

I²C シリアル EEPROM ライターの試作と漢字ディスプレイへの応用

Design and Trial Manufacture of an I²C Serial EEPROM Writer System and its Application to a Kanji Display

袴田 吉朗*

Yoshiro HAKAMATA

Abstract: The paper describes the design and the trial manufacture of an I²C serial EEPROM writer system. It consists of a personal computer, a PIC micro computer and computer programs installed in them. An I²C serial EEPROM in that eight different kanji data were written, was built in the kanji display made last year. Distributed read process from an EEPROM was used and it confirmed to work well. The paper also summarizes the results on the kanji display.

1. はじめに

多くの高校生に電気電子に対して興味を持って貰うためには、"動く", "光る", "音が出る"などの要素を取り入れた展示物が効果的ではないかと考えている。この観点から昨年度には文字が"光り"ながら"動く", という2つの要素を取り込んだ漢字表示電光掲示板(以下漢字ディスプレイ)を設計、試作し[1], いろいろな所で使用してみて効果がありそうな感触を得た。

試作した漢字ディスプレイでは、漢字データをプログラムメモリに格納しており、表示できる漢字の個数は最大103個である。また表示内容を変えるためには、漢字データを作成する専用のプログラムを使う必要があり、汎用性が劣る点で設計者以外には使いにくいと言える。

そこで種々の催し物などにおいて誰でも使えるようにするために、あらかじめ外部EEPROMに多くの漢字データを格納しておきスイッチによってデータを切り替えられるようにしておく方法を検討した。

外部EEPROMにはI²CシリアルEEPROMを使用することにし、本論文の前半では、PCのプログラムからPICマイコンに搭載したプログラムを介してEEPROMに漢字データを書き込み/読み出すためのシステムの設計、試作を行った結果を示す。また論文の後半では、シリアルEEPROMとして256Kビット(32Kバイト)のメモリである24LC256を適用して、試作した漢字ディスプレイの表示内容を大容量化した検討結果を示す。漢字ディスプレイへのEEPROMの搭載により、スイッチ切り替えによって8組の漢字データ(1組当たりの漢字数は最大128個)を表示することが可能となり、所期の目的を達成することができた。

2. I²CシリアルEEPROMライターのシステム概要

図2.1にシステム構成の概要を示す。本システムは、PCおよびPICマイコンと両者に搭載された2つのプログラムからなる。PCとPIC間の通信にはレガシーインタフェースであるRS232Cを使用している。ボーレートは19.2kbaudである。PCのプログラムはVC++6.0 Standard Editionを使用して作成し

ており、コマンドを選択してクリックするとRS232Cインタフェースを介してPICに伝送されるようになっている。EEPROMの開始アドレスや漢字の個数はエディットボックスに16進数で入力するようにしている。

PICのプログラムは、Microchip社のアセンブラで作成しており、文献[2]に掲載されている「Appendix 8 EEPROMの読み書き処理テスト・プログラム2」の処理内容を基本とし、一部必要な変更を加えて作成した。変更した内容を以下に示す。

- (1) 専用のライタープログラム(ターミナルソフト)を作成
文献[2]ではPC側のターミナルソフトとしてハイパーターミナルを使用するようになっている。しかし漢字データはバイナリファイルでありこれをハイパーターミナルの形式に合わせる方法が理解できなかったこと、ハイパーターミナルでは読み出したEEPROMの内容が単に羅列されるだけで見難かったこと、から専用のライタープログラムを作成することにした。既に作成済みのプログラムがほぼ応用できると言う事情もあった。
- (2) PICにはPIC16F648Aを使用

文献[2]ではPIC16F877Aと内蔵のSSPモジュールをI²C通信処理に使用している。しかし本検討では従来から使用してきた18ピンのPIC16F648Aを用いることにした。PIC16F648AはSSPモジュールを内蔵していないので、文献[3]に掲載されているI²C通信処理プログラムをファームウェアとして使用することにした。PICとEEPROM間のボーレートは約50kbaudである(漢字ディスプレイでは約100kbaudの読み出し速度)。

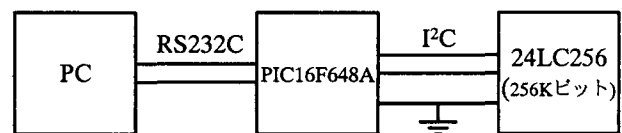


図2.1 I²CシリアルEEPROMライターシステムの構成

3. PICマイコンにおける構成

3.1 ハードウェア構成

図3.1にPICマイコンを使用したI²CシリアルEEPROMライターのハードウェア構成を示す。PICマイコンには漢字ディス

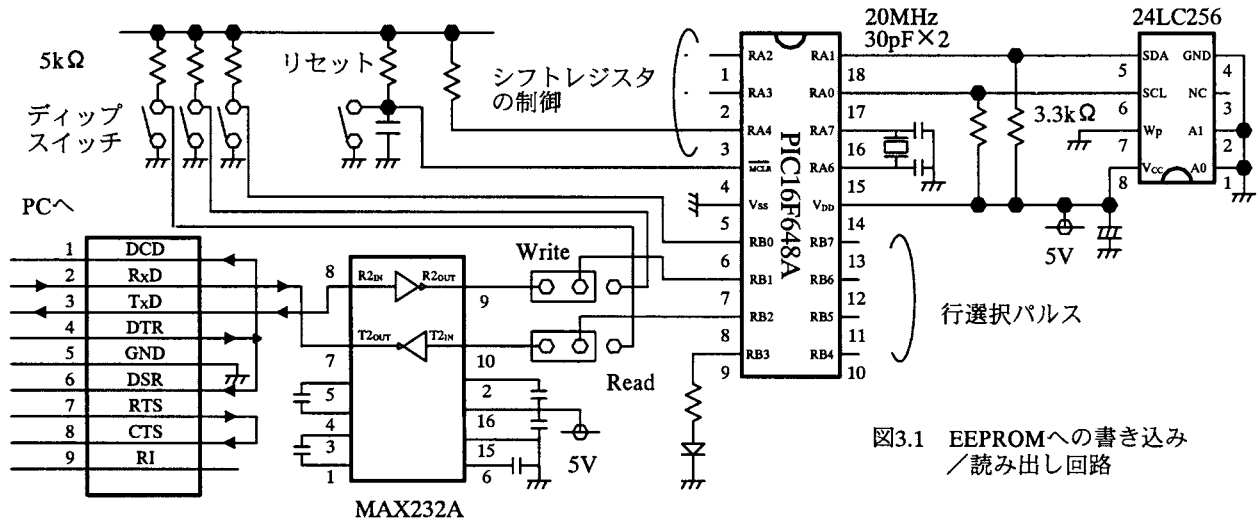


図3.1 EEPROMへの書き込み／読み出し回路

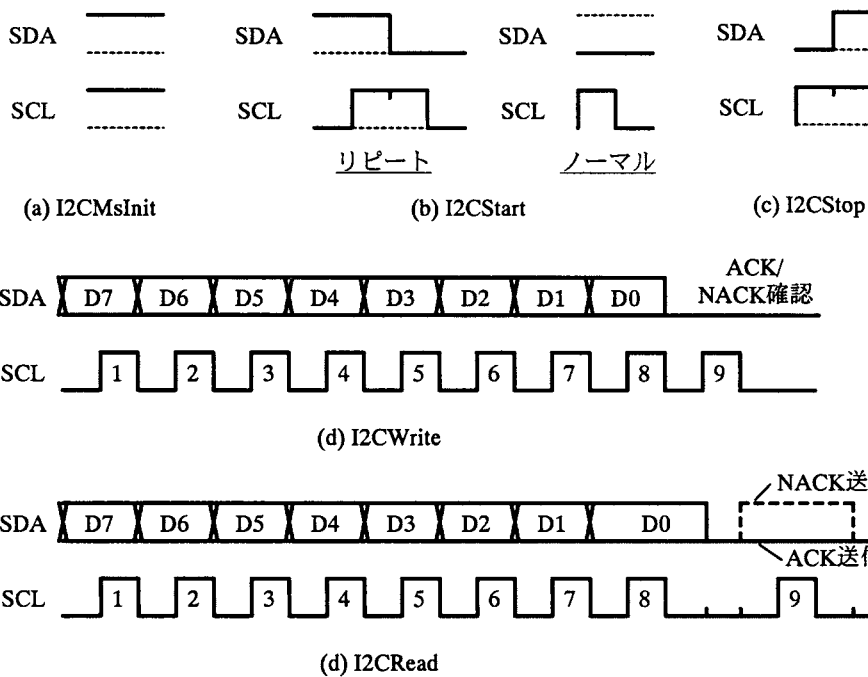


図3.2 ファームウェアを用いたI²C通信におけるコマンド

プレイとの共用を考慮してPIC16F648Aを使用しているが、プログラム容量は大きくないのでPIC16F628Aも使用可能である。クロック周波数は20MHzである。ポートAのRA0をSCLラインに、RA1をSDAラインに割り当てた。なおRA2~RA4、ポートBのRB4~RB7は漢字ディスプレイ用である。

使用したシリアルEEPROMは256Kビット(32Kバイト)の24LC256である。3ビットのディップスイッチはシリアルEEPROMを8分割し、1つ当たり最大128個の漢字データを格納するために使用する。24LC256のアドレスA0、A1は0に固定している。3.3kΩの抵抗は、EEPROMのSDAラインおよびSCLラインがオープンドレインとなっており、その負荷抵抗である。PICのピン数が十分多くないため書き込み時にはWriteに、読み出し時にはReadにストラップを接続する必要がある。

MAX232AはRS232CとTTLのレベル変換用である。

3.2 ファームウェアによるI²C通信

図3.2にファームウェアを用いたI²C通信におけるコマンドを示す。

I2CmsInitはバスをHにしてアイドルにするためにプログラムの動作開始時に実行する。I2CStartはスタートコンディションである。EEPROMの内容を読み取る際には、アドレス等の書き込みを行い、次に読み出しを行う。この時にはリピート・スタートコンディションを使用するが、それ以外の時にはノーマル・スタートコンディションを用いる。I2CWriteはPICがEEPROMに書き込むときに使用するコマンドであり、1Bのデータを書き込むために10SCLの時間が必要である。文献[3]の例ではSCLの周期は20μs(SCLの半周期は10μs)になっている。

これはボーレートにすると約50kbaudであり、この速度はライターの書き込み速度として特に問題ないことを確認している。

I2CReadはPICがEEPROMから1Bのデータを読み出す時に使用するコマンドである。1Bの読み出しに11SCLの時間が必要である。SCLの周期を20μsとしてもライターの読み出し速度としては特に問題ないが、漢字ディスプレイにおいて分散読み込みを行うためには速度が不足している。このため、漢字ディスプレイにおいてはSCLの周期を4μsに短縮して実行させている。この値はボーレートに直すと約100kbaudに対応する。

3.3 EEPROMの書き込み／読み出し手順

(1) 書き込み手順

図3.3(a)にPICからEEPROMへページ・ライト・モードを用いて書き込む場合の手順を示す。この手順は後述するモードeおよびモードfにおいて使用する。

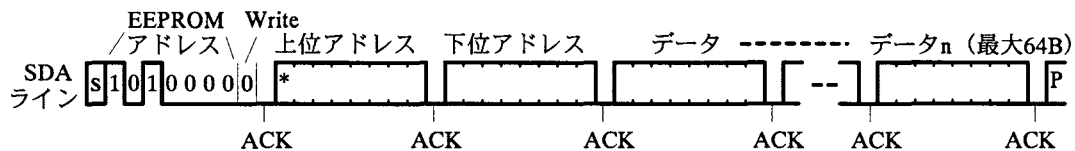


図3.3(a) EEPROMへ書き込む手順

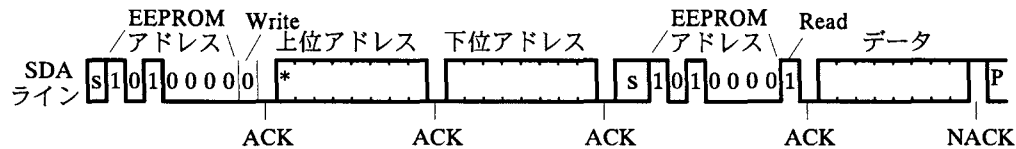


図3.3(b) EEPROMから1バイト読み出す手順

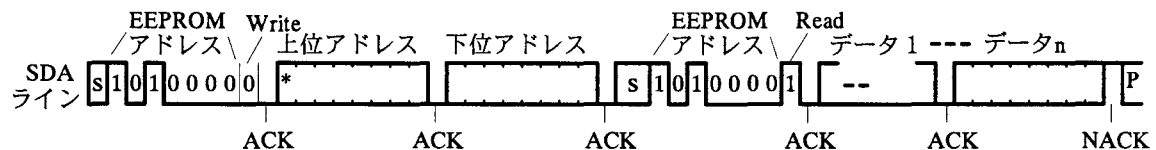


図3.3(c) EEPROMからnバイト読み出す手順

PICはマスターとして、I2CStartに引き続きI2CWriteコマンドを用いて1バイト目にコントロールバイト(1010+EEPROMの物理アドレス(0に固定)+R/W=0)を送信する。2~3バイト目にEEPROMのアドレスを15ビットで送信する。4バイト目から最大68バイト目までEEPROMから送信されるACKを受信しながら連続的にデータを送信でき、その後ストップコンディション(P)を送信する。最大送信可能なデータ量は64バイトであり、それ以上の場合には以上を繰り返す。

(2) EEPROMからの1バイト読み出し手順

図3.3(b)にEEPROMから1バイトを読み出す場合の手順を示す。この手順は漢字ディスプレイにおいて「漢字データの分散読み込み」において使用する。

PICはマスターとして、15ビットのEEPROMのアドレスを送信するために1バイト目にR/W=0としてI2CWriteコマンドを用いてコントロールバイトを送る。また書き込みの場合と全く同様に、データを読み出すためにEEPROMの15ビットのアドレスを送信する。引き続きI2CStartの後にReadモード(R/W=1)にしてI2CWriteコマンドを用いてコントロールバイトを送信し、その後I2CReadを送信して1バイトのデータを読み出す。受信完了後PICはマスターとしてNACKを送信する。

(3) EEPROMからの複数バイト読み出し手順

最初にコントロールバイトを送信し、引き続きEEPROMの15ビットのアドレスを送信し、2番目のスタートコンディション(S)としてリピートスタートコンディションを用い、コントロールバイトを送信し、I2CReadを用いるのは(2)の場合と全く同じである。違うのは、データを受信した後にNACKを送信せずACKを送信し、必要な個数のデータを受信後に初めてNACKを送信する点である。なおストップコンディション(P)の後に15msの遅延を入れるようになっている。これは非常に重要なポイントであり、この遅延がないと正しく書き込み/読み出しができないことを確認した。

3.4 PC~PIC間の通信プロトコル

図3.4にPC~PIC間の通信プロトコルを示す。文献[2]では、コマンドstart、コマンドa、コマンドd、コマンドeおよびコマンドrが定義されているが、本検討ではさらにコマンドkおよびコマンドfを追加した。またコマンドrに変更を加えた。なお漢字データはバイナリデータであり、トランスペアレントな伝送を行うためにバイナリ-アスキー変換を行って伝送している。以下にプロトコルの概要を示す。

(1) start コマンド

PCからキャリッジリターン \%r (0Dh)を送信することによりPICのプログラムが起動する。PICは \%r をエコーバックし、またプロンプト \%redp をPCに送信する。

(2) コマンド a

PCからアスキー文字“a”(61h)を送信する。PICは“a”(61h)をエコーバックし、またプロンプト \%r_address をPCに送信する。PCは4桁の開始アドレスd1~d4(dnは16進数)をアスキー文字に変換してPICに送信する。PICでは、アスキー文字をバイナリに変換後、上位桁を変数data01に、下位桁を変数data02に格納する。その後プロンプト \%redp をPICに送信する。

(3) コマンド d

PCからアスキー文字“d”(64h)を送信する。PICは“d”(64h)をエコーバックし、またプロンプト \%r_setadr をPCに送信する。引き続きコマンドaで受信した4桁の開始アドレスd1~d4(dnは16進数)をアスキー文字に変換してPCに送信し、最後にプロンプト \%redp を送信する。

(4) コマンド e

PCからアスキー文字“e”(65h)を送信する。PICは“e”(65h)をエコーバックし、またプロンプト \%r_write をPCに送信する。引き続きコマンドaで受信した4桁の開始アドレスd1~d4(dnは16進数)をアスキー文字に変換してPCに送信する。PCは送信したいアスキー文字をエディットボックスに書き込

んで送信する。データの最後を示す終結デリミタとしてキャリッジリターン $\text{\$}$ (0Dh)を送信する。

PICは受信したアスキーデータをエコーバックし、また d1~d4 で指定される EEPROM の開始アドレス以降にそのまま書き込む。キャリッジリターン $\text{\$}$ (0Dh)を受信すると書き込みを止め、これをエコーバックする。キャリッジリターンは EEPROM には書き込まない。最後にプロンプト $\text{\$redp}$ を送信する。

(5) コマンド k

PCからアスキー文字“k” (6Ch)を送信する。PICは“k” (6Ch)をエコーバックし、またプロンプト $\text{\$}_k$ をPCに送信する。引き続きコマンド a で受信した 4 桁の開始アドレス d1~d4 (dn は 16 進数)をアスキー文字に変換してPCに送信する。PCは2桁の 16 進数でエディットボックスに書き込まれた漢字個数をPICに送信する。

PIC は受信したアスキーデータをエコーバックし、2 桁目を受信後バイナリに変換し、漢字個数として変数 kanji_kosu に格納する。最後にプロンプト $\text{\$redp}$ を送信する。

(6) コマンド f

PCからアスキー文字“f” (66h)を送信する。PICは“f” (66h)をエコーバックし、またプロンプト $\text{\$}_f$ をPCに送信する。引き続きコマンド a で受信した 4 桁の開始アドレス d1~d4 (dn は 16 進数)をアスキー文字に変換してPCに送信する。PCは、ファイルからPCのメモリに読み出されている送信すべき漢字データ(バイナリ)をアスキー文字に変換してPICに送信する。送信するデータ量はコマンド k で送信した漢字個数分 (+1) × 32B である。PIC は受信したアスキーデータをエコーバックし、バイナリに変換してメモリに格納し、最後にプロンプト $\text{\$redp}$ を送信する。

(7) コマンド r

PCからアスキー文字“r” (72h)を送信する。PICは“r” (72h)をエコーバックし、またプロンプト $\text{\$}_r$ をPCに送信する。その後コマンド a で受信した 4 桁の開始アドレス d1~d4 (dn は 16 進数)をアスキー文字に変換してPCに送信する。

引き続きEEPROMから 32B × kanji_kosu 分の漢字データを読み出し、1 バイト読み出すたびにアスキー文字に変換してPCに送信する。最後にプロンプト $\text{\$redp}$ を送信する。

4. I²C シリアル EEPROM ライタープログラム

4.1 ターミナルソフトにハイパーターミナルを使用する場合の問題点

ターミナルソフトとしてハイパーターミナルを用い、3で示したPICのプログラムをPCから制御する場合には、以下のような解決すべき課題がある。

(1) バイナリデータである漢字データを格納したファイルの送信方法。

(2) コマンド k およびコマンド r を用いて EEPROM の内容を表示するとき、単にデータが 16 進数で連続的に表示されるので内容が分かりにくい。

このため、VC++を使用してターミナルソフトを作成することにしたが、これには昨年度に別の目的で作成したプログラムに多少の改造を加えることによって実現できるであろうとの思惑もあった。

4.2 RS232C 通信を扱うクラス

RS232C を使用した通信には、インターネットからダウンロードした「C++におけるRS232C通信クラス」を基本的に使わせて頂いた。以下に示したコマンドを主として用いた。

- RS232C::Connect()・・・パラメータの設定
- RS232C::Read(char* Buff, int n)・・・nバイトのデータをバッファ Buff に読み込む
- RS232C::Read_CRLF(char* Buff, int n)・・・nバイトのデータをCRLFが来るまでバッファ Buff に読み込む
- RS232C::Send(char* word)・・・アスキーデータ word の送信
- RS232C::bufinit()・・・入出力バッファのクリア

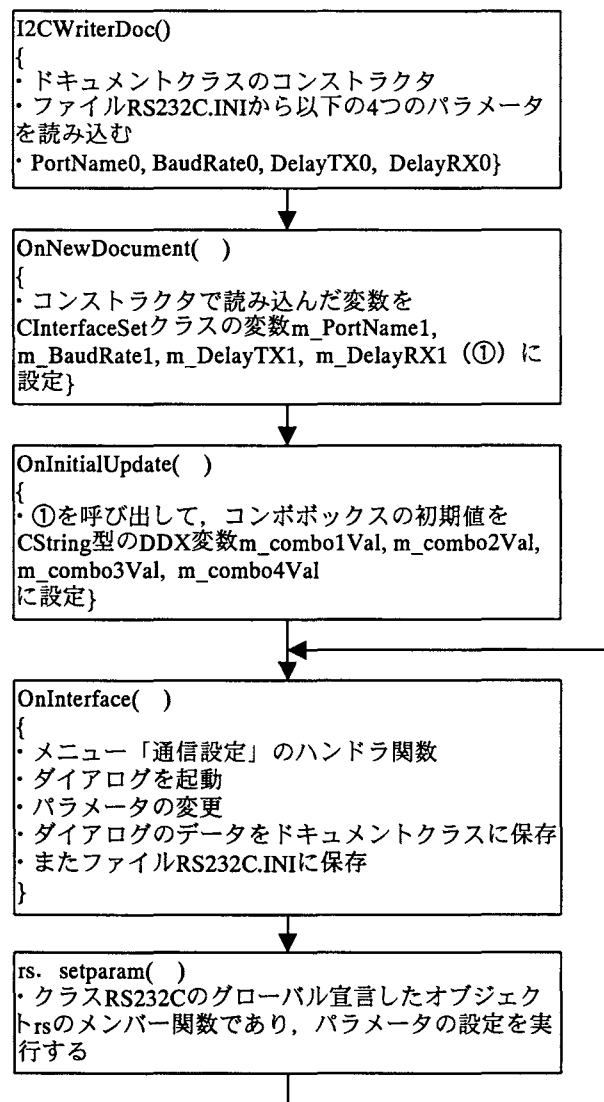


図4.1 コンボボックスによる通信インターフェースの設定

4.3 コンボボックスによる通信インターフェースの設定

メニュー「通信設定」からダイアログを起動して通信パラメータを変更できるようにするためにメンバー関数 SetParam() を追加し、使用するポート (Portname)、ボーレート (BaudRate)、送信遅延 (Delay_TX)、受信遅延 (Delay_RX) を変更できるようにした。データ長 (8 ビット)、パリティ (なし)、ストップビット長 (1 ビット)、フロー制御 (なし) は固定とした。

図 4.1 にメニュー「通信設定」からダイアログを起動し、ダイアログ上のコンボボックスに入力したデータによってパラメータの修正・変更を行うための手順を示す。

4.4 送受信プログラムの具体例

一例としてコマンド f を送信する場合のデータ送受信の流れを図 4.2 に示す。data は char* である。本来は①の命令でコロン(:)まで受信できる筈であるが、うまくいかなかったので②を追加した。③の rs.Send(SendbufASC); で漢字データ (32B バイナリ) を 64B のアスキー文字に変換して送信している。④⑤で 64B のアスキー文字を 2 回に分けて受信しているが、これは 1 回で受信できなかったためである。なお図 4.2 にはないが、クラス RS232C のオブジェクト rs をグローバル宣言している。

```
rs.bufinit(); //バッファークリア
rs.Send(data); //コマンド"f"送信

buf1[1]='\0';
rs.Read(buf1,1); //エコーバック("f")検出
m_response1 = buf1;

buf[13]='\0';
rs.Read(buf,13); //_fwrite>xxxxを検出①
m_response2 = buf;

buf2[1]='\0';
rs.Read(buf2,1); //:を検出②
m_response3 = buf2;

if(pDoc->first==1) //1回だけ実行
{
    rs.Connect();
    pDoc->first=0;
}

rs.SetParam(pDoc->GetPortName2(), //パラメータ設定
            pDoc->GetBaudRate2(),
            pDoc->GetDelay_TX2(),
            pDoc->GetDelay_RX2());

rs.Send(SendbufASC); //64Bのアスキー
pSendbufASC += SendSizeASC; //データ送信③

rs.Read(RecbufASC1, 42); //42Bを受信④
rs.Read(RecbufASC2, 22); //22Bを受信⑤

strcpy(RecbufASC,RecbufASC1);
strcat(RecbufASC,RecbufASC2); //64Bの受信データ

buf[6]='\0'; //\redp>を検出
rs.Read(buf,6);
```

図 4.2 コマンド f 送信時の具体的なプログラム

受信データのバイト数が既知の場合には、Read() を使う方が Read_CRLF() を使うよりも高速に受信できる。

4.5 ライタープログラムの実際

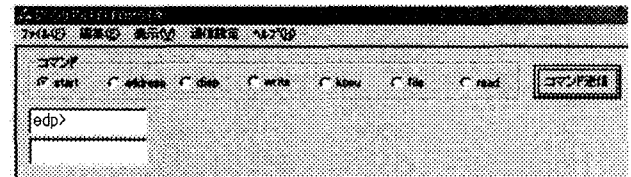


図 4.3 start コマンド→コマンド送信を押したとき

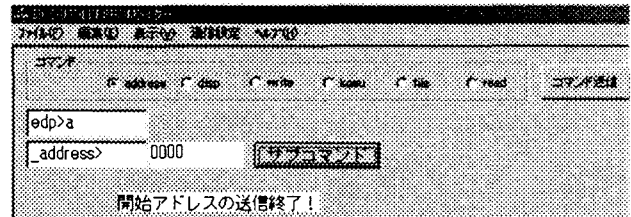


図 4.4 address コマンド→コマンド送信→0000 入力→サブコマンドを押したとき

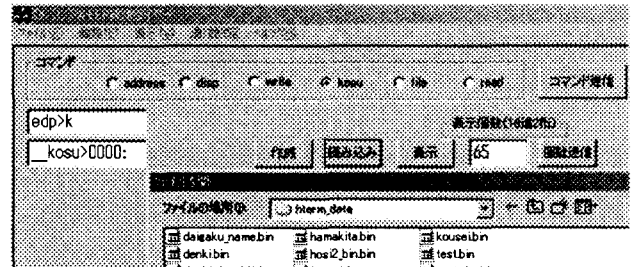


図 4.5 kosu コマンド→コマンド送信→読み込みを押したとき

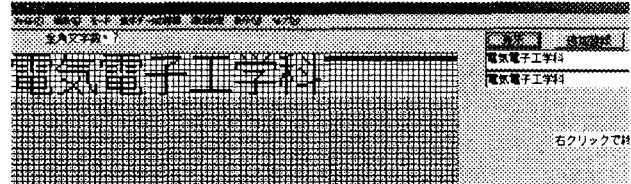


図 4.6 kosu コマンド→コマンド送信→作成を押したとき
漢字データ作成ソフト kanji_input.exe が起動される

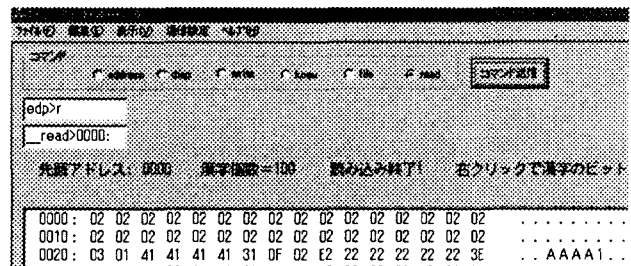


図 4.7 read コマンド→コマンド送信を押したとき

図 4.3 はプログラムを起動し、start コマンドを選んでコマンド送信ボタンを押した場合である。エディットボックスに PIC から返送されたプロンプトが表示されている。図 4.4 はコマンド a の送信結果である。図 4.5 は漢字データをバイナリファイルから読み込む場合である。図 4.6 は漢字データ作成ソフトを起動した場合である。図 4.7 は read コマンドを実行した場合である。エディットボックスにメモリの内容が表示される。

5. 漢字データの分散読み込みを適用した漢字ディスプレイ

5.1 回路構成

図3.1に示したEEPROMライターの回路構成とはほぼ同様でよい。但し従来の回路におけるポートの割り当てを幾分変更する必要がある。また図3.1と異なりWP=1に設定しておく。

5.2 24LC256 (256K I²C CMOS Serial EEPROM) について

このEEPROMはI²Cインタフェースを持つ256Kビット(32Kバイト)のメモリである。漢字データは1文字が32バイトであるから、このメモリを使えば最大1024個の漢字データを格納できる。したがってスイッチ切り替えで8組のデータに分割し、1組当たりの漢字数を最大128個とする。

アドレス長は15ビットであり、2バイトで表す。この上位バイト(ADDR_HIGH)のビット6~4の3ビットにスイッチの番号を埋め込み図5.1のようにアドレスを表すものとする(12ビット長のアドレスで4096ビット=32バイト×128個になる)。

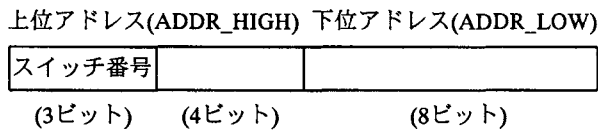


図5.1 シリアルEEPROMのアドレス構成

5.3 漢字データの読み込みに関する時間の見積もり

図5.2に漢字ディスプレイの概略フローチャートを示す。5.2で示したようにSWの値は0~7までの8組であり、SW=7のときはEEPROMからデータを読み込むのではなく、従来通りプログラムメモリに格納したデータを読み込むようにしてある。漢字データの読み込みを行うのは図5.4のシミュレーション結果に示すようにsft_kaisu=0, tcnt=0になった割り込み周期である。バッファシフトを行った後、バンク2の空いたバッファにプログラムメモリに格納した漢字データ(1個分、32バイト)を一挙に読み込んでいる。この一連の処理を次の割り込みがかかる前(start_flag=1になる前)に終えなければならない。

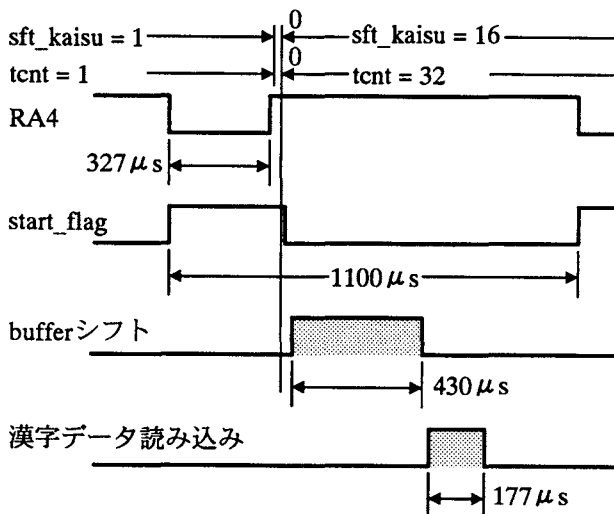


図5.4 バッファシフトおよび漢字データの読み込みに関するタイミングチャート

図5.4に示すsft_kaisu=1, tcnt=1になった直後の割り込み周期における時間関係は以下のようにになっている。

- シフトレジスタへのデータの読み込み 310 μs
- バッファシフト 430 μs
- 漢字データの読み込み 177 μs

さらに命令の実行時間もあり、このため割り込み周期を1.1msに設定している。

これより従来の漢字データ読み込み方法を用いると、漢字データの読み込みに許容できる時間は最大でも300 μs程度となり、I²CシリアルEEPROMを使用するとこの時間内に読み込むことができない。しかし読み込みの周期は回数で表して

行の点灯回数×1文字の列数=32×16=512回と長い。したがって1回の読み込みで1バイトずつ、32回に分けて読み込んでやればシリアルインタフェースを用いたEEPROMを用いても分散読み込みが十分に実現可能である。

シフトレジスタへのデータ読み込みは全ての割り込み周期において必要であり、更に命令実行時間を考慮すると分散読み込みに許容できる時間は最大でも700 μsである。

以上を考慮したフローチャートが図5.2, 図5.3である。また図5.5にタイミングチャートを示す。

5.3 漢字データの分散読み込み

(1) 電源投入直後のバッファメモリの状態

5個の16×16 LEDを使用して漢字を左シフトさせるために、電源投入直後には6個分の漢字データをバッファメモリに読み込んでいる。したがって1個の漢字がシフトされると最初のバッファシフトが行われ、バンク2のバッファメモリが空く。その後初めて漢字データをEEPROMから読み出してバンク2のバッファメモリに格納できるようになる。

1つの漢字をシフトするのに要する時間は

$$1.1\text{ms} \times 512 = 563.2\text{ms}$$

である。電源投入後563msの間はバンク2のバッファはまだ空いていないのでEEPROMからの漢字読み込みを禁止する必要がある。このため変数delay_565msを導入した。図5.2および図5.3から電源投入後delay_565ms=0としておき、最初にsft_kaisu=0, tcnt=0になった直後にdelay_565ms=1としこれ以後は同じ状態を保つようにプログラムすることで対処した。

(2) 分散読み込みのタイミング

delay_565ms=1になった以降は1バイトずつ32回に分けて32Bを読み出せば良い。但しsft_kaisu=16, tcnt=32のタイミングでバッファシフトを行うので、sft_kaisu=16は避ける必要がある。したがってsft_kaisu=15, tcnt=32~tcnt=1において分散読み込みを行うことにした。

(4) 読み込み時のポーレートの見積もり

EEPROMから1B読み出すには、図3.3(b)に示した1Bランダムリードの手順に従う。図3.2からI2CWriteおよびI2CReadの

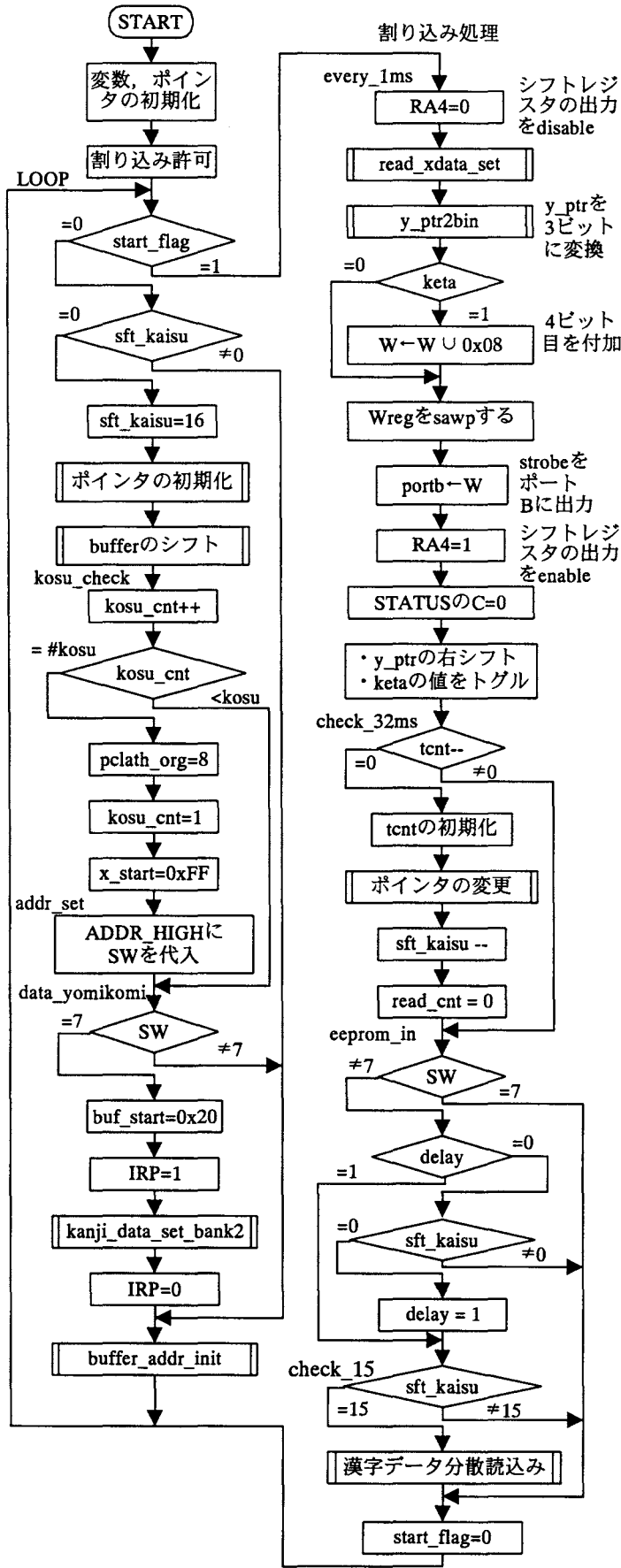


図5.2 漢字ディスプレイの概略フローチャート

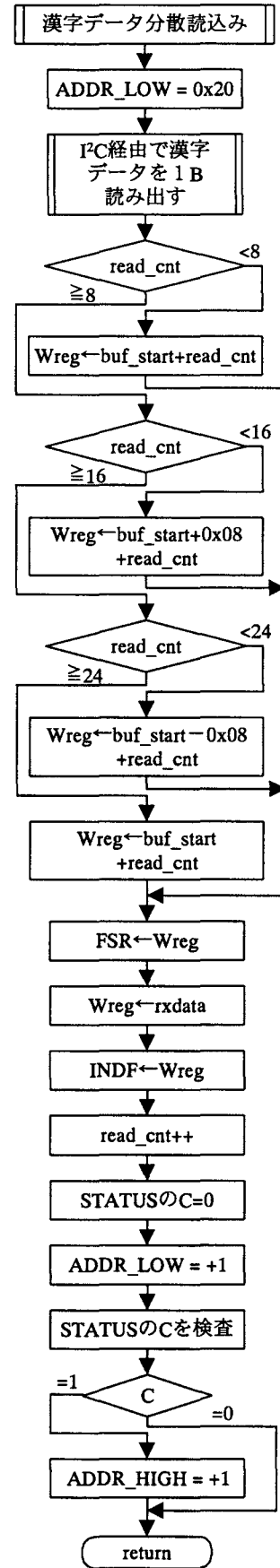


図5.3 漢字データの分散読み込み

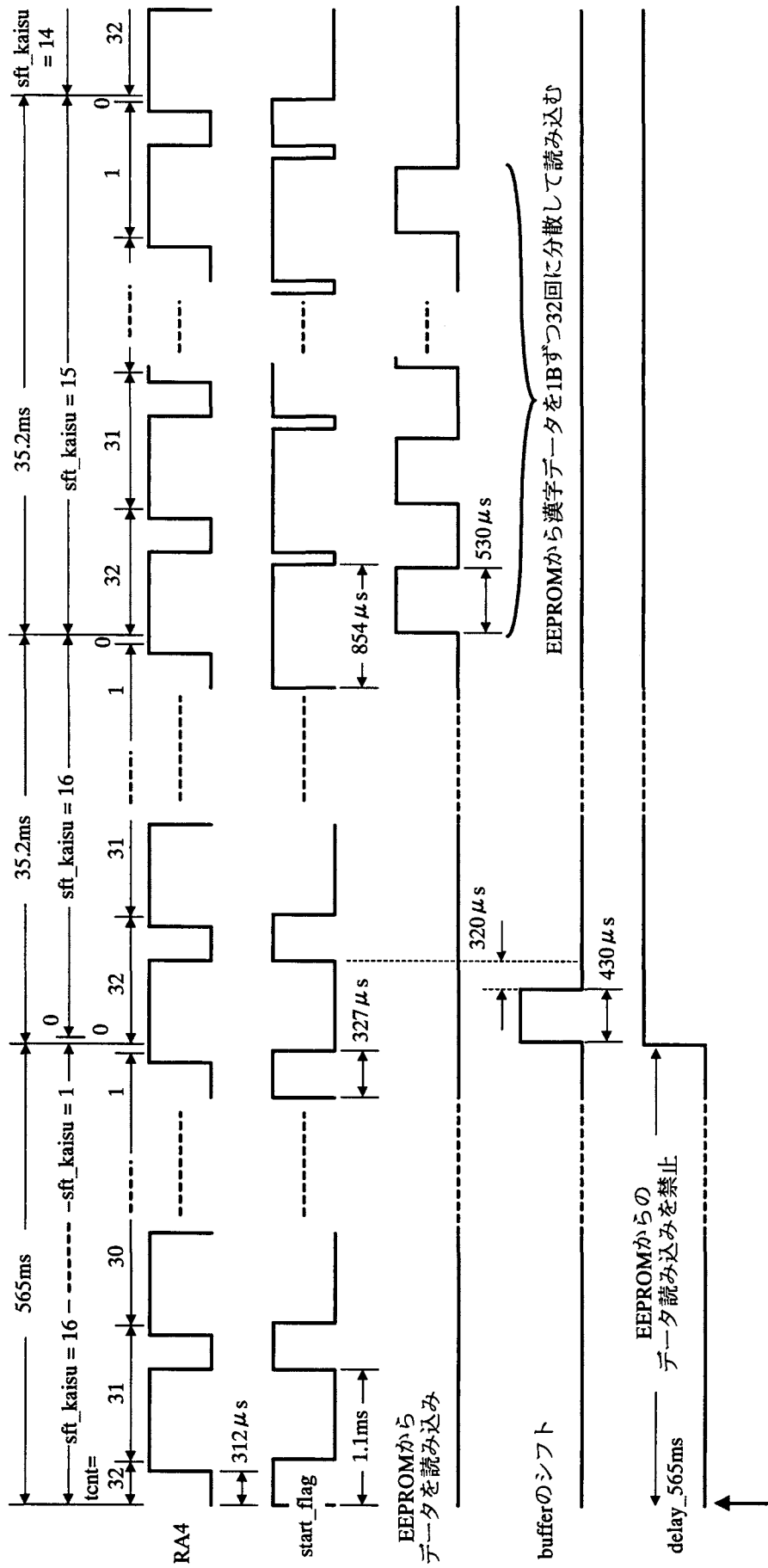


図5.5 漢字ディスプレイのタイミングチャート

リセット後無限ループの実行を開始した時点

実行時間はSCLクロック換算で10クロックおよび11クロックである。したがって図3.3(b)の1Bランダムリードには単純計算で51 SCLの時間がかかるが、命令の実行時間を考慮して100 SCLと考えることにする。

文献 [3] では命令の実行時間を除く SCL の周期は $20\mu s$ であり、100 SCL では $20\mu s \times 100 = 2000\mu s$ となり許容値の $700\mu s$ を超えてしまう。したがって SCL の周期を短くする必要がある。いま SCL の周期と 100 SCL の計算を表 5.1 に示す。ポーレートは 2 SCL の逆数の計算結果とした。

表5.1 SCLの周期とポーレートの関係

SCLの周期(μs)	100 SCL(μs)	ポーレート(kbaud)
10	1000	50
8	800	63
6	600	83
4	400	125

24LC256 の最大クロック周波数は 400kHz であるから、SCL の周期を $4\mu s \sim 6\mu s$ に選べば良さそうである。今回は、多少ポーレートは高くなるものの、時間の設計に余裕を持たせて命令の実行時間を除く SCL の設計周期を $4\mu s$ に選んだ。このときシミュレーション結果では図 3(b)を実行して 1B のランダムリードにかかる時間は $530\mu s$ であった。

なお EEPROM ライターにおいては、SCL の設計上の周期を $20\mu s$ としているが、この値で特に問題はなかった。

5.4 漢字 ROM の作成

ライターを用いて作成した漢字 ROM の内容を表 5.1 に示す。スペースの関係で最初の 4 組のみを示したが、4000H 以降に格納したデータも正しく格納できることを確認している。なおライターには、PIC16F628A を 10MHz クロックで使用した。

表 5.1 漢字 ROM の内容 (最初の 4 組のみ)

アドレ	内容
0000H	この電光掲示板は、下から順番に一行単位でLEDを点灯させ、目の残像現象を利用して文字を表示しています。このようなやり方は、ダイナミック点灯と言われています。(78文字)
1000H	WELCOME! 電気電子工学科 電気電子技術は身近な所にたくさん隠れています。今日は、その一部を見ていって下さい。(59文字)
2000H	歓迎! 高校生の皆さん。SIST説明会へようこそ。電気電子の面白さの一端を体験して下さい。(46文字)
3000H	!! 歓迎!! 高校生の皆さん。電気電子工学科へ入学して、電気電子の面白さを一所懸命に勉強しよう。(47文字)

5.5 実行結果

表 5.1 に示した漢字 ROM を漢字ディスプレイに搭載して表

示試験を行い、正しく動作することを確認した。図 5.6 に SDA ラインと SCL ラインのモニタ結果を示す。図 5.2 のフローチャートにおいて信号delayを空きポートであるRB3端子に出力し、この信号によりトリガーをかけて 1B ランダムリードによる分散読み込みの結果を観測した。EEPROM のアドレスは 30E1H であり、最初に EEPROM から読み出される漢字「高」の第二バイト (7FH) が正しく読み出されている。SDA ラインおよび SCL ラインとも 100kbaud の伝送速度において良好な応答波形が得られている。

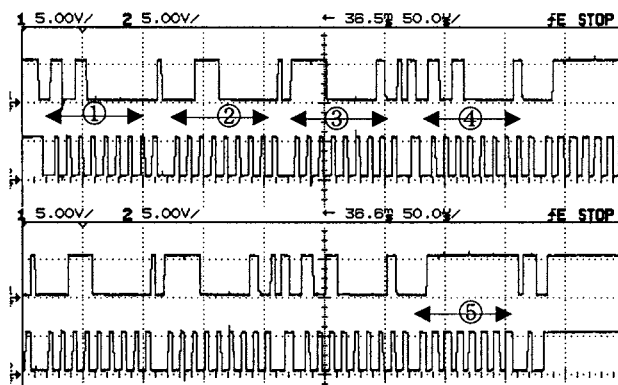


図 5.6 SDA ラインおよび SCL ラインのモニタ結果

(上段: SDA, 下段: SCL, ①: コントロールバイト(R/W=0), ②EEPROM 上位アドレス:30H, ③下位アドレス:E1H, ④: コントロールバイト(R/W=1), ⑤データ: 7FH)

6. まとめ

高校生に電気電子に対する興味を喚起して貰うことを狙いとして、H20 年度に漢字表示電光掲示板 (漢字ディスプレイ) を試作した。本論文では I²C シリアル EEPROM を用いてその表示内容を増大させるために行った検討結果を示した。

- (1) PIC マイコンから I²C シリアル EEPROM にデータを書き込み/読み出しする手順を整理して示した。
- (2) RS232C インタフェースを用いて PC から PIC マイコンを介して I²C シリアル EEPROM に書き込み/読み出しするライタープログラムを検討し試作した。
- (3) 漢字ディスプレイに I²C シリアル EEPROM を搭載するためには、時間の関係から「分散読み込み」をする必要のあることを示し、実際に漢字ディスプレイに適用して正しく動作することを確認した。

【参考文献】

[1] 袴田吉朗, “PIC マイコンと 16×16 LED を用いた漢字表示電光掲示板の設計と試作”, 静岡理科大学紀要, Vol.17, pp.133-142 (2009)

[2] 神崎康宏, “作りながら学ぶ PIC マイコン入門”, CQ 出版社, 2007 年

[3] 中尾 司, “マイコンの 1 線 2 線 3 線インターフェース活用入門”, CQ 出版, 2007 年

[4] 24AA256/24LC256/24FC256 256K I²C™ CMOS Serial EEPROM, Microchip