MCS-6502をCPUとするマイクロコンピュータシステム

—高速ドットプリンタPA6651-Aとのインタフェース—

八田 信*・高浪 五男**・井上 克司**

6502 Microcomputer System-Interface with Dot Printer PA6651-A-

Makoto HATTA, Itsuo TAKANAMI and Katsushi INOUE

Abstract

We connect a high speed dot-printer (120 characters/sec) with the MCS-6502 microcomputer system. To do so, we study the function of PIA (Peripheral Interface Adapter) and design the hardware and software for controlling the dot-printer.

1. はじめに

大規模集積回路（LSI）技術の進歩により，マイクロプロセッサが低廉かつ手軽に入手できるようになった．これらを縦横に駆使できるようになれば，我々の意図にマッチした複雑なシステムを極めて安価に作ることができる．一方，これらのプロセッサ，キーボード，表示器などを組み合せたワンボードマイクロコンピュータキットが数多く市販されており，これを用いてシステムの拡張ができれば，安価ばかりでなく，相当の手間が省け，かつ信頼性も大きいものとなる．その際これに接続する装置（例えば，カセットレコーダ，キャラクタディスプレイ，ADコンバータ，テレタイプ，紙テープ読取器，パンチャー，フロッピーディスク，等々）との間のインターフェイスを構成して

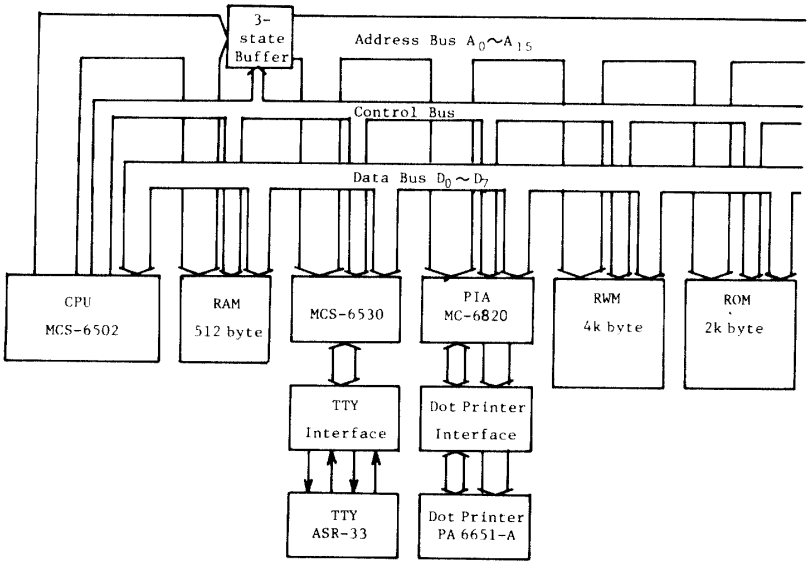

Fig. 1 Organization of MCS-6502 Microcomputer System

* 大学院電気工学専攻
** 電子工学科

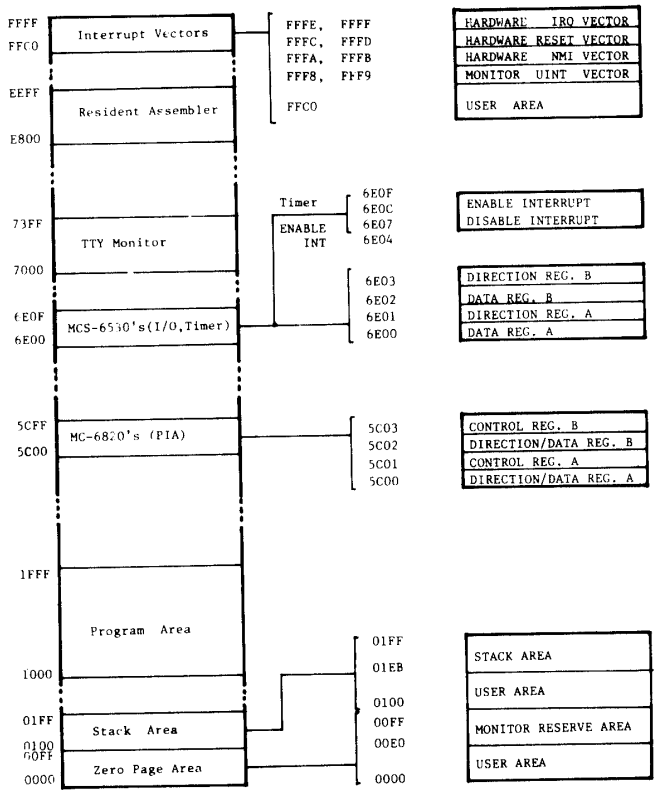

Fig. 2　Address Map of MCS-6502 Microcomputer System

これを制御するソフトウェアを作る必要がある．このように，マイクロコンピュータを応用したシステムを作るには，ハードウェアの面とソフトウェアの面の両方の理解を必要とする．

さて，市販されているマイクロコンピュータキットは大きくインテル8080系とモトローラ6800系とに分けることができる．ここでは6800系に属すモステクノロジー社のマイクロプロセッサMCS-6502を用いたキットJOLTをベースにしたシステムの拡張の一環として，これに高速ドットプリンタ（120文字／秒）を接続したので，そのインターフェイスの設計（ハードウェアとソフトウェア）について報告する．第2章ではインターフェイス回路とこれをコントロールするソフトウェアの説明に必要な本システムの概要について，システム構成及び構成要素であるCPU・MCS-6502，多機能LSI・MCS-6530，PIA・MC-6820に関して最小限の説明を行なう．第3章では，ドットプリンタの仕様とそれに基づくインターフェイス回路の設計及びこれをコントロールするプログラムについて述べる．

2.　システムの概要

まず，図1に本システムの現在の構成図を示す．
図2にアドレスマップを示す．

2.1　MCS-6502

システムの心臓部となる8ビットのマイクロプロセッサで，次の様な特長をもっている．

① 56種類の命令と13種類のアドレッシングモードをもっている．（豊富なアドレッシングモード）
② BCD演算が容易である．
③ パイプラインアーキテクチャーにより演算速度が速い．
④ 2相クロックゼネレータを内蔵している．
⑤ モトローラ MC-6800 とピンコンパチブルである（ハードウェアの機能の相違はある）

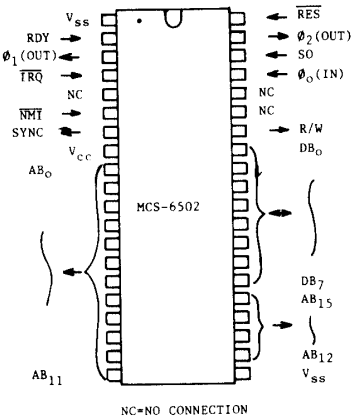

Fig. 3　Pinout Designation of MCS-6502

ピン接続図を図3に示す.

インターフェイス設計に必要な制御信号についてのみ説明する.

◦R/W　リードライトコントロールライン

ライトサイクルのときLowレベルとなり，そうでないときはHighレベルのリード状態となる.

◦$\overline{RES}$　リセットライン

いつもはHighレベルであるが，Lowレベルにしたとき，プロセッサが初期設定され，再びHighレベルになったとき*$FFFCとFFFD番地の内容の示す番地から処理を始める.（このような方法をベクトルスタートという）

◦ϕ_1, ϕ_2　クロックライン

ϕ_1, ϕ_2は内部で発生される2相（互いに逆相）のクロックパルス出力である.アドレス信号，データ信号，命令の実行のタイミングをきめる重要な働きをする.

内部は，アキュームレータ〔ACC〕，インデクスレジスタ2種〔IXとIY〕，プロセッサステイタスレジスタ〔PS〕，プログラムカウンタ〔PC〕，スタックポインタ〔SP〕などから構成されている.

PSは，図4に示すような7種類のフラグからなる8ビットのレジスタでCPUの状態を示している.

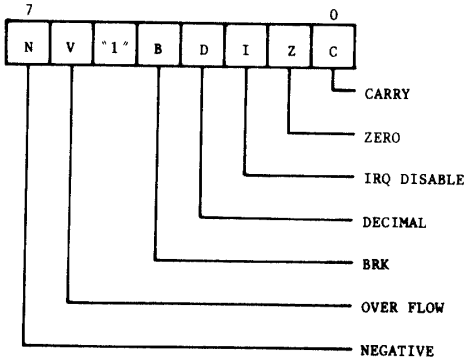

Fig. 4　Processor Status Register

PCは16ビットのレジスタで，プロセッサはこのレジスタの示す番地の内容をフェッチする.レジスタの値は1回のフェッチ毎にインクリメントされる.

SPはACCやPSの内容を退避させる場合の退避先のアドレスを指す（ポイントする）ためのレジスタである.通常アドレスは16ビットで示されるがMCS-6502では，スタックエリアの上位アドレスが$01にきめられているので実質は8ビットとみることができる.レジスタの値は1回の退避ごとに$01FFからデクリメントされていく.（アドレスマップ参照）

2.2 MCS-6530

このチップは，MCS-6502のために設計された多機能LSIで，主に以下の4つの機能を行なう部分を含んでいる.

① ROM (Read Only Memory)

本システムのモニタープログラムDEMON（またはTIM）が書き込まれている.アドレスは$7000～$73FF番地が割り当てられている.（アドレスマップ参照）.DEMONの主な働きは，システムの初期設定，内部レジスタやメモリの内容表示や変更，テレタイプの入出力制御プログラムの実行などである.

② RWM (Read Write Memory)

64バイトからなり，$FFC0～$FFFF番地に割り当てられている.（アドレスマップ参照）.これらは，リセットや割り込みのさいのベクトルスタートエリアとして用いられる.

③ PIA (Peripheral Interface Adapter)

8ビットからなるA，B2つのポートがあり，それぞれに，DDR (Data Direction Register) とPDR (Peripheral Data Register) とを持っている.8ビットのデータの各ビットはDDRの各ビットの値によって，PIAに入力されるのかPIAから出力されるのかが独立に決定される.AポートではPDRに$6E00, DDRに$6E01が，BポートではPDRに$6E02, DDRに$6E03が割り当てられている.（アドレスマップ参照）

④ Interval Timer

割り込み発生機能をもったプログラマブルタイマーである.

2.3 MC-6820

CPUと周辺装置とのインターフェイスに用いるために設計された汎用LSIで，あらゆる周辺装置に対応できるように，その機能はプログラマブルになっている.以下ここで必要と思われる部分のみ簡単に説明する.

2.3.1 CPU側との接続

◦CS1, CS2, CS3：チップセレクト端子で，PIAに割り当てられる番地が決定される.

◦RS0, RS1：レジスタセレクト端子で，ここに与えられる信号によって対応する内部レジスタが選択される.

◦D_0～D_7：CPU側のデータバスラインと直結する.

◦$\overline{IRQA}$, $\overline{IRQB}$：周辺装置からCPUへの割り込み要求を行なう信号端子.

2.3.2 周辺装置側との接続

脚注 *$は Hexadecimal（16進）表記.以下同じ.

Flag Status		CA(B)2 Control			DDR Access	CA(B)1 Control	
7	6	5	4	3	2	1	0
IRQA(B)1 set by CA(B)1	IRQA(B)2 set by CA(B)2	0 Input Mode	Active Edge	Interrupt Enable	0 Data Direction Register	Active Edge	Interrupt Enable
			0 Negative	0 Disable		0 Negative	0 Disable
			1 Positive	1 Enable		1 Positive	1 Enable
0 Rest		1 Output Mode	0 Hand-Shaking	0 Level	1 Peripheral Interface	Input only	
1 Set				1 Pulse			
reset by "Read Peripheral Output Register"			1 Manual Output	0 "L" out			
				1 "H" out			

Fig. 5　Control Register

A, B 2つのポートがあり，それぞれ周辺装置と CX1, CX2, PX_0～PX_7端子*を通してデータのやり取りを行なう．これらのラインの働きは内部レジスタの内容によって決定される．

(i)　Data Direction Register (DDR)

8ビットのレジスタで．これの内容によって周辺装置との8ビットのデータの各ビットについて，それを PIA に入力するか，PIA から出力するかが決定される．Aポートの DDRには $5C00, Bポートの DDR には $5C02 番地が割り当てられている．(アドレスマップ参照)

(ii)　Control Register (CRX)

PIAの機能を決定するもので，このためにこのチップがプログラマブルといわれる．Aポートの CRA には $5C01, BポートのCRBには $5C03番地が割り当てられている．(アドレスマップ参照) これの機能について詳述する．これは8ビットのレジスタで，各ビットにはそれぞれ図5に示すような意味がある．以下これらについて説明する．CRXの第iビット目を CRX_i で表わす．

◦ DDR Access Bit; CRX_2

このビットは RS0, RS1 と共に内部レジスタの選択を行なう．図6参照

RS1	RS0	CRA-2	CRB-2	Register Selected
0	0	1	-	Peripheral Interface A
0	0	0	-	Data Direction Register A
0	1	-	-	Control Register A
1	0	-	1	Peripheral Interface B
1	0	-	0	Data Direction Register B
1	1	-	-	Control Register B

Fig. 6　Register Selected

脚注 ＊記号Xは　Aポートのときには“A”
　　Bポートのときは“B”を意味する．

◦ Interrupt Flag Status Bits; CRX_6, CRX_7

CRX_6 は CRX_5=0 (Input mode) のときピン端子 CX2 へのアクティブエッジ (CRX_4=0 のときはネガティブエッジ，1のときポジティブエッジ) でセットされる．

CRX_7 はピン端子 CX1 へのアクティブエッジ (CRX_1=0 のときはネガティブエッジ，1のときポジティブエッジ) でセットされる．

これらのビットのリセットは CPU が Peripheral I/O ポートを読む (すなわち，ポートからデータを入力する) ときに自動的に行なわれる．

このフラグによる割り込み要求は CRX_0 と CRX_3 で制御される．すなわち，$\overline{IRQX}$ が Low レベルになるのは CRX_7=CRX_0=1 かまたは CRX_6=CRX_3=1 のときである．

◦ CX2 Control Bits; CRX_3, CRX_4, 及び CRX_5.

CRX_5 が0のとき，CX2 は Input mode と呼ばれ上で説明した通りである．1のときは Output mode と呼ばれ，これは更に CRX_4 の値によって2つの mode に分かれる．すなわち，CRX_4=1 のとき Manual Output mode と呼ばれ，CX2 は1ビットの出力ポートとなる．このとき CX2 から出力される信号は CRX_3=0 のとき Low レベル，1のとき High レベルとなる．CRX_4=0 のときは Hand-shake mode

	$CRA(B)_3$	Set	Reset
CA2	0	CA1 の Active Edge で1となる．(*1)	CPUの "Read Data" 命令実行後の Enable Pulse (ϕ_2) の Negative Edge で0となる (*2)
	1	CPU の "Read Data" 命令実行後の Enable Pulse の Negative Edge で1となる．(*3)	CPUの "Read Data" 命令実行後の Enable Pulse の Negative Edge で0となる．(*4)
CB2	0	CB1 の Active Edge で1となる．(*5)	CPU の "Write Data" 命令実行後の Enable Pulse の Positive Edge で0となる．(*6)
	1	CPU の "Write Data" 命令実行後の Enable Pulse の Positive Edge で1となる．(*7)	CPUの "Write Data" 命令実行後の Enable Pulse の Positive Edge で0となる．(*8)

Fig. 7　Handshake mode

と呼ばれ，CRX_3に依存して図7のような動作をする．

このモードのBポートでのタイムチャートを図8に示す．

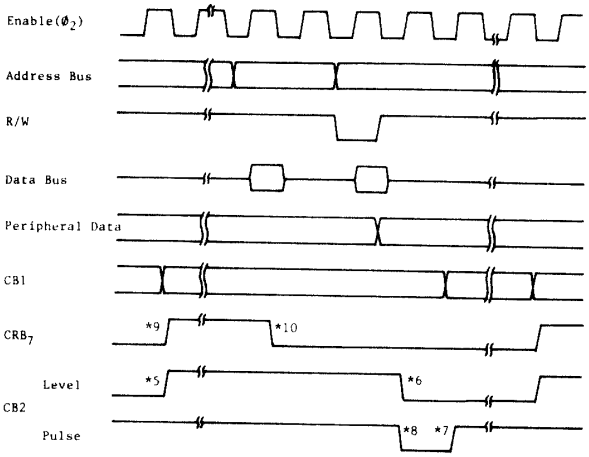

Fig. 8 Peripheral B Interface Timing

3. ハンドシェーク技法による高速ドットプリンターとのインターフェース

3.1 ハンドシャーク技法

ハンドシェーク技法は，データ処理速度の異なる装置間で確実なデータ転送を行なうための1方法である．この方法を一般的に眺めてみる．*T*を送信側　*R*を受信側とする．

① *R*から*T*へ，データを要求する．(Data Request)

② *T*から*R*へ，送るべきデータの準備ができたことを知らせる．(Data Ready)

③ *T*から*R*へ，データを転送する．(Data Transmit or Data Receive)

④ *R*から*T*へ，データを受信したことを知らせる．(Data Accept)

以下①～④を繰り返す．④の Data Accept 信号は場合によっては，次のサイクルの Data Request 信号となることもある．

次に，この①～④を具体的に PIA によるデータ転送に当てはめて考えてみる．今の場合は CPU 側からプリンタへ信号を送るわけであるから，*T* は CPU，*R*はドットプリンタとなる．この場合は*B*ポートを用いる．(周辺装置から CPU 側へ送る場合は *A* ポートを用いるが，今の場合は関係ないので省略する．) *T*から*R*への信号伝送にはCB2ラインが用いられ，*R*から*T*への信号伝送にはCB1ラインが用いられる．また，データ転送にはPB_0～PB_7ラインが用いられる．

① CB1 ラインが反転し (Data Request) CRB_7のフラグが立つ．CPU は CB1 を直接見ることはできないので，CRB をアクセスし，このフラグが立っているかどうかで Data Request があるかどうかを知る．

② CPU はデータを準備し，Peripheral Output Register *B*にデータを書き込み，CB2 から 0 を出力する．(Data Ready)

③ 周辺装置は，CB2＝0を受信すると出力ポートのデータを読み取り (Data Receive)，CB1 を再び反転させる (Data Accept)．①に戻って次の Data Request を行なう．

3.2 トットプリンタとのインターフェイス

これまでは，かなり一般的なハンドシェークを述べてきたが，これをドットプリンタに適用するには，プリンタの入出力機能と条件を知る必要がある．これを図9に示す．

端子	L レベル	H レベル
$\overline{\text{READY}}$ Output	プリント開始かできる状態 ・プリントヘッドがホームポジションにある．	プリント開始かできない状態 ・プリント中 ・紙送り中 ・ヘッドの保護回路が作動
$\overline{\text{PRINT}}$ Input	プリントヘッドを FWD 方向に移動させ 入力データに従って印字してゆく．	LからH への変化で一行の印字が終了し プリントヘッドをホームポジションに戻す．
$\overline{\text{DATA REQ}}$ Output	一文字分のデータを送信側に要求する．2.5msecのパルス信号．	受信データを処理 (印字) する．
$\overline{\text{FEED}}$ Input	紙送り信号で、Lの間は 200 msec 毎に一行紙を送る．最低パルス幅 100 msec	紙送り禁止．
$\overline{\text{FEED END}}$ Output	一行紙を送る毎に 3 msec のパルス信号を出力．	紙送りなし．
$\overline{\text{DATA}}$ 1～7	7bits 1 文字のパラレルデータ インプット． 文字コードは JIS 7単位規格．	

Fig. 9 Pinout Description of Dot Printer

図9をもとに次のような設計方針を立てる．

CPU とのインターフェースは MC-6820 によって行ない，MC-6820 とプリンタとのインターフェースは次の様にする．

① 7ビットの文字データは MC6820 の*B* ポートのPB_0～PB_6によって出力する．

② $\overline{\text{PRINT}}$ 信号はPB_7の電圧レベルでコントロールする．

③ $\overline{\text{FEED}}$ 信号は図9より100msec以上のパルス幅を必要とする．これの実現には種々考えられるが，ここではハードウェア的に行なう．すなわち，PB_7のポジティブエッヂ（立ち上がり）で単安定マルチバイブレータをトリガし，約 140msec だけ $\overline{\text{FEED}}$ ラインを Low レベルにして1行紙送りする．

④ プリントは1行80文字とし，1行分の文字データはプリント動作が始まる前にあらかじめメモリ内の所定のエリア（ここでは＄0100～＄0150）にストア

しておく．

⑤ $\overline{\text{READY}}$ 信号と $\overline{\text{DATA REQ}}$ 信号はハンドシェークを行なうために，これらの OR（負論理）をとって CB1 に入力するが，ここでのハンドシェークは2段階に分けて行なわれる．1段目は1行分の文字データのためのハンドシェーク．しかし，実際に文字データを転送するのは1文字ずつであるので，ここで，1文字ずつのデータ転送のためのハンドシェークが行なわれる．これが，2段目である．これらを行なうための制御信号ラインは次の様になる．

(1)	1行分のData Request	$\overline{\text{READY}}$
(2)	1行分のData Ready	$\overline{\text{PRINT}}$
(3)	1行分の Data Transmit	
	(i) 1文字分の Data Request	$\overline{\text{DATA REQ}}$
	(ii) 1文字分の Data Ready	省略
	(iii) 1文字分の Data Transmit	
	(iv) 1文字分の Data Accept	$\overline{\text{DATA REQ}}$
(4)	1行分の Data Accept	$\overline{\text{FEED END}}$ or $\overline{\text{READY}}$

1文字分の Data Ready が省略されているのは，CPUのデータ処理速度が速いため，プリンタが $\overline{\text{DATA REQ}}$ を出力して次の文字のプリント動作に入る前に，1文字分のデータは確実に準備されるからである．

次に，1行のプリント行程を図10のタイムチャートを見ながら説明する．

① パワー ON

プリントヘッドがホームポジションにあることが検出されたとき，プリンタは $\overline{\text{READY}}$ 信号（Lowレベル）を出力し，CB1 に伝達される．この CB1 への立ち下がりで CB2 は High レベルになり，CRB_7 は1にセットされる．

② $\overline{\text{PRINT}}$ 信号 ON

$PB_7 = 0$ によってプリントが開始される．このとき $\overline{\text{READY}}$ 信号は High レベルとなり，CB1 も High レベルとなる．CB2 は Peripheral Output Register *B* へのデータの書き込み動作によってリセットされる．

③ CRB_7 のリセット

CPU の読み込み動作によってリセットされる．

④ $\overline{\text{DATA REQ}}$ 信号 ON

$\overline{\text{DATA REQ}}$ ラインが Low レベルとなり，CB1 ラインが Low レベルとなる．この立ち下がりにより CB2 が High レベルに，CRB_7 が1になる．CPU は CRB をアクセスし，$CRB_7=1$ を検知して文字データを1文字分 Peripheral Output Register に書き込む．同時に CB2 は Low レベルになる．

⑤ CRB_7 のリセット

③に同じ．

以下④↔⑤を繰り返して1文字ずつプリントされる

⑥ $\overline{\text{FEED}}$ 信号 ON

$\overline{\text{DATA REQ}}$ ラインが Low レベルとなり，④のように CB2 は High レベル，CRB_7 は1となる．このとき文字データとして $80 以上の値（すなわち，$PB_7=1$）が出力されたならば，$\overline{\text{PRINT}}$ ラインは High レベルとなり，$\overline{\text{FEED}}$ ラインは Lowレベルとなる．すなわち，文字データはプリントされずに，1行だけ紙送りが行なわれる．

⑦ CRB_7 のリセット．

③に同じ．

∞ $\overline{\text{FEED END}}$ 信号

$\overline{\text{FEED END}}$ 信号は $\overline{\text{FEED}}$ 信号が Low レベルになって約 200msec 後に Low レベルになり，さらに 3msec 後に再び High レベルに戻る．このとき $\overline{\text{FEEDEND}}$ 信号の立ち上がりによって $\overline{\text{READY}}$ 信号は Low レベルとなって，次の行のプリントの待機状

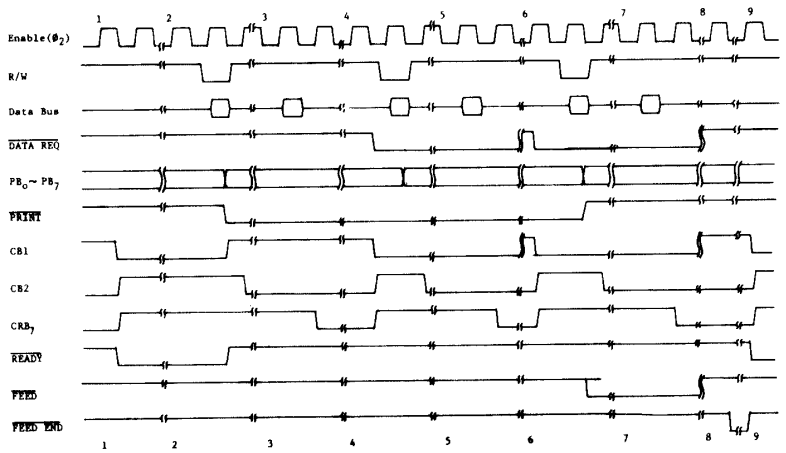

Fig. 10　Timing Chart

態となる．(①へ戻る)

3.3 PIAとドットプリンタの間のハードウェア

① PB_7ライン

$\overline{PRINT}$ 信号はPIA側でも負論理とする．(したがって文字コードの8ビット目，すなわち$PB_7=0$のときプリントさせるようにする)

$\overline{FEED}$信号は単安定マルチバイブレータによって発生させる．トリガパルスはPB_7信号の立ち上がりで発生させる．単安定マルチバイブレータの入力は定常状態ではHighレベルにプルアップしておく．$\overline{FEED}$信号の負のパルス幅は約140msec(≃0.7CexRex)に設定している．

② CB1ライン

CB1ラインはハンドシェークを行なう場合のプリンタからCPUへの信号ラインであり，$\overline{DATA\ REQ}$と$\overline{READY}$(受信準備完了の合図)のOR(負論理)をとったものがCB1に与えられる．CPUがこのラインの状態を知りたいときは，前述のように，CB1のアクティブエッヂによって立つフラグ(CRB_7)を見ることによって知る．ところが，このフラグはリセット信号によってもクリアされるので，フラグが立っているときにリセットがかかると，フラグの値とCB1のレベルとの間に矛盾が生じる．これを防ぐためには，リセットがかかった後もう一度CB1ラインの信号をHighレベルからLowレベルに入力しなおせば良い．そのため，リセットのかかった時点でプリンタからの$\overline{DATA\ REQ}$と$\overline{READY}$信号のOR(負論理)出力をマスクしてCB1をいったんHighレベルにし，リセットが解除されたときそれより少し遅れてマスクを解除すれば，プリンタからの出力が再びCB1に入力され，フラグ(CRB_7)は正しくセットされる．

3.4 ソフトウェア

インターフェースを制御してドットプリンタを動かすには，まずポートの初期設定をし，ついでプリントを行なうプログラムを用意し，これを実行することになる．

3.4.1 I/Oポートの初期設定

(ルーチンPORTI. スタート番地 $1F00)

PIAは，リセット状態ではすべての内部レジスタは0にリセットされる．この状態から*B*ポートについて次の様に機能の初期設定をする．

① Peripheral Output Registerに$80を書き込む．($\overline{PRINT}$信号OFF)

② DDRBに$FFを書き込む(*B*ポートを出力ポー

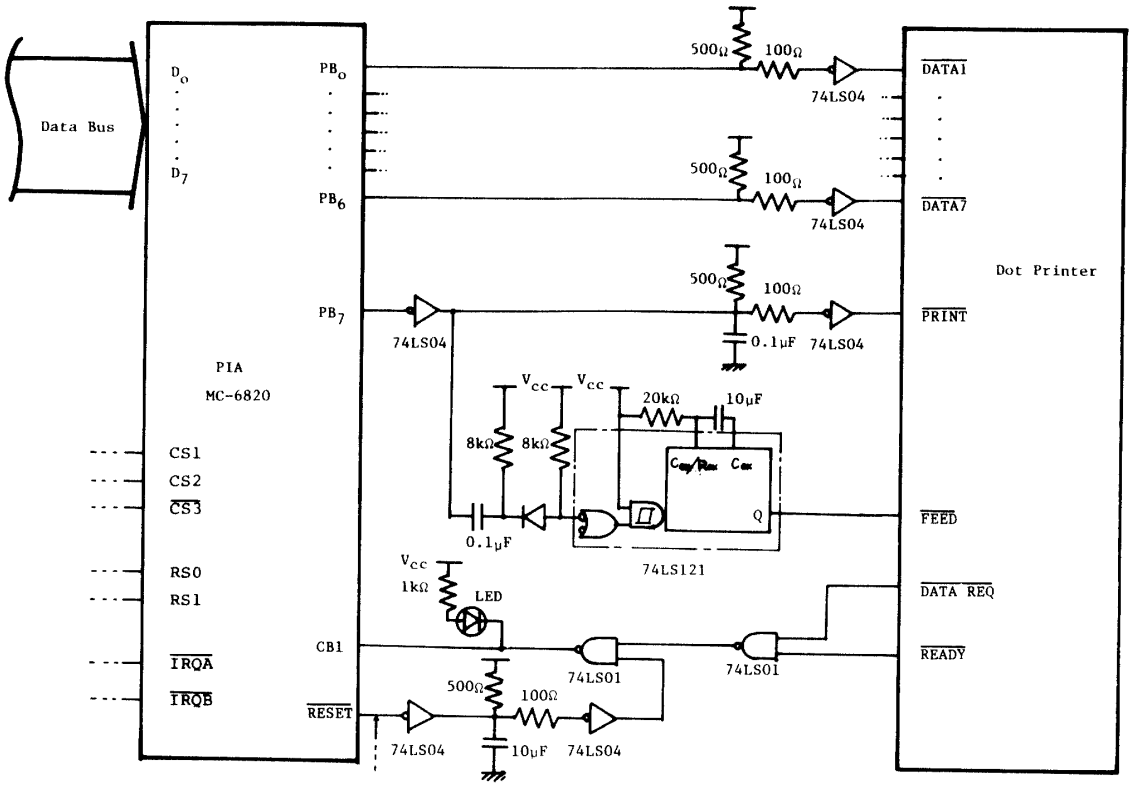

Fig. 11 Interface Circuit

トに設定)

③ CB 2 をOutput-Level-Handshake mode にする. ($CRB_5=1$, $CRB_4=0$, $CRB_3=0$)

④ CB1 を Interrupt Disable mode かつ Negative Edge mode にする. ($CRB_0=0$, $CRB_1=0$)

⑤ ①～④の全ての設定がなされた後 $CRB_2=1$ にセットして, Peripheral Output Register をアクセスする.

このとき, CRB の内容は$24となっている. 上記のフローチャートを図 12-a に示す.

3.4.2 JIS コードのプリント

(ルーチン PRJIS, スタート番地$1F29)

① $\overline{\text{READY}}$ 信号が ON (Low レベル)になっているかどうかを確認するために CRB_7 をチェックする. ($\overline{\text{READY}}$ 信号と $\overline{\text{DATA REQ}}$信号とが同時にONになることはない.)

② $\overline{\text{READY}}$ 信号 ON を確認した後 $\overline{\text{PRINT}}$ 信号を ON にする. ($\overline{\text{READY}}$ 信号が OFF のとき $\overline{\text{PRINT}}$ 信号を ON にすると保護回路が作動し, プリンタは動作しない) プリント信号としては $PB_7=0$ なら何でもかまわないが, ここでは $20 を B ポートラインから出力する. このデータは, まだプリントされない.

③ CRB_7 をリセットする. これは LDA 命令によって B ポートラインに出力されている値を CPU が読み込むことによって自動的になされる.

④ $\overline{\text{DATA REQ}}$ 信号が ON になったかをチェックする. これも①と同様に CRB_7 をチェックすることによりなされる.

⑤ $\overline{\text{DATA REQ}}$信号がONならば, メモリーにあらかじめストアしていた 1 行分の文字データの中から順序に従って 1 文字分だけ Peripheral Output Register に書き込む.

⑥ ③と同様にして, CRB_7 をリセットする.

⑦ $\overline{\text{FEED}}$ 信号がONになったかどうかをチェックする. (これは⑤で文字データとして$80以上の値が Peripheral Output Register に書き込まれたかどうかでわかる. ここでは $A0 を用いている.)
$\overline{\text{FEED}}$ 信号が OFF ならば, 次の文字を出力するために④以下を繰り返す.

⑧ 1 行に最大80文字を印字する. 80文字印字したら $\overline{\text{FEED}}$ 信号を出力する.

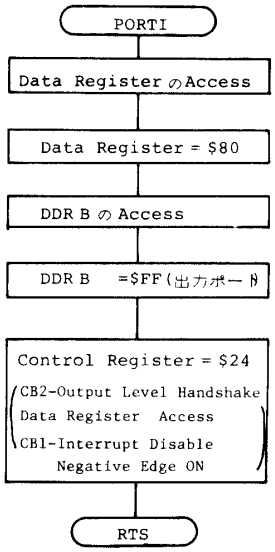

Fig. 12-a

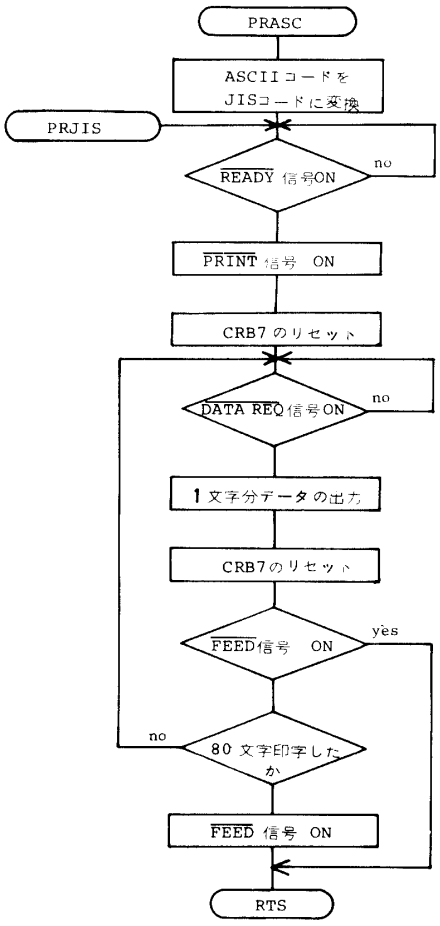

Fig. 12-b

⑨ $\overline{\text{FEED}}$信号が出力されたならば，次の行のプリントを行なうために①に戻る．

以上のフローチャートを図12-bに示す．

3.4.3 プログラム

ここで用いるドットプリンタはJISコードであるが，本システムのモニタ“DEMON”（またはTIM）はASCIIコードを文字コードとしているので，ASCIIコートをJISコードに変換する必要がある．以下サブルーチンの簡単な説明を行なう．

◦PRASC（スタート番地$1F1A，以下，スタート番地は省く）

ASCIIコードをJISコードに変換してPRJISルーチンへ進む．特殊な3つのキャラクタを除けば，ASCIIコードの7ビット目をマスクすれば良い．

◦STOC（$1FAE）

1バイトの文字コードを$00E2番地（コラムポインタと呼ぶ）の内容でインデクス修飾された番地にストアする．（$E2)* は1回のストアごとにインクリメントされ，もし（$E2)=$(80)_{10}$ になったならば，1行80文字のプリントを行ない（$E2）をクリアする．

◦STOB（$1FA4）

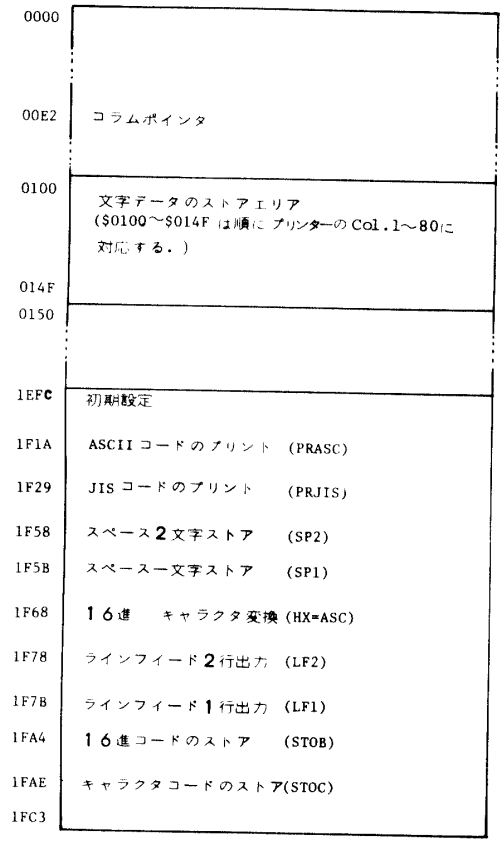

Fig. 13 Address Map

脚注 ＊（$E2）は $E2 番地の内容を意味する．

1バイトの16進コードを2つの文字コードに変換して，それぞれのコードをSTOCルーチンによりストアする．

```
1EFC  A0 00       LDY #$00
1EFE  84 E2       STY  $E2
1F00  A9 24       LDA #$24
1F02  8D 035C     STA  $5C03
1F05  A9 80       LDA #$80
1F07  8D 025C     STA  $5C02
1F0A  A9 20       LDA #$20
1F0C  8D 035C     STA  $5C03
1F0F  A9 FF       LDA #$FF
1F11  8D 025C     STA  $5C02
1F14  A9 24       LDA #$24
1F16  8D 035C     STA  $5C03
1F19  60          RTS


1F1A  A2 00       LDX #$00
1F1C  BD 0001     LDA  $0100,X
1F1F  29 BF       AND #$BF
1F21  9D 0001     STA  $0100,X
1F24  E8          INX
1F25  E0 50       CPX #$50
1F27  D0 F3       BNE  $1F1C

1F29  AD 035C     LDA  $5C03
1F2C  10 FB       BPL  $1F29

1F2E  A9 20       LDA #$20
1F30  8D 025C     STA  $5C02
1F33  AD 025C     LDA  $5C02
1F36  A2 AF       LDX #$AF
1F38  AD 035C     LDA  $5C03
1F3B  10 FB       BPL  $1F38

1F3D  BD 5100     LDA  $0051,X
1F40  8D 025C     STA  $5C02
1F43  AD 025C     LDA  $5C02
1F46  C9 80       CMP #$80
1F48  10 08       BPL  $1F52

1F4A  E8          INX
1F4B  30 EB       BMI  $1F38

1F4D  A9 A0       LDA #$A0
1F4F  8D 025C     STA  $5C02
1F52  AD 035C     LDA  $5C03
1F55  10 FB       BPL  $1F52

1F57  60          RTS


1F58  20 5B1F     JSR  $1F5B
1F5B  48          PHA
1F5C  8A          TXA
1F5D  48          PHA
1F5E  A9 20       LDA #$20
1F60  20 AE1F     JSR  $1FAE
1F63  68          PLA
1F64  AA          TAX
1F65  68          PLA
1F66  60          RTS
```

Fig. 14-a Progrm List for Dot Printer

```
1F67  EA          NOP
1F68  48          PHA
1F69  29 0F       AND #$0F
1F6B  20 5873     JSR  $7358
1F6E  AA          TAX
1F6F  68          PLA
1F70  4A          LSR A
1F71  4A          LSR A
1F72  4A          LSR A
1F73  4A          LSR A
1F74  20 5873     JSR  $7358
1F77  60          RTS


1F78  20 7B1F     JSR  $1F7B
1F7B  48          PHA
1F7C  A9 A0       LDA #$A0
1F7E  8D 0001     STA  $0100
1F81  20 291F     JSR  $1F29
1F84  68          PLA
1F85  60          RTS


1F86  20 7B1F     JSR  $1F7B
1F89  A9 2A       LDA #$2A
1F8B  A2 AF       LDX #$AF
1F8D  9D 5100     STA  $0051,X
1F90  E8          INX
1F91  30 FA       BMI  $1F8D

1F93  20 291F     JSR  $1F29
1F96  20 7B1F     JSR  $1F7B
1F99  A5 E0       LDA  $E0
1F9B  85 F6       STA  $F6
1F9D  A5 E1       LDA  $E1
1F9F  85 F7       STA  $F7
1FA1  4C 5C71     JMP  $715C


1FA4  20 681F     JSR  $1F68
1FA7  86 DF       STX  $DF
1FA9  20 AE1F     JSR  $1FAE
1FAC  A5 DF       LDA  $DF
1FAE  A6 E2       LDX  $E2
1FB0  9D 0001     STA  $0100,X
1FB3  E8          INX
1FB4  86 E2       STX  $E2
1FB6  E0 50       CPX #$50
1FB8  D0 07       BNE  $1FC1

1FBA  20 1A1F     JSR  $1F1A
1FBD  A2 00       LDX #$00
1FBF  86 E2       STX  $E2
1FC1  A2 00       LDX #$00
1FC3  60          RTS
```

Fig. 14-b Program List for Dot Printer

◦SP2, SP 1（$1 F58, $1F 5B）

SP2はスペース２文字，SP1はスペース１文字をメモリーにストアする．

◦LF 2, LF 1（$1 F78, $1 F7B）

LF2はキャリッジリターン，ラインフィードを２回，LF1は１回実行する．

◦HX＝ASC（$1F68）

ACCの値の上位４ビット，下位４ビットをそれぞれ16進数として表示するためにコード変換を行なう．上位４ビットに対応する文字コードは ACCに下位４ビットのそれは，IXにストアする．

以上のアドレスマップを図13に示す．

最終のプログラムを図14に示す．

4. むすび

ドットプリンタをマイクロコンピュータシステムに接続するために，コンピュータと外部装置とのデータのやり取りの仕組を述べ，ポピュラーなLSI PIA 6820を用いたインタフェースの設計を行なった．このような議論はマイクロコンピュータをいろんな分野に応用するときに生ずる外部装置との接続問題に有効である．

次のシステム拡張として，カセットテープレコーダ，フロッピーディスク，キャラクタディスプレー等を考えている

参 考 文 献

(1) MCS6500 Microcomputer Family Hardware Manual, MOS TECHNOLOGY, INC. (1976)

(2) 松本吉彦：わかるマイクロコンピュータ〔第１回〕～〔第７回〕，トランジスタ技術 1976. 8～1977. 2

(3) 八田，高浪，井上：MCS6502 マイクロコンピュータの逆アセンブラー，山口大学工学部研究報告，**29**, 113 (1978)

(昭和53年４月12日　受理)