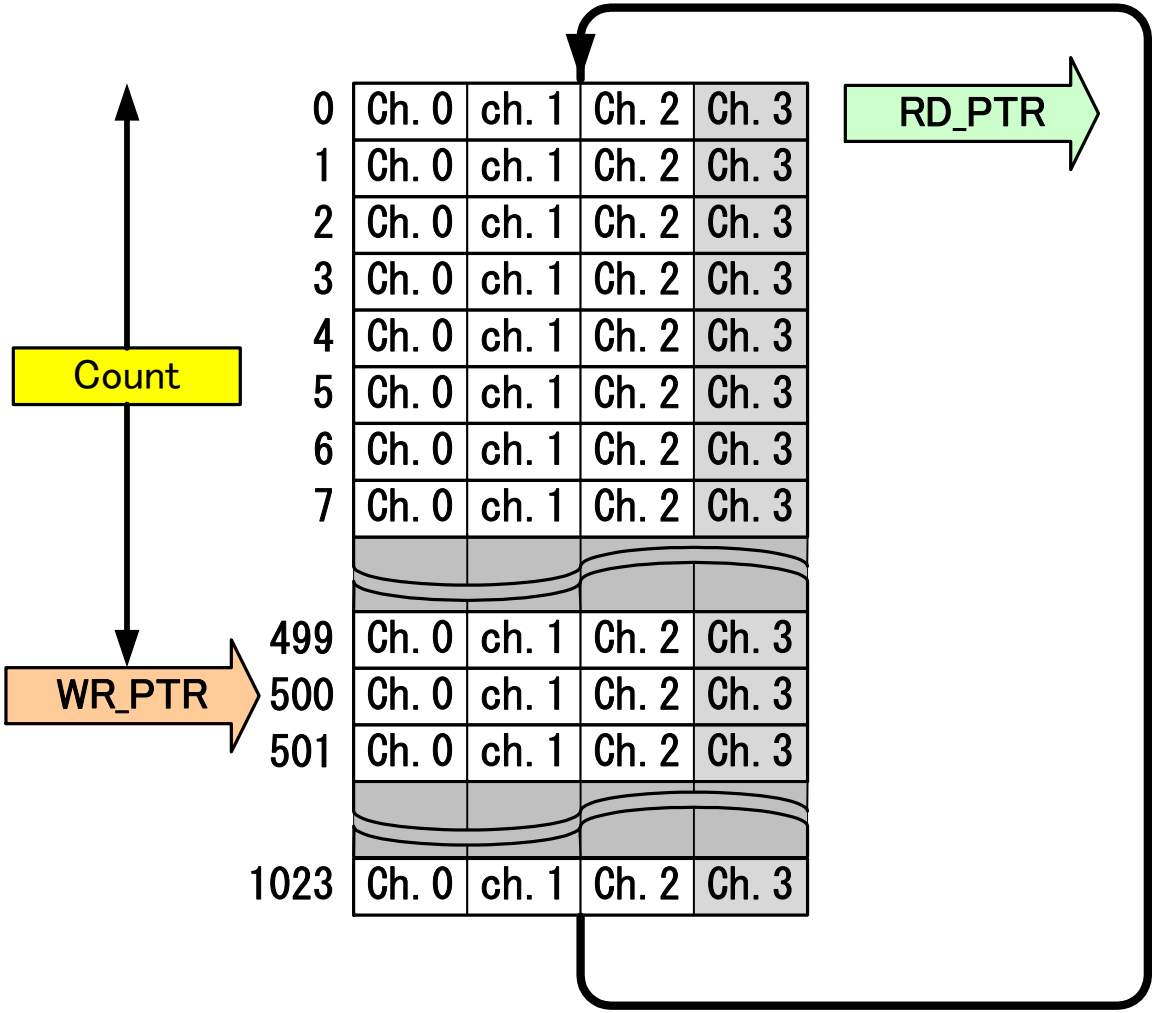
データを書込むポインタ操作で、縦の最終行まで行ったら　また先頭に戻る事により、延々ぐるぐる回るイメージを作り出します。

そして、書込みに使う ポインタと　読出しに使うポインタの２本を用意します。 あと通常は、リングバッファに　書込み後に　ポインタをインクリメントして　書込み個数を　＋１します。 リングバッファを読出し後に　ポインタを　インクリメントして、書き込み個数を －１する カウンタも　用意します。 今回のような遅延バッファの場合、定常状態になったら　書込みポインタと　読出しポインタが　常に同期して　インクリメントします。

リングバッファの　データ個数が　定常状態になったらというのは、遅延データを作るのが　目的なので、書込みポインタにより、書き込まれたデータの個数が、遅延データを作るための、サンプル数データが溜まれば、遅延用のリングバッファは　レディ状態になり、遅延データを読出す事が可能となります。 仮に　100Hzサンプリングで　5秒の遅延の場合は、リングバッファに　500サンプルの　データが　溜まれば レディ状態に　なります。 よってシステムが　最初に起動した直後は　リングバッファ内に　データが溜まって無いので　ビジー状態で　5秒間 待たされます。 5秒経過後　レディ状態で、トリガの受け付けが　可能となります。

ちょっと、イメージが　掴みずらい説明だったと思われるので、次のページで　図で 示します。

データ遅延用リングバッファのイメージ



WR_PTR は　書込み用ポインタです。
RD_PTR は 読出し用ポインタです。
Count は　サンプルデータ格納個数です。

　Countが　遅延時間に必要なデータ数蓄積出来たら　リングバッファは　レディ状態に　なります。　左の図の場合はサンプル数　500 で　レディになるイメージを　表しています。
　よって、Count が　500になるまではRD_PTRは　読み出し動作を行いません。

SDカードに記録するデータフォーマット

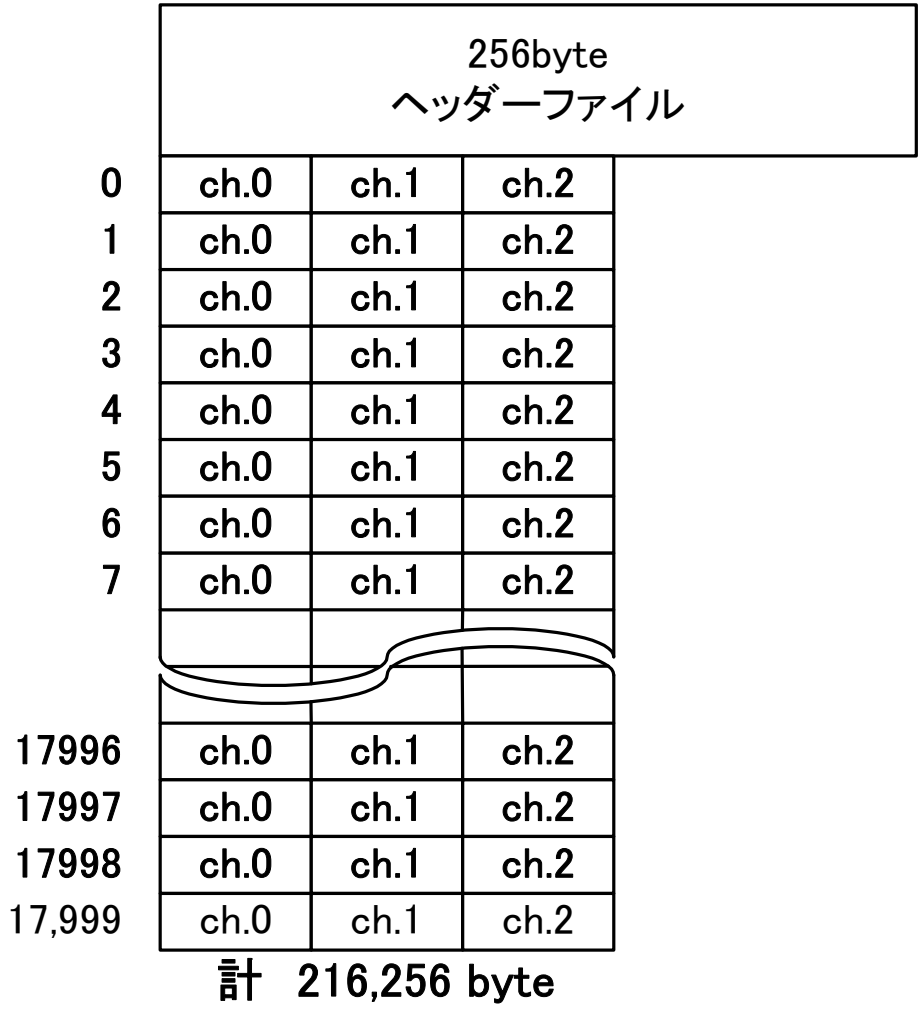
SDカードに　書き込むデータフォーマットは
先頭に 256byte の　ヘッダ情報を記録します。

その直後に　今回のMMA8452Q 3軸加速度センサーの場合、4byte Float データを　X軸 Y軸　Z軸の　3チャネルで　計 12byteを 1サンプルとして、時刻順に　記録して行きます。

記録サンプル数は、収録時間 3分で　100Hz サンプルの場合、3×60×100 で　18,000 サンプルに なります。 ヘッダー情報を含め ファイルサイズは　計　216,256 byte に　なります。

右に　図で示します。

データフォーマット



計　216,256 byte

正確な サンプルレイト生成

正確なサンプルレイト生成のため、タイマー割り込みを使用する予定で おります。

正確なサンプルレイトとは、今回の場合は3軸加速度センサから、正確な時刻の刻みでセンサのデータを 取り込む事を意味します。

で、加速度センサーの場合は 100Hzとかやや速い サンプルレイトで データを取り込み続ける必要が あります。この 100Hzの インターバルを 正確に刻み続けるために タイマー割り込みを 使用したいと考えます。

で、タイマー割込み処理の中では いくつかの制限事項があります。 まず、時間のかかる事は やってはならないという事です。 100Hzという事は インターバルは 10msです。
タイマー割込み処理の中で 10msかかる処理を 行うと システムが破綻します。
一般的に言われているのは、最悪でも タイムインターバルの 1/2 以内にする事と言われています。 インターバルが 10msという事は タイマー割り込み処理は 5ms以内に 処理を 完了しなければならない。という事です。

それと、タイマー割り込みに限らず 割り込み処理の中で 割り込み処理を 含む処理を 呼び出すと 処理が正常に実行されない。 あるいは最悪の場合 システムが 破綻する危険もあります。

シリアル通信は 割り込みの機能を利用しています。 場合によってはうまく行く場合もあるかもしれませんが、割り込み処理の中では 呼び出さない方が いいです。

場合によっては うまく行く場合も あるかもしれない。 というのは CPUのハードの話になりますが、ESP32にも CPUコアの 周辺回路として割り込みコントローラが あると思います。

割り込みコントローラには、いくつかのモードがあるので 決め付ける訳にはいかないのですが、一般的に 各割り込み要因毎に 割り込みの優先レベルを 設定する事が 出来ます。

で、且つ多重割り込みが 許可されている場合は、タイマー割り込み処理内にて タイマー割り込みの 優先レベルより 高い割り込みレベルを 使用する処理を 読み出した場合は 割り込み処理中に ネスト的に 更に高いレベルの割り込みを 受け付ける事が 出来ます。

逆にタイマー割り込み中に 低いレベルの割り込みが発生した場合は、タイマー割り込みが 終了するまで、待たされる事に なります。
で、タイマー割り込み中に タイマー割り込みより レベルの低い割り込みを使用する処理を 読み出して、その中で レベルの低い割り込み応答がある事を 待つ処理が 入っていると タイマー割り込み内にて 無限ループとなり システムが、破綻します。 CPU周辺ハードを イメージ出来ないと 分かりにくい話でしたが、割り込み処理内では、割り込みを 使用する可能性がある関数を 実行する場合、十分注意して下さい。

何故、ここで 割り込みの 多重の話をしたかというと、I2Cの Wireライブラリが ちょっと気になっているのです。割り込みを 使用して無ければ 問題ないです。 使用しているならば、優先レベルが 高くて多重割り込みを許可されている環境である事を 祈るばかりです。
実際に 試して確認します。

SDカードのデータ書込みに要する時間

先ほどの実験動画で保留にしましたが、タイマー割り込みを使用する もう一つの理由が、SDカードの データ書込みに要する時間です。

SDカードのデータを 読み出す時は 早いのですが、書き込みは やはりフラッシュメモリなので多少時間が かかると思います。で、SDカードは 1セクタ 512 byteなので データが512 byte以上 メモリ上のバッファに溜まった時に、1セクタ データを SDカード上に 書き込む事になります。 で、この 1セクタの書き込みに 10ms かかると また、処理が間に合わないという事に なります。 よって、SDカードの書き込み処理が 多少遅れても センサーからのデータ取り込みは、遅れなく一定のサンプルレイトで データを 取り込みたいのです。

それを 実現するために タイマー割り込みと遅延用のリングバッファを 使用するという事です。 タイマー割り込みで センサーからのデータ取り込みは、確実に定周期で 行われます。

SDカードに データ書込み時、一時的に発生する若干の遅れは、リングバッファに データを 貯め込む事で 対応します。

で、リングバッファの書き込み処理は タイマー割り込み内に 入れますが、リングバッファの読み出し処理は、メインの バックグランドループで読み出します。 で、 リングバッファの書き込み処理と 読み出し処理が、非同期で動く事になるので、排他制御も必要となります。

何か いろいろ考えてると だんだん アプリがややこしくなってくる感じですね。

状態遷移図に　ついて

　簡易データロガーを作るのに　必要な部品が揃ってきたので、全体の動きを制御するために状態遷移図を作成します。 初心者の方は、また聞きなれない言葉が出て来たと　思いますが、さほど難しい話では無いです。

　言葉の通りなのですが、その時の状態が、何らかの入力が発生する事により、状態が移り変わる事を 意味します。 まず、どのような状態が存在するのかを検討します。

①　待機モード:
　電源オン直後の状態です。 最低限の条件が揃っていたら　待機状態になります。
　最低限の条件とは、SDカードが スロットに挿入されているか、センサーが適切に接続されているかです。

待機状態では、PCからのコマンドは　受け付けますが、自動でトリガ判定し　データを収録する事は　出来ません。

②　運転モード:
運転モードは、PCからのコマンドによって移行します。 一旦　運転モードになると、PCのコマンドは、「停止」ボタン以外は　受け付けません。
　代わりに　自動でトリガ判定しデータ収録する事が　可能となります。

③　データ記録中モード:
　これは、トリガ判定で　ON状態になると、センサからのデータを取り込み、リングバッファ経由で　SDカードに　データを書き込みます。
　所定のサンプル数書き込むと終了で、元のモードに　移行します。 記録中の　場合は「停止」ボタンも　受け付けない事にします。

簡易ロガーの 状態遷移図

簡易ロガーの　状態遷移図は、大雑把に
このように なります。
待機モードでのセンサ読み取りは
検討中です。