マイコンH8/3052Fと16×16 緑色LEDを用いたキャラクタディスプレイ

Design and Trial Manufacture of a Character Display with an H8/3052F Micro Computer and 16×16 LED Arrays

袴田 吉朗*

Yoshiro HAKAMATA

Abstract: The material describes the design of a Character Display that uses an H8/3052F micro computer as a controller and 16×16 green LED arrays. The controller receives ASCII codes from a Green LED transmission system via a plastic optical fiber through RS232C protocol. And it displays characters into green 16 ×16 LED matrix arrays using dynamic ON-OFF technique. A few know-how to use HEW simulator debugger is also taken into account.

1. はじめに

2011年3月11日に東北大震災が勃発し，東北地方の物流が一時停滞した．その結果世界の自動車の生産量が低下したが，その一因にH8マイコンの供給ができなくなったことが挙げられている．このとき初めてH8マイコンが世界の自動車産業に占めるシェアが非常に大きいことを認識したものである．

筆者は数年来DSPやPICマイコンを使用して電子回路の開発を行ってきた．特にPICマイコンは，形状が小さく通常のICと同様に実装場所を選ばない特徴がある．この点に大きな魅力を感じ種々の回路の製作に適用してきた．一方市販のH8マイコンは通常ドータボードに搭載されており(搭載せざるを得ず)，これをマザーボードに乗せて使用するため，どうしても形状が大型化する．この点で使用するのにためらいがあった．しかし18ピンのPICマイコンは語長が短く，またRAM容量が小さいためページ切り替えや，バンク切り替えなどに余分な注意を払う必要がある．この点に不満を感じていたことも事実である．

折しもH23年度の後期から「マイクロプロセッサ応用」の講義を担当することになった．この講義のテキストは，H8マイコンのアセンブラを題材にしている．講義を行うためにはH8マイコンについて十分に知っておく必要がある．それにはH8マイコンを用いた回路を作ってみるのが早道であり，またマイコンを理解するにはそのマイコンのアセンブラ・プログラミングを一度は経験しておく必要があると考えている． そのため，製作するのに相応しい題材を検討してみた．

ここ数年高校生に対する電気電子への興味を喚起する目的で“動く”，“光る”，“音が出る”などの要素を含む回路を試作してきた．その一つにH21年度に試作した赤／緑色LEDを使用するWDM光通信システムがある［1］．この赤色通信系では試作した16×16 LED漢字ディスプレイ［2］に受信結果を表示している．一方緑色通信系では市販の小型LEDディスプレイに表示しているだけであり，この通信系においても緑色LEDを用いたディスプレイがあればデモ効果が上がるのではないかと考えたところである．

2012年3月2日受理

*理工学部 電気電子工学科

緑色通信系は数字やアルファベットを伝送する通信系として構成している．そこでH8/3052Fを緑色通信系の送信結果を表示するキャラクタディスプレイのコントローラに適用してみることにした．H8マイコンを使用するのは全く初めてであり，以下に示すような種々のハードルを超える必要があった．

① シミュレータ／デバッガHEWの使用方法．特にモード7におけるアドレスの設定方法
② モニタプログラムを用いたデバッグの方法．特に擬似割り込みを発生させる方法
③ タイマ割り込みとシリアル通信割り込みを混在させてモニタプログラムにより動作させる方法

7月の第1次オープンキャンパスまでに装置を完成させることを目指したが，結局8月末まで完成がずれ込んだ．その後大学祭の研究室開放において装置を実際に展示することができた．今回資料にまとめるに当たり再度プログラムを見直した．半年も立つと忘れている部分もあり，また冷静に見てみると不要な処理や可読性の悪い処理もあった．これらの点を再考してプログラムを整備し，動作を確認した結果を整理して資料に反映させた．またHEWに関するノウハウも合わせて示した．

2. キャラクタディスプレイのハードウェア構成

(1) 設計目標

以下を設計目標とした．

- コントローラとしてH8/3052F（モード7）を使用する
- 16×16ドットの緑色LEDマトリックスアレイを5個使用して，8ビットフォントのキャラクタ（数字，アルファベット）を10文字同時に表示する
- 左シフトにより，10文字以上を表示できるようにする
- 表示するキャラクタはRS232C通信により緑色通信系から受信する

(2) 回路の外観

図2.1に試作した回路の外観を示す．中央に5個の16×16 LEDディスプレイを配置している（①）．下部左の子基板がH8マイコンである（②）．LEDの下部に8ビット長のシフトレジスタ10個を細長く配置し（③），その上にLEDの電流制限抵抗（75Ω×80個）を配置している（④）．LEDの上部には，シ

フトレジスタ×2 (⑤), 論理反転用のバッファ (⑥), LEDをドライブするダーリントン・ソース・ドライバ (⑦) を配置している. 金属筐体により回路を保持している.

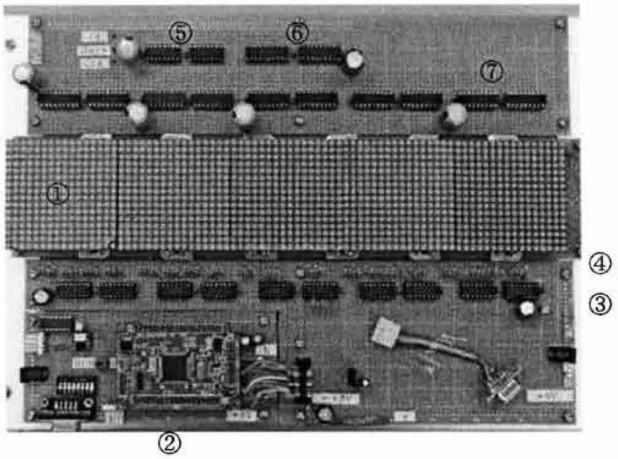

図2.1 装置の外観写真

(3) 回路構成

図2.2にLED表示回路の回路図を, 主要部品を以下に示す.

- 16×16 LEDディスプレイ (×1)
- H8/3052F マイコン (モード7)
- 25MHz水晶振動子
- 74HC4094 8ステージ・シフトレジスター (×2) X軸選択
- 74HC4094 8ステージ・シフトレジスター (×2) Y軸選択 (行選択)
- 74HC541 反転ドライバ (×2)
- TD62783AP 8CH高電圧ダーリントン・ソース・ドライバ (非反転) (×2)

回路図および上記の部品はキャラクタ4個分 (LEDディスプレイ 1個分) のものである. LEDディスプレイ, X軸選択シフトレジスタおよびドライバは, 他に4×4キャラクタ分の部品が必要である. 反転ドライバの出力はバス接続になっている.

X軸のビット数は80ビットであるが, 全シフトレジスタを直列にせず, 2個ずつに分けて16ビット分のデータを入力するようにした. LEDの輝度が期待したほど明るくなかったので, ダーリントン・ソース・ドライバの電源電圧を6Vに増大させた. 他の回路には3端子レギュレータにより5Vに降圧して印加した. なおH8マイコンのドータボード上には3端子レギュレータが搭載されているが, このICの入力端子を回路から取り外し, マイコンには直接5Vを印加するようにしている.

図2.3にマザーボードの配線図を示す. 使用しているポートは, ポート1 (P1) およびポート2 (P2) だけである. またRUN/BOOTを切り替えるスイッチを付けている.

シリアルポートSCIはモニタ用にSCI1を, 緑色通信系の受信にSCI0の2チャネルを用いている. フォトカップラーを介して電源を分離するために緑色通信系の出力は5Vの差動出力としている. インバータを2段介したのは, フォトカップラーのコレクタが完全にLレベルにまで低下しなかったためである. このTTLレベルのSCI0入力を直接ポートP92に入力している.

図2.4は緑色通信系におけるRX (受信機) との接続図である. 受信機PIC16F84Aには表示器としてLCDも接続しており, 回路全体を作り直すには手間の面でも, 時間の面でもタイトであった. そこで新たにPIC16F628Aを追加し, フォトダイオードの出力を取り出して, 緑色通信系のボーレート100baudを19.2kbaudに変換してRS232C通信を用いてバースト伝送しディスプレイに入力するようにした.

(4) タイミングの設計

図2.5にタイミングチャートを示す. 1.5msの周期はタイマ割り込みを使用して作成している. シフトレジスタの出力をハイインピーダンス状態に設定している約0.2ms間にシフトレジスタにクロックを送り, データをポート1に出力してシフトレジスタに格納する. このときのクロックの幅は1.3μs, 周期は約13μsである. シフトレジスタで電流をシンクするために, VRAMにおけるデータの反転を出力している. なおシフトレジスタのQS2を次段のD入力に接続しているが, これはQS2がクロックの立ち上がりより半クロックだけ位相が遅れて出力され, 次段のクロックの立ち上がりが緩くてもミスカウントすることのないように配慮して設計されているためである.

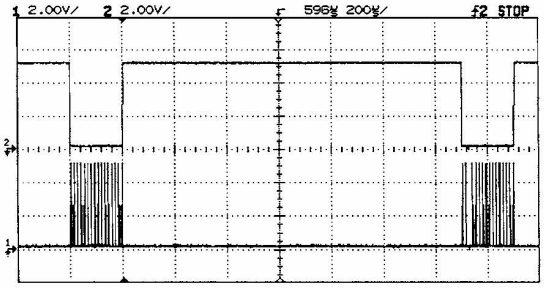

図2.6 LED出力波形 (OE) およびシフトクロック波形

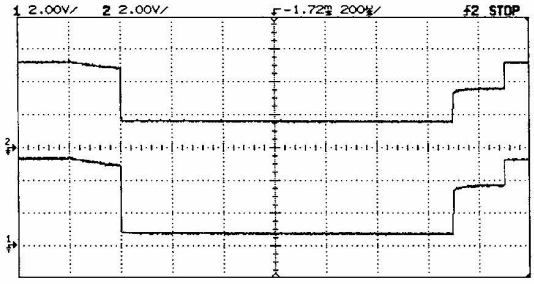

図2.7 LEDがON時のLEDカソードの波形 (上) およびシフトレジスタの出力波形 (下)

図2.6の上段は, 図2.4におけるLED出力波形 (OE) である. 下段はシフトクロックの観測波形である.

図2.7の上段はLEDが点灯しているときのLEDのカソード波形であり, 下段はシフトレジスタの出力波形である.

(5) 電流制限抵抗

当初期待したほどLEDの輝度がなかった. そこで電流制限抵抗を75Ωとし, ピーク電流を20mAにまで増大させ対処した.

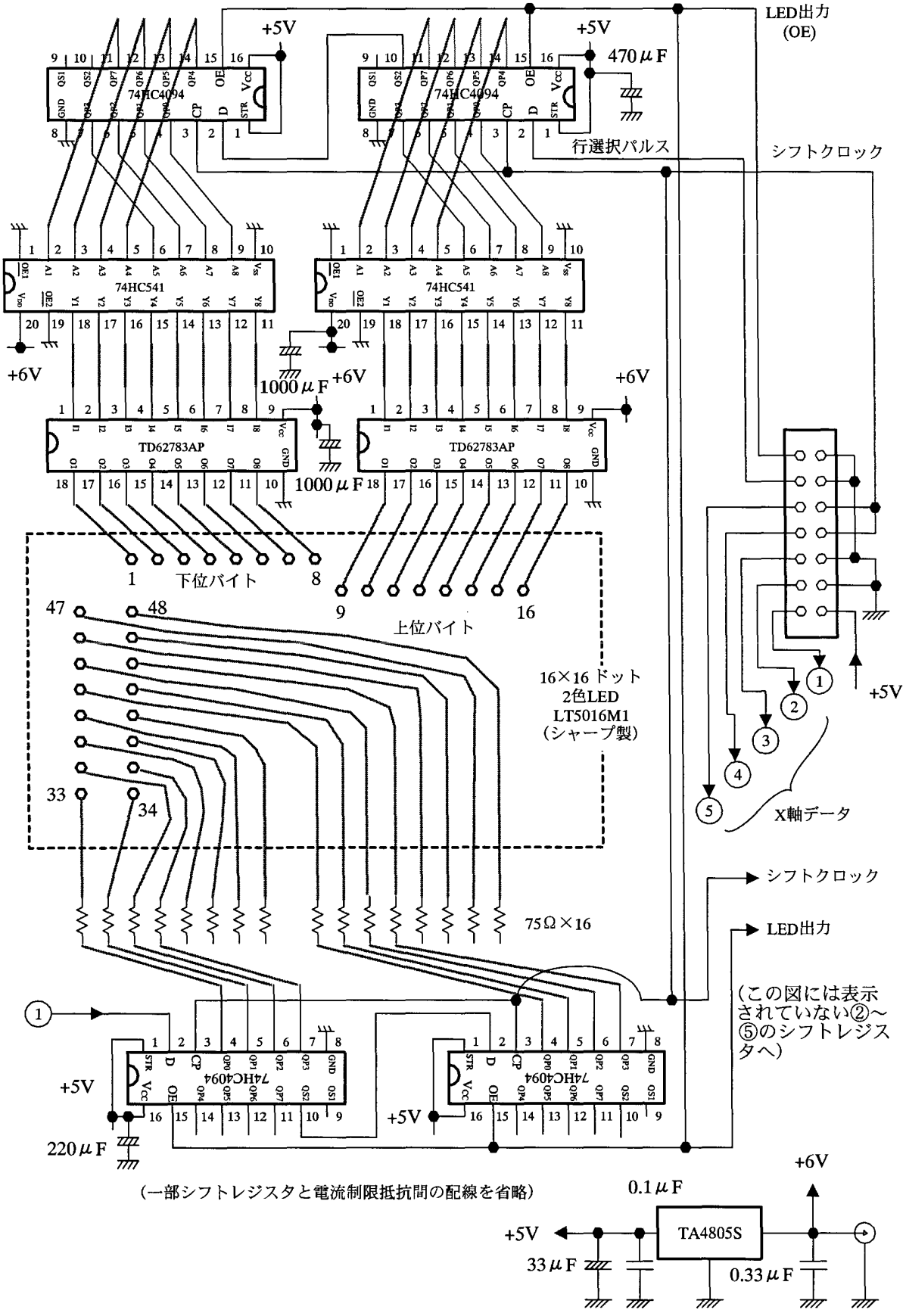

図2.2 LED表示回路ボードの配線図（bottom view）

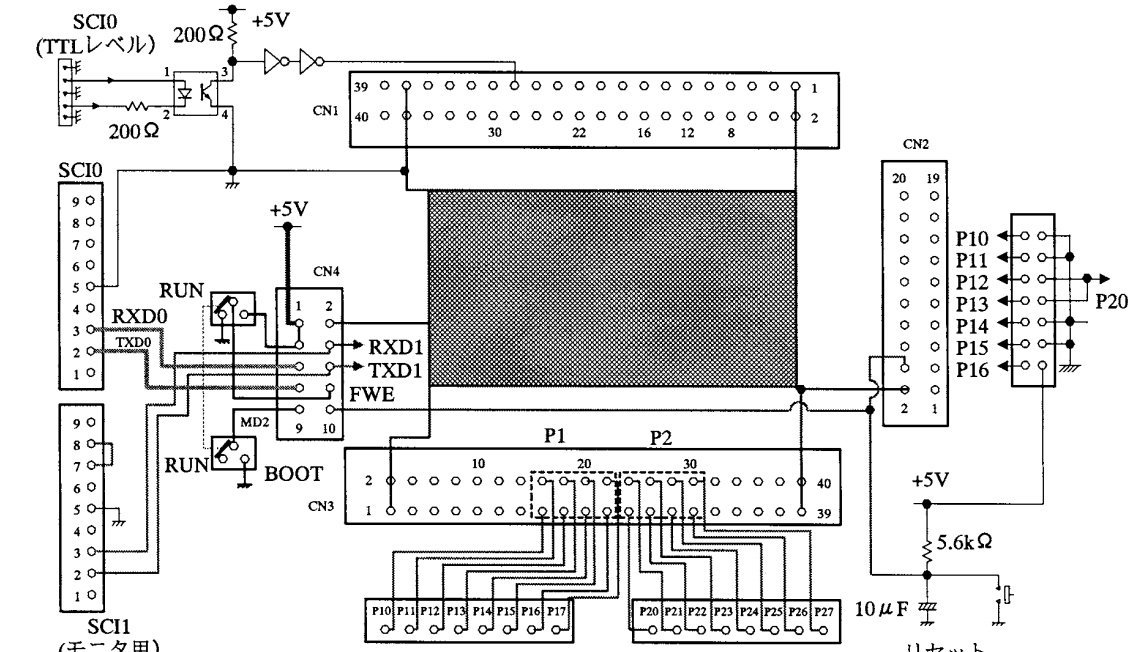

図2.3　マザーボードの配線図（top　view）

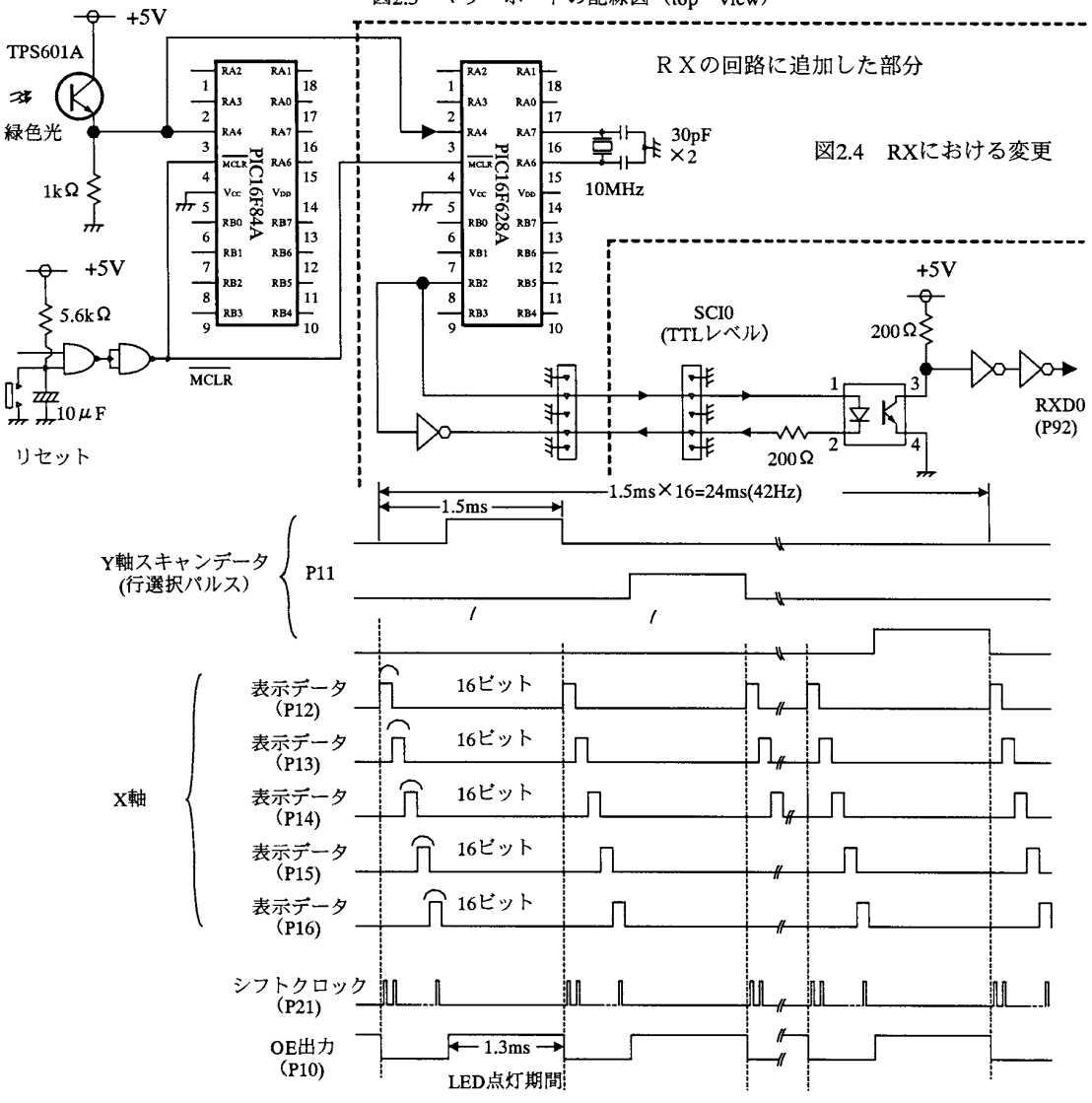

図2.4　RXにおける変更

図2.5　LED点灯に関するタイムチャート

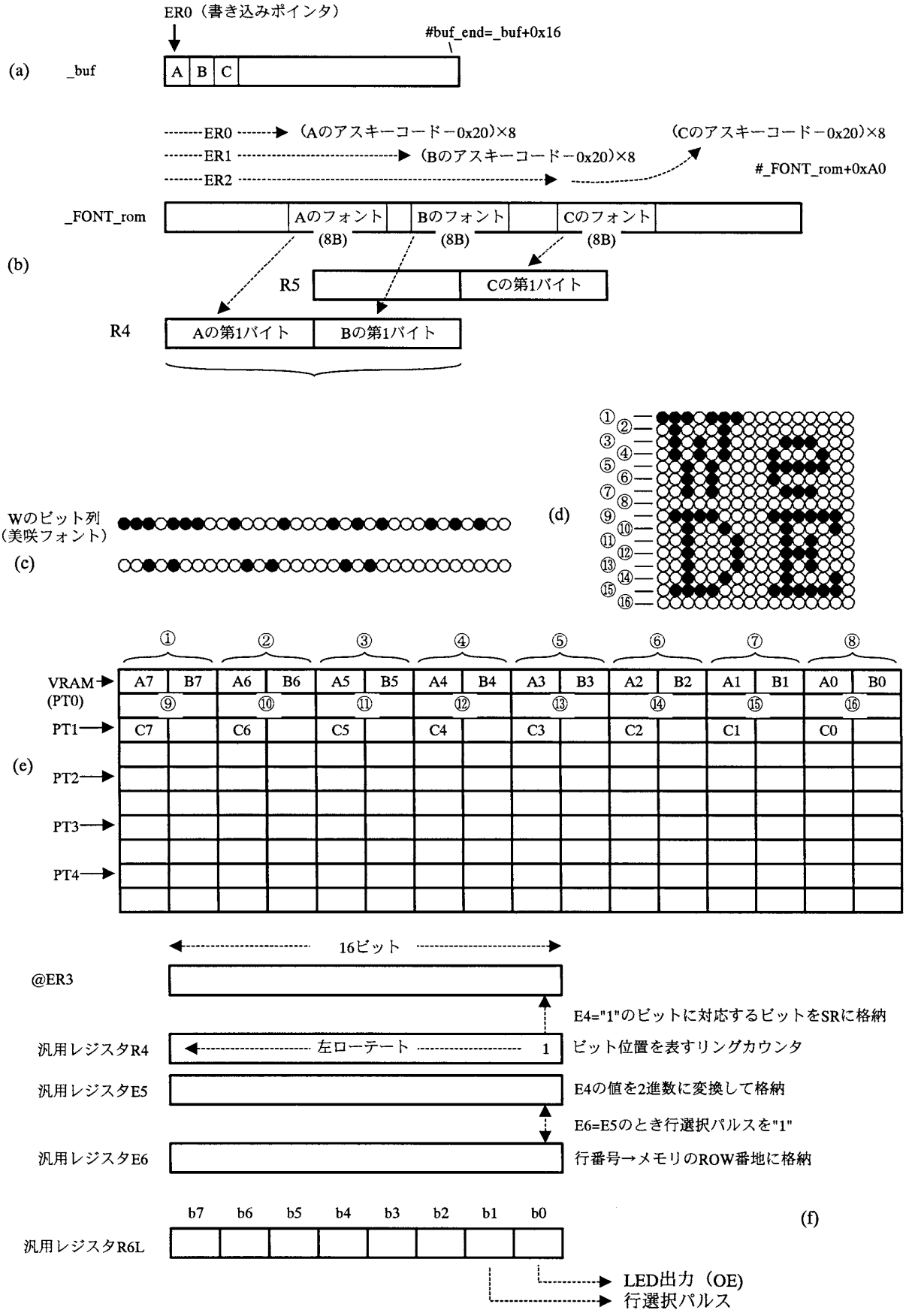
ER0（書き込みポインタ）
#buf_end=_buf+0x16
(a) _buf
A B C
ER0 (Aのアスキーコード−0x20)×8
ER1 (Bのアスキーコード−0x20)×8
ER2
(Cのアスキーコード−0x20)×8
#_FONT_rom+0xA0
_FONT_rom
Aのフォント (8B)
Bのフォント (8B)
Cのフォント (8B)
(b)
R5
Cの第1バイト
R4
Aの第1バイト
Bの第1バイト
Wのビット列（美咲フォント）
(c)
(d)
①②③④⑤⑥⑦⑧⑨⑩⑪⑫⑬⑭⑮⑯
(e)
VRAM (PT0)
PT1
PT2
PT3
PT4
A7 B7 A6 B6 A5 B5 A4 B4 A3 B3 A2 B2 A1 B1 A0 B0
C7 C6 C5 C4 C3 C2 C1 C0
16ビット
@ER3
E4="1"のビットに対応するビットをSRに格納
汎用レジスタR4
左ローテート
1
ビット位置を表すリングカウンタ
汎用レジスタE5
E4の値を2進数に変換して格納
E6=E5のとき行選択パルスを"1"
汎用レジスタE6
行番号→メモリのROW番地に格納
b7 b6 b5 b4 b3 b2 b1 b0
(f)
汎用レジスタR6L
LED出力（OE）
行選択パルス

図3.1 処理の流れ

３．プログラムの構成

(1) 動作の概要

動作の概略は以下のようになる．なお図 3.1 に処理の流れをバッファやレジスタと関連づけて示した．

① 緑色通信系の RX から RS232C 通信によりアスキーコードを受信し，これを受信リングバッファ_buf に書き込む．

② 受信したアスキーコードを 8 ビットフォントの美咲フォントに変換し，リングバッファにおいて連続する 3 文字分をバイト単位で汎用レジスタ R4 および R5 に格納する．

③ R4 および R5 を用いてビットを左に 1 ビットシフトし，その結果（R4）をバッファ VRAM に格納する．

④ 汎用レジスタ R4 には 1 を循環させている．この 1 のビット位置に対応するフォントデータを順番に取り出し，汎用レジスタ R6L を経由してシフトレジスタに格納する．

⑤ 同様にして行選択パルスもシフトレジスタに格納する．

以下①～⑤を繰り返す．

(2) UART（RS232C 通信）によるデータの受信

図3.2 にシリアル通信 SCI0 を用いたデータ受信サブルーチンのフローチャートを示す．最初に緑色通信系の手順に従って 5 個の＊を受信（後方保護 5 段）してからデータの受信に移行し，順次リングバッファである_buf に格納する．バッファ長は 20 バイトである．

SCI0 の割込優先レベルは 53 であり，タイマ割込 IMIA0 の優先レベル 24 よりも低い．このためタイマ割込が生起して LED のダイナミック点灯の処理が行われているときに，シリアルポートに着信があり SCI0 割込が生起した場合には，タイマ割込が終了するまで待たされてから SCI0 割込の処理が行われる．

緑色通信系では RS232C 通信を用いてボーレート 100baud のデータ伝送を行っている．これを PIC16F628A においてソフトウェア UART を作成してセンスループにより受信し（図 3.2），PIC におけるハードウェアにより 19.2kbaud にボーレートを変換してディスプレイに出力するようにしている．

(3) タイマ割込の処理

本ディスプレイはタイマ割込と SCI0 割込の両方を使用する多重割込を使用している．これを実現するために優先レベルの高いタイマ割込の ISR の中に関係する全処理を記述し，タイマ割込実行中に優先レベルの低い SCI0 割込が生起したときには，前述のように SCI0 割込を待ち合わせるようにしている．

図 3.3 にタイマ割込に関するフローチャートを示す．主な処理は LED のダイナミック点灯と，SCI0 受信バッファから VRAM へのデータの書き込みおよびビットシフト処理である．

(4) SCI0 受信バッファから VRAM へのデータの書き込み

H8 マイコンは図3.3 におけるタイマ割込の処理を 1.5ms 毎に繰り返しながら行単位で LED をダイナミック点灯させる．この割込の回数を変数 Gamen_cnt でカウントしており，50 回カウント後に SCI0 受信バッファからデータを読みだし，VRAM に書き込む．この処理は図 3.4 に示すサブルーチン Save_VRAM の中の「アドレスの設定」の処理において行っている．

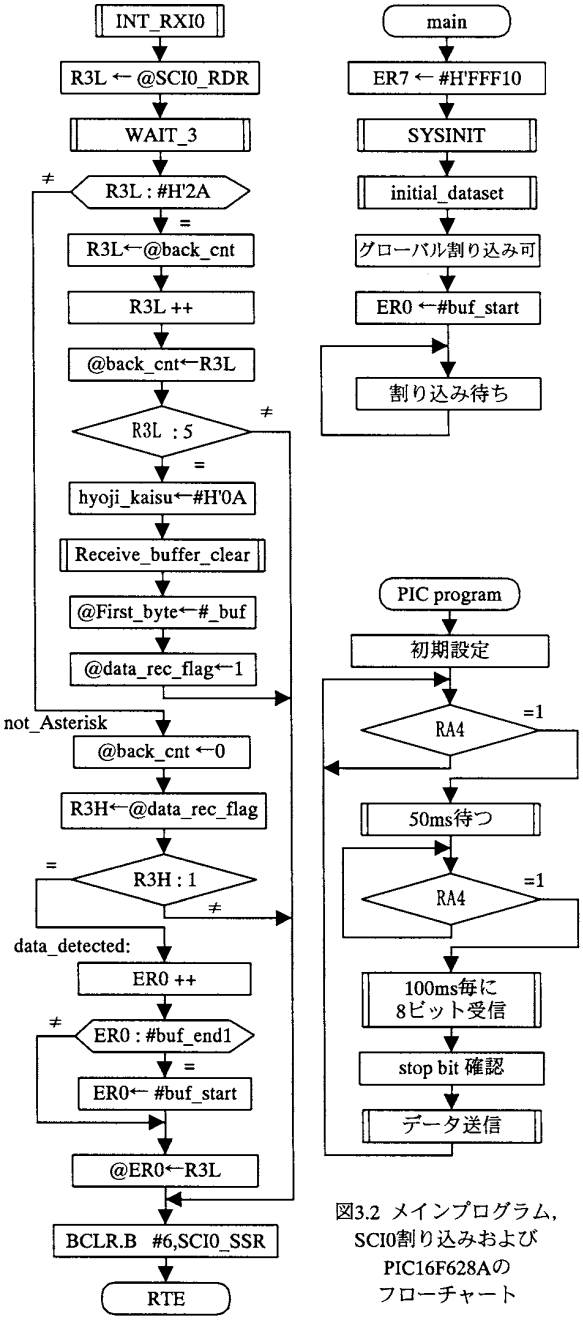

図3.2 メインプログラム，SCI0割り込みおよびPIC16F628Aのフローチャート

このサブルーチンの処理は以下のようになる．

① 受信バッファから ER4 のポイントする値（アスキーコード）を取り出し，VRAM の先頭アドレスに格納しているスペースのアスキーコード（0x20)との差を取る．

② この値を 8 倍して_FONT_rom の先頭アドレスに加え，これを第 1 文字のアドレスとしてレジスタ ER0 に格納する．

④ 第 2 文字ついても同様に行い，取り出したアドレスをレジスタ ER1 に格納する．

⑤ 第 3 文字についてはアドレスをレジスタ ER2 に格納する．

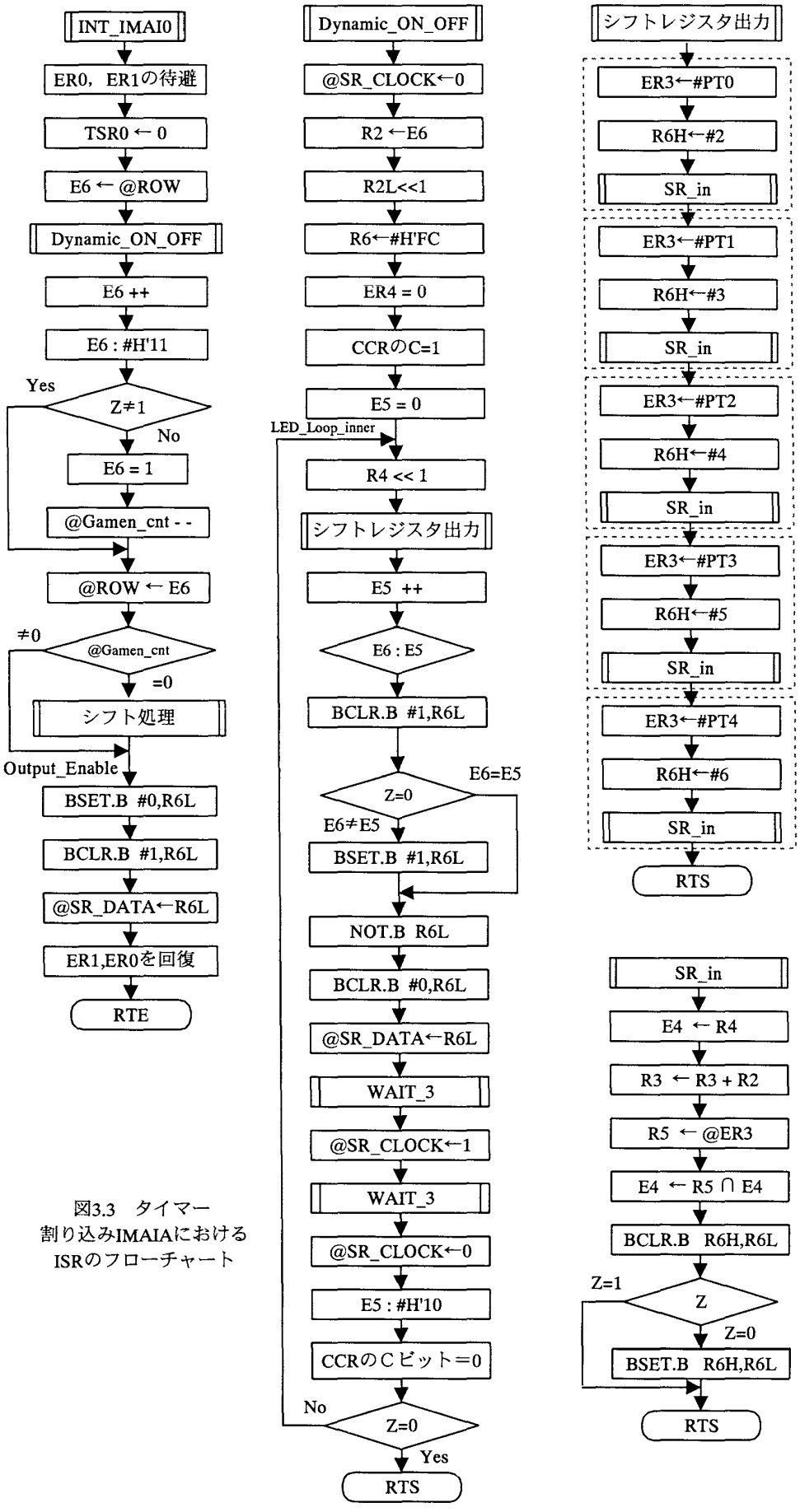

図3.3 タイマー
割り込みIMAIAにおける
ISRのフローチャート

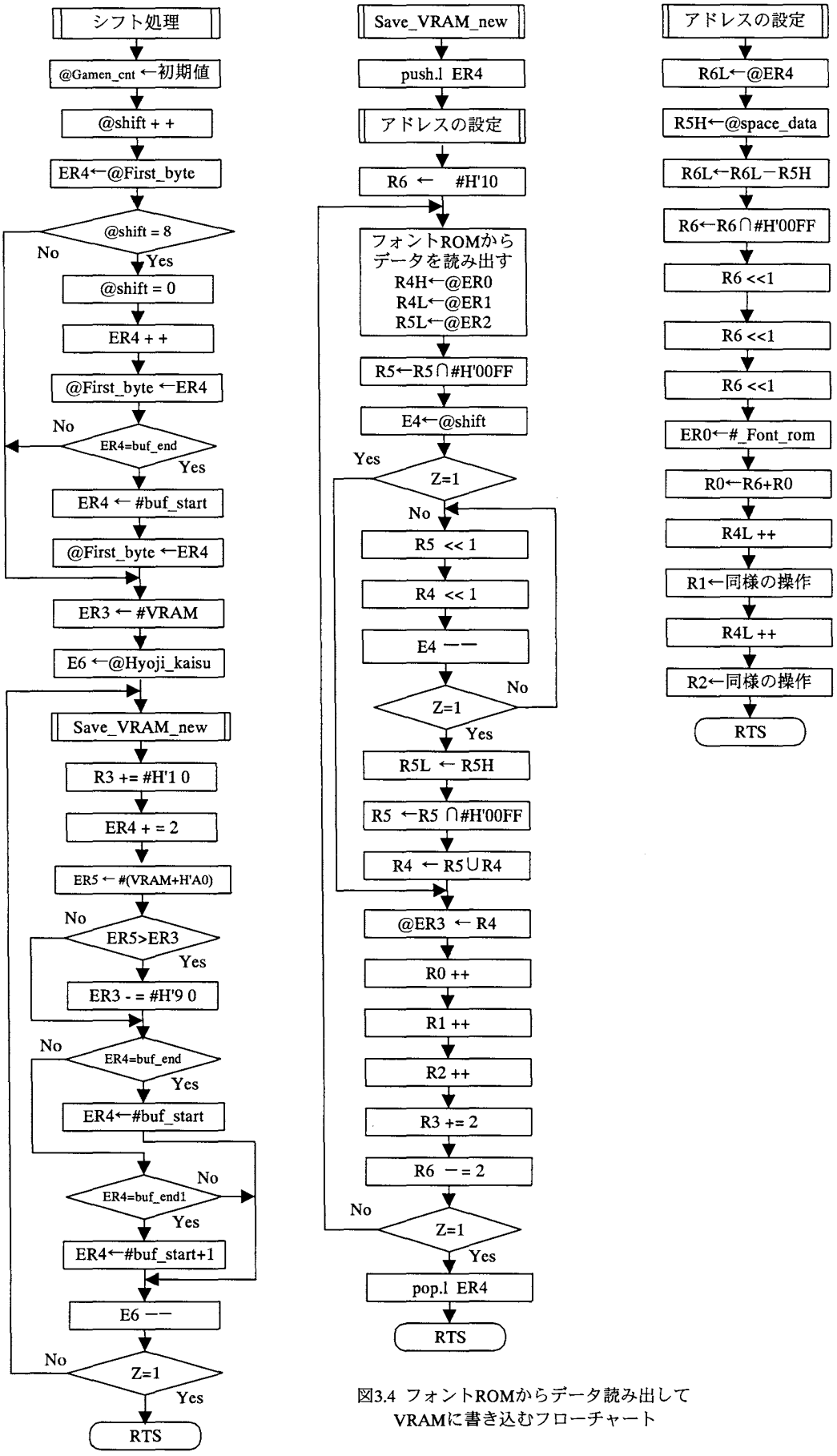

図3.4 フォントROMからデータ読み出して
VRAMに書き込むフローチャート

(4)で行ったアドレスの処理に従って，サブルーチンSave_VRAMにおいてフォントROMから3文字分のデータを読み出し，レジスタR4Hに第1文字のバイト，R4Lに第2文字のバイト，R5に第3文字のバイトを格納する．

その後レジスタの内容を左に1ビットシフトし，論理和を取ることによりVRAMに書き込むためのデータを作成する．シフト回数はメモリアドレスのshiftに格納しており，シフト処理が1回コールされる度に1だけインクリメントし，7回シフトすると元に戻すようにしている．

(6) 初期画面の表示（上半分のみのシフト）

電源投入直後にはまだ緑色通信系からデータが入力されていない．このためデータを受信するまでは以下のように表示する．

① 上半分には「Waiting for DATA!!」を左シフトしながらサイクリックに表示する．

② 下半分には，「DENKIDENSHI」を固定的に表示する．

この処理は以下のように行っている．

まず①は，図3.4におけるシフト処理のフローチャートにおいて，下半分のループの回数を決める変数Hyoji_kaisuにより制御する．この値を5にすると上半分だけがシフトし，10にすると上下が丁度アルファベットのZのような順序で左シフトするようになる．

②は電源投入後の初期設定において，以下のように2回に分けてデータをVRAMに書き込み実現している．

③ まず上半分がダミー，下半分が「DENKIDENSI」であるデータをVRAMに書き込む．

④ 次にHyoji_kaisuに5をセットして「Waiting for DATA!!」をVRAMに書き込む．

(7) VRAMへのデータの格納

レジスタR4に格納されたビット処理の結果をVRAMの対応するアドレス（レジスタER3がポイントする）に格納する．その後レジスタR0，R1およびR2を1だけインクリメントし，後続の処理に備える．

VRAMには2バイト単位で書き込むので，レジスタR3を2だけインクリメントする．

(8) 8×8ビットフォントおよびVRAMへの格納状況

8×8ビットフォントにはフリーフォントである美咲フォントのうち「数字，アルファベット，スペース，！および＊」だけを抜き出して使用している．美咲フォントを使用したのは，最下行および最右行が空白になっている実質的な7×7フォントであり，文字が連続しても見やすいためである．図3.1の(c)にフォントのビット列の例（Wの場合），図3.1(d)に16×16ドットLEDへの表示例を示した．

(c)のようなビット列が(b)の_FONT_romに書き込まれているので，これを読み出してVRAMの先頭から第1文字（この例ではW），第2文字（この例ではe）として各々8バイト分VRAMに書き込む．第3文字以降はVRAM＋0x20番地以降に書き込む．また(d)における下半分の行は，VRAM＋0x10番地以降に書き込むように制御している．

(9) LEDのダイナミック点灯

ダイナミック点灯は，図3.3に示すサブルーチンDynamic_ON_OFFにおいて実行している．このサブルーチンの中の「シフトレジスタ出力」においてLEDの各行に対応するVRAMのビットを1ビットずつ順番に右端から読み出し，シフトレジスタにクロックを送って16ビット分のデータを格納してX軸データを作成している．このデータの作成は5個のシフトレジスタ毎に行っている．クロックの幅は0.16μsである．

全16ビットがシフトレジスタに格納されると，シフトレジスタのLED出力（OE）をHにしてLEDを点灯させる．図2.5に示すように，LEDの点灯時間は1.3msである．

なお，行を選択するパルスも，X軸データと同じような処理を行いY軸用のシフトレジスタに循環させて作成する．

4．シミュレータデバッガHEWに関するノウハウ

今回初めてHEW4(High-performance Embedded Workshop)を使ってH8マイコンのプログラムをデバッグし開発した．しかし初心者であるが故にその使用方法において少なからず苦労を味わった．以下に有用と思われるノウハウを整理しておく．

(1) モード7を使用した場合のアドレスの指定方法

H8/3052Fは内蔵ROM512KB，内蔵RAM8KBであり，このROM，RAMの使い方により7つのモードがある．ここでは内蔵ROM，内蔵RAMを使用して手軽に開発ができるためモード7を使用した．モード7のアドレス空間は最大1MBであり，アドレスは20ビットで表される．

H8マイコンの教科書の例題には，通常24ビットアドレスで書かれており，HEWのデフォルトの設定のままで実行すると「メモリアクセスエラー」が発生して正しくシミュレーションできない．これはシミュレータデバッガに20ビットアドレス

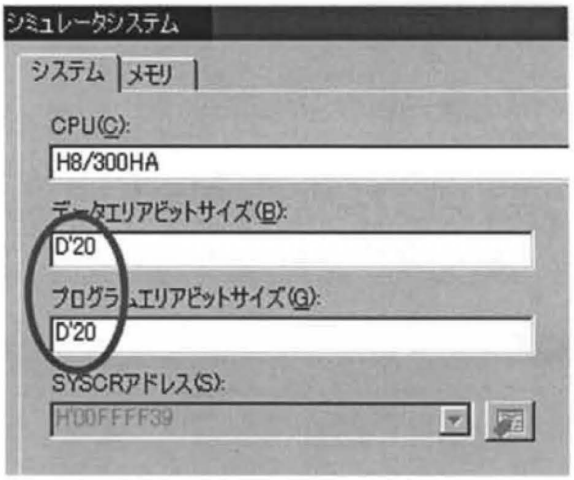

図4.1 20ビットアドレスの指定方法

を正しく設定していないためであるが，この20ビットアドレスを指定する方法に到達するまでかなり時間を要した．図4.1に示すようにシミュレータシステムを正しく20ビットアドレスに指定する必要がある．

(2) モニタプログラム使用時における擬似割り込みの発生

今回開発したプログラムは割り込みを使用しているが，最初は割り込みを発生させる方法が理解できなかった．図4.2はデバッガの 表示／グラフィック／GUI I/O により作成する擬似割り込みを発生させるボタンである．このボタンに割り込みの条件を割り付けることにより擬似的に割り込みを生成させることができる．

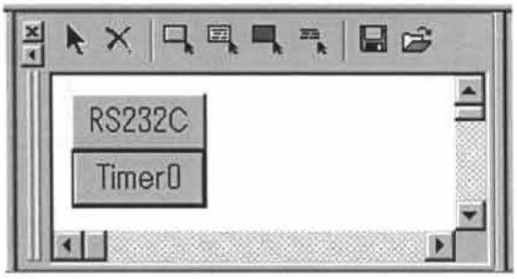

図4.2 擬似割り込みの発生

(3) RAM動作時における割り込みの発生

H8マイコンのROMへの書き込み回数制限は公称100回程度と少ないらしい．そこでデバッグ時にはプログラムをRAMにロードして動作させるようにし，極力ROMへの書き込み回数を減少させるようにした．

図4.2のボタンを押すことによりデッバッガ上では擬似的に割り込みを発生させ，デバッグを行うことができる．しかしROMで動作するプログラムを，モニタプログラムを用いてRAMにロードしても割り込み動作は実行されない．

これはユーザが生成したプログラムをRAMにロードすると，ユーザの割り込みベクタは，例えばアドレスのH'FFF30から配置される．しかしH8マイコンマイコンが割り込みベクタとして認識するのは0番地からの割り込みベクタであるため，H'FFF30以降を割り込みベクタとして認識しないためである．したがって本来の割り込みベクタから，ユーザの割り込みベクタにジャンプさせる中継機能が必要になる．本検討では図4.3に示す関数IntPRG_mon.prgに，H'FFF30に書かれているユーザの割り込みベクタにジャンプさせるためのジャンプ命令を記述し，この関数をプロジェクトに追加して実現した．

```
        .section    IntPRG_mon, code
        .export     _PowerON_Reset_mon
        .export     _INT_RXI0_mon
        .export     _INT_IMIA0_mon
        .import     _PowerON_Reset
        .import     _INT_RXI0
        .import     _INT_IMIA0
        .org        H'00000
_PowerON_Reset_mon: JMP     @_PowerON_Reset
        .org        H'00060
_INT_IMIA0_mon:     JMP     @_INT_IMIA0
        .org        H'000D4
_INT_RXI0_mon:      JMP     @_INT_RXI0
        .END
```

図4.3 仮想割り込みベクタの記述（IntPRG_mon.prg）

(4) タイマ割り込みとSCI0割り込みの混在

各々の処理を関係するISRの中に全て記述することにより，割り込み優先レベルの違いにより期待通りに動作したものと思われる．

5． むすび

ここ数年高校生に対する電気電子への興味を喚起する目的で“動く”，“光る”，“音が出る”などの要素を含む回路を試作してきた．その一つにH21年度に試作した赤／緑色LEDを使用するWDM光通信システムがある．この緑色通信系におけるデモ効果を高める目的で，H8/3052Fをコントローラに適用し，16×16ドットの緑色LEDマトリクスディスプレイを用いるキャラクタディスプレイの設計法および製作結果を示した．大学祭における研究室開放において展示し，LEDの輝度が幾分低いものの，ディスプレイが設計通りに動作していることを確認した．

なお，H8マイコンを使用したのは今回が初めてであり，初心者として難儀した点で有用と思われるものをノウハウとしてまとめて示した．

[参考文献]

1) 袴田吉朗，“展示用波長多重光通信システムの設計と構築”，静岡理工科大学紀要，Vol.18，pp.21-30，(2010)
2) 袴田吉朗，“PICマイコンと16×16LEDを用いた漢字表示電光掲示板の設計と試作”，静岡理工科大学紀要，Vol.17，pp.133-142，(2009)
3) HEWデバッガユーザーズマニュアル，日立製作所，(2002)
4) H8S、H8/300シリーズ シミュレータ／デバッガ，ルネサスマイクロコンピュータ開発環境システム，(2007)
5) H8/3052F-ZTATTM ハードウェアマニュアル，日立製作所，(2002)
6) TekuRobo工作室

[付録] 最終的なプログラムの所在

本資料は約2ヶ月半にわたり検討を進めてきて，2011.9.8にデバグを完了したプロジェクトled_disp_3052F_monに基づいて作成した．しかしながら資料作成を進めるに当たりプログラムに若干の見直しを行う必要が生じた．このため新たにプロジェクトled_disp_3052F_mon_2012_2_20を作成し，このプロジェクトに修正点を反映させた．