計算機の周辺インターフェース

概要

テーマは MPU 周辺インターフェース(周辺LSI)とそれに関連するインターフェース素子(ドライバ/レシーバ/トランシーバ/センスアンプなど)である。

周辺LSI やインターフェース素子は、本文２節に説明するように多岐にわたっている。このため全てのものを取り上げることはできない。従って、本実験では、周辺LSI としてパラレル通信コントローラの一つである PPI(8255)、タイマ/カウンタの 8253、割り込みコントローラの 8259とその周辺LSIに関わるインターフェース素子であるデコーダIC、バスバッファIC を題材に取り上げて、それらの機能と取り扱いについて学び、これらのデバイスを使ってパソコンの拡張スロット用のインターフェースボードを設計・製作する。また、アセンブラ言語や高級言語(BASIC言語、Ｃ言語)を利用して製作したＩ／Ｏボードを駆動させることで計算機周辺インターフェースの一端を学ぶものである。

スケジュール

(1)１回目：テーマ全体を把握し、デコーダICとバスバッファICに関する初歩的な実験を行う。
２回目からの準備を行う。
(2)２日目からは、入出力ボード等の設計・製作を行う。
(3)４回目：アセンブラ、Ｃ言語、ＢＡＳＩＣ言語を用いて製作したＩ／Ｏボードを動作させる。

執筆：長岡敏彦 名古屋大学工学部・工学研究科技術部計測・制御技術系技術長(技術専門官)
協力：関川純哉 名古屋大学大学院工学研究科博士課程後期課程ｴﾈﾙｷﾞｰ理工学専攻満了
工学博士(名古屋大学 1993.3)
小野孝男 名古屋大学大学院工学研究科博士課程博士課程後期課程情報工学専攻満了
同 電子工学専攻助手 工学博士(名古屋大学 1993.3)

(Ver1.0:1994.08, Ver1.1:1995.02, Ver2.0:1996.08, Ver.3.0:1997.08, Ver.3.1:1999.08)

目　次

1．目 的 …… 1

2．インターフェース …… 1
(1)インターフェース用ＩＣ …… 1
(2)周辺ＬＳＩ …… 2
(3)８０８０Ａ／８０８５ＡとＺ-８０のファミリ …… 2

3．パラレルＩ／Ｏポート …… 3

4．インターフェース用ＩＣ …… 3
4．1　アドレス・デコーダ用ＩＣ …… 3
(1) ７４ＬＳ１３８ …… 3
(2) ７４ＬＳ１３９ …… 3
課題１　デコーダの実験 …… 4
4．2　バス・バッファＩＣ …… 4
(1) 信号線の方向とバッファの向き …… 5
(2) バッファ用ＩＣ …… 5
①７４ＬＳ２４５（双方向用） …… 5
②７４ＬＳ２４４（出力用） …… 6
課題２　双方向バスバッファ回路の実験 …… 6
4．3　コントロール バス・バッファ …… 7
4．4　アドレスバスバッファ …… 7
4．5　データバスバッファ …… 7
(1) データバスの方向がＣＰＵに向くとき …… 7

5．　８２５５　ＰＰＩ (Programable　Perripheral　Interface) …… １０
5．1　特徴と機能の概要 …… １０
5．2　８２５５の動作説明 …… １１
5．3　８２５５のモード …… １３
5．3．1　モードＯ（ベーシックモード） …… １３
5．3．2　モード１(ストローブ入出力) …… １４
5．3．3　モード２(ストローブ双方向バス入力) …… １６
(1) 制御信号の読み出し …… １７
モードセットのコントロールワード …… １７
5．3．4　８２５５の動作実験プログラム …… １８
例 ポートＡから入力したデータをそのままポートＢ、Ｃに出力する …… １８
8ﾋﾞｯﾄﾎﾟｰﾄを用いる場合　① 8086ｱｾﾝﾌﾞﾗで記述 …… １８
② N88-BASICで記述、③ TURBO C の場合 …… １９
16ﾋﾞｯﾄﾎﾟｰﾄを使用した場合　④ TURBO C の場合 、⑤ 8086ｱｾﾝﾌﾞﾗの場合 …… １９
例 LEDを右から左へ順次点灯 …… ２０
例 ﾋﾞｯﾄOのスイッチが ONで全部のLEDを点滅/OFFで点滅停止 …… ２０
5．3．5　割り込み処理 …… ２２
(1) ｉ８０８６の割込み …… ２２
(2) ＰＣ－９８０１の外部割込 …… ２３
(3) 割り込み処理プログラムの実験 …… ２４
実験、準備１、準備２ …… ２４
割り込み実験プログラム …… ２５
5．3．6　８２５９Ａ …… ２５
(1)８２５９Ａのイニシャライズの方法 …… ２６
(2)８２５９Ａのイニシャライズプログラム …… ２７
(3)ＥＯＩコマンドの送り方 …… ２８

(4)ＯＣＷコマンドフォーマット…… ２９
(5)外部割込み実験 …… ２９

６．　８２５３(Programmable　Interval Timer) …… ３３
(1)８２５３の基本動作…… ３３
(2)８２５３の特徴 …… ３３
(3)８２５３の動作説明 …… ３４
(4)ＭＰＵとの接続 …… ３５
６．１　８２５３の動作モード …… ３５
(1)モード０（カウンタ完了割り込み）…… ３５
(2)モード１（プログラマブルワンショット） …… ３６
(3)モード２（レートジェネレータ） …… ３６
(4)モード３（矩形波レートジェネレータ） …… ３６
(5)モード４（ソフトウエアトリガストローブ）…… ３７
(6)モード５（ハードウエアトリガストローブ）…… ３７
６．２　コントロールワードと初期値のロード ……３７
(1) コントロールワード ……３８
(2) カウンタのモニタ方法 ……３８
①読み出し動作 ……３８
②リードオンザフライ(Read on the fly) ……３９
③カウンタ２にカウンタラッチ・オペレーションを行うプログラミング ……３９
６．３　８２５３の割込み実験用のプログラム ……３９
実験

７．　８２５５の拡張Ｉ／Ｏボードの製作 ……４１
７．１　ＰＣ-９８０１／ＰＣ-８８０１の拡張スロットバス ……４１
７．２　周辺ＩＣをＭＰＵに接続する方法 ……４４
７．３　パソコンの拡張Ｉ／Ｏスロットに接続 ……４４
①ＡＣ特性について……４４
②ＤＣ特性に関して……４４
③リセットの接続方法…… ４４
④アドレスデコード回路…… ４５
１６ビットＩ／Ｏアドレスデコーダ…… ４５
７．３．１　ＰＣ－９８０１拡張Ｉ／Ｏﾎﾞｰﾄﾞ回路(８ビットデコード) …… ４６
７．３．２　ＰＣ－９８０１拡張Ｉ／Ｏﾎﾞｰﾄﾞ回路（１６ビットデコード）…… ４８
(1)アドレスデコーダ…… ４８
(2)アドレス／コントロールバスバッファ（７４ＬＳ２４４）…… ４８
(3)データバスバッファ（７４ＬＳ２４５）…… ４９
(4)８２５５の各ポート（ペリフェラル部）…… ４９
(5)１６ビットＩ／Ｏポートの部品とＩ／Ｏﾎﾞｰﾄﾞ部品配置図…… ４９
７．３．３　実験ボード …… ５１
７．３．４　Ｉ／Ｏボードの動作確認テストプログラム …… ５３
(1)Ｉ／Ｏボード動作確認テストプログラム(16ﾋﾞｯﾄﾃﾞｺｰﾄﾞ) …… ５３
(2)Ｉ／Ｏボード動作確認テストプログラム(8ﾋﾞｯﾄﾃﾞｺｰﾄﾞ) …… ５３
７．３．５　ＰＣ－９８０１拡張Ｉ／Ｏボード回路（１６ビットデコード）…… ５４
(1)拡張スロット用８２５５Ｉ／Ｏポートのアドレス …… ５５
(2)Ｃ言語を用いるときの手続き …… ５５
(3)動作確認テストプログラム …… ５５
実験　Ｃ言語を用いて全ポート出力、各ポートへ同じデータを出力 …… ５５
実験　Ｎ８８－ＢＡＳＩＣ（８６）を使う場合 …… ５６
７．６．６　ＰＣ－９８０１拡張Ｉ／Ｏボード回路（１６ビットデコード）…… ５７
８．　課　題 …… ５７
９．　参考資料 …… ５８

計算機の周辺インターフェース

インターフェースという言葉は曖昧で明確な定義はない。取り合えず，二つの異質なものを接続する境界あるいは同質でない別の系に属するものを接続すものとして捉えることにする。

マイコンに目を向けてみよう。 マイコンは CPU が１つの系をなしており，CPUの外界に他方の異なる系が存在している。さらに，2つの系を結んで信号の往来を可能にさせるものとしてバスが存在している。この状態において信号をバスにうまく乗せ，2つの系を取り持つように機能するものが周辺インターフェースと言える。このように見れば，周辺インターフェースは CPU と周辺の入出力機器との間にあって CPUの制御下で入出力機構を代行し効率的なデータ変換を行うデバイスであると言うことができる(周辺LSI)。

周辺LSIは，CPUおよびデバイスの進化にともなって多種多様のものが作られている。周辺LSIは，CPUによって制御し易く，かつ，CPUとデータ交換が出来易いように工夫されている。特殊な周辺機器コントロール用LSIは，どのような種類の CPU にも接続できるように作られている。RS232C，EIA232D，RS422や光アイソレータ(LED-トランジスタ/ダイオード型フォトカプラ)などの伝送用ラインインターフェースもある。インターフェースを信号レベルでみるとTTLレベルのものと，そうでないレベルのもの(例:C-MOSやRS-232C)がある。アナログ物理量とディジタル物理量の相互変換 (A/D-D/Aコンバータ)，トランスデユーサからの電流信号や電圧信号を相互に変換する信号処理用のLSIや，TTLレベルと他の信号レベル(C-MOS，RS-232Cなど)の整合を行うものなどもある。

１． 目 的

8080A/8085A と Z-80 の周辺インターフェースである PPI(8255)，PIT(8253)，さらにこれらを使うにあたって必要になるアドレスデコーダIC(LS138,LS139)，バスバッファIC(LS244，LS245)，割り込みコントローラIC(8259A)，多入力論理素子(LS30，LS133，LS688)を取り上げて，その特徴や動作，CPUとの接続方法，使い方などを学ぶ。得た知識をもとにしてZ80系あるいは8086系のパソコンであるPC-9801の拡張用I/Oポートに取り付けて使うインターフェースボードを設計・製作する。これによってアドレスデコードの方法と回路構成法，バスバッファ回路の構成法を，アドレスバス／データバス／コントロールバスと CPU との関係などを学ぶ。I/Oボードに LED，A/D変換器などを接続して駆動させることによって周辺インターフェースの一端を学ぶ。

２．インターフェース

(1)インターフェース用ＩＣ

① バス・ドライバ　　：TTLレベル　　→ 高出力TTLレベル
② バス・レシーバ　　：小入力TTLレベル → TTLレベル
③ バス・トランシーバ　：①と②の組み合わせ
④ ライン・ドライバ　　：TTLレベル　　→ 不平衡信号
　　TTLレベル　　→ 平衡信号
⑤ ライン・レシーバ　　：不平衡信号　　→ TTLレベル
　　平衡信号　　→ TTLレベル
⑥ ペリフェラル・ドライバ：TTLレベル　　→ 高出力オープンコレクタ
⑦ 専用ドライバ　　：TTLレベル　　→ 専用信号レベル
⑧ 専用センスアンプ　　：専用信号レベル → TTLレベル
⑨ その他

◇ ④，⑤は伝送用ラインのインターフェースでRS232C，EIA232D，RS422，IBMなどの規格を満たす。
◇ ⑥は電球を点灯したり，リレーやモータなどを動作させたりできる高出力が特徴である。
◇ ⑦，⑧はコアメモリ用のICなども含む。

(2) 周辺ＬＳＩ

マイコンの周辺ＬＳＩ(インターフェース)を機能別に分類し一覧表にしておく。これらの詳細はメーカが出している個別半導体データブックやＩＣデータブックを参照されたい。

表1．周辺ＬＳＩ一覧

Bus Arbiter/Transceiver	CDP68HC68 (4) , 8289 (1) , μPD71613 (5) , 8282/8283 (ALL) , 8286/8287 (ALL)
Clock Generator	μPD71011C (1, 2, 5) , MBL82284 (2) , 82C84A (2) , TMPZ84C60 (3) MC6875 (4) , MB8867 (4) , μPD71611 (5) , Z8581 (ALL)
real time Clock	SMC5242C (1, 2) , MC146818 (ALL) , MSM6242 (ALL) , RP5C01 (ALL) RP5C62 (ALL) , TC8250P (ALL) , TC8521P (ALL) , **μPD4990** (ALL)
Time/Counter	**8253** (1, 2) , **Z8430/Z80CTC** (3) , MC6840 (4) , Z8036 (6) , CDP1878 (ALL)
DMA Controller	μPD8257 (1) , **μPD8237** (1, 2) , Z8410R (3) , SCB68430 (7) , MC6844 (7) MC68450 (7) , μPD71071C (ALL)
DRAM Controller	M66201 (ALL)
Data Separator/VFO	MB4107 (ALL) , μPD71065 (ALL) , FDC9216 (ALL) , SED9420CAC (ALL) SMC9520C (ALL)
GPIB	MC68488 (7) , **μPD7210** (ALL) , TMS9914 (ALL)
SCSI	WD33C93/A (ALL) , NCR5380/S (ALL) , NCR5386 (ALL) , HD64951 (ALL) μPD72111 (ALL) , MB89351(ALL), MB89352P (ALL) , MB87030/31, AIC6250
Serial通信Controller	**8251A** (1, 2) , μPD7201A (1, 2) , Z8470 (3) , Z8440~Z8442 (3) , MB8868 (4) MC6850 (4) , Z8030 (6) , SCN68562R (7) , MC68681 (7) , MC68652 (7) MB8875 (ALL) , SCN2691 (ALL) , IM6402/6403 (ALL) , S65C51 (ALL) HD6852 (ALL) , LH8572 (ALL) INS8250 (ALL) , RF5C59 (ALL) SCN2641 (ALL) , SCN2651 (ALL) , STC9610F (ALL) , μPD72001 (ALL)
Parallel通信Controller	**8212** (1, 2) , 8216/8226 (1, 2) , **8255A** (1, 2) , 8279A (1, 2) , MBL8288 (1, 2) MBL82288 (1, 2) , MB89363 (1, 2) , **Z8420** (3) , MBL8243/8CS48(8048CPU) MC146823 (4) , MC6821 (4) , MC68230 (7) , CDP1851 (ALL)
割り込みController	**8259A** (1, 2) , CDP1877 (1800) , SCB68154 (7) , SCB68155 (7) , MB472 (4)
複合Controller	MC68901, 8256 (1, 2) , M5M82C800 (1, 2) , MB89392 (1, 2) **μPD71059C**
CRT&GRAPHIS-Controller	μPD3301 (1, 2) , MB89322 (1, 2) , TMS34010 (ALL) , HD63484 (ALL) MC6845 (ALL) , **μPD7220** (ALL) , MB89321 (4) , RP5C16 (ALL)
LCD Controller	HD64645 (1, 2. 3) , MSM6255 (1, 2. 3) , HD63645 (4) , MSM6240 (ALL) MSM6262 (ALL) , MSM6265 (ALL) , SED1330F (ALL) , T6963 (ALL) T7754 (ALL)
Floppy/Hard Disk Controller	HD63463 (7) , HD63265 (ALL) , WD1770/1772 (ALL) , FD179X-02 (ALL) MB89342 (1, 2) , MSM6241 (1, 2) , LH0110 (3) , MB89311 (ALL) μPD7260 (1, 2) , μPD7261 (1, 2) , **μPD765AC** (ALL) , μPD7262 (ALL)

下付きカッコ内の数字　1:8080A,8085A系　2:8086,80186,80286系　3:Z80系　4:6800系
5:Vｼﾘｰｽﾞ系　6:Z8000系　7:68000系　ALL:汎用ﾀｲﾌﾟ　その他：CPU名

(3) ８０８０Ａ／８０８５ＡとＺ-８０のファミリ

8255A/8255A-5(PPI)と8253/8253-5(PIT) および Z80PIO(Z8420)とZ80CTC(Z8430)を主たる題材として取り上げる。

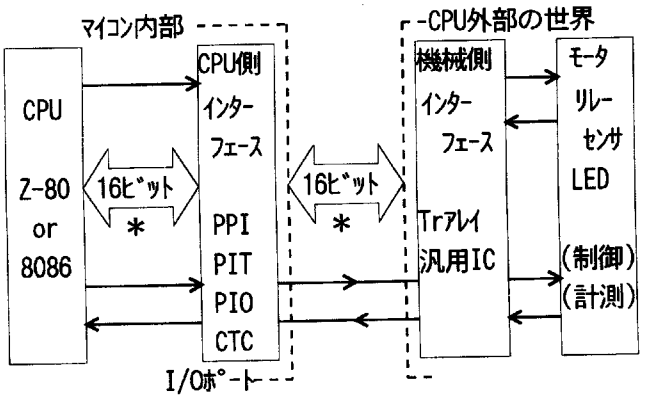
① 8255A／8255A-5(ｲﾝﾃﾙ) : CPU-8080A/8085A の Programmable Peripheral Interface(ﾊﾟﾗﾚﾙI/Oﾎﾟｰﾄ)
② 8253-5　(ｲﾝﾃﾙ) :　〃　の Programmable Interval Timer
③ Z8420(Z80PIO)(ｻﾞｲﾛｸﾞ) : CPU-Z80 の Parallel Input / Output Controller

◇ MK3881(ﾓｽﾃｯｸ社), LH-0081(ｼｬｰﾌﾟ:ﾉｰﾊﾞｰｼﾞｮﾝ2.5MHz MAX), LH-0081A(Aﾊﾞｰｼﾞｮﾝ 4MHz MAX)
LH-0081B(Bﾊﾞｰｼﾞｮﾝ 6MHz MAX), μPD780(NEC)

④ Z8430(Z80CTC)(ｻﾞｲﾛｸﾞ): 〃 の Counter Timer Circut(ｶｳﾝﾀ/ﾀｲﾏ)

3．パラレルＩ／Ｏポート

パラレル I/Oは図１に示すように，CPUから 8ビット(16ビット)のデータをパソコンの外部に伝えたり，外部の信号を 8ビット(16ビット)のデータとして CPUに伝えるポートである。



＊ Z-80CPUの場合は8ビット

図1 ＣＰＵとＣＰＵの外界との関係

4．インターフェース用ＩＣ

4．1 アドレス・デコーダ用ＩＣ

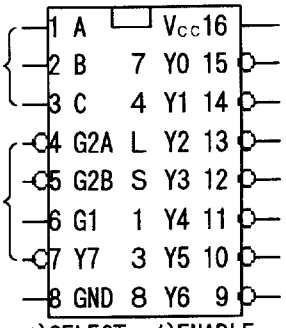
デコーダ入力端子にn個の2進数を入力したときの組み合わせの合計数に等しい数の出力端子が用意されていて入力端子に 0か 1の入力をしたとき，その組合せに対応した 1つの出力端子が選され，その出力端子から 0を，他の出力端子から 1を出力するように設計された ICをデコーダと呼ぶ。デコーダは CPUに接続された周辺機器の接続切り換えに用いられている。ここでは74LS138と74LS139とを取り上げて簡単な説明を加えておく。

表2 デコーダ・マルチプレクサ用ＩＣ

74LS42	74LS43	74LS44	74LS45	74LS137
74LS138	**74LS139**	74LS145	74LS154	74LS155
74LS156	74LS159	74LS255	74LS454	74LS2536
74LS2537	74LS2538	74LS2539	M54402	

(1)７４ＬＳ１３８

(3 to 8 Demultiplexer)
イネーブル入力のG1をH(1) ｱ)にし，GA(2)と(G2B)とをL(0)にした状態でセレクト力端子の A, B, CにH(1)ま ｲ)たはL(0)の信号を入力したときの状態によって特定の１つの出力端子が Lアクティブとなり他はハイインピーダンスになる。イネーブルが他の状態の時は出力端がすべてハイインピーダンスとなる(表３)。



4,5ﾋﾟﾝ:ｱｸﾃﾌﾞﾛｰ入力
Y_N :ｱｸﾃﾌﾞﾛｰ出力
$N=C\cdot2^2+B\cdot2^1+A\cdot2^0$

表3 ７４ＬＳ１３８(ﾃﾞｺｰﾀﾞ)の真理値表

入力					出力							
ENABLE		SELECT			DATA OUTPUTS							
G1	G2	C	B	A	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
*	1	*	*	*	1	1	1	1	1	1	1	1
0	*	*	*	*	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	1	1	1	1	1	1	0	1	1	1
1	0	1	0	0	1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	0	1	1	1	1	1
1	0	1	1	0	1	0	1	1	1	1	1	1
1	0	1	1	1	0	1	1	1	1	1	1	1

G2=G2A+G2B

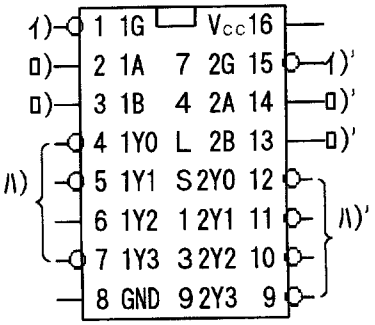
(例) C=1, B=0, A=1とするとコード $CBA=(101)_2=(5)_{10}$, 即ち $Y_5=0$ 他の Y_Nはすべて 1となって Y_5がセレクトされる。

(2)７４ＬＳ１３９(Dual 2 to 4 Demultiplexers)

入力端子はENABLEの G端子 ｲ)とSELECTの B, Aの 3入力であり，OUTPUTSのY0, Y1, Y2, Y3 がデコーダの出力となっている(パッケージには２組入っている)。

① ENABLEの G端子が 1(H レベル)になると B，A 端子の状態がどうあろうとデコードされない。

②ENABLEの G端子が 0(L レベル)になると B，A 端子によってデコードされる。



イ)、イ)' :ENABLE
ロ)、ロ)' :SELECT
ハ)、ハ)' :DATA OUTPUTS

表4　74LS139の真理値表

入力			出力			
ENABLE	SELECT		DATA OUTPUTS			
G	B	A	Y3	Y2	Y1	Y0
1	*	*	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

G=0 のとき 2ﾋﾞｯﾄの 2進数ｺｰﾄﾞ A,B に対して該当する出力 Y_Nが 0になる。

課題1　デコーダの実験

図2は，74LS138 を使ったデコーダの実験回路である。表3の真理値表を参照し適宜必要な入力を印加して Y_0から Y_7 の出力を順次セレクトして動作状況をテスタやオシロスコープを使って観察しデコーダの動作と使い方を深め考察と感想を加えよ。

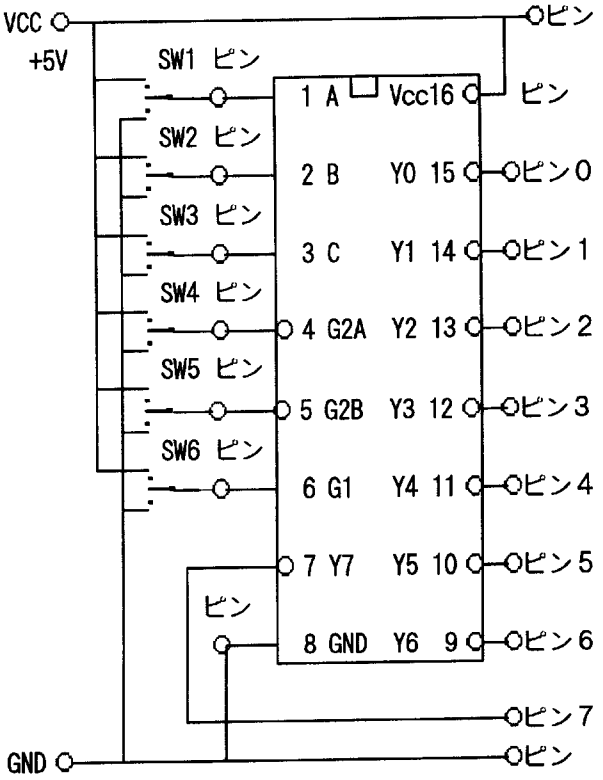
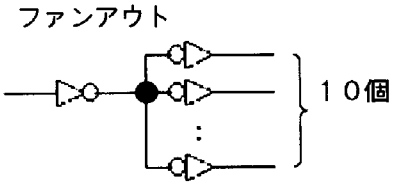


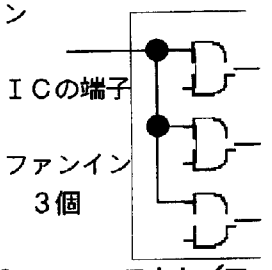
図2　74LS138を用いたデコーダの実験回路

4．2　バス・バッファIC

ICを使うときファンイン，ファンアウトという言葉に出会う。これは1個の TTL ICに対して何個まで TTL IC が接続できるかを示す数である(図3)。マイコン・システム・ディバイスは MOS型であり，トライステート機能をもっていてハイインピーダンス状態をとるので単純にはファンイン，ファンアウトは計算できない。Z-80 CPU の出力端子のドライブ(駆動)能力は，ノーマルタイプの TTLで1個，LSタイプの TTLなら4個までである。それ以上のドライブを行うと Hと Lの規定電圧を保てなくなり，誤動作の原因になる。実装面積が狭く，配線長が短い場合や接続する TTLの個数が少ないときはバッファ(緩衝器)は必要ない。それ以外のときにはバッファを設けてドライブ能力を強化する必要がある。



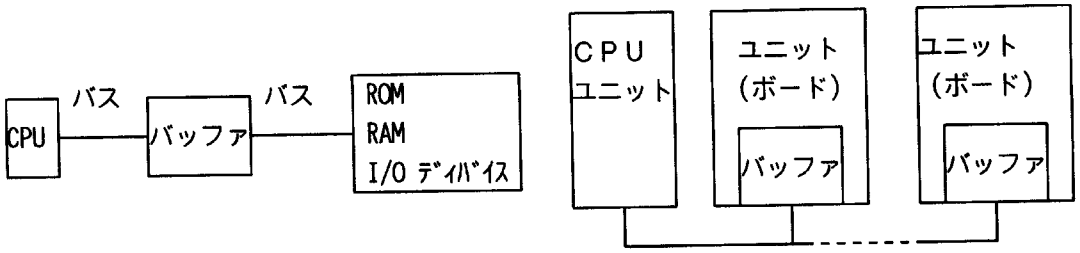
標準ＴＴＬでは１０個程度



ＩＣによっては１つの端子にＩＣ内部で幾つもゲートが接続されているものもある。この場合には，ＴＴＬの接続できる数は減る。

図3　TTL　ICのファンアウト／ファンイン

CPUのバスにはアドレス・バスとデータ・バス，コントロール・バスがあるが，図4(a)はこれらのバスにバッファを入れるときの基本的な挿入位置を示している。何枚もの基板に分かれてシステムが構成されていたり，各回路を別ボードに組むような場合には図4(b)のように各ボードの出入り口にトライステートのバッファを設け，ゲートの役割も持たせておく。必要に応じてバスバッファをハイインピーダンスにして CPUから切離して負担を軽くしたりする。



(a) バッファの基本的な挿入位置　　(b)　必要ならＣＰＵユニットにも挿入

図4　ＣＰＵとユニット（ボード）を接続する場合のバッファの挿入位置

（1）信号線の方向とバッファの向き

バッフの向きは CPUの各信号線の方向性に従って決める。

Z-80 CPU を基準として各信号線を分類すると入力，出力，双方向の３つがある(図５)。

①入　力：$\overline{BUSRQ}$ ←CPUﾊﾞｽ制御
$\overline{RESET}$, $\overline{WAIT}$, $\overline{INT}$, $\overline{NMI}$ ←CPU制御
ϕ ←ｸﾛｯｸ

②出　力：A0～A15 ←ｱﾄﾞﾚｽ・ﾊﾞｽ
$\overline{IORQ}$, $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$, $\overline{MI}$←ｼｽﾃﾑ制御
$\overline{HALT}$, $\overline{RFSH}$ ←CPU制御
$\overline{BUSAK}$ ←CPUﾊﾞｽ制御

③双方向：D0～D7 ←ﾃﾞｰﾀ･ﾊﾞｽ

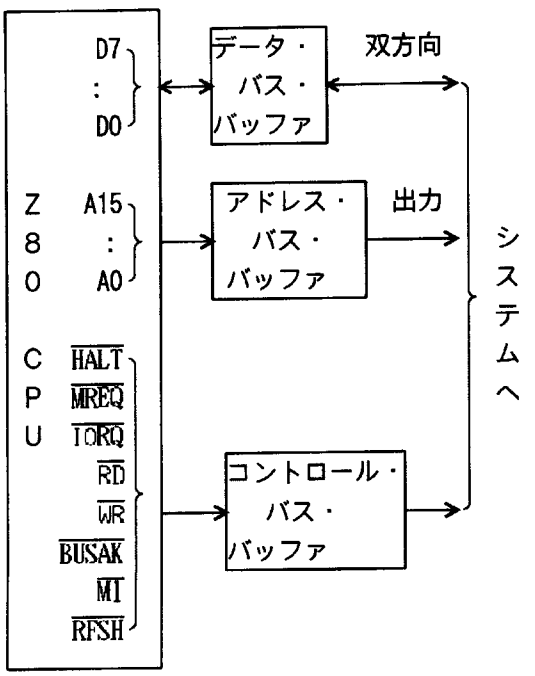


図5　信号線の方向とバッファの向き

（2）バッファ用ＩＣ

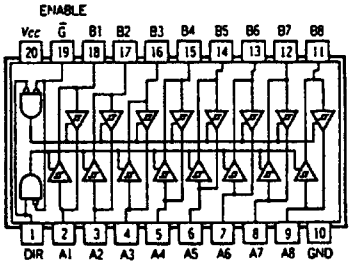
データ転送にDMA(Direct Memory Access)を使うと複雑になるので使わないことにした。表5に，トライステート・バスバッファ(Tri State Bus Buffers)ICを示す。74LS244(出力信号用)と74LS245を取り上げて簡単な説明を加える。

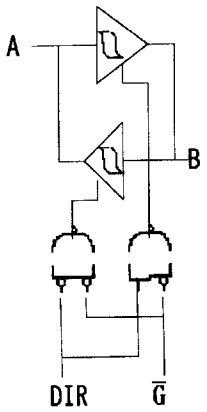
表5　トライステート・バス・バッファＩＣ

出力用	双方向用
125,126,240,241,**244**	242,243,8216,8T28,8T128
365,367,368,425,426	8T26,8T126,8T34,8T38,440
540,541,8T95,8T96	441,448,2905,2906,2907
8T97,8T98,81LS95,	226,8304,**245**,640,641,642
81LS96,81LS97,81LS98	643,644,645

①　７４ＬＳ２４５(双方向用)

図6に 74LS245 のピン接続図を示す。双方向用(Transceivers)として設計されているためゲート(G)の他に方向性を切り換える端子(DIR)が付加されている。ゲートの$\overline{G}$を Hレベルにすると双方向の両端が切断される(A off B)。ゲートの$\overline{G}$を Lレベルにすると DIR信号によって方向が決定され，双方向バッファとして機能する。





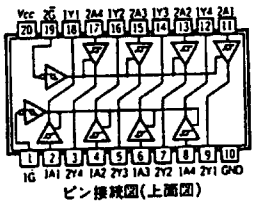
G端子	DIR端子	信号の伝達方向
L	H	A → B
L	L	A ← B
H	H,L	A<断>B

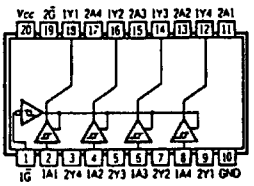
Gが L:この ICは双方向バッファ
動作し，その方向はDIRに
加わる信号によって変わる。
Gが H:バッファは<断>となる。

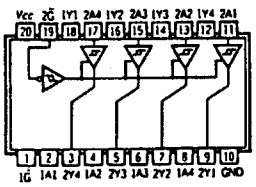
図6　７４ＬＳ２４５のピン接続図

②　７４ＬＳ２４４（出力用）

図7に 74LS244のピン接続図を示す。 ICには$\overline{1G}$と$\overline{2G}$のゲートがあり，各々 4つのバッファを制御している。$\overline{1G}$と$\overline{2G}$が Hレベルのとき Yはハイインピーダンスとなる。$\overline{1G}$と$\overline{2G}$が Lのとき Y=A となる。このように $\overline{1G}$と$\overline{2G}$のゲートが４つのバッファを制御している。 2つの回路を共通にして同時に制御すると，8回路(8ﾋﾞｯﾄ)分のバッファとなる。







(a)　　(b)　　(c)

図7　７４ＬＳ２４４のピン接続図と動作説明図

課題２　双方向バス・バッファ回路の実験

図８は，74LS245を使った双方向バス・バッファの実験回路である。74LS245のENABLEとDIRに信号を印加して A側から入ってくるデータは B側へ，B側から入ってくるデータは A 側へ通過，あるいはデータが入ってくるのを遮断するようにせよ。この動作状況をテスタやオシロスコープを使って観察しバッファの動作と使い方を理解し考察を加えよ。

(注) 入力，出力信号の印加，観測などはピンを両端をミノ虫クリップを取り付けたリード線を使って行うこと。

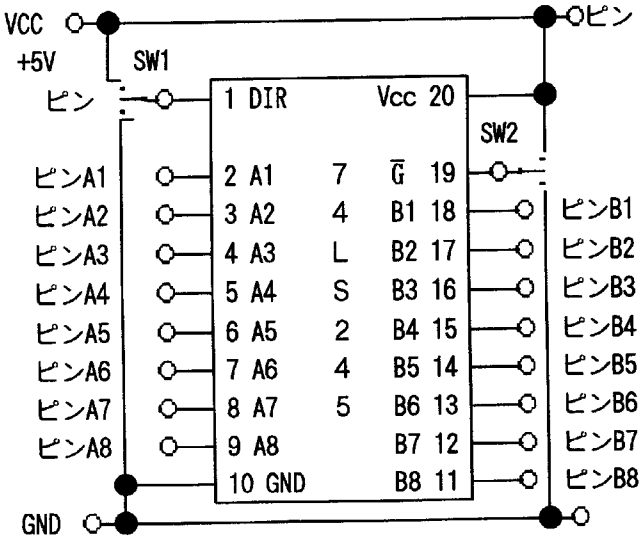


図8　双方向バス・バッファ７４ＬＳ２４５の実験回路

4．3　コントロールバスバッファ

コントロールバスバッファの信号方向は双方向性ではないので切り換える必要はない。コントロールバスの８本の信号線はトライステート出力線(４本)と非トライステート出力線(４本)に分けられる。

●トライステート出力　：$\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$

●非トライステート出力：$\overline{HALT}$, $\overline{BUSAK}$, $\overline{MT}$, $\overline{RFSH}$

74LS244(出力信号線用)を使ってトライステート出力グループは CPUの$\overline{BUSAK}$ 信号で制御し，非トライステートグループはゲート $\overline{G}$を Lレベルにして常に出力になるようにしておく。

【説　明】 $\overline{BUSAK}$ が Lレベルになるとトライステート出力信号線はハイインピーダンスになる。このため 74LS244のゲートを Hレベルにしなければならない。そこで，インバータを介してゲート $\overline{G}$を制御する。

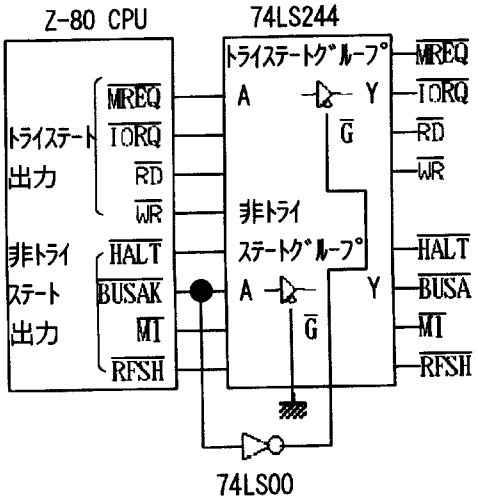


図9　コントロールバスバッファ

4．4　アドレスバスバッファ

アドレスバスの信号線は 16本(16ﾋﾞｯﾄ)あり，信号の方向は出力であるから切り換える必要がないので 74LS244(出力用)を２個使用する。但し，トライステート出力端子なので$\overline{BUSAK}$ 信号で制御した(**図１０**)。74LS244 のゲートコントロール端子(1ﾋﾟﾝ、19ﾋﾟﾝ)は GNDに接続しておいてもよい。**図１０**のように$\overline{BUSAK}$ を NOT回路を通して接続しておけばよい。通常は，$\overline{BUSAK}$ はノーアクティブ(Hレベル)であるので 1、19ピンは Lレベルとなってゲートが開く。

4．5　データ・バス・バッファ

データバスは，信号の方向が双方向性であるから何らかの方法で切り換える必要がある。さらにトライステート出力端子であるからアドレスと同様に$\overline{BUSAK}$ 信号で制御しなければならない。このため74LS245(双方向用)を用いることにした。74LS245の $\overline{G}$端子を Hレベルにすると機能としては方向性を持たなくなり，両端子間は OFFとなり，ハイインピーダンスになる。このことから方向を決めるには $\overline{G}$端子を$\overline{BUSAK}$ 信号で制御すればよいことが解るであろう。

データバスの信号の方向はメモリや I/Oディバイス等からデータを読み込む場合のみ CPUに向け，そうでない場合には常に CPUから外へ向ける(出力の方向)。このようにしておかないと CPUが正規の動作以外にバスラインからデータを取り入れてしまって誤動作の原因になる。

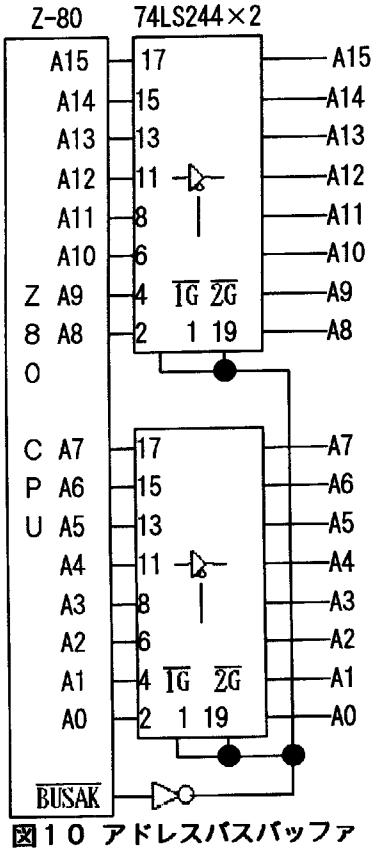


図１０ アドレスバスバッファ

(１) データ・バスの方向がＣＰＵに向くとき

データバスの信号の向きが CPUに向かうのは次の場合である

①オペコードフェッチのとき：

CPU が命令を実行する時に，まずプログラムカウンタで示された番地のデータをメモリから取り入れることである。

CPU はオペコードで命令の種類を判別する。

●アクテブ(Lレベル)になる制御信号線は $\overline{MREQ}$と $\overline{RD}$と $\overline{MT}$で

ある。

②メモリリードのとき：

CPUがメモリからデータを読み込む命令である。

●アクテブ(Lレベル)になる制御信号線は$\overline{MREQ}$と$\overline{RD}$である。

③入力ポートからのデータ・リードのとき：

I/O用ディバイスからデータを読み込む場合である。

●アクテブ(Lレベル)になる制御信号線は$\overline{IORQ}$と$\overline{RD}$である。

④割り込み時のベクトル・リードのとき：

プログラムの実行中に何か特別の仕事が割り込んだ場合，周辺ディバイスから前もって書き込んであったデータアドレスの一部として読み出すこと。 Z-80CPUのファミリディバイスを用いてモード2の割り込みを実行した場合，CPUの Iレジスタと，割り込み用周辺ディバイスが持つ Vレジスタの値で構成されるアドレスに割り込み処理としてプログラムの実行が移る。

●アクテブ(Lレベル)になる制御信号線は$\overline{IORQ}$と$\overline{M1}$である。

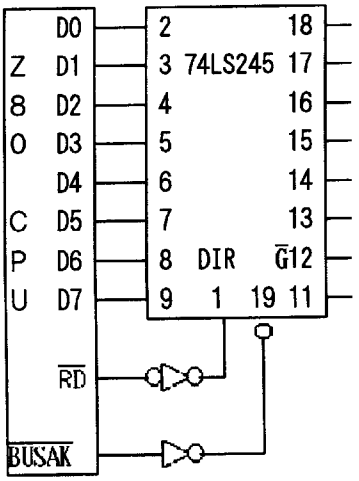
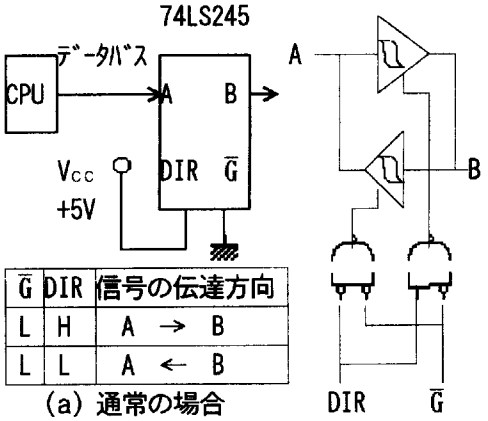


図１１　データバスバッファ例

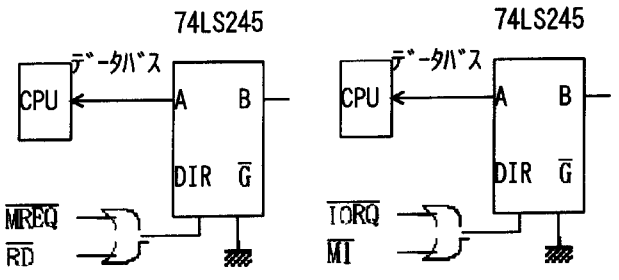
図１１にデータバスバッファの考え方の例を示した。データバスは双方向性であるので，DIR(1番端子)に印加する信号によって切り換える。 実際のシステムでは I/Oボードのデータバスも接続されるので DIR に加える信号は他の信号とのデコードが必要になる。

図１２に 74LS245を用いた方向の切り換えを図解した。図１２(a)の場合はデータバスは常に外を向いている。逆に DIRを Lレベルにするとデータバスは CPUの方に向う。必要に応じて向きを換えられるようにすることが重要である。すなわち，前述の ①～④ の状況のときだけデータバスを CPUに向けるようにすることである。これを分解して図示したものが図１２(b)～(e)である。 これを合成すると図１２(f)となる。図１２(f)を簡略化したものが図１２(g)である。$\overline{MREQ}$信号が無くなっているが他の回路との関係でここでは方向切り換えの信号要素としなくてもよいことになる。

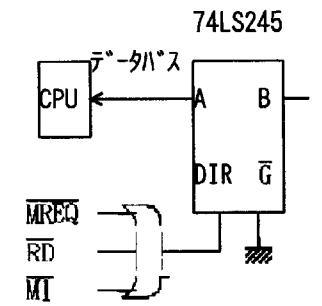


$\overline{G}$	DIR	信号の伝達方向
L	H	A → B
L	L	A ← B

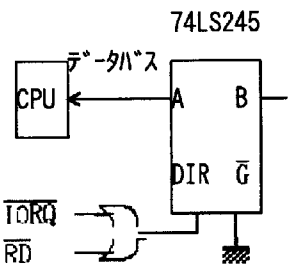
(a) 通常の場合



(c)メモリ・リードのとき　(e)割込時ﾍﾞｸﾄﾙ･ﾘｰﾄﾞ



(b)オペコード・フェッチのとき



(d)入力ﾎﾟｰﾄからのﾃﾞｰﾀ･ﾘｰﾄﾞ

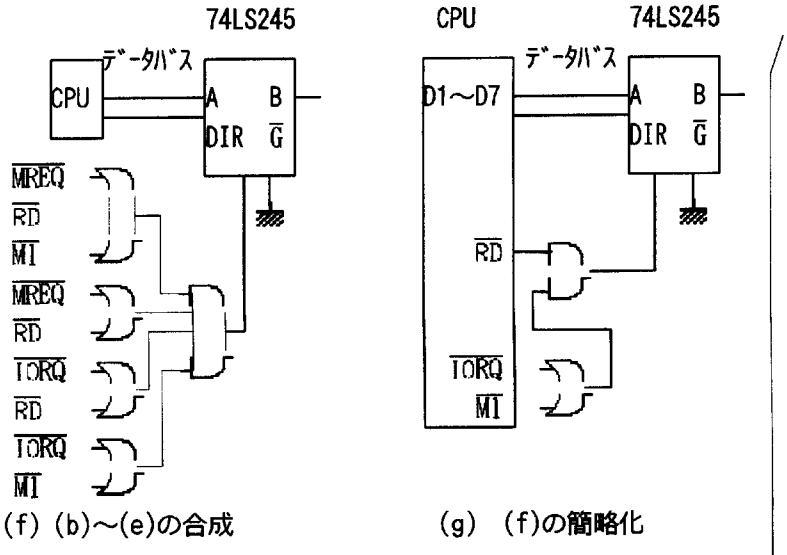


図１２　データ・バスの方向の図解

図１３に Z-80CPUに接続したバスバッファ回路図を示しておく。I/Oボードの制作の参考にするとよい。

バスバッファをつけるとバスを拡張してもトラブルが減少する。

またノーマル TTLが 15個程度ドライブ可能となる。

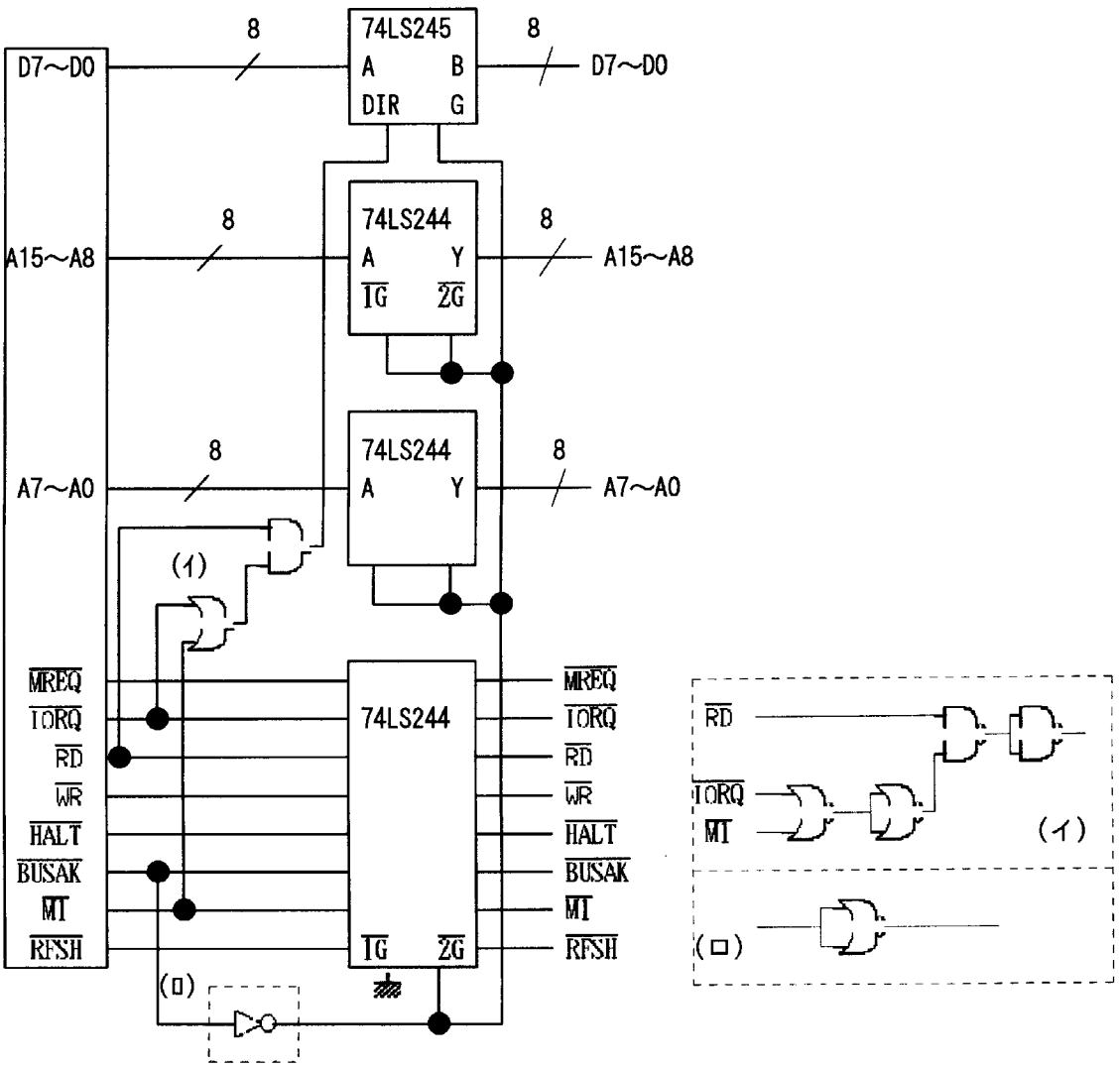


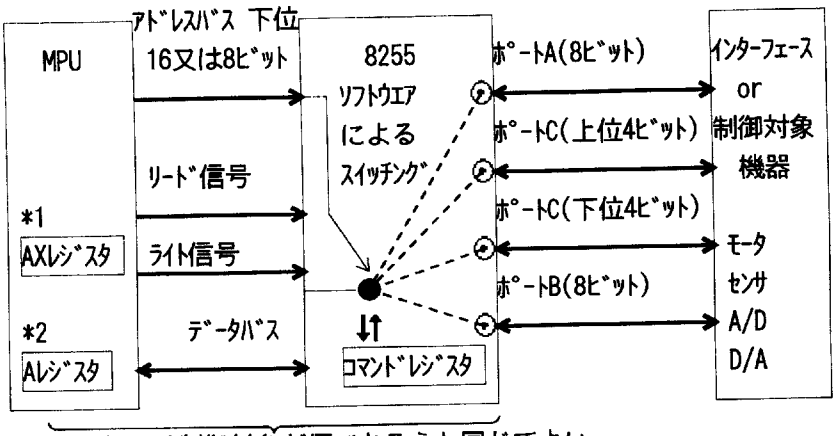
図１３　バス・バッファ回路図の詳細

5．８２５５　ＰＰＩ(Programable Peripheral Interface)

8255 は8080A MPU のパラレル入出力用の LSI として開発されタイミングや端子信号などは 8080A に合わせて作られている。8255A(8085A用)と 8255A-5(8085用)，スピードアップ化された8255A-2がある。違いは極わずかでソフトウエア上では全く同一。8255 は Z-80 MPU の汎用パラレルI/Oポートとしても利用できる。図１４に 8255 の機能を示した。MPU から出力命令またはデータ転送命令によって８ビットパラレルのデータがデータバスを通して 8255 に送り出される。8255は，このデータを受けて出力レジスタにデータをラッチ(固定)する。

外部のデータを入力する場合には， 8255 の外部端子に加わっている 8ビットデータがそのまま MPUに取り込まれる。 8255 は 8ビットの入出力ポートを 3組持っている。

3組のどのポートを使うかはプログラムでアドレス指定を行って決める。また入力ポートとして使うか出力ポートとして使うかはイニシャライズと呼ぶプログラムによって指定する。更に，オートマチックハンドシェークと呼ばれる入出力も可能である。



この部分は制御対象が何であろうと同じでよい

*1:i8086,i80386,i80486,iDX4など　*2:Z-80A,8085Aなど

図１４　８２５５の機能説明

5．1　特徴と機能の概要

8255 の特徴と機能を概説する。

① 入出力電圧レベル ： TTLコンパチブル。

② Hレベルの出力時 1.5V で 1mA 以上の電流が取り出せる。
小信号トランジスタ(ダーリントントランジスタ)を直接駆動できる(PBおよびPCﾎﾟｰﾄの任意の8ﾋﾞｯﾄ)。

③ モードO，1，2 の３種類のモードがある。

④ 8ビットのポートが 3つ(A，B，C)がある。入出力端子はプログラムにより機能を選択できる。

⑤ Cポートは上位 4ビット，下位 4ビットに分けて入出力の指定ができる。
ビットセット／リセット機能も付いている。

⑥ リセット入力(RESET=H)で入出力ポートの内部レジスタがすべてリセットされ，全入出力端子は入力モードになる(モードO　の入力ポートになる。入力端子は高インピーダンス状態)。

⑦ 出力ポートとして使う場合はイニシャライズが必要である。
出力ポートになる端子は，初期状態で Lレベルが出力される。

⑧ モードO (単純なデータの入出力)では，24ビット(8ﾋﾞｯﾄ*3ﾎﾟｰﾄ)となり，他のI/OポートLSI(Z80 PIO、PIA6821)に比べて有利である。

⑨ モード1(オートマチックハンドシェーク入出力)およびモード2(Aポートが双方向ポートになる)では Cポートの入出力線が制御信号線として使用される。

⑩ アドレスはアドレスバスの下位 16ビットまたは 8ビットで決まる。

⑪ 各々のポートは別々のアドレスを持ち，アドレスによって切り替えられる。

⑫ 各々のポートはデータの入出力にも使える。MPUや方式によって異なるが，データは主として AX(AL)レジスタ，Z-80では Aレジスタとの間で 8ビットパラレルに転送される。

⑬ どのような使い方をするかは，データの入出力をする前に決めておく必要がある(イニシャライズと呼び，データをコマンドレジスタ(CR)に出力する)。

⑭ 8255 は 4つのアドレスを持っている。

5．2　８２５５の動作説明

図１５に８２５５のブロック・ダイアグラムを，図１６にピンの定義を示して動作の概要を述べる。

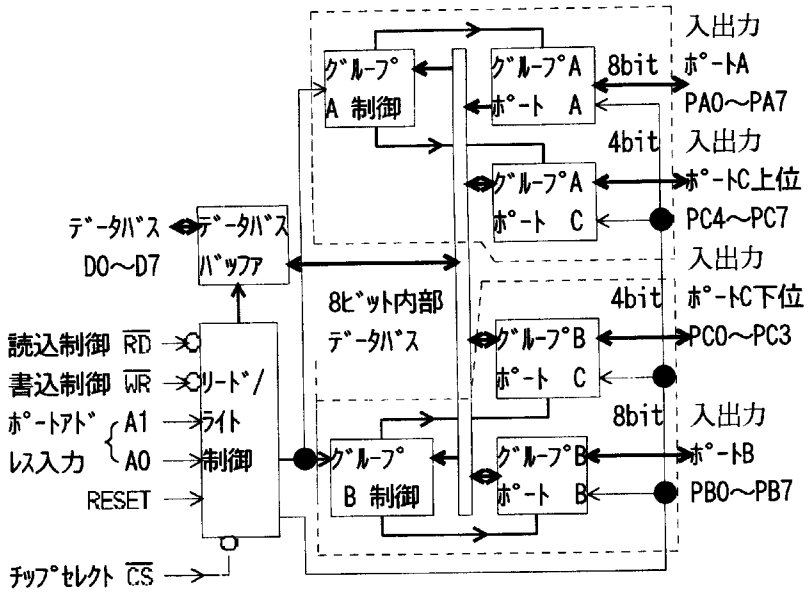


図１５　８２５５のブロック図

図１６　ピンの定義

① Ｖｃｃ(+5V)：ピン26，ＧＮＤ(0V)　：ピン７　・+5V 単一電源。消費電流は最大120mA(8255AC-2)。

② データバス(D7~D0)：ピン 27 ～ 34

・MPU のデータバスと接続される 8ビット双方向性バス。
・データバスバッファを通って 8255の 8ビット内部データバスにつながる。
・イニシャライズ時のコントロールデータの書き込み，その後に行われる入出力ポートと MPUデータバスとのデータ受け渡しに使用される。

③ PA_0～PA_7：ピン1～4，40～37：３ステート入出力

・ 8ビット入出力ポートA。
・ソフトウエアによってモード設定および入出力の設定を行う。
・ポートAは出力時はラッチ，入力時はラッチ，バッファとして動作。双方向性ﾊﾞｽとして動作する。

④ PB_0～PB_7：ピン18～25：３ステート入出力

・ 8ビット入出力ポートB。
・ソフトウエアによってモード設定および入出力の設定を行う。
・ポートB は出力時はラッチ，入力時はラッチやバッファとして動作。双方向性動作はしない。

⑤ PC_0～PC_7：ピン14～17，13～10：３ステート入出力

・8ビット入出力ポートC。
・ソフトウエアによってモード設定および入出力の設定を行う。
・モードコントロールによてモード0 を選択したときは，上位４ビットと下位４ ビットに分割された入出力ポートになる(出力時はラッチ，入力時はバッファ)。
・さらにモード１，２のハンドシェイクのコントロール信号にもなる。この時は，PC_0～PC_2の 3ビットをポートB 用へ，PC_3～PC_7 の 5ビットをポートA 用に使用する。

⑥ $\overline{CS}$：ピン 6：入力

・チップセレクト入力。
・複数のI/Oポートを使うとき，$\overline{CS}$には，MPUのアドレスをデコードして選択信号を与える。
・Lレベルで 8255 と MPU とのデータ転送が可能となる。

・Hレベルでは，データバスがハイインピーダンス状態になり MPU から切り離される。但し，出力ポートには前に出力されたデータが保持されている。
・複数のI/Oポートを使用するには，$\overline{CS}$ にアドレスデコーダより選択信号を加える。

⑦ リード／ライトコントロールロジック

データ，コントロールワード(イニシャライズ用のデータ)の転送を行う部分で，MPUのアドレス出力(A_0，A_1，$\overline{CS}$)，コントロールバス出力($\overline{RD}$，$\overline{WR}$)，リセット(RESET)の各信号を受けて，8255 内部の 2 つのコントロールグループに命令を出力する。

(ｲ)$\overline{RD}$：ピン 5 ：入力

・ MPUの $\overline{RD}$信号によってコントロールする。
・ リード信号， Lレベルで 8255 のポートに入力されているデータを MPU に転送する。
・ Z-80 と接続するときは，IN/OUT命令を実行するときに出力される $\overline{IORQ}$(I/Oﾘｸｴｽﾄ)信号と組み合わせて作られた信号を加える(図１７)。一般には，MPUからの$\overline{RD}$信号を加える。

(ﾛ)$\overline{WR}$：ピン36：入力

・ライト信号，Lレベルで MPU から 8255 にデータあるいはイニシャライズのコントロールワードを書き込む。
・Z-80 と接続するときは， $\overline{IORQ}$信号と組み合わせて作った信号を加える(図１７)。一般にMPUからの$\overline{WR}$信号を加える。

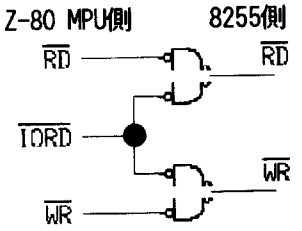


図１７ Z-80 MPU の場合の $\overline{RD}$ と $\overline{WR}$信号の作り方

⑧ A_0，A_1(ﾎﾟｰﾄ選択)：ピン9，8：入力

・ポートA，ポートB，ポートC およびコントロールレジスタの選択に使う。MPU のアドレスバス下位２ビットに接続する。16ビットMPUに接続するときは，ｱﾄﾞﾚｽﾊﾞｽのﾋﾞｯﾄ2とﾋﾞｯﾄ1に接続する。ﾋﾞｯﾄ0は偶数ｱﾄﾞﾚｽの選択に使う。 $\overline{CS}$，$\overline{RD}$，$\overline{WR}$の信号と合わせて 8255 の各ポートおよびコントロールレジスタのアドレスが決まる(表６)。

⑨ ＲＥＳＥＴ：ピン35：入力

・ H でコントロールレジスタを含む全内部レジスタをクリアする。
・全ポート(ポートA，B，C)はモード0 のインプットモード(ハイインピーダンス)になる。
・8255をﾘｾｯﾄする入力端子。8255 は電源を投入したあとﾘｾｯﾄされるまで入出力端子の機能や状態は不定である(ﾊﾟﾜｰ-ONﾘｾｯﾄができるようにする)。
・ Z-80 MPU はリセット入力が負論理(Lレベルでリセットされる)のため MPU のリセット信号をインバータを通して加える必要がある。

表6　８２５５のセレクト

A1	A0	$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	機能
0	0	0	0	1	ﾃﾞｰﾀﾊﾞｽ ← ﾎﾟｰﾄA
0	1	0	0	1	ﾃﾞｰﾀﾊﾞｽ ← ﾎﾟｰﾄB
1	0	0	0	1	ﾃﾞｰﾀﾊﾞｽ ← ﾎﾟｰﾄC
0	0	0	1	0	ﾎﾟｰﾄA ← ﾃﾞｰﾀﾊﾞｽ
0	1	0	1	0	ﾎﾟｰﾄB ← ﾃﾞｰﾀﾊﾞｽ
1	0	0	1	0	ﾎﾟｰﾄC ← ﾃﾞｰﾀﾊﾞｽ
1	1	0	1	0	ｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ ← ﾃﾞｰﾀﾊﾞｽ
X	X	1	X	X	ﾃﾞｰﾀﾊﾞｽはﾊｲｲﾝﾋﾟｰﾀﾞﾝｽ状態
1	1	0	0	1	組み合わせ禁止

⑩ グループA，グループBコントロール

・リード／ライトコントロールロジックからの命令に応じて，内部データバスのコントロールワードを受け取り，各ポートに対して命令を出す。
・これらは意識する必要はなくコントロールレジスタにコントロールワードを書き込むことによって各ポートの機能が決まり，また動作すると考えればよい。読出しは禁止されている。

⑪ ビットセット／リセット

・ポートC が出力ポートに指定されているとき，MPUからコントロールワードによって 8ビットのうちの任意の１ビットをセット(Hレベル)あるいはリセット(Lレベル)することができる。
・ビットセット／リセットの実行方法は，モードセット(イニシャライズ)の場合と同様であるが，コントロールワードが異なる。この機能は，モード1，モード2 のときのINT E(インタラプトイネーブルフラグ)のセット／リセットにも使われる。

5．3　8255のモード

8255には３種類(モード0，1，2)のモードがある。モードの選択は，8255のモードセットコントロールワードで行う。

(1) モードセット・コントロールワード

D7	D6	D5	D4	D3	D2	D1	D0

① D7 ビット：モードセットのコントロールワード　D7ﾋﾞｯﾄに 0 を立てるとﾋﾞｯﾄｾｯﾄﾓｰﾄﾞになる。
② D6,D5 ビット：グループＡモードセット
・ D6､D5=00 → モード0，　D6､D5=01 → モード1，　D6､D5=1X → モード２
③ D4 ビット：ポートA IN/OUT セット；D4=0 → OUTPUT ，D4=1 → INPUT
④ D3 ビット：ポートC(上位４ビット) IN/OUTセット；D3=0 → OUTPUT ，D3=1 → INPUT
⑤ D2 ビット：グループB モードセット；D2=0 → ﾓｰﾄﾞ0 ，　D2=1 → ﾓｰﾄﾞ1
⑥ D1 ビット：ポートB IN/OUTセット ；D1=0 → OUTPUT，　D1=1 → INPUT
⑦ D0 ビット： ポートC(下位４ビット) IN/OUTセット；D0=0 → OUTPUT，D0=1 → INPUT

5．3．1　モード0 (ベーシックモード)

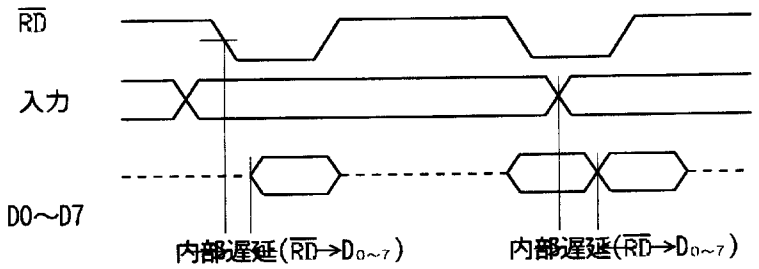
モードセット・コントロールワードのD7,D6,D5ビットを 100 にするとモード０となる。モード０は基本的な使い方で，データの入出力は周辺機器の都合や状態は考えず，MPUの一方的な命令によってデータの入出力を行う。モード０でも別に外部の状態を別のポートを使って入力し，またデータを出力する旨の通知を行うようにすればハンドシェイク転送が可能になる。一般的にはMPUの一方的な命令で動作させる。表７は 8255のコントロールワードである。ちなみにポートＡを入力，ポートＢ，ポートＣを出力とするときはコントロールワードは 90H となる。ＮＥＣのPC-9801の拡張I/Oポートの D0 番地を用いたときは，8255のコントロールレジスタは D6H となり，ここに90Hが出力される。プログラムは，次のように行う。

```
MOV AL,90H
OUT 0D6H,AL
IN  AL,0D0H
OUT 0D2H,AL
OUT 0D4H,AL
```

} ポートＡよりデータを入力。ポートＢとポートＣに出力する。

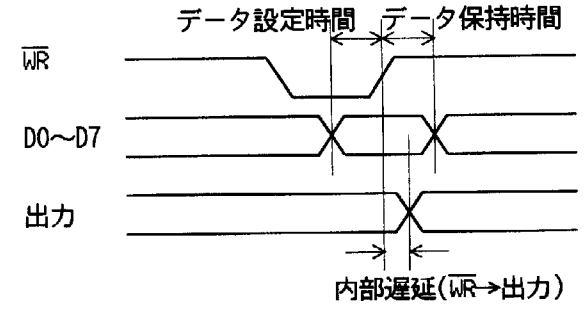
この５行分のプログラムの中にＩ／Ｏポートを通してのデータの入出力のほとんどが含まれている。データの入力と出力のタイミングを以下の図に示す。

モード0の入力タイミング



RD信号の立ち下がりでポート内にデータがを取り込み，データバス上に出力する。

モード0の出力タイミング



WR信号の立ち上がりで，データが入力されて，そのままポート出力となる。各ポートのデータは次のデータが入力されるまで保持される。

表7　モード0設定用コントロールワード

コントロールワード								16進	グループA		グループB	
D7	D6	D5	D4	D3	D2	D1	D0		ﾎﾟｰﾄA	ﾎﾟｰﾄC(上位)	ﾎﾟｰﾄC(下位)	ﾎﾟｰﾄB
1	0	0	0	0	0	0	0	80H	OUT	OUT	OUT	OUT
1	0	0	0	0	0	0	1	81H	OUT	OUT	IN	OUT
1	0	0	0	0	0	1	0	82H	OUT	OUT	OUT	1N
1	0	0	0	0	0	1	1	83H	OUT	OUT	IN	IN
1	0	0	0	1	0	0	0	88H	OUT	IN	OUT	OUT
1	0	0	0	1	0	0	1	89H	OUT	IN	IN	OUT
1	0	0	0	1	0	1	0	8AH	OUT	IN	OUT	1N
1	0	0	0	1	0	1	1	8BH	OUT	IN	IN	IN
1	0	0	1	0	0	0	0	90H	IN	OUT	OUT	OUT
1	0	0	1	0	0	0	1	91H	IN	OUT	IN	OUT
1	0	0	1	0	0	1	0	92H	IN	OUT	OUT	1N
1	0	0	1	0	0	1	1	93H	IN	OUT	IN	IN
1	0	0	1	1	0	0	0	98H	IN	IN	OUT	OUT
1	0	0	1	1	0	0	1	99H	IN	IN	IN	OUT
1	0	0	1	1	0	1	0	9AH	IN	IN	OUT	1N
1	0	0	1	1	0	1	1	9BH	IN	IN	IN	IN

OUT:出力ポート，　IN:入力ポート　の意味

5．3．2　モード1（ストローブ入出力）

モード1 は，外部機器とMPU(または8255)が互いに確認を行いながらデータの入出力を行う。ポートAとポートB をデータの入出力ポートに使い，ポートC はそれぞれのコントロール信号として使う(オートマティック・ハンドシェークと呼ぶ)。このためコントロール，ステータス信号として表８に示すようにポートＣの各ビットを信号線として使用する。

・ポートＡとポートＢの入出力の組み合わせは次の４通りとなる。

① ポートＡ，ポートＢともに入力
② ポートＡ，ポートＢともに出力
③ ポートＡ入力，ポートＢ出力
④ ポートＡ出力，ポートＢ入力

表8　モード1のポートCの機能

	制御機能	ﾎﾟｰﾄA	ﾎﾟｰﾄB
入力	$\overline{STB}$	PC4	PC2
	IBF	PC5	PC1
	INTR	PC3	PC0
出力	$\overline{OBF}$	PC7	PC1
	$\overline{ACK}$	PC6	PC2
	INTR	PC3	PC0

●$\overline{STB}$（ストローブ入力）

・Ｌレベルで外部機器から出力されているデータ，即ち 8255 の入力ポートに加わっているデータを 8255 の入力レジスタにラッチする。
・データをラッチするためのクロック信号の働きをしている。MPU からのコントロール信号には関係なく任意の時点で外部からのデータを 8255 にラッチする。
・このデータは MPU が IN 命令を実行するまで(8255に$\overline{RD}$信号が加わるまで)，データバスには出力されない。

●ＩＢＦ（入力バッファフルフラグ出力）

・外部からのデータが 8255 の入力レジスタに保持されると，

(1)モード１の入力例

ｺﾝﾄﾛｰﾙﾜｰﾄﾞ

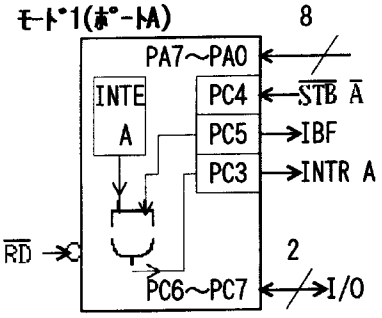
D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	I/O	X	X	X

↓

PC6,PC7
1:入力，0:出力

この出力が H レベルになる。
・この信号は $\overline{STB}$ の立ち下がりでセット(Hレベル)され，$\overline{RD}$ 信号の立ち上がりでリセット(Lレベル)される。

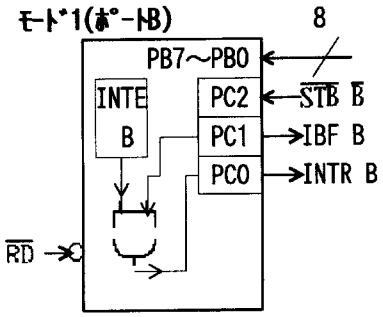
<信号の意味>
・ 8255 の入力レジスタにデータがラッチされており，これ以上入力できないことを外部に知らせる。
・ MPU が IN 命令を実行して読み込むと，$\overline{RD}$信号が出てIBF がLレベルになり，入力バッファは空になって次のデータが入力可能であることを外部に知らせる。



ｺﾝﾄﾛｰﾙﾜｰﾄﾞ

D7	D6	D5	D4	D3	D2	D1	D0
1	X	X	X	X	1	1	X

●ＩＮＴＲ（割り込み要求出力，データ入力の場合）
・入力レジスタにラッチされているデータの割り込み処理に用いる。
・８２５５内部のＩＮＴＥ（インタラプトイネーブルフラグ）がセットされているとき(Ｈレベル)，$\overline{STB}$入力が入るとＩＢＦがＨレベルになるが，ＩＮＴＲはこの$\overline{STB}$入力の立ち上がった直後，Ｈレベルになる。さらに$\overline{RD}$信号の立ち下がりでリセットされＬレベルになる。
・ＩＮＴＲ出力を割り込み要求信号としてＭＰＵの割り込み要求入力に加えておけば，外部よりデータが送られた合図の$\overline{STB}$入力の後， 割り込み処理で８２５５入力レジスタのデータをＭＰＵに取り込むことができる。
・いつ送られてくるかわからないデータの処理や，ＭＰＵの動作に比べて遅いデータの処理に適する。
・ＭＰＵは，外部データが用意できたか否かを気にすることなくプログラムを進めることができる。
・グループA の $INTE_A$ は PC4(ポートC のビット4)のビットセットによって立てる。
・グループB の $INTE_B$ は PC2(ポートC のビット2)のビットセットによって立てる。



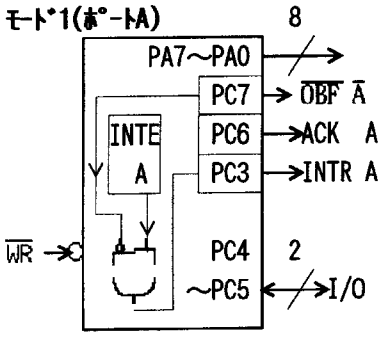
(2)モード１の出力例

ｺﾝﾄﾛｰﾙﾜｰﾄﾞ

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	I/O	X	X	X

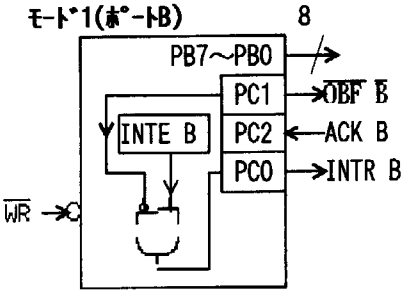
↓

PC4,PC5,1:入力，0:出力



ｺﾝﾄﾛｰﾙﾜｰﾄﾞ

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	I/O	X	X	X



●$\overline{OBF}$（出力バッファフルフラグ出力）
・$\overline{WR}$信号の立ち上がりでＬレベルになる。
・更に $\overline{ACK}$（アクノレッジ入力)の立ち下がりで H レベルになる。
・ MPU が 8255 に対してデータを出力すると，ポート出力にデータが出力されたことを外部に知らせる信号となる。
・外部では$\overline{OBF}$が L レベルになったらデータを受け取り，受け取ったら $\overline{ACK}$ を返す。

●$\overline{ACK}$（アクノレッジ入力）
・外部からの 8255 に対してデータを受け取ったことを知らせる信号である。
・ Lレベルのパルス信号を加える。

●ＩＮＴＲ（割り込み要求出力，データ出力の場合）
・外部が MPU から 8255 を通してデータを受け取った時，この INTR を H レベルにして MPU に割り込みをかけるのに使う。
・INTE(インタラプトイネーブルフラグ)が H レベルのとき，

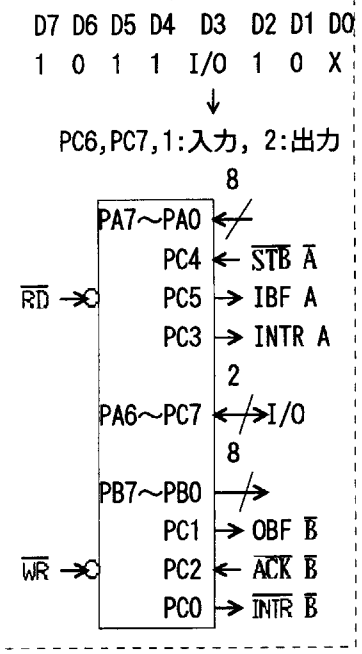
$\overline{ACK}$信号が入ると $\overline{OBF}$ が H レベルになり，INTR信号は $\overline{ACK}$ 信号の立ち上がった直後 H レベルになる。

・MPU が 8255 にデータを書き込んだとき $\overline{WR}$ の立ち下がりでリセットされ L レベルになる。

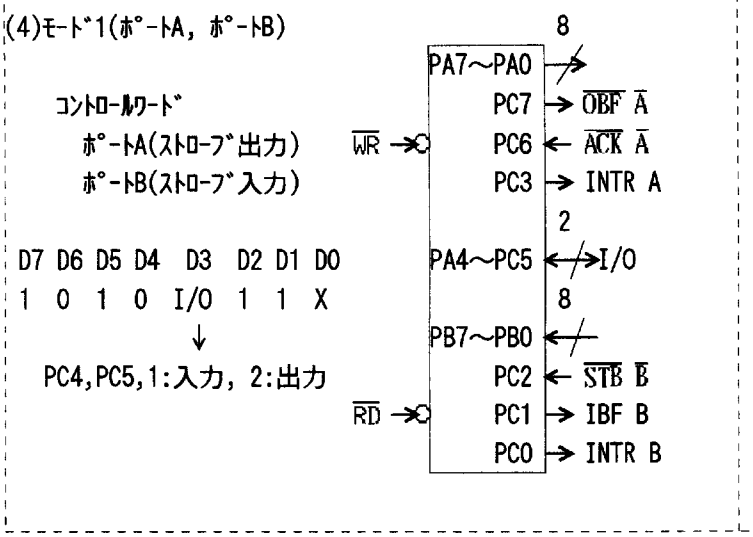
・グループA の $INTE_A$ は PC6 のビットセットで立てる。

・グループB の $INTE_B$ は PC2 のビットセットで立てる。

(3) ﾓｰﾄﾞ1(ﾎﾟｰﾄA, ﾎﾟｰﾄB)



(4)ﾓｰﾄﾞ1(ﾎﾟｰﾄA, ﾎﾟｰﾄB)



5．3．3　モード2(ストローブ双方向バス入出力)

モード2 はポートA(グループA)だけに適用可能で， 8ビット双方向性バスポートとして動作する。ポートA が 8ビットの双方向性バスポートになってポートC の上位 5ビットがコントロールポートになる。バスポートは入出力ともに内部レジスタを持っていて，コントロールポートからの制御信号によく似ていて，MPU への割り込みコントロールも可能である。

グループA をモード2 で設定するとき，グループB はモード0 または モード1 に設定できる。

これよりグループAをモード2 で使うときの制御信号について簡単な説明を加える。

●$\overline{OBF}$ (出力バッファフルフラグ出力)

・MPU がポートA にデータを書き込んだとき，$\overline{OBF}$ は L レベルになる。

・この信号を利用して MPU からデータが受け取り可能であることを外部に知らせる。

・モード1 とは違って，この時，まだポートA はフローティング(ハイインピーダンス)状態である。

●$\overline{ACK}$ (アクノレッジ入力)

・ $\overline{ACK}$ 入力を L レベルにすると，内部出力レジスタの内容はポートA に出力される。

・ 即ち外部機器側が $\overline{OBF}$ を見て $\overline{ACK}$ を返すことでポートA にデータが出力される。

・ $\overline{ACK}$ を返すまでは(Hレベルでは)ポートA はハイインピーダンス状態のままである。

●$\overline{STB}$ (ストローブ入力)，ＩＢＦ (入力バッファフルフラグ出力)

・この２つは(実質的には $\overline{OBF}$ とINTRも) モード1 における場合と同じ働きをする。

●ＩＮＴＲ (割り込み要求出力)

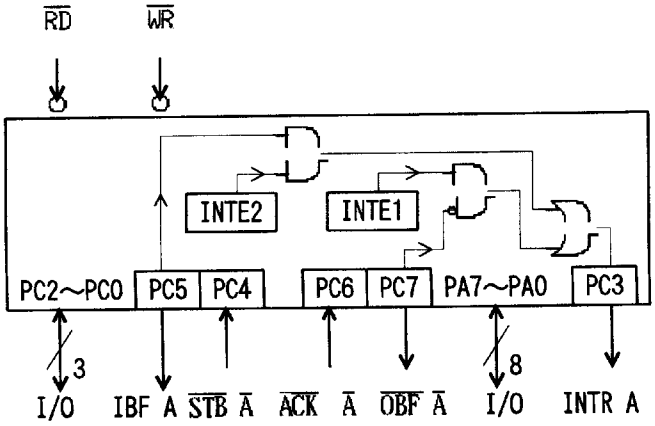
・MPU に割り込みをかけるための出力で動作はモード1 と同じである。

・割り込み受け付けフラグ(INTE)は２個あって，モード1 アウトプットの時の$INTE_A$ とモード1 インプットの時の$INTE_B$にそれぞれ対応する。

・INTE1 は $\overline{OBF}$ ， $\overline{ACK}$ と組み合わせてINTR信号を作るときに用いる。
MPU からの PC6 のビットセットによって立てることができる。

・INTE2 は IBF, $\overline{STB}$ と組み合わせてINTR信号を作るときに用いる。

MPU からの PC4 のビットセットによって立てることができる。



コントロールワード

D7	D6	D5	D4	D3	D2	D1	D0
1	1	X	X	X	I/O	1/0	1/0

D0ビット
PC2～PC0
1:入力，0:出力

D1ビット
ポートB
1:入力，0:出力

D2ビット
グループB モード
0: モード0，1:モード1

（1）制御信号の読み出し

ポートC をコントロールポートとして使っている場合(モード1，モード2としている場合)は，ポートC を入力することによりコントロール信号およびバスステータス信号を読み出すことができ，これによって制御信号の状態を知ることができる。表8に各ビットの一覧を示す。

表9　制御信号の読み出し

データ / モード	ビット D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
モード1入力	I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$
モード1出力	$\overline{OBF}_A$	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	$\overline{OBF}_B$	$INTR_B$
モード2	$\overline{OBF}_A$	INTE1	IBF_A	INTE2	$INTR_A$	ｸﾞﾙｰﾌﾟBのﾓｰﾄﾞによる		

表10　モード1設定コントロールワード

コントロールワード									グループA					グループB			
										ポートC				ポートC			
D7	D6	D5	D4	D3	D2	D1	D0	16進	ﾎﾟｰﾄA	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	ﾎﾟｰﾄB
1	0	1	0	0	1	0	X	A4 A5	OUT	$\overline{OBF}_A$	$\overline{ACK}_A$	OUT		$INTR_A$	$\overline{ACK}_B$	$\overline{OBF}_B$	$INTR_B$	OUT
1	0	1	0	0	1	1	X	A6 A7	OUT	$\overline{OBF}_A$	$\overline{ACK}_A$	OUT		$INTR_A$	$\overline{STB}_B$	IBF_B	$INTR_B$	IN
1	0	1	0	1	1	0	X	AC AD	OUT	$\overline{OBF}_A$	$\overline{ACK}_A$	IN		$INTR_A$	$\overline{ACK}_B$	$\overline{OBF}_B$	$INTR_B$	OUT
1	0	1	0	1	1	1	X	AE AF	OUT	$\overline{OBF}_A$	$\overline{ACK}_A$	IN		$INTR_A$	$\overline{STB}_B$	IBF_B	$INTR_B$	IN
1	0	1	1	0	1	0	X	B4 B5	IN	OUT		IBF_A	$\overline{STB}_A$	$INTR_A$	$\overline{ACK}_B$	$\overline{OBF}_B$	$INTR_B$	OUT
1	0	1	1	0	1	1	X	B6 B7	IN	OUT		IBF_A	$\overline{STB}_A$	$INTR_A$	$\overline{STB}_B$	IBF_B	$INTR_B$	IN
1	0	1	1	1	1	0	X	BC BD	IN	IN		IBF_A	$\overline{STB}_A$	$INTR_A$	$\overline{ACK}_B$	$\overline{OBF}_B$	$INTR_B$	OUT
1	0	1	1	1	1	1	X	BE BF	IN	IN		IBF_A	$\overline{STB}_A$	$INTR_A$	$\overline{STB}_B$	IBF_B	$INTR_B$	IN

グループA，グループBのモードは独立に設定できる。
グループA，グループBともモード1である必要はない。

表１１　モード２設定用コントロールワード

コントロールワード									グループA						グループB			
D7	D6	D5	D4	D3	D2	D1	D0	16進	ポートA	ポートC					ポートC			ﾎﾟｰﾄB
										PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	
1	1	X	X	X	0	0	0	C0	双方向バス	$\overline{OBF}_A$	$\overline{ACK}_A$	IBF_A	$\overline{STB}_A$	$INTR_A$	OUT			OUT
1	1	X	X	X	0	0	1	C1		$\overline{OBF}_A$	$\overline{ACK}_A$	IBF_A	$\overline{STB}_A$	$INTR_A$	IN			OUT
1	1	X	X	X	0	1	0	C2		$\overline{OBF}_A$	$\overline{ACK}_A$	IBF_A	$\overline{STB}_A$	$INTR_A$	OUT			IN
1	1	X	X	X	0	1	1	C3		$\overline{OBF}_A$	$\overline{ACK}_A$	IBF_A	$\overline{STB}_A$	$INTR_A$	IN			IN
1	1	X	X	X	1	0	X	C4		$\overline{OBF}_A$	$\overline{ACK}_A$	IBF_A	$\overline{STB}_A$	$INTR_A$	$\overline{ACK}_B$	$\overline{OBF}_B$	$INTR_B$	OUT
1	1	X	X	X	1	1	X	C6		$\overline{OBF}_A$	$\overline{ACK}_A$	IBF_A	$\overline{STB}_A$	$INTR_A$	$\overline{STB}_B$	IBF_B	$INTR_B$	IN

５．３．４　８２５５の動作実験プログラム

単に例題のプログラムを実行して分かったと言っているだけでは意味がない。コンピュータシステムを使って単にプログラムする人にとっては 8255などを知らなくても済むであろう。しかし，システムの設計者は，周辺機器などの初期状態を知り，電源をＯＮしたら暴走してシステムを破壊してしまうような動作が生じないようにしなければならない。万一，コンピュータや周辺機器が破壊されたとしても，機械が安全に停止するように設計しなくてはならない（災害と危害防止などの安全対策）。そのためにはソフトウエアとハードウエアの両方の知識が必要となる。

8255等のプログラマブルなLSIを使うときは，次のように考えればよい。

①基本的にどのような動作をするデバイスであるかを知る。
②PMUおよび周辺機器との接続方法およびどのように接続されているかを調べる。
③イニシャライズノの方法。
④各モードにおける動作を調べ，イニシャライズの方法を知り，プログラミングしてみる。
⑤データの入出力タイミング。
⑥初期状態はどのようになっているか。
⑦イニシャライズによるポートの変化。
⑧電源投入時またはスタート時の周辺機器の正常状態の維持方法。

[準備]

・PC-9801の電源を切り，拡張ポート＃１に図３０または図３３の8255I/Oﾎﾞｰﾄﾞを挿入する。
・８ビットまたは１６ビットの実験ボードを接続する。
・制御用に使うソフトウエアとして，MS-DOS Ver2.1かVer3.3，MASM　Ver1.27(MS-DOS添付のもの)，N_{88}-BASIC(86)，TURBO C Ver2.0 を準備する。

【例１】ポートＡから入力したデータをそのままポートＢ，ポートＣに出力する。

[８ビットボードを用いる場合]

①　8086アセンブラで記述

```
;   8255 TEST (8BITS)
;
      PA EQU 0D0H
      PB EQU 0D2H
      PC EQU 0D4H
      CR EQU 0D6H
      CW EQU 90H
CODE  SEGMENT                    ┐
      ASSUME CS:CODE,DS:CODE     ├ MASMの疑似命令
      ORG   100H                 ┘
START:
      MOV     AL,CW      ┐  8255のｲﾆｼｬﾗｲｽﾞ
      OUT     CR,AL      ┘
M1    IN      AL,PA   ﾎﾟｰﾄAよりﾃﾞｰﾀ入力
      OUT     PB,AL   ﾎﾟｰﾄBへﾃﾞｰﾀ出力
      OUT     PC,AL   ﾎﾟｰﾄCへﾃﾞｰﾀ出力
      JMP     M1   繰り返し
;
CODE  ENDS                       ┐ MASMの疑似命令
      END     START              ┘
```

② N88-BASIC86)で記述

```
100 '8255 TEST (8BITS)
110 PA=&HD0  '8255 ADDRESS
120 PB=PA+2
130 PC=PA+4
140 CR=PA+6
150 CW=&H90 'CONTROL WORD
160 OUT CR,CW '8255 INITIALIZE
170 D=INP(PA) 'DATA INPUT
180 OUT PB,D  'DATA OUT
190 OUT PC,D  'DATA OUT
200 GOTO 170
210 END
```

③ TURBO C で記述

```
/* 8255 TEST PROGRAM */
#include <stdio.h>
#include <dos.h>
#define PA 0xd0
#define PB 0xd2
#define PC 0xd4
#define CR 0xd6
#define CW 0x90
main()
{
   unsigned char d;
   outportb(CR,CW);
   while(d != 0xff){
      d = inportb(PA);
      outportb(PB,d);
      outportb(PC,d);
    }
}
```

[１６ビットボードを用いる場合]

8ビットボードを用いる場合と同様のプログラム例を示す。データは16ビットであってもアドレスは 8ビットによる指定ができる。この16ビットボードは，後述の7.3.2節で示したように，ディップスイッチの切り替えによって上位8ビットのアドレスを自由にセットできるようにしてある。

ここでは**ディップスイッチのビット7 を OFF(HLﾍﾞﾙ)にして上位アドレスを 80H** にした。従って**8255のアドレスは 80D0H** となる。

- 8086のアセンブラにおける，16ビットによるアドレスの場合は，DXレジスタにアドレスを入れて，このレジスタによってアドレス指定をしなければならない。
- N88-BASIC(86)では，アドレスもデータも 8ビットしか扱えない。従って，アドレス選択のディップスイッチを全部 ON にして上位8ビットが 0 になるようにする。
- TURBO C では，8ビットと16ビットデータでは，I/Oポートに対する入出力関数に違いがある。

④ TURBO C の場合

```
/* 8255 I/O BOARD TEST(16BITS) */
#include <stdio.h>
#include <dos.h>
#define PA 0x80d0   ┐ 0x80d0
#define PB 0x80d2   │   ～
#define PC 0x80d4   │ 0x80d6
#define CR 0x80d6   │
#define CW 0x9090   ┘
main()
{
   unsigned int d=0;
   outport(CR,CW);
   while(d != 0xffff){
         d = inport(PA);
         outport(PB,d);
         outport(PC,d);
      }
}
```

⑤ 8086アセンブラの場合

```
;  8255 I/O BOARD TEST(16BITS)
        PA EQU  80D0H ;8255 ADDRESS
        PB EQU  80D2H
        PC EQU  80D4H
        CR EQU  80D6H
        CW EQU  9090H
CODE    SEGMENT
        ASSUME CS:CODE,DS:CODE
        ORG     100H
START:
        MOV     DX,CR  ;8255 INITIALIZE
        MOV     AX,9090
        OUT     DX,AX
MAIN:
M1      MOV     DX,PA  ;DATA INPUT
        IN      AX,DX
        MOV     DX,PB ;DATA OUTPUT
        OUT     DX,AX
        MOV     DX,PC
        OUT     DX,AX
        JMP     M1
CODE    ENDS
        END     START
```

【例２】　ＬＥＤを右から左へ順次点灯

```
100 ' TEST (8BITS)
110 PA=&HD0 '8255 ADDRESS
120 PB=PA+2
130 PC=PA+4
140 CR=PA+6
150 CW=&H90 'CONTROL WORD
160 OUT CR,CW '8255 INITIALIZE
170 D%=1       'DATA SET
180 OUT PB,D% 'DATA OUT
190 GOSUB *TIMER
200 D%=D%*2    'DATA SHIFT
210 IF D%>128 THEN D%=1
220 GOTO 170
230 *TIMER
240 FOR T=1 TO 1000
250 NEXT I
260 RETURN
270 END
```

```
100 TEST (16BITS)
110 PA0=&HD0 '8255ADDRESS
120 PA1=PA0+1
130 PB0=PA0+2
140 PB1=PA0+3
150 PC0=PA0+4
160 PC1=PA0+5
170 CR0=PA0+6
180 CR1=PA0+7
190 CW=&H90 'CONTROL WORA
200 OUT CR0,CW '8255INITIALIZE
210 OUT CR1,CW
230 D%=&HFF
240 OUT PB0,D%:OUT PB1,D%
250 OUT PC0,D%:OUT PC1,D%
270 GOSUB *TIMER
280 D%=NOT D% AND &HFF
290 GOTO 240
300 *TIMER
310 FOR T=1 TO 2000:NEXT T
320 RETURN
```

```
/* 8255 TEST(8BITS) */
#includ <stdio.h>
#includ<dos.h>
#define PA 0xd0
#define PB 0xd2
#define PC 0xd4
#define CR 0xd6
#define CW 0x90
void wait(void);
main()
{
  int i,j;
  unsigned char d;
  outportb(CR,CW);
  for(i=0;i<10;i++){
      d = 1;
        for(j=0;j<=8;j++){
        outportb(PB,d);
        outportb(PC,d);
        d = d<< 1;
        wait();
       }
      }
}
void wait()
{
  int  t=0;
  while(--t);
}
```

【例３】ビット０のスイッチがＯＮで
　　全部のＬＥＤを点滅／ＯＦＦで点滅を停止

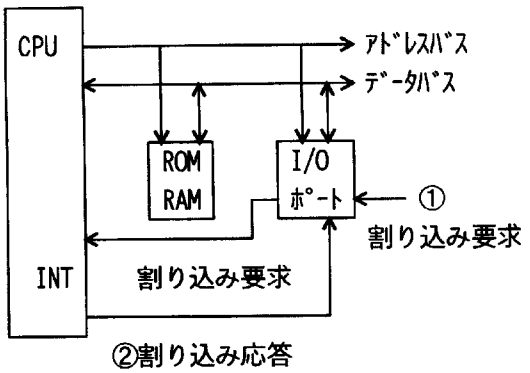
```
100 ' TEST (8BITS)
110 PA=&HD0 '8255 ADDRESS
120 PB=PA+2
130 PC=PA+4
140 CR=PA+6
150 CW=&H90 'CONTROL WORD
160 OUT CR,CW '8255 INITIALIZE
170 D%=&HFF   'DATA SET
180 S=INP(PA)
190 S=S AND 1
200 IF S=0 THEN 180
210 OUT PB,D% 'DATA OUT
220 OUT PC,D%
230 GOSUB *TIMER
240 D%=NOT D% AND &HFF
250 GOTO 180
260 *TIMER
270 FOR T=1 TO 1000
280 NEXT T
290 RETURN
300 END
```

```
100 'TEST (16BITS)
110 PA0=&HD0 '8255AD
120 PA1=PA0+1
130 PB0=PA0+2
140 PB1=PA0+3
150 PC0=PA0+4
160 PC1=PA0+5
170 CR0=PA0+6
180 CR1=PA0+7
190 CW=&H90 'CONTROL WORD
200 OUT CR0,CW '8255INITIALIZE
210 OUT CR1,CW
230 D%=&HFF
240 S=INP(PA0)
250 S=S AND 1
260 IF S=0 THEN 240
270 OUT PB0,D% :OUT PB1,D%
280 OUT PC0,D% :OUT PC1,D%
290 GOSUB *TIMER
300 D%=NOT D% AND &HFF
310 GOTO 240
320 *TIMER
330 FOR T=1 TO 2000 :NEXT T
340 RETURN
350 END
```

```
/* 8255TEST(16BITS)
#includ <stdio.h>
#includ <dos.h>
#define PA 0x80d0
#define PB 0x80d2
#define PC 0x80d4
#dedine CR 0x80d6
#define CW 0x9090
void wait(void);
main()
{
  int i,j;
  unsigned int d=0;
  outport(CR,CW);
  for(i=0;i<10;i++){
    d = 1;
    for(j=0;j<=16;j++){
    outport(PB,d);
    outport(PC,d);
    d = d<<1;
    wait();
    }
   }
}
void wait()
{
  int t=0;
  while(--t);
}
```

５．３．５ 割り込み処理

いつ発生するか分からないデータ処理や緊急性の高いプログラムを割り込みによって実行すると効率の高い処理が可能となる。ｉ８０８６の割り込みは，Ｚ－８０に比べて非常に難しくなっているので，ここでは概要を図１８に示して簡略的に説明を加える。詳しく知りたければｉ８０８６のハードウエアマニュアルを手に入れて調べよ。



①外部よりCPUに割込み要求をする
②CPUは割込みを受け付けた旨の信号を出す
③実行中のプログラムを中断する
④割込みの種類によって決められたｱﾄﾞﾚｽにﾌﾟﾛｸﾞﾗﾑを移す(このｱﾄﾞﾚｽは外部からﾃﾞｰﾀﾊﾞｽを介して送らねばならない場合もある)
⑤そのｱﾄﾞﾚｽの割込みﾌﾟﾛｸﾞﾗﾑを実行する
⑥割込み処理ﾌﾟﾛｸﾞﾗﾑが終わると，実行を中断したﾌﾟﾛｸﾞﾗﾑが実行される。

図１８　割り込み処理の概念図

（１）ｉ８０８６の割込み

割込みには外部割込みと内部割込みがある。内部割込みにはソフトウエア割込みとハードウエア割込みがある。

ハードウエア割込みは，CPUの演算結果による特別な割込みである。ソフトウエア割込みは，Z-80の RST命令に相当するもので一種のCALL命令である。なお，CALLとの違いは，直接処理サブルーチンのアドレスへジャンプするのではなく，0～255(00H～FFH)のタイプ番号を指定し，そのタイプ番号に従ったアドレスにストアされているデータにより，処理ルーチンのアドレスを決定する。

INT 命令は，ジャンプするサブルーチンのアドレスを間接指定している。

BIOS(基本的な入出力装置を動かすプログラム)は，そのパソコン固有のものであり，同じMPUを使っていても異なっている場合が多い。BIOSは，パソコンのメーカが作成して，付属しているのが一般的である。ユーザは，BIOSのルーチンがどのようになっているかを知らなくても，その呼び出しに INT命令を使用する。

割込みの種類			タイプ	優先度
外部割込み	NMI		2	2
	INTR		0～255	3
内部割込み	ソフトウエア割込み	INTn	0～255	1
		INT3	3	1
		INTO	4	1
	ハードウエア割り込み	ﾄﾗｯﾌﾟ割込み	1	4
		ｾﾞﾛによる割込み	0	1

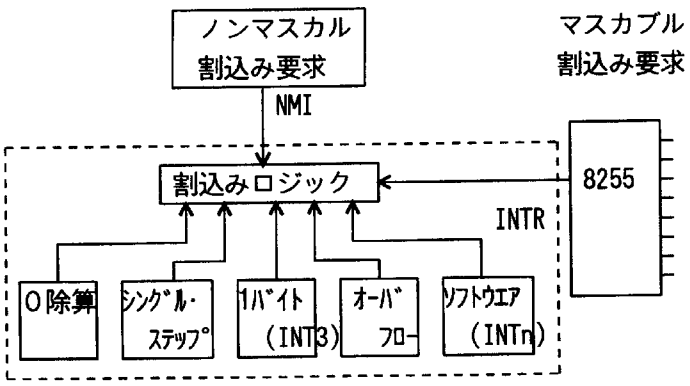


図１９　ｉ８０８６の割り込み

BIOSのプログラムやそのアドレスが異なっていても，同じ動作をするルーチンは，同じ割込み番号を決めておく。このようにするとユーザプログラムでは，機種の違いによるプログラムの違いがなくなり，異機種間の互換性が保たれる。さらに BIOSなるプログラムの作成から開放される。

参考として図２０に割込みベクタと割込み処理アドレスの関係を示す。

(1)割込みアドレスの配置

物理ｱﾄﾞﾚｽ	内容	
00000	IP下位	INT 0Hに
00001	IP上位	対する
00002	CS下位	分岐先
00003	CS上位	
00004	IP下位	INT 1Hに
00005	IP上位	対する
00006	CS下位	分岐先
00007	CS上位	
～	～	
003FC	IP下位	INT 0FFHに
003FD	IP上位	対する
003FE	CS下位	分岐先
003FF	CS上位	

注：INTR　ソフトウエアで割込み禁止ができる。
　　NMI　ソフトウエアで割込み禁止ができない。

(2)割込み受け付け処理

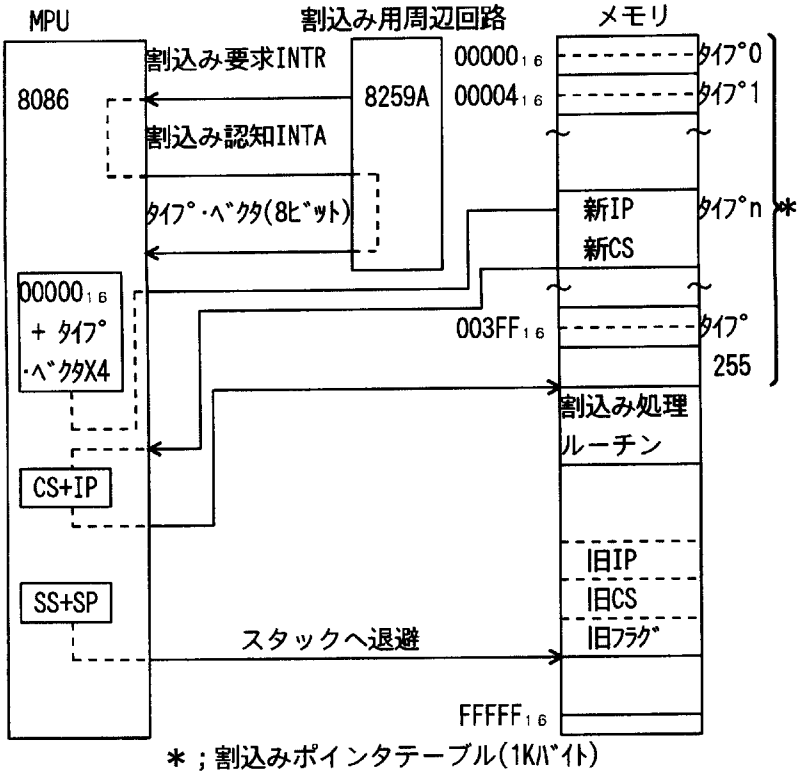


図２０　割込みベクタと割込み処理アドレス

（２）ＰＣ－９８０１の外部割込み

NMI は ＰＣ－９８０１では使われていない。このためユーザはこれを使える。この割込みは，ハードウエアの致命的状態に落ちいたり，電源断等の事故対策などに使われる。

INTR割込みは，CLI(ｸﾘｱ・ｲﾝﾀﾗﾌﾟﾄ)の命令で割込み禁止になり，STI(ｾｯﾄ・ｲﾝﾀﾗﾌﾟﾄ)の命令で割込みが許可となる。INTRに割込み要求があると１バイトの割込みタイプのベクトルを読みにゆく。なお，この割込はベクトルは外部から与える必要がある。このためのハードウエアを用意する必要がある。

８０８６では，一般的に，このタイプベクトルの発生のために ８２５９Ａを使っている。

8259Aは，カスケード接続することで，最大64レベルまでのベクトルを発生できる。

PC-9801では 2個の 8259Aを使っていて，14レベルの外部割込みが使えるようになっている。INTR からの割込みの流れは図２０の(2)のようになる。MPU の INTR 端子に割込み要求を出した後，タイプベクタを読みにくるので，データバスよりタイプベクタを与える。このベクタをｎとすれば，後の流れは，INTn 命令実行と同じになる。 PC-9801VX の 割込みベクタは図２１のようになっている。 ベクタ番号 8H から 17H の16個は，外部割込みに使われている。図２１においてベクタ番号 20H～27Hはシステム予約となっている部分が，MS-DOSのシステムコールに使われている。ユーザプログラムで内部割込みを使う場合，ベクタ番号40H～7FEHを使う。

ｱﾄﾞﾚｽ	ﾍﾞｸﾀ番号	用途	ｱﾄﾞﾚｽ	ﾍﾞｸﾀ番号	用途
0 ～ 3	0	除算エラー	1C ～1F	7	ｲﾝﾀｰﾊﾞﾙ・ﾀｲﾏ
4 ～ 7	1	ｼﾝｸﾞﾙｽﾃｯﾌﾟ	20 ～23	8	ﾀｲﾏ
8 ～ B	2	MNI	24 ～27	9	ｷｰﾎﾞｰﾄﾞ
C ～ F	3	INT3	28 ～2B	A	CRTV
10 ～13	4	ｵｰﾊﾞｰﾌﾛｰ	2C ～2F	B	拡張ﾊﾞｽINT0
14 ～17	5	HARD COPY	30 ～33	C	RS-32C
18 ～1B	6	STOP	34 ～37	D	拡張ﾊﾞｽINT1

図２１-ａ　ＰＣ－９８０１ＶＸ２の割込みベクタ(ﾍﾞｸﾀ番号8～17はﾊｰﾄﾞｳｴｱ割込み)

ｱﾄﾞﾚｽ	ﾍﾞｸﾀ番号	用途	ｱﾄﾞﾚｽ	ﾍﾞｸﾀ番号	用途
38 ～3B	E	拡張ﾊﾞｽINT2	64 ～67	19	RS-232CI/O
3C ～3F	F	ｼｽﾃﾑ予約	68 ～6B	1A	ｶｾｯﾄ/ ﾌﾟﾘﾝﾀI/O
40 ～43	10	ﾌﾟﾘﾝﾀ(ｾﾝﾄﾛﾆｸｽ)	6C ～6F	1B	FD(5″/8″)I/O
44 ～47	11	拡張ﾊﾞｽINT3(5″HD)	70 ～73	1C	ｶﾚﾝﾀﾞ/ｲﾝﾀｰﾊﾞﾙ･ﾀｲﾏI/O
48 ～4B	12	拡張ﾊﾞｽ(ｽﾛｯﾄ#1)INT41	74 ～77	1D	ｼｽﾃﾑ予約
4C ～4F	13	拡張ﾊﾞｽ(ｽﾛｯﾄ#2,8″FD) INT42	78 ～7B	1E	N88-BASIC
			7C ～7F	1F	ｼｽﾃﾑ予約
50 ～53	14	拡ﾊﾞｽINT5	80 ～9F	20～27	ｼｽﾃﾑ予約
54 ～57	15	拡張ﾊﾞｽINT6	A0 ～FF	28～3F	ｼｽﾃﾑ予約
58 ～5B	16	数値ﾃﾞｰﾀ･ﾌﾟﾛｾｯｻ	100 ～1FF	40～7F	ﾕｰｻﾞ用
5C ～5F	17	ﾉｲｽﾞ(ｼｽﾃﾑ予約)	200 ～3C3	80～F0	BASICｼｽﾃﾑ用
60 ～63	18	KB/CRT/ I/O	3C4 ～3FF	F1～FF	ﾕｰｻﾞ用

図２１-ｂ　ＰＣ－９８０１ＶＸ２の割込みベクタ(ﾍﾞｸﾀ番号8～17はﾊｰﾄﾞｳｴｱ割込み)

(３) 割込み処理プログラムの実験

割込み処理プログラムは以下に示す項目に注意を払って作成する。

①割込みベクタに基づく，割込みポインターテーブルへの書込方法。
②割込み処理ルーチンにおけるレジスタの退避とメインルーチンとのデータの授受方法
③外部割込みでは，タイプベクタを発生させるために，割込みコントローラLSI を使用するので，このLSI の使用方法。
④その他にコントローラやI/OポートLSI を使用するのが一般的なので，これらの使用方法。

これがわかれば，何をするプログラムなのか，メインルーチンと割込み処理ルーチンの関係を整理すればよい。

【実験】　ユーザーに解放されている INT40H を使って 8255 に接続した LED を点滅することを考えてみる。

【準備１】システムコールによる割込みポインタテーブルの設定

ＭＳ-ＤＯＳ のシステムコールの中に，割込みベクトルを自動的に設定するファンクションがある。これを使用すると，次のようにプログラムを書くことができる。
このようにすれば，INTnx 4 のアドレスに，INTROUTIN のあるアドレスのIPとCSが設定される。

```
MOV  AX, CS          ┐          DS を CS と同じにする
MOV  DS, AX          ┘
MOV  DX,OFFSET INTROUTIN  DXに割込みアドレスをいれる
MOV  AL,INTn              割込ベクタをALに入れる
MOV  AH,25H               割込ベクトル設定のファンクション番号の書込
INT  21H                  システムコール
```

【準備２】システムコールをしない割込みポインタテーブルの設定

```
MOV  AX, 0
MOV  ES, AX
MOV  BX,INTn *4           テーブル・アドレス
MOV  AX, OFFSET INTROUTIN 割込みアドレス
```

```
MOV  ES:[BX],AX          オフセット(IP)設定
ADD  BX,2
MOV  AX, CS
MOV  ES:[BX],AX          セグメント(CS)設定
```

内部割込み実験プログラム

```
; INTERRUPU TEST
INT_n  EQU  40H
PB     EQU  0D2H
PC     EQU  0D4H
CR     EQU  0D6H
CW     EQU  90H
;
CODE   SEGMENT
       ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE
;
       ORG  100H
START:
      MOV   AX,CS
      MOV   DS,AX
      MOV   DX,OFFSET INTROUTIN
      MOV   AL,INT_n
      MOV   AH,25H
      INT   21H
;
      MOV   AL,CW
      OUT   CR,AL
; MAIN ROUTINE
MAIN:
      MOV   AL,0
M1:   OUT   PB,AL
      OUT   PC,AL
      CALL  WAIT
      INT   INT_n
      JMP   M1
;
WAIT:
      MOV   CX,0
W1:   NOP
      LOOP  W1
      RET
;
; INTERRUPUT ROUTINE
INTROUTIN:
      NOT   AL
      IRET
;
CODE  ENDS
      END   START
```

	割込み信号	優先順位	ベクタ信号	用途
マ	IR0	↑	08	ﾀｲﾏ(8253)
ス	IR1	最	09	ｷｰﾎﾞｰﾄﾞ(8251A)
タ	IR2	高	0A	CRTV(ﾏｽﾀμPD7220 V-SYNC)
｜	IR3	優	0B	拡張ｽﾛｯﾄINT0
P	IR4	先	0C	RS-232C(8251A)
I	IR5		0D	拡張ｽﾛｯﾄINT1(CMT)
C	IR6		0E	拡張ｽﾛｯﾄINT2(ODAﾌﾟﾘﾝﾀ)
	IR7		0F	ｽﾚｰﾌﾞPIC(8259A)
	IR8		10	ｾﾝﾄﾛﾆｸｽ・ﾌﾟﾘﾝﾀ(8255A)
ス	IR9		11	拡張ｽﾛｯﾄINT3(ﾊｰﾄﾞ・ﾃﾞｨｽｸ)
レ	IR10		12	拡張ｽﾛｯﾄINT41(640KBFD)
｜	IR11	最	13	拡張ｽﾛｯﾄINT42(1MBFD)
ブ	IR12	低	14	拡張ｽﾛｯﾄINT5
P	IR13	優	15	拡張ｽﾛｯﾄINT6(ﾏｳｽ)
I	IR14	先	16	NPD(8097)
C	IR15	↓	17	ノイズ(GND)

図２２　ＰＣ－９８０１の外部割込み

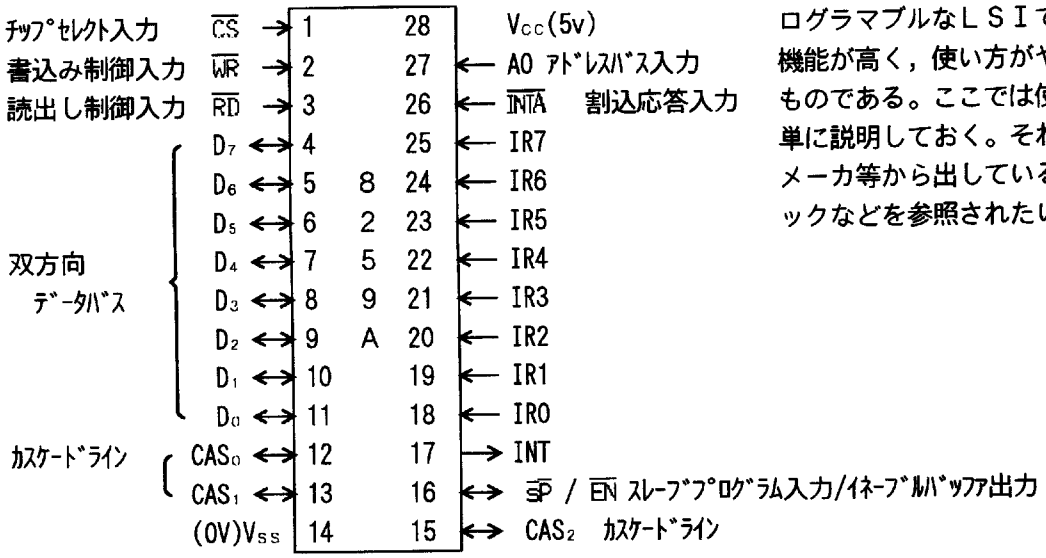
５．３．６　８２５９Ａ

パソコンを計測制御やメカトロニクスに利用するには**外部割込**を駆使するとよいであろう。

ＰＣ－９８０１では，ユーザーに開放されている割込みは図２２に示す通りの２レベルである。いまではＣＭＴ(ｶｾｯﾄﾃｰﾌﾟﾚｺｰﾀﾞ)を使わなくなったので，この部分を利用することができる。またマウスやＯＤＡプリンタ，ハードディスクを使用しなければ，これらを合わせて６レベル分が利用できる。

ＰＣ－９８０１に使用されている割込みコントローラの８２５９Ａは図２３のように接続されている。なお，外部割込みによるプログラムを作るには8259Aの使い方を多少は知っておる必要がある。

8259Aは，8ビットの8080A，8085A，Z-80などのMPUにも使われているプログラマブルなＬＳＩであるが，機能が高く，使い方がややこしいものである。ここでは使い方を簡単に説明しておく。それ以上は，メーカ等から出しているハンドブックなどを参照されたい。



８２５９Ａのピンの定義

8259A は８本の割込み要求入力を持っていて，割込み要求をINT により出力して，ＭＰＵに伝える。ＭＰＵは割込み応答である INTA を返すので，この返事を受けて１バイトの割込みベクタ番号をデータ・バスに出力する（８ビットＭＰＵに対してはCALL命令と，これに続いて２バイトのアドレスを出力できる）。

このベクタ番号はプログラムできる。さらに，８本ある割込み要求入力の優先順位や割込みの禁止であるマスク等自由に設定できる。

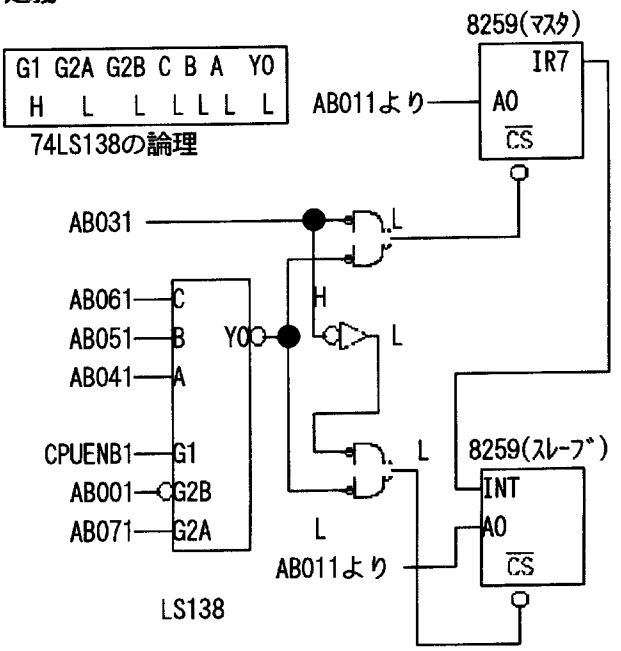


図２３ PC-9801に使われている割込コントローラ 8259Aの接続回路

（１）８２５９Ａのイニシャライズの方法

図２４に8259A のイニシャライズであるＩＣＷコマンドフローを示す。

図２５にＩＣＷのコマンドフォーマットを示す。

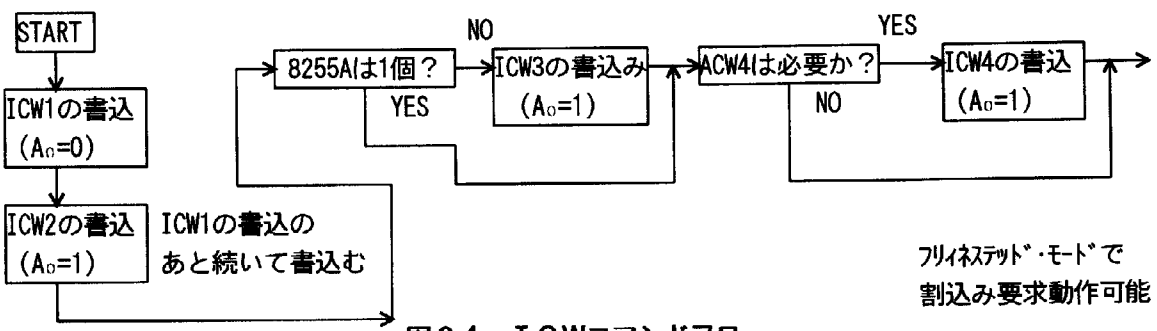


図２４　ＩＣＷコマンドフロー

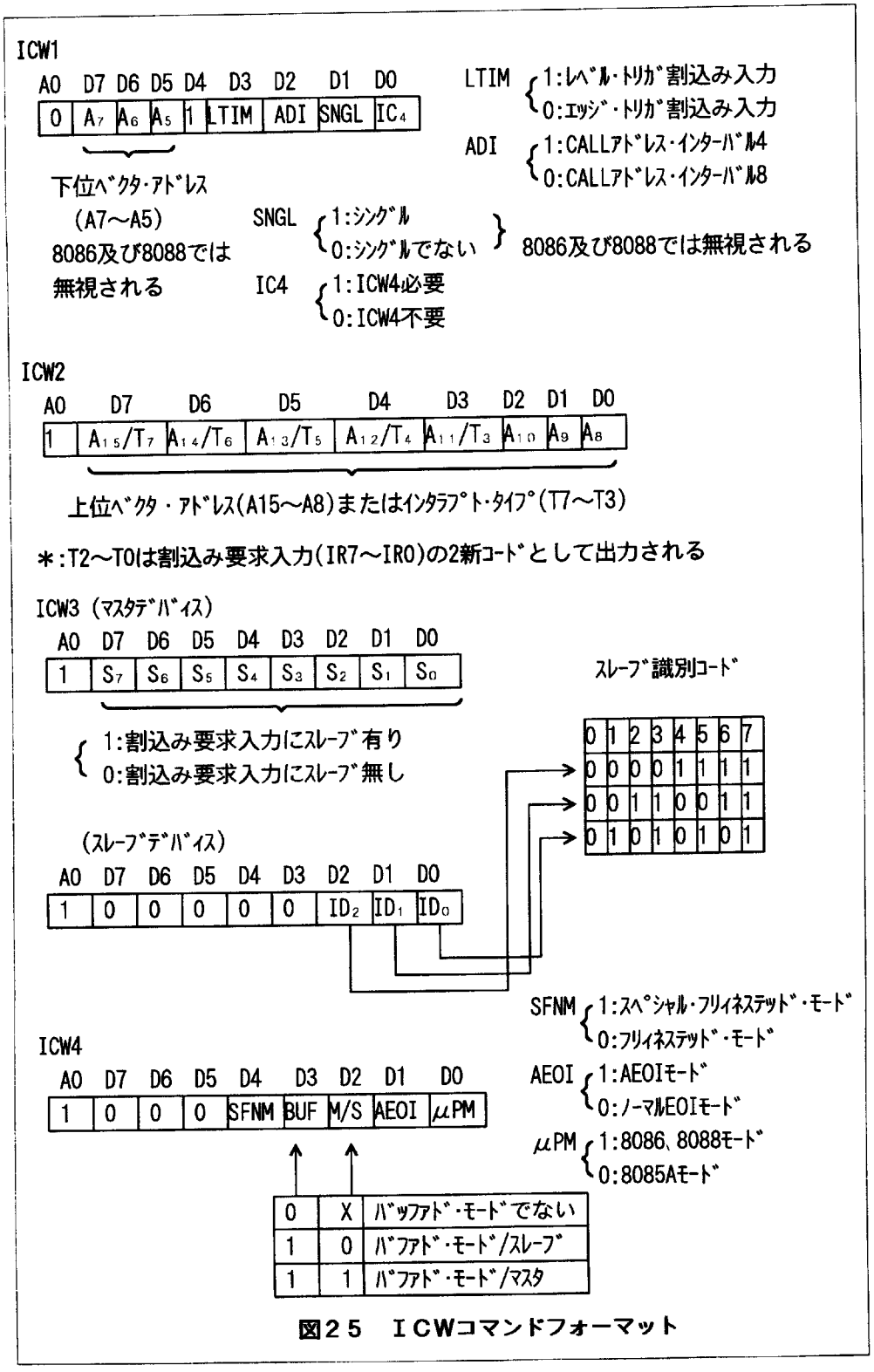


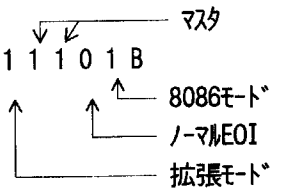
図２５　ＩＣＷコマンドフォーマット

（２）８２５９Ａのイニシャライズプログラム

(1)マスタ８２５９Ａ

```
MOV  AL,00010001B  エッジ・トリガ，スレーブ有り
OUT  00H,AL        ICW4必要(拡張ﾓｰﾄﾞ)
MOV  AL,00001000B  インタラプト・タイプ
```

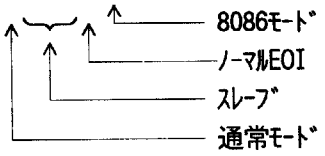
```
OUT  02H,AL                  (08H)
MOV  AL,1 0 0 0 0 0 0 0 B    IR7にスレーブ有り
OUT  02H,AL
```




```
MOV  AL, 0 0 0 1 1 1 0 1 B
OUT  02H,AL
```

(2)スレーブ８２５９Ａ

```
MOV  AL,0 0 0 1 0 0 0 1 B
OUT  08H,AL
MOV  AL,0 0 0 1 0 0 0 0 B    インタラプト・タイプ
OUT  0AH,AL                  (10H)
MOV  AL,0 0 0 0 0 1 1 1 B    スレーブ・デバイス・マスタのIR7 に接続
OUT  0AH,AL
MOV  AL,0 0 0 0 1 0 0 1 B
OUT  0AH,AL
```



（３）ＥＯＩコマンドの送り方

(1)マスタ8259A

```
PUSH  AX
 ～       割込み処理ルーチン
MOV   AL, 20H    EOIコマンド
OUT   00H,AL
POP   AX
IRET
```

(2)スレーブ8259A

```
PUSH  AX
 ～       割込み処理ルーチン
MOV   AL,20H     } スレーブにEOIを送る
OUT   08H,AL     }
MOV   AL,0BH     }
OUT   08H,AL     } スレーブのISRを読む
NOP              }
IN    AL,08H     }
CMP   AL,00H     } ・まだスレーブに割込み処理が残って
JNZ   MORE       }   いる場合は，マスタにEOIを送らない。
MOV   AL,20H     } ・スレーブの割込み処理が全部終わって
OUT   00H,AL     }   いる場合は，マスタにEOIを送る。
MORE: POP  AX
      IRET
```

(4) OCWコマンドフォーマット

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

1:割込みマスクセット
0:割込みマスク・リセット

リセットされるISRビットあるいは優先度最下位となる割込み要求レベル (SL=1 の時有効)

0	1	2	3	4	5	6	7
0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1

OCW2

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	R	SL	EOI	0	0	L2	L1	L0

0	0	1	ノンスペシフィックEOI	EOI
0	1	1	スペシフィックEOI(L2～L0のISRビットをリセット)	EOI
1	0	1	ノンスペシフィックEOIコマンドでそのビットを湯煎順位最下位にする。	オートマチック・ローテーション
1	0	0	AEOIモードにおいてローテート(セット)	オートマチック・ローテーション
0	0	0	AEOIモードにおいてローテート(リセット)	オートマチック・ローテーション
1	1	1	スペシフィックEOIと同時にそのビットを優先順位最下位にする。	スペシフィック・ローテイション
1	1	0	EOIを行うことなく，L2～L0ビットを優先順位最下位にする。	スペシフィック・ローテイション
0	1	0	ノーオペレーション	

OCW3

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	ESSM	SMM	0	I	P	RP	RIS

P 1:ポールコマンド
P 0:ポールコマンドでない。

0	X	ノーオペレーション
1	0	スペシャルマスク・モード・リセット
1	1	スペシャルマスク・モード・セット

0	X	ノーオペレーション
1	0	ステータス読出しレジスタをIRRにセット
1	1	ステータス読出しレジスタをISRにセット

(5) 外部割込み実験

外部割込みの実験をおこなう。この実験に用いるプログラムでは，RUN した後，それぞれのスイッチを ON(Hレベル)にすると割込みが発生させている。プログラムと割込みをわかりやすくするため 8255 の LED を点滅させるようにした。実際には，SW の ON の状態が外部からの割込み要求信号に変わるもので，リミットスイッチやセンサ等からの入力になる。すなわち，この割込み要求によって各種の処理がおこなわれる。

割込みプログラムは，一般のサブルーチンにより手続きが多く，プログラムが複雑になる。

外部割込みをわかりやすくするためにサンプルプログラムを示す。なお，このプログラムで用いているルーチンの構成は以下のようになっている。

1. INT0, INT1, INT2, INT5 の割込みベクトルのセット。
2. マスタ，スレーブそれぞれの 8259A のイニシャライズ。
3. 8255A のイニシャライズ。
4. メインルーチン。
   ポートB と ポートC に接続された LED を，右から左へ順次点灯させる。
5. WAITルーチン
   LED の点灯インターバルタイムを作るためのものである。

6. INT0ルーチン
   SW1 からの割込み処理で，8255 の ﾎﾟｰﾄB と ﾎﾟｰﾄC の LED を全部同時に点滅させる。
7. INT1ルーチン
   SW2 からの割込み処理で，LED を１個おきに点滅させる。
8. INT2ルーチン
   SW3 からの割込み処理で，LED を２個おきに点滅させる。
9. INT5ルーチン
   SW4 からの割込み処理で，ﾎﾟｰﾄB と ﾎﾟｰﾄC の LED を交互に点灯させる。
   ・INT5 はスレーブの 8259A よりの割込みであり，EOIコマンドの送り方が異なっている点に注意すること。

【実験】 外部割込みプログラム

この実験では実験ボードを使用する。なお，C言語は TurboCを使っている。Turbo Cには割込み関数が組み込まれているので，これを使用した。

```
/* external interruput test */
#include <stdio.h>
#include <dos.h>
#define PIC1   0X00
#define PIC2   0x08
#define IMASK  0x08
#define IVEC0  0x0b
#define PA     0xd0
#define PB     0xd2
#define PC     0xd4
#define CR     0xd6
#define CW     ox90
void  wait(void);
void install(void interrupt(*fadder)(),int inum)
{
   setvect(inum,fadder);
}
void interrupt testint()
{
   int c=5;
   unsigned char dd;
   dd=inportb(PA);
   while(c--){
        outportb(PB,dd);
        outportb(PC,dd);
        wait();
        outportb(PB,0);
        outportb(PC,0);
        wait();
        }
   outportb(PIC1,0x20);
  }
void wait()
{
    long int t=40000;
    while(t--);
}
main()
{
    int i;
    unsigned char d;
    outportb(CR,CW);
    install(testint,IVEC0);
    outportb(PIC1+2,inportb(PIC1+2)&~IMASK);
         do{
         d=1;
         for(i=0;i<8;i++){
         outportb(PB,d);
         outportb(PC,d);
         d=d << 1;
         wait();
         }
      }
      while(!kbhit());
    }
```

１６ビット実験ボード用の外部割込みプログラム(８０８６アセンブラ)

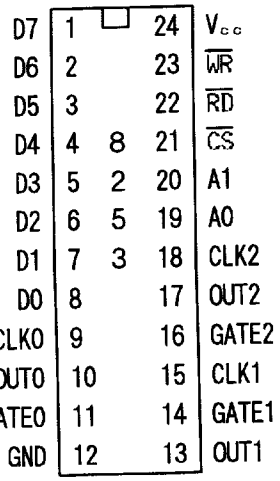
```
;  PC9801 EXTERNAL INTERRUPT TEST
;
        PA      EQU   80D0H  ;8255ADDRESS
        PB      EQU   80D2H
        PC      EQU   80D4H
        CR      EQU   80D6H
CODE    SEGMENT
        ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE
        ORG   100H
START:
        CLI
        MOV   AX,0
        MOV   ES,AX
        MOV   BX,0BH*4     ;INT0 VECTOR
        MOV   ES:WORD PTR [BX],OFFSET SW1
        MOV   ES:2[BX],CS
        MOV   BX,0DH*4     ;INT1 VECTOR
        MOV   ES:WORD PTR  [BX],OFFSET SW2
        MOV   ES:2[BX],CS
        MOV   BX,0EH*4     ;INT2 VECTOR
        MOV   ES:WORD PTR  [BX],OFFSET SW3
        MOV   ES:2[BX],CS
        MOV   BX,14H*4     ;INT5 VECTOR
        MOV   ES:WORD PTR  [BX],OFFSET SW4
        MOV   ES:2[BX],CS
;  8255 INITIALIZE
        MOV   AL,11H       ;MASTER ICW1
        OUT   00H,AL
        MOV   AL,08H       ;ICW2
        OUT   02H,AL
        MOV   AL,80H       ;ICW3
        OUT   02H,AL
        MOV   AL,1DH       ;ICW4
        OUT   02H,AL
        MOV   AL,11H       ;SLAVE ICW1
        OUT   08H,AL
        MOV   AL,10H       ;ICW2
        OUT   0AH,AL
        MOV   AL,07H       ;ICW3
        OUT   0AH,AL
        MOV   AL,09H       ;ICW4
        OUT   0AH,AL
        MOV   AL,00H       ;MASTER OCW1
        OUT   02H,AL
        OUT   0AH,AL       ;SLAVE  OCW1
        MOV   DX,CR
        MOV   AX,9090H     ;8255 INITIALIZE
        OUT   DX,AX
        STI
;
MAIN:
        MOV   AX,1         ;MAIN ROUTINE
M1   :  MOV   DX,PB
        OUT   DX,AX
        MOV   DX,PC
        OUT   DX,AX
        CALL  WAIT
        ROL   AX,01H
        JMP   M1
;
WAIT:
        PUSH  CX
        MOV   CX,0
W1:     NOP
        NOP
        LOOP  W1
        POP   CX
        RET
;  INTERRUPT ROUTINE
SW1:                       ;INT0
        PUSH  AX
        PUSH  CX
        PUSH  DX
        MOV   CX,20
        MOV   AX,0FFFFH
SW11:   MOV   DX,PB
        OUT   DX,AX
        MOV   DX,PC
        OUT   DX,AX
        CALL  WAIT
        NOT   AX
        LOOP  SW11
        MOV   AX,0
        OUT   DX,AX
        POP   DX
        POP   CX
        MOV   AL,20H
        OUT   00H,AL
        POP   AX
        IRET
```

```
;
SW2:                        ;INT1
        PUSH   AX
        PUSH   CX
        PUSH   DX
        MOV    CX,20
        MOV    AX,5555H
        JMP    SW11
;
SW3:                        ;INT2
        PUSH   AX
        PUSH   CX
        PUSH   DX
        MOV    CX,20
        MOV    AX,3333H
        JMP    SW11
;
SW4:                        ;INT5
        PUSH   AX
        PUSH   CX
        PUSH   DX
        MOV    CX,10
        MOV    AX,0FFFFH
SW41:   MOV    DX,PB
        OUT    DX,AX
        NOT    AX
        MOV    DX,PC
        OUT    DX,AX
        CALL   WAIT
        LOOP   SW41
        MOV    AL,20H
        OUT    08H,AL
        MOV    AL,0BH
        OUT    08H,AL
        NOP
        IN     AL,08H
        CMP    AL,0
        JNZ    MORE
        MOV    AL,20H
        OUT    00H,AL
MORE:   POP    DX
        POP    CX
        POP    AX
        IRET
;
CODE    ENDS
        END    START
```

6. 8253(Programmable Interval Timer)

8253 は Programmable Interval Timerである。8253 には，8253(8080A用)と 8253A-5(8085A用)とがあるが機能は同じである。8253A-5 をもとに図２６にブロック・ダイアグラムとピン接続を示して機能を説明する。

ピン名称	意　　味
D7～D0	DATA BUS (8 BIT)
CLK N	COUNTER CLOCK INPUTS
GATE N	COUNTER GATE INPUTS
OUT N	COUNTER OUTPUTS
$\overline{RD}$	READ COUNTER
$\overline{WR}$	WRITE COMMAND OR DATA
$\overline{CS}$	CHIP SELECT
A0,A1	COUNTER SELECT
Vcc	+5 VOLTS
GND	GROUND



(1) ８２５３の基本動作

① CLK に入力されたクロック(パルス信号)によりカウンタの値をデクリメントしていき，ゼロになったらＯＵＴ端子に出力する。

② GATE端子が L レベルになるとカウントが進まなくなる。即ち，カウントをマスクする機能を持っている。

(2) ８２５３の特徴

①互いに独立した３個の１６ビット・ダウンカウンタを内蔵している。

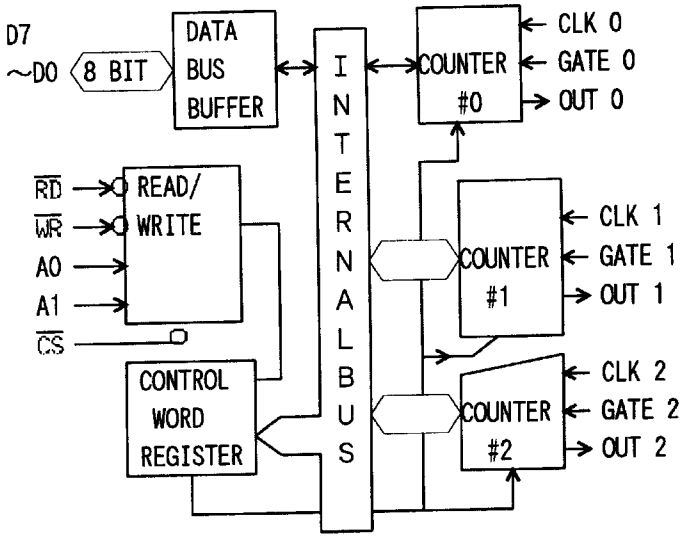


図２６　８２５３のブロックダイアグラムとピンの定義

②カウント可能なクロック周期は，次の通りである。

8253　DC ～ 2 MHz，　8253-5　DC ～ 2 MHz，　8253-2　DC ～ 5 MHz

③６種類のカウンタモードが各カウンタに任意に割当可能である。

④２進または１０進カウンタが可能である。

ＰＣ－９８０１に使われている 8253 の接続を図２７に示す。３つあるカウンタのクロック入力 (CLK_0～CLK_2)はすべて MPU のクロック周波数を４分周した周波数のクロック信号が加わっている。

GATE0～GATE2 はすべて +5V にプルアップされているのでマスクはできない。

カウンタ#0の出力は，割込みコントローラである 8259A の IR0 に接続されている。これは図２２に示したように INT08H の割込みを発生する。このカウンタ#0 を使って任意の時間を作り出すことができる。カウンタ#1 はPC-9801E/F/M ではメモリリフレシュ信号として使われているのでユーザは使えない。これ以後のモデルではスピーカの周波数設定用に使われているのでユーザが利用しても異常は起きない。カウンタ#2 はRS-232C のボーレートを決めるクロック信号用として使われている。

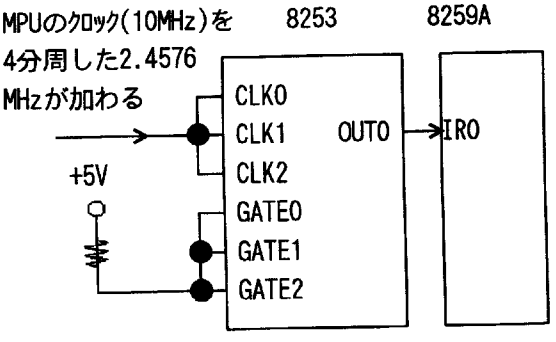


図２７．ＰＣ－９８０１の８２５３の接続

（3）８２５３の動作説明

図２６に従って説明を加える。

表１２　PC-9801の8253のアドレス

８２５３　＃０	７１Ｈ
８２５３　＃１	７３Ｈ
８２５３　＃２	７５Ｈ
コントロールレジスタ	７７Ｈ

(A)入出力信号線

①チップセレクト($\overline{CS}$)：入力

- L レベルにすることによって 8253 をイネーブルにする。
- この端子が H レベルだと読出/書込み操作は行えないが，カウンタとしては動作する。
- この端子は MPU のアドレスバスをデコードして 8253 のアドレスを指定するのに用いる。

②アドレス(A_0, A_1)：入力

- 内部の３つのカウンタまたはコントロールワードレジスタのいずれか１つを選択するのに用いる。
- 通常は MPU のアドレスバス下位２ビットに接続する。

③ライト($\overline{WR}$)：入力

8253 に対する書き込みストローブ信号で，L レベルを加えることによってモード設定やカウンタ値の書き込みができる。

④リード($\overline{RD}$)：入力

8253 に対する読み出しストローブ信号で，L レベルを加えることによって内部のカウンタの値を読み出すことができる。

⑤データバス(D_0～D_7)：入出力

- 8ビットMPU のデータバスに接続される 8ビットの双方行性３ステートのデータバス端子である。
- イニシャライズ時のコントロールの書き込み，カウンタ値の書き込みと読み出しはこのバスを通して MPU とコミュニケーションされる。

⑥クロック(CLK_0～CLK_2)：入力

- カウンタへのクロック入力端子である。
- この信号の立ち下がりで 8253 の内部カウンタに書き込まれた値がダウンカウントされる。

⑦ゲート($GATE_0$～$GATE_2$)：入力

- この端子が H レベルのときダウンカウントが行われる。
- L レベルにするとカウントを停止するマスク機能を持っている。但し，その機能は 8253 のモード選択により異なる。

⑧アウト(OUT_0～OUT_2)：出力

- 8253 に書き込まれたカウンタの値がゼロになると出力される端子である。
- 但し，出力のされ方は(即ち波形)は，モード選択によって異なる。

(B) 内部レジスタの動作

①データバスバッファ

- ８ビットマイコンのシステムデータバスと接続するための双方行性３ステートの８ビット・データバスバッファである。
- モード選定のためのコントロールワード，カウンタへのデータの書き込み，カウンタからのデータの読み出しは，すべてこのバッファを介して行う。

②リード／ライトロジック

- システム側から送られてくるコントロール信号($\overline{RD}$, $\overline{WR}$)を受け取り，8253 の動作全体に対する制御信号を発生する。
- チップセレクト入力($\overline{CS}$)が H レベルのとき(ディセーブル状態)は，コントロール信号は内部の動作に影響を与えない。

③コントロールワードレジスタ

・A0，A1端子がともに 1 のとき選択される。
・データバス(データバスバッファを通して)から情報は，このレジスタにストアされ，各カウンタの動作モード，2進，10進カウントの選択，更に各カウンタへのロード方法の指定を制御する。
・コントロールワードレジスタは，書き込み可能で読み出しはできない。

④カウンタ0～カウンタ2

・各カウンタは，互いに同等で独立した１６ビットのプリセット可能なダウンカウンタである。
・それぞれのクロック入力，ゲート入力，アウト出力端子を持っている。
・各クロックの立ち下がりで，２進か１０進かのいずれかの方式でカウントダウンする。
・カウンタがどのようなモードで動作するか，どのような初期値からカウントを開始するかは，ソフトウェアによって指定できる。
・各カウンタの値は，単に入力命令により随時読み出せるだけでなく，特別な命令（カウンタラッチ命令）を与えることで，各瞬間の値をレジスタにラッチし，安定して読み出せる（リードオンザフライ）機能も持っている。
・表１３に基本機能を示した。表から推察できるようにコントロールワードレジスタを指定して(A_1，A_0ともに H，$\overline{CS}$=L)，$\overline{RD}$信号を与えた場合，データバスは高インピーダンス状態になってしまう。

表１３　８２５３の基本機能

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A1	A0	機　能
0	1	0	0	0	ﾃﾞｰﾀﾊﾞｽ→ｶｳﾝﾀ0
0	1	0	0	1	ﾃﾞｰﾀﾊﾞｽ→ｶｳﾝﾀ1
0	1	0	1	0	ﾃﾞｰﾀﾊﾞｽ→ｶｳﾝﾀ2
0	1	0	1	1	ﾃﾞｰﾀﾊﾞｽ→ｺﾝﾄﾛｰﾙﾜｰﾄﾞﾚｼﾞｽﾀ
0	0	1	0	0	ﾃﾞｰﾀﾊﾞｽ←ｶｳﾝﾀ0
0	0	1	0	1	ﾃﾞｰﾀﾊﾞｽ←ｶｳﾝﾀ1
0	0	1	1	0	ﾃﾞｰﾀﾊﾞｽ←ｶｳﾝﾀ2
0	0	1	1	1	} ﾃﾞｰﾀﾊﾞｽは高ｲﾝﾋﾟｰﾀﾞﾝｽ状態 X: 未定義
1	X	X	X	X	
0	1	1	X	X	

（4）ＭＰＵとの接続

8253 には RESET入力はない。それ以外は動作指令の与え方などは 8255 と同じである。

６．１　８２５３の動作モード

8253 は 6つの動作モードを持っており，イニシャライズによって選択できる。基本的にはカウンタにセットされた値のクロック信号がクロック入力に加わるとカウンタに出力が現れる。出力の現れ方は，モードによって違ってくる。以下に，モードの動作を説明する。

（1）モード0(カウンタ完了割り込み)

モード0は，「ある時間後に MPU に割り込みをかけたり，カウントアップ時に割り込みをかけたりする」場合に利用する。

・モード指定や初期値設定によりカウント出力(OUT端子)は L レベルになる。
・カウンタに初期値をロードするとクロック入力のカウントが開始される。
・ CLK信号の立ち下がりでカウンタはダウンカウントし，0 になると OUT端子が Hレベルになる。

※　このカウントはロード値 n+1 になるので注意すること。

・モード0 では，モード指定や初期値設定が再設定されるまで OUT端子は H のままである。
※　ただし，カウントダウンは続けられる。

図aは，初期値４をセットした場合の例である。
・ゲート入力をＬレベルにするとカウントが中断される。
・カウント中に再ロードすれば最初の１バイトロードに

よりこれまでのカウントが中止され，2バイト目のロードよりカウンタが再スタートする。

(2) モード1 (プログラマブルワンショット)

モード1はゲート入力をトリガ入力とするワンショットマルチバイブレータの機能を持つ。

・ゲートの立ち上がりによって，その次のクロックから始まる，あらかじめ設定されたクロック長のワンショットの L レベル出力を発生する。

図bは，初期値4をセットした場合の例である。

・カウンタ出力がLレベルの間に新たな初期値をロードしても，トリガ入力がなければ，すでに出力されているワンショットパルスのパルス幅は変更されない。

・出力中のワンショットパルス幅に何ら影響を与えることなく，進行中のカウンタ値を読み出すことができる。

・このモードは再トリガ可能である。従って，モード0のように初期値を設定し直さなくても，次のトリガ入力で同じ幅のワンショット出力を出すことができる。

(3) モード2 (レートジェネレータ)

カウンタに初期セットした値nによって，クロック入力n個に 1回の割で 1クロックの間 L レベルがカウンタに出力される(反復して出力する)。

・カウンタが動作中に新たな値をロードすると，現在進行中のカウントによるパルスが出力された後，次の出力から新たにロードされた値でカウンタ出力に現れる。

図cは，最初にレート値として n=4 を与え途中でレート値を n=3 にした場合の例である。

・ L レベルのときはカウンタ出力は強制的に Hレベルになる。

・ゲート入力の立ち上がりによってカウンタは初期値で再スタートする。

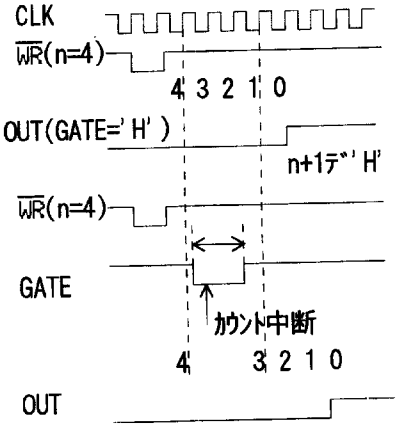
・ゲート入力はハードウエアによるカウンタの外部同期が可能となる

・モード指定後はカウンタにレート値nをロードするまでカウントが開始されず，カウンタ出力は H レベルのままである。

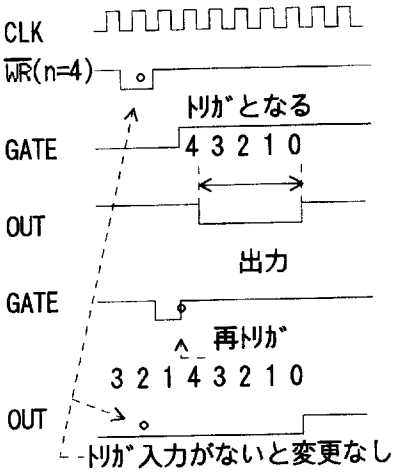
(4) モード3 (矩形波レートジェネレータ)

モード2と同様であるが，設定レート値nの 1/2 のカウント数の矩形波出力を発生する。

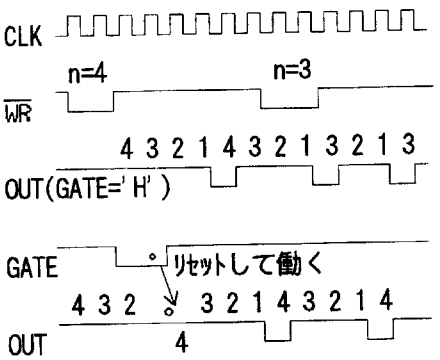
・設定値n：奇数のときは，クロック入力 (n+1)/2 だけ H レベル出力，(n-1)/2 だけ L レベル出力の矩形波となる。



図a モード0(WRの初期値n=4)

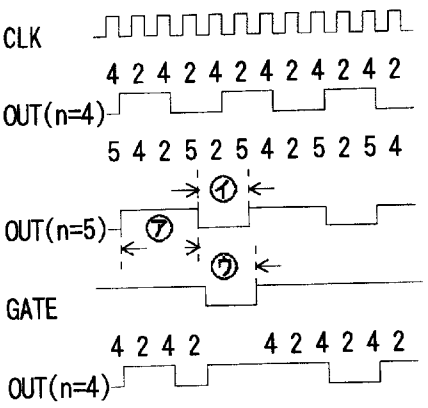


図b モード1(WRの初期値n=4)



図c モード2(WRの初期値n=4)

・カウンタ動作中にカウンタの新たなレート値を再ロードしたときは，次に実行されるカウンタ出力の変化後，次のカウントから再ロード値によるカウンタ出力となる。
・ゲート入力の作用は，モード２のときと同じである。
図dにモード３の動作例を示す。

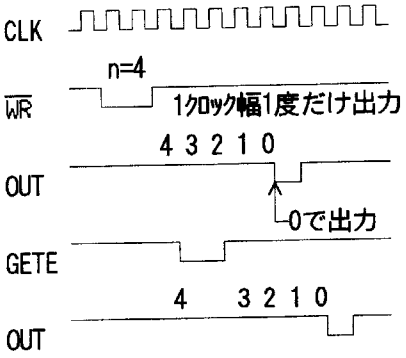


⑦:(5+1)/2 ，①:(5-1)/2
⑦:ｶｳﾝﾀ停止

図d　モード３

(5) モード４（ソフトウエアトリガストローブ）

モードセット後，カウンタ出力はＨレベルであるが，カウンタに値をロードすることでクロック入力のカウントを開始し，カウントアウトで１クロック間隔だけＬパルスを１度だけ出力する。
・モード２はカウンタに設定した値で反復してパルスを出力するが，モード４は反復してパルスを出力しない。
図eに出力パルスの状況を示す。
・出力パルスは，モード２の場合よりも１クロック遅れている。
・カウント中に新たな値がロードされるとカウンタの内容が更新され，新しい値でカウントが継続される。
・ゲート入力をＬレベルにすると，カウントは一時中断され，Ｈレベルで再開される。

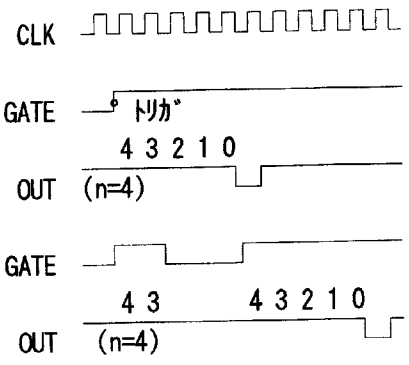


図e　モード４

(6) モード５（ハードウエアトリガストローブ）

・モード１の変形である。
・ゲート入力がトリガ機能を受け持っている。
・ゲート入力の立ち上がりでカウントが開始され，カウントアウトで１度だけ１クロック間隔の Ｌレベルパルスが出力される。
・モード１と同様に，ゲート入力によって再トリガが可能である。
図fにモード５のタイミング例を示す。

なお， 8253 のゲート信号入力の役割を表１４に示しておく。



図f　モード５

6．2　コントロールワードと初期値のロード

8253の各カウンタの動作モード指定は，コントロールレジスタにコントロールワードを書き込むことで行う。
カウンタは 0～2 までの３個あり，それぞれ独立して動作できる。

（注）・ＰＣ－９８０１に内蔵の８２５３は，ゲート入力がすべてプルアップされて各部に引き出されていない。
・クロック入力には４分周された信号が入力されている。

このため８２５３を使う場合はモード０かモード３に限定される。

(1) コントロールワード

8253 のコントロールワードのフォーマットを図１４に示す。

図１４によって作成したコントロールワードをコントロールレジスタに書き込む。その後，それぞれのカウンタアドレスに初期値を書き込む。この初期値すなわち８２５３のカウンタレジスタに書き込むデータは１６ビットであるので，最初に下位８ビットを書き込み，続いて上位８ビットを書き込む。全部のカウンタを使う場合には，合計９回の書き込みが必要である。

コントロールワードは，４つのセクションから成り立っている。

(1)ＳＣセクションはカウンタの選択を行う。

(2)ＲＬセクションはカウンタ中のカウンタの読み出しや初期値設定をどのように行うかを指定する。

・通常は，１１を指定し，下位８ビットに続いて上位８ビットを書き込む。

(3)Mセクションはモード指定を行うビットである。

(4)ＢＣＤセクションは，カウンタをバイナリとするかＢＣＤとするかの選択である。

・バイナリでは，1～65536 までのカウンタとして使える。
０を初期設定したときには最大になる。

・ＢＣＤを選択した場合には，0～9999 までの 4 桁の 10進数を書き込む。

８２５３は，システムリセットや電源投入によって何ら特定の値に初期設定されない。まずコントロールワードを書き込み，次にカウンタにデータを書き込むことによって動作する。なお，３つのカウンタ間の順序については全く自由である。

表１４　８２５３のコントロールワード

D7	D6	D5	D4	D3	D2	D1	D0	←ビット
SC1	SC0	RL1	RL0	M2	M1	M0	BCD	←ビットの役割
SC		RL		M			BCD	

M（モード）

M2	M1	M2	ﾓｰﾄﾞ
0	0	0	ﾓｰﾄﾞ0
0	0	1	ﾓｰﾄﾞ1
x	1	0	ﾓｰﾄﾞ2
x	1	1	ﾓｰﾄﾞ3
1	0	0	ﾓｰﾄﾞ4
1	0	1	ﾓｰﾄﾞ5

ＲＬ（リード／ロード）

RL1	RL0	意　味
0	0	カウンタラッチオペレーション
0	1	下位８ビットのみ読み出し／ロード
1	0	上位８ビットのみ読み出し／ロード
1	1	下位８ビット，次いで上位８ビットの読み出し／ロード

ＳＣ（セレクトカウンタ）

SC1	SC0	意　味
0	0	カウンタ０を選択
0	1	カウンタ１を選択
1	0	カウンタ２を選択
1	1	組み合わせ禁止

BCD

0	ﾊﾞｲﾅﾘ(2進16ﾋﾞｯﾄ)ｶｳﾝﾄ
1	BCD(4桁)ｶｳﾝﾄ

(2) カウンタのモニタ方法

8253のカウンタ中の値を読み出してモニタした場合，特にイベントカウンタとして利用するとき，カウンタの値を調べてその値によって処理内容を変更することがある。

8253のカウンタの読み出し方には２通りの方法がある。

①読み出し動作

モニタしたいカウンタのアドレスを指定して入力命令を実行すればカウンタの値を読み出すことができる。ただし，読み出し中に変化することが考えられるので正しく読み出すにはクロック入力を一時中

断するか，ゲート入力によりカウンタ値の変化を停止する必要がある。

カウンタ値の読み出し方法(読み出されるバイト)は，イニシャライズのコントロールワード D5 とD4 (RLセクション)によって変わってくる。

いま，D5，D4 に 11 を指定してあれば IN命令を 2回実行し，初めの IN で下位 8ビット，次の IN 命令で上位 8ビットが読み出される。

②リードオンザフライ(Read on the fly)動作

これは進行中のカウント動作に何ら影響を与えずに，カウント値を読み出すために用いる方法である。このためにはコントロールワードレジスタにカウンタラッチ(特別なコマンド)を書き込む。

・カウンタラッチとは；
D7，D6 ビットで読み出すカウンタを選択し，D5，D4 ビットを 0，0 とするデータである。

D7	D6	D5	D4	D3	D2	D1	D0
		0	0	×	×	×	×
カウンタの選択		00にすると カウンタラッチの動作		1 又は 0 （どちらでも良い）			

・D3～D0 ビット ： 1 でも 0 でもかまわない。
・D3～D0 ビットの書き込みによってカウンタの各瞬間の値はレジスタにラッチされて安定したカウンタ値を得ることができる。
・カウンタの読み出しには，初期設定で指定した読み出し手順を守る必要がある。

> コントロールワードレジスタにカウンタラッチのコマンドを書き込んでも以前に指定したイニシャライズは影響を受けずそのまま残っている。

③カウンタ２にカウンタラッチ・オペレーションを行うプログラミング

```
MOV  AL, 0000XXXX (00H)    カウンタラッチコマンド
OUT  77H,AL                コントロールレジスタに書き込む
IN   AL,71H                カウンタ＃０の読み出し
IN   AH,71H                カウンタ＃０の上位読み出し，この時点でカウンタラッチから抜ける。
```

・ × は 0 または 1
・ＩＮ命令を２回実行するか，１回ですませるかはモード設定時の D5，D4 によって決まる。

６．３　８２５３の割込み実験用のプログラム

【実験】カウンタ#0 をモード3 て使用し，周期20msecの矩形波を発生させる。この矩形波を用いて8259に割込みをかけて，8255のポートBとポートCに接続したLEDを１秒間隔で交互に点灯させる。

① 8253のクロック信号周波数　10MHz(9.8304MHz)－2.4576MHz
8MHz(7.9872MHz)－1.9968MHz
5MHz(4.9152MHz)－1.2288MHz

10MHzを使うとして，20msecに１回割込みを発生させるには

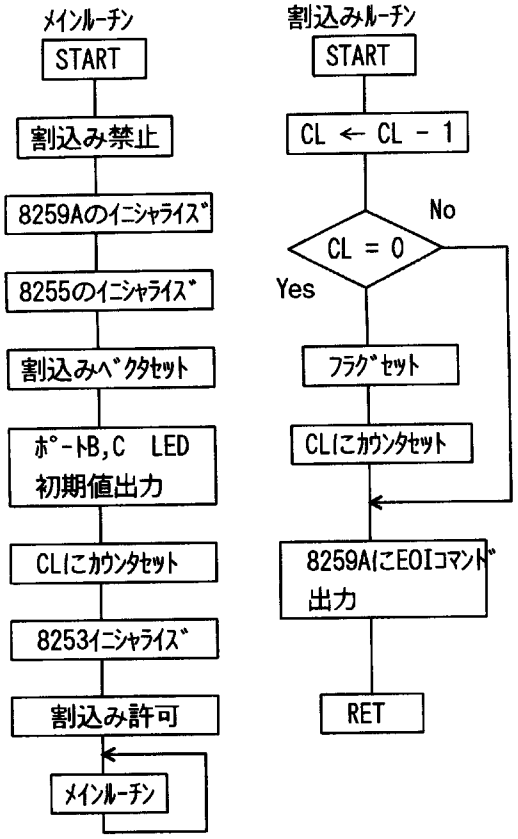
$20 * 10^{-3} = \{ 1/(2.4576 * 10^{6})\} * n$

$n = 20 * 10^{-3} * 2.4576 * 10^{6} = 4.9152 * 10^{4} = C000H$

計算から，8253 に初期値として 49152 をセットすると，クロック信号１周期で１ずつダウンカウトし，20msec後に 0 となり OUT端子に出力が発生し，これが8259への割込み要求となる。

② 8253をモード３にイニシャライズ……①の周期での矩形波を発生するのでモード３で使用する。
③ 20msecを1secにするには，割込み50回で，１度出力するようにプログラムする。

これだけがわかれば，8259A，8255のイニシャライズ，割込みベクタのセット，メインルーチン，8253の割込みルーチンを作成していけばよい。




```
; 8253 INTERRUPT
        PA    EQU  0D0H
        PB    EQU  0D2H
        PC    EQU  0D4H
        CR    EQU  0D6H
        PIT0 EQU  71H
        PITC EQU  77H
CODE    SEGMENT
        ASSUME  CS:CODE,DS:CODE,ES:CODE
        ORG  100H
START:
        CLI
        CALL S8259
        CALL S8255
        CALL SINTV
        MOV  AL,0
        OUT  PB,AL
        NOT  AL
        OUT PC,AL
        MOV CL,50  ;COUNTE
        CALL S8253
        STI
 ;MAIN ROUTINE
 L1:    MOV CH,0   ;FLAG CLEAR
 L2:    OR  CH,CH  ;FLAG CHEAK
        JZ  L2
        OUT PB,AL
        NOT AL
        OUT PC,AL
        JMP L1
S8259:              ;MASTER ONLY
        MOV AL,11H
        OUT 00H,AL
        MOV AL,08H
        OUT 02H,AL
        MOV AL,80H
        OUT 02H,AL
        MOV AL,1DH
        OUT 02H,AL
        RET
S8255:              ;PA IN,PB & PC OUT
        MOV AL,90H
        OUT CR,AL
        RET
SINTV:
        MOV AX,0
        MOV ES,AX
        MOV BX,8*4  ;PIT INTERRUPT VECTER
        MOV ES:WORD PTR[BX],OFFSET PITINT
        MOV ES:2[BX],CS
        RET
S8253:
        PUSH AX
        MOV  AL,36H ;MODE 3
        OUT  PITC,AL
        MOV  AL,0    ;LSB SET
        OUT  PIT0,AL
        MOV  AL,0C0H ;MSB SET
        OUT  PIT0,AL
        POP  AX
        RET
;   PIT INTERRUPT ROUTINE
PITINT:
        DEC  CL
        JNZ  SRET
        MOV  CH,1
        MOV  CL,50
SRET:
        PUSH  AX
        MOV   AL,20H
        OUT   00H,AL
        POP   AX
        IRET
CODE    ENDS
        END   START
```

7．　８２５５の拡張Ｉ／Ｏボードの製作

計算機のインターフェース、特に周辺ＬＳＩに関して理解を深めるには、Ｉ／Ｏ拡張ボードを設計・製作し、それを使って実際に動作させると非常にわかりやすい。このために、この節では８２５５を搭載したパソコン用のＩ／Ｏ拡張ボードの設計・製作をおこなうことにする。

Ｉ／Ｏ拡張ボード基板の設計・製作には、使用するパソコンから引き出されているバスの信号名、信号線ピン数、コネクタの形状などを知る必要がある。これは使用するパソコンのユーザーズマニュアルなど参考にして拡張用Ｉ／Ｏの仕様を調べて、それに合わせて設計する。さらにユーザに開放されているＩ／Ｏアドレスも重要となるのでこれも調べておく必要がある。

なお、製作する場合には、対象とするパソコンを指定する必要がある。そこでＮＥＣのＰＣ－９８０１とＰＣ－８８０１シリーズを取り上げ、具体的な説明を順次加える。

7．1　ＰＣ９８０１／ＰＣ－８８０１の拡張スロットバス

表１６に PC-8801 シリーズの拡張 I/O スロットバスインターフェースを示す。これはPC-8001のI/O ユニット PC-8012 が基本になっている(異なる部分も少しある)。

① コネクタは サイドA(32ﾋﾟﾝ) と サイドB(32ﾋﾟﾝ) のカードエッジになっている。
② Z-80 MPU バスが加工されている部分があるので注意が必要。… $\overline{IOW}$ 、$\overline{IOR}$ 、$\overline{INT}$ n （n:0～7）
③ $\overline{INT}$ n ：割り込みコントローラが組み込まれている。ユーザは別にこの部分のハードウエアを作る必要がない。但し、MPU の $\overline{INT}$ 入力には直接割り込みがかけられないので拡張はできない。
④ユーザに解放されている部分(システムで使用していない部分)は３レベルである。
◇この３つを外部コネクタＣＮ３に引き出して使うようにした。
◇PC-8801 は MPU の割り込み要求の$\overline{INT}$ が サイドB の 5番ピンに引き出されている。
◇8801MK Ⅱ からは $\overline{INT}$ 入力が直接拡張バスに引き出されていない。割り込み以外は共通である。

表１６(a)

PC-8801 1/0 スロットバスのピン				
	バス1,2		バス3,4	
端子番号	信号名 (ｻｲﾄﾞA)	信号名 (ｻｲﾄﾞB)	信号名 (ｻｲﾄﾞA)	信号名 (ｻｲﾄﾞB)
1	GND	GND		
2	GND	GND		
3	+5V	+5V		
4	+5V	+5V		
5	AB0			
6	AB1			
7	AB2	$\overline{MWAIT}$		
8	AB3	$\overline{INT\ 4}$		
9	AB4	$\overline{INT\ 3}$		
10	AB5	$\overline{INT\ 2}$		
11	AB6	$\overline{FDINT\ 1}$		
12	AB7	$\overline{FDINT\ 2}$		
13	AB8	DB0		

表１６(b)

PC-8801 1/0 スロットバスのピン				
	バス1,2		バス3,4	
端子番号	信号名 (ｻｲﾄﾞA)	信号名 (ｻｲﾄﾞB)	信号名 (ｻｲﾄﾞA)	信号名 (ｻｲﾄﾞB)
14	AB9	DB1		
15	AB10	DB2		
16	AB11	DB3		
17	AB12	DB4		
18	AB13	DB5		
19	AB14	DB6		
20	AB15	DB7		
21	$\overline{RD}$	$\overline{MEMR}$		
22	$\overline{WR}$	POWER		
23	$\overline{MREQ}$	$\overline{IOW}$		
24	$\overline{IORQ}$	$\overline{IOR}$		
25	$\overline{MI}$	$\overline{MEMW}$		
26	$\overline{RAS0}$	$\overline{DMATC}$		

＊ﾊﾞｽ 3,4　空白部分は ﾊﾞｽ 1,2 と同じ

表１６(C)

PC-8801 1/0 スロットバスのピン

	バス1,2		バス3,4	
端子番号	信号名(ｻｲﾄﾞA)	信号名(ｻｲﾄﾞB)	信号名(ｻｲﾄﾞA)	信号名(ｻｲﾄﾞB)
27	$\overline{RAS}$ I	$\overline{FDRDY}$		
28	$\overline{RFSH}$	$\overline{DRQ}$ I		$\overline{DRQ}$ 2
29	$\overline{MUX}$	$\overline{DACK}$ I		$\overline{DACK}$2
30	$\overline{WE}$	4CLK		
31	$\overline{ROMKIL}$	$\overline{NMI}$		
32	$\overline{RESET}$	$\overline{WAITRQ}$		
33	SCLK	+12V		
34	CLK	-12V		
35	V1 1A	V1 1A	V3 1A	V3 1A
36	V2	V2	V4	V4

＊ﾊﾞｽ 3,4　空白部分はﾊﾞｽ 1,2 と同じ

表１７(a)

PC-9801 I/O スロットバスのピン

端子番号	信号名(ｻｲﾄﾞA)	信号名(ｻｲﾄﾞB)
1	GND	GND
2	V1	V1
3	V2	V2
4	AB001	DB001
5	AB011	DB011
6	AB021	DB021
7	AB031	DB031
8	AB041	DB041
9	AB051	DB051
10	AB061	DB061

表１７(b)

PC-9801 I/O スロットバスのピン

端子番号	信号名(ｻｲﾄﾞA)	信号名(ｻｲﾄﾞB)
11	GND	GND
12	AB071	DB071
13	AB081	DB081
14	AB091	DB091
15	AB101	DB101
16	AB111	DB111
17	AB121	DB121
18	AB131	DB131
19	AB141	DB141
20	AB151	DB151

表１７(C)

PC-9801 I/O スロットバスのピン

端子番号	信号名(ｻｲﾄﾞA)	信号名(ｻｲﾄﾞB)
21	GND	GND
22	AB161	+12V
23	AB171	+12V
24	AB181	IR31
25	AB191	IR51
26	AB201	IR61
27	AB211	IR91
28	AB221	IR101/IR111(INT4
29	AB231	IR121 (INT5
30	INT0	IR131 (INT6

表１７(d)

PC-9801 I/O スロットバスのピン

端子番号	信号名(ｻｲﾄﾞA)	信号名(ｻｲﾄﾞB)
31	GND	GND
32	IOCHK0	-12V
33	IOR0	-12V
34	IOW0	RESET0
35	MRC0	DACK00
36	MWC0	DACK30/DACK20
37	S00(INTA0	DRQ00
38	S10(NOWAIT0	RQ30/DRQ20
39	S20(SALE1	WORD0
40	LOCK0(MACS0	CPKILL0(EXHRQ10

表１７(e)

PC-9801 I/O スロットバスのピン

端子番号	信号名(ｻｲﾄﾞA)	信号名(ｻｲﾄﾞB)
41	GND	GND
42	CPUENB10	RQGT0
43	RFSH 0	DMATC0
44	BHE 0	NMI0
45	IORDY 1	MWE0
46	SCLK 1	HLDA00(EXHLA20
47	S18CLK 1	HRQ00 (EXHRQ20
48	POWER 0	DMAHLD0(SBUSRQ1
49	+5V	+5V
50	+5V	+5V

(注)　PC-9801VXの場合においてPC-9801VMと異なるところを()の中に示す。

表１７(ａ)～(ｅ)に，PC-9801 の拡張スロットバスのピン配置を示す。

表１８に，PC-9801VMからRAのI/Oポートアドレスを示す。$(XXD0)_{16}$～$(XnDF)_{16}$，$(xnE0)_{16}$～$(xnEF)_{16}$は，ユーザに開放されている I/O アドレスである。これらを除いた全てはシステムで使用済みか予約されている。00E0～00ECは，Ｎ８８－ＢＡＳＩＣのＩＮＰ文使用時，キーボードのスキャンコードとして使用されるので注意が必要である。 X はできるだけデコードすること。新機種では，その一部がシステムに使われているものもある。マニュアルを参照すること。現在のところ 16ビットアドレスの下位の &HD0，　&HD2，&HD4，&HD6，&HD8 を使うのが無難である。

表18　PC－9801VMからRA，RXまでのI／Oポートアドレス

（ノーマルモード）

	ポートアドレス																装置名	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	X	X	X	X	0	0	X	X	0	0	0	0	0	X	A_0	0	割込みｺﾝﾄﾛｰﾗ(ﾏｽﾀ)	μPD71059C
	X	X	X	X	0	0	X	X	0	0	0	0	1	X	A_0	0	割込みｺﾝﾄﾛｰﾗ(ｽﾚｲﾌﾞ)	μPD71059C
	X	X	X	X	0	0	X	X	0	0	0	A_3	A_2	A_1	A_0	1	DMAｺﾝﾄﾛｰﾗ	μPD8237A-5
	X	X	X	X	0	0	X	X	0	0	1	0	X	X	X	0	ｶﾚﾝﾀﾞ時計	μPD4990A
	X	X	X	X	0	0	X	X	0	0	1	0	X	A_1	A_0	1	DMAﾊﾞﾝｸ	8237
	X	X	X	X	0	0	X	X	0	0	1	1	X	X	A_0	0	RS-232Cｲﾝﾀｰﾌｪｰｽ	μPD8251A
	X	X	X	X	0	0	X	X	0	0	1	1	X	A_1	A_0	1	ｼｽﾃﾑﾎﾟｰﾄ	μPD8255A-5
	X	X	X	X	0	1	X	X	0	0	1	1	1	A_1	A_0	1	予約	
	X	X	X	X	0	0	X	X	0	1	0	0	X	A_1	A_0	0	ﾌﾟﾘﾝﾀｲﾝﾀｰﾌｪｰｽ(ｾﾝﾄﾛ)	μPD8255A-5
	X	X	X	X	0	0	X	X	0	1	0	0	X	X	A_0	1	ｷｰﾎﾞｰﾄﾞｲﾝﾀｰﾌｪｰｽ	μPD8251A
	X	X	X	X	0	0	X	X	0	1	0	1	X	X	A_0	0	NMIｺﾝﾄﾛｰﾙ	
	X	X	X	X	0	0	X	X	0	1	1	0	A_2	A_1	A_0	0	CRTｺﾝﾄﾛｰﾗ(ﾃｷｽﾄ)	μPD7220A
	X	X	X	X	0	0	X	X	0	1	1	0	X	X	X	1	予約	
	X	X	X	X	0	0	X	X	0	1	1	1	A_2	A_1	A_0	0	CRTｺﾝﾄﾛｰﾗ	
	X	X	X	X	0	0	X	X	0	1	1	1	X	A_1	A_0	1	ﾀｲﾏｺﾝﾄﾛｰﾗ	μPD8253-5
	X	X	X	X	X	X	X	X	1	0	0	0	0	0	A_0	0	5ｲﾝﾁ固定ﾃﾞｨｽｸｲﾝﾀｰﾌｪｰｽ	
	X	X	X	X	0	0	X	X	1	0	0	0	0	1	A_0	0		
	X	X	X	X	X	X	X	1	1	0	0	0	1	A_1	A_0	0	ｻｳﾝﾄﾞﾎﾞｰﾄﾞ	YM2203
	X	X	X	X	0	0	X	X	1	0	0	0	1	A_1	A_0	1	BRANCH	4670
	X	X	X	X	X	X	X	X	1	0	0	0	1	A_1	A_0	1	ﾈｯﾄﾜｰｸｲﾝﾀｰﾌｪｰｽﾎﾞｰﾄﾞ	
	X	X	X	X	X	X	X	X	1	0	0	1	X	A_1	A_0	0	1MBﾌﾛｯﾋﾟｰﾃﾞｽｸｺﾝﾄﾛｰﾗ	μPD765A
	X	X	X	X	X	X	X	X	1	0	0	1	0	A_1	A_0	1	CMTｲﾝﾀｰﾌｪｰｽ	μPD8251A
	X	X	X	X	X	X	X	X	1	0	0	1	1	0	A_0	1	GP-IBｽｲｯﾁ	7210
	X	X	X	X	X	X	X	X	1	0	0	1	1	1	0	1	予約	
	X	X	X	X	0	0	0	0	1	0	0	1	1	1	1	1	68000ﾎﾞｰﾄﾞ	
	X	X	X	X	0	A_3	X	X	1	0	1	0	A_2	A_1	A_0	0	CRTｺﾝﾄﾛｰﾗ(ｸﾞﾗﾌ)	μPD7220A
	X	X	X	X	0	1	X	X	1	0	1	0	A_2	A_1	A_0	0	EGC拡張ｱﾄﾞﾚｽ	
	X	X	X	X	0	0	X	X	1	0	1	0	A_2	A_1	A_0	1	文字ﾊﾟﾀｰﾝROM	
	X	X	X	X	X	X	X	X	1	0	1	1	A3	A_2	A_1	A_0	通信制御ｱﾀﾞﾌﾟﾀ	7201
	X	X	X	X	X	X	X	X	1	0	1	1	A3	A_2	A_1	A_0	RS-232C拡張ｲﾝﾀｰﾌｪｰｽ	8251
	X	X	X	X	0	0	X	X	1	0	1	1	1	1	1	0	1MB/640KB切換ｲﾝﾀｰﾌｪｰｽ	
	X	X	X	X	X	X	X	X	1	1	0	0	1	A_1	A_0	0	640KBﾌﾛｯﾋﾟｰﾃﾞｨｽｸｺﾝﾄﾛｰﾗ	μPD765A
	X	X	X	X	X	X	X	X	1	1	0	0	A_2	A_1	A_0	1	GP-IB	μPD7210
00D0	X	X	X	X	X	X	X	X	1	1	0	1	X	X	X	0	未使用	
	0	1	1	1	1	1	1	1	1	1	0	1	1	A_1	A_0	1	ﾏｳｽｲﾝﾀｰﾌｪｽ	μPD8255A-5
00ED	X	X	X	X	X	X	X	X	1	1	1	0	1	1	X	1	未使用	
	0	0	1	1	1	1	1	1	1	1	0	1	1	A_0	1	1	ﾀｲﾏｺﾝﾄﾛｰﾗ	μPD8253-5
	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	ﾏｳｽ割込み間隔時間設定	
	X	X	X	X	0	0	X	X	1	1	1	1	0	A_1	A_0	0	CPU	
00F1	X	X	X	X	X	X	X	X	1	1	1	1	0	X	X	1	未使用	
	X	X	X	X	0	0	X	X	1	1	1	1	1	A_2	A_1	A_0	数値演算ﾌﾟﾛｾｯｻNDP	80287

7．2　周辺ＩＣをＭＰＵに接続する方法

図２８に 8255 を Z-80 MPUに接続する例を示す。基本的には 8255 と同名の信号どうしを接続すればよい。 8255 は Z-80 MPUのファミリではないので一部に異なるところがある。

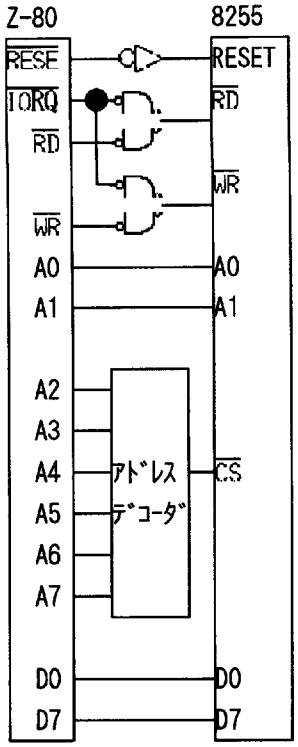


図２８ Z80 MPUと8255の接続

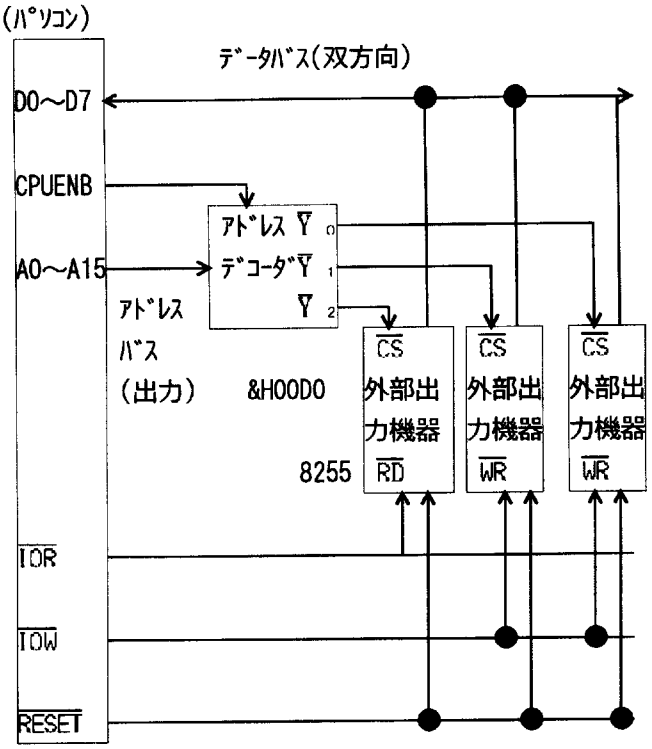


図２９　パソコンに周辺ＩＣを接続する方法

7．3　パソコンの拡張Ｉ／Ｏスロットに接続

パソコンの拡張I/Oスロットバスに接続するにはユーザに開放されている I/Oアドレスを考慮して接続するればよい(図２９，表１８）。

PC-9801拡張スロットバスと 8255を結ぶ。結線においては，AC特性とDC特性および8255のアドレスに関して考慮する必要がある。

PC-9801の電源容量（全スロット合計）	
DC	A
+ 5V	2.5
+12V	0.3
-12V	0.35

①ＡＣ特性について

AC特性は MPUのスピードと8255のスピードを合わせるのに必要であるが専門的な知識が必要なので省略する。

②ＤＣ特性に関して

DC特性に関しては I/Oボードからパソコン本体をドライブするときの最小出力電流の値はＬレベルで12mA，Ｈレベルで 2.5mA，となっている。8255のデータバスの出力電流はＬレベルで 2.5mA，Ｈレベルで 0.4mAしかない。一般的には 8255を直接に接続せず，バスドライバを使用する。しかし，図２８のように8255を１個程度搭載するのであれば電流はそれ程流れないのでバスバッファを省略しても十分動作する。

③リセットの接続方法

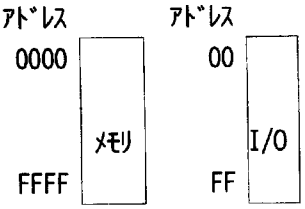
MPUからのリセット出力は負論理(Ｌアクティブ)であり，8255のリセット入力はＨアクティブであるのでインバータを介して 8255 のリセット入力に接続する。

④アドレスデコード回路

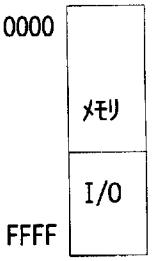
8255のアドレスを決めるアドレスデコード回路の設計に関して，I/Oボードのアドレス指定にはメモリマップドI/O方式とI/Oマップド I/O方式がある。 80系の MPU では一般にI/Oマップド I/O方式を採用している。8086MPU(80286MPU)では，I/Oのアドレス指定には 8ビットまたは16ビットを用いている。

◇ 8ビットMPUを例にして簡単な図解で説明する。

I／Oマップド I／O方式



メモリマップド I／O方式



メモリとI／Oを同じアドレス内に割り付ける。I／Oも16ビットでアドレスされる。メモリとのアクセス命令がすべて使える。

・メモリとI／Oポートを分ける。

メモリ：16ビットでアドレス ，I／O： 8ビットでアドレス

・I／Oとのアクセスは IN，OUT命令で行う。

8086MPUでは，アドレスバスは 20ビット，メモリは 20ビットでアドレス，I／Oは16ビットまたは8ビットでアドレス。

I/Oを16ビットでアドレスする場合

```
IN  Acc, DX
OUT DX, Acc
```

DX：I/Oﾎﾟｰﾄのｱﾄﾞﾚｽを入れておく。

I/Oを8ビットでアドレスする場合

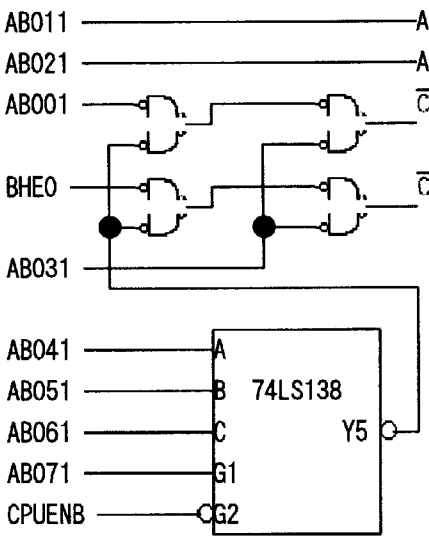
```
IN Acc, Port
OUT port, Acc
```

Acc: AX または AL， Port:8ビット固定ｱﾄﾞﾚｽ

端子番号	A42	A12	A10	A9	A8	A7	A6	A5	A4		
MPUﾊﾞｽ名	CPUENB	AB071	AB061	AB051	AB041	AB031	AB021	AB011	AB001	8255ﾎﾟｰﾄ	ｱﾄﾞﾚｽ
74LS138	G2	G1	C	B	A		A1	A0			
	(74LS138のY5に0が出力)						(8255)				
ﾊﾞｽ信号	0	1	1	0	1	0	0	0	0	ﾎﾟｰﾄA	D0H
出力論理	0	1	1	0	1	0	0	1	0	ﾎﾟｰﾄB	D2H
	0	1	1	0	1	0	1	0	0	ﾎﾟｰﾄC	D4H
	0	1	1	0	1	0	1	1	0	ｺﾏﾝﾄﾞﾚｼﾞｽﾀ	D6H

(注) AB001を "1" とすると 8255 のチップセレクト$\overline{CS}$は "1" になりアクセスされない。

◇１６ビットＩ／Ｏアドレスデコーダ



Y5の出力	AB 031	AB 021	AB 011	AB 001	BHE0	8255ｱﾄﾞﾚｽ (ﾜｰﾄﾞｱｸｾｽ)
1 1 0 1	0	0	0	0	0	D0H
1 1 0 1	0	0	1	0	0	D2H
1 1 0 1	0	1	0	0	0	D4H
1 1 0 1	0	1	1	0	0	D6H

AB001 = 1 のとき，偶数ｱﾄﾞﾚｽの8255は非ｱｸｾｽ

BHE0 = 1 のとき，奇数ｱﾄﾞﾚｽの8255は非ｱｸｾｽ

アドレスデコードの出力を 8255の$\overline{CS}$端子に加える(接続)。

$\overline{RD}$, $\overline{WR}$の端子は IOR0 と IOW0 からの信号を加える(接続)。

7．3．1　PC－9801拡張I／Oボード回路（8ビットデコード）

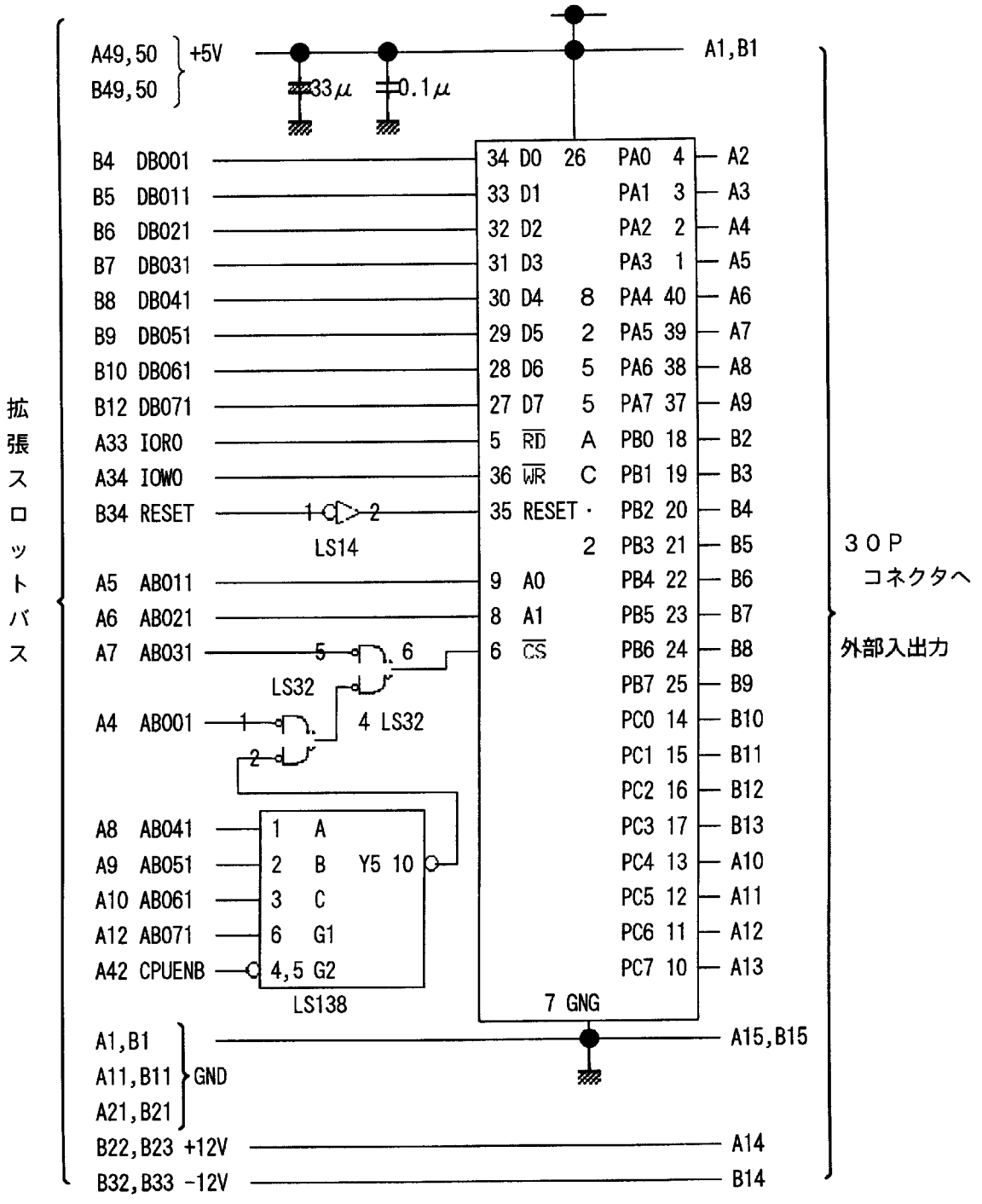


図30a　PC－9801用8255ボード（1／2）

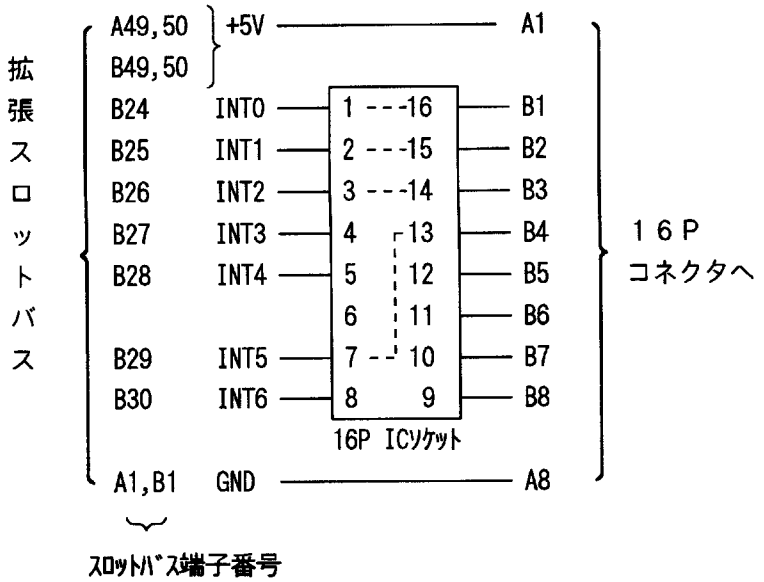


図30b　PC－9801用8255ボード（2／2）

8255　I／Oボードの部品表

プリント基板	サンハヤト	MCC-193	1枚	LSI	8255AC-2	1個
コネクタ	YAMAICHI	FAP30-07#2	1個	TTL	74LS138	1個
	YAMAICHI	FAP16-07#2	1個		74LS14	1個
ICソケット	40P		1個		74LS32	1個
	16P		3個	ケミコン	3.3μF/10V	1個
	14P		2個	セラミックコンデンサ	0.1μF	1個

錫メッキ線　0.5φ，　ビニール線(耐熱)　PVC　7/0,12　外径0.8mm，　はんだ　1φ　60%

接続ケーブル:30芯スダレ(30Pコネクタ付フラットケーブル)　50cm

接続ケーブル:16芯スダレ(16Pコネクタ付フラットケーブル)　50cm

配線の順序　① ＧＮＤまわりの配線，　② ＋５Ｖ(Vcc)まわり配線
③ 入力側の配線　，　④ 出力側の配線

【注意】・ＴＴＬ　ＩＣは，一般に回路図上ではＶccとＧＮＤ線は省略されている。しかし，実際に回路製作するに際にはＴＴＬのＶcc，ＧＮＤ線の配線を行わなければならない。

・配線が終了したら，誤配線，ランド間の短絡，未配線がないか，テスターのΩ計を使って導通テストを行う。

・ＶccとＧＮＤ間の導通を調べる。導通が無限大になっていればＯＫである。

7．3．2　PC－9801拡張Ｉ／Ｏボード回路（１６ビットデコード）

16ビットデータを入出力できるＩ／Ｏボードの作り方を説明し，図３３に回路例を示す。なお，16ビットデータの入出力をおこなうようにするため 8255を 2個使用した。

（１）アドレスデコーダ

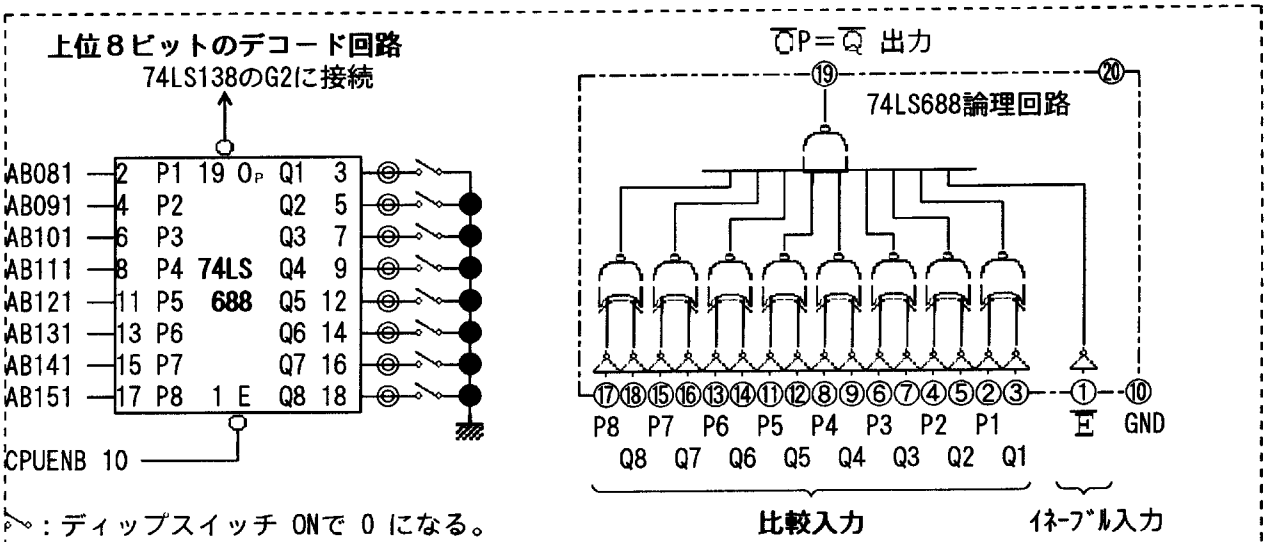
アドレスが 16ビットであってもアドレスを 8ビットでデコードしてよい。8086のアセンブリ言語には，アドレスを 8ビットで指定する命令と，16ビットで指定する命令があるが，Ｎ88－日本語ＢＡＳＩＣでは 8ビットで指定する命令しか用意されていない。このため上位 8ビットは自由にアドレスを設定できるようにする。このため，74LS688(エクスクルシブ・オア)と 8ビットのディップスイッチの組み合わせて，アドレスバスの上位 8ビットの任意のアドレスをデコードできるようにした。このデコードの出力を下位アドレスデコーダである 74LS138 の G2端子に入力する。

図３１のディップスイッチをすべて ONにすると Q1 ～ Q8 の入力が Ｌレベルになる。 PC9801の拡張I/Oの AB081 ～ AB151 がすべてＯのとき P1～P8 がＬレベルとなり，エクスクルシブ・オアの動作を比較するビットが同じ場合(1と1，0と0)には出力はＯとなり，74LS138はアクティブになる。

8086MPUの命令で I/Oのアドレスを 8ビットで指定する命令を使用した場合，上位 8ビットの出力はすべてＯとなるので 8ビットで指定する命令がそのまま使える。

下位 8ビットと 2個の8255の $\overline{CS}$ 信号を作り出す回路は，既に７．３節の④において図解をまじえて説明したものと同じものでよい。 2個の 8255 は，偶数アドレスと奇数アドレスに配置する。8255のアドレスは，図３１中に記したようになる(参考：メモリの場合も同様でる)。

16ビットデータを扱うには(ワード・アクセス)，I/Oポートの基底アドレスを偶数にする必要がある。奇数からおこなうと，16ビットMPUと言えども 2回のアクセスが必要となる。



：ディップスイッチ ONで 0 になる。

◎：1/4W 3.3KΩの固定抵抗でプルアップ(V_{cc}に接続)。

８２５５　ｱﾄﾞﾚｽ	
ﾎﾟｰﾄA(下位)	XXD0H
ﾎﾟｰﾄA(上位)	XXD1H
ﾎﾟｰﾄB(下位)	XXD2H
ﾎﾟｰﾄB(上位)	XXD3H
ﾎﾟｰﾄC(下位)	XXD4H
ﾎﾟｰﾄC(上位)	XXD5H
CWR　(下位)	XXD6H
CWR　(上位)	XXD7H

XXは上位8ﾋﾞｯﾄでDIPｽｲｯﾁで自由に選択する

回路動作

・74LS688の論理回路によりP1～P8とQ1～Q8のそれぞれのレベルが一致すると，エクシクルシブ・オアのゲート出力が "H" になる。

・よって，Q1～Q8に印加されたディップスイッチによる論理と，P1～P8 にに加えられるアドレスバスの論理が一致し，かつ，イネーブル入力(CPUENB)が "L" のとき，74LS388 の出力Ｏｐは "0" になる。

・この信号により下位 8ビットのデコード用 74LS138 がアクティブになる。

・ディップスイッチの ON/OFF の組合わせによって 256 通りの選択が可能。

図３１　上位８ビットのデコード方法

（２）アドレス，コントロールバスバッファ(７４ＬＳ２４４)

・リセット信号：PC9801の拡張I/Oバスから負論理で出ている。8255の RESETはＨでアクティブになるので PC-9801の拡張I/Oバスの RESET 端子と 8255のRESET 端子の間にインバータを入れて接続する。

・アドレスバスのビット0(AB011)と1(AB021)およびIOR0とIOW0 は 74LS244 のバスバッファを介して 8255 の A0とA1，$\overline{RD}$と$\overline{WR}$に接続する(バッファなしでも動作可能)。

・74LS244 にはゲートコントロール入力があるので 1ピンと 19ピンをＬレベルにしないとハイインピーダンス状態になって動作しなくなるので　1ピンを GND に接続する。

（３）データバスバッファ（７４ＬＳ２４５）

・8255 のドライブ能力は，拡張スロットバスをドライブするには不十分である。このためデータバスには双方向性のバスドライバを入れる必要がある(ドライバなしでも動作は可能)。

・データバスバッファのドライバとして 74LS245を用いる。 74LS245 は方向制御用の DIR 端子とゲート制御の $\overline{G}$(または $\overline{OC}$)端子を持っている(４．２の図６と８参照)。 DIR 端子がＨレベルでＡ端子からＢ端子のゲートが ON し，Ｌレベルでの端子からＡ端子のゲートが ON する。よって，IORO の信号を DIR に加えようにする。$\overline{G}$端子はＨレベルでハイインピーダンスとなる(74LS244の動作と同様である)。データバスが衝突しないようにするため 8255 のアドレスが出力されている時だけＬレベルの信号を加える。

（４）８２５５の各ポート（ペリフェラル部）

・8255 には 8ビットの ﾎﾟｰﾄA､ﾎﾟｰﾄB､ﾎﾟｰﾄCがある。２基搭載するので 合計 48ビットとなる。+5V と GNDを引き出すので，最小で50ピンのコネクタが必要である。8255 の使い勝手をよくするために各ポートに GND を取る。このため 60ピンのコネクタを用いた。

・対雑音特性の向上のために全ビットを 33KΩの抵抗で プルアップする。

（５）１６ビットＩ／Ｏボードの部品とＩ／Ｏボード部品配置図

部品名	数量	部品名	数量	部品名	数量
基板　ｻﾝﾊﾔﾄMCC-156	1	LSI　82C55AP-2	2	抵抗ｱﾚｲ　33KΩX４	4
ｺﾈｸﾀ　YAMAICHI FAP16-07 #2	1	TTL　74LS688	1	抵抗ｱﾚｲ　3.3KΩX４	4
〃　ﾋﾛｾ 3BB･60PA-DS	1	〃　74LS138	1	抵抗　1/4W　33KΩ	6
ICｿｹｯﾄ　40P	2	〃　74LS245	2	ｾﾗﾐｯｸｺﾝﾃﾞﾝｻ　0.1μF	5
〃　20P	4	〃　74LS244	1	電解ｺﾝﾃﾞﾝｻ　33μF/10V	1
〃　16P	4	〃　74LS04	1	8bit DIP SW	1
〃　14P	2	抵抗ｱﾚｲ　33KΩX８	7	配線材　0.5φ錫メッキ線	
配線材　PVC 7/0.12 外径0.8mm　(黒，白，赤，青，黄，緑，紫など)					
接続ｹｰﾌﾞﾙ　60芯ｽﾀﾞﾚ(両端に60ピンのｺﾈｸﾀ付)　50cm					

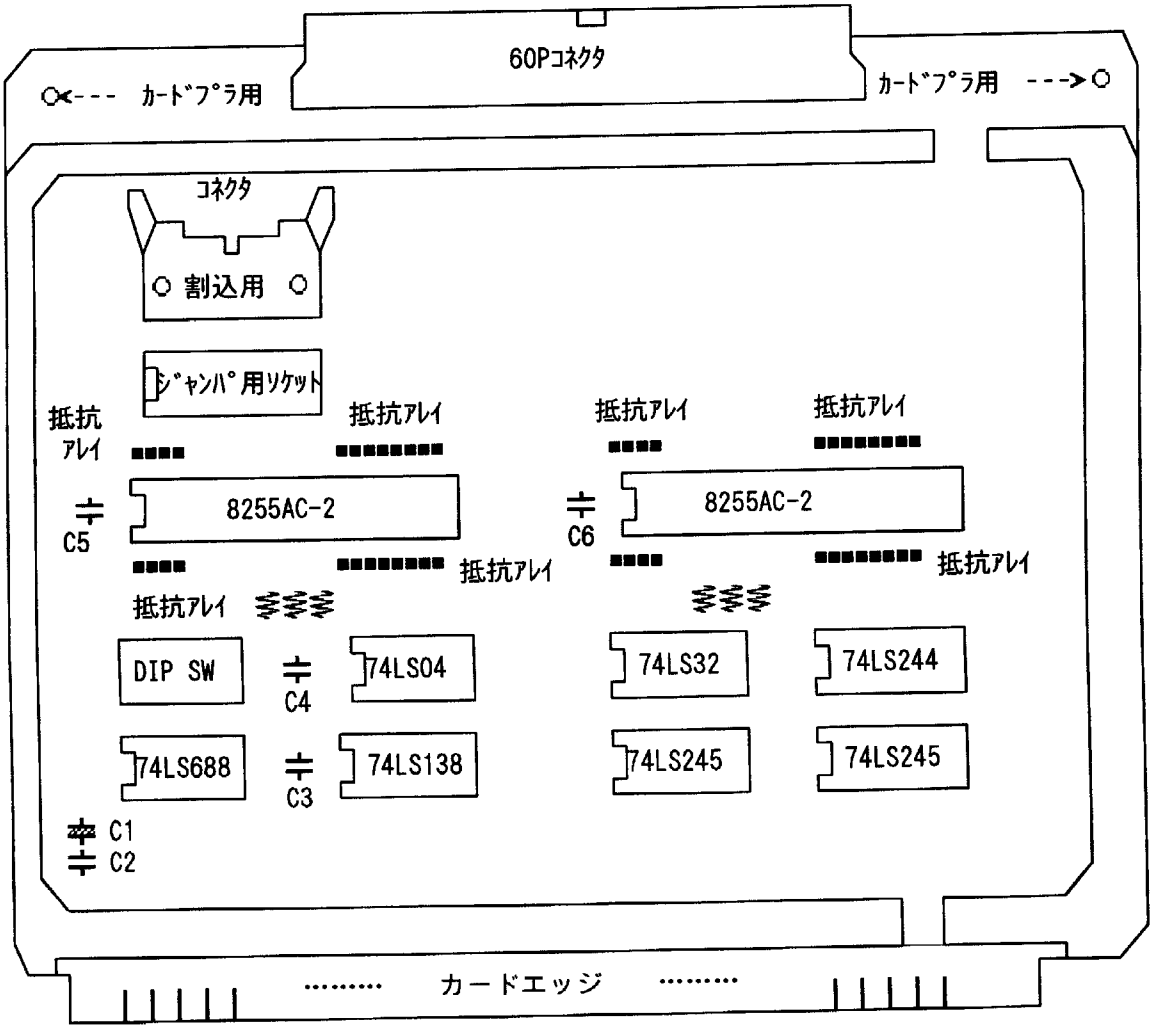


図３２．　１６ビットＩ／Ｏボード部品配置

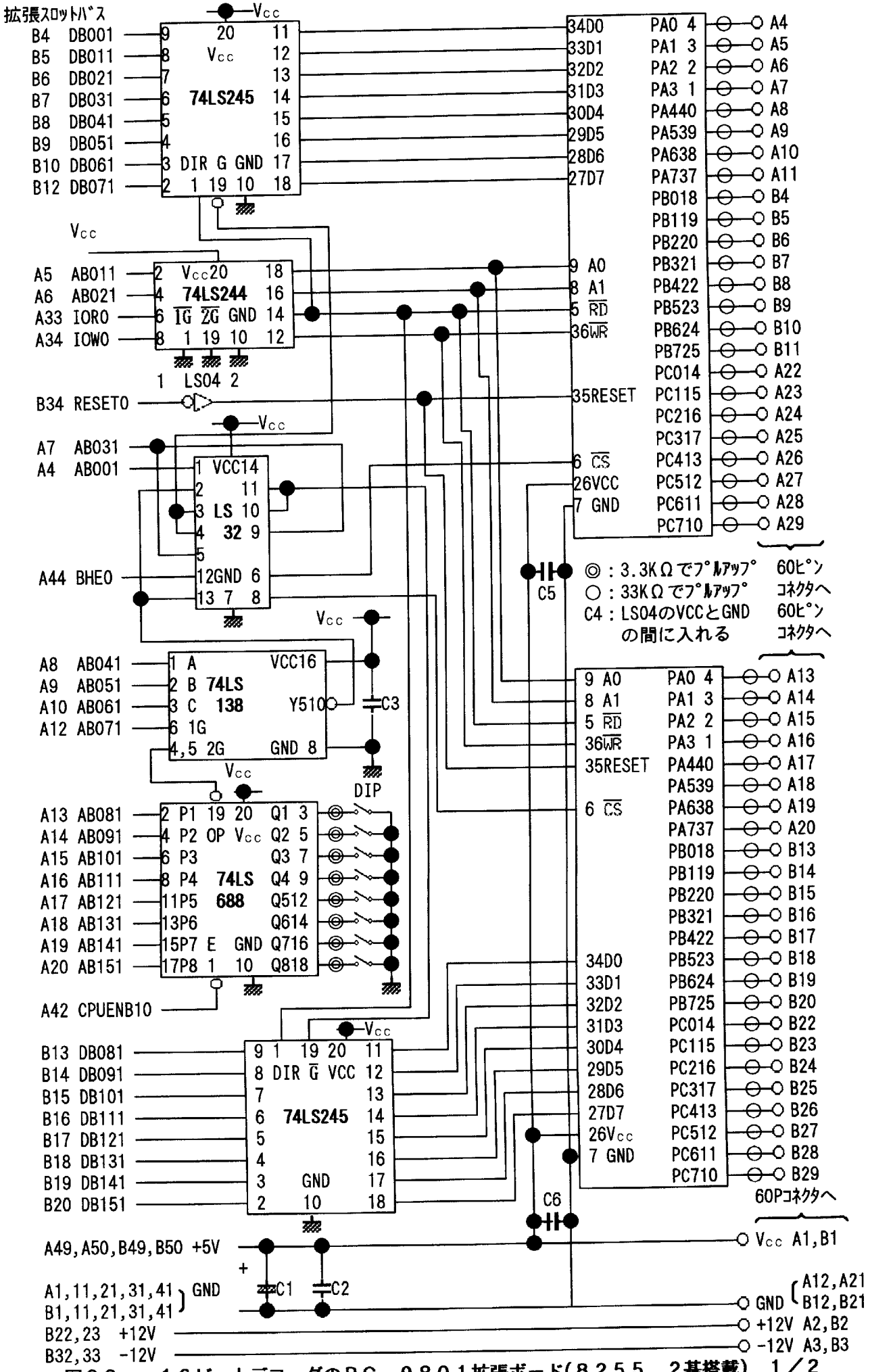
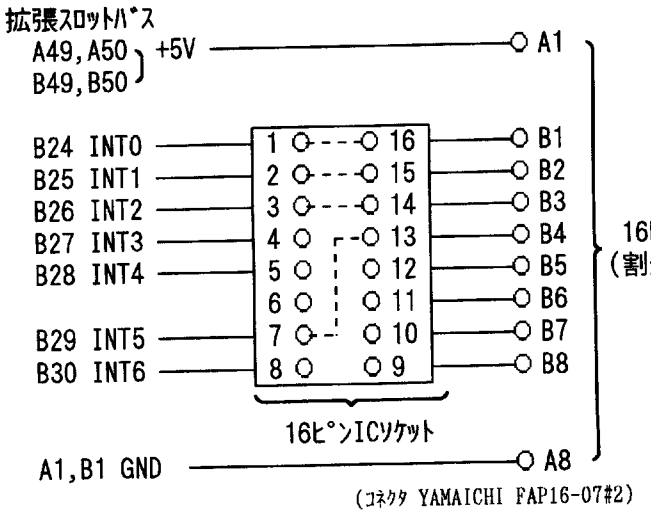


図３３ａ　１６ビットデコーダのＰＣ－９８０１拡張ボード（８２５５　２基搭載）　１／２



ポート A(下位)	XXD0H	ポート C(下位)	XXD4H
ポート A(上位)	XXD1H	ポート C(上位)	XXD5H
ポート B(下位)	XXD2H	CWR (下位)	XXD6H
ポート B(上位)	XXD3H	CWR (上位)	XXD7H

注・ハンダ付け，なるだけ分散させる
・配線の済んだ部分は回路図に色を付けるようにする。
・ハンダ付けのムラやショートがないか検査する。
・テスターで導通チェックをし，誤配線や配線もれがないかチェック
・VCCとGNDラインの配線もれやショートには注意して検査する。
・+5V電源を接続し，+5Vを加えて，ICソケットのVCCピンに+5Vが加わっているかチェックする。
・以上がＯＫならばＩＣを挿入する。

図３３ｂ　１６ビットデコーダのＰＣ－９８０１拡張ボード(８２５５　2基搭載)　２／２

７．３．３　実験ボード

8255の使い方，入出力プログラムの実際を習得するための実験ボードを製作する。実験ボードは，出力データを確認する LEDと，0，1のデータを作り出すトグルスイッチ，さらに割り込み実験用の入力部からなる。ポートAは，トグルスイッチから入力データを加える。ポートBとポートCは出力データでLEDを点灯させる。8ビット用と16ビット用の回路図を図２８と図２９に示す。入力用のトグルスイッチとコネクタの間には1KΩの抵抗を入れてあるのは，ポートAを出力ポートに指定してしまったとき，8255に過大電流が流れて込んで，これを破壊しないようにするためである。４ビットのチャタレススイッチによる割り込み実験に用いる入出力用のスイッチ回路も組み込んである。割り込みの実験において，チャタリングが生じると１回の割り込み要求のつもりでもチャタリングのために数回以上の割り込み要求をしたようになってしまってよろしくない。このため RS-FF を用いてチャタリングを防止するようにした。なお，割り込みの実験をしないのなら，この部分の回路を作る必要はない。

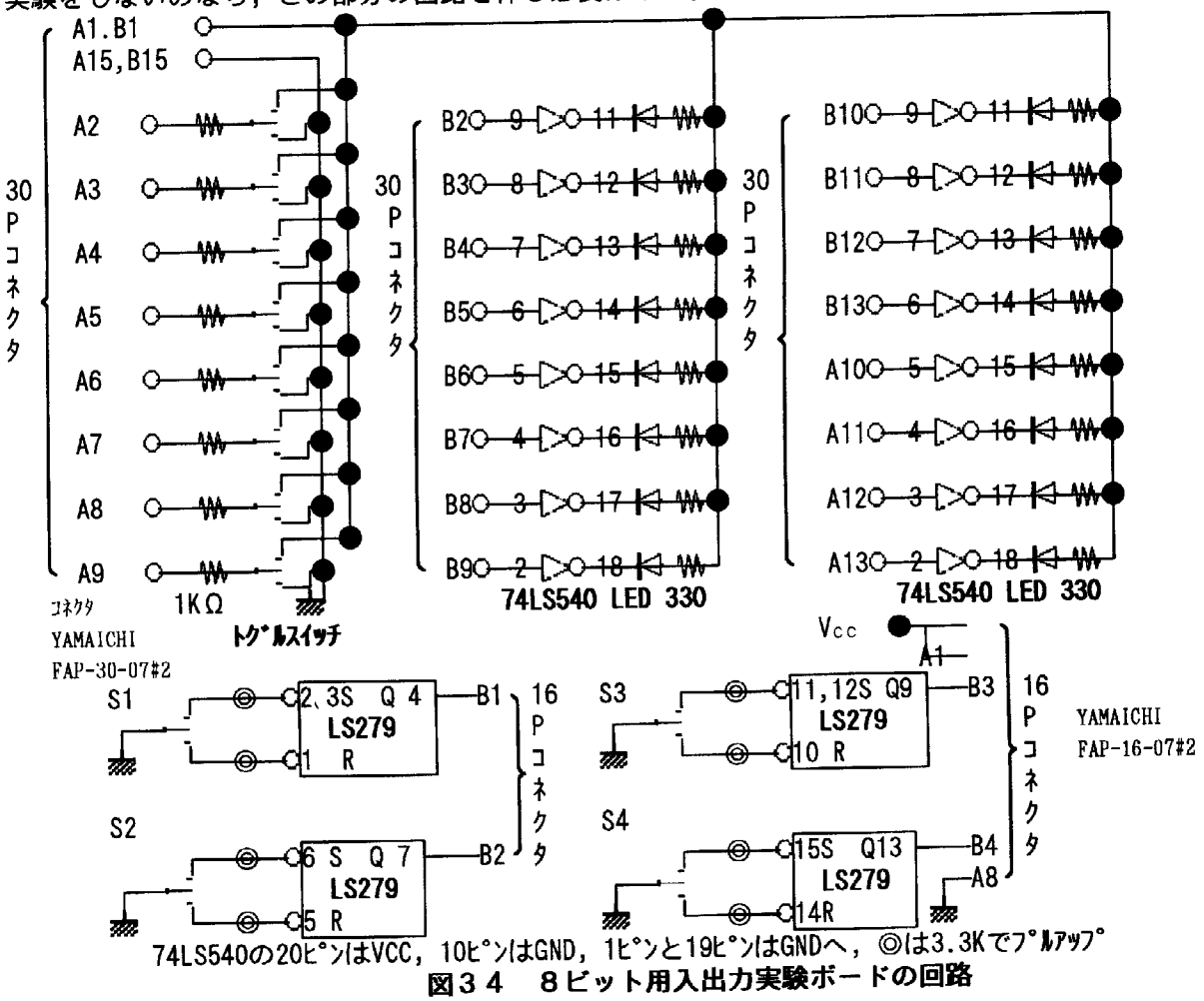
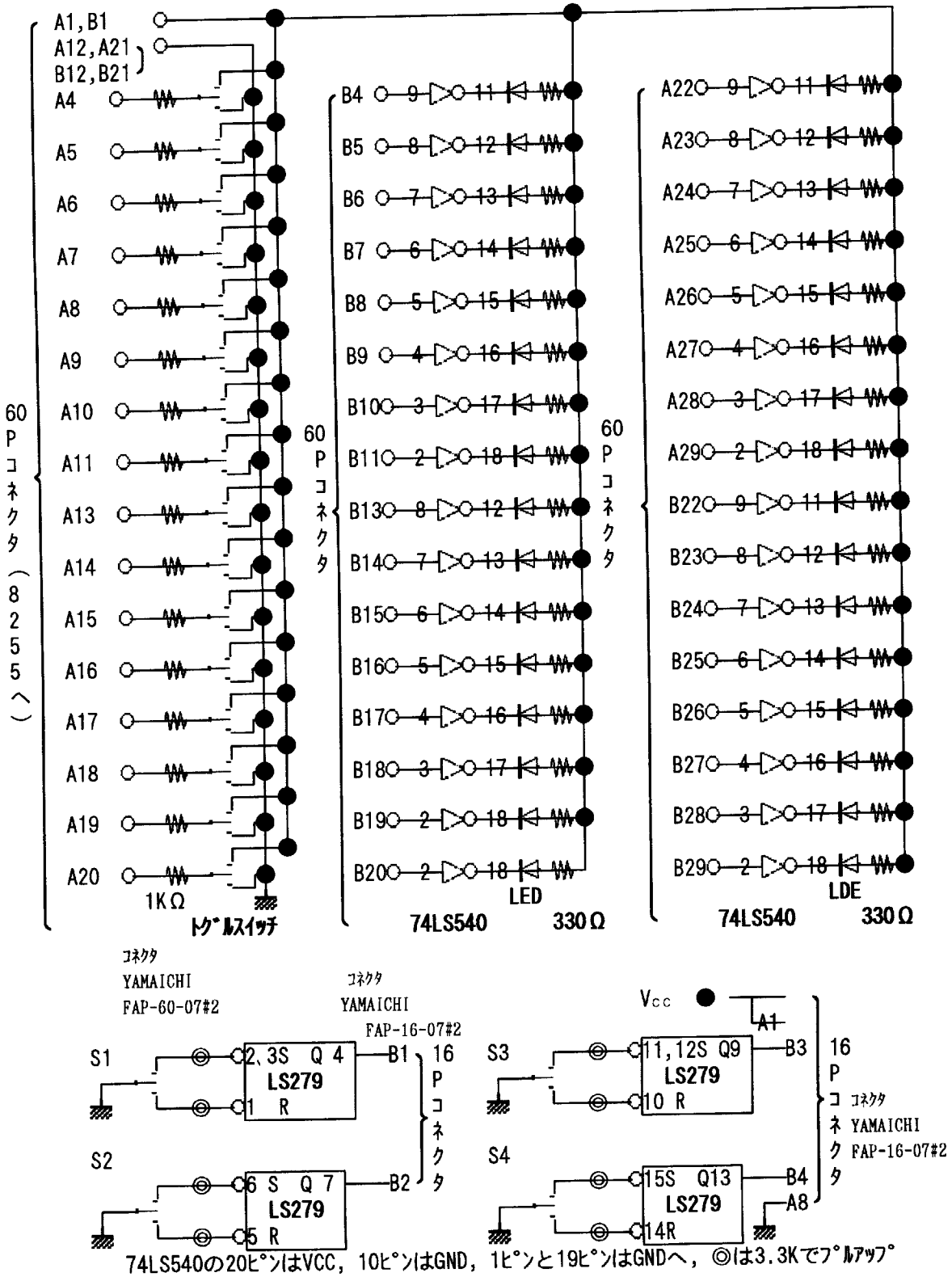


図３４　８ビット用入出力実験ボードの回路



基板	サンハヤト	ICB-502H	1	抵抗	1KΩ1/4W	16
コネクタ	YAMAICHI	FAP-16-07-#2	1	ICソケット	20P	4
〃	ヒロセ	3BB-60PA-DS	1	〃	16P	1
LED	TLR-143		32	TTL	74LS540	4
トグルスイッチ	ミヤマMS-611	3P	20	〃	74LS279	1
抵抗アレイ	330Ω*8		4			
〃	3.3KΩ*8		1			

図３５　１６ビット入出力実験ボードの回路と必要部品

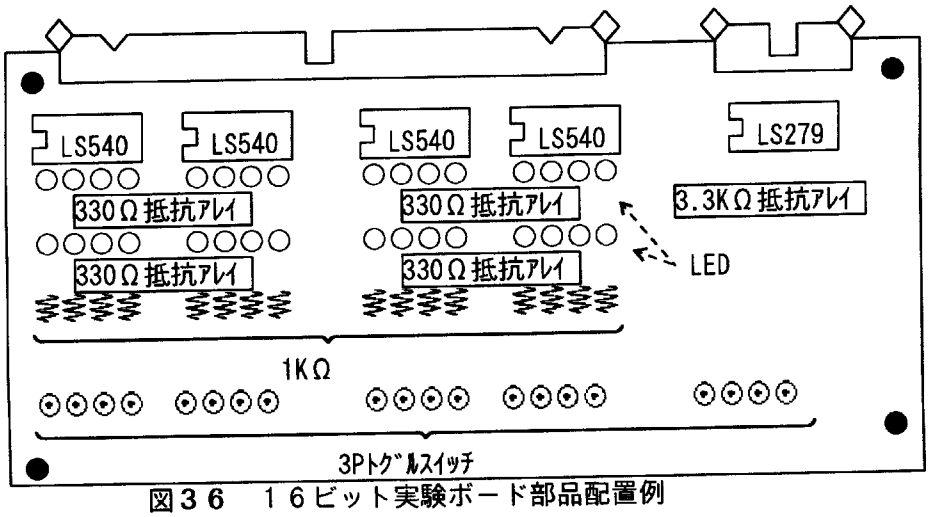


図３６　１６ビット実験ボード部品配置例

※ 定電圧電源を用いて VCCと GND間に +5Vを加える。+5Vを加えると全てのLEDが点灯する。点灯していないものがあれば誤配線かLEDの極性を逆に取り付けたかである。LEDが付いているｺﾈｸﾀのﾋﾟﾝをGNDにおとす(短絡)とLEDが消灯する。消灯しなければその部分の誤配線。ｽｲｯﾁが取いているｺﾈｸﾀの各ﾋﾟﾝとGNDの間をﾃｽﾀでﾁｪｯｸする。ﾄｸﾞﾙｽｲｯﾁをONした時(上側)に+5V，OFF時(下側)0Vになればﾎﾞｰﾄﾞは完成。

７．３．４　Ｉ／Ｏボードの動作確認テストプログラム

<準備>

- パソコンの電源を切った状態で，8255I/Oﾎﾞｰﾄﾞ を拡張ポートの＃１スロットに差し込む。
- コネクタ付ケーブルを用いて8255I/Oボードと入出力実験ボードとを接続する。
- パソコンの電源を入れる。無事にパソコンが立ち上がればＯＫである。パソコンが立ち上がらず，画面が乱れたりしているときは，直ちにパソコンの電源を切り，製作したボードを点検する。
- パソコンが正常に立ち上がった場合は，次に例示したＩ／Ｏボードテストプログラムを入力し，これを実行する。
- 実行すると，スイッチのＯＮ／ＯＦＦが入力され，そのままＬＥＤに出力される。
- スイッチのデータが正しく表示され，対応したＬＥＤが点灯すれば正常に動作している。

※　**使用言語**：NECのN88-BASIC(86)，MS-DOSver2.11/3.3，MASM(MS-DOS付属)，TURBO C Ver 2.0。

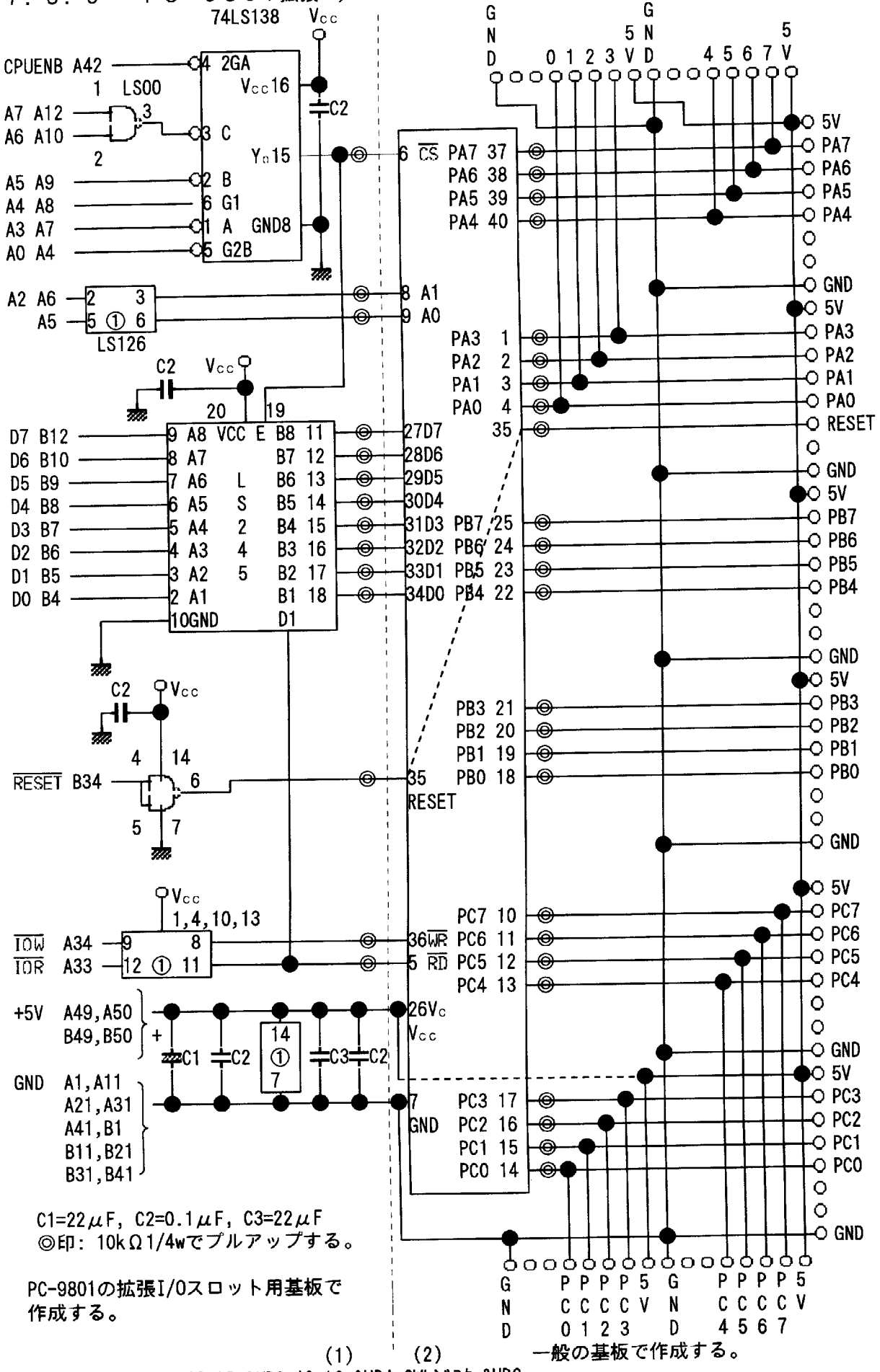
(1) Ｉ／Ｏボード動作確認テストプログラム(１６ビットデコード，図３３)

```
 10 '8255 I/O TEST PROGRAM(16BITS)
 20 '  8255 ADDRESS
 30 '  MODE 0
 40   PA0=&HD0    <--> &HFFD0
 50   PA1=PA0+1
 60   PB0=PA0+2
 70   PB1=PA0+3
 80   PC0=PA0+4
 90   PC1=PA0+5
100   CR0=PA0+6
110   CR1=PA0+7
120   CW=&H90   'CONTROL WORD
130   OUT CR0,CW '8255 INITIALAIZ
140   OUT CR1,CW
150 '
160 CLS
170 D0=INP(PA0) 'DATA INPUT
180 LOCATE 30,12 :PRINT "LDATA=";D0
190 D1=INP(PA1)
200 LOCATE 30,10 :PRINT "HDATA=";D1
210 OUT PB0,D0 'DATA OUTPUT
220 OUT PB1,D1
230 OUT PC0,D0
240 OUT PC1,D0  <--> D1
250 GOTO 170
260 END
```

(2) Ｉ／Ｏボード動作確認テストプログラム(８ビットデコード，図２４)

```
 10 '8255 I/O TEST PROGRAM(8BITS)
 20 '  8255 ADDRESS
 30 '  MODE 0
 40   PA0=&HD0
 50   PB0=PA0+2
 60   PC0=PA0+4
 70   CR0=PA0+6
 80   CW=&H90   'CONTROL WORD
 90   OUT CR0,CW '8255 INITIALIZE
100 '
110   D0=INP(PA0) 'DATA INPUT
120   LOCATE 30,10:PRINT "SW DATA=";D0
130   OUT PB0,D0 'DATA OUT
190 OUT PC0,D0 'DATA OUT
200 GOTO 110
210 END
```

７．３．５　ＰＣ－９８０１拡張Ｉ／Ｏボード回路（１６ビットデコード）



C1=22μF, C2=0.1μF, C3=22μF
◎印: 10kΩ1/4wでプルアップする。

PC-9801の拡張I/Oスロット用基板で
作成する。

一般の基板で作成する。

ﾎﾟｰﾄA:&HD0、ﾎﾟｰﾄB:&HD2、ﾎﾟｰﾄC:&HD4、CWﾚｼﾞｽﾀ:&HD6
(1)と(2)は 20芯フラットケーブルで接続する。

図３６　ＰＣ－９８０１拡張スロットバス用８２５５Ｉ／Ｏボード

（１） 拡張スロット用８２５５Ｉ／Ｏボードのアドレス

A1	A0	ｵﾌｾｯﾄ	16進数	ﾎﾟｰﾄ/ﾚｼﾞｽﾀﾞｽﾀ	ｱﾄﾞﾚｽ(PC-9801用として製作したﾎﾞｰﾄﾞの場合) 10進数	BASIC的表現	C言語的表現
0	0	0	00H	ﾎﾟｰﾄA	208	&HD0	0XD0+0*2=0XD0
0	1	2	01H	ﾎﾟｰﾄB	210	&HD2	0XD0+1*2=0XD2
1	0	4	02H	ﾎﾟｰﾄC	212	&HD4	0XD0+2*2=0XD4
1	1	6	03H	ｺﾝﾄﾛｰﾙﾚｼﾞｽﾀ	214	&HD6	0XD0+3*2=0XD6

（２） Ｃ言語を用いる場合の手続き

①予めカレントディレクトリに以下のヘッダファイルを作って用いる。

PC-9801に内蔵されている 8255(プリンタ)を使う場合

PRINTER.h

```
#define STATUS 0x42
#define BUSYBIT 4
#define BUSY    0
#define COMMAND 0x46
#define STROBE0 0x0e
#define STROBE1 0x0f
#define PRDATA  0X40
```

製作した8255I/Oボードを使う場合

TNKIN.h

```
#define PD8255   0xd0        /* 8255 Base Address */
#define PORTA    PD8255+0*2  /*  porta            */
#define PORTB    PD8255+1*2  /*  portb            */
#define PORTC    PD8255+2*2  /*  portc            */
#define CWREG    PD8255+3*2  /*  Control Register */
```

②標準入出力関数

標準入出力関数を使う場合は，そのヘッダファイルを予めインクルードしておく。

```
outp(第１引数,第２引数);
```

第１引数：出力ポートアドレスを書く。
第２引数：出力データを書く。

```
inp(引数);
    inp(PORTC);
    inp(AD);
```

ＭＳ-Ｃの標準入出力は　conio.h
ターボＣの標準入出力は　dos.h

③ #include の後ろのファイル名について

"ファイル名"　カレントディレクトリからファイルを読み込む。
<ファイル名>　環境変数INCLUDEで指定したディレクトリからファイルを読み込む。

（３）動作確認テストプログラム

全てのポートを出力とすればコントロールワードは８０Ｈとなる。全てのポートにＬＥＤが接続されているならば入力モードのときには点灯し，出力モードのときには消灯する。

```
#include "TNTKIN.h"  /* ﾌｧｲﾙ TNKIN.h の取り込み */
#include <dos.h>
void main()
{
 out(CWREG,0x80);  /* 8255 の初期化  */
 }
```

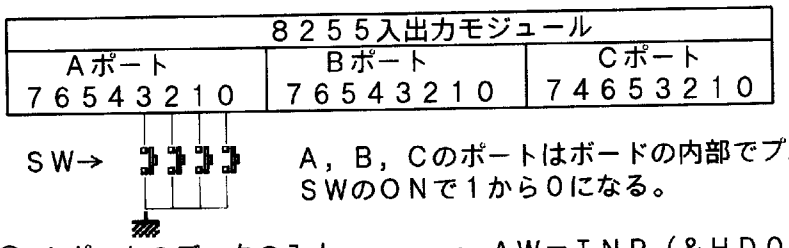
【実験】　Ｃ言語を用いて全ポート出力，各ポートへ同じデータを出力する。

```
#include "TNKIN.h"
#include <conio.h>
#include <dos.h>
main()
{
int out_data;
outp(CWREG,0x80);
out_data = 0;
while{(bdos(6,0xff) & 0xff) == 0){
outp(PORTA,++out_data);
outp(PORTB,out_data);
outp(PORTC,out_data);
timer(20000);
}  /* end of while */
} /* end of main */
timer(t)
unsibned t;
{
  for(; t; --t);
}
```

【実験】Ｎ８８－ＢＡＳＩＣ（８６）を使う場合

コントロールワードをＯＵＴ命令でＣＷレジスタ(アドレス＆ＨＤ６)へ出力する。
ポートＡ入力，ポートＢとＣを出力…… ＯＵＴ　＆ＨＤ６，＆Ｈ９０

８２５５のポートに接続した装置からＰＣ－９８０１にデータを取り込む方法

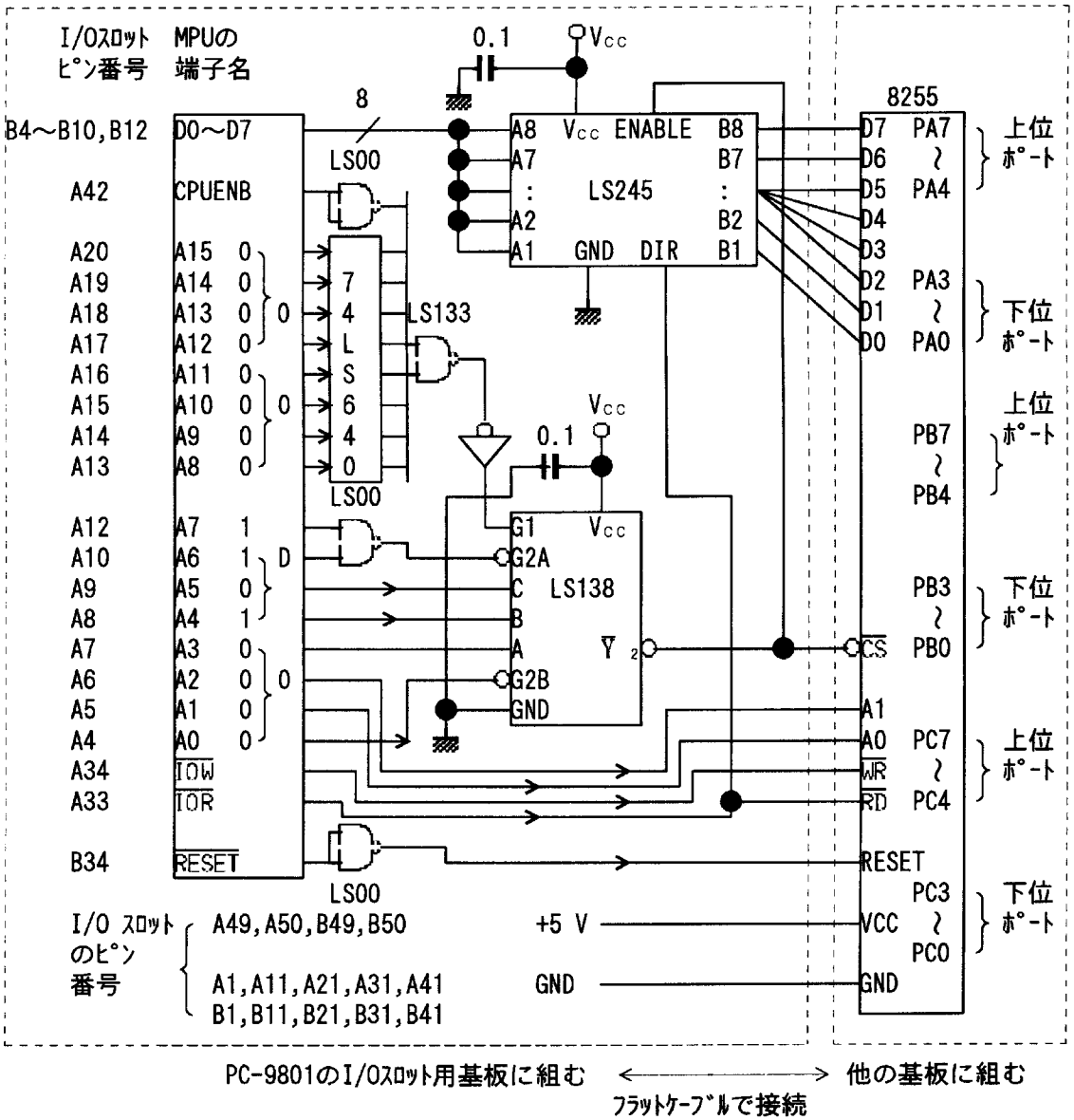


Ａ，Ｂ，Ｃのポートはボードの内部でプルアップされているので，
ＳＷのＯＮで１から０になる。

① Ａポートのデータの入力　　：ＡＷ＝ＩＮＰ（＆ＨＤ０）
② 入力データのビット反転　　：ＡＷ＝２５５－ＡＷ
③ ＳＷのＯＮ，ＯＦＦの場合の処理：ＩＦ　ＡＷ＝１　ＴＨＥＮ　～

```
110 '  メインルーチン
120 GOSUB 170 :'初期設定
130 GOSUB 250 :'ｺﾝﾄﾛｰﾙﾜｰﾄﾞ
140 GOSUB 280 :'入力ﾁｪｯｸ
150 GOSUB 320 :'表示へ
160 GOTO  140
170 '初期設定
180 WIDTH 80,25:CONSOLE 0,25,0,1:COLOR 7,0,0,7
190 SCREEN 3:CLS 3:LOCATE 0,0
200 PRINT "****************************  スイッチ　チェック　  **********************"
210 LOCATE 20,4
220 PRINT "押しボタンスイッチの　どれかを押して下さい"
230 LOCATE 0,7:PRINT STRING$(80,"-")
240 RETURN
250 'コントロールワードセット
260 OUT &HD6,&H90
270 RETURN
280 '入力チェック
290 A=INP(&HD0):A=255-A
300 S3=A AND 8:S2=A AND 4:S1=A AND 2:S0=A AND 1
310 RETURN
320 '表示
330 COLOR 2
340 IF S3<>0 THEN LOCATE 28,12:PRINT "スイッチＳ３が　ＯＮです":GOTO 360
350 LOCATE 28,12:PRINT SPC(25)
360 COLOR 4
370 IF S2<>0 THEN LOCATE 28,14:PRINT "スイッチＳ２が　ＯＮです":GOTO 390
380 LOCATE 28,14:PRINT SPC(25)
390 COLOR 5
400 IF S1<>0 THEN LOCATE 28,16:PRINT "スイッチＳ１が　ＯＮです":GOTO 420
410 LOCATE 28,16:PRINT SPC(25)
420 COLOR 6
430 IF S0<>0 THEN LOCATE 28,18:PRINT "スイッチＳ０が　ＯＮです":GOTO 450
440 LOCATE 28,18:PRINT SPC(25)
450 RETURN
```

7．3．6　PC－9801拡張I／Oボード回路（16ビットデコード）



1) 8255 への信号線は 10KΩ1/W でプルアップ(Vccに接続)する。
2) 8255 の各ポートは上位４ビットと下位４ビットに分け，それぞれに Vcc と GND を配する。
3) LS640はMPU側から8255側への動作となるようにLS340の１番ピンをVccに接続すること。

図３８　８２５５を搭載した拡張スロット用ボード(PC-9801)

8　課　題

（１）　４節の課題１と課題２の回路を組んで実験を実施せよ。

（２）　図３０，図３３，図３７のどれか１つを製作せよ。
・図３０を選択したときは図３４を，図３３を選択した場合は図３５の入出力実験ボードも製作せよ。

（３）製作した１／Ｏボードを使って，指導書に例示してあるプログラムを参考にして動作実験をおこなえ。

（４）マイコン実験で用いた入出力モジュールやステッピングモータ駆動回路あるいはＡ／Ｄ変換器を動作させよ。

（５）指導書に例示したプログラム例などを参考にし，製作したボードを用いて8255,8253,8259を使った簡単なプログラムを製作せよ。

（６）表１の周辺ＬＳＩを１つ取り上げて特徴を簡単に説明せよ。

（７）今回製作したボードのアドレスは D0 であった。これを D2，D4，D6，D8，B2，B4，B6，B8，C2，C4，C8，E2，E4，E8 から１つ選択し，回路図を変更せよ。

（８）インターフェースについて考察と感想を加えよ。

9．参考資料

1)栗山，他　：PC-Techknow8800Vol.1，ｱｽｷｰ出版(1982)

2)藤田，他　：PC-Techknow9800Vol.1，ｱｽｷｰ出版(1988)

3)ｱｽｷｰ出版局ﾃｸﾗｲﾄ編、新版PC-9800ｼﾘｰｽﾞ ﾃｸﾆｶﾙﾃﾞｰﾀﾌﾞｯｸ(1990.4)

4)ｱｽｷｰ出版局ﾃｸﾗｲﾄ編、PC-9800ｼﾘｰｽﾞ ﾃｸﾆｶﾙﾃﾞｰﾀﾌﾞｯｸ 製品情報編(1996.3)

5)ｱｽｷｰ出版局ﾃｸﾗｲﾄ編、PC-9800ｼﾘｰｽﾞ ﾃｸﾆｶﾙﾃﾞｰﾀﾌﾞｯｸ BIOS･DEVICE編(1996.3)

6)ｱｽｷｰ出版局ﾃｸﾗｲﾄ編、PC-9800ｼﾘｰｽﾞ ﾃｸﾆｶﾙﾃﾞｰﾀﾌﾞｯｸ Windows API編(1996.4)

7)NEC PC-8801 USER'S MANUAL，日本電気(株)

8)NEC PC-9801 USER'S MANUAL，日本電気(株)

9)CQ 出版　：'89 最新 TTL IC 規格表

10)Robert L.Hummel,槌田浩一　訳:80x86/80x87　ﾌｧﾐﾘｰ･ﾃｸﾆｶﾙﾊﾝﾄﾞﾌﾞｯｸ，技術評論社(平成5年10月)

11)末松　　：制御用マイコン入門，オーム社(昭和58年)

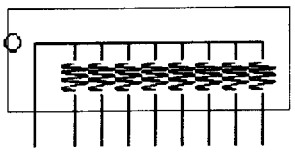
12)大久保陽一：機械に知力をつける「制御用マイコン初歩から応用まで」，日刊工業新聞社(昭和60)

13)相原隆文　：初歩のディジタル回路4 「Ｚ－８０実用マイコン製作技術」，技術評論社(昭和61年3月)

14)天良和男　：知的実験ツールとしてのﾊﾟｿｺﾝ活用「ﾊｰﾄﾞ･ｿﾌﾄ･ｾﾝｻ技術」，東京電機大学出版局(1991.1)

付録　製作上の心得

抵抗アレイ



○コモンマーク

(1)組立順序

部品の組立は次のように行うのが原則的である。

①部品の位置を決定する。
- ・基板上に部品を並べて相互の関係、全体のバランスをとる。
- ・配線の関係を調べる。

②スイッチやボリューム等の取り付け穴をあける。
③抵抗を取り付ける。

抵抗

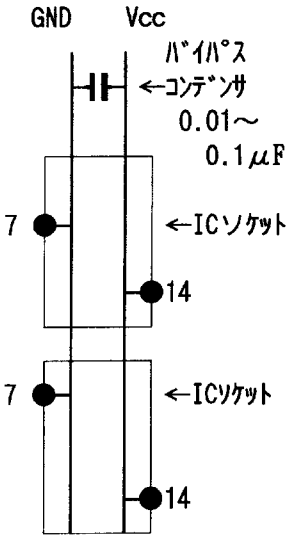
④極性のない部品の取り付け方

基板に対して水平に配置する場合は;
表示が左から右に向くように取り付ける。
基板に対して垂直に配置する場合は；表示が下から上に向くように取り付ける。

⑤ＩＣソケットの取り付け

対角線のピンのみをハンダ付けし、全部のＩＣソケットを取り付けたら、ソケットの向きや基板からの浮き上がり等がないことを確認した後で、全部のピンをハンダ付けする。

⑥コネクタの取り付け
⑦ＬＥＤの取り付け
⑧スイッチの取り付け
⑨ＶｃｃとＧＮＤの配線
⑩　全体の配線



(2)ＶｃｃとＧＮＤの配線方法

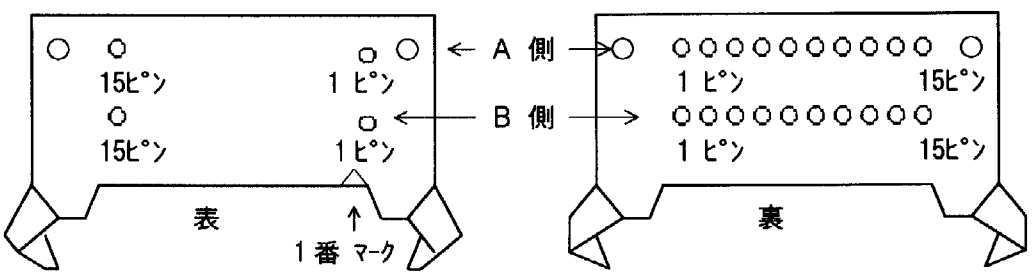
０．５ｍｍの錫メッキ線を使って、図のように配線する。
配線が交差することのないように部品配置や線の引き回しを考えること。どうしても交差する場合には、基板の表側に配線するか、あるいは絶縁チューブ（エンパイアチューブ）を被せるか、または被覆線を使う。

(3)一般の配線方法

回路図に従ってビニール被覆線で１本１本確実にハンダ付けをする
線の引き回しは、原則としてＩＣソケット・ピンの上は避ける。
多少遠回りしてもよいから線は、原則として直角になるように配線すること。
１ヶ所にハンダ付けをするのは２本までとする。それ以上となる場合は２本になるように分割すること。

(4)コネクタの配線

コネクタのピン番号は裏と表で、左右が逆になる。勘違いしないように注意すること。
間違えないように配線するには、油性の極細ペンを使って、基板にピン番号を書いておくとよい。



８２５５ボードの有効利用のための入出力用コネクタボード
（ 図３０または図３３用）