PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。- 内蔵クロックで駆動 -

「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。」
「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。- Interrupt —」
「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。- Timerを追加 —」
のつづき。

３か月も経って、もうだいぶ忘れているが、思い出しながら再挑戦だ。

前回までは、外付けのクロックを使っていたが、面倒なので内部クロックを使う。

その分回路はシンプルになる。

クロックの設定の前回までとの違いは赤枠の部分だ。

PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。- Timerを追加 -

「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。」
「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。- Interrupt —」
ときたので、今度は前回作ったプロジェクトでタイマーをもうひとつ追加してみる。

タイマーを追加する。

LED接続用に追加でRB15を設定する。

Genereteすると、見慣れない窓が開く。

初めての作成ではないので、既存のファイルと、今回新たに作成するファイルとの違いが生じる。

その差異をどう扱うかを決めろということのようだ。

矢印をクリックするとコードを追加し、バツをクリックすると削除する。

前回のコードはそのままにして、新たなタイマー部分を追加するので、矢印だけをクリックする。

こうして作成されたsystem_interrupt.cに、赤線部分を自分で追加する。

main.cはこうする。

以上で、タイマーの追加が完了する。

ちゃんと動いているようだ。

せっかくなので、点滅速度を落として、動画を撮影してみた。

参考記事

PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。」
「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。(改めて見直し)」
「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。- Interrupt –」

PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。- Interrupt -

HarmonyでＬチカのInterruptバージョンもやってみた。
「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。」の続き。

回路は前回と全く同じだ。

Clockの設定も前回と同じ。

変更するところだけ書く。

TimerはInterruptモードで使うので、以下のように変更する。

この部分を新たに追加した。

コードは、以下の２つに変更を加えた。

まず、Harmonyの吐き出したmain.cに以下の変更を加える。

さらに、system_interrupt.cに赤線の部分を追加する。

20000000 / 64 / 78 = 4006Hz

ということなので、ほぼそれらしい実測値となっている。

今回参考にしたサイト

Using Harmony Static Drivers to Control Timers

手動で書き足す必要があるコードについては、以下の２つの部分に書かれている。

追記(2017/09/24)

上の通りだと4kHzで点滅する。

ロジアナなどを使わないと肉眼では動作を確認できない。

下のように変更すると10Hzになる。

20000000 ÷ 256 ÷ 7800 = 10Hz

これなら肉眼でも点滅を確認可能だ。

関連記事

PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。」

「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。(改めて見直し)」

「PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた。- Timerを追加 —」

PIC32MX250F128B MPLABX XC32 Harmony でＬチカしてみた(MPLAB

Harmony Configurator入門)。

よく目にするHarmony(MPLAB Harmony Configurator = MHC)って何?

ってことで、簡単な例として、お決まりのLEDチカチカをやってみる。

環境

MPLAB X IDE v3.30

MPLAB XC32 v1.40

MPLAB Harmony 1.08

MPLAB Harmony Configurator 1.0.8.7

PICkit3

PIC32MX250F128B

参考にしたサイト

SK1632 Tutorial, and Introduction to MPLAB’s Harmony(追記 若干疑問点があったのでのちにこちらで再検討している。)

インストールしてあるツールのバージョンの違うものがあり、やってみて多少異なるところもあったが、まあ、迷って止まってしまうような差ではなかった。

このサイトは、どうもSK1632というボードの使い方を説明していて、その中でこのボードにPIC32MX250F128Bを載せてLEDを点滅させる例を、Harmonyでやってみたということらしい。

なので、回路図はこのボードのものを参考(pdfをもらっておいたSK1632 Schematic)にして必要最低限のものだけを取り付けてある。

ちなみに、最初は1つもコンデンサをつけてなかった。

20ピンのVCAPにつけないとプログラムが書き込めなかったのでここだけ0.1uFをつけた。

出来れば内蔵のオシレータで駆動したかった。

ただ、余計なトラブルを避けるためには最初は参考サイトの通りに正直にやるのがいい。

水晶発振子は手持ちが無かったので8MHzのセラロックを取り付けた。

以下、MPLAB X IDEでの手順を書く。

まずは MPLAB X IDE 上で Harmony の準備をする。

Harmonyをインストールしたフォルダの、¥utilities¥mhc¥com-microchip-mplab-modules-mhc.nbm を指定する。

ここで、MPLAB X IDEを 再起動する。

新規プロジェクトを作る。

ここでHarmonyを使う設定にする。

プロジェクト名を適当につける。

使うチップを決めて[Finish]をクリックする。

プロジェクトの保存場所だが、水色の線がこちらで決めたフォルダの場所だ。

緑色の線は Project Name の欄に入れた名前をもとに自動で作られるようだ。

ユーザーは変更できない模様。

[MPLAB Harmony Configurator]が開いてなかったら、以下のようにして開く。

[MPLAB Harmony Configurator]から[Options]を選ぶ。

MPLAB Harmony & Application Configuration > Harmony Framework Configuration > Drivers > Timer

を開いて、以下のように設定する。

この後、下でクロックを2MHzにするので、

20000000 / 2 / 256 / 78 = 500Hz

となる予定だ。

MPLAB Harmony & Application Configuration > Harmony Framework Configuration > System Services > Clock

を開いてClock Configurator [Execute]をクリックする。

[Clock Diagram]が開くので下のように設定する。

データシートにある図と同じだが、クリックで設定していくことが出来るようになっている。

つまり、データシートと首っ引きでコードを書いていたところを、GUIっぽくしたということのようだ。

確かにこれは煩雑さが減って便利だ。

ちなみに、私の場合は幾度か試行錯誤する過程で右下のFNOSCがPRIPLLになってないのを見落とし、少し悩んだ。

続いて「Pin Settings」からピンの使い道を設定する。

今回はRA0のLEDを点滅させるだけなので以下のように設定する。

あとは、[Save Configuration]の後に[Generate Code]をクリックすれば、設定が保存されて、必要なコードが生成される。

びっくりするほどたくさんのファイルが作られるが、

その詮索はいずれ行うとして、まずはmain.cに自分のやりたいことのコードを書く。

下で、赤線の引いてある部分が、参考サイトを見て追加したりコメントアウトしたりした部分だ。

タイマーのカウントをチェックしてLEDを点滅させるだけのごく普通のコードだ。

```
#inclide <p32xxxx.h>
```

入れないとエラーでコンパイルできなかった。

一応、ちゃんと予定通り500ms、2Hzで点滅しているようだ。

少なくとも、データシートを眺めながらちまちまコードするよりは楽だと思う。

ただ、その自動で生成されたコードがどこにあって、どんなのかはまだまったく調べていない。

関連記事

「PIC32MX250F128B MPLABX XC32 Harmony でLチカしてみた。(改めて見直し)」

「PIC32MX250F128B MPLABX XC32 Harmony でLチカしてみた。- Interrupt —」

「PIC32MX250F128B MPLABX XC32 Harmony でLチカしてみた。- Timerを追加 —」

PIC32MX - SYSTEMConfigPerformance();って何? -

PIC32MXのオシレータの設定を調べる。

まず、System Clock(SYSCLK)について考える。

Primary、Secondary、Internal Fast そして Internal Low-Power の4つのクロック・ソースから選べる。

Primary、Secondaryは外付けなので、基本的に内蔵を使うことにし、速い方を使うことにすると Internal Fast RC (FRC) Oscillator を使えばいい。

かつ、PLLを利用してとにかく高速で動かしたい(これは何となく)ので、結果として、Internal FRC Oscillator with PLL Module を選ぶことにする。

```
#pragma config FNOSC = FRCPLL // Oscillator Selection Bits (Fast RC Osc with PLL)
```

あとは、PLLだが、これはFRCPLLで設定する。

8MHzそのままでは速すぎるので、半分の4MHzにしてから入れるために、FRCDIVで設定する。

```
#pragma config FPLLIDIV = DIV_2 // PLL Input Divider (2x Divider)
#pragma config FPLLMUL = MUL_20 // PLL Multiplier (20x Multiplier)
```

OSCCONないのレジスタで設定するという説明になっているが、上のように、Configuration Bitsで設定してもいいんだよねえ、きっと。

というわけで、以下のようなコードを走らせてLチカさせてみる。

[crayon-6717490cb51c4190744605/]

出力をオシロスコープで見てみると、5.48Hzとなっている。

遅すぎないか?

80MHzのつもりなんだけど。

とりあえずMPLABXのシミュレーションでストップウォッチで時間を計ってみることにする。

オシレータは80MHzとする。

Instruction Frequencyはどうすりゃいいんだろう。

１クロックで１サイクルでよかったかな?

とりあえず、同じにしておこう。

すると、LEDの点滅間隔はシミュレータ上では11.5msとなるので、約87Hzとなる。

なぜ実測と15倍も違うのだろう。

この後、設定をいろいろ見直したりしたが、この違いは解消されなかった。

で、いろいろさまよった挙句に見つけたのが、表題にもある SYSTEMConfigPerformance(); だ。

```
#include  SYSTEMConfigPerformance(80000000L);
```

を追加するだけで、実測はこうなる。

38.36Hzとなって、これはさっきの5.48Hzの実に7倍だ。

> *SYSTEMConfigPerformance() is a very helpful library function used to easily optimize the performance of the PIC32. You provide the system (instruction) clock frequency and this function will do the rest. It will perform the following tasks for you:*
> *Enable instruction and data caching*
> *Enable instruction prefetch*
> *Configure the Flash and SRAM for minimum wait states*
> *Maximize the peripheral bus clock frequency*

とのことだ。

ループで時間稼ぎとかいう無駄な処理を行っている場合は、適度な最適化が必要ってことか。

何だか謎が多いねPIC32は。

もらったサンプルで遊んでるだけだからいいけど、自腹で買って何かに使おうとしたら、なかなか紆余曲折が多くてめげそうなチップだ。

PIC32MX470F512H-I/PT　自力でLチカ(for文で時間稼ぎ)

自力でというのはサンプルそのままでなく、データシートだけを頼りにやってみようということだ。

当然やるのはLEDチカチカ、通称「Lチカ」だ。

その前に以前やったサンプルをもう一度動作確認のために動かしておくことにする。

さらにその前にXC32が1.33になっていたのでIDEもその他のコンパイラーも含めてバージョン・アップしておく。

で、あらためて、

C:¥Program Files (x86)\Microchip¥xc32¥v1.31¥examples¥plib_examples¥timer

にある timer1_int というサンプルを実行してみようとして Build すると、何だかえらいたくさんのエラーだか警告

だかが表示されてから完了する。

以前は無かったことだ。

で、

> *The PLIB functions and macros in this file will be removed from the MPLAB XC32 C/C++ Compiler in future releases*

と書いてある。

「このファイル内のPLIB関数やマクロは、将来のリリースでは MPLAB XC32のC/C++ コンパイラから除かれます。」

確かに

#include <plib.h>

という一行があって、plib.h というヘッダファイルはPIC32を使う前は見たことが無かったものだ。

そもそも、このサンプル自体が plib_example の中に含まれているものだ。

PLIB function というものが絡んでいることは分かる。

で、#include <plib.h>をコメント・アウトしてみると、一気に赤線を引かれる部分が増えるので、この今まで見かけたことの無い記述がPLIB function やら マクロなんだろうなとわかる。

前回、これ覚えなおさなきゃいかんのかいなと思って放置していたものだが、将来は無くなるとのことだ。

無くなってどうなるのかは分からないが、無理に解読する必要も無いような感じがするので、分かる範囲で従来どおりの記述方法でLチカをやってみることにした。

RD0につないだLEDが点滅するだけのものだが、Configuration Word が大部分を占めるという上位チップでのLチカではありがちな状況にはなる。

[crayon-6717490cb5681540721488/]

待ち時間もfor文のループなので、全くの適当だ。

次は、前回せっかく８つLEDが載ってる基板を作ったので全部光らそうというわけで、一方からもう一方へLEDが流れるように点灯する例だ。

[crayon-6717490cb5688441112219/]

アナログ用途にも使えるピンが入るので、お決まりのANSELD、そしてもちろんTRISDも設定する。

時間稼ぎは相変わらず適当なfor文ループ。

PIC32MX470F512H-I/PT led_message

力尽きて１灯だけの点滅で放置だった

Microchip¥xc32¥v1.31¥examples¥xc32_examples¥led_message

は、そもそもExplorer-16 boardとかいう$130もするボード用のサンプルだ。

だが、このサンプルを実行するのにこのボードが必須というわけでもなく、要はLEDが８つ並んでりゃいいんだろうということで、もしかしたら２度と使わないかもしれないが、LEDボードを作成してみた。

秋月の変換基板に載せた PIC32MX470F512H-I/PT につないだところがこんな感じ。

コードと説明を読んだ限りでは Versa Writer 的な感じで HELLO と表示するらしい。

> *Run this example on Explorer-16 board with PIC32MX PIM.*
> *Hold the board vertically from the PICtail connector size and wave the board back-and-forth to see message “HELLO” on LEDs*

とは書いてあるが、やってみてもどうもうまく読み取れない。

何だかよく分からなかったので、動画を撮って画像を並べてみた。

ああ、確かに HELLO と表示してるのね。

PIC32MX470F512H-I/PT + 0.5mmピッチQFP(64ピン)変換基板 動作確認編

先日ハンダ付けした PIC32MX470F512H-I/PT + 0.5mmピッチQFP(64ピン)変換基板 の動作確認をした。

DMMで全PIN導通テストを行い、問題なさそうだった。

PICkit 3 Programmer を使ってPICkit 3 と接続したが認識しないので、MPLAB PIEで試したがやはりだめだった。

ハンダ付けのミスなのか、配線のミスなのか、ＰＩＣ３２特有の問題なのか切り分けるために、ＤＩＰパッケージのPIC32MX220F032Bで認識するのかをまず試してみた。

その後、PIC32MX470F512H でやってみたところ、認識はしたがVerifyでエラーが出る。

認識するのにエラーってことは壊れた?

私の非常に貧相な経験から考えると、接触不良ならいざ知らず、ハンダ付けで素子が壊れるということはめったに無いことなので、もう一度データ・シートを読み直して must be connected to とあるものをピックアップした。

AVDDやVSS以外にも以下のものが書かれている。

USBを使わないとき、VUSB3V3はVDDに接続しなさい。

VCAPには適当なコンデンサを接続しなさい。

以上を整えて再度チェックすると、今度はちゃんとEraseもReadもVerifyも出来るようになった。

追記 2014/08/13

最終的に、必ず必要な配線は変換基板の裏で接続し、表はピンソケットを取り付けることにした。

当面は実験基板としてしか使わないだろうと考えたからだ。

だとすると、ブレッドボード用のジャンパー線がつなげるのが最も都合がいい。

いまひとつポリウレタン線のハンダ付けがしっくりしないが、一応DMMで電源周りの導通テストは完了した。

この段階で、VDD､VSS､AVDD､AVSSはそれぞれ電源かGNDにつながっている。

VCAPには4.7μFのセラミック・コンデンサ(データシートでは10μFのタンタル・コンデンサかセラミック・コンデンサとある。)がつながっている。

VUSB3V3はつなげていない。

追記 2014/08/14

４隅にVDDとGNDのランドがあるので、そこにはピンヘッダを取り付けて、ブレッドボードに差し込めるようにした。

どうせなら、ブレッドボードの電源ラインに載せて電源を取れるように、赤丸の位置にドリルで穴をあけて、ピンの位置をずらすことにした。

どの向きでブレッドボードに載せるか悩んだので、右の穴はあけただけでピンはずらしてなく、左のピンだけをずらしてある。

PIC32MX470F512H-I/PT　Lチカでもやっとくか

やっと　PIC32MX470F512H-I/PT　を使える状態にできたので、Lチカでもやっとくかと思って始めたはいいが、他のチップとは全然違うので右往左往して、結局サンプルでお茶を濁した。

その、右往左往の覚書き。

とにかく、今までのPICとは全く別物なのだということらしい。
これで、Lチカなんてやるのは馬鹿げている。

Microchipからもらってきたデータ・シートを読むと、何だか説明が全部中途半端なので、よくよく確認すると

となっている。
だよねえ。
で、ここへ行ってReference Manualってヤツを頂こうとするわけだが、この分量だ。(※)

これみてほとんど嫌になったわけだが、気を取り直してLチカに必要そうなものだけもらっとくことにした。

で、
Microchip¥xc32¥v1.31¥examples¥plib_examples¥timer¥timer1_int
Microchip¥xc32¥v1.31¥examples¥xc32_examples¥led_message
にあるプロジェクトをそのまま書き込んで、Lチカが出来ることを確認したところで力尽きるw。
Microchip¥xc32¥v1.31¥examples¥plib_examples　の方は　PIC32 Peripheral Library　のサンプルということで、なかなか使えそうなライブラリだが、その反面1から読み直さないといけない。
Microchip¥xc32¥v1.31¥examples¥xc32_examples　の方は眺めていて何となく読める感じで、従来の書式と大きく違うことも無いようだ。
言い訳するわけじゃないが、自分的にはPIC32MX470F512H-I/PTを問題なく0.5mmピッチQFP(64ピン)変換基板にハンダ付けできたことが確認されれば当初の目的は達成ということだ。

(※)
下の方でこうやると一覧が表示されて、見たいのをクリックすると表示やダウンロードができる。

PIC32MX220F032B + MPLAB IPE v2.15

PICkit 3 Programmerで一向にPIC32MX470F512Hを認識しないので、いよいよMPLAB IPEの出番かということで使ってみる。

バージョンはv2.15だ。

問題の切り分けが必要なので、ここではDIPパッケージのPIC32MX220F032Bで試してみることにする。

IPEを起動して何も設定を変えずにConnectを押すと以下のように

Target device was not found. You must connect to a target device to use PICkit 3.

と表示される。

SettingからAdvanced Modeをクリックして、

PasswordにmicrochipとRemember Passwordにチェックを入れてLog onをクリックする。

左のPowerをクリックしてからPower Target Circuit from Toolにチェックを入れる。

左のOperateをクリックすると元の表示に戻るので、あらためてConnectをクリックすると、今度はちゃんとTarget detectedと表示される。

下は、続けてVerifyをクリックしてみたところ。

ちなみに、PGEDとPGECを逆に接続してあったらこんな表示になった。

また、IPEで接続を確認したそのままの状態でPICkit 3 Programmerを起動して試してみると、やはり認識しなかった。

PICkit 3 Programmerは新しいPICにはもう対応しないということかも知れない。

接続するPINについては上の図の通り。

必ずつなぎなさいとデータ・シート書いてあるのは、AVDDとVSSで、VDDはつながなくても認識したがAVDDをつながなければ

認識すらしなかった。

データの書き込み時にはもちろんVDDもつなぐ。

VSSはどちらか一方でよく、PGECとPGEDは３組のうちのいずれか１組を接続、MCLRはもちろん接続する。