マイクロコンピュータ 25－2
（1982. 12. 2）

MC68000開発支援システム

五十嵐智　石丸知之　細井雄次　鵜沢繁行
（東京農工大学　工学部　数理情報工学科）

1. はじめに

著者等のグループでは，モトローラ社の16ビットCPU（MC68000）を用いたシステム開発を行なっている。現在，ソフトウェアの開発にあたっては，大型計算機ACOS 600 TSSシステム上にインプリメントしたクロスアセンブラ（文献1）とシミュレータ（文献2）を利用している。しかし，システムとしての開発を行なうには，ハードウェアも含めた総合的な開発支援システムが必要となる。

従来，開発支援のために以下のようなツールが利用されてきた。

- プログラム開発を目的とした開発機およびセルフ／クロスのソフトウェア
- バスの状態の観測やヒストリの記録，CPUのRUN／STOP等の制御を目的としたハードウェアツール
- システム全体の実行状態をシミュレートするインサーキットエミュレータ等のツール

それぞれの形式の開発ツールは主として8ビットCPU用を中心に多数市販され，その機能，開発過程における役割り等も良く周知されていると思う。

16ビットCPUを用いた応用システムの場合には，8ビットCPUのものよりもその規模・機能の面で相当複雑さが増すと考えられる。そして，システムが複雑であればあるほど，ハードウェアおよびソフトウェアの適切なモジュール化およびモジュール単位での完全なデバッグが重要な意味を持ってくる。

著者等は，こうした要請に対応するものとして，68000CPUを用いて以下の機能を有する開発用計算機システムを製作した。

（1）デバッグの対象となる基板（以下ではユーザ基板と呼ぶ）に対して，動作の観測やエラーの検出等を行なうためのデバッグ専用のハードウェア機構とこれを支援するソフトウェアを提供する。ユーザ基板には，表1に記したバス構造を指定しており，基板はKEL製5960－100シリーズを標準として定めた。

（2）アセンブラ言語によるプログラム開発ならびにそのデバッグ機能を提供する。そのために，テキストエディタ，アセンブラ，デバッガを備えている。

（1），（2）の機能を使って，たとえば，ユーザは試作したペリフェラル基板を本機上の専用スロット（ユーザコネクタ）に差し込み，これを制御するプログラムを本機上で開発して実行させながら，ハードウェアとソフトウェア両方向から動作をテストすることができる。本機は，そのようなハードウェアとソフトウェアの接点にあたるような部分の開発に特に有効である。また，通常のプログラム開発においても，我々の利用している大型計算機上のクロスソフトウェアよりもはるかに効率が良い。このように本機は，ハードとソフトの両面にわたって開発を総合的に支援する開発システムである。図1に本機の外観を示す。

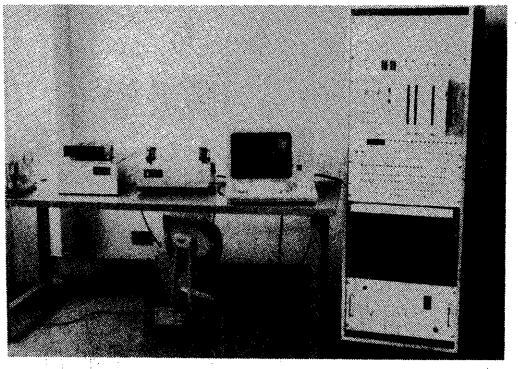

図1. システム外観

2. システム構成

本機のハードウェア構成を図2に示す。本機のハードウェアは通常の計算機システムとしての部分と，これに付属した各種の開発・デバッグ専用ハードウェア機構よりなる。

前者は68000CPUを中心にコンソールパネル(ミニコン等に備わっているものと同等の機能を持つ)，EPROM 64KB，SRAM 16KB，DRAM 256KB，コンソール端末，プリンタ，PTP，PTR，モデムインターフェース等を配したものである。後者については3.で述べる。

図2はソフトウェア構成である。デバッガはモニタとしての機能を持ち，入出力やシステムソフトの管理も行なう。システムソフトとしてはエディタ，アセンブラ(文献3)の他にROMライタ等のデバイスハンドラルーチン群を備えている。

表1. ピン接続

GND	02	01	GND
+5V	04	03	+5V
D01	06	05	D00
D03	08	07	D02
D05	10	09	D04
D07	12	11	D06
D09	14	13	D08
D11	16	15	D10
D13	18	17	D12
D15	20	19	D14
level 7	22	21	level 6
level 5	24	23	level 4
level 3	26	25	level 2
level 1	28	27	
	30	29	
GND	32	31	GND
VMA	34	33	VPA
	36	35	E
R/W	38	37	AS
LDS	40	39	UDS
	42	41	DTACK
+5V	44	43	+5V
GND	46	45	GND
	48	47	CLOCK
GND	50	49	GND
A01	52	51	
A03	54	53	A02
A05	56	55	A04
A07	58	57	A06
A09	60	59	A08
A11	62	61	A10
A13	64	63	A12
A15	66	65	A14
A17	68	67	A16
A19	70	69	A18
A21	72	71	A20
A23	74	73	A22
FC1	76	75	FC0
	78	77	FC2
	80	79	
	82	81	
GND	84	83	GND
	86	85	BERR
HALT-OUT	88	87	HALT-IN
RESET-OUT	90	89	RESET-IN
	92	91	
	94	93	
BG	96	95	BR
	98	97	BGACK
GND	100	99	GND

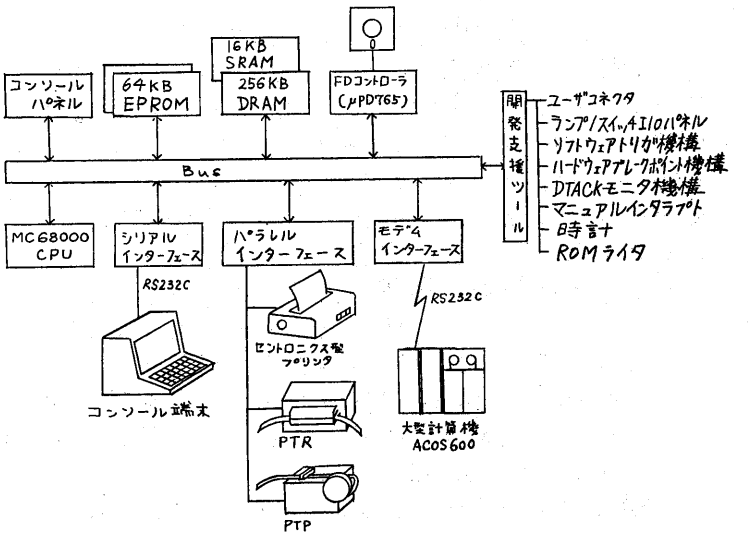

図2. ハードウェア構成

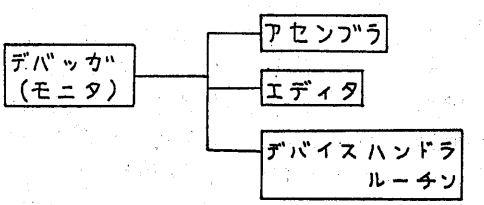

図3. ソフトウェア構成

3. 開発・デバッグ専用ツール

本機では，1(1)に対応するハードウェア機構として以下のものを備えている。

3.1 ユーザコネクタ

ユーザコネクタは，本機の持つ資源をユーザに開放するためのものである。これにより，68000 CPUを用いた応用システムのために製作されたメモリ基板やペリフェラル基板等のハードウェアモジュール（ユーザ基板）を自由かつ安全に本機に接続できる。つまり，ユーザ基板で使用している空間や割込みに関して何ら制限を与えず，ユーザは本機の持つハードウェア・ソフトウェアを用いて，本来ユーザ基板のホストとなるべきシステムをシミュレートしながら，デバッグ・調整ができる。図4はユーザコネクタの外観である。

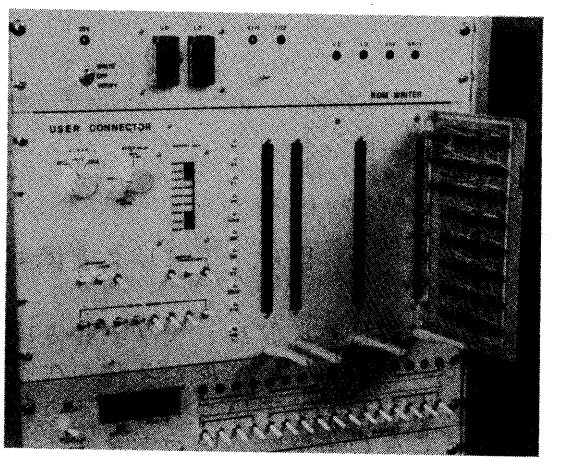

図4．ユーザコネクタ外観

図にあるように，ユーザコネクタにはユーザ基板を接続するためのコネクタが取り付けられていて，表1に記したバス接続端子を持っている。

一方，こうしたシステム資源の開放に伴ない，システムの保護およびユーザ基板とシステム間での空間の取合いが問題となる。ユーザコネクタには，このための機構としてアドレス変換機構，割込みレベル・ベクトル変換機構，バス衝突検出機構等を備えている。

(1) アドレス変換機構

本機では0～2MBの領域をシステム領域としている。したがって，この領域のアドレスを使用しているユーザ基板は直接には本機に接続できない。この問題を解決するために，ユーザコネクタではCPUの送出するアドレスを変換してユーザ基板に供給している。この変換機構により，ユーザ基板で使用している領域にかかわらず，システム側から見るとユーザ基板は8～16 MBの領域に写像される。図5は変換機構の回路である。図に示すように，システム側のアドレスバスA23, A22, A21に対応させたスイッチ S_3, S_2, S_1 を設け，そのスイッチの値をA23, A22, A21に加えることによりアドレス変換を行なっている。つまり，アドレスとスイッチの関係は次式のようになる。

$$\begin{array}{r} A23\ \ A22\ \ A21\ \ A20\ \cdots\ A1 \\ +)\ \ S3\ \ \ S2\ \ \ S1 \\ \hline A23'\ A22'\ A21'\ A20\ \cdots\ A1 \end{array}$$

（桁上りは無視）

この機構により，システムは2MB単位でユーザ基板の使用している空間を自分の空間内の領域のものとして参照できる。変換例を図6に示す。

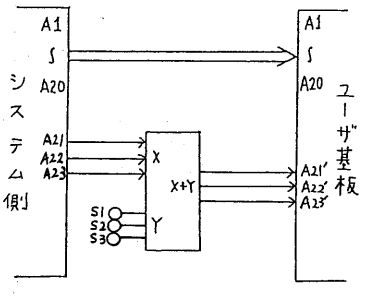

図5．アドレス変換回路

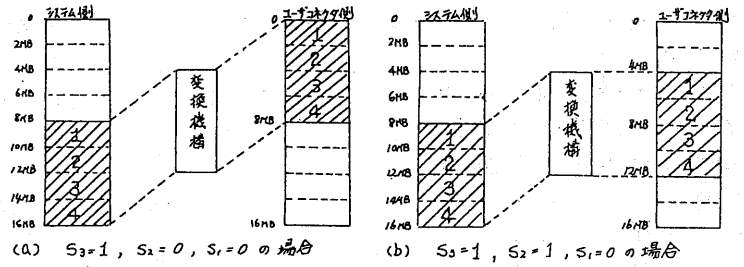

図6. アドレス変換例

（2）割込みレベル・ベクトル変換機構

68000CPUは，7レベル192種のベクトル割込み機能を持つ。ユーザ基板がこの割込み機能を使っている場合，その割込みをシステム自体が用いている割込みと競合させないように，かつ完全にシステム側の開発支援ソフトの制御下に置かなければならない。

割込みレベル変換機構は，ユーザ基板からの割込み要求レベルをすべてレベル1（最低位の優先レベル）に変換する機構である。この機構により，システム側の割込み処理を優先させることができる。割込みベクトル変換機構は，ユーザ基板から送出されるベクトル番号をすべて固定のベクトル番号（FE_{16}）に変換する機構である。これは，ユーザ基板で使用しているベクトル番号とシステム側で使用しているベクトル番号との重複を避けるためである。さらに，これらの機構はユーザ基板のレベルとベクトルが正しく送出されているかどうかをチェックする機能も備えている。そのためにレベルチェック用とベクトルチェック用の外付けスイッチ（それぞれ3ビットと8ビット）が設けてあり，あらかじめユーザはそのスイッチにユーザ基板で使用しているレベルとベクトル番号を設定する必要がある。この機構の動作フローを図7に示す。

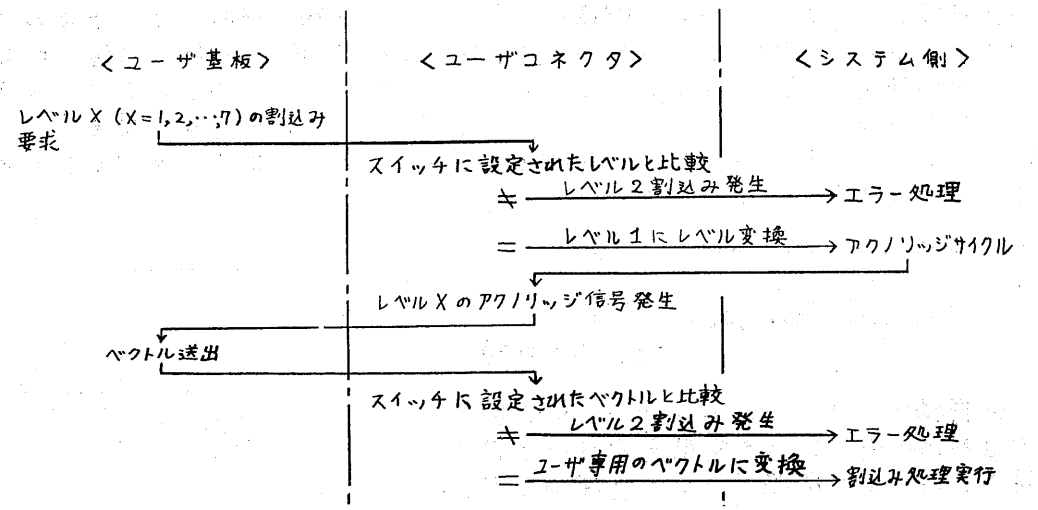

図7. 割込みレベル・ベクトル変換機構の動作フロー

（3）バス衝突検出機構

ユーザコネクタにはシステムバス（アドレスバス，データバス，制御信号）を供給しているが，その場合，相互の電気的な保護が問題となる。ユーザ基板の誤配線等によりユーザ基板のバスとシステムバスが衝突を起こすと，ユーザ基板のみならずシステム側の回路も破壊される恐れがある。

本機では，この種のエラーの影響を最小限に防ぎ止めるために，システムバスとユーザ基板のバスとの間はバッファゲートを介している。単方向の信号線に関する限りこのエラーは少ないが，双方向の信号線であるデータバスに関しては比較的多い。そのために，データバスに関しては衝突の検出を行ない，検出したときには警報を発する機構を設けた。衝突の検出は，バッファゲートの電源電流の変化を検出することにより行なっている。この回路を図8に示す。以下，回路の動作原理を述べる。

IC1（74LS126）は，システム側からユーザ基板へのバッファゲートで，ライトサイクルのときデータスルーとなる。ライトサイクルにもかかわらず，ユーザ基板がデータを出力しようとしたときに衝突を起こす。衝突を起こすとIC1のVccおよびGNDの電流が変化する（注）。衝突を起こしていないとき，IC1のVcc（GND）には23mA（26mA）の電流が流れている。そのときA点（C点）の電圧は6.7V（−6.2V）でB点（D点）の電圧を2.1V（−6.5V）に保っている。したがって，IC2（IC3）の出力は"L"である。ただし，Vcc（GND）はTr1（Tr2）によって5V（0V）が供給されている。衝突を起こすと，Vcc へ（GNDから）流れる電流は50mA（70mA）となりB点（D点）の電圧は5V（3.5V）となる。その結果，B点（D点）の電圧が−0.65V（2.1V）になりIC2（IC3）の出力は"H"となる。したがって，この信号により衝突の検出ができる。

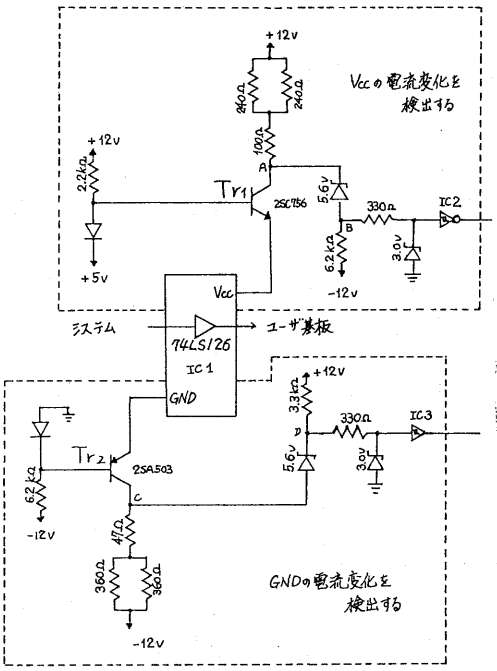

図8. 衝突検出回路

以上の（1），（2），（3）はシステムの保護のための機構であるが，その他に（4），（5）の機構を備え付けた。

（4）DTACKディレイ機構

DTACK（Data Transfer Acknowledge）信号は，CPUのメモリアクセス時に，アクセスされたデバイスがCPUに返送しなければならない同期信号である。CPUはDTACKが返送されてから1.5クロック後にアクセスを終了する（8MHz動作の場合，CPUのアクセス時間は312.5nsec）。したがって，アクセス時間の大きなメモリなどはDTACK信号の返送が早過ぎると，正常のリード／ライト動作ができ

（注）この場合，システム側が"H"（"L"）を出力したときユーザ基板が"L"（"H"）を出力すると，Vcc（GND）に規格値以上の電流が流れる。

きない。
本機構は，ユーザ基板のDTACK返送タイミングが不適切で正常なアクセスができない場合，その遅れ時間を調整し，適切な補正時間を求めるためのものである。遅れ時間は0～700msecで可変である。

（5）同期型クロック変換機構

68000CPUのバージョンには2，4，6，8，10MHzのものがある。開発機という性格上，その動作周波数はターゲットシステムのものと合致させる必要がある。本機構により，動作周波数を任意に選択できる。また，同期型としてあるため，プログラム実行中でも変更することができる。

3.2 スイッチ／ランプI/Oパネル

コンソールパネルとは別に，入力として16ビットのスイッチ，出力として16ビットのLEDと16進4桁の7セグメントLEDを備えたパネルである。このパネルにより，プログラムの任意の時点でのデータバスやCPUのレジスタ類を表示したり，任意のデータを読み込んだりすることができる。

3.3 ソフトウェアトリガ機構

通常CPUのバスは同期がとりにくく，プログラム実行中に任意の命令に対する信号波形をシンクロスコープで観測することは非常に難しい。本機構は，シンクロスコープに対する同期信号をプログラム内で定義して，発生させることを目的としている。本機構を使用するには，観測すべき波形に対応した命令の直前に，駆動命令を挿入し，その命令群をループさせる。そして，発生する同期信号でシンクロスコープの同期をとる。

3.4 ハードウェアブレークポイント機構

68000CPUでブレークポイント機構を実現する方法としては，以下のものが考えられる。

（1）トレースモードの利用
（2）命令の置換え
（3）ハードウェアによるアドレスバスの監視

(1)の方法では一命令実行毎にCPUは例外処理を行なう。このため，プログラムの実行速度は数十分の一に低下する。このことは，割込み等の実時間で動作するプログラムのデバッグには支障を来たす。(2)の方法の欠点はROMベースのプログラムは命令を書き換えることができないので，ブレークポイントを設定できないという点である。以上の理由から，本機構では(3)の方法を採用した。

実際の回路では，ユーザの指定したアドレスをレジスタにラッチし，それとアドレスバスの内容を比較し，一致した場合にはレベル7の割込みを発生する。バスエラーを用いなかったのは，命令の再実行ができないためである。

本機構の特徴は，第一に，ファンクションコードも比較の対象としているために，68000CPUの持つ四種の空間（スーパーバイザ／ユーザ，プログラム／データ）に対して独立に設定できること，第二に，ブレーク発生時点でのCPUの状態（リード／ライトの区別，上位バイト／下位バイトアクセスの区別，ファンクションコード）を記録しているということ，第三に，比較部を二組設けてあるため，

空間の領域に対しても設定でき，プロテクション用としても利用できるという点である。

一方，本機構の問題としては次の二点である。

（1）インストラクションに対して設定された場合には，プリフェッチの段階でブレークがかかる。

（2）フェッチされるオペランドに対して設定された場合には，命令の実行後にブレークがかかる。

これらの点は，68000CPU のアーキテクチャにかかわることである。なお，68010 ではバスエラーからの完全な復帰が可能であるということなので（文献4），（2）の問題は解決されるであろうが，（1）の問題はさらに複雑になるであろう。

3.5 DTACKモニタ機構

68000CPUのメモリアクセスは非同期方式であり，DTACK の返送をもってそのアクセスを完了する。このため，メモリの実装されていない領域をアクセスしたような場合にはDTACK が返送されないので"CPUは停止してしまうことになる。本機構はこのような異常事態を検出する。この機構はコンデンサの充電時間を利用して時間を測り，CPUのメモリアクセスが始まってから一定時間（700msec）経過後にもDTACK が返送されないとき，割込みフラグ（レベル7）をセットした上で疑似的なDTACKを返し，割込み処理ルーチンでこのエラー処理を行なう。

3.6 マニュアルインタラプト

プログラムの暴走や無限ループに入ってしまったときの救済として設けた。このスイッチがONにされるとレベル7の割込みが発生し，強制的にデバッガがホットスタートする。

3.7 その他の機構

以上述べた各種機構の他に，タイマ割込み機能（設定時間は0～約6.5秒）と2716/2732用のROMライタを持つ。

4. デバッガ

前章で述べた各種ツールを統合して開発効率を高めるためのデバッガを作成した。本デバッガは，ユーザプログラムの実行および中断の制御ならびに3.で述べた開発・デバッグ専用ツールの管理を行なう。また，モニタとしての機能も持ち，ユーティリティプログラム等の実行制御を行なう。

デバッガとユーザプログラム間の制御の移行方法を図9に示す。デバッガからユーザプログラムへはG(GO)指令による。一方，ユーザプログラムからデバッガへの復帰は，

(1) TRAP#13命令
(2) ブレークポイント
(3) マニュアルインタラプト
(4) ユーザ未定義の例外の発生

の，いずれかの場合による。

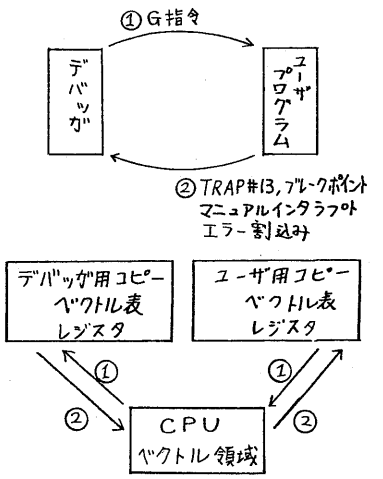

図9. 制御の移行

本デバッガは，レジスタ類のコピーエリアと共にベクトルテーブル用のコピーも持つ。そして，システムとユーザ間で制御の移行があるたびに，デバッガ用のものとユーザ用のものとを交換している。ユーザ用のテーブルには，あらかじめデバッガに復帰させるための処理ルーチンをデフォルト値として設定してあるため，ユーザが新たに定義していない例外が発生した場合（例えばバスエラー，アドレスエラー等の内部的に発生する例外）には，すべてデバッガに復帰する。

本デバッガは，ユーザプログラムの実行・制御の他，メモリ・レジスタ類のアクセス，入出力装置の起動に関する機能を持つ。指令の一般形式を以下に示す。

*C {アドレス}/{値}:{値}↵ または "=" または ">" または "<"}
*C {アドレス1}-{アドレス2}{↵ または "="}
*C↵

〔*は入力促進記号，Cは指令を表わす英字1文字，アドレス（レジスタ番号）は16進，下線部はユーザ入力を表わす。↵はCRキー，"="は同一指令の再実行，">"はアドレス部をインクリメント，"<"はデクリメントして同一指令の実行を行なう。〕

表2に指令の一覧を，図10に指令の実行例を示す。

表2. デバッガ指令一覧

指令	機能
D (data)	データレジスタの表示・書換え
A (address)	アドレスレジスタ 〃
P (program)	プログラムカウンタ 〃
S (status)	ステータスレジスタ 〃
R (register)	全レジスタの表示
B (byte)	メモリの表示・書換え（バイト単位）
W (word)	〃 （ワード単位）

L	(long)	〃 (ロングワード単位)
M	(memory)	メモリの表示
T	(trace)	トレース範囲の表示・書換え
C	(copy)	プリンタの起動
N		PTR の起動 (アスキー形式)
Y		〃 (バイナリー形式)
Z		PTP の起動 (バイナリー形式)
G	(go)	プログラムの実行
X		ROM ライタの起動
Q		アセンブラの起動
H		ブレークポイントの設定
I	(initialize)	ベクトルテーブル・レジスタ類のイニシャライズ

<1> レジスタ・メモリのアクセスに関する指令実行例

```
*D0/77805288:FE)                    データレジスタの表示・書換え
 D1/00000000:FECC)
 D2/00000000:60FE=
 D2/000060FE:(
 D1/0000FECC:(
 D0/000000FE:↙
*A0/00000000:2000)                  アドレスレジスタの表示・書換え
 A1/00000000:↙
*P/000000:1C0000=                   プログラムカウンタの表示・書換え
 /1C0000:↙
*S/2000:2700                        ステータスレジスタの表示・書換え
*R↙                                 全レジスタの表示
   (0)----- (1)----- (2)----- (3)----- (4)----- (5)----- (6)----- (7)-----
DR 000000FE 0000FECC 000060FE 00000000 00000000 00000000 00000000 00000000
AR 00002000 00000000 00000000 00000000 00000000 00000000 00000000 00001800
        PC    1C0000          SR  2700                  USP(A8)  00001400
*B400/9D:00)                        メモリの表示・書換え (バイト単位)
  000401/ED:02)
  000402/00:FE=
  000402/FE:(
  000401/02:↙
*W400/0002:4E71=                    メモリの表示・書換え (ワード単位)
  000400/4E71:↙
*M400-42F↙                          メモリダンプ
        0    2    4    6    8    A    C    E
000400  4E71 FECD 290D 50CD BC3D 423D 7C3D 343D
000410  26CD 1DCD 2DCD BA5D 2D3D 143D 9D3D 7C3D
000420  23CD 11CD 93CD B3CD 6B3D 143D 8B3D C13D
*M2000-202F↙                        メモリダンプ (XXはメモリを実装していないことを表わす)
        0    2    4    6    8    A    C    E
002000  XX08 XX00 XXXX XXXX XXFF XXFF XX00 XX0D
002010  FC20 XXXX XXXX XXXX 0FXX XXXX XXXX XXXX
002020  XXE0 XXXX XXXX XXXX FFFF XXXX XXXX XXXX
```

<2> プログラムの実行に関する指令実行例

```
*T/000000-000000:1C0000-1D0000↙     トレース範囲の指定
*S/2700:A000↙                       トレースビットをセットし，割込みマスクを0にする
*R↙
   (0)----- (1)----- (2)----- (3)----- (4)----- (5)----- (6)----- (7)-----
DR 000000FE 0000FECC 000060FE 00000000 00000000 00000000 00000000 00000000
AR 00002000 00000000 00000000 00000000 00000000 00000000 00000000 00001800
        PC    1C0000          SR  A200                  USP(A8)  00001400
*G↙   実行 (シングルステップ)
    Address  1C0000    Instruction  13FC
*G↙
    Address  1C0008    Instruction  0000
*G↙
    Address  1C000C    Instruction  0800
```

図10. デバッグ指令実行例

5．使用実績と問題点

本機は現在68000CPUを用いた各種システム（IEEE488通信線，バブルメモリシステム等）の開発に利用されている。その中ですでに開発を終えたものに関して，本機を使用した経過，結果および明らかになった問題を以下に記す。

（1）DRAMボードの製作

DRAMは，各種信号のタイミング制御が複雑でノイズにも影響されやすいため，その製作には多くの困難が伴なう。そのため，信号波形を正確に観測する必要がある。これにはトリガ機構が有効であった。またI/Oパネルにより簡単なメモリテストが視覚的にでき，有用であった。

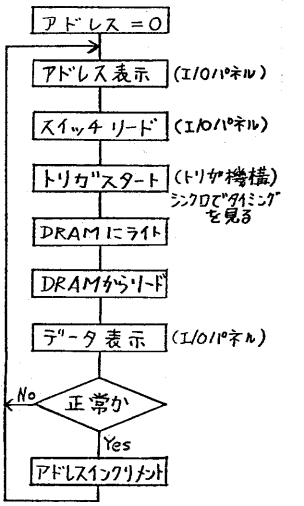

図11. DRAMのテスト

（2）LBP10コントローラ（文献5）

この開発では，本機の有効性と共に問題点も指摘された。第一に，本機ではバスマスタとなる基板のデバッグを支援していないことである。このため，止むを得ずユーザのCPUボードは直接本機のものと差し換えてテストした。第二に，DTACKモニタに関することである。LBP10のフレームメモリにはDRAMを用いたが，スキャン時のアクセスとリフレッシュが重なる場合，CPUは70μsec以上DTACKが返送されないこととなり，DTACKモニタが作動してしまった。こうした特殊なメモリのデバッグに際しては，DTACKモニタの設定時間をスイッチ等で可変にすることが望ましい。

この他の問題としては，ファイル管理等を行なう強力なOSがないことである。このため，大規模なソフトウェアを作成するには問題がある。現在，この作業を進めており，今後はFDOS下での使用を目指している。

6．おわりに

本機の製作の一部は，56年度卒業研究として行なわれ，設計・製作は，著者等の他に小平和正，松浦生明によって行なわれた。

＜参考文献＞

（1）高田他「MC68000のクロスインターシックアセンブラの開発」第23回情処全大 1C-6
（2）岩崎他「MC68000のシミュレータの開発とシステム記述言語の比較」同上 1C-5
（3）石丸他「言語C記述のハンドコンパイルにより作成したMC68000のレジデントアセンブラとその生産性の考察」第24回情処全大 6L-1
（4）本田「MC68000ファミリーの上位機種について」マイクロコンピュータ研究会資料22-3(1982)
（5）池田他「レーザビームプリンタのインテリジェントインターフェース/コントロールの試作」第24回情処全大 3G-10