MPLAB X IDE v5.45でPIC32MM0064GPL028用の新規プロジェクトを作成

環境のインストールについては以下の記事に書いた。
「MPLAB X IDE 5.45 をインストールする。」

以下のいずれかでMPLAB X IDEを起動する。

初めて起動するといろいろな窓が開いた状態だが、混乱の元なのでとにかく全部閉じる。

使用するプログラマをPCのUSBにつなぐ。
Toolのリストに表示されるので選択する。
ここではPICkit3をつないでいる。

Project NameとProject Locationを決める。
Projct Folderは「Project Name + .X」という名前で自動的に作成される。

以下のようにPICkit3の電源設定をしておく。

ゼロからスタートしてPIC32MM0064GPL028でLチカするまでの全過程 (開発環境のインストールやパーツ購入も

含む）のまとめ

この記事には新しい内容は含まれていないので、このサイトに散在している関連記事を自分で追える方には無用の記事だ。
まったく何も無いところから始めて、開発環境のダウンロード、インストールや秋月での必要パーツの購入から「PIC32MM0064GPL028でLチカする」までの全過程の関連記事を順にリストアップしただけの記事だ。

スタート段階で想定している環境は「Windows10のPCからインターネットへ接続できる環境」だ。
PICの開発環境はLinuxやOS Xにも対応しているので同様だと思われるがここでは触れない。

1. 開発環境（ソフトウェア）を整える（無料）。
□MPLAB X IDE, XC32 Compiler, Microchip Code Configurator(MCC)の準備
□「MPLAB X IDE 5.45 をインストールする。」
2. ハード関連を秋月で購入する（送料込み4780円）。
□PIC32 手始めキット（SNAPあり）
□参考記事
□□「PIC32MM0064GPL028 — Lチカ用の回路の準備（DIP編） —」
□□記事後半「追記」以降
3. ブレッドボード上にPICの回路を準備する。
□「PIC32MM0064GPL028 — Lチカ用の回路の準備（DIP編） —」
□□記事前半部分
4. MPLAB X IDEで新しいプロジェクトを作成する。
□「MPLAB X IDE v5.45でPIC32MM0064GPL028用の新規プロジェクトを作成」
5. 「PIC32MM0064GPL028 — MPLAB Code Configuratorを使ってTimer1の割り込みでLチカさせる。 —」

とあるモジュールを買ったマイコン初心者がそれを使えるようになるまでの道のり。

> 基本的なことがまだ理解出来ていないと思いますが、*LCD*（*I2C*接続）について教えてください。
> *LCD 1602A*
> *I2C*変換モジュール□*PCF8574*

PIC 12F1822

2行で使用したいのですが

電源投入後のLCD初期化処理は次のコマンドでいいのでしょうか?

0x28

0x29

0x14

0x70

0x56

0x6c

delay 200ms

0x28

0x01

0x0c

4bitモードにしたあとは送信データだけでなく、アドレスや制御コードも4bitで送信しなくてはいけないのでしょうか?

今手元に現物がなくてわかる範囲になります、すいません。

購入したLCDとI2C変換

Amazonで下記の製品名の物になります。

WayinTop 1602 LCD ディスプレイモジュール 16×2キャラクタ 青 バックライト付き 白抜き + IIC/I2C/TWI/SPI シリアルインターフェイス シリアルI/F モジュールポート

参考にしたサイト

http://machoto2.g2.xrea.com/page/P1822/P12_A05.htm

コードについては参考にさせてもらってるところと同じです。アドレスは変えました。これをもとに4ビット試したりしました。

某掲示板でとある質問者による投稿が上の２つ。

私ならどうするだろうと考えてみた。

まあ、掲示板へ質問しに行かないことだけは確かだけど。

販売サイトに以下の情報がちゃんと書かれている。

LCDディスプレイ:

動作電圧: 5V

インターフェースIC: HD44780

表示: 16文字 x 2行

バックライト色: 青

文字カラー: 白

コントラスト調整: V0(ピン3)

サイズ: 80x36x9mm

I2Cシリアル I/F:
動作電圧: 5V
チップ: PCF8574
I2Cアドレス: デフォルト0x27、0x20~27に変更可能
ピン: GND、VCC、SDA、SCL
ピン接続: GND-GND、VCC-5V、SDA-A4、SCL-A5
半固定抵抗付き、コントラスト調整可能
サイズ: 41.5x19x15.3mm

LCDディスプレイのピンアサイン:
ピン1(VSS): GND
ピン2(VDD): 電源(+5V)
ピン3(V0): Contrast Adj—VRでVDD-VSS間で電圧調整できるよう接続
ピン4(RS): Register Select—コマンド発行の際0, データ転送の際1にセット
ピン5(RW): Read/Write—読出:High, 書込:Low
ピン6(E): Enable Signal—通常はLowにしておき、送受信の際Positive pulseを与える
ピン7~14(D0~D7): Data Bit0~Bit7—8bitモード
ピン15(A): バックライト 電源プラス—VDDに接続可
ピン16(K): バックライト 電源マイナス—VSSに接続可

I2Cシリアルインターフェイス アドレスの対応:
A0□□A1□□A2□□I2Cアドレス
0□□□0□□□0□□□0x20
1□□□0□□□0□□□0x21
0□□□1□□□0□□□0x22
1□□□1□□□0□□□0x23
0□□□0□□□1□□□0x24
1□□□0□□□1□□□0x25
0□□□1□□□1□□□0x26
1□□□1□□□1□□□0x27

当然、真っ先にGoogle先生にお尋ねする。
「PIC HD44780 PCF8574 1602」で検索して上から7番め。

写真から見てもこれに違いない。

ソースコードと

モジュールのライブラリまで提供されていた。

持ってるはずのモジュールが見つからないので動作確認はしてない。

それでも、結論はただ一つだ。

「四の五の言わずにまずググれ。」

追記

探したら出てきた。

液晶の方は単独で使うようにピンヘッダがはんだ付けしてあった。

IICのモジュールの方もブレッドボードに挿して使うようにピンヘッダをはんだ付けした。

2014年8月1日に購入し、

「1602 16×2 HD44780 Character LCD with I2C Serial Interface Adapter Module」

2015年3月3日にIIC部分を切り離していた。

「16×2 Character LCD with I2C Module の分離」

ずいぶん荒っぽいことをやって、しかも放置してあったので、まずArduinoで動作テストした。

問題なく動作するようだ。

続けて、上で見つけたサイトから拝借したコードとライブラリをPIC16F1827に書き込む。

表示テスト無事完了。

追記

探したらPIC12F1822が出てきた。

が、メモリーが足りずにBuildできない。

と思ったらC90にしたらBuildできた。

こちらも問題なく動作した。

8pinなので実行時にICSP関連の線を外すのを忘れずに。

追記

で、肝心の質問者さんはどうなったのか、毎度毎度のフェードアウト、自然消滅、音沙汰無し。

ただのネタだったんでしょうね。

PIC32MM0064GPL028 + ST7735 □□ □□□□□□□□□□□□□□□□□□□- □□□□□ □GIMP□ -

□PIC32MM0064GPL028 + ST7735 □□□□□□□□□□□□□□□□□□□□□□□- □□□□□□ —□□GIMP□□□□□□

□□□□□□□PaintShop Pro(PSP)□□□□□□□□□□□□□
□□□GIMP□□□□□□□PSP□□□□□□□□□□□□
□Gimp□□□□□□PaintShop Pro□□□□□□□□□
GIMP□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□GIMP□□□□□□□□□□□

□□□□□□□□□□□□□□□□□GIMP□□□□□□□□□□

128×128□□□□□□□□□□□□□
□□□□□□□

□□□□□□□□□□

□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□

□□128×128□□□□□

□□□□□□□□□□□□□□□□□

□画像 ＞ モード ＞ インデックス

□□「色ディザリング」は「ディザリングしない」の方がきれいかも

あとはBMPとしてエクスポートすれば出来上がり。

Cのデータ・ファイルの作成

以前PSPで使った変換プログラムでやったら、上に載せたように色がおかしくなってしまった。

以前の画像と比較したら、そもそもBMPのファイル・サイズが違っていた。

バイナリ・エディタで見てみる。

下のように、情報ヘッダサイズが異なっていることが分かった。

PSPで作ったファイルは0x28 = 40、GIMPだと0x6C = 108となっている。

前回作った変換プログラムは値決め打ちで40に合わせて処理していたのでうまくなかった。

プログラムを書き直して情報ヘッダサイズを読み込んでその値を反映させるように改良した。

問題なく表示されるようになった。

この1.44インチのカラー液晶に実際に写真を表示させる話は「PIC32MM0064GPL028 + ST7735 でプログラム・メモリに置いた画像を表示させる。 ー 表示する ー」

へ続く。

MPLAB X IDEでCopyしたProjectがBuild出来ない。

MPLAB X IDE v5.40
XC32 v2.50

こんなエラー

「XC32 hex does not exist or is not an excutable」で検索したら以下の手順で解決したという投稿があった。

プロジェクト名の右クリックからProperties

Manage configurations

新しいConfigurationを作る

名前は何でも

新しく作った方をActiveにする

とやって、プロジェクトの設定をチップの選択からやり直す。

すると問題なくBuildが完了した。

古い方(特に変更していなければdefaultという名前になっている)はRemoveしてしまっても構わないと思う。

PIC12F1822とMOSFETと中華DC-DCモジュールでタッチライト自作

今回新たに必要となるパーツ

- PIC12F1822 (8bit PICマイコン)

- IRMLM6344 (Nch MOSFET)

- 1kΩ 抵抗 × 2 (MOSFET用)

すでにある前提のもの

- 「LEDライトを作った。」
- 「昇圧モジュール基板裏LEDライト」
- NSSW157T × 3 (白色チップLED)

- MT3608 DC-DC Step Up Power Module (可変昇圧モジュール)
- 数10Ω 抵抗 (LED電流制限用)

触れてON/OFFするライトを上のパーツで作る。

某掲示板でPIC12F1501でPICに入門してる人がいて、久しぶりに8bit 8pin PICを触ってみたくなって手持ちを見たらPIC12F1822しかなかった。

目の前にある自作のLEDライトのスイッチが小さすぎてON/OFFがやりにくいのを改善したい。

この２つの動機を合わせた結果、PICでTouch機能を使ってMOSFETをONすることになった。

モジュール以外はチップ部品なので一瞬PIC10Fがいいなと思ったけど、データシートを斜めに簡易検索したところ、Touchには対応してない様子だった。

で、やはりPIC12F1822に戻った。

上にも書いたが、LEDライトは既存の改良なので既に存在する。

「LEDライトを作った。」

「昇圧モジュール基板裏LEDライト」

PICのプログラムは以前作ったMPLAB Xのプロジェクトを探したらTouchの作例があったので流用する。

MPLAB Xのプロジェクト・ファイル

Touch02_06_1sensor

KiCADの回路図

ブレッドボードでのテスト

https://neo-sahara.com/wp/wp-content/uploads/2020/10/201003_222850_2.mp4

PIC32MM0064GPL028 + MCC で外部クロックで駆動する。

セラロックとか水晶発振子でなく発振器からのクロックで駆動する。

と思ったら、発振器を持ってなかった。

セラロックと水晶発振子しか持ってなかった。

なので、Arduino Nanoで4MHzを作ってクロックにした。

[crayon-671749dce2f93091141245/]

上は拾い物のコードで、中身は見てない、いずれまた機会があったら見る。

とりあえず、これだけで11ピンから4MHzのクロックが出る。

これを、当初、SOSCIにつないだのだがうんともすんともいわない。

Secondary Oscillator (SOSC) on the SOSCI and SOSCO pins

って書いてるのに。

でもExternal Clock Input Operationという図でクロックはOSC1につないである。

なので、Nanoの11ピンをPICのOSC1につないだらPICが動き出した。

追記(2020/03/06)

Arduinoのフォルダ内をfindstrで探してみた。

[crayon-671749dce2f9c150741309/]

いっぱい出て来るけど328pがNanoなんだろうということでそこだけ拾う。

[crayon-671749dce2f9f288036288/]

つまり

[crayon-671749dce2fa1119216249/]

これですっきりした。

XCで書くなら、

TCCR2Abits.WGA21 = 1;

TCCR2Abits.COM2A0 = 1;

ってこと。

PIC32MM0064GPL028 + MCC で Ceralock で駆動する。

いまさら外付けでセラロックをつけて駆動するってのもあまり意味が無いように思う。

でも、精度とか求めたらそういう場面も出てくるかもしれない。

ということでちょっとやってみた。

PIC32 Family Reference Manual Section 6. Oscillators

を読む。

こう書いてある。

Primary Oscillator(POSC)の設定にしてOSC1とOSC2にセラロックをつなげばいいらしいとわかる。

MCCではOSC2の方が書いてないのだが、データシートでは9ピンと10ピンなので上のようにした。

で、Timer1の割込みで簡単なLチカをやってみたらちゃんと動いた。

試しにセラロックを引っこ抜いてみたらLチカが止まったので、ちゃんとセラロックで動いているんだろう。

System ClockをCLKO/OSC2につなぐ(上の(1))とクロックが出力される(下)。

デューティ比が50%でないのがちょっと不思議だが、周波数はちゃんと合ってる。

REFCLKOも設定すれば(上の(2))ちゃんと分周後のクロックが26ピンに出力される(下)。

MCCとデータシートで表記に統一性が無いような気がずっとしている。

クロックに関しても以下のような感じだ。

前半がMCCの表記で[]の中がデータシートの表記。

FRC Oscillator (8.0 MHz) Clock Source [Internal Fast RC (FRC) oscillator]

Primary Oscillator (3.5 MHz – 25 MHz) Clock Source [Primary Oscillator (SOSC) on the OSC1 and OSC2 pins]

External Clock (2 MHz – 25 MHz) Clock Source [Secondary Oscillator (SOSC) on the SOSCI and SOSCO pins]

LPRC Oscillator (32 kHz) Clock Source [Internal Low-Power RC (LPRC) Oscillator]

PrimaryとSecondaryの違いや意味がわかってないせいなのかな?

ESP-WROOM-02(Deep-Sleep) + Si7021 + OPiPC2 + 18650

ESP-WROOM-02を測定時間以外はDeep-Sleepさせる。

Wi-Fiとの接続待ちも含めて25秒程度の観測時間、約1分Deep-Sleepする、というセットを繰り返した。

センサーはSi7021を使ってWi-FiでOPiPC2上のサーバにデータを投げる。

電源はダイソー・モバイルバッテリーから取った18650だ。

これでどれくらいの時間駆動できるかテストしてみた。

電源電圧の変化の様子

観測結果全体

肝心の駆動時間

このサイトを使わせていただいた(https://www.calc-site.com/times/calc_elapsed_time)

3日半。

これくらいならバッテリー交換も苦ではないか。

ただ、調べた限りでは、強制吸気ファンつきの観測装置が主流のようなので、その場合はファンの分の消費電源も考慮する必要がある。

「ESP-WROOM-02(Deep-Sleep) + Si7021 + OPiPC2 + 18650 + Fan」

ESP-WROOM-02 + Si7021(SHT21 HTU21 GY-21) + Orange Pi PC2

BME280モジュールは実践投入して久しい。

気圧はどこで測定しても同じだろうから、温度と湿度だけのセンサーでいいやと思って、Si7021の載ったSHT21　HTU21 GY-21　などどシルク印刷のあるセンサー・モジュール(以後SHT21)を買ってあった。

だいぶ放置していたが、やっと使ってみることにしたのでその覚書き。

ESP-WROOM-02+SSD1306+BME280で使っている外注基板がまだ余っているのでこれを流用する。
BME280モジュールとSHT21はピン配列が同じなのでそのまま基板に取り付けて使うことができる。

バッテリー駆動で野外で使おうと思うので、とりあえず表示装置は無しにしてある。

ここのスケッチを参考にさせていただいた。

http://www.esp8266learning.com/esp8266-si7021-temperature-sensor-example.php#codesyntax_1

基板を作るときにI2Cの端子が逆になっていて、その設定がうまくいかないせいか、AdafluitのサンプルやSparkFunのサンプルはうまく動かなかった。

上のコードでは、`Wire.begin();`を`Wire.begin(5, 4);`とするだけで測定値が表示された。

AdafluitやSparkFunのサンプルではこれがどこにあるのか、どこで設定しているのか探せなかった。

このスケッチに、BME280の基板で使っているサーバー用のデータ転送処理を加えて、すぐにOrange Pi PC２上のサーバー

にデータを送ることができた。

左から順に、時刻、気温、湿度、電圧のADC値の順だ。

もっとてこずるかと思ったが、以前書いたコードをコピペするだけでほぼそのまま動いた。

関連ソフトウェアー式を売り出してみた。

https://neo-sahara.booth.pm/items/1924398