教育用仮想マイクロコンピュータの開発－LEDによる動作の可視化

Development of a Virtual Micro Computer for Education Use – Visualization of an Operation with LEDs

袴田　吉朗*　　村田　英之**

Yoshiro　HAKAMATA　　**Hideyuki　MURATA**

Abstract: The material summarizes the design and manufacture of a Micro Computer System for education use. It consists of LEDs, CMOS SSIs, a CPLD for Parallel IO and a PIC micro computer. The system visualizes an operation of a Virtual Micro Computer by making LEDs On and OFF. The system hardware configuration and program configuration are precisely taken into account in the material.

1．　はじめに

2000年度の卒業研究においてZ80，8255A PIO（パラレルIO）およびCMOS　SSIを用いてマイクロコンピュータの動作を可視化する装置を試作した[1]．試作した目的は，オープンキャンパスにおいて使用し高校生に電気・電子に興味を持って貰うことであったが，オープンキャンパスでは説明時間が十分に取れず，この装置は難しい，分からない，などと高校生にはあまり評判が良くないという結果に終わってしまった．

来年度から袴田が「マイクロコンピュータ応用」の科目を再度担当することになった．これを機会に，同装置を授業において学生に見せ，マイクロコンピュータの動作の理解を深めて貰うことを目指して同装置を棚から取り出して久しぶりに動作させてみた．残念ながら正常に動作しなかった．そこで同装置を正常に動作させるために，以下の調査を順番に進めていった．

① CPUの代わりにトグルスイッチを用いて，表示パネルにおけるLEDの点灯具合を調べた．

② 表示パネルのラッチ回路にアドレスを送出する3－8デコーダ回路の動作を調べた．

これらの回路については動作に問題がなかった．

③ 次に，Z80で作成されていたプログラムにおいてLEDの点滅制御を行っている部分のみをPIC16F648Aに部分的に移植し動作させてみた．この処置では装置は正常に動作しなかった．

④ 8255A PIOについても，以前に買い求めていた残り少ない在庫品と取り換えてみた．オシロスコープを用いて入出力の波形をつぶさに観測し調査してみたが，ポートPAは正しく動作しているものの，ポートPBおよびポートPCは正しく動作していないことが判明した．

⑤ そのためゲートアレイCYC374i-100JCを用いて8255Aと同等の機能を有するPIOを試作することにした．これを適用したところ，ハードウェアの問題を解決することができた．

⑥ そこでプログラム全体をPIC16F648Aに移植してみたが，その動作はプログラムのシミュレーション結果とは全く異なる異質のものであった．冷静に調査を進めたところ，PIOの制御プログラムに考え間違いがあること，3－8デコーダの配線にも齟齬のあることが分かった．これらの不具合を修正することにより装置を正常に動作させることができた．

なお新たに，表示できるプログラムの数を従来の7から15まで増加させること，動作の渦中にあるLEDを5回フラッシュさせることによりプログラムの流れを分かりやすくすること，などの処理を追加した．

2011年3月2日受理

* 理工学部　電気電子工学科

**理工学部　電気電子情報工学科4年生

2．ハードウェア構成

2.1　システムのブロック構成および表示パネル

図2.1にシステムのブロック構成を示す．全体で5つのブロックからなる．CPUはPIC16F648Aであり，内蔵の4MHzクロックで動作させている．PIOは8255Aのモード0との同等品をCPLDにより試作した．3－8デコーダ回路，表示パネルおよびスイッチ回路は2000年度に試作したものに一部修正を加えて使用した．図2.2に表示パネルの図を示す．背面のプリント基板上に32個のトランスペアレントラッチ74HC573，250個のLEDおよび同数の電流制限抵抗（1kΩ）を配置している．

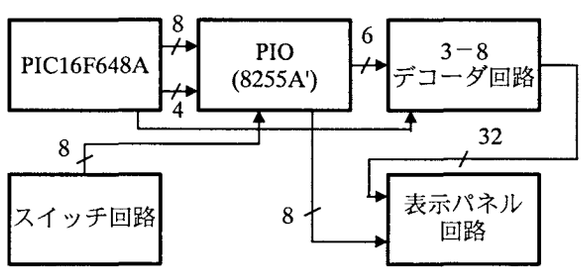

図2.1　システムのブロック構成

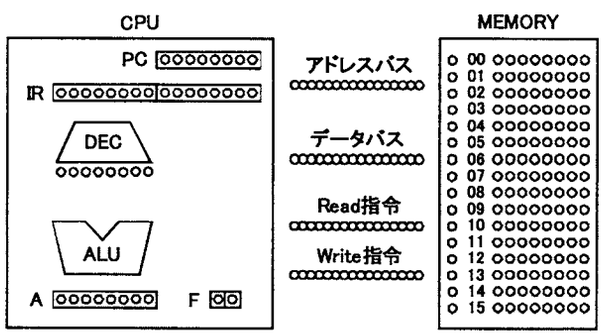

図2.2　表示パネルの図

2.2 3−8 デコーダ回路および表示パネル回路

図2.3 に表示パネル回路における1ビット分の回路図を示す．スタティック点灯のため電流制限抵抗の値を 1kΩと大きくしている．LED を点灯させるデータは，PIO のポート PA から導かれる 8 ビットのデータを 2 分岐し，非反転バッファ 74HC250 により駆動されるバスを介して 16x2 個のラッチ 74HC573 の D 端子に（ほぼ同一タイミングで）伝送される．

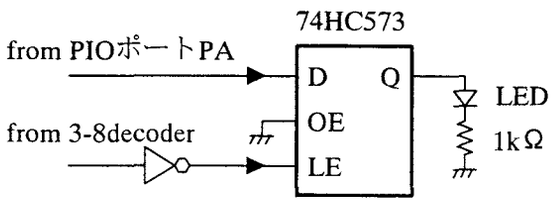

図2.3 表示パネル回路（1ビット分）

トランスペアレントラッチ 74HC573 の真理値表を表 2.1 に示す．これより OUTPOUT ENABLE（OE）端子を L に固定し，一方 LATCH ENABLE（LE）端子には後述する 3−8 デコーダ（アクティブロー）出力をインバータで反転して加えている．表 2.2 に示すように 3−8 デコーダの出力は，アドレスが選択されると L になる．この間インバータ出力すなわち 74HC573 の LE は H となり，74HC573 はトランスペアレントになる．一方 LE=L になると直前のデータが保持（ラッチ）される．

なお選択されていないアドレスに対応する 74HC573 の出力は，3−8 デコーダの出力=H，インバータ出力（LE）=L であり前のデータを保持し続ける（no change）ことになる．

表 2.1 トランスペアレントラッチ 74HC573 の真理値表

入力			出力
OUTPUT ENABLE	LATCH ENABLE	D	Q
L	H	H	H
L	H	L	L
L	L	×	no change
H	×	×	Z

2.3 3−8 デコーダ回路

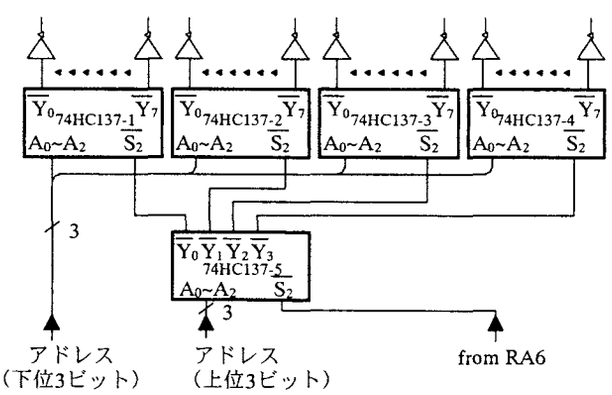

図2.4 3−8回路のブロック図

図 2.4 に 3−8 デコーダ回路のブロック図を示す．PIO のポート PB から伝送されるアドレス信号は，5 個の 3−8 デコーダ 74HC137 からなる 3−8 デコーダ回路においてデコードされ，32 本のアドレス線のうち特定のアドレス線 1 本だけを L とし，PIO のポート PA から出力されるデータを特定の 74HC573 にラッチする働きをする．アドレス信号の下位 3 ビットを 3−8 デコーダ 1〜4 に直接導き，1 個の 3−8 デコーダによって 8 個の 74HC573 を制御する．アドレス信号の上位 3 ビットにより，アクティブとなる 3−8 デコーダ 1〜4 を選択しているが，最上位のビット 2 は 0 に固定してある．また図に明示していない制御信号は固定的に /LE=L，S1=H である．

表 2.2 74HC137 の真理値表

/LE	S1	/S2	/Y0〜/Y7
don't care	don't care	H	H
L	H	L	A0〜A2 に対応する出力が L になる

表 2.2 は 3−8 デコーダ 74HC137 の真理値表である．2000 年度に試作した装置では/LE=L，/S2=L に固定されていたが，このままでは PIC により良好にラッチ 74HC573 を制御することができなかった．このため PIC の空きポートである RA6 を 3−8 デコーダ回路の/S2 端子に接続し，図 2.5 に示すようにアドレス信号が PIO のポート PB にラッチされてから，RA6 を H→L→H と変化させて 74HC573 にアドレスを入力するようにした．これにより良好な動作を実現できるようになった．

なおアドレスされていない 74HC573 の出力は，表 2.2 に示すように 3−8 デコーダの出力が H，インバータ出力（LE）が L

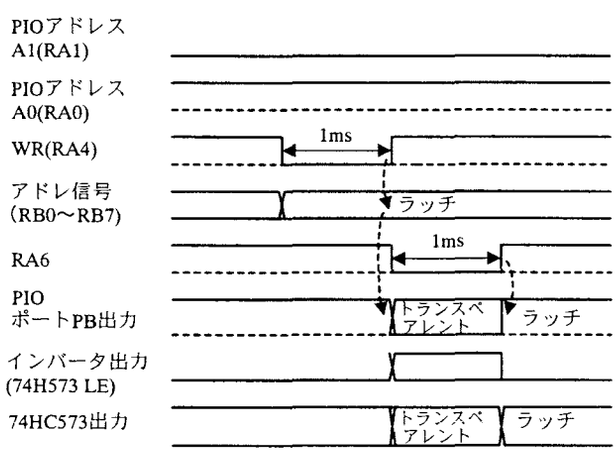

図2.5 74HC137におけるタイミングチャート（アドレスの出力）

表 2.3 アドレス信号と 3−8 デコーダの動作

アドレス信号（上位 3 ビット）	アクティブになる 3−8 デコーダ
000	74HC137−1
001	74HC137−2
010	74HC137−3
011	74HC137−4
1xx	なし（全 74HC137 出力が H）

xx：don't care

であり前述のように前のデータを保持し続ける.

アドレス信号と3−8デコーダの動作の整理を表2.3に示す.

2.4 PIOの試作

図2.6に試作したPIOのブロック構成を示す. PIOはアドレス選択回路, ラッチおよびバッファからなる. 汎用品である8255Aはコントロールワード CWR を用いて各ポートの機能を必要に応じて適切に設定できるようになっている. しかし今回は単一機能を実現すれば良いので, ポートPAおよびポートPBは出力, ポートPCは入力専用とし, 8255Aのモード0に相当する機能を設計しCPLD CYC374i-100JC を用いて試作した.

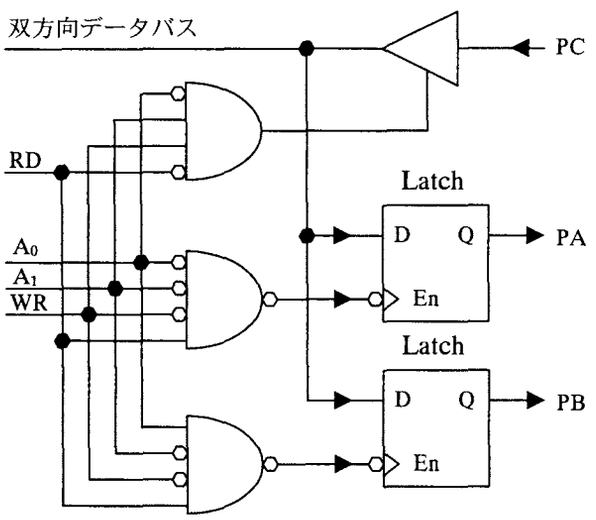

図2.6 試作PIOのブロック図（1ビット分）

クッロク周波数10MHzにおいてPIOのVHDLによるシミュレーションを行った. その結果を図2.7に示す. 動作は以下の通りである.

① PICからポートPA（あるいはPB）へのデータ出力

ポートアドレスA1, A0がセットされ, かつRD=Hであり WR=Lのときポート PA（あるいはポート PB）はトランスペアレントになる. 次にWR=Hになると直前のデータがラッチされる. 図2.7では, ポートPAには一例として0x01が, ポートPBには0x02が出力されラッチされている. このときにはEnC=Lであり, 3ステートバッファはハイインピーダンスになっている.

② ポートPCからPICへのデータ入力

ポートアドレスがA1=H, A0=Lにセットされ, かつWR=H, RD=Lのとき EnC=H となる. このとき3ステートバッファはアクティブとなり, ポートPCの値が双方向バッファDに出力される筈である. しかし使用したCypress社の廉価版シミュレータでは, DにPCの値（この場合には0x03）が正しく出力されず, うまくシミュレーションできなかった.

図2.7のD1は, Latch−A, Latch−Bとは無接続の双方向バス（inout指定）である. RD=Lになると PCの値が正しくバスD1に出力されるので, タイミングとしては特に問題がなく, 実機においても図2.5の設計で正しく動作することを確認した.

2.5 スイッチ回路

図2.8はスイッチ回路の回路図である. プルアップ抵抗（5.6kΩ), トグルスイッチおよび押しボタンスイッチから構成されている. このトグルスイッチの役目はプログラムの入力である. 押しボタンスイッチの役目はプログラムの進行である.

PIOにおけるポートCのPC0〜PC6に接続したトグルスイッチによってMEMORY部に表示するプログラムをPICに読み込み, かつLEDに表示する. PC7に接続してある押しボタンスイッチを押すとマニュアル動作となり, 表示パネルに表示される処理を1ステップずつ進めることができる. またPC7と並列にトグルスイッチを接続してあり, このスイッチをON（論理のL）にするとオート動作になり自動的に処理が進む. このようにPICのプログラムを作成している.

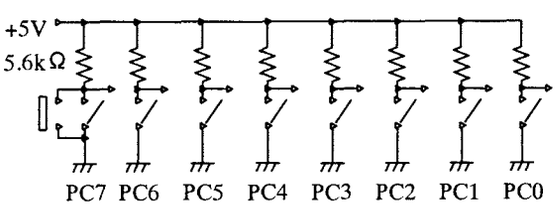

図2.8 スイッチ回路

2.6 PICマイコン

PIC16F648Aを内蔵の4MHzクロックを用いて駆動している. 表2.4にポートの割り当てを示す. ポートAを各種の制御信号に割り当て, 一方ポートBを双方向データバスに接続してPIOへの書き込みあるいはPIOからの読み出しを行っている.

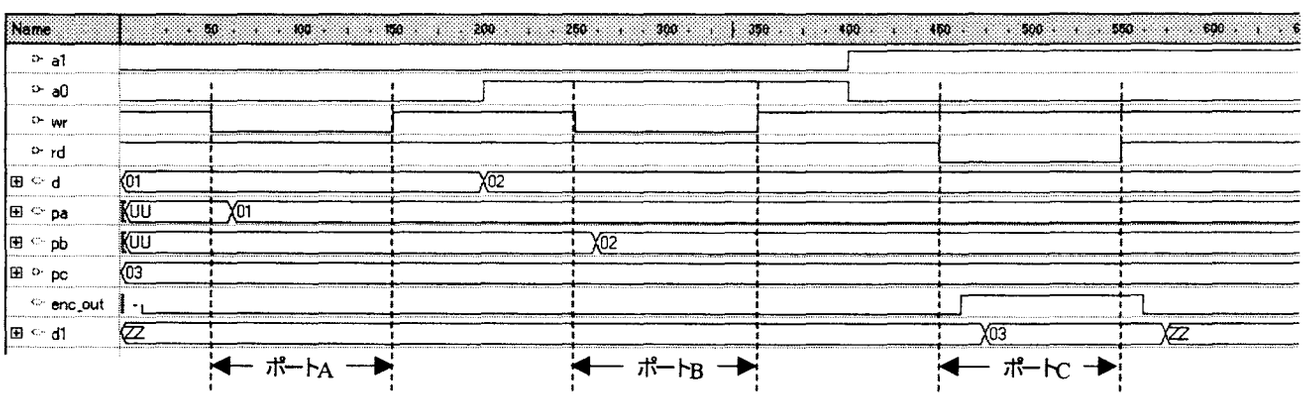

図2.7 試作したPIOのタイミングチャート

表2.4 PIC16F648Aにおけるポートの割り当て

PICのポート	信号の割り当て
RA0	PIOのアドレスA0
RA1	PIOのアドレスA1
RA2	
RA3	PIOのREAD
RA4	PIOのWRITE
RA5	
RA6	3-8デコーダにおける出力パルス後縁の制御
RA7	
RB0～RB7	データ（双方向）

3. 処理プログラムの構成

3.1 仮想マイクロコンピュータの仕様

仮想マイクロコンピュータのブロック構成を図3.1に示す. 1バス構成である. 仕様を表3.1に示す. 命令の語長はオペコード1バイト, オペランド1バイトの2バイト構成である. 1命令は, 4命令サイクルからなる. 命令の種類を表3.2に示す. ANDの演算命令は, 現時点では未実装である.

仮想マイクロコンピュータのプログラムを, PIC16F648Aの0x0F番地～0xEF番地にRETLW命令を用いて格納できる.

表3.1 仮想マイクロコンピュータの仕様

メモリ容量	16バイト
命令の語長	1バイト
命令サイクル	オペコードフェッチ, オペランドフェッチ 命令解読, 命令実行
レジスタ	プログラムカウンタ（@PC)
	オペコードレジスタ（@IROC)
	オペランドレジスタ（@IROL)
	アキュムレータ（@AREG)
	フラグレジスタ（@FREG)
	上位メモリ番地の表示（@MSELU)
	下位メモリ番地の表示（@MSELD)
命令の種類	表3.2に示す

表3.2 仮想マイクロコンピュータにおける命令の種類

ニーモニック	オペコード	意味
LD A, n	0x80	Aレジスタ←リテラルn
LD (n), A	0x81	メモリのn番地←Aレジスタ
LD A, (n),	0x82	Aレジスタ←メモリのn番地
ADD A, n	0x20	A ← A + リテラルn
XOR A, n	0x21	A ← A xor リテラルn
AND A, n	0x22	A ← A and リテラルn
NOT A	0x23	A ← not A
SUB A, n	0x24	A ← A − リテラルn
JP n	0x10	メモリのn番地にジャンプ
JP Z, n	0x11	演算結果がゼロであればメモリのn番地にジャンプ

これによりADDWF命令を用いてデータを読み出す場合のpclathの制御を省略している. このプログラムをPICの0xF0～0xFF番地（先頭アドレスが@MEM00）に設定した16バイトのワークエリアに読み出し, この内容に従ってPIC16F648Aにおいて実際に演算を実施し, その結果をLEDにおいて表示するようにしている. したがって格納プログラム数は最大15である.

3.2 仮想マイクロコンピュータを実現するためのPICプログラムの概略フローチャート

図3.3にPICプログラムにおける概略フローチャートを示す. 初期設定では以下に示すPICの設定を行う.

① 内部クロック（4MHz）の設定

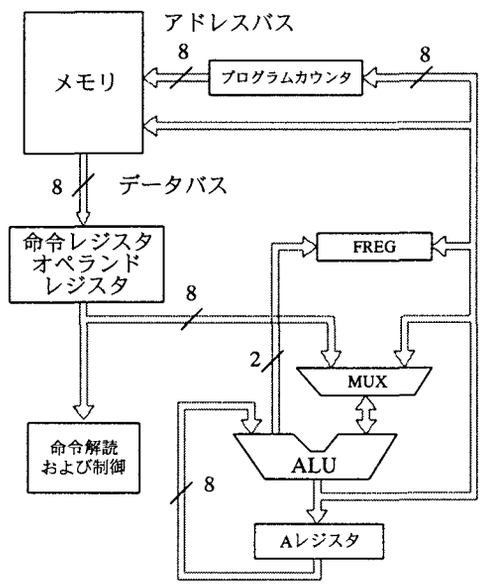

図3.1 仮想マイクロコンピュータのブロック構成

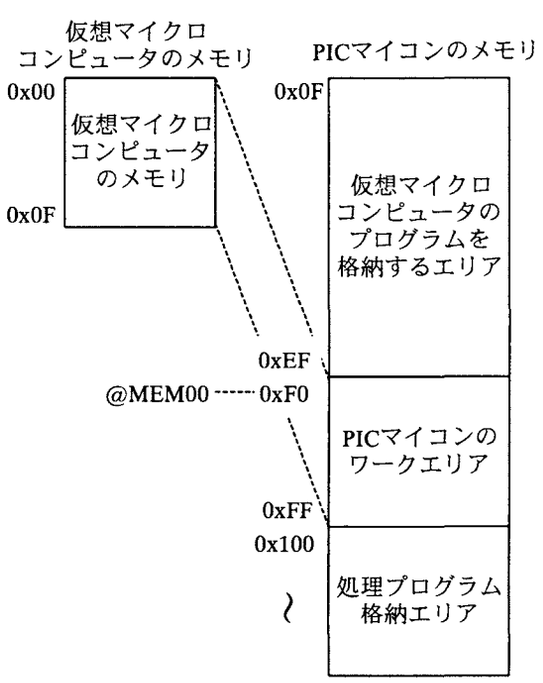

図3.2 メモリの割り当て

② ポートAをディジタルポートに設定（CMCONレジスタ）
③ TRISレジスタの設定
④ ポートBをウィークプルアップに設定
なお割り込み処理は使用していない.

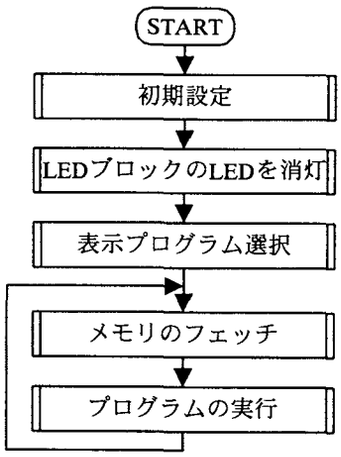

図3.3 概略フローチャート

3.3 LEDの点灯／消灯の処理

図3.4にLEDの点灯／消灯を制御するフローチャートを示す. これにより図2.5および図2.7に示したタイミングを実現している. どのLEDを点灯し，消灯するかのデータ（LEDデータ）は8ビットのデータバスを介して全トランスペアレントラッチ74HC573に伝送される. このLEDデータを変数DTAに代入し，またどのラッチを動作させるかのアドレス信号を変数AR1に代入後，PIOを制御するサブルーチンOUT_to_PAおよびOUT_to_PBをコールしている.

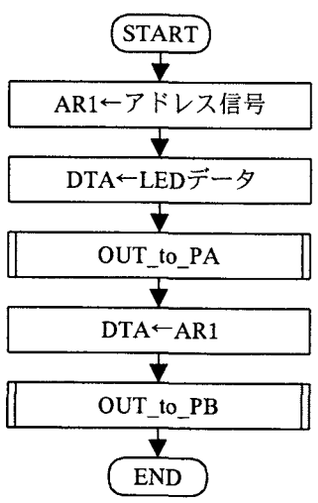

図3.4 LED点灯のフローチャート

図3.5および図3.6にサブルーチンOUT_to_PAおよびOUT_to_PBのフローチャートを示す. PIOのポートアドレスA1，A0を設定後，WRITE制御信号RA4をH→Lにした後でデータ（LEDの点灯／消灯のデータ，アドレス信号）をPORTBに出力し，最後にRA4=Hとするのは両サブルーチンにおいて共通である.

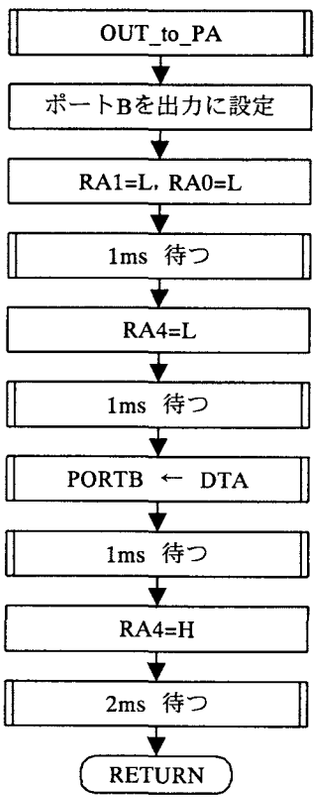

図3.5 PIOのポートPAへの出力

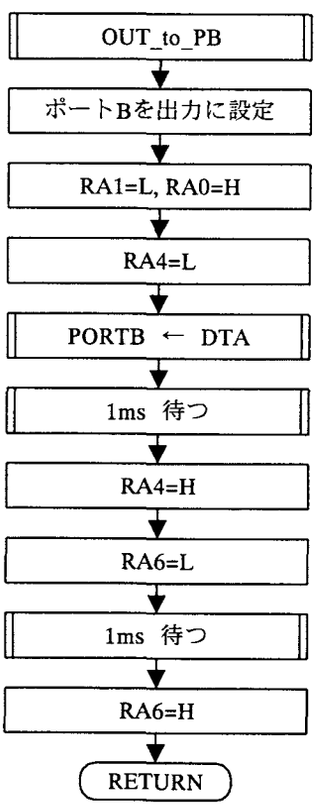

図3.6 PIOのポートPBへの出力

OUT_to_PBには，切り分けは良くないが，3−8デコーダ回路のラッチタイミングを制御するRA6の制御（H→L→H）を

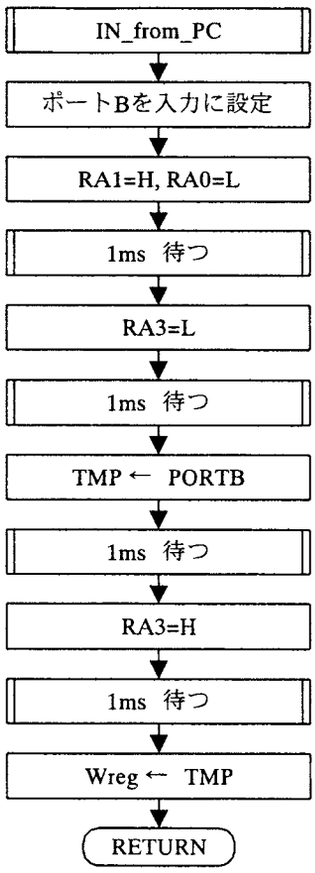

図3.7 PIOのポートPCからの入力

組み込んでいる.

図3.7はPIOを介してスイッチのON／OFFの状態を入力するためのフローチャートである. PIOのアドレスを設定後, READ制御信号RA3をLに下げ, その後PORTBからデータを一時変数TMPに取り込む. 次に1ms待ってからRA3をHに戻す. 最終的なデータはWregを介して受け渡しするが, 1ms待つサブルーチンにおいてWregを使用しているためデータを一時変数TMPに待避している.

3.4 メモリフェッチ

図3.8にメモリフェッチのフローチャートを示す. プログラムを格納しているワークエリアの先頭番地@MEM00からプログラムカウンタ@PCをディスプレースメントとする内容を順番に読み出し, 処理を実行させる.

@PCが偶数時には読み出した内容をオペコードレジスタ@IROCに格納し, 奇数時にはオペランドレジスタ@IROLに格納する. @IROCがHALT(0xFF)の場合にはプログラムをSLEEPさせている.

サブルーチンMEMSELにより, アドレスされているメモリを示すLEDを点灯する. サブルーチンAORMは, オート処理あるいはマニュアル処理の切り替え用である. AORMの前でLEDを5回点滅させるのは, 処理が現在どのポイントにあるかを明確に示し, 理解を深めて貰うことを意図している.

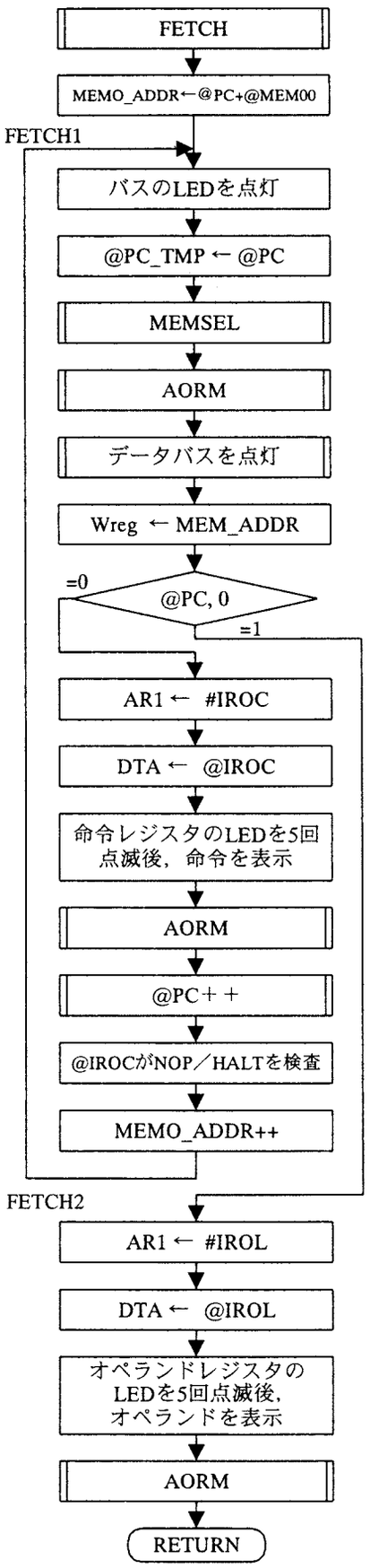

図3.8 メモリフェッチ

3.5 命令の解読と実行

命令レジスタ@IROCの内容をDECと表示してある部分のLED（アドレスは定数＃DECOUT）に表示後，命令振り分けを行い，実際にPICにおいてその命令を実行しその結果をLEDの表示に反映させる．

図3.9には一例として ADD A, n の場合のフローチャートを示した．本仮想マイクロコンピュータはIOを持っていないので，計算結果はAレジスタか，あるいはメモリへ出力される．図3.9はAレジスタに出力される場合の例である．

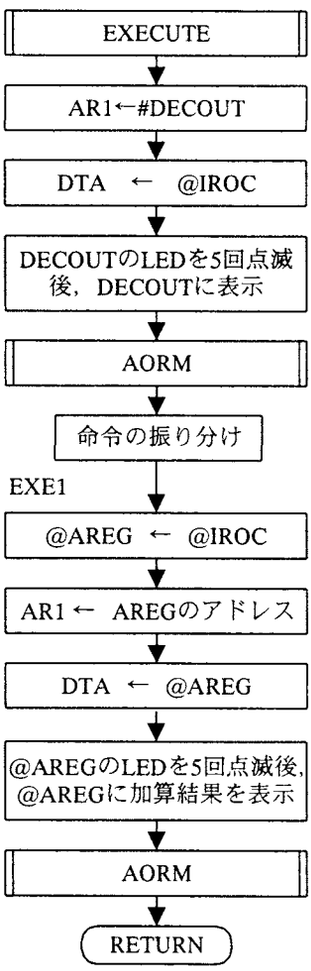

図3.9 命令解読，実行

3.6 スイッチからの入力

図3.10に処理のオート／マニュアル切り替え用を制御するサブルーチンであるAORMのフローチャートを示す．PC7に接続されたトグルスイッチをONにするとPC7=Lとなる．その後図3.10のサブルーチンSWINBにおいてスイッチのチャタリング回避のためのウェイト処理を経て，論理が反転された後スイッチの状態が変数@swにコピーされ，そのビット7が1となる．これより図3.10において処理が800ms毎に自動的に進行する．

一方，PC7に接続されたトグルスイッチをOFFにしておくと処理は図3.10におけるMAN_MODEに移る．すなわち図3.11のサブルーチンSWINAを実行する．SWINAの冒頭にはスイッチが押されたか／否かを判断する部分を有するので，スイッチ

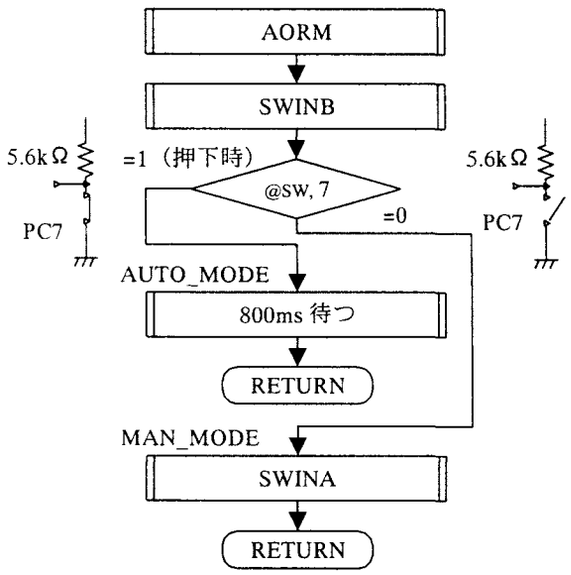

図3.10 オート／マニュアルの設定

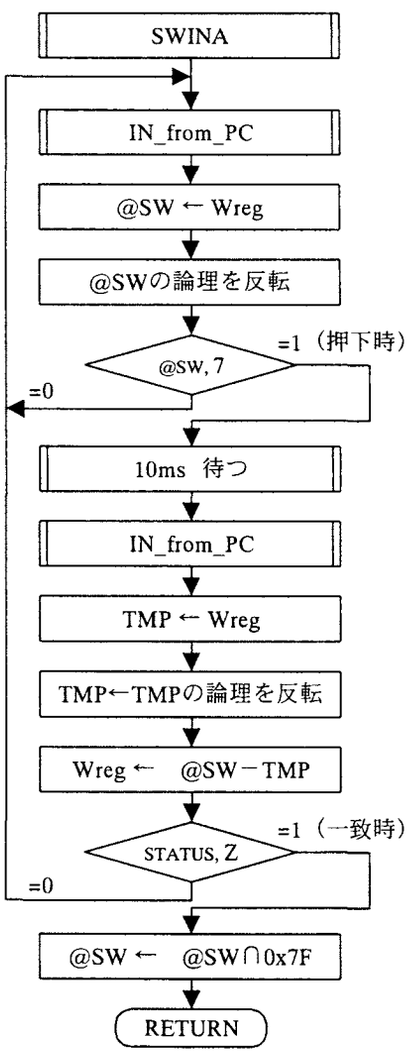

図3.11 SWINA

が押されて（その後論理が反転されて）@SW=1となるまで処理をループさせる．スイッチが押されて@SW=1になると，チャタリングがなければ次のステップに進むことになる．

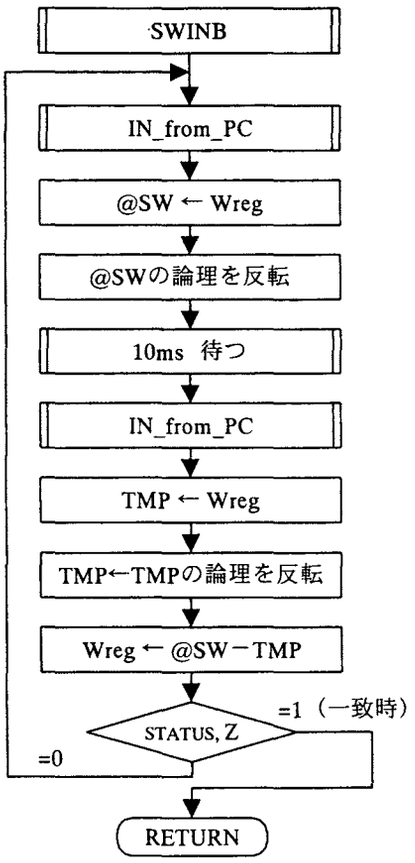

図3.12 SWINB

3.7 LEDをサイクリックに点滅させる処理

デモを行うときに，アドレスバスやデータバス，メモリ番地を示すLEDブロックのうちどのブロックを介してデータが伝送されているかを明確に示すために，渦中にあるLEDブロックを1ビットだけ点灯させこれをサイクリックに動かしながら2周期だけ繰り返すようにプログラムを作成した．

アドレスバスやデータバス，メモリ番地を示すLEDブロックなどは16ビット構成であり，キャリーレジスタを介した論理的な16ビットのレジスタを構成して処理を行っている．右シフトの場合のフローチャートを図3.13に示す．処理の概要は以下に示すとおりである．

① 8ビットのリングバッファRING_UおよびRING_Dをメモリに確保する．

② 両レジスタに0xFFを代入し，1ビットずつ右にシフトする．このときシフトの順番はRING_U → RING_Dとする．なお最初にRING_Uのシフトを行う前にキャリーフラグレジスタCをクリアしておく必要がある．

③ バッファRING_Uから溢れたデータはキャリーフラグレジスタCに入る．

④ キャリーフラグレジスタCをクリアせずにRING_Dのシフトを行えば，キャリーフラグレジスタCの内容がRING_Dの最上位ビットに入る．

⑤ 結局物理的には1個しかないキャリーフラグレジスタCを介して，論理的には図3.14に示すように16ビットのリン

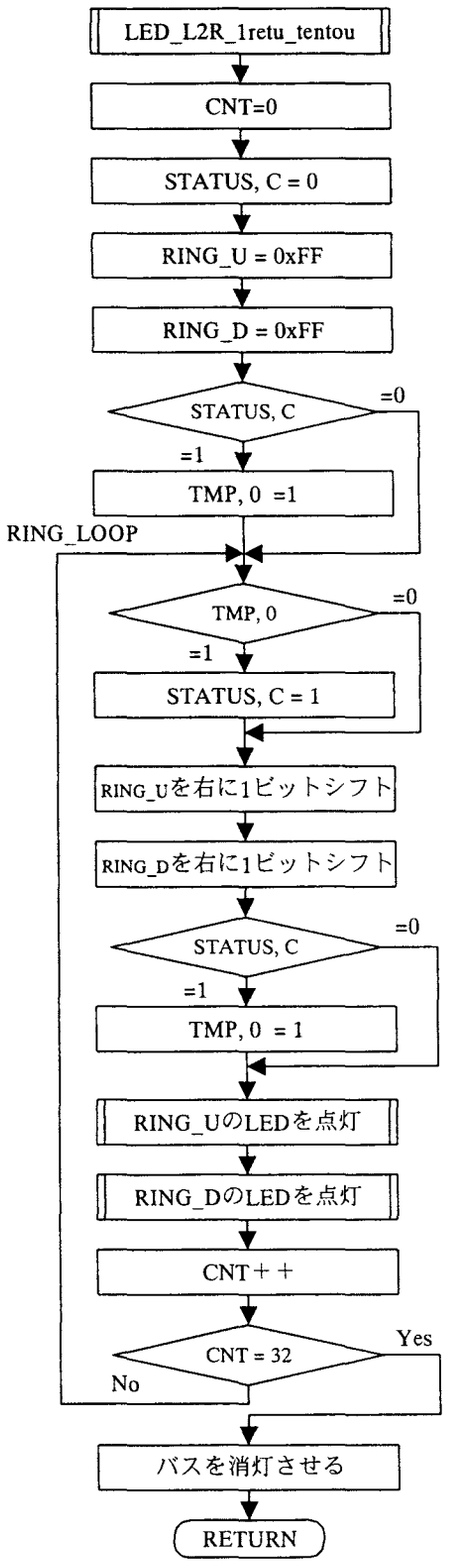

図3.13 バスの点灯フロー

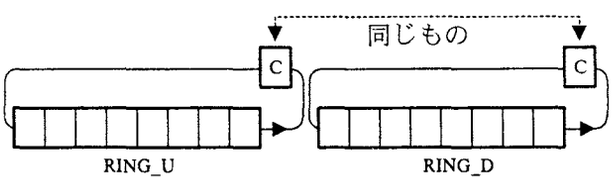

図3.14 16ビットリングバッファの構成

グバッファとして動作させることができる.

⑥ 左シフトの場合にはシフトの順番を RING_D→RING_U とする必要があるが, それ以外は全く右シフトの場合と同様に処理すれば良い.

3.8 フラグレジスタの処理

図3.15にフローチャートを示す. PICマイコンで行った演算結果がSTATUSレジスタのフラグに反映される. この結果のうちZフラグおよびCフラグの内容を取り出し, @FREGを点灯させる. @FREGは2ビットのレジスタであり, 演算結果がゼロであることを示すZフラグをビット1に, キャリーが1であることを示すCフラグをビット0に割り当てている.

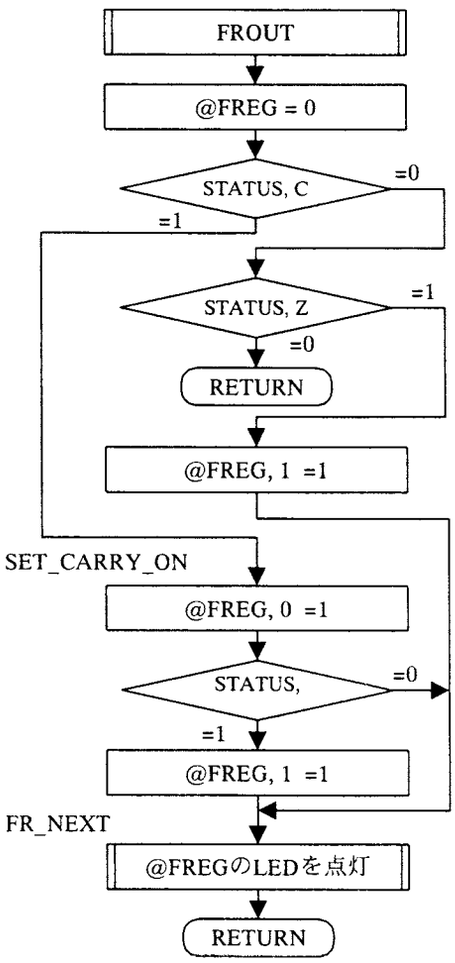

図3.15 フラグレジスタの制御

3.9 ジャンプ命令に対する処理

(1) JP n

オペランド@IROLをプログラムカウンタ@PCに転送後, LEDの点灯を行うことにより単純に実現できた.

(2) JP Z, n

仮想マイクロコンピュータにおけるフラグレジスタ@FREGのビット1 (Zフラグ) を検査して, これが1であれば無条件ジャンプ命令と同じ動作を行う.

Zフラグが0の場合には, CALL ADDPCを実行し単にプログラムカウンタをインクレメントする.

3.10 3−8デコーダ回路の別の制御方法

図2.5に示したタイミングチャートでは, 3−8デコーダ回路の制御をPIOのポートPBだけでなく, PICマイコンから導かれる制御信号RA6を併用して行っており, 処理の切り分けが良くない. そこでより明快となる制御を検討した.

図2.5に示した制御の一連の流れは以下のようになっている.

① PIOのポートPBから74HC137−5にアドレス信号が送られ, /S2をLに設定する. これにより特定の74HC573に対してアドレス信号が送出される.

② 74HC573はアドレス信号を受信するとトランスペアレントになる.

③ その後, RA6を制御して3−8デコーダの/S2をHに設定する. この結果その3−8デコーダの出力がHにセットされ, 結果的に全3−8デコーダがインアクティブになる.

④ それ以後74HC573は, no changeの状態になる.

ポイントは下線を引いた部分である. しかしRA6を使用しなくても, 表2.3の最下行に示したようにアドレス信号の最上位ビットを1にすれば全74HC137出力をHにできるので, RA6を使用した場合と同じように制御できるようになる. ただし固定的に/S2=Lにしておく必要がある.

制御のタイミングチャートを図3.16に示す. PIOにおいてWR=LにするとPIOはトランスペアレントになる. この間にアドレス信号の上位3ビットを 1xx → 0xx → 1xx に変化させると3−8デコーダ74HC137−5の出力も同じように変化し, 74HC137−1〜74HC137−4のいずれかを選択しいている状態からどの3−8デコーダも選択していない状態になる. その結果, 全ての3−8デコーダの出力がHになる. これがインバータを介して74HC573に入力されるので, 74HC573はトランスペアレント状態からラッチされた状態 (no change) に移行し目的を達成できる筈であるが, まだデバッグは終了していない.

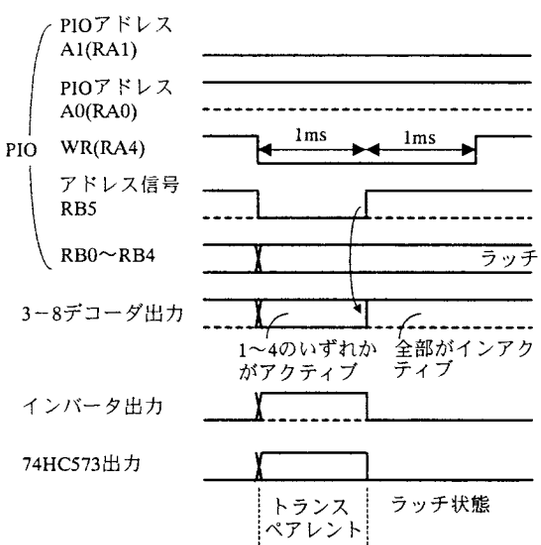

図3.16 3-8デコーダ回路の別の制御方法

3.11 実装したプログラムのリスト

```
;PGM0（加算1）

LD      A,1
ADD     A,1
LD      (15), A
HALT

;PGM1（加算2）

LD      A, 10
ADD     A,  8
LD      (14), A
HALT

;PGM2（加算3）

LD      A,  4
ADD     A, 12
LD      (13), A
HALT

;PGM3（無条件ジャンプ）

LD      A, 11111110B
ADD     A, 1
JP      2
HALT

;PGM4（無条件ジャンプ）

LD      A, 11111100B
ADD     A, 00000011B
LD      (14), A
JP      2
HALT

;PGM5（メモリへの格納）

LD      A,8
ADD     A,8
LD      (14), A
LD      A, 1
ADD     A, 1
LD      (15), A
HALT

;PGM6（条件付ジャンプ）

LD      A, 0
ADD     A, 01000000B
JP      Z, 8
JP      2
LD      A, 01010101B
XOR     A, 01010101B
JP      Z, 0
HALT

;PGM7（減算）

LD      A, 1
NOT     A
ADD     A, 1
ADD     A, 2
SUB     A, 1
LD      A, (15)
HALT
```

図3.17　実装したプログラムのリスト

実装したプログラムのリストを図3.17に示す．プログラムの番号は図2.8におけるスイッチ回路のPC6（MSB）～PC0(LSB)を2進数表示した場合の番号に対応している．内容は概略以下のようになっている．

PGM0～PGM2：加算の例．明示のない数値は10進数である．
PGM3～PGM4：無条件ジャンプ命令の使用例
PGM5　：加算結果をメモリに格納する例
PGM6　：条件付きジャンプ命令の使用例．またXOR命令も使用している
PGM7　：2の補数計算による減算の例．メモリが少ないため最初に減数である1の補数を計算し，この値に被減数である2を加えて2－1＝1を計算している．その後でSUB命令を実行しその結果をメモリの15番地に格納している．

なお表2.3の命令の一覧表において，オペコード0x23：未使用→NOT，オペコード0x24：OR→SUBの変更を行った．

4．　むすび

来年度から袴田が「マイクロコンピュータ応用」の科目を再度担当することになった．これを機会に，2000年度の卒業研究において試作した「仮想マイクロコンピュータ」装置を授業において学生に見せ，マイクロコンピュータの動作の理解を深めて貰うことを目指してこの装置を棚から取り出して久しぶりに動作させてみた．残念ながら正常に動作しなかった．そこで同装置を正常に動作させるための検討を進め，回路動作の確認を行い，またCPUをPICマイコンに変更してプログラムを移植することにより装置を完成させた．検討した内容を以下に示す．

(1) 表示パネル回路，3-8デコーダ回路およびスイッチ回路は2000年度に試作したものをほぼそのまま使用した．
(2) CPLD（CYC374i－100JC）を用いて8255Aのモード0相当のPIO（パラレルIO）を試作して適用した．
(3) CPUをPIC16F648Aに変更し処理プログラムを移植した．新たに動作の渦中にあるLEDを5回フラッシュさせて流れを分かりやすくすること，表示できるプログラム数を増大させること，などの処理を追加した．

来年度の「マイクロコンピュータ応用」の授業においてこの装置を実際に使用してみて，不具合があれば更に改良を進めていく積もりである．

なお本検討は，2010年度の卒業研究の一環として行ったものであるが，袴田がPICプログラムの作成およびシミュレーション，PIOのVHDLによる設計およびシミュレーションを担当した．村田はPIC回路の製作，PIOのCPLDへの書き込み，表示パネル回路，3－8デコーダ回路およびスイッチ回路の特性の調査を行った．プログラムのデバッグは，共同で実施した．

［参考文献］

1) 瀧浪博允，“ステートマシンの試作によるマイクロコンピュータの可視化に関する研究”，2000年度静岡理工科大学電子工学科卒業論文
2) PIC16F627A/628A/648A Data Sheet Flash-Based 8-Bit CMOS Micro controllers with nano Watt Technology, Microchip
3) 神崎康宏，“作りながら学ぶPICマイコン入門”，CQ出版社，2007年
4) 後閑哲也，“改訂版　電子工作のためのPIC16F活用ガイドブック”，技術評論社，2007年

［付録］最終的なプログラムの所在

本資料は約4ヶ月間にわたり検討を進めてきて，2010.12.23にデバグを完了したプログラム「Z80_648A_final_20101219.asm」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．この修正点については，ファイル名「Z80_648A_20110206.asm」に示している．またPIOのVHDLファイルは，2010.11.19作成の「PIO_8255A_3.vhd」に基づいている．